Some results on new statistical randomness tests based on length of runs

Linh Hoang Dinh

Abstract— **Random Sequences and random numbers play a very important role in cryptography. In symmetric cryptography primitives, a secret key is the most important component to ensure their security. While cryptographic protocols or digital signature schemes are also strongly dependent on random values. In addition, one of the criteria for evaluating security for cryptographic primitives such as block cipher, hash function... is to evaluate the output randomness. Therefore, the assessment of randomness according to statistical tests is really important for measuring the security of cryptographic algorithms. In this paper, we present some research results on randomness tests based on the length of runs proposed by A. Doğanaksoy et al in 2015. First, we show that some probability values for tests based on lengths 1 and 2 are inaccurate and suggest editing. Secondly, we have given and demonstrated for the general case the runs of any length *k*. Finally, we built a randomness testing tool and applied evaluations to true random sources.**

Tóm tắt— **Các dãy và các số ngẫu nhiên đóng một vai trò rất quan trọng trong mật mã. Trong các nguyên thuỷ mật mã đối xứng, khoá bí mật chính là thành phần quan trọng nhất nhằm đảm bảo tính an toàn của chúng. Trong khi đó, các giao thức mật mã hay lược đồ chữ ký số cũng phụ thuộc nhiều vào các giá trị ngẫu nhiên. Ngoài ra, một trong các tiêu chí để đánh giá tính an toàn cho các nguyên thuỷ mật mã như mã khối, hàm băm… là đánh giá tính ngẫu nhiên đầu ra. Do đó, việc đánh giá tính ngẫu nhiên theo các kiểm tra thống kê thực sự rất quan trọng đối với việc đánh giá tính an toàn của các thuật toán mật mã. Trong bài báo này, chúng tôi trình bày một số kết quả nghiên cứu về các tiêu chuẩn kiểm tra loạt dựa trên độ dài đã được đề xuất bởi A. Doğanaksoy cùng đồng sự năm 2015. Đầu tiên, chúng tôi chỉ ra rằng một số giá trị xác suất cho các loạt độ dài 1 và 2 là chưa chính xác và đề xuất chỉnh sửa. Sau đó, chúng tôi đã đưa ra và chứng minh cho trường hợp tổng quát các loạt có độ dài *k* bất kỳ. Cuối cùng, chúng tôi đã xây dựng một công cụ kiểm tra tính ngẫu nhiên dựa trên độ dài các loạt và áp dụng đánh giá cho các nguồn ngẫu nhiên thực sự.**

Keywords— **Randomness testing; Block cipher; Hash function; Runs.**

Từ khóa— **Kiểm tra tính ngẫu nhiên; Mã khối; Hàm băm; Loạt.**

I. INTRODUCTION

Statistical randomness tests play an important role in assessing the security of cryptographic algorithms. There have been many independently statistical randomness tests in the literature. Knuth [1] presented a number of statistical tests including frequency check, serial test, poker test, series test (run)... Another test suite is the DIEHARD tests [2] including 18 statistical tests. In addition, there is a Crypt-XS test suite [3] proposed by the Information Security Research Center of Queensland University of Technology. Finally, the currently widely used test suite is the SP 800-22 statistical test suite [4] originally developed by NIST with 16 tests but then shortened to 15 tests (omitted Lempel-Ziv complexity test).

In addition, there are a number of randomness testing standards that are not presented in test suites or independently used. In 1992 Maurer proposed a universal statistical test for random bit generators. In 2004, Hernandez et al. proposed a new test called the Strict Avalanche Criterion (SAC)... And most recently, Doğanaksoy et al. [5] proposed three new randomness tests based on the length of runs in 2015.

Our Contributions. In this paper, we present some new results on randomness tests based on the length of runs. Specifically, we have given and demonstrated in detail the probability calculation formula for the general case of a binary sequence of length n having exactly l_k runs of length k, with $1 \le k \le n$. Furthermore, we have shown that some

This manuscript is received on December 1, 2018. It is commented on December 6, 2018 and is accepted on December 12, 2018 by the first reviewer. It is commented on December 16, 2018 and is accepted on December 22, 2018 by the second reviewer.

probability values given in [5] are inaccurate, and this may lead to a mistake in assessing the randomness of the input sequences, thereby giving to incorrect assertions about the security of cryptographic algorithms. Finally, we built a tool that quickly, accurately and efficiently checks a data file that is random or not using three new randomness tests based on length of runs.

Construction. The rest of the paper includes: Part II presents three postulates on randomness given by Golomb [6] as well as some tests related to runs. The new results of research on tests based on length of run are presented in section III. In section IV, we present a randomness assessment tool using 3 new tests. Finally, the conclusions and future research directions are presented in Section V.

II. PRELIMINARIES

In this section, we present the three propositions of randomness given by Golomb in [6]. This is one of the bases for assessing the randomness of a sequence. Then, we outlined some of the testing standards related to the runs as well as the reasons for studying new standards based on length of runs.

A. Golomb's Randomness Postulates

Let $S = s_0, s_1, \cdots, s_{n-1}, \cdots$ be an infnite binary sequence periodic with *n* or a finite sequence of length *n*. A run is defined as an uninterrupted maximal sequence of identical bits. Runs of 0's are called *gap*; runs of 1's are called *block*. R1, R2, and R3 are Golomb's randomness postulates which are given as follows:

- (R1) In a period of *S*, the number of 1's should differ from the number of 0's by at most 1. In other words, the sequence should be balanced.
- (R2) In a period of *S*, at least half of the total number of runs of 0's or 1's should have length one, at least onefourth should have length 2, at least one-eighth should have length 3, and the like. Moreover, for each of these lengths, there should be (almost) equally many gaps and blocks.
- (R3) The autocorrelation function $C(t)$ should be two-valued. That is, for some integer *K* and for all $t = 0,1,\cdots,n-1$,

$$C(t) = \sum_{i=0}^{n-1} (-1)^{s_i + s_{i+t}} = \begin{cases} n, t = 0 \\ K, 1 \le t \le n-1. \end{cases}$$

In this paper, we mainly focus on the first and second postulates, and the last one is not a matter of concern.

B. Some basic run tests

Golomb's second postulate is on the number of runs in a sequence. Tests which consider the number of runs, are called run tests and are also included in many test suites such as Knuth [1], DIEHARD [2], TestU01 [7], NIST [4]. Since calculating the expected number of fixed-length runs in a random sequence is a difficult task (especially when the length of runs is large), most tests only consider the total number of runs and do not consider the number of runs of different lengths as follows:

Run tests in Knuth* [1] *and DIEHARD* [2] *test suites:

These test suites define the run test on random numbers. They define runs as *runs up* and *runs down* in a sequence. For example, consider a sequence of length 10, $S_{10} = 138742975349$. Runs are indicated by putting a vertical line between s_j' when $s_j > s_{j+1}$. Therefore, runs of the sequence 138742975349 can be seen as $|138|7|4|29|7|5|349|$. In other words, the run test examines the length of monotone subsequences.

Run test in TestU01* [7] *test suite:

This test suite defines runs and gap tests for checking the randomness of long binary sequences of length *n*. This test calculates the runs of 1 and 0 until the total number of runs is $2r$. Next, number of runs 1 and 0 correspond to each length of $j = 1,2,\cdots,k$ is calculated and recorded. Finally, applying χ^2 test on these values. Test of the longest run of one also be defined for subsequences of length *m* that obtained from the original binary sequence of length *n*.

Run test in NIST* [4] *test suite:

NIST test suite is widely used to checks the randomness of pseudorandom sequences. In the suite, 2 of 15 tests are variations of run tests. They are called run test and longest run of ones in a block test. The first one deals with

the total number of runs in a sequence. It calculates the total number of runs in a sequence and determines whether it is consistent with the expected number of runs, which is supposed to be close to $n/2$ in a sequence or not. The second one determines whether the longest run of ones in the sequence is consistent with the length of the longest runs of ones which is in a random sequence. In NIST test suite the reference distributions for the run tests are a χ^2 distribution.

III. TESTS BASED ON LENGTH OF RUNS

In this section, we present the theoretical and practical basis for calculating the number of sequences with a certain number of runs of specific lengths. Then we calculate the expected probability from the total number of sequences of length *n*. Calculating the exact probabilities of the number of runs of length *k* (for $1 \le k \le n$) in a sequence allows us to define new runs tests. However, when the runs are of great length, the computational complexity increases exponentially so it is not feasible to accurately calculate probability values.

A. Notations

Let's denote the total number of runs and number of runs of lengths one, two,…, and *k* as $r_t, r_1, r_2, \ldots$ and r_k ans we use samples of these variables $r, l_1, l_2, ..., l_k$ for $1 \le k \le n,$ respectively. We denote the probability of randomly chosen binary sequence with *r* runs by $\Pr(r_t = r)$. In the same way, $\Pr(r_k = l_k)$ is the probability of randomly chosen binary sequence with l_k runs of length *k* for $1 \le k \le n$.

We denote $S_1, S_2, \cdots, S_m$ be the blocks of a long sequence or outputs of block ciphers and hash functions.

Lastly, let denote L_1, L_2, … and L_k be the set of number of runs of lengths one, two, … and *k* in the sequences for $1 \le k \le n$, respectively. That is $L_i = \{l_i^1, l_i^2, \cdots, l_i^m\}$ and l_i^j corresponds the number of runs of length *i* in the *j* sequence.

B. Evaluation of Probabilities

In the calculations of probabilities we use the following Propositions:

Proposition 1 (Fact 2, [5]) The number of positive integer solutions of $x_1 + x_2 + \cdots + x_r = n, n \in \mathbb{Z}^+$ is $\binom{n-1}{r-1}$.

Proposition 2 (Fact 1, [5]) The number of nonnegative integer solutions of $x_1 + x_2 + \cdots + x_r = n, n \in \mathbb{Z}^+$ is $\binom{n+r-1}{r-1}$.

Moreover, in order to illustrate the runs of a sequence we use the equation $x_1 + x_2 + \cdots + x_r = n$ for a sequence with length *n* and having *r* runs. $x_i \left(i = 1, 2, \cdots, r\right)$ represents the number of bits in i^{th}. An important property of this illustration is that it gives no information about content of x_i, s; that is x_i can be a run of 0's or 1's. Therefore, each positive integer solution of the equation $x_1 + x_2 + \cdots + x_r = n$ corresponds to two sequences: one starts with 1 and the other starts with 0. Thus, the number of sequences with length *n* and having exactly *r* runs is $2\binom{n-1}{r-1}$ by Proposition 1.

Example 1.

Let $S = 00101010011111001100011101010100$ be a binary sequence of length 32 and having 19 runs. Then,

$$x_1 + x_2 + \cdots + x_{19} = 32,$$

$$\underbrace{00}_{x_1}\underbrace{1}_{x_2}\underbrace{0}_{x_3}\underbrace{1}_{x_4}\underbrace{0}_{x_5}\underbrace{1}_{x_6}\underbrace{00}_{x_7}\underbrace{11111}_{x_8}\underbrace{00}_{x_9}\underbrace{11}_{x_{10}}\underbrace{000}_{x_{11}}\underbrace{111}_{x_{12}}\underbrace{0}_{x_{13}}\underbrace{1}_{x_{14}}\underbrace{0}_{x_{15}}\underbrace{1}_{x_{16}}\underbrace{0}_{x_{17}}\underbrace{1}_{x_{18}}\underbrace{00}_{x_{19}},$$

$$\begin{cases} x_1 = 2, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 1, \\ x_6 = 1, x_7 = 2, x_8 = 5, x_9 = 2, x_{10} = 2, \\ x_{11} = 3, x_{12} = 3, x_{13} = 1, x_{14} = 1, x_{15} = 1, \\ x_{16} = 1, x_{17} = 1, x_{18} = 1, x_{19} = 2. \end{cases}$$

Now, we consider the case that a sequence of length *n* and having exactly total *r* runs, l_1 of which are runs of length 1, l_2 of which are runs of length 2, …, l_k of which are runs of length *k*. We have the following Theorem:

Theorem 1. The probability of randomly chosen binary sequence $S = s_1, s_2, \cdots, s_n$ with length *n*, having total of *r* runs, l_1 of which are

runs of length 1, l_2 of which are runs of length 2, …, l_k of which are runs of length *k*, is

$$\Pr\left(r_t = r, r_1 = l_1, ..., r_k = l_k\right) = \begin{pmatrix} n - kr + \left(k-1\right)l_1 + \left(k-2\right)l_2 + ... + 2l_{k-2} + l_{k-1} - 1 \\ r - l_1 - l_2 - ... - l_k - 1 \end{pmatrix} \times \times \frac{\begin{pmatrix} r \\ l_1 \end{pmatrix}\begin{pmatrix} r-l_1 \\ l_2 \end{pmatrix}...\begin{pmatrix} r-l_1-l_2-...-l_{k-1} \\ l_k \end{pmatrix}}{2^{n-1}}.$$

Proof. We can illustrate the sequence as follow:

$$x_1 + x_2 + \cdots + x_r = n.$$

Let us first assume that the last l_1 are of length 1 and l_2 are of length 2, …, and l_k are of length *k*. The rest are of at least length $\left(k+1\right)$. Consider,

$$x_{r-l_1+1} = \cdots = x_{r-1} = x_r = 1,$$
$$x_{r-l_1-l_2+1} = \cdots = x_{r-l_1-1} = x_{r-l_1} = 2,$$
$$...$$
$$x_{r-l_1-l_2-...-l_k+1} = \cdots = x_{r-l_1-l_2...-l_{k-1}-1} = x_{r-l_1-l_2-...-l_{k-1}} = k.$$

Substituting these into the above equation, we have:

$$x_1 + x_2 + \cdots + x_{r-\left(l_1+l_2+...+l_k\right)} + \overbrace{k + k + \cdots + k}^{l_k} + ... + \overbrace{2 + 2 + \cdots + 2}^{l_2} + \overbrace{1 + 1 + \cdots + 1}^{l_1} = n,$$
$$\Leftrightarrow \quad x_1 + x_2 + \cdots + x_{r-\left(l_1+l_2+...+l_k\right)} = n - l_1 - 2l_2 - ... - kl_k.$$

Note that, $x_i \geq k+1,$ for $1 \leq i \leq r - \left(l_1 + l_2 + ... + l_k\right)$.

Set $y_i = x_i - \left(k+1\right)$ for $1 \leq i \leq r - \left(l_1 + l_2 + ... + l_k\right)$, and substituting into the above equation, we have:

$$\left(y_1 + k + 1\right) + \left(y_2 + k + 1\right) + \cdots + \left(y_{r-\left(l_1+l_2+...+l_k\right)} + k + 1\right) = n - l_1 - 2l_2 - ... - kl_k,$$
$$\Leftrightarrow y_1 + y_2 + \cdots + y_{r-\left(l_1+l_2+...+l_k\right)} = n - \left(l_1 + 2l_2 + ... + kl_k\right) - \left(k+1\right)\left[r - \left(l_1 + l_2 + ... + l_k\right)\right]$$
$$= n - \left(k+1\right)r + kl_1 + \left(k-1\right)l_2 + ... + l_k.$$

Applying the Proposition 2, number of nonnegative solution of this equation is:

$$\begin{pmatrix} n - kr + \left(k-1\right)l_1 + \left(k-2\right)l_2 + ... + l_{k-1} - 1 \\ r - l_1 - l_2 - ... - l_k - 1 \end{pmatrix}.$$

Therefore, the number of all binary sequences of length *n* with conditions stated above is:

$$2\begin{pmatrix} n - kr + \left(k-1\right)l_1 + \left(k-2\right)l_2 + ... + l_{k-1} - 1 \\ r - l_1 - l_2 - ... - l_k - 1 \end{pmatrix} \times \begin{pmatrix} r \\ l_1 \end{pmatrix}\begin{pmatrix} r-l_1 \\ l_2 \end{pmatrix}...\begin{pmatrix} r-l_1-l_2-...-l_{k-1} \\ l_k \end{pmatrix}.$$

Hence, the probability of a randomly chosen sequence with length *n*, having total of *r* runs, l_1 of which are runs of length 1, l_2 of which are runs of length 2, …, l_k of which are runs of length *k*, is:

$$\Pr\left(r_t = r, r_1 = l_1, ..., r_k = l_k\right) = \begin{pmatrix} n - kr + \left(k-1\right)l_1 + \left(k-2\right)l_2 + ... + 2l_{k-2} + l_{k-1} - 1 \\ r - l_1 - l_2 - ... - l_k - 1 \end{pmatrix} \times \times \frac{\begin{pmatrix} r \\ l_1 \end{pmatrix}\begin{pmatrix} r-l_1 \\ l_2 \end{pmatrix}...\begin{pmatrix} r-l_1-l_2-...-l_{k-1} \\ l_k \end{pmatrix}}{2^{n-1}} \quad \square$$

Now, we evaluate the number of sequences with length *n* and l_k runs of length *k*, without depending on the other variables. We have the following corollary:

Corollary 1. Let $N_k\left(l_k\right)$ denote the number of sequences with length *n* and having exactly l_k runs of length *k*. Clearly, we have maximum $\left\lfloor n/k \right\rfloor$ runs of length *k*. Otherwise, sequence length exceeds *n*. Then, for $l_k = 0, 1, \cdots, \left\lfloor n/k \right\rfloor$,

$$N_k\left(l_k\right) = \sum_{l_{k-1}=0}^{\left\lfloor n/(k-1) \right\rfloor} \cdots \sum_{l_1=0}^{n} \sum_{r=1}^{n} 2\begin{pmatrix} n - kr + \left(k-1\right)l_1 + ... + l_{k-1} - 1 \\ r - l_1 - l_2 - ... - l_k - 1 \end{pmatrix} \times \begin{pmatrix} r \\ l_1 \end{pmatrix}\begin{pmatrix} r-l_1 \\ l_2 \end{pmatrix}...\begin{pmatrix} r-l_1-...-l_{k-1} \\ l_k \end{pmatrix}.$$

It implies the probabilities

$$\Pr\left(r_k = l_k\right) = \frac{N_k\left(l_k\right)}{2^n}.$$

We use the Algorithm 1 in order to evaluate the probabilities $\Pr\left(r_k = l_k\right)$.

Algorithm 1: Evaluate $\Pr\left(r_k = l_k\right)$ for $l_k = 0,1,\cdots,\lfloor n/k \rfloor$

$l_k \leftarrow 0,\cdots,l_2 \leftarrow 0, l_1 \leftarrow 0, r \leftarrow 1, N_k\left(l_k\right) \leftarrow 0,$

while $l_k \le \lfloor n/k \rfloor$ do

…

while $l_2 \le \lfloor n/2 \rfloor$ do

while $l_1 \le n$ do

while $r \le n$ do

$$N_k\left(l_k\right) \leftarrow N_k\left(l_k\right) + \frac{1}{2^{n-1}} \times \sum_{l_{k-1}=0}^{\lfloor n/(k-1) \rfloor} \cdots \sum_{l_1=0}^{n} \sum_{r=1}^{n} \binom{n - kr + \left(k-1\right)l_1 + \ldots + l_{k-1} - 1}{r - l_1 - l_2 - \ldots - l_k - 1} \times \binom{r}{l_1}\binom{r-l_1}{l_2}\cdots\binom{r-l_1-\ldots-l_{k-1}}{l_k}$$

$r \leftarrow r+1$

end while

$l_1 \leftarrow l_1 + 1$

end while

$l_2 \leftarrow l_2 + 1$

end while

…

$l_k \leftarrow l_k + 1$

end while

return N_k

Computational Complexity: Let the complexity of computing $N_k\left(l_k\right)$ be $\tau\left(k\right)$ then the complexity of probability searching algorithm for runs of length k is about $\mathcal{O}\left(n^{k+1}\tau\left(k\right)\right)$.

After evaluating all probability values, we divide these into 5 subintervals as in [5].

Case $k = 1$.

Choose $n = 128$, we have calculated all probability values and divide into subintervals as follow

$$Box1 = \sum_{l_1=0}^{27} \Pr_1\left(r_1 = l_1\right), \quad Box2 = \sum_{l_1=28}^{31} \Pr_1\left(r_1 = l_1\right),$$

$$Box3 = \sum_{l_1=31}^{34} \Pr_1\left(r_1 = l_1\right), \quad Box4 = \sum_{l_1=35}^{38} \Pr_1\left(r_1 = l_1\right),$$

$$Box5 = \sum_{l_1=39}^{128} \Pr_1\left(r_1 = l_1\right).$$

Then, we get Table 1 for probability subintervals.

Table 1. Interval and probability values for runs of length one test for 128-bit blocks

	Interval	Probability
Box 1	0-27	0.2191945278
Box 2	28-31	0.2304573984
Box 3	32-34	0.1843489091
Box 4	35-38	0.1945435197
Box 5	39-128	0.1714556450
Total		**1**

Remark 1: *In the Table 1, we have use the intervals given in [5], however the calculated probabilities of Box 4 and Box 5 are not match with the probabilities given in [5]. After retesting, we find that the authors in [5] give correct intervals but wrong probabilities. The correct probabilities are as in Table 1. Moreover, the probabilities given in [5] are belong to the intervals 35-40, and 41-128, that can not belong to the intervals 35-38 and 39-128.*

Similarly, we can calculate probabilitiy intervals for sequences with different lengths. The subinterval probabilities for runs of length 1 can be seen in Table 2.

Table 2. Interval and probability values for runs of length one test for 64, 128, 256, and 512-bit blocks

	***n* = 64**		***n* = 128**	
	Interval	Probability	Interval	Probability
Box 1	0-12	0.1900823444	0-27	0.2191945278
Box 2	13-15	0.2388774637	28-31	0.2304573984
Box 3	16-17	0.1745603741	32-34	0.1843489091
Box 4	18-20	0.2114704014	35-38	0.1945435197
Box 5	21-64	0.1850094164	39-128	0.1714556450
Total		**1**		**1**
	***n* = 256**		***n* = 512**	
	Interval	Probability	Interval	Probability
Box 1	0-56	0.1872558409	0-117	0.1935663836
Box 2	57-61	0.1892809185	118-125	0.2186301142
Box 3	62-66	0.2198594518	126-132	0.2170766790
Box 4	67-72	0.2187759227	133-140	0.1995155492
Box 5	73-256	0.1848278661	141-512	0.1712112740
Total		**1**		**1**

Remark 2: In [5], the authors give the intervals and the corresponding probabilities for runs of length 1. However, we found that some value in [5] are incorrect! For $n = 64$, if we use the intervals in [5], then the correct probabilities should be:

Table 2.1. Interval and probability values for runs of length 1 test for 64-bit blocks

	n = 64	
	Interval	Probability
Box 1	0-13	0.2613425337
Box 2	14-16	0.2561417553
Box 3	17-18	0.1659176815
Box 4	19-21	0.1812433426
Box 5	22-64	0.1353546869
Total		**1**

These probabilities are not match with the probabilities given in [5]. Moreover, these intervals are not equivalent. Therefore, we have re-divide into new intervals and recalculate probabilities in new intervals. Interestingly, these probability values approximate the values given in [5] but belong to other intervals.

Similar to the case $n = 128$, intervals and the probability values given in [5] is not match. Specifically, if we take the given intervals in [5], we have recalculated the probabilities exactly as shown in Table 1 and Table 2. If we use the intervals as follows, the probability values coincide with the values given in [5].

Table 2.2. Interval and probability values for runs of length 1 test for 128-bit blocks

	n = 128	
	Interval	Probability
Box 1	0-26	0.1731718548
Box 2	27-30	0.2142651725
Box 3	31-33	0.1869770204
Box 4	34-37	0.2133929800
Box 5	38-128	0.2121929722
Total		**0.9999999999**

In case $k = 2$, we have calculated all probability values and divide into subintervals as follow:

Table 3. Interval and probability values for runs of length two test for 64, 128, 256, and 512-bit blocks

	***n* = 64**		***n* = 128**	
	Interval	Probability	Interval	Probability
Box 1	0-5	0.1613445444	0-12	0.1670758001
Box 2	6-7	0.2609640039	13-14	0.1740756080
Box 3	8	0.1490939694	15-16	0.2097940761
Box 4	9-10	0.2452877567	17-19	0.2665905046
Box 5	11-32	0.1833097255	20-64	0.1824640111
Total		**0.9999999999**		**0.9999999999**
	***n* = 256**		***n* = 512**	
	Interval	Probability	Interval	Probability
Box 1	0-27	0.1925793149	0-57	0.1889384182
Box 2	28-30	0.1940519689	58-61	0.1787949730
Box 3	31-33	0.2229235033	62-65	0.2104962769
Box 4	34-36	0.1878533079	66-70	0.2256155170
Box 5	37-128	0.2025919051	71-256	0.1961548150
Total		**1.0000000001**		**1.0000000001**

In case $k = 3$, we have calculated all probability values and divide into subintervals as follow:

Table 4. Interval and probability values for runs of length three test for 64, 128, 256, and 512-bit blocks

	***n* = 64**		***n* = 128**	
	Interval	Probability	Interval	Probability
Box 1	0-2	0.2078250899	0-5	0.1632090084
Box 2	3	0.2043198109	6-7	0.2745007990
Box 3	4	0.2167320408	8	0.1548547229
Box 4	5-6	0.2832454760	9-10	0.2450590118
Box 5	7-21	0.0878775825	11-42	0.1623764579
Total		**1.0000000001**		**1.0000000000**
	***n* = 256**		***n* = 512**	
	Interval	Probability	Interval	Probability
Box 1	0-13	0.2487347584	0-27	0.1898526054
Box 2	14-15	0.2071647158	28-30	0.2014971073
Box 3	16-17	0.2137437681	31-33	0.2311816473
Box 4	18-20	0.2221445069	34-36	0.1900066126
Box 5	20-85	0.1082122508	37-170	0.1874620274
Total		**1.0000000000**		**1.0000000000**

Remark 3: *In the case of* $n = 512$ *we used Magma software to divide the intervals and calculate the probability values because it takes quite a long time to run in C ++ language. The calculation time on Magma for this case is about 5000 seconds.*

C. Tests Descriptions

After calculating the probabilities, we begin to build a new test based on the number of runs of length *k*. Specifically, to test a sequence of $N = n \times m$ bits, where *n* is the block size we choose. Or we consider *m* outputs of a cryptographic primitive (a block cipher or a hash function) that have output block size is *n*-bit. First, we'll count the number of runs of length *k* of each sequence in *m* blocks, and increases the count value of the corresponding sub-interval to 1. After calculating, we record the counting values of each sub-interval, denoted by $F_1, F_2, \cdots, F_5$ respectively. We use the approach as in [8], using χ^2 test to evaluate the randomness of the sequence.

Consider:

$$\chi^2 = \sum_{i=1}^{5} \frac{\left(F_i - m \Pr_i\right)^2}{m \Pr_i},$$

Lastly p-value is calculated according to the given values:

$$p - value = igamc\left(\frac{5-1}{2}, \frac{\chi^2}{2}\right).$$

By comparing the produced p-value with the level of significance α, we can conclude about the randomness of the input sequence.

Note that for χ^2 test, we require $m \Pr_i \geq 5$ therefore for $\Pr_i \approx 0.2$ we need $m \geq 25$. Thus, the new tests can be applied for short sequences of length $N = n \times m$ bit for $m \geq 25$.

In addition, counting the total runs numbers and runs numbers with length k of a sequence by definition is difficult. Therefore, we use the concept of "derivative" of a sequence.

Definition 1 (Remark 11, [5]) (derivative of a sequence) Let $S = s_0, s_1, \cdots, s_{n-1}$ be a binary sequence of length *n*, the derivative of *S*, denoted by $\Delta S = \Delta s_0, \Delta s_1, \cdots, \Delta s_{n-1}$ is defined as follows

$$\Delta s_i = \begin{cases} s_i \oplus s_{i+1} & \text{if } i = 0,1,\cdots,n-2, \\ 1 & \text{if } i = n-1. \end{cases}$$

Also we use a variation of ΔS, denoted by $\Delta S'$ of length $n+1$ by adding 1's at the beginning the sequence ΔS. The variation of derivative is an important part of new defned run tests, since the number of runs of different length is determined by this sequence.

Let $S = s_0, s_1, \cdots, s_{n-1}$ be a binary sequence and derivative of *S* is $\Delta S = \Delta s_0, \Delta s_1, \cdots, \Delta s_{n-1}$.

Then, $\Delta S' = \Delta s'_0, \Delta s'_1, \cdots, \Delta s'_n$ is defined as follows

$$\Delta s'_i = \begin{cases} \Delta s_{i-1} & \text{if } i = 1,2,\cdots,n, \\ 1 & \text{if } i = 0. \end{cases}$$

It is easy to prove that the total runs number of a sequence is the weight of the derivative of that sequence, and the runs number with the length *k* in a given sequence *S* is the number of samples $1\underbrace{0\cdots0}_{(k-1)}1$ overlaps in $\Delta S'$.

Algorithm 2 presents the pseudocode of the test based on the number of runs of length *k*:

Algorithm 2: Test based on the number of runs of length *k* $\left(S_1, S_2, \cdots, S_m\right), L_k = \left\{l_k^1, l_k^2, \cdots, l_k^m\right\}$

$$\Delta S'_i = \Delta s'_{i,0}, \Delta s'_{i,1}, \cdots, \Delta s'_{i,n}$$

$$i \leftarrow 0, l_k^i \leftarrow 0$$

while $j \leq n - k$ do

$$temp = \Delta s'_{i,j} \times 2^k + \Delta s'_{i,j+1} \times 2^{k-1} + \Delta s'_{i,j+2} \times 2^{k-2} + \cdots + \Delta s'_{i,j+k} \times 2^0$$

if $temp = 2^k + 1$ then

$$l_k^i \leftarrow l_k^i + 1$$

end if

$$i \leftarrow i + 1$$

end while

Applying χ^2 test for L_k

return p-value.

IV. SOME EXPERIMENTAL RESULTS

We have developed a randomness test program using tests based on runs of lengths 1, 2 and 3. The program interface is shown in Figure 1. Specifically, we have tested 4 files true random: sample1.rng, sample2.rng, sample3.rng, sample4.rng (these true random files are actually 32 KB in size, that is, bits of $N = 32 \times 1024 \times 8$ length, downloaded at http://www.rngresearch.com/download/) with the following cases *n* = 64, 128, 256 and 512. The results of all 4 files have passed 3 new runs

tests with n = 64, 128, 256 and 512. Specifically, for the case n = 64, select the significance level $\alpha = 0.01$, the file to be checked is sample1.rng, we get the result as shown in Figure 1:

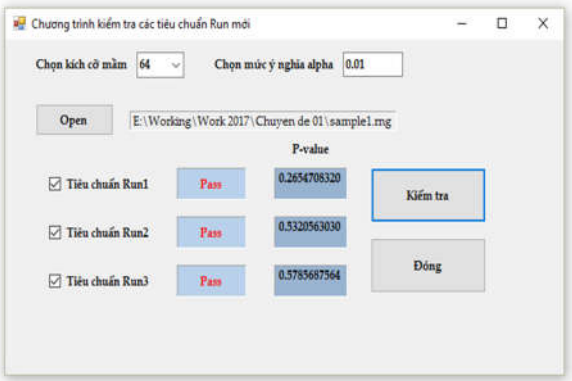

Fig 1. The program interface of 3 new runs tests for n = 64, $\alpha = 0.01$ for file sample1.rng

Similarly, we perform tests for the cases $n = 128, 256, 512$ and for files sample2.rng, sample3.rng, sample4.rng. The results are summarized in the following Table 5:

Table 5. Results of 3 new runs tests for true random files

Case n = 64				
	sample 1.rng	sample2.rng	sample3.rng	sample4.rng
Runs of length 1 test	0.265471	0.177239	0.249560	0.857602
Runs of length 2 test	0.532056	0.054239	0.509319	0.219101
Runs of length 3 test	0.578569	0.832500	0.445590	0.941098
Case n = 128				
	sample 1.rng	sample2.rng	sample3.rng	sample4.rng
Runs of length 1 test	0.089778	0.601941	0.251491	0.941470
Runs of length 2 test	0.169505	0.435659	0.645554	0.416198
Runs of length 3 test	0.264185	0.893517	0.393173	0.978088
Case n = 256				
	sample 1.rng	sample2.rng	sample3.rng	sample4.rng
Runs of length 1 test	0.640939	0.308548	0.272620	0.422990
Runs of length 2 test	0.899293	0.231489	0.571770	0.779767
Runs of length 3 test	0.506332	0.814081	0.011770	0.591287
Case n = 512				
	sample 1.rng	sample2.rng	sample3.rng	sample4.rng
Runs of length 1 test	0.120585	0.292832	0.480338	0.861397
Runs of length 2 test	0.252212	0.821268	0.105730	0.726579
Runs of length 3 test	0.811172	0.471682	0.110607	0.834620

V. CONCLUSION

In this paper, we present some results on new randomness tests based on length of run proposed by A. Doğanaksoy et al. [5]. First, we have given and demonstrated in detail the probability calculation formula for runs of length k, with $1 \le k \le n$. Second, we show that some probability values for runs lengths 1 and 2 are inaccurate and suggest corrections. Third, we have built a randomness testing algorithm based on the length of runs. Finally, we programmed to build an accurate and efficient tool to test randomness based on the length of runs and apply evaluations to true random sources.

Further research directions: Note that the criteria presented in this paper can only be used to evaluate sequences of lengths greater than 512 bits, so it is not applicable to assess randomness output for cryptographic primitives such as block ciphers or hash functions. To be able to evaluate for sequences of length less than or equal to 512 bits, we need to recalculate the probability distribution for blocks of smaller lengths and divide the probability interval accordingly. This is an open problem that needs further research in the future. In addition, the evaluation of probability values for series with a length greater than or equal to 4 and the correlation between these tests also need further consideration and research.

REFERENCES

[1]. M. D. MacLaren, "The art of computer programming. Volume 2: Seminumerical algorithms (Donald E. Knuth)", *SIAM Review* 12, pp. 306-308, 1970.

[2]. G. Marsaglia, "The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995". *URL http://www. stat. fsu. edu/pub/diehard*, 2008.

[3]. W. Caelli, "Crypt x package documentation". *Information Security Research Centre and School of Mathematics, Queensland University of Technology*, 1992.

[4]. A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, J. Dray, S. Vo, "Statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST special publication", 2010.

[5]. A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, Z. Akcengiz, "New statistical randomness tests based on length of runs". *Mathematical Problems in Engineering*, 2015.

[6]. S. W. Golomb, "*Shift register sequences"*. Aegean Park Press, 1982.

[7]. P. L'Ecuyer, R. Simard, "TestU01: AC library for empirical testing of random number generators". *ACM Transactions on Mathematical Software (TOMS)* 33, 22, 2007.

[8]. F. Sulak, A. Doğanaksoy, B. Ege, O. Koçak, "Evaluation of randomness test results for short sequences", in *International Conference on Sequences and Their Applications*. Springer, pp. 309-319, 2010.

ABOUT THE AUTHOR

B.S. Linh Hoang Dinh

Workplace: Institute of Cryptography Science and Technology.

Email: linhhd@bcy.gov.vn

The education process: has received a mathematical bachelor degree in Ha Noi University of Science, in 2014.

Research today: symmetric cryptography algorithm.