

UNIVERSITY OF MASSACHUSETTS
Dept. of Electrical & Computer Engineering

Introduction to Cryptography
ECE 597XX/697XX

Part 2

Stream Ciphers

Israel Koren

ECE597/697 Koren Part.2 .1

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

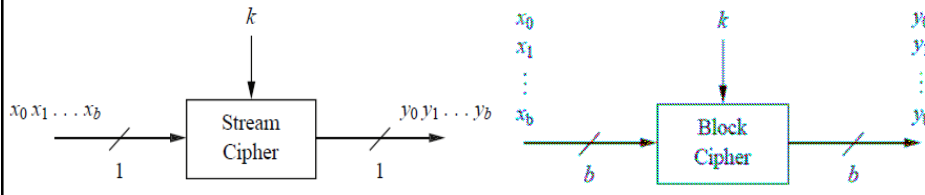
Content of this part

- ♦ **Intro to stream ciphers**
- ♦ **Random number generators (RNGs)**
- ♦ **One-Time Pad (OTP)**
- ♦ **Linear feedback shift registers (LFSRs)**
- ♦ **Trivium: a modern stream cipher**

ECE597/697 Koren Part.2 .2

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Stream Cipher vs. Block Cipher



- **Stream Ciphers**

- Encrypt bits individually
- Usually small and fast → common in embedded devices (e.g., A5/1 for GSM phones)

- **Block Ciphers:**

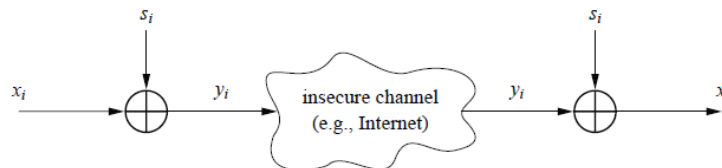
- Always encrypt a full block (several bits)
- Are common for Internet applications

ECE597/697 Koren Part.2 .3

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Encryption/decryption with Stream Ciphers

Plaintext x_i , ciphertext y_i and key stream s_i consist of individual bits

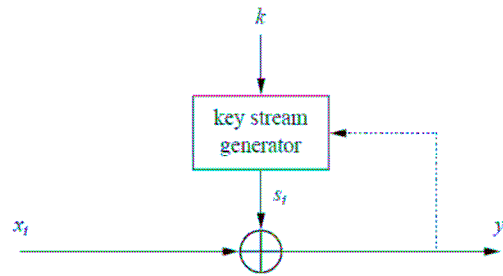


- ♦ Encryption and decryption are simple additions modulo 2 (XOR)
- ♦ Encryption and decryption are the same functions
- ♦ **Encryption:** $y_i = e_{s_i}(x_i) = x_i + s_i \bmod 2$
 $x_i, y_i, s_i \in \{0, 1\}$
- ♦ **Decryption:** $x_i = e_{s_i}(y_i) = y_i + s_i \bmod 2$

ECE597/697 Koren Part.2 .4

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Synchronous vs. Asynchronous Stream Cipher



- Security of stream cipher depends entirely on the key stream s_i :
 - Should be **random**, i.e., $\Pr(s_i = 0) = \Pr(s_i = 1) = 0.5$
 - Must be **reproducible** by sender and receiver
- **Synchronous Stream Cipher**
 - Key stream depends only on the key (and possibly an initialization vector IV)
- **Asynchronous Stream Ciphers**
 - Key stream depends also on the ciphertext (feedback enabled)

ECE597/697 Koren Part.2 .5

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Why is Modulo 2 Addition a Good Encryption Function?

- ♦ For perfectly random key stream s_i , each ciphertext output bit has a 50% chance to be 0 or 1
→ Good statistical property for ciphertext
- ♦ Inverting XOR is simple, since it is the same XOR operation

x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

ECE597/697 Koren Part.2 .6

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Stream Cipher: Throughput

Performance comparison of symmetric ciphers (Pentium4):

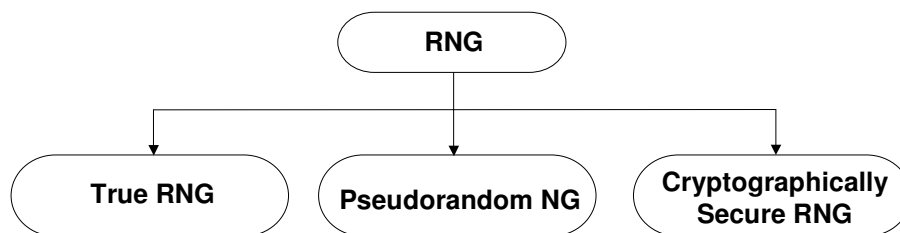
Cipher	Key length	Mbit/s
DES	56	36.95
3DES	112	13.32
AES	128	51.19
RC4 (stream cipher)	(variable)	211.34

Source: Zhao et al., Anatomy and Performance of SSL Processing, ISPASS 2005

ECE597/697 Koren Part.2 .7

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Random number generators (RNGs)



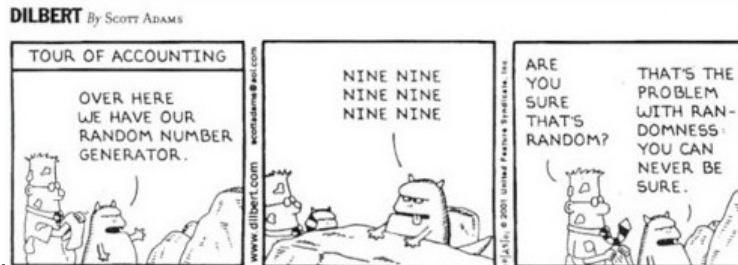
ECE597/697 Koren Part.2 .8

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

True Random Number Generators (TRNGs)

- ◆ Based on physical random processes: coin flipping, dice rolling, semiconductor noise, radioactive decay, clock jitter of digital circuits
- ◆ Output stream s_i should have good statistical properties:
 $\Pr(s_i = 0) = \Pr(s_i = 1) = 50\%$ (often achieved by post-processing)
- ◆ Output can neither be predicted nor be reproduced

Typically used for generation of keys, nonces (used only-once values) and for many other purposes



ECE597/697 Koren Part.2 .9

Pseudo-random Number Generator (PRNG)

- ◆ Generate sequences from initial seed value
- ◆ Typically, output stream has good statistical properties
- ◆ Output can be reproduced and can be predicted

Often computed in a recursive way:

$$s_0 = seed$$

$$s_{i+1} = f(s_i, s_{i-1}, \dots, s_{i-t})$$

Example: *rand()* function in ANSI C:

$s_0 = 12345$

$$s_{i+1} = 1103515245s_i + 12345 \bmod 2^{31}$$

Most PRNGs have poor cryptographic properties

ECE597/697 Koren Part.2 .10

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Cryptanalyzing a Simple PRNG

Simple PRNG: Linear Congruential Generator

$$S_0 = \text{seed}$$

$$S_{i+1} = AS_i + B \bmod m$$

Assume

- unknown A , B and S_0 as key
- Size of A , B and S_i = 100 bit
- 300 bit of output are known, i.e., S_1 , S_2 and S_3

Solving

$$S_2 = AS_1 + B \bmod m$$

$$S_3 = AS_2 + B \bmod m$$

directly reveals A and B . All S_i can be computed easily

Bad cryptographic properties due to the linearity of most PRNGs

ECE597/697 Koren Part.2 .11

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

♦ Special PRNG with additional property:

- Output must be unpredictable

More precisely: Given n consecutive bits of output s_i the following output bits s_{n+1} cannot be predicted (in polynomial time)

♦ Needed in cryptography, in particular for stream ciphers

♦ Remark: There are almost no other applications that need unpredictability, whereas many (technical) applications need PRNGs

ECE597/697 Koren Part.2 .12

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

♦ A cryptosystem is unconditionally secure if it cannot be broken even with *infinite* computational resources

♦ One-Time Pad

- A cryptosystem developed by Mauborgne that is based on Vernam's stream cipher

♦ Properties:

Let the plaintext, ciphertext and key consist of individual bits $x_i, y_i, k_i \in \{0,1\}$.

$$\begin{aligned} \text{Encryption: } e_k(x_i) &= x_i \oplus k_i \\ \text{Decryption: } d_k(y_i) &= y_i \oplus k_i \end{aligned}$$

OTP is unconditionally secure if and only if the key k_i is only used once

ECE597/697 Koren Part.2 .13

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

$$\begin{aligned} y_0 &= x_0 \oplus k_0 \\ y_1 &= x_1 \oplus k_1 \\ &\vdots \end{aligned}$$

Every equation is a linear equation with two unknowns

⇒ for every y_i : $x_i = 0$ and $x_i = 1$ are equiprobable

⇒ This is true iff $k_0, k_1, \dots$ are independent, i.e., all k_i have to be generated truly random

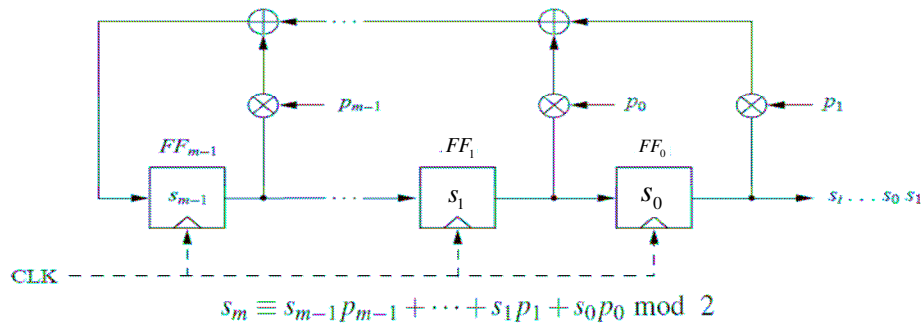
⇒ It can be shown that this systems can *provably* not be solved.

Disadvantage: For almost all applications the OTP is impractical since the key must be as long as the message! (Imagine you have to encrypt a 1GByte email attachment.)

ECE597/697 Koren Part.2 .14

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Linear Feedback Shift Registers (LFSRs)

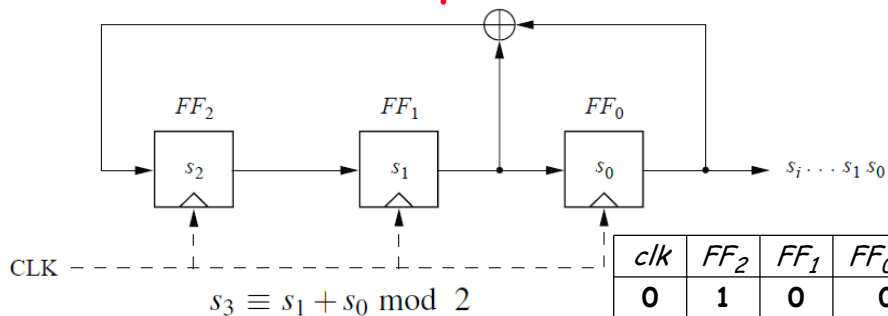


- ◆ A shift register with a feedback path computing new value by XOR of certain state bits
- ◆ Degree m given by number of storage elements
- ◆ If $p_i = 1$, the feedback connection is present ("closed" switch), otherwise there is no feedback
- ◆ Output sequence repeats periodically
- ◆ Maximum output length: $2^m - 1$

ECE597/697 Koren Part.2 .15

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

LFSR: Example with $m=3$



- ◆ LFSR output described by recursive equation:

$$s_{i+3} = s_{i+1} + s_i \pmod{2}$$

- ◆ Maximum output length (of $2^3 - 1 = 7$) achieved only for certain feedback configurations, e.g., the one shown here.

clk	FF ₂	FF ₁	FF ₀ =s _i
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

ECE597/697 Koren Part.2 .16

Adapted from Paar & Pelzl, "Understanding Cryptography"

Polynomial representation of LFSRs

LFSRs typically described by polynomials:

$$P(X) = X^m + p_{m-1}X^{m-1} + \dots + p_1X + p_0$$

Degree 3 LFSR from previous slide: $P(X) = X^3 + X + 1$

♦ Not all LFSR generate maximum length sequence

♦ Example: $m=3$, $p_2 = p_1 = p_0 = 1$:

♦ LFSR has maximum length *iff* the polynomial $P(x)$ is a primitive polynomial, special kind of irreducible polynomial - divisible only by itself and 1

♦ A list of known primitive polynomials is available

clk	FF_2	FF_1	FF_0
0	1	0	0
1	1	1	0
2	0	1	1
3	0	0	1
4	1	0	0
5	1	1	0
6	0	1	1

ECE597/697 Koren Part.2 .17

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Security of LFSRs

♦ If an LFSR is used to generate the stream of bits s_i so that $y_i = x_i + s_i \text{ mod } 2 = x_i \oplus s_i$

• The coefficients p_i can constitute the secret key

♦ If $2m$ output bits of an LFSR of degree m are known, the feedback coefficients p_i of the LFSR can be found by solving a system of linear equations

• Assume Oscar has the plaintext x_i and the ciphertext y_i and knows the degree m (not necessary)

• $s_i = x_i \oplus y_i$ and $s_m \equiv p_{m-1}s_{m-1} + \dots + p_1s_1 + p_0s_0$ for $m, m+1, \dots, 2m$

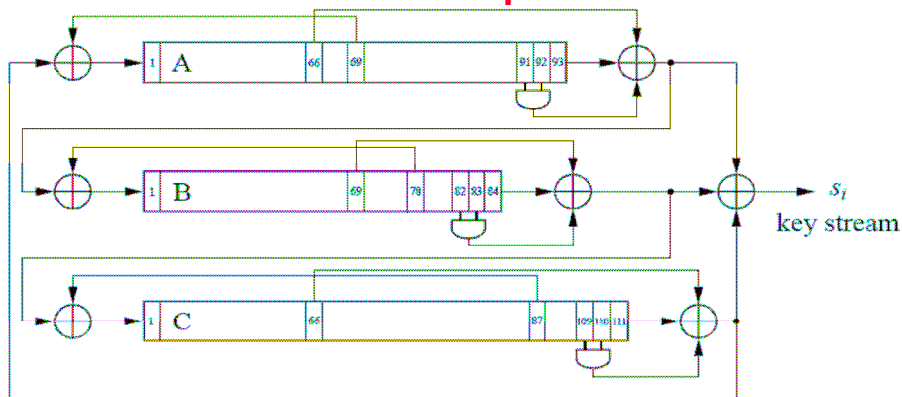
♦ Because of this many stream ciphers use combinations of LFSRs

• A5/1 for GSM phones uses 3 LFSRs

ECE597/697 Koren Part.2 .18

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

A Modern Stream Cipher - Trivium



- ◆ Three *nonlinear* FSRs (NLFSR) of length 93, 84, 111
- ◆ XOR-Sum of all three NLFSR outputs generates key stream s_i
- ◆ Small hardware:
 - Total FF count: 288; Non-linearity: 3 AND Gates
 - 7 XOR Gates (4 with three inputs)

ECE597/697 Koren Part.2 .19

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

Trivium

Initialization:

- ◆ Load 80-bit IV into A
- ◆ Load 80-bit key into B
- ◆ $c_{109}, c_{110}, c_{111} = 1$, other=0

Warm-Up: Clock 4x288

=1152 times without generating output

IV: a randomizer with a new value for every encryption (nonces - values used only once); no need to keep secret

Without IV the sequence s_i will be deterministic

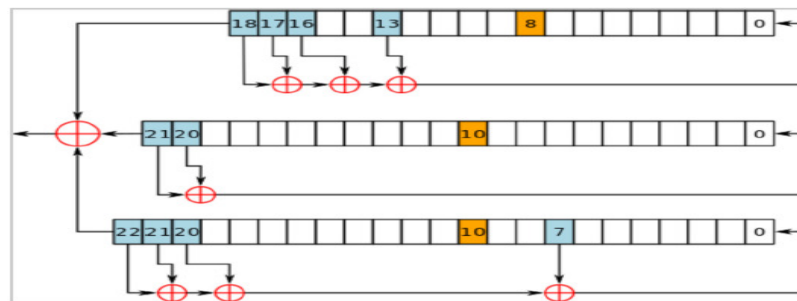
It is yet unknown how secure is Trivium

	Register length	Feedback bit	Feedforward bit	AND inputs
A	93	69	66	91, 92
B	84	78	69	82, 83
C	111	87	66	109, 110

ECE597/697 Koren Part.2 .20

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources

A5/1 Stream Cipher



LFSR number	Length in bits	Feedback polynomial	Clocking bit	Tapped bits
1	19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	8	13, 16, 17, 18
2	22	$x^{22} + x^{21} + 1$	10	20, 21
3	23	$x^{23} + x^{22} + x^{21} + x^8 + 1$	10	7, 20, 21, 22

A register is clocked if its Clocking bit is equal to at least another clocking bit

ECE597/697 Koren Part.2 .21

Copyright 2016 Koren UMass

Lessons Learned

- ♦ Stream ciphers are less popular than block ciphers in most domains such as Internet security. There are exceptions, for instance, the popular stream cipher RC4.
- ♦ Stream ciphers sometimes require fewer resources, e.g., code size or chip area, for implementation than block ciphers, and they are attractive for use in constrained environments such as cell phones.
- ♦ The requirements for a *cryptographically secure pseudorandom number generator* are far more demanding than the requirements for pseudorandom number generators used in other applications such as testing or simulation
- ♦ The One-Time Pad is a provable secure symmetric cipher. However, it is highly impractical for most applications because the key length has to equal the message length.
- ♦ Single LFSRs make poor stream ciphers despite their good statistical properties. However, careful combinations of several LFSR can yield strong ciphers.

ECE597/697 Koren Part.2 .22

Adapted from Paar & Pelzl, "Understanding Cryptography," and other sources