A Byte-Based Guess and Determine Attack on SOSEMANUK*

Xiutao Feng, Jun Liu, Zhaocun Zhou, Chuankun Wu, and Dengguo Feng

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing, 100190, China
{fengxt,liuj,zhouzc,ckwu,feng}@is.iscas.ac.cn

Abstract. SOSEMANUK is a software-oriented stream cipher proposed by C. Berbain et al for the eSTREAM project and has been selected into the final portfolio. It is noticed that most components of SOSEMANUK can be calculated byte-oriented. Hence an attacker can observe SOSEMANUK from the view of byte units instead of the original 32-bit word units. Based on the above idea, in this work we present a new byte-based guess and determine attack on SOSEMANUK, where we view a byte as a basic data unit and guess some certain bytes of the internal states instead of the whole 32-bit words during the execution of the attack. Surprisingly, our attack only needs a few words of known key stream to recover all the internal states of SOSEMANUK, and the time complexity can be dramatically reduced to $O(2^{176})$. Since SOSEMANUK has a key with the length varying from 128 to 256 bits, our results show that when the length of an encryption key is larger than 176 bits, our guess and determine attack is more efficient than an exhaustive key search.

Keywords: eSTREAM, SOSEMANUK, Guess and Determine Attack.

1 Introduction

The European eSTREAM project [1] was launched in 2004 to call for stream ciphers and was ended in 2008. At first about 34 stream cipher candidates were submitted to the eSTREAM project, and after the challenge of three rounds, 7 of them were selected into the final portfolio. SOSEMANUK proposed by C. Berbain et al [2] is one of the above seven algorithms. SOSEMANUK is a software-oriented stream cipher and has a key with the length varying from 128 to 256 bits. The design of SOSEMANUK adopted the ideas of both the stream cipher SNOW 2.0 [3] and the block cipher SERPENT [4], and aimed at improving SNOW 2.0 both from the security and from the efficiency points of view.

The guess and determine attack is a common attack on stream ciphers [5,6,7,8]. Its main idea is that: an attacker first guesses the values of a portion of the internal states of the target algorithm, then it takes a little cost to deduce the

* This work was supported by the National Natural Science Foundation (Grant No. 60833008 and 60902024).

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 146–157, 2010.

values of all the rest of the internal states of the algorithm by making use of the values of the guessed portion of internal states and a few known key stream. When the values of all the internal states of the algorithm are recovered, the attacker tests the correctness of these values by producing a key stream using the above recovered values and comparing it with the known key stream. If the key streams agree, it shows that the recovered states are correct. If the key streams don't agree, then the attacker repeats the above process until the correct internal states are found. As for SOSEMANUK, the designers of SOSEMANUK [2] presented a guess and determine attack method, whose time complexity is $O(2^{256})$. In 2006 H. Ahmadi et al [9] revised the attack and reduced the time complexity to $O(2^{226})$, and this result was further reduced to $O(2^{224})$ by Y. Tsunoo et al [10]. Recently Lin and Jie [11] gave a new result that they could recover all internal states of SOSEMANUK with time complexity $O(2^{192})$. Unfortunately, a mistake was made in their work. In step 1 of their attack, $f_{t-2}, f_{t-1}, f_t, f_{t+1}$ and $s_{t-2}, s_{t-1}, s_t, s_{t+1}$ were used to output key words $z_{t-2}, z_{t-1}, z_t, z_{t+1}$, and in step 14, $f_t, f_{t+1}, f_{t+1}, f_{t+2}$ and $s_t, s_{t+1}, s_{t+1}, s_{t+2}$ were used to output key words $z_t, z_{t+1}, z_{t+1}, z_{t+2}$. However, according to the description of SOSEMANUK, the output key words in the next pad should be $z_{t+2}, z_{t+3}, z_{t+4}, z_{t+5}$, which should be produced by $f_{t+2}, f_{t+3}, f_{t+4}, f_{t+5}$ and $s_{t+2}, s_{t+3}, s_{t+4}, s_{t+5}$. Therefore the time complexity they gave is incorrect.

It is known that most word-oriented stream ciphers make a trade-off between security and efficiency. From the view of a designer, for pursuit of more efficient software implementation of the algorithm, some certain operators, for example, the exclusive OR, S-boxes, the modulo 2^n addition, the multiplication or the division by a primitive element in the finite field F_{2^n}, where n may be equal to 8, 16 or 32, are often used. We notice that most of these operations can be done based on the smaller units, for example, 16-bit words or bytes. Therefore from the view of an attacker, he can observe the algorithm from the viewpoints of smaller units instead of the original word units. Based on the above idea, in this work we present a byte-based guess and determine attack on SOSEMANUK, where we view a byte as a basic data unit and guess some certain bytes of the internal states instead of the whole 32-bit words during the execution of the attack. Surprisingly, our attack only needs a few known key stream to recover all the internal states of SOSEMANUK, and the time complexity can be dramatically reduced to $O(2^{176})$. It shows that when the length of an encryption key is larger than 176 bits, the guess and determine attack is more efficient than an exhaustive key search. What's more, our results also show that during the design of stream cipher algorithms, it is necessary to break the bound between different operands.

The rest of this paper is organized as follows: in section 2 we recall the SOSEMANUK algorithm briefly, and in section 3 we give some basic properties of SOSEMANUK. In section 4 we describe all the detail of our attack on SOSEMANUK. In section 5 we give a estimate on the time and data complexity of our attack. Section 6 gives a further discussion, and Section 7 concludes the paper.

2 Description of SOMEMANUK

In this section we recall the SOSEMANUK algorithm briefly and all the details can be found in [2].

SOSEMANUK is a 32-bit word-oriented stream cipher, and logically composed of three parts: a linear feedback shift register (LFSR), a finite state machine (FSM) and a round function Serpent1, see Figure 1.

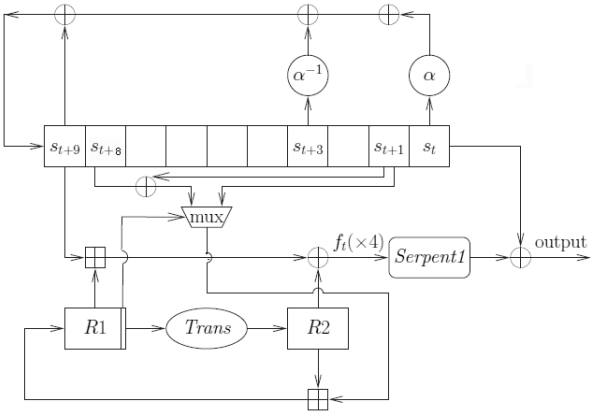

Fig. 1. The structure of SOSEMANUK

2.1 The LFSR

The LFSR of SOSEMANUK is defined over the finite field $F_{2^{32}}$, and contains 10 of 32-bit registers s_i, $1 \leq i \leq 10$. The feedback polynomial $\pi(x)$ of LFSR is defined as follows:

$$\pi(x) = \alpha x^{10} + \alpha^{-1} x^7 + x + 1, \tag{1}$$

where α is a root of the polynomial

$$P(x) = x^4 + \beta^{23} x^3 + \beta^{245} x^2 + \beta^{48} x + \beta^{239}$$

over the finite field F_{2^8}, and β is a root of the binary polynomial

$$Q(x) = x^8 + x^7 + x^5 + x^3 + 1.$$

Let $\{ s_t \}_{t \geq 1}$ be a sequence generated by the LFSR. Then it satisfies

$$s_{t+10} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t, \forall t \geq 1. \tag{2}$$

2.2 The FSM

The nonlinear filtering part of SOSEMANUK is a finite state machine (FSM), which contains two 32-bit memory units $R1$ and $R2$. At time t, the FSM takes

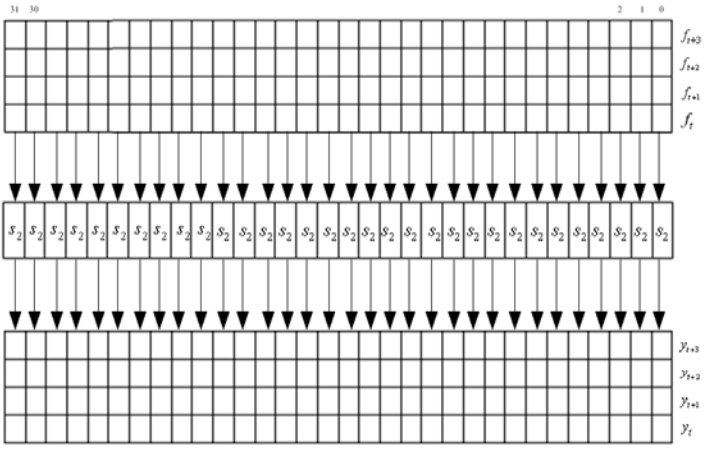

Fig. 2. The round function Serpent1 in the bit-slice mode

the values s_{t+1}, s_{t+8} and s_{t+9} of registers s_1, s_8 and s_9 of the LFSR as inputs, and outputs a 32-bit word f_t. The execution of the FSM is as follows:

$$R1_t = R2_{t-1} \boxplus \mathrm{mux}(\mathrm{lsb}(R1_{t-1}), s_{t+1}, s_{t+1} \oplus s_{t+8}), \tag{3}$$

$$R2_t = \mathrm{Trans}(R1_{t-1}), \tag{4}$$

$$f_t = (s_{t+9} \boxplus R1_t) \oplus R2_t, \tag{5}$$

where $\boxplus$ is the modulo 2^{32} addition; $\mathrm{lsb}(x)$ is the least significant bit of x; $\mathrm{mux}(c, x, y)$ is equal to x if $c = 0$, or equal to y if $c = 1$; and the internal transition function Trans on 32-bit integers is defined by

$$\mathrm{Trans}(z) = (0\mathrm{x}54655307 \cdot z \bmod 2^{32}) \lll 7,$$

where $\lll$ is the left cyclic shift operator on 32-bit strings.

2.3 The Round Function Serpent1

In the block cipher SERPENT a raw SERPENT round consists of, in that order:

- a subkey addition;
- S-boxes transformations;
- a linear transformation.

Here the function Serpent1 is one round of SERPENT without the subkey addition and the linear transformation. The S-box used in Serpent1 is the S-box S_2 of SERPENT and runs in the bit-slice mode. Serpent1 takes outputs f_{t+i} of the FSM at four successive times as inputs and outputs four 32-bit words y_{t+i}, where $i = 0, 1, 2, 3$, see Figure 2.

2.4 Generation of Key Stream

Let $s_t, s_{t+1}, s_{t+2}, s_{t+3}$ and $f_t, f_{t+1}, f_{t+2}, f_{t+3}$ be the outputs of the LFSR and that of the FSM respectively at the successive times starting from time t, and $z_t, z_{t+1}, z_{t+2}, z_{t+3}$ be the key words generated by SOSEMANUK at those four successive times. Then we have

$$(z_{t+3}, z_{t+2}, z_{t+1}, z_t) = \text{Serpent1}(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (s_{t+3}, s_{t+2}, s_{t+1}, s_t). \quad (6)$$

3 Some Properties of SOSEMANUK

In this section we view a byte as a basic data unit and give some basic properties of SOSEMANUK from the view of byte units. First we introduce some notations.

Let x be a 32-bit word. We denote by $x^{(i)}$ the i-th byte component of x, $0 \le i \le 3$, that is,

$$x = x^{(3)} \parallel x^{(2)} \parallel x^{(1)} \parallel x^{(0)},$$

where each $x^{(i)}$ is a byte, and $\parallel$ is the concatenation of two bit strings. For simplicity we write $x^{(1)} \parallel x^{(0)}$ as $x^{(0,1)}$ and $x^{(2)} \parallel x^{(1)} \parallel x^{(0)}$ as $x^{(0,1,2)}$.

For any given 32-bit word x, the word x may have the different meanings in different contexts as follows:

1. As an operand of the operator $\oplus$. Here x is a 32-bit string, and $\oplus$ is the bitwise exclusive OR.
2. As an operand of the integer addition $+$ or the modulo 2^{32} addition $\boxplus$. Here x denotes the integer $\sum_{i=0}^{3} x^{(i)}(2^8)^i$.
3. As an element of the finite field $F_{2^{32}}$. Here x denotes the element $x^{(3)}\alpha^3 + x^{(2)}\alpha^2 + x^{(1)}\alpha + x^{(0)}$ in $F_{2^{32}}$, where α is defined as in equation (1).

Now we consider the SOSEMANUK algorithm from the view of byte units. First we notice that the feedback calculation of the LFSR (see equation (2)) can be represented in the byte form.

Lemma 1. *Equation (2) can be written in the byte form as follows:*

$$s_{t+10}^{(0)} = s_{t+9}^{(0)} \oplus s_{t+3}^{(1)} \oplus \beta^{64} s_{t+3}^{(0)} \oplus \beta^{239} s_t^{(3)}, \quad (2a)$$

$$s_{t+10}^{(1)} = s_{t+9}^{(1)} \oplus s_{t+3}^{(2)} \oplus \beta^{6} s_{t+3}^{(0)} \oplus \beta^{48} s_t^{(3)} \oplus s_t^{(0)}, \quad (2b)$$

$$s_{t+10}^{(2)} = s_{t+9}^{(2)} \oplus s_{t+3}^{(3)} \oplus \beta^{39} s_{t+3}^{(0)} \oplus \beta^{245} s_t^{(3)} \oplus s_t^{(1)}, \quad (2c)$$

$$s_{t+10}^{(3)} = s_{t+9}^{(3)} \oplus \beta^{16} s_{t+3}^{(0)} \oplus \beta^{23} s_t^{(3)} \oplus s_t^{(2)}. \quad (2d)$$

Proof. By the definition of α, we have,

$$\alpha^4 + \beta^{23}\alpha^3 + \beta^{245}\alpha^2 + \beta^{48}\alpha + \beta^{239} = 0.$$

It follows that

$$\alpha^{-1} = \beta^{16}\alpha^3 + \beta^{39}\alpha^2 + \beta^{6}\alpha + \beta^{64}.$$

Let $s_t = \sum_{i=0}^{3} s_t^{(i)} \alpha^i$ and $s_{t+3} = \sum_{i=0}^{3} s_{t+3}^{(i)} \alpha^i$. Then we have

$$\begin{aligned}\alpha s_t &= s_t^{(3)}\alpha^4 + s_t^{(2)}\alpha^3 + s_t^{(1)}\alpha^2 + s_t^{(0)}\alpha \\ &= s_t^{(3)}(\beta^{23}\alpha^3 + \beta^{245}\alpha^2 + \beta^{48}\alpha + \beta^{239}) + s_t^{(2)}\alpha^3 + s_t^{(1)}\alpha^2 + s_t^{(0)}\alpha \\ &= (\beta^{23}s_t^{(3)} + s_t^{(2)})\alpha^3 + (\beta^{245}s_t^{(3)} + s_t^{(1)})\alpha^2 + (\beta^{48}x^{(3)} + s_t^{(0)})\alpha + \beta^{239}s_t^{(3)}\end{aligned}$$

and

$$\begin{aligned}\alpha^{-1}s_{t+3} &= s_{t+3}^{(3)}\alpha^2 + s_{t+3}^{(2)}\alpha^1 + s_{t+3}^{(1)} + s_{t+3}^{(0)}\alpha^{-1} \\ &= s_{t+3}^{(3)}\alpha^2 + s_{t+3}^{(2)}\alpha^1 + s_{t+3}^{(1)} + x^{(0)}(\beta^{16}\alpha^3 + \beta^{39}\alpha^2 + \beta^6\alpha + \beta^{64}) \\ &= \beta^{16}s_{t+3}^{(0)}\alpha^3 + (s_{t+3}^{(3)} + \beta^{39}s_{t+3}^{(0)})\alpha^2 + (s_{t+3}^{(2)} + \beta^6 x^{(0)})\alpha + (s_{t+3}^{(1)} + \beta^{64}s_{t+3}^{(0)}).\end{aligned}$$

Combine the above equations and equation (2), and we immediately get the desired conclusion. ■

Second we observe the update of $R1$ and the output of the FSM and have the following conclusions:

Lemma 2. *Equations (3) and (5) also hold in the sense of modulo 2^k for all $1 \le k < 32$, that is,*

$$R1_t^{[k]} = R2_{t-1}^{[k]} \boxplus \mathrm{mux}(\mathrm{lsb}(R1_{t-1}), s_{t+1}^{[k]}, s_{t+1}^{[k]} \oplus s_{t+8}^{[k]}), \tag{3'}$$

$$f_t^{[k]} = (s_{t+9}^{[k]} \boxplus R1_t^{[k]}) \oplus R2_t^{[k]}, \tag{5'}$$

where $x^{[k]}$ denotes the lowest k bits of x, and the operator $\boxplus$ still denotes the modulo 2^k addition without confusion. In particular, the cases $k = 8, 16$ and 24 are considered in this paper.

Finally we observe the round function Serpent1 and have the following conclusion:

Lemma 3. *For any $1 \le k \le 32$, if the values of the k-th bit of each s_{t+i} ($i = 0, 1, 2, 3$) are known, then the values of the k-th bit of each f_{t+i} can be calculated by the definition of* Serpent1 *given some known key stream, that is,*

$$f'_k = S_2^{-1}(z'_k \oplus s'_k), \tag{7}$$

where

$$\begin{aligned}f'_k &= f_{t+3,k} \parallel f_{t+2,k} \parallel f_{t+1,k} \parallel f_{t,k}, \\ s'_k &= s_{t+3,k} \parallel s_{t+2,k} \parallel s_{t+1,k} \parallel s_{t,k}, \\ z'_k &= z_{t+3,k} \parallel z_{t+2,k} \parallel z_{t+1,k} \parallel z_{t,k},\end{aligned}$$

and $f_{t+i,k}$, $s_{t+i,k}$ and $z_{t+i,k}$ are the k-th bits of f_{t+i}, s_{t+i} and z_{t+i} respectively, $i = 0, 1, 2, 3$. Similarly, if the i-th bytes of each s_{t+i} are known, then we can calculate the i-th bytes of each f_{t+i}, $i = 0, 1, 2, 3$.

4 Execution of the Attack

In this section we always assume that a portion of key stream words $\{ z_t \}$ have been observed, where $t = 1, 2, \cdots, N$, and N is large enough for the attack to work. For convenience, we denote by

$$\mathcal{A} \quad \stackrel{(*)}{\Longrightarrow} \quad \mathcal{B}$$

the deduction of $\mathcal{B}$ from $\mathcal{A}$ by equation (*).

Before the description of the attack, we make the following assumption:

Assumption 1. *The least significant bit of* $R1_1$ *is one, that is,* $\mathrm{lsb}(R1_1) = 1$.

The whole description of the attack on SOSEMANUK can be divided into five phases as follows.

Phase 1. We first guess the total 159-bit values of s_1, s_2, s_3, $s_4^{(0)}$, $R2_1^{(0,1,2)}$ and the rest 31-bit values of $R1_1$.

Step 1.1 We first deduce $s_{10}^{(0)}$, $R1_2^{(0)}$, $R2_2$, $s_{11}^{(0)}$ and $s_4^{(1)}$ as follows:

$$\begin{aligned}
\{ s_1^{(0)}, s_2^{(0)}, s_3^{(0)}, s_4^{(0)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(0)}, f_2^{(0)}, f_3^{(0)}, f_4^{(0)} \}, \\
\{ f_1^{(0)}, R1_1^{(0)}, R2_1^{(0)} \} &\stackrel{(5')}{\Longrightarrow} s_{10}^{(0)}, \\
\{ R1_1^{(0)}, R2_1^{(0)}, s_3^{(0)}, s_{10}^{(0)} \} &\stackrel{(3')}{\Longrightarrow} R1_2^{(0)}, \\
R1_1 &\stackrel{(4)}{\Longrightarrow} R2_2, \\
\{ f_2^{(0)}, R1_2^{(0)}, R2_2^{(0)} \} &\stackrel{(5')}{\Longrightarrow} s_{11}^{(0)}, \\
\{ s_1^{(3)}, s_4^{(0)}, s_{10}^{(0)}, s_{11}^{(0)} \} &\stackrel{(2a)}{\Longrightarrow} s_4^{(1)}.
\end{aligned}$$

Step 1.2 Similar to Step 1.1, we further deduce $s_{10}^{(1)}$, $R1_2^{(1)}$, $s_{11}^{(1)}$ and $s_4^{(2)}$ as follows:

$$\begin{aligned}
\{ s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)} \}, \\
\{ f_1^{(0,1)}, R1_1^{(0,1)}, R2_1^{(0,1)} \} &\stackrel{(5')}{\Longrightarrow} s_{10}^{(0,1)}, \\
\{ R1_1^{(0,1)}, R2_1^{(0,1)}, s_3^{(0,1)}, s_{10}^{(0,1)} \} &\stackrel{(3')}{\Longrightarrow} R1_2^{(0,1)}, \\
\{ f_2^{(0,1)}, R1_2^{(0,1)}, R2_2^{(0,1)} \} &\stackrel{(5')}{\Longrightarrow} s_{11}^{(0,1)}, \\
\{ s_1^{(0)}, s_1^{(3)}, s_4^{(0)}, s_{10}^{(1)}, s_{11}^{(1)} \} &\stackrel{(2b)}{\Longrightarrow} s_4^{(2)}.
\end{aligned}$$

Step 1.3 Similar to Step 1.2, we further deduce $s_{10}^{(2)}$, $R1_2^{(2)}$, $s_{11}^{(2)}$ and $s_4^{(3)}$ as follows:

$$\begin{aligned}
\{ s_1^{(2)}, s_2^{(2)}, s_3^{(2)}, s_4^{(2)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(2)}, f_2^{(2)}, f_3^{(2)}, f_4^{(2)} \}, \\
\{ f_1^{(0,1,2)}, R1_1^{(0,1,2)}, R2_1^{(0,1,2)} \} &\stackrel{(5')}{\Longrightarrow} s_{10}^{(0,1,2)},
\end{aligned}$$

$$\{ R1_1^{(0,1,2)}, R2_1^{(0,1,2)}, s_3^{(0,1,2)}, s_{10}^{(0,1,2)} \} \stackrel{(3')}{\Longrightarrow} R1_2^{(0,1,2)},$$
$$\{ f_2^{(0,1,2)}, R1_2^{(0,1,2)}, R2_2^{(0,1,2)} \} \stackrel{(5')}{\Longrightarrow} s_{11}^{(0,1,2)},$$
$$\{ s_1^{(1)}, s_1^{(3)}, s_4^{(0)}, s_{10}^{(2)}, s_{11}^{(2)} \} \stackrel{(2c)}{\Longrightarrow} s_4^{(3)}.$$

In this phase we have obtained s_1, s_2, s_3, s_4, $s_{10}^{(0,1,2)}$, $s_{11}^{(0,1,2)}$, $R1_1$, $R1_2^{(0,1,2)}$, $R2_1^{(0,1,2)}$ and $R2_2$.

Phase 2. Since we have obtained $s_1^{(3)}$, $s_2^{(3)}$, $s_3^{(3)}$ and $s_4^{(3)}$ in phase 1, thus by equation (7), we can calculate $f_1^{(3)}$, $f_2^{(3)}$, $f_3^{(3)}$ and $f_4^{(3)}$, that is,

$$\{ s_1^{(3)}, s_2^{(3)}, s_3^{(3)}, s_4^{(3)} \} \stackrel{(7)}{\Longrightarrow} \{ f_1^{(3)}, f_2^{(3)}, f_3^{(3)}, f_4^{(3)} \}.$$

Furthermore, by equations $(5')$ and $(2d)$, we have

$$f_1^{(3)} = (s_{10}^{(3)} + R1_1^{(3)} + c_1 \bmod 2^8) \oplus R2_1^{(3)}, \tag{8}$$
$$f_2^{(3)} = (s_{11}^{(3)} + R1_2^{(3)} + c_2 \bmod 2^8) \oplus R2_2^{(3)}, \tag{9}$$
$$s_{11}^{(3)} = s_{10}^{(3)} \oplus \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}. \tag{10}$$

where $c_1 = 1$ if $s_{10}^{(0,1,2)} + R1_1^{(0,1,2)} \geq 2^{24}$, or $c_1 = 0$, otherwise; and $c_2 = 1$ if $s_{11}^{(0,1,2)} + R1_2^{(0,1,2)} \geq 2^{24}$, or $c_2 = 0$, otherwise.

By the assumption $\mathrm{lsb}(R1_1) = 1$, we have $R1_2 = R2_1 \boxplus (S_3 \oplus S_{10})$. It follows that

$$R1_2^{(3)} = R2_1^{(3)} + (s_3^{(3)} \oplus s_{10}^{(3)}) + c_3 \bmod 2^8, \tag{11}$$

where $c_3 = 1$ if $R2_1^{(0,1,2)} + (s_3^{(0,1,2)} \oplus s_{10}^{(0,1,2)}) \geq 2^{24}$, or $c_3 = 0$, otherwise.

Combine equations (8), (9), (10) and (11), and then we have the equation on the variable $s_{10}^{(3)}$:

$$d = (s_{10}^{(3)} \oplus a) + (s_{10}^{(3)} \oplus s_3^{(3)}) + (f_1^{(3)} \oplus (s_{10}^{(3)} + b \bmod 2^8)) + c \bmod 2^8, \tag{12}$$

where $a = \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}$, $b = R1_1^{(3)} + c_1 \bmod 2^8$, $c = c_2 + c_3$ and $d = f_2^{(3)} \oplus R2_2^{(3)}$.

In equation (12), all variables except $s_{10}^{(3)}$ are known, since $s_{10}^{(3)}$ occurs three times in the above equation, it is easy to verify that equation (12) has exactly one solution. Denote its solution by $s_{10}^{(3)}$. When $s_{10}^{(3)}$ has been obtained, we deduce $R2_1^{(3)}$, $s_{11}^{(3)}$ and $R1_2^{(3)}$ by equations (8), (10) and (9) respectively.

Up to now we have obtained s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$.

Phase 3. In this phase we further deduce $R1_3$, $R2_3$, $R1_4$, $R2_4$, $R1_5$, $R2_5$, $R2_6$, s_5, s_6, s_{12} and s_{13} as follows:

$$\begin{aligned}
\{\, R1_2, R2_2, s_4, s_{11} \,\} &\overset{(3)}{\Longrightarrow} R1_3,\\
R1_2 &\overset{(4)}{\Longrightarrow} R2_3,\\
\{\, f_3, R1_3, R2_3 \,\} &\overset{(5)}{\Longrightarrow} s_{12},\\
\{\, s_2, s_{11}, s_{12} \,\} &\overset{(2)}{\Longrightarrow} s_5,\\
\{\, R1_3, R2_3, s_5, s_{12} \,\} &\overset{(3)}{\Longrightarrow} R1_4,\\
R1_3 &\overset{(4)}{\Longrightarrow} R2_4,\\
\{\, f_4, R1_4, R2_4 \,\} &\overset{(5)}{\Longrightarrow} s_{13},\\
\{\, s_3, s_{12}, s_{13} \,\} &\overset{(2)}{\Longrightarrow} s_6,\\
\{\, R1_4, R2_4, s_6, s_{13} \,\} &\overset{(3)}{\Longrightarrow} R1_5,\\
R1_4 &\overset{(4)}{\Longrightarrow} R2_5,\\
R1_5 &\overset{(4)}{\Longrightarrow} R2_6.
\end{aligned}$$

Phase 4. We guess both $s_7^{(0)}$ and $s_8^{(0)}$. The following deductions are entirely similar to phase 1, and we can recover both $s_7^{(1,2,3)}$ and $s_8^{(1,2,3)}$.

$$\begin{aligned}
\{\, s_5^{(0)}, s_6^{(0)}, s_7^{(0)}, s_8^{(0)} \,\} &\overset{(7)}{\Longrightarrow} \{\, f_5^{(0)}, f_6^{(0)}, f_7^{(0)}, f_8^{(0)} \,\},\\
\{\, f_5^{(0)}, R1_5^{(0)}, R2_5^{(0)} \,\} &\overset{(5')}{\Longrightarrow} s_{14}^{(0)},\\
\{\, s_4^{(3)}, s_7^{(0)}, s_{13}^{(0)}, s_{14}^{(0)} \,\} &\overset{(2a)}{\Longrightarrow} s_7^{(1)},\\
\{\, R1_5^{(0)}, R2_5^{(0)}, s_7^{(0)}, s_{14}^{(0)} \,\} &\overset{(3')}{\Longrightarrow} R1_6^{(0)},\\
\{\, f_6^{(0)}, R1_6^{(0)}, R2_6^{(0)} \,\} &\overset{(5')}{\Longrightarrow} s_{15}^{(0)},\\
\{\, s_5^{(3)}, s_8^{(0)}, s_{14}^{(0)}, s_{15}^{(0)} \,\} &\overset{(2a)}{\Longrightarrow} s_8^{(1)},\\
\{\, s_5^{(1)}, s_6^{(1)}, s_7^{(1)}, s_8^{(1)} \,\} &\overset{(7)}{\Longrightarrow} \{\, f_5^{(1)}, f_6^{(1)}, f_7^{(1)}, f_8^{(1)} \,\},\\
\{\, f_5^{(0,1)}, R1_5^{(0,1)}, R2_5^{(0,1)} \,\} &\overset{(5')}{\Longrightarrow} s_{14}^{(0,1)},\\
\{\, s_4^{(0)}, s_4^{(3)}, s_7^{(0)}, s_{13}^{(1)}, s_{14}^{(1)} \,\} &\overset{(2b)}{\Longrightarrow} s_7^{(2)},\\
\{\, R1_5^{(0,1)}, R2_5^{(0,1)}, s_7^{(0,1)}, s_{14}^{(0,1)} \,\} &\overset{(3')}{\Longrightarrow} R1_6^{(0,1)},\\
\{\, f_6^{(0,1)}, R1_6^{(0,1)}, R2_6^{(0,1)} \,\} &\overset{(5')}{\Longrightarrow} s_{15}^{(0,1)},\\
\{\, s_5^{(0)}, s_5^{(3)}, s_8^{(0)}, s_{14}^{(1)}, s_{15}^{(1)} \,\} &\overset{(2b)}{\Longrightarrow} s_8^{(2)},\\
\{\, s_5^{(2)}, s_6^{(2)}, s_7^{(2)}, s_8^{(2)} \,\} &\overset{(7)}{\Longrightarrow} \{\, f_5^{(2)}, f_6^{(2)}, f_7^{(2)}, f_8^{(2)} \,\},\\
\{\, f_5^{(0,1,2)}, R1_5^{(0,1,2)}, R2_5^{(0,1,2)} \,\} &\overset{(5')}{\Longrightarrow} s_{14}^{(0,1,2)},
\end{aligned}$$

$$\{ s_4^{(1)}, s_4^{(3)}, s_7^{(0)}, s_{13}^{(2)}, s_{14}^{(2)} \} \stackrel{(2c)}{\Longrightarrow} s_7^{(3)},$$
$$\{ R1_5^{(0,1,2)}, R2_5^{(0,1,2)}, s_7^{(0,1,2)}, s_{14}^{(0,1,2)} \} \stackrel{(3')}{\Longrightarrow} R1_6^{(0,1,2)},$$
$$\{ f_6^{(0,1,2)}, R1_6^{(0,1,2)}, R2_6^{(0,1,2)} \} \stackrel{(5')}{\Longrightarrow} s_{15}^{(0,1,2)},$$
$$\{ s_5^{(1)}, s_5^{(3)}, s_8^{(0)}, s_{14}^{(2)}, s_{15}^{(2)} \} \stackrel{(2c)}{\Longrightarrow} s_8^{(3)}.$$

Phase 5. Finally we deduce s_9 as follows:

$$\{ s_5, s_6, s_7, s_8 \} \stackrel{(7)}{\Longrightarrow} \{ f_5, f_6, f_7, f_8 \},$$
$$\{ f_5, R1_5, R2_5 \} \stackrel{(5)}{\Longrightarrow} s_{14}^{(3)},$$
$$\{ R1_5, R2_5, s_7, s_{14} \} \stackrel{(3)}{\Longrightarrow} R1_6^{(3)},$$
$$\{ f_6, R1_6, R2_6 \} \stackrel{(5)}{\Longrightarrow} s_{15}^{(3)},$$
$$\{ R1_6, R2_6, s_8, s_{15} \} \stackrel{(3)}{\Longrightarrow} R1_7,$$
$$R1_6 \stackrel{(4)}{\Longrightarrow} R2_7,$$
$$\{ f_7, R1_7, R2_7 \} \stackrel{(5)}{\Longrightarrow} s_{16},$$
$$\{ s_6, s_{15}, s_{16} \} \stackrel{(2)}{\Longrightarrow} s_9.$$

Up to now we have recovered all internal states $s_1, s_2, \cdots, s_{10}$, $R1_1$ and $R2_1$ of the SOSEMANUK algorithm. And then we test the correctness of those values by producing a key stream using the above recovered values and comparing it with the known key stream. If the key streams agree, it shows that the recovered states are correct. If the key streams don't agree, then we will repeat the above process until the correct internal states are found.

The process of the above attack is demonstrated in Table 1.

5 On Time and Data Complexities of Our Attack

The execution of the above attack needs to guess a total of 175 bits of the internal states, including 159 bits of the internal states at phase 1 and 16 bits at phase 4, and then all the rest of the internal states can be deduced under the assumption $\mathrm{lsb}(R1_1) = 1$. Since the probability for $\mathrm{lsb}(R1_1) = 1$ to hold is $\frac{1}{2}$, thus the time complexity of the above attack on SOSEMANUK is $O(2^{176})$. In the attack we only make use of 8 words of the known key stream, and during the verification we need about another 8 words of the known key stream to verify whether the guessed internal states are correct or not. Since by shifting the keystream by 4 words we can test two cases, thus the total data complexity is about 20 words of the known key stream.

6 Further Discussion on the Assumption $\mathrm{lsb}(R1_1) = 1$

In the above attack, we make the assumption $\mathrm{lsb}(R1_1) = 1$, which will guarantee that equation (12) in phase 2 has exactly one solution. However it should be

Table 1. The process of our attack

Assume that $\mathrm{lsb}(R1_1) = 1$				
Steps		Guessed internal states	Determined internal states	Num. of States
Phase 1		$s_1, s_2, s_3, s_4^{(1)}, R1_1^*, R2_1^{(0,1,2)}$		2^{159}
	Step 1.1		$s_{10}^{(0)}, R1_2^{(0)}, R2_2, s_{11}^{(0)}, s_4^{(1)}$	2^{159}
	Step 1.2		$s_{10}^{(1)}, R1_2^{(1)}, s_{11}^{(1)}, s_4^{(2)}$	2^{159}
	Step 1.3		$s_{10}^{(2)}, R1_2^{(2)}, s_{11}^{(2)}, s_4^{(3)}$	2^{159}
Phase 2			$s_{10}^{(3)}, R2_1^{(3)}, R1_2^{(3)}, s_{11}^{(3)}$	2^{159}
Phase 3			$R1_3, R2_3, s_{12}, s_5, R1_4, R2_4$ $s_{13}, s_6, R1_5, R2_5, R2_6$	2^{159}
Phase 4		$s_7^{(0)}, s_8^{(0)}$	$R1_6^{(0,1,2)}, s_{14}^{(0,1,2)}, s_{15}^{(0,1,2)}, s_7, s_8$	2^{175}
Phase 5			$R1_6^{(3)}, s_{14}^{(3)}, s_{15}^{(3)}, R1_7, R2_7, s_{16}, s_9$	2^{175}

Note: $R1_1^*$ denotes the 31 bits of $R1_1$ from the second significant bit to the most significant bit.

pointed out that this assumption is not necessary for our attack to work. In fact, we directly guess the 160-bit values of s_1, s_2, s_3, $s_4^{(0)}$, $R1_1$ and $R2_1^{(0,1,2)}$ in phase 1. When $\mathrm{lsb}(R1_1) = 0$, we have $R1_2 = R2_1 \boxplus s_3$ in phase 2. It follows that

$$R1_2^{(3)} = R2_1^{(3)} + s_3^{(3)} + c_4 \bmod 2^8, \tag{11'}$$

where $c_4 = 1$ if $R2_1^{(0,1,2)} + s_3^{(0,1,2)} \geq 2^{24}$, or $c_4 = 0$, otherwise.

Similar to equation (12), combine equations (8), (9), (10) and (11′), and then we have the equation on the variable $s_{10}^{(3)}$:

$$d' = (s_{10}^{(3)} \oplus a') + (f_1^{(3)} \oplus (s_{10}^{(3)} + b' \bmod 2^8)) + c' \bmod 2^8, \tag{12'}$$

where $a' = \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}$, $b' = R1_1^{(3)} + c_1 \bmod 2^8$, $c' = s_3^{(3)} + c_2 + c_4 \bmod 2^8$ and $d' = f_2^{(3)} \oplus R2_2^{(3)}$. Since $s_{10}^{(3)}$ occurs two times in equation (12′), it is easy to verify that equation (12′) has either no solution, or 2^k solutions for some non-negative integer k. When equation (12′) has no solution, we will come back to phase 1 and repeat guessing new values of those internal states. When equation (12′) has 2^k solutions for some integer k, we write down all solutions, and then for each solution we go on the deductions according to phases 3, 4 and 5. Finally we obtain 2^k different values of the internal states of SOSEMANUK and verify their correctness respectively.

Now we estimate the time and data complexity of the above method. In phase 1 we guess total 160-bit values of the internal states instead of 159-bit values. 2^{159} of those values satisfy $\mathrm{lsb}(R1_1) = 1$ and another 2^{159} values satisfy $\mathrm{lsb}(R1_1) = 0$. As for 2^{159} values satisfying $\mathrm{lsb}(R1_1) = 1$, we have 2^{159} possible values of s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$ after phase 2. As for 2^{159} values satisfying $\mathrm{lsb}(R1_1) = 0$, since equation (12′) has the same number of solutions as that of the possible values of the variables when all the variables except $s_{10}^{(3)}$ go through

all possible values, thus we have also 2^{159} possible values of s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$ after phase 2. Therefore we have total 2^{160} possible values. For each possible values, we go on deducing according to phases 3, 4 and 5, hence the total time complexity is still $O(2^{176})$. But without the assumption, the data complexity reduces to 16 words of the known key stream.

7 Conclusion

In this paper, we presented a byte-based guess and determine attack on SOSEMANUK, which only needs a few words of key stream to recover all internal states of SOSEMANUK with time complexity $O(2^{176})$. The results show that when the length of an encryption key is larger than 176 bits, the guess and determine attack is more efficient than an exhaustive key search.

Acknowledgement

The authors gratefully acknowledge the anonymous referees, whose comments helped to improve the presentation.

References

1. The eSTREAM project, `http://www.ecrypt.eu.org/stream/`
2. 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, the ECRYPT Stream Cipher Project, Report 2005/027 (2005)
3. 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)
4. Anderson, R., Biham, E., Knudsen, L.R.: Serpent: A proposal for the advanced encryption standard, NIST AES Proposal (1998)
5. Bleichenbacher, D., Patel, S.: SOBER cryptanalysis, Fast Software Encryption. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 305–316. Springer, Heidelberg (1999)
6. Hawkes, P., Rose, G.: Exploiting multiplies of the connection polynomial in word-oriented stream ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 302–316. Springer, Heidelberg (2000)
7. Hawkes, P., Rose, G.: Guess and determine attacks on SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 37–46. Springer, Heidelberg (2003)
8. Canniere, C.D.: Guess and determine attacks on SNOW, NESSIE Public Document, NES/DOC/KUL/WP5/011/a (2001)
9. Ahmadi, H., Eghlidos, T., Khazaei, S.: Improved guess and determine Attack on SOSEMANUK, Tehran, Iran (2006), `http://www.ecrypt.eu.org/stream/sosemanukp3.html`
10. Tsunoo, Y., Saito, T., Shigeri, M., Suzaki, T., Ahmadi, H., Eghlidos, T., Khazaei, S.: Evaluation of SOSEMANUK with regard to guess-and-determine attacks (2006), `http://www.ecrypt.eu.org/stream/sosemanukp3.html`
11. Lin, D., Jie, G.: Guess and Determine Attack on SOSEMANUK. In: 2009 Fifth International Conference on Information Assurance and Security, vol. 1, pp. 658–661 (2009)