Proc. of the Second Intl. Conf. on Advances in Electronics and Electrical Engineering — AEEE 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-5939-1

New Cryptosystem Based on Lucifer with Controlled Key and Efficient S-boxes
(LKES)

H. Elkamchouchi, , Senior member IEEE [1]
[1]Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt

M.R.M. Rizk, Senior member IEEE [2]
[2]Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt

Fatma Ahmed, Member UACEE [3]
[3]Electrical Engineering Department, Alexandria Higher Institute of Engineering and Technology, Alexandria, Egypt.

***Abstract*—Networked computers are omnipresent, and are subject to attack, misuse, and abuse. One method to counteract the ever increasing cyber threat is to provide new symmetric algorithms satisfies the security requirements. In this paper we introduce new symmetric system based on Lucifer cryptosystem. The plaintext block is divided into basic sub-blocks each of thirty-two bits in length. The new Proposal can encrypt blocks of plaintext of length 256 bits into blocks of the same length. The key length is 512 bits. It uses new S-boxes implemented using Gold Exponent function one of APN (Almost Perfect Nonlinear) function. In this system, we try to get the minimum correlation between plaintext and ciphertext, highly avalanche effect and defeat the frequency analysis and most well-known attacks. The new algorithm is compared with Lucifer cryptosystem and gives excellent results from the viewpoint of the security characteristics and the statistics of the ciphertext. Also, we apply the randomness test to the proposed algorithm and the results shown that the new design passes all tests which proven its security.**

***Keywords*— Lucifer cryptographic system, APN, Gold exponent, frequency analysis.**

I. Introduction

Lucifer [1], a direct predecessor of the DES algorithm, is a block cipher having l28 bit block size and 128 bit key length. The message block to be enciphered is divided into two halves, the upper and lower, each containing eight bytes (64 bits). The bytes of the message are initially ordered so that the rightmost byte is the highest, and the leftmost byte is the lowest. Encryption (and decryption) is divided into sixteen rounds. During a round, the lower half of the message is transformed; the upper half is not changed, but its contents are used as input to the transformation. Between rounds, the upper and lower halves of the message are exchanged. The halves of the message move in step and rotate one position after each byte is used. The nonlinear transformations contain two different non-linear substitution boxes (S-boxes), S_0 and S_1. Each S-box has four input bits and four output bits. An S-box can be considered to implement a permutation of the numbers from 0 to 15. If the interchange control-bit is zero, then the right bits of the message byte are input to S_1 and the left bits are input to S_0. If the interchange control bit is one, then the bits input to the S-boxes are interchanged; the right bits are input to S_0 and the left bits are input to S_1. The generation of transformed bytes using the bytes of the upper half of the message as input to the key-controlled S-boxes is called confusion. It is then bitwise XORed (addition modulo 2) with the subkey. This process is called key interruption, since the use of the key acts as a barrier to cryptanalysis by merging some secret information into the confused bytes. The eight bits of each resulting interrupted byte are permuted according to a fixed Permutation. The permuted bits are then XORed with eight bits of the lower part of the message. The eight bits in the lower half of the message are chosen according to the bit pattern of convolution XOR cells. This process is called diffusion, since the result of the transformation of one half of the message is diffused throughout the other half of the message. The confusion, key interruption, and diffusion (c-i-d) Cycle described above forms a single round as shown in Fig.1.

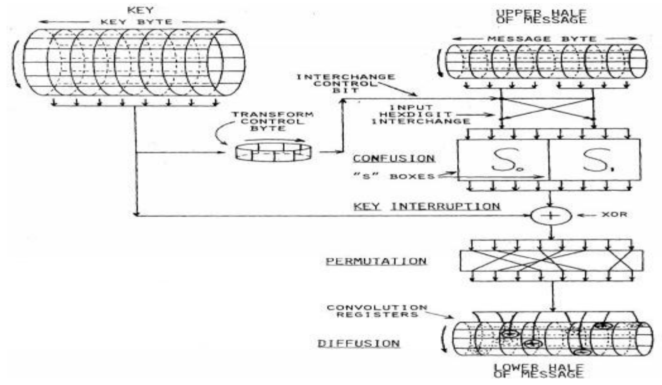

Figure 1. Block diagram of CID Lucifer

Table 1 shows the order in which the key bytes (rows) are accessed for each round. Each entry in the table is the number of the key byte to be accessed.

II. The New Proposed System

The new system is a block cipher; it can encrypt blocks of plaintext of length 256 bits into blocks of the same length. The key length is 512 bits. The total number of rounds is 16. In each round we work on the entire data block. In the new system we: propose new S-boxes using Gold exponent function, introduce row shift of data dependent on subkey of

Proc. of the Second Intl. Conf. on Advances in Electronics and Electrical Engineering — AEEE 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-5939-1

the round, permute each word in the block using fixed permutation which reduces the correlation coefficient as small as possible and in convolution step the two halves of a message depend on each other and we defeat the frequency analysis of ciphertext.

TABLE 1. KEY-BYTE ACCESS SCHEDULE

		Message Byte							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
C-D-I Round	**1**	0	1	2	3	4	5	6	7
	2	7	8	9	10	11	12	13	14
	3	14	15	0	1	2	3	4	5
	4	5	6	7	8	9	10	11	12
	5	12	13	14	15	0	1	2	3
	6	3	4	5	6	7	8	9	10
	7	10	11	12	13	14	15	0	1
	8	1	2	3	4	5	6	7	8
	9	8	9	10	11	12	13	14	15
	10	15	0	1	2	3	4	5	6
	11	6	7	8	9	10	11	12	13
	12	13	14	15	0	1	2	3	4
	13	4	5	6	7	8	9	10	11
	14	11	12	13	14	15	0	1	2
	15	2	3	4	5	6	7	8	9
	16	9	10	11	12	13	14	15	0

A. *Shift Row Step*

The LKES is cryptosystem aims to make the system controlled by key. The first step in LKES is shift row controlled by the user key. The plaintext block is 256 bits divided into eight rows each row is 32 bits. At the first step we merge the first two bytes of round subkey to produce inter-controlled key block with length 16 bits. Then, the inter-controlled key block divided into sub blocks (each one has a length of 2 bits). The value in each sub block is converted to a decimal number. We shift each row of data using the value in inter-controlled key sub block (the number of sub block is the number of row).

B. *The proposed S-boxes*

T In this paper we introduce new S-boxes implemented using the Gold exponent function. To understand the new S-boxes, we need first adopt the following definitions:

Definition 1: Let G_1 and G_2 be finite Abelian groups [2]. A mapping $F: G_1 \to G_2$ is called differentially δ-uniform if for all $\alpha \in G_1, \alpha \neq 0$ and $\beta \in G_2$

$$\left|\{z \in G_1 \mid F(z+\alpha) - F(z) = \beta\}\right| \le \delta \tag{1}$$

Differentially 2-uniform mapping are the almost perfect nonlinear permutations of $GF(2^n)$ as defined in [3].

Definition 2: Let the Gold function $F(x) = x^{2^k+1}$ $1 < k \le (n/2)$ be a power polynomial in $GF(2^n)$ and let $s = \gcd(k,n)$. Then F is differentially 2^s-uniform. If $\frac{n}{s}$ is odd, that is, F is a permutation.

LKES has two types of S-box. The first S-box uses the Gold function $F(x) = x^{2^5+1}$ in $GF(2^9)$ and the second S-box uses the Gold function $F(x) = x^{2^5+1}$ in $GF(2^7)$. These S-boxes provide permutation function since $\gcd(5,7) = \gcd(5,9) = 1$ and $\frac{n}{s}$ is odd, also F is differentially 2 -uniform. In the case of $GF(2^8)$ we can't satisfy these two conditions [4]. The S-boxes construction was attempt in the following way:

1- Initialize the S-box: The first column contains 0x00, 0x01,……, 0x0F. The second column contains 0x10, 0x11,… 0x1F etc., and so on. Thus, the value of the Byte at column y and row x is [xy] where $1 \le x \le 32$ for the first S-box and $1 \le x \le 8$ for the second S-box.

2- Map each byte in the S-box using the Gold function in the finite field $GF(2^9)$ for the first S-box and $GF(2^7)$ for the second S-box; the value {00} is mapped to itself.

3- Apply the following transformation to each bit of each byte in the S-box:

$$b_i^{'} = b_i \oplus c_i \tag{2}$$

Where c_i is the ith bit of byte c ; with the value {1F7};that is, $(c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$ for the first S-box or with the value {59};that is $(c_6, c_5, c_4, c_3, c_2, c_1, c_0)$ for the second S-box. The constant c was chosen so that the new S-boxes have no fixed points [S-box (a) = a] and no "opposite fixed points" [S-box (a) = $\bar{a}$], where $\bar{a}$ is the bitwise complement of a. Also this constant provides minimum correlation between input and output $\approx 8 \times 10^{-3}$.

The inverse S-boxes is constructed by applying the following steps:

1- Initialize the inverse S-box: The first column contains 0x00, 0x01,……, 0x0F. The second column contains 0x10, 0x11,… 0x1F etc., and so on. Thus, the value of the Byte at column y and row x is [xy] where $1 \le x \le 32$ for the first S-box and $1 \le x \le 8$ for the second S-box.

2- Apply the following transformation to each bit of each byte in the S-box:

$$b_i^{'} = b_i \oplus c_i \tag{3}$$

Where c_i is the ith bit of byte c ; with the value {1F7};that is, $(c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$ for the first S-box or with the value {59};that is $(c_6, c_5, c_4, c_3, c_2, c_1, c_0)$ for the second S-box.

3- Map each byte in the inverse S-box using the Gold function inverse equation in the finite field $GF(2^9)$ for the first S-box and $GF(2^7)$ for the second S-box; the value {00} is mapped to itself.

The inverse of Gold function: Let n be odd and $\gcd(n,k) = 1$. Then $F^{-1}(x) = x^l$ where:

$$l = \frac{2^{k(n+1)} - 1}{2^{2k} - 1} = \sum_{i=0}^{\frac{n-1}{2}} 2^{2ik} \bmod (2^n - 1) \tag{4}$$

Proc. of the Second Intl. Conf. on Advances in Electronics and Electrical Engineering — AEEE 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-5939-1

C. *Permutation*

The plaintext of LKES is 256 bits is divided into sub blocks each of 32 bits. The plaintext is XORed with the subkey after S-box step. Then every sub block is permuted by fixed permutation. This fixed permutation, given in table 2, was tested on a random sample of the messages, in each time gives the minimum correlation coefficient value between input and output data $\approx 0.05\times10^{-3}$.

TABLE 2 FIXED PERMUTATIONS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	20	12	25	21	29	28	27	13	3	8	16	6	10	18	7
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
24	11	17	5	22	2	4	31	19	9	23	1	26	15	0	30

D. *Convolution register*

In this step, LKES makes every bit in plaintext depends on the whole data block. At first, we divide the data in this step into two halves each one is 128 bit (16 byte). We update the right half bytes with the data from left half bytes and update the left half bytes with the data from the right half depends on number of rounds. We use by the following mathematica.9 sub-program:

```
$Left\_half = Partition[Left\_half, 32]$
$Right\_half = Partition[Right\_half, 32]$
$If\ [Mod[r,2] == 0,$
$For[i = 1, i \le 4, i++,$
$aux = (Leftbyte[1] <<< i+2) \oplus$
$(Leftbyte[2] <<< i+10) \oplus$
$(Leftbyte[3] <<< i+12) \oplus$
$(Leftbyte[4] <<< i+20);$
$Rightbyte[i] = aux \oplus Rightbyte[i]]],$
$For[i = 1, i \le 4, i++,$
$aux = (Rightbyte[1] <<< i+2) \oplus$
$(Rightbyte[2] <<< i+10) \oplus$
$(Rightbyte[3] <<< i+12) \oplus$
$(Rightbyte[4] <<< i+20);$
$Leftbyte[i] = aux \oplus Leftbyte[i]]]$
```

Each byte in the one half is rotated to the left. The amount of rotation tested to give us minimum correlation coefficient $\approx 0.8\times10^{-3}$. After this step, every byte in the one half is modified by using all bytes in the plaintext block. So the convolution register step provides high avalanche effect since every bit in it depends in whole plaintext data.

E. *Subkeys Generation*

The LKES key expansion algorithm takes as input a 16-word (64-byte) key and produces 128 words. This is sufficient to provide 8-word (256 bits) subkeys for each of the 16 rounds of the cipher. In subkey generation process, we try to have maximum avalanche effect between the user key and the ciphertext and to have minimum correlation coefficient. The subkeys are generated by first divide the user key into blocks each block with length 16 bits. To provide the maximum avalanche effect in subkeys we first XOR the first block with the second block in the user key. The resulting output is replaced with old content of first block. Then we update the key blocks from second block to 31th block by applying the following operation:

$$keyblock[i]_{new} = keyblock[i-1]_{old} \oplus keyblock[i+1]_{old} \qquad (5)$$

At final step we will be XORing the old content of first block with the updating 31th block in the key. The resulting output is replaced with old content of 32th block. Then we XOR the old content of first block with the updating 32th block in the key. The resulting output is replaced with old content of first block. In this step, each updating block is function of all blocks before it. So if we exchange only one bit in user key, most of output blocks will also exchange. To provide minimum correlation coefficient, we apply S-boxes to the pervious output array. Using S-boxes in subkeys generation make the subkeys resist the known cryptanalytic attacks. We repeat the S-boxes step eight times because we need 16 256-bit subkeys. After each time using S-boxes we rotate the user key by 54 bits and repeat the mix blocks before using S-box again.

F. *Encryption Process*

Our proposal is purely block cipher. The input plaintext length is 32 bytes. These bytes are divided into 8 sub-blocks each of 32 bits. The output of this system is also 8 sub-blocks arranged sequentially each of 32 bits. These sub-blocks are combined again to form 32 bytes blocks. The key length is 512 bits. This key is divided into 16 subkeys each of 32 bits.

Description of a Single Round

The input blocks of 256 bits are divided into 8 sub-blocks each of 32 bits. The first step we perform the shift row step. Then at the S-boxes step, the input data is divided into sub blocks each with length 16 bits. Each sub block entered the S-boxes will divide into two parts. First part with length 9 bits is encrypting using the S-box in $GF(2^9)$. These 9 bits represent the number of row (first 5 bits) and the number of column (last 4 bits). The second part with length 7 bits is encrypting using the S-box in $GF(2^7)$. These 7 bits represent the number of row (first 3 bits) and the number of column (last 4 bits). After all rounds, to resist the frequency analysis we perform the following step.

Shift data block

In this step we want to make sure that even for repeated data blocks, the ciphertext will not be repeated blocks. First we perform XORing between the first output byte from rounds and the first byte of subkey. The 8 bits output represent the number of bits rotating in the next input block. After the encryption process for second block we apply the same operation but using the second bytes of subkeys and so on. The overall structure of cipher is shown in figure 2.

III. Security Analysis

A. *Avalanche Effect*

In cryptography, the **avalanche effect** refers to a desirable property of cryptographic algorithms. The avalanche effect is

Proc. of the Second Intl. Conf. on Advances in Electronics and Electrical Engineering — AEEE 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-5939-1

evident when an input is changed slightly (for example, flipping a single bit) the output changes significantly (e.g. half the output bits flip). In the case of quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext. Constructing a cipher to exhibit a substantial avalanche effect is one of the primary design objectives. The avalanche effect is calculated as:

$$\text{Avalanche Effect}=\frac{\text{No. of flipped in the ciphered text}}{\text{No. of bits in the ciphered text}}\times 100\% \quad (6)$$

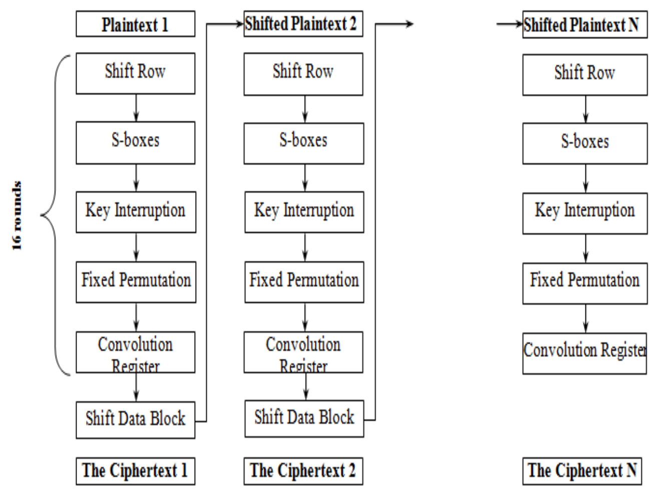

Figure 2 LKES overall structure

In our case, we take two plaintexts and two blocks of data flipping one bit from everyone in different positions and calculate the avalanche effect. Then we flip the user key in different positions and calculate the avalanche effect [5]. The following results are obtained after calculating the respective Avalanche Effects.

TABLE.3 AV EFFECT FOR 1 BIT CHANGE IN THE PLAINTEXT

Plaintext	Length of plaintext in bits	Change first bit in plaintext		Change last bit in plaintext		Change middle bit in plaintext	
		Lucifer	LKES	Lucifer	LKES	Lucifer	LKES
Case 1	158720	0.03%	29%	0.03%	0.08%	0.03%	12%
Case 2	135936	0.04%	17%	0.03%	0.09%	0.04%	25.1%
Case 3	1024	28%	49%	23%	50%	25%	54%
Case 4	1024	26%	51%	27%	52%	27%	52%

TABLE.4 AVALANCHE EFFECT FOR 1 BIT CHANGE IN THE USER KEY

Plaintext	Length of plaintext in bits	Change first bit in key		Change middle bit in key		Change last bit in key	
		Lucifer	LKES	Lucifer	LKES	Lucifer	LKES
Case 1	158720	49.8%	50.1%	49.8%	50.1%	49.8%	50.1%
Case 2	135936	50.%	50.1%	50.1%	53.3%	49.9%	50.2%
Case 3	1024	43%	52%	53%	55%	43%	53%
Case 4	1024	51%	51%	46%	50%	50%	50%

The avalanche effect of the proposed algorithm is producing very high as comparison Lucifer because in Lucifer if only one bit changes, it effects on its data block not all the blocks, while in LKES because we rotate the data block using the output ciphertext and the user key, so if one bit changes it produces different output.

B. *Secret Data Groups*

Considering the secret data used in Lucifer, the brute force attack for the key in the case of 128 bit block is $(2^{128}=3.4\times10^{38})$. The brute force attack for the data block in the case of 128 bit block is $(2^{128}=3.4\times10^{38})$. Considering the secret data used in LKES, the brute force attack for the key for 512 bits block is $2^{512}=1.34\times10^{154}$. The brute force attack for the data block for 256 bits block is $2^{256}=1.2\times10^{77}$.

C. *Language Statistics*

Language redundancy [6] is the greatest problem for any cryptosystem. The cryptanalyst uses the language redundancy to attack cryptosystems ciphertext. If the message is long enough, the cryptanalyst computes the frequency of each of the characters and consider different number of combinations up to the length of the cryptosystem block. The cryptanalyst will then try to estimate the plaintext from this statistical result. A cryptosystem is considered unbreakable against statistical analysis if its ciphertext has flat distribution. To implement the strength of new LKES, Figs 3&4 show the plaintext statistics of the used file. The ciphertext statistics of Lucifer and new LKES are plotted in Figs 5 to 8.

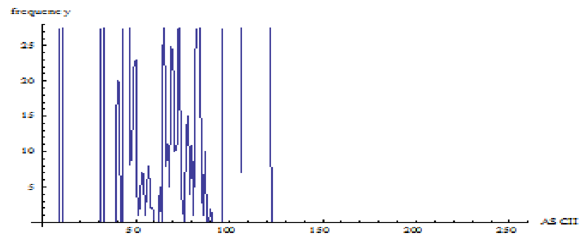

Figure 3 Plaintext statistics of a text file

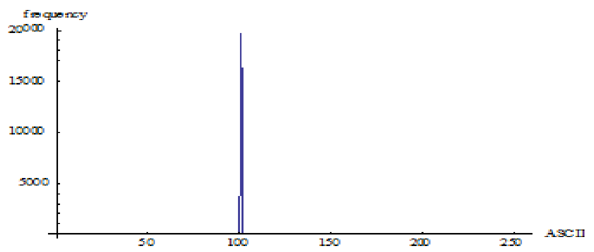

Figure.4 Plaintext statistics of repeated text file

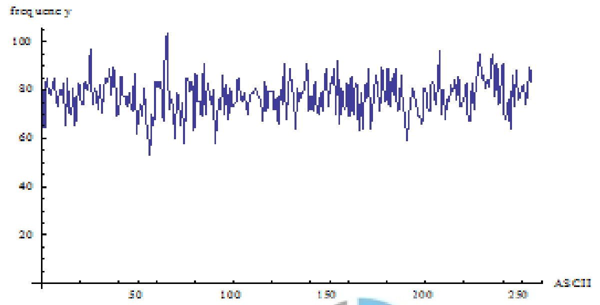

Figure 5. LKES ciphertext statistics

Proc. of the Second Intl. Conf. on Advances in Electronics and Electrical Engineering — AEEE 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-5939-1

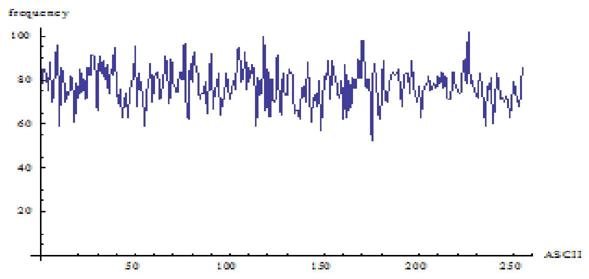

Figure.6 Lucifer ciphertext statistics

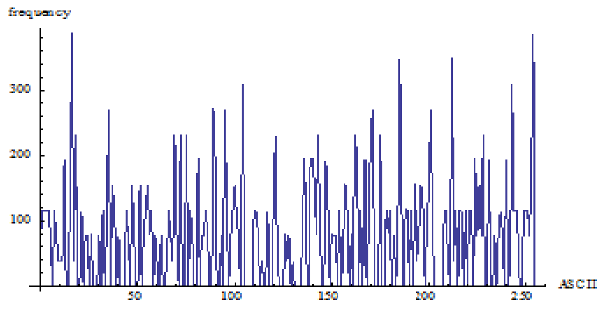

Figure.7 LKES ciphertext statistics of a message consisting of 20 Kbytes of character "e"

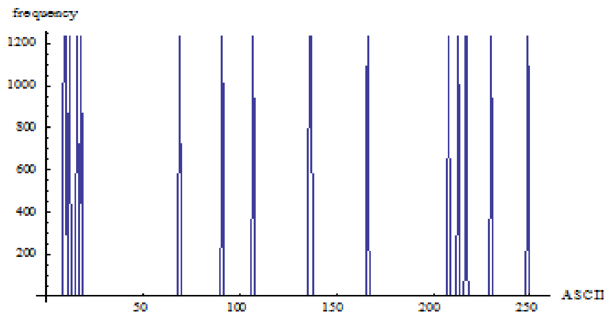

Figure.8 Lucifer ciphertext statistics of a message consisting of 20 Kbytes of character "e"

D. *NIST Statistical Suites*

The National Institute of Standards and Technology (NIST) [7] develops a Test Suite as a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based Cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The average values of the statistical tests for both algorithms were given in Table 5.

TABLE.5 LKES VS. LUCIFER STATISTICAL TESTS

Test name / **Algorithm**	LKES		Lucifer	
Frequency (Monobit) Test	100%	Pass	99%	Pass
Frequency Test within a Block	100%	Pass	97%	Failed
Runs Test	100%	Pass	99%	Pass
the Longest Run of Ones in a Block Test	100%	Pass	100%	Pass
Binary Matrix Rank Test	100%	Pass	100%	Pass
Discrete Fourier Transform Test	100%	Pass	100%	Pass
Non-overlap Template Matching Test	100%	Pass	100%	Pass
Overlap Template Matching Test	100%	Pass	100%	Pass
Maurer's "Universal Statistical" Test	100%	Pass	100%	Pass
Lempel-Ziv Compression Test	100%	Pass	100%	Pass
Linear Complexity Test	100%	Pass	99%	Pass
Serial Test	100%	Pass	99%	Pass
Approximate Entropy Test	100%	Pass	99%	Pass
Cumulative Sums (Cusum) Test	100%	Pass	99%	Pass
Random Excursions Test	99%	Pass	99%	Pass
Random Excursions Variant Test($\alpha = 0.05$)	98%	Pass	92%	Failed

IV. Conclusion

In this paper, we introduce a new cipher LKES based on Lucifer cryptosystem. We have improved the security of LKES by increasing the size of data block to 256 bits and the size of key to 512 bits. Also we introduce a new S-box based on one of APN functions "the Gold exponent function" in to generation fields $GF(2^7)$ and $GF(2^9)$. We use the S-boxes in the key expansion procedure to make it strong against the known attacks. In LKES system if we change a few bits in the plaintext or the user key it cause more than half of the ciphertext to be change. Finally, our proposal is rigid to withstand the well-known methods of brute-force.

References

[1] A. Sorkin, (1984). "LUCIFER: a cryptographic algorithm". Cryptologia, 8(1), 22–35, 1984.

[2] K. Nyberg, "Deferentially Uniform Mappings for Cryptography", Advances in Cryptology, Eurocrypt 93, Springer Verlag, 1994, vol. 765, pp. 55-64.

[3] K. Nyberg and L. R. Knudsen, "Provable Security Against Differential Cryptanalysis", Proceedings of Crypto '92 (to appear).

[4] C. Carlet. Vectorial (multi-output) Boolean Functions for Cryptography. Chapter to appear in Cambridge University Press. Preliminary version available at http://wwwrocq.inria.fr/codes/Claude.Carlet/pubs.html

[5] Amish Kumar, "effective implementation and avalanche effect of AES", International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, No 3/4, August 2012.

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

[7] NIST, "A Statistical Test Suite for Random and Pseudorandom Generators for Cryptographic Applications", NIST Special Publication 800-22, 2003.