Stream: Independent Submission
RFC: 9058
Category: Informational
Published: June 2021
ISSN: 2070-1721

Authors: S. Smyshlyaev, Ed. *CryptoPro*
V. Nozdrunov *TC 26*
V. Shishkin *TC 26*
E. Griboedova *CryptoPro*

# RFC 9058
# Multilinear Galois Mode (MGM)

## Abstract

Multilinear Galois Mode (MGM) is an Authenticated Encryption with Associated Data (AEAD) block cipher mode based on the Encrypt-then-MAC (EtM) principle. MGM is defined for use with 64-bit and 128-bit block ciphers.

MGM has been standardized in Russia. It is used as an AEAD mode for the GOST block cipher algorithms in many protocols, e.g., TLS 1.3 and IPsec. This document provides a reference for MGM to enable review of the mechanisms in use and to make MGM available for use with any block cipher.

## Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9058.

## Copyright Notice

# Table of Contents

# 1. Introduction

Multilinear Galois Mode (MGM) is an Authenticated Encryption with Associated Data (AEAD) block cipher mode based on EtM principle. MGM is defined for use with 64-bit and 128-bit block ciphers. The MGM design principles can easily be applied to other block sizes.

MGM has been standardized in Russia [AUTH-ENC-BLOCK-CIPHER]. It is used as an AEAD mode for the GOST block cipher algorithms in many protocols, e.g., TLS 1.3 and IPsec. This document provides a reference for MGM to enable review of the mechanisms in use and to make MGM available for use with any block cipher.

This document does not have IETF consensus and does not imply IETF support for MGM.

## 2. Conventions Used in This Document

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Basic Terms and Definitions

This document uses the following terms and definitions for the sets and operations on the elements of these sets:

| | |
|---|---|
| V* | The set of all bit strings of a finite length (hereinafter referred to as strings), including the empty string; substrings and string components are enumerated from right to left starting from zero. |
| V_s | The set of all bit strings of length s, where s is a non-negative integer. For s = 0, the V_0 consists of a single empty string. |
| \|X\| | The bit length of the bit string X (if X is an empty string, then \|X\| = 0). |
| X \|\| Y | Concatenation of strings X and Y both belonging to V*, i.e., a string from V_{\|X\|+\|Y\|}, where the left substring from V_{\|X\|} is equal to X, and the right substring from V_{\|Y\|} is equal to Y. |
| a^s | The string in V_s that consists of s 'a' bits. |
| (xor) | Exclusive-or of two bit strings of the same length. |
| Z_{2^s} | Ring of residues modulo 2^s. |
| MSB_i | V_s -> V_i<br>The transformation that maps the string X = (x_{s-1}, ... , x_0) in V_s into the string MSB_i(X) = (x_{s-1}, ... , x_{s-i}) in V_i, i <= s (most significant bits). |
| Int_s | V_s -> Z_{2^s}<br>The transformation that maps the string X = (x_{s-1}, ... , x_0) in V_s, s > 0, into the integer Int_s(X) = 2^{s-1} * x_{s-1} + ... + 2 * x_1 + x_0 (the interpretation of the bit string as an integer). |

Vec_s Z_{2^s} -> V_s

The transformation inverse to the mapping Int_s (the interpretation of an integer as a bit string).

E_K V_n -> V_n

The block cipher permutation under the key K in V_k.

k The bit length of the block cipher key.

n The block size of the block cipher (in bits).

len V_s -> V_{n/2}

The transformation that maps a string X in V_s, 0 <= s <= 2^{n/2} - 1, into the string len(X) = Vec_{n/2}(|X|) in V_{n/2}, where n is the block size of the used block cipher.

[+] The addition operation in Z_{2^{n/2}}, where n is the block size of the used block cipher.

(x) The transformation that maps two strings, X = (x_{n-1}, ... , x_0) in V_n and Y = (y_{n-1}, ... , y_0), in V_n into the string Z = X (x) Y = (z_{n-1}, ... , z_0) in V_n; the string Z corresponds to the polynomial Z(w) = z_{n-1} * w^{n-1} + ... + z_1 * w + z_0, which is the result of multiplying the polynomials X(w) = x_{n-1} * w^{n-1} + ... + x_1 * w + x_0 and Y(w) = y_{n-1} * w^{n-1} + ... + y_1 * w + y_0 in the field GF(2^n), where n is the block size of the used block cipher; if n = 64, then the field polynomial is equal to f(w) = w^64 + w^4 + w^3 + w + 1; if n = 128, then the field polynomial is equal to f(w) = w^128 + w^7 + w^2 + w + 1.

incr_l V_n -> V_n

The transformation that maps an n-byte string A = L || R into the n-byte string incr_l(A) = Vec_{n/2}(Int_{n/2}(L) [+] 1) || R, where L and R are n/2-byte strings.

incr_r V_n -> V_n

The transformation that maps an n-byte string A = L || R into the n-byte string incr_r(A) = L || Vec_{n/2}(Int_{n/2}(R) [+] 1), where L and R are n/2-byte strings.

## 4. Specification

An additional parameter that defines the functioning of MGM is the bit length S of the authentication tag, 32 <= S <= n. The value of S **MUST** be fixed for a particular protocol. The choice of the value S involves a trade-off between message expansion and the forgery probability.

## 4.1. MGM Encryption and Tag Generation Procedure

The MGM encryption and tag generation procedure takes the following parameters as inputs:

1. Encryption key K in V_k.
2. Initial counter nonce ICN in V_{n-1}.
3. Associated authenticated data A, 0 <= |A| < 2^{n/2}. If |A| > 0, then A = A_1 || ... || A*_h, A_j in V_n, for j = 1, ... , h - 1, A*_h in V_t, 1 <= t <= n. If |A| = 0, then by definition A*_h is empty, and the h and t parameters are set as follows: h = 0, t = n. The associated data is authenticated but is not encrypted.
4. Plaintext P, 0 <= |P| < 2^{n/2}. If |P| > 0, then P = P_1 || ... || P*_q, P_i in V_n, for i = 1, ... , q - 1, P*_q in V_u, 1 <= u <= n. If |P| = 0, then by definition P*_q is empty, and the q and u parameters are set as follows: q = 0, u = n.

The MGM encryption and tag generation procedure outputs the following parameters:

1. Initial counter nonce ICN.
2. Associated authenticated data A.
3. Ciphertext C in V_{|P|}.
4. Authentication tag T in V_S.

The MGM encryption and tag generation procedure consists of the following steps:

```
+----------------------------------------------------------------+
|  MGM-Encrypt(K, ICN, A, P)                                     |
|----------------------------------------------------------------|
|  1. Encryption step:                                           |
|      - if |P| = 0 then                                         |
|            - C*_q = P*_q                                       |
|            - C = P                                             |
|      - else                                                    |
|            - Y_1 = E_K(0^1 || ICN),                            |
|            - For i = 2, 3, ... , q do                          |
|                    Y_i = incr_r(Y_{i-1}),                      |
|            - For i = 1, 2, ... , q - 1 do                      |
|                    C_i = P_i (xor) E_K(Y_i),                   |
|            - C*_q = P*_q (xor) MSB_u(E_K(Y_q)),                |
|            - C = C_1 || ... || C*_q.                           |
|                                                                |
|  2. Padding step:                                              |
|      - A_h = A*_h || 0^{n-t},                                  |
|      - C_q = C*_q || 0^{n-u}.                                  |
|                                                                |
|  3. Authentication tag T generation step:                      |
|      - Z_1 = E_K(1^1 || ICN),                                  |
|      - sum = 0^n,                                              |
|      - For i = 1, 2, ..., h do                                 |
|              H_i = E_K(Z_i),                                   |
|              sum = sum (xor) ( H_i (x) A_i ),                  |
|              Z_{i+1} = incr_l(Z_i),                            |
|      - For j = 1, 2, ..., q do                                 |
|              H_{h+j} = E_K(Z_{h+j}),                           |
|              sum = sum (xor) ( H_{h+j} (x) C_j ),              |
|              Z_{h+j+1} = incr_l(Z_{h+j}),                      |
|      - H_{h+q+1} = E_K(Z_{h+q+1}),                             |
|      - T = MSB_S(E_K(sum (xor) ( H_{h+q+1} (x)                 |
|                       ( len(A) || len(C) ) ))).                |
|                                                                |
|  4. Return (ICN, A, C, T).                                     |
+----------------------------------------------------------------+
```

The ICN value for each message that is encrypted under the given key K must be chosen in a unique manner.

Users who do not wish to encrypt plaintext can provide a string P of zero length. Users who do not wish to authenticate associated data can provide a string A of zero length. The length of the associated data A and of the plaintext P **MUST** be such that 0 < |A| + |P| < 2^{n/2}.

## 4.2. MGM Decryption and Tag Verification Check Procedure

The MGM decryption and tag verification procedure takes the following parameters as inputs:

1. Encryption key K in V_k.
2. Initial counter nonce ICN in V_{n-1}.

3. Associated authenticated data A, 0 <= |A| < 2^{n/2}. If |A| > 0, then A = A_1 || ... || A*_h, A_j in V_n, for j = 1, ... , h - 1, A*_h in V_t, 1 <= t <= n. If |A| = 0, then by definition A*_h is empty, and the h and t parameters are set as follows: h = 0, t = n. The associated data is authenticated but is not encrypted.
4. Ciphertext C, 0 <= |C| < 2^{n/2}. If |C| > 0, then C = C_1 || ... || C*_q, C_i in V_n, for i = 1, ... , q - 1, C*_q in V_u, 1 <= u <= n. If |C| = 0, then by definition C*_q is empty, and the q and u parameters are set as follows: q = 0, u = n.
5. Authentication tag T in V_S.

The MGM decryption and tag verification procedure outputs FAIL or the following parameters:

1. Associated authenticated data A.
2. Plaintext P in V_{|C|}.

The MGM decryption and tag verification procedure consists of the following steps:

```
+----------------------------------------------------------------+
|  MGM-Decrypt(K, ICN, A, C, T)                                  |
|----------------------------------------------------------------|
|  1. Padding step:                                              |
|      - A_h = A*_h || 0^{n-t},                                  |
|      - C_q = C*_q || 0^{n-u}.                                  |
|                                                                |
|  2. Authentication tag T verification step:                    |
|      - Z_1 = E_K(1^1 || ICN),                                  |
|      - sum = 0^n,                                              |
|      - For i = 1, 2, ..., h do                                 |
|              H_i = E_K(Z_i),                                   |
|              sum = sum (xor) ( H_i (x) A_i ),                  |
|              Z_{i+1} = incr_l(Z_i),                            |
|      - For j = 1,  2, ..., q do                                |
|              H_{h+j} = E_K(Z_{h+j}),                           |
|              sum = sum (xor) ( H_{h+j} (x) C_j ),              |
|              Z_{h+j+1} = incr_l(Z_{h+j}),                      |
|      - H_{h+q+1} = E_K(Z_{h+q+1}),                             |
|      - T' = MSB_S(E_K(sum (xor) ( H_{h+q+1} (x)                |
|                       ( len(A) || len(C) ) ))),                |
|      - If T' != T then return FAIL.                            |
|                                                                |
|  3. Decryption step:                                           |
|      - if |C| = 0 then                                         |
|            - P = C                                             |
|      - else                                                    |
|            - Y_1 = E_K(0^1 || ICN),                            |
|            - For i = 2, 3, ... , q do                          |
|                    Y_i = incr_r(Y_{i-1}),                      |
|            - For i = 1, 2, ... , q - 1 do                      |
|                    P_i = C_i (xor) E_K(Y_i),                   |
|            - P*_q = C*_q (xor) MSB_u(E_K(Y_q)),                |
|            - P = P_1 || ... || P*_q.                           |
|                                                                |
|  4. Return (A, P).                                             |
+----------------------------------------------------------------+
```

The length of the associated data A and of the ciphertext C **MUST** be such that $0 < |A| + |C| < 2^{n/2}$.

## 5. Rationale

MGM was originally proposed in [PDMODE].

From the operational point of view, MGM is designed to be parallelizable, inverse free, and online and is also designed to provide availability of precomputations.

Parallelizability of MGM is achieved due to its counter-type structure and the usage of the multilinear function for authentication. Indeed, both encryption blocks E_K(Y_i) and authentication blocks H_i are produced in the counter mode manner, and the multilinear function determined by H_i is parallelizable in itself. Additionally, the counter-type structure of the mode provides the inverse-free property.

The online property means the possibility of processing messages even if it is not completely received (so its length is unknown). To provide this property, MGM uses blocks E_K(Y_i) and H_i, which are produced based on two independent source blocks Y_i and Z_i.

Availability of precomputations for MGM means the possibility of calculating H_i and E_K(Y_i) even before data is retrieved. It holds again due to the usage of counters for calculating them.

## 6. Security Considerations

The security properties of MGM are based on the following:

Different functions generating the counter values:
: The functions incr_r and incr_l are chosen to minimize intersection (if it happens) of counter values Y_i and Z_i.

Encryption of the multilinear function output:
: It allows attacks based on padding and linear properties to be resisted (see [FERG05] for details).

Multilinear function for authentication:
: It allows the small subgroup attacks to be resisted [SAAR12].

Encryption of the nonces (0^1 || ICN) and (1^1 || ICN):
: The use of this encryption minimizes the number of plaintext/ciphertext pairs of blocks known to an adversary. It prevents attacks that need a substantial amount of such material (e.g., linear and differential cryptanalysis and side-channel attacks).

It is crucial to the security of MGM to use unique ICN values. Using the same ICN values for two different messages encrypted with the same key eliminates the security properties of this mode.

It is crucial for the security of MGM not to process empty plaintext and empty associated data at the same time. Otherwise, a tag becomes independent from a nonce value, leading to vulnerability to forgery attacks.

Security analysis for MGM with E_K being a random permutation was performed in [SEC-MGM]. More precisely, the bounds for confidentiality advantage (CA) and integrity advantage (IA) (for details, see [AEAD-LIMITS]) were obtained. According to these results, for an adversary making at most q encryption queries with the total length of plaintexts and associated data of at most s blocks, and allowed to output a forgery with the summary length of ciphertext and associated data of at most l blocks:

CA <= ( 3( s + 4q )^2 )/ 2^n,

IA <= ( 3( s + 4q + l + 3 )^2 )/ 2^n + 2/2^S,

where n is the block size and S is the authentication tag size.

These bounds can be used as guidelines on how to calculate confidentiality and integrity limits (for details, also see [AEAD-LIMITS]).

# 7. IANA Considerations

This document has no IANA actions.

# 8. References

## 8.1. Normative References

**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

**[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>.

**[RFC8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

**[RFC8891]** Dolmatov, V., Ed. and D. Baryshkov, "GOST R 34.12-2015: Block Cipher "Magma"", RFC 8891, DOI 10.17487/RFC8891, September 2020, <https://www.rfc-editor.org/info/rfc8891>.

## 8.2. Informative References

**[AEAD-LIMITS]**

Günther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-02, 22 February 2021, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-02>.

**[AUTH-ENC-BLOCK-CIPHER]** Federal Agency on Technical Regulating and Metrology, "Information technology. Cryptographic data security. Authenticated encryption block cipher operation modes", R 1323565.1.026-2019, 2019.

**[FERG05]** Ferguson, N., "Authentication weaknesses in GCM", May 2005.

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

**[PDMODE]** Nozdrunov, V., "Parallel and double block cipher mode of operation (PD-mode) for authenticated encryption", CTCrypt 2017 proceedings, pp. 36-45 , June 2017.

**[SAAR12]** Saarinen, M-J., "Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes", FSE 2012 proceedings, pp. 216-225, DOI 10.1007/978-3-642-34047-5_13, 2012, <https://doi.org/10.1007/978-3-642-34047-5_13>.

**[SEC-MGM]** Akhmetzyanova, L., Alekseev, E., Karpunin, G., and V. Nozdrunov, "Security of Multilinear Galois Mode (MGM)", IACR Cryptology ePrint Archive 2019, pp. 123, 2019.

# Appendix A. Test Vectors

## A.1. Test Vectors for the Kuznyechik Block Cipher

Test vectors for the Kuznyechik block cipher (n = 128, k = 256) are defined in [GOST3412-2015] (the English version can be found in [RFC7801]).

### A.1.1. Example 1

```
Encryption key K:
00000:   88 99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77
00010:   FE DC BA 98 76 54 32 10 01 23 45 67 89 AB CD EF

ICN:
00000:   11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Associated authenticated data A:
00000:   02 02 02 02 02 02 02 02 01 01 01 01 01 01 01 01
00010:   04 04 04 04 04 04 04 04 03 03 03 03 03 03 03 03
00020:   EA 05 05 05 05 05 05 05 05

Plaintext P:
00000:   11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88
00010:   00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A
00020:   11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00
00030:   22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11
00040:   AA BB CC
```

1. Encryption step:

```
0^1 || ICN:
00000:   11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Y_1:
00000:   7F 67 9D 90 BE BC 24 30 5A 46 8D 42 B9 D4 ED CD
E_K(Y_1):
00000:   B8 57 48 C5 12 F3 19 90 AA 56 7E F1 53 35 DB 74

Y_2:
00000:   7F 67 9D 90 BE BC 24 30 5A 46 8D 42 B9 D4 ED CE
E_K(Y_2):
00000:   80 64 F0 12 6F AC 9B 2C 5B 6E AC 21 61 2F 94 33

Y_3:
00000:   7F 67 9D 90 BE BC 24 30 5A 46 8D 42 B9 D4 ED CF
E_K(Y_3):
00000:   58 58 82 1D 40 C0 CD 0D 0A C1 E6 C2 47 09 8F 1C

Y_4:
00000:   7F 67 9D 90 BE BC 24 30 5A 46 8D 42 B9 D4 ED D0
E_K(Y_4):
00000:   E4 3F 50 81 B5 8F 0B 49 01 2F 8E E8 6A CD 6D FA

Y_5:
00000:   7F 67 9D 90 BE BC 24 30 5A 46 8D 42 B9 D4 ED D1
E_K(Y_5):
00000:   86 CE 9E 2A 0A 12 25 E3 33 56 91 B2 0D 5A 33 48

C:
00000:   A9 75 7B 81 47 95 6E 90 55 B8 A3 3D E8 9F 42 FC
00010:   80 75 D2 21 2B F9 FD 5B D3 F7 06 9A AD C1 6B 39
00020:   49 7A B1 59 15 A6 BA 85 93 6B 5D 0E A9 F6 85 1C
00030:   C6 0C 14 D4 D3 F8 83 D0 AB 94 42 06 95 C7 6D EB
00040:   2C 75 52
```

2. Padding step:

```
A_1 || ... || A_h:
00000:   02 02 02 02 02 02 02 02 01 01 01 01 01 01 01 01
00010:   04 04 04 04 04 04 04 04 03 03 03 03 03 03 03 03
00020:   EA 05 05 05 05 05 05 05 05 00 00 00 00 00 00 00

C_1 || ... || C_q:
00000:   A9 75 7B 81 47 95 6E 90 55 B8 A3 3D E8 9F 42 FC
00010:   80 75 D2 21 2B F9 FD 5B D3 F7 06 9A AD C1 6B 39
00020:   49 7A B1 59 15 A6 BA 85 93 6B 5D 0E A9 F6 85 1C
00030:   C6 0C 14 D4 D3 F8 83 D0 AB 94 42 06 95 C7 6D EB
00040:   2C 75 52 00 00 00 00 00 00 00 00 00 00 00 00 00
```

3. Authentication tag T generation step:

```
1^1 || ICN:
00000:   91 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Z_1:
00000:   7F C2 45 A8 58 6E 66 02 A7 BB DB 27 86 BD C6 6F
H_1:
00000:   8D B1 87 D6 53 83 0E A4 BC 44 64 76 95 2C 30 0B
current sum:
00000:   4C F4 27 F4 AD B7 5C F4 C0 DA 39 D5 AB 48 CF 38

Z_2:
00000:   7F C2 45 A8 58 6E 66 03 A7 BB DB 27 86 BD C6 6F
H_2:
00000:   7A 24 F7 26 30 E3 76 37 21 C8 F3 CD B1 DA 0E 31
current sum:
00000:   94 95 44 0E F6 24 A1 DD C6 F5 D9 77 28 50 C5 73

Z_3:
00000:   7F C2 45 A8 58 6E 66 04 A7 BB DB 27 86 BD C6 6F
H_3:
00000:   44 11 96 21 17 D2 06 35 C5 25 E0 A2 4D B4 B9 0A
current sum:
00000:   A4 9A 8C D8 A6 F2 74 23 DB 79 E4 4A B3 06 D9 42

Z_4:
00000:   7F C2 45 A8 58 6E 66 05 A7 BB DB 27 86 BD C6 6F
H_4:
00000:   D8 C9 62 3C 4D BF E8 14 CE 7C 1C 0C EA A9 59 DB
current sum:
00000:   09 FE 3F 6A 83 3C 21 B3 90 27 D0 20 6A 84 E1 5A

Z_5:
00000:   7F C2 45 A8 58 6E 66 06 A7 BB DB 27 86 BD C6 6F
H_5:
00000:   A5 E1 F1 95 33 3E 14 82 96 99 31 BF BE 6D FD 43
current sum:
00000:   B5 DA 26 BB 00 EB A8 04 35 D7 97 6B C6 B5 46 4D

Z_6:
00000:   7F C2 45 A8 58 6E 66 07 A7 BB DB 27 86 BD C6 6F
H_6:
00000:   B4 CA 80 8C AC CF B3 F9 17 24 E4 8A 2C 7E E9 D2
current sum:
00000:   DD 1C 0E EE F7 83 C8 EB 2A 33 F3 58 D7 23 0E E5

Z_7:
00000:   7F C2 45 A8 58 6E 66 08 A7 BB DB 27 86 BD C6 6F
H_7:
00000:   72 90 8F C0 74 E4 69 E8 90 1B D1 88 EA 91 C3 31
current sum:
00000:   89 6C E1 08 32 EB EA F9 06 9F 3F 73 76 59 4D 40

Z_8:
00000:   7F C2 45 A8 58 6E 66 09 A7 BB DB 27 86 BD C6 6F
H_8:
00000:   23 CA 27 15 B0 2C 68 31 3B FD AC B3 9E 4D 0F B8
current sum:
```

```
00000:   99 1A F5 C9 D0 80 F7 63 87 FE 64 9E 7C 93 C6 42

Z_9:
00000:   7F C2 45 A8 58 6E 66 0A A7 BB DB 27 86 BD C6 6F
H_9:
00000:   BC BC E6 C4 1A A3 55 A4 14 88 62 BF 64 BD 83 0D
len(A) || len(C):
00000:   00 00 00 00 00 00 01 48 00 00 00 00 00 00 02 18
sum (xor) ( H_9 (x) ( len(A) || len(C) ) ):
00000:   C0 C7 22 DB 5E 0B D6 DB 25 76 73 83 3D 56 71 28


Tag T:
00000:   CF 5D 65 6F 40 C3 4F 5C 46 E8 BB 0E 29 FC DB 4C
```

### A.1.2. Example 2

```
Encryption key K:
00000:   99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77 FE
00010:   DC BA 98 76 54 32 10 01 23 45 67 89 AB CD EF 88

ICN:
00000:   11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Associated authenticated data A:
00000:   01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

Plaintext P:
00000:
```

1. Encryption step:

```
C:
00000:
```

2. Padding step:

```
A_1 || ... || A_h:
00000:   01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

C_1 || ... || C_q:
00000:
```

3. Authentication tag T generation step:

```
1^1 || ICN:
00000:   91 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Z_1:
00000:   79 32 72 68 96 C4 3E 3F BF D6 50 89 EB F1 E5 B6
H_1:
00000:   99 3A 80 66 CC C0 A4 0F AC 4A 14 F7 A2 F6 6D 9B
current sum:
00000:   0A C1 1E 2C 1C D6 07 D8 2F E3 55 54 B4 01 02 81

Z_2:
00000:   79 32 72 68 96 C4 3E 40 BF D6 50 89 EB F1 E5 B6
H_2:
00000:   0C 38 A7 1E E7 93 BF 76 89 81 BF CD 7C DA 78 C8
len(A) || len(C):
00000:   00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 00
sum (xor) ( H_2 (x) ( len(A) || len(C) ) ):
00000:   CA 1E F8 92 71 EA 60 C4 53 9E 40 EB 26 C2 80 5D

Tag T:
00000:   79 01 E9 EA 20 85 CD 24 7E D2 49 69 5F 9F 8A 85
```

## A.2. Test Vectors for the Magma Block Cipher

Test vectors for the Magma block cipher (n = 64, k = 256) are defined in [GOST3412-2015] (the English version can be found in [RFC8891]).

### A.2.1. Example 1

```
Encryption key K:
00000:   FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00
00010:   F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

ICN:
00000:   12 DE F0 6B 3C 13 0A 59

Associated authenticated data A:
00000:   01 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02
00010:   03 03 03 03 03 03 03 03 04 04 04 04 04 04 04 04
00020:   05 05 05 05 05 05 05 05 EA

Plaintext P:
00000:   FF EE DD CC BB AA 99 88 11 22 33 44 55 66 77 00
00010:   88 99 AA BB CC EE FF 0A 00 11 22 33 44 55 66 77
00020:   99 AA BB CC EE FF 0A 00 11 22 33 44 55 66 77 88
00030:   AA BB CC EE FF 0A 00 11 22 33 44 55 66 77 88 99
00040:   AA BB CC
```

1. Encryption step:

```
0^1 || ICN:
00000:   12 DE F0 6B 3C 13 0A 59

Y_1:
00000:   56 23 89 01 62 DE 31 BF
E_K(Y_1):
00000:   38 7B DB A0 E4 34 39 B3

Y_2:
00000:   56 23 89 01 62 DE 31 C0
E_K(Y_2):
00000:   94 33 00 06 10 F7 F2 AE

Y_3:
00000:   56 23 89 01 62 DE 31 C1
E_K(Y_3):
00000:   97 B7 AA 6D 73 C5 87 57

Y_4:
00000:   56 23 89 01 62 DE 31 C2
E_K(Y_4):
00000:   94 15 52 8B FF C9 E8 0A

Y_5:
00000:   56 23 89 01 62 DE 31 C3
E_K(Y_5):
00000:   03 F7 68 BF F1 82 D6 70

Y_6:
00000:   56 23 89 01 62 DE 31 C4
E_K(Y_6):
00000:   FD 05 F8 4E 9B 09 D2 FE

Y_7:
00000:   56 23 89 01 62 DE 31 C5
E_K(Y_7):
00000:   DA 4D 90 8A 95 B1 75 C4

Y_8:
00000:   56 23 89 01 62 DE 31 C6
E_K(Y_8):
00000:   65 99 73 96 DA C2 4B D7

Y_9:
00000:   56 23 89 01 62 DE 31 C7
E_K(Y_9):
00000:   A9 00 50 4A 14 8D EE 26

C:
00000:   C7 95 06 6C 5F 9E A0 3B 85 11 33 42 45 91 85 AE
00010:   1F 2E 00 D6 BF 2B 78 5D 94 04 70 B8 BB 9C 8E 7D
00020:   9A 5D D3 73 1F 7D DC 70 EC 27 CB 0A CE 6F A5 76
00030:   70 F6 5C 64 6A BB 75 D5 47 AA 37 C3 BC B5 C3 4E
00040:   03 BB 9C
```

2. Padding step:

```
A_1 || ... || A_h:
00000:   01 01 01 01 01 01 01 01 02 02 02 02 02 02 02 02
00010:   03 03 03 03 03 03 03 03 04 04 04 04 04 04 04 04
00020:   05 05 05 05 05 05 05 05 EA 00 00 00 00 00 00 00

C_1 || ... || C_q:
00000:   C7 95 06 6C 5F 9E A0 3B 85 11 33 42 45 91 85 AE
00010:   1F 2E 00 D6 BF 2B 78 5D 94 04 70 B8 BB 9C 8E 7D
00020:   9A 5D D3 73 1F 7D DC 70 EC 27 CB 0A CE 6F A5 76
00030:   70 F6 5C 64 6A BB 75 D5 47 AA 37 C3 BC B5 C3 4E
00040:   03 BB 9C 00 00 00 00 00
```

3. Authentication tag T generation step:

```
1^1 || ICN:
00000:   92 DE F0 6B 3C 13 0A 59

Z_1:
00000:   2B 07 3F 04 94 F3 72 A0
H_1:
00000:   70 8A 78 19 1C DD 22 AA
current sum:
00000:   D6 BB 5B EA 81 93 12 62

Z_2:
00000:   2B 07 3F 05 94 F3 72 A0
H_2:
00000:   6F 02 CC 46 4B 2F A0 A3
current sum:
00000:   DD 1C 82 4E 91 78 49 A5

Z_3:
00000:   2B 07 3F 06 94 F3 72 A0
H_3:
00000:   9F 81 F2 26 FD 19 6F 05
current sum:
00000:   05 89 22 17 F6 5A DA C7

Z_4:
00000:   2B 07 3F 07 94 F3 72 A0
H_4:
00000:   B9 C2 AC 9B E5 B5 DF F9
current sum:
00000:   D1 DB 9B 7F C4 9E 7C 97

Z_5:
00000:   2B 07 3F 08 94 F3 72 A0
H_5:
00000:   74 B5 EC 96 55 1B F8 88
current sum:
00000:   56 45 F6 B5 18 5C B7 1A

Z_6:
00000:   2B 07 3F 09 94 F3 72 A0
H_6:
00000:   7E B0 21 A4 03 5B 04 C3
current sum:
00000:   3F C2 C2 E6 FB EE D0 4D

Z_7:
00000:   2B 07 3F 0A 94 F3 72 A0
H_7:
00000:   C2 A9 C3 A8 70 4D 9B B0
current sum:
00000:   15 47 1F B5 CD 8E 6C 02

Z_8:
00000:   2B 07 3F 0B 94 F3 72 A0
H_8:
00000:   F5 D5 05 A8 7B 83 83 B5
current sum:
```

```
00000:   12 56 78 96 1D 40 E0 93

Z_9:
00000:   2B 07 3F 0C 94 F3 72 A0
H_9:
00000:   F7 95 E7 5F DE B8 93 3C
current sum:
00000:   6E F4 0A B0 C1 5F 20 48

Z_10:
00000:   2B 07 3F 0D 94 F3 72 A0
H_10:
00000:   65 A1 A3 E6 80 F0 81 45
current sum:
00000:   A4 64 A7 08 FF 45 14 22

Z_11:
00000:   2B 07 3F 0E 94 F3 72 A0
H_11:
00000:   1C 74 A5 76 4C B0 D5 95
current sum:
00000:   60 94 4E 05 D0 85 75 14

Z_12:
00000:   2B 07 3F 0F 94 F3 72 A0
H_12:
00000:   DC 84 47 A5 14 E7 83 E7
current sum:
00000:   EE 98 B9 B5 0F F7 83 E8

Z_13:
00000:   2B 07 3F 10 94 F3 72 A0
H_13:
00000:   A7 E3 AF E0 04 EE 16 E3
current sum:
00000:   C0 39 0F A2 28 AF 6D CB

Z_14:
00000:   2B 07 3F 11 94 F3 72 A0
H_14:
00000:   A5 AA BB 0B 79 80 D0 71
current sum:
00000:   73 E0 6E 07 EF 37 CD CC

Z_15:
00000:   2B 07 3F 12 94 F3 72 A0
H_15:
00000:   6E 10 4C C9 33 52 5C 5D
current sum:
00000:   2F 40 69 0A EB 53 F5 39

Z_16:
00000:   2B 07 3F 13 94 F3 72 A0
H_16:
00000:   83 11 B6 02 4A A9 66 C1
len(A) || len(C):
00000:   00 00 01 48 00 00 02 18
sum (xor) ( H_16 (x) ( len(A) || len(C) ) ):
```

```
00000:   73 CE F4 4B AE 6B DB 61


Tag T:
00000:   A7 92 80 69 AA 10 FD 10
```

### A.2.2. Example 2

```
Encryption key K:
00000:   99 AA BB CC DD EE FF 00 11 22 33 44 55 66 77 FE
00010:   DC BA 98 76 54 32 10 01 23 45 67 89 AB CD EF 88

ICN:
00000:   00 77 66 55 44 33 22 11

Associated authenticated data A:
00000:

Plaintext P:
00000:   22 33 44 55 66 77 00 FF
```

1. Encryption step:

```
0^1 || ICN:
00000:   00 77 66 55 44 33 22 11

Y_1:
00000:   5B 2A 7E 60 4F 9F BB 95
E_K(Y_1):
00000:   48 A6 A5 17 0D 52 9D B1

C:
00000:   6A 95 E1 42 6B 25 9D 4E
```

2. Padding step:

```
A_1 || ... || A_h:
00000:

C_1 || ... || C_q:
00000:   6A 95 E1 42 6B 25 9D 4E
```

3. Authentication tag T generation step:

```
1^1 || ICN:
00000:   80 77 66 55 44 33 22 11

Z_1:
00000:   59 73 54 78 7E 52 E6 EB
H_1:
00000:   EC E3 F9 DA 11 8C 7D 95
current sum:
00000:   25 D0 E4 20 7B 6B F6 3D

Z_2:
00000:   59 73 54 79 7E 52 E6 EB
H_2:
00000:   31 0C 0D AC C9 D0 4D 93
len(A) || len(C):
00000:   00 00 00 00 00 00 00 40
sum (xor) ( H_2 (x) ( len(A) || len(C) ) ):
00000:   66 D3 8F 12 0F 78 92 49


Tag T:
00000:   33 4E E2 70 45 0B EC 9E
```

## Contributors

**Evgeny Alekseev**
CryptoPro
Email: alekseev@cryptopro.ru

**Alexandra Babueva**
CryptoPro
Email: babueva@cryptopro.ru

**Lilia Akhmetzyanova**
CryptoPro
Email: lah@cryptopro.ru

**Grigory Marshalko**
TC 26
Email: marshalko_gb@tc26.ru

**Vladimir Rudskoy**
TC 26
Email: rudskoy_vi@tc26.ru

**Alexey Nesterenko**
National Research University Higher School of Economics
Email: anesterenko@hse.ru

**Lidia Nikiforova**
CryptoPro
Email: nikiforova@cryptopro.ru

# Authors' Addresses

**Stanislav Smyshlyaev (EDITOR)**
CryptoPro
Phone: +7 (495) 995-48-20
Email: svs@cryptopro.ru

**Vladislav Nozdrunov**
TC 26
Email: nozdrunov_vi@tc26.ru

**Vasily Shishkin**
TC 26
Email: shishkin_va@tc26.ru

**Ekaterina Griboedova**
CryptoPro
Email: griboedovaekaterina@gmail.com