Cryptanalysis Based on 2-Adic Rational Approximation

Andrew Klapper

Dept. of Computer Science
University of Kentucky
Lexington, KY 40506-0046 USA
klapper@cs.engr.uky.edu

Mark Goresky

Dept. of Mathematics
Northeastern University
Boston, MA 03115 USA
goresky@nuhub.neu.edu

Abstract. This paper presents a new algorithm for cryptanalytically attacking stream ciphers. There is an associated measure of security, the *2-adic span*. In order for a stream cipher to be secure, its 2-adic span must be large. This attack exposes a weakness of Rueppel and Massey's summation combiner. The algorithm, based on De Weger and Mahler's rational approximation theory for 2-adic numbers, synthesizes a shortest *feedback with carry shift register* that outputs a particular key stream, given a small number of bits of the key stream. It is adaptive in that it does not neeed to know the number of available bits beforehand.

Index Terms – **Binary sequences, feedback with carry shift registers, cryptanalysis, rational approximation, 2-adic numbers.**

1 Introduction

In this paper a new general purpose attack on stream ciphers is presented. This attack can be used successfully, for example, against Rueppel and Massey's summation combiner [16, 19]. All future stream cipher designers will need to consider resistance to this attack.

The development of cryptosystems tends to alternate between the design of new systems that resist known attacks, and the design of new attacks against systems. Often such attacks are highly specialized and only work against particular systems. Occasionally a very general attack is found that can potentially be used against a large class of cryptosystems. Examples include differential cryptanalysis, which can be used against many round based block ciphers such as DES, [1], and the Berlekamp-Massey algorithm [15], which can be used against stream ciphers. Sometimes we can numerically measure the extent to which cryptosystems resist a particular attack. This is the case with the Berlekamp-Massey algorithm: the *linear span* of a binary sequence S is the size of the smallest linear feedback shift register (or LFSR) that generates S. The higher the linear span of a sequence, the greater the resistance to the Berlekamp-Massey algorithm when S is used as the key in a stream cipher. Thus when a cryptographer designs a new stream cipher, it is imperative that she show that the key stream has large linear span.

D. Coppersmith (Ed.): Advances in Cryptology - CRYPTO '95, LNCS 963, pp. 262-273, 1995.

As with the Berlekamp-Massey algorithm, the method of cryptanalysis described in this paper has an associated measure of security, the *2-adic span*. There is a precise way in which the Berlekamp-Massey algorithm can be thought of as a rational approximation algorithm in the ring of power series with integer coefficients. Similarly, the new attack can be thought of as a rational approximation algorithm in the ring of 2-adic integers. The Berlekamp-Massey algorithm is closely related to continued fractions [3, 4, 13, 17, 22]. Over the 2-adic numbers, however, continued fractions fail to converge in general. The algorithm presented here is based on Mahler and De Weger's lattice theoretic substitute for 2-adic continued fractions [12, 21]. Unlike Mahler and De Weger's approach, our algorithm is adaptive. The number of known bits of key stream does not need to be determined before the algorithm can be executed. Another nonadaptive 2-adic rational approximation algorithm was discovered by Gregory and Krishnamurthy [6].

Recall that a stream cipher is a private key system in which the message, a binary sequence, is encrypted by adding it bit by bit modulo two to a key stream, which is another binary sequence. The key stream is known to the legitimate receiver of the message who can thus recover the message by adding the key to the ciphertext. Individual stream ciphers are defined by the manner in which the key is generated. A cryptosystem (or the associated key) is secure if the message cannot be determined by an eavesdropper who does not know the key. A cryptosystem is generally only considered secure if it is secure against a known plaintext attack. In the case of stream ciphers, this means that an eavesdropper should be unable to determine the key even if part of the key is known.

For an attack to be practical, it must also be possible for the eavesdropper to generate the key at a rate commensurate with the rate at which it is generated by the sender of the message. The power of the Berlekamp-Massey algorithm is that it produces a description of a fast device – a short LFSR – that generates the key, if such a device exists. Recently, a new class of feedback registers, *feedback with carry shift registers* or FCSRs was developed [7, 9]. These registers are designed to quickly output the 2-adic representation of a rational number. Moreover, they have associated with them a number of algebraic structures that are analogs of the algebraic structures used to analyze linear feedback shift registers.

When FCSRs were originally described by Klapper and Goresky, no adaptive, provably convergent algorithm was known for synthesizing FCSRs from short prefixes of eventually periodic sequences. The algorithm described in this paper fills this gap. It proceeds in two stages. The eavesdropper is assumed to have access to the first few bits of the key stream $a_0, a_1, \cdots$. A rational approximation algorithm (described in Section 3) is used to find the best rational approximation p/q to the 2-adic number $\alpha = \sum_{i=0}^{\infty} a_i 2^i$. This approximation will be best in the sense that, for all approximations that are accurate modulo a particular power of 2, the maximum of $|p|$ and $|q|$ is as small as possible. We show that if in fact α is rational, $\alpha = p'/q'$, then our rational approximation algorithm finds the reduced rational representation of α if $\lceil 2\log(\max(|p'|, |q'|))\rceil + 2$ bits are known (see Theorem 6). This is approximately twice the size of the smallest

FCSR that generates the key (see Corollary 8). Thus the rate of convergence of our algorithm is analogous to that of the Berlekamp-Massey algorithm. The complexity of our algorithm is $\mathcal{O}(T^2 \log T \log\log T)$, where T is the number of known bits. This exceeds the complexity of the Berlekamp-Massey algorithm only by the log factors. Once the rational representation of α is found, there is a fast algorithm for finding an initial loading of a FCSR that outputs α. This is described in Section 4.

It follows that any key stream that can be generated by a small FCSR (or euivalently, whose associated 2-adic number is a rational number p/q, with $|p|$ and $|q|$ small) leads to an insecure stream cipher. This fact gives rise to a security measure.

Definition 1. The *2-adic span* of an eventually periodic sequence $\mathbf{a} = a_0, a_1, \cdots$ is the number of bits of storage used by the smallest FCSR that generates $\mathbf{a}$. It is denoted by $\lambda_2(\mathbf{a})$.

The idea of cryptanalysis using FCSRs is that if a key stream can be generated by a short FCSR, and we can determine this FCSR efficiently from a small subsequence of the key stream, then we can construct an efficient generator for the key stream. It may not be the same device that was originally used to generate the key stream, but this is immaterial to the cryptanalyst. Thus in order for a stream cipher to be secure, its key stream must have large 2-adic span. In the conclusions we describe the effect on the security of a well known key stream generator, the summation combiner [16, 19].

2 Review of Feedback with Carry Shift Registers

In this section we describe FCSRs and some of their basic properties. Details of the construction and results in this section may be found in [7, 9].

FCSRs are based on division in the ring of 2-adic integers. Recall that a 2-adic integer is a series $\sum_{i=0}^{\infty} a_i 2^i$, $a_i \in \{0,1\}$, where we have replaced the indeterminate x by the integer 2. Addition and multiplication are defined term by term as for ordinary integers. The set of 2-adic integers forms a complete valued ring $\mathbf{Z}_2$. The ordinary rational numbers intersect $\mathbf{Z}_2$ in the set of rationals with odd denominator or, equivalently, in the set of 2-adic integers whose sequence of coefficients is eventually periodic. See [11] for background on 2-adic numbers.

Definition 2. ([7, 9]) The FCSR with connection integer $q = -1 + \sum_{i=1}^{r} q_i 2^i$, $q_i \in \{0,1\}$, is a device with r bits of storage plus an auxiliary memory containing an integer. If the auxiliary memory is m, and the contents of the register consists of the r bits $(a_{r-1}, a_{r-2}, \ldots, a_1, a_0)$, then the operation of the shift register is defined as follows:

A1. Form the integer sum $\sigma = \sum_{k=1}^{r} q_k a_{r-k} + m$.
A2. Shift the contents one step to the right, outputting the rightmost bit a_0.
A3. Place $a_r = \sigma \bmod 2$ into the leftmost cell of the shift register.
A4. Replace the memory integer with $m = (\sigma - a_r)/2 = \lfloor \sigma/2 \rfloor$.

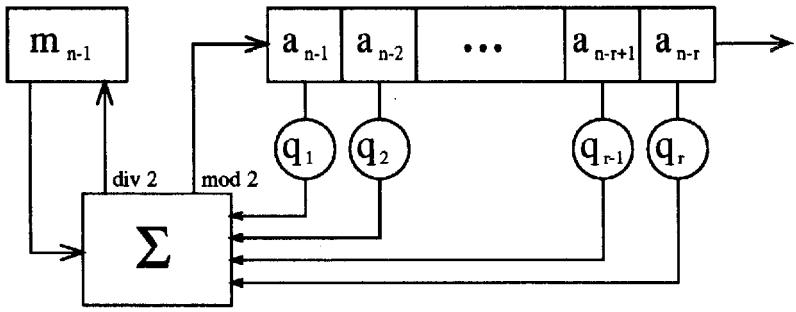

Fig. 1. Feedback with Carry Shift Register

The integer q is referred to as the connection integer because it is the analog of the connection polynomial associated with a LFSR. Notice that $q_0 = -1$ does not correspond to a feedback tap. A FCSR is depicted in Figure 1.

We have the following facts about FCSRs

Theorem 3.

1. *The output from a FCSR is eventually periodic. If q is the connection integer, then the 2-adic number associated with the output sequence is a rational number of the form p/q.*
2. *Conversely, if p is any integer, then the 2-adic expansion of the rational number p/q can be realized as the output of a FCSR with connection integer q. At times we identify p/q with its associated 2-adic number and its associated sequence of coefficients and simply refer to p/q as the output of the register.*
3. *The rational number p/q is strictly periodic if and only if $-q < p \leq 0$.*
4. *Consider a FCSR with connection integer $q = \sum_{i=0}^{r} q_i 2^i$ and initial memory m. Suppose $0 \leq m \leq |\{i : q_i = 1\}|$. Then this condition holds throughout the operation of the register. Also, this condition holds for strictly periodic p/q. Thus for these sequences the space required for the auxiliary memory is fewer than $\log r$ bits.*
5. *Adding b to the memory of a FCSR with connection integer q changes the 2-adic value of the output by $-b2^r/q$.*

3 The Rational Approximation Algorithm

In this section we present an algorithm for synthesizing FCSRs. The goal is to find the rational approximation p/q to a given 2-adic number α with $\Phi(p,q) \stackrel{def}{=} \max(|p|,|q|)$ minimal. For eventually periodic sequences, this quantity is related to the 2-adic span by Theorem 7. If α is eventually periodic, the algorithm finds the precise rational representation p/q of α if $\lceil 2\log(\Phi(p,q))\rceil + 2$ consecutive bits of $\mathbf{a}$ are known. It follows that the number of bits of $\mathbf{a}$ needed to find such a register is at most $2\lambda_2(\mathbf{a}) + 2\lceil \log(\lambda_2(\mathbf{a}))\rceil + 4$.

Assume we have consecutive terms $a_0, a_1, \ldots$ of a binary sequence $\mathbf{a}$ which is the 2-adic expansion of a number α. We wish to determine integers p and q such that $\alpha = p/q$. The algorithm proceeds by computing successive approximations g_1/g_2 to α such that at stage k

$$\alpha \cdot g_2 - g_1 \equiv 0 \ (\bmod\ 2^k). \tag{1}$$

The algorithm is given in Figure 2. It is based on Mahler and De Weger's lattice theoretic approach to p-adic rational approximation [12, 21], but has the advantage that the number of known bits of α need not be predetermined. Each new bit that is found with certainty leads to a better approximation. The symbols $f = (f_1, f_2)$ and $g = (g_1, g_2)$ denote pairs of integers. Note that the minimization steps can be performed by a pair of integer divisions. This is explained in more detail in the last paragraph of this section.

To understand the algorithm some notation is useful. For any $h = (h_1, h_2) \in \mathbf{Z} \times \mathbf{Z}$, let $h(\alpha) = \alpha \cdot h_2 - h_1$. Let $L_k = \{h : h(\alpha) \equiv 0 \bmod 2^k\} \subseteq \mathbf{Z} \times \mathbf{Z}$. Thus the goal of the algorithm is to find the element of L_k with h_2 odd and the minimum value of Φ over all elements of L_k with h_2 odd. The set L_k is a **Z**-*lattice* – a finitely generated module over **Z**. De Weger characterized bases for L_k as follows.

Lemma 4. *The pair f, g is a basis for L_k if and only if $|f_1 g_2 - f_2 g_1| = 2^k$.*

Correctness of the algorithm is proved by the following lemma.

Lemma 5. *At each stage we have*

1. *f and g are in L_k;*
2. *$\langle f, g\rangle$ is a basis for L_k;*
3. *$f \in 2(\mathbf{Z} \times \mathbf{Z}) - L_{k+1}$;*
4. *g minimizes Φ over all elements of L_k with g_2 odd.*
5. *f minimizes Φ over all elements of L_k with f_1 and f_2 even.*

Proof: We prove this by induction. It is straightforward to check that the conditions hold initially.

Let us suppose that the conditions hold at stage k. If $g(\alpha) \equiv 0 \bmod 2^{k+1}$, or equivalently, $g \in L_{k+1}$, then it is again straightforward to check the conditions. Therefore, assume $g \notin L_{k+1}$. We treat the case when $\Phi(g) < \Phi(f)$. The other case is similar.

Let f' and g' be the new values after updating.

Rational_Approximation()

begin
Input $a_0, a_1, \cdots, a_{k-1} = 0$, where $a_0 = \cdots = a_{k-2}$, and $a_{k-1} = 1$
$\alpha = a_0 + a_1 \cdot 2 + \cdots + a_{k-1} \cdot 2^{k-1}$
$f = (0, 2)$
$g = (2^{k-1}, 1)$
while there are more known bits **do**
    Input a_k
    $\alpha = \alpha + a_k 2^k$
    if $g(\alpha) \equiv 0 \bmod 2^{k+1}$ **then**
        $f = 2f$
    else if $\Phi(g) < \Phi(f)$ **then**
        Let d be odd and minimize $\Phi(f + dg)$
        $\langle g, f \rangle = \langle f + dg, 2g \rangle$
    else
        Let d be odd and minimize $\Phi(g + df)$
        $\langle g, f \rangle = \langle g + df, 2f \rangle$
    fi fi
    $k = k + 1$
od
return g
end

Fig. 2. The 2-Adic Rational Approximation Algorithm.

1. We have

$$\begin{aligned} g'(\alpha) &= \alpha \cdot g_2' - g_1' \\ &= \alpha \cdot (f_2 + dg_2) - (f_1 + dg_1) \\ &= f(\alpha) + dg(\alpha) \\ &\equiv 2^k + d2^k \bmod 2^{k+1} \\ &\equiv 0 \bmod 2^{k+1}, \end{aligned}$$

   since f and g are in $L_k - L_{k+1}$ and d is odd. It follows that $g' \in L_{k+1}$. Also, g is in L_k, so $f' = 2g$ is in L_{k+1}.
2. By De Weger's lemma, we have $|f_1 g_2 - f_2 g_1| = 2^k$. Therefore $|f_1' g_2' - f_2' g_1'| = |2g_1(f_2 + dg_2) - 2g_2(f_1 + dg_1)| = |2(f_1 g_2 - f_2 g_1)| = 2^{k+1}$. Again by De Weger's lemma, g', f' is a basis for L_{k+1}.
3. We have $g \in \mathbf{Z} \times \mathbf{Z} - L_{k+1}$, so $f' = 2g \in 2(\mathbf{Z} \times \mathbf{Z}) - L_{k+2}$.
4. Suppose that minimality fails. By the fact that f', g' form a basis for L_{k+1}, this means that there are integers a and b so that

$$\Phi(ag' + bf') < \Phi(g') \tag{2}$$

and $ag_2' + bf_2'$ is odd. The latter condition is equivalent to a being odd since f_2' is even and g_2' is odd. By possibly negating both a and b, we can assume a is nonnegative. Further, if $a = 1$, then $ag' + bf' = f + (d + 2b)g$ and this contradicts the choice of d in the algorithm. Thus we can assume that $a > 1$. Equation (2) can be rewritten

$$\Phi(af + (ad + 2b)g) < \Phi(f + dg).$$

Let c be the odd integer closest to $d + 2b/a$. Then $|c - (d + 2b/a)| \leq (a-1)/a$. It follows that

$$\begin{aligned}
\Phi(f + cg) &= \Phi\left(f + \left(d + \frac{2b}{a}\right)g + \left(c - \left(d + \frac{2b}{a}\right)\right)g\right) \\
&\leq \Phi\left(f + \left(d + \frac{2b}{a}\right)g\right) + \Phi\left(\left(c - \left(d + \frac{2b}{a}\right)\right)g\right) \\
&\leq \frac{1}{a}\Phi(af + (d + 2b)g) + \frac{a-1}{a}\Phi(g) \\
&< \frac{1}{a}\Phi(f + dg) + \frac{a-1}{a}\Phi(g) \\
&\leq \frac{1}{a}\Phi(f + dg) + \frac{a-1}{a}\Phi(f + dg) \\
&= \Phi(f + dg),
\end{aligned}$$

where we have used the triangle inequality for Φ, and inductively used the minimality condition on g. This contradicts the choice of d.

5. Suppose there is an element $h' \in L_{k+1}$ with h_1' and h_2' even, such that $\Phi(h') < \Phi(f') = 2\Phi(g)$. We can write $h' = 2h$ for some $h \in L_k$. If both h_1 and h_2 are even, then $\Phi(h) < \Phi(g) < \Phi(f)$ by the inequality in the algorithm leading to this case. This contradicts the minimality of f. If h_2 is odd, then $\Phi(h) < \Phi(g)$ contradicts the minimality of g. It is impossible that h_1 is odd and h_2 is even for $k \geq 1$ since $h(\alpha) \equiv 0 \bmod 2^k$. □

We observe that point (5) of the lemma is not strictly necessary for convergence of the algorithm. In fact, the algorithm runs correctly if we always update g and f by the first method of update, independent of the relation between $\Phi(f)$ and $\Phi(g)$. However, point (5) ensures that the size of f is small and leads to better bounds on the complexity.

Suppose the input sequence to the algorithm is, in fact, an eventually periodic sequence giving a rational 2-adic $\alpha = p/q$. We want to know how many iterations of the rational approximation algorithm are required to output (p, q).

Theorem 6. *Suppose* $\mathbf{a} = a_0, a_1, a_2, \cdots$ *is an eventually periodic sequence with associated 2-adic number* $\alpha = \sum a_i 2^i = p/q$, $p, q \in \mathbf{Z}$, *and* $\gcd(p, q) = 1$. *Then the 2-adic rational approximation algorithm outputs* (p, q) *if* $T \geq \lceil 2 \log \Phi(p, q) \rceil + 2$ *bits are used.*

Proof: The output from the algorithm is a pair (g_1, g_2) satisfying $g_1 - \alpha g_2 \equiv 0 \bmod 2^T$. Hence $g_1 q \equiv p g_2 \bmod 2^T$. By the Φ-minimality, we have $\Phi(g_1, g_2) \leq \Phi(p, q)$. It follows that $|g_1 q| \leq \Phi(p,q)^2 \leq 2^{T-2}$ and $|p g_2| \leq \Phi(p,q)^2 \leq 2^{T-2}$. Therefore, $g_1/g_2 = p/q$. Again by the Φ-minimality, we must have $g_1 = p$ and $g_2 = q$. □

It is preferable to bound the number of bits needed by the 2-adic span (the size of the smallest FCSR that outputs **a**). The 2-adic span is related to Φ by the following.

Theorem 7. *If $\alpha = \sum_{i=0}^{\infty} a_i 2^i = p/q$ is the rational number corresponding to* **a**, *reduced to lowest terms, then the 2-adic span is bounded by*

$$\lceil \log \Phi(p,q) \rceil - \lceil \log \log \Phi(p,q) \rceil \leq \lambda_2(\mathbf{a}) \leq \lceil \log \Phi(p,q) \rceil + \lceil \log \log \Phi(p,q) \rceil .$$

It follows that

$$\lambda_2(\mathbf{a}) - \lceil \log(\lambda_2(\mathbf{a})) \rceil - 1 \leq \lceil \log \Phi(p,q) \rceil \leq \lambda_2(\mathbf{a}) + \lceil \log(\lambda_2(\mathbf{a})) \rceil + 1.$$

This gives the desired bound.

Corollary 8. *Suppose* **a** *is an eventually periodic sequence with associated 2-adic number $\alpha = p/q$, $\gcd(p,q) = 1$. Then the 2-adic rational approximation algorithm outputs (p,q) if $T \geq 2\lambda_2(\mathbf{a}) + 2\lceil \log(\lambda_2(\mathbf{a})) \rceil + 4$ bits are used.*

In implementing the algorithm, the value of d can be found by division. For example, suppose we are in the case where $g(\alpha) \not\equiv 0 \bmod 2^{k+1}$ and $\Phi(g) < \Phi(f)$. If $g_1 \neq \pm g_2$, then d is an odd integer among the four odd integers immediately less than or greater than $(f_2 - f_1)/(g_1 - g_2)$ and $-(f_1 + f_2)/(g_1 + g_2)$. Thus it suffices to consider the value of $\Phi(f + dg)$ for these four values of d. When $g_1 = \pm g_2$, one or the other of these quotients is not considered, and in the second case of the algorithm, the roles of f and g are switched.

4 Initial Loading of a FCSR

Now let us show how to construct a FCSR which generates the bit sequence for a given rational number p/q. We assume q is a positive odd integer and let $r = \lfloor \log_2(q+1) \rfloor$. Write $q = \sum_{i=0}^{r} q_i 2^i$ with $q_0 = -1$ and $q_i \in \{0, 1\}$ for $i > 0$. We want to determine the initial setting (including the extra memory) of the FCSR with connection integer q that outputs the 2-adic expansion of p/q. The number of nonzero taps in such a FCSR is $t = wt(q+1)$, the Hamming weight of the binary expansion of $q+1$. The initial setting is related to p and q by the following proposition.

Proposition 9. *Suppose a FCSR with connection integer $q = \sum_{i=0}^{r} q_i 2^i$ is set up with initial setting $a_0, \cdots, a_{r-1}$ and initial memory m. Then its output is the 2-adic expansion of the fraction p/q where*

$$p = \sum_{k=0}^{r-1} \sum_{i=0}^{k} q_i a_{k-i} 2^k - m 2^r. \tag{3}$$

Furthermore, if we let $x = \sum_{j=0}^{r-1} a_j 2^j$, then the double sum in equation (3) is the product qx with all products of terms whose indices sum to r or more omitted. Such a double sum can be computed essentially as quickly as the full product qx. It follows that, for a given fraction p/q, the initial loading can be derived by the following steps.

B1. Compute $a_0 + a_1 2 + \cdots + a_{r-1} 2^{r-1} = p/q \bmod 2^r$. In general computing modular quotients is apparently hard. However, when the modulus is a power of a prime they can be computed efficiently. It is straightforward to do so in time $\mathcal{O}(r^2)$.

B2. Compute $y = \sum_{k=0}^{r-1} \sum_{i=0}^{k} q_i a_{k-i} 2^k$, say by a modified multiplication algorithm.

B3. Compute $m = (y - p)/2^r$ in time $\mathcal{O}(r)$.

We can then use $a_0, \cdots, a_{r-1}$ as the initial loading and m as the initial memory in a FCSR with connection integer q. This FCSR will output the 2-adic expansion of p/q. Of course if p and q have been determined by the rational approximation algorithm, then we already have the initial loading $a_0, \cdots, a_{r-1}$ and need only determine the initial memory m from steps (**B2**) and (**B3**).

5 Complexity Issues

Suppose the rational approximation algorithm is executed with a sequence $\mathbf{a}$ which is eventually periodic, with rational associated 2-adic number $\alpha = p/q$. Let λ be the 2-adic span of $\mathbf{a}$. Then the rational approximation algorithm takes $T = 2\lambda + 2\lceil \log(\lambda) \rceil + 4$ steps.

Consider the kth step. If $g(\alpha) \not\equiv 0 \bmod 2^{k+1}$, then we say that a discrepancy has occurred. The complexity of the algorithm depends on the number of discrepancies. To simplify the computation of αg_2, we maintain αf_2 as well. When no discrepancy occurs, these values and the value of f can be updated with k bit operations.

Suppose a discrepancy occurs. The minimization step can be done with two divisions of k bit integers. The remaining steps take time $\mathcal{O}(k)$. Then αg_2 and αf_2 can be updated with $\mathcal{O}(k)$ bit operations and two multiplications of k bit integers by d.

Let D be the number of discrepancies, and let M be the maximum time taken by a multiplication or division of T bit integers. The Schönhage-Strassen algorithm [20], gives $M = \mathcal{O}(T \log T \log\log T)$. This can be improved to $M \sim r \log r$ using Pollard's nonasymptotic algorithm and Newton interpolation for $T < 2^{37}$ on a 32 bit machine or $T < 2^{70}$ on a 64 bit machine [18]. These are ranges that are typical in current usage.

When T bits of input are used, this form of the algorithm guarantees that both f and g have at most T bits. This follows for g because $(\alpha \bmod 2^k, 1)$ is in L_k, and by induction for f since the f at stage k requires at most one nore bit than the maximum required by f and g at stage $k-1$. The complexity of the algorithm is thus $4DM + \mathcal{O}(T^2)$. Strictly in terms of T, this is $\mathcal{O}(T^2 \log T \log\log T)$.

However, if the sequence is chosen so the number of discrepancies is small, the complexity is lower. In particular a cryptographer designing a stream cipher should try to choose sequences for which many discrepancies occur.

6 Conclusions

We have exhibited an algorithm for synthesizing a minimal size feedback with carry shift register that outputs a sequence given a relatively small number of its bits. This is a general approach to attacking stream ciphers, and 2-adic span, the associated security measure, must now be considered whenever cryptologists design stream cipher.

This approach can be used to attack Massey and Rueppel's summation combiner [16, 19]. In their setup, the outputs from several short maximal period LFSRs (the outputs of such sequences are called *m-sequences*) with pairwise relatively prime periods are combined using addition with carry. It has been shown that the linear span of the resulting sequence tends to be close to the period of the sequence. This period is the product of the periods of the m-sequences, and is exponentially larger than the sizes of the original LFSRs. For this reason, summation combiners have been suggested for use in stream ciphers.

However, addition with carry is precisely addition in the 2-adic numbers. (In fact, it was this observation that motivated this work.) If $\mathbf{a}$ and $\mathbf{b}$ are two sequences, and $\mathbf{c}$ is the result of combining them with a summation combiner, then $\lambda_2(\mathbf{c})$ is at most $\lambda_2(\mathbf{a}) + \lambda_2(\mathbf{b}) + 2\lceil\log(\lambda_2(\mathbf{a}))\rceil + 2\lceil\log(\lambda_2(\mathbf{b}))\rceil + 6$. Even if these 2-adic spans are maximal (and ensuring this is problematic), we will be able to determine the resulting sequence from far fewer bits than previously thought. For example, if we combine m-sequences of period $2^n - 1$ for $n = 7, 11, 13, 15, 16, 17$, then the resulting sequence has linear span nearly 2^{79}, but the 2-adic span is less than 2^{18}. Thus 2^{19} bits suffice to determine this sequence – *and far fewer unless care is taken in the choice of the m-sequences.*

How can we build summation combiners that are secure against this sort of attack? It is necessary to choose underlying LFSRs whose output sequences have large 2-adic span. Suppose, for example, we hope to have 2^{60} secure bits. We might build a summation combiner based on maximal period LFSRs with length about 60. The resulting sequence is certainly secure against the classical Berlekamp-Massey attack. However, it is only secure against the attack described in this paper if we choose the individual output sequences have 2-adic spans close to their periods.

A slightly different analysis arises if we consider security based on complexity. If we consider a system secure if it takes 2^{60} word operations to crack it, then it must have 2-adic span at least 2^{27}. If we allow 2^{70} word operations, the 2-adic span must be 2^{32}. However, the analysis we have given is a worst case analysis. The actual speed of the rational approximation algorithm depends on the number of updates that must be performed. Thus the sequences chosen must not only have large 2-adic span, they must guarantee that many updates occur in the rational approximation algorithm.

Many other questions remain concerning FCSRs and 2-adic span. To understand the average behavior of the algorithm, we are led to the question of how the 2-adic span varies as a random sequence is lengthened. This is closely related to the question of what the 2-adic span of an average sequence is. Experimental evidence suggests that on average the 2-adic span is about half the length of the sequence. This would be consistent with what happens in the case of linear span, and would imply that on average many updates occur in the rational approximation algorithm.

The question of how to generate sequences with large 2-adic span is now an important one. We cannot have secure stream ciphers without such sequences. In fact we need to be able to generate sequence that simultaneously have large 2-adic and large linear span,

Finally, a number of generalizations of FCSRs have been suggested, based on other complete valued fields over number fields [8, 10]. Generalization of the ideas in this paper to p-adic shift registers (p a prime) is straightforward. Generalizations to registers over ramified and unramified extensions of the 2-adic (or p-adic) numbers are more difficult. With care, the registers can be constructed and have the appropriate algebraic structures. However, the rational approximation algorithm only appears to generalize under very special conditions. For example, we must have an algebraic number field whose ring of integers is a Euclidean domain. Furthermore, it seems that the convergence rate can only be guaranteed if the norm function on this field has certain properties.

7 Acknowledgements

The authors thank J. Goldsmith for several helpful suggestions for improving the manuscript.

References

1. E. Biham and A. Shamir: Differential Cryptanalysis of DES-like Cryptosystems, *Journal of Cryptology*, vol. 4, 1991, pp.3-72.
2. L. Blum, M. Blum, and M. Shub: A simple unpredictable pseudo-random number generator, *Siam J. Comput.* vol. 15, pp. 364-383 (1986).
3. U. Cheng: On the continued fraction and Berlekamp's algorithm. *IEEE Trans. Info. Theory* vol. 30, 1984 pp. 541-544.
4. Z. D. Dai and K. C. Zeng: Continued fractions and the Berlekamp-Massey algorithm. *Auscrypt '90*, Springer Lecture Notes in Comp. Sci. vol. 453, Springer Verlag, N. Y., 1990.
5. S. Golomb: *Shift Register Sequences*. Aegean Park Press,
6. R. T. Gregory and E. V. Krishnamurthy: *Methods and Applications of Error-Free Computation*, Springer Verlag, N. Y., 1984.
7. A. Klapper and M. Goresky: 2-Adic Shift Registers, *Fast Software Encryption: Proceedings of 1993 Cambridge Security Workshop, Springer-Verlag LNCS*, vol. 809, 1994, pp. 174-178.

8. A. Klapper, and M. Goresky: Feedback Registers Based on Ramified Extensions of the 2-Adic Numbers, *Proceedings, Eurocrypt 1994*, Perugia, Italy,
9. A. Klapper and M. Goresky: Feedback Shift Registers, Combiners with Memory, and Arithmetic Codes, *University of Kentucky, Deptartment of Computer Science Technical Report No. 239-93.*
10. A. Klapper: Feedback with Carry Shift Registers over Finite Fields, *Proceedings of Leuven Algorithms Workshop*, Leuven, Belgium, December, 1994.
11. N. Koblitz: *p-Adic Numbers, p-Adic Analysis, and Zeta Functions.* Graduate Texts in Mathematics Vol. 58, Springer Verlag, N. Y. 1984.
12. K. Mahler: On a geometrical representation of p-adic numbers, *Ann. of Math.* vol. 41, 1940 pp. 8-56.
13. D. Mandelbaum: An approach to an arithmetic analog of Berlekamp's algorithm. *IEEE Trans. Info. Theory,* vol. IT-30, 1984 pp. 758-762.
14. G. Marsaglia and A. Zaman: A new class of random number generators, *Annals of Applied Probability.* vol. 1, 1991 pp. 462-480.
15. J.L. Massey: Shift register sequences and BCH decoding, *IEEE Transactions on Infoormation Theory,* vol. IT-15, pp. 122-127, 1969.
16. J. Massey and R. Rueppel: Method of, and Apparatus for, Transforming a Digital Data Sequence into an Encoded Form, U.S. Patent No. 4,797,922, 1989.
17. W. H. Mills: Continued fractions and linear recurrences, *Math. Comp.* vol. 29, 1975, pp. 173-180
18. J. Pollard: The Fast Fourier Transform in a Finite Field, *Math. Comp.*, vol. 25, 1971, pp. 365-374.
19. R. Rueppel: *Analysis and Design of Stream Ciphers.* Springer Verlag, New York, 1986.
20. A. Schönhage and V. Strassen: Schnelle Multiplikation Grosser Zahlen, *Computing,* vol. 7, 1971, pp. 281-292.
21. B. M. M. de Weger: Approximation lattices of p-adic numbers, *J. Num. Th.* vol. 24, 1986, pp. 70-88.
22. L. R. Welch and R. A. Scholtz: Continued fractions and Berlekamp's algorithm, *IEEE Trans. Info. Theory,* vol. 25, 1979 pp. 19-27.