2.7 Nonlinear Feedback Shift Registers

As another example of the general prediction method we consider arbitrary, not necessarily linear, feedback shift registers as illustrated in Figure 2.3.

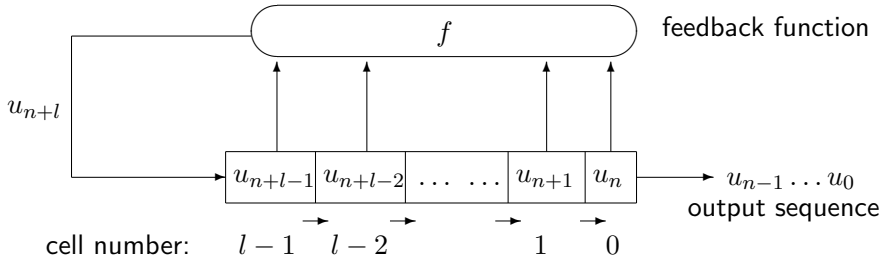

Figure 2.3: A feedback shift register (FSR) of length l

Here the feedback function is an arbitrary Boolean function $f\colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2$ whose algebraic normal form is a polynomial

$$f(y_1, \ldots, y_l) = \sum_{I \subseteq \{1,\ldots,l\}} a_I y^I \qquad \text{with } y^I = \prod_{j \in I} y_j.$$

We want to apply the prediction method with $R = X = \mathbb{F}_2$, $h = l$, $Z = \mathbb{F}_2^{2^l}$. For $i \geq l$

$$\Phi^{(i)}\colon \mathbb{F}_2^i \longrightarrow Z$$

is given by

$$z_i := \Phi^{(i)}(x_1, \ldots, x_i) = (y^I)_{I \subseteq \{1,\ldots,l\}} \qquad \text{with } y = (x_{i-l+1}, \ldots, x_i).$$

And finally we set

$$\alpha\colon Z \longrightarrow X, \quad \alpha((t_I)_{I \subseteq \{1,\ldots,l\}}) = \sum a_I t_I.$$

First we treat two concrete examples:

Examples

1. $l = 2$, $f = T_1 T_2 + T_2$. From the initial values $u_0 = 1$, $u_1 = 0$ we generate the sequence (manually or by Sage example 2.2)

   $$u_0 = 1,\ u_1 = 0,\ u_2 = 1,\ u_3 = 0,\ \ldots$$

   (that evidently has period 2). We have

   $$Z = \mathbb{F}_2^4, \qquad z_n = \begin{pmatrix} u_{n-1}u_{n-2} \\ u_{n-1} \\ u_{n-2} \\ 1 \end{pmatrix},$$

$$z_2 = \begin{pmatrix}0\\0\\1\\1\end{pmatrix}, \quad z_3 = \begin{pmatrix}0\\1\\0\\1\end{pmatrix}, \quad z_4 = \begin{pmatrix}0\\0\\1\\1\end{pmatrix} = z_2, \quad \ldots$$

From this the cryptanalyst recognizes the linear recursion

$$z_n = z_{n-2} = 0 \cdot z_{n-1} + 1 \cdot z_{n-2} \quad \text{for } n \geq 4.$$

She even recognizes the period, and correctly predicts

$$u_n = 0 \cdot u_{n-1} + 1 \cdot u_{n-2} = u_{n-2} \quad \text{for } n \geq 4.$$

Note that the very same sequence can be generated by a *linear* FSR of length 2. The analysis used the elements u_0, u_1, u_2, u_3.

2. $l = 3$, $f = T_1T_3 + T_2$. From the initial values $u_0 = 0$, $u_1 = 1$, $u_2 = 1$ we generate the elements (manually or by Sage example 2.3)

$$u_3 = 1,\ u_4 = 0,\ u_5 = 1,\ u_6 = 1,\ u_7 = 1,\ u_8 = 0,\ u_9 = 1,\ \ldots$$

of the output sequence. We have

$$Z = \mathbb{F}_2^8, \qquad z_n = \begin{pmatrix} u_{n-1}u_{n-2}u_{n-3}\\ u_{n-1}u_{n-2}\\ u_{n-1}u_{n-3}\\ u_{n-2}u_{n-3}\\ u_{n-1}\\ u_{n-2}\\ u_{n-3}\\ 1 \end{pmatrix},$$

$$z_3 = \begin{pmatrix}0\\1\\0\\0\\1\\1\\0\\1\end{pmatrix}, \quad z_4 = \begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}, \quad z_5 = \begin{pmatrix}0\\0\\0\\1\\0\\1\\1\\1\end{pmatrix}, \quad z_6 = \begin{pmatrix}0\\0\\1\\0\\1\\0\\1\\1\end{pmatrix}, \quad z_7 = \begin{pmatrix}0\\1\\0\\0\\1\\1\\0\\1\end{pmatrix} = z_3,$$

and so on. Hence the supposed linear recursion is

$$z_n = z_{n-4} \quad \text{for } n \geq 4,$$

again it reflects the periodicity. We get the correct prediction formula

$$u_n = u_{n-4} \quad \text{for } n \geq 4.$$

We needed the elements from u_0 to u_6; and again we found an "equivalent" LFSR, this time of length 4.

Sage Example 2.2 $f_1 = T_1T_2 + T_2$—monomials with exponent pairs `[1,1]` $\hat{=} 3$ and `[0,1]` $\hat{=} 1$, hence ANF bitblock `[0,1,0,1]`

```
f1 = BoolF([0,1,0,1],method="ANF")
y = f1.getTT(); y
[0, 1, 0, 0]
start = [0,1]
seq = fsr(f1,start,10); seq
[1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
```

Sage Example 2.3 $f_2 = T_1T_3 + T_2$—monomials with exponent triples `[1,0,1]` $\hat{=} 5$ and `[0,1,0]` $\hat{=} 2$, hence ANF bitblock `[0,0,1,0,0,1,0,0]`

```
f2 = BoolF([0,0,1,0,0,1,0,0],method="ANF")
y = f2.getTT(); y
[0, 1, 0, 0]
start = [1,1,0]
seq = fsr(f2,start,10); seq
[0, 1, 1, 1, 0, 1, 1, 1, 0, 1]
```

Since the dimension of Z grows exponentially with the register length the prediction algorithm reaches its limits soon. In the worst case the stationary state of the ascending chain of subspaces—and the needed linear relation—occurs only after 2^l steps. This observation would make shift registers up to a length of about 32 predictable with manageable cost using linear algebra in a binary vector space of dimension 2^{32}.

However in the examples we observed that the linear relation we found is nothing other than the formula for the final periodic repetition. This was not a fortunate coincidence but is a general phenomen that has an easy proof. For details see the paper [6]. Hence instead of solving large systems of linear equations we can apply an algorithm for period search that needs significantly less resources. This approach enables a realistic attack on shift registers of lengths up to about 80.

From a general point of view there is another objection against using arbitrary FSRs: The feedback function f depends on 2^l parameters. To have f efficiently computable and to deal with a manageable key space we have to restrict the choice of f, say by forcing "almost all" coefficients a_I in the ANF of the "admissible" feedback functions f to 0. Thus we specify a "small" set $\mathcal{M} \subseteq \mathcal{P}(\{1, \dots l\})$ a priori, and use only functions f whose ANF

$$f(x_1, \dots, x_l) = \sum_{I \in \mathcal{P}(\{1,\dots l\})} a_I x^I$$

has coefficients $a_I = 0$ for $I \notin \mathcal{M}$. Then the key space has size $2^{\#\mathcal{M}}$. However the coice of $\mathcal{M}$ is part of the encryption algorithm—in particular for a hardware FSR—, not a part of the key. KERCKHOFFS' principle warns us that the enemy will learn about $\mathcal{M}$ sooner or later. In the model of Figure 2.1 we treat the a priori "monomial supply" $\mathcal{M}$ as public parameter, and the concrete "monomial selection" I as secret parameter.

The necessity of choosing an efficiently computable feedback function and a manageable key space enforces restrictions that make the prediction method efficient too. Expressed in a somewhat sloppy way:

Proposition 9 *Each bit sequence that is generated by an FSR with efficiently computable feedback function is efficiently predictable.*

Our treatment of this problem was quite coarse. To derive mathematically correct statements there are two approaches:

1. Directly estimate the circuit complexity of the prediction algorithm by the circuit complexity of the feedback function.

2. Consider families of Boolean functions—that define families of FSRs—whose complexity grows polynomially with the register length, and show that the costs of the corresponding prediction procedures also grow at most polynomially.

For a comprehensive treatment see the cited paper [6].

We conclude that FSRs, no matter whether linear or nonlinear, are unsuited for generating pseudorandom sequences of cryptographic value—at least if naively applied. The method of BOYAR/KRAWCZYK breaks also nonlinear FSRs in realistic scenarios. And the result of BETH/DAI in Section 3.6 will open another promising way of predicting an FSR using the BERLEKAMP/MASSEY algorithm, see Section 3.3.