Independent Submission
Request for Comments: 7801
Category: Informational
ISSN: 2070-1721

V. Dolmatov, Ed.
Research Computer Center MSU
March 2016

GOST R 34.12-2015: Block Cipher "Kuznyechik"

Abstract

This document is intended to be a source of information about the Russian Federal standard GOST R 34.12-2015 describing the block cipher with a block length of n=128 bits and a key length of k=256 bits, which is also referred to as "Kuznyechik". This algorithm is one of the set of Russian cryptographic standard algorithms (called GOST algorithms).

Status of This Memo

Copyright Notice

Table of Contents

1. Scope 2
2. General Information 3
3. Definitions and Notations 3
   3.1. Definitions 3
   3.2. Notations 4
4. Parameter Values 6
   4.1. Nonlinear Bijection 6
   4.2. Linear Transformation 7
   4.3. Transformations 8
   4.4. Key Schedule 9
   4.5. Basic Encryption Algorithm 9
      4.5.1. Encryption 9
      4.5.2. Decryption 9
5. Examples (Informative) 10
   5.1. Transformation S 10
   5.2. Transformation R 10
   5.3. Transformation L 10
   5.4. Key Schedule 11
   5.5. Test Encryption 12
   5.6. Test Decryption 13
6. Security Considerations 13
7. References 14
   7.1. Normative References 14
   7.2. Informative References 14
Author’s Address 14

1. Scope

The Russian Federal standard [GOST3412-2015] specifies basic block ciphers used as cryptographic techniques for information processing and information protection including the provision of confidentiality, authenticity, and integrity of information during information transmission, processing, and storage in computer-aided systems.

The cryptographic algorithms specified in this standard are designed both for hardware and software implementation. They comply with modern cryptographic requirements and put no restrictions on the confidentiality level of the protected information.

The standard applies to development, operation, and modernization of the information systems of various purposes.

2. General Information

The block cipher "Kuznyechik" [GOST3412-2015] was developed by the Center for Information Protection and Special Communications of the Federal Security Service of the Russian Federation with participation of the Open Joint-Stock company "Information Technologies and Communication Systems" (InfoTeCS JSC). GOST R 34.12-2015 was approved and introduced by Decree #749 of the Federal Agency on Technical Regulating and Metrology on June 19, 2015.

Terms and concepts in the standard comply with the following international standards:

- ISO/IEC 10116 [ISO-IEC10116] and

- series of standards ISO/IEC 18033 [ISO-IEC18033-1] [ISO-IEC18033-3].

3. Definitions and Notations

The following terms and their corresponding definitions are used in the standard.

3.1. Definitions

Definitions

encryption algorithm: process that transforms plaintext into ciphertext (Section 2.19 of [ISO-IEC18033-1]),

decryption algorithm: process that transforms ciphertext into plaintext (Section 2.14 of [ISO-IEC18033-1]),

basic block cipher: block cipher that for a given key provides a single invertible mapping of the set of fixed-length plaintext blocks into ciphertext blocks of the same length,

block: string of bits of a defined length (Section 2.6 of [ISO-IEC18033-1]),

block cipher: symmetric encipherment system with the property that the encryption algorithm operates on a block of plaintext, i.e., a string of bits of a defined length, to yield a block of ciphertext (Section 2.7 of [ISO-IEC18033-1]),

Note: In GOST R 34.12-2015, it is established that the terms "block cipher" and "block encryption algorithm" are synonyms.

encryption: reversible transformation of data by a cryptographic algorithm to produce ciphertext, i.e., to hide the information content of the data (Section 2.18 of [ISO-IEC18033-1]),

round key: sequence of symbols that is calculated from the key and controls a transformation for one round of a block cipher,

key: sequence of symbols that controls the operation of a cryptographic transformation (e.g., encipherment and decipherment) (Section 2.21 of [ISO-IEC18033-1]),

Note: In GOST R 34.12-2015, the key must be a binary sequence.

plaintext: unencrypted information (Section 3.11 of [ISO-IEC10116]),

key schedule: calculation of round keys from the key,

decryption: reversal of a corresponding encipherment (Section 2.13 of [ISO-IEC18033-1]),

symmetric cryptographic technique: cryptographic technique that uses the same secret key for both the originator’s and the recipient’s transformation (Section 2.32 of [ISO-IEC18033-1]),

cipher: alternative term for encipherment system (Section 2.20 of [ISO-IEC18033-1]), and

ciphertext: data that has been transformed to hide its information content (Section 3.3 of [ISO-IEC10116]).

3.2. Notations

The following notations are used in the standard:

V*	the set of all binary vector strings of a finite length (hereinafter referred to as the strings) including the empty string,
V_s	the set of all binary strings of length s, where s is a non-negative integer; substrings and string components are enumerated from right to left starting from zero,
U[*]W	direct (Cartesian) product of two sets, U and W,
|A|	the number of components (the length) of a string A belonging to V* (if A is an empty string, then |A| = 0),

```
   A||B    concatenation of strings A and B both belonging to V*, i.e.,
           a string from V_(|A|+|B|), where the left substring from
           V_|A| is equal to A, and the right substring from V_|B| is
           equal to B,

   Z_(2^n) ring of residues modulo 2^n,

   Q       finite field GF(2)[x]/p(x), where p(x)=x^8+x^7+x^6+x+1
           belongs to GF(2)[x]; elements of field Q are represented by
           integers in such way that element
           z_0+z_1*theta+...+z_7*theta^7 belonging to Q corresponds to
           integer z_0+2*z_1+...+2^7*z_7, where z_i=0 or z_i=1,
           i=0,1,...,7 and theta denotes a residue class modulo p(x)
           containing x,

   (xor)   exclusive-or of the two binary strings of the same length,

   Vec_s: Z_(2^s) -> V_s  bijective mapping that maps an element from
           ring Z_(2^s) into its binary representation, i.e., for an
           element z of the ring Z_(2^s), represented by the residue z_0
           + (2*z_1) + ... + (2^(s-1)*z_(s-1)), where z_i in {0, 1}, i =
           0, ..., n-1, the equality Vec_s(z) = z_(s-1)||...||z_1||z_0
           holds,

   Int_s: V_s -> Z_(2^s)  the mapping inverse to the mapping Vec_s,
           i.e., Int_s = Vec_s^(-1),

   delta: V_8 -> Q  bijective mapping that maps a binary string from V_8
           into an element from field Q as follows: string
           z_7||...||z_1||z_0, where z_i in {0, 1}, i = 0, ..., 7,
           corresponds to the element z_0+(z_1*theta)+...+(z_7*theta^7)
           belonging to Z,

   nabla: Q -> V8  the mapping inverse to the mapping delta, i.e., delta
           = nabla^(-1),

   PS      composition of mappings, where the mapping S applies first,
           and

   P^s     composition of mappings P^(s-1) and P, where P^1=P.
```

4. Parameter Values

4.1. Nonlinear Bijection

The bijective nonlinear mapping is a substitution: Pi = (Vec_8)Pi’(Int_8): V_8 -> V_8, where Pi’: Z_(2^8) -> Z_(2^8). The values of the substitution Pi’ are specified below as an array Pi’ = (Pi’(0), Pi’(1), ... , Pi’(255)):

```
    Pi’ =
   (       252, 238, 221,  17, 207, 110,  49,  22, 251, 196, 250,
           218,  35, 197,   4,  77, 233, 119, 240, 219, 147,  46,
           153, 186,  23,  54, 241, 187,  20, 205,  95, 193, 249,
            24, 101,  90, 226,  92, 239,  33, 129,  28,  60,  66,
           139,   1, 142,  79,   5, 132,   2, 174, 227, 106, 143,
           160,   6,  11, 237, 152, 127, 212, 211,  31, 235,  52,
            44,  81, 234, 200,  72, 171, 242,  42, 104, 162, 253,
            58, 206, 204, 181, 112,  14,  86,   8,  12, 118,  18,
           191, 114,  19,  71, 156, 183,  93, 135,  21, 161, 150,
            41,  16, 123, 154, 199, 243, 145, 120, 111, 157, 158,
           178, 177,  50, 117,  25,  61, 255,  53, 138, 126, 109,
            84, 198, 128, 195, 189,  13,  87, 223, 245,  36, 169,
            62, 168,  67, 201, 215, 121, 214, 246, 124,  34, 185,
             3, 224,  15, 236, 222, 122, 148, 176, 188, 220, 232,
            40,  80,  78,  51,  10,  74, 167, 151,  96, 115,  30,
             0,  98,  68,  26, 184,  56, 130, 100, 159,  38,  65,
           173,  69,  70, 146,  39,  94,  85,  47, 140, 163, 165,
           125, 105, 213, 149,  59,   7,  88, 179,  64, 134, 172,
            29, 247,  48,  55, 107, 228, 136, 217, 231, 137, 225,
            27, 131,  73,  76,  63, 248, 254, 141,  83, 170, 144,
           202, 216, 133,  97,  32, 113, 103, 164,  45,  43,   9,
            91, 203, 155,  37, 208, 190, 229, 108,  82,  89, 166,
           116, 210, 230, 244, 180, 192, 209, 102, 175, 194,  57,
            75,  99, 182).
```

Pi^(-1) is the inverse of Pi; the values of the substitution Pi^(-1)' are specified below as an array Pi^(-1)' = (Pi^(-1)'(0), Pi^(-1)'(1), ... , Pi^(-1)'(255)):

```
 Pi^(-1)' =
(    165,  45,  50, 143,  14,  48,  56, 192,  84, 230, 158,
      57,  85, 126,  82, 145, 100,   3,  87,  90,  28,  96,
       7,  24,  33, 114, 168, 209,  41, 198, 164,  63, 224,
      39, 141,  12, 130, 234, 174, 180, 154,  99,  73, 229,
      66, 228,  21, 183, 200,   6, 112, 157,  65, 117,  25,
     201, 170, 252,  77, 191,  42, 115, 132, 213, 195, 175,
      43, 134, 167, 177, 178,  91,  70, 211, 159, 253, 212,
      15, 156,  47, 155,  67, 239, 217, 121, 182,  83, 127,
     193, 240,  35, 231,  37,  94, 181,  30, 162, 223, 166,
     254, 172,  34, 249, 226,  74, 188,  53, 202, 238, 120,
       5, 107,  81, 225,  89, 163, 242, 113,  86,  17, 106,
     137, 148, 101, 140, 187, 119,  60, 123,  40, 171, 210,
      49, 222, 196,  95, 204, 207, 118,  44, 184, 216,  46,
      54, 219, 105, 179,  20, 149, 190,  98, 161,  59,  22,
     102, 233,  92, 108, 109, 173,  55,  97,  75, 185, 227,
     186, 241, 160, 133, 131, 218,  71, 197, 176,  51, 250,
     150, 111, 110, 194, 246,  80, 255,  93, 169, 142,  23,
      27, 151, 125, 236,  88, 247,  31, 251, 124,   9,  13,
     122, 103,  69, 135, 220, 232,  79,  29,  78,   4, 235,
     248, 243,  62,  61, 189, 138, 136, 221, 205,  11,  19,
     152,   2, 147, 128, 144, 208,  36,  52, 203, 237, 244,
     206, 153,  16,  68,  64, 146,  58,   1,  38,  18,  26,
      72, 104, 245, 129, 139, 199, 214,  32,  10,   8,   0,
      76, 215, 116 ).
```

4.2. Linear Transformation

The linear transformation is denoted by l: (V_8)^16 -> V_8, and defined as:

```
l(a_15,...,a_0) = nabla(148*delta(a_15) + 32*delta(a_15) +
133*delta(a_13) + 16*delta(a_12) + 194*delta(a_11) +
192*delta(a_10) + 1*delta(a_9) + 251*delta(a_8) + 1*delta(a_7) +
192*delta(a_6) + 194*delta(a_5) + 16*delta(a_4) + 133*delta(a_3) +
32*delta(a_2) + 148*delta(a_1) +1*delta(a_0)),
```

for all a_i belonging to V_8, i = 0, 1, ..., 15, where the addition and multiplication operations are in the field Q, and constants are elements of the field as defined above.

4.3. Transformations

The following transformations are applicable for encryption and decryption algorithms:

```
X[x]:V_128->V_128  X[k](a)=k(xor)a, where k, a belong to V_128,

S:V_128-> V_128  S(a)=(a_15||...||a_0)=pi(a_15)||...||pi(a_0), where
   a_15||...||a_0 belongs to V_128, a_i belongs to V_8, i=0,1,...,15,

S^(-1):V_128-> V_128  the inverse transformation of S, which may be
   calculated, for example, as follows:
   S^(-1)(a_15||...||a_0)=pi^(-1) (a_15)||...||pi^(-1)(a_0), where
   a_15||...||a_0 belongs to V_128, a_i belongs to V_8, i=0,1,...,15,

R:V_128-> V_128  R(a_15||...||a_0)=l(a_15,...,a_0)||a_15||...||a_1,
   where a_15||...||a_0 belongs to V_128, a_i belongs to V_8,
   i=0,1,...,15,

L:V_128-> V_128  L(a)=R^(16)(a), where a belongs to V_128,

R^(-1):V_128-> V_128  the inverse transformation of R, which may be
   calculated, for example, as follows: R^(-1)(a_15||...||a_0)=a_14||
   a_13||...||a_0||l(a_14,a_13,...,a_0,a_15), where a_15||...||a_0
   belongs to V_128, a_i belongs to V_8, i=0,1,...,15,

L^(-1):V_128-> V_128  L^(-1)(a)=(R^(-1))(16)(a), where a belongs to
   V_128, and

F[k]:V_128[*]V_128 -> V_128[*]V_128
   F[k](a_1,a_0)=(LSX[k](a_1)(xor)a_0,a_1), where k, a_0, a_1 belong
   to V_128.
```

4.4. Key Schedule

Key schedule uses round constants C_i belonging to V_{128}, $i=1, 2, ..., 32$, defined as

$$C_i=L(Vec_{128}(i)), i=1,2,...,32.$$

Round keys K_i, $i=1, 2, ..., 10$ are derived from key $K=k_{255}||...||k_0$ belonging to V_{256}, k_i belongs to V_1, $i=0, 1, ..., 255$, as follows:

$$K_1=k_{255}||...||k_{128};$$
$$K_2=k_{127}||...||k_0;$$
$$(K_{(2i+1)},K_{(2i+2)})=F[C_{(8(i-1)+8)}]... F[C_{(8(i-1)+1)}](K_{(2i-1)},K_{(2i)}), i=1,2,3,4.$$

4.5. Basic Encryption Algorithm

4.5.1. Encryption

Depending on the values of round keys $K_1,...,K_{10}$, the encryption algorithm is a substitution $E_{(K_1,...,K_{10})}$ defined as follows:

$$E_{(K_1,...,K_{10})}(a)=X[K_{10}]LSX[K_9]...LSX[K_2]LSX[K_1](a),$$

where a belongs to V_{128}.

4.5.2. Decryption

Depending on the values of round keys $K_1,...,K_{10}$, the decryption algorithm is a substitution $D_{(K_1,...,K_{10})}$ defined as follows:

$$D_{(K_1,...,K_{10})}(a)=X[K_1]L^{(-1)}S^{(-1)}X[K_2]... L^{(-1)}S^{(-1)}X[K_9] L^{(-1)}S^{(-1)}X[K_{10}](a),$$

where a belongs to V_{128}.

5. Examples (Informative)

This section is for information only and is not a normative part of the standard.

5.1. Transformation S

```
   S(ffeeddccbbaa99881122334455667700) =
    b66cd8887d38e8d77765aeea0c9a7efc,
   S(b66cd8887d38e8d77765aeea0c9a7efc) =
    559d8dd7bd06cbfe7e7b262523280d39,
   S(559d8dd7bd06cbfe7e7b262523280d39) =
    0c3322fed531e4630d80ef5c5a81c50b,
   S(0c3322fed531e4630d80ef5c5a81c50b) =
    23ae65633f842d29c5df529c13f5acda.
```

5.2. Transformation R

```
   R(00000000000000000000000000000100) =
    94000000000000000000000000000001,
   R(94000000000000000000000000000001) =
    a5940000000000000000000000000000,
   R(a5940000000000000000000000000000) =
    64a59400000000000000000000000000,
   R(64a59400000000000000000000000000) =
    0d64a594000000000000000000000000.
```

5.3. Transformation L

```
   L(64a59400000000000000000000000000) =
    d456584dd0e3e84cc3166e4b7fa2890d,
   L(d456584dd0e3e84cc3166e4b7fa2890d) =
    79d26221b87b584cd42fbc4ffea5de9a,
   L(79d26221b87b584cd42fbc4ffea5de9a) =
    0e93691a0cfc60408b7b68f66b513c13,
   L(0e93691a0cfc60408b7b68f66b513c13) =
    e6a8094fee0aa204fd97bcb0b44b8580.
```

5.4. Key Schedule

In this test example, the key is equal to:

```
   K = 8899aabbccddeeff0011223344556677fedcba98765432100123456789abcdef.

   K_1 = 8899aabbccddeeff0011223344556677,
   K_2 = fedcba98765432100123456789abcdef.

   C_1 = 6ea276726c487ab85d27bd10dd849401,
   X[C_1](K_1) = e63bdcc9a09594475d369f2399d1f276,
   SX[C_1](K_1) = 0998ca37a7947aabb78f4a5ae81b748a,
   LSX[C_1](K_1) = 3d0940999db75d6a9257071d5e6144a6,
   F[C_1](K_1, K_2) = = (c3d5fa01ebe36f7a9374427ad7ca8949,
          8899aabbccddeeff0011223344556677).

   C_2 = dc87ece4d890f4b3ba4eb92079cbeb02,
   F [C_2]F [C_1](K_1, K_2) = (37777748e56453377d5e262d90903f87,
          c3d5fa01ebe36f7a9374427ad7ca8949).

   C_3 = b2259a96b4d88e0be7690430a44f7f03,
   F[C_3]...F[C_1](K_1, K_2) = (f9eae5f29b2815e31f11ac5d9c29fb01,
          37777748e56453377d5e262d90903f87).

   C_4 = 7bcd1b0b73e32ba5b79cb140f2551504,
   F[C_4]...F[C_1](K_1, K_2) = (e980089683d00d4be37dd3434699b98f,
          f9eae5f29b2815e31f11ac5d9c29fb01).

   C_5 = 156f6d791fab511deabb0c502fd18105,
   F[C_5]...F[C_1](K_1, K_2) = (b7bd70acea4460714f4ebe13835cf004,
          e980089683d00d4be37dd3434699b98f).

   C_6 = a74af7efab73df160dd208608b9efe06,
   F[C_6]...F[C_1](K_1, K_2) = (1a46ea1cf6ccd236467287df93fdf974,
          b7bd70acea4460714f4ebe13835cf004).

   C_7 = c9e8819dc73ba5ae50f5b570561a6a07,
   F[C_7]...F [C_1](K_1, K_2) = (3d4553d8e9cfec6815ebadc40a9ffd04,
          1a46ea1cf6ccd236467287df93fdf974).

   C_8 = f6593616e6055689adfba18027aa2a08,
   (K_3, K_4) = F [C_8]...F [C_1](K_1, K_2) =
          (db31485315694343228d6aef8cc78c44,
           3d4553d8e9cfec6815ebadc40a9ffd04).
```

The round keys K_i, i = 1, 2, ..., 10, take the following values:

```
K_1 = 8899aabbccddeeff0011223344556677,
K_2 = fedcba98765432100123456789abcdef,
K_3 = db31485315694343228d6aef8cc78c44,
K_4 = 3d4553d8e9cfec6815ebadc40a9ffd04,
K_5 = 57646468c44a5e28d3e59246f429f1ac,
K_6 = bd079435165c6432b532e82834da581b,
K_7 = 51e640757e8745de705727265a0098b1,
K_8 = 5a7925017b9fdd3ed72a91a22286f984,
K_9 = bb44e25378c73123a5f32f73cdb6e517,
K_10 = 72e9dd7416bcf45b755dbaa88e4a4043.
```

5.5. Test Encryption

In this test example, encryption is performed on the round keys specified in Section 5.4. Let the plaintext be

```
a = 1122334455667700ffeeddccbbaa9988,
```

then

```
X[K_1](a) = 99bb99ff99bb99ffffffffffffffffff,
SX[K_1](a) = e87de8b6e87de8b6b6b6b6b6b6b6b6b6,
LSX[K_1](a) = e297b686e355b0a1cf4a2f9249140830,
LSX[K_2]LSX[K_1](a) = 285e497a0862d596b36f4258a1c69072,
LSX[K_3]...LSX[K_1](a) = 0187a3a429b567841ad50d29207cc34e,
LSX[K_4]...LSX[K_1](a) = ec9bdba057d4f4d77c5d70619dcad206,
LSX[K_5]...LSX[K_1](a) = 1357fd11de9257290c2a1473eb6bcde1,
LSX[K_6]...LSX[K_1](a) = 28ae31e7d4c2354261027ef0b32897df,
LSX[K_7]...LSX[K_1](a) = 07e223d56002c013d3f5e6f714b86d2d,
LSX[K_8]...LSX[K_1](a) = cd8ef6cd97e0e092a8e4cca61b38bf65,
LSX[K_9]...LSX[K_1](a) = 0d8e40e4a800d06b2f1b37ea379ead8e.
```

Then the ciphertext is

```
b = X[K_10]LSX[K_9]...LSX[K_1](a) = 7f679d90bebc24305a468d42b9d4edcd.
```

5.6. Test Decryption

In this test example, decryption is performed on the round keys specified in Section 5.4. Let the ciphertext be

```
b = 7f679d90bebc24305a468d42b9d4edcd,
```

then

```
X[K_10](b) = 0d8e40e4a800d06b2f1b37ea379ead8e,
L^(-1)X[K_10](b) = 8a6b930a52211b45c5baa43ff8b91319,
S^(-1)L^(-1)X[K_10](b) = 76ca149eef27d1b10d17e3d5d68e5a72,
S^(-1)L^(-1)X[K_9]S^(-1)L^(-1)X[K_10](b) =
 5d9b06d41b9d1d2d04df7755363e94a9,
S^(-1)L^(-1)X[K_8]...S^(-1)L^(-1)X[K_10](b) =
 79487192aa45709c115559d6e9280f6e,
S^(-1)L^(-1)X[K_7]...S^(-1)L^(-1)X[K_10](b) =
 ae506924c8ce331bb918fc5bdfb195fa,
S^(-1)L^(-1)X[K_6]...S^(-1)L^(-1)X[K_10](b) =
 bbffbfc8939eaaffafb8e22769e323aa,
S^(-1)L^(-1)X[K_5]...S^(-1)L^(-1)X[K_10](b) =
 3cc2f07cc07a8bec0f3ea0ed2ae33e4a,
S^(-1)L^(-1)X[K_4]...S^(-1)L^(-1)X[K_10](b) =
 f36f01291d0b96d591e228b72d011c36,
S^(-1)L^(-1)X[K_3]...S^(-1)L^(-1)X[K_10](b) =
 1c4b0c1e950182b1ce696af5c0bfc5df,
S^(-1)L^(-1)X[K_2]...S^(-1)L^(-1)X[K_10](b) =
 99bb99ff99bb99ffffffffffffffffff.
```

Then the plaintext is

```
a = X[K_1]S^(-1)L^(-1)X[K_2]...S^(-1)L^(-1)X[K_10](b) =
 1122334455667700ffeeddccbbaa9988.
```

6. Security Considerations

This entire document is about security considerations.

7. References

7.1. Normative References

[GOST3412-2015]
"Information technology. Cryptographic data security. Block ciphers", GOST R 34.12-2015, Federal Agency on Technical Regulating and Metrology, 2015.

7.2. Informative References

[ISO-IEC10116]
ISO/IEC, "Information technology -- Security techniques -- Modes of operation for an n-bit block cipher", ISO/IEC 10116, 2006.

[ISO-IEC18033-1]
ISO/IEC, "Information technology -- Security techniques -- Encryption algorithms -- Part 1: General", ISO/IEC 18033-1, 2015.

[ISO-IEC18033-3]
ISO/IEC, "Information technology -- Security techniques -- Encryption algorithms -- Part 3: Block ciphers", ISO/IEC 18033-3, 2010.

Author’s Address

Vasily Dolmatov (editor)
Research Computer Center MSU
Leninskiye Gory, 1, Building 4, MGU NIVC
Moscow 119991
Russian Federation

Email: dol@srcc.msu.ru