Proposed a Variable Length Block Cipher Algorithm

Rana Saad Mohammed

Computer Science Department, Education Collage, Al-Mustansiriyah University
Baghdad, Iraq
ranasaad_m@yahoo.com

Abstract:

This paper will introduce a propose algorithm to improve an elastic block cipher algorithm by benefiting from an efficient properties of a secure cryptographic mode New Plaintext-Ciphertext Block Chaining mode (NPCBC) and by creating a good key schedule and two new S-Boxes. This paper will describe the concept of a proposed elastic block cipher that refer to stretch the supported block size into any length up to twice of the original block size. Also it defines a method for converting any existing block cipher into a new elastic block cipher. The results show that the security is increased by using the multiplication-addition operations in NPCBC mode which it provides the confusion and diffusion properties that cause difficulty of attacks on a new algorithm. And Also by using a good key schedule and two new S-Boxes which they increase the complexity with keep on speed of a new algorithm when compare it with a traditional algorithm which it has a weakness point when encrypt multiple blocks with using a fixed secret key.

Keyword: Block Cipher Design, Fixed Length Block Cipher, Variable Length Block Cipher, Elastic Block Cipher, NPCBC mode, S-boxes, key schedule.

اقتراح خوارزمية تشفير الكتلي ذات حجم متغير

م.د. رنا سعد محمد
الجامعة المستنصرية – كلية التربية – قسم علوم الحاسبات
Ranasaad_m@yahoo.com

المستخلص

هذا البحث سوف يقدم خوارزمية مقترحة لتحسين خوارزمية التشفير الكتلي المطاطية بواسطة الاستفادة من خصائص الكفوءة لطريقة التشفير الامنة (NPCBC) و كذلك بواسطة اقتراح جدولة المفاتيح و صندوقين تعويضين المعتمدين على المفتاح و هما متغيران في الحجم عند الادخال و الاخراج .

هذا البحث سوف يقدم مفهوم تشفير الكتلي جديد مطاطي الذي يشير الى امتداد حجم الكتلة لخوارزميات التشفير الكتلي ثابتة الحجم مثل (SERPENT,TWOFISH,AES RC6,MARS الى ضعف حجم الاصلي للكتلة . كذلك يعرف طريقة لتحويل اي تشفير كتلي ثابت الحجم الى تشفير كتلي جديد مطاطي .

النتائج بينت ان الخوارزمية المقترحة حققت في زيادة امنية هيكلة الخوارزمية عند استخدام عمليتين الضرب_الجمع الموجودة في طريقة NPCBC التي توفر خصائص التشويش و الانتشار التي تسبب صعوبة الهجوم على خوارزمية الجديدة. كذلك بواسطة استخدام جدولة المفاتيح و صندوقين تعويضين المعتمدين على المفتاح و هما متغيران في الحجم عند الادخال و الاخراج التي تؤدي الى زيادة التعقيد و مع المحافظة على سرعة خوارزمية الجديدة عند مقارنتها مع خوارزمية المطاطية السابقة التي هي مهددة عندما هي تعالج مدخلات ذات اطوال متعددة تحت مفتاح سري ثابت.

الكلمات المفتاحية: تصميم نظام تشفير كتلي، تشفير كتلة ذات حجم ثابت، تشفير كتلة ذات حجم متغير، نظام تشفير كتلة مطاطي، طور NPCBC ، صندوق التعويض، جدولة المفاتيح.

1. Introduction

A cryptography algorithm that has a fixed size input is called a Fixed Input Length (FIL) primitive. For example, all block ciphers are common FIL primitives. A block cipher algorithm transforms a block of unencrypted text (commonly called "plaintext") into a block of encrypted text (commonly called "ciphertext") under the action of a secret key. The plaintext and ciphertext have the same length when transformed through a block cipher. Decryption process applies a reverse transformation of encryption process using the same secret key. A block size is a length of the block. It is can be 64 or 128 bits [1][2].

The recent applications of internet and wireless communications need develop of cryptography algorithms that operate on Variable Input Length (VIL) primitives. A need therefore exists for techniques that provide constructions made of VIL primitives that are efficient and that provide relatively high security. These techniques are used to encode a message to create an encrypted resultant message and to decode the encrypted resultant message to recreate the original message. On encryption and its corresponding decryption technique is more efficient than a comparable conventional encryption and decryption technique, while a second encryption and its corresponding decryption technique has relatively high security. These constructions may be implemented in any number of ways, such as through hardware devices or computer systems [2][3].

In [4][5] proposed an elastic block cipher but it has a weakness point when it encrypts multiple blocks with using a fixed secret key [6]. In [7] gave a method of providing a Feistel-based variable length block cipher.

This paper proposes a new elastic block cipher algorithm with any network (substitution-permutation (SP) or Feistel) that allow us to “stretch” the supported block size up to double of the original block size with do not use plaintext padding process. And also maintaining the diffusion property of traditional encryption algorithms and change their computational load proportionally to the increase of a size.

The organization of this paper is as follows. Section 2 explains the construction of new elastic block ciphers from existing block ciphers. Section 3 presents a flowchart of algorithm. Section 4 presents a practical implementation and conclusions in the last section.

2. The construction of a New proposed elastic block cipher

A. The proposed network structure

The proposed algorithm makes the functions of the encryption and decryption of an existing block cipher process on blocks with size 2b bits rather than size b bits of the original block cipher.

The proposed network structure uses the same cycle and round function of the original block cipher. And it makes a substitution and permutation on b+y bits where $0 \le y \le b$.

This proposed structure involves initial and final whitening, use the cycle of the existing fixed-length block cipher, and use proposed key schedule and two S-boxes. Figure (1) shows this structure.

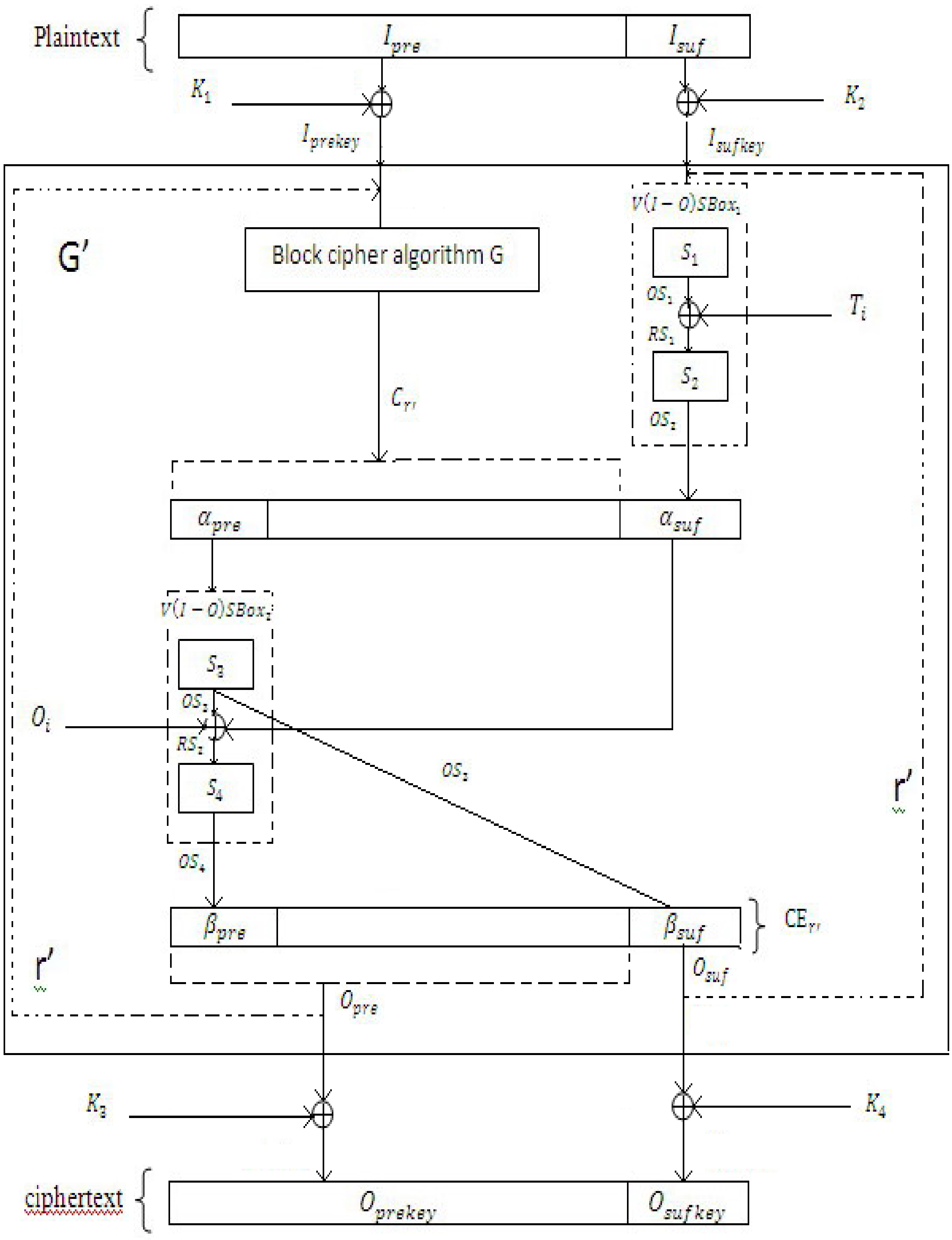

Fig. (1): Proposed network structure

Table (1) describes the notations and its definitions for construct this new structure:

Table 1 Proposed notations and its definitions

Notation	Definition
G	Any traditional block cipher algorithm that has a fixed-length block size
r	The number of cycles in G. for example: - 1 round of SP network= 1 cycle - 2 round of Feistel network = 1 cycle - 4 round of RC6 = 1 cycle
b	The length in bits of the input block to G
y	An integer number in the range [0, b]
$\mathbf{G'}$	An proposed elastic and modified of G with an input length (b+y) bit
$\mathbf{r'}$	The number of rounds in G′. such that - r′ = r in SP network - r′ = r*2 in Feistel network
$\mathbf{I_{pre}}$	The prefix of input with b bits
$\mathbf{I_{suf}}$	The suffix of input with y bits.
$\mathbf{I_{prekey}}$	Output of XOR operation between I_{pre} and K_1 in initial whitening step.
$\mathbf{I_{sufkey}}$	Output of XOR operation between I_{suf} and K_2 in initial whitening step.
$\mathbf{K_i}$	Subkeys are derived from key schedule.
$\mathbf{C_{r'}}$	The output from one round function of G
$\mathbf{CE_{r'}}$	The output from one round function of G′.
$\mathbf{O_{pre}}$	The prefix of $CE_{r'}$with b bits.
$\mathbf{O_{suf}}$	The suffix of $CE_{r'}$ with y bits.
$\mathbf{O_{prekey}}$	Output of XOR operation between $O_{pre}$and K_3 in final whitening step.
$\mathbf{O_{sufkey}}$	Output of XOR operation between O_{suf} and K_4 in final whitening step.
$\boldsymbol{\alpha_{pre}}$	The left of $C_{r'}$ with y bit and consider input of $V(I-O)SBox_2$.
$\boldsymbol{\alpha_{suf}}$	The output of $V(I-O)Sbox_1$ with y bit.
$\boldsymbol{\beta_{suf}}$	The output of S_3 with y bit (OS_3).
$\boldsymbol{\beta_{pre}}$	The output of $V(I-O)Sbox_2$ and also it is the left of $CE_{r'}$ with y bit.
NPCBC[7]	Shortcut of "New Plaintext-Ciphertext Block Chaining"
$\mathbf{V(I-O)SBox_1}$	Shortcut of the proposed first Variable Input- Output Substitution Box with y bits size of input and output. This box is key dependent Sbox. It has two of no key dependent Sboxes S_1 and S_2. The output of S_1 is exclusive-OR (XOR) with key T_i and the result is the

	input of S_2 to get output α_{suf} with size y bits.
S_1	1st no key dependent Sbox with size y bits of input and y bits of output.
OS_1	The output of S_1 with size y bits
RS_1	The result of XOR operation between OS_1 and left bits of key T_i
S_2	2nd no key dependent Sbox with size y bits of input and y bits of output.
OS_2	The output of S_2 with size y bits
T_i	Set of keys when $V(I-O)SBox_1$ is depended on them. These keys are derived from NPCBC mode. The number of these keys depends on r′.
$V(I-O)SBox_2$	Shortcut of the second Variable Input- Output Substitution Box with size y bits of input and y bits of output. This box is key dependent Sbox and its contain two no key dependent Sboxes S_3 and S_4 when the output of S_3 is exclusive-OR (XOR) with α_{suf} and O_i, and the result is the input of S_4
S_3	3rd no key dependent Sbox with size y bits of input and y bits of output
OS_3	The output of S_3 with size y bits
RS_2	The output of XOR operation between the output of S_3, α_{suf} and left bits of key O_i
S_4	4th no key dependent Sbox with size y bits of input and y bits of output
OS_4	The output of S_4 with size y bits
O_i	Set of keys when $V(I-O)SBox_2$ is depended on them. These keys are derived from NPCBC mode. The number of these keys depends on r′
IV_i	Set of initialize vectors with b bits derived from the user key. These set consider the inputs of NPCBC mode to generate T_i and O_i

The following proposed steps convert an existing block cipher (G) with a fixed size b-bit into its new version (G′) that can process b + y bits:

1. Set the number of rounds (r′). It is equal to (r) in a Substitution_Permutation network of exiting block cipher and equal to r*2 in a balanced Feistel network.
2. In prior to the 1st round function of G′, apply initial whitening step which is XOR process between the input plaintext with size b+y (I_{pre}, I_{suf}) and the keys (K_1, K_2) are generated from a key schedule section. The output from this step is (I_{prekey}, I_{sufkey}).
3. The output from step (2) (I_{prekey}, I_{sufkey}) is input to round function G′ with r’ round. The round function is contain on any exiting block cipher G, $V(I-O)Sbox_1$ and $V(I-O)Sbox_2$, see previous figure (1). I_{prekey} with size b bit is input to G. I_{sufkey} with size y bit is input to $V(I-O)Sbox_1$. The leftmost of output G with size y bit (α_{pre}) is input to $V(I-O)Sbox_2$. The output from each round function is (O_{pre}, O_{suf}) with size b+y bit.

4. After the last round function G′ apply final whitening step which is XOR process between the output of last round with size b+y (O_{pre}, O_{suf}) and the keys (K_3, K_4) are generated from a key schedule section. The output from this step is ciphertext with size b+y (O_{prekey}, O_{sufkey}).
   The round function in the proposed algorithm is used key dependent S-box which provide high security compared with elastic algorithm [1,4,5]. The proposed algorithm use swap step for OS_3 rather than α_{pre} which used in elastic algorithm to increase the security against distinguish attack.

B. The proposed key schedule

The purpose of key schedule is to produce additional keys or increasing the original key length. In the implementations, a pseudorandom will be used as first part of proposed key schedule. The second part is derived from the structure of NPCBC mode to increase the randomness of the expanded key bits compared with those are produced by an existing key schedules.

The first part is used to produce a random sequence with length (r′+4)*b bit. This sequence consider as an input of a second part to decrease the possibility of the attacks.

The sender and receiver are agree on the key with length (5*b) bit where a first b bit block consider as a key of G. The remain (4*b) bits are divided into 4 blocks (IV1, IV2, IV3, IV4) where each with b bit and they consider as an initial vector of the second part of the key schedule of G′ to produce (r′+4) round subkeys of G′.

For example, they are agree on a key with length 640 bit (80 byte) where first 128 bit consider as a key of G and the remain 512 bits are divided into 4 blocks (IV1, IV2, IV3, IV4) where each with 128 bit and they are consider as initial vector of the second part of key schedule G′ to produce (r′+4) round subkeys of G′.

The first part is any strong pseudorandom which can produce sequence with length (r′+4)*128 bit and have randomness property. The sender produce this sequence as input to a second part of the proposed key schedule to produce random (r′+4) subkeys and each with 128 bit. Where (r′+4) mean generate r′ subkeys by depend on number of round r′, the number 4 mean generate 2 subkeys for initial whitening and other 2 subkeys for final whitening.

A proposed second part of key schedule is derived from the structure of NPCBC mode. NPCBC mode has a better security compared with CBC mode of block ciphers [8]. The deriving is similar to a structure of NPCBC but different in use random sequence as input (I_i) rather than plaintext. The output are subkeys (O_i) rather than ciphertext. And also different in use function F rather than use system of block cipher E_K in NPCBC mode.

Figure (2) shows this proposed 2nd part. Function F uses three different stages of AES [9,10] structure to get permutation and substitution. The stages are Substitute Bytes Transformation, ShiftRows Transformation, and MixColumns Transformation [9]. Also in the second part will use the multiplication $\odot_b$ –addition $\boxplus_b$ operations that provide the confusion and diffusion properties.

This paper can denote the generation of subkeys by the following equations where suppose b=128 bit and the output of a first part is consider as input to a second part which is divided into blocks I_1, I_2,… each with b bits.

2nd part of a proposed key schedule:

T_i=$(IV_1 \odot_{128} IV_2) \boxplus_{128}(IV_3 \odot_{128} (IV_4 \boxplus_{128}(IV_1 \odot_{128} IV_2)))$

O_{i-1}=$F(((IV_1 \odot_{128} IV_2) \boxplus_{128} IV_4) \odot_{128} IV_3)$

O_i=$F(((IV_2 \odot_{128} I_i) \boxplus_{128} O_{i-1}) \odot_{128} T_i)$

Where:

- i= 1 to (r'+4)
- T_i is key dependent of $V(I-O)SBox_1$
- O_i is key dependent of $V(I-O)SBox_2$
- $\odot_{128}$ denotes $x \odot_{128} y$ =$(x_1 \odot_{128} y_1, x_2 \odot_n y_2, \ldots, x_{128} \odot_{128} y_{128}) \in GF(2)^{128}$, where

  $x_b \odot_{128} y_b$= 1 mod $(2^{128}+1)$, x_b *and* y_b=(0,0,. . . ,0),

  $x_b . y_b$mod $(2^{128}+1)$, x_b *and* $y_b \neq$ (0,0,. . . ,0),

  x_b =(0,0,. . . ,0), *and* $y_b \neq$ (0,0,. . . ,0),

  $x_b \neq$ (0,0,. . . ,0), *and* y_b= (0,0,. . . ,0),
- $\boxplus_{128}$ means $x \boxplus_{128} y$ =$(x_1 \boxplus_{128} y_1, x_2 \boxplus_{128} y_2,$ …. $., x_{128} \boxplus_{128} y_{128}) \in GF(2)^{128}$,where $x_b \boxplus_{128} y_b$=$(x_b + y_b)$mod 2^{128} .

The sender uses these subkeys in order with i from 1 to r′+4 while the receiver uses these subkeys in reverse order with i from r′+4 down to 1.

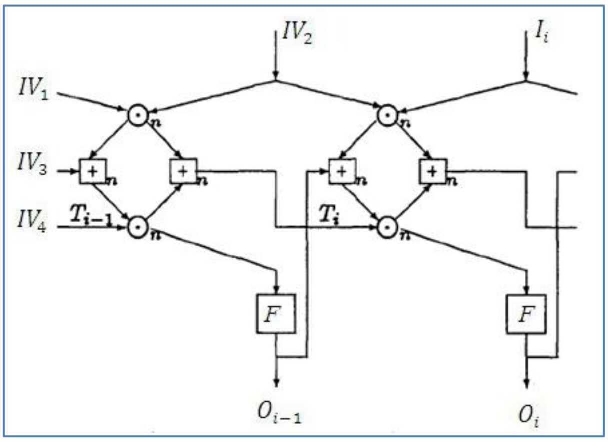

Fig. (2): 2nd part of a proposed key schedule

C. The proposed two S-boxes

1. $\underline{V(I-O)SBox_1}$

This Sbox is depend on key T_i so it is called key dependent Sbox and has non linear property. See figure (1) which show the work of this Sbox when the I_{sufkey} with y bit is input of $\mathrm{V(I-O)Sbox_1}$ which go to S_1 which its contain on table of bits. I_{sufkey} and its number of y bits are index of this table to out new bits with size y bit called OS_1. Then exclusive-OR (XOR) process is performed between the output of $S_1(OS_1)$ and left bits of key T_i with size y bit. Then the result, called RS_1, is the input of S_2 which it contains on table of bits and the input with its number of y bits are index of this table to out new bits with size y bit called OS_2. Then the α_{suf}takes the bits of OS_2 as output of $\mathrm{V(I-O)Sbox_1}$.

1.1 $\mathbf{S_1}$

This sbox can be written as a table with set of random elements is shown in table (2) and arranged by depending on number of bits (y) for I_{sufkey}. $\mathrm{S_1}$ has value and y bit for input I_{sufkey} as they index of table and output new value with y bit called OS_1. See the following procedure that shows the work of $\mathrm{S_1}$.

Procedure S_1 (I_{sufkey} as input, OS_1 as output)

R_1=y mod 8, where y is number of bits for I_{sufkey}

Case R_1 of

$R_{2[s]}=I_{sufkey}/8$,divide the input into sets of 8 bit and each set is saved in array called $R_{2[s]}$,where [s] is number of set.

$R_{3[s]}= S_1(R_{2[s]},8)$, $R_{2[s]}$ consider as input to S_1 when the value of $R_{2[s]}$ in hex and the number of y which equal to 8 as they index of table (1). The output is new sets of 8 bit and each set is saved in array called$R_{3[s]}$, where [s] is number of set.

$R_{3mer}=R_{3mer}\&R_{3[s]}$, merge the sets of array which each with 8 bit into one set with y bit.

$OS_1=R_{3mer}$

1 or 2 or 3 or 4 or 5 or 6 or 7: if (y<8) then

begin

R_4=left(I_{sufkey}, R_1),cut R_1bits value from left of input

$R_5= S_1(R_4, R_1)$, where $R_1$is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.

$OS_1=R_5$

end

else

if (y>8) then

begin

R_4=left(I_{sufkey}, R_1),cut R_1bits value from left of input

$R_5= S_1(R_4, R_1)$,where $R_1$is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.

R_6=right(I_{sufkey},y- R_1)

$I_{sufkey2}=R_6$

$R_{7[s]}=I_{sufkey2}/8$, divide $I_{sufkey2}$ into sets of 8 bit and each set is saved in array called$R_{7[s]}$, ,where [s] is number of set.

$R_{8[s]}= S_1(R_{7[s]},8)$

$R_{8mer}=R_{8mer}\&R_{8[s]}$, merge the sets of array which each with 8 bit into one set with y bit.

$R_9= R_5\& R_{8mer}$.

$OS_1=R_9$

end

end of case

Table 2 proposed S1

y	
1	0 1
2	0 3 1 2
3	1 5 3 0 4 2 6 7
4	0 E B 3 4 5 D 8 9 6 A F 1 7 C 2
5	1C 9 C 3 1D 16 13 19 8 1 A 5 2 0 E 10 1F 11 12 6 B 15 14 17 18 D 1A 1B 7 4 1E F
6	7 1 2 34 37 5 18 39 B 9 A 12 C 3C E 3 10 11 1D 13 14 23 16 17 29 4 6 2D 36 1A 3F 1F 3A 22 2F 15 24 30 26 27 1E 21 2A 2B 2C 1B 2E F 25 31 32 33 28 35 38 0 3E 20 1C 3B D 3D 8 19
7	45 1 58 66 35 5 3C 54 48 63 67 B 16 62 47 49 50 4A 12 13 0 15 7F 44 18 19 1A 3D 1C 43 1E 1F 20 21 5B 40 6F 25 71 27 32 29 2A 28 2C F 5A 2F 30 42 C 53 70 38 36 37 3B 39 3A 31 6 17 3E 41 33 3F 68 4 2B 73 46 2D 74 E 26 4B 4C 4D 4E 4F 5E 51 52 57 2 55 6A 6D 7 59 2E 23 5C 5D 76 5F 60 61 6B 9 64 D 24 69 1D A 56 3 6C 22 6E 7E 34 8 1B 14 11 75 10 77 78 65 7A 7B 7C 7D 79 72
8	0 A4 6B 3 19 C4 D2 7 8 CB A B C D E F F3 13 12 11 A6 D7 33 17 18 44 C0 1B 1C 23 8F 1F 20 21 8D 7F E0 25 CF 89 7B 52 2D 5E 2F EA E7 35 30 63 2C 16 AD 55 3F 32 6 39 9A 3B 8A 9C 7C 2E B3 CC CA AE 4 D0 E4 47 4B 71 4A 2B 4C 8B 4E 9B 66 51 29 53 54 E8 60 96 58 F1 FC 82 5C 75 86 5F D6 F2 62 40 64 65 56 CE 9F F8 6A 87 6C 3D 41 42 57 49 72 73 74 5D 76 C7 FB 79 59 28 2A 37 7E 27 EE A5 FE 83 5A 85 61 B9 E1 15 3C 4D 8C 36 8E 1E 70 B8 92 93 A2 C1 84 9 98 99 3A 7A 6D 5 45 68 A0 AB 94 A3 1 BE 90 A7 A8 1A AA 46 AC CD 95 FF B0 B1 B2 31 22 B5 BF DF 91 2 BA BB E2 BD 69 DC A9 43 C2 C3 38 C5 C6 77 A1 C9 6F 67 FD 97 34 26 9E D1 14 D3 F7 D5 50 F0 D8 80 DA DB BC DD 1D B7 DE 88 9D E3 48 E5 E6 B4 7D E9 ED EB EC 3E D9 EF 6E 4F AF 10 78 F5 F6 D4 81 F9 FA F4 B6 24 5B C8

1.2 S_2

This sbox can be written as a table with set of random elements is shown in table (3) and arranged by depending on number of bits for RS_1 . S_2 has value and y bit for input RS_1 as they index of table and output new value with y bit called OS_2. See the following procedure that shows the work of S_2.

Procedure S_2 (RS_1 as input, OS_2 as output)
R_{10}=y mod 8, where y is number of bits for RS_1
Case R_{10} of
0 : $R_{11[s]}$=RS_1/8 ,divide the input into sets of 8 bit and each set is saved in array called $R_{11[s]}$,where [s] is number of set.

$R_{12[s]}$= $S_2(R_{11[s]}$,8), $R_{11[s]}$ consider as input to S_2 when the value of $R_{11[s]}$ in hex and the number of y which equal to 8 as they index of table (2). The output is new sets of 8 bit and each set is saved in array called$R_{12[s]}$, where [s] is number of set.

R_{12mer}=R_{12mer}&$R_{12[s]}$, merge the sets of array which each with 8 bit into one set with y bit.

OS_2=R_{12mer}

1 or 2 or 3 or 4 or 5 or 6 or 7: if (y<8) then
begin
R_{13}=left(RS_1, R_{10}),cut R_{10}bits value from left of input
R_{14}= S_2(R_{13}, R_{10}),where $R_{10}$is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.
OS_2=R_{14}
end

```
    else
if (y>8) then
    begin
     R13=left(RS1, R1),cut  R10 bits value from left of input
     R14= S1(R13, R10),where R10 is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.
    R15=right(RS1,y- R10)
    RS12=R15
    R16[s]=RS12/8 , divide RS12 into sets of 8 bit and each set is saved in array called R16[s], ,where [s] is number of set.
    R17[s]= S2(R16[s],8)
    R17mer=R17mer&R17[s], merge the sets of array which each with 8 bit into one set with y bit.
    R18= R14& R17mer.
   OS2=R18
  end
end of case
```

Table 3 proposed S2

y	
1	0 1
2	1 3 2 0
3	4 0 1 3 6 5 2 7
4	D 4 2 0 6 5 A 1 8 9 F 3 C B E 7
5	1B 1F 18 7 19 1C 4 B 14 9 2 1 6 E 0 12 10 11 F 13 8 15 16 17 A C 1A D 5 1D 1E 3
6	0 11 32 3 4 5 6 25 8 9 A 3C C 2 12 1A 30 1D 27 13 36 38 2D 3E 1F E 2F 1B 1C 14 24 18 39 35 2E 23 B F 3A 7 28 29 D 2C 2B 16 1 2A 26 31 19 1E 34 21 22 37 15 20 10 3B 33 3D 17 3F
7	24 1 1B 3 1F B 6 7 21 6B 64 2 C 7D 23 F 72 11 41 4B 14 15 61 3F 18 43 49 40 6D 69 1E 5A 2B A 2D E 53 2E 26 27 28 29 7B 5D 2C 22 25 4C 42 31 4D 3A 7F 35 19 37 13 39 33 3B 5 7C 3E 3D 67 46 62 2A 6F 45 12 47 6E 1A 5B 8 7A 32 4E 4F 30 51 D 0 54 74 71 57 58 59 56 73 5C 20 5E 5F 60 75 48 17 38 65 66 79 68 4 4A 9 6C 1C 1D 36 70 6A 10 50 76 44 55 77 78 3C 2F 16 63 52 7E 34
8	0 30 73 B5 4 D 6 7 8 5C AB B 6D 5 DF D9 10 11 D8 13 E5 15 A 17 23 19 88 1B 1C 1D 47 27 A4 21 76 18 BA 25 32 1F 2D 3C 2A 3A 2C D4 2E F0 8D AD 9F 33 4A 35 36 37 29 39 62 9A 8C 3D 3E 3F 52 41 6A 7B 44 45 46 DA C4 49 B1 4B 4C AC 57 4F 50 38 40 59 E4 55 71 31 F5 53 5A 22 E7 74 7D 7C 8F 61 E C1 64 D7 77 63 68 FD B3 54 6C 6B 6E C0 E0 56 72 2 5D C6 69 ED 78 51 1A 95 5F 5E 7E 7F A1 81 A8 83 20 85 6F 87 EB 89 CF 96 E8 EE 8E 9B 90 91 92 93 94 FC 28 97 DC 2F 67 4D 9C A3 9E 3 D6 80 A2 9D 60 B0 E3 A7 82 A9 F6 16 D2 C2 AE 43 C5 3B B2 42 F9 F3 B6 B7 EF B9 86 BB BC A6 BE BF 24 65 98 C3 48 A5 DB C7 C8 C9 84 CB 70 FE F7 8A D0 D1 BD E2 CA D5 A0 26 14 7A 12 75 4E 58 DE B8 99 E1 D3 8B C 1E E6 2B CC E9 EA F EC 66 1 CE 34 79 F2 AA F4 DD 9 F1 F8 B4 FA FB AF 5B CD FF

2. $V(I-O)SBox_2$

This Sbox is depend on key O_i and also on α_{suf} so it is called key dependent Sbox and has non linear property. See figure (1) which show the work of this Sbox when the left of C_r, with y bit called α_{pre} is input of $V(I-O)Sbox_2$ which go to S_3 which its contain on table of bits. α_{pre} and its number of y bits are index of this table to out new bits with size y bit called OS_3. Then exclusive-OR (XOR) process is performed with the output of S_3 (OS_3), left bits of key O_i and the output of $V(I-O)Sbox_1$ (α_{suf}). Then the result, called RS_2, is the input of S_4 which it contains on table of bits and the input with its number of y bits are index of this table to out new bits with size y bit called OS_4. Then the β_{pre} takes the bits of OS_4 as output of $V(I-O)Sbox_2$.

2.1 S_3

This sbox can be written as a table with set of random elements is shown in table (4) and arranged by depending on number of bits (y) for α_{pre}. S_3 has value and y bit for input α_{pre} as they index of table and output new value with y bit called OS_3. See the following procedure that shows the work of S_3.

Procedure S_3 (α_{pre} as input, OS_3 as output)

R_{19}=y mod 8, where y is number of bits for α_{pre}
Case R_{19} of
0 : $R_{20[s]}=\alpha_{pre}/8$,divide the input into sets of 8 bit and each set is saved in array called $R_{20[s]}$,where [s] is number of set.
$R_{21[s]}= S_3(R_{20[s]},8)$, $R_{20[s]}$ consider as input to S_3 when the value of $R_{20[s]}$ in hex and the number of y which equal to 8 as they index of table (3). The output is new sets of 8 bit and each set is saved in array called$R_{21[s]}$, where [s] is number of set.
$R_{21mer}=R_{21mer}\&R_{21[s]}$, merge the sets of array which each with 8 bit into one set with y bit.
$OS_3=R_{21mer}$
1 or 2 or 3 or 4 or 5 or 6 or 7: if (y<8) then
begin
R_{22}=left(α_{pre}, R_{19}),cut R_{19}bits value from left of input
$R_{23}= S_3(R_{22}, R_{19})$,where R_{19} is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.
$OS_3=R_{23}$
end
else
if (y>8) then
begin
R_{22}=left(α_{pre}, R_{19}),cut R_{19} bits value from left of input
$R_{23}= S_3(R_{22}, R_{19})$,where R_{19} is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.
R_{24}=right(α_{pre},y- R_{19})
$\alpha_{pre2}=R_6$
$R_{25[s]}=\alpha_{pre2}/8$, divide α_{pre2} into sets of 8 bit and each set is saved in array called$R_{25[s]}$, ,where [s] is number of set.

$R_{26[s]}$= $S_3(R_{25[s]},8)$
R_{26mer}=R_{26mer}&$R_{26[s]}$, merge the sets of array which each with 8 bit into one set with y bit.
R_{27}= R_{23}& R_{26mer}.
OS_3=R_{27}
end
end of case

Table 4 proposed S3

Y	
1	1 0
2	0 3 2 1
3	0 4 6 7 5 1 2 3
4	0 C 8 5 F 2 1 7 3 9 E B 4 D A 6
5	4 1 2 3 10 11 14 9 A 7 E B C D 8 13 12 0 1F 15 6 F 16 18 5 19 1A 1C 1B 1D 1E 17
6	3A 3F 30 2D 4 5 0 1E 3C 6 16 B C D 23 F 10 11 9 E 36 15 7 26 18 19 1A 1B 1C 1D 31 1F 20 8 22 39 3E 25 3B 27 28 29 2A 2B 2C 17 2E 2F 2 21 13 33 34 3 35 37 14 38 12 24 3D 1 32 A
7	0 6 2F D 75 5 56 7 8 9 3 B C 39 E 3C 5E 1 12 13 14 3F 16 17 18 5C 1A 42 1C 1D 1E 1F 55 73 22 5D 6F 10 62 27 4A 33 54 2B 2C 2D 2E 2 30 60 28 29 35 34 19 15 24 47 F 3B 67 78 3E 7D 40 41 7B 48 4C 45 7A A 3A 77 69 4B 3D 61 70 4F 50 23 52 59 2A 37 32 57 58 53 5A 20 36 76 11 46 31 71 26 63 64 5F 66 1B 68 5B 6A 6B 38 51 6E 6C 49 7E 72 21 74 44 6D 25 4 79 65 43 7C 4E 4D 7F
8	3C E2 44 3 BB 1 A 28 E8 9 51 8C 3F C0 FC 50 A2 70 95 12 17 C3 B7 86 19 32 47 58 55 1D E0 8B DB D2 22 23 CD 56 26 27 5 4D 11 18 D6 2D 2E 2F 30 24 2B 33 82 9D CA 7E 38 FF 80 3B 14 3D 3E C DD BA 42 13 2 45 46 CE 48 AA 4A 8D 4C 29 F5 4F 2C 6 F3 53 FD 1C A3 57 E6 D4 EE 7D 5F F7 B B4 AC EF 99 60 64 65 E5 67 9B 69 BE 59 5E 6D 6E 6F 2A 98 54 73 61 A9 76 77 75 7C 8 7B 79 6B 37 7F 3A 81 F A7 84 85 0 B1 90 89 B2 10 8E 4B 6C F6 6A 91 92 F2 94 D3 96 97 71 62 9A 68 9C 49 9E 9F A0 D7 1F 66 AE A5 DE 83 A8 78 ED AB C5 AD FE AF B0 7 BD 35 8F B5 DF F8 B8 B9 41 4 E 8A F0 BF D C1 5D 39 C4 63 B3 C7 D5 C9 5A CB CC F4 52 CF D0 D1 21 BC 5B C6 34 A1 87 D9 DA 20 DC 40 A6 93 1E E1 D8 E3 5C 72 1B E7 7A E9 EA EB EC C8 36 74 B6 F1 31 88 1A 4E E4 C2 16 F9 FA FB 43 25 A4 15

2.2 S_4

This sbox can be written as a table with set of random elements is shown in table (5) and arranged by depending on number of bits for RS_2 . S_4 has value and y bit for input RS_2 as they index of table and output new value with y bit called OS_4. See the following procedure that shows the work of S_4.

Procedure S_4 (RS_2 as input, OS_4 as output)
R_{28}=y mod 8, where y is number of bits for RS_2
Case R_{28} of
0 : $R_{29[s]}$=RS_2/8 ,divide the input into sets of 8 bit and each set is saved in array called $R_{29[s]}$,where [s] is number of set.
$R_{30[s]}$= $S_4(R_{29[s]},8)$, $R_{29[s]}$ consider as input to S_4 when the value of $R_{29[s]}$ in hex and the number of y which equal to 8 as they index of table (4). The output is new sets of 8 bit and each set is saved in array called$R_{30[s]}$, where [s] is number of set.

R_{30mer}=R_{30mer}&$R_{30[s]}$, merge the sets of array which each with 8 bit into one set with y bit.

OS_4=R_{30mer}

1 or 2 or 3 or 4 or 5 or 6 or 7: if (y<8) then

begin

R_{31}=left(RS_2, R_{28}),cut R_{28}bits value from left of input

R_{32}= S_4(R_{31}, R_{28}),where R_{28} is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.

OS_4=R_{32}

end

else

if (y>8) then

begin

R_{31}=left(RS_2, R_{28}),cut R_{28}bits value from left of input

R_{32}= S_4(R_{31}, R_{28}),where $R_{28}$is equal to 1 or 2 or 3 or 4 or 5 or 6 or 7.

R_{33}=right(RS_2,y- R_{28})

RS_{22}=R_{33}

$R_{34[s]}$=RS_{22}/8 , divide RS_{22} into sets of 8 bit and each set is saved in array called $R_{34[s]}$, ,where [s] is number of set.

$R_{35[s]}$= S_4($R_{34[s]}$,8)

R_{35mer}=R_{35mer}&$R_{35[s]}$, merge the sets of array which each with 8 bit into one set with y bit.

R_{36}= R_{32}& R_{35mer}.

OS_4=R_{36}

end

end of case

Table 5 proposed S4

Y	
1	1 0
2	2 1 0 3
3	5 6 2 0 4 3 1 7
4	3 8 2 4 9 5 6 7 1 F A B 0 D E C
5	C 1 1A 3 6 11 E 7 8 9 A 1D 2 B 4 F 10 17 1B 13 16 15 0 1E 18 19 5 12 D 14 1C 1F
6	0 7 2F 37 24 3A 31 1E 16 9 A 3 E D 25 F 10 1 12 13 5 15 32 26 18 19 1A 1B 20 14 11 1F 1C 21 22 23 8 29 17 27 C 36 28 2B 2 6 2E 2C 30 39 B 33 4 35 2D 1D 38 34 2A 3B 3C 3D 3E 3F
7	5A 1 6D 7F 4 68 6 7 20 9 A 60 C D E 23 75 5F 32 1A 28 18 16 17 50 19 13 1B 29 55 5D 62 8 3F 22 58 2B 2C 2F 1E 39 72 3A 24 14 B 54 30 3B 2E 12 33 4D 70 36 37 38 7D 2A 53 3C 3D 0 74 4B 79 42 43 40 45 46 11 1F 49 27 F 4C 34 61 4F 15 51 52 2 73 1D 56 57 3E 59 25 5B 21 77 5E 47 2D 7B 6A 63 64 65 66 5C 67 69 48 6B 6C 44 6E 6F 35 71 1C 31 5 41 76 4A 78 4E 7A 10 7C 26 7E 3
8	0 61 2 3 4 1A 6 7 DD FE A 2B C 48 E F 10 6D 12 13 83 15 B1 B7 8E 19 5 1B 1C A0 9D 1F AE 58 33 23 24 6A 26 D3 64 99 DE AF 2C 2D 2E 9B DB 31 73 B4 51 A3 36 37 4B 39 3A 3B 9E 3D 8C 32 C1 CC D9 D7 44 45 46 47 7B 49 4A 25 96 CE 9F C9 50 B5 52 53 54 55 DC 2F 16 77 5A 5B 5C 3E C5 5F 30 F4 62 63 28 D5 65 D4 68 BA CA D BB 80 6E 6F 70 F6 35 8 74 27 EE C8 41 18 DF F8 F9 7D FB 7F CF 5D 82 76 84 86 85 87 88 89 8A 8B 9 F2 1 7C 56 91 92 AA 94 F5 5E 22 98 8D 9A B3 9C 97 66 40 FF A1 A2 C4 A4 A5 95 A7 A8 A9 42 AB C3 AD B6 E5 B0 C0 B2 43 38 A6 20 17 FC D1 CB 6C 1E BD BE BF CD 75 C2 34 EF AC C6 C7 71 4F 3F 69 78 21 1D 11 D0 B9 D2 81 67 29 D6 E2 D8 93 DA B8 90 3C 2A 7A E0 E1 B E3 57 72 14 E7 E8 E9 EA EB EC ED 4C E6 F0 F1 BC F3 79 6B 59 F7 4E 8F FA 7E FD 60 E4 4D

A. The general flowchart of proposed algorithm

In this section will introduce a general flowchart in figure (3) to describe the work of proposed algorithm.

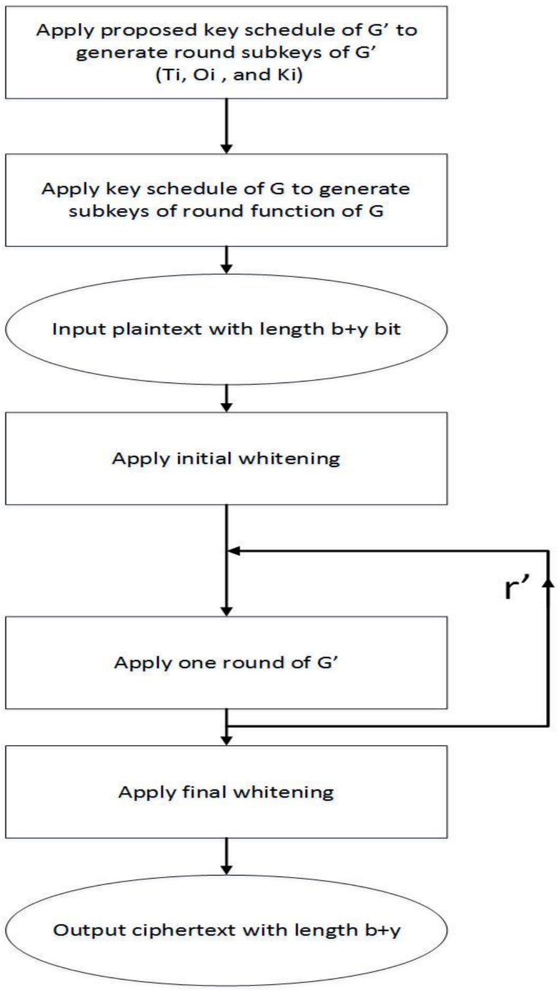

Fig. (3): A general flowchart of a proposed algorithm

B. The practical implementation of proposed algorithm

At first to generate two new variable input-output key dependent SBoxes by generate four fixed SBox S_1, S_2, S_3, S_4 where they have random and no iteration positions in case ($2 \leq y \leq 8$). Then insert these Sbox to a proposed algorithm which has a proposed key schedule. The result shows that an algorithm has a good structure, it has a high linearity complexity (6n) rather than (1n) where n is number of operations in one round, it has an increase in the speed, and it is more random compared with a previous elastic block cipher algorithm as shown in the following tables (5) and (6) of randomness and NIST tests respectively when the proposed algorithm has smaller values in the most results.

Table 5 The randomness tests

The tests	frequency	serial	Poker	autocorrelation	Run
Elastic algorithm [4]	0.40	1.60	2.05	0.25	21.69
Proposed elastic algorithm	0.98	**1.15**	3.67	**-1.25**	**11.44**

Table 6 NIST statistical tests

No.	NIST test	Elastic algorithm [4]	Proposed elastic algorithm
1	Frequency	0.5022	0.8025
2	Block frequency M=128	0.8818	0.9997
3	Runs	0.4775	**0.0169**
4	Longest Runs of Ones	0.3247	**0.0437**
5	Rank	0.3576	**0.1959**
6	FFT	0.2629	0.3018
7	NonOverlapping Templates m=9	0.5260	**0.4872**
8	Overlapping Templates all ones B='111111111'	0.0423	**0.0351**
9	Universal statistical L=6 & Q=1000	0.7864	**0.6532**
10	Linear complexity M=500	0.1842	0.4087
11	Serial m=12	0.3310	**0.0656**
		0.5001	0.6851
12	Approximate Entropy m=8	0.1365	**0.0715**
13	Cumulative Sums (Forward)	0.5219	0.9784

C. The conclusions

This paper proposes a new elastic block cipher algorithm with any network (substitution-permutation (SP) or Feistel) compared with the existing method that provides a Feistel-based variable length block cipher only.

A proposed algorithm allows us to "stretch" the supported block size up to double of the original block size with do not use plaintext padding process. This new structure has a good construction because it uses two proposed S-Boxes which depend on keys inside a one cycle rather than use XOR operation as in an existing elastic structure.

A number of round r′ in the new elastic algorithm is decreased into (r) in a Substitution Permutation (SP) network of exiting block cipher. And it equal to (r*2) in a balanced Feistel network rather than (r+[ry/b]) in existing elastic algorithm that cause to increase the speed of algorithm.

The complexity of a proposed round function of G′ is increased to (6n), where n is the number of operations in one round, with preserve on the speed at the same time because the round function of G′ use faster operations which consist of lookup in the table of S-Box and XOR operation. While in round function of G, the complexity is (1n) because it is use one operation (XOR operation). So the security of a new elastic structure and the randomness are increased when (y) value is increased forward to (b) bit while the existing elastic structure is vulnerable and has less security.

The existing elastic network has a weakness point when encrypt multiple blocks with using a fixed secret key that mean it has not well designed of key schedule. The new elastic network has good structure and good key schedule to prevent this weakness point in elastic network. A proposed key schedule has strong properties that derived from NPCBC mode.

References

[1] Cook, D., Yung, M., & Keromytis, A. (2004), " **Elastic Block Ciphers**",Columbia University.

[2] Patel, S., Ramazan, Z., & Sundaram, G. (2007)," **Constructions of variable input length cryptographic primitives for high efficiency and high security**", Patent Appliaction publication, pun. No. US20030191950.

[3] Bellare, M., Rogaway, P. (1999)," **On the Construction of Variable – Input length ciphers**", In: Knudsen L. (eds) Fast Software Encryption. FSE 1999. Lecture Notes in Computer Science, vol 1636. Springer, Berlin, Heidelberg.

[4] Cook, D., Yung, M., & Keromytis, A. (2007),"**Elastic Block ciphers:the basic design**", ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security.

[5] Cook, D., Yung, M., & Keromytis, A. (2004)," **Elastic Block ciphers: The feistel cipher case**", Columbia University.

[6] Zhang, L., Wu, W., Zhang, L., Li, Y. (2009), "**A note on Cook's elastic block cipher**", Conference: Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12.

[7] Lee, J., Koo, B., Roh, D., Kim, W. & Kwon, D. (2016),"**Appartus and method for providing feistel based variable length block cipher**". Patent Appliaction publication, pun. No. Us 2016/0056954 A1.

[8] Gu Dawu and Wang Yi, "**On the Techniques of Enhancing the Security of Block Ciphers**". ACM SIGOPS Operating Systems Review 35(4):94-96 · October 2001.

[9] Stallings, W. (2005)," **Cryptography and Network Security Principles and Practices, Fourth Edition**", Prentice Hall.

[10] Daemen, J., & Rijmen, V. (1999) "**AES Proposal: Rijndael**",Available:http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.36.