Multilinear Galois Mode (MGM)  
draft-smyshlyaev-mgm-14

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

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 April 24, 2020.

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

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.

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 \rightarrow 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 \rightarrow 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 bit length S of the authentication tag, 32 <= S <= 128. 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 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, 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.
4. 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.

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:

\[ \text{MGM-Encrypt}(K, ICN, P, A) \]

1. Encryption step:
   - \( Y_1 = E_K(0 \mid ICN) \),
   - For \( i = 2, 3, \ldots, q \) do
     - \( Y_i = \text{incr}_r(Y_{i-1}) \),
   - For \( i = 1, 2, \ldots, q - 1 \) do
     - \( C_i = P_i \oplus E_K(Y_i) \),
   - \( C_q = P_q \oplus \text{MSB}_u(E_K(Y_q)) \),
   - \( C = C_1 \mid \ldots \mid C_q \).

2. Padding step:
   - \( A_h = A*_h \mid 0^{n-t} \),
   - \( C_q = C*_q \mid 0^{n-u} \).

3. Authentication tag \( T \) generation step:
   - \( Z_1 = E_K(1 \mid ICN) \),
   - \( \text{sum} = 0 \),
   - For \( i = 1, 2, \ldots, h \) do
     - \( H_i = E_K(Z_i) \),
     - \( \text{sum} = \text{sum} \oplus (H_i \oplus A_i) \),
     - \( Z_{i+1} = \text{incr}_l(Z_i) \),
   - For \( j = 1, 2, \ldots, q \) do
     - \( H_{h+j} = E_K(Z_{h+j}) \),
     - \( \text{sum} = \text{sum} \oplus (H_{h+j} \oplus C_j) \),
     - \( Z_{h+j+1} = \text{incr}_l(Z_{h+j}) \),
   - \( H_{h+q+1} = E_K(Z_{h+q+1}) \),
   - \( T = \text{MSB}_S(E_K(\text{sum} \oplus H_{h+q+1} \mid \text{len}(A) \mid \text{len}(C))) \).

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 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 \( A, 0 \leq |A| < 2^{(n/2)}. A = A_1 \ || \ldots \ || A^*_h, A^*_j \) in \( V_n \), for \( j = 1, \ldots, h - 1 \), \( A^*_h \) in \( V_t \), \( 1 \leq t \leq n \).

4. The ciphertext \( C, 0 \leq |C| < 2^{(n/2)}. C = C_1 \ || \ldots \ || C^*_q, C^*_i \) in \( V_n \), for \( i = 1, \ldots, q - 1 \), \( C^*_q \) in \( V_u \), \( 1 \leq u \leq n \).

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 verification step:
   - \( Z_1 = E_K(1 || ICN), \)
   - \( \text{sum} = 0, \)
   - For \( i = 1, 2, \ldots, h \) do
     \[ H_i = E_K(Z_i), \]
     \[ \text{sum} = \text{sum} \oplus ( H_i (x) A_i ), \]
     \[ Z_{i+1} = \text{incr}_a(Z_i), \]
   - For \( j = 1, 2, \ldots, q \) do
     \[ H_{h+j} = E_K(Z_{h+j}), \]
     \[ \text{sum} = \text{sum} \oplus ( H_{h+j} (x) C_j ), \]
     \[ Z_{h+j+1} = \text{incr}_a(Z_{h+j}), \]
   - \( H_{h+q+1} = E_K(Z_{h+q+1}), \)
   - \( T' = \text{MSB}_s(E_K(\text{sum} \oplus H_{h+q+1} (x) (\text{len}(A) || \text{len}(C)))), \)
   - If \( T' \neq T \) then return FAIL.

3. Decryption step:
   - \( Y_1 = E_K(0 || ICN), \)
   - For \( i = 2, 3, \ldots, q \) do
     \[ Y_i = \text{incr}_r(Y_{i-1}), \]
   - For \( i = 1, 2, \ldots, q-1 \) do
     \[ P_i = C_i (x) E_K(Y_i), \]
   - \( P*_q = C*_q (x) \text{MSB}_u(E_K(Y_q)), \)
   - \( P = P_1 || \ldots || P*_q. \)

4. Return \( (P, A) \).

5. Rationale

The MGM mode was originally proposed in [PDMODE].

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

Parallelizability of the MGM mode 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 to process message even if it is not completely received (so its length is unknown). To provide this property the MGM mode uses blocks $E_K(Y_i)$ and $H_i$ which are produced basing on two independent source blocks $Y_i$ and $Z_i$.

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

6. Security Considerations

The security properties of the MGM mode are based on the following:

- Different functions generating the counter values:
  The functions $\text{incr}_r$ and $\text{incr}_l$ are chosen to minimize intersection (if it happens) of counter values $Y_i$ and $Z_i$.

- Encryption of the multilinear function output:
  It allows to resist attacks based on padding and linear properties (see [Ferg05] for details).

- Multilinear function for authentication:
  It allows to resist the small subgroup attacks [Saar12].

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

7. IANA Considerations

This document does not require any IANA actions.
8. References

8.1. Normative References


8.2. Informative References


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 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
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: 4F 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 02 02 01 01 01 01 01 01
00010: 04 04 04 04 04 04 04 04 04 03 03 03 03 03 03 03
00020: EA 05 05 05 05 05 05 05 05 00 00 00 00 00 00 00

3. Authentication tag T generation step:

1^1 || ICN:

Z_1:

H_1:

current sum:

Z_2:

H_2:

current sum:

Z_3:

H_3:

current sum:

Z_4:

H_4:

current sum:

Z_5:

H_5:

current sum:

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 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

- Alexandra Babueva
  CryptoPro
  babueva@cryptopro.ru

- Lilia Akhmetzyanova
  CryptoPro
  lah@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 Smyshlyaeva
CryptoPro
Email: ess@cryptopro.ru