Radbound University

Stream ciphers: stream encryption and LFSRs

Cryptography, Autumn 2021

Lecturers: J. Daemen, B. Mennink

September 9, 2021

Institute for Computing and Information Sciences
Radboud University

Outline

Modular arithmetic

The one-time pad and stream encryption

Linear feedback shift registers

Attacks on stream ciphers

Modular arithmetic

Modular (clock) equivalence

Integers

$\{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$ form the set of integers $\mathbb{Z}$

- On the clock, '**1 o'clock**' looks the same as '**13 o'clock**'
  - we say "1 and 13 are congruent modulo 12"
  - mathematically: write $1 \equiv 13 \pmod{12}$
- Extending it:

$$\begin{aligned} 5 &\equiv 29 \pmod{12} && \text{since } 5 + (2 \cdot 12) = 29 \\ 5 &\equiv 53 \pmod{12} && \text{since } 5 + (4 \cdot 12) = 53 \\ 7 &\equiv -5 \pmod{12} && \text{since } 7 + (-1 \cdot 12) = -5 \end{aligned}$$

Modular equivalence of integers

$a, b \in \mathbb{Z}$ are congruent modulo $n \in \mathbb{N}$ if $a - b$ is divisible by n

Modular arithmetic

- Reduction modulo n of an integer
  - returns its equivalent in the interval $[0, n-1]$
  - $c \leftarrow a \bmod n$
  - c is the remainder after division of a by n
- Addition modulo n as an operation
  (1) $c \leftarrow a + b$
  (2) if $c \geq n$, $c \leftarrow c - n$

  Notation: $a + b \bmod n$ or just $a + b$
- Multiplication modulo n as an operation
  (1) $c \leftarrow a \cdot b$
  (2) do the result modulo n: $c \leftarrow c \bmod n$

  Notation: $a \cdot b \bmod n$ or just $a \cdot b$
- We speak of *addition and multiplication in* $\mathbb{Z}/n\mathbb{Z}$

The one-time pad and stream encryption

The one-time pad

Encryption:

$$\begin{array}{rccccccccccccccc} M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \bigoplus$$

Decryption:

$$\begin{array}{rccccccccccccccc} C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array} \bigoplus$$

Bits are elements of $\mathbb{Z}/2\mathbb{Z}$ that are added modulo 2

- One-time pad [wikipedia] gives perfect secrecy if
  - key has same length as all plaintext together
  - adversary has no information about the key bits

Stream encryption

Encryption where a keystream is bitwise added to plaintext

- Addition can be over other sets
- Historically one used the 26-letter alphabet a lot
  - letters map to $\mathbb{Z}/26\mathbb{Z}$: $A = 0, B = 1, \ldots$
  - addition of letters modulo 26: e.g. $C + D = F$
- Main point: encryption is a simple symbol-by-symbol operation

Stream ciphers

To make stream encryption practical: generate a long keystream Z from a short key K

Stream cipher [wikipedia]

Algorithm to convert a short key K into a long keystream Z

Questions we will address in this part of the course:

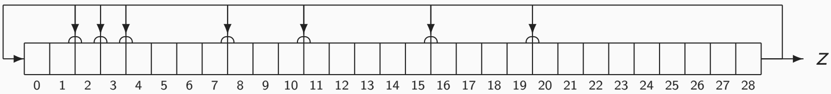
- How do we build a secure stream cipher?
- What does *secure* mean in the first place?

Stream cipher attempt: Vigenère cipher [wikipedia]

- Historical cipher for pen-and-paper encryption/decryption
- Operation
  - plaintext: sequence of letters
  - K: a password, e.g., LEMON
  - Z: K repeated all over, LEMONLEMONLEMONL . . .
  - addition modulo 26 gives ciphertext
  - plaintext ATTACKATDAWN gives ciphertext LXFOPVEFRNHR
- Compact and efficient
- Problems:
  - knowledge of short plaintext sequence reveals full keystream: *known plaintext attack*
  - long ciphertext enciphered leak via letter frequencies: *ciphertext-only attack*

Linear feedback shift registers

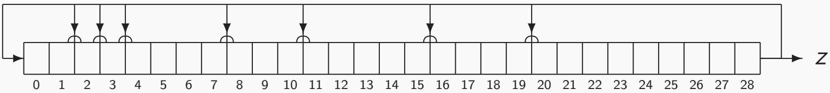
Linear feedback shift register (LFSR)



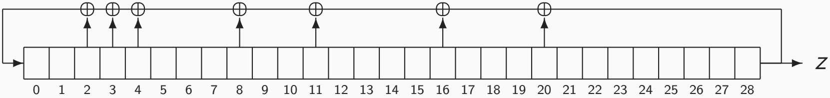
- Goal: efficiently generate a non-repeating sequence Z
- Mechanism
  - circuit with state s that is regularly clocked
  - each cell contains a bit s_i
  - each clock cycle: cells move right $s_{i+1} \leftarrow s_i$
  - ...for some positions (feedback taps) $s_{i+1} \leftarrow s_i + s_{28}$
  - rightmost cell is output: $z \leftarrow s_{28}$
- Can be studied with *finite fields* [for info only]
- Maximum-length LFSR
  - If feedback taps are well chosen, cycle length is $2^n - 1$

Galois vs Fibonacci LFSRs [for info only]

Galois LFSR:

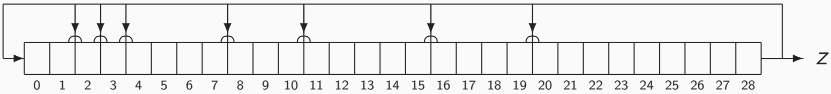


Fibonacci LFSR:



- Different configurations but similar output sequences
- One can prove that ∀ Fibonacci LFSR, ∃ Galois LFSR generating same sequence Z
- Each has its own advantages
  - Galois is more *parallel*, Fibonacci more *serial*
  - Galois reveals finite field operation, Fibonacci recursion in sequence

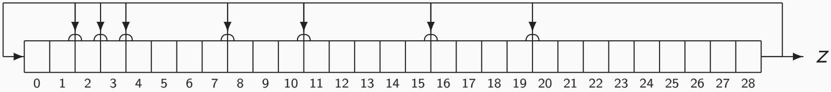
LFSRs, continued



- LFSR features
  - very simple to implement: just a shift and some XORs
  - keystream has good local statistical properties
  - bits of Z satisfy recurrence relation
- How to use it as a stream cipher?
  - write cipher key K in n-bit state ($|K| = n$)
  - each clock cycle a keystream bit z_t is generated
  - run for at most $2^n - 1$ cycles
- Distinction between algorithm and key:
  - public algorithm AKA cipher: LFSR length and tap positions
  - security should be based on secrecy of K (Kerckhoffs principle)

Attacks on stream ciphers

Attacks on LFSR: exhaustive key search

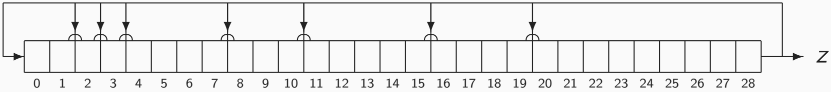


- Setting: adversary gets C and $C = P + Z$ with P a *meaningful* plaintext: ciphertext-only attack
- Exhaustive key search
  - make a guess K' for the value of K
  - generate the corresponding keystream Z'
  - compute $P' = C + Z'$ and check if P' is meaningful
  - if so, ready. Otherwise, keep on guessing
- Implications
  - for k-bit key, probability to find key after N guesses: $N2^{-k}$
- Generically true for any cipher if adversary has $\geq k$ output bits

Lesson learnt: upper bound to the security strength s of a cipher

Security strength s of a cipher with a k-bit key is at most k

Attack on LFSR: state reconstruction using linear algebra



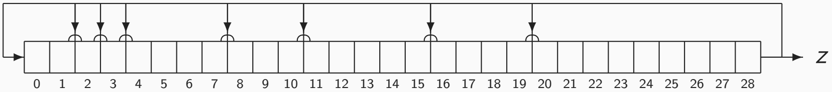
- Setting: adversary can obtain n subsequent bits of keystream z_t: known plaintext attack
- Actually, n keystream bits allow reconstructing the full state!
  - make sure you see why that is
  - countermeasure: *decimate* the keystream
  - so we only give out one bit per 10 (or so) cycles, creating *holes*
- This is not good enough, due to linearity of LFSR
  - update function of LFSR is linear function

Linearity

A function f is linear (over $\mathbb{Z}/2\mathbb{Z}$) if $f(x+y) = f(x) + f(y)$

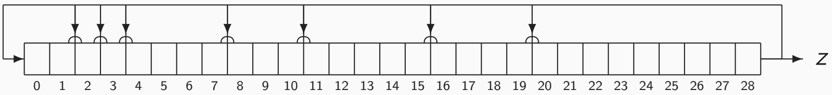
If f_1 and f_2 are linear, $f_2 \circ f_1$ is linear

LFSR state update: matrix multiplication



$$
\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \\ s_{15} \\ s_{16} \\ s_{17} \\ s_{18} \\ s_{19} \\ s_{20} \\ s_{21} \\ s_{22} \\ s_{23} \\ s_{24} \\ s_{25} \\ s_{26} \\ s_{27} \\ s_{28} \end{pmatrix}
\leftarrow
\begin{pmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot
\end{pmatrix}
\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \\ s_{15} \\ s_{16} \\ s_{17} \\ s_{18} \\ s_{19} \\ s_{20} \\ s_{21} \\ s_{22} \\ s_{23} \\ s_{24} \\ s_{25} \\ s_{26} \\ s_{27} \\ s_{28} \end{pmatrix}
$$

LFSR state reconstruction using linear algebra (cont'd)

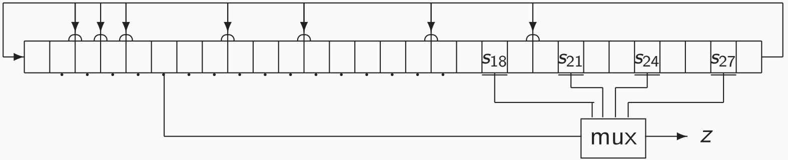


- We have $s^t \leftarrow M \cdot s^{t-1}$ and $s^t \leftarrow M \cdot M \cdot s^{t-2} = M^2 s^{t-2}$, etc.
  - hence $s^t = M^t s^0$ and $s^0 = K$ so $s^t = M^t K$
  - for some iterations, adversary knows z, the bit s_{28} of s^t
  - last row of $s^t = M^t K$ lefthand known: 1 linear equation of K
  - if we have n or more such equations, we can solve for K
  - solving: Gaussian elimination with negligible effort: $O(n^3)$
- This is generic: linear ciphers can be broken with linear algebra

Lesson learnt: need for non-linearity

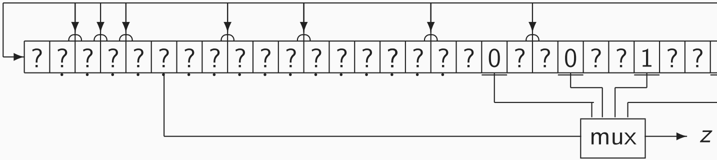
Purely linear ciphers offer no security

Filtered LFSR



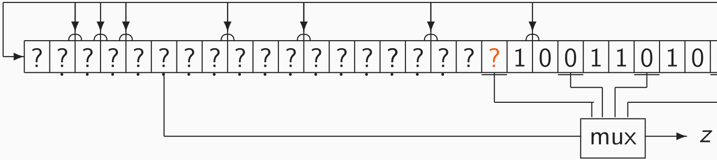
- Introduce a non-linear *output function*
  - instead of using an LFSR statebit as keystream bit $z_t = s_{n-1}^t$
  - ...compute z as a function of statebits: $z = f(s_0, \ldots s_{n-1})$
  - ...with f a non-linear function
- Example on this slide: a 16-to-1 multiplexer
  - z selected from a position in a range of 16 possibilities
  - by *address bits*: $z = s_A$ with $A = 1 + s_{18} + 2s_{21} + 4s_{24} + 8s_{27}$
- It is a non-linear function. See for example a 2-to-1 multiplexer
  - address bit s_0 and range $[1, 2]$: $z = s_{(s_0+1)} = (s_0 + 1)s_1 + s_0 s_2$
- Uncertainty on where output bit comes from complicates attacks
- Attacks are still possible but require more sophistication

Filtered LFSR and guess-and-determine attack



- Setting: adversary can obtain n subsequent bits of keystream z_t: known plaintext attack
- Principle of a guess-and-determine attack
  - make a guess for a subset of the bits of the state
  - combined with output Z, this determines other statebits
- In our specific MUX-LFSR case here:
  - given address bits, we can locate where z_t comes from
  - guessing 4 bits of state s^t gives us one statebit of s^t for free
  - we can transfer the knowledge of s^t to s^{t+1}
  - then guess 4 more statebits and get one more statebit for free
  - this will be faster than exhaustive key search

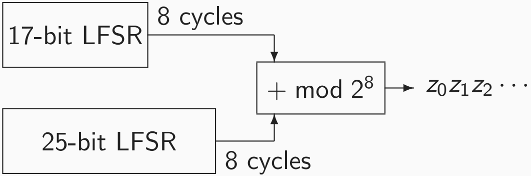
Filtered LFSR and guess-and-determine attack



- Recursive algorithm specific for our LFSR example
  - starting for all possible values of rightmost 10 bits
  - for two guesses of bit in position indicated with "?"
    - use output z to determine the statebit *chosen* by the mux
    - if contradiction, cut this branch
    - else, fill in in LFSR and repeat procedure
  - tree search where each node has at most two children
    - only one child if value of "?" is known
    - no children if contradiction
  - LFSR state with all bits known and no contradiction: ready!

Combiner LFSR and divide-and-conquer attacks

- Combiner LFSR:
  - non-linear output function taking bits from several LFSRs
  - real-world content-scrambling cipher (for pay TV in 80s):



- Divide-and-conquer attack, adversary has Z (known plaintext)
  - guess state of top LFSR
  - each byte z_i allows reconstructing output byte of bottom LFSR
  - 4 output bytes z_t give 32 output bits of bottom LFSR
  - should satisfy recurrence relationship
  - total complexity: some subtractions modulo 2^8 and checking recurrence relation for about 2^{17} guesses