Publication VI

Joo Yeon Cho and Miia Hermelin. 2010. Improved linear cryptanalysis of SOSEMANUK. In: Donghoon Lee and Seokhie Hong (editors). Revised Selected Papers of the 12th International Conference on Information Security and Cryptology (ICISC 2009). Seoul, Korea. 2-4 December 2009. Berlin, Heidelberg, Germany. Springer. Lecture Notes in Computer Science, volume 5984, pages 101-117. ISBN 978-3-642-14422-6.

© 2010 Springer Science+Business Media

Reprinted with kind permission from Springer Science and Business Media.

http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/978-3-642-14423-3_8

Improved Linear Cryptanalysis of SOSEMANUK

Joo Yeon Cho and Miia Hermelin

Helsinki University of Technology,
Department of Information and Computer Science,
P.O. Box 5400, FI-02015 TKK, Finland
{joo.cho,miia.hermelin}@tkk.fi

Abstract. The SOSEMANUK stream cipher is one of the finalists of the eSTREAM project. In this paper, we improve the linear cryptanalysis of SOSEMANUK presented in Asiacrypt 2008. We apply the generalized linear masking technique to SOSEMANUK and derive many linear approximations holding with the correlations of up to $2^{-25.5}$. We show that the data complexity of the linear attack on SOSEMANUK can be reduced by a factor of 2^{10} if multiple linear approximations are used. Since SOSEMANUK claims 128-bit security, our attack would not be a real threat on the security of SOSEMANUK.

Keywords: Stream Ciphers, Linear Cryptanalysis, SOSEMANUK, SOBER-128.

1 Introduction

SOSEMANUK [3] is a synchronous software-oriented stream cipher proposed by Berbain et al. in 2005. The SOSEMANUK cipher was submitted to the eSTREAM competition [12] and was selected as one of the four finalists of Profile 1 (software category) in the eSTREAM Portfolio. The eSTREAM project concluded in the final report that SOSEMANUK offers a very considerable margin for security as well as very reasonable performance trade-offs [2].

After the eSTREAM project closed, a linear attack against SOSEMANUK was presented by Lee et al. in Asiacrypt 2008 [10]. In this attack, authors used the linear masking method [7] to derive the best linear approximation of the nonlinear function. Then, they mounted a state recovery attack which was originally developed to cryptanalyze the Grain stream cipher version 0 [4]. The main idea of this attack is to collect a number of linear approximations which depend on partial initial state bits and use them to distinguish the right value of partial initial states from the wrong ones. Authors claimed that the full initial states of SOSEMANUK can be recovered with the time complexity of $2^{147.9}$, the memory complexity of $2^{147.1}$ and the data complexity of $2^{145.5}$.

In this paper, we improve Lee et al.'s linear attack on SOSEMANUK. We derive the best linear approximation of SOSEMANUK by the generalized linear masking method which was applied to the distinguishing attack on SNOW 2.0

D. Lee and S. Hong (Eds.): ICISC 2009, LNCS 5984, pp. 101–116, 2010.
© Springer-Verlag Berlin Heidelberg 2010

by Nyberg et al. [14]. Our results show that the best linear approximation of SOSEMANUK is not a single but multiple. Moreover, many linear approximations have the same order of magnitude of the correlations as the highest one. If Lee et al.'s attack uses such multiple linear approximations holding with strong correlations, the data complexity of the attack can be reduced significantly. On the other hand, the time complexity of the attack is not much affected since the total amount of linear approximations is determined by the correlation of the dominant linear approximations. We estimate that the best attack requires around $2^{135.7}$ keystream bits with the time complexity $2^{147.4}$ and memory complexity $2^{146.8}$.

We note that SOSEMANUK claims the security level of 2^{128} complexity so that our analysis would not threaten the security of SOSEMANUK. Rather, we focus on the security analysis of each component of SOSEMANUK and the effect of their combinations. As a result, we hope to evaluate the security margin of the whole cipher more accurately. We also show that our method can enhance the performance of the distinguishing attack against SOBER-128 which adapts similar nonlinear components to SOSEMANUK.

This paper is organized as follows. In Section 2, the structure of the SOSEMANUK stream cipher is briefly described and the previous linear attacks are discussed. In Section 3, the linear approximations are derived and its capacity is computed. In Section 4, the improved correlation attack against SOSEMANUK is presented. In Section 5, our attack is applied to SOBER-128. Section 6 concludes this paper.

2 Preliminaries

2.1 Brief Description of SOSEMANUK

SOSEMANUK inherits the design structure of the stream cipher SNOW 2.0 [8] which is known for both strong security and high performance. SOSEMANUK aims at improving SNOW 2.0 by reducing the internal state size of the linear feedback shift register (LFSR) for better performance and adding a multiplexing function for avoiding some structural properties. SOSEMANUK also adapts the transformation function from the block cipher SERPENT [1] which was one of the five finalists of AES competition [11]. The structure of SOSEMANUK is shown in Figure 1.

SOSEMANUK uses a single 320-bit (10-word) LFSR which is operated on $\mathbb{F}_{2^{32}}$ with the following recurrence function:

$$s_{t+10} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t, \quad t \geq 1 \tag{1}$$

where α is a root of the primitive polynomial $P(X) = X^4 + \beta^{23}X^3 + \beta^{245}X^2 + \beta^{48}X + \beta^{239}$ on $\mathbb{F}_{2^8}[X]$ and β is a root of the primitive polynomial $Q(X) = X^8 + X^7 + X^5 + X^3 + 1$ on $\mathbb{F}_2[X]$. The nonlinear block of SNOW-like structure is called *the Finite State Machine* (FSM). The FSM of SOSEMANUK contains two 32-bit registers $R1$ and $R2$ with the following relations:

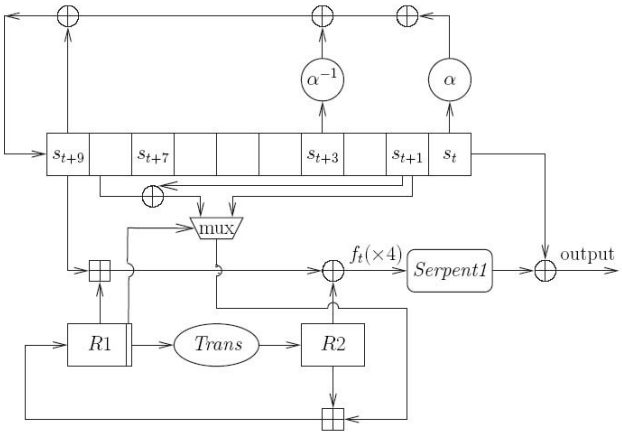

Fig. 1. Overview of SOSEMANUK

$$\begin{aligned} R1_{t+1} &= R2_t \boxplus (r_t s_{t+9} \oplus s_{t+2}) \\ R2_{t+1} &= Trans(R1_t) \\ f_t &= (s_{t+9} \boxplus R1_t) \oplus R2_t \end{aligned} \tag{2}$$

where r_t denotes the least significant bit of $R1_t$. The transition function Trans which is operated on $\mathbb{F}_{2^{32}}$ is defined as

$$\text{Trans}(R1_t) = (R1_t \times \text{0x54655307 mod } 2^{32})_{\lll 7}$$

where $x_{\lll 7}$ denotes x left-rotated by 7 bits and $\times$ denotes an arithmetic multiplication.

Four consecutive outputs of FSM become the input of the transformation function, which is called Serpent1, defined as

$$(z_{t+3}, z_{t+2}, z_{t+1}, z_t) = \text{Serpent1}(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (s_{t+3}, s_{t+2}, s_{t+1}, s_t). \tag{3}$$

Serpent1 takes four 32-bit words as input and provides four 32-bit words as output in bitslice mode. Serpent1 uses an identical 4×4 transformation functions 32 times in parallel, each of which uses 4×4 S-box S_2 which is one of the eight distinct S-boxes used in SERPENT. For complete description of SOSEMANUK we refer to the paper [3].

2.2 Lee et al.'s Attack on SOSEMANUK in Asiacrypt 2008

Let n be a non-negative integer. Given two vectors $x = (x_0, \ldots, x_{n-1})$ and $y = (y_0, \ldots, y_{n-1})$ where $x, y \in \mathbb{F}_2^n$, let $x \cdot y$ denote a standard inner product defined as $x \cdot y = x_0 y_0 \oplus \ldots \oplus x_{n-1} y_{n-1}$. A linear mask is a constant vector that is used to compute an inner product of a n-bit string.

Let $f : \mathbb{F}_{2^n} \mapsto \mathbb{F}_{2^m}$ for some positive integers m and n. The correlation of f is $c(f) = c(f(x)) = 2^{-n}\left(\#\{x : f(x) = 0\} - \#\{x : f(x) = 1\}\right)$. Given a linear input mask $\Lambda \in \mathbb{F}_2^n$ and a linear output mask $\Gamma \in \mathbb{F}_2^m$, the correlation of the linear approximation $\Lambda \cdot x = \Gamma \cdot f(x)$ of f is $c_f(\Lambda; \Gamma) = c(\Lambda \cdot x \oplus \Gamma \cdot f(x))$.

In [10], the best linear approximations of FSM and Serpent1 were derived using a single linear mask Γ as follows:

$$\text{FSM}: \quad \Gamma \cdot f_t \oplus \Gamma \cdot f_{t+1} \oplus \Gamma \cdot s_{t+10} \oplus \Gamma \cdot s_{t+2} = 0 \tag{4}$$

$$\text{Serpent1}: \Gamma \cdot f_t \oplus \Gamma \cdot f_{t+1} \oplus \Gamma \cdot (s_t \oplus z_t) \oplus \Gamma \cdot (s_{t+3} \oplus z_{t+3}) = 0. \tag{5}$$

If (4) and (5) are linearly combined, f_t and f_{t+1} terms are canceled out and the linear approximation of SOSEMANUK is derived as

$$\Gamma \cdot s_{t+10} \oplus \Gamma \cdot s_{t+2} = \Gamma \cdot (s_t \oplus z_t) \oplus \Gamma \cdot (s_{t+3} \oplus z_{t+3}). \tag{6}$$

The highest correlation of (6) holds with the correlation of $2^{-21.4}$ [10]. The correlation attack presented in [10] reduced the data complexity of the attack by the so-called *Second LFSR derivative technique* that was developed by Berbain et al. in [4]. We will discuss this technique in Section 3 . Finally, authors claimed that the attack requires around $2^{145.5}$ data, $2^{147.9}$ computing time and $2^{147.1}$ memory complexity.

3 Deriving Linear Approximations of SOSEMANUK

In this section, we derive the linear approximations of two nonlinear blocks: FSM and Serpent1. By combining them, we derive the linear approximation of SOSEMANUK which uses only the internal states of LFSR and the keystream bits as variables.

3.1 Linear Approximation of FSM

FSM uses the Trans-function and modular additions as the nonlinear components. If the linear masks of each nonlinear component are allowed to be different, a wider range of linear masks search is possible, which enables us to obtain multiple linear approximations with strong correlations. Our idea is depicted in Figure 2.

Firstly, we establish the linear approximations of each nonlinear components as follows:

$$\begin{aligned}
\Gamma_2 \cdot R2_{t+1} &= \Phi \cdot R1_t \\
\Lambda \cdot R1_{t+1} &= \Gamma_1 \cdot R2_t \oplus \Gamma_4 \cdot (s_{t+2} \oplus r_t s_{t+9}) \\
\Gamma_1 \cdot f_t &= \Gamma_3 \cdot s_{t+9} \oplus \Phi \cdot R1_t \oplus \Gamma_1 \cdot R2_t \\
\Gamma_2 \cdot f_{t+1} &= \Gamma_5 \cdot s_{t+10} \oplus \Lambda \cdot R1_{t+1} \oplus \Gamma_2 \cdot R2_{t+1}.
\end{aligned}$$

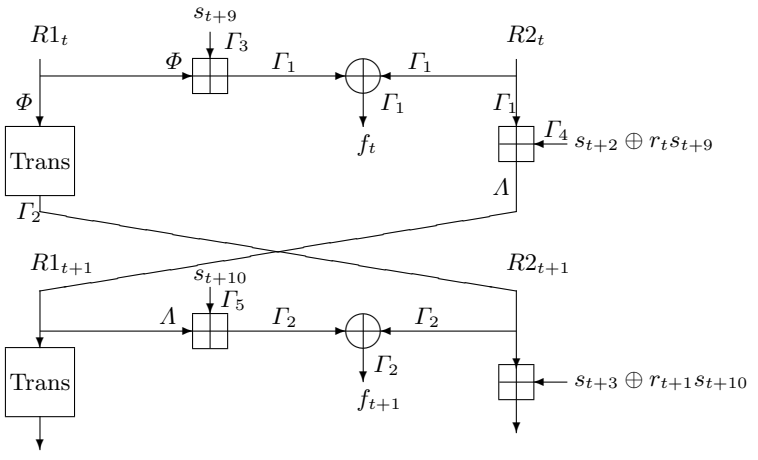

Fig. 2. Generalized linear masking of FSM

where $(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5) \in \mathbb{F}_2^{32}$. If above approximations are linearly combined, the terms of $R1$ and $R2$ registers vanish. Then, we get the following approximation of FSM:

$$\Gamma_1 \cdot f_t \oplus \Gamma_2 \cdot f_{t+1} = \Gamma_3 \cdot s_{t+9} \oplus \Gamma_5 \cdot s_{t+10} \oplus \Gamma_4 \cdot (s_{t+2} \oplus r_t s_{t+9}). \tag{7}$$

Since $r_t \in \{0, 1\}$, we get the following two approximations from (7):

$$(r_t = 0): \quad \Gamma_1 \cdot f_t \oplus \Gamma_2 \cdot f_{t+1} = \Gamma_3 \cdot s_{t+9} \oplus \Gamma_5 \cdot s_{t+10} \oplus \Gamma_4 \cdot s_{t+2} \tag{8}$$

$$(r_t = 1): \quad \Gamma_1 \cdot f_t \oplus \Gamma_2 \cdot f_{t+1} = (\Gamma_3 \oplus \Gamma_4) \cdot s_{t+9} \oplus \Gamma_5 \cdot s_{t+10} \oplus \Gamma_4 \cdot s_{t+2}. \tag{9}$$

Let us denote the correlations of modular addition and the Trans-function by

$$c_+(\Lambda_1, \Lambda_2; \Gamma) = 2\Pr[\Lambda_1 \cdot x \oplus \Lambda_2 \cdot y = \Gamma \cdot (x \boxplus y)] - 1$$
$$c_{\mathrm{Trans}}(\Lambda; \Gamma) = 2\Pr[\Lambda \cdot x = \Gamma \cdot \mathrm{Trans}(x)] - 1.$$

According to Correlation Theorem in [13], the correlations of both (8) and (9) are obtained by computing

$$c_{\mathrm{FSM}}(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5) = \frac{1}{2} \sum_{\Lambda=1}^{2^{32}-1} c_+(\Gamma_1, \Gamma_4; \Lambda) c_+(\Gamma_5, \Lambda; \Gamma_2) \sum_{\Phi=1}^{2^{32}-1} c_+(\Gamma_3, \Phi; \Gamma_1) c_{\mathrm{Trans}}(\Phi; \Gamma_2) \tag{10}$$

where the constant $\frac{1}{2}$ comes from the assumption that $\Pr[r_t = 0] = \Pr[r_t = 1] = \frac{1}{2}$.

3.2 Linear Approximations of Serpent1

At every four clocks, Serpent1 substitutes 128-bit (4-word) inputs into 128-bit (4-word) outputs by 32 parallel S-boxes operated in the bitslice mode. For a

fixed clock t, the inputs and outputs of Serpent1 are $(f_{t+i})_{i=0,1,2,3}$ and $(s_{t+i} \oplus z_{t+i})_{i=0,1,2,3}$, respectively. Hence, the general form of the linear approximation of Serpent1 is

$$\bigoplus_{i=0}^{3} A_i \cdot f_{t+i} = \bigoplus_{i=0}^{3} B_i \cdot (s_{t+i} \oplus z_{t+i}), \quad t \equiv 1 \pmod 4 \tag{11}$$

where $A_i, B_i \in \mathbb{F}_2^{32}$ are the input and output linear masks, respectively.

In bitslice mode, the 4-bit input of the j-th S-box (out of 32 S-boxes) of Serpent1 is the concatenation of each j-th bit of $(f_{t+i})_{i=0,1,2,3}$. Let $a_j, b_j \in \mathbb{F}_{2^4}$ denote the input and output masks of the j-th S-box. The correlation of linear approximation using a_j and b_j is denoted by $c_S(a_j; b_j)$. Then, the correlation of (11) is equal to the multiplication of all the nonzero $c_S(a_j; b_j)$ where $0 \le j \le 31$ as

$$c_{\text{Serpent1}}(A_0, A_1, A_2, A_3, B_0, B_1, B_2, B_3) = \prod_{j \in J} c_S(a_j; b_j) \tag{12}$$

where $J = \{j |\ c_S(a_j; b_j) \neq 0,\ 0 \le j \le 31\}$. Figure 3 shows an example of the linear approximation of Serpent1.

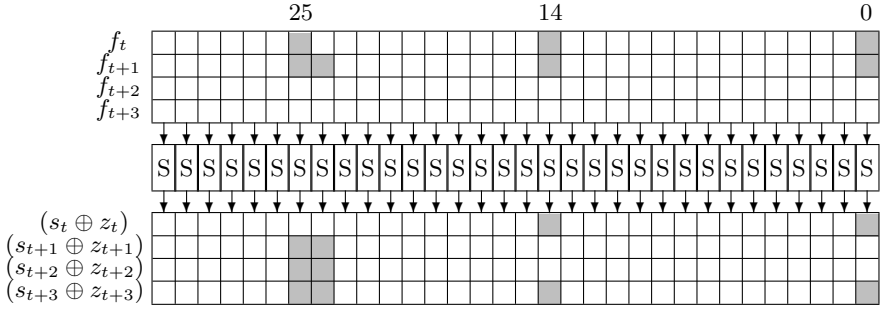

Fig. 3. An example of the linear approximation of Serpent1 with correlation of 2^{-4}

3.3 Approximations of SOSEMANUK

If we combine (7) and (11) in such a way that $(f_{t+i})_{i=0,1,2,3}$ terms vanish, we obtain the linear approximations of SOSEMANUK of which variables come from only the internal states of LFSR and the keystream. Obviously, such combination should satisfy the following condition:

$$(A_0, A_1, A_2, A_3) \in \{(\Gamma_1, \Gamma_2, 0, 0), (0, \Gamma_1, \Gamma_2, 0), (0, 0, \Gamma_1, \Gamma_2)\}.$$

Note that we can obtain (7) at clock $t, t+1$ and $t+2$. Hence, we derive the following form of the linear approximation for $t \equiv 1 \pmod 4$ as

$$\Gamma_3 \cdot s_{t+9+\tau} \oplus \Gamma_4 \cdot s_{t+2+\tau} \oplus \Gamma_5 \cdot s_{t+10+\tau} = \bigoplus_{i=0}^{3} B_i \cdot (s_{t+i} \oplus z_{t+i}), \quad \tau \in \{0, 1, 2\}. \tag{13}$$

Let c_{sose} denote the correlation of (13). Then, $c_{sose} = \sum_{\Gamma_1,\Gamma_2} c_{\mathrm{FSM}} \times c_{\mathrm{Serpent1}}$ and due to the bitslice mode of Serpent1, c_{sose} is equal to $c_{\mathrm{FSM}} \times c_{\mathrm{Serpent1}}$ for some single pair (Γ_1, Γ_2).

Searching the Linear Masks. We searched the linear masks of (13) of which correlations are as strong as possible. Since the search space of the relevant linear masks $(\Gamma_i)_{i=1,\ldots,5}$ and $(B_i)_{j=0,\ldots,3}$ over $\mathbb{F}_2^{32}$ is too large, we allowed the linear masks that are of Hamming weights up to six. The reason for this decision is as follows. Due to the bitslice mode of Serpent1, c_{Serpent1} is determined by the Hamming weight of Γ_1 and Γ_2. Also, the correlation of FSM has three terms of c_+ which is limited by the Hamming weight of Γ_1 and Γ_2. Hence, the c_{sose} using the linear masks of the Hamming weight six is likely to be smaller than $2^{-6\cdot 4} = 2^{-24}$ and it is much smaller than the highest correlation.

For efficient search, the following results help us to reduce the search space. For each $i = 0, \ldots, 31$, we denote the i-th bit of $X \in \mathbb{F}_{2^{32}}$ by X_i. Moreover, the vector of i least significant bits of X is denoted by $X_i' = (X_{i-1}, \ldots, X_0)$. Consider first the modular addition in $\mathbb{F}_{2^{32}}$, denoted by $\boxplus$. Lemma 4 given in [5] is stated as follows:

Lemma 1. *Let $X, Y \in \mathbb{F}_{2^{32}}$ and let $Z = X \boxplus Y$ be their sum modulo 2^{32}. Then $Z_0 = X_0 \oplus Y_0$, $Z_1 = X_1 \oplus Y_1 \oplus X_0 Y_0$ and for all $i = 2, \ldots, 31$, the bit $Z_i = X_i \oplus Y_i \oplus f_i(X_i', Y_i')$, where the function f_i is given by*

$$f_i(X_i', Y_i') = X_{i-1}Y_{i-1} \oplus \bigoplus_{j=0}^{i-2} X_j Y_j \left(\prod_{t=j+1}^{i-1} X_t \oplus Y_t \right).$$

We need the following concepts to formalize the next results. Let $p = \max\{i = 0, \ldots, 32 : X_i \neq 0\}$, that is, p is the largest index such that $X_p \neq 0$ and $X_i = 0$, if $i > p$. Then p is called the *most significant effective bit position* (MSEBP) of X. We denote $p = \mathrm{MSP}(X)$.

Let $X = g(S) \in \mathbb{F}_{2^{32}}$ and $Y = h(S) \in \mathbb{F}_{2^{32}}$ be calculated from the n-bit internal state $S \in \mathbb{F}_{2^n}$ of the cipher using some functions g and h. We say that X and Y are statistically independent, if for all masks $\alpha, \beta \in \mathbb{F}_2^{32}$ and $(\alpha, \beta) \neq (0, 0)$, the correlation $c(\alpha \cdot X \oplus \beta \cdot Y) = c(\alpha \cdot f(S) \oplus \beta \cdot g(S)) = 0$. Hence, if X and Y are statistically independent, each non-trivial linear combination of their bits has zero correlation. We have the following result about the possible input and output masks of the addition of statistically independent inputs:

Lemma 2. *Let $X, Y \in \mathbb{F}_{2^{32}}$ be statistically independent and let $Z = X \boxplus Y$. Let α, β and γ be 32-bit masks of the linear approximation $\alpha \cdot X \oplus \beta \cdot Y \oplus \gamma \cdot Z$ with correlation $c_+(\alpha, \beta; \gamma)$. If the correlation is non-zero, then* $\mathrm{MSP}(\alpha) = \mathrm{MSP}(\beta) = \mathrm{MSP}(\gamma)$.

Proof. Let $p = \mathrm{MSP}(\alpha), q = \mathrm{MSP}(\beta)$ and $r = \mathrm{MSP}(\gamma)$. Using Lemma 1, we have

$$\alpha \cdot X \oplus \beta \cdot Y \oplus \gamma \cdot Z = X_p \oplus Y_q \oplus Z_r \oplus L = X_p \oplus Y_q \oplus X_r \oplus Y_r \oplus f_r(X_r', Y_r') \oplus L,$$

where $L = L(X'_p, Y'_q, Z'_r)$ is a nonlinear function. But since X and Y are statistically independent, the correlation $c_+(\alpha, \beta; \gamma)$ can be non-zero only if $p = q = r$. □

The previous lemma shows how to restrict the search space of the $\boxplus$-operation. We consider the Trans-function next. Let us denote $Z = R \times 0\text{x}54655307 \bmod 2^{32}$. The multiplication by 0x54655307 is equal to the 14-consecutive modular additions as

$$Z = R \boxplus (R \ll 1) \boxplus (R \ll 2) \boxplus (R \ll 8) \boxplus (R \ll 9) \boxplus (R \ll 12) \boxplus (R \ll 14) \boxplus (R \ll 16) \\ \boxplus (R \ll 18) \boxplus (R \ll 21) \boxplus (R \ll 22) \boxplus (R \ll 26) \boxplus (R \ll 28) \boxplus (R \ll 30),$$

where $\ll$ denotes the left-shift operation. Similarly as for $\boxplus$-operation, the bit $Z_i = R_i \oplus g_i(R'_i)$, for all $i = 0, \ldots, 31$. The following corollary shows how the space of possible masks for Trans-function can be restricted.

Corollary 1. *Let $R \in \mathbb{F}_{2^{32}}$ be the input of the* Trans*-function. Let α and β be 32-bit input and output masks of the* Trans*-function, respectively. If the correlation c_{FSM} of the linear approximation of* FSM *is non-zero, then for some $q = 0, \ldots, 6$, we have $q + 25 = \text{MSP}(\alpha) = \text{MSP}(\beta)$. Moreover, $\beta_q = 1$ and $\beta_i = 0$ for all $i = q + 1, \ldots, 6$.*

Proof. Let us first show that $\text{MSP}(\alpha) = \text{MSP}(\beta)$. In all the $\boxplus$-additions in the FSM, the other input consists of some of the statistically independent LFSR state words $s_t, \ldots, s_{t+9}$. Hence, in all three $\boxplus$-additions, the two inputs are statistically independent of each other and by formula (10) and Lemma 2, the MSEBP of all the masks in the triples $(\Gamma_1, \Gamma_4, \Lambda)$, $(\Gamma_5, \Lambda, \Gamma_2)$ and $(\Gamma_3, \Phi, \Gamma_1)$ must be equal. Since $\alpha = \Phi$ and $\beta = \Gamma_2$, we have $\text{MSP}(\alpha) = \text{MSP}(\beta) = p$, for some $p = 0, \ldots, 31$.

Next we show that $p = q + 25$ for some $q = 0, \ldots, 6$ and $\beta_q = 1$. We divide the Trans-function to two steps: $Z = R \times 0x54655307 \bmod 2^{32}$ and $W = Z_{\lll 7}$, such that for each $i = 0, \ldots, 31$, we have $W_i = Z_{(i-7) \bmod 32}$. If the correlation $c_{\text{Trans}}(\alpha; \beta)$ of the approximation is non-zero there must be no statistically independent linear terms in the approximation. Since $\alpha_p = 1$, the term R_p is included in the approximation. For all $i = 0, \ldots, 31$, the bit $Z_i = R_i \oplus g_i(R'_i)$. Hence, at least one of the bits $Z_p, \ldots, Z_{31}$, should be used in the approximation, otherwise R_p would be a statistically independent linear term and the correlation $c_{\text{Trans}}(\alpha; \beta) = 0$. If $p = 31$, then bit $Z_{31} = W_6$ is used in the approximation, we have $\beta_6 = 1$ and the claim holds for $q = 6$.

Assume now $p < 31$. Since p is the MSEBP of α, we must have $\beta_6 = 0$. Otherwise, we would have the bit $Z_{31} = W_6 = R_{31} + g_{31}(R'_{31})$ in the approximation, but R_{31} would then be a lonely, statistically independent linear term giving zero correlation. Similarly we conclude that $\beta_i = 0$, for $i = (p + 8) \bmod 32, \ldots, 6$. Hence, we must have $\beta_{(p+7) \bmod 32} = 1$. Again, since p is MSEBP, we have $p > (p + 7) \bmod 32$. Hence, $p \geq 25$ such that $p = q + 25$, for some $q = 0, \ldots, 6$ and $\beta_{(p+7) \bmod 32} = \beta_q = 1$. □

We note that part of Corollary 1 was heuristically used in [10] with the assumption that $\alpha = \beta$.

Our Results. We used Wallén's algorithm proposed in [15] by which the linear approximations of the modular addition of 2^n could be efficiently delivered. Unfortunately, we could not find the stronger approximation than the one reported in [10]. Instead, we found out there exist many linear approximations that have the same magnitude of correlations as the strongest one. The linear masks of the approximations with strong correlations are partially listed in Table 1. Note that $W_H(X)$ denotes the Hamming weight of X; that is, the number of nonzero bits of $X \in \mathbb{F}_2^{32}$. SOSEMANUK is composed of two nonlinear blocks that operate independently, which intends to remove the possibility of linear approximation that has strong correlation on both blocks simultaneously. On the other hand, the linear approximations of both blocks can be combined independently, which yields multiple linear approximations with equal correlations. Here is an example. Let us take the linear approximation of FSM which is located in the first line in Table 1: $(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5) = (0x02004001, 0x03004001, 0x02004001, 0x02004001, 0x03004001)$ with the correlation of $2^{-17.4}$. The Γ_1 and Γ_2 are transformed into the input masks of Serpent1 that have four nonzero inputs of the S-boxes at the bit positions of $25, 24, 14, 0$, as shown in Figure 3. According to the S-box profile of Serpent1 displayed in Table 5, there exist multiple input and output masks of S-box that yield nonzero correlations. For instance, $c_S(3;9) = c_S(3;14) = 2^{-1}$ and $c_S(2;7) = c_S(2;14) =$

Table 1. Linear masks of FSM with $|c_{\mathrm{FSM}}| \geq 2^{-18.5}$ where $\vee$ denotes a bitwise logical OR operation

Γ_1	Γ_2	Γ_3	Γ_4	Γ_5	c_{FSM}	$W_H(\Gamma_1 \vee \Gamma_2)$
02004001	03004001	02004001	02004001	03004001	$2^{-17.4}$	4
03004001	03004001	03004001	03004001	03004001	$2^{-17.4}$	4
02006001	03004001	02006001	02006001	03004001	$2^{-17.4}$	5
03006001	03004001	03006001	03006001	03004001	$2^{-17.4}$	5
02004001	03004001	02004001	02006001	03006001	$2^{-18.4}$	4
02004001	03004001	02004001	03004001	02004001	$2^{-18.4}$	4
03004001	03004001	03004001	03006001	03006001	$2^{-18.4}$	4
03004001	03004001	03004001	02004001	02004001	$2^{-18.4}$	4
02000201	02000301	03000301	02000201	02000301	$2^{-18.5}$	4
02000301	02000301	03000201	02000301	02000301	$2^{-18.5}$	4

Table 2. Evaluation of the number of linear approximations with respect to the correlations

source	$\lvert c_{sose}\rvert$	M	$M \times c_{sose}^2$
Lee et al.'s attack [10]	$2^{-21.4}$	8	$2^{-39.8}$
this paper	$2^{-21.4}$	896	$2^{-33.0}$
	$2^{-22.5}$	7680	$2^{-32.1}$
	$2^{-23.5}$	63104	$2^{-31.1}$
	$2^{-24.5}$	331776	$2^{-30.7}$
	$2^{-25.5}$	1391872	$2^{-30.6}$

2^{-1}. We obtain more linear approximations by taking (13) at clock $t, t+1$ and $t+2$. Also, both (8) and (9) have equal correlations. In total, there are $896 \approx 2^{9.8}$ linear approximations holding with the correlation of $2^{-21.4}$. Furthermore, we found that a large number of linear approximations have strong correlations slightly less than the strongest one. Table 2 summarizes the number of the linear approximations of (13) that have the correlations of up to $2^{-25.5}$.

4 Linear Cryptanalysis of SOSEMANUK

4.1 Generating Linear Approximations by Linear Recurrence

Given the linear approximation (13), a new linear approximation can be generated by applying the linear recurrence function of the LFSR to (13) at every clock. This technique was described in [4,10] and we give a simpler description using the matrix on this method.

Recall the linear recurrence function of SOSEMANUK. It is well known that the function (1) is equivalently expressed by the following transition matrix:

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_9 \end{pmatrix}$$

where $(a_0, a_3, a_9) = (\alpha, \alpha^{-1}, 1)$ and the other $(a_i)_{i=1,2,4,5,6,7,8}$ are zeros over F_2^{32}. Let us denote the states of the LFSR at the clock t as $S_t = (s_t \ s_{t+1} \ \cdots \ s_{t+9})^T$ where $s_t \in F_2^{32}$ and the superscript T stands for the transpose of the matrix. The state update of LFSR is expressed as $S_{t+1} = AS_t$ for $t \geq 0$. By induction, the current state of LFSR is expressed as $S_t = A^t S_0$ and S_0 is called *the initial states* of the LFSR.

Suppose that $U = (u_0 \ u_1 \ \cdots \ u_9)$ and $W = (w_0 \ w_1 \ w_2 \ w_3)$ denotes linear mask matrices where $u_i, w_i \in F_2^{32}$, respectively. Then, the linear approximation of SOSEMANUK (13) is expressed as the following form:

$$US_t \oplus WZ_t = 0 \Longleftrightarrow UA^t S_0 \oplus WZ_t = 0, \quad t \equiv 1 \pmod 4 \tag{14}$$

where $Z_t = (z_t, z_{t+1}, z_{t+2}, z_{t+3})^T$.

4.2 Attack Method

Our attack algorithm is exactly same as [4,10] except that multiple linear approximations are derived at a fixed clock. Let us assume that N is the number of keystream words observation and M is the linear approximations of the form (14) derived at each clock. Then, we get totally $M \times N$ linear approximations for the attack and they are expressed as the following form:

$$\begin{pmatrix} U_0 \\ U_1 \\ \vdots \\ U_{M-1} \end{pmatrix} A^t S_0 \oplus \begin{pmatrix} W_0 \\ W_1 \\ \vdots \\ W_{M-1} \end{pmatrix} Z_t = 0, \quad t = 1, 2, \ldots, N. \tag{15}$$

Let l denote the length of the internal states of the LFSR over $\mathbb{F}_2$. Our attack aims at recovering m bits out of l state bits where $0 < m < l$. Let Ω_m denote a subspace of $\mathbb{F}_2^l$ such that $l - m$ coordinates at each U in Ω_m are always zeros. Without loss of generality, we assume that the vectors of the Ω_m have zero values from the first to the $(l - m)$-th coordinates. Hence, $|\Omega_m| = 2^m$.

The attack algorithm to recover the m state bits is described as follows;

1. Collect a sufficient number of linear approximations which satisfy $U_i A^t \in \Omega_m$ where $0 \le i \le M$ and $0 \le t \le N$.
2. For $K = 0$ to $K = 2^m - 1$,
   (a) Assign the values of m state bits by K;
   (b) Compute the correlation of the linear approximations using K;
3. Choose K whose correlation is maximal.

In Step 1, the expected number of linear approximations is $M \times N \times 2^{m-l}$. If we combine the $N \times M$ linear approximations pairwise, we can derive new linear approximations (holding with lower correlations) for the attack without increasing the number of the keystream observations. This technique is called *Second LFSR derivation* in [4]. From (15), a pairwise combined linear approximation is of the following form:

$$(U_i A^{\tau_1} \oplus U_j A^{\tau_2}) S_0 \oplus (W_i Z_{\tau_1} \oplus W_j Z_{\tau_2}) = 0, \quad 1 \le i, j \le M,\ 1 \le \tau_1, \tau_2 \le N.$$

The amount of possible combinations are $N \times M \times 2$. Among those, we choose the linear approximations such that $(U_{i_1} A^{j_1} \oplus U_{i_2} A^{j_2}) \in \Omega_m$. Obviously, such approximations have the correlation of c_{sose}^2. The number of approximations that satisfy this condition is expected to be $N' = 2^{m-l}(N \times M)^2$.

Let us denote N' linear approximations by

$$U_i' S_0 \oplus W_i' Z_t = 0, \quad i = 0, \cdots, N' - 1. \tag{16}$$

where $U_i' \in \Omega_m$. In Step 2 and Step 3, the correlations of (16) are evaluated for all possible values of m state bits as follows:

$$\forall K \in \Omega_m, \quad D_K = (\#\{U_i' K \oplus W_i' Z_t = 0'\} - \#\{U_i' K \oplus W_i' Z_t = 1\})/N'.$$

For correctly guessed m state bits, D_K is close to c_{sose}^2. On the other hand, for incorrectly guessed state bits, D_K is close to zero.

Instead of evaluating (16) for all possible values of m state bits independently, we can reduce the computing complexity by the fast Walsh-Hadamard Transform. Let $f : \Omega_m \to \mathbb{R}$ be a real valued function. The Walsh-Hadamard Transform F of f is defined as

$$F(\nu) = \sum_{\eta \in \Omega_m} f(\eta)(-1)^{\eta \cdot \nu}, \quad \nu \in \Omega_m.$$

If the mapping f is defined as the frequencies of the vectors U_i' and W_i' for $i = 0, \cdots, N' - 1$, the fast Walsh-Hadamard Transform $F(K)$ for a fixed K indicates the D_K.

4.3 Attack Complexity

We estimate the complexity of the attack by the statistic method presented in [4,10]. Let l denote the length of the LFSR of SOSEMANUK in bits, i.e. $l = 320$. We target to recover m bits out of l bits by using the linear approximations whose correlations are larger than c_{sose}.

Data Complexity. Let Φ be the normal cumulative distribution function which is defined as

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

For the right value K_0 of the m state bits, the non-detection probability is

$$\Pr\left[D_{K_0} < \frac{3}{2} N' c_{sose}\right] = 1 - \Phi\left(3/\lambda\right)$$

and for the wrong value $K_i \neq K_0$ of the m state bits, the false-alarm probability is

$$\Pr\left[D_{K_i} < \frac{3}{2} N' c_{sose}\right] = 2^{-m}$$

where λ is determined by the condition $1 - \Phi(\lambda) = 2^{-m}$. Then, the number of approximation relations needed for the state recovery attack is $N' = (\frac{4\lambda}{3c_{sose}^2})^2$. Hence, the number of keystream observations N required for the attack is calculated as

$$N' = 2^{m-l-1}(N \times M)^2 = (\frac{4\lambda}{3c_{sose}^2})^2 \Longrightarrow N = \frac{4\lambda 2^{(l-m+1)/2}}{3Mc_{sose}^2}. \tag{17}$$

Since a 128-bit keystream is produced at each observation (every four clocks), the attack requires $128 \times N$ bits of data.

Time Complexity. Suppose that $M \times N$ linear approximations are obtained by observing the keystream and calculating the state recurrence matrix of LFSR. In order to perform the Second LFSR derivative technique, we need $(M \times N)^2$ operations in general. However, the operations can be reduced by applying sorting-and-combining technique used in [4,10]. First, $M \times N$ approximations are sorted out according to the value of $l - m$ state bits. Let the sorted approximations be represented by $X_1, X_2, \ldots, X_{M \times N}$. Then, two consecutive approximations X_i and X_{i+1} for $i = 1, \ldots, M \times N - 1$ are checked whether their $l - m$ state bits are same. If they are same, we know $X_i \oplus X_{i+1} \in \Omega_m$. It is known that the fast sorting algorithm requires around $(M \times N)\log(M \times N)$ operations. higher Let us assume that the N' linear approximations are generated by the Second LFSR derivative technique. As mentioned before, the evaluation of the N' linear approximations can be sped up by the fast Walsh-Hadamard Transform [4,10]. Since the space of the targeted state bits is 2^m, the evaluation by the fast Walsh-Hadamard Transform requires $2^m \log(2^m) = m \times 2^m$ operations. Hence,

the time complexity of the attack for recovering the m state bits is approximately $m \times 2^m + (M \times N)\log(M \times N)$.

Let us assume $m \geq 138$ which is the parameter used in [10]. If the attack is performed on two non-overlapping sets of m state bits, we can recover $2m$ bits out of $320 + 64$ state bits. Then the remaining $384 - 2m$ bits can be searched exhaustively. Therefore, the time complexity required for the recovery of full internal state bits is around $T = 2 \times m \times 2^m + 2 \times M \times N \log(M \times N) + 2^{384-2\times m}$.

Memory Complexity. In order to carry out the sorting-and-combining technique, we need to store $M \times N$ linear approximations, which needs around $l \times M \times N$ memory bits. The Fast Walsh transform needs around $2^m \times \lceil \log N' \rceil$ memory bits. Hence, the memory complexity is around $l \times M \times N + 2^m \times \lceil \log N' \rceil$.

Table 3 summarizes the best attack complexity achievable by using multiple linear approximations against SOSEMANUK.

Table 3. Comparison of the complexity with respect to the number of linear approximations

$\lvert c_{sose}\rvert$	M	m	λ	data (bits)	time	memory (bits)
$2^{-21.4}$	1	138	13.6	$2^{145.5}$	$2^{147.4}$	$2^{146.8}$
$2^{-21.4}$	896	138	13.6	$2^{135.7}$	$2^{147.4}$	$2^{146.8}$
$\geq 2^{-22.5}$	$896 + 7680$	139	13.6	$2^{134.1}$	$2^{148.8}$	$2^{148.5}$
$\geq 2^{-23.5}$	$896 + 7680 + 63104$	140	13.7	$2^{132.6}$	$2^{150.2}$	$2^{150.0}$
$\geq 2^{-24.5}$	$896 + 7680 + 63104 + 331776$	141	13.7	$2^{131.6}$	$2^{151.6}$	$2^{151.5}$
$\geq 2^{-25.5}$	$896 + 7680 + 63104 + 331776 + 1391872$	143	13.8	$2^{130.4}$	$2^{152.9}$	$2^{152.5}$

5 Improved Distinguishing Attack on SOBER-128

SOBER-128 is a software oriented stream cipher proposed in 2003 by Qualcomm Australia [9]. SOBER-128 consists of a 544-bit LFSR and a nonlinear filter (NLF). The length of supporting key size is 128-bit. The brief description of SOBER-128 algorithm is given in Appendix B.

The best attack against SOBER-128 is a distinguishing attack using a linear approximation with the correlation of $2^{-8.8}$ [6]. We discovered that there exist many linear approximations which hold with equal to or slightly less correlations than the highest one. The number of linear approximations with strong correlations is listed in Table 4. If these 96 linear approximations are used for the distinguishing attack, the data complexity of the attack is reduced to

$$N = 1/\sum_{i=1}^{96}(2c_{sober,i}^{-6})^2 = (16 \cdot 2^{-103.6} + 24 \cdot 2^{-104.8} + 56 \cdot 2^{-106})^{-1} = 2^{98.4}.$$

For comparison, the distinguishing attack using a single linear approximation requires $2^{103.6}$ data complexity [6].

Table 4. The number of linear approximations of SOBER-128 and their correlations

source	$\lvert c_{Sober} \rvert$	# of linear approximations
[6]	$2^{-8.8}$	8
this paper	$2^{-8.8}$	16
	$2^{-8.9}$	24
	$2^{-9.0}$	56

6 Conclusion

SOSEMANUK adapts the core structures of two strong ciphering algorithms, aiming at reducing the possibility of attacks which are applicable to both ciphering blocks simultaneously. The existence of many linear approximations holding with strong correlations in both ciphering blocks seems to be an unexpected weakness of SOSEMANUK. We showed that the data complexity of the linear cryptanalysis presented in Asiacrypt 2008 can be reduced by a factor of 2^{10} if such multiple linear approximations are used. Even though we could not present any practical attack threatening the security of SOSEMANUK, we believe that our analysis techniques and results can be useful for analyzing SOSEMANUK-like ciphering algorithms.

Acknowledgment

We are grateful to anonymous reviewers of ICISC'09 for their very valuable comments that helped to improve the paper.

References

1. Anderson, R., Biham, E., Knudsen, L.: Serpent: A proposal for the advanced encryption standard. In: First Advanced Encryption Standard (AES) conference (1998)
2. Babbage, S., Canniere, C.: The eSTREAM portfolio (2008), `http://www.ecrypt.eu.org/stream/portfolio.pdf`
3. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: SOSEMANUK: a fast software-oriented stream cipher, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/027 (2005), `http://www.ecrypt.eu.org/stream/sosemanukp3.html`
4. Berbain, C., Gilbert, H., Maximov, A.: Cryptanalysis of grain. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 15–29. Springer, Heidelberg (2006)
5. Cho, J., Pieprzyk, J.: Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 49–64. Springer, Heidelberg (2004)
6. Cho, J., Pieprzyk, J.: Distinguishing attack on SOBER-128 with linear masking. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 29–39. Springer, Heidelberg (2006)

7. Coppersmith, D., Halevi, S., Jutla, C.: Cryptanalysis of stream ciphers with linear masking. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 515–532. Springer, Heidelberg (2002)
8. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
9. Hawkes, P., Rose, G.: Primitive specification for SOBER-128, Cryptology ePrint Archive, Report 2003/081 (2003), http://eprint.iacr.org/
10. Lee, J., Lee, D., Park, S.: Cryptanalysis of SOSEMANUK and SNOW 2.0 using linear masks. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 524–538. Springer, Heidelberg (2008)
11. NIST, Nist announces encryption standard finalists (1999), http://csrc.nist.gov/archive/aes/round2/r2report.pdf
12. ECRYPT NoE, eSTREAM - the ECRYPT stream cipher project (2005), http://www.ecrypt.eu.org/stream/
13. Nyberg, K.: Correlation theorems in cryptanalysis. Discrete Applied Mathematics 111, 177–188 (2001)
14. Nyberg, K., Wallen, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)
15. Wallén, J.: Linear approximations of addition modulo 2^n. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 261–273. Springer, Heidelberg (2003)

A Correlation Table of S-Box of Serpent1

Given an input mask a and an output mask b where $a, b \in \mathbb{F}_2^4$, the correlation of the linear approximation $a \cdot x \oplus b \cdot S(x) = 0$ of the S-box is measured as follows:

$$c(a;b) = 2^{-4}(\#(a \cdot x \oplus b \cdot S(x) = 0) - \#(a \cdot x \oplus b \cdot S(x) = 1))$$

where the $\cdot$ notation stands for the standard inner product. The correlation table of the S-box is given in Table 5.

B Brief Description of SOBER-128

SOBER-128 consists of an LFSR and a nonlinear filter (NLF). The LFSR consists of 17 words state registers which is denoted by the vector $(s_t, \cdots, s_{t+16})$. Since each s_i is a 32-bit integer, the size of LFSR is 544 bits. The new state of the LFSR is generated by the following connection polynomial

$$s_{t+17} = s_{t+15} \oplus s_{t+4} \oplus \gamma s_t,$$

where the constant γ = 0x00000100 (hexadecimal).

A Nonlinear Filter (NLF) produces an output word z_t by taking s_t, s_{t+1}, s_{t+6}, s_{t+13}, s_{t+16} from the LFSR states and the 32-bit constant K. The NLF consists of two substitution functions (S-box), one rotation, four adders modulo 2^{32} and three XOR additions.

Table 5. Correlation table of S-box used in Serpent1: $c(a;b)$

$a\backslash b$	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0	0	0	0	2^{-1}	0	2^{-1}	0	-2^{-1}	0	2^{-1}	0	0	0	0
2	0	2^{-2}	2^{-2}	2^{-2}	-2^{-2}	0	2^{-1}	0	0	-2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	2^{-1}	0
3	0	2^{-2}	2^{-2}	2^{-2}	2^{-2}	0	0	0	2^{-1}	-2^{-2}	2^{-2}	2^{-2}	-2^{-2}	-2^{-1}	0
4	0	2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	2^{-1}	0	0	0	-2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	0	-2^{-1}
5	0	-2^{-2}	2^{-2}	-2^{-2}	2^{-2}	0	0	-2^{-1}	0	-2^{-2}	-2^{-2}	2^{-2}	2^{-2}	0	-2^{-1}
6	0	0	-2^{-1}	0	2^{-1}	0	0	0	0	0	-2^{-1}	0	-2^{-1}	0	0
7	0	2^{-1}	0	0	0	-2^{-1}	0	-2^{-1}	0	0	0	-2^{-1}	0	0	0
8	0	-2^{-2}	2^{-2}	0	0	2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	-2^{-1}	0	-2^{-2}	-2^{-2}	0	2^{-1}
9	0	-2^{-2}	2^{-2}	2^{-1}	0	-2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	0	0	-2^{-2}	-2^{-2}	0	-2^{-1}
10	2^{-1}	0	0	-2^{-2}	-2^{-2}	-2^{-2}	2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	0	0	-2^{-1}	0
11	-2^{-1}	0	0	2^{-2}	-2^{-2}	2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	0	0	-2^{-1}	0
12	0	0	0	2^{-2}	2^{-2}	2^{-2}	2^{-2}	2^{-2}	2^{-2}	-2^{-2}	-2^{-2}	-2^{-1}	2^{-1}	0	0
13	0	2^{-1}	2^{-1}	-2^{-2}	2^{-2}	2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	2^{-2}	-2^{-2}	0	0	0	0
14	2^{-1}	-2^{-2}	2^{-2}	0	0	2^{-2}	2^{-2}	-2^{-2}	2^{-2}	2^{-1}	0	-2^{-2}	-2^{-2}	0	0
15	2^{-1}	2^{-2}	-2^{-2}	2^{-1}	0	2^{-2}	-2^{-2}	-2^{-2}	-2^{-2}	0	0	2^{-2}	2^{-2}	0	0

The function f is defined as $f(a) = \text{S-box}(a_H) \oplus a$, where the S-box takes 8-bit inputs and generates 32-bit outputs. Note that a_H is the most significant 8 bits of 32-bit word a. The output z_t of the nonlinear filter is described as follows

$$z_t = f((((f(s_t \boxplus s_{t+16}) \ggg 8) \boxplus s_{t+1}) \oplus K) \boxplus s_{t+6}) \boxplus s_{t+13},$$

where $\boxplus$ denotes an addition modulo 2^{32} and $\ggg 8$ denotes the right rotation by 8 bits.. The LFSR states and the constant K are initialized from the 128-bit secret key using the initialization procedure. More details can be found in the original paper describing SOBER-128 [9].

C Example of Linear Masks with the Strongest Correlations

Let us recall (13). If Γ_1 and Γ_2 is used in the bitslice mode and $\tau = 0$, the input of S-box can be 2 or 3. Since $c_S(2;7) = c_S(2;14) = 2^{-1}$ and $c_S(3;9) = c_S(3;14) = 2^{-1}$, there are 32 possible combinations. For $\tau = 1$, the input of S-box can be 4 or 6. Since $c_S(6;3) = c_S(6;5) = c_S(6;11) = c_S(6;13) = 2^{-1}$ and $c_S(4;6) = c_S(4;15) = 2^{-1}$, we get 384 possible combinations. For $\tau = 2$, the input of S-box can be 8 or 12. Since $c_S(8;10) = c_S(8;15) = 2^{-1}$ and $c_S(12;12) = c_S(12;13) = 2^{-1}$, there are 32 possible combinations. If we use both (8) and (9), we can get $2 \times (32 + 384 + 32) = 896$ linear approximations. Table 6 shows some linear masks with the strongest correlations for $\tau = 0$.

Table 6. Linear masks of approximations (13) with the correlation of $2^{-21.4}$ for $\tau = 0$

Γ_1	Γ_2	Γ_3	Γ_4	Γ_5	B_0	B_1	B_2	B_2	$\lvert c_{sose} \rvert$
02004001	03004001	02004001	02004001	03004001	03004001	01000000	01000000	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	01004001	03000000	03000000	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	02004001	01000000	01000000	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	00004001	03000000	03000000	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	03000001	01004000	01004000	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	01000001	03004000	03004000	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	02000001	01004000	01004000	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	00000001	03004000	03004000	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	03004000	01000001	01000001	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	01004000	03000001	03000001	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	02004000	01000001	01000001	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	00004000	03000001	03000001	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	03000000	01004001	01004001	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	01000000	03004001	03004001	02004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	02000000	01004001	01004001	03004001	$2^-21.4$
02004001	03004001	02004001	02004001	03004001	00000000	03004001	03004001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	03004001	00000000	00000000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	01004001	02000000	02000000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	02004001	01000000	01000000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	00004001	03000000	03000000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	03000001	00004000	00004000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	01000001	02004000	02004000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	02000001	01004000	01004000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	00000001	03004000	03004000	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	03004000	00000001	00000001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	01004000	02000001	02000001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	02004000	01000001	01000001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	00004000	03000001	03000001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	03000000	00004001	00004001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	01000000	02004001	02004001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	02000000	01004001	01004001	03004001	$2^-21.4$
03004001	03004001	03004001	03004001	03004001	00000000	03004001	03004001	03004001	$2^-21.4$