Classic Cryptosystems

Key Points

- Field: set of elements with + & *
- Modular Arithmetic: reduces all numbers to fixed set [0…n-1]
- GCD: largest positive integer dividing
- Finite Field: finite number of elements
- Order Finite Field: power of a prime P^n *where n = integer*
- Finite Field: of order p can be defined using normal arithmetic mod p

Modulo Operation

- Q: What is 12 mod 9?
- A: 12 mod 9 ≡3
- Let $a,r,m \in Z$
  (Z = *set of all integers*) and $m > 0$.

We write

- $r \equiv a$ mod m if m–r divides a.
- m is called the modulus.
- r is called the remainder.

$q \cdot a = m - r \qquad 0 \leq r < m$

٣/62

Ring

- Ring Z_m *is:*
  - Set of integers: $Z_m=\{0,1,2,\ldots,m-1\}$
  - Two operation: "+" and "×"
    - "+" → $a + b \equiv c \bmod m$ ($c \in Z_m$)
    - "×" → $a \times b \equiv d \bmod m$ ($d \in Z_m$)
- *Example:*
  - $m = 7$, $Z_7=\{0,1,2,3,4,5,6\}$
    - $6 + 5 = 11 \bmod 7 = 4$
    - $6 \times 5 = 30 \bmod 7 = 2$

٤/62

Ring Z_m Properties & Operations

- Identity: additive '0' , multiplicative '1'
  $a+0=a$, $a \times 1=a \bmod m$
- Inverse: additive '-a', multiplicative 'a^{-1}'
  $a+(-a)=0 \bmod m$, $a \times a^{-1} =1 \bmod m$
  Multiplicative inverse exist if gcd (a,m) = 1
- *Ring Addition and Multiplication is: Closed, Commutative, Associative*

?/62

Division on Ring Z_m

Ring Division: 4/15 mod 26???

- $4/15 \bmod 26 = 4 \times 15^{-1} \bmod 26$
- $15^{-1} \bmod 26$ exist if gcd(15,26)=1
- $15^{-1} \bmod 26 = 7$

➔ $4/15 \bmod 26 = 4 \times 7 \bmod 26 = 28 \bmod 26=2$

Note that the modulo operation can be applied whenever we want:

- (a + b) mod m = [(a mod m)+ (b mod m)] mod m
- (a × b) mod m = [(a mod m) × (b mod m)] mod m

?/62

Exponentiation in Z_m

Ring Exponentiation: $3^8 \bmod 7$ =???

- $3^8 \bmod 7 = 6561 \bmod 7$
- $6561 \bmod 7 = 2 \rightarrow 6561=(937\times7)+2$
- Or = $3^8 = 3^4\times3^4 = 3^2\times3^2\times3^2\times3^2$
- $3^8 \bmod 7 = [(3^2 \bmod 7)\times(3^2 \bmod 7)\times(3^2 \bmod 7)\times(3^2 \bmod 7)] \bmod 7$
- $3^8 \bmod 7 = (2 \times 2 \times 2 \times 2) \bmod 7 = 16 \bmod 7=2$
- Note that ring Z_m (modulo arithmetic) is of central importance to modern public-key cryptography. In practice,the order of the integers involved in PKC are in the range of[2^{160} , 2^{1024}]. Perhaps even larger

Classic Cryptography

Substitution
Transposition
Enigma Machine
Shift
Affine
Vigenere
Block (Hill)
Vernam (one time pad)
Stream

Substitution

NUCLEAR

⇩

Encryption

⇩

OVDMFBS

A	=>	B
B	=>	C
C	=>	D
D	=>	E
E	=>	F
.		.
.		.
.		.
X	=>	Y
Y	=>	Z
Z	=>	A

Key

٩/62

Substitution

NUCLEAR

⇩

Encryption

⇩

13 20 2 11 4 0 17

A	=>	0
B	=>	1
C	=>	2
D	=>	3
E	=>	4
.		.
.		.
.		.
X	=>	23
Y	=>	24
Z	=>	25

Key

١٠/62

Advance Substitution
(Random)

Key

A => D	F => I	K => N	P => S	U => G
B => A	G => J	L => O	Q => F	V => Y
C => T	H => U	M => P	R => K	W => Q
D => X	I => L	N => Z	S => V	X => E
E => H	J => R	O => M	T => W	Y => B
				Z => C

NUCLEAR → **Encryption** → ???????

١١/62

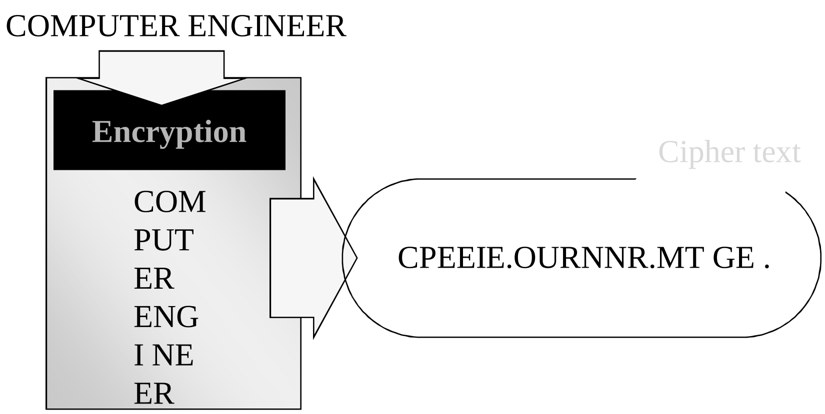
Transposition (Permutation)

Substitution reserves places
But
Transposition reserves content

NUCLEAR → **Encryption** → LUCNARE

١٢/62

Transposition (Permutation)



Security

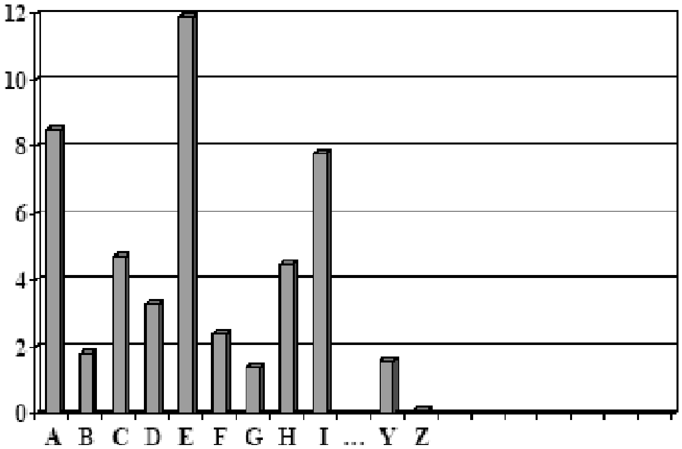
- there are n! different substitutions on an alphabet with n letters
- there are n! different transpositions of n letters
- n=26: **n!=403291461126605635584000000 = 4 . 10^{26} keys**
- trying all possibilities at 1 nanosecond per key requires....

$$4.10^{26} \ /(10^9 \ . \ 10^6 \ . \ 4 \ 10^2) = 10^9 \text{ years}$$

keys per second | seconds per day | days per year

Easy to break simple subsitution using statistical techniques

Letter distributions



Breaking a Monoalphabetic Substitution

```
X ydis pq yjc xzpvpyw ya icqdepzc ayjceq xq
A tact is the ability to describe others as

yjcw qcc yjcuqcvrcq.
they see themselves.

                    Xzexjxu Vpsdavs
                    Abraham Lincoln
```

Character Frequency: c-10, y-8, q-7, x-6, j-5, p-5, v-4, d-3
a-3, e-3, z-3, s-2, u-2, w-2, i-1, r-1

Alphabet frequency: e t a o i n s r h l d c u m f p g w y b v k x j q z

Enigma Machine

Germany- World War 1

Encryption: Keys are typed in normally

Machine output: Cipher text - encrypted message typed on paper

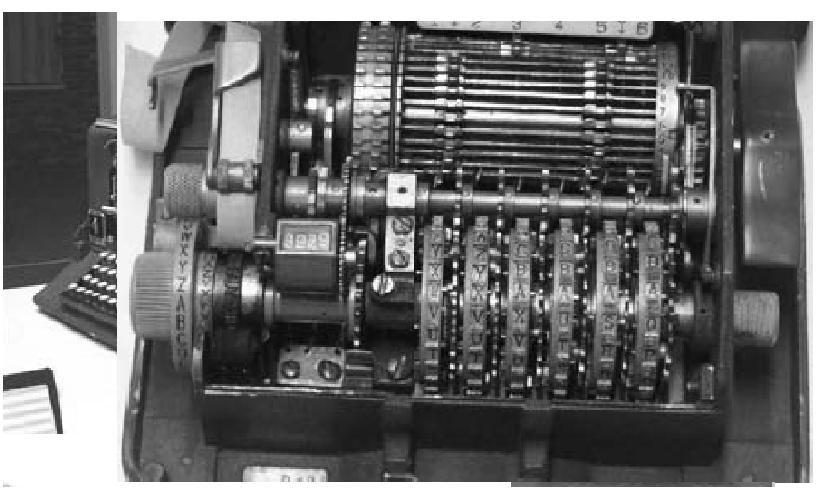
Decryption: Normal typing cipher text

- Machine output: Plain text on paper

Keys: Mechanical rotors

Wheel Cipher

Mechanical: Hagelin C38



Shift Cipher Analysis

- Alphabet letters are *substituted* by numbers:

A 0	B 1	C 2	D 3	E 4	F 5	G 6	H 7	I 8	J 9	K 10	L 11	M 12
N 13	O 14	P 15	Q 16	R 17	S 18	T 19	U 20	V 21	W 22	X 23	Y 24	Z 25

Ring: Z_{26} *x = plaintext k = key*

- $E_k(x) = x + k \mod 26$ (Encryption)
- $D_k(x) = x - k \mod 26$ (Decryption)

- Caser Cipher: k = 3

Caesar Shift

PLAINTEXT	a b c d e f g h i j k l m
CIPHERTEXT	D E F G H I J K L M N O P
PLAINTEXT	n o p q r s t u v w x y z
CIPHERTEXT	Q R S T U V W X Y Z A B C

Hello There → khoor wkhuh

Shift Cipher Example

- Assume: key $k = 17$
- Plaintext: X = A T T A C K = (0, 19, 19, 0, 2, 10).
- Ciphertext: Y = (0+17 mod 26, 19+17 mod 26,…)
- Y = (17, 10, 10, 17, 19, 1) = R K K R T B

Attacks on Shift Cipher

1. Exhaustive Search:
   - Try all possible keys. $|K|=26$.
   - Nowadays, for moderate security, $|K| \geq 280$,
   - recommended security $|K| \geq 2100$.

2. Letter frequency analysis
   (Same plaintext maps to same ciphertext)

Affine Cipher

Algorithm:

- *Encryption:* $E_k(x) = y = \alpha \cdot x + \beta \bmod 26$.
- Key: $k = (\alpha, \beta)$ where $\alpha, \beta \in Z_{26}$ *Key Space???*
- Key space = 26 . 26 = 676 Possibilities *are they all possible?*

Example:

$k = (\alpha, \beta) = (13, 4)$

- INPUT = (8, 13, 15, 20, 19)
- Y = (4, 17, 17, 4, 17) = ERRER
- ALTER = (0, 11, 19, 4, 17)
- Y = (4, 17, 17, 4, 17) = ERRER

No one-to-one map within plaintext and ciphertext.

What went wrong?

- *Decryption:* $D_k(x) = x = \alpha^{-1} \cdot y + \gamma$

Affine Cipher Analysis

Key Space:

- β can be any number in $Z_{26} \Rightarrow$ 26 possibilities
- Since α^{-1} has to exist, only selected integers in Z_{26} are useful .e.g. gcd(α, 26) = 1. ➔ {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}
- Therefore, the key space has $12 \cdot 26 = 312$ candidates.

Attack types:

1. *Ciphertext only:* exhaustive search or frequency analysis
2. *Known plaintext:* two letters in the plaintext and corresponding ciphertext letters would be sufficient to find the key.

   Example : plaintext: IF=(8, 5) and ciphertext PQ=(15, 16)
   - $8 \cdot \alpha + \beta \equiv 15 \bmod 26$
   - $5 \cdot \alpha + \beta \equiv 16 \bmod 26$ $\rightarrow \alpha = 17$ and $\beta = 9$

What happens if we have only one letter of known plaintext?

3. *Chosen plaintext:* Chose A and B as the plaintext. The first character of the ciphertext will be equal to $0{\cdot}\alpha + \beta = \beta$ and the second will be $\alpha + \beta$.
4. *Chosen ciphertext :* Chose A and B as the ciphertext.

Vigenere Cipher

- *Vigenere Cipher* encrypts *m* alphabetic characters at a time
- each plaintext element is equivalent to *m* alphabetic characters
- key *K* is a *keyword* that associate with an alphabetic string of length *m*

Example

- m = 5; K = (2, 8, 15, 7, 20).
- P = 4,5,2,8,11,2,14,20,1,2,4,5,16
- Encryption:

4	5	2	8	11	2	14	20	1	2	3	4	5	16
2	8	15	7	20	2	8	15	7	20	2	8	15	7
6	13	17	15	31	4	22	9	8	22	5	12	20	23

٢٥/62

Vigenere Cipher Secrecy

- number of possible keywords of length m ➔ 26^m
- if m = 5, then the keyspace has size exceeding 1.1×10^7.
- This is already large enough to preclude exhaustive key search by hand (but not by computer).
- having keyword length m, an alphabetic character can be mapped to one of m possible alphabetic characters (assuming that the keyword contains m distinct characters).
- Such a cryptosystem is called *polyalphabetic*.
- In general, cryptanalysis is more difficult for polyalphabetic than for monoalphabetic cryptosystems.

٢٦/62

Vigenere Cipher Attack

- observe two identical segments in Ciphertext each of length at least three, then there is a good chance that they do correspond to identical segments of plaintext.

Block ciphers

Substitution ciphers: changing one letter in the plaintext changes exactly one letter in the ciphertext.

- This greatly facilitates finding the key using frequency analysis.

Block ciphers: prevents this by encrypting a block of letters simultaneously.

- Many of the modern (symmetric) cryptosystems are block ciphers.
- DES operates on 64 bits of blocks
- AES uses blocks of 128 bits (192 and 256 are also possible).

Example: Hill Cipher (1929)

- The key is an $n \times n$ matrix whose entries are integers in Z_{26}.

Block cipher: Hill cipher

- Encryption: vector-matrix multiplication

- **Example:** Let n=3, key matrix *'M'* be assume the plaintext is ABC=(0,1,2)

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 11 & 9 & 8 \end{pmatrix}$$

$$(0,1,2) \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 11 & 9 & 8 \end{pmatrix} \equiv (26,23,22) \bmod 26 = (0,23,22) \Rightarrow AXW(ciphertext)$$

Decryption:

$$\begin{pmatrix} 22 & 5 & 1 \end{pmatrix}$$

$$(0,23,22) \times \begin{pmatrix} 22 & 5 & 1 \\ 6 & 17 & 24 \\ 15 & 13 & 1 \end{pmatrix} \equiv (468,677,574) \bmod 26 = (0,1,2) \Rightarrow ABC(plain-text)$$

Hill Cipher

- If we change one letter in the plaintext, all the letters of the ciphertext will be affected.

Example:

- Let the plaintext be ABB instead of ABC then the ciphertext is

$$(0,1,1) \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 11 & 9 & 8 \end{pmatrix} \equiv (15,14,14) \bmod 26 = (15,14,14) \Rightarrow POO(ciphertext)$$

Another Example

- Use Key:

$$M = \begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

- Decryption Key:

$$N = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$$

Hill Cipher Attack

- Ciphertext:
  - Hill Cipher is more difficult to break with a ciphertext-only attack.

- Plaintext + Ciphertext:
  1. Opponent has determined the value of *m*
  2. Compute the key

Properties of Good Cryptosystems

- **Diffusion:** one character change in the plaintext should effect as many ciphertext characters as possible.
- **Confusion:** The key should not relate to the ciphertext in a simple way.

Shannon (1949)

One-Time Pad (Vernam Cipher)

- Vernam in 1918, proposed the one-time pad, which is a provably secure cryptosystem.
- Messages are represented as a binary string (a sequence of 0's and 1's using a coding mechanism such as ASCII coding.)
- The key is a truly random sequence of 0's and 1's of the same length as the message.
- The encryption is done by adding the key to the message modulo 2, bit by bit as *exclusive OR,* $\oplus$ (XOR).

One-time pad

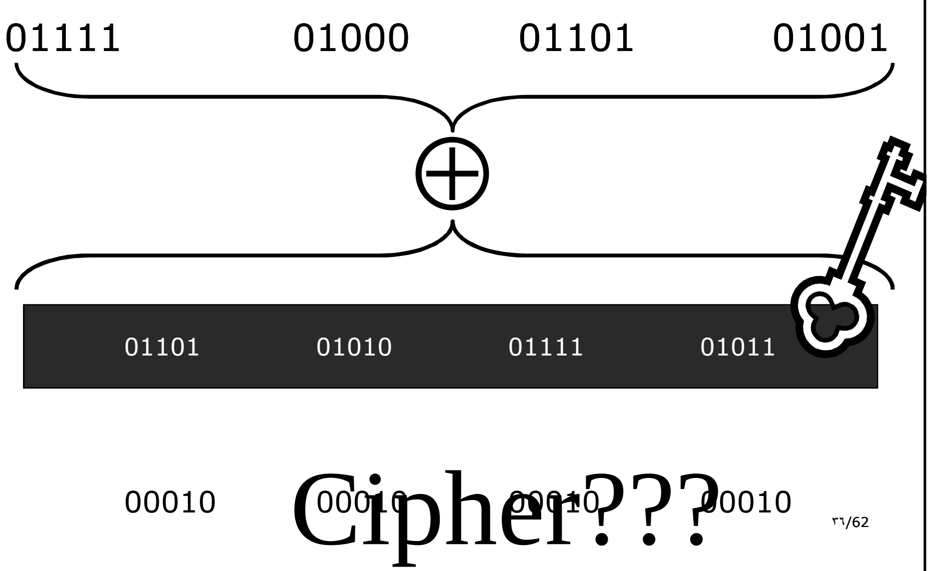
- Secret-key encryption scheme (symmetric)
  - Encrypt plaintext by XOR with sequence of bits
  - Decrypt ciphertext by XOR with same bit sequence
- Scheme for pad of length *n*
  - Set P of plaintexts: all n-bit sequences
  - Set C of ciphertexts: all n-bit sequences
  - Set K of keys: all n-bit sequences
  - Encryption and decryption functions

encrypt(key, text) = key $\oplus$ text (bit-by-bit)

decrypt(key, text) = key $\oplus$ text (bit-by-bit)

٣٥/62

Unconditional Secure



٣٦/62

Vernam scheme: perfect secrecy

- general: C = (P + K) mod 26; P = (C - K) mod 26
  - with C, P, K ∈ [0,25]; A=0, B=1, …, Z=25
- consider ciphertext C= XHGRQ
  - with key AAAAA   P = XHGRQ
  - with key VAYEK   P = CHINA
  - with key EZANZ   P = TIGER
  - …
  - with key ZZZZZ   P = YIHSR
- conclusion: for every 5-character plaintext there is a 5-character key which maps the ciphertext to that plaintext

Evaluation of one-time pad

- Advantages
  - Easy to compute encrypt, decrypt from key, text
  - As hard to break as possible
    - This is an information-theoretically secure cipher
    - Given ciphertext, all possible plaintexts are equally likely, assuming that key is chosen randomly
- Disadvantage
  - Key is as long as the plaintext
    - How does sender get key to receiver securely?

Idea for stream cipher: use pseudo-random generators for key...

۳۸/62

Randomness & Pseudo-randomness

Randomness: Closely related to *unpredictability*

Pseudo-randomness : PR sequences appears random to a computationally bounded adversary

Cryptosystems need random unpredictable numbers for

- One-time pad
- Secret key for DES, AES, etc.
- Primes *p*, *q* for RSA
- Private key for ECC
- Challenges used in challenge based identification systems

True random number generation (RNG)

Requires a naturally occurring source of randomness (randomness exists in the nature)

- Hardware based random number generators (RNG) exploit the randomness which occurs in some physical phenomena
  - Elapsed time between emission of particles during radioactive decay
  - Thermal noise from a semiconductor diode or resistor
  - Frequency instability of a free running oscillator
  - The amount which a metal insulator semiconductor capacitor is charged during a fixed period of time.
- The first two are subject to observation and manipulation by adversaries.

Software base RNG

1. The system clock
2. Elapsed time between keystrokes or mouse movement
3. Content of input/output buffer
4. User input
5. OS values such as system load and network statistics.

- All of them are subject to observation and manipulation.
- Individually these sources are very “weak”.
- The randomness can be increased by combining the outputs of these sources using a complex mixing function
  (e.g. hashing the concatenation of the output bits).

- Still, not quite secure!

٤١/62

Pseudorandom number generation

- A pseudorandom number generator (PRNG) is a deterministic algorithm, which, given a truly random binary sequence of length k (*random seed*), outputs a binary sequence of length $l >> k$ which “appears” to be random.
- The output of a PRNG is not random. However, it is impractical (improbable) for a anyone (adversary) to distinguish a pseudorandom sequence from a truly random sequence of the same length.
- No practical test to check if a sequence is truly random.
- Thus, we can’t define exactly what the pseudo randomness.
- Golomb’s postulates was one of the first attempt to establish necessary conditions for a periodic sequence to look random. It has only historical importance nowadays.
- However, more recent attempts may not offer a more thorough conditions.

٤٢/62

Statistical Tests for Pseudo-randomness

1. Frequency test (mono bit test):
- # of 1s and 0s must be approximately the same

2. Poker test
- A sequence is divided into *k* non-overlapping segments of length *m*.
- This test determines if the segments of length *m* each appear approximately the same number of times.

3. Runs Test
- Determines if the # of runs of various lengths is similar to those of truly random sequences

4. Long run test
- The long run test is passed if there are no runs of length 34 or more.

٤٣/62

Stream Ciphers

٤٤/62

Basic Idea

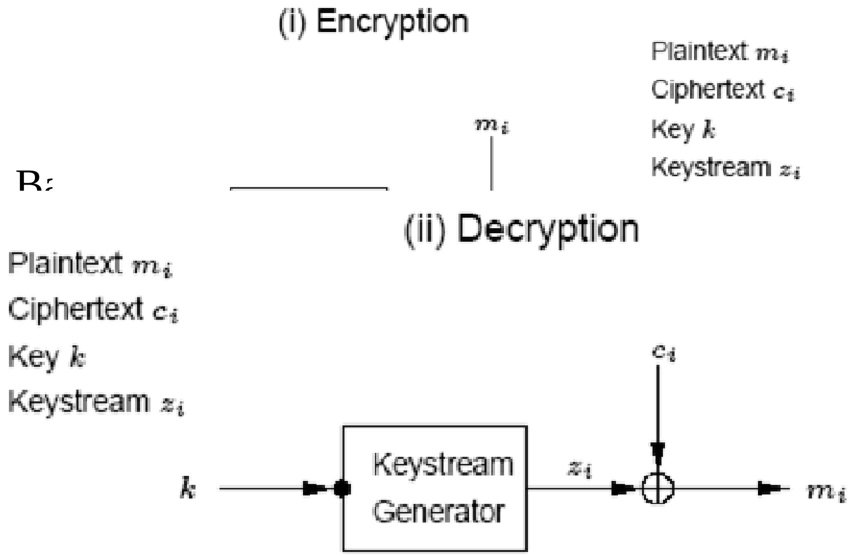
- Block ciphers: $y = y_1 y_2 y_3 = E_K(x_1)\ E_K(x_2)\ E_K(x_3)$
- Stream cipher: $y = y_1 y_2 y_3 = E_{z1}(x_1) E_{z2}(x_2) E_{z3}(x_3)$
- Stream cipher Key: $z_i = f\,(K, x_1\,, x_2)$
- block cipher can be a special case of a stream cipher where the key-stream is constant

٤٥/62

Binary Stream

- Stream ciphers are often described in terms of binary alphabets
- the encryption and decryption operation are just addition modulo 2
- exclusive-or operation: XOR ‘ $\oplus$ ’
- implemented very efficiently in hardware

٤٦/62



- Decryption : $m_i = c_i \oplus z_i$ $i = 1,2,3,4\ldots..$

٤٧/62

Stream Cipher

- **Drawback :**
  - Key-stream should be as long as plain-text.
  - Key distribution & Management difficult.
- **Solution :**
  - Stream Ciphers (in which key-stream is generated in pseudo-random fashion from relatively short *secret key*.)

٤٨/62

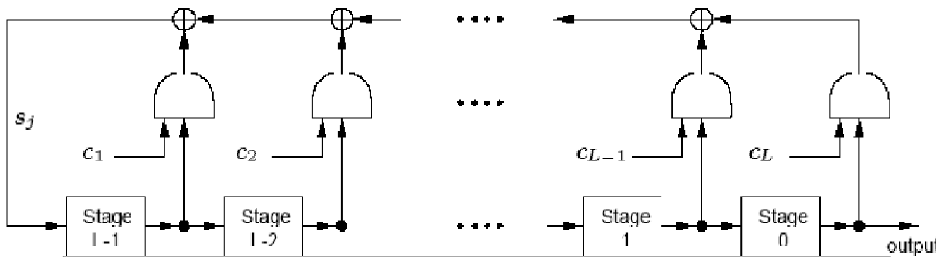
Stream ciphers

- **Randomness :**
  - Closely related to *unpredictability*.
- **Pseudo-randomness :**
  - PR sequences appears random to a computationally bounded adversary.
  - Stream Ciphers can be modeled as Finite-state machine.

Linear Feedback Shift Register (LFSR)

- Well-suited for hardware implementation
- Very low implementation costs
- Produce sequences:
  - having large periods
  - having good statistical properties
  - readily analyzed using algebraic techniques
- But, the output sequences of LFSRs are easily predictable.

Linear Feedback Shift Register (LFSR)



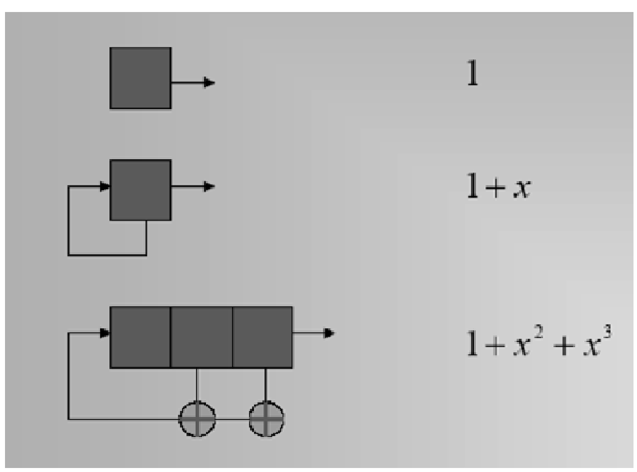
s_j=(c_1.Stage L-1)⊕(c_1.Stage L-2)…. (c_{L-1}.Stage 1)⊕(c_L.Stage 0)

Connection Polynomial: $C(x)=$

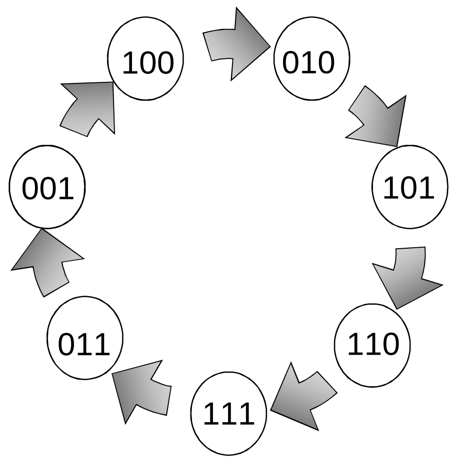
$$1+c_1 x + c_2 x^2 + c_3 x^3 + \ldots\ldots + c_L x^L$$

If $C(x)$ is chosen carefully the output of LFSR can have maximum period of 2^L-1

LFSR Connection Polynomial Generation



LFSR
Example: $C(x) = x^3 + x^2 + 1$

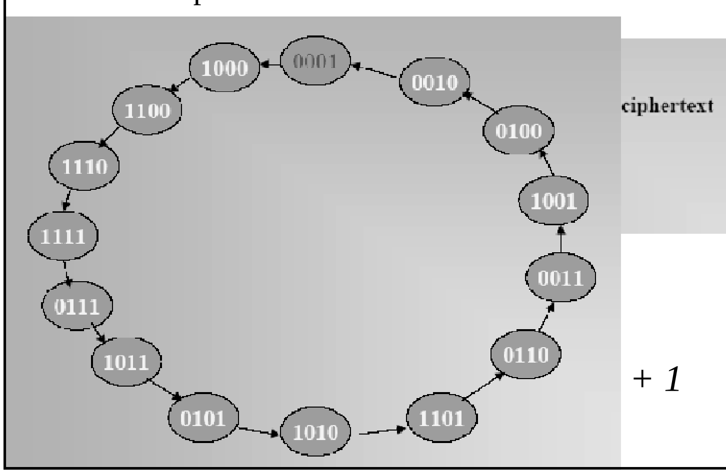


Compare with: $C(x) = x^3 + x + 1$

LFSR Example

when initial state is (0001)

LSFR Output: 100011110101100 100011110101100 …



+ 1

LFSR

- *LFSR* have good statistical properties.
- However, they may be predictable

Caveat/Warning:

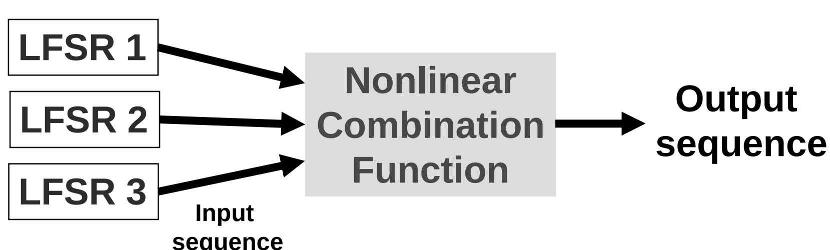
- Mathematical proofs of security of such generators are not known.
- They are deemed to be computationally secure after having withstood sufficient public scrutiny and inspection.

٥٥/62

Nonlinear Combination Generator

Combiner function must be

- Balanced
- highly nonlinear
- carefully selected ➔ no dependence between any subset of LFSR sequences and the output sequence



٥٦/62

Example: Geffe generator

$F(x_1,x_2,x_3)=x_1x_2 \oplus x_2x_3 \oplus x_3$

- inspect the truth table of the combiner function to gain more insight about the security of Geffe generator.
- The combiner function is balanced
- However, the correlation of z to
  - x_1 is $P(z=x_1) = 3/4$
  - x_2 is $P(z=x_2) = 1/2$
  - x_3 is $P(z=x_3) = 3/4$

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Geffe Generator Example Study

- LFSR#1: $1+x+x^4$. Initial key1: 1000
- LFSR#2: $1+x+x^3$. Initial key2: 110
- LFSR#3: $1+x^2+x^5$. Initial key3: 10101

- Key sequence 1 (x_1): 100011110101100
- Key sequence 2 (x_2): 010011101001110
- Key sequence 3 (x_3): 101011101100011
- Output sequence (z): 101011100101101

- Exhaustive search: 15×7×31= 3255

Correlation Attack

- If we have *n* LFSRs, the key space of the non-linear combination generator is the product of their non-repetitive shortest sequence terms.

  Exhaustive search = Brute force attack: 15×7×31= 3255 trial

- However, if there is correlation between the output sequence and each input sequence then the *effective* key length can be reduced to the summation of their non-repetitive shortest sequence terms.

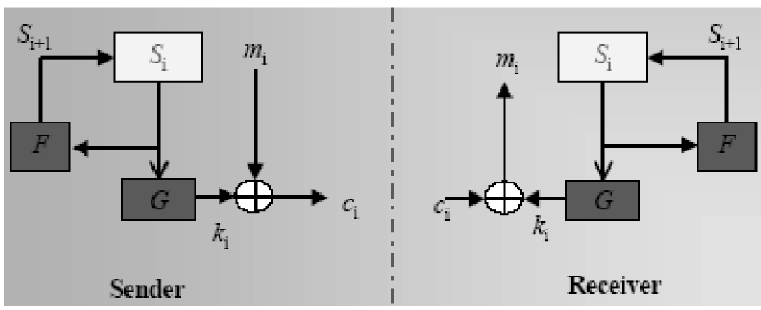
  Correlation attack: 15+7+31 = 53 trial

Correlation Attack

z	1	0	1	0	1	1	1	0	0	1	0	1	1	0	1	
$x^{(1)}_1$	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	8/15
$x^{(2)}_1$	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	8/15
$x^{(3)}_1$	1	0	1	0	1	1	0	0	1	0	0	0	1	1	1	10/15
$x^{(4)}_1$	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	6/15
$x^{(5)}_1$	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0	8/15
$x^{(6)}_1$	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	10/15
$x^{(7)}_1$	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	6/15
$x^{(8)}_1$	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	6/15
$x^{(9)}_1$	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	8/15
$x^{(10)}_1$	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	4/15
$x^{(11)}_1$	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	**12/15**
$x^{(12)}_1$	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	7/15

Synchronous Stream Ciphers

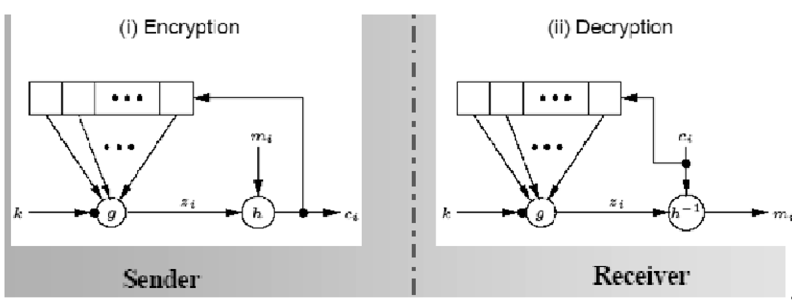
- Key-stream is independent of plain and cipher-text.
- Both sender &receiver must be synchronized.
- Resynchronization can be needed.
- Active attacks can easily be detected. (insertion, deletion, replay)
- No Error Propagation.



٦١/62

Self-Synchronizing (Asynchronous) Stream Ciphers

- key stream generated as function of fixed number of previous ciphertext bits
- Active attacks cannot be detected.
- At most *t* bits later than synchronization is lost, it resynchronizes itself
- Limited error propagation (up to *t* bits).



٦٢/62

SEAL (just an idea)

- SEAL (Software-optimized Encryption Algorithm) is a binary additive stream cipher (proposed 1993)
- specifically designed for efficient software implementation for 32-bit processors
- it has not yet received much scrutiny from the cryptographic community