Cryptanalysis of Achterbahn-128/80 with a New Keystream Limitation

María Naya-Plasencia*

Projet CODES, INRIA Paris-Rocquencourt, France
Maria.Naya_Plasencia@inria.fr

Abstract. This paper presents two key-recovery attacks against the modification of Achterbahn-128/80 proposed by the authors at SASC 2007 due to the previous attacks. The 80-bit variant, Achterbahn-80, was limited to produce at most 2^{52} bits of keystream with the same pair of key and IV, while Achterbahn-128 was limited to 2^{56} bits. The attack against Achterbahn-80 has complexity $2^{64.85}$ and needs fewer than 2^{52} bits of keystream, and the one against Achterbahn-128 has complexity 2^{104} and needs fewer than 2^{56} keystream bits. These attacks are based on the previous ones. The attack against Achterbahn-80 uses a new idea which allows us to reduce the required keystream length.

Keywords: eSTREAM, stream cipher, Achterbahn, cryptanalysis, correlation attack, linear approximation, parity check, key-recovery attack.

1 Introduction

The invention of public-key cryptography in the mid 1970's was a great progress. However, symmetric ciphers are still widely used because they are the only ones that can achieve high-speed or low-cost encryption. Today, we find symmetric ciphers in GSM mobile phones, in credit cards... Stream ciphers then form a subgroup of symmetric ciphers. In synchronous additive stream ciphers, the ciphertext is obtained by combining with a bitwise XOR the message with a secret binary sequence of the same length. This secret sequence is usually a pseudo-random one, that is generated with the help of a secret key by a pseudo-random generator, and it is called the keystream. Such pseudo-random generators are initialized by the secret key and they build in a deterministic way a long sequence that we cannot distinguish from a random one if we do not know the secret key. The eSTREAM project is a project launched by the European network ECRYPT about the conception of new stream ciphers. About thirty algorithms have been proposed in April 2005. Actually, in phase 3 of the project, 16 are

* This work was supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and by the ANR-06-SETI-013 project RAPIDE. The information in this document reflects only the author's views, is provided as is and no warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

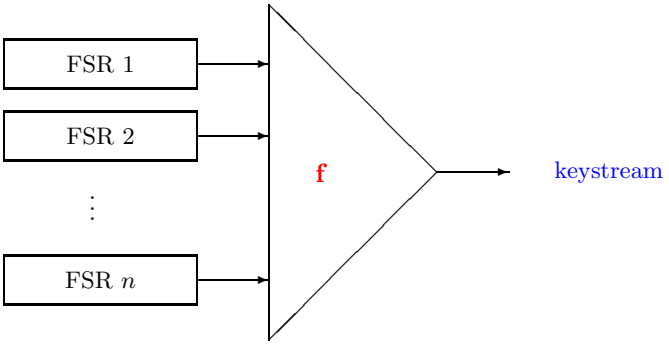
S. Lucks, A.-R. Sadeghi, and C. Wolf (Eds.): WEWoRC 2007, LNCS 4945, pp. 142–152, 2008.

still being evaluated. Achterbahn [3, 5] is a stream cipher proposal submitted to the eSTREAM project that passed to phase 2 but not to phase 3 of eSTREAM. After the cryptanalysis of the first two versions [8, 10], it moved on to a new one called Achterbahn-128/80 [4] published in June 2006. Achterbahn-128/80 corresponds to two keystream generators with key sizes of 128 bits and 80 bits, respectively. Their maximal keystream length was limited to 2^{63}, but, in order to avoid the attacks presented in [9, 11], the maximal keystream length was re-limited to produce at most 2^{52} bits of keystream with the same pair of key and IV for Achterbahn-80, while Achterbahn-128 was limited to 2^{56} bits. This paper presents two key-recovery attacks against this modification to Achterbahn-128/80, proposed by the authors at SASC 2007 [6]. The attack against Achterbahn-80 has complexity $2^{64.85}$ and needs fewer than 2^{52} bits of keystream, and the one against Achterbahn-128 has complexity 2^{104} and needs fewer than 2^{56} keystream bits. These attacks are based on the previous ones. The attack against Achterbahn-80 uses a new idea which allows us to reduce the required keystream length.

The paper is organized as follows. Section 2 presents the main specifications of Achterbahn-128/80. Section 3 then describes a distinguishing attack against Achterbahn-80. Section 4 presents a distinguishing attack against Achterbahn-128. Section 5 describes how this previous distinguishing attacks can be transformed into key-recovery attacks.

2 Achterbahn-128/80

Achterbahn-128 and Achterbahn-80 are composed of a number of feedback shift registers whose outputs are taken as inputs of a Boolean combining function and where the keystream is the output of this function at each instant t.



2.1 Main Specifications of Achterbahn-128

Achterbahn-128 consistis of 13 binary nonlinear feedback shift registers (NLFSRs) denoted by $R0, R1, \ldots, R12$. The length of register i is $L_i = 21 + i$ for $i = 0, 1, \ldots, 12$. These NLFSRs are primitive in the sense that their periods T_i are equal to $2^{L_i} - 1$. Each sequence which is used as an input to the Boolean

combining function is not the output sequence of the NLFSR directly, but a shifted version of itself. The shift amount depends on the register number, but it is fixed for each register. In the following, $x_i = (x_i(t))_{t\geq 0}$ for $0 \leq i \leq 12$ denotes the shifted version of the output of the register i at time t. The output of the keystream generator at time t, denoted by $S(t)$, is the one of the Boolean combining function F with the inputs corresponding to the output sequences of the NLFSRs correctly shifted, i.e. $S(t) = F(x_0(t), \ldots, x_{12}(t))$. The algebraic normal form of the 13-variable combining function F is given in [4]. Its main cryptographic properties are: balancedness, algebraic degree 4, correlation immunity order 8, nonlinearity 3584, algebraic immunity 4.

2.2 Main Specifications of Achterbahn-80

Achterbahn-80 consists of 11 registers, which are the same ones as in the above case, except for the first and the last ones. The Boolean combining function, G, is a sub-function of F :

$$G(x_1, \ldots, x_{11}) = F(0, x_1, \ldots, x_{11}, 0).$$

Its main cryptographic properties are: balancedness, algebraic degree 4, correlation immunity order 6, nonlinearity 896, algebraic immunity 4. As we can see, Achterbahn-128 contains Achterbahn-80 as a substructure.

2.3 The Key-Loading Algorithm

The key-loading algorithm uses the key K of 128/80 bits and an initial value IV of 128/80 bits. The method for initializing the registers is the following one: first of all, all registers are filled with the bits of $K||IV$. After that, Register i is clocked $a - L_i$ times where a is the number of bits of $K||IV$, and the remaining bits of $K||IV$ are added to the feedback bit. Then, each register outputs one bit. Those bits are taken as inputs of the Boolean combining function, which outputs a new bit. This bit is now added to the feedbacks for 32 additional clockings. Then we overwrite the last cell of each register with a 1, in order to avoid the all zero state.

This algorithm has been modified in relation to the initial versions of Achterbahn. The aim of this modification is to prevent the attacker from recovering the key K from the knowledge of the initial states of some registers.

2.4 Keystream Maximal Length

In the first version of Achterbahn-128/80, the maximal keystream length was limited to 2^{63}. As this version was attacked [9, 11], the authors proposed a new limitation of the keystream length [6], which was 2^{52} for Achterbahn-80 and 2^{56} for Achterbahn-128. We present here two attacks against both generators, which are based on the previous ones. The attack against the 80-bit variant, Achterbahn-80, has complexity $2^{64.85}$ and needs fewer than 2^{52} keystream bits. The attack against Achterbahn-128 requires 2^{104} operations and fewer than 2^{56} keystream bits.

3 Distinguishing Attack against Achterbahn-80

Now, we describe a new attack against Achterbahn-80 with a complexity of $2^{64.85}$ where a linear approximation of the output function is considered. The attack is a distinguishing attack but it also allows to recover the initial states of certain constituent registers.

This attack is very similar to the previous attack against Achterbahn-80 presented in [11]. It relies on a biased parity-check relation between the keystream bits which holds with probability

$$p = \frac{1}{2}(1+\eta) \text{ with } |\eta| \ll 1,$$

where η is the bias of the relation. The attack exploits an s-variable linear approximation ℓ of the combining function G. For now on we denote by $T_{i,j}$ the least common multiple of the periods of Registers i and j. We build the parity-check equations, as the ones introduced in [10] and used in [11] derived from ℓ:

$$\ell(t) = \sum_{j=1}^{s} x_{i_j}(t)$$

at 2^m different instants $(t+\tau)$, where τ varies in the set of the linear combinations with $0-1$ coefficients of $T_{i_1,i_2}, T_{i_3,i_4}, \ldots, T_{i_{2m-1},i_{2m}}$. In the following, this set is denoted by $\langle T_{i_1,i_2}, \ldots, T_{i_{2m-1},i_{2m}} \rangle$, i.e.,

$$\mathcal{I} = \langle T_{i_1,i_2}, \ldots, T_{i_{2m-1},i_{2m}} \rangle = \left\{ \sum_{j=1}^{m} c_j T_{i_{2j-1},i_{2j}}, c_1, \ldots, c_m \in \{0,1\} \right\}.$$

We know that:

$$\sum_{\tau \in \mathcal{I}} x_{i_1}(t+\tau) + \ldots + x_{i_{2m}}(t+\tau) = 0,$$

this leads to a parity-check sequence $\ell\ell$ defined by:

$$\ell\ell(t) = \sum_{\tau \in \mathcal{I}} \ell(t+\tau) = \sum_{\tau \in \mathcal{I}} \left(x_{i_{2m+1}}(t+\tau) + \ldots + x_{i_s}(t+\tau) \right).$$

Note that each term with index i_{2j-1} is associated to the corresponding term i_{2j} to build the parity check, because it enables us to eliminate the influence of $2m$ registers in a parity-check with 2^m terms only.

Approximation of the combining function. Following this general principle, our attack exploits the following linear approximation of the combining function G:

$$\ell(x_1, \ldots, x_{11}) = x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_{10}.$$

It is worth noticing that, since the combining function G is 6-resilient, any approximation of G involves at least 7 input variables. Moreover, the highest bias corresponding to an approximation of G by a 7-variable function is achieved by a function of degree one as proved in [2].

For $\ell(t) = x_1(t) + x_3(t) + x_4(t) + x_5(t) + x_6(t) + x_7(t) + x_{10}(t)$, the keystream $(S(t))_{t \geq 0}$ satisfies $\Pr[S(t) = \ell(t)] = \frac{1}{2}(1 - 2^{-3})$.

Parity-checks. Let us build a parity-check as follows:

$$\ell\ell(t) = \ell(t) + \ell(t + T_{3,7}) + \ell(t + T_{4,5}) + \ell(t + T_{3,7} + T_{4,5}).$$

Therefore, this corresponds to $s = 7$ and $m = 2$ in the general description of the attack. The terms containing the sequences x_3, x_4, x_5, x_7 vanish in $\ell\ell(t)$, so $\ell\ell(t)$ depends exclusively on the sequences x_1, x_6 and x_{10}. Thus, we have

$$\ell\ell(t) = \sigma_1(t) + \sigma_6(t) + \sigma_{10}(t),$$

where

$$\sigma_i(t) = x_i(t) + x_i(t + T_{3,7}) + x_i(t + T_{4,5}) + x_i(t + T_{3,7} + T_{4,5}).$$

The period $T_{4,5}$ is 2^{51} and the period $T_{3,7}$ is smaller than 2^{49} as T_3 and T_7 have common factors, so to build those parity checks we need less than the maximal keystream length allowed.

Adding four times the approximation has the effect of multiplying the bias four times, so

$$\sigma(t) = S(t) + S(t + T_{3,7}) + S(t + T_{4,5}) + S(t + T_{3,7} + T_{4,5})$$

where $(S(t))_{t\geq 0}$ is the keystream satisfies

$$Pr[\sigma(t) = \sigma_1(t) + \sigma_6(t) + \sigma_{10}(t)] = \frac{1}{2}(1 + \eta)$$

with $\eta = 2^{-4\times 3}$.

We now decimate $\sigma(t)$ by the period of Register 1, which is involved in the parity-check, so we create like this a new parity-check:

$$\sigma'(t) = \sigma(t(2^{22} - 1)).$$

Now, we have that $\sigma'(t)$ is an approximation of $(\sigma_6(t(2^{22} - 1)) + \sigma_{10}(t(2^{22} - 1)))$ with biais $+\eta$ or $-\eta$. Then, if we did as in the previous attack in [11], the one before the new keystream limitation, where we performed an exhaustive search for the initial states of Registers 6 and 10, we would need

$$2^{3\times 4\times 2} \times 2 \times (58 - 2) \times \ln(2) = 2^{30.29}$$

parity-checks $\sigma'(t)$ to detect this bias. As we are decimating by the period of the Register 1, we would need $2^{30.29} \times 2^{22} = 2^{52.29}$ keystream bits to perform the attack, and it is over the limitation, so we cannot do that.

In the previous attack we took only the first bit of the keystream and decimated by the period of the first register $2^{30.29}$ times. What we do now is to consider the first four consecutive shifts of the keystream and for each one, we obtain a sequence of $\frac{2^{30.29}}{4} = 2^{28.29}$ bits by decimating it by the period of the first register $2^{28.29}$ times. Thus, we consider the first $2^{50.29}$ bits of the keystream and we compute the $4 \times 2^{28.29} = 2^{30.29}$ parity checks:

$$S\ (t(2^{22} - 1) + i) + S(t(2^{22} - 1) + i + T_{3,7}) + S(t(2^{22} - 1) + i + T_{4,5}) +$$
$$S\ (t(2^{22} - 1) + i + T_{3,7} + T_{4,5})$$

for $i \in \{0, \ldots, 3\}$ and $0 \leq t < 2^{28.29}$. This way, the required number of keystream bits is reduced to $2^{28.29} \times 2^{22} = 2^{50.29}$ and respects the maximal keystream length permitted.

Thus, we perform an exhaustive search over Registers 6 and 10, adapting to our new situation the algorithm introduced in [11]. We will have to compute, for each one of the previously mentioned sequences, so for each $i \in \{0, 1, 2, 3\}$, the following sum:

$$\mathcal{S} = \sum_{t'=0}^{2^{28.29}-1} \sigma(t'T_1 + i) \oplus \ell\ell(t'T_1 + i)$$

Using the decomposition

$$2^{28.29} = 2T_6 + T' \text{ with } T' = 2^{25.83},$$

we obtain

$$\begin{aligned}
\mathcal{S} &= \sum_{t'=0}^{2^{28.29}-1} \sigma(t'T_1 + i) \oplus \ell\ell(t'T_1 + i) \\
&= \sum_{k=0}^{T'} \sum_{t=0}^{2} \sigma((T_6 t + k)T_1 + i) \oplus \ell\ell((T_6 t + k)T_1 + i) \\
&+ \sum_{k=T'+1}^{T_6-1} \sum_{t=0}^{1} \sigma((T_6 t + k)T_1 + i) \oplus \ell\ell((T_6 t + k)T_1 + i) \\
&= \sum_{k=0}^{T'} \sum_{t=0}^{2} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \oplus \sigma_6((T_6 t + k)T_1 + i) \\
&+ \sum_{k=T'+1}^{T_6-1} \sum_{t=0}^{1} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \oplus \sigma_6((T_6 t + k)T_1 + i)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{k=0}^{T'} \left[(\sigma_6(kT_1 + i) \oplus 1) \left(\sum_{t=0}^{2} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \right) \right. \\
&\left. + \sigma_6(kT_1 + i) \left(3 - \sum_{t=0}^{2} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \right) \right] \\
&+ \sum_{k=T'}^{T_6-1} \left[(\sigma_6(kT_1 + i) \oplus 1) \left(\sum_{t=0}^{1} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \right) \right. \\
&\left. + \sigma_6(kT_1 + i) \left(2 - \sum_{t=0}^{1} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i) \right) \right],
\end{aligned}$$

where $\sigma(t), \sigma_6(t)$ and $\sigma_{10}(t)$ are the parity checks computed at the instant t with the keystream, the sequence generated by Register 6 and the one generated by

Register 10 respectively. Note that we have to split the sum at T', because for $k \leq T'$ we have to sum the parity checks at 3 instants but for $k > T'$ we only have to sum them at 2 instants, since $T' + 2 \times T_6 = 2^{28.29}$. The sum can be written in the previous way since $\sigma_6((T_6 t + k)T_1 + i)$ is constant for fixed values of k and i. The attack then consists of the following steps:

- We choose an initial state for Register 6, e.g. the all-one initial state. We compute and save a binary vector V_6 of length T_6, $V_6[k] = \sigma_6(k)$, where the sequence with whom we are computing $\sigma_6(k)$ is generated from the chosen initial state. The complexity of this state is $T_6 \times 2^2$ operations.
- For each possible initial state of Register 10 (so 2^{31-1} possibilities):
  - we compute and save four vectors $V_{10,i}$, where $i \in \{0, 1, 2, 3\}$, each one composed of T_6 integers of 2 bits.

$$V_{10,i}[k] = \sum_{t=0}^{q} \sigma((T_6 t + k)T_1 + i) \oplus \sigma_{10}((T_6 t + k)T_1 + i),$$

    where $q = 2$ if $k \leq T'$ and $q = 1$ if $k > T'$. The time complexity of this step is:

$$2^2 \left(3 \times 2^{25.83} + 2(2^{27} - 1 - 2^{25.83})\right)(7 + 2) = 2^2 \times 2^{28.29} \times 2^{3.1} = 2^{33.49}$$

    for each possible initial state of Register 10, where 2^2 is the number of vectors that we are computing, 7 corresponds to the number of operations required for computing each $(\sigma(t) + \sigma_{10}(t))$ and $2^{28.29} \times 2$ is the cost of summing up $2^{28.29}$ integers of 2 bits.
  - For each possible p from 0 to $T_6 - 1$:
    - we define $V_{6,i}$ of length T_6, $\forall i \in \{0, 1, 2, 3\}$: $V_{6,i}[k] = V_6[k + p + i \bmod T_6]$.

      Actually, $(V_{6,i}[k])_{k<T_6}$ corresponds to $(\sigma_6(k))_{k<T_6}$ when the initial state of Register 6 corresponds to the internal state obtained after clocking $(i + p)$ times Register 6 from the all-one initial state.
    - With the eight vectors that we have obtained

$$(V_{10,0}, \ldots, V_{10,3}, V_{6,0}, \ldots, V_{6,3}),$$

      we compute for each $i \in \{0, 1, 2, 3\}$:

$$W_i = \sum_{k=0}^{T'} \left[(V_{6,i}[k] \oplus 1)\, V_{10,i}[k] + V_{6,i}[k]\,(3 - V_{10,i}[k])\right] + \sum_{k=T'+1}^{T_6-1} \left[(V_{6,i}[k] \oplus 1)\, V_{10,i}[k] + V_{6,i}[k]\,(2 - V_{10,i}[k])\right].$$

      When we do this with the correct initial states of Registers 6 and 10, we will find an important bias for the four W_i.

The complexity of this point would be, for each p $2^2 \times T_6 \times 8 = 2^{32}$, so $2^{32} \times 2^{27} = 2^{59}$. The total number of memory accesses for each possible initial state of Register 10 is

$$4 \times T_6 + 4 \times 2 \times T_6 = 2^{30.3},$$

where the first term corresponds to the storage of $V_{10,i}$ and the second one corresponds to the accesses to $V_{6,i}$ and $V_{10,i}$ to compute W_i. But we can speed up the process by defining a new vector,

$$V'_{10,j}[k] = V_{10,j}[k] + ct$$

where $ct = 0$ if $k \leq T'$ and $ct = 0.5$ if $k > T'$.
Then, for each i we are going to compute:

$$\sum_{k'=0}^{T_6-1} (-1)^{V_{6,i}[k+p]} \left(V'_{10,i}[k] - \frac{3}{2} \right) + (T' \times 1.5 + (T_6 - T') \times 1).$$

The issue is now to find the p that maximizes this sum, this is the same as computing the maximum of the crosscorrelation of two sequences of length T_6. We can do that efficiently using a fast Fourier transform as explained in [1, pages 306-312]. The final complexity for computing this sum will be in $T_6 \log_2(T_6)$.

Thus, the total complexity of this state will be $4T_6 \log_2(T_6) \approx 2^{34}$.

We now compute the false alarm and the non detection probabilities. First of all we consider as the bias threshold $S = 0.55 \times \eta = 2^{-12.86}$. Let n be the length of the sequences used and i be the number of sequences. The false alarm probability for i sequences is the probability that, while trying wrong initial states of Registers 6 and 10 (which would generate random sequences) we find a bias higher than $2^{-12.86}$ or lower than $-2^{-12.86}$ for all the i W_j. Using Chernoff's bound on the tail of the binomial distribution we get:

$$P_{fa4}(S) = (P_{fa1}(S))^i \leq (2e^{-2S^2 n})^i,$$

where P_{fa1} is the false alarm probability for one sequence. In our case $n = 2^{28.29}$ and $i = 4$, so $P_{fa4} = 2^{-64.29}$. The number of initial states that will pass the test without being the correct one will be $(2^{56} - 1) \times 2^{-64.29} = 2^{-8.29}$. The non detection probability is the probability that while trying the correct initial states of Registers 6 and 10 we find a bias between $-2^{-12.86}$ and $2^{-12.86}$. For one sequence it will be:

$$P_{nd1}(S) \leq 2e^{-2(\eta - S)^2 n},$$

So the probability of non detection for i sequences, that is, the probability of not detecting the threshold at one of the i W_j will be:

$$P_{nd4}(S) = 1 - (1 - P_{nd1}(S))^i.$$

It is used only for the correct initial states. As we can see, it increases with i, the number of used sequences. In our case, $i = 4$, leading to $P_{nd4}(S) = 2^{-9.43}$.

The time complexity is, finally

$$2^{L_{10}-1} \times \left[2^{33.69} + 4T_6 \log_2 4T_6\right] + T_6 \times 2^2 = 2^{64.85} \text{ steps.}$$

The required number of keystream bits is

$$2^{28.29} \times T_1 + T_3T_7 + T_4T_5 = 2^{50.29} + 2^{48.1} + 2^{51} < 2^{52}.$$

The memory used is

$$2^{30} + 2^{29} = 2^{30.58},$$

where 2^{30} is the size of the four $V_{10,i}$ vectors and 2^{29} of the $V_{6,i}$ vectors.

4 Distinguishing Attack against Achterbahn-128

Now, we present a distinguishing attack against the 128-bit version of Achterbahn which also recovers the initial states of two registers.

We consider the following approximation of the combining function F:

$$\ell(x_0, \ldots, x_{12}) = x_0 + x_1 + x_2 + x_3 + x_4 + x_7 + x_8 + x_9 + x_{10}.$$

Then, for $\ell(t) = x_0(t)+x_1(t)+x_2(t)+x_3(t)+x_4(t)+x_7(t)+x_8(t)+x_9(t)+x_{10}(t)$, we have $\Pr[S(t) = \ell(t)] = \frac{1}{2}(1 + 2^{-3})$.

Parity-checks. If we build a parity check as follows:

$$\ell\ell\ell(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} \ell(t + \tau),$$

the terms containing the sequences x_1, x_2, x_3, x_8, x_9, x_{10} will disappear from $\ell\ell\ell(t)$, so $\ell\ell\ell(t)$ depends exclusively on the sequences x_0, x_4 and x_7:

$$\ell\ell\ell(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} x_0(t+\tau) + x_4(t+\tau) + x_7(t+\tau) = \sigma_0(t) + \sigma_4(t) + \sigma_7(t),$$

where $\sigma_0(t)$, $\sigma_4(t)$ and $\sigma_7(t)$ are the parity-checks computed on the sequences generated by Registers 0, 4 and 7. Adding eight times the approximation has the effect of multiplying the bias eight times, so the bias of

$$\sigma(t) = \sum_{\tau \in \langle T_{3,8}, T_{1,10}, T_{2,9} \rangle} S(t + \tau),$$

where $(S(t))_{t\geq 0}$ is the keystream, is $2^{-8\times 3}$. So:

$$\Pr[\sigma(t)+\sigma_0(t)+\sigma_4(t)+\sigma_7(t)=1]=\frac{1}{2}(1-\varepsilon^8).$$

This means that we need $2^{3\times 8\times 2}\times 2\times(74-3)\times\ln(2)=2^{54.63}$ values of $\sigma(t)+\sigma_0(t)+\sigma_4(t)+\sigma_7(t)$ to detect this bias, when we perform an exhaustive search on Registers 0, 4 and 7.

We use the previously proposed algorithm for the attack of Achterbahn-128 for computing the sum $\sigma(t)+\sigma_0(t)+\sigma_4(t)+\sigma_7(t)$ over all values of t. This algorithm has a lower complexity than an exhaustive search for the initial states of the Registers 0, 4 and 7 simultaneously. We use it considering Register 0 and Register 4 together.

The complexity is going to be, finally

$$2^{L_0-1}\times 2^{L_4-1}\times\left[2^{54.63}\times\left(2^4+2^{4.7}\right)+T_7\log T_7\right]+T_7\times 2^3=2^{104}\text{ steps.}$$

The required keystream length is:

$$2^{54.63}+T_{1,10}+T_{2,9}+T_{3,8}=2^{54.63}+2^{53}+2^{53}+2^{53}<2^{56}\text{ bits.}$$

The memory used is

$$2^{32}+2^{28}=2^{32.08},$$

where 2^{32} is the size of the V_{0-4} vector and 2^{28} of the V_7 vector.

5 Recovering the Key

As explained in the previous attacks [11] and introduced in [9], we can recover the key with a variant of a meet-in-the-middle attack once we have found the initial state of some registers. The time complexity of this part of the attack is smaller than the one of the previously described distinguishing attack that we need to get the initial states of several registers. So the complexity of the total key-recovery attack is the same one as for the distinguishing attacks.

6 Conclusion

We have proposed an attack against Achterbahn-80 in $2^{64.85}$ steps where fewer than 2^{52} bits are needed. That is $2^{64.85}$ boolean operations, which makes it much more efficient than a brute force attack. The memory needed for this attack is $2^{30.58}$. An attack against Achterbahn-128 is also proposed in 2^{104} steps where fewer than 2^{56} bits of keystream are required. The memory needed is 2^{32}. After that we can recover the key of Achterbahn-80 with a complexity of 2^{40} in time and 2^{41} in memory (the time complexity is less than for the distinguishing part of the attack). For Achterbahn-128 we can recover the key with a complexity of 2^{73} in time and 2^{48} in memory. After those attacks, the authors proposed a new keystream limitation for both Achterbahn-128/80 [7]. This new limitation is 2^{44}. With this limitation the known attacks are not applicable.

References

1. Blahut, R.E.: Fast Algorithms for Digital Signal Processing. Addison Wesley, Reading (1985)
2. Canteaut, A., Trabbia, M.: Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 573–588. Springer, Heidelberg (2000)
3. Gammel, B.M., Gottfert, R., Kniffler, O.: The Achterbahn stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/002 (2005), http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf
4. Gammel, B.M., Gottfert, R., Kniffler, O.: Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/001 (2006), http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf
5. Gammel, B.M., Gottfert, R., Kniffler, O.: Status of Achterbahn and tweaks. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/027 (2006), http://www.ecrypt.eu.org/stream/papersdir/2006/027.pdf
6. Gammel, B.M., Gottfert, R., Kniffler, O.: Achterbahn-128/80: Design and analysis. In: ECRYPT Network of Excellence - SASC Workshop Record, pp. 152–165 (2007)
7. Gammel, B.M., Gottfert, R.: On the frame length of Achterbahn-128/80. In: IEEE Information Theory Workshop on Information Theory for Wireless Networks, pp. 91–95 (2007)
8. Hell, M., Johansson, T.: Cryptanalysis of Achterbahn-version 2. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 45–55. Springer, Heidelberg (2007)
9. Hell, M., Johansson, T.: Cryptanalysis of Achterbahn-128/80. IET Information Security 1(2) (2007)
10. Johansson, T., Meier, W., Muller, F.: Cryptanalysis of Achterbahn. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 1–14. Springer, Heidelberg (2006)
11. Naya-Plasencia, M.: Cryptanalysis of Achterbahn-128/80. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 73–86. Springer, Heidelberg (2007)