A 64 BITS ROTOR ENHANCED BLOCK CIPHER (REBC3)

Ahmed ElShafee[1]

[1]Department of Computer Networks, Ahram Canadian University, 6th October City, Egypt
aelshafee@ieee.org

ABSTRACT

This paper gives a new proposed cryptosystem (REBC3) that is designed to take advantages of the new generation of 64bits microprocessors which commercially known as x64 systems. The old version REBC2, which was published in Africon 2007. REBC2 was basically developed for the 32bits microprocessors which is commercially known as x86 systems. REBC3 like REBC2 use the concept of rotor enhanced block cipher which was initially proposed by the author in [NRSC 2002] on the first version of REBC. REBC2 used the same concept from a another point of view, which is using rotors to achieve two basic cryptographic operations; permutation, and substitution. Round key is generated using rotor too, which is used to achieve ciphertext key dependency. To enhance non-linearity and to resist linear cryptanalysis, REBC3 has a variable block, and key lengths. Each round has its own block length which depends on round the key and round key length. Dependency is based upon the previous round generated key. Rotors implemented using successive affine transformation. The 32 bits version was proposed in KAMFEE cipher, then the 64bits version was proposed in KAMFEE-X64 cipher. This achieved memory-less, normalized ciphertext statistics, and small processing speed trend. The strength of this system is compared with the REBC2 and RIJNDAEL (AES) ciphers. REBC3 cipher gives excellent results from security characteristics and statistical point of view of. So authors suggests to use REBC3 in the area of banking and electronic fund transfer.

KEYWORDS

X64 systems, block cipher, rotor cipher, brute force attack

1. INTRODUCTION

Network Security is becoming more and more crucial as the volume of data being exchanged on the internet is growing. Based on that, the security involves many important aspects like; Confidentiality, message authentication, integrity and non – repudiation. Cryptography, which is the science, concerned with how to protect information and Cryptanalysis which is the science concerned with how to unsecure information that is thought to be protected and secured by cryptographic means. Encryption is the process of transforming information (Plaintext) into unintelligible form (Ciphertext). Thus, cryptology is an active science that is in continuous study and big challenges. It always seeks to overwhelm the increasing computer speed/architecture.

The 64-bit computing implies computing on a 64-bit processor which characterize a processor's data stream. 64-bit wide memory buses imply that the address lines are 64 bit wide and virtual addressing mechanisms use 64 bit sized pointers. Although we hear the term "64-bit code," it actually refers to code that operates on 64-bit data. It also implies that by special instructions (or modes) one can access the 64-bit registers or computing capability.

DOI : 10.5121/ijnsa.2013.5206

The theoretical limit of memory in 64-bit systems is 2^{64} bytes which is 16 ExaBytes (1021). Although systems with such a huge amount of memory will be prohibitively expensive, the current systems provide 40 bits of physical ad dress space or 48 bits of virtual address space. Large-number math is an obvious advantage offered by a 64-bit processor. A 32-bit processor can handle the integer range of -2.1 billion to +2.1 billion (approximately). However, it is not to say that a 32-bit processor cannot handle a 64-bit number. In old 32-bit systems a number larger than 32-bits can be stored in multiple memory locations as lower and higher 32-bits and the software can be programmed to treat it as a single 64-bit number (long long). But these are at best workarounds and are not fast. On the other hand, a 64-bit processor will be able to handle bigger numbers without having to resort to the workarounds and hence are inherently faster.

It is quite obvious that applications written to exploit 64-bit architectures can gain from faster access to data, availability of 64-bit resources like 64-bit and 128 bit registers, 64-bit pointers and larger data types. Applications can also have larger file caches and map large process data in virtual address space and can support larger files using standard system library calls, etc.

Encryption systems are often grouped into families. Common families include symmetric systems (e.g. AES) and asymmetric systems (e.g. RSA), or may be grouped according to the central algorithm used (e.g. elliptic curve cryptography). As each of these is of a different level of cryptographic complexity, it is usual to have different key sizes for the same level of security, depending upon the algorithm used. The actual degree of security achieved over time varies, as more computational power and more powerful mathematical analytic methods become available. For this reason cryptologists tend to look at indicators that an algorithm or key length shows signs of potential vulnerability, to move to longer key sizes or more difficult algorithms. Focusing on symmetric ciphers, block length is an important parameter that characterize the cipher.

Now in the presence of 64-bits systems, cryptosystems has to gain advantages if these systems by using 64-bits module operations [2] [3] (instead of 32 bits module operations [1]), and 64-bits basic block (instead of 32 bits basic block [1]) without any effect of cryptosystem processing speed.

2. REBC3 OPERATIONAL STRUCTURE

2.1. Overview

Cryptosystem designer focuses in three main items while designing an new cryptosystem those are; substitution, permutation [4], and key dependency to achieve privacy of cryptosystem per user. All modern well-known cryptosystems, follow the same rules with little bits of variations depending on designer point of view. For example Rivest [5] used rotation, with their rotating order depending on the encrypted data itself. RIJNDAEL [6] used simple mathematical operations to achieve ordinary permutation and substitution. The author in his lately published cryptosystems like; REBC [9], KAMFEE [8], CYCLONE [7], ROTRIX [10], and REBC2 [11], URESC [12], [13] rotors are always used.

other block ciphers that use the same concept of rotor to enhance their ciphertext characterestics are SCER [14], KAMKAR [15], and RTCKP [16].

The proposed REBC3 is an enhanced version of REBC2 [11] which was published by authors in 2002. REBC3 has a 64 bits basic block (Rotor Enhanced Block Cipher) which make use of 64-bits systems to enhance cryptosystem performance from security and speed point of view. This section describes the REBC3 structure, and characteristics. The following section is a

comparison between RIJNDAEL and REBC3 from speed, security and memory requirements point of view.

2.2. Key generation

REBC3 has three different standard key lengths, 8 bytes (64 bits), 16 bytes (128 bits), and 32 bytes (256 bits). "User key" is expanded to the nearest larger standard length. A small rotor which is called key-rotor consists of 8 cylinders, each cylinder contains 256 elements is used to expand user key and then used to generate the key of each round Figure 1 shows first round key expansion and generation from user key.

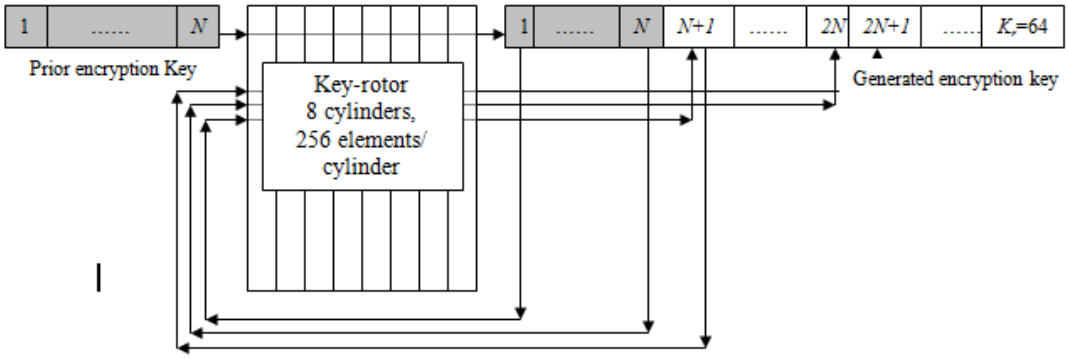


Figure 1. 1st round key generation from user key

Considering K_r to present the encryption key of round r, and K_{r-1} is to present the encryption key of round (r-1), Figure 2 shows the key generation process.

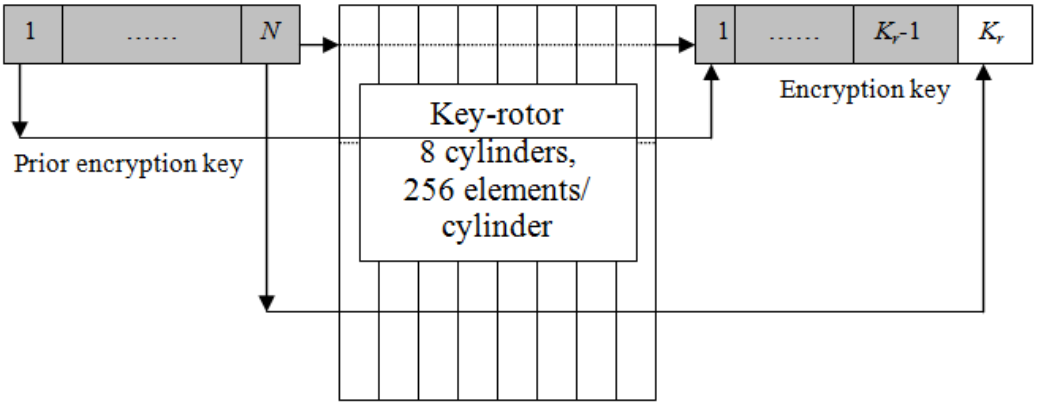


Figure 2. round key generation

Key rotor is implemented using successive affine transformations as shown in the following equation

$$y_i = ((a_j \times x_i) + b_j + c_j) \bmod 256 \quad \text{..... (1)}$$

Where x_i ; is a user key character
i: presents the character position in user key
y_i ; is an expanded key character

a_j: rotor pre-selected constant (should be reversible module 256)
j: presents wheel number of rotor
b_j: rotor pre-selected constant
c_j: cylinder rotation step

2.3. Key length dependency

The length of the proposed system basic block length is 64-bits. REBC3 has three different block lengths, which are multiple of basic block length. So REBC3 of 64 bits block length consists of ONE basic blocks, REBC3 of 128 bits block length consists of two basic blocks, and 265 bit block consists of four different basic blocks.

The expanded key length of REBC3 will select few parameters of cipher structure, those are;

1. Block length, which has the same length of expanded key length
2. Number of rounds depends on key length too, table (1), shows the relation between user key, and REBC3 structure.

Table 1. REBC3 number of rounds

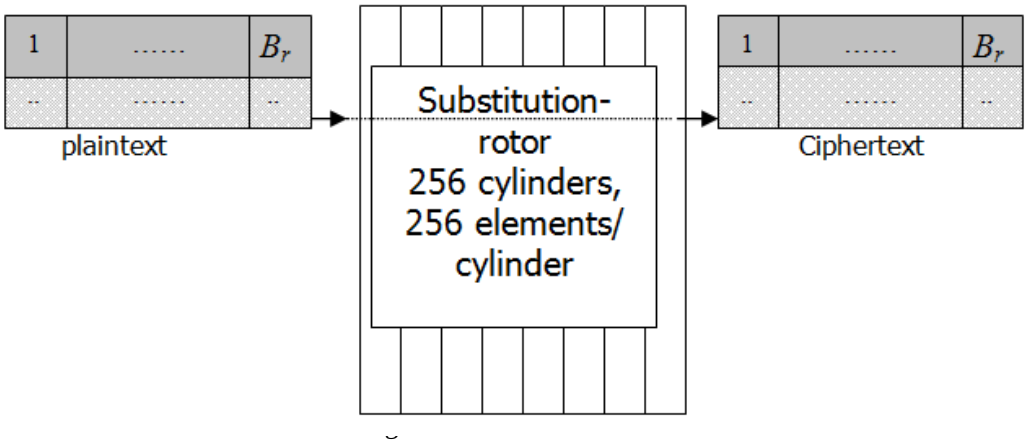
	REBC3, 64 bits	**REBC3, 128 bits**	**REBC3, 256 bits**
Key length (bytes)	8	16	32
# rounds	2	4	8

2.4. Single round structure

REBC3 round consists of three consequent steps. Step one is substitution, second step is permutation, and the third step is the key dependency.

2.4.1. Substitution

Substitution is a nonlinear operation that enhance the security of the block cipher. Substitution is performed using a rotor with 256 cylinders, each cylinder containing 256 characters presenting the ASCII characters. That rotor is called substitution rotor as shown in Figure 3 Each byte of plaintext is successively substituted in each cylinder of rotors. After each encryption process for a plaintext byte, the rotor cylinders are rotated.



From implementation point of view, substitution rotor is impossible (each cylinder contains 256 bytes, the whole rotor contains 256^{256}=3.23 x 10^{616} bytes), so a 8 bits affine transformation is used to dynamically generate sbox.

$$y_i = ((a_j \times x_i) + b_j + c_j) \bmod 256 \quad \text{..... (2)}$$

Where x_i ; is a plaintext character
i: presents the character position in user key
y_i ; is a ciphertext character
a_j: rotor pre-selected constant (should be reversible module 256)
j: presents wheel number of rotor
b_j: rotor pre-selected constant
c_j: cylinder rotation step
Decryption process using 8-bits inverse affine transformation as follows

$$x_i = ((y_i \times a_j^{-1}) - b_j - c_j) \bmod 256 \quad \text{..... (3)}$$

a_j^{-1}: multiplication inverse module 256 of preselected rotor constant
The main constrain here in sbox generation is the pre-selected constant a. It should be reversible module 2^8 ($a * a^{-1} = 1$), in order to make the affine transformation reversible.

2.4.2. Permutation

Permutation is re-arrangement of plaintext block contents which provide some sort of diffusion in the resulted ciphertext block.

This permutation rotor consists of 256 cylinders, each cylinders contains 64 elements, presenting all possible 6 bits values. The input plaintext is divided into 64 bytes basic blocks; each is permutated using a permutation rotor. After each permutation of the whole 64 bytes basic block, the permutation rotor is rotated, the last sub-block is one byte left rotated. Fig 4. Show the permutation process of cipher basic block.

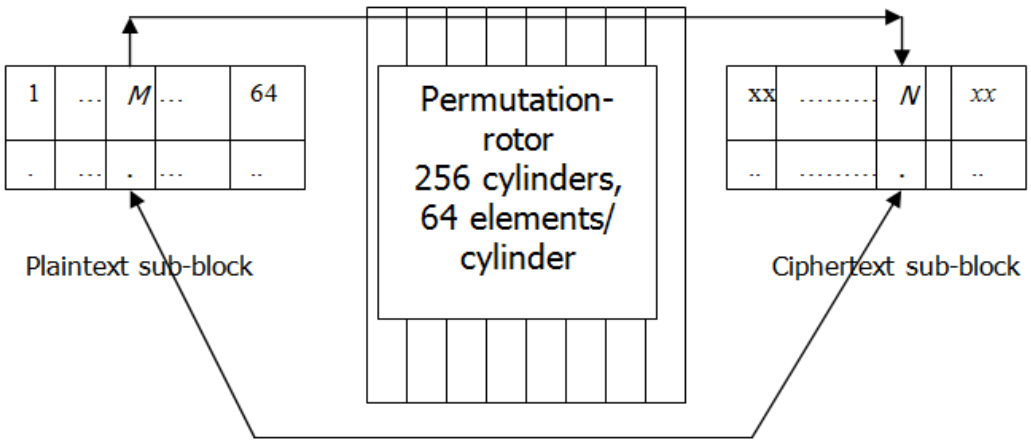


Figure 4. Permutation Rotor

From implementation point of view, substitution rotor is impossible (each cylinder contains 64 bytes, the whole rotor 64^{256}=2.41 x 10^{462} bytes), so a 6 bits affine transformation is used to dynamically generate sbox.

$$y_i = ((a_j \times x_i) + b_j + c_j) \bmod 64 \quad \text{.....(4)}$$

Where x_i ; is the old position of the basic block character
i: presents the character position in user key
y_i ; is the new position of the basic block character
a_j: rotor pre-selected constant (should be reversible module 64)
j: presents wheel number of rotor
b_j: rotor pre-selected constant
c_j: cylinder rotation step

2.4.3. Key Dependency

The key dependency is the last step in the encryption process. The modulo 2^{64} addition is used to add a plaintext basic block to a round key basic block if the basic plaintext basic block order is an even number. The 64 bits XORING is used if the plaintext basic block order is an odd number. Figure 5 shows the key addition process for even basic block. Figure 6 shows the key addition process for odd basic block.

The following equation, shows the key dependency of encryption process.

$$y_n = \begin{cases} ((a_n \times x_n) + k_n) \bmod 2^{64} \rightarrow n : even \\ x_n \otimes k_n \rightarrow n : odd \end{cases} \quad \text{.....(5)}$$

Where x_n ; is a plaintext basic block.
n: the basic block number
y_n ; is a ciphertext basic block
a_n: pre-selected constant (should be reversible module 2^{64})
kn: generated key

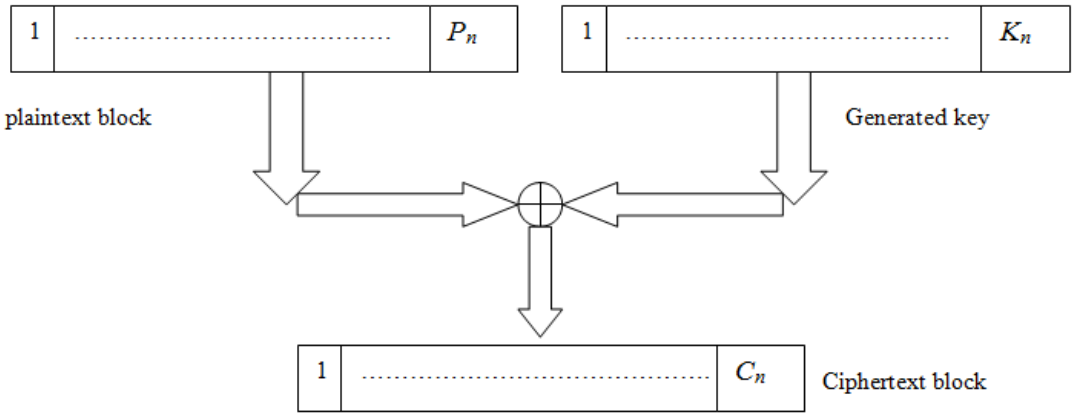


Figure 5. Key Addition for even basic blocks

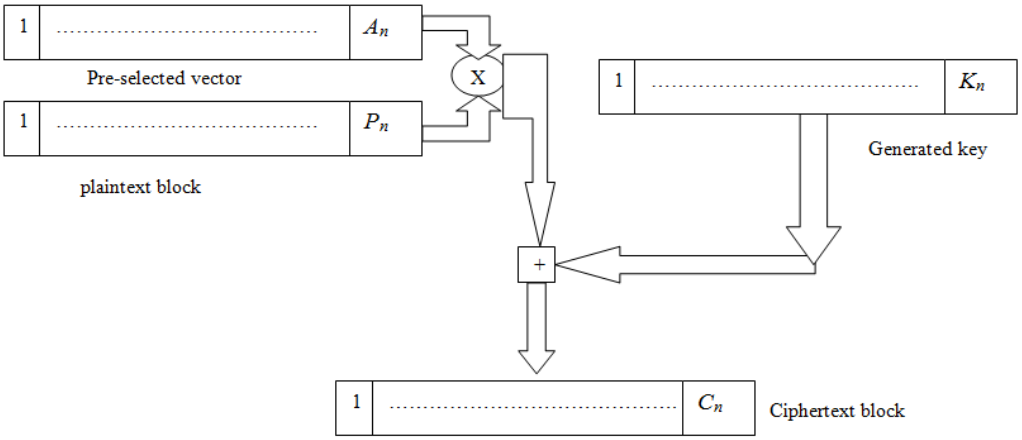


Figure 6. Key Addition for odd basic blocks

Decryption process using 2^{64} modulo inverse affine transformation or simple 64 bits-wise XOR as follows

$$x_n = \begin{cases} (y_n - k_n) \times a_n^{-1} \bmod 2^{64} \rightarrow n : even \\ y_n \otimes k_n \rightarrow n : odd \end{cases} \quad \text{..... (6)}$$

Where a^{-1}_n: 2^{64} module invers of the pre-selected constant a_n

3. REBC3 OVERALL STRUCTURE

3.1. REBC3 Single Round

REBC3 uses three sequent stages of encryption process, as described above. Which presents a single round encryption process. The structure of a single round of REBC3 is shown in the following script.

```
Expand_user_key();

Block_length=key_length; /* table (1)*/

number_of_rounds= length/4; */table (1)*/
round_number=0;
for(int n=0;n< number_of_rounds;n++)
/*Enc()*/
        {
        Substitution();
        Permutation ();
        Key_dependency();
        }
```

3.2. REBC3 rounds

The number of rounds of REBC3 depends on user key length as shown in table (1). REBC3 consists of successive repetition of a single round discussed in the previous section. As mentioned

before, there are three different versions of REBC3, shown in following table (2). The following Figures 7 to 9, show REBC3 versions structure.

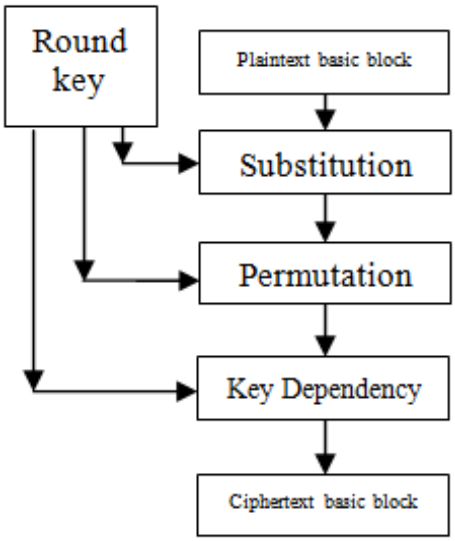


Figure 7. REBC3, 64 bits single round structure.

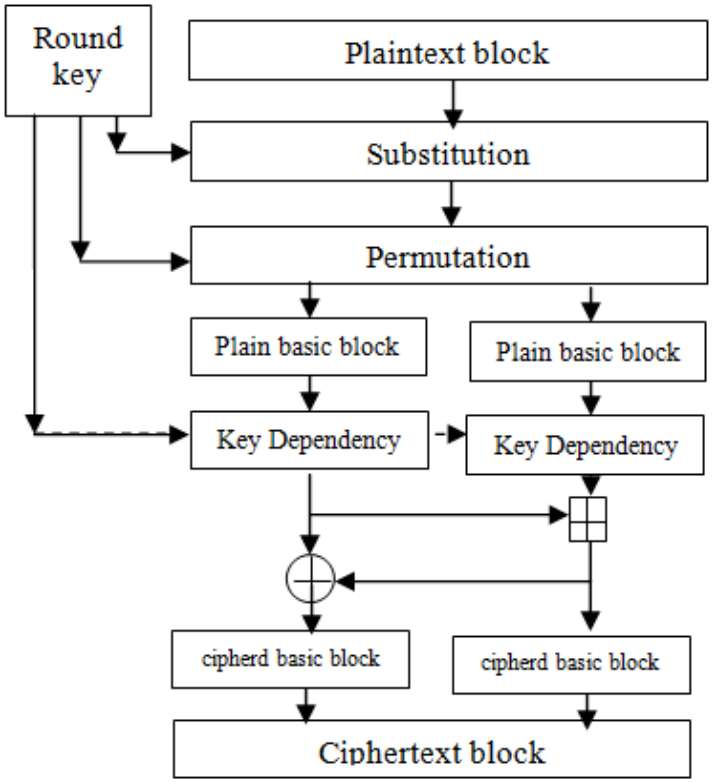


Figure 8. REBC3, 128 bits single round structure.

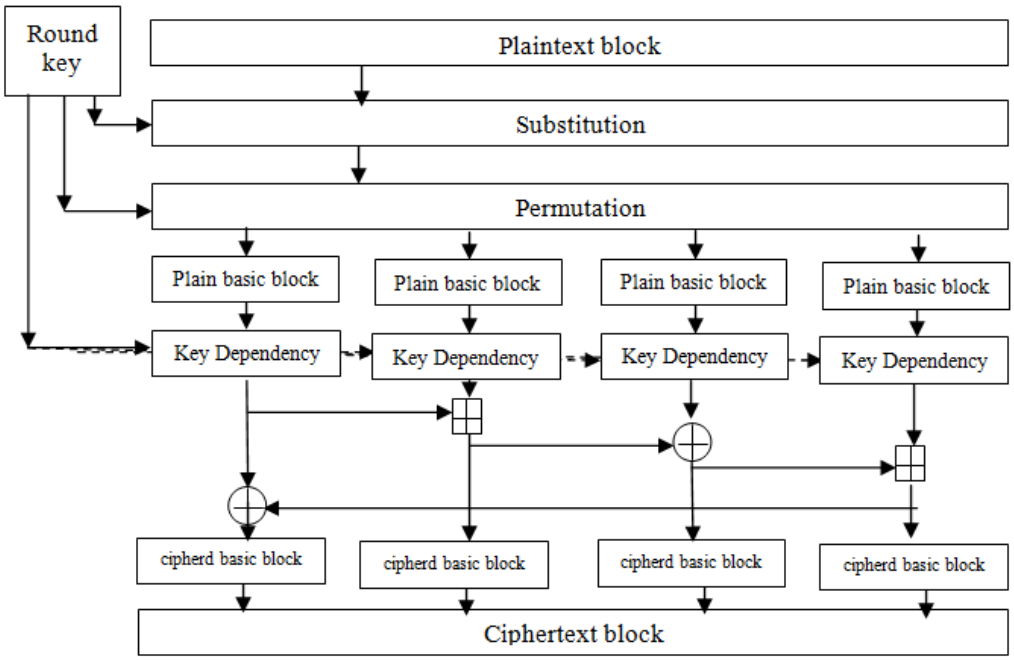


Figure 9. REBC3, 256 bits single round structure.

Table 2. Features of REBC3 versions

	REBC3, 64 bits	REBC3, 128 bits	REBC3, 256 bits
Key length (bytes)	8	16	32
Block length (bytes)	8	16	32
# basic blocks	1	2	4
# rounds	2	4	8

4. ANALYSIS OF REBC3

4.1. Secret data groups

This section discusses the secret data groups that was used in REBC3. Then compare between the amount of secret data used in RIJNDAEL (AES).

4.1.1. User key

REBC3 uses three different key lengths; 8 byte (64 bits), 16 bytes (128 bits), and 32 bytes (256 bits). Considering the last case, the total number of available keys equals to 256^{32}, which equals to the total number of trials to find the used key using brute force attack method in order to crack the cryptosystem. On the other hand RIJNDAEL uses three different key lengths, those are 128 bits, 192 bits and 256 bits. Considering the last case, the total number of available keys equals to 2^{256}, which equals to the total number of trials to find the used key using brute force attack method in order to crack the cryptosystem.

Round key is generated in each round using key rotor which is implemented using successive 8-bits affine transformation. The total number of available affine transformation is $(\phi\ (2^8) \times 2^8)$*, REBC3 pre-selects 8 different transformations from them. The static data required per affine transformation is 2 bytes, so the total amount of static data needed to implement key rotor is 16 bytes.

RIJNDAEL expanded key is a linear array of 4-bytes words that is defined recursively in terms of words with smaller indices. It uses three different functions, SubByte; which is already used by encryption algorithm itself, "RotByte"; which permute word bytes, and finally XORing with "Rcon" pre-selected secret data.

* Considering $\phi\ (2^n)$ is the number of integers that are less than 2^n and has an inverse modulo 2^n, which equals to $(2^n\text{-}2^{n-1})$, and equals to $(2^n/2)$.

4.1.2. Substation

REBC3 uses a substitution rotor, which is implemented using successive 8-bits affine transformations. REBC3 uses 256 different transformations form the total number of available transformation to implement that rotor. The total amount of static data to implement substitution rotor is 512 bytes.

4.1.3. Permutation

REBC3 uses permutation rotor, which is implemented using successive 6-bits affine transformations. The total number of available transformation in this case is $(\phi\ (2^6) \times 2^6)$. REBC-64 uses 256 different transformations from total number of available transformations. The total amount of static data to implement substitution rotor is 512 bytes

4.1.4 Key Dependency

Key dependency achieved using 64-bits affine transformations. The number of transformations depends on the cryptosystem key length. The 64 bits and 128 bits REBC3 use single affine transformation, and the 256 bits version uses two different affine transformations. Affine transformation uses two different variables as shown in equation 5, the "k" variable is a part of generated key, the "a" is a 64 bits pre-selected variable. The number of available variables is $\phi\ (2^{64}) = 2^{63}$. The amount of needed static data to implement single 64 bits affine transformation is 64 bits.

4.2. Brute Force attack

Considering secret data used in REBC3, the total number of trials to break ciphertext shown in the following table (3). RIJNDAEL brute force attack is shown in table (4).

Table 3. REBC3 brute force attack

	REBC3,64	**REBC3,128**	**REBC3,256**
Key	$2^{64} \approx 1.84$ E 19	$2^{128} \approx 3.8$ E 38	$2^{256} \approx 1.16$ E 77
Key Rotor (key generation)	$((2^7)$ x $(2^8))P_8 \approx 1.33$ E 31		
Substitution	$((2^7)$ x $(2^8))\ P_{256} \approx 3.25$ E 1155		
Permutation	$(2^5 \times 2^6)\ P_{256} \approx 1.81$ E 840		
Key dependency	$2^{63}\ P_1 \approx 9.22$ E 18	$2^{63}\ P_1 \approx 9.22$ E 18	$2^{63}\ P_2 \approx 8.51$ E 37
Total trials to attack REBC3	1.32 E 2065	2.74 E 2084	1.11 E 2142

Table 4. RIJNDAEL brute force attack

	RIJNDAEL-128	**RIJNDAEL-192**	**RIJNDAEL-265**
Key	$2^{128} \approx 3.8E38$	$2^{192} \approx 6.28E57$	$2^{256} \approx 1.16E77$
Key generation	$^{256}P_{32} \approx 4.97E76$		
Substitution	$(2^{8}P_1 \times 2^{7}P_1) \approx 3.2E4$		
Permutation	$(^{256}P_4) \approx 4.2E9$		
Total trials to attack RIJNDAEL	2.33E129	4.3E148	7.92E167

4.3. Static data requirements

The following tables (5), and (6) summarize the static data requirements for REBC3, and RIJNDAEL respectively.

Table 5. REBC3 static data requirement

	REBC3,64	**REBC3,128**	**REBC3,256**
Key	16		
Substitution	512		
Permutation	512		
Key dependency	1	2	2
Total required memory (bytes)	1041	1041	1042

Table 6. RIJNDAEL static data requirement

	RIJNDAEL-128	**RIJNDAEL-192**	**RIJNDAEL-265**
Key	32		
Substitution	64		
Permutation	4	8	16
Total req. mem. (bytes)	100	104	112

4.4. Period

The period of REBC3 is the product of two elements. These are rotor period, block length. The following table (7) shows REBC3 period. RIJNDAEL period depends on its block length only, as shown in table (8).

Table 7. REBC3 period in bytes

	REBC3,64	**REBC3,128**	**REBC3,256**
Block	8	16	32
Key rotor	$256^{8} \approx 1.84$ E 19		
Substitution	$256^{256} \approx 3.23$ E 616		
Permutation	$256^{256} \approx 3.23$ E 616		
Total period	1.54 E 1253	3.07 E 1263	6.14 E 1263

Table 8. RIJNDAEL period in bytes

	RIJNDAEL-128	**RIJNDAEL-192**	**RIJNDAEL-265**
Total period	16	24	32

4.5. Language Statistics

REBC3 is a rotor based block cipher that uses rotor to implement blocks confusions (substitution process) and diffusion (permutation process) which make it immune against brute force attack [17]. Cryptanalyst faces many difficulties to produce a general linear expression between input plaintext and output ciphertext. Such difficulty produced by different rotors used inside the cryptosystem, that add a huge period to the cryptosystem make it capable of ciphering tons of data before catching single period or reveal a repeatable pattern. The following Figure 10 to 15 show a comparison between ciphertext statistics of REBC3, and RINJDAEL produced from encrypting a plain English text file of 64K bytes. Another file contain repeatable single character of 64K bytes of size used as plaintext for both REBC3, and RIJNDAEL

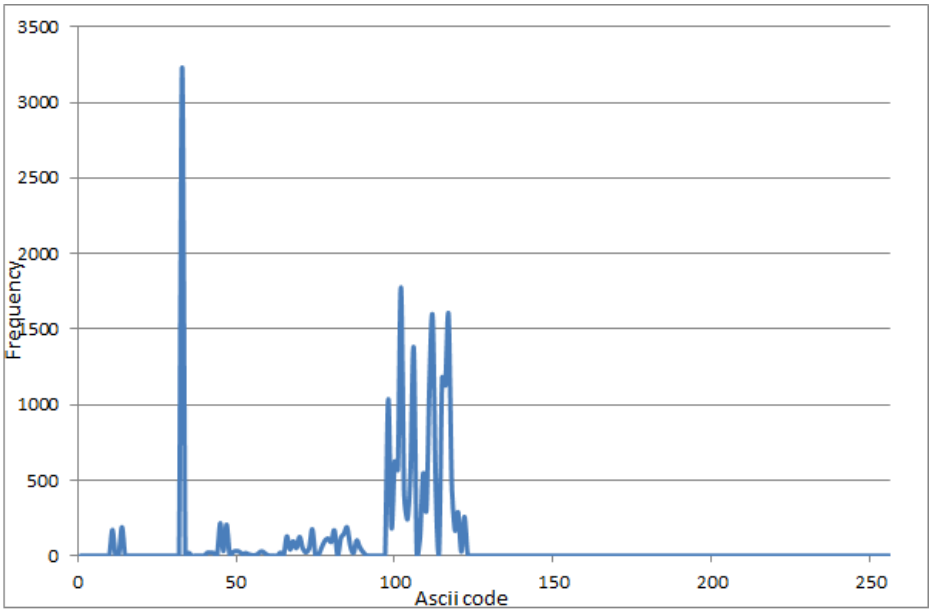


Figure 10. Plaintext statistics

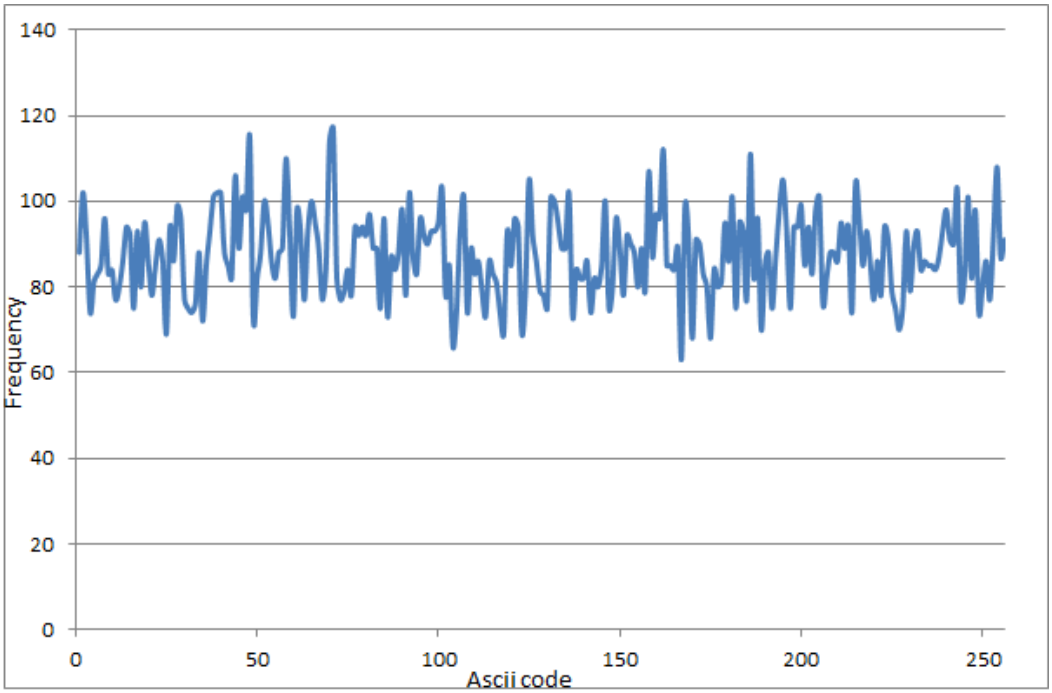


Figure 11. REBC3,265 ciphertext statistics

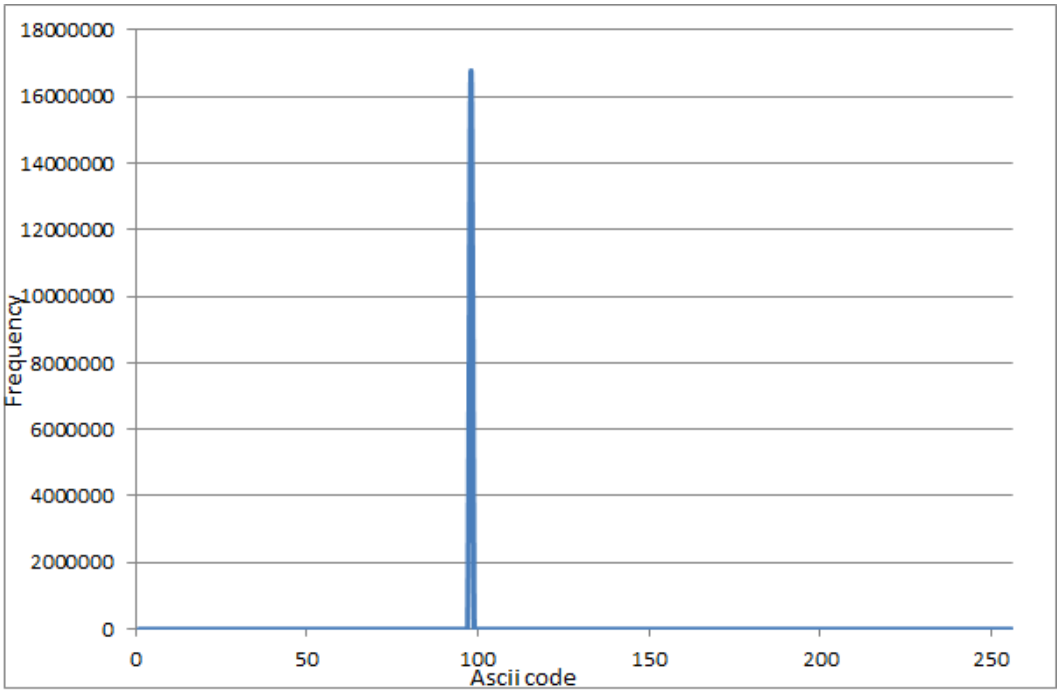


Figure 12. Delta plaintext statistics

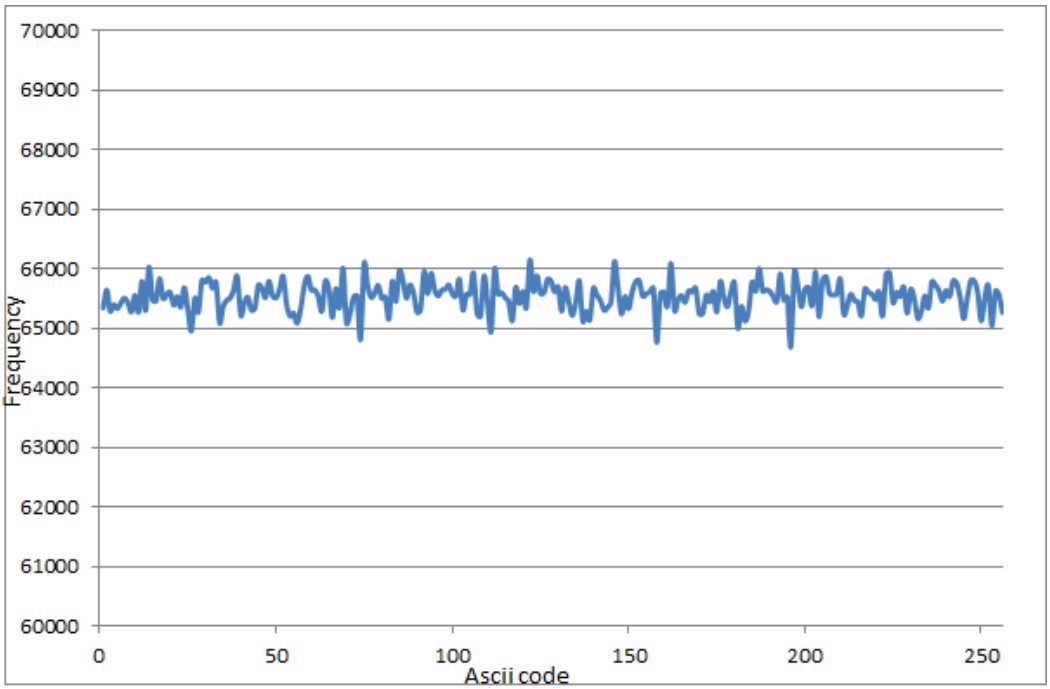


Figure 13. REBC3,256 ciphertext statistics (for delta file)

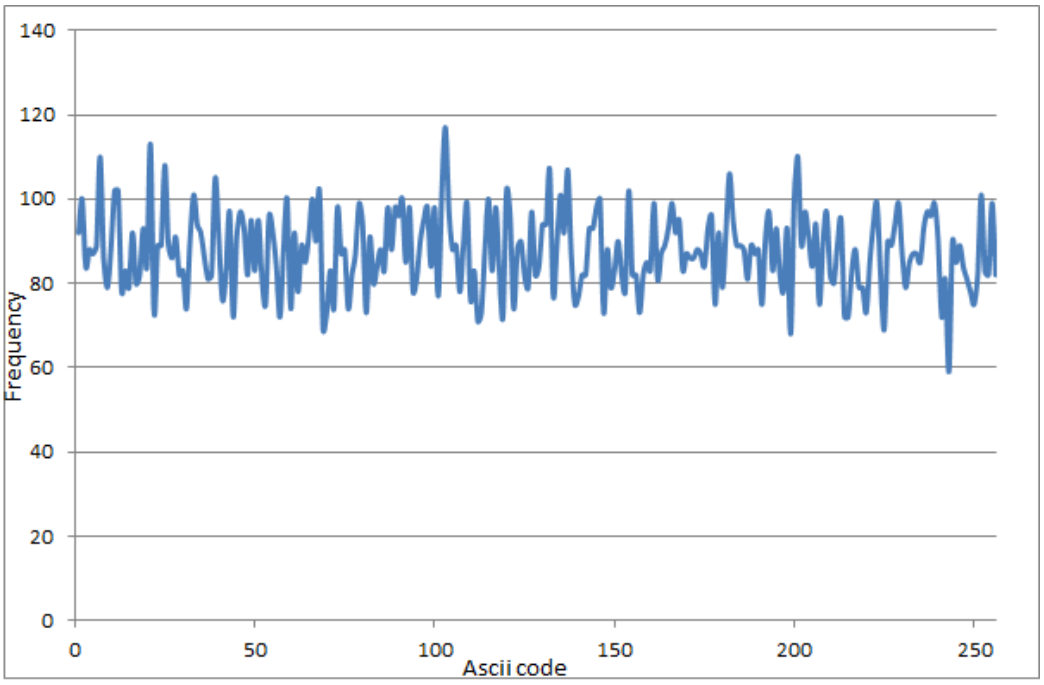


Figure 14. Rijndael,265 ciphertext statistics

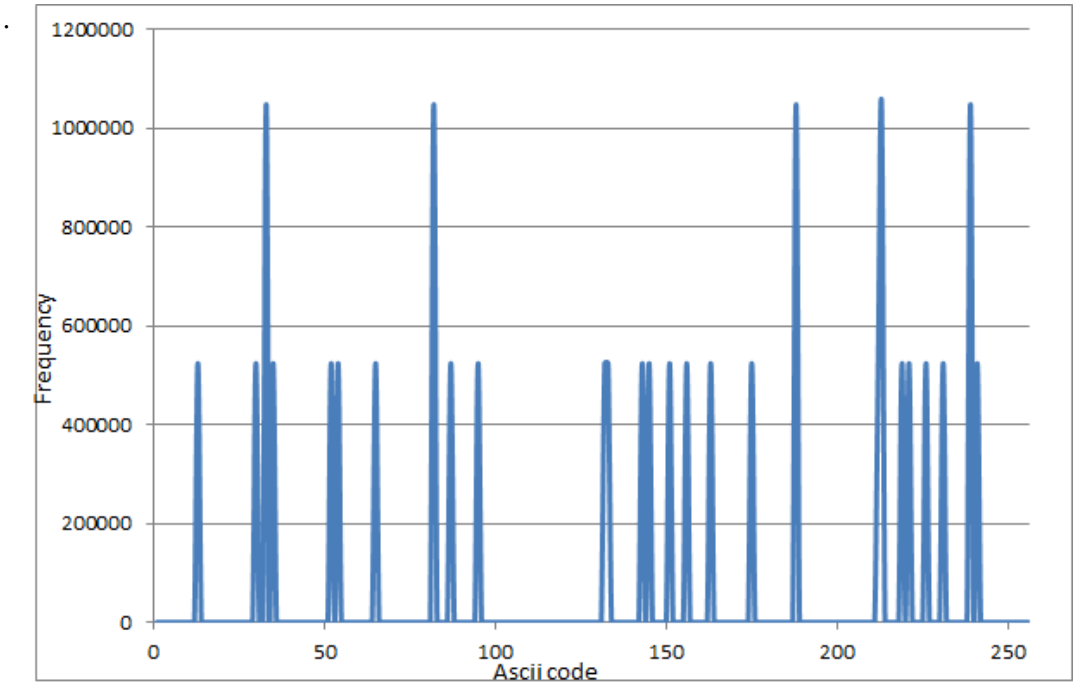


Figure 15. Rijndael,265 ciphertext statistics (for delta file)

5. CONCLUSIONS

The proposed REBC3 cryptosystem uses the conventional main cryptographic terms of substitution and permutation using rotors, and key dependency achieving the requested goals of like-perfect-statistics, large period, and resistance to linear and differential cryptanalysis. REBC3 can be considered as a high performance cryptosystem from speed point of view and security point of view as well, because it was developed as a 64 bits based cryptosystem, that take advantages of newly 64 bits microprocessors and operating systems available in technology market. Although REBC3 uses rotor (a large memory consuming ciphering technique), its static memory requirements is less than one kbytes because of its mathematical implementation that can perform like a classical rotor from ciphertext statistical point of view. 64 bits version of REBC3 is suitable for mobile devices applications. REBC3 has a huge period that makes it superior than the commonly known cryptosystems and it make its ciphertext statistics like one-time-pad cryptosystems that can make REBC3 suitable for encrypting huge messages without the need of operation modes.

REFERENCES

[1] Barry B. Brey, "*Intel Microprocessors: Architecture, Programming, and Interfacing*", Prentice Hall; 8th edition, 2008.

[2] Elkamchouchi, H.M.; Elshafee, A.M., *"Dynamically Key-controlled Symmetric Block Cipher KAMFEE*"; Radio Science Conference, 2003. NRSC 2003. Proceedings of the Twentieth National, 18-20 March 2003 Page(s):C19 - 1-12, Digital Object Identifier 10.1109/NRSC.2003.1217353

[3] A. ElShafee, *"A 64 bits Dynamically Key Controlled Symmetric Cipher (KAMFEE-X64)"*, International Journal of Computer Applications (0975 – 8887) Volume 57– No.13, November 2012, page16-24

[4] Bruce Schneier, "*Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*". Wiley Computer Publishing, John Wiley & Sons, Inc.

[5] Rivest: Ronald L. Rivest, "*The RC5 Encryption Algorithm"*, document made available by FTP and World Wide Web, 1994

[6] J. Daemen, J. T. Rijmen, "*AES Proposal: RIJNDAEL. AES Algorithm Submission*", 1999.

[7] Elkamchouchi, H.M.; Elshafee, A.M., "*REBC, Rotor Enhanced Block Cipher*"; Radio Science Conference, 2002. (NRSC 2002). Proceedings of the Nineteenth National Radio Science Conference, 19-21 March 2002 Page(s):262 - 269. Digital Object Identifier 10.1109/NRSC.2002.1022631

[8] Elkamchouchi, H.M.; Elshafee, A.M., "*Dynamically Key-controlled Symmetric Block Cipher KAMFEE*"; Radio Science Conference, 2003. NRSC 2003. Proceedings of the Twentieth National, 18-20 March 2003 Page(s):C19 - 1-12, Digital Object Identifier 10.1109/NRSC.2003.1217353

[9] ElKamchouchi, H.; ElShafee, A., "*Cyclone, the two Dimensional Rotor, Rotor's New Generation*"; Radio Science Conference, 2005. NRSC 2005. Proceedings of the Twenty-Second National, March 15-17, 2005 Page(s):269 – 276.

[10] ElKamchouchi, H.; ElShafee, A., "*RotRix, The Arrayed Rotors*"; Radio Science Conference, 2006. NRSC 2006. Proceedings of the Twenty-Third National Radio Science Conference.

[11] Elkamchouchi, H.M.; Elshafee, A.,"*REBC2 cipher*"; IEEE Africon 2007. Proceedings of the Africon 2007, September 26-28, 2007, Namebia, paper ID 624.

[12] ElKamchouchi, H.; ElShafee, A., "*New Rotor Based Symmetric Cipher*"; IEEE International Conference on Signal Processing and Communication. Proceedings of IEEE ICSPC, 24-27 November, 2007, Dubai, United Arab Emirates, paper ID 1569047958.

[13] ElKamchouchi, H.; ElShafee, A., "*URESC, Unbalanced Rotor Enhanced Symmetric Cipher*"; The 14th IEEE Mediterranean Electrotechnical Conference, Ajaccio, France, May 5-7, 2008, paper ID t1-sd1018.

[14] Elkamchouchi Hassan M., Ahmed Fatma; Elsoud A. Khairy Abo; Elkamchouchi Dalia H., "New Symmetric Cipher Enhanced by Rotor in Real & Gaussian Domains (SCER)", ICFN '10. Second International Conference on Future Networks, 2010.

[15] Elkamchouchi, Hassan M. Makar, Mina A., "Kamkar symmetric block cipher ", Radio Science Conference, 2004. NRSC 2004. Proceedings of the Twenty-First National

[16] Elkamachouchi H.M., Ahmed Fatma , "Rotor cipher with time controlled key and encryption process (RTCKP)", National Radio Science Conference, 2009. NRSC 2009.

[17] C.Shannon, "*communication Theory of Secrecy Systems*", Bell System Tech.. J., Vol. 28, 1949.

Authors

Ahmed M. ElShafee, Held a Bachelor degree in Electrical Engineering from Faculty of Engineering, Alexandria University, Masters' of science in Electrical Engineering from Faculty of Engineering, Alexandria University, Ph.D. degree in Electrical Engineering from Faculty of Engineering, Alexandria University. He published many scientific papers, international conferences in Egypt, France, Dubai, Namibia, and India.He won The Best Young Scientist Award as per the conference council recommendation (National Radio Science Conference 2001), Alexandria, Egypt, for his paper entitled "Rotor Enhanced Block Cipher (REBC)". He worked in telecommunication engineering field (Operations and Research & Development) for more than 8 years. Now he works as Assistant Professor, in Faculty of Computer Science and Information Technology (Ahram Canadian University), and as Researcher in Information Technology Research & Consultation Center (ITRCC), Ahram Canadian University