Linear feedback shift registers

Here is another basic idea to generate pseudorandom numbers:

(linear feedback shift register (LFSR) Let ℓ and $c_1, c_2, \ldots, c_\ell$ be chosen parameters. From the seed $(x_1, x_2, \ldots, x_\ell)$, where each x_i is one bit, we produce the sequence

$$x_{n+\ell} \equiv c_1 x_{n+\ell-1} + c_2 x_{n+\ell-2} + \ldots + c_\ell x_n \pmod 2.$$

This method is particularly easy to implement in hardware (see Example 55), and hence suited for applications that value speed over security (think, for instance, encrypted television).

Example 54. Which sequence is generated by the LFSR $x_{n+2} \equiv x_{n+1} + x_n \pmod 2$, starting with the seed $(x_1, x_2) = (0, 1)$?

Solution. $(x_1, x_2, x_3, \ldots) = (0, 1, 1, 0, 1, 1, \ldots)$ has period 3.

Note. Observe that the two previous values determine the state, so there is $2^2 = 4$ states of the LFSR. The state $(0, 0)$ is special (it generates the zero sequence $(0, 0, 0, 0, \ldots)$), so there is 3 other states. Hence, it is clear that the generated sequence has to repeat after at most 3 terms.

Comment. Of course, if we don't reduce modulo 2, then the sequence $x_{n+2} = x_{n+1} + x_n$ generates the Fibonacci numbers $0, 1, 1, 2, 3, 5, 8, 13, \ldots$

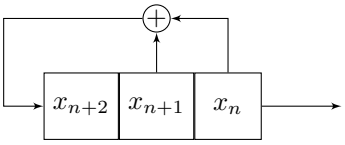
Example 55. Which sequence is generated by the LFSR $x_{n+3} \equiv x_{n+1} + x_n \pmod 2$, starting with the seed $(x_1, x_2, x_3) = (0, 0, 1)$? What is the period?

[Let us first note that the LFSR has $2^3 = 8$ states. Since the state $(0, 0, 0)$ remains zero forever, 7 states remain. This means that the generated sequence must be periodic, with period at most 7.]

Solution. $(x_1, x_2, x_3, \ldots) = (0, 0, 1, 0, 1, 1, 1, 0, 0, 1, \ldots)$ has period 7.

Again, this is not surprising: 3 previous values determine the state, so there is $2^3 = 8$ states. The state $(0, 0, 0)$ is special, so there is 7 other states.

Note that this LFSR can be implemented in hardware using three registers (labeled x_n, x_{n+1}, x_{n+2} in the sketch to the right). During each cycle, the value of x_n is read off as the next value produced by the LFSR.



Note. In the part $0, 0, 1, 0, 1, 1, 1$ that repeats, the bit 1 occurs more frequently than 0.

The reason for that is that the special state $(0, 0, 0)$ cannot not appear.

For the same reason, the bit 1 will always occur slightly more frequently than 0 in LFSRs. However, this becomes negligible if the period is huge, like $2^{31} - 1$ in Example 56.

Example 56. The recurrence $x_{n+31} \equiv x_{n+28} + x_n \pmod 2$, with a nonzero seed, generates a sequence that has period $2^{31} - 1$.

Note that this is the maximal possible period: this LFSR has 2^{31} states. Again, the state $(0, 0, \ldots, 0)$ is special (the entire sequence will be zero), so that there is $2^{31} - 1$ other states. This means that the terms must be periodic with period at most $2^{31} - 1$.

Comment. glibc (the second implementation) essentially uses this LFSR.

Advanced comment. One can show that, if the characteristic polynomial $f(T) = x^\ell + c_1 x^{\ell-1} + c_2 x^{\ell-2} + \ldots + c_\ell$ is irreducible modulo 2, then the period divides $2^\ell - 1$. Here, $f(T) = T^{31} + T^{28} + 1$ is irreducible modulo 2, so that the period divides $2^{31} - 1$. However, $2^{31} - 1$ is a prime, so that the period must be exactly $2^{31} - 1$.

Armin Straub
straub@southalabama.edu

Example 57. Eve intercepts the ciphertext $c = (1111\ 1011\ 0000)_2$ from Alice to Bob. She knows that the plaintext begins with $m = (1100\ 0\ldots)_2$. Eve thinks a stream cipher using a LFSR with $x_{n+3} \equiv x_{n+2} + x_n \pmod 2$ was used. If that's the case, what is the plaintext?

Solution. The initial piece of the keystream is $\mathrm{PRG} = m \oplus c = (1100\ 0\ldots)_2 \oplus (1111\ 1\ldots)_2 = (0011\ 1\ldots)_2$. Each x_n is a single bit, and we have $x_1 \equiv 0$, $x_2 \equiv 0$, $x_3 \equiv 1$. The given LFSR produces $x_4 \equiv x_3 + x_1 \equiv 1$, $x_5 \equiv x_4 + x_2 \equiv 1$, $x_6 \equiv 0$, $x_7 \equiv 1$, and so on. Continuing, we obtain $\mathrm{PRG} = x_1 x_2 \ldots = (0011\ 1010\ 0111)_2$. Hence, the plaintext would be $m = c \oplus \mathrm{PRG} = (1111\ 1011\ 0000)_2 \oplus (0011\ 1010\ 0111)_2 = (1100\ 0001\ 0111)_2$.

We have seen two simple examples of PRGs so far:

- linear congruential generators $x_{n+1} \equiv a x_n + b \pmod m$
- LFSRs $x_{n+\ell} \equiv c_1 x_{n+\ell-1} + c_2 x_{n+\ell-2} + \ldots + c_\ell x_n \pmod 2$

Of course, we could also combine LFSRs and linear congruential generators (i.e. look at recurrences like for LFSRs but modulo any parameter m).

However, much of the appeal of an LFSR comes from its extremely simple hardware realization, as the sketch in Example 55 indicates.

Example 58. (extra) One can also consider nonlinear recurrences (it mitigates some issues). Our book mentions $x_{n+3} \equiv x_{n+2} x_n + x_{n+1} \pmod 2$. Generate some numbers.

Solution. For instance, using the seed $0, 0, 1$, we generate $\overbrace{0, 0, 1}^{\text{seed}}, 0, 1, 1, 1, 0, 1, \ldots$ which now repeats (with period 4) because the state $1, 0, 1$ appeared before. Observe that the generated sequences is only what is called eventually periodic (it is not strictly periodic because $0, 0, 1$ never shows up again).

A PRG is **predictable** if, given the stream it outputs (but not the seed), one can with some precision predict what the next bit will be (i.e. do better than just guessing the next bit).

In other words: the bits generated by the PRG must be indistinguishable from truly random bits, even in the eyes of someone who knows everything about the PRG except the seed.

The PRGs we discussed so far are entirely predictable because the state of the PRGs is part of the random stream they output.

For instance, for a given LFSR, it is enough to know any ℓ consecutive outputs $x_n, x_{n+1}, \ldots, x_{n+\ell-1}$ in order to predict all future output.

A popular way to reduce predictability is to combine several LFSRs:

Example 59. Let us consider a baby version of CSS (discussed next class). Our PRG uses the LFSR $x_{n+3} \equiv x_{n+1} + x_n \pmod 2$ as well as the LFSR $x_{n+4} \equiv x_{n+2} + x_n \pmod 2$. The output of the PRG is the output of these two LFSRs added with carry.

Adding with carry just means that we are adding bits modulo 2 but add an extra 1 to the next bits if the sum exceeded 1. This is the same as interpreting the output of each LFSR as the binary representation of a (huge) number, then adding these two numbers, and outputting the binary representation of the sum.

If we use $(0,0,1)$ as the seed for LFSR-1, and $(0,1,0,1)$ for LFSR-2, what are the first 10 bits output by our PRG?

Solution. With seed $0,0,1$ LSFR-1 produces $0,1,1,1,0,0,1,0,1,1,\ldots$

With seed $0,1,0,1$ LSFR-2 produces $0,0,0,1,0,1,0,0,0,1,\ldots$

We now add these two:

	0	1	1	1	0	0	1	0	1	1	$\cdots$
$+$	0	0	0	1	0	1	0	0	0	1	$\cdots$
carry					1						1
	0	1	1	0	1	1	1	0	1	0	$\cdots$

Hence, the output of our PRG is $0,1,1,0,1,1,1,0,1,0,\ldots$.

Important comment. Make sure you realize in which way this CSS PRG is much less predictable than a single LFSR! A single LFSR with ℓ registers is completely predictable since knowing ℓ bits of output (determines the state of the LFSR and) allows us to predict all future output. On the other hand, it is not so simple to deduce the state of the CSS PRG from the output. For instance, the initial $(0,1,\ldots)$ output could have been generated as $(0,0,\ldots)+(0,1,\ldots)$ or $(0,1,\ldots)+(0,0,\ldots)$ or $(1,0,\ldots)+(1,0,\ldots)$ or $(1,1,\ldots)+(1,1,\ldots)$.

[In this case, we actually don't learn anything about the registers of each individual LFSR. However, we do learn how their values have to match up. That's the correlation that is exploited in correlation attacks, like the one described last class for the actual CSS scheme.]