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).

Example 59. Suppose we have two PRGs that output bits. The first repeats after 14 bits, the second after 18 bits. After how many bits do they repeat simultaneously?

What if the two PRGs repeat after 13 and 17 bits instead?

Solution. Note that the first PRG again repeats after 28 bits, after 42 bits and, in general after $14m$ bits where m is any positive integer. Likewise, the second PRG repeats after $18m$ bits where m is any positive integer. Therefore, both PRGs repeat simultaneously after $\operatorname{lcm}(14, 18) = \frac{14 \cdot 18}{2} = 126$ bits.

Review. Here, lcm is the **least common multiple**. We can always compute the lcm through the Euclidean algorithm by using $\operatorname{lcm}(a, b) = \frac{a \cdot b}{\gcd(a, b)}$.

If the two PRGs repeat after 13 and 17 bits instead, then they repeat simultaneously after $\operatorname{lcm}(13, 17) = 13 \cdot 17 = 221$ bits.

Comment. Certain cicadas spend more than 99% of their life underground as nymphs and only emerge as adults for 4-6 weeks. Interestingly, this life cycle is highly synchronized: cicadas of one species in a region appear all at once. In 2024, "Brood XIII" and "Brood XIX" co-emerged. These emerge every 17 and 13 years, respectively. Therefore this co-emergence is a rare event that only happens every 221 years (though, the same thing happened 2015 with two different broods).

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

Example 60. (bonus!) Eventually the output of the baby CSS in Example 61 has to repeat (though it need not be perfectly periodic; see Example 58). Once it repeats, what is the period?

Note. The state of the system is determined by $3 + 4 + 1 = 8$ bits (3 bits for LFSR-1, 4 bits for LFSR-2, and 1 bit for the carry). Hence, there are $2^8 = 256$ many states. Since the state with everything 0 is again special, that means that after at most 255 steps our PRG will reach a state it has been in before. At that point, everything will repeat.

(To collect a bonus point, send me an email within the next week with the period and how you found it.)

Armin Straub
straub@southalabama.edu

Combining two LFSRs to get the CSS (content scramble system)

A popular way to reduce predictability is to combine several LFSRs (in a nonlinear fashion):

Example 61. The CSS (content scramble system) is based on 2 LFSRs and used for the encryption of DVDs. Before discussing the actual CSS let us consider a baby version of CSS. 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 next class for the actual CSS scheme.]

Advanced comment. Is the carry important? Yes! Let $a_1, a_2, \ldots$ and $b_1, b_2, \ldots$ be the outputs of LFSR-1 and LFSR-2. Suppose we sum without carry. Then the output is $a_1 + b_1, a_2 + b_2, \ldots$ (with addition mod 2). If Eve assigns variables k_1, k_2, ..., k_7 to the $3 + 4$ seed bits (the key in the stream cipher), then the output of the combined LFSR will be linear in these seven variables (because the a_i and b_i are linear combinations of the k_i). Given just a few more than 7 output bits, a little bit of linear algebra (mod 2) is therefore enough to solve for $k_1, k_2, \ldots, k_7$.

On the other hand, suppose we include the carry. Then the output is $a_1 + b_1, a_2 + b_2 + a_1 b_1, \ldots$ (note how $a_1 b_1$ is $1 \pmod{2}$ precisely if both a_1 and b_1 are $1 \pmod{2}$, which is when we have a carry). This is not linear in the a_i and b_i (and, hence, not linear in the k_i), and we cannot use linear algebra to solve for $k_1, k_2, \ldots, k_7$ as before.

Example 62. 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) $\ldots, 1, 0, 0, 1, 1, 1, 1, 0, 1, \ldots$ (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 in Example 63.