مجلة

كلية التـــراث الجامعة

مجلة علمية محكمة

متعددة التخصصات نصف سنوية

العدد الثامن والثلاثون

15 شباط 2024

ISSN 2074-5621

رقم الايداع في دار الكتب والوثائق 719 لسنة 2011

مجلة كلية التراث الجامعة معترف بها من قبل وزارة التعليم العالي والبحث العلمي بكتابها المرقم (ب 3059/4) والمؤرخ في (2014/ 4/7)

New approach for Modified S-box for GOST

Dr.Ahmed Ghandour
Islamic University of Lebanon

Ali Hussein Ahmed
Islamic University of Lebanon

Dr.Saad Abdual Azize
Al_Mamon University Collage

Abstract

These days, it is much of the time tracked down an extensive variety of the getting data. The comprehension that the creator needs to configuration is about cryptography. Cryptography is the investigation of the security of information or data, regularly tracked down in moving information. One of them is the strategy for GOST, GOST is a shortening of "Gosudarstvennyi Standard" or "Government Standard." The calculation is straightforward encryption calculation which has a few cycles upwards of 32 adjusts and utilizes 64-digit block figure with 256-bit key. GOST technique likewise utilizes the S-Box 8 bits of long-lasting and XOR tasks and Pivot Left Shift, the proposed structure creates subkeys utilizing replacement of S-box powerfully and introductory table to gives the presentation of GOST. The move toward thinks about various security perspectives and measurements assessment for confirmation.

Keywords- Cryptography, GOST, Block ciphers, S-Box, Algorithm

INTRODUCTION

Cryptography or encryption may convert data or information from a regular, readable formula (original text) to an incoherent text that can't be read (encrypted text) using a specific algorithm and a secret key. Rather than erasing data, its purpose is to guarantee secrecy while protecting information from unauthorized access and change by third parties. Both symmetric key encryption and public key encryption are crucial encryption techniques [1].

In this paper, we will discuss the GOST algorithm, one of the various techniques that are used to encrypt data.

GOST is a Russian encryption technology that is also known as GOST Cipher or GOST Block Cipher. It is a symmetric key technique created by the Soviet Union in the 1970s as part of the encryption system and is used to safeguard data and secure communications in the military and government sectors.

The GOST algorithm is a block cipher, which means it runs on fixed-length plaintext blocks. As a result, its outputs are coded text blocks of the same length, with a 256-bit key when it operates on 64-bit data blocks. The method contains 32 cycles and employs a key switching and key mixing network.

The security of this algorithm is determined by the strength of the S-boxes, which are nonlinear substitution tables that provide the cryptographic confusion mode and are chosen in a way that has a high degree of nonlinearity that makes it resistant to cryptanalysis attacks, as well as the strength and complexity of the key used [2].

It was widely used in the Soviet Union and other eastern countries, but newer encryption algorithms such as AES (Advanced Encryption Standard), Twofish, and others have since supplanted it.

Many researchers have been interested in the issue of encryption throughout history.

BACKGROUND

GOST, an acronym for "Gosudarstvennyy Standard" or "Government Standard," is an acronym for a collection of regulations and standards used by the Soviet Union and other countries. The Soviet authorities set the GOST standards to assure interoperability, monotheism, and service quality. As a result, it currently encompasses a wide variety of disciplines, such as engineering, technology, industry, and quality control. These standards were needed at the time, but after the fall of the Soviet Union, they were subject to updates and alterations, and their obligation varied based on the nature of their usage in different countries. GOST is a soft algorithm that has 32 rounds. Feistel implements a 64-bit Block Cipher together with a 256-bit key, in addition to eight distinct S boxes, and divides the Block Cipher into two 32-bit portions called "R" and "L" [3].

GOST STRUCTURE

GOST structure such as:

1. Key 256-bit string 32-bit register (K1, K2, …, K8).
2. 32 Rounds.
3. Two parts of 32-bit register (R1, L1)
4. 32-bit add modulo 2^32
5. Add XOR
6. Switch block (S), 8 of 64-bit S-Boxes.
7. rotation shift Left register (R),11 bit.

A. Key Structure

The key structure for GOST (GOST 28147-89), a symmetric key encryption algorithm, employs a 256-bit key. The key's eight 32-bit subkeys are K1, K2, K3, K4, K5, K6, K7, and K8. The key structure technique is a method for establishing a password that may be used to encrypt plain text. K1, K2,..., K8 are used. The first 24 rounds, in particular, use K1, K2,..., K8 as ascending subkeys; "K1, K2, K3, K4, K5, K6, K7, and K8". Beginning with round 24, the key sequence is inverted: "K8, K7, K6, K5, K4, K3, K2, K1". This method, according to rounds [4], is as follows: [3][4][5]:

Round 0 - Round 7: K1, …, K8
Round 8 - Round 15: K1, …, K8
Round 16 - Round 21: K1, …, K8
Round 24 - Round 31: K8, …, K1

B. S-box

Four-bit inputs and four-bit outputs are supported by the S-boxes. Eight 4 4 S-boxes make up the S-box substitution in the round function. Parties who want to use GOST to encrypt their messages must use similar S-boxes, which are implementation-dependent. The S-boxes can be kept a secret for added security. S-boxes weren't provided in the initial standard where GOST

S 0:	4 10 9 2 13 8 0 14 6 11 1 12 7 15 5 3
S 1:	14 11 4 12 6 13 15 10 2 3 8 1 0 7 5 9
S 2:	5 8 1 13 10 3 4 2 14 15 12 7 6 0 9 11
S 3:	7 13 10 1 0 8 9 15 14 4 6 12 11 2 5 3
S 4:	6 12 7 1 5 15 13 8 4 10 9 14 0 3 11 2
S 5:	4 11 10 0 7 2 1 13 3 6 8 5 9 12 15 14
S 6:	13 11 4 1 3 15 5 9 0 10 14 7 6 8 2 12
S 7:	1 15 13 0 5 7 10 4 9 2 3 14 6 11 8 12

Table 1: GOST S-Boxes

was established, but they had to be provided in some other way. The Central Bank of the Russian Federation, for instance, employs the following S-boxes as shown in Table1:

RESULTS AND COMMENTS

1.Key Generator

This approach requires data input with a key range of 64 hexadecimal digits, 256 bits, or 32-character chunks. The following example exemplifies this procedure: will make use of the following Support key: "GOSTAlgorithmuseshideinformation" [3]:

No	range	Decimal	binary
1	G	47	1000111
2	O	79	1001111
3	S	83	1010011
4	T	84	1010100
. .	. .	. .	. .
32	n	110	1101110

The result of key generation process is 256-Bit as follows:

00101010110010101111001011100010111101101110011000110110100000100001011000101110100101100100111010100101101100111010101110101101100010011010010110000010110110011001100111011010010110101001101000011010110110010011101111011011101111011010010110001011110

The key will be categorized into eight parts.

K1:00101010110010101111001011100010

K2:11110110111001100011011010000010

K3:00010110001011101001011001001110

K4:10100110110011101010111010110110

K5:00100110100101100001011011001110

K6:01100110011101101001011010100110

K7:10000110101101100100111011110110

K8:11110110111101101001011000101110

2.Encryption Process

GOST encryption procedure of converting the input plain text data 64-bit or 16 hexadecimal digits or characters 8 through 32 rounds (iterations). Assume you choose the key on the form and the plain text "PASSWORD," then the encryption procedure will look like this: Figures 1, 2 and 3:

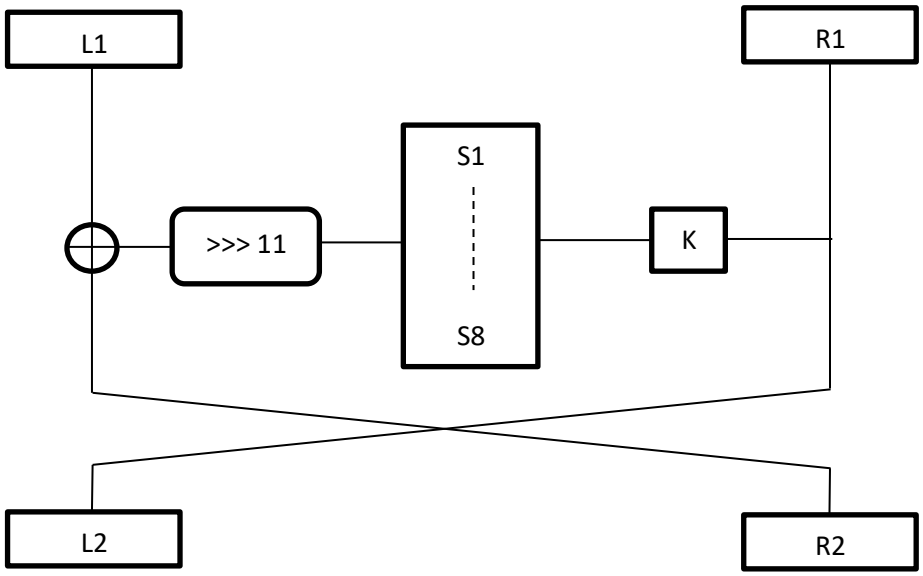
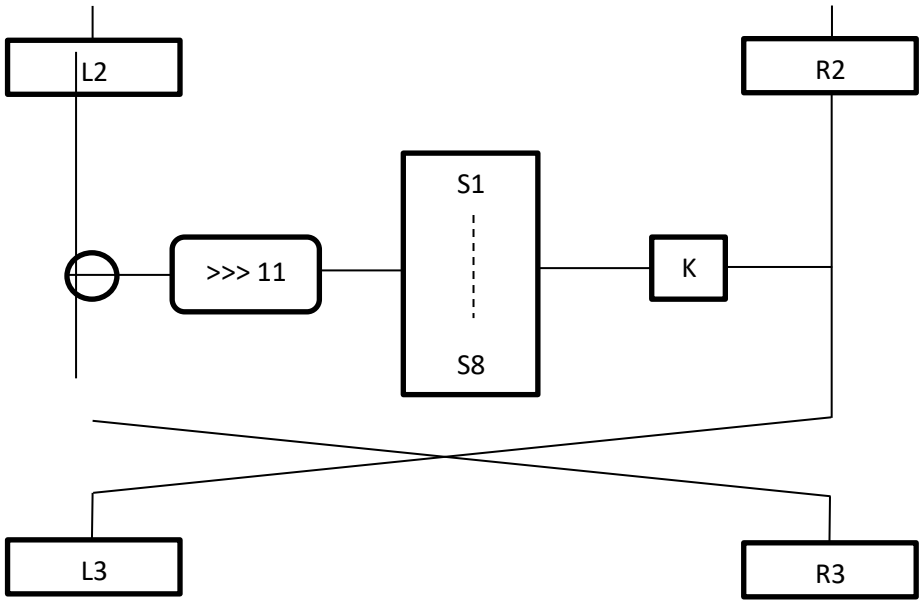
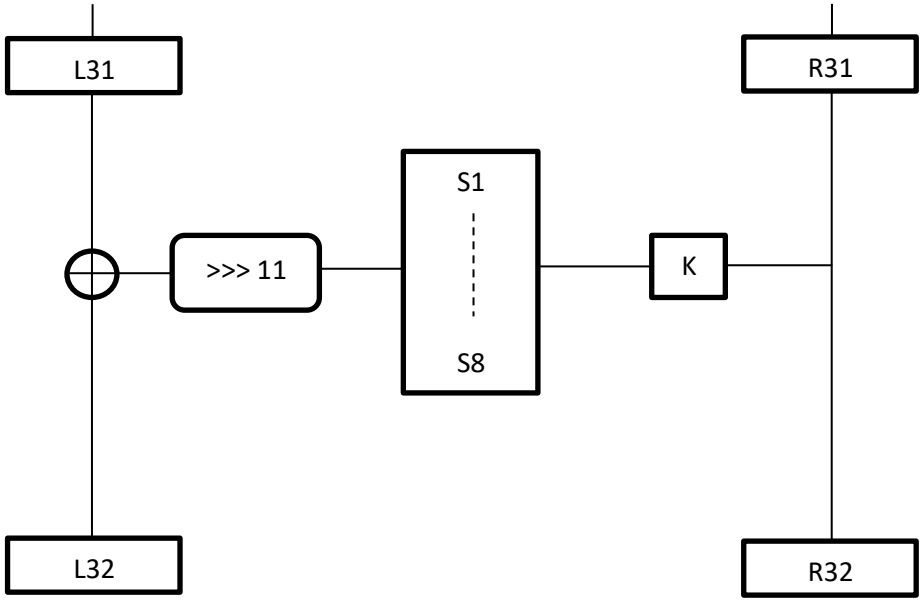


Figure1: Shown Encryption Round 0



2: Shown Encryption Round 1 Figure



3: Shown Encryption Round 32 Figure

1. **Decryption Process**

The decryption procedure is the inverse of the encryption operation. The GOST decryption method uses the same algorithm as the encryption method. Assuming that the key and ciphertext have been selected, the decoding process is as follows: Figures 4, 5, and 6:

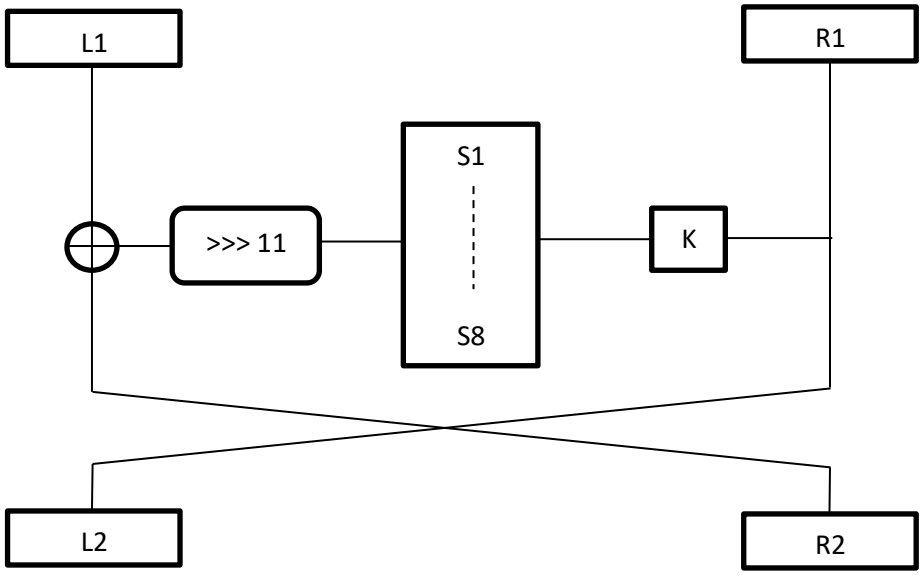


Figure 4: Shown Decryption Round 0

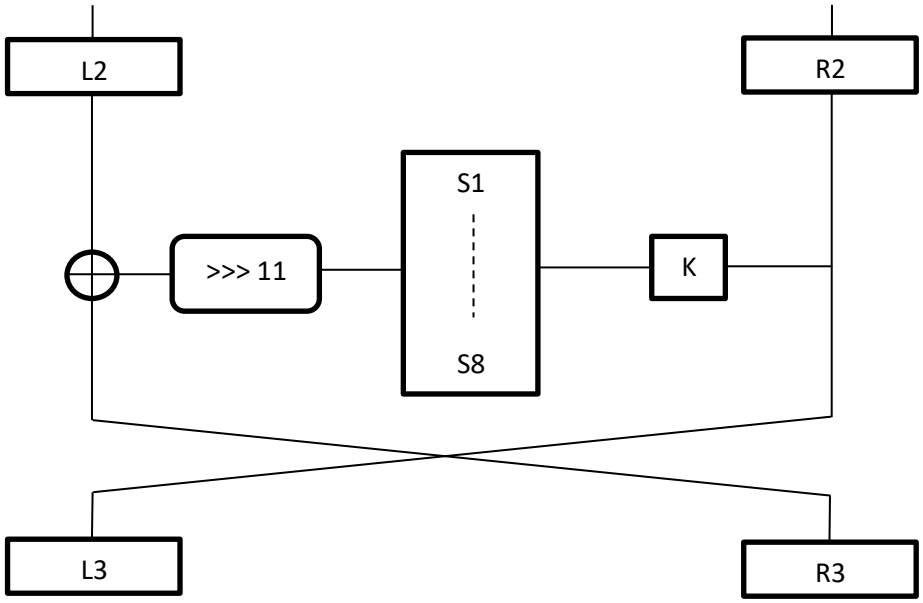
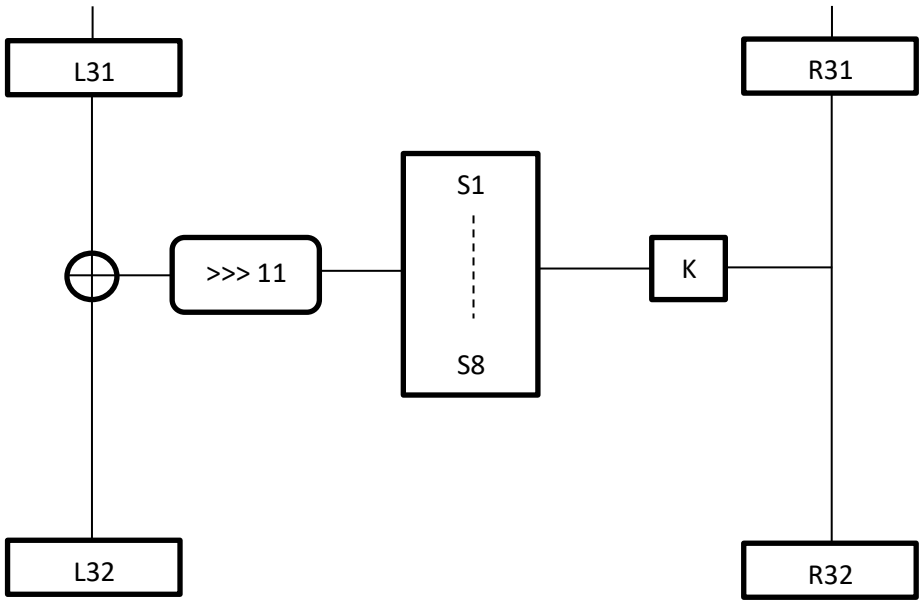


Figure 5: Shown Decryption Round 2



6: Shown Decryption Round 32 Figure

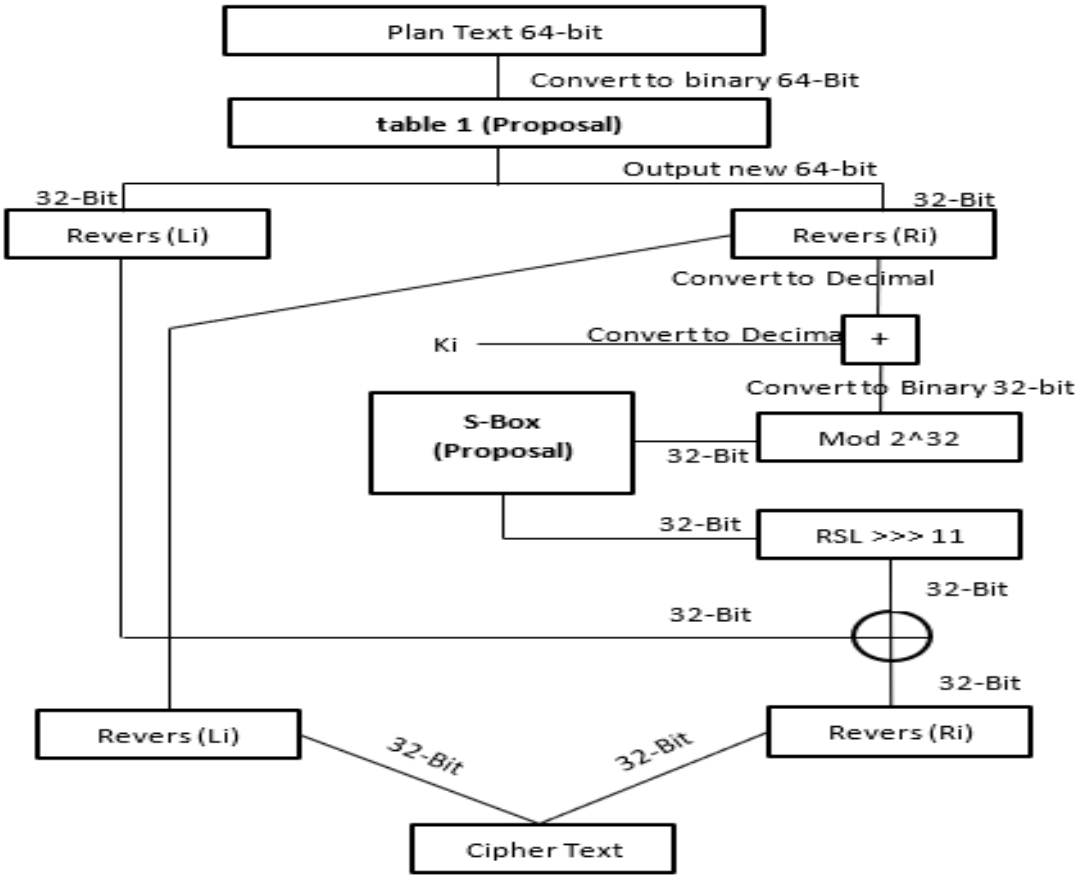


Figure 7: Encryption round proposal

PROPOSAL

We will discuss two proposed modifications to the GOST algorithm that try to make it harder to defeat and stronger against the attacks it is vulnerable to in this work. The first suggestion would be to establish a permutation table, which will be generated using a special algorithm as stated in Algorithm 2, and is referred to as initial permutation table 1 in this paper, with regard to the second suggestion, we will modify the S-box so that it has the same dimensions as the S-box standard established by the Central Bank of the Russian Federation and indicated in Table 1. We will explore this proposal in more detail later in this paper.

Algorithm (1)

-To generate random number
Input:
a,b,xo :initial value
output : generate random number between 0 to 15
begin:
step1: read initial value
step2: loop from 1 to 30

calculate the equation of random number x1=(ax0 +b) mod m
check the output between 0 to 15 and store in array
swap xo to x1
endloop
step3:change the location of array
swap location 2 with location 5
swap location 3 with location 7
swap location 4 with location 9
step4 :end

Algorithm (2)

-generate initial permutation table (8,8)
Input : a, b, xo :initial value
output : generate random number between 1 to 64
begin:
step1: read initial value
step2: loop from 1 to 30
calculate the equation of random number x1=(ax0 +b) mod m
check the output between 0 to 15 and store in column1
swap xo to x1
end loop
step3 :
loop from 2 to 8
column i = column (i-1) + 8
end loop

Whereas, by implementing Algorithm 1, we will create the initial permutation table1, which will be used in the encryption phase, in addition to generating its inverse, which is called the initial permutation table2, which will be used in the decryption phase. Tables 1 and 2 below show the generated tables.

table1

	0	1	2	3	4	5	6	7
0	18	10	2	26	34	42	50	58
1	20	12	4	28	36	44	52	60
2	22	14	6	30	38	46	54	62
3	24	16	8	32	40	48	56	64
4	17	9	1	25	33	41	49	57
5	19	11	3	27	35	43	51	59
6	21	13	5	29	37	45	53	61
7	23	15	7	31	39	47	55	63

Table 2: (initial permutation table1) used to Encryption process.

table2

	0	1	2	3	4	5	6	7
0	24	64	32	56	16	48	8	40
1	23	63	31	55	15	47	7	39
2	22	62	30	54	14	46	6	38
3	21	61	29	53	13	45	5	37
4	20	60	28	52	12	44	4	36
5	19	59	27	51	11	43	3	35
6	10	58	26	50	10	42	2	43
7	17	57	25	49	9	41	1	33

Table 3: (initial permutation table2) used to Decryption process.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
2	X15	X16	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
3	X13	X14	X15	X16	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
4	X11	X12	X13	X14	X15	X16	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
5	X9	X10	X11	X12	X13	X14	X15	X16	X1	X2	X3	X4	X5	X6	X7	X8
6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X1	X2	X3	X4	X5	X6
7	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X1	X2	X3	X4
8	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X1	X2

Table 4: structure of S-Boxes

The standard S-boxes will also be modified in this paper, so that the proposed S-boxes will be the same size as the standard S-boxes but in a different arrangement than the existing ones. Therefore, the proposed S-Box will be generated by Algorithm (1) above and used in the encryption and decryption rounds and will be modified as follows: The algorithm will produce the first row, which consists of non-repeating numbers from 0 to 15, and then the second row will be generated after by taking the last two numbers from the first row and placing them at the beginning, followed by the rest of the numbers of the first row in order to complete the second row. The third row follows the second row in the same manner, and so on until the table is complete. Therefore, the proposed S boxes will be as shown in Table 4.
From the arrangement of the Table 4 above, the S-Boxes will be in the following Table 5:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	4	6	8	10	12	14	1	3	5	7	9	11	13	15	2	0
2	2	0	4	6	8	10	12	14	1	3	5	7	9	11	13	15
3	13	15	2	0	4	6	8	10	12	14	1	3	5	7	9	11
4	9	11	13	15	2	0	4	6	8	10	12	14	1	3	5	7
5	5	7	9	11	13	15	2	0	4	6	8	10	12	14	1	3
6	1	3	5	7	9	11	13	15	2	0	4	6	8	10	12	14
7	12	14	1	3	5	7	9	11	13	15	2	0	4	6	8	10
8	8	10	12	14	1	3	5	7	9	11	13	15	2	0	4	6

Table 5: S-Boxes proposed.

IMPLEMENTATION

We will implement the GOST algorithm after making modifications to it:

- Support key: “usedGOSTAlgorithmprocessmodified”.
- Plain text: “COMPUTER”.

1.Key Generator Process

1.1. conversion the Support key to the binary key of the 256-bit is as follow:
01110101011100110110010101100100010001110100111101010011010101000100001011
01100011001110110111101110010011010010111010001101000011011010111000001110 0
10011011110110001101100101011100110111001101101101011011110110010001101001 0
11001100110100101100101011001 00

1.2. The key as above will be categorized into 8 parts.
K[0]:01110101011100110110010101100100
K[1]:01000111010011110101001101010100
K[2]:01000001011011000110011101101111
K[3]:01110010011010010111010001101000
K[4]:01101101011100000111001001101111
K[5]:01100011011001010111001101110011
K[6]:01101101011011110110010001101001
K[7]:01100110011010010110010101100100

2.Encryption process

The encryption process is as follows:

2.1. Encryption, round 0

2.1.1. Convert the Plain text to binary =
01000011010011110100110101010000010101010101010001000101010100 10

2.1.2. Enter the Binary of Plaintext as above to table (table1) as show output as follow:
11111111000111010110111011101010000000000000000001100000110000 01
L0 = 10000011000001100000000000000000
R0 = 01010111011101101011100011111111

2.1.3.R0 + K0 mod 2^32
R0 = 1467398399
K0 = 648466094
---------- +

R= 2115864493 mod 2^32 = 2115864493. convert to binary:
01111110000111011000011110101101

2.1.4. Split it into eight parts and put it into S-Boxes.

0111 = 7 = S-Box (1) = 3 = 0011
1110 = 14 = S-Box (2) = 13 = 1101
0001 = 1 = S-Box (3) = 15 = 1111
1101 = 13 = S-Box (4) = 3 = 0011
1000 = 8 = S-Box (5) = 4 = 0100
0111 = 7 = S-Box (6) = 15 = 1111
1010 = 10 = S-Box (7) = 2 = 0010
1101 = 13 = S-Box (8) = 0 = 0000
00111101111100110100111100100000

2.1.5. Rotate Left Shift 11 times.

RLS (11) = 10011010011110010000000111101111

2.1.6. R1 = R0 XOR L0

R0 = 10011010011110010000000111101111
L0 = 10000011000001100000000000000000
----------XOR
R1 = 00011001011111110000000111101111

2.1.7. L1 = R1

L1 = 01010111011101101011100011111111

2.2. Encryption, round 1

2.2.1. L1 = 01010111011101101011100011111111
R1 = 00011001011111110000000111101111

2.2.2. R1 + K1 mod 2^32

R1 = 427753967
K1 = 717943522
---------- +

R= 1145697489 mod 2^32 = 1145697489. convert to binary:
01000100010010011111010011010001

2.2.3. Split it into eight parts and put to S-Boxes.

0100 = 4 = S-Box (1) = 12 = 1100
0100 = 4 = S-Box (2) = 8 = 1000
0100 = 4 = S-Box (3) = 4 = 0100
1001 = 9 = S-Box (4) = 10 = 1010
1111 = 15 = S-Box (5) = 3 = 0011
0100 = 4 = S-Box (6) = 9 = 1001
1101 = 13 = S-Box (7) = 6 = 0110
0001 = 1 = S-Box (8) = 10 = 1010
11001000010010100011100101101010

2.2.4. Rotate Left Shift 11 times.

RLS (11) = 01010001110010110101011001000010

2.2.5. R2 = R1 XOR L1

R1 = 01010001110010110101011001000010

L1 = 01010111011101101011100011111111
----------XOR
R2 = 00000110101111011111011010111101

2.2.6. L2 = R1
L2 = 00011001011111110000000111101111

2.3. Encryption, round 31

2.3.1. L31 = 11010111010011111111100111011111
R31 = 11010000001111101010001010010100

2.3.2. R31 + K31 mod 2^32
R31 = 3493765780
K0 = 648466094
---------- +
4142231874 mod 2^32 = 4142231874 convert to binary:
11110110111001010111000101000010

2.3.3. Split it into eight parts and put it into S-Boxes:
1111 = 15 = S-Box (1) = 0 = 0000
0110 = 6 = S-Box (2) = 12 = 1100
1110 = 14 = S-Box (3) = 9 = 1001
0101 = 5 = S-Box (4) = 0 = 0000
0111 = 7 = S-Box (5) = 0 = 0000
0001 = 1 = S-Box (6) = 3 = 0011
0100 = 4 = S-Box (7) = 5 = 0101
0010 = 2 = S-Box (8) = 12 = 1100
00001100100100000000001101011100

2.3.4. Rotate Left Shift 11 times.
RLS (11) = 10000000000110101110000001100100

2.3.5.L32 = R31 XOR L31
R31 = 10000000000110101110000001100100
L31 = 11010111010011111111100111011111
----------XOR
L32 = 01010111010101010001100110111011

2.3.6. R32 = R31
R32 = 11010000001111101010001010010100

2.3.7. L32 = b32, b31, ... b1
R32 = a32, a31, ... a1
R= a1, ……., a32, b1, ……., b32
R in Binary =
11111111000111010110111011101010000000000000000000110000011000001
Convert to text:
Cipher Text = **)E|VT▌ ÿ¬Ω**

3.Decryption process

The decryption method is the opposite of the encryption method. GOST decryption method uses the same algorithm as the encryption method. Suppose selection up the key and ciphertext above, the decryption method is as follows:

3.1. Decryption, round 0

3.1.1. Cipher Text =)E|VT▌ ÿ¬Ω

Convert to Binary:

00101001010001010111100000010111101110110011000101010101101010

L0 = 01010111010101010001100110111011

R0 =11010000001111101010001010010100

3.1.2.R0 + K0 mod 2^32

R0 = 3493765780

K0 = 648466094

---------- +

4142231874 mod 2^32 = 4142231874convert to binary:

11110110111001010111000101000010

3.1.3. Split it into eight parts and put it into S-Boxes:

1111 = 15 = S-Box (1) = 0 = 0000

0110 = 6 = S-Box (2) = 12 = 1100

1110 = 14 = S-Box (3) = 9 = 1001

0101 = 5 = S-Box (4) = 0 = 0000

0111 = 7 = S-Box (5) = 0 = 0000

0001 = 1 = S-Box (6) = 3 = 0011

0100 = 4 = S-Box (7) = 5 = 0101

0010 = 2 = S-Box (8) = 12 = 1100

00001100100100000000001101011100

3.1.4. Rotate Left Shift 11 times.

RLS (11) = 10000000000110101110000001100100

3.1.5.R1 = R0 XOR L0

R0 = 10000000000110101110000001100100

L0 = 01010111010101010001100110111011

---------- XOR

R1 = 11010111010011111111100111011111

3.1.6.L1 = R0

L1 = 11010000001111101010001010010100

3.2. Decryption, round 1

3.2.1.L1 = 11010000001111101010001010010100

R1 = 11010111010011111111100111011111

3.2.2.R1 + K1 mod 2^32

R1 = 3612342751

K1 = 717943522

---------- +

4330286273 mod 2^32 = 35318977 convert to binary: 00000010000110101110110011000001

3.2.3. Split to eight part and put to S-Boxes:

0000 = 0 = S-Box (1) = 4 = 0100

0010 = 2 = S-Box (2) = 4 = 0100

0001 = 1 = S-Box (3) = 15 = 1111

1010 = 10 = S-Box (4) = 12 = 1100

1110 = 14 = S-Box (5) = 1 = 0001
1100 = 12 = S-Box (6) = 8 = 1000
1100 = 12 = S-Box (7) = 4 = 0100
0001 = 1 = S-Box (8) = 10 = 1010
01000100111111000001100001001010

3.2.4. Rotate Left Shift 11 times.
RLS (11) = 11100000110000100101001000100111

3.2.5. R2 = R1 XOR L1
R1 = 11100000110000100101001000100111
L1 = 11010000001111101010001010010100
---------- XOR

R2 = 00110000111111001111000010110011

3.2.6. L2 = R1
L2 = 11010111010011111111100111011111

3.3. Decryption, round 31

3.3.1. L31 = 00011001011111110000000111101111
R31 = 01010111011101101011100011111111

3.3.2. R31 + K0 mod 2^32
R31 = 1467398399
K0 = 648466094
---------- +
2115864493 mod 2^32 = 2115864493 convert to binary:
01111110000111011000011110101101

3.3.3. Split to eight part and put to S-Boxes
0111 = 7 = S-Box (1) = 3 = 0011
1110 = 14 = S-Box (2) = 13 = 1101
0001 = 1 = S-Box (3) = 15 = 1111
1101 = 13 = S-Box (4) = 3 = 0011
1000 = 8 = S-Box (5) = 4 = 0100
0111 = 7 = S-Box (6) = 15 = 1111
1010 = 10 = S-Box (7) = 2 = 0010
1101 = 13 = S-Box (8) = 0 = 0000
00111101111100110100111100100000

3.3.4. Rotate Left Shift 11 times.
RLS (11) = 10011010011110010000000111101111

3.3.5. L32 = R31 XOR L31
R31 = 10011010011110010000000111101111
L31 = 00011001011111110000000111101111
---------- XOR
L32 = 10000011000001100000000000000000

3.3.6. R32 = R31
R32 = 01010111011101101011100011111111

3.3.7. L32 = b32, b31, ... b1

R32 = a32, a31, ... a1
R= a1, ……., a32, b1, ……., b32
R in Binary =
11111111000111010110111011101010000000000000000001100000011000001

3.3.8. Enter the R as above to Table (table2) to show output as follow:
01000011010011110100110101010000010101010101000100010101010010
Convert to text:
Plain Text = **COMPUTER**

CONCLUSION

Since the S-boxes are not specified in the original standard, we presented a new version of GOST that employs a new, cryptographically robust S-box. We also talked about how to choose an appropriate method for choosing S-boxes. To achieve the best outcomes, an S-box must be designed with the size and time characteristics minimized. The boxes were built in the form of a moving table of values, where in each round we get new and different values, and this change gives the time to break the encryption. A table was also added when entering the plain text to change the locations of the entered values.

Reference

[1]Paar,C.,&Pelzl,J.(2010).Understanding Cryptography:ATextbook for Students and Practitioners.
Springer Science & Business Media.
[2]Schneier, B. (1996). Applied Cryptography (2nd ed.). John Wiley & Sons.
[3]Muhammad Iqbal1,Yudi Sahputra2,Andysah Putera Utama Siahaan3The Understanding of GOST
Crytography Technique.
[4] Nicolas T. Courtois Security Evaluation of GOST 28147-89In View Of International Standardisation
[5] Poschmann, A., Ling, S., & Wang, H. (2010). 256 bit standardized crypto for 650 GE – GOST
revisited. Lecture Notes in Computer Science, 6225, 219-233