Journal of Theoretical and Applied Information Technology

www.jatit.org

UTRAN CRYPTROGRAPHIC ALGORITHMS OPERATION AND COMPLEXITY STUDY

[1]Ghizlane ORHANOU, [2]Saïd EL HAJJI

Département Mathématique et Informatique, Laboratoire Mathématiques Informatique et Applications

Université Med V Agdal, Faculté des Sciences, Maroc

ABSTRACT

The Universal Mobile Telecommunications System (UMTS) offers panoply of 3G services that require a high protection of the transmitted information. In the present paper, we are interested by network access security, especially the protection of the data integrity and the provisioning of data encryption. A special interest will be given to the second set of 3GPP cryptographic algorithms UEA2/UIA2 based on the SNOW 3G algorithm. This set was, in fact, a subject only of few research works in comparison with the first one based on KASUMI algorithm. A closer look is taken at both the encryption algorithm UEA2 and the integrity algorithm UIA2 operations. The different operation modes of the algorithm SNOW 3G will be presented as well. A study of the time and space complexity of three algorithms has been carried out.

Keywords: *UMTS, Encryption, Integrity, SNOW 3G, UEA2, UIA2, Time complexity, Space complexity*

1. UMTS CONFIDENTIALITY AND INTEGRITY MECHANISMS

The confidentiality mechanism in the UMTS Access Network (UTRAN) is presented by the cryptographic function f8. It ensures the protection of user and signaling data over the air interface. On the other hand, the UTRAN integrity mechanism is based on the cryptographic function f9 which ensures the integrity protection of only signaling data, user data aren't integrity protected.

Both mechanisms are implemented in two UMTS network entities: ME (Mobile Equipment) which represents the end-user and RNC (Radio Network Controller) which is responsible of the Radio Access Network security [1, 2, 3].

1.1. UMTS Encryption Function f8

The user and signaling data confidentiality mechanism is based on the cryptographic function f8 [2] 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.

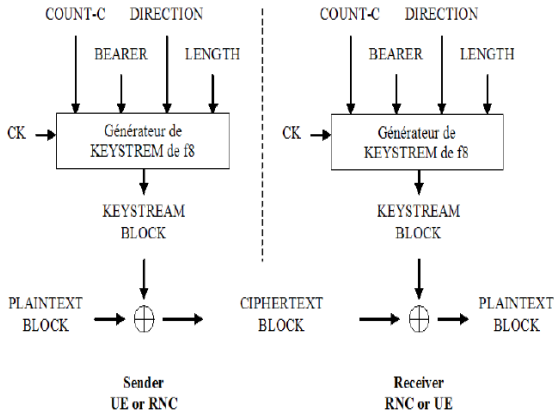

Figure 1. Encryption/Decryption mechanism

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 and UEA2. UEA1, which was used since the genesis of the UMTS network in 1999, is stream cipher based on KASUMI, a block cipher used under its OFB operation mode. 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.

1.2. UMTS Integrity Function f9

To ensure signaling data protection between the ME (Mobile Equipment) and the RNC (Radio Network Controller), a message authentication function f9 is used. It's a one-way function which generates a 32-bit output MAC-I under the control of 128-bit Integrity key IK [1, 3, 4].

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

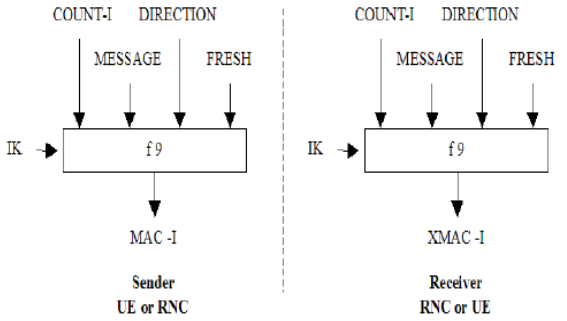

Figure 2. Derivation of MAC-I (or XMAC-I) [1, 2]

The algorithm input parameters are the following [2] :

- 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.

Like the algorithm UEA2, the integrity algorithm UIA2 is based on the stream cipher SNOW 3G, and was completely standardized on 2005.

In the present paper, we will present in the following section the SNOW 3G algorithm which constituted the heart of UEA2 and UIA2 algorithm. We'll be interested on its operation modes and its time and space complexity. After that, we will focus on studying the algorithms UEA2 and UIA2 operations and their complexities as well. Studying this later issue gives us a clear idea on the efficiency of the concerned algorithms.

2. SNOW 3G ALGORITHM

The algorithm SNOW 3G was chosen in 2006 as the cryptographic engine of the second set of UMTS confidentiality and Integrity algorithms f8 and f9 (UEA2 and UIA2) used over the air interface. The reason is that SNOW 3G respects properly the 3GPP requirements regarding the time and memory resources.

The stream cipher SNOW 3G is a two components stream cipher with an internal state of 608 bits initialized by a 128-bit key and a 128-bit initialization vector IV. In this section, we are interested in studying the algorithm SNOW 3G operation and efficiency.

2.1. SNOW 3G Algorithm Structure

SNOW 3G consists of two interacting modules, a Linear Feedback Shift Register (LFSR) and a Finite State Machine (FSM). The generated keystream is the result of the combined LFSR and FSM operations. Furthermore, the SNOW 3G security depends on the security of its two components.

The LFSR is constructed from 16 stages, each holding 32 bits and the feedback is defined by a primitive polynomial over the finite field $GF(2^{32})$.

As far as the FSM is concerned, it is based upon three 32-bit registers R1, R2 and R3 and uses two substitution box ensembles S_1 and S_2. The mixing operations are exclusive-OR and addition modulo 2^{32} [5, 6].

SNOW 3G is first initialized; during this step, the algorithm is clocked, so it is working without producing output. After being synchronized, it produces a sequence of 32-bit words under the control of 128-bit key. These words are then used to mask the different blocks of the plaintext in order to produce the ciphertext.

2.2. Initialization

SNOW 3G is initialized, as illustrated in "Figure 3" bellow, with a 128-bit key consisting of four 32-bit words k_0, k_1, k_2, k_3 and an 128-bit initialization variable consisting of four 32-bit words IV_0, IV_1, IV_2, IV_3 as follow [5]:

To simplify the formulas, we replace the 32-bit word 0xffffffff by 1.

s15 = k3 ⊕ IV0 s14= k2 s13 = k1

s12 = k0 ⊕ IV1 s11 = k3 ⊕ 1

s10= k2 ⊕ 1 ⊕ IV2 s9 = k1 ⊕ 1 ⊕ IV3

s8 = k0 ⊕ 1 s7 = k3 s6= k2

s5 = k1 s4 =k0 s3 = k3 ⊕ 1

s2= k2 ⊕ 1 s1 = k1 ⊕ 1 s0 =k0 ⊕ 1

The FSM is initialized with R1 = R2 = R3 = 0.

Then, the cipher runs in a special mode without producing output:

Repeat 32 times:

{

Step1: The FSM is clocked producing the 32-bit words F.

Step2: Then, the LFSR is clocked in Initialization Mode consuming F.

}

"Figure 3" bellow represents the Initialization Mode of SNOW 3G.

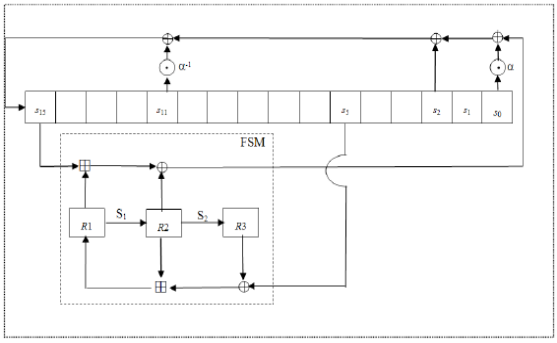

Figure 3. SNOW 3G Initialization mode [5]

2.3. Keystream Generation

After being clocked, SNOW 3G enters in its Keystream generation mode.

First, the FSM is clocked once and the FSM output word is discarded. Then, the LFSR is clocked in Keystream mode [5].

After that, n 32-bit words of keystream are produced [5]:

for t = 1 to n:

{

Step1: The FSM is clocked and produces a 32-bit output word F.

Step2: The next keystream word is calculated as follow: $z_t = F \oplus s0$.

Step3: Then, the LFSR is clocked in Keystream mode.

}

"Figure 4" illustrates SNOW 3G Keystream Mode.

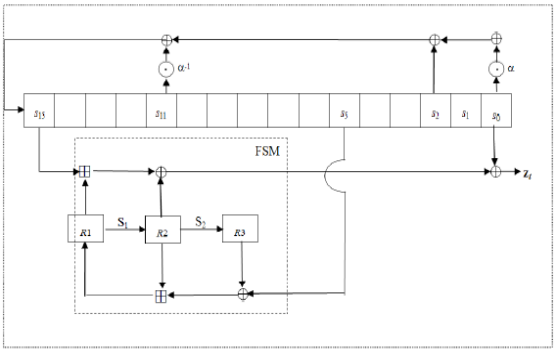

Figure 4. SNOW 3G Keystream Mode

2.4. Time And Space Complexity

As seen above, SNOW 3G consists of two interacting modules: an LFSR and a FSM. These two security modules LFSR and FSM use some functions and security components to accomplish their respective tasks.

In order to calculate the complexity, it is necessary to give a brief description of these components. More details can be found in 3GPP specification documents [1, 5].

2.4.1. SNOW 3G functions complexity

We begin by presenting the different functions used:

- MULx: The function MULx maps 16 bits to 8 bits. Let V and c be 8-bit input values, MULx is defined as follow:

  If the leftmost (i.e. the most significant) bit of V equals 1, then:

  $MULx(V, c) = (V <<_8 1) \oplus c$,

  else

  $MULx(V, c) = V <<_8 1$.

- MULxPOW: MULxPOW maps 16 bits and a positive integer i to 8 bit. MULxPOW(V, i, c) is recursively defined as follow:

  If i equals 0, then:

  $MULxPOW(V, i, c) = V$,

  else

MULxPOW(V, i, c) =
MULx(MULxPOW(V, i - 1,c), c).

- MULα: The function MULα maps 8 bits to 32 bits. It is defined as follow:

  MULα (c) = (MULxPOW(c, 23, 0xA9) || MULxPOW(c, 245, 0xA9) || MULxPOW(c, 48, 0xA9) || MULxPOW(c, 239, 0xA9)).

- DIVα: The function DIVα maps 8 bits to 32 bits. It is defined as follow:

  DIVα (c) = (MULxPOW(c, 16, 0xA9) || MULxPOW(c, 39, 0xA9) || MULxPOW(c, 6, 0xA9) || MULxPOW(c, 64, 0xA9)).

When calculating the time complexity of a certain function, if this function receives a finite number of input data, then its time complexity is O(1) and this is the case of most the functions used during SNOW 3G operation.

Propriety 1. ***The time complexity of the functions MULx, MULxPOW, MULα and DIVα is O(1).***

Since we are interested in calculating the amount of temporary storage used by SNOW 3G during its operation, a simple analysis of its functions exposed above, shows that they use a constant temporary storage that doesn't depend on the parameter "n" which represent the plaintext (and the keystream) length.

This leads to the following propierty:

Propriety 2. ***The space complexity of the functions MULx, MULxPOW, MULα and DIVα is O(1).***

2.4.2. SNOW 3G security components complexity

As far as the security components are concerned, the FSM uses two S-boxes S1 and S2 to update the registers R2 and R3 and that provides a strong diffusion of the input bits. Besides this, the FSM uses also two combining functions which are the bitwise exclusive-OR operation and the addition modulo 2^{32} [5, 6].

The substitution box S1, based on the bytewise substitution of Rijndael [7], was carefully selected to provide strong security. Combined with the good diffusion properties of the Mix-Column operation of Rijndael, the S1 construction offers good protection against many known attacks especially against differential and linear attacks [5, 6]. It maps a 32-bit input to a 32-bit output.

Moreover, in order to increase the resistance against algebraic attacks, the third memory element which is the 32-bit register R3 and the second ensemble of S-boxes S2 were introduced in the FSM-component of SNOW 3G.

The new S-box S2 consists of four new 8-bit to 8-bit substitutions followed by the MixColumn operation of Rijndael [5, 7]. The S-Box S2 maps a 32-bit input to a 32-bit output.

Furthermore, there is two combining functions used by the SNOW 3G FSM module.

- The linear function consists on the bitwise exclusive-OR operation.
- The non-linear combining function of SNOW 3G is the Integer addition modulo 2^{32}. This addition can be implemented in the SNOW3G algorithm as illustrated in the following example.

  Let X, Y and Z be three integers:

  $Z = X \boxplus Y$ means:
  $Z = (X + Y)$ & 0xffffffff
  with '+' is the normal integer addition and '&' is a bitwise AND operation.

Based on the definition of these components, given by the 3GPP specifications and their implementation, we have calculated their time and space complexity [5, 6].

Propriety 3. ***The time complexity of the S-Boxes S1 and S2 and of the addition modulo 2^{32} is O(1).***

Propriety 4. ***The space complexity of the S-Boxes S1 and S2 and of the addition modulo 2^{32} is O(1).***

2.4.3. SNOW 3G complexity

As we have seen before, SNOW 3G has two operation modes: Initialization Mode and Keystream Mode. These two modes are performed sequentially, the Initialization Mode, first, to initialize the components of the two principal SNOW 3G modules and then during the Keystream Mode there is the production of the 32-bit output words.

The number of words to produce as output depends on the input parameter n. This input parameter presents the length of the plaintext to cipher (or the ciphertext to decrypt). When given as input parameter to SNOW 3G, the algorithm produces exactly n 32-bit output words which will be added by an exclusive-OR operation to the n 32-

bit words of the plaintext (or ciphertext) to obtain the ciphertext (or the plaintext).

Based on what we have described before regarding the SNOW 3G operations, the Initialization Mode can be presented as illustrated in "Figure 5".

$s_{15} = \mathbf{k_3} \oplus \mathbf{IV_0}$ $s_{14} = \mathbf{k_2}$ $s_{13} = \mathbf{k_1}$
$s_{12} = \mathbf{k_0} \oplus \mathbf{IV_1}$ $s_{11} = \mathbf{k_3} \oplus \mathbf{1}$ $s_{10} = \mathbf{k_2} \oplus \mathbf{1} \oplus \mathbf{IV_2}$
$s_9 = \mathbf{k_1} \oplus \mathbf{1} \oplus \mathbf{IV_3}$ $s_8 = \mathbf{k_0} \oplus \mathbf{1}$ $s_7 = \mathbf{k_3}$
$s_6 = \mathbf{k_2}$ $s_5 = \mathbf{k_1}$ $s_4 = \mathbf{k_0}$
$s_3 = \mathbf{k_3} \oplus \mathbf{1}$ $s_2 = \mathbf{k_2} \oplus \mathbf{1}$ $s_1 = \mathbf{k_1} \oplus \mathbf{1}$
$s_0 = \mathbf{k_0} \oplus \mathbf{1}$

$R1 = R2 = R3 = 0$.

repeat 32-times {
STEP 1: The FSM is clocked producing the 32-bit word F.

$F = (s_{15} \boxplus R1) \oplus R2$
$r = R2 \boxplus (R3 \oplus s_5)$.
Set
$R3 = S_2(R2)$,
$R2 = S_1(R1)$,
$R1 = r$.

STEP 2: Then the LFSR is clocked in Initialisation Mode consuming F.

$v = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2$
$\oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F$.

$s_0 = s_1$, $s_1 = s_2$, $s_2 = s_3$, $s_3 = s_4$, $s_4 = s_5$, $s_5 = s_6$,
$s_6 = s_7$, $s_7 = s_8$, $s_8 = s_9$, $s_9 = s_{10}$, $s_{10} = s_{11}$, $s_{11} = s_{12}$,
$s_{12} = s_{13}$, $s_{13} = s_{14}$, $s_{14} = s_{15}$, $s_{15} = v$.
}

Figure 5. SNOW 3G Initialization Mode algorithm

We can see that this initialization stage is composed of some initialization assignments and a 32 time loop of a set of operations based on functions whose time complexity is O(1).

We will, then, be more interested in the second stage of SNOW 3G operation which consists on generating a keystream whose length is n. The Keystream Mode can be presented as illustrated in "Figure 6".

for $t = 1$ to n {
STEP 1: The FSM is clocked and produces a 32-bit output word F.

$F = (s_{15} \boxplus R1) \oplus R2$
$r = R2 \boxplus (R3 \oplus s_5)$.
Set
$R3 = S_2(R2)$,
$R2 = S_1(R1)$,
$R1 = r$.

STEP 2: The next keystream word is computed as $z_t = F \oplus s_0$.

STEP 3: Then the LFSR is clocked in Keystream Mode.

$v = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2$
$\oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_\alpha(s_{11,3})$.

$s_0 = s_1$, $s_1 = s_2$, $s_2 = s_3$, $s_3 = s_4$, $s_4 = s_5$, $s_5 = s_6$,
$s_6 = s_7$, $s_7 = s_8$, $s_8 = s_9$, $s_9 = s_{10}$, $s_{10} = s_{11}$, $s_{11} = s_{12}$,
$s_{12} = s_{13}$, $s_{13} = s_{14}$, $s_{14} = s_{15}$, $s_{15} = v$.
}

Figure 6. SNOW 3G Keystream Mode algorithm

We note that the Keystream Mode operation is based on an n loop of functions whose time complexity is O(1), which leads to a time complexity O(n).

Theorem 1. ***SNOW 3G algorithm has a linear time complexity TSNOW3G = O(n).***

On the other hand, whereas the number of steps taking by an algorithm during its execution is the primary measure of its efficiency, the amount of temporary storage used by the algorithm is also a major concern. Furthermore, in some settings, space is even more scarce than time.

So, as far as SNOW 3G space complexity is concerned, and after calculating the space complexity of the different functions and security components of the stream cipher algorithm, we can conclude that the space complexity of SNOW 3G in its two modes, initialization and keystream modes is O(1), which leads to the theorem bellow.

Theorem 2. ***SNOW3G algorithm has a constant space complexity.***

3. ENCRYPTION ALGORITHM UEA2

After studying the stream cipher SNOW 3G, we will focus now on the confidentiality algorithm UEA2 which was introduced the UMTS Access Network security since 2006.

As for SNOW 3G, We will be interested in UEA2 operation and then in its time and space complexity.

3.1. UEA2 Algorithm Structure

The confidentiality algorithm UEA2 is a stream cipher which encrypt/decrypt data blocks (between 1 and 20.000 bits) under the 128-bit cipher key CK.

It is based on the synchronous word-oriented stream cipher SNOW 3G who produces 32-bit words as seen above. These words are added by an exclusive-OR (XOR) operation to the input stream (the plaintext) to produce the ciphertext [1, 8].

The decryption operation is done in the same way as the encryption. It is important to note that the sender and the receiver are synchronized with each other thanks to the input parameter COUNT-C.

UEA2 operation is done in three principal steps, keystream generator initialization, keystream generation and encryption/decryption.

"Figure 7" bellow shows globally UEA2 operation.

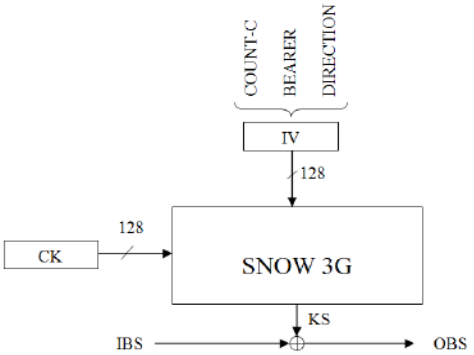

Figure 7. UEA2 Encryption/Decryption mechanism

3.2. Initialization And Keystream Generation

The UEA2 algorithm structure, based on SNOW 3G, is presented in the subsections bellow.

3.2.1. Initialization

All the used variables are 32-bit length and are presented with the most significant bit on the left hand side. They are initialized with the input parameters.

K3 = CK[0] || CK[1] || CK[2] || … || CK[31]

K2 = CK[32] || CK[33] || CK[34] || … || CK[63]

K1 = CK[64] || CK[65] || CK[66] || … || CK[95]

K0 = CK[96] || CK[97] || CK[98] || … || CK[127]

IV3 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || … || COUNT-C[31]

IV2 = BEARER[0] || BEARER[1] || … || BEARER[4] || DIRECTION[0] || 0 || … || 0

IV1 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || … || COUNT-C[31]

IV0 = BEARER[0] || BEARER[1] || … || BEARER[4] || DIRECTION[0] || 0 || … || 0

3.2.2. Keystream generation

We set $L = \lceil LENGTH / 32 \rceil$.

UEA2 uses SNOW 3G to produce the 32-bit keystream words z_1 … z_L. z_1 is the first word produced and so on.

The sequence of keystream bits is KS[0] … KS[LENGTH-1], where KS[0] is the most significant bit of z_1, and KS[31] is the least significant one and so on.

"Figure 8" bellow presents a summary of the two first steps, initialization and keystream generation.

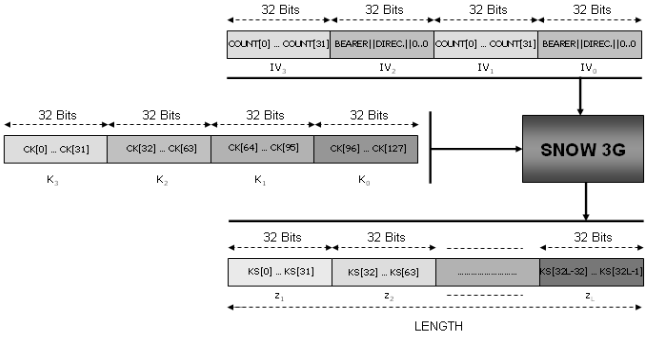

Figure 8. UEA2 Keystream generator

3.3. Encryption / Decryption

UEA2 Encryption and Decryption operations are identical and are performed by the exclusive-OR operation (XOR) of the input data which represents the message to encrypt/decrypt with the generated keystream (KS).

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

$$ciphertext[i] = plaintext[i] \oplus KS[i]$$

3.4. UEA2 Time And Space Complexity

In order to calculate the UEA2 time and space complexity, we propose to reassemble all its operation steps in the following algorithm illustrated in "Figure 9". Thus, we can easily see all the instructions used and estimate also the memory space used during the algorithm operation.

UEA2 Initialization:

K3 = CK[0] || CK[1] || CK[2] || ... || CK[31]

K2 = CK[32] || CK[33] || CK[34] || ... || CK[63]

K1 = CK[64] || CK[65] || CK[66] || ... || CK[95]

K0 = CK[96] || CK[97] || CK[98] || ... || CK[127]

IV3 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || ... || COUNT-C[31]

IV2 = BEARER[0] || BEARER[1] || ... || BEARER[4] || DIRECTION[0] || 0 || ... || 0

IV1 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || ... || COUNT-C[31]

IV0 = BEARER[0] || BEARER[1] || ... || BEARER[4] || DIRECTION[0] || 0 || ... || 0

KEYSTREAM Generation:

We set L = ⌈LENGTH / 32⌉

Initialize_Snow3G (K, IV);

/* Allocate memory space for the generated KEYSTREAM KS */

KS = (u32 *) malloc (4*L);

GenerateKeystream_Snow3G(L, *KS);

Encryption:

For 0 ≤ i ≤ LENGTH-1:

ciphertext[i] = plaintext[i] ⊕ KS[i]

Figure 9. UEA2 algorithm

The UEA2 is based on SNOW 3G algorithm which has a linear time complexity. Besides this, in the last step which concern the encryption (eventually decryption), the number of exclusive-OR operation execution depends, like SNOW 3G Keystream generation, on the length of the text to encrypt or decrypt. These two issues lead to the fact that the UEA2 time complexity is O(n).

Theorem 3. ***UEA2 algorithm has a linear time complexity.***

On the other hand, during its operation, UEA2 needs to allocate a memory space whose size is 4*n for the generated keystream. In addition, SNOW 3G used by UEA2 has a constant space complexity. So, the UEA2 space complexity is then O(n).

Theorem 4. ***UEA2 algorithm has a linear space complexity.***

4. INTEGRITY ALGORITHM UIA2

Like UEA2, UIA2 is also based on the stream cipher SNOW 3G, but uses in addition to the kernel algorithm, some functions for later processing of the keystream generated by SNOW 3G [1, 8].

4.1. UEA2 Algorithm Structure

The message authentication function which was chosen for the new integrity algorithm UIA2 is based on a universal hash functions and is similar to Galois Message Authentication Code (GMAC) [9]. The final algorithm design combines the standard GMAC operation with a last processing step in order to generate a high level 32-bit MAC-I security.

"Figure 10" illustrates the global UIA2 algorithm operation, which will be detailed in the following subsections.

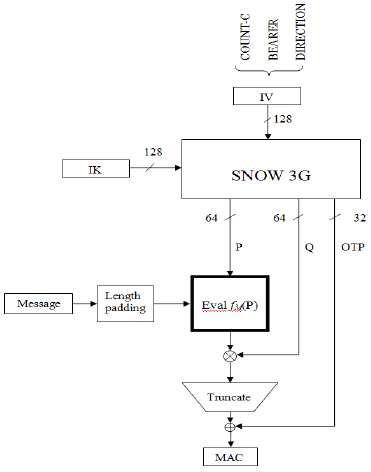

Figure 10. UIA2 algorithm operation [6]

4.2. Initialization And MAC-I Calculation

4.2.1. Initialization

Before processing the signaling message, SNOW 3G generator is initialized with the integrity key IK and the initialization vector IV as follow:

K3 = IK[0] || IK[1] || IK[2] || ... || IK[31]

K2 = IK[32] || IK[33] || IK[34] || … || IK[63]

K1 = IK[64] || IK[65] || IK[66] || … || IK[95]

K0 = IK[96] || IK[97] || IK[98] || … || IK[127]

IV3 = COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || … || COUNT- I[31]

IV2 = FRESH[0] || FRESH[1] || FRESH[2] || … || FRESH[31]

IV1 = DIRECTION[0] ⊕ COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || … || COUNT-I[31]

IV0 = FRESH[0] || FRESH[1] || … || FRESH[15] || FRESH[16] ⊕ DIRECTION[0] || FRESH[17] || … || FRESH[31]

Three random values are then generated: two 64-bit values P and Q, and 32-bit value OTP (One-Time-Pad) [1, 8]. "Figure 11" shows how SNOW 3G is initialized to produce P, Q and OTP.

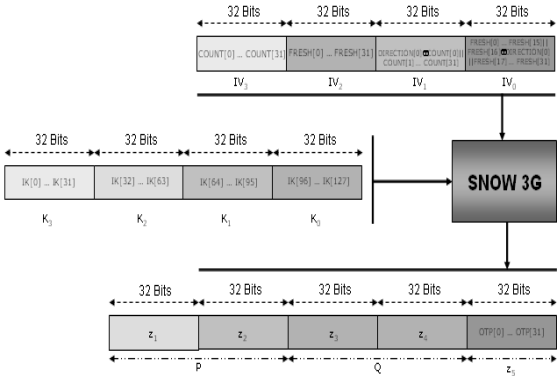

Figure 11. UIA2 algorithm 1st processing part

4.2.2. MAC-I calculation

MAC-I calculation is performed in two steps.

At the beginning of UIA2 operation, SNOW 3G is used to produce five 32-bit keystream words z_1, z_2, z_3, z_4, z_5.

We set P = z_1 || z_2 et Q = z_3 || z_4.

"Figure 11" above illustrates this.

After that, begins the second step of the UIA2 operation where many operations are performed to produce the MAC-I (eventually the XMAC-I [1, 2]).

We set D=⌈ LENGTH/64 ⌉+1

For $0 \le i \le D - 3$, we set:

M_i = MESSAGE[64i] || MESSAGE[64i+1] || ... || MESSAGE[64i+63]

and

M_{D-2} = MESSAGE[64(D-2)] || … || MESSAGE[LENGTH-1] || 0 … 0.

We note LENGTH[0], LENGTH[1], … , LENGTH[63] the 64 bits of the LENGTH parameter.

We set then:

M_{D-1} = LENGTH[0] || LENGTH[1] || … || LENGTH[63].

After having defined input parameters of the different functions of UIA2, we will explicit, through "Figure 12" bellow, the different operation steps, the input and the output of each step until obtaining MAC-I:

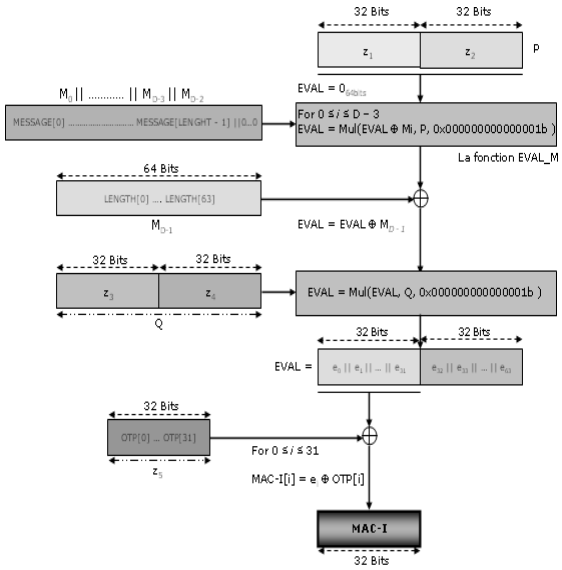

Figure 12. UIA2 algorithm 2nd processing part

The MAC-I is the result of an exclusive-OR operation between the EVAL 32-bit left half (the most significant 32 bits) and the 32-bit word z_5:

For $0 \le i \le 31$, we have:

$$MAC\text{-}I[i] = e_i \oplus OTP[i]$$

The bits $e_{32}, \dots, e_{63}$ are discarded.

4.3. UIA2 Time And Space Complexity

Unlike UEA2, UIA2 uses, in addition to SNOW 3G as a kernel, other functions are used to make the generated MAC-I as stronger as possible.

So before calculating UIA2 complexity, we will, first, introduce these functions and calculate their time and space complexity.

4.3.1. UIA2 functions complexity

We begin by presenting the different functions used:

- MUL64x: This function maps 128 bits to 64 bits. It has the same definition as the MULx function used by SNOW 3G and defined in subsection 2.4.1. of the present paper. The only difference is that the input V and c for MUL64x are 64-bit instead of 8-bit and then the output is 64-bit as well.
- MUL64xPOW: This function maps 128 bits and a positive integer i to 64-bit. It is equivalent to MULxPOW defined for SNOW 3G except for the input and output size (64 bits instead of 8 bits).
- MUL64: The function MUL64 maps 192 bits to 64 bits. Let V, P and c be 64-bit input values, MUL64 is defined as follow:

```
for 0 ≤ i < 64
{
   If the least significant bit of (P>>i)
   equals 1, then:
      result = result ⊕ MUL64xPOW(V, i, c);
}
```

- mask32bit: This function prepares a 32 bit mask with required number of 1 bits on the MSB (most significant bit) side. The input n is an integer in 1-32.

```
If n (mod 32) = 0 , then:
   return 0xffffffff;
Tant que n-1 ≠ 0, then:
   mask = (mask>>1) ⊕ 0x80000000;
return mask ;
```

So, from these definitions, we can see that all the functions above receive a finite number of input data, so their time complexity is O(1). Furthermore, they take a fixed amount of memory during their operations, so the space complexity is also O(1). These results lead to the following proprieties.

Propriety 5. *The functions MUL64x, MUL64xPOW, MUL64 and mask32bit have a constant time complexity.*

Propriety 6. ***The functions MUL64x, MUL64xPOW, MUL64 and mask32bit have a constant space complexity.***

4.3.2. UIA2 algorithm complexity

After calculating the time and space complexity of the different parts constituting the UIA2 algorithm, we will be interested, now, in UIA2 complexity itself.

"Figure 13" bellow gives a general overview of the UIA2 operation.

UIA2 Initialization :
K3 = IK[0] || IK[1] || IK[2] || ... || IK[31];
K2 = IK[32] || IK[33] || IK[34] || ... || IK[63];
K1 = IK[64] || IK[65] || IK[66] || ... || IK[95];
K0 = IK[96] || IK[97] || IK[98] || ... || IK[127];
IV3 = COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || ... || COUNT- I[31];
IV2 = FRESH[0] || FRESH[1] || FRESH[2] || ... || FRESH[31];
IV1 = DIRECTION[0] ⊕ COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || ...
|| COUNT-I[31];
IV0 = FRESH[0] || FRESH[1] || ... || FRESH[15] ||
FRESH[16] ⊕ DIRECTION[0] || FRESH[17] || ... || FRESH[31];
z1 = z2 = z3 = z4 = z5 = 0;

Generation of 5 Keystream 32-bit words :
Initialize_Snow3G (K, IV);
GenerateKeystream_Snow3G(5, z);
P = z1|| z2;
Q = z3 || z4;

MAC-I Calculation:

We set D = ⌈LENGTH / 64⌉ + 1;
EVAL = 0;
c = 0x1b;
for 0 ≤ i ≤ D-3:
{
V = EVAL ⊕ (64-bit message[2*i] || 64-bit message[2*i + 1]);
EVAL = MUL64 (V, P, c);
}
remaining_bits = LENGTH (mod 64);
If remaining_bits is equal 0, then:
We put remaining_bits = 64;
mask = mask32bit(remaining_bits (mod 32));
if (remaining_bits > 32)
M_D_2 = 64-bit message[2* (D-2)] || (64-bit message[2* (D-2) + 1] AND mask);
else
M_D_2 = 64-bit message[2* (D-2)] & mask;
V = EVAL ⊕ M_D_2 ;
EVAL = MUL64 (V, P, c) ;
EVAL = EVAL ⊕ LENGTH ;
EVAL = MUL64 (EVAL, Q, c) ;

MAC-I = (32-bit MBS (EVAL)) ⊕ z5 ;

Figure 13. UIA2 Algorithm

Indeed, the first and second step of UIA2 operation which consist on the algorithm initialization and the generation of 5 keystream 32-bit words, don't depend on the message length and has a finite number of instructions. Furthermore, the amount of memory used in this part of UIA2 is fixed. So the time and space complexity of this part of the algorithm is O(1).

For the MAC-I calculation step, UIA2 needs to calculate the V and EVAL values which depend on the whole concerned message. So, the V and EVAL calculating instructions depend on the message length. So, we can conclude that the time complexity of UIA2 algorithm is O(n).

Theorem 5. ***UIA2 algorithm has a linear time complexity.***

On the other hand, as we can notice that, in the UIA2 algorithm, the amount of temporary storage used doesn't depend on the message length and the same parameters are used to store the result of all the performed operations. So, UIA2 has a constant space complexity.

Theorem 6. ***UIA2 algorithm has a constant space complexity.***

5. CONCLUSION

In this paper, a detailed study of the confidentiality and integrity algorithms UEA2 and UIA2 based on the stream cipher SNOW 3G has been carried out. A study of their heart algorithm has been accomplished as well. The objective is to understand enough UEA2, UIA2 and SNOW 3G operations and to study their time and space complexity. We have found that three algorithms have a linear time complexity, which guarantee efficiency and rapidity during the encryption and the decryption process.

Furthermore, UEA2 has linear space complexity while SNOW 3G and UIA2 have a constant space complexity. The two later algorithms consume, then, a constant and already known amount of temporary memory which is very useful for systems with small working memory such as mobile equipments. The linear space complexity of the encryption algorithm is also a good result since the maximum length that a message can have is already fixed by 3GPP specifications and the Mobile Equipments can easily handle this amount of memory.

Finally, from the study of the time and space complexity of UEA2 an UIA2 especially, we can conclude that these cryptographic algorithms were well chosen to be the second set of the confidentiality and integrity algorithms (UEA2/UIA2) of the 3rd generation of mobile Telecommunications since 2006, and also, for the same efficiency reasons, have been kept as the first set of security algorithms (EEA1/EIA1) for LTE (Long Term Evolution).

6. FUTURE WORK

After studying the operation and complexity of the two UTRAN cryptographic algorithms UEA2 and UIA2, we will be interested in our future work in the implementation of these algorithms. In fact, 3GPP specification documents give the algorithm codes. Our task will be the verification of 3GPP algorithms code versions, their adaption to meet the 3GPP requirements and finally their implementation. The Testsets given by 3GPP will help us to ensure the correctness of our propositions.

REFRENCES:

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

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

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

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

[5] ETSI/SAGE Specification: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification". Version: 1.1. Date: 6th September 2006.

[6] ETSI/SAGE Technical report: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 5: Design and Evaluation Report". Version: 1.1. Date: 6th September 2006.

[7] FIPS Publication 197, Advanced Encryption Standard (AES). U.S. DoC/NIST, November 26, 2001.

[8] ETSI/SAGE Specification: "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 1: UEA2 and UIA2 Specification". Version: 1.1. Date: 6th September 2006.

[9] David McGrew, Cisco Systems, Inc., USA, "Efficient authentication of large, dynamic data sets using Galois/Counter Mode (GCM)". Proceedings of the Third IEEE International Security in Storage Workshop, Pages: 89 – 94, ISBN:0-7695-2537-7, 2005.