3GPP TR 35.937 V19.0.0 (2025-01)

Technical Report

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 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 4: summary and results of design and evaluation (Release 19)

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

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.

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 the summary and results of the design and evaluation of 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**.

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 [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 3: Implementors' Test and Design Conformance Test Data".
- **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".**

1 Scope

The present document contains a detailed summary of the work performed during the design and evaluation of MILENAGE-256 algorithm set. It contains results and findings from this work and should be read as a supplement to the specifications of the algorithms [3] and the general project report [2].

The example set is based on the block cipher Rijndael-256-256 with 256-bit key and block size [8, 24] (recall that the 128-bit Advanced Encryption Standard, AES-256, corresponds to Rijndael-128-256 [9], where the notation Rijndael-b-n is defined in clause 3.2 below).

An optional-to-use function, ***f5***** with the aim of countering certain replay attacks that can lead to traceability of subscribers [20], was designed according to candidate solutions discussed in 3GPP TR 33.846 [12]. When used, the optional function ***f5***** replaces ***f5****.

The specification and associated test data for the example algorithm set is documented in two documents:

- A formal specification of the mode and the example kernel [3]
- A test data document, covering mode and the example kernel [4]

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 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".

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

[6] 3GPP TS 33.105: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements".

[7] 3GPP TS 33.501: "Security architecture and procedures for 5G system".

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

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

[10] 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".

[11] 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".

[12] 3GPP TR 33.846, "Study on authentication enhancements in the 5G System (5GS)".

[13] ETSI TR 133 909, "Report on the design and evaluation of the MILENAGE algorithm set; Deliverable 5: An example algorithm for the 3GPP authentication and key generation functions", 2001.

[14] J. Daemen, and V. Rijmen, email correspondence with the SAGE AF TF, Dec 2019.

[15] J. Balasch et al., "Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices", https://eprint.iacr.org/2012/507.pdf.

[16] Z. Bao, J. Guo, T. Iwata, L. Song, "SIV-Rijndael256 Authenticated Encryption and Hash Family", submission to NIST, csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/SIV-Rijndael256-Spec.pdf

[17] M. Bellare, J. Killian, and P. Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code", Journal of Computer and System Sciences 61(2000), 362-399.

[18] R. Bhaumik, M. Nandi, and A. Raychaudhuri, "Improved indifferentiability security proof for 3-round tweakable Luby–Rackoff", Designs, Codes and Cryptogryptography 89(2021), 2255–2281.

[19] A. Biryukov A and D. Khovratovich, "Related-Key Attack on the Full AES-192 and AES-256", in S. Halevi (ed), Proceedings of ASIACRYPT'09, LNCS 5912, Springer Verlag, pp. 1–18.

[20] R. Borgaonkar, "New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols", in Proceedings on Privacy Enhancing Technologies 2019(3):108-127.Also available at https://eprint.iacr.org/2018/1175.pdf (published online: July 2019).

[21] M. Brisfors, S. Forsmark, and E. Dubrova, "How Deep Learning Helps Compromising USIM", in P.-Y. Liardet, N. Mentens (Eds): Proceedings of the 19th Smart Card Research and Advanced Application Conference (CARDIS'2020), LNCS 12609, Springer Verlag, pp. 135-150.

[22] J-Sébastien Coron, Y. Dodis, A. Mandal, and Y. Seurin. "A Domain Extender for the Ideal Cipher", in Proceedings of Theory of Cryptography (TCC) 2010, LNCS 5978, Springer-Verlag, pp. 273–289.

[23] J-S. Coron, T. Holenstein, J. Patarin, Y. Seurin, and S. Tessaro, "How to Build an Ideal Cipher: The Indifferentiability of the Feistel Construction", J. Cryptology (2016) 29:61–114.

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

[25] I. Damgård, "A Design Principle for Hash Functions", in proceedings of CRYPTO '89, LNCS Vol. 435, Springer-Verlag, 1989, pp. 416-427.

[26] P-A. Fouque, C. Onete, and B. Richard, "Achieving Better Privacy for the 3GPP AKA Protocol", Proceedings on Privacy Enhancing Technologies ; 2016 (4):255–275.

[27] H. Gilbert: "The Security of One-Block-to-Many Modes of Operation", in T. Johansson (Ed): Proceedings of FSE 2003, LNCS 2887, Springer Verlag, 376-395.

[28] L. Goubin and J.-S. Coron, "On boolean and arithmetic masking against differential power analysis", in Ç. K. Koç, C. Paar (Eds): Proceedings of CHES '00, LNCS 1965, Springer Verlag, pp. 231-237.

[29] L. Goubin and J. Patarin, "DES and differential power analysis", in CHES´'99, LNCS 1717, Springer Verlag, pp. 158-172.

[30] R. Guanciale, H. Nemati, C. Baumann, and M. Dam, "Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures", in Proceedings of 2016 IEEE Symposium on Security and Privacy, SP 2016, pp. 38-55.

[31] S. Gueron, White Paper "Intel Advanced Encryption Standard (AES) New Instructions Set", https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[32] S. Gueron and Y. Lindell, "GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte", in proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS), 2015, pp. 109–119.

[33] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side Channel Cryptanalysis of Product Ciphers", in Procedings of ESORICS'98, LNCS 1485, Springer Verlag, pp. 97-110.

[34] P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in N. Koblitz (Ed): Proceedings of CRYPTO'96, LNCS 1109, Springer Verlag, pp. 104-113.

[35] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", in M. Wiener (Ed), Proceedings of CRYPTO '99, LNCS 1666, Springer-Verlag, pp. 388-397.

[36] M. Liskov, R. L. Rivest, and D. Wagner, "Tweakable Block Ciphers", in M. Yung (Ed.): Proceedings of CRYPTO 2002, LNCS 2442, Springer Verlag, pp. 31–46, 2002.

[37] Y. Liu, Y. Shi, D. Gu, B. Dai, F. Zhao, W. Li, Z. Liu, and Z. Zeng, "Improved impossible differential cryptanalysis of large-block Rijndael", Science China Information Sciences, 62(3):32101, 2019.

[38] U. Maurer, R. Renner, and C. Holenstein, “Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology”, proceedings of Theory of Cryptography ’04, LNCS, vol 2951, Springer Verlag 2004, pp 21–39.

[39] A. Maximov and M. Näslund, "Security analysis of the Milenage-construction based on a PRF", Cryptology ePrint Archive, available at https://eprint.iacr.org/2023/607.

[40] R. C. Merkle, “Secrecy, authentication, and public key systems”, Stanford Ph.D. thesis 1979.

[41] R. C. Merkle, "A Certified Digital Signature", in proceedings CRYPTO '89 Proceedings, LNCS Vol. 435, 1989, pp. 218-238.

[42] T. S. Messerges, "Securing the AES finalists against Power Analysis Attacks", in B. Schneier (Ed): Proceedings of the Seventh Fast Software Encryption Workshop (FSE ’00), LNCS 1978, Springer Verlag, pp. 150-164.

[43] J. Patarin, "Luby-Rackoff: 7 Rounds Are Enough for 2n(1−ε) Security", in D. Boneh (Ed.): Proceedings of CRYPTO 2003, LNCS 2729, Springer Verlag, pp 513–529.

[44] V. Ulitzsch and J.-P. Seifert, "Breaking the quadratic barrier: Quantum cryptanalysis of Milenage telecommunications’ cryptographic backbone", available at https://eprint.iacr.org/2022/733.

[45] R. Wang, H. Wang, E. Dubrova, and M. Brisfors, "Advanced Far Field EM Side-Channel Attack on AES", in Proceedings of the 7th ACM on Cyber-Physical System Security Workshop (CPSS '21), pp. 29-39.

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms 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].

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

SQN_{HE}: Local value of **SQN** as available in the HE

SQN_{MS}: Local value of **SQN** as available at the MS

XMAC-A: Expected value of **MAC-A**

XMAC-S: Expected value of **MAC-S**

XRES: Expected Response to Challenge

Additional terminology

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****

IN_0, IN_1, IN_2, IN_3, IN_4, IN_5, IN_6, IN_7: 256-bit instance-specific input values constructed within the computation of the functions **f1, f1*, f2, f3, f4, f5, f5***, and **f5****

K_{SZ}: The size in bytes of the subscriber key **K**

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****

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 [5]. Additional explanation of the usage of boldface, italics, etc within MILENAGE-256 appears in the MILENAGE-256 Algorithm Specification [3].

3.2 Symbols

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

=	The assignment operator
==	The equality comparison operator, returns True or False
:=	The definition operator
$\| x \|$	The bit-length of x
$\mathbb{N}_n$	The set of natural numbers representable by n bits
$\{\mathbb{N}_n\}^b$	The set of arrays of length b containing natural numbers, each representable by n bits
{ }	Curly brackets denote arrays, with indices starting from 0, e.g. $X = \{X[0], X[1], \ldots, X[b\text{-}1]\}$
‖	The concatenation of two operands. For arrays X and Y of lengths a and b, respectively X ‖Y := {X[0], …, X[a-1], Y[0], …, Y[b-1]}; For integers of n-bit A and m-bit B $A\|\|B := A \cdot 2^m + B$. For arrays, the concatenation symbol is also used when an array of length $2b$ is partitioned into a representation as two arrays of length b, see below.
$\mathbf{0}_b$	The byte array $\{\mathbb{N}_8\}^b$ of length b consisting of all zeroes, i.e. $\mathbf{0}_b = \{0, 0, \ldots, 0\}$
$\mathbf{1}_b$	The byte array $\{\mathbb{N}_8\}^b$ of length b consisting of all zeroes, except for the first byte with the value 1, i.e. $\mathbf{1}_b = \{1, 0, \ldots, 0\}$
$\mathbf{2}_b$	The byte array $\{\mathbb{N}_8\}^b$ of length b consisting of all zeroes, except for the first byte with the value 1, i.e. $\mathbf{2}_b = \{2, 0, \ldots, 0\}$
$X = X^0 \| \| X^1$	An array of bytes $X \in \{\mathbb{N}_8\}^{2b}$ of length $2b$ is represented as two arrays of bytes $X^0 \in \{\mathbb{N}_8\}^b$ and $X^1 \in \{\mathbb{N}_8\}^b$, each of length b, defined as: $X^0 := \{X[0], X[1], \ldots, X[b-1]\}$, $X^1 := \{X[b], X[b+1], \ldots, X[2b-1]\}$
$(v)_3$	The first (i.e. most significant) 3 bits of the integer v. If $v < 2^3$, then $(v)_3$ simply denotes the three-bit representation of v EXAMPLE: $(0)_3 = 000$, $(1)_3 = 001$, $(15)_3 = 111$
$\oplus$	The bitwise exclusive-OR operation. For byte arrays, this operates byte-wise
$x <<< r$	Circular rotation to the left of the binary representation of x by r bits
byte(a, b, …)	A function that encodes integers a, b, … smaller than 8 bits into an array $\{\mathbb{N}_8\}^1$ containing a single byte. The exact definition is not relevant for the present document, except for the fact that over the domain of definition, the **byte**() encoding function is one-to-one. Details can be found in the specification [3]
$F(x \in \mathcal{A}) \rightarrow \mathcal{B}$	Denotes that a function F maps elements from a domain $\mathcal{A}$ to elements of a domain $\mathcal{B}$
AES-n	AES with n-bit key
$E_K(X)$	Encryption of X under the key K
MD^E	Message Digest based on block cipher E
$PRF_{\mathbf{K}}$	Pseudo-random function defined by key **K**
Rijndael-b-n	The Rijndael block cipher with b-bit blocks and n-bit key

3.3 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].

3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
ASCII	American Standard Code for Information Interchange
AV	Authentication Vector
DPA	Differential Power Analysis
eSIM	Embedded SIM
HE	Home Environment
MAC	Message Authentication Code
MD	Message Digest
ME	Mobile Equipment

MS	Mobile Station
RKA	Related Key Attack
SPA	Simple Power Analysis
TA	Timing Attack
UE	User Equipment
UICC	Universal Integrated Circuit Card
USIM	Universal Subscriber Identity Module

NOTE: MS is a legacy term from GPRS specifications which in later specifications is usually replaced by the term UE (or ME, if omitting the USIM-part). Nevertheless, the base specification of the AKA framework [5] still uses the term MS for certain AKA-specific purposes.

4 Structure

- Clause 5 provides background information to the design work of the example set for 3GPP Authentication and Key Agreement Functions.
- Clause 6 provides a summary of the algorithm requirements.
- Clause 7 describes the design criteria used for the work.
- Clause 8 consists of a brief presentation of the actual framework design.
- Clause 9 provides some rationale on the chosen design.
- Clause 10 gives an overview of the evaluation work carried out.
- Clause 11 contains the conclusions from the work.

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 cellular mobile systems. Within the mobile communication systems specified by 3GPP, there is a need to provide security features. These security features have gradually increased in sophistication and security level from the 3rd generation (UMTS) networks up to the current 5th generation (5G) [7], and are realised with the use of cryptographic functions and algorithms. One such set of algorithms is the authentication and key generation (AKA) algorithms. While ciphering and data integrity algorithms are fully standardised, the AKA algorithms can be selected by each operator individually. Since the overall security rests on the cryptographic strength of the AKA algorithms, these are required to provide strong security assurances. Standardised, but operator-customisable AKA algorithms, based on state-of-the-art cryptographic functions that have undergone wider analysis by cryptographic experts, are therefore desired. An operator's customisations should follow simple guiding rules so that the risk of insecurity due to unfortunate customisation choices is virtually non-existent.

A first new design criterion in the 5G context is that the security algorithms, in particular the AKA algorithms, need to be able to provide a 256-bit security level. This is motivated by the increased criticality of the 5G services and also by the possible evolution of quantum computers within the economic lifetime of 5G. Support for 256-bit keys is already provided by TUAK [11]. Therefore, the new design is based on a cryptographic core which substantially differed from TUAK, so that future advances in cryptanalysis are unlikely to affect both TUAK and the new design.

In UMTS and LTE, the AKA algorithms consisted of eight cryptographic functions ***f0*** – ***f5***, **f1*** and ***f5****, each serving different purposes, as described in more detail in clause 5. For 5G, an additional function ***f5**** has been defined as also described in clause 5.

The major design goal was to design a framework for the authentication and key generation functions that is secure and flexible. This goal was achieved through a well-analysed construction using a 256-bit pseudo-random function as the kernel function and including additional configurable parameters selectable by the operator.

MILENAGE-256 algorithm set was designed based on the Rijndael block cipher with 256-bit blocks. Since MILENAGE-256 is a straight-forward generalisation of the previous 128-bit MILENAGE, the security analysis is relatively similar to existing analyses, in particular the security proof [27].

6 Summary of algorithm requirements

6.1 General requirements for 5G cryptographic AKA functions and algorithms

The main new requirements, i.e. increasing the security level to 256-bits, support for various parameter sizes, etc. Ensuring adequate complexity of both exhaustive key search and other generic attacks against 256-bit crypto primitives was the main target.

Additionally, an optional-to-use function ***f5***** was defined to counteract discovered subscriber-tracing attack [20].

MILENAGE-256 is based on cryptographic principles that are already available in the public domain, to minimise issues related to export control.

6.2 The authentication and key agreement functions

The mechanism for authentication and key agreement (AKA) [5, 6] requires the following cryptographic functions:

f0 the random challenge generating function

f1 the network authentication function

f1* the re-synchronisation 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 the re-synchronisation message

Additionally, the present document defines:

f5** an alternative to f5* which provides additional protection against subscriber tracing attacks via linkability of AKA protocol executions.

The implementation of the random challenge generation function, ***f0***, is completely determined by the operator.

According to requirements, all functions ***f1-f5, f1*, f5**** and ***f5***** have been defined to support at least 128 and 256-bit length values for the subscriber key K and the RAND-values.

NOTE 1: All eight functions ***f1-f5, f1*, f5**** and ***f5***** depend also on the input values OP_c and *ALGONAME*, as well as encodings of parameter sizes and operator selectable customisation constants. However, for notational simplicity these are not shown below. Details can be found elsewhere [3], and, where relevant for the security analysis, in various clauses of the present report.

NOTE 2: The functions ***f1**** and ***f5*/f5***** need not be computed unless a synchronisation error is detected at the UE during an AKA procedure. Conversely, if a synchronisation error occurs, the functions ***f2, f3*** and ***f4*** need not be computed.

6.3 Implementation and operational considerations

The existing requirements on performance [6, § 5.1.5] pertain to the five functions ***f1-f5***. All these five outputs need to be produced within 500msec on an IC card running with 3MHz clock. The same timing bound is required for eSIM.

It is further required [6] that an implementation can be done with 8KByte and 300Byte RAM.

6.4 Requirements on the functions *f1-f5, f1*, f5*,* and *f5***

6.4.1 *f1*

f1: the network authentication function

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

The function **f1** is defined as a MAC function and is designed to produce output lengths from at least the set {32, 64, 128, 256} bits.

6.4.2 *f1**

f1*: the re-synchronisation message authentication function

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

The requirements for ***f1*** also apply to ***f1****.

6.4.3 *f2*

f2: the user authentication function

f2: (**K**; **RAND**) → **RES** (or **XRES**)

The requirements for ***f1*** also apply to ***f2.*** The function ***f2*** is designed to produce output lengths from at least the set {32, 64, 128, 256}.

6.4.4 *f3*

f3: the ciphering key generation function

f3: (**K**; **RAND**) → **CK**

The requirements for ***f1*** also apply to ***f3***. The function ***f3*** is designed to produce output lengths from at least the set {128, 256}.

6.4.5 *f4*

f4: the integrity key generation function

f4: (**K**; **RAND**) → **IK**

The requirements for ***f3*** also apply to ***f4***.

6.4.6 *f5*

f5: the anonymity key generation function

f5: (**K**; **RAND**) → **AK**

The requirements for ***f1*** also apply to ***f5***. The function ***f5*** is designed to produce output lengths from at least the set {48, 64, 80, 96}. The use of ***f5*** is optional in the standard.

6.4.7 *f5**

f5*: anonymity key generation function for re-synchronisation

f5*: (**K**; **RAND**) → **AK***

The requirements for ***f5*** also apply to ***f5**** The use of ***f5**** is optional in the standard.

6.4.8 *f5***

f5**: anonymity key generation function for re-synchronisation with linkability protection

f5**: (**K**; **RAND**, **MAC-S**) → **AK***

The requirements for ***f5*** output length also apply to ***f5*****. The use of ***f5***** is optional in the standard.

NOTE: When f5** is used, it replaces the use of f5*.

7 Design criteria

Based on the requirements and given starting points, the following essential design criteria for MILENAGE-256 were established.

7.1 Cryptographic design criteria

1) Without knowledge of secret keys, the functions f1, f1*, f2, f3, f4, f5, f5* and f5** are to be practically indistinguishable from independent random functions of their inputs, i.e. RAND, and, for f1, f1*, and f5**, one or more of SQN, AMF, MAC-S.

2) It is to be practically infeasible to determine any part of the secret key K, or the operator customisation values (e.g. OP), by manipulation of the inputs and examination of the outputs to the algorithm.

3) Events tending to violate criteria 1 and 2 are to be regarded as insignificant if they; a) occur with probability approximately 2-256 or less; b) require approximately 2256 operations or more.

NOTE: The algorithm set is designed with resistance against quantum computing-based attacks in mind. However, the quantitative ramifications of requirements 2 and 3 could depend on the precise definition of the quantum-empowered attacker model. This is elaborated in clause 10.5.

4) Events tending to violate criteria 1 and 2 are to be examined if they occur with probability approximately 2-128 (or require approximately 2128 operations) to ensure they do not have serious consequences. Serious consequences would include recovery of the secret key or ability to emulate or predict the algorithm outputs on future inputs.

5) The design is to be built upon well-known structures, and avoiding unnecessary complexity. This simplifies analysis and avoids the need for a formal external evaluation.

6) It is strongly preferred that the security analysis can be supported by a formal security proof covering the entire design or critical properties thereof.

7) Simple (hard-to-get-wrong) guidelines for how to securely perform operator customisation of the algorithms are to be defined.

8) As specified in clause 6.2, the MILENAGE-256 algorithms need to be able to accept input parameters of different sizes and also produce output parameters of different sizes, with this flexibility supported without introducing weaknesses, beyond those inherent to the selected parameter sizes.

9) For a specific instantiation of MILENAGE-256, with a fixed set of input/output parameter sizes, the instance produces, with high probability, outputs that are distinct from those produced by another instance, defined by other choices for the parameter sizes.

EXAMPLE: An instance of ***f2***, configured to produce 64-bit output **RES**-values, is designed to, with high probability, produce **RES**-outputs which are distinct from proper prefixes of outputs produced by another instance, configured to produced 128-bit **RES**-values, even if the subscriber key **K** and **RAND**-inputs are the same.

7.2 Implementation criteria

In addition to the performance requirements listed in clause 6.3, it was ensured that the listed requirements would be met even after implementation of protection mechanisms against side channel attacks like differential power analysis (DPA).

Well-studied DPA-countermeasures of AES implementations can be reused. Since the Rijndael kernel of MILENAGE-256 is structurally identical to AES, available countermeasures for AES can also be deployed in this case.

It is recommended that, as a general principle, a specific implementation of MILENAGE-256 will only support a given set of parameter sizes among those stated in clause 6.2. Exceptions from this principle could be motivated for certain parameters, e.g. the size of the subscriber key **K** and/or the size of **RAND** as part of a migration strategy towards increased security levels.

EXAMPLE: An implementation could initially be deployed with 128-bit **K** and then later upgraded to 256-bit **K**.

7.3 Need for algorithm-set customisation

To retain operator customisation options similar to those available for 128-bit MILENAGE, the feature of an Operator Variant Algorithm Configuration Parameter, OP, is maintained, in order to:

1) Make each operator's implementation different.
2) Prevent USIMs for operators being interchangeable, either through trivial modification of inputs and outputs or by reprogramming of a blank USIM.
3) Allow some algorithm details to be kept secret. And,
4) Provide some additional protection against a poorly chosen kernel.

Additionally, the feature of the per ***f***-function operator selectable customisation constants has been maintained, but the mechanism has been simplified by making bad choices virtually impossible.

NOTE: Those constants, at the same time, are used to provide instance separation between instantiations with different parameter sizes as described in clause 7.1, item 9.

Finally, an input *ALGONAME* parameter has been inherited from the TUAK algorithm set [11]. This parameter can be used to provide cryptographic separation between usage of MILENAGE-256 within 3GPP, and usage in other contexts/standardisation bodies.

7.4 Criteria for the cryptographic kernel

7.4.1 General

The cryptographic kernel function, a key-dependent pseudo-random function denoted PRFK, is used by MILENAGE-256 ("the framework") to produce a set of 256-bit output blocks, derived from the input parameters. Each of the ***f1 – f5***, ***f1**** or ***f5*/f5***** outputs are derived from these blocks (truncating the blocks, if fewer than 256 output bits are desired). These output values are produced under the control of a 128 or 256-bit subscriber specific key **K**. It needs to be observed that **K** is a long-term secret which must be duly protected under all circumstances.

NOTE: The instantiation of PRF_K [3] is based on the use of a block cipher, i.e. a pseudo-random permutation function. The resulting PRF is also a one-to-one permutation. It would be appropriate to use the more accurate notation PRPK (for Pseudo-Random Permutation). However, since the framework allows to instantiate the kernel as a non-one-to-one function, a choice has been made to use the more general notation PRFK throughout the present specification.

7.4.2 Implementation and operational considerations

Since (under normal conditions) six applications of PRF_K within the framework are required to produce the five outputs ***f1 – f5*** (which are the set of values required under normal, error- free operation), this leads to the need for PRF_K to be

computable within about 80msec to meet the overall 500msec bound. This in turn corresponds to six evaluations of Rijndael-256-256 (plus possibly one execution of the Rijndael key schedule).

7.4.3 Functional and cryptographic requirements

The purpose of the kernel PRF_K is to map an input value X to an output value Y under the control of the key **K**, $Y = PRF_{\mathbf{K}}(X)$. The cryptographic requirements on the MILENAGE-256 framework, as described in clause 7.1, can be projected onto security requirements of the kernel. It needs to be (computationally) infeasible to derive **K** if a large number of pairs (X, Y) are known for a fixed **K**. The same needs to hold if the X-values are chosen by an attacker.The latter *chosen input attack* should be infeasible even if the attacker has access to side channel information, e.g. power consumption or execution timings of an IC card which holds an implementation of the kernel function [21, 28-30, 33-35, 42, 45].

Furthermore, given X (but not **K**) it needs to be infeasible both to compute Y and to distinguish given Y-values from a random bit string of the same length, even if a large number of (X, Y)-pairs, produced using the same **K**, is known. In one-to-one function (or permutation), "large" implies values on the order of 2^{128} for a 256-bit output $PRF_{\mathbf{K}}$.

7.4.4 Types and parameters of the kernel

A one-to-one PRF_K can be instantiated directly by a symmetric block cipher with a block size of 256 bits and this is the choice made for MILENAGE-256.

Both the key and the input/output blocks are unstructured data from the kernel function's view, though the MILENAGE-256 framework constructs/formats these in a specific way from the available parameters.

8 The Milenage-256 framework

8.1 General

The following diagram shows the MILENAGE-256 framework for the functions ***f1, f1*, f2, f3, f4, f5, f5**** and **f5**** using the kernel function denoted $PRF_{\mathbf{K}}$.

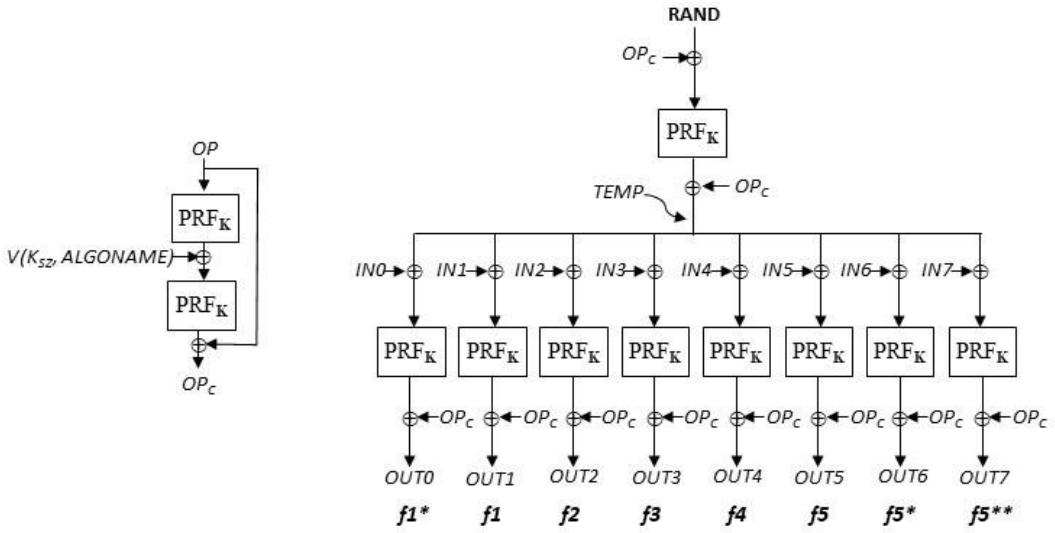

Figure 8.1-1: MILENAGE-256 framework. Use of the functions *f5 and *f5*** is mutually exclusive. When used, precisely one of them is configured for use within an AKA protocol**

The value OP_C is derived from the subscriber key K and the operator dependent value OP by

$$OP_C := OP \oplus \mathrm{PRF}_{\mathbf{K}}(\mathrm{PRF}_{\mathbf{K}}(OP) \oplus V(K_{SZ}, ALGONAME)) \quad \text{(EQ 1)}$$

Here, the function *V* formats K_{SZ} (the size of the subscriber key) and an encoding of the algorithm name into a 256-bit input to the PRF [3].

As with 128-bit MILENAGE [10], it is recommended that OP_C is calculated outside the USIM cards and then stored in each card as an individual value. This provides better protection for *OP*, relative to the alternative choice of storing *OP* in every card.

The output values can then be expressed as

$$TEMP = \mathrm{PRF}_{\mathbf{K}}(\mathbf{RAND} \oplus OP_C) \oplus OP_C \quad \text{(EQ 2)}$$

$$OUT_i = \mathrm{PRF}_{\mathbf{K}}(TEMP \oplus IN_i) \oplus OP_C, \quad \text{(EQ 3)}$$

where *IN*i contains the operator-selectable value *c*i and a unique coding of the input parameters and the sizes of the input/output parameter(s) of the corresponding f-function. The details of the *IN*i format can be found in the Algorithm Specification [3].

The encodings of *IN*i, as well as the inclusion of *V*( ) in the *OP*-derivation, serve to satisfy a cryptographic instance separation requirement: it would be highly unlikely that algorithm instances used in different contexts, or instances using/producing parameters of different sizes, result in the same (or otherwise correlated) outputs.

The changes relative to the MILENAGE-128 algorithm set [10] can be summarized as follows. More details and rationale are provided in clause 9.1.2.

Removal of rotations: In MILENAGE, operator customisation of the separation between distinct f-functions could be achieved by selecting customised rotations, as well as additive offsets, of the inputs to the kernel. MILENAGE-256 only retains customisation of the ***f-*** function separation via the additive offsets (denoted *c0* to *c7* above).

MAC input format: The MAC input format used in MILENAGE [10] added a parity requirement when selecting the additive customisation offsets.

MAC outputs: In MILENAGE, the MAC values for network authentication and re-synch procedure verification were each taken as half of one kernel output.

Restructuring of f2, f5 and f5* outputs: In MILENAGE, ***f2*** and ***f5*** were taken from a single output of the kernel, whereas **f5*** was taken from a separate invocation of the kernel.

8.2 The MILENAGE-256 kernel

The kernel PRFK of Milenage-256 is the block cipher Rijndael [8, 14] with 256-bit blocks and 128 or 256-bit keys. An existing security proof holds under the assumption that Rijndael- 256-256 behaves as a pseudo-random permutation [27].

9 Rationale for the chosen design

9.1 The MILENAGE-256 framework

9.1.1 Features in common with 128-bit MILENAGE

9.1.1.1 Use of *OP*

The feature of operator selectable values *OP* and OP_C (the latter also subscriber-dependent) is, like in 128-bit MILENAGE, achieved by a non-invertible mapping from *OP* to OP_C, using the PRF kernel. This makes reverse engineering of *OP* from OP_C infeasible. For MILENAGE- 256, the exact details differ slightly from MILENAGE; see clause 9.1.2.

9.1.1.2 Mode of operation

MILENAGE-256, like its predecessor, runs in a mode of operation similar to counter mode to compute ***f2-f5***, ***f5****: an intermediate value is first created by applying the cryptographic kernel to the **RAND**-value. This is done to prevent distinguishing attacks that would be possible if an attacker could choose inputs [27]. A set of distinct input blocks are created by offsetting the intermediate value, and these are then fed to the cryptographic kernel and the outputs of the ***f-*** functions are taken from the corresponding output blocks. A security proof is also provided for the case of a PRP kernel [18], while the proof for a PRF (non-permutation) appears in a separate work [39]. The functions ***f1*** and ***f1**** are created in a similar way, the main difference being that the offsets of the intermediate value (*IN*) is created also with dependence on **SQN** and **AMF**. The security proof [39] also covers ***f1*** (or ***f1****).

The new function f5** whose input also differs structurally from the inputs of ***f2-f5*** and ***f5**** is analysed in clause 10.3.4.

9.1.2 Differences to 128-bit MILENAGE

The following table summarises differences between MILENAGE and MILENAGE-256.

Table 9.1.2-1: Differences between MILENAGE and MILENAGE-256

Feature	MILENAGE	MILENAGE-256	Rationale
Key-size	128	128, 256	Security requirement and backward compatibility/ migration aspects
Kernel block size	128-bit	256-bit	Consequence of requirement on security-level
Basis for kernel	AES-128	Rijndael- 256-256	See clauses 9.2 and 9.3
OP_Ccalculation	-	Different structure and parameters	See clause 9.1.2.1
Operator- selectable constants	Rotations and additive offsets	Offsets only, and only part of the constants freely selectable	See clause 9.1.2.2
Format of input to ***f1/f1****	(**SQN** || **AMF**) repeated twice	(**SQN** || **AMF**) occurs once	See clause 9.1.2.3
Invocation of kernel operations	***f1, f1**** both obtained from a single kernel (input, output)- pair, and similarly for ***f2/f5****	All four of ***f1, f1*, f2, f5**** obtained from distinct kernel (input, output)-pairs	See clause 9.1.2.4
f5** function	Not defined	Defined, optional	Support for additional security requirement, see clauses 6.2 and 10.3.4

9.1.2.1 OP_C calculation

The OP_C calculation of MILENAGE-256 differs from that of MILENAGE in that two sequential applications of the kernel are made, rather than just one. This is done in order to allow OP_C being created with a dependence on the parameters ALGONAME (an encoding of context information in which MILENAGE-256 is being used) and also the key-size, K_{SZ} of the long-term subscriber key. Both of these dependencies serve to create instance separation, and the double application of the kernel is more likely to strengthen the construction than weaken it.

9.1.2.1 Operator-selectable customisation constants

The previous 128-bit MILENAGE algorithm set allowed the operator to choose two sets of operational constants, affecting the values fed to the second layer of the PRF-kernel.

Specifically, for each input to the second layer PRFs, the values were formed as

$$input_i = (TEMP <<< r_i) \oplus c_i, \quad i = 2, 3, 4, 5 \quad \text{(EQ 4)}$$

where *TEMP* corresponds the output of the first application of the PRF, corresponding to (EQ 2), c_i is an n-bit (n = 128) offset constant and r_i is a rotation constant taking integer values in the range [0..127]. Both c_i and r_i could be selected freely by the operator. However, not all pairs of (c_i, r_i)-values lead to an optimally secure implementation; a simple condition such as $c_i \neq c_j$ and $r_i \neq r_j$ if $i \neq j$ does not automatically imply maximum security since even under this condition, it could happen that for a significant number of values of *TEMP*, $(TEMP \oplus c_i) <<< r_i = (TEMP \oplus c_j) <<< r_j$ which could reduce security. The simplest way to ensure optimum security would be to require $c_i \neq c_j$ for $i \neq j$ and setting all $r_i = 0$. However, this defeats the purpose of allowing operators to select all the available (secure) values.

A second issue was that to ensure distinctness of the input to ***f1*** from the inputs of ***f2 - f5***, it was necessary that *ci* has even parity, while *c2…c5* were to have odd parity; see also the next clause.

In MILENAGE-256, the rotation values *ri* have therefore been removed. Further, the values *IN0, IN1, …, IN7* (see the Algorithm Specification document for details [3]) now comprise two parts: one pre-determined part providing instance separation of the different f-functions (including also separation of functions at different parameter sizes) and one part dependent on ci which can be freely selected by the operator, without any risk of causing insecurity, even if all the operator-selectable values are identical. This also removes the requirements on parity of the ci values. Observe also that keeping a rotation amount secret only adds 8 bits of entropy for an attacker.

This approach provides sufficient configurability while at the same time mitigating all risks of unfortunate choices.

9.1.2.3 Input to ***f1/f1**** vs ***f2-f5/f5****

Previously, the input to the second layer PRF of 128-bit MILENAGE when computing ***f1*** (and similarly for ***f1****) was formed as

$$input_1 = ((OP_C \oplus (\mathbf{SQN} \,||\, \mathbf{AMF} \,||\, \mathbf{SQN} \,||\, \mathbf{AMF})) <<< r_1) \oplus TEMP \oplus c_1,$$

where *TEMP* is the intermediate value, as produced by the first application of the PRF as in (EQ 2) above. For 128-bit MILENAGE purposes, since **SQN** || **AMF** || **SQN** || **AMF** always had even parity, with c_1 also having even parity while all $c_2 \ldots c_5$ had odd parity, it made it impossible for $input_1$ to be identical to any of $input_2 \ldots input_5$. However, with the simpler instance-separation mechanism of MILENAGE-256, with the definition of the values IN_0, IN_1,…, leading to ***f1***/***f1****-input, of the form

$$IN_1 = [\mathbf{byte}((1)_3, \mathrm{RAND}_{sz}, K_{sz}) \,||\, \mathbf{byte}(\mathrm{SQN}_{sz}, \mathrm{MAC}_{sz}) \,||\, \mathbf{AMF} \,||\, \mathbf{SQN} \,||\, \mathbf{0}_{12\text{-}SQN_{sz}} \,||\, c_1] \oplus TEMP,$$

the distinctness of the inputs relative to other ***f***-functions is ensured by the first three bits $(1)_3$ (or, for ***f1**** the value $(0)_3$), without the need for a special parity requirement.

9.1.2.4 Invocation of kernel operations

In the previous 128-bit MILENAGE specification ***f1/f1**** were both obtained from a single kernel (input, output)-pair, and similarly for ***f2/f5***. The value ***f5**** was however taken from a separate invocation of the kernel. The rationale for this was to save one or two additional invocations of the kernel under normal operational conditions (when both of ***f2/f5*** typically needs to be computed but ***f5**** does not). It might appear possible to avoid one invocation of the kernel by taking ***f1**** and ***f5**** from the same block. However, this would not work since the computation of ***f5**** cannot depend on

$\mathbf{SQN_{MS}}$, whereas ***f1**** depends on $\mathbf{SQN_{MS}}$.Thus, five or six computations of the MILENAGE kernel were necessary in all cases.

NOTE: It would be possible to limit computations of ***f2***, ***f3*** and ***f4*** if a re-synchronization procedure is needed, leaving the total number at four.

With MILENAGE-256, it is necessary to support output parameters of sizes greater than 128 bits which means that it becomes more difficult to save kernel computations. The only parameters that it appears possible to extract from the same block are ***f5/f5**** or ***f5*/f5*****since each of these four values are unlikely to ever need to be 128-bits (or more) in size. However, extracting ***f5*/f5***** from one block is not feasible since their inputs are different. Taking all these aspects into account and noting the relatively small saving made possible by combining ***f5/f5****, a cleaner design was chosen, which also permits stronger instance separation between each pair of ***f***-functions. At the same time, the cost of computing ***f2*** can now in principle be saved if a re-synchronisation occurs. Thus, in most cases MILENAGE-256 requires six invocations of the PRF kernel.

The restructuring in MILENAGE-256, which always takes each ***f***-output from a distinct output of the kernel, in combination with the instance separation, means the only way that two distinct ***f***-functions could collide under a given **RAND**-input, is by chance.

9.2 Block ciphers vs hash functions as kernel

The MILENAGE-256 algorithms employ a kernel, denoted as $\mathrm{PRF}_{\mathbf{K}}$. The algorithm set was designed to permit plug-in replacements for this kernel. This replaceability allows operators to freely employ variant kernels, without adversely impacting the security of the algorithm set, provided the replacement kernel is a suitable keyed function employing a 256-bit key. Using a function with 256-bit block size according to MILENAGE-256 principles is recommended.

The qualitative security requirements on $\mathrm{PRF}_{\mathbf{K}}$ require that it is infeasible (or strongly believed to be infeasible) to distinguish the outputs of $\mathrm{PRF}_{\mathbf{K}}$ from the outputs of a randomly chosen function. The quantitative security requirements on $\mathrm{PRF}_{\mathbf{K}}$ require that the probability of distinguishing the output remains "small", even after observing on the order of 2^{128} (input, output) pairs. This latter constraint is the strongest requirement that can be satisfied if the kernel function is a one-to-one (permutation) mapping, such as a block cipher, in which case the PRF is actually a pseudo-random permutation (PRP). If the kernel function is not a permutation, stronger quantitative bounds might be possible *in theory*, allowing observation of a larger number of (input, output)-pairs.

9.3 Choice of Rijndael and AES

MILENAGE-256 employs the block cipher Rijndael-256-256, which has 256-bit inputs/outputs and uses a 256-bit key [8, 14] (in case of a 128-bit key, it is padded with zeroes to make a 256-bit key). Rijndael-256- 256 was chosen owing to the extensive body of cryptanalysis and research already undertaken on the Rijndael family of block ciphers. Moreover, the Rijndael block cipher can be efficiently implemented in software or hardware and is generally held as being available IPR- free. The Rijndael-128-{128,192,256} algorithms were also selected as the AES [9] and have been well studied in this context.

Due to the fact that the AES (and Rijndael) block cipher has undergone extensive analysis, no real cryptanalysis of Rijndael was performed, but rather the strength of the construction for deriving the ***f1*** to ***f5**** modes from a strong block cipher was analyzed. However, a survey of known attacks against AES/Rijndael was performed, including a verification of the cryptologic status of the 256-bit block version of Rijndael. A summary is provided in clause 10.4.1.

10 Evaluation

10.1 Security evaluation criteria

Due to the fact that Rijndael and AES have undergone extensive analyses during the AES selection process, the mathematical evaluation performed by the AF Task Force did not duplicate that work, though a summary of known attacks on AES with 256-bit keys, as well as on Rijndael with 256-bit blocks, is provided in clause 10.4.1.

Instead, the focus was on assessing the strength of the construction for deriving the functions ***f1/f1**** to ***f5/f5*/f5*** ** from a strong 256-bit key block cipher E. Similar to the case of MILENAGE, the main purpose of the mathematical evaluation is to check that the construction satisfies the two following requirements:

i. Under the assumption that the 256-bit block cipher E underlying the PRF is a "strong" (with meaning to be defined later) block cipher, there must be no attack of complexity substantially less than 2^{256} computations of E allowing one to recover any information on the value of the key **K** or to forge outputs of the algorithm for a large set of arbitrary **RAND** inputs, based on the knowledge of known values of the eight ***f***- function outputs corresponding to any chosen **RAND, SQN, AMF** inputs, even if the *OP* and c_i-constants are known.

ii. There must be no other attack enabling one to distinguish the eight ***f***-function generators from independent random functions of the inputs **RAND, SQN, AMF** with substantially less than 2^{128} queries, even if the *OP* and c_i-constants are known.

For that purpose, the mathematical evaluation needs to consider:

A. The strength of each of the ***f1/f1*-f5/f5*f5**** modes, considered individually.

B. The independence/separation between the eight ***f***-functions.

The evaluation points above need in particular to consider changes made to MILENAGE-256 compared to the predecessor MILENAGE such as changes in the OP_C calculation, changes to input formats, and the additional function ***f5****.

Related key attacks (RKA) directly targeting the Rijndael/AES block ciphers need not, and have not, been considered due to the MILENAGE operational context in which it is judged extremely difficult to mount such attacks in practice.

The mathematical evaluation approach combines:

a) Formal proofs allowing one to validate some aspects of the construction.

b) Informal security arguments regarding aspects of the construction not covered by formal proofs.

c) Investigation of "certificational attacks", in particular forgery or distinguishing attacks of complexity close to the 2^{128} bound of the requirement (ii).

10.2 Operational context

As with the 128-bit MILENAGE, the operational context in which the algorithms are used needs to be considered of estimating the practical feasibility of attacks. The most exposed environment is that of the UICC (or eSIM). An attacker has full control over what he can choose as MILENAGE-256 inputs, which creates the potential for side-channel attacks such as DPA.

The output of ***f1*** is checked within the USIM and is therefore not directly available to an attacker. A **MAC-A** (***f1***-value) which does not verify correctly in the USIM also implies that other AKA parameters are not observable if the USIM is configured to terminate computation after a failed **MAC**. However, this provides limited protection against side-channel attacks since the observation of these side-channels could be possible even during the computation of ***f1*** itself.

The input/output bandwidth of the USIM is however limited so that obtaining very large quantities of ***f***-function outputs is unlikely to be possible, in particular if the USIM is equipped with mechanisms to block operations after several failed MAC verifications.

As already noted, RKA directly targeting the Rijndael kernel are not considered to be a realistic threat.

10.3 Security analysis

The mathematical evaluation focused on verifying the strength of the ***f1/f1*-f5/f5*, f5**** constructions provided by MILENAGE-256, under the assumption that the underlying PRF-kernel is a strong block cipher.

The main criteria investigated were [10]:

- The strength of each algorithm, considered individually (resilience of key and unpredictability/unforgeability of subsequent outputs).

- The independence between algorithms (one algorithm's strength is not harmed by knowledge of input/outputs for other algorithms).

10.3.1 Soundness of the *f1/f1** MAC-functions

Similar to 128-bit MILENAGE, the ***f1*/f1*** MAC functions are equivalent to a CBC-MAC on a two times 256-bit message (two blocks) M_1 and M_2 of the format

$M_1 = \text{RAND} \oplus OP_C$,

$M_2 = (t_i \,||\, \mathbf{AMF} \,||\, \mathbf{SQN} \,||\, \mathbf{0}_{12\text{-}SQNsz} \,||\, c_i) \oplus OP_C$, $i = 0, 1$,

where $t_i = \mathbf{byte}((i)_3, \text{RAND}_{sz}, K_{sz}) \,||\, \mathbf{byte}(\text{SQN}_{sz}, \text{MAC}_{sz})$, see clause 9.1.2.3. The CBC-MAC has been proven secure if the block cipher used models a random permutation [17]. Security of the CBC-MAC is established for up to $2^{n/2}$ oracle queries, where n is the block size (i.e. $n = 256$ in this case), which is also a sharp bound.

10.3.2 Separation between *f1/f1**, *f2-f5* and *f5*/f5***

10.3.2.1 Collisions in single AKA protocol execution

With the simpler instance-separation mechanism of MILENAGE-256 through the definition of the instance value (the i-value present in IN_i), and as discussed in clause 9.1.2, the XOR between any two inputs to any distinct pair of functions computing the second layer PRFs among { ***f1*, f1, f2, … f5, f5*, f5***} always has the format

(**byte**($(i)_3$, ...) $\oplus$ **byte**($(j)_3$, ...) $\|\ldots\|$ ($c_i \oplus c_j$)),

since the two *TEMP* values of (EQ 2) are identical and will cancel. Since, $i \neq j$ and the **byte**() encoding is one-to-one, distinctness is always guaranteed through (at least) the 3-bit encoding of the term $(i)_3 \oplus (j)_3$, and this holds even if the operator has selected $c_i = c_j$. This means that collisions of outputs are expected to only happen by chance. Collisions are in fact impossible, unless the outputs are truncated to less than 256-bits. The probability of collisions can then directly be evaluated in terms of sizes of the outputs.

10.3.2.2 Collisions between distinct AKA protocol executions

Necessary conditions for collisions among (***fi***, ***fj)*** values from different protocol executions are

1. For full 256-bit ***f***-outputs:
   a. that $i = j$ and outputs are computed from the same RAND, or,

NOTE: For ***f1/f1**** and ***f5*** there is an additional requirement that **SQN**-values are the same of that **MAC-S** inputs to ***f5**** collide by chance.

   b. that $i \neq j$, the corresponding **RAND, RAND'** are distinct, but inputs to the second layer PRF are equal, i.e. if $TEMP \oplus TEMP' = IN_i \oplus IN_j$,

2. the ***f***-outputs are truncated to $s < 256$ bits

Collisions are expected to occur after about

$\min(2^{|\mathbf{RAND}|/2}, 2^{s/2})$ (EQ 5)

executions, so with current **RAND**-values of 128-bits, this constitutes the main bottleneck. However, collisions between outputs of ***f3*** or ***f4*** (i.e. **CK** and **IK** values) would still have a limited impact since these keys normally undergo post-processing as part of the 4G and 5G 3GPP standards. This post-processing also involves the **SQN**-values, which will be distinct, thereby reducing the collision probability of the final keys used on the air interface.

As with MILENAGE, for case 1b, the collisions occur in pairs, i.e. if ***fi***(**RAND**) = ***fj***(**RAND'**), then it follows by symmetry that also ***fj***(**RAND**) = ***fi***(**RAND'**), which could lead to harmful situations. For example, it could enable MAC-forgery of ***f1***(**RAND'**), through existing knowledge of ***f2***(**RAND**). However, such a collision occurs only with probability 2^{-128}.

Therefore, a main conclusion is that increasing the **RAND**-sizes to a value larger than 128- bits in a later release is advisable to maintain a security level of 256 bits.

10.3.3 Formal proof of the *f2-f5** construction

Since the original design of 128-bit MILENAGE, a formal security proof for the functions ***f2- f5/f5**** was published [27] for the case of a PRP used as the kernel. This proof applies directly also to MILENAGE-256, under the assumption that Rijndael mimics a pseudo-random permutation with security bounds scaled up accordingly. The result proves that the output of ***f2-f5*** (and ***f5****) is indistinguishable from the output of a random function within an attack complexity up to slightly below 2^{128} attacker-selected queries.

The quantitative security bounds [27, 39] change slightly if ***f5***** is used instead of ***f5****; for analysis see clause 10.3.4.3. (As mentioned, the proof [39] also covers ***f1***, but not ***f5*****.)

10.3.4 Properties of the *f5***-function

10.3.4.1 Desired and obtained security features

During a resynchronisation procedure, the current **SQN$_{MS}$**-value is sent from the ME to the network and can then also (optionally) be concealed as

$$Conc(SQN_{MS}) = SQN_{MS} \oplus f5^*_K(RAND),$$

i.e. XORing **SQN**$_{MS}$ with the output of ***f5****. This is done to prevent linkability via observation of the **SQN$_{MS}$**-value. Thus, for a given subscriber and two resynchronisation procedures occurring when the current **SQN**-values at the ME are **SQN$_{MS}$** and **SQN$_{MS}$**', respectively, it holds that

$$Conc(SQN_{MS}) \oplus Conc(SQN_{MS}') = SQN_{MS} \oplus SQN_{MS}' \oplus f5^*_K(RAND) \oplus f5^*_K(RAND'),$$

where **RAND**, **RAND'** are the two corresponding **RAND**-values from the network. Therefore, if an attacker replays the **RAND**, **AUTN** associated with the first (legitimate) resynchronisation procedure, it holds that

$$Conc(\mathbf{SQN_{MS}}) \oplus Conc(\mathbf{SQN_{MS}}') = \mathbf{SQN_{MS}} \oplus \mathbf{SQN_{MS}}',$$

since the two identical $f5^*_K(\mathbf{RAND})$ cancels. It is likely that **($\mathbf{SQN_{MS}} \oplus \mathbf{SQN_{MS}}'$)** has low Hamming-weight and/or predictable structure for a given subscriber/ME. The conclusion is that this can be leveraged by an attacker in order to determine if a certain ME/subscriber is identical to a previously observed subscriber/ME.

NOTE 1: During AKA, the ME detects that **SQN$_{HE}$** lies outside a window of acceptable values as determined by **SQN$_{MS}$**.

NOTE 2: **AUTN** will still be verified as being correct by the UE.

The defined ***f5***** function addresses this by including **MAC-S** in the input: **MAC-S** depends on **SQN$_{MS}$** in such a way that the **MAC-S** associated with distinct **SQN$_{MS}$**-values will, under cryptographic assumptions of the kernel used to derive **MAC-S**, appear cryptographically independent and pseudo-random.

Therefore, the residual success-possibility of the attack as described above is limited to the case that the ***f5***** outputs during two resynchronisation procedures are identical. This can happen if

i. **SQN$_{MS}$** has not changed since the first resynchronisation.

ii. the two **MAC-S** values of **SQN$_{MS}$** and **SQN$_{MS}$**' collide, even if **SQN$_{MS}$** ≠ **SQN$_{MS}$**', which could occur if **MAC-S** is obtained by truncation if ***f1**** output, or, if **AK*** is obtained by truncating ***f5***** output.

iii. inputs to ***f5***** collide, even though the **MAC-S** values are distinct.

Unless the second re-synch occurs for a **RAND** used in the previous re-synch (e.g. if the value is intentionally or unintentionally replayed), the events (ii) and (iii) occur basically at random due to truncations. E.g., event (ii) occurs with probability ~ max($2^{-|\mathbf{AK^*}|}$, $2^{-|\mathbf{MAC\text{-}S}|}$) and will enable an attacker to deduce $\mathbf{SQN_{MS}} \oplus \mathbf{SQN_{MS}}'$. Observe that the attacker knows if **MAC-S** values have collided.

If the same **RAND**-value is reused, event (iii) occurs with probability max ($2^{-|\mathbf{MAC\text{-}S}|}$, 2^{-240}), due to the way that the input to ***f5***** is formed, based on the **MAC-S**. In general, if **RAND** is reused, event (i) implies that Conc($\mathbf{SQN_{MS}}$) $\oplus$ Conc($\mathbf{SQN_{MS}}$') = 0, an otherwise improbable event.

In any case, setting aside the case of $\mathbf{SQN_{MS}}$ not having changed (which is nothing that can be addressed cryptographically), the total probability of adverse effects of the events above is always bounded by

$$\Pr[\mathbf{AK^*}\text{ collision}] \leq \max(\, 2^{-|\mathbf{AK^*}|}, 2^{-|\mathbf{MAC\text{-}S}|}, 2^{-240}\,). \qquad \text{(EQ 6)}$$

It should be noted that other potential vulnerabilities related to traceability and linkability, e.g. as described in the literature [26], are not addressed by the definition of ***f5*****, nor has it been the intention to do so.

NOTE: The exact collision probability is also dependent on whether or not the kernel PRF is a one-to-one function, but the difference is in practice negligible.

10.3.4.2 General soundness of the ***f5*****-function

The functions ***f5*** and ***f5**** are basically key-stream generators used for additive stream cipher encryption of the **SQN**-values, using **RAND**-value as IV. The key issue underlying the resynchronisation attack just described is that such ciphers are in general not tolerant to IV- reuse, and such IV-reuse can, in the case of ***f5****, be enforced by an attacker by active replay. Without large modifications to the AKA protocol (introducing new messages and/or exchanging additional cryptographic values), the only approach to provide some freshness assurance to the IV is to make the IV dependent on the **MAC-S** value, i.e. the output of ***f1****, since it is the only available value that encodes stateful-information (**SQN**) at the ME. However, just because it is the only *feasible* approach, this does not imply that it is cryptographically sound, so this also needs to be analysed.

The problem of IV-reuse in stream ciphers has been thoroughly studied. Several so called *synthetic IV-schemes* have been proposed [16, 32]. In these solutions, a MAC of the message serves as (part of) the stream cipher encryption IV. Unless two messages are identical, their MAC-values (and thus IV-values) then only collide by pure chance, similar to the analysis above. Additionally, the paper [32] establishes a formal security proof for the construction given therein. While there are differences in the details (e.g. the exact construction of MAC-function, key management, etc) of these works, the similarities suggest that the construction of ***f5***** is, in itself, cryptographically sound.

10.3.4.3 Effects on the ***f2-f5*** formal security proof when ***f5***** replaces ***f5****

Existing proven security bounds [27, 39] are mainly based on a collision-probability analysis of inputs and intermediate values occurring during computation of ***f2-f5****. In the case of [39], covering also ***f1/f1****, collisions in the ***f1***-inputs need also be accounted for. Since ***f5***** has slightly different behaviour from ***f5**** with respect to occurrence of such collisions, it is of interest to study how the probabilities change when ***f5***** replaces ***f5****.

Before continuing, notice that in practice, an attacker trying to distinguish the MILENAGE- 256 functions from random functions would either observe (at most) ***f1-f5*** (under normal operation), or, ***f1***, ***f1****, ***f2,*** and precisely one of ***f5**** / ***f5***** (if a re-synch procedure occurs). Therefore, allowing the attacker to observe all ***f***-functions (including ***f5*****) is a worst-case scenario that does not really occur in practice. Nevertheless, this is the case we will analyse.

Looking at the details of the proof [39], which includes functions ***f1-f5***, ***f5****, but not ***f5*****, the main ingredient is the analysis of a number of different input-collisions and these are each bounded by about 2^{-n}, where n is the bit-size of the inputs/outputs of the PRF, i.e. $n = 256$.

How does ***f5***** affect those probabilities?

Going back to the collision probabilities described by (EQ 6), the fact that truncated output **AK*** values could collide is as such not a problem, since the proof [39] establishes indistinguishability from a random function, and the random function should then also be assumed to truncate the output corresponding to **AK*** accordingly. Further, the fact that truncated **MAC-S** values, being input to ***f5*****, could cause collisions is also not an issue. This is because the attacker-model of the proof [39] needs to assume that the attacker is not allowed to query the same **RAND** more than once. Therefore, since **RAND** also affects the input to ***f5*****, collisions in the input to ***f5***** still occur basically at random, with probability about 2^{-n}. Though the discussion given here does not constitute a formal proof, there is strong reason to believe that the existing proof [39] could be adapted to establish a quantitatively similar security bound, including also ***f5*****.

10.3.5 Statistical evaluation

Statistical tests on the MILENAGE-256 framework were considered to only yield results about the underlying kernel function. No statistical tests were performed on the kernels either, given that Rijndael/AES is considered sufficiently tested and secure through the AES process and later analysis by academia. Consequently, statistical tests were not performed.

10.3.6 Side-channel attacks on AES/Rijndael

The design process concluded that it was not feasible to design a general algorithm framework that, by itself, would not be vulnerable to side channel attacks. AES/Rijndael, as with most other block ciphers, is potentially vulnerable to simple and differential power analysis (SPA and DPA) aiming to recover the secret key.

It was also concluded that the use of operator constants, OP_C, and $c_0 \ldots c_7$, in the USIM cards can only play a limited role in protecting against these kinds of attacks since it is usually possible to first retrieve the subscriber **K**, then, given **K**, other unknown values are easily reverse engineered. Hardware protection measures and masking and other implementation protection techniques such as those discussed in the literature [21, 28-29, 35, 42, 45] need to be specifically implemented for protection. Also timing attacks (TA) [30, 34] could require implementation specific countermeasures.

No result on side-channel attacks were found on Rijndael with 256-bit blocks that are not also relevant for 128-bit blocks.

Rijndael, as the AES, has been shown to readily lend itself to protection measures against side channel attacks.

10.4 MILENAGE-256

10.4.1 Published cryptographic attacks on Rijndael with 256-bit blocks

A study [16] of known attacks against Rijndael-256-256 concluded that no cryptographic attack is known which can attack more than 10 rounds (out of the 14). The best attack is based on impossible differentials [37] which for 10 rounds has a data/time/memory complexity of about 2^{244} / 2^{140} / 2^{186}.

Most cryptanalysis work for Rijndael with 128-bit blocks (i.e. AES) also attempts to attack the 256-bit block version and it is therefore reasonable to assume that the 256-bit block version has also been thoroughly analysed.

10.4.2 MILENAGE-256: Summary

Given the status of Rijndael-256-256 and existing security proof of the MILENAGE construction when based on a PRP [27], the security of this algorithm set meets the design objectives, given the current state of knowledge in the literature.

10.5 Resistance to quantum computing attacks

Since the algorithm sets were designed to resist attacks leveraging cryptographically relevant quantum computers, this resistance also needs to be evaluated.

A quantum-empowered attacker may undertake generic attacks, such as a generic search using Grover's algorithm to find the key. Such generic attacks cannot be prevented by design choices; however, Grover's algorithm only gives a mild (quadratic) speedup, relative to classical attacks. Hence, one has a theoretical attack complexity of order 2^{128} (for 256-bit keys), though the complexity is generally regarded to be much higher than 2^{128} in practice.

Further, implementations of Grover's algorithm on quantum computers are less efficient than the classical model when it comes to parallelisation: on N computers, only $\sqrt{N}$ –fold speed-up is achieved. Hence, this is not considered as a relevant threat.

More generally, a recent report [44] presented several quantum computing-based attack vectors. It was concluded that the reported attacks fall into one of three categories, as follows.

1) The generic key search attack using Grover's algorithm, which, as mentioned above, cannot be protected against (beyond increasing parameter length, such as the key size).

2) Attacks using a non-standard model permitting "quantum superposition queries".

3) Other potentially relevant attacks in the "standard" quantum computing model.

NOTE 1: The generic attack leveraging Grover's algorithm also employs the standard quantum computing attack model and could therefore be grouped together with attacks in category 3. Nonetheless category 1 is here separated from category 3 following previous conventions, to distinguish the standard generic Grover's attack from the other attacks discussed.

Notice that, when investigating quantum-empowered adversaries that attack a keyed algorithm, the attacker may or may not have the ability to make quantum (i.e. superposition) queries to the algorithm. In category 2 above, the attacker is empowered to submit both classical and quantum queries to the keyed algorithm. This differs from the standard quantum computing model of category 3, in which an attacker can only make classical queries to the keyed algorithm but may use an offline quantum computer to make quantum queries to the unkeyed algorithm.

NOTE 2: Specially, such offline attacks do not involve the secret key but typically only keys chosen by the adversary and deployed within an offline quantum implementation of the algorithm.

In the present context, the attacks in category 2 can be ignored as practically irrelevant, at least for the economic lifetime of the MILENAGE-256 designs [3]. A classical attack on Rijndael is, though not very likely, still much more likely than the realisation of the quantum attack model needed in case 2. Regarding category 3, it appears reasonable to consider an attack model in which the attacker may submit classical queries to the keyed algorithm and use a quantum computer to construct an unkeyed (quantum) implementation of MILENAGE-256 offline, availing offline superposition queries. Accordingly, attacks in category 3 cannot be dismissed *a priori*.

Two attacks in category 3 are considered. The first is still a generic attack, in the sense that it rests on Grover's algorithm to retrieve the long-term key **K**, even if *OP* is unknown. The reported complexity is $2^{128 + m/2}$, where *m* is the size of OP [44]. This result displays the standard quadratic (theoretical) speedup relative to classical attacks. Since the size of *OP* is bounded by the block size of the underlying PRF, this attack might be an issue if a kernel with 128-bit block size is under consideration. However, for a 256-bit block size PRF, as employed in MILENAGE-256, there is no practical relevance.

The other attack in category 3 is an offline key-recovery attack targeting the ***f2*** MAC function, with complexity 2^{128+m}, where *m* is the PRF block size [44]. There is no practical need to mitigate against this attack. For MILENAGE-256, which uses a 256-bit block size, the feasibility of this attack is further diminished.

An additional related key attack in category 3 [44] is not considered here, for reasons discussed in clause 10.1.

10.6 Complexity evaluation

10.6.1 MILENAGE-256 framework

As observed in clause 7.4.2, each evaluation of the kernel PRF_K needs to be computable in 80msec or less. Specifically, for each PRF_K evaluation, MILENAGE-256 requires one evaluation of Rijndael-256-256 (and possibly, one computation of the key schedule). The total number of PRF invocations to compute all seven f-functions is bounded by eight.

NOTE: Use of ***f5**** an ***f5***** is mutually exclusive. In fact, it is never necessary to compute more than five ***f***-functions: ***f1*-*f5*** under normal operation and ***f1***, ***f5***, ***f1**** and ***f5***** if a resynchronisation is necessary.

An efficient implementation of Rijndael with 256-bit block and key size could be 50-100% slower than a similarly efficient implementation of AES with 128-bit block size and the same key size: it can be observed that an evaluation of the Rijndael-256-256 round function can be performed by two times invoking a black-box implementation of the AES-256 round function, plus application of a fixed byte permutation (see figure 30 in [40]). Since the number of rounds in Rijndael-256-256 and AES-256 are identical, this suggest that Rijndael- 256-256 is not much more than twice as computationally expensive as AES-256 in the worst case.

The most memory consuming part of an implementation would be the space needed for storing the doubled state and the expanded keys (while the round keys can, as well, be derived on-fly during the encryption process). For MILENAGE-

256, this requires 15 x 32 = 480 bytes. The storage space for MILENAGE-256 can be static (since each PRF instance uses the same key).

Other quantities that need to be stored or fused are the OP_C value and the seven INi values, totalling 7 x 32 = 224 bytes.

10.6.2 Complexity of Rijndael

For an idea of performance of Rijndael-256-256 implementations on an 8-bit microcontroller, the following numbers are reported [15]: code size 696 bytes, RAM size 136 bytes, cycle count (for input data size less than one block) ca 13k. At a low clock frequency of 3MHz, this corresponds to about 4.5msec execution time.

NOTE: Since RAM access is expensive on the target platform, the implementation seeks to minimise RAM usage.

11 Conclusions

This report presented the rationale behind the design, security, and performance evaluation of the MILENAGE-256 algorithm set. MILENAGE-256 meets the security assurance level required of authentication and key agreement algorithms targeting an overall 256-bit security level.

Annex A :
Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2024-11	SA3#119	S3-245101				TS skeleton	0.0.0
2024-11	SA3#119	S3-245105				Addition of the text based on the selection of Rijndael-based Milenage-256 to specify Milenage-256 algorithm, and there is an editorial clean up for presentation to TSG-SA.	0.1.0
2024-12	SA#106	SP-241790				Presented for information and approval	1.0.0
2025-01						Upgrade to change control version	19.0.0