New Guess-and-Determine Attack on the Self-Shrinking Generator ⋆

Bin Zhang and Dengguo Feng

State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences,
Beijing 100080, P. R. China
martin_zhangbin@yahoo.com.cn

Abstract. We propose a new type of guess-and-determine attack on the self-shrinking generator (SSG). The inherent flexibility of the new attack enables us to deal with different attack conditions and requirements smoothly. For the SSG with a length L LFSR of arbitrary form, our attack can reliably restore the initial state with time complexity $O(2^{0.556L})$, memory complexity $O(L^2)$ from $O(2^{0.161L})$-bit keystream for $L \geq 100$ and time complexity $O(2^{0.571L})$, memory complexity $O(L^2)$ from $O(2^{0.194L})$-bit keystream for $L < 100$. Therefore, our attack is better than all the previously known attacks on the SSG and especially, it compares favorably with the time/memory/data tradeoff attack which typically has time complexity $O(2^{0.5L})$, memory complexity $O(2^{0.5L})$ and data complexity $O(2^{0.25L})$-bit keystream after a pre-computation phase of complexity $O(2^{0.75L})$. It is well-known that one of the open research problems in stream ciphers specified by the European STORK (Strategic Roadmap for Crypto) project is to find an attack on the self-shrinking generator with complexity lower than that of a generic time/memory/data tradeoff attack. Our result is the best answer to this problem known so far.

Keywords: Stream cipher, Self-shrinking, Guess-and-determine, Linear feedback shift register (LFSR).

1 Introduction

The self-shrinking generator is an elegant keystream generator proposed by W. Meier and O. Staffelbach at EUROCRYPT'94 [22]. It applies the shrinking idea [7] to only one maximal length LFSR and generates the keystream according to the following rule: let $a = a_0, a_1, \ldots$ be a binary sequence produced by the LFSR, consider the bit pair (a_i, a_{i+1}), if $a_i = 1$, output a_{i+1} as a keystream bit, otherwise no output is produced. It is suggested in [22] that the key of the SSG consists of the initial state of the LFSR and (preferably) also of the

⋆ Supported by the National Natural Science Foundation of China (Grant No. 90604036, 60373047) and the National Grand Fundamental Research 973 program of China (Grant No. 2004CB318004)

LFSR feedback logic. As in other articles, e.g. [18, 26, 31, 3], we assume that the primitive feedback polynomial is known to the attacker.

Although many LFSR based stream ciphers are found vulnerable to (fast) correlation attacks [4, 5, 14–16, 23–25] and algebraic attacks [1, 2, 8, 9], the self-shrinking generator has shown remarkable resistance against such cryptanalysis. For a length L LFSR, the previously known best concrete attack is the BDD attack in [18], which has time complexity $O(2^{0.656L})$ at the expense of $O(2^{0.656L})$ memory from $\lceil 2.41 \cdot L \rceil$ bits keystream. One of the open research problems in stream ciphers specified by the STORK (Strategic Roadmap for Crypto) project [29] is to find an attack on the self-shrinking generator with complexity lower than that of a generic time/memory/data (TMD) tradeoff attack, which typically has time complexity $O(2^{0.5L})$, memory complexity $O(2^{0.5L})$ by using $O(2^{0.25L})$-bit keystream after a pre-computation phase of complexity $O(2^{0.75L})$.

In [22], a simple method of reducing the key space is introduced and the entropy leakage analysis shows that the average key space of the self-shrinking generator is $O(2^{0.75L})$. A faster cryptanalysis of the SSG is proposed by Mihaljević in [26] with time complexity varying from $O(2^{0.5L})$ to $O(2^{0.75L})$ and the required keystream length ranging from $2^{0.5L}$ to $2^{0.25L}$ accordingly. To get the best complexity estimation $O(2^{0.5L})$, the intercepted keystream length must be greater than $L/2 \cdot 2^{L/2}$, which is beyond the realistic scope for large value of L. In [31], a search tree algorithm is presented to restore an equivalent state of the LFSR from a short segment of the keystream with time complexity $O(2^{0.694L})$. However, the main bottleneck of the attacks in [31, 18] is their unrealistically large requirement of memory. Since the self-shrinking generator uses only one LFSR, the method of reducing the memory complexity in [17] is inapplicable. In 2003, P. Ekdahl et al. showed that certain week feedback polynomials allow very efficient distinguishing attacks on the SSG [10]. Except for these concrete attacks, there is a general time/memory/data tradeoff attack [3] applicable to all stream ciphers in theory. This kind of attack should be taken into consideration especially when a technique called BSW sampling [3] is applicable to the cipher system. It is known that the sampling resistance of the self-shrinking generator is $2^{-L/4}$, thus the reduced search space is $O(2^{0.75L})$. However, such an attack always has a time-consuming preprocessing phase and requires large amount of memory, which are usually impossible for individual cryptanalysts.

In this paper, we propose a new type of guess-and-determine attack on the self-shrinking generator. The large flexibility inherent in the new attack enables us to handle different attack conditions and requirements smoothly. It has no restriction on the form of the LFSR and can reliably recover the initial state of the LFSR with time complexity $O(2^{0.556L})$, memory complexity $O(L^2)$ from $O(2^{0.161L})$-bit keystream for $L \geq 100$ and time complexity $O(2^{0.571L})$, memory complexity $O(L^2)$ from $O(2^{0.194L})$-bit keystream for $L < 100$. Compared with the general time/memory/data tradeoff attack, our attack avoids the time-consumptive pre-computation phase and the large memory requirement in the TMD attack, while without a substantial compromise of the real processing complexity. Comparisons with other known attacks against the self-shrinking generator show that our attack offers the best tradeoff between the complexities (time, memory and pre-computation) and the required keystream length. Therefore, our result is the best answer to the open problem in STORK project known so far.

The rest of this paper is organized as follows. We present a detailed description of our attack in Section 2 with theoretical analysis. In Section 3, experimental results to verify the feasibility of our attack and comprehensive comparisons with the previously known attacks on the self-shrinking generator are provided. Finally, some conclusions are given in Section 4.

2 Our Attack

The aim of our attack is to restore the initial state or an equivalent initial state of the LFSR used in the self-shrinking generator from a keystream segment of realistic length. We first state some basic facts on the self-shrinking generator and on the underlying maximal length sequences, then the guess-and-determine attack is presented in detail followed by the theoretical complexity analysis.

2.1 Basic Facts

Let $a = a_0, a_1, \ldots$ be the maximal length sequence produced by LFSR A used in the self-shrinking generator and $z = z_0, z_1, \ldots$ be the keystream. First note that the two decimated sequences $a_0, a_2, \ldots, a_{2i}, \ldots$ and $a_1, a_3, \ldots, a_{2i+1}, \ldots$ are shift equivalent to the original sequence a [13]. They share the same feedback polynomial as that of sequence a and differ only by some shift. The following lemma determines the shift value between sequence $\{a_{2i}\}$ and $\{a_{2i+1}\}$.

Lemma 1. *Let $a = a_0, a_1, \ldots$ be a binary maximal length sequence produced by a LFSR of length L, then the shift value τ between the two decimated sequences $c = \{a_{2i}\}$ and $b = \{a_{2i+1}\}$ is 2^{L-1}, i.e. for each integer $i \geq 0$, $b_i = c_{i+2^{L-1}}$.*

Proof. It suffices to note that $c_{i+2^{L-1}} = a_{2\cdot(i+2^{L-1})} = a_{2i+2^L} = a_{2i+1+2^L-1} = a_{2i+1} = b_i$.

Lemma 1 shows the exact shift value between $\{a_{2i}\}$ and $\{a_{2i+1}\}$, which will facilitate the determination of the relationship between them. Keep the notations as above, we have the following lemma.

Lemma 2. *Let $f(x) = 1 + c_1x + c_2x^2 + \cdots + c_{L-1}x^{L-1} + x^L$ be the primitive feedback polynomial of LFSR A over $GF(2)$, 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 there exists a polynomial $h(x) = \sum_{i=0}^{L-1} h_i x^i$ such that $h(x) \equiv x^\tau \bmod f^*(x)$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$ and $\tau = 2^{L-1}$ is the shift value between $c = \{a_{2i}\}$ and $b = \{a_{2i+1}\}$. Besides, the polynomial $h(x)$ can be efficiently computed as illustrated below for very large value of L.*

Proof. The former part of this lemma is a straightforward conclusion according to the theory of maximal sequences [13]. It reveals that

$$b_i = a_{2i+1} = \sum_{j=0}^{L-1} h_j c_{i+j} = \sum_{j=0}^{L-1} h_j a_{2(i+j)}, \tag{1}$$

i.e. each b_i is a linear combination of some c_i.

We follow the following recursive procedures to compute $h(x)$. More precisely, the linear coefficients h_j can be determined by recursively computing $x^i \bmod f^*(x) = x(x^{i-1} \bmod f^*(x)) \bmod f^*(x)$ for moderately large L. For very large value of L, this can be fulfilled by the combination of the recursive procedure with the following small step strategy, i.e. we first determine a set of values $\{\tau_1, \cdots, \tau_t\}$ such that

$$x^{2^{L-1}} \bmod f^*(x) = x^{\prod_{j=1}^{t} \tau_j} \bmod f^*(x) = ((x^{\tau_1} \bmod f^*(x))^{\tau_2} \cdots)^{\tau_t} \bmod f^*(x),$$

where $\prod_{j=1}^{t} \tau_t = 2^{L-1}$ and each τ_j is chosen so that $x^{\tau_j} \bmod f^*(x)$ can be computed efficiently by the available method such as the Square-and-Multiply method [20] in rational time. Hence, the linear coefficients h_j can be computed in an acceptable time for very large L in this way.

Table 1 lists the corresponding $h(x)$, obtained by the above combination method, of some primitive polynomials of length up to 300. Here we use $\tau_i = 2^{10}$ for $i = 1, \cdots, \lceil (L-1)/10 \rceil - 1$ and $\tau_{\lceil (L-1)/10 \rceil} = 2^{L-1-10\cdot(\lceil (L-1)/10 \rceil - 1)}$ so that even $x^{2^{299}} \bmod f^*(x)$ with $f(x)$ being a primitive polynomial of degree 300 can be computed in about one hour on a Pentium 4 Processor. This completes the proof.

Lemma 2 shows that compared with the real attack complexity $O(2^{0.556L})$ or $O(2^{0.571L})$, the complexity of computing the linear relationship between $\{a_{2i}\}$ and $\{a_{2i+1}\}$ is negligible. The overall complexity of our attack is dominated by the complexity of the guess-and-determine algorithm given below.

2.2 The Guess-and-Determine Algorithm

The basic idea of a guess-and-determine attack on a stream cipher is to guess some bits of the internal state and derive other bits of the internal state through the relationship between the keystream bits and the internal state bits introduced by the keystream generation process. The validity of a guessed and determined internal state is checked by running the cipher forward from that state. If the generated keystream matches the intercepted keystream, we accept it. Otherwise, we discard the current candidate and try the attack again to get new state candidates.

Oppositely to the methods in other articles, here we do not directly apply the guess-and-determine idea to sequence $\{a_i\}$. Instead we consider the decimated sequence $\{a_{2i}\}$. With the knowledge of $\{a_{2i}\}$, $\{a_i\}$ can be easily recovered from simple linear algebra.

Table 1. Computational results of $h(x)$ on a Pentium 4 processor using Mathematica with the above combination method.

$f(x)$	$x^{2^{L-1}} \bmod f^*(x)$
$1+x+x^{37}+x^{38}+x^{80}$	$x^2+x^4+x^5+x^6+x^7+x^{11}$ $+x^{14}+x^{15}+x^{17}+x^{19}+x^{20}$ $+x^{21}+x^{23}+x^{24}+x^{25}+x^{29}$ $+x^{32}+x^{33}+x^{35}+x^{37}+x^{38}$ $+x^{39}+x^{41}+x^{44}+x^{45}+x^{46}$ $+x^{47}+x^{51}+x^{54}+x^{55}+x^{57}$ $+x^{59}+x^{60}+x^{62}+x^{63}+x^{64}$ $+x^{65}+x^{69}+x^{72}+x^{73}+x^{75}$ $+x^{77}+x^{78}+x^{79}$
$1+x^{37}+x^{100}$	$x^{19}+x^{32}+x^{69}$
$1+x^2+x^{15}+x^{17}+x^{168}$	$x^8+x^{76}+x^{77}+x^{91}+x^{92}$
$1+x^7+x^{18}+x^{36}+x^{83}+x^{130}+x^{206}+x^{253}+x^{300}$	$x^6+x^9+x^{11}+x^{16}+x^{21}+x^{23}$ $+x^{24}+x^{25}+x^{26}+x^{30}+x^{32}$ $+x^{33}+x^{34}+x^{35}+x^{36}+x^{37}$ $+x^{38}+x^{41}+x^{43}+x^{44}+x^{45}$ $+x^{46}+x^{54}+x^{55}+x^{56}+x^{57}$ $+x^{60}+x^{65}+x^{68}+x^{70}+x^{71}$ $+x^{75}+x^{76}+x^{78}+x^{80}+x^{82}$ $+x^{83}+x^{84}+x^{85}+x^{87}+x^{89}$ $+x^{91}+x^{92}+x^{93}+x^{94}+x^{95}$ $+x^{96}+x^{97}+x^{98}+x^{102}$ $+x^{104}+x^{105}+x^{107}+x^{109}$ $+x^{110}+x^{112}+x^{113}+x^{115}$ $+x^{118}+x^{120}+x^{122}+x^{125}$ $+x^{126}+x^{128}+x^{129}+x^{136}$ $+x^{139}+x^{141}+x^{146}+x^{147}$ $+x^{151}+x^{153}+x^{154}+x^{155}$ $+x^{156}+x^{160}+x^{162}+x^{163}$ $+x^{164}+x^{165}+x^{166}+x^{167}$ $+x^{168}+x^{171}+x^{173}+x^{174}$ $+x^{175}+x^{179}+x^{181}+x^{183}$ $+x^{184}+x^{185}+x^{186}+x^{187}$ $+x^{188}+x^{190}+x^{191}+x^{196}$ $+x^{200}+x^{201}+x^{203}+x^{204}$ $+x^{209}+x^{213}+x^{214}+x^{215}$ $+x^{216}+x^{217}+x^{218}+x^{219}$ $+x^{220}+x^{228}+x^{231}+x^{232}$ $+x^{233}+x^{238}+x^{239}+x^{241}$ $+x^{243}+x^{245}+x^{246}+x^{247}$ $+x^{248}+x^{252}+x^{254}+x^{255}$ $+x^{256}+x^{257}+x^{258}+x^{260}$ $+x^{263}+x^{265}+x^{266}+x^{267}$ $+x^{270}+x^{273}+x^{276}+x^{277}$ $+x^{282}+x^{289}+x^{290}+x^{291}$ $+x^{295}+x^{296}+x^{298}+x^{299}$

More precisely, to attack a self-shrinking generator, we first guess a l-bit length segment

$$A_0^{l-1} = (a_0, a_2, \cdots, a_{2(l-1)}) \tag{2}$$

of the initial state $(a_0, a_2, \cdots, a_{2(L-1)})$ of $\{a_{2i}\}$, as shown in Figure 1, thus there are $L - l$ bits (black points in Figure 1) of the initial state left unknown. Let $W_H(\cdot)$ be the hamming weight of the corresponding vector, then from the guessed segment, we can get $W_H(A_0^{l-1})$ linear equations on the remaining $L - l$ bits via the shift structure (illustrated by arrowhead in Figure 1). For example, if $a_{2i} = 1$ $(0 \leq i \leq l-1)$, then we have

$$b_i = a_{2i+1} = \sum_{j=0}^{L-1} h_j a_{2(i+j)} = \sum_{j=0}^{l-1} h_j a_{2(i+j)} + \sum_{j=l}^{L-1} h_j a_{2(i+j)} = z_{\sum_{j=0}^{i-1} a_{2i}}, \tag{3}$$

where $h(x) = \sum_{j=0}^{L-1} h_j x^j$ is the polynomial satisfying $h(x) \equiv x^{2^{L-1}} \bmod f^*(x)$ found by Lemma 2. Note that the partial sum $\sum_{j=0}^{l-1} h_j a_{2(i+j)}$ in (3) is a known

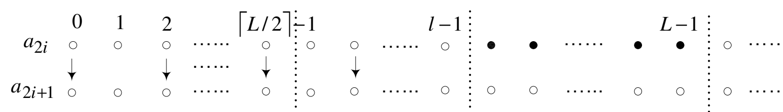

Fig. 1. Guess-and-determine process

parameter because we guessed the value of $(a_0, a_2, \cdots, a_{2(l-1)})$, thus (3) is a linear equation on $L - l$ variables $(a_{2l}, \cdots, a_{2(L-1)})$. Once there is a bit $a_{2i} = 1$ for $0 \leq i \leq l - 1$, we will have one linear equation on $(a_{2l}, \cdots, a_{2(L-1)})$. *Our observation* is that the more 1 in the guessed segment A_0^{l-1}, the more linear equations on the remaining $L - l$ bits we can get. The extreme case is that if $(a_0, a_2, \cdots, a_{2(l-1)}) = (1, 1, \cdots, 1)$, then we will have l linear equations on $L - l$ variables. In order to get an efficient attack, here we do not exhaustively search over all the possible values of A_0^{l-1}. Instead, we just search over those possible values of A_0^{l-1} satisfying (without loss of generality, we assume $a_0 = 1$)

$$W_H(A_0^{l-1}) \geq \lceil \alpha \cdot l \rceil, \tag{4}$$

where $\lceil x \rceil$ gives the smallest integer greater than or equal to x and α $(0.5 \leq \alpha \leq 1)$ is a parameter to be determined later. Hence, we can get at least $\lceil \alpha \cdot l \rceil$ linear equations on the remaining $L - l$ bits by this method.

Now a crucial question arises naturally, i.e. how about the linear dependency of these linear equations? Fortunately, from the initial state $(a_0, a_2, \cdots, a_{2(L-1)})$ of $\{a_{2i}\}$, we have

$$(a_0, \cdots, a_{2(L-1)}, a_{2L}, \cdots, a_{2(N-1)}) = (a_0, a_2, \cdots, a_{2(L-1)}) \cdot G,$$

where N is the length of sequence $\{a_{2i}\}$ under consideration and G is a $L \times N$ matrix over $GF(2)$:

$$G = \begin{pmatrix} g_0^0 & g_1^0 & \cdots & g_{N-1}^0 \\ g_0^1 & g_1^1 & \cdots & g_{N-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ g_0^{L-1} & g_1^{L-1} & \cdots & g_{N-1}^{L-1} \end{pmatrix},$$

i.e. each a_{2i} is a linear combination of $(a_0, a_2, \cdots, a_{2(L-1)})$. Since for each $i \geq 0$, $a_{2i+1} = a_{2i+2^{L-1}}$, the column vectors $g_i = (g_i^0, g_i^1, \cdots, g_i^{L-1})^T$ corresponding to the bits selected in $(a_1, a_3, \cdots, a_{2l-1})$ according to the pattern of $(a_0, a_2, \cdots, a_{2(l-1)})$ can be regarded as random vectors over $GF(2)^L$. Thus, this holds also for the truncated versions of g_i over $GF(2)^{L-l}$ which form the coefficient matrix on the remaining $L-l$ unknown bits. The following lemma guarantees that the matrix formed by the truncated random column vectors always has the rank close to its maximum.

Lemma 3. *([30]) The probability that a random generated $m \times n$ binary matrix has rank r $(1 \leq r \leq min(m,n))$ is*

$$P_r = 2^{r(m+n-r)-nm} \prod_{i=0}^{r-1} \frac{(1-2^{i-m})(1-2^{i-n})}{1-2^{i-r}} \ . \tag{5}$$

Although we can sometimes get more than $\lceil \alpha l \rceil$ linear equations by the above searching method, we only use the lower bound $\lceil \alpha l \rceil$ in the estimation of the linear independent equations and let $\lceil \alpha \cdot l \rceil = L - l$. The reason for doing so is to derive the worst case complexity of our guess-and-determine algorithm in the Section 2.3. By lemma 3, the probability that a random generated $\lceil \alpha l \rceil \times (L-l)$ binary matrix has rank $r \geq \lceil \alpha l \rceil - 5$ is

$$P(r \geq \lceil \alpha l \rceil - 5) = \sum_{r=\lceil \alpha l \rceil - 5}^{\lceil \alpha l \rceil} 2^{-(r-\lceil \alpha l \rceil)(r-L+l)} \prod_{i=0}^{r-1} \frac{(1-2^{i-\lceil \alpha l \rceil})(1-2^{i-L+l})}{1-2^{i-r}}. \tag{6}$$

Simulation results show that $P(r \geq \lceil \alpha l \rceil - 5) \geq 0.99$ for $L \leq 1500$, i.e. the linear equations we get are almost linear independent. We can compensate the linear dependency of the linear system by an exhaustive search at a small scale.

The entire description of the guess-and-determine attack (algorithm A) is as follows (in C-like notation).

- **Parameter:** α, L
- **Input:** keystream $\{z_i\}_{i=0}^{N-1}$, feedback polynomial $f(x)$
- **Processing:**
  1. Apply the combination strategy illustrated in Section 2.1 to compute $x^{2^{L-1}}$ mod $f^*(x)$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$
  2. **for** all l-bit segment A_0^{l-1} satisfying (4) **do**
     - **for** $k = 0$ **to** $l - 1$ **do**

* **if** $a_{2k} = 1$ **then**
  Using $h(x)$ obtained in step 1 and $f(x)$, derive a linear expression on the remaining bits in $A_l^{L-1} = (a_{2l}, \cdots, a_{2(L-1)})$ and store the expression in matrix U
  end if

end for

- **for** $j = 0$ **to** $N - 1 - \lceil \alpha \cdot l \rceil$ **do**
  (a) Check the linear consistency [32] of the linear system using keystream indexed from z_j
  (b) **if** the linear consistency test is OK **then**
  * Solve the linear system in U according to the keystream indexed from z_j to get a state candidate $(a'_0, a'_2, \cdots, a'_{2(L-1)})$ or a small list of candidates
  * **for** each candidate state **do**
    i. Run the SSG forward from the candidate state and check the generated keystreams with $\{z_i\}_{i=j}^{N-1}$
    ii. **if** the correlation test is OK **then**
    Output that candidate and break the loop
    else continue
    end if

    end for

  else continue
  end if

  end for

end for

– **Output:** the initial state or an equivalent state $(a_0, a_2, \cdots, a_{2(L-1)})$

Here the *for* loop works in the same way as in C language. Assume we start with the keystream $\{z_i\}_{i=0}^{N-1}$. We first derive the linear expressions as in (3) from the guessed segment A_0^{l-1}, then associate them with the keystream indexed from z_0 and test the linear consistency of the resulting system. If the test fails, then try the keystream indexed from z_1, indexed from z_2, ..., and so on. If we cannot get a consistent linear system based on the keystream in hand, discard the current guess of A_0^{l-1} and try another guess to restart. If we find it, solve the system to get a candidate state $(a'_0, a'_2, \cdots, a'_{2(L-1)})$ or a small list of candidate states. Run the self-shrinking generator forward from each candidate state and generate the corresponding keystream. If the generated keystream does not match the intercepted keystream, discard that candidate and try another one. If all the candidates failed to find a match, then try another guess of A_0^{l-1} to restart the above whole process. If enough keystream is available, we expect to find the initial state (or an equivalent state) corresponding to the intercepted keystream with high success probability.

2.3 Complexity Analysis

Now we analyze the time, memory and data complexity of the algorithm A. We first establish the basic equation of our attack. Then, the corresponding time, memory and data complexity are derived in the most general case, respectively.

Finally, we discuss the success rate of the algorithm A and point out the optimal choices of the attack parameters.

From algorithm A, to cover the $L-l$ unknown bits by $O(\alpha \cdot l)$ linear independent equations, we let

$$O(\alpha \cdot l) = L - l \implies l = O(\frac{1}{1+\alpha} \cdot L) \ . \tag{7}$$

Since we just want to derive the magnitude, here we ignore the possible small number of linear dependent equations.

In algorithm A, we only search over those possible values of A_0^{l-1} that satisfy (4). Let $H = \{A_0^{l-1} \mid \lceil \alpha l \rceil \leq W_H(A_0^{l-1}) \leq l \text{ and } a_0 = 1\}$, then

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

where $|\cdot|$ denotes the cardinality of a set. The proportion between the l-bit values contained in $|H|$ and all the 2^l possible values is $\frac{|H|}{2^l}$, we rewrite it as

$$\frac{\sum_{i=\lceil \alpha l \rceil -1}^{l-1} \binom{l-1}{i}}{2^l} = \frac{2^{\beta l}}{2^l} = 2^{-(1-\beta)\cdot l} \ , \tag{8}$$

where β is a parameter determined by α and l. From (8), we have

$$\beta = \frac{1}{l} \cdot \log_2 \sum_{i=\lceil \alpha l \rceil -1}^{l-1} \binom{l-1}{i}. \tag{9}$$

Combining with (7), we have a function $\beta = \beta(\alpha, L)$, as shown in Figure 2. It is worth noting that β decreases with α increasing.

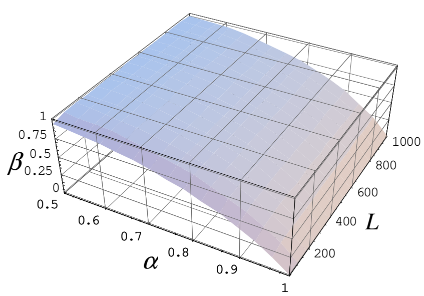

Fig. 2. β as a function of α and the LFSR length L

For the algorithm A to succeed, we must find at least one match pair between the state set H and the keystream segments involved in algorithm A. Assume sequence $\{a_i\}$ is purely random (consisting of independent and uniformly distributed binary random variables), thus the keystream length N should satisfy

$$(N-L) \cdot \sum_{i=\lceil \alpha l \rceil -1}^{l-1} \binom{l-1}{i} (\frac{1}{2})^{l-1} \geq 1,$$

i.e.

$$N > \frac{2^{l-1}}{\sum_{i=\lceil \alpha l \rceil -1}^{l-1} \binom{l-1}{i}} = \frac{2^{l-1}}{2^{\beta \cdot l}} = 2^{(1-\beta)\cdot l-1} \Longrightarrow N \sim O(2^{\frac{1-\beta}{1+\alpha}\cdot L}) \ . \qquad (10)$$

Algorithm A searches over the state set H and at each iteration, it checks along the keystream $\{z_i\}_{i=0}^{N-1}$ to find the suited segment. Therefore, the worst case time complexity is

$$O(N-L) \cdot O(2^{\beta \cdot l}) = O(2^{\frac{1}{1+\alpha}\cdot L}) \ . \qquad (11)$$

The following theorem summarizes the above results.

Theorem 1. *Keep the notations as above. The guess-and-determine algorithm A in section 2.2 has time complexity $O(L^3 \cdot 2^{\frac{1}{1+\alpha}\cdot L})$, memory complexity $O(L^2)$ and data complexity $O(2^{\frac{1-\beta}{1+\alpha}\cdot L})$, where L is the length of the LFSR used in the SSG, $0.5 \leq \alpha \leq 1$, β is a parameter determined by α and L.*

Proof. For the time complexity, note (11) and that in each iteration of algorithm A, we have to check the linear consistency of the linear system and then solve it. This contributes the L^3 factor to time complexity. For the memory complexity, it suffices to note that in the algorithm A, we only need to store the matrix U corresponding to the current guess of A_0^{l-1} and the memory usage in step 2 is dominating. The data complexity follows (10).

Corollary 1. *Keep the notations as those in Theorem 1 and under the above complexities, the success probability of algorithm A is*

$$P_{succ} = 1 - (1 - 2 \cdot 2^{-\frac{1-\beta}{1+\alpha}\cdot L})^{N-L} \ ,$$

where N is the length of the keystream used in the attack.

Proof. It suffices to note that in algorithm A, we totally check $N-L$ keystream segments and each segment matches to a state in H with probability $2 \cdot 2^{-\frac{1-\beta}{1+\alpha}\cdot L}$.

To get the optimal performance of our attack, we should optimize the parameters α and β of the algorithm A. Table 2 lists the asymptotic time, memory and data complexities corresponding to the different choices of α with the LFSR length $L \geq 100$. It is worth noting that the values of β are just approximations. In a real attack, we recommend using (7) and (9) to compute the more accurate values. (In Table 2, 3 and 4, we ignore the polynomial factors in the corresponding time complexities of these attacks, e.g. for the attack in [31], this factor is L^4 and for the BDD-based attack in [18], this factor is $L^{O(1)}$).To beat the general time/memory/data tradeoff attack, we recommend using $\alpha = 0.8$. Accordingly, the asymptotic time, memory and data complexities are $O(2^{0.556L})$, $O(L^2)$ and $O(2^{0.161L})$, respectively.

Note that the values listed in Table 2 are asymptotic. For $40 \leq L < 100$, the corresponding values are listed in Table 3.To beat the TMD attack with $40 \leq L < 100$, we recommend using $\alpha = 0.75$ or $\alpha = 0.8$. In both cases, the corresponding memory and data complexities are better than those of the TMD attack, while without a substantial compromise of the time complexity.

Table 2. The asymptotic time, memory and data complexities of algorithm A corresponding to different choices of α ($L \geq 100$).

α	β	Time	Memory	Data
0.5	0.99	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.007L})$
0.6	0.96	$O(2^{0.625L})$	$O(L^2)$	$O(2^{0.025L})$
0.75	0.80	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.114L})$
0.8	0.71	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$
0.9	0.46	$O(2^{0.526L})$	$O(L^2)$	$O(2^{0.284L})$
1.0	0.00	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.5L})$

Table 3. The time, memory and data complexities of algorithm A corresponding to different choices of α ($40 \leq L < 100$).

α	β	Time	Memory	Data
0.5	0.93	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.047L})$
0.6	0.88	$O(2^{0.625L})$	$O(L^2)$	$O(2^{0.075L})$
0.75	0.66	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.194L})$
0.8	0.57	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.239L})$
0.9	0.36	$O(2^{0.526L})$	$O(L^2)$	$O(2^{0.337L})$
1.0	0.00	$O(2^{0.5L})$	$O(L^2)$	$O(2^{0.5L})$

3 Comparisons and Experimental Results

We first present a detailed comparison with some other well-known attacks against the self-shrinking generator. Then, a number of experimental results are provided to verify the actual performance of the new attack. The advantages of our attack are pointed out at the end of this section.

3.1 Comparisons with Other Attacks

We mainly focus on the following attacks against the self-shrinking generator, i.e. the Mihaljević's attack in [26], the search tree attack in [31], the BDD-based attack in [18] and the time/memory/data tradeoff attack in [3]. Table 4 summarizes the corresponding results.

We can see from Table 4 that our attack achieves the best tradeoff between the time, memory, data and pre-computation complexities. More precisely, The attack in [26] suffers from the large amount of the keystream, which reaches $O(2^{0.5L})$ to obtain the best time complexity $O(2^{0.5L})$. Both the search tree attack in [31] and the BDD-based attack in [18] are unrealistic in terms of the memory requirement. In addition, the data complexity of our attack with $\alpha = 0.5$ are in the same order as those in [31] and [18] for the LFSR length L up to 2000. The two typical TMD attacks are derived according to the two points $T = N^{2/3}$, $M = D = N^{1/3}$ and $T = M = N^{1/2}$, $D = N^{1/4}$ on the curve $TM^2D^2 = N^2$ with pre-computation $P = N/D$, where T, M, D, N denote time, memory, data and search key space, respectively. Even regardless of the heavy pre-computation

Table 4. Asymptotic complexity comparisons with some other well-known attacks against the self-shrinking generator with the LFSR of length L.

Attack	Pre-computation	Time	Memory	Data
[26]A	-	$O(2^{0.5L})$	$O(L)$	$O(2^{0.5L})$
[26]B	-	$O(2^{0.75L})$	$O(L)$	$O(2^{0.25L})$
[31]	-	$O(2^{0.694L})$	$O(2^{0.694L})$	$O(L)$
[18]	-	$O(2^{0.656L})$	$O(2^{0.656L})$	$O(L)$
[3]A	$O(2^{0.75L})$	$O(2^{0.5L})$	$O(2^{0.5L})$	$O(2^{0.25L})$
[3]B	$O(2^{0.67L})$	$O(2^{0.67L})$	$O(2^{0.33L})$	$O(2^{0.33L})$
Ours ($\alpha = 0.5$)	-	$O(2^{0.667L})$	$O(L^2)$	$O(2^{0.007L})$
Ours ($\alpha = 0.75$)	-	$O(2^{0.571L})$	$O(L^2)$	$O(2^{0.114L})$
Ours ($\alpha = 0.8$)	-	$O(2^{0.556L})$	$O(L^2)$	$O(2^{0.161L})$

phase of the TMD attack, our attack with $\alpha = 0.8$ has much better memory and data complexity compared with the two TMD attacks, while without a substantial compromise of the real time complexity.

On the other hand, our attack can deal with different attack conditions and requirements smoothly due to the flexible choices of α. If only very short keystream and very limited disk space are available to the attacker, we still can launch a guess-and-determine attack successfully against the SSG with $\alpha \leq 0.6$. In this way, we avoid the large memory requirement of the two attacks in [31] and [18].

3.2 Experimental Results

We made a number of experimental results in C language on a Pentium 4 processor to check the actual performance of our attack.

Since the guess-and-determine attack in Section 2.2 has no restriction on the LFSR form, it has been implemented and tested many times for random chosen initial states and primitive polynomials of degree $10 \leq L \leq 50$ involved in the self-shrinking generator. For $10 \leq L \leq 40$, we use $\alpha = 0.6$ to mount the attack on the self-shrinking generator. For $40 < L \leq 50$, we use $\alpha = 0.8$. The results are rather satisfactory. The required keystream length are very close to the theoretical value in magnitude and the time complexity seems to be upper bounded by the theoretical value, which is just in expectation.

For example, let the LFSR's feedback polynomial be $f(x) = 1 + x^2 + x^{19} + x^{21} + x^{40}$, then the shift value is $x^{2^{39}} \bmod f^*(x) = x^{11} + x^{29} + x^{30}$, where $f^*(x)$ is the reciprocal polynomial of $f(x)$. For a random chosen initial state, our attack takes several minutes to recover the initial state or an equivalent state with success rate (see Table 3 and Corollary 1)

$$1 - (1 - 2 \cdot 2^{-(1-0.88)\cdot 40/(1+0.6)})^{(200-40)} > 0.99$$

from 200 bits keystream.

As a summary, our attack has at least the following advantages over the past relevant attacks against the self-shrinking generator:

– significantly smaller memory complexity with the time complexity quite close to $O(2^{0.5L})$.
– no pre-computation or if like (pre-compute $h(x)$), significantly smaller preprocessing time complexity without a compromise of the real attack complexity.
– flexibility to different attack conditions and requirements

These features guarantee that the proposed guess-and-determine attack can provide a better tradeoff between the time, memory and data complexities than all the previously known attacks against the self-shrinking generator. Especially, it compares favorably with the general time/memory/data tradeoff attack. Thus, our attack is the best answer known so far to a well-known open problem specified by the European STORK project.

4 Conclusions

In this paper, we proposed a new type of guess-and-determine attack on the self-shrinking generator. The new attack adapts well to different attack conditions and enables us to analyze the self-shrinking generator with the best tradeoff between the time, memory, data and pre-computation complexities known so far. So our result is the best answer to the corresponding open problem in STORK project known so far.

Acknowledgements. We would like to thank one of the anonymous reviewers for very helpful comments.

References

1. F. Armknecht, M. Krause, "Algebraic Attacks on Combiner with Memory", *Advances in Cryptology-Crypto'2003*, LNCS vol. 2729, Springer-Verlag, (2003), pp. 162-175.
2. F. Armknecht, "Improving Fast Algebraic Attacks", *Fast Software Encryption-FSE'2004*, LNCS, pp. 47-63, February 2004.
3. A. Biryukov, A. Shamir, "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers", *Advances in Cryptology-ASIACRYPT'2000*, LNCS vol. 1976, Springer-Verlag, (2000), pp. 1-13.
4. A. Canteaut, M. Trabbia, "Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5", *Advances in Cryptology-EUROCRYPT'2000*, LNCS vol. 1807, Springer-Verlag, (2000), pp. 573-588.
5. P. Chose, A. Joux, M. Mitton, "Fast Correlation Attacks: An Algorithmic Point of View", *Advances in Cryptology-EUROCRYPT'2002*, LNCS vol. 2332, Springer-Verlag, (2002), pp. 209-221.
6. S.R. Blackburn, "The linear complexity of the self-shrinking generator", *IEEE Transactions on Information Theory*, 45(6), September 1999, pp. 2073-2077.
7. D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", *Advances in Cryptology-Crypto'93*, LNCS vol. 773, Springer-Verlag, (1994), pp.22-39.

8. N. T. Courtois, "Fast Algebraic Attacks on Stream ciphers with Linear Feedback", *Advances in Cryptology-Crypto'2003*, LNCS vol. 2729, Springer-Verlag, (2003), pp. 176-194.
9. N. T. Courtois, W. Meier, "Algebraic Attacks on Stream ciphers with Linear Feedback", *Advances in Cryptology-EUROCRYPT'2003*, LNCS vol. 2656, Springer-Verlag, (2003), pp. 345-359.
10. P. Ekdahl, T.Johansson and W. Meier, "A note on the self-shrinking generator", *Proceeding of IEEE symposium on Information Theory*, (2003), pp. 166.
11. J. Dj. Golić, "Embedding and probabilistic correlation attacks on clock-controlled shift registers", *Advances in Cryptology-EUROCRYPT'94*, LNCS vol. 950, Springer-Verlag, (1994), pp. 230-243.
12. J. Dj. Golić, "Correlation analysis of the shrinking Generator", *Advances in Cryptology-Crypto'2001*, LNCS vol. 2139 Springer-Verlag, (2001), pp. 440-457.
13. S. W. Golomb, "*Shift Register Sequences*", Aegean Park Press, Laguna Hills(CA), Revised edition, 1982.
14. T. Johansson, "Reduced complexity correlation attacks on two clock-controlled generators", *Advances in Cryptology-ASIACRYPT'98*, LNCS vol. 1514, Springer-Verlag, (1998), pp. 342-357.
15. T. Johansson, F. Jonnson, "Improved fast correlation attack on stream ciphers via convolutional codes", *Advances in Cryptology-EUROCRYPT'1999*, LNCS vol. 1592, Springer-Verlag, (1999), pp. 347-362.
16. T. Johansson, F. Jönsson, "Fast correlation attacks through reconstruction of linear polynomials", *Advances in Cryptology-Crypto'2000*, LNCS vol. 1880, Springer-Verlag, (2000), pp. 300-315.
17. M. Krause and D.Stegemann, "Reducing the Space Complexity of BDD-based Attacks on Keystream Generators", *Fast Software Encryption-FSE'2006*, to appear.
18. M. Krause, "BDD-based cryptanalysis of keystream generators", *Advances in Cryptology-EUROCRYPT'2002*, LNCS vol. 2332, Springer-Verlag, (2002), pp. 222-237.
19. H. Krawczyk, "The shrinking generator: Some practical considerations", *Fast Software Encryption-FSE'94*, LNCS vol. 809, Springer-Verlag, (1994), pp. 45-46.
20. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
21. J. L. Massey, "Shift register synthesis and BCH decoding", *IEEE Transactions on Information Theory*, 15. 1969, pp. 122-127.
22. W. Meier, and O. Staffelbach, "The Self-Shrinking generator", *Advances in Cryptology-EUROCRYPT'1994*, LNCS vol. 950, Springer-Verlag, (1995), pp. 205-214.
23. W. Meier, O. Staffelbach, "Fast correlation attacks on certain stream ciphers", *Journal of Cryptology*, (1989) 1, pp. 159-176.
24. M. Mihaljević, P.C. Fossorier, H.Imai, "Fast correlation attack algorithm with list decoding and an application", *Fast Software Encryption-FSE'2001*, LNCS vol. 2355, Springer-Verlag, (2002), pp. 196-210.
25. M. Mihaljević, P.C. Fossorier, H.Imai, "A Low-complexity and high-performance algorithm for fast correlation attack", *Fast Software Encryption-FSE'2000*, LNCS vol. 1978, Springer-Verlag, (2001), pp. 196-212.
26. M. J. Mihaljević, "A faster cryptanalysis of the self-shrinking generator", *ACISP'96*, LNCS vol. 1172, Springer-Verlag, (1998), pp. 147-158.
27. L. Simpson, J. Dj. Golić, "A probabilistic correlation attack on the shrinking generator", *ACISP'98*, LNCS vol. 1438, Springer-Verlag, (1996), pp. 182-189.

28. I. Shparlinski, "On some properties of the shrinking generator", http://www.comp.mq.edu.au/ igor/Shrink.ps.
29. STORK project, http://www.stork.eu.org/documents/RUB-D6 − 2_1.pdf
30. Z. Wan, "*Geometry of Classical Groups over Finite Fields*", Science Press, New York, Second edition, 2002.
31. E. Zenner, M. Krause, S. Lucks, "Improved Cryptanalysis of the Self-Shrinking Generator", *Proc. ACISP'2001*, LNCS vol. 2119, Springer-Verlag, (2001), pp. 21-35.
32. K. Zeng, C. Yang, T. Rao, "On the linear consistency test (LCT) in cryptanalysis with applications", *Advances in Cryptology-Crypto'1989*, LNCS vol. 435, Springer-Verlag, (1989), pp. 164-174.