Contemporary Engineering Sciences, Vol. 3, 2010, no. 3, 97 - 111

SNOW 3G Stream Cipher Operation and Complexity Study

Ghizlane ORHANOU

ghizlane.orhanou@gmail.com

Said EL HAJJI

elhajji@fsr.ac.ma

Youssef BENTALEB

youssef_bentaleb2003@yahoo.fr

Laboratoire Mathematiques, Informatique et Applications,
Universite Mohammed V Agdal, Faculté des Sciences
BP 1014, Rabat, Maroc

Abstract

SNOW 3G is a stream cipher algorithm that had been conceived and chosen in 2006 as the heart of the second set of UMTS confidentiality and integrity algorithms. It has been kept as the engine of the first set of LTE cryptographic algorithms as well. The reason is that SNOW 3G respects properly the 3GPP requirements regarding the time and memory resources. This stream cipher 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 the present paper, we will interested first in SNOW 3G algorithm structure, its components and then its two different operation modes. After that, we will focus on its efficiency by studying its time and space complexity.

Keywords: SNOW 3G, stream cipher, LFSR, Encryption, Time complexity, Space complexity, LTE, 3GPP

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 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}).
The FSM 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} [2, 3].
The two security modules LFSR and FSM, the functions and the security components that they use to accomplish their tasks will be presented in the following subsections [2].

1.1 SNOW 3G Functions

The following mathematical symbols will be used bellow:

$=$	The assignment operator.
$\oplus$	The bitwise exclusive-OR operation.
$\boxplus$	Integer addition modulo 2^{32}.
$\|\|$	The concatenation of the two operands.
$<<_n$ t	t-bit left shift in an n-bit register.

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

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

  $MUL_\alpha(c) = (MULxPOW(c, 23, 0xA9) \,||\, MULxPOW(c, 245, 0xA9) \,||\, MULxPOW(c, 48, 0xA9) \,||\, MULxPOW(c, 239, 0xA9)).$

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

  $DIV_\alpha(c) = (MULxPOW(c, 16, 0xA9) \,||\, MULxPOW(c, 39, 0xA9) \,||\, MULxPOW(c, 6, 0xA9) \,||\, MULxPOW(c, 64, 0xA9)).$

1.2 LFSR Component

The Linear Feedback Shift Register (LFSR) consists of 16 stages $s_0, s_1, ..., s_{15}$, each holding 32 bits.

1.2.1 General LFSR definition

An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Generally, some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. The only linear functions of single bits are xor and inverse-xor; thus it is a shift register whose input bit is driven by the exclusive-or (xor) of some bits of the overall shift register value 1.

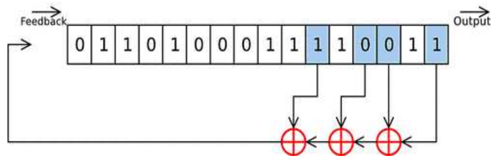


Figure 1: Linear Feedback Shift Register

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.
As far as SNOW 3G is concerned, the feedback loop involves two multiplications that can be implemented as a byte shift together with an unconditional XOR with one of 256 possible patterns. This results in a better spreading of the bits in the feedback loop and improves the resistance against correlation attacks. The use of two constants in the feedback loops also improves resistance against bitwise linear approximation attacks [3].

1.2.2 Clocking the LFSR in the SNOW 3G algorithm

The clocking of the LFSR has two different modes of operation, the Initialisation Mode and the Keystream Mode. In both modes the functions MUL_α and DIV_α, defined above, are used in the feedback loop.

1. **Initialization Mode:**

   In the Initialization Mode, the LFSR receives a 32-bit input word F, which is the input of the FSM.

   Let $s_0 = s_{0,0} \parallel s_{0,1} \parallel s_{0,2} \parallel s_{0,3}$.

   Let $s_{11} = s_{11,0} \parallel s_{11,1} \parallel s_{11,2} \parallel s_{11,3}$.

An intermediate value v is calculate as follow:

$\boldsymbol{v} = (\mathbf{s}_{0,1} \parallel \mathbf{s}_{0,2} \parallel \mathbf{s}_{0,3} \parallel \mathbf{0x00}) \oplus \mathbf{MUL}_\alpha(\mathbf{s}_{0,0}) \oplus \mathbf{s2} \oplus (\mathbf{0x00} \parallel \mathbf{s}_{11,0} \parallel \mathbf{s}_{11,1} \parallel \mathbf{s}_{11,2}) \oplus \mathbf{DIV}_\alpha(\mathbf{s}_{11,3}) \oplus \mathbf{F}.$

Then, the different stages of the LFSR are synchronized as presented bellow:

$\mathbf{s}_0 = \mathbf{s}_1$, $\mathbf{s}_1 = \mathbf{s}_2$, $\mathbf{s}_2 = \mathbf{s}_3$, $\mathbf{s}_3 = \mathbf{s}_4$, $\mathbf{s}_4 = \mathbf{s}_5$, $\mathbf{s}_5 = \mathbf{s}_6$, $\mathbf{s}_6 = \mathbf{s}_7$, $\mathbf{s}_7 = \mathbf{s}_8$,

$\mathbf{s}_8 = \mathbf{s}_9$, $\mathbf{s}_9 = \mathbf{s}_{10}$, $\mathbf{s}_{10} = \mathbf{s}_{11}$, $\mathbf{s}_{11} = \mathbf{s}_{12}$, $\mathbf{s}_{12} = \mathbf{s}_{13}$, $\mathbf{s}_{13} = \mathbf{s}_{14}$, $\mathbf{s}_{14} = \mathbf{s}_{15}$, $\mathbf{s}_{15} = \boldsymbol{v}$.

Figure 2 bellow illustrates the LFSR Initialization Mode at time t and t+1:

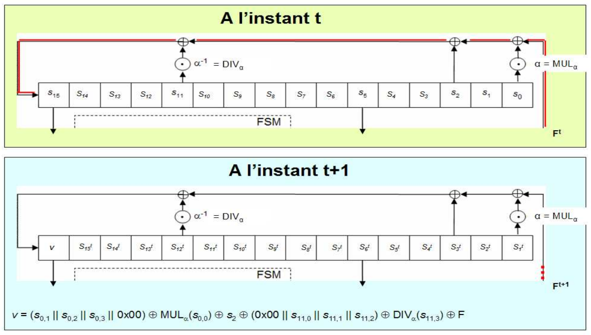

A l'instant t
α^{-1} = DIV$_\alpha$
α = MUL$_\alpha$
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0
FSM
F^t
A l'instant t+1
α^{-1} = DIV$_\alpha$
α = MUL$_\alpha$
v s_{15}^t s_{14}^t s_{13}^t s_{12}^t s_{11}^t s_{10}^t s_9^t s_8^t s_7^t s_6^t s_5^t s_4^t s_3^t s_2^t s_1^t
FSM
F^{t+1}
$v = (s_{0,1} \parallel s_{0,2} \parallel s_{0,3} \parallel 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \parallel s_{11,0} \parallel s_{11,1} \parallel s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F$

Figure 2: LFSR Initialization Mode

2. **Keystream Mode:**

   In the Keystream Mode, the LFSR does not receive any input.

   Let $s_0 = s_{0,0} \parallel s_{0,1} \parallel s_{0,2} \parallel s_{0,3}$.

   Let $s_{11} = s_{11,0} \parallel s_{11,1} \parallel s_{11,2} \parallel s_{11,3}$.

   An intermediate value v is calculate as follow:

   $\boldsymbol{v} = (\mathbf{s}_{0,1} \parallel \mathbf{s}_{0,2} \parallel \mathbf{s}_{0,3} \parallel \mathbf{0x00}) \oplus \mathbf{MUL}_\alpha(\mathbf{s}_{0,0}) \oplus \mathbf{s2} \oplus (\mathbf{0x00} \parallel \mathbf{s}_{11,0} \parallel \mathbf{s}_{11,1} \parallel \mathbf{s}_{11,2}) \oplus \mathbf{DIV}_\alpha(\mathbf{s}_{11,3})$.

   Then, the different stages of the LFSR are synchronized as presented bellow:

   $\mathbf{s}_0 = \mathbf{s}_1$, $\mathbf{s}_1 = \mathbf{s}_2$, $\mathbf{s}_2 = \mathbf{s}_3$, $\mathbf{s}_3 = \mathbf{s}_4$, $\mathbf{s}_4 = \mathbf{s}_5$, $\mathbf{s}_5 = \mathbf{s}_6$, $\mathbf{s}_6 = \mathbf{s}_7$, $\mathbf{s}_7 = \mathbf{s}_8$,

   $\mathbf{s}_8 = \mathbf{s}_9$, $\mathbf{s}_9 = \mathbf{s}_{10}$, $\mathbf{s}_{10} = \mathbf{s}_{11}$, $\mathbf{s}_{11} = \mathbf{s}_{12}$, $\mathbf{s}_{12} = \mathbf{s}_{13}$, $\mathbf{s}_{13} = \mathbf{s}_{14}$, $\mathbf{s}_{14} = \mathbf{s}_{15}$, $\mathbf{s}_{15} = \boldsymbol{v}$.

   Figure 3 bellow illustrates the LFSR Keystream Mode at time t and t+1:

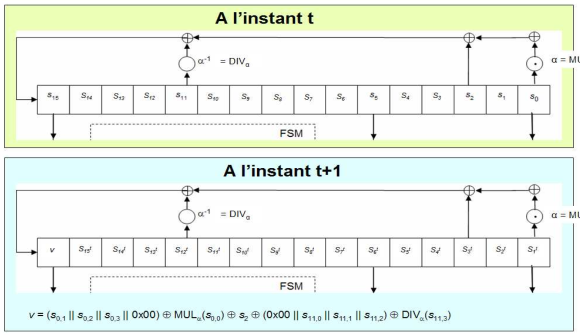

A l'instant t
α^{-1} = DIV_α
α = MUL_α
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0
FSM
A l'instant t+1
α^{-1} = DIV_α
α = MUL_α
v s_{15}^t s_{14}^t s_{13}^t s_{12}^t s_{11}^t s_{10}^t s_9^t s_8^t s_7^t s_6^t s_5^t s_4^t s_3^t s_2^t s_1^t
FSM
$v = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_\alpha(s_{11,3})$

Figure 3: LFSR Keystream Mode

1.3 FSM Component

The Finite State Machine (FSM) constitute the nonlinear combiner of SNOW 3G.It has three 32-bit registers R1, R2, et R3.
During its operation, the FSM involves two input data from the LFSR, s_5 and s_{15} stages contents [3]. The introduction of two inputs to the FSM part makes a guess-and-determine attack more difficult.

The FSM uses two S-boxes S_1 and S_2 to update the registers R2 and R3 and that provides stronger diffusion since each output bit depends on each input bit thanks to the use of the S-Box component.
The combining functions used are the bitwise exclusive-OR operation and the addition modulo 2^{32}.

1.3.1 Substitution boxes (S-boxes)

1. **S-Box S1 32x32-bit**:

   In SNOW 3G, the substitution box S1, based on the bytewise substitution of Rijndael, was carefully selected to provide strong security. Combined with the good diffusion properties of the MixColumn operation of Rijndael, the S1 construction offers good protection against many known attacks especially against differential and linear attacks [3].

   The S-Box S_1 maps a 32-bit input to a 32-bit output [2].

   Let w = w0 $\|$ w1 $\|$ w2 $\|$ w3

   Let S1(w)= r0 $\|$ r1 $\|$ r2 $\|$ r3. We use the 8 to 8 bit Rijindael S-Box S_R.

   Then r_0, r_1, r_2, r_3 are defined as follow:

   $\mathbf{r0} = \mathbf{MULx}(\mathbf{S}_R(\mathbf{w}_0), \mathbf{0x1B}) \oplus \mathbf{S}_R(\mathbf{w}_1) \oplus \mathbf{S}_R(\mathbf{w}_2) \oplus \mathbf{MULx}(\mathbf{S}_R(\mathbf{w}_3),\mathbf{0x1B}) \oplus \mathbf{S}_R(\mathbf{w}_3)$,

r_1= MULx($S_R(w_0)$,0x1B) $\oplus$ $S_R(w_0)$ $\oplus$ MULx($S_R(w_1)$,0x1B) $\oplus$ $S_R(w_2)$ $\oplus$ $S_R(w_3)$,

r_2= $S_R(w_0)$ $\oplus$ MULx($S_R(w_1)$,0x1B) $\oplus$ $S_R(w_1)$ $\oplus$ MULx($S_R(w_2)$,0x1B) $\oplus$ $S_R(w_3)$,

r_3= $S_R(w_0)$ $\oplus$ $S_R(w_1)$ $\oplus$ MULx($S_R(w_2)$,0x1B) $\oplus$ MULx($S_R(w_3)$,0x1B) $\oplus$ $S_R(w_2)$.

2. **S-Box S2 32x32-bit**:

   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 S_2 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 [3]. The S-Box S_2 maps a 32-bit input to a 32-bit output.

   Let $w = w_0 \parallel w_1 \parallel w_2 \parallel w_3$, Let S2(w)= $r_0 \parallel r_1 \parallel r_2 \parallel r_3$.

   Then r0, r1, r2, r3 are defined as follow:

   r0= MULx($S_Q(w_0)$, 0x69) $\oplus$ $S_Q(w_1)$ $\oplus$ $S_Q(w_2)$ $\oplus$ MULx($S_Q(w_3)$,0x69) $\oplus$ $S_Q(w_3)$,

   r_1= MULx($S_Q(w_0)$,0x69) $\oplus$ $S_Q(w_0)$ $\oplus$ MULx($S_Q(w_1)$,0x69) $\oplus$ $S_Q(w_2)$ $\oplus$ $S_Q(w_3)$,

   r_2= $S_Q(w_0)$ $\oplus$ MULx($S_Q(w_1)$,0x69) $\oplus$ $S_Q(w_1)$ $\oplus$ MULx($S_Q(w_2)$,0x69) $\oplus$ $S_Q(w_3)$,

   r_3= $S_Q(w_0)$ $\oplus$ $S_Q(w_1)$ $\oplus$ MULx($S_Q(w_2)$,0x69) $\oplus$ MULx($S_Q(w_3)$,0x69) $\oplus$ $S_Q(w_2)$.

1.3.2 Combining functions

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 SNOW 3G 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.

1.3.3 Clocking the FSM

The FSM has two input words s_{15} and s_5 from the LFSR. It produces a 32-bit output word F:

$$\mathbf{F} = (\mathbf{s}_{15} \boxplus \mathbf{R1}) \oplus \mathbf{R2}$$

Then the registers are updated. We compute the intermediate value r as follow:

$$r = \mathrm{R2} \boxplus (\mathrm{R3} \oplus s_5).$$

Then we set:

$$\mathbf{R3} = \mathbf{S}_2(\mathbf{R2}),$$
$$\mathbf{R2} = \mathbf{S}_1(\mathbf{R1}),$$
$$\mathbf{R1} = \boldsymbol{r}.$$

Figure 4 illustrates the FSM Module clocking:

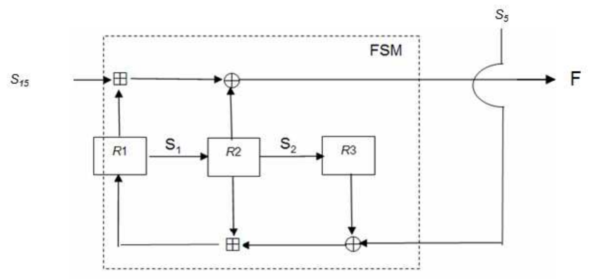


Figure 4: FSM Synchronization

2 SNOW 3G algorithm operation

SNOW 3G has been designed for the use as the base algorithm for the second set of 3GPP confidentiality and integrity algorithms [4]. As most symmetric key algorithms, it uses simple linear operations such as byte or word shifts and exclusive-OR operations, modular addition as well as substitutions through non linear 32 by 32 bit S-boxes. It also heavily uses polynomial evaluation over GF(2^{32}) during the linear feedback shift register computations.

SNOW 3G is a word oriented stream cipher that generates a sequence of 32-bit words under the control of a 128-bit key and a 128-bit initialization variable. These words can be used to mask the plaintext. First a key initialization is performed, i.e. the cipher is clocked without producing output, then with every clock tick it produces a 32-bit word of output.

2.1 Initialisation Mode

SNOW 3G is initialized, as illustrated in Figure 5 with a 128-bit key consisting of four 32-bit words $\mathbf{k}_0$, $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{k}_3$ and an 128-bit initialization variable consisting of four 32-bit words $\mathbf{IV}_0$, $\mathbf{IV}_1$, $\mathbf{IV}_2$, $\mathbf{IV}_3$ as follow [2]:
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 5 bellow represents the Initialization Mode of SNOW 3G.

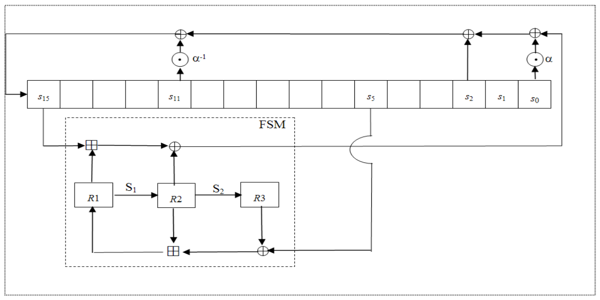


Figure 5: Mode Initialization de SNOW 3G [2]

2.2 Keystream Generation

First, the FSM is clocked once and the FSM output word is discarded. Then, the LFSR is clocked in Keystream mode. After that, n 32-bit words of keystream are produced [2]:

```
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: $\mathbf{z}_t = \mathbf{F} \oplus \mathbf{s0}$.
```

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

}

Figure 6 illustrate SNOW 3G Keystream Mode.

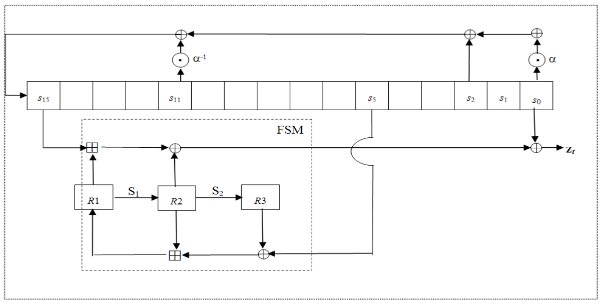


Figure 6: Mode Keystream de SNOW 3G [2]

3 Time Complexity

3.1 Definition

Fixing a model of computation and focusing on algorithms that halt on each input, we consider the number of steps (i.e., application of the computation rule) taken by the algorithm on each possible input. The later function is called the **time complexity** of the algorithm; that is t_A: $\{0, 1\}^* \rightarrow \mathbb{N}$ is called the time complexity of the algorithm A if, for every x, on input x algorithm A halts after exctly $t_A(x)$ steps.

We will mostly interested in the dependence of the time complexity on the input length, when taking the maximum over all inputs of the relevant length. That is, for t_A as in the forgoing, we will consider $T_A : \mathbb{N} \rightarrow \mathbb{N}$ defined by $T_A(n) = \max_{x\in\{0,1\}^n} \{t_A(x)\}$. Abusing terminology, we sometimes refer to T_A as the time complexity of A [1].

3.2 Time complexity of SNOW 3G components and functions

Mathematical Recall:

Definition of "**O**": A function f(n) is called O(g(n)) if there exists a constant c > 0: f(n)<cg(n) for n>n_0.

3.2.1 Time complexity of SNOW 3G functions

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, MUL_α and DIV_α is* ***O(1)****.*

Proof. As we have seen above, the functions that are involved by SNOW 3G security modules are:

- MULx function: This function receives two 8-bit input parameters c and V, and gives an 8-bit output. Then the MULx time complexity T_{MULx} is **O(1)**.
- MULxPOW function: This function receives three 8-bit input parameters c and V and a positive integer i, and gives an 8-bit output. Even if we can remark that MULxPOW is a recursive function, but it receives a finite input data, then the MULxPOW time complexity $T_{MULxPOW}$ is also **O(1)**.
- MUL_α, DIV_α functions: These functions are a concatenation of the result of MULxPOW function with different sets of finite number of input parameters, and give an 8-bit output. Then the MUL_α and DIV_α time complexity is **O(1)**.

□

3.2.2 Time complexity of SNOW 3G security components

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

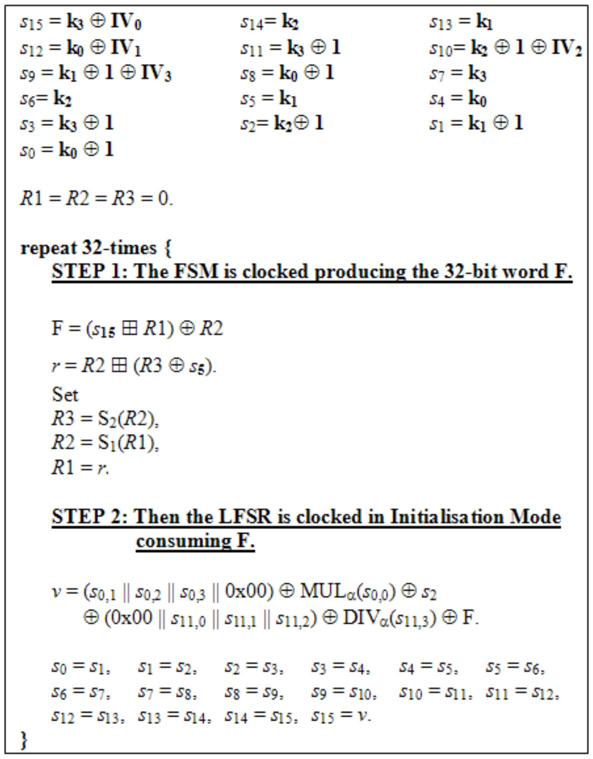
Proof. You find bellow the property proof of the three components.

- S-Box S_1 and S-Box S_2: The S-Boxes S1 and S2 map a 32-bit input to a 32-bit output. So, they deal with a finite number of data; Then the time complexity T_{S_1} and T_{S_2} is **O(1)**.
- addition modulo 2^{32}: As defined before, we can conclude that the time complexity of the addition modulo 2^{32} is **O(1)**.

□

3.3 Time complexity of SNOW 3G

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 produces as output depend on the input parameter n. This input parameter present 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 operation, the Initialization Mode can be presented by Figure 7.

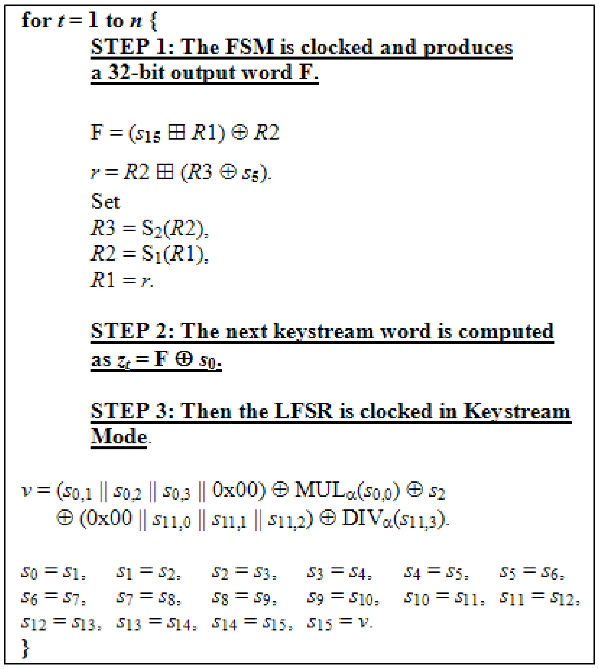


```
$s_{15} = \mathbf{k_3} \oplus \mathbf{IV_0}$        $s_{14} = \mathbf{k_2}$            $s_{13} = \mathbf{k_1}$
$s_{12} = \mathbf{k_0} \oplus \mathbf{IV_1}$        $s_{11} = \mathbf{k_3} \oplus \mathbf{1}$      $s_{10} = \mathbf{k_2} \oplus \mathbf{1} \oplus \mathbf{IV_2}$
$s_9 = \mathbf{k_1} \oplus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

$R1 = R2 = R3 = 0.$

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 7: SNOW 3G Initialisation Mode algorithm

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

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 by Figure 8.



```
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 8: SNOW 3G Keystream Mode algorithm

We note that the Keystream Mode operation is based on an n loop of functions whose time complexity is $\mathbf{O(1)}$.

Theorem 3.1. *SNOW 3G algorithm has a linear time complexity* $T_{SNOW3G} = \boldsymbol{O(n)}$.

4 Space Complexity

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.

4.1 Definition

Space complexity is meant to measure the amount of temporary storage (i.e. computer's memory) used when performing a computational task.

We refer to the memory used for storing some intermediate results of the computation. Since our focus will be on using memory that is sub-linear in the input length, it is important to use a model in which one can differentiate memory used from for computation from memory used for storing the initial input or the final output. In the context of Turing machines, this is done by considering multi-tape Turing machines such that the input is presented on a special read-only tape (called the input tape), the output is written on a special write-only tape (called the output tape), and the intermediate results are stored on a work-tape. Thus, the input and output tapes cannot be used for storing intermediate results. The **space complexity** of such a machine M is defined as a function s_M such as $s_M(x)$ is the number of cells of the work-tape that are scanned by M on input x. As in the case of time complexity, we will usually refer to $S_A(\text{n}) = \max_{x\in\{0,1\}^n} \{s_A(x)\}$ [1].
There is however, an alternative definition, which provides a more accurate account of the actual storage. Specially, the **binary space complexity** of a computation refers to the number of bits that can be stored in these cells, thus multiplying the number of cells by the logarithm of the finite set of work-symbols of the machine [1].

4.2 Space complexity of SNOW 3G components and functions

4.2.1 Space complexity of SNOW 3G functions

Since we are interested in calculating the amount of temporary storage used by SNOW 3G during its operation, a simple analysis of its functions and security components exposed above, shows that SNOW 3G uses a constant temporary storage that doesn't depend on the parameter "n" which represent the plaintext (and the keystream) length. Indeed, in the following paragraph, we will calculate the space complexity of each functions and components used in SNOW 3G and then SNOW 3G space complexity.

Propriety 3. *The space complexity of the functions MULx, MULxPOW, MUL_α and DIV_α is* ***O(1)****.*

Proof. As we have seen above, the functions that are involved by SNOW 3G security modules are:

- MULx function: According to its definition, the MULx function space complexity T_{MULx} is $\mathbf{O(1)}$.
- MULxPOW function: Like MULx, the MULxPOW function space complexity $T_{MULxPOW}$ is also $\mathbf{O(1)}$.

- MUL_α and DIV_α functions: These functions are a concatenation of the result of MULxPOW function with different sets of input parameters. Then the MUL_α and DIV_α space complexity are $\mathbf{O(1)}$.

$\square$

4.2.2 Space complexity of SNOW 3G security components

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

Proof. You find bellow the property proof of the three components.

- S-Box S_1: The S-Box S1 maps a 32-bit input to a 32-bit output and uses a finite number of temporary storage parameters to accomplish its task; Then its space complexity T_{S_1} is $\mathbf{O(1)}$.
- S-Box S_2: As for the S-Box S1, the S-Box S2 space complexity T_{S_2} is $\mathbf{O(1)}$.
- addition modulo 2^{32}: As defined before, we can conclude that the space complexity of the addition modulo 2^{32} is $\mathbf{O(1)}$.

$\square$

4.3 Space complexity of SNOW 3G

Theorem 4.1. *SNOW 3G algorithm has a constant space complexity.*

Proof. After calculating the space complexity of the different functions and security components of SNOW 3G, we can conclude that the space complexity of SNOW 3G in its two modes, initialization and keystream modes is $\mathbf{O(1)}$. $\square$

5 Conclusion

In this paper, a detailed study of the stream cipher SNOW 3G structure has been carried out. We have studied the two interactive modules which constitute the SNOW 3G algorithm and their respective components. The objective is to understand enough SNOW 3G operation and to study its time and space complexity. We have found that SNOW 3G has a linear time complexity, which guarantee efficiency and rapidity during the encryption/decryption process. Furthermore, SNOW 3G has a constant space complexity. SNOW 3G consume a constant and already known amount of temporary memory which is very useful for systems with small working memory such as mobile equipments. From the study of the time and space complexity of SNOW 3G, we can conclude that this stream cipher was well chosen to be the heart of the confidentiality and integrity algorithms (UEA2/UIA2) of the 3^{rd} generation of mobile Telecommunications since 2006, and also for the first set of security algorithms (EEA1/EIA1) for LTE.

References

[1] Oded Goldreich, "*Computational Complexity: A Conceptual Perspective*", Cambridge University Press, 2008.

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

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

[4] 3GPP specifications: http://www.3gpp.org

Received: March, 2010