

CFRG  
Internet-Draft  
Intended status: Informational  
Expires: September 8, 2017

S. Smyshlyaev, Ed.  
CryptoPro  
R. Housley  
Vigil Security, LLC  
M. Bellare  
University of California, San Diego  
E. Alekseev  
E. Smyshlyaeva  
CryptoPro  
March 7, 2017

**Re-keying Mechanisms for Symmetric Keys**  
**draft-irtf-cfrg-re-keying-01**

**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**

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 <http://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 September 8, 2017.

**Copyright Notice**

Copyright (c) 2017 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 (<http://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 the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Conventions Used in This Document . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Basic Terms and Definitions . . . . .	<a href="#">3</a>
<a href="#">4.</a>	External Re-keying Mechanisms . . . . .	<a href="#">5</a>
<a href="#">4.1.</a>	Parallel Constructions . . . . .	<a href="#">5</a>
4.1.1.	Parallel Construction Based on a KDF on a Block Cipher . . . . .	<a href="#">6</a>
<a href="#">4.1.2.</a>	Parallel Construction Based on HKDF . . . . .	<a href="#">6</a>
<a href="#">4.2.</a>	Serial Constructions . . . . .	<a href="#">7</a>
4.2.1.	Serial Construction Based on a KDF on a Block Cipher	<a href="#">7</a>
<a href="#">4.2.2.</a>	Serial Construction Based on HKDF . . . . .	<a href="#">8</a>
<a href="#">5.</a>	Internal Re-keying Mechanisms . . . . .	<a href="#">8</a>
<a href="#">5.1.</a>	Constructions that Do Not Require Master Key . . . . .	<a href="#">8</a>
<a href="#">5.1.1.</a>	ACPKM Re-keying Mechanisms . . . . .	<a href="#">8</a>
<a href="#">5.1.2.</a>	CTR-ACPKM Encryption Mode . . . . .	<a href="#">10</a>
<a href="#">5.1.3.</a>	GCM-ACPKM Encryption Mode . . . . .	<a href="#">12</a>
<a href="#">5.2.</a>	Constructions that Require Master Key . . . . .	<a href="#">14</a>
<a href="#">5.2.1.</a>	ACPKM-Master Key Generation from the Master Key . . .	<a href="#">15</a>
<a href="#">5.2.2.</a>	CTR Mode Key Meshing . . . . .	<a href="#">16</a>
<a href="#">5.2.3.</a>	GCM Mode Key Meshing . . . . .	<a href="#">19</a>
<a href="#">5.2.4.</a>	CBC Mode Key Meshing . . . . .	<a href="#">22</a>
<a href="#">5.2.5.</a>	CFB Mode Key Meshing . . . . .	<a href="#">24</a>
<a href="#">5.2.6.</a>	OFB Mode Key Meshing . . . . .	<a href="#">26</a>
<a href="#">5.2.7.</a>	OMAC Mode Key Meshing . . . . .	<a href="#">27</a>
<a href="#">6.</a>	Joint Usage of External and Internal Re-keying . . . . .	<a href="#">29</a>
<a href="#">7.</a>	Scope of Usage of Rekeying-Based Schemas . . . . .	<a href="#">29</a>
<a href="#">7.1.</a>	Key Transformation Rules . . . . .	<a href="#">29</a>
7.2.	Principles of Choice of Constructions and Security Parameters . . . . .	<a href="#">30</a>
<a href="#">8.</a>	Security Considerations . . . . .	<a href="#">32</a>
<a href="#">9.</a>	References . . . . .	<a href="#">32</a>
<a href="#">9.1.</a>	Normative References . . . . .	<a href="#">33</a>
<a href="#">9.2.</a>	Informative References . . . . .	<a href="#">33</a>
<a href="#">Appendix A.</a>	Test examples . . . . .	<a href="#">34</a>
<a href="#">Appendix B.</a>	Contributors . . . . .	<a href="#">37</a>
Authors' Addresses	. . . . .	<a href="#">38</a>



## **1. Introduction**

Common cryptographic 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 the threshold, this key must be replaced.

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  $2^n$ ;

$\text{Int}_s: V_s \rightarrow Z_{\{2^s\}}$  the transformation that maps a string  $a = (a_s, \dots, a_1)$ ,  $a$  in  $V_s$ , into the integer  $\text{Int}_s(a) = 2^s a_s + \dots + 2 a_2 + a_1$ ;

$\text{Vec}_s: Z_{\{2^s\}} \rightarrow V_s$  the transformation inverse to the mapping  $\text{Int}_s$ ;

$\text{MSB}_i: V_s \rightarrow V_i$  the transformation that maps the string  $a = (a_s, \dots, a_1)$  in  $V_s$ , into the string  $\text{MSB}_i(a) = (a_s, \dots, a_{\{s-i+1\}})$  in  $V_i$ ;

$\text{LSB}_i: V_s \rightarrow V_i$  the transformation that maps the string  $a = (a_s, \dots, a_1)$  in  $V_s$ , into the string  $\text{LSB}_i(a) = (a_i, \dots, a_1)$  in  $V_i$ ;

$\text{Inc}_c: V_s \rightarrow V_s$  the transformation that maps the string  $a = (a_s, \dots, a_1)$  in  $V_s$ , into the string  $\text{Inc}_c(a) = \text{MSB}_{\{|a|-c\}}(a) \mid \text{Vec}_c(\text{Int}_c(\text{LSB}_c(a)) + 1(\bmod 2^c))$  in  $V_s$ ;

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

$E_{\{K\}}: V_n \rightarrow V_n$  the block cipher permutation under the key  $K$  in  $V_k$ ;

$\text{ceil}(x)$  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 ( $b = \text{ceil}(m/n)$ );

$N$  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);

$\text{phi}_i: V_s \rightarrow V_s$  the transformation that maps a string  $a = (a_s, \dots, a_1)$  into the string  $\text{phi}_i(a) = a' = (a'_s, \dots,$



$a'_1$ ),  $1 \leq i \leq s$ , such that  $a'_i = 1$  and  $a'_j = a_j$  for all  $j$  in  $\{1, \dots, s\} \setminus \{i\}$ .

A plaintext message  $P$  and a ciphertext  $C$  are divided into  $b = \lceil m/n \rceil$  segments denoted as  $P = P_1 \mid P_2 \mid \dots \mid P_b$  and  $C = C_1 \mid C_2 \mid \dots \mid C_b$ , where  $P_i$  and  $C_i$  are in  $V_n$ , for  $i = 1, 2, \dots, b-1$ , and  $P_b, C_b$  are in  $V_r$ , where  $r \leq n$  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:





Maximum message size =  $m_{\max}$ .

---

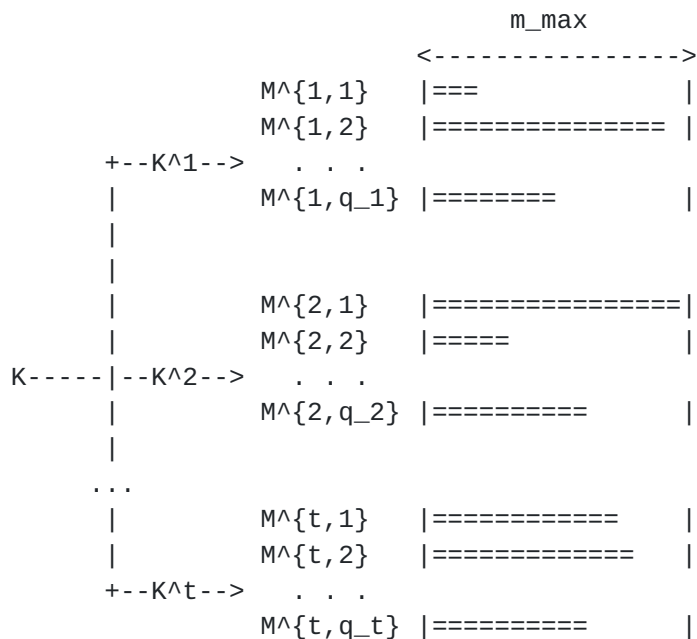


Figure 1: External parallel re-keying mechanisms

The key  $K^i$ ,  $i = 1, \dots, t-1$ , is updated after processing a certain amount of data (see [Section 7.1](#)).

#### 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:

$$K^1 \mid K^2 \mid \dots \mid K^t = \text{ExtParallelC}(K, t \cdot k) = \text{MSB}_{\{t \cdot k\}}(E_{\{K\}}(0) \mid E_{\{K\}}(1) \mid \dots \mid E_{\{K\}}(J-1)),$$

where  $J = \text{ceil}(k/n)$ .

#### 4.1.2. Parallel Construction Based on HKDF

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

$$K^1 \mid K^2 \mid \dots \mid K^t = \text{ExtParallelH}(K, t \cdot k) = \text{HKDF-Expand}(K, \text{label}, t \cdot k),$$



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:

Maximum message size =  $m_{\max}$ .

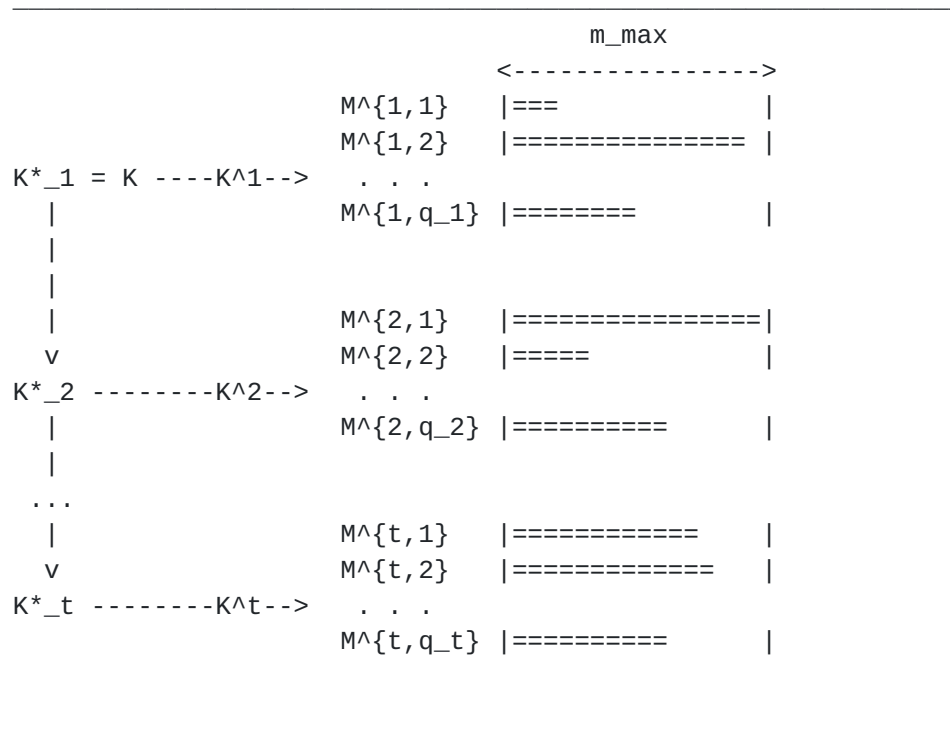


Figure 2: External serial re-keying mechanisms

The key  $K^i$ ,  $i = 1, \dots, t-1$ , is updated after processing a certain amount of data (see [Section 7.1](#)).

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

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

$$K^i = \text{ExtSerialC}(K, i) = \text{MSB}_k(E_{\{K^*_i\}}(0) \parallel E_{\{K^*_i\}}(1) \parallel \dots \parallel E_{\{K^*_i\}}(J-1)),$$

where  $J = \text{ceil}(k/n)$ ,  $i = 1, \dots, t$ ,  $K^*_i$  is calculated as follows:



$$K^*_1 = K,$$

$$K^*_{\{j+1\}} = \text{MSB}_k(E_{\{K^*_j\}}(J) \mid E_{\{K^*_j\}}(J+1) \mid \dots \mid E_{\{K^*_j\}}(2J-1)),$$

where  $j = 1, \dots, t-1$ .

#### **4.2.2. Serial Construction Based on HKDF**

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

$$K^i = \text{ExtSerialH}(K, i) = \text{HKDF-Expand}(K^*_i, \text{label1}, k),$$

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

$$K^*_1 = K,$$

$$K^*_{\{j+1\}} = \text{HKDF-Expand}(K^*_j, \text{label2}, k), \text{ where } j = 1, \dots, t-1,$$

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, 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**

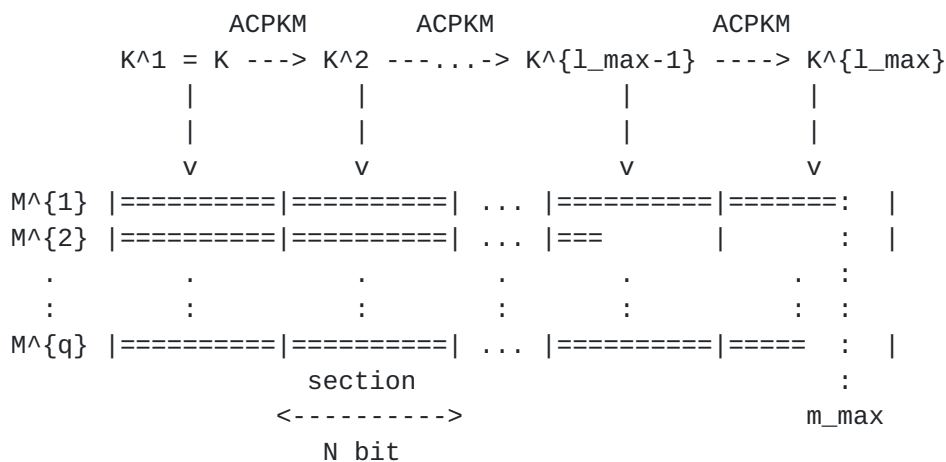
This section defines periodical key transformation with no master key which is called ACPKM re-keying mechanism. This mechanism can be applied to one of the basic encryption modes (CTR and GCM block cipher modes) for getting an extension of this encryption mode that uses periodical key transformation 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 8](#)). 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:

Section size = const =  $N$ ,  
maximum message size =  $m_{\max}$ .



$l_{\max} = \text{ceil}(m_{\max}/N)$ .

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  $l = \text{ceil}(m/N)$  sections (denoted as  $M = M_1 | M_2 | \dots | M_l$ , where  $M_i$  is in  $V_N$  for  $i = 1, 2, \dots, l-1$  and  $M_l$  is in  $V_r$ ,  $r \leq N$ ). The first section of each message is processed with the initial key  $K^1 = K$ . To process the  $(i+1)$ -th section of each message the  $K^{i+1}$  key value is calculated using ACPKM transformation as follows:

$$K^{i+1} = \text{ACPKM}(K^i) = \text{MSB}_k(E_{\{K^i\}}(W_1) | \dots | E_{\{K^i\}}(W_J)),$$

where  $J = \text{ceil}(k/n)$ ,  $W_t = \text{phi}_c(D_t)$  for any  $t$  in  $\{1, \dots, J\}$  and  $D_1, D_2, \dots, D_J$  are in  $V_n$  and are calculated as follows:





$$D\_1 \mid D\_2 \mid \dots \mid D\_J = \text{MSB}_{\{J \cdot n\}}(D),$$

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  $\text{phi}_c(D\_1), \dots, \text{phi}_c(D\_J)$  are pairwise different for any allowed  $n, k, c$  values.

**N o t e :** The constant  $D$  is such that  $D = \text{sha512}(\text{streebog512}(0^{1024})) \mid \text{sha512}(\text{streebog512}(1^{1024}))$ , where  $\text{sha512}$  is a hash function with 512-bit output corresponding to the algorithm SHA-512 [[SHA-512](#)],  $\text{streebog512}$  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:

- o  $64 \leq n \leq 512$ ;
- o  $128 \leq k \leq 512$ ;
- o the number of bits  $c$  in a specific part of the block to be incremented is such that  $32 \leq c \leq 3/4 n$ .



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  $P = P_1 \mid \dots \mid P_b$ ,  $|P| < n * 2^{\{c-1\}}$ . |
| Output: |
| - Ciphertext C. |
+-----+
| 1.  $CTR_1 = ICN \mid 0^c$  |
| 2. For  $j = 2, 3, \dots, b$  do |
|      $CTR_{\{j\}} = Inc_c(CTR_{\{j-1\}})$  |
| 3.  $K^1 = K$  |
| 4. For  $i = 2, 3, \dots, \text{ceil}(|P|/N)$  |
|      $K^i = ACPKM(K^{i-1})$  |
| 5. For  $j = 1, 2, \dots, b$  do |
|      $i = \text{ceil}(j*n / N)$ , |
|      $G_j = E_{\{K^i\}}(CTR_j)$  |
| 6.  $C = P \text{ (xor) } MSB_{\{|P|\}}(G_1 \mid \dots \mid G_b)$  |
| 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  $C = C_1 \mid \dots \mid C_b$ ,  $|C| < n * 2^{\{c-1\}}$ . |
| Output: |
| - Plaintext P. |
+-----+
| 1.  $P = CTR\text{-}ACPKM\text{-}Decrypt(N, K, ICN, C)$  |
| 2. 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.

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:

- o  $n$  in  $\{128, 256\}$ ;
- o  $128 \leq k \leq 512$ ;
- o the number of bits  $c$  in a specific part of the block to be incremented is such that  $32 \leq c \leq 3/4 n$ ;
- o authentication tag length  $t$ .

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

```

+-----+
| GHASH(X, H)                                     |
+-----+
| Input:                                           |
| - Bit string  $X = X_1 \mid \dots \mid X_m$ ,  $X_i$  in  $V_n$  for  $i$  in  $1, \dots, m$ . |
| Output:                                          |
| - Block  $\text{GHASH}(X, H)$  in  $V_n$ .                |
+-----+
| 1.  $Y_0 = 0^n$                                    |
| 2. For  $i = 1, \dots, m$  do                       |
|      $Y_i = (Y_{i-1} \text{ (xor) } X_i) * H$           |
| 3. Return  $Y_m$                                    |
+-----+

+-----+
| GCTR(N, K, ICB, X)                             |
+-----+
| Input:                                           |
| - Section size  $N$ ,                               |
| - key  $K$ ,                                         |
| - initial counter block  $\text{ICB}$ ,                 |
| -  $X = X_1 \mid \dots \mid X_b$ ,  $X_i$  in  $V_n$  for  $i = 1, \dots, b-1$  and |
|      $X_b$  in  $V_r$ , where  $r \leq n$ .                |
| 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,  $|C| \leq n \cdot (2^{c-1} - 2)$ ,  $C = C_1 \mid \dots \mid C_b$ ,
| - authentication tag T
| Output:
| - Plaintext P or FAIL.
|-----+
| 1.  $H = E_{\{K\}}(0^n)$ 
| 2. If  $c = 32$ , then  $ICB_0 = ICN \mid 0^{31} \mid 1$ 
|    if  $c \neq 32$ , then  $s = n \cdot \lceil |ICN|/n \rceil - |ICN|$ ,
|         $ICB_0 = \text{GHASH}(ICN \mid 0^{s+n-64} \mid \text{Vec}_{64}(|ICN|), H)$ 
| 3.  $P = \text{GCTR}(N, K, \text{Inc}_{32}(ICB_0), C)$ 
| 4.  $u = n \cdot \lceil |C|/n \rceil - |C|$ 
|     $v = n \cdot \lceil |A|/n \rceil - |A|$ 
| 5.  $S = \text{GHASH}(A \mid 0^v \mid C \mid 0^u \mid 0^{n-128} \mid \text{Vec}_{64}(|A|) \mid$ 
|         $\mid \text{Vec}_{64}(|C|), H)$ 
| 6.  $T' = \text{MSB}_t(E_{\{K\}}(ICB_0) \text{ (xor) } S)$ 
| 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  $n \cdot (2^{c-1} - 2)$  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, 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**

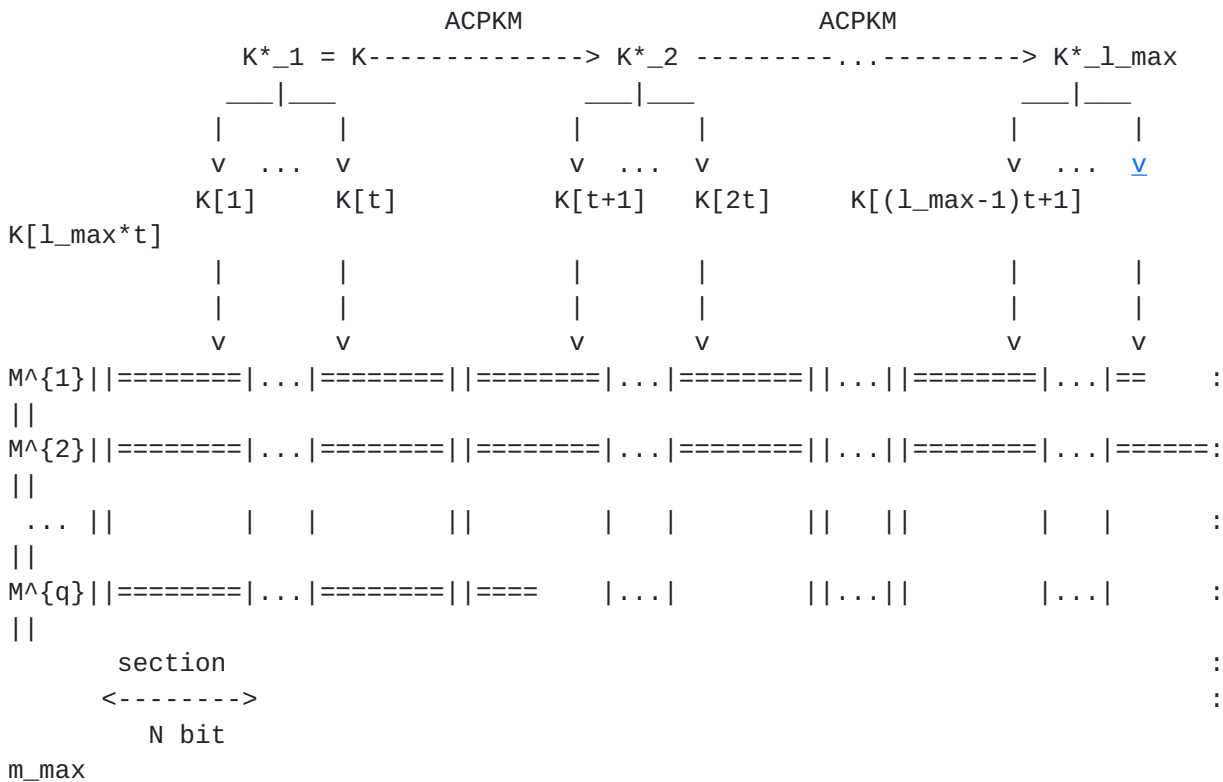
This section defines periodical key transformation with master key  $K$  which is called ACPKM-Master re-keying mechanism. This mechanism can be applied to one of the basic encryption modes (CTR, GCM, CBC, CFB, OFB, OMAC encryption modes) for getting an extension of this encryption mode that uses periodical 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 8](#)). 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:



Change frequency  $T^*$ ,  
section size  $N$ ,  
maximum message size =  $m_{\max}$ .



```

|K[i]| = d,
t = T*/d,
l_max = ceil(m_max/N).

```

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  $l = \text{ceil}(m/N)$  sections (denoted as  $M = M_1 \mid M_2 \mid \dots \mid M_l$ , where  $M_i$  is in  $V_N$  for  $i$  in  $\{1, 2, \dots, l-1\}$  and  $M_l$  is in  $V_r$ ,  $r \leq N$ ). The  $j$ -th section of each message is processed with the key material  $K[j]$ ,  $j$  in  $\{1, \dots, l\}$ ,  $|K[j]| = d$ , that has been calculated with the ACPKM-Master algorithm as follows:

$$K[1] \parallel \dots \parallel K[l] = \text{ACPKM-Master}(T^*, K, d^*l) = \text{CTR-ACPKM-Encrypt}(T^*, K, 1^{n/2}, \theta^{d^*l}).$$

### 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:

- o  $64 \leq n \leq 512$ ;
- o  $128 \leq k \leq 512$ ;
- o the number of bits  $c$  in a specific part of the block to be incremented is such that  $32 \leq c \leq 3/4 n$ .

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^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-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. P = CTR-ACPKM-Master-Encrypt(N, K, T*, ICN, C) |
| 1. 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. 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:

- o  $n$  in  $\{128, 256\}$ ;
- o  $128 \leq k \leq 512$ ;
- o the number of bits  $c$  in a specific part of the block to be incremented is such that  $32 \leq c \leq 3/4 n$ ;
- o 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:

$$K^1 \parallel \dots \parallel K^j \parallel \dots \parallel K^l = \text{ACPKM-Master}(T^*, K, k \cdot l).$$

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

```
+-----+
| GHASH(X, H)                                     |
+-----+
| Input:                                           |
| - Bit string  $X = X_1 \parallel \dots \parallel X_m$ ,  $X_i$  in  $V_n$  for  $i$  in  $\{1, \dots, m\}$  |
| Output:                                         |
| - Block  $\text{GHASH}(X, H)$  in  $V_n$                  |
+-----+
| 1.  $Y_0 = 0^n$                                    |
| 2. For  $i = 1, \dots, m$  do                       |
|      $Y_i = (Y_{i-1} \text{ (xor) } X_i) * H$            |
| 3. Return  $Y_m$                                    |
+-----+

+-----+
| GCTR(N, K,  $T^*$ , ICB, X)                         |
+-----+
```



```

| Input:
| - Section size N,
| - master key K,
| - change frequency T*,
| - initial counter block ICB,
| -  $X = X_1 \mid \dots \mid X_b$ ,  $X_i$  in  $V_n$  for  $i = 1, \dots, b-1$  and
|    $X_b$  in  $V_r$ , where  $r \leq n$ .
| 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  $i = 2, \dots, b$  do
|    $GCTR_i = Inc_c(GCTR_{\{i-1\}})$ 
| 4.  $l = \lceil b \cdot n / N \rceil$ 
| 5.  $K^1 \mid \dots \mid K^l = ACPKM-Master(T^*, K, k^1)$ 
| 6. For  $j = 1, \dots, b$  do
|    $i = \lceil j \cdot n / N \rceil$ ,
|    $G_j = E_{\{K^i\}}(GCTR_j)$ 
| 7.  $Y = X$  (xor)  $MSB_{\{|X|\}}(G_1 \mid \dots \mid G_b)$ 
| 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,  $|P| \leq n \cdot (2^{\{c-1\}} - 2)$ .
| - 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  $ICB_0 = ICN \mid 0^{31} \mid 1$ 
|   if  $c \neq 32$ , then  $s = n \cdot \lceil |ICN| / n \rceil - |ICN|$ ,
|    $ICB_0 = GHASH(ICN \mid 0^{\{s+n-64\}} \mid Vec_{64}(|ICN|), H)$ 
| 4.  $C = GCTR(N, K, T^*, Inc_{32}(J_0), P)$ 
| 5.  $u = n \cdot \lceil |C| / n \rceil - |C|$ 
|    $v = n \cdot \lceil |A| / n \rceil - |A|$ 
| 6.  $S = GHASH(A \mid 0^v \mid C \mid 0^u \mid 0^{\{n-128\}} \mid Vec_{64}(|A|) \mid$ 
|    $Vec_{64}(|C|), H)$ 
| 7.  $T = MSB_t(E_{\{K^1\}}(J_0) \text{ (xor) } S)$ 

```



```

| 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  $(2^{\lceil n/2 \rceil} * n * N / k)$  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:

- o  $64 \leq n \leq 512$ ;
- o  $128 \leq k \leq 512$ .

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*1) |
| 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*1) |
| 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:

- o  $64 \leq n \leq 512$ ;
- o  $128 \leq k \leq 512$ .

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*1) |
| 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*1) |
| 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:

- o  $64 \leq n \leq 512$ ;
- o  $128 \leq k \leq 512$ .

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^1) |
| 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, which is also known as CMAC (see [[RFC4493](#)]).

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

- o  $n$  in  $\{64, 128, 256\}$ ;
- o  $128 \leq k \leq 512$ .

The key material  $K[j]$  that is used for one section processing is equal to  $K^j \parallel K^{j-1}$ , where  $|K^j| = k$  and  $|K^{j-1}| = n$ .

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

```
+-----+
| Generate_Subkey(K1, r)                                     |
|-----+
| Input:                                                     |
| - Key K1,                                                 |
| Output:                                                    |
| - Key SK.                                                 |
|-----+
| 1. If  $r = n$  then return K1                                |
| 2. If  $r < n$  then                                          |
|     if  $MSB_1(K1) = 0$                                      |
|         return  $K1 \ll 1$                                    |
|     else                                                  |
|         return  $(K1 \ll 1) \oplus R_n$                      |
|-----+
```

Where  $R_n$  takes the following values:

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

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. SK = 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^1}(M*_b (xor) C_{b-1} (xor) SK) |
| 8. Return T |
+-----+

```

The message size m MUST NOT exceed  $2^{\{n/2\}} \cdot n^2 \cdot N / (k + n)$  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. Scope of Usage of Rekeying-Based Schemas

### 7.1. Key Transformation Rules

External re-keying mechanisms increase the number of messages that can be processed with one negotiated key.

The key  $K^i$  (see Figure 1 and Figure 2) can be transformed in accordance with one of the following two approaches:

#### o Explicit approach:

$|M^{\{i,1\}}| + \dots + |M^{\{i,q_i\}}| \leq L$ ,  $|M^{\{i,1\}}| + \dots + |M^{\{i,q_i + 1\}}| > L$ ,  $i = 1, \dots, t$ .

This approach allows to use the key  $K^i$  in almost optimal way but it cannot be applied in case when messages may be lost or reordered (e.g. DTLS packets).





- o Implicit approach:

$$q_i = L / m_{\max}, i = 1, \dots, t.$$

The amount of data processed with one key  $K^i$  is calculated under the assumption that every message has the maximum length  $m_{\max}$ . Hence this amount can be considerably less than the key lifetime limitation  $L$ . On the other hand this approach can be applied in case when messages may be lost or reordered (e.g. DTLS packets).

Internal re-keying mechanisms increase the length of messages that can be processed with one negotiated key.

The key  $K$  (see Figure 3 and Figure 4) can be updated in accordance with one of the following two approaches:

- o Explicit approach:

$$|M^{\{1\}}_1| + \dots + |M^{\{q\}}_1| \leq L, |M^{\{1\}}_1| + \dots + |M^{\{q+1\}}_1| > L$$

(where  $M^{\{i\}}_1$  is the first section of message  $M^{\{i\}}$ ,  $i = 1, \dots, q$ ).

This approach allows to use the key  $K^i$  in almost optimal way but it cannot be applied in case messages data may be lost or reordered (e.g. DTLS packets).

- o Implicit approach:

$$q = L / N.$$

The amount of data processed with one key  $K^i$  is calculated under the assumption that the length of every message is equal or more then section size  $N$  and so it can be considerably less than the key lifetime limitation  $L$ . On the other hand this approach can be applied in case when messages may be lost or reordered (e.g. DTLS packets).

## 7.2. 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 records are  $2^{14}$  bytes or less).

Consider an example. Let the message size in some protocol  $P$  be equal to 1 KB ( $m_{\max} = 1$  KB). Suppose a cipher  $E$  is used for encrypting and  $L_1 = 128$  MB is the key lifetime limitation induced by side channels analysis methods. Let the key lifetime limitation  $L_2$  induced by the analysis of encryption mode used in this protocol be equal to 1 TB. The most restrictive resulting key lifetime limitation is equal to 128 MB.

Thus, if external re-keying mechanism is not used, the key  $K$  must be renegotiated after processing  $128 \text{ MB} / 1 \text{ KB} = 131072$  messages.



If an external re-keying mechanism with parameter  $L = 64$  MB (see [Section 7.1](#)) that limits the amount of data processed with one key  $K^i$  is used, the key lifetime limitation  $L_1$  induced by the side channels analysis methods goes off. Thus the resulting key lifetime limitation of the negotiated key  $K$  can be calculated on the basis of the used encryption mode analysis. It is proven that the security of the encryption mode that uses external re-keying leads to an increase when compared to base encryption mode without re-keying (see [\[AbBell\]](#)). Hence the resulting key lifetime limitation in case of using external re-keying is equal to 1 TB.

Thus if an external re-keying mechanism is used, then  $1 \text{ TB} / 1 \text{ KB} = 2^{30}$  messages can be processed before the key  $K$  is renegotiated, which is 8192 times greater than the number of messages that can be processed, when external re-keying mechanism is not used.

An internal re-keying mechanism is recommended to be used in protocols that can process large single messages (e.g. CMS messages).

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

Consider an example. Let the message size in some protocol  $P'$  is large/unlimited. Suppose a cipher  $E$  is used for encrypting and  $L_1 = 128$  MB is the most restrictive key lifetime limitation induced by the side channels analysis methods.

Thus, there is a need to put a limit on maximum message size  $m_{\max}$ . For example, if  $m_{\max} = 32$  MB, it may happen that the renegotiation of key  $K$  would be required after processing only four messages.

If an internal re-keying mechanism with section size  $N = 1$  MB (see Figure 3 and Figure 4) is used, maximum message size limit  $m_{\max}$  can be increased to hundreds of terabytes and  $L / N = 128 \text{ MB} / 1 \text{ MB} = 128$  messages can be processed before the renegotiation of key  $K$  (instead of 4 messages in case when an internal re-keying mechanism is not used).

For the protocols that process messages of different lengths it is recommended to use joint methods (see [Section 6](#)).



## 8. Security Considerations

Re-keying should be used to increase "a priori" security properties of ciphers in hostile 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 cannot 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.

The key lifetime limitation can be subject to the following considerations:

1. Methods of analysis based on the used encryption mode properties.
  - \* These methods do not depend on the used block cipher permutation  $E_{\{K\}}$ .
  - \* For standard encryption modes this restriction has the order  $2^{n/2}$ .
2. Methods based on the side channels analysis.
  - \* These methods do not depend on the used encryption modes.
  - \* These methods are weakly dependent on the used block cipher features (only the way of elementary internal transformation that uses key material matter, in most cases this is (xor)).
  - \* Restrictions resulting from these methods are usually the strongest ones.
3. Methods based on the properties of the used block cipher permutation  $E_{\{K\}}$  (for example, linear or differential cryptanalysis).
  - \* In most cases these methods do not depend on the used encryption modes.
  - \* In case of secure block ciphers restrictions resulting from such methods are roughly the same as the natural limitation  $2^n$ .

## 9. References



### **9.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4493]        Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", [RFC 4493](#), DOI 10.17487/RFC4493, June 2006, <<http://www.rfc-editor.org/info/rfc4493>>.
- [RFC5869]        Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", [RFC 5869](#), DOI 10.17487/RFC5869, May 2010, <<http://www.rfc-editor.org/info/rfc5869>>.
- [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, <<https://tools.ietf.org/html/draft-ietf-tls-tls13-18>>.

### **9.2. 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., Ed. and A. Degtyarev, "GOST R 34.11-2012: Hash Function", [RFC 6986](https://www.rfc-editor.org/info/rfc6986), DOI 10.17487/RFC6986, August 2013, <<http://www.rfc-editor.org/info/rfc6986>>.

## 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 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 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 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 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

## [Appendix B.](#) Contributors

- o Daniel Fox Franke  
Akamai Technologies  
dfoxfranke@gmail.com
- o Lilia Ahmetzyanova  
CryptoPro  
lah@cryptopro.ru
- o Ruth Ng  
University of California, San Diego  
ring@eng.ucsd.edu
- o Shay Gueron  
University of Haifa, Israel





Intel Corporation, Israel Development Center, Israel  
shay.gueron@gmail.com

Authors' Addresses

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

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

Russ Housley  
Vigil Security, LLC  
918 Spring Knoll Drive  
Herndon VA 20170  
USA

Email: [housley@vigilsec.com](mailto: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](mailto:mihir@eng.ucsd.edu)

Evgeny Alekseev  
CryptoPro  
18, Suschevsky val  
Moscow 127018  
Russian Federation

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



Ekaterina Smyshlyaeva

CryptoPro

18, Sushevsky val

Moscow 127018

Russian Federation

Phone: +7 (495) 995-48-20

Email: [ess@cryptopro.ru](mailto:ess@cryptopro.ru)