Chapter 5
Cryptographic Algorithms For 3G Mobile Systems

5.1 Introduction: Voice and data need to be protected when transmitted over the radio link between the mobile terminal and the network and so an encryption algorithm is required to protect the confidentiality of the traffic and an integrity algorithm is required to protect the traffic against malicious modification. One of the goals of the 3G had been to fix problems in 2G.The benefits of making cryptographic algorithms public were evident. In 1999, Security Association(SA) had the task of defining and agreeing on the design process for encryption and integrity algorithms. The AES algorithm was too large to fit within 10000 gates of hardware that had been specified as the maximum size for handset implementation. In [66], the following three specification strategies were identified :

1. Select an off-the-shelf algorithm
2. invite submissions
3. commission a special group to design an algorithm

The principles defined in [66] determined to a large extent the design and evaluation strategies that were adopted for the design of the standard 3GPP confidentiality and integrity algorithms for UMTS.

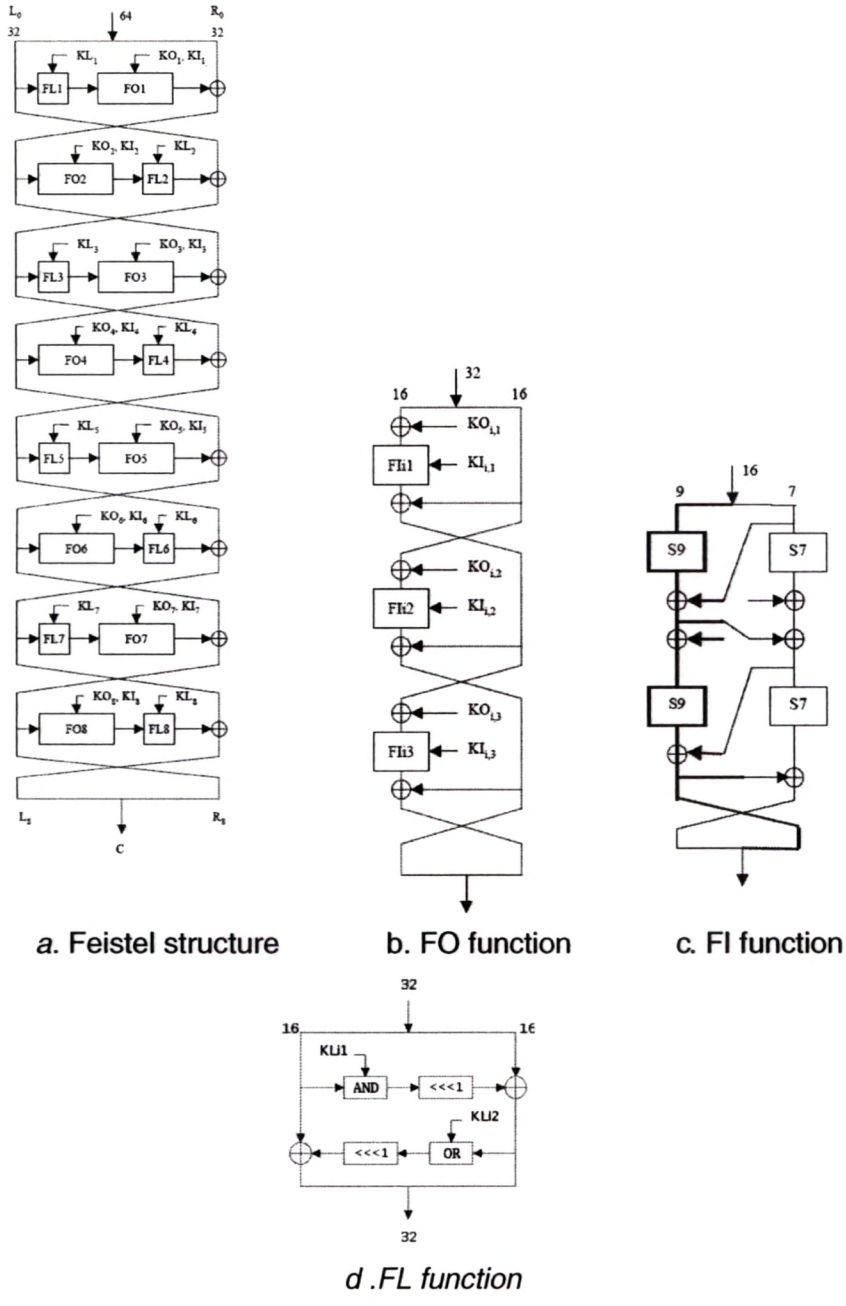
5.2 The Block Cipher KASUMI

The efforts towards the standardization of a block ciphering algorithm to be used in UMTS networks produced the specification of the KASUMI algorithm by the Third Generation Partnership Program (3GPP) [16]. KASUMI has a Feistel structure and operates on 64-bit data blocks under control of a 128-bit encryption key *K* [70]. KASUMI has the following features, as a consequence of its Feistel structure:

- Input plaintext is the input to the first round.
- Ciphertext is the last round's output.
- *K* is used to generate a set of round keys *{KLi, KOi, KIi}* for each round *i*.
- Each round computes a different function, as long as the round keys are different.
- The same algorithm is used both for encryption and decryption.

KASUMI is based on a previous block cipher called MISTY1 [31]. MISTY1 was chosen as the foundation for the 3GPP ciphering algorithm due to its proven security against the most advanced methods to break block ciphers, namely cryptanalysis techniques. In addition, MISTY1 was heavily optimized for hardware implementation.

As can be seen in Fig. 5.1, the KASUMI block cipher has a different setup for the functions within rounds. For odd rounds the round-function is computed by applying the FL function followed by the FO function. For even rounds FO is applied before FL. FL, shown in Fig. 5.1d is a 32-bit function made up of simple AND, OR, XOR and left rotation operations. FO, depicted in Fig. 5.1*b*, is also a 32-bit function having a three-round Feistel organization which contains one FI block per round. FI, seen in Fig. 5.1*c*, is a non-linear 16-bit function having itself a four-round Feistel structure; it is made up of two nine-bit substitution boxes (S-boxes) and two seven-bit S-boxes. Fig .5.1*c* shows that data in the FI function flow along two different paths: a nine-bit long path (thick lines) and a seven-bit path (thin lines). Notice that in Feistel structures, such as the ones used in this algorithm, each round's output is twisted before being applied as input to the following round. After completing eight rounds, KASUMI produces a 64-bit long ciphertext block corresponding to the input plaintext block.



a. Feistel structure b. FO function c. FI function

d .FL function

Fig. 5. 1 The KASUMI block cipher

5.3 Confidentiality and Integrity Algorithms: Confidentiality and integrity algorithms were constructed together as special, new block cipher modes. The block cipher itself is a new design. The requirements for UMTS confidentiality and integrity algorithms are specified by 3GPP in the technical specification document 33.105. The mechanism for data confidentiality of user data and signalling data in UMTS requires a cryptographic function f8.Function f8 must be a synchronous stream cipher algorithm and for interoperability within UMTS it must be fully standardized. It will only be used to protect the confidentiality of user data and signalling data sent over the radio access link between the user equipment (UE) and the Radio Network Controller (RNC) and therefore is implemented in the UE and the RNC. Encryption will be applied in the Medium Access Control (MAC) sublayer and in the Radio Link Control (RLC) sublayer of the data link layer.

In the UE, the algorithm may be implemented as hardware , while in the RNC it may also be implemented in software on a general purpose processor. Therefore, the algorithm should be designed to accommodate a range of implementation options. For hardware implementations, it should be possible to implement one instance of algorithm using less than 10000 gates.

A wide range of UE with different bearer capabilities is expected so encryption throughput requirements on the algorithm will vary depending on the implementation. It must be possible to implement the algorithm to achieve an encryption speed of 2 Mbps at downlink and uplink. Encryption throughput requirements should be met based on clock speeds of 20 MHz.

The algorithm is used in three modes and nominates a different type of data for each mode .The requirements for keystream generator in each mode are as follows:

1. In RLC –transparent mode a new keystream block of 10 ms is required for each new, physical layer frame. The length of a frame varies from 1 bit to 20000 bits with a granularity of 1 bit.
2. In UM(Unacknowledged Mode) RLC mode a new keystream block is required for each new PDU(Protocol Data Unit).The length of PDU varies from 16 bits to 5000 bits ,with granularity of 8 bits.

3. In AM(Acknowledged Mode) RLC mode a new keystream block is required for each new PDU. The length of PDU varies from 24 bits to 5000 bits with granularity of 8 bits.

5.4 Confidentiality Algorithm operation:

The f8 function is a synchronous stream cipher whose encryption and decryption operations are based on the same secret key and same set of values as the initialization parameters. Encryption and decryption operations for a synchronous stream cipher are the same. For both operations a keystream block is generated. The values of keystream bit depend on the given key and initialization parameters. The keystream block and the received data block are added together using a bitwise XOR operation. The length of data block determines the length of keystream block.

Figure 5.2 illustrates the use of f8 to encrypt plaintext by applying a keystream using a bitwise XOR operation. The plaintext may be recovered by generating the same keystream using the same input parameters and applying it to ciphertext using a bitwise XOR operation. The input parameters to f8 are the cipher key (CK),the time dependent input(Count-C),the bearer identity(BEARER),the direction of transmission(DIRECTION) and the length (LENGTH) of the plaintext block. The CK is renewed at every authentication process .COUNT-C,BEARER and DIRECTION can be considered as initialization parameters as they are renewed for each keystream block.The time dependent input COUNT-C is also sent in cleartext and used as a synchronization parameter for the synchronous stream cipher. The input parameter LENGTH only affects the length of keystream ,not the actual bits in it .Based on these input parameters, the algorithm generates the output keystrem block(KEYSTREAM) which is used to encrypt the input plaintext block (PLAINTEXT) and to produce the output ciphertext block.(CIPHERTEXT).

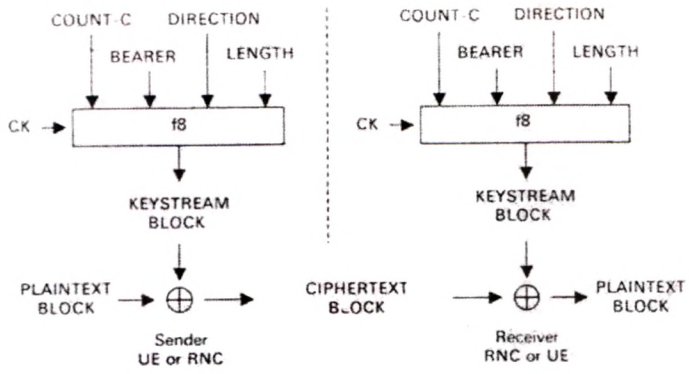


Figure 5.2 Ciphering user and signalling data transmitted over the radio access link

5.5 Interfaces to the confidentiality algorithm:

5.5.1 Cipher Key (CK): The length of CK is 128 bits. In case the length k of the generated key is smaller than 128 bits, the most significant bits of CK carry the nominal key information, whereas the remaining bits repeat the key information as follows:

CK=CK[0],CK[1],.....,CK[127]where Ck[0],..CK[k-1] carry the key information and CK[n]=CK[n mod k] for all n such that k≤=n<128

5.5.2 Time Dependent Input : (Count-C) : The length of the Count-C parameter is 32 bits. Synchronization of the keystream is based on the use of a physical layer frame counter combined with a hyperframe counter introduced to avoid reuse of keysream. The counter is initialized at connection establishment.

5.5.3 Radio Bearer Identity(BEARER) :The length of BEARER is 5 bits.The same cipher key may be simultaneously used for different radio bearers associated with a single user. To avoid using the same keystream to encrypt more than one bearer,the algorithm generates keystream based on identity of the radio bearer.

5.5.4 Transmission Direction : (DIRECTION) The value of the DIRECTION Bit is 0 for uplink and and 1 for downlink. The same cipher key may be used for uplink and downlink channels.The purpose of the DIRECTION bit is to avoid using the same keystream to encrypt both uplink and downlink transmissions.

5.5.5 Keystream length(LENGTH):LENGTH is an integer between 1 and 20000.The length of LENGTH is 16 bits.For a given bearer and transmission direction the length of

the plaintext block that is transmitted during a single physical layer frame may vary.The algorithm will generate a keystream block of variable length based on the value of the LENGTH parameter.The range of values of the LENGTH parameter will depend not only on the RLC PDU/MAC SDU size but also the number of RLC PDUs/MAC SDUs that may be sent in a single ,physical layer,10-ms frame for a given bearer and transmission direction.

5.5.6 KEYSTREAM,PLAINTEXT and CIPHERTEXT: The length of these blocks equals the value of input parameter LENGTH.

5. 6 Description of f8 : The f8 algorithm makes use of the KASUMI key dependent function, which operates on 64 bit data blocks and produces 64 bit blocks under the control of 128 bit key K. The algorithm makes use of two 64 bit registers :the static register A and the counter BLKCNT .Register A is initialized using the 64 bit initialization value :

IV=COUNT | | BEARER|| DIRECTION | | 0..0

obtained as the concatenation of the 32 bit COUNT,5 bit BEARER ,1 bit DIRECTION values and a string of 26 zero bits. The counter BLKCNT is set to zero.

The f8 algorithm makes use of a Key Modifier (KM) constant that is equal to the octet 01010101 repeated 16 times. First, a single operation of KASUMI is applied to register A ,using modified version of CK to compute prewhitening value:

W=KASUMI CK $\oplus$ KM(IV) which is stored in register A. Once the keystream generator has been initialized in this manner, it is ready to be used to generate keystream bits. The plaintext / ciphertext to be encrypted/decrypted consists of LENGTH bits ,where LENGTH varies between 1 to 20000 with granularity of 1 bit ,while the keystream generator produces keystream bits in multiples of 64 bits. Between 0 and 63 of least significant bits are discarded from the last block depending on the total number of bits required by LENGTH.

The number of required keystream bits is denoted by BLOCKS, whose value is determined by value of the LENGTH parameter as follows: the value of LENGTH is divided by 64 and result is rounded up to the nearest integer .The keystream blocks are denoted

as KSB_1,KSB_2,...,KSB Blocks .Set KSB_0 =0 and let n be an integer with 1≤=n≤BLOCKS such that n=BLKCNT+1 and set :

KSBn=KASUMI CK(W ⊕ (n-1) ⊕KSBn-1)

Individual bits KS[0],KS[1],....KS[LENGTH-1] of the keystream are extracted in turn from KSB_1 to KSB blocks ,with the most significant bit extracted first by applying the following operation. For n=1,....BLOCKS and for each integer i ,with 0≤=i≤63,set

KS[(((n-1)*64)+i]=KSBn[i]

Encryption/Decryption operations are identical and carried out by bitwise XOR of the input data using the generated keystream.

5.7 Requirements for the Integrity Algorithm:

5.7.1 Overview: A cryptographic function f9 is used to protect data integrity and authenticate the data origin of signalling data at the RRC layer. The f9 function is implemented in the UE and the RNC, and to support interoperability it must be fully standardized. Similarly to the confidentiality algorithm, the integrity algorithm should be designed to accommodate a range of implementation options including hardware and software implementations.

A cryptographic message authentication algorithm generates a fixed length MAC from a message of arbitrary length, under the control of the secret parameter key and a set of initialization values. The sender and receiver generate the MAC using the same function. The sender sends its MAC result to the receiver , who compares the received MAC value with the expected MAC value computed by the receiver. The receiver accepts MAC if the compared values are equal. Figure 5.3 illustrates how f9 is used to derive a MAC-I(I=Integrity of the signalling data) on a signalling message.

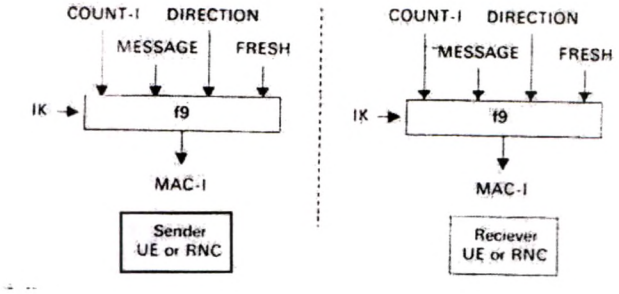


Figure 5.3 Derivation of MAC-I using f9

The input parameters to the integrity algorithm are the Integrity Key(IK),a time and frame dependent input (COUNT-I),a random value generated by the network side (FRESH), the direction bit(DIRECTION) and the signalling message (MESSAGE).IK is a cryptographic key that is newly generated at each authentication process. The COUNT-I, FRESH and DIRECTION parameters can be considered as a set of initialization parameters that are renewed for each message to be authenticated.

Based on these input parameters, the user computes using the f9 function the MAC-I for data integrity, which is appended to the message when sent over the radio access link. The receiver computes the expected MAC (XMAC-I) value on the message received in the same way as the sender computed MAC-I on the message sent.

5.7.2 Interface

5.7.2.1 Integrity Key(IK): The length of IK is 128 bits.

5.7.2.2 Frame dependent input (COUNT-I) : The length of COUNT-I is 32 bits .It protects against replay during a connection because its value is incremented by 1 for each input message. COUNT-I consists of two parts: The Hyper Frame Number(HFN) as the most significant part and a Radio Resource Control (RRC) sequence number (SQN) as the least significant part.

The COUNT- I parameter is initialized at connection set- up using the same procedure as for COUNT-C .The initial value of HFN is sent by the user to the network at connection set up. The user stores the most significant part of the greatest-used HFN from the

previous connection and increments it for the new connection. In this way, the user is assured that no COUNT-I value is reused by the network using the same IK.

5.7.2.3 One time random number (FRESH): The length of fresh is 32 bits.The same IK may be used for several consecutive connections. This fresh value is input to the algorithm to assure the network side that the user is not replaying old MAC-Is.

5.7.2.4 The signalling data(MESSAGE): The maximum value of the keystream length generated by f8 is defined to be 20000 , while no limit is put on the input message length of f9.

5.7.2.5 Direction of Transmission(DIRECTION): The length of DIRECTION is 1 bit. The same IK may be used for uplink and downlink channels associated with the UE. The value of the DIRECTION is 0 for messages from the UE to the RNC and 1 for messages from the RNC to the UE.

5.7.2.6 Message Authentication Code(MAC-I) and expected MAC-I(XMAC-I) : The length of MAC-I is 32 bits. The UMTS security specification makes use of two other MACs-MAC-A and MAC-S –which are used in the authentication process.

5.8 Integrity Algorithm f9: The 3GPP integrity algorithm f9 computes a 32 bit MAC on an input message under an IK and imposes no limitation on the input message length.

For ease of implementation the algorithm is based on the same block cipher KASUMI as used by confidentiality algorithm f8.The approach adopted uses KASUMI in a form of CBC-MAC mode, which modifies the standard CBC-MAC mode by adding an operation that combines all intermediate outputs using bitwise XOR and applies one more KASUMI operation on the combined block. The 64 bit output from the last KASUMI operation is then truncated to produce a 32 bit MAC-I. The following figure shows the operation of the 3GPP standard algorithm f9.

5.9 f9 Description: The 3GPP standard f9 function makes use of two 64-bit registers A and B. The initial value for both registers is set equal to 0:A=0 and B=0.The function also makes use of a constant value for a 128-bit KM that is equal to 16 repetitions of the octet 0xAA=10101010.The inputs to the f9 function are defined: the 32 bit value COUNT,32 bit

value FRESH, a bit string MESSAGE of unlimited length LENGTH and the 1-bit value DIRECTION.

The values of all inputs are concatenated and then a single "1"bit is appended to this string, followed by between 0 and 63 "0"bits,so that the total length of the resulting string is an integer multiple of 64 bits. This string is called the Padded String(PS).Then:

PS=COUNT || FRESH || MESSAGE || DIRECTION || 1 || 0...0

Where the number of "0" bits in the padding field is between 0 and 63. The PS is then split into 64 bit blocks Ps_i. Let BLOCKS be the number of the resulting 64-bit blocks. Then :

$PS = PS_0 \,||\, PS_1 \,||\, PS_2 \,||\, \ldots \,||\, PS_{BLOCKS-1}$

This PS is the data input to the MAC algorithm. It would also be possible to intepret the first block PS_0 of PS as the initial value, since PS_0 = COUNT || FRESH, and this initial value would be different for each message. For each integer n with $0 \le n \le BLOCKS - 1$ the following operations are performed.

$A = KASUMI_{IK}(A \oplus PSn)$

$B = B \oplus A$

Finally a further application of KASUMI is performed using a modified form of the IK as follows :

$B = KASUMI_{IK \oplus KM}(B)$

The output from KASUMI has 64 bits: MAC-I comprises the leftmost 32 bits of the result and the rightmost 32 bits are discarded.

5.10 IDEA (International Data Encryption Algorithm)

This algorithm can be considered for application layer security. PGP(Pretty Good Privacy) is a freeware for email security and it uses IDEA for encryption. IDEA is a block cipher that uses a 128-bit key to encrypt data in blocks of 64 bits. By contrast, DES also uses 64 bit blocks but a 56 bit key.

5.10.1 Implementation Considerations: IDEA is designed to facilitate both software and hardware implementation. Hardware implementation, typically in VLSI, is designed to achieve high speed. Design principles for hardware implementation are as follows:

(i) Similarity of encryption and decryption: Encryption and decryption should differ only in the way of using the key so that the same device can be used for both encryption and decryption. IDEA has a structure that can satisfy this requirement.

(ii) Regular structure: The cipher should have a regular modular structure to facilitate VLSI implementation. IDEA is constructed from two basic modular building blocks repeated multiple times.

5.10.2 IDEA encryption: The single round of IDEA encryption is shown in the *Figure 5.4* There are two inputs to the encryption function: I) the plaintext to be encrypted II) the key. Plaintext is 64 bits in length and the key is 128 bits in length in this particular case. The 64 bit data block is divided into four 16-bit sub blocks: X1, X2, X3 and X4.These four sub blocks become input to the first round of algorithm.

The IDEA algorithm consists of eight rounds followed by a final transformation. In each round, the sequence of events is as follows:

1. Multiply X1 and the first subkey
2. Add X2 and the second subkey
3. Add X3 and the third subkey
4. Multiply X4 and the fourth subkey
5. XOR the results of steps (1) and (3)
6. XOR the results of steps (2) and (4)
7. Multiply the results of step (5) with the fifth subkey
8. Add the results of steps (6) and (7)
9. Multiply the results of step (8) with the sixth subkey
10. Add the results of steps (7) and (9)
11. XOR the results of steps (1) and (9)
12. XOR the results of steps (3) and (9)
13. XOR the results of steps (2) and (10)
14. XOR the results of steps (4) and (10)

The output of the round is the four subblocks that are results of steps (11) to (14). Swap the two inner blocks (except for the last round) and that's the input to the next round. After the eighth round, there is final output transformation as shown in figure 5.5 :

(i) Multiply W81 and the first subkey Z49

(ii) Add W82 and the second subkey Z50

(iii) Add W83 and the third subkey Z51

(iv) Multiply W84 and the fourth subkey Z52

Finally, the four sub blocks are reattached to produce cipher text. Creating the subkeys is easier. The algorithm uses 52 of them (six for each of the eight rounds and four more for the output transformation.) First, the 128 –bit key is divided into eight 16-bit subkeys. These are first eight subkeys for the algorithm. Then the key is rotated 25 bits to the left and again divided into eight subkeys. The first four are used in round 2; the last four are used in round 3. The key is rotated another 25 bits to the left for the next eight subkeys, and so on until the end of the algorithm.

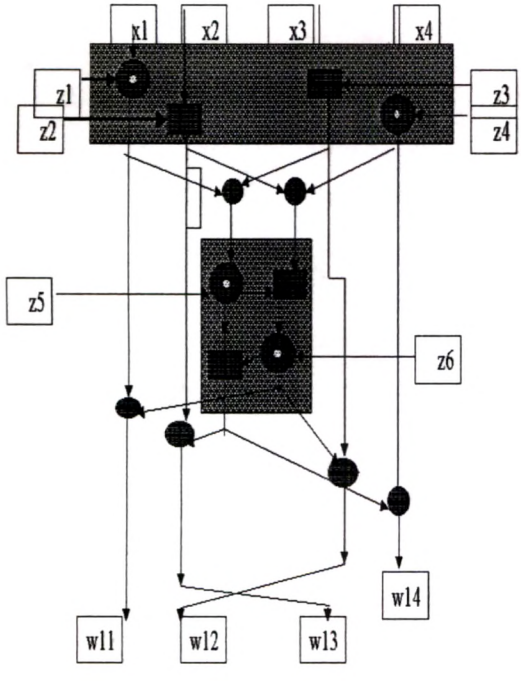


Figure *5.4 : Single Round of IDEA*

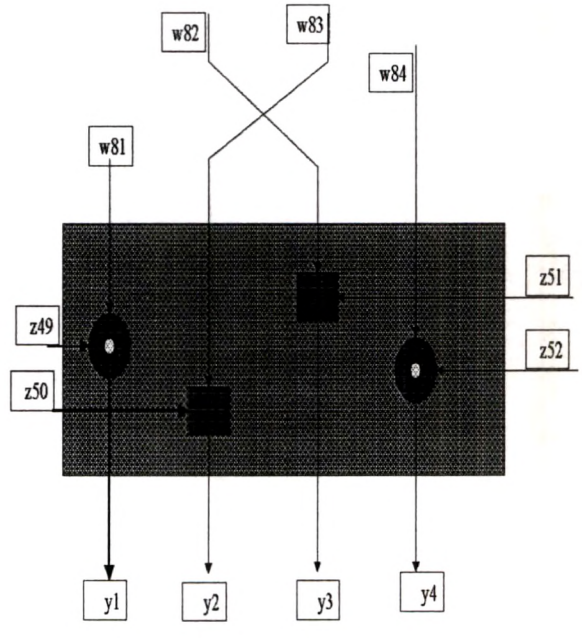


Figure *5.5 : Output Transformation*

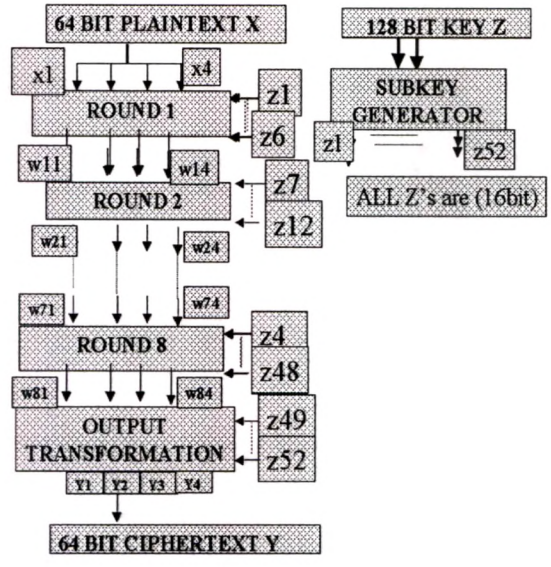


Figure 5.6 : Overall IDEA Structure

5.11 Advanced Encryption Standard (Rijnadael Algorithm) : Rijndael algorithm is a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. Rijndael was also designed to handle additional block sizes and key lengths. AES algorithm may be used with the three different key lengths indicated above, and therefore these different "flavors" may be referred to as "AES-128", "AES-192", and "AES-256".

5.11.1 Algorithm Parameters, Symbols, and Functions

The following algorithm parameters, symbols, and functions are used throughout the description of AES.

AddRoundKey() Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation. The length of a Round Key equals the size of the State (i.e., for *Nb* = 4, the Round Key length equals 128 bits/16 bytes).

InvMixColumns() Transformation in the Inverse Cipher that is the inverse of MixColumns()

InvShiftRows ()	Transformation in the Inverse Cipher that is the inverse of ShiftRows().
InvSubBytes ()	Transformation in the Inverse Cipher that is the inverse of SubBytes().
K	Cipher Key.
MixColumns ()	Transformation in the Cipher that takes all of the columns of the State and mixes their data (independently of one another) to produce new columns.
Nb	Number of columns (32-bit words) comprising the State. For this standard, *Nb* = 4.
Nk	Number of 32-bit words comprising the Cipher Key. For this standard, *Nk* = 4, 6, or 8.
Nr	Number of rounds, which is a function of *Nk* and *Nb* (which is fixed). For this standard, *Nr* = 10, 12, or 14.
Rcon [ ]	The round constant word array.
RotWord ()	Function used in the Key Expansion routine that takes a four-byte word and performs a cyclic permutation.
ShiftRows ()	Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.
SubBytes ()	Transformation in the Cipher that processes the State using a non-linear byte substitution table (S-box) that operates on each of the State bytes independently.
SubWord ()	Function used in the Key Expansion routine that takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word.
XOR	Exclusive-OR operation.

5.11.2 Inputs and Outputs

The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. Other input, output and Cipher Key lengths are not permitted by this standard.

The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges $0 \le i < 128$, $0 \le i < 192$ or $0 \le i < 256$ depending on the block length and key length (specified above).

5.11.3 Bytes

The basic unit for processing in the AES algorithm is a byte, a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes.

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order $\{b7, b6, b5, b4, b3, b2, b1, b0\}$. For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by $Nb = 4$, which reflects the number of 32-bit words (number of columns) in the State.

5.11.4 Algorithm Operation : For the AES algorithm, the length of the Cipher Key, K, is 128, 192, or 256 bits. The key length is represented by $Nk = 4$, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key.

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$. For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the State array by

different offsets, 3) mixing the data within each column of the State array, and 4) adding a Round Key to the State.

5.11.5 Cipher

At the start of the Cipher, the input is copied to the State array using the conventions described in Sec. 5.11.1. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first $Nr\tilde{1}$ rounds. The final State is then copied to the output .The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine .

5.12 Authentication and Key Generation Algorithm :

5.12.1 Introduction: Authentication and key generation algorithms for UMTS need not be standardized. Operators are given opportunity to choose the algorithm they want to rely on for network access and call origin authentication. Algorithms will be implemented at the network operator's Authentication Centres (AuCs) and in the USIMs of the mobile devices of the same operator's subscribers. This eliminates the need of interoperability between different operators. However, use of standard algorithm can achieve interoperability between different USIM implementations and relieves many operators from a big task of design and implementation of a cryptographic algorithm.

The design of the example 3GPP AKA function was completed in December 2000 .The design was based on the Rijnadael block cipher algorithm, which has become the AES standard.

The UMTS AKA system makes use of a number of different types of cryptographic algorithms to perform various security tasks. Total eight different functions are required .They are as follows:

f0 the random challenge generating function

f1 the network authentication function

f1* the resynchronization message authentication function

f2 the user authentication function(AUTN)

f3 the Cipher Key (CK) derviation function

f4 the Integrity Key (IK) derviation function

f5 the Anonymity Key (AK) derivation function for normal operation

f5* the AK derivation function for resynchronization

All authentication functions other than f0 are keyed cryptographic functions , which means that they are given as families of functions with the key K as parameter. Application of function fi with key K is denoted as fi_K.

5.12.2 Generation of quintets in the AuC :

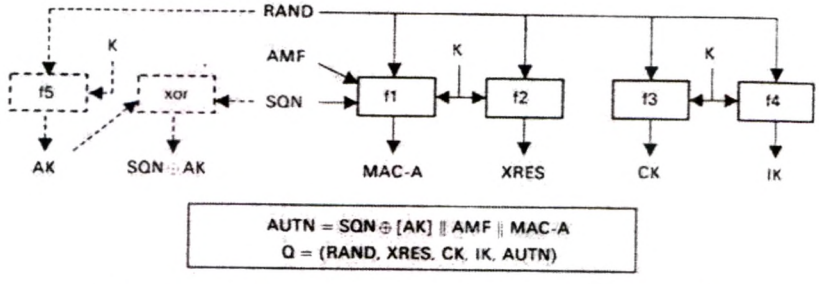


Figure 5.7 Generation of Quintets in the AuC

To generate a quintet (fig. 5.7), the Home Location Register(HLR)/AuC first generates a random number (RAND) using the f0 function. Then it computes a Message Authentication Code (MAC) for authentication MAC-A=f1K (SQN‖RAND‖AMF),an expected response XRES=$f2_k$(RAND),a CK=$f3_k$(RAND) and IK=$f4_k$(RAND).The HLR/AuC assembles the authentication token AUTN= SQN ‖AMF‖MAC-A and forms quintet Q=(RAND,XRES,CK,IK,AUTN).The quintet Q is sometimes called the "authentication vector".

If the SQN is to be concealed, the HLR/AuC also computes an AK=$f5_k$(RAND) and computes the concealed SQN as SQN exor AK.In this case the AUTN in the quintet is formed as AUTN =(SQN ⊕ AK) ‖AMF‖MAC-A. Concealment of SQN is optional.

5.12.3 Authentication and key derivation in the USIM : On receipt of a (RAND,AUTN) pair, the USIM acts as follows: First it retrieves the unconcealed SQN. If the SQN is concealed, the USIM computes AK=f5k(RAND) and retrieves the SQN by computing SQN=(SQN⊕AK) ⊕AK. Then the USIM computes XMAC-A =$f1_k$(SQN‖RAND‖AMF),the

user response RES=$f2_K$(RAND),the CK=$f3_k$(RAND) and the IK=$f4_k$(RAND). This is illustrated in fig 5.8.

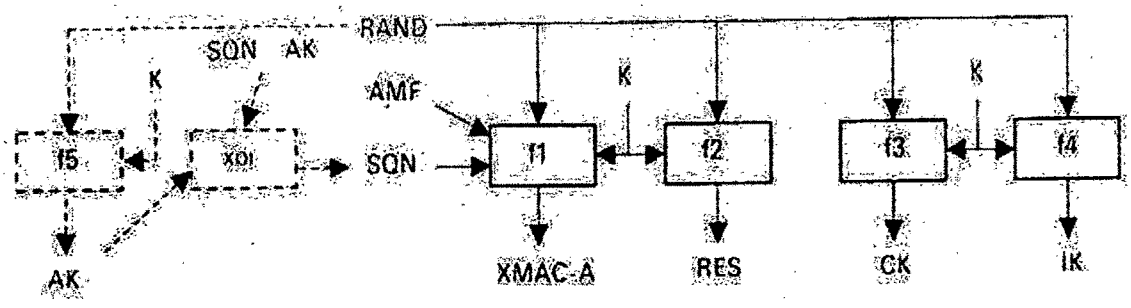


Figure 5.8 Authentication and key derivation in USIM

5.12.4 Generation of resynchronization token in the USIM: When there is a synchronization failure, the USIM generates a resynchronization token.It computes MAC-S=$f1^*_K$ (SQN_{MS} || RAND || AMF^*) where AMF^* is the default value for AMF used in resynchronization. Then, the resynchronization token is constructed as AUTS=SQN_{MS} || MAC-S. If SQN_{MS} is to be concealed by means of an AK, the USIM computes AK=$f5^*_K$ (RAND) and the concealed counter value is then computed as SQN_{MS} ⊕ AK. In this case the resynchronization token is formed as AUTS=(SQN_{MS} ⊕ AK) || MAC-S.

5.12.5 Resynchronization at the HLR/AuC: On receipt of an indication of synchronization failure and an (AUTS,RAND) pair, the HLR/AuC takes the following actions. It computes XMAC-S = $f1^*_K$ (SQN_{MS} || RAND || AMF^*),where AMF^* is the default value for AMF used in resynchronization. If SQN_{MS} is concealed by means of an AK, then the HLR / AuC must complete AK = $f5^*_K$ (RAND) and retrieve the unconcealed counter value as SQN_{MS} = (SQN_{MS} ⊕ AK) ⊕ AK. Then, the AuC verifies whether MAC-S included in the AUTS is equal to the computed value XMAC-S.

5.13 Functional Requirements for UMTS authentication: In document on Cryptographic algorithm requirements (Release 4)[62], the functional requirements are specified. They are described in this section.

5.13.1 Use: Functions f0-f5 are only used to provide mutual entity authentication between USIM and the AuC, to derive keys to protect user and signalling data transmitted over the radio access link and to conceal the SQN to protect user identity confidentiality. The f1* function is only used to provide data origin authentication for synchronization failure information sent by USIM to the AuC. The f5* function is only used to provide user identity confidentiality during resynchronization .

5.13.2 Allocation : Functions f1-f5 and f5* are allocated to the AuC and the USIM. The f0 function is allocated to the AuC.

5.13.3 Extent of standardization: Functions f0-f5, f1* and f5* are proprietary to the Home Environment (HE). Examples of f1, f1* and f2 functions are CBC MACs or HMACs.[34]

5.13.4 Implementation and operational considerations : Functions f1-f5,f1* and f5* are designed so that they can be implemented on an Integrated Circuit (IC) card equipped with a 8-bit microprocessor running at 3 MHz with 8 Kbyte of ROM and 300 bytes of RAM and produce an AK,XMAC-A ,RES,CK and IK in less than 500 ms of execution time.

5.13.5 Types of Cryptographic functions: The following cryptographic requirements were allocated to each function in [62]:

- f0 is the random challenge –generating function. It should be a pseudo random number-generating function and map the internal state of the generator to the challenge value RAND.
- f1 is the network authentication function. It should be a keyed MAC function that takes the subscriber key K as the key and maps the data (SQN,RAND,AMF) to MAC-A (or XMAC-A).In particular, it must be computationally infeasible to derive K from knowledge of RAND,SQN, AMF and MAC-A or XMAC-A.
- f1* is the resynchronization message authentication function. It should be a keyed MAC function that takes the subscriber key K as the key and maps the data (SQN, RAND, AMF*) to MAC-S or XMAC-S. In particular, it must be computationally infeasible to derive K from knowledge of RAND, SQN, AMF*and MAC-S or XMAC-S.

- f2 is the user authentication function. It should be a keyed MAC function that takes the subscriber key K as the key and maps the challenge RAND to RES or XRES. In particular, it must be computationally infeasible to derive K from knowledge of RAND and RES or XRES.
- f3 is the CK derivation function that takes the subscriber key K and the random challenge RAND as inputs and produces the CK as output. It should be a key derivation function .In particular, it must be computationally infeasible to derive K from knowledge of RAND and CK.
- f4 is the IK derivation function that takes the subscriber key K and the random challenge RAND as inputs and produces the IK as output. It should be a key derivation function. In particular, it must be computationally infeasible to derive K from knowledge of RAND and IK.
- f5 is the AK derivation function for normal operation that takes the subscriber key K and the random challenge RAND as inputs and produces the AK as output. It should be a key derivation function. In particular, it must be computationally infeasible to derive K from knowledge of RAND and AK. Its use is optional.

The defined set of algorithms is commonly called the MILENAGE algorithms. The MILENAGE algorithm set makes use of the following components :

- A block cipher encryption function that takes a 128-bit input and 128 bit key and returns a 128 bit output.
- A 128 bit value OP .This is an operator –variant algorithm configuration field. The algorithm set is designed to be secure whether or not OP is publicly known or not.

The block cipher Rijnadael is used as the kernel in the MILENAGE constructions because it is a strong encryption algorithm, effective on several platform, highly suitable for smart card implementations and freely available without any kind of IPR limitations. The MILENAGE function computation is graphically shown in the following figure 5.9

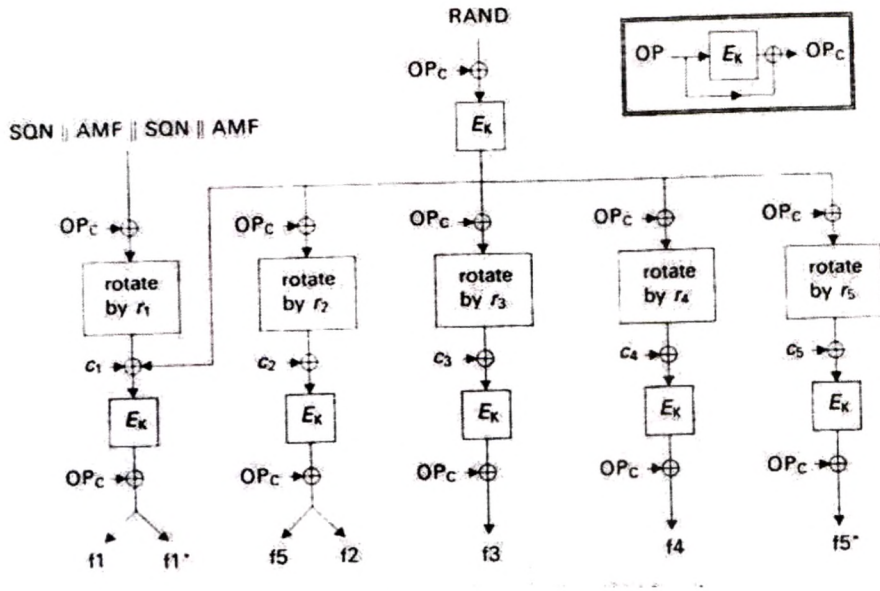


Figure 5.9 Computation of MILENAGE functions

5.14 Summary : This chapter has described the algorithms f8,f9,KASUMI,IDEA, AES and MILENAGE. Their basic structure and general implementation aspects are discussed. The implementation of these algorithms basically depends upon blocks of arithmetic and logical operations performing required transformation from plaintext to ciphertext. The actual hardware and software implementations are described in the following chapters.