New Symmetric Key Cipher Based on Quasigroup

Umesh Kumar, Aayush Agarwal, and V. Ch. Venkaiah

Abstract Stream ciphers that use the XOR function for mixing the plaintext and the keystream are vulnerable to attacks such as known-plaintext attack and insertion attack. To overcome such shortcomings of the existing ciphers, we hereby propose a new stream cipher that uses AES. The proposed cipher is based on a large-order quasigroup. It is resistant to brute force attack, due to the exponential number of quasigroups of its order. It is also analyzed against the chosen-ciphertext, chosen-plaintext and known-plaintext attacks, and it is found to resist these attacks. The output of the cipher is subjected to various statistical tests, such as the NIST-STS test suite, and the results show a high degree of randomness of the ciphertext. Hence, it is resistant to correlation-type attacks.

Keywords AES · NIST-STS test · Latin squares · Quasigroup · Stream cipher

1 Introduction

The need for securing the data has been increasing day by day and with that there has been tremendous need for new encryption/decryption methods. A cryptosystem typically consists of an encryption algorithm, a decryption algorithm and a key generation algorithm. A cryptosystem may be analyzed to determine either the message or the secret key used.

Various cryptographic algorithms such as DES and AES are designed to achieve message confidentiality. Depending on how the data is encrypted, ciphers can be of two types: stream ciphers and block ciphers. Stream ciphers encrypt data bit by bit or byte by byte using a keystream which is as long as the plaintext and is generated

U. Kumar (✉) · V. Ch. Venkaiah
School of Computer & Information Sciences, University of Hyderabad, Hyderabad, India
e-mail: kumar.umesh285@gmail.com

V. Ch. Venkaiah
e-mail: vvcs@uohyd.ernet.in

A. Agarwal
Manipal Institute of Technology, Manipal, India

D. P. Agrawal et al. (eds.), *Cyber Security, Privacy and Networking*, Lecture Notes in Networks and Systems 370, https://doi.org/10.1007/978-981-16-8664-1_8

using a secret key. The keystream used for stream ciphers cannot be used again since stream ciphers are vulnerable to reused key attack [19, 20]. These ciphers are also vulnerable to attacks such as a known-plaintext attack and insertion attack since they use XOR function with plaintext and keystream [2]. The proposed algorithm is resistant to both these attacks since it uses quasigroup operation in place of XOR function and the keystream can be reused for encryption. Block ciphers encrypt a fixed size of data called blocks at one time. The size of the block depends on the encryption algorithm. DES and AES are examples of block ciphers. The DES was previously used as the standard for encryption. It was vulnerable to attacks such as brute force attack [4] because of its small key size of 56 bits, chosen-plaintext attack [3] and known-plaintext attack [11]. Hence, a new standard was required, and therefore, DES was replaced by AES [16].

AES is currently the standard for encryption and decryption. It is known to withstand various attacks. Though AES is highly secure, there is a need for an algorithm that is stronger than AES. Required promise comes from a mathematical object called a quasigroup.

Quasigroups are very simple non-associative algebraic structures. Since the number of quasigroups grows exponentially with the size, they make an important case for the design of cryptosystems. Using quasigroups, simple and efficient encryption algorithms can be produced. One of the factors that favor quasigroups is that they can be efficiently stored. Previous works [10, 15] that use quasigroups in the design of secure systems are vulnerable to the chosen-plaintext and chosen-ciphertext attacks [9, 21]. The proposed algorithm resists these attacks and hence is secure.

This paper proposes a new cipher algorithm for the encryption/decryption of the messages that replaces the XOR operation of the conventional stream ciphers by the quasigroup operation and its inverse. It is a stream cipher and uses AES for its keystream generation. In fact, any secure pseudorandom number generator such as CRT-DPR 4 [1] can be employed for the generation of the keystream. However, we choose to describe the proposed stream cipher using the AES 256. The security of the cipher is analyzed, and the randomness of the obtained ciphertext is tested. It was found that the proposed system satisfied all the required properties.

In this context, we would like to mention that similar scheme without any analysis was proposed in [5]. This scheme is based on deterministic finite automaton and uses Latin squares for encryption and decryption.

2 Preliminaries

In this section, a brief overview of quasigroups and their uses in the design of cryptosystems is discussed.

2.1 Latin Squares

A Latin square of order m is a $m \times m$ array in which the entries are taken from a finite set S and the symbols are arranged in such a way that each symbol occurs only once in each column and only once in each row.

Example 1: The following array (Table 1) is an example of a Latin square of order 4. It can be seen that each symbol occurs only once in each row and each column. The number of Latin squares of order n increases greatly as n increases. An estimate of the number of Latin squares of order n is given in [6, 7]

$$\prod_{k=1}^{n}(k!)^{\frac{n}{k}} \geq LS(n) \geq \frac{(n!)^{2n}}{n^{n^2}}, \tag{1}$$

where $LS(n)$ denotes the number of Latin squares of order n. For $n = 2^k$, $k = 7, 8$ the estimated number is:

$$0.164 \times 10^{21091} \geq LS(128) \geq 0.337 \times 10^{20666}, \tag{2}$$

$$0.753 \times 10^{102805} \geq LS(256) \geq 0.304 \times 10^{101724}. \tag{3}$$

2.2 Quasigroup

A quasigroup $Q =< S, * >$ is a groupoid which has the following properties:

(i) If a pair $a, b \in S$, then $a * b \in S$ (Closure property).
(ii) $\forall\ a, b \in S$, there exists unique $x, y \in S$ such that
$a * x = b$ and $y * a = b$.

Let $Q = <S, *>$ be a quasigroup. Let $\backslash$ (Left Inverse) and / (Right Inverse) be two operations on Q such that

$$a * b = c \Leftrightarrow b = a \backslash c, \tag{4}$$

$$b * a = c \Leftrightarrow b = c/a, \tag{5}$$

Table 1 Latin square of order 4

0	1	2	3
1	2	3	0
2	3	0	1
3	0	1	2

Table 2 Operation table of Q

*	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Table 3 Operation table of LIQ

\	0	1	2
0	0	1	2
1	2	0	1
2	1	2	0

Table 4 Operation table of RIQ

/	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

where a, b and c are elements of the quasigroup Q. Then, $LIQ = <S, \backslash>$ and $RIQ = <S, />$ are called the Left Inverse and Right Inverse quasigroups of the quasigroup $Q =< S, * >$, respectively.

Example 2: Consider the quasigroup $Q = <S, *>$ with $S = \{0, 1, 2\}$, let its operation table be as in Table 2. Then the corresponding Left Inverse quasigroup is $LIQ = <S, \backslash>$ whose operation table is given in Table 3. And the corresponding Right Inverse quasigroup is $RIQ = <S, />$ whose operation table is given in Table 4.

Properties (i) and (ii) of quasigroups enforce the operation table of a quasigroup to be a Latin square. Therefore, the number of quasigroups is same as that of the number of Latin squares. Therefore, Eq. 1 is also an estimate of the number of quasigroups. Hence, an estimate of the number of quasigroups of order 256 is given by Eq. 3.

2.3 Encryption and Decryption Using Quasigroups

Quasigroups are used for encryption purposes. Let $PT = p_1\, p_2\, p_3 \ldots p_n$, $K' = k_1\, k_2\, k_3 \ldots k_n$ and $CT = c_1\, c_2\, c_3 \ldots c_n$ denote the plaintext to be encrypted, keystream to be used for encryption, and the resulting ciphertext, respectively. Then a way of encrypting PT with the keystream K' to obtain the corresponding CT is as follows: $c_1 = p_1 * k_1\,, c_2 = p_2 * k_2\,, \ldots\,, c_n = p_n * k_n\,.$

For decryption of the ciphertext, the Right Inverse quasigroup $RIQ = <S, />$ is used. The following procedure decrypts the ciphertext CT, obtained from the foregoing encryption procedure, using the keystream K' to arrive at the corresponding plaintext PT. $p_1 = c_1/k_1$, $p_2 = c_2/k_2$, . . . , $p_n = c_n/k_n$.

Note that the conventional stream ciphers that exist in the literature use XOR operation in place of $*$ and / operations. The algorithm works on characters of 1 byte (8 bits) each. Hence, all p_i, k_i, c_i are characters. Similarly, the message can be encrypted and decrypted with the quasigroup and its Left Inverse quasigroup, respectively.

Example 3: Consider the plaintext as $PT = p_1\, p_2\, p_3 = 021$ and the keystream to be $K' = k_1\, k_2\, k_3 = 011$. Then applying the foregoing encryption procedure using the quasigroup Q (Table 2) of example 2, we have the ciphertext as

$$CT = c_1\, c_2\, c_3 = 002.$$

To decrypt the ciphertext, Right Inverse Quasigroup RIQ (Table 4) is used. The recovered plaintext will be

$$PT = p_1\, p_2\, p_3 = 021.$$

Other methods of encryption/decryption using quasigroups are addressed in [7].

2.4 Advanced Encryption Standard

Advanced Encryption Standard or AES is a symmetric key block cipher. It has 128-bit data with 128/192/256-bit key. AES is widely used for encryption/decryption process in various fields. It has fast implementation in both software and hardware. AES is used in various modes of operation such as Cipher Block Chaining (CBC) mode, Cipher Feedback (CFB) mode , Output Feedback (OFB) and Counter (CTR) mode. Each mode of operation has their advantages and disadvantages [8, 18].

3 Proposed Cipher Algorithm Structure

3.1 Quasigroup Selection

The main principle used is the quasigroup operation of 1-byte plaintext characters with 1-byte random keystream characters to produce 1-byte ciphertext characters at a time. Any order quasigroup can be used in the algorithm. Since as the order increases the number of quasigroups grows exponentially, higher-order quasigroups are preferred. The order of 256 is used in our proposed algorithm because all the

ASCII values can be represented in 8 bits and each character has an integer value of 0 to 255. Also, from Eq. 3 we can see that the number of quasigroups of order 256 is very large and therefore practically impossible to guess the correct quasigroup selected. That is, it is impossible to determine the quasigroup $Q = <S, *>$, where $S = \{0, 1, 2, \ldots, 255\}$ and $*$ is the binary operation used in our algorithm. The issue of generating large-order quasigroups is addressed in [13, 14].

3.2 Keystream Generation

The encryption/decryption algorithm uses a keystream of size as long as the message. To generate such a long keystream, the algorithm uses AES. Using an initialization vector (IV), a secret key (K), and a $Counter$, the AES algorithm produces keystream of required length. The keystream generated is represented by K'. Since the encryption algorithm requires the keystream to be random, AES ensures this. Block diagram of the keystream generation is given in Fig. 1.

Note that the secret key K used in AES is different from the keystream K', generated using AES. Each round generates 16 bytes of keystream and is repeated until the keystream size is the same as that of the plaintext. This generates the keystream $K' = k_1\, k_2\, k_3 \ldots k_n$, where each k_i is a 1-byte character and will be used to encrypt 1-byte character of the plaintext. The keystream generation can use AES in encryption or

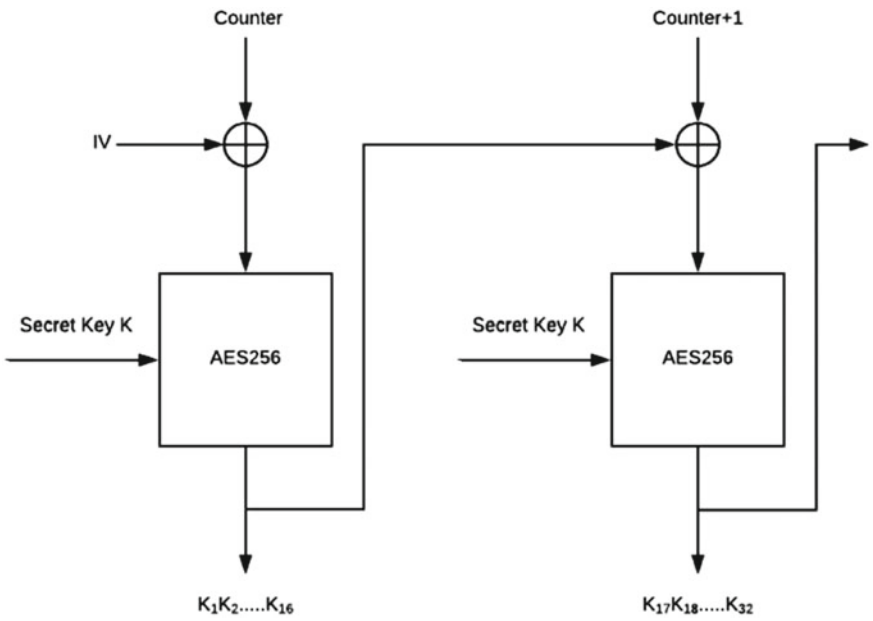

Fig. 1 Keystream generation using AES 256

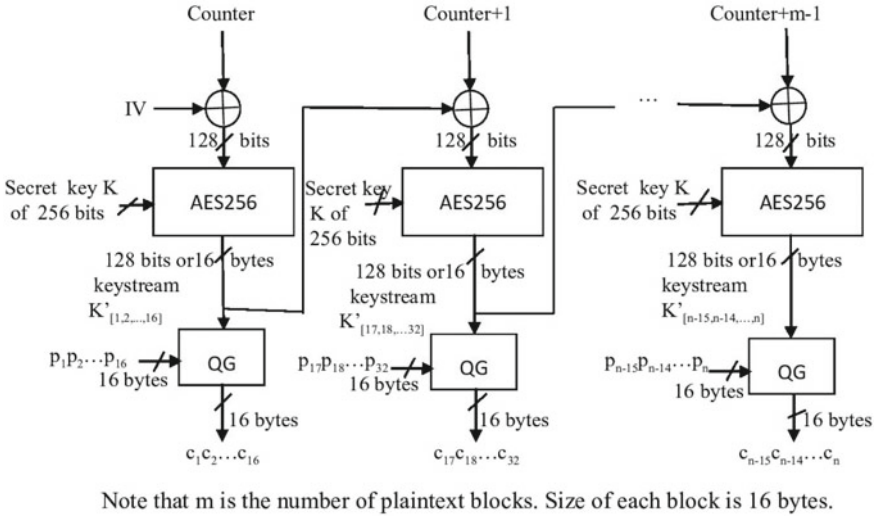

Fig. 2 Encryption algorithm

decryption mode. The keystream generation procedure is used in both encryption and decryption processes of the proposed stream cipher. The secret key size can be 128/192/256 bits. The AES algorithm used in the proposed cipher is AES 256 encryption algorithm.

3.3 Encryption Algorithm

The encryption algorithm uses the method described earlier. The main principle used is as follows: For plaintext, $PT = a_1\, a_2\, a_3 \ldots a_n$ and the generated keystream, $K' = k_1\, k_2\, k_3 \ldots k_n$ the ciphertext, $CT = c_1\, c_2\, c_3 \ldots c_n$ is produced as $c_1 = a_1 * k_1, \quad c_2 = a_2 * k_2, \ldots, c_n = a_n * k_n$, where '$*$' is the operation of the chosen quasigroup. The algorithm can encrypt 16 bytes of plaintext in one iteration. Block diagram of the encryption algorithm is given in Fig. 2.

3.4 Decryption Algorithm

The decryption algorithm uses the inverse of the quasigroup $Q =< S, * >$ chosen for encryption. Depending on the operation used in encryption either the Left Inverse or the Right Inverse can be used to generate the plaintext PT. The main principle used is as follows: For ciphertext, $CT = c_1\, c_2\, c_3 \ldots c_n$ and the keystream, $K' = k_1\, k_2\, k_3 \ldots k_n$ the corresponding plaintext $PT = a_1 a_2 a_3 \ldots a_n$ can be recov-

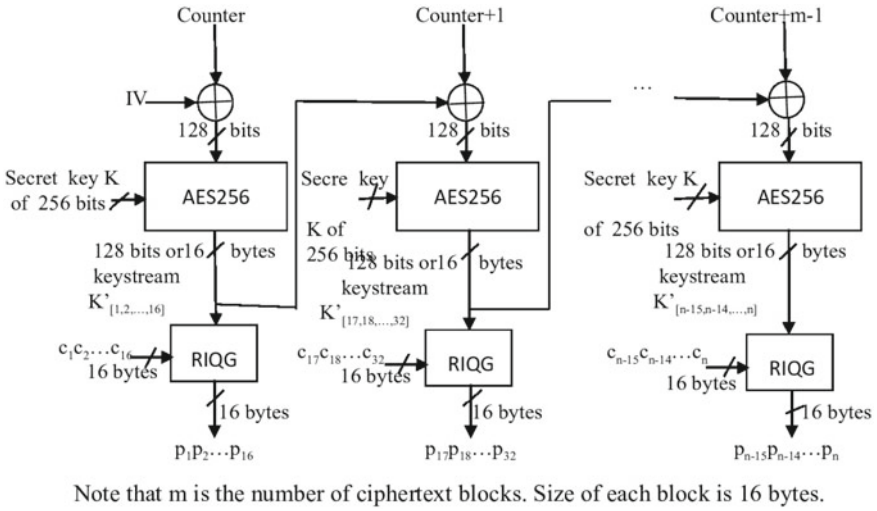

Fig. 3 Decryption algorithm

ered as $a_1 = c_1/k_1, a_2 = c_2/k_2, \ldots, a_n = c_n/k_n$, where '/' is the Right Inverse of the operation of the chosen quasigroup. The algorithm uses the same secret key as that of the encryption algorithm. It can decrypt 16 bytes of ciphertext in one iteration. The cipher algorithm uses the Right Inverse quasigroup $RIQ = <S, />$ for decryption. Block diagram of the decryption algorithm is given in Fig. 3.

4 Security Analysis

The key elements for the algorithm consist of the secret key K, the initialization vector IV, the $Counter$, and the selected quasigroup Q of order 256. Since the secret key size is 256 bits, it is resistant to brute force approach since there are 2^{256} possible keys. Since the keystream generation for the proposed stream cipher is similar to AES in Cipher Block Chaining (CBC) mode encryption, it is already secure from attacks. Hence, the keystream K' generation is safe from various attacks.

The other element is the quasigroup. Since the order of used quasigroups is 256, the number of possible quasigroups is at least

$$0.304 \times 10^{101724}.$$

Therefore, it is impossible to determine the selected quasigroup. The general method given in [10, 15] used for encryption using quasigroups is found to be vulnerable to chosen-ciphertext attack and chosen-plaintext attack [9, 21], whereas the proposed algorithm is resistant to these attacks because of the following argument: Suppose the

cryptanalyst chooses the ciphertext $CT = c_1\, c_2\, c_3 \ldots c_n$, and obtains the plaintext $PT = p_1\, p_2\, p_3 \ldots p_n$, corresponding to the chosen-ciphertext and tries to determine the quasigroup Q employed in the encryption-decryption process. The adversary, then, for the keystream K', must solve the system of equations:

$$\begin{aligned} c_1 &= p_1 * k_1\,, \\ c_2 &= p_2 * k_2\,, \\ &\ldots \\ c_n &= p_n * k_n\,, \end{aligned}$$

where

$$K' = k_1\, k_2\, k_3 \ldots k_n$$

is the generated keystream. This system has as many solutions as there are quasigroups of order 256. Hence, determining the quasigroup makes it practically impossible. Therefore, the cipher is resistant to chosen-ciphertext attack. Even if the attacker has knowledge of both the plaintext and the corresponding ciphertext, the cipher still remains secure as determining K' still requires finding the selected quasigroup. Therefore, it is resistant to known-plaintext attack as well. Similar argument shows that the system is secure from the chosen-plaintext attack also. The decryption algorithm uses the Right Inverse (same can be achieved using Left Inverse LIQ quasigroups as well) of the quasigroup Q denoted by RIQ. The cipher encrypts/decrypts a stream of characters; hence no padding of plaintext is required and hence is safe from padding oracle attacks which are a known threat to ciphers where padding is required as shown in [12].

Note that the computational complexity is exactly the same as any stream cipher that uses AES for keystream generation and XOR function for mixing the plaintext and the keystream. Except that it differs in the following: Existing ciphers use XOR function for every bit of the message; whereas our cipher works on a character by character and uses one quasigroup operation for every character of the message. The space complexity of our cipher is also the same as that of the existing ciphers, except that our cipher needs one quasigroup table of 256×256. That is, our cipher needs 64k bytes of extra space.

4.1 Statistical Test for Randomness

The obtained ciphertext after encryption with the proposed algorithm passes various tests of randomness. One such battery of tests is the NIST-STS test suite [17]. Each test of the NIST-STS package gives a P-value which lies between 0 and 1 (both included) and indicates Success/Fail status. The P-value is the probability that a perfect random number generator would have produced a less random sequence than the one being tested [17]. For these tests, we have chosen the significance level (α)

Table 5 Parameters for the NIST-STS test

Tests	Block length(m)
Block frequency test	128
Non-overlapping template test	9
Overlapping template test	9
Approximate entropy test	10
Serial test	16
Linear complexity test	500

Table 6 Results of the NIST-STS test

Tests	*P*-value
Frequency	0.507678
Block frequency	0.513950
Cumulative sums-forward	0.675720
Cumulative sums-backward	0.530005
Runs	0.542138
Longest run	0.552834
Rank	0.443481
Discrete Fourier transform	0.500707
Overlapping template	0.479753
Approximate entropy	0.471052
Serial-1	0.550659
Serial-2	0.553913
Linear complexity	0.576939

to be 0.01 and the other parameters as shown in Table 5. For the randomness of a sequence, we compare the P-value of a sequence to a significance level (α). If P-value $\geq \alpha$, then the sequence is considered to be random, otherwise non-random. We run each of these tests over 20 obtained ciphertext sequences. The size of each sequence is 200 KB. Table 6 shows the average P-value of NIST-STS tests. We can observe that the P-value of each test crosses the significance level ($\alpha = 0.01$), so, we conclude that the obtained ciphertext sequences are random.

5 Conclusion

A new stream cipher algorithm that uses the existing cipher for keystream generation and a quasigroup for encryption and decryption is proposed. It masks the weaknesses of stream ciphers and adds extra security. The new cipher is resistant to most common

attacks such as chosen-ciphertext attack, chosen-plaintext attack and known-plaintext attack. The randomness of the obtained ciphertext is analyzed by the NIST-STS test suit, and we noted that the new cipher produces high degree of randomness of the ciphertext. The keystream in our proposed cipher as in the case of any stream cipher can be preprocessed and kept ready, and then the encryption/decryption can be performed parallelly. Storing of the generated key can be a challenge for very large messages since the generated key is as long as the message. Our cipher can be deployed in all the applications of stream ciphers such as secure wireless connection.

References

1. Barker E, Kelsey J (2007) Recommendation for random number generation using deterministic random bit generators. Technical report, NIST (revised)
2. Bayer R, Metzger J (1976) On the encipherment of search trees and random access files. ACM Trans Database Syst (TODS) 1:37–52
3. Biham E, Shamir A (1993) Differential cryptanalysis of the data encryption standard. Springer, Berlin
4. Diffie W, Hellman ME (1977) Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer 10:74–84
5. Domosi H (2017) A novel stream cipher based on deterministic finite automaton. Ninth workshop on non-classical models of automata and applications (NCMA 2017), pp 11–16
6. Jacobson MT, Matthews P (1996) Generating uniformly distributed random Latin squares. J Combin Des 4:405–437
7. Koscielny C (2002) Generating quasigroups for cryptographic applications. Int J Appl Math Comput Sci 12:559–570
8. Lipmaa H, Wagner D, Rogaway P (2000) Comments to NIST concerning AES modes of operation: CTR-mode encryption
9. Malyutina NN (2019) Cryptanalysis of some stream ciphers. Quasigroups Rel Syst 27:281–292
10. Markovski S, Gligoroski D, Andova S (1997) Using quasigroups for one-one secure encoding. In: Proceedings of VIII Conference on logic and computer science "LIRA", vol 97, pp 157–162
11. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Workshop on the theory and application of of cryptographic techniques. Springer, Berlin, pp 386–397
12. Paterson KG, Yau A (2004) Padding oracle attacks on the ISO CBC mode encryption standard. In: Cryptographers' track at the RSA conference. Springer, Berlin, pp. 305–323
13. Petrescu A (2007) Applications of quasigroups in cryptography. In: Proceedings of interdisciplinarity in engineering. TG-Mures, Romania. Academic Press
14. Petrescu A (2009) A 3-quasigroup stream cipher. In: The international conference interdisciplinarity in engineering INTER-ENG. Editura Universitatii "Petru Maior" din Tirgu Mures, p 168
15. Petrescu A (2010) n-quasigroup cryptographic primitives: stream ciphers. Stud Univ Babeş-Bolyai Inf 55 [On table of contents: Anul LIV]:27–34
16. Rijmen V, Daemen J (2001) Advanced encryption standard. In: Proceedings of federal information processing standards publications. National Institute of Standards and Technology (NIST), pp 19–22
17. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E (2010) A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST, special publication 800-22, revision 1a
18. Schneier B (2007) Applied cryptography protocols, algorithms, and source code in C. Wiley
19. Stallings W (2006) Cryptography and network security, 4th edn. Pearson education, Inc., India

20. Stinson D (1995) Cryptography: theory and practice. CRC Press, CRC Press LLC
21. Vojvoda M (2004) Stream ciphers and hash functions-analysis of some new design approaches. Ph.D. thesis. Slovak University of Technology