A generalized birthday approach for efficiently finding linear relations in l-sequences

Wang, Hui; Stankovski, Paul; Johansson, Thomas

Published in:
Designs, Codes and Cryptography

DOI:
10.1007/s10623-013-9845-0

2015

A Generalized Birthday Approach for Efficiently Finding Linear Relations in ℓ-sequences[*]

Hui Wang[†,‡] and Paul Stankovski[‡] and Thomas Johansson[‡]

[†] Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, P.R. China
[‡] Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden

Abstract. Feedback with Carry Shift Registers (FCSRs) have previously been available in two configurations, the Fibonacci and Galois architectures. Recently, a generalized and unifying FCSR structure and theory was presented. The new ring FCSRs model repairs some weaknesses of the older architectures. Most notably, the carry cell bias property that was exploited for an attack on the eSTREAM final portfolio cipher F-FCSR-H v2 is no longer possible for the updated (and unbroken) F-FCSR-H v3 stream cipher.

In this paper we show how to exploit a particular set of linear relations in ring FCSR sequences. We show what biases can be expected, and we also present a generalized birthday algorithm for actually realizing these relations. As all prerequisites of a distinguishing attack are present, we explicitly show a new such attack on F-FCSR-H v3 with an online time complexity of only $2^{37.2}$. The offline time complexity (for finding a linear relation) is $2^{56.2}$. This is the first successful attack on F-FCSR-H v3, the first attack to breach the exhaustive search complexity limit. Note that this attack is *completely* different from that of F-FCSR-H v2. We focus on this particular application in the paper, but the presented algorithm is actually very general. The algorithm can be applied to any FCSR automaton, so linearly filtered FCSRs and FCSR combiners may be particularly interesting targets for cryptanalysis.

Keywords: FCSR, Linear Relations, Generalized Birthday Attack, Distinguisher

1 Introduction

Feedback with Carry Shift Registers (FCSRs) were introduced by A. Klapper and M. Goresky in [10] as a nonlinear alternative to Linear Feedback Shift Registers (LFSRs). This intrinsic nonlinearity is due to the fact that FCSRs use integer addition with carries instead of the modular addition (xor) operator used in LFSRs, and this property somewhat relaxes the

[*] The final publication is available at `http://springerlink.com`.

nonlinearity requirements on other building blocks of cryptographic designs. The structural similarities between LFSRs and FCSRs imply that many of the good properties of LFSRs are inherited, but unlike most other nonlinear sequence generators, FCSRs have a well-studied algebraic structure, making analysis much easier.

Two different flavors of FCSRs have been available in the past; the Fibonacci and Galois architectures. Very recently, a new approach for FCSRs, ring FCSRs or ring representation, was proposed in [2]. An extended description of ring FCSRs is available in the journal version [3]. The new ring FCSR automaton and its associated theory truly do unify the previous architectures, but more interestingly, it also generalizes the entire FCSR concept by adding flexibility. While Fibonacci- and Galois-FCSRs use many-to-one- and one-to-many-feedbacks, respectively, ring FCSRs allow many-to-many-feedbacks (see Figure 3), resulting in a generalized theory based on the adjacency matrix of the corresponding automaton graph. Ring FCSRs retain many good properties of FCSRs and are more resistant to side-channel attacks. The new ring FCSRs model also repairs some of the weaknesses of the older architectures. Most notably, the carry cell bias property that was exploited by Hell and Johansson in [7] for an attack on the eSTREAM final portfolio cipher F-FCSR-H v2 is no longer possible for the updated and yet unbroken F-FCSR-H v3 stream cipher.

In this paper we show how to exploit a particular set of linear relations in ring FCSR sequences. We show what biases can be expected, and we also present a generalized birthday algorithm for actually realizing these relations. As all prerequisites of a distinguishing attack are present, we explicitly show a new such attack on F-FCSR-H v3 with an online time complexity of only $2^{37.2}$. The offline time complexity (for finding a linear relation) is $2^{56.2}$. This is the first successful attack on F-FCSR-H v3, the first attack to breach the exhaustive search complexity limit of 2^{80} (initializations). Note that this attack is *completely* different from that of F-FCSR-H v2. We focus on this application in the paper, but the presented algorithm is actually very general. The algorithm can be applied to any FCSR automaton, so linearly filtered FCSRs and FCSR combiners may be particularly interesting targets for cryptanalysis.

The paper is organized as follows. In Section 2 we cover some basics of FCSRs and ℓ-sequences. In Section 3 we introduce some theory for new linear relations and show our generalized birthday algorithm, and Section 4 shows some simulation results that verify our theory. We discuss applications in Section 5, where we detail the attack on F-FCSR-H v3. The paper is concluded in Section 6.

2 Preliminaries

An FCSR is a device that produces the 2-adic expansion of a rational number p/q for some integers p and q, where q is odd. Traditionally, this device has been available in two different configurations; Fibonacci and Galois representation, named analogously to the corresponding LFSR representations that are very similar in structure. Only recently, a new and more general FCSR configuration was presented, the ring representation, which generalizes the two previous architectures.

In FCSR-based stream ciphers, p usually depends on the secret key and the initialization vector (IV), and q is a public parameter. The choice of q induces a number of FCSR properties. The arguably most important property is that it completely determines the length of the period of the binary sequence that the FCSR outputs.

Roughly following [6], we now give a very brief overview of Fibonacci and Galois FCSRs. We then introduce the relevant background information on ring FCSRs and FCSR sequences that we will be using in this paper.

2.1 Fibonacci FCSR basics

The state of a Fibonacci FCSR of size n consists of two principal parts, the main register $\mathbf{m} = (m_0, m_1, ..., m_{n-1})$ and a $(\lfloor \log_2 (wt\,(q+1)) \rfloor + 1)$-bit memory register b, where wt is the weight function that reports the number of ones in the binary expansion of a number (Hamming weight). Let $\mathbf{m}(t)$ and $b(t)$ denote the state of the registers $\mathbf{m}$ and b at time t. State updates are performed according to

$$\begin{cases} m_i(t+1) = m_{i+1}(t), \qquad \text{for } 0 \leq i \leq n-2, \\ \sigma(t+1) = b(t) + \displaystyle\sum_{i=1}^{n} q_i m_{n-i}(t), \\ m_{n-1}(t+1) = \sigma(t+1) \bmod 2, \\ b(t+1) = \sigma(t+1) \text{ div } 2. \end{cases}$$

The auxiliary values $\sigma(\cdot)$ are used for clarity.

The connection integer q of the corresponding FCSR is defined as

$$q = q_n 2^n + q_{n-1} 2^{n-1} + \ldots + q_2 2^2 + q_1 2 - 1.$$

An example of a Fibonacci FCSR (with $q = 85$) is given in Figure 1. It should be noted that the box labeled $\boxplus$ is a full adder that adds all inputs

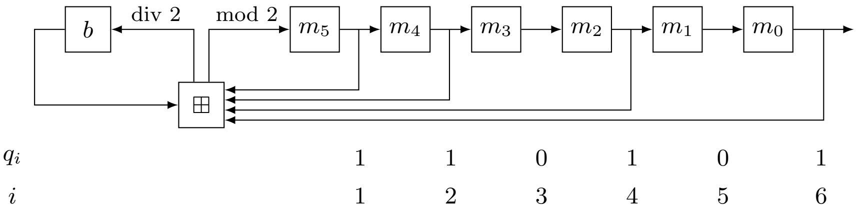

Fig. 1. The Fibonacci FCSR with connection integer $q = 85$.

into a (regular) sum σ. The value σ mod 2 is then fed into m_{n-1}, while σ div 2 is stored in the memory register b.

2.2 Galois FCSR basics

The state of a Galois FCSR of size n also consists of two principal parts, the main register $\mathbf{m} = (m_0, m_1, ..., m_{n-1})$, as before, and a carries register $\mathbf{c} = (c_0, c_1, ..., c_{n-2})$. We additionally let $\mathbf{c}(t)$ denote the state of the register $\mathbf{c}$ at time t.

As for Fibonacci FCSRs, define the connection integer as

$$q = q_n 2^n + q_{n-1} 2^{n-1} + \ldots + q_2 2^2 + q_1 2 - 1,$$

and let $d = \frac{q+1}{2} = (d_{n-1} \ldots d_0)_{binary}$. The carries register contains l active cells, where $l + 1$ is the number of nonzero binary digits d_i in d. Disregarding d_{n-1}, which must always be set, the active carry cells correspond to the $d_i = 1$ in the interval $0 \leq i \leq n - 2$.

We continue to let $\boxplus$ denote (regular) addition with carry, but in the Galois case we will only need one memory bit per carry. The $\boxplus$ operator now takes three inputs in total, two external inputs and the carry bit. For every clocking it first computes the (regular) sum σ of all three input bits. It then feeds σ mod 2 into the succeeding main memory cell and stores the bit σ div 2 in the carry register. The state update mechanism can be summarized as follows.

$$\begin{cases} \sigma_i(t+1) = m_{i+1}(t) + c_i(t) + d_i m_0(t), & \text{for } 0 \leq i \leq n-2, \\ m_i(t+1) = \sigma_i(t+1) \bmod 2, & \text{for } 0 \leq i \leq n-2, \\ c_i(t+1) = \sigma_i(t+1) \operatorname{div} 2, & \text{for } 0 \leq i \leq n-2, \\ m_{n-1}(t+1) = m_0(t). & \end{cases}$$

Following [1], we specifically illustrate the case $q = 347$ in Figure 2, which gives us $d = 174 = (10101110)_{binary}$.

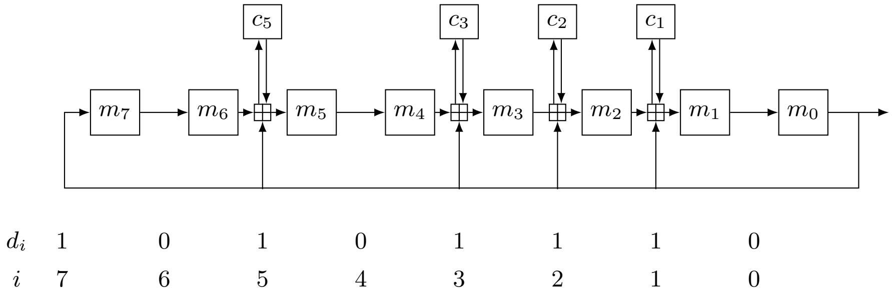

Fig. 2. The Galois FCSR with connection integer $q = 347$.

2.3 Ring FCSR basics

Contrary to the Fibonacci and Galois representations, ring FCSRs allow full freedom in tap placement and can be seen as a generalization of the two previous architectures. Using main and carries registers $\mathbf{m} = (m_0, m_1, ..., m_{n-1})$ and $\mathbf{c} = (c_0, c_1, ..., c_{n-1})$ to describe the state of a ring FCSR of size n, updates are performed according to

$$\begin{cases} \boldsymbol{\sigma}(t+1) = \mathbf{Tm}(t) + \mathbf{c}(t), \\ \mathbf{m}(t+1) = \boldsymbol{\sigma}(t+1) \bmod 2, \\ \mathbf{c}(t+1) = \boldsymbol{\sigma}(t+1) \text{ div } 2, \end{cases}$$

where $\mathbf{T} = (t_{i,j})_{0 \le i,j < n}$ is the $n \times n$ transition matrix of the corresponding automaton graph. We have

$$t_{i,j} = \begin{cases} 1 & \text{if } m_j \text{ is used to update } m_i, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1 (Connection integer of a ring FCSR). *Let* $\mathbf{T}$ *be the transition matrix of a ring FCSR. The connection integer* q *is then given by* $q = \det(\mathbf{I} - 2\mathbf{T})$.

While Definition 1 defines q as a negative number, we will abuse this notation by disregarding the sign in the sequel. To obtain a ring FCSR with desirable properties, $\mathbf{T}$ should be chosen according to the guidelines listed in Section 5.1 in [2]. A ring FCSR with connection integer $q = 243$ is illustrated in Figure 3 with an equivalent alternative view in Figure 4.

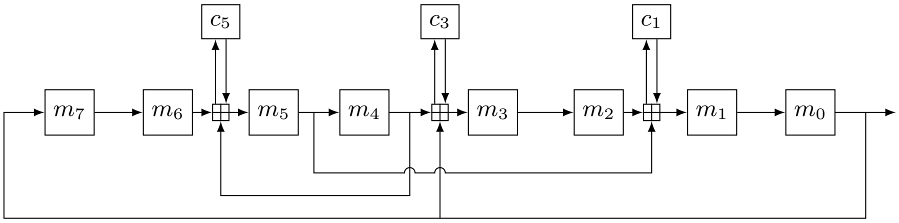

Fig. 3. A ring FCSR with connection integer $q = 243$.

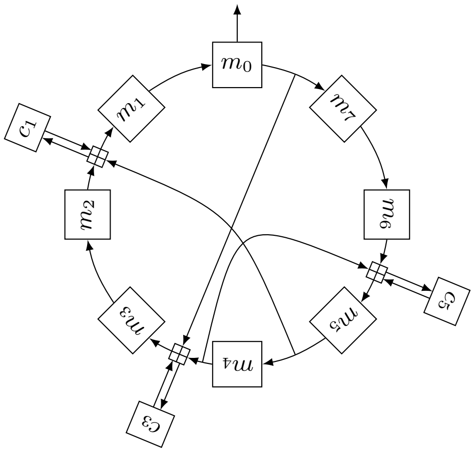

Fig. 4. Alternative view of a ring FCSR with connection integer $q = 243$.

2.4 Basic properties of FCSR sequences

The output sequence $\underline{a} = \{a_t\}_{t=0}^{\infty}$ of an FCSR automaton is called an FCSR sequence. For all FCSR architectures we let the output bits be defined by the contents of the main register cell m_0 according to

$$a_{t+1} = m_0(t).$$

Lemmas 1 and 2 from [11] introduce some notations and basic properties of FCSR sequences.

Lemma 1. *The output sequence $\underline{a} = \{a_t\}_{t=0}^{\infty}$ of an FCSR with connection integer q corresponds to a rational number $\beta = \frac{p}{q}$. Specifically, $\sum_{t=0}^{\infty} a_t 2^t = \frac{p}{q}$, which means that $\underline{a}$ is the 2-adic representation of β. The sequence $\underline{a}$ is strictly periodic if and only if $\beta \leq 0$ and $|\beta| < 1$.*

The same FCSR sequence can be generated by many different FCSRs with different connection integers, but we will further abuse notation somewhat by letting the connection integer q of an FCSR sequence refer to the smallest possible such integer.

Definition 2 (ℓ-sequence). *An ℓ-sequence $\underline{a}$ is an FCSR sequence with connection integer q for which 2 is a primitive root, where $q = p^e$ ($e \geq 1$) and p is an odd prime. The period of $\underline{a}$ is $\varphi(q)$, where φ is the Euler totient function.*

Lemma 2. *Given an ℓ-sequence $\underline{a} = \{a_t\}_{t=0}^{\infty}$ with connection integer q, there exists a unique integer A, $0 \leq A < q$, such that*

$$a_t = \alpha_t \bmod 2, \quad t \geq 0,$$

with $\alpha_t = (A\gamma^t) \bmod q$, where γ is the multiplicative inverse of 2 in the ring of integers modulo q.

We will also be using an interesting observation from [2] that relates the contents of the main memory cells.

Lemma 3. *Given a ring FCSR with connection integer q for which 2 is a primitive root, $q = p^e$ ($e \geq 1$) and p is an odd prime, the output sequence of each main register cell is an ℓ-sequence with connection integer q.*

2.5 Linear properties of ℓ-sequences

We now define a particular set of linear relations.

Definition 3 ($(i + j)$-relations $\mathcal{R}_q$). *Let a connection integer q and positive integers $u_1, \cdots, u_i$ and $v_1, \cdots, v_j$ be given. The linear relation*

$$2^{-u_1} + \cdots + 2^{-u_i} \equiv 2^{-v_1} + \cdots + 2^{-v_j} \bmod q$$

*is called a $(i + j)$-*relation *and is denoted $\mathcal{R}_q(u_1, \ldots, u_i; v_1, \ldots, v_j)$.*

From [14] we take some definitions and results on ℓ-sequences that we either state directly or expand upon. Define the r-tuple set

$$\Omega_r(q) = \{\langle i_1, \ldots, i_r\rangle \,|i_1, \ldots, i_r \in \mathbb{Z}_q \backslash \{0\}, i_1 + \cdots + i_r \not\equiv 0 \bmod q\}.$$

The available keystream is always finite in practice, so from now on we consider finite ℓ-sequences only. A linear property of ℓ-sequences can now be described as follows.

Theorem 1. **[14]((3+1)-relation bias)** *Let an ℓ-sequence $\underline{a} = \{a_t\}_{t=s+d}^{s+e+k-1}$ with connection integer q and a $(3+1)$-relation $\mathcal{R}_q(w, x, y; z)$, where $d = \min(w, x, y, z)$, $e = \max(w, x, y, z)$, $s \geq 0$ and $\alpha_t = (A2^{-t}) \bmod q$ as in Lemma 2 be given. If the triplet sequence*

$$\{\langle \alpha_{t+w}, \alpha_{t+x}, \alpha_{t+y} \rangle\}_{t=s}^{s+k-1}$$

cannot be distinguished from a uniformly random triplet sequence over $\Omega_3(q)$, then the k events

$$a_{t+w} \oplus a_{t+x} \oplus a_{t+y} \oplus a_{t+z} = 0, \qquad 0 \leq t < k,$$

can be seen as k independent Bernoulli trials with success probability $\frac{1}{3}$.

In short, Theorem 1 shows that $(3+1)$-relations have bias $\frac{1}{3}$, and Theorem 1 can easily be generalized to $(m+1)$-relations. Specifically, the bias is zero when m is even. That is, relations of odd weight do not exhibit a bias that we can detect. Furthermore, the bias is non-zero when m is odd, decreasing as m grows. These non-zero biases stem from properties of modular addition.

Although [14] considers Galois and Fibonacci FCSRs, the proof of Theorem 1 only assumes ℓ-sequences, so the result immediately carries over to ring FCSRs.

It should also be noted that Theorem 1 above is stated as in [14]. The condition on the triplet sequence over $\Omega_3(q)$ is of particular interest. This condition surely does not hold for a known value q since $\alpha_{t+1} = (\alpha_t 2^{-1}) \bmod q$. However, the situation is not as bad as it appears. Loosely put, it is sufficient if the corresponding $a_t = LSB(\alpha_t)$ behave randomly in a statistical sense. While a sequence $\{\alpha_t\}$ is fully determined after the first term has been observed, the corresponding binary sequence $\{a_t\}$ is much more well-behaved, even for a known q. The requirement stated for Theorem 1 is stronger than it needs to be – it is possible to relax the requirement somewhat.

As Theorem 1 is stated it appears that one would need to find a suitable triplet sequence as described above in order to realize the attack, but this is not necessary in practice. In [14], Tian and Qi used simulations to verify that the triplet condition is reasonable for practical applications.

The possibility to use $(3+1)$-relations for cryptanalytic attacks was identified in [14], but the briefly outlined algorithm has a complexity that exceeds 2^{80}, which is the exhaustive search complexity for F-FCSR-H v3. Also, the estimated data complexity of the resulting distinguisher is underestimated by a factor of about 32, which is pointed out in Section 5.

3 Generalizing linear relations of ℓ-sequences

3.1 Bias of $(2+2)$-relations

Armed with the bias for $(3+1)$-relations, we are now encouraged to find corresponding results for $(2+2)$-relations. A motivational factor here is that balanced equations, with two terms on either side in this case, are suitable for birthday attack approaches.

It turns out that transforming Theorem 1 from $(3+1)$- to $(2+2)$-relations is not all that hard. All we need is two simple lemmas.

Lemma 4. *If 2 is a primitive root of q, then $2^{\frac{\varphi(q)}{2}} \equiv -1 \bmod q$. We thus have $2^{i+\frac{\varphi(q)}{2}} \equiv -2^i \bmod q$.*

The following lemma is a direct consequence of Proposition 1 in [5].

Lemma 5. *Using the notation in Lemma 2, we have $a_t = a_{t+\frac{\varphi(q)}{2}} \oplus 1$ and $\alpha_t = q - \alpha_{t+\frac{\varphi(q)}{2}}$.*

Using Lemma 5, we can now relate a triplet sequence to a $(2+2)$-relation.

Theorem 2 **($(2+2)$-relations have bias $\frac{1}{3}$).** *Let $\underline{a} = \{a_t\}_{t=s+d}^{s+e+k-1}$ be an ℓ-sequence with connection integer q and a $(2+2)$-relation $\mathcal{R}_q(w,x;y,z)$ where $d = \min(w,x,y,z)$, $e = \max(w,x,y,z)$, $s \geq 0$ and $\alpha_t = (A2^{-t}) \bmod q$ as in Lemma 2 be given. If the triplet sequence*

$$\{\langle \alpha_{t+w}, \alpha_{t+x}, q - \alpha_{t+y}\rangle\}_{t=s}^{s+k-1}$$

cannot be distinguished from a uniformly random triplet sequence over $\Omega_3(q)$, then the k events

$$a_{t+w} \oplus a_{t+x} \oplus a_{t+y} \oplus a_{t+z} = 0, \qquad 0 \leq t < k, \tag{1}$$

can be seen as k independent Bernoulli trials with success probability $\frac{2}{3} = \frac{1}{2}\left(1+\frac{1}{3}\right)$.

Proof. Given $2^w + 2^x \equiv 2^y + 2^z \bmod q$, we have $2^w + 2^x - 2^y \equiv 2^z \bmod q$. Using Lemma 4, since 2 is a primitive root of q, we have $2^w + 2^x + 2^{y+\frac{\varphi(q)}{2}} \equiv 2^z \bmod q$. Thus $\mathcal{R}_q(w, x, y + \frac{\varphi(q)}{2}; z)$ is a $(3+1)$-relation. Using Lemma 5, we can write the triplet sequence given above as

$$\left\{\left\langle \alpha_{t+w}, \alpha_{t+x}, \alpha_{t+y+\frac{\varphi(q)}{2}} \right\rangle\right\}_{t=s}^{s+k-1}.$$

From Theorem 1 we know that the k events

$$a_{t+w} \oplus a_{t+x} \oplus a_{t+y+\frac{\varphi(q)}{2}} \oplus a_{t+z} = 1, \qquad 0 \leq t < k, \tag{2}$$

can be seen as k independent Bernoulli trials with success probability $\frac{2}{3}$. Using Lemma 5, if we replace $a_{t+y+\frac{\varphi(q)}{2}}$ in Equation (2) by $a_{t+y} \oplus 1$, then Equation (1) can be obtained. □

The point of Theorem 2 is that the bias of $(2+2)$-relations is $\frac{1}{3}$, which is very large in this context. Theorem 2 can also easily be generalized to $(n+n)$-relations.

For the sake of our upcoming F-FCSR-H v3 analysis, we also consider what happens when several ℓ-sequences are combined by bitwise xor. Theorem 3 provides some answers.

Theorem 3 (Bias of xored ℓ-sequences). *Let $\mathcal{R}_q(w,x;y,z)$ be a given $(2+2)$-relation. Given m* independent *ℓ-sequences $\underline{a^i} = \{a^i_t\}_{t=s+d}^{s+e+k-1}, 1 \leq i \leq m$, with connection integer q where $d = \min(w,x,y,z)$ and $e = \max(w,x,y,z)$, with α^i_t defined correspondingly for each i. If the following m triplet sequences*

$$\left\{\left\langle \alpha^i_{t+w}, \alpha^i_{t+x}, q - \alpha^i_{t+y} \right\rangle\right\}_{t=s}^{s+k-1}$$

for $1 \leq i \leq m$ cannot be distinguished from a uniformly random triplet sequence over $\Omega_3(q)$, then the k events

$$\bigoplus_{i=1}^{m} a^i_{t+w} \oplus a^i_{t+x} \oplus a^i_{t+y} \oplus a^i_{t+z} = 0$$

for $s \leq t < s+k$ are k independent Bernoulli trials with success probability $\frac{1}{2}\left(1 + \left(\frac{1}{3}\right)^m\right)$.

Proof. The independence of the m ℓ-sequences motivates using the piling-up lemma, see [12]. □

Theorem 3 considers the expected value of the bias, but for practical applications we need to know what level of accuracy we may expect. Proposition 1 gives us a practical error-bounding formula.

Proposition 1 (Bounding formula for bias of xored ℓ-sequences). *Using the notation from Theorem 3, let $\underline{b} = \{b_t\}_{t=s+d}^{s+e+k-1}$ be the bitwise* *xor of the sequences* $\underline{a}^i, 1 \leq i \leq m$. *For* $\mathcal{R}_q(w, x; y, z)$, *the bias of sequence* $\underline{b}$ *is defined as*

$$\varepsilon^q_{\underline{b}} = \frac{1}{k} \sum_{t=0}^{k-1} (-1)^{b_{t+w} \oplus b_{t+x} \oplus b_{t+y} \oplus b_{t+z}}. \tag{3}$$

Then, $\varepsilon^q_{\underline{b}}$ *satisfies*

$$\Pr \left[\varepsilon^q_{\underline{b}} \geq \left(\frac{1}{3} \right)^m - \frac{4.6802}{\sqrt{k}} \right] \approx 0.9999. \tag{4}$$

Proof. According to Theorem 3, the k events

$$b^i_{t+w} \oplus b^i_{t+x} \oplus b^i_{t+y} \oplus b^i_{t+z} = 0$$

for $0 \leq t < k$ can be seen as independent Bernoulli trials with success probability $p = \frac{1}{2} \left(1 + \left(\frac{1}{3} \right)^m \right)$. If $X_1, X_2, \ldots, X_k$ are random variables associated with these Bernoulli trials, we have

$$\Pr[X_t = 1] = p \text{ and } \Pr[X_t = 0] = 1 - p,$$

with $E(X_t) = \mu = p$ and $Var(X_t) = \sigma^2 = p(1-p)$ for each $0 \leq t < k$. Defining $Y_j = \sum_{i=1}^{j} X_i$, the Central Limit Theorem states that the probability distribution of

$$W_k = \frac{\frac{Y_k}{k} - \mu}{\frac{\sigma}{\sqrt{k}}} = \frac{\frac{Y_k}{k} - p}{\sqrt{\frac{p(1-p)}{k}}}$$

goes to $N(0, 1)$ in the limit as $k \to \infty$, see [9]. The value k can be regarded as sufficiently large when $k(1-p) \geq 5$ and $kp \geq 5$. According to the normal distribution table, we have

$$\Pr[W_k \geq -4.6802] \approx 0.9999,$$

that is

$$\Pr \left[\frac{Y_k}{k} \geq p - \frac{4.6802 \times \sqrt{p(1-p)}}{\sqrt{k}} \right] \approx 0.9999.$$

When m is large enough, say $m = 10$, $\sqrt{p(1-p)} \approx \frac{1}{2}$, so

$$\Pr \left[\frac{Y_k}{k} \geq p - \frac{2.3401}{\sqrt{k}} \right] \approx 0.9999.$$

Noting that $\varepsilon^q_{\underline{b}} = \frac{2Y_k}{k} - 1$, we finally deduce

$$\Pr \left[\varepsilon^q_{\underline{b}} \geq \left(\frac{1}{3} \right)^m - \frac{4.6802}{\sqrt{k}} \right] \approx 0.9999. \quad \square$$

Simulations indicating that Proposition 1 holds for practical applications can be found in Section 4.

3.2 A generalized birthday attack algorithm for finding $(2+2)$-relations

We now know the bias of $(2+2)$-relations, but we still need to be able to actually find them. In this section we introduce a generalized birthday attack algorithm that efficiently solves the problem. Our algorithm will find a $(2+2)$-relation $\mathcal{R}_q(w,x;y,z)$ with width $\max(w,x,y,z) - \min(w,x,y,z) < N$. The value N is tunable, chosen so that the running time of the algorithm is minimized while the success probability is sufficiently high. We need a definition for reduction purposes.

Definition 4 (k-small). *Given an integer q and a real number k, the integer n-tuple $\langle i_1, \ldots, i_n\rangle$ is k-small if $2^{-i_1} + \ldots + 2^{-i_n} \bmod q < \frac{q}{k}$.*

The k-small 1- and 2-tuples will be referred to as k-small numbers and pairs, respectively. Neither q nor k should be assumed to be small in the general case.

We first generate all k-small numbers in a specific interval, storing them in a table T_1. Note that each such number is smaller than $\frac{q}{k}$. We then form all possible (unordered) pairs $\langle w, x\rangle, 0 \leq w, x < N$, of these k-small numbers, storing them in a hash table T_2 (cuckoo hashing with $O(1)$ insertion and lookups is appropriate, see [13]) that stores value pairs keyed on their sum modulo q. We keep adding such pairs to T_2 until we find a collision. That is, we look for a set $\{\langle w,x\rangle, \langle y,z\rangle\}$ satisfying

$$2^{-w} + 2^{-x} \equiv 2^{-y} + 2^{-z} \bmod q.$$

Algorithm 1 specifies the details[1].

Because of the reduction, tables T_1 and T_2 will contain approximately $\frac{N}{k}$ and $\binom{\frac{N}{k}}{2}$ entries, respectively. The time complexity of Algorithm 1 is $N + \binom{\frac{N}{k}}{2}$, since this is the time it takes to build tables T_1 and T_2. We expect to find a collision after about

$$\binom{\frac{N}{k}}{2} = \sqrt{\frac{2q\alpha}{k}}$$

insertions into T_2, where we use $\alpha = 9.22$ to set the collision probability to at least 99% (see [16]). Minimizing the time complexity we get the

[1] From a notational point of view, the reader may think of both T_1 and T_2 as hash tables, where, e.g., $T_1[k] = v$ means insertion of value v keyed on k. While T_1 can be implemented as linear storage (an array) in practice (this should become clear in Section 3.3), T_2 needs to be implemented as a hash table.

Algorithm 1 – Generalized Birthday Approach to Finding a $(2+2)$-relation

Input: Integers N and q, real number k.
Output: $\mathcal{R}_q(w,x;y,z)$ or the phrase "No $(2+2)$-relation was found".

```
create empty tables $T_1$ and $T_2$;
$B$ = an integer in the interval $[0,q)$ chosen uniformly at random;
for ($i = B$; $i < B + N$; $i$++) { /* insert all $k$-small numbers in $[B, B+N)$ into $T_1$ */
    if ($i$ is $k$-small) {
        $T_1[i] = 2^{-i} \bmod q$;
    }
}
for (all pairs $\langle a, b\rangle$ of keys $a, b$ from $T_1$) {
    $s = T_1[a] + T_1[b]$;
    if (key $s$ is in $T_2$) {
        $\langle x, y\rangle = T_2[s]$; /* get pair $\langle x, y\rangle$ from $T_2$ */
        $m = \min(a, b, x, y)$; /* normalize relation */
        return $\mathcal{R}_q(a-m, b-m; x-m, y-m)$;
    }
    $T_2[s] = \langle a, b\rangle$; /* insert pair $\langle a, b\rangle$ into $T_2$ */
}
return "No $(2+2)$-relation was found";
```

conditions

$$\begin{cases} N = \binom{\frac{N}{k}}{2} \\ \binom{\frac{N}{k}}{2} = \sqrt{\frac{2q\alpha}{k}} \end{cases} \quad \Leftrightarrow \quad \begin{cases} N = 2k^2 \\ N^4 = 8k^3\alpha q, \end{cases}$$

which for F-FCSR-H v3 (using $\log_2 q < 160.26$) gives us $k = \left(\frac{\alpha q}{2}\right)^{1/5}$ and $N = 2^{66.0}$ for a total time complexity of $2N = 2^{67.0}$.

A few observations may and should be made at this point. First of all, Algorithm 1 is trivially generalized to $(n+n)$-relations, $n \geq 2$, and the corresponding minimization conditions are given by

$$\begin{cases} N = \binom{\frac{N}{k}}{n} \\ \binom{\frac{N}{k}}{n} = \sqrt{\frac{nq\alpha}{k}} \end{cases} \quad \Leftrightarrow \quad \begin{cases} N^{n-1} = n!k^n \\ N^{2n} = n(n!)^2 k^{2n-1}\alpha q. \end{cases}$$

For $n = 3$ for F-FCSR-H v3 we get $k = \left(\frac{\alpha q}{2}\right)^{1/4}$ and $N = 2^{62.3}$ for a total time complexity of $2N = 2^{63.3}$. The bias in this case is $\frac{2}{15}$.

Also, for $n \geq 4$, we can reduce the time complexity further by applying additional smallness reductions in the lower layers according to the generalized birthday approach [15]. In general, the time complexity decreases as n grows.

Secondly, we have tuned the parameters to make it sufficiently probable for us to find *one* linear relation. When we have reached the point of finding the first relation, it quickly becomes quite cheap to find many more collisions by increasing N slightly. This turns the algorithm into a cornucopia of $(n+n)$-relations. One possible usage for this is for a fast correlation attack. This would work very well in distinguishing situations where it is cheap to find many different relations compared to the amount of keystream needed for the distinguisher. One particular observation is important in this case. If we have

$$2^{-u_1} + \cdots + 2^{-u_n} \equiv 2^{-v_1} + \cdots + 2^{-v_n} \bmod q$$

for some numbers $u_1, \cdots, u_n$ and $v_1, \cdots, v_n$, then we also have

$$2^{-u_1+i} + \cdots + 2^{-u_n+i} \equiv 2^{-v_1+i} + \cdots + 2^{-v_n+i} \bmod q$$

for all i. This shows that we need to take dependency into account when we search for multiple linear relations. However, since we employ k-smallness, this effect will only be valid for a limited number of i in our case. Furthermore, the above observation could also be used for normalizing the parameters in an $(n+n)$-relation, as we do in Algorithm 1.

This normalization is performed at the end, just before the $(2+2)$-relation is output. Performing it any sooner would ruin the generalized birthday approach, as we then would be required to find a collision between *different* sets of pairs, ultimately degrading the complexity. Some further comments on dependency can be found in Section 3.4.

Third, the offline complexity measure used here involves mostly table insertions and lookups. A better comparison to exhaustive key search would compare these complexities to those of initializations. For F-FCSR-H v3, one initialization involves 48 FCSR updates. This should be compared to the operations in Algorithm 1, one modular exponentiation (that can be translated into a shift and a conditional subtraction) and insertion into tables T_1 and T_2. Since these operations do not need to be too complicated, we conclude that our offline complexity measure constitutes a conservative measure compared to initializations.

The online complexity measure, xoring $2n$ bits, is *much* cheaper than 48 FCSR updates (for n of reasonable size).

Fourth, the purpose of the randomly selected value B is to prevent algorithm designers from choosing some suitable q that will extend the running time of Algorithm 1. By choosing the starting point B at random, the expected time complexity does not depend on q.

And last but not least, Algorithm 1 is specified with a fixed N for clarity. In practice one can simply keep extending T_1 and T_2 until sufficiently many collisions have been found.

3.3 Improvement by modular interval summation

We now describe an improvement that makes it even cheaper to find (2+2)-relations, a technique that is also generalizable to $(n+n)$-relations. The general idea here is to improve Algorithm 1 by employing a more efficient utilization of the numbers that we generate for storage in table T_1. In the original setting described above, all numbers that are not k-small are simply discarded, and this is a waste of resources.

In the second part of Algorithm 1 we form all possible pairs of k-small numbers from table T_1 and store all such pairs in table T_2. Note that these pairs are $\frac{k}{2}$-small. The improvement is made possible by the fact that we do not necessarily need to sum two k-small numbers to make $\frac{k}{2}$-small pairs. The previously discarded numbers can be used to this end in the following way.

Instead of using one table T_1 we now use k tables $T_{1,0}, \ldots, T_{1,k-1}$ for the first step to store *all* numbers $2^{-i} \bmod q, B \leq i < B+N$. Divide the interval $[0, q)$ into k equally wide subintervals $\left[\frac{jq}{k}, \frac{(j+1)q}{k}\right), 0 \leq j < k$, and a number $2^{-i} \bmod q$ is put in table $T_{1,j}$ if it is in the j^{th} subinterval.

As before, we use all entries in table $T_{1,0}$ to form all possible $\frac{k}{2}$-small pairs and store these in table T_2. But we can also form $\frac{k}{2}$-small pairs by pairing any two number from tables $T_{1,1}$ and $T_{1,k-1}$, and any two numbers from tables $T_{1,2}$ and $T_{1,k-2}$, and so on, as shown in Figure 5. In this way we can increase the number of pairs we can generate by a factor of about k.

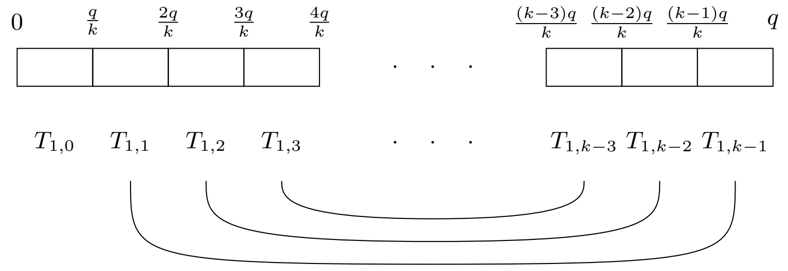

Fig. 5. Interval subdivision and pairing of tables $T_{1,0}, \ldots, T_{1,k-1}$.

Revising the previous complexity analysis, tables $T_{1,0}, \ldots, T_{1,k-1}$ now contain about $\frac{N}{k}$ elements each for a total of N numbers and a total build time of N insertions.

T_2 will contain about $\binom{\frac{N}{k}}{2} + \frac{k-1}{2}\left(\frac{N}{k}\right)^2 \approx \frac{N^2}{2k}$ entries. The time complexity of the revised Algorithm 1, the time taken to build tables T_1 and T_2, is $N + \frac{N^2}{2k}$. We now expect to find a collision after about

$$\frac{N^2}{2k} = \sqrt{\frac{2q\alpha}{k}} \tag{5}$$

insertions into T_2, where we continue to use $\alpha = 9.22$ as before. Minimizing the time complexity we get the revised conditions

$$\begin{cases} N = \frac{N^2}{2k} \\ \frac{N^2}{2k} = \sqrt{\frac{2q\alpha}{k}} \end{cases} \quad \Leftrightarrow \quad \begin{cases} N = 2k \\ k = \left(\frac{q\alpha}{2}\right)^{\frac{1}{3}}, \end{cases}$$

which gives us $k = 2^{54.2}$ and $N = 2^{55.2}$ for a total time complexity of $2N = 2^{56.2}$ for finding a $(2+2)$-relation for the FCSR used in F-FCSR-H v3.

It is possible to double the number of pairs by using, say, $T_{1,0}$ and $T_{1,1}$, and $T_{1,0}$ and $T_{1,k-1}$ together, and so on. Half of the pairs will be $\frac{k}{2}$-small in these cases. However, there is a small additional cost involved and we did not find that these additional pairs affect the time complexity in a positive way.

Generalizing to $(n+n)$-relations, we need to use the entries in tables $T_{1,0}, \ldots, T_{1,k-1}$ to form $\frac{k}{n}$-small n-tuples. This can be done by first choosing entries from any $n-1$ tables $T_{1,0}, \ldots, T_{1,k-1}$, and then choosing the last table so that the sum ends up in $\left[0, \frac{nq}{k}\right)$ (modular interval summation). The new conditions become

$$\begin{cases} N = \frac{k^{n-1}}{n!}\left(\frac{N}{k}\right)^n \\ \frac{k^{n-1}}{n!}\left(\frac{N}{k}\right)^n = \sqrt{\frac{nq\alpha}{k}} \end{cases} \quad \Leftrightarrow \quad \begin{cases} N = (n!k)^{\frac{1}{n-1}} \\ k = \left(\frac{(nq\alpha)^{n-1}}{(n!)^2}\right)^{\frac{1}{n+1}}. \end{cases}$$

For F-FCSR-H v3, the corresponding complexities are given in Table 1.

In Table 2 we also summarize the off- and online complexities for the F-FCSR-H v3 distinguishers for various relation types developed this far.

3.4 Dependency between linear relations

Consider once more the linear dependency condition of linear relations that was noted in Section 3.2.

Table 1. Time complexity for finding an $(n + n)$-relation for F-FCSR-H v3.

relation type	k	N	time
$(2 + 2)$	$2^{54.2}$	$2^{55.2}$	$2^{56.2}$
$(3 + 3)$	$2^{81.3}$	$2^{42.0}$	$2^{43.0}$
$(4 + 4)$	$2^{97.5}$	$2^{34.1}$	$2^{35.1}$

Table 2. F-FCSR-H v3 distinguisher complexities.

relation type	offline	online
$(2 + 2)$	$2^{56.2}$	$2^{37.2}$
$(3 + 3)$	$2^{43.0}$	$2^{63.6}$
$(4 + 4)$	$2^{35.1}$	$2^{89.7}$

Definition 5. *The linear relations $\mathcal{R}_q(w, x; y, z)$ and $\mathcal{R}_q(w', x'; y', z')$ are linearly dependent if and only if*

$$w - w' = x - x' = y - y' = z - z'.$$

In Algorithm 1, table T_2 is used to store k-small pairs $\langle w, x \rangle$, and pairs are continuously added to this table until a collision is found. When two pairs $\langle w, x \rangle$ and $\langle y, z \rangle$ finally do collide, when their sums are equal, we have obtained a linear relation $\mathcal{R}_q(w, x; y, z)$. For a relation $\mathcal{R}_q(w, x; y, z)$ we have $0 \leq w, x, y, z < N$, so any pair of linear relations $\mathcal{R}_q(w, x; y, z)$ and $\mathcal{R}_q(w', x'; y', z')$ are linearly dependent with a probability of about $\frac{1}{N^3}$. Thus, if we use Algorithm 1 to produce a set of L linear relations, by expected value we should find that about $\frac{L^2}{2N^3}$ of these linear relations are redundant. For the values of N that we consider in our applications, it is hard to see how this dependency could come into play, even for a fast correlation attack where one would need to produce many relations.

However, in order to ensure that this dependency does not deplete the search space of Algorithm 1, we also consider the search procedure itself. Table T_2 contains about $\frac{N^2}{2k}$ pairs, so the number of quadruples (pairs of pairs) that we test is about $\binom{\frac{N^2}{2k}}{2} \approx \frac{N^4}{8k^2}$. The linear dependency condition in Definition 5 applies to quadruples as well, in the sense that testing a quadruple that is linearly dependent on a previously tested quadruple is redundant work (searchspace redundancy). Therefore, among $\frac{N^4}{8k^2}$

quadruples we will find that at most

$$\binom{\frac{N^4}{8k^2}}{2}\frac{1}{N^3} \approx \frac{N^5}{2^7k^4}$$

of them are linearly dependent, so that the fraction of linearly dependent quadruples is no more than $\frac{N}{16k^2}$. For the values of N and k that we encounter in our applications (see Table 1), we can again and finally conclude that the dependency issue is a non-issue.

4 Simulations

We have run a set of simulations for verifying the validity of Proposition 1. The guidelines in [2] were followed for random generation of F-FCSRs for these simulations. For each test, a new ring FCSRs of size n and a linear filter with m input bits was used. Algorithm 1 and the modular interval summation technique presented in Section 3.3 were employed to exhibit a $(2+2)$-relation. We randomly selected 100 keystream subsequences $\underline{b}$ of sufficient length and calculated their bias according to Equation (3).

To verify Proposition 1, we ran 10000 tests with parameters $20 \leq n \leq 30$ and $2 \leq m \leq 4$. We found Equation (4) to hold for all subsequences $\underline{b}$ in all tests. The results of these tests can also be seen as an indication that the triplet sequence condition over $\Omega_3(q)$ in Theorems 2 and 3 is reasonable.

We also ran 100 tests for bigger F-FCSRs, $n = 40$ and $m = 4$, with the same result. Example 1 shows one of the F-FCSR ciphers in detail.

Example 1. F-FCSR cipher based on a 40-bit ring FCSR with connection integer $q = 1111855758899$ and transition matrix $T = (t_{i,j})_{0\leq i,j<40}$ with

$$t_{i,j} = \begin{cases} 1, & (i,j) \in S \textit{ or } j \equiv i+1 \bmod 40, \\ 0, & otherwise \ , \end{cases}$$

where S is the set of pairs given in Table 3.

The linear filter xors the values of the four main register cells m_i with $i \in \{0, 5, 9, 17\}$. The $(2+2)$-relation $\mathcal{R}_q(0, 22802; 2166, 27778)$ with bias $\frac{1}{3}$ was found. Initializing the main memory and carry registers with random bits, we randomly chose 100 subsequences of length 200000 for bias calculation.

We also performed several negative tests using the trials above, but randomly altered one or more of the parameters in the $(2+2)$-relation obtained from Algorithm 1. As expected, in this case, only some of the selected keystream subsequences satisfied Equation (4).

Table 3. The set S of nontrivial connections in the transition matrix of a 40-bit F-FCSR.

$(0,6)$	$(1,29)$	$(2,4)$	$(3,15)$	$(5,20)$	$(7,21)$
$(9,37)$	$(10,19)$	$(12,36)$	$(13,34)$	$(15,7)$	$(17,26)$
$(18,17)$	$(20,13)$	$(21,18)$	$(23,39)$	$(25,31)$	$(26,10)$
$(34,11)$	$(35,12)$	$(36,30)$			

Beyond this, we also performed simulations to verify our complexity estimation of Algorithm 1 with modular interval summation. More than 10000 experiments with prime parameters $2^{35} \leq q \leq 2^{40}$ were conducted, and the results suggest that the average value of N is in fact somewhat lower than the estimation.

5 Applications to ring FCSRs, F-FCSRs, FCSR combiners and F-FCSR-H v3

Algorithm 1 is applicable to *all* ring FCSRs, so the underlying FCSRs of all linearly filtered FCSRs and FCSR combiners may be targeted. We now focus specifically on distinguishing attacks on F-FCSR-H v3.

A distinguisher is a decision mechanism that decides if a given sequence is the output of a cipher or if it is some truly random sequence. The decision mechanism takes a sequence as input and outputs either ”CIPHER” or ”RANDOM”.

The F-FCSR-H v3 stream cipher uses 80-bit keys and a ring FCSR with a connection integer q with $\log_2 q < 160.26$. Eight bits are output for every clocking, and these bits are derived using eight separate output subfilters that xor a number of register cells. Six of the filters xor ten bits, while the remaining two xor eleven.

From Proposition 1 we have

$$\Pr\left[\varepsilon^q_{\underline{b}} \geq \left(\frac{1}{3}\right)^m - \frac{4.6802}{\sqrt{k}}\right] \approx 0.9999.$$

A common rule of thumb (see [8]) is that approximately $(\varepsilon^q_{\underline{b}})^{-2}$ samples are needed to distinguish a given sequence from a uniform distribution, which means we need

$$k \leq \left(\left(\frac{1}{3}\right)^m - \frac{4.6802}{\sqrt{k}}\right)^{-2}$$

keystream bits for the distinguishing attack, implying that

$$k \leq 32.2647 \times 3^{2m},$$

where k is the expected number of required keystream bits. In [14] it is claimed that the required length of keystream for the $(3+3)$-relation case is simply 3^{2m}, which is not entirely accurate since that disregards the factor of 32.2647.

Getting back to F-FCSR-H v3, making use of the six smaller subfilters only (out of eight), we can apply the above formula with $m = 10$ to get

$$k \leq \frac{8}{6} \times 32.2647 \times 3^{2m} = 2^{37.2}.$$

As stated before, the offline time complexity is $2^{56.2}$. This is the time required for finding the $(2+2)$-relation. As for keystream requirements, we need four separate chunks of $2^{37.2}$ keystream bits each for a total of $2^{39.2}$ keystream bits to perform the attack in time $2^{37.2}$. However, these four chunks can be quite far apart due to the width

$$\max(w, x, y, z) - \min(w, x, y, z)$$

of the $(2+2)$-relation $\mathcal{R}_q(w, x; y, z)$. Thus, we can perform the attack in time $2^{37.2}$, but we still need to observe $2^{56.2}$ keystream bits. The keystream requirement is therefore $2^{56.2}$ bits, while we can manage with only $2^{39.2}$ bits of storage.

The relation width can be made smaller by using higher-order $(n+n)$-relations, but the bias quickly diminishes as n gets large. Also, the fact that we are employing a generalized birthday attack contributes to this width. One would expect to find shorter relations using a standard birthday attack, but the width is then improved only at the expense of an increased time complexity.

The F-FCSR-H v3 distinguisher is made explicit in Algorithm 2, where the last if-clause may use any efficient statistical decision mechanism, for example the Neyman-Pearson lemma in [4].

6 Conclusions

We have presented a generalized birthday algorithm that can be used to find a specific set of linear relations. The algorithm was applied to produce different distinguishing attacks on F-FCSR-H v3. The one we advocate uses a $(2+2)$-relation and has off- and online time complexities

Algorithm 2 – F-FCSR-H v3 Distinguisher

Input: Keystream bits z_t for $t \geq 0$.
Output: The classification "CIPHER" or "RANDOM".

```
/* offline */
Use Algorithm 1 to find a $(2+2)$-relation $\mathcal{R}_q(w, x; y, z)$;
/* online */
$c = 0$; /* relation counter */
for ($i = 0$; $i < 2^{37.2}$; $i$++) {
    if ($i \bmod 8 \geq 2$) {/* if small subfilter was used */
        if ($z_{i+w} \oplus z_{i+x} \oplus z_{i+y} \oplus z_{i+z} = 0$) {
            $c$++;
        }
    }
}
if ($c$ significantly deviates from $\frac{1}{2} \times \frac{6}{8} \times 2^{37.2}$) {
    return "CIPHER";
}
return "RANDOM";
```

$2^{56.2}$ and $2^{37.2}$, respectively. These are the first successful attacks on F-FCSR-H v3, breaking the exhaustive search complexity bound. While this application was very specific, the presented algorithm itself is very general and applicable to all ring FCSRs. In particular, it can be applied to the underlying FCSRs of all linearly filtered FCSRs and FCSR combiners.

Acknowledgements

This work was supported in part by the Swedish Research Council (Vetenskapsrådet) under grant 621-2006-5249, the National Natural Science Foundations of China under grant 61170208, the Shanghai Key Program of Basic Research under grant 12JC1401400, the Shanghai Shuguang Project under grant 10SG01, the National Defense Pre-Research Project under grant 2012004.

References

1. Arnault, F., Berger, T., Lauradoux, C.: Update on F-FCSR stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/025 (2006). Available at: `http://www.ecrypt.eu.org/stream/p3ciphers/ffcsr/ffcsr_p3.pdf`.
2. Arnault, F., Berger, T., Lauradoux, C., Minier, M., Pousse, B.: A New Approach for F-FCSRs. In: M.J.J. Jr., V. Rijmen, R. Safavi-Naini (eds.) Selected Areas in Cryptography—SAC 2009, *Lecture Notes in Computer Science*, vol. 5867, pp. 433–448. Springer-Verlag (2009). DOI: 10.1007/978-3-642-05445-7_27.

3. Arnault, F., Berger, T., Pousse, B.: A matrix approach for FCSR automata. Cryptography and Communications **3**, 109–139 (2011). DOI: 10.1007/s12095-010-0041-z.
4. Cover, T., Thomas, J.A.: Elements of Information Theory. Wiley series in Telecommunication. Wiley (1991). Available at: `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471241954.html`.
5. Goresky, M., Klapper, A.: Arithmetic Cross-correlations of FCSR Sequences. IEEE Trans. Info. Theory **43**, 1342–1346 (1997). Available at: `http://www.math.ias.edu/~goresky/pdf/arith.jour.pdf`.
6. Goresky, M., Klapper, A.: Fibonacci and Galois Representations of Feedback–With–Carry Shift Registers. IEEE Transactions on Information Theory **48**(11), 2826–2836 (2002). DOI: 10.1109/TIT.2002.804048.
7. Hell, M., Johansson, T.: Breaking the Stream Ciphers F-FCSR-H and F-FCSR-16 in Real Time. Journal of Cryptology, Online First$^{\mathrm{TM}}$(2009).
8. Hell, M., Johansson, T., Brynielsson, L.: An overview of distinguishing attacks on stream ciphers. Cryptography and Communications **1**(1), 71–94 (2009). DOI: 10.1007/s12095-008-0006-7.
9. Hogg, R.V., Tanis, E.A.: Probability and statistical inference, 4 edn. MacMillan Publishing Co. (1993).
10. Klapper, A., Goresky, M.: 2-adic Shift Registers. In: R.J. Anderson (ed.) Fast Software Encryption'93, *Lecture Notes in Computer Science*, vol. 809, pp. 174–178. Springer-Verlag (1994). DOI: 10.1007/3-540-58108-1_21.
11. Klapper, A., Goresky, M.: Feedback shift registers, 2-adic span, and combiners with memory. Journal of Cryptology **10**(2), 111–147 (1997). DOI: 10.1007/s001459900024.
12. Matsui, M.: Linear cryptanalysis method for DES cipher. In: T. Helleseth (ed.) Advances in Cryptology—EUROCRYPT'93, *Lecture Notes in Computer Science*, vol. 765, pp. 386–397. Springer-Verlag (1994). DOI: 10.1007/3-540-48285-7_33.
13. Pagh, R., Rodler, F.F.: Cuckoo hashing. In: Journal of Algorithms, vol. 51, pp. 122–144. Academic Press, Inc., Duluth, MN, USA (2004). DOI: 10.1016/j.jalgor.2003.12.002.
14. Tian, T., Qi, W.F.: Linearity properties of binary FCSR sequences. Designs, Codes and Cryptography **52**, 249–262 (2009). DOI: 10.1007/s10623-009-9280-4.
15. Wagner, D.: A generalized birthday problem. In: M. Yung (ed.) Advances in Cryptology—CRYPTO 2002, *Lecture Notes in Computer Science*, vol. 2442, pp. 288–304. Springer-Verlag (2002). DOI: 10.1007/3-540-45708-9_19.
16. Wikipedia: Birthday problem — Wikipedia, the free encyclopedia. Available at: `http://en.wikipedia.org/wiki/Birthday_problem`. Available at: `http://en.wikipedia.org/wiki/Birthday_problem`. Last accessed on February 17, 2012.