International Journal of Cryptology Research 9(1): 52 - 75 (2019)

Enhanced Statistical Analysis Evaluation Using CSM Randomness Test Tool

Abdul Alif Zakaria[*1], **Hazlin Abdul Rani**[1], and **Nik Azura Nik Abdullah**[1]

[1]*Cryptography Development Department, CyberSecurity Malaysia, Kuala Lumpur, Malaysia.*

E-mail: alif@cybersecurity.my, hazlin@cybersecurity.my, azura@cybersecurity.my
[*]*Corresponding author*

ABSTRACT

Random numbers are at the heart of modern cryptography, and having access to a source of randomness is crucial for information security. Tests for randomness determine whether a data set has a recognizable pattern which indicates that it is significantly non-random. However, the randomness test is very time consuming which requires a large file size of data to be tested. Also, the randomness test requires a repeated process for a different types of tests. In this research paper, we introduced a cryptographic evaluation tool to evaluate the randomness of a cryptographic algorithm. This tool is an extended work done on the NIST Statistical Test Suite which is the most popular statistical test for the evaluation of the quality of random number generators. All of the results from each of the 15 tests in the test suite are compiled and analyzed simultaneously. The proposed tool finally generated a complete randomness evaluation report which provides a conclusion to indicate whether the tested cryptographic algorithm passes or fails each test. It took about three hours to produce a complete analysis report compared to eight hours using a conventional method. With the presence of this

tool, the process of evaluating the randomness of a cryptographic algorithm is simplified and automated which resulted in reducing the evaluation timeframe.

Keywords: Cryptography, Randomness, Statistical Analysis, NIST Statistical Test Suite, Automated Report

1 INTRODUCTION

The need for random and pseudorandom numbers arises in many cryptographic applications. The randomness of the cryptographic algorithm signicantly inuences the security of any cryptographic systems (Chen et al., 2015). For example, common cryptosystems employ keys that must be generated in a random fashion. Many cryptographic protocols also require random or pseudorandom inputs at various points, for example, auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols.

Random sequences or random numbers can be generated from true random sources, such as air particles, atmospheric noise and radioactive decay (Doğanaksoy et al., 2015). However, implementation of these sources in a cryptographic algorithm is unpractical. Cryptographic algorithm supposed to produce more random sequences than true random sources to avoid statistical analysis that can defeat the cryptographic system. Randomness can be measured by statistical tests and hence security evaluation of a cryptographic algorithm deeply depends on statistical randomness tests.

NIST Statistical Test Suite (Rukhin et al., 2001, Zhu et al., 2016) is the most popular statistical tests for evaluation of the quality of random number generators. This test suite includes 15 different tests which must be passed by the tested cryptographic algorithm. Should the algorithm have failed any of the tests, it is considered failed the NIST test and declared as non-random.

NIST Statistical Test Suite allows users to process a large file size for randomness test (Suciu et al., 2010a). However, a complete calculation of the

randomness test for a large amount of data is very time consuming (Dabal and Pelka, 2012). CyberSecurity Malaysia as the main cryptographic algorithm evaluator in Malaysia, has been experiencing evaluating a large number of samples using the NIST Statistical Test Suite. For 1,000 test samples, it takes about three hours to run the NIST Statistical Test Suite. Approximately 1GB of text files size or equivalent to one billion bits that need to be processed by the test suite.

Cryptography Development Department has proposed a cryptographic randomness evaluation tool namely CSM Randomness Test Tool. This development is inspired by our routine work as a cryptographic module evaluator in CyberSecurity Malaysia. There are three objectives of this project which are (1) to run randomness test in one platform, (2) to reduce evaluation time frame, and (3) to generate a complete evaluation report automatically.

The organization of this paper is divided into six sections. Section 2 describes statistical analysis packages focusing on the NIST Statistical Test Suite and previous works done on this package. Section 3 presents common practice for randomness evaluation. Section 4 introduces the proposed randomness evaluation tool. Section 6 demonstrates the time performance analysis of the proposed randomness evaluation tool. Finally, section 7 points out the conclusion and future works of this research.

2 STATISTICAL ANALYSIS PACKAGES

There have existed many approaches in evaluating cryptographic algorithm, and among these, the randomness testing has been identified to be very important. Therefore, when designing a cryptographic algorithm, it is vital to make sure that the ciphertext produce from this algorithm must be random. To determine whether the tested algorithm fulfills this requirement, randomness testing using statistical analysis can be applied. Apart from the NIST Statistical Test Suite, there are other statistical analysis packages that can be used to evaluate the randomness of a cryptographic algorithm.

Knuth (2014) describes eleven statistical tests in his book. The eleven tests are Frequency Test, Serial Test, Gap Test, Poker Test, Coupon Collector's Test, Permutation Test, Ryun Test, Maximum-of-t Test, Collision Test, Birthday Spacing Test and Serial Correlation Test.

Crypt-XS suite of statistical test (Caelli et al., 1992) was developed by researchers at the Information Security Research Centre at the Queensland University of Technology in Australia. Seven tests included in this analysis package namely: Frequency Test, Binary Derivative Test, Change Point Test, Runs Test, Sequence Complexity Test and Linear Complexity Test.

Marsaglia's DIEHARD (Marsaglia, 2008) is the oldest and most well-known test suite. This test suite contains eighteen tests namely: Birthday Spacing Test, Overlapping 5-Permutation Test, Binary Rank Test for 31 X 31, 32 X 32 and 6 X 8 matrices, Bit Stream Test, Overlapping-Pair-Sparse-Occupancy (OPSO) Test, Overlapping-Quadruples-Sparse-Occupancy (OQSO) Test, DNA Test, Count-The-1's Test in a stream bytes and in specific bytes, Parking Lot Test, Minimum Distance Test, 3D Sphere Test, Squeeze Test, Overlapping Sums Test, Runs Test and Craps Test.

DIEHARDER (Brown et al., 2013) Battery of Test is a test suite developed by Robert G. Brown in 2004, which comprises all eighteen tests from DIEHARD Battery of Test, three tests from the NIST Statistical Test Suite (Frequency Test, Runs Test and Serial Test), and several other test (RGB Bit Distance Test, RGB minimum Distance Test, RGB Permutations Test, RGB Lagged Sum Test and KS Test Test) in a single package. It fixes several problems with the Diehard package and uses the GNU Scientific Library interface.

After reviewing many possible randomness testing tools available, NIST Statistical Test Suite is contemplated to be a reliable tool to perform randomness tests in this work. This test package is presented in a simple GUI that permits the user to select or re-select the set of statistical tests to be executed (Abdullah et al., 2015). Although DIEHARDER Battery of Test covers a wider aspect of randomness, tests in the NIST Statistical Test Suite are sufficient as it focuses on a variety of different types of non-randomness that could exist in a sequence. Detail description of NIST Statistical Test Suite is presented in next section.

(a) NIST Statistical Test Suite

The NIST Statistical Test Suite is a statistical package that was developed by National Institute of Standards and Technology (NIST) to test the randomness of binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators (Rukhin et al., 2001). This statistical package focuses on a variety of different types of non-randomness that could exist in a sequence. The NIST Statistical Test Suite consists of fifteen tests which are separated into two categories (Lot et al., 2014). Eight tests categorize as Non-Parameterized Test Selection does not require users to enter any parameter in obtaining the p-values for each test, whereas the other seven tests categorized as the Parameterized Test Selection requires users to define one or two parameter value(s). All fifteen tests are divided according to their category as per listed in Table 1 .

Test No.	Non-Parameterized Test Selection	Test No.	Parameterized Test Selection
1	Frequency Test	9	Block Frequency Test
2	Runs Test	10	Non-Overlapping Templates Test
3	Longest Runs of Ones Test	11	Overlapping Template Test
4	Random Excursion Variant Test	12	Serial Test
5	Random Excursion Test	13	Approximate Entropy Test
6	Binary Matrix Rank Test	14	Maurer's Universal Test
7	Spectral Discrete Fourier Transform (DFT) Test	15	Linear Complexity Test
8	Cumulative Sums (Forward /Reverse) Test		

Table 1: 15 NIST Statistical Tests. Inputs are required for parameterized test selection, and it is suggested to follow the recommendation from NIST.

When conducting randomness testing using the NIST Statistical Test Suite, a statistical test is formulated to test a specific null hypothesis (H0), where the sequence being tested is random (Suresh et al., 2013). To relate with this null hypothesis is the alternative hypothesis (Ha), where the sequence being tested is non-random. A test statistic value is computed on the sequence being tested and is then compared to the critical value, where a critical value is a relevant

randomness statistic computed by mathematical methods chosen and used to determine the acceptance or rejection of the null hypothesis. If the test statistic value exceeds the critical value, the null hypothesis for randomness is rejected. Otherwise, the null hypothesis is accepted.

The level of significance denoted as α, of a test is the probability that the test statistic will reject the null hypothesis. NIST recommended $\alpha = 0.001$ (0.1%) when describing all tests in the statistical test suite (Rukhin et al., 2001). The test statistic value is used to calculate a probability value denoted as p-value to determine the randomness of a sequence. If the p-value $\geq \alpha = p$-value ≥ 0.001, then the null hypothesis is accepted. This means that the sequence tested appears to be random with a confidence level of 99.9 %. If the p-value $< \alpha = p$-value < 0.001, then the null hypothesis is rejected. This means that the sequence tested appears to be nonrandom with a confidence level of 99.9 %.

To summarize this section, Table 2 presents the minimum number of bits required for each test as recommended by NIST (Zawawi et al., 2013). Since the requirement for a cryptographic algorithm to pass the randomness test is to pass all 15 statistical tests, it is recommended that users provide a minimum of 1,000,000 bits test samples so that all test fulfilled the minimum bits requirements and can be tested simultaneously.

Test No.	Non-Parameterized Test Selection	Required bits
1	Frequency Test	$n \geq 100$
2	Runs Test	$n \geq 100$
3	Longest Runs of Ones Test	$n \geq 128$
4	Random Excursion Variant Test	$n \geq 1{,}000{,}000$
5	Random Excursion Test	$n \geq 1{,}000{,}000$
6	Binary Matrix Rank Test	$n \geq 38{,}912$
7	Spectral Discrete Fourier Transform (DFT) Test	$n \geq 1{,}000$
8	Cumulative Sums (Forward /Reverse) Test	$n \geq 100$
Test No.	**Parameterized Test Selection**	**Required bits**
9	Block Frequency Test	$n \geq 100$
10	Non-Overlapping Templates Test	Not specified
11	Overlapping Template Test	$n \geq 1{,}000{,}000$
12	Serial Test	Not specified
13	Approximate Entropy Test	Not specified
14	Maurer's Universal Test	$n \geq 387{,}480$
15	Linear Complexity Test	$n \geq 1{,}000{,}000$

Table 2: Minimum bits required for each test. A minimum of 1,000,000 bits of test samples is sufficient and fulfilled the requirement for all tests.

Table 3 presents the number of p-values produced by each test on a single test sample and its number of rejection allowed for 1,000 test samples.

Test No.	Non-Parameterized Test Selection	No. of *p*-values	Max Rejection
1	Frequency Test	1	3
2	Runs Test	1	3
3	Longest Runs of Ones Test	1	3
4	Random Excursion Variant Test	18	30
5	Random Excursion Test	8	16
6	Binary Matrix Rank Test	1	3
7	Spectral Discrete Fourier Transform (DFT) Test	1	3
8	Cumulative Sums (Forward /Reverse) Test	2	3
Test No.	**Parameterized Test Selection**	**No. of *p*-values**	**Max Rejection**
9	Block Frequency Test	1	3
10	Non-Overlapping Templates Test	148	184
11	Overlapping Template Test	1	3
12	Serial Test	2	3
13	Approximate Entropy Test	1	3
14	Maurer's Universal Test	1	3
15	Linear Complexity Test	1	3

Table 3: Breakdown of the 188 p-values obtained by testing a single binary sequence. Test exceeds the maximum rejection number is consider failed.

(b) Previous Works on NIST Statistical Test Suite

The NIST Statistical Test Suite is the most used statistical analysis package where the users are from different background such as student, researcher, analyst and developer (Rukhin et al., 2001, Zhu et al., 2016). Its simplicity, and user-friendliness are the main factor of the increasing number of users choosing this tool over other statistical analysis packages. Therefore, a number of improvements have been proposed over the years to improve the performance of this tool so that the cryptographic algorithm evaluation process can be much easier and faster.

Suciu et al. (2010a) proposed a performance efficient version of the NIST statistical test suite for random and pseudorandom number generators based on a paradigm shift towards byte stream processing mode inside the tests. The research work presented the improvements applied to the NIST statistical test

suite for assessing the quality of random number generators and to emphasize the efficiency of the improved version displaying the benchmark results that demonstrate significant performance improvements. The results show that the execution time was reduced up to approximately fourteen times on the average for the described system.

Suciu et al. (2010b) presents the results on improving the NIST Statistical Test Suite by introducing parallelism and a paradigm shift towards byte processing delivering a design that is more suitable for todays multicore architectures. There is a stringent need for providing highly efficient statistical tests by improving and parallelizing the NIST test suite intends to fill this need. The results produced proves that the execution time using the proposed method being reduced up to 103 times compared to the original version provided by NIST.

Suciu et al. (2009) aimed at adapting GPGPU for several algorithms used for statistical testing of random and pseudorandom number sequences, based on the tests described in the NIST statistical test suite. The proposed method obtained a great speedup in comparison with existing implementation, up to 31.43 times faster. In more recent experiments show that this speedup can be increased up to 219.33 times faster using a CUDA based approach.

Chen et al. (2015) proposed a scheme to optimize the efciency of NIST Statistical Test Suite, that is, provide an optimized order of the tests in the test suite, so that an unqualied sequence can be rejected as early as possible. Evaluation results indicate that the optimized order improves the testing efciency by 33% in comparison with the original version of the test suite, and also approaches the optimum among all the possible orders. In conclusion, the optimization scheme provides a fast-implemented method to achieve the most optimal speed for on-the-y tests.

Previous works on NIST Statistical Test Suite are mainly focusing on reducing the execution time of the tool as discussed above. However, this paper is focusing on different improvement of the statistical analysis package which is to simplified randomness evaluation process for a cryptographic algorithm. The proposed work is using the original package that is available on NIST website and extended the work to give a better experience for users to evaluate

any cryptographic algorithm in a much simpler way. The idea of this proposed work is to skip the analysis part and reporting process so that everyone that does not have knowledge of cryptographic evaluation can test users algorithm themselves without hassle.

3 RANDOMNESS EVALUATION PRACTICE

This section describes the overall process of cryptographic algorithm randomness evaluation as illustrated in Figure 1. The first phase starts with the preparation of samples by the cryptographic algorithm developer. Preparation of samples must align with the type of algorithm desired to be tested such as stream cipher, block cipher, and pseudorandom number generator. For stream cipher and pseudorandom number generator algorithm, only one set of 1,000 keystream or samples is required. Meanwhile, the block cipher algorithm requires nine sets of 1,000 samples that represent nine data categories (Abdullah et al., 2011) which are the NIST requirements for block cipher test.

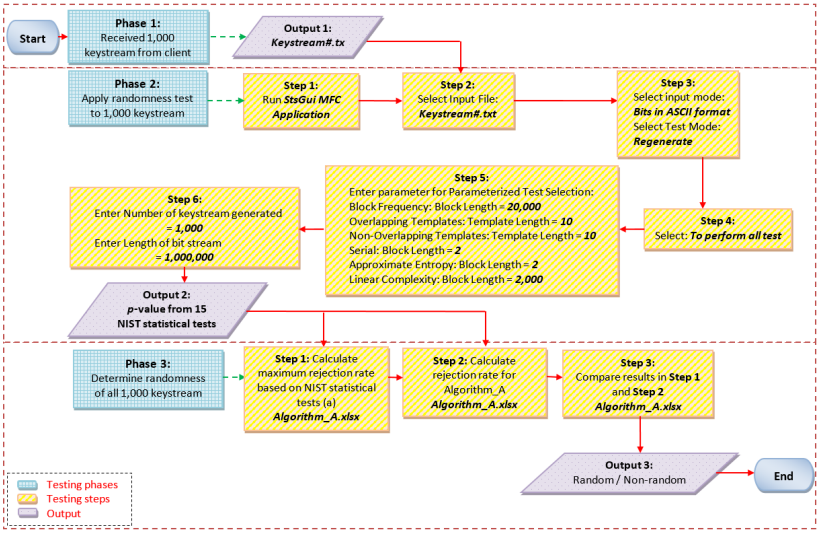


Figure 1: Process for Cryptographic Algorithm Randomness Evaluation.

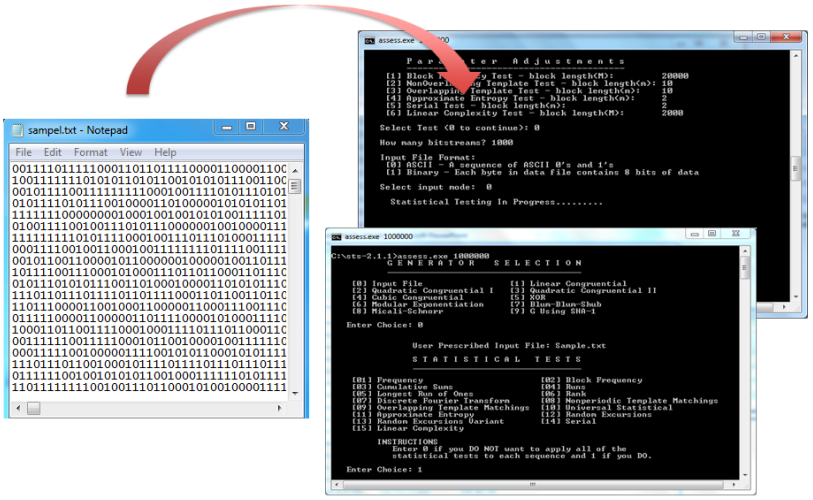
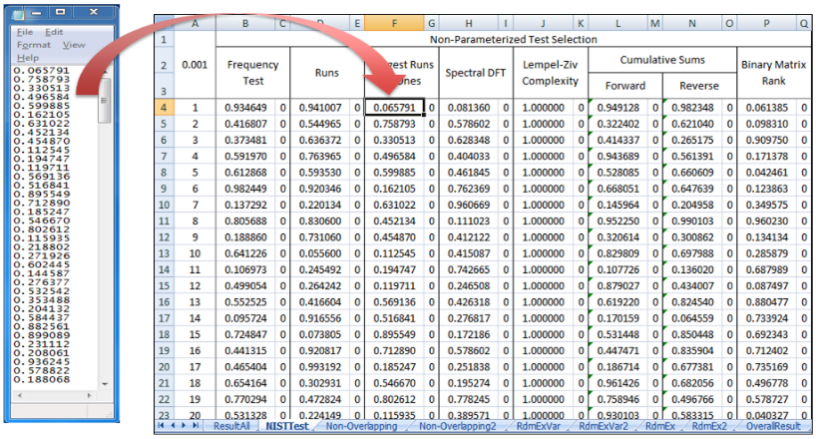


Figure 2: Run NIST Randomness Test. The Statistical test suite is available online from the NIST website. Command Prompt interacts with the user through a command-line interface in order to run the tests.

In Phase 2, NIST Statistical Test Suite is used by the evaluator to test the randomness of the test samples that have been received from the developer in phase 1. Upon execution of the test suite, as shown in Figure 2, the developer requires to select the input file, input mode, test mode, and test to be performed. In the parameterized test selection, parameters of each test are required to be entered, as well as the number of keystreams generated and the length of the bit stream. 15 different text files are generated from the test suite which contains the p-values of each test as shown in Figure 3.

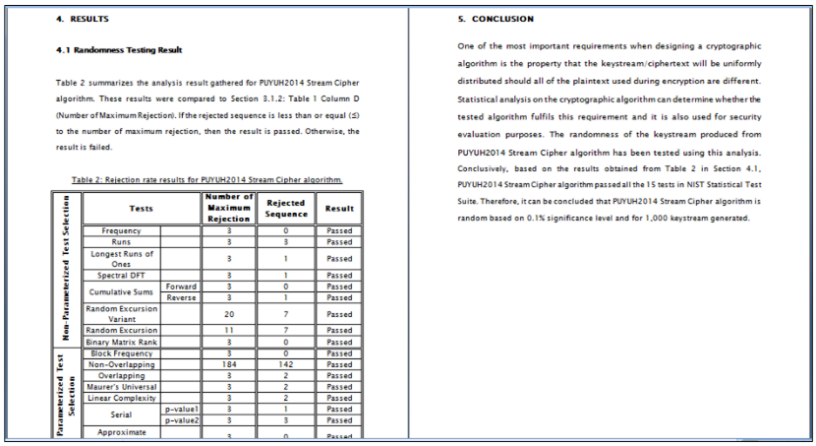
The analysis part of the generated p-values takes place in Phase 3. Firstly, the maximum rejection rate based on NIST Statistical Tests were calculated. Secondly, the rejected or failed samples are identified and summed up. Finally, compare the total number of rejected samples to the number of maximum rejection rate. If the rejected samples exceed the maximum rejection, the test is considered failed. Thus, the algorithm is declared as non-random according to NIST Statistical Test Suite. This is results the will be reported in the randomness evaluation report which will be prepared manually by the evaluator as shown in Figure 4.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Non-Parameterized Test Selection															
2	0.001	Frequency Test		Runs		est Runs nes		Spectral DFT		Lempel-Ziv Complexity		Cumulative Sums				Binary Matrix Rank	
3												Forward		Reverse			
4	1	0.934649	0	0.941007	0	0.065791	0	0.081360	0	1.000000	0	0.949128	0	0.982348	0	0.061385	0
5	2	0.416807	0	0.544965	0	0.758793	0	0.578602	0	1.000000	0	0.322402	0	0.621040	0	0.098310	0
6	3	0.373481	0	0.636372	0	0.330513	0	0.628348	0	1.000000	0	0.414337	0	0.265175	0	0.909750	0
7	4	0.591970	0	0.763965	0	0.496584	0	0.404033	0	1.000000	0	0.943689	0	0.561391	0	0.171378	0
8	5	0.612868	0	0.593530	0	0.599885	0	0.461845	0	1.000000	0	0.528085	0	0.660609	0	0.042461	0
9	6	0.982449	0	0.920346	0	0.162105	0	0.762369	0	1.000000	0	0.668051	0	0.647639	0	0.123863	0
10	7	0.137292	0	0.220134	0	0.631022	0	0.960669	0	1.000000	0	0.145964	0	0.204958	0	0.349575	0
11	8	0.805688	0	0.830600	0	0.452134	0	0.111023	0	1.000000	0	0.952250	0	0.990103	0	0.960230	0
12	9	0.188860	0	0.731060	0	0.454870	0	0.412122	0	1.000000	0	0.320614	0	0.300862	0	0.134134	0
13	10	0.641226	0	0.055600	0	0.112545	0	0.415087	0	1.000000	0	0.829809	0	0.697988	0	0.285879	0
14	11	0.106973	0	0.245492	0	0.194747	0	0.742665	0	1.000000	0	0.107726	0	0.136020	0	0.687989	0
15	12	0.499054	0	0.264242	0	0.119711	0	0.246508	0	1.000000	0	0.879027	0	0.434007	0	0.087497	0
16	13	0.552525	0	0.416604	0	0.569136	0	0.426318	0	1.000000	0	0.619220	0	0.824540	0	0.880477	0
17	14	0.095724	0	0.916556	0	0.516841	0	0.276817	0	1.000000	0	0.170159	0	0.064559	0	0.733924	0
18	15	0.724847	0	0.073805	0	0.895549	0	0.172186	0	1.000000	0	0.531448	0	0.850448	0	0.692343	0
19	16	0.441315	0	0.920817	0	0.712890	0	0.578602	0	1.000000	0	0.447471	0	0.835904	0	0.712402	0
20	17	0.465404	0	0.993192	0	0.185247	0	0.251838	0	1.000000	0	0.186714	0	0.677381	0	0.735169	0
21	18	0.654164	0	0.302931	0	0.546670	0	0.195274	0	1.000000	0	0.961426	0	0.682056	0	0.496778	0
22	19	0.770294	0	0.472824	0	0.802612	0	0.778245	0	1.000000	0	0.758946	0	0.496766	0	0.578727	0
23	20	0.531328	0	0.224149	0	0.115935	0	0.389571	0	1.000000	0	0.930103	0	0.583315	0	0.040327	0

ResultAll | NISTTest | Non-Overlapping | Non-Overlapping2 | RdmExVar | RdmExVar2 | RdmEx | RdmEx2 | OverallResult

Figure 3: Results Analysis Process. Results from each of the tests are required to be analyzed separately to determine whether it passed or failed.



4. RESULTS

4.1 Randomness Testing Result

Table 2 summarizes the analysis result gathered for PUYUH2014 Stream Cipher algorithm. These results were compared to Section 3.1.2; Table 1 Column D (Number of Maximum Rejection). If the rejected sequence is less than or equal (≤) to the number of maximum rejection, then the result is passed. Otherwise, the result is failed.

Table 2: Rejection rate results for PUYUH2014 Stream Cipher algorithm.

	Tests		Number of Maximum Rejection	Rejected Sequence	Result
Non Parameterized Test Selection	Frequency		3	0	Passed
	Runs		3	3	Passed
	Longest Runs of Ones		3	1	Passed
	Spectral DFT		3	1	Passed
	Cumulative Sums	Forward	3	0	Passed
		Reverse	3	1	Passed
	Random Excursion Variant		20	7	Passed
	Random Excursion		11	7	Passed
	Binary Matrix Rank		3	0	Passed
Parameterized Test Selection	Block Frequency		3	0	Passed
	Non-Overlapping		184	142	Passed
	Overlapping		3	2	Passed
	Maurer's Universal		3	2	Passed
	Linear Complexity		3	2	Passed
	Serial	p-value1	3	1	Passed
		p-value2	3	3	Passed
	Approximate		3	0	Passed

5. CONCLUSION

One of the most important requirements when designing a cryptographic algorithm is the property that the keystream/ciphertext will be uniformly distributed should all of the plaintext used during encryption are different. Statistical analysis on the cryptographic algorithm can determine whether the tested algorithm fulfils this requirement and it is also used for security evaluation purposes. The randomness of the keystream produced from PUYUH2014 Stream Cipher algorithm has been tested using this analysis. Conclusively, based on the results obtained from Table 2 in Section 4.1, PUYUH2014 Stream Cipher algorithm passed all the 15 tests in NIST Statistical Test Suite. Therefore, it can be concluded that PUYUH2014 Stream Cipher algorithm is random based on 0.1% significance level and for 1,000 keystream generated.

Figure 4: Writing Report Process. The results obtained from the analysis process is transformed into a report to provide the overview of randomness tests on a cryptographic algorithm.

4 PROPOSED WORK

This project proposed a new cryptographic tool to evaluate randomness of binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. The original statistical test package that was developed by the National Institute of Standards and Technology (NIST) focuses on a variety of different types of non-randomness that could exist in a sequence. Cryptography Development Department has developed a cryptographic randomness evaluation tool namely CSM Randomness Test Tool (CRTT) as an extended work from the NIST statistical test package that able to produce a complete randomness test report. This development is inspired by our routine work as a cryptographic module evaluator in CyberSecurity Malaysia.

The proposed tool will assist cryptography analyst to evaluate the randomness of a cryptographic algorithm and facilitate them to produce a complete analysis report within a short time. CRTT development process follows the flow in Figure 5. Study on randomness properties had been conducted during the literature review before the development of this evaluation tool started. The 15 tests that were developed by NIST to test the randomness of a cryptographic algorithm were studied in detail to ensure the tool can accurately evaluate the randomness properties of an algorithm. Inaccurate research may lead to inaccurate analysis report produced.

The graphical user interface of this tool was designed to fit users requirements specification so that it can be operated as simple as possible because we have to consider our potential users that might come from different background. Our aim was to develop a user-friendly evaluation tool so that everybody can operate this tool without hassle. The interface of this tool was developed using the Ext-JS framework which can be viewed in Figure 7 to Figure 10.

CRTT is a web-based application which has been developed using Ubuntu as the platform and Apache as the server. The code of this application was written in C and Perl programming language. Programming skill is very crucial where cryptography analyst needs to construct complicated mathematical

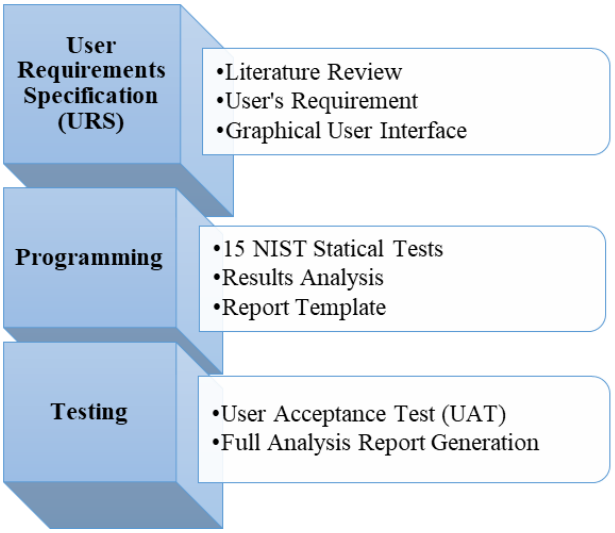


Figure 5: CRTT Development Process. URS describes the users requirement, statistical tests implementation, and the interface of CRTT. Programming process manages the randomness using codes from running the test to produce the final results. Testing process confirmed the execution of CRTT works well as required.

formula coding that will produce the desired testing result. In the programming process, results obtained from the 15 NIST Statistical Test Suite were compiled and analyzed automatically. Compared to manual testing on randomness, an analyst is required to analyze the raw results separately which takes a lot of time in order to obtain the final results. Each program has to be repeatedly run to make sure it will consistently produce results as expected.

Report templates for each test module were prepared to help cryptography analysts or users in producing a complete analysis report as shown in Figure 6. The report templates are containing each tests module process flowchart, definition, result, conclusion, and reference. Users need to provide a sample file, the name of the algorithm and choose test option. Within a short period of time, the randomness evaluation analysis report is ready to be presented as shown in Figure 11.

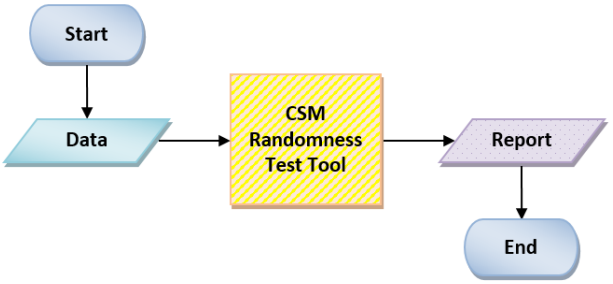


Figure 6: Randomness Evaluation Process Using CRTT. Would be much simpler and save a lot of time.

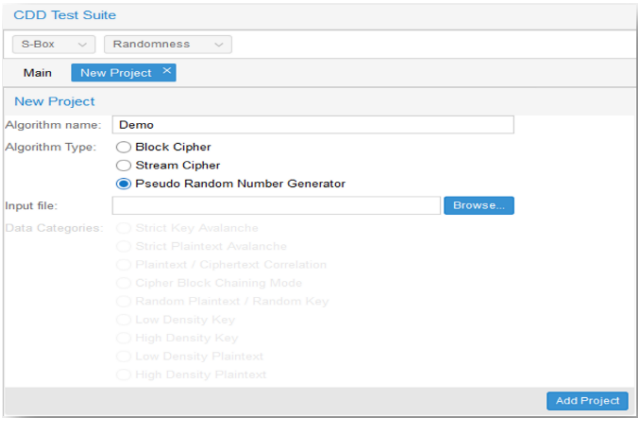


Figure 7: New Project Page. Name of the algorithm is entered, and algorithm type is ticked. The user requires to upload the test sample file. Add project when all the details are entered.

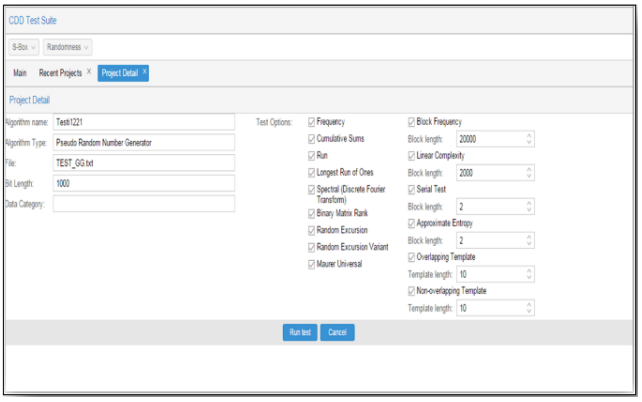


Figure 8: Project Detail Page. Tick on the desired statistical test and enter the parameters for parameterized test selection.

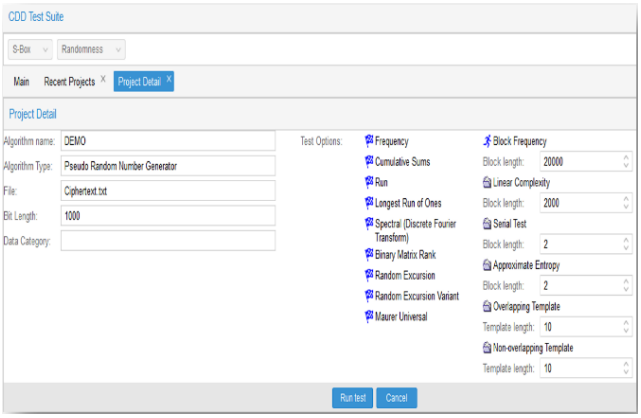


Figure 9: Queue Progress Page. Different icons appeared beside each test indicates that the test is either in queue, on-going or completed.

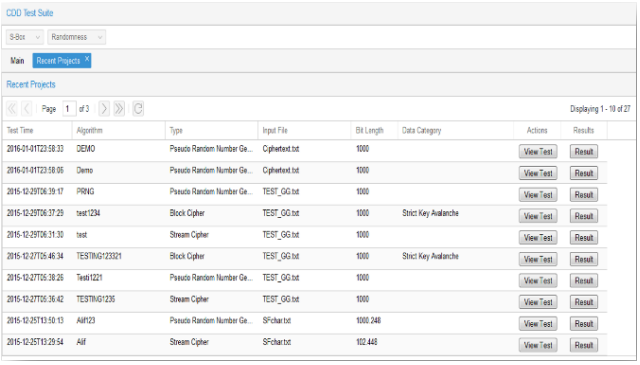
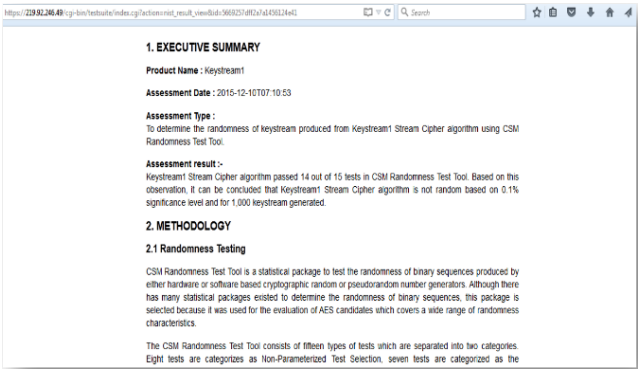


Figure 10: Recent Project Page. All of the tested projects are listed on this page. Click result button to view full randomness evaluation report of the cryptographic algorithm.



1. EXECUTIVE SUMMARY

Product Name : Keystream1

Assessment Date : 2015-12-10T07:10:53

Assessment Type :
To determine the randomness of keystream produced from Keystream1 Stream Cipher algorithm using CSM Randomness Test Tool.

Assessment result :-
Keystream1 Stream Cipher algorithm passed 14 out of 15 tests in CSM Randomness Test Tool. Based on this observation, it can be concluded that Keystream1 Stream Cipher algorithm is not random based on 0.1% significance level and for 1,000 keystream generated.

2. METHODOLOGY

2.1 Randomness Testing

CSM Randomness Test Tool is a statistical package to test the randomness of binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. Although there has many statistical packages existed to determine the randomness of binary sequences, this package is selected because it was used for the evaluation of AES candidates which covers a wide range of randomness characteristics.

The CSM Randomness Test Tool consists of fifteen types of tests which are separated into two categories. Eight tests are categorizes as Non-Parameterized Test Selection, seven tests are categorized as the

Figure 11: Randomness Evaluation Report. The results of the tested cryptographic algorithm are presented in this report.

5 CONFORMANCE ANALYSIS

Conformance testing assures that an implementation of a cryptographic mechanism is correct whether implemented in hardware, software or firmware. It also confirms that it runs correctly in a specific operating environment. Testing can consist of known-answer or Monte Carlo testing, or a combination of test methods. The testing can be performed on the actual implementation or modelled in a simulation environment. Thus, we implemented a conformance analysis to compare the testing tools implementation to make sure that all tests in the system are correct.

Table 4 shows ten randomness test results of the Grain-128 algorithm (Lot et al., 2014) produced using the NIST Statistical Test Suite compares to CRTT. We only display ten tests that produced one p-value for each test sample. The other tests that produced more than one p-value which are not displayed in the table are the Random Excursion Variant (Test 4; 18 p-values), Random Excursion (Test 5; 8 p-values), Cumulative Sums (Test 8; 2 p-values), Non-Overlapping Template (Test 10; 148 p-values), and Serial (Test 12; 2 p-values). Results produced by NIST and CRTT show similar output which confirms that CRTT implementation is correct and accurate.

Test No.	Tool	Sample									
		1	2	3	4	6	8	7	8	9	10
1	NIST	0.6542	0.2510	0.9968	0.3271	0.8415	0.0649	0.7054	0.4990	0.2027	0.2748
	CRTT	0.6542	0.2510	0.9968	0.3271	0.8415	0.0649	0.7054	0.4990	0.2027	0.2748
2	NIST	0.5457	0.3717	0.0615	0.6433	0.1178	0.1158	0.7719	0.1475	0.1177	0.9895
	CRTT	0.5457	0.3717	0.0615	0.6433	0.1178	0.1158	0.7719	0.1475	0.1177	0.9895
3	NIST	0.2829	0.3326	0.1321	0.2402	0.5568	0.6708	0.4753	0.3359	0.0077	0.1347
	CRTT	0.2829	0.3326	0.1321	0.2402	0.5568	0.6708	0.4753	0.3359	0.0077	0.1347
6	NIST	0.1790	0.0796	0.1521	0.3836	0.7275	0.4122	0.0093	0.8283	0.2481	0.1012
	CRTT	0.1790	0.0796	0.1521	0.3836	0.7275	0.4122	0.0093	0.8283	0.2481	0.1012
7	NIST	0.5549	0.9173	0.5248	0.9586	0.3177	0.2757	0.7212	0.0361	0.4875	0.3740
	CRTT	0.5549	0.9173	0.5248	0.9586	0.3177	0.2757	0.7212	0.0361	0.4875	0.3740
9	NIST	0.1716	0.7252	0.9877	0.6455	0.8913	0.3089	0.4461	0.5638	0.5707	0.4994
	CRTT	0.1716	0.7252	0.9877	0.6455	0.8913	0.3089	0.4461	0.5638	0.5707	0.4994
11	NIST	0.3351	0.2068	0.6114	0.1770	0.4020	0.7958	0.5566	0.0201	0.2214	0.9471
	CRTT	0.3351	0.2068	0.6114	0.1770	0.4020	0.7958	0.5566	0.0201	0.2214	0.9471
13	NIST	0.7331	0.3668	0.2816	0.6237	0.2851	0.1905	0.9909	0.6009	0.3940	0.6002
	CRTT	0.7331	0.3668	0.2816	0.6237	0.2851	0.1905	0.9909	0.6009	0.3940	0.6002
14	NIST	0.5963	0.9718	0.2244	0.9460	0.0048	0.2986	0.2133	0.3152	0.3083	0.1742
	CRTT	0.5963	0.9718	0.2244	0.9460	0.0048	0.2986	0.2133	0.3152	0.3083	0.1742
15	NIST	0.9976	0.5546	0.3519	0.0583	0.2586	0.7655	0.5273	0.1894	0.7549	0.0061
	CRTT	0.9976	0.5546	0.3519	0.0583	0.2586	0.7655	0.5273	0.1894	0.7549	0.0061

Table 4: Conformance Analysis. Comparison of test results produced by the NIST Statistical Test Suite and CRTT.

6 TIME PERFORMANCE ANALYSIS

The process of evaluating and producing a complete analysis report for the randomness of a cryptographic algorithm consists of three processes which are (1) run NIST randomness test, (2) analysis, and (3) writing report as shown in Figure 12. These three steps require different skill sets. Programming skills are needed to write the codes for testing purpose. Analytic thinking skills are necessary for making decisions and conclusions of the result. Meanwhile, language and writing skills are important to transform numbers (results) into sentences (conclusions) so that readers will understand the evaluation report.

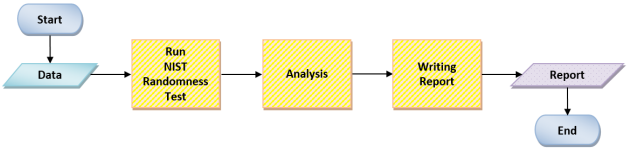


Figure 12: Conventional Randomness Evaluation Process. Users require to perform all of the three processes starting from running the NIST test suite, analyzing results, and complete it with writing report.

Without using the CSM Randomness Test Tool that we have developed, the evaluator has to go through all three steps separately for each tests module. For example, an evaluator requires running the test program, analyzing the result and writing report separately. By using the CSM Randomness Test Tool, the process of testing, analyzing and reporting can be managed simultaneously and automated. Table 5 presents the comparison of the evaluation process using CRTT and without using CRTT.

	Without CRTT	**With CRTT**
Run Tests	Consume a lot of time and requires a repeated process. This process also needs more resources for a certain test.	Reduce significant process time and requires limited action from the user. The process can be carried out by one person only.
Analyze Results	Prone to human error which could lead to an inaccurate result.	All results data are well managed and automatically compiled.
Report Production	Consume a lot of time and proper writing skills.	Generate an evaluation report automatically.

Table 5: Benefits of CRTT. Each process in testing the randomness of a cryptographic algorithm is compared to view the differences by using the conventional method and CRTT.

Table 6 displays the difference in time taken for each of the steps to produce a complete report using the CSM Randomness Test Tool and without using the tool to evaluate all 15 tests. The analysis was implemented on Intel(R) Core(TM) i7 2.70GHz CPU with 8 GB RAM on Windows 10. The results show CRTT takes approximately 3 hours to produce a complete analysis re-

port compared to 8 hours without using the tool. The time could be shorter if the user runs lesser than 15 available tests. In general, evaluation time may reduce up to 5 hours by using CRTT. With the presence of this tool, the process of evaluating the randomness of the cryptographic algorithm will be simplified and automated which resulted in reducing the evaluation period significantly.

Run Tests		Analyze Results		Report Production		Total	
Without CRTT	With CRTT	Without CRTT	With CRTT	Without CRTT	With CRTT	Without CRTT	With CRTT
3 hours	3 hours	1 hour	15 seconds	4 hours	5 seconds	8 hours	3 hours & 20 seconds

Table 6: Time Performance Analysis. A comparison of time taken for each process is recorded to distinguish the performance of the conventional evaluation method and CRTT.

7 CONCLUSION AND FUTURE WORK

With the presence of CRTT, the process of evaluating the randomness of a cryptographic algorithm will be simplified and automated which resulted in reducing the evaluation time frame. It takes approximately three hours to produce a complete randomness analysis report compared to eight hours without using the tool. This is one of our initiatives to improve the service of ICT Product Evaluation for Cryptographic Module provided by the Cryptography Development Department. This evaluation tool is the second in a series of tools that have been developed by the Cryptography Development Department to be used in the Cryptography Evaluation Laboratory. The first tool that has been developed was S-Box Evaluation Tools (CSET) in 2014 which has been used extensively by our Cryptographic Algorithm Development unit to enhance the performance of the CSM cryptographic algorithm (Zakaria et al., 2016). For CRTT and CSET service, please email crtt@cybersecurity.my. The Cryptography Development department of CyberSecurity Malaysia provides ICT Product Evaluation for Cryptographic Module services for the Malaysian

government and the private sector. Our future target is to promote the CSM Randomness Test Tool for targeted users such as University, Cryptography Researcher, and Cryptography Evaluator. The development of this tool is part of the preparation for the FIPS140-2 lab in CyberSecurity Malaysia and will also be used as a pre-evaluation tool for the National Cryptography Algorithm activity. This effort is in the line of making CyberSecurity Malaysia a pioneer player in the cryptography field as well as becoming the Cryptography Reference Centre in Malaysia.

REFERENCES

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

Brown, R. G., Eddelbuettel, D., and Bauer, D. (2013). Dieharder: A random number test suite. *Open Source software library, under development, URL http://www. phy. duke. edu/˜ rgb/General/dieharder. php*.

Caelli, W. et al. (1992). Crypt x package documentation. *Information Security Research Centre and School of Mathematics, Queensland University of Technology*.

Chen, T., Ma, Y., Lin, J., Wang, Z., and Jing, J. (2015). An Efficiency Optimization Scheme for the On-the-Fly Statistical Randomness Test. In *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pages 515–517. IEEE.

Dabal, P. and Pelka, R. (2012). An integrated system for statistical testing of pseudo-random generators in FPGA devices. In *2012 International Conference on Signals and Electronic Systems (ICSES)*, pages 1–5. IEEE.

Doğanaksoy, A., Sulak, F., Uğuz, M., Şeker, O., and Akcengiz, Z. (2015). New statistical randomness tests based on length of runs. *Mathematical Problems in Engineering*, 2015.

Knuth, D. E. (2014). *Art of computer programming, volume 2: Seminumerical algorithms*. Addison-Wesley Professional.

Lot, N. H., Zawawi, A., Seman, K., and Zaizi, N. J. M. (2014). A new proposed design of a stream cipher algorithm: Modified grain-128. *International Journal of Computer and Information Technology*, 3(5):902–908.

Marsaglia, G. (2008). The Marsaglia Random Number CD-ROM Including the Diehard Battery of Tests of Randomness.

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. Technical report, Booz-Allen and Hamilton Inc Mclean Va.

Suciu, A., Marton, K., Nagy, I., and Pinca, I. (2010a). Byte-oriented efficient implementation of the NIST statistical test suite. In *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, volume 2, pages 1–6. IEEE.

Suciu, A., Nagy, I., Marton, K., and Pinca, I. (2010b). Parallel implementation of the nist statistical test suite. In *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, pages 363–368. IEEE.

Suciu, A., Zegreanu, L., and Zima, C. (2009). Statistical testing of random number sequences using graphics processing units. In *2009 Fourth Balkan Conference in Informatics*, pages 39–43. IEEE.

Suresh, V. B., Antonioli, D., and Burleson, W. P. (2013). On-chip lightweight implementation of reduced nist randomness test suite. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 93–98. IEEE.

Zakaria, A. A., Abdullah, N. A. N., Zariman, W., Omar, N. A. M. Y., and Rani, H. A. (2016). Automated analysis report generation using csm s-box evaluation tool (cset). *International Journal of Cryptology Research*, 6(1):47–63.

Zawawi, N. H. L. A., Seman, K., and Mohd Zaizi, N. J. (2013). Randomness analysis on grain-128 stream cipher. In *AIP Conference Proceedings*, volume 1557, pages 15–20. AIP.

Zhu, S., Ma, Y., Lin, J., Zhuang, J., and Jing, J. (2016). 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*, pages 307–329. Springer.