Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 2, 2018

S. Smyshlyaev, Ed.
CryptoPro
V. Nozdrunov
V. Shishkin
TC26

March 1, 2018

Multiline Galois Mode (MGM) Specification
draft-smyshlyaev-mgm-01

Abstract

Multiline 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.

Table of Contents

1. Introduction

Multiline 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 and other block cipher.

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;

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

Additional parameter that define the functioning of MGM mode is the the size S of the authentication field (in bits). The value of S MUST be such that 32 <= S <= 128 The choice of the value S involves a trade-off between message expansion and the probability that an attacker can undetectably modify a message.

4.1. MGM Encryption and Authentication Procedure

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

1. Encryption key K in V_k.

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

3. Plaintext P, 0 <= |P| < 2^{n/2}. P = P_1 || ... || P*_q, P_i in V_n, i = 1, ... , q - 1, P*_q in V_u, 1 <= u <= n.

4. 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. 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),                                   |
|      - 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)))).        |
|                                                                 |
|  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 destroys 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 as inputs the following parameters:

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-Encrypt(K, ICN, P, A)                                      |
|-----------------------------------------------------------------|
|  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

During the construction of MGM mode our task was to create fast, paralleziable, inverse free, online and secure block cipher mode. It is well known that one of the fastest mode for encryption is CTR. That's why we developed MGM mode based on counters. The first counter is used for message encryption, the second counter is used for authentication.

For providing parallelize authentication we use multilinear function. By encrypting second counter we produce elements H_i with the

property that if you know any information about value H_k you can't obtain any information about value H_l (l not equal k) besides that H_k not equal H_l.

By adding the length of associated data A and encrypted message C and encrypting authentication tag we avoid attacks based on padding and linear properties of multilinear function.

Collision of "usual" counters lead to obtaining information about values H_i, that could be dangerous to authentication. For minimizing probability of this event we change the principle of counters operating by functions incr_l and incr_l. To avoid a theoretical ability to calculate a point of counters collision we encrypt the initialization value of each counter.

6. 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>.

Appendix A. Contributors

o Evgeny Alekseev
  CryptoPro
  alekseev@cryptopro.ru

o Ekaterina Smyshlyaeva
  CryptoPro
  ess@cryptopro.ru

o Lilia Ahmetzyanova
  CryptoPro
  lah@cryptopro.ru

o Grigory Marshalko
  TK26
  marshalko_gb@tc26.ru

Authors' Addresses

Stanislav Smyshlyaev (editor)
CryptoPro

Phone: +7 (495) 995-48-20
Email: svs@cryptopro.ru

Vladislav Nozdrunov
TC26

Email: nozdrunov_vi@tc26.ru

Vasily Shishkin
TC26

Email: shishkin_va@tc26.ru