ETSI TS 135 236 V19.0.0 (2026-01)

TECHNICAL SPECIFICATION

5G;
Specification of the MILENAGE-256 algorithm set;
An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5, f5* and f5;**
Document 3: Implementors' test data and design conformance test data
(3GPP TS 35.236 version 19.0.0 Release 19)

Reference
DTS/TSGS-0335236vj00

Keywords
5G,SECURITY

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is (or any other verb in the indicative mood) indicates a statement of fact

is not (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

Introduction

The present document contains a 256-bit example of set of algorithms, collectively called MILENAGE-256, which may be used as the authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**. It is not mandatory to use the particular algorithms specified in this document – all eight functions are operator-specifiable rather than being fully standardised. Operators electing to employ this example set can further personalise the algorithms (as described in the text).

The present document is one of four documents, which collectively comprise the entire specification of the example authentication and key generation algorithms. Namely:

- 3GPP TS 35.234 [2]: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 1: MILENAGE-256 General".
- 3GPP TS 35.235 [3]: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 2: MILENAGE-256 Algorithm Specification".
- **3GPP TS 35.236: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 3: Implementors' Test and Design Conformance Test Data".**
- 3GPP TR 35.937 [4]: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 4: Summary and Results of Design and Evaluation".

1 Scope

The present document provides test data (also known as test vectors) that implementors can use to help verify that their implementations are correct, according to the technical specifications of MILENAGE-256 algorithm set [2,3]. A reference implementation in C/C++ is also provided in the Annex.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document.*

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 35.234: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 1: MILENAGE-256 General".

[3] 3GPP TS 35.235: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 2: MILENAGE-256 Algorithm Specification".

[4] 3GPP TR 35.937: "Specification of the MILENAGE-256 algorithm set: An example set of 256-bit 3GPP authentication and key generation functions f1, f1*, f2, f2, f3, f5, f5, f5* and f5**; Document 4: Summary and Results of Design and Evaluation".

[5] 3GPP TS 33.102: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture".

[6] Rijndael information page, NIST archived AES submissions, https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto- projects/aes-development#rijndael

[7] The Advanced Encryption Standard (AES), NIST FIPS 197, 2001.

[8] J. Daemen and V. Rijmen, "The design of Rijndael", Springer Verlag, 2002.

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

AKA-specific terminology

AMF: Authentication Management Field

AK: Anonymity key

AK*: Anonymity key used during resynchronisation

CK: Cipher Key

f1, f1*, f2, f3, f4, f5, f5*, f5:** Cryptographic functions used to derive AKA parameters

IK: Integrity Key

K: Subscriber key

MAC-A: Network Authentication Code

MAC-S: Resynchronisation Authentication Code

RAND: Random Challenge

RES: Response to Challenge

SQN: Sequence Number

Additional terminology

AK_{SZ}: The length of the anonymity key **AK**, in octets

ALGONAME: An ASCII character string encoding of a name assigned for a particular instance/application of the MILENAGE-256 algorithm set instance

c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7: 128-bit operator-customisable constants, used during the computation of ***f1, f1*, f2, f3, f4, f5, f5**,*** and ***f5*****

CK_{SZ}: The length of the ciphering key response **CK**, in octets

$IN0, IN1, IN2, IN3, IN4, IN5, IN6, IN7$: 256-bit instance-specific input values constructed within the computation of the functions ***f1, f1*, f2, f3, f4, f5, f5**,*** and ***f5*****

IK_{SZ}: The length of the integrity key response **IK**, in octets

K_{SZ}: The length of the subscriber key **K**, in octets

MAC_{SZ}: The length of the message authentication codes **MAC-A**, and **MAC-S**, in octets

OP: A 256-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1, f1*, f2, f3, f4, f5, f5**** and ***f5*****

OP_C: A 256-bit value derived from *OP*, *ALGONAME*, K_{SZ} and **K,** and used within the computations of the functions ***f1, f1*, f2, f3, f4, f5, f5**** and ***f5*****

RES_{SZ}: The length of the response **RES**, in octets

V: A 256-bit intermediate value constructed from *ALGONAME* and K_{SZ}, and used in the computation of OP_C

NOTE: Bold variables above are part of the general AKA specification [8]. Additional explanation of the usage of boldface, italics, etc within MILENAGE-256 appears in the MILENAGE-256 Algorithm Specification [7]. In the printout of test data, values are printed in courier typeface and indices are indicated by an underscore.

EXAMPLE: Values corresponding to K_{SZ} are shown as *K_sz*.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

=	The assignment operator
:=	The definition operator
$\oplus$	The bitwise exclusive-OR operation
{ }	Brackets are used to indicate a value given as a byte-array EXAMPLE: X = { $X[0]$ $X[1]$ … $X[m]$ }, where each $X[j]$ is a byte

1_A	The *n*-bit binary value 00…001, given as a byte array, i.e. $X = \{ X[0] \dots X[m] \}$, with $X[0] = 1$, and $X[j] = 0$ for $j > 0$
2_A	The *n*-bit binary value 00…010, given as a byte-array, i.e. $X = \{ X[0] \dots X[m] \}$, with $X[0] = 2$, and $X[j] = 0$ for $j > 0$
‖	The concatenation of two byte arrays. If $X = \{ X[0] \dots X[m] \}$ and $Y = \{ Y[0] \dots Y[n] \}$, then $X \| Y := \{ X[0] \dots X[m] Y[0] \dots Y[n] \}$.
AES-n	AES with n-bit key (and 128-bit block size)
PRF_K	Pseudo-random function defined by key K
Rijndael-b-n	Rijndael block cipher with b-bit block and n-bit key

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
MAC	Message Authentication Code
MDPH	Merkle-Damgård with Permutation and Hirose compression function
PRF	Pseudo-Random Function

3.4 Radix

All test data print-outs below are given in hexadecimal notation, but omitting the usual "0x"-prefix.

EXAMPLE: A given value a3 corresponds to the decimal value 10*16 + 3 = 163.

4 Structure

The test data provided in the present document comes in two categories. First, implementors' test data that can be used to verify the underlying block cipher component of MILENAGE-256, and test data for the PRF kernel constructed from block cipher. This includes also intermediate values produced during the computation of the kernel(s). This is sometimes refered to as "white-box" test data.

NOTE: For MILENAGE-256, the PRF-kernel is identical to the block cipher and thus only one common test-set is needed.

Input values for these test vectors were selected such that they change incrementally from one test to another, this strategy could be helpful for implementors to track possible errors in their implementation. There is also one test vector with pseudo random input values in order to catch other, unforeseen errors.

Next, conformance test data for the algorithm in the complete MILENAGE-256 algorithm set is provided, for different input and output parameter alternatives. This is sometimes referred to as "black-box" test data. The first test data for each input parameter set is more extensive and includes also intermediate values occurring during the computation.

This report is organised as follows:

Clause 5 provides detailed test data for implementors, including also intermediate values occurring during computation, i.e. related to the cryptographic kernels (PRFs);

Clause 6 provides design conformance data for MILENAGE-256. Principles for selecting test data is provided in sub-clause 6.1.

Source code for a reference implementation is provided in the Annex.

5 Implementors' test data

5.1 MILENAGE-256

5.1.1 General

The kernel of MILENAGE-256 is a PRF obtained directly as the Rijndael-256-256 block cipher. Therefore, test data for the PRF is identical to test data for Rijndael-256-256. Refer to the Algorithm Specification [3, clause 11] for details.

5.1.2 PRF based on Rijndael-256-256

```
=== PRF-RIJNDAEL-256-256 TEST #1 ===
        KEY = { 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
          X = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
  RoundKey0 = { 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
  RoundKey1 = { e2 63 63 63 e2 63 63 63 e2 63 63 63 e2 63 63 63
                98 fb fb fb 98 fb fb fb 98 fb fb fb 98 fb fb fb }
  RoundKey2 = { ef 6c 6c 25 0d 0f 0f 46 ef 6c 6c 25 0d 0f 0f 46
                4f 8d 8d a1 d7 76 76 5a 4f 8d 8d a1 d7 76 76 5a }
  RoundKey3 = { d3 54 d2 2b de 5b dd 6d 31 37 b1 48 3c 38 be 0e
                a4 8a 23 0a 73 fc 55 50 3c 71 d8 f1 eb 07 ae ab }
  RoundKey4 = { 1e b0 b0 c2 c0 eb 6d af f1 dc dc e7 cd e4 62 e9
                19 e3 89 14 6a 1f dc 44 56 6e 04 b5 bd 69 aa 1e }
  RoundKey5 = { f7 1c c2 b8 37 f7 af 17 c6 2b 73 f0 0b cf 11 19
                32 69 0b c0 58 76 d7 84 0e 18 d3 31 b3 71 79 2f }
  RoundKey6 = { 74 aa d7 d5 43 5d 78 c2 85 76 0b 32 8e b9 1a 2b
                2b 3f a9 31 73 49 7e b5 7d 51 ad 84 ce 20 d4 ab }
  RoundKey7 = { 83 e2 b5 5e c0 bf cd 9c 45 c9 c6 ae cb 70 dc 85
                34 6e 2f a6 47 27 51 13 3a 76 fc 97 f4 56 28 3c }
  RoundKey8 = { b2 d6 5e e1 72 69 93 7d 37 a0 55 d3 fc d0 89 56
                84 1e 88 17 c3 39 d9 04 f9 4f 25 93 0d 19 0d af }
  RoundKey9 = { 7d 01 27 36 0f 68 b4 4b 38 c8 e1 98 c4 18 68 ce
                98 b3 cd 9c 5b 8a 14 98 a2 c5 31 0b af dc 3c a4 }
 RoundKey10 = { cd ea 6e 4f c2 82 da 04 fa 4a 3b 9c 3e 52 53 52
                2a b3 20 9c 71 39 34 04 d3 fc 05 0f 7c 20 39 ab }
 RoundKey11 = { 16 f8 0c 5f d4 7a d6 5b 2e 30 ed c7 10 62 be 95
                e0 19 8e b6 91 20 ba b2 42 dc bf bd 3e fc 86 16 }
 RoundKey12 = { 7e bc 4b ed aa c6 9d b6 84 f6 70 71 94 94 ce e4
                c2 3b 05 df 53 1b bf 6d 11 c7 00 d0 2f 3b 86 c6 }
 RoundKey13 = { 37 f8 ff f8 9d 3e 62 4e 19 c8 12 3f 8d 5c dc db
                9f 71 83 66 cc 6a 3c 0b dd ad 3c db f2 96 ba 1d }
 RoundKey14 = { ea 0c 5b 71 77 32 39 3f 6e fa 2b 00 e3 a6 f7 db
                8e 55 eb df 42 3f d7 d4 9f 92 eb 0f 6d 04 51 12 }
          Y = { e6 2a bc e0 69 83 7b 65 30 9b e4 ed a2 c0 e1 49
                fe 56 c0 7b 70 82 d3 28 7f 59 2c 4a 49 27 a2 77 }

=== PRF-RIJNDAEL-256-256 TEST  #2 ===
        KEY = { 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
                01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 }
          X = { 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
                01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 }
          Y = { f6 f9 7c 67 72 f2 04 88 e3 c0 ee c5 48 29 81 b2
                bd 00 b1 5b bd f9 40 06 9f bf 51 42 ce b3 96 88 }
```

```
=== PRF-RIJNDAEL-256-256 TEST #3 ===
       KEY = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
         X = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
         Y = { c6 22 7e 77 40 b7 e5 3b 5c b7 78 65 27 8e ab 07
               26 f6 23 66 d9 aa ba d9 08 93 61 23 a1 fc 8a f3 }

=== RIJNDAEL-256-256 TEST #4 ===
       KEY = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
         X = { 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
         Y = { 15 9a 08 e4 6e 61 6e 6e 99 78 50 20 10 da ff 92
               2e b3 62 e7 7d ca af 02 ea eb 73 54 eb 8b 8d ba }

=== RIJNDAEL-256-256 TEST #5 ===
       KEY = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
         X = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 }
         Y = { 4e 76 ca 69 96 71 25 a9 63 6f 35 54 22 95 56 f6
               e2 b2 35 1c b4 fd 10 b4 e0 52 af d8 5b eb df a8 }
```

=== PRF-RIJNDAEL-256-256 TEST #6 ===

```
       KEY = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
         X = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
         Y = { f3 6c b6 c7 a7 57 2f 19 30 7a 31 e4 ec 4c a4 c8
               2d 27 31 fb 21 f5 9c af 13 3f e8 16 a5 44 24 a5 }

=== PRF-RIJNDAEL-256-256 TEST #7 ===
       KEY = { 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
               10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f }
         X = { 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
               10 21 32 43 54 65 76 87 98 a9 ba cb dc ed fe 0f }
         Y = { 28 8f a9 d2 3d 00 d9 dc 0a 39 b3 3f a9 28 67 c6
               48 8b 5e 0f 18 a6 f7 4c 07 20 78 ec 81 54 62 e6 }

=== PRF-RIJNDAEL-256-256 TEST #8 ===
       KEY = { 0f 3a 40 4e b7 f3 49 d4 7b e0 0f 1c 19 df c9 0a
               3a 04 0a 79 96 30 fe 74 5f e5 f0 a9 dd 58 10 1a }
         X = { 56 c8 74 16 30 0f 5c 68 59 77 ba ce e7 ce f2 ad
               fc bd de 02 ed 55 15 9c a3 eb 6d 89 41 26 be ce }
         Y = { a5 f0 c2 bb a8 b1 31 cb 82 ea 65 4e ba b1 52 21
               3d f5 0b d1 84 f4 87 b2 2d 5a ea b8 7c c3 b0 24 }
```

6 Design conformance test data

6.1 Test data selection principles

The overall MILENAGE-256 construct has a huge number of possible instantiations based on exact choices of the parameter sizes (*KEYsz*, *RESsz*, …), and it is impossible to provide test vectors for each combination. The strategy has instead been to provide test vectors for a few selections that are, with high confidence, providing assurance in the correctness of an implementation.

There are five main test-sets, each first selecting the different inputs, i.e. the input parameter sizes, the input parameter values, and values for the operator configurable constants, having the following structure:

Set	KEY_{sz}	$RAND_{sz}$	SQN_{sz}	$c_0..c_7$	KEY/OP/RAND/ SQN/AMF	Set characterisation
#1	32	32	12	All distinct	All distinct	Maximum
#2	16	16	6	All distinct	All distinct	Minimum
#3	32	22	9	All ones	All ones	Odd
#4	32	16	6	Default values	All random	Expected
#5	32	32	6	All zeroes	All random	Desired

Then, for each of the above input selections, the following five sub-tests, selecting the five output parameters sizes, are provided as follows:

Subset	RES_{sz}	CK_{sz}	IK_{sz}	MAC_{sz}	AK_{sz}	Set characterisation
a	32	32	32	32	12	Maximum (includes also debug info)
b	4	16	16	8	6	Minimum
c	7	29	17	23	9	Odd
d	8	32	32	8	6	Expected
e	32	32	32	16	6	Desired

The sets "Maximum" and "Minimum" adopt the maximum and minimum allowed sizes of the corresponding parameters, respectively. Sets "Odd" check borderline unusual sizes. Sets "Expected" and "Desired" are two variants of what could likely be used in practice.

Thus, there are in total 25 test vectors, where five of them, the tests #{1-5}a, additionally include details of intermediate values for debugging purposes.

6.2 Reference figure for identification of printed data

For the purpose of this clause, the labels employed in the printed data below can be identified by reference to the following figure.

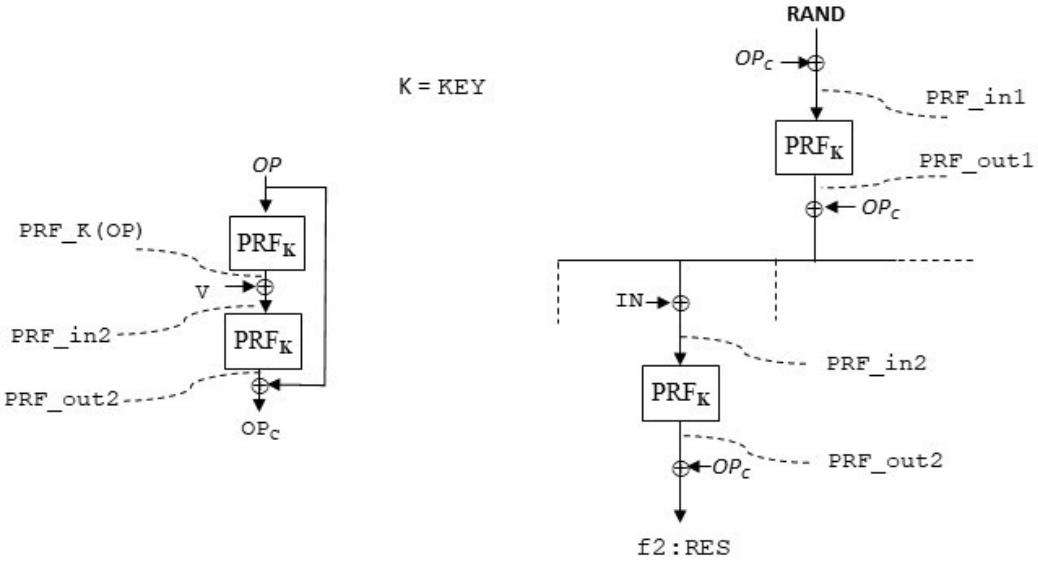

Figure 6.2-1: Reference figure for identification of printed data. For simplicity, only *f2* is shown.

In all cases, data has been produced using *ALGONAME* = "MILENAGE2.0" as defined in the Algorithm Specification [2].

NOTE: Some combinations of parameter-values will most likely not be encountered in practice, such as values with $SQNsz \neq AKsz$, since these two parameters are typically identical. Nevertheless, such values are provided below for completeness.

6.3 Milenage-256

Tests #{1-5}a contain intermediate values and may be used for debugging purposes, other tests #{1- 5}{b-e} can be seen as conformance tests without intermediate values.

```
=== Milenage-256 TEST #1 ===
Inputs: KEY_sz=32, RAND_sz=32, SQN_sz=12
        OP = { 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 }
       KEY = { 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 }
      RAND = { 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 }
       SQN = { a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab }
       AMF = { b0 b1 }
        c0 = { 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f }
        c1 = { 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f }
        c2 = { 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f }
        c3 = { 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f }
        c4 = { 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f }
        c5 = { 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f }
        c6 = { 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f }
        c7 = { 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f }
Computation of OPc:
 PRF_K(OP) = { 72 f7 9b b5 d1 50 97 2f 91 53 18 41 5e c3 8b 12
               0a 1a 0d 81 31 5d e7 a8 0a 65 1d 7e 47 41 2c 2d }
         V = { 01 4d 49 4c 45 4e 41 47 45 32 2e 30 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 73 ba d2 f9 94 1e d6 68 d4 61 36 71 5e c3 8b 12
               0a 1a 0d 81 31 5d e7 a8 0a 65 1d 7e 47 41 2c 2d }
  PRF_out2 = { a0 51 ac 90 c4 a7 ea b0 ce e7 ac 48 58 e3 47 df
               d0 de 1d e8 55 29 4c 71 0f 1d 28 79 86 8e 4c 73 }
       OPc = { 60 90 6e 53 00 62 2c 77 06 2e 66 83 94 2e 89 10
               00 0f cf 3b 81 fc 9a a6 d7 c4 f2 a2 5a 53 92 ac }

Case #1a: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=32, AK_sz=12
Intermediates for the function index 0:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
               90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
               f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { 1f ff b0 b1 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab
               00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f }
   PRF_in2 = { 32 b6 8f c6 1d b1 78 8f e9 01 6d f7 7e 36 8a 2e
               f1 c8 33 8b 8c 20 99 63 0f 41 79 75 90 79 10 c3 }
  PRF_out2 = { 48 dc a0 bd ae f1 ce 83 9e 3b 25 fd c6 a7 48 1a
               ea 6f c7 ac 34 cc f5 d0 35 3c f3 38 11 e3 7a 71 }
 f1*:MAC_S = { 28 4c ce ee ae 93 e2 f4 98 15 43 7e 52 89 c1 0a
               ea 60 08 97 b5 30 6f 76 e2 f8 01 9a 4b b0 e8 dd }
Intermediates for the function index 1:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
               90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
               f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { 3f ff b0 b1 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab
               10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f }
   PRF_in2 = { 12 b6 8f c6 1d b1 78 8f e9 01 6d f7 7e 36 8a 2e
               e1 d8 23 9b 9c 30 89 73 1f 51 69 65 80 69 00 d3 }
  PRF_out2 = { f7 84 30 33 06 72 bd 99 b6 bc 1a b6 60 c4 98 64
               1e f0 bf 63 00 75 ba 0f 6d 0f ae ba 0e bd 96 d2 }
  f1:MAC_A = { 97 14 5e 60 06 10 91 ee b0 92 7c 35 f4 ea 11 74
               1e ff 70 58 81 89 20 a9 ba cb 5c 18 54 ee 04 7e }
Intermediates for the function index 2:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
               90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
               f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { 5f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f }
   PRF_in2 = { 72 56 3f 77 bd 10 da 2c 4d a4 cb 50 d6 9f 20 85
               d1 e8 13 ab ac 00 b9 43 2f 61 59 55 b0 59 30 e3 }
  PRF_out2 = { a3 9d fd 91 ab e0 8c da b7 fe a9 6e 72 a2 75 e5
               4a 45 40 4f 09 72 d2 31 70 4e 24 6b 94 07 61 42 }
    f2:RES = { c3 0d 93 c2 ab 82 a0 ad b1 d0 cf ed e6 8c fc f5
               4a 4a 8f 74 88 8e 48 97 a7 8a d6 c9 ce 54 f3 ee }
Intermediates for the function index 3:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
```

```
                90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
                f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { 7f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f }
   PRF_in2 = { 52 56 3f 77 bd 10 da 2c 4d a4 cb 50 d6 9f 20 85
                c1 f8 03 bb bc 10 a9 53 3f 71 49 45 a0 49 20 f3 }
  PRF_out2 = { d5 ed a0 4d 23 51 d1 f5 ff 9b 02 52 e7 eb fc 92
                78 5f 9e 05 0b ba 08 b1 d5 0d 3d 08 64 50 fe 10 }
     f3:CK = { b5 7d ce 1e 23 33 fd 82 f9 b5 64 d1 73 c5 75 82
                78 50 51 3e 8a 46 92 17 02 c9 cf aa 3e 03 6c bc }
Intermediates for the function index 4:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
                90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
                f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { 9f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f }
   PRF_in2 = { b2 56 3f 77 bd 10 da 2c 4d a4 cb 50 d6 9f 20 85
                b1 88 73 cb cc 60 d9 23 4f 01 39 35 d0 39 50 83 }
  PRF_out2 = { 71 26 b7 f0 ca ba b9 47 1d 90 24 be ad 3b 82 04
                c7 28 fc 5d 8f 40 14 2b e1 66 99 13 87 e8 1c 46 }
     f4:IK = { 11 b6 d9 a3 ca d8 95 30 1b be 42 3d 39 15 0b 14
                c7 27 33 66 0e bc 8e 8d 36 a2 6b b1 dd bb 8e ea }
Intermediates for the function index 5:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
                90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
                f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { bf 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f }
   PRF_in2 = { 92 4e 3f 77 bd 10 da 2c 4d a4 cb 50 d6 9f 20 85
                a1 98 63 db dc 70 c9 33 5f 11 29 25 c0 29 40 93 }
  PRF_out2 = { 97 cb e9 f1 88 62 b3 88 75 24 f8 ae 60 81 61 96
                dc ec 31 c7 89 72 5b fe 7a 93 0f 13 66 54 d1 24 }
     f5:AK = { f7 5b 87 a2 88 00 9f ff 73 0a 9e 2d }
Intermediates for the function index 6:
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
                90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
                f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { df 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f }
   PRF_in2 = { f2 4e 3f 77 bd 10 da 2c 4d a4 cb 50 d6 9f 20 85
                91 a8 53 eb ec 40 f9 03 6f 21 19 15 f0 19 70 a3 }
  PRF_out2 = { 27 dd 48 01 94 a0 6d f2 9f 8c 4c 2b 42 1d 3a 57
                79 03 78 b1 97 79 8a 5d 7c 3c f7 98 1a 60 0d 62 }
    f5*:AK* = { 47 4d 26 52 94 c2 41 85 99 a2 2a a8 }
Intermediates for the function index 7:
     MAC_S = { 28 4c ce ee ae 93 e2 f4 98 15 43 7e 52 89 c1 0a
                ea 60 08 97 b5 30 6f 76 e2 f8 01 9a 4b b0 e8 dd }
   PRF_in1 = { e0 11 ec d0 84 e7 aa f0 8e a7 ec 08 18 a3 07 9f
                90 9e 5d a8 15 69 0c 31 4f 5d 68 39 c6 ce 0c 33 }
  PRF_out1 = { 4d d9 51 24 bd 72 f6 5b 4b 8a ad d3 42 b1 a9 95
                f1 c6 fe b3 09 d9 05 c2 d0 8c 81 dc c6 27 8c 60 }
        IN = { ff ff 28 4c ce ee ae 93 e2 f4 98 15 43 7e 52 89
                b1 7b 98 13 7c e2 c3 47 17 0f 98 83 7d e7 35 cf }
   PRF_in2 = { d2 b6 17 3b 73 fe 74 bf af 50 53 45 95 e1 72 0c
                40 b2 a9 9b f4 c7 5c 23 10 47 eb fd e1 93 2b 03 }
  PRF_out2 = { 8a 54 40 ed 65 23 21 91 28 b1 f8 9b 2d 2a 66 fb
                16 2f ae f4 42 28 aa d4 29 a8 0d 35 85 0e 98 6f }
  f5**:AK* = { ea c4 2e be 65 41 0d e6 2e 9f 9e 18 }
Case #1b: RES_sz= 4, CK_sz=16, IK_sz=16, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { db 14 35 14 1d 60 77 dc }
  f1:MAC_A = { b8 52 be 8a 60 14 11 1d }
    f2:RES = { 7c a2 0b 16 }
     f3:CK = { 31 10 d2 fa b9 94 3c e6 58 bc 58 dc 23 df d7 86 }
     f4:IK = { c8 a8 39 1d ae f0 f3 c0 a6 a9 66 9f 1f 18 6e 60 }
     f5:AK = { d9 45 29 ba 0a b8 }
   f5*:AK* = { 3f da 96 ed 1c 79 }
  f5**:AK* = { 8f 02 72 bb 13 bb }

Case #1c: RES_sz= 7, CK_sz=29, IK_sz=17, MAC_sz=23, AK_sz= 9
 f1*:MAC_S = { 71 80 18 24 e5 55 a2 17 9a ee a3 07 c5 7a f4 60
                4c ab 55 a9 7f 5e ab }
  f1:MAC_A = { ee 3d 91 79 17 88 20 04 78 19 58 d6 7a 0f 2a 82
                ef 46 d9 21 da 9c c9 }
    f2:RES = { 62 22 70 07 ee 41 ff }
```

```
     f3:CK = { 65 59 fd 01 ef 18 08 18 ad f4 b5 bf d9 aa 10 a0
               d9 52 41 f3 d3 6e 1b 23 bf e9 d1 b8 00 }
     f4:IK = { 0f b7 36 fa 84 81 84 2d ae d9 92 ee d2 fa 97 ad
               1f }
     f5:AK = { 02 0f cf 3d 7d 13 4d b9 56 }
   f5*:AK* = { d7 91 da 08 07 62 69 61 30 }
  f5**:AK* = { 3a f8 f8 02 d4 f0 93 a8 a5 }

Case #1d: RES_sz= 8, CK_sz=32, IK_sz=32, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { db 14 35 14 1d 60 77 dc }
  f1:MAC_A = { b8 52 be 8a 60 14 11 1d }
    f2:RES = { 07 39 20 04 2f 59 6a ba }
     f3:CK = { b5 7d ce 1e 23 33 fd 82 f9 b5 64 d1 73 c5 75 82
               78 50 51 3e 8a 46 92 17 02 c9 cf aa 3e 03 6c bc }
     f4:IK = { 11 b6 d9 a3 ca d8 95 30 1b be 42 3d 39 15 0b 14
               c7 27 33 66 0e bc 8e 8d 36 a2 6b b1 dd bb 8e ea }
     f5:AK = { d9 45 29 ba 0a b8 }
   f5*:AK* = { 3f da 96 ed 1c 79 }
  f5**:AK* = { 8f 02 72 bb 13 bb }

Case #1e: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=16, AK_sz= 6
 f1*:MAC_S = { 03 4c d6 43 7f a7 91 84 dc 85 40 7a 12 86 6e af }
  f1:MAC_A = { c1 5c 4b da f2 d4 db 2d 87 65 c2 1e 45 ab 9e eb }
    f2:RES = { c3 0d 93 c2 ab 82 a0 ad b1 d0 cf ed e6 8c fc f5
               4a 4a 8f 74 88 8e 48 97 a7 8a d6 c9 ce 54 f3 ee }
     f3:CK = { b5 7d ce 1e 23 33 fd 82 f9 b5 64 d1 73 c5 75 82
               78 50 51 3e 8a 46 92 17 02 c9 cf aa 3e 03 6c bc }
     f4:IK = { 11 b6 d9 a3 ca d8 95 30 1b be 42 3d 39 15 0b 14
               c7 27 33 66 0e bc 8e 8d 36 a2 6b b1 dd bb 8e ea }
     f5:AK = { d9 45 29 ba 0a b8 }
   f5*:AK* = { 3f da 96 ed 1c 79 }
  f5**:AK* = { 35 1e aa 56 07 04 }

=== Milenage-256 TEST #2 ===
Inputs: KEY_sz=16, RAND_sz=16, SQN_sz= 6
        OP = { 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 }
       KEY = { e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef }
      RAND = { 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f }
       SQN = { a0 a1 a2 a3 a4 a5 }
       AMF = { b0 b1 }
        c0 = { 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f }
        c1 = { 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f }
        c2 = { 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f }
        c3 = { 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f }
        c4 = { 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f }
        c5 = { 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f }
        c6 = { 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f }
        c7 = { 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f }
Computation of OPc:
 PRF_K(OP) = { 12 1e c9 d4 bb 81 75 3c 58 bc 57 2a cc fe 78 5e
               ca 57 5b 55 64 bf 27 f8 50 10 73 75 5a 21 37 ea }
         V = { 00 4d 49 4c 45 4e 41 47 45 32 2e 30 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 12 53 80 98 fe cf 34 7b 1d 8e 79 1a cc fe 78 5e
               ca 57 5b 55 64 bf 27 f8 50 10 73 75 5a 21 37 ea }
  PRF_out2 = { 4c 63 06 04 e3 4c 0a 62 1a 78 78 fd 97 81 4a a6
               4d f6 a0 bf b9 9e 6f 3f d7 39 b4 40 83 20 a7 10 }
       OPc = { 8c a2 c4 c7 27 89 cc a5 d2 b1 b2 36 5b 4c 84 69
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }

Case #2a: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=32, AK_sz=12
Intermediates for the function index 0:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { 0e 3f b0 b1 a0 a1 a2 a3 a4 a5 00 00 00 00 00 00
               00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f }
   PRF_in2 = { e7 53 92 b3 fe 91 9d 55 d3 9a 66 a2 14 e1 f3 df
               ce e2 be 46 c2 68 c6 99 5c 12 b5 20 a0 f7 25 d9 }
  PRF_out2 = { ce 73 6d 74 18 dc c4 e6 db 0d bf 9e ae 85 2f 83
               52 df 1a a8 7f 39 10 e9 0d ba 30 81 78 7d 82 60 }
 f1*:MAC_S = { 42 d1 a9 b3 3f 55 08 43 09 bc 0d a8 f5 c9 ab ea
               cf f8 68 c4 12 72 a9 01 02 5a 5e 1a 27 80 fb af }
Intermediates for the function index 1:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
```

```
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { 2e 3f b0 b1 a0 a1 a2 a3 a4 a5 00 00 00 00 00 00
               10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f }
   PRF_in2 = { c7 53 92 b3 fe 91 9d 55 d3 9a 66 a2 14 e1 f3 df
               de f2 ae 56 d2 78 d6 89 4c 02 a5 30 b0 e7 35 c9 }
  PRF_out2 = { 78 bb 62 5c ca 9e 18 6b 78 24 ba 5a 3e fe d9 b0
               e9 7f b3 d1 dd 6e 6b 20 52 63 78 a7 12 58 0b 9a }
  f1:MAC_A = { f4 19 a6 9b ed 17 d4 ce aa 95 08 6c 65 b2 5d d9
               74 58 c1 bd b0 25 d2 c8 5d 83 16 3c 4d a5 72 55 }
Intermediates for the function index 2:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { 4e 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f }
   PRF_in2 = { a7 73 22 02 5e 30 3f f6 77 3f 66 a2 14 e1 f3 df
               ee c2 9e 66 e2 48 e6 b9 7c 32 95 00 80 d7 05 f9 }
  PRF_out2 = { d4 6b 18 fa c3 a4 28 e7 28 4c d2 c1 7f fc cd 87
               76 a1 da 94 15 b8 ba 12 22 99 73 66 c6 5c f4 7e }
    f2:RES = { 58 c9 dc 3d e4 2d e4 42 fa fd 60 f7 24 b0 49 ee
               eb 86 a8 f8 78 f3 03 fa 2d 79 1d fd 99 a1 8d b1 }
Intermediates for the function index 3:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { 6e 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f }
   PRF_in2 = { 87 73 22 02 5e 30 3f f6 77 3f 66 a2 14 e1 f3 df
               fe d2 8e 76 f2 58 f6 a9 6c 22 85 10 90 c7 15 e9 }
  PRF_out2 = { 69 75 f2 55 29 7d 8d 98 19 38 3f 49 2b b7 2f 71
               e6 0e 6a 8a 3a 24 cc 91 77 66 c7 b5 c5 46 5a 1b }
     f3:CK = { e5 d7 36 92 0e f4 41 3d cb 89 8d 7f 70 fb ab 18
               7b 29 18 e6 57 6f 75 79 78 86 a9 2e 9a bb 23 d4 }
Intermediates for the function index 4:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { 8e 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f }
   PRF_in2 = { 67 73 22 02 5e 30 3f f6 77 3f 66 a2 14 e1 f3 df
               8e a2 fe 06 82 28 86 d9 1c 52 f5 60 e0 b7 65 99 }
  PRF_out2 = { 72 d6 5e 12 90 85 b1 e3 b9 49 19 24 4b 37 fa 9d
               bc 17 5a d8 48 55 37 a1 11 7c e8 91 29 d4 cb df }
     f4:IK = { fe 74 9a d5 b7 0c 7d 46 6b f8 ab 12 10 7b 7e f4
               21 30 28 b4 25 1e 8e 49 1e 9c 86 0a 76 29 b2 10 }
Intermediates for the function index 5:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { ae 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f }
   PRF_in2 = { 47 6b 22 02 5e 30 3f f6 77 3f 66 a2 14 e1 f3 df
               9e b2 ee 16 92 38 96 c9 0c 42 e5 70 f0 a7 75 89 }
  PRF_out2 = { 31 83 d6 a8 22 3e 23 2a 9c 5d 48 d7 21 27 85 f8
               6a cc 23 02 28 5e 5a a7 21 92 8c e6 4e c4 f1 51 }
     f5:AK = { bd 21 12 6f 05 b7 ef 8f 4e ec fa e1 }
Intermediates for the function index 6:
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { ce 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f }
   PRF_in2 = { 27 6b 22 02 5e 30 3f f6 77 3f 66 a2 14 e1 f3 df
               ae 82 de 26 a2 08 a6 f9 3c 72 d5 40 c0 97 45 b9 }
  PRF_out2 = { 22 5b 71 40 76 cc 9c 2d 51 44 67 66 90 4d 25 1b
               17 db e0 0c 2e a0 50 5e 2a 5e 54 65 32 78 74 19 }
   f5*:AK* = { ae f9 b5 87 51 45 50 88 83 f5 d5 50 }
Intermediates for the function index 7:
     MAC_S = { 42 d1 a9 b3 3f 55 08 43 09 bc 0d a8 f5 c9 ab ea
               cf f8 68 c4 12 72 a9 01 02 5a 5e 1a 27 80 fb af }
   PRF_in1 = { 0c 23 46 44 a3 0c 4a 22 5a 38 38 bd d7 c1 0a e6
               9d 27 72 6c 6d 4b b9 e8 0f e0 6e 9b 5f fd 79 cf }
```

```
  PRF_out1 = { 65 ce e6 c5 79 b9 f3 53 a5 8e d4 94 4f ad 77 b6
               53 c4 ce 29 ab 26 79 76 5b fb d1 b0 f3 07 52 19 }
        IN = { ee ff 42 d1 a9 b3 3f 55 08 43 09 bc 0d a8 f5 c9
               db 9b bd 8b 1c b1 64 05 d1 78 78 21 22 67 59 ff }
   PRF_in2 = { 07 93 60 d3 f7 83 00 a3 7f 7c 6f 1e 19 49 06 16
               15 78 01 ce da dc a4 9b 85 63 c7 0a 8e 9d 72 29 }
  PRF_out2 = { bf 30 15 e0 59 d2 0b dd 7d 0f 6d a6 f8 e2 2f 85
               a3 d5 3b bb 64 d9 31 94 a2 4d d2 27 58 cd 07 ee }
  f5**:AK* = { 33 92 d1 27 7e 5b c7 78 af be df 90 }

Case #2b: RES_sz= 4, CK_sz=16, IK_sz=16, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 40 6d da 97 ce cd e1 cd }
  f1:MAC_A = { ca a3 71 9f 29 c3 8a 27 }
    f2:RES = { 65 4f 3c c5 }
     f3:CK = { d8 2c 3d d0 3f 2c 3c 8b 14 79 4b b8 d4 4f e9 34 }
     f4:IK = { 96 72 c3 90 f2 54 4d 8d f1 b9 eb 88 e6 24 20 2e }
     f5:AK = { e5 3a 62 f3 2b b0 }
   f5*:AK* = { 39 71 4a 02 f0 03 }
  f5**:AK* = { 0c 75 87 b1 f6 01 }

Case #2c: RES_sz= 7, CK_sz=29, IK_sz=17, MAC_sz=23, AK_sz= 9
 f1*:MAC_S = { 01 23 e0 47 3f cd 2d 9a a2 83 60 f7 3b e1 34 82
               22 09 12 a6 42 a8 58 }
  f1:MAC_A = { 51 13 69 9b 6c 1c e1 8a cb f7 e6 bd 68 f8 3e fa
               4a bf 8b b3 6d 79 44 }
    f2:RES = { 41 70 32 89 94 ff 89 }
     f3:CK = { 0f 36 bf 6c 50 ed 2c 20 87 ca ae 1c b2 05 7d 2c
               19 aa 45 6c 4f 9d b6 ad 0f cb 4b f7 a5 }
     f4:IK = { b0 ef af 06 06 81 82 25 13 c6 9e b6 0c 6b 43 87
               3b }
     f5:AK = { df e0 1a cd 48 1f bb bd 78 }
   f5*:AK* = { 07 e0 11 02 92 d9 ab 61 71 }
  f5**:AK* = { 2b 49 74 31 8d 4a 15 44 f1 }

Case #2d: RES_sz= 8, CK_sz=32, IK_sz=32, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 40 6d da 97 ce cd e1 cd }
  f1:MAC_A = { ca a3 71 9f 29 c3 8a 27 }
    f2:RES = { 7f 35 74 22 f0 bc c5 02 }
     f3:CK = { e5 d7 36 92 0e f4 41 3d cb 89 8d 7f 70 fb ab 18
               7b 29 18 e6 57 6f 75 79 78 86 a9 2e 9a bb 23 d4 }
     f4:IK = { fe 74 9a d5 b7 0c 7d 46 6b f8 ab 12 10 7b 7e f4
               21 30 28 b4 25 1e 8e 49 1e 9c 86 0a 76 29 b2 10 }
     f5:AK = { e5 3a 62 f3 2b b0 }
   f5*:AK* = { 39 71 4a 02 f0 03 }
  f5**:AK* = { 0c 75 87 b1 f6 01 }

Case #2e: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=16, AK_sz= 6
 f1*:MAC_S = { cf 3c 12 20 79 96 19 a5 46 11 01 a4 8c f7 9a 5a }
  f1:MAC_A = { 76 37 5e ff e5 88 99 9b 5c e3 8e 64 55 d6 c2 19 }
    f2:RES = { 58 c9 dc 3d e4 2d e4 42 fa fd 60 f7 24 b0 49 ee
               eb 86 a8 f8 78 f3 03 fa 2d 79 1d fd 99 a1 8d b1 }
     f3:CK = { e5 d7 36 92 0e f4 41 3d cb 89 8d 7f 70 fb ab 18
               7b 29 18 e6 57 6f 75 79 78 86 a9 2e 9a bb 23 d4 }
     f4:IK = { fe 74 9a d5 b7 0c 7d 46 6b f8 ab 12 10 7b 7e f4
               21 30 28 b4 25 1e 8e 49 1e 9c 86 0a 76 29 b2 10 }
     f5:AK = { e5 3a 62 f3 2b b0 }
   f5*:AK* = { 39 71 4a 02 f0 03 }
  f5**:AK* = { 66 f4 7c 45 40 a2 }

=== Milenage-256 TEST #3 ===
Inputs: KEY_sz=32, RAND_sz=22, SQN_sz= 9
        OP = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
       KEY = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
      RAND = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
               ff ff ff ff ff ff }
       SQN = { ff ff ff ff ff ff ff ff ff }
       AMF = { ff ff }
        c0 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c1 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c2 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c3 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c4 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c5 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c6 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
        c7 = { ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
Computation of OPc:
```

```
 PRF_K(OP) = { f3 6c b6 c7 a7 57 2f 19 30 7a 31 e4 ec 4c a4 c8
               2d 27 31 fb 21 f5 9c af 13 3f e8 16 a5 44 24 a5 }
         V = { 01 4d 49 4c 45 4e 41 47 45 32 2e 30 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { f2 21 ff 8b e2 19 6e 5e 75 48 1f d4 ec 4c a4 c8
               2d 27 31 fb 21 f5 9c af 13 3f e8 16 a5 44 24 a5 }
  PRF_out2 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f af b5 ad 82 5f ce 91 6c df 08 }
       OPc = { 41 61 84 6b f0 0b c4 bb e5 25 80 eb f9 09 ad 8d
               cc 74 43 a1 60 90 50 4a 52 7d a0 31 6e 93 20 f7 }

Case #3a: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=32, AK_sz=12
Intermediates for the function index 0:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { 15 9f ff ff ff ff ff ff ff ff ff ff ff 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { 20 4d 40 55 8d b7 cc 9b 5b d6 5d eb df d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { b2 57 70 2a 73 de 09 57 0e c2 be 82 a4 41 6d fc
               7d 06 7f a5 9f a0 42 15 4a 78 51 11 c5 81 b3 e1 }
 f1*:MAC_S = { f3 36 f4 41 83 d5 cd ec eb e7 3e 69 5d 48 c0 71
               b1 72 3c 04 ff 30 12 5f 18 05 f1 20 ab 12 93 16 }
Intermediates for the function index 1:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { 35 9f ff ff ff ff ff ff ff ff ff ff ff 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { 00 4d 40 55 8d b7 cc 9b 5b d6 5d eb df d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { 41 43 cf c7 8e 31 65 2f 30 e6 af 09 36 6d d0 a0
               a7 d4 3e 41 62 ec 49 bb 0c e9 7c 74 2a 10 99 1e }
  f1:MAC_A = { 00 22 4b ac 7e 3a a1 94 d5 c3 2f e2 cf 64 7d 2d
               6b a0 7d e0 02 7c 19 f1 5e 94 dc 45 44 83 b9 e9 }
Intermediates for the function index 2:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { 55 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { 60 cd bf aa 72 48 33 64 a4 29 a2 14 20 d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { 99 f7 8f ad 62 97 a6 39 56 f4 17 bd 7f cc ce db
               37 25 3e 63 78 bf 82 ed 7c b7 ae 41 03 de 45 3a }
    f2:RES = { d8 96 0b c6 92 9c 62 82 b3 d1 97 56 86 c5 63 56
               fb 51 7d c2 18 2f d2 a7 2e ca 0e 70 6d 4d 65 cd }
Intermediates for the function index 3:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { 75 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { 40 cd bf aa 72 48 33 64 a4 29 a2 14 20 d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { a2 33 70 ef ab 36 cb 27 06 33 1a ea 8c 2d 73 3c
               b9 37 fe 85 0f 67 b5 6c 52 d3 06 cb 34 41 f3 38 }
     f3:CK = { e3 52 f4 84 5b 3d 0f 9c e3 16 9a 01 75 24 de b1
               75 43 bd 24 6f f7 e5 26 00 ae a6 fa 5a d2 d3 cf }
Intermediates for the function index 4:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { 95 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { a0 cd bf aa 72 48 33 64 a4 29 a2 14 20 d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { 27 26 21 10 7d 36 0f 87 47 ee ed 7b 20 89 9e 2f
               4e d3 9f e7 d8 28 52 da f6 0b df 10 33 c0 82 6a }
     f4:IK = { 66 47 a5 7b 8d 3d cb 3c a2 cb 6d 90 d9 80 33 a2
               82 a7 dc 46 b8 b8 02 90 a4 76 7f 21 5d 53 a2 9d }
Intermediates for the function index 5:
```

```
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { b5 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { 80 d5 bf aa 72 48 33 64 a4 29 a2 14 20 d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { d8 40 96 e0 db 54 d5 3d a3 46 11 d2 18 bc 4e 09
               6a ee ec 66 f9 c4 22 12 8c 66 c3 24 db 9d 8c 97 }
     f5:AK = { 99 21 12 8b 2b 5f 11 86 46 63 91 39 }
Intermediates for the function index 6:
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { d5 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff }
   PRF_in2 = { e0 d5 bf aa 72 48 33 64 a4 29 a2 14 20 d8 99 1b
               49 f0 6a c6 54 05 bc 67 68 5c 51 b5 f8 01 b1 b4 }
  PRF_out2 = { c7 44 7c ad 53 1f 9f 49 92 47 28 d8 08 1d 13 9b
               eb da 3c 04 78 31 a1 d4 89 82 aa 76 4c 4a 7a 63 }
   f5*:AK* = { 86 25 f8 c6 a3 14 5b f2 77 62 a8 33 }
Intermediates for the function index 7:
     MAC_S = { f3 36 f4 41 83 d5 cd ec eb e7 3e 69 5d 48 c0 71
               b1 72 3c 04 ff 30 12 5f 18 05 f1 20 ab 12 93 16 }
   PRF_in1 = { be 9e 7b 94 0f f4 3b 44 1a da 7f 14 06 f6 52 72
               33 8b bc 5e 9f 6f 50 4a 52 7d a0 31 6e 93 20 f7 }
  PRF_out1 = { 74 b3 3b c1 82 43 f7 df 41 0c 22 ff d9 d1 34 96
               7a 7b d6 98 cb 6a 13 d2 c5 de 0e 7b 69 6d 6e bc }
        IN = { f5 ff f3 36 f4 41 83 d5 cd ec eb e7 3e 69 5d 48
               3f 8e 4e 8d c3 fb 00 cf ed a0 e7 fa 0e df 54 ed }
   PRF_in2 = { c0 2d 4c 9c 86 09 b0 b1 69 c5 49 f3 1e b1 c4 53
               89 81 db b4 68 01 43 57 7a 03 49 b0 09 21 1a a6 }
  PRF_out2 = { 03 a8 31 2b 5c 0b 15 36 b8 fe 2f 33 69 64 63 6d
               3f dd fb 08 1d 89 77 77 fe 8a 32 b0 d8 d1 79 40 }
  f5**:AK* = { 42 c9 b5 40 ac 00 d1 8d 5d db af d8 }

Case #3b: RES_sz= 4, CK_sz=16, IK_sz=16, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 5c e4 56 c2 b7 42 86 80 }
  f1:MAC_A = { 28 43 21 45 13 e3 18 50 }
    f2:RES = { bb 9c 70 81 }
     f3:CK = { b8 5d f6 0b 83 db a9 d8 59 5d 2a e0 a9 7f fc 63 }
     f4:IK = { 67 6a 0e c8 16 5c 2e 51 68 12 77 7e 70 20 39 d7 }
     f5:AK = { ae 02 e3 a4 81 00 }
   f5*:AK* = { fd 57 20 e5 a4 01 }
  f5**:AK* = { 35 95 02 21 0e 01 }

Case #3c: RES_sz= 7, CK_sz=29, IK_sz=17, MAC_sz=23, AK_sz= 9
 f1*:MAC_S = { 38 29 55 45 00 a6 e9 5d 97 e1 b0 10 97 ff a3 e5
               47 9c 85 c6 3a e7 fd }
  f1:MAC_A = { 02 b8 dc 8c 5f fa 59 2c 10 09 e6 cf 60 c8 80 27
               7e 2f 75 17 b5 ec 41 }
    f2:RES = { a3 d2 c9 60 18 29 8b }
     f3:CK = { 9a cb 00 85 79 a0 be 1a 73 e5 f8 01 52 59 2c 89
               3a 24 24 7a 37 e6 98 78 65 0b 6a bc 46 }
     f4:IK = { be e1 c4 30 ef 3d 67 d8 39 f9 e5 f9 c6 be d0 09
               45 }
     f5:AK = { d7 1f bb 16 dc 39 50 40 af }
   f5*:AK* = { 57 6b dc 16 fa 82 cd 71 2a }
  f5**:AK* = { d8 82 3f 89 e3 77 2a 98 32 }

Case #3d: RES_sz= 8, CK_sz=32, IK_sz=32, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 5c e4 56 c2 b7 42 86 80 }
  f1:MAC_A = { 28 43 21 45 13 e3 18 50 }
    f2:RES = { 4a f3 14 9b ae 51 63 5d }
     f3:CK = { e3 52 f4 84 5b 3d 0f 9c e3 16 9a 01 75 24 de b1
               75 43 bd 24 6f f7 e5 26 00 ae a6 fa 5a d2 d3 cf }
     f4:IK = { 66 47 a5 7b 8d 3d cb 3c a2 cb 6d 90 d9 80 33 a2
               82 a7 dc 46 b8 b8 02 90 a4 76 7f 21 5d 53 a2 9d }
     f5:AK = { ae 02 e3 a4 81 00 }
   f5*:AK* = { fd 57 20 e5 a4 01 }
  f5**:AK* = { 35 95 02 21 0e 01 }

Case #3e: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=16, AK_sz= 6
 f1*:MAC_S = { b4 c7 6e 6a b9 14 7b 09 d0 75 67 66 49 c2 b7 39 }
  f1:MAC_A = { a9 5c 18 90 e7 60 d8 54 e7 69 f2 32 36 59 bf 5a }
    f2:RES = { d8 96 0b c6 92 9c 62 82 b3 d1 97 56 86 c5 63 56
```

```
               fb 51 7d c2 18 2f d2 a7 2e ca 0e 70 6d 4d 65 cd }
     f3:CK = { e3 52 f4 84 5b 3d 0f 9c e3 16 9a 01 75 24 de b1
               75 43 bd 24 6f f7 e5 26 00 ae a6 fa 5a d2 d3 cf }
     f4:IK = { 66 47 a5 7b 8d 3d cb 3c a2 cb 6d 90 d9 80 33 a2
               82 a7 dc 46 b8 b8 02 90 a4 76 7f 21 5d 53 a2 9d }
     f5:AK = { ae 02 e3 a4 81 00 }
   f5*:AK* = { fd 57 20 e5 a4 01 }
  f5**:AK* = { d3 29 5e ae f1 e7 }

=== Milenage-256 TEST #4 ===
Inputs: KEY_sz=32, RAND_sz=16, SQN_sz= 6
        OP = { 3d 5f 05 9e 24 d3 75 33 f7 dd 09 a1 74 5a fd c2
               56 22 99 51 c0 dd b4 59 df 19 77 ed cc 9a 63 1a }
       KEY = { af f1 95 1a 2a 51 49 ca f5 9d 9e 5f c5 c5 99 54
               73 53 6b a6 5a 41 f7 44 01 0e 8f c1 fa 11 fe 4d }
      RAND = { 09 0c cc e3 89 04 bd c4 0c 50 9b 23 42 f1 35 22 }
       SQN = { dc 14 98 b4 d7 bd }
       AMF = { 93 d7 }
        c0 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c1 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 }
        c2 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 }
        c3 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 }
        c4 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 }
        c5 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 }
        c6 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 }
        c7 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 }
Computation of OPc:
 PRF_K(OP) = { dd de 4f 8b 85 d2 2a 14 e6 fc d3 61 e9 26 35 95
               32 c8 d5 3e 43 44 83 5f 27 f1 1c 86 35 5a 94 3b }
         V = { 01 4d 49 4c 45 4e 41 47 45 32 2e 30 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { dc 93 06 c7 c0 9c 6b 53 a3 ce fd 51 e9 26 35 95
               32 c8 d5 3e 43 44 83 5f 27 f1 1c 86 35 5a 94 3b }
  PRF_out2 = { 88 fc 15 c4 f1 70 6d bf 32 41 bd c7 e4 fe 22 eb
               d5 1b b8 6c d6 6f f8 2a 2a 35 12 a2 7c ac 1f ec }
       OPc = { b5 a3 10 5a d5 a3 18 8c c5 9c b4 66 90 a4 df 29
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }

Case #4a: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=32, AK_sz=12
Intermediates for the function index 0:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { 0f 3f 93 d7 dc 14 98 b4 d7 bd 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 37 f0 95 3a 6b 98 61 55 6c bd 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 97 }
  PRF_out2 = { bb 22 a5 55 6c 5e 33 0b ce 1f 43 8a 69 38 12 0c
               aa 90 39 28 40 68 f4 46 3a 15 4b 79 ed 26 a6 21 }
 f1*:MAC_S = { 0e 81 b5 0f b9 fd 2b 87 0b 83 f7 ec f9 9c cd 25
               29 a9 18 15 56 da b8 35 cf 39 2e 36 5d 10 da d7 }
Intermediates for the function index 1:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { 2f 3f 93 d7 dc 14 98 b4 d7 bd 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 }
   PRF_in2 = { 17 f0 95 3a 6b 98 61 55 6c bd 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 96 }
  PRF_out2 = { 5f 30 06 86 5e f9 c8 f6 5f f4 b1 74 df e6 a6 72
               1b 24 10 ae 0a 48 3f 46 aa e6 3a 6f 20 0e 3d fb }
  f1:MAC_A = { ea 93 16 dc 8b 5a d0 7a 9a 68 05 12 4f 42 79 5b
               98 1d 31 93 1c fa 73 35 5f ca 5f 20 90 38 41 0d }
Intermediates for the function index 2:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { 4f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 }
   PRF_in2 = { 77 d0 06 ed b7 8c f9 e1 bb 00 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 95 }
  PRF_out2 = { c6 fa a4 51 65 f0 06 f8 de d2 80 ea 1f d8 6c 05
               2a 5e 9b ee 86 95 e4 95 84 78 71 85 c1 b8 8d bb }
    f2:RES = { 73 59 b4 0b b0 53 1e 74 1b 4e 34 8c 8f 7c b3 2c
               a9 67 ba d3 90 27 a8 e6 71 54 14 ca 71 8e f1 4d }
```

```
Intermediates for the function index 3:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { 6f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 }
   PRF_in2 = { 57 d0 06 ed b7 8c f9 e1 bb 00 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 93 }
  PRF_out2 = { 02 68 8b 0f 04 d8 cb 9d 73 d1 10 77 66 f5 e1 8c
               72 c6 d4 44 4d 4f dd d6 21 4f 94 c8 b4 f4 1d 8e }
     f3:CK = { b7 cb 9b 55 d1 7b d3 11 b6 4d a4 11 f6 51 3e a5
               f1 ff f5 79 5b fd 91 a5 d4 63 f1 87 04 c2 61 78 }
Intermediates for the function index 4:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { 8f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 }
   PRF_in2 = { b7 d0 06 ed b7 8c f9 e1 bb 00 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 9f }
  PRF_out2 = { ca aa 4b d5 0d 54 fd 8d 3a ce 6c ff dd 8d 91 ba
               eb c9 0f 10 a6 64 56 a8 e0 82 0c 17 b9 ca 88 74 }
     f4:IK = { 7f 09 5b 8f d8 f7 e5 01 ff 52 d8 99 4d 29 4e 93
               68 f0 2e 2d b0 d6 1a db 15 ae 69 58 09 fc f4 82 }
Intermediates for the function index 5:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { af 07 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 }
   PRF_in2 = { 97 c8 06 ed b7 8c f9 e1 bb 00 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 87 }
  PRF_out2 = { c6 1b e1 e6 44 ab 10 b8 1a 34 12 e1 5f 7a d9 d5
               fa 8b cb 2e 06 11 2d 74 53 79 c3 5b 96 5b 26 45 }
     f5:AK = { 73 b8 f1 bc 91 08 08 34 df a8 a6 87 }
Intermediates for the function index 6:
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { cf 07 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 }
   PRF_in2 = { f7 c8 06 ed b7 8c f9 e1 bb 00 38 1d 4d 80 a6 8e
               40 71 04 f0 ed cb a9 13 8c 46 a8 79 4a d0 44 b7 }
  PRF_out2 = { d8 23 3e 27 ae 2d da 99 9e 95 f2 39 7c de 1c b8
               5b f2 29 d8 a4 38 82 93 d3 d8 e5 d3 cd a0 69 67 }
   f5*:AK* = { 6d 80 2e 7d 7b 8e c2 15 5b 09 46 5f }
Intermediates for the function index 7:
     MAC_S = { 0e 81 b5 0f b9 fd 2b 87 0b 83 f7 ec f9 9c cd 25
               29 a9 18 15 56 da b8 35 cf 39 2e 36 5d 10 da d7 }
   PRF_in1 = { bc af dc b9 5c a7 a5 48 c9 cc 2f 45 d2 55 ea 0b
               83 39 21 3d 16 b2 4c 73 f5 2c 65 4f b0 36 7c f6 }
  PRF_out1 = { 8d 6c 16 b7 62 2f e1 6d 7e 9c 8c 7b dd 24 79 a7
               c3 48 25 cd fb 79 e5 60 79 6a cd 36 fa e6 38 61 }
        IN = { ef ff 0e 81 b5 0f b9 fd 2b 87 0b 83 f7 ec f9 9c
               cd 25 29 a9 18 15 56 da b8 35 cf 39 2e 36 5d 50 }
   PRF_in2 = { d7 30 08 6c 02 83 40 1c 90 87 33 9e ba 6c 5f 12
               8d 54 2d 59 f5 de ff c9 34 73 67 40 64 e6 19 c7 }
  PRF_out2 = { 61 20 fb ee 5f 9d 6c a4 a7 5f f2 69 80 9b bf df
               fb dd ec ea 4e 8f 91 fc 49 25 03 ad 03 7c 02 f7 }
  f5**:AK* = { d4 83 eb b4 8a 3e 74 28 62 c3 46 0f }

Case #4b: RES_sz= 4, CK_sz=16, IK_sz=16, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { a2 f0 62 a6 ee 18 1e 24 }
  f1:MAC_A = { 9c 79 c4 a4 5b 77 11 87 }
    f2:RES = { 01 32 ac fa }
     f3:CK = { f5 24 a0 13 a3 31 e0 7d a9 da ff fc 32 2e 34 57 }
     f4:IK = { 72 6c 31 30 ea ac a2 95 ce f8 fc b0 a7 a9 b9 5a }
     f5:AK = { fc cd 9c 20 4f 14 }
   f5*:AK* = { ac 87 e0 34 28 b2 }
  f5**:AK* = { 36 cb ae 3c 64 6b }

Case #4c: RES_sz= 7, CK_sz=29, IK_sz=17, MAC_sz=23, AK_sz= 9
 f1*:MAC_S = { 5f b4 4b d7 51 97 ec bd 5e c1 48 7e f1 28 19 7b
               de 31 a2 e1 3b 4e 84 }
```

```
  f1:MAC_A = { 6f f1 58 20 ba 12 f2 82 d2 29 2a 68 a2 ad 70 da
               62 43 ba c3 90 33 55 }
    f2:RES = { 19 a1 aa 4b 5d e0 81 }
     f3:CK = { d0 24 88 9f 59 46 4c f5 15 e8 a4 ee ad bf 68 05
               f2 91 e9 ad 3a 06 dc 36 fe b2 6e c2 78 }
     f4:IK = { e6 09 df 64 4b 23 45 2b 65 0c c9 b1 e7 5a ef ae
               bc }
     f5:AK = { 89 63 c0 2d 8b 8d 95 00 cc }
   f5*:AK* = { 32 44 84 05 9a bf 60 96 88 }
  f5**:AK* = { 26 01 31 3c d9 63 a6 0a 1a }

Case #4d: RES_sz= 8, CK_sz=32, IK_sz=32, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { a2 f0 62 a6 ee 18 1e 24 }
  f1:MAC_A = { 9c 79 c4 a4 5b 77 11 87 }
    f2:RES = { ae dd 7f f3 5e 13 75 f6 }
     f3:CK = { b7 cb 9b 55 d1 7b d3 11 b6 4d a4 11 f6 51 3e a5
               f1 ff f5 79 5b fd 91 a5 d4 63 f1 87 04 c2 61 78 }
     f4:IK = { 7f 09 5b 8f d8 f7 e5 01 ff 52 d8 99 4d 29 4e 93
               68 f0 2e 2d b0 d6 1a db 15 ae 69 58 09 fc f4 82 }
     f5:AK = { fc cd 9c 20 4f 14 }
   f5*:AK* = { ac 87 e0 34 28 b2 }
  f5**:AK* = { 36 cb ae 3c 64 6b }

Case #4e: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=16, AK_sz= 6
 f1*:MAC_S = { 12 3a d4 d4 c9 4d 02 c9 77 17 5d ae f3 63 19 c6 }
  f1:MAC_A = { a0 e4 39 b6 fc 06 49 05 f0 2d 5c 4b 17 35 01 c6 }
    f2:RES = { 73 59 b4 0b b0 53 1e 74 1b 4e 34 8c 8f 7c b3 2c
               a9 67 ba d3 90 27 a8 e6 71 54 14 ca 71 8e f1 4d }
     f3:CK = { b7 cb 9b 55 d1 7b d3 11 b6 4d a4 11 f6 51 3e a5
               f1 ff f5 79 5b fd 91 a5 d4 63 f1 87 04 c2 61 78 }
     f4:IK = { 7f 09 5b 8f d8 f7 e5 01 ff 52 d8 99 4d 29 4e 93
               68 f0 2e 2d b0 d6 1a db 15 ae 69 58 09 fc f4 82 }
     f5:AK = { fc cd 9c 20 4f 14 }
   f5*:AK* = { ac 87 e0 34 28 b2 }
  f5**:AK* = { ab e6 2d 6c be 57 }

=== Milenage-256 TEST #5 ===
Inputs: KEY_sz=32, RAND_sz=32, SQN_sz= 6
        OP = { 28 5d 52 6d 1d 37 d5 e8 da b8 6f 04 03 01 89 19
               e5 3a 7d a1 5d e1 9f 34 f5 61 7e b7 51 e7 67 d0 }
       KEY = { ca 71 2b 9c 58 28 22 a5 b5 ef 70 0e bf 3c 87 11
               1f 26 ee ac 28 c0 40 96 01 a6 77 78 07 d5 6a 5a }
      RAND = { 6d c0 95 5b 25 a0 19 a6 04 0b 03 a1 21 c8 d1 68
               7e e2 33 79 ea fb 0e b1 25 e3 c6 77 46 69 bd 37 }
       SQN = { 44 b9 3d 47 26 a3 }
       AMF = { 5a 71 }
        c0 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c1 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c2 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c3 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c4 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c5 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c6 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
        c7 = { 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
Computation of OPc:
 PRF_K(OP) = { 04 ff 95 d3 41 15 9e 45 ca 40 5a 0c 6f 85 f5 3e
               f1 5f cb 9c 2e c5 11 8f fa bf 6e e8 1d 85 34 f0 }
         V = { 01 4d 49 4c 45 4e 41 47 45 32 2e 30 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 05 b2 dc 9f 04 5b df 02 8f 72 74 3c 6f 85 f5 3e
               f1 5f cb 9c 2e c5 11 8f fa bf 6e e8 1d 85 34 f0 }
  PRF_out2 = { 7f 40 bf d7 21 99 f6 29 bd 91 3b 3a 5c 84 b1 f2
               04 ec 5f ff 16 1e 44 27 dd 43 6b 4b 83 a4 36 13 }
       OPc = { 57 1d ed ba 3c ae 23 c1 67 29 54 3e 5f 85 38 eb
               e1 d6 22 5e 4b ff db 13 28 22 15 fc d2 43 51 c3 }

Case #5a: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=32, AK_sz=12
Intermediates for the function index 0:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { 1f 3f 5a 71 44 b9 3d 47 26 a3 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 25 51 d9 16 6d 91 67 26 0e 90 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { aa 00 32 bf a0 09 eb 10 78 70 df be 5f 9d f0 06
               b2 63 82 3c 54 66 a8 6b 6e 2f de f5 a8 b3 99 ef }
```

```
 f1*:MAC_S = { fd 1d df 05 9c a7 c8 d1 1f 59 8b 80 00 18 c8 ed
               53 b5 a0 62 1f 99 73 78 46 0d cb 09 7a f0 c8 2c }
Intermediates for the function index 1:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { 3f 3f 5a 71 44 b9 3d 47 26 a3 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 05 51 d9 16 6d 91 67 26 0e 90 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { 35 6d b7 90 40 23 b8 cd 40 9c b0 13 a6 0b 98 ed
               4e d2 36 96 4b 2d 42 a3 14 d9 46 60 04 2b bd e1 }
  f1:MAC_A = { 62 70 5a 2a 7c 8d 9b 0c 27 b5 e4 2d f9 8e a0 06
               af 04 14 c8 00 d2 99 b0 3c fb 53 9c d6 68 ec 22 }
Intermediates for the function index 2:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { 5f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 65 71 83 67 29 28 5a 61 28 33 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { 2f 98 14 7d 31 9d 53 f5 4c 40 c3 85 97 1c 7f ec
               af 5f 27 f9 84 bd 8e ce 2b 4b 6e 73 e3 9a 58 52 }
    f2:RES = { 78 85 f9 c7 0d 33 70 34 2b 69 97 bb c8 99 47 07
               4e 89 05 a7 cf 42 55 dd 03 69 7b 8f 31 d9 09 91 }
Intermediates for the function index 3:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { 7f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 45 71 83 67 29 28 5a 61 28 33 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { c4 49 ca 7b 94 56 9f 4c 3d 55 ad ea 13 7c 97 a7
               56 fd 46 c2 59 25 b0 9a 06 6a dd 65 30 d0 0a b7 }
     f3:CK = { 93 54 27 c1 a8 f8 bc 8d 5a 7c f9 d4 4c f9 af 4c
               b7 2b 64 9c 12 da 6b 89 2e 48 c8 99 e2 93 5b 74 }
Intermediates for the function index 4:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { 9f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { a5 71 83 67 29 28 5a 61 28 33 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { 7d 5a 6b 46 bc 07 24 42 6a 55 8e 3f d7 04 4d dc
               3b 35 86 d4 d0 2f 77 9d 43 87 dd 63 c1 51 0c 96 }
     f4:IK = { 2a 47 86 fc 80 a9 07 83 0d 7c da 01 88 81 75 37
               da e3 a4 8a 9b d0 ac 8e 6b a5 c8 9f 13 12 5d 55 }
Intermediates for the function index 5:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { bf 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { 85 69 83 67 29 28 5a 61 28 33 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { db c2 2d 09 fa 93 ba 16 9b 39 2e d4 9c e3 df f9
               8b 9c a1 d4 3d ca 7f 70 a2 dd 32 22 90 b4 2e 63 }
     f5:AK = { 8c df c0 b3 c6 3d 99 d7 fc 10 7a ea }
Intermediates for the function index 6:
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { df 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00
               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 }
   PRF_in2 = { e5 69 83 67 29 28 5a 61 28 33 54 41 b1 e3 db 37
               df ec 61 9f 8f 1c 1d 11 f2 eb ee fd 78 38 7e 5d }
  PRF_out2 = { 0d ae a0 5a f1 fe bb b9 ec a5 19 ad 46 c9 b0 21
               76 0e 5b 2a 2e 97 6d 94 90 0e 3d d6 fe 4b b3 48 }
    f5*:AK* = { 5a b3 4d e0 cd 50 98 78 8b 8c 4d 93 }
```

```
Intermediates for the function index 7:
     MAC_S = { fd 1d df 05 9c a7 c8 d1 1f 59 8b 80 00 18 c8 ed
               53 b5 a0 62 1f 99 73 78 46 0d cb 09 7a f0 c8 2c }
   PRF_in1 = { 3a dd 78 e1 19 0e 3a 67 63 22 57 9f 7e 4d e9 83
               9f 34 11 27 a1 04 d5 a2 0d c1 d3 8b 94 2a ec f4 }
  PRF_out1 = { 6d 73 6e dd 15 86 79 a0 4f 1a 00 7f ee 66 e3 dc
               3e 3a 43 c1 c4 e3 c6 02 da c9 fb 01 aa 7b 2f 9e }
        IN = { ff ff fd 1d df 05 9c a7 c8 d1 1f 59 8b 80 00 18
               c8 ed 53 b5 a0 62 1f 99 73 78 46 0d cb 09 7a f0 }
   PRF_in2 = { c5 91 7e 7a f6 2d c6 c6 e0 e2 4b 18 3a 63 db 2f
               17 01 32 2a 2f 7e 02 88 81 93 a8 f0 b3 31 04 ad }
  PRF_out2 = { c8 07 d2 8f f2 d8 b7 6f a2 58 f3 a9 22 73 8c 84
               a0 61 91 10 a3 bb 84 d1 51 af a2 41 81 63 1e a4 }
  f5**:AK* = { 9f 1a 3f 35 ce 76 94 ae c5 71 a7 97 }

Case #5b: RES_sz= 4, CK_sz=16, IK_sz=16, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 60 5c c4 ad 83 e8 d3 2d }
  f1:MAC_A = { eb e7 6a 45 a4 bd d8 03 }
    f2:RES = { 27 ed 2e 48 }
     f3:CK = { 8c f5 37 54 dd 1d 08 ba b0 21 7b a0 ee 92 3f 34 }
     f4:IK = { 1f 20 06 e2 43 36 38 91 a8 e8 1e ca 72 40 df a7 }
     f5:AK = { 6d 8d 89 4d e0 91 }
   f5*:AK* = { fb 12 22 f9 3b 35 }
  f5**:AK* = { 3a c9 49 db 1e 5f }

Case #5c: RES_sz= 7, CK_sz=29, IK_sz=17, MAC_sz=23, AK_sz= 9
 f1*:MAC_S = { e8 a6 06 f8 be 46 82 a4 d1 92 73 98 cf 64 ca 17
               e0 f3 2c 97 7d 9e 4f }
  f1:MAC_A = { 48 3f d5 6a 49 62 86 49 65 3c 97 f5 19 60 56 c6
               00 1d ea 1b ee d1 ef }
    f2:RES = { 2e 0a 05 66 32 d5 0f }
     f3:CK = { f3 eb e6 d9 3b b3 6c 12 c7 59 23 69 55 1f b2 6b
               74 d2 5f 51 60 b6 43 d8 66 f8 8c 31 41 }
     f4:IK = { ab 47 c4 77 51 fe 05 d9 8a a4 15 67 ee 96 8b 1c
               07 }
     f5:AK = { 15 77 c0 8c bd 1b 32 22 66 }
   f5*:AK* = { c7 26 94 67 cb 55 72 bc 6b }
  f5**:AK* = { c7 80 83 57 ec b8 ae c2 58 }

Case #5d: RES_sz= 8, CK_sz=32, IK_sz=32, MAC_sz= 8, AK_sz= 6
 f1*:MAC_S = { 60 5c c4 ad 83 e8 d3 2d }
  f1:MAC_A = { eb e7 6a 45 a4 bd d8 03 }
    f2:RES = { 99 aa 0e b4 a2 af 44 15 }
     f3:CK = { 93 54 27 c1 a8 f8 bc 8d 5a 7c f9 d4 4c f9 af 4c
               b7 2b 64 9c 12 da 6b 89 2e 48 c8 99 e2 93 5b 74 }
     f4:IK = { 2a 47 86 fc 80 a9 07 83 0d 7c da 01 88 81 75 37
               da e3 a4 8a 9b d0 ac 8e 6b a5 c8 9f 13 12 5d 55 }
     f5:AK = { 6d 8d 89 4d e0 91 }
   f5*:AK* = { fb 12 22 f9 3b 35 }
  f5**:AK* = { 3a c9 49 db 1e 5f }

Case #5e: RES_sz=32, CK_sz=32, IK_sz=32, MAC_sz=16, AK_sz= 6
 f1*:MAC_S = { b9 45 c4 c2 4e dc c8 10 80 f3 76 2b 26 c7 36 88 }
  f1:MAC_A = { 9c b9 4d f0 1a 61 bd 57 08 8c 48 0e c7 58 c3 70 }
    f2:RES = { 78 85 f9 c7 0d 33 70 34 2b 69 97 bb c8 99 47 07
               4e 89 05 a7 cf 42 55 dd 03 69 7b 8f 31 d9 09 91 }
     f3:CK = { 93 54 27 c1 a8 f8 bc 8d 5a 7c f9 d4 4c f9 af 4c
               b7 2b 64 9c 12 da 6b 89 2e 48 c8 99 e2 93 5b 74 }
     f4:IK = { 2a 47 86 fc 80 a9 07 83 0d 7c da 01 88 81 75 37
               da e3 a4 8a 9b d0 ac 8e 6b a5 c8 9f 13 12 5d 55 }
     f5:AK = { 6d 8d 89 4d e0 91 }
   f5*:AK* = { fb 12 22 f9 3b 35 }
  f5**:AK* = { 0f 15 4a 48 b3 10 }
```

Annex A (informative): Reference implementation (C/C++)

In this clause an informal C/C++ implementation example of Milenage-256 is provided.

A.1 General

Shared components (`milenage256_common.h`)

```
#ifndef _SAGE256_MILENAGE256_COMMON_
#define _SAGE256_MILENAGE256_COMMON_

// ----------------------------------------------------------------------------
// Shared components
// ----------------------------------------------------------------------------
#include <stdint.h>
#include <stdlib.h>
#include <memory.h>

typedef uint8_t  u8;
typedef uint32_t u32;
typedef uint64_t u64;

void xor128(void * dst, const void * in1, const void * in2)
{
    ((u64*)(dst))[0] = ((u64*)(in1))[0] ^ ((u64*)(in2))[0];
    ((u64*)(dst))[1] = ((u64*)(in1))[1] ^ ((u64*)(in2))[1];
}

static const u8 aes_SBox[256] =
{
    0x63,0x7C,0x77,0x7B,0xF2,0x6B,0x6F,0xC5,0x30,0x01,0x67,0x2B,0xFE,0xD7,0xAB,0x76,
    0xCA,0x82,0xC9,0x7D,0xFA,0x59,0x47,0xF0,0xAD,0xD4,0xA2,0xAF,0x9C,0xA4,0x72,0xC0,
    0xB7,0xFD,0x93,0x26,0x36,0x3F,0xF7,0xCC,0x34,0xA5,0xE5,0xF1,0x71,0xD8,0x31,0x15,
    0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x12,0x80,0xE2,0xEB,0x27,0xB2,0x75,
    0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3B,0xD6,0xB3,0x29,0xE3,0x2F,0x84,
    0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xCB,0xBE,0x39,0x4A,0x4C,0x58,0xCF,
    0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF9,0x02,0x7F,0x50,0x3C,0x9F,0xA8,
    0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB6,0xDA,0x21,0x10,0xFF,0xF3,0xD2,
    0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA7,0x7E,0x3D,0x64,0x5D,0x19,0x73,
    0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xEE,0xB8,0x14,0xDE,0x5E,0x0B,0xDB,
    0xE0,0x32,0x3A,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79,
    0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,0xA9,0x6C,0x56,0xF4,0xEA,0x65,0x7A,0xAE,0x08,
    0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xDD,0x74,0x1F,0x4B,0xBD,0x8B,0x8A,
    0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x35,0x57,0xB9,0x86,0xC1,0x1D,0x9E,
    0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1E,0x87,0xE9,0xCE,0x55,0x28,0xDF,
    0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x99,0x2D,0x0F,0xB0,0x54,0xBB,0x16
};

void aes_SubBytes(u8 * state)
{
    for (int i = 0; i < 16; i++)
        state[i] = aes_SBox[state[i]];
}

void aes_ShiftRows(u8 * state)
{
    u8 tmp[16];
    for (int r = 0; r < 4; r++)
        for (int c = 0; c < 4; c++)
            tmp[r + 4 * c] = state[r + 4 * ((c + r) & 3)];
    memcpy(state, tmp, 16);
}

void aes_MixColumns(u8 * state)
{
    for (int c = 0; c < 4; c++)
    {
        u8 s = 0, u[4];
        for (int r = 0; r < 4; r++)
        {
```

```
            s ^= u[r] = state[r + 4 * c];
            u[r] = (u[r] << 1) ^ ((-(u[r] >> 7)) & 0x1b);
        }

        for (int r = 0; r < 4; r++)
            state[r + 4 * c] ^= s ^ u[r] ^ u[(r + 1) & 3];
    }
}

void aes_EncRound(u8 * state, const u8 * RoundKey)
{
    aes_ShiftRows(state);
    aes_SubBytes(state);
    aes_MixColumns(state);
    xor128(state, state, RoundKey);
}

void aes_EncLast(u8 * state, const u8 * RoundKey)
{
    aes_ShiftRows(state);
    aes_SubBytes(state);
    xor128(state, state, RoundKey);
}

void aes_KeyExpand256(u32 * W, int i)
{
    static const u8 RC[] = { 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40,
                             0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d };
    u8 T[5];
    *(u32*)T = W[i - 1];

    int mod = i % 8;
    if (mod == 0)
    {
        T[4] = T[0];
        *(u32*)T = *(u32*)(T + 1);
    }

    if (mod == 0 || mod == 4)
        for (int k = 0; k < 4; k++)
            T[k] = aes_SBox[T[k]];

    if (mod == 0) T[0] ^= RC[i / 8];
    W[i] = W[i - 8] ^ *(u32*)(T);
}

#endif /* _SAGE256_MILENAGE256_COMMON_ */
```

PRF kernel for Milenage-256 (`milenage256_r.h`)

```
#ifndef _SAGE256_MILENAGE256_R_
#define _SAGE256_MILENAGE256_R_
#include "milenage256_common.h"

// ------------------------------------------------------------------------------
// Black-box for Milenage-R: PRP, base on Rijndael-256-256
// ------------------------------------------------------------------------------
struct Rijndael256_256
{
    u32 W[120]; // round keys

    // Key schedule is the same as in AES-128-256, but 2x times longer
    void key_schedule(u8 * key /* [32] */)
    {
        memcpy(W, key, 32);

        for (int i = 8; i < 120; i++)
            aes_KeyExpand256(W, i);
    }

    void permute(u8 * state /* [32] */)
    {
#if 1   /* Method 1 -- in case the permutation (vpermb) can be done in a 256-bit register */
        static const u8 CPi[32] = {
            0x00, 0x11, 0x16, 0x17, 0x04, 0x05, 0x1a, 0x1b,
            0x08, 0x09, 0x0e, 0x1f, 0x0c, 0x0d, 0x12, 0x13,
            0x10, 0x01, 0x06, 0x07, 0x14, 0x15, 0x0a, 0x0b,
            0x18, 0x19, 0x1e, 0x0f, 0x1c, 0x1d, 0x02, 0x03
```

```
        };

        u8 tmp[32];

        for (int i = 0; i < 32; i++)
            tmp[i] = state[CPi[i]];

        memcpy(state, tmp, 32);

#else   /* Method 2 -- in case only 128-bit registers are available (3 SIMD instructions).
           It is also more suitable for HW implementations in pipelining architecture when
           only one 128-bit block AES is utilised */
        u8 lo[16], hi[16];

        // let us have the lower and higher 16-byte halves of the 32-byte state
        memcpy(lo, state, 16);
        memcpy(hi, state + 16, 16);

        // swap bytes at certain fixed indexes (vpblendvb)
        for (int i = 0; i < 16; i++)
            if ((0x8cce >> i) & 1)
            {
                u8 tmp = lo[i];
                lo[i] = hi[i];
                hi[i] = tmp;
            }

        // permute lo and hi individually (vpshufb) and combine into 32-byte resulting state
        static const u8 CPi[16] = { 0, 1, 6, 7, 4, 5, 10, 11, 8, 9, 14, 15, 12, 13, 2, 3 };
        for (int i = 0; i < 16; i++)
        {
            state[i] = lo[CPi[i]];
            state[i + 16] = hi[CPi[i]];
        }
#endif
    }

    void encrypt(u8 * out /* [32] */, u8 * in /* [32] */)
    {
        xor128(out, in, W + 0);
        xor128(out + 16, in + 16, W + 4);

        // Rijndael-256-256 can reuse the standard AES-256
        for (int i = 1; i < 14; i++)
        {
            // The only additional step is the 32-byte permutation between the rounds
            permute(out);

            // call two AES encryption rounds in parallel on the 2x16 bytes state
            // these calls may be done sequentially in case of HW implementation
            aes_EncRound(out, (u8*)(W + i * 8));
            aes_EncRound(out + 16, (u8*)(W + i * 8 + 4));
        }

        permute(out);
        aes_EncLast(out, (u8*)(W + 112));
        aes_EncLast(out + 16, (u8*)(W + 116));
    }

};

void Milenage256_PRP_Rijndael(u8 out[32], u8 in[32], u8 key[32])
{
    Rijndael256_256 R;
    R.key_schedule(key);
    R.encrypt(out, in);
}

#endif /* _SAGE256_MILENAGE256_R_ */
```

Milenage-256 (`milenage256.h`)

```
#ifndef _SAGE256_MILENAGE256_
#define _SAGE256_MILENAGE256_
#include "milenage256_common.h"

// ------------------------------------------------------------------------
// MILENAGE-256 is based on an external "black-box", PRP or PRF
```

```
// An operator needs to decide which external "black-box" plugin is to be used
// ------------------------------------------------------------------------------
typedef void (Milenage256_ExternalPRF256)(u8 out[32], u8 in[32], u8 key[32]);

// ------------------------------------------------------------------------------
// MILENAGE-256 Instance Configuration
// if dynamic   => can be set/modified at run-time
// if constants => fixed instance of the algorithm
// ------------------------------------------------------------------------------
u8 OP[32];             /* Operator's Configuration             */
u8 c[8][16] = { {0} }; /* may be all zeroes, if Operator wants */

u8 KEY_sz  = 32;    /* = 16 or 32 bytes */
u8 RES_sz  = 8;     /* = 1-32 bytes */
u8 CK_sz   = 32;    /* = 1-32 bytes */
u8 IK_sz   = 32;    /* = 1-32 bytes */
u8 MAC_sz  = 16;    /* = 1-32 bytes */
u8 RAND_sz = 16;    /* = 2-32 bytes, even value */
u8 SQN_sz  = 6;     /* = 5-12 bytes */
u8 AK_sz   = 6;     /* = 5-12 bytes */

const char * ALGONAME = "MILENAGE2.0";

u8 OPc[32];

// ------------------------------------------------------------------------------
// Universal function used by MILENAGE-256 API
// ------------------------------------------------------------------------------
void Milenage256_Main(
    u8 fn_idx,          /* in, u8           */
    u8 IN1,             /* in, u8           */
    u8 *key,            /* in, u8[16/32]    */
    u8 *rand,           /* in, u8[RAND_sz]  */
    u8 *amf,            /* in, u8[2]        */
    u8 *sqn,            /* in, u8[SQN_sz]   */
    u8 *mac_s,          /* in, u8[MAC_sz]   */
    u8 *out,            /* out, u8[out_sz]  */
    u8 out_sz
)
{
    u8 K[32], state[32], IN[32] = { 0 };

    // Prepare full 256-bit key
    memcpy(K, key, KEY_sz);
    memset(K + KEY_sz, 0, 32 - KEY_sz);

    // Prepare (OPc + RAND)
    memcpy(state, OPc, 32);
    for (int i = 0; i < RAND_sz; i++)
        state[i] ^= rand[i];

    // First invocation
    Milenage256_PRP_Rijndael(state, state, K);

    // add IN0, IN1, AMF, SQN, MAC-S, ci
    IN[0] ^= (fn_idx << 5) | (RAND_sz - 2) | (KEY_sz >> 5);
    IN[1] ^= IN1;

    if (amf)
        for (int i = 0; i < 2; i++)
            IN[i + 2] ^= amf[i];

    if (sqn)
        for (int i = 0; i < SQN_sz; i++)
            IN[i + 4] ^= sqn[i];

    if (mac_s)
        for (int i = 0; i < (MAC_sz > 30 ? 30 : MAC_sz); i++)
            IN[i + 2] ^= mac_s[i];

    for (int i = 0; i < 16; i++)
        IN[i + 16] ^= c[fn_idx][i];

    // add OPc and IN
    for (int i = 0; i < 32; i++)
        state[i] ^= OPc[i] ^ IN[i];

    // Second invocation
```

```
    Milenage256_PRP_Rijndael(state, state, K);

    // add OPc
    for (int i = 0; i < 32; i++)
        state[i] ^= OPc[i];

    // Output the result
    memcpy(out, state, out_sz);
}

// ------------------------------------------------------------------------------
// MILENAGE-256 API Definitions
// ------------------------------------------------------------------------------
void Milenage256_ComputeTOPC(
    u8 *key             /* in,  u8[16/32]       */
)
{
    u8 K[32], V[32] = { 0 };

    // Prepare the 256-bit encryption key
    memcpy(K, key, KEY_sz);
    memset(K + KEY_sz, 0, 32 - KEY_sz);

    // First invocation
    Milenage256_PRP_Rijndael(OPc, OP, K);

    // add INSc, ALGONAME, and do the second invocation
    V[0] ^= KEY_sz >> 5;

    for (int i = 0; ALGONAME[i] != 0; i++)
        V[i + 1] ^= ALGONAME[i];

    for (int i = 0; i < 32; i++)
        OPc[i] ^= V[i];

    // Second invocation
    Milenage256_PRP_Rijndael(OPc, OPc, K);

    // Add OP as the final step
    for (int i = 0; i < 32; i++)
        OPc[i] ^= OP[i];
}

void Milenage256_f1s(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *sqn,            /* in,  u8[SQN_sz]      */
    u8 *amf,            /* in,  u8[2]           */
    u8 *mac_s           /* out, u8[MAC_sz]      */
)
{
    Milenage256_Main(0, ((SQN_sz - 5) << 5) | (MAC_sz - 1),
        key, rand, amf, sqn, NULL, mac_s, MAC_sz);
}

void Milenage256_f1(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *sqn,            /* in,  u8[SQN_sz]      */
    u8 *amf,            /* in,  u8[2]           */
    u8 *mac_a           /* out, u8[MAC_sz]      */
)
{
    Milenage256_Main(1, ((SQN_sz - 5) << 5) | (MAC_sz - 1),
        key, rand, amf, sqn, NULL, mac_a, MAC_sz);
}

void Milenage256_f2(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *res             /* out, u8[RES_sz]      */
)
{
    Milenage256_Main(2, RES_sz - 1, key, rand, NULL, NULL, NULL, res, RES_sz);
}

void Milenage256_f3(
    u8 *key,            /* in,  u8[KEY_sz]      */
```

```
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *ck              /* out, u8[CK_sz]       */
)
{
    Milenage256_Main(3, CK_sz - 1, key, rand, NULL, NULL, NULL, ck, CK_sz);
}

void Milenage256_f4(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *ik              /* out, u8[IK_sz]       */
)
{
    Milenage256_Main(4, IK_sz - 1, key, rand, NULL, NULL, NULL, ik, IK_sz);
}

void Milenage256_f5(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *ak              /* out, u8[AK_sz]       */
)
{
    Milenage256_Main(5, AK_sz - 5, key, rand, NULL, NULL, NULL, ak, AK_sz);
}

void Milenage256_f5s(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *ak              /* out, u8[AK_sz]       */
)
{
    Milenage256_Main(6, AK_sz - 5, key, rand, NULL, NULL, NULL, ak, AK_sz);
}

void Milenage256_f5ss(
    u8 *key,            /* in,  u8[KEY_sz]      */
    u8 *rand,           /* in,  u8[RAND_sz]     */
    u8 *mac_s,          /* in,  u8[MAC_sz]      */
    u8 *ak              /* out, u8[AK_sz]       */
)
{
    Milenage256_Main(7, ((MAC_sz - 1) << 3) | (AK_sz - 5), key, rand, NULL, NULL, mac_s, ak, AK_sz);
}

#endif /* _SAGE256_MILENAGE256_ */
```

Annex B (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2024-02	SA3#115	S3-240403				TS skeleton	0.0.0
2024-02	SA3#115	S3-240817				TS skeleton using 3GPP template	0.0.1
2024-02	SA3#115	S3-240407				Addition of Introduction	0.1.0
2024-08	SA3#117	S3-243422				Addition of the text based on the selection of Rijndael-based Milenage-256 to specify Milenage-256 algorithm.	0.2.0
2024-11	SA3#119	S3-245104				TR 35.937 replaces TS 35.237, and there is an editorial clean up for presentation to TSG-SA.	0.3.0
2024-12	SA#106	SP-241789				Presented for information and approval	1.0.0
2025-01						Upgrade to change control version	19.0.0

History

Document history		
V19.0.0	January 2026	Publication