Cryptanalysis of Achterbahn-128/80

María Naya-Plasencia*

INRIA, projet CODES, Domaine de Voluceau
78153 Le Chesnay Cedex, FRANCE
Maria.Naya_Plasencia@inria.fr

Abstract. This paper presents two key-recovery attacks against Achterbahn-128/80, the last version of one of the stream cipher proposals in the eSTREAM project. The attack against the 80-bit variant, Achterbahn-80, has complexity 2^{61}. The attack against Achterbahn-128 requires $2^{80.58}$ operations and 2^{60} keystream bits. These attacks are based on an improvement of the attack due to Hell and Johansson against Achterbahn version 2. They mainly rely on an algorithm that makes profit of the independence of the constituent registers.

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

1 Introduction

Achterbahn [4, 6] is a stream cipher proposal submitted to the eSTREAM project. After the cryptanalysis of the first two versions [10, 9], it has moved on to a new one called Achterbahn-128/80 [5] 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 is limited to 2^{63}.

We present here two attacks against both generators. The attack against the 80-bit variant, Achterbahn-80, has complexity 2^{61}. The attack against Achterbahn-128 requires $2^{80.58}$ operations and 2^{61} keystream bits. These attacks are based on an improvement of the attack against Achterbahn version 2 and also on an algorithm that makes profit of the independence of the constituent registers.

The paper is organized as follows. Section 2 presents the main specifications of Achterbahn-128/80. Section 3 then describes the general principle of the attack proposed by Hell and Johansson [9] against the previous version of the cipher Achterbahn version 2, since our attacks rely on a similar technique. We also exhibit a new attack against Achterbahn version 2 with complexity 2^{53}, while the best previously known attack had complexity 2^{64}. Section 4 then presents two

* This work was supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. 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.

distinguishing attacks against Achterbahn-80 and Achterbahn-128 respectively. Section 5 describes how this previous distinguishing attacks can be transformed into key-recovery attacks.

2 Main specifications of Achterbahn-128/80

2.1 Main specifications of Achterbahn-128

Achterbahn-128 is a keystream generator, consisting 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 $2^{L_i} - 1$. The 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 [5].

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 input on 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 previous versions. 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.

3 Attack against Achterbahn version 2 in 2^{53}

3.1 Principle of Hell and Johansson attack

Achterbahn version 2 was the previous version of Achterbahn. The main and most important differences to this last one, which are used by the attack are that:

- it had 10 registers, with lengths between 19 and 32 bits,
- the Boolean function, f, had correlation immunity order 5.

This version has been broken by Hell and Johansson [9] using a quadratic approximation. Their attack is a distinguishing attack that 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 then consists of an exhaustive search on 2^k initial states. For each of those states, the parity-check relation is computed for N samples in order to detect the bias. As noticed in [8], the usual estimate [9–11] of the number of samples which are required for distinguishing the keystream,

$$N \sim \frac{1}{\eta^2},$$

is a bit underestimated. Actually, this problem can be seen as a decoding problem where the received word corresponds to the sequence formed by the N parity-check evaluations. And this received word can be seen as the result of the transmission of a codeword through a binary symmetric channel with cross-over probability p. Then, the number of samples N required for decoding is

$$N = \frac{k}{C(p)},$$

where $C(p)$ is the capacity of the channel, i.e.,

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p).$$

Moreover, when $p = \frac{1}{2}(1+\eta)$ with $|\eta| \ll 1$, we have $C(p) \sim \frac{\eta^2}{2\ln(2)}$, leading to

$$N \sim \frac{2k\ln 2}{\eta^2},$$

where 2^k is the number of possible initial states of the guessing registers, as we will see.

The attack proposed by Hell and Johansson exploits a quadratic approximation q of the combining function f:

$$Q(y_1, \ldots, y_n) = \sum_{j=1}^{s} y_{i_j} + \sum_{i=1}^{m} (y_{j_i} y_{k_i})$$

with m quadratic terms and which satisfies

$$\Pr[F(y_1, \ldots, y_n) = Q(y_1, \ldots, y_n)] = \frac{1}{2}(1 + \varepsilon).$$

We build the parity-check equations, as the ones introduced by [10], that make disappear the quadratic terms by summing up:

$$q(t) = \sum_{j=1}^{s} x_{i_j}(t) + \sum_{i=1}^{m} x_{j_i}(t) x_{k_i}(t)$$

at 2^m different epochs $(t+\tau)$, where τ varies in the set of the linear combinations with $0-1$ coefficients of $T_{j_1}T_{k_1}, T_{j_2}T_{k_2}, \ldots, T_{j_m}T_{k_m}$, where T_i denotes the period of Ri. In the following, this set is denoted by $\langle T_{j_1}T_{k_1}, \ldots, T_{j_m}T_{k_m} \rangle$, i.e.,

$$\mathcal{I} = \langle T_{j_1}T_{k_1}, \ldots, T_{j_m}T_{k_m} \rangle = \left\{ \sum_{i=1}^{m} c_i T_{j_i} T_{k_i}, c_1, \ldots, c_m \in \{0, 1\} \right\}.$$

This leads to a parity-check sequence pc defined by:

$$pc(t) = \sum_{\tau \in \mathcal{I}} q(t + \tau) = \sum_{\tau \in \mathcal{I}} \left(x_{i_1}(t + \tau) + \ldots + x_{i_s}(t + \tau) \right).$$

We then decimate the sequence $(pc(t))_{t \geq 0}$ by the periods of r sequences among $(x_{i_1}(t))_{t \geq 0}, \ldots, (x_{i_s}(t))_{t \geq 0}$. We can suppose here without loss of generality that the periods of the first r sequences have been chosen. Now a new parity-check, pc_r, can be defined by:

$$pc_r(t) = pc(tT_{i_1} \ldots T_{i_r}).$$

This way, the influence of those r registers on the parity-check $pc_r(t)$ corresponds to the addition of a constant for all $t \geq 0$, so it will be 0 or 1 for all the parity-checks.

Now, the attack consists in performing an exhaustive search for the initial states of the $(s-r)$ remaining registers, i.e. those of indices $i_{r+1}, \ldots, i_s$. For each possible values for these initial states, we compute the sequence:

$$\sigma(t) = \sum_{\tau \in \langle T_{j_1}T_{k_1}, \ldots, T_{j_m}T_{k_m} \rangle} \left[S(tT_{i_1} \ldots T_{i_r} + \tau) + \sum_{j=r+1}^{s} x_{i_j}(tT_{i_1} \ldots T_{i_r} + \tau) \right] \quad (1)$$

We have

$$\Pr[\sigma(t) = 0] \geq \frac{1}{2}(1 + \varepsilon^{2^m}).$$

It has been recently observed by Hell and Johansson that the total bias may be much higher than this bound. However, it can be shown that equality holds in some particular cases, as noted in [7]. An interesting case of equality is when f is v-resilient, and we build parity-checks from the terms appearing in a linear approximation of $(v+1)$ variables (see Appendix). This also provides the bias of the parity-checks obtained in [9, 8] from some quadratic approximations, since they can also be derived from such linear approximations.

This result is going to be used all along our attacks, as we will work with linear approximations of $(v+1)$ variables.

3.2 Complexity

Using the previously computed bias, we can distinguish the keystream $(S(t))_{t\geq 0}$ from a random sequence and also recover the initial states of $(s-r)$ constituent registers.

- We will have 2^m terms in each parity-check. That means that we need to compute $\varepsilon^{-2^{m+1}} \times 2 \times \sum_{j=r+1}^{s}(L_{i_j}-1) \times \ln(2) = 2^{n_b 2^{m+1}} \times 2 \times \sum_{j=r+1}^{s}(L_{i_j}-1) \times \ln(2)$ values of $\sigma(t)$ for mounting the distinguishing attack, where $n_b = \log_2 \varepsilon^{-1}$. Besides, $\sigma(t)$ is defined by (1), implying that the attack requires

  $$2^{n_b 2^{m+1}+\sum_{j=1}^{r} L_{i_j}} \times 2 \times \sum_{j=r+1}^{s} (L_{i_j}-1) \times \ln(2) + \sum_{i=1}^{m} 2^{L_{j_i}+L_{k_i}} \text{ keystream bits,}$$

  where L_{i_j} are the lengths of the registers associated to the periods by which we have decimated, and the last term corresponds to the maximal distance between the bits involved in each parity-check.
- Time complexity will be

  $$2^m 2^{n_b 2^{m+1}+\sum_{j=r+1}^{s}(L_{i_j}-1)} \times 2 \times \sum_{j=r+1}^{s} (L_{i_j}-1) \times \ln(2)$$

  where $i_{r+1}, \ldots, i_s$ are the indices of the registers over whom we have made an exhaustive search and whose initial state we are going to find.

3.3 Example with Achterbahn version 2

Hell and Johansson [9] have used this attack against Achterbahn version 2 with the following quadratic approximation:

$$Q(x_1, \ldots, x_{10}) = x_1 + x_2 + x_3 x_8 + x_4 x_6.$$

Then, they decimate by the period of the second register, whose length is 22. After that, they make an exhaustive search over the first register, of length 19. Time complexity will be 2^{67} and data complexity 2^{64} (the complexity given in [9], equal to $2^{59.02}$, is obtained by using the estimation $N = \varepsilon^{-2}$ instead of the one given in Section 3.1). Using the small lengths of the registers, time complexity can be reduced below data complexity, so the overall complexity of the attack will be 2^{64}.

3.4 Improvement of the attack against Achterbahn version 2

We are going to improve the previously described attack against Achterbahn version 2 and we reduce the complexity to 2^{53}.

For this attack, we use the idea of associating the variables in order to reduce the number of terms that we will have in the parity-checks. The only negative effect that this could have on the final complexity of the attack is to enlarge the number of required keystream bits; but being careful, we make it stay the same while reducing the time complexity.

The chosen approximation. At first, we searched for all the quadratics approximations of f with one and two quadratic terms, as the original attack presented by Hell and Johansson was based on a quadratic approximation. Finally, after looking for a trade-off between the number of terms, the number of variables, the bias, etc., we found that none quadratic approximation was better for this attack than linear ones. It is worth noticing that, since the combining function f is 5-resilient, any approximation of f involves at least 6 input variables. Moreover, the highest bias corresponding to an approximation of f by a 6-variable function is achieved by a function of degree one as proved in [3]. After analyzing all linear approximations of the Boolean combining function, we found that the best one was:

$$g(x_1, \ldots, x_{10}) = x_8 + x_6 + x_4 + x_3 + x_2 + x_1.$$

We have $f(x_1, \ldots, x_{10}) = g(x_1, \ldots, x_{10})$ with a probability of $\frac{1}{2}(1 + 2^{-3})$.

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

$$ggg(t) = g(t) + g(t + T_1T_8) + g(t + T_2T_6) + g(t + T_1T_8 + T_2T_6),$$

with

$$g(t) = x_8(t) + x_6(t) + x_4(t) + x_3(t) + x_2(t) + x_1(t).$$

The terms x_8, x_6, x_2, x_1 will disappear and, so, $ggg(t)$ is a sequence that depends uniquely on the sequences x_3 and x_4. Adding four times the approximation has the effect of multiplying the bias four times, so the bias of

$$\sigma(t) = S(t) + S(t + T_1T_8) + S(t + T_2T_6) + S(t + T_1T_8 + T_2T_6)$$

is $2^{-3\times4} = 2^{-12}$ because 4 is the number of terms in $ggg(t)$. That means that we will need $2^{3\times4\times2} \times 2 \times (L_4 - 1) \times \ln(2) = 2^{29}$ values of the parity-check for detecting this bias. If we decimate $ggg(t)$ by the period of register 3, we will need

$$2^{29}T_3 + T_1T_8 + T_2T_6 = 2^{29+23} + 2^{29+19} + 2^{27+22} = 2^{52} \text{ bits of keystream,}$$

and time complexity will be $2^{29} \times 2^{L_4-1} = 2^{53}$ as we only guess the initial state of register 4. This complexity is 2^{53} while the complexity of the previous attack was equal to 2^{64}.

4 Distinguishing attacks against Achterbahn-128/80

4.1 Distinguishing attack against Achterbahn-80

This attack is very similar to the improvement of the attack against Achterbahn version 2 which has been described in the previous section.

Our attack exploits the following linear approximation of the combining function G:

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

Since G is 6-resilient, ℓ is the best approximation by a 7-variable function.

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:

$$\ell\ell(t) = \ell(t) + \ell(t + T_4T_7) + \ell(t + T_6T_5) + \ell(t + T_4T_7 + T_6T_5).$$

The terms containing the sequences x_4, x_5, x_6, x_7 vanish in $\ell\ell(t)$, so $\ell\ell(t)$ depends exclusively on the sequences x_1, x_3 and x_{10}.

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

$$\sigma(t) = S(t) + S(t + T_7T_4) + S(t + T_6T_5) + S(t + T_7T_4 + T_6T_5)$$

where $(S(t))_{t\geq 0}$ is the keystream, is $2^{-4\times 3}$.

We now decimate $\sigma(t)$ by the period of the R_{10}, which is involved in the parity-check, so we create like this a new parity-check $\sigma'(t) = \sigma(t(2^{31} - 1))$.

Then, the attack performs an exhaustive search for the initial states of registers 1 and 3. Then we need $2^{3\times 4\times 2} \times 2 \times (46 - 2) \times \ln(2) = 2^{30}$ parity-checks $\sigma'(t)$ to detect this bias. Its time complexity is $2^{30} \times 2^{L_1+L_3-2} = 2^{74}$.

The number of keystream bits that we need is $2^{30} \times T_{10} + T_4T_7 + T_6T_5 = 2^{61}$.

4.2 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:

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

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

Parity-checks. The period of any sequence obtained by combining the registers 0, 3 and 7 is equal to $\mathrm{lcm}(T_0, T_3, T_7)$, i.e. $2^{59.3}$ as T_0 T_3 and T_7 have common divisors. We are going to denote this value by $T_{0,3,7}$.

If we build a parity check as follows:

$$\ell\ell\ell(t) = \sum_{\tau \in \langle T_{0,3,7}, T_{4,10}, T_{8,9} \rangle} \ell(t+\tau),$$

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

$$\begin{aligned}
\ell\ell\ell(t) &= \sum_{\tau \in \langle T_{0,3,7}, T_{4,10}, T_{8,9} \rangle} \ell(t+\tau) \\
&= \sum_{\tau \in \langle T_{0,3,7}, T_{4,10}, T_{8,9} \rangle} x_1(t+\tau) + x_2(t+\tau) \\
&= \sigma_1(t) + \sigma_2(t),
\end{aligned}$$

where $\sigma_1(t)$ and $\sigma_2(t)$ are the parity-checks computed over the sequences generated by NLFSRs 1 and 2.

Adding eight times the approximation has the effect of multiplying the bias eight times, so the bias of $\sigma(t) = \sum_{\tau \in \langle T_{0,3,7}, T_{4,10}, T_{8,9} \rangle} S(t+\tau)$ where $(S(t))_{t \geq 0}$ is the keystream, is $2^{-8 \times 3}$. So:

$$\Pr[\sigma(t) + \sigma_1(t) + \sigma_2(t) = 1] = \frac{1}{2}(1 - \varepsilon^8).$$

This means that we need $2^{3 \times 8 \times 2} \times 2 \times (45-2) \times \ln(2) = 2^{54}$ values of $\sigma(t)+\sigma_1(t)+\sigma_2(t)$ to detect this bias, when we perform an exhaustive search on registers 1 and 2.

We now describe an algorithm for computing the sum $\sigma(t)+\sigma_1(t)+\sigma_2(t)$ over all values of t. This algorithm has a lower complexity than the trivial algorithm which consists on computing the 2^{54} parity-checks for all the initial states of the registers 1 and 2. Here we use $(2^{54}-2^8)$ values of t since $(2^{54}-2^8) = T_2 \times (2^{31}+2^8)$. We can write it down as follows:

$$\begin{aligned}
\sum_{t'=0}^{2^{54}-2^8-1} \sigma(t') \oplus \ell\ell\ell(t') &= \sum_{k=0}^{T_2-1} \sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2 t + k) \oplus \ell\ell\ell(T_2 t + k) \\
&= \sum_{k=0}^{T_2-1} \sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2 t + k) \oplus \sigma_1(T_2 t + k) \oplus \sigma_2(T_2 t + k) \\
&= \sum_{k=0}^{T_2-1} \left[(\sigma_2(k) \oplus 1) \left(\sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2 t + k) \oplus \sigma_1(T_2 t + k) \right) + \right. \\
&\qquad \left. \sigma_2(k) \left((2^{31} + 2^8) - \sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2 t + k) \oplus \sigma_1(T_2 t + k) \right) \right],
\end{aligned}$$

since $\sigma_2(T_2t + k)$ is constant for a fixed value of k.

At this point, we can obtain $\sigma(t)$ from the keystream and we can make an exhaustive search for the initial state of register 1. More precisely:

- We choose an initial state for register 2, e.g. the all one initial state. We compute and save a binary vector V_2 of length T_2:

$$V_2[k] = \sigma_2(k),$$

  where the sequence x_2 is generated from the chosen initial state. The complexity of this step is $T_2 \times 2^3$ operations.
- For each possible initial state of register 1:
  - we compute and save a vector V_1 composed of T_2 integers of 32 bits.

$$V_1[k] = \sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2t + k) \oplus \sigma_1(T_2t + k).$$

    The complexity of this step is $2^{54} \times (2^4 + 2^5) = 2^{59.58}$ for each possible initial state of register 1, where 2^4 corresponds to the number of operations required for computing each $(\sigma(t) + \sigma_1(t))$ and $(2^{31} + 2^8) \times 2^5 = (2^{31} + 2^8) \times 32$ is the cost of summing up $2^{31} + 2^8$ integers of 32 bits.
  - For each possible i from 0 to $T_2 - 1$:
    * we define V_2' of length T_2:

$$V_2'[k] = V_2[k + i \mod T_2].$$

      Actually, $(V_2'[k])_{k<T_2}$ corresponds to $(\sigma_2(k))_{k<T_2}$ when the initial state of register 2 corresponds to the internal state after clocking register 2 i times from the all-one initial state.
    * With the two vectors that we have obtained, we compute:

$$\sum_{k=0}^{T_2-1} \left[(V_2'[k] \oplus 1)\, V_1[k] + V_2'[k] \left(2^{31} + 2^8 - V_1[k]\right)\right]. \qquad (2)$$

When we do this with the correct initial states of registers 1 and 2, we will find the expected bias. The major difference with the classical exhaustive search used in [9, 8, 10] is that the sequence $V_1[k]$ is computed independently of the choice of the initial state of R_2. As a comparison, the classical algorithm has time complexity 2^{102}.

for each possible initial state of $R1$ **do**
  for $k = 0$ to $T_2 - 1$ **do**
    $V_1[k] = \sum_{t=0}^{2^{31}+2^8-1} \sigma(T_2 t + k) \oplus \sigma_1(T_2 t + k)$
  end for
  for each possible initial i state of $R2$ **do**
    for $k = 0$ to $T_2 - 1$ **do**
      $V_2'[k] = V_2[k + i \mod T_2]$
    end for
    $\sum_{k=0}^{T_2-1} (V_2'[k] \oplus 1)\, V_1[k] + V_2'[k] \;\; 2^{31} + 2^8 - V_1[k]$
    if we find the bias **then**
      return the initial states of $R1$ and $R2$
    end if
  end for
end for

Table 1: Algorithm for finding the initial states of registers 1 and 2

The total time complexity of the attack is going to be:

$$2^{L_1-1} \times \left[2^{54} \times \left(2^4 + 2^5\right) + T_2 \times 2 \times T_2 \times 2^5\right] + T_2 \times 2^3 = 2^{80.58},$$

where $2 \times T_2 \times 2^5$ is the time it takes to compute the sum described by (2). Actually, we can speed up the process by rewriting the sum (2) in the following way

$$\sum_{k=0}^{T_2-1} (-1)^{V_2[k+i]} \left(V_1[k] - \frac{2^{31}+2^8}{2}\right) + T_2 \frac{2^{31}+2^8}{2}.$$

The issue is now to find the i that maximizes this sum, this is the same as computing the maximum of the crosscorrelation of two sequences of length T_2. We can do that efficiently using a fast Fourier transform as explained in [1, pages 306-312]. The final complexity will be in $O(T_2 \log T_2)$. Anyway, this does not change our total complexity as the higher term is the first one.

The complexity is going to be, finally:

$$2^{L_1-1} \times \left[2^{54} \times \left(2^4 + 2^5\right) + O(T_2 \log T_2)\right] + T_2 \times 2^3 = 2^{80.58}.$$

The length of keystream needed is $T_{0,3,7} + T_{4,10} + T_{8,9} + 2^{54} < 2^{61}$ bits.

We can apply the algorithm to the attack against Achterbahn-80 described in Section 4.1 and its time complexity will be reduced to:

$$2^{L_1-1} \times \left[2^{30} \times \left(2^3 + 2^{2.59}\right) + O(T_3 \log T_3)\right] + T_3 \times 2^2 = 2^{54.8}.$$

4.3 Attack with a new keystream limitation

Recently, the authors of Achterbahn have proposed a new limitation of the keystream length [7], which is 2^{52} for Achterbahn-80 and 2^{56} for Achterbahn-128. Those limitations are not restrictive enough to prevent the cipher from being cryptanalysed. In fact, we can mount an attack against the 128-bit version which is very similar to the last one with the same linear approximation, where

the sequences considered for building the parity-checks are generated by only two terms (so R_1 and R_{10}, R_2 and R_9, R_3 and R_8). Then we perform an exhaustive search over registers 0, 4 and 7 with the previously described the algorithm, where we consider register 0 and register 4 together. The complexity is, finally:

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

The length of keystream needed is

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

For Achterbahn-80 there is also a succesfull attack which is only slightly different from the one we have previously described [12].

5 Recovering the key

As explained by Hell and Johansson in [8], if we recover the initial states of all the registers, we will be able to retrieve the key as all the initialization steps which do not involve the key become invertible. It is easy to show that once we have found the initial states of two registers, the complexity of finding the remaining ones will be lower (for the other registers appearing in the used approximation it is quite obvious: we apply the same method but simplified, as now we know two variables. For the other registers we can use the same method but with other linear approximations making profit of the already-known variables). Once we have found the initial states of all the registers, we can invert all the initializing steps until the end of the second step, which corresponds to the introduction of the key bits. At this point, there are two methods proposed in [8]. The first one is clocking backwards register i ($|k| - L_i$) times for each i. We do this for all the possibles values of the last $|k| - L_m$ key bits, where L_m is 21 for Achterbahn-128 and 22 for Achterbahn-80. When all the registers have the same first L_m bits, we have found the correct $|k| - L_m$ bits of the key. The second method proposed is a meet-in-the-middle attack with time-memory tradeoff as explained in [10]. It leads to a complexity of:

- For Achterbahn-80: 2^{58} in time or 2^{40} in memory and 2^{40} in time.
- For Achterbahn-128: 2^{107} in time or 2^{40} in memory and 2^{88} in time.

We can do better. We are going to explain the technique for Achterbahn-128. The idea is that we do not need to invert all the clocking steps in the meet-in-the-middle attack, if we split the key into 2 parts composed of the first 40 bits and the last 88 bits, we could make an exhaustive search for the first part and store in a table the states of the registers obtained after applying the initialization process for each set of 40 bits. Then, if we make an exhaustive search through the 88 remaining key bits, and we clock backwards the registers from the known states, we will find a match in the table. But we do not need to make this search over all the 88 remaining bits. Instead, we make it through the last 73 bits (that means that 15 rounds are not inverted). At the end of doing this, we need that,

for all i, the first $L_i - 15$ bits of the state of register i match with the last $L_i - 15$ bits of the states of the registers saved in the table. For instance, for the register 0 we will have a match on 6 bits, and for register 12 we will have 18. We do not have to worry about matches coming from wrong values of the 73 bits since the number of such false alarms is:

$$2^{88-15} \times \frac{2^{40}}{2^{329-13\times 15}} = 2^{-21},$$

as $(329 - 13 \times 15)$ is the number of bits we consider for a match, 2^{40} is the size of the table, and 2^{73} is the number of possibilities for the exhaustive search. As we can see, with such a match we have found 113 bits of the key. The other 15 can be found with very low complexity by clocking the registers until finding the desired state. So the final complexity for the step of retrieving the key in Achterbahn-128 once we have the initial states of all the registers is 2^{73} in time and $2^{40} \times (329 - 13 \times 15) \simeq 2^{48}$ in memory. If we do the same thing with Achterbahn-80, we could have a complexity of 2^{40} in time and 2^{41} in memory.

6 Conclusion

We have proposed an attack against Achterbahn-80 in 2^{55} operations, so we can consider as the total complexity the data complexity which is equal to 2^{61}, since it is bigger. An attack against Achterbahn-128 is also proposed in $2^{80.58}$ where fewer than 2^{61} bits of keystream are required. 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. The complexities of the best attacks against all versions of Achterbahn are summarized in the following table:

version	data complexity	time complexity	references
v1 (80-bit)	2^{32}	2^{55}	[10]
v2 (80-bit)	2^{64}	2^{67}	[9]
v2 (80-bit)	2^{52}	2^{53}	
v80 (80-bit)	2^{61}	2^{55}	
v128 (128-bit)	2^{60}	$2^{80.58}$	

Table 2: Attacks complexities against all versions of Achterbahn (Each complexity corresponds to the best key-recovery attack).

Acknowledgments

The author would like to thank Anne Canteaut for her helpful advice, discussions, suggestions and support. Also, many thanks to Yann Laigle-Chapuy, Andrea Röck and Frederic Didier for their useful comments.

References

1. R. E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison Wesley, 1985.
2. A. Canteaut and P. Charpin. Decomposing bent functions. *IEEE Transactions on Information Theory*, 49(8):2004–2019, 2003.
3. A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer-Verlag, 2000.
4. B. M. Gammel, R. Gottfert, and O. Kniffler. The Achterbahn stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/002, 2005. `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`.
5. B. M. Gammel, R. Gottfert, and O. Kniffler. Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/001, 2006. `http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf`.
6. B. M. Gammel, R. Gottfert, and O. Kniffler. Status of Achterbahn and tweaks. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/027, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/027.pdf`.
7. B. M. Gammel, R. Gottfert, and O. Kniffler. Achterbahn-128/80: Design and analysis. In *ECRYPT Network of Excellence - SASC Workshop Record*, pages 152–165, 2007.
8. M. Hell and T. Johansson. Cryptanalysis of Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/054, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/054.pdf`.
9. M. Hell and T. Johansson. Cryptanalysis of Achterbahn-version 2. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/042, 2006. `http://www.ecrypt.eu.org/stream/ciphers/achterbahn/achterbahn.pdf`.
10. T. Johansson, W. Meier, and F. Muller. Cryptanalysis of Achterbahn. In *Advances in Cryptology - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.
11. M. Naya-Plasencia. Cryptanalysis of Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/055, 2006. `http://www.ecrypt.eu.org/stream/papersdir/2006/055.pdf`.
12. M. Naya-Plasencia. Cryptanalysis of Achterbahn-128/80 with a new keystream limitation. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/004, 2007. `http://www.ecrypt.eu.org/stream/papersdir/2007/004.pdf`.

A On the biases of parity-checks derived from linear approximations

Proposition 1. *Given a v-resilient Boolean function f, the bias of a parity-check built from a $(v+1)$-variable linear approximation of f with bias ε is ε raised to the power of the number of terms in the parity check.*

Proof. Let f be a v-resilient Boolean function of n variables and $\ell = x_{j_0} + \ldots + x_{j_v} = \alpha \cdot x$ be a linear approximation of f with bias ε. We can now build $g(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) + \ell(x_{j_0}, \ldots, x_{j_v})$. We have

$$\Pr[g(x_1, \ldots, x_n) = 0] = \frac{1}{2}(1 + \varepsilon).$$

Let W denote the subspace of $\mathbf{F}_2^n$ spanned by the basis vectors $e_{j_0}, \ldots, e_{j_v}$, and let V be in direct sum with W. Then, for any n-variable function f, and any $a \in \mathbf{F}_2^{v+1}$, $f_{|a+V}$ denotes the restriction of f to $(a+V)$. In other words, $f_{|a+V}$ is the function of $(n-v-1)$ variables derived from f when $x_{j_0}, \ldots, x_{j_v}$ are fixed and equal to $a_0, \ldots, a_v$. If we build the parity-checks with g considering the sequences defined by the terms of the linear approximation we will have:

$$\Pr\left[\sum_{\tau \in \langle T_{j_0}, \ldots, T_{j_v} \rangle} g(x_1(t+\tau), \ldots, x_n(t+\tau)) = 0\right]$$
$$= \frac{1}{2^{v+1}} \sum_{a \in \mathbf{F}_2^{v+1}} \Pr\left[\sum_{\tau \in \langle T_{j_0}, \ldots, T_{j_v} \rangle} g_{|a+V}(x_1(t+\tau), \ldots, x_n(t+\tau)) = 0\right].$$

It is quite obvious that the variables appearing in the terms of the sum over τ are independent, as the variables that could be repeated are the $(v+1)$ fixed ones. So, as all the variables appearing are independent, each sum has the effect of multiplying the corresponding bias by itself 2^{v+1} times. Now we want to show that this bias is also equal to ε. This equivalently means that

$$\frac{1}{2^{n-v-1}} \sum_{x \in a+V} (-1)^{g_{|a+V}(x)} = \frac{1}{2^n} \sum_{x \in \mathbf{F}_2^n} (-1)^{g(x)}.$$

Let $\widehat{f}(a)$ denote the Walsh coefficient of f at point $a \in \mathbf{F}_2^n$, i.e:

$$\widehat{f}(a) = \sum_{x \in \mathbf{F}_2^n} (-1)^{f(x)+ax}.$$

Then, from [2, pages 2005-2006] we have

$$\begin{aligned} 2^{v+1} \sum_{x \in a+V} (-1)^{g_{|a+V}(x)} &= \sum_{u \in W} \widehat{g}(u) \\ &= \sum_{u \in W} \widehat{f}(\alpha + u) \\ &= \widehat{f}(\alpha) = \widehat{g}(0) \end{aligned}$$

since f is v-resilient. So:

$$\Pr\left[\sum_{\tau \in \langle T_{j_0}, \ldots, T_{j_v} \rangle} g(x_1(t+\tau), \ldots, x_n(t+\tau)) = 0\right] = \frac{1}{2^{vr+1}} \times 2^{vr+1} \times 0.5(1+\varepsilon^{2^v}).$$

And then, the bias of the parity check will be ε^{2^v}. It is obvious that if, instead of building the parity-checks by considering each term in the linear approximation separately, we do it by associating several terms, the result will not change. The final bias of the parity-check will also be the bias of the linear approximation raised to the power of the number of terms in the parity-check. $\diamond$