Cryptanalysis of Sober-t32

Steve Babbage[1], Christophe De Cannière[2] ⋆, Joseph Lano[2] ⋆⋆, Bart Preneel[2], and Joos Vandewalle[2]

[1] Vodafone Group Research & Development,
The Courtyard,
2-4 London Road,
Newbury, Berkshire RG14 1JX, U.K.
steve.babbage@vodafone.com

[2] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC⋆⋆⋆,
Kasteelpark Arenberg 10,
B3001 Heverlee, Belgium
{cdecanni,jlano,preneel,vdwalle}@esat.kuleuven.ac.be

Abstract. Sober-t32 is a candidate stream cipher in the NESSIE competition. Some new attacks are presented in this paper. A Guess and Determine attack is mounted against Sober-t32 without the decimation of the key stream by the so-called stuttering phase. Also, two distinguishing attacks are mounted against full Sober-t32. These attacks are not practically feasible, but they are theoretically more efficient than exhaustive key search.
Keywords: NESSIE, Cryptanalysis, Security Evaluation, Sober-t32, Guess and Determine Attack, Distinguishing Attack.

1 Introduction

The European NESSIE [2] competition evaluates a variety of cryptographic primitives (both asymmetric and symmetric) for standardization. Sober-t32, a software-oriented synchronous stream cipher designed by G. Rose and P. Hawkes [1], is one of the candidates. The cipher uses a Linear Feedback Shift Register (LFSR), a Non-Linear Function (NLF) and a so-called *stuttering* unit for the generation of the pseudo-random key stream.

The NESSIE competition demands that a stream cipher offers full security, i.e., that there is no known attack faster than exhaustive key search. In cryptanalysis, one considers that the pseudo-random key stream is known, and the aim is to recover the key (a so-called key-recovery attack). Another possible attack is a distinguishing attack, in which one tries to distinguish the key stream of the stream cipher from a truly random sequence.

⋆ F.W.O. Research Assistant, sponsored by the Fund for Scientific Research - Flanders (Belgium)

⋆⋆ Research financed by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)

⋆⋆⋆ Research Council KUL: Concerted Research Action (GOA) Mefisto 666

In this paper, we will present new attacks on Sober-t32. The first attack is a Guess and Determine (GD) attack against unstuttered Sober-t32. This attack exploits a probabilistic factor in the design and appears to be faster than exhaustive search. This is not expected by the designers, as they state in [1]: *Our analysis indicates that the combination of the LFSR and NLF appears to be sufficient to resist GD-attacks.* The second type of attacks are distinguishing attacks. In [5], Ekdahl and Johansson mount distinguishing attacks on Sober-t16 – a cipher very similar to Sober-t32 – and on Sober-t32 without the stuttering unit. In this paper, the attacks from [5] will be improved and distinguishing attacks on full Sober-t32 will be presented.

The outline of this paper is as follows: In Sect. 2, a description of Sober-t32 is given. In Sect. 3, the GD attack is elaborated, based on a probabilistic factor in the design of Sober-t32. Finally, Sect. 4 presents two distinguishing attacks on Sober-t32.

2 Description of the Sober-t32 Stream Cipher

Sober-t32 is a word-oriented synchronous stream cipher. It uses 32-bit words and has a secret key of 256 bits. Sober-t16 is a very similar stream cipher that uses 16-bit words and has a 128-bit key.

In a synchronous stream cipher such as Sober-t32, the key stream is generated independently from the plaintext. The sender encrypts the plaintext by performing an XOR (exclusive or, $\oplus$) operation between plaintext and key stream. The recipient can, if he knows the secret key, reconstruct the key stream and recover the plaintext by performing an XOR operation between the ciphertext and the key stream.

Sober-t32 is based on 32-bit operations within the Galois Field $GF(2^{32})$. Every word $a = (a_{31}, a_{30} \dots a_1, a_0)$ is represented by a polynomial of degree less than 32:

$$A = a_{31}x^{31} + \dots + a_0 x^0 \,. \tag{1}$$

When adding two words in $GF(2^{32})$, the polynomials are added and their coefficients are reduced modulo 2. This is the same as a bitwise XOR. For a multiplication, the polynomials are multiplied, the coefficients reduced modulo 2, and the resulting polynomial is reduced modulo a polynomial of degree 32. For Sober-t32 the polynomial is:

$$x^{32} + (x^{24} + x^{16} + x^8 + 1)(x^6 + x^5 + x^2 + 1) \,. \tag{2}$$

Sober-t32 consists of three main building blocks. First there is a *Linear Feedback Shift Register (LFSR)*, which uses a recursion formula to produce a state sequence s_n. Next a *Non-Linear Function (NLF)* combines these words in a non-linear way to produce the NLF-stream v_n. Finally, the so-called *stuttering* produces the key stream z_j by decimating the NLF-stream in an irregular fashion. All three parts are explained in detail below. An overview of the general structure of Sober-t32 is shown in Fig. 1.

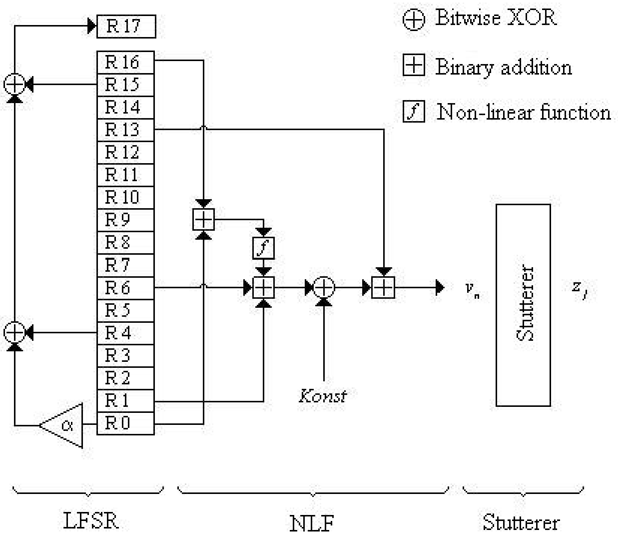

Fig. 1. Overall structure of SOBER-t32

2.1 The Linear Feedback Shift Register (LFSR)

The LFSR is a shift register of length 17, where every register contains one word. The internal memory is thus 544 bits. The state of the LFSR at a certain time t is respresented by the vector

$$\overrightarrow{S_t} = (s_t, s_{t+1}, s_{t+2}, \dots s_{t+16}) = (r_0, r_1, r_2, \dots r_{16}) \,. \tag{3}$$

The next state of the LFSR is calculated by shifting the previous state one step, and calculating a new word s_{t+17} as a linear combination of the words in the LFSR. The word s_{t+17} is calculated as follows:

$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \alpha \cdot s_t \,, \tag{4}$$

with $\alpha = \texttt{C2DB2AA3}_{\texttt{x}}$.

2.2 The Non-Linear Function (NLF)

At any time t, the NLF takes five words from the LFSR state and calculates one output, called v_t. This output can be written as:

$$v_t = ((f(s_t \boxplus s_{t+16}) \boxplus s_{t+1} \boxplus s_{t+6}) \oplus K) \boxplus s_{t+13} \,. \tag{5}$$

In this equation, K is a word that is determined during the initialization of the LFSR, $\boxplus$ denotes addition modulo 2^{32} and f is a non-linear function. The structure of the function f is shown in Fig. 2. First, the word is partitioned into the Most Significant Byte (MSB) and the three remaining bytes. The MSB is used as an input to a substitution box (S-box), which outputs 32 bits. Of these, the MSB becomes the MSB of the output, and the three remaining bytes are XORed with the three remaining bytes of the input word to form the rest of the output word.

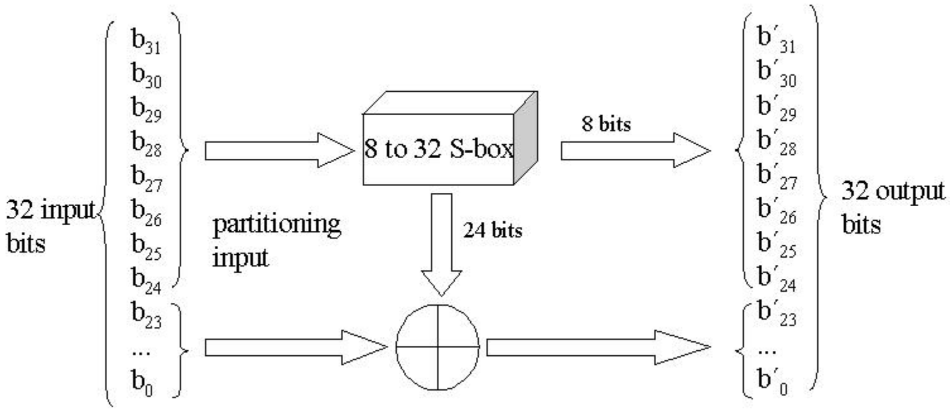

Fig. 2. Structure of the function f

It is important to notice that the S-box only uses the MSB as an input. This implies that most of the non-linearity is caused by the MSB. One can see that any differential in the less significant bits goes straight through the f-function, and that the MSB of the output is solely determined by the MSB of the input (and vice-versa). The f-function can be written as:

$$f(a) = SBOX(a_{MSB}) \oplus (0 \| a_R) . \tag{6}$$

In this equation, a_R respresents the three remaining bytes of a 32-bit word a, thus without the MSB.

2.3 The Stuttering

The output of the NLF is used as the input for the stuttering phase. The stuttering decimates the stream in an irregular fashion. The first output of the NLF is taken as the first *Stutter Control Word (SCW)*. This SCW is divided into pairs of bits, called *dibits*. The value of these dibits determines what happens with the following words, as explained in Table 1.

The constant C is $\texttt{6996C53A}_\texttt{x}$, and C' is the bitwise complement of C. When all dibits have been used, the next word from the NLF-stream is taken as the next SCW. In this way, the stuttering allows only about 48% of the NLF-stream words to go to the key stream.

Table 1. The possible actions of the stuttering unit as a function of the dibit

dibit	action
00	The next word is excluded from the key stream.
01	The next word is XORed with C, and goes to the key stream, the word after that is excluded from the key stream.
10	The next word is excluded from the key stream, the word after that goes to the key stream.
11	The next word is XORed with C', and goes to the key stream.

3 A Guess and Determine Atttack on Unstuttered Sober-t32

In a standard Guess and Determine attack, some words of the LFSR are guessed and the remaining words are determined by exploiting the LFSR and NLF equations. In [3], Bleichenbacher *et al.* describe a GD attack on Sober-II which also applies to Sober-t32 and has a complexity of 2^{320}. De Cannière [4] improves this attack to 2^{304}.

In this section, the weakness of the NLF, as explained in Sect. 2.2, will be exploited to elaborate a better GD attack. First a simplified attack will be presented, where the carry bits are not taken into account. Next the real attack will be presented. Finally, the attack will be improved in different ways.

3.1 Attack without Carry Bits

First, we will rewrite the equation of the NLF (5) by separating the MSB and the three other bytes and by using (6). This yields the following equation:

$$v_t = \begin{array}{l}(((SBOX(s_{t,MSB} \boxplus s_{t+16,MSB} \boxplus o_1) \oplus (0\|s_{t,R} \boxplus s_{t+16,R})) \boxplus s_{t+1} \boxplus s_{t+6}) \oplus \\ K) \boxplus s_{t+13} \, .\end{array} \tag{7}$$

In this equation, o_1 represents the carry bit towards the MSB. Under the assumption that the MSB of K is zero, this equation can be split up in two separate parts:

$$\begin{cases} v_{t,MSB} = \begin{array}{l}((SBOX_1(s_{t,MSB} \boxplus s_{t+16,MSB} \boxplus o_1)) \boxplus s_{t+1,MSB} \boxplus s_{t+6,MSB}) \\ \boxplus s_{t+13,MSB} \boxplus o_2\end{array} \\ v_{t,R} = \begin{array}{l}(((SBOX_2(s_{t,MSB} \boxplus s_{t+16,MSB} \boxplus o_1) \oplus (s_{t,R} \boxplus s_{t+16,R})) \\ \boxplus s_{t+1,R} \boxplus s_{t+6,R}) \oplus K_R) \boxplus s_{t+13,R} \, .\end{array} \end{cases} \tag{8}$$

In this equation, $SBOX_1$ represents the MSB of the output of the S-box, and $SBOX_2$ the three remaining bytes of the S-box output. o_2 represents the carry bits from the additions.

Especially the first equation is interesting. Given the value of the MSB of s_t, s_{t+1}, s_{t+6} and s_{t+13}, the value of the MSB of the key stream v_t and the value of the carry bits o_1 and o_2, it is possible to calculate the MSB of s_{t+16}.

For the moment, the carry bits are not taken into account. Now the attack starts. First, the MSB of the first 16 words of the LFSR $(s_t, s_{t+1}, s_{t+2} \ldots s_{t+15})$ are guessed, a total of 128 bits. Knowing these, together with the key stream v_t, it is possible to calculate the MSB of $s_{t+16}, s_{t+17}, s_{t+18}, s_{t+19}, \ldots$

The LFSR will now be clocked a few times. We get a system of linear equations in bits, where the unknowns are the 24 least significant bits of every word that appears in the LFSR. Meanwhile, at every iteration a number of linear equations is obtained. These equations are the following:

- At every iteration, we get 32 new linear equations from the derivation of the new LFSR word. In fact, the linear recurrence over $GF(2^{32})$ is equivalent to 32 bit-wise LFSR's (see [6]). These LFSR's all have the same linear recurrence, which is given in [7].
- We also get an extra linear equation per iteration in the least significant bit. As the input to the S-box is known, one can easily see that the following equation holds for the least significant bit:

  $$v_t^0 = SBOX^0(s_{t,MSB} \boxplus s_{t+16,MSB} \oplus o_1) \oplus s_t^0 \oplus s_{t+16}^0 \oplus s_{t+1}^0 \oplus s_{t+6}^0 \oplus K^0 \oplus s_{t+13}^0 \,. \tag{9}$$

  In this equation, the superscript ^{0} stands for the least significant bit of the word. This equation is perfectly linear. We also get one such equation before iterating: before the LFSR is clocked, we already get such an equation from the initial state.

After clocking the LFSR k times, $32 \cdot k + k + 1 = 33 \cdot k + 1$ linear equations are obtained. The remaining unknowns in these equations are:

- The value of the 24 least significant bits of $s_t, s_{t+1}, s_{t+2} \ldots s_{t+16}$.
- The value of the least significant bit of K.
- The value of the 24 least significant bits of the new word at each iteration.

After k iterations, the total is $24 \cdot 17 + 1 + 24 \cdot k = 24 \cdot (17 + k) + 1$ unknowns.

In order to obtain a solvable set of equations, the number of equations must be larger than the number of unknowns:

$$33 \cdot k + 1 \geq 24 \cdot (17 + k) + 1 \Longleftrightarrow k \geq 45.33 \,. \tag{10}$$

Remark that the attack recovers the whole state of the cipher, except for the 23 remaining bits of K. However, once the attack is finished, these can be recovered easily from the second equation of (8).

In order to evaluate the complexity of the attack, one should consider the number of bits that has to be guessed. The following bits should be guessed:

- The MSB of $s_t, s_{t+1}, s_{t+2} \ldots s_{t+15}$, a total of 128 bits.
- The MSB of K. In fact, this MSB will always be assumed to be zero. This assumption can be seen to be equivalent to guessing the MSB of K: It is possible to mount the attack on a number of different key streams until we get a key stream in which the MSB of K is zero.

This means guessing 136 bits in total, which implies a complexity of 2^{136}. It should be noted that solving the set of equations does not increase the complexity of the attack: We know in advance the linear equations relating the deduced bits to the initial state bits, so we can precompute the inverse matrix and all we have to do is a matrix multiplication.

Of course, we have not taken into account the carry bits that are unknown to us. In the next section, we will take the carry bits into account and show the complexity of the full attack.

3.2 Taking Account of the Carry Bits

In the previous section, we have assumed that the carry bits are known. In reality we will also have to guess these carry bits. Carry bits are not purely random, the distribution of their value is non-uniform. One can take advantage of this by first trying to guess the more probable values. The number of times we have to guess on average will be well approximated by the entropy, as this equals the amount of information that is present in the carry bits.

The Entropy of the Carry Bits. The entropy of the carry bits o_1 and o_2 can be derived theoretically. This is done in Appendix A. It is shown there that the entropy of o_1 is 1, and that the entropy of o_2 is 1.65.

Complexity of the Full Attack. The full attack requires guessing the following bits:

- The 136 bits that had to be guessed in the attack without carry bits.
- The carry bits, a total of $47 \cdot (1 + 1.65) = 124.55$ bits. (That is, the number of iterations plus one. This extra guess comes from determining s_{t+16}: there is no iteration here, but these carry bits also have to be guessed.)

This means a total of 260.55 bits. The complexity of the attack is thus $2^{260.55}$. This is only a little above the complexity of doing an exhaustive search for the 256-bit key. In the following section, a number of improvements will be presented in order to get the complexity below that of exhaustive key search.

3.3 Further Improvements

Instead of assuming that the 8 most significant bits of K are zero, one could also assume that the nine most significant bits of K are zero. This will lower the entropy of the carry bit o_2. It can be calculated that the new entropy for o_2 is 1.48. Now we can recalculate the total complexity for the attack with the procedure described above. We get a complexity of $2^{253.56}$.

We can also assume that the 10 most significant bits of K are zero. The entropy of o_2 is then 1.43, and the complexity of the attack is $2^{252.21}$.

Another improvement would be to guess the carry bits of several rounds together. The entropy of the carry bit in $(s_0 + s_{16})$ is 1; and the entropy of the carry bit in $(s_{16}+s_{32})$ is also 1. The entropy of the two together is 1.92 however. If we take (s_0+s_{16}), $(s_{16}+s_{32})$ and $(s_{32}+s_{48})$ together, the entropy is 2.83. There is scope for more improvements of this sort.

Another possible improvement would be to use all relations in the NLF, so not only the linear ones. This approach is described in App. B. The complexity of this approach is not well understood and requires further research.

4 Distinguishing Attacks

At FSE 2002, P. Ekdahl and T. Johansson presented distinguishing attacks on full SOBER-t16 and on unstuttered SOBER-t32 [5]. In this section, both attacks will be adapted to obtain two distinguishing attacks on full SOBER-t32.

4.1 Extending the Attack on Unstuttered Sober-t32 to Full Sober-t32

In this section, the attack on unstuttered SOBER-t32, described by P. Ekdahl and T. Johansson in [5], is adapted so that it also works on full SOBER-t32. First an overview of the attack on unstuttered SOBER-t32 will be given. For a complete description we refer to [5]. Then the attack on full SOBER-t32 will be described.

The Attack on Unstuttered Sober-t32. The attack starts by linearizing the equation of the NLF (5):

$$v_t = \Big[s_t \oplus s_{t+1} \oplus s_{t+6} \oplus s_{t+13} \oplus s_{t+16}\Big] \oplus w_t = \Omega_t \oplus w_t \,. \qquad (11)$$

Then it will be argued that the noise w_t, introduced by this approximation, has a biased distribution.

In the next step, a new linear recurrence is obtained by repetitive squaring of the LFSR equation (4):

$$s_{t+\tau_5} \oplus s_{t+\tau_4} \oplus s_{t+\tau_3} \oplus s_{t+\tau_2} \oplus s_{t+\tau_1} \oplus s_t = 0\,, \qquad (12)$$

with $\tau_1 = 11$, $\tau_2 = 13$, $\tau_3 = 4 \cdot 2^{32} - 4$, $\tau_4 = 15 \cdot 2^{32} - 4$ and $\tau_5 = 17 \cdot 2^{32} - 4$. This linear recurrence is valid for each bit position individually.

Then the XOR between two adjacent bits in the stream v_t are considered:

$$v_t[i] \oplus v_t[i-1] = \Omega_t[i] \oplus \Omega_t[i-1] \oplus w_t[i] \oplus w_t[i-1]\,. \qquad (13)$$

The distribution $F[i]$ of $w_t[i] \oplus w_t[i-1]$ is then calculated. Simulation indicates that this distribution is quite biased. The largest bias was found for the XOR of bit 29 and 30. The bias depends on the corresponding bits of K, and for bit 29 and bit 30 it is at least $\epsilon_{30} = 0.0052$.

Now, given the NLF-stream $v_0, v_1, \ldots v_{N-1}$, the linear recurrence (12) can be used to calculate

$$v_{t+\tau_5} \oplus v_{t+\tau_4} \oplus v_{t+\tau_3} \oplus v_{t+\tau_2} \oplus v_{t+\tau_1} \oplus v_t = \begin{array}{l} \Omega_{t+\tau_5} \oplus w_{t+\tau_5} \oplus \Omega_{t+\tau_4} \oplus w_{t+\tau_4} \oplus \\ \Omega_{t+\tau_3} \oplus w_{t+\tau_3} \oplus \Omega_{t+\tau_2} \oplus w_{t+\tau_2} \oplus \\ \Omega_{t+\tau_1} \oplus w_{t+\tau_1} \oplus \Omega_t \oplus w_t \,, \end{array} \tag{14}$$

where the sum of all Ω_j terms is zero because of (12). This equation can be rewritten as:

$$v_{t+\tau_5} \oplus v_{t+\tau_4} \oplus v_{t+\tau_3} \oplus v_{t+\tau_2} \oplus v_{t+\tau_1} \oplus v_t = \bigoplus_{j=0}^{5} w_{t+\tau_j} \,. \tag{15}$$

The left hand side of this equation is noted as V_t, the right hand side as W_t. It is now possible to calculate the following probability:

$$P(V_t[i] \oplus V_t[i-1] = 0) = P(W_t[i] \oplus W_t[i-1] = 0) = \frac{1}{2} + 2^5 \epsilon_i^6 \,. \tag{16}$$

The final correlation probability for the six independent key stream positions can then be obtained for $i = 30$:

$$p_0 = P(V_t[30] \oplus V_t[29] = 0) = \frac{1}{2} + 2^5(0.0052)^6 \approx \frac{1}{2} + 2^{-40.5} \,. \tag{17}$$

In order to distinguish this nonuniform distribution P_0 from a uniform source P_U, the Chernoff information between the two distributions is calculated:

$$C(P_0, P_U) = -\min_{0 \le \lambda \le 1} \log_2 \sum_x P_0^{\lambda}(x) P_U^{1-\lambda}(x) \approx 2^{-81.5} \,. \tag{18}$$

In order to obtain an error probability of $P_e = 2^{-32}$, $N = 2^{86.5}$ samples from the key stream are needed. Each sample spans $\tau_5 = 17 \cdot 2^{32} \approx 2^{36}$ bits, so in total $N + \tau_5 \le 2^{87}$ words from the NLF-stream are needed to distinguish unstuttered SOBER-t32 from a uniform source.

The Attack on Full Sober-t32. This distinguishing attack requires the words v_t, v_{11}, v_{13}, $v_{4\cdot 2^{32}-4}$, $v_{15\cdot 2^{32}-4}$ and $v_{17\cdot 2^{32}-4}$ from the NLF-stream. Our aim is to find these words in the key stream z_j, i.e., after the stuttering. One might think that the probability of guessing the right positions for the large values $4 \cdot 2^{32} - 4$, $15 \cdot 2^{32} - 4$ and $17 \cdot 2^{32} - 4$ will be so small that the distinguishing attack will no longer be successful. In this section we will show that this is not the case.

First of all, an expression is derived for the probability that a particular word will be at its most probable position in the key stream. A key stream word z_i is taken, and this word comes from the word v_t in the NLF-stream. Then the probability is calculated that the following words appear at their most probable position in the key stream.

The most probable position of v_{t+11} in the key stream is z_{i+6}. Simulations indicate that the probability that this is a correct guess is 21.7%. The most probable position of v_{t+13} in the key stream is z_{i+7}. Simulations indicate that the probability that this is a correct guess is 19.8%.

For the remaining three words, the situation is more complex. The probabilities will be calculated through a theoretical deduction. For the n-th word that goes to the stuttering unit, it can be expected that $\lfloor \frac{n}{25} \rfloor$ stutter control words (SCW) have been used before. Of all the remaining (non-SCW) words, 50% are expected to go to the key stream. The most probable position in the key stream of the word v_n is thus:

$$E[position(v_n)] = \frac{n - \lfloor \frac{n}{25} \rfloor}{2} \, . \tag{19}$$

In order to calculate the probability that the word v_n will be at this most probable position, we will first calculate the probability that the n-th SCW appears at its most probable position in the NLF-stream. This probability is easier to calculate theoretically, and it is easy to see that the asymptotical behaviour of both values will be the same.

Two dibits, 00 and 11, determine what is going to happen with the next word. The two other dibits, 01 and 10, determine the stuttering of the two following words. As every dibit appears with the same probability, it is expected that a dibit uses 1.5 words of the NLF-stream on average.

A SCW gives 16 dibits and uses thus an average of 24 words. Because the SCW is also coming from the NLF-stream, it is expected that the n-th SCW is at the position $25n$ of the NLF-stream. The probability that the n-th SCW is indeed on this position, is equal to the probability that n SCW's determine the stuttering of *exactly* $24n$ words from the NLF-stream. This means that half of the $16n$ dibits should determine the stuttering of one word, the other half should determine the stuttering of two words. This probability can be expressed as follows:

$$P(position(SCW[n]) = 25n) = \binom{16n}{8n} (\frac{1}{2})^{8n+8n} = \frac{16n!}{(8n!)^2 \cdot 2^{16n}} \, . \tag{20}$$

This equation can be approximated by using the Stirling equation for large faculties:

$$n! \approx \sqrt{2\pi n} \cdot n^n \cdot e^{-n} \, . \tag{21}$$

This gives the following equation:

$$P(position(SCW[n]) = 25n) \approx \frac{\sqrt{2\pi 16n} \cdot (16n)^{16n} \cdot e^{-16n}}{(\sqrt{2\pi 8n} \cdot (8n)^{8n} \cdot e^{-8n})^2 \cdot 2^{16n}} = \frac{1}{\sqrt{8\pi n}} \, . \tag{22}$$

Simulation shows that this equation is a very good approximation.

When this line of reasoning is reversed, we see that the probability that the $25n$-th word of the NLF-stream will be the n-th SCW, is also proportional to $1/\sqrt{n}$.

The probability for the following words to be at their most probable position in the key stream will be similar. It can be written as:

$$P(position(v_n) = \frac{n - \lfloor \frac{n}{25} \rfloor}{2}) = \frac{\lambda}{\sqrt{\frac{8\pi n}{25}}}\,, \tag{23}$$

where λ is a constant. Simulation shows that this hypothesis is indeed correct. From the simulations it follows that $\lambda \simeq 0.84$.

The probability that $v_{4\cdot 2^{32}-4}$, $v_{15\cdot 2^{32}-4}$, $v_{17\cdot 2^{32}-4}$ appear in the key stream at their most probable position can now be calculated with (23). These probabilities – given that the words before are present in the key stream at their most probable position – are $2^{-17.3}$, $2^{-18.0}$ and $2^{-16.8}$ respectively.

The total probability p_0 can now be calculated:

$$p_0 = 0.217 \cdot 0.198 \cdot 2^{-17.3} \cdot 2^{-18.0} \cdot 2^{-16.8} = 2^{-56.6} \tag{24}$$

With probability p_0, $v_{17\cdot 2^{32}-4}$, $v_{15\cdot 2^{32}-4}$, $v_{4\cdot 2^{32}-4}$, v_{13} and v_{11} are present in the key stream at their most probable position. They will appear in the key stream, XORed with 0, C or C'. This will however not affect the distribution: we are considering the XOR of bit 29 and bit 30, and for C both bits are 1. Thus, for C, C' and for 0, the XOR of bit 29 and 30 is always zero, so the constant values are always eliminated. This yields the following for the calculation of the Chernoff information between the distribution of the key stream P_Y and the uniform distribution P_U:

$$C(P_Y, P_U) \approx p_0^2 \cdot C(P_W, P_U) = 2^{2\cdot -56.6} \cdot 2^{-81.5} = 2^{-194.7}\,. \tag{25}$$

P_W is the distribution of the NLF-stream. For an error probability of $P_e = 2^{-32}$ this means we need $32 \cdot 2^{194.7} = 2^{199.7}$ samples. The attack requires $2^{199.7}$ sequences of $17 \cdot 2^{32}$ words from the key stream, this is a total of $L = 2^{200} \geq 2^{199.7} + 17 \cdot 2^{32}$ words.

4.2 Extending the Attack on Sober-t16 to Sober-t32

In [5], Ekdahl and Johansson present a distinguishing attack on the full Sober-t16 (with stuttering). As the paper mentions, the proposed methods are expected to be applicable against Sober-t32 as well, but no complexity expression is given due to computational limitations. In this section, we derive the expected complexity by making a number of (realistic) probabilistic assumptions.

Distributions. The distinguishing attack in [5] starts by approximating the equation of the NLF (5) of the cipher with a linear function and analyzing the distribution of the noise w_t for different values of K.

$$v_t = \Big[s_t \oplus s_{t+1} \oplus s_{t+6} \oplus s_{t+13} \oplus s_{t+16}\Big] \oplus w_t = \Omega_t \oplus w_t \tag{26}$$

In the case of SOBER-t16, the noise w_t can take on 2^{16} values and the probability of each of these values can easily be estimated by simulations (in [5], an accurate estimation is obtained by taking 2^{38} samples). A similar simulation for SOBER-t32, however, would require an impractical amount of memory and a huge processing time. This motivates us to derive a complexity expression based on the average non-uniformity of w_t, instead of on its full distribution.

In the following, $P_u(w) = p = 1/N = 2^{-32}$ stands for the uniform distribution, $P_w(w) = p \cdot (1 + \epsilon_w(w))$ for the noise distribution and $\sigma^2_{\epsilon_w}$ for the non-uniformity.

To estimate the non-uniformity of the noise distribution, we will now simulate $\epsilon_w(w)$ for a limited number of values w and assume that these samples are representative for the full distribution.

A first straightforward way to find an estimate of $\epsilon_w(w)$ for a given value of w would consist in randomly choosing n sets $(s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16})$, computing w_t for each of them and analyzing the frequency at which w_t equals w.

However, in order to speed up the convergence, we will follow a somewhat different approach. We first uniformly choose n sets $(s_t, s_{t+1}, s_{t+6}, s_{t+16})$ and compute

$$a_t = \big(f_w(s_t \boxplus s_{t+16}) \boxplus s_{t+1} \boxplus s_{t+6}\big) \oplus K \tag{27}$$

$$b_t = s_t \oplus s_{t+1} \oplus s_{t+6} \oplus s_{t+16} \,. \tag{28}$$

Then for each set we count the number of s_{t+13} for which

$$w_t = (a_t \boxplus s_{t+13}) \oplus (b_t \oplus s_{t+13}) = w \ . \tag{29}$$

This number can directly be derived from the bits of a_t and b_t, i.e., it is not needed to run through all possible values of s_{t+13}. The estimation for $\epsilon_w(w)$ obtained this way is expected to converge more rapidly, as each step takes into account 2^{32} possible values for $(s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16})$. In case a_t and b_t were uncorrelated and uniformly distributed, one would find that

$$\text{error} = \sqrt{\frac{\left(\frac{3}{4}\right)^{32}}{p \cdot n}} \approx \frac{1}{\sqrt{2^{13} \cdot p \cdot n}} \,. \tag{30}$$

For both SOBER-t16 and SOBER-t32, we performed the simulations for different values of K and different sets of 256 consecutive w. From the estimated values of $\epsilon_w(w)$, we calculated the non-uniformity $\sigma^2_{\epsilon_w}$. Eventually, the minimal and average non-uniformity were found to be 2^{-10} and 2^{-8} for SOBER-t16 and 2^{-9} and 2^{-7} for SOBER-t32.

Combining Distributions. In the next step of the attack described in [5], different shifted versions of w_t are combined in order to eliminate the unknown LFSR words s_t.

$$W_t = w_{t+17} \oplus w_{t+15} \oplus w_{t+4} \oplus \alpha \cdot w_t \,. \tag{31}$$

Next, we need to calculate the non-uniformity of the full noise W_t. In Appendix C, a general expression for the non-uniformity of $z = x \oplus y$ given $\sigma^2_{\epsilon_x}$ and $\sigma^2_{\epsilon_y}$ is derived.

Distinguishing Distributions: Chernoff Information. The number of samples required to distinguish a distribution $P_x(x)$ from the uniform distribution is determined by the Chernoff information [5]. This quantity can easily be derived from the non-uniformity $\sigma^2_{\epsilon_x}$:

$$-\log_2 \sum_x \sqrt{P_x(x) \cdot p} = -\log_2 \frac{1}{N} \sum_x \sqrt{1 + \epsilon_x(x)} \tag{32}$$

$$\approx -\log_2 \frac{1}{N} \sum_x \left[1 + \frac{1}{2} \cdot \epsilon_x(x) - \frac{1}{8} \cdot \epsilon_x(x)^2\right] \tag{33}$$

$$= -\log_2 \left(1 - \frac{1}{8} \cdot \sigma^2_{\epsilon_x}\right) \approx \frac{1}{8 \cdot \ln 2} \cdot \sigma^2_{\epsilon_x} \,. \tag{34}$$

Complexity of the Attack on Sober-t32. Using the formulae in the previous sections we are now able to estimate the complexity of a distinguishing attack on Sober-t32.

- From the simulated approximation for the minimal non-uniformity of the noise w_t, 2^{-9}, we can find the expected non-uniformity of the full noise W_t by using (51) and (52):

$$\sigma^2_{\epsilon_W} \approx \frac{3}{N^3} \cdot \left(\sigma^2_{\epsilon_w}\right)^4 \approx 2^{-130} \,. \tag{35}$$

  To derive this result, the last two sums of the expression $W_t = (w_{t+17} \oplus w_{t+15}) \oplus (w_{t+4} \oplus \alpha \cdot w_t)$ are considered to be sums of independent distributions.
- The stuttering adds an additional unknown constant C_t to the noise W_t. This constant can be written as $C_t = c_{t+17} \oplus c_{t+15} \oplus c_{t+4} \oplus \alpha \cdot c_t$ with $c_t \in \{0, C, C'\}$. Assuming that all 12 possible values of C_t are equiprobable (which is the worst case), we find $\sigma^2_{\epsilon_C} \approx N/12$ and

$$\sigma^2_{\epsilon_{W \oplus C}} \approx \frac{1}{12} \cdot \sigma^2_{\epsilon_W} \approx 2^{-134} \,. \tag{36}$$

- As explained in [5], we will only get this non-uniform distribution if we made correct guesses for the positions of the w_t in the key stream. Simulations show that this happens with a probability $p_0 = 2^{-5.5}$. The final non-uniformity will therefore be reduced to:

$$\sigma^2_{\epsilon} \approx p_0^2 \cdot \sigma^2_{\epsilon_{W \oplus C}} \approx 2^{-145} \,. \tag{37}$$

- In order to obtain a sufficiently small probability of error (say 2^{-16}), even after applying the distinguisher for all 2^{32} possible values of K, we need a stream of at least 48 times the inverse of the Chernoff information. This yields the final complexity:

$$48 \cdot \frac{8 \cdot \ln 2}{\sigma^2_{\epsilon}} \approx 2^{153} \,. \tag{38}$$

5 Conclusion

In this paper, some new attacks on Sober-t32 have been presented.

A first attack is a $2^{252.21}$ Guess and Determine attack on unstuttered Sober-t32. This attack is due to a probabilistic property of the t-class of stream ciphers found in their S-box construction: The relationship between 8 bits in and 8 bits out is not diffused to other positions in the word. Even a cyclic shift at the end of the S-box would have destroyed the attack. In order to prevent similar attacks, we suggest that in word-based LFSRs, the NLF should implicate the whole word, and not just a part of the word as in Sober-t. Then the attacker will not gain any profit by guessing some bits of the words.

Stuttering prevents the attack - not so much by the uncertainty it introduces as by the fact that consecutive words don't appear in the key stream. In fact, a timing attack [7] on the stuttering can reveal a long sequence of consecutive words that are not eliminated, thus enabling the GD-attack described above (see [8]).

Next, two ways of mounting distinguishing attacks on full Sober-t32 have been elaborated. Both attacks are based on the attacks described in [5]. The first attack is an adaptation of the attack on unstuttered Sober-t32, such that it also works on full Sober-t32. This attack could distinguish the Sober-t32-key stream from a uniform source with about 2^{200} output words. The second attack extends the attack on full Sober-t16 to full Sober-t32. This attack could distinguish the Sober-t32 key stream from a uniform source with about 2^{153} output words. Furthermore, these distinguishing attacks show that the stuttering cannot frustrate all attacks requiring vast amounts of key stream. The stuttering unit is however very expensive as it lowers the performance of the cipher by 52%. We would thus not recommend the usage of such parts in stream ciphers.

The attacks described are only possible theoretically. However, they are more efficient than exhaustive key search. This implies that Sober-t32 does not offer the security required by the NESSIE competition.

References

1. P. Hawkes and G. Rose, *Primitive Specification and Supporting Documentation for* Sober*-t32 Submission to NESSIE*, Proceedings of the First Open NESSIE Workshop, 2000.
2. New European Schemes for Signature, Integrity and Encryption, http://www.cryptonessie.org
3. D. Bleichenbacher, S. Patel and W. Meier, *Analysis of the SOBER stream cipher*, TIA contribution TR45.AHAG/99.08.30.12, 1999.
4. C. De Cannière, *Guess and Determine Attack on SOBER*, NESSIE report NES/DOC/KUL/WP5/010/a, 2001.
5. P. Ekdahl and T. Johansson, *Distinguishing Attacks on* Sober*-t16 and t32*, Fast Software Encryption 2002, LNCS 2365, J. Daemen, V. Rijmen, Eds., Springer-Verlag, pp. 210-224, 2002.
6. T. Herlestam, *On Functions of Linear Shift Register Sequences*, Eurocrypt 85, LNCS 219, F. Pichler, Ed., Springer-Verlag, pp. 119-129, 1985.

7. M. Schafheutle, *A First Report on the Stream Ciphers* SOBER-*t16 and* SOBER-*t32*, NESSIE document NES/DOC/SAG/WP3/025/02, NESSIE, 2001.
8. J. Lano and G. Peeters, *Cryptanalyse van NESSIE kandidaten (Dutch)*, Master's Thesis, K.U. Leuven, May 2002.
9. N. Courtois, A. Klimov, J. Patarin, A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt 2000, LNCS 1807, B. Preneel, Ed., pp. 392-407, Springer-Verlag, 2000.
10. N. Courtois and J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Cryptology ePrint Archive, Report 2002/044, http://eprint.iacr.org, 2002.

A Calculating the Entropy of the Carry Bits

In the following, p_i^j (and also q_i^j and r_i^j) stands for: the probability that the value of the carry bit to the i-th bit is equal to j. The bits are numbered from least to most significant. Bit 0 is the least significant bit, bit 31 is the most significant bit. The notations p_i^j, q_i^j and r_i^j are used to distinguish between the three different scenarios which are discussed below:

- Two random 32-bit words are added. One can see that p_1^0 equals $\frac{3}{4}$, and that p_1^1 equals $\frac{1}{4}$. The values of the subsequent carry bits can be obtained with the following recursion:

$$\begin{cases} p_{i+1}^0 = \frac{3}{4}.p_i^0 + \frac{1}{4}.p_i^1 \\ p_{i+1}^1 = \frac{1}{4}.p_i^0 + \frac{3}{4}.p_i^1 \end{cases} \tag{39}$$

  The values of p_i^0 and p_i^1 can now be calculated for all i. Both values converge rapidly to $\frac{1}{2}$. For $i = 24$, the carry bit of interest here, this is a very good approximation.
  The entropy H for this carry value is thus:

$$H = -\sum p^i.\log(p^i) = -\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2}) = 1 \tag{40}$$

  The carry bit o_1 corresponds to this case. The entropy of the carry bit o_1 is 1.
- Now three words are added up. The carry value can now be 0, 1 or 2. The probabilities for the first carry value are $q_1^0 = \frac{1}{2}$, $q_1^1 = \frac{1}{2}$ and $q_1^2 = 0$, and the recursion formula is:

$$\begin{cases} q_{i+1}^0 = \frac{1}{2}.q_i^0 + \frac{1}{8}.q_i^1 \\ q_{i+1}^1 = \frac{1}{2}.q_i^0 + \frac{3}{4}.q_i^1 + \frac{1}{2}.q_i^2 \\ q_{i+1}^2 = \frac{1}{8}.q_i^1 + \frac{1}{2}.q_i^2 \end{cases} \tag{41}$$

  For increasing i, the values converge rapidly towards $q^0 = \frac{1}{6}, q^1 = \frac{2}{3}$ and $q^2 = \frac{1}{6}$.

The entropy H converges thus towards the value:

$$H = -\sum q^i.\log(q^i) = -\frac{1}{6}\log(\frac{1}{6}) - \frac{2}{3}\log(\frac{2}{3}) - \frac{1}{6}\log(\frac{1}{6}) = 1.25 \qquad (42)$$

– In a third scenario, we consider the carry value for the sum $((x+y) \oplus C)+z+u$. As an extra constraint, all bits of C that are more significant (i.e. more to the left) than the carry value considered must be zero. It can be seen that this case is a combination of the two previously considered cases: the total carry value is the sum of the carry value from the addition of two words (x and y) and of the carry value from the addition of three words ($x + y \oplus C$, z and u). It is then easy to see that the probabilities converge towards the following values:

$$\begin{cases} r^0 = p^0.q^0 = \frac{1}{2}\cdot\frac{1}{6} = \frac{1}{12} \\ r^1 = p^0.q^1 + p^1.q^0 = \frac{1}{2}\cdot\frac{2}{3} + \frac{1}{2}\cdot\frac{1}{6} = \frac{5}{12} \\ r^2 = p^0.q^2 + p^1.q^1 = \frac{1}{2}\cdot\frac{1}{6} + \frac{1}{2}\cdot\frac{2}{3} = \frac{5}{12} \\ r^3 = p^1.q^2 = \frac{1}{2}\cdot\frac{1}{6} = \frac{1}{12} \end{cases} \qquad (43)$$

The entropy H converges thus towards the following value.

$$H = -\sum r^i.\log(r^i) = -\frac{1}{12}\log(\frac{1}{12}) - \frac{5}{12}\log(\frac{5}{12}) - \frac{5}{12}\log(\frac{5}{12}) - \frac{1}{12}\log(\frac{1}{12}) = 1.65 \qquad (44)$$

The carry value o_2 corresponds to this case. The entropy of the carry value o_2 is 1.65.

B Using Multivariate Quadratic Equations

By knowing the MSB of the words of the LFSR, we have eliminated the main non-linearity in the algorithm. A consequence is that the equation (9) in the least significant bit is completely linear, a fact that has been exploited above.

For the 23 other bits, the equations will be similar to this equation, but some carry bits will appear in these equations. This is the only (small) non-linearity in the system. These carry bits do not represent new unknowns as they can be written as the product of bits. We can write the following bitwise equations for each v_t^i for i going from 1 to 23 (and similarly for $v_{t+1}, \ldots v_{t+k}$:

$$\begin{cases} v_t^i = SBOX^i + s_t^i + s_{t+16}^i + ca_t^i + s_{t+1}^i + s_{t+6}^i + cb_t^i + K^i + s_{t+13}^j + cc_t^i \\ ca_t^i = s_t^{i-1} \cdot s_{t+16}^{i-1} + s_t^{i-1} \cdot ca_t^{i-1} + s_{t+16}^{i-1} \cdot ca_t^{i-1} \\ cb_t^i = \ldots \\ cc_t^i = \ldots \end{cases} \qquad (45)$$

We have introduced $3\cdot 23\cdot(k+1)$ new unknowns and $4\cdot 23\cdot(k+1)$ new equations. In order to have enough equations we need:

$$33\cdot k+1+4\cdot 23\cdot(k+1)\geq 24\cdot(17+k)+24+3\cdot 23\cdot(k+1) \Longleftrightarrow k\geq 12.75\,. \quad (46)$$

So we can now consider all possible number of iterations beginning with 13. The more iterations we will use, the more our system will be overdefined. The so-called XL[9] and XSL techniques[10] can be used to solve this system. This may lead to a more efficient attack. However, the complexity of these algorithms is not well understood and may be clarified in future research.

C Calculating the Non-Uniformity of the Sum of Distributions

C.1 The Sum of Two Independent Distributions

Let x an y be drawn from two independent distributions with non-uniformity $\sigma^2_{\epsilon_x}$ and $\sigma^2_{\epsilon_y}$.

$$P_x(x) = p\cdot\big(1+\epsilon_x(x)\big) \quad (47)$$
$$P_y(y) = p\cdot\big(1+\epsilon_y(y)\big)\,. \quad (48)$$

The distribution of $z = x\oplus y$ can be written as:

$$P_z(z) = \sum_{x\oplus y=z} P_x(x)\cdot P_y(y) \quad (49)$$
$$= p\cdot\big(1+\epsilon_z(z)\big)\,. \quad (50)$$

Exploiting the fact that the sum of all ϵ equals zero and that the distributions of x and y are independent, we obtain:

$$\begin{aligned}
E\left(\sigma^2_{\epsilon_z}\right) &= \frac{1}{N}\sum_z E\left(\epsilon_z(z)^2\right)\\
&= \frac{1}{N}\sum_z E\left(\left[p^{-1}\cdot\sum_{x\oplus y=z}P_x(x)\cdot P_y(y)-1\right]^2\right)\\
&= \frac{1}{N}\sum_z E\left(\left[\frac{1}{N}\sum_x\epsilon_x(x)+\frac{1}{N}\sum_y\epsilon_y(y)+\frac{1}{N}\sum_{x\oplus y=z}\epsilon_x(x)\cdot\epsilon_y(y)\right]^2\right)\\
&= \frac{1}{N}\sum_z E\left(\left[\frac{1}{N}\sum_{x\oplus y=z}\epsilon_x(x)\cdot\epsilon_y(y)\right]^2\right)\\
&= \frac{1}{N}\sum_d\left[\frac{1}{N}\sum_x E\big(\epsilon_x(x)\cdot\epsilon_x(x\oplus d)\big)\cdot\frac{1}{N}\sum_y E\big(\epsilon_y(y)\cdot\epsilon_y(y\oplus d)\big)\right]\,.
\end{aligned}$$

To calculate the expression between the square brackets we distinguish the cases $d = 0$ and $d \neq 0$. When $d = 0$, we have $E\big(\epsilon_x(x) \cdot \epsilon_x(x \oplus 0)\big) = \sigma^2_{\epsilon_x}$. In all other cases we may assume that the expected value of $\epsilon_x(x) \cdot \epsilon_x(x \oplus d)$, over all possible distributions with non-uniformity $\sigma^2_{\epsilon_x}$, is independent of d and equal to $-\sigma^2_{\epsilon_x}/(N-1)$ (because ϵ_x sums to zero). The same applies for ϵ_y and hence

$$
\begin{aligned}
E\left(\sigma^2_{\epsilon_z}\right) &= \frac{1}{N}\left[\sigma^2_{\epsilon_x} \cdot \sigma^2_{\epsilon_y} + (N-1) \cdot \frac{-\sigma^2_{\epsilon_x}}{N-1} \cdot \frac{-\sigma^2_{\epsilon_y}}{N-1}\right] \\
&= \frac{1}{N-1} \cdot \sigma^2_{\epsilon_x} \cdot \sigma^2_{\epsilon_y} \,. \qquad (51)
\end{aligned}
$$

C.2 The Sum of Two Identical Distributions

A similar expression can be derived for the non-uniformity of $z = x \oplus x'$ with x and x' drawn from a single distribution.

$$
\begin{aligned}
E\left(\sigma^2_{\epsilon_z}\right) &= \frac{1}{N}\sum_z E\left(\epsilon_z(z)^2\right) \\
&= \frac{1}{N}\sum_z E\left(\left[p^{-1} \cdot \sum_{x \oplus x' = z} P_x(x) \cdot P_x(x') - 1\right]^2\right) \\
&= \frac{1}{N}\sum_z E\left(\left[\frac{2}{N}\sum_x \epsilon_x(x) + \frac{1}{N}\sum_{x \oplus x' = z} \epsilon_x(x) \cdot \epsilon_x(x')\right]^2\right) \\
&= \frac{1}{N}\sum_z E\left(\left[\frac{1}{N}\sum_{x \oplus x' = z} \epsilon_x(x) \cdot \epsilon_x(x')\right]^2\right) \\
&= \frac{1}{N}\sum_z \left[\frac{1}{N}\sum_{x \oplus x' = z} \frac{1}{N}\sum_{x'' \oplus x''' = z} E\big(\epsilon_x(x) \cdot \epsilon_x(x') \cdot \epsilon_x(x'') \cdot \epsilon_x(x''')\big)\right] \\
&= \frac{1}{N}\left[\frac{\tau^4_{\epsilon_x}}{N} + 3 \cdot \frac{N \cdot \sigma^4_{\epsilon_x} - \tau^4_{\epsilon_x}}{N} + \frac{3 \cdot N \cdot \sigma^4_{\epsilon_x} - 6 \cdot \tau^4_{\epsilon_x}}{N \cdot (N-3)}\right] \\
&= \frac{1}{N \cdot (N-3)}\left[3 \cdot (N-2) \cdot \sigma^4_{\epsilon_x} - 2 \cdot \tau^4_{\epsilon_x}\right] , \qquad (52)
\end{aligned}
$$

with

$$
\begin{aligned}
\tau^4_{\epsilon_x} &= \frac{1}{N}\sum_x \epsilon_x(x)^4 \qquad (53) \\
&= \mathcal{O}\left(\sigma^4_{\epsilon_x}\right) . \qquad (54)
\end{aligned}
$$