MICRO ROTOR ENHANCED BLOCK CIPHER DESIGNED FOR EIGHT BITS MICRO-CONTROLLERS (MREBC)

Ahmed ElShafee[1]
[1]Department of Computer Networks, Ahram Canadian University, 6th October City, Egypt

ABSTRACT

The sensor network is a wireless network environment that consists of the many sensors of lightweight and low-power. Authentication between nodes is very vital for network reliability and the integrity of information collected by these nodes. Therefore, encryption algorithm for the implementation of reliable sensor network environments is required to the applicable sensor network. This paper gives a new proposed cryptosystem (MREBC) that is designed for 8 bits microcontroller systems. MREBC uses the concept of rotor enhanced block cipher which was initially proposed by the author in [NRSC 2002] on the first version of REBC. MREBC uses 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. Rotors implemented using 8 bits successive affine transformation, which achieves memory-less, normalized ciphertext statistics, and small processing speed trend. The strength of this system is compared with the RIJNDAEL (AES) cipher. MREBC cipher gives excellent results from security characteristics and statistical point of view of. communication efficiency of MREBC is compared with AES through measuring performance by plaintext size, and cost of operation per hop according to the network scale. Arduino microcontroller board is used to implement both MREBC, and AES in order to compare the performance of algorithms. Authors suggests to use MREBC to implement a reliable sensor network environments.

KEYWORDS

8 Bits Microcontroller, block cipher, rotor cipher, brute force attack, Sensor Network.

1. INTRODUCTION

The sensor network is a wireless network environment that consists of the many sensors of lightweight and low-power [1]. The main constrains that faces sensor network are sensor resources, and communication channel reliability. All security approaches require a certain amount of resources for the implementation, including data memory, code space, and energy to power the sensor. However, currently these resources are very limited in a tiny wireless sensor. unreliable communication is another threat to sensor security. The security of the network relies heavily on a defined protocol, which in turn depends on communication. Sensor network security can

Sensor network shares some commonalities with a typical computer network, but also poses unique requirements and has many constraints compared to a traditional computer network. The main security requirements are; Data Confidentiality, Data Integrity, Data Freshness, Availability, Self-Organization, Time Synchronization, Secure Localization, and finally Authentication

Sensor networks faces several key types of attacks. Attacks can be performed in a variety of ways, most notably as denial of service attacks, but also through traffic analysis, privacy violation, physical attacks, and so on.

The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess. Achieving security will not only affects the operation of the network, but also is highly important in maintaining the availability of the whole network.

Public key encryption algorithm is a fundamental and widely used, but it has hardware requirements that exceeds sensor network capabilities, so it is not applied to the sensor network [2i]. Symmetric key encryption algorithms are simpler and has low-Energy consumption, so it can be used in the sensor networks.

This paper, gives MREBC, a new symmetric key cryptosystem that meets the security requirements for sensor network and take into consideration the hardware limitations of sensors. encryption and decryption performance is measured on the 8-bit Microcontroller. Then the communication efficiency through the total delay per hop in sensor network is analysed. All results are compared with AES (Rijndael) cryptosystem.

2. MREBC 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. Rotor is a well-known cryptosystem since the 2nd world ware [3i]. simply rotor is a cascaded substitution boxes that changes their structure after each encryption process. Rotor has a long period that makes the produced ciphertext has a very statistical properties. If the rotor period is larger than the ciphertext length the produced statistics would be like the one time pad cryptosystem statistics. The main drawback of rotor is its huge memory requirements and slow decryption speed.

Author has lately published cryptosystems like; REBC [9], KAMFEE [8], CYCLONE [7], ROTRIX [10], and REBC2 [11], URESC [12], [13] which all use rotor to enhance the block cipher properties. other block ciphers that use the same concept of rotor to enhance their ciphertext characteristics are SCER [14], KAMKAR [15], and RTCKP [16].

The proposed MREBC is an customized version of REBC2 [11] which was published by authors in 2002. MREBC designed to work on 8-bits embedded systems, having a low memory requirements, and can operate on low clock rate, with a very limited degradation of data transfer rate.

2.2. Key generation

MREBC has a 32 bytes (256 bits) key length. User is allowed to enter any key that is smaller than the standard key length. User key is expanded if needed to reach the standard key length using a small rotor called key-rotor which consists of 8 cylinders, each cylinder contains 256 elements. Key rotor is also used to generate the key of each round Figure. 1 shows first round key expansion and generation from user key. Considering K_r to present the encryption key of round r, and $K_{r\text{-}1}$ is to present the encryption key of round (r-1), Figure 2 shows the key generation process for round r.

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

$$y_i = ((a_j \times x_i) + b_j + c_j) \bmod 256 \qquad (1)$$

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

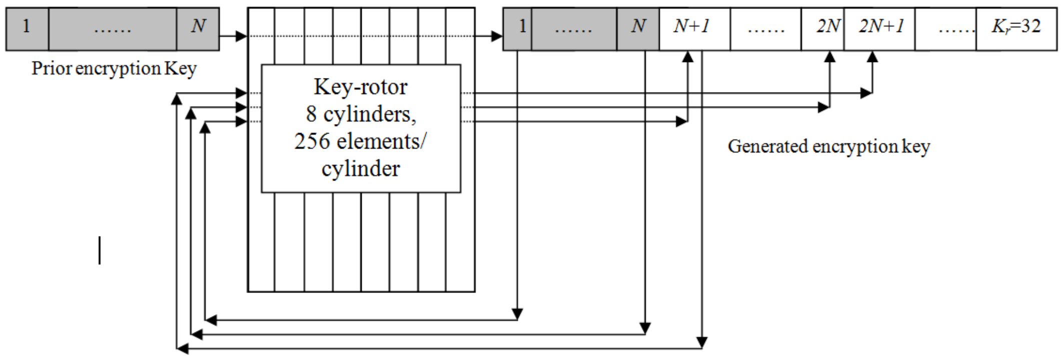

Figure 1. 1st round key generation from user key

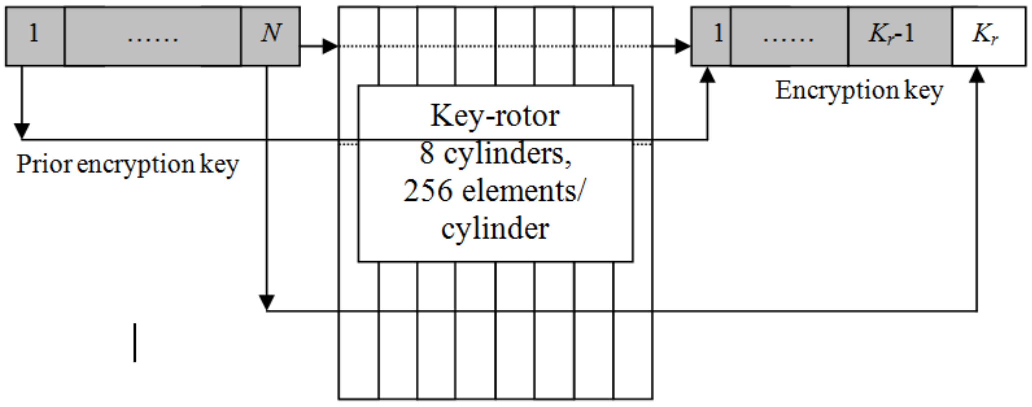

Figure 2. round key generation

2.3. Single round structure

MREBC round consists of three consequent steps, these are substitution, permutation, and key dependency.

2.3.1. Substitution

Substitution process is a nonlinear operation that enhance the security of the block cipher. Substitution is performed using an 18 cylinders rotor called substitution rotor. Each cylinder containing 256 characters presenting ASCII characters space. Figure 3 shows substitution rotor operation, as 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.

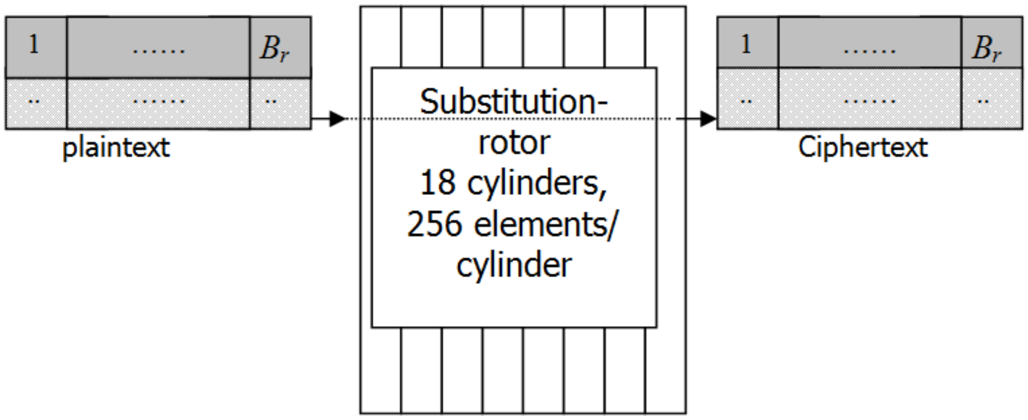

Figure 3. Substitution Rotor

To implement a traditional substitution rotor a huge amount of memory is needed (about 256^8=1.85 x 10^{19} bytes) (each cylinder contains 256 bytes). To implement the substitution rotor on 8 bits microcontroller successive affine transformations are used as shown in the following equation 2.

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

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.

2.3.2. Permutation

Permutation process is a re-arrangement of plaintext block contents which is also known as diffusion process.

This permutation rotor consists of 18 cylinders, each cylinders contains 32 elements, presenting all possible 5 bits values. The input plaintext basic block length is 32 bytes is permutated using a permutation rotor. After each permutation process the permutation rotor is rotated. Fig 4. Show the permutation process of cipher basic block.

To implement a traditional permutation rotor a huge amount of ROM is needed (about 32^8=1.1 x 10^{12} bytes) (each cylinder contains 32 bytes). To implement the substitution rotor on 8 bits microcontroller successive affine transformations are used as shown in the following equation 4.

$$y_i = ((a_j \times x_i) + b_j + c_j) \bmod 32 \quad \text{.......} (4)$$

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

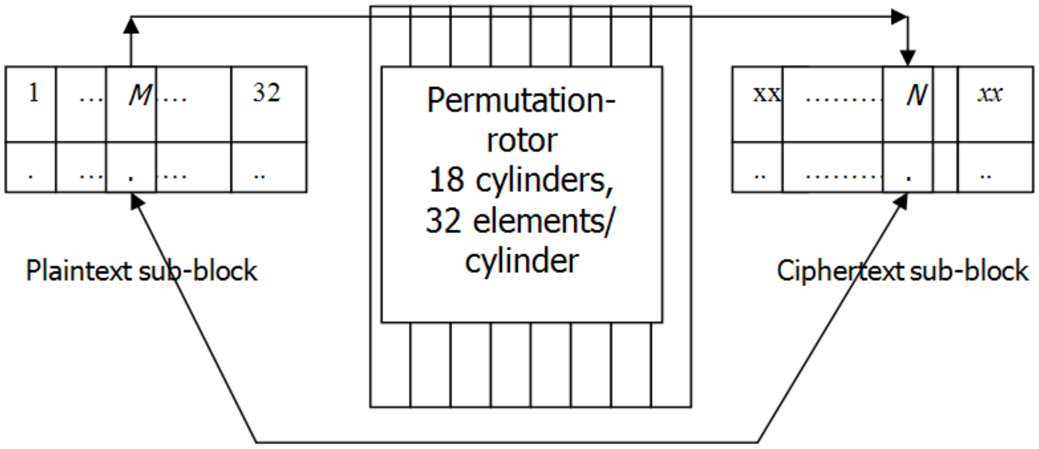

Figure 4. Permutation Rotor

2.3.3. Key Dependency

The key dependency is the last step in the encryption process. The modulo 2^8 addition is used to add a plaintext to a round key if the plaintext character order is an even number. The 8 bits XORING is used if the plaintext character order is an odd number. Figure 5 shows the key addition process for even characters. Figure 6 shows the key addition process for odd characters. The following equation shows the key dependency of encryption process.

$$y_n = \begin{cases} (x_n + k_n) \bmod 2^8 \rightarrow n : even \\ x_n \otimes k_n \rightarrow n : odd \end{cases} \quad \text{.......} (5)$$

Where x_n ; is a plaintext basic block.
n: the basic block number
y_n ; is a ciphertext basic block
k_n: generated key

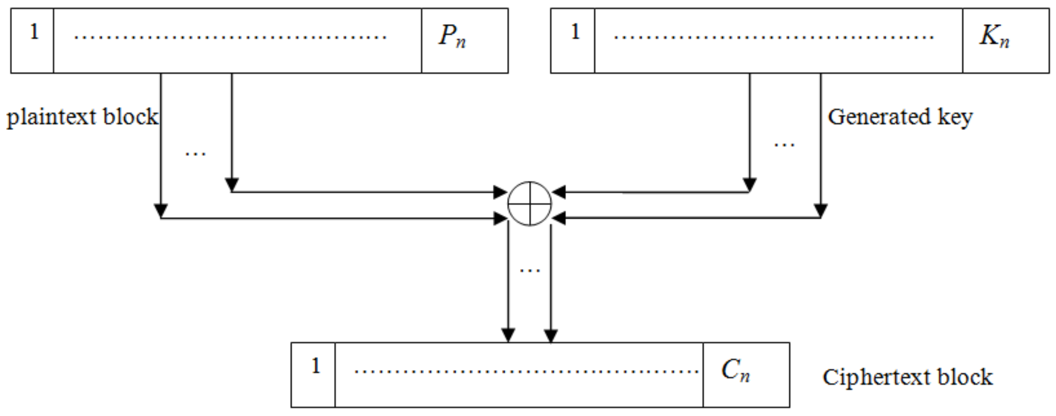

Figure 5. Key Addition for even ordered character

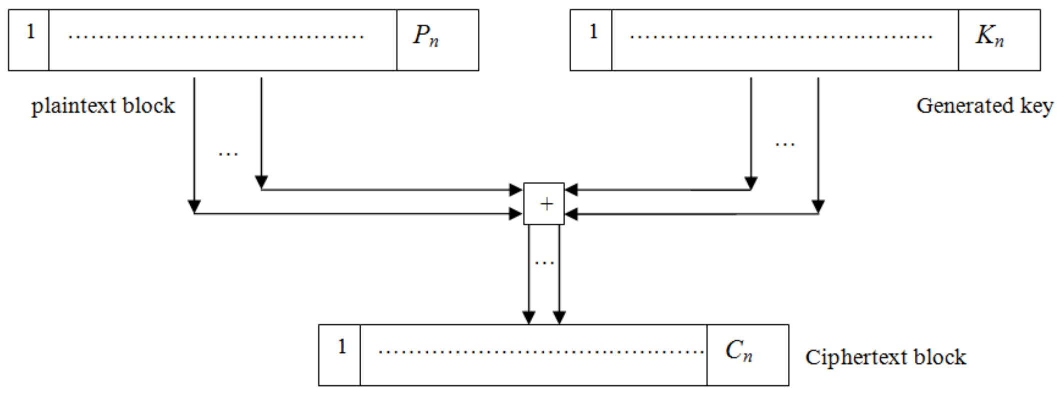

Figure 6. Key Addition for odd ordered characters

Decryption process using 2^8 modulo inverse affine transformation or simple 8 bits-wise XOR as follows

$$x_n = \{ \begin{matrix} (y_n - k_n) \bmod 2^8 \rightarrow n : even \\ y_n \otimes k_n \rightarrow n : odd \end{matrix} \quad \text{.......} (6)$$

3. MREBC OVERALL STRUCTURE

3.1. MREBC Single Round

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

```
Expand_user_key();
round_number=2;
for(int n=0;n< number_of_rounds;n++)
/*Enc()*/
        {
```

```
Substitution();
Permutation ();
Key_dependency();
}
```

3.2. MREBC rounds

MREBC consists of two successive rounds. The following Figure 7 shows single round structure.

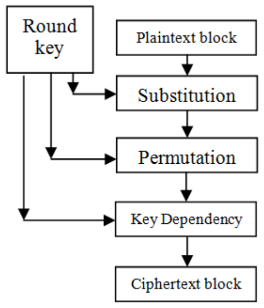

Figure 7. MREBC single round structure.

4. SYSTEM IMPLEMENTATION

4.1. Experiment and device

For the performance analysis of MREBC and AES encryption algorithms in the sensor network, 8-bit Microcontroller board called Arduino Uno was used [18]. Figure 8 show the Ardunio Uno board with Ethernet shield. Arduino Uno board uses ATmega328 microcontroller by Atmel. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, and power connector. It contains everything needed to support the microcontroller; it can simply programmed through USB connection. The Arduino Ethernet Shield allows an Arduino board to connect local area network and internet. It provides a network (IP) stack capable of both TCP and UDP. There is an onboard micro-SD card slot, which can be used to store files for serving over the network.

Figure 8. Arduino Uno with Ethernet Shield

4.2. The implementation of principle

Plaintext needed to be encrypted is sent to microcontroller as UDP packets. The data field of UDP packet consists to 3 parts as shown in the following table 1.

Table 1. UDP packet data field construction

Data field			**Command description**
Command (3bytes)	Packet identifier (5 bytes) 00 00 00 00 00 00 : FF FF FF FF FF FF	Data (32 bytes)	
key	5 bytes unique identifier	Key up to 32 bytes	Key exchange
enc		Plaintext data	Plaintext data to be encrypted
dec		Ciphertext data	Ciphertext data to be decrypted
AcK		-	Acknowledge of receiving ciphering key
AcE		Ciphertext data	Ciphetxt produced from last encryption process
AcD		Plaintext data	Plaintext produced form last decryption process

4.3 Implementation results

For the comparison between MREBC and RIJDAEL performance both algorithms were written in C code and burnt into Arduino Uno. The operation time of the encryption and decryption is measured to the data sizes of 32, 64, 128, 256 and 512 Byte. Table 2, and table 3 shows the encryption and decryption operation time and CPU cycle according to the data size for MREBC and RIJDAEL correspondently. Figures 9, 10 show MREBC encryption and decryption operation time and CPU cycle. Figures 11, 12 show RIJNDAEL encryption and decryption operation time and CPU cycle.

Table 2. Comparison between MREBC encryption and decryption time by data sizes

Data size (bytes)		32	64	128	256	512
End	Time (ms)	962	1914	3733	7130	13333
	CPU cycles	19240	38288	74661	142602	266666
Dec	Time (ms)	832	1656	3229	6167	11532
	CPU cycles	16640	33114	64572	123332	230630

Table 3. Comparison between RIJNDAEL encryption and decryption time by data sizes

Data size (bytes)		32	64	128	256	512
End	Time (ms)	898	1796	3592	7184	14368
	CPU cycles	17960	35920	71840	143680	287360
Dec	Time (ms)	912	1825	3649	7297	14592
	CPU cycles	18240	36500	72980	145940	291840

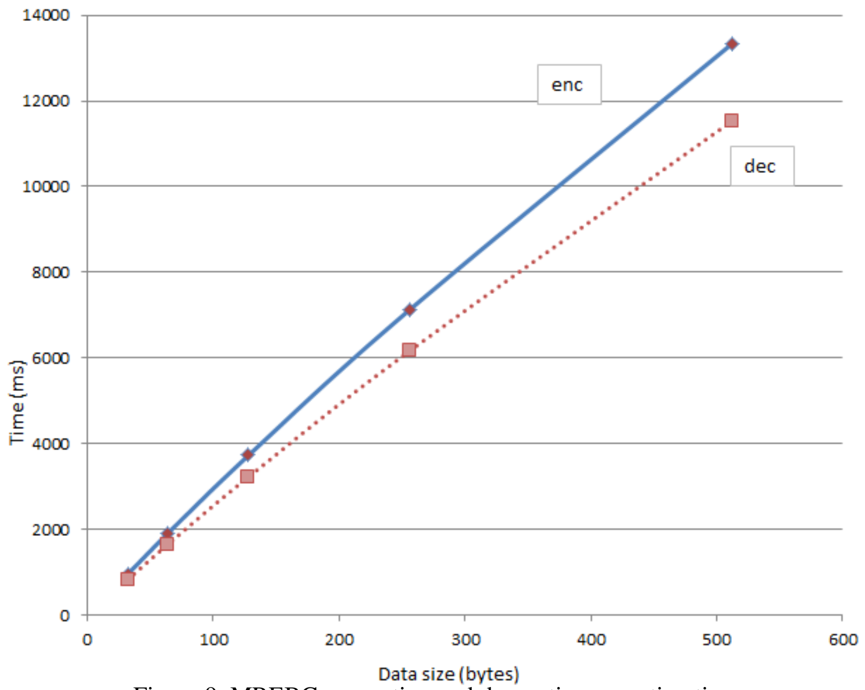

Figure 9. MREBC encryption and decryption operation time

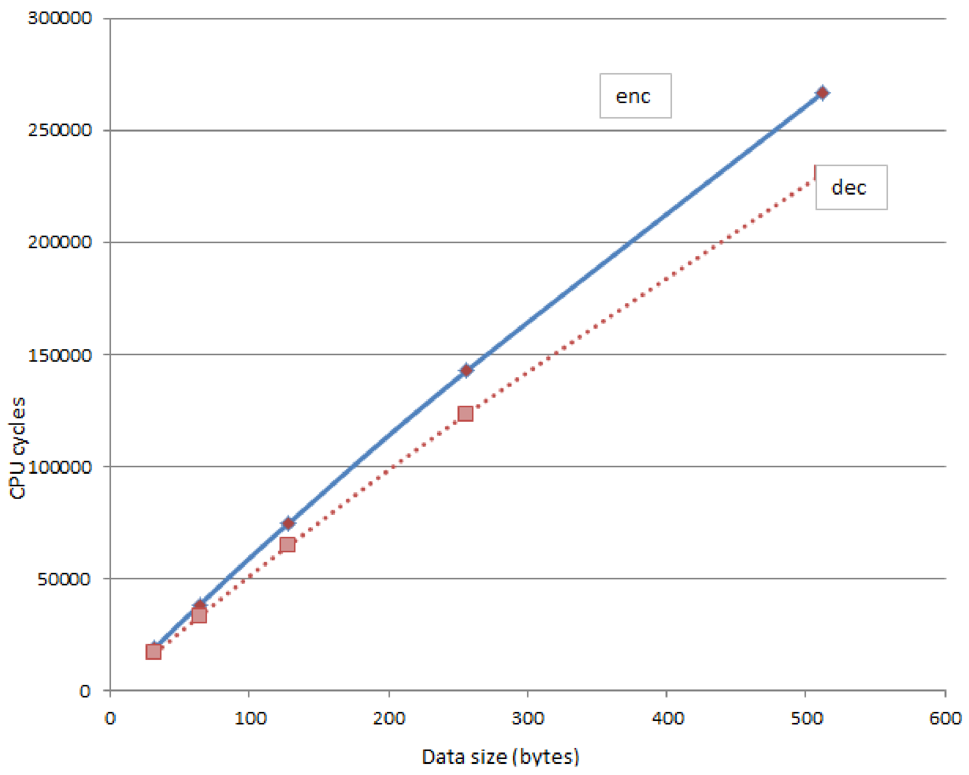

Figure 10. MREBC encryption and decryption operation CPU cycle

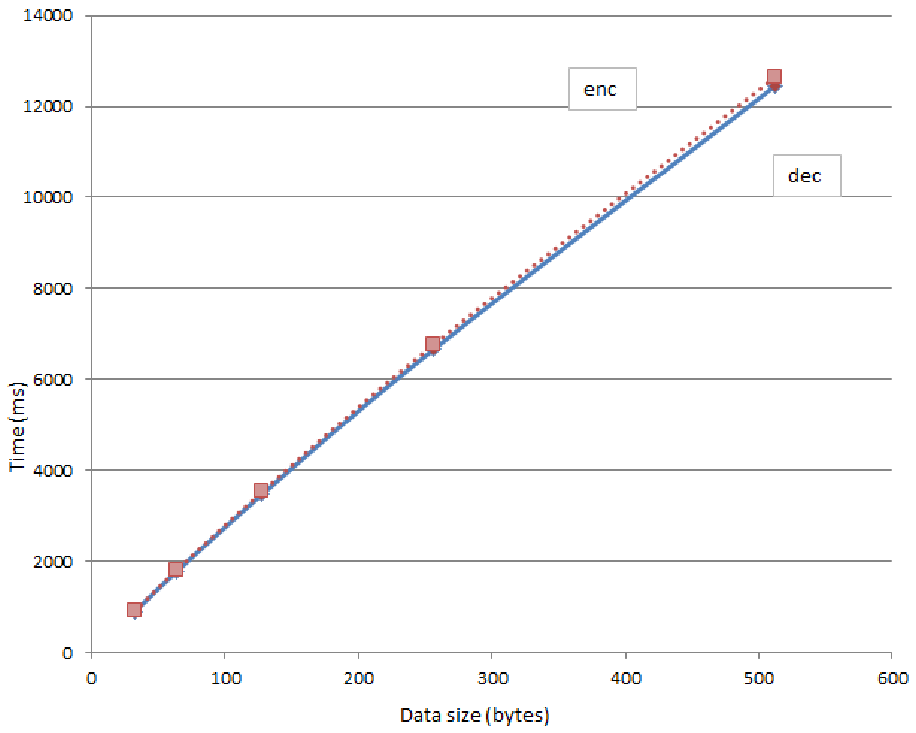

Figure 11. RIJNDAEL encryption and decryption operation time

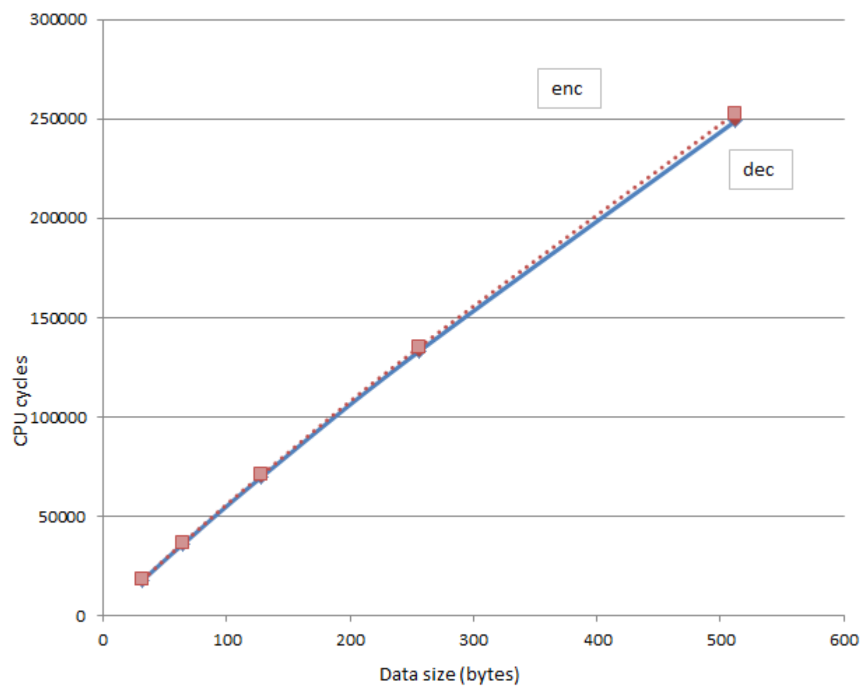

Figure 12. RIJNDAEL encryption and decryption operation CPU cycle

5. Analysis Of MREBC

5.1. Secret data groups

This section discusses the secret data groups that is used in MREBC. Then compares between the amount of secret data used in RIJNDAEL (AES).

5.1.1. User key

MREBC uses 32 bytes (256 bits) key, the total number of available keys equals to 2^{256}, which equals to the total number of trials crack the cryptosystem using brute force attack. RIJNDAEL uses three different key lengths; 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 crack the system using brute force attack.

MREBC rounds' key is generated using key rotor which is implemented using successive 8-bits affine transformation. The total number of available affine transformation is $(\phi\ (2^8) \times 2^8)^*$, MREBC pre-selects only 8 different transformations. The amount of memory required per affine transformation is 2 bytes, so the total amount of static data needed to implement key rotor is 16 bytes only.

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 ϕ (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-2^{n-1}), and equals to ($2^n/2$).

5.1.2. Substation

MREBC uses a substitution rotor, which is implemented using successive 8-bits affine transformations. MREBC uses 18 different transformations form the total number of available transformation to implement that rotor. The total amount of memory required to implement substitution rotor is 36 bytes.

5.1.3. Permutation

MREBC uses permutation rotor, which is implemented using successive 5-bits affine transformations. The total number of available transformation in this case is (ϕ (2^5) $\times$ 2^5). MREBC uses 18 different transformations from total number of available transformations. The total amount of memory required to implement substitution rotor is 36 bytes

5.2. Brute Force attack

Considering secret data used in MREBC, the total number of trials to break ciphertext shown in the following table (4). RIJNDAEL brute force attack is shown in table (5).

Table 4. MREBC brute force attack

	MREBC
Key	2^{32} $\approx$ 4.29 E 9
Key Rotor (key generation)	((2^7) x (2^8))P_8 $\approx$ 1.33 E 36
Substitution	((2^7) x (2^8)) P_{18} $\approx$ 1.89 E 81
Permutation	($2^5 \times 2^4$) P_{18} $\approx$ 4.52 E 48
Total trials to attack MREBC	4.87 E 175

Table 5. 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	(2^8P_1 x 2^7P_1) $\approx$ 3.2E4		
Permutation	($^{256}P_4$) $\approx$ 4.2E9		
Total trials to attack RIJNDAEL	2.33E129	4.3E148	7.92E167

5.3. Read only memory requirements

The following tables (6), and (7) summarize the static data requirements for MREBC, and RIJNDAEL respectively.

Table 6. MREBC memory requirement

Item	**MREBC**
Key	16
Substitution	36
Permutation	36
Total required memory (bytes)	88

Table 7. 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

5.4. Period

The period of MREBC is the product of two elements. These are rotor period, block length. The following table (8) shows MREBC period. RIJNDAEL period depends on its block length only, as shown in table (9).

Table 8. MREBC period in bytes

item	**MREBC**
Block	32
Key rotor	$256^8 \approx 1.84$ E 19
Substitution	$256^{18} \approx 2.23$ E 43
Permutation	$32^{18} \approx 1.24$ E 27
Total period	1.63 E 91

Table 9. RIJNDAEL period in bytes

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

5.5. Language Statistics

MREBC is a rotor based block cipher that uses rotor to implement blocks confusions (substitution process) and diffusion (permutation process). Using encryption rotor mechanism gives MREBC a huge period that makes it immune against brute force attack [19]. Another advantage of using encryption rotor is nonlinearity, which means cryptanalysis faces difficulties to produce a linear expression between input plaintext and output ciphertext. Finally the huge period of MREBC makes it capable of ciphering tons of data before catching single period or reveal a repeatable pattern. Figure (13) shows an English plaintext statistics. Figure (14) shows MREBC ciphertext statistics. Figure (15) shows RIJNDAEL ciphertext statistics. Figure (16) shows statistics of an English plaintext contain a repeated single character (delta plaintext files). Figure (17) shows MREBC ciphertext statistics produced of encrypting the delta plaintext file. Figure (18) shows RIJNDAEL ciphertext statistics produced of encrypting the delta plaintext file.

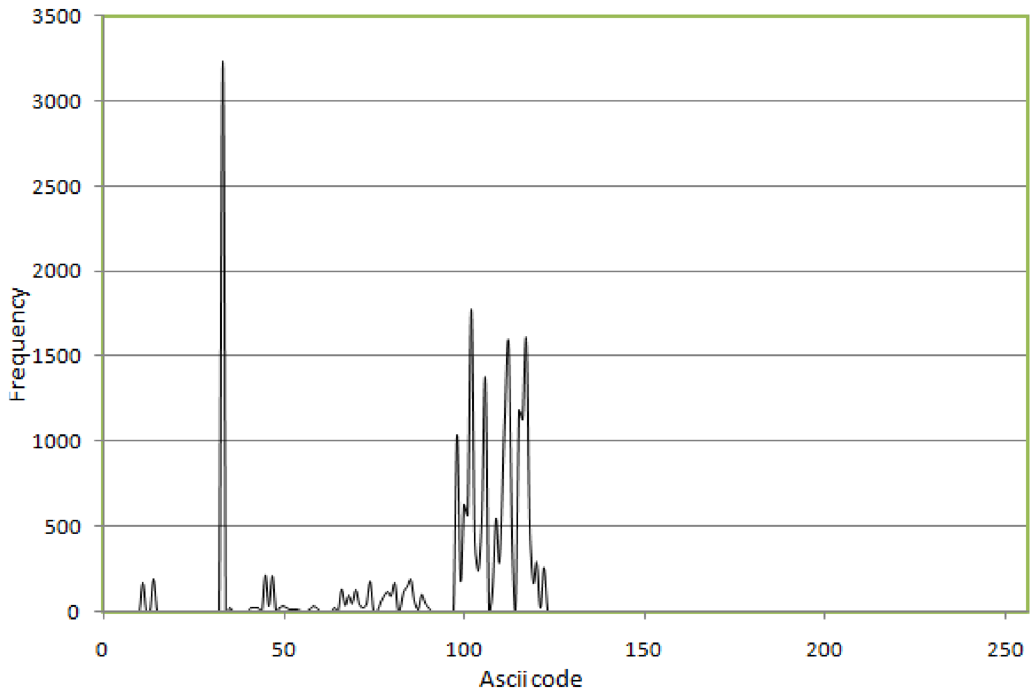

Figure 13. Plaintext statistics

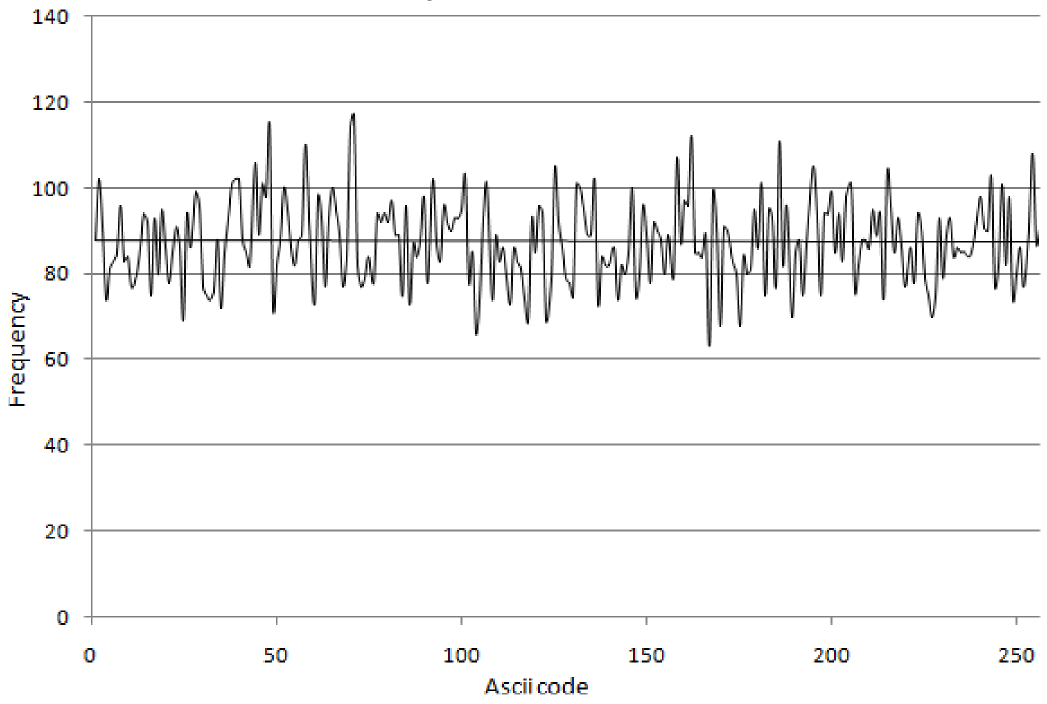

Figure 14. MREBC ciphertext statistics

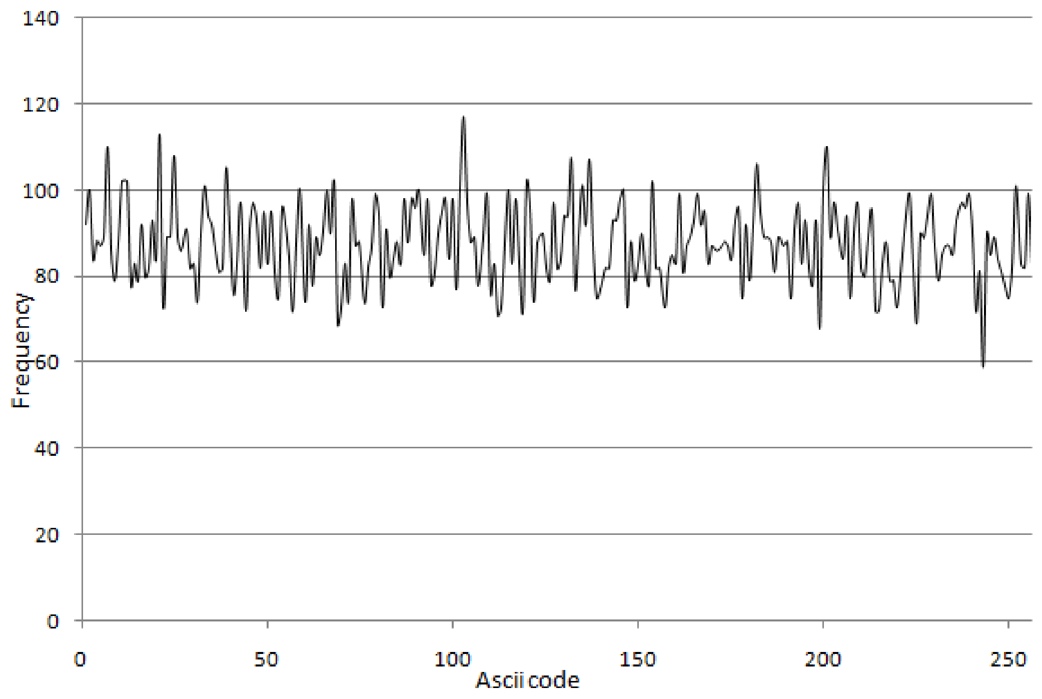

Figure 15. Rijndael,265 ciphertext statistics

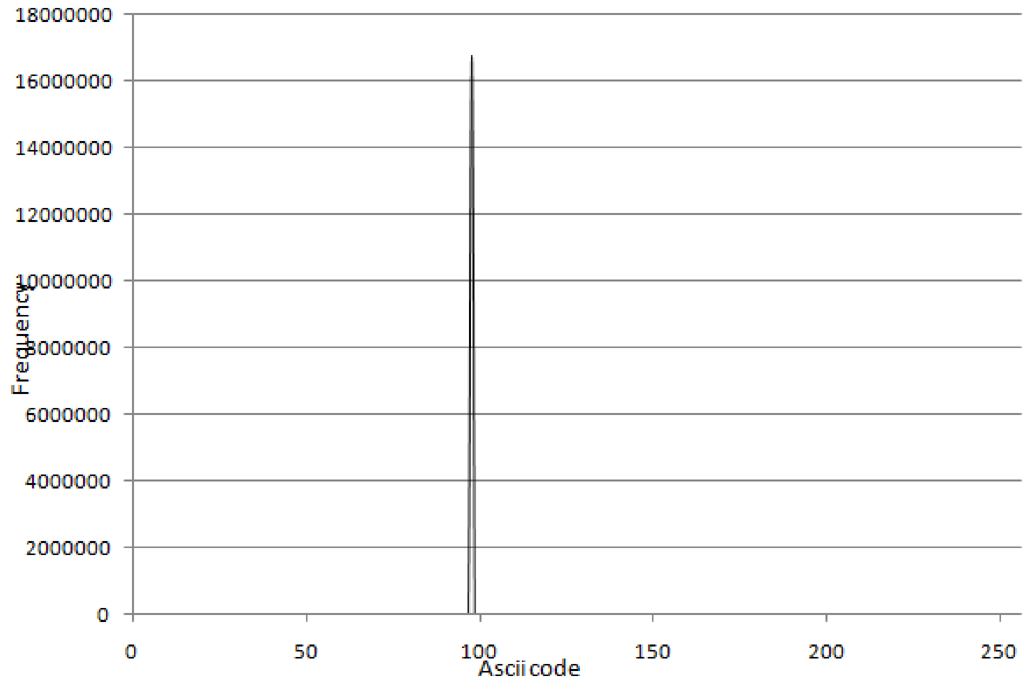

Figure 16. Delta plaintext statistics

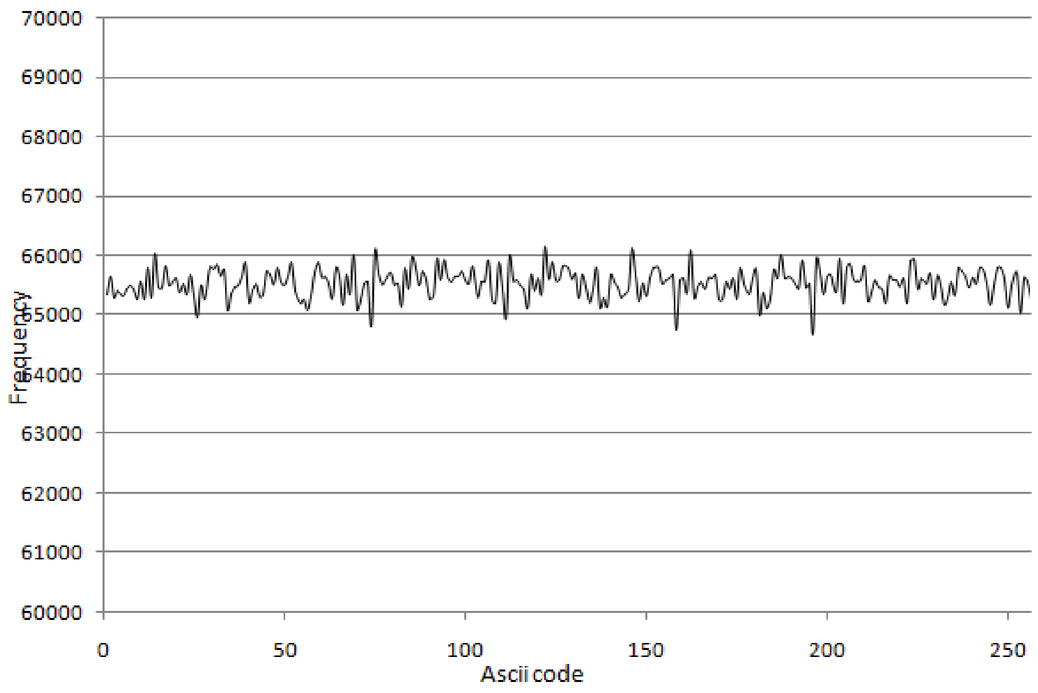

Figure 17. MREBC ciphertext statistics (for delta file)

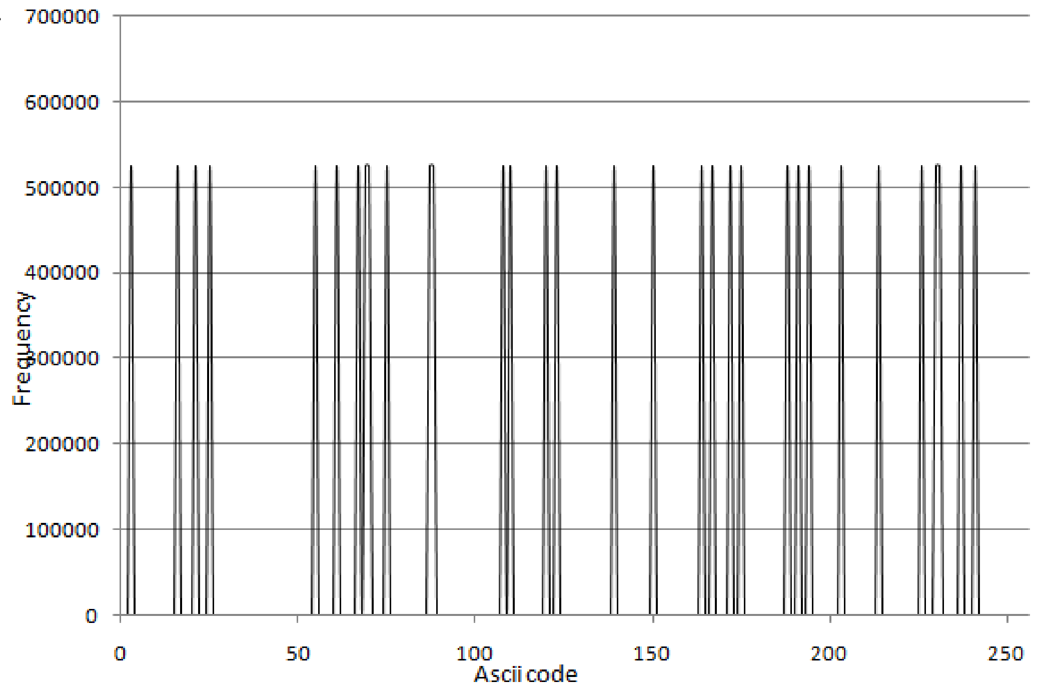

Figure 18. Rijndael,265 ciphertext statistics (for delta file)

6. CONCLUSIONS

The proposed MREBC cryptosystem is a block cipher uses ciphering rotor mechanism to enhance its security characteristics. MREBC uses main conventional cryptographic terms like confusion and diffusion -which are implemented using rotors-, and finally key dependency to achieve the requested goals of like-perfect-statistics, large period, and resistance to linear and differential cryptanalysis. MREBC take into consideration the hardware limitations of sensor networks, it can be implemented on 8 bits microcontroller embedded systems giving acceptable performance from speed point of view and excellent performance from security point of view as well. Although MREBC uses rotor (a large memory consuming ciphering technique), its read only memory requirements is a few bytes because of its mathematical implementation that can perform like a classical rotor from ciphertext statistical point of view. MREBC 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 MREBC suitable for encrypting huge messages without the need of operation modes.

REFERENCES

[1] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary, "Security in Distributed, Grid, and Pervasive Computing", Auerbach Publications, CRC Press, 2006.

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

[3] 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

[4] 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

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

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

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

[8] 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

[9] 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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

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

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

[18] "Arduino microcontroller", [online:] http://Arduino.cc

[19] 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 p aper 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