

# COS 126

General Computer Science  
Fall 2011

Robert Sedgewick

What is COS 126? Broad, but technical, introduction to computer science.

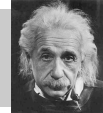
Goals.

- Demystify computer systems.
- Empower you to exploit available technology.
- Build awareness of substantial intellectual underpinnings.

Topics.

- Programming in Java.
- Machine architecture.
- Theory of computation.
- Applications to science, engineering, and commercial computing.

“Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate, and brilliant; together they are powerful beyond imagination.” – Albert Einstein



## The Basics

■ Lectures. [Sedgewick]

■ RS office hours. ← everyone needs to meet me!

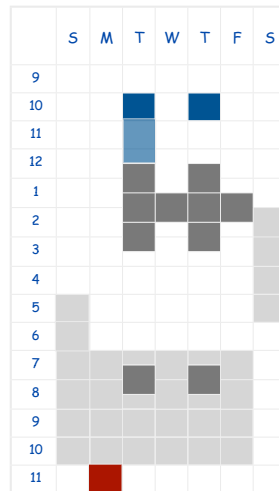
■ Precepts. [Gabai · Moretti · August · Berkiten · Ghosh · Hug · Kiefer · Lee · Suleimenov · Wang]

- Tips on assignments / worked examples
- Questions on lecture material.
- Informal and interactive.

■ Friend 016/017(?) lab. [Ugrad assistants]

- Help with systems/debugging.
- No help with course material.

Reading period. No lectures; precepts T and W.



See [www.princeton.edu/~cos126](http://www.princeton.edu/~cos126) for full details and preceptor office hours.

## Grades

Course grades. No preset curve or quota.

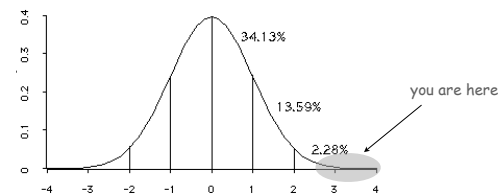
9 programming assignments. 40%.

2 exams (in class, 10/13-14 and 12/15-16). 50%.

Final programming project (due Dean's date - 1). 10%.

Extra credit / staff discretion. Adjust borderline cases.

participation helps, frequent absence hurts



Due dates

|     | Su | Mo | Tu | We | Th | Fr | Sa |
|-----|----|----|----|----|----|----|----|
|     |    |    |    |    | 1  | 2  | 3  |
|     | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
| SEP | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|     | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|     | 25 | 26 | 27 | 28 | 29 | 30 |    |
|     |    |    |    |    |    | 1  |    |
|     | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
|     | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
| OCT | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|     | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|     | 30 | 31 |    |    |    |    |    |
|     |    |    | 1  | 2  | 3  | 4  | 5  |
|     | 6  | 7  | 8  | 9  | 10 | 11 | 12 |
|     | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| NOV | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
|     | 27 | 28 | 29 | 30 |    |    |    |
|     |    |    |    |    | 1  | 2  | 3  |
|     | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|     | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| DEC | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|     | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|     |    |    |    |    |    |    |    |
|     | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|     | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
|     | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| JAN | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|     | 29 | 30 | 31 |    |    |    |    |

## Course Materials

Course website. [[www.princeton.edu/~cos126](http://www.princeton.edu/~cos126)]

- Submit assignments
- Programming assignments.
- Lecture slides.

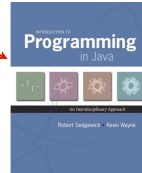
(print before lecture)  
annotate during lecture

Course text.

Sedgwick and Wayne.

*Intro to Programming in Java: An Interdisciplinary Approach.*

skim before lecture;  
read thoroughly  
afterwards



Recommended reading (lectures 18-20).

Harel. *What computers can't do.*

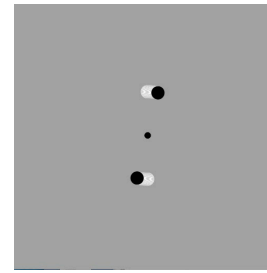


5

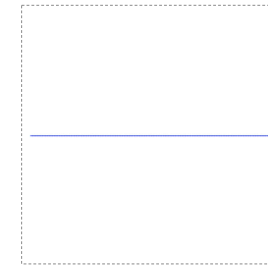
## Programming Assignments

Desiderata.

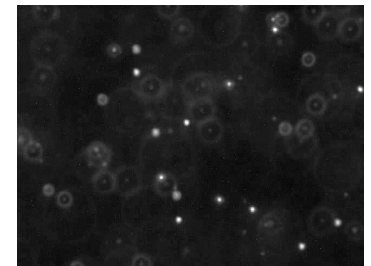
- Address an important scientific or commercial problem.
- Illustrate the importance of a fundamental CS concept.
- You solve problem from scratch!



N-body simulation



pluck a guitar string



estimate Avogadro's number

6

## Programming Assignments

Desiderata.

- Address an important scientific or commercial problem.
- Illustrate the importance of a fundamental CS concept.
- You solve problem from scratch!

Due. Mondays 11pm via Web submission.

Computing equipment.

- Your laptop. [OS X, Windows, Linux, iPhone, ... ]
- OIT desktop. [Friend 016 and 017(?) labs]

7

## What's Ahead?

Lecture 2. Intro to Java.

Precept 1. Meets today/tomorrow.

Not registered? Go to any precept now; officially register ASAP.

Change precepts? Use SCORE.

see Colleen Kenny-McGinley in CS 210  
if the only precept you can attend is closed

Assignment 0. [[www.princeton.edu/~cos126/assignments.php](http://www.princeton.edu/~cos126/assignments.php)]

- Due Monday 11PM.
- Read Sections 1.1 and 1.2 in textbook.
- Install Java programming environment + a few exercises.
- Lots of help available, don't be bashful.

END OF ADMINISTRATIVE STUFF

8

# 0. Prologue: A Simple Machine

## Secure Chat with a One-Time Pad

Alice wants to send a secret message to Bob

- Sometime in the past, they exchange a **one-time pad**.
- Alice uses the pad to encrypt the message.

"use yT25a5i/S if I ever send you an encrypted message"



```
Secure Chat 1.0 [alice]
[alice]: Hey, Bob
[bob]: Hi, Alice!
[alice]: SENDMONEY

Encrypt SENDMONEY with yT25a5i/S
```

```
Secure Chat 1.0 [bob]
[alice]: Hey, Bob
[bob]: Hi, Alice!
[alice]: gX76W3v7K
```

Key point: Without the pad, Eve cannot understand the message.



## Encryption Machine

Goal. Design a machine to encrypt and decrypt data.

S E N D M O N E Y



g X 7 6 W 3 v 7 K



S E N D M O N E Y



Enigma encryption machine.

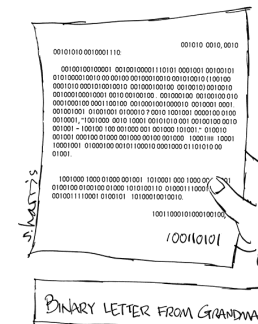
- "Unbreakable" German code during WWII.
- Broken by Turing bombe.
- One of first uses of computers.
- Helped win Battle of Atlantic by locating U-boats.

## A Digital World

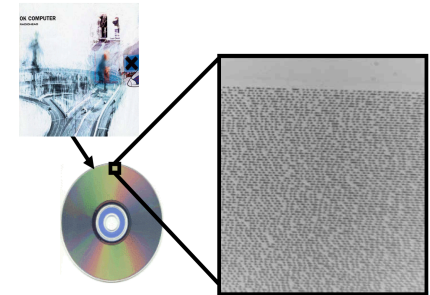
Data is a sequence of bits. [bit = 0 or 1] ← can use decimal digits, letters, or some other system, but bits are more easily encoded physically ("on-off", "up-down", "hot-cold", ...)

- Text.
- Programs, executables.
- Documents, pictures, sounds, movies, ...

thousands of bits



billions of bits



Data is a sequence of bits. [bit = 0 or 1]

- Text.
- Programs, executables.
- Documents, pictures, sounds, movies, ...

Ex. Base64 encoding of text.

- Simple method for representing A-Z, a-z, 0-9, +, /
- 6 bits to represent each symbol (64 symbols)

|        |   |        |   |        |   |        |   |        |   |        |   |        |   |        |   |
|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|--------|---|
| 000000 | A | 001000 | I | 010000 | Q | 011000 | Y | 100000 | g | 101000 | o | 110000 | w | 111000 | 4 |
| 000001 | B | 001001 | J | 010001 | R | 011001 | Z | 100001 | h | 101001 | p | 110001 | x | 111001 | 5 |
| 000010 | C | 001010 | K | 010010 | S | 011010 | a | 100010 | i | 101010 | q | 110010 | y | 111010 | 6 |
| 000011 | D | 001011 | L | 010011 | T | 011011 | b | 100011 | j | 101011 | r | 110011 | z | 111011 | 7 |
| 000100 | E | 001100 | M | 010100 | U | 011100 | c | 100100 | k | 101100 | s | 110100 | 0 | 111100 | 8 |
| 000101 | F | 001101 | N | 010101 | V | 011101 | d | 100101 | l | 101101 | t | 110101 | 1 | 111101 | 9 |
| 000110 | G | 001110 | O | 010110 | W | 011110 | e | 100110 | m | 101110 | u | 110110 | 2 | 111110 | + |
| 000111 | H | 001111 | P | 010111 | X | 011111 | f | 100111 | n | 101111 | v | 110111 | 3 | 111111 | / |

First challenge: Create a one-time pad.

Good choice: A random sequence of bits (stay tuned).

Note: any sequence of bits can be encoded as characters

|        |        |        |        |        |        |        |        |        |                       |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----------------------|
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | one-time pad          |
| y      | T      | 2      | 5      | a      | 5      | i      | /      | s      | encoded as characters |

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.

Base64 Encoding

| char | dec | binary |
|------|-----|--------|
| A    | 0   | 000000 |
| B    | 1   | 000001 |
| ...  | ... | ...    |
| M    | 12  | 001100 |
| ...  | ... | ...    |

|        |        |        |        |        |        |        |        |        |         |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------|
| S      | E      | N      | D      | M      | O      | N      | E      | Y      | message |
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | base64  |

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Use N random bits as one-time pad.

|        |        |        |        |        |        |        |        |        |              |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|
| S      | E      | N      | D      | M      | O      | N      | E      | Y      | message      |
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | base64       |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | one-time pad |
| y      | T      | 2      | 5      | a      | 5      | i      | /      | s      |              |

## One-Time Pad Encryption

### Encryption.

- Convert text message to N bits.
- Use N random bits as one-time pad.
- Take bitwise XOR of two bitstrings.

sum corresponding pair of bits: 1 if sum is odd, 0 if even

XOR Truth Table

| x | y | $x \oplus y$ |
|---|---|--------------|
| 0 | 0 | 0            |
| 0 | 1 | 1            |
| 1 | 0 | 1            |
| 1 | 1 | 0            |

| S      | E      | N      | D      | M      | O      | N      | E      | Y      |              |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | message      |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | base64       |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | one-time pad |
|        |        |        |        |        |        |        |        |        | XOR          |

$0 \oplus 1 = 1$

17

## One-Time Pad Encryption

### Encryption.

- Convert text message to N bits.
- Use N random bits as one-time pad.
- Take bitwise XOR of two bitstrings.
- Convert binary back into text.

Base64 Encoding

| char | dec | binary |
|------|-----|--------|
| A    | 0   | 000000 |
| B    | 1   | 000001 |
| ...  | ... | ...    |
| w    | 22  | 010110 |
| ...  | ... | ...    |

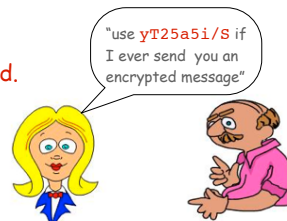
| S      | E      | N      | D      | M      | O      | N      | E      | Y      |              |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | message      |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | base64       |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | one-time pad |
|        |        |        |        |        |        |        |        |        | XOR          |
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted    |

18

## Secure Chat with a One-Time Pad

### Alice wants to send a secret message to Bob

- Sometime in the past, they exchange a **one-time pad**.
- Alice uses the pad to encrypt the message.



Secure Chat 1.0 [alice]  
 [alice]: Hey, Bob  
 [bob]: Hi, Alice!  
 [alice]: SENDMONEY  
 Encrypt SENDMONEY with yT25a5i/S

Secure Chat 1.0 [bob]  
 [alice]: Hey, Bob  
 [bob]: Hi, Alice!  
 [alice]: gX76W3v7K

Key point: Without the pad, Eve cannot understand the message.  
 But how can Bob understand the message?

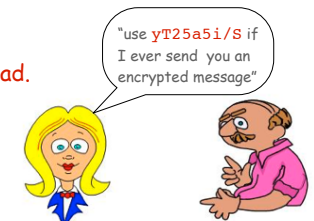


19

## Secure Chat with a One-Time Pad

### Alice wants to send a secret message to Bob

- Sometime in the past, they exchange a **one-time pad**.
- Alice uses the pad to encrypt the message.
- Bob uses the same pad to decrypt the message.



Secure Chat 1.0 [alice]  
 [alice]: Hey, Bob  
 [bob]: Hi, Alice!  
 [alice]: SENDMONEY  
 Encrypt SENDMONEY with yT25a5i/S

Secure Chat 1.0 [bob]  
 [alice]: Hey, Bob  
 [bob]: Hi, Alice!  
 [alice]: gX76W3v7K  
 SENDMONEY  
 Decrypt with yT25a5i/S

Key point: Without the pad, Eve cannot understand the message.



20

## One-Time Pad Decryption

### Decryption.

- Convert encrypted message to binary.

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| g | x | 7 | 6 | w | 3 | v | 7 | k |
|---|---|---|---|---|---|---|---|---|

encrypted

21

## One-Time Pad Decryption

### Decryption.

- Convert encrypted message to binary.

Base64 Encoding

| char | dec | binary |
|------|-----|--------|
| A    | 0   | 000000 |
| B    | 1   | 000001 |
| ...  | ... | ...    |
| W    | 22  | 010110 |
| ...  | ... | ...    |

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| g | x | 7 | 6 | w | 3 | v | 7 | k |
|---|---|---|---|---|---|---|---|---|

encrypted

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

base64

22

## One-Time Pad Decryption

### Decryption.

- Convert encrypted message to binary.
- Use **same** N "random" bits (one-time pad).

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| g | x | 7 | 6 | w | 3 | v | 7 | k |
|---|---|---|---|---|---|---|---|---|

encrypted

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

base64

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

one-time pad

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| y | t | 2 | 5 | a | 5 | i | / | s |
|---|---|---|---|---|---|---|---|---|

23

## One-Time Pad Decryption

### Decryption.

- Convert encrypted message to binary.
- Use same N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.

XOR Truth Table

| x | y | $x \oplus y$ |
|---|---|--------------|
| 0 | 0 | 0            |
| 0 | 1 | 1            |
| 1 | 0 | 1            |
| 1 | 1 | 0            |

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| g | x | 7 | 6 | w | 3 | v | 7 | k |
|---|---|---|---|---|---|---|---|---|

encrypted

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

base64

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

one-time pad

|        |        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|

XOR

$1 \oplus 1 = 0$

24

## One-Time Pad Decryption

### Decryption.

- Convert encrypted message to binary.
- Use same N random bits (one-time pad).
- Take bitwise XOR of two bitstrings.
- Convert back into text.

| Base64 Encoding |     |        |
|-----------------|-----|--------|
| char            | dec | binary |
| A               | 0   | 000000 |
| B               | 1   | 000001 |
| ...             | ... | ...    |
| M               | 12  | 001100 |
| ...             | ... | ...    |

|        |        |        |        |        |        |        |        |        |              |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------|
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted    |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64       |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | one-time pad |
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | XOR          |
| S      | E      | N      | D      | M      | O      | N      | E      | Y      | message      |

25

## Why Does It Work?

Crucial property. Decrypted message = original message.

| Notation    | Meaning               |
|-------------|-----------------------|
| a           | original message bit  |
| b           | one-time pad bit      |
| ^           | XOR operator          |
| a ^ b       | encrypted message bit |
| (a ^ b) ^ b | decrypted message bit |

XOR Truth Table

| x | y | x ^ y |
|---|---|-------|
| 0 | 0 | 0     |
| 0 | 1 | 1     |
| 1 | 0 | 1     |
| 1 | 1 | 0     |

### Why is crucial property true?

- Use properties of XOR.
  - $(a \oplus b) \oplus b = a \oplus (b \oplus b) = a \oplus 0 = a$
- ↑ associativity of ^
↑ always 0
↑ identity

26

## One-Time Pad Decryption (with the wrong pad)

### Decryption.

- Convert encrypted message to binary.

|   |   |   |   |   |   |   |   |   |           |
|---|---|---|---|---|---|---|---|---|-----------|
| g | x | 7 | 6 | w | 3 | v | 7 | k | encrypted |
|---|---|---|---|---|---|---|---|---|-----------|

27

## One-Time Pad Decryption (with the wrong pad)

### Decryption.

- Convert encrypted message to binary.

|        |        |        |        |        |        |        |        |        |           |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----------|
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64    |

28

## One-Time Pad Decryption (with the wrong pad)

### Decryption.

- Convert encrypted message to binary.
- Use **wrong** N bits (bogus one-time pad).

|        |        |        |        |        |        |        |        |        |            |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|------------|
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted  |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64     |
| 101000 | 011100 | 110101 | 101111 | 010010 | 111001 | 100101 | 101010 | 001010 | wrong bits |

29

## One-Time Pad Decryption (with the wrong pad)

### Decryption.

- Convert encrypted message to binary.
- Use **wrong** N bits (bogus one-time pad).
- Take bitwise XOR of two bitstrings.

|        |        |        |        |        |        |        |        |        |            |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|------------|
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted  |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64     |
| 101000 | 011100 | 110101 | 101111 | 010010 | 111001 | 100101 | 101010 | 001010 | wrong bits |
| 001000 | 001011 | 001110 | 010101 | 000100 | 001110 | 001010 | 010001 | 000000 | XOR        |

30

## One-Time Pad Decryption (with the wrong pad)

### Decryption.

- Convert encrypted message to binary.
- Use **wrong** N bits (bogus one-time pad).
- Take bitwise XOR of two bitstrings.
- Convert back into text: **Oops**.

|        |        |        |        |        |        |        |        |        |                                      |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------------------------------------|
| g      | x      | 7      | 6      | w      | 3      | v      | 7      | k      | encrypted                            |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64                               |
| 101010 | 110000 | 000011 | 100000 | 011011 | 000011 | 101110 | 011010 | 101111 | wrong bits                           |
| 001010 | 100111 | 111000 | 011010 | 001101 | 110100 | 000001 | 100001 | 100101 | XOR                                  |
| K      | n      | 4      | a      | N      | 0      | B      | h      | l      | wrong message<br>[usually gibberish] |

31

## Eve's Problem (one-time pads)

**Key point:** Without the pad, Eve cannot understand the message.



But Eve has a computer. Why not try all possible pads?

One problem: it might take a long time [stay tuned].

**Worse problem:** she would see all possible messages!

- 54 bits
- $2^{54}$  possible messages, all different.
- $2^{54}$  possible **encoded** messages, all different.
- No way for Eve to distinguish real message from any other message.

One-time pad is "provably secure".

|            |           |
|------------|-----------|
| AAAAAAAAA  | gX76W3v7K |
| AAAAAAAAAB | gX76W3v7L |
| AAAAAAAAAC | gX76W3v7I |
| ...        |           |
| qwDgbDuav  | Kn4aN0Bh1 |
| ...        |           |
| tTtpWk+1E  | NEWTATTOO |
| ...        |           |
| yT25a5i/s  | SENDMONEY |
| ...        |           |
| /////////+ | fo7FpIQE0 |
| /////////  | fo7FpIQE1 |

32



## Goods and Bads of One-Time Pads

### Good.

- Easily computed by hand.
- Very simple encryption/decryption processes.
- Provably unbreakable if bits are truly random. [Shannon, 1940s]

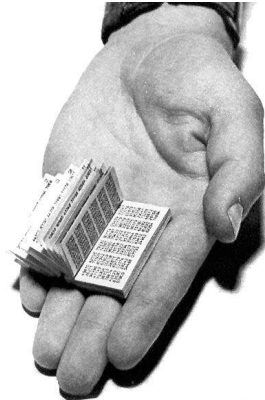
eavesdropper Eve sees only random bits

"one time" means one time only

### Bad.

- Easily breakable if pad is re-used.
- Pad must be as long as the message.
- Truly random bits are very hard to come by.
- Pad must be distributed securely.

impractical for Web commerce



a Russian one-time pad

## Pseudo-Random Bit Generator

### Practical middle-ground.

- Make a "random" bit generator gadget.
- Alice and Bob each get identical small gadgets [same gadget works for both]
- Alice and Bob also each get identical books of small seeds.

instead of identical large one-time pads

Goal. Small gadget that produces a long sequence of bits.



34

## Pseudo-Random Bit Generator

Small deterministic gadgets that produce long sequences of pseudo-random bits:

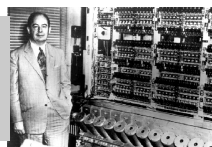
- Enigma
- Linear feedback shift register.
- Linear congruential generator.
- Blum-Blum-Shub generator.
- [many others have been invented]

Pseudo-random? Bits are not really random:

- Bob's and Alice's gadgets must produce the same bits from the same seed.
- Bits must have as many properties of random bits as possible (to foil Eve).

Ex 1. approx 1/2 0s and 1/2 1s  
Ex 2. approx 1/4 each of 00, 01, 10 11

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."  
– Jon von Neumann (left)  
– ENIAC (right)

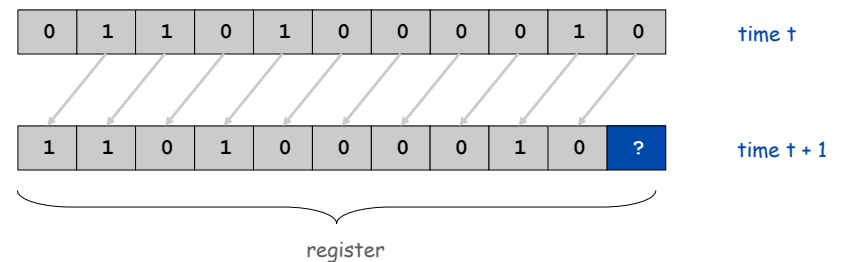


35

## Shift Register

Shift register terminology.

- Bit: 0 or 1.
- Cell: storage element that holds one bit.
- Register: sequence of cells.
- Seed: initial sequence of bits.
- Shift register: when clock ticks, bits propagate one position to left.

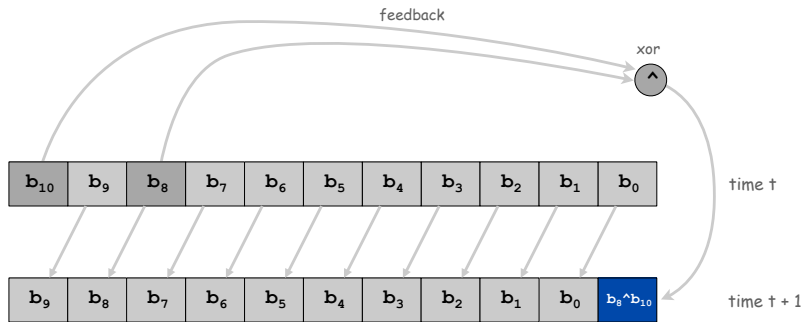


36

## Linear Feedback Shift Register (LFSR)

{8, 10} linear feedback shift register.

- Shift register with 11 cells.
- Bit  $b_0$  is XOR of previous bits  $b_8$  and  $b_{10}$ .
- Pseudo-random bit =  $b_0$ .

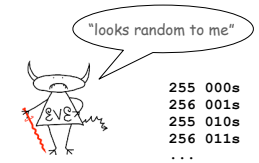


37

## Random Numbers

Q. Are these 2000 numbers random?

If not, what is the pattern?



```
110010010011110110111001011010111001100010111111010010000100110100101111001100100111111011100
00010101100010000111010100110100001110010011001110111110101000001000010001010101000011000
00101110001001001101010111100011010011011001111010111001000100111010111101000001010010001
0001101010101110000000101100000100110001011101101001010110011000011111100110000011111000010
000110111001110100111010011100100111011011010101010100000000010000000101000001000100001
01010100100000011010000111001000110111010110100010100001010000101011010100001100001100001
0011110010111001110010111011001001010110110000101011001000010111010001010011011000111101
1101100101010111000000010011000001111100100100011010101011011000110011101110101010010110
00011001110011111011100001010011001000111110101100001000111001010101100001101011001110001
11110110100010101110100110101001111000011100110011011111110100000010010000010110100010011
00101011111000010000110010100111100011100011010101101101101010110100000110110001101011
011010001101100101110111001100101001110000011101100011010110111000010101010000001101000011
101001100010011110101110001000101010101001100000011110000110001100111011110010100001110
001001101010111011000100101110101100101000111000111000101000111100011100011110011000011101
111111001000000111010000110100011100110111010101000100000101010000100001001010000101
1000101001110100011101001011010011001100111111111000000001100000011100000110011000111111
1011000001011100001001011001011001110011110011100011100110100111101111000101000101000
101110010100101100011001010111100110100011110010100011100110110110101010010001001101
01111110001000001101010001110000101101001001101110111010010100100010111110111010000101
0100101000001100010001111010101100100001111010001100101011110110100100101110101001000011
01101001110110110101111010001000101010101000000011100000110110000111011001101010111
100001000110001010111010
```

A. No. This is output of {8, 10} LFSR with seed 01101000010!

38

## LFSR Encryption

Encryption.

- Convert text message to N bits.
- Initialize LFSR with given seed
- Generate N bits **with LFSR**.
- Take bitwise XOR of two bitstrings.
- Convert binary back into text.

| Base64 Encoding |     |        |
|-----------------|-----|--------|
| char            | dec | binary |
| A               | 0   | 000000 |
| B               | 1   | 000001 |
| ...             | ... | ...    |
| w               | 22  | 010110 |
| ...             | ... | ...    |

|        |        |        |        |        |        |        |        |        |           |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----------|
| S      | E      | N      | D      | M      | O      | N      | E      | Y      | message   |
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | base64    |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | LFSR bits |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | XOR       |
| g      | X      | 7      | 6      | W      | 3      | v      | 7      | K      | encrypted |

39

## LFSR Decryption

Decryption.

- Convert encrypted message to binary.
- Initialize identical LFSR with **same seed**
- Generate N bits **with LFSR**.
- Take bitwise XOR of two bitstrings.
- Convert back into text.

| Base64 Encoding |     |        |
|-----------------|-----|--------|
| char            | dec | binary |
| A               | 0   | 000000 |
| B               | 1   | 000001 |
| ...             | ... | ...    |
| M               | 12  | 001100 |
| ...             | ... | ...    |

|        |        |        |        |        |        |        |        |        |           |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----------|
| g      | X      | 7      | 6      | W      | 3      | v      | 7      | K      | encrypted |
| 100000 | 010111 | 111011 | 111010 | 010110 | 110111 | 101111 | 111011 | 001010 | base64    |
| 110010 | 010011 | 110110 | 111001 | 011010 | 111001 | 100010 | 111111 | 010010 | LFSR bits |
| 010010 | 000100 | 001101 | 000011 | 001100 | 001110 | 001101 | 000100 | 011000 | XOR       |
| S      | E      | N      | D      | M      | O      | N      | E      | Y      | message   |

40

## Eve's Problem (LFSR encryption/decryption)

**Key point:** Without the (short) seed

Eve cannot understand the (long) message.



But Eve has a computer. Why not try all possible seeds?

- Seeds are short, messages are long.
- All seeds give a tiny fraction of all messages.
- Extremely likely that all but real seed will produce gibberish.

**Bad news (for Eve): There are still too many possibilities!**

- Ex: 60-bit seed implies  $2^{60}$  possibilities.
- If Eve could check 1 million seeds per second, it would take her **365 centuries** to try them all!

Exponential growth dwarfs technological improvements [stay tuned].

- 1000 bits:  $2^{1000}$  possibilities.
- Age of the universe in microseconds:  $2^{70}$



## Goods and Bads of LFSRs

**Good.**

- Easily computed with simple machine.
- Very simple encryption/decryption processes.
- Bits have many of the same properties as random bits.
- Scalable: 20 cells for 1 million bits; 30 cells for 1 billion bits.  
[ but need theory of finite groups to know where to put taps ]



a commercially available LFSR

**Bad.**

- Still need secure, independent way to distribute LFSR seed.
- The bits are not truly random.  
[ bits in our 11-bit LFSR cycle after  $2^{11} - 1 = 2047$  steps ]
- Experts have cracked LFSR encryption.  
[ need more complicated machines ]

## Other LFSR Applications

**What else can we do with a LFSR?**

- DVD encryption with CSS.
- DVD decryption with DeCSS!
- Subroutine in military cryptosystems.

```

/*  efdtt.c  Author: Charles M. Hannum <root@ihack.net>  */
/*  Usage is: cat title-key scrambled.vob | efdtt >clear.vob  */

#define m(i) (x[i]^s[i+84])<<

    unsigned char x[5]          ,y,s[2048];main(
n){for( read(0,x,5          );read(0,s ,n=2048
); write(1 ,s,n          )if(s
[y=s [13]%8+20] /16%4 ==1 )if(s
i=m( 1)17 ^256 +m(0) 8,k =m(2)
0,j= m(4) 17^ m(3) 9*k* 2-k%8
^8,a =0,c =26;for (s[y] -=16;
--c;j *=2)a= a*2^i% 1,i=i /2^j&1
<<24;for(j= 127; ++j<n;c=c>
y)
c
+*=i^i/8^i>>4^i>>12,
i=i>>8^y<<17,a^=a>>14,y=a^a*8^a<<6,a=a
>>8^y<<9,k=s[j],k =7Wo-'G_216"[k
&7]+2^"cr3sfw6w;*k->>n." [k>>4]*2^k+257/
8,s[j]=k^(k&k*2&34)*6^c+-y
;)}
    
```

<http://www.cs.cmu.edu/~dst/DeCSS/gallery>

## Typical Exam Question (TEQ) on LFSRs 1

Give first 10 steps of {3, 4} LFSR with initial fill 00001.

## TEQ on LFSRs 2

**Goal.** Decrypt/encrypt 300 characters (1800 bits).

**Challenge.** Is it a good idea to use an 11-bit LFSR?

A. Yes, no problem.

B. No, the bits it produces are not truly random.

C. No, need a longer LFSR.

D. No, experts have cracked LFSRs

45

## A Profound Idea

**Programming.** Can write a Java program to simulate the operations of **any** abstract machine.

- Basis for theoretical understanding of computation. [stay tuned]
- Basis for bootstrapping real machines into existence. [stay tuned]

**Stay tuned.** See Assignment 5.

```
public class LFSR
{
    private int[] seed;
    private int tap;
    private int N;

    public LFSR(String seed, int tap) { ... }

    public int step() { ... }

    public static void main(String[] args)
    {
        LFSR lfsr = new LFSR("01101000010", 8);
        for (int i = 0; i < 2000; i++)
            StdOut.println(lfsr.step());
    }
}
```

```
% java LFSR
1100100100111101101110010110101
1100110001011111101001000010011
0100101111001100100111...
```

47

## LFSR and "General Purpose Computer"

**Important properties.**

- Built from simple components.
- Scales to handle huge problems.
- Requires a deep understanding to use effectively.

| Basic Component | LFSR               | Computer               |
|-----------------|--------------------|------------------------|
| control         | start, stop, load  | same                   |
| clock           | regular pulse      | 2.8 GHz pulse          |
| memory          | 11 bits            | 1 GB                   |
| input           | seed               | sequence of bits       |
| computation     | shift, XOR         | logic, arithmetic, ... |
| output          | pseudo-random bits | Sequence of bits       |

**Critical difference.** General purpose machine can be programmed to simulate ANY abstract machine.

46

## A Profound Question

Q. What is a random number?

**LFSR does not produce random numbers.**

- It is a very simple deterministic machine.
- Not obvious how to distinguish the bits it produces from random.
- Experts have figured out how to do so.

Q. Are random processes found in nature?

- Motion of cosmic rays or subatomic particles?
- Mutations in DNA?

Q. Is the natural world a (not-so-simple) deterministic machine?

"God does not play dice."  
– Albert Einstein



48