ETSI TS 135 231 V17.0.0 (2022-04)

Universal Mobile Telecommunications System (UMTS);
LTE;
Specification of the TUAK algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
Document 1: Algorithm specification
(3GPP TS 35.231 version 17.0.0 Release 17)

Reference
RTS/TSGS-0335231vh00

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

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 prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

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
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

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

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

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 Web server (https://ipr.etsi.org/).

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™** and **LTE™** 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 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 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

The present document is one of three, which between them form the entire specification of the example algorithms, entitled:

- 3GPP TS 35.231: "Specification of the Tuak algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 1: Algorithm specification ".

- 3GPP TS 35.232: "Specification of the Tuak algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 2: Implementers' test data".

- 3GPP TS 35.233: "Specification of the Tuak algorithm Set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
  Document 3: Design conformance test data".

1 Scope

The present document and the other Technical Specifications in the series, TS 35.232 [15] and 35.233 [16] contain an example set of algorithms which could be used as the authentication and key generation functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5**** for 3GPP systems. All seven functions are operator-specifiable rather than being fully standardised and other algorithms could be envisaged.

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 TS 33.102: "3G Security; Security Architecture3G 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.[3] "The KECCAK Reference", version 3.0, 14 January 2011, G. Bertoni, J. Daemen, M. Peeters, G. van Aasche, (available at http://keccak.noekeon.org/Keccak-reference-3.0.pdf).

[4] "KECCAK Implementation Overview", version 3.2, 29 May 2012, G. Bertoni, J. Daemen, M. Peeters, G. van Aasche, R. van Keer (available at http://keccak.noekeon.org/Keccak-implementation-3.2.pdf).

[5] "SAKURA: a flexible coding for tree hashing", 3 June 2013, G. Bertoni, J. Daemen, M. Peeters, G. van Aasche, (available at http://keccak.noekeon.org/Sakura.pdf).

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

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

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

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

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

[11] "On Boolean and Arithmetic Masking against Differential Power Analysis", in CHES'00, Lecture Notes in Computer Science series, Springer Verlag, 2000, L. Goubin, J.-S. Coron.

[12] 3GPP TS 33.401: "3GPP System Architecture Evolution (SAE); Security architecture".

[13] Void

[14] 3GPP TR 21.905: "Vocabulary for 3GPP specifications".

[15] 3GPP TS 35.232: "3G Security; Specification of the Tuak Algorithm Set: a Second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Implementers' test data".

[16] 3GPP TS 35.233: "3G Security; Specification of the Tuak Algorithm Set: a second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 3: Design conformance test data".

3 Definitions and symbols

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TR 21.905 [14] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [14].

Tuak: The name of this algorithm set is "Tuak". It should be pronounced like "too-ack".

3.2 Symbols

=	The assignment operator
⊕	The bitwise exclusive-OR operation
||	The concatenation of the two operands
X[i]	The i^{th} bit of the variable **X**. (**X = X[0] || X[1] || X[2] || …..**)
Π	the permutation Keccak-f[1600] (See clause 5.2 and annex C)

The following represent variables used in the algorithm:

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**
CK	a 128-bit or 256-bit confidentiality key that is the output of the function *f3*
IK	a 128-bit or 256-bit integrity key that is the output of the function *f4*
IN	a 1600-bit value that is used as the input to the permutation Π when computing the functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**
INSTANCE	an 8-bit value that is used to specify different modes of operation and different parameter lengths within the algorithm set
K	a 128-bit or 256-bit subscriber key that is an input to the functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**
MAC-A	a 64-bit, 128-bit or 256-bit network authentication code that is the output of the function *f1*
MAC-S	a 64-bit, 128-bit or 256-bit resynchronization authentication code that is the output of the function *f1**
OP	Operator Variant Algorithm Configuration Field (used in MILENAGE)
OUT	a 1600-bit value that is taken as the output of the permutation Π when computing the functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**
RAND	a 128-bit random challenge that is an input to the functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**
RES	a 32-bit, 64-bit, 128-bit or 256-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}.) See informative Annex C of [1] for methods of encoding sequence numbers
SQN_{MS}	(See SQN)
TOP	a 256-bit Operator Variant Algorithm Configuration Field that is a component of the functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**
TOP_C	a 256-bit value derived from TOP and K and used within the computation of the functions

4 Preliminary information

4.1 Introduction

Within the security architecture of the 3GPP system there are seven security functions related to authentication and key agreement: 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 standardized. The algorithms specified in the present document are examples that may be used by an operator who does not wish to design his own.

The algorithm specified is called Tuak (pronounced "too-ack").

It is not mandatory that the particular algorithms specified in the present document are used.

The inputs and outputs of all seven algorithms are defined in clause 4.4.

4.2 Notation

4.2.1 Radix

The prefix 0x is used to indicate hexadecimal numbers.

4.2.2 Bit-numbering for inputs and outputs

3GPP TS 33.102 [1] includes the following convention. (There is similar text in the specification of MILENAGE, as defined in 3GPP TS 35.206 [2]):

All data variables in the present document are presented with the most significant substring on the left hand side and the least significant substring on the right hand side. A substring may be a bit, byte or other arbitrary length bit string. Where a variable is broken down into a number of substrings, the left-most (most significant) substring is numbered 0, the next most significant is numbered 1, and so on through to the least significant.

So, for example, RAND[0] is the most-significant bit of RAND and RAND[127] is the least significant bit of RAND.

This convention applies to all inputs and outputs to Tuak, as listed in tables 1 to 9 below.

However, internally to the Tuak specification variables are simply treated as indexed bit strings, without a specific indication of bit, byte or word order.

4.2.3 Assignment operations

The assignment operator '='is used in many programming languages. Thus:

<*variable*> = <*expression*>

It means 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.

Also

<*variables*> = <*expressions*>

for lists of variables and expressions, then the left-most variable assumes the value the left-most expression had before the assignment took place, the next left-most variable assumes the value the next left-most expression had before the assignment took place, and so on.

For instance,

x[0]..*x*[2] = 3, 4, 5

means

(new value of *x*[0]) becomes 3,
(new value of *x*[1]) becomes 4,
(new value of *x*[2]) becomes 5.

Whereas:

x[0]..x[2] = y[2]..y[0]

means

(new value of *x*[0]) becomes (old value of *y*[2]),
(new value of *x*[1]) becomes (old value of *y*[1]),
(new value of *x*[2]) becomes (old value of *y*[0]).

4.2.4 Void

4.3 Void

5 Inputs and outputs

5.1 Tuak inputs and outputs

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

There are a few differences from the inputs and outputs to MILENAGE [2].

We allow tThe key K may be 128 bits or 256 bits. MAC-A and MAC-S may be 64, 128 or 256 bits. RES may be 32, 64, 128 or 256 bits. CK and IK may be 128 or 256 bits. Existing 3GPP specifications (see [1] and [12]) do not support all these possibilities, but they are included in Tuak for future flexibility in case future releases of these specifications may want to support them.

NOTE 1: The 3G security architecture specification [1] calls the output of the f1 function 'MAC' while the present document and [2] call it 'MAC-A'.

Any sizes for the parameters K, MAC-A, MAC-S, RES, CK and IK mentioned in the present document shall not be supported nor used in entities defined in 3GPP specifications until these specifications explicitly allow their use.

In any particular implementation, the parameters shall have a fixed length, chosen in advance. For example an operator may fix K at length 256 bits, RES at length 64 bits, CK and IK at length 128 bits. As the lengths do not vary with input, they are not specified as formal input parameters.

Table 1: Inputs to *f1* and *f1

Parameter	Size (bits)	Comment
K	128 or 256	Subscriber key K[0]…K[127] or K[0]…K[255]
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 or 256	Subscriber key K[0]…K[127] or K[0]…K[255]
RAND	128	Random challenge RAND[0]…RAND[127]

Table 3: f1 output

Parameter	Size (bits)	Comment
MAC-A	64, 128 or 256	Network authentication code MAC-A[0]…MAC-A[63] or MAC-A[0]…MAC-A[127] or MAC-A[0]…MAC-A[255]

Table 4: *f1 output**

Parameter	Size (bits)	Comment
MAC-S	64, 128 or 256	Resynch authentication code MAC-S[0]…MAC-S[63] or MAC-S[0]…MAC-S[127] or MAC-S[0]…MAC-S[255]

Table 5: *f2* output

Parameter	Size (bits)	Comment
RES	32, 64, 128 or 256	Response RES[0]…RES[31] or RES[0]…RES[63] or RES[0]…RES[127] or RES[0]…RES[255]

Table 6: *f3* output

Parameter	Size (bits)	Comment
CK	128 or 256	Confidentiality key CK[0]…CK[127] or CK[0]…CK[255]

Table 7: *f4* output

Parameter	Size (bits)	Comment
IK	128 or 256	Integrity key IK[0]…IK[127] or IK[0]…IK[255]

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 2: Both f5 and f5* outputs are called AK according to [1]. In practice only one of them at a time will be calculated in any given call to the authentication and key agreement algorithms.

5.2 Keccak and its inputs and outputs

This clause refers to the Keccak reference specification [3]. Use is made of the permutation Keccak-f[1600], which is abbreviated to Π, and defined formally in Annex C.

We use Strings **IN**[0] .. **IN**[1599] and **OUT**[0] .. **OUT**[1599] are used to represent the input and output of Π. As in [3], these are treated as simple bit strings. However, to support efficient implementations of Keccak (see [4]), inputs are mapped to **IN** and outputs are extracted from **OUT** in such a way that bits of input and output should not need to be reversed within bytes for such implementations.

The Keccak specification includes the concept of a security parameter which the designers call "capacity". Based on the designers' recommendations, a formal capacity of 512 bits is used: all input strings to the Keccak permutation shall be padded to 1088 bits, and then have 512 zero bits appended. The padding used to extend the input string to 1088 bits is the "1 0* 1" padding defined in [3], immediately preceded by "1 1 1 1" for consistency with Sakura coding and domain separation (see e.g. " SAKURA: a flexible coding for tree hashing" [5], start of section 6 for the Sakura coding, and

Table 4 for the domain separation coding). Note that our input strings before padding are always shorter than 832 bits, with the remaining bits of IN always the same in all modes, and output is only ever extracted from the first 832 bits of OUT – so in practice the effective capacity of the construction is at least 1600 - 832 = 768 bits.

5.3 Other inputs and substrings

MILENAGE uses a 128-bit value **OP**, and derives a 128-bit value $\mathbf{OP_C}$. **OP** is an Operator Variant Algorithm Configuration Field.

For Tuak a 256-bit Operator Variant Algorithm Configuration Field is specified, **TOP**; and a derived 256-bit value $\mathbf{TOP_C}$.

The following internal variables are defined in the algorithm definition:

- A 56-bit string **ALGONAME**[0] .. **ALGONAME**[55], with an arbitrary fixed value. This is specified as the ASCII representation of the string "TUAK1.0": to be explicit, **ALGONAME**[0] .. **ALGONAME**[55] = 0,1,0,1,0,1,0,0, 0,1,0,1,0,1,0,1, 0,1,0,0,0,0,0,1, 0,1,0,0,1,0,1,1, 0,0,1,1,0,0,0,1, 0,0,1,0,1,1,1,0, 0,0,1,1,0,0,0,0

- An 8-bit string **INSTANCE**[0] .. **INSTANCE**[7] which will be given different values for different algorithms within the set.

The internal variable **INSTANCE** is coded using the following schema (sections 6 gives the exact details) :

INSTANCE[0] .. **INSTANCE**[1] indicate which function is being implemented

INSTANCE[2]…**INSTANCE**[4] indicate the length of the MAC-A/MAC-S or RES output,

or they are all set to zero when deriving $\mathbf{TOP_C}$

INSTANCE[5] .. **INSTANCE**[7] indicate whether the CK/IK/K lengths are 256 bit.

6 Definition of the example algorithms

6.1 Derivation of $\mathbf{TOP_C}$

The **INSTANCE** variable is constructed as follows:

INSTANCE[0] .. **INSTANCE**[6] = 0,0,0,0,0,0,0

INSTANCE[7] = 0 if the length of K is 128 bits

= 1 if the length of K is 256 bits

The 1600-bit value **IN** is then constructed as follows:

IN[0] .. **IN**[255] = **TOP**[255] .. **TOP**[0]

IN[256] .. **IN**[263] = **INSTANCE**[7] .. **INSTANCE**[0]

IN[264] .. **IN**[319] = **ALGONAME**[55] .. **ALGONAME**[0]

IN[i] = 0 for $320 \leq i \leq 511$

IN[512] .. **IN**[767] = **K**[255] .. **K** [0] if the length of K is 256 bits

IN[512] .. **IN**[639] = **K**[127] .. **K** [0] if the length of K is 128 bits

IN[i] = 0 for $640 \leq i \leq 767$ if the length of K is 128 bits

IN[i] = 1 for $768 \leq i \leq 772$

IN[i] = 0 for $773 \leq i \leq 1086$