Fault Attacks on Combiners with Memory^{*}

Frederik Armknecht¹ and Willi Meier²

¹ Universität Mannheim, 68131 Mannheim, Germany

`armknecht@th.informatik.uni-mannheim.de`

² FH Aargau, CH-5210 Windisch, Switzerland

`meierw@fh-aargau.ch`

Abstract. Fault attacks are powerful cryptanalytic tools that are applicable to many types of cryptosystems. Recently, general techniques have been developed which can be used to attack many standard constructions of stream ciphers based on LFSR's. Some more elaborated methods have been invented to attack RC4. These fault attacks are not applicable in general to combiners with memory.

In this paper, techniques are developed that specifically allow to attack this class of stream ciphers. These methods are expected to work against any LFSR-based construction that uses only a small memory and few input bits in its output function. In particular, efficient attacks are described against the stream cipher E0 used in Bluetooth, either by inducing faults in the memory or in one of its LFSR's. In both cases, the outputs derived from the faulty runs finally allow to describe the secret key by a system of linear equations. Computer simulations showed that inducing 12 faults sufficed in most cases if about 2500 output bits were available. Another specific fault attack is developed against the stream cipher SNOW 2.0, whose output function has a 64-bit memory. Similar to E0, the secret key is finally the solution of a system of linear equations. We expect that one fault is enough if about 2^{12} output words are known.

Keywords: Stream cipher, combiner with memory, LFSR, fault attack, Bluetooth E0, SNOW 2.0.

1 Introduction

Fault analysis was first introduced in 1996 in [6] to attack number theoretic public key cryptosystems such as RSA, and later in [7] to attack product block ciphers such as DES. These attacks are practical, and various techniques have been described that induce faults during cryptographic computations (cf. [17]). More recently, fault analysis of stream ciphers has been introduced by Hoch and Shamir in [12]. As for other cryptosystems, fault analysis on stream ciphers is a powerful cryptanalytic tool which can work even if direct attacks are inefficient. In [12], general techniques are applied to attack standard constructions of stream ciphers based on LFSR's, and some specialized techniques are introduced that

^{*} The first author has been supported by grant Kr 1521/7-2 of the DFG (German Research Foundation).

work, e.g., against RC4. In [4], different and improved fault analysis methods on RC4 are derived. The methods developed in [12] apply for most schemes based on memoryless filters and combiners. However, as pointed out in [12], they don't work in general for combiners with memory. Well known stream ciphers that use a combiner with memory are E0, the keystream generator used in Bluetooth, and SNOW 2.0, that is proposed for standardization.

In this paper, fault attacks against general combiners with memory based on LFSR's are developed. It is assumed that faults may be induced either in the memory or in one of the LFSR's. These methods are expected to work against any LFSR-based stream cipher whose output function uses only a small amount of memory. In particular, our methods are adapted and refined to an efficient fault attack on E0, whose output function has a memory of only 4 bit. Our attack heavily exploits the particular structure of E0. Another specific attack is developed against the stream cipher SNOW 2.0 that uses a 64 bit memory.

2 Combiners with Memory

Keystream generators are used for the online encryption of bitstreams having arbitrary length, e.g. between two mobile phones. Initialized with a secret value, the key, they produce a bitstream $Z = z_1, z_2, \dots$, called the keystream. A plaintext $p_1, p_2, \dots$ is encrypted to $c_1, c_2, \dots$ via $c_i := p_i \oplus z_i$ where $\oplus$ denotes the XOR-operation on two bits. A legal receiver who uses the same keystream generator and has knowledge of the key, can produce the keystream by himself and decrypt the received bitstream by computing $p_i = c_i \oplus z_i$.

Many keystream generators used in practice and discussed in theory are combiners with memory or, more formally, (k, ℓ) -combiners. A (k, ℓ) - combiner $\mathcal{C} = (f, \varphi)$ with k inputs and ℓ memory bits is a finite state machine (FSM) which is defined by an output function $f : \{0, 1\}^\ell \times \{0, 1\}^k \rightarrow \{0, 1\}$ and a memory update function $\varphi : \{0, 1\}^\ell \times \{0, 1\}^k \rightarrow \{0, 1\}^\ell$. Given a stream $(X_1, X_2, \dots)$ of inputs, $X_i \in \{0, 1\}^k$, and an initial assignment $Q_1 \in \{0, 1\}^\ell$ to the memory bits, the corresponding output bitstream $(z_1, z_2, \dots)$ is defined according to

$$z_t = f(Q_t, X_t) \quad \text{and} \quad Q_{t+1} = \varphi(Q_t, X_t), \quad (1)$$

for all $t \geq 1$. Using (1), we define $f_\varphi(Q, X_1, \dots, X_r) := (z_1, \dots, z_r)$ and $\varphi(Q, X_1, \dots, X_r) := Q_{r+1}$ where $z_i = f(Q_i, X_i)$, $Q_{i+1} = \varphi(Q_i, X_i)$ and $Q_1 := Q$.

For keystream generation, the stream of inputs $(X_1, X_2, \dots)$ is produced by the output of k driving devices where the initial states are determined by the secret key $K \in \{0, 1\}^n$. We assume that these sequences are generated by LFSRs, i.e., each input is computed by $X_t = L_t(K)$ where L_t is a known linear function $\{0, 1\}^n \rightarrow \{0, 1\}^k$. Consequently, the initial state can be reconstructed, if enough linear equations relating input bits X_t are known.

An example for a combiner with memory is the summation generator which is based on the integer addition. In the case of $k = 2$ input bits $X_t = (a_t, b_t)$, the number of memory bits is $\ell = 1$ and the functions are defined by $z_t = f(Q_t, a_t, b_t) := a_t \oplus b_t \oplus Q_t$ and $Q_{t+1} = \varphi(Q_t, a_t, b_t) := a_t b_t \oplus a_t Q_t \oplus b_t Q_t$. A

practical example for a combiner with memory using LFSRs is the keystream generator E_0 used in the Bluetooth standard. It is a (4, 4) combiner with a 128-bit key.

It is commonly accepted to evaluate the security of combiners with memory in the following attack model. An adversary knows the functions f and φ and the driving devices. In the case that LFSRs are incorporated, he knows the linear functions L_t . Furtheron, he is able to observe the values of some keystream bits z_t . An attack is to use these informations to recover the secret key K .

It is clear from information theory that the knowledge of some keystream bits z_t must provide some information about the corresponding inputs X_t . This gives rise to the following definition:

Definition 1. For a (k, ℓ) -combiner $\mathcal{C} = (f, \varphi)$, $r \geq 1$ and $(z_1, \dots, z_r) \in \{0, 1\}^r$, we define the set of possible inputs by

$$X_{(z_1, \dots, z_r)} := \{(X_1, \dots, X_r) \mid \exists Q : f_\varphi(Q, X_1, \dots, X_r) = (z_1, \dots, z_r)\}. \quad (2)$$

An important property of combiners with memory is the following result which has been also essential in algebraic attacks [1]:

Theorem 1. Let $\mathcal{C} = (f, \varphi)$ be an arbitrary (k, ℓ) -combiner with k inputs and ℓ memory bits. Then there exists at least one output $(z_1, \dots, z_{\ell+1}) \in \{0, 1\}^{\ell+1}$ such that $|X_{(z_1, \dots, z_r)}| \leq \frac{1}{2} \cdot 2^{k \cdot (\ell+1)}$. This means that this specific output can be generated by at most half of all possible inputs in $\{0, 1\}^{k \cdot (\ell+1)}$. In particular this allows to rule out some values for $(X_1, \dots, X_{\ell+1})$.

Proof. Note that $(Q, X_1, \dots, X_{\ell+1})$ uniquely determines the output $Z = (z_1, \dots, z_{\ell+1})$ and therefore $\{0, 1\}^{\ell+k \cdot (\ell+1)} = \bigcup_Z f_\varphi^{-1}(Z)$. Furtheron, it is by definition $|f_\varphi^{-1}(Z)| \geq |X_Z|$.

Assume that the proposition is not true, i.e., $|X_Z| > 2^{k \cdot (\ell+1) - 1}$ for each $Z \in \{0, 1\}^{\ell+1}$. This leads to the contradiction

$$\begin{aligned} 2^{\ell+k(\ell+1)} &= \sum_Z |f_\varphi^{-1}(Z)| \geq \sum_Z |X_Z| \\ &> \sum_Z 2^{k(\ell+1)-1} = 2^{\ell+1} \cdot 2^{k(\ell+1)-1} = 2^{\ell+k(\ell+1)}. \end{aligned}$$

3 A General Fault Attack on Combiners with Memory

The model assumptions for fault attacks are that an adversary has access to the keystream generator as a physical device. Considering the proliferation of mobile phones or devices using Bluetooth, this is certainly a realistic scenario.

Furthermore, the attacker can re-set the device to the same unknown initial setting as often as he wants. As stream ciphers are descendants of the one-time pad, the normal mode of operation would disallow to re-set the stream cipher and to generate the same keystream more than once. On the other hand, an attacker

which has acquired the physical device is certainly more powerful than a normal passive attacker. For example, consider the case of a stream cipher that uses a resynchronization mechanism where the frame key is generated from a master key and a public initial value IV (as for example it is the case in Bluetooth). The attacker could feed the same IV into the device several times which would practically re-set the stream cipher each time to the same unknown setting.

In addition, it is assumed that the attacker can cause an unknown fault into one of the registers during the computations. He knows the affected register, but can neither control the point in time of the disturbance nor the induced error. We refer in this context to [17] where a low-tech, low-cost method to induce faults at very specific locations were proposed.

A certainly important question is whether fault attacks on stream ciphers can be considered to be practical or not. For the reasons mentioned above, we would agree and believe that fault attacks on stream ciphers pose a potential threat being worth to be investigated further. On the other hand, no practical fault attacks on stream ciphers have been conducted so far and it might finally turn out that the attack assumptions are too powerful. We hope that the results in [12] and in this paper animate further research to finally settle this question (positive or negative).

In this section, we introduce a fault attack which works in principle for every (k, ℓ) -combiner based on LFSRs if k and ℓ are not too big. We assume the fault to be either induced into the memory register or into one of the LFSRs. In the first case our attack is directly applicable. This will be demonstrated by successful fault attacks on the memory of E0 and of SNOW 2.0. In the second case the additional problem to determine the induced fault in the LFSR is imposed. This may be practically infeasible for certain keystream generators. On the other hand, we will show in the next section that this problem can be solved for the keystream generators E_0 .

Let $Q_1 \in \{0, 1\}^\ell$, and $(X_1, \dots, X_R) \in \{0, 1\}^{k \cdot R}$, $R \geq 1$, be the output of the k LFSRs, depending on an initial setting K . The keystream is denoted by $(z_1, \dots, z_R) := f_\varphi(Q_1, X_1, \dots, X_R)$. The attacker is able to re-run the keystream generator an arbitrary number of times with the same unknown initial states Q_1 and K and to observe the produced keystream. During the run, he can induce a unknown fault either into the memory register or into one of the LFSRs, but has no control about the point in time t' . Let $(z'_1, \dots, z'_R)$ be the corresponding keystream. The keystream bits z'_t and z_t are equal for $t < t'$ but may be different for $t \geq t'$. By comparing both keystreams, the first clock t with $z_t \neq z'_t$ indicates that the fault has already occurred.

Let $X'_t \in \{0, 1\}^k$ be the output of the LFSRs at clock t of the faulty run and $\delta_t := X_t \oplus X'_t$ the unknown differences. The location of the induced fault and the kind of the keystream generator determine how to deal with the differences.

In the case where the memory register has been tampered, the LFSRs have been left unchanged. Hence, in this case $\delta_t = 0$ for all clocks t .

If the fault has been induced in one of the LFSRs, the situation is more complicated. Some of the δ_t are different to zero but the attacker does not know

which. For general (k, ℓ) -combiner, $f(Q_t, X_t) = z_t \neq z'_t = f(Q'_t, X'_t)$ does only implicate that $(X_t, Q_t) \neq (X'_t, Q'_t)$ but not that $X_t \neq X'_t$. Therefore, comparing both keystreams does not reveal immediately information about the induced fault. Nevertheless, the keystream generator may have additional properties, allowing the detection of the whole fault. One such example is E_0 , as discussed in the next two sections.

A special case are simple combiners. Simple combiner means that the combiner is memoryless, e.g. $\ell = 0$, and that the k LFSRs are all distinct. As no memory bits are present, $z_t \neq z'_t$ implies $f(X_t) \neq f(X'_t)$ and therefore $X_t \neq X'_t$. Sampling enough such time instances permit to compute the whole fault. Therefore, simple combiners seem to be vulnerable to fault attacks in general.

Altogether, we assume that the adversary has the capability of finding out the induced fault.

The proposed fault attack is related to algebraic attacks. After observing the original keystream $z_1, \dots, z_R$, an attacker knows (for a fixed value $\tau \geq 0$) that

$$X_t^{t+\tau} := (X_t, \dots, X_{t+\tau}) \in X_{z_t, \dots, z_{t+\tau}}, \quad 1 \leq t \leq R - \tau. \quad (3)$$

In the positive case, he may use this information to derive algebraic equations in $X_t^{t+\tau}$. Because of $X_t = L_t(K)$, this gives a system of equations in K . If the equations are linear, Gaussian elimination can be used to find K . In the case of non-linear equations, more sophisticated methods as relinearization or Gröbner bases might be helpful, in particular, if the number of equations exceeds the number of unknowns. It is known by [1] that for $\tau = \ell$, algebraic equations of degree at most $\lceil \frac{k \cdot (\ell+1)}{2} \rceil$ always do exist.

However, with the additional information at hand by several runs, an attacker can hope for equations of lower degree, or even linear equations. By observing the faulty keystream $z'_1, \dots, z'_R$, the attacker can deduce that, additionally to (3), it holds that

$$X_t'^{t'+\tau} \in X_{z'_t, \dots, z'_{t+\tau}}, \quad t' \leq t \leq R - \tau. \quad (4)$$

If he can determine the differences $\delta_t := X_t \oplus X'_t$ for $t \geq t'$, combining (3) and (4) reduces the set of possible assignments of $X_t^{t+\tau}$ even further. It is

$$X_t^{t+\tau} \in X_{z_t, \dots, z_{t+\tau}} \cap (X_{z'_t, \dots, z'_{t+\tau}} \oplus (\delta_t, \dots, \delta_{t+\tau})) \quad (5)$$

with

$$X_{z'_t, \dots, z'_{t+\tau}} \oplus (\delta_t, \dots, \delta_{t+\tau}) := \{(Y_0 \oplus \delta_t, \dots, Y_\tau \oplus \delta_{t+\tau}) \mid (Y_0, \dots, Y_\tau) \in X_{z'_t, \dots, z'_{t+\tau}}\}.$$

As explained above, if the fault was in the memory bits, it is $\delta_t = 0$. This simplifies (5) to

$$X_t^{t+\tau} \in X_{z_t, \dots, z_{t+\tau}} \cap X_{z'_t, \dots, z'_{t+\tau}}. \quad (6)$$

After each run, the adversary checks if the remaining possible values of $X_t^{t+\tau}$ allow to set up equations of low degree in $X_t^{t+\tau}$. Algorithms to find such equations

have been used in several papers [8, 5, 1, 16], an explicit description is given in [3, Alg. 1]. An alternative is to use Gröbner bases.

Repeating these steps sufficiently often can result in a system of low degree (or even linear) functions. Thus, the final step consists in computing the solution to derive the LFSR's initial state. An example for this attack on the summation generator with $k = 2$ LFSRs is given in appendix A.

The whole attack is summarized in the following pseudo code:

General fault attack on (k, l) -combiners

Input: A (k, l) -combiner C , and integers $d, N \geq 1$ and $\tau \geq 0$

Task: Recover the initial state of the LFSRs

Algorithm:

- Derive for all possible outputs $(z_0, \dots, z_\tau)$ the sets $X_{(z_0, \dots, z_\tau)}$ of possible inputs
- Run the generator once and observe a keystream $z_1, \dots, z_R$.
- For $t = 1, \dots, R - \tau$, initialize the sets of possible inputs $\mathcal{X}_t = X_{(z_t, \dots, z_{t+\tau})}$
- While number of equations of degree $\leq d$ is less than N do
 - Induce an unknown fault in the memory register and observe the output $(z'_1, \dots, z'_R)$.
 - Look for the first clock t' with $z_t \neq z'_{t'}$.
 - Determine the differences $\delta_t := X_t \oplus X'_t$ for $t \geq t'$.
 - For $t \geq t'$, reduce the set of possible inputs $X_t^{t+\tau}$ according to (5) or (6)
 - Derive the number of equations on $\mathcal{X}_t$ of degree $\leq d$ for $t = 0, \dots, T - \tau$ and count them.
- end while;
- Recover the initial state by finding a solution to these equations.

4 Fault Attacks to the Bluetooth Keystream Generator

4.1 Preliminary Notes

In this section, we discuss fault attacks on the keystream generator E_0 , being part of the Bluetooth encryption. E_0 employs four LFSRs, named A , B , C and D , of lengths 25, 31, 33 and 39, respectively. We denote their outputs at clock t by a_t, b_t, c_t and d_t , i.e., it is $X_t = (a_t, b_t, c_t, d_t)$. The memory bits Q_t are referred to as $Q_t = (p_t, q_t, p_{t-1}, q_{t-1})$ and the output z_t is computed via $z_t = q_t \oplus s_t$ where $s_t := a_t \oplus b_t \oplus c_t \oplus d_t$. We assume that the attacker is always able to produce and to observe R keystream bits $z_1, \dots, z_R$. In the Bluetooth standard, the key changes after 2745 outputs. Hence, this value states a natural upper bound for R , at least in respect of practical attacks.

Referring to section 3, we consider two different scenarios: the fault does occur either in the memory register Q_t or in one of the four LFSRs. In [1], it was shown that for any four fixed successive outputs $z_t, z_{t+2}, z_{t+3}, z_{t+4}$, the set $X_{(z_t, z_{t+2}, z_{t+3}, z_{t+4})}$ of possible inputs X_t^{t+3} is a strict subset of $\{0, 1\}^{4 \cdot 4}$. Therefore, every quadruple of four known successive outputs excludes some values for X_t^{t+3} .

We will show how the keystreams $z'_1, \dots, z'_R$ derived from faulty runs can be used to rule out as many inputs as possible. Computer tests showed that this does not allow to single out the actual value of X_t^{t+3} directly, but the information is nonetheless valuable for an attacker. It turned out that in every case, the remaining set of possible values of X_t^{t+3} satisfied a linear equation $L(X_t^{t+3}) = 0$. Implementing this for several clocks finally allows to describe the LFSRs' initial state by a system of linear equations.

4.2 Fault Attack on the Memory Register

In this section, we discuss the fault attack for the case that the memory register is disturbed. Let t' denote the point in time of the fault. As the LFSRs' output remain the same for all faulty runs, it is $\delta_t = 0$ for all t . The keystream bits $z'_1, \dots, z'_R$ allow directly to reduce the set of possible keys as formulated in (6).

We simulated this attack on a computer. For 1000 times, we have chosen a random initial setting and produced faulty keystreams until we were able to set up 128 different linear equations in the initial state. As the time instance t' differs from fault to fault, we made the moderate assumption of 500 generated faulted keystream bits per fault. The results are displayed in Table 1. It shows in how many test cases which number of faults were necessary to derive the system of linear equations. For example, in 17.3 percent of our test cases, 10 faults were required. We see that in more than 90 percent of all simulations, 12 faults were enough.

Table 1. Simulation results for the fault attack on the memory register

# induced faults	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Percentage	10.3	18.9	17.1	17.3	14.2	9.6	5.7	2.5	2.1	0.9	0.7	0.5	0.1	0	0	0.1

4.3 Fault Attack on One LFSR

Now we consider the second attack scenario, where the fault is induced into one of the LFSRs. We assume that the affected LFSR is A which has the shortest length. The state of A is altered at a certain uncontrollable point in time t' and afterwards, some of the inputs $X_t = (a_t, b_t, c_t, d_t)$, memory states Q_t and keystream bits z_t are changed to $X'_t = (a_t \oplus \tilde{\delta}_t, b_t, c_t, d_t)$, Q'_t and z'_t , respectively. This is depicted in Table 2. Let $s'_t := s_t \oplus \tilde{\delta}_t$ and Δ be the unknown fault $\tilde{\delta}_1, \dots, \tilde{\delta}_R$. Observe that the first clock t with $z_t \neq z'_t$ implies immediately that $\tilde{\delta}_t = 1$. W.l.o.g., we assume that $\tilde{\delta}_1 = 1$. This is no restriction as we did not specify the value of R .

The general attack described in section 3 requires to identify the values of δ_t . We present an appropriate method, using correlations between linear combinations of the known z_t , z'_t and the correct δ_t . More precisely, for certain coefficient vectors $\Gamma = (\gamma_0, \dots, \gamma_r)$, it holds that

$$\bigoplus_{i=0}^r \gamma_i \cdot (z_{t+i} \oplus z'_{t+i}) = \bigoplus_{i=0}^r \gamma_i \cdot \delta_{t+i} \quad (7)$$

Table 2. The fault attack on E_0 with faulty LFSR. The grey marked entries may change their values.

Clock	1	...	$t' - 1$	t'	$t' + 1$	...	R
Input	X_0	...	$X_{t'-1}$	$X'_{t'}$	$X'_{t'+1}$	...	X'_R
Memory	Q_0	...	$Q_{t'-1}$	$Q_{t'}$	$Q'_{t'+1}$	...	Q'_R
Keystream	z_0	...	$z_{t'-1}$	$z'_{t'}$	$z'_{t'+1}$	...	z'_R

is true with probability $> 1/2$ if the δ_t agree with the induced fault, i.e. $\delta_t = \tilde{\delta}_t$ for all t . On the other hand, if the $\delta_t \neq \tilde{\delta}_t$, the value on the right hand side is independent of the left side and (7) is true in only half of the cases. We use this as a distinguishing feature to separate the value $\tilde{\Delta}$ of the induced fault from all possible fault values Δ .

Some Γ together with their probabilities that (7) is true are displayed in Table 3. The probabilities were derived by simply counting the positive events for all possible cases. These probabilities cannot be derived directly from the correlations as determined in [11] for E_0 . However, Equation (7) is related to (conditional) correlated linear combinations of the the memory bits q_t which have proven to be useful for the cryptanalysis of combiners with memory before, especially in the case of E_0 (cf. [13, 14, 15]).

Let keystreams Z and Z' be given and coefficient vectors $\Gamma_1, \dots, \Gamma_s$ be fixed. For a potential fault Δ , we define the value $\Sigma_\Gamma(\Delta) := \Sigma_\Gamma(Z, Z', \Delta)$ to be the number of clocks where (7) holds. An example for computing $\Sigma(\Delta)$ is given in appendix B.1. As explained above, we expect for the right Δ to have *on average* a higher value $\Sigma_\Gamma(\Delta)$ than for the wrong guesses.

Consequently, a logical approach is to use the values $\Sigma_\Gamma(\Delta)$ to set up a ranking list of all possible faults. This means that the value Δ with the highest value $\Sigma_\Gamma(\Delta)$ is the most probable fault and so on. Indeed, computer simulations showed that if the ratio of number of possible keys and number of outputs is too small, then the right fault was not amongst those values Δ at the top of the ranking lists.

Table 3. Some correlations together with their probabilities. * means that the probability is independent of the penultimate entry.

Γ	$Pr[\bigoplus_i \gamma_i \cdot (z_{t+i} \oplus z'_{t+i} \oplus \delta_{t+i}) = 0]$
(1, 0, *, 1)	0.508
(1, 1, *, 1)	0.523
(1, 0, 0, *, 1)	0.505
(1, 0, 1, *, 1)	0.505
(1, 1, 0, *, 1)	0.505
(1, 1, 1, *, 1)	0.524
(1, 0, 0, 0, *, 1)	0.530

Therefore, we refined this approach by using multiple ranking lists, defined by the s coefficient vectors Γ_i . The idea is now to look for values Δ which occur amongst the top elements of as many ranking lists as possible.

This is exactly our approach. For each Γ , we determine h different values Δ , having the highest values $\Sigma_\Gamma(\Delta)$. This gives so to speak a ranking list $L(\Gamma) := L(\Gamma, h)$ in terms of the values $\Sigma_\Gamma(\Delta)$, consisting of h different potential faults Δ . The next step is to regroup these according to their occurrences in the ranking lists. We define for each Δ its *index* by $\text{Ind}(\Delta) := \text{Ind}_{Z, Z'}(\Delta)$ to be the number of ranking lists $L(\Gamma_1), \dots, L(\Gamma_s)$ containing Δ . An example for computing the index is given in appendix B.2. As explained above, if one candidate has a higher index than the others, we assume it to be the unknown induced fault.

Computer experiments using a data of 2745 keystream bits showed that often there was no unique Δ with maximum index. In this case, one can restart the keystream generator, induce another fault and hope for a better result this time.

A more refined alternative is to combine the results from two different runs. Let Z be the original keystream and Z' and Z'' be faulty keystreams, resulting from unknown faults $\tilde{\Delta}'$ and $\tilde{\Delta}''$, respectively. To keep the explanation simple, we assume that both faults $\tilde{\Delta}'$ and $\tilde{\Delta}''$ occurred at the same clock. The method does work without this condition too.

Now we bring a new viewpoint into play. Z'' can be seen also as the faulty keystream resulting from the “original” keystream Z' and the induced fault $\tilde{\Delta}' \oplus \tilde{\Delta}''$. This imposes additional conditions on the candidates for $\tilde{\Delta}'$ and $\tilde{\Delta}''$. Consequently, instead of looking for the most probable values for $\tilde{\Delta}'$ and $\tilde{\Delta}''$ independently, one can search for the most probable *pair* for $(\tilde{\Delta}', \tilde{\Delta}'')$. Candidates (Δ_1, Δ_2) which have high indices $\text{Ind}_{Z, Z'}(\Delta_1)$, $\text{Ind}_{Z, Z''}(\Delta_2)$ and $\text{Ind}_{Z', Z''}(\Delta_1 \oplus \Delta_2)$ are therefore preferred. For our simulations, we simply considered the sum of these three values.

The computer experiments with a reduced length LFSR A proved the practicability of our method, at least in the considered test cases. For our tests, we assumed that the attacker is able to produce each time the whole 2745 keystream bits and that all faults occurred during the first 245 clocks. This is no severe condition. If after inducing the fault, the first clock t with $z_t \neq z'_t \leq 245$, we can be sure that the condition is met. If this is not the case, one discards this fault and reattempts again. Therefore, the effort caused by useless runs is comparatively small. Furtheron, one can expect that in almost one out of ten attempts, the faults occurs within the required bounds.

The first tests showed that if all Δ had an index of 2 or less, the event of finding the right fault was very unlikely. Therefore, we restarted to induce faults for a fixed unknown initial setting until at least on index was 3 or higher.

We did 21 tests for the case that LFSR A has length 19 (using some random primitive polynomial for feedback). The correct pair $(\tilde{\Delta}', \tilde{\Delta}'')$ was uniquely reconstructed each time. In average, it was necessary to induce 13.86 faults whereas the maximum was 41. Furthermore, we conducted 3 experiments where A has length 20. Also here, the right pair was reconstructed every time. The average number of faults was 27.66 with the maximum being 59. Experiments with the

original length of 25 have been conducted but have always crashed, presumably because of memory shortage. It remains as an open question whether faults of length 25 could be recovered by a more efficient implementation.

The final step consists in reducing the set of possible inputs as described in section 3, once the faults are determined. We simulated this attack 1000 times and counted the number of faults needed to set up at least 128 linear equations in the last 500 clocks (i.e, in $X_{R-500}, \dots, X_R$). Thereby, we always assumed that we figured out the right fault Δ . It turned out that in 763 cases, 11 faults were necessary, and in the remaining ones 12 faults. Similar to the attack where the memory register is tampered, 12 faults seem to be enough whereas 11 faults are always necessary.

4.4 Fault Attacks on Two-Level- E_0

The keystream generator is only one part of the Bluetooth stream cipher. Therefore, we shortly describe the whole system and point out the consequences of our attacks. During Bluetooth encryption, the secret key is re-initialized frequently, using a secret parameter $\mathcal{K}$, denoted as the master key, and publicly known values IV . The period using the same secret key is called a frame. The whole Bluetooth keystream generation consists of the following steps:

1. Derive the frame key $K = L(\mathcal{K}, IV)$ where L is a known linear Boolean function, $\mathcal{K}$ the master key and IV a publicly known parameter. This means that the Bluetooth cipher uses a linear key schedule. Use K to initialize the LFSRs in E_0 .
2. Clock E_0 several times and compute 128 bit intermediate outputs $y_1, \dots, y_{128}$.
3. Permute the 128 bits using a fixed permutation.
4. Use the 128 bit output to re-initialize the four LFSRs of E_0 and produce the keystream $z_1, \dots, z_{2745}$

In [2], it was shown that if the intermediate outputs y_i are known for several frames, the master key $\mathcal{K}$ can be efficiently reconstructed. The proposed fault attacks provide efficient methods to attack one instantiation of E_0 . Thus, our attacks can be applied to the second call of E_0 for several frames to determine the y_i 's. Afterwards, the methods in [2] can be used to reconstruct $\mathcal{K}$. This shows that the whole Bluetooth encryption is affected by the fault attacks.

5 Fault Attack on SNOW 2.0

SNOW 2.0 has been introduced in [9] and is proposed for standardization. It consists of a length 16 LFSR with 32 bit words, feeding a finite state machine. The FSM consists of two 32 bit memory registers, called $R1$ and $R2$, as well as some operations to calculate the output and the next state (i.e., the next values of $R1$ and $R2$). A schematic illustration is given in Figure 1. For a precise description we refer to [9]. We only recall those facts that are needed in our attack.

Fig. 1. A schematic picture of SNOW 2.0

Denote the 16 words in the LFSR at clock t by $s_t, \dots, s_{t+15}$. The output word z_t at clock t is defined by

$$z_t = (s_{t+15} + R1_t) \oplus R2_t \oplus s_t$$

where “+” means integer addition of 32 bit words. The rule to update registers $R1$ and $R2$ at clock t is given by

$$R2_{t+1} := S(R1_t) \quad \text{and} \quad R1_{t+1} := R2_t + s_{t+5} \quad (8)$$

where S is a nonlinear 32 bit to 32 bit bijection, which is composed of the S-box and the MixColumn step in the AES block cipher. However, we don’t make use of an explicit description of S .

Our fault attack on SNOW 2.0 assumes that one can induce a fault in the memory register $R2$ (for a time instant t' over which there is no control). Now we point out a particular property of SNOW 2.0. Assume for one clock t that either $R2_t$ or $R1_t$ is faulty but not both registers at the same time. Then due

Fault was induced into $R2_{t'}$ before $z_{t'}$ is computed							
Clock	...	$t' - 1$	t'	$t' + 1$	$t' + 2$	$t' + 1$	...
Register $R1$	...	$R1_{t'-1}$	$R1_{t'}$	$R1'_{t'+1}$	$R1_{t'+2}$	$R1'_{t'+3}$	...
Register $R2$	...	$R2_{t'-1}$	$R2'_{t'}$	$R2_{t'+1}$	$R2'_{t'+2}$	$R2_{t'+3}$	...
Keystream	...	$z_{t'-1}$	$z'_{t'}$	$z'_{t'+1}$	$z'_{t'+2}$	$z'_{t'+3}$	...

Fault was induced into $R2_{t'-1}$ after $z_{t'-1}$ has been computed							
Clock	...	$t' - 1$	t'	$t' + 1$	$t' + 2$	$t' + 1$	...
Register $R1$	...	$R1_{t'-1}$	$R1'_{t'}$	$R1_{t'+1}$	$R1'_{t'+2}$	$R1_{t'+3}$	...
Register $R2$	...	$R2'_{t'-1}$	$R2_{t'}$	$R2'_{t'+1}$	$R2_{t'+2}$	$R2'_{t'+3}$	...
Keystream	...	$z_{t'-1}$	$z'_{t'}$	$z'_{t'+1}$	$z'_{t'+2}$	$z'_{t'+3}$	...

Fig. 2. The two different cases if inducing a fault into $R2$

to (8), one clock later the perturbed register is now undisturbed and vice versa. Therefore, for each clock $t \geq t'$, either $R2_t$ or $R1_t$ is faulty but not both together.

Assume that after inducing the fault, the output is changed starting from a clock t . Then, two different cases are possible. Either the value of $R2_{t'}$ has been changed *before* the computation of $z_{t'}$ or $R2_{t'-1}$ has been disturbed *after* the computation of $z_{t'-1}$. This is shown in Figure 2. The gray marked entries indicate the faulty ones.

In both cases several outputs z_t with faulty registers $R2_t$ are known. In the first case, it is $t \in \{t', t' + 2, t' + 4, \dots\}$, in the second $t \in \{t' + 1, t' + 3, t' + 5, \dots\}$. As we do not know which case is true, we have to do the attack once for each case. Afterwards, the wrong solution can be easily discarded by computing additional keystream and comparing it with the observed data.

We describe the attack for the case that $R2_t$ has been disturbed before z_t has been computed. The attack for the other case is straightforward. In a basic version, the attack works as follows:

Consider a fault induced in $R2_t$, that complements the least significant bit, but does not complement the 2nd lowest bit in $R2_t$. The fault in the higher bits may be arbitrary. The faulty memory register is called $R2'_t$. That is $R2_t \oplus R2'_t = (*, \dots, *, 0, 1)$. This fault is identically visible by comparing the output z'_t of the faulty generator by the original output z_t .

Now the memory registers are updated to

$$R2_{t+1} = \mathcal{S}(R1_t)$$

$$R1_{t+1} = R2_t + s_{t+5}$$

$$R1'_{t+1} = R2'_t + s_{t+5}$$

In $R1'_{t+1}$ a fault is induced that becomes visible by comparing z_{t+1} and z'_{t+1} (in $R2_{t+1}$ no difference has been induced). Thus we have

$z_{t+1} = (s_{t+16} + R2_t + s_{t+5}) \oplus R2_{t+1} \oplus s_{t+1}$ and $z'_{t+1} = (s_{t+16} + R2'_t + s_{t+5}) \oplus R2_{t+1} \oplus s_{t+1}$. Focus now on $S_1 = s_{t+16} + R2_t + s_{t+5} = s_{t+16} + s_{t+5} + R2_t$ and $S_2 = s_{t+16} + R2'_t + s_{t+5} = s_{t+16} + s_{t+5} + R2'_t$. Then the XOR of S_1 and S_2 agrees with that of z_{t+1} and z'_{t+1} . There are now two cases: Either the second last bit of S_1 agrees with that of S_2 . Then $lsb(s_{t+16} + s_{t+5}) = 0$, otherwise $lsb(s_{t+16} + s_{t+5}) = 1$. In the first case there is thus no change of the carry, and in the second case there is one. Anyway, this gives a linear equation in the LFSR's state bits.

Carry out these steps for at least $512 = 32 \cdot 16$ different clock, until we have sufficiently many linear equations, so that we can solve for the contents of the LFSR. Observe that some linear equations might be linearly dependent, so that it might be necessary to consider more clocks. As in average every 4th fault induced in $R2_t$ has the required form and as we can use only every second output, we need at least $2 \cdot 4 \cdot 512 = 2^{12}$ output words produced by the correct and 2^{12} output words produced by the faulted generator.

One can modify and further relax the conditions on the fault induced, e.g., one could apply this attack on the 2nd lowest bit. Then one obtains quadratic instead of linear equations.

6 Conclusion

We have shown that fault attacks can be successfully applied to LFSR-based stream ciphers that use a combiner with memory. This is in particular demonstrated by fault attacks against the stream cipher E0 used in Bluetooth and against the stream cipher SNOW 2.0. As our general attack proposed in Section 3 relies on computing the set of possible inputs $X_{(z_1, \dots, z_r)}$, an obvious countermeasure would be to choose the number k of inputs as well as the number ℓ of memory bits of the output function not too small (e.g., $k, \ell \geq 10$). Furtheron, the attack described in Section 4.3 would fail if LFSRs of greater length were used. Together with the results from [12], it seems to be reasonable to use few long LFSRs instead of many but short ones. Indeed, our attack on SNOW 2.0 shows that these countermeasures are not sufficient. As a result, it would be interesting to see if better countermeasures against fault attacks on stream ciphers exist. In particular, it seems open whether fault attacks apply in general against stream ciphers which are not based on LFSR's and whose driving devices are nonlinear.

Acknowledgement

The second author receives partial funding through GEBERT RÜF STIFTUNG. We would like to thank the anonymous reviewers which helped to improve the presentation and Li Yu and Stefan Lucks for helpful discussions.

References

1. Armknecht, Krause: *Algebraic Attacks on Combiners with Memory*, Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003.
2. Armknecht, Lano, Preneel: *Extending the resynchronization attack*, SAC 2004, LNCS 3357, pp. 19-38, Springer, 2004.
3. Armknecht: *On the existence of low-degree equations for algebraic attacks*, Cryptology ePrint Archive: Report 2004/185.
4. Biham, Granboulan, Nguyen: *Impossible Fault Analysis of RC4 and Differential Fault Analysis of RC4*, FSE 2005, Springer, 2005.
5. Biryukov, De Cannière: *Block Ciphers and Systems of Quadratic Equations*, FSE 2003, LNCS 2887, pp. 274-289, Springer, 2003.
6. Boneh, DeMillo, R.J. Lipton: *On the Importance of Checking Cryptographic Protocols for Faults*, Eurocrypt 1997, LNCS 1233, pp. 37-51, Springer, 1997.
7. Biham, Shamir: *Differential fault analysis of secret key cryptosystems*, Crypto 1997, LNCS 1294, pp. 513-525, Springer, 1997.
8. Courtois, Pieprzyk: *Cryptanalysis of block ciphers with overdefined systems of equations*, Asiacrypt 2002, LNCS 2501, pp. 267-287, Springer, 2002.
9. Ekdahl, Johansson: *A new version of the stream cipher SNOW*, SAC 2002, LNCS 2595, pp. 47-61, Springer, 2002.
10. Ekdahl: *On LFSR-based Stream Ciphers - Analysis and Design*, Ph.D. Thesis, Lund University, Sweden, November 21, 2003.
11. Golić, Bagini, Morgari: *Linear Cryptanalysis of Bluetooth Stream Cipher*, Eurocrypt 2002, LNCS 2332, pp. 238-255, Springer 2002.

12. Hoch, Shamir: *Fault Analysis of Stream Ciphers*, CHES 2004, LNCS 3156, pp. 240-253, Springer, 2004.
13. Lu, Vaudenay: *Faster Correlation Attack on the Bluetooth Keystream Generator*, Crypto 2004, LNCS 3152, pp. 407-425, Springer, 2004.
14. Lu, Vaudenay: *Cryptanalysis of Bluetooth Keystream Generator Two-Level E0*, Asiacrypt 2004, LNCS 3329, pp. 483-499, Springer, 2004.
15. Lu, Vaudenay, Meier: *The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption*, Crypto 2005, to appear.
16. Meier, Pasalic, Carlet: *Algebraic attacks and decomposition of Boolean functions*, Eurocrypt 2004, LNCS 3027, pp. 474-491, Springer, 2004.
17. Skorobogatov, Anderson, *Optical Fault Induction Attacks*, CHES 2002, LNCS 2523, pp. 2-12, Springer, 2002.

A An Example for Fault Attack on the Summation Generator with $k = 2$ Inputs

In this section we illustrate the general fault attack described in section 3 on the summation generator with $k = 2$ inputs $X_t = (a_t, b_t)$ and $\ell = 1$ memory bits from section 2. By theorem 1, not all outputs $(z_t, z_{t+1}) \in \{0, 1\}^2$ can be produced by all 16 possible inputs $(X_t, X_{t+1}) \in \{0, 1\}^4$. Due to the small values of k and ℓ , it is possible to exhaustively compute all possible input-output combinations over 2 clocks. From this, it is easy to derive for a given output (z_t, z_{t+1}) the set of possible inputs.

$$\begin{aligned}
 X_{0,0} &= \{(0000), (0011), (1101), (1110), (0101), (0110), (1001), (1010)\} \\
 X_{0,1} &= \{(0001), (0010), (1100), (1111), (0100), (0111), (1000), (1011)\} \\
 X_{1,0} &= \{(0100), (0111), (1000), (1011), (0000), (0011), (1101), (1110)\} \\
 X_{1,1} &= \{(0101), (0110), (1001), (1010), (0001), (0010), (1100), (1111)\}
 \end{aligned}$$

Note that each output (z_t, z_{t+1}) can be produced by only half of the 16 possible inputs $(a_t, b_t, a_{t+1}, b_{t+1})$, i.e., the property described in theorem 1 holds for each possible output. Any observed output segment (z_t, z_{t+1}) provides the information that $(X_t, X_{t+1}) \in X_{(z_t, z_{t+1})}$. It is easy to see that for each (z_t, z_{t+1}) , no linear functions $g \neq 0$ exists with $g(X_t, X_{t+1}) = 0$ for all $(X_t, X_{t+1}) \in X_{(z_t, z_{t+1})}$. More specifically, it is not possible for the adversary to derive a linear function $L : \{0, 1\}^4 \rightarrow \{0, 1\}$ such that $L(X_t, X_{t+1}) = 0$ holds.

In the fault attack scenario, the adversary can now re-run the keystream generator with the same initial setting and induce a unknown fault in the memory register during the run. Let $(z'_t, z'_{t+1}, \dots)$ be the keystream of this second disturbed run, also observed by the adversary. He cannot choose the point in time t' of the disturbance, but estimate it by comparing the two keystreams $(z_1, \dots)$ and $(z'_1, \dots)$. Suppose that the fault is induced in $Q_{t'}$ with $t' \leq t$ and a different keystream segment $(z'_t, z'_{t+1}) \neq (z_t, z_{t+1})$ has occurred. Hence, it is $X_t \in X_{(z_t, z_{t+1})} \cap X_{(z'_t, z'_{t+1})}$ and the number of possible assignments of X_t is further reduced. For example, if $(z_t, z_{t+1}) = (0, 0)$ and $(z'_t, z'_{t+1}) = (1, 0)$, then

$$X_t^{t+1} = (a_t, b_t, a_{t+1}, b_{t+1}) \in X_{(0,0)} \cap X_{(1,0)} = \{(0000), (0011), (1101), (1110)\}.$$

X_t^{t+1} is still not uniquely determined, but the adversary has gained the additional information that $a_t \oplus b_t = 0$ is true in any case. Repeating these steps for several clocks t , it is possible to derive a system of linear equations in X_t and hence in K . By computing the solution, it is easily possible to determine the unknown initial setting K .

B Examples Concerning Section 4.3

B.1 Example for Computing $\Sigma_\Gamma(Z, Z', \Delta)$

We demonstrate on an example how the value $\Sigma(\Delta)$ is computed. Let $\Gamma = (\gamma_0, \gamma_1, \gamma_2) := (1, 1, 0, 1)$. Now we consider the following situation:

t	1	2	3	4	5	6	7	8	9	10
z_t	1	1	0	0	0	1	1	1	0	0
z'_t	0	1	1	0	1	1	0	1	1	1
δ_t	0	0	1	1	1	0	1	0	0	1

By definition, $\Sigma_\Gamma(Z, Z', \Delta)$ is the number of clocks $1 \leq t \leq t$ with

$$(z_t \oplus z'_t) \oplus (z_{t+1} \oplus z'_{t+1}) \oplus (z_{t+3} \oplus z'_{t+3}) = \delta_t \oplus \delta_{t+1} \oplus \delta_{t+3}.$$

In this case, the values of $\bigoplus \gamma_i (z_t \oplus z'_t \oplus \delta_t)$ for $t = 1, \dots, 7$ are 0, 0, 1, 1, 0, 1 and 0, respectively. Hence, it is $\Sigma_\Gamma(Z, Z', \Delta) = 4$.

B.2 Example for Computing the Index $\text{Ind}(\Delta)$

In our example we assume that altogether 8 different possible faults Δ exist, denoted by $\Delta_0, \dots, \Delta_7$. We use four different coefficient vectors $\Gamma_1, \dots, \Gamma_4$ and consider for each of them the three Δ 's having the highest values $\Sigma_{\Gamma_i}(Z, Z', \Delta)$ (see also previous section). We abbreviate $\Sigma_{\Gamma_i}(Z, Z', \Delta_j)$ by Σ_i^j .

Assume that computing the values Σ_i^j gives the following orderings:

$$\begin{aligned} \Sigma_1^7 &< \Sigma_1^0 < \Sigma_1^3 < \Sigma_1^4 < \Sigma_1^2 < \Sigma_1^1 < \Sigma_1^6 < \Sigma_1^5 \\ \Sigma_2^1 &< \Sigma_2^5 < \Sigma_2^0 < \Sigma_2^3 < \Sigma_2^7 < \Sigma_2^6 < \Sigma_2^2 < \Sigma_2^4 \\ \Sigma_3^2 &< \Sigma_3^0 < \Sigma_3^1 < \Sigma_3^4 < \Sigma_3^5 < \Sigma_3^7 < \Sigma_3^3 < \Sigma_3^6 \\ \Sigma_4^7 &< \Sigma_4^2 < \Sigma_4^5 < \Sigma_4^1 < \Sigma_4^3 < \Sigma_4^0 < \Sigma_4^4 < \Sigma_4^6 \end{aligned}$$

The first Top-3-list $L(\Gamma_1)$ contains the three Δ_j having the highest values of Σ_1 . In this case, these are Δ_1 , Δ_6 and Δ_5 . $L(\Gamma_2)$, $L(\Gamma_3)$ and $L(\Gamma_4)$ are defined analogously. It is $L(\Gamma_1) = [\Delta_1, \Delta_6, \Delta_5]$, $L(\Gamma_2) = [\Delta_6, \Delta_2, \Delta_4]$, $L(\Gamma_3) = [\Delta_7, \Delta_5, \Delta_3]$ and $L(\Gamma_4) = [\Delta_0, \Delta_4, \Delta_6]$.

The next step consists in counting for each Δ_j its number of occurrences in $L(\Gamma_1), \dots, L(\Gamma_4)$, called its index. As Δ_6 has the highest index, namely 3, it is an apparent candidate for the unknown guess. But simulations showed that this strategy results in wrong guesses in significantly many cases. Hence, we imposed a stronger condition for a guess: the index of the guess must be at least two higher than the other indices.