Iraqi Journal of Science, 2024, Vol. 65, No.1, pp: 346- 357
DOI: 10.24996/ijs.2024.65.1.28

ISSN: 0067-2904

Randomness of Stream Cipher Digital Sequences in the Field GF(2^8)

Adnan Muhammad Ali[1], Faez Hassan Ali[2*], Sabah Manfi Redha[3]
[1,2]Mathematics department, Collage of Science, Mustansiriyah University, Baghdad, Iraq
[3]Statistic department, Collage of Administration and Economics, University of Baghdad, Baghdad, Iraq

Received: 7/10/2022 Accepted: 12/32/2023 Published: 30/1/2024

Abstract

A binary stream cipher cryptosystem can be used to encrypt/decrypt many types of digital files, especially those can be considered huge data files like images. To guarantee that the encryption or decryption processes need a reasonable time to encrypt/decrypt the images, so we have to make the stream cipher key generator that acts quickly without effect in the complexity or randomness of the output key binary sequences. In this paper, we increase the size of the output sequence from binary to digital sequence in the field $GF(2^8)$ to obtain byte sequence, then we test this new sequence not only as binary but also 2^8-sequence. So we have to test the new output sequence in the new mathematical field. This is done by changing the base of the randomness tests and extending Golomb's postulates from binary to 256. Some theorems and lemmas are proved to find the new testing laws that are suitable to the new sequences field. The results of using the extended randomness tests are compared to the results of binary randomness tests to guarantee that the decision of pass or fail is identical, the results also prove the precision of identicalness.

Keywords: Randomness, Stream Cipher, Basic Efficiency Criteria, digital Sequences.

عشوائية متتابعات التشفير الانسيابي الرقمية في حقل غالوا (2^8)

عدنان محمد علي[1], فائز حسن علي[2]*, صباح منفي رضا[3]
[1]قسم الرياضيات, كلية العلوم, الجامعة المستنصرية, بغداد, العراق
[1]قسم الرياضيات, كلية العلوم, الجامعة المستنصرية, بغداد, العراق
[3]الاحصاء, كلية الادارة والاقتصاد, جامعة بغداد, بغداد, العراق

الخلاصة

يمكن استخدام نظام التشفير الانسيابي الثنائي لتشفير/ فك تشفير العديد من أنواع الملفات الرقمية، خاصةً التي يمكن اعتبارها ملفات بيانات ضخمة مثل الصور. لضمان أن عمليات التشفير أو فك التشفير تحتاج إلى وقت معقول لتشفير / فك تشفير الصور، لذلك يتعين علينا جعل مولد مفتاح التشفير يعمل بسرعة دون التأثير على التعقيد أو العشوائية للمتتابعات الثنائية للمفتاح المخرج. في هذه البحث، سنقوم أولاً بزيادة حجم المتتابعة الثنائية المخرجة الى متتابعة رقمية في الحقل GF(2^8) للحصول على متتابعة رقمية ثمانية (Byte)، ثم نعمل على اختبار المتتابعة الجديدة على انها متتابعة رقمية من الحقل−(2^8) وليست متتابعة ثنائية. لذلك علينا اختبار

*Email: faezhassan@yahoo.com

المتتابعة المخرجة الجديدة في الحقل الرياضي الجديد. وهذا يتم عن طريق تغيير اساس اختبارات العشوائية، عن طريق توسيع فرضيات Golomb من ثنائي إلى 256. ولايجاد القوانين الجديدة التي تلائم حقل المتتابعات الجديد تم اثبات بعض المبرهنات والماخذ النظرية. لقد تم مقارنة نتائج استخدام اختبارات العشوائية الموسعة بنتائج اختبارات العشوائية الثنائية لضمان تطابق قرار النجاح أو الفشل، ولقد اثبتت النتائج دقة التطابق.

1. Introduction

A random bit generator is a device or algorithm that generates a series of statistically independent and unbiased binary digits. The frequency, run, and autocorrelation tests are referred to as the Main Binary Standard Randomness Tests (MBSRT).

A good Pseudo Random Number Generator (PRNG) should meet a number of statistical criteria. The statistical properties are as follows: the output symbols should be uniformly distributed, and the binary symbols must be balanced. There are several statistical tests including the frequency, serial, run, poker, and autocorrelation test which are considered the five stander tests [1].

Since any sequence that is generated from any generator is a statistically random experiment, so if we want to test its randomness, we can depend on the statistical random tests. The randomness judgment is done by the following two conditions [2]:

1. The length of the tested sequence must equal the period of the sequence.
2. If condition (1) cannot be satisfied, so we test the sequence with as high length as possible and we apply the test for many random experiments for different lengths.

In stream ciphers, the message units are bits, and the key is usually produced by a PRBG where the plaintext is encrypted on a bit-by-bit basis. The key is fed into Random Bit Generator (RBG) to create a long sequence of binary signals. This key stream is usually mixed with the plaintext by XOR (Exclusive-OR modulo 2 addition) to produce the ciphertext stream using the same PRBG, see [1].

Universal tests were presented by Schrift and Shamir in 1993 [3] for verifying the assumed properties of a PRBG in which the output sequences of PRBG are not necessarily uniformly distributed. Gustafson et al. in 1994 [4] describe a computer package that implements various statistical tests for assessing the strength of a PRBG. In 1996, Gustafson [5] considered alternative statistics for the runs and autocorrelation tests. In 1996, Gustafson et al. [6] proposed a new repetition test that measures the number of repetitions of l-bit blocks. The test requires a count of the number of repeated patterns, however, it does not require the frequency of each pattern. Naser [7] proposed an extension to the binary standard randomness tests that are suitable to be applied on penta-sequences ($GF(5)$), and he applied the extended randomness to the output of an example of the digital penta-sequences. Ibraheem [8] attempted to extend the 2-tuple to d-tuple ($d \geq 3$) for the serial test, then we will generalize the 2-tuple binary serial test to 2-tuple digital serial test for the digital ($s-$) sequences ($s \geq 3$) that are generated from digital generators. Mohammed [9] generalized the binary poker test that is suitable to apply not only to binary sequences but also to digital ($m-$) sequences (for $m \geq 2$), in other words, the generalized poker test could be applied to digital ($m-$) sequences (for $m \geq 2$).

2. Statistical Randomness [10], [11]

In this section, we will introduce some basic concepts of the statistical randomness tests for binary sequences.

2.1 Basic Concepts of Randomness [1, 12, 7, 1, 1]

Remark (1): (random bits vs. random numbers) A RBG can be used to generate (uniformly distributed) random numbers. For example, a random integer in the interval $[0, n]$ can be obtained by generating a random bit sequence of length $log_2|n+1|$ bits, and it converts to an integer; if the resulting integer exceeds n, then there is a one option that is to discard it and a new random bit sequence is generated.

Definition (1): A Pseudo-Random Bit Generator (PRBG) is a deterministic algorithm that generates a truly random binary sequence given a truly random binary sequence. The seed is the input to the PRBG, and its output is a pseudorandom bit sequence of bits.

Remark (2): The chi-square (χ^2) distribution can be used to compare the goodness of fit of the observed frequencies of the events to their expected frequencies under a hypothesized distribution. The $\chi^2(v)$ distribution with the degree of freedom v arises in practice when the square of v independent random variables having standard normal distributions is a summed.

2.2 The Randomness Postulates of Golomb [13]

Definition (2): Let S be a periodic sequence of period N. The randomness postulates of Golomb are given as follows:

R1: In the cycle S_N of S, the number of 1′s differs from the number of 0′s by at most 1.

R2: In the cycle S_N at least:

(a). Half the runs have a length of1, at least one-fourth have a length 2, at least one-eighth has length 3, etc., as long as the number of runs so indicated exceeds 1. In other words, the number of gaps (sequence of zeros) with length 1 equals double (2^1) number of gaps with length 2, and it equals 4-times (2^2) number of gaps with length 3, and so on, and the same fact is true for blocks, namely the sequence of ones.

(b). Moreover, for each of these lengths, there are almost equal many gaps and blocks.

R3: The autocorrelation function $C(\tau)$ is two-valued. That is for some integer T:

$$N.C(\tau) = \sum_{i=0}^{N-1}(2S_i - 1)(2S_{i+\tau} - 1) = \begin{Bmatrix} N, \tau = 0 \\ T, 0 \leq \tau \leq N-1 \end{Bmatrix}$$

Definition (3) [13]: A binary sequence that satisfies Golomb′s randomness postulates is called a pseudo-noise sequence or a pn-sequence. The pseudo-noise sequences arise in practice as output sequences of maximum-length linear feedback shift registers.

3. Generalization of Golomb's Randomness Postulates

In this section, we will study the randomness of sequence $S = \{s_j\}_{j=0}^{P-1}$, where $s_j \in GF(2^8)$ and P is the period of the sequence S. In order to check the randomness of S, we have to generalize the randomness postulates of the Golomb firstly. It is known that $2^8 = 256$, so the sequence S has 256 distinct digits $s_i \in \{0,1,\dots,255\}$, where $i = 0,1,\dots,255$.

In the next subsections, we will generalize three postulates for the randomness of Golomb, namely R1, R2 and R3 which are mentioned in section two.

3.1 256-Digital Frequency Postulate

It is obvious that if the frequency n_i of each distinct digit i is approximated to other frequencies, then the 256-digital sequence is satisfied this postulate, so it must be as follows:

$n_0 \approx n_1 \approx \cdots \approx n_{255}$

Statistically, n_i represents the observed number occurrence of digit i, $i = 0,1, \ldots,255$. The expected number of occurrences is:

$$E^F = \frac{P}{256} \approx 0.004P \quad (1)$$

To obtain a sequence S which satisfies the 256-digital frequency postulate, then we have:

$n_i \approx E^F, \forall\, i = 0,1, \ldots,255.$

It is important to mention that: $\sum_{i=0}^{255} n_i = P$.

3.2 256-Digital Run Postulate

The 256-digital run here can be defined as the number of similar digits which are lie between two different digits. Let R_{ij} be the observed number of runs with type i runs with length j. Now we can depend on the mathematical deduction to deduce the two new conditions of run postulates:

1- R_{ij} is approximately equal to $1/256^{j-1}$ of the $R_{ij-1}, R_{ij} \approx \left(1/256^{j-1} R_{ij-1}\right)$, where $2 \le j \le M_i, M_i$ denotes the length of maximum run of kind i.

2- All kinds of runs of length j are approximated to each other, s.t. $R_{0j} \approx R_{1j} \approx \cdots \approx R_{255,j}$ where $1 \le j \le M_i$, it is obvious that:

$$\sum_{j=0}^{M_i} jR_{ij} = n_i, 0 \le i \le 255 \quad (2)$$

To obtain a sequence S that satisfies the 256-digital run postulate, then we have:

$R_{ij} \approx E_j^R, \forall\, i = 0,1, \ldots,255$ and $j = 1,2, \ldots, M_i$.

where E_j^R is the expected number of runs with length j.

Now we need to calculate the expected number of runs with length $\left(E_j^R\right)$, the next theorem will help to calculate it.

Theorem (1): Let S be a sequence that satisfies the run postulate, then the expected number of runs with length j, E_j^R can calculated as follows:

$E_j^R = \frac{255P}{256^{j+2}}$, $1 \le j \le M$, where $M = max(M_0, M_1, \cdots, M_{255})$.

Proof

From relation (2), and $n_i \approx E^F$, then:

$$E^F = \frac{P}{256} = E_1^R + 2E_2^R + \cdots + ME_M^R = \sum_{j=1}^{M} j.E_j^R. \quad (3)$$

And since $R_{i2} = \frac{R_{i1}}{256}$ then $E_2^R = \frac{E_1^R}{256}$, since S satisfies the runs postulate, so in general,

$$E_j^R = \frac{E_1^R}{256^{j-1}}, 2 \le j \le M \quad (4)$$

substitute Eq. (3) in Eq. (4), we get:

$$\frac{P}{256} = \sum_{j=1}^{M} \frac{j.E_1^R}{256^{j-1}} = 256E_1^R . \sum_{j=1}^{M} \frac{j}{256^j} \quad . \quad (5)$$

By using the ratio test:

as $M \to \infty$ then $S' = \lim_{M\to\infty} \sum_{j=1}^{M} \frac{j}{256^j} = \sum_{j=1}^{\infty} \frac{j}{256^j}$.

For S' we have:

$s_1 = \frac{1}{256}$,

$s_2 = \frac{1}{256} + \frac{2}{256^2}$, so

$$s_M = \frac{1}{256} + \frac{2}{256^2} + \cdots + \frac{M}{256^M} \quad (6)$$

$$\frac{1}{256} s_M = \frac{1}{256^2} + \frac{2}{256^3} + \cdots + \frac{M}{256^{M+1}} \quad (7)$$

By subtract relation (7) from relation (6), we have:

$s_M - \frac{1}{256} s_M = \frac{1}{256} + \frac{2}{256^2} + \cdots + \frac{M}{256^M} - \frac{1}{256^2} - \frac{2}{256^3} - \cdots - \frac{M}{256^{M+1}}$

then:

$\frac{255 s_M}{256} = \frac{1}{256} + \frac{1}{256^2} + \cdots + \frac{1}{256^M} - \frac{M}{256^{M+1}}$

$255 s_M = 1 + \frac{1}{256} + \frac{1}{256^2} + \cdots + \frac{1}{256^{M-1}} - \frac{M}{256^M} = \sum_{j=1}^{M} \frac{1}{256^{j-1}} - \frac{M}{256^M}$

$s_M = \frac{1}{255}\left(\sum_{j=1}^{M} \frac{1}{256^{j-1}} - \frac{M}{256^M}\right)$

let $M \to \infty$, the series

$$S' = \lim_{M\to\infty} s_M = \frac{1}{255}\left(\sum_{j=1}^{\infty} \frac{1}{256^{j-1}} - \lim_{M\to\infty} \frac{M}{256^M}\right) \quad (8)$$

Notice that $\lim_{M\to\infty} \frac{M}{256^M} = \lim_{M\to\infty} M . \lim_{M\to\infty} \frac{1}{256^M} = \lim_{M\to\infty} M . 0 = 0$

Since the series $\sum_{j=1}^{\infty} \frac{1}{256^{j-1}}$ is geometric series [14] with $a = 1$ and $r = \frac{1}{256}$, and since $|r| = \frac{1}{256} < 1$, then the series is convergence series and the sum:

$$S = \sum_{j=1}^{\infty} \frac{1}{256^{j-1}} = \frac{a}{1-r} = \frac{1}{1-1/256} = \frac{256}{255} \approx 1.004 \quad (9)$$

By substituting relation (9) in relation (8), we have:

$$S' = \frac{1}{255} . \frac{256}{255} = \frac{256}{255^2} = \frac{256}{65025}. \quad (10)$$

Substitute Eq. (10) in Eq. (5) we get:

$\frac{P}{256} = 256 E_1^R . \frac{256}{65025}$

$\therefore E_1^R = \frac{255^2 P}{256^3} \approx 0.004P$

In general, and by using mathematical induction, we have:

$E_j^R = \frac{255^2 P}{256^{j+2}} = \frac{0.992P}{256^j}$ where $j = 1,2, \ldots, M$.

Theorem (2): The expected number of total numbers E^R of runs for any kind i, where $0 \le i \le 255$ is $E^R = \frac{255P}{256^2}$.

Proof

$$E^R = \sum_{j=1}^{M} E_j^R = \sum_{j=1}^{M} \frac{255^2 P}{256^{j+2}} = \frac{255^2 P}{256^3} \cdot \sum_{j=1}^{M} \frac{1}{256^{j-1}} \quad (11)$$

As we have proved in Theorem (1), that the series $\sum_{j=1}^{M} \frac{1}{256^{j-1}}$ is geometric convergent series as M→∞, then:

$$\sum_{j=1}^{M} \frac{1}{256^{j-1}} = \frac{256}{255} \approx 1.004 \quad (12)$$

Using relation (12) in relation (11), we obtain:

$E^R = \frac{255^2 P}{256^3} . \frac{256}{255} = \frac{255P}{256^2} \approx (0.004)(1.004)P = 0.004P$

3.3 256-Digital Auto-Correlation Postulate

As mentioned before, this postulate is found to specify if the tested 256-digital sequence has a repetition with itself. Let $n_0(\tau)$ be the number of similar digits in S after shifting it by τ, and let $n_1(\tau)$ be the number of distinct digits in S after shifting it by τ, where $\tau = 1,2, \ldots, P-1$, s.t.

$n_0(\tau) = \#\{s_i = s_{i+\tau}: \forall\, i = 1,2, \dots, P-1\}$
$n_1(\tau) = \#\{s_i \neq s_{i+\tau}: \forall\, i = 1,2, \dots, P-1\}$
such that $\tau = 1,2, \dots, P-1$.
Where $n_0(\tau) + n_1(\tau) = P - \tau$.
As it is known that the probability of similarity of one digit from 256 digits is $1/256$, then the expected number of similarity for the auto-correlation postulate is:
$E_0^A(\tau) = \frac{P-\tau}{256} \approx 0.004(P-\tau)$
and the expected number of differences for the auto-correlation postulate is:
$E_1^A(\tau) = \frac{255(P-\tau)}{256} \approx 0.996(P-\tau)$
Note that $E_1^A(\tau) = (P-\tau) - E_0^A(\tau)$

4. Modified the 256-Digital Randomness Test

In this section, we will reformulate three main testing laws that are suitable to apply to 256-digital sequences. We called the new digital randomness tests the Main 256-Digital Standard Randomness Tests (M256DSRT). Let S be the 256-digital sequence, which has to be tested with length L has an element(s) $s_i \in \{0,1, \dots,255\}$ and $i = 0,1, \dots,255$.

4.1 256-Digital Frequency Test (256DFT)

Let n_i be the observed number of occurrences of digit i, where $i = 0,1, \dots,255$, and the expected number of occurrences of digit i is $E^F = \frac{L}{256} \approx 0.004L$, then the statistic value T^F of 256DFT is calculated as follows:

$$T^F = \sum_{i=0}^{255} \frac{\left(n_i - E^F\right)^2}{E^F} = \sum_{i=0}^{255} \frac{(n_i - L/256)^2}{L/256} = \frac{256}{L} \sum_{i=0}^{255} (n_i - L/256)^2 \ . \quad (13)$$

Or it can be written as:

$$T^F \approx \frac{256}{L} \sum_{i=0}^{255} (n_i - 0.004L)^2 \ , \quad (13')$$

with freedom degree $\upsilon = 255$.

Next lemma gives more simple formula to apply the 256DFT to calculate T^F by using formula (13).

Lemma (1): For 256DFT of 256-digital sequence S: $T^F = \frac{P}{L} \sum_{i=0}^{255} n_i^2 - L$.

Proof: From relation (13):

$$T^F = \sum_{i=0}^{255} \frac{(n_i - L/256)^2}{L/256} = \frac{256}{L} \sum_{i=0}^{255} (n_i - L/256)^2$$

$$T^F = \frac{256}{L} \sum_{i=0}^{255} n_i^2 - 2\frac{L}{256} \sum_{i=0}^{255} n_i + \left(\frac{L}{256}\right)^2$$

But $\sum_{i=0}^{255} n_i = L$, then

$$T^F = \frac{256}{L} \sum_{i=0}^{255} n_i^2 - 2L + L$$

$$\therefore T^F = \frac{256}{L} \sum_{i=0}^{255} n_i^2 - L \quad (14)$$

4.2 256-Digital Run Test (256DRN)

Let R_{ij} be the observed number of runs with type i runs with length j, and let E_j^R be the expected number of runs with length j, then, the statistic value T_i^R of 256DRT is calculated as follows:

$$T_i^R = \sum_{j=0}^{M_i} \frac{\left(R_{ij} - E_j^R\right)^2}{E_j^R} = \sum_{j=0}^{M_i} \frac{\left(R_{ij} - 255^2 L/256^{j+2}\right)^2}{255^2 L/256^{j+2}} \quad , \tag{15}$$

or approximately, can be written as:

$$T_i^R \approx \sum_{j=0}^{M_i} \frac{\left(R_{ij} - 0.992L/256^j\right)^2}{0.992L/256^j} \tag{15'}$$

for $0 \le i \le 255$.

With freedom degree $\upsilon_i = M_i - 1$. A formula (15) can be reformulated in another face:

$$T_i^R = (255/256)^2/L \sum_{j=0}^{M_i} 256^j R_{ij}^2 - 2\sum_{j=0}^{M_i} R_{ij} + (255/256)L \, , \tag{16}$$

or it can be written as:

$$T_i^R \approx 0.992/L \sum_{j=0}^{M_i} 256^j R_{ij}^2 - 2\sum_{j=0}^{M_i} R_{ij} + 0.996L \tag{16'}$$

In another style of calculating a 256-run test, we can find the value T^R which represents the sum of the values T_i^R for $0 \le i \le 255$ as follows:

$$T^R = \sum_{i=0}^{255} T_i^R \quad . \tag{17}$$

However, here it is compared with T_0 at freedom degree $\upsilon_i = 256(M_i - 1)$.

4.3 256-Digital Auto-Correlation Test (256DACT)

In this subsection, we will attempt to calculate the statistical value of the **256**-Digital Auto-Correlation Test (256DACT) in two faces.

4.3.1 256DACT using Classical Model

In classical 256DACT, we want to estimate $T^A(\tau)$ in this model so we add the 256-sequence S before it is shifted by τ with 256-sequence S after it is shifted by τ (mod 256), so we get a new 256-digital sequence S' with length $L - \tau$ and the new expected mean of occurrence of any digit is $E^A(\tau) = \frac{L-\tau}{256}$, then applying the 256-DACT using the following relation:

$$T^A(\tau) = \sum_{i=0}^{255} \frac{\left(n_i(\tau) - E^A(\tau)\right)^2}{E^A(\tau)} = \sum_{i=0}^{255} \frac{\left(n_i(\tau) - \frac{L-\tau}{256}\right)^2}{\frac{L-\tau}{256}} = \frac{256}{L-\tau}\sum_{i=0}^{255} n_i^2(\tau) - (L-\tau) \, . \tag{18}$$

Where $n_i(\tau)$ is the frequency of the digit i in the sequence S', and the freedom degree $\upsilon = 255$.

4.3.2 256DACT using Modern Model

In this model, let $n_0(\tau)$ be the number of similar digits in S after it is shifted by τ, and $n_1(\tau)$ be the number of distinct digits in S after it is shifted by τ, respectively, where $\tau = 1,2, \dots, L-1$. While the expected number of similarities and differences, respectively are:

$E_0^A(\tau) = \frac{L-\tau}{256}$ and $E_1^A(\tau) = \frac{255(L-\tau)}{256}$

The following lemma proves that the chi square of auto correlation test for the 256-digital sequence S is:

$$T^A(\tau) = \sum_{i=0}^{1} \frac{\left(n_i(\tau) - E_i^A(\tau)\right)^2}{E_i^A(\tau)} = \frac{\left(n_0(\tau) - \frac{L-\tau}{256}\right)^2}{\frac{L-\tau}{256}} + \frac{\left(n_1(\tau) - \frac{255(L-\tau)}{256}\right)^2}{\frac{255(L-\tau)}{256}} \, , \tag{19}$$

with freedom degree $\upsilon = 1$.

Lemma (2): The Chi-square of autocorrelation test for the 256-digital sequence S that is shifted by τ is:

$T^A(\tau) = \frac{1}{255(L-\tau)}\left(256n_0(\tau) - (L-\tau)\right)^2$

Proof: For simplicity, take $L' = L - \tau, n_0 = n_0(\tau)$ and $n_1 = n_1(\tau)$.

$$T^A(\tau)=\sum_{i=0}^{1}\frac{\left(n_i-E_i^A(\tau)\right)^2}{E_i^A(\tau)}=\frac{\left(n_0-\frac{L'}{256}\right)^2}{\frac{L'}{256}}+\frac{\left(n_1-\frac{255L'}{256}\right)^2}{\frac{255L'}{256}}.$$

Since $n_0+n_1=L-\tau=L'$, then $N_1=L'-n_0$

Thus:

$$T^A(\tau)=\frac{255\left(n_0-\frac{L'}{256}\right)^2+\left(L'-n_0-\frac{255L'}{256}\right)^2}{\frac{255L'}{256}}$$

$$T^A(\tau)=\frac{256}{255L'}\left(255\left(n_0-\frac{L'}{256}\right)^2+\left(L'-\frac{255L'}{256}-n_0\right)^2\right)$$

$$T^A(\tau)=\frac{256}{255L'}\left(255\left(n_0-\frac{L'}{256}\right)^2+\left(L'\left(1-\frac{255}{256}\right)-n_0\right)^2\right)$$

$$T^A(\tau)=\frac{256}{255L'}\left(255\left(n_0-\frac{L'}{256}\right)^2+\left(\frac{L'}{256}-n_0\right)^2\right)$$

Since

$\left(n_0-\frac{L'}{256}\right)^2=\left(\frac{L'}{256}-n_0\right)^2$, then:

$$T^A(\tau)=\frac{256}{255L'}\left(256\left(n_0-\frac{L'}{256}\right)^2\right)=\frac{256^2}{255L'}\left(n_0-\frac{L'}{256}\right)^2$$

$$T^A(\tau)=\frac{256^2}{255L'}\left(\frac{256n_0-L'}{256}\right)^2=\frac{1}{255L'}(256n_0-L')^2$$

$$\therefore T^A(\tau)=\frac{1}{255(L-\tau)}\left(256n_0(\tau)-(L-\tau)\right)^2 \quad (20)$$

Or approximately, can be written as:

$$T^A(\tau)\approx\frac{0.004}{(L-\tau)}\left(256n_0(\tau)-(L-\tau)\right)^2 \quad (20')$$

5. Implementation of M256DSRT on 256-Digital Sequences (256DS)

Now we will try to implement the M256DSRT on output sequences of 256DS. In this manner, two cryptosystems are chosen to test their output, these cryptosystems are:

1- **Stream cipher cryptosystem (SCC) [14]:** consists of a number of LFSR's with balanced nonlinear Boolean function.

2- **Multiplicative Cyclic Group System (MCGS) [15]**: This new generator can be consisting of a single or more than one MCG unit with a balance combining 133, 55, 235, … 127 215 41.

1- **256-Frequency test**: **Table 1** shows the frequency values N_i.

Table 1: frequencies (n_i) of digits (0,1,…,255).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	5	5	7	3	2	4	4	6	4	2	3	5	4	5	4	6
1	2	7	5	3	6	5	3	5	7	1	4	2	3	4	5	2
2	3	6	7	3	7	9	5	5	4	5	2	6	4	3	0	8
3	5	3	4	0	4	1	3	2	2	3	4	5	7	3	2	6
4	1	2	1	2	2	7	3	3	2	8	5	6	1	3	5	2
5	5	5	4	1	3	3	2	1	3	0	7	5	5	4	6	6
6	4	5	3	6	6	5	5	5	2	2	1	4	3	2	4	5
7	5	1	2	6	2	4	5	4	5	4	4	5	7	4	3	5
8	4	3	6	1	4	3	4	4	5	4	6	4	6	4	1	4
9	7	3	7	7	4	4	4	3	3	3	6	5	5	3	2	4
A	4	2	7	2	4	1	6	2	6	6	4	1	2	1	6	3
B	5	2	4	2	6	2	2	2	1	3	4	1	6	3	5	6
C	5	4	1	3	8	7	3	2	3	2	4	5	3	3	2	5
D	3	4	3	4	4	3	5	9	6	1	6	6	8	7	4	3
F	11	3	2	3	3	2	3	2	4	3	4	1	6	0	5	1

By using formula (14), we get:

$T^F = \frac{256}{1000}(25 + 25 + 49 + \cdots + 0 + 25 + 1) - 1000 = 238.528$

This value is compared with $T_0 = 288.016$ where $\upsilon = 255$. The sequence S passed this test since $T^F \leq T_0$.

1. **256-RUN test**: Table (1) approximately shows the same values for the run test, with runs with length 2 for the bytes 7, 212 and 240. In this part, we applied a 256-run test in two faces:

(a). By using formula (16), all the values of T_i^R are passed with $T_0 = 3.841$ where $\upsilon = 1$ except the (10) values mentioned in **Table 2** are failed:

Table 2: the failed values of T_i^R to pass 256-run test.

	1	2	3	4	5	6	7	8	9	10
Bytes	53	62	63	67	89	105	231	236	240	253
Values	6.775	3.876	4.389	3.876	4.389	3.876	6.775	4.389	6.775	3.876

Notice that the average of failed values is 4.9 that means the ratio of failed compared with $T_0 = 3.841$ is very little.

(b). By using formula (17), $T^R = 226.663$, this value is compared with $T_0 = 289.072$ where $\upsilon = 256$. The sequence S passed this test since $T^R \leq T_0$.

2. **256-Auto-Corrleation test**: **Table 3** shows the frequency values n_0 for similarity digits and n_1 for difference for 256-Auto-Correlation test for shifting $\tau = 1,2, \ldots, 100$.

Table 3: frequencies of n_0 and n_1 of bytes with $T^A(\tau)$ values

τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$
1	3	996	0.209	26	11	963	13.661	51	2	947	0.789	76	1	923	1.894
2	1	997	2.163	27	3	970	0.169	52	5	943	0.456	77	4	919	0.043
3	1	996	2.160	28	3	969	0.168	53	4	943	0.025	78	4	918	0.044
4	2	994	0.922	29	4	967	0.011	54	2	944	0.781	79	2	919	0.712
5	4	991	0.003	30	3	967	0.165	55	8	937	5.049	80	3	917	0.098
6	6	988	1.159	31	7	962	2.741	56	5	939	0.469	81	5	914	0.556
7	4	989	0.004	32	4	964	0.013	57	1	942	1.963	82	6	912	1.632
8	3	989	0.198	33	3	964	0.161	58	3	939	0.126	83	4	913	0.049
9	6	985	1.175	34	3	963	0.159	59	4	937	0.029	84	4	912	0.050
10	7	983	2.548	35	1	964	2.043	60	2	938	0.764	85	6	909	1.653
11	2	987	0.902	36	3	961	0.156	61	4	935	0.030	86	3	911	0.091
12	3	985	0.192	37	6	957	1.337	62	3	935	0.121	87	1	912	1.854
13	5	982	0.341	38	5	957	0.412	63	4	933	0.032	88	4	908	0.054
14	4	982	0.006	39	3	958	0.152	64	3	933	0.118	89	3	908	0.088
15	3	982	0.187	40	1	959	2.025	65	0	935	3.667	90	3	907	0.087
16	5	979	0.349	41	5	954	0.421	66	5	929	0.503	91	3	906	0.086
17	6	977	1.220	42	5	953	0.424	67	1	932	1.926	92	4	904	0.058
18	5	977	0.355	43	5	952	0.428	68	3	929	0.113	93	4	903	0.059
19	3	978	0.181	44	6	950	1.380	69	4	927	0.036	94	3	903	0.082
20	2	978	0.876	45	3	952	0.144	70	3	927	0.111	95	6	899	1.725
21	2	977	0.874	46	3	951	0.142	71	3	926	0.109	96	5	899	0.613
22	5	973	0.366	47	2	951	0.800	72	2	926	0.731	97	10	893	11.924
23	9	968	7.068	48	2	950	0.797	73	3	924	0.107	98	5	897	0.621
24	1	975	2.083	49	1	950	1.992	74	6	920	1.576	99	5	896	0.625
25	1	974	2.079	50	2	948	0.792	75	3	922	0.105	100	5	895	0.629

Of course, the $T^A(\tau)$ values are calculated using formula (19).

5.2 Testing 256DS of MCGS

Consider the following sequence S with length $L = 10000$ bytes:
171, 74, 73, ... , 117, 155, 210.

1- **256-Frequency test**: **Table 4** shows the frequency values n_i

Table 4: frequencies (n_i) of digits $(0,1,\ldots,255)$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	33	39	41	36	41	41	37	39	38	38	39	39	41	34	39	39
1	37	40	41	40	39	42	37	38	41	37	42	40	35	39	43	41
2	40	38	38	41	40	39	37	37	39	39	41	35	38	40	38	41
3	42	38	37	37	38	39	40	37	39	41	39	38	41	37	41	36
4	37	41	37	39	38	40	41	40	41	42	43	42	38	40	41	42
5	39	41	39	39	41	38	38	38	37	40	40	39	40	39	38	40
6	38	38	40	37	38	39	40	39	39	34	41	41	41	43	42	39
7	40	41	38	38	40	39	41	37	41	42	41	40	38	36	40	40
8	39	41	41	37	40	36	40	41	37	41	39	40	40	39	41	39
9	38	41	35	38	41	37	42	39	37	41	37	36	40	38	33	41
A	38	37	41	36	42	36	42	40	36	42	37	41	38	43	38	39
B	39	39	39	39	35	37	40	41	38	40	39	35	40	40	41	37
C	38	36	39	38	41	40	37	40	41	37	36	39	41	39	36	39
D	40	40	41	40	41	37	41	40	38	39	41	39	39	38	41	39
E	39	42	41	40	40	39	36	40	37	42	35	40	38	40	36	41
F	40	38	40	40	38	39	41	38	37	38	38	41	38	35	38	38

By using formula (14), we get:
$T^F = \frac{256}{10000}(33^2 + 39^2 + \cdots + 38^2 + 38^2) - 10000 = 24.243$
This value is compared with $T_0 = 288.016$ where $\upsilon = 255$. Sequence S passed this test since $T^F \leq T_0$.

2- **256-RUN test**: Table (1) approximately shows the same values for a run test, with runs with length 2 for the (30) bytes. In this part, we applied 256-run test in two faces:
(a). By using formula (16), all the values of T_i^R are passed with $T_0 = 3.841$ where $\upsilon = 1$ with no fail.
(b). By using formula (17), $T^R = 27.009$, this value is compared with $T_0 = 289.072$ where $\upsilon = 256$. Sequence S passed this test since $T^R \leq T_0$.

3- **256-Auto-Corrleation test**: **Table 5** shows the frequency values n_0 for similarity digits and n_1 for difference for 256-Auto-Correlation test for shifting $\tau = 1,2, \ldots,100$.

Table 5 : frequencies of n_0 and n_1 of bytes with $T^A(\tau)$ values.

τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$	τ	n_0	n_1	$T^A(\tau)$
1	33	9966	0.943	26	34	9940	0.634	51	38	9911	0.019	76	32	9892	1.185
2	43	9955	0.400	27	44	9929	0.655	52	34	9914	0.610	77	42	9881	0.272
3	30	9967	2.106	28	33	9939	0.913	53	41	9906	0.119	78	39	9883	0.002
4	39	9957	0.000	29	33	9938	0.912	54	40	9906	0.034	79	36	9885	0.196
5	32	9963	1.275	30	38	9932	0.023	55	53	9892	5.176	80	29	9891	2.463
6	40	9954	0.024	31	40	9929	0.029	56	44	9900	0.687	81	35	9884	0.364
7	42	9951	0.226	32	33	9935	0.909	57	40	9903	0.035	82	41	9877	0.132
8	39	9953	0.000	33	41	9926	0.110	58	47	9895	1.723	83	38	9879	0.014
9	33	9958	0.935	34	40	9926	0.030	59	36	9905	0.207	84	28	9888	2.986

10	26	9964	4.363	35	35	9930	0.397	60	40	9900	0.036	85	31	9884	1.549
11	48	9941	2.075	36	42	9922	0.244	61	35	9904	0.378	86	38	9876	0.014
12	41	9947	0.101	37	42	9921	0.245	62	47	9891	1.730	87	39	9874	0.002
13	45	9942	0.923	38	43	9919	0.431	63	31	9906	1.580	88	37	9875	0.077
14	32	9954	1.264	39	43	9918	0.432	64	36	9900	0.205	89	28	9883	2.977
15	22	9963	7.442	40	33	9927	0.900	65	47	9888	1.736	90	44	9866	0.725
16	39	9945	0.000	41	35	9924	0.393	66	42	9892	0.264	91	36	9873	0.190
17	45	9938	0.928	42	33	9925	0.898	67	36	9897	0.203	92	42	9866	0.282
18	36	9946	0.231	43	37	9920	0.093	68	35	9897	0.373	93	27	9880	3.551
19	28	9953	3.109	44	29	9927	2.525	69	34	9897	0.595	94	33	9873	0.842
20	36	9944	0.229	45	39	9916	0.000	70	37	9893	0.083	95	36	9869	0.188
21	34	9945	0.639	46	37	9917	0.092	71	43	9886	0.460	96	48	9856	2.250
22	44	9934	0.650	47	40	9913	0.032	72	31	9897	1.567	97	35	9868	0.352
23	42	9935	0.236	48	36	9916	0.213	73	38	9889	0.016	98	23	9879	6.381
24	42	9934	0.237	49	37	9914	0.090	74	41	9885	0.128	99	36	9865	0.186
25	38	9937	0.024	50	38	9912	0.019	75	37	9888	0.081	100	36	9864	0.185

Of course, the $T^A(\tau)$ values are calculated using formula (19).

6. Conclusions and Future Works

This work concludes the following aspects:

1- We see that relation (19) needs freedom degree $\upsilon = 255$, while relation (20) needs freedom degree $\upsilon = 1$, so, we believe that relation (20) is better than relation (19).

2- It is known that if the length of the tested 256-sequence is as long as possible, then we may obtain more correct randomness results, so we see that to judge correctly the randomness of the cryptosystem, we must take 256-sequence with length $L \gg 2000$.

3- In Lemmas (1) and (2), it is noted that we do not need to calculate the expected value of any sample in the three tests, so we do not need Eq. (13) and Eq. (15) anymore.

4- We have to compare the results of applying the two models (classical and modern) for **256**DACT to guarantee the correctness of 256-randomness tests decision.

5- We may show that if the 256-sequence passes the (3) 256-randomness tests then it must pass the binary-randomness tests and vice versa.

6- We have to expand more randomness tests, like serial, Poker,…etc. to be applied on 256-digital tests, in order to estimate the real randomness of the 256-sequence.

References

[1] A. A. Naser and F. H. Ali, "Robust and Efficient Dynamic Stream Cipher Cryptosystem," *Iraqi Journal of Science,* vol. 59, no. 2C.1, pp. 1105-1114, 2018.

[2] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, New York: John Wiley & Sons, 2nd edition, 1996.

[3] A. a. S. A. Schrift, "Universal Tests for Nonuniform Distributions," *Journal of Cryptology,* vol. 6, pp. 119-133, 1993.

[4] H. D. E. N. L. a. C. W. Gustafson, "A Computer Package for Measuring the Strength of Encryption Algorithms," *Computers & Security,* vol. 13, 1994.

[5] H. Gustafson, *Statistical Analysis of Symmetric Ciphers,* Ph. D. thesis, Queensland University of Technology, 1996.

[6] H. M. Gustafson, E. P. Dawson and J. D. Golić , "Randomness Measures Related to Subset Occurrence," in *Cryptography: Policy and Algorithms, International Conference*, Brisbane, Queensland, Australia, July 1995.

[7] A. G. Naser, "Testing the Randomness and Using the Digital Sequences of GF(5) in Cryptography," in *12th Scientific Conference of Al-Mansour College*, Baghdad, Iraq., 24-25 May 2013.

[8] S. K. Ibraheem, "Serial Test Extension and Generalization to Test the Digital Sequences," *Al-Mustansiriyah Journal of Science*, vol. 25, no. 4, pp. 83-96, 2014.

[9] S. A. Mohammed, "Applied Poker Test for General Digital Sequences," *IOSR Journal of Mathematics (IOSR-JM)*, vol. 12, no. 1, pp. 17-23, 2016.

[10] S. Y. Yan, Number Theory for Computing, Berlin Heidelberg, New York: Springer-Verlag, 2000.

[11] W. L. a. M. A. R. Martinez, Computational Statistics Handbook with MATLAB, Chapman & Hall/CRC, Library of Congress Cataloging-in-Publication Data, 2002.

[12] W. J. Gilbert, Modern Algebra with Applications, Wiley-Interscince, March 2002.

[13] A. P. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[14] A. A. Naser and F. H. Ali, "Design of New Dynamic Cryptosystem with High Software Protection," *Iraqi Journal of Science*, vol. 59, no. 4C, pp. 2301-2309, 2018.

[15] F. H. Ali, "Use the Multiplicative Cyclic Group to Generate Pseudo Random Digital Sequences," *Journal of Al-Rafidain University College for Sciences*, vol. 20, pp. 122-135, 2006.