Review.

- A **pseudorandom generator** (PRG) takes a seed x_0 and produces a stream $\mathrm{PRG}(x_0) = x_1 x_2 x_3 \ldots$ of numbers, which should be close to random numbers.

  For cryptographic purposes, these numbers should be indistinguishable from random numbers. Even for somebody who knows everything about the PRG except the seed. (See Example 64.)

- Once we have a PRG, we can use it as a **stream cipher**: Using the key k, we encrypt $E_k(m) = m \oplus \mathrm{PRG}(k)$. [Here, the key stream $g(k)$ is assumed to be in bits.]

  As with the one-time pad, we must never reuse the same keystream!

- To reuse the key, we can use a **nonce**: $E_k(m) = m \oplus \mathrm{PRG}((\mathrm{nonce}, k))$, where the seed is produced by combining the nonce and k (for instance, just concatenating them).

  The nonce is then passed (unencrypted) along with the message.

  To make sure that we never reuse the same keystream, we must never use the same nonce with the same key.

Example 61. Let's use the PRG $x_{n+1} = 5x_n + 3 \pmod 8$ as a stream cipher with the key $k = 4 = (100)_2$. The key is used as the seed x_0 and the keystream is $\mathrm{PRG}(k) = x_1 x_2 \ldots$ (where each x_i is 3 bits). Encrypt the message $m = (101\ 111\ 001)_2$.

Solution. We first use the PRG with seed $x_0 = k$ to produce the keystream $\mathrm{PRG}(k) = 7,\ 6,\ 1,\ \ldots = (111\ 110\ 001\ \ldots)_2$.

We then encrypt and get $c = E_k(m) = m \oplus \mathrm{PRG}(k) = (101\ 111\ 001)_2 \oplus (111\ 110\ 001)_2 = (010\ 001\ 000)_2$.

Decryption. Observe that decryption works in the exact same way:

$D_k(c) = c \oplus \mathrm{PRG}(k) = (010\ 001\ 000)_2 \oplus (111\ 110\ 001)_2 = (101\ 111\ 001)_2$.

Note. The keystream continues as $\mathrm{PRG}(k) = 7, 6, 1, 0, 3, 2, 5, 4, \ldots$ At this point it repeats itself because we obtained the value 4, which was our seed. Since the state of this PRG only depends on the value of x_n, and there is 8 possible values for x_n, the period 8 is the longest possible. The homework from last class gave conditions on the PRG that guarantee that the period is as long as possible.

Example 62. (homework) Similarly, let's use the PRG $x_{n+1} = 9x_n + 5 \pmod{64}$ as a stream cipher with the key $k = 4 = (100)_2$ and nonce $5 = (101)_2$. The key prefixed with the nonce is used as the seed x_0, so that the keystream is $\mathrm{PRG}(\mathrm{nonce}, k) = x_1 x_2 \ldots$ (each x_i is now 6 bits). Encrypt the message $m = (101111\ 001000)_2$.

Solution. The seed is $x_0 = (101\ 100)_2 = 44$.

We first use the PRG with that seed to produce the keystream $\mathrm{PRG}(44) = 17, 30, \ldots = (010001\ 011110\ \ldots)_2$.

Then $c = E_k(m) = m \oplus \mathrm{PRG}(\mathrm{nonce}, k) = (101111\ 001000)_2 \oplus (010001\ 011110)_2 = (111110\ 010110)_2$.

The nonce is transmitted (unencrypted) together with the ciphertext c.

Comment. In some cryptosystems, the nonce is incremented each time a message is encrypted. In other cryptosystems, it must be unpredictable (and hence chosen randomly).

Example 63. Can you think of a way to distinguish the numbers produced by a linear congruential generator from truly random ones?

Solution. An easy observation is the following: by construction, $x_{n+1} = ax_n + b \pmod m$, individual numbers don't repeat unless a full period is reached and everything repeats. Truly random numbers do repeat every now and then.

Of course, knowing the parameters a, b, m, the numbers generated by the PRG are terribly **predictable**. Knowing just one number, we can produce all the next ones (as well as the ones before). A PRG that is safe for cryptographic purposes should not be predictable like that! (See next example.)

Armin Straub
straub@southalabama.edu

The next example illustrates the vulnerability of stream ciphers, based on predictable PRGs, to attacks based on partial information about the plaintext (recall that it is common to know or guess pieces of plaintexts; for instance every PDF begins with $\%\mathrm{PDF}$).

Example 64. (homework) Eve intercepts the ciphertext $c=(111\ 111\ 111)_2$. It is known that a stream cipher with PRG $x_{n+1}=5x_n+3 \pmod 8$ was used for encryption. Eve also knows that the plaintext begins with $m=(110\ 1...)_2$. Help her crack the ciphertext!

Solution. Ignore the solution until you had a chance to think about the problem!

The final answer is $m=(110\ 111\ 100)_2$. (It's not unlikely that I made a typo. Let me know!)

...

Since $c = m \oplus \mathrm{PRG}$, we learn that the initial piece of the keystream is $\mathrm{PRG} = m \oplus c = (110\ 1...)_2 \oplus (111\ 1...)_2 = (001\ 0...)_2$. Since each x_n is 3 bits, we conclude that $x_1=(001)_2=1$.

Because the PRG is predictable, we can now recreate the entire keystream! Using $x_{n+1}=5x_n+3 \pmod 8$, we find $x_2=0$, $x_3=3$, ... In other words, $\mathrm{PRG}=1,0,3,...=(001\ 000\ 011\ ...)_2$.

Hence, Eve can decrypt the ciphertext and obtain $m = c \oplus \mathrm{PRG} = (111\ 111\ 111)_2 \oplus (001\ 000\ 011)_2 = (110\ 111\ 100)_2$.

Here is another basic idea to generate pseudorandom numbers:

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

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

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

Example 65. 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. $(x_1, x_2, x_3, ...) = (0, 1, 1, 0, 1, 1, ...)$ 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 $(0, 0, 0, 0, ...)$), 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 $0, 1, 1, 2, 3, 5, 8, 13, ...$

Example 66. 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 $(0, 0, ..., 0)$ 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 $f(T)=x^\ell+c_1x^{\ell-1}+c_2x^{\ell-2}+...+c_\ell$ 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 $2^{31} - 1$.