

## Block vs. Stream cipher

**Idea of a block cipher:** partition the text into relatively large (e.g. 128 bits) blocks and encode each block separately. The encoding of each block generally depends on at most one of the previous blocks.

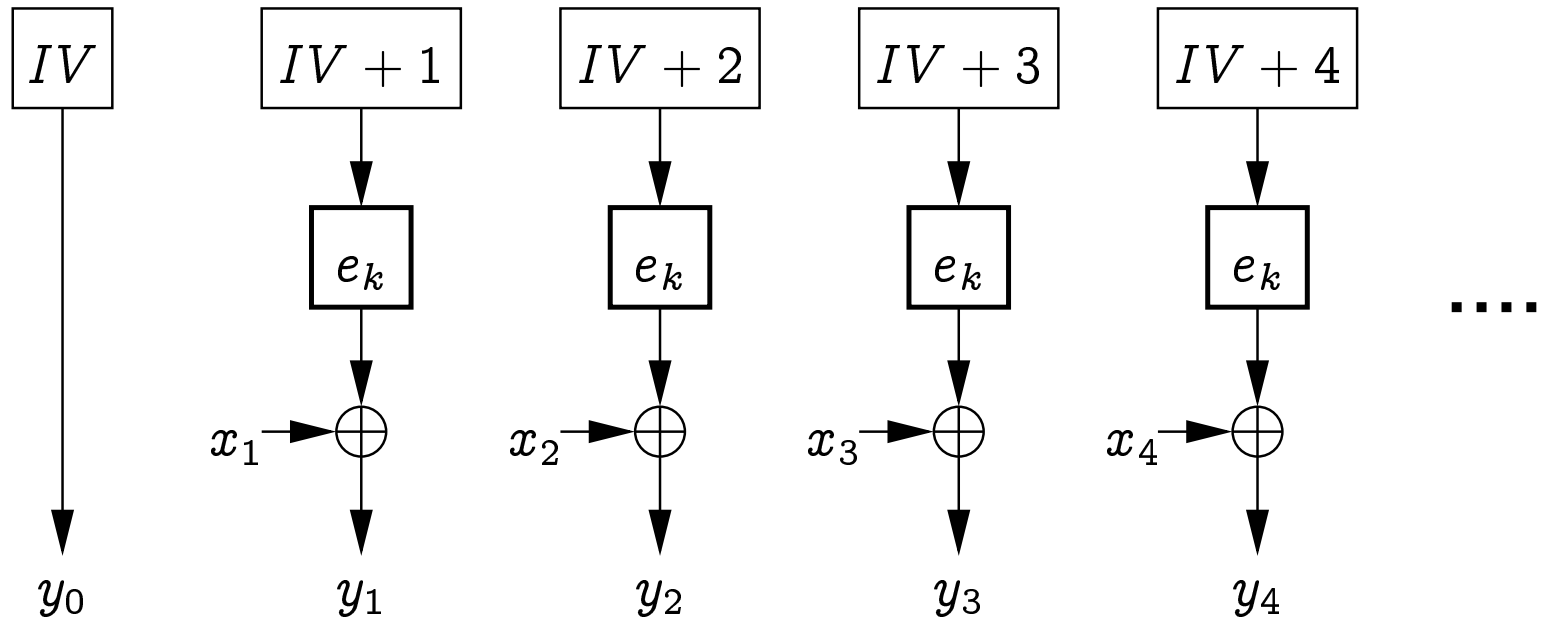
- the same “key” is used at each block.

**Idea of a stream cipher:** partition the text into small (e.g. 1 bit) blocks and let the encoding of each block depend on many previous blocks.

- for each block, a different “key” is generated.

## Examples of stream ciphers

- One-time pad.
- Block cipher in OFB or **CTR** mode.

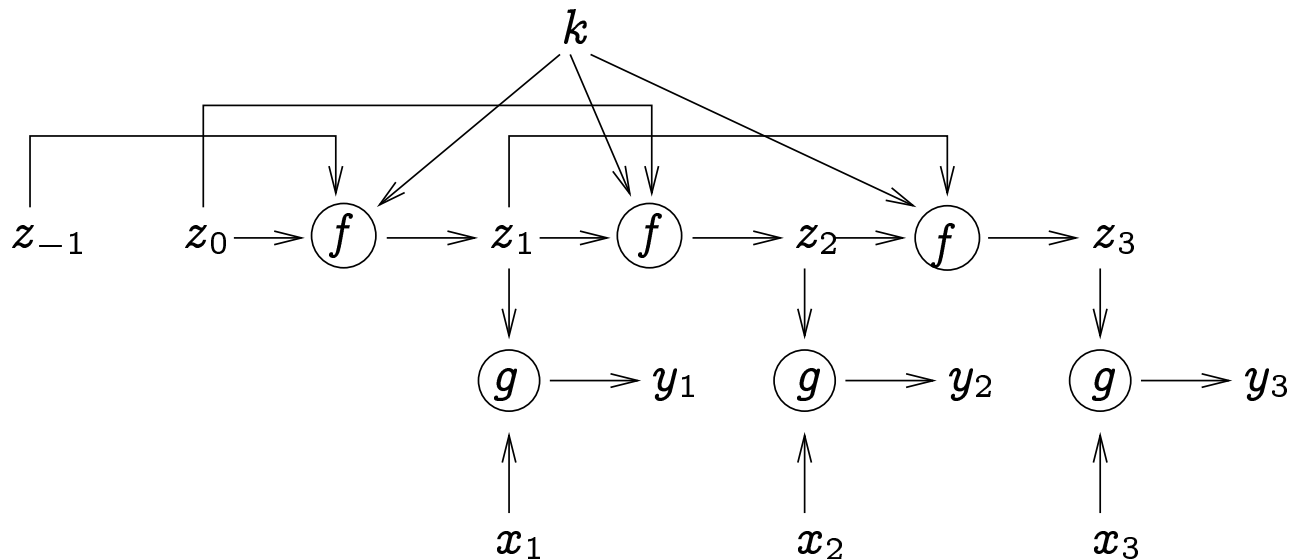


# Synchronous stream ciphers

**Definition 1.** A stream cipher is **synchronous** if its key sequence does not depend on the plain- and ciphertexts but only on the previous elements of the key sequence and the initial key.

$$z_i = f(z_{i-1}, z_{i-2}, \dots, z_{i-t}, k),$$

$$y_i = g(x_i, z_i).$$



## Properties of the synchronous stream cipher

1. The encoder and decoder must be synchronized, i.e. the decoder must always make sure that it applies the right element of the key sequence to the given element of the ciphertext sequence.
2. If an element of the ciphertext sequence has been changed (but not deleted) then only the corresponding plaintext element is affected.

One-time pad is a synchronous stream cipher.

Other synchronous stream ciphers could be called “pseudo one-time pads”.

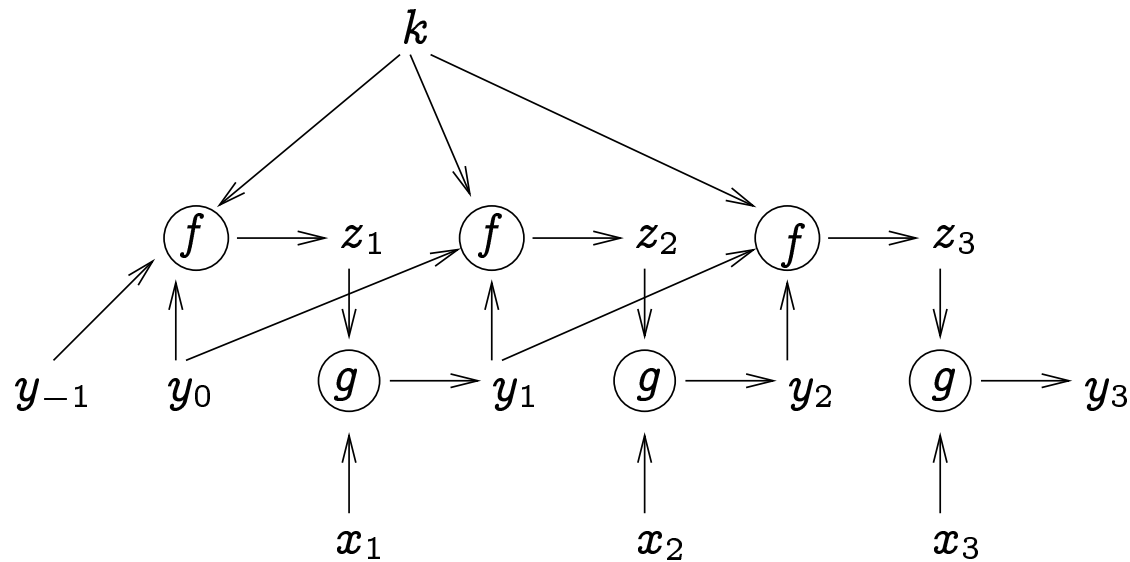
They are as secure as hard it is to distinguish  $(z_i)$  from a truly random sequence.

## Self-synchronizing stream ciphers

**Definition 2.** A stream cipher is **self-synchronizing** if its keystream depends on the plain- or ciphertext.

$$z_i = f(y_{i-1}, y_{i-2}, \dots, y_{i-t}, k),$$

$$y_i = g(x_i, z_i).$$



## Properties of a self-synchronizing stream cipher

1. If a ciphertext block is changed somehow (either randomly or adventurously) then only the decryptions of the next  $t$  blocks are affected. Hence the decoding process synchronizes itself.
2. The rather quick reappearance of the correct decoding means that the tampering of some ciphertext blocks may remain unnoticed.
3. As the cryptotext blocks depend on all preceding plaintext blocks, the statistical analysis of the cryptotext is hopefully more difficult.

## Linear keystream generator

Let  $c_1, \dots, c_t \in \{0, 1\}$  certain fixed bits and  $z_0, \dots, z_{t-1}$  the initial keystream bits. The subsequent bits  $z_i$  of the keystream ( $z_n$ ), where  $i \geq t$ , are generated using the rule

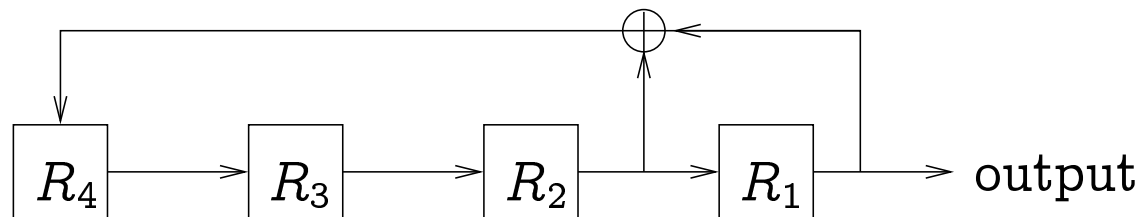
$$\begin{aligned} z_i &= f(z_{i-1}, z_{i-2}, \dots, z_{i-t}) = \\ &= (c_1 \cdot z_{i-1} + c_2 \cdot z_{i-2} + \dots + c_t \cdot z_{i-t}) \bmod 2. \end{aligned}$$

**Example:** let  $t = 4$ ,  $c_1 = c_2 = 0$  ja  $c_3 = c_4 = 1$  and  $(z_0, z_1, z_2, z_3) = (0, 1, 0, 0)$ . The output of the generator is then

0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, ...

## Linear feedback shift register (*Lineaarse tagasisidega nihkeregister*)

... is an electronic gadget for generating a linear keystream. The LFSR corresponding to the previous example is

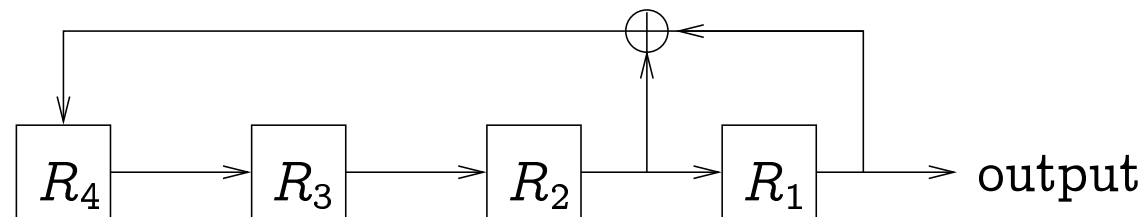




# LFSR works like this...

step	$R_4$	$R_3$	$R_2$	$R_1$
0	0	0	1	0
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	1	0	1	1
5	0	1	0	1
6	1	0	1	0
7	1	1	0	1

step	$R_4$	$R_3$	$R_2$	$R_1$
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	0	0	1	1
12	0	0	0	1
13	1	0	0	0
14	0	1	0	0
15	0	0	1	0



## Periodic sequences

**Definition 3.** A sequence  $z_0, z_1, z_2, \dots$  is **periodic** if there exists a  $d \geq 1$ , such that  $z_i = z_{i+d}$  for all  $i \geq 0$ . The smallest such  $d$  is called the **period** of that sequence.

If  $z_i = z_{i+d}$  holds only for all sufficiently large  $i$ -s, then  $z$  is called **eventually periodic**.

**Exercise.** Show that the sequence of bits generated by any LFSR is eventually periodic. Show that the period of a sequence generated by a  $t$ -register LFSR is at most  $2^t - 1$ . Show that if this bound is reached then each period contains  $2^{t-1}$  bits 1 and  $2^{t-1} - 1$  bits 0.

**Exercise.** Show that any (eventually) periodic sequence can be generated by an LFSR.

The **linear complexity**  $L(z)$  of a sequence  $z$  is the minimal number of registers that a LFSR needs to output this sequence.

## LFSR as a stream cipher

- $\mathcal{P} = \mathcal{C} = \{0, 1\}^*$ .
- A key  $k \in \mathcal{K}$  consists of
  - $t \in \mathbb{N}$ ;
  - $c_1, \dots, c_t \in \{0, 1\}$ ;
  - $z_0, \dots, z_{t-1} \in \{0, 1\}$ .
- To encode or decode  $x \in \{0, 1\}^n$ : compute  $z_i = c_1 z_{i-1} + \dots + c_t z_{i-t} \pmod{2}$  for  $t \leq i \leq n - 1$  and output  $x \oplus z$ .

A synchronous stream cipher... how difficult it is to distinguish  $(z_i)$  from a truly random sequence of bits?

I.e. if we know a part of the sequence  $(z_i)$ , how difficult it is to predict the next element(s)?



## If we do not know the linear complexity

Then we must (over)estimate it. Let us know  $t \geq L(z) = t'$ .

We need  $2t$  consecutive keystream bits; solve the same system of equations.

It has a solution: if  $(c'_1, \dots, c'_{t'})$  are the coefficients of a minimal LFSR generating  $z$  then

- $c_i = c'_i$  if  $1 \leq i \leq t'$ ;
- $c_i = 0$  if  $t' + 1 \leq i \leq t$

is a solution to the system of equations.

There are other solutions as well.

We can even determine the linear complexity.

## Polynomials over a ring $R$

Formally, a polynomial  $p$  is a mapping  $\mathbb{N} \rightarrow R$  (here  $\mathbb{N} = \{0, 1, 2, \dots\}$ ) with only finitely many non-zero entries. Denote  $p(i)$  by  $p_i$ .

The values of  $p$  are called its **coefficients**.

$p$  is usually written as  $p_0 + p_1x + p_2x^2 + \dots + p_mx^m$  where  $p_{m+1} = p_{m+2} = \dots = 0$ .

Polynomials can be

- added, multiplied, divided (with remainder),
  - division  $p/q$  is possible if the leading coefficient of  $q$  is invertible.
- multiplied with a scalar
- used as arguments to Euclid's algorithm, giving their gcd.

The set of all polynomials over  $R$ , denoted  $R[x]$ , is a ring.

## Formal series over a ring $R$

A formal series  $f$  is a mapping  $\mathbb{N} \rightarrow R$ . The values of  $f$  are called its coefficients.  $f$  is usually written as  $\sum_{i=0}^{\infty} f_i x^i$ .

Formal series can be added, multiplied, multiplied with a scalar:

$$(f + g)_i = f_i + g_i \quad (f \cdot g)_i = \sum_{j=0}^i f_j \cdot g_{i-j} \quad (kf)_i = k \cdot f_i$$

If  $f_0$  is invertible then  $f^{-1}$  exists and is given by

$$(f^{-1})_0 = f_0^{-1} \quad (f^{-1})_i = -f_0^{-1} \cdot \left( \sum_{j=0}^{i-1} (f^{-1})_j f_{i-j} \right)$$

The set of all formal series over  $R$ , denoted  $R[[x]]$ , is a ring.

A sequence  $(z_i)$  can also be seen as a formal series (over  $\mathbb{Z}_2$ ).

## Rational formal series

A formal series  $f$  is **rational** if it can be expressed as  $pq^{-1}$ , where  $p$  and  $q$  are polynomials. Then  $f_0 = p_0q_0^{-1}$  and

$$\begin{aligned}
 f_i &= \sum_{j=0}^i p_{i-j}(q^{-1})_j = q_0^{-1} \left( p_i - \sum_{j=1}^i \sum_{k=1}^j p_{i-j}(q^{-1})_{j-k} q_k \right) = \\
 q_0^{-1} \left( p_i - \sum_{k=1}^i q_k \sum_{j=k}^i p_{i-j}(q^{-1})_{j-k} \right) &= q_0^{-1} \left( p_i - \sum_{k=1}^i q_k \sum_{j=0}^{i-k} p_{i-k-j}(q^{-1})_j \right) = \\
 &= q_0^{-1} \left( p_i - \sum_{k=1}^i q_k f_{i-k} \right) .
 \end{aligned}$$

I.e. the coefficients of a rational formal series satisfy a linear recurrence.

If  $R$  is finite then the coefficients of  $f$  are eventually periodic.



**Theorem 1.** *Let the coefficients of  $f \in R[[x]]$  be eventually periodic. Then  $f$  is rational.*

**Proof.** Let  $f_0, \dots, f_{r-1}$  be the non-periodic part of  $f$ 's coefficients and let  $f_i = f_{i+t}$  for all  $i \geq r$ . Let

$$p^{(0)} = \sum_{i=0}^{r-1} f_i x^i \quad \text{and} \quad p = \sum_{i=0}^{t-1} f_{r+i} x^i .$$

Then

$$\begin{aligned} f &= p^{(0)} + p \cdot x^r + p \cdot x^{r+t} + p \cdot x^{r+2t} + \dots = p^{(0)} + (1 + x^t + x^{2t} + \dots) \cdot p \cdot x^r = \\ &= p^{(0)} + (1 - x^t)^{-1} \cdot p \cdot x^r = \frac{p^{(0)} \cdot (1 - x^t) + p \cdot x^r}{1 - x^t} \end{aligned}$$

**Corollary.** If  $f$ 's coefficients are periodic (i.e.  $r = 0$ ) then  $f = \frac{p}{q}$  where  $\deg p < \deg q$ .

## Rational formal series over $\mathbb{Z}_2$ vs. LFSRs

Let  $c_1, \dots, c_t$  and  $z_1, \dots, z_t$  be given. Compare:

$$f_i = p_i + \sum_{k=1}^i q_k f_{i-k} \quad \text{and} \quad z_i = \sum_{k=1}^t c_k z_{i-k}$$

Take

- $q_0 = 1$ ,  $\deg q = t$  and  $q_i = c_i$  for  $1 \leq i \leq t$ .
- $p_i = z_i + \sum_{k=1}^i c_k z_{i-k}$  for  $1 \leq i \leq t$ .

Then  $f = z$ .

$q$  is the [characteristic polynomial](#) of the LFSR generating  $z$ .

If  $L(z) = t$  then  $q$  is the [characteristic polynomial](#) of  $z$ .

## Finding the characteristic polynomial

Given:  $z_r, \dots, z_{r+2t-1}$ , such that  $t \geq L(z)$ . Assume  $r = 1$ .

- Compute  $c_1, \dots, c_t$  by solving a system of linear equations.
- Find  $q$  and  $p$  as in the previous slide.
- Return  $q/\text{gcd}(p, q)$ .

Berlekamp-Massey algorithm is faster...  $O(t^2)$ .

**Exercise:** if we know  $L(z)$  and solve the system of equations containing exactly  $L(z)$  unknowns, then is it possible to get more than one solution?

**Exercise:** Let  $z$  and  $z'$  be two eventually periodic sequences of bits. Let  $z''$  be a sequence defined by  $z''_i = z_i \oplus z'_i$ . Find an upper bound for  $L(z'')$  in terms of  $L(z)$  and  $L(z')$ .

## Berlekamp-Massey algorithm

B-M algorithm reads the coefficients  $f_0, f_1, f_2, \dots$  one by one.

At the  $i$ -th stage of the algorithm (having read  $f_0, \dots, f_{i-1}$ ) it has computed polynomials  $p^{(i)}, q^{(i)}$ , such that

- $f \cdot q^{(i)} = p^{(i)} \pmod{x^i}$
- The size  $\Phi(p^{(i)}, q^{(i)}) = \max\{\deg q^{(i)}, (\deg p^{(i)}) + 1\}$  of the LFSR is minimal.

**Step** of the algorithm: read  $f_i$ . If  $f \cdot q^{(i)} = p^{(i)} \pmod{x^{i+1}}$  then  $q^{(i+1)} = q^{(i)}, p^{(i+1)} = p^{(i)}$ .

Otherwise we have a [discrepancy](#).

## Recomputing $p$ and $q$

We have  $f \cdot q^{(i)} = p^{(i)} + b^{(i)}x^i \pmod{x^{i+1}}$  for some  $b^{(i)} \neq 0$ .

Let  $j$  be some earlier discrepancy:  $f \cdot q^{(j)} = p^{(j)} + b^{(j)}x^j \pmod{x^{j+1}}$ .

Take  $p^{(i+1)} = p^{(i)} - \frac{b^{(i)}}{b^{(j)}}x^{i-j}p^{(j)}$  and  $q^{(i+1)} = q^{(i)} - \frac{b^{(i)}}{b^{(j)}}x^{i-j}q^{(j)}$

**Exercise.** Show that  $f \cdot q^{(i+1)} = p^{(i+1)} \pmod{x^{i+1}}$

In the B-M algorithm, the last such  $j$  is used where  $\Phi(p^{(j)}, q^{(j)}) < \Phi(p^{(j+1)}, q^{(j+1)})$

## B-M algorithm: initialization

- If  $f_0 = f_1 = f_2 = \dots = 0$  then return  $(p = 0, q = 1)$ .
- Let  $m$  be such, that  $f_0, \dots, f_{m-1} = 0, f_m \neq 0$ .
- Let  $p^{(m)} = 0, q^{(m)} = 1, p^{(m+1)} = f_m x^m,$   
 $q^{(m+1)} = 1 + x^m,$  if  $m > 0$  and  $q^{(m+1)} = 1$  if  $m = 0$ .
- At step  $m$ , the linear complexity changed.

## Proof of correctness

For a **finite** or infinite sequence  $s$  let  $L(s)$  be the length of the shortest LFSR outputting it.

**Linear complexity change lemma.** Let  $s$  be a finite sequence of length  $k + 1$  and  $s'$  the prefix of  $s$  of length  $k$ . Let  $\mathcal{G}$  be an LFSR of size  $L(s')$  generating  $s'$ . If  $\mathcal{G}$  does not generate  $s$  then  $L(s) \geq \max\{L(s'), k + 1 - L(s')\}$ .

**Proof.** Obviously  $L(s) \geq L(s')$ .

Let  $\mathbf{1}$  be the sequence  $0^k 1$ . Then  $\mathcal{G}$  generates  $s \oplus \mathbf{1}$  and

$$k + 1 = L(\mathbf{1}) = L((s \oplus \mathbf{1}) \oplus s) \leq L(s \oplus \mathbf{1}) + L(s) \leq L(s') + L(s) .$$

Hence  $L(s) \geq k + 1 - L(s')$

□

## LFSR-s with long periods

We know that an LFSR with  $t$  registers can output a sequence with a period at most  $2^t - 1$ . How to achieve this upper bound?

**Definition 4.** A polynomial  $p \in R[x]$  is **reducible** if there exists a polynomial  $q \in R[x]$ , such that  $1 \leq \deg q < \deg p$  and  $q \mid p$ . Otherwise we call  $p$  **irreducible**.



## Residue classes modulo polynomials

Given  $p \in R[x]$  with invertible leading coefficient we can consider the set  $R[x]/p$  of all *residue classes* of  $R[x]$  modulo  $p$ . We can define addition and multiplication on this set.

- $R[x]/p \equiv \{\bar{q} \mid q \in R[x], \deg q < \deg p\}$ ;
- $\bar{q}_1 + \bar{q}_2 = \overline{q_1 + q_2}$ ;
- $\bar{q}_1 \cdot \bar{q}_2 = \overline{q_1 \cdot q_2 \bmod p}$ .

The structure  $(R[x]/p, +, \cdot)$  is a ring. If  $R$  is a field and  $p$  is irreducible then  $R[x]/p$  is a field.

- Finding inverses in  $R[x]/p$  — just like in  $\mathbb{Z}_q$ , where  $q \in \mathbb{P}$ .
  - Main tool — Euclid's algorithm.

## Finite fields

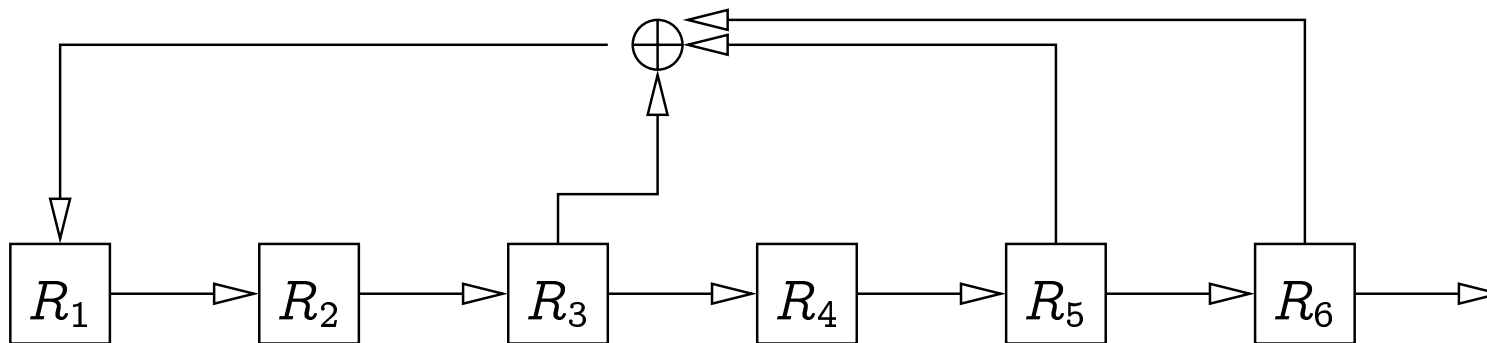
- Let  $p$  be an irreducible polynomial of degree  $m$  over a finite field  $K$ . Then  $K[x]/p$  is a field with  $p^m$  elements.
- Up to isomorphism, there exists only one field with  $p^m$  elements.
- If  $K$  is a finite field then  $K^*$  is **cyclic**, i.e. it can be generated by a single element of  $K^*$ .

**Definition 5.** A polynomial  $p \in K[x]$  is **primitive** if  $\bar{x}$  is a generator of  $(K[x]/p)^*$ .

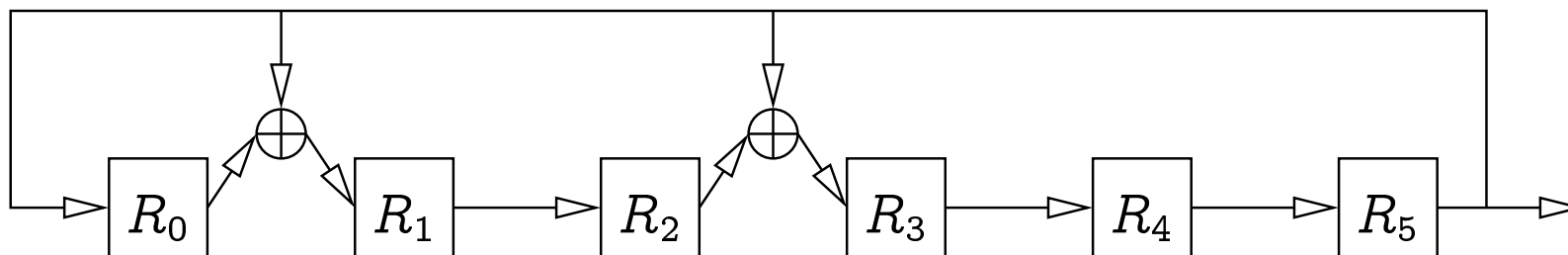
**Theorem 2.** *If a LFSR's characteristic polynomial  $q$  is primitive then the period of the sequence produced by this LFSR is  $2^{\deg q} - 1$  for any non-zero initialization vector.*

Proof follows...

## LFSRs in Galois configuration



Normal; characteristic polynomial:  $1 + x^3 + x^5 + x^6$



Galois configuration; char. polynomial:  $1 + x + x^3 + x^6$

Char. poly. of Galois conf: contains  $x^i$  if there is extra input to  $R_i$ .  
 Also contains  $x^t$  where  $t$  is the number of registers.

## Evolution of state of a LFSR in Galois conf.

A state  $S$  of a  $t$ -register LFSR can be represented by a polynomial  $p^{(S)} = \sum_{i=0}^{t-1} S(R_i)x^i$  where  $S(R_i)$  is the contents of  $R_i$  in  $S$ .

The next state of LFSR in Galois conf. is then  $(x \cdot p^{(S)}) \bmod q$  where  $q$  is the characteristic polynomial.

If  $q$  is primitive then the state of that LFSR in Galois conf. passes through all  $2^t - 1$  possible values.

We can define an isomorphism between the states of a normal LFSR and its corresponding LFSR in Galois conf.

Hence the states of a LFSR pass through all  $2^t - 1$  possible values if the corresponding LFSR in Galois conf. has a primitive characteristic polynomial. The period of the output sequence is then  $2^t - 1$  as well.

## Reciprocals of polynomials

For a polynomial  $q = \sum_{i=0}^k q_i x^i$  with  $q_0 \neq 0 \neq q_k$  define its **reciprocal** by  $q^* = x^k \cdot q(1/x) = \sum_{i=0}^k q_{k-i} x^i$ .

If a (normal) LFSR has the characteristic polynomial  $q$  then the corresponding LFSR in Galois conf. has the characteristic polynomial  $q^*$ .

$q^* \cdot (q')^* = (q \cdot q')^*$ . Just compare the coefficients.

Hence  $q$  is irreducible iff  $q^*$  is.

$q$  (of degree  $k$ ) is primitive iff  $q^*$  is. Indeed, suppose an irreducible  $q$  is not primitive, then  $x^i \equiv 1 \pmod{q}$  for some  $i < p^k - 1$ . I.e. there exists a polynomial  $q'$ , such that  $x^i - 1 = qq'$ . Then also  $(x^i - 1)^* = q^* \cdot (q')^*$ . But  $(x^i - 1)^* = -(x^i - 1)$ . Hence  $q^* \cdot (-(q')^*) = x^i - 1$  and  $q^*$  is not primitive.

## Testing irreducibility

**Theorem 3.** *An irreducible polynomial  $m$  of  $d$ -th degree divides  $x^{p^d} - x$  in  $\mathbb{Z}_p[x]$ .*

**Proof.**  $\mathbb{Z}_p[x]/m$  is a field with  $p^d$  elements. Hence  $\alpha^{p^d} - \alpha = 0$  in  $\mathbb{Z}_p[x]/m$  for all  $\alpha \in \mathbb{Z}_p[x]/m$  (Fermat's little theorem). The polynomial  $x$  is also a member  $\mathbb{Z}_p[x]/m$ , hence  $x^{p^d} - x \equiv 0 \pmod{m}$  in  $\mathbb{Z}_p[x]$ .

**Theorem 4.** *If an irreducible polynomial  $m$  of  $d$ -th degree divides  $x^{p^{d'}} - x$  then  $d \mid d'$ .*

**Proof.** Let  $K$  be the field with  $p^{d'}$  elements; it contains exactly the roots of  $x^{p^{d'}} - x$ .  $K$  is a vector space over  $\mathbb{Z}_p$  with dimension  $d'$ . It contains also the roots of  $m$ , let  $\alpha$  be a root of  $m$ . Let  $S = \{\sum_{i=0}^{d-1} \lambda_i \alpha^i \mid \lambda_i \in \mathbb{Z}_p\}$ . Then  $S$  is a field. We have  $K \supseteq S \supseteq \mathbb{Z}_p$ , hence  $d' = \dim_{\mathbb{Z}_p} K = (\dim_{\mathbb{Z}_p} S) \cdot (\dim_S K) = d \cdot (\dim_S K)$  and  $d \mid d'$ .

## Testing irreducibility

$q \in \mathbb{Z}_p[x]$  is irreducible if  $\gcd(q, x^{p^i} - x) = 1$  for all  $i \leq (\deg q)/2$ .

Here  $x^{p^i}$  may be computed modulo  $q$ .

## Testing primitiveness

For  $q \in \mathbb{Z}_p[x]$  to be primitive, it must first be irreducible. Let  $m = \deg q$ .

We must have  $x^i \not\equiv 1 \pmod{q}$  for all  $i < p^m - 1$ . It is sufficient to test this only for the values  $i = (p^m - 1)/p'$  where  $p'$  is a prime factor of  $p^m - 1$ .



## Polynomials: exercise

Which of those polynomials are reducible, which are irreducible and which are primitive over  $\mathbb{Z}_2$ ?

1.  $x + 1$ ,

2.  $x^2$ ,

3.  $x^2 + 1$ ,

4.  $x^2 + x + 1$

5.  $x^3 + 1$

6.  $x^3 + x + 1$ ,

7.  $x^4 + x + 1$ ,

8.  $x^4 + x^2 + 1$ .

## Linear complexity: exercises

**Exercise.** Determine the linear complexities of the following sequences.

1.  $1, 0, 1, 0, 1, 0, \dots$
2.  $1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, \dots$
3.  $1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, \dots$

**Exercise.**

1. Construct a sequence with infinite linear complexity (i.e. a sequence that is generated by no LFSR).

## LFSR-s in real cryptosystems

Use one or more LFSR-s and apply some non-linear function to their outputs.

## Shrinking generator

... is constructed from two LFSRs working synchronously. The shrinking generator produces up to one bit for each bit-pair generated by these two LFSRs. It is defined as follows:

1. If the first LFSR outputs 1 then return the output of the second LFSR.
2. If the first LFSR outputs 0 then return nothing (discard the output of the second LFSR).

If the linear complexities of these two LFSR-s are  $L_1$  and  $L_2$  and their periods are maximal, then the linear complexity  $L$  of the shrinking generator satisfies

$$L_2 \cdot 2^{L_1-2} < L < L_2 \cdot 2^{L_1-1}.$$

## Other non-linear combiners

- Self-shrinking generator:
  - Generate two bits,  $b_1, b_2$ .
  - If  $b_1 = 1$  then output  $b_2$ . If  $b_1 = 0$  then output nothing.
- Majority: use three LFSR-s clocked synchronously, output the majority of their bits.
- Irregular clocking: use several LFSR-s, do not clock each of them at each clock cycle, decide the clocked LFSR-s in a non-linear way.