ETSI SAGE Task Force for 3GPP
Authentication Function Algorithms

Date: 2000-11-22 ***VERSION 1.0***

Security Algorithms Group of Experts (SAGE);

General Report on the Design, Specification and Evaluation of The MILENAGE Algorithm Set: An Example Algorithm Set for the 3GPP Authentication and Key Generation Functions

Reference

Keywords

3GPP, security, SAGE, algorithm

Contents

Intellectual Property Rights 4
Foreword 4
1 Scope 5
2 References 5
3 Abbreviations 5
4 Structure of this report 6
5 Background to the 3GPP Authentication and Key Generation algorithms 7
6 SAGE 3GPP AF TF work plan 7
7 Outline of algorithm requirements specification 7
7.1 The authentication and key generation functions 7
7.2 Use of the algorithms on the AuC side 8
7.3 Use of the algorithms in the USIM 8
7.4 Use of the algorithms for resynchronisation in the USIM 8
7.5 Use of the algorithms for resynchronisation in the HLR/AuC 9
7.6 Implementation aspects 9
7.7 Generic requirements for 3GPP cryptographic functions and algorithms 9
7.8 Subsequent requirements on the authentication and key generation functions 10
8 Algorithms design 10
8.1 Design criteria 10
8.2 Chosen design for the framework 10
8.3 Analysis of the role of OP and OPc 11
8.4 Choice of kernel. 11
8.5 Design methodology 12
8.6 Specification and test data 12
9 Algorithm evaluation 12
9.1 Evaluation criteria 12
9.2 Mathematical Evaluation of the modes 13
9.3 Statistical Evaluation 13
9.4 Side channel attacks evaluation 13
9.5 Complexity evaluation 13
9.6 Evaluation report 13
10 Release of algorithm, specification and test data by SAGE 13
10.1 SAGE 3GPP AF TF approval for release 13
10.2 Publication of the algorithm set specification 14
10.3 Export of the algorithm set specification 14

Intellectual Property Rights

Foreword

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.

1 Scope

This report is a description of the work undertaken by an ETSI SAGE Task Force on the design of the Milenage Algorithm Set: an example set of 3GPP Authentication and Key Generation Functions.

The 3GPP Authentication and Key Generation Functions are not standardized. An example set of these algorithms has been produced on request from 3GPP with the intent that it shall be offered to the UMTS operators, to utilise instead of developing their own. An ETSI SAGE Task Force has carried out this work. The requirement specification from 3GPP SA3 stated that operator personalisation of the example set must be possible and that the basic kernel must be possible to replace.

The example set is based on the block cipher Rijndael, which at the time was one of the AES candidates and the specification describes how the 7 algorithms used in 3GPP authentication and key generation are scheduled around this basic kernel. The specification and associated test data for the example algorithm set is documented in three documents:

- A formal specification of both the modes and the example kernel [3]
- A detailed test data document, covering modes and the example kernel [4]
- A "black box" test data document [5]

A detailed summary of the evaluation is provided in a public evaluation report [6]

This report gives an overview of the overall work by the task force.

2 References

[1] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (3G TS 33.102 version 3.5.0)

[2] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements; (3G TS 33.105 version 3.4.0)

[3] 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.

[4] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
Document 2: Implementors' Test Data.

[5] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
Document 3: Design Conformance Test Data.

[6] ETSI SAGE 3GPP AF TF; Report on the design and evaluation of 3GPP Authentication and Key Generation Functions

[7] 3GPP TSG SA WG3 liaison statement to SAGE (S3-000089); Authentication algorithm for 3GPP

3 Abbreviations

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

3GPP	3rd Generation Partnership Project

AES	Advanced Encryption Standard
AMF	Authentication Management Field
AK	Anonymity key
AuC	Authentication Centre
AUTS	Re-synchronisation Token
CK	Cipher Key
DPA	Differential Power Analysis
$E(X)_K$	Encryption of X under key K
IK	Integrity Key
K	Subscriber key
MAC	Message Authentication Code
MAC-A	Network Authentication Code
MAC-S	Resynchronisation Authentication Code
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****.
OFB	Output Feedback
RAND	Random Challenge
RES	Response to Challenge
RNC	Radio Network Controller
SAGE	Security Algorithms Group of Experts
SAGE 3GPP AF TF	SAGE Task Force for the design of the 3GPP Authentication and Key Agreement Functions
SQN	Sequence Number
SPA	Simple Power Analysis
TA	Timing Attack
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
USIM	User Services Identity Module

4 Structure of this report

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

- clause 5 provides background information on the Authentication and Key Generation algorithms;
- clause 6 provides an outline of the work plan adopted by the SAGE Task Force to design and evaluate the example algorithm set and to produce the associated test data for release to 3GPP;
- clause 7 consists of a summary of the main points in the algorithm requirements specification produced by 3GPP TSG SA ;
- clause 8 describes how the SAGE Task Force designed the algorithm and produced the specification and associated test data;
- clause 9 gives an overview of the evaluation work carried out by the SAGE Task Force and the conclusions of the evaluations;
- clause 10 gives statements on the task force procedure for approval and release of the specification and the considerations for publication and export control.

5 Background to the 3GPP Authentication and Key Generation algorithms

Within the mobile communication system UMTS specified by 3GPP there is a need to provide security features. These security features are realised with the use of cryptographic functions and algorithms. In total 3GPP identified the need for 9 cryptographic algorithms and functions (ref. [1]). Two of these, f8 and f9, for cipher and integrity protection of the 3GPP radio interface have already been developed and are now part of the 3GPP standard specifications.

It was decided that the algorithms for authentication and key generation should not be standardised as they can well be proprietary to each operator and by his own choice (just like in GSM). The context for these algorithms, called f1, f1*, f2, f3, f4, f5, f5*, are described in ref. [1]. The generic requirements for these algorithms are specified in ref. [2].

It was discussed in 3GPP SA 3 if an example set of these algorithms should be produced and offered to the UMTS operators, to utilise instead of developing their own. A need for such an example set was identified with the additional requirement that operators should have a means to personalise their own algorithms. ETSI SAGE was asked to design the algorithms. To carry out this work SAGE set up a Task Force (SAGE 3GPP AF TF) based on SAGE and enlarged with cryptographers form UMTS manufacturers.

6 SAGE 3GPP AF TF work plan

The workplan for 3GPP authentication example algorithms was approved by 3GPP in July 2000. The SAGE 3GPP AF TF formally started work in August 2000 as an ETSI Task Force. This SAGE 3GPP AF TF consisted of the regular SAGE members, and three 3GPP manufacturers (Gemplus, Mitsubishi and Nokia).

The work was funded by 3GPP. The total resource budget for the SAGE 3GPP AF TF work was 16.75 man months.

The work was divided into five main tasks:

- Project Management;
- Design (approximately 14% of the budget)
- Evaluation (the major task, approximately 60% of the budget)
- Specification testing (approximately 20% of the budget)
- Liaison and publication activities

The design of the algorithm example set and a complete set of specification documents should be finalised in November 2000.

The work should be reported in 5 deliverables according to the requirements from 3GPP TSG SA.

- a short public report on the design and evaluation work (***this document)***.
- formal specification of both the modes and the example kernel [3]
- two test data reports, covering modes and the example kernel (detailed test results in [4]; black box test data in [5]
- a summary of the evaluation results in a public report [6]

The results of the evaluations was to be approved and agreed by the whole group before the final algorithms specifications were released to 3GPP.

7 Outline of algorithm requirements specification

The requirements for the authentication and key generation functions were specified by 3GPP TSG SA in: 3rd Generation Partnership Project: technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements (3G TS 33.105 version 3.1.0) [2]

7.1 The authentication and key generation functions

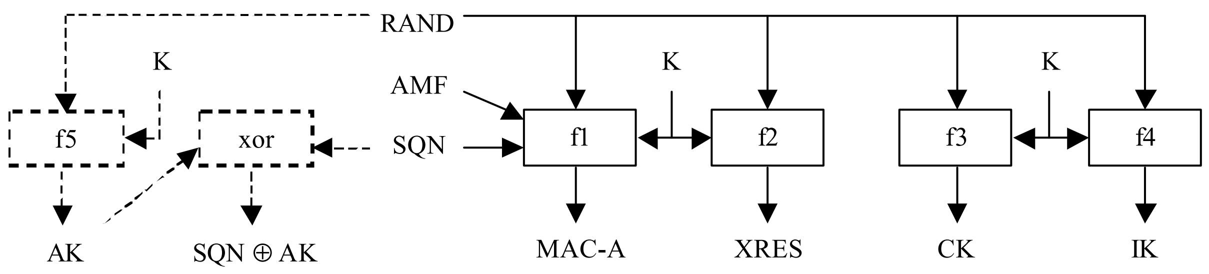
The mechanism for authentication and key agreement described in [1] requires the following cryptographic functions:

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.

Regarding f0, the random generation function, it was agreed with 3GPP SA3 that an example for this function should not be proposed by the Task Force.

For each of the algorithms f1 to f5* there is a general requirement that it shall be computationally infeasible to derive K from knowledge of input(s) and output.

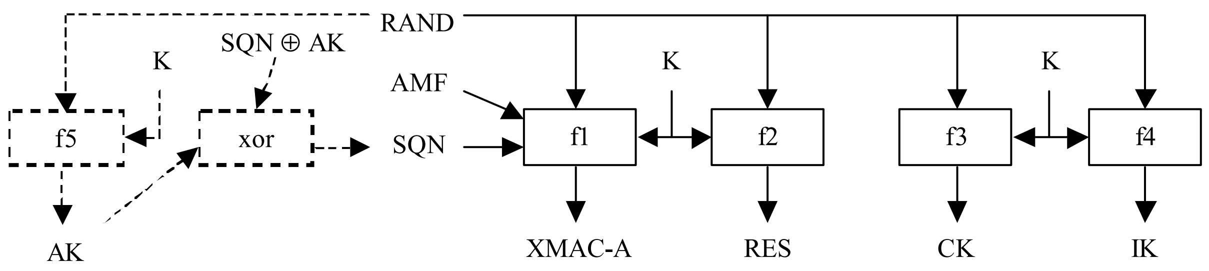
7.2 Use of the algorithms on the AuC side



AUTN = SQN [⊕ AK] || AMF || MAC-A
Quintet = (RAND, XRES, CK, IK, AUTN)

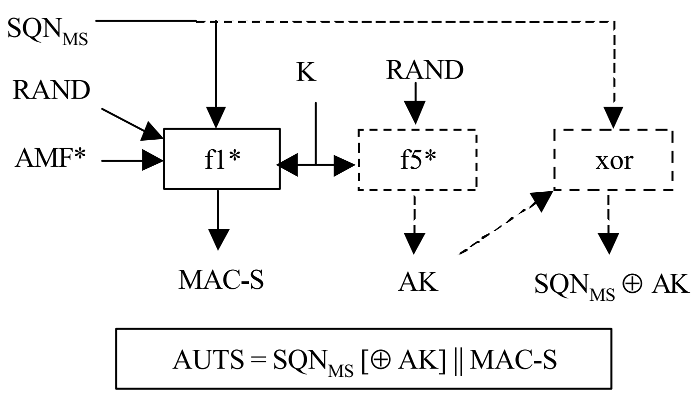
This figure describes the generation of the authentication and key generation values in the HLR/AuC

7.3 Use of the algorithms in the USIM



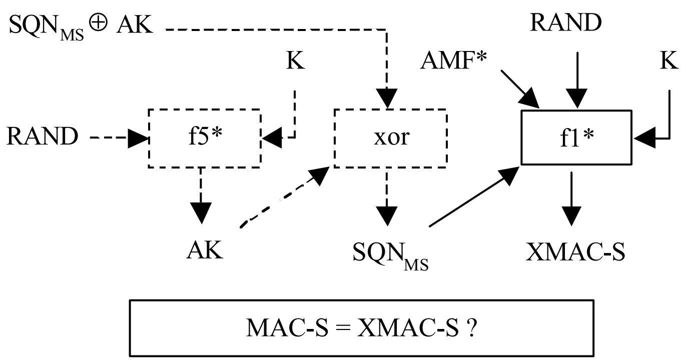
This figure describes the generation of the authentication and key generation values in the USIM

7.4 Use of the algorithms for resynchronisation in the USIM



This figure describes the generation of the re-synchronisation token, AUTS, in the USIM.

7.5 Use of the algorithms for resynchronisation in the HLR/AuC



This figure describes the control of the re-synchronisation token in the USIM

7.6 Implementation aspects

All the functions f1 to f5* shall be designed so that they can be implemented on an IC card equipped with a 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.

7.7 Generic requirements for 3GPP cryptographic functions and algorithms

In section 4 of [2] generic requirements are given for all 3GPP cryptographic functions and algorithms. These are summarized below (in Italics).

Resilience

The functions should be designed with a view to its 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.

World-wide availability and use

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 which 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 which embody such algorithms should be exportable under the conditions of the Wassenaar Arrangement*

7.8 Subsequent requirements on the authentication and key generation functions

In a liaison statement to SAGE the 3GPP Security Group TSG SA WG3 presented some further requirements on the authentication functions [7]:

“The basic principles of the algorithm design, as agreed by S3, are listed below:

- It is required that the algorithm fulfil the requirements specified in 3G TS 33.105.
- It is required that controlled personalisation of the algorithm is possible based on an operator variant algorithm configuration field of at least 128 bits.
- It is desirable that the algorithm is designed around a replaceable kernel function to provide an additional degree of variety.
- If an algorithm is to be designed around a kernel function, then it is required that one specific kernel function be provided.
- If an algorithm is to be designed around a kernel function, then it is desirable that a list of suitable alternative kernel functions be provided.
- If an algorithm is to be designed around a kernel function, then it is desirable that standard / publicly available algorithms may be used to implement the kernel function. However, the type or types of kernel function that could be supported is left to SAGE.

It is required that the algorithm lends itself to implementations which are resistant to Simple Power Analysis, Differential Power Analysis and other ‘side-channel’ attacks as appropriate when implemented on a USIM. It is acknowledged that SAGE may need to consult with smart card experts in order to be able to address this requirement.”

8 Algorithms design

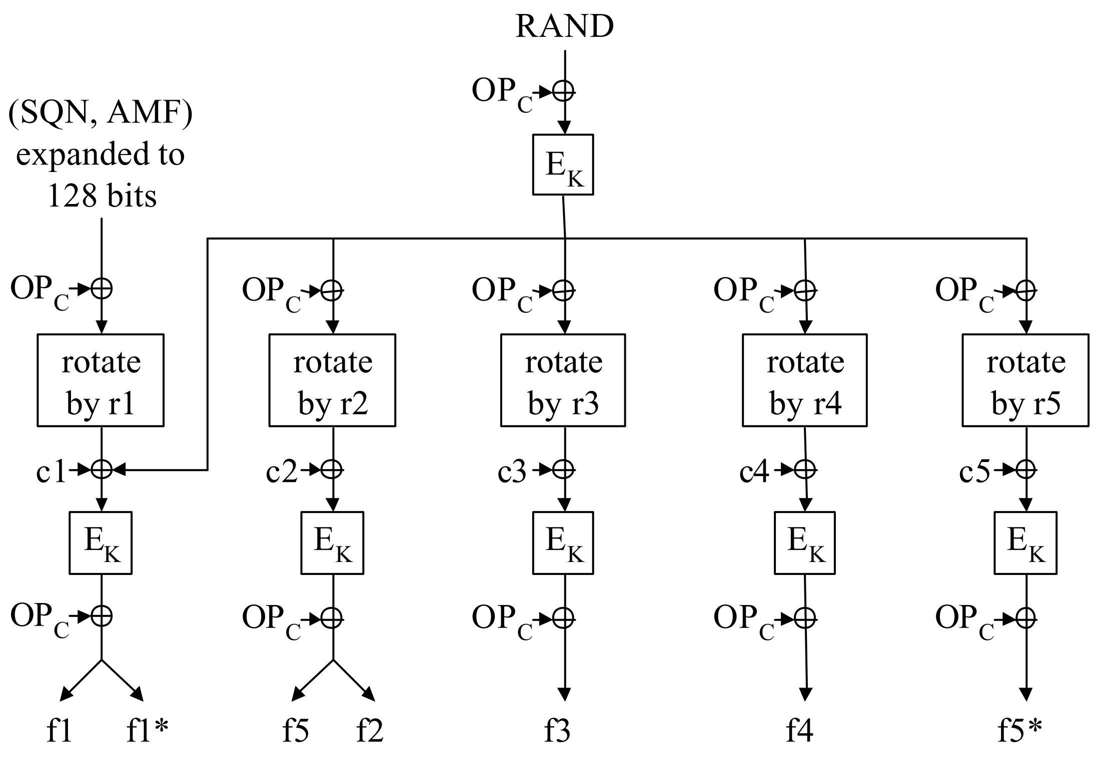
Based on the requirements and fixed starting points SAGE 3GPP AF TF established the following essential design criteria.

8.1 Design 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.
2. It should be practically impossible to determine any part of the secret key K, or the operator variant algorithm 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 require approximately 2^{128} operations) or less.
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.

8.2 Chosen design for the framework

The detailed specifications of the 3GPP algorithms are found in ref. [3]. The following diagram shows the framework for the functions ***f1, f1*, f2, f3, f4, f5*** and ***f5**** using the kernel function denoted E_K.



Definition of ***f1, f1*, f2, f3, f4, f5*** and ***f5****

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

$$OP_C = OP \oplus E(OP)_K$$

It is assumed and recommended in [3] that OPc is calculated outside the USIM cards and then stored in each card as an individual value. This will allow OP to be better protected than if OP had been stored in every card.

r1, …, r5 are five fixed rotation constants and c1, .., c5 are five fixed addition constants defined in [2].

8.3 Analysis of the role of OP and OPc

The 128-bit value OP is the Operator Variant Algorithm Configuration Field, which the Task Force was asked to include to provide separation between the functionality of the algorithms when used by different operators.
It is left to each operator to select a value for OP.

The algorithm set is designed to be secure whether or not OP is publicly known; however, operators may see some advantage in keeping their value of OP secret as a secret OP is one more hurdle in the attacker's path.

It should be difficult for someone who has discovered even a large number of (OPc, K) pairs to deduce OP. That means that the OPc associated with any other value of K will be unknown, which may make it (slightly) harder to mount some kinds of cryptanalytic and forgery attacks.

It is more likely that OP can be kept secret if OP is not stored on the USIM, as it then only takes one USIM to be reverse engineered for OP to be discovered and published. Hence the task force recommends that OPc is calculated off USIM.

8.4 Choice of kernel.

Because of the short time scales it was decided to base the algorithm example set on an existing algorithm which had undergone sufficient analysis to serve as a kernel for the algorithm set so as to obviate the need for extensive cryptographic analyses and testing. A 128 bit input/output block cipher with a 128 bit key was chosen as kernel as this was considered to be a type of kernel which could easily be substituted if any operator wanted to use the framework with another algorithm.

Document [3] describes the use of the Rijndael algorithm in encryption mode for the kernel function E_K, but this choice could be replaced by any suitable 128-bit keyed function employing a 128 bit key. Rijndael was chosen as being then one of the five remaining AES candidates, was well studied, could be efficiently implemented in S/W or H/W and was available IPR-free.

Ref [6] gives general requirements on the kernel which should be considered in case of replacement.

8.5 Design methodology

The algorithms were designed using a phased iterative and interactive approach. The design process is summarised below.

- **Phase 1:** The starting points for the algorithms and design criteria were agreed. The design team then produced a first design proposal for a framework algorithm set including different options for the inclusion of OP. A block cipher was proposed as kernel with Rijndael as prime candidate

- **Phase 2:** The results of first evaluations were discussed. Choice of block cipher as kernel versus keyed hash was evaluated. Use of OP as protection against DPA was discussed. The counter mode was adopted instead of OFB. Based on these results, the design team revised the design to a second design proposal for the algorithm set. 3GPP SA3 agreed to limit RES to 64 bits in the example and to change input to the synchronization token.

- **Phase 3:** The results of the second evaluation were discussed. The use of Rijndael and the design for the algorithm set was confirmed, except for some small details. Considering the extensive analyses which have been made on Rijndael it was decided that the normal range of statistical testing need not be performed. Evaluation of complexity of implementation started.

- **Phase 4:** Details were fixed after mathematical evaluation of modes. The method for deriving OPc from OP was changed. Resistance against side channel attacks was decided to be left to implementers, with references to methods discussed in connection with Rijndael as candidate to AES. The specification documents were drafted and two parties independently carried out specification testing to check the correctness and completeness of the specification and the accompanying C-code.

- **Phase 5:** After the final review the specifications were confirmed. A summary report of the evaluation undertaken by the task force was produced and agreed [6], as well as this report.

8.6 Specification and test data

The algorithm specification and associated test data are documented in [3], [4] and [5]. An annex in [3] consists of example program listings of the algorithm set in ‘C’.

Document [4] provides design conformance test data designed to help verify implementations of the algorithms. The document identifies intermediate points in the algorithms where test data is provided. Then it gives input, internal and output parameters at these points, and provides different sets of test data listings.
Document [5] is informative and provides test data designed to help verify the correct functioning of the algorithms seen as a ‘black box’. The document identifies the input and output interfaces and provides a number of test sets for the different modes of operation of the algorithms.

Two parties not directly involved in the design and evaluation teams also evaluated the adequacy of the specification. To this end, these parties made independent simulations of the algorithm from the specification and confirmed these against the test data.

9 Algorithm evaluation

9.1 Evaluation criteria

The algorithm requirements as summarised in section 7 and design criteria as listed in section 8.1 leads to evaluation criteria for the mathematical evaluation and statistical evaluations. Due to the fact that the Rijndael block cipher has undergone an extensive analysis during the AES process, the task force performed no real cryptanalysis of Rijndael, but rather focused on the strength of the constructions for deriving the ***f1*** to ***f5**** modes from a strong 128-bit block cipher.

9.2 Mathematical Evaluation of the modes

The mathematical evaluation concentrated on verifying the strength of the f1-f5 construction, under the assumption that the underlying kernel is a strong block cipher.

The main criteria investigated were:

- The strength of each algorithm, considered individually (resilience of key and subsequent outputs)
- The independence between algorithms (one algorithm's strength is not harmed by knowledge of input/outputs for other algorithms)

For further details on these investigations, see [6].

9.3 Statistical Evaluation

No statistical tests were performed on the kernel, given Rijndael can be trusted to be sufficiently tested and secure through the AES process. Statistical tests on MILENAGE were considered to 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.

9.4 Side channel attacks evaluation

In the design process it was concluded not to be feasible to design a general algorithm framework that by itself would not be vulnerable to side channel attacks. Rijndael, as 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, OPc, in the USIM cards can only play a limited role in protecting against these kinds of attacks. Hardware protection measures and masking techniques, as referenced in [6], need to be specifically implemented for protection. Also timing attacks (TA) may need implementation specific countermeasures. Rijndael as an AES candidate has been shown to readily lend itself to protection measures against side channel attacks.

9.5 Complexity evaluation

It is estimated that a non-optimized implementation of kernel and modes could be implemented with no more than 2kB ROM, 120B RAM and execute in less than 80ms. (One instance of the Rijndael cipher will execute in less than 10 ms at 3.25 MHz). Even with full protection measures against side channel attacks (SPA/DPA/TA) on Rijndael it is expected that performance will be well under the stated requirements [2].

9.6 Evaluation report

The evaluation report [6] is a summary of all results of the complete design and evaluation process. It provides the main conclusions of the evaluation work carried out by the task force..

10 Release of algorithm specification and test data by SAGE

10.1 SAGE 3GPP AF TF approval for release

Prior to release of the specification of the algorithm set the following approvals were gained.

All members of SAGE 3GPP AF TF stated that they were satisfied that the example algorithm set provides the high level of security for the authentication and key generation functions required by 3GPP.

- All members of SAGE 3GPP AF TF approved release of the algorithm set specifications and the accompanying documents to 3GPP.

10.2 Publication of the algorithm set specification

The SAGE 3GPP AF TF does not see from a security point of view any obstacles that would prevent publication of the algorithm set specifications.

10.3 Export of the algorithm set specification

The SAGE 3GPP AF TF does not see any obstacles that would prevent export of the algorithm set specifications or corresponding implementations in congruence with the requirements as given in [2].

ETSI/SAGE Specification | **Version: 1.0**
Date: 22nd November 2000

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

Document History		
V1.0	**22nd November 2000**	**Publication**

PREFACE

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

This document is the first of three, which between them form the entire specification of the example algorithms:

- 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.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
  Document 2: Implementors' Test Data.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
  Document 3: Design Conformance Test Data.

Blank Page

TABLE OF CONTENTS

0. THE NAME "MILENAGE" 7

1. OUTLINE OF THE DOCUMENT 7

2. INTRODUCTORY INFORMATION 7
   2.1. Introduction 7
   2.2. Notation 7
   2.3. List of Variables 8
   2.4. Algorithm Inputs and Outputs 9

3. THE ALGORITHM FRAMEWORK AND THE SPECIFIC EXAMPLE ALGORITHMS 10

4. DEFINITION OF THE EXAMPLE ALGORITHMS 11
   4.1. Algorithm Framework 11
   4.2. Specific Example Algorithms 12

5. IMPLEMENTATION CONSIDERATIONS 12
   5.1. OP_C computed on or off the USIM? 12
   5.2. Customising the choice of block cipher 13
   5.3. Further customisation 13
   5.4. Resistance to side channel attacks 13

ANNEX 1 Figure of the Algorithms 14

ANNEX 2 Specification of the Block Cipher Algorithm Rijndael 15

ANNEX 3 Simulation Program Listing — Byte Oriented 20

ANNEX 4 Rijndael Listing — 32-Bit Word Oriented 29

REFERENCES

[1] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (3G TS 33.102 version 3.5.0)

[2] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements; (3G TS 33.105 version 3.4.0)

[3] 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 (this document).

[4] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
Document 2: Implementors' Test Data.

[5] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
Document 3: Design Conformance Test Data.

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

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

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

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

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

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

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

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

0. THE NAME "MILENAGE"

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

1. OUTLINE OF THE DOCUMENT

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

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

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

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

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

2. INTRODUCTORY INFORMATION

2.1. Introduction

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

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

2.2. Notation

2.2.1. Radix

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

2.2.2. Conventions

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

<center>&lt;*variable*> = <*expression*&gt;</center>

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

$$x = x + y + 3$$

means

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

2.2.3. Bit/Byte ordering

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

2.2.4. List of Symbols

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

2.3. List of Variables

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

RAND	a 128-bit random challenge that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
RES	a 64-bit signed response that is the output of the function ***f2***.
SQN	a 48-bit sequence number that is an input to either of the functions ***f1*** and ***f1****. (For ***f1**** this input is more precisely called SQN_{MS}.)
TEMP	a 128-bit value used within the computation of the functions.

2.4. Algorithm Inputs and Outputs

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

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

Table 1. inputs to ***f1*** and ***f1****

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

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

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

Table 3. ***f1*** output

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

Table 4. ***f1**** output

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

Table 5. ***f2*** output

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

Table 6. ***f3*** output

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

Table 7. ***f4*** output

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

Table 8. ***f5*** output

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

Table 9. ***f5**** output

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

3. THE ALGORITHM FRAMEWORK AND THE SPECIFIC EXAMPLE ALGORITHMS

The example algorithm set makes use of the following components:

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

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

4. DEFINITION OF THE EXAMPLE ALGORITHMS

4.1. Algorithm Framework

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

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

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

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

A 128-bit value **IN1** is constructed as follows:

$$\mathbf{IN1}[0] \,..\, \mathbf{IN1}[47] = \mathbf{SQN}[0] \,..\, \mathbf{SQN}[47]$$

$$\mathbf{IN1}[48] \,..\, \mathbf{IN1}[63] = \mathbf{AMF}[0] \,..\, \mathbf{AMF}[15]$$

$$\mathbf{IN1}[64] \,..\, \mathbf{IN1}[111] = \mathbf{SQN}[0] \,..\, \mathbf{SQN}[47]$$

$$\mathbf{IN1}[112] \,..\, \mathbf{IN1}[127] = \mathbf{AMF}[0] \,..\, \mathbf{AMF}[15]$$

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

$$\mathbf{c1}[i] = 0 \text{ for } 0 \le i \le 127$$

$$\mathbf{c2}[i] = 0 \text{ for } 0 \le i \le 127, \text{ except that } \mathbf{c2}[127] = 1$$

$$\mathbf{c3}[i] = 0 \text{ for } 0 \le i \le 127, \text{ except that } \mathbf{c3}[126] = 1$$

$$\mathbf{c4}[i] = 0 \text{ for } 0 \le i \le 127, \text{ except that } \mathbf{c4}[125] = 1$$

$$\mathbf{c5}[i] = 0 \text{ for } 0 \le i \le 127, \text{ except that } \mathbf{c5}[124] = 1$$

Five integers **r1**, **r2**, **r3**, **r4**, **r5** are defined as follows:

$$\mathbf{r1} = 64;\ \mathbf{r2} = 0;\ \mathbf{r3} = 32;\ \mathbf{r4} = 64;\ \mathbf{r5} = 96$$

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

$$\mathbf{OUT1} = \mathrm{E}[\mathbf{TEMP} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1}]_{\mathbf{K}} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT2} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r2}) \oplus \mathbf{c2}]_{\mathbf{K}} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT3} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r3}) \oplus \mathbf{c3}]_{\mathbf{K}} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT4} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r4}) \oplus \mathbf{c4}]_{\mathbf{K}} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT5} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r5}) \oplus \mathbf{c5}]_{\mathbf{K}} \oplus \mathbf{OP_C}$$

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

Output of ***f1*** = MAC-A, where MAC-A[0] .. MAC-A[63] = **OUT1**[0] .. **OUT1**[63]

Output of ***f1**** = MAC-S, where MAC-S[0] .. MAC-S[63] = **OUT1**[64] .. **OUT1**[127]

Output of ***f2*** = RES, where RES[0] .. RES[63] = **OUT2**[64] .. **OUT2**[127]

Output of ***f3*** = CK, where CK[0] .. CK[127] = **OUT3**[0] .. **OUT3**[127]

Output of ***f4*** = IK, where IK[0] .. IK[127] = **OUT4**[0] .. **OUT4**[127]

Output of ***f5*** = AK, where AK[0] .. AK[47] = **OUT2**[0] .. **OUT2**[47]

Output of ***f5**** = AK, where AK[0] .. AK[47] = **OUT5**[0] .. **OUT5**[47]

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

4.2. Specific Example Algorithms

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

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

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

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

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

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

5. IMPLEMENTATION CONSIDERATIONS

5.1. $\mathbf{OP_C}$ computed on or off the USIM?

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

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

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

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

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

- The complexity of the algorithms run on the USIM is reduced.

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

5.2. Customising the choice of block cipher

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

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

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

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

5.3. Further customisation

If an operator wishes to customise the algorithms still further, a simple approach is to select different values for the constants **c1**–**c5** and **r1**–**r5**. If this is done, the pairs (**ci**, **ri**) must all be different. It must not be the case that both **ci** = **cj** and **ri** = **rj** for **i** ≠ **j**. For instance, it must not be the case that both **c2** = **c4** and **r2** = **r4**. Additionally it is recommended that the following restrictions are applied:

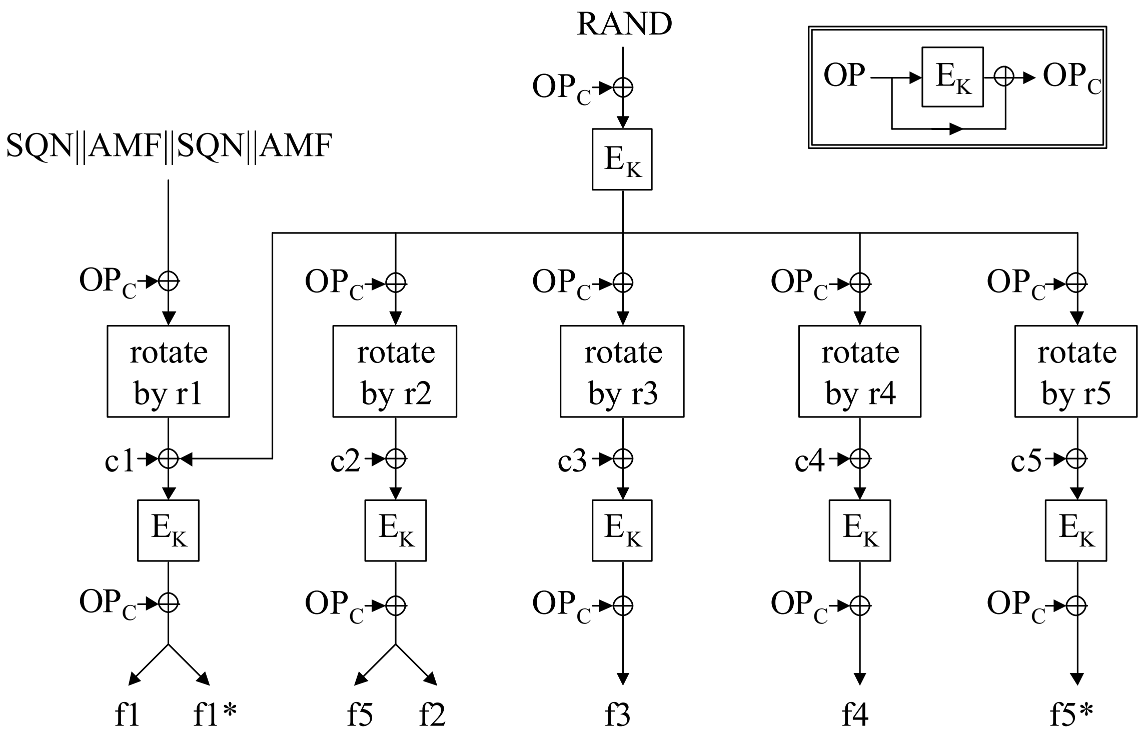
- **c1** has even parity. (A 128-bit value has even parity if the number of '1's in its binary representation is even.)

- **c2**–**c5** all have odd parity.

5.4. Resistance to side channel attacks

When these algorithms are implemented on a USIM, consideration should be given to protecting them against side channel attacks such as differential power analysis (DPA).
[8, 9, 10, 11, 12, 13] may be useful references.

ANNEX 1 Figure of the Algorithms



ANNEX 2
Specification of the Block Cipher Algorithm Rijndael

A2.1 Introduction

This section provides a specification for the example kernel function. The block cipher used is Rijndael. The complete specification of Rijndael is given elsewhere [6]. To quote from [6]:

> "Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 or 256 bits."

For 3GPP purpose, Rijndael is used only in encryption mode and has the block and key length both set to 128 bits. For the rest of this section, when we refer to Rijndael we mean Rijndael with 128-bit block and key length. This document describes a simple byte oriented implementation of the encryption mode of Rijndael. Readers wishing more detail on the design of the cipher or implementation speed-ups are referred to the original document [6].

A2.2 The State and External Interfaces of Rijndael

Rijndael is composed of a series of rounds that transform the input into the output. An intermediate result is called the **State**. The **State** can be pictured as a 4x4 rectangular array of bytes (128 bits in total). The Cipher Key is similarly pictured as a 4x4 rectangular array. These representations are illustrated in Figure 1.

$\mathbf{a}_{0,0}$	$\mathbf{a}_{0,1}$	$\mathbf{a}_{0,2}$	$\mathbf{a}_{0,3}$
$\mathbf{a}_{1,0}$	$\mathbf{a}_{1,1}$	$\mathbf{a}_{1,2}$	$\mathbf{a}_{1,3}$
$\mathbf{a}_{2,0}$	$\mathbf{a}_{2,1}$	$\mathbf{a}_{2,2}$	$\mathbf{a}_{2,3}$
$\mathbf{a}_{3,0}$	$\mathbf{a}_{3,1}$	$\mathbf{a}_{3,2}$	$\mathbf{a}_{3,3}$

$\mathbf{k}_{0,0}$	$\mathbf{k}_{0,1}$	$\mathbf{k}_{0,2}$	$\mathbf{k}_{0,3}$
$\mathbf{k}_{1,0}$	$\mathbf{k}_{1,1}$	$\mathbf{k}_{1,2}$	$\mathbf{k}_{1,3}$
$\mathbf{k}_{2,0}$	$\mathbf{k}_{2,1}$	$\mathbf{k}_{2,2}$	$\mathbf{k}_{2,3}$
$\mathbf{k}_{3,0}$	$\mathbf{k}_{3,1}$	$\mathbf{k}_{3,2}$	$\mathbf{k}_{3,3}$

Figure 1: Example of State and Cipher Key layout.

Rijndael takes plaintext bytes $\mathbf{P}_0$, $\mathbf{P}_1$, …,$\mathbf{P}_{15}$ and key bytes $\mathbf{K}_0$, $\mathbf{K}_1$, …$\mathbf{K}_{15}$ as input and ciphertext bytes $\mathbf{C}_0$, $\mathbf{C}_1$, …, $\mathbf{C}_{15}$ as output. The plaintext bytes are mapped onto the state bytes in the order $\mathbf{a}_{0,0}$, $\mathbf{a}_{1,0}$, $\mathbf{a}_{2,0}$, $\mathbf{a}_{3,0}$, $\mathbf{a}_{0,1}$, $\mathbf{a}_{1,1}$, $\mathbf{a}_{2,1}$, $\mathbf{a}_{3,1}$,… and the key bytes in the order $\mathbf{k}_{0,0}$, $\mathbf{k}_{1,0}$, $\mathbf{k}_{2,0}$, $\mathbf{k}_{3,0}$, $\mathbf{k}_{0,1}$, $\mathbf{k}_{1,1}$, $\mathbf{k}_{2,1}$, $\mathbf{k}_{3,1}$,…. At the end of the cipher operation, the ciphertext is extracted from the **State** by taking the **State** bytes in the same order.

Hence if the one-dimensional index of a byte within a block is **n** and the two dimensional index is (**i, j**), we have:

$$\mathbf{i} = \mathbf{n} \bmod 4; \quad \mathbf{j} = \lfloor \mathbf{n}/4 \rfloor; \quad \mathbf{n} = \mathbf{i} + 4 * \mathbf{j}$$

A2.3 Internal Structure

Rijndael consists of the following operations:

- an initial Round Key addition
- 9 rounds, numbered 1-9, each consisting of
    - a byte substitution transformation

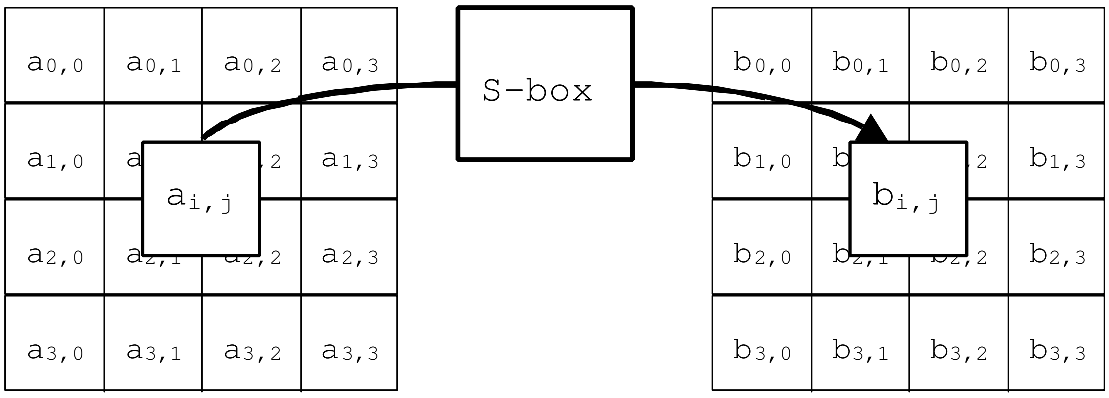
  - a shift row transformation
  - a mix column transformation
  - a Round Key addition
- A final round (round 10) consisting of
  - a byte substitution transformation
  - a shift row transformation
  - a Round Key addition

The component transformations and how the Round Keys are derived from the cipher keys are specified in the following subsections.

A2.4 The Byte Substitution Transformation

The byte substitution transformation is a non-linear byte substitution, operating on each of the **State** bytes independently. The substitution table (S-box) is given in section A2.9.

Figure 2 illustrates the effect of the byte substitution transformation on the **State**.



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,3}$
$a_{i,j}$
$a_{2,0}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
S-box
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,3}$
$b_{i,j}$
$b_{2,0}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Figure 2: Byte substitution acts on the individual bytes of the State.

So, for every element of **State,** we apply the transformation:

$$\mathbf{b_{i,j}} = \text{S-box}[\ \mathbf{a_{i,j}}\ ]$$

Where $\mathbf{a_{i,j}}$ is the initial value of the element in **State**,
and $\mathbf{b_{i,j}}$ is the output value of the element in **State**.

A2.5 The Shift Row Transformation

In the shift row transformation, the rows of the **State** are cyclically left shifted by different amounts. Row 0 is not shifted, row 1 is shifted by 1 byte, row 2 by 2 bytes and row 3 by 3 bytes.

Figure 3 illustrates the effect of the shift row transformation on the **State**.

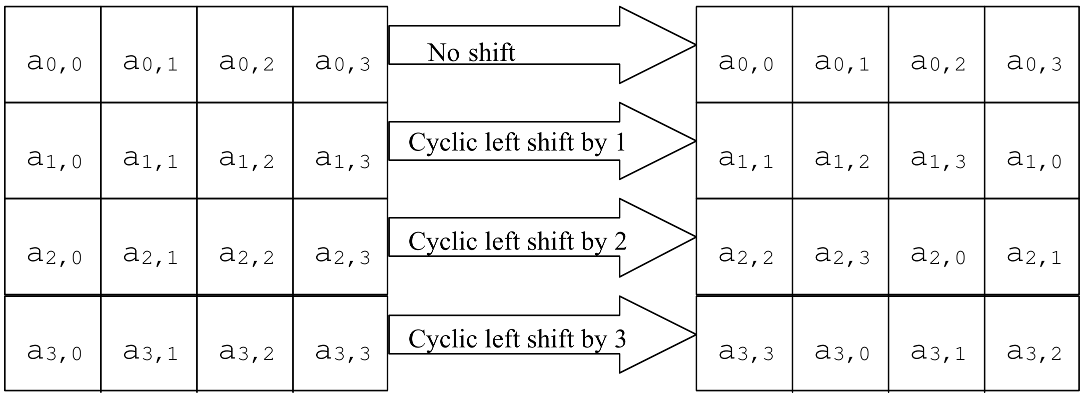


Figure 3: Shift Row operates on the rows of the State

A2.6 The Mix Column Transformation

The mix column transformation operates on each column of the **State** independently. For column `j`, we have

$$\mathbf{b_{0,j}} = T_2(\mathbf{a_{0,j}}) \oplus T_3(\mathbf{a_{1,j}}) \oplus \mathbf{a_{2,j}} \oplus \mathbf{a_{3,j}}$$
$$\mathbf{b_{1,j}} = \mathbf{a_{0,j}} \oplus T_2(\mathbf{a_{1,j}}) \oplus T_3(\mathbf{a_{2,j}}) \oplus \mathbf{a_{3,j}}$$
$$\mathbf{b_{2,j}} = \mathbf{a_{0,j}} \oplus \mathbf{a_{1,j}} \oplus T_2(\mathbf{a_{2,j}}) \oplus T_3(\mathbf{a_{3,j}})$$
$$\mathbf{b_{3,j}} = T_3(\mathbf{a_{0,j}}) \oplus \mathbf{a_{1,j}} \oplus \mathbf{a_{2,j}} \oplus T_2(\mathbf{a_{3,j}})$$

where:

	$T_2(\mathbf{a}) = 2*\mathbf{a}$	if $\mathbf{a} < 128$
or	$T_2(\mathbf{a}) = (2*\mathbf{a}) \oplus 283$	if $\mathbf{a} \geq 128$
and	$T_3(\mathbf{a}) = T_2(\mathbf{a}) \oplus \mathbf{a}$.	

For example:

If $\mathbf{a} = 63$ then $T_2(63) = 126$; $T_3(63) = T_2(63) \oplus 63 = 65$
If $\mathbf{a} = 143$ then $T_2(143) = 5$; $T_3(143) = T_2(143) \oplus 143 = 138$.

Figure 4 illustrates the effect of the mix column transformation on the **State**.

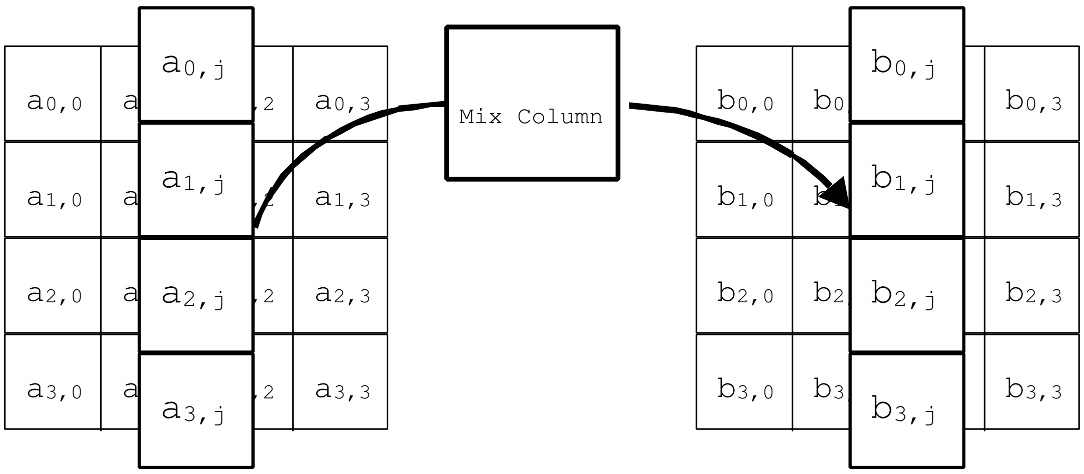


Figure 4: Mix Column operates on the columns of the State.

A2.7 The Round Key addition

In this operation, a Round Key is applied to the **State** by a simple bitwise exclusive-or. The Round Key is derived from the Cipher Key by means of the key schedule. The Round Key length is equal to the block length.

This transformation is illustrated in Figure 5.

$$\begin{array}{|c|c|c|c|}\hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline rk_{0,0} & rk_{0,1} & rk_{0,2} & rk_{0,3} \\ \hline rk_{1,0} & rk_{1,1} & rk_{1,2} & rk_{1,3} \\ \hline rk_{2,0} & rk_{2,1} & rk_{2,2} & rk_{2,3} \\ \hline rk_{3,0} & rk_{3,1} & rk_{3,2} & rk_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Figure 5: In the key addition the Round Key is bitwise XORed to the State.

So, for every element of **State**, we have:

$$\mathbf{b_{i,j}} = \mathbf{a_{i,j}} \oplus \mathbf{rk_{i,j}}$$

Where $\mathbf{a_{i,j}}$ is the initial value of the element in **State**,
$\mathbf{b_{i,j}}$ is the output value of the element in **State**.
and $\mathbf{rk_{i,j}}$ is the round key byte.

A2.8 Key schedule

Rijndael has 11 Round Keys, numbered 0-10, that are each 4x4 rectangular arrays of bytes. The Round Keys are derived from the Cipher Key by means of the key schedule. The initial Round Key (thought of as the zeroth Round Key) is formed directly from the cipher key. This zeroth Round Key is used unaltered for the initial key addition. The remaining Round Keys are used in the ten rounds. Each new round key is derived from the previous round key. Note: It is possible to run the key schedule round by round on an "as required" basis and so only use a total of 16 bytes to store the Round Key.

Let $\mathbf{rk_{r,i,j}}$ be the value of the **r**th Round Key at position (**i, j**) in the array and $\mathbf{k_{i,j}}$ be the cipher key loaded into a 4x4 array.

Intialisation: $\mathbf{rk_{0,i,j}} = \mathbf{k_{i,j}}$ for all **i** and **j**.

The other Round Keys (**r**=1 to 10 inclusive) are calculated from the previous one as follows. First the 0th column is constructed:

$$\mathbf{rk_{r,0,0}} = \mathbf{rk_{r-1,0,0}} \oplus \text{S-box}[\mathbf{rk_{r-1,1,3}}] \oplus \text{round\_const}[\mathbf{r}]$$
$$\mathbf{rk_{r,1,0}} = \mathbf{rk_{r-1,1,0}} \oplus \text{S-box}[\mathbf{rk_{r-1,2,3}}]$$
$$\mathbf{rk_{r,2,0}} = \mathbf{rk_{r-1,2,0}} \oplus \text{S-box}[\mathbf{rk_{r-1,3,3}}]$$
$$\mathbf{rk_{r,3,0}} = \mathbf{rk_{r-1,3,0}} \oplus \text{S-box}[\mathbf{rk_{r-1,0,3}}]$$

where round_const[1] = 1 and round_const[**r**] = T_2(round_const[**r**-1]), and S-box is the previously mentioned byte substitution.

Then the remaining three columns are constructed in turn from the corresponding column of the previous Round Key and the previous column of the current Round Key:

$\mathbf{rk}_{\mathbf{r,i,j}} = \mathbf{rk}_{\mathbf{r-1,i,j}} \oplus \mathbf{rk}_{\mathbf{r,i,j-1}}$ for **i**=0,1,2,3 and **j**=1,2,3.

Note: The ten round constants computed from the equations:

round_const[1] = 1
round_const[**r**] = T_2(round_const[**r**-1]) r=2,3…,10

are: 1, 2, 4, 8, 16, 32, 64, 128, 27, 54.

A2.9 The Rijndael S-box

```
Sbox[256] = {
 99,124,119,123,242,107,111,197, 48,  1,103, 43,254,215,171,118,
202,130,201,125,250, 89, 71,240,173,212,162,175,156,164,114,192,
183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
  4,199, 35,195, 24,150,  5,154,  7, 18,128,226,235, 39,178,117,
  9,131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,
 83,209,  0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,
208,239,170,251, 67, 77, 51,133, 69,249,  2,127, 80, 60,159,168,
 81,163, 64,143,146,157, 56,245,188,182,218, 33, 16,255,243,210,
 96,129, 79,220, 34, 42,144,136, 70,238,184, 20,222, 94, 11,219,
224, 50, 58, 10, 73,  6, 36, 92,194,211,172, 98,145,149,228,121,
231,200, 55,109,141,213, 78,169,108, 86,244,234,101,122,174,  8,
186,120, 37, 46, 28,166,180,198,232,221,116, 31, 75,189,139,138,
112, 62,181,102, 72,  3,246, 14, 97, 53, 87,185,134,193, 29,158,
225,248,152, 17,105,217,142,148,155, 30,135,233,206, 85, 40,223,
140,161,137, 13,191,230, 66,104, 65,153, 45, 15,176, 84,187, 22};
```

ANNEX 3
Simulation Program Listing — Byte Oriented

```
/*-------------------------------------------------------------------
 *          Example algorithms f1, f1*, f2, f3, f4, f5, f5*
 *-------------------------------------------------------------------
 *
 *  A sample implementation of the example 3GPP authentication and
 *  key agreement functions f1, f1*, f2, f3, f4, f5 and f5*.  This is
 *  a byte-oriented implementation of the functions, and of the block
 *  cipher kernel function Rijndael.
 *
 *  This has been coded for clarity, not necessarily for efficiency.
 *
 *  The functions f2, f3, f4 and f5 share the same inputs and have
 *  been coded together as a single function.  f1, f1* and f5* are
 *  all coded separately.
 *
 *-----------------------------------------------------------------*/

typedef unsigned char u8;


/*--------- Operator Variant Algorithm Configuration Field --------*/

            /*------- Insert your value of OP here -------*/
u8 OP[16] = {0x63, 0xbf, 0xa5, 0x0e, 0xe6, 0x52, 0x33, 0x65,
             0xff, 0x14, 0xc1, 0xf4, 0x5f, 0x88, 0x73, 0x7d};
            /*------- Insert your value of OP here -------*/


/*--------------------------- prototypes --------------------------*/

void f1    ( u8 k[16], u8 rand[16], u8 sqn[6], u8 amf[2],
             u8 mac_a[8] );
void f2345 ( u8 k[16], u8 rand[16],
             u8 res[8], u8 ck[16], u8 ik[16], u8 ak[6] );
void f1star( u8 k[16], u8 rand[16], u8 sqn[6], u8 amf[2],
             u8 mac_s[8] );
void f5star( u8 k[16], u8 rand[16],
             u8 ak[6] );
void ComputeOPc( u8 op_c[16] );
void RijndaelKeySchedule( u8 key[16] );
void RijndaelEncrypt( u8 input[16], u8 output[16] );


/*-------------------------------------------------------------------
 *                            Algorithm f1
 *-------------------------------------------------------------------
 *
 *  Computes network authentication code MAC-A from key K, random
 *  challenge RAND, sequence number SQN and authentication management
 *  field AMF.
 *
 *-----------------------------------------------------------------*/

void f1    ( u8 k[16], u8 rand[16], u8 sqn[6], u8 amf[2],
             u8 mac_a[8] )
{
  u8 op_c[16];
  u8 temp[16];
  u8 in1[16];
  u8 out1[16];
  u8 rijndaelInput[16];
  u8 i;

  RijndaelKeySchedule( k );

  ComputeOPc( op_c );

  for (i=0; i<16; i++)
    rijndaelInput[i] = rand[i] ^ op_c[i];
  RijndaelEncrypt( rijndaelInput, temp );

  for (i=0; i<6; i++)
  {
    in1[i]    = sqn[i];
    in1[i+8]  = sqn[i];
  }
  for (i=0; i<2; i++)
  {
    in1[i+6]  = amf[i];
    in1[i+14] = amf[i];
  }

  /* XOR op_c and in1, rotate by r1=64, and XOR *
   * on the constant c1 (which is all zeroes)   */

  for (i=0; i<16; i++)
    rijndaelInput[(i+8) % 16] = in1[i] ^ op_c[i];

  /* XOR on the value temp computed before */

  for (i=0; i<16; i++)
    rijndaelInput[i] ^= temp[i];

  RijndaelEncrypt( rijndaelInput, out1 );
  for (i=0; i<16; i++)
    out1[i] ^= op_c[i];

  for (i=0; i<8; i++)
    mac_a[i] = out1[i];

  return;
} /* end of function f1 */



/*-------------------------------------------------------------------
 *                            Algorithms f2-f5
 *-------------------------------------------------------------------
 *
 *  Takes key K and random challenge RAND, and returns response RES,
 *  confidentiality key CK, integrity key IK and anonymity key AK.
 *
 *-----------------------------------------------------------------*/

void f2345 ( u8 k[16], u8 rand[16],
             u8 res[8], u8 ck[16], u8 ik[16], u8 ak[6] )
{
  u8 op_c[16];
  u8 temp[16];
  u8 out[16];
  u8 rijndaelInput[16];
  u8 i;

  RijndaelKeySchedule( k );

  ComputeOPc( op_c );

  for (i=0; i<16; i++)
    rijndaelInput[i] = rand[i] ^ op_c[i];
  RijndaelEncrypt( rijndaelInput, temp );

  /* To obtain output block OUT2: XOR OPc and TEMP,    *
   * rotate by r2=0, and XOR on the constant c2 (which *
   * is all zeroes except that the last bit is 1).     */

  for (i=0; i<16; i++)
    rijndaelInput[i] = temp[i] ^ op_c[i];
  rijndaelInput[15] ^= 1;

  RijndaelEncrypt( rijndaelInput, out );
  for (i=0; i<16; i++)
    out[i] ^= op_c[i];

  for (i=0; i<8; i++)
    res[i] = out[i+8];
  for (i=0; i<6; i++)
    ak[i]  = out[i];

  /* To obtain output block OUT3: XOR OPc and TEMP,        *
   * rotate by r3=32, and XOR on the constant c3 (which    *
   * is all zeroes except that the next to last bit is 1). */

  for (i=0; i<16; i++)
    rijndaelInput[(i+12) % 16] = temp[i] ^ op_c[i];
  rijndaelInput[15] ^= 2;

  RijndaelEncrypt( rijndaelInput, out );
  for (i=0; i<16; i++)
    out[i] ^= op_c[i];

  for (i=0; i<16; i++)
    ck[i] = out[i];

  /* To obtain output block OUT4: XOR OPc and TEMP,         *
   * rotate by r4=64, and XOR on the constant c4 (which     *
   * is all zeroes except that the 2nd from last bit is 1). */

  for (i=0; i<16; i++)
    rijndaelInput[(i+8) % 16] = temp[i] ^ op_c[i];
  rijndaelInput[15] ^= 4;

  RijndaelEncrypt( rijndaelInput, out );
  for (i=0; i<16; i++)
    out[i] ^= op_c[i];

  for (i=0; i<16; i++)
    ik[i] = out[i];

  return;
```

```
} /* end of function f2345 */


/*-------------------------------------------------------------------
 *                            Algorithm f1*
 *-------------------------------------------------------------------
 *
 *  Computes resynch authentication code MAC-S from key K, random
 *  challenge RAND, sequence number SQN and authentication management
 *  field AMF.
 *
 *-----------------------------------------------------------------*/

void f1star( u8 k[16], u8 rand[16], u8 sqn[6], u8 amf[2],
             u8 mac_s[8] )
{
  u8 op_c[16];
  u8 temp[16];
  u8 in1[16];
  u8 out1[16];
  u8 rijndaelInput[16];
  u8 i;

  RijndaelKeySchedule( k );

  ComputeOPc( op_c );

  for (i=0; i<16; i++)
    rijndaelInput[i] = rand[i] ^ op_c[i];
  RijndaelEncrypt( rijndaelInput, temp );

  for (i=0; i<6; i++)
  {
    in1[i]    = sqn[i];
    in1[i+8]  = sqn[i];
  }
  for (i=0; i<2; i++)
  {
    in1[i+6]  = amf[i];
    in1[i+14] = amf[i];
  }

  /* XOR op_c and in1, rotate by r1=64, and XOR *
   * on the constant c1 (which is all zeroes)   */

  for (i=0; i<16; i++)
    rijndaelInput[(i+8) % 16] = in1[i] ^ op_c[i];

  /* XOR on the value temp computed before */

  for (i=0; i<16; i++)
    rijndaelInput[i] ^= temp[i];

  RijndaelEncrypt( rijndaelInput, out1 );
  for (i=0; i<16; i++)
    out1[i] ^= op_c[i];

  for (i=0; i<8; i++)
    mac_s[i] = out1[i+8];

  return;
```

```
} /* end of function f1star */


/*-------------------------------------------------------------------
 *                            Algorithm f5*
 *-------------------------------------------------------------------
 *
 *  Takes key K and random challenge RAND, and returns resynch
 *  anonymity key AK.
 *
 *-----------------------------------------------------------------*/

void f5star( u8 k[16], u8 rand[16],
             u8 ak[6] )
{
  u8 op_c[16];
  u8 temp[16];
  u8 out[16];
  u8 rijndaelInput[16];
  u8 i;

  RijndaelKeySchedule( k );

  ComputeOPc( op_c );

  for (i=0; i<16; i++)
    rijndaelInput[i] = rand[i] ^ op_c[i];
  RijndaelEncrypt( rijndaelInput, temp );

  /* To obtain output block OUT5: XOR OPc and TEMP,          *
   * rotate by r5=96, and XOR on the constant c5 (which      *
   * is all zeroes except that the 3rd from last bit is 1). */

  for (i=0; i<16; i++)
    rijndaelInput[(i+4) % 16] = temp[i] ^ op_c[i];
  rijndaelInput[15] ^= 8;

  RijndaelEncrypt( rijndaelInput, out );
  for (i=0; i<16; i++)
    out[i] ^= op_c[i];

  for (i=0; i<6; i++)
    ak[i] = out[i];

  return;
} /* end of function f5star */


/*-------------------------------------------------------------------
 *  Function to compute OPc from OP and K.  Assumes key schedule has
    already been performed.
 *-----------------------------------------------------------------*/

void ComputeOPc( u8 op_c[16] )
{
  u8 i;

  RijndaelEncrypt( OP, op_c );
  for (i=0; i<16; i++)
    op_c[i] ^= OP[i];
  return;
} /* end of function ComputeOPc */



/*-------------------- Rijndael round subkeys ---------------------*/
u8 roundKeys[11][4][4];

/*--------------------- Rijndael S box table ----------------------*/
u8 S[256] = {
 99,124,119,123,242,107,111,197, 48,  1,103, 43,254,215,171,118,
202,130,201,125,250, 89, 71,240,173,212,162,175,156,164,114,192,
183,253,147, 38, 54, 63,247,204, 52,165,229,241,113,216, 49, 21,
  4,199, 35,195, 24,150,  5,154,  7, 18,128,226,235, 39,178,117,
  9,131, 44, 26, 27,110, 90,160, 82, 59,214,179, 41,227, 47,132,
 83,209,  0,237, 32,252,177, 91,106,203,190, 57, 74, 76, 88,207,
208,239,170,251, 67, 77, 51,133, 69,249,  2,127, 80, 60,159,168,
 81,163, 64,143,146,157, 56,245,188,182,218, 33, 16,255,243,210,
205, 12, 19,236, 95,151, 68, 23,196,167,126, 61,100, 93, 25,115,
 96,129, 79,220, 34, 42,144,136, 70,238,184, 20,222, 94, 11,219,
224, 50, 58, 10, 73,  6, 36, 92,194,211,172, 98,145,149,228,121,
231,200, 55,109,141,213, 78,169,108, 86,244,234,101,122,174,  8,
186,120, 37, 46, 28,166,180,198,232,221,116, 31, 75,189,139,138,
112, 62,181,102, 72,  3,246, 14, 97, 53, 87,185,134,193, 29,158,
225,248,152, 17,105,217,142,148,155, 30,135,233,206, 85, 40,223,
140,161,137, 13,191,230, 66,104, 65,153, 45, 15,176, 84,187, 22,
};

/*------- This array does the multiplication by x in GF(2^8) ------*/
u8 Xtime[256] = {
  0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30,
 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62,
 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94,
 96, 98,100,102,104,106,108,110,112,114,116,118,120,122,124,126,
128,130,132,134,136,138,140,142,144,146,148,150,152,154,156,158,
160,162,164,166,168,170,172,174,176,178,180,182,184,186,188,190,
192,194,196,198,200,202,204,206,208,210,212,214,216,218,220,222,
224,226,228,230,232,234,236,238,240,242,244,246,248,250,252,254,
 27, 25, 31, 29, 19, 17, 23, 21, 11,  9, 15, 13,  3,  1,  7,  5,
 59, 57, 63, 61, 51, 49, 55, 53, 43, 41, 47, 45, 35, 33, 39, 37,
 91, 89, 95, 93, 83, 81, 87, 85, 75, 73, 79, 77, 67, 65, 71, 69,
123,121,127,125,115,113,119,117,107,105,111,109, 99, 97,103,101,
155,153,159,157,147,145,151,149,139,137,143,141,131,129,135,133,
187,185,191,189,179,177,183,181,171,169,175,173,163,161,167,165,
219,217,223,221,211,209,215,213,203,201,207,205,195,193,199,197,
251,249,255,253,243,241,247,245,235,233,239,237,227,225,231,229
};


/*-------------------------------------------------------------------
 *  Rijndael key schedule function.  Takes 16-byte key and creates
 *  all Rijndael's internal subkeys ready for encryption.
 *-----------------------------------------------------------------*/

void RijndaelKeySchedule( u8 key[16] )
{
  u8 roundConst;
  int i, j;

  /* first round key equals key */
  for (i=0; i<16; i++)
```

```
    roundKeys[0][i & 0x03][i>>2] = key[i];

  roundConst = 1;

  /* now calculate round keys */
  for (i=1; i<11; i++)
  {
    roundKeys[i][0][0] = S[roundKeys[i-1][1][3]]
                         ^ roundKeys[i-1][0][0] ^ roundConst;
    roundKeys[i][1][0] = S[roundKeys[i-1][2][3]]
                         ^ roundKeys[i-1][1][0];
    roundKeys[i][2][0] = S[roundKeys[i-1][3][3]]
                         ^ roundKeys[i-1][2][0];
    roundKeys[i][3][0] = S[roundKeys[i-1][0][3]]
                         ^ roundKeys[i-1][3][0];

    for (j=0; j<4; j++)
    {
      roundKeys[i][j][1] = roundKeys[i-1][j][1] ^ roundKeys[i][j][0];
      roundKeys[i][j][2] = roundKeys[i-1][j][2] ^ roundKeys[i][j][1];
      roundKeys[i][j][3] = roundKeys[i-1][j][3] ^ roundKeys[i][j][2];
    }

    /* update round constant */
    roundConst = Xtime[roundConst];
  }

  return;
} /* end of function RijndaelKeySchedule */


/* Round key addition function */
void KeyAdd(u8 state[4][4], u8 roundKeys[11][4][4], int round)
{
  int i, j;

  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
      state[i][j] ^= roundKeys[round][i][j];

  return;
}


/* Byte substitution transformation */
int ByteSub(u8 state[4][4])
{
  int i, j;

  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
      state[i][j] = S[state[i][j]];

  return 0;
}


/* Row shift transformation */
void ShiftRow(u8 state[4][4])
{
  u8 temp;
  /* left rotate row 1 by 1 */
  temp = state[1][0];
  state[1][0] = state[1][1];
  state[1][1] = state[1][2];
  state[1][2] = state[1][3];
  state[1][3] = temp;

  /* left rotate row 2 by 2 */
  temp = state[2][0];
  state[2][0] = state[2][2];
  state[2][2] = temp;
  temp = state[2][1];
  state[2][1] = state[2][3];
  state[2][3] = temp;

  /* left rotate row 3 by 3 */
  temp = state[3][0];
  state[3][0] = state[3][3];
  state[3][3] = state[3][2];
  state[3][2] = state[3][1];
  state[3][1] = temp;

  return;
}


/* MixColumn transformation*/
void MixColumn(u8 state[4][4])
{
  u8 temp, tmp, tmp0;
  int i;

  /* do one column at a time */
  for (i=0; i<4;i++)
  {
    temp = state[0][i] ^ state[1][i] ^ state[2][i] ^ state[3][i];
    tmp0 = state[0][i];

    /* Xtime array does multiply by x in GF2^8 */
    tmp = Xtime[state[0][i] ^ state[1][i]];
    state[0][i] ^= temp ^ tmp;

    tmp = Xtime[state[1][i] ^ state[2][i]];
    state[1][i] ^= temp ^ tmp;

    tmp = Xtime[state[2][i] ^ state[3][i]];
    state[2][i] ^= temp ^ tmp;

    tmp = Xtime[state[3][i] ^ tmp0];
    state[3][i] ^= temp ^ tmp;
  }

  return;
}


/*-------------------------------------------------------------------
 *  Rijndael encryption function.  Takes 16-byte input and creates
 *  16-byte output (using round keys already derived from 16-byte
 *  key).
 *-----------------------------------------------------------------*/

void RijndaelEncrypt( u8 input[16], u8 output[16] )
{
  u8 state[4][4];
  int i, r;

  /* initialise state array from input byte string */
  for (i=0; i<16; i++)
    state[i & 0x3][i>>2] = input[i];

  /* add first round_key */
  KeyAdd(state, roundKeys, 0);

  /* do lots of full rounds */
  for (r=1; r<=9; r++)
  {
    ByteSub(state);
    ShiftRow(state);
    MixColumn(state);
    KeyAdd(state, roundKeys, r);
  }

  /* final round */
  ByteSub(state);
  ShiftRow(state);
  KeyAdd(state, roundKeys, r);

  /* produce output byte string from state array */
  for (i=0; i<16; i++)
  {
    output[i] = state[i & 0x3][i>>2];
  }

  return;
} /* end of function RijndaelEncrypt */
```

ANNEX 4
Rijndael Listing — 32-Bit Word Oriented

```
/*-------------------------------------------------------------------
 *                     Rijndael Implementation
 *-------------------------------------------------------------------
 *
 *  A sample 32-bit orientated implementation of Rijndael, the
 *  suggested kernel for the example 3GPP authentication and key
 *  agreement functions.
 *
 *  This implementation draws on the description in section 5.2 of
 *  the AES proposal and also on the implementation by
 *  Dr B. R. Gladman <brg@gladman.uk.net> 9th October 2000.
 *  It uses a number of large (4k) lookup tables to implement the
 *  algorithm in an efficient manner.
 *
 *  Note: in this implementation the State is stored in four 32-bit
 *  words, one per column of the State, with the top byte of the
 *  column being the _least_ significant byte of the word.
 *
*-----------------------------------------------------------------*/

#define LITTLE_ENDIAN /* For INTEL architecture */

typedef unsigned char   u8;
typedef unsigned int   u32;

/* Circular byte rotates of 32 bit values */

#define rot1(x) ((x <<  8) | (x >> 24))
#define rot2(x) ((x << 16) | (x >> 16))
#define rot3(x) ((x << 24) | (x >>  8))

/* Extract a byte from a 32-bit u32 */

#define byte0(x)    ((u8)(x))
#define byte1(x)    ((u8)(x >>  8))
#define byte2(x)    ((u8)(x >> 16))
#define byte3(x)    ((u8)(x >> 24))


/* Put or get a 32 bit u32 (v) in machine order from a byte *
 * address in (x)                                           */

#ifdef  LITTLE_ENDIAN

#define u32_in(x)      (*(u32*)(x))
#define u32_out(x,y)   (*(u32*)(x) = y)

#else

/* Invert byte order in a 32 bit variable */

__inline u32 byte_swap(const u32 x)
{
    return rot1(x) & 0x00ff00ff | rot3(x) & 0xff00ff00;
}
__inline u32 u32_in(const u8 x[])
{
  return byte_swap(*(u32*)x);
};
__inline void u32_out(u8 x[], const u32 v)
{
  *(u32*)x = byte_swap(v);
};

#endif

/*--------------- The lookup tables ----------------------------*/

static u32 rnd_con[10] =
{
 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36
};

static u32 ft_tab[4][256] =
{
 {
 0xA56363C6,0x847C7CF8,0x997777EE,0x8D7B7BF6,0x0DF2F2FF,0xBD6B6BD6,0xB16F6FDE,0x54C5C591,
 0x50303060,0x03010102,0xA96767CE,0x7D2B2B56,0x19FEFEE7,0x62D7D7B5,0xE6ABAB4D,0x9A7676EC,
 0x45CACA8F,0x9D82821F,0x40C9C989,0x877D7DFA,0x15FAFAEF,0xEB5959B2,0xC947478E,0x0BF0F0FB,
 0xECADAD41,0x67D4D4B3,0xFDA2A25F,0xEAAFAF45,0xBF9C9C23,0xF7A4A453,0x967272E4,0x5BC0C09B,
 0xC2B7B775,0x1CFDFDE1,0xAE93933D,0x6A26264C,0x5A36366C,0x413F3F7E,0x02F7F7F5,0x4FCCCC83,
 0x5C343468,0xF4A5A551,0x34E5E5D1,0x08F1F1F9,0x937171E2,0x73D8D8AB,0x53313162,0x3F15152A,
 0x0C040408,0x52C7C795,0x65232346,0x5EC3C39D,0x28181830,0xA1969637,0x0F05050A,0xB59A9A2F,
 0x0907070E,0x36121224,0x9B80801B,0x3DE2E2DF,0x26EBEBCD,0x6927274E,0xCDB2B27F,0x9F7575EA,
 0x1B090912,0x9E83831D,0x742C2C58,0x2E1A1A34,0x2D1B1B36,0xB26E6EDC,0xEE5A5AB4,0xFBA0A05B,
 0xF65252A4,0x4D3B3B76,0x61D6D6B7,0xCEB3B37D,0x7B292952,0x3EE3E3DD,0x712F2F5E,0x97848413,
 0xF55353A6,0x68D1D1B9,0000000000,0x2CEDEDC1,0x60202040,0x1FFCFCE3,0xC8B1B179,0xED5B5BB6,
 0xBE6A6AD4,0x46CBCB8D,0xD9BEBE67,0x4B393972,0xDE4A4A94,0xD44C4C98,0xE85858B0,0x4ACFCF85,
 0x6BD0D0BB,0x2AEFEFC5,0xE5AAAA4F,0x16FBFBED,0xC5434386,0xD74D4D9A,0x55333366,0x94858511,
 0xCF45458A,0x10F9F9E9,0x06020204,0x817F7FFE,0xF05050A0,0x443C3C78,0xBA9F9F25,0xE3A8A84B,
 0xF35151A2,0xFEA3A35D,0xC0404080,0x8A8F8F05,0xAD92923F,0xBC9D9D21,0x48383870,0x04F5F5F1,
 0xDFBCBC63,0xC1B6B677,0x75DADAAF,0x63212142,0x30101020,0x1AFFFFE5,0x0EF3F3FD,0x6DD2D2BF,
 0x4CCDCD81,0x140C0C18,0x35131326,0x2FECECC3,0xE15F5FBE,0xA2979735,0xCC444488,0x3917172E,
 0x57C4C493,0xF2A7A755,0x827E7EFC,0x473D3D7A,0xAC6464C8,0xE75D5DBA,0x2B191932,0x957373E6,
 0xA06060C0,0x98818119,0xD14F4F9E,0x7FDCDCA3,0x66222244,0x7E2A2A54,0xAB90903B,0x8388880B,
 0xCA46468C,0x29EEEEC7,0xD3B8B86B,0x3C141428,0x79DEDEA7,0xE25E5EBC,0x1D0B0B16,0x76DBDBAD,
 0x3BE0E0DB,0x56323264,0x4E3A3A74,0x1E0A0A14,0xDB494992,0x0A06060C,0x6C242448,0xE45C5CB8,
 0x5DC2C29F,0x6ED3D3BD,0xEFACAC43,0xA66262C4,0xA8919139,0xA4959531,0x37E4E4D3,0x8B7979F2,
 0x32E7E7D5,0x43C8C88B,0x5937376E,0xB76D6DDA,0x8C8D8D01,0x64D5D5B1,0xD24E4E9C,0xE0A9A949,
 0xB46C6CD8,0xFA5656AC,0x07F4F4F3,0x25EAEACF,0xAF6565CA,0x8E7A7AF4,0xE9AEAE47,0x18080810,
 0xD5BABA6F,0x887878F0,0x6F25254A,0x722E2E5C,0x241C1C38,0xF1A6A657,0xC7B4B473,0x51C6C697,
 0x23E8E8CB,0x7CDDDDA1,0x9C7474E8,0x211F1F3E,0xDD4B4B96,0xDCBDBD61,0x868B8B0D,0x858A8A0F,
 0x907070E0,0x423E3E7C,0xC4B5B571,0xAA6666CC,0xD8484890,0x05030306,0x01F6F6F7,0x120E0E1C,
 0xA36161C2,0x5F35356A,0xF95757AE,0xD0B9B969,0x91868617,0x58C1C199,0x271D1D3A,0xB99E9E27,
 0x38E1E1D9,0x13F8F8EB,0xB398982B,0x33111122,0xBB6969D2,0x70D9D9A9,0x898E8E07,0xA7949433,
 0xB69B9B2D,0x221E1E3C,0x92878715,0x20E9E9C9,0x49CECE87,0xFF5555AA,0x78282850,0x7ADFDFA5,
 0x8F8C8C03,0xF8A1A159,0x80898909,0x170D0D1A,0xDABFBF65,0x31E6E6D7,0xC6424284,0xB86868D0,
 0xC3414182,0xB0999929,0x772D2D5A,0x110F0F1E,0xCBB0B07B,0xFC5454A8,0xD6BBBB6D,0x3A16162C
 },
 {
 0x6363C6A5,0x7C7CF884,0x7777EE99,0x7B7BF68D,0xF2F2FF0D,0x6B6BD6BD,0x6F6FDEB1,0xC5C59154,
 0x30306050,0x01010203,0x6767CEA9,0x2B2B567D,0xFEFEE719,0xD7D7B562,0xABAB4DE6,0x7676EC9A,
 0xCACA8F45,0x82821F9D,0xC9C98940,0x7D7DFA87,0xFAFAEF15,0x5959B2EB,0x47478EC9,0xF0F0FB0B,
 0xADAD41EC,0xD4D4B367,0xA2A25FFD,0xAFAF45EA,0x9C9C23BF,0xA4A453F7,0x7272E496,0xC0C09B5B,
 0xB7B775C2,0xFDFDE11C,0x93933DAE,0x26264C6A,0x36366C5A,0x3F3F7E41,0xF7F7F502,0xCCCC834F,
 0x3434685C,0xA5A551F4,0xE5E5D134,0xF1F1F908,0x7171E293,0xD8D8AB73,0x31316253,0x15152A3F,
 0x0404080C,0xC7C79552,0x23234665,0xC3C39D5E,0x18183028,0x969637A1,0x05050A0F,0x9A9A2FB5,
 0x07070E09,0x12122436,0x80801B9B,0xE2E2DF3D,0xEBEBCD26,0x27274E69,0xB2B27FCD,0x7575EA9F,
 0x0909121B,0x83831D9E,0x2C2C5874,0x1A1A342E,0x1B1B362D,0x6E6EDCB2,0x5A5AB4EE,0xA0A05BFB,
 0x5252A4F6,0x3B3B764D,0xD6D6B761,0xB3B37DCE,0x2929527B,0xE3E3DD3E,0x2F2F5E71,0x84841397,
 0x5353A6F5,0xD1D1B968,0000000000,0xEDEDC12C,0x20204060,0xFCFCE31F,0xB1B179C8,0x5B5BB6ED,
 0x6A6AD4BE,0xCBCB8D46,0xBEBE67D9,0x3939724B,0x4A4A94DE,0x4C4C98D4,0x5858B0E8,0xCFCF854A,
 0xD0D0BB6B,0xEFEFC52A,0xAAAA4FE5,0xFBFBED16,0x434386C5,0x4D4D9AD7,0x33336655,0x85851194,
 0x45458ACF,0xF9F9E910,0x02020406,0x7F7FFE81,0x5050A0F0,0x3C3C7844,0x9F9F25BA,0xA8A84BE3,
 0x5151A2F3,0xA3A35DFE,0x404080C0,0x8F8F058A,0x92923FAD,0x9D9D21BC,0x38387048,0xF5F5F104,
 0xBCBC63DF,0xB6B677C1,0xDADAAF75,0x21214263,0x10102030,0xFFFFE51A,0xF3F3FD0E,0xD2D2BF6D,
 0xCDCD814C,0x0C0C1814,0x13132635,0xECECC32F,0x5F5FBEE1,0x979735A2,0x444488CC,0x17172E39,
 0xC4C49357,0xA7A755F2,0x7E7EFC82,0x3D3D7A47,0x6464C8AC,0x5D5DBAE7,0x1919322B,0x7373E695,
 0x6060C0A0,0x81811998,0x4F4F9ED1,0xDCDCA37F,0x22224466,0x2A2A547E,0x90903BAB,0x88880B83,
 0x46468CCA,0xEEEEC729,0xB8B86BD3,0x1414283C,0xDEDEA779,0x5E5EBCE2,0x0B0B161D,0xDBDBAD76,
 0xE0E0DB3B,0x32326456,0x3A3A744E,0x0A0A141E,0x494992DB,0x06060C0A,0x2424486C,0x5C5CB8E4,
 0xC2C29F5D,0xD3D3BD6E,0xACAC43EF,0x6262C4A6,0x919139A8,0x959531A4,0xE4E4D337,0x7979F28B,
 0xE7E7D532,0xC8C88B43,0x37376E59,0x6D6DDAB7,0x8D8D018C,0xD5D5B164,0x4E4E9CD2,0xA9A949E0,
 0x6C6CD8B4,0x5656ACFA,0xF4F4F307,0xEAEACF25,0x6565CAAF,0x7A7AF48E,0xAEAE47E9,0x08081018,
 0xBABA6FD5,0x7878F088,0x25254A6F,0x2E2E5C72,0x1C1C3824,0xA6A657F1,0xB4B473C7,0xC6C69751,
 0xE8E8CB23,0xDDDDA17C,0x7474E89C,0x1F1F3E21,0x4B4B96DD,0xBDBD61DC,0x8B8B0D86,0x8A8A0F85,
 0x7070E090,0x3E3E7C42,0xB5B571C4,0x6666CCAA,0x484890D8,0x03030605,0xF6F6F701,0x0E0E1C12,
 0x6161C2A3,0x35356A5F,0x5757AEF9,0xB9B969D0,0x86861791,0xC1C19958,0x1D1D3A27,0x9E9E27B9,
 0xE1E1D938,0xF8F8EB13,0x98982BB3,0x11112233,0x6969D2BB,0xD9D9A970,0x8E8E0789,0x949433A7,
 0x9B9B2DB6,0x1E1E3C22,0x87871592,0xE9E9C920,0xCECE8749,0x5555AAFF,0x28285078,0xDFDFA57A,
 0x8C8C038F,0xA1A159F8,0x89890980,0x0D0D1A17,0xBFBF65DA,0xE6E6D731,0x424284C6,0x6868D0B8,
 0x414182C3,0x999929B0,0x2D2D5A77,0x0F0F1E11,0xB0B07BCB,0x5454A8FC,0xBBBB6DD6,0x16162C3A
 },
 {
 0x63C6A563,0x7CF8847C,0x77EE9977,0x7BF68D7B,0xF2FF0DF2,0x6BD6BD6B,0x6FDEB16F,0xC59154C5,
 0x30605030,0x01020301,0x67CEA967,0x2B567D2B,0xFEE719FE,0xD7B562D7,0xAB4DE6AB,0x76EC9A76,
 0xCA8F45CA,0x821F9D82,0xC98940C9,0x7DFA877D,0xFAEF15FA,0x59B2EB59,0x478EC947,0xF0FB0BF0,
 0xAD41ECAD,0xD4B367D4,0xA25FFDA2,0xAF45EAAF,0x9C23BF9C,0xA453F7A4,0x72E49672,0xC09B5BC0,
 0xB775C2B7,0xFDE11CFD,0x933DAE93,0x264C6A26,0x366C5A36,0x3F7E413F,0xF7F502F7,0xCC834FCC,
 0x34685C34,0xA551F4A5,0xE5D134E5,0xF1F908F1,0x71E29371,0xD8AB73D8,0x31625331,0x152A3F15,
 0x04080C04,0xC79552C7,0x23466523,0xC39D5EC3,0x18302818,0x9637A196,0x050A0F05,0x9A2FB59A,
 0x070E0907,0x12243612,0x801B9B80,0xE2DF3DE2,0xEBCD26EB,0x274E6927,0xB27FCDB2,0x75EA9F75,
 0x09121B09,0x831D9E83,0x2C58742C,0x1A342E1A,0x1B362D1B,0x6EDCB26E,0x5AB4EE5A,0xA05BFBA0,
 0x52A4F652,0x3B764D3B,0xD6B761D6,0xB37DCEB3,0x29527B29,0xE3DD3EE3,0x2F5E712F,0x84139784,
 0x53A6F553,0xD1B968D1,0000000000,0xEDC12CED,0x20406020,0xFCE31FFC,0xB179C8B1,0x5BB6ED5B,
 0x6AD4BE6A,0xCB8D46CB,0xBE67D9BE,0x39724B39,0x4A94DE4A,0x4C98D44C,0x58B0E858,0xCF854ACF,
 0xD0BB6BD0,0xEFC52AEF,0xAA4FE5AA,0xFBED16FB,0x4386C543,0x4D9AD74D,0x33665533,0x85119485,
 0x458ACF45,0xF9E910F9,0x02040602,0x7FFE817F,0x50A0F050,0x3C78443C,0x9F25BA9F,0xA84BE3A8,
 0x51A2F351,0xA35DFEA3,0x4080C040,0x8F058A8F,0x923FAD92,0x9D21BC9D,0x38704838,0xF5F104F5,
 0xBC63DFBC,0xB677C1B6,0xDAAF75DA,0x21426321,0x10203010,0xFFE51AFF,0xF3FD0EF3,0xD2BF6DD2,
 0xCD814CCD,0x0C18140C,0x13263513,0xECC32FEC,0x5FBEE15F,0x9735A297,0x4488CC44,0x172E3917,
 0xC49357C4,0xA755F2A7,0x7EFC827E,0x3D7A473D,0x64C8AC64,0x5DBAE75D,0x19322B19,0x73E69573,
 0x60C0A060,0x81199881,0x4F9ED14F,0xDCA37FDC,0x22446622,0x2A547E2A,0x903BAB90,0x880B8388,
 0x468CCA46,0xEEC729EE,0xB86BD3B8,0x14283C14,0xDEA779DE,0x5EBCE25E,0x0B161D0B,0xDBAD76DB,
 0xE0DB3BE0,0x32645632,0x3A744E3A,0x0A141E0A,0x4992DB49,0x060C0A06,0x24486C24,0x5CB8E45C,
 0xC29F5DC2,0xD3BD6ED3,0xAC43EFAC,0x62C4A662,0x9139A891,0x9531A495,0xE4D337E4,0x79F28B79,
 0xE7D532E7,0xC88B43C8,0x376E5937,0x6DDAB76D,0x8D018C8D,0xD5B164D5,0x4E9CD24E,0xA949E0A9,
 0x6CD8B46C,0x56ACFA56,0xF4F307F4,0xEACF25EA,0x65CAAF65,0x7AF48E7A,0xAE47E9AE,0x08101808,
 0xBA6FD5BA,0x78F08878,0x254A6F25,0x2E5C722E,0x1C38241C,0xA657F1A6,0xB473C7B4,0xC69751C6,
 0xE8CB23E8,0xDDA17CDD,0x74E89C74,0x1F3E211F,0x4B96DD4B,0xBD61DCBD,0x8B0D868B,0x8A0F858A,
 0x70E09070,0x3E7C423E,0xB571C4B5,0x66CCAA66,0x4890D848,0x03060503,0xF6F701F6,0x0E1C120E,
 0x61C2A361,0x356A5F35,0x57AEF957,0xB969D0B9,0x86179186,0xC19958C1,0x1D3A271D,0x9E27B99E,
 0xE1D938E1,0xF8EB13F8,0x982BB398,0x11223311,0x69D2BB69,0xD9A970D9,0x8E07898E,0x9433A794,
 0x9B2DB69B,0x1E3C221E,0x87159287,0xE9C920E9,0xCE8749CE,0x55AAFF55,0x28507828,0xDFA57ADF,
 0x8C038F8C,0xA159F8A1,0x89098089,0x0D1A170D,0xBF65DABF,0xE6D731E6,0x4284C642,0x68D0B868,
 0x4182C341,0x9929B099,0x2D5A772D,0x0F1E110F,0xB07BCBB0,0x54A8FC54,0xBB6DD6BB,0x162C3A16
 },
 {
 0xC6A56363,0xF8847C7C,0xEE997777,0xF68D7B7B,0xFF0DF2F2,0xD6BD6B6B,0xDEB16F6F,0x9154C5C5,
 0x60503030,0x02030101,0xCEA96767,0x567D2B2B,0xE719FEFE,0xB562D7D7,0x4DE6ABAB,0xEC9A7676,
 0x8F45CACA,0x1F9D8282,0x8940C9C9,0xFA877D7D,0xEF15FAFA,0xB2EB5959,0x8EC94747,0xFB0BF0F0,
 0x41ECADAD,0xB367D4D4,0x5FFDA2A2,0x45EAAFAF,0x23BF9C9C,0x53F7A4A4,0xE4967272,0x9B5BC0C0,
 0x75C2B7B7,0xE11CFDFD,0x3DAE9393,0x4C6A2626,0x6C5A3636,0x7E413F3F,0xF502F7F7,0x834FCCCC,
 0x685C3434,0x51F4A5A5,0xD134E5E5,0xF908F1F1,0xE2937171,0xAB73D8D8,0x62533131,0x2A3F1515,
 0x080C0404,0x9552C7C7,0x46652323,0x9D5EC3C3,0x30281818,0x37A19696,0x0A0F0505,0x2FB59A9A,
 0x0E090707,0x24361212,0x1B9B8080,0xDF3DE2E2,0xCD26EBEB,0x4E692727,0x7FCDB2B2,0xEA9F7575,
 0x121B0909,0x1D9E8383,0x58742C2C,0x342E1A1A,0x362D1B1B,0xDCB26E6E,0xB4EE5A5A,0x5BFBA0A0,
 0xA4F65252,0x764D3B3B,0xB761D6D6,0x7DCEB3B3,0x527B2929,0xDD3EE3E3,0x5E712F2F,0x13978484,
 0xA6F55353,0xB968D1D1,0000000000,0xC12CEDED,0x40602020,0xE31FFCFC,0x79C8B1B1,0xB6ED5B5B,
 0xD4BE6A6A,0x8D46CBCB,0x67D9BEBE,0x724B3939,0x94DE4A4A,0x98D44C4C,0xB0E85858,0x854ACFCF,
 0xBB6BD0D0,0xC52AEFEF,0x4FE5AAAA,0xED16FBFB,0x86C54343,0x9AD74D4D,0x66553333,0x11948585,
 0x8ACF4545,0xE910F9F9,0x04060202,0xFE817F7F,0xA0F05050,0x78443C3C,0x25BA9F9F,0x4BE3A8A8,
 0xA2F35151,0x5DFEA3A3,0x80C04040,0x058A8F8F,0x3FAD9292,0x21BC9D9D,0x70483838,0xF104F5F5,
 0x63DFBCBC,0x77C1B6B6,0xAF75DADA,0x42632121,0x20301010,0xE51AFFFF,0xFD0EF3F3,0xBF6DD2D2,
 0x814CCDCD,0x18140C0C,0x26351313,0xC32FECEC,0xBEE15F5F,0x35A29797,0x88CC4444,0x2E391717,
 0x9357C4C4,0x55F2A7A7,0xFC827E7E,0x7A473D3D,0xC8AC6464,0xBAE75D5D,0x322B1919,0xE6957373,
 0xC0A06060,0x19988181,0x9ED14F4F,0xA37FDCDC,0x44662222,0x547E2A2A,0x3BAB9090,0x0B838888,
 0x8CCA4646,0xC729EEEE,0x6BD3B8B8,0x283C1414,0xA779DEDE,0xBCE25E5E,0x161D0B0B,0xAD76DBDB,
 0xDB3BE0E0,0x64563232,0x744E3A3A,0x141E0A0A,0x92DB4949,0x0C0A0606,0x486C2424,0xB8E45C5C,
 0x9F5DC2C2,0xBD6ED3D3,0x43EFACAC,0xC4A66262,0x39A89191,0x31A49595,0xD337E4E4,0xF28B7979,
 0xD532E7E7,0x8B43C8C8,0x6E593737,0xDAB76D6D,0x018C8D8D,0xB164D5D5,0x9CD24E4E,0x49E0A9A9,
 0xD8B46C6C,0xACFA5656,0xF307F4F4,0xCF25EAEA,0xCAAF6565,0xF48E7A7A,0x47E9AEAE,0x10180808,
 0x6FD5BABA,0xF0887878,0x4A6F2525,0x5C722E2E,0x38241C1C,0x57F1A6A6,0x73C7B4B4,0x9751C6C6,
 0xCB23E8E8,0xA17CDDDD,0xE89C7474,0x3E211F1F,0x96DD4B4B,0x61DCBDBD,0x0D868B8B,0x0F858A8A,
 0xE0907070,0x7C423E3E,0x71C4B5B5,0xCCAA6666,0x90D84848,0x06050303,0xF701F6F6,0x1C120E0E,
 0xC2A36161,0x6A5F3535,0xAEF95757,0x69D0B9B9,0x17918686,0x9958C1C1,0x3A271D1D,0x27B99E9E,
 0xD938E1E1,0xEB13F8F8,0x2BB39898,0x22331111,0xD2BB6969,0xA970D9D9,0x07898E8E,0x33A79494,
 0x2DB69B9B,0x3C221E1E,0x15928787,0xC920E9E9,0x8749CECE,0xAAFF5555,0x50782828,0xA57ADFDF,
 0x038F8C8C,0x59F8A1A1,0x09808989,0x1A170D0D,0x65DABFBF,0xD731E6E6,0x84C64242,0xD0B86868,
 0x82C34141,0x29B09999,0x5A772D2D,0x1E110F0F,0x7BCBB0B0,0xA8FC5454,0x6DD6BBBB,0x2C3A1616
 }
};

static u32 it_tab[4][256] =
{
 {
 0x50A7F451,0x5365417E,0xC3A4171A,0x965E273A,0xCB6BAB3B,0xF1459D1F,0xAB58FAAC,0x9303E34B,
 0x55FA3020,0xF66D76AD,0x9176CC88,0x254C02F5,0xFCD7E54F,0xD7CB2AC5,0x80443526,0x8FA362B5,
 0x495AB1DE,0x671BBA25,0x980EEA45,0xE1C0FE5D,0x02752FC3,0x12F04C81,0xA397468D,0xC6F9D36B,
 0xE75F8F03,0x959C9215,0xEB7A6DBF,0xDA595295,0x2D83BED4,0xD3217458,0x2969E049,0x44C8C98E,
 0x6A89C275,0x78798EF4,0x6B3E5899,0xDD71B927,0xB64FE1BE,0x17AD88F0,0x66AC20C9,0xB43ACE7D,
 0x184ADF63,0x82311AE5,0x60335197,0x457F5362,0xE07764B1,0x84AE6BBB,0x1CA081FE,0x942B08F9,
 0x58684870,0x19FD458F,0x876CDE94,0xB7F87B52,0x23D373AB,0xE2024B72,0x578F1FE3,0x2AAB5566,
 0x0728EBB2,0x03C2B52F,0x9A7BC586,0xA50837D3,0xF2872830,0xB2A5BF23,0xBA6A0302,0x5C8216ED,
 0x2B1CCF8A,0x92B479A7,0xF0F207F3,0xA1E2694E,0xCDF4DA65,0xD5BE0506,0x1F6234D1,0x8AFEA6C4,
 0x9D532E34,0xA055F3A2,0x32E18A05,0x75EBF6A4,0x39EC830B,0xAAEF6040,0x069F715E,0x51106EBD,
 0xF98A213E,0x3D06DD96,0xAE053EDD,0x46BDE64D,0xB58D5491,0x055DC471,0x6FD40604,0xFF155060,
 0x24FB9819,0x97E9BDD6,0xCC434089,0x779ED967,0xBD42E8B0,0x888B8907,0x385B19E7,0xDBEEC879,
 0x470A7CA1,0xE90F427C,0xC91E84F8,0000000000,0x83868009,0x48ED2B32,0xAC70111E,0x4E725A6C,
 0xFBFF0EFD,0x5638850F,0x1ED5AE3D,0x27392D36,0x64D90F0A,0x21A65C68,0xD1545B9B,0x3A2E3624,
 0xB1670A0C,0x0FE75793,0xD296EEB4,0x9E919B1B,0x4FC5C080,0xA220DC61,0x694B775A,0x161A121C,
 0x0ABA93E2,0xE52AA0C0,0x43E0223C,0x1D171B12,0x0B0D090E,0xADC78BF2,0xB9A8B62D,0xC8A91E14,
 0x8519F157,0x4C0775AF,0xBBDD99EE,0xFD607FA3,0x9F2601F7,0xBCF5725C,0xC53B6644,0x347EFB5B,
 0x7629438B,0xDCC623CB,0x68FCEDB6,0x63F1E4B8,0xCADC31D7,0x10856342,0x40229713,0x2011C684,
 0x7D244A85,0xF83DBBD2,0x1132F9AE,0x6DA129C7,0x4B2F9E1D,0xF330B2DC,0xEC52860D,0xD0E3C177,
 0x6C16B32B,0x99B970A9,0xFA489411,0x2264E947,0xC48CFCA8,0x1A3FF0A0,0xD82C7D56,0xEF903322,
 0xC74E4987,0xC1D138D9,0xFEA2CA8C,0x360BD498,0xCF81F5A6,0x28DE7AA5,0x268EB7DA,0xA4BFAD3F,
 0xE49D3A2C,0x0D927850,0x9BCC5F6A,0x62467E54,0xC2138DF6,0xE8B8D890,0x5EF7392E,0xF5AFC382,
 0xBE805D9F,0x7C93D069,0xA92DD56F,0xB31225CF,0x3B99ACC8,0xA77D1810,0x6E639CE8,0x7BBB3BDB,
 0x097826CD,0xF418596E,0x01B79AEC,0xA89A4F83,0x656E95E6,0x7EE6FFAA,0x08CFBC21,0xE6E815EF,
 0xD99BE7BA,0xCE366F4A,0xD4099FEA,0xD67CB029,0xAFB2A431,0x31233F2A,0x3094A5C6,0xC066A235,
 0x37BC4E74,0xA6CA82FC,0xB0D090E0,0x15D8A733,0x4A9804F1,0xF7DAEC41,0x0E50CD7F,0x2FF69117,
 0x8DD64D76,0x4DB0EF43,0x544DAACC,0xDF0496E4,0xE3B5D19E,0x1B886A4C,0xB81F2CC1,0x7F516546,
 0x04EA5E9D,0x5D358C01,0x737487FA,0x2E410BFB,0x5A1D67B3,0x52D2DB92,0x335610E9,0x1347D66D,
 0x8C61D79A,0x7A0CA137,0x8E14F859,0x893C13EB,0xEE27A9CE,0x35C961B7,0xEDE51CE1,0x3CB1477A,
 0x59DFD29C,0x3F73F255,0x79CE1418,0xBF37C773,0xEACDF753,0x5BAAFD5F,0x146F3DDF,0x86DB4478,
 0x81F3AFCA,0x3EC468B9,0x2C342438,0x5F40A3C2,0x72C31D16,0x0C25E2BC,0x8B493C28,0x41950DFF,
 0x7101A839,0xDEB30C08,0x9CE4B4D8,0x90C15664,0x6184CB7B,0x70B632D5,0x745C6C48,0x4257B8D0
 },
 {
 0xA7F45150,0x65417E53,0xA4171AC3,0x5E273A96,0x6BAB3BCB,0x459D1FF1,0x58FAACAB,0x03E34B93,
 0xFA302055,0x6D76ADF6,0x76CC8891,0x4C02F525,0xD7E54FFC,0xCB2AC5D7,0x44352680,0xA362B58F,
 0x5AB1DE49,0x1BBA2567,0x0EEA4598,0xC0FE5DE1,0x752FC302,0xF04C8112,0x97468DA3,0xF9D36BC6,
 0x5F8F03E7,0x9C921595,0x7A6DBFEB,0x595295DA,0x83BED42D,0x217458D3,0x69E04929,0xC8C98E44,
 0x89C2756A,0x798EF478,0x3E58996B,0x71B927DD,0x4FE1BEB6,0xAD88F017,0xAC20C966,0x3ACE7DB4,
 0x4ADF6318,0x311AE582,0x33519760,0x7F536245,0x7764B1E0,0xAE6BBB84,0xA081FE1C,0x2B08F994,
 0x68487058,0xFD458F19,0x6CDE9487,0xF87B52B7,0xD373AB23,0x024B72E2,0x8F1FE357,0xAB55662A,
 0x28EBB207,0xC2B52F03,0x7BC5869A,0x0837D3A5,0x872830F2,0xA5BF23B2,0x6A0302BA,0x8216ED5C,
 0x1CCF8A2B,0xB479A792,0xF207F3F0,0xE2694EA1,0xF4DA65CD,0xBE0506D5,0x6234D11F,0xFEA6C48A,
 0x532E349D,0x55F3A2A0,0xE18A0532,0xEBF6A475,0xEC830B39,0xEF6040AA,0x9F715E06,0x106EBD51,
 0x8A213EF9,0x06DD963D,0x053EDDAE,0xBDE64D46,0x8D5491B5,0x5DC47105,0xD406046F,0x155060FF,
 0xFB981924,0xE9BDD697,0x434089CC,0x9ED96777,0x42E8B0BD,0x8B890788,0x5B19E738,0xEEC879DB,
 0x0A7CA147,0x0F427CE9,0x1E84F8C9,0000000000,0x86800983,0xED2B3248,0x70111EAC,0x725A6C4E,
 0xFF0EFDFB,0x38850F56,0xD5AE3D1E,0x392D3627,0xD90F0A64,0xA65C6821,0x545B9BD1,0x2E36243A,
 0x670A0CB1,0xE757930F,0x96EEB4D2,0x919B1B9E,0xC5C0804F,0x20DC61A2,0x4B775A69,0x1A121C16,
 0xBA93E20A,0x2AA0C0E5,0xE0223C43,0x171B121D,0x0D090E0B,0xC78BF2AD,0xA8B62DB9,0xA91E14C8,
 0x19F15785,0x0775AF4C,0xDD99EEBB,0x607FA3FD,0x2601F79F,0xF5725CBC,0x3B6644C5,0x7EFB5B34,
 0x29438B76,0xC623CBDC,0xFCEDB668,0xF1E4B863,0xDC31D7CA,0x85634210,0x22971340,0x11C68420,
 0x244A857D,0x3DBBD2F8,0x32F9AE11,0xA129C76D,0x2F9E1D4B,0x30B2DCF3,0x52860DEC,0xE3C177D0,
 0x16B32B6C,0xB970A999,0x489411FA,0x64E94722,0x8CFCA8C4,0x3FF0A01A,0x2C7D56D8,0x903322EF,
 0x4E4987C7,0xD138D9C1,0xA2CA8CFE,0x0BD49836,0x81F5A6CF,0xDE7AA528,0x8EB7DA26,0xBFAD3FA4,
 0x9D3A2CE4,0x9278500D,0xCC5F6A9B,0x467E5462,0x138DF6C2,0xB8D890E8,0xF7392E5E,0xAFC382F5,
 0x805D9FBE,0x93D0697C,0x2DD56FA9,0x1225CFB3,0x99ACC83B,0x7D1810A7,0x639CE86E,0xBB3BDB7B,
 0x7826CD09,0x18596EF4,0xB79AEC01,0x9A4F83A8,0x6E95E665,0xE6FFAA7E,0xCFBC2108,0xE815EFE6,
 0x9BE7BAD9,0x366F4ACE,0x099FEAD4,0x7CB029D6,0xB2A431AF,0x233F2A31,0x94A5C630,0x66A235C0,
 0xBC4E7437,0xCA82FCA6,0xD090E0B0,0xD8A73315,0x9804F14A,0xDAEC41F7,0x50CD7F0E,0xF691172F,
 0xD64D768D,0xB0EF434D,0x4DAACC54,0x0496E4DF,0xB5D19EE3,0x886A4C1B,0x1F2CC1B8,0x5165467F,
 0xEA5E9D04,0x358C015D,0x7487FA73,0x410BFB2E,0x1D67B35A,0xD2DB9252,0x5610E933,0x47D66D13,
 0x61D79A8C,0x0CA1377A,0x14F8598E,0x3C13EB89,0x27A9CEEE,0xC961B735,0xE51CE1ED,0xB1477A3C,
 0xDFD29C59,0x73F2553F,0xCE141879,0x37C773BF,0xCDF753EA,0xAAFD5F5B,0x6F3DDF14,0xDB447886,
 0xF3AFCA81,0xC468B93E,0x3424382C,0x40A3C25F,0xC31D1672,0x25E2BC0C,0x493C288B,0x950DFF41,
 0x01A83971,0xB30C08DE,0xE4B4D89C,0xC1566490,0x84CB7B61,0xB632D570,0x5C6C4874,0x57B8D042
 },
 {
 0xF45150A7,0x417E5365,0x171AC3A4,0x273A965E,0xAB3BCB6B,0x9D1FF145,0xFAACAB58,0xE34B9303,
 0x302055FA,0x76ADF66D,0xCC889176,0x02F5254C,0xE54FFCD7,0x2AC5D7CB,0x35268044,0x62B58FA3,
 0xB1DE495A,0xBA25671B,0xEA45980E,0xFE5DE1C0,0x2FC30275,0x4C8112F0,0x468DA397,0xD36BC6F9,
 0x8F03E75F,0x9215959C,0x6DBFEB7A,0x5295DA59,0xBED42D83,0x7458D321,0xE0492969,0xC98E44C8,
 0xC2756A89,0x8EF47879,0x58996B3E,0xB927DD71,0xE1BEB64F,0x88F017AD,0x20C966AC,0xCE7DB43A,
 0xDF63184A,0x1AE58231,0x51976033,0x5362457F,0x64B1E077,0x6BBB84AE,0x81FE1CA0,0x08F9942B,
 0x48705868,0x458F19FD,0xDE94876C,0x7B52B7F8,0x73AB23D3,0x4B72E202,0x1FE3578F,0x55662AAB,
 0xEBB20728,0xB52F03C2,0xC5869A7B,0x37D3A508,0x2830F287,0xBF23B2A5,0x0302BA6A,0x16ED5C82,
 0xCF8A2B1C,0x79A792B4,0x07F3F0F2,0x694EA1E2,0xDA65CDF4,0x0506D5BE,0x34D11F62,0xA6C48AFE,
 0x2E349D53,0xF3A2A055,0x8A0532E1,0xF6A475EB,0x830B39EC,0x6040AAEF,0x715E069F,0x6EBD5110,
 0x213EF98A,0xDD963D06,0x3EDDAE05,0xE64D46BD,0x5491B58D,0xC471055D,0x06046FD4,0x5060FF15,
 0x981924FB,0xBDD697E9,0x4089CC43,0xD967779E,0xE8B0BD42,0x8907888B,0x19E7385B,0xC879DBEE,
 0x7CA1470A,0x427CE90F,0x84F8C91E,0000000000,0x80098386,0x2B3248ED,0x111EAC70,0x5A6C4E72,
 0x0EFDFBFF,0x850F5638,0xAE3D1ED5,0x2D362739,0x0F0A64D9,0x5C6821A6,0x5B9BD154,0x36243A2E,
 0x0A0CB167,0x57930FE7,0xEEB4D296,0x9B1B9E91,0xC0804FC5,0xDC61A220,0x775A694B,0x121C161A,
 0x93E20ABA,0xA0C0E52A,0x223C43E0,0x1B121D17,0x090E0B0D,0x8BF2ADC7,0xB62DB9A8,0x1E14C8A9,
 0xF1578519,0x75AF4C07,0x99EEBBDD,0x7FA3FD60,0x01F79F26,0x725CBCF5,0x6644C53B,0xFB5B347E,
 0x438B7629,0x23CBDCC6,0xEDB668FC,0xE4B863F1,0x31D7CADC,0x63421085,0x97134022,0xC6842011,
 0x4A857D24,0xBBD2F83D,0xF9AE1132,0x29C76DA1,0x9E1D4B2F,0xB2DCF330,0x860DEC52,0xC177D0E3,
 0xB32B6C16,0x70A999B9,0x9411FA48,0xE9472264,0xFCA8C48C,0xF0A01A3F,0x7D56D82C,0x3322EF90,
 0x4987C74E,0x38D9C1D1,0xCA8CFEA2,0xD498360B,0xF5A6CF81,0x7AA528DE,0xB7DA268E,0xAD3FA4BF,
 0x3A2CE49D,0x78500D92,0x5F6A9BCC,0x7E546246,0x8DF6C213,0xD890E8B8,0x392E5EF7,0xC382F5AF,
 0x5D9FBE80,0xD0697C93,0xD56FA92D,0x25CFB312,0xACC83B99,0x1810A77D,0x9CE86E63,0x3BDB7BBB,
 0x26CD0978,0x596EF418,0x9AEC01B7,0x4F83A89A,0x95E6656E,0xFFAA7EE6,0xBC2108CF,0x15EFE6E8,
 0xE7BAD99B,0x6F4ACE36,0x9FEAD409,0xB029D67C,0xA431AFB2,0x3F2A3123,0xA5C63094,0xA235C066,
 0x4E7437BC,0x82FCA6CA,0x90E0B0D0,0xA73315D8,0x04F14A98,0xEC41F7DA,0xCD7F0E50,0x91172FF6,
 0x4D768DD6,0xEF434DB0,0xAACC544D,0x96E4DF04,0xD19EE3B5,0x6A4C1B88,0x2CC1B81F,0x65467F51,
 0x5E9D04EA,0x8C015D35,0x87FA7374,0x0BFB2E41,0x67B35A1D,0xDB9252D2,0x10E93356,0xD66D1347,
 0xD79A8C61,0xA1377A0C,0xF8598E14,0x13EB893C,0xA9CEEE27,0x61B735C9,0x1CE1EDE5,0x477A3CB1,
 0xD29C59DF,0xF2553F73,0x141879CE,0xC773BF37,0xF753EACD,0xFD5F5BAA,0x3DDF146F,0x447886DB,
 0xAFCA81F3,0x68B93EC4,0x24382C34,0xA3C25F40,0x1D1672C3,0xE2BC0C25,0x3C288B49,0x0DFF4195,
 0xA8397101,0x0C08DEB3,0xB4D89CE4,0x566490C1,0xCB7B6184,0x32D570B6,0x6C48745C,0xB8D04257
 },
 {
 0x5150A7F4,0x7E536541,0x1AC3A417,0x3A965E27,0x3BCB6BAB,0x1FF1459D,0xACAB58FA,0x4B9303E3,
 0x2055FA30,0xADF66D76,0x889176CC,0xF5254C02,0x4FFCD7E5,0xC5D7CB2A,0x26804435,0xB58FA362,
 0xDE495AB1,0x25671BBA,0x45980EEA,0x5DE1C0FE,0xC302752F,0x8112F04C,0x8DA39746,0x6BC6F9D3,
 0x03E75F8F,0x15959C92,0xBFEB7A6D,0x95DA5952,0xD42D83BE,0x58D32174,0x492969E0,0x8E44C8C9,
 0x756A89C2,0xF478798E,0x996B3E58,0x27DD71B9,0xBEB64FE1,0xF017AD88,0xC966AC20,0x7DB43ACE,
 0x63184ADF,0xE582311A,0x97603351,0x62457F53,0xB1E07764,0xBB84AE6B,0xFE1CA081,0xF9942B08,
 0x70586848,0x8F19FD45,0x94876CDE,0x52B7F87B,0xAB23D373,0x72E2024B,0xE3578F1F,0x662AAB55,
 0xB20728EB,0x2F03C2B5,0x869A7BC5,0xD3A50837,0x30F28728,0x23B2A5BF,0x02BA6A03,0xED5C8216,
 0x8A2B1CCF,0xA792B479,0xF3F0F207,0x4EA1E269,0x65CDF4DA,0x06D5BE05,0xD11F6234,0xC48AFEA6,
 0x349D532E,0xA2A055F3,0x0532E18A,0xA475EBF6,0x0B39EC83,0x40AAEF60,0x5E069F71,0xBD51106E,
 0x3EF98A21,0x963D06DD,0xDDAE053E,0x4D46BDE6,0x91B58D54,0x71055DC4,0x046FD406,0x60FF1550,
 0x1924FB98,0xD697E9BD,0x89CC4340,0x67779ED9,0xB0BD42E8,0x07888B89,0xE7385B19,0x79DBEEC8,
 0xA1470A7C,0x7CE90F42,0xF8C91E84,0000000000,0x09838680,0x3248ED2B,0x1EAC7011,0x6C4E725A,
 0xFDFBFF0E,0x0F563885,0x3D1ED5AE,0x3627392D,0x0A64D90F,0x6821A65C,0x9BD1545B,0x243A2E36,
 0x0CB1670A,0x930FE757,0xB4D296EE,0x1B9E919B,0x804FC5C0,0x61A220DC,0x5A694B77,0x1C161A12,
 0xE20ABA93,0xC0E52AA0,0x3C43E022,0x121D171B,0x0E0B0D09,0xF2ADC78B,0x2DB9A8B6,0x14C8A91E,
 0x578519F1,0xAF4C0775,0xEEBBDD99,0xA3FD607F,0xF79F2601,0x5CBCF572,0x44C53B66,0x5B347EFB,
 0x8B762943,0xCBDCC623,0xB668FCED,0xB863F1E4,0xD7CADC31,0x42108563,0x13402297,0x842011C6,
 0x857D244A,0xD2F83DBB,0xAE1132F9,0xC76DA129,0x1D4B2F9E,0xDCF330B2,0x0DEC5286,0x77D0E3C1,
 0x2B6C16B3,0xA999B970,0x11FA4894,0x472264E9,0xA8C48CFC,0xA01A3FF0,0x56D82C7D,0x22EF9033,
 0x87C74E49,0xD9C1D138,0x8CFEA2CA,0x98360BD4,0xA6CF81F5,0xA528DE7A,0xDA268EB7,0x3FA4BFAD,
 0x2CE49D3A,0x500D9278,0x6A9BCC5F,0x5462467E,0xF6C2138D,0x90E8B8D8,0x2E5EF739,0x82F5AFC3,
 0x9FBE805D,0x697C93D0,0x6FA92DD5,0xCFB31225,0xC83B99AC,0x10A77D18,0xE86E639C,0xDB7BBB3B,
 0xCD097826,0x6EF41859,0xEC01B79A,0x83A89A4F,0xE6656E95,0xAA7EE6FF,0x2108CFBC,0xEFE6E815,
 0xBAD99BE7,0x4ACE366F,0xEAD4099F,0x29D67CB0,0x31AFB2A4,0x2A31233F,0xC63094A5,0x35C066A2,
 0x7437BC4E,0xFCA6CA82,0xE0B0D090,0x3315D8A7,0xF14A9804,0x41F7DAEC,0x7F0E50CD,0x172FF691,
 0x768DD64D,0x434DB0EF,0xCC544DAA,0xE4DF0496,0x9EE3B5D1,0x4C1B886A,0xC1B81F2C,0x467F5165,
 0x9D04EA5E,0x015D358C,0xFA737487,0xFB2E410B,0xB35A1D67,0x9252D2DB,0xE9335610,0x6D1347D6,
 0x9A8C61D7,0x377A0CA1,0x598E14F8,0xEB893C13,0xCEEE27A9,0xB735C961,0xE1EDE51C,0x7A3CB147,
 0x9C59DFD2,0x553F73F2,0x1879CE14,0x73BF37C7,0x53EACDF7,0x5F5BAAFD,0xDF146F3D,0x7886DB44,
 0xCA81F3AF,0xB93EC468,0x382C3424,0xC25F40A3,0x1672C31D,0xBC0C25E2,0x288B493C,0xFF41950D,
 0x397101A8,0x08DEB30C,0xD89CE4B4,0x6490C156,0x7B6184CB,0xD570B632,0x48745C6C,0xD04257B8
 }
};

static u32 fl_tab[4][256] =
{
```

```
{
0x00000063,0x0000007C,0x00000077,0x0000007B,0x000000F2,0x0000006B,0x0000006F,0x000000C5,
0x00000030,0x00000001,0x00000067,0x0000002B,0x000000FE,0x000000D7,0x000000AB,0x00000076,
0x000000CA,0x00000082,0x000000C9,0x0000007D,0x000000FA,0x00000059,0x00000047,0x000000F0,
0x000000AD,0x000000D4,0x000000A2,0x000000AF,0x0000009C,0x000000A4,0x00000072,0x000000C0,
0x000000B7,0x000000FD,0x00000093,0x00000026,0x00000036,0x0000003F,0x000000F7,0x000000CC,
0x00000034,0x000000A5,0x000000E5,0x000000F1,0x00000071,0x000000D8,0x00000031,0x00000015,
0x00000004,0x000000C7,0x00000023,0x000000C3,0x00000018,0x00000096,0x00000005,0x0000009A,
0x00000007,0x00000012,0x00000080,0x000000E2,0x000000EB,0x00000027,0x000000B2,0x00000075,
0x00000009,0x00000083,0x0000002C,0x0000001A,0x0000001B,0x0000006E,0x0000005A,0x000000A0,
0x00000052,0x0000003B,0x000000D6,0x000000B3,0x00000029,0x000000E3,0x0000002F,0x00000084,
0x00000053,0x000000D1,0x00000000,0x000000ED,0x00000020,0x000000FC,0x000000B1,0x0000005B,
0x0000006A,0x000000CB,0x000000BE,0x00000039,0x0000004A,0x0000004C,0x00000058,0x000000CF,
0x000000D0,0x000000EF,0x000000AA,0x000000FB,0x00000043,0x0000004D,0x00000033,0x00000085,
0x00000045,0x000000F9,0x00000002,0x0000007F,0x00000050,0x0000003C,0x0000009F,0x000000A8,
0x00000051,0x000000A3,0x00000040,0x0000008F,0x00000092,0x0000009D,0x00000038,0x000000F5,
0x000000BC,0x000000B6,0x000000DA,0x00000021,0x00000010,0x000000FF,0x000000F3,0x000000D2,
0x000000CD,0x0000000C,0x00000013,0x000000EC,0x0000005F,0x00000097,0x00000044,0x00000017,
0x000000C4,0x000000A7,0x0000007E,0x0000003D,0x00000064,0x0000005D,0x00000019,0x00000073,
0x00000060,0x00000081,0x0000004F,0x000000DC,0x00000022,0x0000002A,0x00000090,0x00000088,
0x00000046,0x000000EE,0x000000B8,0x00000014,0x000000DE,0x0000005E,0x0000000B,0x000000DB,
0x000000E0,0x00000032,0x0000003A,0x0000000A,0x00000049,0x00000006,0x00000024,0x0000005C,
0x000000C2,0x000000D3,0x000000AC,0x00000062,0x00000091,0x00000095,0x000000E4,0x00000079,
0x000000E7,0x000000C8,0x00000037,0x0000006D,0x0000008D,0x000000D5,0x0000004E,0x000000A9,
0x0000006C,0x00000056,0x000000F4,0x000000EA,0x00000065,0x0000007A,0x000000AE,0x00000008,
0x000000BA,0x00000078,0x00000025,0x0000002E,0x0000001C,0x000000A6,0x000000B4,0x000000C6,
0x000000E8,0x000000DD,0x00000074,0x0000001F,0x0000004B,0x000000BD,0x0000008B,0x0000008A,
0x00000070,0x0000003E,0x000000B5,0x00000066,0x00000048,0x00000003,0x000000F6,0x0000000E,
0x00000061,0x00000035,0x00000057,0x000000B9,0x00000086,0x000000C1,0x0000001D,0x0000009E,
0x000000E1,0x000000F8,0x00000098,0x00000011,0x00000069,0x000000D9,0x0000008E,0x00000094,
0x0000009B,0x0000001E,0x00000087,0x000000E9,0x000000CE,0x00000055,0x00000028,0x000000DF,
0x0000008C,0x000000A1,0x00000089,0x0000000D,0x000000BF,0x000000E6,0x00000042,0x00000068,
0x00000041,0x00000099,0x0000002D,0x0000000F,0x000000B0,0x00000054,0x000000BB,0x00000016
},
{
0x00006300,0x00007C00,0x00007700,0x00007B00,0x0000F200,0x00006B00,0x00006F00,0x0000C500,
0x00003000,0x00000100,0x00006700,0x00002B00,0x0000FE00,0x0000D700,0x0000AB00,0x00007600,
0x0000CA00,0x00008200,0x0000C900,0x00007D00,0x0000FA00,0x00005900,0x00004700,0x0000F000,
0x0000AD00,0x0000D400,0x0000A200,0x0000AF00,0x00009C00,0x0000A400,0x00007200,0x0000C000,
0x0000B700,0x0000FD00,0x00009300,0x00002600,0x00003600,0x00003F00,0x0000F700,0x0000CC00,
0x00003400,0x0000A500,0x0000E500,0x0000F100,0x00007100,0x0000D800,0x00003100,0x00001500,
0x00000400,0x0000C700,0x00002300,0x0000C300,0x00001800,0x00009600,0x00000500,0x00009A00,
0x00000700,0x00001200,0x00008000,0x0000E200,0x0000EB00,0x00002700,0x0000B200,0x00007500,
0x00000900,0x00008300,0x00002C00,0x00001A00,0x00001B00,0x00006E00,0x00005A00,0x0000A000,
0x00005200,0x00003B00,0x0000D600,0x0000B300,0x00002900,0x0000E300,0x00002F00,0x00008400,
0x00005300,0x0000D100,0000000000,0x0000ED00,0x00002000,0x0000FC00,0x0000B100,0x00005B00,
0x00006A00,0x0000CB00,0x0000BE00,0x00003900,0x00004A00,0x00004C00,0x00005800,0x0000CF00,
0x0000D000,0x0000EF00,0x0000AA00,0x0000FB00,0x00004300,0x00004D00,0x00003300,0x00008500,
0x00004500,0x0000F900,0x00000200,0x00007F00,0x00005000,0x00003C00,0x00009F00,0x0000A800,
0x00005100,0x0000A300,0x00004000,0x00008F00,0x00009200,0x00009D00,0x00003800,0x0000F500,
0x0000BC00,0x0000B600,0x0000DA00,0x00002100,0x00001000,0x0000FF00,0x0000F300,0x0000D200,
0x0000CD00,0x00000C00,0x00001300,0x0000EC00,0x00005F00,0x00009700,0x00004400,0x00001700,
0x0000C400,0x0000A700,0x00007E00,0x00003D00,0x00006400,0x00005D00,0x00001900,0x00007300,
0x00006000,0x00008100,0x00004F00,0x0000DC00,0x00002200,0x00002A00,0x00009000,0x00008800,
0x00004600,0x0000EE00,0x0000B800,0x00001400,0x0000DE00,0x00005E00,0x00000B00,0x0000DB00,
0x0000E000,0x00003200,0x00003A00,0x00000A00,0x00004900,0x00000600,0x00002400,0x00005C00,
0x0000C200,0x0000D300,0x0000AC00,0x00006200,0x00009100,0x00009500,0x0000E400,0x00007900,
0x0000E700,0x0000C800,0x00003700,0x00006D00,0x00008D00,0x0000D500,0x00004E00,0x0000A900,
0x00006C00,0x00005600,0x0000F400,0x0000EA00,0x00006500,0x00007A00,0x0000AE00,0x00000800,
0x0000BA00,0x00007800,0x00002500,0x00002E00,0x00001C00,0x0000A600,0x0000B400,0x0000C600,
0x0000E800,0x0000DD00,0x00007400,0x00001F00,0x00004B00,0x0000BD00,0x00008B00,0x00008A00,
0x00007000,0x00003E00,0x0000B500,0x00006600,0x00004800,0x00000300,0x0000F600,0x00000E00,
0x00006100,0x00003500,0x00005700,0x0000B900,0x00008600,0x0000C100,0x00001D00,0x00009E00,
0x0000E100,0x0000F800,0x00009800,0x00001100,0x00006900,0x0000D900,0x00008E00,0x00009400,
0x00009B00,0x00001E00,0x00008700,0x0000E900,0x0000CE00,0x00005500,0x00002800,0x0000DF00,
0x00008C00,0x0000A100,0x00008900,0x00000D00,0x0000BF00,0x0000E600,0x00004200,0x00006800,
0x00004100,0x00009900,0x00002D00,0x00000F00,0x0000B000,0x00005400,0x0000BB00,0x00001600
},
{
0x00630000,0x007C0000,0x00770000,0x007B0000,0x00F20000,0x006B0000,0x006F0000,0x00C50000,
0x00300000,0x00010000,0x00670000,0x002B0000,0x00FE0000,0x00D70000,0x00AB0000,0x00760000,
0x00CA0000,0x00820000,0x00C90000,0x007D0000,0x00FA0000,0x00590000,0x00470000,0x00F00000,
0x00AD0000,0x00D40000,0x00A20000,0x00AF0000,0x009C0000,0x00A40000,0x00720000,0x00C00000,
0x00B70000,0x00FD0000,0x00930000,0x00260000,0x00360000,0x003F0000,0x00F70000,0x00CC0000,
0x00340000,0x00A50000,0x00E50000,0x00F10000,0x00710000,0x00D80000,0x00310000,0x00150000,
0x00040000,0x00C70000,0x00230000,0x00C30000,0x00180000,0x00960000,0x00050000,0x009A0000,
 0x00070000,0x00120000,0x00800000,0x00E20000,0x00EB0000,0x00270000,0x00B20000,0x00750000,
 0x00090000,0x00830000,0x002C0000,0x001A0000,0x001B0000,0x006E0000,0x005A0000,0x00A00000,
 0x00520000,0x003B0000,0x00D60000,0x00B30000,0x00290000,0x00E30000,0x002F0000,0x00840000,
 0x00530000,0x00D10000,0000000000,0x00ED0000,0x00200000,0x00FC0000,0x00B10000,0x005B0000,
 0x006A0000,0x00CB0000,0x00BE0000,0x00390000,0x004A0000,0x004C0000,0x00580000,0x00CF0000,
 0x00D00000,0x00EF0000,0x00AA0000,0x00FB0000,0x00430000,0x004D0000,0x00330000,0x00850000,
 0x00450000,0x00F90000,0x00020000,0x007F0000,0x00500000,0x003C0000,0x009F0000,0x00A80000,
 0x00510000,0x00A30000,0x00400000,0x008F0000,0x00920000,0x009D0000,0x00380000,0x00F50000,
 0x00BC0000,0x00B60000,0x00DA0000,0x00210000,0x00100000,0x00FF0000,0x00F30000,0x00D20000,
 0x00CD0000,0x000C0000,0x00130000,0x00EC0000,0x005F0000,0x00970000,0x00440000,0x00170000,
 0x00C40000,0x00A70000,0x007E0000,0x003D0000,0x00640000,0x005D0000,0x00190000,0x00730000,
 0x00600000,0x00810000,0x004F0000,0x00DC0000,0x00220000,0x002A0000,0x00900000,0x00880000,
 0x00460000,0x00EE0000,0x00B80000,0x00140000,0x00DE0000,0x005E0000,0x000B0000,0x00DB0000,
 0x00E00000,0x00320000,0x003A0000,0x000A0000,0x00490000,0x00060000,0x00240000,0x005C0000,
 0x00C20000,0x00D30000,0x00AC0000,0x00620000,0x00910000,0x00950000,0x00E40000,0x00790000,
 0x00E70000,0x00C80000,0x00370000,0x006D0000,0x008D0000,0x00D50000,0x004E0000,0x00A90000,
 0x006C0000,0x00560000,0x00F40000,0x00EA0000,0x00650000,0x007A0000,0x00AE0000,0x00080000,
 0x00BA0000,0x00780000,0x00250000,0x002E0000,0x001C0000,0x00A60000,0x00B40000,0x00C60000,
 0x00E80000,0x00DD0000,0x00740000,0x001F0000,0x004B0000,0x00BD0000,0x008B0000,0x008A0000,
 0x00700000,0x003E0000,0x00B50000,0x00660000,0x00480000,0x00030000,0x00F60000,0x000E0000,
 0x00610000,0x00350000,0x00570000,0x00B90000,0x00860000,0x00C10000,0x001D0000,0x009E0000,
 0x00E10000,0x00F80000,0x00980000,0x00110000,0x00690000,0x00D90000,0x008E0000,0x00940000,
 0x009B0000,0x001E0000,0x00870000,0x00E90000,0x00CE0000,0x00550000,0x00280000,0x00DF0000,
 0x008C0000,0x00A10000,0x00890000,0x000D0000,0x00BF0000,0x00E60000,0x00420000,0x00680000,
 0x00410000,0x00990000,0x002D0000,0x000F0000,0x00B00000,0x00540000,0x00BB0000,0x00160000
 },
 {
 0x63000000,0x7C000000,0x77000000,0x7B000000,0xF2000000,0x6B000000,0x6F000000,0xC5000000,
 0x30000000,0x01000000,0x67000000,0x2B000000,0xFE000000,0xD7000000,0xAB000000,0x76000000,
 0xCA000000,0x82000000,0xC9000000,0x7D000000,0xFA000000,0x59000000,0x47000000,0xF0000000,
 0xAD000000,0xD4000000,0xA2000000,0xAF000000,0x9C000000,0xA4000000,0x72000000,0xC0000000,
 0xB7000000,0xFD000000,0x93000000,0x26000000,0x36000000,0x3F000000,0xF7000000,0xCC000000,
 0x34000000,0xA5000000,0xE5000000,0xF1000000,0x71000000,0xD8000000,0x31000000,0x15000000,
 0x04000000,0xC7000000,0x23000000,0xC3000000,0x18000000,0x96000000,0x05000000,0x9A000000,
 0x07000000,0x12000000,0x80000000,0xE2000000,0xEB000000,0x27000000,0xB2000000,0x75000000,
 0x09000000,0x83000000,0x2C000000,0x1A000000,0x1B000000,0x6E000000,0x5A000000,0xA0000000,
 0x52000000,0x3B000000,0xD6000000,0xB3000000,0x29000000,0xE3000000,0x2F000000,0x84000000,
 0x53000000,0xD1000000,0000000000,0xED000000,0x20000000,0xFC000000,0xB1000000,0x5B000000,
 0x6A000000,0xCB000000,0xBE000000,0x39000000,0x4A000000,0x4C000000,0x58000000,0xCF000000,
 0xD0000000,0xEF000000,0xAA000000,0xFB000000,0x43000000,0x4D000000,0x33000000,0x85000000,
 0x45000000,0xF9000000,0x02000000,0x7F000000,0x50000000,0x3C000000,0x9F000000,0xA8000000,
 0x51000000,0xA3000000,0x40000000,0x8F000000,0x92000000,0x9D000000,0x38000000,0xF5000000,
 0xBC000000,0xB6000000,0xDA000000,0x21000000,0x10000000,0xFF000000,0xF3000000,0xD2000000,
 0xCD000000,0x0C000000,0x13000000,0xEC000000,0x5F000000,0x97000000,0x44000000,0x17000000,
 0xC4000000,0xA7000000,0x7E000000,0x3D000000,0x64000000,0x5D000000,0x19000000,0x73000000,
 0x60000000,0x81000000,0x4F000000,0xDC000000,0x22000000,0x2A000000,0x90000000,0x88000000,
 0x46000000,0xEE000000,0xB8000000,0x14000000,0xDE000000,0x5E000000,0x0B000000,0xDB000000,
 0xE0000000,0x32000000,0x3A000000,0x0A000000,0x49000000,0x06000000,0x24000000,0x5C000000,
 0xC2000000,0xD3000000,0xAC000000,0x62000000,0x91000000,0x95000000,0xE4000000,0x79000000,
 0xE7000000,0xC8000000,0x37000000,0x6D000000,0x8D000000,0xD5000000,0x4E000000,0xA9000000,
 0x6C000000,0x56000000,0xF4000000,0xEA000000,0x65000000,0x7A000000,0xAE000000,0x08000000,
 0xBA000000,0x78000000,0x25000000,0x2E000000,0x1C000000,0xA6000000,0xB4000000,0xC6000000,
 0xE8000000,0xDD000000,0x74000000,0x1F000000,0x4B000000,0xBD000000,0x8B000000,0x8A000000,
 0x70000000,0x3E000000,0xB5000000,0x66000000,0x48000000,0x03000000,0xF6000000,0x0E000000,
 0x61000000,0x35000000,0x57000000,0xB9000000,0x86000000,0xC1000000,0x1D000000,0x9E000000,
 0xE1000000,0xF8000000,0x98000000,0x11000000,0x69000000,0xD9000000,0x8E000000,0x94000000,
 0x9B000000,0x1E000000,0x87000000,0xE9000000,0xCE000000,0x55000000,0x28000000,0xDF000000,
 0x8C000000,0xA1000000,0x89000000,0x0D000000,0xBF000000,0xE6000000,0x42000000,0x68000000,
 0x41000000,0x99000000,0x2D000000,0x0F000000,0xB0000000,0x54000000,0xBB000000,0x16000000
 }
};

static u32 il_tab[4][256] =
{
 {
 0x00000052,0x00000009,0x0000006A,0x000000D5,0x00000030,0x00000036,0x000000A5,0x00000038,
 0x000000BF,0x00000040,0x000000A3,0x0000009E,0x00000081,0x000000F3,0x000000D7,0x000000FB,
 0x0000007C,0x000000E3,0x00000039,0x00000082,0x0000009B,0x0000002F,0x000000FF,0x00000087,
 0x00000034,0x0000008E,0x00000043,0x00000044,0x000000C4,0x000000DE,0x000000E9,0x000000CB,
 0x00000054,0x0000007B,0x00000094,0x00000032,0x000000A6,0x000000C2,0x00000023,0x0000003D,
 0x000000EE,0x0000004C,0x00000095,0x0000000B,0x00000042,0x000000FA,0x000000C3,0x0000004E,
 0x00000008,0x0000002E,0x000000A1,0x00000066,0x00000028,0x000000D9,0x00000024,0x000000B2,
 0x00000076,0x0000005B,0x000000A2,0x00000049,0x0000006D,0x0000008B,0x000000D1,0x00000025,
 0x00000072,0x000000F8,0x000000F6,0x00000064,0x00000086,0x00000068,0x00000098,0x00000016,
 0x000000D4,0x000000A4,0x0000005C,0x000000CC,0x0000005D,0x00000065,0x000000B6,0x00000092,
 0x0000006C,0x00000070,0x00000048,0x00000050,0x000000FD,0x000000ED,0x000000B9,0x000000DA,
 0x0000005E,0x00000015,0x00000046,0x00000057,0x000000A7,0x0000008D,0x0000009D,0x00000084,
 0x00000090,0x000000D8,0x000000AB,0000000000,0x0000008C,0x000000BC,0x000000D3,0x0000000A,
 0x000000F7,0x000000E4,0x00000058,0x00000005,0x000000B8,0x000000B3,0x00000045,0x00000006,
 0x000000D0,0x0000002C,0x0000001E,0x0000008F,0x000000CA,0x0000003F,0x0000000F,0x00000002,
 0x000000C1,0x000000AF,0x000000BD,0x00000003,0x00000001,0x00000013,0x0000008A,0x0000006B,
 0x0000003A,0x00000091,0x00000011,0x00000041,0x0000004F,0x00000067,0x000000DC,0x000000EA,
 0x00000097,0x000000F2,0x000000CF,0x000000CE,0x000000F0,0x000000B4,0x000000E6,0x00000073,
 0x00000096,0x000000AC,0x00000074,0x00000022,0x000000E7,0x000000AD,0x00000035,0x00000085,
 0x000000E2,0x000000F9,0x00000037,0x000000E8,0x0000001C,0x00000075,0x000000DF,0x0000006E,
 0x00000047,0x000000F1,0x0000001A,0x00000071,0x0000001D,0x00000029,0x000000C5,0x00000089,
 0x0000006F,0x000000B7,0x00000062,0x0000000E,0x000000AA,0x00000018,0x000000BE,0x0000001B,
 0x000000FC,0x00000056,0x0000003E,0x0000004B,0x000000C6,0x000000D2,0x00000079,0x00000020,
 0x0000009A,0x000000DB,0x000000C0,0x000000FE,0x00000078,0x000000CD,0x0000005A,0x000000F4,
 0x0000001F,0x000000DD,0x000000A8,0x00000033,0x00000088,0x00000007,0x000000C7,0x00000031,
 0x000000B1,0x00000012,0x00000010,0x00000059,0x00000027,0x00000080,0x000000EC,0x0000005F,
 0x00000060,0x00000051,0x0000007F,0x000000A9,0x00000019,0x000000B5,0x0000004A,0x0000000D,
 0x0000002D,0x000000E5,0x0000007A,0x0000009F,0x00000093,0x000000C9,0x0000009C,0x000000EF,
 0x000000A0,0x000000E0,0x0000003B,0x0000004D,0x000000AE,0x0000002A,0x000000F5,0x000000B0,
 0x000000C8,0x000000EB,0x000000BB,0x0000003C,0x00000083,0x00000053,0x00000099,0x00000061,
 0x00000017,0x0000002B,0x00000004,0x0000007E,0x000000BA,0x00000077,0x000000D6,0x00000026,
 0x000000E1,0x00000069,0x00000014,0x00000063,0x00000055,0x00000021,0x0000000C,0x0000007D
 },
 {
 0x00005200,0x00000900,0x00006A00,0x0000D500,0x00003000,0x00003600,0x0000A500,0x00003800,
 0x0000BF00,0x00004000,0x0000A300,0x00009E00,0x00008100,0x0000F300,0x0000D700,0x0000FB00,
 0x00007C00,0x0000E300,0x00003900,0x00008200,0x00009B00,0x00002F00,0x0000FF00,0x00008700,
 0x00003400,0x00008E00,0x00004300,0x00004400,0x0000C400,0x0000DE00,0x0000E900,0x0000CB00,
 0x00005400,0x00007B00,0x00009400,0x00003200,0x0000A600,0x0000C200,0x00002300,0x00003D00,
 0x0000EE00,0x00004C00,0x00009500,0x00000B00,0x00004200,0x0000FA00,0x0000C300,0x00004E00,
 0x00000800,0x00002E00,0x0000A100,0x00006600,0x00002800,0x0000D900,0x00002400,0x0000B200,
 0x00007600,0x00005B00,0x0000A200,0x00004900,0x00006D00,0x00008B00,0x0000D100,0x00002500,
 0x00007200,0x0000F800,0x0000F600,0x00006400,0x00008600,0x00006800,0x00009800,0x00001600,
 0x0000D400,0x0000A400,0x00005C00,0x0000CC00,0x00005D00,0x00006500,0x0000B600,0x00009200,
 0x00006C00,0x00007000,0x00004800,0x00005000,0x0000FD00,0x0000ED00,0x0000B900,0x0000DA00,
 0x00005E00,0x00001500,0x00004600,0x00005700,0x0000A700,0x00008D00,0x00009D00,0x00008400,
 0x00009000,0x0000D800,0x0000AB00,0000000000,0x00008C00,0x0000BC00,0x0000D300,0x00000A00,
 0x0000F700,0x0000E400,0x00005800,0x00000500,0x0000B800,0x0000B300,0x00004500,0x00000600,
 0x0000D000,0x00002C00,0x00001E00,0x00008F00,0x0000CA00,0x00003F00,0x00000F00,0x00000200,
 0x0000C100,0x0000AF00,0x0000BD00,0x00000300,0x00000100,0x00001300,0x00008A00,0x00006B00,
 0x00003A00,0x00009100,0x00001100,0x00004100,0x00004F00,0x00006700,0x0000DC00,0x0000EA00,
 0x00009700,0x0000F200,0x0000CF00,0x0000CE00,0x0000F000,0x0000B400,0x0000E600,0x00007300,
 0x00009600,0x0000AC00,0x00007400,0x00002200,0x0000E700,0x0000AD00,0x00003500,0x00008500,
 0x0000E200,0x0000F900,0x00003700,0x0000E800,0x00001C00,0x00007500,0x0000DF00,0x00006E00,
 0x00004700,0x0000F100,0x00001A00,0x00007100,0x00001D00,0x00002900,0x0000C500,0x00008900,
 0x00006F00,0x0000B700,0x00006200,0x00000E00,0x0000AA00,0x00001800,0x0000BE00,0x00001B00,
 0x0000FC00,0x00005600,0x00003E00,0x00004B00,0x0000C600,0x0000D200,0x00007900,0x00002000,
 0x00009A00,0x0000DB00,0x0000C000,0x0000FE00,0x00007800,0x0000CD00,0x00005A00,0x0000F400,
 0x00001F00,0x0000DD00,0x0000A800,0x00003300,0x00008800,0x00000700,0x0000C700,0x00003100,
 0x0000B100,0x00001200,0x00001000,0x00005900,0x00002700,0x00008000,0x0000EC00,0x00005F00,
 0x00006000,0x00005100,0x00007F00,0x0000A900,0x00001900,0x0000B500,0x00004A00,0x00000D00,
 0x00002D00,0x0000E500,0x00007A00,0x00009F00,0x00009300,0x0000C900,0x00009C00,0x0000EF00,
 0x0000A000,0x0000E000,0x00003B00,0x00004D00,0x0000AE00,0x00002A00,0x0000F500,0x0000B000,
 0x0000C800,0x0000EB00,0x0000BB00,0x00003C00,0x00008300,0x00005300,0x00009900,0x00006100,
 0x00001700,0x00002B00,0x00000400,0x00007E00,0x0000BA00,0x00007700,0x0000D600,0x00002600,
 0x0000E100,0x00006900,0x00001400,0x00006300,0x00005500,0x00002100,0x00000C00,0x00007D00
 },
 {
 0x00520000,0x00090000,0x006A0000,0x00D50000,0x00300000,0x00360000,0x00A50000,0x00380000,
 0x00BF0000,0x00400000,0x00A30000,0x009E0000,0x00810000,0x00F30000,0x00D70000,0x00FB0000,
 0x007C0000,0x00E30000,0x00390000,0x00820000,0x009B0000,0x002F0000,0x00FF0000,0x00870000,
 0x00340000,0x008E0000,0x00430000,0x00440000,0x00C40000,0x00DE0000,0x00E90000,0x00CB0000,
 0x00540000,0x007B0000,0x00940000,0x00320000,0x00A60000,0x00C20000,0x00230000,0x003D0000,
 0x00EE0000,0x004C0000,0x00950000,0x000B0000,0x00420000,0x00FA0000,0x00C30000,0x004E0000,
 0x00080000,0x002E0000,0x00A10000,0x00660000,0x00280000,0x00D90000,0x00240000,0x00B20000,
 0x00760000,0x005B0000,0x00A20000,0x00490000,0x006D0000,0x008B0000,0x00D10000,0x00250000,
 0x00720000,0x00F80000,0x00F60000,0x00640000,0x00860000,0x00680000,0x00980000,0x00160000,
 0x00D40000,0x00A40000,0x005C0000,0x00CC0000,0x005D0000,0x00650000,0x00B60000,0x00920000,
 0x006C0000,0x00700000,0x00480000,0x00500000,0x00FD0000,0x00ED0000,0x00B90000,0x00DA0000,
 0x005E0000,0x00150000,0x00460000,0x00570000,0x00A70000,0x008D0000,0x009D0000,0x00840000,
 0x00900000,0x00D80000,0x00AB0000,0000000000,0x008C0000,0x00BC0000,0x00D30000,0x000A0000,
 0x00F70000,0x00E40000,0x00580000,0x00050000,0x00B80000,0x00B30000,0x00450000,0x00060000,
 0x00D00000,0x002C0000,0x001E0000,0x008F0000,0x00CA0000,0x003F0000,0x000F0000,0x00020000,
 0x00C10000,0x00AF0000,0x00BD0000,0x00030000,0x00010000,0x00130000,0x008A0000,0x006B0000,
 0x003A0000,0x00910000,0x00110000,0x00410000,0x004F0000,0x00670000,0x00DC0000,0x00EA0000,
 0x00970000,0x00F20000,0x00CF0000,0x00CE0000,0x00F00000,0x00B40000,0x00E60000,0x00730000,
 0x00960000,0x00AC0000,0x00740000,0x00220000,0x00E70000,0x00AD0000,0x00350000,0x00850000,
 0x00E20000,0x00F90000,0x00370000,0x00E80000,0x001C0000,0x00750000,0x00DF0000,0x006E0000,
 0x00470000,0x00F10000,0x001A0000,0x00710000,0x001D0000,0x00290000,0x00C50000,0x00890000,
 0x006F0000,0x00B70000,0x00620000,0x000E0000,0x00AA0000,0x00180000,0x00BE0000,0x001B0000,
 0x00FC0000,0x00560000,0x003E0000,0x004B0000,0x00C60000,0x00D20000,0x00790000,0x00200000,
 0x009A0000,0x00DB0000,0x00C00000,0x00FE0000,0x00780000,0x00CD0000,0x005A0000,0x00F40000,
 0x001F0000,0x00DD0000,0x00A80000,0x00330000,0x00880000,0x00070000,0x00C70000,0x00310000,
 0x00B10000,0x00120000,0x00100000,0x00590000,0x00270000,0x00800000,0x00EC0000,0x005F0000,
 0x00600000,0x00510000,0x007F0000,0x00A90000,0x00190000,0x00B50000,0x004A0000,0x000D0000,
 0x002D0000,0x00E50000,0x007A0000,0x009F0000,0x00930000,0x00C90000,0x009C0000,0x00EF0000,
 0x00A00000,0x00E00000,0x003B0000,0x004D0000,0x00AE0000,0x002A0000,0x00F50000,0x00B00000,
 0x00C80000,0x00EB0000,0x00BB0000,0x003C0000,0x00830000,0x00530000,0x00990000,0x00610000,
 0x00170000,0x002B0000,0x00040000,0x007E0000,0x00BA0000,0x00770000,0x00D60000,0x00260000,
 0x00E10000,0x00690000,0x00140000,0x00630000,0x00550000,0x00210000,0x000C0000,0x007D0000
 },
 {
 0x52000000,0x09000000,0x6A000000,0xD5000000,0x30000000,0x36000000,0xA5000000,0x38000000,
 0xBF000000,0x40000000,0xA3000000,0x9E000000,0x81000000,0xF3000000,0xD7000000,0xFB000000,
 0x7C000000,0xE3000000,0x39000000,0x82000000,0x9B000000,0x2F000000,0xFF000000,0x87000000,
 0x34000000,0x8E000000,0x43000000,0x44000000,0xC4000000,0xDE000000,0xE9000000,0xCB000000,
 0x54000000,0x7B000000,0x94000000,0x32000000,0xA6000000,0xC2000000,0x23000000,0x3D000000,
 0xEE000000,0x4C000000,0x95000000,0x0B000000,0x42000000,0xFA000000,0xC3000000,0x4E000000,
 0x08000000,0x2E000000,0xA1000000,0x66000000,0x28000000,0xD9000000,0x24000000,0xB2000000,
 0x76000000,0x5B000000,0xA2000000,0x49000000,0x6D000000,0x8B000000,0xD1000000,0x25000000,
 0x72000000,0xF8000000,0xF6000000,0x64000000,0x86000000,0x68000000,0x98000000,0x16000000,
 0xD4000000,0xA4000000,0x5C000000,0xCC000000,0x5D000000,0x65000000,0xB6000000,0x92000000,
 0x6C000000,0x70000000,0x48000000,0x50000000,0xFD000000,0xED000000,0xB9000000,0xDA000000,
 0x5E000000,0x15000000,0x46000000,0x57000000,0xA7000000,0x8D000000,0x9D000000,0x84000000,
 0x90000000,0xD8000000,0xAB000000,0000000000,0x8C000000,0xBC000000,0xD3000000,0x0A000000,
 0xF7000000,0xE4000000,0x58000000,0x05000000,0xB8000000,0xB3000000,0x45000000,0x06000000,
 0xD0000000,0x2C000000,0x1E000000,0x8F000000,0xCA000000,0x3F000000,0x0F000000,0x02000000,
 0xC1000000,0xAF000000,0xBD000000,0x03000000,0x01000000,0x13000000,0x8A000000,0x6B000000,
 0x3A000000,0x91000000,0x11000000,0x41000000,0x4F000000,0x67000000,0xDC000000,0xEA000000,
 0x97000000,0xF2000000,0xCF000000,0xCE000000,0xF0000000,0xB4000000,0xE6000000,0x73000000,
 0x96000000,0xAC000000,0x74000000,0x22000000,0xE7000000,0xAD000000,0x35000000,0x85000000,
 0xE2000000,0xF9000000,0x37000000,0xE8000000,0x1C000000,0x75000000,0xDF000000,0x6E000000,
 0x47000000,0xF1000000,0x1A000000,0x71000000,0x1D000000,0x29000000,0xC5000000,0x89000000,
 0x6F000000,0xB7000000,0x62000000,0x0E000000,0xAA000000,0x18000000,0xBE000000,0x1B000000,
 0xFC000000,0x56000000,0x3E000000,0x4B000000,0xC6000000,0xD2000000,0x79000000,0x20000000,
 0x9A000000,0xDB000000,0xC0000000,0xFE000000,0x78000000,0xCD000000,0x5A000000,0xF4000000,
 0x1F000000,0xDD000000,0xA8000000,0x33000000,0x88000000,0x07000000,0xC7000000,0x31000000,
 0xB1000000,0x12000000,0x10000000,0x59000000,0x27000000,0x80000000,0xEC000000,0x5F000000,
 0x60000000,0x51000000,0x7F000000,0xA9000000,0x19000000,0xB5000000,0x4A000000,0x0D000000,
 0x2D000000,0xE5000000,0x7A000000,0x9F000000,0x93000000,0xC9000000,0x9C000000,0xEF000000,
 0xA0000000,0xE0000000,0x3B000000,0x4D000000,0xAE000000,0x2A000000,0xF5000000,0xB0000000,
 0xC8000000,0xEB000000,0xBB000000,0x3C000000,0x83000000,0x53000000,0x99000000,0x61000000,
 0x17000000,0x2B000000,0x04000000,0x7E000000,0xBA000000,0x77000000,0xD6000000,0x26000000,
 0xE1000000,0x69000000,0x14000000,0x63000000,0x55000000,0x21000000,0x0C000000,0x7D000000
 }
};

/*----------------- The workspace ------------------------------*/

static u32 Ekey[44];   /* The expanded key */

/*------ The round Function.  4 table lookups and 4 Exors ------*/
#define f_rnd(x, n)                         \
  ( ft_tab[0][byte0(x[n])]                  \
  ^ ft_tab[1][byte1(x[(n + 1) & 3])]        \
  ^ ft_tab[2][byte2(x[(n + 2) & 3])]        \
  ^ ft_tab[3][byte3(x[(n + 3) & 3])] )

#define f_round(bo, bi, k)          \
    bo[0] = f_rnd(bi, 0) ^ k[0];    \
    bo[1] = f_rnd(bi, 1) ^ k[1];    \
    bo[2] = f_rnd(bi, 2) ^ k[2];    \
    bo[3] = f_rnd(bi, 3) ^ k[3];    \
    k += 4

/*--- The S Box lookup used in constructing the Key schedule ---*/
#define ls_box(x)       \
 (  fl_tab[0][byte0(x)] \
  ^ fl_tab[1][byte1(x)] \
  ^ fl_tab[2][byte2(x)] \
  ^ fl_tab[3][byte3(x)] )

/*------------ The last round function (no MixColumn) ----------*/
#define lf_rnd(x, n)                      \
  ( fl_tab[0][byte0(x[n])]                \
  ^ fl_tab[1][byte1(x[(n + 1) & 3])]     \
  ^ fl_tab[2][byte2(x[(n + 2) & 3])]     \
  ^ fl_tab[3][byte3(x[(n + 3) & 3])] )


/*-------------------------------------------------------------
 * RijndaelKeySchedule
 *   Initialise the key schedule from a supplied key
 */
void RijndaelKeySchedule(u8 key[16])
{
    u32  t;
    u32  *ek=Ekey,     /* pointer to the expanded key   */
         *rc=rnd_con;   /* pointer to the round constant */

    Ekey[0] = u32_in(key     );
    Ekey[1] = u32_in(key +  4);
    Ekey[2] = u32_in(key +  8);
    Ekey[3] = u32_in(key + 12);

    while(ek < Ekey + 40)
    {
       t = rot3(ek[3]);
        ek[4] = ek[0] ^ ls_box(t) ^ *rc++;
        ek[5] = ek[1] ^ ek[4];
        ek[6] = ek[2] ^ ek[5];
        ek[7] = ek[3] ^ ek[6];
        ek += 4;
    }
}

/*-------------------------------------------------------------
 * RijndaelEncrypt
 *   Encrypt an input block
 */
void RijndaelEncrypt(u8 in[16], u8 out[16])
{
    u32    b0[4], b1[4], *kp = Ekey;

    b0[0] = u32_in(in     ) ^ *kp++;
    b0[1] = u32_in(in +  4) ^ *kp++;
    b0[2] = u32_in(in +  8) ^ *kp++;
    b0[3] = u32_in(in + 12) ^ *kp++;

    f_round(b1, b0, kp);
    f_round(b0, b1, kp);
    f_round(b1, b0, kp);
    f_round(b0, b1, kp);
    f_round(b1, b0, kp);
    f_round(b0, b1, kp);
    f_round(b1, b0, kp);
    f_round(b0, b1, kp);
    f_round(b1, b0, kp);
    u32_out(out,      lf_rnd(b1, 0) ^ kp[0]);
    u32_out(out +  4, lf_rnd(b1, 1) ^ kp[1]);
    u32_out(out +  8, lf_rnd(b1, 2) ^ kp[2]);
    u32_out(out + 12, lf_rnd(b1, 3) ^ kp[3]);
}
```

ETSI/SAGE Specification

Version: 1.0
Date: 22nd November 2000

Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**

Document 2: Implementors' Test Data

Document History		
1.0	**22nd November 2000**	**Initial Release**

PREFACE

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

This document is the second of three, which between them form the entire specification of the example algorithms:

- 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.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 2: Implementors' Test Data.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 3: Algorithm Conformance Test Data.

Blank Page

TABLE OF CONTENTS

1. OUTLINE OF THE IMPLEMENTORS' TEST DATA 7

2. INTRODUCTORY INFORMATION 7
2.1. Introduction 7
2.2. Radix 7
2.3. Bit/Byte ordering 7
2.4. List of Variables... 7
2.5. Algorithm Inputs and Outputs 8
2.6. Coverage 9

3. RIJNDAEL TEST DATA 10
3.1. Overview 10
3.2. Format...10
3.3. Test Set 1 10
3.4. Test Set 2 11
3.5. Test Set 3 12
3.6. Test Set 4 12
3.7. Test Set 5 13
3.8. Test Set 6 14

4. AUTHENTICATION ALGORITHMS *f1* AND *f1** 14
4.1. Overview 14
4.2. Format...14
4.3. Test Set 1 14
4.4. Test Set 2 15
4.5. Test Set 3 15
4.6. Test Set 4 15
4.7. Test Set 5 16
4.8. Test Set 6 16

5. ALGORITHMS *f2*, *f5* AND *f3* 16
5.1. Overview 16
5.2. Format...17
5.3. Test Set 1 17
5.4. Test Set 2 17
5.5. Test Set 3 17
5.6. Test Set 4 18
5.7. Test Set 5 18
5.8. Test Set 6 18

6. ALGORITHMS *f4* AND *f5** 18
6.1. Overview 18
6.2. Format...18
6.3. Test Set 1 19
6.4. Test Set 2 19
6.5. Test Set 3 19
6.6. Test Set 4 20
6.7. Test Set 5 20
6.8. Test Set 6 20

REFERENCES

[1] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (3G TS 33.102 version 3.5.0)

[2] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements; (3G TS 33.105 version 3.4.0)

[3] 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.

[4] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 2: Implementors' Test Data (this document).

[5] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 3: Design Conformance Test Data.

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

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

1. OUTLINE OF THE IMPLEMENTORS' TEST DATA

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

Section 3 provides test data for the Rijndael kernel function.

Section 4 provides test data for the authentication algorithms *f1* and *f1**.

Section 5 provides test data for the algorithms *f2, f5* and *f3*.

Section 6 provides test data for the algorithms *f4* and *f5**.

2. INTRODUCTORY INFORMATION

2.1. Introduction

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

The inputs and outputs of all seven algorithms are defined in section 2.5.

2.2. Radix

Unless stated otherwise, all test data values presented in this document are in hexadecimal.

2.3. Bit/Byte ordering

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

2.4. List of Variables

AK	a 48-bit anonymity key that is the output of either of the functions ***f5*** and ***f5****.
AMF	a 16-bit authentication management field that is an input to the functions ***f1*** and ***f1****.
c1,c2,c3,c4,c5	128-bit constants, which are XORed onto intermediate variables.
CK	a 128-bit confidentiality key that is the output of the function ***f3***.
IK	a 128-bit integrity key that is the output of the function ***f4***.
K	a 128-bit subscriber key that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
MAC-A	a 64-bit network authentication code that is the output of the function ***f1***.

MAC-S	a 64-bit resynchronisation authentication code that is the output of the function ***f1****.
OP	a 128-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
OP_C	a 128-bit value derived from **OP** and **K** and used within the computation of the functions.
r1,r2,r3,r4,r5	integers in the range 0–127 inclusive, which define amounts by which intermediate variables are cyclically rotated.
RAND	a 128-bit random challenge that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
RES	a 64-bit signed response that is the output of the function ***f2***.
SQN	a 48-bit sequence number that is an input to either of the functions ***f1*** and ***f1****. (For ***f1**** this input is more precisely called SQN_{MS}.)

2.5. Algorithm Inputs and Outputs

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

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

Table 1. inputs to ***f1*** and ***f1****

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

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

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

Table 3. ***f1*** output

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

Table 4. ***f1**** output

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

Table 5. ***f2*** output

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

Table 6. ***f3*** output

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

Table 7. ***f4*** output

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

Table 8. ***f5*** output

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

Table 9. ***f5**** output

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

2.6. Coverage

The test data sets for the kernel function Rijndael have been chosen in a way that, provided all data sets are tested:

- Every S-Box entry is being used.
- Each input bit has been in both the ‘0’ and ‘1’ state.

The test data sets for all seven functions are based on the test data sets above. The values for OP, K and RAND have been chosen such that the input values of the first encryption are the test data sets of Rijndael. This way, the following coverage is being reached, provided all test data sets are tested:

- The conditions for Rijndael seen above.
- Each input bit for the functions has been in both the ‘0’ and ‘1’ state.

3. RIJNDAEL TEST DATA

3.1. Overview

The test data sets presented here are for the cryptographic kernel function Rijndael with 128-bit key and data as it is specified in [3].

3.2. Format

Rijndael is composed of 10 rounds that transform the input into the output. An intermediate result is called the State. The State can be pictured as a 4x4 rectangular array of bytes (128 bits in total). The cipher key is similarly pictured as a 4x4 rectangular array. In each of the data intermediate values of the round key array and of the State are given. For the first set the value of the State after each step of the algorithm is given. In the remaining data sets only the value of the State as it is at the end of each round is given.

The internal states will be written as hexadecimal strings, column by column and from top to bottom within each column (the same way as plaintext bytes are fed into the matrix).

Example: The State

C2	37	2E	21
3C	69	51	9E
62	EC	9D	23
CC	29	D8	F7

is represented by the string `c23c62cc 3769ec29 2e519dd8 219e23f7.`

3.3. Test Set 1

```
Key: 465b5ce8 b199b49f aa5f0a2e e238a6bc
Plaintext: ee36f7cf 037d37d3 692f7f03 99e7949a

Round key  0: 465b5ce8 b199b49f aa5f0a2e e238a6bc
Round key  1: 407f3970 f1e68def 5bb987c1 b981217d
Round key  2: 4e82c626 bf644bc9 e4ddcc08 5d5ced75
Round key  3: 00d75b6a bfb310a3 5b6edcab 063231de
Round key  4: 2b104605 94a356a6 cfcd8a0d c9ffbbd3
Round key  5: 2dfa20d8 b959767e 7694fc73 bf6b47a0
Round key  6: 725ac0d0 cb03b6ae bd974add 02fc0d7d
Round key  7: 828d3fa7 498e8909 f419c3d4 f6e5cea9
Round key  8: db06ece5 928865ec 6691a638 90746891
Round key  9: 52436d85 c0cb0869 a65aae51 362ec6c0
Round key 10: 55f7d780 953cdfe9 336671b8 0548b778

add keys(0):      a86dab27 b2e4834c c370752d 7bdf3226

Substitution(1): c23c62cc 3769ec29 2e519dd8 219e23f7
Row shift(1):     c2699df7 375123cc 2e9e6229 213cecd8
mix column(1):    4e5b885c 723c6fa8 ae860fdc 32aead18
add keys(1):      0e24b12c 83dae247 f53f881d 8b2f8c65
```

```
Substitution(2): ab36c871 ec5798a0 e675c4a4 3d15644d
Row shift(2):    ab57c44d ec756471 e615c8a0 3d3698a4
mix column(2):   3d1fb8ef 49dbc2dc 802f83b7 1c46d7ba
add keys(2):     739d7ec9 f6bf8915 64f24fbf 411a3acf

Substitution(3): 8f5ef3dd 4208a759 43898408 83a2808a
Row shift(3):    8f08848a 428980dd 43a2f359 835ea708
mix column(3):   13821109 590dac6e d14bf726 50c59077
add keys(3):     13554a63 e6bebccd 8a252b8d 56f7a1a9

Substitution(4): 7dfcd6fb 8eae65bd 7e3ff15d b16832d3
Row shift(4):    7daef1d3 8e3f32fb 7e68d6bd b1fc655d
mix column(4):   31e14465 8f5dc369 2f727d5d 5ea060eb
add keys(4):     1af10260 1bfe95cf e0bff750 975fdb38

Substitution(5): a2a177d0 afbb2a8a e1086853 88cfb907
Row shift(5):    a2bb6807 af08b9d0 e1cf778a 88a12a53
mix column(5):   e670c020 34bfa5e0 6e77458f 8afc88ae
add keys(5):     cb8ae0f8 8de6d39e 18e3b9fc 3597cf0e

Substitution(6): 1f7ee141 5d8e660b ad1156b0 96888aab
Row shift(6):    1f8e56ab 5d118a41 ad88e10b 967e66b0
mix column(6):   4a49dbb4 42bb80fe 2895e193 6370efc2
add keys(6):     38131b64 89b83650 9502ab4e 618ce2bf

Substitution(7): 077daf43 a76c0553 2a77622f ef649808
Row shift(7):    076c6208 a7779843 2a64af53 ef7d052f
mix column(7):   d071b717 17b93e9b 045bfe13 6835e90c
add keys(7):     52fc88b0 5e37b792 f0423dc7 9ed027a5

Substitution(8): 00b0c4e7 589aa94f 8c2c27c6 0b70cc06
Row shift(8):    009a2706 582ccce7 8c70c44f 0bb0a9c6
mix column(8):   9440deb1 efa8c5dd 1874bea5 b256a393
add keys(8):     4f463254 7d20a031 7ee5189d 2222cb02

Substitution(9): 845a2320 ffb7e0c7 f3d9ad5e 93931f77
Row shift(9):    84b7ad77 ffd91f20 f39323c7 935ae05e
mix column(9):   0b6aeb63 aa57789c b76c742b 6d42f0a8
add keys(9):     592986e6 6a9c70f5 1136da7a 5b6c3668

Substitution(10):cba5448e 02de51e6 820557da 39500545
Row shift(10):   cbde5745 0205058e 825044e6 39a551da
add keys(10):    9e2980c5 9739da67 b136355e 3cede6a2

Ciphertext:   9e2980c5 9739da67 b136355e 3cede6a2
```

3.4. Test Set 2

```
Key: 0396eb31 7b6d1c36 f19c1c84 cd6ffd16
Plaintext: 93cc3640 c5d6a521 d81235bd 0882bf0a

Round key  0: 0396eb31 7b6d1c36 f19c1c84 cd6ffd16
Round key  1: aac2ac8c d1afb0ba 2033ac3e ed5c5128
Round key  2: e21398d9 33bc2863 138f845d fed3d575
Round key  3: 80100562 b3ac2d01 a023a95c 5ef07c29
Round key  4: 0400a03a b7ac8d3b 178f2467 497f584e
Round key  5: c66a8f01 71c6023a 6649265d 2f367e13
Round key  6: e399f214 925ff02e f416d673 db20a860
Round key  7: 145b22ad 8604d283 721204f0 a932ac90
Round key  8: b7ca427e 31ce90fd 43dc940d eaee389d
Round key  9: 84cd1cf9 b5038c04 f6df1809 1c312094
Round key 10: 757a3e65 c079b261 36a6aa68 2a978afc

End of Round 0:  905add71 bebbb917 298e2939 c5ed421c
End of round 1:  7605c840 32e4a13b bf94cea5 2775d315
End of round 2:  fb262fc1 7c78fe50 b567e7ef f4991c6f
End of round 3:  7d736610 e36a13d8 7e5d4d65 5db3231a
End of round 4:  a6677d9e ad85d9ed 0f927ff5 6bfcb6f2
End of round 5:  779f0321 d4145989 eb0bfa22 96e1dff5
End of round 6:  cc129610 1c05a8a2 f23ec385 cec8c0e6
End of round 7:  9f3ad732 18f8d6bb f2c1107c b1fad328
End of round 8:  27e20beb bd1aec49 d9f70961 1c2cb788
End of round 9:  3f992786 b9a0f782 1f4e477a ad2089b8

Ciphertext:   009a9e09 96561525 f611667b bf79e226
```

3.5. Test Set 3

```
Key: fec86ba6 eb707ed0 8905757b 1bb44b8f
Plaintext: 8f7a8f0d 108b7f2d 97a53eac c1d958d9

Round key  0: fec86ba6 eb707ed0 8905757b 1bb44b8f
Round key  1: 727b1809 990b66d9 100e13a2 0bba582d
Round key  2: 8411c022 1d1aa6fb 0d14b559 06aeed74
Round key  3: 6444524d 795ef4b6 744a41ef 72e4ac9b
Round key  4: 05d5460d 7c8bb2bb 08c1f354 7a255fcf
Round key  5: 2a1accd7 56917e6c 5e508d38 2475d2f7
Round key  6: 97afa4e1 c13eda8d 9f6e57b5 bb1b8542
Round key  7: 7838880b b9065286 26680533 9d738071
Round key  8: 77f52b55 cef379d3 e89b7ce0 75e8fc91
Round key  9: f745aac8 39b6d31b d12daffb a4c5536a
Round key 10: 67a8a881 5e1e7b9a 8f33d461 2bf6870b

End of Round 0:  71b2e4ab fbfb01fd 1ea04bd7 da6d1356
End of round 1:  3cb90132 a33ad591 8deb73c9 8e09d283
End of round 2:  b18781c8 bf3e51ff 494e89da 10c3d8ab
End of round 3:  e763dbdf 9143322b 6a2f76ac 423f31b6
End of round 4:  6e736a14 03ec0ad6 db08e567 8610665a
End of round 5:  21cced1d 3925460d 8696fbd7 c41843c2
End of round 6:  1d181787 f09d9b62 79437634 0a71746b
End of round 7:  8e178364 1104c2af f5220eee b3714a51
End of round 8:  3224e59e ea6e1a8d 5476716c a93953a3
End of round 9:  a2b75585 8266b04a 2304d3ea b1d71930

Ciphertext:   5d9bce85 4decaf0d a93d28b7 e35f608c
```

3.6. Test Set 4

```
Key: 9e5944ae a94b8116 5c82fbf9 f32db751
Plaintext: 68c98bbf ab628ec1 adf2a3d9 0c34a751
```

```
Round key  0: 9e5944ae a94b8116 5c82fbf9 f32db751
Round key  1: 47f095a3 eebb14b5 b239ef4c 4114581d
Round key  2: bf9a3120 51212595 e318cad9 a20c92c4
Round key  3: 45d52d1a 14f4088f f7ecc256 55e05092
Round key  4: ac8662e6 b8726a69 4f9ea83f 1a7ef8ad
Round key  5: 4fc7f744 f7b59d2d b82b3512 a255cdbf
Round key  6: 937aff7e 64cf6253 dce45741 7eb19afe
Round key  7: 1bc2448d 7f0d26de a3e9719f dd58eb61
Round key  8: f12bab4c 8e268d92 2dcffc0d f097176c
Round key  9: 62dbfbc0 ecfd7652 c1328a5f 31a59d33
Round key 10: 52853807 be784e55 7f4ac40a 4eef5939

End of Round 0:  f690cf11 02290fd7 f1705820 ff191000
End of round 1:  3e3e036c bba920a8 08a087f6 0cef0044
End of round 2:  a2b7531f 96e51993 40c28eb2 7d0d6d5c
End of round 3:  12272200 bcaa9ea6 b6a0a2d4 b306ec9b
End of round 4:  e4564f18 e4e6cd2e d584d859 ccc5974d
End of round 5:  96e9eccd e14c4c00 fad9d057 8a7010e3
End of round 6:  b23995ae a7a5fcde d841096f d2d345fd
End of round 7:  2aae8b7b d31a1204 fc27054a 82aad44c
End of round 8:  9f0541e3 643ad4fe 768997a8 fec108c3
End of round 9:  f2b6d9e2 b9aac122 01df0181 6bd28059

Ciphertext:  db2944cc e8e683cd 03fff199 31a12135
```

3.7. Test Set 5

```
Key: 4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
Plaintext: a840b1dd 60249aa3 22016b4b 31daf3b8

Round key  0: 4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
Round key  1: 3cb3814f 60154fff 31e9d718 4cebbd9c
Round key  2: d7c95f66 b7dc1099 8635c781 cade7a1d
Round key  3: ce13fb12 79cfeb8b fffa2c0a 35245617
Round key  4: f0a20b84 896de00f 7697cc05 43b39a12
Round key  5: 8d1ac29e 04772291 72e0ee94 31537486
Round key  6: 40888659 44ffa4c8 361f4a5c 074c3eda
Round key  7: 293ad19c 6dc57554 5bda3f08 5c9601d2
Round key  8: 394664d6 54831182 0f592e8a 53cf2f58
Round key  9: a8530e3b fcd01fb9 f3893133 a0461e6b
Round key 10: c42171db 38f16e62 cb785f51 6b3e413a

End of Round 0:  e2f16f6d 3c825413 73fdf3ac 4cd8993c
End of round 1:  c4f1362f 8343731b 423af5a1 576add5f
End of round 2:  e81fc43b 3b66dadd 72bd09b7 3964d3ba
End of round 3:  43195665 ac918275 67d94f0c b4fdcaff
End of round 4:  ce20b983 d6477b7c b7efd855 c846fcbe
End of round 5:  b4c7c29e d5035f3c 93178158 e55176d0
End of round 6:  b097f842 7a443a13 33fe2b1a 5a221a77
End of round 7:  d516d0b5 9aa33f60 549a6a7e 9a1ad15c
End of round 8:  691db74f 07c70966 12662783 77953444
End of round 9:  c7699f17 a4df4ed5 9ec7ce96 4b0f6209

Ciphertext:  02bffada 7137c492 c00e8452 d8c76eaa
```

3.8. Test Set 6

```
Key: 6c38a116 ac280c45 4f59332e e35c8c4f
Plaintext: d66789ef f5996b9c ffd89e0a 77148657

Round key  0: 6c38a116 ac280c45 4f59332e e35c8c4f
Round key  1: 275c2507 8b742942 c42d1a6c 27719623
Round key  2: 86cc03cb 0db82a89 c99530e5 eee4a6c6
Round key  3: ebe8b7e3 e6509d6a 2fc5ad8f c1210b49
Round key  4: 1ec38c9b f89311f1 d756bc7e 1677b737
Round key  5: fb6a16dc 03f9072d d4afbb53 c2d80c64
Round key  6: ba9455f9 b96d52d4 6dc2e987 af1ae5e3
Round key  7: 584d4480 e1201654 8ce2ffd3 23f81a30
Round key  8: 99ef40a6 78cf56f2 f42da921 d7d5b311
Round key  9: 8182c2a8 f94d945a 0d603d7b dab58e6a
Round key 10: 629bc0ff 9bd654a5 96b669de 4c03e7b4

End of Round 0:  ba5f28f9 59b167d9 b081ad24 94480a18
End of round 1:  af2ac41c ec979046 e6079852 9a743063
End of round 2:  4a8f6863 46779622 255afa56 d12d3b90
End of round 3:  15d86e9a 92abb035 f40d6e6e 09e3b591
End of round 4:  14650a94 69b5eb49 88e7961d cf19b897
End of round 5:  68028187 03a5d481 7c4a28c1 574cf516
End of round 6:  49c6b78a df871147 3698dc8a b00fd1c1
End of round 7:  e9b1f7ac 4c0e3069 154b3e58 cd8fd4c8
End of round 8:  1999a956 3dd2f44b 42d34338 9b8570d5
End of round 9:  ef0a9266 fde4bf7d 97a4f536 63bb1809

Ciphertext:  bdf226fe cf9ff996 1e5c2621 b764efb1
```

4. AUTHENTICATION ALGORITHMS *f1* AND *f1**

4.1. Overview

The test data sets presented here are for the authentication algorithms *f1, f1**. No detailed data of the internal states of Rijndael are presented here as these are covered in chapter 3.

4.2. Format

Each test starts by showing the various inputs (K, RAND, SQN, AMF) to the functions. This will be followed by the configuration field OP. Thereafter a table is shown with various intermediate values described in the left column. The value OP_C in the second row should not be computed on but off the USIM. In the example code OP_C is computed inside the functions, so it was included in the table.

4.3. Test Set 1

```
K:    465b5ce8 b199b49f aa5f0a2e e238a6bc
RAND: 23553cbe 9637a89d 218ae64d ae47bf35
SQN:  ff9bb4d0 b607
AMF:  b9b9
OP:   cdc202d5 123e20f6 2b6d676a c72cb318
```

SQN,AMF expanded to 128 bits	ff9bb4d0 b607b9b9 ff9bb4d0 b607b9b9
OP_C	cd63cb71 954a9f4e 48a5994e 37a02baf
Value after 1st encryption	9e2980c5 9739da67 b136355e 3cede6a2
(SQN,AMF) XOR OP_C, rotated	b73e2d9e 81a79216 32f87fa1 234d26f7
Input to 2nd encryption	2917ad5b 169e4871 83ce4aff 1fa0c055
Output of 2nd encryption	87fc31b2 c19530fd 496a36d0 f3485a46
Value of *f1*	4a9ffac3 54dfafb3
Value of *f1**	01cfaf9e c4e871e9

4.4. Test Set 2

```
K:    0396eb31 7b6d1c36 f19c1c84 cd6ffd16
RAND: c00d6031 03dcee52 c4478119 494202e8
SQN:  fd8eef40 df7d
AMF:  af17
OP:   ff53bade 17df5d4e 793073ce 9d7579fa
```

SQN,AMF expanded to 128 bits	fd8eef40 df7daf17 fd8eef40 df7daf17
OP_C	53c15671 c60a4b73 1c55b4a4 41c0bde2
Value after 1st encryption	009a9e09 96561525 f611667b bf79e226
(SQN,AMF) XOR OP_C, rotated	e1db5be4 9ebd12f5 ae4fb931 1977e464
Input to 2nd encryption	e141c5ed 08eb07d0 585edf4a a60e0642
Output of 2nd encryption	0e34e569 c1e813c3 b495a241 5f341ea1
Value of *f1*	5df5b318 07e258b0
Value of *f1**	a8c016e5 1ef4a343

4.5. Test Set 3

```
K:    fec86ba6 eb707ed0 8905757b 1bb44b8f
RAND: 9f7c8d02 1accf4db 213ccff0 c7f71a6a
SQN:  9d027759 5ffc
AMF:  725c
OP:   dbc59adc b6f9a0ef 735477b7 fadf8374
```

SQN,AMF expanded to 128 bits	9d027759 5ffc725c 9d027759 5ffc725c
OP_C	1006020f 0a478bf6 b699f15c 062e42b3
Value after 1st encryption	5d9bce85 4decaf0d a93d28b7 e35f608c
(SQN,AMF) XOR OP_C, rotated	2b9b8605 59d230ef 8d047556 55bbf9aa
Input to 2nd encryption	76004880 143e9fe2 24395de1 b6e49926
Output of 2nd encryption	8cadc1e6 91e8f977 2318bafe b52a0197
Value of *f1*	9cabc3e9 9baf7281
Value of *f1**	95814ba2 b3044324

4.6. Test Set 4

```
K:    9e5944ae a94b8116 5c82fbf9 f32db751
RAND: ce83dbc5 4ac0274a 157c17f8 0d017bd6
SQN:  0b604a81 eca8
AMF:  9e09
OP:   223014c5 806694c0 07ca1eee f57f004f
```

SQN,AMF expanded to 128 bits	`0b604a81 eca89e09 0b604a81 eca89e09`
OP_C	`a64a507a e1a2a98b b88eb421 0135dc87`
Value after 1st encryption	`db2944cc e8e683cd 03fff199 31a12135`
(SQN,AMF) XOR OP_C, rotated	`b3eefea0 ed9d428e ad2a1afb 0d0a3782`
Input to 2nd encryption	`68c7ba6c 057bc143 aed5eb62 3cab16b7`
Output of 2nd encryption	`d2efd25a 2a0ae5c2 14a2736b 97b2c4b0`
Value of *f1*	`74a58220 cba84c49`
Value of *f1**	`ac2cc74a 96871837`

4.7. Test Set 5

```
K:    4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
RAND: 74b0cd60 31a1c833 9b2b6ce2 b8c4a186
SQN:  e880a1b5 80b6
AMF:  9f07
OP:   2d16c5cd 1fdf6b22 383584e3 bef2a8d8
```

SQN,AMF expanded to 128 bits	`e880a1b5 80b69f07 e880a1b5 80b69f07`
OP_C	`dcf07cbd 51855290 b92a07a9 891e523e`
Value after 1st encryption	`02bffada 7137c492 c00e8452 d8c76eaa`
(SQN,AMF) XOR OP_C, rotated	`51aaa61c 09a8cd39 3470dd08 d133cd97`
Input to 2nd encryption	`53155cc6 789f09ab f47e595a 09f4a33d`
Output of 2nd encryption	`9517f960 43e73c62 27af7eaa bfa56d9c`
Value of *f1*	`49e785dd 12626ef2`
Value of *f1**	`9e857903 36bb3fa2`

4.8. Test Set 6

```
K:    6c38a116 ac280c45 4f59332e e35c8c4f
RAND: ee6466bc 96202c5a 557abbef f8babf63
SQN:  414b9822 2181
AMF:  4464
OP:   1ba00a1a 7c6700ac 8c3ff3e9 6ad08725
```

SQN,AMF expanded to 128 bits	`414b9822 21814464 414b9822 21814464`
OP_C	`3803ef53 63b947c6 aaa225e5 8fae3934`
Value after 1st encryption	`bdf226fe cf9ff996 1e5c2621 b764efb1`
(SQN,AMF) XOR OP_C, rotated	`ebe9bdc7 ae2f7d50 79487771 423803a2`
Input to 2nd encryption	`561b9b39 61b084c6 67145150 f55cec13`
Output of 2nd encryption	`3f8930e7 eb9d5d91 2a864e68 8e2885c5`
Value of *f1*	`078adfb4 88241a57`
Value of *f1**	`80246b8d 0186bcf1`

5. ALGORITHMS *f2, f5* AND *f3*

5.1. Overview

The test data sets presented here are for the algorithms *f2, f5* and *f3*. No detailed data of the internal states of Rijndael are presented here as these are covered in chapter 3.

5.2. Format

Each Test starts by showing the inputs K and RAND to the algorithms, followed by the configuration field OP.

Thereafter five rows of data are shown:

Row 1, denoted a, shows the value of OP_C.
Row 2, denoted b, shows the output of the first encryption after XORing the value OP_C.
Row 3, denoted c, shows the input of the second encryption.
Row 4, denoted d, shows the output of the second encryption.
Row 5, denoted e, shows the values of *f2, f5* and *f3.*

The value OP_C in the first row should not be computed on but off the USIM. In the example code OP_C is computed inside the functions, so it was included in the table.

5.3. Test Set 1

```
K:     465b5ce8 b199b49f aa5f0a2e e238a6bc
RAND:  23553cbe 9637a89d 218ae64d ae47bf35
OP:    cdc202d5 123e20f6 2b6d676a c72cb318
```

	f2 and *f5*		*f3*
a	cd63cb71 954a9f4e 48a5994e 37a02baf		
b	534a4bb4 02734529 f993ac10 0b4dcd0d		
c	534a4bb4 02734529 f993ac10 0b4dcd0c		02734529 f993ac10 0b4dcd0d 534a4bb6
d	670b5715 163a3350 ede7889b d41a7b10		796862d2 50c1b54b f35540c9 85bbd364
e	a54211d5 e3ba50bf	aa689c64 8370	b40ba9a3 c58b2a05 bbf0d987 b21bf8cb

5.4. Test Set 2

```
K:     0396eb31 7b6d1c36 f19c1c84 cd6ffd16
RAND:  c00d6031 03dcee52 c4478119 494202e8
OP:    ff53bade 17df5d4e 793073ce 9d7579fa
```

	f2 and *f5*		*f3*
a	53c15671 c60a4b73 1c55b4a4 41c0bde2		
b	535bc878 505c5e56 ea44d2df feb95fc4		
c	535bc878 505c5e56 ea44d2df feb95fc5		505c5e56 ea44d2df feb95fc4 535bc87a
d	97b6d5e8 9978a10d cff39c49 d9469d12		0b05658e bc7ac9df c87196ab 6aa778b4
e	d3a628ed 988620f0	c4778399 5f72	58c433ff 7a7082ac d424220f 2b67c556

5.5. Test Set 3

```
K:     fec86ba6 eb707ed0 8905757b 1bb44b8f
RAND:  9f7c8d02 1accf4db 213ccff0 c7f71a6a
OP:    dbc59adc b6f9a0ef 735477b7 fadf8374
```

	f2 and *f5*		*f3*
a	1006020f 0a478bf6 b699f15c 062e42b3		
b	4d9dcc8a 47ab24fb 1fa4d9eb e571223f		
c	4d9dcc8a 47ab24fb 1fa4d9eb e571223e		47ab24fb 1fa4d9eb e571223f 4d9dcc88
d	234e4fcd 192cdf7e 368835d0 0a0f0c61		4dbbb926 5eaf783b 50fc411a 11b4122b
e	8011c48c 0c214ed2	33484dc2 136b	5dbdbb29 54e8f3cd e665b046 179a5098

5.6. Test Set 4

```
K:     9e5944ae a94b8116 5c82fbf9 f32db751
RAND: ce83dbc5 4ac0274a 157c17f8 0d017bd6
OP:    223014c5 806694c0 07ca1eee f57f004f
```

	f2 and *f5*		*f3*
a	a64a507a e1a2a98b b88eb421 0135dc87		
b	7d6314b6 09442a46 bb7145b8 3094fdb2		
c	7d6314b6 09442a46 bb7145b8 3094fdb3		09442a46 bb7145b8 3094fdb2 7d6314b4
d	56f390f0 318c0249 4beb7949 3decf211		4449bdc9 76b7dd7e 11c5b940 b923e8da
e	f365cd68 3cd92e96	f0b9c08a d02e	e203edb3 971574f5 a94b0d61 b816345d

5.7. Test Set 5

```
K:     4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
RAND: 74b0cd60 31a1c833 9b2b6ce2 b8c4a186
OP:    2d16c5cd 1fdf6b22 383584e3 bef2a8d8
```

	f2 and *f5*		*f3*
a	dcf07cbd 51855290 b92a07a9 891e523e		
b	de4f8667 20b29602 792483fb 51d93c94		
c	de4f8667 20b29602 792483fb 51d93c95		20b29602 792483fb 51d93c94 de4f8665
d	ed1166dd c09d1433 e14afbb2 472b4c40		aaa70ad6 66b84eb1 81d9004a 578c10c7
e	5860fc1b ce351e7e	31e11a60 9118	7657766b 373d1c21 38f307e3 de9242f9

5.8. Test Set 6

```
K:     6c38a116 ac280c45 4f59332e e35c8c4f
RAND: ee6466bc 96202c5a 557abbef f8babf63
OP:    1ba00a1a 7c6700ac 8c3ff3e9 6ad08725
```

	f2 and *f5*		*f3*
a	3803ef53 63b947c6 aaa225e5 8fae3934		
b	85f1c9ad ac26be50 b4fe03c4 38cad685		
c	85f1c9ad ac26be50 b4fe03c4 38cad684		ac26be50 b4fe03c4 38cad685 85f1c9af
d	7db319c9 d3d51ccb bc6a06da 8a0e951c		078f9ad4 9d370ce5 9054534b 519e830f
e	16c8233f 05a0ac28	45b0f69a b06c	3f8c7587 fe8e4b23 3af676ae de30ba3b

6. ALGORITHMS *f4* AND *f5**

6.1. Overview

The test data sets presented here are for the algorithms *f4* and *f5**. No detailed data of the internal states of Rijndael are presented here as these are covered in chapter 3.

6.2. Format

Each Test starts by showing the inputs K and RAND to the algorithms, followed by the configuration field OP.

Thereafter five rows of data are shown:

Row 1, denoted a, shows the value of OP_C.
Row 2, denoted b, shows the output of the first encryption after XORing the value OP_C.
Row 3, denoted c, shows the input of the second encryption.
Row 4, denoted d, shows the output of the second encryption.
Row 5, denoted e, shows the values of *f4* and *f5**.

The value OP_C in the first row should not be computed on but off the USIM. In the example code OP_C is computed inside the functions, so it was included in the table.

6.3. Test Set 1

```
K:    465b5ce8 b199b49f aa5f0a2e e238a6bc
RAND: 23553cbe 9637a89d 218ae64d ae47bf35
OP:   cdc202d5 123e20f6 2b6d676a c72cb318
```

	f4	*f5**
a	cd63cb71 954a9f4e 48a5994e 37a02baf	
b	534a4bb4 02734529 f993ac10 0b4dcd0d	
c	f993ac10 0b4dcd0d 534a4bb4 0273452d	0b4dcd0d 534a4bb4 02734529 f993ac18
d	3a0a77a6 c44ed94a 5ad3eb3f 2bcd1fee	887d409d 3171e7ae b1e55195 635d0a6e
e	f769bcd7 51044604 12767271 1c6d3441	451e8bec a43b

6.4. Test Set 2

```
K:    0396eb31 7b6d1c36 f19c1c84 cd6ffd16
RAND: c00d6031 03dcee52 c4478119 494202e8
OP:   ff53bade 17df5d4e 793073ce 9d7579fa
```

	f4	*f5**
a	53c15671 c60a4b73 1c55b4a4 41c0bde2	
b	535bc878 505c5e56 ea44d2df feb95fc4	
c	ea44d2df feb95fc4 535bc878 505c5e52	feb95fc4 535bc878 505c5e56 ea44d2d7
d	72699788 ef7a61a8 2226302c f8357838	63304f01 a7cb2601 408d9704 046ac887
e	21a8c1f9 29702adb 3e738488 b9f5c5da	30f11970 61c1

6.5. Test Set 3

```
K:    fec86ba6 eb707ed0 8905757b 1bb44b8f
RAND: 9f7c8d02 1accf4db 213ccff0 c7f71a6a
OP:   dbc59adc b6f9a0ef 735477b7 fadf8374
```

	f4	*f5**
a	1006020f 0a478bf6 b699f15c 062e42b3	
b	4d9dcc8a 47ab24fb 1fa4d9eb e571223f	
c	1fa4d9eb e571223f 4d9dcc8a 47ab24ff	e571223f 4d9dcc8a 47ab24fb 1fa4d9e3
d	49af2f34 4d2d8fb5 fee9a493 8e9c72c8	ceaadf8b 86811f35 71465e88 8475bd38
e	59a92d3b 476a0443 487055cf 88b2307b	deacdd84 8cc6

6.6. Test Set 4

```
K:     9e5944ae a94b8116 5c82fbf9 f32db751
RAND: ce83dbc5 4ac0274a 157c17f8 0d017bd6
OP:    223014c5 806694c0 07ca1eee f57f004f
```

	f4	*f5**
a	a64a507a e1a2a98b b88eb421 0135dc87	
b	7d6314b6 09442a46 bb7145b8 3094fdb2	
c	bb7145b8 3094fdb2 7d6314b6 09442a42	3094fdb2 7d6314b6 09442a46 bb7145b0
d	aa0f74d7 0b62e84f 650db901 847a18ec	c6cff816 8ec16553 38d688d7 a64aae30
e	0c4524ad eac041c4 dd830d20 854fc46b	6085a86c 6f63

6.7. Test Set 5

```
K:     4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
RAND: 74b0cd60 31a1c833 9b2b6ce2 b8c4a186
OP:    2d16c5cd 1fdf6b22 383584e3 bef2a8d8
```

	f4	*f5**
a	dcf07cbd 51855290 b92a07a9 891e523e	
b	de4f8667 20b29602 792483fb 51d93c94	
c	792483fb 51d93c94 de4f8667 20b29606	51d93c94 de4f8667 20b29602 792483f3
d	c0b295dd 891edd39 260d4349 f992996d	22d52958 1b2c5255 c399f91e 11dff7d5
e	1c42e960 d89b8fa9 9f2744e0 708ccb53	fe2555e5 4aa9

6.8. Test Set 6

```
K:     6c38a116 ac280c45 4f59332e e35c8c4f
RAND: ee6466bc 96202c5a 557abbef f8babf63
OP:    1ba00a1a 7c6700ac 8c3ff3e9 6ad08725
```

	f4	*f5**
a	3803ef53 63b947c6 aaa225e5 8fae3934	
b	85f1c9ad ac26be50 b4fe03c4 38cad685	
c	b4fe03c4 38cad685 85f1c9ad ac26be54	38cad685 85f1c9ad ac26be50 b4fe03cc
d	9f458392 850be6f5 d7ebf653 e13bee80	27502278 72aa1db7 919ff0c7 b0c849cf
e	a7466cc1 e6b2a133 7d49d3b6 6e95d7b4	1f53cd2b 1113

ETSI/SAGE Specification **Version: 1.0**

Specification **Date: 22nd November 2000**

Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions *f1*, *f1**, *f2*, *f3*, *f4*, *f5* and *f5**

Document 3: Design Conformance Test Data

Document History		
1.0	**22nd November 2000**	**Initial Release**

PREFACE

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

This document is the third of three, which between them form the entire specification of the example algorithms:

- 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.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 2: Implementors' Test Data.

- Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 3: Design Conformance Test Data.

Blank Page

TABLE OF CONTENTS

1. OUTLINE OF THE DESIGN CONFORMANCE TEST DATA 7

2. INTRODUCTORY INFORMATION 7
   2.1. Introduction 7
   2.2. Radix 7
   2.3. Bit/Byte ordering 7
   2.4. List of Variables 7
   2.5. Algorithm Inputs and Outputs 8
   2.6. Coverage 9

3. CONFORMANCE TEST DATA FOR RIJNDAEL 9
   3.1. Overview 9
   3.2. Format 10
   3.3. Test Sets 10

4. CONFORMANCE TEST DATA FOR MILENAGE 11
   4.1. Overview 11
   4.2. Format 11
   4.3. Test Sets 11

REFERENCES

[1] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (3G TS 33.102 version 3.5.0)

[2] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements; (3G TS 33.105 version 3.4.0)

[3] 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.

[4] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 2: Implementors' Test Data.

[5] Specification of the MILENAGE Algorithm Set: an Example Algorithm Set for the 3GPP Authentication and Key Generation Functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****. Document 3: Design Conformance Test Data (this document).

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

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

1. OUTLINE OF THE DESIGN CONFORMANCE TEST DATA

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

Section 3 provides test data for the cryptographic kernel function Rijndael.

Section 4 provides test data for the MILENAGE authentication and key generation algorithms *f1, f1*, f2, f3, f4, f5* and *f5**.

2. INTRODUCTORY INFORMATION

2.1. Introduction

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

The inputs and outputs of all seven algorithms are defined in section 2.5.

This document provides sets of input/output test data for 'black box' testing of physical realizations of all algorithms.

2.2. Radix

Unless stated otherwise, all test data values presented in this document are in hexadecimal.

2.3. Bit/Byte ordering

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

2.4. List of Variables

AK	a 48-bit anonymity key that is the output of either of the functions ***f5*** and ***f5****.
AMF	a 16-bit authentication management field that is an input to the functions ***f1*** and ***f1****.
c1,c2,c3,c4,c5	128-bit constants, which are XORed onto intermediate variables.
CK	a 128-bit confidentiality key that is the output of the function ***f3***.
IK	a 128-bit integrity key that is the output of the function ***f4***.
K	a 128-bit subscriber key that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
MAC-A	a 64-bit network authentication code that is the output of the function ***f1***.

MAC-S	a 64-bit resynchronisation authentication code that is the output of the function ***f1****.
OP	a 128-bit Operator Variant Algorithm Configuration Field that is a component of the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
OP_C	a 128-bit value derived from **OP** and **K** and used within the computation of the functions.
r1,r2,r3,r4,r5	integers in the range 0–127 inclusive, which define amounts by which intermediate variables are cyclically rotated.
RAND	a 128-bit random challenge that is an input to the functions ***f1***, ***f1****, ***f2***, ***f3***, ***f4***, ***f5*** and ***f5****.
RES	a 64-bit signed response that is the output of the function ***f2***.
SQN	a 48-bit sequence number that is an input to either of the functions ***f1*** and ***f1****. (For ***f1**** this input is more precisely called SQN_{MS}.)

2.5. Algorithm Inputs and Outputs

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

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

Table 1. inputs to ***f1*** and ***f1****

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

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

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

Table 3. ***f1*** output

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

Table 4. ***f1**** output

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

Table 5. ***f2*** output

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

Table 6. ***f3*** output

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

Table 7. ***f4*** output

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

Table 8. ***f5*** output

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

Table 9. ***f5**** output

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

2.6. Coverage

For each of the algorithms the test data sets have been selected such that, provided the entire set of tests is run:

- Each input bit of the Rijndael kernel function will have been in both the ‘0’ and ‘1’ states.
- Each input bit of the modes (RAND, K, SQN, AMF) will have been in both the ‘0’ and ‘1’ states.
- Every S-Box entry of the Rijndael kernel function will have been used.

3. CONFORMANCE TEST DATA FOR RIJNDAEL

3.1. Overview

The test data sets presented here are for the cryptographic kernel function Rijndael. The first 6 test sets are the same as in document 2: implementors’ test data.

3.2. Format

The first test set is shown twice, once in binary format, once in hexadecimal format. This is to explicitly show the relationship between the binary data and the hexadecimal representation. The remainder of the test sets are presented in hexadecimal format only. A hexadecimal number will be broken up in two rows of two 32-bit words. For example, the number `ee36f7cf037d37d3692f7f0399e7949a` will be written as

```
ee36f7cf 037d37d3
692f7f03 99e7949a.
```

3.3. Test Sets

3.3.1. Set 1 Binary Format

```
Plaintext: 11101110 00110110 11110111 11001111 00000011 01111101
           00110111 11010011 01101001 00101111 01111111 00000011
           10011001 11100111 10010100 10011010

Ciphertext: 10011110 00101001 10000000 11000101 10010111 00111001
            11011010 01100111 10110001 00110110 00110101 01011110
            00111100 11101101 11100110 10100010

Key: 01000110 01011011 01011100 11101000 10110001 10011001 10110100
     10011111 10101010 01011111 00001010 00101110 11100010 00111000
     10100110 10111100
```

3.3.2. Hexadecimal Format

Set	Plaintext	Ciphertext	Key
1	ee36f7cf 037d37d3 692f7f03 99e7949a	9e2980c5 9739da67 b136355e 3cede6a2	465b5ce8 b199b49f aa5f0a2e e238a6bc
2	93cc3640 c5d6a521 d81235bd 0882bf0a	009a9e09 96561525 f611667b bf79e226	0396eb31 7b6d1c36 f19c1c84 cd6ffd16
3	8f7a8f0d 108b7f2d 97a53eac c1d958d9	5d9bce85 4decaf0d a93d28b7 e35f608c	fec86ba6 eb707ed0 8905757b 1bb44b8f
4	68c98bbf ab628ec1 adf2a3d9 0c34a751	db2944cc e8e683cd 03fff199 31a12135	9e5944ae a94b8116 5c82fbf9 f32db751
5	a840b1dd 60249aa3 22016b4b 31daf3b8	02bffada 7137c492 c00e8452 d8c76eaa	4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
6	d66789ef f5996b9c ffd89e0a 77148657	bdf226fe cf9ff996 1e5c2621 b764efb1	6c38a116 ac280c45 4f59332e e35c8c4f
7	8cbdb88c 620ffe88 f8ce0042 a5052568	fe8d6888 b5a5c146 efea6660 f7b4e699	523ace48 994925ca 0495efd5 0c7c71e2
8	83a4dbcc f3ff12e2 154dbf45 512b5a32	1c3aff64 cd717a6e 959c95c8 b9cd7d3f	e4aee4fa eeb2f93d 43604f5f 2e65ff25
9	d117610c 04f5d3d6 1d8cb6bc 910a918b	21bc0073 cc9aa7a3 81257774 43f8663f	b2c211ec 004d0323 022e49cc d363c8ff
10	9d7cf334 be57ea4d 37f28c8f 8f0a7259	1591d87d 8fa69176 eae9fae9 02cd61a4	443c5a67 59e04dc2 c9c6e465 823dd5b6
11	0f3c382c 488efe2d 8ce7eaa7 7093a486	c59e0669 76d92f8e 567b806f 94fb09d2	23e5c9b1 8034df64 21f22972 c3bd2d93
12	51c0d9dc cb03b860 43d378ea 5f0dd0f8	ff00e5a1 a02f7594 3b4a5a1e 39bf5dc3	d6856512 fd11bf4f 91781e3b 1ea9d8de
13	886f7d2c 6f1dc0cc 851fdd41 82e09d99	3f65bc17 b47b645b 0fbfbfe8 da4269bf	e5e436d4 593ef7cb a221cd07 1ccadfdf

14	dfb0cd05 5b43bd5b 7a31c294 0174b44f	32976e2e d7fb97b2 bcaaba00 80faf1e5	d9c9453e 1b858f22 dab53e20 44e849de
15	7ced704d b8df58f4 4972cecc 0531bc04	53772447 4cf17c9b 105673b5 f37581c8	38fe9997 cbac8d9a c4fc78cc 1fcb1e22
16	878d0fee ad58387e 1b4aaacf b6805e9f	e555cfb7 9663f0ea fdcc9665 55d6498d	8d493a28 f17cfb7d e5be8354 333a66f4
17	29f852ce 631605e3 1722fd87 3bc74208	d78106ad 6fdae41c 95dd2f7b 5b479a79	5d733139 16bde011 c7c26700 0bfe2a9c
18	8995e0d7 0281690a 666070db bd6f0f27	70b059ad f647d183 a18512a3 72ae683c	32448511 c137459f f1d12c17 2cdf7262
19	d1ce62ef b1010adb 2f6eabb2 63e16ee4	ec7fc8b1 10246889 cff7293f e30c8465	93833efa 7bb1316e ebb3dbe3 20a5448d
20	260a7f00 98fe903e 9bb255f6 56840627	91747de6 7effaff8 578381ef 27e497ec	09fa653a cbf5acc8 3e307caa 6e18aa67

4. CONFORMANCE TEST DATA FOR MILENAGE

4.1. Overview

The test data sets presented here are for the seven functions *f1, f1*, f2, f3, f4, f5* and *f5**. The first 6 test sets are the same as in document 2: implementors' test data.

4.2. Format

Each Test shows the various inputs to the algorithm. This is followed by the configuration field OP, the value $OP_C = OP \oplus E[OP]_K$ and finally by the function outputs.

The first test set is shown twice, once in binary format, once in hexadecimal format. This is to explicitly show the relationship between the binary data and the hexadecimal representation. The remainder of the test sets are presented in hexadecimal format only.

4.3. Test Sets

4.3.1. Set 1

Binary Format

```
K:      01000110 01011011 01011100 11101000 10110001 10011001 10110100
        10011111 10101010 01011111 00001010 00101110 11100010 00111000
        10100110 10111100

RAND:   00100011 01010101 00111100 10111110 10010110 00110111 10101000
        10011101 00100001 10001010 11100110 01001101 10101110 01000111
        10111111 00110101

SQN:    11111111 10011011 10110100 11010000 10110110 00000111

AMF:    10111001 10111001

OP:     11001101 11000010 00000010 11010101 00010010 00111110 00100000
        11110110 00101011 01101101 01100111 01101010 11000111 00101100
        10110011 00011000
```

OP_C:	`11001101 01100011 11001011 01110001 10010101 01001010 10011111 01001110 01001000 10100101 10011001 01001110 00110111 10100000 00101011 10101111`
f1:	`01001010 10011111 11111010 11000011 01010100 11011111 10101111 10110011`
*f1**:	`00000001 11001111 10101111 10011110 11000100 11101000 01110001 11101001`
f2:	`10100101 01000010 00010001 11010101 11100011 10111010 01010000 10111111`
f5:	`10101010 01101000 10011100 01100100 10000011 01110000`
f3:	`10110100 00001011 10101001 10100011 11000101 10001011 00101010 00000101 10111011 11110000 11011001 10000111 10110010 00011011 11111000 11001011`
f4:	`11110111 01101001 10111100 11010111 01010001 00000100 01000110 00000100 00010010 01110110 01110010 01110001 00011100 01101101 00110100 01000001`
*f5**:	`01000101 00011110 10001011 11101100 10100100 00111011`

Hexadecimal Format

Variable	Value
K	`465b5ce8 b199b49f aa5f0a2e e238a6bc`
RAND	`23553cbe 9637a89d 218ae64d ae47bf35`
SQN	`ff9bb4d0 b607`
AMF	`b9b9`
OP	`cdc202d5 123e20f6 2b6d676a c72cb318`
OP_C	`cd63cb71 954a9f4e 48a5994e 37a02baf`
f1	`4a9ffac3 54dfafb3`
f1*	`01cfaf9e c4e871e9`
f2	`a54211d5 e3ba50bf`
f5	`aa689c64 8370`
f3	`b40ba9a3 c58b2a05 bbf0d987 b21bf8cb`
f4	`f769bcd7 51044604 12767271 1c6d3441`
f5*	`451e8bec a43b`

4.3.2. Test Set 2

Variable	Value
K	465b5ce8 b199b49f aa5f0a2e e238a6bc
RAND	23553cbe 9637a89d 218ae64d ae47bf35
SQN	ff9bb4d0 b607
AMF	b9b9
OP	cdc202d5 123e20f6 2b6d676a c72cb318
OP_C	cd63cb71 954a9f4e 48a5994e 37a02baf
f1	4a9ffac3 54dfafb3
f1*	01cfaf9e c4e871e9
f2	a54211d5 e3ba50bf
f5	aa689c64 8370
f3	b40ba9a3 c58b2a05 bbf0d987 b21bf8cb
f4	f769bcd7 51044604 12767271 1c6d3441
f5*	451e8bec a43b

4.3.3. Test Set 3

Variable	Value
K	fec86ba6 eb707ed0 8905757b 1bb44b8f
RAND	9f7c8d02 1accf4db 213ccff0 c7f71a6a
SQN	9d027759 5ffc
AMF	725c
OP	dbc59adc b6f9a0ef 735477b7 fadf8374
OP_C	1006020f 0a478bf6 b699f15c 062e42b3
f1	9cabc3e9 9baf7281
f1*	95814ba2 b3044324
f2	8011c48c 0c214ed2
f5	33484dc2 136b
f3	5dbdbb29 54e8f3cd e665b046 179a5098
f4	59a92d3b 476a0443 487055cf 88b2307b
f5*	deacdd84 8cc6

4.3.4. Test Set 4

Variable	Value
K	9e5944ae a94b8116 5c82fbf9 f32db751
RAND	ce83dbc5 4ac0274a 157c17f8 0d017bd6
SQN	0b604a81 eca8
AMF	9e09
OP	223014c5 806694c0 07ca1eee f57f004f
OP_C	a64a507a e1a2a98b b88eb421 0135dc87
f1	74a58220 cba84c49
f1*	ac2cc74a 96871837
f2	f365cd68 3cd92e96
f5	f0b9c08a d02e
f3	e203edb3 971574f5 a94b0d61 b816345d
f4	0c4524ad eac041c4 dd830d20 854fc46b
f5*	6085a86c 6f63

4.3.5. Test Set 5

Variable	Value
K	4ab1deb0 5ca6ceb0 51fc98e7 7d026a84
RAND	74b0cd60 31a1c833 9b2b6ce2 b8c4a186
SQN	e880a1b5 80b6
AMF	9f07
OP	2d16c5cd 1fdf6b22 383584e3 bef2a8d8
OP_C	dcf07cbd 51855290 b92a07a9 891e523e
f1	49e785dd 12626ef2
f1*	9e857903 36bb3fa2
f2	5860fc1b ce351e7e
f5	31e11a60 9118
f3	7657766b 373d1c21 38f307e3 de9242f9
f4	1c42e960 d89b8fa9 9f2744e0 708ccb53
f5*	fe2555e5 4aa9

4.3.6. Test Set 6

Variable	Value
K	6c38a116 ac280c45 4f59332e e35c8c4f
RAND	ee6466bc 96202c5a 557abbef f8babf63
SQN	414b9822 2181
AMF	4464
OP	1ba00a1a 7c6700ac 8c3ff3e9 6ad08725
OP_C	3803ef53 63b947c6 aaa225e5 8fae3934
f1	078adfb4 88241a57
f1*	80246b8d 0186bcf1
f2	16c8233f 05a0ac28
f5	45b0f69a b06c
f3	3f8c7587 fe8e4b23 3af676ae de30ba3b
f4	a7466cc1 e6b2a133 7d49d3b6 6e95d7b4
f5*	1f53cd2b 1113

4.3.7. Test Set 7

Variable	Value
K	2d609d4d b0ac5bf0 d2c0de26 7014de0d
RAND	194aa756 013896b7 4b4a2a3b 0af4539e
SQN	6bf69438 c2e4
AMF	5f67
OP	460a4838 5427aa39 264aac8e fc9e73e8
OP_C	c35a0ab0 bcbfc925 2caff15f 24efbde0
f1	bd07d300 3b9e5cc3
f1*	bcb6c2fc ad152250
f2	8c25a16c d918a1df
f5	7e6455f3 4cf3
f3	4cd08460 20f8fa07 31dd47cb dc6be411
f4	88ab80a4 15f15c73 711254a1 d388f696
f5*	dc6dd01e 8f15

4.3.8. Test Set 8

Variable	Value
K	a530a7fe 428fad10 82c45edd fce13884
RAND	3a4c2b32 45c50eb5 c71d0863 9395764d
SQN	f63f5d76 8784
AMF	b90e
OP	511c6c4e 83e38c89 b1c5d8dd e62426fa
OP_C	27953e49 bc8af6dc c6e730eb 80286be3
f1	53761fbd 679b0bad
f1*	21adfd33 4a10e7ce
f2	a63241e1 ffc3e5ab
f5	88196c47 986f
f3	10f05bab 75a99a5f bb98a9c2 87679c3b
f4	f9ec0865 eb32f223 69cade40 c59c3a44
f5*	c987a3d2 3115

4.3.9. Test Set 9

Variable	Value
K	d9151cf0 4896e258 30bf2e08 267b8360
RAND	f761e5e9 3d603feb 730e2755 6cb8a2ca
SQN	47ee0199 820a
AMF	9113
OP	75fc2233 a44294ee 8e6de25c 4353d26b
OP_C	c4c93eff e8a08138 c203d4c2 7ce4e3d9
f1	66cc4be4 4862af1f
f1*	7a4b8d7a 8753f246
f2	4a90b217 1ac83a76
f5	82a0f528 7a71
f3	71236b71 29f9b22a b77ea7a5 4c96da22
f4	90527eba a5588968 db417273 25a04d9e
f5*	527dbf41 f35f

4.3.10. Test Set 10

Variable	Value
K	a0e2971b 6822e8d3 54a18cc2 35624ecb
RAND	08eff828 b13fdb56 2722c65c 7f30a9b2
SQN	db5c0664 81e0
AMF	716b
OP	323792fa ca21fb4d 5d6f13c1 45a9d2c1
OP_C	82a26f22 bba9e948 8f949a10 d98e9cc4
f1	9485fe24 621cb9f6
f1*	bce325ce 03e2e9b9
f2	4bc2212d 8624910a
f5	a2f858aa 9e5d
f3	08cef6d0 04ec6147 1a3c3cda 048137fa
f4	ed0318ca 5deb9206 272f6e8f a64ba411
f5*	74e76fbb ec38

4.3.11. Test Set 11

Variable	Value
K	0da6f7ba 86d5eac8 a19cf563 ac58642d
RAND	679ac4db acd7d233 ff9d6806 f4149ce3
SQN	6e2331d6 92ad
AMF	224a
OP	4b9a26fa 459e3acb ff36f401 5de3bdc1
OP_C	0db1071f 8767562c a43a0a64 c41e8d08
f1	2831d7ae 9088e492
f1*	9b2e1695 1135d523
f2	6fc30fee 6d123523
f5	4c539a26 e1fa
f3	69b1cae7 c7429d97 5e245cac b05a517c
f4	74f24e8c 26df58e1 b38d7dcd 4f1b7fbd
f5*	07861e12 6928

4.3.12. Test Set 12

Variable	Value
K	77b45843 c88e58c1 0d202684 515ed430
RAND	4c47eb30 76dc55fe 5106cb20 34b8cd78
SQN	fe1a8731 005d
AMF	ad25
OP	bf3286c7 a51409ce 95724d50 3bfe6e70
OP_C	d483afae 562409a3 26b5bb0b 20c4d762
f1	08332d7e 9f484570
f1*	ed41b734 489d5207
f2	aefa357b eac2a87a
f5	30ff25cd adf6
f3	908c43f0 569cb8f7 4bc971e7 06c36c5f
f4	c251df0d 888dd932 9bcf4665 5b226e40
f5*	e84ed0d4 677e

4.3.13. Test Set 13

Variable	Value
K	729b1772 9270dd87 ccdf1bfe 29b4e9bb
RAND	311c4c92 9744d675 b720f3b7 e9b1cbd0
SQN	c85c4cf6 5916
AMF	5bb2
OP	d04c9c35 bd2262fa 810d2924 d036fd13
OP_C	228c2f2f 06ac3268 a9e616ee 16db4ba1
f1	ff794fe2 f827ebf8
f1*	24fe4dc6 1e874b52
f2	98dbbd09 9b3b408d
f5	5380d158 cfe3
f3	44c0f23c 5493cfd2 41e48f19 7e1d1012
f4	0c9fb816 13884c25 35dd0eab f3b440d8
f5*	87ac3b55 9fb6

4.3.14. Test Set 14

Variable	Value
K	d32dd23e 89dc6623 54ca12eb 79dd32fa
RAND	cf7d0ab1 d9430695 0bf12018 fbd46887
SQN	484107e5 6a43
AMF	b5e6
OP	fe75905b 9da47d35 6236d031 4e09c32e
OP_C	d22a4b41 80a53257 08a5ff70 d9f67ec7
f1	cf19d62b 6a809866
f1*	5d269537 e45e2ce6
f2	af4a411e 1139f2c2
f5	217af492 72ad
f3	5af86b80 edb70df5 292cc112 1cbad50c
f4	7f4d6ae7 440e1878 9a8b75ad 3f42f03a
f5*	900e101c 677e

4.3.15. Test Set 15

Variable	Value
K	af7c65e1 927221de 591187a2 c5987a53
RAND	1f0f8578 464fd59b 64bed2d0 9436b57a
SQN	3d627b01 418d
AMF	84f6
OP	0c7acb8d 95b7d4a3 1c5aca6d 26345a88
OP_C	a4cf5c81 55c08a7e ff418e54 43b98e55
f1	c37cae78 05642032
f1*	68cd09a4 52d8db7c
f2	7bffa5c2 f41fbc05
f5	837fd7b7 4419
f3	3f8c3f3c cf7625bf 77fc94bc fd22fd26
f4	abcbae8f d46115e9 961a55d0 da5f2078
f5*	56e97a60 90b1

4.3.16. Test Set 16

Variable	Value
K	5bd7ecd3 d3127a41 d12539be d4e7cf71
RAND	59b75f14 251c7503 1d0bcbac 1c2c04c7
SQN	a298ae89 29dc
AMF	d056
OP	f967f760 38b920a9 cd25e10c 08b49924
OP_C	76089d3c 0ff3efdc 6e36721d 4fceb747
f1	c3f25cd9 4309107e
f1*	b0c8ba34 3665afcc
f2	7e3f44c7 591f6f45
f5	5be11495 525d
f3	d42b2d61 5e49a03a c275a5ae f97af892
f4	0b3f8d02 4fe6bfaf aa982b8f 82e319c2
f5*	4d6a34a1 e4eb

4.3.17. Test Set 17

Variable	Value
K	6cd1c6ce b1e01e14 f1b82316 a90b7f3d
RAND	f69b78f3 00a0568b ce9f0cb9 3c4be4c9
SQN	b4fce5fe b059
AMF	e4bb
OP	078bfca9 564659ec d8851e84 e6c59b48
OP_C	a219dc37 f1dc7d66 738b5843 c799f206
f1	69a90869 c268cb7b
f1*	2e0fdcf9 fd1cfa6a
f2	70f6bdb9 ad21525f
f5	1c408a85 8b3e
f3	6edaf99e 5bd9f85d 5f36d91c 1272fb4b
f4	d61c853c 280dd9c4 6f297bae c386de17
f5*	aa4ae52d aa30

4.3.18. Test Set 18

Variable	Value
K	b73a90cb cf3afb62 2dba83c5 8a8415df
RAND	b120f1c1 a0102a2f 507dd543 de68281f
SQN	f1e8a523 a36d
AMF	471b
OP	b672047e 003bb952 dca6cb8a f0e5b779
OP_C	df0c6786 8fa25f74 8b7044c6 e7c245b8
f1	ebd70341 bcd415b0
f1*	12359f5d 82220c14
f2	479dd25c 20792d63
f5	aefdaa5d dd99
f3	66195dbe d0313274 c5ca7766 615fa25e
f4	66bec707 eb2afc47 6d7408a8 f2927b36
f5*	12ec2b87 fbb1

4.3.19. Test Set 19

Variable	Value
K	51222502 14c33e72 3a5dd523 fc145fc0
RAND	81e92b6c 0ee0e12e bceba8d9 2a99dfa5
SQN	16f3b3f7 0fc2
AMF	c3ab
OP	c9e87632 86b5b9ff bdf56e12 97d0887b
OP_C	981d464c 7c52eb6e 50362349 84ad0bcf
f1	2a5c23d1 5ee351d5
f1*	62dae385 3f3af9d2
f2	28d7b0f2 a2ec3de5
f5	ada15aeb 7bb8
f3	5349fbe0 98649f94 8f5d2e97 3a81c00f
f4	9744871a d32bf9bb d1dd5ce5 4e3e2e5a
f5*	d461bc15 475d

4.3.20. Test Set 20

Variable	Value
K	`90dca4ed a45b53cf 0f12d7c9 c3bc6a89`
RAND	`9fddc720 92c6ad03 6b6e4647 89315b78`
SQN	`20f813bd 4141`
AMF	`61df`
OP	`3ffcfe5b 7b111158 9920d352 8e84e655`
OP_C	`cb9cccc4 b9258e6d ca476037 9fb82581`
f1	`09db94ea b4f8149e`
f1*	`a29468aa 9775b527`
f2	`a95100e2 760952cd`
f5	`83cfd54d b913`
f3	`b5f2da03 883b69f9 6bf52e02 9ed9ac45`
f4	`b4721368 bc16ea67 875c5598 688bb0ef`
f5*	`4f203939 2ddc`