On the Existence of Secure Keystream Generators

Andrew Klapper*
Department of Computer Science
763H Anderson Hall
University of Kentucky
Lexington KY 40506-0046
Phone: (606) 257-6743; FAX: (606) 323-1971
E-mail: klapper@cs.engr.uky.edu
WWW: http://www.cs.engr.uky.edu/~klapper/andy.html

Abstract

Designers of stream ciphers have generally used ad hoc methods to build systems that are secure against known attacks. There is often a sense that this is the best that can be done, that any system will eventually fall to a practical attack. In this paper we show that there are families of keystream generators that resist all possible attacks of a very general type in which a small number of known bits of a keystream are used to synthesize a generator of the keystream (called a synthesizing algorithm). Such attacks are exemplified by the Berlekamp-Massey attack. We first formalize the notions of a family of finite keystream generators and of a synthesizing algorithm. We then show that for any function $h(n)$ that is in $\mathcal{O}(2^{n/d})$ for every $d > 0$, there is a family $\mathcal{B}$ of periodic sequences such that any efficient synthesizing algorithm outputs a generator of size $h(\log(\mathrm{per}(B)))$ given the required number of bits of a sequence $B \in \mathcal{B}$ of large enough period. This result is tight in the sense that it fails for any faster growing function $h(n)$. We also consider several variations on this scenario.

Index Terms – Binary sequences, keystream generators, security, cryptography, stream ciphers.

*This research funded in part by NSF grant #NCR-9400762. Part of this work was carried out while the author was visiting the Isaac Newton Institute, Cambridge University, Cambridge, UK. Parts of this paper have appeared in the proceedings of Eurocrypt '96, Zaragoza, Spain.

1 Introduction

A stream cipher is often used when it is necessary to encrypt large amounts of data very quickly. It is considered secure if knowledge of a small number of bits of the keystream cannot be used to recover the entire keystream (a "known plaintext" attack).

Historically, the design of stream ciphers has been largely a matter of finding ad hoc methods of foiling existing cryptanalytic attacks. Designers often feel that seeking a truly secure and efficient stream cipher is hopeless, that the best they can do is design a system that resists known attacks. The purpose of this paper is to explore the possibility that there exist families of stream ciphers that resist cryptanalysis by very large classes of attacks. We use asymptotic complexity rather than Shannon theory as the basis for notions of security. A family of stream ciphers is secure against all efficient attacks of a certain general type if all such attacks require asymptotically large numbers of bits of the keystream.

The attacks we are concerned with use a small number of known bits of a keystream to synthesize a fast generator for the keystream. For example, if the keystream can be generated by a linear feedback shift register (or LFSR) of length n, then $2n$ bits of the sequence suffice for the Berlekamp-Massey algorithm to determine the LFSR that generates the keystream [15]. The ingredients that make this attack of concern are as follows.

1. A class of fast devices (LFSRs) that generate all possible eventually periodic sequences.

2. A polynomial time algorithm A and a polynomial $p(n)$ such that if a sequence can be generated by a device of size n, then $p(n)$ bits of the sequence suffice for A to determine the device.

A great deal of energy has gone into the design of (nonlinear) feedback registers that resist the Berlekamp-Massey attack, often by ad hoc methods (see for example [6, 10, 11, 18, 19]). Also, several similar attacks exist based on other types of keystream generators [12, 14].

In this paper we show that there is a family of efficiently generated sequences that resist all such attacks. However, the techniques used to show their existence, while recursive, give no practical method for finding such a family. The essential idea here is that whatever sequence is predicted by a cryptanalytic algorithm given a short prefix, there is an efficiently generated sequence with the same prefix that is distinct from the predicted sequence. We also consider the case where the cryptanalytic algorithm is only expected to approximate the keystream. We use previously known results on the covering radii of Reed-Muller codes to show that there is an efficiently generated sequence whose Hamming distance from the predicted sequence is large.

We want to be clear that we are only describing sequences that are secure against a large class of attacks. We do not claim that these sequences are usable in secure stream ciphers, even if we had a practical construction for generators of the sequences. In fact, the sequences described have m-sequences as large prefixes and hence would leave large prefixes of messages insecure if they were used in stream ciphers.

The existence of provably secure sequence generators was studied previously by Yao [21] and by Blum and Micali [2]. Their models and results were different, however. First, their sequence generators were arbitrary polynomial time computable generators (in the size of the seed). We use a much more restrictive model: sequences are generated by finite state machines whose state change functions can be computed by fast circuits. Second, the attacks considered by Yao and by Blum and Micali required the availability of all previously generated bits to predict the next bit (by a so-called next bit test). The attacks considered here require that only a small number (polynomially many in the size of the resulting generator) of bits be available to generate all remaining bits. Third, the attacks consider by Yao and by Blum and Micali were probabilistic while those considered here are deterministic. Fourth, the existence results they gave were based on unproved complexity theoretic assumptions, such as the intractability of the discrete logarithm problem. Our results hold independent of any such assumptions. Finally, the generators we consider are far more efficient than those considered by Yao and by Blum and Micali.

Maurer also considered the design of private key cryptosystems that resist all attacks [16]. His point of view differed from ours in that the system he designed required a globally accessible source of public randomness. Also, the notion of security was probabilistic – the probability that an enemy could obtain information was shown to be exceedingly small.

In Section 2 we abstract the notions of fast keystream generator and of efficient algorithms for synthesizing such generators given a small number of initial bits. In Section 3 we first show that there is a family of keystream generators that admits no such synthesizing algorithm. We then show that an efficient, secure family $\mathcal{B}$ of sequences exists. This family is secure in the sense that, for every family of keystream generators $\mathcal{F}$ that admits a synthesizing algorithm, the size of the smallest generator in $\mathcal{F}$ that outputs a given sequence B in $\mathcal{B}$ grows at a superpolynomial rate in the size of the smallest efficient generator for B. In Section 4 we show that the bounds in Section 3 are optimal. In Sections 5 and 6 we consider two variants: the case where the cryptanalyst is only required to generate a fraction of the keystream; and the case where the number of bits the cryptanalyst has access to is linear in the size of the smallest generator.

2 Definitions

In this section we describe keystream generators, the basic objects of study of this paper, and notions of security for families of keystream generators. These are finite state machines with output, whose states are given by bit vectors.

Definition 2.1 *A* (keystream) generator *is a 4-tuple* (S, F, g, s_0) *such that*

1. S *is a finite set (the* states*);*
2. $F : S \to S$ *is a function (the* state change function*);*
3. $g : S \to \{0, 1\}$ *is a function (the* output function*); and*
4. s_0 *is an element of* S *(the* initial state*).*

A keystream generator outputs an infinite eventually periodic binary sequence by iterating the state change function and applying the output function to the sequence of states: $g(s_0), g(F(s_0)), g(F(F(s_0))), \cdots$.

In the sequel, because we are concerned with the size and speed of a generator, we assume that S is a set of n bit vectors $\bar{x} = (x_0, \cdots, x_{n-1})$ for some n. In this case we say that the generator has *length* n. We further generally use generators whose output functions are of the form $g(\bar{x}) = x_0$. In this case the generator is completely determined by F and s_0, and we often abuse the notation by identifying the generator with the pair (F, s_0) or simply F if the state is irrelevant. Note that a generator of length n with a more general output function can always be replaced by one of length $n + 1$ with this special form of output function with no increase in the complexity (by any reasonable measure of complexity).

In algorithms dealing with descriptions of such state change functions, we assume that the functions are described by circuits using bounded fan-in, unbounded fan-out gates. We usually use binary (fan-in two) AND gates (denoted $\wedge$), binary XOR gates (denoted $\oplus$), and NOT gates (denoted $\neg$). Such circuits can be encoded as binary strings [1]. The *size* of a generator F is the minimum number of gates in a circuit that computes the function F. The *depth* of a generator F is the depth of the minimum depth circuit that computes F.

One might wonder whether there is a single efficient generator whose output resists all cryptanalytic attacks based on knowledge of the first few bits of the sequence. To see that this cannot be, suppose B is such a sequence, generated by F. The algorithm that outputs F whenever the known plaintext bits coincide with a prefix of B is always successful against B (although it does badly against other sequences). Thus we turn to asymptotic security and families of sequences.

A *family of (keystream) generators,* $\mathcal{F}$, is an infinite collection of keystream generators. We let $\mathcal{F}_n$ denote the set of keystream generators in $\mathcal{F}$ of length n. If B is an infinite eventually periodic binary sequence, then the $\mathcal{F}$-span of B, denoted $\lambda_{\mathcal{F}}(B)$, is the least integer n such that B can be output by a generator in $\mathcal{F}_n$ (or ∞ if there is no such n). We denote the period of B by $\mathrm{per}(B)$.

We are concerned with generators whose state change functions can be computed quickly. Let $\delta(n)$ be the maximum over all F in $\mathcal{F}_n$ of the depth of F. We say $\mathcal{F}$ is

1. *fast* if $\delta(n) \in \mathcal{O}(\log(n))$.

2. *short* if whenever $F \in \mathcal{F}$ generates sequence B, then $\lambda_{\mathcal{F}}(B)$ is $\mathcal{O}(\log(\mathrm{per}(B)))$.

Note that for a fast family the sizes (numbers of gates) of the generators in $\mathcal{F}$ are polynomial in the lengths of the generators, since a depth δ circuit with fan-in at most two has at most 2^{δ} gates. In fact, in all but one case in this paper, the sizes of fast families of generators described are linear in the lengths of the generators. For a fast short family, the depth of the smallest generator of a sequence B is $\mathcal{O}(\log(\log(\mathrm{per}(B)))$, and the size is a polynomial in the log of the period.

Our basic concern is whether, given a small number of bits of a sequence B, we can efficiently synthesize the smallest generator in $\mathcal{F}$ that outputs B.

Definition 2.2 *1. Algorithm T is an $\mathcal{F}$-*synthesizing algorithm *if, when given the input $b_0, \cdots, b_{k-1}$, T outputs the encoding of a generator $(F, s) \in \mathcal{F}$ such that the first k output bits of F with initial state s are $b_0, \cdots, b_{k-1}$.*

2. T is effective *if:*

(a) It runs in polynomial time in k; and

(b) There is a polynomial $p(n)$ such that if $\lambda_{\mathcal{F}}(B) = n < \infty$, on input $b_0, \cdots, b_{k-1}$ with $k \geq p(n)$, T outputs an $(F, s) \in \mathcal{F}$ of length n that generates all of B.

3. *A family $\mathcal{F}$ of generators is* synthetic *if there is an effective $\mathcal{F}$-synthesizing algorithm.*

Fact 2.3 *The family of Linear Feedback Shift Registers and the family of Feedback with Carry Shift Registers [14] are synthetic families.*

We say that a family of sequences is secure with respect to a family of generators if there is either no way to synthesize the best generator in the family for a given sequence, or the length of the best generator grows quickly with the period of the sequence.

Definition 2.4 *Let $\mathcal{B} = B^1, B^2, \cdots$ be a sequence of binary sequences of increasing periods. Let*

$$\Lambda_{\mathcal{F},\mathcal{B}}(n) = \lambda_{\mathcal{F}}(B^n).$$

*Then $\mathcal{B}$ is $\mathcal{F}$-*secure *if either*

1. *$\mathcal{F}$ is not synthetic; or*

2. *For every $k > 0$, we have*

$$\Lambda_{\mathcal{F},\mathcal{B}}(n) \in \Omega(\log(\mathrm{per}(B^n))^k).$$

In either case, for large enough n the shortest generator in $\mathcal{F}$ generating B^n cannot be found effectively.

Observe that we have required a synthesis algorithm to find the smallest generator in the family $\mathcal{F}$ that outputs B. One might more generally consider algorithms that output generators whose length is only close to minimal. However, any family that is secure against all attacks of the restricted type must also be secure against all attacks of this more general type. Suppose T is such an algorithm, synthesizing generators in a family $\mathcal{F}$. Assume that with enough (poynomially many) bits of a sequence, T outputs a generator that generates the correct sequence and that the length of this generator is at most polynomial in the length of the minimal such generator. Let $\mathcal{F}'$ be the set of generators that is actually output by T on various inputs. Then T is also an $\mathcal{F}'$-synthesizing algorithm. The $\mathcal{F}$-span and $\mathcal{F}'$-span of a sequence are polynomially related, so the $\mathcal{F}'$-security of a family of sequences implies it is secure against T in the more general sense.

In describing the growth rates of functions, we say a function $f(n)$ is *subexponential* if for every $d > 0$,

$$f(n) \in \mathcal{O}(2^{n/d}).$$

It is *superpolynomial* if for every $k > 0$,

$$f(n) \in \Omega(n^k).$$

A family is secure against $\mathcal{F}$ if the $\mathcal{F}$-span of its sequences is superpolynomial in the logs of their periods. In fact, we show that for any subexponential function $h(n)$, there are families whose $\mathcal{F}$-spans are greater than $h(\log(\text{period}))$. It is well known that there are subexponential superpolynomial functions.

3 Existence of Secure Keystream generators

In this section we prove the existence of families of keystream generators that resist all synthesis attacks. We prove this in the strong sense that for every attack, all but finitely many sequences in the constructed family resist the attack.

Theorem 3.1 *Let $h(n)$ be any subexponential function. There exists a sequence of binary periodic sequences $\mathcal{B} = B^1, B^2, \cdots$ such that*

a. *$\mathcal{B}$ can be generated by a family $\mathcal{F}$ of fast short generators such that the length of the generator of B^m is at most twice the log of the period of B^m;*

b. *For every synthetic family $\mathcal{F}'$, if m is sufficiently large, then*

$$\Lambda_{\mathcal{F}',\mathcal{B}}(m) \geq h(\log(\text{per}(B^m))).$$

In particular, if we let h be superpolynomial, then $\mathcal{B}$ is $\mathcal{F}'$-secure for every family $\mathcal{F}'$ of keystream generators.

Proof: For each synthesis algorithm T, let $\mathcal{F}_T$ be the family of generators that is output by T. Let $\mathcal{F}^1, \mathcal{F}^2, \cdots$ be an enumeration of the synthetic families $\mathcal{F}^T$ of generators such that each $\mathcal{F}_T$ occurs infinitely often. Such an enumeration is possible because the set of all algorithms is enumerable. (We must be careful here. The set of all synthetic families of generators is not enumerable. Many families have generators that are simply never output by the synthesis algorithm, so many families correspond to the same algorithm. If, however, we restrict attention to those families all of whose generators are actually output by the synthesis algorithm, then there is at most one family per algorithm.) Let the corresponding synthesis algorithms be $T^1, T^2, \cdots$.

We construct $\mathcal{B}$ in stages by a diagonalization argument. At the ith stage we construct B^i to have appropriate properties with respect to $\mathcal{F}^1, \cdots, \mathcal{F}^i$. Let

$$p(n) = n^d$$

be so that for $1 \leq j \leq i$, T^j synthesizes a generator in $\mathcal{F}^j$ that outputs any sequence S given

$$p(\lambda_{\mathcal{F}^j}(S))$$

bits of S.

Let $t = \lceil \log(i+1) \rceil$. We construct B^i so that it has period

$$\text{per}(B^i) = 2^{k+1} + t - 1,$$

and can be generated by a generator of length $k + t + 3$ and depth

$$\max(\lceil \log(k+1) \rceil, \lceil \log(t) \rceil) + 3$$

for some k.

Let r be

1. larger than the period of any previous B^j;
2. larger than t; and
3. large enough that, for every $k \geq r$, we have

$$2^{k/d} > h(\log(2^k + t)).$$

Choose n and k so that

$$n^d > 2^r \text{ and } 2^k \leq n^d < 2^{k+1}.$$

We construct $i + 1$ generators whose outputs are identical to one period of an m-sequence for $2^{k+1} - 1$ bits. The jth generator then outputs the binary expansion of the integer j. This can be done by adding circuitry to the generator of the m-sequence that checks for an all zero state, and, when found, switches to a separate pure cycling register of length $\lceil \log(i+1) \rceil$ that has been loaded with the binary expansion of j. The total length of this generator is

$$\begin{aligned} \log(i+1) + k + 3 &\leq 2k + 2 \\ &\leq 2\log(\text{per}(B)) \\ &= \mathcal{O}(\log(\text{per}(B))). \end{aligned}$$

where B is the output sequence. (Note that two extra bits are needed for the output and for a flag to tell which part of the generator is currently active). The depth of the circuit is

$$\mathcal{O}(\log(k) + \log\log(i)) = \mathcal{O}(\log(k)).$$

Hence this is a fast short generator.

There must be at least one of these sequences, which we denote B^i, that satisfies

$$\begin{aligned} \lambda_{\mathcal{F}^j}(B^i) &> n \\ &\geq 2^{k/d} \\ &> h(\log(2^k + t)) \\ &= h(\log(\mathrm{per}(B^i))) \end{aligned}$$

for $1 \leq j \leq i$. This concludes stage i. □

For sequences B generated by a short family $\mathcal{F}$, $\lambda_{\mathcal{F}}(B)$ is $\mathcal{O}(\log(\mathrm{per}(B)))$.

Corollary 3.2 *Let $h(n)$ be any subexponential function. There exists a sequence of binary periodic sequences $\mathcal{B} = B^1, B^2, \cdots$ such that for every synthetic family $\mathcal{F}'$,*

$$\lambda_{\mathcal{F}',\mathcal{B}}(m) \in \Omega(h(\Lambda_{\mathcal{F},\mathcal{B}}(m))).$$

Corollary 3.3 *There exist (uncountably many) nonsynthetic families of fast short generators.*

The constructions in Theorem 3.1 can be made recursive. That is, there is an effective procedure which, given i, outputs a list of the generators in $\mathcal{F}_i$ in the first case or B^i (or a generator of B^i) in the second case. Such a procedure, however, is likely to be impractically slow.

4 Exponential Bounds Are Impossible

In this section we show that Theorem 3.1 is sharp in the sense that the function h cannot be replaced by an exponential function. We first need a lemma.

Lemma 4.1 *There is a polynomial time algorithm which, given $2p$ or more consecutive bits of a periodic sequence, outputs the period of the sequence. (If the algorithm is given fewer than $2p$ bits, it may output anything.)*

Proof: Suppose we are given $2k$ bits of a sequence. In time $\mathcal{O}(k^2)$, we can find the least p such that

$$b_i = b_{i+p} \text{ for all } 0 \leq i < 2k - p.$$

We claim that if q is the true period and $k \geq q$, then $p = q$.

We have $p \leq q$ since q satisfies the above condition. Let j be any index and let $j = xq + y$ with $0 \leq y < q$. Then $y < 2k - p$ so

$$\begin{aligned} b_j &= b_y \\ &= b_{y+p} \\ &= b_{j+p}. \end{aligned}$$

Thus q divides p and so $q = p$. □

Theorem 4.2 *Let $h(n) = 2^{n/d}$ be an exponential function, and let $\mathcal{B} = B^1, B^2, \cdots$ be any sequence of periodic binary sequences. There exists a fast synthetic family of generators $\mathcal{F}$ such that for every i,*

$$\lambda_{\mathcal{F}}(B^i) \leq h(\log(\mathrm{per}(B^i))).$$

Proof: We construct the family $\mathcal{F}$ by describing a generator synthesis algorithm T. $\mathcal{F}$ is then the set of generators output by T.

A k bit generator generated by T has the following form. The last $k-1$ bits operate independently of the first bit. The first bit is computed as a function of the last $k-1$ bits. Thus the generators are, in effect, nonlinear feedback registers with nonlinear feedforward functions.

Algorithm T will produce a generator of length

$$\left\lfloor p^{1/d} \right\rfloor$$

when acting on a sequence of period p. Thus

$$\begin{aligned} \lambda_{\mathcal{F}}(B^i) &= \left\lfloor \mathrm{per}(B^i)^{1/d} \right\rfloor \\ &\leq h(\log(\mathrm{per}(B^i))). \end{aligned}$$

Since the number of bits the algorithm can have access to is polynomial in $\lambda_{\mathcal{F}}(B^i)$, we can assume T knows two complete periods of B^i. Using the algorithm promised by Lemma 4.1, we first compute the period p.

The next step is to construct a fast keystream generator whose state sequence has period p. This can be done, for example, by constructing a maximal period LFSR of length k, with

$$2^{k-1} \leq p < 2^k.$$

Thus the period of this LFSR is $2^k - 1$. Such a LFSR can be found by an exhaustive search for a primitive polynomial of degree k. There are $2^k \leq 2p$ polynomials of degree k, and each can be checked for primitivity in time quasi-linear in p. Thus such a LFSR can be found in polynomial time. It can then be modified to switch back to its initial state after p states by using a k-bit AND to check for the pth state. We call the resulting generator G.

The construction is completed by finding a binary function on k bits that has the bits of the sequence as values on the p states of G. This can be written as an XOR of p terms, each an AND of k bits. Such an expression can be implemented as a circuit of depth

$$\lceil \log(p) \rceil + \lceil \log(k) \rceil .$$

Finally, the resulting generator is extended to length

$$\left\lfloor p^{1/d} \right\rfloor$$

by padding it with

$$\left\lfloor p^{1/d} \right\rfloor - k$$

dummy bits on the left. $\Box$

5 Partial Attacks

For many purposes the attacks considered in the preceding sections are too weak. A system is also vulnerable if an adversary can find a substantial number of bits of the keystream. This is especially true if there is enough context in the message to recover the remaining bits. If $\mathcal{F}$ is a family of generators, B is a sequence of (eventual) period m, and $0 < r \leq m$, then $\lambda_{\mathcal{F},r}(B)$ is the size of the smallest generator F in $\mathcal{F}$ whose output agrees with B on at least r bits of each period[1] of B.

[1]Some care must be taken here. The sequence B and the output sequence B' of F may have different periods, and in fact may not be strictly periodic, only eventually periodic. A reasonable interpretation is that $r/\mathrm{per}(B)$ is less than or equal to the limit as n goes to ∞ of $|\{i, 1 \leq i \leq n : b_i = b'_i\}|/n$.

Definition 5.1 *Let T be an $\mathcal{F}$-synthesizing algorithm and $0 < r(m) \leq m$. We say that T is $r(m)$*-effective for *$\mathcal{F}$ if*

1. *It runs in polynomial time; and*

2. *There is a polynomial $p(n)$ such that if B is a sequence with (eventual) period m and $n = \lambda_{\mathcal{F},r(m)}(B)$, then on input $b_0, \cdots, b_{k-1}$ with $k \geq p(n)$, T outputs an $F \in \mathcal{F}$ of length n. If the sequence generated by F is B', then for any k,*

$$|\{i, k \leq i \leq k+m-1 : b_i = b'_i\}| \geq r(m).$$

$\mathcal{F}$ is $r(m)$-synthetic *if there is an $r(m)$-effective algorithm for $\mathcal{F}$.*

Theorem 5.2 *Let $h(n)$ be subexponential and let*

$$r(m) = \frac{m}{2} + \mathcal{O}(m^{1/2}).$$

There exists a family $\mathcal{F}$ of fast short generators such that for every $r(m)$-synthetic family $\mathcal{F}'$, there are infinitely many generators F in $\mathcal{F}$ with output sequence B of eventual period m satisfying

$$\lambda_{\mathcal{F}',r(m)}(B) \geq h(\log(\mathrm{per}(B))).$$

Proof: The goal is to find efficiently generated sequences with a large Hamming distance from whatever sequence is produced by the algorithm we are trying to diagonalize against. We do so by using Reed-Muller codes and well known results from coding theory concerning the covering radii of these codes.

Let $f(\bar{x})$ be a polynomial of degree at most d in n variables. Then the Reed-Muller codeword associated with f is the length 2^n Boolean vector whose components are the 2^n values of f. The Reed-Muller code of length n and degree d, $RM(n,d)$, consists of all these codewords.

The output from the generators we construct consist of an m-sequence of period $2^k - 1$ followed by a $RM(n,d)$ codeword c for some k, n, and d. The parameter d is independent of the stage of the diagonalization (but depends on $r(m)$). The first step is to see that it is possible to construct a fast short generator that outputs such a sequence. The generator consists of two parts: an LFSR that generates the m-sequence, and a generator that outputs c. The overall generator can be made to output the m-sequence, switch to the generator of c. This is accomplished by detecting the last state of the LFSR with an AND of k bits. This takes depth $\log(k)$. When c has been output, the generator then switches back to the LFSR similarly.

The generator that outputs c can be constructed by starting with a generator of length n and period 2^n – modify a LFSR of period $2^n - 1$ so it outputs an extra 0 when it reaches the all zero state. This requires at most depth $\log(n)$ and one extra bit of state. Now c can be generated by computing the value of the function f on the state. Since the degree of f is bounded by d, it is a sum of at most a polynomial in n number of monomials, each of degree at most d, and so can be computed by a circuit of depth

$$\mathcal{O}(\log(\text{polynomial}(n))) = \mathcal{O}(\log(n)).$$

Thus we have a sequence of period $2^n + 2^k - 1$ generated by a generator of length $n + k + 3$ (two extra bits are usd for output and for switching between the two modes of operation) and depth

$$\mathcal{O}(\log(n) + \log(k)) = \mathcal{O}(\log(n + k + 3)).$$

Furthermore,

$$\begin{aligned} n + k + 3 &\leq 3\max(n, k) \\ &\leq 3\log(2^n + 2^k - 1), \end{aligned}$$

so this is a fast short generator.

Recall that the *covering radius* of a code C is the smallest integer ρ such that every vector in the ambient space is within ρ of at least one codeword in C. In the past 15 years or so, a large body of literature has been built up concerning the covering radii of codes (see the excellent surveys [3, 5]). It is known that the covering radius of the Reed-Muller code $RM(n, d)$ for fixed k is at most

$$\frac{2^n - t2^{n/2}}{2},$$

where t depends only on d [4]. The constant t can be made arbitrarily large by the choice of d. In our case, if

$$r(m) < \frac{m}{2} + t'm^{1/2},$$

then we choose d so that $t > 4t' + 1$.

Of course we really want sequences that are far from given sequences, but in the Hamming metric, if c is close to b, then the complement c' of c is far from b: $dist(c', b) = 2^n - dist(c, b)$ if the length of the code is 2^n. The Reed-Muller code is close under complementation (add 1 to f), so we see that there is a Reed-Muller codeword whose distance from any given sequence of length 2^n is at least

$$\frac{2^n + t2^{n/2}}{2}.$$

As previously, for each $r(m)$-synthesis algorithm T, let $\mathcal{F}_T$ be the family of generators that is output by T. Let $\mathcal{F}^1, \mathcal{F}^2, \cdots$ be an enumeration of the synthetic families of generators such that each $\mathcal{F}_T$ occurs infinitely often. Let the corresponding synthesis algorithms be $T^1, T^2, \cdots$ and assume T^i is successful when given

$$p_i(\lambda_{\mathcal{F}^i, r(m)}(B))$$

bits of any sequence B. At the ith stage of the diagonalization we want to include in $\mathcal{F}$ a fast generator F, as described above, with output B so that $\lambda_{\mathcal{F}^i, r(m)}(B)$ is large.

Let $p_i(x) < x^\ell$. Let $k' = 3k$ be large enough that

1. $$h(k') < 2^{k'/(4\ell)} < (2^k - 1)^{1/\ell}$$

   and

2. $$m = 2^k + 2^{k^2} - 1$$

   is larger than the period of any sequence generated by a generator already incuded in $\mathcal{F}$.

Let $n = k^2$. Thus the generator constructed above generates an m-sequence of period

$$\begin{aligned} 2^k - 1 &> h(3k)^\ell \\ &> p_i(h(3k)) \\ &> p_i(h(\log(2^{2k} + 2^k - 1))) \\ &> p_i(h(\log(m))). \end{aligned}$$

Recall that if

$$r(m) \leq \frac{m}{2} + t' m^{1/2},$$

and $t > 4t' + 1$, we can pick d so that the covering radius of $RM(n, d)$ is at most

$$\frac{2^n + t2^{n/2}}{2}.$$

We then choose a Reed-Muller codeword $c \in RM(n, d)$ so that whatever sequence T^i outputs given the $2^k - 1$ bits of the initial m-sequence, the last 2^n bits disagree with the codeword on at least

$$\begin{aligned}
\frac{2^n + t2^{n/2}}{2} &> \frac{2^n + 2^k - 1}{2} + (t-1)2^{k-1} \\
&> \frac{m}{2} + (t-1)2^{k-1} \\
&> \frac{m}{2} + t'(2^n + 2^k - 1)^{1/2} \\
&= \frac{m}{2} + t' m^{1/2}
\end{aligned}$$

bits. As we have seen, such a codeword always exists. Let B be the sequence one of whose periods is the m-sequence followed by the codeword. We include the above generator of B in $\mathcal{F}$. Then

$$\begin{aligned}
\lambda_{\mathcal{F}', r(m)}(B) &> h\left(\log\left(\frac{2^n + 2^k - 1}{2}\right)\right) \\
&> h(\log(\mathrm{per}(B))).
\end{aligned}$$

$\square$

This result will be improved if easily generated codes with small covering radii can be constructed. Coding theorists have studied covering radii for some years, but good asymptotic bounds are difficult to obtain and they seem to have not considered the question of generation of the codewords by short fast registers (although there has been work on finding good codewords in polynomial time and space [17]). It would also be desirable to find a sequence of sequences B^i so that B^i resists the first i $r(m)$-synthetic attacks. Using our techniques, such a construction would depend on finding easily generated codes with small *multicovering radii*, i.e., the smallest ℓ such that every set of i sequence is within distance ℓ of at least one codeword [13]. Improved results along these lines will be the subject of a future paper.

6 Linear Synthesis Attacks

In this section we discuss the effect on our results of restricting the power of the synthesis algorithms.

As defined, synthesis algorithms depend on polynomial bounds. A synthesis algorithm for a family $\mathcal{F}$ of generators must work correctly if the number of bits available is at least a fixed polynomial in the $\mathcal{F}$-span, and the running time must be polynomially bounded in the number of bits available. If the degree of the polynomial is large, however, it is questionable whether such an attack should be considered strong enough to be of practical concern. By contrast, the Berlekamp-Massey and the 2-adic rational approximation algorithms work correctly if at least a linear number of bits are available. The former algorithm has quadratic running time, while the latter has quasi-quadratic running time. An algorithm is said to be a *linear synthesis algorithm* for a family $\mathcal{F}$ if it requires only a linear number of bits in $\lambda_{\mathcal{F}}(B)$ to synthesize a generator in $\mathcal{F}$ that outputs B. Then $\mathcal{F}$ is said to be *linearly synthetic*. Theorem 3.1 can be improved if we restrict our attention to linear synthesis.

Theorem 6.1 *Let $h(n) \in o(2^n)$. There exists a sequence of binary periodic sequences $\mathcal{B} = B^1, B^2, \cdots$ such that*

a. $\mathcal{B}$ can be generated by a fast family $\mathcal{F}$ of generators such that the length of the generator of B^i is at most twice the log of the period of B^i;

b. Let $\mathcal{F}'$ be a linearly synthetic family. For every sufficiently large i we have

$$\lambda_{\mathcal{F}'}(B^i) \geq h(\log(\mathrm{per}(B^i))).$$

Proof: The proof is similar to that of Theorem 3.1. The difference is that now the polynomial bound $p(n)$ on the number of bits needed for the ith linear synthesis algorithm to be successful is a linear bound,

$$p(n) = an$$

for some a. We then choose r to be large enough that for every $k \geq r$,

$$\frac{2^k}{a} \geq h(k+3).$$

This is possible due to the asymptotic bound on h. The remainder of the construction is unchanged. $\square$

This result can then be shown to be tight.

Theorem 6.2 *Let $h(n) = \Omega(2^n)$. Let $\mathcal{B} = B^1, B^2, \cdots$ be a sequence of periodic binary sequences. There exists a fast linearly synthetic family of generators $\mathcal{F}$ such that for every i,*

$$\lambda_{\mathcal{F}}(B^i) \leq h(\log(\mathrm{per}(B^i))).$$

Proof: The proof is similar to that of Theorem 4.2. Let

$$h(n) > c2^n$$

for n sufficiently large. The synthesis algorithm produces a generator of length $\lfloor cp \rfloor$ when acting on a sequence of period p. Thus

$$\begin{aligned} \lambda_{\mathcal{F}}(B^i) &= \left\lfloor c \cdot \mathrm{per}(B^i) \right\rfloor \\ &\leq h(\log(\mathrm{per}(B^i))) \end{aligned}$$

By the definition of linear synthesis, it can have access to two entire periods of the sequence. The remainder of the construction is identical to that in the proof of Theorem 4.2, except that we must pad the generator to length $\lfloor cp \rfloor$. □

7 Conclusions and Open Questions

We have described a general model for attacks on stream ciphers of a very general type. Using this model, we have proved the existence of families of keystream generators that resist all such attacks. The proof, however, does not give a practical construction. We hope to inspire researchers to search for such highly secure keystream generators with more natural descriptions. The basic open question we leave is whether practical constructions can be found for generator and sequence families that satisfy the conclusions of Theorem 3.1.

One interpretation of the results here is that the common approach to stream cipher design and analysis – describe a class of efficiently generated sequences with large linear span, or even resistance to several cryptanalytic attacks – is largely useless. Some researchers have improved this approach with refined security measures such as the linear complexity profile [18] and sphere complexity [8]. In a sense, security with respect to these measures means security for all "pieces" of a sequence, but only against the Berlekamp-Massey attack. What is really needed is a combination of these approaches and the approach in the current paper. To wit, a family of practically generated sequences for which all pieces (in some sense) are asymptotically secure against all possible attacks.

We have also considered only one class of attack on stream ciphers. Other attacks are possible, for example probabilistic attacks such as correlation attacks [20], differential cryptanalysis [7], and linear cryptanalysis [9]. It is desirable to formalize such probabilistic attacks and (hopefully) prove the existence of classes of keystream generators that universally resist them.

We have concentrated on the depth of circuits as a measure of feasibility. This corresponds to time of evaluation. Size (number of gates) is also a concern as it impacts area and hence cost. In all our theorems except Theorem 4.2 the sizes of the resulting fast generators are linear in the lengths of the generators. In Theorem 4.2, the sizes of the fast generators are polynomial in their lengths. We leave open the question as to whether one can achieve smaller sizes than these.

References

[1] J. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I*, Springer-Verlag, Berlin, 1988.

[2] M. Blum and S. Micali, How to generate cryptographically strong sequences of pseudorandom bits, *SIAM Journal on Computing,* vol. 13 (1984), pp. 850-864.

[3] G.D. Cohen, M.G. Karpovsky, H.F. Mattson, Jr., and J.R. Schatz, Covering radius – survey and recent results, *IEEE Trans. Info. Theory,* vol. IT-31 (1985), pp. 328-343.

[4] G. Cohen and S. Litsyn, On the covering radius of Reed-Muller codes, *Discrete Mathematics,* vol. 106-107 (1992), pp. 147-155.

[5] G.D. Cohen, S.N. Litsyn, A.C. Lobstein, and H.F. Mattson, Jr., *Covering Radius 1985-1994,* Technical Report 94 D 025, Dept. Informatique,Ecole Nationale Supérieure des Télécommunications, 1994.

[6] A.H. Chan and R.A. Games, On the linear span of binary sequences from finite geometries, q odd, *IEEE Trans. Info. Theory*, vol. 36 (1990), pp. 548-552.

[7] C. Ding, The differential cryptanalysis and design of natural stream ciphers, *Fast Software Encryption: Proceedings of 1993 Cambridge Security Workshop,* R. Anderson, ed., *Lecture Notes in Computer Science,* vol. 809, Springer-Verlag, Berlin, 1994, pp. 101-120.

[8] C. Ding, G. Xiao, and W. Shan, *The Stability Theory of Stream Ciphers*, *Lecture Notes in Computer Science*, vol. 561. Springer-Verlag, Berlin, 1991.

[9] J. Golić, Linear cryptanalyis of stream ciphers, *Fast Software Encryption: Proceedings of 1994 Leuven Security Workshop,* B. Preneel, ed., *Lecture Notes in Computer Science,* vol. 1008, Springer-Verlag, Berlin, 1995, pp. 154-169.

[10] E.J. Groth, Generation of binary sequences with controllable complexity, *IEEE Trans. Info. Theory,* vol. IT-17 (1971), pp. 288-296.

[11] E.L. Key, An Analysis of the structure and complexity of nonlinear binary sequence generators, *IEEE Trans. Info. Theory,* vol. IT-22 (1976), pp. 732-736.

[12] A. Klapper, The Vulnerability of geometric sequences based on fields of odd characteristic, *Journal of Cryptology,* vol. 7 (1994), pp. 33-51.

[13] A. Klapper, The multicovering radii of codes, *IEEE Trans. Info. Theory*, vol. 43 (1997), pp. 1372-1377.

[14] A. Klapper and M. Goresky, Feedback shift registers, 2-adic span, and combiners with memory, *J. Cryptology*, vol. 10 (1997), pp. 111-147.

[15] J.L. Massey, Shift register sequences and BCH decoding, *IEEE Trans. Info. Theory,* vol. IT-15 (1969), pp. 122-127.

[16] U. M. Maurer, A provably-secure strongly-randomized cipher, *Advances in Cryptology – CRYPTO '90,* S. Vanstone, ed., *Lecture Notes in Computer Science,* vol. 473, Springer-Verlag, Berlin, 1991, pp. 361-73.

[17] J. Pach and J. Spencer, Explicit codes with low covering radius, *IEEE Trans. Info. Theory,* vol. IT-34 (1988), pp. 1281-1285.

[18] R. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, New York, 1986.

[19] R.A. Rueppel and O.J. Staffelbach, Products of linear recurring sequences with maximum complexity, *IEEE Trans. Info. Theory,* vol. IT-33 (1987), pp. 124-129.

[20] T. Siegenthaler, Decrypting a class of stream ciphers using cipertext only, *IEEE Trans. Comp.,* vol. 34 (1985), pp. 81-85.

[21] A. Yao, Theory and applications of trapdoor functions, *Proceedings, 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 80-91.