Review. We have already seen two examples of PRGs:

- linear congruential generators $x_{n+1} = a x_n + b \pmod m$
- 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 the next example indicates.

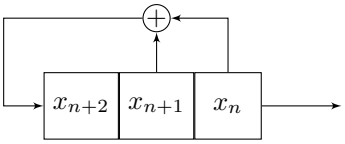
Example 67. Which sequence is generated by the LFSR $x_{n+3} \equiv x_{n+1} + x_n \pmod 2$, starting with the seed $(x_1, x_2, x_3) = (0, 0, 1)$? What is the period?

[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 66.

Example 68. In each case, determine if the stream could have been produced by the LFSR $x_{n+5} \equiv x_{n+2} + x_n \pmod 2$. If yes, predict the next three terms.

(STREAM-1) $\ldots, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, \ldots$ (STREAM-2) $\ldots, 1, 1, 0, 0, 0, 1, 1, 0, 1, \ldots$

Solution. Using the LFSR, the values $1, 0, 0, 1, 1$ are followed by $1, 1, 1, 0, \ldots$ Hence, STREAM-1 was not produced by this LFSR.

On the other hand, using the LFSR, the values $1, 1, 0, 0, 0$ are followed by $1, 1, 0, 1, 1, 1, 0, \ldots$ Hence, it is possible that STREAM-2 was produced by the LFSR (for a random stream, the chance is only $1/2^4 = 6.25\%$ that 4 bits matched up). We predict that the next values are $1, 1, 0, \ldots$

Comment. This observation is crucial for the attack on CSS described below.

Example 69. (homework) 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 output (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.

Armin Straub
straub@southalabama.edu

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 70. (CSS) The CSS (content scramble system) is based on 2 LFSRs and used for the encryption of DVDs. Let us indicate how to break it.

CSS was introduced in 1996 and first compromised in 1999. One big issue is that its key size is 40 bits. Since $2^{40} \approx 1.1 \cdot 10^{12}$ is small by modern standards, even a direct brute-force attack in time 2^{40} is possible.

However, we will see below that poor design makes it possible to attack it in time 2^{16}.

Historic comment. 40 bits was the maximum allowed by US export limitations at the time.

https://en.wikipedia.org/wiki/Export_of_cryptography_from_the_United_States

CSS PRG combines one 17-bit LFSR and one 25-bit LFSR. The bits output by the CSS PRG are the sum of the bits output by the two LFSRs (this is the usual sum, including carries).

The 40 bit key is used to seed the LFSRs (the 4th bit of each seed is "1", so we need $16+24=40$ other bits).

Here's how we break CSS in time 2^{16}:

- If a movie is encrypted using MPEG then we know the first few, say x (6-20), bytes of the plaintext.
- As in Example 64, this allows us to compute the first x bytes of the CSS keystream.
- We now go through all 2^{16} possibilities for the seed of LFSR-17. For each seed:
  - We generate x bytes using LFSR-17 and subtract these from the known CSS keystream.
  - This would be the output of LFSR-25. As in Example 68, we can actually easily tell if such an output could have been produced by LFSR-25. If yes, then we found (most likely) the correct seed of LFSR-17 and now also have the correct state of LFSR-25.

This kind of attack is known as a correlation attack.

https://en.wikipedia.org/wiki/Correlation_attack

Comment. Similar combinations of LFSRs are used in GSM encryption (A5/1,2, 3 LFSRs); Bluetooth (E0, 4 LFSRs). All of these are broken; so, of course, they shouldn't be used. However, it is difficult to update things implemented in hardware...