

# Security analysis of the variant of the self-shrinking generator proposed at ICISC 2006

Dong Hoon Lee, Je Hong Park, and Jaewoo Han

National Security Research Institute  
161 Gajeong-dong, Yuseong-Gu, Daejeon, South Korea  
{dlee, jhpark, jwhan}@etri.re.kr

**Abstract.** In this paper, we revisit the variant of the self-shrinking generator (SSG) proposed by Chang et al. at ICISC 2006. This variant, which we call SSG-XOR was claimed to have better cryptographic properties than SSG in a practical setting. But we show that SSG-XOR has no advantage over SSG from the viewpoint of practical cryptanalysis.

**Keywords:** Stream cipher, LFSR, Self-Shrinking generator, Cryptanalysis

## 1 Introduction

The self-shrinking generator (SSG) is a well-known keystream generator proposed by Meier and Staffelbach [4]. SSG requires only one LFSR, which generates a binary sequence  $\mathbf{a} = (a_i)_{i \geq 0}$  in the usual way. For each bit pair  $(a_{2i}, a_{2i+1})$ , if  $a_{2i} = 1$ , SSG outputs  $a_{2i+1}$  as a keystream bit, otherwise no output is produced.

Until now, several methods have been proposed for attacking SSG [5, 3, 2, 6]. The designers of SSG have also described two kinds of simple attacks called exhaustive search and entropy attack whose time complexity is  $O(2^{0.79L})$  and  $O(2^{0.75L})$  respectively, where  $L$  is a length of the underlying LFSR [4]. The time complexity was reduced to  $O(2^{0.695L})$  in [5]. In [3] the BDD-attack was proposed, it requires  $O(2^{0.656L})$  time complexity from  $\lceil 2.41L \rceil$  bits keystream. However the memory requirement for the BDD-attack is infeasible. This attack was improved in [2]. The advantage of the HJ attack [2] over the BDD-attack is to have the almost same time complexity with only  $O(L^2)$  memory from  $L$ -bit keystream. Recently a new guess-and-determine attack was proposed [6]. It requires  $O(2^{0.556L})$  time with memory  $O(L^2)$  from  $O(2^{0.161L})$ -bit keystream for  $L \geq 100$  and requires  $O(2^{0.571L})$  time with memory  $O(L^2)$  from  $O(2^{0.194L})$ -bit keystream for  $L < 100$ .

The variant of SSG (denoted by SSG-XOR) was proposed by Chang et al. [1] to improve some cryptographic properties of SSG. It has a similar structure to SSG, but it handles 4-tuple of consecutive bits produced by the underlying LFSR to produce two keystream bits in a lump. For each 4-tuple  $(a_{4i}, a_{4i+1}, a_{4i+2}, a_{4i+3})$ , SSG-XOR outputs two bits  $a_{4i+2}$  and  $a_{4i+3}$  if  $a_{4i} \oplus a_{4i+1} = 1$ , and discards otherwise. The following Figure 1 clarifies the difference between SSG and SSG-XOR.

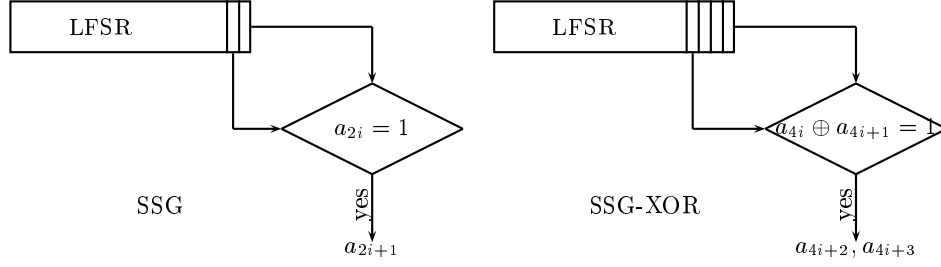


Fig. 1. SSG and SSG-XOR

The authors of [1] analyzed the security of SSG-XOR by applying the existing attacks for SSG and claimed that SSG-XOR is more secure than SSG against attacks using short keystream sequences such as entropy attack [4] or the BDD-attack [3].

In this paper, however, we show that the security of SSG-XOR against several attacks using short keystream sequences can be decreased significantly. First, we re-analyze the security of SSG-XOR against exhaustive search attack, entropy attack and the BDD-attack. And then we investigate the security of SSG-XOR against the HJ attack and the guess-and-determine attack. Our analysis shows that SSG-XOR has no advantage over SSG from the viewpoint of the security. In Table 1, we compare with the complexity of several attacks for SSG and SSG-XOR. (Note that we ignore some polynomial factors in Table 1.)

Table 1. Comparison of complexity of several attacks for SSG and SSG-XOR

	SSG			SSG-XOR		
	Time	Memory	Data	Time	Memory	Data
Exhaus. search [4]	$O(2^{0.79L})$	–	–	$O(2^{0.774L})$	–	–
Entropy attack [4]	$O(2^{0.75L})$	–	–	$O(2^{0.667L})$	–	–
BDD-attack [3]	$O(2^{0.656L})$	infeasible	$\lceil 2.41L \rceil$	$O(2^{0.631L})$	infeasible	$\lceil 2.21L \rceil$
HJ attack [2]	$O(2^{0.66L})$	$O(L^2)$	$L$	$O(2^{0.5L})$	$O(L^2)$	$L$
G & D attack [6]	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$	$O(2^{0.384L})$	$O(L^2)$	$O(2^{0.111L})$

## 2 Security Analysis

Although the designers of SSG-XOR analyzed the security against several attacks which have been mounted to the original SSG, the analysis would not be sufficient. So we re-analyze the resistance against possible attacks.

### 2.1 Exhaustive search and entropy attack

These attacks were described in the paper proposing SSG [4] to reconstruct the initial state with only a few keystream bits. The first attack is called exhaustive search. The initial state of SSG may be reconstructed with complexity  $2^{0.79L}$  by the exhaustive search attack. Let  $\mathbf{z} = (z_0, z_1, \dots, z_i, \dots)$  be a known short keystream of SSG-XOR. The designer of SSG-XOR claimed that the exhaustive search attack requires  $2^{0.8305L}$  steps since there are 10 possibilities to generate each  $(z_{2j}, z_{2j+1})$ . However, given a keystream  $\mathbf{z}$ , it is not necessary to guess  $a_{4i}$  and  $a_{4i+1}$  of the underlying LFSR output sequence  $\mathbf{a}$ , independently. Instead, we only guess one bit information whether  $a_{4i} \oplus a_{4i+1}$  is equal to 1 or not. This way, we will reconstruct an initial state that is not necessarily equal to the original initial state, but it is equivalent in a sense that it will create  $\mathbf{z}$ . From this point of view, there exist five possibilities rather than ten for a 3-tuple  $(a_{4i} \oplus a_{4i+1}, a_{4i+2}, a_{4i+3})$  of  $\mathbf{a}$ . So we can estimate that there exist

$$5^{L/3-1} \approx 5^{L/3} = 2^{((\log_2 5)/3)L} = 2^{0.774L}$$

possible initial states of the LFSR.

The second attack is called entropy attack. For SSG, the entropy per bit is  $3/4$  so an exhaustive search among all different cases in the order of probability would require  $2^{0.75L}$  steps. For SSG-XOR, the designers claimed the the entropy attack requires  $2^{0.8305L}$  steps. However, for each  $(z_{2j}, z_{2j+1})$  there are 5 different possibilities  $(a_{4i} \oplus a_{4i+1}, a_{4i+2}, a_{4i+3})$ , namely  $(1, z_{2j}, z_{2j+1})$ ,  $(0, 0, 0)$ ,  $(0, 0, 1)$ ,  $(0, 1, 0)$ , and  $(0, 1, 1)$ . The probability for  $(1, z_{2j}, z_{2j+1})$  is  $1/2$  and the probability for the others is  $1/8$ . Thus the entropy of 3-tuple is

$$H = -(1/2) \log_2(1/2) - 4 \cdot (1/8) \log_2(1/8) = 2.$$

Therefore, the entropy per bit is  $2/3$  so the complexity of the attack for SSG-XOR would be  $2^{0.667L}$ .

### 2.2 BDD-attack

For the BDD-attack [3], the required length of consecutive keystream bits is  $\lceil \gamma \delta^{-1} L \rceil$  and the time complexity is  $L^{O(1)} 2^{((1-\delta)/(1+\delta))L}$ , where  $\gamma$  and  $\delta$  are defined as follows:

- $\gamma$  is the maximal ratio of the length of the keystream  $\mathbf{z}$  to the length of the output sequence  $\mathbf{a}$  of the underlying LFSR.

- $\delta$  is the information rate (per bit) which would be revealed about the output sequence  $\mathbf{a}$  of the underlying LFSR from the keystream  $\mathbf{z}$ .

For SSG,  $\delta \approx 0.2075$  and  $\gamma = 0.5$ . Thus the BDD-attack for SSG can compute the initial state with  $\lceil 2.41L \rceil$  consecutive keystream bits in time  $L^{O(1)}2^{0.6563L}$ . The designers of SSG-XOR claimed that the BDD-attack for SSG-XOR can reconstruct the initial state from the  $\lceil 2.95L \rceil$  consecutive keystream bits in time  $L^{O(1)}2^{0.7101L}$ .

Now we re-analyze the security against the BDD-attack for SSG-XOR. We can also set  $\gamma = 0.5$  for SSG-XOR. For a fixed  $m$ , let  $p(m)$  be the probability that the shrinking result of a randomly chosen bitstream from  $\{0, 1\}^m$  is a prefix of the given keystream. If chosen bitstreams are uniformly distributed in  $\{0, 1\}^m$ , there are  $p(m)2^m$  possible  $z$ 's such that the shrinking result of  $z$  is a prefix of  $\mathbf{z}$ . Note that  $p(m)$  can be supposed to behave as  $p(m) = 2^{-\delta m}$ .

On the other hand, we observe that for all  $m$  with  $m \equiv 0 \pmod{4}$  and all keystreams  $\mathbf{z}$ , there are exactly  $5^{m/3}$  bitstreams  $z \in \{0, 1\}^m$  such that the shrinking result of  $z$  is a prefix of  $\mathbf{z}$ . Hence, we obtain an information rate  $\delta = 1 - (\log_2 5)/3 \approx 0.226$  for SSG-XOR by evaluating the relation  $2^{(1-\delta)m} = 5^{m/3}$ . So the required length of consecutive output bits is  $\lceil 2.21L \rceil$  and the time complexity is  $L^{O(1)}2^{0.6313L}$ .

### 2.3 HJ attack

Each known keystream bit gives, by default, a few equations in the initial state. Assume that we know  $2N$  keystream bits

$$z_0, z_1, \dots, z_{2N-1}. \quad (1)$$

In the case of SSG-XOR, each known keystream bit pair  $(z_{2i}, z_{2i+1})$  will give us three equations for some  $j$ :

$$a_{4j} \oplus a_{4j+1} = 1, \quad a_{4j+2} = z_{2i}, \quad a_{4j+3} = z_{2i+1}.$$

Additionally, we only know that the observed keystream sequence (1) corresponds to the output sequence of the underlying LFSR

$$a_0, \bar{a}_0, z_0, z_1, X_0, a_1, \bar{a}_1, z_2, z_3, X_1, \dots, X_{N-2}, a_{N-1}, \bar{a}_{N-1}, z_{2N-2}, z_{2N-1},$$

where  $\bar{a}_i = 1 \oplus a_i$  and each  $X_i$  corresponds to a sequence of zero or more 4-tuples in  $\{(0, 0, *, *), (1, 1, *, *)\}$ . For each of these 4-tuples that we guess correctly, we will get one more equation since the first two bits have the same bit parity. The total number of equations available is thus  $3N + k$  where  $k$  is the number of 4-tuples discarded. To get a complete system of equations in the (equivalent) initial state bits we require that  $N = \lceil (L - k)/3 \rceil$ . The probability that in total  $k$  4-tuples are discarded is, for each possible assumption,

$$2^{-N-k+1}.$$

The number of ways to discard  $k$  4-tuples in a total of  $N - 1$  gaps is given by

$$\binom{N - 2 + k}{k}.$$

We start by testing the case when 0 bits are discarded, then the case when 2 bits has been discarded, etc. The probability that we guess correctly within

$$\sum_{k=0}^{k_{\max}} \binom{N - 2 + k}{k}$$

guesses is

$$\sum_{k=0}^{k_{\max}} \binom{N - 2 + k}{k} 2^{-N-k+1}.$$

By fixing the probability of success to 0.5, we calculate the complexity of the attack for some different LFSR lengths. In Table 2, we can see that the complexity is approximately  $O(2^{0.5L})$ .

**Table 2.** The Complexity of the Attack for some LFSR Lengths

LFSR length	Complexity	$k_{\max}$
128	$2^{61.3}$	34
256	$2^{125.4}$	67
512	$2^{253.7}$	132
1024	$2^{510.5}$	262

## 2.4 Guess-and-determine attack

The guess-and-determine attack for SSG was recently proposed in Asiacrypt 2006 [6]. The proposed attack can restore the initial state with time complexity  $O(2^{0.556L})$  and memory complexity  $O(L^2)$  from  $O(2^{0.161L})$  keystream bits when  $L \geq 100$ . It utilizes the fact that the two decimated sequences  $\{a_{2i}\}$  and  $\{a_{2i+1}\}$  share the feedback polynomial as that of the sequence  $\{a_i\}$  which is a binary maximal length sequence produced by a LFSR of length  $L$  and differ by a shift value  $2^{L-1}$ . This approach can be applied to SSG-XOR immediately.

Let  $f(x) = 1 + c_1x + \dots + c_{L-1}x^{L-1} + x^L$  be the primitive feedback polynomial of the LFSR for the SSG-XOR, i.e. for each  $i \geq 0$ ,  $a_{i+L} = \sum_{j=1}^L c_j a_{i+L-j}$  where  $c_L = 1$ . Then the reciprocal of  $f(x)$  is  $x^L + c_1x^{L-1} + \dots + c_{L-1}x + 1$  which is

denoted by  $f^*(x)$ . It is easy to show that each  $a_i$  corresponds to  $x^i \bmod f^*(x)$  from the recurrence relation.

$$x^{i+L} = \sum_{j=1}^L c_j x^{i+L-j} \bmod f^*(x).$$

By squaring (or exponentiating by  $2^k$ ) the above formula, we can show that the decimated sequence  $\{a_{2i}\}$  (or  $\{a_{2^k i}\}$ ) shares the feedback polynomial with the original sequence  $\{a_i\}$ . Thus once we know any consecutive  $L$ -bit sequence of the decimated sequence, we can immediately compute the next sequences using the given recurrence relation. However other decimated sequences (for example  $\{a_{3i}\}$ ) would not share the feedback polynomial.

**Lemma 1.** *Let  $\mathbf{a} = \{a_0, a_1, \dots\}$  be a binary maximal length sequence produced by a LFSR of length  $L$ . Let  $\mathbf{s}^{(0)} = \{a_{4j}\}$ ,  $\mathbf{s}^{(1)} = \{a_{4j+1}\}$ ,  $\mathbf{s}^{(2)} = \{a_{4j+2}\}$ , and  $\mathbf{s}^{(3)} = \{a_{4j+3}\}$  be decimated sequences of  $\mathbf{a}$ . Then they share the feedback polynomial with the sequence  $\mathbf{a}$  and the shift value between  $\mathbf{s}^{(i)}$  and  $\mathbf{s}^{(i+1)}$  for  $i = 0, 1, 2$  is  $2^{L-2}$ .*

*Proof.* As mentioned before, each decimated sequence  $\mathbf{s}^{(i)} = \{s_j^{(i)}\}$  shares the feedback polynomial with the original sequence  $\mathbf{a}$ . Thus they differs each other by only some shift. Our lemma suffices to note that

$$s_{j+2^{L-2}}^{(i)} = a_{4(j+2^{L-2})+i} = a_{4j+2^L+i} = a_{4j+(i+1)+(2^L-1)} = s_j^{(i+1)}.$$

□

We define polynomials  $h_i(x)$  for  $i = 1, 2, 3$  as follows.

$$h_1(x) = \sum_{i=0}^{L-1} h_{1,i} x^i, \quad h_2(x) = \sum_{i=0}^{L-1} h_{2,i} x^i, \quad h_3(x) = \sum_{i=0}^{L-1} h_{3,i} x^i,$$

such that  $h_1(x) \equiv x^{2^{L-2}} \bmod f^*(x)$ ,  $h_2(x) \equiv x^{2^{L-1}} \bmod f^*(x)$ , and  $h_3(x) \equiv x^{3 \cdot 2^{L-2}} \bmod f^*(x)$ . Then we have

$$a_{4i+1} = \sum_{j=0}^{L-1} h_{1,j} a_{4(i+j)}, \quad a_{4i+2} = \sum_{j=0}^{L-1} h_{2,j} a_{4(i+j)}, \quad a_{4i+3} = \sum_{j=0}^{L-1} h_{3,j} a_{4(i+j)}.$$

Now we are ready to attack SSG-XOR. Let  $\{z_i\}_{i=0}^{N-1}$  be the keystream of SSG-XOR. We first set  $A = (a_0, a_4, \dots, a_{4(L-1)})$  with  $L$  variables. It is enough to find these unknowns for attacking SSG-XOR. Instead of guessing the unknowns directly, we guess an  $l$ -bit length segment  $\text{guess} = (g_0, g_1, \dots, g_{l-1})$  for  $(a_0 \oplus a_1, a_4 \oplus a_5, \dots, a_{4(l-1)} \oplus a_{4(l-1)+1})$ . Let  $H_w(\cdot)$  be the Hamming weight of the corresponding vector. Then from the guessed segment  $\text{guess}$ , we can obtain  $l +$

$2H_w(\text{guess})$  linear equations with  $L$  variables as follows.

$$\begin{aligned}
 a_{4i} \oplus a_{4i+1} &= a_{4i} \oplus \sum_{j=0}^{L-1} h_{1,j} a_{4(i+j)} = g_i, \quad \text{for } 0 \leq i < l, \\
 a_{4i+2} &= \sum_{j=0}^{L-1} h_{2,j} a_{4(i+j)} = z_{2(\sum_{j=0}^i g_j)-2}, \quad \text{for } g_i = 1 \\
 a_{4i+3} &= \sum_{j=0}^{L-1} h_{3,j} a_{4(i+j)} = z_{2(\sum_{j=0}^i g_j)-1}, \quad \text{for } g_i = 1.
 \end{aligned}$$

If we can solve the above system of linear equations, we can recover the initial state of the SSG-XOR. In order to solve the system, we have to get linear equations as many as possible. We observe that the more 1 in the guessed segment `guess`, the more linear equations can be obtained. To mount efficient attack, we just search over those possible `guess` satisfying the following condition instead of exhaustively searching over all the possible value.

$$H_w(\text{guess}) \geq \lceil \alpha l \rceil,$$

where  $\alpha$  ( $0.5 \leq \alpha \leq 1$ ) is a parameter to be determined later.

By the argument in [6], the obtained equations in the above process are almost linearly independent. Thus we have  $O(l + 2\alpha l)$  linearly independent equations with  $L$  variables. In order to solve the system of equations, we let

$$O(l + 2\alpha l) = L \implies l = O\left(\frac{1}{1 + 2\alpha} L\right).$$

The attack proceeds as follows.

1. For each guessed segment `guess` satisfying that  $H_w(\text{guess}) \geq \lceil \alpha l \rceil$  for a given parameter  $\alpha$ , derive linear expressions on the  $L$  variables without filling the constant terms (keystream bits) as explained above and store them in matrix  $U$ .
  - (a) For each  $0 \leq j \leq N - 1 - (l + 2\lceil \alpha l \rceil)$ , make (by filling constant terms) the system of linear equations with the linear expression  $U$  using the keystream bits starting from  $z_j$ .
  - (b) Solve the linear system  $U$ , find the candidate initial state, and check the candidate state is correct by running SSG-XOR with the state and comparing the generated stream with the (original) keystream bits  $\{z_i\}_{i=j}^{N-1}$ .
  - (c) If the above test succeeds, we find the initial state (or an equivalent state). Thus stop this process and output the state as a solution.
  - (d) If the above test fails, repeat the process from the step (a) with incrementing  $j$ .
2. If we could not find the initial state with the segment `guess`, we choose another `guess` at random and try again from the first step 1.

Now we determine the length  $N$  of keystream bits in order to succeed the above attack. Our search space is  $H = \{\text{guess} \mid \lceil \alpha l \rceil \leq H_w(\text{guess}) \leq l, \text{ and } g_0 = 1\}$ , thus the cardinality of  $H$  is

$$|H| = \sum_{i=\lceil \alpha l \rceil - 1}^{l-1} \binom{l-1}{i}.$$

For each  $l$ -bit guessed segment  $\text{guess}$ , we will try  $N - L$  times to find an equivalent state. To succeed the attack, we have to find at least one match pair between the guess set  $H$  and the initial state derived from the keystream segments involved in each  $N - L$  trials. Thus the length  $N$  should satisfy

$$(N - L) \cdot |H| \geq 2^{l-1} \implies N = O\left(2^{\frac{1-\beta}{1+2\alpha}L}\right),$$

where  $|H| = 2^{\beta l}$  and  $\beta$  is a parameter determined by  $\alpha$ .

Since for each guessed segment  $\text{guess}$  we have to try at most  $N - L$  times, the total time complexity of the attack in worst case is

$$O(L^3)O(N - L)O(2^{\beta l}) = O\left(L^3 \cdot 2^{\frac{1}{1+2\alpha}L}\right),$$

where  $O(L^3)$  factor reflects the complexity of solving a system of linear equations of size  $L$ .

**Theorem 1.** *The guess-and-determine attack for SSG-XOR has time complexity  $O\left(L^3 \cdot 2^{\frac{1}{1+2\alpha}L}\right)$ , memory complexity  $O(L^2)$  and data complexity  $O\left(2^{\frac{1-\beta}{1+2\alpha}L}\right)$ , where  $L$  is the length of the underlying LFSR of SSG-XOR,  $0.5 \leq \alpha \leq 1$ , and  $\beta$  is a parameter determined by  $\alpha$ .*

Now we give comparison result in the following table between SSG and SSG-XOR ignoring the polynomial factor  $L^3$ .

**Table 3.** The asymptotic time, memory, and data complexity to attack SSG and SSG-XOR when  $L \geq 100$

$\alpha$	$\beta$	SSG			SSG-XOR		
		Time	Memory	Data	Time	Memory	Data
0.5	0.99	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.007L})$	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.005L})$
0.8	0.71	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$	$O(2^{0.384L})$	$O(L^2)$	$O(2^{0.111L})$
1.0	0.00	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.5L})$	$O(2^{0.333L})$	$O(L^2)$	$O(2^{0.333L})$



*Experiments* We made several experimental results in C language on a general PC to check the validity of our attack. For example, we choose an LFSR's feedback polynomial of length 30 as follows.

$$f(x) = x^{30} + x^{27} + x^{23} + x^{22} + x^{17} + x^{16} + x^{14} + x^{11} + x^3 + x + 1.$$

We note that  $f(x)$  is a primitive polynomial, thus the generated sequence would have a maximal length. Then  $h_1(x)$ ,  $h_2(x)$ , and  $h_3(x)$  modulo the reciprocal  $f^*(x)$  can be obtained as follows.

$$\begin{aligned} h_1(x) &= x^{2^{28}} \bmod f^*(x) \\ &= x^{28} + x^{26} + x^{21} + x^{20} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^4 + x^2 + 1 \\ h_2(x) &= x^{2^{29}} \bmod f^*(x) \\ &= x^{29} + x^{28} + x^{28} + x^{27} + x^{21} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^4 \\ h_3(x) &= x^{3 \cdot 2^{28}} \bmod f^*(x) \\ &= x^{27} + x^{24} + x^{23} + x^{17} + x^{14} + x^{11} + x^{10} + x^6 + x^5 + x^3 \end{aligned}$$

For a random chosen initial state, our attack recovers the initial state or an equivalent state in a minute from 200 bits keystream.

### 3 Conclusion

In this paper, we investigated the security aspects for a variant of self-shrinking generator called SSG-XOR which was proposed at ICISC 2006. The author of SSG-XOR alleged that SSG-XOR is more secure than the original SSG in a sense that the complexity of attacks for SSG-XOR is higher than that of SSG. However we showed that the security of SSG-XOR does not reach that of SSG.

### References

1. K.-Y. Chang, J.-S. Kang, M.-K. Lee, H. Lee and D. Hong. New variant of the self-shrinking generator and its cryptographic properties. *Information Security and Cryptology - ICISC 2006*, Lecture Notes in Comput. Sci., vol. 4296, pp. 41–50, 2006.
2. M. Hell and T. Johansson. Two new attacks on the self-shrinking generator. *IEEE Trans. Infor. Theory*, vol. 52, no. 8, pp. 3837–3843, 2006.
3. M. Krause. BDD-based cryptanalysis of keystream generator. *Advances in Cryptology - EUROCRYPT 2002*, Lecture Notes in Comput. Sci., vol. 2332, pp. 222–237, 2002.
4. W. Meier and O. Staffelbach. The self-shrinking generator. *Advances in Cryptology - EUROCRYPT'94*, Lecture Notes in Comput. Sci., vol. 950, pp. 205–214, 1994.
5. E. Zenner, M. Krause and S. Lucks. Improved cryptanalysis of the self-shrinking generator. *Information Security and Privacy - ACISP 2001*, Lecture Notes in Comput. Sci., vol. 2119, pp. 21–35, 2001.
6. B. Zhang and D. Feng. New guess-and-determine attack on the self-shrinking generator. *Advances in Cryptology - ASIACRYPT 2006*, Lecture Notes in Comput. Sci., vol. 4284, pp. 54–68, 2006.