                    Re-keying Mechanisms for Symmetric Keys
                         draft-irtf-cfrg-re-keying-02

Abstract

   If encryption is performed under a single key, there is a certain
   maximum threshold amount of data that can be safely encrypted.  This
   amount is called key lifetime.  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 based on hash functions and on block ciphers that can be
   used with such modes of operations as CTR, GCM, CCM, 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 December 7, 2017.

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

1.  Introduction

   If encryption is performed under a single key, there is a certain
   maximum threshold amount of data that can be safely encrypted.  This
   amount is called key lifetime and can be calculated from the
   following considerations:

   1.  Methods based on the combinatorial properties of used encryption
       mode

          [Sweet32] is an example of attack that is based on such
          methods.  These methods do not depend on the used block cipher
          permutation $E_{K}$.  Сommon encryption modes restrictions
          resulting from such methods are of order $2^{n/2}$.

   2.  Methods based on side-channel analysis issues

          In most cases these methods do not depend on the used
          encryption modes and weakly depend on the used block cipher
          features.  Restrictions resulting from these methods are
          usually the strongest ones.

   3.  Methods based on the properties of the used block cipher
       permutation $E_{K}$

          The most common methods of this type are linear and
          differential cryptanalysis [LDC].  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 $2^n$ and so can be
          excluded from consideration as they become trivial.

   Therefore, as soon as the total size of a plaintext processed with a
   single key reaches the key lifetime limitation, that key must be
   replaced.  A specific value of the key lifetime is determined in
   accordance with safety margin for protocol security and methods
   outlined above.

   Suppose L is a key lifetime limitation in some protocol P.  For
   simplicity, assume that all messages have the same length m.  Hence
   the number of messages q that can be processed with a single key K
   should be such that $m*q <= L$.  This can be depicted graphically as a
   rectangle with sides m and q which is enclosed by area L:

```
      +-----------------------+
      |                   L   |
      |  +--------m---------+  |
      |  |=================|  |
      |  |=================|  |
      |  q=================|  |        m*q <= L
      |  |=================|  |
      |  |=================|  |
      |  |=================|  |
      |  +-----------------+  |
      +-----------------------+
```
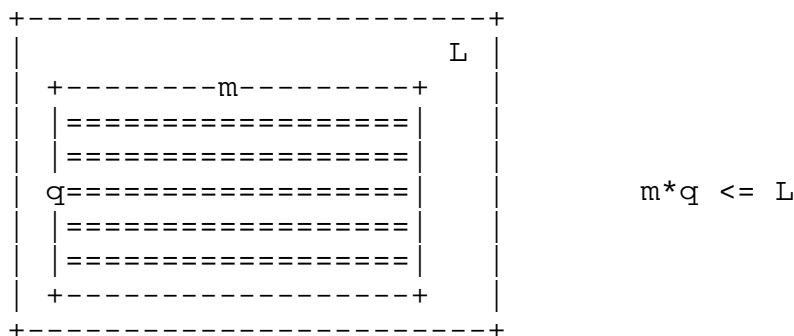
Thus, with increasing one of the parameters m or q, the second
parameter should be reduced in proportion to the first.

In practice, such amount of data that corresponds to limitation L may
not be enough.  The most simple and obvious way in this situation is
a regular renegotiation of a session key.  However, this reduces the
total performance since it usually entails termination of application
data transmission, additional service messages, the use of random
number generator and many other additional calculations, including
resource-intensive asymmetric cryptography.

This specification presents the description of two approaches that
allow to avoid renegotiation by extending the key lifetime for a
single agreed key: external and internal re-keying.  External re-
keying is chosen at a protocol level independently of a block cipher
and encryption mode, while an internal re-keying is chosen depending
on block and key sizes of a used block cipher and its encryption
mode.

The re-keying approaches extend the key lifetime for a single agreed
key by providing the possibility to strictly limit the key leakage
(to meet side channel limitations) and by improving combinatorial
properties of a used block cipher encryption mode.

Re-keying has already been applied in TLS 1.3 protocol.  As for
practical issues, re-keying can be particularly useful in such fields
as protocols functioning in hostile environments (additional side
channel resistance against DPA or EMI style attacks) or lightweight
cryptography (usage of ciphers with small block size leads to very
strong combinatorial limitations).  Moreover, many mechanisms that
use external and internal re-keying provide particular types of PFS
security.  Also re-keying can provide additional security against
possible future attacks on the used ciphers (by limiting the number
of plaintext-ciphertext pairs collected by an adversary), however, it
must not be used as a method to prolong life of ciphers that are
already known to be vulnerable.

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 =
            $(a\_s, \dots , a\_1)$, a in $V\_s$, into the integer $Int\_s(a)$ =
            $2^{\{s-1\}}*a\_s + \dots + 2*a\_2 + a\_1$;

    $Vec\_s$: $Z_{\{2^s\}}$ -> $V\_s$  the transformation inverse to the mapping
            $Int\_s$;

    $MSB\_i$: $V\_s$ -> $V\_i$  the transformation that maps the string a = $(a\_s,$
            $\dots , a\_1)$ in $V\_s$, into the string $MSB\_i(a)$ = $(a\_s, \dots ,$
            $a\_{\{s-i+1\}})$ in $V\_i$;

    $LSB\_i$: $V\_s$ -> $V\_i$  the transformation that maps the string a = $(a\_s,$
            $\dots , a\_1)$ in $V\_s$, into the string $LSB\_i(a)$ = $(a\_i, \dots ,$
            $a\_1)$ in $V\_i$;

    $Inc\_c$: $V\_s$ -> $V\_s$  the transformation that maps the string a = $(a\_s,$
            $\dots , a\_1)$ 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), k is multiple of 8;

n        the block size of the block cipher (in bits), n is multiple
         of 8;

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;

phi_i: V_s -> {0,1}  the transformation that maps a string a = (a_s,
         ... , a_1) into the value phi_i(a) = a_i for all i in {1, ...
         , s}.

A plaintext message P and a ciphertext C are divided into b =
ceil(|P|/n) blocks 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.  Principles of Choice of Constructions and Security Parameters

   External re-keying provides an approach, decording to which a key is
   transformed after encrypting a limited number of messages.  A
   specific external re-keying method is chosen at the protocol level
   regardless of a used block cipher or encryption mode.  External re-
   keying approach is recommended for usage in protocols that process
   quite small messages.

   The use of external re-keying has the following advantages:

   1.  the lifetime of a negotiated key drastically increases by
       increasing the number of messages processed with one key;

   2.  it almost does not affect performance in case when a number of
       messages processed with one key is sufficiently large;

   3.  provides forward and backward security of session keys for all
       messages.

   Homever, the use of external re-keying has the following
   disadvantages:

1.  in case of restrictive key lifetime limitations the message sizes
    can become inconvenient due to impossibility of processing
    sufficiently large messages, so it could be necessary to perform
    additional fragmentation at the protocol level;

2.  it is not transparent: procedures (like IVs generation) must be
    handled separately.

Internal re-keying provides an approach according to which a key is
transformed during each separate message processing.  Such approaches
are integrated into the base modes of operations so every internal
re-keying mechanism is defined for a particular mode and block cipher
(e.g. depending of block and key sizes).  Internal re-keying approach
is recommended to be used in protocols that process large messages.

The use of internal re-keying has the following advantages:

1.  the lifetime of a negotiated key drastically increases by
    increasing the size of messages processed with one key;

2.  it almost does not affect performance for long messages;

3.  internal re-keying mechanisms without master key does not affect
    short messages transformation at all;

4.  transparent (works like any encryption mode): does not require
    changes of IV's and restarting MACing.

Homever, the use of internal re-keying has the following
disadvantages:

1.  a specific method must not be chosen independently of a mode of
    operation;

2.  internal re-keying mechanisms with master key provide backward
    security of sessin keys only for one separate message;

3.  internal re-keying mechanisms without master key do not provide
    backward security of session keys.

Any block cipher modes of operations with internal re-keying can be
jointly used with any external re-keying mechanisms.  Such joint
usage increases both the number of messages processed with one key
and their maximum possible size.

The use of the same cryptographic primitives both for data processing
and re-keying transformation decreases the code size but can lead to
some possible vulnerabilities because the adversary always have an

access to the data processing interface.  This vulnerability can be eliminated by using different primitives for data processing and re-keying, however, in this case the security of the whole scheme cannot be reduced to standard notions like PRF or PRP so security estimations become more difficult and unclear.

5.  External Re-keying Mechanisms

This section presents an approach to increase the key lifetime by using a transformation of a previously negotiated key 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" mechanisms in this document.

External re-keying mechanisms are recommended for usage in protocols that process quite small messages (e.g.  TLS records are $2^{14}$ bytes or less) since the maximum gain in increasing the key lifetime is achieved by increasing the number of messages.

External re-keying increases the key lifetime through the following approach.  Suppose there is a protocol P with some mode of operation (base encryption or authentication mode).  Let L1 be a key lifetime limitation induced by side-channel analysis methods (side-channel limitation), let L2 be a key lifetime limitation induced by methods based on the combinatorial properties of used mode of operation (combinatorial limitation) and let q1, q2 be the total numbers of messages of length m, that can be safely processed with a single key K according to these limitations.

As L1 limitation is usually much stronger then L2 limitation (L1 < L2), the final key lifetime restriction is equal to the most restrictive limitation L1.  Thus, without re-keying only q1 (q1*m <= L1 ) messages can be safely processed.

```
            <--------m------->
            +----------------+  ^  ^
            |================|  |  |
            |================|  |  |
     K-->|================|  q1|  |
            |================|  |  |
            |============L1|  |  |
            +----------------+  v  |
            |                |     |
            |                |     |
            |                |   q2
            |                |     |
            |                |     |
            |                |     |
            |                |     |
            |                |     |
            |                |     |
            |                |     |
            |                |     |
            |             L2|     |
            +----------------+     v
```
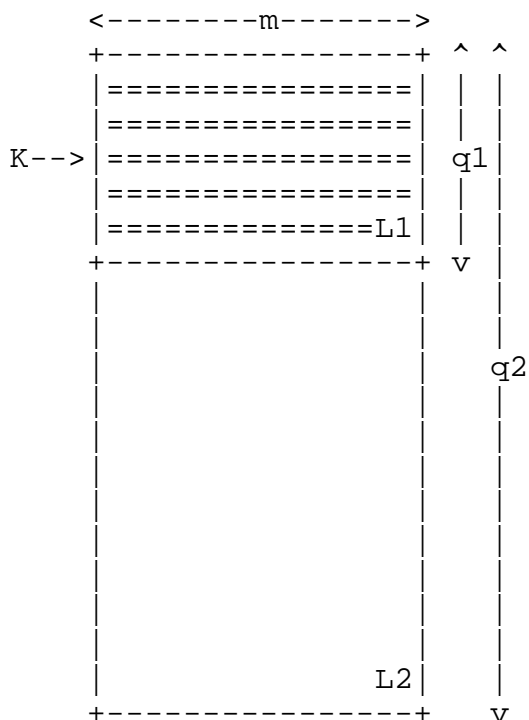
Suppose that the safety margin for the protocol P is fixed and the
external re-keying approach is applied.  As the key is transformed
with an external re-keying mechanism, the leakage of a previous key
does not have any impact on the following one, so the side channel
limitation L1 goes off.  Thus, the resulting key lifetime limitation
of the negotiated key K can be calculated on the basis of a new
combinatorial limitation L2'.  It is proven (see [AbBell]) that the
security of the mode of operation that uses external re-keying leads
to an increase when compared to base mode without re-keying (thus, L2
< L2').  Hence the resulting key lifetime limitation in case of using
external re-keying can be increased up to L2'.

```
                 <-------m------->
               +----------------+
               |===============|
               |===============|
   K--->       |===============|
    |          |===============|
    |          |============L1|
    |          +----------------+
    |          |===============|
    v          |===============|
  K^2-->       |===============|
    |          |===============|
    |          |============L1|
    |          +----------------+
    |          |===============|
    v          |===============|
  ...          |      . . .     |
               |                |
               |                |
               |             L2|
               +----------------+
               |             L2'|
               +----------------+
```
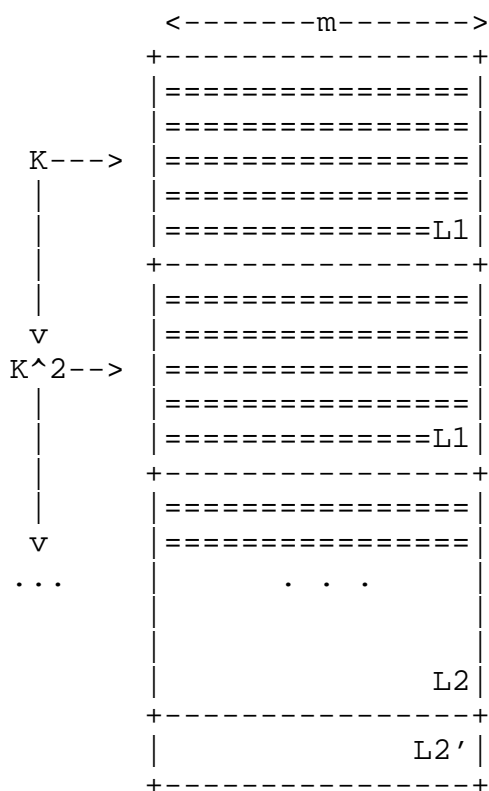
Note: the key transformation process is depicted in a simplified
form.  A specific approach (parallel and serial) is described below.

Consider an example.  Let the message size in protocol P be equal to
1 KB.  Suppose L1 = 128 MB and L2 = 1 TB.  Thus, if an external re-
keying mechanism is not used, the key K must be renegotiated after
processing 128 MB / 1 KB = 131072 messages.

If an external re-keying mechanism is used, the key lifetime
limitation L1 goes off.  Hence the resulting key lifetime limitation
in case of using external re-keying can be set to 1 TB (and even
more).  Thus if an external re-keying mechanism is used, then 1 TB /
1 KB = 2^30 messages can be processed before the master key K is
renegotiated.  This is 8192 times greater than the number of messages
that can be processed, when external re-keying mechanism is not used.

5.1.  Methods of Key Lifetime Control

Suppose L is an amount of data that can be safely processed with one
key (without re-keying).  For i in {1, 2, ..., t} the key $K^i$ (see
Figure 1 and Figure 2) should be transformed after processing $q_i$

integral messages, where $q_i$ can be calculated in accordance with one of the following two approaches:

o  Explicit approach:
   $|M^{i,1}| + ... + |M^{i,q_i}| <= L$, $|M^{i,1}| + ... + |M^{i,q_i + 1}| > L$.
   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:
   $q_i = L / m\_max$, $i = 1, ... , t$.
   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).

5.2.  Parallel Constructions

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

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} |==========       |
```

_____
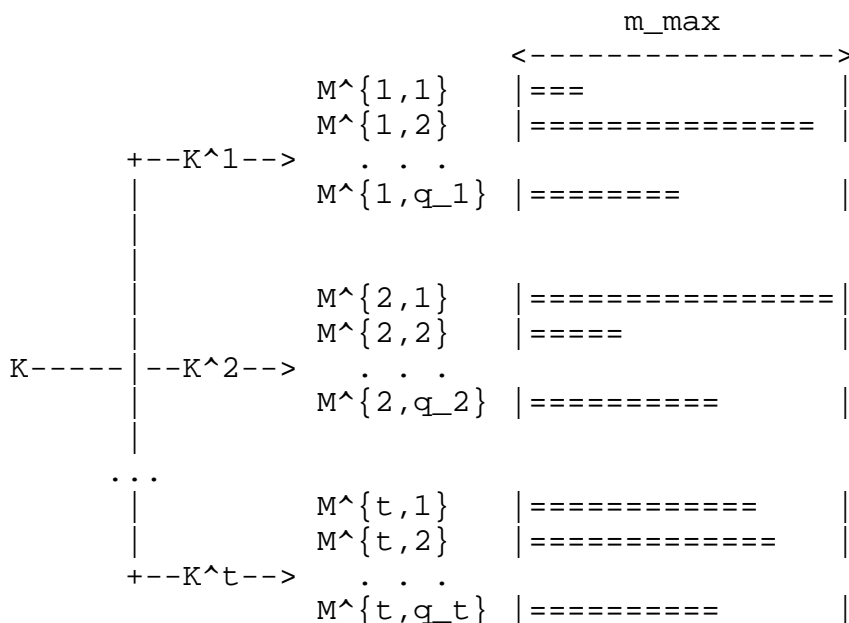
        Figure 1: External parallel re-keying mechanisms


The key $K^i$, i = 1, ... , t-1, is updated after processing a certain amount of data (see Section 5.1).

5.2.1.  Parallel Construction Based on a KDF on a Block Cipher

   ExtParallelC re-keying mechanism is based on key derivation function 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) =
   $MSB_{t*k}(E_{K}(0)$ | $E_{K}(1)$ | ... | $E_{K}(R-1))$,

   where R = ceil(t*k/n).

5.2.2.  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 | 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.

## 5.3.  Serial Constructions

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

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}  |==========         |


_____
```
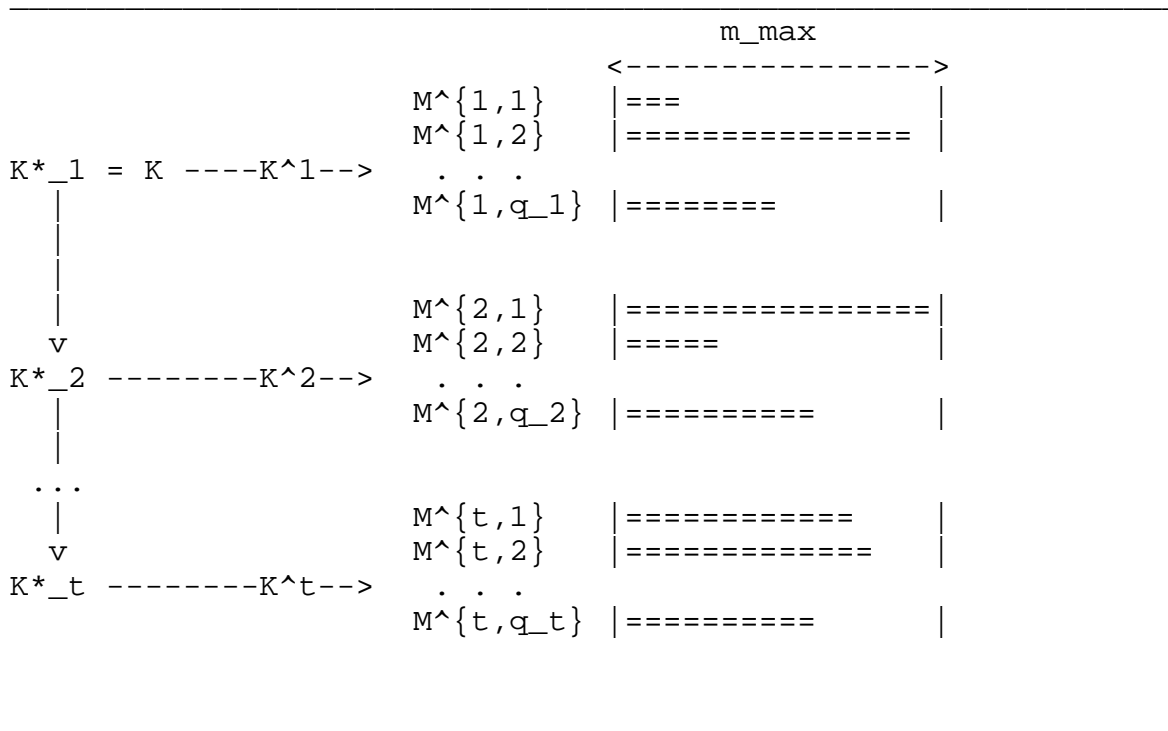
        Figure 2: External serial re-keying mechanisms

The key K^i, i = 1, ... , t-1, is updated after processing a certain amount of data (see Section 5.1).

## 5.3.1.  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 $J = \text{ceil}(k/n)$, $i = 1, \ldots, t$, $K^*\_i$ 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 $j = 1, \ldots, t-1$.

## 5.3.2.  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 $i = 1, \ldots, t$, 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]).

## 6.  Internal Re-keying Mechanisms

This section presents an approach to increase the key lifetime by using a transformation of a previously negotiated key during each separate message processing.

It provides internal re-keying mechanisms called ACPKM (Advanced cryptographic prolongation of key material) and ACPKM-Master that do not use and use a master key respectively.  Such mechanisms are integrated into the base modes of operations and actually form new modes of operation, therefore they are called "internal re-keying" mechanisms in this document.

Internal re-keying mechanism is recommended to be used in protocols that process large single messages (e.g.  CMS messages) since the maximum gain in increasing the key lifetime is achieved by increasing the length of a message, while it almost does not affect performance for increasing the number of messages.

Internal re-keying increases the key lifetime through the following approach.  Suppose there is a protocol P with some base mode of operation.  Let L1 and L2 be a side channel and combinatorial

limitations respectively and for some fixed amount of messages q let
m1, m2 be the length of each separate message, that can be safely
processed with a single key K according to these limitations.

Thus, by analogy with the Section 5 without re-keying the final key
lifetime restriction is equal to L1 and only q messages of the length
m1 can be safely processed.

```
              K
              |
              v
  ^  +---------------+--------------------------------+
  |  |============L1|                              L2|
  |  |===============|                                |
  q  |===============|                                |
  |  |===============|                                |
  |  |===============|                                |
  v  +---------------+--------------------------------+
     <-------m1------>
     <---------------------m2---------------------->
```

Suppose that the safety margin for the protocol P is fixed and
internal re-keying approach is applied to the base mode of operation.
Suppose further that for every message the key is transformed after
processing N bits of data, where N is a parameter.  If q*N does not
exceed L1 then the side channel limitation L1 goes off and the
resulting key lifetime limitation of the negotiated key K can be
calculated on the basis of a new combinatorial limitation L2'.  The
security of the mode of operation that uses external re-keying must
lead to an increase when compared to base mode of operation without
re-keying (thus, L2 < L2').  Hence the resulting key lifetime
limitation in case of using external re-keying can be increased up to
L2'.

```
              K ------------> K^2 ----------> . . .
              |                |
              v                v
    ^  +---------------+---------------+------------------+----+
    |  |============L1|============L1|======          L2| L2'|
    |  |============== |============== |======            |    |
    q  |============== |============== |======  . . .     |    |
    |  |============== |============== |======            |    |
    |  |============== |============== |======            |    |
    v  +---------------+---------------+------------------+----+
       <-------N-------->
```

Note: the key transformation process is depicted in a simplified
form.  A specific approach (ACPKM and ACPKM-Master re-keying
mechanisms) is described below.

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.  Suppose L1 = 128 MB and L2 = 10 TB.  Let the
message size in the protocol be large/unlimited (may exhaust the
whole key lifetime L2').  The most restrictive resulting key lifetime
limitation is equal to 128 MB.

Thus, there is a need to put a limit on the 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, more then L1 / N = 128 MB / 1 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).  Note that only one section of each message is
processed with one key K^i, and, consequently, the key lifetime
limitation L1 goes off.  Hence the resulting key lifetime limitation
in case of using external re-keying can be set to at least 10 TB (in
the case when the single large message is processed using the key K).

6.1.  Methods of Key Lifetime Control

Suppose L is an amount of data that can be safely processed with one
key (without re-keying), N is a section size (fixed parameter).
Suppose M^{i}_1 is the first section of message M^{i}, i = 1, ... , q

(see Figure 3 and Figure 4), then the parameter q can be calculated
in accordance with one of the following two approaches:

o  Explicit approach:
   $|M^{\{1\}}\_1| + ... + |M^{\{q\}}\_1| <= L, |M^{\{1\}}\_1| + ... + |M^{\{q+1\}}\_1| >$
   L
   This approach allows to use the key K^i in an 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 = 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).

## 6.2.  Constructions that Do Not Require Master Key

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

### 6.2.1.  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 base
   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 choice of 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.

_____

```
                  ACPKM         ACPKM                    ACPKM
          K^1 = K ---> K^2 ---...-> K^{l_max-1} ----> K^{l_max}
               |             |                |            |
               |             |                |            |
               |             |                |            |
               v             v                v            v
   M^{1} |==========|==========| ... |==========|=======:   |
   M^{2} |==========|==========| ... |===            |      :   |
     .        .          .         .       .           .    :
     :        :          :         :       :           :    :
   M^{q} |==========|==========| ... |==========|=====   :   |
                   section                                  :
               <---------->                          m_max
                  N bit
```

_____

l_max = ceil(m_max/N).

                     Figure 3: Internal  re-keying 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) sections (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 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} = ACPKM(K^i) = MSB_k(E_{K^i}(D_1) | ... | E_{K^i}(D_J)),

where J = ceil(k/n), parameter c is fixed by a specific encryption
mode which uses ACPKM key transformation 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 = ( 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87
    | 88 | 89 | 8a | 8b | 8c | 8d | 8e | 8f
    | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97
    | 98 | 99 | 9a | 9b | 9c | 9d | 9e | 9f
    | a0 | a1 | a2 | a3 | a4 | a5 | a6 | a7
    | a8 | a9 | aa | ab | ac | ad | ae | af
    | b0 | b1 | b2 | b3 | b4 | b5 | b6 | b7
    | b8 | b9 | ba | bb | bc | bd | be | bf
    | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7
    | c8 | c9 | ca | cb | cc | cd | ce | cf
    | d0 | d1 | d2 | d3 | d4 | d5 | d6 | d7
    | d8 | d9 | da | db | dc | dd | de | df
    | e0 | e1 | e2 | e3 | e4 | e5 | e6 | e7
    | e8 | e9 | ea | eb | ec | ed | ee | ef
    | f0 | f1 | f2 | f3 | f4 | f5 | f6 | f7
    | f8 | f9 | fa | fb | fc | fd | fe | ff )
```

N o t e : The constant D is such that $D\_1, \ldots , D\_J$ are pairwise different for any allowed n, k values.

N o t e : The constant D is such that $phi\_c(D\_t) = 1$ for any allowed n, k, c and t in $\{1, \ldots , J\}$.  This condition is important, as it allows to prevent collisions of blocks of transformed key and block cipher permutation inputs.

## 6.2.2.  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   $64 <= n <= 512$;

o   $128 <= k <= 512$;

o   the number of bits c in a specific part of the block to be incremented is such that $16 <= c <= 3/4 \, n$, c is multiple of 8.

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. P = CTR-ACPKM-Encrypt(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 MUST NOT exceed $n * 2^{c-1}$ bits.

6.2.3.  GCM-ACPKM Encryption Mode

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

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

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

   o  n in {128, 256};

   o  128 <= k <= 512;

   o  the number of bits c in a specific part of the block to be
      incremented is such that 32 <= c <= 3/4 n, c is multiple of 8.;

   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 | ... | 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 plaintext size 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.

6.2.4.  CCM Mode Key Meshing

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

   The CCM-ACPKM mode can be considered as the extended by the ACPKM re-
   keying mechanism basic authenticated encryption mode CCM (see
   [RFC3610]).

Since [RFC3610] defines CCM authenticated encryption mode only for
128-bit block size, the CCM-ACPKM authenticated encryption mode can
be used only with the parameter n = 128.  However, the CCM-ACPKM
design principles can easily be applied to other block sizes, but
these modes will require their own specifications.

The CCM-ACPKM authenticated encryption mode differs from CCM mode in
keys that are used for encryption during CBC-MAC calculation (see
Section 2.2 of [RFC3610]) and key stream blocks generation (see
Section 2.3 of [RFC3610]).

The CCM mode uses the same initial key K block cipher encryption
operations, while the CCM-ACPKM mode uses the keys $K^1$, $K^2$, ...,
which are generated from the key K as follows:

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

The keys $K^1$, $K^2$, ..., which are used as follows.

CBC-MAC calculation: under a separate message processing during the
first N/n block cipher encryption operations the key $K^1$ is used, the
key $K^2$ is used for the next N/n block cipher encryption operations
and so on.  For example, if N = 2n, then CBC-MAC calculation for a
sequence of t blocks $B_0$, $B_1$, ..., $B_t$ is as follows:

        $X_1 = E( K^1, B_0 )$,
        $X_2 = E( K^1, X_1 XOR B_1 )$,
        $X_3 = E( K^2, X_2 XOR B_2 )$,
        $X_4 = E( K^2, X_3 XOR B_3 )$,
        $X_5 = E( K^3, X_4 XOR B_4 )$,
        ...
        $T = first-M-bytes( X_{t+1} )$

The key stream blocks generation: under a separate message processing
during the first N/n block cipher encryption operations the key $K^1$
is used, the key $K^2$ is used for the next N/n block cipher encryption
operations and so on.  For example, if N = 2n, then the key stream
blocks are generated as follows:

        $S_0 = E( K^1, A_0 )$,
        $S_1 = E( K^1, A_1 )$,
        $S_2 = E( K^2, A_2 )$,
        $S_3 = E( K^2, A_3 )$,
        $S_4 = E( K^3, A_4 )$,
        ...

6.3.  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.

6.3.1.  ACPKM-Master Key Derivation 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 modes of operation (CTR, GCM, CBC,
   CFB, OFB, OMAC modes) for getting an extension of this modes of
   operations that uses periodical key transformation with master key.
   This extension can be considered as a new mode of operation .

   Additional parameters that defines the functioning of basic modes of
   operation 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.

_____
_____

```
                         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
M^{1}||=========|...|=========||=========|...|=========||...||=========|...|==
  : ||
M^{2}||=========|...|=========||=========|...|=========||...||=========|...|=====
=: ||
  ...  ||        |    |         ||       |    |         ||   ||       |    |
  : ||
M^{q}||=========|...|=========||====    |...|          ||...||        |...|
  : ||
         section
  :
     <-------->
  :
        N bit                                                               m
_max
```
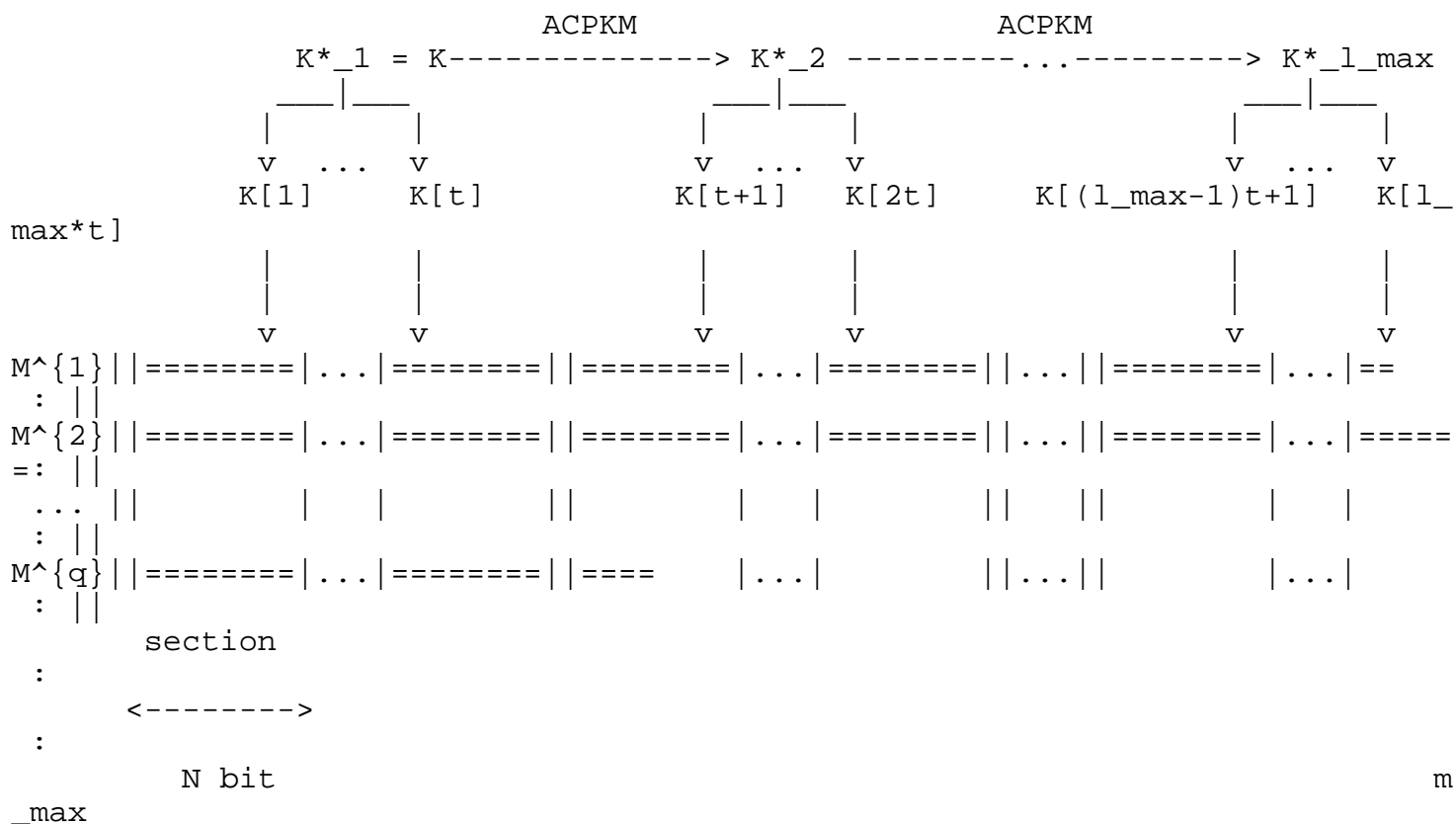
_____
_____
|K[i]| = d,
t = T*/d,
l_max = ceil(m_max/N).

Figure 4: Key meshing with master key

During the processing of the input message M with the length m in
some mode of operation 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) sections (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 of each message 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:

    K[1] | ... | K[l] = ACPKM-Master(T*, K, d*l) = CTR-ACPKM-Encrypt
    (T*, K, 1^{n/2}, 0^{d*l}).

6.3.2.  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 <= n <= 512;

o   128 <= k <= 512;

o   the number of bits c in a specific part of the block to be incremented is such that 32 <= c <= 3/4 n, c is multiple of 8.

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

6.3.3.  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 <= k <= 512;

   o  the number of bits c in a specific part of the block to be
      incremented is such that $32 <= c <= 3/4\ n$, c is multiple of 8;

   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:

      $K^1 \mid ... \mid K^j \mid ... \mid K^l$ = 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 plaintext size MUST NOT exceed $(2^{n/2-1}*n*N / k)$ bits.

6.3.4.  CCM Mode Key Meshing

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

   The CCM-ACPKM-Master authenticated encryption mode is differed from
   CCM-ACPKM mode in the way the keys K^1, K^2, ... are generated.  For
   CCM-ACPKM-Master mode the keys are generated as follows: K^i = K[i],
   where |K^i|=k and K[1]|K[2]|...|K[l] = ACPKM-Master( T*, K, k*l ).

6.3.5.  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 <= n <= 512;

   o  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 MUST NOT exceed $(2^{n/2-1}*n*N / k)$ bits.

6.3.6.  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 <= n <= 512;

   o  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 MUST NOT exceed $2^{n/2-1}*n*N/k$ bits.

6.3.7.  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 <= n <= 512;

   o  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 MUST NOT exceed $2^{n/2-1}*n*N / k$ bits.

6.3.8.  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 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 <= 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(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 << 1                                      |
|          else                                                    |
|               return (K1 << 1) (xor) R_n                          |
|                                                                   |
+-------------------------------------------------------------------+
```

Where R_n takes the following values:

o  n = 64: R_{64} = 0^{59} | 11011;

o  n = 128: R_{128} = 0^{120} | 10000111;

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

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

7.  Joint Usage of External and Internal Re-keying

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

   Re-keying is not intended to solve any post-quantum security issues
   for symmetric crypto since the reduction of security caused by
   Grover's algorithm is not connected with a size of plaintext
   transformed by a cipher - only a negligible (sufficient for key
   uniqueness) material is needed and the aim of re-keying is to limit a
   size of plaintext transformed on one key.

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.

   [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>.

   [RFC3610]   Whiting, D., Housley, R., and N. Ferguson, "Counter with
               CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September
               2003, <http://www.rfc-editor.org/info/rfc3610>.

   [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>.

   [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.

   [LDC]       Howard M. Heys, "A Tutorial on Linear and Differential
               Cryptanalysis", 2017,
               <http://www.cs.bc.edu/~straubin/crypto2017/heys.pdf>.

   [Sweet32]  Karthikeyan Bhargavan, Gaetan Leurent, "On the Practical
              (In-)Security of 64-bit Block Ciphers. Collision Attacks
              on HTTP over TLS and OpenVPN", 2016,
              <https://sweet32.info/SWEET32_CCS16.pdf>.

Appendix A.  Test examples


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


   W_0
   80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F


   W_1
   90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F


   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

Input block (ctr)
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

Output block (ctr)
F6 80 D1 21 2F A4 3D F4 EC 3A 91 DE 2A B1 6F 1B

Input block (ctr)
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

Output block (ctr)
36 B0 48 8A 4F C1 2E 09 98 D2 E4 A8 88 E8 4F 3D

Updated key:
F6 80 D1 21 2F A4 3D F4 EC 3A 91 DE 2A B1 6F 1B
36 B0 48 8A 4F C1 2E 09 98 D2 E4 A8 88 E8 4F 3D

ACPKM's iteration 2
Process block 1
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 02

Output block (ctr)
E4 88 89 4F B6 02 87 DB 77 5A 07 D9 2C 89 46 EA

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

Cipher text
F5 AA BA 0B E3 64 F0 53 EE F0 BC 15 C2 76 4C EA

Process block 2
Input block (ctr)

```
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 03

Output block (ctr)
BC 4F 87 23 DB F0 91 50 DD B4 06 C3 1D A9 7C A4

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

Cipher text
9E 7C C3 76 BD 87 19 C9 77 0F CA 2D E2 A3 7C B5

Input block (ctr)
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

Output block (ctr)
8E B9 7E 43 27 1A 42 F1 CA 8E E2 5F 5C C7 C8 3B

Input block (ctr)
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

Output block (ctr)
1A CE 9E 5E D0 6A A5 3B 57 B9 6A CF 36 5D 24 B8

Updated key:
8E B9 7E 43 27 1A 42 F1 CA 8E E2 5F 5C C7 C8 3B
1A CE 9E 5E D0 6A A5 3B 57 B9 6A CF 36 5D 24 B8

ACPKM's iteration 3
Process block 1
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 04

Output block (ctr)
68 6F 22 7D 8F B2 9C BD 05 C8 C3 7D 22 FE 3B B7

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

Cipher text
5B 2B 77 1B F8 3A 05 17 BE 04 2D 82 28 FE 2A 95

Process block 2
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 05

Output block (ctr)
C0 1B F9 7F 75 6E 12 2F 80 59 55 BD DE 2D 45 87
```

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

Cipher text
84 4E 9F 08 FD F7 B8 94 4C B7 AA B7 DE 3C 67 B4

Input block (ctr)
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

Output block (ctr)
C5 71 6C C9 67 98 BC 2D 4A 17 87 B7 8A DF 94 AC

Input block (ctr)
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

Output block (ctr)
E8 16 F8 0B DB BC AD 7D 60 78 12 9C 0C B4 02 F5

Updated key:
C5 71 6C C9 67 98 BC 2D 4A 17 87 B7 8A DF 94 AC
E8 16 F8 0B DB BC AD 7D 60 78 12 9C 0C B4 02 F5

ACPKM's iteration 4
Process block 1
Input block (ctr)
12 34 56 78 90 AB CE F0 00 00 00 00 00 00 00 06

Output block (ctr)
03 DE 34 74 AB 9B 65 8A 3B 54 1E F8 BD 2B F4 7D

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

Cipher text
56 B8 43 FC 32 31 DE 46 D5 AB 14 F8 AC 09 C7 39

Input block (ctr)
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F

Output block (ctr)
74 1E B5 88 D6 AB DA B6 89 AA FD BA A9 3E A2 46

Input block (ctr)
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F

Output block (ctr)
16 3A A6 C2 3C E7 C3 74 CD 38 BF C6 FE 8C C5 FF

```
Updated key:
74 1E B5 88 D6 AB DA B6 89 AA FD BA A9 3E A2 46
16 3A A6 C2 3C E7 C3 74 CD 38 BF C6 FE 8C C5 FF

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
F5 AA BA 0B E3 64 F0 53 EE F0 BC 15 C2 76 4C EA
9E 7C C3 76 BD 87 19 C9 77 0F CA 2D E2 A3 7C B5
5B 2B 77 1B F8 3A 05 17 BE 04 2D 82 28 FE 2A 95
84 4E 9F 08 FD F7 B8 94 4C B7 AA B7 DE 3C 67 B4
56 B8 43 FC 32 31 DE 46 D5 AB 14 F8 AC 09 C7 39
```

Appendix B.  Contributors

   o  Russ Housley
      Vigil Security, LLC
      housley@vigilsec.com

   o  Mihir Bellare
      University of California
      mihir@eng.ucsd.edu

   o  Evgeny Alekseev
      CryptoPro
      alekseev@cryptopro.ru

   o  Ekaterina Smyshlyaeva
      CryptoPro
      ess@cryptopro.ru

   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

Appendix C.  Acknowledgments

We thank Scott Fluhrer, Dorothy Cooley, Yoav Nir, Jim Schaad and Paul
Hoffman for their useful comments.

Author's Address

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

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