Modified S-box to Archive Accelerated GOST

George S Oreku1† Jianzhong Li2 † Tamara Pazynyuk3 † and Fredrick J. Mtenzi 4††,

†Department of Computer Science and Engineering
Harbin Institute of Technology, Foreign Student Building
A13 Room 601, P.O.Box 773, 92 Xi Dazhi Street,
Nangang District, Harbin 150001 China
††Dublin Institute of Technology, Dublin 8, Ireland

Summary
The block cipher GOST was proposed in former Soviet Union in 1989. In this paper we present the principal algorithm of GOST block with reduced number of rounds. By introducing the idea reduced round version of GOST the modules speed processing of using a set of differential characteristics, we can simply increase the effectiveness and estimate the efficient for cryptosystem. We describe GOST and DES general principal encryptions algorithms for comparison .We present reduced GOST principals cryptographic basic steps and flowcharts for blocks information development. Simple modules were created and speed processing comparison for GOST and DES was made. The speed processing conducted in different machines classes gave good results for modified GOST.

Key words:
GOST, DES, Algorithms, Cipher.

1. Introduction

Since the introduction of public key cryptography in the 70's [13], many cryptosystems have been proposed and many cryptographic schemes have been broken. The most used cryptosystems rely on number theory problem like the factorization problem [14] and the discrete logarithm over suitable group [2]. The McEliece cryptosystem [15] and the Neiderreiter variante [16] rely on coding theory, they are ones of the few cryptosystems, which are very secure and which are not broken although they do not rely on number theory. These cryptosystems present many advantages: they are very fast for both encryption and decryption and the best attacks complexity are exponential in the length of the code. These cryptosystems have the drawback to have a large public key which is a generator matrix or a check parity matrix of a long code. Another drawback related to the belief that we can not deduce a digital signature from these public key cryptosystems.

In 1989 former Soviet Union proposed the block cipher GOST [1]. GOST is an acronym for "Gosudarstvennyi Standard", or Government Standard. GOST has key addition modulo 2^{32} in each round function.

Manuscript received June 5, 2007
Manuscript revised June 20, 2007

The block cipher GOST is based on the framework of the Feistel cipher. GOST has 32 rounds, 64-bit block size, and 256-bit key size. The *F*-function consists of operations specified as follows (see also Figure 1).
+: Addition modulo 2^{32}
S-boxes: 8 different 4 × 4-bit S-boxes $S_1, S_2 \ldots\ldots S_8$
<<< 11: 11-bit left rotation

Central Bank of the Russian Federation (tables of S-boxes are given in page 333 of [2], see also Appendix A.) Key-schedule is simple. The 256-bit master-key is divided to eight 32-bit blocks: $K_1, K_2 \ldots K_8$ each round uses the sub key as shown in Table below.

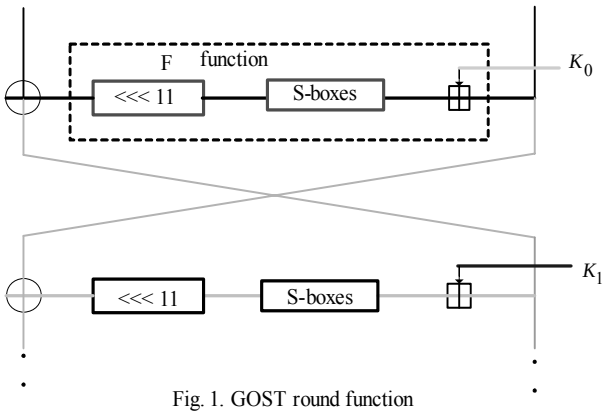

Fig. 1. GOST round function

The S-boxes are not specified in the standard. In this paper we use a set of S-boxes used in an application for the

Round	1	2	3	4	5	6	7	8	9	10	≈	15	16	17...
Key	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_1	K_2	≈	K_7	K_8	K_1...

18 ≈	23	24	25	26	27	28	29	30	31	32
K_2 ≈	K_7	K_8	K_8	K_7	K_6	K_5	K_4	K_3	K_2	K_1

We introduce a modified GOST performant encryption algorithms schemes based on modified S-box. The idea of our schemes is transformation function 64-bit block of data on algorithm of a basic cycle we call it *"X"* In this paper also we introduce and present the result of GOST

with reduced number of rounds. Next the analysis is expanded with reliability realizations experiment comparison with all GOST's encoding regime. We, define our modified S-box using flow chart simple replacement mode algorithm encoding data and simple replacement mode algorithm decoding data. We test the speed using modules construction principles program in "asm code". Unlike, the scheme based on the DES cryptosystem, our scheme has practical parameters and it is performant in terms of encoding key data and public key length speed is higher compared to DES. Before concluding, we present the graphical results performance measurement of speed modules of enciphering of this scheme in different machines.

2. Analysis of GOST and Key Generations

GOST, the Russian encryption standard [3] was published in 1989. [It was translated into English in 1993 and since then became well known to open cryptographic community.] It is a network of 32 round functions, add 32-bit sub-key module $2^{32,}$ put the results through layer of S-boxes and rotate that result left by 11 bits. The result is the out put of the round function [9]. Even after considerable amount of time and effort, no progress in cryptanalysis of the standard was made in the open literature except for a brief overview of a GOST structure in [4] and a related key attack in [5]. The GOST encryption algorithm is a block cipher with 256-bit keys and a 64-bit block length. GOST is designed as a 32-round Feistel network, with 32-bit round sub keys. See Figure 2 for a picture of one round of GOST.

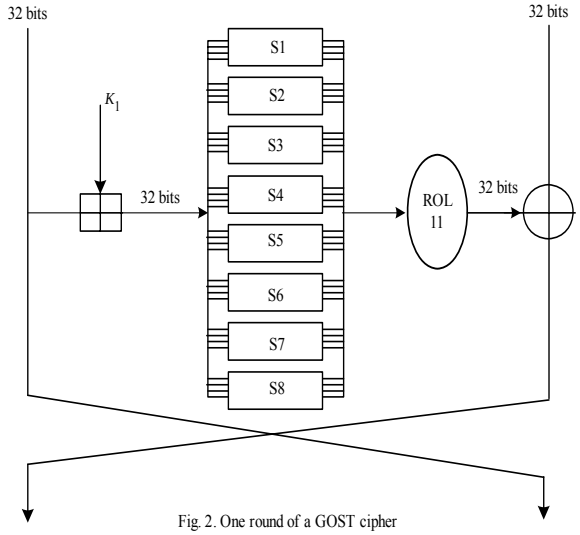

Fig. 2. One round of a GOST cipher

Table 1: Sub key's generation in GOST 28147

Round	1	2	3	4	5	6	7	8
sub key	1	2	3	4	5	6	7	8
Round	9	10	11	12	13	14	15	16
sub key	1	2	3	4	5	6	7	8
Round	17	18	19	20	21	22	23	24
sub key	1	2	3	4	5	6	7	8
Round	25	26	27	28	29	30	31	32
sub key	8	7	6	5	4	3	2	1

Key's generation is simple; 256-bits key is partitioned on the eight 32-bit sub keys. Algorithm has 32 rounds so each sub key uses at four rounds accordingly the scheme:
The GOST stability is formed by S-boxes structure. For long time S-boxes structure has not been published. Input and output of S-box are 4-bit numbers so each S-box can be presented by number's sequence from 0 to 15. Then sequence number will be input and output of S-box.

The key schedule divides the 256-bit key into eight 32-bit words $K_0,\ldots,K_7$ and then uses those key words in the order $K_0,\ldots K_7,K_0,\ldots K_7,K_0,\ldots K_7,K_6,\ldots K_0$. Notice the `twist' in the last 8 rounds.

3. Security argument

A huge number of rounds (32) and a well studied Feistel construction combined with Shannon's substitution-permutation sequence provide a solid basis for GOST's security. However, as in DES everything depends on the exact choice of the S-boxes and the key-schedule. This is where GOST conceptually differs from DES: the S-boxes are not specified in the standard and are left as a secondary key common to a network of computers [contrary to common belief, the standard does not even require the S-boxes to be permutations.]

The second mystery of GOST is its key-schedule. It is very simple and periodic with the period of eight rounds except for the last eight rounds where a twist happens. It is intriguing to find a reason for the twist in the last eight rounds of the key schedule. Moreover, in many applications we may wish to use shorter 64- or 128-bit keys, yet it is not clear how to extend these to a full 256-bit GOST key securely (fill the rest with zeros, copy the bits till they cover 256 bits, copy bits in a reversed order?).WHY THE TWIST? Consider a GOST cipher with a homogeneous key schedule, i.e., omitting the final twist (let us denote it GOST-H). Is this cipher less secure than GOST? We argue that, if one takes into account the slide attacks, it is. GOST-H can be decomposed into four identical transforms, each consisting of eight rounds of GOST. Furthermore, if one assumes that the round sub key is XORed instead of being added, the cipher will have 2^{128} weak keys of the form $< A;B;C;D;A;B;C;D >$ (here each letter represents a 32-bit GOST sub-key). These keys are weak since they allow for a *sliding with a twist* attack.

There is a known plaintext attack with 232 texts and time, and a chosen plain-text attack with 216 texts and time.

Notice that the 2^{128} keys of the form $< A; B; C; D; D; C; B; A >$ are also weak since GOST-H with these keys is an involution and thus double encryption will reveal the plaintext. Since these keys are invariant under a twist the same property holds for GOST itself. Also, there are 2^{32} fixed points for each key of this form, which demonstrates that there may be problems with using GOST to build a secure hash function.

THE ATTACK ON 20 ROUNDS OF GOST $\oplus$ .Suppose again that the round sub-key is XORed instead of being added, (we will denote this variant of GOST as GOST $\oplus$). Here we show an application of *sliding with a twist* which results in an attack on the last 20 rounds of GOST $\oplus$.

$$K_4K_5K_6K_7K_0K_1K_2K_3K_4K_5K_6K_7K_7K_6K_5K_4K_3K_2K_1K_0$$
$$K_0K_1K_2K_3K_4K_5K_6K_7K_7K_6K_5K_4K_3K_2K_1K_0K_7K_6K_5K_4$$

Let F denote 4 rounds of GOST $\oplus$ with key $K_4,\ldots,K_7$. With a pool of 2^{33} known texts, we expect to find two slid pairs, and each slid pair gives two in-put/output pairs for F. Breaking F with two known texts is straightforward, and can be performed in time comparable to about 2^9 evaluations of 4-round GOST (equivalent to 2^5 20-round trial encryptions). Thus in our attack we examine all 2^{65} text pairs; each pair suggests a value for 128 bits of key material, which we store in a hash table (or sorted list). The right key will be suggested twice, so we expect to be able to recognize it easily. By the birthday paradox, there will be only about two false matches, and they can be eliminated in the next phase.

Once we have recovered $K_4,\ldots,K_7$, it is easy to learn the rest of the key in a second analysis phase. For example, we can peel off the first four rounds and look for fixed points in the same pool of texts. Since the round sub keys are palindromic in the last sixteen rounds of GOST, there are 2^{32} fixed points, and each has the value $< x; x >$ before the last eight rounds of encryption. Thus, given a fixed point, we can try the 2^{32} values of $< x; x >$, encrypt forward and backward eight rounds, and obtain two candidate input/output pairs for 4 rounds of GOST $\oplus$ with key $K_0,\ldots,K_3$, so that a value for $K_0,\ldots,K_3$ is suggested after 2^5 work; then the suggested 256-bit key value is tried on another known text pair.

In all, this gives an attack on the last 20 rounds of GOST $\oplus$ that needs 2^{33} known texts, 2^{70} works, and 2^{65} space to recover the entire 256-bit key. Note that this attack is generic and works for any set of (known) S-boxes. The large memory requirements make the attack highly impractical, but we view it as a first step towards a better understanding of the GOST design.

4. DES Silent Features

DES (Data Encryption Standard) is a symmetric blocks cryptographic cipher based on beginning and ending bit's permutations. DES keys are 56 bits but its data elements are 64 bits likewise GOST key are bits and its data elements are 64 bits.

The main parameters of DES:

64-bit block; 64-bit key; 16 rounds; 48-bit key's element; 16 key's elements.

During each round of DES, the F-function is run on half of the 64 bits, expanding and permutation the 32 bits into 48 bits, then calculating the XOR of the bits with the key, substituting with S-boxes to get 32 bits, and finally performing another permutation. DES is still used in applications that do not require the strongest security.

Permutations implement according following formula $D[i]=S[[P]]$, *where*

S initial string
D permutation result
P table of permutations

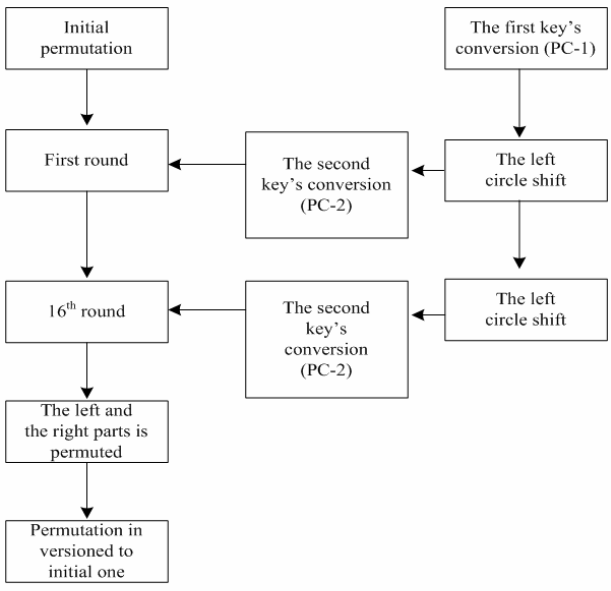

Fig.3. The DES common scheme

4.1 Existing Questions on DES Problems

There are 2^{56} possible keys length which are equal to 56 bits. These are not enough because frontal attacks can still happen. Triple DES, IDEA, AES Rijndael are the alternative to DES but questions to ask with no answer is can we use the cryptanalysis with existing DES

description [12]. Eight S-boxes are the base of DES in each iteration. In each iteration S-boxes are the basis of algorithm. There is a danger of hacker knowing a weak spots of S-boxes. Finding weak spots have been an outstanding questions for researchers with no answer till today. However blocks with the same data in DES (for example SPace - blocks) will have the same meaning in the cipher text. And that is wrong from crypto - analysis's point of view;

4.2 The main different between DES and GOST

- DES uses more difficult procedure of sub keys generation than GOST. In GOST this procedure is very simple.
- DES uses 56-bits key, GOST uses 256-bits key with that there is strong S-boxes in GOST
- DES has a 6-bit's inputs and 4-bit's outputs in S-boxes. GOST has 4-bit's inputs and outputs this makes GOST to have smaller S-boxes size than DES.
- In DES there are irregular P-permutations. In GOST there is 11-bits circle left shift. DES permutation increases avalanche effect. In GOST, modification of one input bit influence S-box in first round which affect two S-boxes in the next nested rounds etc. GOST needs eight rounds to influence each bit in result but DES needs 5 rounds only.
- DES has a 16 rounds, GOST has 32 rounds so GOST is more stable to differential and linear cryptanalysis.
- GOST keys are 256 bits with 64 bits data elements likewise DES keys are 56 with 64 bits data elements.

5. GOST Encryption Algorithm Analysis

If to study closely the original GOST 28147-89, it is possible to notice that, in its description of algorithms contains several levels. On the uppermost, there are practical algorithms intended for enciphering of data array insertions. All of them lean on three algorithms of the lowest level named in the text GOST by cycles [6]. These fundamental algorithms are mentioned in this paper as base cycles to distinguish them from all other cycles. They have following names and design:
A cycle encoded (32-Z);
A cycle decoded (32-P);
Creations of insertion circle 16-Z (16-Z)
See also section 6.2. for more details

Each base cycle presents repeated reoccurrence of the unique procedures named for definiteness further in the present work by the cryptographic basic step. So to understand the GOST, it is necessary to understand the following three concepts:

- What are Cryptographic *basic step*,
- From a *basic steps* how does *basic cycles* increases;
- From three *basic cycles* how does practical algorithms increases in GOST.

Before looking to above questions, it is necessary to look on the key information used by algorithms GOST. In the GOST the key information consists of two data structures .Besides *key* which is necessary for all codes, it has also the *table of replacements*. Below are the basic key characteristics structures for GOST.

1. *The key* is a file from eight 32-bit elements of a code, in the present work it is designated by a symbol **K**: $K = \{K_i\}_{0 \le i \le 7}$ In the GOST elements key used ere 32-bit integers without a sign $0 \le K_i \le 2^{32}$. Thus, the size of a key makes 32·8=256 bits or 32 byte.

2. *The table of replacements* is a matrix of *8×16* containing 4-bit elements which can be presented in the form of integers from 0 up to 15. Lines *of table of replacements* refer to as *units of replacements*, they should contain various values, which mean each unit of replacements should contain 16 various numbers from 0 up to 15 in any order. In present work the table of replacements is designated by symbol $\mathrm{H} : H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}$, $0 \le H_{i,j} \le 15$. Thus, the total amount of the table of replacements is equal: 8 units and 16 elements/units and $\times 4$ bit/element = 512 bits or 64 byte.

5.1 Key Management and Table Replacement in GOST.

The key should be statistically independent bytes, accepting with equal probability of value ***0*** and ***1***. Therefore the keys developed by means of some gauge of truly random numbers, will be qualitative with the probability differing unit on its insignificant small size. If keys are developed by means of the generator of pseudo-random numbers, the used generator should provide the statistical characteristics specified above, and, should possess high encryptions, not smaller, than at most GOST. In practice two criteria usually suffice, - for check of equally probable distribution bytes a key between values 0 and 1 [8] is used ("their square") and for check of independence bytes a key - criterion of series is usually used.

The table of replacements is a long-term key element that operates separate during long term. It is suppose to be the general key for all units of enciphering within the limits of one system of cryptographic protection. Even at infringement of confidentiality of the table of replacements stability of the code remains extremely high and does not decrease below an admissible limit. It is

possible to make quality of separate units of replacements resulted below the demand. Each unit of replacements can be described by the four logic functions, each of which has four logic arguments. It is necessary to make these functions complex enough.

6. Reduced round of GOST method Description

The description of the standard of data encoding contain in [6] is the general concept appears in our cryptographic transformation. Besides technological connections, several procedures of encoding are closely connected among themselves; the document describes constructions on general principles algorithm of information block development. It appears as a cryptographic control combination that is code developed from initial data with use of confidential key with protection of data against unauthorized changes.

On various steps of GOST algorithm, data in which it operates are interpreted and used by various image. In some cases elements of data are processed as files independent bytes, in other cases - as an integer without a sign, and sometimes as the complex element having structure consisting of several more simple elements. Therefore in order to avoid confusions it is necessary to have standard over used design.

Data elements in given subject are designated by header Latin letters with an inclined tracing (for example, ***X***). Through $|X|$ the size of an element of data X in byte is designed. Thus, if to interpret an element of data ***X*** as the whole non-negative number, it is possible to write down a following inequality: $0 \le X \le 2^{|X|}$.

If data element consists of several elements of smaller size this fact is designed as follows: $X = (X_0, X_1, \ldots, X_{n-1}) = X_0 \| X_1 \| \ldots X_{n-1}$. Procedure of association of several data elements in one refers to data concatenations and is designated by a symbol ||.Naturally, for the sizes of data elements the following parity should be carried out: $|X| = |X_0| + |X_1| + \ldots + |X_{n-1}|$ at the complex task of data elements and concatenations making data elements listed operation in ascending order seniorities. Differently, if to interpret a component and all data elements entering into it, as integers without a sign it is possible to write down following equality:

$$(X_0, X_1, ?\ X_{n-1}) = X_0 \| X_1 \| \text{数} \| \ n\text{-}1 \| = X_0 + 2^{|X_0|}\left(X_1 + 2^{|X_1|}\left(\text{数}\left(\ n\text{-}2 + 2^{|X_{n-2}|} X_{n-1}\right)\right)\right)$$

In algorithm the data element can be interpreted as a massive of separate bytes, in this case bits stand also for the file, but in a lower case variant, as shown in a following example which is designated by the same letter, as $X = (x_0, x_1, ?\ \ x_{n-1}) = x_0 + 2^1 ?\ \ _1 + ?\ \ \ 2^{n-1} ?\ \ _{n-1}$

If in the above data elements, some operations are having logic sense the given operation is carried out by above corresponding bits of elements differently $A.B = (a_0.b_0, a_1.b_1 ?\ \ a_{n-1}.b_{n-1})$ where $n = |A| = |B|$, and the symbol "**.**" This designates any binary logic operation; as a rule, means *excluding or*. It is operation of summation for module $2 : a \oplus b = (a + b) \bmod 2$.

6.1 Cryptographic Basic Steps and Flowcharts

The basic step of Cryptograph inherently in this presentation is the operator defining transformation of the 64-bit block of data. Additional parameter of this operator is the 32-bit block in which any element of a key is used. The scheme algorithm of the basic step is presented in figure 4. Below are explanations to algorithm basic step:

Step 0

N – Transformations, the 64-bit data block, during performance, its younger step (N_1) and senior (N_2) parts are processed as separate 32-bit integers without a sign. Thus, it is possible to write down $N = (N_1, N_2)$.

Step 1

Addition with a key the transformations of younger half the block develops on module 2^{32} with key used element on a step, and the result is transferred to a following step

Step 2

Block replacement. 32-bit value received on the previous step, is interpreted as a file from eight 4-bit blocks of a code: $S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$.

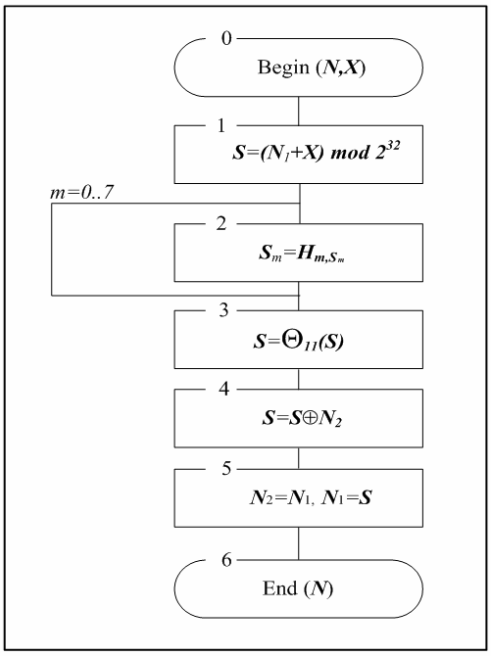

Fig. 4: Basic steps cryptographic algorithm GOST 28147-89. Scheme

Value of each eighth blocks is replaced on new get out under the table of replacements as follows: value of block S_i is replaced on S_{i-th} under the element order (numbering from zero) ***i***-that unit of replacements (i.e. $i-th$ is a line of table of replacements, numbering also starts from zero).Now there is clear size of the table of replacements: the numbers of lines in it are equal to number of 4-bit elements the 32-bit block given, that is eight, and number of columns is equal number of various values of the 4-bit block of the data, as 2^4, base 16th.

Step 3

Cyclic shifts on 11 bits to the left. The result of the previous step moves cyclically on 11 bits aside the senior categories and is transferred to a following step.

On the scheme of algorithm a symbol Θ_{11} means function of cyclic shift of the argument on 11 bits aside the senior categories is designed

Step 1. Bit-by-bit addition: the value received on a step 3 per bit, develops on the module 2 with the senior half transformed block.

Step 2. Shift on a chain: the younger part transformed block moves on a senior place and the result of performance of the previous step is located on its place.

Step 3. The received value transformed block comes back as result of performance of algorithm of the basic step crypt-transformation.

6.2 Cryptographic Transformation Basic Cycles

The algorithms are referred to as *basic cycle* in cryptographic transformations that emphasizes their fundamental value for construction of codes. Basic cycles are constructed from *basic steps* of cryptographic transformation. During performance of the basic step only GOST key contains such eight elements are used. Hence, the key has been used completely; each of base cycles should carry out repeatedly *basic step* with its various elements.

The above assumption, leaning simply on common sense, and seem to be true. Basic cycles consisted in repeated performance of *basic step* with use of different key elements differ from each other only by reoccurrence number of step and key elements use order. Below is the resulted order for various cycles:

1. *Encoded cycle 32-Z:*

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, \ldots$

$K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$

2. *Decoded cycles 32-P:*

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, \ldots$

$K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$

3. *Creations of insertion circle 16-Z*

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7$

The decryption cycle must be the reverse to encryption cycle, i.e. consecutive use these both cycles to arbitrary block must be given in a result of initial block. This can be seen in following correlation: $C_{32\text{-}P}\left(C_{32\text{-}Z}\left(T\right)\right)=T$, where **T** - any 64-bit data block, $C_X\left(T\right)$ - result of performance of cycle **X** above the data block **T**. Performance of this condition for the algorithms similar to GOST, it is necessary the key elements use order of corresponding cycles mutually returned. For considered case it is easy to be convinced validity of the written down condition, having compared results above sequence cycles for 32-C and 32-E. From that, for the cycle to be returned mutually to other cycle, that is cycle 32-Z there are must be a return relations to cycle 32-P, in this case decoded block of data should be executed by encoded cycle. From two mutually-reverse cycles any of them can be used for data encryption and another for data decryption, however standard GOST28147-89 fixes roles to cycles and does not give user option on this.

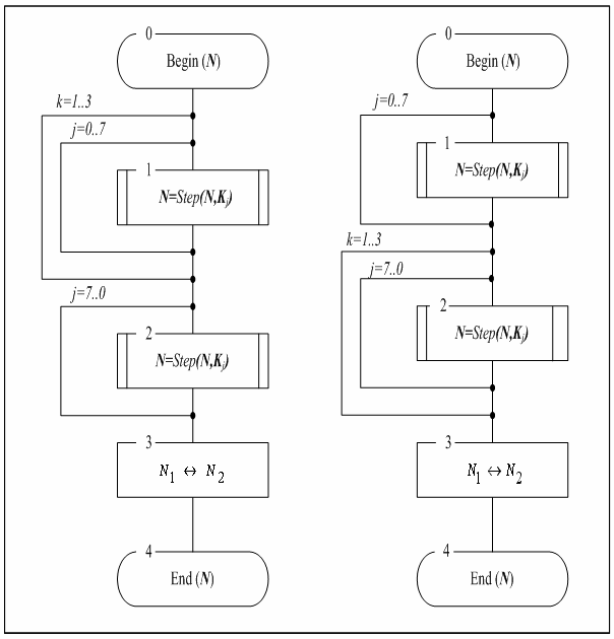

Fig.5. The schemes for a) encoded cycle 32-Z and b) decode cycle 32-P

The creations of circle insertions is twice shorter than encryption circle, the sequence of using key elements in creating insertions cycle, is the same as in the first 16 steps of a encryption cycle, considering the above sequence results , order in a designing cycle is coded by the same letter *"Z"*.

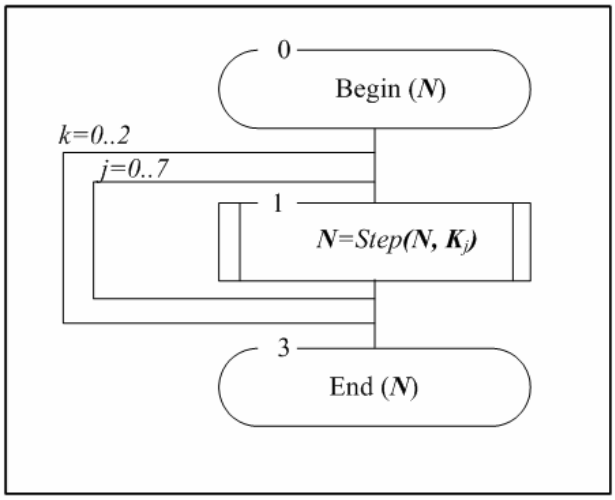

Fig.5c. The insertions scheme circle 16-Z

Schemes of basic cycles are shown in figures 5a-в. Each of them accepted as argument and returned as result 64-bit data block, which is shown as ***N*** on schemes. The step (N, X) means implementation of the main crypto-transformation step for block *N* by using dominant element ***X***.

Between encryption cycles and insertion calculation there is one more different not mentioned above, at the end of basic encryption cycles the first and the second part of result block changes places. It's necessary for its mutual reversibility.

6.3 The Basic Encryption Regimes

GOST 28147-89 provides three following modes of data cryptography: simple replacement, gammal, gammal feedback [10], and one additional creations insertion regime. In any of these modes data are processed by blocks on 64 bit into which the file is broken, subjected to cryptographic transformation, for this reason GOST is concerns to block codes. However in two modes of gammal there is an opportunity of processing the incomplete block given in the size less than 8 bytes that is essential or data encryption files with any size not multiple to 8 bytes.

Before considering specific crypto-transformation algorithm it's necessary to explain the design used in our schemes:

T_O, T_C –Files opening and ciphered data;

T_i^O, T_i^C – i-rd **i** under 64-bit blocks order opening and ciphered

data: $T_o = (T_1^o, T_2^o, ..., T_n^o)$, $T_c = (T_1^c, T_2^c, ..., T_n^c)$, $1 \le i \le n$, Last block can be incomplete:

n –Number of 64-bit blocks in data file;

C_X - Transformation function 64-bit block of data on algorithm of a basic cycle *"X";*

Descriptions of the above cryptographic basic modes will consider only one mode (Simple replacement) encryption using 32-Z cycle to disclosed data blocks in its regime the remaining are beyond the scope of this survey. Decryption using 32-P cycle to encrypted data blocks is the most simple regimes, 64-bit data blocks processing independently from each other. The schemes of encryption and decryption algorithm in the simple replacement regime are showed on the figure 6a and 6b, they are trivial and do not require comments.

The size of open file or encoded data, exposed accordingly or decoded, should be multiple to 64 bits: $|T_O| = |T_C| = 64n$, after operation performance the size of the received data file does not change. The mode of enciphering by simple replacement has following features:

1. Since data blocks encrypted independently from each other and from their positions in array, the two same disclose text blocks encryption gets the same blocks of code-text and vice versa. Noted property will allow crypto-analyst to make the conclusion about blocks identity of initial data in a ciphered data file to its

identical blocks that is inadmissible for the serious code.

2. If the length of data array for encoding is not multiple to 8-byte or 64-bit, arises the problem of what and how to supplement the last incomplete data block of array to complete 64-bit. Besides, the cipher text length will change, having increased up to the nearest whole, multiple 64 bits which often is undesirable.

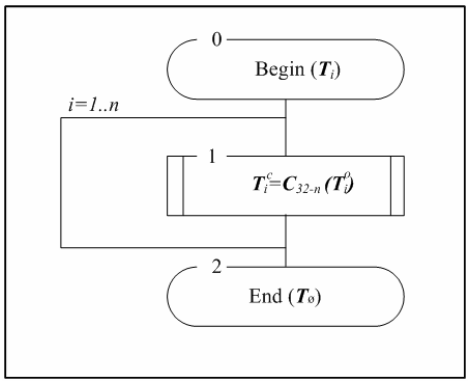

Fig. 6a. Simple replacement mode Algorithm encoding data.

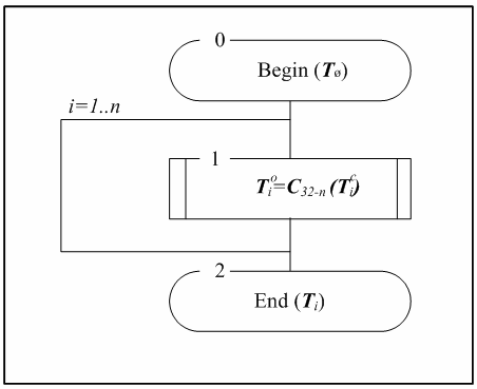

Fig. 6b. Simple replacement mode Algorithm decoding data.

The above features makes practically impossible using simple exchange regime i.e., in fact it can be applied only for data array encoding with multiple 64-bit's size which don't contain periodic 64-bit's blocks. It seems that for any real data to ensure implementations of these conditions is impossible, but there is one important exception; the key size is 32 bytes, and the table of replacements size – 64 bytes. Apart from existing periodic 8-byte blocks in a key or the table of replacements this will only rather show their bad quality, therefore while implementing such recurrence in real key elements will not happen. Thus, we elucidated that the simple exchange regime suites for dominant information encryption, especially other regimes are less comfortable for this goal, since they require additional synchronizing data element presence called synchro-premise. GOST orders uses simple replacement regime only for encoding key data.

7. Modules Construction Principles Program

1. All encoding and calculations insertions functions process (i.e. encoding and insertion calculations) areas with a size, multiple eight. Using these functions the processed area length gives 8 bytes blocks. With simple exchange encoding the size of ciphered area is suppose to be multiple eight bytes.

2. Cryptographic encoding functions and insertion calculations allow implementation of data arrays process by parts. Calling appropriate function once for several data area and this area consequently fragment function several times will give the same result.

3. For encoding and decoding data array in simple exchange regime GOST uses one function from two given operations implemented by given appropriate extended key. And the key's elements sequence should be mutually returned for given operations.

4. For encoding and decoding data block in gamma regime it uses one function i.e. in this regime data encoding and decoding are identical. Function realizations gamma encoding does not implement cyncro-packet transformations. It is necessary to impalement it initially during encoding functions call in simple exchange regime for synchro-packets since it is costly encoding array by parts.

5. For Assembler compilation and assembled given modules Borland - TASM 2.5 and Higher Version, Borland C/C ++ 2.0 resources were used. Probably in other means of development initial programs texts modification are required.

6 For illustrations file, using given crypto-functions added to this paper the crypto-program data file text in C language and project these files were used.
   - cryptor.c Initial texts for code files encoding
   - gost386.mak Project file for the 32-bit version of the program encoding files

8. GOST Implementations, Comparison and Performance Experiments Results

8.1 Overview Comparison of various block encryption schemes.

In this section we provide only a brief overview of the various block encryption schemes we have considered in this paper, namely, DES, IDEA, SAFER and GOST.

Further details can be found in many standard books on encryption such as [7, 17, 18].

A. DES

DES encrypts data in 64-bit blocks. A 64-bit block of plain text is taken as an input by the algorithm and a 64 bit block of cipher is produced as an output. DES is a symmetric algorithm [17]. Therefore, the computations performed by the encryption and decryption algorithms are nearly the same. The encryption key length is 56 bits long. The algorithm is a combination of two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of basic techniques on the plain text, based on the key. This is known as a round. DES has 16 rounds.

B. IDEA

IDEA operates on 64 bit plain text blocks. The key is 128 bits long. The same algorithm is used for both encryption and decryption. The design policy behind the algorithm is one of mixing operations from different algebraic groups. Three algebraic groups are being mixed; they are XOR, addition modulo 2_{16}, and multiplication modulo $2_{16}+1$. All these operations operate on 16 bit sub-blocks. In total IDEA has 8 rounds.

C. GOST

GOST is a 64-bit algorithm with a 256 bit key. The 256 bit key is divided into eight 32- bit blocks. Each round has a different sub key. The algorithm iterates a simple encryption algorithm for 32 rounds. There are eight different S-boxes in GOST [17]. Decryption is same as encryption. The GOST standard does not seem to discuss the S-box is generated. Therefore, including the secret S-box permutations, GOST has a total of about 610 bits of secret information.

D. SAFER

Figure 3 shows the throughput of these algorithms when encrypting in Electronic Code Book (ECB) mode. F or all these algorithms, the speed of decryption is approximately the same as the speed of encryption.

Table 3: Symmetric Ciphers: Long messages

Algorithm	Sparc (Mbit/s)	Alpha (MBit/s)
DEA-1	0.487	1.855
GOST 28147	0.713	2.909
SAFER-K64	2.259	7.680

8.2 Logical Code and Construction, Functions Description and Features Realizations

Taking into consideration presented principles, there are two GOST's realizations for Intel x86 processors [10], similar speed to possible optimum - for 16- and 32-bit processors accordingly. Code for 32-bit processors approximately is one and a half times faster than a corresponding code for 16-digit processors. In this survey a kernel is taken as subprogram which realizes universal basic GOST cycle. Initial texts of all subroutines are led as appendices to the present clause in separate files; they are listed in table 2. All functions are self-documentary; each of them described in a corresponding file with its initial text. The complete set of modules includes functions for the basic modes of cryptographic, and also two auxiliary functions intended for construction expanded key and table of replacements.

Table 3: The list of files

No.	Module Function	
1.	Universal GOST basic cycle	gost$.asm
2.	Gamma-regime functions of encoding and decoding.	simple$.asm
3.	Simple replacement mode function encoding data	gamma$.asm
4.	Function encoding data in gammal mode with a feedback	gammale$.asm
5.	Fuction decoding data in gammal mode with a feedback	gammald$.asm
6.	Data calculation function for file insertions	imito$.asm
7.	Expanded key construction function	expkey$.asm
8.	Expanded construction function (1 Kilobyte) replacement table forms from the usual (128 bytes) form	Expcht.asm
9.	Check processors function on which the appendix is executed, 32 – bit.	expkey$.asm
10.	Head file for Cryptographic functions programs use in C language	Gost.h

8.3 The Speed Processing Comparison for GOST and DES

After development of new program realization the speed processing was checked for this simple complex module created. This task fixed output on display time (time measure tact of frequency timer generator 1193180 hertz) for sub programme implementations testing. By measuring programme running time, then its speed processing was measured (manually) as division work amount to time implementations.

The maximal duration of process measured by the program is equal $2^{32}/1193180 \approx 3599.6$ seconds that is approximately to one hour. The program works correctly and yields correct results, only if loaded from DOS.

For GOST modules encoding durations 1megabyte data was measured which model by 32 encoding divisible of 32 kilobytes memory area . Measurements were conducted in different machine classes. Results of measurements are indicated in table 3. For 32 bits processors also speed processing for 32 bits crypto-modules realizations are indicated (the bottom number in a corresponding cell).

For comparison also conducted speed processing realization of the American standard of enciphering DES measurements which was published in magazine «Monitor No 7/1994». The test results showed that modules speed

processing for all GOST's encoding regime approximately the same and speed processing module calculations insertions approximately exceeds twice encoding processing speed as expected. Encoding realization by GOST essentially increases (twice) DES speed processing.

Then we estimated the achievement indexes from quality point of view. The limit encryption speeds increases more working speed of device encoding «Krypton–3» (up to 70 kilobytes/sec) and almost equal to device «Krypton-4» speed processing (about 400Kilobyte /sec). Achievement productivity for really transparent data encoding which kept on HDD or being transmitted from fast network is not enough Also realization productivity is quite enough for data encoding in commutated connection channels and for many other cases.

The questions whether it is possible to increase GOST speed realization the answer is, it is possible, but not very much if to remain within the limits of the formal GOST specification. For this purpose it is necessary to reject a cycle in the subprogram "gost", doubling cycle body 32 times as it was made by the program emulator author of a "Krypton" device. Thus it is possible not to develop a key in linear sequence elements but then for each base cycle of cryptographic transformation it is necessary to make the program module. Basic step code will be presented at codes of cryptographic procedures in 32+32+16=80 copies. Such way of efficiency incensement leads to code increase repeatedly to smaller productivity; therefore it is hardly to be considered good.

Table 4: Results of measurement of speed of modules of enciphering

Computer Mark	тact.freq	Speed of cryptographic modules					
Type of the processor	MHz	gamma	gammaLD	gammaLE	simple	Imito	DES
Spark 1031, K1810BM88	4.52	8.4	8.6	8.7	8.7	16.9	No data
AMI 286 Intel 80286	10	20.4	20.7	20.8	20.8	40.8	11.2
Prolinea 325 Intel 386SX-25	25	48.0 66.0	48.6 71.1	48.8 67.4	48.0 71.5	93.7 139	22.0
Unknown model Intel 386SX-33	33	63.8 87.6	64.5 94.5	64.7 89.5	63.8 95.0	124 185	25.9
BYTEX Intel 386DX-40	40	89 120	90 135	91 122	91 135	177 264	39.3
Acer Intel486SX33	33	114 150	113 161	114 151	114 162	226 321	41.2
Presario 460 Intel486SX2-66	66	225 298	222 319	229 303	227 324	451 637	82.2
Acer Pentium-66	66	302 351	296 397	307 355	293 405	601 777	88.7

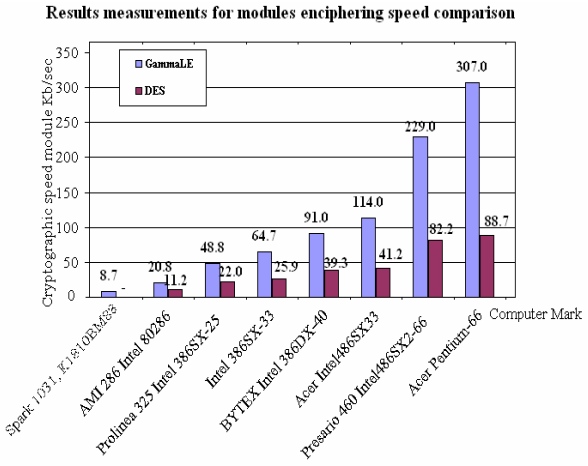

Fig 7: The performance speed comparison between DES and GOST ciphers in different PCs

From table three and figure 7 the results shows that the GOST based ciphers showed good performances. Gamma, Gamma LD and Gamma LE all performed higher when speed comparison was done with DES in encoding and decoding data due to GOST based ciphers having simple rounds functions for key generations in their modified S-box. Table three illustrates these results in more details.

8.4 Reliability Realizations of GOST Reduced-round version

For GOST algorithm system realization to be secured it is necessary to think about protocol elaboration. Two ways of using GOST algorithm with less main steps basic cycle's number is by decreasing key's length and using key element number. The main steps number in encoding basic cycles is $N=n \cdot m$, where **n** – number of 32-bits key's elements, **m** –number of cycles using key's elements, in standard $n=8,n=4$. Decreasing the main cycle's steps number and increasing code speed processing are almost the same.

Unfortunately, information about modification of the weak GOST cryptographic-stability is not there yet. At present many crypto-analysis try to work on sorting out the possible key's means i.e. key's size. This does not predict, how static line crypto-analysis will reduce complexity (analysis of modify data equation during their encoding).

In most practical cases used reduced GOST versions presented are without modification scheme key using $(m=4=3+1)$, but it is reasonable with key decrease by $4(n=2)$ as it allow increase of speed encryption almost by 4. By statistical crypto-analysis this stable methods modification with 64- key bits is more secure than DES with 56- key bits. Crypto-transformation functions, added to this paper also applies the same technique i.e. unfolded

key length passed as parameter in each from crypto-transformation subprograms, and "widening" key program allowing working with any key length and with widening key scheme.

In this case random bits stream can be used in gamma generator on any block cipher basic, including GOST 28147-89, as it possess necessary statistic descriptions and cryptographic stability. Thus, for development of several keys it is necessary to generate only data file on development scale algorithm, cut it for necessary size portion, for 32 bytes standard variant. An interesting future research direction would be to extend the models used in GOST for provable security

9. Conclusion

The discussions of GOST encryptions algorithm and modes of operations was done. GOST and DES implementations and processing speed comparison was done the results shows that modules speed processing for the reduced S-box for GOST and all GOST's encoding regime approximately the same and speed processing module calculations insertions approximately exceeds twice encoding processing speed as expected. Encoding realization by GOST essentially increases (twice) DES speed processing.

Compared to DES , GOST has a very simple round functions for key generations as its large number of rounds and secret S-box makes both liner 1 and differential cryptanalysis difficult.

Appendix A: S-Boxes

A set of S-boxes used in an application for the Central Bank of the Russian Federation are given in page 333 of [2].

$$S8 = \{1,15,13,0,5,7,10,4,9,2,3,14,6,11,8,12\}$$
$$S7 = \{13,11,4,1,3,15,5,9,0,10,14,7,6,8,2,12\}$$
$$S6 = \{4,11,10,0,7,2,1,13,3,6,8,5,9,12,15,14\}$$
$$S5 = \{6,12,7,1,5,15,13,8,4,10,9,14,0,3,11,2\}$$
$$S4 = \{7,13,10,1,0,8,9,15,14,4,6,12,11,2,5,3\}$$
$$S3 = \{5,8,1,13,10,3,4,2,14,15,12,7,6,0,9,11\}$$
$$S2 = \{14,11,4,12,6,13,15,10,2,3,8,1,0,7,5,9\}$$
$$S1 = \{4,10,9,2,13,8,0,14,6,11,1,12,7,15,5,3\}$$

References

[1] GOST, Gosudarstvennyi Standard 28147-89,"*Cryptographic Protection for Data Processing Systems"*, Government Committee of the USSR for Standards, 1989.

[2] B.Shneier,"*Applied Cryptography"*, John Wiley & Sons, pp. 331-334.

[3] I. A. Zabotin, G. P. Glazkov, V. B. Isaeva, *Cryptographic Protection for Information Processing Systems. Cryptographic Transformation Algorithm*, Government Standard of the USSR, GOST 28147-89, 1989. (Translated by A. Malchik, with editorial and typographic assistance of W. Diffie.)

[4] .C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini, Y. Zheng, *Comments on Soviet Encryption Algorithm*, proceedings of EUROCRYPT'94, LNCS 950, pp.433-438, Springer Verlag, 1994.

[5] J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, proceedings of CRYPTO'96, pp.237-251,Springer Verlag, 1996.

[6] Crypto Transformations Algorithm GOST 28147-89 Published in 1989 by the nationalSoviet bereu of Standard.]

[7] B.Schneier, *Applied Cryptography: protocols, algorithms, and source code in C*. John Wiley and Sons, 2002, vol. 2.

[8] Valeev S.G., Sergeys E.S. Algorithmic realization of the approach of dynamic regression modelling // Works of international conference «Methods and means of transformation and processing of the analog information». Ulyanovsk: USTU, 1999. V.3. p. 90-92.

[9] Nicolas T.Courtois "*Feistel Schemes and Bi-Linear Cryptanalysis*" Axalto smart Cards Crypto Research,long extended Version Of Crypto 2004

[10] ElGamal, T., 'A Public-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms', IEEE Transactions on Information Theory, July 1985.

[11] Markus Michels, David Naccache, and Holger Petersen "GOST 34.10 A Brief Overview of Russia's DSA" Published in Computers & Security 15(8):725-732, 1996.

[12] O.R Laponina "*Basic Network Security*": Cryptographic algorithms and protocol Interactions 1st editions Moscow, 2005 pp 25-70

[13] W.Deffie and M.E. Hellman, New directions incryptography, IEEE Transactions on information theory, vol 22,N°.6, pp.644-654 , 1976.

[14] R.L. Rivest, A. Shamir and L.M. Adleman, A method forobtaining digital signatures and public key cryptosystems,Communications of the ACM, Vol.21, No.2, pp.120- 126, 1978.

[15] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory; DSN Prog. Rep., Jet Propulsion Laboratory, California Inst. Technol., Pasadena, CA, pp. 114-116,January1978.

[16] H.Neiderreiter. Knapsack-type crytosystems and algebraic coding theory. Prob. Contr. Inform. Theory, 15(2):157-166, 1986.

[17] Satish Bapatla and R. Chandramouli "Battery Power Optimized Encryption" *IEEE Communications Society* 2004

[18] Nick Moldovyan and Alex Moldovyan *Innovative Cryptography* by (Paperback - Jun 14 2006) Publisher: Charles River Media; 2 edition (Jun 14 2006)