Guess and Determine Attack on SOSEMANUK

Ding Lin
Department 2 of
Information Science and Technology Institute
Zheng Zhou, China
e-mail: dinglin1987dl@163.com

Guan Jie
Department 2 of
Information Science and Technology Institute
Zheng Zhou, China
e-mail: guanjie007@163.com

Abstract—SOSEMANUK is a new synchronous software-oriented stream cipher with a variable-length key between 128 and 256 bits. In this paper, a Guess and Determine (GD) attack on the cipher is introduced with a computational complexity of \( O(2^{192}) \), requiring only 7 keystream words. The results show that the cipher does not provide full security when the key of the length more than 192 bits is used.

Keywords: SOSEMANUK; Guess and Determine attack; eSTREAM

I. INTRODUCTION

In 2005, Symmetric Techniques Virtual Lab (STVL), a working group for ECRYPT [1] established the ECRYPT Stream Cipher Project (eSTREAM) [2], to call for papers on the new stream ciphers. The eSTREAM portfolio had been revised by September, 2008. Stream ciphers submitted to eSTREAM include SOSEMANUK, which became one of the seven eSTREAM finalists [2].

SOSEMANUK [3], which is proposed by Berbain et al, is a new synchronous software-oriented stream cipher dedicated to software applications. Its key length is variable from 128-bit up to 256-bit. According to the authors SOSEMANUK guarantees up to 128-bit security, regardless of the secret key length. It is designed based on the stream cipher SNOW2.0 [4] and aims at improving SNOW2.0 by avoiding some structural properties which may appear as potential weaknesses and reducing the internal state size.

Guess and Determine attack [5] can be considered as one of general attacks on stream ciphers. According to the evaluation made by designers of SOSEMANUK, Guess and Determine can not be made on the cipher with a computational complexity of \( O(2^{256}) \) or less. However, in [6] a Guess and Determine attack, which can recover all of 384 bits of internal state, was made on the cipher with a computational complexity of \( O(2^{226}) \) by Hadi Ahmadi et al. After that, Yukiyasu Tsunoo et al.[7] introduced a Guess and Determine attack on the cipher with a computational complexity of \( O(2^{224}) \), requiring about \( 2^{12} \) words for this attack. In 2008, Jung-Keun Lee et al.[8] presented a correlation attack on SOSEMANUK with a computational complexity of \( O(2^{47.88}) \), a Memory complexity of \( O(2^{47.10}) \) bits, a data complexity of \( O(2^{145.50}) \) bits and success probability 99% to recover the initial internal state of 384 bits.

In this paper, we introduce a Guess and Determine attack on SOSEMANUK with a computational complexity of \( O(2^{192}) \), which shows that the cipher does not provide full security when the key of the length more than 192 bits is used. The attack requires only 7 keystream words, which is much smaller than the introduced attack in [8].

In section 2 a short description of SOSEMANUK is given. In section 3 we introduce our designed Guess and Determine attack on SOSEMANUK. We give an overall view on the paper along with concluding remarks in section 4.

II. A SHORT DESCRIPTION OF SOSEMANUK

SOSEMANUK is a word-oriented stream cipher with 12-word internal state. Each word consists of 32 bits. The keystream generation of SOSEMANUK can be grouped under roughly 3 parts: Linear Feedback Shift Register (LFSR), Finite State Machine (FSM), and output Transformation. The operators \( \oplus \) and \( + \) stand for bitwise XOR and addition modulo \( 2^{32} \) respectively.

Figure.1 is an overview of SOSEMANUK.
The LFSR consists of ten 32-bit registers and is associated with the feedback polynomial over \( \text{GF}(2^{32}) \) as follows.

\[
\pi(x) = \alpha x^{10} + \alpha^{-1} x^7 + x + 1
\]

Here, \( \alpha \) is a root of primitive Polynomial \( P(X) \) over \( \text{GF}(2^{32}) \).

\[
P(X) = \beta^{-32} x^3 + \beta^{-24} x^2 + \beta^{-16} x + \beta^{-2}
\]

Here, \( \beta \) is a root of primitive polynomial \( Q(X) \) over \( \text{GF}(2^{32}) \).

\[
Q(X) = x^8 + x^7 + x^5 + x^3 + 1
\]

So, the recursive relationship between the LFSR and states is as follows.

\[
S_{t+10} = S_{t+9} + \alpha^{-1} S_{t+3} + \alpha S_t
\]

The FSM is a component with three 32-bit words of input \( (S_{t+1}, S_{t+8}, S_{t+9}) \) from the LFSR and a 32-bit word of output \( (f_t) \) and consists of two 32-bit words of memory, \( R1 \) and \( R2 \). The FSM is denoted by three functions as follows.

\[
f_t = \left( S_{t+9} + R1, \mod 2^{32} \right) \oplus R2
\]

\[
R1_t = (R2_{t-1} \oplus \text{max}(\text{lsb}(R1_{t-1}))),
\]

\[
S_{t+1}, S_{t+1} \oplus S_{t+8}) \mod 2^{32}
\]

\[
R2_t = \text{Trans}(R1_{t-1})
\]

Here, \( \text{lsb}(x) \) means the least significant bit of data \( x \), and \( \text{max}(c, x, y) \) is equal to \( x \) if \( c = 0 \), or to \( y \) if \( c = 1 \). The function \( \text{Trans}(x) \) is defined as follows.

\[
\text{Trans}(x) = (M \times x \mod 2^{32}) \ll 7
\]

Here, constant \( M = 0x54655307 \), \( x \ll 7 \) denotes that 32-bit data \( x \) is 7-bit rotated to the left (towards the most significant bit). Obviously, the function \( \text{Trans}(x) \) is reversible.

Using FSM output \( f_t \) and LFSR register \( S_t \), Output Transformation generates keystreams. The function \( \text{Serpent1} \) is applied to each four-word group. The output of \( \text{Serpent1} \) is combined by XOR with the four corresponding outputs of the LFSR and generates a four-word group of key sequence.

\[
(Z_{t+3}, Z_{t+2}, Z_{t+1}, Z_t) = \text{Serpent1}(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (S_{t+3}, S_{t+2}, S_{t+1}, S_t)
\]

\( \text{Serpent1} \) is the round function of the block cipher SERPENT [9], with neither subkey addition by bitwise exclusive OR nor linear transformation. \( \text{Serpent1} \) uses one S-box of SERPEN, \( S_2 \). The implementation of \( \text{Serpent1} \) is in bit-slice mode that makes the function relate each bit of the output to all four words at the input and vice versa.

III. GUESS AND DETERMINE ATTACK ON SOSEMANUK

In this section we introduce our designed Guess and Determine attack on SOSEMANUK. The GD attack has three “phases” [10]:

- **Phase One**: The attacker guesses the values of \( S_{t+2}, S_{t+1}, S_t, S_{t+1}, R1_{t-1}, R2_{t-1} \) (32-bit each : a total of 192 bits).
- **Phase Two**: The attacker determines the LFSR state \( S_{t}, \cdots, S_{t+9} \) from the values guessed in Phase One.
- **Phase Three**: The attacker tests the correctness of the LFSR states \( S_1, \cdots, S_{t+9} \) and FSM states \( R1_t, R2_t \) by producing a key-stream using these states and comparing it with the observed key-stream. If the stream do not agree, and the attacker can try further guesses (Phase One).

The bulk of the detail of the attack is in Phase Two, which is described as follows.

A. Details of Phase Two

This section concerns the details of Phase Two:

determining all of 384 bits of internal states \( S_1, \cdots, S_{t+9}, R1_t, R2_t \). Phase Two is divided into 27 “Steps”. Using the basis of 6 guessed elements proposed in Phase One, we can determine the other cells as depicted in Table 1.
<table>
<thead>
<tr>
<th>Step NO.</th>
<th>Known Element (s)</th>
<th>Relation</th>
<th>Function/Part</th>
<th>Determine Element (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$S_{r-2}, S_{r-1}, S_r, S_{r+1}$</td>
<td>(9)</td>
<td>Serpent1</td>
<td>$f_{r-2}, f_{r-1}, f_r, f_{r+1}$</td>
</tr>
<tr>
<td>2</td>
<td>$R_{2r+1}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{1r-2}$</td>
</tr>
<tr>
<td>3</td>
<td>$f_{r-1}, R_{1r-1}, R_{2r-1}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+8}$</td>
</tr>
<tr>
<td>4</td>
<td>$S_{r+8}, S_{r+1}, S_r, S_{r-2}$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+7}$</td>
</tr>
<tr>
<td>5</td>
<td>$f_{r-2}, R_{1r-2}, S_{r+7}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$R_{2r-2}$</td>
</tr>
<tr>
<td>6</td>
<td>$R_{1r-1}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{2r+1}$</td>
</tr>
<tr>
<td>7</td>
<td>$R_{2r-1}, S_{r+1}, S_{r+8}, lsb(R_{1r-1})$</td>
<td>(6)</td>
<td>mux</td>
<td>$R_{1r}$</td>
</tr>
<tr>
<td>8</td>
<td>$f_{r}, R_{1r}, R_{2r}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+9}$</td>
</tr>
<tr>
<td>9</td>
<td>$S_{r+9}, S_{r+8}, S_{r-1}$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+2}$</td>
</tr>
<tr>
<td>10</td>
<td>$R_{1r}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{2r+1}$</td>
</tr>
<tr>
<td>11</td>
<td>$R_{2r}, S_{r+2}, S_{r+9}, lsb(R_{1r})$</td>
<td>(6)</td>
<td>mux</td>
<td>$R_{1r-1}$</td>
</tr>
<tr>
<td>12</td>
<td>$f_{r+1}, R_{1r+1}, R_{2r+1}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+10}$</td>
</tr>
<tr>
<td>13</td>
<td>$S_{r+10}, S_{r+9}, S_r$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+3}$</td>
</tr>
<tr>
<td>14</td>
<td>$S_r, S_{r+1}, S_{r+2}, S_{r+3}$</td>
<td>(9)</td>
<td>Serpent1</td>
<td>$f_r, f_{r+1}, f_{r+2}, f_{r+3}$</td>
</tr>
<tr>
<td>15</td>
<td>$R_{1r+1}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{2r+2}$</td>
</tr>
<tr>
<td>16</td>
<td>$R_{2r+1}, S_{r+3}, S_{r+10}, lsb(R_{1r+1})$</td>
<td>(6)</td>
<td>mux</td>
<td>$R_{1r+2}$</td>
</tr>
<tr>
<td>17</td>
<td>$f_{r+2}, R_{1r+2}, R_{2r+2}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+11}$</td>
</tr>
<tr>
<td>18</td>
<td>$S_{r+11}, S_{r+10}, S_{r+1}$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+4}$</td>
</tr>
<tr>
<td>19</td>
<td>$R_{1r+2}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{2r+3}$</td>
</tr>
<tr>
<td>20</td>
<td>$R_{2r+2}, S_{r+4}, S_{r+11}, lsb(R_{1r+2})$</td>
<td>(6)</td>
<td>mux</td>
<td>$R_{1r+3}$</td>
</tr>
<tr>
<td>21</td>
<td>$f_{r+3}, R_{1r+3}, R_{2r+3}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+12}$</td>
</tr>
<tr>
<td>22</td>
<td>$S_{r+12}, S_{r+11}, S_{r+2}$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+5}$</td>
</tr>
<tr>
<td>23</td>
<td>$S_{r+4}, S_{r+2}, S_{r+3}, S_{r+4}$</td>
<td>(9)</td>
<td>Serpent1</td>
<td>$f_{r+1}, f_{r+2}, f_{r+3}, f_{r+4}$</td>
</tr>
<tr>
<td>24</td>
<td>$R_{1r+3}$</td>
<td>(7)</td>
<td>Trans</td>
<td>$R_{2r+4}$</td>
</tr>
<tr>
<td>25</td>
<td>$R_{2r+3}, S_{r+5}, S_{r+12}, lsb(R_{1r+3})$</td>
<td>(6)</td>
<td>mux</td>
<td>$R_{1r+4}$</td>
</tr>
<tr>
<td>26</td>
<td>$f_{r+4}, R_{1r+4}, R_{2r+4}$</td>
<td>(5)</td>
<td>FSM</td>
<td>$S_{r+13}$</td>
</tr>
<tr>
<td>27</td>
<td>$S_{r+13}, S_{r+12}, S_{r+3}$</td>
<td>(4)</td>
<td>LFSR</td>
<td>$S_{r+6}$</td>
</tr>
</tbody>
</table>
Hence, up to Step 27 the internal states $S_1, \ldots, S_{2^9}, R_1, R_2$, will be known, then the attacker can pass to Phase Three to test the correctness of these internal states.

B. The Complexity of the Attack

In this introduced GD attack on SOSEMANUK, we guess the values of six 32-bit elements and implement Step 1 to Step 27 for each of $2^{192}$ possible guessed values. Hence, the computational complexity of our designed attack is $O(2^{192})$, requiring only 7 keystream words ($Z_{i-2}, \ldots, Z_{i+4}$) for the attack. When secret key length is longer than 192-bit, it needs less computational effort than an exhaustive key search, to break SOSEMANUK.

IV. CONCLUSIONS

In this paper, a Guess and Determine attack on the stream cipher SOSEMANUK is introduced. The attack can determine all of 384 bits of internal state, using only 7 keystream words. The computational complexity of the attack is $O(2^{192})$, which shows that the cipher does not provide full security when the key of length more than 192 bits is used. The results also show that the cipher can afford 128-bit security as claimed by its designers.

As a future work, it would be interesting to combine with some general attacks such as algebraic attacks to optimize our attack.

REFERENCES

[1] ECRYPT, the European Network of Excellence for Cryptology. Available at http://www.ecrypt.eu.org/ecrypt1/
[2] eSTREAM, the ECRYPT Stream Cipher Project. Available at http://www.ecrypt.eu.org/stream/