Linear Feedback Shift Registers and WG ciphers in GAP

Nuša Zidarič

supervisors:
Mark Aagaard, Guang Gong
University of Waterloo

13.01.2017

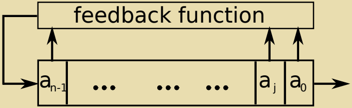
1. (L)FSR
    - what is a feedback shift register
    - properties of (L)FRS sequences
2. LFSR in GAP
3. WG stream ciphers
4. WG in GAP
5. Some WG-16 implementation details

Part I:
(L)FSR

WHAT IS A FEEDBACK SHIFT REGISTER

n-stage FSR consists of 3 components:

- n memory units (stages)
- feedback function (in n variables)
- regular clock: with each step the register contents are *shifted* and a new value is computed



output sequence: $\underline{a} = \{a_k\} = a_0, a_1, \ldots,$ where $a_i \in \mathcal{F}$

WHAT IS A FEEDBACK SHIFT REGISTER

feedback function = polynomial function in n variables

$$f : \mathcal{F}^n \to \mathcal{F}$$
$$f(x_0, x_1, \ldots, x_{n-1}) = \sum c_{i_0 i_1 \ldots i_{n-1}} x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}$$

where the sum runs over all possible n-tuples $(i_0, i_1, \ldots, i_{n-1}) \in \mathcal{F}^n$ and $c_t \in \mathcal{F} \, \forall t$

$\mathcal{F} = \mathbb{F}_q$ (a finite field with q elements) where q=prime or prime power

- if $q = 2$: $\mathcal{F} = \mathbb{F}_2$, f is boolean function , sequence is binary
- otherwise: $\mathcal{F} = \mathbb{F}_q$, f is polynomial function over $\mathbb{F}_q$, sequence is q-ary

note: q-ary simply means having q possible values

example: for $q = 2^m$ we need the shift register stages to be able to hold any of the q possible elements, meaning each stage consists of an m-bit register

note: $i_t \in \mathbb{F}_q = \{0, 1, \ldots, q-1\} \to x_j^{i_t}$ and $x^q = x \; \forall x \in \mathbb{F}_q$

WHAT IS A FEEDBACK SHIFT REGISTER

feedback function $= f(x_0, x_1, \ldots, x_{n-1}) = \sum c_{i_0 i_1 \ldots i_{n-1}} x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}$

$\mathcal{F} = \mathbb{F}_2$: how many boolean functions in n variables exist?

i_0	i_1	i_2	$\ldots$	i_{n-1}	monomial	corresp. c_t
0	0	0	$\ldots$	0	1	c_0
0	0	0	$\ldots$	1	x_{n-1}	c_1
			$\ldots$		$\ldots$	$\ldots$
1	1	0	$\ldots$	1	$x_0 x_1$	$\ldots$
			$\ldots$		$\ldots$	$\ldots$
1	1	1	$\ldots$	1	$x_0 x_1 \ldots x_{n-1}$	c_{2^n-1}

- number of monomials $= 2^n$
- $\forall$ of them has its own coefficient c_t
- $(c_0, c_1, \ldots, c_{2^n-1}) \in \mathbb{F}_2^{2^n}$ $\Rightarrow$ there are 2^{2^n} possible vectors $\Rightarrow$ number of boolean functions $= 2^{2^n}$

$\mathcal{F} = \mathbb{F}_q$: how many polynomial functions in n variables exist fro general case?

- number of monomials $= q^n$
- number of polynomial functions over $\mathbb{F}_q = q^{q^n}$

WHAT IS A FEEDBACK SHIFT REGISTER

degree of monomial $= \sum_{t=0}^{n-1} i_t$ for monomial $x_0^{i_0} x_1^{i_1} \dots x_{n-1}^{i_{n-1}}$

note: for boolean functions over $\mathbb{F}_2$ = number of variables present in monomial

		$q=2$ - Boolean functions			$q \neq 2$ - Polynomial functions		
	deg. d	general n	$n=3$ nr of mon.	$n=3$ monomials	general n	$n=3$ nr of mon.	$n=3$ monomials
const	0	$\binom{n}{0} = 1$	$\binom{3}{0} = 1$	c	$\binom{n}{0} = 1$	$\binom{3}{0} = 1$	c
lin	1	$\binom{n}{1} = n$	$\binom{3}{1} = 3$	x_0, x_1, x_2	$\binom{n}{1} = n$	$\binom{3}{1} = 3$	x_0, x_1, x_2
quad.	2	$\binom{n}{2}$	$\binom{3}{2} = 3$	x_0x_1, x_0x_2, x_1x_2	$\binom{n+2-1}{2}$	$\binom{4}{2} = 6$	x_0x_1, x_0x_2, x_1x_2 x_0^2, x_1^2, x_2^2
cubic	3	$\binom{n}{3}$	$\binom{3}{3} = 1$	$x_0x_1x_2$	$\binom{n+3-1}{3}$	$\binom{5}{3} = 10$	$x_0x_1x_2, x_0^2x_1, x_0x_1^2$ $x_0^2x_2, x_0x_2^2, x_1^2x_2$ $x_1x_2^2, x_0^3, x_1^3, x_2^3$
quad.	4	-	-	-	$\binom{n+4-1}{4}$	$\binom{6}{4} = 15$	$x_0^2x_1x_2, x_0x_1^2x_2,$ $x_0x_1x_2^2, x_0^2x_1^2, x_0^2x_2^2,$ $x_1^2x_2^2, x_0^3x_1, x_0^3x_2,$ $x_1^3x_2, x_0x_1^3, x_1x_2^3,$ $x_0x_2^3, x_0^4, x_1^4, x_2^4$

$$\textbf{degree of function} = \max_{\forall \text{monomials in } f} \{\text{degree of monomial}\}$$

examples of linear, quadratic and cubic functions:

$f(x_0, x_1, x_2) = x_0 + x_1$

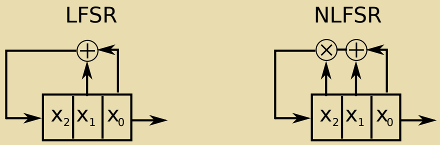
$f(x_0, x_1, x_2) = x_0x_1 + x_2$

$f(x_0, x_1, x_2, x_3) = x_0 + x_1 + x_1x_2x_3 + 1$ note: dB

WHAT IS A FEEDBACK SHIFT REGISTER

degree of the feedback function:

- linear $\rightarrow$ LFSR: n possible linear monomials, 2 possible constants for each $\Rightarrow 2^n$ possible LFSRs over $\mathbb{F}_2$
- nonliner $\rightarrow$ NLFSR: $n = 4$ stages over $\mathbb{F}_{2^4}$: even if limited to monomials of max degree 4, there are $\binom{7}{4} = 35$ such monomials, and for $q = 16$ that amounts to $16^{35} = 2^{140}$ different Boolean functions



$f(x_2, x_1, x_0) = x_0 + x_1$ $\quad$ $f(x_2, x_1, x_0) = x_0 x_1 + x_2$

WHAT IS A FEEDBACK SHIFT REGISTER

state = contents of the FSR at a given moment, as vector: $(a_0, a_1, \dots, a_{n-1}) \in \mathcal{F}^n$

- initial state $(a_0, a_1, \dots, a_{n-1})$
- state transition: $(a_0, a_1, \dots, a_{n-1}) \rightarrow (a_1, a_2, \dots, a_n)$

  where $a_n = f(a_0, a_1, \dots, a_{n-1})$

recursive relation: $a_{k+n} = f(a_k, a_{k+1}, \dots, a_{k+n-1})$, $k = 0, 1, \dots$

any consecutive n elements of the FSR sequence represent a state :

$$\underline{a} = \overbrace{a_0, a_1, \dots a_{n-1}}, a_n, \dots, \overbrace{a_k, a_{k+1}, \dots, a_{k+n-1}}, a_{k+n}, \dots$$

state diagram = directed graph with states as vertices and transitions as edges

state 0 $\rightarrow$ state 1 $\rightarrow \dots \rightarrow$ state k $\rightarrow$state k+1 $\rightarrow \dots$

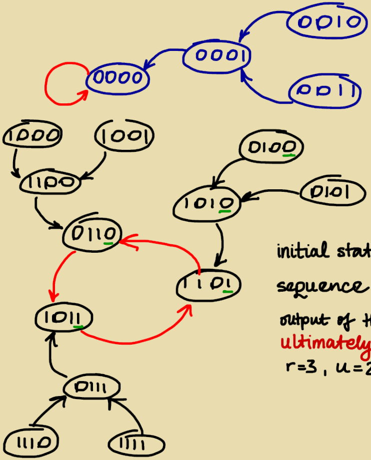
output sequence $\underline{a}$ is completely defined by the *feedback* and the *initial state*

WHAT IS A FEEDBACK SHIFT REGISTER

example 1: reducible polynomial

$f(x_0, x_1, x_2, x_3) = x_2 + x_3$

x_3	x_2	x_1	x_0	NEW x_3	NEXT STATE x_2	x_1	x_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	0	0	1	1	0
1	1	0	1	0	1	1	0
1	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1



seqn.

initial state: 0100

sequence: 00101101101...

output of the LFSR is **ultimately periodic**

$r=3$, $u=2$

PROPERTIES OF (L)FSR SEQUENCES

FSR sequence $\underline{a}$ is

- ultimately periodic: diff. states have the same successor (previous example)
- periodic: diff. states always have diff successors (no branches)

- **least period** $=$ smallest $r \ni: a_{i+r} = a_i \forall i \geq u$

  previous example: u=2, r=3
- if $u = 0$: $\underline{a}$ is periodic with r: $r \ni: a_{i+r} = a_i \forall i$

- any q-ary FSR sequence is ultimately periodic with period $r \leq q^n =$ number of different states

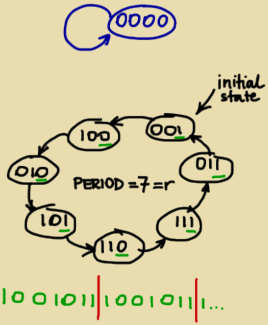
  note: if the LFSR (homogeneous) reaches the all 0 state it can no longer exit this state $\Rightarrow$ max period of LFSR $= q^n - 1$
- a sequence that has maximum period is called **max. length sequence** or **m-sequence**

PROPERTIES OF (L)FSR SEQUENCES

example 2: m-sequence

$$f(x_0, x_1, x_2) = x_0 + x_1$$

			NEW	NEXT STATE	
x_2	x_1	x_0	x_2	x_1	x_0
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	1	1



PROPERTIES OF (L)FSR SEQUENCES

characteristic polynomials of LFSR's

feedback function		characteristic polynomial
$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} c_i x_i$	$\Leftrightarrow$	$f(x) = x^n + \sum_{i=0}^{n-1} c_i x^i$

period of the output sequence is completely determined by the *period of the characteristic polynomial*:
period$(f(x)) = r$ if r is the smallest integer for which $f(x)|x^r + 1$

minimal polynomial is the characteristic polynomial with smallest **degree** (yielding the shortest LFSR), which generates the sequence $\underline{a}$ and period of the sequence equals the **period** of its minimal polynomial:

$$\text{period of sequence } \underline{a} = \text{period}(m(x))$$

If $f(x)$= characteristic polynomial of $\underline{a}$ and $m(x)$= minimal polynomial of the same sequence, then $m(x)|f(x)$. Note that an LFSR with char. poly $x^r + 1$ will generate a sequence with period r.

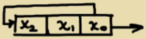
If a characteristic polynomial can be decomposed to irreducible factors as $f(x) = \prod_{i \in I} f_i(x)$ then period$(f(x))$ =lcm$\{$period$(f_i); \forall i \in I\}$.

PROPERTIES OF (L)FSR SEQUENCES

example 3: reducible polynomial

$f(x) = x^3 + 1$

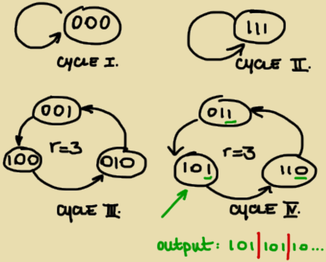
$f(x_0, x_1, x_2) = x_0$



NEW

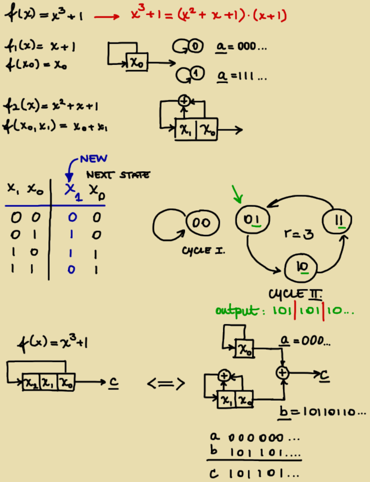
NEXT STATE

x_2	x_1	x_0	x_2	x_1	x_0
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	1	1



PROPERTIES OF (L)FSR SEQUENCES

example 3 - continued: the LFSR belonging to a factor of the reducible polynomial producing the same sequence



PROPERTIES OF (L)FSR SEQUENCES

If sequence $\underline{a}$ is ultimately periodic with (u, r):

$$m(x) = x^u m_1(x), \text{ where } m_1(0) \neq 0 \text{ and } m_1(x) | x^r + 1$$

then $\underline{a}$ can be generated with an LFSR. **Linear span** of $\underline{a}$ is defined as $\deg(m(x))$
recall example 1: $f(x) = x^2(x^2 + x + 1)$ where $u = 2, r = 3$

what can we learn from the polynomial?

- $f(x)$ is primitive $\Rightarrow$ one long cycle: m-sequence (period $q^n - 1$) - recall example 2
- $f(x)$ is irreducible $\Rightarrow$ distinct cycles with period $\leq q^n - 1$ - example 4
- $f(x)$ is reducible $\Rightarrow$ distinct cycles with period $\leq q^n - 1$ are found in each graph - examples 1,3

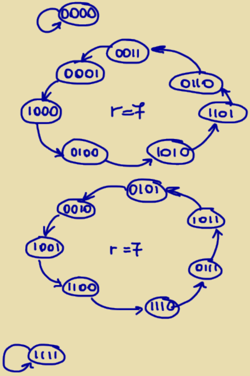
PROPERTIES OF (L)FSR SEQUENCES

example 4: irreducible polynomial

$f(x) = x^4 + x^2 + x + 1$ — IRREDUCIBLE

$f(x_0, x_1, x_2, x_3) = x_2 + x_1 + x_0$

x_3	x_2	x_1	x_0	NEW x_3	NEXT STATE x_2	x_1	x_0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	1	0	0	1	1	1	0
1	1	0	1	0	1	1	0
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	1



note: $x^7 + 1 = (x^4 + x^2 + x^1 + 1)(x^3 + x + 1) \Rightarrow \text{period}(f(x)) = 7 = r$

Part III:
LFSR in GAP

LFSR in GAP

LFSR(*args*)
creating the LFSR: 3 different constructors, versions with optional output-tap selection (marked with [])

LFSR(*K, charpol [, tap]***)**

LFSR(*K, fieldpol, charpol[, tap]***)** K must be a prime field

LFSR(*p, m, n [, tap]***)**

- inputs
  - K - underlying field
  - *charpol* - LFSR defining polynomial
  - *fieldpol* - defining polynomial of the extension field (must be irreducible)
  - p - characteristic
  - m - degree of extension , ie. degree of *fieldpol*
  - n - length of LFSR , ie. degree of *charpol*
  - *tap* - OPTIONAL - the output tap, changed to the default $\mathtt{S}_0$ if the number specified falls out of the LFSR
- outputs: an empty LFSR object with 3 components: *init, state, numsteps*

LFSR in GAP

LFSR(*args***)**

> **LFSR(***K, charpol [, tap]***)**
>
> **LFSR(***K, fieldpol, charpol[, tap]***)** K must be a prime field
>
> **LFSR(***p, m, n [, tap]***)**

returns an empty LFSR object with:

three components:

- *init* - FFE vector of length $n = \deg(charpol)$ to store initial state (with indices $n-1, \ldots, 0$)
- *state* - FFE vector of length $n = \deg(charpol)$ to store current state (with indices $n-1, \ldots, 0$)
- *numsteps* - the number of steps (initialized to -1, set to 0 when loaded with call to LoadLFSR, incremented by 1 with each step StepLFSR)

components can be accessed as `obj_name!.component_name`

LFSR in GAP

LFSR(*args*)

LFSR(*K*, *charpol [, tap]***)**

LFSR(*K*, *fieldpol*, *charpol[, tap]***)** K must be a prime field

LFSR(*p*, *m*, *n [, tap]***)**

attributes set at the time of constructor call:

- `FieldPoly` - to store the defining polynomial *fieldpol* of the underlying field
- `CharPoly` - to store the LFSR polynomial *charpol*
- `FeedbackVec` - the *charpol* without its leading coefficient, stored as a FFE vector of length $n = \deg(charpol)$ to speed up the StepLFSR computation (with indices $n-1, \ldots, 0$, ie the first element of this vector corresponds to the coefficient of the term x^{n-1})
- `Length` - the length of the LFSR (which is equal to $n = \deg(charpol)$)
- `OutputTap` - the list of LFSR stages that are outputting the sequences (default is S_0, but user can choose different stage or even more then one stage, in this case the OutputTap just stores the user-defined *taps*)

properties:

- `IsLinearFeedback` - true if *charpol* is a polynomial in one indeterminate

LFSR in GAP

Other attributes and properties:

InternalStateSize(*lfsr*)

Attribute : size of the LFSR in bits ("length $\times$ width")

IsPeriodic(*lfsr*)

Property : true if the LFSR poly (`charpol`) has a nonzero constant term.

Period (*lfsr*)

Attribute : returns period of LFSR poly (`charpol`)

IsMaxSeqLFSR(*lfsr*)

Property : true if the LFSR outputs an m-sequence (LFSR poly (`charpol`) is primitive)

methods:

Display/View/Print/PrintAll(*lfsr*)

Different detail on the created LFSR:

- **Display/View** show the `CharPoly` and wheter or not the LFSR is empty
- **Print** is the same as Display/View if LFSR is empty, otherwise it also shows the values of the three components *init, state, numsteps*.
- **PrintAll** shows the characteristic, the underlying field, the values of the three components and the tap positions

NOTE: added **Print/PrintAll(*[B]*,*lfsr*)** with given basis B

NOTE: extend to show all known attributes ?

LFSR in GAP

Functionality

- **LoadLFSR(***lfsr, ist***)**
- **StepLFSR(***lfsr [, elm]***)**
- **RunLFSR(***lfsr [, ist] [, num] [, pr]***)**

Writing to a file

- **WriteAllLFSR(***output, [B,] lfsr***)**
- **WriteRunLFSR(***output, [B,] lfsr* **,** *ist***,** *num***)**
- **WriteRunLFSRTEX(***output, [B,] lfsr* **,** *ist***,** *num***)**

Drawing functions

- **Tikz_LFSR(***output***,** *lfsr***)**
- **Tikz_nLFSR(***output***,** *lfsr***)**

LFSR in GAP

LoadLFSR(*lfsr, ist***)**
Loading the LFSR *lfsr* with the initial state *ist*

- inputs
  - *lfsr* - an empty LFSR
  - *ist* - a vector of the same length as LFSR, containing FFEs that must lie in the underlying field given by `FieldPoly` (with indices $n-1,\ldots,0$)
- outputs - the first sequence element
- errors:

  Error("initial state length doesnt match")

  Error("initial state element at index i is not an element of the underlying field !!!")

LFSR in GAP

StepLFSR(*lfsr [, elm]***)**
Perform one step the LFSR *lfsr*, ie. compute the new state and update the numsteps, then output the elements denoted by `OutputTap`.

- inputs
  - *lfsr* - a loaded LFSR
  - *elm* - OPTIONAL - a FFEs that must lie in the underlying field given by `FieldPoly` to break the linearity of the feedback
- outputs
  - *seq* - the next sequence element (thats a FFE(or vector of FFEs) from the state(s) given by `OutputTap`)
- errors:

  Error(“the LFSR is NOT loaded !!!”);

LFSR in GAP

RunLFSR(*lfsr [, elm][, ist] [, num] [, pr]***)**

RunLFSR(*lfsr, num, pr* **)** - run *lfsr* for *num* steps with/without output

RunLFSR(*lfsr, num* **)** - run *lfsr* for *num* steps without output

RunLFSR(*lfsr, pr* **)** - run *lfsr* for `threshold` steps with/without output

linear versions with initial state: load then run (seq_0 is a part of the output sequence)

RunLFSR(*lfsr, ist, num, pr* **)** - load with initial state *ist* and run *lfsr* for *num* steps (output sequence will be *num* or `threshold` elements long) with/without output

RunLFSR(*lfsr, ist, num* **)** - load with initial state *ist* and run *lfsr* for *num* steps (output sequence will be *num* or `threshold` elements long) without output

RunLFSR(*lfsr, ist* **)** - load with initial state *ist* and run *lfsr* for `threshold` steps (output sequence will be `threshold` elements long) without output

LFSR in GAP

RunLFSR(*lfsr [, ist] [, num] [, pr]***)**

RunLFSR(*lfsr, num, pr* **)** - run *lfsr* for *num* steps with/without output

RunLFSR(*lfsr, num* **)** - run *lfsr* for *num* steps without output

RunLFSR(*lfsr, pr* **)** - run *lfsr* for `threshold` steps with/without output

"nonlinear" versions: with an extra element added to feedback

RunLFSR(*lfsr, elm, num, pr* **)** - run *lfsr* for *num* steps, whereby the SAME element *elm* is added to the feedback at each step, with/without output

RunLFSR(*lfsr, elm, num* **)** - run *lfsr* for *num* steps, whereby the SAME element *elm* is added to the feedback at each step, without output

"nonlinear" versions with initial state: load then run (seq_0 is a part of the output sequence)

RunLFSR(*lfsr, ist, elmvec, pr* **)** - oad with initial state *ist* and run *lfsr* for Length(*elmvec*) steps, whereby one element of *elmvec* is added to the feedback at each step (starting with elmvec[1]), with/without output

LFSR in GAP

RunLFSR(*lfsr [, ist] [, num] [, pr]***)**

- inputs
  - *lfsr* - LFSR
  - *ist* - OPTIONAL - initial state if we want to (re)load the LFSR first
  - *num* - OPTIONAL - number of steps
  - *elm* - OPTIONAL - a FFEs that must lie in the underlying field given by `FieldPoly` to break the linearity of the feedback - the SAME FFE is added at each step
  - *elmvec* - OPTIONAL - a list of FFEs that must lie in the underlying field given by `FieldPoly` to break the linearity of the feedback - different FFE is added at next step
  - *pr* - OPTIONAL - print switch
- outputs
  - sequence of length $\leq$ `threshold`
    - Length(`OutputTap`) $= 1 \Rightarrow$ sequence of FFEs : $\text{seq}_0, \text{seq}_1, \text{seq}_2, \ldots$
    - Length(`OutputTap`) $= \text{t} \Rightarrow$ sequence of vectors, each of them with t FFEs : $\text{seq}_0, \text{seq}_1, \text{seq}_2, \ldots,$ where $\text{seq}_i = (\text{seq}_{i1}, \ldots, \text{seq}_{it})$
- notes: calling StepLFSR and LoadLFSR

LFSR in GAP

WriteAllLFSR(*output, [B,] lfsr***)**
writes the (some) details about the *lfsr* to a file given by *output* (with/without basis B for representation of field elements)

example output:

```
LFSR over GF(2^1) defined by FieldPoly=x+Z(2)^0 given by CharPoly =
x^7+x^6+x^5+x^4+Z(2)^0
with feedback coeff =[ 1, 1, 1, 0, 0, 0, 1 ]
with initial state =[ 1, 0, 0, 1, 0, 1, 0 ]
with current state =[ 0, 0, 0, 0, 1, 0, 1 ]
after 19 steps
with output from stages S_[ 0, 6 ]
```

LFSR in GAP

WriteRunLFSR(*output, [B,] lfsr* , *ist*, *num***)**

writes the run (*num* steps) of the *lfsr* to a file given by *output*: it prints the characteristic polynomial followed by the stage indices and output taps, and then each step in a new line, showing both, the internal sate and the output. At the end it writes the output sequence(s) of this LFSR run.

example output:

```
Empty LFSR given by CharPoly = x^7+x^6+x^5+x^4+Z(2)^0
[ 6,..............,0 ] with taps [ 0, 6 ]
[ 1, 0, 0, 1, 0, 1, 0 ]          [ 0, 1 ]
[ 1, 1, 0, 0, 1, 0, 1 ]          [ 1, 1 ]
[ 1, 1, 1, 0, 0, 1, 0 ]          [ 0, 1 ]
[ 1, 1, 1, 1, 0, 0, 1 ]          [ 1, 1 ]
[ 0, 1, 1, 1, 1, 0, 0 ]          [ 0, 0 ]
[ 0, 0, 1, 1, 1, 1, 0 ]          [ 0, 0 ]
[ 1, 0, 0, 1, 1, 1, 1 ]          [ 1, 1 ]
[ 0, 1, 0, 0, 1, 1, 1 ]          [ 1, 0 ]
[ 0, 0, 1, 0, 0, 1, 1 ]          [ 1, 0 ]
[ 0, 0, 0, 1, 0, 0, 1 ]          [ 1, 0 ]
[ 1, 0, 0, 0, 1, 0, 0 ]          [ 0, 1 ]
The whole sequences:
seq from S_0:  0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1
seq from S_6:  1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0
```

LFSR in GAP

WriteRunLFSRTEX(*output, [B,] lfsr* **,** *ist***,** *num***)** latex equivalent of WriteRunLFSR **??**, which produces the following output directly copied into this file :

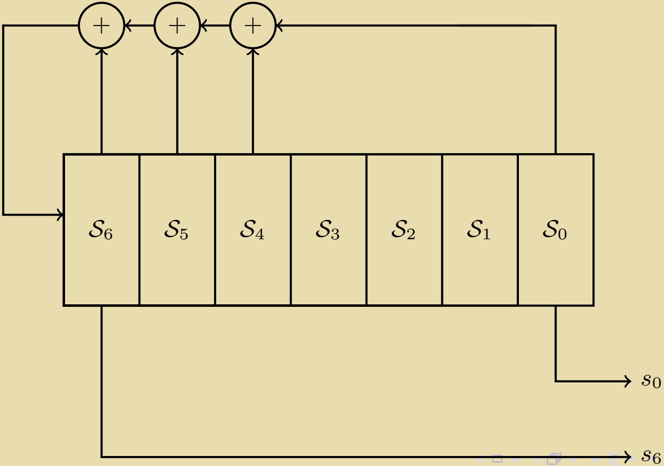
step	state							sequence	
num	$\mathcal{S}_6$	$\mathcal{S}_5$	$\mathcal{S}_4$	$\mathcal{S}_3$	$\mathcal{S}_2$	$\mathcal{S}_1$	$\mathcal{S}_0$	$\mathcal{S}_0$	$\mathcal{S}_6$
0	1	0	0	1	0	1	0	0	1
1	1	1	0	0	1	0	1	1	1
2	1	1	1	0	0	1	0	0	1
3	1	1	1	1	0	0	1	1	1
4	0	1	1	1	1	0	0	0	0
5	0	0	1	1	1	1	0	0	0
6	1	0	0	1	1	1	1	1	1
7	0	1	0	0	1	1	1	1	0
8	0	0	1	0	0	1	1	1	0
9	0	0	0	1	0	0	1	1	0

Table: LFSR with feedback $x^7 + x^6 + x^5 + x^4 + Z(2)^0$ over GF(2) !!!

The whole sequence(s):
seq from $\mathcal{S}_0$: 0101001111
seq from $\mathcal{S}_6$: 1111001000

LFSR in GAP

Tikz_LFSR(*output*, *lfsr*)
Tikz_nLFSR(*output*, *lfsr*)
creates a Tikz figure of the LFSR *lfsr* to a file given by *output*: the only difference between the two figures is that **Tikz_nLFSR** shows an element e added to the feedback.



LFSR in GAP

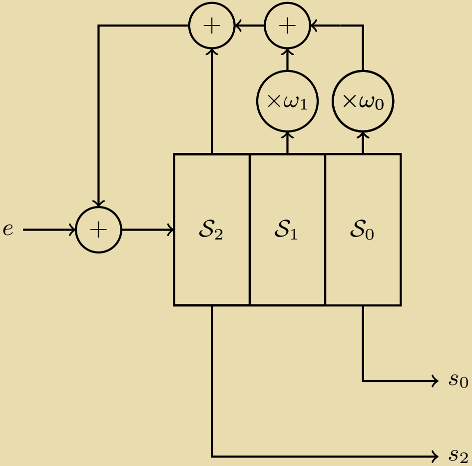


Figure: LFSR with feedback polynomial $f(x) = y^3 + y^2 + Z(2^2)^2 * y + Z(2^2)$

Part IV:
WG stream ciphers

STRUCTURE OF WG-16

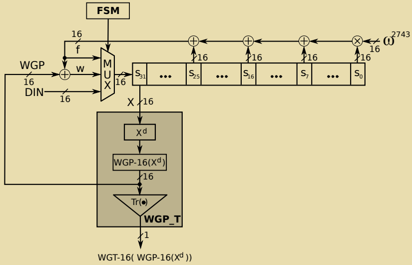
- the LFSR with feedback polynomial

$\ell(x) = x^{32} + x^{25} + x^{16} + x^{7} + \omega^{2743}$

- the WG transformation:

WGT-16(X^d) $= \mathrm{Tr}($WGP-16$(X^d))$,

where $X \in \mathbb{F}_{2^{16}}$ is the LFSR state S_{31} and WGP-16(X^d) $: \mathbb{F}_{2^{16}} \to \mathbb{F}_{2^{16}}$ the WG-16 permutation with decimation value $d = 1057$



- the FSM controlling its operation: the cipher operates in three phases

•	key/IV loading phase (**load**)	$\Rightarrow$	LFSR input: DIN
•	initialization phase (**init**)	$\Rightarrow$	LFSR input: `w = f XOR WGP`
			`WGP`=WGP-16(X^d), `f`= LFSR feedback
•	running phase (**run**)	$\Rightarrow$	LFSR input: `f`
			$k_i =$WGT-16(X^d) $= \mathrm{Tr}(\mathrm{WGP}-16(\mathrm{X}^{\mathrm{d}}))$

STRUCTURE OF WG-16

WG transformation - details:

$$Y = X^d + 1 \tag{1}$$

$$\begin{aligned} q(Y) &= Y + Y^{2^{11}+1} + Y^{2^6+2^{11}+1} \\ &\quad + Y^{2^6-2^{11}+1} + Y^{2^6+2^{11}-1} \end{aligned} \tag{2}$$

$$\text{WGP-16}(X^d) = q(Y) + 1 \tag{3}$$

$$\text{WGT-16}(X^d) = \mathrm{Tr}(\text{WGP-16}(X^d)), \tag{4}$$

from HW perspective: equations (2) and (1) $\Rightarrow$

$\Rightarrow$ **9 multiplications and 2 inversions** for one WGP-16(X^d) !!!

$$q(Y) = Y \underset{16}{\oplus} (Y^{2^{11}} \underset{16}{\otimes} Y) \underset{16}{\oplus} (Y^{2^{11}} \underset{16}{\otimes} Y^{2^6} \underset{16}{\otimes} Y) \underset{16}{\oplus} (Y^{2^6} \underset{16}{\otimes} Y^{-2^{11}} \underset{16}{\otimes} Y) \underset{16}{\oplus} (Y^{2^{11}} \underset{16}{\otimes} Y^{2^6} \underset{16}{\otimes} Y^{-1}) \tag{5}$$

with $\underset{16}{\oplus}$ representing addition in $\mathbb{F}_{2^{16}}$ and $\underset{16}{\otimes}$ multiplication in $\mathbb{F}_{2^{16}}$
Decimation can be written as

$$X^d = X^{1057} = X^{2^{10}} \underset{16}{\otimes} X^{2^5} \underset{16}{\otimes} X \tag{6}$$

Both multiplication and inversion are quite demanding in $\mathbb{F}_{2^{16}}$, which makes the module `WGP_T` the most complex part of WG-16 $\rightarrow$ pipeline!

Part V:
WG in GAP

WG in GAP

WG(*args***)**

WG(*K, fieldpol, LFSRpol[, tap])*

WG(*K, fieldpol, LFSRpol, d[, tap])* - if using decimated version

WG(*F, LFSRpol[, tap])*

WG(*F, LFSRpol, d[, tap])* - if using decimated version

WG(*p, n ,l, d[, tap])* - char, field, length of LFSR (degree of primitive poly), decimation

returns an empty WG object with three components: *decimation*, *LFSRpoly*, *OutTap* $\rightarrow$ everything that could change in the WG's lifetime is a component
and sets the following:

- FieldPoly: attr
- IsNotMmod3: prop; WGm(wg, m): attr; WGk(wg, k): attr, 3k=1 mod m
- WGexponents: attr, stores them as a list for faster computation of permutation polynomial
- IsPrimitiveFeedback: prop
- WGl = Degree(lfsrpol): attr

we do not store the actual LFSR at this point

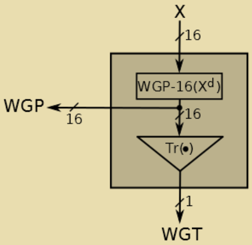
WG in GAP

WGlfsr(*wg*)
stores a component lfsr := LFSR(K, f, l, tap) (calling LFSR contructor we have seen before), but also sets additional (WG specific) filters: SetWGexponents, SetWGdecimation, SetIsRunReady(wglfsr, IsLFSR(lfsr));
LoadWG(*lfsr, ist*)): loads initial state ist into the LFSR (callin LoadLFSR)
StepWG(*lfsr [, ffe]*)): one step of LFSR (callin StepLFSR with or without ffe input)
WGP(*ffe, dec, exponents*)): using equations (2),(1) and (3) compute the WGP value of ffe
StepWGP(*lfsr*): using equations (2),(1) and (3) compute the next WGP value but DONT clock LFSR (this value is then used to call StepWG), also NOT calling the WGP value, but copy-pasted code to save time ?
WGT(*F, K, ffe, dec, exponents*)): analogue to StepWGP , only that the trace is performed as well, again not clocked (instead call StepWG without input)
StepWGT(*lfsr*): analogue to StepWGP , only that the trace is performed as well, again not clocked (instead call StepWG without input)
not yet implemented inside WG package, but “used”: **KiaWG, RunWG**

Appendix 1: Some WG-16 implementation details

WGP_T module and different field constructions

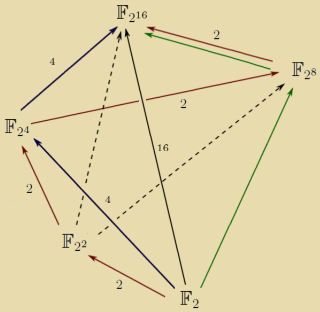
- implementation of the LFSR is straightforward
- different constructions of $\mathbb{F}_{2^{16}}$ to optimize module WGP_T
- implementation of FSM depends on the particular WGP_T implementation



Finite field $\mathbb{F}_{2^{16}}$:

$p(x) = x^{16} + x^5 + x^3 + x^2 + 1$

- $\mathbb{F}_{2^{16}}$ as an extension of degree 16 over $\mathbb{F}_2$
  - $\Rightarrow$ polynomial basis
  - $\Rightarrow$ normal basis
- $\mathbb{F}_{2^{16}}$ as a tower of extensions over $\mathbb{F}_2$
  - $\Rightarrow$ $\mathbb{F}_{(((2^2)^2)^2)^2}$
  - $\Rightarrow$ $\mathbb{F}_{(2^4)^4}$
  - $\Rightarrow$ $\mathbb{F}_{(2^8)^2}$



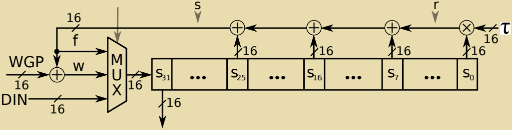
pipelining the feedback

	total matrix Hamming weight	maximum row Hamming weight	XOR tree depth
ω^{2743}	110	11	4
ω^{22363}	118	9	4

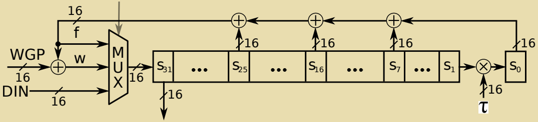
Table: Hamming weights of the old constant term $\omega^{2743}$and the new constant term ω^{22363}

$$
\begin{aligned}
\ell_1(x) &= x^{32}+x^{25}+x^{16}+x^{7}+\omega^{2743}\\
\ell_2(x) &= x^{32}+x^{25}+x^{16}+x^{7}+\omega^{22363}\\
\ell_3(x) &= x^{32}+x^{25}+x^{17}+x^{16}+x^{14}+\\
&\quad x^{13}+x^{11}+x^{10}+x^{7}+\omega^{22363}
\end{aligned}
$$

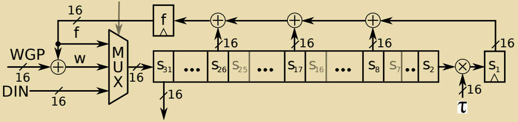
Linear Feedback Shift Registers and WG ciphers in GAP



(a) LFSR with original (un-pipelined) feedback



(b) LFSR with pipelined feedback



(c) LFSR with feedback pipelined twice

Figure: LFSR with original feedback and its pipelined versions