ANALYSIS OF SNOW 3G$^{\oplus}$ RESYNCHRONIZATION MECHANISM

Alex Biryukov, Deike Priemuth-Schmid and Bin Zhang
LACS, University of Luxembourg, Rue Richard Coudenhove-Kalergi 6, Luxembourg, Luxembourg

Keywords: Stream ciphers, SNOW 3G, Resynchronization attack.

Abstract: The stream cipher SNOW 3G designed in 2006 by ETSI/SAGE is a base algorithm for the second set of 3GPP confidentiality and integrity algorithms. This paper is the first attempt of cryptanalysis of this algorithm in the public literature. We look at SNOW 3G in which two modular additions are replaced by xors, which is called SNOW 3G$^{\oplus}$. We show that the feedback from the FSM to the LFSR is very important, since we can break a version without such a feedback using a pair of *known* IVs with practical complexities (2^{57} time and 2^{33} keystream). We then extend this technique into a differential *chosen* IV attack on SNOW 3G$^{\oplus}$ and show how to break 16 out of 33 rounds with the feedback.

1 INTRODUCTION

The SNOW 3G stream cipher is the core of the 3GPP confidentiality and integrity algorithms UEA2 and UIA2, published in 2006 by the 3GPP Task Force (ETSI1, 2006). Compared to its predecessor, SNOW 2.0 (Ekdahl and Johansson, 2002), SNOW 3G adopts a finite state machine (FSM) of three 32-bit words and 2 S-Boxes to increase the resistance against algebraic attacks by Billet and Gilbert (Billet and Gilbert, 2005). Full evaluation of the design is not public, but a survey of this evaluation is given in (ETSI2, 2006). In (ETSI2, 2006), SNOW 3G$^{\oplus}$ (in which the two modular additions are replaced by xors) is defined and evaluated. It shows that SNOW 3G has remarkable resistance against linear distinguishing attacks (Nyberg and Wallén, 2006; Watanabe et al., 2004), while SNOW 3G$^{\oplus}$ offers much better resistance against algebraic attacks.

In this paper, we presents the first attempt of cryptanalysis of the resynchronization mechanism of SNOW 3G$^{\oplus}$. We show that the feedback from the FSM to the LFSR during the key/IV setup phase is vital for the security of this cipher, since we can break a version without such a feedback with two *known* IV's in 2^{57} time, 2^{33} data complexity and for an arbitary number of the key/IV setup rounds! We then restore the feedback and study SNOW 3G$^{\oplus}$ against differential chosen IV attacks. We show attacks on SNOW 3G$^{\oplus}$ with 14, 15 and 16 rounds of initialization with complexity $2^{42.7}$, $2^{92.2}$ and $2^{124.2}$ respectively.

This paper is organized as follows. We give a description of SNOW 3G and SNOW 3G$^{\oplus}$ in Section 2. The known IV attack on SNOW 3G$^{\oplus}$ without the FSM to LFSR feedback is presented in Section 3 and the differential chosen IV attack on SNOW 3G$^{\oplus}$ with the feedback is presented in Section 4. Finally, some conclusions are given in Section 5.

2 DESCRIPTION OF SNOW 3G AND SNOW 3G$^{\oplus}$

SNOW 3G is a word-oriented synchronous stream cipher with 128-bit key and 128-bit IV, each considered as four 32-bit words vector. It consists of a linear feedback shift register (LFSR) of sixteen 32-bit words and a finite state machine (FSM) with three 32-bit words, shown in Figure 1. Here '$\oplus$' denotes

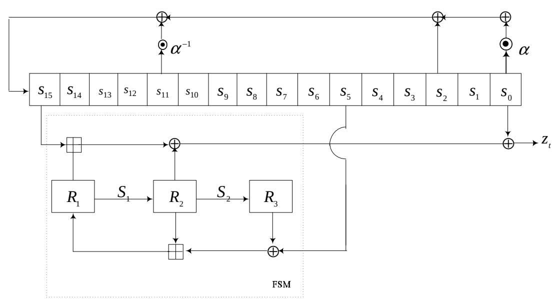

Figure 1: Keystream generation of SNOW 3G.

the bit-wise xor and '$\boxplus$' denotes the addition modulo 2^{32}. The feedback word of the LFSR is recursively

Biryukov A., Priemuth-Schmid D. and Zhang B. (2010).
ANALYSIS OF SNOW 3GâŁŢ RESYNCHRONIZATION MECHANISM.
In *Proceedings of the International Conference on Security and Cryptography*, pages 327-333
DOI: 10.5220/0002926603270333

computed as

$$s^t_{15} = \alpha^{-1} \cdot s^{t-1}_{11} \oplus s^{t-1}_{2} \oplus \alpha \cdot s^{t-1}_{0},$$

where α is the root of the $GF(2^8)[x]$ polynomial $x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}$ with β being the root of the $GF(2)[x]$ polynomial $x^8 + x^7 + x^5 + x^3 + 1$. The FSM has two input words s^t_5 and s^t_{15} from the LFSR and is updated as

$$R^t_1 = R^{t-1}_2 \boxplus (R^{t-1}_3 \oplus s^{t-1}_5),$$
$$R^t_2 = S_1(R^{t-1}_1), \quad R^t_3 = S_2(R^{t-1}_2),$$

with the output word $F^t = (s^t_{15} \boxplus R^t_1) \oplus R^t_2$, where S_1 and S_2 are 32-bit to 32-bit S-boxes defined as compositions of 4 parallel applications of two 8-bit to 8-bit small S-boxes, S_R and S_Q, with a linear diffusion layer respectively. Here S_R is the well known AES S-box and S_Q is defined as $S_Q(x) = x \oplus x^9 \oplus x^{13} \oplus x^{15} \oplus x^{33} \oplus x^{41} \oplus x^{45} \oplus x^{47} \oplus x^{49} \oplus 0x25$ for $x \in GF(2^8)$ defined by $x^8 + x^6 + x^5 + x^3 + 1$. If we decompose a 32-bit word B into four bytes $B = B^0 \| B^1 \| B^2 \| B^3$ with B^0 being the most and B^3 the least significant bytes, then for $i = 1, 2$, the S-boxes are

$$S_i(B) = MC_i \cdot (S_R(B^0), S_R(B^1), S_R(B^2), S_R(B^3))^T,$$

where MC_1 is the AES mix-column for S_1 over $GF(2^8)$ defined by $x^8 + x^4 + x^3 + x + 1$ and MC_2 is the similar operation for S_2 over $GF(2^8)$ defined by $x^8 + x^6 + x^5 + x^3 + 1$.

SNOW 3G is initialized with the key $K = (k_0, k_1, k_2, k_3)$ and the $IV = (IV_0, IV_1, IV_2, IV_3)$ as follows. Let $\mathbf{1}$ be the all-one word, the LFSR is initialized as follows.

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

The FSM is initialized with $R_1 = R_2 = R_3 = 0$. Then run the cipher 32 times with the FSM output F xored to the feedback of the LFSR and no keystream generated. After this, the cipher is switched into the keystream generation mode, but the first keystream word is discarded. Hence, there are 33 initialization rounds. The keystream word generated at clock t is $z^t = s^t_0 \oplus F^t$. If we replace the two modulo additions in SNOW 3G by xors, we get SNOW 3G$^\oplus$.

3 KNOWN IV ATTACK ON SNOW 3G$^\oplus$ WITHOUT FSM TO LFSR FEEDBACK

In this section, we consider a known IV attack on SNOW 3G$^\oplus$ without the FSM to LFSR feedback, in which the attacker has access to two keystreams corresponding to (K, IV_a) and (K, IV_b), where IV_a and IV_b are arbitrary known IVs. This attack works for any number of key/IV setup rounds.

Let $R^t_{i,a}$ and $R^t_{i,b}$ be the individual values in the FSM register R_i at clock t, then we have

$$\Delta R^t_1 = R^t_{1,a} \oplus R^t_{1,b},$$
$$R^t_{2,a} = S_1(R^{t-1}_{1,a}), \quad R^t_{2,b} = S_1(R^{t-1}_{1,b}),$$
$$\Delta R^t_2 = R^t_{2,a} \oplus R^t_{2,b} = S_1(R^{t-1}_{1,a}) \oplus S_1(R^{t-1}_{1,b}).$$

Here we define a new notation

$$\overset{\text{out}}{\Delta} S_1(\Delta R^{t-1}_1) \triangleq S_1(R^{t-1}_{1,a}) \oplus S_1(R^{t-1}_{1,b}).$$

During the keystream generation, we have the following equations for the differences at clock t

$$\Delta z^t = \Delta s^t_{15} \oplus \Delta R^t_1 \oplus \Delta R^t_2 \oplus \Delta s^t_0,$$
$$\Delta R^t_1 = \Delta R^{t-1}_2 \oplus \Delta R^{t-1}_3 \oplus \Delta s^{t-1}_5,$$
$$\Delta R^t_2 = \overset{\text{out}}{\Delta} S_1(\Delta R^{t-1}_1), \quad \Delta R^t_3 = \overset{\text{out}}{\Delta} S_2(\Delta R^{t-1}_2).$$

The differences in the LFSR part propagate linearly and are completely predictable.

The main procedures of our attack are: assume that at time t we have $\Delta R^t_1 = 0$. From the linear evolution of the difference in the LFSR and the keystream difference equations, we deduce potential differences in the other FSM registers at different times. Knowing the input-output difference for the S-boxes, deduce the few possibilities for the actual values of the FSM registers. Combine the knowledge of the FSM state with that of the keystream to get linear equations on the LFSR state. Collect enough equations to get a solvable linear system which will recover the state of the LFSR. By the invertibility of the cipher, run it backwards to find the 128-bit secret key K.

Assume $\Delta R^t_1 = 0$. If this is not true, we just take the next clock and so on. If we try this step 2^{32} times, then it will happen with a good probability. Denote the time that $\Delta R_1 = 0$ by $t = 1$. Then $\Delta R^1_1 = 0$, $\Delta R^2_2 = 0$ and $\Delta R^3_3 = 0$. From the keystream equation at $t = 1$, we know ΔR^1_2; similarly we know ΔR^2_1 at $t = 2$, from which we can derive ΔR^1_3, as shown below. Hereafter, we denote the known difference values by Δk_i.

clock t	ΔR_1	ΔR_2	ΔR_3
1	0	Δk_1	Δk_3
2	Δk_2	0	
3			0

At $t = 3$, we have

$$\Delta R^2_3 \oplus \Delta R^3_2 = \Delta z^3 \oplus \Delta s^3_{15} \oplus \Delta s^2_5 \oplus \Delta s^3_0.$$

By the notations introduced before, we get

$$\overset{\text{out}}{\Delta} S_2(\Delta k_1) \oplus \overset{\text{out}}{\Delta} S_1(\Delta k_2) = \Delta k_4. \tag{1}$$

Here we have $2^{28} \cdot 2^{28}/2^{32} = 2^{24}$ pairs satisfying (1). (In the two 8-bit S-boxes, there are at most 2^7 possible output differences for any fixed input difference.) To enumerate the possible pairs, we proceed as follows. First rewrite (1) as

$$\begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta k_2^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^3) \end{pmatrix} = \begin{pmatrix} \overset{\text{out}}{\Delta} S_Q(\Delta k_1^0) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^1) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^2) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^3) \end{pmatrix} \oplus \begin{pmatrix} p_0^{\text{msb}} \\ p_1^{\text{msb}} \\ p_2^{\text{msb}} \\ p_3^{\text{msb}} \end{pmatrix} \oplus MC_1^{-1} \cdot \begin{pmatrix} \Delta k_4^0 \\ \Delta k_4^1 \\ \Delta k_4^2 \\ \Delta k_4^3 \end{pmatrix},$$

where p_i^{msb} $(i = 0,1,2,3)$ denotes a byte polynomial which contains only the most significant bits of all the four $\overset{\text{out}}{\Delta} S_Q$ values. For a detailed explanation, please see the Appendix. Thus we can fulfill the enumeration byte by byte. For the first row, we need the value of $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^0)$, which has 2^7 possibilities and three more bits for p_0^{msb}. Then we check whether the value computed at the right side of the equation is a correct value for $\overset{\text{out}}{\Delta} S_R(\Delta k_2^0)$. This would cost 2^{10} steps and we will obtain 2^9 solutions for this equation. For the next three equations, since we already know the leading bits, we only have 2^6 possibilities left in each byte equation, which yields the same time complexity and 2^5 solutions. To get the solution of the word equation, we have to combine the corresponding byte solutions and get $2^9 \cdot 2^5 \cdot 2^5 \cdot 2^5 = 2^{24}$ solutions, which needs about $2 \times 2^{24} = 2^{25}$ words of memory. Now, the states of the FSM are as follows.

clock t	ΔR_1	ΔR_2	ΔR_3
1	0	Δk_1	Δk_3
2	Δk_2	0	(2^{24})
3	(2^{24})	(2^{24})	0
4			

next part $\rightarrow$ reduction

clock t	ΔR_1	ΔR_2	ΔR_3
1	0	Δk_1	Δk_3
2	Δk_2	0	(2^{20})
3	(2^{20})	(2^{20})	0
4	(2^{20})	(2^{20})	

Each possible value of ΔR_2^3 results in a possible value of ΔR_1^4. At $t = 4$, we have

$$\Delta R_2^3 \oplus \Delta R_2^4 = \Delta z^4 \oplus \Delta s_{15}^4 \oplus \Delta s_5^3 \oplus \Delta s_0^4.$$

Replacing the difference ΔR_2^4 with the S-Box representation, we receive $\Delta R_2^3 \oplus \overset{\text{out}}{\Delta} S_1(\Delta R_1^3) = \Delta k_5$. Let $\Delta R_1^3 = c^0 \| c^1 \| c^2 \| c^3$, $\Delta R_2^3 = a^0 \| a^1 \| a^2 \| a^3$. Expanding this equation to the byte form, we get

$$\begin{pmatrix} \overset{\text{out}}{\Delta} S_R(c^0) \\ \overset{\text{out}}{\Delta} S_R(c^1) \\ \overset{\text{out}}{\Delta} S_R(c^2) \\ \overset{\text{out}}{\Delta} S_R(c^3) \end{pmatrix} = MC_1^{-1} \cdot \begin{pmatrix} a^0 \\ a^1 \\ a^2 \\ a^3 \end{pmatrix} \oplus MC_1^{-1} \cdot \begin{pmatrix} \Delta k_5^0 \\ \Delta k_5^1 \\ \Delta k_5^2 \\ \Delta k_5^3 \end{pmatrix}.$$

We have to insert all the 2^{24} possible pairs of $(\Delta R_2^3, \Delta R_1^3)$ and verify the value $\overset{\text{out}}{\Delta} S_R$ for the single bytes. This results in a time complexity of 2^{24}. There are $\frac{2^{24} \cdot 2^{28}}{2^{32}} = 2^{20}$ entries satisfy this equation. This means we have 2^{20} sequences $(\Delta R_3^2, \Delta R_1^3, \Delta R_2^3, \Delta R_1^4, \Delta R_2^4)$ left. For each of them, we know the input-output difference of S_1 at clock 2 and 3. Thus, we can recover $(2 \cdot \frac{126}{127} + 4 \cdot \frac{1}{127})^4 = 16.51$ sorted pairs of values for S_1. This means that we have $\frac{16.51}{2} = 8.255$ possible values for ΔR_3^4. Looking at clock 5, we have $\Delta R_2^4 \oplus \Delta R_3^4 \oplus \overset{\text{out}}{\Delta} S_1(\Delta R_1^4) = \Delta k_6$. We can rewrite this equation into byte form and check the 2^{20} remaining sequences by the byte equations. There are $\frac{2^{20} \cdot 8.255 \cdot 2^{28}}{2^{32}} \approx 2^{19.05}$ possible sequences left and the complexity is about $2^{20} \cdot 8.255 = 2^{23.05}$. This identification of the individual values in the FSM for both keystreams has to be repeated for the next 9 clocks. Each step will have a lower time complexity than the one before and will reduce the possible number of differences. The time complexity for all 10 steps together is $\sum_{i=0}^{9} 2^{20} \cdot (\frac{2^{27}}{127^4})^i \cdot \frac{2^{31}}{127^4} = 2^{24.1}$ and the number of sequences left is $2^{20} \cdot (\frac{2^{27}}{127^4})^{10} = 2^{10.5}$. Then we insert the individual values of the FSM into the keystream generation equations and the FSM update equations to get a linear system of the LFSR initial states. This would need a time complexity of $2^{10.5} \cdot 2^{10} = 2^{20.5}$ steps. The overall time complexity is

$$2^{32} \cdot [2^{10} + 2^{24} + \sum_{i=0}^{9} (2^{20} \cdot (\frac{2^{27}}{127^4})^i \cdot \frac{2^{31}}{127^4})] = 2^{57.1}.$$

The memory requirement is 2^{25} words and the keystream is of length 2^{33} words.

4 DIFFERENTIAL CHOSEN IV ATTACKS ON REDUCED ROUND SNOW 3G$^{\oplus}$

Now we look at the full SNOW 3G$^{\oplus}$ (with the feedback). We consider a differential chosen IV attack scenario. Assume that we have two 128-bit IVs differing only in the most significant word IV_0, which gives the difference in s_{15} of the LFSR. As mentioned below in Section 4.2 and Section 4.3, we can restrict the difference to a single byte of IV_0 in order to reduce the complexity of our attacks. Denote this difference by Δd. Then until round 10, this difference will not affect the FSM. In round 11, the known Δd enters the FSM word R_1.

4.1 Reduced Initialization of 12 Rounds

Since all the differences in the FSM are 0, there are no differences fed back into the LFSR. Thus the differences in the LFSR are all known. Our knowledge of differences in the FSM is shown below. We try to compute the unknown values ("?"s) in this table.

round	clock t	ΔR_1	ΔR_2	ΔR_3
11	-1	Δd	0	0
12	0	Δd	?	0
	1	?	?	

From the keystream equation $\Delta z^0 = \Delta s^0_{15} \oplus \Delta R^0_1 \oplus \Delta R^0_2 \oplus \Delta s^0_0$, where $\Delta R^0_1 = \Delta d$, we get ΔR^0_2, which gives us immediately ΔR^1_1 and also ΔR^1_2 from the next keystream equation. Therefore, we have only one known sequence $(\Delta R^{-1}_1 = \Delta d, \Delta R^{-1}_2 = \Delta R^{-1}_3 = 0, \Delta R^0_1 = \Delta d,\ \Delta R^0_2, \Delta R^0_3 = 0, \Delta R^1_1, \Delta R^1_2)$. Now we know the input and output difference of S_1: $\Delta R^{-1}_1 = \Delta d \to S_1 \to \Delta R^0_2$. Thus, we switch from the differences of the FSM words to the individual values of them, similar to the procedures explained in Section 3. The time complexity is $10 \cdot \frac{2^{31}}{127^4} = 2^{6.4}$ steps. Afterwards we insert the individual values of the FSM into the keystream generation equations and the FSM update equations to get a linear system of the LFSR initial states with a complexity of 2^{10}. We use the keystream equation of clock 12 to check the candidates. The total time complexity is $2^{6.4} + 2^{10} = 2^{10.1}$ steps, the memory complexity is small and the known keystream is only 12 words for each IV.

4.2 Reduced Initialization of 13 Rounds

Here we extend the attack above by one more round. In the 13 round case, since all the differences in the FSM until now are either 0 or the known Δd, no unknown difference was fed back into the LFSR. Thus, the differences in the LFSR values are known. We compute "?"s in the following table as follows.

round	clock t	ΔR_1	ΔR_2	ΔR_3
11	-2	Δd	0	0
12	-1	Δd	?	0
13	0	?	?	

From Δz^0 and ΔR^0_1, we have

$$\Delta z^0 = \Delta s^0_{15} \oplus \Delta R^{-1}_2 \oplus \Delta s^{-1}_5 \oplus \Delta R^0_2 \oplus \Delta s^0_0,$$

which is

$$\Delta R^{-1}_2 \oplus \Delta R^0_2 = \Delta z^0 \oplus \Delta s^0_{15} \oplus \Delta s^{-1}_5 \oplus \Delta s^0_0.$$

Then we replace the differences at the left side with their S-Boxes description, denote the known part at the right side with k_0 and get the equation

$$\overset{\text{out}}{\Delta} S_1(\Delta d) \oplus \overset{\text{out}}{\Delta} S_1(\Delta d) = \Delta k_0.$$

Multiplying by MC^{-1}_1, we get the byte form equation

$$\begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta d^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^3) \end{pmatrix} \oplus \begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta d^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^3) \end{pmatrix} = MC^{-1}_1 \begin{pmatrix} \Delta k^0_0 \\ \Delta k^1_0 \\ \Delta k^2_0 \\ \Delta k^3_0 \end{pmatrix}.$$

We can check these four byte equations in $4 \cdot 2^7 = 2^9$ steps. The number of solutions will be $\frac{2^{28} \cdot 2^{28}}{2^{32}} = 2^{24}$ pairs of $(\Delta R^{-1}_2, \Delta R^0_2)$. We have 2^{24} sequences $(\Delta R^{-2}_1 = \Delta d, \Delta R^{-2}_2 = \Delta R^{-2}_3 = 0, \Delta R^{-1}_1 = \Delta d, \Delta R^{-1}_2, \Delta R^{-1}_3 = 0, \Delta R^0_1, \Delta R^0_2)$. Again, we switch from the differences of the FSM words to the individual values of them by using the input and output difference of S_1: $\Delta R^{-2}_1 = \Delta d \to S1 \to \Delta R^{-1}_2$. The time complexity of this step is $\sum_{i=0}^{9} 2^{24} \cdot (\frac{2^{27}}{127^4})^i \cdot \frac{2^{31}}{127^4} = 2^{28.09}$. In the end, we have $2^{24} \cdot (\frac{2^{27}}{127^4})^{10} = 2^{14.45}$ difference sequences left. The memory complexity is $2^{25} \cdot 10 \cdot 3 = 2^{29.91}$ words. We then insert the individual values of the FSM into the keystream generation equations and the FSM update equations to get a linear system of the LFSR initial states. This would need a time complexity of $\frac{2^{294}}{127^{40}} \cdot 2^{10} = 2^{24.45}$. The overall time complexity is

$$2^9 + \sum_{i=0}^{9} \left(2^{24} \cdot (\frac{2^{27}}{127^4})^i \cdot \frac{2^{31}}{127^4} \right) + \frac{2^{294}}{127^{40}} \cdot 2^{10} = 2^{28.2}$$

steps. The memory complexity is $2^{29.91}$ words and the keystream is of length 12 words for each IV.

If we restrict the known arbitrary difference Δd to a word with three bytes equal to zero and only one non zero byte, we can reduce our attack complexity considerably. We then have only one pair $(\Delta R^{-1}_2, \Delta R^0_2)$ of difference left, as in the attack on 12 rounds explained in Section 4.1. In this way, we will have the same time complexity $2^{10.1}$ and the memory requirement is small. The keystream will be of 12 words for each IV.

4.3 Reduced Initialization of 14 Rounds

Nearly all the differences in the LFSR are known, the only unknown difference is ΔR^{-2}_2, which was fed back into the LFSR, the remaining differences are either 0 or the known Δd. We guess the individual value $R_{1,a}^{-3}$ for the first pair (K, IV_a) with complexity of 2^{32}. From the value $R_{1,a}^{-3}$, we get with $\Delta R_1^{-3} = \Delta d$ the value $R1_b^{-3}$ for the second pair (K, IV_b). Furthermore we obtain $R_{2,a}^{-2}, R_{2,b}^{-2}, R_{3,a}^{-1}, R_{3,b}^{-1}$. We denote the known difference ΔR_2^{-2} with Δk_0, the linear dependent ΔR_1^{-1} with Δk_1 and ΔR_3^{-1} with Δk_2. This gives the following differences for the FSM.

round	clock t	ΔR_1	ΔR_2	ΔR_3
11	-3	Δd	0	0
12	-2	Δd	Δk_0	0
13	-1	Δk_1	?	Δk_2
14	0	?	?	

From

$$\Delta z^0 = \Delta s_{15}^0 \oplus \Delta R_1^0 \oplus \Delta R_2^0 \oplus \Delta s_0^0 ,$$

we insert the update equations for ΔR_1^0 and ΔR_2^0 and receive

$$\begin{aligned} \Delta z^0 \quad &= \quad \Delta s_{15}^0 \oplus \overset{\text{out}}{\Delta} S_1(\Delta d) \oplus \Delta k_2 \oplus \Delta s_5^{-1} \\ &\quad \oplus \overset{\text{out}}{\Delta} S_1(\Delta k_1) \oplus \Delta s_0^0 , \end{aligned}$$

which gives

$$\begin{aligned} &\overset{\text{out}}{\Delta} S_1(\Delta d) \oplus \overset{\text{out}}{\Delta} S_1(\Delta k_1) \\ &= \quad \Delta z^0 \oplus \Delta s_{15}^0 \oplus \Delta k_2 \oplus \Delta s_5^{-1} \oplus \Delta s_0^0 . \end{aligned}$$

We denote the known right part by Δk_3, multiply the equation with MC_1^{-1} and rewrite it in byte notation as

$$\begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta d^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta d^3) \end{pmatrix} \oplus \begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta k_1^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_1^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_1^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_1^3) \end{pmatrix} = MC_1^{-1} \begin{pmatrix} \Delta k_3^0 \\ \Delta k_3^1 \\ \Delta k_3^2 \\ \Delta k_3^3 \end{pmatrix} .$$

Then we check this equation line by line for each byte in $4 \cdot 2^7 = 2^9$ steps. The number of solutions will be $\frac{2^{28} \cdot 2^{28}}{2^{32}} = 2^{24}$ pairs of $(\Delta R_2^{-1}, \Delta R_2^0)$. Again, we switch from the differences of the FSM words to the individual values of them by using the input and output difference of S_1: $\Delta R_1^{-2} \to S_1 \to \Delta R_2^{-1}$. Since we start with 2^{24} sequences, we have completely the same procedure as in the attack on 13 rounds of initialization and thus the same complexities. The overall time complexity is the same as that in 12 rounds of initialization for each guess of $R1_1^{-3}$, which gives

$$2^{32} \cdot \left[2^9 + \sum_{i=0}^{9} \left(2^{24} \cdot \left(\frac{2^{27}}{127^4}\right)^i \cdot \frac{2^{31}}{127^4} \right) + \frac{2^{294}}{127^{40}} \cdot 2^{10} \right] = 2^{60.2} .$$

The memory requirement is $2^{29.91}$ words and the keystream is of length 12 words for each IV.

If we restrict the known difference Δd to only one byte in IV_0, we can reduce our attack complexity to $2^{42.7}$ with similar procedures as above. The corresponding memory complexity is 2^9 words and the keystream is of 12 words for each IV.

4.4 Reduced Initialization of 15 Rounds and 16 Rounds

Nearly all the differences in the LFSR are known, only two unknown differences ΔR_2^{-3} and ΔR_2^{-2} were fed back into the LFSR, the remaining differences are either 0 or the known Δd. We guess the individual values of $R_{1,a}^{-4}$ and $R_{1,a}^{-3}$ for the first pair (K, IV_a) with complexity of 2^{64}. From the value $R_{1,a}^{-4}$ and $\Delta R_1^{-4} = \Delta d$, we get the values of $R_{1,b}^{-4}$, $R_{2,a}^{-3}, R_{2,b}^{-3}$, $R_{3,a}^{-2}, R_{3,b}^{-2}$. Denote the known difference ΔR_2^{-3} by Δk_0, ΔR_1^{-2} by Δk_1 and ΔR_3^{-2} by Δk_2. From $R_{1,a}^{-3}$ and $\Delta R_1^{-3} = \Delta d$, we get the values of $R_{1,b}^{-3}, R_{2,a}^{-2}, R_{2,b}^{-2}, R_{3,a}^{-1}, R_{3,b}^{-1}$. Again, we denote the now known difference ΔR_2^{-2} by Δk_3, ΔR_1^{-1} by Δk_4 and ΔR_3^{-1} by Δk_5. This gives the following differences for the FSM.

round	clock t	ΔR_1	ΔR_2	ΔR_3
11	-4	Δd	0	0
12	-3	Δd	Δk_0	0
13	-2	Δk_1	Δk_3	Δk_2
14	-1	Δk_4	?	Δk_5
15	0	?	?	

We have now the same starting point as that of the attack on 14 initialization rounds. We proceed in the way as explained there. Since we guessed one more word in the beginning of the attack, the time complexity becomes

$$2^{32} \cdot 2^{60.2} = 2^{92.2} .$$

The memory complexity remains $2^{29.91}$ words and the keystream is of length 12 words for each IV.

In the 16 rounds case, we guess one more word and then proceed as that of the attack on 15 rounds. The time complexity is

$$2^{32} \cdot 2^{92.2} = 2^{124.2}$$

and the memory complexity remains $2^{29.91}$ words and the keystream is of length 12 words for each IV.

The summary of our results is given in Table 1.

5 CONCLUSIONS

In this paper, we have shown *known* IV and *chosen* IV resynchronization attacks on SNOW $3G^{\oplus}$. We can attack arbitrary many key/IV setup rounds of SNOW $3G^{\oplus}$ if there is no feedback from FSM to LFSR. With such feedback, we show key recovery attacks up to 16 rounds of initialization by using a few keystream words. Our results indicate that about half of the initialization rounds of SNOW $3G^{\oplus}$ might succumb to chosen IV resynchronization attacks. The remaining

Table 1: The summary of our results on SNOW 3G$^{\oplus}$.

attack	data	time	memory
SNOW 3G$^{\oplus}$ without feedback	2^{33}	$2^{57.1}$	2^{25}
SNOW 3G$^{\oplus}$ with feedback 12 rounds	24	$2^{10.1}$	small
13 rounds with 1 byte difference Δd	24	$2^{10.1}$	small
14 rounds with 1 byte difference Δd	24	$2^{42.7}$	2^{9}
15 rounds	24	$2^{92.2}$	$2^{29.91}$
16 rounds	24	$2^{124.2}$	$2^{29.91}$

security margin however is quite significant and thus these attacks pose no threat to the security of SNOW 3G itself.

REFERENCES

Billet, O. and Gilbert, H. (2005). Resistance of SNOW 2.0 Against Algebraic Attacks. In *Topics in Cryptology-CT-RSA'2005*. LNCS vol. 3376, pp. 19-28. Springer-Verlag 2005.

Ekdahl, P. and Johansson T. (2002). A New Version of the Stream Cipher SNOW. In *Selected Areas in Cryptography-SAC 2002*. LNCS vol. 1233, pp. 37-46. Springer-Verlag 2002.

ETSI/SAGE. (2006). Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification, version 1.1 In *September 2006. http://www.3gpp.org/ftp/..*

ETSI/SAGE (2006). Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 5: Design and Evaluation Report, version 1.1 In *September 2006. http://www.3gpp.org/ftp/.*

Nyberg, K. and Wallén, J. (2006). Improved Linear Distinguishers for SNOW 2.0. In *Fast Software Encryption-FSE 2006*. LNCS vol. 4047, pp. 144-162. Springer-Verlag 2006.

Watanabe, D., Biryukov, A., De Cannière, Christophe, (2004). A Distinguishing Attack of SNOW 2.0 with Linear Masking Method. In *Selected Areas in Cryptography-SAC 2003*. LNCS vol. 3006, pp. 222-233. Springer-Verlag 2004.

APPENDIX

We want to simplify the equation

$$\overset{\text{out}}{\Delta} S_2(\Delta k_1) \oplus \overset{\text{out}}{\Delta} S_1(\Delta k_2) = \Delta k_4 \,.$$

The main difficulty is that S_1 and S_2 use the same Mix-Column matrix but over two different fields $GF(2^8)$. At first we rewrite this equation in the byte notation as

$$MC_2 \cdot \begin{pmatrix} \overset{\text{out}}{\Delta} S_Q(\Delta k_1^0) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^1) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^2) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^3) \end{pmatrix} \oplus MC_1 \cdot \begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta k_2^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^3) \end{pmatrix} = \begin{pmatrix} \Delta k_4^0 \\ \Delta k_4^1 \\ \Delta k_4^2 \\ \Delta k_4^3 \end{pmatrix} .$$

Then multiplying this equation with the inverse matrix MC_1^{-1}, we get

$$MC_1^{-1} \cdot \left(MC_2 \cdot \begin{pmatrix} \overset{\text{out}}{\Delta} S_Q(\Delta k_1^0) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^1) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^2) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^3) \end{pmatrix} \right) \oplus \begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta k_2^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^3) \end{pmatrix} = MC_1^{-1} \cdot \begin{pmatrix} \Delta k_4^0 \\ \Delta k_4^1 \\ \Delta k_4^2 \\ \Delta k_4^3 \end{pmatrix} .$$

If we expand the matrix multiplications and have a look at the byte vectors, it shows that the first entry of the first vector contains the byte $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^0)$ and a byte polynomial containing only the most significant bits of all four $\overset{\text{out}}{\Delta} S_Q$ values. We denote this polynomial with p_0^{msb}. The other three rows have similar structures, but with different polynomials p_i^{msb} $(i = 1,2,3)$. Therefore we can rewrite the equation to

$$\begin{pmatrix} \overset{\text{out}}{\Delta} S_R(\Delta k_2^0) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^1) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^2) \\ \overset{\text{out}}{\Delta} S_R(\Delta k_2^3) \end{pmatrix} = \begin{pmatrix} \overset{\text{out}}{\Delta} S_Q(\Delta k_1^0) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^1) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^2) \\ \overset{\text{out}}{\Delta} S_Q(\Delta k_1^3) \end{pmatrix} \oplus \begin{pmatrix} p_0^{\text{msb}} \\ p_1^{\text{msb}} \\ p_2^{\text{msb}} \\ p_3^{\text{msb}} \end{pmatrix} \oplus MC_1^{-1} \cdot \begin{pmatrix} \Delta k_4^0 \\ \Delta k_4^1 \\ \Delta k_4^2 \\ \Delta k_4^3 \end{pmatrix} .$$

We denote by m_0 the most significant bit of the value $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^0)$ and with m_1 the most significant bit of the value $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^1)$ as well as m_2 for $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^2)$ and m_3 for $\overset{\text{out}}{\Delta} S_Q(\Delta k_1^3)$. Then the polynomials p_i^{msb}

$i = 0, \ldots, 3$ are

$$
\begin{aligned}
p_0^{msb} = &(m_1 \oplus m_3)x^7 + (m_0 \oplus m_1)x^6 + (m_2 \oplus m_3)x^5 \\
&+ (m_1 \oplus m_2)x^4 + (m_0 \oplus m_2)x^2 + (m_1 \oplus m_2)x \\
&+ (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \\
p_1^{msb} = &(m_0 \oplus m_2)x^7 + (m_1 \oplus m_2)x^6 + (m_0 \oplus m_3)x^5 \\
&+ (m_2 \oplus m_3)x^4 + (m_1 \oplus m_3)x^2 + (m_2 \oplus m_3)x \\
&+ (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \\
p_2^{msb} = &(m_1 \oplus m_3)x^7 + (m_2 \oplus m_3)x^6 + (m_0 \oplus m_1)x^5 \\
&+ (m_0 \oplus m_3)x^4 + (m_0 \oplus m_2)x^2 + (m_0 \oplus m_3)x \\
&+ (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \\
p_3^{msb} = &(m_0 \oplus m_2)x^7 + (m_0 \oplus m_3)x^6 + (m_1 \oplus m_2)x^5 \\
&+ (m_0 \oplus m_1)x^4 + (m_1 \oplus m_3)x^2 + (m_0 \oplus m_1)x \\
&+ (m_0 \oplus m_1 \oplus m_2 \oplus m_3)
\end{aligned}
$$