Internet Engineering Task Force
Internet-Draft
Updates: 5830 (if approved)
Intended status: Informational
Expires: December 27, 2019

V. Dolmatov, Ed.
JSC "NPK Kryptonite"
D. Eremin-Solenikov
Auriga, Inc

June 25, 2019

GOST R 34.12-2015: Block Cipher "Magma"
draft-dolmatov-magma-00

Abstract

This document is intended to be a source of information about updated version of the block cipher with block length of n=64 bits and key length k=256 bits (RFC5830), which is also referred as "Magma" and is described in the Russian Federal standard GOST R 34.12-2015, containing also the description of block cipher "Kuznechik" (RFC7801)/>. These algorithms are from the set of Russian cryptographic standard algorithms (called GOST algorithms).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 27, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. General Information 3
3. Definitions and Notations 3
3.1. Definitions 3
3.2. Notations 4
4. Description of Kuznechik cipher 5
5. Parameter Values 5
5.1. Nonlinear Bijection 5
5.2. Transformations 6
5.3. Key schedule 6
6. Basic encryption algorithm 7
6.1. Encryption 7
6.2. Decryption 7
7. IANA Considerations 7
8. Security Considerations 7
9. References 7
9.1. Normative References 7
9.2. Informative References 8
Appendix A. Test Examples 8
A.1. Transformation t 8
A.2. Transformation g 8
A.3. Key schedule 9
A.4. Test Encryption 10
A.5. Test Decryption 11
Appendix B. Background 12
Authors' Addresses 12

1. Introduction

The Russian Federal standard [GOSTR3412-2015] 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.

2. General Information

The Russian Federal standard [GOSTR3412-2015] 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 19.06.2015.

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

- ISO/IEC 10116 [ISO-IEC10116],

- 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 which transforms plaintext into ciphertext (Clause 2.19 of [ISO-IEC18033-1]),

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

basic block cipher: block cipher which 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 (Clause 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 (Clause 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 (Clause 2.18 of [ISO-IEC18033-1]),

round key: sequence of symbols which 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, decipherment) (Clause 2.21 of [ISO-IEC18033-1]),

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

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

key schedule: calculation of round keys from the key,

decryption: reversal of a corresponding encipherment (Clause 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 (Clause 2.32 of [ISO-IEC18033-1]),

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

ciphertext: data which has been transformed to hide its information content (Clause 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 set 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,

A<<<_11 cyclic rotation of string A belonging to V_32 by 11 components in the direction of components having greater indices

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

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

[+] addition in the ring Z_(2^32)

Vec_s: Z_(2^s) -> V_s bijective mapping which 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),

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

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

4. Description of Kuznechik cipher

This section corresponds to "Kuznechik" cipher and is described in [RFC7801]

5. Parameter Values

5.1. Nonlinear Bijection

The bijective nonlinear mapping is a set of substitutions:

Pi_i = Vec_4 Pi'_i Int_4: V_4 -> V_4,

where

Pi'_i: Z_(2^4) -> Z_(2^4), i = 0, 1, ..., 7.

The values of the substitution Pi' are specified below as arrays

```
Pi'_i = (Pi'_i(0), Pi'_i(1), ... , Pi'_i(15)), i = 0, 1, ..., 7:

Pi'_0 = (12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1);
Pi'_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15);
Pi'_2 = (11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0);
Pi'_3 = (12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11);
Pi'_4 = (7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12);
Pi'_5 = (5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0);
Pi'_6 = (8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7);
Pi'_7 = (1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2);
```

5.2. Transformations

The following transformations are applicable for encryption and decryption algorithms:

```
t: V_32 -> V_32  t(a) = t(a_7||...||a_0) = Pi_7(a_7)||...||Pi_0(a_0),
   where a=a_7||...||a_0 belongs to V_32, a_i belongs to V_4, i=0, 1,
   ..., 7;

g[k]: V_32 -> V_32  g[k](a) = (t(Vec_32(Int_32(a) [+] Int_32(k))))
   <<<_11, where k, a belong to V_32;

G[k]: V_32[*]V_32 -> V_32[*]V_32  G[k](a_1, a_0) = (a_0, g[k](a_0)
   (xor) a_1), where k, a_0, a_1 belong to V_32;

G^*[k]: V_32[*]V_32 -> V_64  G^*[k](a_1, a_0) = (g[k](a_0) (xor)
   a_1) || a_0, where k, a_0, a_1 belong to V_32.
```

5.3. Key schedule

Round keys K_i belonging to V_32, i=1, 2, ..., 32 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_224;
K_2=k_223||...||k_192;
K_3=k_191||...||k_160;
K_4=k_159||...||k_128;
K_5=k_127||...||k_96;
K_6=k_95||...||k_64;
K_7=k_63||...||k_32;
K_8=k_31||...||k_0;
K_(i+8)=K_i, i = 1, 2, ..., 8;
K_(i+16)=K_i, i = 1, 2, ..., 8;
K_(i+24)=K_(9-i), i = 1, 2, ..., 8.
```

6. Basic encryption algorithm

6.1. Encryption

Depending on the values of round keys K_1,...,K_32, the encryption algorithm is a substitution E_(K_1,...,K_32) defined as follows:

E_(K_1,...,K_32)(a)=G^*[K_32]G[K_31]...G[K_2]G[K_1](a_1, a_0),

where a=(a_1, a_0) belongs to V_64, and a_0, a_1 belong to V_32.

6.2. Decryption

Depending on the values of round keys K_1,...,K_32, the decryption algorithm is a substitution D_(K_1,...,K_32) defined as follows:

D_(K_1,...,K_32)(a)=G^*[K_1]G[K_2]...G[K_31]G[K_32](a_1, a_0),

where a=(a_1, a_0) belongs to V_64, and a_0, a_1 belong to V_32.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This entire document is about security considerations.

9. References

9.1. Normative References

[GOSTR3412-2015]
Federal Agency on Technical Regulating and Metrology, "Information technology. Cryptographic data security. Block ciphers. GOST R 34.12-2015", 2015.

[RFC5830] Dolmatov, V., Ed., "GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms", RFC 5830, DOI 10.17487/RFC5830, March 2010, <https://www.rfc-editor.org/info/rfc5830>.

[RFC7801] Dolmatov, V., Ed., "GOST R 34.12-2015: Block Cipher "Kuznyechik"", RFC 7801, DOI 10.17487/RFC7801, March 2016, <https://www.rfc-editor.org/info/rfc7801>.

[RFC7836] Smyshlyaev, S., Ed., Alekseev, E., Oshkin, I., Popov, V., Leontiev, S., Podobaev, V., and D. Belyavsky, "Guidelines on the Cryptographic Algorithms to Accompany the Usage of Standards GOST R 34.10-2012 and GOST R 34.11-2012", RFC 7836, DOI 10.17487/RFC7836, March 2016, <https://www.rfc-editor.org/info/rfc7836>.

9.2. Informative References

[GOST28147-89]
Government Committee of the USSR for Standards, ""Cryptographic Protection for Data Processing System", GOST 28147-89, Gosudarstvennyi Standard of USSR", 1989.

[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", 2013.

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

Appendix A. Test Examples

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

A.1. Transformation t

```
t(fdb97531) = 2a196f34,
t(2a196f34) = ebd9f03a,
t(ebd9f03a) = b039bb3d,
t(b039bb3d) = 68695433.
```

A.2. Transformation g

```
g[87654321](fedcba98) = fdcbc20c,
g[fdcbc20c](87654321) = 7e791a4b,
g[7e791a4b](fdcbc20c) = c76549ec,
g[c76549ec](7e791a4b) = 9791c849.
```

A.3. Key schedule

With key set to

K = ffeeddccbbaa9988776655443322110 0f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff,

following round keys are generated:

K_1 = ffeeddcc,
K_2 = bbaa9988,
K_3 = 77665544,
K_4 = 33221100,
K_5 = f0f1f2f3,
K_6 = f4f5f6f7,
K_7 = f8f9fafb,
K_8 = fcfdfeff,

K_9 = ffeeddcc,
K_10 = bbaa9988,
K_11 = 77665544,
K_12 = 33221100,
K_13 = f0f1f2f3,
K_14 = f4f5f6f7,
K_15 = f8f9fafb,
K_16 = fcfdfeff,

K_17 = ffeeddcc,
K_18 = bbaa9988,
K_19 = 77665544,
K_20 = 33221100,
K_21 = f0f1f2f3,
K_22 = f4f5f6f7,
K_23 = f8f9fafb,
K_24 = fcfdfeff,

K_25 = fcfdfeff,
K_26 = f8f9fafb,
K_27 = f4f5f6f7,
K_28 = f0f1f2f3,
K_29 = 33221100,
K_30 = 77665544,
K_31 = bbaa9988,
K_32 = ffeeddcc.

A.4. Test Encryption

In this test example, encryption is performed on the round keys specified in clause A.3. Let the plaintext be

a = fedcba9876543210,

then

(a_1, a_0) = (fedcba98, 76543210),
G[K_1](a_1, a_0) = (76543210, 28da3b14),
G[K_2]G[K_1](a_1, a_0) = (28da3b14, b14337a5),
G[K_3]...G[K_1](a_1, a_0) = (b14337a5, 633a7c68),
G[K_4]...G[K_1](a_1, a_0) = (633a7c68, ea89c02c),
G[K_5]...G[K_1](a_1, a_0) = (ea89c02c, 11fe726d),
G[K_6]...G[K_1](a_1, a_0) = (11fe726d, ad0310a4),
G[K_7]...G[K_1](a_1, a_0) = (ad0310a4, 37d97f25),
G[K_8]...G[K_1](a_1, a_0) = (37d97f25, 46324615),
G[K_9]...G[K_1](a_1, a_0) = (46324615, ce995f2a),
G[K_10]...G[K_1](a_1, a_0) = (ce995f2a, 93c1f449),
G[K_11]...G[K_1](a_1, a_0) = (93c1f449, 4811c7ad),
G[K_12]...G[K_1](a_1, a_0) = (4811c7ad, c4b3edca),
G[K_13]...G[K_1](a_1, a_0) = (c4b3edca, 44ca5ce1),
G[K_14]...G[K_1](a_1, a_0) = (44ca5ce1, fef51b68),
G[K_15]...G[K_1](a_1, a_0) = (fef51b68, 2098cd86)
G[K_16]...G[K_1](a_1, a_0) = (2098cd86, 4f15b0bb),
G[K_17]...G[K_1](a_1, a_0) = (4f15b0bb, e32805bc),
G[K_18]...G[K_1](a_1, a_0) = (e32805bc, e7116722),
G[K_19]...G[K_1](a_1, a_0) = (e7116722, 89cadf21),
G[K_20]...G[K_1](a_1, a_0) = (89cadf21, bac8444d),
G[K_21]...G[K_1](a_1, a_0) = (bac8444d, 11263a21),
G[K_22]...G[K_1](a_1, a_0) = (11263a21, 625434c3),
G[K_23]...G[K_1](a_1, a_0) = (625434c3, 8025c0a5),
G[K_24]...G[K_1](a_1, a_0) = (8025c0a5, b0d66514),
G[K_25]...G[K_1](a_1, a_0) = (b0d66514, 47b1d5f4),
G[K_26]...G[K_1](a_1, a_0) = (47b1d5f4, c78e6d50),
G[K_27]...G[K_1](a_1, a_0) = (c78e6d50, 80251e99),
G[K_28]...G[K_1](a_1, a_0) = (80251e99, 2b96eca6),
G[K_29]...G[K_1](a_1, a_0) = (2b96eca6, 05ef4401),
G[K_30]...G[K_1](a_1, a_0) = (05ef4401, 239a4577),
G[K_31]...G[K_1](a_1, a_0) = (239a4577, c2d8ca3d).

Then the ciphertext is

b = G^*[K_32]G[K_31]...G[K_1](a_1, a_0) = 4ee901e5c2d8ca3d.

A.5. Test Decryption

In this test example, decryption is performed on the round keys specified in clause A.3. Let the ciphertext be

b = 4ee901e5c2d8ca3d,

then

(b_1, b_0) = (4ee901e5, c2d8ca3d),
G[K_32](b_1, b_0) = (c2d8ca3d, 239a4577),
G[K_31]G[K_32](b_1, b_0) = (239a4577, 05ef4401),
G[K_30]...G[K_32](b_1, b_0) = (05ef4401, 2b96eca6),
G[K_29]...G[K_32](b_1, b_0) = (2b96eca6, 80251e99),
G[K_28]...G[K_32](b_1, b_0) = (80251e99, c78e6d50),
G[K_27]...G[K_32](b_1, b_0) = (c78e6d50, 47b1d5f4),
G[K_26]...G[K_32](b_1, b_0) = (47b1d5f4, b0d66514),
G[K_25]...G[K_32](b_1, b_0) = (b0d66514, 8025c0a5),
G[K_24]...G[K_32](b_1, b_0) = (8025c0a5, 625434c3),
G[K_23]...G[K_32](b_1, b_0) = (625434c3, 11263a21),
G[K_22]...G[K_32](b_1, b_0) = (11263a21, bac8444d),
G[K_21]...G[K_32](b_1, b_0) = (bac8444d, 89cadf21),
G[K_20]...G[K_32](b_1, b_0) = (89cadf21, e7116722),
G[K_19]...G[K_32](b_1, b_0) = (e7116722, e32805bc),
G[K_18]...G[K_32](b_1, b_0) = (e32805bc, 4f15b0bb),
G[K_17]...G[K_32](b_1, b_0) = (4f15b0bb, 2098cd86),
G[K_16]...G[K_32](b_1, b_0) = (2098cd86, fef51b68),
G[K_15]...G[K_32](b_1, b_0) = (fef51b68, 44ca5ce1),
G[K_14]...G[K_32](b_1, b_0) = (44ca5ce1, c4b3edca),
G[K_13]...G[K_32](b_1, b_0) = (c4b3edca, 4811c7ad),
G[K_12]...G[K_32](b_1, b_0) = (4811c7ad, 93c1f449),
G[K_11]...G[K_32](b_1, b_0) = (93c1f449, ce995f2a),
G[K_10]...G[K_32](b_1, b_0) = (ce995f2a, 46324615),
G[K_9]...G[K_32](b_1, b_0) = (46324615, 37d97f25),
G[K_8]...G[K_32](b_1, b_0) = (37d97f25, ad0310a4),
G[K_7]...G[K_32](b_1, b_0) = (ad0310a4, 11fe726d),
G[K_6]...G[K_32](b_1, b_0) = (11fe726d, ea89c02c),
G[K_5]...G[K_32](b_1, b_0) = (ea89c02c, 633a7c68),
G[K_4]...G[K_32](b_1, b_0) = (633a7c68, b14337a5),
G[K_3]...G[K_32](b_1, b_0) = (b14337a5, 28da3b14),
G[K_2]...G[K_32](b_1, b_0) = (28da3b14, 76543210).

Then the plaintext is

a = G^*[K_1]G[K_2]...G[K_32](b_1, b_0) = fedcba9876543210.

Appendix B. Background

Algoritmically Magma is a variation of block cipher defined in [RFC5830] ([GOST28147-89]) with the following clarifications and minor modifications:

1. S-BOX set is fixed at id-tc26-gost-28147-param-Z (See Appendix C of [RFC7836]);

2. key is parsed as a single big-endian integer (compared to little-endian approach used in [GOST28147-89]), which results in different subkey values being used;

3. data are also parsed as single big-endian integer (instead of being parsed as little-endian integer).

Authors' Addresses

Vasily Dolmatov (editor)
JSC "NPK Kryptonite"
Spartakovskaya sq., 14, bld 2, JSC "NPK Kryptonite"
Moscow 105082
Russian Federation

Email: vdolmatov@gmail.com

Dmitry Eremin-Solenikov
Auriga, Inc
Torfyanaya Doroga, 7F, office 1410
Saint-Petersburg 197374
Russia

Email: dbaryshkov@gmail.com