Security Analysis of the Generalized Self-shrinking Generator*

Bin Zhang[1,2], Hongjun Wu[1], Dengguo Feng[2], and Feng Bao[1]

[1] Institute for Infocomm Research, Singapore
[2] State Key Laboratory of Information Security,
Graduate School of the Chinese Academy of Sciences,
Beijing 100039, P.R. China
{stuzb,hongjun,baofeng}@i2r.a-star.edu.sg

Abstract. In this paper, we analyze the generalized self-shrinking generator newly proposed in [8]. Some properties of this generator are described and an equivalent definition is derived, after which two attacks are developed to evaluate its security. The first attack is an improved clock-guessing attack using short keystream with the filter function (vector G) known. The complexity of this attack is $O(2^{0.694n})$, where n is the length of the LFSR used in the generator. This attack shows that the generalized self-shrinking generator can not be more secure than the self-shrinking generator, although much more computations may be required by it. Our second attack is a fast correlation attack with the filter function (vector G) unknown. We can restore both the initial state of the LFSR with arbitrary weight feedback polynomial and the filter function (vector G) with complexity much lower than the exhaustive search. For example, for a generator with 61-stage LFSR, given a keystream segment of $2^{17.1}$ bits, the complexity is around 2^{56}, which is much lower than 2^{122}, the complexity of the exhaustive search.

Keywords: Stream cipher, Self-shrinking generator, Clock control, Fast correlation attack, Linear feedback shift register.

1 Introduction

The generalized self-shrinking generator is a simple keystream generator newly proposed in [8]. It uses one LFSR to generate a binary keystream. This new generator can be regarded as a specialization of shrinking generator and a generalization of self-shrinking generator. It is proved that the family of such generated keystream has good pseudorandomness in cryptographic sense [8]. However, it is still open whether such a generator can be used as a stream cipher or not. In this paper, we try to answer the open problem being proposed by the designers of the generalized self-shrinking generator [8].

The definition of the generalized self-shrinking generator is as follows:

* Supported by National Natural Science Foundation of China (Grant No. 60273027), National Key Foundation Research 973 project (Grant No. G1999035802) and National Science Fund for Distinguished Young Scholars (Grant No. 60025205).

J. López, S. Qing, and E. Okamoto (Eds.): ICICS 2004, LNCS 3269, pp. 388–400, 2004.

Definition 1. *([8]) Let* $a = \cdots, a_{-2}, a_{-1}, a_0, a_1, a_2, \cdots$ *be an m-sequence over* $GF(2)$, $G = (g_0, g_1, \cdots, g_{n-1}) \in GF(2)^n$. *Construct sequence* $v = \cdots, v_{-2}, v_{-1}, v_0, v_1, v_2, \cdots$ *such that* $v_k = g_0 a_k + g_1 a_{k-1} + \cdots + g_{n-1} a_{k-n+1}$, *for each* k. *If* $a_k = 1$, *output* v_k, *otherwise discard* v_k, *thus we get a generalized self-shrinking keystream denoted by* $b(G) = b_0, b_1, b_2, \cdots$. *The keystream family* $B(a) = \{b(G), G \in GF(2)^n\}$ *is called the family of generalized self-shrinking keystream sequences based on m-sequence* a.

To evaluate the security of the generalized self-shrinking generator, we first describe some properties of this generator and give an equivalent definition which is suitable for hardware implementation. Based on these properties and the equivalent definition, we propose two attacks. One is an improved clock-guessing attack assuming that the vector G is known. This attack generalizes the original version in [13] by making it applicable to the linear combination case. Comparison with the general time/memory/data tradeoff attack shows our attack has its advantages. In addition, we point out that for some special cases of this generator, there are more efficient attacks. In the case that the vector G is unknown to the attacker, we present a fast correlation attack that could recover both the initial state of the LFSR and the vector G.

This paper is organized as follows. In Section 2, we analyze some properties of the generalized self-shrinking generator and give an equivalent definition. The algorithm given in Section 3 deals with the case that the vector G is known to the cryptanalyst. The discussions on some special insecure cases of the generator are also presented. In Section 4, a novel fast correlation attack is developed for the generalized self-shrinking with the vector G unknown. Section 5 concludes this paper.

2 Some Properties of the Generalized Self-shrinking Generator

In this section, we will describe some properties of the generalized self-shrinking generator. First, by investigating the general structure of this generator, an unified upper bound of the linear complexity of each keystream belonging to the family $B(a)$ is derived. Then an equivalent definition of this generator is obtained based on a long division algorithm.

From Definition 1, it is straightforward to obtain the following lemma.

Lemma 1. *For each sequence* v *defined by vector* G *in Definition 1, there exists an integer* τ *such that* $v_k = a_{k+\tau}$ *holds, for each* k.

From this lemma, we know that the sequence v is just a shifted equivalent version of a. The vector G plays the role of a controller in determining the exact shift value. There is a one-to-one mapping between the shift values and the vector Gs. Hence, we have the following theorem.

Theorem 1. *Keeping the notations as above, the linear complexity of a generalized self-shrinking keystream is at most* $2^{n-1} - (n-2)$.

The proof of this theorem is omitted here due to the lack of space. It is available in the full version of this paper. The direct consequence of this theorem is that to resist the Berlekamp-Massey algorithm based attack, the length of the LFSR used in the generalized self-shrinking generator should better be larger than 40.

According to Definition 1, v_k is determined by the state $(a_{k-n+1}, \cdots, a_{k-1}, a_k)$ of sequence a and vector G. It is troublesome to generate an element of sequence v at the initial moment, since we have to store an $(n-1)$-step earlier state of a. In the following, we will present a linear polynomial time algorithm which is a recursive long division of polynomials to represent the kth term of sequence v as a linear combination of a different basis of the sequence a.

Denote by X the *left shift operator* on m-sequence a, i.e. $X\{a_k\} = \{a_{k+1}\}$ for each k. Let $f(x) = 1 + c_1 x + c_2 x^2 + \cdots + c_{n-1}x^{n-1} + x^n$ be the feedback polynomial of sequence a. It is easy to see that for each k

$$a_{n+k} = \sum_{i=1}^{n} c_i a_{n+k-i} = \sum_{i=0}^{n-1} c_i^* a_{i+k} \tag{1}$$

holds where c_i^* denote the coefficients of the reciprocal polynomial $f^*(x)$ of $f(x)$, i.e. $c_i^* = c_{n-i}$ with $c_n = c_0 = 1$. Moreover, we have $f^*(X)\{a\} = \sum c_i^* X^i\{a\} = 0$ where 0 is the all-zero sequence.

Keeping in mind that $\{1, X, X^2, \cdots, X^{2^n-2}\}$ is a cyclic group, $X^{2^n-1-i} = X^{-i}$ holds. Sequence v can be rewritten as: $\{v_k\} = g_0\{a_k\} + g_1 X^{-1}\{a_k\} + \cdots + g_{n-1}X^{-n+1}\{a_k\}$. From Lemma 1, there exists an integer τ such that $\{v_k\} = X^\tau\{a_k\}$, for each vector G. In terms of feedback polynomial, the following two congruence of polynomials hold:

$$g_0 + g_1 X^{-1} + \cdots + g_{n-1}X^{-n+1} \equiv X^\tau \bmod f^*(x), \tag{2}$$

$$X^\tau \equiv g'_0 + g'_1 X^1 + \cdots + g'_{n-1}X^{n-1} \bmod f^*(x), \tag{3}$$

where both vectors $(g_0, g_1, \cdots, g_{n-1})$ and $(g'_0, g'_1, \cdots, g'_{n-1})$ belong to $GF(2)^n$. (2) and (3) indicate that there exists a vector $(g'_0, g'_1, \cdots, g'_{n-1}) \in GF(2)^n$ such that $\{v_k\} = g'_0\{a_k\} + g'_1 X^1\{a_k\} + \cdots + g'_{n-1}X^{n-1}\{a_k\}$, i.e. $v_k = g'_0 a_k + g'_1 a_{k+1} + \cdots + g'_{n-1}a_{k+n-1}$ holds. Furthermore, the above process is obviously invertible which implies that we actually get an equivalent definition of the generalized self-shrinking generator.

Definition 2. *Let $a = a_0, a_1, a_2, \cdots$ be an m-sequence over $GF(2)$, vector $G' = (g'_0, g'_1, \cdots, g'_{n-1}) \in GF(2)^n$. Construct sequence $v = v_0, v_1, v_2, \cdots$ such that $v_k = g'_0 a_k + g'_1 a_{k+1} + \cdots + g'_{n-1}a_{k+n-1}$, for each k, as shown in Figure 1. If $a_k = 1$, output v_k, otherwise discard v_k, we also get a generalized self-shrinking keystream denoted by $b(G') = b'_0, b'_1, b'_2, \cdots$. The keystream family $B(a) = \{b(G'), G' \in GF(2)^n\}$ is also called the family of generalized self-shrinking keystream sequences based on m-sequence a.*

Although Lemma 1 reveals that there exists a τ satisfying $\{v_k\} = X^\tau\{a_k\}$, it is of great importance to note that when transforming from one definition to the other, it is unnecessary to find out the value of τ. In the following, we will

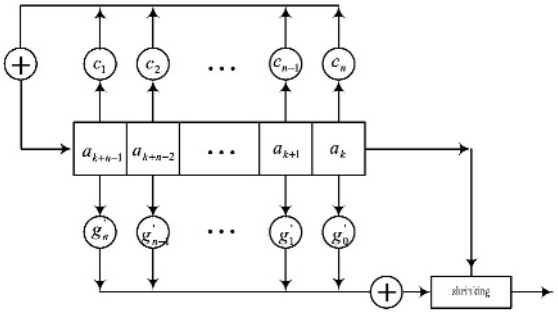

Fig. 1. Equivalent definition of the generalized self-shrinking generator

develop a long division algorithm to fulfill this task without knowing τ. First consider the following example.

Example 1. Let $f(x) = 1 + x^6 + x^7$ be the feedback polynomial of sequence a, its reciprocal polynomial is $f^*(x) = 1 + x + x^7$. Choose the vector G to be $(1, 0, 0, 1, 0, 0, 1)$. From Definition 1, we have $v_k = a_k + a_{k-3} + a_{k-6}$. We use the following long division algorithm to get vector G'.

$$
\begin{array}{r}
 \\
1 + x + x^7 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \\
\end{array}
\begin{array}{l}
x^{-6} + x^{-5} + x^{-4} \\
\hline
) \; x^{-6} + x^{-3} + 1 \\
\quad x^{-6} + x^{-5} + x \\
\hline
\quad x^{-5} + x^{-3} + 1 + x \\
\quad x^{-5} + x^{-4} + x^2 \\
\hline
\quad x^{-4} + x^{-3} + 1 + x + x^2 \\
\quad x^{-4} + x^{-3} + x^3 \\
\hline
\quad 1 + x + x^2 + x^3
\end{array}
\quad .
$$

From the last remainder, we know that $G' = (1, 1, 1, 1, 0, 0, 0)$, i.e. $v_k = a_k + a_{k+1} + a_{k+2} + a_{k+3}$.

When transforming from Definition 2 to Definition 1, a similar long division algorithm can be used. For space limitation, we omit it here.

In general, let

$$
\begin{aligned}
G_0(x) &= G(x) = g_{n-1}x^{-(n-1)} + g_{n-2}x^{-(n-2)} + \cdots + g_1x^{-1} + g_0 \\
G_i(x) &= g_{n-i}x^{-(n-i)}f^*(x) + G_{i-1}(x) \;\text{ i.e. }\; G_i(x) \equiv G_{i-1}(x) \bmod f^*(x).
\end{aligned}
$$

It is obvious that $G(x) \equiv G_i(x) \bmod f^*(x)$ holds. Associated with the equation (2) and (3), this fact implies that after finitely many steps, equation (3) will be ultimately reached. Since there are some g_i may take 0, it is unnecessary to take the above procedures step-by-step, as shown in the above toy example. In nature, this is a long division algorithm which can be carried out recursively.

Complexity of the Algorithm. Noting that the recursive procedures will end whenever the remainder $G_i(x)$ is a polynomial with all its monomials possessing

degrees from 0 to $n-1$, the complexity of the long division algorithm is $O(n)$, i.e. linear polynomial time complexity.

The advantage of Definition 2 over Definition 1 mainly lies in the convenience of hardware implementation. Besides, it facilitates the instant verification of linear dependency in the following attack in section 3.2.

The following theorem shows a weakness in the design of the generalized self-shrinking generator.

Theorem 2. *Given the initial state of sequence v, we can efficiently recover both the initial state of a and the vector G even for LFSR of length up to* 128.

Proof. Since we have v in possession, from Definition 2, we get

$$\begin{cases} v_0 = g'_0a_0 + g'_1a_1 + \cdots + g'_{n-1}a_{n-1} \\ v_1 = g'_0a_1 + g'_1a_2 + \cdots + g'_{n-1}a_n \\ \qquad\vdots \\ v_{n-1} = g'_0a_{n-1} + g'_1a_n + \cdots + g'_{n-1}a_{2n-2}\ . \\ \qquad\vdots \end{cases} \tag{4}$$

This is a system of $2n$-variable equations of degree 2, which is very vulnerable to algebraic attack [6, 7, 1]. In fact, to restore the $2n$ variables $(a_0, a_1, \cdots, a_{n-1})$ and $(g'_0, g'_1, \cdots, g'_{n-1})$, all we have to do is to solve a linear system of $T = \binom{2n}{2} = n \cdot (2n-1)$ variables by Linearization method, noting $a_i^2 = a_i$ and $(g'_i)^2 = g'_i$. Given $m = T$ keystream bits, we can solve this linear system by Gaussian reduction taking $7 \cdot T^{\log_2 7}/64$ CPU clocks. For $n = 40$, it amounts to about 2^{30} CPU clock cycles which takes only about 1 second on a Pentium 4 PC. For $n = 100$, 2^{37} CPU clock cycles. For $n = 128$, 2^{39} CPU clock cycles. This completes the proof.

3 An Improved Clock-Guessing Attack with the Vector G Known

Now we are ready to present our attack on the generalized self-shrinking generator with the vector G known. Note that amongst the vectors $G \in GF(2)^n$, the four trivial vectors $(0, 0, \cdots, 0), (1, 0, \cdot\cdot, 0), (0, 1, \cdots, 1), (1, 1, \cdots, 1)$ which result in keystreams with periods of length 1 or 2 should be avoided when implementing the generator in practice. We first consider some special cases, after which our general attack is presented.

3.1 Some Special Cases

It is easy to see that $X^{2^{n-1}-1}\{a_1, a_3, \cdots\} = \{a_0, a_2, \cdots\}$, for m-sequence a. Since $\gcd(2, 2^{n-1}-1) = 1$, $2i \pmod{2^n-1}$ go through every element of $\{0, 1, \cdots, 2^n-2\}$. Both $\{a_0, a_2, \cdots\}$ and $\{a_1, a_3, \cdots\}$ are shift equivalence to a. This fact implies that the vector G corresponding to the shift value $2^{n-1}-1$ actually defines a

special case of the generator in which the sequence a and v can be combined in such a way as $\{v_0, a_0, v_1, a_1, \cdots\}$, so that the resulting sequence can reproduce the keystream generated by a and v in a self-shrinking manner. Therefore, the original clock-guessing attack in [13] and the classical time/memory/data tradeoff attack can be applied to the resulting sequence, for the BSW sampling in [2] can be easily determined to be $2^{-n/4}$. However, in the general case, it appears to be difficult to determine the BSW sampling [2] of the generalized self-shrinking generator. Except the expensive trial and error method, it seems unfeasible to efficiently enumerate all the special states even with the algebraic attack techniques.

Next, consider using vector $G' = (0, 1, 0, \cdots, 0)$ in the generator . In this case, every run in the keystream having the pattern $\{1, 1, \cdots, 1, 0\}$ reveals that the corresponding elements in a are the pattern $\{1, 1, \cdots, 1, 0\}$, which leaks enough information for us to recover the corresponding initial state. Some similar cases are the vectors $(0, 0, 1, 0, \cdots, 0)$, $(0, 1, 1, 0, \cdots, 0)$ and so on. In all these cases, some special patterns such as $\{1, 1, \cdots, 1\}$, $\{0, 1, \cdots, 1\}$, ..., in the keystream always leak too much information, implying that for real applications, the vector G used in this generator must be carefully chosen.

3.2 The General Attack

Instead of examining the vectors one-by-one as above, we propose a general attack to evaluate the security of the generalized self-shrinking generator with G known. This attack is an improved version of the clock-guessing attack in [13]. We generalize it to the linear combination case. The attack process can be

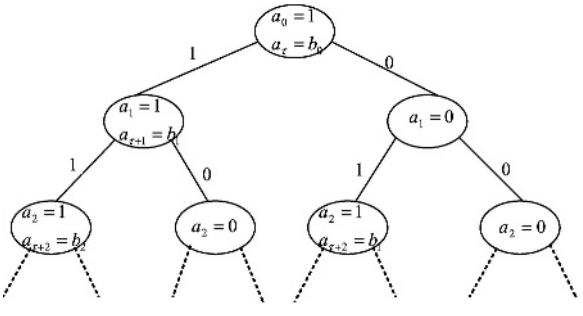

Fig. 2. The Modified Guess Tree

represented by a tree as shown in Figure 2. The development of the modified guess tree is as follows: at the initial moment, we always have $a_0 = 1, a_\tau = b_0$ as we only want to recover the equivalent state that generates the same keystream as the true one. This is represented by the root of the tree. From then on, on guessing one more bit of a, we obtain two different types of linear equations:

1. A linear equation $a_i = 1$ or 0 follows every guess denoted by type 1.
2. If $a_i = 1$, get a second type equation $a_{\tau+i} = b_j$, where the value of j is dependent on the path we choose.

Our aim is to have n independent linear equations to restore the initial state. During the growth process of the guess tree, we may encounter two cases except the linearly independent case: the new equation may be linear consistency with the old ones or contradict to the old ones. Whenever we meet a contradiction, ignore the current branch and go backtracking. As soon as we get n linearly independent equations, stop the growth and solve the equation system and derive a candidate key. Then test this key by running the generator with this initial value, if the candidate keystream matches the known segment of keystream, we accept it.

We must stress here that we simply write $a_\tau, a_{\tau+1}, \cdots$ only for the sake of simplicity and limited space in the figure. Actually, we need not find out this τ. What we have to do is just using the long division algorithm discussed in Section 2 to obtain an equivalent vector $G' = (g'_0, \cdots, g'_{n-1})$ satisfying equation (3) which will facilitate the instant verification of the linear relationship of the linear equation system.

From the attacker's point of view, the vector G is of great importance because it will make the whole attack work. Without the knowledge of G, we will not be able to determine whether or not the newly added equation in Figure 2 is linearly dependent upon known linear equations. If there exists a method by which we can determine the linear relationship among the linear equations with the vector G unknown, a similar clock-guessing attack using this method can be developed with the vector G unknown. However, we did not find such a method.

As in the original attack [13], we need the notions of well-formed and malformed tree. First label the nodes in the guess tree as follows: each node is labeled by a number of linearly independent equations still needed to solve the equation system. T_l denotes a guess tree that l linearly independent equations are still needed in the root to solve the equation system. Note that in Figure 2, $l = n-2$. When a leaf of the tree takes the label 0 or -1, the growth stops.

Definition 3. *([13]) A well-formed tree T_l^* is a binary tree such that for every node that is not a leaf, the following holds: If the label of the node is j, then the label of its left child is $j-2$ and the label of its right child is $j-1$. A malformed tree is an arbitrary guess tree that does not satisfy the above condition.*

Lemma 2. *([13]) Let C_l^* denotes the number of leaves of a well-formed guess tree T_l^*, C_l denotes the maximum number of leaves in a guess tree that may or may not be malformed. Then $C_l \leq C_l^* \leq 2^{0.694l+0.306}$.*

The following theorem gives the time complexity of the improved backtracking algorithm.

Theorem 3. *The total asymptotic running time of the above attack is $O(n^4 \cdot 2^{0.694n})$, that is the same as that on the self-shrinking generator.*

Proof. Notations are kept as above. Note that in Figure 2, for a node of depth i, we have $i+1$ equations of type 1, which means at depth $n-1$, we have exactly n type 1 equations represented using variables $a_0, a_1, \cdots, a_{n-1}$ only. Since these equations must be linearly independent according to the definition of type 1, we already have n linearly independent equations of type 1 in each node of depth $n-1$, which can be used to solve the linear equations system to get a candidate key. We need not to develop the guess tree any more.

Let N_n denote the number of nodes in the modified guess tree and C_n denote the number of leaves in such a guess tree. Taking malformed branches into consideration, the following certainly holds: (the depth of the guess tree is at most n and there are C_n leaves)

$$N_n \leq n \cdot C_n. \tag{5}$$

By Lemma 2, we have $N_n \in O(n \cdot 2^{0.694n})$. Since the operation of testing the linear dependency of new equations has complexity $O(n^3)$, we conclude that the total asymptotic running time of the above attack is $O(n^4 \cdot 2^{0.694n})$. This completes the proof.

Note that our improved algorithm reserves the parallel feature of the original one, which implies that k processors can reduce the complexity by a factor of k. Besides, the following table shows that the generalized attack is comparable to the classical time/memory/data tradeoff attack [2]. We assume a LFSR of length 61 for the generalized self-shrinking generator, as suggested for the shrinking generator in [9].

Table 1. Rough comparison of attacks on the example generalized self-shrinking generator

	Pre-pro.	Time	Memory	Data length
Our attack	0	2^{42}	2^{42}	≈ 61
[2]	2^{41}	$2^{41} - 2^{61}$	2^{20}	2^{20}

In Table 1, we choose the point $P = T = N^{2/3}$, $M = D = N^{1/3}$ on the tradeoff curve $TM^2D^2 = N^2$ $(D^2 \leq T \leq N)$ where N is the size of key space, P is the pre-processing time, T is the attack time, M is the random access memory available to the attacker, and D is the data. From this table, we can see that our attack is better than that in [2] in two aspects: small amount of required keystream and no pre-processing, while at the cost of larger amount of memory. Note that the pre-processing stage of the tradeoff attack on the generalized self-shrinking generator is even more time-consuming, for it is difficult to use the BSW sampling [2] in this case. In addition, it is known that the BDD-based cryptanalysis [10] has a little better bound of $2^{0.656n}$, however, the BDD-based cryptanalysis is too memory consumptive compared with the above attack. From above, we suggest that when the vector G is open, the key length should exceed 100 bits.

4 A Fast Correlation Attack with the Vector G Unknown

In this section, we present a novel fast correlation attack on the generalized self-shrinking generator without the knowledge of G. From Theorem 2, in order to crack the whole system, we only need to recover the sequence v, i.e. restore the initial state of the LFSR generating v.

Actually, our attack exploits a carefully detected correlation between v and a new sequence $\hat{v}$ constructed from the keystream $b(G)$. Then a one-pass fast correlation attack is applied to sequence $\hat{v}$ to recover its initial state. A similar attack is proposed in another paper to attack the shrinking generator. Though there is some doubts that in the case of the generalized self-shrinking generator, sequence a and v may not be statistically independent, our experimental results do conform the validity of our attack.

4.1 Construction Stage

For simplicity, we assume that both sequence a and v are comprised of independent uniformly distributed random variables. Consider the probability that b_k equals v_r $(k \leq r)$. If we regard the event that $a_i = 1$ as success, then the event that b_k equals v_r is equivalent to the event that the kth success of sequence a occurs at the rth trial. Therefore the probability that b_k equals v_r is: $P(b_k = v_r) = \binom{r}{k}(\frac{1}{2})^{r+1}$. On the other hand, if v_r appears in the keystream, we have $v_r = b_{\sum_{i=0}^{r-1} a_i}$. When r grows large, the distribution of the sum $\sum_{i=0}^{r-1} a_i$ can be approximated by the Normal Distribution, i.e. $\sum_{i=0}^{r-1} a_i \mapsto N(r/2, \sqrt{r/4})$.

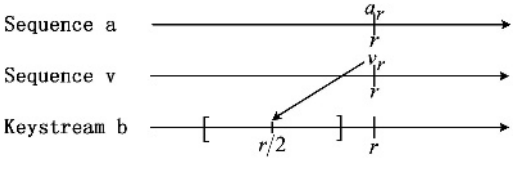

Fig. 3. The interval that v_r probably lies in

For arbitrary probability p, there exists a α such that whenever v_r appears in the keystream $b(G)$, the following equation holds:

$$P(\sum_{i=0}^{r-1} a_i \in I_{r/2}) = p, \tag{6}$$

where $I_{r/2} = [r/2 - \alpha\sqrt{r/4}, r/2 + \alpha\sqrt{r/4}]$. Without loss of generality, we assume the interval $I_{r/2}$ includes odd number of integers. We formally define the intuitive notion of imbalance as follows.

Definition 4. *Let* $A_0 = \{b_i | i \in I_{r/2}, b_i = 0\}$, $A_1 = \{b_i | i \in I_{r/2}, b_i = 1\}$, *the imbalance of the interval* $I_{r/2}$, $Imb(I_{r/2})$, *is defined as* $|A_1| - |A_0|$ *where* $|\cdot|$ *is the cardinality of a set. If* $Imb(I_{r/2}) \neq 0$, *this interval is said to be imbalanced. See Figure 3.*

From the above imbalance, we make a straightforward majority poll to construct a new sequence $\hat{v}$.

Construction Method. Following Definition 4, if $Imb(I_{r/2}) > 0$, let $\hat{v}_r = 1$. Otherwise, let $\hat{v}_r = 0$.

Although sequence a and v share the same feedback polynomial, experimental results do show that sequence $\hat{v}$ constructed as above satisfying $P(\hat{v}_i = v_i) = \frac{1}{2} + \varepsilon$ with $\varepsilon > 0$ as expected whether the initial states of a and v are chosen randomly or not. Hence it is safe to assume that sequence a and v are statistically independent purely random sources. The following theorem confirms the statement above precisely.

Theorem 4. *There is a correlation weakness between sequence* v *and* $\hat{v}$ *which is given by*

$$P(\hat{v}_r = v_r) = \frac{1}{2} + \frac{1}{2^{2e}}\binom{2e}{e}\frac{p}{4} = \frac{1}{2} + \varepsilon_r. \tag{7}$$

where $2e+1$ *satisfying* $e = \lfloor(\alpha\sqrt{r} - 1)/2\rfloor$, *is the closest odd integer to* $\alpha\sqrt{r}$ *and* $p = \frac{1}{\sqrt{2\pi}}\int_{-\alpha}^{\alpha} e^{-x^2/2}dx$ *is the probability in* (6).

The proof of this theorem is omitted here due to space limitations. It is available in the extended version of this paper.

First note that $0.5 < P(\hat{v}_r = v_r) \leq 0.75$, where the upper bound is achieved when $r = 0$. Theorem 4 is in accordance with our assumption before. Besides, experimental results confirm that in the generalized self-shrinking case, the correlation weakness stated in Theorem 4 actually exists. In order to maximize the bias ε_r, we use Mathematica to find out the optimum values of α resulting in the maximums of ε_r, for the expression function $\varepsilon_r = \frac{1}{2^{2e}}\binom{2e}{e}\frac{p}{4}$ is an irregular function, the classical methods to search for the extreme value fail in this case. We use the following two instructions: Findminimum$[-\frac{\binom{2e}{e}}{2^{2e}}\frac{\int_{-a}^{a} e^{-x^2/2}dx}{4\sqrt{2\pi}}, \{a, 0, 5\}]$, for $0 \leq r \leq 243$ or Findminimum$[-\frac{\binom{2e}{e}}{2^{2e}}\frac{\int_{-a}^{a} e^{-x^2/2}dx}{4\sqrt{2\pi}}, \{a, 1, 5\}]$, for $r \geq 244$. Table 2 shows the results of the search for $N = 80000$ keystream bits. We get these values on a Pentium 4 PC in about three hours. The average value of α is 1.36868. Table 2 shows that the optimum values mainly lie in the interval $(1.3, 1.5)$. It is of great importance to note that the pre-computation of the optimum values of α would be applicable to arbitrary LFSR due to the random assumption. With these optimum values, we can construct sequence $\hat{v}$ possessing good enough correlation

Table 2. The distribution of the optimum values of α

Domain	$(1.0, 1.1)$	$(1.1, 1.2)$	$(1.2, 1.3)$	$(1.3, 1.4)$	$(1.4, 1.5)$	$(1.5, 1.6)$	others
No.	248	3139	4308	40386	31315	365	239

to sequence v. Table 3 shows the average biases found with these optimum α both in theory and in experiments.

Table 3. The average biases found with the optimum values of α

N	400	4000	40000	80000	140000
ε(theory)	0.0541859	0.0252652	0.0135484	0.0113329	0.00982376
ε(found)	0.04500	0.0205	0.013200	0.012150	0.008793

It is obvious that the correlations do exist. The practical average values of ε are found according to: $0.5+\varepsilon =$ (number of coincidence bits between $\hat{v}$ and v)/ N. The actual values of ε in Table 3 are found based on a generalized self-shrinking generator with the following primitive polynomial as the feedback polynomial of the LFSR: $f_A(x) = 1 + x + x^3 + x^5 + x^9 + x^{11} + x^{12} + x^{17} + x^{19} + x^{21} + x^{25} + x^{27} + x^{29} + x^{32} + x^{33} + x^{38} + x^{40}$ [4, 12, 11].

4.2 Attack Stage

Based on the correlations found in section 4.1, we use the one-pass correlation attack [4] to recover the initial state of v from $\hat{v}$. Actually, any fast correlation attack can be applied to $\hat{v}$. Since the attack in [4] is the most efficient so far as we know, we follow it in our attack.

As other fast correlation attacks, it also consists of two stages: pre-processing stage for the construction of a large number of appropriate parity-check equations and processing stage in which a majority poll is conducted for each bit under consideration. Precisely speaking, a partial exhaustive search is taken over the first B bits of the initial state of a length-n LFSR and make a majority poll for each of D bits including other $n-B$ bits of the initial state, hoping at least $n-B$ bits can be correctly recovered. In the following, we will give a brief review of the attack, for the details of the formulae and the notations, please see the Appendix A.

The parity-check equations used in this attack are of the form: $x_i = x_{m_1} \oplus \ldots \oplus x_{m_{k-1}} \oplus \sum_{j=0}^{B-1} c_j x_j$ where m_j $(1 \le j \le k-1)$ denote arbitrary indices of the keystream bits z_i and the last sum represents a partial exhaustive search over $(x_0, \cdots, x_{B-1})$. At the processing stage, after regrouping the parity-check equations that contain the same pattern of $B - B_1$ initial bits, using Walsh transform to evaluate the parity-check equations for a given bit, i.e. when $\omega = [x_{B_1}, x_{B_1+1}, \cdots, x_{B-1}]$, $F_i(\omega) = \sum(-1)^{t_i^1 \oplus t_i^2}$ is just the difference between the number of predicted 0 and the number of predicted 1, where $t_i^1 = z_{m_1} \oplus \cdots \oplus z_{m_{k-1}} \oplus \sum_{j=0}^{B_1-1} c_j x_j$ and $t_i^2 = \sum_{j=B_1}^{B-1} c_j x_j$. Then for each of the D considered bits, if $F_i(\omega) > \theta$, let $x_i = 0$. If $F_i(\omega) < -\theta$, let $x_i = 1$, where θ is the decision threshold. In order to have at least $n - B$ correctly recovered bits, a check procedure is used which requires an exhaustive search on all subsets of size $n - B$ among $n - B + \delta$ recovered bits.

Now we use the above attack to analyze a generalized self-shrinking generator with a 61-stage LFSR. We choose the parameters as follows: bias is 0.008793, $D = 36, k = 5, \delta = 3, B = 46, N = 140000 \approx 2^{17.1}$. According to the formulae in Appendix A, the pre-processing time for constructing parity-check equations of weight 5 is $O(2^{43})$, the success probability is 99.9% and the total complexity of the processing stage is $O(2^{56})$. From Theorem 2, for $n = 61$, the complexity of recovering the initial state of a and the vector G is negligible compared to above complexity, so the overall complexity is also $O(2^{56})$.

Comparing this correlation attack with the improved clock-guessing attack, we can see that the knowledge of G facilitates the cryptanalysis of the generator. For a generator with a 61-stage LFSR, when we know the vector G, the complexity of the attack in section 3.2 is $O(2^{42})$ with no pre-processing; while without knowing G, the complexity of the second attack is $O(2^{56})$ with $O(2^{43})$ pre-processing.

5 Conclusion

In this paper, we analyze the security of the generalized self-shrinking generator. Some properties and weaknesses of this generator are pointed out and an equivalent definition suitable for hardware implementation is derived. The two attacks presented in this paper show that it is necessary to keep the vector G secret, for the generalized self-shrinking generator actually does not provide higher security than the self-shrinking generator with the vector G open.

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. 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.
3. S. R. Blackburn, "The linear complexity of the self-shrinking generator", *IEEE Transactions on Information Theory*, vol. 45, no. 6, 1999, pp.2073-2077.
4. 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.
5. D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", *Advances in Cryptology-Crypto'93*, LNCS vol. 773, Springer-Verlag,(1994), pp.22-39.
6. 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.
7. 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.
8. Y. P. Hu, G. Z. Xiao, "Generalized Self-Shrinking Generator", *IEEE Transactions on Information Theory*, Vol. 50, No. 4, pp. 714-719, April 2004.

9. H. Krawczyk, "The shrinking generator: Some practical considerations", *Fast Software Encryption-FSE'94*, LNCS vol. 809, Springer-Verlag,(1994), pp. 45-46.
10. M. Krause, "BDD-based Cryptanalysis of Keystream generators", *Advances in Cryptology-EUROCRYPT'2002*, LNCS vol. 2332, Springer-Verlag,(2002), pp. 222-237.
11. 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.
12. 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.
13. 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.

A Notations and Formulae of a One-Pass Fast Correlation Attack

1. $P(z_i = x_i) = \frac{1}{2}(1+\varepsilon)$, z_i denotes the keystream bit.
2. N is the length of the keystream.
3. n is the length of the LFSR.
4. B is the number of bits partially exhausitive searched.
5. D is the number of bits under consideration.
6. k is the weight of the parity-check equations.
7. $q = \frac{1}{2}(1+\varepsilon^{k-1})$ is the probability that one parity-check equation yielding the correct prediction.
8. Ω is the expected number of weight k parity-check equations for each considered bit.
9. δ is the number of bits that predicted other than the $n-B$ bits.
10. $P_1 = \sum_{j=\Omega-t}^{\Omega}(1-q)^{\Omega-j}q^j\binom{\Omega}{j}$ is the probability that at least $\Omega - t$ parity-check equations give the correct result, where t is the smallest integer satisfying $D \cdot P_1 \geq n - B + \delta$.
11. θ is the threshold such that $\theta = \Omega - 2t$.
12. $P_2 = \sum_{j=\Omega-t}^{\Omega}(1-q)^{j}q^{\Omega-j}\binom{\Omega}{j}$ is the probability that at least $\Omega - t$ parity-check equations give the wrong result.
13. $P_v = P_1/(P_1+P_2)$ is the probability that a bit is correctly predicted with at least $\Omega - t$ parity-check equations give the same prediction.
14. $P_{succ} = \sum_{j=0}^{\delta}\binom{n-B+\delta}{j}P_v^{n-B+\delta-j}(1-P_v)^j$ is the probability that at most δ bits are wrong among the $n - B + \delta$ predicted bits.
15. $E = \frac{1}{2^{\Omega-1}}\sum_{j=\Omega-t}^{\Omega}\binom{\Omega}{j}$ is the probability that a wrong guess yields at least $\Omega - t$ identical predictions for a given bit.
16. $P_{err} = \sum_{j=n-B+\delta}^{D}\binom{D}{j}E^j(1-E)^{D-j}$ is the probability that false alarm occurs.
17. $O(2^B D\log_2\Omega + (1+P_{err}(2^B-1))\binom{n-B+\delta}{\delta}\frac{1}{\varepsilon^2})$ is the total complexity of the processing stage.
18. When $k = 5$, the time complexity of pre-processing stage is $O(DN^2\log N)$. The memory complexity is $O(N)$.