Journal of Computer Science

Original Research Paper

Randomness Analysis on Lightweight Block Cipher, PRESENT

[1]Isma Norshahila Binti Mohammad Shah and [2]Eddie Shahril Bin Ismail

[1]Department of Cryptography Development, CyberSecurity Malaysia, Menara Cyber Axis, 63000 Cyberjaya, Malaysia
[2]Department of Mathematical Sciences, Faculty of Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia

Article history
Received: 24-09-2020
Revised: 23-11-2020
Accepted: 26-11-2020

Corresponding Author:
Isma Norshahila Binti Mohammad Shah
Department of Cryptography Development, CyberSecurity Malaysia, Menara Cyber Axis, 63000 Cyberjaya, Malaysia
Email: isma@cybersecurity.my

Abstract: Lightweight cryptography is an area of current research conducted by academicians and cryptographic experts to ensure the security of data in limited-resource devices such as RFID tags, medical and health care devices and sensor networks. One of the lightweight algorithms built is the PRESENT algorithm. To this day, PRESENT has been a reference for lightweight block cipher algorithms and is incorporated into Lightweight Cryptography Standard ISO/IEC 29192-2. The capacity to act as a random number generator is one of the key requirements when designing an algorithm. Thus, this study aims to examine the capabilities of the PRESENT algorithm as a random number generator. By using the NIST Statistical Test Suite, a randomness analysis is performed on the PRESENT algorithm. A total of six data categories i.e., Strict Key Avalanche, Strict Plaintext Avalanche, High-Density Key, Low- Density Key, Low-Density Plaintext and High-Density Plaintext were applied to generate 100 input sequences for each algorithm. From the analysis, the outputs generated from the PRESENT algorithm are essentially non-random based on the 1% significance level.

Keywords: PRESENT Algorithm, Randomness Analysis, NIST Statistical Test Suite, Lightweight Block Cipher

Introduction

Lightweight cryptography is one of the hot research topics in cryptography. Its main applications include RFID tags, medical and health care devices and sensor networks. Lightweight cryptography is generally divided into four categories, namely lightweight block cipher, lightweight hash function, lightweight message authentication codes and lightweight stream cipher (McKay *et al*., 2016). A lightweight block cipher is a block cipher requiring less computing power. It is designed to support devices with limited resources, e.g., RFID tags and sensor networks. Some existing series of lightweight block ciphers include DESL (Leander *et al*., 2007), KATAN and KTANTAN (De Canniere *et al*., 2009), LBlock (Wu and Zhang, 2011), PRESENT (Bogdanov *et al*., 2007) as well as SIMON and SPECK (Beaulieu *et al*., 2015).

Ultra-lightweight block cipher PRESENT which was introduced by (Bogdanov *et al*., 2007) works in a 64-bit plaintext block that utilizes two 80-bit and 128-bit key sizes. Its 80-bit version is dedicated for hardware implementation. To date, PRESENT is the benchmark for lightweight symmetric ciphering and it is included in the ISOIEC specification (ISO 29192-2:2012(E), 2012). PRESENT is a pioneer of the development of lightweight block ciphers and used together with AES (Pub, 2001) serving as the standard for new proposals.

Several attacks have been performed on the PRESENT algorithm in order to test its effectiveness against various cryptanalysis attacks. These attacks include side-channel attacks (Renauld and Standaert, 2009), side-channel cube attacks (Yang *et al*., 2009) and a related-key attack on the 17 rounds of PRESENT (Özen *et al*., 2009). Certain attacks such as the enhanced differential fault analysis has been documented by (Jeong *et al*., 2013); this attack retrieves the key by causing two or three 2-byte random faults. According to (Jeong *et al*., 2012), full-round biclique cryptanalysis is slightly better than exhaustive search. A truncated differential attack on the reduced 26-round cipher has

been investigated by (Blondeau and Nyberg, 2014). Among all the analyses carried out in evaluating the strength of the lightweight block cipher PRESENT, to the best of our knowledge, the randomness analysis has not been carried out on the PRESENT algorithm so far. Therefore, we wish to address this problem in this study.

This study is structured in the following manner. The second section presents some previous works related to randomness analysis performed on cryptographic algorithms. The third section gives a brief description of the PRESENT algorithm. The methodology used to perform randomness analysis is explained in the fourth section. Results and discussion are presented in the fifth section. Finally, the current work is concluded in the sixth section.

Related Work

Randomness plays an important role in many areas of cryptography (Marton and Suciu, 2015). Cryptographic implementations are based on random numbers with special features (Demirhan and Bitirim, 2016). One of the significant criteria for developing an encryption algorithm is its capability as random number generator (Hathaway, 2003). The Pseudorandom Number Generator (PRNG) statistical test suite can be used to evaluate the randomness of outputs from an algorithm by applying a series of statistical tests on the outputs.

After evaluating several random test suites that may be available i.e., Diehard (Marsaglia, 2008), TestUI (L'Ecuyer and Simard, 2007) and NIST Statistical Test Suite (Bassham III *et al*., 2010), this research study recognizes that the NIST Statistical Test Suite is reliable for executing the test. The NIST Statistical Test Suite is developed by the National Institute of Standards and Technology, USA (NIST). Previously, NIST Statistical Test Suite has been used to test the randomness of candidates from AES (Soto and Bassham, 2000) and AKSA (MySEAL, 2018). Besides, the NIST Statistical Test Suite has been used to check several lightweight block cipher algorithms for their randomness.

Randomness analyses of the lightweight block cipher algorithm using the NIST Statistical Test Suite have been extensively carried out on KTANTAN (Abdullah *et al*., 2011), KATAN (Lot *et al*., 2011), LBlock (Abdullah *et al*., 2014), SPECK (Chew *et al*., 2015), SIMON (Shah *et al*., 2015; 2019), Modified Version of LBlock Block Cipher (Abdullah *et al*., 2015), RECTANGLE (Zakaria *et al*., 2020) and GRAIN-128 (Zawawi *et al*., 2013).

Description of Algorithm

PRESENT is an SPN-based algorithm that runs in 31 rounds. Each PRESENT round is defined by three layers, i.e., AddRoundKey, Substitution and Permutation. Figure 1 shows the PRESENT process.

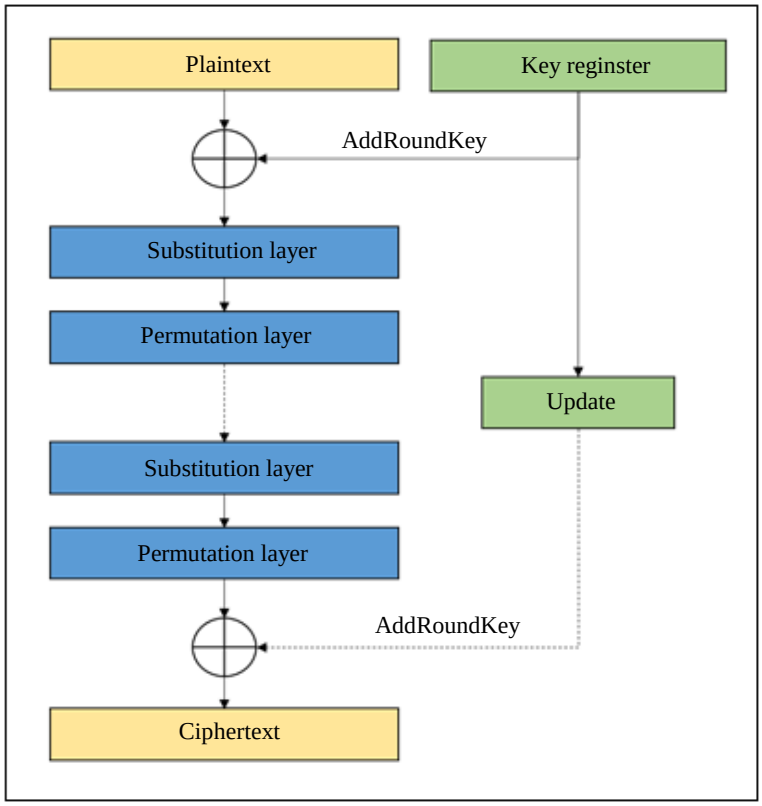

Fig. 1: The PRESENT process

Table 1: S-box used in PRESENT

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2: Permutation box used in PRESENT

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P[i]$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P[i]$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P[i]$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P[i]$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

AddRoundKey Layer

The 64-bit input of the round function is XORed with the AddRoundKey layer sub-key. This layer is described as follows:

$$b_j \rightarrow b_j \oplus k_j^i \quad (1)$$

where, b_j is the current state and k_j^i is the jth subkey bit of round key, K_i. Here, $1 \le i \le 32$, $0 \le j \le 63$.

S-box Layer

The sixteen (16) times 4-bit to 4-bit S-box implementation is used as the parallel non-linear substitution layer just after the XOR sub-key. The contents of the S-box are given in Table 1.

Permutation Layer

Finally, a permutation for diffusion is performed in the permutation layer. The details of the permutation layer is tabulated in Table 2. The permutation layer transfers bits from the x-input to the y-output. These steps are repeated for each round.

Key Schedule

Firstly, the 80-bit key will be registered in the key register K of the PRESENT key system and marked as $K = k_{79}\ldots k_0$. In round j, PRESENT extracts the 64-bit sub-keys, i.e., $K_i = k_{63}\ldots k_0 = k_{79}\ldots k_{16}$. Then, the value of the 80-bit key register is left-rotated by 61 bit positions. After that, the S-box moves the four most important bits (bits of K from 79 to 76). Finally, the $k_{19}k_{18}k_{17}k_{16}k_{15}$ are XORed with the least round counter bits. The whole process is described below:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus rc$

where, S is the S-box and rc is the round counter.

Methodology

The randomness testing method consists of several steps, i.e., sample preparation, performing randomness analysis and evaluating the test result. In order to prepare the samples for the randomness test, six data sets are analyzed. Each data set is selected based on its specific function. After preparing the sample, the algorithms are tested using the NIST Statistical Test Suite in order to evaluate the randomness of the algorithm. Finally, the result of the statistical test is evaluated. Figure 2 shows the research flows.

Data Categories

The randomness test is performed for a complete round of PRESENT based on the 1% significance level. Six data categories are used to construct data input in the form of plaintext or key as shown in Table 3. Data categories included in this analysis are Strict Key Avalanche (StrictKey), Strict Plaintext Avalanche (StrictPT), Low Density Key (LowKey), High Density Key (HighKey), Low Density Plaintext (LowPT) and High Density Plaintext (HighPT). As accordance to (Bassham III *et al*., 2010) a sample size is disproportional to the significance level. Thus, 100 sample size for each data categories are generated. The blocks number formed in each sample is depending on the block and key sizes (Abdullah *et al*., 2014).

To establish a large bit sequence for the test, the derived blocks are concatenated. Due to the large amount of time required to produce each sample, the significance level of 0.01 was selected. In addition, randomness analysis that has been conducted on KTANTAN algorithms (Abdullah *et al*., 2011), KATAN (Lot *et al*., 2011), LBlock (Abdullah *et al*., 2014), SPECK (Chew *et al*., 2015), SIMON (Shah *et al*., 2015; 2019), Modified Version of LBlock Block Cipher (Abdullah *et al*., 2015), RECTANGLE (Zakaria *et al*., 2020) and GRAIN-128 (Zawawi *et al*., 2013) also uses significance level of 0.01.

DOI: 10.3844/jcssp.2020.1639.1647

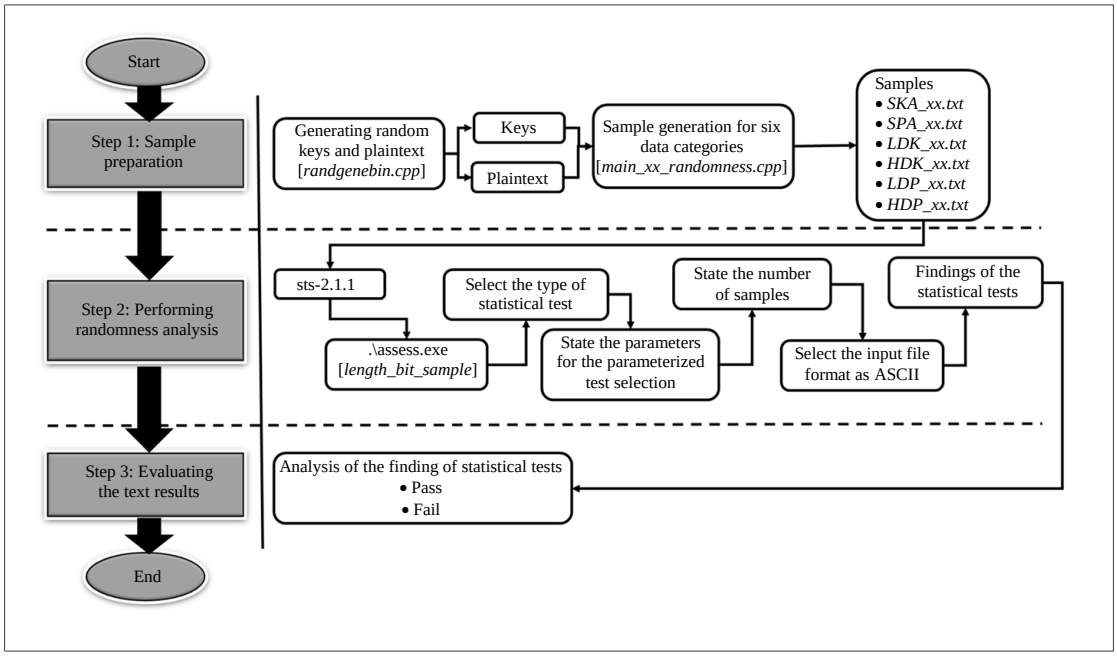

Fig. 2: The research flows

Table 3: Sample preparations using data categories

Data categories	Key	Plaintext	Derived blocks
StrictKey	196 randoms	All zero	15,680
StrictPT	All zero	245 randoms	15,680
LowKey	3,241 specifics	3,241 randoms	3,241
HighKey	3,241 specifics	3,241 randoms	3,241
LowPT	2,081 randoms	2,081 specifics	2,081
HighPT	2,081 randoms	2,081 specifics	2,081

a. Strict Key Avalanche (StrictKey)

StrictKey examines the sensitivity of each algorithm to key changes. One hundred samples are generated. Each sample requires 1,003,520 bits of binary sequences. The samples are constructed from 196 sets of 80-bit random keys and a set of all-zero plaintext blocks. Each block of random key is then used as a base-key. The base-key is encrypted with the all-zero plaintext row in order to create a base-ciphertext block. Then, in order to get the disturbed-ciphertext, each bit of the base-key is flipped and encrypted with its respective length of all-zero plaintext block. Each block of disturbed-ciphertext is then XORed with the base-ciphertext and concatenated in order to generate a binary output containing the least number of bits for each sample.

b. Strict Plaintext Avalanche (StrictPT)

StrictPT examines the sensitivity of each algorithm on the changes in plaintext. One hundred samples are generated and a total of 1,003,520-bit binary sequences are required for each sample. The samples are built from 245 sets of random 64-bit plaintext and a set of key blocks consisting of zeroes. Then, each random plaintext block is used as a base-plaintext. The base-plaintext is encrypted with the key block consisting of zeroes in order to derive a base-ciphertext block. Then, each bit of the base-plaintext is flipped and encrypted with its respective length of the key block to obtain the disturbed-ciphertext. Each disturbed-ciphertext block is then XORed with the base-ciphertext and concatenated to generate a binary output consisting of the least number of bits for each sample.

c. Low Density Key (LowKey)

In this data category, a data set consisting of one hundred sequences is generated based on the low density 80-bit key blocks. For each key block, a random 64-bit plaintext block is used. A total of 3,241 ciphertext blocks are generated for this data category. The first ciphertext block is obtained using a block consisting of zero bit key. The subsequent ciphertext block (up to the ciphertext block number 81) is obtained by using the key blocks (with a single one) in each possible bit position. For the remaining ciphertext blocks, the key blocks with two ones and 78 zeroes are obtained (the two ones appear within the length of the key in each combination of two bits position). The derived block of

ciphertext is then concatenated in order to produce 207,424 bits of binary sequence.

d. High Density Key (HighKey)

In this data category, a data set consisting of hundred sequences is generated based on the high density 80-bit key blocks. For each key block, a random 64-bit plaintext block is used. A total of 3,241 ciphertext blocks are generated for this data category. The first block of ciphertext has been obtained using a key block consisting of all ones. The subsequence ciphertext blocks (up to the ciphertext block number 81) are obtained by using a single zero key block in each possible bit position. Then, for the remaining ciphertext blocks, two key blocks of zeroes and 78 key blocks of ones are adopted (the two zeroes appear within the length of the key in each combination of two bits position). The derived ciphertext block is then concatenated to produce 207,424 bits of binary sequence.

e. Low Density Plaintext (LowPT)

In this data category, a data set consisting of one hundred sequences is generated based on the low density 64-bit plaintext block. For each plaintext block, a random 80-bit key block is used. A total of 2,081 ciphertext blocks are generated for this data category. The first ciphertext block is obtained by using the block consisting of all-zero plaintext. The subsequent ciphertext blocks (up to the ciphertext block number 65) are obtained by using the blocks of plaintext with a single one in each possible bit position. The remaining ciphertext blocks are then obtained by using a plaintext block consisting of two ones and 62 zeroes (both appear in each combination of two bits of position within the length of the plaintext). The derived ciphertext block is then concatenated to produce 133,184 bits of binary sequence.

f. High Density Plaintext (HighPT)

In this data category, a data set consisting of one hundred sequences is generated based on the 64-bit high density plaintext block. Each plaintext block uses a random 80-bit key block. For this category of data, a total of 2,081 ciphertext blocks are generated. The first ciphertext block is generated by using the all-one plaintext blocks. The subsequent ciphertext block (up to the ciphertext block number 65) is obtained by using the plaintext block with a single zero in each possible bit position. The remaining ciphertext blocks are extracted by using plaintext blocks consisting of two zeroes and 62 ones (the two zeroes occur in every combination of two bit locations within the plaintext length). The derived ciphertext block is then concatenated in order to produce 133,184 bits of binary sequence. Table 4 summarizes the length of the output sequence generated for each sample in each data category.

Table 4: Length of output sequences generated according to data categories

Data categories	Length of output (bits) per sample
StrictKey	1,003,520
StrictPT	1,003,520
LowKey	207,424
HighKey	207,424
LowPT	133,184
HighPT	133,184

NIST Statistical Test Suite

The NIST Statistical Test Suite developed by the National Institute of Standards and Technology, USA (NIST) is used to perform the randomness analysis. NIST Statistical Test Suite is a random test kit for binary sequences generated on the hardware- or software basis, either by random cryptography or pseudorandom generation numbers.

This test suite consists of 15 tests which are divided into two categories, i.e., Parameterized and Non-parameterized Test Selections. The statistical tests for parameterized test selection include Block Frequency (BlockFreq), Non-Overlapping Templates (Non-Over), Overlapping Template (Overlapping), Maurer's Universal (MUniversal), Linear Complexity (LinearC), Serial (Serial) and Approximate Entropy (Apen). The statistical tests of Non-Parameterized Test Selection, however, consist of Frequency (Freq), Runs (Runs), Longest Runs of Ones (LongestRuns), Binary Matrix Rank (BMR), Spectral (Spectral), Cumulative Sums (CuSum) (Forward and Reverse), Random Excursion (RanEx) and Random Excursion Variant (RanExVar). The descriptions of statistical randomness tests are given below:

- BlockFreq: To evaluate if the number of blocks in the M-bit block is approximately M/2 where *M* is the length of each block
- Non-Over: To reject sequences that display too many occurrences of a given non-periodic pattern
- Overlapping: To reject sequences that display too many or too few occurrences of m-bit patterns
- MUniversal: Detecting if the sequence can be substantially compressed without loss of information
- LinearC: To determine whether the sequence is random
- Serial: 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)
- Apen: Comparison of the frequency of overlapping blocks consisting of two

consecutive/adjacent lengths (m and m +1) with the predicted result for the series normally distributed (m-bit is the length of each block)
- Freq: In a completely random sequence, deciding whether or not the number of zeroes and ones in a sequence is identical to that
- Runs: To evaluate whether or not the number of runs of one and zeros of different lengths is equivalent to that of a random sequence
- LongestRuns: To evaluate whether the longest run of ones is compatible with the longest run of ones in a random sequence
- BMR: To search for linear dependency between fixed length substrings in the original sequence
- Spectral: To detect periodic features in the sequence being evaluated, which is a useful indicator of randomness error
- Cusum (Forward/Reverse): To assess if the number of partial sequences occurring in the sequence being checked is either too big or too small
- RanEx: To assess if the number of visits to a specific state within a loop will deviate from that in a random series
- RanExVar: To detect the difference between the distribution of the number of visits in a random walk and that in a given state

Every sample in each test requires a minimum number of bit length in which the value is tabulated in Table 5.

All tests except CuSum, Serial, Non-Over, RanEx and RanExVar should produce one p-value for every sample. CuSum and Serial tests produce two p-values for every test. Non-Over test produces 148 p-values for every sample. Table 6 shows the p-values provided by each sample in compliance with the statistical test.

A user should determine the parameter value for each test in the Parameterized Test Selection as explained by (Bassham III *et al*., 2010). Table 7 shows the list of input quantity for the parameters used in each test in the Parameterized Test Selection.

Table 5: Minimum bit of length required for each statistical test for each statistical test

Statistical test	Minimum bit length
BlockFreq	100
Non-Over	100
Overlapping	106
MUniversal	387,840
LinearC	106
Serial	100
Apen	100
Freq	100
Runs	100
LongestRuns	128
BMR	38,912
Spectral	1,000
Cusum (Forward/Reverse)	100
RanEx	106
RanExVar	106

Table 6: Breakdown of the p-value(s) obtained for each statistical test

Statistical test	*p*-value(s)
BlockFreq	1
Non-Over	148
Overlapping	1
MUniversal	1
LinearC	1
Serial	2
Apen	1
Freq	1
Runs	1
LongestRuns	1
BMR	1
Spectral	1
Cusum (Forward/Reverse)	2
RanEx	8
RanExVar	18

Table 7: Input for the parameterized test selection

Statistical test	Parameter(s)
BlockFreq	M = 20,000
Non-Over	m = 9
Overlapping	m = 9
MUniversal	L = 7, Q = 1280
LinearC	M = 500
Serial	m =2

Empirical Results and Analysis

In this analysis, the range of acceptable proportions for the binary sequences is determined using the confidence interval (Bassham III *et al*., 2010):

$$p' \pm 3\sqrt{\frac{p'(1-p)}{s}} \quad (2)$$

where, p' = 1-*sig*, *sig* is the significance level (*sig* = 0.01) and *s* is the sample size which is equal to one hundred ciphertexts except for RanEx and RanExVar tests. If the proportion falls outside the range $\left[p'_a, p'_b\right]$, the data is regarded as non-random.

Test such as Overlapping, LinearC, RanEx and RanExVar require certain number of bits while MUniversal test requires at least 387,840 bits. Therefore, the analysis of the output sequence generated from LowKey, LowPT, HighKey and HighPT data categories cannot be performed in these tests. There are 188 *p*-values obtained from StrictKey and StrictPT and 159 *p*-values obtained from LowKey, HighKey, LowPT and HighPT.

Since this analysis uses one hundred samples and the significance level is set at 0.01, the appropriate ranges for all tests except for RanEx and RanExVar tests are within [0.95, 1.01]. RanEx and RanExVar tests may not require all 100 binary sequences, as some of the binary sequences do not have enough cycles for conducting the test. Only those samples with more than 500 cycles are assessed. Samples with inadequate number of cycles are not considered. Therefore, the ranges of acceptable rejection of these two tests would vary (Table 8) depending on the samples meeting the requirements.

NIST suggests that a data can be considered as random if and only if the sequence(s) pass all testing procedures. If the tested sequence(s) fail one or more randomness testing procedures, there is a clear proof of non-randomness.

The results of the analysis of PRESENT are summarized in Table 9. If the rejected sequence falls within the acceptable rejection range, the result is Pass (P). Otherwise, the result is Fail (F). For data category that has failed sequences, the number of failed sequences is indicated in bracket '()'.

Table 8: Acceptable rejection ranges for PRESENT

Statistical test	Number of sample evaluated	Acceptable rejection range
BlockFreq	100	[0.95, 1.01]
Non-Over	100	[0.95, 1.01]
Overlapping	100	[0.95, 1.01]
MUniversal	100	[0.95, 1.01]
LinearC	100	[0.95, 1.01]
Serial	100	[0.95, 1.01]
Apen	100	[0.95, 1.01]
Freq	100	[0.95, 1.01]
Runs	100	[0.95, 1.01]
LongestRuns	100	[0.95, 1.01]
BMR	100	[0.95, 1.01]
Spectral	100	[0.95, 1.01]
Cusu (Forward/Reverse)	100	[0.95, 1.01]
RanEx	53	[0.95, 1.03]
RanExVar	64	[0.95, 1.03]

Table 9: Randomness analysis result for full round of PRESENT

Statistical test	StrictKey	StrictPT	LowKey	HighKey	LowPT	HighPT
BlockFreq	P	P	P	P	P	P
Non-Over	F(1)	F(11)	P	F(1)	F(1)	F(2)
Overlapping	P	P	P	P	P	P
MUniversal	P	P	P	P	P	P
LinearC	P	P	P	P	P	P
Serial	P	P	P	P	P	P
Apen	P	P	P	P	P	P
Freq	P	P	P	P	P	P
Runs	P	P	P	P	P	P
LongestRuns	P	P	P	P	P	P
BMR	P	F(1)	P	P	P	P
Spectral	P	P	P	P	P	P
Cusum (Forward/Reverse)	P	P	P	P	P	P
RanEx	P	P	P	P	P	P
RanExVar	P	P	P	P	P	P

As shown in Table 9, the total number of failed ciphertext sequences from the PRESENT algorithm is 17. In the StrictKey data category, PRESENT fails 1 statistical test in the Non-Over test. In StrictPT, PRESENT fails 11 statistical tests in the Non-Over test and 1 statistical test in the BMR. Also, PRESENT algorithm fails 1 Non-Over test in HighKey. In LowPT and HighPT, PRESENT shows non-randomness in 1 and 2 Non-Over tests, respectively.

Only one data category shows the evidence of randomness from the binary sequences generated in PRESENT. Therefore, it is evident that output sequences generated from PRESENT are essentially non-random.

Conclusion

By using the NIST Statistical Test Suite, a randomness analysis based on 1% significance level has been performed on PRESENT. This analysis has been conducted on 100 samples falling under six data categories, i.e., StrictKey, StrictPT, LowKey, HighKey, LowPT and HighPT. The significance level has been set to 0.01 in order to determine whether or not the output sequence generated from the algorithm is random. The result shows that the output sequences generated from PRESENT are essentially non-random based on the 1% significance level. An algorithm that passes all of the statistical tests does not guarantee its security (Isa and Z'aba, 2014). However, a secure algorithm should pass all of the tests (Zakaria *et al*., 2020). For security purposes, enhancement on the PRESENT is suggested in the future to improve its security. As mentioned above, the Non-Over test results from StrictKey, StrictPT, HighKey, LowPT and HighPT are largely negative (fail). Therefore, it is advisable to avoid using low density values for plaintext and high density values for keys and plaintext in the PRESENT algorithm.

Acknowledgment

This study has been funded by the Ministry Of Higher Education (MOHE) Malaysia under Fundamental Research Grant Scheme (FRGS) project no. FRGS/1/2020/STG06/UKM/02/2 (*Sifer Blok Ringan Tempatan Menggunakan Struktur Matematik Feistel Terubahsuai*) through Universiti Kebangsaan Malaysia. The authors would like to thank Cryptography Development Department, CyberSecurity Malaysia for the support and guidance given. The authors would also like to thank editors and all anonymous reviewers for their valuable comments.

Author's Contributions

Isma Norshahila Binti Mohammad Shah: Developed the main idea and scheme, executed the experimental tests and prepared the manuscript.

Eddie Shahril Bin Ismail: Proposed some modifications and corrections of the security and efficiency of the scheme. Improved the manuscript writing.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Abdullah, N. A. N., Chew, L. C. N., Zakaria, A. A., Seman, K., & 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., Lot, N. H., Zawawi, A., & Rani, H. A. (2011, December). Analysis on lightweight block cipher, KTANTAN. In 2011 7th International Conference on Information Assurance and Security (IAS) (pp. 46-51). IEEE.

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

Bassham III, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., ... & Heckert, N. A. (2010). Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards & Technology.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015). SIMON and SPECK: Block Ciphers for the Internet of Things. IACR Cryptol. ePrint Arch., 2015, 585.

Blondeau, C., & Nyberg, K. (2014, May). Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 165-182). Springer, Berlin, Heidelberg.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., ... & Vikkelsoe, C. (2007, September). PRESENT: An ultra-lightweight block cipher. In International workshop on cryptographic hardware and embedded systems (pp. 450-466). Springer, Berlin, Heidelberg.

Chew, L., Chew, N., Norshahil, I., Shah, M., Azura, N., Abdullah, N., ... & Zakaria, A. A. (2015). Randomness analysis on Speck family of lightweight block cipher. International Journal of Cryptology Research, 5(1), 44-60.

De Canniere, C., Dunkelman, O., & Knežević, M. (2009, September). KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 272-288). Springer, Berlin, Heidelberg.

Demirhan, H., & Bitirim, N. (2016). Statistical testing of cryptographic randomness. İstatistikçiler Dergisi: İstatistik ve Aktüerya, 9(1), 1-11.

Hathaway, L. (2003). National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information. National Security Agency, 23.

Isa, H., & Z'aba, M. R. (2014). Randomness of the PRINCE block cipher.

Jeong, K., Kang, H., Lee, C., Sung, J., & Hong, S. (2012). Biclique Cryptanalysis of Lightweight Block Ciphers PRESENT, Piccolo and LED. IACR Cryptol. ePrint Arch., 2012, 621.

Jeong, K., Lee, Y., Sung, J., & Hong, S. (2013). Improved differential fault analysis on PRESENT-80/128. International Journal of Computer Mathematics, 90(12), 2553-2563.

ISO 29192-2. (2012). ISO 29192-2:2012(E) Information technology - Security techniques - Lightweight cryptography Part 2 Block ciphers. Standard, International Organization for Standardization, Geneva, CH. https://www.sis.se/api/document/preview/914247/

Leander, G., Paar, C., Poschmann, A., & Schramm, K. (2007, March). New lightweight DES variants. In International Workshop on Fast Software Encryption (pp. 196-210). Springer, Berlin, Heidelberg.

L'Ecuyer, P., & Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. ACM Transactions on Mathematical Software (TOMS), 33(4), 1-40.

Lot, N. H., Abdullah, N. A. N., & Rani, H. A. (2011, November). Statistical analysis on KATAN block cipher. In 2011 International Conference on Research and Innovation in Information Systems (pp. 1-6). IEEE.

Marsaglia, G. (2008). The Marsaglia random number CDROM including the diehard battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/

Marton, K., & Suciu, A. (2015). On the interpretation of results from the NIST statistical test suite. Science and Technology, 18(1), 18-32.

McKay, K., Bassham, L., Sönmez Turan, M., & Mouha, N. (2016). Report on lightweight cryptography (No. NIST Internal or Interagency Report (NISTIR) 8114 (Draft)). National Institute of Standards and Technology.

MySEAL, K. F. (2018). Projek myseal: Kriteria penyerahan dan penilaian versi 2.0 [2018]. Technical report, CyberSecurity Malaysia, Malaysia. https://myseal.cybersecurity.my/en/files/CD-5-RPT-0218-Kriteria_MySEAL_Versi_2.0-V1a.pdf

Özen, O., Varıcı, K., Tezcan, C., & Kocair, Ç. (2009, July). Lightweight block ciphers revisited: Cryptanalysis of reduced round PRESENT and HIGHT. In Australasian Conference on Information Security and Privacy (pp. 90-107). Springer, Berlin, Heidelberg.

Pub, N. F. (2001). 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/NIST, November 26, 2001.

Renauld, M., & Standaert, F. X. (2009, December). Algebraic side-channel attacks. In International Conference on Information Security and Cryptology (pp. 393-410). Springer, Berlin, Heidelberg.

Shah, I. N. M., Chew, L. C. N., Yusof, N. A. M., Abdullah, N. A. N., Zawawi, N. H. L. A., & Rani, H. A. (2015). Statistical analysis on lightweight block cipher, Simon. International Journal of Cryptology Research. 5(2), 28-44.

Shah, I. N. M., Rani, H. A., Ahmad, M. M., & Ismail, E. S. (2019). Cryptographic Randomness Analysis on Simon32/64.

Soto, J., & Bassham, L. (2000). Randomness testing of the advanced encryption standard finalist candidates. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

Wu, W., & Zhang, L. (2011, June). LBlock: a lightweight block cipher. In International Conference on Applied Cryptography and Network Security (pp. 327-344). Springer, Berlin, Heidelberg.

Yang, L., Wang, M., & Qiao, S. (2009, December). Side channel cube attack on PRESENT. In International Conference on Cryptology and Network Security (pp. 379-391). Springer, Berlin, Heidelberg.

Zakaria, A. A., Azni, A. H., Ridzuan, F., Zakaria, N. H., & Daud, M. (2020). Randomness Analysis on RECTANGLE Block Cipher. In Cryptology and Information Security Conference 2020 (p. 133).

Zawawi, N. H. L. A., Seman, K., & Mohd Zaizi, N. J. (2013, September). Randomness analysis on grain-128 stream cipher. In AIP Conference Proceedings (Vol. 1557, No. 1, pp. 15-20). American Institute of Physics.