Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2017

S. Smyshlyaev, Ed.
E. Alekseev
I. Oshkin
L. Ahmetzyanova
E. Smyshlyaeva
CryptoPro

October 21, 2016

The ACPKM internal re-keying mechanism for block cipher modes of operation
draft-smyshlyaev-re-keying-00

Abstract

This specification presents an approach to increase the security of block cipher operation modes based on re-keying (with no additional keys needed) during each separate message processing. It provides an internal re-keying mechanism called ACPKM. This mechanism doesn’t require additional secret parameters or complicated transforms - for key update only the base encryption function is used.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

An important problem related to secure functioning of any cryptographic system is the control of key lifetimes. Regarding symmetric keys, the main concern is constraining the key exposure. It could be done by limiting the maximal amount of data processed with one key. The restrictions can come either from combinatorial properties of the used cipher modes of operation (for example, birthday attack [BDJR]) or from particular cryptographic attacks on the used block cipher (for example, linear cryptanalysis [Matsui]). Moreover, most strict restrictions here follow from the need to resist side-channel attacks. The adversary’s opportunity to obtain an essential amount of data processed with a single key leads not only to theoretic but also to real vulnerabilities (see [BL]). Therefore, when the total size of a plaintext processed with the same key reaches threshold values, this key cannot be used anymore and certain procedures on encryption keys are needed. It leads to several operating limitations, e.g. the impossibility to process long messages and processing overhead caused by derivation of additional keys.

This specification presents a mechanism to increase the key lifetime, which is called ACPKM. This solution ("key meshing") transforms the key value each time when the given amount of data, precisely the amount of plaintext section (not the total amount of separate messages), is processed and proceeds with a new transformed key value for a new plaintext section. Such a transformation does not require any additional secret values. It is integrated into the base mode of operation and can be considered as it’s extension, therefore it is called "internal re-keying" in this document.

This approach seems to be mostly useful in the case when the total amount of data for an established key is not known beforehand: the performance on useless operations won’t be lost if the data size is rather small, and the security won’t be lacked when it occurs to be large. The transformed keys are computed only when they are needed.

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:

(xor)   exclusive-or of two binary vectors of the same length.

V*   the set of all strings of a finite length (hereinafter referred to as 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 one;

|X| the bit length of the bit string X;

A|B   concatenation of strings A and B both belonging to V*, i.e., a string in V_{|A|+|B|}, where the left substring in V_|A| is equal to A, and the right substring in V_|B| is equal to B;

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

Int_s: V_s -> Z_{2^s}   the transformation that maps a string a = (a_s, ...,a_1), a in V_s, into the integer Int_s(a) = 2^s*a_s + ... + 2*a_2 + a_1;

Vec_s: Z_{2^s} -> V_s the transformation inverse to the mapping Int_s;

MSB_i: V_s -> V_i the transformation that maps the string a = (a_s, ...,a_1) in V_s, into the string MSB_i(a) = (a_s, ...,a_{s-i+1}) in V_i;

LSB_i: V_s -> V_i the transformation that maps the string a = (a_s, ...,a_1) in V_s, into the string LSB_i(a) = (a_i, ...,a_1) in V_i;

Inc_c: V_s -> V_s the transformation that maps the string a = (a_s, ...,a_1) in V_s, into the string Inc_c(a) = MSB_{|a|-c}(a) | Vec_c(Int_c(LSB_c(a)) + 1(mod 2^c)) in V_s;

0^s   denotes the string a in V_s that consists of s '0' bits;

E_K: V_n -> V_n the block cipher permutation under the key K in V_k;

k   the key K size (in bits);

n   the block size of the block cipher (in bits);

b   the total number of data blocks in the plaintext;

N   the section size (the number of bits in a data section);

l   the number of data sections in the plaintext;

m   the message M size (in bits);

phi_i: V_s -> V_s the transformation that maps a string a = (a_s, ...,a_1) into the string phi_i(a) = a' = (a'_s, ...,a'_1), 1 <= i <= s, such that a'_i = 1 and a'_j = a_j for all j in {1,...,s}/{i};

ceil(x) the least integer that is not less than x.

4. CTR and GCM Block Cipher Modes

This section describes the families of block cipher modes of operations that are extended by the ACPKM re-keying mechanisms as described in Section 5.

A plaintext message P and a ciphertext C are divided into b = ceil(m/n) parts (denoted as P = P_1 | P_2 |...| P_b and C = C_1 | C_2 |...| C_b, where P_i and C_i are in V_n, for i = 1, 2, ..., b-1, and P_b, C_b are in V_r, where r <= n).

4.1. CTR Block Cipher Mode

The Counter (CTR) mode is a block cipher mode of operation that applies the block cipher transformation E_K to a sequence of input blocks, called counters, to produce a sequence of output blocks that are XORed with a plaintext to produce a ciphertext, and vice versa. It is defined similar to the one specified in [NIST-CTR].

The ACPKM-CTR re-keying mechanisms described in Section 5.1 can be used with the following block cipher and CTR mode parameters:

- 64 <= n <= 512;

- 128 <= k <= 512;

- the number of bits c in a specific part of the block to be incremented is such that 32 <= c <= 3/4 n.

In the current document, the counters for a given message are denoted as CTR_1, CTR_2, ..., CTR_b.

The CTR encryption mode is defined as follows:

```
Input:
    Initial counter nonce ICN in V_{n-c},
    plaintext P, |P| < n*2^c.

Output:
    Ciphertext C.
-------------------------------------------------------------------
CTR Encryption:
    1. CTR_1 = ICN | 0^c.
    2. For j = 1, 2, ..., b-1 do
           CTR_{j+1} = Inc_c(CTR_j).
    3. For j = 1, 2, ..., b do
           G_j = E_K(CTR_j).
    4. C = P (xor) MSB_{|P|}(G_1 |...|G_b).
    5. Return C.
```

The CTR decryption mode is defined as follows:

```
Input:
    Initial counter nonce ICN in V_{n-c},
    ciphertext C, |C| < n*2^c.

Output:
    Plaintext P.

CTR Decryption:
    1. CTR_1 = ICN | 0^c.
    2. For j = 1, 2,..., b-1 do
           CTR_{j+1} = Inc_c(CTR_j).
    3. For j = 1, 2, ..., b do
           G_j = E_K(CTR_j)
    4. P = C (xor) MSB_{|C|}(G_1 |...|G_b).
    5. Return P.
```

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

4.2. GCM Block Cipher Mode

TODO: This section describes the family of block cipher modes of operation with both encryption and authentication. It is defined similar to the one specified in [NIST-GCM].

The ACPKM-GCM re-keying mechanisms described in Section 5.2 can be used with the following GCM block cipher mode parameters:

- 128 <= n <= 512;

- 128 <= k <= 512;

- the number of bits c in a specific part of the block to be incremented is such that 32 <= c <= 3/4 n.

4.2.1. GCM Subprocedures

This section presents three mathematical algorithms that appear in the specification of the authenticated encryption and authenticated decryption functions of the GCM cipher mode described in Section 4.2.2 below.

4.2.1.1. Multiplication Operation on Blocks

The * operation on (pairs of) the 2^n possible blocks corresponds to the multiplication operation for the binary Galois (finite) field of 2^n elements and is defined by a particular GCM mode.

4.2.1.2. GHASH Function

```
Algorithm 2: GHASH_H(X)
=======================
Input:
    Bit string X = X_1 |...| X_m, where X_i in V_n for i in 1,...,m.
Output:
    Block GHASHH (X) in V_n
___________________________________________________________________
1. Y_0 = 0^n.
2. For i = 1,..., m do
       Y_i = (Y_{i-1} (xor) X_i)*H.
3. Return Y_m.
```

4.2.1.3. GCTR Function

```
Algorithm 3: GCTR_K(ICB, X)
===========================
Input:
    Initial counter block ICB;
    X = X_1 |...| X_t, X_i in V_n for i = 1,...,n-1 and X_n in V_r,
    where r <= n.
Output:
    Y in V_{|X|}.
___________________________________________________________________
1. If X in V_0 then return Y, where Y in V_0.
2. t = ceil(|X|/n).
3. GCTR_1 = ICB.
4. For i = 2,...,t do
       GCTR_i = Inc_c(GCTR_{i-1}).
5. For i = 1,...,t do
       G_i = E_K(GCTR_i).
6. Y = X (xor) MSB_{|X|}(G_1 |...| G_t).
7. Return Y.
```

4.2.2. GCM Mode Description

The GCM encryption mode is defined as follows:

```
   Input:
       Initialization vector IV in V_{n-c},
       plaintext P, |P| < n*(2^c - 2).
       additional authenticated data A.
   Output:
       Ciphertext C,
       authentication tag T.
   _________________________________________________________________
   GCM Encryption:
       1. H = E_K(0^n).
       2. if c = 32, then J_0 = IV | 0^31 | 1;
          if c!= 32, then s = n*ceil(|IV|/n)-|IV|,
                          J_0 = GHASH_H(IV | 0^{s+n-64} | Vec_64(|IV|)).
       3. C = GCTR_K(Inc_32(J_0),P).
       4. u = n*ceil(|C|/n)-|C|,
          v = n*ceil(|A|/n)-|A|.
       5. S = GHASH_H(A | 0^v | C | 0^u | 0^{n-128} |
                     |Vec_64(|A|) | Vec_64(|C|)).
       6. T = MSB_t(E_K(J_0) (xor) S).
       7. Return C | T.
```

The GCM decryption mode is defined as follows:

```
   Input:
       Initialization vector IV in V_{n-c},
       ciphertext C, |C| < n*(2^c - 2),
       authentication tag T,
       additional authenticated data A.
   Output:
       Plaintext P or FAIL.
   _________________________________________________________________
   GCM decryption:
       1. H = E_K(0^n).
       2. if c = 32, then J_0 = IV | 0^31 | 1;
          if c!= 32, then s = n*ceil(|IV|/n)-|IV|,
                          J_0 = GHASH_H(IV | 0^{s+n-64} | Vec_64(|IV|)).
       3. P = GCTR_K(Inc_32(J_0),C).
       4. u = n*ceil(|C|/n)-|C|,
          v = n*ceil(|A|/n)-|A|.
       5. S = GHASH_H(A | 0^v | C | 0^u | 0^{n-128}|
                    |Vec_64(|A|) | Vec_64(|C|)).
       6. T’ = MSB_t(E_K(J_0) (xor) S).
       7. IF T=T’ then return P; else return FAIL.
```

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

Note: The encryption part in the GCM-ACPKM mode is the encryption in the CTR-ACPKM mode with several differences: in the CTR mode the counter for the plaintext encryption starts with the first CTR_1 value and in the GCM mode the counter starts with the second GCTR_2 value.

5. ACPKM re-keying mechanisms

This section defines periodical key transformations for long message processing that are considered as extensions of the basic CTR and GCM encryption modes and are called ACPKM-CTR and ACPKM-GCM re-keying mechanisms.

An additional parameter that defines the functioning of CTR and GCM block cipher modes with the ACPKM key transformation algorithm is the section size N. The value of N is fixed within a specific protocol based on the requirements of the system capacity and key lifetime (some recommendations on choosing N will be provided in Section 7). The section size N MUST be divisible by the block size n.

The main idea behind internal re-keying is presented in Fig.1:

Lifetime of a key = L,
section size = const = N,
maximum message size = m_max.

```
                    ACPKM       ACPKM                ACPKM
            K_1 = K ----> K_2 -----...-> K_{l_max-1} ----> K_{l_max}
Message(1) |==========|==========|======|===       |       :  |
           |          |          | ...  |          |       :  |
Message(2) |==========|==========|======|==========|=======:  |
 . . .     |          |          | ...  |          |       :  |
           |          |          |      |          |       :  |
Message(q) |==========|==========|======|==========|=====  :  |
           |          | section  | ...  |          |       :  |
                      <---------->                         :
                         N bit                           m_max
```

l_max = ceil(m_max/N),
q*N <= L.

Figure 1: Key meshing approach

For the {i+1}-th section the K_{i+1} value is calculated as follows:

K_{i+1} = ACPKM-CTR(K_i) = MSB_k(E_{K_i}(W_1)|...|E_{K_i}(W_J)),

where J = ceil(k/n), W_t = phi_c(D_t) for any t in {1,...,J} and D_1, D_2,...,D_J are in V_n and are calculated as follows:

```
   D_1 | D_2 |...| D_J = MSB_{J*n}(D),
```

where D is the following constant in V_1024:

```
   D = ( F3 | 74 | E9 | 23 | FE | AA | D6 | DD
       | 98 | B4 | B6 | 3D | 57 | 8B | 35 | AC
       | A9 | 0F | D7 | 31 | E4 | 1D | 64 | 5E
       | 40 | 8C | 87 | 87 | 28 | CC | 76 | 90
       | 37 | 76 | 49 | 9F | 7D | F3 | 3B | 06
       | 92 | 21 | 7B | 06 | 37 | BA | 9F | B4
       | F2 | 71 | 90 | 3F | 3C | F6 | FD | 1D
       | 70 | BB | BB | 88 | E7 | F4 | 1B | 76
       | 7E | 44 | F9 | 0E | 46 | 91 | 5B | 57
       | 00 | BC | 13 | 45 | BE | 0D | BD | C7
       | 61 | 38 | 19 | 3C | 41 | 30 | 86 | 82
       | 1A | A0 | 45 | 79 | 23 | 4C | 4C | F3
       | 64 | F2 | 6A | CC | EA | 48 | CB | B4
       | 0C | B9 | A9 | 28 | C3 | B9 | 65 | CD
       | 9A | CA | 60 | FB | 9C | A4 | 62 | C7
       | 22 | C0 | 6C | E2 | 4A | C7 | FB | 5B).
```

Note : The constant D is such that phi_c(D_1),..., phi_c(D_J) are pairwise different for any allowed n, k, c values.

5.1. ACPKM internal re-keying mechanism for CTR encryption mode

This section defines a ACPKM-CTR internal re-keying mechanism for the CTR encryption mode that was described in Section 4.1.

During the processing of the input message M with the length m using ACPKM-CTR internal re-keying algorithm and the key K the message is divided into l = ceil(m*N) parts (denoted as M = M_1 | M_2 |...| M_l, where M_i is in V_N for i = 1, 2,..., l-1 and M_l is in V_r, r <= N). The first section is processed with the initial key K_1 = K. To process the (i+1)-th section the K_{i+1} key value is calculated using ACPKM-CTR transformation of the key K_i. The counter value (CTR_{i+1}) is not changed during this process.

The message size m MUST NOT exceed n*2^{c-1} bits.

5.2. ACPKM internal re-keying mechanism for GCM encryption mode

This section defines a ACPKM-GCM internal re-keying mechanism for the GCM encryption mode that was described in Section 4.2.

During the processing of the input message M with the length m using ACPKM-GCM internal re-keying algorithm and the key K the message is divided into l = ceil(m/N) parts (denoted as M = M_1 | M_2 |...| M_l, where M_i is in V_N for i = 1, 2,..., l-1 and M_l is in V_r, r <= N). The first section is processed with the initial key K_1 = K. To process the (i+1)-th section the K_{i+1} key value is calculated using ACPKM-GCM transformation of the key K_i.

The message size m MUST NOT exceed n*(2^{c-1}-2) bits.

The key for computing values E_K(J_0) and H is not updated and is equal to the initial key.

6. Acknowledgments

TODO

7. Security Considerations

The ACPKM re-keying mechanisms provide the CTR and GCM encryption modes extensions that have the following property: a compromise of a key of some section does not lead to a compromise of previous keys but leads to a compromise of next keys.

The ACPKM mechanism allows to increase the CTR and GCM encryption modes security in proportion to the frequency of key changing, which is inversely related to the section size N. Thus, the key lifetime can be noticeably increased: an amount of material that is processed with the key K increases quadratically, divided by N.

Since the performance of encryption can slightly decrease for rather small values of N, the parameter of N SHOULD be selected for a particular protocol as maximum possible to provide necessary key lifetime for the adversary models that are considered.

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, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[BDJR] Bellare M., Desai A., Jokipii E., Rogaway P., "A concrete security treatment of symmetric encryption", In Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS ’97), pages 394-403. 97, 1997.

[BL] Bhargavan K., Leurent G., "On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN", Cryptology ePrint Archive Report 798, 2016.

[Matsui] Matsui M., "Linear Cryptanalysis Method for DES Cipher", Advanced in Cryptology- EUROCRYPT’93. Lect. Notes in Comp. Sci., Springer. V.765.P. 386-397, 1994.

[NIST-CTR] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST Special Publication 800-38A, December 2001.

[NIST-GCM] McGrew, D. and J. Viega, "The Galois/Counter Mode of Operation (GCM)", Submission to NIST http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf, January 2004.

Appendix A. Test examples

TODO

Authors’ Addresses

Stanislav Smyshlyaev (editor)
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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

Evgeny Alekseev
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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

Igor Oshkin
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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

Lilia Ahmetzyanova
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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

Ekaterina Smyshlyaeva
CryptoPro
18, Suschevsky val
Moscow 127018
Russian Federation

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