Algebraic Attacks on SOBER-t32 and SOBER-t16 without stuttering

Joo Yeon Cho and Josef Pieprzyk*

Center for Advanced Computing – Algorithms and Cryptography,
Department of Computing, Macquarie University,
NSW, Australia, 2109
{jcho,josef}@ics.mq.edu.au

Abstract. This paper presents algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. For unstuttered SOBER-t32, two different attacks are implemented. In the first attack, we obtain multivariate equations of degree 10. Then, an algebraic attack is developed using a collection of output bits whose relation to the initial state of the LFSR can be described by low-degree equations. The resulting system of equations contains 2^{69} equations and monomials, which can be solved using the Gaussian elimination with the complexity of $2^{196.5}$. For the second attack, we build a multivariate equation of degree 14. We focus on the property of the equation that the monomials which are combined with output bit are linear. By applying the Berlekamp-Massey algorithm, we can obtain a system of linear equations and the initial states of the LFSR can be recovered. The complexity of attack is around $O(2^{100})$ with 2^{92} keystream observations. The second algebraic attack is applicable to SOBER-t16 without stuttering. The attack takes around $O(2^{85})$ CPU clocks with 2^{78} keystream observations.

Keywords : Algebraic attack, stream ciphers, linearization, NESSIE, SOBER-t32, SOBER-t16, modular addition, multivariate equations

1 Introduction

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters of a plaintext message one at a time, using a stream of pseudorandom bits. Stream ciphers generally offer a better performance compared with block ciphers. They are also more suitable for implementations where computing resources are limited (mobile phones) or when characters must be individually processed (reducing the delay) [4].

Recently, there were two international calls for cryptographic primitives. NESSIE is an European initiative [2] and CRYPTREC [1] is driven by Japan. Many stream ciphers have been submitted and evaluated by the international cryptographic community. NESSIE announced the final decision at Feb. 2003 and none of candidates of stream ciphers was selected in the final report.

* Supported by ARC Discovery grant DP0451484

According to the final NESSIE security report [3], there were four stream cipher primitives which were considered during the phase II : BMGL [15], SNOW [12], SOBER-t16 and SOBER-t32 [17]. The security analysis of these stream ciphers is mainly focused on the distinguishing and guess-determine attacks. Note that ciphers for which such attacks exist are excluded from the contest (even if those attack do not allow to recover the secret elements of the cipher).

For SOBER-t16 and SOBER-t32, there are distinguishing attacks that are faster than the key exhaustive attack. SOBER-t16 is distinguishable from the truly random keystream with the work factor of 2^{92} for the version without stuttering and with the work factor of 2^{111} for the version with stuttering [13]. The same technique is used to construct a distinguisher for SOBER-t32 with complexity of $2^{86.5}$ for the non-stuttering version [13]. For the version with stuttering [14], the distinguisher has the complexity 2^{153}.

The distinguishing attacks are the weakest form of attack and normally identify a potential weakness that may lead to the full attack that allows to determine the secret elements (such as the initial state) of the cipher. Recent development of algebraic attacks on stream ciphers already resulted in a dramatic cull of potential candidates for the stream cipher standards. The casualties of the algebraic attacks include Toyocrypt submitted to CRYPTREC [7] and LILI-128 submitted to NESSIE [10].

In this paper, we present algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. For unstuttered SOBER-t32, two different attacks are implemented. In the first attack, we apply indirectly the algebraic attack on combiner with memory, even though SOBER-t32 does not include an internal memory state. We extract a part of most significant bits of the addition modulo 2^{32} and the carry generated from the part of less significant bits is regarded as the internal memory state (which is unknown). Our attack can recover the initial state of LFSR with the workload of $2^{196.5}$ by 2^{69} keystream observations, which is faster than the exhaustive search of the 256-bit key.

For the second attack, we build a multivariate equation of degree 14. This equation has a property that the monomials which are combined with output bit are linear. By applying the Berlekamp-Massey algorithm, we can obtain a system of simple linear equations and recover the initial states of the LFSR. The attack takes $O(2^{100})$ CPU clocks with around 2^{92} keystream observations.

We apply the second algebraic attack to SOBER-t16 without stuttering. Our attack can recover the initial state of LFSR with the workload of $O(2^{85})$ using 2^{78} keystream observations for unstuttered SOBER-t16.

This paper is organized as follows. In Section 2, an algebraic attack method is briefly described. The structure of SOBER-t32 is given in Section 3. In Section 4, the first algebraic attack on SOBER-t32 is presented. the second algebraic attack on SOBER-t32 is presented in Section 5. In Section 6, an algebraic attack on SOBER-t16 is presented. Section 7 concludes the paper.

2 Algebraic attacks

2.1 Previous works

The first application of algebraic approach for analysis of stream ciphers can be found in the Courtois' work [7]. The idea behind it is to find a relation between the initial state of LFSR and the output bits that is expressible by a polynomial of a low degree. By accumulating enough observations (and corresponding equations) the attacker is able to create a system of equations of a low algebraic degree. The number of monomials in these equations is relatively small (as the degree of each equation is low). We treat monomials as independent variables and solve the system by the Gaussian elimination. For Toyocrypt, the algebraic attack shown in [7] is probabilistic and requires the workload of 2^{92} with 2^{19} keystream observations. In [10], the authors showed that Toyocrypt is breakable in 2^{49} CPU clocks with 20K bytes of keystream. They also analyzed the NESSIE submission of LILI-128 showing that it is breakable within 2^{57} CPU clocks with 762 GBytes memory. Recently, the algebraic approach has been extended to analyze combiners with memory [5, 6, 8]. Very recently, the method that allows a substantial reduction of the complexity of all these attacks is presented in [9].

2.2 General description of algebraic attacks

Let $S_0 = (s_0, \cdots, s_{n-1})$ be an initial state of the linear shift register at the clock 0. The state variables are next input to the nonlinear block producing the output $v_0 = NB(S_0)$ where NB is a nonlinear function transforming the state of the LFSR into the output. At each clock t, the state is updated according to the following relation $S_t = L(s_t, \cdots, s_{n-1+t})$ with L being a multivariate linear transformation. The output is $v_t = NB(S_t)$.

The general algebraic attack on such stream ciphers works as follows. For details see [7] or [10].

- Find a multivariate relation Q of a low degree d between the state bits and the bits of the output. Assume that the relation is $Q(S_0, v_0) = 0$ for the clock 0.
- The same relation holds for all consecutive clocks t so

$$Q(S_t, v_t) = Q(L^t(S_0), v_t) = 0$$

  where the state S_t at the clock t is a linear transformation of the initial state S_0 or $S_t = L^t(S_0)$. Note that all relations are of the same degree d.
- Given consecutive keystream bits $v_0, \cdots v_{M-1}$, we obtain a system of M equations with monomials of degree at most d. If we collect enough observations so the number of linearly independent equations is at least as large as the number T of monomials of degree at most d, then the system has a unique solution revealing the initial state S_0. We can apply the Gaussian reduction algorithm that requires $7 \cdot T^{\log_2 7}$ operations [18].

Finding relations amongst the input and output variables is a crucial task in each algebraic attack. Assume that we have a nonlinear block with n binary inputs and m binary outputs or simply the $n \times m$ S-box over $GF(2)$. The truth table of the box consists of 2^n rows. The columns point out all input and output variables (monomials of degree 1). We can add columns for all terms (monomials) of degree 2. There are $\binom{n+m}{2}$ such terms. We can continue adding columns for higher degree monomials until

$$2^n < \sum_{i=1}^{d} \binom{n+m}{i}$$

where d is the highest degree of the monomials. Informally, the extended truth table can be seen as a matrix having more columns than rows so there are some columns (monomials) that can be expressed as a linear combination of other columns establishing a required relations amongst monomials of the S-box.

3 Brief description of SOBER-t32

3.1 Notation

All variables operates on 32-bit words. Refer to the Figures 1.

- $\oplus$: addition in $GF(2^{32})$, $\boxplus$: addition modulo 2^{32}.
- $s_{i,j}$: the j-th bit of the state register s_i.
- $s_{i,j\to k}$: a consecutive bit stream from the j-th bit to k-th bit of s_i.
- $x = s_0 \boxplus s_{16}$. x_i is the i-th bit of x.
- α : the output of the first S-box. α_i is the i-th bit of α.

3.2 SOBER-t32

SOBER-t32 is a word-oriented synchronous stream cipher. It operates on 32-bit words and has a secret key of 256 bits (or 8 words). SOBER-t32 consists of a linear feedback shift register (LFSR) having 17 words (or 544 bits), a non-linear filter (NLF) and a form of irregular decimation called stuttering. The LFSR produces a stream S_t of words using operations over $GF(2^{32})$. The vector $S_t = (s_t, \cdots, s_{t+16})$ is known as the *state* of the LFSR at time t, and the state $S_0 = (s_0, \cdots, s_{16})$ is called the *initial state*. The initial state and a 32-bit, key-dependent constant called K are initialized from the secret key by the key loading procedure.

A Nonlinear Filter (NLF) takes some of the states, at time t, as inputs and produces a sequence v_t. Each output stream v_t is obtained as $v_t = NLF(S_t) = F(s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16}, K)$. The function F is described in the following subsection. The stuttering decimates the stream that is produced by NLF and outputs the key stream. The detailed description is given in [17].

The linear feedback shift register SOBER-t32 uses an LFSR of length 17 over $GF(2^{32})$. Each register element contains one 32-bit word. The contents of the LFSR at time t is denoted by $s_t, \cdots, s_{t+16}$. The new state of the LFSR is generated by shifting the previous state one step (operations are performed on words) and updating the state of the most significant word according to the following linear equation

$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \beta \cdot s_t$$

where $\beta = $ 0xc2db2aa3.

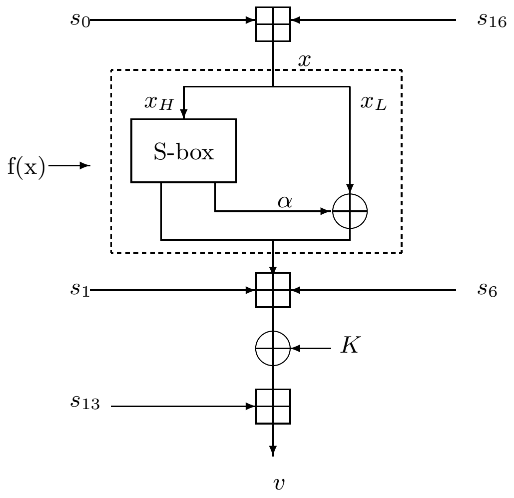

Fig. 1. The non-linear filter of SOBER-32 without stuttering

The nonlinear filter At time t, the nonlinear filter takes five words from the LFSR states, $s_t, s_{t+1},\ s_{t+6},\ s_{t+13}, s_{t+16}$ and the constant K as the input and produces the output v_t. The nonlinear filter consists of function f, three adders modulo 2^{32} and the XOR addition. The value K is a 32-bit key-dependent constant that is determined during the initialization of LFSR. K is kept constant throughout the entire session. The function f translates a single word input into a word output. The output of the nonlinear filter, v_t, is equal to

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

where the function $f(x)$ is illustrated in Figure 1.

The function uses a S-box with 8-bit input and 32-bit output. The input x to the function $f(x)$ is split into two strings, x_H and x_L. The first string x_H is transformed by the S-box that gives α.

The stuttering The output of the stream cipher is obtained by decimating the output of the NLF in an irregular way. It makes correlation attack harder. In this paper, however, we will ignore the stuttering phase and assume that the attacker is able to observe the output v_t directly.

4 The first attack on SOBER-t32 without stuttering

4.1 Building multivariate equations for the non-linear filter

If we look at the structure of the non-linear filter, the following equations relating the state $S_0 = (s_0, \cdots, s_{16})$ of LFSR and the intermediate values x, α with the output keystream v are established.

$$\begin{cases} x_0 = s_{0,0} \oplus s_{16,0} \\ \alpha_0 = x_0 \oplus s_{1,0} \oplus s_{6,0} \oplus K_0 \oplus s_{13,0} \oplus v_0 \\ x_1 = s_{0,1} \oplus s_{16,1} \oplus s_{0,0}s_{16,0} \\ \alpha_1 = x_1 \oplus s_{1,1} \oplus s_{6,1} \oplus K_1 \oplus s_{13,1} \oplus v_1 \oplus K_0 s_{13,0} \oplus \\ \quad (x_0 \oplus \alpha_0)(s_{1,0} \oplus s_{6,0} \oplus s_{13,0}) \oplus s_{1,0}(s_{6,0} \oplus s_{13,0}) \oplus s_{6,0}s_{13,0} \\ \vdots \end{cases} \tag{1}$$

Lemma 1. *Variables x_i and α_i can be expressed as an equation of degree $d \geq i+1$ over the bits of the state variables.*

Let's consider the first modular addition $x = s_0 \boxplus s_{16}$. If we denote *the carry* in the 24-th bit by ca, the addition modulo 2^{32} can be divided into two independent additions. One is the addition modulo 2^8 for the most significant 8 bits and another is the addition modulo 2^{24} for the remaining 24 bits. Then,

$$\begin{aligned} x_{24\to31} &= s_{0,24\to31} \boxplus s_{16,24\to31} \boxplus ca \quad (\text{modulo } 2^8) \\ x_{0\to23} &= s_{0,0\to23} \boxplus s_{16,0\to23} \quad (\text{modulo } 2^{24}) \end{aligned}$$

Lemma 2. *If the carry in the 24-th bit position is regarded as an unknown, the degrees of equations which are related to each x_i $(24 \leq i \leq 31)$ are reduced to $(i-23)$.*

Now, we reconstruct a partial block of non-linear filter : the addition modulo 2^{32} and the S-box. These two blocks can be considered as a single S-box. This is going to open up a possibility of reduction of degree of relations derived for the complex S-box. Furthermore, we consider the carry in the 24-th bit of addition modulo 2^{32} as an another input variable that is unknown (so we will avoid using it in the relation and treat it as an unknown state [6, 8]).

The structure of the new block is shown in Figure 2. The addition is modified to add two 8-bit strings ($s_{0,24\to31}$ and $s_{16,24\to31}$) with the carry c_{24} that is unknown. Thus this part is an addition modulo 2^8. The output is put to the S-box that has 32-bit output and amongst the output bits the least significant two bits are α_1, α_0. If we regard the carry in addition modulo 2^n as an internal memory state, an algebraic attack on combiner with memory is able to be applied to the new block since the structure of block is very similar to the model which is analyzed in [6, 8].

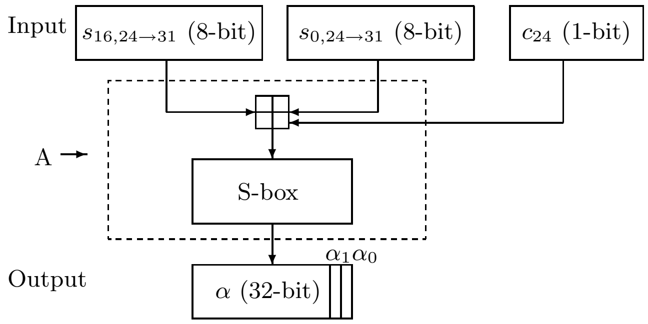

Fig. 2. A combined structure of partial addition modulo 2^8 and S-box

Lemma 3. *Let A be the combined block of modular addition and S-box with input s_0, s_{16} and output α. If the carry in the 24-th bit c_{24} is considered as an unknown, there is a multivariate equation that relates $s_{0,24\to31}, s_{16,24\to31}$ and output bits α_0, α_1 without the carry bit c_{24}.*

Proof. Let's create the following matrix.

- Rows are all the possibilities for $s_{16,24\to31}, s_{0,24\to31}$ and carry bit c_{24}. There are 2^{17} rows enumerating all possible values for the input bits.
- The columns are all the monomials of degree up to 9 which are coming from the input bits $s_{16,24\to31}, s_{0,24\to31}$ and the output bits α_1, α_0. The number of columns is

$$\sum_{i=0}^{9} \binom{18}{i} = 2^{17} + \frac{1}{2}\binom{18}{9} \cong 2^{17} + 2^{14.5}$$

If we apply the Gaussian elimination to this matrix, definitely we can obtain equations of degree up to 9 because the number of columns is greater than that of rows.

Let's denote the equation which is derived above as $F(s_{0,24\to31}, s_{16,24\to31}, \alpha_0, \alpha_1)$ of degree 9. If α_0 and α_1 are replaced by Equation (1) for the multivariate equation F, the degree of F becomes at most 10 which consists of only state bits of the LFSR, K and output bits of the NLF. In Appendix A, we present a toy example for illustrating the idea of the attack.

4.2 Complexity

We can apply the general algebraic attack to unstuttered SOBER-t32 by using the equation F. The number of monomials T of degree up to 10 are chosen from 544 unknowns.

$$T = \sum_{i=0}^{10} \binom{544}{i} \cong 2^{69}$$

From [10], the attack requires roughly $7 \cdot T^{\log_2 7} \cong 2^{196.5}$ CPU clocks with 2^{69} keystream observations.

In CRYPTO'03, a method that can solve the multivariate equations more efficiently by pre-computation is presented [9]. Even though the constant factor of complexity is not precisely estimated, this method seems to allow a further reduction of the complexity of our attack. According to [9], the main workload of algebraic attack is distributed into pre-computation stage and Gaussian reduction stage. The pre-computation is operated by LFSR synthesis method, say the Berlekamp-Massey algorithm. This step needs to take $\mathcal{O}(S \log S + Sn)$ steps if the asymptotically fast versions of the Berlekamp-Massey algorithm is used, where S is the size of the smallest linear dependency and n is the number of the state bits. Then, Gaussian elimination is applied to the monomials of reduced degree.

For our attack, the pre-computation operation needs about $\mathcal{O}(2^{78.3})$ steps. Then, Gaussian reduction can be applied to the monomials of degree up to 9. It takes about $\mathcal{O}(2^{180})$ steps with $2^{126.4}$ bits of memory.

5 The second attack on SOBER-32 without stuttering

5.1 An observation of modular addition

Let us consider the function of modular addition $c = a \boxplus b$ where $a = (a_{31}, \cdots, a_0)$, $b = (b_{31}, \cdots, b_0)$ and $c = (c_{31}, \cdots, c_0)$.

Lemma 4. *Let c_i be the i-th output bit of the modular addition. Then, $c_0 = a_0 \oplus b_0$, $c_1 = a_1 \oplus b_1 \oplus a_0 b_0$ and for $2 \leq i \leq 31$,*

$$c_i = a_i \oplus b_i \oplus a_{i-1} b_{i-1} \oplus \sum_{t=0}^{i-2} a_t b_t \{ \prod_{r=t+1}^{i-1} (a_r \oplus b_r) \}$$

Each c_i is expressed as a function of input bits of degree $i + 1$.

Theorem 1. *Let c_i, $24 \leq i \leq 31$ be the i-th output bit of modular addition $c = a \boxplus b$. If c_i is multiplied by $(1 \oplus a_{23} \oplus b_{23})$, then, the degree of $c_i \cdot (1 \oplus a_{23} \oplus b_{23})$ is reduced to $(i - 22)$.*

Proof. For $24 \leq i \leq 31$, c_i can be separated into two parts : one part includes $(a_{23} \oplus b_{23})$ and the remaining part does not. Therefore,

$$c_i = p(a_{23 \to i}, b_{23 \to i}) \oplus (a_{23} \oplus b_{23}) \cdot q(a_{0 \to i}, b_{0 \to i})$$

where p and q are functions of the input bits. Then,

$$c_i \cdot (1 \oplus a_{23} \oplus b_{23}) = p(a_{23 \to i}, b_{23 \to i}) \cdot (1 \oplus a_{23} \oplus b_{23})$$

From Lemma 4, the degree of $c_i \cdot (1 \oplus a_{23} \oplus b_{23})$ is $(i - 22)$.

5.2 Building a system equation for the non-linear filter

If we look at the structure of the non-linear filter, we can easily see that the following equation holds. (see Figure 1)

$$\alpha_0 = s_{0,0} \oplus s_{16,0} \oplus s_{1,0} \oplus s_{6,0} \oplus s_{13,0} \oplus v_0 \oplus K_0 \tag{2}$$

An equation for $\boldsymbol{\alpha_0}$. The bit α_0 is the least significant output bit of the S-box. α_0 can be represented by a non-linear equation where variables consist of only the input bits of the S-box. Let's construct the following matrix.

- Rows generate all the possibilities for $(x_{31}, \cdots, x_{24})$, so there are 2^8 rows enumerating all possible values for the input bits.
- The columns are all the monomials of degree up to 8 which are coming from the input bits $(x_{31}, \cdots, x_{24})$ and the least significant output bit α_0. The number of columns becomes $2^8 + 1$

If we apply the Gaussian elimination to this matrix, we can obtain a non-linear equation because the number of columns is greater than that of rows. Simulation shows that the degree of the equation for α_0 is 6. (See Appendix B)

In the next step, let's take a look at the first modular addition of the non-linear filter, which is $x = s_0 \boxplus s_{16}$. By Theorem 1, for $24 \leq i \leq 31$, $x_i \cdot (1 \oplus s_{0,23} \oplus s_{16,23})$ becomes

$$x_i \cdot (1 \oplus s_{0,23} \oplus s_{16,23}) = g(s_{0,23\to i}, s_{16,23\to i})$$

where g is a multivariate equation of degree up to $(i-22)$. Let A_i be a monomial which is built over variables from the set $\{x_{24}, \cdots, x_{31}\}$. Then,

$$A_i \cdot (1 \oplus s_{0,23} \oplus s_{16,23}) = G_i(s_{0,23\to 31}, s_{16,23\to 31})$$

where G_i is a non-linear equation of degree d_{G_i}.

Lemma 5. *The degree of $A_i \cdot (1 \oplus s_{0,23} \oplus s_{16,23})$ is at most 16.*

We can see that the number of variables which give effect on the degree d_{G_i} is at most 18, which is the set of variable $\{s_{0,23\to 31},\ s_{16,23\to 31}\}$. However, not all the monomials are available. For example, a monomial $s_{0,31} \cdot s_{0,30} \cdots s_{0,23} \cdot s_{16,31} \cdots s_{16,23}$, which is of degree 18, cannot happen. By careful inspection, we see that the degree of monomials is at most 16.

As shown in Appendix B, $\alpha_0 = \sum_i A_i$. If α_0 is multiplied by $(1 \oplus s_{0,23} \oplus s_{16,23})$, the monomials which include $(s_{0,23} \oplus s_{16,23})$ vanish.

Lemma 6. *The degree of $\alpha_0 \cdot (1 \oplus s_{0,23} \oplus s_{16,23})$ is at most 14.*

The degree of remaining monomials is expected to be not bigger than 16. However, computer simulation shows that the degree of monomials is not bigger than 14.

The degree of Equation (2). If we multiply $(1 \oplus s_{0,23} \oplus s_{16,23})$ by Equation (2), then we get

$$\begin{aligned} & \alpha_0 \cdot (1 \oplus s_{0,23} \oplus s_{16,23}) \\ & = (s_{0,0} \oplus s_{16,0} \oplus s_{1,0} \oplus s_{6,0} \oplus s_{13,0} \oplus v_0 \oplus K_0) \cdot (1 \oplus s_{0,23} \oplus s_{16,23}) \end{aligned} \tag{3}$$

Let's consider the left part of the equation. The bit α_0 plays a major role in determining the degree of the equation. We know that $\alpha_0 \cdot (1 \oplus s_{0,23} \oplus s_{16,23})$ has the monomials of maximum degree 14. The right part equation becomes quadratic by multiplication. Therefore, we can obtain a multivariate equation of degree 14.

5.3 Applying an algebraic attack

Let's recall the recent algebraic attack introduced in [9]. Let S_t^d denote the monomials of state variables of the degree up to d and V_t^d denote the monomials of output variables of the degree up to d at clock t. Then, the final equation of degree 14 can be described as a following way.

$$S_t^{14} \oplus S_t V_t = 0 \tag{4}$$

If we put all the monomials on the left side which do not include the output variables,

$$S_t^{14} = S_t V_t \quad \text{or} \quad \begin{cases} \text{Left}(S_t) = S_t^{14} \\ \text{Right}(S_t, V_t) = S_t V_t \end{cases}$$

We can see that $L^t(S_0) = S_t$ where S_0 is the initial state of the state variables and L is a connection function which is linear over $GF(2)$. If we collect $N > \sum_i^{14} \binom{544}{i}$ consecutive equations, a linear dependency $\gamma = (\gamma_0, \ldots, \gamma_{N-1})$ for left side equations must exist and

$$\sum_{t=0}^{N-1} \gamma_t \cdot \text{Left}(L^t(S_0)) = 0, \quad \gamma_i \in GF(2)$$

Let's recover γ from the given sequence. (see [9]) We choose a non-zero random key S_0' and compute $2T$ outputs bits c_t of the left side equations.

$$c_t = \text{Left}(L^t(S_0')), \quad \text{for } t = 0, \ldots, 2T-1$$

where $c_t \in GF(2)$. Then we apply the well-known Berlekamp-Massey algorithm to find the smallest connection polynomial that generates the sequence $c = (c_0, \ldots, c_{2T-1})$.

If we find γ successfully, the same linear dependency holds for the right hand side. So,

$$0 = \sum_{t=i}^{N+i-1} \gamma_{t-i} \cdot \text{Right}(L^t(S_0), V_t), \quad i = 0, 1, \ldots \tag{5}$$

We can see that Equation (5) is linear. If we collect and solve these equations for consecutive keystreams, we can recover the initial state bits of the LFSR with small complexity.

5.4 The complexity of the algebraic attack

If we denote T as the number of monomials of degree up to 14 that are chosen from $n = 544$ unknowns, then

$$T = \sum_{i=0}^{14} \binom{544}{i} \cong 2^{91}$$

We see that recovering the linear dependency γ dominates the complexity of computation. It is estimated to take $O(T \log(T) + Tn)$ by using improved versions of the Berlekamp-Massey algorithm. [9, 11] Therefore, our attack is estimated to take around $O(2^{100})$ CPU clocks with around 2^{92} keystream observations.

For memory requirement, we need to store at most T bits of memory for the linear dependency γ. We need also some memory for Equation (5) but it is much smaller than T. Therefore, we expect that our attack needs around 2^{91} bits of memory.

6 Algebraic attack on SOBER-t16 without stuttering

The structure of SOBER-t16 is a very similar to that of SOBER-t32. Major differences from SOBER-t32 are

- operation based on 16-bit word
- the linear recurrence equation
- a S-box with 8-bit input and 16-bit output

For detail description of SOBER-t16, see [16].

We can apply a very similar algebraic attack presented in Section 5 for the unstuttered SOBER-t16. Let's look at Equation (2), which holds in SOBER-t16 as well. If we multiply Equation (2) by $(1 \oplus s_{0,7} \oplus s_{16,7})$, then we get

$$\begin{aligned} &\alpha_0 \cdot (1 \oplus s_{0,7} \oplus s_{16,7}) \\ &= (s_{0,0} \oplus s_{16,0} \oplus s_{1,0} \oplus s_{6,0} \oplus s_{13,0} \oplus v_0 \oplus K_0) \cdot (1 \oplus s_{0,7} \oplus s_{16,7}) \end{aligned} \tag{6}$$

Let's consider the left part of the equation. The bit α_0 plays a major role in determining the degree of the equation. Computer simulation shows that $\alpha_0 \cdot (1 \oplus s_{0,7} \oplus s_{16,7})$ has the monomials of maximum degree 14. The right part equation becomes quadratic by multiplication. Therefore, we can obtain a multivariate equation of degree 14. The remaining process for attack follows Section 5.3.

The complexity If we denote T as the number of monomials of degree up to 14 that are chosen from $n = 272$ unknowns, then

$$T = \sum_{i=0}^{14} \binom{272}{i} \cong 2^{76.5}$$

Therefore, our attack is estimated to take around $O(2^{85})$ CPU clocks with 2^{78} keystream observations. For memory requirement, we expect that our attack needs around $2^{76.5}$ bits of memory.

7 Conclusion

In this paper we present two algebraic attacks on SOBER-t32 without stuttering. For the first attack, we have built multivariate equations of degree 10. The carry at a specific bit position in addition modulo 2^n is regarded as an internal memory state. From some subset of the keystream observation, we can derive sufficient multivariate equations which are utilized in the algebraic attack. By solving these equations, we are able to recover all the initial states of LFSR and constant value K with roughly $2^{196.5}$ CPU clocks and 2^{69} keystream observations. Furthermore, fast algebraic attack with pre-computation allows more reduction of the complexity of our attack.

For the second attack, we derive a multivariate equation of degree 14 over the non-linear filter. The equation is obtained by multiplying the initial equation by a carefully chosen polynomial. Then, a new algebraic attack method is presented to recover the initial state bits of the LFSR. The attack is estimated to take $O(T \log(T) + Tn) \cong O(2^{100})$ CPU clocks with 2^{92} keystream observations.

By the similarity of the structure, we can apply the second algebraic attack to SOBER-t16 without stuttering. Our attack takes around $O(2^{85})$ CPU clocks using 2^{78} keystream observations for unstuttered SOBER-t16.

Acknowledgment We are grateful to Greg Rose, Frederik Armknecht and unknown referees for their very helpful comments.

References

1. Cryptrec. http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html.
2. Nessie : New european schemes for signatures, integrity, and encryption. https://www.cryptonessie.org.
3. Nessie security report. Technical Report V2.0, Feb. 2003.
4. S. Vanstone A. Menezes, P. Oorschot. *Handbook of Applied Cryptography*. CRC Press, fifth edition, October 1996.
5. F. Armknecht. A linearization attack on the bluetooth key stream generator. Cryptology ePrint Archive, Report 2002/191, 2002. http://eprint.iacr.org/.
6. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 / 2003, pages 162 – 175. Springer-Verlag, October 2003.
7. N. Courtois. Higher order correlation attacks, xl algorithm and cryptanalysis of toyocrypt. Cryptology ePrint Archive, Report 2002/087, 2002. http://eprint.iacr.org/.
8. N. Courtois. Algebraic attacks on combiners with memory and several outputs. Cryptology ePrint Archive, Report 2003/125, 2003. http://eprint.iacr.org/.
9. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - CRYPTO 2003*, volume LNCS 2729, pages 176 – 194. Springer-Verlag, October 2003.
10. N. Courtois and W.Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology - EUROCRPYT 2003*, LNCS 2656, pages 345 – 359. Springer-Verlag, January 2003.
11. J. Dornstetter. On the equivalence between belekamp's and euclid's algorithms. *IEEE Trans. on Information Theory*, IT-33(3):428–431, May 1987.

12. P. Ekdahl and T.Johansson. Snow. Primitive submitted to NESSIE, Sep. 2000.
13. P. Ekdahl and T.Johansson. Distinguishing attacks on sober-t16 and t32. In V. Rijmen J. Daemen, editor, *Fast Software Encryption*, volume LNCS 2365, pages 210–224. Springer-Verlag, 2002.
14. P. Ekdahl and T.Johansson. Distinguishing attacks on sober-t16 and t32. In *Proceedings of the Third NESSIE Workshop*, 2002.
15. J. Hastad and M. Naslund. Bmgl: Synchronous key-stream generator with provable security. Primitive submitted to NESSIE, Sep. 2000.
16. P.Hawkes and G.Rose. Primitive specification and supporting documentation for sober-t16 submission to nessie. In *Proceedings of the first NESSIE Workshop*, Belgium, Sep. 2000.
17. P.Hawkes and G.Rose. Primitive specification and supporting documentation for sober-t32 submission to nessie. In *Proceedings of the first NESSIE Workshop*, Belgium, Sep. 2000.
18. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

A. A toy example to build equations by reconstructing block

This appendix illustrates a small example for building an low degree equation by reconstructing the non-linear block of SOBER-t32. Figure 3 represents the structure of new built block. The input and output operate on 4-bit. The most

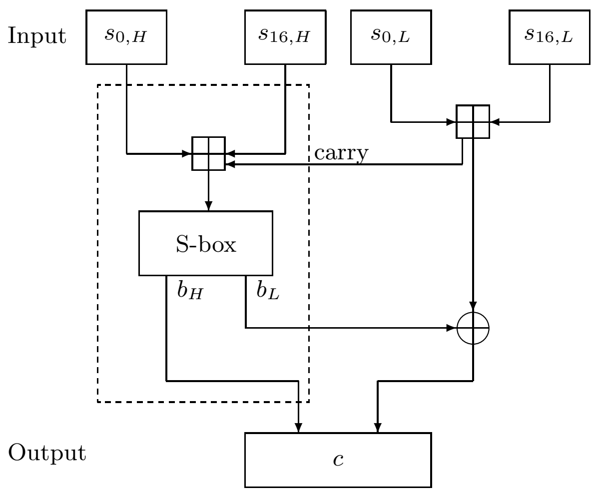

Fig. 3. A reconstructed block for the modular addition and S-box

significant two bits of each input are added modulo 2^2 with carry and become the input of S-box. The S-box is 2×4 substitution defined by the following look-up table : {7,10,9,4}.

Let's denote as following

- $s_{0,H} = \{s_{0,3}, s_{0,2}\}$ and $s_{0,L} = \{s_{0,1}, s_{0,0}\}$
- $s_{16,H} = \{s_{16,3}, s_{16,2}\}$ and $s_{16,L} = \{s_{16,1}, s_{16,0}\}$
- $b_H = \{b_3, b_2\}$ and $b_L = \{b_1, b_0\}$

At first, we construct a matrix containing the input of modular addition and the S-box output.

- Rows are the possibilities for $\{carry, s_{0,3}, s_{0,2}, s_{16,3}, s_{16,2}\}$
- Columns are all monomials of degree to 3 which are coming from $\{s_{0,3}, s_{0,2}, s_{16,3}, s_{16,2}, b_1, b_0\}$.

Applying the Gaussian elimination, we can build at least 10 multivariate equations. The matrix below shows a part of the combination of monomials. Note that the number is represented by hexadecimal format. One of the equations that are built by the Gaussian elimination is

$$s_{0,2}s_{16,2}b_1 \oplus s_{0,2}s_{0,3}s_{16,2} \oplus s_{0,2}s_{16,2}s_{16,3} = 0$$

This equation is verified in the table below. Let us denote the equation as Q. We see that the Q is always zero for all possibilities of input value.

$$
\left[\begin{array}{c|cccccccc}
1 & F & F & F & F & F & F & F & F \\
s_{16,2} & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
s_{16,3} & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
s_{0,2} & 0 & F & 0 & F & 0 & F & 0 & F \\
s_{0,3} & 0 & 0 & F & F & 0 & 0 & F & F \\
b_0 & A & 5 & A & 5 & 5 & A & 5 & A \\
b_1 & C & 9 & 3 & 6 & 9 & 3 & 6 & C \\
\vdots & & & \vdots & & & & & \\
s_{0,2}s_{16,2}b_1 & 0 & 1 & 0 & 4 & 0 & 1 & 0 & 4 \\
\vdots & & & \vdots & & & & & \\
s_{0,2}s_{0,3}s_{16,2} & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 5 \\
\vdots & & & \vdots & & & & & \\
s_{0,2}s_{16,2}s_{16,3} & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\vdots & & & \vdots & & & & &
\end{array}\right]
$$

s_0	s_{16}	$s_0 \boxplus s_{16}$	$(s_0 \boxplus s_{16})_H$	b_1b_0	Q
0	0	0	0	3	0
0	1	1	0	3	0
0	2	2	0	3	0
⋮	⋮	⋮	⋮	⋮	⋮
0	F	F	3	0	0
1	0	1	0	3	0
⋮	⋮	⋮	⋮	⋮	⋮
F	F	E	3	0	0

B. Algebraic equations of S-box in SOBER-t32 and SOBER-t16

For the S-box of SOBER-t32, we denote the 8-bit input stream of S-box as $(a_{31}, a_{30}, \ldots, a_{24})$ and the 32-bit output stream as $(b_{31}, b_{30}, \ldots, b_0)$. Each b_i can be also represented by the combination of monomials which are composed of only the input stream. In particular, the least significant output bit, b_0 is described as the combination of monomials of degree up to 6 as follows.

$$
\begin{aligned}
b_0 = {} & 1 \oplus a_{24} \oplus a_{25} \oplus a_{24}a_{25} \oplus \\
& a_{26} \oplus a_{25}a_{26} \oplus a_{25}a_{27} \oplus a_{24}a_{25}a_{27} \oplus a_{26}a_{27} \oplus \\
& a_{24}a_{26}a_{27} \oplus a_{25}a_{26}a_{27} \oplus a_{25}a_{28} \oplus a_{24}a_{25}a_{28} \oplus a_{25}a_{26}a_{28} \oplus \\
& a_{27}a_{28} \oplus a_{24}a_{27}a_{28} \oplus a_{24}a_{25}a_{27}a_{28} \oplus a_{26}a_{27}a_{28} \oplus a_{25}a_{26}a_{27}a_{28} \oplus \\
& a_{29} \oplus a_{24}a_{29} \oplus a_{24}a_{25}a_{29} \oplus a_{26}a_{29} \oplus a_{24}a_{25}a_{26}a_{29} \oplus \\
& a_{27}a_{29} \oplus a_{24}a_{25}a_{27}a_{29} \oplus a_{24}a_{26}a_{27}a_{29} \oplus a_{25}a_{26}a_{27}a_{29} \oplus a_{24}a_{28}a_{29} \oplus \\
& a_{25}a_{28}a_{29} \oplus a_{24}a_{27}a_{28}a_{29} \oplus a_{25}a_{27}a_{28}a_{29} \oplus a_{24}a_{25}a_{27}a_{28}a_{29} \oplus a_{25}a_{26}a_{27}a_{28}a_{29} \oplus \\
& a_{24}a_{25}a_{26}a_{27}a_{28}a_{29} \oplus a_{25}a_{26}a_{30} \oplus a_{24}a_{25}a_{26}a_{30} \oplus a_{24}a_{27}a_{30} \oplus a_{25}a_{27}a_{30} \oplus \\
& a_{26}a_{27}a_{30} \oplus a_{24}a_{26}a_{27}a_{30} \oplus a_{28}a_{30} \oplus a_{24}a_{28}a_{30} \oplus a_{26}a_{28}a_{30} \oplus \\
& a_{24}a_{25}a_{26}a_{28}a_{30} \oplus a_{27}a_{28}a_{30} \oplus a_{24}a_{27}a_{28}a_{30} \oplus a_{25}a_{26}a_{27}a_{28}a_{30} \oplus a_{29}a_{30} \oplus \\
& a_{24}a_{29}a_{30} \oplus a_{25}a_{29}a_{30} \oplus a_{24}a_{25}a_{26}a_{29}a_{30} \oplus a_{27}a_{29}a_{30} \oplus a_{24}a_{27}a_{29}a_{30} \oplus \\
& a_{24}a_{25}a_{27}a_{29}a_{30} \oplus a_{25}a_{26}a_{27}a_{29}a_{30} \oplus a_{24}a_{28}a_{29}a_{30} \oplus a_{25}a_{28}a_{29}a_{30} \oplus a_{24}a_{25}a_{28}a_{29}a_{30} \oplus \\
& a_{26}a_{28}a_{29}a_{30} \oplus a_{25}a_{26}a_{28}a_{29}a_{30} \oplus a_{27}a_{28}a_{29}a_{30} \oplus a_{24}a_{27}a_{28}a_{29}a_{30} \oplus a_{25}a_{27}a_{28}a_{29}a_{30} \oplus \\
& a_{24}a_{25}a_{27}a_{28}a_{29}a_{30} \oplus a_{26}a_{27}a_{28}a_{29}a_{30} \oplus a_{24}a_{26}a_{27}a_{28}a_{29}a_{30} \oplus a_{25}a_{26}a_{27}a_{28}a_{29}a_{30} \oplus a_{31} \oplus \\
& a_{26}a_{31} \oplus a_{25}a_{26}a_{31} \oplus a_{24}a_{25}a_{26}a_{31} \oplus a_{24}a_{27}a_{31} \oplus a_{25}a_{26}a_{27}a_{31} \oplus \\
& a_{24}a_{25}a_{26}a_{27}a_{31} \oplus a_{25}a_{28}a_{31} \oplus a_{24}a_{25}a_{28}a_{31} \oplus a_{26}a_{28}a_{31} \oplus a_{24}a_{26}a_{28}a_{31} \oplus \\
& a_{24}a_{25}a_{26}a_{28}a_{31} \oplus a_{24}a_{27}a_{28}a_{31} \oplus a_{25}a_{27}a_{28}a_{31} \oplus a_{24}a_{25}a_{27}a_{28}a_{31} \oplus a_{24}a_{25}a_{26}a_{27}a_{28}a_{31} \oplus \\
& a_{25}a_{29}a_{31} \oplus a_{24}a_{25}a_{29}a_{31} \oplus a_{24}a_{26}a_{29}a_{31} \oplus a_{25}a_{26}a_{29}a_{31} \oplus a_{24}a_{25}a_{26}a_{29}a_{31} \oplus \\
& a_{27}a_{29}a_{31} \oplus a_{24}a_{27}a_{29}a_{31} \oplus a_{25}a_{27}a_{29}a_{31} \oplus a_{24}a_{25}a_{27}a_{29}a_{31} \oplus a_{26}a_{27}a_{29}a_{31} \oplus \\
& a_{25}a_{26}a_{27}a_{29}a_{31} \oplus a_{28}a_{29}a_{31} \oplus a_{24}a_{28}a_{29}a_{31} \oplus a_{25}a_{28}a_{29}a_{31} \oplus a_{26}a_{28}a_{29}a_{31} \oplus \\
& a_{25}a_{27}a_{28}a_{29}a_{31} \oplus a_{26}a_{27}a_{28}a_{29}a_{31} \oplus a_{25}a_{26}a_{27}a_{28}a_{29}a_{31} \oplus a_{30}a_{31} \oplus a_{24}a_{30}a_{31} \oplus \\
& a_{24}a_{25}a_{30}a_{31} \oplus a_{26}a_{30}a_{31} \oplus a_{24}a_{26}a_{30}a_{31} \oplus a_{24}a_{25}a_{26}a_{30}a_{31} \oplus a_{24}a_{27}a_{30}a_{31} \oplus \\
& a_{25}a_{26}a_{27}a_{30}a_{31} \oplus a_{24}a_{28}a_{30}a_{31} \oplus a_{24}a_{25}a_{28}a_{30}a_{31} \oplus a_{25}a_{26}a_{28}a_{30}a_{31} \oplus a_{27}a_{28}a_{30}a_{31} \oplus \\
& a_{25}a_{27}a_{28}a_{30}a_{31} \oplus a_{24}a_{25}a_{27}a_{28}a_{30}a_{31} \oplus a_{24}a_{26}a_{27}a_{28}a_{30}a_{31} \oplus a_{29}a_{30}a_{31} \oplus a_{25}a_{26}a_{29}a_{30}a_{31} \oplus \\
& a_{27}a_{29}a_{30}a_{31} \oplus a_{24}a_{27}a_{29}a_{30}a_{31} \oplus a_{25}a_{27}a_{29}a_{30}a_{31} \oplus a_{26}a_{27}a_{29}a_{30}a_{31} \oplus a_{28}a_{29}a_{30}a_{31} \oplus \\
& a_{24}a_{28}a_{29}a_{30}a_{31} \oplus a_{24}a_{25}a_{28}a_{29}a_{30}a_{31} \oplus a_{26}a_{28}a_{29}a_{30}a_{31} \oplus a_{25}a_{27}a_{28}a_{29}a_{30}a_{31}
\end{aligned}
$$

For S-box of SOBER-t16, we denote the 8-bit input stream of S-box as $(a_{15}, a_{14}, \ldots, a_8)$ and the 16-bit output stream as $(b_{15}, b_{14}, \ldots, b_0)$. Then, the least significant output bit, b_0 is described as the combination of monomials of degree up to 6 as follows.

$$
\begin{aligned}
b_0 = 1 &\oplus a_8 \oplus a_9 \oplus a_8a_9 \oplus \\
&a_{10} \oplus a_8a_{10} \oplus a_8a_9a_{10} \oplus a_{11} \oplus a_9a_{11} \oplus \\
&a_8a_9a_{11} \oplus a_8a_{10}a_{11} \oplus a_8a_9a_{12} \oplus a_8a_{10}a_{12} \oplus a_{11}a_{12} \oplus \\
&a_9a_{11}a_{12} \oplus a_8a_9a_{11}a_{12} \oplus a_9a_{10}a_{11}a_{12} \oplus a_9a_{10}a_{13} \oplus a_{11}a_{13} \oplus \\
&a_8a_{11}a_{13} \oplus a_9a_{11}a_{13} \oplus a_8a_9a_{11}a_{13} \oplus a_{10}a_{11}a_{13} \oplus a_9a_{10}a_{11}a_{13} \oplus \\
&a_{12}a_{13} \oplus a_9a_{12}a_{13} \oplus a_8a_9a_{12}a_{13} \oplus a_9a_{10}a_{12}a_{13} \oplus a_{11}a_{12}a_{13} \oplus \\
&a_8a_{11}a_{12}a_{13} \oplus a_9a_{11}a_{12}a_{13} \oplus a_{10}a_{11}a_{12}a_{13} \oplus a_8a_{10}a_{11}a_{12}a_{13} \oplus a_9a_{10}a_{11}a_{12}a_{13} \oplus \\
&a_9a_{14} \oplus a_{10}a_{14} \oplus a_8a_{10}a_{14} \oplus a_9a_{10}a_{14} \oplus a_8a_9a_{10}a_{14} \oplus \\
&a_{11}a_{14} \oplus a_8a_{11}a_{14} \oplus a_9a_{11}a_{14} \oplus a_8a_9a_{11}a_{14} \oplus a_9a_{10}a_{11}a_{14} \oplus \\
&a_8a_9a_{10}a_{11}a_{14} \oplus a_8a_{12}a_{14} \oplus a_9a_{12}a_{14} \oplus a_8a_9a_{12}a_{14} \oplus a_{10}a_{12}a_{14} \oplus \\
&a_8a_{10}a_{12}a_{14} \oplus a_9a_{10}a_{12}a_{14} \oplus a_8a_9a_{10}a_{12}a_{14} \oplus a_9a_{11}a_{12}a_{14} \oplus a_8a_{10}a_{11}a_{12}a_{14} \oplus \\
&a_8a_9a_{13}a_{14} \oplus a_{10}a_{13}a_{14} \oplus a_8a_{10}a_{13}a_{14} \oplus a_8a_9a_{10}a_{13}a_{14} \oplus a_{11}a_{13}a_{14} \oplus \\
&a_8a_9a_{11}a_{13}a_{14} \oplus a_8a_{10}a_{11}a_{13}a_{14} \oplus a_9a_{10}a_{11}a_{13}a_{14} \oplus a_8a_9a_{10}a_{11}a_{13}a_{14} \oplus a_{12}a_{13}a_{14} \oplus \\
&a_8a_{12}a_{13}a_{14} \oplus a_9a_{12}a_{13}a_{14} \oplus a_8a_9a_{12}a_{13}a_{14} \oplus a_{10}a_{12}a_{13}a_{14} \oplus a_8a_{10}a_{12}a_{13}a_{14} \oplus \\
&a_9a_{10}a_{12}a_{13}a_{14} \oplus a_8a_9a_{10}a_{12}a_{13}a_{14} \oplus a_9a_{11}a_{12}a_{13}a_{14} \oplus a_8a_{10}a_{11}a_{12}a_{13}a_{14} \oplus a_9a_{10}a_{11}a_{12}a_{13}a_{14} \oplus \\
&a_8a_{15} \oplus a_9a_{15} \oplus a_8a_9a_{10}a_{15} \oplus a_8a_{10}a_{11}a_{15} \oplus a_8a_9a_{10}a_{11}a_{15} \oplus \\
&a_8a_9a_{12}a_{15} \oplus a_8a_{10}a_{12}a_{15} \oplus a_9a_{10}a_{12}a_{15} \oplus a_8a_9a_{10}a_{12}a_{15} \oplus a_{11}a_{12}a_{15} \oplus \\
&a_8a_{11}a_{12}a_{15} \oplus a_8a_9a_{11}a_{12}a_{15} \oplus a_8a_9a_{10}a_{11}a_{12}a_{15} \oplus a_{13}a_{15} \oplus a_8a_9a_{13}a_{15} \oplus \\
&a_8a_{10}a_{13}a_{15} \oplus a_{11}a_{13}a_{15} \oplus a_{10}a_{11}a_{13}a_{15} \oplus a_9a_{10}a_{11}a_{13}a_{15} \oplus a_{12}a_{13}a_{15} \oplus \\
&a_9a_{12}a_{13}a_{15} \oplus a_8a_9a_{12}a_{13}a_{15} \oplus a_{10}a_{12}a_{13}a_{15} \oplus a_8a_{10}a_{12}a_{13}a_{15} \oplus a_9a_{10}a_{12}a_{13}a_{15} \oplus \\
&a_{10}a_{11}a_{12}a_{13}a_{15} \oplus a_8a_{10}a_{11}a_{12}a_{13}a_{15} \oplus a_9a_{10}a_{11}a_{12}a_{13}a_{15} \oplus a_{14}a_{15} \oplus a_8a_{14}a_{15} \oplus \\
&a_8a_{10}a_{14}a_{15} \oplus a_8a_9a_{10}a_{14}a_{15} \oplus a_{11}a_{14}a_{15} \oplus a_8a_{10}a_{11}a_{14}a_{15} \oplus a_9a_{10}a_{11}a_{14}a_{15} \oplus \\
&a_8a_9a_{10}a_{11}a_{14}a_{15} \oplus a_{12}a_{14}a_{15} \oplus a_9a_{12}a_{14}a_{15} \oplus a_{10}a_{12}a_{14}a_{15} \oplus a_8a_{10}a_{12}a_{14}a_{15} \oplus \\
&a_8a_9a_{10}a_{12}a_{14}a_{15} \oplus a_{11}a_{12}a_{14}a_{15} \oplus a_9a_{11}a_{12}a_{14}a_{15} \oplus a_{10}a_{11}a_{12}a_{14}a_{15} \oplus a_8a_{10}a_{11}a_{12}a_{14}a_{15} \oplus \\
&a_9a_{10}a_{11}a_{12}a_{14}a_{15} \oplus a_{13}a_{14}a_{15} \oplus a_8a_{13}a_{14}a_{15} \oplus a_9a_{10}a_{13}a_{14}a_{15} \oplus a_8a_9a_{10}a_{13}a_{14}a_{15} \oplus \\
&a_{11}a_{13}a_{14}a_{15} \oplus a_9a_{11}a_{13}a_{14}a_{15} \oplus a_8a_{10}a_{11}a_{13}a_{14}a_{15} \oplus a_9a_{10}a_{11}a_{13}a_{14}a_{15} \oplus a_8a_{12}a_{13}a_{14}a_{15} \oplus \\
&a_9a_{12}a_{13}a_{14}a_{15} \oplus a_9a_{10}a_{12}a_{13}a_{14}a_{15} \oplus a_9a_{11}a_{12}a_{13}a_{14}a_{15}
\end{aligned}
$$