Network Working Group S. Smyshlyaev, Ed.
Internet-Draft CryptoPro
Intended status: Informational V. Nozdrunov
Expires: April 22, 2019 V. Shishkin
TC 26
October 19, 2018

Multilinear Galois Mode (MGM)
draft-smyshlyaev-mgm-09

Abstract

Multilinear Galois Mode (MGM) is an authenticated encryption with associated data block cipher mode based on EtM principle. MGM is defined for use with 64-bit and 128-bit block ciphers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Multilinear Galois Mode (MGM) is an authenticated encryption with associated data block cipher mode based on EtM principle. MGM is defined for use with 64-bit and 128-bit block. The MGM design principles can easily be applied to other block sizes.

1.1. Existing Constructions

The text will be added in the future versions of the draft.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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;

|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: a^s = (a, a, ... , a), 'a' in V_1;

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

Z_{2^s} ring of residues modulo 2^s;

MSB_i: V_s -> V_i   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}   the transformation that maps a string X = (x_{s-1}, ... , x_0) in V_s 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)   multiplication in 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 = x^64 + x^4 + x^3 + x + 1; if n = 128, then the field polynomial is equal to f = x^128 + x^7 + x^2 + x + 1;

incr_l: V_n -> V_n the transformation that maps a string L || R, where L, R in V_{n/2}, into the string incr_l(L || R) = Vec_{n/2}(Int_{n/2}(L) [+] 1) || R;

incr_r: V_n -> V_n the transformation that maps a string L || R, where L, R in V_{n/2}, into the string incr_r(L || R) = L || Vec_{n/2}(Int_{n/2}(R) [+] 1).

4. Specification

An additional parameter that defines the functioning of MGM mode is the size S of the authentication field (in bits). The value of S MUST be fixed for a particular protocol, 32 <= S <= 128. The choice of the value S involves a trade-off between message expansion and the probability that an attacker can modify a message undetectably.

4.1. MGM Encryption and Authentication Procedure

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

1. Encryption key K in V_k.

2. Initial counter nonce ICN in V_{n-1}.

3. Plaintext P, 0 <= |P| < 2^{n/2}. If |P| > 0, then P = P_1 || ... || P*_q, P_i in V_n, i = 1, ... , q - 1, P*_q in V_u, 1 <= u <= n. If |P| = 0, then by definition P*_q is empty, q = 0, and u = n.

4. Associated authenticated data A, 0 <= |A| < 2^{n/2}. If |A| > 0, then A = A_1 || ... || A*_h, A_j in V_n, j = 1, ... , h - 1, A*_h in V_t, 1 <= t <= n. If |A| = 0, then by definition A*_h is empty, h = 0, and t = n. The associated data is authenticated but is not encrypted.

The MGM encryption and authentication 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 authentication procedure consists of the following steps:

```
+-----------------------------------------------------------------+
|  MGM-Encrypt(K, ICN, P, A)                                      |
|-----------------------------------------------------------------|
|  1. Encryption step:                                            |
|      - 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,                                                 |
|      - 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. Using the same ICN values for two different messages encrypted with the same key eliminates the security properties of this mode.

Users who do not wish to encrypt plaintext can provide a string P of length zero. Users who do not wish to authenticate associated data can provide a string A of length zero. 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 Authentication Check Procedure

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

1. The encryption key K in V_k.

2. The initial counter nonce ICN in V_{n-1}.

3. The associated authenticated data A, 0 <= |A| < 2^{n/2}. A = A_1 || ... || A*_h, A_j in V_n, j = 1, ... , h - 1, A*_h in V_t, 1 <= t <= n.

4. The ciphertext C, 0 <= |C| < 2^{n/2}. C = C_1 || ... || C*_q, C_i in V_n, i = 1, ... , q - 1, C*_q in V_u, 1 <= u <= n.

5. The authenticated tag T in V_S.

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

1. Plaintext P in V_{|C|}.

2. Associated authenticated data A.

The MGM decryption and authentication 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' generation step:                      |
|      - Z_1 = E_K(1^1 || ICN),                                   |
|      - sum1 = 0, sum2 = 0,                                      |
|      - For i = 1, 2, ..., h do                                  |
|            H_i = E_K(Z_i),                                      |
|            sum1 = sum1 (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}),                              |
|            sum2 = sum2 (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(sum1 (xor) sum2 (xor)                     |
|                     H_{h+q+1} (x) (len(A) || len(C)))),         |
|      - If T' != T then return FAIL                              |
|           return FAIL.                                          |
|                                                                 |
|  3. Decryption step:                                            |
|      - 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 (P, A).                                              |
|-----------------------------------------------------------------+
```

5. Rationale

The MGM mode was originally proposed in [PDMODE].

The MGM mode is designed to be fast, parallelizable, inverse free, online and secure.

The MGM is based on counters for the reasons of performance. The first counter (Y_i, see Section 4.1) is used for message encryption, the second counter (H_i, see Section 4.1) is used for authentication. The second counter is encrypted eliminating the chance of obtaining

any information about the H_k value in case when the H_l value is known to the adversary (here l is not equal to k).

To provide parallelizable authentication a multilinear function is used.

To avoid attacks based on padding and linear properties of multilinear function the lengths of associated data A, encrypted message C, and encrypting authentication tag is added.

A collision of "usual" counters leads to obtaining the information about the H_i values and possible authentication vulnerabilities. To minimize the probability of this event we change the principle of counters operating by using the functions incr_l and incr_r. To counteract finding collisions we encrypt initial values of both counters.

6. References

6.1. 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>.

6.2. Informative References

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

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

Appendix A. Test Vectors

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

```
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

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
```

Appendix B. Contributors

- Evgeny Alekseev
  CryptoPro
  alekseev@cryptopro.ru

- Ekaterina Smyshlyaeva
  CryptoPro
  ess@cryptopro.ru

- Lilia Ahmetzyanova
  CryptoPro
  lah@cryptopro.ru

- Grigory Marshalko
  TC 26
  marshalko_gb@tc26.ru

- Vladimir Rudskoy
  TC 26
  rudskoy_vi@tc26.ru

- Alexey Nesterenko
  National Research University Higher School of Economics
  anesterenko@hse.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