The Editing Generator and Its Cryptanalysis

Guang Gong and Shaoquan Jiang

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
Email:{ggong, sqjiang}@calliope.uwaterloo.ca

Abstract. In this paper, we present a new pseudo-random sequence generator, constructed by using two ternary linear feedback shift registers (LFSR). This new generator is called an *editing generator* which is a combined model of the clock-control generator (viewed as insertion) and the shrinking generator (viewed as deletion). It is shown that the editing generator has good properties of randomness, such as large periods, high linear spans, large ratio of linear span per symbol, and small unbias of occurrences of symbols. Three different attacks for recovering the initial states of the two LFSRs, given that a portion of a key stream is exposed, are analyzed. They are the brute-force like attack, the unconstraint embedding attack (analogous to the binary case), and the entropy attack. The analysis shows that the edited sequences resist to all of these three attacks.

Index words: Pseudo-random sequences, LFSR, cryptanalysis, period, linear span, distribution, random variable, entropy.

1 Introduction

Pseudo-random sequences are commonly used in cryptology. It is most often used as either key streams in stream cipher systems or as pseudo-random number generators. Pseudo-random sequence generators based on linear feedback shift registers (LFSR) are most common structures in practice due to their efficient hardware implementation. We can classify their structures into two classes. One class is to apply a boolean function in n variables to a set of n LFSRs. The resulting generator is called a combinatorial function generator, if not all LFSRs are different. It is called a filtering generator when all LFSR are the same. For the theory and the properties of randomness on those generators, please see Rueppel's survey paper [12]. The other class is to use one LFSR to control outputs of another LFSR. There are two different control models. One is the clock-control generators [1][9] (stop/go or Gunther's generalized form [7]), and the other is the shrinking generators [2], including self-shrinking generators [10]. There are extensive researches on attacks on the clock-control generators and/or the shrinking generators, here we list some of them [3] [5][15] [4].

In this paper, we will present a new type of pseudo-random sequence generators, constructed by using two ternary LFSRs. We call the resulting generator an *editing generator*. This is a combined model of the clock-control generator (viewed as insertion) and the shrinking generator (viewed as deletion). The resulting sequences are ternary sequences. We will show that the editing generators have good properties of randomness. For example, they have large periods, high linear spans, large ratio of linear span per symbol, and small unbias of occurrences of symbols. Although these properties may not guarantee security of the editing generators for cryptographic applications, it is necessary to resist to the attacks from the application of the Berlekamp-Massey algorithm. Following satisfaction in these results on the new generator, we will look at its security under a randomized model. We will consider three different attacks for recovering the initial states of two LFSRs, given that a portion of a key stream is exposed. The first one is the brute-force like attack in search of the initial states of two LFSRs. The complexity of this algorithm will serve as a bench mark for comparison with other attacks. The second one is the unconstraint embedding attack which is analogous to the attack that Golic proposed to the clock-control generator [3]. For the third one, we call it an *entropy attack*. We will derive a formula for computing the conditional entropy of the editing generator. Our analysis shows that the edited sequences resist to all of these three attacks.

This paper is organized as follows. In Section 2, we will introduce the construction of the editing generator and derive some basic properties. In Section 3, we will present the properties of randomness of the editing generator. In Section 4, we discuss security of the editing generator under a randomized model. We will consider three different attacks and compute their running times. Section 5 is the conclusion.

We conclude this section with some preliminaries on m-sequences over $\mathbb{Z}_3$ which will be used in this paper. For more results on m-sequence over non-binary fields, the reader is referred to [14].

A. Some Basic Concepts and Notations

Let $A = \{a_i\}$ be a sequence over $\mathbb{Z}_3$, i.e., $a_i \in \mathbb{Z}_3$. We say that $r > 0$ is a *period* of A if and only if $a_i = a_{i+r}, i = 0, 1, \cdots$. The smallest positive number satisfying $a_i = a_{i+r}, i = 0, 1, \cdots$, then r is called the *least period* of A. If A is a periodic sequence of period r, we also write $A = (a_0, a_1, \cdots, a_{r-1})$, a vector in the r dimensional linear space $\mathbb{Z}_3^{(r)} = \{(a_0, a_1, \cdots, a_{r-1}) | a_i \in \mathbb{Z}_3\}$.

Let $f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1 x - c_0, c_i \in \mathbb{Z}_3$. If the elements of A satisfies the following linear recursive relation:

$$a_{n+k} = c_{n-1}a_{n-1+k} + \cdots + c_1 a_{1+k} + c_0 a_k, k = 0, 1, \cdot \qquad (1)$$

then A is said to be a *LFSR sequence* generated by $f(x)$ and $f(x)$ is called a *characteristic polynomial* of A. The n-tuple $(a_0, a_1, \cdots, a_{n-1})$ is called an *initial state* of A. It is clear that A is completely determined by $f(x)$ and the initial state. We say that the first N elements of A are generated by LFSR $f(x)$ if (1) is true for $k = 0, 1, \cdots, N - n - 1$. The characteristic polynomial of $f(x)$ with the smallest degree is called the *minimal polynomial* of A. The degree of the minimal polynomial of A is called *linear span or linear complexity* of A. If $f(x)$ is a primitive polynomial over $\mathbb{Z}_3$ (i.e., $f(x)$ is irreducible and a root of $f(x)$ in the extension of $\mathbb{Z}_3$ has order $3^n - 1$), then A is called a m-sequence over $\mathbb{Z}_3$ of degree n.

B. The Properties of Randomness of Ternary m-Sequences

Property 1. Let A be a ternary m-sequence of degree n.

(1) The least period of A is $3^n - 1$.

(2) It satisfies the balance property, i.e., each non-zero element in $\mathbb{Z}_3$ occurs 3^{n-1} times in one period of A and zero element occurs $3^{n-1} - 1$ times in one period.

(3) It satisfies the following run property. In each period,

(a) for $1 \leq k \leq n-2$, the runs of each element in $\mathbb{Z}_3$ of length k occur $2^2 3^{n-k-2}$ times,

(b) the runs of each nonzero element of length $n-1$ occur once,

(c) the runs of the zero element occur twice,

(d) the run of each nonzero element of length n occurs once.

(4) 2-level autocorrelation.

(5) Each nonzero n-tuple occurs exactly once.

(6) Let $d = (3^n - 1)/2$. We write $A = (A_1, A_2)$ where $A_1 = (a_0, a_1, \cdots, a_{d-1})$ and $A_2 = (a_d, a_{d+1}, \cdots, a_{d+d-1})$. Then $A_2 = 2A_1$. I.e., $a_{i+d} = 2a_i, i = 0, 1, \cdots, d-1$. In particular, if $a_i = 0$, then $a_{i+d} = 0$ for all $\forall i \geq 0$.

The following is an example of a ternary m-sequence of degree 3 generated by Let $f(x) = x^3 - x^2 - 2$.

0 0 1 1 1 0 2 1 1 2 1 0 1
0 0 2 2 2 0 1 2 2 1 2 0 2

2 The Construction

In this section, we will introduce the construction of the editing generator and derive some basic properties on randomness. First, we take a look at the operations of the stop-and-go generator, a simple model of clock-control generators, and the shrinking generator. Both of these two generators can be represented in the following model in Figure 1. Let $A = \{a_i\}$ and $B = \{b_j\}$ be two binary sequences generated by LFSR 1 and LFSR 2 respectively.

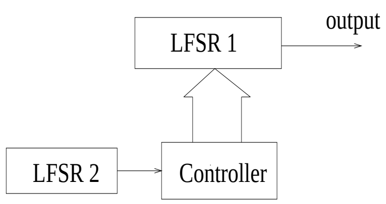

Fig. 1. Model of the Clock-control Generator and the Shrinking Generator

Stop-and-go generator:

Output sequence: $S = \{s_k\}$ whose elements are given by

$$s_{k_j} = a_{i_j}, j = 0, 1, \cdots, \text{ where } k_j = j, i_j = \sum_{t=0}^{j} b_t - 1 \text{ and set } a_{-1} = 0. \tag{2}$$

In other words, at time j, if $b_j = 1$, the generator outputs the current output bit of LFSR 1. Otherwise, the generator outputs the previous output bit (inserting). For example, let

$$\begin{aligned} A &= 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1 \cdots \\ B &= 1\ 1\ 1\ 1\ 1\ \underline{0}\ 1\ 1\ 1\ \underline{0\ 0\ 0}\ 1\ \underline{0}\ 1\ \underline{0}\ 1\ 1\ \underline{0\ 0} \cdots \end{aligned}$$

The inserting process is shown in the following Figure 2.

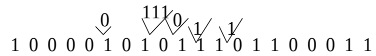

Fig. 2. Inserting Operation in the Clock-control Generator

The first 20 output bits are:

$$1\ 0\ 0\ 0\ 0\ \underline{0}\ 1\ 0\ 1\ \underline{1\ 1\ 1}\ 0\ \underline{0}\ 1\ \underline{1}\ 1\ 1\ \underline{1}\ 0$$

The underlined bits are the ones who have been inserted.

The Shrinking Generator:

Output sequence: $S = \{s_k\}$, at time j, if $b_j = 1$, then the generator outputs the current bit a_j of A. Otherwise this bit is discarded. I.e.,

$$s_{k_j} = a_{i_j}, j = 0, 1, \cdots, \text{ where } i_j = j, k_j = \sum_{t=0}^{j} b_t. \tag{3}$$

For example,

$$\begin{aligned} A &= 1\ 0\ \cancel{0}\ \cancel{0}\ \cancel{0}\ \cancel{1}\ 0\ 1\ 0\ \cancel{1}\ \cancel{1}\ \cancel{1}\ 0\ \cancel{1}\ 1\ \cancel{0}\ 0\ 0\ \cancel{1}\ 1 \cdots \\ B &= 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \cdots \end{aligned}$$

Fig. 3 Deleting Operation in the Shrinking Generator

The first 10 output bits from the shrinking generator are: 1 0 0 1 0 0 1 0 0 1.

The new generator, the editing generator, is to combine these two operations together, in order to resist to the known attacks to those two control models when they are used separately.

Definition 1. *Let $A = \{a_i\}$ and $B = \{b_j\}$ be two ternary sequences generated by LFSRs 1 and 2 respectively, as shown in Figure 1. The output sequence $S = \{s_k\}$ is determined by the following rules. At time j, if $b_j = 2$, the generator discards the current output element in LFSR 1; else if $b_j = 1$, the generator outputs the current output element in LFSR 1. Otherwise, the generator outputs the previous output element in LFSR 1. We will call it an* editing generator, *the resulting sequence an* edited sequence, *denoted as $Edit(A, B)$, the sequence A a* base sequence, *and the sequence B a* control sequence.

This definition can be written into the following pseudo-code:

Algorithm 1 (Edit(A, B)): Algorithm for Generating Edited Sequences

Input: A and B, two ternary LFSR sequences
Output: an edited sequence S of length m

1. **Procedure** $Edit(A, B, S, m)$
2. Set $a_{-1} = 0$
3. Set $k = j = i = 0$
4. **while** $(j < m)$ **do**
   (a) **if** $b_j = 2$, **then** set $i = i + 1$ // discarding the current output element in A
   // (deleting an element from A)
   (b) **else if** $b_j = 1$, **then** set
   $s_k = a_i$; $k = k + 1$; $i = i + 1$ //outputing the current element in A
   // (keeping an element of A)
   (c) **else** set $s_k = a_{i-1}$; $k = k + 1$ // outputting the previous output element in A
   //(inserting an element into A)
   (d) $j = j + 1$
5. **return** S.

Note. The initial value of a_{-1} can be taken as any value in $\mathbb{Z}_3$. In this paper, we will use this initial setting throughout of this paper.

In the following, we will use an example to explain this construction. Let both A and B be ternary sequences of period 27 which are obtained by adding one zero into one of two zero runs of length 2 in the m-sequence given in Section 1. I.e.,

$$\begin{aligned} A = &\ 2\ 0\ 1\ 2\ 2\ 1\ 2\ 0\ 2\ 0\ 001\ 1 \\ &\ 1\ 0\ 2\ 1\ 1\ 2\ 1\ 0\ 1\ 0\ 0\ 2\ 2 \end{aligned}$$

$$\begin{aligned} B = &\ 2\ 1\ 1\ 2\ 1\ 0\ 1\ 0\ 0\ 2\ 2\ 2\ 0 \\ &\ 1\ 2\ 2\ 1\ 2\ 0\ 2\ 0\ 001\ 1\ 1\ 0 \end{aligned}$$

The process of generating the first 12 elements of the edited sequence S is shown as follows:

$$\cancel{2}\ 0\ 1\ \cancel{2}\ 2^{\overset{2}{\vee}}\ 1^{\overset{11}{\vee}}\ \cancel{2}\ \cancel{0}\ \cancel{2}^{\overset{2}{\vee}}\ 0\ \cancel{0}\ \cancel{0}\ 1\ 1^{\overset{1}{\vee}} = 0\ 1\ 2\ 2\ 1\ 1\ 1\ 2\ 0\ 1\ 1\ 1$$

In the following, we will derive some basic properties of the edited sequences. From Algorithm $Edit(A, B)$, we know that the index i will have been incremented in cases (a) and (b) which correspond to $b_j \in \{1, 2\}$. The index k will have been incremented in cases (b) and (c), which correspond to $b_j \in \{0, 1\}$. Thus we have established the following assertion.

Property 2. With the notation in Definition 1, let $b'_j = 1$ if $b_j = 2$, otherwise $b'_j = b_j$. Then the elements of the edited sequence $S = \{s_k\}$ can be represented by

$$s_{k_j} = a_{i_j} \text{ where } i_j = \sum_{t=0}^{j} b'_t - 1 \text{ and } k_j = \sum_{b_t \neq 2, 0 \leq t \leq j} b_t \tag{4}$$

Definition 2. *With the notation in Property 2, we say that a_{i_j} is* controlled *by b_j. In this case, we also call the integer pair (i_j, k_j, j) a* match.

Remark 1. If the elements of A and B only take the values 1 and 2, from Property 2, we have $i_j = j$. According to (3), $Edit(A, B)$ is degenerated into a shrunk sequence. Similarly, if the elements of A and B only take the values 0 and 1, we have $k_j = j$. According to (2), $Edit(A, B)$ is degenerated into a clock-controlled sequence.

Remark 2. The elements in an edited sequence are determined by the domain from which the base sequence A takes its elements. The domain from which the control sequence B takes its elements represents a set of strategies used for editing. Thus one may consider a general model for $Edit(A, B)$ where A is a sequence whose elements taken from $\mathbb{Z}_t = \{0, 1, \cdots, t-1\}$ and B is a sequence whose elements, taken from $\mathbb{Z}_r$. We will continue to investigate the editing generator in this general setting in the future.

The properties of randomness of the edited sequence $Edit(A, B)$ depend on the base sequence A and the control sequence B. In the rest of this paper, we will restrict ourselves into the case that both these two sequences are chosen as modified ternary m-sequences of degree n. The definition of a modified ternary m-sequence of degree n is as follows.

Definition 3. *If a sequence is constructed by adding one zero into one of two zero runs of length $n-1$ in a ternary m-sequence of degree n, then we call it a* modified ternary m-sequence of degree n, mm-sequence *for a short.*

From Property 1, we know that a modified ternary m-sequence of degree n has period 3^n, each element in $\mathbb{Z}_3$ occurs exactly 3^{n-1} times and each n-tuple in $\mathbb{Z}_3^{(n)}$ occurs exactly once in one period of 3^n.

Property 3. Let $S = Edit(A, B)$ where A and B are two mm-sequence of degree n. Then we have the following matches.

$$i_{3^n} = 2 \cdot 3^{n-1} \quad \text{and } k_{3^n} = 2 \cdot 3^{n-1} \tag{5}$$

$$i_{2 \cdot 3^n} = 4 \cdot 3^{n-1} \quad \text{and } k_{2 \cdot 3^n} = 4 \cdot 3^{n-1} \tag{6}$$

$$i_{3 \cdot 3^n} = 2 \cdot 3^n \quad \text{and } k_{3 \cdot 3^n} = 2 \cdot 3^n \tag{7}$$

Moreover, $2 \cdot 3^n$ is a period of S.

Proof. Since both of A and B are mm-sequences of degree n, then each element in $\mathbb{Z}_3$ occurs exactly 3^{n-1} times in one period of A and B, respectively. Notice that in Algorithm 1, i will not be incremented when $b_j = 0$ and neither k will when $b_j = 2$. Therefore we have that $i = 2 \cdot 3^{n-1}$ and $k = 2 \cdot 3^{n-1}$ when $j = 3^n$, which gives the two formulae in (5). The rest of the formulae follows by applying a similar argument. Thus the first assertion is true. From (7), we have $i = k = 2 \cdot 3^n$ when $j = 3^{n+1}$. At this time, A and B go back to the same initial states as $k = j = i = 0$. Thus we have $s_{k+2\cdot 3^n} = s_k$ for all $k \geq 0$, which establishes the assertion 2.

3 The Properties of Randomness

In this section, we will derive the least period of the edited sequence $Edit(A, B)$, a large lower bound for the linear span, ratio of linear span to period, and occurrences of symbols. We will keep the notation in Section 2.

3.1 Period and Linear Span

Theorem 1. *Let $S = \{s_k\}$ be the edited sequence $Edit(A, B)$ where both $A = \{a_i\}$ and $B = \{b_j\}$ are mm-sequence of degree n. Then $Per(S)$, the least period of S, is either $2 \cdot 3^n$ or 3^n.*

Proof. From Property 3, we know that $Per(S)|2 \cdot 3^n$. If $Per(S)$ is a factor of $2 \cdot 3^n$ or 3^n, then $Per(S)|2 \cdot 3^{n-1}$. Since B is a mm-sequence of degree n, then each n-tuple occurs exactly once in one period of B. Thus we can suppose that $b_j = b_{j+1} = \cdots = b_{j+n-1} = 1$ for some $j \geq 0$. Since a_{i_j} is controlled by b_j, then a_{i_j+r} is controlled by b_{j+r} for $r = 0, 1, \cdots, n-1$. Notice that from Property 3, we know that $(2 \cdot 3^{n-1}, 3^n)$ is a match. Thus b_{j+3^n+r} controls $a_{i_j+2\cdot 3^{n-1}+r}$ for $r = 0, 1, \cdots, n-1$. Notice b_{j+3^n} is the element after one period of length 3^n of B, which is just b_j. Therefore, the elements in a period of $2 \cdot 3^{n-1}$ of S are generated under the control of a shift of B: $(b_j, b_{j+1}, \cdots, b_{j+3^n-1})$. Thus we have $a_{i_j+r} = a_{i_j+2\cdot 3^{n-1}+r}$, $r = 0, \cdots, n-1$. Note that $3^{n-1} > n$ when $n > 1$. Hence $2 \cdot 3^{n-1} + n - 1 + 1 < 3^n$. Therefore, all elements in the set of $\{a_{i+r}, a_{i+2\cdot 3^{n-1}+r} | r = 0, \cdots, n-1\}$ are within one period of A. Consequently, the two equal n-tuples $(a_i, \cdots, a_{i+n-1})$ and $(a_{i+2\cdot 3^{n-1}}, \cdots, a_{i+2\cdot 3^{n-1}+n-1})$ appear in one period of A which is a contradiction. Thus we have that $Per(S) = 2 \cdot 3^n$ or 3^n.

In the next theorem, we will establish a lower bound for the linear span of the edited sequences.

Theorem 2. *Let $S = Edit(A, B)$, where A and B are mm-sequences of degree n. Then $LS(S)$, the linear span of S, is lower bounded by*

$$LS(S) > 3^{n-1}.$$

Proof. From Theorem 1, $f(x) = x^{2\cdot 3^n} - 1$ is a characteristic polynomial of S over $\mathbb{Z}_3$. We can write that $f(x) = (x^2 - 1)^{3^n}$. Let $m(x)$ be the minimal polynomial of S. Then $m(x) = (x-1)^a (x+1)^b$ for some integer $a \leq 3^n$ and $b \leq 3^n$. If $deg(m(x)) \leq 3^{n-1}$, then $a \leq 3^{n-1}$ and $b \leq 3^{n-1}$. Consequently, $m(x)|(x^2 - 1)^{3^{n-1}} = x^{2\cdot 3^{n-1}} - 1$. Thus the least period of S is less than or equal to $2 \cdot 3^{n-1}$, which is a contradiction with Theorem 1. So $LS(S) > 3^{n-1}$.

Corollary 1. *With the notation in Theorem 2, let η be the ratio of the linear span to the least period of the edited sequence (represents* linear span per symbol*). Then η satisfies*

$$\eta > \begin{cases} 1/6 & \text{if } Per(S) = 2 \cdot 3^n \\ 1/3 & \text{if } Per(S) = 3^n. \end{cases}$$

3.2 Occurrences of Symbols

In the following theorem, we will establish a bound for frequency of each element in $\mathbb{Z}_3$ occurs in an edited sequence.

Theorem 3. *Let $S = Edit(A, B)$ where both $A = (a_0, a_1, \cdots, a_{3^n-1})$ and $B = (b_0, b_1, \cdots, b_{3^n-1})$ are mm-sequences of degree n. Then each element in $\mathbb{Z}_3$ occurs at least 3^{n-1} times in one period of length $2 \cdot 3^n$ of S.*

Proof. Let $d = (3^n - 1)/2$. We consider that $S = (s_0, s_1, \cdots, s_{2 \cdot 3^n-1})$. Without loss of generality, we may suppose that the zero n-tuple appears in the first $d + 1$ elements of B. Let $\tilde{D} = (\tilde{b}_0, \tilde{b}_1, \cdots, \tilde{b}_{3^{n-1}-1})$ be the resulting sequences from the first $d + 1$ elements of B by deleting all 0's, and let $\tilde{B} = (\tilde{b}_0, \tilde{b}_1, \cdots, \tilde{b}_{2 \cdot 3^{n-1}})$ be the resulting sequence of B by deleting all 0's. Then $\tilde{B}$ has a period of $2 \cdot 3^{n-1}$. Applying Property 1-(6), we have $\tilde{B} = (\tilde{D}, 2\tilde{D})$. We write $A = (A_0, A_1, A_2)$ where $A_i = (a_{i3^{n-1}}, a_{i3^{n-1}+1}, \cdots, a_{i3^{n-1}+3^{n-1}-1}), i = 0, 1, 2$. Let $\tilde{S} = Edit(A, \tilde{B})$, an edited sequence from the base sequence A and the control sequence $\tilde{B}$. Then $\tilde{S}$ is the resulting sequence from S by deleting all elements of S which are contributed from the inserting operation. Thus all elements that occur in $\tilde{S}$ will occur in S (in this case, we say that $\tilde{S}$ is a *subsequence* of S. In the following, we will show that each elements of $\mathbb{Z}_3$ will occur 3^{n-1} times in $\tilde{S}$. Note that there is no zeros in $\tilde{B}$, then $i_j = j, j = 0, 1, \cdots$. In other words, $\tilde{b}_j$ controls a_j for all $j \geq 0$. Therefore, $(\tilde{D}, 2\tilde{D}, \tilde{D}, 2\tilde{D}, \tilde{D}, 2\tilde{D})$ controls $(A_0, A_1, A_2, A_0, A_1, A_2)$ term by term. Consequently, each of $\tilde{D}$ and $2\tilde{D}$ controls $A_i, i = 0, 1, 2$ once. From $Edit(A_0, \tilde{D})$, we get the elements a_j for those j such that $\tilde{b}_j = 1, 0 \leq j < 3^{n-1}$. From $Edit(A_0, 2\tilde{D})$, we get the elements a_j for those j such that $\tilde{b}_j = 2, 0 \leq j < 3^{n-1}$. Thus $\tilde{S}$ contains the entire set of the elements of A_0. Similarly, from $Edit(A_i, \tilde{D})$ and $Edit(A_i, 2\tilde{D})$, we get all the elements of A_i, $i = 1, 2$. Thus the set of the elements of $\tilde{S}$ is equal to the set of the elements of A. Since each element in $\mathbb{Z}_3$ occurs in A exactly 3^{n-1} times and $\tilde{S}$ is a subsequence of S, then each element in $\mathbb{Z}_3$ occurs in S at least 3^{n-1}.

4 Security Analysis

In this section, we will consider security of the edited sequence $S = Edit(A, B)$ where $A = \{a_i\}$ and $B\{b_j\}$ are mm-sequences of degree n. We model both A and B as independent random sequences, which is the same setting as in [5] [3]. For any sequence $U = \{u_i\}$, We denote $U_k^{m-1+k} = (u_k, u_{k+1}, \cdots, u_{k+m-1})$, m consecutive elements of U starting with index $k \geq 0$. We will frequently use this notation in this section. We assume that a portion of the key stream sequence S, say $S_k^{m-1+k} = (s_k, s_{k+1}, \cdots, s_{k+m-1})$, is known (it can be obtained from the known plain-text attack). The primitive polynomials that generate $\{a_i\}$ and $\{b_j\}$ are also assumed to be known. The objective of these attacks is to find out a matched state of $\{a_i\}$ and $\{b_j\}$. From these states, we can generate the entire key stream S by performing the forward (backward) parity check (we will define it below). Note that here we do not request to find the initial states of LFSRs 1 and 2 (see Fig. 1), since S is determined uniquely by any pair of matched states of A and B. Without loss of generality, we suppose that S_0^{m-1} is known. Thus the objective becomes to find the initial state of A and B. In this section, we will propose three different attacks to the editing generator. The first one is a brute-force like attack in search of an initial state of B. The complexity of this algorithm will serve as a bench mark for comparison with other attacks. The second one is the unconstraint embedding attack which is analogous to the attack that Golic proposed to the clock-control generator [3]. We call the third attack an *entropy attack*. We will derive a formula for computing the conditional entropy of the editing generator. Our analysis shows that the edited sequences resist to all of these three attacks.

Next we define the forward (backward) parity check for the mm sequences. Let $f(x) = \sum_{i=0}^{n} t_i x^i$ and $g(x) = \sum_{i=0}^{n} r_i x^i$ with $t_n = r_n = 1$ be the the primitive polynomials associated with A and B, respectively.

The (forward) parity check for A_0^{N-1}:

$$\sum_{j=0}^{n} t_j a_{k+j} = 0, k = 0, 1, \cdots, N-n-1. \tag{8}$$

The backward parity check for A_0^{N-1}:

$$\sum_{j=0}^{n} t_j a_{N-(k+j)-1} = 0, k = 0, 1, \cdots, N-n-1. \tag{9}$$

Note that since A is a mm-sequence, then $f(x)$ generates all A_0^{N-1}, $N \geq n$ except for that A_0^{N-1} contains one of the $(n+1)$-tuples which contain the zero run of length n: $(0, \cdots, 0, u)$ and $(u, 0, \cdots, 0)$, $u \in \{1, 2\}$. Since those $(n+1)$-tuples are easy to detect, so, when we say that an arbitrary sequence $\{a_i\}_0^{N-1}$ passes the above forward (backward) parity check, we include the computation for detecting those $(n+1)$-tuples from the stream $\{a_i\}_0^{N-1}$ if there are any.

4.1 A Brute-Force Like Attack

In the following, we present an algorithm for search of the right control sequence in order to reconstruct the base sequence A given that S_0^{m-1} is known.

Algorithm 2: Algorithm for search of the right B

Input: $S_0^{m-1}, \{b_j\}$ for all $B_0^{n-1} \in \mathbb{Z}_3^{(n)}$
Output: $\{a_i\}$

1. $i = j = k = 0$
2. **while** $(k < m)$
   (a) **if** $(b_j = 0)$
   if (a_{i-1} is not defined) $a_{i-1} = s_k$
   else if $(a_{i-1} \neq s_k)$ exit("failure")
   (b) **else if** $(b_j = 1)$ $\{a_i = s_k; i++; k++; \}$
   (c) **else** $\{a_i = *; i++; \}$
   (d) $j++;$
3. Set $I = i - 1$ and apply the forward and backward parity check to $\{a_i\}_0^I$ for determining unknown * in $\{a_i\}_0^I$;
4. **if** (the check fails) **exit**("failure");
5. **return** $\{a_i\}$

Note. In Step 3 in Algorithm 2, for $0 \leq i \leq I - n$, if $a_{i+n} =$ '$*$', we define $a_{i+n} = -\sum_{j=0}^{n-1} t_j a_{i+j}$.

The running time of Algorithm 2 depends on Step 3. In the following lemma, we will give an average value of the number of $*$ in $\{a_i\}_0^I$ after the while loop is done.

Lemma 1. *Let Z be the number of *'s in $\{a_i\}_0^I$ after the while loop in Algorithm 2 is done. If the while loop does not exit ("failure"), then $Z = \frac{1}{3}m$ in average.*

Proof. Note that the length of $\{a_i\}$ will not be incremented when $b_j = 0$ in the while loop. If the case $b_j = 0$ is not executed, then the ratio of the number of elements which are assigned values (called *digits*) to the number of unknown *'s in $\{a_i\}$ should be equal to 1. The event that * is reassigned to a digit at index $j = j_0$ corresponds to the pair $(b_{j_0}, b_{j_0+1}) = (2, 0)$. Thus at the end of the while loop, one third of *'s in $\{a_i\}_0^I$ will have been reassigned to digits. Therefore, the ratio of the number of the digits to the number of *'s in $\{a\}_0^I$ is equal to $(1 + 1/3) : (1 - 1/3) = 2 : 1$. Recall that both A and B are considered as truly random variables. Thus, in average, the length of S and the length of the corresponding $\{a_i\}$ are equal. Therefore, $Z = \frac{1}{3}m$. □

In the following, we will consider the running time of Algorithm 2 when it successfully finds the right $\{b_j\}$.

Case 1. The while loop does not exit ("failure")

When $(a_0, \cdots, a_{n-1})$ contains symbol *, we randomly assign values in $\mathbb{Z}_3$ for them, and then perform the forward parity check in Step 3. When the check fails, if $a_{i+n} \neq *$, we consider that $\sum_{j=0}^{n-1} t_j a_{i+j}$ as a random variable which takes values in $\mathbb{Z}_3$ equally likely. Thus, in average, this check will stop at $\sum_{i=1}^{\infty} i(\frac{1}{3} + \frac{2}{3} \times \frac{1}{3})^{i-1} \times \frac{2}{3} \times \frac{2}{3} = \frac{9}{4}$ steps. If there exist n consecutive digits in $\{a_i\}$, then we perform the forward and backward parity check for the rest of a_i's. If the check is not successful, it only needs $\frac{9}{4}$ steps in average. But if there do not exist n consecutive digits in $\{a_i\}$, then, in average, there are $\frac{n}{3}$ *'s in $\{a_i\}_k^{k+n-1}$ for any $k \geq 0$. In this case, Step 3 needs to perform a brute force search for $(a_0, \cdots, a_{n-1})$. The running time for this search is given by $\Theta(n3^{\frac{n}{3}})$ (Here $y = \Theta(x)$ means there exists positive constant c_1 and c_2 such that $c_1 x \leq y \leq c_2 x$.) Note that, in general, we have $m = \Theta(n)$. Since the number that the while loop will be executed is $\Theta(m)$, it follows that for a given $\{b_j\}$ if the while loop does not exit("failure"), then the running time of Algorithm 2 is $\Theta(m + n3^{\frac{n}{3}}) = \Theta(n3^{\frac{n}{3}})$.

Case 2. The while loop does exit("failure")

If $\{b_j\}$ is not the right control sequence, then the failure probability at $b_j = 0$ in the while loop is 2/3 when the equality check is performed. We need not to check whether a_{i-1} equals s_k when $b_j = 0$. This will happen with possibility 1/3. Notice that the number that we need to perform this check is $\frac{m}{3}$. This can be seen from the proof of Lemma 1 by noticing that the ratio of the number of 0's in $\{b_j\}$ which control to reassign those * to digits in the while loop is one third of the total of 0's in $\{b_j\}$, in average. Therefore, the probability without executing exit("failure") is $(\frac{1}{3})^{\frac{m}{3}}$. Furthermore, since $\{b_j\}$ is modeled as purely random, the probability that the while loop does exit("failure") at $j = J$ is given by

$$(2/3 + 1/3 \times 1/3 + 1/3 \times 2/3 \times 1/3)^{J-1} 1/3 \times 2/3 \times 2/3 = \left(\frac{23}{27}\right)^{J-1} \frac{4}{27}.$$

So the average of steps that the while loop will execute before exit("failure") is given by $E(j) = \frac{27}{4}$. Thus with probability $1 - (\frac{1}{3})^{\frac{m}{3}}$, the algorithm has running time $\Theta(3^n)$.

Combining these two cases discussed above, we have the average running time is determined by $(1 - (\frac{1}{3})^m)3^n + (\frac{1}{3})^m \cdot 3^{\frac{4n}{3}}$. Since $m \geq n$, we have established the following result.

Theorem 4. *With the above notation, the running time of Algorithm 2 for finding the right initial states of* $\{a_i\}$ *and* $\{b_j\}$ *is given by* $\Theta(3^n)$ *in average.*

4.2 Unconstraint Embedding Attack

In [3], Golic discussed the unconstraint embedding attack to the binary clock control generator. In this subsection, we consider a similar attack to the editing generator. According to the definition of the edited sequences, we can write $S = \{a_{i_k}\}$. We define $D = \{d_k\}$ whose elements are given by $d_k = i_k - i_{k-1}, k \geq 0$. D is called a *jump sequence* of S. Thus we have $i_k = \sum_{i=0}^{k} d_i + i_{-1}$ where $i_{-1} = -1$. The unconstraint embedding attack is to consider the probability that a portion of the key stream S_0^{N-1} can be embedded into a random sequence $\{a_i\}_0^{m-1}$, where $m \geq N$. We denote this probability as $P_S(N, m)$. If $P_S(N, m)$ decreases to zero exponentially, then it is possible to reconstruct the initial state of $\{a_i\}$ efficiently. For these discussions on different alphabetic sets for d_k, please see [15] for $d_k \in \{1, 2\}$, [4] for $d_k \in [1, k]$ and [11] for $d_k \in \mathbb{Z}^+$. The case considered in [3] is $P_S(N,m)$ for $d_k > 0$. Here we will apply a similar mechanism to compute $P_S(N,m)$ in the edited sequences. Here we will consider the case including $d_k = 0$. I.e., the range of d_k: $d_k \geq 0$. In the following, we introduce a definition for a repeating operation.

Definition 4. *Let (i_k, k, j) be a match (i.e., b_j controls a_{i_k} to generate $s_k = a_{i_k}$). If b_{j+1} controls $a_{i_{k+1}}$ to generate $s_{k+1} = a_{i_{k+1}}$ and $i_{k+1} = i_k$, then we say that element s_{k+1} is generated by a* repeating operation*, or s_{k+1} is a* repeated element.

Lemma 2. *In one period of $2 \cdot 3^n$ of the edited sequence S, there are $2 \cdot 3^{n-1}$ that are generated by the repeating operation. In other words, one third of elements of S are generated by the repeating operation.*

Proof. From the definition, we see that each of b_j and b_{j+1} produces one output element. So they are not equal to 2. Since $i_{k+1} = i_k$, we have $b_{j+1} = 0$. Hence (b_j, b_{j+1}) can only be (1, 0) or (0, 0). It is easy to check both of these pairs will produce repeated elements. Furthermore, if s_{k+1} is generated by the repeating operation, then the matched control pair (b_j, b_{j+1}) for (s_k, s_{k+1}) are (b_j, b_{j+1})=(0, 0) or (1, 0). Thus in one period of length 2×3^n of S , there are $\frac{2}{9} \times 3^{n+1} = 2 \times 3^{n-1}$ elements are generated by the repeating operation.

With the notation above, we will construct two new sequences from S. Let $\overline{S} = \{\overline{s}_k\}$ be the resulting sequence obtained from $S = \{s_k\}$ by deleting the elements in $\{s_k\}$ which are generated by the repeating operation. Let $\overline{D} = \{\overline{d}_k\}$ be the jump sequence of $\overline{S}$. Then $\overline{d}_k > 0$ for $k \geq 0$. Let $S^* = \{s_k^*\}$ be the resulting sequence obtained from $\{s_k\}$ by deleting s_k when $s_k = s_{k-1}$. From the definitions, it is immediate that S^* is a subsequence of $\overline{S}$. Note that, unfortunately, we also delete some s_k which does not come from the inserting operation although it satisfies that $s_k = s_{k-1}$.

The following proposition shows the embedding probability of ${S_0^*}^{N-1}$.

Proposition 1. *With the notation above, the the unconstraint embedding probability of ${S_0^*}^{N-1}$ is given by*

$$\begin{aligned} P_{S^*}(N,m) &= \sum_{k=0}^{m-N} \binom{N-1+k}{k} 3^{-N-k} \\ &= 1 - 3^{-m} \sum_{k=0}^{N-1} \binom{m}{k} 2^{m-k} \end{aligned}$$

Proof. Let $X = \{x_i\}_1^m$ be a random sequence into which ${S_0^*}^{N-1}$ can be embedded (note that the starting label does not affect the embedding property, W.O.L.G, we assume that the label of X starts from 1). We define the jump sequence $D^* = \{d_k^*\}$ as follows. We define $d_0^* = j$ where j is the smallest integer such that $s_0^* = x_j$. For $k > 0$, we recursively define that $d_k^* = j$ where j is the smallest integer such that $s_k = x_{i_k}$ where $i_k = \sum_{r=0}^{k-1} d_r^* + j$. From this definition, we can see that D^* is well defined when ${S_0^*}^{N-1}$ can be embedded into $\{x_i\}_1^m$. Notice that $P_{S^*}(N,m) = \frac{A(N,m)}{3^m}$ where $A(N,m)$ represents the number of $\{x_i\}_1^m$ in which ${S_0^*}^{N-1}$ can be embedded. It is clear that one D^* with $\sum_{i=0}^{N-1} d_i^* = m - J$ where $0 \leq J \leq m - N$ corresponds to a set of

$2^{m-N-J}3^J$ different X's. Moreover, different D^* corresponds to disjoint sets. Since there are $\binom{m-J-1}{N-1}$ such D^*, it follows that

$$P_{S^*} = \sum_{J=0}^{m-N} \binom{m-J-1}{N-1} 2^{m-N-J} 3^{J-m} \tag{10}$$

$$= 3^{-N} \sum_{J=0}^{m-N} \binom{N+J-1}{J} (\frac{2}{3})^J \tag{11}$$

$$= 1 - 3^{-m} \sum_{k=0}^{N-1} \binom{m}{k} 2^{m-k} \tag{12}$$

$$= 3^{-m} \sum_{k=0}^{m-N} \binom{m}{k} 2^k \tag{13}$$

Notice that (12) is derived from considering constant string embedding. Thus the equality from (11) to (12) holds.

□

If we expect this attack to be successful, then m must be large enough to guarantee that S_0^{*N-1} can be embedded into the original $(a_0, \cdots, a_{m-1})$. This requests that $m \approx N\overline{d}$, where $\overline{d}$ is the average of the jump steps of S_0^{*N-1} when it is considered as a subsequence of $\{a_i\}$. Notice that if S_0^{*N-1} is derived from two periods of $\{a_i\}$ and its length is l, then $\overline{d} = \frac{2\times 3^N}{l}$. ($l$ is also a period of S_0^{*N-1}). Let $\overline{S}_0^{T-1}$ be constructed from $S_0^{2\cdot 3^n}$. Then $\overline{S}_0^{T-1}$ is a subsequence of $\{a_i\}$. Since $\{a_i\}$ is purely random, then $\overline{S}_0^{T-1}$ is also a purely random sequence. By Lemma 2, $T = (1-\frac{1}{3}) \times 2 \times 3^n = 4 \times 3^{n-1}$. From the construction of S_0^{*l-1}, each element, say u, in S_0^{*l-1}, starts a run of u's in $\overline{S}_0^{T-1}$. Since $\overline{S}_0^{T-1}$ is purely random, from Property 1-(3) in Section 1, it follows that $l \approx 3 \times \sum_{k=3}^{[log_3 T]} \frac{2^2 \times T}{3^k} \approx 8 \times 3^{n-2}$. Therefore, $\overline{d} = \frac{9}{4}$. Thus $P_{S^*} = 3^{-m} \sum_{k=0}^{\frac{5m}{9}} \binom{m}{k} 2^k$. In other words, there are $\sum_{k=0}^{\frac{5m}{9}} \binom{m}{k} 2^k$ sequences $\{a_i\}$ in which S_0^{*N-1} can be embedded. We, then, apply the forward (backward) parity-check to those candidates for finding the right $\{a_i\}$. Consequently, if the attack is successful attack, then the running time is given by

$$\Omega\left(\sum_{k=0}^{\frac{5m}{9}} \binom{m}{k} 2^k\right) \tag{14}$$

where $y = \Omega(x)$ means that there exists a positive constant c such that $y \geq cx$. Note that if the edited sequence S_0^{L-1} is generated from the base sequence $\{a_i\}_0^{m-1}$, then $L = m$ in average.

Theorem 5. *With the notation above, the running time for the successful unconstraint embedding attack is given by* $\Omega(\frac{3^{0.9298m}}{m})$ *where m is the length of the known key stream.*

Proof. From the above analysis, we only need to derive a bound for the expression in (14). Notice $\binom{m}{k} 2^k < \binom{m}{k+1} 2^{k+1} \iff k < \frac{2m-1}{3}$. Let $k_0 = \left[\frac{2m-1}{3}\right]$, then $\binom{m}{k_0} 2^{k_0} > \binom{m}{i} 2^i$ for any $0 \leq i \leq m$. Since

$\binom{m}{5m/9} \geq \binom{m}{k_0}$, it follows that $\binom{m}{5m/9} 2^{k_0} > \frac{3^m}{m+1}$. Therefore, $\sum_{k=0}^{\frac{5m}{9}} \binom{m}{k} \geq \frac{1}{m+1} \cdot 3^m 2^{-m/9} = \frac{1}{m+1} 3^{0.9298m}$. Thus the running is $\Omega(\frac{3^{0.9298m}}{m})$.

Note. m is greater than n. Thus, in general, this attack is not as efficient as the brute-force like attack. The reason is that the unconstaint attack is more general than the brute-force like attack since the brute-force like attack heavily utilizes the structure of the edited sequences.

4.3 Entropy Attack

In this subsection, we will consider the distribution of the jump sequence $D = \{d_k\}$ which leads to a new attack, called *entropy attack*. Since $\{b_j\}$ is modeled as a purely random sequence, then d_k is an i.i.d random variable over a non-negative integer domain. Thus we have $P(d_k = 0) = \frac{1}{3}$, $P(d_k = u) = (\frac{1}{3})^{u-1} \times \frac{4}{9}$ for all $u \geq 1$.

Theorem 6. *Given a key stream S_0^{N-1}, if $s_k = s_{k-1}$, then*

$$P(d_k|S_0^{N-1}) = \begin{cases} \frac{1}{2P(s_k)+1} & \text{if } d_k = 0 \\ \\ \frac{3P(d_k)P(s_k)}{2P(s_k)+1} & \text{if } d_k \neq 0 \end{cases} \tag{15}$$

If $s_k \neq s_{k-1}$, then

$$P(d_k|S_0^{N-1}) = \begin{cases} 0, & \text{if } d_k = 0 \\ \frac{2}{3} \times (\frac{1}{3})^{d_k - 1}, & \text{if } d_k \geq 1. \end{cases} \tag{16}$$

A proof will be attached as an appendix due to short of space.

In the following, we will present an attack based on the result of Theorem 6.

Algorithm 3. Entropy Attack ($\{s_i\}_0^{N-1}$)
Step 1 Determine $P(d_k|\{s_i\}_0^{N-1}, k = 0, \cdots, N-1)$;
Step 2 Randomly choose d_k according to distribution $P(d_k|\{s_i\}_0^{N-1}), k = 0, \cdots, N-1$;
Step 3 Recover $a_{i_k} = s_k, k = 0, \cdots, N-1$, where $i_k = \sum_{j=0}^{k} d_j - 1$;
Step 4 Carry out Parity-Check to recover $\{a_i\}$;
Step 5 If Step 4 is not successful, go to Step 2. Otherwise, claim successful and output $\{a_i\}$.

Analysis. Let $H(k) = H(d_k|$ given specific $\{s_i\}_0^{N-1})$. To get the correct d_k in Step 2 above , we need to try $2^{H(k)}$ times. So to get all the $\{d_i\}_0^{N-1}$ correctly, we need to try $2^{\sum_{k=0}^{N-1} H(k)}$ times. If $s_k = s_{k-1} = u$, then

$$H(k) = log_2(2P(u)+1) - \frac{2P(u)}{2P(u)+1} log_2 P(u) - \frac{0.7548P(u)}{2P(u)+1}.$$

In this case, we denote $H(k)$ as H_u where $u \in \mathbb{Z}_3$, since it does not depend on k. Note that H_u is an increasing function in $P(u)$. From Theorem 3, we can derive that $1/6 < P(u) < 2/3$. It follows $0.96695 \leq H_u \leq 1.3410$. If $s_k \neq s_{k-1}$, then

$$H(k) = \frac{3}{2}log3 - 1 = 1.377.$$

Let $s_0 = \cdots = s_{k(0)} < s_{k(0)+1} = \cdots = s_{k(1)} < \cdots = s_{k(t-1)}$. From the discussion in Section 4.2, we know that $E(t) = \frac{N}{d}$, So, in average, $t = \frac{N}{d} = \frac{4N}{9}$. Thus the conditional entropy of the jump sequence given that S_0^{N-1} is bounded by

$$0.96695 \times 5N/9 + 0.612N = 1.1491N \leq \sum_{k=0}^{N-1} H(k) \leq 1.3410 \times 5N/9 + 0.612N = 1.3570N$$

Therefore, we have established the following result.

Theorem 7. *The average running time of Algorithm 3 is lower bounded by*

$$\Omega(2^{1.1491N}) = \Omega(3^{0.7250N})$$

and upper bounded by

$$O(2^{1.3570N}) = O(3^{0.85619N}).$$

Note. In general, $N > n$, so the entropy attack may not be as efficient as the brute force like attack. However, it is much more efficient than the unco nstaint embedding attack.

We have derived the computational complexities of these three attacks, respectively. These complexities show that the editing generator resists to all of these three attacks.

5 The Conclusion

In this paper, we presented a new pseudo-random sequence generator using two ternary LFSRs where the control part is a combined model of the clock-control generator (insertion) and the shrinking generator (deletion). The new generator is called an editing generator. We have shown that the edited sequences have good properties of randomness, such as large period, high linear span, large rate of linear span per symbol, and low un-bias of occurrences of symbols. We have analyzed three different attacks for finding initials of two ternary LFSRs used in the editing generator, given that a portion of a key stream is known. They are the brute-force like attack, the unconstraint embedding attack and the entropy attack. We have shown the editing generator resists to all of these three attacks. We will continue to investigate security of this generator. We will also consider to apply the brute-force attack and the entropy attack developed in this paper to the binary clock-control generator and the shrinking generator in the future.

Appendix

Proof of Theorem 6 in Section 4. From the conditional probability, we have

$$P(d_k|S_0^{N-1}) = \frac{P(d_k, S_0^{N-1})}{P(S_0^{N-1}} = \frac{P(S_0^{N-1}|d_k)P(d_k)}{P(S_0^{N-1})}. \tag{17}$$

Notice that $P(S_0^{N-1}) = \sum_{d_k=0}^{\infty} P(d_k)P(S_0^{N-1}|d_k)$. Now we show

$$P(S_0^{N-1}|d_k = d_k') = P(S_0^{N-1}|d_k = d_k'') \text{ for } d_k', d_k'' > 0, \tag{18}$$

Using the Bayes formula,

$$\begin{aligned} P(S_0^{N-1}|d_k) &= \sum_{d_j \in \mathbb{Z}^+ \cup \{0\}, j \neq k} P(S_0^{N-1}|d_0^{N-1})P(d_0^{N-1}|d_k) \\ &= \sum_{d_j \in \mathbb{Z}^+ \cup \{0\}, j \neq k} P(S_0^{N-1}|d_0^{N-1})P(d_0, \cdots, d_{k-1}, d_{k+1}, \cdots, d_{N-1}). \end{aligned}$$

Thus we only need to show that

$$P(S_0^{N-1}|d_0^{N-1}, d_k = d_k') = P(S_0^{N-1}|d_0^{N-1}, d_k = d_k''). \tag{19}$$

Let $I_i' = \sum_{j=0}^{i} d_j - 1$ where $d_k = d_k', i = 0, \cdots, N-1$, and $I_i'' = \sum_{j=0}^{i} d_j - 1$ where $d_k = d_k'', i = 0, \cdots, N-1$. Notice that $I_i' = I_{i+1}' \Leftrightarrow I_i'' = I_{i+1}''$ for $i = 0, \cdots, N-2$. Let

$$I_0' = \cdots = I_{i(0)}' < I_{i(0)+1}' = \cdots = I_{i(1)}' < \cdots = I_{i(l)}'$$

for some integer l for which $i(l) = N-1$. Since $\{a_i\}$ is purely random, if $P(S_0^{N-1}|d_0^{N-1}, d_k = d_k') \neq 0$, then

$$P(S_0^{N-1}|d_0^{N-1}, d_k = d_k') = \prod_{r=0}^{l} P(a_{I_{i(r)}} = s_{i(r)}) = \prod_{r=0}^{l} P(a = s_{i(r)}).$$

The right hand side of the above identity does not depend on d_k'. Together with

$$P(S_0^{N-1}|d_0^{N-1}, d_k = d_k') = 0 \Longleftrightarrow P(S_0^{N-1}|d_0^{N-1}, d_k = d_k'') = 0,$$

it follows that (19) is true. Thus **??** is true. Since if $s_k \neq s_{k-1}$, then $P(S_0^{N-1}|d_k = 0) = 0$. Therefore, the assertion on the case $s_k \neq s_{k-1}$ is established. For the case $s_k = s_{k-1}$, applying a similar method as above, we have

$$P(S_0^{N-1}|d_0^{N-1}, d_k = d_k'') = P(s_k)P(S_0^{N-1}|d_0^{N-1}, d_k = 0) \tag{20}$$

where $d_k'' > 0$. From (17), we have

$$P(d_k = 0|S_0^{N-1}) = \frac{P(S_0^{N-1}|d_k = 0)P(d_k = 0)}{(1 - P(d_k = 0))P(S_0^{N-1}|d_k = 0)P(s_k) + P(d_k = 0)P(S_0^{N-1}|d_k = 0)} \tag{21}$$

$$= \frac{1}{2P(s_k) + 1}. \tag{22}$$

Similarly,

$$P(d_k = u|S_0^{N-1}) = \frac{P(S_0^{N-1}|d_k = u)P(d_k = u)}{(1 - P(d_k = 0))P(S_0^{N-1}|d_k = u) + P(d_k = 0)P(S_0^{N-1}|d_k = u)P(s_k)^{-1}} \quad (23)$$

$$= \frac{3P(d_k = u)P(s_k)}{2P(s_k) + 1} \quad (24)$$

Thus the proof is completed.

References

1. T. Beth and F.Piper, The stop-and-go generator, *Advances in Cryptology, Eurocrypto'94*, vol. 209, Springer-Verlag, 1985, pp. 88-92.
2. D. Coppersmith, H. Krawczys and Y. Mansour, The shrinking generator, *Advances in Cryptology-Crypt'93*, Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1994, pp. 22-39.
3. J.D.Golic and L.O'Connor, Embedding and probability correlation attacks on clock-controlled shift registers, *Advances in Cryptology, Eurocrypto'94*, LNCS 950, A.D.Santis(Ed.), Spring-Verlag, pp.230-243, 1994.
4. J.D. Golic and M.J. Mihaljevic, A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance, *Journal of Cryptology*, 3(3): pp.201-212, 1991.
5. J.D. Golic, Correlation analysis of the shrinking generator, Advances in Cryptology, *Advances in Cryptology-Crypt'2001*, Kilian (Ed.), LNCS 2139, pp.440-457, 2001.
6. E.J. Groth, Generation of binary sequences with controllable complexity, *IEEE Trans. on Inform. Theory* vol. IT-17, No. 3, May 1971, pp. 288-296.
7. C.G. Gunther, Alternating step generators controlled by de Bruijn sequences, *Advances in Cryptology, Eurocrypto'88*, Lecture Notes in Computer Science, vol. 304, Springer-Verlag, 1988, pp. 88-92.
8. E.L. Key, An analysis of the structure and complexity of nonlinear binary sequence generators, *IEEE Trans. on Inform. Theory* vol. IT-22, No. 6, November 1976, pp. 732-736.
9. Dieter Gollman and W.G. Chambers, Clock-controlled shift registers: a review, *IEEE Journal on Selected Areas in Communications*, vol. 7, No. 4, May 1989, pp.525-533.
10. W. Meier and O. Staffelbach, The self-shrinking generator, *Advances in Cryptology, Eurocrypto'94*, Lecture Notes in Computer Science, Springer-Verlag, 1994, pp. 205-214.
11. M.J. Mihaljevic, An approach to the initial state reconstruction of a clock controlled shift register based on a novel distance measure, *Advances in Cryptology, Auscrypto'92*, LNCS 718, J. Seberry and Y. Zheng(Eds.), Springer-Verlag, pp.349-356, 1993.
12. R.A. Rueppel, Stream ciphers, *Contemporary Cryptology, the Science of Inormation*, IEEE Press, 1992, pp.65-134, Gustovos J. Simmons, editor.
13. T. Siegenthaler, Correlation-immunity of nonlinear combing functions for cryptographic applications, *IEEE Trans. Inform. Theory*, vol. IT-30, Sep. 1984, pp. 776-780.
14. M.K. Simon, J.K. Omura, R.A. Scholtz and B.K. Levitt, *Spread Spectrum Communications Handbook*, McGraw-Hill, Inc., Revised version, 1994. Chapter 5.
15. M.V. Zivkovic, An algorithm for the initial state reconstruction of the clock-controlled shift register, IEEE Trans. on Inform. Theory, vol.37, pp. 1488-1490, Sep. 1991.