Cryptanalysis of Achterbahn-Version 2

Martin Hell and Thomas Johansson

Dept. of Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{martin,thomas}@it.lth.se

Abstract. Achterbahn is one of the stream cipher proposals in the eSTREAM project. After the first version had been successfully cryptanalyzed, the second version, denoted Achterbahn-Version 2, was proposed. This paper demonstrates an attack on this second version. In the attack, a quadratic approximation of the output function is considered. The attack uses less keystream bits than the upper limit given by the designers and the computational complexity is significantly less than exhaustive key search.

Keywords: Achterbahn, cryptanalysis, stream ciphers, key recovery attack.

1 Introduction

The Achterbahn stream cipher is one of many candidates submitted to the eSTREAM [1] project. It is to be considered as a hardware efficient cipher, using a key size of 80 bits. There have been some successful attacks on Achterbahn [6,5]. As a response to these attacks, the cipher was updated to a more secure version, denoted Achterbahn-Version 2 [4]. Recently, eSTREAM moved into the second phase of the evaluation process and based on the design of Achterbahn-Version 2, the cipher qualified as one of the phase 2 ciphers. After receiving a preliminary version of this paper, the designers tweaked the cipher one more time and the version that is to be considered for the second phase of eSTREAM is the third version, denoted Achterbahn-128/80. This third version will not be considered in this paper.

The design of Achterbahn is based on the idea of a nonlinear combiner, but using nonlinear feedback shift registers instead of registers with linear feedback. When Achterbahn was tweaked, the designers focused on improving the cipher such that approximations of the output function was not a threat. In this paper, we show that the tweak was not enough, it is still possible to attack the cipher using approximations of the output function. This is the first attack on Achterbahn-Version 2.

The paper is outlined as follows. Section 2 will discuss some background theory. Section 3 gives a description of the Achterbahn stream cipher. In Section 4 we give the previous results on Achterbahn that are important to our analysis, which is then given in Section 5. Section 6 will conclude the paper.

E. Biham and A.M. Youssef (Eds.): SAC 2006, LNCS 4356, pp. 45–55, 2007.

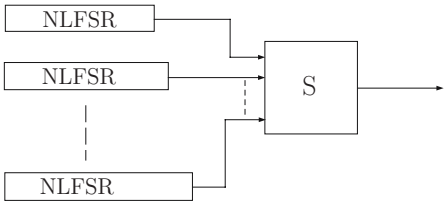

Fig. 1. Overview of the Achterbahn design idea

2 Preliminaries

In this paper we will repeatedly refer to the bias of an approximation. The bias ϵ of an approximation A of a Boolean function P is usually defined in one of two ways.

1. $\Pr(P = A) = 1/2 + \epsilon$. In this case, when n independent bits are xored the bias of the sum is given by $2^{n-1}\epsilon^n$.
2. $\Pr(P = A) = 1/2(1+\epsilon)$. In this case, when n independent bits are xored the bias of the sum is given by ϵ^n. This bias is also commonly referred to as the imbalance.

The bias in the first case will always be half of the bias in the second case. Nevertheless, it is common to approximate the number of keystream bits needed in a distinguisher as

$$\#\text{ samples needed} = \frac{1}{\epsilon^2} \tag{1}$$

regardless which definition of the bias that has been used. The error probability of the distinguisher decreases exponentially with a constant factor multiplied with the number of samples given in (1). Following the notation used in all previous papers on Achterbahn, we will adopt the second case in this paper. Thus, $\epsilon = 2\Pr(P = A) - 1$. Obviously, the sign of ϵ is irrelevant in the theoretical analysis. In the following, when P equals A with probability α, we will write this as $P \stackrel{\alpha}{=} A$.

3 Description of Achterbahn

The Achterbahn stream cipher was first proposed in [2] and later tweaked in [4]. This section will describe both versions of Achterbahn.

Achterbahn supports a key of size 80 bits. The size of the IV is variable and all multiples of 8 between 0 and 64 bits are supported. The cipher consists of a set of nonlinear feedback shift registers and an output function, see Fig. 1. All registers are primitive, which in this context means that the period of register R_i is $2^{N_i} - 1$, where N_i is the length of register R_i. We denote this period by T_i. Hence,

$$T_i = 2^{N_i} - 1, \quad \forall i.$$

The output function is a Boolean function that takes one input bit from each shift register and outputs a keystream bit. The input bit to the Boolean function from register R_i at time t will be denoted $x_i(t)$ and if the time instance t is fixed the simplified notation x_i will sometimes be used.

Achterbahn comes in two variants, denoted reduced Achterbahn and full Achterbahn. In reduced Achterbahn the input bit to the Boolean function from shift register R_i is simply the output bit of R_i. In full Achterbahn the bit used in the Boolean function is a key dependent linear combination of a few bits in R_i. Achterbahn-Version 1 uses 8 shift registers. Their size ranges from 22 to 31 bits. The keystream bit, denoted z, is produced by the Boolean function

$$R(x_1, \ldots, x_8) = x_1 + x_2 + x_3 + x_4 + x_5x_7 + x_6x_7 + x_6x_8 + x_5x_6x_7 + x_6x_7x_8.$$

Achterbahn-Version 2 uses two extra shift registers, hence, it consists of 10 nonlinear feedback shift registers of size ranging from 19 to 32 bits. Their sizes are $N = 19, 22, 23, 25, 26, 27, 28, 29, 31$ and 32. The Boolean output function in Achterbahn-Version 2 is much larger than the function used in Version 1. It is defined as

$$\begin{aligned} S(x_1, \ldots, x_{10}) &= x_1 + x_2 + x_3 + x_9 + G(x_4, x_5, x_6, x_7, x_{10}) \\ &\quad +(x_8 + x_9)(G(x_4, x_5, x_6, x_7, x_{10}) + H(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{10})), \end{aligned}$$

where

$$\begin{aligned} G(x_4, x_5, x_6, x_7, x_{10}) &= x_4(x_5 \vee x_{10}) + x_5(x_6 \vee x_7) + x_6(x_4 \vee x_{10}) \\ &\quad +x_7(x_4 \vee x_6) + x_{10}(x_5 \vee x_7) \end{aligned}$$

and

$$\begin{aligned} H(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{10}) &= x_2 + x_5 + x_7 + x_{10} + (x_3 + x_4)\overline{x}_6 \\ &\quad +(x_1 + x_2)(x_3\overline{x}_6 + x_6(x_4 + x_5)). \end{aligned}$$

The function S has resiliency 5 and nonlinearity 448. This means that any biased linear or nonlinear approximation has to consider at least 6 variables.

3.1 Initialization

The initialization of Achterbahn is simple. It is divided into 5 or 6 steps depending on if the reduced or the full variant is used. These steps are the following.

1. All registers R_i are loaded in parallel with the first N_i bits of the key.
2. The registers are updated and the remaining key bits are xored with the input to the registers.
3. The registers are updated and the IV bits are xored with the input to the registers.
4. The least significant bit of each NLFSR is set to 1. This prevents the NLFSRs to be initialized with all zeros.

5. Warm up phase. The registers are clocked such that the total number of clocks for each register in the initialization phase is $112 + |IV|$, where $|IV|$ is the chosen length of the IV.
6. A key and IV dependent vector is prepared, defining which of the positions in the registers that are to be xored to form the input to the Boolean combining function. (This step is only used in the full variant of Achterbahn.)

Because of step 4 above, the entropy of the content of register R_i at the beginning of step 5 is only $N_i - 1$. No secrets (key bits) are used in phase 5 and thus, exhaustively searching register R_i requires 2^{N_i-1} tries. Note that in previous papers on Achterbahn the entropy loss in step 4 has not been taken into account and when we discuss the previous analysis the factor 2^{N_i} will be used as was done in the original papers.

4 Previous Analysis of Achterbahn

There are several papers analyzing the Achterbahn stream cipher. In this section we take a closer look at them and give the results that are relevant to the new attack given in Section 5.

4.1 Analysis of Achterbahn-Version 1

Achterbahn-Version 1 was first cryptanalyzed in [5], taking advantage of weaknesses found in the Boolean output function. The designers answered by giving two alternative combining functions R' and R'' in [3]. In [6], which is an extended and published version of [5], the authors show that the cipher is weak even if the new combining functions are used. An important observation in [5] is the following. Assume that $x_5 = x_6 = 0$ in $R(x_1, \ldots, x_8)$. Then $R(x_1, \ldots, x_8)$ is a purely linear function. The linear complexity of the resulting function $l(t) = x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t)$ is then bounded by the sum of the linear complexities of the registers R_1, R_2, R_3 and R_4, which is approximately 2^{26}. Hence assuming that $x_5 = x_6 = 0$, there are parity checks involving at most 2^{26} consecutive bits. This parity check equation could be found by noting that $l(t) \oplus l(t + T_i)$ does not depend on the variable x_i. Doing this for $i = 1, 2, 3$ and 4, a parity check equation involving 16 terms within a time interval of $2^{26.75}$ keystream bits can be found. By knowing for which initial states of R_5 and R_6 these 16 terms will be zero, i.e., when the parity check was valid with probability 1, the key could be recovered.

The method of finding a parity check was nicely refined and generalized in [4]. They note that the sequence generated by R_i has characteristic polynomial $x^{T_i} - 1$. Furthermore,

$$g(x) = (x^{T_1} - 1)(x^{T_2} - 1)(x^{T_3} - 1)(x^{T_4} - 1)$$

is a characteristic polynomial of $l(t)$. Even if all variables do not appear linearly in the ANF of a Boolean function, a sparse parity check can easily be found. For

instance, the sequence produced by the function $F(t)=x_1(t)x_2(t)\oplus x_1(t)x_2(t)x_3(t)$ has characteristic polynomial

$$g(x) = (x^{T_1T_2} - 1)(x^{T_1T_2T_3} - 1),$$

giving a parity check equation involving only 4 terms.

In [6], the authors also demonstrated that it is possible to break Achterbahn by considering biased linear approximations of the output function. The approximation

$$z(t) \stackrel{\alpha}{=} x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t) \oplus x_6(t)$$

holds with probability $\alpha = 0.75$, i.e., it has a bias $\epsilon = 0.5$. Since there are 32 terms in the corresponding parity check equation, the total bias is 2^{-32} and a distinguishing attack using 2^{64} bits exists. Furthermore, they note that by guessing the state of register R_1, the parity check will only involve 16 terms and the distinguisher will only need 2^{32} bits. Additionally, the computational complexity will increase by a factor of 2^{23}. Now the attack is a key recovery attack with computational complexity 2^{55} using 2^{32} bits of keystream. This is the best known attack on reduced Achterbahn. The same attack is possible on the full version, but the computational complexity is then 2^{61} instead.

4.2 Analysis of Achterbahn-Version 2

In [4], the designers of Achterbahn demonstrate that the attacks mentioned above will not work when applied to Version 2. This is mostly due to the fact that the combining function $S(x_1, \ldots, x_{10})$ is 5-resilient, thus any biased linear approximation has at least 6 terms and the corresponding parity check will have 64 terms. By guessing the state of the first two registers, the number of terms in the parity check will be 16, but even then the computational complexity and the keystream needed will be far above exhaustive key search.

Further, the designers also considered quadratic and cubic approximations of $S(x_1, \ldots, x_{10})$. In this section we give a description of the cubic case since the result of this analysis gives a very important prerequisite for Achterbahn-Version 2. Our attack will use a quadratic approximation. The cubic approximation that is considered to be most threatening is given by

$$C(x_1, \ldots, x_{10}) = x_4 + x_6x_9 + x_1x_2x_3.$$

This approximation will agree with $S(x_1, \ldots, x_{10})$ with probability

$$\frac{63}{128} = \frac{1}{2}\left(1 - \frac{1}{64}\right),$$

implying that $\epsilon = 2^{-6}$. We can guess the content of register R_4 with $N_4 = 25$. The characteristic polynomial of the sequence generated by the two nonlinear terms is

$$g(x) = (x^{T_6T_9} - 1)(x^{T_1T_2T_3} - 1).$$

This will give a parity check equation with 4 terms and bias $\epsilon^4 = (2^{-6})^4 = 2^{-24}$ assuming that the variables are independent. The distance between the first and the last bit in the parity check is $T_1T_2T_3 + T_6T_9 \approx 2^{64}$ bits. The time complexity of this attack is $2^{48}2^{N_4} = 2^{73}$. This is less than exhaustive key search and consequently *the designers restrict the frame length of Achterbahn-Version 2 to* 2^{63} *bits.*

Note that the previously described attack is *impossible* when the keystream length is limited to 2^{63} since then we cannot create any biased samples at all. In most distinguishing attacks on stream ciphers, you can usually create biased samples even if the keystream length is limited, it is just the case that you cannot collect enough samples to detect the bias for sure.

5 Cryptanalysis of Achterbahn-Version 2

Since there is an attack requiring approximately 2^{64} keystream bits, and the frame length is restricted to 2^{63} bits, a new attack has to require less than 2^{63} keystream bits in order to be regarded as successful. A danger of restricting the amount of keystream to some number due to the existence of an attack is that someone might find an improvement of the attack. This would render the cipher insecure. In this section we demonstrate exactly that. A straightforward approach of our attack is given first and in Section 5.4 an improved variant is given, reducing the computational complexity significantly.

5.1 Attack on the Reduced Variant

The complexities given in this subsection will be based on the reduced variant of the cipher, i.e., the input to the Boolean combining function will be the rightmost bit in each NLFSR.

The attack will consider the quadratic approximation

$$Q(x_1, \ldots, x_{10}) = x_1 + x_2 + x_3x_8 + x_4x_6.$$

This approximation will agree with S with probability

$$\frac{33}{64} = \frac{1}{2}\left(1 + \frac{1}{32}\right),$$

implying that $\epsilon = 2^{-5}$. Denote the sequence produced by Q by $z'(t)$. Using this approximation, we can use the characteristic polynomial

$$g(x) = (x^{T_3T_8} - 1)(x^{T_4T_6} - 1).$$

which gives a parity check equation involving 4 terms. Looking at the sequence generated by Achterbahn-Version 2, we know that if we consider the sequence

$$d(t) = z(t) \oplus z(t + T_3T_8) \oplus z(t + T_4T_6) \oplus z(t + T_3T_8 + T_4T_6) \qquad (2)$$

then $d(t)$ will not depend on the quadratic terms in $Q(x_1, \ldots, x_{10})$. Under the assumption that the 4 keystream bits in (2) are independent, the bias of (2) is $\epsilon^4 = 2^{-20}$. If these keystream bits are not independent the bias will be larger, so we are considering a worst case scenario. The dependency of the keystream bits will be further examined in a separate paper. Hence, with probability $\alpha = 1/2(1 + 2^{-20})$, the sequence $d(t)$ will equal

$$\begin{aligned} d(t) &\stackrel{\alpha}{=} z'(t) \oplus z'(t + T_3T_8) \oplus z'(t + T_4T_6) \oplus z'(t + T_3T_8 + T_4T_6) \\ &= x_1(t) \oplus x_2(t) \oplus x_1(t + T_3T_8) \oplus x_2(t + T_3T_8) \oplus x_1(t + T_4T_6) \\ &\quad \oplus x_2(t + T_4T_6) \oplus x_1(t + T_3T_8 + T_4T_6) \oplus x_2(t + T_3T_8 + T_4T_6). \end{aligned}$$

At this point we can guess the initial state of the registers R_1 and R_2 as suggested in [4], where they used another approximation. The length of these two registers is $N_1 = 19$ and $N_2 = 22$ respectively. The amount of keystream needed to distinguish the output sequence from random is 2^{40} so the computational complexity would be $2^{18+21+40} = 2^{79}$, which is the same as the expected complexity in an exhaustive search. The distance between the bits in the sum is $T_3T_8 + T_4T_6 \approx 2^{53}$ so this would be the amount of keystream needed.

Instead of taking this approach we note that the length of register R_1 is $N_1 = 19$, hence,

$$x_1(t) = x_1(t + T_1) = x_1(t + 2^{19} - 1).$$

Thus, for all keystream bits, distance $T_1 = 2^{19} - 1$ bits apart, x_1 will always contribute with the same value to the output function. Consequently, instead of taking the sequence $d(t)$ for $t = 0 \ldots 2^{40} - 1$ we can instead take the sequence $d'(t) = d(t(2^{19} - 1))$ for $t = 0 \ldots 2^{40} - 1$, i.e., jump forward T_1 steps for each sample. Hence,

$$\begin{aligned} d'(t) &= z(tT_1) \oplus z(tT_1 + T_3T_8) \oplus z(tT_1 + T_4T_6) \oplus z(tT_1 + T_3T_8 + T_4T_6) \\ &\stackrel{\alpha}{=} x_2(tT_1) \oplus x_2(tT_1 + T_3T_8) \oplus x_2(tT_1 + T_4T_6) \\ &\quad \oplus x_2(tT_1 + T_3T_8 + T_4T_6) \oplus \gamma(t), \end{aligned}$$

where

$$\gamma(t) = x_1(tT_1) \oplus x_1(tT_1 + T_3T_8) \oplus x_1(tT_1 + T_4T_6) \oplus x_1(tT_1 + T_3T_8 + T_4T_6)$$

is a constant. If the value of $\gamma(t) = 0$, then the probability $\alpha = 1/2(1 + 2^{-20})$. If the value $\gamma(t) = 1$ then $\alpha = 1/2(1 - 2^{-20})$. In any case, the number of samples needed to detect the bias is 2^{40}. The total amount of keystream required in this approach will increase with a factor of 2^{T_1}, i.e.,

$$\#\text{ keystream bits needed} = 2^{53} + 2^{19}2^{40} = 2^{59.02}.$$

This value is less than the maximum length of a frame. The computational complexity will be $2^{40}2^{21} = 2^{61}$, since now we only need to guess R_2 with $N_1 = 22$, requiring 2^{21} guesses. This will give us the initial state of register R_2 after step 4 in the initialization process.

5.2 Recovering the Key

When one state is known, finding the actual key used can be done using a meet-in-the-middle attack and a time/memory tradeoff approach. First, R_2 is clocked backwards until we reach the state that ended the introduction of the key. We denote this state Δ. Then the key is divided into two parts, k_1 and k_2 bits each and $k_2 = 80 - k_1$. We guess the first k_1 bits of the key and clock the register until after the introduction of this part. All possible 2^{k_1} states are saved in a table. Then the last k_2 bits of the key are guessed, and the state Δ is clocked backwards k_2 times reversing the introduction of the key. Any intersection of the two states reached, gives a possible key candidate. Since R_2 has size $N_2 = 22$ we expect the number of intersections to be $2^{80}2^{-22} = 2^{58}$, i.e., less than the complexity of finding the state of R_2. The step of finding the intersections will require memory 2^{k_1} and time $2^{k_1} + 2^{k_2}$. Appropriate values can be e.g., $k_1 = 30$ and $k_2 = 50$. The total computational complexity of the attack would then be $2^{61} + 2^{58} = 2^{61.17}$.

5.3 Attack on the Full Variant

The full Achterbahn-Version 2 uses a key dependent linear combination of the shift register bits as input to the Boolean combining function. To the best of our knowledge, there is no specification of Version 2 that explicitly gives the amount of bits in each register that is used in the linear combination. However, in the analysis given in [4, Sect. 3.3], the designers imply that for the registers R_1, R_2 and R_3, 3 register bits are used in each. In our attack we are only interested in the amount of bits used from R_1 and R_2 so this information is sufficient. The consequence is that, when attacking the full variant, an extra factor of 2^3 has to be multiplied when finding the state register R_2.

5.4 Improving the Computational Complexity

In the previous subsection, a simple approach for the attack was given resulting in computational complexity $2^{61.17}$ and $2^{59.02}$ keystream bits for the reduced variant. The computational complexity of the attack can be significantly reduced using the fact that the period of the registers are very short. In this subsection we go through each step in the attack and give the computational complexity in each step. It is assumed that the cryptanalyst observes a sequence of $2^{59.02}$ keystream bits.

- **Produce d'(t).** From the observed keystream sequence, the sequence $d'(t)$ of length 2^{40} is computed. This will have computational complexity 2^{42} since each bit in $d'(t)$ is the sum of 4 bits in the keystream. The amount of memory required to save this sequence is 2^{40} bits, i.e., 2^{37} bytes.
- **Build a table from d'(t).** The straightforward approach when $d'(t)$ is available is to compare the bits in $d'(t)$ with the bits produced by

$$x_2(tT_1) \oplus x_2(tT_1 + T_3T_8) \oplus x_2(tT_1 + T_4T_6) \oplus x_2(tT_1 + T_3T_8 + T_4T_6),$$

Position in d'(t)	# Zeros	# Ones
$0 + iT_2$		
$1 + iT_2$		
$2 + iT_2$		
$\vdots$		
$T_2 - 1 + iT_2$		

Fig. 2. Store the number of ones and zeros in a table

$0 \leq t < 2^{40}$, for all possible initial states of R_2. Indeed, this would require a computational complexity of 2^{61} as given in Section 5.1. To speed up the exhaustive search of register R_2 we note that

$$x_2(t) = x_2(t \bmod T_2).$$

This means that all $d'(t + iT_2)$, $\forall i$, will be compared with the same value. In order to take advantage of this, we suggest to build a table with the bits in $d'(t)$. We go through $d'(t)$ and count the number of zeros and ones in $d'(0+iT_2)$, $d'(1+iT_2)$, $d'(2+iT_2)$, etc. These numbers are stored in a table, see Fig 2. This step will have computational complexity 2^{40} and requires about 2^{22} words of memory.

- **Recover $\mathbf{R_2}$.** When the state of register R_2 is to be recovered the table in Fig 2 is used. For each possible initial state of R_2 the sum of the four bits

$$x_2(tT_1) \oplus x_2(tT_1 + T_3T_8) \oplus x_2(tT_1 + T_4T_6) \oplus x_2(tT_1 + T_3T_8 + T_4T_6), \quad (3)$$

$0 \leq t < T_2$, is found. Note that all positions are taken modulo T_2. The number of occurrences in the precomputed table is then added together where the column used is the value of the sum (3). The bias will be detected for the correct initial state of R_2. Because of the precomputed table, this step will now only have computational complexity $T_2 2^{21} = 2^{43}$ instead of 2^{61}. For full Achterbahn-Version 2, this complexity will be increased to 2^{46}.
- **Recover the key.** To recover the key, the meet-in-the-middle approach given in section 5.1 can be used. In that case 2^{58} keys will be candidates as correct key which is much higher than the computational complexity to find the state of R_2. To reduce this number, we first find the state of R_1. This is easy now since R_2 is known. A similar table can be produced from the sequence $d(t)$ and the initial state of R_1 is found with complexity $T_1 2^{18} = 2^{37}$. When both R_1 and R_2 are known the expected number of key candidates decreases to $2^{80-22-19} = 2^{39}$. All these key candidates can be tested without this step being a computational bottleneck.

It is interesting to note that once we have received $2^{59.02}$ keystream bits, the maximum computational step is only 2^{43} and 2^{46} for the reduced and full variants respectively. This is due to the fact that we only use a fraction of the received keystream and that we can take advantage of the fact that the registers have

short period. It is debatable if we can claim that the computational complexity of the attack much lower than the required amount of keystream since producing and receiving the keystream will require at least $2^{59.02}$ clockings of the cipher. On the other hand, if we are given a randomly accessible memory with $2^{59.02}$ keystream bits, then the key is found with much fewer computational steps since not all bits on the memory will be accessed. This could be a possible scenario in the case of future DVD formats with extremely high resolution, though the access time would probably be a bottleneck in that case. Anyway, we will be conservative in our claims and consider the computational complexity to be the same as the amount of keystream needed, i.e., $2^{59.02}$. Consequently, the attack on full and reduced Achterbahn-Version 2 will have the same complexity.

5.5 On the Problem of Finding the Initial State of R_2

When we try to recover the initial state of R_2 we assume that it is enough to consider $1/\epsilon^2$ bits in order to detect the bias and thus identifying the correct initial state. In total 2^{21} different candidate states are tested and while it is true that the bias will be detected for the correct initial state it is very likely that several other states will report a detected bias. This will happen because we can assume that the sum of the positions in the precomputed table will be distributed according to a normal distribution with expected value 2^{39}. In our case, this is not really a problem. The correct initial state can still be found using only $2^{59.02}$ bits and 2^{40} samples to detect the bias. For all states that report a bias, we can do the same thing again, shifting the sequence $d'(t)$ one step. This will give 2^{40} new samples and we can check which of the remaining states will report a detected bias. The total amount of keystream needed in our attack will be increased by only 1 bit and the extra amount of computations will be about 2^{40}, the complexity of building a new table. Since our claimed complexity is $2^{59.02}$ we can do this procedure many times if necessary without exceeding the claimed computational complexity.

6 Conclusion

Achterbahn-Version 2 was designed to resist approximations of the output functions, linear approximations as well as quadratic and cubic approximations. Due to a cubic approximation, the amount of keystream that is allowed to be generated is limited to 2^{63}. In this paper we have shown that it is still possible to find an attack using a quadratic approximation. The amount of keystream needed in the attack is below the given limit. Instead of guessing both R_1 and R_2, as was done in a previous analysis, we guess only one of the registers. The attack on Achterbahn-Version 2 has computational complexity slightly more than 2^{59} and needs slightly more than 2^{59} keystream bits. After receiving the keystream bits the computational step is very fast due to the fact that we do not use all keystream bits and that the periods of the registers are very short. The complexities will be the same for both the full and the reduced variants of the cipher.

Acknowledgement

The work described in this paper has been 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 guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

1. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at http://www.ecrypt.eu.org/stream/
2. Gammel, B.M., Göttfert, R., Kniffler, O.: The Achterbahn stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/002 (2005), http://www.ecrypt.eu.org/stream
3. Gammel, B.M., Göttfert, R., Kniffler, O.: Improved Boolean combining functions for Achterbahn. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/072 (2005), http://www.ecrypt.eu.org/stream
4. Gammel, B.M., Göttfert, R., Kniffler, O.: Status of Achterbahn and tweaks. The State of the Art of Stream Ciphers. In: Workshop Record, SASC 2006, Leuven, Belgium (February 2006)
5. Johansson, T., Meier, W., Müller, F.: Cryptanalysis of Achterbahn. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/064 (2005), http://www.ecrypt.eu.org/stream
6. Johansson, T., Meier, W., Müller, F.: Cryptanalysis of Achterbahn. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, Springer, Heidelberg (2006)