ETSI TS 135 206 V13.0.0 (2016-01)

TECHNICAL SPECIFICATION

Universal Mobile Telecommunications System (UMTS);
LTE;
3G Security;
Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
Document 2: Algorithm specification
(3GPP TS 35.206 version 13.0.0 Release 13)

Reference
RTS/TSGS-0335206vd00

Keywords
LTE,SECURITY,UMTS

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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, 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.

Foreword

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, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

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 (TS) 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.

Introduction

This document has been prepared by the 3GPP Task Force, and contains an example set of algorithms which may be used as the authentication and key generation functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. (It is not mandatory that the particular algorithms specified in this document are used — all seven functions are operator-specifiable rather than being fully standardised). This document is one five, which between them form the entire specification of the example algorithms, entitled:

- 3GPP TS 35.205: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 1: General".
- 3GPP TS 35.206: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 2: Algorithm Specification".
- 3GPP TS 35.207: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 3: Implementors' Test Data".
- 3GPP TS 35.208: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 4: Design Conformance Test Data".
- 3GPP TR 35.909: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 5: Summary and results of design and evaluation".

0 The name "MILENAGE"

The name of this algorithm set is "MILENAGE". It should be pronounced like a French word — something like "**mi-le-nahj**".

1 Outline of the document

Section 2 introduces the algorithms and describes the notation used in the subsequent sections.

Section 3 explains how the algorithms are designed as a framework in such a way that various "customising components" can be selected in order to customise the algorithm for a particular operator.

Section 4 defines the example algorithms. The algorithm framework is defined in section 4.1; in section 4.2, specific instances of the components are selected to define the specific example algorithm set.

Section 5 explains various options and considerations for implementation of the algorithms, including considerations to be borne in mind when modifying the customising components.

Illustrative pictures are given in Annex 1. Annex 2 gives a specification of the block cipher algorithm which is used as a cryptographic kernel in the definition of the example algorithms. Annexes 3 and 4 contain source code in the C programming language: Annex 3 gives a complete and straightforward implementation of the algorithm set, while Annex 4 gives an example of an alternative high-performance implementation just of the kernel function.

1.1 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 TS 33.102 v3.5.0: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture".

[2] 3GPP TS 33.105 v3.4.0: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements".

[3] 3GPP TS 35.206: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm Specification" (this document).

[4] 3GPP TS 35.207: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 3: Implementors' Test Data".

[5] 3GPP TS 35.208: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 4: Design Conformance Test Data".

[6] Joan Daemen and Vincent Rijmen: "AES Proposal: Rijndael", available at http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Rijndael/Rijndael.pdf or http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip

[7] http://csrc.nist.gov/encryption/aes/

[8] Thomas S. Messerges, "Securing the AES finalists against Power Analysis Attacks", in FSE 2000, Seventh Fast Software Encryption Workshop, ed. Schneier, Springer Verlag, 2000.

[9] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in CRYPTO'96, Lecture Notes in Computer Science #1109, Springer Verlag, 1996.

[10] J. Kelsey, B. Schneier, D. Wagner, C. Hall, "Side Channel Cryptanalysis of Product Ciphers", in ESORICS'98, Lecture Notes in Computer Science #1485, Springer Verlag, 1998.

[11] L. Goubin, J. Patarin, "DES and differential power analysis", in CHES'99, Lecture Notes in Computer Science #1717, Springer Verlag, 1999.

[12] P. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis", in CRYPTO'99, Lecture Notes in Computer Science #1666, Springer Verlag, 1999.

[13] L. Goubin, J.-S. Coron, "On boolean and arithmetic masking against differential power analysis", in CHES'00, Lecture Notes in Computer Science series, Springer Verlag (to appear).

2 INTRODUCTORY INFORMATION

2.1 Introduction

Within the security architecture of the 3GPP system there are seven security functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. The operation of these functions falls within the domain of one operator, and the functions are therefore to be specified by each operator rather than being fully standardised. The algorithms specified in this document are examples that may be used by an operator who does not wish to design his own.

The inputs and outputs of all seven algorithms are defined in section 2.4.

2.2 Notation

2.2.1 Radix

We use the prefix **0x** to indicate **hexadecimal** numbers.

2.2.2 Conventions

We use the assignment operator '=', as used in several programming languages. When we write

<variable> = <expression>

we mean that <*variable*> assumes the value that <*expression*> had before the assignment took place. For instance,

$$x = x + y + 3$$

means

(new value of x) becomes (old value of x) + (old value of y) + 3.

2.2.3 Bit/Byte ordering

All data variables in this specification are presented with the most significant bit (or byte) on the left hand side and the least significant bit (or byte) on the right hand side. Where a variable is broken down into a number of substrings, the leftmost (most significant) substring is numbered 0, the next most significant is numbered 1, and so on through to the least significant.

2.2.4 List of Symbols

=	The assignment operator.
⊕	The bitwise exclusive-OR operation
||	The concatenation of the two operands.
$E[x]_k$	The result of applying a block cipher encryption to the input value **x** using the key **k**.
rot(x,r)	The result of cyclically rotating the 128-bit value **x** by **r** bit positions towards the most significant bit. If **x** = **x[0]** || **x[1]** || … **x[127]**, and **y** = rot(**x**,**r**), then **y** = **x[r]** || **x[r+1]** || … **x[127]** || **x[0]** || **x[1]** || **x[r-1]**.
X[i]	The i^{th} bit of the variable **X**. (**X = X[0]** || **X[1]** || **X[2]** || **…..**).

2.3 List of Variables

AK	a 48-bit anonymity key that is the output of either of the functions ***f5*** and ***f5****.
AMF	a 16-bit authentication management field that is an input to the functions ***f1*** and ***f1****.
c1,c2,c3,c4,c5	128-bit constants, which are XORed onto intermediate variables.
CK	a 128-bit confidentiality key that is the output of the function ***f3***.
IK	a 128-bit integrity key that is the output of the function ***f4***.
IN1	a 128-bit value constructed from **SQN** and **AMF** and used in the computation of the functions ***f1*** and ***f1****.
K	a 128-bit subscriber key that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
MAC-A	a 64-bit network authentication code that is the output of the function ***f1***.
MAC-S	a 64-bit resynchronisation authentication code that is the output of the function ***f1****.
OP	a 128-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
OP_C	a 128-bit value derived from **OP** and **K** and used within the computation of the functions.
OUT1,OUT2,OUT3,OUT4,OUT5	128-bit computed values from which the outputs of the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5**** are obtained.
r1,r2,r3,r4,r5	integers in the range 0–127 inclusive, which define amounts by which intermediate variables are cyclically rotated.
RAND	a 128-bit random challenge that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
RES	a 64-bit signed response that is the output of the function ***f2***.
SQN	a 48-bit sequence number that is an input to either of the functions ***f1*** and ***f1****. (For ***f1**** this input is more precisely called SQN_{MS}.)
TEMP	a 128-bit value used within the computation of the functions.

2.4 Algorithm Inputs and Outputs

The inputs to the algorithms are given in tables 1 and 2, the outputs in tables 3–9 below.

Table 1: inputs to *f1* and *f1**

Parameter	Size (bits)	Comment
K	128	Subscriber key K[0]…K[127]
RAND	128	Random challenge RAND[0]…RAND[127]
SQN	48	Sequence number SQN[0]…SQN[47]. (For ***f1**** this input is more precisely called SQN_{MS}.)
AMF	16	Authentication management field AMF[0]…AMF[15]

Table 2: inputs to *f2*, *f3*, *f4*, *f5* and *f5**

Parameter	Size (bits)	Comment
K	128	Subscriber key K[0]…K[127]
RAND	128	Random challenge RAND[0]…RAND[127]

Table 3: *f1* output

Parameter	Size (bits)	Comment
MAC-A	64	Network authentication code MAC-A[0]…MAC-A[63]

Table 4: *f1 output**

Parameter	Size (bits)	Comment
MAC-S	64	Resynch authentication code MAC-S[0]…MAC-S[63]

Table 5. *f2* output

Parameter	Size (bits)	Comment
RES	64	Response RES[0]…RES[63]

Table 6. *f3* output

Parameter	Size (bits)	Comment
CK	128	Confidentiality key CK[0]…CK[127]

Table 7. *f4* output

Parameter	Size (bits)	Comment
IK	128	Integrity key IK[0]…IK[127]

Table 8. *f5* output

Parameter	Size (bits)	Comment
AK	48	Anonymity key AK[0]…AK[47]

Table 9. *f5 output**

Parameter	Size (bits)	Comment
AK	48	Resynch anonymity key AK[0]…AK[47]

Note: Both f5 and f5* outputs are called AK according to reference [2]. In practice only one of them will be calculated in each instance of the authentication and key agreement procedure.

3 The algorithm framework and the specific example algorithms

The example algorithm set makes use of the following components:

- A block cipher encryption function, which takes a 128-bit input and a 128-bit key and returns a 128-bit output. If the input is **x**, the key is **k** and the output is **y**, we write $\mathbf{y} = E[\mathbf{x}]_{\mathbf{k}}$.
- A 128-bit value **OP**. This is an Operator Variant Algorithm Configuration Field, which the Task Force was asked to include as a simple means to provide separation between the functionality of the algorithms when used by different operators. It is left to each operator to select a value for **OP**. The algorithm set is designed to be secure whether or not **OP** is publicly known; however, operators may see some advantage in keeping their value of **OP** secret. This and other aspects of the use of **OP** are discussed further in section 5.

In the specific example algorithm set, a particular block cipher is used. But the algorithms have been designed so that this component can be replaced by any operator who wishes to create his own customised algorithm set. In that sense this document defines an algorithm framework, and the example algorithm set is one that fits within the framework. This is how the algorithm set is defined in section 4: in section 4.1 the framework is defined in terms of the block cipher, and then in section 4.2 a block cipher is selected to give a fully specified algorithm set.

4 Definition of the example algorithms

4.1 Algorithm Framework

A 128-bit value $\mathbf{OP_C}$ is derived from **OP** and **K** as follows:

$\mathbf{OP_C} = \mathbf{OP} \oplus \mathrm{E}[\mathbf{OP}]_{\mathbf{K}}$.

An intermediate 128-bit value **TEMP** is computed as follows:

$\mathbf{TEMP} = \mathrm{E}[\mathbf{RAND} \oplus \mathbf{OP_C}]_{\mathbf{K}}$.
A 128-bit value **IN1** is constructed as follows:
IN1[0] .. **IN1**[47] = **SQN**[0] .. **SQN**[47]
IN1[48] .. **IN1**[63] = **AMF**[0] .. **AMF**[15]
IN1[64] .. **IN1**[111] = **SQN**[0] .. **SQN**[47]
IN1[112] .. **IN1**[127] = **AMF**[0] .. **AMF**[15]

Five 128-bit constants **c1**, **c2**, **c3**, **c4**, **c5** are defined as follows:

c1[i] = 0 for $0 \le i \le 127$
c2[i] = 0 for $0 \le i \le 127$, except that **c2**[127] = 1
c3[i] = 0 for $0 \le i \le 127$, except that **c3**[126] = 1
c4[i] = 0 for $0 \le i \le 127$, except that **c4**[125] = 1
c5[i] = 0 for $0 \le i \le 127$, except that **c5**[124] = 1
Five integers **r1**, **r2**, **r3**, **r4**, **r5** are defined as follows:
r1 = 64; **r2** = 0; **r3** = 32; **r4** = 64; **r5** = 96

Five 128-bit blocks **OUT1**, **OUT2**, **OUT3**, **OUT4**, **OUT5** are computed as follows:

$\mathbf{OUT1} = \mathrm{E}[\mathbf{TEMP} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1}]_{\mathbf{K}} \oplus \mathbf{OP_C}$
$\mathbf{OUT2} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r2}) \oplus \mathbf{c2}]_{\mathbf{K}} \oplus \mathbf{OP_C}$
$\mathbf{OUT3} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r3}) \oplus \mathbf{c3}]_{\mathbf{K}} \oplus \mathbf{OP_C}$
$\mathbf{OUT4} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r4}) \oplus \mathbf{c4}]_{\mathbf{K}} \oplus \mathbf{OP_C}$
$\mathbf{OUT5} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r5}) \oplus \mathbf{c5}]_{\mathbf{K}} \oplus \mathbf{OP_C}$

The outputs of the various functions are then defined as follows:

Output of ***f1*** = MAC-A, where MAC-A[0] .. MAC-A[63] = **OUT1**[0] .. **OUT1**[63]
Output of ***f1**** = MAC-S, where MAC-S[0] .. MAC-S[63] = **OUT1**[64] .. **OUT1**[127]
Output of ***f2*** = RES, where RES[0] .. RES[63] = **OUT2**[64] .. **OUT2**[127]
Output of ***f3*** = CK, where CK[0] .. CK[127] = **OUT3**[0] .. **OUT3**[127]
Output of ***f4*** = IK, where IK[0] .. IK[127] = **OUT4**[0] .. **OUT4**[127]
Output of ***f5*** = AK, where AK[0] .. AK[47] = **OUT2**[0] .. **OUT2**[47]
Output of ***f5**** = AK, where AK[0] .. AK[47] = **OUT5**[0] .. **OUT5**[47]

(The repeated reference to AK is not a mistake: AK is the name of the output of either ***f5*** or ***f5****, and these two functions will not in practice be computed simultaneously.)

4.2 Specific Example Algorithms

The specific example algorithm set is defined by specifying the block cipher encryption function E[], which we do in this section. (It is left to each operator to specify the Operator Variant Algorithm Configuration Field **OP**.)

The block cipher selected is Rijndael [6]. This is the algorithm proposed as the Advanced Encryption Standard [7]. More precisely, it is Rijndael with 128-bit key and 128-bit block size.

$\mathrm{E}[\mathbf{x}]_{\mathbf{k}}$ = the result of applying the Rijndael encryption algorithm
to the 128-bit value **x** under the 128-bit key **k**.

Although the definitive specification of Rijndael is in [6], a complete specification of Rijndael with 128-bit key and 128-bit block size is also given in Annex 2 of this document.

The inputs to and output of Rijndael are defined as strings of bytes. The 128-bit string **x** = **x[0]** || **x[1]** || … **x[127]** is treated as a string of bytes by taking **x[0]** || **x[1]** || … **x[7]** as the first byte, **x[8]** || **x[9]** || … **x[15]** as the second byte, and so on. The key and output string are converted in the same way.

Note that the following patent statement has been made publicly (and included in [6]) by the authors of the Rijndael algorithm: "Rijndael or any of its implementations is not and will not be subject to patents."

5 Implementation considerations

5.1 OP_C computed on or off the USIM?

Recall that **OP** is an Operator Variant Algorithm Configuration Field. It is expected that each operator will define a value of **OP** which will then be used for all its subscribers. (It is up to operators to decide how to manage **OP**. The value of **OP** used for new batches of USIMs could be changed occasionally; or perhaps a different value could be given to each different USIM supplier. **OP** could even be given a different value for every subscriber if desired, but that is not really the intention.)

It will be seen in section 4.1 that $\mathbf{OP_C}$ is computed from **OP** and **K**, and that it is only $\mathbf{OP_C}$, not **OP**, that is ever used in subsequent computations. This gives two alternative options for implementation of the algorithms on the USIM:

(a) **$\mathbf{OP_C}$ computed off the USIM:** $\mathbf{OP_C}$ is computed as part of the USIM prepersonalisation process, and $\mathbf{OP_C}$ is stored on the USIM. **OP** itself is not stored on the USIM.

(b) **$\mathbf{OP_C}$ computed on the USIM:** **OP** is stored on the USIM (it may be considered as a hard-coded part of the algorithm if preferred). $\mathbf{OP_C}$ is recomputed each time the algorithms are called.

The SAGE Task Force recommends that $\mathbf{OP_C}$ be computed off the USIM if possible, since this gives the following benefits:

- The complexity of the algorithms run on the USIM is reduced.
- It is more likely that **OP** can be kept secret. (If **OP** is stored on the USIM, it only takes one USIM to be reverse engineered for **OP** to be discovered and published. But it should be difficult for someone who has discovered even a large number of ($\mathbf{OP_C}$, **K**) pairs to deduce **OP**. That means that the $\mathbf{OP_C}$ associated with any other value of **K** will be unknown, which may make it harder to mount some kinds of cryptanalytic and forgery attacks. The algorithms are designed to be secure whether or not **OP** is known to the attacker, but a secret **OP** is one more hurdle in the attacker's path.)

5.2 Customising the choice of block cipher

It was explained in section 3 that an operator may create a variant algorithm set by selecting a block cipher other than Rijndael. It is vitally important that whatever block cipher is chosen is one that has been extensively analysed and is still believed to be secure. The security of the authentication and key generation functions is crucially dependent on the strength of the block cipher.

Strictly speaking, in fact, the kernel function does not have to be a block cipher; it just has to be a keyed function (with 128-bit input, key and output) satisfying the following cryptographic requirement:

- Let the key be fixed. Without initial knowledge of the key, but with a large number of pairs of chosen input and resulting output, it must be infeasible to determine the key, and also infeasible to predict the output for any other chosen input with probability significantly greater than 2^{-128}.

See also section 5.4 about protecting against side channel attacks; this will need to be borne in mind when selecting/implementing a replacement kernel function.