3GPP TR 35.909 V4.0.0 (2001-04)

Technical Report

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
3G Security;
Specification of the MILENAGE Algorithm Set:
An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*;
Document 5: Summary and results of design and evaluation
(Release 4)

Keywords
Security, Algorithm

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 Report (TR) has been produced by the 3rd Generation Partnership Project (3GPP).

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

Version x.y.z

where:

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

Introduction

This Report has been produced by ETSI SAGE Task Force 172 on the design of an example set for 3GPP Authentication and Key Generation Algorithms.

The work described in this report was undertaken in response to a request made by 3GPP TSG SA.

SAGE Version 1.0 of this report was submitted to the 3GPP SA WG3 group in December 2000. Version 1.1 (with updated C-code in Annex 4) was approved by TSG SA#10 in December 2000.

1 Scope

This report contains a detailed summary of the work performed during the design and evaluation of the 3GPP Authentication Functions denoted as the MILENAGE algorithm set. It contains all results and findings from this work and should be read as a supplement to the specifications of the algorithms in ref. [3] and the general project report, ref. [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] 3G TS 33. 102 V 3.5.0 (2000-07) 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture.

[2] 3G TS 33. 105 V 3.4.0 (2000-07) 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements. (Release 1999)

[3] ETSI/SAGE Specification. 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: Algorithm Specification. Version: 1.0; Date: 22nd November 2000.

[4] ETSI/SAGE Report. Report on the Design and Evaluation of the 3GPP Authentication and Key generation Functions; Version: 1.0; Date: 22nd November 2000.

[5] Wassenaar Arrangement, December 1998. http://www.wassenaar.org.

[6] P. C. Kocher, 'Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems', *CRYPTO'96, LNCS 1109*, Springer-Verlag, 1996, pp. 104-113.

[7] J. Kelsey, B. Schneier, D. Wagner, C. Hall, Side Channel Cryptanalysis of Product Ciphers', *ESORICS'98, LNCS 1485*, Springer-Verlag, 1998, pp. 97-110.

[8] L. Goubin, J. Patarin, 'DES and differential power analysis', *CHES'99, LNCS 1717*, Springer-Verlag, 1999, pp. 158-172

[9] P. Kocher, J. Jaffe, B. Jun, 'Differential Power Analysis', *CRYPTO'99, LNCS 1666*, Springer-Verlag, 1999, pp. 388-397.

[10] T. S. Messerges, 'Securing the AES finalists against Power Analysis Attacks', *FSE'00, LNCS*, Springer-Verlag, to appear.

[11] L. Goubin, J.-S. Coron, 'On boolean and arithmetic masking against differential power analysis,' *CHES'00, LNCS*, Springer-Verlag, to appear.

[12] Nechvatal, Barker, Bassham, Burr, Dworkin, Foti and Roback, 'Report on the Development of the Advanced Encryption Standard (AES)', NIST, October 2, 2000.

[13] F. Sano, M. Koike, S. Kawamura and M. Shiba, 'Performance evaluation of AES Finalists on the High-End Smart Card', The Third AES Candidate Conference, New York, April 2000.

[14] M. Bellare, J. Kilian, P. Rogaway, The Security of Cipher Block Chaining, proceedings of Crypto'94, Springer Verlag, pp341-358.

[15] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES algorithm submission. September 3, 1999, available at http://www.nist.gov/aes.

[16] H. Gilbert and M. Minier, *A collision attack on 7 rounds of Rijndael*, in The Third AES Candidate Conference, printed by the National Institute of Standards and Technology, April 13-14, 2000, pp. 230-241.

[17] S. Lucks, *Attacking Seven Rounds of Rijndael Under 192-bit and 256-bit Keys*, in The Third AES Candidate Conference, printed by the National Institute of Standards and Technology, April 13-14, 2000, pp. 215-229.

[18] N. Ferguson, et al., *Improved Cryptanalysis of Rijndael*, in the preproceedings of the Fast Software Encryption Workshop 2000, April 10-12, 2000.

3 Abbreviations

For the purposes of the present report, the following abbreviations apply:

AES	Advanced Encryption Standard
AMF	Authentication Management Field
AK	Anonymity Key
AuC	Authentication Centre
CBC	Cipher Block Chaining
CK	Cipher Key
DES	Data Encryption Standard
DPA	Differential Power Analysis
EEPROM	Electronically Erasable Programmable Read-Only Memory
GF(q)	The finite field of q elements
3GPP	3rd Generation Partnership Project
IPA	Inferential Power Analysis
IK	Integrity Key
IV	Initialisation Vector
K	Subscriber Key
MAC	Message Authentication Code
MAC-A	Network Authentication Code
MAC-S	Resynchronisation Authentication Code
OFB	Output feedback mode
OP	a 128-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1, f1*, f2, f3, f4, f5 and f5****
OP_C	a 128-bit value derived from OP and K and used within the computations of the functions ***f1, f1*, f2, f3, f4, f5 and f5****.
RAM	Random Access Memory
RES	Response to Challenge
RNC	Radio Network Controller
ROM	Read Only Memory
SAGE	Security Algorithms Group of Experts
SPA	Simple Power Analysis
SQN	Sequence Number
TA	Timing Attack
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
USIM	User Services Identity Module
XRAM	Extended RAM
XRES	Expected User Response

4 Structure of this report

The material presented in this report is organised in the subsequent clauses, as follows:

- Clause 5 provides background information to the design work of the example set for 3GPP Authentication and Key generation 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 designs;
- Clause 9 provides some background information on the chosen design;
- Clause 10 gives an overview of the evaluation work carried out by SAGE 3GPP TF and other parties;
- 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 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. [1]. The implementation of these security services must be based on a variety of cryptographic functions/algorithms and the requirements for these functions are provided in ref. [2]. Out of the full algorithm suite, only the UMTS encryption algorithm (***f8***) and the UMTS integrity algorithm (***f9***) are fully standardized. This work was conducted by a dedicated task force based on ETSI SAGE and external experts from 3G manufacturers.

The remaining cryptographic functions for authentication and key agreement (***f0*** – ***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, the 3G Security Group agreed to develop an example set of functions that could be offered to operators that chose not to develop their own solutions. Again a task force was set up based on ETSI SAGE enlarged with cryptographers from 3G manufacturers.

Note that the random challenge generating function ***f0*** is not included in the example set of functions provided by this work. The implementation of this function is completely determined by the operator.

The major design goal for the task force was to design a framework for the authentication and key generation functions that was secure and flexible. This goal was achieved through the development of a well-analysed construction using a 128-bit encryption algorithm as a kernel function and including an additional configuration field parameter selected by the operator. The example design recommends the use of the AES algorithm Rijndael as the kernel function, but an operator could change this to any block cipher meeting the interface parameters. The list of candidates for the AES standard includes a large set of suitable algorithms to choose from.

The defined set of algorithms is commonly denoted as the MILENAGE algorithms.

6 Summary of algorithm requirements

The requirements for the cryptographic algorithms used in the 3G Security are found in ref. [2]. We include the requirements that are essential for the reading of this report.

6.1 General requirements for 3GPP cryptographic functions and algorithms

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, ref.[5].

6.2 Authentication and key agreement functions

The mechanisms for authentication and key agreement described in clause 6.3 of [1] require the following cryptographic functions:

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 re-synchronisation

6.2.1 Implementation and operational considerations

The functions ***f1—f5**** shall 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 300byte 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 shall 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-synchronisation message authentication function

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

f1* should be a MAC function. In particular, it shall 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 shall 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 shall 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 shall 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 shall 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-synchronisation

f5*: (K; RAND) # AK

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

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

7 Design criteria

Based upon the general requirements, the task force developed a set of design criteria for the work.

7.1 Cryptographic Criteria

1. Without knowledge of secret keys, the functions ***f1, f1*, f2, f3, f4, f5 and f5**** should be practically indistinguishable from independent random functions of their inputs (RAND||SQN||AMF) and RAND.

   Examples: Knowledge of the values of one function on a fairly large number of given inputs should not enable its values to be predicted on other inputs. The outputs from any one function should not be predictable from the values of the other functions (on the same or other inputs).

2. It should be infeasible to determine any part of the secret key K, or the operator variant configuration field, OP, by manipulation of the inputs and examination of the outputs to the algorithm.

3. Events tending to violate criteria 1 and 2 should be regarded as insignificant if they occur with probability approximately 2^{-128} or less (or require approximately 2^{128} operations).

4. Events tending to violate criteria 1 and 2 should be examined if they occur with probability approximately 2^{-64} (or require approximately 2^{64} operations) to ensure that they do not have serious consequences. Serious consequences would include recovery of a secret key, or ability to emulate the algorithm on a large number of future inputs.

5. The design should build upon well-known structures and avoid unnecessary complexity. This will simplify analysis and avoid the need for a formal external evaluation.

7.2 Implementation Criteria

In addition to the performance requirements listed in 6.2.1, the task force agreed to ensure that the listed requirements would be met even after implementation of protection mechanisms against side channel attacks like differential power analysis (DPA).

7.3 The need for an Operator Variant Algorithm Configuration Field

In response to a request from SA3 the task force decided to include the use of the Operator variant field, OP. This configuration field is used for adding operator dependent information to the design even if the choice of the kernel function is the same.

The roles of OP are:

1. To make each operator's implementation different.

2. To prevent USIMs for operators being interchangeable, either through trivial modification of inputs and outputs or by reprogramming of a blank USIM.

3. To keep some algorithm details secret.

4. To provide some protection against a poorly chosen kernel.

7.4 Criteria for the cryptographic kernel

The kernel function is used by MILENAGE ("the framework") to produce a 128 bit output value from a 128 bit input value from which the output of the specific mode (one of the functions **f1** – **f5**, **f1*** or **f5***) is derived. These output values are produced under the control of a 128 bit user specific key K. It should be noted that K is a long term secret which must be protected under any circumstances.

7.4.1 Implementation and operational considerations

The performance requirements for the full 3GPP algorithm set are given in 6.2.1. From this budget we allocate at most 6Kbytes ROM and about 200 bytes RAM to the kernel function. The kernel function shall produce a 128 bit output value in less than 50 ms execution time.

7.4.2 Functional requirements

The purpose of the kernel function is to map an input value *p* (the plaintext) to an output value *c* (the ciphertext) under the control of a key K. The key shall be hidden, i.e. it shall be (computationally) infeasible to derive K if an arbitrary amount of pairs (*p*,*c*) are known and K is fixed. It shall also be infeasible to compute K by repeatingly choosing *p*, applying the kernel function and observing the resulting *c* several times. The latter chosen plaintext attack shall even be impossible 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 (see also ref. [3], section 5.2).

Furthermore, it shall be infeasible to compute *c* given *p* if K is not known but an arbitrary amount of plaintext/ ciphertext pairs are known which were produced using the same K.

There is no need for the kernel function to be invertible. However, since the input and output values have the same size and collisions should be avoided, a bijective function would be a good choice.

7.4.3 Types and parameters for the kernel

The kernel is to be a keyed function from *n* bit blocks to *n* bit blocks. An example of such a keyed function is a symmetric block cipher with a blocksize of *n*.

The parameters of the kernel are as follows:

block length: 128 bits

key length: 128 bits.

Both the key and the input/output blocks are unstructured data (at least from the kernel function's view).

The AES candidates are good examples for kernels, which meet these requirements.

Interfaces to the kernel:

The following interfaces to the kernel function are defined:

data input: X[0], X[1], …, X[127] where X[i] is the data input bit with label i;
data output: Y[0], Y[1], …, Y[127] where Y[i] is the data output bit with label i;
key input: K[0], K[1], …, K[127] where K[i] is the key bit with label i.

8 The 3GPP MILENAGE algorithms

The detailed specifications of the 3GPP MILENAGE algorithms are found in ref. [3]. The following diagram shows the design of the functions ***f1, f1*, f2, f3, f4, f5*** and ***f5**** using the kernel function denoted $E_{K.}$.

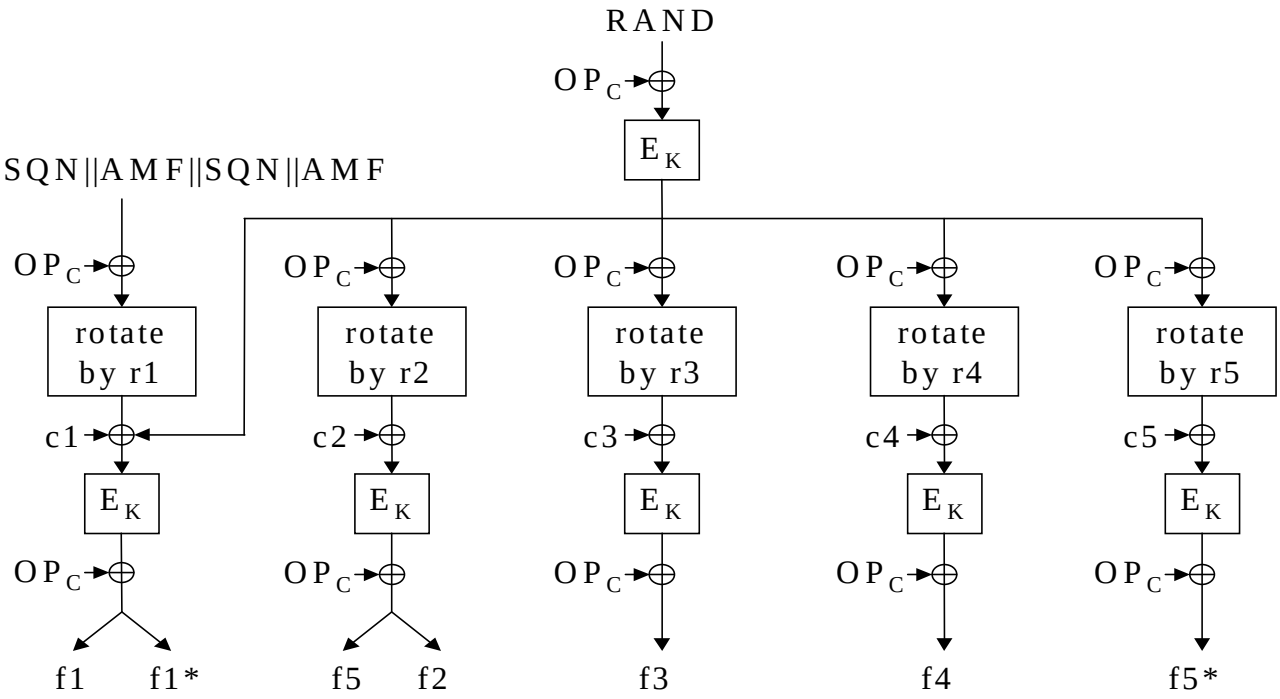

Figure 1: Definition of f1, f1*, f2, f3, f4, f5 and f5*

Document [3] recommends use of the AES algorithm Rijndael for the kernel function E_K, but this choice could be replaced by any 128-bit encryption algorithm employing a 128 bit key or keyed function fulfilling the requirements of section 7.4. The value OP_C is derived from the subscriber key K and the operator dependent value OP by encrypting OP using K as the secret key, i.e.

$$OP_C = OP \oplus E[OP]_K$$

r1, …, r5 are five fixed rotation constants and c1, …, c5 are five fixed addition constants defined in ref. [3]. These values will ensure that the inputs to the different functions will be different. Finally, ref. [3] defines which part of the outputs that are used for the different functions.

9 Rationale for the chosen design

9.1 Block ciphers vs. hash functions

It was decided to design MILENAGE around a cryptographic kernel function with strong one-wayness properties. To be realistic such an approach requires that well scrutinized examples of suitable kernel functions are publicly available and preferably on royalty-free basis. Among the cryptographic functions that would offer the required one-wayness properties two different types can be identified, block ciphers and hash-functions. The pros and cons offered by these two alternatives were weighed up against the specific requirements of the use and the implementation environment. The decision of selecting a 128-bit block cipher as a cryptographic kernel was justified by the following aspects.

1. Efficiency of the smart card implementation.

   It is required that the kernel function can be efficiently implemented on smart cards with eight-bit processors. The known and commonly used hash-functions are all optimised for larger word-size, typically 32 bits.

2. Security of the smart card implementation.

   The DPA and other side channel attacks are better understood and analysed in the open literature for certain block ciphers than for any hash-functions. Also protection measures are better developed for certain block cipher structures.

3. Fixed input length.

   The inputs to the kernel function are parameters of fixed length less than or equal to 128 bits. New block ciphers with 128-bit block size are suitable for handling such inputs.

4. Secret key input.

   Block ciphers are designed to take a secret key input. For hash-functions such a functionality must be constructed artificially. Special keyed modes of operation have been designed for hash functions in the Internet and ISO standards. In ISO 9797 Part 2 three MAC algorithms for dedicated hash functions have been specified. Two of them take at least two applications of the round function of the hash function, which adds extra complexity. One of them, MAC Algorithm 3 is specially designed to take a short maximum 256 bits input and only one application of the round function of the hash-function. Of these three MAC algorithms only the Internet HMAC standard is freely available

5. Availability of block ciphers.

   There have been many block ciphers around for many years and knowledge about their designs and implementations are well understood and widely known. Even if published 128-bit block ciphers using a 128-bit key have not been around for that many years, the AES process has provided a suite of good candidates for the kernel. On the other hand, there are only a handful candidates of hash functions that are considered secure today.

9.2 The choice of Rijndael

The task force agreed to propose the block cipher Rijndael for use as the kernel in the ***f1-f5**** constructions. There were several arguments to support this choice:

- It was a strong encryption algorithm. At that time it was one of the five AES finalists.
- It was effective and fast on several platforms.
- It was highly suitable for smart card implementation.
- It was freely available without any kind of IPR limitations.
- It could be protected against side channel attacks.
- It had the required input/output interface.
- It had been published and studied for some time and was built upon the design of a previous algorithm called SQUARE.

In October 2000 Rijndael was chosen as the winner of the AES contest and this should be seen as a strong qualifier for its suitability in the 3GPP environment. We refer to ref. [12] for a detailed description on the evaluation and merits of Rijndael and the other AES finalists. To quote from the conclusions of this report:

Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environment regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it well suited for resticted-space environments, in which it also demonstrates excellent performance. Rijndael's operations are among the easiest to defend against power and timing attacks. Additionally, it appears that some defence can be provided against such attacks without significantly impacting Rijndael's performance.

Note that the kernel function of ***f1-f5**** will only use Rijndael in encryption directions and the concerns related to complexity of both encryption and decryption mode will not apply to its use in MILENAGE.

9.3 The MILENAGE architecture

9.3.1 Use of OP

OP is a 128-bit Operator Variant Algorithm Configuration Field used to provide unique variants of MILENAGE.

It was discussed if OP should be used directly in the algorithms or rather a derived value should be involved. The task force decided to derive a subscriber dependant value OP_C from OP and the secret key K in a non-invertible way as defined in Section 8. This construction is non-invertible in both variables even if one of them is known. In this case there is no need for storage of OP in each USIM. This means that even if the USIM is compromised, the value of OP could still be kept secret.

The value OP_C is exclusive ored to input and output of the kernel functions. This will provide additional protection against attacks.

The task force recommends to compute OP_C off the USIM as part of the pre-personalisation process. This will simplify the algorithms in the card and avoid the storage of OP on the card.

An operator could also select different values of OP for different subscribers or subscriber groups.

It is recommended that OP is kept secret, but MILENAGE is designed to be secure even if the value of OP is known to the cryptanalyst.

9.3.2 Rotations and constants

The rotations r1, r2, …,r5 and the addition constants c1, c2, …,c5 are carefully selected to ensure separation between all the cryptographic functions involved. It is shown in Section 10.3.2 that the selected values will protect against collisions in the input (and thus the output) of the final E_k computations. If an operator decides to implement other values for these constants, it is strongly advised that the requirements of Section 5.3 in ref. [3] are taken into account.

9.3.3 Protection against side-channel attacks

The protection against side-channel attacks is achieved through the selection of a kernel that allows for a protected implementation within the time constraints given by the requirements in Section 6.2.1. No attempts were given to provide such protection by the surrounding architecture.

9.3.4 The number of kernel operations

For each of the seven functions the input value RAND passes through two complete rounds of the kernel function before the output values are produced. The encryption of OP in the pre-personalisation procedure provides an extra level of security. The other inputs to **f1/f1*** are obfuscated by xor with a random value ($E[OP_C \oplus RAND]_K$) and an unknown constant (OP_C) before they enter the kernel function.

As discussed in Section 10.3.3 there are certain forgery attacks against the proposed architecture that involve 2^{64} computations. These attacks are not considered feasible within the operational context of 3GPP and would not justify the computational overhead of adding another operation of the kernel function.

9.3.5 Mode of operation

The **f1** and **f1*** constructions are essentially equivalent to the standard CBC MAC mode applied to the input blocks RAND and SQN || AMF || SQN || AMF. The soundness of this construction is theoretically justified by the results in ref. [14].

The functions **f2**, **f3**, **f4** and **f5** are defined as a kind of double encryption with a "counter-mode" construction caused by rotations and constant additions before the second encryption. The soundness of this construction is therefore a direct consequence of the use of a strong kernel function. See Section 10.3.1 for more details.

Section 10.3.2 provides analysis about the necessary separation between the different cryptographic functions involved.

10 Evaluation

10.1 Evaluation criteria

Due to the fact that the Rijndael (key length =128, block length =128 and number of rounds =10) block cipher has undergone an extensive analysis during the AES process [12], the mathematical evaluation to be done by the 3GPP AF task force will not duplicate that work and perform extra research on the cryptanalysis of Rijndael, but rather focus upon assessing the strength of the construction for deriving the **f1** to **f5*** modes of [2] from a strong 128-bit oriented block cipher E.

The main purpose of the mathematical evaluation is to check that the **f1**-**f5*** construction does satisfy the two following requirements :

Under the assumption that the underlying 128-bit block cipher E_K is a strong block cipher, i.e. that there is no efficient test allowing to distinguish the E_K permutation generator from a randomly drawn permutation of $\{0,1\}^{128}$ with substantially less than 2^{128} encryption or decryption results and significantly less than 2^{128} E_K operations,

(1) There must be no attack of complexity substantially less than 2^{128} E_K computations allowing to recover any information on the value of K key or to forge outputs of the algorithm for a large set of arbitrary RAND inputs, based on the knowledge of **f1**-**f5*** outputs corresponding to any chosen RAND, SQN, AMF inputs, even if the OP, ci and ri values are known.

(2) There must be no other attack enabling to distinguish the 7 function generators $K \mapsto$ **f1**, **f1***, **f2**, **f3**, **f4**, **f5**, **f5*** from independent random functions of the 192-bit input RAND||SQN||AMF (**f1** and **f1***modes) or the 128-bit input RAND (other modes) with substantially less than 2^{64} queries, even if the OP, ci and ri values are known. Thus in particular given any combination of $n << 2^{64}$ **f1**-**f5*** outputs related to any chosen inputs, it must be computationally infeasible to predict any additional output for any of the fi or fi* function – even if outputs corresponding to the same RAND value are known for the other modes.

For that purpose, the mathematical evaluation needs to consider:

- the strength of each of the **f1**-**f5*** modes considered individually ;
- the independence between the **f1**-**f5***modes.

Given the operational context of use of the algorithm, related key attacks do not need to be considered.

The mathematical evaluation approach will combine:

- formal proofs allowing to validate some aspects of the modes construction ;
- informal security arguments on aspects of the modes construction not covered by formal proofs ;
- an investigation of "certificational attacks", in particular forgery or distinguishing attacks of complexity close to the 2^{64} bound of requirement 2.

The following particular issues need to be taken into account in the analysis:

- role of the ci and ri constants and security conditions on their values;
- protection of OP_C.

10.2 Operational Context

The evaluation criteria discussed in Section 10.1 are basis for the mathematical analysis conducted below. Besides the theoretical work related to the security of MILENAGE, it is also important to consider the operational context in which the algorithms are used. It should be borne in mind that in practice the following operational factors exist:

The prime point of attack is directly on the USIM. In this environment:

- an attacker has full control over what he can choose for RAND, SQN, AMF.
- The output of **f1** (MAC-A) is checked within the USIM and is not directly available to an attacker.
- The input/output bandwidth of the USIM is limited (as is its processing power). As a result the practical rate at which input/output pairs can be collected is severely limited (~10 pairs or less per second?).

10.3 Analysis

As explained in Section 10.1, the main purpose of the mathematical evaluation is to analyse the construction for the **f1** to **f5*** functions under the assumption that the 128-bit block cipher used in the construction (e.g. Rijndael) is strong. The analysis do not only want to investigate the strength of each of the **f1** to **f5*** functions considered individually, but also their cryptographic separation.

The evaluation results presented here cover the following complementary aspects of the **f1-f5*** construction :

- Section 10.3.1considers the **f2** to **f5*** functions and provides a formal proof (in a certain security model) of the soundness of the individual functions and their separation;
- Section 10.3.2 considers the **f1-f1*** construction and its separation from **f2-f5***, and provides informal arguments supporting that part of the design;
- finally, Section 10.3.3 investigates certificational forgery or distinguishing attacks, and checks that none of the "attacks" identified is stronger than the attacks requiring 2^{64} queries anticipated in the design criteria.

10.3.1 A formal proof of the soundness of the f2-f5* construction

This section contains an investigation of the pseudorandomness of the simplified scheme of Figure 2, which keeps the most distinctive features of the actual **f2-f5*** construction. In Figure 2 a_1 to a_t are assumed to be any t fixed known distinct constants. The t parameter denotes the number of distinct output blocks used in the construction. In practice, for the **f2** to **f5*** functions, t is equal to 4.

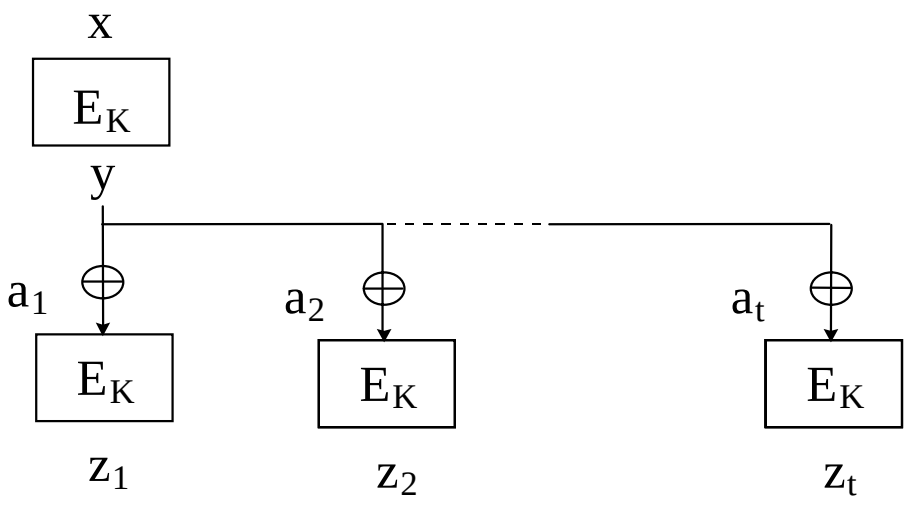

Figure 2: A simplified counter mode construction

The aim is to give some evidence that there is no way to use any combination of significantly less than 2^{64} output values z_1 to z_t to predict any new z_i output value. More generally, it will be shown that if E_K behaves as a random permutation of the $\{0,1\}^n$ set (where n = 128) , then the $x \mapsto (z1,z2, ..., zt)$ function behaves as a random function from $\{0,1\}^n$ to $\{0,1\}^{n \cdot t}$.

For that purpose, let the occurrences of the E_K function in Figure 2 be replaced by a perfect random permutation c* (i.e. a uniformly drawn element of the set of $\{0,1\}^n$ permutations). The construction then becomes a random function generator (or in other words a random function) f from $\{0,1\}^n$ to $\{0,1\}^{n.t}$, that will be compared with a uniformly drawn random function f* from $\{0,1\}^n$ to $\{0,1\}^{n \cdot t}$ (cf Figure 3).

Let A be any distinguishing algorithm of unlimited power that, when input with a φ function from $\{0,1\}^n$ to $\{0,1\}^{n \cdot t}$ (which can be modelled as an "oracle tape" of a Turing Machine) selects a fixed number q of distinct chosen or adaptively chosen input values x_1 to x_q (the queries), obtains the q corresponding φ output values y_1 to y_q, and based on these results tries to determine whether φ function is an instance of the f or of the f* generator. Denote by p the probability of A to answer 1 when fed with a random instance of f, and by p* the probability of A to output 1 when fed with a random instance of f*.

The following theorem, whose proof is given in a more comprehensive paper appended to this report, provides an upper bound on the advantage $Adv_A(f, f^*) = |p-p^*|$ of A for distinguishing f from f* in q queries:

Theorem: *Let n be any fixed integer. Denote by c* any perfect random permutation of $\{0,1\}^n$. Let f = Φ(c*) denote the random function of $\{0,1\}^n$ to $\{0,1\}^{n \cdot t}$ obtained by applying the counter mode construction of Figure 3 to c*, and let f* denote a perfect random function of $\{0,1\}^n$ to $\{0,1\}^{n \cdot t}$. For any distinguishing algorithm A using a fixed number q of queries we have*

$$Adv_A(f, f^*) \leq 3 \cdot t^2 \cdot q^2 / 2^{n+1}$$

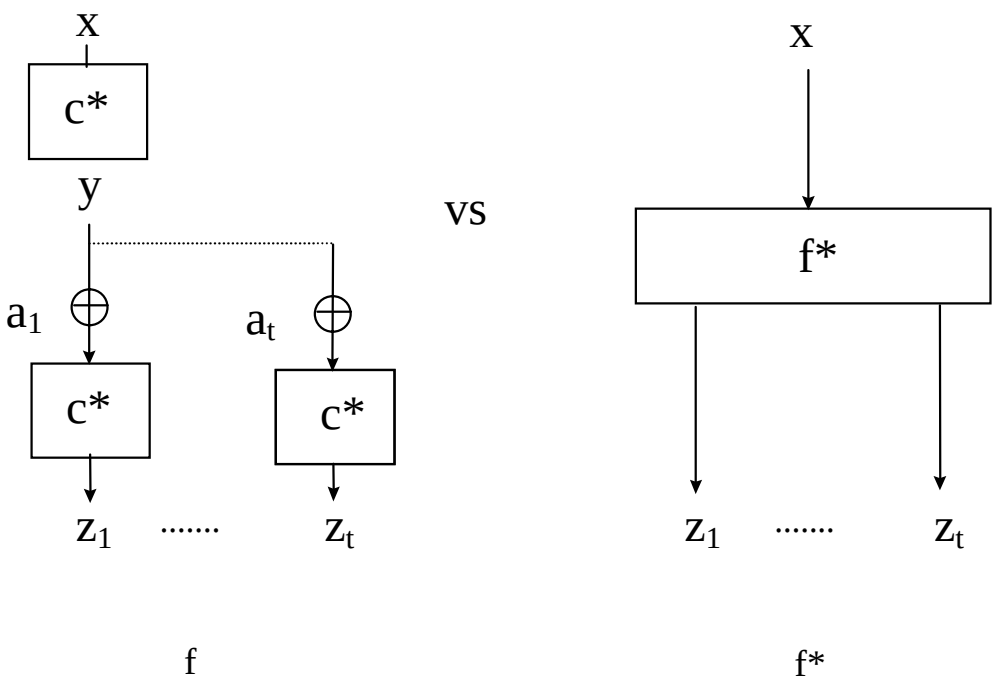

Figure 3: The two random function generators f and f* to be compared
c* is a perfect random permutation of $\{0,1\}^n$,
f* is a perfect random function from $\{0,1\}^n$ to $\{0,1\}^{n.t}$

The previous theorem gives some evidence that there is no attack requiring substantially less than 2^{64} queries against the slightly simplified version of **f2** to **f5*** in which OP_C is omitted (or replaced by a known constant) and the rotations are also omitted. It is trivial to adapt the proof of the theorem to accommodate the rotations; and the involvement of OP_C does not seem to strongly degrade the security of **f2** to **f5** (it does not seem easy to derive OP_C, and even if OP was known, it will only provide a very slight information concerning K, namely the E_K input-output pair determined by the equation $E[OP]_K = OP_C \oplus OP$).

Thus in summary, the design of the **f2-f5*** functions appears to be sound and to comply with the design criteria.

10.3.2 On the f1-f1* construction and its separation from f2-f5*

10.3.2.1 Soundness of the f1-f1* construction

The **f1** and **f1*** constructions seem essentially equivalent to a standard CBC MAC applied to the 2-block message M1|| M2, where M_1 = RAND and M_2 = SQN||AMF||SQN||AMF, with a final truncation of the CBC computation output. Moreover **f1** and **f1*** use distinct output bits - so that the cryptographic separation between **f1** and **f1*** appears to be sufficient.

A formal proof of the soundness of the standard CBC MAC was first established by Bellare, Kilian and Rogaway, ref.[14]. Some results of [14] can be transposed to the actual **f1-f1*** construction, using techniques similar to those applied in the previous section, to show that if we ignore OP_C, but keep the c1 and r1 constants and replace E_K by a perfect random permutation c*, then the 2-block to 1-block random function f associated with the **f1-f1*** construction is indistinguishable from a random function with substantially less than 2^{64} queries.

Conversely, there exists a simple internal collision attack against the standard CBC MAC requiring about 2^{64} queries, so that the conjectured 2^{64} lower bound is also an upper bound. This internal collision "attack" (which has no practical significance in the context of use of **f1** and **f1***) can be transposed to the **f1** and **f1*** functions, as shown in Section 10.3.3.

So in summary the **f1-f1*** construction appears to be sound, and no stronger weakness than the impractical attacks with 2^{64} queries anticipated in the design criteria was identified.

10.3.2.2 Separation between f1-f1* and f2-f5*

From now on and until the end of Section 10.2, the following short notation for the various input, output and intermediate variables involved in the **f1-f5*** computations will be used (cf Figure 1) :

- **x denotes the RAND input ;**
- **y denotes the intermediate value $E[RAND \oplus OP_C]_K$**
- **t denotes the SQN||AMF||SQN||AMF additional input;**
- **z1, z1*, z2, z3, z4, z5, z5* denote the f1-f5* outputs;**
- **zz1, zz2, zz3, zz4, zz5 denote the 128-bit E_K outputs from which z1 and z1*, z2 and z5, z3, z4, and z5* are respectively extracted;**
- **the OP_C, r1 to r5, c1 to c5 notations of the f1-f5* specifications are kept.**

This section investigates the independence (as seen from an adversary's point of view) between the **f1-f1*** functions and the **f2** to **f5*** functions.

In practice it is of importance to look at the pairwise independence between **f1** or **f1*** and any of the **f2** to **f5*** functions. A connection between the zz1 128-bit output block from which the **f1** or **f1*** output is extracted and the zzi = zz2, zz3, zz4, or zz5 128-bit output blocks from which **f2** to **f5*** outputs are extracted could lead to attacks from an adversary who are able to predict events of the form zz1(x,t) = zzi(x').

Express zz1 and zzi as : $zz1(x,t) = OP_C \oplus E_K(y \oplus c1 \oplus rot(t \oplus OP_C, r1))$ and $zzi(x) = OP_C \oplus E_K(ci \oplus rot(y \oplus OP_C, ri))$. Events of the form zz1(x,t) = zzi(x), i.e. collisions involving the same x random challenge value, have to be looked at with particular care because :

- despite the fact that y is unknown, the equation zz1(x,t) = zzi(x), which can be rewritten $y \oplus rot(y, ri) = c_1 \oplus ci \oplus rot(t, r_1) \oplus rot(OP_C, r1) \oplus rot(OP_C, ri)$ provides some partial information on OP_C because $y \oplus rot(y, r_i)$ is at least partially known, and even entirely known if ri = 0 (it is then equal to 0).
- if an event of the form zz1(x,t) = zzi(x) occurs for a particular (x,t) value, then equation zz1(x',t) = zzi(x') still holds for any x' value such that the corresponding $y' = E_K(x' \oplus OP_C)$ satisfies equation $y' \oplus rot(y', r_i) = y \oplus rot(y, r_i)$. Thus if ri = 0, equality zz1(x,t) = zzi(x) for all x values and this particular t value.

The c1 to c5 choices suggested in the **f1**-**f5*** specification were made as to avoid any weakness that might result from events of the form zz1(x,t) = zzi(x). Due to the facts that:

- c1 is an even 128-bit word,
- t is obtained by repeating the 64-bit word SQN||AMF twice, so that t is an even word,
- all the other ci constants are odd,
- rotations do not affect the parity of any word,

the parity of $E_K^{-1}(zz1 \oplus OP_C) = y \oplus c_1 \oplus rot(t \oplus OP_C, r_1)$ is equal to the parity of $y \oplus OP_C$, whereas the parity of $E_K^{-1}(zzi \oplus OP_C) = c_i \oplus rot(y \oplus OP_C, r_i)$ is the inverse of the parity of $y \oplus OP_C$, so that the event zz1(x,t) = zzi(x) can never occur.

So in summary the criteria on the c1 to c5 and r1 to r5 introduced in the **f1**-**f5*** specification seem to ensure an effective separation of the **f1**-**f1*** functions from the **f2**-**f5*** functions.

10.3.3 Investigation of forgery or distinguishing attacks with 2^{64} queries

This section investigates properties allowing to distinguish the f1-f5* functions from ideal independent random functions of their inputs. The trivial birthday based distinguishers allowing to guess that f1-f5* are derived from permutations, not only functions, with about 2^{64} known input values is not considered here.

Several "attacks" against one single function (namely f1 or f1*) or against combinations of several functions requiring about 2^{64} queries are identified. Those "attacks" are described in this section. The presentation uses the abbreviated notation introduced in section 10.3.2.

10.3.3.1 An internal collision attack against f1 (or f1*)

Not surprisingly, the well-known CBC MAC internal collisions attack is applicable to **f1**:

Consider a set of about 2^{64} (x,t) pairs such that both the x and t values are pairwise distinct, and the corresponding z1 64-bit **f1** outputs. With a large probability, there exist two (x,t) pairs (x',t') and (x",t") such that the two corresponding 128-bit output words zz'1 and zz"1 collide.

Such a collision occurs iff

$$y' \oplus c_1 \oplus rot(t' \oplus OP_C, r_1) = y'' \oplus c_1 \oplus rot(t'' \oplus OP_C, r_1), \text{ i.e. iff}$$

$$y' \oplus rot(t' \oplus OP_C, r_1) = y'' \oplus rot(t'' \oplus OP_C, r_1).$$

If we XOR each of the t' and t" values with any $\delta t = \delta SQN||\delta AMF||\delta SQN||\delta AMF$ difference value, the above condition still holds, so that

$$f1(x', t' \oplus \delta t) = f1(x'', t'' \oplus \delta t) \quad \text{(i)}$$

This property can first be used to detect collisions, and then to actually forge new z1 values.

To detect a collision, one can test each of the approximately 2^{64} [(x',t');(x",t")] partial collisions such that z'1=z"1 (which can be efficiently enumerated), and then test whether property (i) holds for one or two randomly selected δ values. This allows to find a full collision on the entire zz1 output in about 2^{64} operations. Once such a collision [(x',t') ; (x",t")] has been detected, property (i) can be used, for any δ value, to forge $z1(x'', t'' \oplus \delta)$ based on $z1(x', t' \oplus \delta)$.

As said before, the above attack is only due to the fact that the **f1** function is essentially a standard CBC MAC. Since the attack requires about 2^{64} **f1** outputs corresponding to distinct RAND inputs, it has no practical significance: the use of a **f1** mode essentially equivalent to a standard CBC mode seems appropriate in the 3GPP context of operation. Moreover, we have not identified any simple modification of **f1** allowing to prevent the above internal collisions attack (or a variant) while avoiding the introduction of three invocations of E_K (instead of two) in each **f1** computation.

10.3.3.2 Forgery or distinguishing attacks against combinations of several modes

For some particular values of the r1 to r5 and c1 to c5 constants, there may also exist "attacks" against some combinations of several modes requiring about 2^{64} queries.

10.3.3.2.1 Attacks against combinations of f2-f5

Two of the one-block outputs zz2 to zz5 corresponding to two equal or distinct random challenges x' and x", denoted by zz'i and zz''j (where i and j are distinct values of the set {2, 3, 4, 5}) are equal iff:

$$rot(y', ri) \oplus rot(y'', rj) = ci \oplus cj \oplus rot(OP_C, ri) \oplus rot(OP_C, rj) \quad (a)$$

In the two following particular cases, there exists a simple attack requiring about 2^{64} queries

Case 1 : ri = rj (i.e. two of the constant rotations are equal)

The following forgery attack holds in case 1: given a set of 2^{64} x inputs and the corresponding zzi and zzj outputs. With a large probability there exist two input values x' and x" such that zz'i = zz"j. It is easy to see, using equation (a) and the fact that ri=rj, that one then also has zz"i=zz'j. In other words, if an adversary finds two x' and x" inputs such that the zz'i and zz"j are equal and obtains the zz"i value corresponding to the x" input, she can forge the zz'j value of zzj corresponding to the x' input. Such a phenomenon would be extremely unlikely to happen if zzi and zzj were the outputs of two independent permutations of x.

Case 2 : ri – rj = 64 mod 128 and ci ⊕ cj belongs to the Im[rot(.,ri) ⊕ rot(.,rj)] 64-dimensional vector subspace of $\{0,1\}^{128}$ (for instance, ri=0 and rj=64 and $c_i \oplus$ cj consists of two equal 64-bit halves).

The following distinguishing attack holds in case 2 : given a set of 2^{64} x inputs and the corresponding zzi and zzj outputs. With a large probablity, for one of these x values, one has: zzi = zzj. As a matter of fact this equality occurs iff rot(y ⊕ OP_C, ri) ⊕ rot(y ⊕ OP_C, rj)=ci ⊕ cj, and there are 2^{64} possible y ⊕ OP_C values satisfying that equation. Such an event would be extremely unlikely to occur if zzi and zzj were the outputs of two independent random permutations of x.

10.3.3.2.2 Attacks against combinations of f1-f1* and f2-f5*

A zz'1 value of the zz1 f1-f*1 output corresponding to a (x', t) input value is equal to a zz"i (i ∈ {2, 3, 4, 5}) one-block output of one of the f2-f5*computations corresponding to a x" input value iff

$$y' \oplus rot(y'', ri) = c1 \oplus ci \oplus rot(t, r_1) \oplus rot(OP_C, r1) \oplus rot(OP_C, ri) \quad (b)$$

As said before, the parities of the c1 to c5 constants prevent equation (b) from being satisfied if y'=y". In some particular cases, there nevertheless remain simple "attacks", requiring about 2^{64} queries.

Case 3 : ri = 0

The following forgery attack holds in case 3: given a set of 2^{64} (x,t) inputs and the corresponding zz1 and zzi outputs. With a large probability there exist x', t' and x" such that zz'1 (x',t') = zz"i(x"). It is easy to see, using equation (b) and the fact that ri=0, that one then also has zz"1(x",t') = zz'i(x'). This allows to forge zz'i(x') based on zz"1(x",t'). If z1 outputs are available instead of entire zz1 outputs, the above property still leads to a distinguishing "attack" requiring about 2^{64} queries.

Case 4 : r1 ∈ {0, 64} and there exist two distinct values i, j ∈ {2, 3, 4,.5} such that ci ⊕ cj consists of two equal 64-bit halves and ri = rj.

The following forgery attack holds in case 4: given a set of 2^{64} (x,t) inputs and the corresponding zz1 and zzi outputs. With a large probability there exist x', ti and x" blocks such that zz'1 (x',ti) = zz"i(x"). Let us now replace the ti expanded SQN||AMF sequence by the tj = ti ⊕ ci ⊕ cj sequence. It is easy to see, using equation (b), that equality zz'i(x',tj) = zz"j(x") also holds (in other words, a collision between a zz1 and a zzi value allows to predict a collision between a zz1 value and a zzj value). If z1 outputs are available instead of entire zz1 outputs, the above property still leads to a distinguishing "attack" requiring about 2^{64} queries.

10.3.3.3 Conclusion about the identified forgery or distinguishing attacks

All the attacks listed above require about 2^{64} queries, and can be considered highly impractical. None of these "attacks" is stronger than anticipated in the design criteria. It should also be noted that only the internal collision attack of section 10.3.3.1 and Case 3 of the previous section applies to the MILENAGE specifications.

10.4 Statistical evaluation

The algorithm "MILENAGE" has been designed such that the only cryptographically strong function used to process input depending on a key is the kernel function, which is the block cipher algorithm Rijndael for the example algorithm. All other operations used in the algorithm are XOR, cyclic rotation and splitting/concatenation of bit strings. These operations do not have any cryptographical properties such as confusion or diffusion. Both the designers and evaluators of MILENAGE were thus thoroughly convinced that all statistical tests which are to be performed on MILENAGE only yield results about the underlying kernel function. Since it was not the intention of the task force to evaluate Rijndael, statistical tests have not been performed as a consequence. Care has been taken on the design that the full entropy of the kernel function's outputs is mapped to the outputs of the algorithm (MAC, RES, keys).

10.5 Published attacks on Rijndael

This section gives a summary of known and published attacks against reduced variants of Rijndael as described in the AES report ref.[12].

The Rijndael specification describes a truncated differential attack on 4, 5, and 6 round variants of Rijndael [15], based on a 3 round distinguisher of Rijndael. This attack is called the "Square" attack, named after the cipher on which the attack was first mounted.

In Ref. [16], truncated differentials are used to construct a different distinguisher on 4 rounds, based on the experimentally confirmed existence of collisions between some partial functions induced by the cipher. This distinguisher leads to a collision attack on 7 round variants of Rijndael.

The other papers that present attacks on variants of Rijndael build directly on the Square attack. In Ref. [17], the Square attack is extended to 7 round variants of Rijndael by guessing an extra round of subkeys. Table 1 indicates the results for the 192 and 256-bit key sizes, where the total number of operations remains below those required for exhaustive search. Similar attacks are described in Ref. [18].

Table 1: Summary of reported attacks related to Rijndael versions

Reference	Round (Key size)	Type of attack	Texts	Memory	Operations
[15]	4 5 6	Truncated Diff Truncated Diff Truncated Diff.	2^9 2^{11} 2^{32}	small small $7 \cdot 2^{32}$	2^9 2^{40} 2^{72}
[18]	6 7 (192) 7 (256) 7 8 (256) 9 (256)	Truncated Diff. Truncated Diff. Truncated Diff Truncated Diff Truncated Diff Related Key	$6 \cdot 2^{32}$ $19 \cdot 2^{32}$ $21 \cdot 2^{32}$ $2^{128} - 2^{119}$ $2^{128} - 2^{119}$ 2^{77}	$7 \cdot 2^{32}$ $7 \cdot 2^{32}$ $7 \cdot 2^{32}$ 2^{61} 2^{101} NA	2^{44} 2^{155} 2^{172} 2^{120} 2^{204} 2^{224}
[17]	7 (192) 7 (256)	Truncated Diff Truncated Diff	2^{32} 2^{32} 2^{32}	$7 \cdot 2^{32}$ $7 \cdot 2^{32}$ $7 \cdot 2^{32}$	2^{184} 2^{200} 2^{140}
[16]	7 (192, 256)	Truncated Diff			

It should be noted that many of the reported attacks shown in table 1 are related to versions of Rijndael with larger keylength than used in MILENAGE.

The attacks reported in ref. [18] are improved by a partial summing technique that reduces the number of operations. The partial summing technique is also combined with a technique for trading off operations for information, yielding attacks on 7 and 8 round variants that require almost the entire codebook. The same paper also presents a related key attack on a 9 round variant with 256-bit keys. This attack requires not only encryptions of chosen plaintexts under the secret key, but also encryptions under 255 other keys that are related to the secret key in a manner chosen by the adversary.

10.6 Complexity evaluation

10.6.1 Complexity of draft Rijndael implementation

A straightforward implementation result in 8051 assembly language (no optimisations made) without taking advantage of Rijndael's rather particular linear component achieves the following results:

RAM: 48 bytes

ROM: about 1k

Cycles: 30.000

Max 6 applications (5 + 1 if OPc is computed on the card at each authentication) of Rijndael are considered: 180.000 cycles, which takes 55,38 ms @ 3.25 Mhz.

10.6.2 Estimate complexity of modes

Once Rijndael is implemented, considering the **f1-f5** construction of the modes, we obtain:

RAM:

RAND/E_K(RAND xor OPc)/IK 16 bytes

OPc: 16 bytes

SQN xor AK || AMF || MAC-A / CK / SQN_MS xor AK* || AMF* || MAC-S: 16 bytes

RES: 8 bytes

K: 16 bytes

Total: 72 bytes

ROM: 5*200 bytes = 1k

Cycles: overhead of 5*15.000 cycles = 75.000 cycles, which gives 23,07 ms overhead @ 3.25 Mhz.

10.6.3 Estimate of total MILENAGE

RAM: 120 bytes

ROM: 2k

Cycles: 255.000, which represents 78,46 ms @ 3.25 Mhz.

10.6.4 SPA/DPA, Timing attack countermeasures

See Section 10.8 for a wider discussion on these implementation attacks and the possibility for protection against such attacks.

RAM: additional 280 bytes XRAM

ROM: no significant change

Cycles: (180.000 + 75.000)*3 = 765.000 cyles, which takes 235 ms @ 3.25 Mhz.

10.6.5 Conclusion on algorithm complexity

The figures given in this section are rough estimates on complexity for a Rijndael kernel with and without full DPA/SPA/TA protections.

For an unmasked kernel and unmasked modes, the total estimate for the **f1** – **f5*** functions fits into 2k of ROM, using 120 bytes of RAM and executing under 80 ms @ 3.25 Mhz for our draft implementation. This lies well within the required 8k of ROM, 300 bytes of RAM and 500 ms execution time.

For a masked version of the kernel and modes, the total estimate for the **f1**-**f5** functions still meets the requirement of 8k of ROM, furthermore executing in less then 235 ms @ 3.25 Mhz for our draft implementation. However, masking techniques such as those described later on in this evaluation report require at least 256 additional bytes of XRAM, which sums up to slightly under 400 bytes of RAM for the kernel and modes."

Additional comment: The Task Force's expertise is in the design of the specified cryptographic functions. It is not familiar with (and doesn't really want to know) the fine detail of the wider context within the USIM. For example the Task Force do not know the USIM command set; the time involved in getting parameters into and out of the USIM; exactly how the cryptographic functions will be called in an operational environment; whether all results will be calculated at once or whether some results (e.g. CK) will be retrieved before others (e.g. IK) are calculated.

The Task Force have ensured to the best of its ability that the example algorithm set can be implemented efficiently, and that the size and performance parameters lie well within the requirements given in ref [2] to allow for other operational constraints that lie outside of its control.

10.7 External complexity evaluations

As part of the AES process there have been a number of investigations into physical realisations of Rijndael and these have included implementations for smartcards and protection against DPA. The results of three of these investigations are reproduced here.

Ref. [10] reports the following results for evaluating the Rijndael algorithm against DPA. The algorithm was implemented on a 32-bit ARM processor, both unmasked and masked against DPA. Security Cost is the ratio of Masked to Unmasked.

	Cycle count	RAM (bytes)	ROM (bytes)
Unmasked	7.086	52	1.756
Masked	13.867	326	2.393
Security Costs	1.96	6.27	1.36

The following results are given in ref. [13] and show the complexity and the performance of Rijndael implemented on a Toshiba T6N55 chip that contains a Z80 microprocessor and a coprocessor on a smart card. Internal means the size of required CRAM for the coprocessor's operations. External means the other work area.

	Total RAM	Internal RAM	External RAM	ROM (bytes)	Time (clocks)
Encrypt	34	32	2	700	25.494
Schedule	32	32	0	280	10,318
Total	66	64	2	980	35.812

The Rijndael Specification itself, ref. [15], gives the following table for three different implementations on the Intel 8051 microprocessor:

Number of Cycles	Code Length (bytes)
4065	768
3744	826
3168	1016

Ref. [15] also provides the following results for an implementation on a Motorola 68HC08 microprocessor

Number of Cycles	Required RAM(bytes)	Code Length (bytes)
8390	36	919

10.8 Evaluation of side channel attacks

In this section we will focus on the resistance of MILENAGE against side-channel attacks, or 'implementation attacks', i.e. attacks which make use of an additional (physical) information channel [7] through which secrets might leak. These additional channels include, but are not limited to, timing measurements [6], power consumption [8], [9], [10], [11]electromagnetic radiation, etc.

10.8.1 Evaluation of the kernel algorithm

Rijndael has been chosen as a kernel function for the 3GPP authentication algorithm. As most block ciphers, it uses simple linear operations such as bit shifts and rotations, exclusive or operations, as well as substitutions through a non linear 8 by 8 bit S-box.

Similarly to DES, Rijndael is vulnerable to SPA (Simple Power Analysis), DPA (Differential Power Analysis) and timing attacks. These attacks may however be thwarted by using adequate software and hardware protections on the mobile station. Note however that some techniques and/or methods for such protections might be subject to pending patents.

10.8.1.1 Timing Attacks

We refer to [6] for a complete description of timing attacks. In the case of Rijndael, the most vulnerable operation is the polynomial remaindering over $GF(2^8)$. In practice, this operation is implemented either by a 1-bit shift to the left, followed by a conditional exclusive-or with the modulus, or by a conditional table look-up. In both cases, the 'if' operation may leak information on intermediate results, and thus reveal parts of the secret key.

Therefore, care should be taken to ensure that this operation is implemented in constant time using operations that are the same, whatever the intermediate result. In particular, no 'if', 'case' or otherwise conditional 'jump' and 'call' instructions should be used. The MixColumn routine shall be implemented in an 'operation constant' way.

In this way, efficient protection against timing attacks is easily achieved.

10.8.1.2 Simple Power Analysis

Every block cipher is vulnerable to Simple Power Analysis. We refer to [9] and [10] for a complete description of this attack. In a very simple approach, an attacker monitors the power consumption curves of the device while executing cryptographic operations on different input data. The overall form of the curve reveals which actual data are being manipulated, thus leaking information on the secret key. This kind of attack may be thwarted by appropriate hardware countermeasures such as random noise generators or current scramblers on the device. Ad-hoc wait states or dummy instruction routines may be added in order to confuse the 'visual' analysis as much as possible.

It is a simple matter to provide good resistance against Simple Power Analysis.

10.8.1.3 Differential Power Analysis

We refer to [8], [9], [10] and [11] for a detailed description of Differential Power Analysis. Rijndael, as many other block ciphers is vulnerable to DPA. In particular, bytewise key-addition followed by a non-linear substitution table is generally enough to attack a block cipher successfully, meaning that the entire secret key may be recovered using a few hundred power curves. The attacker defines a selection function, which consists for example in one output bit of the non-linear S-box. The differential attack proceeds as follows:

- Guess the value of a given secret key byte.
- Using knowledge of the input (or output) data, compute the estimated one-bit result of the substitution of the input byte exclusive-ored with the key byte, according to the selection function.
- Sort the power curves according to the one-bit result and compute the mean of all curves in each of the two resulting sets.
- Subtract the two means and visualise the resulting curve.
- Repeat for all key guesses.

- For at most a few key guesses, the resulting curve will show a DPA 'peak' (whereas all other curves will be completely flat). These key guesses comprise the right key byte.
- Iterate on other key bytes and complete the whole secret key by exhaustive search as appropriate.

In the case of Rijndael, the secret key may be found byte by byte using a few hundred curves. Thus Rijndael is as vulnerable to DPA as DES.

We refer to [8], [10] and [11] for protections and countermeasures against this attack. All input data as well as the secret key may be masked by a random value throughout the cryptographic computations. Special care should be taken when handling the S-box, as this component is non-linear. Following [10], a boolean masked table S' can be defined in terms of S, and of the input and output masks r_in and r_out, such that $S'[x] = S[x \oplus r\_in] \oplus r\_out$. This masked table S' takes inputs that are masked with r_in and produces outputs that are masked with r_out. Thus the look-up operation is convertible into an operation which can 'handle' random boolean masks. All other components in Rijndael are linear and are thus not affected by the random masks.

The masks should be refreshed at every execution on different input data, in order to decorrelate every cryptographic operation from the actual value of the manipulated bytes. Together with hardware countermeasures against SPA, these protections should achieve a reasonable level of security against first order power attacks.

10.8.1.4 Other side channels

Other side channels such as carry bit propagation analysis or differential electromagnetic radiation or IPA (Inferential Power Analysis) are still under investigation in the open literature and are thus not addressed in this evaluation report.

10.8.2 Evaluation of the f1-f5 modes

The authentication and key derivation modes use Rijndael as the kernel function. As such, these modes are equally vulnerable to the different power and timing attacks. They can be efficiently protected by the same methods as for the kernel function itself. In particular, random boolean masking should also be applied to each function individually, not only to the underlying kernel.

10.8.2.1 Operator Constants (OP or OPc)

The use of secret operator constants enables to mask the real inputs to the kernel function, but side-channel attacks may still be applied to recover the exclusive or between these constants and the first round-key of the kernel. Subsequently, the second round-key may be derived and finally the original key may be recovered.

Thus, the operator constants have the effect of transforming a straightforward known plaintext attack into a slightly but not significantly more complex attack. Care should be taken to protect the operator constants as well as the secret key of the block cipher itself. This is achieved using above mentioned masking techniques.

10.8.2.2 Rotations and constants

These values are meant to be publicly defined and need not be protected in any way. However, if an operator chooses to diversify them, computations involving the rotations and constants should also be protected against timing and power attacks using previously mentioned techniques..

10.8.3 Conclusion on side channel attacks

Rijndael as a kernel function is vulnerable to side-channel attacks but may be efficiently protected against these attacks on the USIM. The modes do not add any security with respect to this kind of attacks and in case secret values such as an operator constant, rotations and other constants are used, it is strongly advised to protect them by random boolean masking together with the kernel function.

11 Conclusions

The MILENAGE algorithm set forms a complete set of cryptographic functions suitable for use in the 3GPP authentication framework. It constitutes a well-founded architecture based upon a highly trusted kernel and achieves all goals related to efficiency and security. The design supports the possibility for operator specific modifications by introduction of dedicated parameters and interchangeable modules.

The combination of the mathematical analysis carried out by the ETSI SAGE Task Force and related external evaluation results support the soundness of the design. This report includes a description of certain forgery attacks with complexity lower than the security level set by the 128 bits subscriber key, but it is explained that these attacks have been considered during the design process and are not considered to be feasible within the operational context of 3GPP.

The Task Force has specifically considered the threat of different side channel attacks and the report provides references to several techniques that can be used to ensure that MILENAGE can be implemented in a way that protect the USIM

Annex A (informative): Change history

Change history					
TSG SA #	**Version**	**CR**	**Tdoc SA**	**New Version**	**Subject/Comment**
SP_11	SAGE v 1.0	-	SP-010144	4.0.0	Approved as Release 4