Improved Guess and Determine Attack on SOSEMANUK

Hadi Ahmadi [1], Taraneh Eghlidos [2], Shahram Khazaei [3]

[1] School of Electrical Engineering, Sharif University of Technology, Tehran, Iran.
[2] Electronics Research Center, Sharif University of Technology, Tehran, Iran.
[3] Zaeim Electronic Industries Co., Tehran, Iran.
hadimc@mehr.sharif.edu, teghlidos@sharif.edu,
khazaei@zaeim.com

Abstract. SOSEMANUK is a word-oriented stream cipher submitted to the ECRYPT stream cipher project, with a variable-length key between 128 and 256 bits. The algorithm is similar to the stream cipher SNOW 2.0 except having a smaller LFSR that can lead to higher efficiency. In this paper, we introduce a Guess and Determine (GD) attack on the stream cipher SOSEMANUK with a complexity of $O(2^{226})$. The results show that the cipher has still the 128-bit security claimed by the authors, but does not provide full security when the key of length more than 226 bits is used.

1 Introduction

Word-oriented stream ciphers are a class of stream ciphers in which the operations are done on blocks of w bits long, called word, and over the Galois Field $GF(2^w)$. The most important characteristic of these ciphers is their efficiency because they generate blocks of several bits instead of one bit per clock. Having much higher speed while being much smaller in hardware, word-oriented stream ciphers became of great interests to the industry and many cryptologists.

The NESSIE project [1] started its first announcement in 2000 on cryptographic primitives from different fields of cryptography. The main objective was to collect and introduce strong cryptographic algorithms as standards in different fields of cryptography such as block ciphers, stream ciphers, MAC algorithms, hash functions, public key schemes and so on. During this project, several new stream ciphers were proposed, among them SNOW [2] was accepted for the second phase. Despite introducing new ideas in designing word-oriented stream ciphers none of the accepted proposals were included in the final portfolio due to their weaknesses [1].

SOSEMANUK [3] is a software-oriented stream cipher with a 128- to 256- bit key and a 128-bit Initial Vector (IV). This cipher is designed based on the stream cipher SNOW 2.0 [4]; so, it has a linear part that consists of a 320-bit LFSR and a Finite State Machine (FSM) including two 32-bit registers. SOSEMANUK aims at improving SNOW 2.0 by avoiding some structural properties, in the FSM, that appear as potential weaknesses and also by reducing the internal state size that improves the

efficiency. According to the authors this cipher is claimed to accomplish a 128-bit security [3].

Guess and Determine (GD) attacks are one of the general attacks which have been effective on some stream ciphers. As it comes from the name, in Guess and Determine attacks, we attempt to obtain the states of all cells of the whole cipher system by guessing the contents of some of them initially and comparing the resulting key sequence with the running key sequence. In [5] a systematic way of implementing some GD attacks, called Advanced GD attacks, is introduced. The result of implementing Advanced GD attacks on SNOW 1.0 and SNOW 2.0 show the complexity of $O(2^{205})$ and $O(2^{267})$, respectively, while no former heuristic GD attacks on SNOW 1.0 have resulted in a better complexity and there is no heuristic GD attack introduced on SNOW 2.0 [5].

In Advanced GD attacks [5], we were supposed to use cryptographic functions, extracted from the cipher algorithm, which are deterministic and definitely true for each time, but in the non-linear component of SOSEMANUK, a multiplexer *MUX* and a function *Serpent1* are used that break this assumption. Based on the design method of Advanced GD attacks, we first analyze the stream cipher SOSEMANUK by considering some simplifying assumptions on *MUX* and *Serpent1* which leads to an attack with the complexity of $O(2^{160})$. Next, we modify the attack by taking into account the real *MUX* and *Serpent1* which results in an attack with computational complexity of $O(2^{226})$ on the cipher.

In section 2 a short description of SOSEMANUK is given. Afterwards, we apply an Advanced GD attack on SOSEMANUK in section 3. Considering the multiplexer *MUX* and the function *Serpent1*, we introduce our designed Guess and Determine attack on SOSEMANUK in section 4. Section 5 includes the results along with concluding remarks.

2 A Short Description of SOSEMANUK

SOSEMANUK is a word oriented stream cipher with a 12-word internal state. Each word consists of 32 bits. The LFSR contains 10 elements of $GF(2^{32})$, a finite field which is represented exactly the same as SNOW 2.0, and is associated with the following feedback polynomial:

$$\pi(X) = \alpha X^{10} + \alpha^{-1} X^{7} + X + 1 \quad \textbf{(1)}$$

α is a primitive element in $GF(2^{32})$ [3]. So, the recursive relationship between the LFSR states is as follows:

$$S_{t+10} = S_{t+9} + \alpha^{-1} S_{t+3} + \alpha S_t \quad \textbf{(2)}$$

The FSM is a component with three 32-bit words of input from the LFSR and a 32-bit word of output and consists of two 32-bit words of memory, R1 and R2. The FSM is defined by the three following functions:

$$R1_t = (R2_{t-1} + mux(lsb(R1_{t-1}), S_{t+1}, S_{t+1} \oplus S_{t+8})) \bmod 2^{32} \quad (3)$$

$$R2_t = Trans(R1_{t-1}) \quad (4)$$

$$f_t = (S_{t+9} + R1_t \bmod 2^{32}) \oplus R2_t \quad (5)$$

where *lsb(x)* is the least significant bit of *x*, and *mux(c,x,y)* is equal to *x* if $c = 0$, or to *y* if $c = 1$.The transition function *Trans(z)* operates on GF(2^{32}) and is defined in [3].

The outputs of the FSM are buffered and grouped by four, and *Serpent1* is applied to each four-word group; the output of *Serpent1* is then combined by XOR with the four corresponding outputs of the LFSR and generates a four-word group of key sequence.

$$(Z_{t+3}, Z_{t+2}, Z_{t+1}, Z_t) = Serpent1(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (S_{t+3}, S_{t+2}, S_{t+1}, S_t) \quad (6)$$

The function *Serpent1* uses one S-box of the block cipher SERPENT [6], S_2, 32 copies of which are applied in parallel to make a substitution of a 128-bit input. The implementation of *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. The main structure of SOSEMANUK is shown in Figure 1.

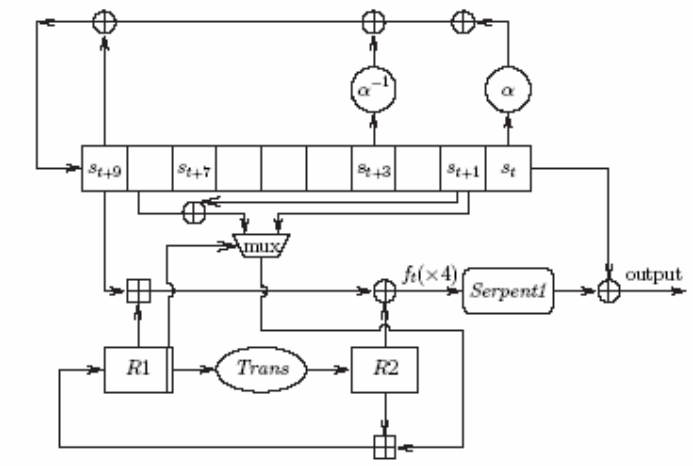

Fig. 1. the block diagram of SOSEMANUK

3 Advanced Guess and Determine Attack on a Simplified Version of SOSEMANUK

In Advanced GD attacks, we exploit the cryptographic functions describing the whole cipher algorithm. For instance, in addition to all of the relationships introduced in section 2, we can also use the square of the feedback polynomial as a useful cryptographic function. Note that the recursive relationship corresponding to the square of the feedback polynomial can also produce the same output sequence as the output of the LFSR. The square polynomial and the related recurrence relation are as follows:

$$\gamma(X) = \alpha^2 X^{20} + \alpha^{-2} X^{14} + X^2 + 1 \quad \textbf{(7)}$$

$$S_{t+20} = S_{t+18} + \alpha^{-2} S_{t+6} + \alpha^2 S_t \quad \textbf{(8)}$$

In implementing Advanced GD attacks, we suppose that the selection bit of *MUX* is always 0. Considering this assumption and the relation (4), we can simplify the relation (3) as below:

$$R1_t = (Trans(R1_{t-2}) + S_{t+1}) \bmod 2^{32} \quad \textbf{(9)}$$

We also assume that Z_{t+i} can be found whenever f_{t+i} and S_{t+i} are known. Precisely, the assumed relations for *Serpent1* are written below:

$$Z_{t+i} = SB_i(f_{t+i}) \oplus S_{t+i};\ \ i \in \{1,2,3,4\},\ t = 4n, n = 0,1,2,... \quad \textbf{(10)}$$

Where $SB_i(.)$, i=1,2,3,4 are virtual 32-bit input/output invertible functions that replaces the function *Serpent1*. So, by replacing the relation (6) by (10), and using (4), we can simplify the relation (5) as follows.

$$SB_i^{-1}(Z_t \oplus S_t) = (S_{t+9} + R1_t \bmod 2^{32}) \oplus Trans(R1_{t-1}) \quad \textbf{(11)}$$

Now, according to [5], we construct four matrices corresponding to the four simplified cryptographic relations (2), (8), (9) and (11).

Each matrix consists of 30 rows and each row stands for the indices of one cryptographic equation at a definite time. In this presentation, we replaced S_{t+i} by i and $R1_{t+i}$ by $i+53$. The constructed matrices are shown in Table 1.

Implementing an Advanced GD attack on SOSEMANUK results in guessing a basis of 5 elements {1, 2, 3, 10, 51} that represents $\{S_{t+1}, S_{t+2}, S_{t+3}, S_{t+10}, R1_{t-2}\}$. So, by guessing 5*32=160 bits of the internal state, one can determine other variables and the resulting output sequence and compare it with the observed running key sequence. Consequently, the complexity of this attack is of $O(2^{160})$ basic operations over $GF(2^{32})$.

Table 1. the corresponding cryptographic matrices in SOSEMANUK

10	9	3	0	20	18	6	0	52	50	0		0	9	53	52
11	10	4	1	21	19	7	1	53	51	1		1	10	54	53
12	11	5	2	22	20	8	2	54	52	2		2	11	55	54
13	12	6	3	23	21	9	3	55	53	3		3	12	56	55
14	13	7	4	24	22	10	4	56	54	4		4	13	57	56
15	14	8	5	25	23	11	5	57	55	5		5	14	58	57
16	15	9	6	26	24	12	6	58	56	6		6	15	59	58
17	16	10	7	27	25	13	7	59	57	7		7	16	60	59
18	17	11	8	28	26	14	8	60	58	8		8	17	61	60
19	18	12	9	29	27	15	9	61	59	9		9	18	62	61
20	19	13	10	30	28	16	10	62	60	10		10	19	63	62
21	20	14	11	31	29	17	11	63	61	11		11	20	64	63
22	21	15	12	32	30	18	12	64	62	12		12	21	65	64
23	22	16	13	33	31	19	13	65	63	13		13	22	66	65
24	23	17	14	34	32	20	14	66	64	14		14	23	67	66
25	24	18	15	35	33	21	15	67	65	15		15	24	68	67
26	25	19	16	36	34	22	16	68	66	16		16	25	69	68
27	26	20	17	37	35	23	17	69	67	17		17	26	70	69
28	27	21	18	38	36	24	18	70	68	18		18	27	71	70
29	28	22	19	39	37	25	19	71	69	19		19	28	72	71
30	29	23	20	40	38	26	20	72	70	20		20	29	73	72
31	30	24	21	41	39	27	21	73	71	21		21	30	74	73
32	31	25	22	42	40	28	22	74	72	22		22	31	75	74
33	32	26	23	43	41	29	23	75	73	23		23	32	76	75
34	33	27	24	44	42	30	24	76	74	24		24	33	77	76
35	34	28	25	45	43	31	25	77	75	25		25	34	78	77
36	35	29	26	46	44	32	26	78	76	26		26	35	79	78
37	36	30	27	47	45	33	27	79	77	27		27	36	80	79
38	37	31	28	48	46	34	28	80	78	28		28	37	81	80
39	38	32	29	49	47	35	29	81	79	29		29	38	82	81
40	39	33	30	50	48	36	30	82	80	30		30	39	83	82
M_1, relation (2)				M_2, relation (8)				M_3, relation (9)				M_4, relation (11)			

4 Guess and Determine Attack Considering *MUX* and *Serpent1*

Using the basis of 5 guessed elements proposed in section 3, we can determine the other cells as depicted in Table 2.

Table 2. Determination phase in Advanced GD attack on SOSEMANUK

Step Number	Known Cells	Matrix	Determined Cell(s)
1	1,51	M_3	53
2	1,10,53	M_4	54
3	3,53	M_3	55
4	2,54,55	M_4	11
5	1,10,11	M_1	4
. . .	. . .	. . .	. . .

The matrix M_3 represents the relation between R1 and the output of *MUX* assuming the selection bit of *MUX* is 0. This assumption hold with the probability ½ each time we use the matrix M_3 in the determination phase. The matrix M_4 is related to the replacement of *Serpent1* by four 32-bit input/output SB_i, i=0,1,2,3. But in the main designed structure of SOSEMANUK, we should know a complete set of 128-bit input to find the output of *Serpent1* and vice versa, due to the bit-slice mode implementation of the S-boxes. The modified GD attack based on the Advanced GD attack, described in section 3, is written as follows:

The basis of initial guessed elements is considered $\{S_{t+1}, S_{t+2}, S_{t+3}, S_{t+4}, S_{t+10}, R2_{t-1}\}$.

Table 3. Determination phase in modified GD attack on SOSEMANUK

Step No.	Known Element(s)	Relation	Function/ Part	Assumption	Determined Element(s)
1	$R2_{t-1}, S_{t+1}$	(3)	*MUX*	$lsb(R1_{t-1})=0$	$R1_t$
2	$R1_t$	(4)	*Trans()*	-	$R2_{t+1}$
3	$R2_{t+1}, S_{t+3}$	(3)	*MUX*	$lsb(R1_{t+1})=0$	$R1_{t+2}$
4	$S_{t+1}, S_{t+2}, S_{t+3}, S_{t+4}$	(10)	*Serpent1()*	-	$f_{t+1}, f_{t+2}, f_{t+3}, f_{t+4}$
5	$f_{t+1}, S_{t+10}, R2_{t+1}$	(5)	*FSM*	-	$R1_{t+1}$
6	$R1_{t+1}$	(4)	*Trans()*	-	$R2_{t+2}$
7	$R1_{t+2}$	(4)	*Trans()*	-	$R2_{t+3}$
8	$f_{t+2}, R1_{t+2}, R2_{t+2}$	(5)	*FSM*	-	S_{t+11}

Following the steps 1 to 8 of Table 3, we can find a check equation as follows.

$$S_{t+11} \stackrel{?}{=} S_{t+10} + \alpha^{-1} S_{t+4} + \alpha S_{t+1} \quad \textbf{(12)}$$

This equation can filter some initial guesses and the remaining guesses will pass to other steps. The number of successful guesses during this part is $2^{6*32} * 2^{-32} = 2^{160}$. We then continue the determination phase with the successful ones and guess two more elements S_{t+7} and S_{t+8}, in order to find the output of *Serpent1*, that increases the complexity of the attack by $O(2^{64})$ as it is depicted in Table 4.

Table 4. Determination phase in modified GD attack on SOSEMANUK (Contd.)

Step No.	Known Element(s)	Relation	Function/ Part	Assumption	Determined Element(s)
9	$R2_{t+2}$, S_{t+4}, S_{t+11}, $lsb(R1_{t+2})$	(3)	*MUX*	-	$R1_{t+3}$
10	f_{t+3}, $R1_{t+3}$, $R2_{t+3}$	(5)	*FSM*	-	S_{t+12}
11	S_{t+2}, S_{t+11}, S_{t+12}	(2)	*LFSR*	-	S_{t+5}
12	$R2_{t+3}$, S_{t+5}, S_{t+12}, $lsb(R1_{t+3})$	(3)	*MUX*	-	$R1_{t+4}$
13	f_{t+4}, $R1_{t+4}$, $R2_{t+4}$	(5)	*FSM*	-	S_{t+13}
14	S_{t+3}, S_{t+12}, S_{t+13}	(2)	*LFSR*	-	S_{t+6}
15	$R2_{t+4}$, S_{t+6}, S_{t+13}, $lsb(R1_{t+4})$	(3)	*MUX*	-	$R1_{t+5}$
16	S_{t+5}, S_{t+6}, S_{t+7}, S_{t+8}	(10)	*Serpent1()*	-	f_{t+5}, f_{t+6}, f_{t+7}, f_{t+8}
17	S_{t+4}, S_{t+7}, S_{t+13}	(2)	*LFSR*	-	S_{t+14}
18	S_{t+5}, S_{t+8}, S_{t+14}	(2)	*LFSR*	-	S_{t+15}
19	$R1_{t+4}$	(4)	*Trans()*	-	$R2_{t+5}$
20	$R1_{t+5}$	(4)	*Trans()*	-	$R2_{t+6}$
21	$R2_{t+5}$, S_{t+7}, S_{t+14}, $lsb(R1_{t+5})$	(3)	*MUX*	-	$R1_{t+6}$
22	$R2_{t+6}$, S_{t+8}, S_{t+15}, $lsb(R1_{t+6})$	(3)	*MUX*	-	$R1_{t+7}$
23	$R1_{t+6}$	(4)	*Trans()*	-	$R2_{t+7}$
24	f_{t+7}, $R1_{t+7}$, $R2_{t+7}$	(5)	*FSM*	-	S_{t+16}
25	S_{t+6}, S_{t+15}, S_{t+16}	(2)	*LFSR*	-	S_{t+9}

So, up to step 25 the variables S_{t+1} to S_{t+16} will be known, based on which the other variables are determined.

4.1 The Complexity of the Attack

In the introduced GD attack, we made an initial guess of six 32-bit elements and implement step 1 to step 8 for each of 2^{192} possible guessed values. In this stage, we

assumed two bits of memory be zero that makes us run the attack four times in average to find the correct guessed values. Considering the relation (12), we continued the process for about 2^{160} out of 2^{192} possible guess values by filtering some wrong initial guesses. During the remaining process, we had to guess two more 32-bit variables that increases the complexity of the process by $O(2^{64})$. The total complexity is 2^{194} times implementing steps 1 to 8 added by 2^{162} times guessing 64 bits of state and continuing the process, that is $O(2^{194}) + O(2^{226})$. Therefore, the designed attack has a complexity of $O(2^{226})$ basic operations over $GF(2^{32})$.

5 Conclusions

In this paper, we have introduced a Guess and Determine attack on the stream cipher SOSEMANUK, based on Advanced Guess and Determine attacks. The complexity of the introduced attack is of $O(2^{226})$ that is less than the complexity of an exhaustive search of the key space for more than 226-bit key lengths. So, this result shows that the proposed algorithm is still alive and can afford 128-bit security according to the claim of the authors. However, from a security point of view, it is preferred to use the cipher with keys up to 226-bit length. As a future work, we suggest to approximate the function *Serpent1* in a proper way to decrease the complexity of the introduced Guess and Determine attack.

References

1. New European Schemes for Signature, Integrity and Encryption, see: `https://www.cosic.east.kuleuven.be/nessie.`
2. Ekdahl P., *On LFSR Based Stream Ciphers Analysis and Design,* Ph.D. Thesis, Department of Information Technology, Lund University, Sweden, 2003.
3. Berbain C., Billet O., Canteaut A., Courtios N., Gilbert H., Goubin L., Gouget A., Granboulan L., Lauradoux C., Minier M., Ptonin T. and Sibert H., *"SOSEMANUK, a fast software-oriented stream cipher"*, eSTREAM, ECRYPT Stream Cipher Project Report 2005/027 (2005) `http://www.ecrypt.eu.org/stream/.`
4. Ekdahl P., Johansson T., "*A new version of the stream cipher SNOW"*, SAC 2002, LNCS 2595, pp. 47-61, 2002.
5. Ahmadi H. and Eghlidos T., "*Advanced Guess and Determine Attacks on Stream Ciphers*", IST 2005, pp. 87-91, 2005.
6. Biham E., Anderson R. and Knudsen L., "SERPENT: A new block cipher proposal", FSE' 98, LNCS 1372, pp. 222-238, 1998.