Example 60. (bonus challenge) Eventually the output in Example 59 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.

Example 61. 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 62.

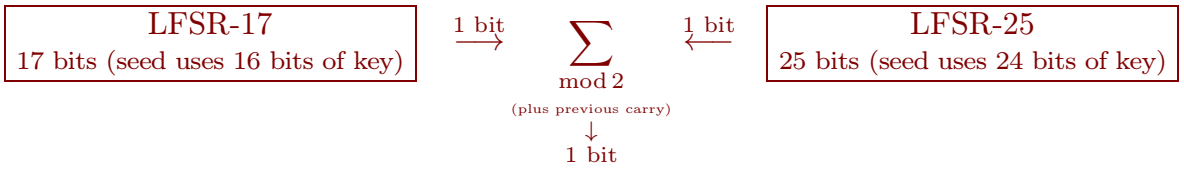
Example 62. (CSS) The CSS (content scramble system) is based on 2 LFSRs and used for the encryption of DVDs. Let us indicate how to break it.

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 53, 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 61, 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...

A sad but important lesson

CSS (and many other examples in recent history) teach us one important lesson:

Do not implement your own ideas for serious crypto!

We will soon see that there exist cryptosystems which are believed to be secure. While this is not proven in any case, we do know that certain of these are in fact secure (if implemented correctly) if and only if a certain important mathematical problem cannot be easily solved.

- So, to crack such a system, one has to solve a mathematical problem that many people care about deeply. If this happens, you will most likely read about it in the (academic) news, and you will have an opportunity to update your system in time (most likely, you'll hear about progress much earlier).
- On the other hand, if you use a cryptosystem that is not well-studied, then it may well happen that an adversary breaks your system and keeps exploiting the security leak without you ever learning about it.

Chinese remainder theorem

Example 63. (warmup)

(a) If $x \equiv 3 \pmod{10}$, what can we say about $x \pmod 5$?

(b) If $x \equiv 3 \pmod 7$, what can we say about $x \pmod 5$?

Solution.

(a) If $x \equiv 3 \pmod{10}$, then $x \equiv 3 \pmod 5$.
[Why?! Because $x \equiv 3 \pmod{10}$ if and only if $x = 3 + 10m$, which modulo 5 reduces to $x \equiv 3 \pmod 5$.]

(b) Absolutely nothing! $x = 3 + 7m$ can be anything modulo 5 (because $7 \equiv 2$ is invertible modulo 5).

Example 64. If $x \equiv 32 \pmod{35}$, then $x \equiv 2 \pmod 5$, $x \equiv 4 \pmod 7$.

Why?! As in the first part of the warmup, if $x \equiv 32 \pmod{35}$, then $x \equiv 32 \pmod 5$ and $x \equiv 32 \pmod 7$.

The Chinese remainder theorem says that this can be reversed! (More next class!)

That is, if $x \equiv 2 \pmod 5$ and $x \equiv 4 \pmod 7$, then the value of x modulo $5 \cdot 7 = 35$ is determined.
[How to find the exact $x \equiv 32 \pmod{35}$ is discussed in the next example.]

Example 65. Solve $x \equiv 2 \pmod 5$, $x \equiv 4 \pmod 7$.

Solution. $x \equiv 2 \cdot 7 \cdot \underbrace{7^{-1}_{\bmod 5}}_{3} + 4 \cdot 5 \cdot \underbrace{5^{-1}_{\bmod 7}}_{3} \equiv 42 + 60 \equiv 32 \pmod{35}$

Important comment. Can you see how we need 5 and 7 to be coprime here?

Brute force solution. Note that, while in principle we can always perform a brute force search, this is not practical for larger problems. Here, if x is a solution, then so is $x + 35$. So we only look for solutions modulo 35.
Since $x \equiv 4 \pmod 7$, the only candidates for solutions are $4, 11, 18, \ldots$ Among these, we find $x = 32$.
[We can also focus on $x \equiv 2 \pmod 5$ and consider the candidates $2, 7, 12, \ldots$, but that is even more work.]