Fault Jumping Attacks against Shrinking Generator

Marcin Gomułkiewicz[1], Mirosław Kutyłowski[1], Paweł Wlaź[2]

Wrocław University of Technology[1]
Lublin University of Technology[2]

Abstract

In this paper we outline two cryptoanalytic attacks against hardware implementation of the shrinking generator by Coppersmith *et al.*, a classic design in low-cost, simple-design pseudorandom bitstream generator. This is a report on work on progress, since implementation and careful adjusting the attack strategy in order to optimize the atatck is still not completed.

1 Introduction

This paper briefly presents a preliminary version of two new fault attacks ([2]) against the LFSR-based shrinking generator proposed by Coppersmith *et al.* [4]. The shrinking generator is one of the major designs for efficient and secure pseudorandom generators, due to its simplicity and resistance to the known cryptographic attacks. Our attacks are unique in many ways. As far as we know, the use of a fault attack against the shrinking generator is fairly new concept (see [10]); on top of that the fault model assumed here seems to be quite in line with the technical feasibility.

The paper is organized as follows: first we give extremely brief overview of the shrinking generator and previous fault attacks against it. Then we describe the new ideas, pointing out the missing parts.

The Shrinking Generator The shrinking generator [4] is an attempt to create cryptographically strong pseudorandom bitstream generator out of relatively weak components. Many other solutions of this kind [7, 1, 3] were proven to be weak [14, 15]. The shrinking generator successfully faces the trial of time: the best known attacks against it are exponential in the LFSR's length [5, 8, 11, 12, 13], or based on the assumption that the feedback is known [6].

Amazingly, the construction of the shrinking generator is very simple. It consists of two bitstream generators (most frequently LFSRs) we shall call the base (or input) generator A and the control generator C; their output is denoted as $a_1, a_2, a_3, \ldots$ and $c_1, c_2, c_3, \ldots$, respectively. The output $Z = z_1, z_2, z_3, \ldots$ is composed of those and only those of a_i for which $c_i = 1$. Formally:

$$z_t = a_i \quad \text{if} \quad t = \sum_{j=1}^{i} c_j \text{ and } c_i = 1 \tag{1}$$

Previous Fault Attacks on Shrinking Generator The paper [10] we show two fault attacks against the shrinking generator. The first one is based on the assumption that the clocks of the internal generators can be desynchronized (a similar assumption was used in [9]), the second one is essentially based on possibility of replacing the control register by a source of independent, random bits, while keeping the input register's contents. The first attack gives quite powerful results (with high probability only a couple of possible control sequences, including the correct one), but the assumptions made require the cooperation of the device's manufacturer (and / or a very careless design of the integrated circuit implementing the shrinking generator). The second attack, while quite feasible from a technical point of view, gives only moderately strong results (candidates for the bits of the input sequence, correlated, but not neccessarily equal to the correct one).

2 New Attack

We shall now show the outline of a new attack against the shrinking generator. This time, in opposition to the previous work [10] we shall assume that the control register is an LFSR with known feedback, while the input generator may be arbitrarily chosen random bit generator. Our aim is to discover the contents of the control register.

Notations and Assumptions We shall adopt the notations from Section 1. On top of that we shall assume that one can cause exactly one bit-flip within the control register and rerun the generator. That is, if the register has length n, and its' cells contain (from the beginning to the end) $c_n, c_{n-1}, \dots, c_2, c_1$, then it is possible to get the output sequence Z' corresponding to the $c'_n, c'_{n-1}, \dots, c'_2, c'_1$, where all but one c'_i are equal to c_i. Of course, the fault injected into the control generator will propagate when the generator is working and gradually many bits are influenced by the change.

We also assume that we posses nice algorithm that can detect insertions or deletions of (single) bits in the binary stream. That is, we assume taht if $\mathcal{A}$ is given two streams:

$$S = s_1, s_2, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_k$$

and

$$S' = s_1, s_2, \dots, s_{i-1}, s_{i+1}, \dots, s_k$$

it should reply "REMOVE(i)", or given two streams:

$$S = s_1, s_2, \dots, s_i, s_{i+1}, \dots, s_k$$

and

$$S' = s_1, s_2, \dots, s_i, r, s_{i+1}, \dots, s_k$$

it should reply "INSERT(i)". For a couple of delete / insert operations $\mathcal{A}$ should say e.g. "INSERT(2) and REMOVE(5) and REMOVE(10)"; it is also acceptable that the algorithm $\mathcal{A}$ outputs a couple of possible replies to one query, say: "INSERT(2) or INSERT(3)", or something like "(REMOVE(2) or REMOVE(3)) and INSERT(7)".

Attack Having all this we proceed as follows:

Since the feedback of the control register is known, we can express all its' output as linear functions of its' (yet unknown) state. For example, if register C has length 5 and its' feedback is the sum of the last two bits in register, its' output equals (all sums are regarded to be modulo 2):

$$
\begin{aligned}
&c_1,\, c_2,\, c_3,\, c_4,\, c_5,\\
&c_1+c_2,\, c_2+c_3,\, c_3+c_4,\, c_4+c_5,\, c_1+c_2+c_5,\\
&c_1+c_3,\, c_2+c_4,\, c_3+c_5,\, c_1+c_2+c_4,\, c_2+c_3+c_5,\\
&c_1+c_2+c_3+c_4,\, c_2+c_3+c_4+c_5,\, c_1+c_2+c_3+c_4+c_5,\\
&c_1+c_3+c_4+c_5,\, c_1+c_4+c_5,\, c_1+c_5,\\
&c_1,\, c_2,\, c_3,\, c_4,\, c_5,\\
&\ldots
\end{aligned}
$$

We take two outputs: one from the correct computation, the other one – from a computation with exactly one bit in C flipped. We guess the position of the flipped bit and comparing the answer given by $\mathcal{A}$ on these outputs with (unknown) output of the LSFR we construct a system of linear equations.

For example, assume that an algorithm said "REMOVE(1) and INSERT(3) and (REMOVE(8) or REMOVE(9))". We assume that flipped bit was c_1 and we check the first possibility: "REMOVE(1) and INSERT(3) and REMOVE(8)". If so, then the changes in the control sequence must occur at positions 1, 6, 10, 11, 14, 16, 18, 19, 20, 21, and so on. Since we observed an insertion of an element on the first position in the output, we may think that c_1 was one (and it was flipped to zero), c_1+c_2 was zero (and was changed to one), and $c_1+c_2+c_5$ was one (and was changed to zero). This leads to a linear system:

$$
\begin{cases}
c_1 & = 1\\
c_1+c_2 & = 0\\
c_1+c_2+c_5 & = 1
\end{cases}
$$

We can also deduce some information from the location of points where changes occured. If the first change was visible at the first position, the c_1 must have been one; since second change was visible at the third position, then $c_1+c_2+c_3+c_4+c_5+(c_1+c_2 \bmod 2)=3$, and

$$
\begin{aligned}
&c_1+c_2+c_3+c_4+c_5+(c_1+c_2 \bmod 2)+(c_2+c_3 \bmod 2)+\\
&+(c_3+c_4 \bmod 2)+(c_4+c_5 \bmod 2)+(c_1+c_2+c_5 \bmod 2)=8\ .
\end{aligned}
$$

Having all that information we try to solve given linear equations to find c_i's or to learn that they are contradictory (which means that assumed flipped bit was guessed wrong, or $\mathcal{A}$ has given a wrong answer).

Let us comment that the answers of $\mathcal{A}$ should take into account the following phenomena:

1. The first difference between two outputs is due to the bit flipped. If a new bit occurs in the second output, we know for sure that the control bit was flipped from 0 to 1.

2. If the faults occured at a position c_i where i is not much less than n then, due to propagation of faults via feedback function for quite a long (and fixed) time the control generator outputs exactly the same data, so the output of the shrinking generator is exactly the same. This long sequence

is a witness of the correctness of our hypothesis. Usually, the opposite hypothesis is excluded for the same reasons. The only case, when the both possibilities remain open is when the output of the generator from this moment, say $b_1, b_2, \ldots$, has the property that

$$r, b_1, b_2, \ldots, b_k = b_2, b_3, \ldots, b_{k+2}$$

where r is a new bit inserted and k corresponds to the number of ones in the output of the control generator between the first fault position and the next position where the fault yields a change. Obviously, then we have $b_1 = b_3$, $b_2 = b_4$, $b_3 = b_5$, $\ldots$. There are only 4 solutions to these equations.

3. The distance bitween the first and the second position in the output sequences where operations REMOVE and INSERT are proposed by $\mathcal{A}$ yield also information on how many ones occur in the output of the control generator in a certain (known) interval.

4. Gradually, the distances between positions where the changes at the control sequence occur become small, so the analysis becomes useless. Therefore it is better to commence with injecting a new fault. However, now we know more: certain positions of the control sequence are known.

3 Dual Attack

This attack is in some way dual to the attack presented in Section 2: here we assume that the input register is a LFSR with a known feedback, the control register is an arbitrarily chosen bitstream generator, and our aim is to find the contents of the input register. Although this attacks seems to be more difficult, it can also present a threat to some shrinking generator's implementations.

Notations and Assumptions As before, we adopt the notation from Section 1. Let the input register A be LFSR of length n with known feedback and its' internal registers contain bits $a_n, a_{n-1}, \ldots, a_2, a_1$ (a_1 is at the output position). As before, we assume that one can flip exactly one of a_i, and get the output sequence Z (with the same bitstream C).

Attack If we assume that a certain bit in A have been flipped, then since A is described by linear equations we can predict precisely all the changes in A's output sequence. Because C was not changed, we should see about half of the changed bits changed. Then, by a careful analysis we can deduce some linear expressions with c_i's as variables.

For example, if we expect changes in A sequence on positions 1, 10, 15, 22, etc., and we see changes on 1, 5 and 8 position in output (Z) bitstream, it is sound to assume that:

$$\left\{\begin{array}{rcl} c_1 & = & 1 \\ \sum_{j=1}^{10} c_j & = & 5 \\ c_{10} & = & 1 \\ \sum_{j=1}^{15} c_j & = & 8 \\ c_{15} & = & 1 \end{array}\right.$$

(see 1). Of course, there are also different valid (although less probable) assumptions, such as:

$$\begin{cases} c_1 &= 1 \\ \sum_{j=1}^{10} c_j &= 5 \\ c_{10} &= 1 \\ \sum_{j=1}^{22} c_j &= 8 \\ c_{22} &= 1 \end{cases}$$

etc.

Having those linear expressions we try to solve the appropriate systems to find C's output or learn that our conditions were contradictory.

4 Conclusions

The yet unimplemented attack ideas show that properly tailored fault analysis should yield a lot of information about the internal state of the shrinking generator or similar design, for very realistic fault assumptions. This shows that the designs such as the shrinking generator deserve a lot of attention and either a tamper resistant hardware implementation or redesigning the generator on the algorithmic level.

References

[1] T. Beth, F. C. Piper, **The Stop-and-Go Generator**, Advances in Cryptology – EUROCRYPT '84, LNCS tom 209, str. 88–92, Springer-Verlag, 1985

[2] Dan Boneh, Richard A. DeMillo, Richard J. Lipton, **On the Importance of Checking Cryptographic Protocols for Faults**, Advances in Cryptology – EUROCRYPT '97, LNCS 1233, pp. 37–51, Springer-Verlag, 1997

[3] W. Chambers, D. Gollmann, **Clock-Controlled Shift Registers: A Review**, IEEE J. Selected Areas Comm., 7(4): 525–533, May 1989

[4] Don Coppersmith, Hugo Krawczyk, Yishay Mansour, **The Shrinking Generator**, Advances in Cryptology – CRYPTO '93, LNCS 773, pp. 22–39, Springer-Verlag, 1993

[5] Ed Dawson, Jovan Dj. Golič, Leone Simpson, **A Probabilistic Correlation Attack on the Shrinking Generator**, Information Security and Privacy – ACISP '98, LNCS 1438, pp. 147–158, Springer-Verlag, 1998

[6] Patrik Ekdahl, Thomas Johansson, Willi Meier, **Predicting the Shrinking Generator with Fixed Connections**, Advances in Cryptology – EUROCRYPT 2003, LNCS 2656, pp. 330–344, Springer-Verlag, 2003

[7] P. R. Geffe, **How to Protect Data with Ciphers That Are Really Hard to Break**, Electronics, Jan. 4, pp. 99–10, 1973

[8] Jovan Dj. Golič, Luke O'Connor, **Embedding and Probabilistic Correlation Attacks on Clock-Controlled Shift Registers**, Advances in

Cryptology – EUROCRYPT '94, LNCS 950, pp. 230–243, Springer-Verlag, 1995

[9] Marcin Gomułkiewicz, Mirosław Kutyłowski, Theodor Heinrich Vierhaus, Paweł Wlaź, **Synchronization Fault Cryptanalysis for Breaking A5/1**, International Workshop on Efficient and Experimental Algorithms – WEA'05, LNCS 3503, pp. 415–427, Springer Verlag, 2005.

[10] Marcin Gomułkiewicz, Mirosław Kutyłowski, Paweł Wlaź, **Fault Cryptanalysis and the Shrinking Generator**, International Workshop on Efficient and Experimental Algorithms – WEA'06, LNCS 4007, pp. 61–72, Springer Verlag, 2006.

[11] Matthias Krause, Stefan Lucks, Erik Zenner, **Improved Cryptanalysis of the Self-Shrinking Generator**, Information Security and Privacy – ACISP 2001, LNCS tom 2119, str. 21–35, Springer-Verlag, 2001

[12] W. Meier, O. Staffelbach, **The Self-shrinking Generator**, Advances in Cryptology – EUROCRYPT '94, LNCS 950, pp. 205–214, Springer-Verlag, 1995

[13] Miodrag Mihaljevic, **A Faster Cryptanalysis of the Self-shrinking Generator**, Information Security and Privacy – ACISP '96, LNCS 1172, pp. 182–188, Springer-Verlag, 1996

[14] T. R. N. Rao, Chung-Huang Yang, Kencheng Zeng, **An Improved Linear Syndrome Algorithm in Cryptanalysis With Applications**, Advances in Cryptology – CRYPTO '90, LNCS 537, pp. 34–47, Springer-Verlag, 1991

[15] Erik Zenner, **On the Efficiency of the Clock Control Guessing Attack**, Information Security and Cryptology – ICISC 2002, LNCS 2587, pp. 200–212, Springer-Verlag, 2003