The Shrinking Generator

Don Coppersmith Hugo Krawczyk Yishay Mansour

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

Abstract. We present a new construction of a pseudorandom generator based on a simple combination of two LFSRs. The construction has attractive properties as simplicity (conceptual and implementation-wise), scalability (hardware and security), proven minimal security conditions (exponential period, exponential linear complexity, good statistical properties), and resistance to known attacks. The construction is suitable for practical implementation of efficient stream cipher cryptosystems.

1 Introduction

We present a new construction of a pseudorandom generator that uses as basic modules a pair of LFSRs. The inherent simplicity of LFSRs, the ease and efficiency of implementation, some good statistical properties of the LFSR sequences, and the algebraic theory underlying these devices turn them into natural candidates for use in the construction of pseudorandom generators, especially, targeted to the implementation of efficient stream cipher cryptosystems. Indeed, many such constructions were proposed in the literature. (See Rueppel's comprehensive survey on LFSR-based constructions of pseudorandom generators and related analysis tools [18]). On the other hand, some of the attractive properties listed above are also the reason for the failure of many of these constructions to meet a good cryptographic strength. In particular, the inherent linearity of LFSRs and the algebraic structure are many times the basis for breaking these systems.

Nevertheless, due to their technological advantages for simple hardware implementation of fast cryptosystems, LFSRs are still studied (and used!) as basic modules for these systems. In particular, the increasing speeds of transmitted information and the simple methods for LFSR parallelization and pipelining indicate that this interest is plausible to persist in the (visible) future. In addition, there is no reason to believe that good *and* simple constructions are impossible.

This paper presents a construction which is attractive in the sense that it is very simple (conceptually and implementation-wise) and passes the *minimal* tests that such constructions require to be worth being considered. We can prove that both the period and linear complexity of the resultant sequences is exponential in the LFSR's length, and that these sequences have some nice distributional statistics (measured in a rigorous way). The construction appears to be free of traditional weaknesses and has stood (up to now) several potential attacks. As said, these are just minimal conditions for the construction to deserve the attention of the cryptographic community, not a "proof" of their ultimate strength.

The practical strength of such a construction can be determined only after public scrutiny. (A desirable goal is to have a real proof of the strength of such a system - i.e., a proof of the unpredictability by efficient means of the generated sequences. Unfortunately, such a proof is not known for any efficient pseudorandom generator and, moreover, such a proof will require a breakthrough in complexity theory. On the other hand, the theoretically well-founded approach of relating the strength of a pseudorandom generator to the hardness of generic or specific problems [4, 20] has led to a beautiful theory and constructions but these are still too impractical for many real-world applications).

1.1 The Construction

Our construction uses two sources of pseudorandom bits to create a third source of pseudorandom bits of (potentially) better quality than the original sources. Here quality stands for the difficulty of predicting the pseudorandom sequence. (In general, through this paper, we use the notion of pseudorandomness and predictability in a rather informal way, although we rigorously analyze and prove some of the random-like properties of the resultant sequences). The sequence we build is a subsequence from the first source where the subsequence elements are chosen according to the positions of '1' bits in the second source. In other words, let $a_0, a_1, \ldots$ denote the first sequence and $s_0, s_1, \ldots$ the second one. We construct a third sequence $z_0, z_1, \ldots$ which includes those bits a_i for which the corresponding s_i is '1'. Other bits from the first sequence are discarded. (Therefore, the resultant sequence is a "shrunken" version of the first one). Formally, for all $k = 0, 1, \ldots$, $z_k = a_{i_k}$, where i_k is the position of the k-th '1' in the sequence $s_0, s_1, \ldots$. We call the resultant pseudorandom generator, *the shrinking generator (SG)* .

This generic idea can be applied to any pair of pseudorandom sources. Here we analyze the construction where the two sources are generated using Linear Feedback Shift Registers (LFSR). LFSRs are very well known structures consisting of a shift register controlled by a clock, which at each clock pulse outputs its most significant bit, shifts its contents in the most significant direction and inputs a bit to its less significant position. This *feedback bit* is computed as a linear combination (over GF(2)) of the bits in the shift register. This linear combination can be fixed (e.g. wired in a hardware implementation) or variable. In the latter case, the linear combination (or *connections*) is defined by a binary vector of the length of the LFSR. (In a hardware implementation this is achieved using, in addition to the shift register, a programmable control register which determines the shift register cells that are connected to the XOR circuit).

We denote by A the first LFSR in our construction, and by S (for *S*elector) the second one. $|A|$ and $|S|$ denote their lengths and the sequences they produce (after fixing the connections and initial contents of the registers) are denoted $a_0, a_1, \ldots$ and $s_0, s_1, \ldots$, respectively. We also refer to these sequences as *A-sequence* and *S-sequence*. Finally the resultant shrunken sequence is denoted by $z_0, z_1, \ldots$, and called the *Z-sequence*.

This construction is well defined for both fixed and variable connection LFSRs. In general, we recommend the use of variable connections both for security and flexibility. This issue is discussed throughout this paper in the appropriate places. Let us mention that in the case of a fixed connection implementation only the seeds (i.e., the initial contents of the shift registers) for the LFSRs A and S constitute the *secret key* for the pseudorandom generator (or the encryption/decryption key, when used as a stream cipher). If variable (programmable) connections are used then the value of these connections is also part of the key.

1.2 Properties

We analyze some of the properties of the resultant LFSR-based shrinking generator. We show that the period of the Z-sequences is exponential in both $|A|$ and $|S|$, and that its linear complexity is exponential in $|S|$. The linear complexity of a sequence is the length of the shortest LFSR that generates that sequence (or equivalently, the shortest recursive linear dependence over GF(2) satisfied by the sequence bits). The importance of this property is that sequences with low linear complexity are easily predictable (see section 2) and constructions based on LFSRs tend to preserve much of the linearity inherent to LFSRs. The above properties equally hold for fixed or variable connections. On the other hand, our statistical analysis of these sequences takes into account variable connections (chosen with uniform probability over the set of primitive connections[1]). We show that the space of resultant sequences has some of the necessary statistical properties for a pseudorandom generator: low correlation between the sequence bits, normalized appearance of 0's and 1's, and balanced distribution of subpatterns. Our statistical analysis uses Fourier analysis and ε-biased distributions as the main tools. The period and linear complexity bounds are proven mainly through algebraic techniques.

We stress, again, that all these properties are only *necessary* (but far from sufficient) conditions on the cryptographic strength of the pseudorandom generator. They just show that the elemental goals for an LFSR-based construction are achieved, namely, the destruction of the linearity while preserving the good statistical properties.

In section 4 we present some attacks and analyze their effect on our construction. These attacks work in time exponential in the length of register S, and indicate an effective key length bound of about half of the total key length.

Practical considerations regarding the implementation and practical use of our generator are discussed in section 5. In particular, we show how the problem

[1] Connection vectors for LFSRs are closely related to polynomials over GF(2). Best connections for LFSRs are those which correspond to primitive polynomials of the same degree as the LFSR's length. In that case, the sequence generated by the LFSR has *maximal length*, namely, a period of $2^n - 1$, where n is the length of the register [9]. Throughout this paper we implicitly assume a construction of the SG using primitive connections. Such connections are easy to find using probabilistic methods, e.g. see [17].

of irregular rate of the output bits present in our basic scheme can be solved at a moderate cost in hardware implementation.

Finally, in section 6 we discuss some existing alternative constructions and their relation to the shrinking generator.

2 Period and Linear Complexity

In this section we prove exponential bounds on the period and linear complexity of sequences produced by the shrinking generator. In the case of the period this bound is tight; for the linear complexity there is a gap by a factor of 2 between the lower and upper bound.

The importance of a long period is to avoid the repetition of the sequence after short period of times. An exponentially large linear complexity avoids one of the more generic attacks on pseudorandom sequences and/or stream ciphers. There is no need to even know the way a sequence is generated in order to break it through its linear complexity. Any sequence of linear complexity ℓ can be entirely reconstructed out of 2ℓ known bits by using the Berlekamp-Massey algorithm, which in time $O(\ell^2)$ finds the shortest linear dependency satisfied by the sequence (a-priori knowledge of the value of ℓ is not necessary). See, e.g. [3]. (On the other hand, high linear complexity by itself is far from being an indication of the sequence unpredictability. It suffices to mention that the sequence 00...001 has linear complexity as the length of the sequence).

Our results on the period and linear complexity of sequences generated with the shrinking generator are stated in the next theorems.

Theorem 1. *Let A and S form a shrinking generator Z. Denote by T_A , T_S, the periods of the A- and S- sequences respectively. If*

- *A and S are maximal length (i.e. have primitive connections)*
- *$(T_A, T_S) = 1$*

then the shrunken sequence Z has period $T_A \cdot 2^{|S|-1} = (2^{|A|} - 1) \cdot 2^{|S|-1}$.

Note: S must not be of maximal length. In the general case the period of the Z-sequence is $T_A \cdot W_S$, where W_S is the number of 1's in a full period of S. If both A and S are of maximal length then the condition $(T_A, T_S) = 1$ is equivalent to $(|A|, |S|) = 1$. For the next theorem S may also not be a maximal length sequence but we do need that W_S be a power of 2.

Theorem 2. *Under the conditions of Theorem 1, the shrunken sequence Z has linear complexity LC, where $|A| \cdot 2^{|S|-2} < LC \leq |A| \cdot 2^{|S|-1}$*

In the following proofs of theorems 1 and 2 we use some well-nown algebraic facts about sequences produced by LFSRs. These properties can be found in many textbooks (e.g. [9, 14]).

Notation: For the sake of readiness we use the following notation through these proofs: $a(i)$ denotes the A-sequence, $s(i)$ the S-sequence, and $z(i)$ the shrunken sequence Z. By k_i we denote the position of the i-th '1' in the S sequence. In other words, $\forall i, z(i) = a(k_i)$. We denote by W_S the number of 1's in a full period of S. For a maximal length sequence S this number is $2^{|S|-1}$.

Proof of Theorem 1:

Assumption: For simplicity of the proof we assume

$$|S| \leq T_A \;\; (i.e.\ |A| > log|S|). \tag{1}$$

The following fact is immediate from the definition of the shrunken sequence Z. **Fact 1:** Advancing W_S elements in the sequence z results in advancing T_S elements in the sequence a. Formally, $z(i + W_S) = a(k_i + T_S)$.
In general, for all $j = 0, 1, \ldots,$

$$z(i + jW_S) = a(k_i + jT_S). \tag{2}$$

Fact 2: Let k and k' be any pair of indices. If for all j: $a(k+jT_S) = a(k'+jT_S)$, then T_A divides $k - k'$.

Proof: Because of the A-sequence being of maximal length and $(T_A, T_S) = 1$ then the sequence $a(k + jT_S)$, $j = 0, 1, \ldots,$ is also maximal length and thus its period is T_A . ◇

Denote by T the (minimal) period of the sequence z. Clearly, the sequence z becomes periodic after $T_A \cdot W_S$ elements (since then both sequences a and s simultaneously complete a period). Therefore, T divides $T_A \cdot W_S$. We now proceed to show that $T_A \cdot W_S$ divides T.
By definition of T, for all i , $z(i) = z(i + T)$. In particular, for all i and j, $z(i + jW_S) = z(i + T + jW_S)$. Using (2) we get, for all i and j : $a(k_i + jT_S) = a(k_{i+T} + jT_S)$. Using Fact 2, we have

$$\forall i,\ T_A \ divides\ k_{i+T} - k_i. \tag{3}$$

Next step is to show, that (3) is possible only if W_S divides T. We reformulate (3) as:

$$\forall i, \exists j_i : k_{i+T} = k_i + j_i T_A \tag{4}$$

Putting $i + 1$ instead of i in (4) we get

$$k_{i+1+T} = k_{i+1} + j_{i+1} T_A \tag{5}$$

Subtracting (4) from (5) we get:

$$\forall i,\ k_{i+T+1} - k_{i+T} = k_{i+1} - k_i + (j_{i+1} - j_i) T_A. \tag{6}$$

Notice that k_{i+T} and k_{i+T+1} , as well as k_i and k_{i+1} , are the positions of consecutive 1's in the S-sequence. If $j_{i+1} - j_i$ would be different than zero, it would imply the existence of at least T_A consecutive zeros in the S-sequence, which is impossible by assumption (1). Therefore we get $j_{i+1} - j_i = 0$, and then for all i, $k_{i+T+1} - k_{i+T} = k_{i+1} - k_i$.
The later implies that the subsequence of s starting at $s(k_i)$ is identical to the subsequence starting at $s(k_{i+T})$. This means that T_S divides $k_{i+T} - k_i$, or

equivalently, that the number of elements in the S-sequence between $s(k_i)$ and $s(k_{i+T})$ is a multiple of its period . But then the number of 1's in this segment is a multiple of W_S. On the other hand, the number of 1's is exactly T, thus proving that W_S divides T.

Let t be such that

$$T = tW_S. \tag{7}$$

We have, for all j:

$$a(k_0) = z(0) = z(jT) = z(jtW_S) = a(k_0 + jtT_S). \tag{8}$$

Last equality follows from (2). We got that for all $j : a(k_0) = a(k_0 + jtT_S)$. This implies that T_A divides tT_S , and since $(T_A, T_S) = 1$, then T_A divides t. From (7) we get $T_A \cdot W_S$ divides T. □

The lower bound in the following proof of Theorem 2 is derived using the proven exponential period through an elegant technique from Gunther [10].

Proof of Theorem 2:

Upper bound on the linear complexity: Let z denote the variable corresponding to the sequence Z. To show an upper bound on the linear complexity of the sequence Z it suffices to present a polynomial $P(\cdot)$ for which $P(z) = 0$ (i.e. the coefficients of P represent a linear relation satisfied by the elements of Z). The variable z^{W_S} denotes the sequence Z decimated by W_S, i.e. the sequence $z(jW_S), j = 0, 1, \dots$ Fact 1 in the proof of Theorem 1 states that this decimation, written in terms of the A-sequence, results in a sequence of the form $a(i + jT_S)$. Since we assume $(T_S, T_A) = 1$, the latter is a maximal length sequence with same linear complexity as the original A-sequence, and then it satisfies a polynomial $Q(\cdot)$ of degree $|A|$. But then also the decimated sequence z^{W_S} satisfies the polynomial, i.e. $Q(z^{W_S}) = 0$. Therefore, we have found a polynomial $P(z) = Q(z^{W_S})$ of degree $|A| \cdot W_S$, such that $P(z) = 0$, and then the linear complexity of the Z-sequence is at most $|A| \cdot |W_S| = |A| \cdot 2^{|S|-1}$.

Lower bound on the linear complexity: Let $M(z)$ denote the minimal polynomial of z. Since the sequence Z satisfies $Q(z^{W_S}) = 0$, we have that $M(z)$ must divide $Q(z^{W_S})$. Since $W_S = 2^{|S|-1}$, we have $Q(z^{W_S}) = Q(z^{2^{|S|-1}}) = (Q(z))^{2^{|S|-1}}$, and then $M(z)$ must be of the form $(Q(z))^t$ for $t \le 2^{|S|-1}$. Assume $t \le 2^{|S|-2}$. Then, $M(z)$ divides $(Q(z))^{2^{|S|-2}}$. Since $Q(z)$ is an irreducible polynomial of degree $|A|$ it divides the polynomial $1 + x^{T_A}$. Therefore, $M(z)$ divides $(1 + x^{T_A})^{2^{|S|-2}} = 1 + x^{T_A \cdot 2^{|S|-2}}$, but then the period of Z is at most $T_A \cdot 2^{|S|-2}$ contradicting Theorem 1. Therefore, $t > 2^{|S|-2}$ and the lower bound follows. □

3 Statistical Properties

3.1 Background

In this subsection we bring the required background on the techniques used in our analysis of the statistical properties of the shrinking generator; specifically, the notions of Fourier Transform (for Boolean domains) and ε-bias distributions.

Fourier Transform Boolean functions on n variables are considered as real valued functions $f : \{0,1\}^n \to \{-1,1\}$. The set of all real functions on the cube is a 2^n–dimensional real vector space with an inner product defined by:

$$< g, f >= 2^{-n} \sum_{x \in \{0,1\}^n} f(x)g(x) = E(gf)$$

(where E is expectation) and as usual the *norm* of a function is defined: $\|f\| = \sqrt{< f, f >}$, which is the Euclidean norm.

The basis of the cube $\mathbf{Z}_2^n$ is defined as follows: For each subset S of $\{1, \cdots, n\}$, define the function χ_S:

$$\chi_S(x_1, \cdots, x_n) = \begin{cases} +1 & \text{if } \sum_{i \in S} x_i \text{ is even} \\ -1 & \text{if } \sum_{i \in S} x_i \text{ is odd} \end{cases}$$

The following properties of this basis functions can be easily verified:

- For every A, B: $\chi_A \chi_B = \chi_{A \Delta B}$, where $A \Delta B$ is the symmetric difference of A and B.
- The family $\{\chi_S\}$ for all $S \subset \{1 \cdots n\}$ forms an orthonormal basis, i.e., if $A \neq B$, then $< \chi_A, \chi_B >= 0$, and for every A, $< \chi_A, \chi_A >= 1$.

Any real valued function on the cube can be uniquely expressed as a linear combination of the basis functions, i.e. $\sum_S c_S \chi_S$, where c_S are real constants. The *Fourier transform* of a function f is the expression of f as a linear combination of the χ_S's. For a function f and $S \subset \{1, \cdots, n\}$, the S'th Fourier coefficient of S denoted by $\hat{f}(S)$ is what was previously called c_S, i.e., $f = \sum_S \hat{f}(S)\chi_S$.

Since the χ_S's are an orthonormal basis, Fourier coefficients are found via:

$$\hat{f}(S) =< f, \chi_S >$$

For boolean f this specializes to:

$$\hat{f}(S) = Pr[f(x) = \oplus_{i \in S} x_i] - Pr[f(x) \neq \oplus_{i \in S} x_i]$$

where $x = (x_1, x_2, \ldots, x_n)$ is chosen uniformly at random.

ε-biased Distributions We consider a distribution function as a function from $\{0,1\}^n$ to the interval $[0,1]$. Given a probability distribution μ, then $\sum_x \mu(x) = 1$ and $\mu(x) \geq 0$. We can treat μ as any other function, and consider its Fourier coefficients. For example the uniform distribution is $U(x) = \frac{1}{2^n}$, which implies that $\hat{U}(S) = 0$, for $S \neq \emptyset$, and $\hat{U}(\emptyset) = \frac{1}{2^n}$.

A distribution is ε-bias if it is "close" to the uniform distribution in the following sense.

Definition 3. A distribution μ over $\{0,1\}^n$ is called an ε-bias distribution if for every subset $S \subset \{1 \ldots n\}$, $|\hat{\mu}(S)| \leq \varepsilon 2^{-n}$.

The notion of ε-bias distribution was introduced in [16], the main motivation being the derandomization of randomized algorithms, and the construction of small sample spaces that approximate the uniform distribution.

The following theorem from [1] connects LFSRs and ε-bias distributions.

Theorem 4. *([1]) Consider the distribution $\mathcal{D}(m,n)$ of strings of length n output by a LFSR A of length m, where the connections for A are chosen with uniform probability among all primitive polynomials over GF(2) of degree m, and the seed for A is chosen uniformly over all non-zero binary strings of length m. Then, $\mathcal{D}(m,n)$ is an $\frac{n-1}{2^m}$-bias distribution.*

Definition 5. Let f be a function from $\{0,1\}^n$ to the real numbers. Define $L_1(f) = \sum_S |\hat{f}(S)|$.

The following lemma relates ε-bias distributions and the norm $L_1(f)$. (See [13].) The function f can be seen as a test for distinguishing the distribution μ from the uniform distribution. The lemma states an upper bound on the quality of distinction, and therefore it is useful for tests of pseudorandomness.

Lemma 6. *([13])*

$$|E_U[f] - E_\mu[f]| \leq \varepsilon L_1(f)$$

where U is the uniform distribution and μ is an ε-bias distribution.

Proof. By simple arithmetic

$$E_\mu[f] = \sum_S \hat{f}(s)\hat{\mu}(S) = \hat{f}(\emptyset)\hat{\mu}(\emptyset) + \sum_{S\neq\emptyset} \hat{f}(S)\hat{\mu}(S).$$

Note that by definition $\hat{f}(\emptyset) = E_U[f]$. Since $\hat{\mu}(\emptyset) = 1/2^n$,

$$|E_U[f] - E_\mu[f]| = \sum_{S\neq\emptyset} \hat{f}(S)\hat{\mu}(S) \leq \varepsilon L_1(f).$$

Here we used the fact that each $\hat{\mu}(S)$ is bounded by ε. □

L_1 norm Lemma 6 is useful if we can upper bound the value $L_1(f)$. In this section we present some methods for bounding the L_1 norm of a function. The following technical Lemma gives a tool for doing that.

Lemma 7. *Let f and g be functions from $\{0,1\}^n$ to the real numbers. Then, $L_1(fg) \leq L_1(f)L_1(g)$ and $L_1(f+g) \leq L_1(f) + L_1(g)$.*

For many simple functions we can show that the L_1 is small. Here are a few examples.

Lemma 8.

- *Let $sum(x) = \sum_{i=1}^{n} x_i$, then $L_1(sum) = n$.*
- *Let $AND(x) = \prod_i x_i$, then $L_1(AND) = 1$.*
- *For $B \in \{0,1,*\}^n$ we define a template $\texttt{template}_B(x) = 1$ iff x and B agree on each 0 or 1 in B, i.e. for each $b_i \neq *$ then $b_i = x_i$. (For example $\texttt{template}_{10*1*}(10110) = 1$ while $\texttt{template}_{10*1*}(00110) = 0$.) For any $B \in \{0,1,*\}^n$ then $L_1(\texttt{template}_B) = 1$.*

Proof. For the *sum*, we can rewrite it as $n/2 + \sum_i \chi_{\{i\}}(x)/2$. Using the additivity of the L_1 the claim follows.

Note that the *AND* function is either 0 or 1 (and not ± 1). We rewrite the *AND* to be

$$\prod_{i=1}^{n} \frac{1 - \chi_{\{i\}}(x)}{2}$$

Note that $L_1(\frac{1-\chi_{\{i\}}(x)}{2}) = 1$, and the claim follows from the multiplicative properties of L_1 (see Lemma 7).

For $\texttt{template}_B(x)$ the proof is the same as for the AND function. We rewrite the function as,

$$\texttt{template}_B(x) = \Big(\prod_{i:b_i=1} \frac{1 - \chi_{\{i\}}(x)}{2}\Big)\Big(\prod_{j:b_j=0} \frac{1 + \chi_{\{i\}}(x)}{2}\Big)$$

and again we use the multiplicative property of the L_1. □

3.2 Applications to LFSR with variable connections

In this subsection we show that LFSR with variable connections have many properties that resemble random strings. In fact the only property that we use is that LFSR where the connections are chosen at random generates an ε-bias distribution, with exponentially small ε (see Theorem 4).

Theorem 9. *Let A be an LFSR where the connections for A are chosen with uniform probability among all primitive polynomials of degree m over GF(2). Let X be the sum of n different bits $i_1, \ldots, i_n$ in the A-sequence (we assume that $i_j \leq 2^m - 1$). Let $Y = \sum_{i=1}^{n} y_i$ where y_i are i.i.d. $\{0,1\}$-random variables and $Prob[y_i = 1] = 1/2$. Then, the expected value of X is at most $\frac{n^2}{2^m}$. The difference between the variance of X and Y is bounded by $\frac{n^3+n^2}{2^m}$. Furthermore, $|E[X^k] - E[Y^k]| \leq \frac{n^k}{2^m}$.*

Proof. By definition $X = sum(a_{i_1}, \ldots, a_{i_n})$. By Lemma 8, $L_1(X) = n$. Also, $L_1(X^2) \leq n^2$ and $L_1(X^k) \leq n^k$. The theorem follows from Theorem 4 and Lemma 6. □

The following theorem applies the ideas of a general template to an LFSR sequence and shows that the probability that the template appears is close to the probability it appears in a random string.

Theorem 10. *Let A be an LFSR where the connections for A are chosen with uniform probability among all primitive polynomials of degree m over GF(2). Let X be the first n output bits of A and Y a random string of n bits. Let $B \in \{0,1,*\}^n$ be a template. Then*

$$|E[\mathtt{template}_B(X)] - E[\mathtt{template}_B(Y)]| \leq \frac{n}{2^m}.$$

Proof. By Lemma 8, we have that $L_1(template_B) = 1$. By Theorem 4 the string X is an ε-bias distribution, with $\varepsilon \leq \frac{n}{2^m}$. The theorem follows from Lemma 6. □

When we will consider the selector register S of the shrinking generator it would be important to argue how many bits we should consider in order to generate k output bits. The following theorem shows that the expected number is $O(k)$.

Theorem 11. *Let S be an LFSR where the connections for S are chosen with uniform probability among all primitive polynomials of degree m over GF(2). Let $i_k(S)$ be the location of the kth 1 bit in S, then the expectation*

$$E_S[i_k(S)] = O(k).$$

The proof of the above theorem will be given in the final version.
Remark: Note that all the proofs in this subsection were based only on the ε-bias properties, and therefore would hold for any ε-bias distribution.

3.3 Applications to the Shrinking Generator

In this subsection we apply the results in the previous subsection to the shrinking generator. Basically we show that the good random-like properties that existed in LFSR with variable connection remain in the shrinking generator. (Clearly, the shrinking generator has other essential properties not present in LFSR sequences, e.g. the exponential linear complexity.)

The following is a simple corollary of theorem 9, which states that the moments of the output of the shrinking generator are very close to the moments of a random string.

Corollary 12. *Let Z be a sequence generated by a shrinking generator with registers A and S. Let X be the sum of consecutive n bits in the Z-sequence (we assume that $n|S| \leq 2^{|A|}$). Let $Y = \sum_{i=1}^{n} y_i$ where y_i are i.i.d. $\{0,1\}$-random variables and $Prob[y_i = 1] = 1/2$. Then, the expected value of $|X|$ is at most $\frac{n^2}{2^{|A|}}$. The difference between the variance of X and Y is bounded by $\frac{n^3+n^2}{2^{|A|}}$. Furthermore, $|E[X^k] - E[Y^k]| \leq \frac{n^k}{2^{|A|}}$.*

Proof. Fix a specific S-sequence. The consecutive n bits in Z were generated by some n non consecutive different bits in the A-sequence, denote their indeces in this sequence by $i_1, \ldots, i_n$. Since $n|S| \leq 2^{|A|}$, we are in the same period of A, i.e. $i_j \leq 2^{|A|} - 1$. Since X is the sum of those bits, the corollary follows from Theorem 9. □

The following theorem shows that each template is distributed similarly in the output of the shrinking generator and a random string.

Theorem 13. *Let Z be a sequence generated by a shrinking generator with registers A and S. Let X be the first n bits in Z and Y be a random string of n bits. Let $B \in \{0,1,*\}^n$ be a template. Then*

$$|E[\texttt{template}_B(Z)] - E[\texttt{template}_B(Y)]| = O(\frac{n}{2^{|A|}}).$$

Proof. The bits of X come from the first $i_n(S)$ bits of A, where $i_n(S)$ is the index of the nth '1' bit in the S-sequence. Given S and B we can create a template B_S of size $i_n(S)$ for A (we simply put $*$ in any location that S is 0, and copy B in the locations where S is 1).

Note that $\texttt{template}_B(X) = \texttt{template}_{B_S}(A)$, once we fix S. Therefore it is sufficient to bound

$$\sum_S Prob[S] \left| E_A[\texttt{template}_{B_S}(A)] - E_Y[\texttt{template}_B(Y)] \right|.$$

By Theorem 10 the difference between the expectation is bounded by $i_n(S)/2^{|A|}$. Therefore,

$$\sum_S Prob[S] |E[\texttt{template}_{B_S}(A)] - E[\texttt{template}_B(Y)]| \leq$$

$$\leq \sum_S Prob[S] \frac{i_n(S)}{2^{|A|}} = \frac{E_S[i_n(S)]}{2^{|A|}} = O(\frac{n}{2^{|A|}}).$$

The last identity follows from Theorem 11. □

We now show some interesting applications of the above theorem. First we consider correlation between pairs of output bits. The correlation between two bit positions is the difference (in absolute value) between the probability that the two bits are equal and the probability that they differ.

Corollary 14. *Let Z be a sequence generated by a shrinking generator with registers A and S. Let X_1, X_2 be two bits in the Z-sequence that are at distance ℓ. The correlation between X_1 and X_2 is bounded by $O(\frac{\ell}{2^{|A|}})$.*

Proof. Simply use the four templates $\sigma_1 \overbrace{* \cdots *}^{\ell} \sigma_2$, where $\sigma_1, \sigma_2 \in \{0,1\}$, and apply Theorem 13. □

The next corollary shows that the distribution of patterns is almost uniform.

Corollary 15. *Let P be any binary string (pattern) of k bits and let X_k be the k consecutive bits in the Z-sequence. The probability that $X_k = P$ is in the range $2^{-k} \pm O(\frac{k}{2^{|A|}})$.*

Note that this corollary is a special case of Theorem 13.

4 Attacks

In this section we present some attacks on the shrinking generator. These attacks indicate an effective key length of the length of register S, or about twice this length if the connections for the registers are part of the key (i.e. the connections are variable and secret). More details on these and other attacks will be presented in the final version of this paper.

4.1 Attacking through S

If the connections for both S and A are known then one can exhaustively search for S's seed; each such seed can be expanded to a prefix of the S-sequence using the connection of S. Let $n = |A|$ and suppose we expand the S-sequence until its n-th '1' is produced. From this prefix, and from knowledge of a corresponding n-long prefix of the Z-sequence, one derives the value of n (non-consecutive) bits in the A-sequence. Since A's connections are known then A's seed can be recovered given these n bits by solving a system of linear equations (in general, the dimension of this system is about $n/2$ since about half of the seed bits – corresponding to 1's in S – are known). Therefore the attack's complexity is exponential in $|S|$ and polynomial in $|A|$, or more precisely, $O(2^{|S|} \cdot |A|^3)$.

If the connections of A are secret as we recommend, then the above procedure does not work since in order to write the system of equations one needs to know these connections. In this case the following attack avoids doing an exhaustive search on A's connections. This attack tries all possible seeds and connections for S (assuming S's connections are secret). Each pair of seed and connections for S is used to expand the seed into a t-long prefix of the S-sequence, for some integer t. With this prefix and sufficiently many bits (about $t/2$ bits) from the Z-sequence (known plaintext) it is possible to generate the first t bits of the product sequence $p_i = a_i \cdot s_i$. (Notice that bits from the A-sequence corresponding to positions of 1's in the S-sequence are known using the known part of the Z-sequence, and positions in which the $s_i = 0$ are also 0's in the product sequence). The interesting property of this product sequence is that its linear complexity is at most $|A| \cdot |S|$ (see [18]) and therefore having $t = 2 \cdot |A| \cdot |S|$ in the above attack suffices to find the whole product sequence p_i. The cost is quadratic in $|A| \cdot |S|$. This information together with the S-sequence, which is known, permits deriving the full sequence Z_i. Therefore the cost of the attack is the number of seeds and connections to be tried for S (about $2^{2|S|}/|S|$) times the complexity of recovering p_i through its linear complexity (i.e. $O((|A| \cdot |S|)^2)$). The necessary amount of plaintext (i.e. bits from Z) is $|A| \cdot |S|$. As before this attack indicates an effective key length of at most twice the length of S, or about half of the total key length.

4.2 Linear Complexity

Attacking the SG through its linear complexity requires the knowledge of an exponential in $|S|$ number of bits from the sequence, more precisely, $2^{|S|-2} \cdot |A|$

bits at least (see Theorem 2). On the other hand, the typically quadratic work that takes to derive the sequence from a prefix of that length is not necessary here. Having $2^{|S|} \cdot |A|$ consecutive bits from the sequence one can derive the whole sequence. The proof of Theorem 1 indicates that a decimation of the Z-sequence by factors of $W_S = 2^{|S|-1}$ implies the decimation of the A-sequence by a factor of $T_S = 2^{|S|} - 1$. Therefore, from $z(i + jW_S)$, $j = 0, 1, \ldots, 2 \cdot |A| - 1$ one derives $z(i + jW_S)$, for all j.

The complexity to break the whole sequence in this way is $O(2^{|S|} \cdot |A|^2)$ (even if the connections are secret). In addition to this computational complexity this attack requires $2^{|S|} \cdot |A|$ consecutive bits from the sequence. In any case, the parameters for the SG should be chosen such that collecting this many number of sequence bits be infeasible.

4.3 Other Attacks

The more traditional attacks on LFSR-based construction seem not to apply to our construction due to its different nature. These attacks include the analysis of boolean functions used for the combination of LFSR outputs, the correlation of generated bits relative to subcomponents in the system, and others (See [18] for more details on these attacks and their applications).

It is worth mentioning that a typical weakness of LFSR-based systems is encountered in implementations where the connection polynomials are chosen to be very sparse (i.e. only a few coefficients chosen to be non-zero). In this case, special attacks can be mounted taking advantage of this fact. We recommend not to implement any of these systems in such a way, including ours. (In a hardware implementation having sparse connections may be advantageous only if the connections are fixed). On the other hand, most of these attacks will work not only if the connection polynomial itself is sparse, but also if this polynomial has a multiple of moderately large degree which is sparse. We can mount special attacks on our system against such sparse multiples, although they are all exponential in $|S|$. Again these attacks are more relevant to fixed connection implementations, where heavy preprocessing can be done against the particular connections, than in the case of variable connections.

5 Practical Considerations

5.1 Overcoming Irregular Output Rate

The way the SG is defined, bits are output at a rate that depends on the appearance of 1's in S output. Therefore, this rate is on *average* 1 bit for each 2 pulses of the clock governing the LFSRs. This problem has two aspects. One is the reduced throughput relative to the LFSRs speed, the other the irregularity of the output. We show here that this apparently practical weaknesses can be overcome at a moderate price in hardware implementation (on the other hand, these "weaknesses" give most of the cryptographic strength to this construction).

We stress that this hardware cost is usually less than the required for adding more LFSRs (even one) to the construction (as many constructions do).

In order to achieve an average of 1 bit per clock pulse, the LFSRs can be easily speeded up with a very moderate cost in hardware: only the XOR tree is to be replicated (this is true also if the connections are variable!). Notice that whether this speed-up is necessary depends on the relation between the LFSR clock speed and the required throughput from the SG (e.g., when used in a stream cipher system this throughput depends on the data speed). If the clock is fast enough this speed-up may be not necessary at all. On the other hand, for fast data encryption a speedup mechanism may be necessary regardless of the reduced throughput of our construction.

The problem of irregular output rate can be serious in real-time applications where repeated delays are not acceptable. Fortunately, this problem can be also solved at a moderate cost. The solution is to use a short buffer for the SG output intended to gather bits from the SG output when they abound in order to compensate for sections of the sequence where the rate output is reduced. In [11] Markov analysis is applied to analyze the influence of such a buffer for the output rate of the SG. It is shown that even with short buffers (e.g., 16 or 24 bits) and with a speed of the LFSRs of above twice the necessary throughput from the SG the probability to have a byte of pseudorandom bits not ready in time is very small. (Examples are a probability of $5 \cdot 10^{-3}$ for buffer of size 16 and speedup factor of 9/4, or a probability of $3 \cdot 10^{-7}$ for a buffer of size 24 and speedup factor of 10/4. These probabilities decrease exponentially with increasing buffer sizes and speedup factors). We note that in most implementations of stream ciphers, some buffering naturally exist because of data coming in blocks of a given size (e.g depending on the bus width). Therefore the above technique may add none or very little bits to the buffer size. In many cases the above small probabilities of delayed pseudorandom bits is affordable. In cases it is not, we propose filling the missing bits with arbitrary values (e.g. alternate 0's and 1's) which can hardly hurt with a miss probability of $3 \cdot 10^{-7}$ or so. An alternative (but somewhat less simple) heuristic solution is to periodically buffer some bits of the A-sequence corresponding to 0's in S in order to use them for filling the missing bits in case of need.

5.2 Fixed vs. Variable Connections

Throughout the paper we have recommended several times the use of variable connections for the LFSRs A and S. Although variable connection do not influence the period and linear complexity of the resultant sequences, their advantage is apparent from the attacks discussed in section 4 (e.g., to avoid attacks using heavy precomputation for analyzing the particular connections, or the preparation of big preprocessing tables), and from the statistical analysis of section 3. They may be also beneficial in standing future attacks to the system.

In addition to these security advantages, using variable connections provides a large degree of flexibility to the construction (this is true for other LFSR-based construction rogramming of these connections the security of the sytem can be tuned down or up with no change in the hardware. This is most important for systems where versions of different security levels use the same physical device (e.g. cryptographic systems sold in different countries with different levels of permitted security). Tuning down the security is done through a virtual shortening of the registers by loading zeros into the most significant locations of the connection registers.

We stress that while there is a cost in hardware associated with the connection registers, this cost is compensated with the possible choice of shorter registers when using variable connections, and by the above advantages. Moreover, having shorter registers implies having shorter seeds. The latter are the part of the key which keeps changing with bit generation while the connections are kept unchanged for long periods. Having shorter seeds help the key management and synchronization aspects (especially, when used in a stream cipher cryptosystem).

6 Discussion and Related Work

LFSR-based constructions are encountered today in many practical systems, especially for implementation of stream ciphers. Because of their conceptual and implementation simplicity they will keep being attractive; in particular, since they are simple to parallelize and pipeline they are natural candidates for high speed encryption, too. Moreover, LFSRs are widely used in non-cryptographic applications (coding, CRCs, whitening, etc), and then it's plausible to have new technologies supporting the construction of efficient LFSRs. In addition, LFSR-based constructions have the important practical property that the amount of required hardware can be traded-off against different levels of security; on the other hand, same hardware can handle different levels of security (see Section 5.2). From a theoretical point of view, it is puzzling whether such simple constructions may have a good cryptographic strength. For all these reasons it seems important to have some good construction(s) well evaluated by the cryptographic community. The one presented in this paper may be a good candidate for evaluation, as it compares to the best existing alternatives, and may have the potential to prove better.

Interesting examples of existing LFSR-based constructions for comparison with the shrinking generator are Gunther's *alternating step generator* [10], and some of the clock-controlled generators discussed in [8], in particular the *1-2 generator*. They have similar proven properties as ours, but both are developments of the weak "stop-and-go generator" [2]. This generator uses two LFSRs where the first one is used to control the clock of the second LFSR. Therefore, a '1' output by the first LFSR causes the second one to shift its state, while a '0' implies that the state keeps unchanged (but still a bit, same as the previous one, is output). The output of this second LFSR is then the output of the stop-and-go generator; and the weakness of the repeated bit is clear. The 1-2 generator solves this problem by shifting one bit of the second LFSR when the first LFSR outputs '0', and shifting two bits when the first LFSR outputs '1'.

Gunther's construction uses three registers and outputs the bitwise XOR of two stop-and-go sequences controlled by the same third LFSR. Actually, Gunther's generator is equivalent to a generator that merges two LFSR sequences S_0 and S_1 according to the '0's and '1's output by a third LFSR (a '0' implies taking next bit from S_0 a '1' implies taking next bit from S_1). This construction has the nice property that each bit in the output may (a-priori) correspond to any of the two sequences; on the other hand, it lacks the property of omitting bits from these sequences.

One advantage of Gunther's generator is that it guarantees one output bit per LFSR clock pulse, but it pays for it with a third LFSR. In our construction, the hardware prize we pay in order to regulate the output rate (see section 5.1) is usually lower than introducing a third LFSR (this is due to the fact that XOR gates usually cost significantly less than memory elements). Moreover, this third LFSR brings the effective key length of Gunther's scheme to one third of the total length (it can be broken through exhaustive search on only one of the three registers). The 1-2 generator has the effect of omitting bits through its irregular clocking, but this omission is by nature very local, e.g. one of any two consecutive bits originally output by one of the LFSRs appears in the generator's output sequence.

Locality appears in other versions of clock-controlled generators as well. In our construction the uncertainty about omission of bits is significantly superior (e.g., in clock-controlled constructions t bits from the control sequence determine the original locations in the other register of t output bits; in the shrinking generator, however, $2t$ bits in the selecting register S are necessary (on average) to determine the original locations of t bits in the Z-sequence). In particular, notice that the shrinking generator is *not* a special case of a clock-controlled generator (e.g., its output is not synchronized with the selecting register as it is the case in any clock-controlled scheme). Moreover, the general techniques on clock-controlled generators [8] do not directly apply to our construction.

Finally, the work by Golic and Zivkovic [7] shows that most *irregularly decimated* LFSR-sequences have high linear complexity; however, their result is non-constructive by nature and has no implication on our construction.

We stress that the omission of bits is important not only in LFSR-based constructions but also in other constructions as well. On the other hand, not every scheme for omission of bits is effective (e.g. a decimated LFSR sequence is as bad as the original sequence itself). For the linear congruential number generator outputting all of the bits of a generated number makes the task of breaking it a very easy one [5]. Even if some bits are omitted but a block of consecutive bits are output, efficient predicting methods are known [6, 19]. The extended family of congruential generators is efficiently predictable if sequence elements are output with no omission [12], but no efficient methods are reported for these sequences if part of the bits are omitted. It is an interesting open problem what can be proven for a shrinking generator based on congruential generators. Finally, let us mention that the idea of outputting individual bits of a sequence, is best captured by the notion of *hard bits* of a one-way function, a notion that plays a central role in the construction of complexity-theory based pseudorandom generators (see [4, 20] and subsequent works). It would be interesting to know whether the shrinking generator applied to two ε-predictable sequences guarantees, in general, a third sequence which is ε'-predictable for $\varepsilon' < \varepsilon < \frac{1}{2}$. (Roughly speaking, a sequence is ε-predictable if no polynomial-time algorithm can predict it with probability greater than $\frac{1}{2} + \varepsilon$).

Acknowledgement

We owe special thanks to Celso Brites, Amir Herzberg and Shay Kutten for their help and involvement during the development of the shrinking generator. Many people have contributed in different ways to this investigation; they include: Aaron Kershenbaum, Ilan Kessler, Ronny Roth, Kumar Sivarajan, and Moti Yung. To all of them many thanks.

References

1. Noga Alon, Oded Goldreich, Johan Hastad, and Rene Peralta. Simple constructions of almost k-wise independent random variables. In 31th *Annual Symposium on Foundations of Computer Science, St. Louis, Missouri*, pages 544–553, 1990.
2. Beth, T., and Piper, F., "The stop-and-go Generator", in *Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt '84*, Berlin: Springer-Verlag, 1985, pp. 88-92.
3. Blahut, R., *Theory and Practice of Error Control Codes*, Addison-Wesley, 1984.
4. Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits", *SIAM Jour. on Computing*, Vol. 13, 1984, pp. 850-864.
5. Boyar, J. "Inferring Sequences Produced by Pseudo-Random Number Generators", *Jour. of ACM*, Vol. 36, No. 1, 1989, pp.129-141.
6. Frieze, A.M., Hastad, J., Kannan, R., Lagarias, J.C., and Shamir, A. "Reconstructing Truncated Integer Variables Satisfying Linear Congruences", *SIAM J. Comput.*, Vol. 17, 1988, pp. 262-280.
7. Golic, J.DJ., and Zivkovic, M.V., "On the Linear Complexity of Nonuniformly Decimated PN-sequences", *IEEE Trans. Inform. Theory*, Vol 34, Sept. 1988, pp. 1077-1079.
8. D. Gollmann and W.G. Chambers, "Clock-controlled shift registers: A review", *IEEE J. Selected Areas Commun.*, vol. 7, pp. 525-533, May 1989,
9. S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.
10. Gunther, C.G., "Alternating Step Generators Controlled by de Bruijn Sequences", in *Lecture Notes in Computer Science 304; Advances in Cryptology: Proc. Eurocrypt '87*, Berlin: Springer-Verlag, 1988, pp. 88-92.
11. Kessler, I., and Krawczyk, H., "Buffer Length and Clock Rate for the Shrinking Generator", preprint.
12. Krawczyk, H., "How to Predict Congruential Generators", *Journal of Algorithms*, Vol. 13, 1992. pp. 527-545.
13. E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the* 23rd *Annual ACM Symposium on Theory of Computing*, pages 455–464, May 1991.

14. Lidl, R., and Niederreiter, H., "Finite Fields", in *Encyclopedia of Mathematics and Its Applications*, Vol 20, Reading, MA: Addison-Wesley, 1983.
15. Yishay Mansour. An $o(n^{\log \log n})$ learning algorihm for DNF under the uniform distribution. In 5th *Annual Workshop on Computational Learning Theory*, pages 53–61, July 1992.
16. Joseph Naor and Moni Naor. Small bias probability spaces: efficient construction and applications. In *Proceedings of the* 22nd *Annual ACM Symposium on Theory of Computing, Baltimore, Maryland*, pages 213–223, May 1990.
17. Rabin, M.O., "Probabilistic Algorithms in Finite Fields", *SIAM J. on Computing*, Vol. 9, 1980, pp. 273-280.
18. Rueppel, R. A., "Stream Ciphers", in Gustavos J. Simmons, editor, *Contemporary Cryptology, The Science of Information*, IEEE Press, 1992, pp. 65-134.
19. Stern, J., "Secret Linear Congruential Generators Are Not Cryptographically Secure", *Proc. of the 28rd IEEE Symp. on Foundations of Computer Science*, 1987.
20. Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. of the 23rd IEEE Symp. on Foundation of Computer Science*, 1982, pp. 80-91.