ETSI TR 135 934 V12.0.0 (2015-01)

Universal Mobile Telecommunications System (UMTS);
LTE;
Report on the design and evaluation of the Tuak algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions;
(3GPP TR 35.934 version 12.0.0 Release 12)

Reference
DTR/TSGS-0335934vc00

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

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 (http://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 Report (TR) 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**", "**may not**", "**need**", "**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 Report 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.

1 Scope

The present document (together with three accompanying documents, [8], [9] and [10] describes the design rationale, and presents evaluation results, on the Tuak algorithm set [5] – a second example set of algorithms which may be used as the authentication and key generation functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****, e.g. as an alternative to MILENAGE.

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 33.102: "3G Security; Security Architecture", (available at http://www.3gpp.org/ftp/specs/html-info/33102.htm).

[3] 3G TS 33.105 (V 3.4.0) (2000-07): "3G Security; Cryptographic Algorithm Requirements (Release 1999)".

[4] 3GPP TS 35.206: "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", (available at http://www.3gpp.org/ftp/Specs/html-info/35206.htm).

[5] 3GPP TS 35.231: "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 1: Algorithm specification", (available at http://www.3gpp.org/ftp/Specs/html-info/35231.htm).

[6] 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", (available at http://www.3gpp.org/ftp/Specs/html-info/35232.htm).

[7] 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", (available at http://www.3gpp.org/ftp/Specs/html-info/35233.htm).

[8] "Security Assessment of Tuak Algorithm Set", Guang Gong, Kalikinkar Mandal, Yin Tan and Teng Wu, included as an accompanying document to the present report (available at http://www.3gpp.org/ftp/Specs/archive/35_series/35.935/SAGE_report/Secassesment.zip).

[9] "Performance Evaluation of the Tuak algorithm in support of the ETSI SAGE standardisation group", Keith Mayes, included as an accompanying document to the present report (available at http://www.3gpp.org/ftp/Specs/archive/35_series/35.936/SAGE_report/Perfevaluation.zip).

[10] "Performance Evaluation of the Tuak algorithm in support of the ETSI SAGE standardisation group – extension report", Keith Mayes, included as an accompanying document to the present report (available at http://www.3gpp.org/ftp/Specs/archive/35_series/35.936/SAGE_report/Perfevaluationext.zip).

[11] "Note on side-channel attacks and their countermeasures", G. Bertoni, J. Daemen, M. Peeters, G. van Assche (available at http://keccak.noekeon.org/NoteSideChannelAttacks.pdf).

[12] "Building power analysis resistant implementations of Keccak", G. Bertoni, J. Daemen, M. Peeters, G. van Assche (available at http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/BERTONI_KeccakAntiDPA.pdf).

[13] Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies, http://www.wassenaar.org.

[14] "Announcing Draft Federal Information Processing Standard (FIPS) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, and Draft Revision of the Applicability Clause of FIPS 180-4, Secure Hash Standard, and Request for Comments", NIST, 28th May 2014, available at https://www.federalregister.gov/articles/2014/05/28/2014-12336/announcing-draft-federal-information-processing-standard-fips-202-sha-3-standard-permutation-based.

[15] "Early Symmetric Crypto (ESC) seminar 2013" (available at https://www.cryptolux.org/mediawiki-esc2013/index.php/ESC_2013)

[16] "The KECCAK sponge function family" (available at http://www.noekeon.org)

[17] https://www.cdt.org/blogs/joseph-lorenzo-hall/2409-nist-sha-3

[18] http://yro.slashdot.org/story/13/09/28/0219235/did-nist-cripple-sha-3

[19] https://www.schneier.com/blog/archives/2013/10/will_keccak_sha-3.html

[20] http://keccak.noekeon.org/yes_this_is_keccak.html

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 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 TR 21.905 [1].

Keccak: algorithm selected as the winner of the SHA-3 competition

MILENAGE: previously designed example algorithm set for the 3GPP Authentication and Key Generation Functions

TOP_C: value derived from TOP and K and used within the computations of the functions *f1, f1*, f2, f3, f4, f5* and *f5**

Tuak: newly designed example algorithm set for the 3GPP Authentication and Key Generation Functions. It should be pronounced like "too-ack"

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 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 TR 21.905 [1].

AES	Advanced Encryption Standard block cipher
AK	Anonymity Key
AMF	Algorithm Management Field
AuC	Authentication Centre
CK	Cipher Key
CPU	Central Processing Unit
DEMA	Differential Electromagnetic Analysis
DPA	Differential Power Analysis
IC	Integrated Circuit

IK	Integrity Key
K	Long lived subscriber unique key
MAC	Message Authentication Code
MAC-A	MAC for normal authentication vectors
MAC-S	MAC for resynchronization vectors
MULTOS	Multi-application smart card operating system
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NVM	Non-Volatile Memory
RAM	Random Access Memory
RAND	Random input parameter to authentication and key generation functions
RES	Response value
RNC	Radio Network Controller
ROM	Read-Only Memory
SAGE	Security Algorithms Group of Experts

NOTE: This is an ETSI Technical Committee.

SHA-2	Secure Hash Algorithm already standardized by NIST
SHA-3	Secure Hash Algorithm soon to be standardized by NIST
TOP	Tuak Operator Variant Algorithm Configuration Field
SEMA	Simple Electromagnetic Analysis
SIM	Subscriber Identity Module
SPA	Simple Power Analysis
SQN	Sequence Number
UICC	Universal Integrated Circuit Card
USIM	Universal Subscriber Identity Module
XMAC	Expected MAC value

4 Structure of this report

The main content of the present document is organized as follows:

- Clause 5 and 6 give the requirements and background that were considered during the design of Tuak – first recalling the functional and performance requirements that were used for MILENAGE, then noting some differences and additional points that apply for Tuak.
- Clause 7 gives a brief overview of the Tuak design.
- Clause 8 runs through choices made during the design of Tuak, and the reasons behind those choices.
- Clause 9 introduces independent assessments that have been carried out on the security and performance of Tuak. The full independent assessment reports are included as companion documents to this one.
- Clause 10 gives some further observations on software implementation and protection against side channel attacks.
- Clause 11 concludes with an overall assessment of Tuak's fitness for purpose.

Three further documents [8], [9] and [10] complete the present document, as explained in clause 9.

5 Background to the design and evaluation work

The 3rd Generation Partnership Project (3GPP) is a global initiative dedicated to the development of specifications for the next generations of cellular mobile systems. Integration of strong security services is an important feature of this system and the general security architecture is defined in ref. [2]. The implementation of these security services should be based on a variety of cryptographic functions/algorithms.

Out of the full algorithm suite, only the UMTS encryption algorithms (***f8***) and the UMTS integrity algorithms (***f9***) are fully standardized. ***f0*** represents a random number generation algorithm, and has no standardization or interoperability

requirements at all. The remaining cryptographic functions for authentication and key agreement (***f1*** – ***f5****) are allocated to the Authentication Centre (AuC) and the USIM; this means that the functions are proprietary to the home environment, and there is no need for formal standardization of these algorithms. However, there are good reasons to have a well trusted example set of functions available for this purpose, for use by operators that choose not to develop their own solutions. The MILENAGE algorithm set [4] was created to meet this need.

There are also good reasons to have a second trusted example set of (***f1*** – ***f5****) algorithms available:

- To have a fallback already in place in case MILENAGE is ever compromised.
- In particular, for the embedded UICC, where it may be sensible to have two strong algorithms installed on the platform and available for selection by subsequently loaded USIM applications. This provides choice to operators; it also provides resilience against future cryptanalysis of either algorithm, in devices that may have a long lifetime in the field.

The Tuak algorithm set [5], [6] and [7] has been created to serve as this second trusted example algorithm set.

6 Summary of algorithm requirements

6.0 Introduction

When MILENAGE was created, the requirements specification was taken from [3]. Clauses 6.1 and 6.2 below reproduce the main requirements necessary to understand the present document. Clause 6.3 describes some new requirements that came into play when designing Tuak.

6.1 General requirements for 3GPP cryptographic functions and algorithms (as stated for MILENAGE)

The functions should be designed with a view to their continued use for a period of at least 20 years. Successful attacks with a workload significantly less than exhaustive key search through the effective key space should be impossible.

The designers of above functions should design algorithms to a strength that reflects the above qualitative requirements.

Legal restrictions on the use or export of equipment containing cryptographic functions may prevent the use of such equipment in certain countries.

It is the intention that UE and USIMs that embody such algorithms should be free from restrictions on export or use, in order to allow the free circulation of 3G terminals. Network equipment, including RNC and AuC, may be expected to come under more stringent restrictions. It is the intention that RNC and AuC that embody such algorithms should be exportable under the conditions of the Wassenaar Arrangement, see reference [13].

6.2 Authentication and key agreement functions (as stated for MILENAGE)

6.2.0 Introduction

The mechanisms for authentication and key agreement described in clause 6.3 of [2] require the following cryptographic functions:

f1 The network authentication function;

f1* The re-synchronization message authentication function;

f2 The user authentication function;

f3 The cipher key derivation function;

f4 The integrity key derivation function;

f5 The anonymity key derivation function;

f5* The anonymity key derivation function for re-synchronization.

6.2.1 Implementation and operational considerations

The functions ***f1***–***f5**** should be designed so that they can be implemented on an IC card equipped with an 8-bit microprocessor running at 3,25 MHz with 8 kbyte ROM and 300 byte RAM and produce AK, XMAC-A, RES, CK and IK in less than 500 ms execution time.

6.2.2 Type of algorithm

6.2.2.1 *f1*

f1: the network authentication function

f1: (K; SQN, RAND, AMF) → MAC-A (or XMAC-A)

f1 should be a MAC function. In particular, it should be computationally infeasible to derive K from knowledge of RAND, SQN, AMF and MAC-A (or XMAC-A).

6.2.2.2 *f1**

f1*: the re-synchronization message authentication function

f1*: (K; SQN, RAND, AMF) → MAC-S (or XMAC-S)

f1* should be a MAC function. In particular, it should be computationally infeasible to derive K from knowledge of RAND, SQN, AMF and MAC-S (or XMAC-S).

6.2.2.3 *f2*

f2: the user authentication function

f2: (K; RAND) → RES (or XRES)

f2 should be a MAC function. In particular, it should be computationally infeasible to derive K from knowledge of RAND and RES (or XRES).

6.2.2.4 *f3*

f3: the cipher key derivation function

f3: (K; RAND) → CK

f3 should be a key derivation function. In particular, it should be computationally infeasible to derive K from knowledge of RAND and CK.

6.2.2.5 *f4*

f4: the integrity key derivation function

f4: (K; RAND) → IK

f4 should be a key derivation function. In particular, it should be computationally infeasible to derive K from knowledge of RAND and IK.

6.2.2.6 *f5*

f5: the anonymity key derivation function

f5: (K; RAND) → AK

f5 should be a key derivation function. In particular, it should be computationally infeasible to derive K from knowledge of RAND and AK.

The use of ***f5*** is optional.

6.2.2.7 *f5**

f5*: the anonymity key derivation function for re-synchronization

f5*: (K; RAND) → AK

f5* should be a key derivation function. In particular, it should be computationally infeasible to derive K from knowledge of RAND and AK.

The use of ***f5**** is optional.

6.3 Tuak-specific requirements

6.3.1 Difference from MILENAGE

It is important that this new algorithm should be fundamentally different from MILENAGE, in such a way that any advance in cryptanalysis that impact the security of one algorithm set should be unlikely to impact the security of the other.

6.3.2 256-bit key support

MILENAGE was designed for UMTS, before LTE had been standardized. UMTS supports only a 128-bit K value, but LTE allows K to be either 128 or 256 bits long. It was therefore felt highly desirable for Tuak to accommodate either a 128-bit or a 256-bit K.

6.3.3 Operator customization

MILENAGE allows some customization by each mobile operator. In particular, each operator needs to choose its own value of an "Operator Variant Algorithm Configuration Field" called OP. There are also other constants within the MILENAGE algorithm that can be varied if required. This operator customization serves two main purposes:

- It means that USIMs for different operators are not interchangeable, either through trivial modification of inputs and outputs or by reprogramming of a blank USIM.
- By keeping some algorithm details secret, some attacks (such as side channel attacks like power analysis) become a little harder to carry out.

This operator customization was not a required feature when MILENAGE was being design, but it has been well received by operators. It was therefore felt desirable to include a similar operator customization feature in Tuak.

6.3.4 Implementation and operational considerations

The performance and complexity requirements stated at the time of the MILENAGE design stipulate a maximum run time of 500 ms on "an IC card equipped with an 8-bit microprocessor running at 3,25 MHz with 8 kbyte ROM and 300 byte RAM". As noted in [9], however:

> Technology has moved on quite significantly and it might be quite hard to even find a SIM chip that has these minimal capabilities, and indeed many do not have ROM any more. Furthermore the target is a little ambiguous and could be interpreted that if you ran all the functions in sequence each could take 500 ms. It is also not clear how much of the ROM and RAM can be used. A more appropriate and modern target could be:
>
> *"The functions **f1–f5** and **f1*** should be designed so that they can be implemented on a midrange microprocessor IC card (typically 16-bit CPU), occupying no more than 8kbytes nonvolatile-memory (NVM), reserving no more than 300 bytes of RAM and producing AK, XMACA, RES, CK and IK in less than 500 ms total execution time."*

7 Overview of the Tuak design

The detailed specifications of the 3GPP Tuak algorithms can be found in [5]. The following diagram illustrates the design of the Tuak algorithms. See clause 8.10 for an explanation of TOP and TOP_C.

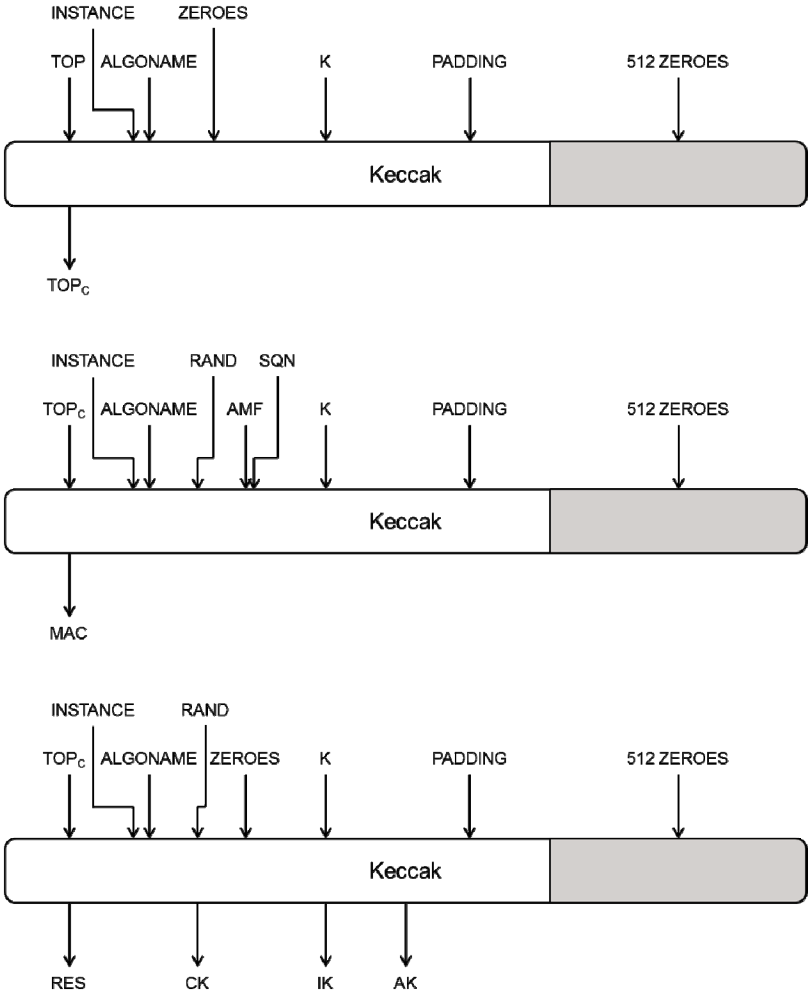

Figure 1

8 Design rationale

8.0 Introduction

This clause outlines the decision process that led to the Tuak design.

8.1 Brand new design, or design based on an existing public algorithm?

For obvious reasons of confidence in the design – both SAGE's confidence and the public's confidence – it was preferable to base the new authentication and key agreement algorithm on a well studied existing public algorithm.

8.2 Block cipher, stream cipher, MAC or hash function?

While it is possible in principle to use a public key algorithm as a building block, this would intuitively be a strange choice in a naturally symmetric key context, and a brief analysis indicated only dizadvantages. So in practice it makes sense to take a symmetric key or keyless algorithm as the starting point.

Stream ciphers would be a poor fit here – they typically need a pre-run before output can be extracted, making them expensive for short data sizes, even though they may be very efficient for large quantities of data. And there are no dedicated MAC functions (as opposed to e.g. block ciphers in a MAC mode) with enough public trust. So, realistically, the candidate building blocks are block ciphers or hash functions.

At first glance, block ciphers might seem a more natural starting point, since it might be possible to slot them directly into the MILENAGE framework, in place of AES. But after looking at the obvious choices of well established public domain block ciphers, it appeared that they all had limitations:

- CAMELLIA, SERPENT and ARIA all seem too close to AES – if an advance in cryptanalytic theory threatens AES, it was thought too likely that it would threaten these algorithms too. One of SAGE's fundamental design goals was to avoid this.
- IDEA, SEED and TEA / XTEA don't support a 256-bit key. Accommodating a 256-bit key was considered desirable, since LTE allows that.
- CAST, CLEFIA and RC6 seemed to be encumbered with IPR. SAGE preferred to choose a building block that was open for public use without IPR considerations.
- BLOWFISH and TWOFISH use key dependent S-boxes, which SAGE felt were likely to add implementation or computational overheads.
- MARS was felt by many commentators during the AES competition not to be well suited for smart card implementation.

Note that there is no need for any decryption process as part of the authentication and key agreement algorithm. A one-way function, such as a hash function, is fine for this purpose.

So SAGE determined on a hash function as the best type of building block to use.

8.3 Which hash function?

The ideal starting point would be a hash function that was well studied, reasonably efficient, and appeared to have a good security margin. When SAGE began the design process, the SHA-3 competition was still under way, with five finalists selected (BLAKE, Grøstl, JH, Keccak and Skein); the other natural candidate was SHA-2. Grøstl was quickly dismissed as being too close to AES. Once Keccak was selected as the SHA-3 winner, however, it would have been

perverse to choose any of the other SHA-3 finalists without a clear reason to do so, and there was no such clear reason – Keccak seemed an excellent candidate for our purposes. So, SAGE's choice was between Keccak and SHA-2.

Either of Keccak or SHA-2 would have been a good choice for the core building block. Referring back to the reasons above for rejecting various block ciphers, it was noted that both Keccak and SHA-2 are very different from AES in their designs; can easily accommodate a 256-bit key; are believed to be free of restrictive IPR; and could be reasonably compact and efficient on smartcard platforms. There were arguments in favour of each:

- SHA-2 had been in the public eye for longer. However, Keccak was more intensively scrutinised during the SHA-3 competition.
- More implementations were available for SHA-2.
- Keccak could become a particular focus for attack after being announced as the SHA-3 winner.
- Keccak's recommended MAC construction is simpler and more efficient than the HMAC construction that would normally be used to create a MAC from SHA-2. (Simpler MAC constructions are possible for SHA-2, and would in practice be fine for our purposes … but would be less well trusted than HMAC.)
- There were already publications showing how to realize Keccak in a way that protects against side channel attacks, with a modest implementation overhead. Protecting SHA-2 against side channel attacks seemed less straightforward, and the complexity of doing so less clear. Side channel attacks are of course very important for USIM-based authentication and key agreement algorithms. Clause 9 gives more information about this subject.
- Although both SHA-2 and Keccak currently seem to have a good security margin, the design philosophy and security arguments for Keccak are significantly clearer.
- The large input and output size of the Keccak permutation allow a very simple, one-round construction.

The SAGE chairman took the opportunity to canvas the opinion of a number of eminent symmetric cryptography specialists, at the Early Symmetric Crypto conference [15]. A substantial majority expressed a preference for Keccak as the more robust choice with the sounder security justification (and that is true even excluding the Keccak designers who were there.).

8.4 What sort of Keccak function to use

Keccak is a family of "cryptographic sponge functions" [16]. Each instance of Keccak has a particular permutation at its core. The SHA-3 submission takes a particular permutation from this family, and proposes a hash function based on the sponge paradigm.

SAGE could have waited for the SHA-3 standard to be published, and used SHA-3 in its entirety as a building block for Tuak. There might have been some advantage in doing so, in terms of public trust in the design. This would have entailed a long delay, however: NIST did not publish their draft SHA-3 standard until May 2014 [14], with public comments invited until August 2014, and the final standard to appear some time later. Moreover, using the exact input/output interface of SHA-3 would probably make the design significantly more complicated than it needed to be. SAGE decided instead to base our design on the Keccak sponge function, which in practice is what provides all the security assurance for SHA-3.

8.5 Keccak parameter selection

Keccak permutations are available in seven different sizes, operating on blocks of 25, 50, 100, 200, 400, 800 or 1600 bits. For the sizes of input and output parameters needed for Tuak, 800-bit or 1600-bit block sizes were the only two worthy of consideration.

Once a block size B is chosen, there are two further parameters in the choice of Keccak function, called the rate R and the capacity C, with the constraint that $R + C = B$. The capacity C is a security parameter: to achieve 256-bit security against all attacks (reflecting the maximum subscriber key size of 256 bits in LTE), it is necessary to choose a capacity of at least $2 \times 256 = 512$. The rate R indicates how many bits of input can be fed into each instance of the permutation, and how many bits of output can be extracted from each instance of the permutation.

With a 1600-bit block size B, the rate R can be up to $1600 – 512 = 1088$ bits; this is easily enough to accommodate all the inputs and outputs of each Tuak function in a single Keccak permutation instance. With an 800-bit block size B,

however, the rate R can be at most 800 – 512 = 288 bits; this would mean that several Keccak permutation iterations would be needed to accommodate all the required inputs and outputs. So, while the 800-bit block size might naïvely seem more efficient than the 1600-bit size, in practice this would not be the case.

Another clear argument in favour of the 1600-bit permutation block size is that this is what will be used in SHA-3. Choosing a smaller block size would risk unfavourable comparison with SHA-3, and a public impression that Tuak was not as secure as it could have been. It is also true that using the same block size as SHA-3 makes it more likely that implementations developed for the NIST standards can be largely reused for Tuak.

For these reasons SAGE selected the 1600-bit permutation size for Keccak. Being able to accommodate all inputs and outputs in a single Keccak permutation allows a rather simpler construction than was necessary for MILENAGE, where the basic building block was a 128-bit block cipher.

The largest set of inputs or outputs that one might want to deal with in a single Tuak function was 816 bits in total, implying that a rate $R \geq 816$ should be chosen for greatest efficiency. SAGE could, therefore, have chosen C up to 784. After a brief discussion with the Keccak design team (who were very helpful and responsive throughout our design process), SAGE decided to choose C = 512, for close alignment with the forthcoming NIST standards. Setting C = 512 means that the "rightmost" 512 input bits of each 1600-bit Keccak input block are always set to zero, and that no Tuak output bits are ever extracted from the "rightmost" 512 bits of the 1600-bit Keccak permutation output. (See figure 1 in clause 7.)

However, it can be argued that in practice Tuak has an effective capacity of at least 768 bits – see clause 3.2 of the Tuak algorithm specification [5].

8.6 Security evaluation of Keccak

8.6.0 Introduction

Obviously, a very intensive public expert evaluation of Keccak took place during the SHA-3 competition – there is no need to quote from this in detail here. SAGE's job as Tuak designers was to make sure that Tuak benefited from the security of Keccak, and the expert and general public confidence in Keccak, with good efficiency and to provide the required functionality.

8.6.1 What about the internet stories about NIST weakening SHA-3?

There were suggestions from some quarters in 2013 that NIST (maybe under influence from NSA) were deliberately weakening SHA-3 (or, as some people carelessly wrote, deliberately weakened Keccak). See for example [17], [18] and [19].

SAGE always felt that these "reduced security" concerns were unjustified. One may refer also to the Keccak team's own response in [20]. Here they made it clear:

- Firstly, that NIST have made no changes at all to Keccak itself, which remains exactly as the Keccak team designed it.

- Secondly, that what NIST were doing was to propose choices of the parameters C and R (capacity and rate, as mentioned above) that give either a very strong or an extremely strong level of security, while not unnecessarily hampering performance; and that these proposed choices are all part of the original Keccak family proposal, again with no changes at all.

- Thirdly: although it's true that the original SHA-3 submission used a larger value for the capacity C, this was done to meet an originally stated but frankly rather meaningless security target – namely, the target of 512-bit pre-image resistance for the 512-bit hash function. Conventional 512-bit hash function designs like SHA-512 do naturally achieve this target (unless "broken") – which is probably why it was stated as a requirement in the original NIST call for submissions – but the sponge function design approach used by Keccak does not. Instead, a capacity of C bits leads to a security level of C/2 bits against all attacks, so C=512 implies a security level of 256 bits against all attacks, including 256-bit pre-image resistance. NIST recognized – and the Keccak team themselves repeatedly argued – that 256-bit pre-image resistance is strong enough for all purposes, and that setting C any higher than 512 just reduces performance without serving any real purpose.

However, public confidence is very important. And as it turned out, NIST decided in the end that in the interests of public confidence it was necessary to revert to the original (less efficient but ostensibly more secure) capacity value. So the draft SHA-3 standard [14] uses the original higher capacity value, exactly as in the original Keccak proposal for SHA-3, and there is no longer any plausible reason to suspect any intentional weakness.

8.7 A note on IPR

8.7.1 Keccak IPR

Submitters to the SHA-3 competition signed a statement that, amongst other text, promised: "*Should my submission be selected for SHA-3, I hereby agree not to place any restrictions on the use of the algorithm, intending it to be available on a worldwide, non-exclusive, royalty-free basis.*"

8.7.2 Tuak IPR

SAGE is not aware of any IPR relating to the construction of Tuak from the Keccak sponge function core.

8.8 Padding bits

As well as the 3GPP-specified input fields, and the 512 zero bits dictated by the choice of capacity explained in clause 8.5, it is necessary to make up the 1600-bit input block to the Keccak permutation by including padding bits. (Note that these should be fixed padding bits, the same at the Authentication Centre end and the USIM end – one cannot use random padding bits here.) The choice of padding bits has little effect on security in this context. Again following discussion with the Keccak designers, SAGE chose padding bits that were expected to align with NIST's, to give the best possible chance that implementations developed for the NIST standards could be reused as much as possible for Tuak.

The publication by NIST of the draft SHA-3 standard [14] is encouraging in this regard – see clause 10.2.

8.9 Flexible input and output sizes

As already mentioned, it was desirable to accommodate the 256-bit subscriber key option in LTE (which is not yet widely used, but could become more popular in future). But also, bearing this 256-bit security level in mind, it was felt that it was prudent also to allow for some possible future increases in the sizes of other 3GPP security parameters, such as the cipher key CK, the integrity key IK and the Message Authentication Codes. This was a nice-to-have, not a hard requirement on the Tuak design, but since 1600-bit Keccak can accommodate these extensions with no loss of efficiency, they were included as options in Tuak. There is no immediate expectation that other 3GPP standards will incorporate these extended security parameter sizes, and indeed it may never happen – but the option is there if required.

8.10 Operator customization

Clause 6.3.3 explains how MILENAGE allows some customization by each mobile operator, and the benefits that this brings. In MILENAGE, each operator needs to choose its own value of an "Operator Variant Algorithm Configuration Field" called OP. In similar vein, Tuak includes an Operator Variant Algorithm Configuration Field, this time called TOP.

MILENAGE also included a feature whereby an intermediate value OP_C is derived from OP and the secret key K, and it is sufficient for the SIM card to be programmed with OP_C rather than with OP itself. This means that an attacker who is able to extract OP_C from one card does not learn OP or OP_C for other cards. An equivalent feature was included for Tuak, with a value (called TOP_C in the case of Tuak) being derived from TOP and K, in such a way that a SIM card can include only TOP_C, and TOP itself need never appear on a SIM.

The algorithm specification [5] also describes a very straightforward way in which Tuak could be modified to provide a higher security margin if ever required (if, say, some unexpected cryptanalysis emerged against Keccak), by performing multiple iterations of the Keccak sponge function instead of just one. At the moment, though, Keccak is widely agreed to have a very high security margin, and it is considered unlikely that this further-strengthened version will be needed.

9 Independent security and performance evaluation

9.0 Introduction

Independent expert analysis has been carried out on both the security and the SIM card performance of Tuak.

9.1 Independent security evaluation

A team at the University of Waterloo, led by Professor Guang Gong, carried out an independent assessment of the security of Tuak. Their extensive report [8] is included as an accompanying document to this one.

This report considers known cryptanalytic attacks, as well as a new type of attack, and concludes that the Tuak algorithms appear to resist each of these attacks. It also provides a security proof, showing that any weakness in Tuak would translate to a weakness in Keccak.

The conclusion is that "the algorithms in Tuak perfectly inherit the good security performance from Keccak and it can be used with confidence as message authentication functions and key derivation function".

9.2 Independent SIM card performance evaluation

An independent assessment of Tuak's performance and implementation complexity on SIM card platforms was carried out by the Smart Card Centre at Royal Holloway University of London. The resulting report [9] is again included as an accompanying document to this one.

The report concludes that:

- If Tuak is implemented as native code, the speed and performance goals can be met very comfortably, without the need for a crypto coprocessor, including on a low-end platform (16-bit or even 8-bit).
- Native code implementation does appear to be necessary to achieve the performance targets – an implementation on MULTOS cards was a lot slower.
- Although the study could not include a deep assessment of side channel attack resistance, there appears to be plenty of performance overhead available to allow for software protection measures to be included, if the platform itself does not give enough protection. Some brief notes are included about timing attacks, and it seems that a careful native code implementation should be able to protect against timing attacks without too much trouble (side channel attacks are explored a bit further in clause 9).

An extension to the above report was also produced by the same author. The extension report [10] is again included as an accompanying document to this one.The extension report:

- Confirms that, using older, more resource limited smart card platforms than were used in the study for [9], it is still possible to meet the Tuak speed and performance goals in native code, without the need for a crypto coprocessor.
- Describes initial investigations into possible side channel data leakage from an implementation with no software protection.
- Indicates (tentatively) that, with the possible exception of cards using certain chips that claim innovative hardware defences against side-channel and fault attacks, it is likely that smart card/chips will require some protection against side channel data leakage in the software implementation of Tuak.
- Confirms that there should be a performance overhead available to accommodate such software protection measures, with an efficient implementation of Tuak.

10 More notes on implementation and side channel attacks

10.1 Protecting implementations against side channel attacks

Some consideration of side channel attacks was given in [9] and [10] – see clause 9.2.

Research results also exist concerning how Keccak implementations can be protected against side channel attacks, and these results are directly useful for Tuak implementations too.

[11] considers power attacks – Simple Power Analysis (SPA) and Differential Power Analysis (DPA), including higher order DPA – and electromagnetic radiation attacks – Simple Electromagnetic Analysis (SEMA) and Differential Electromagnetic Analysis (DEMA), including higher order DEMA. Although the paper isn't exclusively about Keccak, Keccak is given particular attention, and the indications are that protecting an implementation, particularly in software, can be reasonably straightforward and efficient.

[12] gives further consideration to power analysis, and shows how implementations of Keccak in either hardware or software can be efficiently protected.

It was already noted in clause 9.2 that protection against "standard" timing attacks seems fairly straightforward. Keccak implementations do not naturally make use of large table lookups, so cache timing attacks should not be a concern.

10.2 Software implementation and the NIST SHA-3 standard

When the Tuak design was finalized, NIST had selected Keccak as the winner of the SHA-3 competition, but the SHA-3 standard had not been published. SAGE liaised closely with the Keccak design team, to understand what the forthcoming NIST standard was likely to include, and to align the Tuak specification with the anticipated SHA-3 standard as far as possible.

At the time of writing this design and evaluation report, NIST has now published a *draft* SHA-3 standard, together with a request for comments. If this draft standard remains unchanged, then the alignment of Tuak will be very good. In particular, as well as drop-in replacements for the current SHA-2 hash family, the draft NIST standard includes two "Extendable-Output Functions" SHAKE-128 and SHAKE-256, which allow for variable-size outputs. All of the padding and formatting lines up perfectly for us, so that the Tuak functions can all be defined very straightforwardly in terms of SHAKE-256; thus a Tuak implementation could directly and quickly be built from a SHAKE-256 implementation.

11 Conclusions

Keccak was selected as the winner of the SHA-3 computation after several years of intense scrutiny by the world's cryptographic community. It is widely agreed by this expert community to have a very clear and well thought out security design, and a very high margin of security. The Tuak design makes direct and straightforward use of Keccak, and clearly inherits Keccak's security benefits. Tuak uses a very strong choice of parameters from the Keccak function family, in line with what is now expected to be standardized by NIST. SAGE consulted with the Keccak design team several times, including sharing its proposed final design with them, giving extra assurance that Tuak (a) in no way fail to capture the security benefits of Keccak and (b) aligns with likely SHA-3-related implementations of Keccak as far as possible. Independent security assessment reinforces this position.

Implementation and performance aspects (including side channel attack resistance) seem comfortable too, based on independent assessment of Tuak and published research on Keccak.

In summary, SAGE believes that Tuak meets its targets of security, performance, and resilience as a complement to MILENAGE.

Annex A:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Nov 2014					First TR version		0.1.0
Dec 2014	SA#66	SP-140816			Version for information and approval	0.1.0	1.0.0
					Version after approval	1.0.0	12.0.0

History

Document history		
V12.0.0	January 2015	Publication