| | |
|---|---|
| CFRG | S. Smyshlyaev, Ed. |
| Internet-Draft | CryptoPro |
| Intended status: Informational | R. Housley |
| Expires: August 24, 2017 | Vigil Security, LLC |
| | M. Bellare |
| | University of California, San Diego |
| | E. Alekseev |
| | E. Smyshlyaeva |
| | CryptoPro |
| | D. Franke |
| | Akamai Technologies |
| | L. Ahmetzyanova |
| | CryptoPro |
| | R. Ng |
| | University of California, San Diego |
| | February 20, 2017 |


# Re-keying Mechanisms for Symmetric Keys

**draft-cfrg-re-keying-00**

## Abstract


This specification contains a description of a variety of methods to increase the lifetime of symmetric keys. It provides external and internal re-keying mechanisms that can be used with such modes of operations as CTR, GCM, CBC, CFB, OFB and OMAC.


## Status of This Memo



## Copyright Notice

## Table of Contents



# 1. Introduction

Common attacks base their success on the ability to get many encryptions under a single key. If encryption is performed under a single key, there is a certain maximum threshold number of messages that can be safely encrypted. These 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 practical vulnerabilities (see [BL]). Therefore, when the total size of a plaintext processed with a single key reaches threshold values, this key cannot be used anymore and certain procedures with encryption keys are needed.

The most simple and obvious way for overcoming the key lifetimes limitations is a renegotiation of a regular session key. However, this reduces the total performance since it usually entails the frequent use of a public key cryptography.

Another way is to use a transformation of a previously negotiated key. This specification presents the description of such mechanisms and the description of the cases when these mechanisms should be applied.

## 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;

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

E_{K}: V_n -> V_n
: the block cipher permutation under the key K in V_k;

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

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 (b = ceil(m/n));

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}.

A plaintext message P and a ciphertext C are divided into b = ceil(m/n) segments 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 if not otherwise stated.

# 4. External Re-keying Mechanisms

This section presents an approach to increase the lifetime of negotiated keys after processing a limited number of integral messages. It provides an external parallel and serial re-keying mechanisms (see [AbBell]). These mechanisms use an initial (negotiated) key as a master key, which is never used directly for the data processing but is used for key generation. Such mechanisms operate outside of the base modes of operations and do not change them at all, therefore they are called "external re-keying" in this document.

## 4.1. Parallel Constructions

The main idea behind external re-keying with parallel construction is presented in Fig.1:

```
Lifetime of a key = L,
maximum message size = m_max.
_____________________________________________________________

                        m_max
                 <---------------->
            M_{1,1}   |===              |
            M_{1,2}   |===============  |
    +--K^1-->   . . .
    |       M_{1,q_1} |========         |
    |
    |
    |       M_{2,1}   |================|
    |       M_{2,2}   |=====            |
K*----|--K^2-->   . . .
    |       M_{2,q_2} |==========       |
    |
   ...
    |       M_{t,1}   |============     |
    |       M_{t,2}   |=============    |
    +--K^t-->   . . .
            M_{t,q_t} |==========       |

_____________________________________________________________
```

```
|M_{i,1}| + ... + |M_{i,q_i}| <= L, i = 1, ... , t.

     Figure 1: External parallel re-keying mechanisms
```

### 4.1.1. Parallel Construction Based on a KDF on a Block Cipher

ExtParallelC re-keying mechanism is based on a block cipher and is used to generate t keys for t sections as follows:

K^1 | K^2 | ... | K^t = ExtParallelC(K*, t*k) = MSB_{t*k}(E_{K*}(0) | E_{K*}(1) | ... | E_{K*}(J-1)),

where J = ceil(k/n).

### 4.1.2. Parallel Construction Based on HKDF

ExtParallelH re-keying mechanism is based on HMAC-based key derivation function HKDF-Expand, described in [RFC5869], and is used to generate t keys for t sections as follows:

K^1 | K^2 | ... | K^t = ExtParallelH(K*, t*k) = HKDF-Expand(K*, label, t*k),

where label is a string (can be a zero-length string) that is defined by a specific protocol.

## 4.2. Serial Constructions

The main idea behind external re-keying with serial construction is presented in Fig.2:

```
Lifetime of a key = L,
maximum message size = m_max.
______________________________________________________________
                                   m_max
                             <---------------->
               M_{1,1}   |===              |
               M_{1,2}   |=============== |
K*_1 = K* ---K^1-->   . . .
  |            M_{1,q_1} |========         |
  |
  |
  |            M_{2,1}   |================|
  v            M_{2,2}   |=====            |
K*_2 --------K^2-->   . . .
  |            M_{2,q_2} |==========       |
  |
 ...
  |            M_{t,1}   |============     |
  v            M_{t,2}   |=============    |
K*_t --------K^t-->   . . .
               M_{t,q_t} |==========       |


______________________________________________________________
|M_{i,1}| + ... + |M_{i,q_i}| <= L, i = 1, ... , t.
```

Figure 2: External serial re-keying mechanisms

### 4.2.1. Serial Construction Based on a KDF on a Block Cipher

The key K^i is calculated using ExtSerialC transformation as follows:

K^i = ExtSerialC(K*, i) = MSB_k(E_{K*_i}(0) | E_{K*_i}(1) | ... | E_{K*_i}(J-1)),

where J = ceil(k/n), i = 1, ... , t, K*_i is calculated as follows:

K*_1 = K*,

K*_{j+1} = MSB_k(E_{K*_j}(J) | E_{K*_j}(J+1) | ... | E_{K*_j}(2J-1)),

where j = 1, ... , t-1.

### 4.2.2. Serial Construction Based on HKDF

The key K^i is calculated using ExtSerialH transformation as follows:

K^i = ExtSerialH(K*, i) = HKDF-Expand(K*_i, label1, k),

where i = 1, ... , t, HKDF-Expand is an HMAC-based key derivation function, described in [RFC5869], K*_i is calculated as follows:

K*_1 = K*,

K*_{j+1} = HKDF-Expand(K*_j, label2, k), where j = 1, ... , t-1,

where label1 and label2 are different strings (can be a zero-length strings) that are defined by a specific protocol (see, for example, TLS 1.3 updating traffic keys algorithm [TLSDraft]).

## 5. Internal Re-keying Mechanisms

This section presents an approach to increase the lifetime of negotiated key by re-keying during each separate message processing. It provides an internal re-keying mechanisms called ACPKM and ACPKM-Master that do not use and use a master key respectively. Such mechanisms are integrated into the base modes of operations and can be considered as the base mode extensions, therefore they are called "internal re-keying" in this document.

### 5.1. Constructions that Do Not Require Master Key

This section describes the block cipher modes that uses the ACPKM re-keying mechanism (described in Section 5.1.1), which does not use master key: an initial key is used directly for the encryption of the data.

### 5.1.1. ACPKM Re-keying Mechanisms

This section defines periodical key transformation with no master key which is called ACPKM re-keying mechanism. This mechanism can be applied to one of the basic encryption modes (CTR and GCM block cipher modes) for getting an extension of this encryption mode that uses periodical key transformation with no master key. This extension can be considered as a new encryption mode.

An additional parameter that defines the functioning of basic encryption modes with the ACPKM re-keying mechanism is the section size N. The value of N is measured in bits and 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 with no master key is presented in Fig.3:

```
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}
              |         |              |          |
              |         |              |          |
              v         v              v          v
Message(1) |==========|==========| ... |==========|=======:  |
Message(2) |==========|==========| ... |===       |       :  |
   .     .         .         .    .         .       :  .
   :     :         :         :    :         :       :  :
Message(q) |==========|==========| ... |==========|=====  :  |
                    section                          :
                  <---------->                     m_max
                     N bit
______________________________________________________________________
l_max = ceil(m_max/N),
q*N <= L.
                Figure 3: Key meshing with no master key
```

During the processing of the input message M with the length m in some encryption mode that uses ACPKM key transformation of 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 transformation as follows:

K^{i+1} = ACPKM(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).
```

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

N o t e : The constant D is such that D = sha512(streebog512(0^1024)) | sha512(streebog512(1^1024)), where sha512 is a hash function with 512-bit output corresponding to the algorithm SHA-512 [SHA-512], streebog512 is a hash function with 512-bit output, corresponding to the algorithm GOST R 34.11-2012 [GOST3411-2012], [RFC6986].

## 5.1.2. CTR-ACPKM Encryption Mode

This section defines a CTR-ACPKM encryption mode that uses internal ACPKM re-keying mechanism for the periodical key transformation.

The CTR-ACPKM mode can be considered as the extended by the ACPKM re-keying mechanism basic encryption mode CTR (see [MODES]).

The CTR-ACPKM encryption mode can be used with the following 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.

The CTR-ACPKM mode encryption and decryption procedures are defined as follows:

```
+----------------------------------------------------------------+
|  CTR-ACPKM-Encrypt(N, K, ICN, P)                               |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - key K,                                                      |
|  - initial counter nonce ICN in V_{n-c},                       |
|  - plaintext P = P_1 | ... | P_b, |P| < n * 2^{c-1}.           |
|  Output:                                                       |
|  - Ciphertext C.                                               |
|----------------------------------------------------------------|
|  1. CTR_1 = ICN | 0^c                                          |
|  2. For j = 2, 3, ... , b do                                   |
|         CTR_{j} = Inc_c(CTR_{j-1})                             |
|  3. K^1 = K                                                    |
|  4. For i = 2, 3, ... , ceil(|P|/N)                            |
|         K^i = ACPKM(K^{i-1})                                   |
|  5. For j = 1, 2, ... , b do                                   |
|         i = ceil(j*n / N),                                     |
|         G_j = E_{K^i}(CTR_j)                                   |
|  6. C = P (xor) MSB_{|P|}(G_1 | ... | G_b)                     |
|  7. Return C                                                   |
+----------------------------------------------------------------+
```

```
+----------------------------------------------------------------+
|  CTR-ACPKM-Decrypt(N, K, ICN, C)                               |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - key K,                                                      |
|  - initial counter nonce ICN in V_{n-c},                       |
|  - ciphertext C = C_1 | ... | C_b, |C| < n * 2^{c-1}.          |
|  Output:                                                       |
|  - Plaintext P.                                                |
|----------------------------------------------------------------|
|  1. Return CTR-ACPKM-Encrypt(N, K, ICN, C)                     |
+----------------------------------------------------------------+
```

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

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

### 5.1.3. GCM-ACPKM Encryption Mode

This section defines a GCM-ACPKM encryption mode that uses internal ACPKM re-keying mechanism for the periodical key transformation.

The GCM-ACPKM mode can be considered as the extended by the ACPKM re-keying mechanism basic encryption mode GCM (see [GCM]).

The GCM-ACPKM encryption mode can be used with the following parameters:

- n in {128, 256};
- 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;
- authentication tag length t.

The GCM-ACPKM mode encryption and decryption procedures are defined as follows:

```
+-------------------------------------------------------------------+
|  GHASH(X, H)                                                      |
|-------------------------------------------------------------------|
|  Input:                                                           |
|  - Bit string X = X_1 | ... | X_m, X_i in V_n for i in 1, ... , m. |
|  Output:                                                          |
|  - Block GHASH(X, H) 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                                                    |
+-------------------------------------------------------------------+

+-------------------------------------------------------------------+
|  GCTR(N, K, ICB, X)                                               |
|-------------------------------------------------------------------|
```

```
|  Input:                                                        |
|  - Section size N,                                             |
|  - key K,                                                      |
|  - initial counter block ICB,                                  |
|  - X = X_1 | ... | X_b, X_i in V_n for i = 1, ... , b-1 and    |
|                     X_b 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. GCTR_1 = ICB                                               |
|  3. For i = 2, ... , b do                                      |
|         GCTR_i = Inc_c(GCTR_{i-1})                             |
|  4. K^1 = K                                                    |
|  5. For j = 2, ... , ceil(l*n / N)                             |
|         K^j = ACPKM(K^{j-1})                                   |
|  6. For i = 1, ... , b do                                      |
|         j = ceil(i*n / N),                                     |
|         G_i = E_{K_j}(GCTR_i)                                  |
|  7. Y = X (xor) MSB_{|X|}(G_1 | ... | G_b)                     |
|  8. Return Y.                                                  |
+----------------------------------------------------------------+

+----------------------------------------------------------------+
|  GCM-ACPKM-Encrypt(N, K, IV, P, A)                             |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - key K,                                                      |
|  - initial counter nonce ICN in V_{n-c},                       |
|  - plaintext P, |P| <= n*(2^{c-1} - 2), P = P_1 | ... | P_b,   |
|  - additional authenticated data A.                            |
|  Output:                                                       |
|  - Ciphertext C,                                               |
|  - authentication tag T.                                       |
|----------------------------------------------------------------|
|  1. H = E_{K}(0^n)                                             |
|  2. If c = 32, then ICB_0 = ICN | 0^31 | 1                     |
|     if c!= 32, then s = n * ceil(|ICN| / n) - |ICN|,           |
|            ICB_0 = GHASH(ICN | 0^{s+n-64} | Vec_64(|ICN|), H)  |
|  3. C = GCTR(N, K, Inc_32(ICB_0), P)                           |
|  4. u = n*ceil(|C| / n) - |C|                                  |
|     v = n*ceil(|A| / n) - |A|                                  |
|  5. S = GHASH(A | 0^v | C | 0^u | 0^{n-128} | Vec_64(|A|) |    |
|             | Vec_64(|C|), H)                                  |
|  6. T = MSB_t(E_{K}(ICB_0) (xor) S)                            |
|  7. Return C | T                                               |
+----------------------------------------------------------------+

+----------------------------------------------------------------+
|  GCM-ACPKM-Decrypt(N, K, IV, A, C, T)                          |
|----------------------------------------------------------------|
|  Input:                                                        |
```

```
| - Section size N,                                  |
| - key K,                                           |
| - initial counter block ICB,                       |
| - additional authenticated data A.                 |
| - ciphertext C, |C| <= n*(2^{c-1} - 2), C = C_1 | ... | C_b,  |
| - authentication tag T                             |
| Output:                                            |
| - Plaintext P or FAIL.                             |
|----------------------------------------------------|
| 1. H = E_{K}(0^n)                                  |
| 2. If c = 32, then ICB_0 = ICN | 0^31 | 1          |
|    if c!= 32, then s = n*ceil(|ICN|/n)-|ICN|,      |
|            ICB_0 = GHASH(ICN | 0^{s+n-64} | Vec_64(|ICN|), H)  |
| 3. P = GCTR(N, K, Inc_32(ICB_0), C)                |
| 4. u = n*ceil(|C| / n)-|C|                         |
|    v = n*ceil(|A| / n)-|A|                         |
| 5. S = GHASH(A | 0^v | C | 0^u | 0^{n-128} | Vec_64(|A|) |   |
|           | Vec_64(|C|), H)                         |
| 6. T' = MSB_t(E_{K}(ICB_0) (xor) S)                |
| 7. If T = T' then return P; else return FAIL       |
+----------------------------------------------------+
```

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 defined by the polynomial f as follows (by analogy with [GCM]):

n = 128:
    f = a^128 + a^7 + a^2 + a^1 + 1.

n = 256:
    f = a^256 + a^10 + a^5 + a^2 + 1.

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

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

The key for computing values E_{K}(ICB_0) and H is not updated and is equal to the initial key K.

## 5.2. Constructions that Require Master Key

This section describes the block cipher modes that uses the ACPKM-Master re-keying mechanism (described in Section 5.2.1), which use the initial key K as a master key K*, so K is never used directly for the data processing but is used for key derivation.

### 5.2.1. ACPKM-Master Key Generation from the Master Key

This section defines periodical key transformation with master key K* which is called ACPKM-Master re-keying mechanism. This mechanism can be applied to one of the basic encryption modes (CTR, GCM, CBC, CFB, OFB, OMAC encryption modes) for getting an extension of this encryption mode that uses periodical key transformation with master key. This extension can be considered as a new encryption mode.

Additional parameters that defines the functioning of basic encryption modes with the ACPKM-Master re-keying mechanism are the section size N and change frequency T* of the key K*. The values of N and T* are measured in bits and are fixed within a specific protocol based on the requirements of the system capacity and key lifetime (some recommendations on choosing N and T* will be provided in Section 7). The section size N MUST be divisible by the block size n. The key frequency T* MUST be divisible by n.

The main idea behind internal re-keying with master key is presented in Fig.4:

```
Lifetime of a key = L,
change frequency T*,
section size N,
maximum message size = m_max.
________________________________________________________________________________

                  ACPKM               ACPKM
         K*_1 = K*-------------> K*_2 ---------...---------> K*_l_max
         ___|___               ___|___                   ___|___
        |       |             |       |                 |       |
        v  ...  v             v  ...  v                 v  ...  v
      K[1]     K[t]        K[t+1]   K[2t]     K[(l_max-1)t+1]   K[l_max*t]
        |       |             |       |                 |       |
        |       |             |       |                 |       |
        v       v             v       v                 v       v
Message(1)||========|...|========||========|...|========||...||========|...|==    : ||
Message(2)||========|...|========||========|...|========||...||========|...|======: ||
  ...     ||        |   |        ||        |   |        ||   ||        |   |      : ||
Message(q)||========|...|========||====    |...|        ||...||        |...|      : ||
          section                                                             :
         <-------->                                                           :
            N bit                                                           m_max
________________________________________________________________________________
|K[i]| = d,
t = T*/d,
l_max = ceil(m_max/N),
q*N <= L.
                Figure 4: Key meshing with master key
```

During the processing of the input message M with the length m in some encryption mode that uses ACPKM-Master key transformation with the master key K* and key frequency T* the message M 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 in {1, 2, ... , l-1} and M_l is in V_r, r <= N). The j-th section is processed with the key material K[j], j in {1, ... ,l}, |K[j]| = d, that has been calculated with the ACPKM-Master algorithm as follows:

IV = 1^{n/2},

K[1] | ... | K[l] = ACPKM-Master(T*, K*, d*l) = CTR-ACPKM-Encrypt (T*, K*, IV, 0^{d*l}).

### 5.2.2. CTR Mode Key Meshing

This section defines a CTR-ACPKM-Master encryption mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The CTR-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic encryption mode CTR (see [MODES]).

The CTR-ACPKM-Master encryption mode can be used with the following 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.

The key material K[j] that is used for one section processing is equal to K^j, |K^j| = k bits.

The CTR-ACPKM-Master mode encryption and decryption procedures are defined as follows:

```
+----------------------------------------------------------------+
|  CTR-ACPKM-Master-Encrypt(N, K*, T*, ICN, P)                    |
|----------------------------------------------------------------|
|  Input:                                                         |
|  - Section size N,                                              |
|  - master key K*,                                               |
|  - change frequency T*,                                         |
|  - initial counter nonce ICN in V_{n-c},                        |
|  - plaintext P = P_1 | ... | P_b, |P| <= 2^{n/2-1}*n*N / k.     |
|  Output:                                                        |
|  - Ciphertext C.                                                |
|----------------------------------------------------------------|
|  1. CTR_1 = ICN | 0^c                                           |
|  2. For j = 2, 3, ... , b do                                    |
|         CTR_{j} = Inc_c(CTR_{j-1})                              |
|  3. l = ceil(b*n / N)                                           |
|  4. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                 |
|  5. For j = 1, 2, ... , b do                                    |
|         i = ceil(j*n / N),                                      |
|         G_j = E_{K^i}(CTR_j)                                    |
|  6. C = P (xor) MSB_{|P|}(G_1 | ... |G_b)                       |
|  7. Return C                                                    |
|----------------------------------------------------------------+

+----------------------------------------------------------------+
|  CTR-ACPKM-Master-Decrypt(N, K*, T*, ICN, C)                    |
|----------------------------------------------------------------|
|  Input:                                                         |
|  - Section size N,                                              |
|  - master key K*,                                               |
|  - change frequency T*,                                         |
|  - initial counter nonce ICN in V_{n-c},                        |
|  - ciphertext C = C_1 | ... | C_b, |C| <= 2^{n/2-1}*n*N / k.    |
|  Output:                                                        |
|  - Plaintext P.                                                 |
|----------------------------------------------------------------|
|  1. Return CTR-ACPKM-Master-Encrypt(N, K*, T*, ICN, C)          |
+----------------------------------------------------------------+
```

The initial counter nonce ICN value for each message that is encrypted under the given key must be chosen in a unique manner. The counter (CTR_{i+1}) value does not change during key transformation.

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

### 5.2.3. GCM Mode Key Meshing

This section defines a GCM-ACPKM-Master encryption mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The GCM-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic encryption mode GCM (see [GCM]).

The GCM-ACPKM-Master encryption mode can be used with the following parameters:

- n in {128, 256};
- 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;
- authentication tag length t.

The key material K[j] that is used for one section processing is equal to K^j, |K^j| = k bits, that is calculated as follows:

K^1 | ... | K^j | ... | K^l = ACPKM-Master(T*, K*, k*l).

The GCM-ACPKM-Master mode encryption and decryption procedures are defined as follows:

```
+-------------------------------------------------------------------+
|  GHASH(X, H)                                                      |
|-------------------------------------------------------------------|
|  Input:                                                           |
|  - Bit string X = X_1 | ... | X_m, X_i in V_n for i in {1, ... , m}|
|  Output:                                                          |
|  - Block GHASH(X, H) 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                                                    |
+-------------------------------------------------------------------+


+-------------------------------------------------------------------+
|  GCTR(N, K*, T*, ICB, X)                                          |
|-------------------------------------------------------------------|
|  Input:                                                           |
|  - Section size N,                                                |
|  - master key K*,                                                 |
|  - change frequency T*,                                           |
|  - initial counter block ICB,                                     |
|  - X = X_1 | ... | X_b, X_i in V_n for i = 1, ... , b-1 and       |
|               X_b 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. GCTR_1 = ICB                                                  |
|  3. For i = 2, ... , b do                                         |
|         GCTR_i = Inc_c(GCTR_{i-1})                                |
|  4. l = ceil(b*n / N)                                             |
```

```
|  5. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                    |
|  6. For j = 1, ... , b do                                          |
|         i = ceil(j*n / N),                                         |
|         G_j = E_{K^i}(GCTR_j)                                      |
|  7. Y = X (xor) MSB_{|X|}(G_1 | ... | G_b)                         |
|  8. Return Y                                                       |
+--------------------------------------------------------------------+


+--------------------------------------------------------------------+
|  GCM-ACPKM-Master-Encrypt(N, K*, T*, IV, P, A)                     |
|--------------------------------------------------------------------|
|  Input:                                                            |
|  - Section size N,                                                 |
|  - master key K*,                                                  |
|  - change frequency T*,                                            |
|  - initial counter nonce ICN in V_{n-c},                           |
|  - plaintext P, |P| <= n*(2^{c-1} - 2).                            |
|  - additional authenticated data A.                                |
|  Output:                                                           |
|  - Ciphertext C,                                                   |
|  - authentication tag T.                                           |
|--------------------------------------------------------------------|
|  1. K^1 = ACPKM-Master(T*, K*, k)                                  |
|  2. H = E_{K^1}(0^n)                                               |
|  3. If c = 32, then ICB_0 = ICN | 0^31 | 1                         |
|     if c!= 32, then s = n*ceil(|ICN|/n) - |ICN|,                   |
|                ICB_0 = GHASH(ICN | 0^{s+n-64} | Vec_64(|ICN|), H)  |
|  4. C = GCTR(N, K*, T*, Inc_32(J_0), P)                            |
|  5. u = n*ceil(|C| / n) - |C|                                      |
|     v = n*ceil(|A| / n) - |A|                                      |
|  6. S = GHASH(A | 0^v | C | 0^u | 0^{n-128} | Vec_64(|A|) |        |
|               | Vec_64(|C|), H)                                     |
|  7. T = MSB_t(E_{K^1}(J_0) (xor) S)                                |
|  8. Return C | T                                                   |
+--------------------------------------------------------------------+


+--------------------------------------------------------------------+
|  GCM-ACPKM-Master-Decrypt(N, K*, T*, IV, A, C, T)                  |
|--------------------------------------------------------------------|
|  Input:                                                            |
|  - Section size N,                                                 |
|  - master key K*,                                                  |
|  - change frequency T*,                                            |
|  - initial counter nonce ICN in V_{n-c},                           |
|  - additional authenticated data A.                                |
|  - ciphertext C, |C| <= n*(2^{c-1} - 2),                           |
|  - authentication tag T,                                           |
|  Output:                                                           |
|  - Plaintext P or FAIL.                                            |
|--------------------------------------------------------------------|
|  1. K^1 = ACPKM-Master(T*, K*, k)                                  |
|  2. H = E_{K^1}(0^n)                                               |
|  3. If c = 32, then ICB_0 = ICN | 0^31 | 1                         |
```

```
|     if c!= 32, then s = n*ceil(|ICN| / n) - |ICN|,                  |
|                ICB_0 = GHASH(ICN | 0^{s+n-64} | Vec_64(|ICN|), H)  |
|  4. P = GCTR(N, K*, T*, Inc_32(J_0), C)                             |
|  5. u = n*ceil(|C| / n) - |C|                                       |
|     v = n*ceil(|A| / n) - |A|                                       |
|  6. S = GHASH(A | 0^v | C | 0^u | 0^{n-128} |  Vec_64(|A|) |       |
|              | Vec_64(|C|), H)                                      |
|  7. T' = MSB_t(E_{K^1}(ICB_0) (xor) S)                              |
|  8. IF T = T' then return P; else return FAIL.                      |
+---------------------------------------------------------------------+
```

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 defined by the polynomial f as follows (by analogy with [GCM]):

n = 128:
    f = a^128 + a^7 + a^2 + a^1 + 1.

n = 256:
    f = a^256 + a^10 + a^5 + a^2 + 1.

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

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

### 5.2.4. CBC Mode Key Meshing

This section defines a CBC-ACPKM-Master encryption mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The CBC-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic encryption mode CBC (see [MODES]).

The CBC-ACPKM-Master encryption mode can be used with the following parameters:

- 64 <= n <= 512;
- 128 <= k <= 512.

In the specification of the CBC-ACPKM-Master mode the plaintext and ciphertext must be a sequence of one or more complete data blocks. If the data string to be encrypted does not initially satisfy this property, then it MUST be padded to form complete data blocks. The padding methods are outside the scope of this document. An example of a padding method can be found in Appendix A of [MODES].

The key material K[j] that is used for one section processing is equal to K^j, |K^j| = k bits.

We will denote by D_{K} the decryption function which is a permutation inverse to the E_{K}.

The CBC-ACPKM-Master mode encryption and decryption procedures are defined as follows:

```
+-----------------------------------------------------------------+
|  CBC-ACPKM-Master-Encrypt(N, K*, T*, IV, P)                      |
|-----------------------------------------------------------------|
|  Input:                                                          |
|  - Section size N,                                               |
|  - master key K*,                                                |
|  - change frequency T*,                                          |
```

```
|  - initialization vector IV in V_n,                              |
|  - plaintext P = P_1 | ... | P_b, |P| <= 2^{n/2-1}*n*N / k,     |
|                  |P_b| = n.                                    |
|  Output:                                                      |
|  - Ciphertext C.                                              |
|---------------------------------------------------------------|
|  1. l = ceil(b*n/N)                                           |
|  2. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                 |
|  3. C_0 = IV                                                  |
|  4. For j = 1, 2, ... , b do                                  |
|         i = ceil(j*n / N),                                    |
|         C_j = E_{K^i}(P_j (xor) C_{j-1})                      |
|  5. Return C = C_1 | ... | C_b                                 |
|---------------------------------------------------------------+

+---------------------------------------------------------------+
|  CBC-ACPKM-Master-Decrypt(N, K*, T*, IV, C)                    |
|---------------------------------------------------------------|
|  Input:                                                       |
|  - Section size N,                                            |
|  - master key K*,                                             |
|  - change frequency T*,                                       |
|  - initialization vector IV in V_n,                           |
|  - ciphertext C = C_1 | ... | C_b, |C| <= 2^{n/2-1}*n*N/k,      |
|                  |C_b| = n.                                    |
|  Output:                                                      |
|  - Plaintext P.                                               |
|---------------------------------------------------------------|
|  1. l = ceil(b*n / N)                                         |
|  2. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                 |
|  3. C_0 = IV                                                  |
|  4. For j = 1, 2, ... , b do                                  |
|         i = ceil(j*n/N)                                       |
|         P_j = D_{K^i}(C_j) (xor) C_{j-1}                      |
|  5. Return P = P_1 | ... | P_b                                 |
+---------------------------------------------------------------+
```

The initialization vector IV for each message that is encrypted under the given key need not to be secret, but must be unpredictable.

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

### 5.2.5. CFB Mode Key Meshing

This section defines a CFB-ACPKM-Master encryption mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The CFB-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic encryption mode CFB (see [MODES]).

The CFB-ACPKM-Master encryption mode can be used with the following parameters:

- 64 <= n <= 512;
- 128 <= k <= 512.

The key material K[j] that is used for one section processing is equal to K^j, |K^j| = k bits.

The CFB-ACPKM-Master mode encryption and decryption procedures are defined as follows:

```
+----------------------------------------------------------------+
|  CFB-ACPKM-Master-Encrypt(N, K*, T*, IV, P)                    |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - master key K*,                                              |
|  - change frequency T*,                                        |
|  - initialization vector IV in V_n,                            |
|  - plaintext P = P_1 | ... | P_b, |P| <=  2^{n/2-1}*n*N / k.   |
|  Output:                                                       |
|  - Ciphertext C.                                               |
|----------------------------------------------------------------|
|  1. l = ceil(b*n / N)                                          |
|  2. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                |
|  3. C_0 = IV                                                   |
|  4. For j = 1, 2, ... , b do                                   |
|         i = ceil(j*n / N)                                      |
|         C_j = E_{K^i}(C_{j-1}) (xor) P_j                       |
|  5. Return C = C_1 | ... | C_b.                                |
|----------------------------------------------------------------+

+----------------------------------------------------------------+
|  CFB-ACPKM-Master-Decrypt(N, K*, T*, IV, C#)                   |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - master key K*,                                              |
|  - change frequency T*,                                        |
|  - initialization vector IV in V_n,                            |
|  - ciphertext C = C_1 | ... | C_b, |C| <=  2^{n/2-1}*n*N / k.  |
|  Output:                                                       |
|  - Plaintext P.                                                |
|----------------------------------------------------------------|
|  1. l = ceil(b*n / N)                                          |
|  2. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                |
|  3. C_0 = IV                                                   |
|  4. For j = 1, 2, ... , b do                                   |
|         i = ceil(j*n / N),                                     |
|         P_j = E_{K^i}(C_{j-1}) (xor) C_j                       |
|  5. Return P = P_1 | ... | P_b                                 |
+----------------------------------------------------------------+
```

The initialization vector IV for each message that is encrypted under the given key need not to be secret, but must be unpredictable.

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

### 5.2.6. OFB Mode Key Meshing

This section defines an OFB-ACPKM-Master encryption mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The OFB-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic encryption mode OFB (see [MODES]).

The OFB-ACPKM-Master encryption mode can be used with the following parameters:

- 64 <= n <= 512;
- 128 <= k <= 512.

The key material K[j] used for one section processing is equal to K^j, |K^j| = k bits.

The OFB-ACPKM-Master mode encryption and decryption procedures are defined as follows:

```
+----------------------------------------------------------------+
|  OFB-ACPKM-Master-Encrypt(N, K*, T*, IV, P)                    |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - master key K*,                                              |
|  - change frequency T*,                                        |
|  - initialization vector IV in V_n,                            |
|  - plaintext P = P_1 | ... | P_b, |P| <= 2^{n/2-1}*n*N / k.    |
|  Output:                                                       |
|  - Ciphertext C.                                               |
|----------------------------------------------------------------|
|  1. l = ceil(b*n / N)                                          |
|  2. K^1 | ... | K^l = ACPKM-Master(T*, K*, k*l)                |
|  3. G_0 = IV                                                   |
|  4. For j = 1, 2, ... , b do                                   |
|         i = ceil(j*n / N),                                     |
|         G_j = E_{K_i}(G_{j-1})                                 |
|  5. Return C = P (xor) MSB_{|P|}(G_1 | ... | G_b)              |
|----------------------------------------------------------------+

+----------------------------------------------------------------+
|  OFB-ACPKM-Master-Decrypt(N, K*, T*, IV, C)                    |
|----------------------------------------------------------------|
|  Input:                                                        |
|  - Section size N,                                             |
|  - master key K*,                                              |
|  - change frequency T*,                                        |
|  - initialization vector IV in V_n,                            |
|  - ciphertext C = C_1 | ... | C_b, |C| <= 2^{n/2-1}*n*N / k.   |
|  Output:                                                       |
|  - Plaintext P.                                                |
|----------------------------------------------------------------|
|  1. Return OFB-ACPKM-Master-Encrypt(N, K*, T*, IV, C)          |
+----------------------------------------------------------------+
```

The initialization vector IV for each message that is encrypted under the given key need not be unpredictable, but it must be a nonce that is unique to each execution of the encryption operation.

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

### 5.2.7. OMAC Mode Key Meshing

This section defines an OMAC-ACPKM-Master message authentication code calculation mode that uses internal ACPKM-Master re-keying mechanism for the periodical key transformation.

The OMAC-ACPKM-Master encryption mode can be considered as the extended by the ACPKM-Master re-keying mechanism basic message authentication code calculation mode OMAC (see [RFC4493]).

The OMAC-ACPKM-Master message authentication code calculation mode can be used with the following parameters:

- n in {64, 128, 256};
- 128 <= k <= 512.

The key material K[j] that is used for one section processing is equal to K^j | K^j_1, where |K^j| = k and |K^j_1| = n.

The following is a specification of the subkey generation process of OMAC:

```
+-------------------------------------------------------------------+
| Generate_Subkey(K, r)                                             |
|-------------------------------------------------------------------|
| Input:                                                            |
|  - Key K,                                                         |
|  Output:                                                          |
|  - Key [K].                                                       |
|-------------------------------------------------------------------|
|   1. If r = n then return K                                       |
|   2. If r < n then                                                |
|          if MSB_1(K1) = 0                                         |
|              return K1 << 1                                       |
|          else                                                     |
|              return (K1 << 1) (xor) R_n                           |
|                                                                   |
+-------------------------------------------------------------------+
```

Where R_n takes the following values:

- n = 64: R_{64} = 0^{59} | 11011;
- n = 128: R_{128} = 0^{120} | 10000111;
- n = 256: R_{256} = 0^{145} | 10000100101.

The OMAC-ACPKM-Master message authentication code calculation mode is defined as follows:

```
+-------------------------------------------------------------------+
| OMAC-ACPKM-Master(K*, N, T*, M)                                   |
|-------------------------------------------------------------------|
| Input:                                                            |
|  - Section size N,                                                |
|  - master key K*,                                                 |
|  - key frequency T*,                                              |
```

```
|  - plaintext M = M_1 | ... | M_b, |M| <= 2^{n/2}*n^2*N / (k + n).   |
|  Output:                                                           |
|  - message authentication code T.                                  |
|-------------------------------------------------------------------|
|  1. C_0 = 0^n                                                      |
|  2. l = ceil(b*n / N)                                              |
|  3. K^1 | K^1_1 | ... | K^l | K^l_1 = ACPKM-Master(T*, K*, (k+n)*l) |
|  4. For j = 1, 2, ... , b-1 do                                     |
|         i = ceil(j*n / N),                                          |
|         C_j = E_{K^i}(M_j (xor) C_{j-1})                            |
|  5. [K] = Generate_Subkey(K^l_1, |M_b|)                            |
|  6. If |M_b| = n then M*_b = M_b                                   |
|                 else M*_b = M_b | 1 | 0^{n - 1 -|M_b|}             |
|  7. T = E_{K^l}(M*_b (xor) C_{b-1} (xor) [K])                      |
|  8. Return T                                                       |
+-------------------------------------------------------------------+
```

The message size m MUST NOT exceed 2^{n/2}*n^2*N / (k + n) bits.

## 6. Joint Usage of External and Internal Re-keying

Any mechanism described in Section 4 can be used with any mechanism described in Section 5.

## 7. Security Considerations

## 7.1. Principles of Choice of Constructions and Security Parameters

External re-keying mechanism is RECOMMENDED to be used in protocols that process pretty small messages (e.g. TLS).

Internal re-keying mechanism is RECOMMENDED to be used in protocols that can process large messages (e.g. IPSec).

For the protocols that process messages of different lengths it is RECOMMENDED to use joint methods described in Section 6.

## 7.2. Requirements For Base Primitives

Re-keying should be used to increase "a priori" security properties of ciphers in hoslile environments (e.g. with side-channel adversaries). If some non-negligible attacks are known for a cipher, it MUST NOT be used. So re-keying can not be used as a patch for vulnerable ciphers. Base cipher properties must be well analyzed, because security of re-keying mechanisms is based on security of a block cipher as a pseudorandom function.

## 8. References

## 8.1. Normative References


**[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.

**[GOST3411-2012]** Federal Agency on Technical Regulating and Metrology (In Russian), "Information technology. Cryptographic Data Security. Hashing function", GOST R 34.11-2012, 2012.

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

**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

**[RFC4493]** Song, JH., Poovendran, R., Lee, J. and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006.

**[RFC5869]** Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010.

**[SHA-512]** National Institute of Standards and Technology., "Secure Hash Standard", FIPS 180-2, August, with Change Notice 1 dated February 2004 2002.

**[TLSDraft]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", 2017.


## 8.2. Informative References


**[AbBell]** Michel Abdalla and Mihir Bellare, "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques", ASIACRYPT2000, LNCS 1976, pp. 546–559, 2000.

**[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.

**[RFC6986]** Dolmatov, V. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", RFC 6986, DOI 10.17487/RFC6986, August 2013.


## Appendix A. Test examples

```
CTR-ACPKM mode with AES-256
*********
c = 64
k = 256
N = 256
n = 128

W_0:
F3 74 E9 23 FE AA D6 DD 98 B4 B6 3D 57 8B 35 AC

W_1:
A9 0F D7 31 E4 1D 64 5E C0 8C 87 87 28 CC 76 90

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

Plain text P:
11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88
00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A
11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00
22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11
```

33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22
44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22 33
55 66 77 88 99 AA BB CC EE FF 0A 00 11 22 33 44

ICN:
12 34 56 78 90 AB CE F0

ACPKM's iteration 1

Process block 1

Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 00

Output block (ctr)
FD 7E F8 9A D9 7E A4 B8 8D B8 B5 1C 1C 9D 6D D0

Plain text
11 22 33 44 55 66 77 00 FF EE DD CC BB AA 99 88

Cipher text
EC 5C CB DE 8C 18 D3 B8 72 56 68 D0 A7 37 F4 58

Process block 2

Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 01

Output block (ctr)
19 98 C5 71 76 37 FB 17 11 E4 48 F0 0C 0D 60 B2

Plain text
00 11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A

Cipher text
19 89 E7 42 32 62 9D 60 99 7D E2 4B C0 E3 9F B8

Updated key
C6 C1 AF 82 3F 52 22 F8 97 CF F1 94 5D F7 21 9E
21 6F 29 0C EF C4 C7 E6 DC C8 B7 DD 83 E0 AE 60

ACPKM's iteration 2

Process block 3
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 02

Output block (ctr)
92 B4 85 B5 B7 AD 3C 19 7E 53 92 32 13 9C 8E 7A

Plain text
11 22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00

Cipher text
83 96 B6 F1 E2 CB 4B 91 E7 F9 29 FE FD 63 84 7A

Process block 4
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 03

Output block (ctr)
59 3A AA 96 7C E3 58 FB 1B 7E 41 A1 77 34 B1 4A

Plain text
22 33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11

Cipher text
7B 09 EE C3 1A 94 D0 62 B1 C5 8D 4F 88 3E B1 5B

Updated key
65 3E FA 18 0B 0E 68 01 6F 56 54 A5 F3 EE BC D5
04 F1 1F E3 F1 7A 92 07 57 A8 82 BE A5 9E CA 16

ACPKM's iteration 3
Process block 5
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 04

Output block (ctr)
CE E5 51 54 12 2F 3F E7 8D 8E 86 21 C5 E5 47 12

Plain text
33 44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22

Cipher text
FD A1 04 32 65 A7 A6 4D 36 42 68 DE CF E5 56 30

Process block 6
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 05

Output block (ctr)
DE D6 8F 03 FA C5 C5 B6 16 11 A3 78 2C 0D C1 EB

Plain text
44 55 66 77 88 99 AA BB CC EE FF 0A 00 11 22 33

Cipher text
9A 83 E9 74 72 5C 6F 0D DA FF 5C 72 2C 1C E3 D8

Updated key
C0 D5 50 26 4F DA CE 59 EF 80 9A 50 24 72 06 7D
29 83 74 25 78 C9 60 4F E3 B8 88 4F F8 F5 E2 BD

ACPKM's iteration 4
Process block 7

```
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 06

Output block (ctr)
D9 23 A6 CD 8A 00 A1 55 90 09 EC 87 40 B9 D6 AB

Plain text
55 66 77 88 99 AA BB CC EE FF 0A 00 11 22 33 44

Cipher text
8C 45 D1 45 13 AA 1A 99 7E F6 E6 87 51 9B E5 EF

Updated key
6A A0 92 07 73 31 63 50 46 FA 48 1C 9C 98 7B 6B
FC 99 48 DC BC AE AB C2 6D 46 E9 DD 43 F6 CA 56

Encrypted src
EC 5C CB DE 8C 18 D3 B8 72 56 68 D0 A7 37 F4 58
19 89 E7 42 32 62 9D 60 99 7D E2 4B C0 E3 9F B8
83 96 B6 F1 E2 CB 4B 91 E7 F9 29 FE FD 63 84 7A
7B 09 EE C3 1A 94 D0 62 B1 C5 8D 4F 88 3E B1 5B
FD A1 04 32 65 A7 A6 4D 36 42 68 DE CF E5 56 30
9A 83 E9 74 72 5C 6F 0D DA FF 5C 72 2C 1C E3 D8
8C 45 D1 45 13 AA 1A 99 7E F6 E6 87 51 9B E5 EF
```

## Authors' Addresses

**Stanislav Smyshlyaev** (editor)
CryptoPro
18, Suschevsky val
Moscow, 127018
Russian Federation
Phone: +7 (495) 995-48-20
EMail: svs@cryptopro.ru

**Russ Housley**
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
EMail: housley@vigilsec.com

**Mihir Bellare**
University of California, San Diego
9500 Gilman Drive
La Jolla, California 92093-0404
USA
Phone: (858) 534-4544
EMail: mihir@eng.ucsd.edu

**Evgeny Alekseev**

CryptoPro
18, Suschevsky val
Moscow, 127018
Russian Federation
Phone: +7 (495) 995-48-20
EMail: alekseev@cryptopro.ru

**Ekaterina Smyshlyaeva**
CryptoPro
18, Suschevsky val
Moscow, 127018
Russian Federation
Phone: +7 (495) 995-48-20
EMail: ess@cryptopro.ru

**Daniel Fox Franke**
Akamai Technologies
150 Broadway
Cambridge, MA 02142-1413
USA
EMail: dfoxfranke@gmail.com

**Lilia Ahmetzyanova**
CryptoPro
18, Suschevsky val
Moscow, 127018
Russian Federation
Phone: +7 (495) 995-48-20
EMail: lah@cryptopro.ru

**Ruth Ng**
University of California, San Diego
9500 Gilman Drive
La Jolla, California 92093-0404
USA
EMail: ring@eng.ucsd.edu