BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 66 (70), Numărul 2, 2020
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

THE EFFECTIVENESS OF THE STATISTICAL TESTING OF RANDOMNESS IN A COMPLETE CRYPTOGRAPHIC SYSTEM

BY

EUGEN NEACȘU[*]

Advanced Technology Institute, Bucureşti, Romania

Received: November 24, 2020
Accepted for publication: December 8, 2020

Abstract. The aim of this paper is to describe the large classes of pseudo-random number generators (the congruent linear generator and the generators with displacement registers) and their functional properties. The theory of pseudo-random number generators is necessary in defining the theoretical methods underlying the design and analysis of string algorithms. Those algorithms, that have the necessary random characteristics, are currently adopted to generate cryptographic keys, used in complete cryptographic systems (CSS). The paper concludes with a case study on the effectiveness of statistical testing of randomness, one of the fundamental methods of analyzing the security of a CCS.

Keywords: generator; bit; cryptography; PRNG; algorithm.

1. Introduction

Random number generators (RNG) have a wide range of use (neural networks, image processing, genetic algorithms, video games), and the best known generator types are true random number generators (TRNG) and pseudo-random number generators (PRNG). While the former use truly random factors (cosmic noise, oscillators), the others are based on different types of algorithms,

[*]Corresponding author; *e-mail*: neacsu.eugen@yahoo.com

or input values obtained from the states of the devices on which they run (Meyer and Tuchman, 1979).

PRNGs are successfully used in GIS (Geographic Information System) applications, for dynamic modeling or stochastic simulation. More recently, they are used for password protection operations in RFID (Radio Frequency IDentification) devices. TRNGs have been successfully implemented in FPGAs (Muhammad and Qasim, 2009).

The generation of pseudo-random numbers, used in the construction of unpredictable cryptographic keys for an attacker is a fundamental cryptographic primitive. A method often used in the construction of cryptosystems is given by applying the bitwise XOR operation between a random string and the message to be encrypted. Algorithms that generate strings of numbers relatively independent of each other and that have random properties are the basis for the implementation of PRNGs. A random number represents a singular value of a random variable. If the distribution is not specified, then it is assumed to be uniform in the range [0, 1]. It can be said about a number that it is (pseudo)random only from the analysis of the way it was generated. A number in a string is random if all the numbers in the string generated before it had the same probability of occurrence. These numbers must be statistically independent, so that the information about the numbers already generated does not determine the knowledge of the number to be generated.

To obtain true random numbers it is recommended to use hardware generators, as any PRNG running on a computer is a deterministic algorithm and generates strings of numbers with properties different from those of true random ones. One such property is periodicity, caused by the fact that the generator uses the computer's memory, so that, after a number of iterations, the algorithm returns to an internal state already crossed, thus entering an infinite cycle. Another property is that by entering the same initialization values into a PRNG, it will produce the same string of numbers.

PRNG implementations may have features considered unacceptable in order to pass statistical tests:

- poor dimensional distribution;
- for certain initial conditions, the period may be shorter than expected;
- certain bits may have more random characteristics than others;
- successive values generated in a string may not be independent;
- lack of uniformity of the generated numbers.

There are statistical tests of randomness that allow an efficient analysis of the output of a PRNG. PRNGs that successfully pass this analysis are called cryptographically secure (CSPRNG) and have good statistical properties, being able to withstand a cryptographic attack (NIST Special Publication 800-57, 2007).

2. Features of CSPRNG

The first requirement of CSPRNG (Cryptographically Secure Pseudo-Random Number Generator) is that it must satisfy the "next-bit" test, in the sense that given *k* bits of a randomly generated string, there is no algorithm to predict the next bit generated, with a probability greater than 1/2. Practice has shown that a generator that manages to pass this test, will pass the other statistical tests in a polynomial time to verify the randomness of the generator.

CSPRNGs are characterized by resistance to attacks if the current state is known and do not allow the determination of the previous string of numbers generated, neither the deduction of the next state of the generator. One such high-security generator is BBS (Blum Blum Shub), but it has the disadvantage of long processing time. Block ciphers can be converted to CSPRNGs by running in CBC, CFB, or OFB modes. For an arbitrarily chosen key, a data block with the values 0, 1, 2, ..., is encrypted. The initial value of the counter can be any non-zero value. Thus, a CSPRNG generator with a period of 2^n for an *n*-bit block can be obtained (Kelsey *et al*., 1998).

From a mathematical point of view, the algorithm for generating a string of pseudo-random numbers is based on the calculation of a uniformly distributed discrete random variable.

A discrete random variable $x \in \{1, 2, \ldots, n\}$ is evenly distributed if $P(x_i) = 1/n$ for $1 \le i \le n$. The calculation of such a variable uses type functions $g{:}M^k \to M$, where M represents the subset of natural numbers that can be represented in the computer, and k is a value that defines the generator. Generating a string of pseudo-random numbers requires establishing the initial values $x_1, x_2, ..., x_k$, which form the seed of the generator, using the recurrence relation:

$$x_n = g\,(x_{n-1}, x_{n-2}, ..., x_{n-k}), \quad n > k \tag{1}$$

Because M is a finite set, it means that the string x_n is periodic. There are at least two conditions that the generator thus defined must meet:

– the period of the generator to be long in relation to the number of generated values;

– the generated values should not be correlated.

These conditions can be met by using the appropriate g function. The most frequently used methods are the congruent ones, in which recurrence is of the form:

$$x_n = f(x_{n-1}, x_{n-2}, ..., x_{n-k}) \bmod m \tag{2}$$

with $f : M^k \to M$ a generation function, and k and m values that define the generator.

The most used functions f are the linear ones:

$$x_n = (a_1 x_{n-1} + a_2 x_{n-2} + ... + a_k x_{n-k} + c) \bmod m \qquad (3)$$

where a_1, a_2, ..., a_k, c and m are integer values that characterize the generator. Usually x_1, x_2, …, x_k are the initial values, and a_i , $c \in \{0, 1, …, m\text{-}1\}$. The generated values belong to the set $\{0, 1, …, m\text{-}1\}$, and the maximum period m, this being usually the maximum positive integer that can be represented in the computer.

The choice of parameters a_i and c is a difficult one, these being determined, as a rule, following the efficiency tests. The most commonly used generators are those of the 1st order:

$$x_{n+1} = ax_n + c \mod m \qquad (4)$$

If $c = 0$, the generator is called multiplicative congruent, and if $c \neq 0$, the generator is called mixed congruent.

3. The Linear Congruential Generator

The linear congruential generator (LCG) is a pseudo-random string generator, which respects the mathematical relation:

$$x_n = (ax_{n\text{-}1} + b) \bmod m \qquad (5)$$

where x_n represents the n element of the string, while $x_{n\text{-}1}$ is the previous element in the string. a, b and mod m are constant values, while the seed is given by the value of x_0.

The generator period is less than or equal to m. For a, b and m chosen accordingly (b is relatively prime with m), the generator will have the maximum period m. The constants thusly chosen are used in the implementations of linear congruent generators that pass the spectral test within the randomness test batteries, for dimensions 2, 3, 4, 5 and 6.

The advantage of this type of generator is the speed, as few operations are required, but due to their predictability, their use in cryptography is not indicated. All types of polynomial congruent generators (linear, square and cubic) have been broken (Paar and Pelzl, 2010).

4. The Linear Feedback Shift Register

The linear feedback shift register (LFSR) has two component parts: the shift register and the feedback function. In essence, the shift register is represented by a string of bits, having a length equal to their number. When generating a bit, all bits in the register are shifted to the right. The last bit on the right that leaves the register after moving is the generator output. Completing the register with a new bit on the first position on the left is done by calculating a value given by a function that uses the other bits of the register (Fig. 1). Generators that use displacement registers are easy to implement.

The feedback function is performed by the XOR operation between certain bits in the register, the list of these bits being called the sequence "tap" or Fibonacci configuration. LFSR is the most common type of shift register used in cryptography, a reason being the fact that due to the relatively simple feedback function, both its optimized implementations and mathematical methods for analyzing the randomness of the generated string were developed.

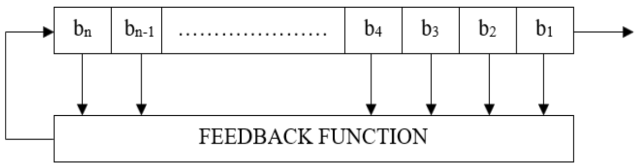

Fig. 1 – Shift register with feedback.

An *n*-bit LFSR can be in one of the 2^n-1 possible states (the state in which all the bits of the register are *0* is not taken into account), and can generate a string of 2^n-1 pseudo-random bits before repeating the process.

The construction of a polynomial primitive of degree *n* mod 2, is realized by choosing a polynomial and testing the primality condition, which is relatively difficult to achieve, due to the large number of non-zero coefficients, preferred in cryptographic applications (Perrig *et al*., 2004).

The internal state of an LFSR consists of the next *n* generated bits, which can lead to the determination of the feedback scheme from only *2n* generated bits, by using the Berlekamp-Massey algorithm.

5. The Generalized Feedback Shift Register

The generalized feedback shift register (GFSR) is an attempt to improve the results obtained by running statistical tests, based on the theory of trinomial primitives of the form $x^p + x^q + 1$. This type of shift register can be expressed by the expression:

$$x_i = x_{i-p} \oplus x_{i-q} \tag{6}$$

where each x_i represents a vector of dimension $w \in \{0, 1\}$. The generator has a maximum period of 2^p-1 and is obtained by dividing x^n-1 by the trinomial primitive $x^p + x^q + 1$, taking into account the smallest value of *n* ($n = 2p$-1).

A number that respects the relation $M_n = 2^n$-1, with *n* integer, is called Mersenne number, and if M_n is prime, then it is called prime number Mersenne, having useful properties for generators. A variant of GFSR based on linear recurrence is Twisted GFSR (TGFSR), proposed in 1992 by Matsumoto and Kurita to solve the problem of the initial values of the GFSR generator. In turn, TGFSR was improved, obtaining the MT algorithm (Mersenne Twister), with a period of 2^{19937}-1 and a uniform distribution in space with 623 dimensions (Prouff and Roche, 2011).

6. Feedback with Carry Shift Register

This generator (FCSR – Feedback with Carry Shift Register) is similar to an LFSR generator, having in common both the displacement register and the feedback function. FCSR also uses a transport register. The bits in the "tap" sequence are not XOR operated, but are added to the contents of the shift register, and the result of mod 2 represents the new bit, which will enter the shift register on the first position to the left of the register. The same result obtained at summation, is divided by 2, thus becoming the new content of the transport register (Fig. 2).

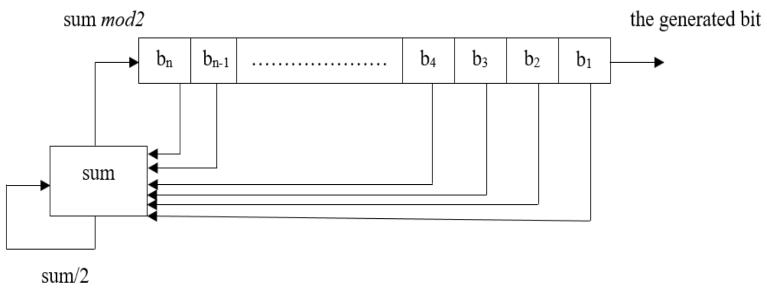

Fig. 2 – Feedback with Carry Shift Register.

The length of the transport register must be at least equal to $\log_2 t$, where t represents the number of bits in the "tap" sequence. The maximum period of an FCSR is q-1, where q (connection integer) is a prime number of the form:

$$q = 2q_1 + 2^2q_2 + 2^4q_4 + \ldots + 2^nq_n - 1 \quad (7)$$

for which 2 must be primitive root.

If the FCSR leads to a string of 0 or 1 of n bit, then the initial state must be rejected. Since the initial state of an FCSR is actually a key of a string digit, it follows that a generator based on an FCSR can have a lot of weak keys.

7. Non-Linear Feedback Shift Register

If in LFSR or FCSR a complex feedback function is implemented, a shift register can be obtained with non-linear feedback (NFSR). In this case there is no mathematical ground on which the analysis should be based, so in practice several problems may arise:

– the output string may contain a significant difference between bits with value 0 and those with value 1;

– the periods of the generator depend on the choice of the initial values;

– the maximum periods are much shorter than expected;

– despite the fact that the output string initially has a random appearance, it can end up cycling on a single value.

The difficulty of the theoretical analysis of the non-linearity of the generator is also valid in terms of cryptanalysis, resulting in few methods of attacking the string figures based on these generators.

8. Statistical Testing of Randomness

PRNGs used in cryptographic applications (especially in key generation) must meet certain quality conditions of the randomness of the generated bits. Their output must be unpredictable in the absence of any information on the input data. The degree of randomness of the generated string can be highlighted by statistical tests, to determine whether a generator is qualified to be used for cryptographic purposes. However, no set of such tests offers the absolute guarantee that a particular generator is suitable for a particular cryptographic application, meaning statistical tests cannot replace cryptanalysis.

The National Institute of Standards and Technologies (NIST) in the USA, has developed a battery of statistical tests of randomness, which aims to determine the deviations of a binary string from the quality of being random. However, the interpretation of these deviations must take into account as possible causes both the fact that the generator has design defects and the fact that the tested binary string has anomalies, which can be explained by the random appearance of the generated data.

A string of random bits can be interpreted as the result of tossing a coin with faces marked "0" and "1", in which each toss will produce a 0 or 1 with the same probability of 1/2. Moreover, coin tosses are independent of each other: the result of a toss must not in any way influence a future toss. Such a mechanism is a perfect generator of random numbers, which is used as a term of comparison in the evaluation of real pseudo-random number generators.

Numbers generated by a PRNG must be unpredictable, that is if the initial values are not known, the next number generated cannot be anticipated, no matter how many previously generated numbers are known. This property is called *unpredictability before*. It is also necessary that it is impossible to find the initial values, no matter how many generated values are known, which means *unpredictability after*. There must be no obvious correlation between the initial values and any of the values generated from them; each element in the generated string must appear as the output of an independent random event with probability (1/2).

To ensure unpredictability in advance, special care must be taken to obtain the initial values. The values produced by a PRNG are completely predictable if the initial values and the generation algorithm are known. As in many cases the algorithm is public, the initial values must be kept secret and chosen so that they cannot be deducted from the string it generates, as these initial values must also be unpredictable.

Various statistical tests can be applied to a string in order to compare and evaluate its random nature. The properties of a random string can be characterized and described in probabilistic terms. There are an infinite number of possible statistical tests, each assuming the presence or absence of a repetitive element which, if detected, would indicate that the string is not random. In conclusion, no battery of tests is complete, and the results of statistical tests should be interpreted with caution to avoid incorrect conclusions.

A statistical test verifies a specific null hypothesis (H_0), in the sense that the tested string is random. Associated with the null hypothesis is the alternative hypothesis (H_a) which assumes that the string is not random. For each test, accepting or rejecting the null hypothesis will lead to the conclusion that the generator produces or not, random values. The acceptance or rejection of the null hypothesis must be based on the choice and use of relevant statistics on randomness. In the case of randomness, such a statistic has a certain distribution of possible values. A theoretical reference distribution of this statistic in the case of the null hypothesis is determined by mathematical methods, and on its basis a critical value is established. During the test, a test value of the statistics is calculated based on the data, which is compared with the critical value. If the test value is greater than the critical value, the null randomness hypothesis is rejected, otherwise the null hypothesis is accepted.

Testing statistical hypotheses is a decision-making procedure with two possible results: either accepting hypothesis H_0 (data is random) or accepting hypothesis H_a (data is not random). Possible errors are:

- type I: the data is random, but hypothesis H_0 is rejected;
- type II: data is not random, but hypothesis H_0 is accepted.

The probability of a type I error is called the significance level of the test; can be fixed before the test and is denoted by α. For a given test, α is the probability that the data is random and the test indicates that the data is not random. The common value of the threshold α in cryptography is about 0.01.

The probability of a type II error is denoted by β. For a given test, β is the probability that the data is not random and the test indicates that the data is random. Unlike α, β is not a pre-set value, but can take different values corresponding to different ways in which a string may not be random and its calculation is difficult. One of the main purposes of a statistical test is to minimize the probability of a type II error. The probabilities α and β are dependent both on each other and on n (the length of the tested string), so that if two of these values are given, the third can be determined. In practice, the size n of the data sample and the level of significance α are established. A critical value is then chosen for a statistic so as to obtain the lowest possible value for the probability β.

Each test is based on a calculated value of the test statistics, which is a function of the data tested. If this value is denoted S and the critical value t, then:

α = probability of type I error = $P(S > t \,||\, H_0$ is true) = P(rejection H_0 | H_0 is true)

β = probability of type II error = $P(S \leq t \,||\, H_0$ is false) = P(acceptance H_0 | H_0 is false)

Statistical tests are used to calculate a p value, which expresses the degree of contradiction of the null hypothesis. For these tests, each p value represents the probability that a random number generator will produce a less random string than the tested string.

For a value $p = 1$, then the string is considered perfectly random, while for a value $p = 0$ the generated string is considered completely non-random. For tests, a significance level α is chosen; if the value $p \geq \alpha$, then the null hypothesis is accepted, so the string appears as random. If the value $p < \alpha$, then the null hypothesis is rejected, and the string appears as non-random. The parameter α indicates the probability of a type I error. Normally α is chosen in the range [0.001, 0.01]. A value of $\alpha = 0.001$ indicates the possibility that a string of 1000 strings may be rejected by the randomness test. For a value $p \geq 0.001$ a string will be considered random with a confidence threshold of 99.9% (Punnaiah *et al*., 2012).

9. Case Study

The Rijndael algorithm is already quite old and despite the fact that certain implementations have proven vulnerabilities following cryptographic analysis, it is still widely used in both hardware devices and software applications.

The study presents a complete cryptographic system (CCS), wich is composed of three elements: the key generator, the database and the encryption and decryption application. The key generator is an optimized variant of the MWC generator (multiply-with-carry).

The implementation has as premises the use of a computer network (Internet) in which encryption devices and storage media (computer stations and servers) are considered secure (by using firewall policies as well as antivirus and antispyware programs). If the computers on which the data encryption is performed were considered unsafe, the whole process would be useless.

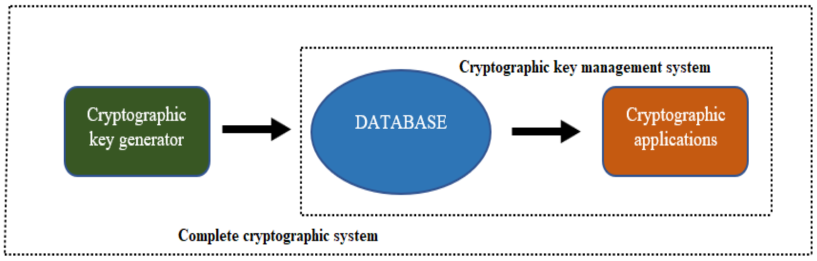
Cryptographic key management system
Cryptographic key generator
DATABASE
Cryptographic applications
Complete cryptographic system

Fig. 3 – Component relationship mode.

To test the application, two stations in a local network are sufficient, on which the encryption application is installed together with the database. No physical machines are required, and virtual machines can be used successfully (Microsoft and Oracle offer software for creating and using virtual machines).

For the transmission of the database containing the partial cryptographic keys to CCS users, it is recommended to use a data network considered secure, or tunneling encryption methods. In the absence of a network, mobile storage devices can also be used to load databases on the computer stations where this system is installed.

Algorithm for generating and storing partial keys:
for an integer 'base', b ≥ 2 and integer coefficients a_0, a_1 . . . a_r with a_0 prime with *b*, the MWC generator of order *r* and base *b* has the state σ:

$$\sigma = (x_{-1},\dots,x_{-r};c) \tag{8}$$

where $0 \le x_i' < b$ and c ∈ Z and *T* is a transformation rule:

$$T:\sigma \to \sigma' = (x'_{-1},\dots,x'_{-r};c') \tag{9}$$

for $i < -1$, $x_i' = x_{i+1}$. x'_{-1} and c' are unique solutions for:

$$a_0 x'_{-1} + c' b = \sum_{i=1}^{r} a_i x_{-i} + c \tag{10}$$

with $0 \le x_i' < b$. x'_{-1} and c'calculated as follows:

$$A = a_0^{-1} (\operatorname{mod} b) \tag{11}$$

It is performed as a whole in the range [0, *b*-1). Choose τ:

$$\tau = \sum_{i=1}^{r} a_i x_{-i} + c \tag{12}$$

and calculate:

$$x'_{-1} = (A\tau)(\operatorname{mod} b) \tag{13}$$

$$c' = (\tau - a_0 x'_{-1})/b \tag{14}$$

The value c, is the "carrier" (or "memory") of the state. The result of the state σ is OUT(σ)=x_{-r}, where σ takes values according to the (8) formula.

The normalized value is the real number x_{-r}/b and $c \in Z$ is an arbitrary value and therefore, an infinity of states and output sequences are obtained. In a finite interval $w^{-} \leq c \leq w^{+}$ there is a finite number of periodic states. For any initial state, the generator output is probably periodic, depending on how far the c value is from the initial segment.

A generalized scheme of the MWC generator, using as seed (initialization value) the clock of the system on which it runs, is shown in Fig. 4:

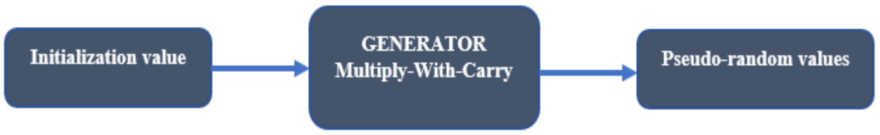

Fig. 4 – The classic MWC generator.

The XOR-MWC generator used in the composition of the CCS is composed of two independent MWC generators (MWC1 and MWC2) working in parallel, with a time delay to generate different outputs. These MWC generator outputs represent the inputs for a function that performs the XOR operation for the binary values of the generator outputs, thus obtaining a final binary value which, transposed to decimal, returns the corresponding character value in the extended ASCII code (Fig. 5).

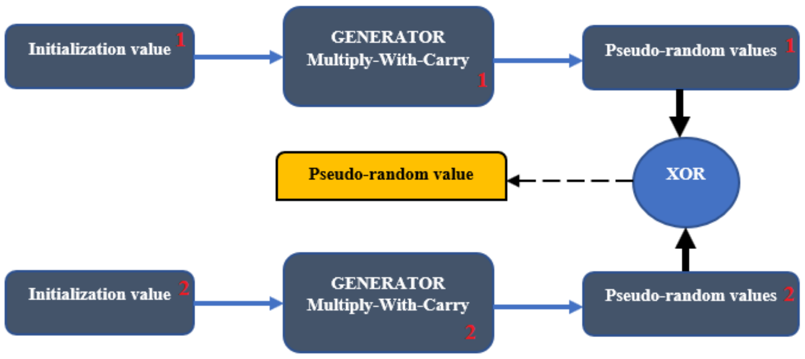

Fig. 5 – MWC generators working in parallel
With the XOR function implemented at the outputs.

Table 1
Cryptographic Keys Used

ID	Key_Used	Number of Generations
0	1e4a1b03d1b6cd8a174a826f76e009f4	0
1	8f32dfc68a0047a8e8bf1960e7ce79a0	1
2	c738fa985a1800dedb297eb41c038bfe	0

In the idea of using a unique key for each encryption operation, the algorithm checks in *Key_Used* column, hash of previously used keys. If the hash is found in the list, the process of generating the encryption key is resumed. In the *Number of Generations* column you can see how many times the algorithm generated the same one. Weak key hashes can also be entered in this table to avoid using them (Table 1). For example, the hash 1e4a1b03d1b6cd8a174a826f76e009f4 corresponds to the weak key 0000000000000000 (for ID = 0), while for the key *thisisasecretkey*, the hash is 8f32dfc68a0047a8e8bf1960e7ce79a0 (for ID = 1).

The values obtained can be entered in a MySQL database from which the cryptographic application, with the help of other pseudo-random generators, selects the partial keys of which the encryption key is composed.

Any attempt to obtain clear text from the encrypted text without holding the secret key, it is considered a cryptanalytic attack. Cryptographic analysis debates attack methods starting from minimal information about encryption keys, algorithms used, protocols authentication, clear text segments and the corresponding segments of the encrypted text, or just on the basis of one or a set of encrypted texts using the same algorithm.

In essence, it is trying to determine a vulnerable point of the algorithm, which could be operated using methods for which the search time is considerably shorter than the time necessary to check all possible key combinations (brute force attack).

There is still no cryptographic system that can be said to be complete safe, but those cryptosystems for which known attacks require a time too long to be considered practical.

This study presents the advantage offered by the use of the implemented encryption algorithm of a very large number of cryptographic keys. To increase operational security, both the working mode that the algorithm chooses at a given time and the encryption key are not to be presented to the user who uses the simplified interface of the cryptographic application.

10. Conclusions

Freedom of choice brings into question the concept of entropy, which measures the uncertainty of a system. When the number of elements from which the source can choose to create messages increases, the uncertainty or entropy increases in the same proportion. In contrast, when the elements are well organized and there is no possibility of random choices, the entropy is low. In other words, when the recipient knows the probability of the messages, the entropy or the amount of information is low. The value of a specific segment of information depends on the probability of its occurrence. In general, when the probability of an item appearing in a message increases, its informational value decreases in the same proportion.

In 1965 A.N. Kolmogorov introduced the idea that the complexity of a data string can be defined by the length of the shortest binary program made to compute the string. So, the complexity of the data represents the length of their minimum description, specifying the final compressibility of the data. The Kolmogorov complexity of a string, also known as algorithmic information theory, is approximately equal to the entropy defined by C.E. Shannon for the same string, thus reaching a unification of the theory of descriptive complexity with information theory. These theories aim to provide a means of measuring information, and this association was possible due to the fact that Kolmogorov used in the demonstrations of theories the same fundamental element: the *bit*.

The study presents this new system both to those interested in cryptography, for analysis and attempts to demonstrate the insecurity of the solution, and to potential users from the commercial environment. Demonstrating performance as well as security, recommends the complete cryptographic system as a new solution for protecting information in computer networks, with the amendment that for information with a high degree of confidentiality, it is necessary to adopt additional security solutions. The constant evolution of cryptanalytic methods, supported by continuous technological development, determines the entire community of cryptologists to constantly look for ways to optimize current security solutions, or to develop new ones.

This paper marks a vital topic in Cryptography World: PRNG, highlighting the large classes of pseudo-random number generators and their functional properties. Additionally, the effectiveness of randomness statistical tests is pinpointed, which is a basic method of analyzing the security of an complete cryptographic system.

REFERENCES

Kelsey J., Schneier B., Wagner D., Hall C., *Cryptanalytic Attacks on Pseudorandom Number Generators*, Springer-Verlag, 1998.

Meyer C.H., Tuchman W.L., *Design Considerations for Cryptography*. Proceedings of the NCC, AFIPS Press, **42**, 594-597, Nov.1979.

Muhammad H., Qasim R., Qasim S.M., *Efficient Hardware Realization of Advanced Encryption Standard Algorithm Using Virtex-5 FPGA*, International Journal of Computer Science and Network Security, **9**, *9*, 201-205, 2009.

Paar C., Pelzl J., *Understanding Cryptography*. *A Textbook for Students and Practitioners*, Springer-Verlag Berlin Heidelberg, 2010.

Perrig A., Stankovic J., Wagner D., *Security in Wireless Sensor Networks*, Communication of the ACM, **47**, *6*, 53-57, 2004.

Prouff E., Roche T., *Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols*, Cryptographic Hardware and Embedded Systems - CHES 2011, Lecture Notes in Computer Science, **6917**, 63-78, Springer Berlin / Heidelberg, 2011.

Punnaiah S., Ganesh G., Beechu N., *Hardware Optimal Power Calculation for the Cryptography AES Algorithm Using Clock Gating Technique*, Special Issue of International Journal of Engineering Science & Advanced Technology, Jan-Feb, 2012.

*** NIST Special Publication 800-57, *Recommendation for Key Management - Part 1*, March, 2007.

EFICACITATEA TESTĂRII STATISTICE A CARACTERULUI ALEATORIU AL DATELOR FOLOSITE ÎNTR-UN SISTEM CRIPTOGRAFIC COMPLET

(Rezumat)

Scopul acestei lucrări este de a descrie clasele mari de generatoare de numere pseudo-aleatoare (generatorul liniar congruent și generatoarele cu registre de deplasare) dar și proprietățile lor funcționale. Teoria generatoarelor de numere pseudo-aleatoare este necesară în definirea metodelor teoretice care stau la baza proiectării și analizei algoritmilor de șiruri. Acei algoritmi, care au caracteristicile aleatoare necesare, sunt adoptați în prezent pentru a genera chei criptografice, utilizate în sisteme criptografice complete (SCC). Lucrarea se încheie cu un studiu de caz privind eficacitatea testării statistice a caracterului aleatoriu a datelor, una dintre metodele fundamentale de analiză a securității unui SCC.