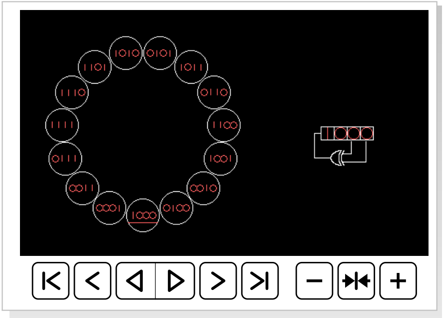



# CS 553

# CRYPTOGRAPHY

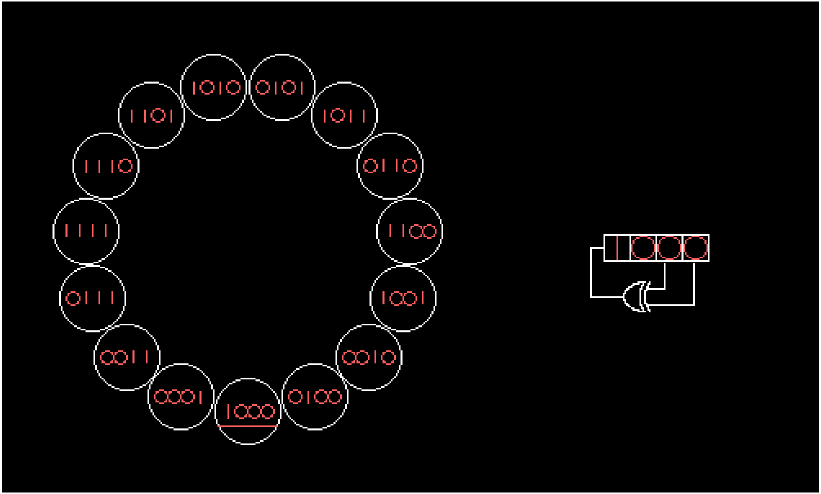
## Lecture 17

More on Stream Ciphers

Instructor
Dr. Dhiman Saha

Image Source: Google, Animation works in Acrobat

# Zooming-In



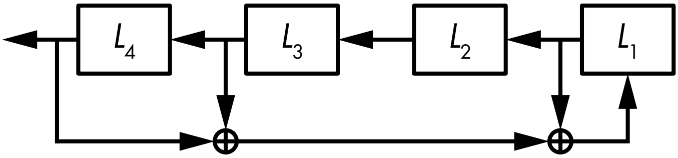

Animation works in Acrobat

# **Linear** Feedback Shift Registers

LFSR

**FSRs with a linear feedback function**

- A function thats the XOR of **some** bits of the state



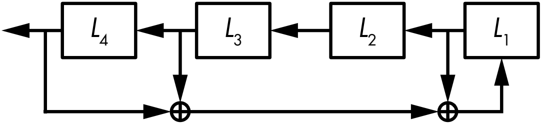
What is the cryptographic significance of the choice of the bits?

- Choice of bits is crucial for the period of the LFSR
- Signifies cryptographic value

**For $n-$bit LFSR** **Known**

Selection of the position of the bits in order to guarantee a maximal period of $2^n - 1$.

- We take the indices of the bits,
  - from 1 for the rightmost
  - to n for the leftmost
- And write the polynomial expression $1 + x + x^2 + \cdots + x^n$ , where the term $x^i$ is only included if the $i^{th}$ bit is one of the bits XORed in the feedback function.




$$1 + x + x^3 + x^4$$

The period is maximal **if and only if** that polynomial is **primitive**.

# The Theory Behind LFSRs

Galois Fields

LFSR operation equivalent to multiplication in a field.

## Math Recap — Field

A field is defined as a set with the following:

- Two operations defined on it
  - "addition" and "multiplication"
- closed under these operations
- associative and distributive laws hold
- additive and multiplicative identity elements
- additive inverse for every element
- multiplicative inverse for every non-zero element

# Fields



## Example fields

- set of rational numbers
- set of real numbers

- Is set of integers a field?

Finite fields are called Galois fields.

## Example

Binary numbers 0,1 with `XOR` as “addition” and AND as “multiplication”

- Called GF(2)

- Consider polynomials whose coefficients come from $GF(2)$
- Each term of the form $x^i$ is either **present** or **absent**

**Example**

$0, 1, x, x^2$ and $x^7 + x^6 + x^4 + 1$

**$x^7 + x^6 + x^4 + 1$**

$$1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$$

With addition and multiplication these form a field

These polynomials form a Galois (**finite**) field if we take the results of this multiplication modulo a **prime** polynomial $p(x)$.

**Definition (Prime polynomial)**

A prime polynomial is one that cannot be written as the product of two non-trivial polynomials $q(x)r(x)$

- aka Irreducible polynomials
  - meaning that it cant be **factorized**;
  - i.e., written as a product of **smaller polynomials**

**Example ($x + x^3$ is not irreducible)**

$(1 + x)(x + x^2) = x + x^2 + x^2 + x^3 = x + x^3$

For any degree, there exists at least **one** prime polynomial.

- With it we can form $GF(2^n)$. How?

Primitive Element

Every Galois field has a **primitive** element, $\alpha$, such that **all non-zero** elements of the field can be expressed as a **power** of $\alpha$.

- By raising $\alpha$ to powers (modulo $p(x)$), all non-zero field elements can be formed.

For any degree, there exists at least **one** prime polynomial.

- With it we can form $GF(2^n)$. How?

**Primitive Element**

Every Galois field has a **primitive** element, $\alpha$, such that **all non-zero** elements of the field can be expressed as a **power** of $\alpha$.

- By raising $\alpha$ to powers (modulo $p(x)$), all non-zero field elements can be formed.

Certain choices of $p(x)$ make the simple polynomial $x$ the primitive element. These polynomials are called **primitive**.

- One **exists** for every degree.

**Example ($x^4 + x + 1$ is primitive)**

- So $\alpha = x$ is a **primitive** element

Successive powers of $\alpha$ will generate
all non-zero elements of $GF(16)$

# Powers of $\alpha$ — Non-Zero Elements of $GF(16)$

$$
\begin{aligned}
\alpha^0 &= 1\\
\alpha^1 &= x\\
\alpha^2 &= x^2\\
\alpha^3 &= x^3\\
\alpha^4 &= x + 1\\
\alpha^5 &= x^2 + x\\
\alpha^6 &= x^3 + x^2\\
\alpha^7 &= x^3 + x + 1\\
\alpha^8 &= x^2 + 1\\
\alpha^9 &= x^3 + x\\
\alpha^{10} &= x^2 + x + 1\\
\alpha^{11} &= x^3 + x^2 + x\\
\alpha^{12} &= x^3 + x^2 + x + 1\\
\alpha^{13} &= x^3 + x^2 + 1\\
\alpha^{14} &= x^3 + 1\\
\alpha^{15} &= 1
\end{aligned}
$$

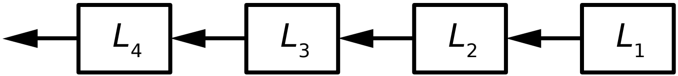
**Example**

$$
\begin{aligned}
\alpha^4 &= x^4 \mod x^4 + x + 1\\
&= x^4 \ \mathtt{xor}\ x^4 + x + 1\\
&= x + 1
\end{aligned}
$$

In general finding primitive polynomials is **difficult**.

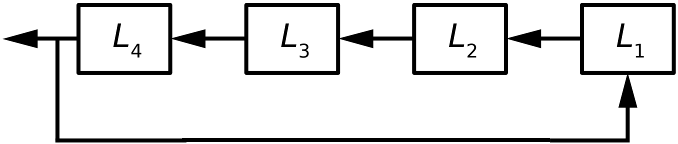
# Building LFSRs from Primitive Polynomials

# Number the cells based on the shift direction



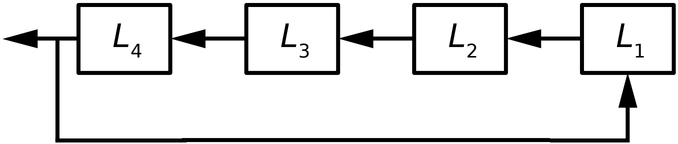

Primitive Polynomial
$x^4 + x^3 + 1$

# $x^0 = 1$ term corresponds to connecting the feedback



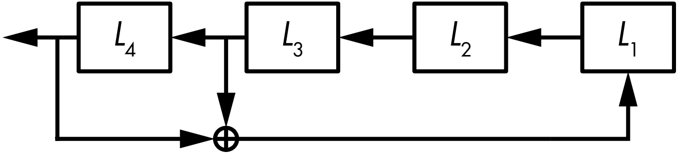

Primitive Polynomial
$x^4 + x^3 + 1$

# $x^4$ term corresponds to using current output




Primitive Polynomial
$x^4 + x^3 + 1$

# $x^i$ exists if part of XOR




Primitive Polynomial
$x^4 + x^3 + 1$

- Cross-check if the period is maximal.
- Now check the correspondence between the powers of $\alpha$ and the states of the LFSR starting from 0001

LFSR as a stream cipher is **insecure**

- If $n$ consecutive bits produced by an $n-$bits LFSR
- And the feedback polynomial associated are known
- Then we can deduce the $(n+1)^{th}$ bit produced by the register

What if feedback polynomial is not known?

The Berlekamp-Massey algorithm

### Berlekamp-Massey algorithm

It is an algorithm that will find the shortest linear feedback shift register (LFSR) for a given binary output sequence

The BerlekampMassey algorithm can be used to

- solve the equations defined by the LFSRs mathematical structure
- to find not only the LFSRs initial state but also its feedback polynomial.

UNIVERSITY OF Waterloo

Search uwaterloo.ca Search

## An Online Calculator of Berlekamp-Massey Algorithm

Berlekamp-Massey algorithm is an algorithm that will find the shortest linear feedback shift register (LFSR) for a given binary output sequence. Here we present a web-based implementation to compute the shortest LFSR and linear span of a given binary sequence. If you have any questions or suggestions, please do not hesitate to contact Bo Zhu.

Please enter the binary sequence (separated by commas):

```
0, 0, 1, 0, 1, 1, 1, 0, 1, 0 ,1
```

Press to Compute

LFSR: x^6 + x^5 + x^4 + x^2 + 1 Linear Span: 6 Time Used: 0.00 se

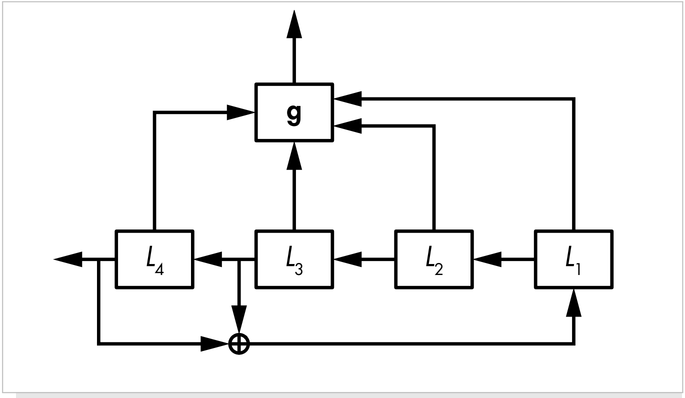
Please note the output polynomial is using the form that its degree is always equal to the linear span. For example, x^3 + x + 1 means tap positions are 0th and 1st (*not 0th and 2nd*).

If the input sequence is too long, it may take a long time to process. As a result, the Google App Engine server may cut off HTTP connections, so the final result won't be sent back. In this case, please download the Python source code from here, and run it on local computers.

University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
519 888 4567

# How to recover?

Non-linearity to the rescue!!!



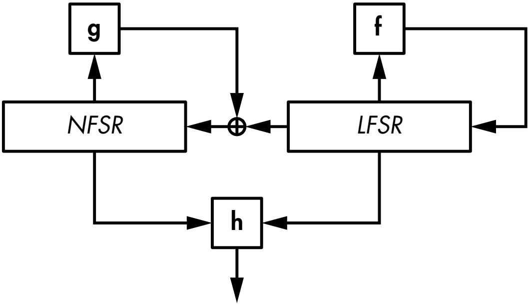

- $g$ is **non-linear**
- Both XORs bits together and combines them with logical AND or OR operations

- **Algebraic attacks** will solve the nonlinear equation systems
- **Cube attacks** will compute derivatives of the nonlinear equations
- **Fast correlation attacks** will exploit filtering functions

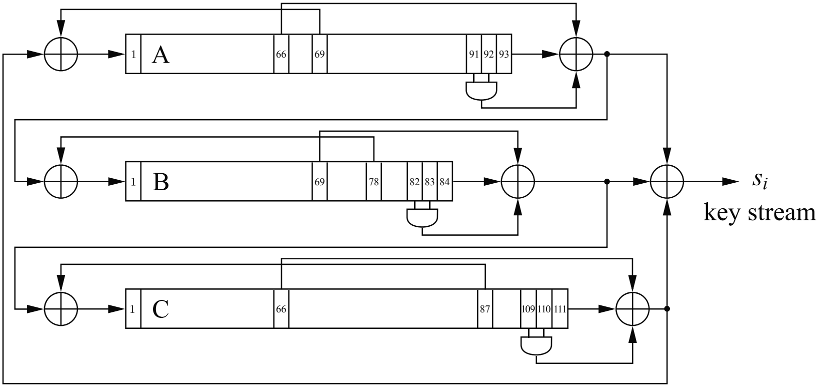
### Point-to-ponder

Patch-work does not suffice

- Need more concrete solutions : **NFSR**
- Modern Stream Ciphers


g
f
NFSR
LFSR
h

# Trivium



1
A
66
69
91
92
93
1
B
69
78
82
83
84
$s_i$
key stream
1
C
66
87
109
110
111