Fault analysis and weak key-IV attack on Sprout

Dibyendu Roy and Sourav Mukhopadhyay
Department of Mathematics,
Indian Institute of Technology Kharagpur,
Kharagpur-721302, India
{dibyendu.roy,sourav}@maths.iitkgp.ernet.in

Abstract:- Armknecht and Mikhalev proposed a new stream cipher 'Sprout' based on the design specification of the stream cipher, Grain-128a. Sprout has shorter state size than Grain family with a round key function. The output of the round key function is XOR'ed with the feedback bit of the NFSR of the cipher. In this paper, we propose a new fault attack on Sprout by injecting a single bit fault after the key initialization phase at any arbitrary position of the NFSR of the cipher. By injecting a single bit fault, we recover the bits of the secret key of the cipher by observing the normal and faulty keystream bits at certain clockings of the cipher. By implementing the attack, we verify our result for one particular case. We also show that the Sprout generates same states for several rounds in key initialization phase for two different key-IV pairs, which proves that the key initialization round is having very poor period.

Keywords:- Boolean function, Sprout, Fault attack, Weak key-IV.

1 Introduction

The design specification of Sprout [2] was proposed by Armknecht and Mikhalev at FSE, 2015. Sprout is based on the design specification of the Grain family of stream ciphers [5], [6], [1] mainly on Grain 128a [1]. This new cipher is based on one 40 bit LFSR, one 40 bit NFSR, one nonlinear filter function, one counter function and one round key function. The nonlinear filter functions of Sprout and Grain-128a are same. Interesting part of the design specification of Sprout is that first time round key function has been introduced in any stream cipher.

In the design specification of Sprout, designers have claimed many security aspects. But, recently in many literatures many authors have shown some significant security flaws in the design specification of the cipher. Recently, Maitra et al. [9] have proposed a key recovery attack on this new stream cipher. They recovered 40 bits of the NFSR and few bits of the LFSR from the given 850 keystream bits. They also proposed a fault attack on this cipher. Within a feasible time they are able to recover the secret key bits of the cipher by using some statistical observation of the normal and faulty keystream bits. Also they found some drawbacks of the cipher which were claimed by the designers. In the same year Banik [3] has found some security flaws in this cipher. He has obtained some better result than the Maitra et al. The main idea of both the paper was to recover the secret key by using some known state bits of the NFSR and LFSR at certain clockings. Banik [3] has shown that the keystream generation phase of the cipher is having very poor period. He has shown that there exists weak key-IV pair for which the initial state of the keystream generation phase and

the 80-th state of the same phase of the cipher become same. From this result we can observe that keystream generation phase of the cipher is having small period.

In Crypto 2015, Lallemand and Naya-Plasencia [8] have proposed a cryptanalysis on full round of Sprout. Although the idea of their attack is very complicated but the attack has practical complexity. One nice idea of their method of attack is that even if the cipher has the round key function but still anybody can find some keystream bits only by knowing full state of the cipher at certain clocking. Also they used some sieving techniques to reduce the search space of the secret key.

The main idea of the fault attack is to first inject a fault into the cipher and then observe the normal & faulty keystream bits to recover the secret information of the cipher. Using the fault attack technique many authors have proposed attack on many ciphers like as Grain 128, Trivium etc. In 2011, Karmakar et al. [7] proposed fault attack on Grain 128 stream cipher by targeting the NFSR. Also in 2012, Banik et al. [4] proposed differential fault attack on Grain family of stream cipher. Recently Maitra et al. [9] proposed a fault attack on Sprout. They proposed signature based fault attack on this cipher, which is based on some statistical observations in the keystream bits. After injecting a fault into the cipher (either NFSR or LFSR) they observed some statistical properties of the normal & faulty keystream bits, and from there they first detect the position of the fault and then recovered secret key bits.

In this paper we propose a new fault attack on Sprout by injecting a single bit fault into the NFSR of the cipher. In our fault attack we inject a single bit fault into any arbitrary position of the NFSR in the keystream generation phase. After that we detect the position and propagation of the fault by observing the normal & faulty keystream bits. In the next phase of the attack we recover the secret key bits by observing the difference between normal and faulty keystream bits. The detail procedure of the attack is described in Section 4. The main difference between our work and Maitra et al. [9] work is that after injecting a single bit fault we observe some simple deterministic properties on the normal and faulty keystream bits to recover the secret key bits, where as they used signature based approach. Because of this reason we need less number of keystream bits to mount the attack. In this paper we also find two weak key-IV pairs of Sprout for which the cipher generates same state in the key initialization phase and the state is null state.

The rest of the article is organized as follows: In Section 2 we describe some basic definitions. The design specification of the cipher is given in Section 3. Our new fault attack is described in Section 4. Weak key-IV attack is presented in Section 5. Finally the paper is concluded in Section 6.

2 Definitions

Definition 1. ***Boolean function***
Boolean function of n variables is a mapping from $\{0,1\}^n$ *to* $\{0,1\}$.
Definition 2. ***Algebraic normal form of a Boolean function***
The algebraic normal form of f involving n variables is given by the following expression,

$$f(x_1,....,x_n) = a_0 \oplus \bigoplus_{1\leq i\leq n} a_i x_i \oplus \bigoplus_{1\leq i<j\leq n} a_{ij}x_i x_j \oplus ..... \oplus a_{12...n}x_1x_2....x_n,$$

where all the coefficients belongs to $\{0,1\}$.
Definition 3. ***Degree of Boolean function***
Degree of a Boolean function f is defined as the number of variables in the highest order product

term in the algebraic normal form of f. Functions of degree at most one are called affine function. An affine function with constant term equal to zero is called linear function.

3 Design specification of Sprout

In FSE 2015, Armknecht and Mikhalev [2] proposed a new light weight stream cipher based on the design specification of the Grain family of stream cipher, mainly on Grain 128a [1] named as Sprout. Sprout is based on one 40 bit NFSR, one 40 bit LFSR, one nonlinear filter function, one counter function and one round key function. Through out the article we use the following notations:

- t denotes the clocking number.
- $L_t = (l_t, l_{t+1}, ..., l_{t+39})$ denotes the state of the LFSR at the t^{th} clocking.
- $N_t = (n_t, n_{t+1}, ..., n_{t+39})$ denotes the state of the NFSR at the t^{th} clocking.
- $C_t = (c_t^0, c_t^1, ..., c_t^8)$ denotes the state of the counter at the t^{th} clocking.
- $k = (k_0, k_1, ..., k_{79})$ denotes the secret key of the cipher.
- $iv = (iv_0, iv_1, ..., iv_{79})$ denotes the initialization vector of the cipher.
- k_t^* denotes the round key at t^{th} clocking.
- z_t denotes the keystream bits.

The primitive polynomial corresponding to the linear feedback function of the LFSR is given by,

$$f(x) = x^{40} + x^{35} + x^{25} + x^{20} + x^{15} + x^6 + 1.$$

The main difference between the nonlinear feedback functions of Grain-128a and Sprout is that in case of Sprout the output of the round key function and the output of the counter function are involved in the feedback function of the NFSR, where as, Grain-128a does not have any counter function and round key function. The algebraic normal form of the nonlinear feedback function of the NFSR of Sprout is given by,

$$\begin{aligned} n_{t+40} =& k_t^* + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} + n_{t+39} + n_{t+2}n_{t+25} + n_{t+3}n_{t+5} + n_{t+7}n_{t+8} \\ &+ n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} + n_{t+26}n_{t+32} + n_{t+33}n_{t+36}n_{t+37}n_{t+38} \\ &+ n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31}. \end{aligned}$$

Counter function of the cipher is based on 9 bits. First 7 bits computes the index of the secret key bit which is going to be involved in the current clocking. Other two bits are used in the key initialization phase. In each clocking the round key function outputs one bit, which depends on the secret key bit. The expression of the round key function is given by the following function,

$$k_t^* = \begin{cases} k_t & 0 \leq t \leq 79 \\ (k_{t \mod 80}) \cdot (l_4 + l_{21} + l_{37} + n_9 + n_{20} + n_{29}) & t \geq 80 \end{cases}$$

The nonlinear filter function of Sprout [2] and Grain-128a [1] are same. It is a function of 9 variables. Among which 7 variables are from the LFSR state variables and 2 variables are from the

NFSR state variables. The algebraic normal form of the nonlinear filter function $h(\cdot)$ is given by the following expression,

$$h(\cdot) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8,$$

where the variables $x_0, x_1, ...x_8$ correspond to the state variables n_{t+4}, l_{t+6}, l_{t+8}, l_{t+10}, l_{t+32}, l_{t+17}, l_{t+19}, l_{t+23}, and n_{t+38}, respectively. The expression of the keystream bits of the cipher is given by the following expression,

$$z_t = h(\cdot) + l_{t+30} + \sum_{j \in B} n_{t+j},$$

where $B = \{1, 6, 15, 17, 23, 28, 34\}$.

The cipher is based on two phases. First phase is key initialization phase and the second phase is keystream generation phase. In the key initialization phase, the cipher will be initialized by the secret key and the initialization vector. The first 40 bits of the NFSR is filled by the first 40 bits of the initialization vector IV, by the process $n_i = iv_i$, $0 \leq i \leq 39$ and the remaining 30 bits of the initialization vector is loaded in the first 30 position of the LFSR, by the process $l_{i-40} = iv_i$, $40 \leq i \leq 69$. The remaining positions of the LFSR are filled by 0 or 1 bits by the process $l_{30} = l_{31} = ... = l_{38} = 1$ and $l_{39} = 0$. Then the cipher is clocked for 320 times without producing any keystream bits instead the output is XORed with the feedback bits of the LFSR and NFSR. i.e. the expressions of the feedback bits will be, $l_{t+40} = z_t + f(L)$ and $n_{t+40} = z_t + k_t^* + l_t + c_t^4 + g(N_t)$. Detailed design specification of the cipher is given in the following figure 1.

After the key initialization phase is over, the cipher will start producing the keystream bits. The detailed design specification of this cipher is available in [2].

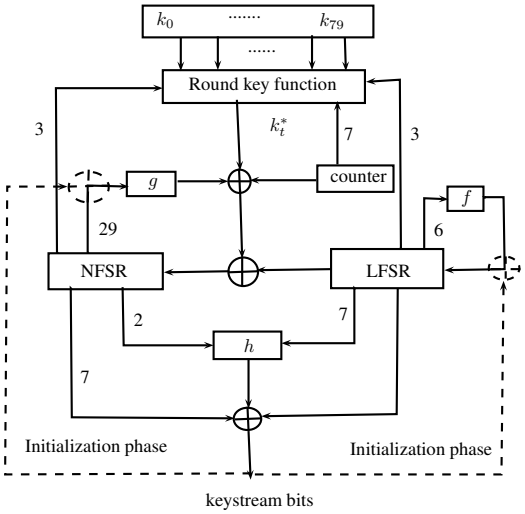

Figure 1: Design specification of Sprout

4 Fault attack on Sprout

In this section we discuss about our new fault attack on Sprout. Before describing the attack process we first assume that the attacker has following freedom:

- after the key initialization phase attacker can inject a single bit fault at any position of the NFSR.
- also attacker can reset the cipher into the normal position after injecting the fault.

Injecting a fault at any position means change the corresponding bit to its compliment. After injecting a single bit fault to any position of the NFSR of the cipher, the injected fault will start propagate and it may or may not affect the keystream bits depending upon the position of the fault. In general there are two phases in the fault attack first one is detection of the fault position and second one is finding the secret information from the normal & faulty keystream bits.

In the process of fault detection phase, the attacker observes the difference between the normal and faulty keystream bits at different clockings. In the second phase also attacker finds some equations involving the secret informations using the difference between the normal & faulty keystream bits. As the design specification of Sprout is different than the usual NFSR based cipher (like as Grain), so the key recovery phase will also be different from the other ciphers.

Now we describe our fault attack on Sprout. At first attacker injects a fault at any arbitrary position of the NFSR of the cipher. After injecting a single bit fault at any arbitrary position of the NFSR, the attacker will start observing the normal and faulty keystream bits of the cipher. The first aim of the attacker is to detect the position of the fault at certain clocking.

Fault detection phase: After injecting a single bit fault at any arbitrary position (initially unknown to the attacker) of the NFSR of the cipher, the first task of the attacker is to detect the position of the fault by observing the normal and faulty keystream bits. Now from the design specification of Sprout we observe that the expression of the keystream bit is $z_t = h(\cdot) + l_{t+30} + \sum_{j \in B} n_{t+j}$, where $B = \{1, 6, 15, 17, 23, 28, 34\}$. We denote the normal keystream bit by z_t and faulty keystream bit by $\tilde{z}_t$. Now we can easily observe that if fault presents at n_{t+1} at t^{th} clocking then the value of $z_t + \tilde{z}_t$ will be 1. So its very trivial to observe that the cipher will output a faulty keystream bit if fault arrives at any one position of the seven bits which are involved linearly in the expression of the keystream bit. In these cases the value of $z_t + \tilde{z}_t$ will always be 1. Suppose after injecting a single bit fault at any arbitrary position of the NFSR if we observe that the cipher outputs the first faulty keystream bit after 2 clocking, another faulty keystream bits after 5 clockings, then after 6 clockings. Then we can say that initially the fault was at 30^{th} position of the NFSR. So from the pattern of the value of $z_t + \tilde{z}_t$ we can detect the position of the injected fault in the NFSR at certain clocking. After injecting the fault, it will start moving through out the NFSR and also start generating some faulty feedbacks into the NFSR. By following the previously described procedure we can detect that whether the feedback bit into the NFSR at certain clocking is faulty or not (by observing the value of $z_m + \tilde{z}_m$ at certain clocking m, where in the expression of keystream bit the required feedback bit is present as linearly). So from the pattern of the values of $z_t + \tilde{z}_t$ for different clockings we can detect the position of the fault and also can determine whether the feedback bit corresponding to one certain clocking is faulty or not. The following algorithm presents the general procedure to detect the position of the injected fault. The following algorithm states the general procedure to recover the secret key bits from normal & faulty keystream bits.

Algorithm 1 Fault position detection

Input: Normal and faulty keystream bits.

Output: Position of the fault.

1: Fault location $F = \{\}$.
2: Observe the value $\Delta_t = z_t + \tilde{z}_t$ for several clockings.
3: Store the values of Δ_t and t, corresponding to several clockings t.
4: **if** Δ_t follows particular pattern (as previously mentioned)
determine the fault position f and update $F = F \cup \{(f, clocking\ number)\}$.
5: **return** The set F corresponding to different clockings.

Secret key recovery phase: In this phase after injecting a single bit fault at any arbitrary position of the NFSR of the cipher the attacker will try to find out the secret key bits by observing the normal and faulty keystream bits. Fault detection phase must be performed before secret key recovery phase. To find the secret key; attacker needs to know the position of the fault.

Now the expression of the round key bit of the cipher in the keystream generation phase is $k_t^* = (k_{t \mod 80}) \cdot (l_{t+4} + l_{t+21} + l_{t+37} + n_{t+9} + n_{t+20} + n_{t+29})$. Now if fault is at n_9 then the expression of the round key function will be $\tilde{k_t^*} = (k_{t \mod 80}) \cdot (l_{t+4} + l_{t+21} + l_{t+37} + n'_{t+9} + n_{t+20} + n_{t+29})$, as $t > 80$. The expression of the normal feedback bit was,

$$\begin{aligned} n_{t+40} =& k_t^* + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} + n_{t+39} + n_{t+2}n_{t+25} + n_{t+3}n_{t+5} + n_{t+7}n_{t+8} \\ &+ n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} + n_{t+26}n_{t+32} + n_{t+33}n_{t+36}n_{t+37}n_{t+38} \\ &+ n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31}. \end{aligned}$$

And the expression of the faulty feedback bit will be,

$$\begin{aligned} \tilde{n}_{t+40} =& \tilde{k_t^*} + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} + n_{t+39} + n_{t+2}n_{t+25} + n_{t+3}n_{t+5} + n_{t+7}n_{t+8} \\ &+ n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} + n_{t+26}n_{t+32} + n_{t+33}n_{t+36}n_{t+37}n_{t+38} \\ &+ n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31}. \end{aligned}$$

Now if we XOR the normal and faulty feedback bit we get,

$$\begin{aligned} n_{t+40} + \tilde{n}_{t+40} &= k_t^* + \tilde{k_t^*} \\ \Rightarrow n_{t+40} + \tilde{n}_{t+40} &= (k_{t \mod 80}) \cdot (n_{t+9} + n'_{t+9}) \\ \Rightarrow n_{t+40} + \tilde{n}_{t+40} &= k_{t \mod 80}. \end{aligned}$$

Similarly if fault presents at n_{t+20}, n_{t+29} we can still recover one secret key bit. The value of $n_{t+40} + \tilde{n}_{t+40}$ can be found by observing the normal and faulty keystream bits of the cipher (method is already described). So, from the known value of $n_{t+40} + \tilde{n}_{t+40}$ we can get the value of one secret key bit $k_{t \mod 80}$. Now suppose fault presents at any other bit except n_{t+9}, n_{t+20} or n_{t+29}, then when the fault reaches to any of these positions we can get value of one secret key bit. For example fault is at n_{t+29}, so from there we can automatically recover one secret key bit. When fault moves to 20^{th} position again, we can recover another secret key bit. In this way we can recover several secret key bits by injecting a single bit fault into the NFSR in the keystream generation phase.

So from the above discussion we observe that if fault presents at any of the NFSR bit then we can recover one secret key bit by observing the normal and faulty keystream bits. We have observed that

if fault presents at any of the NFSR bits involved in the round key function then, $n_{t+40} + \tilde{n}_{t+40} = k_{t \mod 80}$. Now if $k_{t \mod 80} = 0$ then $n_{t+40} + \tilde{n}_{t+40} = 0$ that means that the feedback value is fault free. Now if $k_{t \mod 80} = 1$ then $n_{t+40} + \tilde{n}_{t+40} = 1$ it implies that the feedback is faulty. If the feedback bit is faulty then it will start propagating into the NFSR and after certain number of clockings this faulty feedback will again affect the value of round key bit and from there again we can recover another secret key bit. By following this procedure we can recover the secret key bits very easily, so we can continue the procedure until we can recover the full secret key. The faulty nature of the feedback bit depends on the secret key bits. It may happen that the whole state of the NFSR becomes fault free after certain number of clockings. In that case, we need to inject a new single bit fault into the NFSR to recover the secret key bits. As the faulty nature of the feedback bit depends on the secret key bits, so the total number of fault required to recover the full secret key depends on the corresponding secret key. The following algorithm is the general procedure to recover the secret key bits from the normal and faulty keystream bits.

Algorithm 2 Recover the secret key bit

Input: Normal and faulty keystream bits and position of the fault.
Output: Secret key bit.

1: Observe the position of the fault at certain clocking.
2: **if** position of the fault at desired position
calculate the value of $n_{t+40} + \tilde{n}_{t+40} = \Delta_m = z_m + \tilde{z}_m$, for certain clocking m.
3: **return** $k_{t \mod 80}$.

Implementation:- To implement our attack model we have used Sage [11]. We have assumed that for one *key IV* pair we will get 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1 this state after the key initialization phase, and also we have assumed that fault is injected just after the key initialization phase (at the first clocking). We have observed 25 normal and faulty keystream bits after injecting the fault into the NFSR. The following table gives the details of the position of the fault (at the first clocking after the key initialization phase) and the corresponding recovered secret key bits.

Here k_i denotes the secret key bits. In the table 1 we have shown only the location of the injected fault and the corresponding recovered secret key bits, when the fault is injected at the first clocking of the cipher just after the key initialization phase. Similar implementation can be done for the other clockings by observing many normal and faulty keystream bits.

From the table 1 we can observe that we are able to recover 19 secret key bits $k_0, ..., k_{18}$ directly by injecting faults at 40 different positions of the NFSR and observing normal and faulty keystream bits for first 25 clockings. We also obtain many equations involving the secret key bits $k_0, ..., k_{24}$. As we are able to recover $k_0, ..., k_{18}$ directly by observing normal and faulty keystream bits of the cipher, then we substitute the values of these bits $k_0, ..., k_{18}$ into the obtained equations to recover other secret key bits $k_{19}, ..., k_{24}$. So, after injecting 40 single bit faults at different positions of the NFSR in the first clocking of keystream generation phase of Sprout, we are able to recover 25 secret key bits. Now we reset the cipher into its normal position.

After reseting the cipher into its normal position we start injecting single bit faults into the cipher at the 26-th clocking of the cipher. We do the same thing here, we observe 25 normal and faulty keystream bits corresponding to different fault positions. By following the same procedure (previ-

Fault location	secret key bits obtained	Fault location	secret key bits obtained
40	k_{10}, k_{11}, k_{12}	39	k_9
38	k_8, k_{12}, k_{14}	37	k_7, k_{12}, y_{13}
36	k_6, k_{11}, k_{12}	35	k_5, k_{13}, k_{15}
34	k_4, k_{11}, k_{14}	33	k_3, k_{11}
32	k_2, k_{11}	31	k_1, k_{10}, k_{15}
30	k_0, k_9	29	$k_8, k_{13}, k_{15}, k_{16}$
28	k_7, k_{13}, k_{16}	27	k_6, k_{12}, k_{14}
26	$k_5, k_{12}, k_{13}, k_{14}, k_{15}$	25	$k_4, k_{11}, k_{12}, k_{13}, k_{14}$
24	k_3, k_{12}, k_{13}	23	k_2, k_{11}
22	$k_1, k_{11}, k_{14}, k_{15}$	21	k_0, k_{13}
20	$k_{10}, k_{11}, k_{13}, k_{14}, k_{15}$	19	k_9, k_{13}, k_{14}
18	$k_8, k_{12}, k_{13}, k_{14}, k_{18}$	17	k_7, k_{13}, k_{15}
16	k_6, k_{12}	15	k_5, k_{12}
14	$k_4, k_{11}, k_{12}, k_{13}, k_{14}$	13	k_3, k_{13}
12	k_2, k_{12}	11	k_1, k_{17}
10	k_0, k_{13}	9	$k_{11}, k_{12}, k_{14}, k_{15}$
8	$k_{11}, k_{12}, k_{13}, k_{17}$	7	k_{12}, k_{13}, k_{14}
6	$k_{11}, k_{12}, k_{13}, k_{14}, k_{15}$	5	k_{13}, k_{14}
4	k_{11}, k_{13}, k_{16}	3	$k_{11}, k_{12}, k_{13}, k_{17}$
2	k_{12}, k_{13}	1	$k_{11}, k_{12}, k_{14}, k_{15}$

Table 1: Fault location vs secret key bit obtained

ously described) we will able to recover more 25 secret key bits $k_{25}, ..., k_{49}$. We will able to recover the secret key bits $k_{25}, ..., k_{49}$, because these secret key bits are involved in the round key function for these 25 clockings of the keystream generation phase. So, 40 single bit faults helps us to recover another 25 secret key bits. Again we reset the cipher in to normal position and do the same thing form 51-th clocking. By doing same process for next 25 clockings (from 51-th clocking) we will able to recover the secret key bits $k_{50}, ..., k_{74}$. Finally, we are able to recover 75 secret key bits by injecting 120 faults. For the remaining 5 secret key bits we do the same thing. In this case, if we consider the average case then we need average $\frac{5}{75} \times 120 = 8$ faults. Hence, to recover the full set of secret key bits we need on an average 128 many faults.

5 Weak key-IV pair of Sprout

In this section we discuss about weak key-IV pair of Sprout [2]. To find weak key-IV pair our main aim was to find at least two key-IV pairs for which the cipher will generate same state in the key initialization phase. Our main aim is to make the whole state of the cipher to zero after certain number of clockings by starting with some non-trivial key-IV pair. Let us choose the secret keys are of the form $K = (k_0, ..., k_{39}, k_{40}, ..., k_{79})$, and the initialization vectors are of the form $IV = (iv_0, iv_1, ..., iv_{79})$. With this we start the key initialization phase of the cipher. We put the secret key in the round key function and put the initialization vector into the NFSR and LFSR ($n_i = iv_i$, $0 \leq i \leq 39$ and $l_{i-40} = iv_i$, $40 \leq i \leq 69$). Now our next target is to make whole state of the cipher to a null state (i.e. all state bits are 0). For this key-IV pair the expression of the

output bit will be,

$$\begin{aligned} z_t &= h(\cdot) + l_{t+30} + \sum_{j\in B} n_{t+j} \\ &= n_{t+4}l_{t+6} + l_{t+8}l_{t+10} + l_{t+32}l_{t+17} + l_{t+19}l_{t+23} + n_{t+4}l_{t+32}n_{t+38} + l_{t+30} + \sum_{j\in B} n_{t+j}. \end{aligned}$$

The expression of the feedback bit of the NFSR will be,

$$\begin{aligned} n_{t+40} =& z_t + k_t + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} + n_{t+39} + n_{t+2}n_{t+25} + n_{t+3}n_{t+5} + n_{t+7}n_{t+8} \\ &+ n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} + n_{t+26}n_{t+32} + n_{t+33}n_{t+36}n_{t+37}n_{t+38} \\ &+ n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31} \end{aligned}$$

$$\begin{aligned} n_{t+40} =& n_{t+4}l_{t+6} + l_{t+8}l_{t+10} + l_{t+32}l_{t+17} + l_{t+19}l_{t+23} + n_{t+4}l_{t+32}n_{t+38} + l_{t+30} + n_{t+1} + n_{t+6} \\ &+ n_{t+15} + n_{t+17} + n_{t+23} + n_{t+28} + n_{t+34} + k_t + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} \\ &+ n_{t+39} + n_{t+2}n_{t+25} + n_{t+3}n_{t+5} + n_{t+7}n_{t+8} + n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} \\ &+ n_{t+26}n_{t+32} + n_{t+33}n_{t+36}n_{t+37}n_{t+38} + n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31}. \end{aligned}$$

Similarly, the expression of the feedback bit of the LFSR will be,

$$\begin{aligned} l_{t+40} &= z_t + f(L) \\ &= l_t + l_{t+5} + l_{t+15} + l_{t+20} + l_{t+25} + l_{t+34} + n_{t+4}l_{t+6} + l_{t+8}l_{t+10} + l_{t+32}l_{t+17} + l_{t+19}l_{t+23} \\ &+ n_{t+4}l_{t+32}n_{t+38} + l_{t+30} + n_{t+1} + n_{t+6} + n_{t+15} + n_{t+17} + n_{t+23} + n_{t+28} + n_{t+34}. \end{aligned}$$

Intentionally we make both the feedback bit to 0; i.e., $n_{t+40} = 0$ and $l_{t+40} = 0$ for the first 40 key initialization rounds. So in each key initialization rounds we have two simultaneous equations,

$$\begin{aligned} &n_{t+4}l_{t+6} + l_{t+8}l_{t+10} + l_{t+32}l_{t+17} + l_{t+19}l_{t+23} + n_{t+4}l_{t+32}n_{t+38} + l_{t+30} + n_{t+1} + n_{t+6} + n_{t+15} + n_{t+17} \\ &+ n_{t+23} + n_{t+28} + n_{t+34} + k_t + l_t + c_t^4 + n_t + n_{t+13} + n_{t+19} + n_{t+35} + n_{t+39} + n_{t+2}n_{t+25} \\ &+ n_{t+3}n_{t+5} + n_{t+7}n_{t+8} + n_{t+14}n_{t+21} + n_{t+16}n_{t+18} + n_{t+22}n_{t+24} + n_{t+26}n_{t+32} \\ &+ n_{t+33}n_{t+36}n_{t+37}n_{t+38} + n_{t+10}n_{t+11}n_{t+12} + n_{t+27}n_{t+30}n_{t+31} = 0 \end{aligned} \tag{1}$$

and

$$\begin{aligned} &l_t + l_{t+5} + l_{t+15} + l_{t+20} + l_{t+25} + l_{t+34} + n_{t+4}l_{t+6} + l_{t+8}l_{t+10} + l_{t+32}l_{t+17} + l_{t+19}l_{t+23} \\ &+ n_{t+4}l_{t+32}n_{t+38} + l_{t+30} + n_{t+1} + n_{t+6} + n_{t+15} + n_{t+17} + n_{t+23} + n_{t+28} + n_{t+34} = 0. \end{aligned} \tag{2}$$

By imposing these two conditions we can construct 80 equations involving $k_0, ..., k_{39}$ and $iv_0, ..., iv_{79}$. As we are making the feedback bits of the NFSR to 0 for the first 40 key initialization rounds, so after 40 key initialization rounds the state of the NFSR will be null. Also we are making the feedback bits of the LFSR to 0 for first 40 key initialization rounds but it has been observed that for 40^{th} key initialization round the feedback bit corresponding to the LFSR is becoming 0 (independent of the variables). After 40 key initialization rounds the state of the NFSR will be $N^{40} = (0, 0, ..., 0)$ and the state of the LFSR will be $L^{40} = (0, ..., 0, ..., 0)$. Now we consider 40 equations corresponding to $n_{t+40} = 0$ and 39 equations corresponding to $l_{t+40} = 0$ (we are not considering the 40^{th} feedback bit equation as it is constant and value is 0). After constructing the final system we solve the

system to get the values of $k_0, ..., k_{39}$ and $iv_0, ..., iv_{79}$. Now if this system has multiple solutions then we can conclude that there exist two pair of (K, IV) for which we will have same state after 40 key initialization rounds and the state will be $N^{40} = (0, 0, ..., 0)$ and $L^{40} = (0, ..., 0, ..., 0)$. Now depending upon the choice of the last 40 bits of both the secret keys the state of the cipher will change, if we assume the last 40 bits are 0 then the state bits of the cipher for next 40 rounds will depend only on the output of the counter function (which is public). Now if we assume that the last 40 bits of both the secret keys are same and all the bits are equal to the output of the counter function (which is public) for the next 40 key initialization rounds, then the state of the cipher will remain null for next 40 rounds, as L^{40} and N^{40} are null and the value of the feedback bits will be $k_t + c_t^4 = 0$ for these 40 key initialization rounds. If we assume that the last 40 bits of both the secret keys are same then also we can find collision for next 40 rounds, as L^{40} and N^{40} are null and last 40 bits of both the secret keys are same. From this observation we can claim that the key initialization phase of the cipher is having very poor period.

Firstly we constructed these types of equations in SAGE [11] and then passed the final system in to SAT solver [10] to get the values corresponding to $k_0, ..., k_{39}$ and $iv_0, ..., iv_{79}$. If this system has at least two solutions then we can say that there are two key-IV pairs (K_1, IV_1) and (K_2, IV_2) for which the cipher generates same state (null state) after the 40 key initialization rounds. It has been observed that this system has multiple solutions. The first 40 bits of K_1 is,
0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1
and the corresponding IV_1 is,
IV_1 =(0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0).

The first 40 bits of K_2 is,
1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1
and the corresponding IV_2 is,
IV_2 =(1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0).

Let K' be the last 40 bits of both the secret keys then the weak key-IV pairs (K_1, IV_1) and (K_2, IV_2) will be,
K_1 =(0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, K'),
IV_1 =(0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0),
and
K_2 =(1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, K'),
IV_2 =(1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0).

For this (K_1, IV_1) and (K_2, IV_2) pairs the cipher produces same state after 40 key initialization rounds and we also observe that depending upon different choices of the last 40 bits of both the secret keys the state collision will continue for next 40 rounds. These key-IV pairs (K_1, IV_1) and (K_2, IV_2) are weak pairs. From the above obtained result we can say that the key initialization

round is having very bad period (as states are colliding for several rounds). In [3] Banik has shown that keystream generation phase is having poor period, from our obtained result we can say that even key initialization phase also has very poor period.

6 Conclusion

In this paper we have proposed a new fault attack on the stream cipher Sprout. By injecting a single bit fault into any arbitrary position of the NFSR we are able to recover some secret key bits. To recover the secret key bits in our fault attack model we have followed some deterministic observation in the normal and faulty keystream bits of the cipher. By injecting a single bit fault into any arbritary position of the NFSR of the cipher we are able to recover some secret key bits by observing the normal & faulty keystream bits. We have observed that the faulty nature of the feedback bit of the NFSR is depending on the secret key bits. We will able to recover the secret key bits until the state bits of the NFSR remain faulty. We have also verified our result by implementing the attack for one particular case. We have also found two weak key-IV pairs for which the cipher generates same states in the key initialization phase for several rounds. From this observation we can claim that the key initialization phase also has very poor period.

References

[1] Ågren, M., Hell, M., Johansson, T., Meier, W.: A new version of grain-128 with authentication. In: Symmetric Key Encryption Workshop (2011)

[2] Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. FSE (2015)

[3] Banik, S.: Some results on sprout. In: Progress in Cryptology-INDOCRYPT 2015, pp. 124–139. Springer (2015)

[4] Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the grain family of stream ciphers. In: Cryptographic Hardware and Embedded Systems-CHES 2012, pp. 122–139. Springer (2012)

[5] Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: IEEE International Symposium on Information Theory (ISIT 2006). Citeseer (2006)

[6] Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. International Journal of Wireless and Mobile Computing 2(1), 86–93 (2007)

[7] Karmakar, S., Chowdhury, D.R.: Fault analysis of grain-128 by targeting nfsr. In: Progress in Cryptology-AFRICACRYPT 2011, pp. 298–315. Springer (2011)

[8] Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. In: Advances in Cryptology-CRYPTO 2015, pp. 663–682. Springer (2015)

[9] Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key recovery from state information of sprout: Application to cryptanalysis and fault attack. Tech. rep., Cryptology ePrint Archive, Report 2015/236, 2015. http://eprint. iacr. org

[10] Soos, M.: Cryptominisat 2.9.6. Solver description. SAT Race (2010)

[11] Stein, W., et al.: Sage: Open source mathematical software (2008)