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 present 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 IVs in \(2^{57}\) time, \(2^{33}\) data complexity and for an arbitrary 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 \(2^{42}\), \(2^{92}\) and \(2^{124}\) 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 the bit-wise xor and \(\boxplus\) denotes the addition modulo 2\(^{32}\). The feedback word of the LFSR is recursively

Figure 1: Keystream generation of SNOW 3G.
3 KNOWN IV ATTACK ON SNOW 3G\(^2\) WITHOUT FSM TO LFSR FEEDBACK

In this section, we consider a known IV attack on SNOW 3G\(^2\) 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 \(R_{1,t}^i\) and \(R_{2,t}^i\) be the individual values in the FSM register \(R_t\) at clock \(t\), then we have
\[
\Delta R_{1,t}^i = R_{1,t}^i \oplus R_{1,t}^b,
\]
\[
\Delta R_{2,t}^i = S_1(R_{1,t}^i) \oplus S_1(R_{1,t}^b),
\]
\[
\Delta R_{3,t}^i = R_{2,t}^i \oplus R_{2,t}^b,
\]
\[
\Delta R_{4,t}^i = S_1(R_{1,t}^i) \oplus S_1(R_{1,t}^b).
\]

Here we define a new notation
\[
\Delta S_1(\Delta R_{1,t}^i) = S_1(R_{1,t}^i) \oplus S_1(R_{1,t}^b).
\]

During the keystream generation, we have the following equations for the differences at clock \(t\)
\[
\Delta \xi = \Delta x_{13} \oplus \Delta R_{1,t}^i \oplus \Delta R_{2,t}^i \oplus \Delta x_0,
\]
\[
\Delta \eta = \Delta R_{3,t}^i \oplus \Delta R_{4,t}^i \oplus \Delta x_1
\]
\[
\Delta \zeta = \Delta \xi - \Delta \eta.
\]

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 \(\Delta R_{1,t}^1 = 0\). 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 \(\Delta R_1^1 = 0\). 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 \(\Delta R_1^1 = 0\) by \(t=1\). Then \(\Delta R_1^1 = 0\) and \(\Delta R_2^1 = 0\). From the keystream equation at \(t=1\), we know \(\Delta R_1^2\); similarly we know \(\Delta R_2^2\) at \(t=2\), from which we can derive \(\Delta R_3^2\), as shown below. Hereafter, we denote the known difference values by \(\Delta k_i\).

<table>
<thead>
<tr>
<th>clock</th>
<th>(\Delta R_1)</th>
<th>(\Delta R_2)</th>
<th>(\Delta R_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(\Delta k_1)</td>
<td>(\Delta k_3)</td>
</tr>
<tr>
<td>2</td>
<td>(\Delta k_2)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

At \(t=3\), we have
\[
\Delta R_1^3 \oplus \Delta R_2^3 = \Delta \xi \oplus \Delta \xi_3 \oplus \Delta \xi_1 \oplus \Delta \xi_2 \oplus \Delta x_0.
\]

By the notations introduced before, we get
\[
\Delta S_2(\Delta k_1) \oplus \Delta S_1(\Delta k_2) = \Delta k_4. \quad (1)
\]
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 $2^{8}$ possible output differences for any fixed input difference.) To enumerate the possible pairs, we proceed as follows. First rewrite (1) as

$$\begin{pmatrix}
\Delta S_R(\Delta k_0^0) \\
\Delta S_R(\Delta k_1^0) \\
\Delta S_R(\Delta k_2^0) \\
\Delta S_R(\Delta k_3^0)
\end{pmatrix} = \begin{pmatrix}
\Delta S_Q(\Delta k_0^1) \\
\Delta S_Q(\Delta k_1^1) \\
\Delta S_Q(\Delta k_2^1) \\
\Delta S_Q(\Delta k_3^1)
\end{pmatrix} \oplus \begin{pmatrix}
\rho_{\text{mob}}^0 \\
\rho_{\text{mob}}^1 \\
\rho_{\text{mob}}^2 \\
\rho_{\text{mob}}^3
\end{pmatrix} \oplus MC_{-1}^{-1}
$$

where $\rho_{\text{mob}}^i (i = 0, 1, 2, 3)$ denotes a byte polynomial which contains only the most significant bits of all the four $\Delta S_Q$ 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 $\Delta S_Q(\Delta k_0^1)$, which has $2^{7}$ possibilities and three more bits for $\rho_{\text{mob}}^0$. Then we check whether the value computed at the right side of the equation is a correct value for $\Delta S_R(\Delta k_0^0)$. 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 $2^{5}$ 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 $2^{9} \cdot 2^{5} \cdot 2^{5} = 2^{24}$ solutions, which needs about $2 \times 2^{24} = 2^{25}$ words of memory. Now, the states of the FSM are as follows.

<table>
<thead>
<tr>
<th>clock t</th>
<th>$\Delta R_1$</th>
<th>$\Delta R_2$</th>
<th>$\Delta R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$\Delta k_1$</td>
<td>$\Delta k_3$</td>
</tr>
<tr>
<td>2</td>
<td>$\Delta k_2$</td>
<td>0</td>
<td>$(2^{24})$</td>
</tr>
<tr>
<td>3</td>
<td>$(2^{24})$</td>
<td>$(2^{24})$</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Each possible value of $\Delta R_3^2$ results in a possible value of $\Delta R_1^2$. At $t = 4$, we have

$$\Delta R_3^2 \oplus \Delta R_2^2 = \Delta c_2 \oplus \Delta c_{15} \oplus \Delta c_3 \oplus \Delta k_0.$$ 

Replacing the difference $\Delta R_3^2$ with the S-Box representation, we receive $\Delta R_3^2 \oplus \Delta S_1(\Delta R_3^2) = \Delta k_5$. Let $\Delta R_3^2 = c_0||c_1||c_2||c_3$, $\Delta R_2^2 = d_0||d_1||d_2||d_3$. Expanding this equation to the byte form, we get

$$\begin{pmatrix}
\Delta R_{out}(\Delta k_0^0) \\
\Delta R_{out}(\Delta k_1^0) \\
\Delta R_{out}(\Delta k_2^0) \\
\Delta R_{out}(\Delta k_3^0)
\end{pmatrix} = \begin{pmatrix}
\Delta S_{out}(\Delta k_0^1) \\
\Delta S_{out}(\Delta k_1^1) \\
\Delta S_{out}(\Delta k_2^1) \\
\Delta S_{out}(\Delta k_3^1)
\end{pmatrix} \oplus \begin{pmatrix}
\rho_{\text{mob}}^0 \\
\rho_{\text{mob}}^1 \\
\rho_{\text{mob}}^2 \\
\rho_{\text{mob}}^3
\end{pmatrix} \oplus MC_{-1}^{-1} \cdot \begin{pmatrix}
\Delta k_1^0 \\
\Delta k_1^1 \\
\Delta k_1^2 \\
\Delta k_1^3
\end{pmatrix}.$$ 

We have to insert all the $2^{24}$ possible pairs of $(\Delta R_3^2, \Delta R_2^2)$ and verify the value $\Delta S_R$ for the single bytes. This results in a time complexity of $2^{24}$. There are $2^{24} \cdot 2^{28} = 2^{50}$ entries satisfy this equation. This means we have $2^{20}$ sequences $(\Delta R_3^1, \Delta R_3^2, \Delta R_3^3, \Delta R_3^4, \Delta R_2^2)$ left. For each of them, we know the input-output difference of $S_1$ at clock 2 and 3. Thus, we can recover $(2 \cdot 2^{20} + 4 \cdot 2^{15}) = 16.51$ sorted pairs of values for $S_1$. This means that we have $10.61 \approx 8.255$ possible values for $\Delta R_3^1$. Looking at clock 5, we have $\Delta R_2^2 \oplus \Delta R_3^2 \oplus \Delta S_1(\Delta R_3^2) = \Delta k_5$. We can rewrite this equation into byte form and check the $2^{20}$ remaining sequences by the byte equations. There are $2^{20} \cdot 2^{25} \cdot 2^{28} \approx 2^{19.05}$ 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 $\sum_{i=0}^{9} 2^{20} \cdot (2^{27}/127)^i \cdot (2^{15}/127) = 2^{24.1}$ and the number of sequences left is $2^{20} \cdot (27/127)^{10} = 2^{10.5}$. 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 $2^{32} \cdot [2^{10} + 2^{24} + \sum_{i=0}^{9} (2^{20} \cdot (27/127)^i \cdot (31/127))] = 2^{57.1}$. 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

Now we look at the full SNOW 3G (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 $H_0$, 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 IV0 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 R1.

### 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.

<table>
<thead>
<tr>
<th>round</th>
<th>clock t</th>
<th>ΔR1</th>
<th>ΔR2</th>
<th>ΔR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>−1</td>
<td>Δd</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>Δd</td>
<td>?</td>
<td>0</td>
</tr>
</tbody>
</table>

From the keystream equation Δs0 = Δs31 + ΔR1 + ΔR2 + ΔR3, where ΔR1 = Δd, we get ΔR2, which gives us immediately ΔR2 and also ΔR3 from the next keystream equation. Therefore, we have only one known sequence (ΔR1 = Δd, ΔR2−1 = ΔR3−1 = 0, ΔR0 = Δd, R0, R1, R2, R3) and also ΔR2 from the next keystream equation. Now we know the input and output difference of S1: ΔR1−1 = Δd → S1. 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 · 216 = 24 steps. Afterward 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 210. We use the keystream equation of clock 12 to check the candidates. The total time complexity is 26 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.

<table>
<thead>
<tr>
<th>round</th>
<th>clock t</th>
<th>ΔR1</th>
<th>ΔR2</th>
<th>ΔR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>−2</td>
<td>Δd</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>−1</td>
<td>Δd</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

From Δs0 and ΔR0, we have

Δs0 = Δs31 + ΔR2−1 + Δs−1 + ΔR0 + Δs0,

which is

ΔR2−1 + ΔR2 = Δs0 + Δs31 + Δs−1 + Δs0.

Then we replace the differences at the left side with their S-Boxes description, denote the known part at the right side with k0 and get the equation

out ΔS1(Δd) + out ΔS1(Δd) = Δk0.

Multiplying by MC−1, we get the byte form equation

(ΔS1(Δd)) + (ΔS1(Δd)) = MC−1(Δk0).

We can check these four byte equations in 4 · 27 = 29 steps. The number of solutions will be 255·29 pairs of (ΔR2, ΔR3). We have 224 sequences (ΔR−1 = Δd, ΔR−2 = ΔR−2 = 0, ΔR−1 = Δd, ΔR−2, ΔR−1 = 0, ΔR−1, ΔR−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 S1: ΔR1−1 = Δd → S1 → ΔR21−1. The time complexity of this step is 23 · 27 = 28 steps. In the end, we have 224 · (237 · 27) = 210·24.45 difference sequences left. The memory complexity is 224·2 = 279·10. 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 229·10 = 229·10·2 in rounds. In this way, we have the same time complexity 229·10·2 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 ΔR2−2, which was fed back
into the LFSR, the remaining differences are either 0 or the known $\Delta d$. We guess the individual value $R_{1,a}^{-3}$ for the first pair $(K, IV_o)$ with complexity of $2^{32}$. From the value $R_{1,a}^{-3}$, we get with $\Delta R_{1}^{-3} = \Delta d$ the value $R_{1,b}^{-1}$ for the second pair $(K, IV_o)$. Furthermore we obtain $R_{2,a}^{-2}, R_{2,b}^{-2}, R_{3,a}^{-1}, R_{3,b}^{-1}$. We denote the known difference $\Delta R_{1}^{-2}$ with $\Delta k_0$, the linear dependent $\Delta R_{1}^{-1}$ with $\Delta k_1$ and $\Delta R_{3}^{-1}$ with $\Delta k_2$. This gives the following differences for the FSM:

<table>
<thead>
<tr>
<th>round</th>
<th>clock t</th>
<th>$\Delta R_1$</th>
<th>$\Delta R_2$</th>
<th>$\Delta R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>-3</td>
<td>$\Delta d$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>-2</td>
<td>$\Delta d$</td>
<td>$\Delta k_0$</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>-1</td>
<td>$\Delta k_1$</td>
<td>$\Delta k_2$</td>
<td>$\Delta k_2$</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

From $\Delta a^0 = \Delta s_1^0 \oplus \Delta s_2^0 \oplus \Delta s_3^0$, we insert the update equations for $\Delta R_1^2$ and $\Delta R_2^2$ and receive

$\Delta a^0 = \Delta s_1^{out} \oplus \Delta s_1^{out} \oplus \Delta s_5^{-1}
\oplus \Delta s_1^{out} \oplus \Delta s_1^{out} \oplus \Delta s_0^0$,

which gives

$\Delta s_1^{out} \oplus \Delta s_1^{out} \oplus \Delta s_1^{out} = \Delta a^0
\oplus \Delta s_2^0 \oplus \Delta s_3^0 \oplus \Delta s_5^{-1} \oplus \Delta s_0^0$.

We denote the known right part by $\Delta k_3$, multiply the equation with $MC_{1}^{-1}$ and rewrite it in byte notation as

$$\begin{bmatrix}
\text{out} \\ \text{out} \\ \text{out} \\ \text{out}
\end{bmatrix} \Delta S_k(\Delta a^0)
\oplus
\begin{bmatrix}
\text{out} \\ \text{out} \\ \text{out} \\ \text{out}
\end{bmatrix} \Delta S_k(\Delta d)
= MC_{1}^{-1}
\begin{bmatrix}
\Delta a^0 \\ \Delta k_3 \\ \Delta k_3 \\ \Delta k_3
\end{bmatrix}$$.

Then we check this equation line by line for each byte in $4 \cdot 2^4 = 2^{28}$ steps. The number of solutions will be $2^{28} \cdot 2^{28} = 2^{56}$ pairs of $(\Delta R_{1}^{-1}, \Delta R_{2}^{-1})$. Again, we switch from the differences of the FSM words to the individual values of them by using the input and output difference of $S_1$: $\Delta R_{1}^{-2} \rightarrow S_1 \rightarrow \Delta R_{3}^{-1}$. 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 $R_{1,a}^{-3}$, which gives

$2^{32} \left[2^9 + \sum_{i=0}^{9} \left(2^{24} \cdot \left(\frac{2^{27}}{1274} \cdot \frac{2^{31}}{1274} \cdot \frac{2^{94}}{1274} \cdot \frac{2^{10}}{274} \right)ight) \right] = 2^{60,2}$.

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 $\Delta d$ to only one byte in $IV_0$, we can reduce our attack complexity to $2^{22,7}$ 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 $\Delta R_{1}^{-3}$ and $\Delta R_{2}^{-2}$ were fed back into the LFSR, the remaining differences are either 0 or the known $\Delta d$. We guess the individual values of $R_{1,a}^{-4}$ and $R_{1,a}^{-3}$ for the first pair $(K, IV_o)$ with complexity of $2^{64}$. From the value $R_{1,a}^{-4}$ and $\Delta R_{1}^{-4}$ = $\Delta d$, we get the values of $R_{1,b}^{-1}, R_{2,a}^{-1}, R_{2,b}^{-1}, R_{3,a}^{-1}, R_{3,b}^{-1}$. Denote the known difference $\Delta R_{1}^{-2}$ by $\Delta k_0$, $\Delta R_{1}^{-1}$ by $\Delta k_1$ and $\Delta R_{3}^{-1}$ by $\Delta k_2$. From $R_{1,a}^{-4}$ and $\Delta R_{1}^{-3}$ = $\Delta d$, we get the values of $R_{1,b}^{-1}, R_{2,a}^{-2}, R_{2,b}^{-2}, R_{3,a}^{-1}, R_{3,b}^{-1}$. Again, we denote the now known difference $\Delta R_{1}^{-2}$ by $\Delta k_3$, $\Delta R_{1}^{-1}$ by $\Delta k_4$ and $\Delta R_{3}^{-1}$ by $\Delta k_5$. This gives the following differences for the FSM:

<table>
<thead>
<tr>
<th>round</th>
<th>clock t</th>
<th>$\Delta R_1$</th>
<th>$\Delta R_2$</th>
<th>$\Delta R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>-4</td>
<td>$\Delta d$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>-3</td>
<td>$\Delta d$</td>
<td>$\Delta k_0$</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>-2</td>
<td>$\Delta k_1$</td>
<td>$\Delta k_2$</td>
<td>$\Delta k_2$</td>
</tr>
<tr>
<td>14</td>
<td>-1</td>
<td>$\Delta k_4$</td>
<td>$\Delta k_3$</td>
<td>$\Delta k_3$</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

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 13 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. We can attack arbitrary many key/IV setup rounds of SNOW 3G 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 might succumb to chosen IV resynchronization attacks. The remaining
Table 1: The summary of our results on SNOW 3G.  

<table>
<thead>
<tr>
<th>Attack</th>
<th>Data</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNOW 3G(^{\oplus}) without feedback</td>
<td>(2^{33})</td>
<td>(2^{57.1})</td>
<td>(2^{25})</td>
</tr>
<tr>
<td>SNOW 3G(^{\oplus}) with feedback</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 rounds</td>
<td>24</td>
<td>(2^{10.1})</td>
<td>small</td>
</tr>
<tr>
<td>13 rounds with 1 byte difference (\Delta d)</td>
<td>24</td>
<td>(2^{10.1})</td>
<td>small</td>
</tr>
<tr>
<td>14 rounds with 1 byte difference (\Delta d)</td>
<td>24</td>
<td>(2^{42.7})</td>
<td>(2^{9})</td>
</tr>
<tr>
<td>15 rounds</td>
<td>24</td>
<td>(2^{92.2})</td>
<td>(2^{29.91})</td>
</tr>
<tr>
<td>16 rounds</td>
<td>24</td>
<td>(2^{124.2})</td>
<td>(2^{29.91})</td>
</tr>
</tbody>
</table>

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

REFERENCES


APPENDIX

We want to simplify the equation

\[ \Delta S_i(\Delta k_1) \oplus \Delta S_i(\Delta k_2) = \Delta k_4. \]

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

\[ MC_2 \cdot \left( \begin{array}{c} \Delta S_Q(\Delta k_1^0) \\ \Delta S_Q(\Delta k_1^1) \\ \Delta S_Q(\Delta k_1^2) \end{array} \right) \oplus MC_1 \cdot \left( \begin{array}{c} \Delta S_R(\Delta k_2^0) \\ \Delta S_R(\Delta k_2^1) \\ \Delta S_R(\Delta k_2^2) \end{array} \right) = \left( \begin{array}{c} \Delta k_1^0 \\ \Delta k_1^1 \\ \Delta k_1^2 \end{array} \right). \]

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

\[ MC_1^{-1} \cdot MC_2 \cdot \left( \begin{array}{c} \Delta S_Q(\Delta k_1^0) \\ \Delta S_Q(\Delta k_1^1) \\ \Delta S_Q(\Delta k_1^2) \end{array} \right) \oplus \left( \begin{array}{c} \Delta S_R(\Delta k_2^0) \\ \Delta S_R(\Delta k_2^1) \\ \Delta S_R(\Delta k_2^2) \end{array} \right) = MC_1^{-1} \cdot \left( \begin{array}{c} \Delta k_1^0 \\ \Delta k_1^1 \\ \Delta k_1^2 \\ \Delta k_1^3 \end{array} \right). \]

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 \(\Delta S_Q(\Delta k_1^0)\) and a byte polynomial containing only the most significant bits of all four \(\Delta S_Q\) values. We denote this polynomial with \(p_0^{\text{mb}}\). The other three rows have similar structures, but with different polynomials \(p_i^{\text{mb}}\) \((i = 1, 2, 3)\). Therefore we can rewrite the equation to

\[ \left( \begin{array}{c} \Delta S_R(\Delta k_2^0) \\ \Delta S_R(\Delta k_2^1) \\ \Delta S_R(\Delta k_2^2) \end{array} \right) = \left( \begin{array}{c} \Delta S_Q(\Delta k_1^0) \\ \Delta S_Q(\Delta k_1^1) \\ \Delta S_Q(\Delta k_1^2) \end{array} \right) \oplus \left( \begin{array}{c} p_0^{\text{mb}} \\ p_1^{\text{mb}} \\ p_2^{\text{mb}} \end{array} \right) \oplus MC_1^{-1} \cdot \left( \begin{array}{c} \Delta k_1^0 \\ \Delta k_1^1 \\ \Delta k_1^2 \\ \Delta k_1^3 \end{array} \right). \]

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

\[ p_{0}^{\text{msb}} = (m_1 \oplus m_3)x^7 + (m_0 \oplus m_1)x^6 + (m_2 \oplus m_3)x^5 \]
\[ + (m_1 \oplus m_2)x^4 + (m_0 \oplus m_2)x^2 + (m_1 \oplus m_2)x \]
\[ + (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \]

\[ p_{1}^{\text{msb}} = (m_0 \oplus m_2)x^7 + (m_1 \oplus m_2)x^6 + (m_0 \oplus m_3)x^5 \]
\[ + (m_2 \oplus m_3)x^4 + (m_1 \oplus m_3)x^2 + (m_2 \oplus m_3)x \]
\[ + (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \]

\[ p_{2}^{\text{msb}} = (m_1 \oplus m_3)x^7 + (m_2 \oplus m_3)x^6 + (m_0 \oplus m_1)x^5 \]
\[ + (m_0 \oplus m_3)x^4 + (m_0 \oplus m_2)x^2 + (m_0 \oplus m_3)x \]
\[ + (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \]

\[ p_{3}^{\text{msb}} = (m_0 \oplus m_2)x^7 + (m_0 \oplus m_3)x^6 + (m_1 \oplus m_2)x^5 \]
\[ + (m_0 \oplus m_1)x^4 + (m_1 \oplus m_3)x^2 + (m_0 \oplus m_1)x \]
\[ + (m_0 \oplus m_1 \oplus m_2 \oplus m_3) \]