UTRAN CRYPTOGRAPHIC ALGORITHMS VERIFICATION AND IMPLEMENTATION

[1]GHIZLANE ORHANOU, [2]SAÏD EL HAJJI, [3]JALAL LAASSIRI, [4]YOUSSEF BENTALEB

[1, 3, 4] Doctor, Département Math. et Informatique, Université Med V Agdal, Faculté des Sciences, Maroc

[2] Professor, Département Math. et Informatique, Université Med V Agdal, Faculté des Sciences, Rabat, Maroc

ABSTRACT

In the present paper, we are interested in Universal Mobile Telecommunications System (UMTS) Access Network security. A special interest is given to the protection of the data integrity and the provisioning of data encryption. Indeed, the appropriate procedures and cryptographic algorithms are discussed. In previous work, we were interested in the study of the operation and complexity of the algorithms, but actually we will focus on other aspects. A closer look is taken at the two sets of UMTS cryptographic algorithms: UEA1/UIA1 (UEA indicates UMTS Encryption Algorithm and UIA UMTS Integrity Algorithm) based on the KASUMI algorithm and UEA2/UIA2 based on the SNOW 3G algorithm. Furthermore, this paper includes the results of the verification and the implementation of the two sets of the UMTS cryptographic algorithms. The necessary corrections and/or adaptation of the 3GPP algorithms codes having carried out to meet the 3GPP algorithms specifications. Furthermore, we propose an adaptation of the second set of algorithms to the little-endian machines since the 3GPP proposed codes are only limited to the big-endian machines. These corrections and adaptations are presented in the present paper and some implementation examples are presented as well.

Keywords: *UMTS, Confidentiality, Integrity, SNOW 3G, Verification, Implementation*

1. SECURITY MECHANISMS IN THE UMTS ACCESS NETWORK

The Access Network security is carried out through a set of security features which offer to the UMTS user a safe and secure access to 3G services over the air interface [1, 2, 3]. The following features are provided:

- User identity confidentiality;
- Mutual authentication of the network and the user;
- Confidentiality;
- Data integrity.

These functionalities protect against attacks which threaten data on the network access link [3, 4].

In the present paper, we will focus on the two last security features.

1.1. Confidentiality And Data Integrity In The UMTS

1.1.1. Confidentiality

User data and some signaling data are considered sensitive and their confidentiality should be protected over the radio access link. To ensure this data confidentiality on the air interface, the following features are provided [1, 2]:

- Cipher algorithm (f8) agreement: nowadays, there exist two variants of the cipher algorithm: UEA1 based on KASUMI algorithm [1, 5, 6] and UEA2 based on SNOW 3G algorithm [1, 7, 8, 15]. The MS (*Mobile Station*) and the SN (*Serving Network*) can securely negotiate the algorithm to use in their mutual communication.
- Cipher key (CK) agreement: the agreement is done between the MS and SN during the Authentication and Key Agreement procedure;
- Confidentiality of user and signaling data;

1.1.2. Integrity

Data integrity in the UMTS network ensures the protection of the signaling data integrity and allows the authentication of the signaling messages transmitted between the user and the serving network [1, 2, 9].

The following security features are provided to ensure the signaling data integrity on the network access link:

- Integrity algorithm (f9) agreement: as for the data confidentiality, there is actually two variants of the integrity algorithm: UIA1 based on KASUMI algorithm and UIA2 based on SNOW 3G algorithm.
- Integrity key (IK) agreement;
- Data integrity and origin authentication of signaling data: the receiving entity (MS or SN) must be able to check that the signaling data wasn't modified during its transition over the network access link and to check the expected origin of the message (SN or MS).

In the following subsections, we will introduce the UMTS confidentiality and integrity mechanisms.

1.2. UMTS Encryption Function f8

The need for a confidentiality protected mode of transmission is fulfilled by an UMTS confidentiality cryptographic function f8 [1, 2, 3] which is a symmetric synchronous stream cipher. This type of ciphering has the advantage to generate the mask of data before even receiving the data to encrypt, which help to save time. Furthermore, it is based on bitwise operations which are carried out quickly.

"Figure 1" bellow illustrates the Encryption/ Decryption operations using the f8 function.

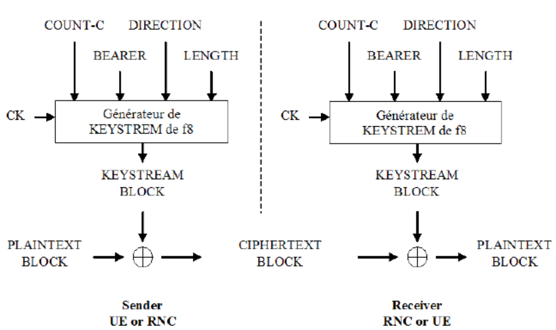

Figure 1. Encryption/Decryption mechanism

The input parameters of f8 are the following:

- CK : Cipher Key;
- COUNT-C: Frame dependent input used to synchronize the sender and the receiver;
- BEARER: Service bearer identity;
- DIRECTION: Direction of the transmission;
- LENGTH: Number of bits to be encrypted/decrypted;

As mentioned above, there exist nowadays two encryption algorithms UEA1 et UEA2.

UEA1, which was used since the genesis of the UMTS network in 1999, is a stream cipher based on KASUMI [10, 11]. This last algorithm is a block cipher used under its OFB operation mode [12].

The second one, UEA2, is also a stream cipher but based on another stream cipher named SNOW 3G. It was introduced as 3GPP standard on 2006.

1.3. UMTS Integrity Function f9

To ensure signaling data protection, a message authentication function f9 shall be applied to these information elements transmitted between the ME (Mobile Equipment) and the RNC (Radio Network Controller). It's a one-way function which generates a 32-bit output MAC-I under the control of 128-bit Integrity Key IK [2, 3, 11].

"Figure 2" bellow illustrates the calculation mechanism of the message authentication code MAC-I using the f9 function.

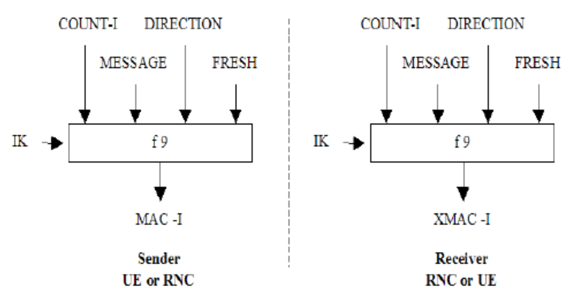

Figure 2. Derivation of MAC-I (or XMAC-I) [1, 2]

The algorithm input parameters are the following:

- IK: Integrity Key;
- COUNT-I: Frame dependent input;
- FRESH: Random number generated by the network;
- DIRECTION: Direction of the transmission;
- MESSAGE: Input bit stream;

Based on these input parameters, the message authentication code MAC-I is calculated.

2. KASUMI BASED ALGORITHMS - UEA1/UIA1

In the present section, we will study first the algorithms UEA1 and UIA1 and their operation modes. Then, we will focus on the verification of the UEA1 and UIA1 codes given by the 3GPP specification documents [1, 5]. After that, we will expose the results of the implementation of both algorithms after having carried out the necessary corrections to the 3GPP algorithms codes to meet the 3GPP algorithms specifications and requirements. These corrections will be exposed as well.

2.1. Encryption Algorithm UEA1

UEA1 uses, as keystream generator, the cipher block KASUMI under its OFB (*Output-FeedBack mode*) operation mode to produce an output KEYSTREAM [10, 12]. This keystream, which the length is multiple of 64 bits will be used to encrypt/decrypt the user or signaling data.

Concerning KASUMI algorithm, in the present paper, we will just say that it is a block cipher algorithm which take a 64-bit input to produce a 64-bit output under a 128-bit control [1, 6, 11].

We can distinguish three principal steps during the UEA1 operation: initialization of the keystream generator, keystream generation and finally data encryption/decryption. These steps are presented bellow.

2.1.1. Initialization

Before generating the keystream, the keystream generator is initialized with the input parameters [1, 5].

- The 64-bit register A_0 is set to:

  COUNT || BEARER || DIRECTION || 0 … 0

  i.e. A_0 = COUNT[0] … COUNT[31] BEARER[0] … BEARER[4] DIRECTION[0] 0 … 0

- The counter BLKCNT is set to zero and the key modifier KM is set to the 128-bit value:

  KM = 0x55555555555555555555555555555555

- KSB_0 is also set to zero.

Then, we apply one operation of the cipher block KASUMI to the register A_0 under the control of the modified confidentiality key CK ⊕ KM.

A = KASUMI[A] $_{CK \oplus KM}$

"Figure 3" bellow illustrates this step.

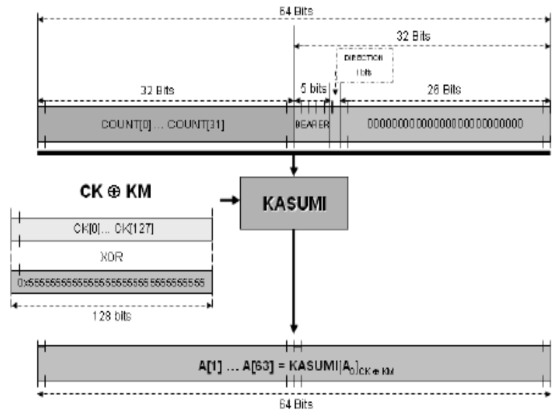

Figure 3. Keystream generator initialization

2.1.2. Keystream generation

Once the keystream generator is initialized, it becomes ready for the keystream bits generation. "Figure 4" illustrates keystream generation principal for the UEA1 algorithm.

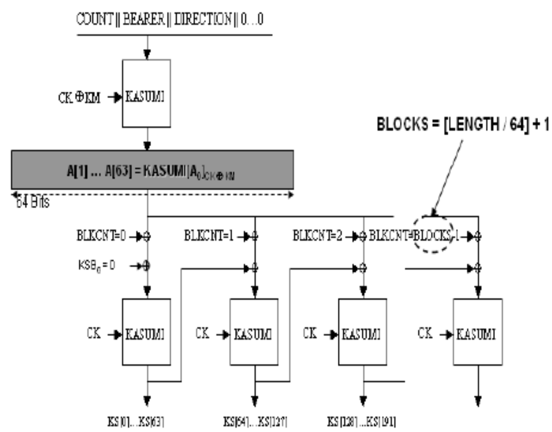

Figure 4. UEA1 Keystream generation

It is important to mention that the PLAINTEXT /CIPHERTEXT number of bits is determinated by the input parameter LENGTH (which isn't necessary a multiple of 64). So since the generated keystream bits number is multiple of 64, some bits, between 0 and 63 bits, in the last produced keystream block will be discarded to meet the exact length of the message to encrypt/decrypt.

"Figure 5" bellow shows the n^{th} UEA1 execution state, with $1 \le n \le$ BLOCKS = [LENGTH/64] + 1.

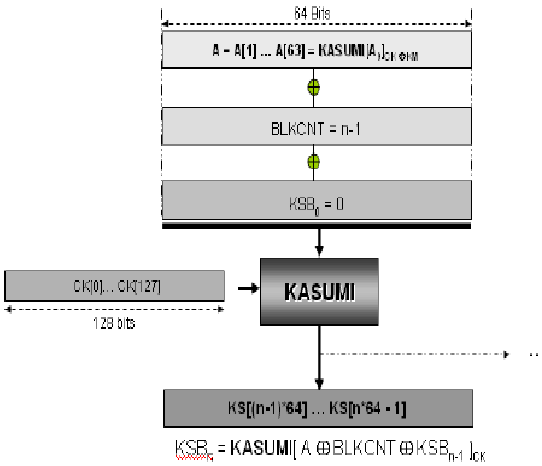

Figure 5. n^{th} block keystream generation

2.1.3. Data encryption / decryption

Encyption/decryption operations are identical and are performed by the exclusive-OR operation (XOR) of the input data IBS (Input Bit Stream) (which is the message to encrypt/decrypt) with the generated keystream (KS) to generate the output OBS (Output Bit Stream).

For each integer i with $0 \leq i \leq$ LENGTH-1 we define:

$$OBS[i] = IBS[i] \oplus KS[i].$$

2.2. Integrity Algorithm UIA1

The integrity algorithm UIA1 is a message authentication function which produces a 32-bit Message Authentication Code (MAC) as an output, under the control of the 128-bit integrity key IK. It's based on the cipher block KASUMI used under its CBC-MAC operation mode [10, 11, 12]. A 64-bit digest is generated and only its left half (the most significant 32 bits) constitutes the output value MAC-I.

To do this, UIA1 performs two important steps presented bellow [1, 5].

2.2.1. Initialization

First, the keystream generator is initialized with the input parameters before generating the keystream bits.

- The registers A and B are set to 0: A = 0 and B = 0
- The UIA1 key modifier is set to:

  KM = 0xAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAA

- After concatenating the input data, we append a single '1' bit followed by between 0 and 63 '0' bits so that the total length of the resulting string PS (Padded String) is multiple of 64 bits:

  PS = COUNT[0] … COUNT[31] FRESH[0] … FRESH[31] MESSAGE[0] … MESSAGE[LENGTH-1] DIRECTION[0] 1 0* (0* indicate `0' bits between 0 and 63.)

2.2.2. MAC-I calculation

After the initialization step, the cipher block KASUMI will be used in its CBC-MAC operation mode to generate the MAC-I.

The padded string PS, introduced in the initialization step, is splitted into 64-bit blocks PS_i where:

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

Then, the following operations are performed for each integer n with $0 \leq n \leq$ BLOCKS-1:

$$A=KASUMI[A \oplus PS_n]_{IK}$$

$$B=B \oplus A$$

Finally, the algorithm KASUMI, using the modified integrity key, is applied to the result as shown bellow.

$$B = KASUMI[B]_{IK \oplus KM}$$

MAC-I is the 32-bit left half of the result:

$$MAC\text{-}I=lefthalf[B]$$

"Figure 6" bellow shows the steps followed to calculate the MAC-I.

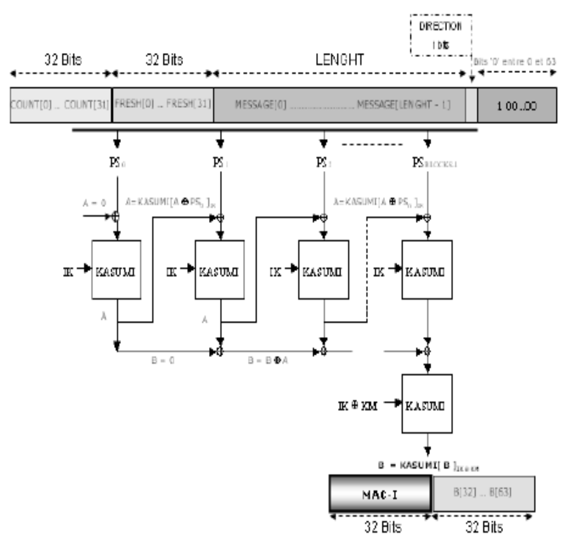

Figure 6. MAC-I (or XMAC-I) calculation

2.3. Verification and Implementation

After the close study of the algorithms UEA1 and UIA1, we will expose now the result of our verification, rectification of the both algorithms whose codes are given by 3GPP specifications. Then, we will present the practical implementation of the correct version of the codes. We note that the algorithms UEA1 and UIA1 are coded in the C language.

2.3.1. UEA1 verification and rectification

In our UEA1 algorithm verification task, we have used the TestSets given by the 3GPP Implementation document [1, 13, 14]. These TestSets are given by the 3GPP for the algorithms implementors to test the correctness of their implementations in respect to 3GPP requirements.

Based on this, we have found that for a given input data and for a given key and plaintext (Data taken from the TestSets), the ciphertext generated by the algorithm UEA1 in its 3GPP code version was incorrect. Thus, our objective was to check these codes, and give a correct version of these security algorithms which are supposed to be implemented in both MEs (*Mobile Equipments*) and RNCs (*Radio Network Controller*) in a UMTS Access Network in order to ensure data confidentiality and integrity over the air interface. In the following paragraphs, we will expose the corrections made.

Indeed, after studying deeply the UEA1 C-language code, we have found two main problems:

1. The first one was that the instruction used to generate the ciphertext from the exclusive-OR operation between the plaintext and the keystream was coded incorrectly. So, to remedy to this problem, we have replaced the 3GPP instruction bellow: **data++ ^= temp.b8[i];*

   by the following instruction block:

   ```
   *data ^= temp.b8[i];
   d[i] = *data;
   *data = *data++;
   ```

2. and then, we have added the following instructions to control the length of the ciphertext produced in accordance with the LENGTH parameter value. This issue is very important since the ciphertext and the plaintext must have the same number of bits but it was omitted in 3GPP code version. You find bellow the instruction block added:

   ```
   j = 8 - j;
   if (length < 64)
   {
       d[n-1] = d[n-1] >> j;
       d[n-1] = d[n-1] << j;
   }
   ```

   d[n-1] represents de n^{th} block of the generated ciphertext, which represents the last block where the length control has to be performed.

It is important to mention here that the corrections brought to the algorithm have no impact on its security and its robustness since it doesn't modify any thing in the algorithm internal processing.

2.3.2. UEA1 implementation example [1]

The implementation data parameters are given by the 3GPP tests and implementation documents [1, 14].

These tests were chosen by the standardization group to give the implementors a way to check the correctness of their implementations. You find bellow one Testset example for the UEA1 algorithm.

<u>Input Data:</u>

Key = 2BD6459F 82C440E0 952C4910 4805FF48

Count = C675A64B

Bearer = 0C

Direction = 1

Length = 798 bits

Plaintext:
7EC61272743BF1614726446A6C38CED1

66F6CA76EB5430044286346CEF130F92

922B03450D3A9975E5BD2EA0EB55AD8E

1B199E3EC4316020E9A1B285E7627953

59B7BDFD39BEF4B2484583D5AFE082AE

E638BF5FD5A606193901A08F4AB41AAB

9B134880

Expected Ciphertext [14]:

1061793DAAACBE40C9431E292B7FF494

96DB0D31CE24710C01ACFF1B2C441FA9

3BB3BD65DE18027A14CCA571A42E8B12

74AE30AC411AB6AFD88F924E65F9812D

FA80EF8E9A7EA753391D09F480D9147C

B39C23A1ACB9AC9B2A6B4709F7E6DD84

D8FA59A4

With the UEA1 3GPP version after correction of the first problem presented in subsection 2.3.1, we have obtained the result presented in the following "Figure 7".

UEA1 EXECUTION

blkcnt	Kasumi input	KASUMI ---> KEYSTREAM	XOR PLAINTEXT	= CIPHERTEXT
0	72ab58897d8acdaa	6ea76b4fde974f21	7ec61272743bf161	1061793daaacbe40
1	1c0c33c6a31d828a	8e655a4347473a45	4726446a6c38ced1	c9431e292b7ff494
2	fcce02ca3acdf7ed	f02dc74725704108	66f6ca76eb543004	96db0d31ce24710c
3	82869fce58fa8ca1	432acb77c357103b	4286346cef130f92	01acff1b2c441fa9
4	318193febedddd95	a998be20d3229b0f	922b03450d3a9975	3bb3bd65de18027a
5	db33e6a9aea856a0	f1718bd14f7b269c	e5bd2ea0eb55ad8e	14cca571a42e8b12
6	83dad35832f1eb30	6fb7ae92852bd68f	1b199e3ec4316020	74ae30ac411ab6af
7	1d1cf61bf8a11b22	312e20cb829bf87e	e9a1b285e7627953	d88f924e65f9812d
8	43857842ff1135dc	a3375273a3c053e1	59b7bdfd39bef4b2	fa80ef8e9a7ea753
9	d19c0afade4a9e42	71588a212f3996d2	484583d5afe082ae	391d09f480d9147c
10	03f3d2a852b35b72	55a49cfe791faa82	e638bf5fd5a60619	b39c23a1acb9ac9b
11	270fc47704956723	136ae786bd52c72f	3901a08f4ab41aab	2a6b4709f7e6dd84
12	61c1bf0fc0d80a89	43e91125a2cf331e	9b134880	d8fa59a5

Figure 7. Encryption with UEA1 3GPP version

We notice that the CIPHERTEXT length is 800 bits and not 798 bits as it should be, regarding to the parameters LENGTH parameter value.

This means that the UEA1 algorithm didn't take into account the LENGTH parameter value given as input value, to be able to control the length of ciphertext resulting from the exclusive-OR operation between the plaintext and keystream.

We have carried out the necessary corrections to 3GPP algorithm code, as presented in the subsection 2.3.1 to make it able to discard the unnecessary bits from the last 64-bit keystream block. We find then the expected result shown in "Figure 8" bellow.

UEA1 EXECUTION

blkcnt	Kasumi input	KASUMI ---> KEYSTREAM	XOR PLAINTEXT	= CIPHERTEXT
0	72ab58897d8acdaa	6ea76b4fde974f21	7ec61272743bf161	1061793daaacbe40
1	1c0c33c6a31d828a	8e655a4347473a45	4726446a6c38ced1	c9431e292b7ff494
2	fcce02ca3acdf7ed	f02dc74725704108	66f6ca76eb543004	96db0d31ce24710c
3	82869fce58fa8ca1	432acb77c357103b	4286346cef130f92	01acff1b2c441fa9
4	318193febedddd95	a998be20d3229b0f	922b03450d3a9975	3bb3bd65de18027a
5	db33e6a9aea856a0	f1718bd14f7b269c	e5bd2ea0eb55ad8e	14cca571a42e8b12
6	83dad35832f1eb30	6fb7ae92852bd68f	1b199e3ec4316020	74ae30ac411ab6af
7	1d1cf61bf8a11b22	312e20cb829bf87e	e9a1b285e7627953	d88f924e65f9812d
8	43857842ff1135dc	a3375273a3c053e1	59b7bdfd39bef4b2	fa80ef8e9a7ea753
9	d19c0afade4a9e42	71588a212f3996d2	484583d5afe082ae	391d09f480d9147c
10	03f3d2a852b35b72	55a49cfe791faa82	e638bf5fd5a60619	b39c23a1acb9ac9b
11	270fc47704956723	136ae786bd52c72f	3901a08f4ab41aab	2a6b4709f7e6dd84
12	61c1bf0fc0d80a89	43e91125a2cf331e	9b134880	d8fa59a4

Figure 8. Encryption with UEA1 rectified version

2.3.3. UIA1 implementation example [1, 14]

As far as the UIA1 algorithm is concerned, the result of our verification shows that the algorithm is coded in respect of all 3GPP requirements, and the result of the implementation tests were as expected by the 3GPP implementation documents.

Bellow is presented an example of one of the testsets implemented.

<u>Input Data:</u>

Key = D42F6824 28201CAF CD9F9794 5E6DE7B7

Count = 3EDC87E2

Fresh = A4F2D8E2

Direction = 1

Length = 254 bits

Message:

B5924384328A4AE00B737109F8B6C8DD

2B4DB63DD533981CEB19AAD52A5B2BC0

UIA1 Algorithm - MAC-I CALCULATION

MAC-I Calculation steps

	Input	KASUMI Kasumi input --->	Kasumi Output	Input XOR Kasumi output Accumulated XOR
length >= 64				
	3edc87e2a4f2d8e2	3edc87e2a4f2d8e2	3541b47339dd4168	3541b47339dd4168
	b5924384328a4ae0	80d3f7f70b570b88	52ec81194ecedda0	67ad356a77139cc8
	0b737109f8b6c8dd	599ff010b678157d	792bfe1f07a1a8b0	1e86cb7570b23478
	2b4db63dd533981c	52664822d29230ac	c92f7e2c38d22b6d	d7a9b55949601f15
length < 64 Last 64-bit bloc processing:				
	eb19aad52a5b2bc3	2236d4f9128900ae	4c2bef9c82233403	9b825ac5ca432b16

LAST CALCULATION STEP :

Modified key: IK XOR KM = 7e85c28e828ab60567353d3ef4c74d1d

Kasumi input	KASUMI --->	Kasumi Output
9b825ac5ca432b16		a9daf1ff12f71de7

MAC-I = a9daf1ff

Figure 9. UIA1 MAC-I calculation

3. SNOW 3G BASED ALGORITHMS - UEA2/UIA2

After studying and implementing the first set of 3GPP cryptographic algorithms, we will focus now on the second algorithms set. These two new algorithms, based on the stream cipher SNOW 3G, has been introduced in the UMTS Access Network security since 2006 [7, 8].

Since we have already studied the different operation steps of both UEA2 and UIA2 algorithms in our previous work [15], we will be interested, in the present section, in the results of their implementations after having carried out the necessary corrections to the 3GPP algorithms codes to meet the 3GPP algorithms specifications. The verification and the implementation tasks for these two algorithms are more difficult then for the first set because the codes given by the 3GPP specification document are compatible only with big-endian machines, so they are not compatible with all UMTS equipments present in the market. Our task was then to propose an adaptation of the algorithm codes to little-endian machines and give the necessary corrections. More details about these issues are given in the following subsections.

3.1. UEA2 Verification And Implementation

In addition to the adaptation of the algorithm to little-endian machines, we have added, as for UEA1, the instructions to control the length of the generated ciphertext in accordance with the LENGTH parameter value. This issue was omitted in the 3GPP algorithm code version.

Bellow is the corrections we have done in the code to meet the 3GPP requirements and to adapt it to little endian machines.

1. First, there was a problem in loading the confidentiality key for SNOW 3G initialization. The memory unit is the byte. Reading a byte in little or big endian machines doesn't differ but the problem was found because the algorithm need to read a 32 bit-word (4 Bytes) from the memory. The instructions which were used in 3GPP documents are the following:

```
memcpy(K+3,key+0,4);
memcpy(K+2,key+4,4);
memcpy(K+1,key+8,4);
memcpy(K+0,key+12,4);
```

"key" is the cipher key given as an argument. And we assume that K[3]=key[0] || key[1] || … || key[31] and so on for K[2], K[1] and K[0] (K[0]=key[96] || key[97] || … || key[127]).

We have replaced these instructions by the following instruction block:

```
for (i=0;i<16;i++)
    memcpy(Kk+(15-i),key+i,1);
K[0] |= (u32) (Kk[3]<<24);
K[0] |= (u32) (Kk[2]<<16);
K[0] |= (u32) (Kk[1]<<8);
K[0] |= (u32) (Kk[0]);
K[1] |= (u32) (Kk[7]<<24);
K[1] |= (u32) (Kk[6]<<16);
K[1] |= (u32) (Kk[5]<<8);
K[1] |= (u32) (Kk[4]);
K[2] |= (u32) (Kk[11]<<24);
K[2] |= (u32) (Kk[10]<<16);
K[2] |= (u32) (Kk[9]<<8);
K[2] |= (u32) (Kk[8]);
K[3] |= (u32) (Kk[15]<<24);
K[3] |= (u32) (Kk[14]<<16);
K[3] |= (u32) (Kk[13]<<8);
K[3] |= (u32) (Kk[12]);
```

After this correction, the cipher key is read correctly by the algorithm codes.

2. The same problem was found with the following instructions:

```
for (i=0;i<n*4;i++)
    data[i] ^= *(((u8*)KS)+i);
```

These instructions are used to generate the ciphertext blocks from the plaintext and the keystream blocks.

We have replaced them by the following block of instructions. The control of the generated ciphertext length is done also in this step.

```
r = (length / 8);
s = 8 - (length - r * 8);
lengthtemp = length;
for (j=0;j<n;j++)
{
    i=0;
    while ((length>0) && (i<4))
    {
        KStemp.b32[0] = (u32) (*(KS+j));
        KStemp.b8[i] = (u8) (KStemp.b32[0] >> ((3-i)*8));
        if ((length < 32) \&\& ((length - i*8) < 8))
        {
```

```
                KStemp.b8[i] = KStemp.b8[i] >> s;
                KStemp.b8[i] = KStemp.b8[i] << s;
            }
            data[(j*4)+i] ^= KStemp.b8[i];
            i++;
        }
        length -=32;
    }
```

3.2. UIA2 verification and rectification

During the verification of the UIA2 algorithm, we were faced to the same endianness issues seen before with UEA2.

1. The first problem faced was with the MUL64 function. MUL64 works with 64-bit words and makes bit-shifting operations. These shifting instructions are understood differently in a big or little endian machines. So an adaptation was necessary.

   We then replace the following 3GPP function:

```
u64 MUL64(u64 V, u64 P, u64 c)
{
    u64 result = 0;
    int i = 0;
    for ( i=0; i<64; i++)
    {
        if( ( P>>i ) & 0x1 )
            result ^= MUL64xPOW(V,i,c);
    }
    return result;
}
```

   by the following function:

```
u64 MUL64(u64 V, u64 P, u64 c)
{
    u32 z[5],t[5];
    u64 result = 0;
    int i = 0;
    z[0] = (u32)P;
    z[1] = (u32)P >> 32;
    for ( i=0; i<64; i++)
    {
        if( ( (u64)P>>i ) & ((u64)0x1 << 32) )
            result ^= MUL64xPOW(V,i,c);
    }
    return result;
}
```

2. Then, to solve the message reading problem and to operate a 32-bit words internally, it was necessary to replace the following 3GPP instruction:

```
message = (u32*)data;
```

   by the instructions block bellow:

```
 l = length / 32;
 if (length % 8 ==0)

    for (i=0;i<(length / 8);i++)
        memcpy(m+i,data+i,1);
else
    for (i=0;i<(length / 8)+1;i++)
        memcpy(m+i,data+i,1);
for (j=0;j<l+1;j++)
{
    message[j] = (u32) m[j*4] << 24;
    for (i=1;i<4;i++)
        message[j] |= (u32) m[i + j*4] << (8*(4-i-1));
}
```

3.3. UEA2 implementation example

After adapting UEA2 algorithm code to read correctly from memory, and after making the necessary corrections, we were able to get the right ciphertexts for the textsets given by the implementation document [16]. We present here an example of the UEA2 implementation.

<u>Input Data:</u>

Count-C = E28BCF7B   Bearer = 18

Direction = 0   Length = 510 bits

CK = EFA8B2229E720C2A7C36EA55E9605695

Plaintext:

10111231E060253A43FD3F57E37607AB

2827B599B6B1BBDA37A8ABCC5A8C550D

1BFB2F494624FB50367FA36CE3BC68F1

1CF93B1510376B02130F812A9FA169D8

```
      UEA2 Confidentiality algorithm
   k0          k1          k2          k3
e9605695    7c36ea55    9e720c2a    efa8b222

  IV0         IV1         IV2         IV3
c0000000    e28bcf7b    c0000000    e28bcf7b

wordnumber     keystream           CipherText
    0      f0cb07fb 6e4571cf    e0da15ca 8e2554f5
    2      a691ab3f 3f1a7bb9    e56c9468 dc6c7c12
    4      b4713f3c b592ac3a    9c568aa5 032317e0
    6      79af82a8 3627baab    4e072964 6cabefa6
    8      927d6308 49000249    89864c41 0f24f919
    10      d0619e91 196b16a7   e61e3dfd fad77e56
    12      114992d8 26f421e6   0db0a9cd 36c34ae4
    14      0b1b1198 00fecb25   181490b2 9f5fa2fc
```

Figure 10. Encryption with UEA2 rectified version

3.4. UIA2 implementation example

After carrying out the necessary corrections and adaptations for the UIA2 algorithms, we present bellow an example of one of the testsets implemented.

Input Data:

COUNT-I = 14793E41   FRESH = 0397E8FD

DIRECTION = 1   LENGTH = 384 bits

IK = C736C6AAB22BFFF91E2698D2E22AD57E

MESSAGE:

D0A7D463DF9FB2B278833FA02E235AA1

72BD970C1473E12907FB648B6599AAA0

B24A038665422B20A499276A50427009

```
     UIA2 Algorithm - MAC-I CALCULATION

   k0          k1          k2          k3
e22ad57e    1e2698d2    b22bfff9    c736c6aa
  IV0         IV1         IV2         IV3
039768fd    94793e41    0397e8fd    14793e41

  z1        z2        z3        z4        z5
45898e82  8f27eb98  e3230709  a00cb70a  8f75ac4b

P = 45898e828f27eb98
Q = e3230709a00cb70a

 i          Mi                  EVAL
 0    d0a7d463df9fb2b2    9e80b47b98010914
 1    78833fa02e235aa1    5ea34890532d5ffb
 2    72bd970c1473e129    eae8e5595661fcf
 3    7fb648b6599aaa0    7ea00d6d65c8f93f
 4    b24a038665422b20    bec91666b07f7551
 5    a499276a50427009    689ef15153554dc2
 6    0000000000000180    689ef15153554c42

Multiplying by Q: EVAL = b7c0f88b24b5417c

           MAC-I = 38b554c0
```

Figure 11. UIA2 MAC-I calculation

4. CONCLUSION

In this paper, a detailed study of the UMTS cryptographic functions f8 and f9 has been carried out. The first objective was to make it easy to understand each UMTS confidentiality or integrity algorithms operation. On the other hand, we were interested in the verification of the 3GPP algorithms codes given by 3GPP specification. Our verification leads to the result that the algorithm codes were incorrect (except for UIA1) and didn't respond totally to the 3GPP requirements. So some rectifications and adaptations were necessary before proceeding to the algorithms implementation.

As far as the confidentiality algorithm UEA1 and UEA2 are concerned, the ciphertext length control have been omitted. So we have carried out the necessary correction. Concerning the UEA2 and UIA2 algorithms implementation, we were faced to the code compatibility problem with little endian-machines in both algorithms. So, we have corrected the 3GPP algorithms codes version and recoded them with respect to endian issues. Thus, the practical aspect of our present work was the verification, the rectification and finally the implementation of the UTRAN cryptographic algorithms in C language. Some results of the implementation tests were exposed. Through this paper, we are proposing a correct codes version of the UMTS confidentiality and integrity algorithms to contribute and facilitate their use and integration by the interested entities.

REFRENCES:

[1] 3GPP Specifications: http://www.3gpp.org

[2] 3GPP TS 33.102: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture". December 2006.

[3] Abdul Bais, Walter T. Penzhorn, Peter Palensky. "Evaluation of UMTS security architecture and services". IEEE International Conference on Industrial Informatics, 2006.

[4] 3GPP TS 21.133: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Threats and Requirements (Release 4)". December 2001.

[5] 3GPP TS35.201: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification". June 2007.

[6] 3GPP TS35.202: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification". june 2007.

[7] ETSI/SAGE Specification: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 1: UEA2 and UIA2 Specification". Version: 1.1, September 2006.

[8] ETSI/SAGE Specification: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification". Version: 1.1, September 2006.

[9] Ivo Pooters, University of Twente, Faculty of EEMCS: "An Approach to full User Data Integrity Protection in UMTS Access Networks". 04-2006.

[10] Kaisa Nyberg: "Cryptographic Algorithms for UMTS". Nokia Research Center, 2004.

[11] Valtteri Niemi and Kaisa Nyberg: "UMTS Security". John Wiley & Sons, 2003.

[12] Morris Dworkin: NIST Special Publication 800-38A, "Recommendation for Block Cipher Modes of Operation - Methods and Techniques". 2001 Edition.

[13] 3GPP TS35.203: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 3: Implementors' Test Data (Release 7)". June 2007.

[14] 3GPP TS35.204: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 4: Design Conformance Test Data (Release 7)". June 2007.

[15] G. Orhanou, S. El Hajji, Université Mohammed V – Agdal, Faculté des sciences – Rabat, Laboratoire Mathématiques Informatique et Applications : "UTRAN Cryptographic Algorithms – Operation and complexity study". Journal of Theoretical and Applied Information Technology, ISSN: 1817-3195, Volume 14, n°2, 97 – 106, 2010.

[16] ETSI/SAGE Specification: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2 Document 3: Implementors Test Data". Version: 1.0, January 2006.