International Journal of Computer Applications (0975 – 8887)
Volume 57– No.13, November 2012

A 64 bits Dynamically Key Controlled Symmetric Cipher (KAMFEE-X64)

Ahmed El Shafee
Asst. Prof. Ahram Canadian University
4th Industrial Zone, 6th October City
Giza, 12451, Egypt

ABSTRACT

This paper presents an improved version of KAMFEE cipher [1]. The proposed cipher (KAMFEE-X64) is designed to be compatible with the new 64 bits microprocessors unlike the old cipher (KAMFEE) which is designed for systems based on x86 microprocessors. KAMFEE-X64 has a key dependent block length and key dependent rounds, enhanced by a rotor. The number of rotor's wheels depends on key length too, rotor is implemented using successive 64 bits affine transformations. Its block is divided into basic blocks of 64 bits length. 264 modulo addition and 64 bits XORING are used. It uses 64 bits sboxes implemented using 64 bits affine transformation. Two steps of permutation are used, first step is basic block permutation and second step is basic block mixing. The strength of this system is compared with the KAMFEE and the well-known RC6, and RIJNDAEL ciphers. KAMFEE-X64 cipher gives excellent results from security characteristics and statistical point of view of. So authors suggests to use KAMFEE-X64 in the area of banking and electronic fund transfer.

Keywords

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

1. INTRODUCTION

Most [1] traditional communications media including telephone, music, film, and television are reshaped or redefined by the Internet, giving birth to new services such as Voice over Internet Protocol (VoIP) and Internet Protocol Television (IPTV). Newspaper, book and other printed publications are adapting to Web site technology. The Internet has enabled and accelerated new forms of human interactions through instant messaging, Internet forums, and social networking. Online shopping has boomed both for major retail outlets and small artisans and traders. Business-to-business and financial services on the Internet affect supply chains across entire industries. So computer network security and information security issues are on rise, as almost all internet users are getting concerned about security of the information they deal with in any of these communication media on the internet. Cryptology is the science that deals with information security. It comprises both of 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. Thus, cryptology is an active science that is in continuous study and big challenges. It always seeks to overwhelm the increasing computer speed/architecture. There are two types of ciphers; symmetric and asymmetric. In symmetric ciphers, the encryption process consists of an algorithm and a single key. For given plaintext, the encryption algorithm gives its output depending on the key used. The ciphertext is transmitted through the communication channel towards destination. At the receiver the decryption algorithm uses the same key to convert ciphertext into plaintext again. Encryption is expressed by the mathematical expression:

$$Y = E_K(X) \quad \text{.......} (1)$$

Then decryption is expressed by a similar mathematical expression as follows:

$$X = D_K(Y) = D_K(E_K(X)) = E_K(D_K(X)) \quad \text{........} (2)$$

A cryptanalyst tries to attack the ciphertext by generating the plaintext X` by an estimated key k`[2]. In the early days of cryptography there are two general methods. Those are substitution and transposition. In substitution each element in the plaintext (bits, bytes,..) is mapped into another element, and in transposition elements of plaintext are rearranged. Nowadays cryptographic systems involve multiple stages of substitutions and transpositions (permutations).

There are two basic types of symmetric ciphers; block ciphers and stream ciphers. Block ciphers operate on blocks of plaintext and ciphertext. The input of the system is one block each time and output is one block. The block may usually be 32 bits, 64 bits or longer. The block length is constant for all the processes in a certain cipher. In stream ciphers the input and hence the output of the system are continuous i.e. character by character or bit by bit. In block ciphers, the same plaintext block will always encrypt to the same ciphertext block, using the same key. In stream ciphers the same plaintext bit or byte will encrypt to a different bit or byte every time it is encrypted.

In asymmetric ciphers, keys come in pairs an encryption key and a decryption key. Public key cryptosystems is an example of the asymmetric cryptosystem. Two different keys are used one is public and the other is private. It is computationally hard to obtain the private key from the public key. Anyone with the public key can encrypt a message. Only the person with the private key can decrypt the message.

2. KAMFEE-X64 OPERATIONAL STRUCTURE

2.1 Overview

In order to develop a new cryptosystem, there are three guide lines, which must be taken into consideration while building block ciphers. These are substitution, permutation which were defined by the cryptography godfather Shannon [1] in the middle of the twentieth century, then key dependency to achieve privacy of cryptosystem per user. All successfully known cryptosystems, follows the same rules with little bits of variations depending on designer point of view. For example Rivest [3] used rotation, with their rotating order depending on the encrypted data itself. RIJNDAEL [4] used simple mathematical operations to achieve ordinary permutation and substitution. In other lately published cryptosystems REBC [5], KAMFEE [6], CYCLONE [7], ROTRIX [8], and REBC2 [9] [10], URESC [11], rotors are always used. In each of these cryptosystem rotors are being used with unique concept, and different methodology.

The proposed KAMFEE-X64 is an enhanced version of KAMFEE [6] which was published by authors in 2003. KAMFEE-x64 has a 64 bits basic block (dynamically key-controlled symmetric block cipher) which satisfies the today's speed, memory requirements, and make use of X64 bits microprocessors. This section describes the KAMFEE-X64 structure, and characteristics. The following section is a comparison between RIJNDAEL and KAMFEE-X64 from speed, security and memory requirements point of view.

2.2 Key generation

KAMFEE-X64 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 Fig. 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-1} is to present the encryption key of round (r-1), Fig. 2 shows the key generation process.

2.3 Key length dependency

The length of the proposed system basic block length is 64 bits. KAMFEE-X64 has three different block lengths, which are multiple of basic block length. So KAMFEE-X64 of 128 bits block length consists of two basic blocks, and 265 bit block consists of four different basic blocks.

As KAMFEE-X64 is a dynamically controlled cipher, then expanded key length will select few parameters of cipher structure;

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 KAMFEE-X64 structure.
3. Number of rotors' wheels, depends on key length too as shown in the following table (2)

Table (1) KAMFEE-X64 number of rounds

User key length (bytes)	**Block length (bytes)**	**Number of basic blocks**	**Number of rounds**
< 8	8	1	2
< 16	16	2	4
< 32	32	4	8

Table (2) KAMFEE-X64, number of rotors' wheels

User key length (bytes)	**Number of rotors' wheels**
< 8	8
< 16	16
< 32	32

2.4 Single round structure

KAMFEE-X64 round consists of three consequent steps. Step one is the key dependency, second step is substitution, and the third step is permutation.

2.4.1 *key dependency*

The first step is the round key addition. The modulo 264 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 (3) shows the key addition process.

2.4.2 *Substitution*

Substitution is a nonlinear operation that enhance the security of the block cipher. A 64 bits substitution boxes (sboxes) are used. Number of used sboxes depends on the number of basic blocks, which means, KAMFEE-X64 with 8 bytes block uses one sbox, while the 16 bytes version uses 2 different sboxes, and the 32 bytes version uses 4 different sboxes.

In order to make sbox depending on user key, KAMFEE-X64 arranged its static sboxes using the generated round key. Figure 4. Shows the substitution boxes arrangement using round key.

From implementation point of view, a static 64 bits sbox is impossible (each sbox contains 7.3819 bytes), so a 64 bits affine transformation is used to dynamically generate sbox.

$$Y = A * X + K(r,n) \qquad (3)$$

Where X ; is the plain block (64 bits)
Y ; is the cipher block (64 bits)
A ; is the pre-chosen constant (64 bits)
$K(r, n)$; is the round key basic block (64 bits)
r: round number
n: basic block order (1,2, or 4)
the pre-selected constant A, presents the cipher static data, and $K(r, n)$, achieves the key dependency.

Decryption process using 64 bits inverse affine transformation as follows

$$X = (Y - K(r,n)) * A^{-1} \qquad (4)$$

The main constrain here in sbox generation is the pre-selected constant A. It should be reversible module 2^{64} ($A * A^{-1} = 1$), in order to make the affine transformation reversible.

2.4.3 *permutation*

Permutation is re-arrangement of plaintext block contents which provide some sort of diffusion in the resulted ciphertext block. There are two successive permutations being used in KAMFEE-X64.

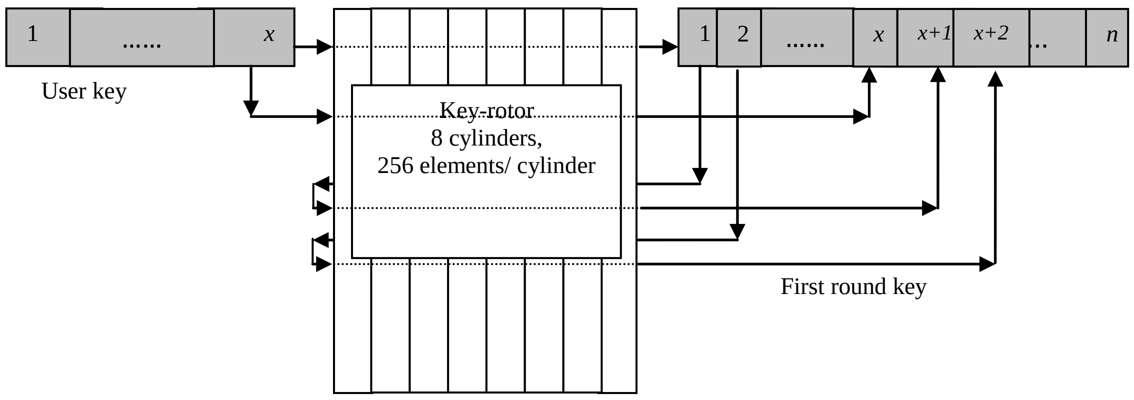

Fig. 1. First round key generation from user key

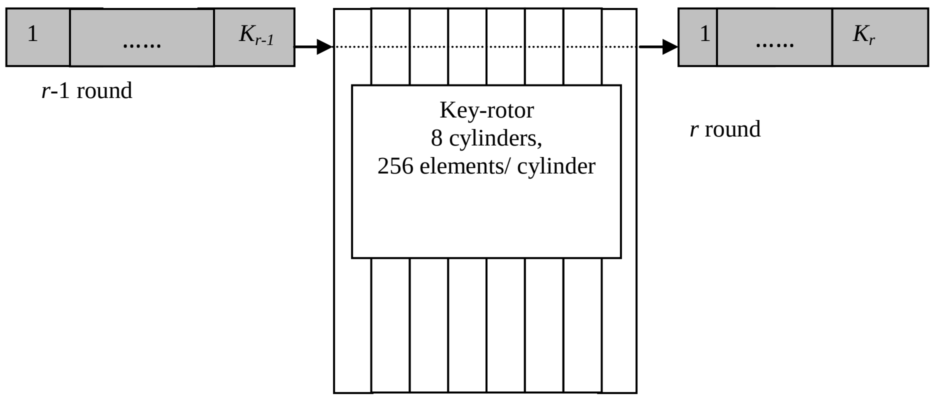

Fig. 2. Round r key generation

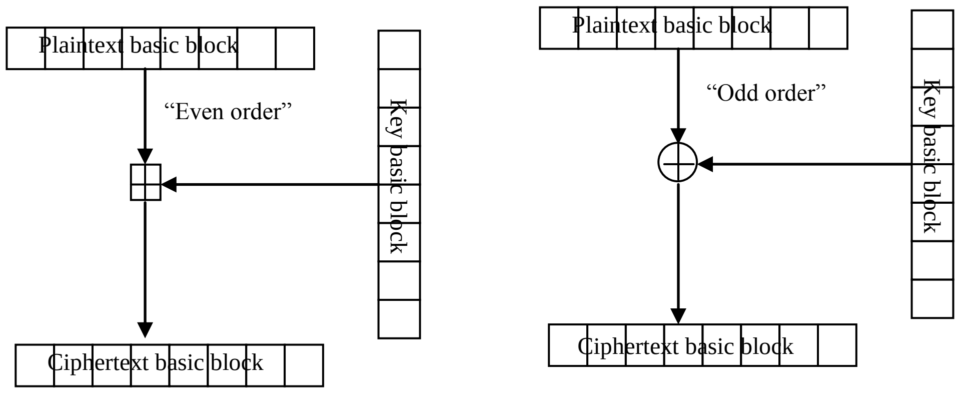

Fig. 3. Key dependency

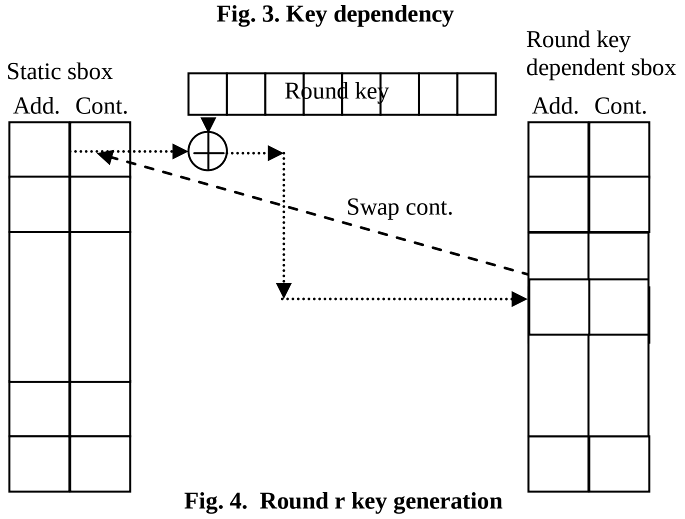

Fig. 4. Round r key generation

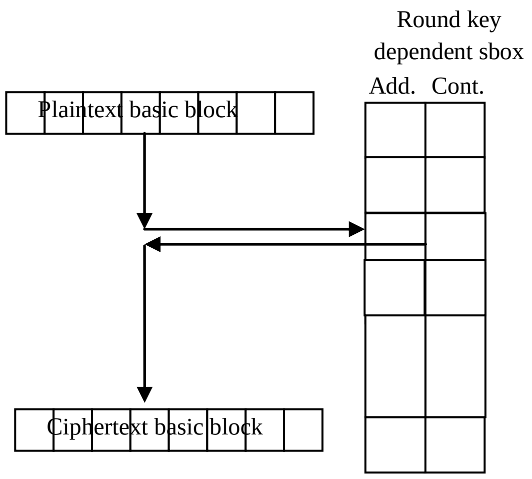

Fig. 5. Substitution process

2.4.4 Basic block permutation

In this step basic block is divided into four sub blocks each contains two bytes. The first two sub blocks form a 2x2 matrix and the last two sub blocks form another 2x2 matrix. Each 2x2 matrix is multiplied with two different 2x2 permutation matrix. Multiplication result re-form 8 bytes ciphertext basic block.

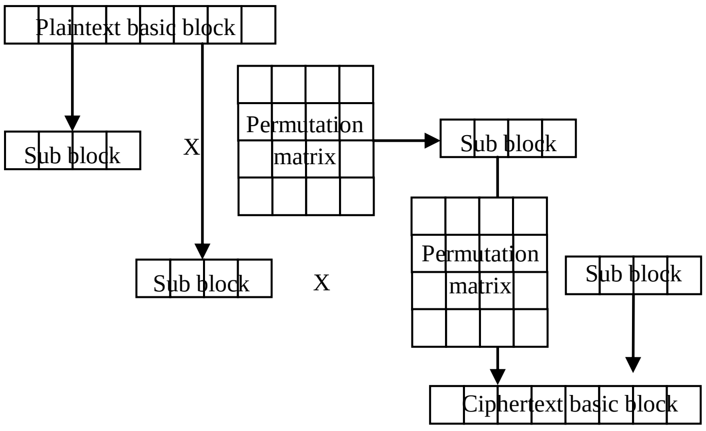

Fig. 6. Basic block permutation process

A new permutation matrix is generated in each round by generating 8 extra bytes after round key generation, using the key rotor.

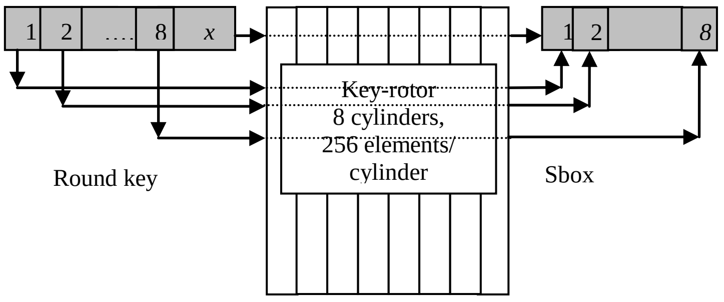

Fig. 7. Permutation matrix generation

2.4.5 Basic blocks mixing

This step is only used in the 16 bytes and 32 bytes versions of KAMFEE-X64. As this step needs more than one basic block mixing process. Each basic block (even orders) is mixed with the previous basic block using modulo 2^{64} addition (if destination basic block order is odd), and 64 bits XORING (if destination basic block order is even).

$$C[n] = P[n] \otimes P[(n-1) \bmod m] \quad (5)$$

n : basic block order (even)
m: total number of basic blocks

$$C[n] = P[n] + P[(n-1) \bmod m] \quad (6)$$

n : basic block order (odd)
m: total number of basic blocks

2.4.6 Rotor

The rotor is a very efficient cryptosystem from security point of view as the ciphertext statistics is almost flat. Another strength of rotor cryptosystem is its period. Long period enhances cryptosystem performance and makes it looks like one time pad cryptosystem as long as the length of ciphertext block that produced by the system doesn't exceed cryptosystem's period. The main weak points of rotor cryptosystems are;

Huge number of data required for its implementation

Processing speed as it consists of successive look up table processes, depending on the number of rotor's wheel.

Equation (5) shows the amount of static data required to implement a rotor of 8 bits word width.

$$RotorStaticDataSize = 256 * w \quad (7)$$

w : number if rotor wheels

To solve these weak points, a new implementation technique called successive affine transformation [1] is being used to implement rotor. Successive affine transformation is so fast and need few amount of data (data size in bytes is twice number of wheels).

$$SuccsessiveAffineTrnas.StaticData = 2 * w \quad (8)$$

w : number of rotor wheels

the following equation shows the mathematical description of successive affine transformation.

$$C[n] = ((A[n] * C[n-1] + Pos[n]) + Key[n \bmod KeyLength]) \bmod 256 \quad (9)$$

C : ciphertext character
n : wheel order
POS : wheel rotation order
Key : generated round key
$KeyLength$: generated round key length = cipher block length.

Affine functions here uses 28 modulo addition and 28 modulo multiplication. After a complete operation the first wheel of successive affine transformation is rotated by adding 1 to its POSvariable. If POS exceeds 256 it overflows, then POS resets to zero, and rotate the second wheel. And so on till the last wheel.

The following algorithm shows the successive affine transformation rotation process.

```
Rotate_wheel(n)
{
        POS[n] = POS[n] + 1 mod (256)
        If (POS[n]==0)
        {
                POS[n]=0;
                Rotate_wheel(n+1);
        }}
```

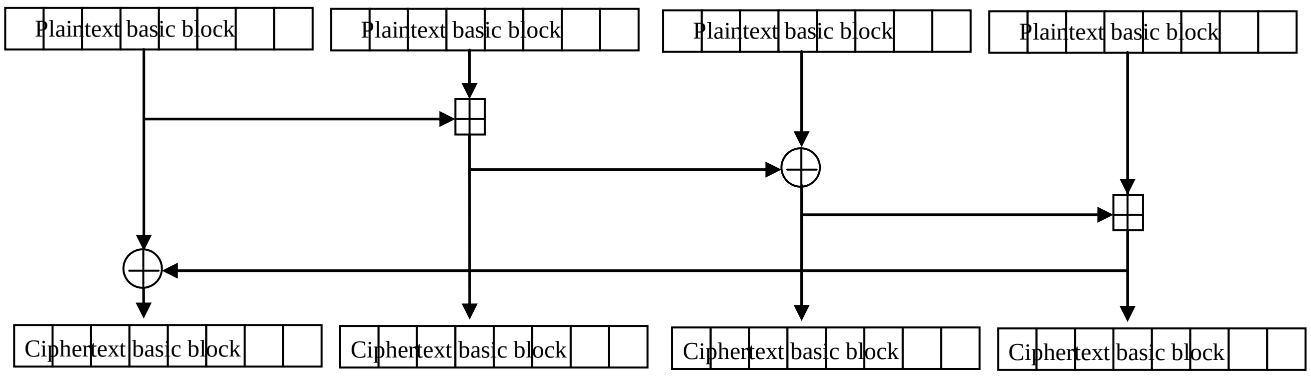

Fig. 8. Basic blocks mixing

The following equation shows inverse successive affine transform

$$C[n-1]=A^{-1}[n]*(C[n]-Key[n \bmod KeyLength]-Pos[n] \quad \text{.....................} \; (10)$$

$A^{-1}[n]$: is the inversion of $A[n]$ (pre-chosen nth constant)

3. KAMFEE-X64 OVERALL STRUCTURE

3.1 KAMFEE-X64 rounds

The number of rounds of KAMFEE-X64 depends on user key length as shown in table (1). Using the successive affine transformation in the middle of its rounds enhances KAMFEE-X64. Figure (9) shows the overall structure of KAMFEE-X64 cryptosystem.

Fig. 9. KAMFEE-X64 overall structure

3.2 KAMFEE-X64 Single Round

The structure of a single round of KAMFEE-X64 is shown in the following script.

```
Expand_user_key();
Block_length=key_length; /* table (1)*/
rotor_wheels=key_length; /*table (2)*/
number_of_rounds= length/4; */table (1)*/
round_number=0;
Enc()
{
round_number++;
Key_dependency();
substitution();
Permutation_part1(); /*basic block permutation*/
If(number_of_basic_blocks>2)          permutation_part2();
/*mixing basic blocks*/
If(round_number==(( number_of_rounds /2)))
successive_affine_transfomation();
}
```

As mentioned before, there are three different versions of KAMFEE-X64, shown in following table (3).

The basic block mixing process is only used in the 2nd and 3rd version of KAMFEE-X64. Figures (10) to (12) single round structure of each version.

3.3 Secret data groups

This section discusses the secret data groups that was used in KAMFEE-X64. Then compare between the amount of secret data used in KAMFEE-X64, and RIJNDAEL (AES).

Table (3), features of KAMFEE-X64 versions

	KAMFEE-X64, 64 bits	KAMFEE-X64, 128 bits	KAMFEE-X64, 256 bits
Key length (bytes)	8	16	32
Block length (bytes)	8	16	32
# basic blocks	1	2	4
# rounds	2	4	8
# rotor wheels	8	16	32

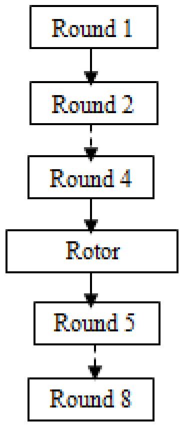

Fig. 9. KAMFEE-X64 overall structure

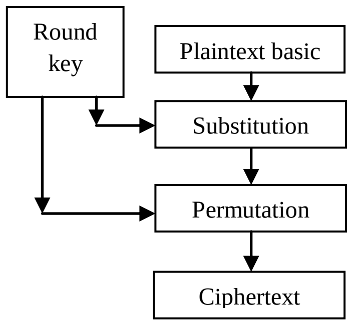

Fig. 10. KAMFEE-X64, 64 bits single round structure

3.3.1 User key

KAMFEE-X64 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.

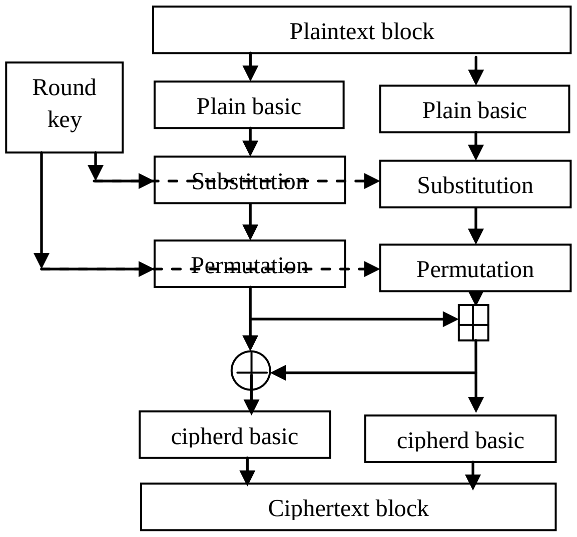

Fig. 11. KAMFEE-X64, 128 bits single round structure

Round key is generated in each round using key rotor. As key rotor consists of 8 cylinders each contains 256 bytes presented by ASCII characters. Key rotor is implemented using 8 successive affine transformation, each presents a wheel of the rotor. Total number of available affine transformation is ($\phi(2^8) \times 2^8$). The static data required per affine transformation is 2 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.

3.3.2 Substition boxes

The number of sboxes used by KAMFEE-X64 rquals to number of its basic block. Sboxes are generated using 64 bits affine transformation, which uses a 64 bits pre-selected random constant and a key basic block of 64 bits length [equations (3) and (4)]. Sboxes can be generated on the fly –as needed so the total number of available s-boxes equals to $\phi(2^{64}) \times (2^{64})$.

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

The total number of static data used by sbox is 2 x 64 = 128 bytes.

RIJNDAEL uses eight bits s-box. Generated by using 8 bits affine transformation, which uses two different constants. Those are 8×8 bits binary matrix (the first column is selected and the other columns are a rotated version of the previous column) and 8×1 bits binary matrix. The s-box can be generated on the fly as needed. The total number of available s-boxes is ($\phi(2^8) \times 2^8$). The total number of trials to find the used s-boxes equals to $(\phi(2^8) \times (2^8))!$.

3.3.3 Permutation boxes

Permutation in the KAMFEE-X64 consists of two steps. The first step is permutation of the basic block contents which uses two different square matrices generated from secret key using key rotor. The total number of available matrices are 256^4. As the permutation matrix generated from round key and key rotor, the trials to find the permutation matrix is already calculated in secret key brute force attack. The total amount of static data is 4 bytes. The number of permutation boxes equals to number of basic blocks.

The second step is mixing of basic blocks as described in equations (3) and equation (4), needs no secret data.

RIJNDAEL uses two types of permutations. First type is a script re-arrange basic blocks –8 bits- called SHIFTROW(). Second type called MIXCOL(), which multiplies a block of four characters with a pre-selected 4×4 square matrix, the first column is selected and next column is a rotated version of the previous column, so the total number of available matrices is (256^4). The total number of trials to find the used matrices equals to $(256^4)!$

3.3.4 Sucessive affine transformation

Successive affine transformation is an efficient implementation of the Rotor, which consists of a successive eight bits affine function, number of affine transformations which presents the rotor wheels equals to key length in bytes. Each affine transformation contains a pre-selected random constant and a byte from the previous round's key. Total number of available affine transformation is ($\phi(2^8) \times 2^8$). The static data required per affine is 2 bytes.

4. ANALYSIS OF KAMFEE-X64

4.1 Brute Force attack

Considering secret data used in KAMFEE-X64, the total number of trials to break KAMFEE-X64 ciphertext shown in the following table (4). RIJNDAEL brute force attack is shown in table (5).

Table (4), KAMFEE-X64 brute force attack

	KAMFEE-X64,64	KAMFEE-X64,128	KAMFEE-X64,256
Key	$2^{64} \approx$ 1.84E19	$2^{128} \approx$ 3.8E38	$2^{256} \approx$ 1.16E77
Key Rotor (key generation)	$(2^7)P_8 \times (2^8)P_8 \approx$ 9.52E35		
Substitution	$2^{63}P_1 \times 2^{64}P_1 \approx$ 1.7E38	$2^{63}P_2 \times 2^{64}P_2 \approx$ 2.89E76	$2^{63}P_4 \times 2^{64}P_4 \approx$ 8.38E152
Permutation	none		
successive affine transformations	$2^7P_8 \times 2^8P_8 \approx$ 9.52E35	$2^7P_{16} \times 2^8P_{16} \approx$ 4.11E71	$2^7P_{32} \times 2^8P_{32} \approx$ 5.9E141
Total trials to attack KAMFEE-X64	2.85E129	3.86E222	5.45E407

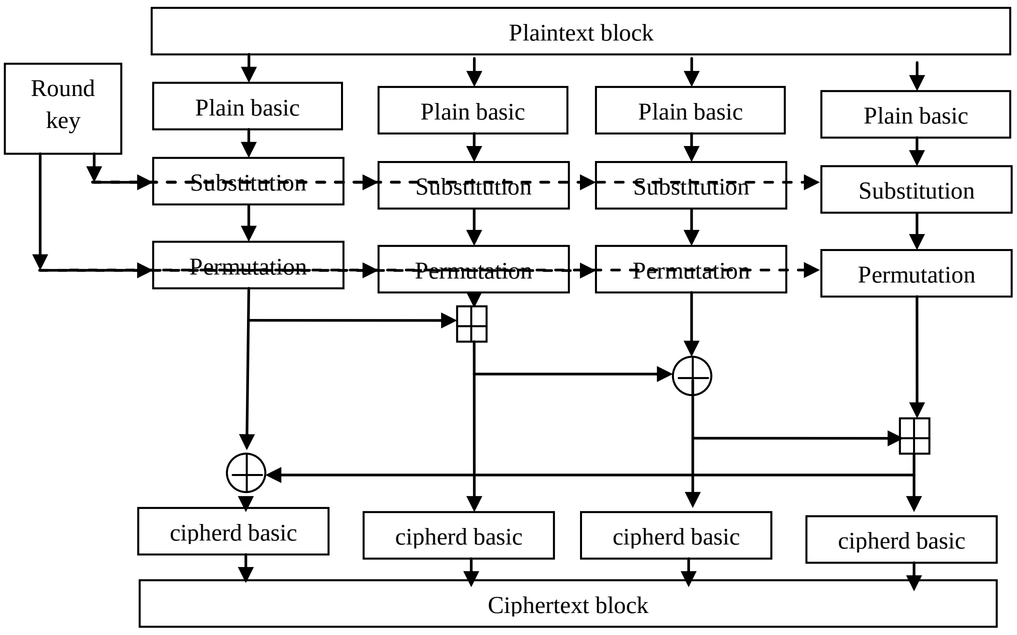

Fig. 12. KAMFEE-X64, 256 bits single round structure

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

4.2 Static data requirements

The following tables (6), and (7) summarize the static data requirements for KAMFEE-X64, and RIJNDAEL respectively.

Table (6), KAMFEE-X64 static data requirement

	KAMFEE-X64,64	KAMFEE-X64,128	KAMFEE-X64,256
Key	24		
Substitution	128	256	512
Permutation	4	8	16
successive affine transformations	24	48	96
Total required memory (bytes)	180	336	648

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

4.3 Period

The period of KAMFEE-X64 is the product of two elements. These are rotor period, block length. The following table (8) shows KAMFEE-X64 period. RIJNDAEL period depends on its block length only, as shown in table (9).

Table (8), KAMFEE-X64 period in bytes

	KAMFEE-X64,64	KAMFEE-X64,128	KAMFEE-X64,256
Key	8	16	32
# of wheels	8	16	32
successive affine transformations	256^8	256^{16}	256^{32}
Total period	$256^8 \times 8 \approx$ 1.48E20	$256^{16} \times 16 \approx$ 5.44E39	$256^{32} \times 32 \approx$ 3.7E78

Table (9), RIJNDAEL period in bytes

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

4.4 Language Statistics

Due to the rotor used in the middle of KAMFEE-X64, key cryptanalysis become extremely difficult. Cryptanalyst faces many difficulties to produce a general linear expression between input plaintext and output ciphertext. The main difficulty is the rotor, which 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 figures (13) to (18) show a comparison between ciphertext statistics of KAMFEE-X64, 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 KAMFEE-X64, and RIJNDAEL

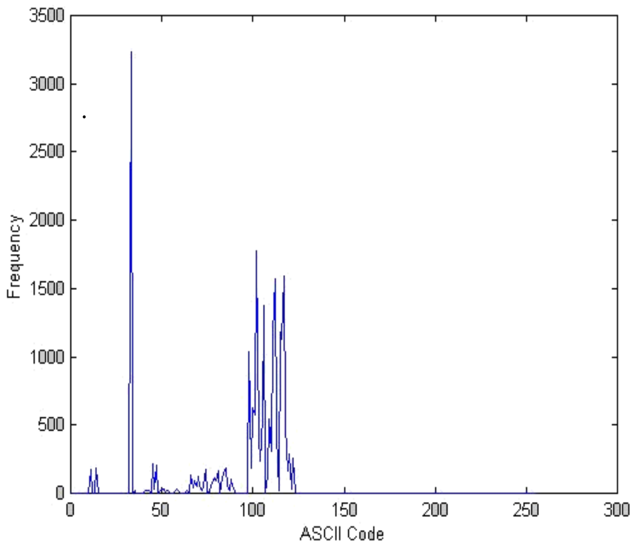

Fig. 13. Plaintext statistics

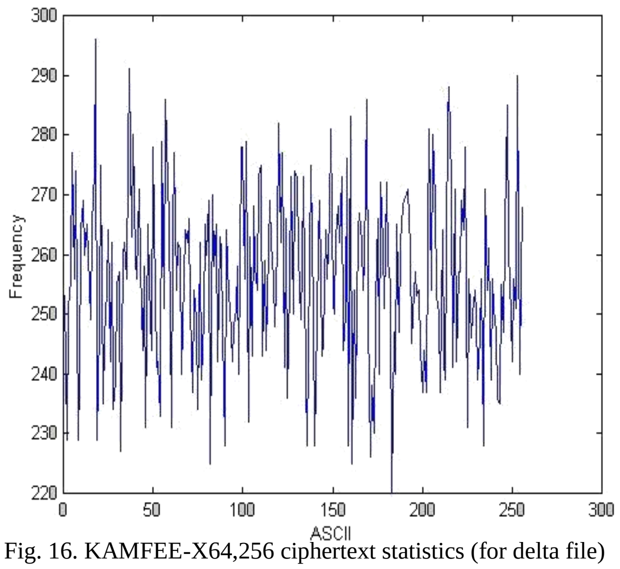

Fig. 16. KAMFEE-X64,256 ciphertext statistics (for delta file)

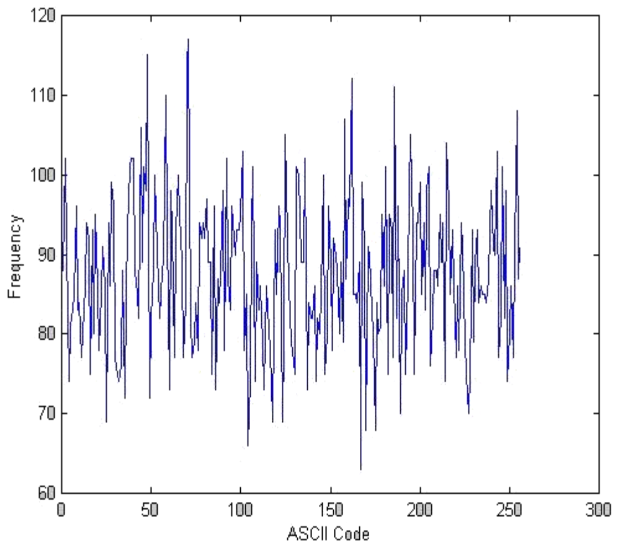

Fig. 14. KAMFEE-X64,265 ciphertext statistics

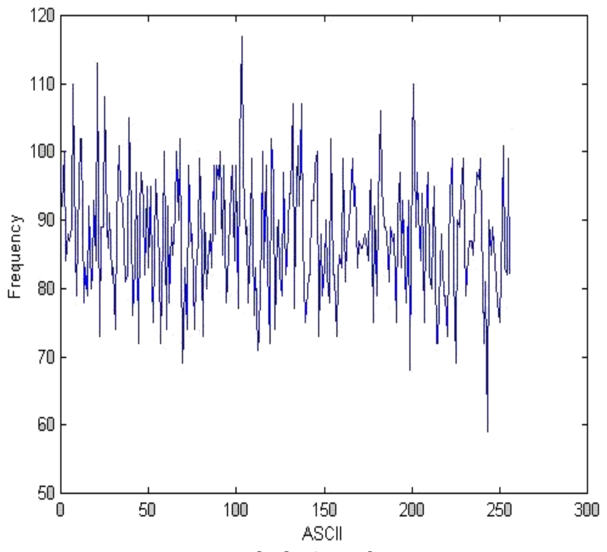

Fig. 17. Rijndael,265 ciphertext statistics

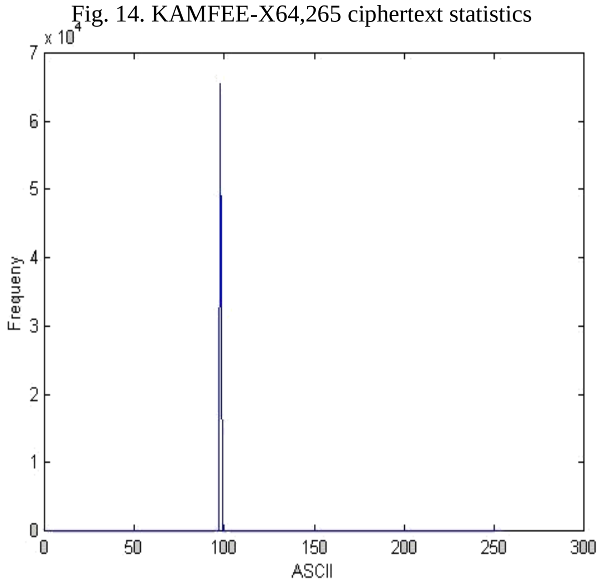

Fig. 15. Delta plaintext statistics

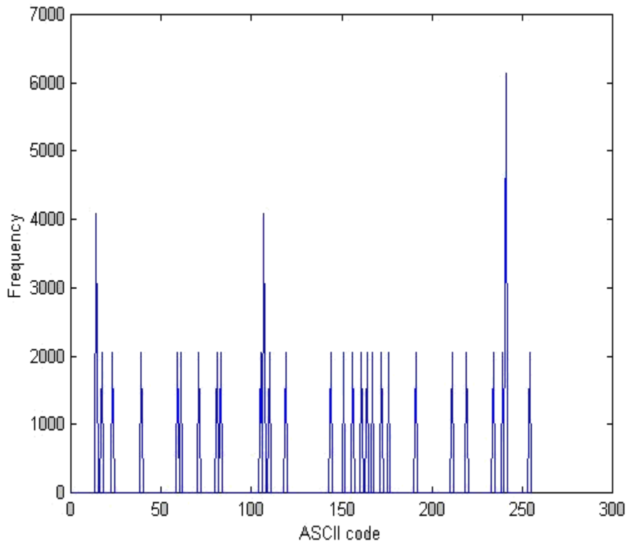

Fig. 18. Rijndael,265 ciphertext statistics (for delta file)

5. CONCLUSIONS

The proposed KAMFEE-X64 cryptosystem uses the conventional main cryptographic terms of substitution and permutation, and key dependency back-to-back with rotor to achieve the requested goals of like-perfect-statistics, large period, and resistance to linear and differential cryptanalysis.

KAMFEE-X64 is a 64 bits based cryptosystem, that take advantages of newly 64 bits microprocessors and operating systems available in market to increase cryptosystem performance from speed and security point of view.

Even KAMFEE-X64 uses rotor ciphering technique, its static memory requirements is less than one kbytes.

KAMFEE-X64 comes in three different version a small version (64 bits) of 8 bytes block/key length, only two rounds plus 8 wheels rotor (successive affine transformation), which make it suitable for mobile devices. The other versions (128 bits and 256 bits) are more complex and provides more security and larger period.

KAMFEE-X64 huge period make it suitable for encrypting huge messages without the need of operation modes.

6. REFERENCES

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

[2] C.Shannon, Conication Theory of Secrecy Systems, Bell System Tech.. J., Vol. 28, 1949.

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

[4] J. Daemen, J. T. Rijmen, “AES Proposal: RIJNDAEL. AES Algorithm Submission”, 1999.

[5] 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

[6] 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

[7] 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.

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

[9] "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.

[10] "REBC2 cipher"; IEEE Africon 2007. Proceedings of the Africon 2007, September 26-28, 2007, Namebia, paper ID 624.

[11] “URESC, Unbalanced Rotor Enhanced Symmetric Cipher”; The 14th IEEE Mediterranean Electrotechnical Conference, Ajaccio, France, May 5-7, 2008, paper ID t1-sd1018.