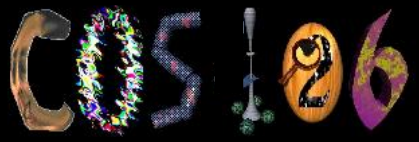


General Computer Science
Princeton University
Spring 2011

Kevin Wayne

www.princeton.edu/~cos126

Overview

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

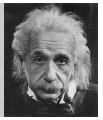
Goals.

- 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. [Kevin Wayne]

- Tuesdays and Thursdays, Frist 302.
- Same lecture at 10am and 11am.
- Office hours: M Th 2-3pm in CS 207. ← or just drop by

Precepts. [Donna Gabai (co-lead) · Keith Vertanen (co-lead) · 11 others]

- Tue+Thu or Wed+Fri.
- Tips on assignments, worked examples, clarify lecture material.

Computing laboratory. [Undergrad lab assistants]

- Sun 5-11pm, Mon-Fri 7-11pm, Sat 2-6pm in Friend 017.
- Help with debugging.

Full details and office hours. See `www.princeton.edu/~cos126`

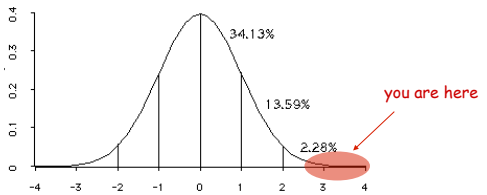
Grades

Course grades. No preset curve or quota.

9 programming assignments. 40%.
2 exams. 50%.
Final programming project. 10%.
Extra credit and staff discretion. Adjust borderline cases.

↖ participation helps, frequent absences hurts

Check grades. Blackboard. [`www.blackboard.princeton.edu`]



Course Materials

Course website. [www.princeton.edu/~cos126]

- Programming assignments and checklists.
- Submit assignments.
- Lecture slides. ← print before lecture; annotate during lecture
- Exam archive.

skim before lecture; read thoroughly afterwards

Required readings. Sedgewick and Wayne. *Intro to Programming in Java: An Interdisciplinary Approach.* [Labyrinth]

Princeton royalties donated to ACM-W this semester →

Recommended readings. Harel. *What computers can't do.* [Labyrinth]

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]

What's Ahead?

Lecture 2. Intro to Java.

Precept 1. Meets today/tomorrow.
Precept 2. Meets Thu/Fri.

Not registered? Go to any precept now; officially register ASAP.
Change precepts? Use SCORE.

for enrollment problems, see Colleen Kenny-McGinley in CS 210

Assignment 0. 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.

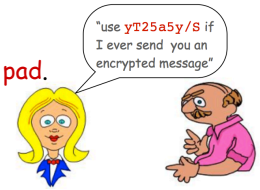
END OF ADMINISTRATIVE STUFF

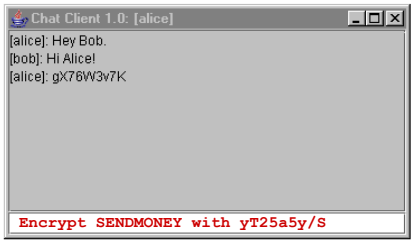
0. Prologue: A Simple Machine

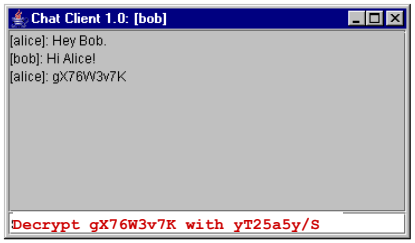
Secure Chat

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.



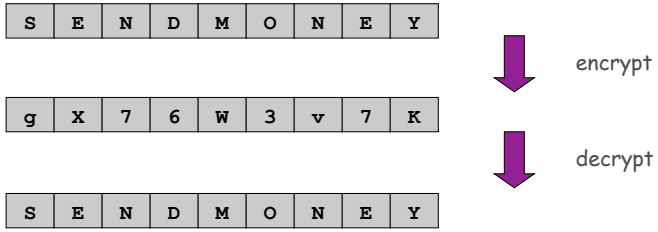




Key point. Without the pad, Eve cannot understand the message.

Encryption Machine

Goal. Design a machine to encrypt and decrypt data.



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]

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

File formats. txt, pdf, java, exe, docx, pptx, jpeg, mp3, divx, ...

computer with a lens

computer with earbuds

computer with a radio

A Digital World

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

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

Base64 encoding. Use 6 bits to represent each alphanumeric symbol.

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

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	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (one-time pad).

S	E	N	D	M	O	N	E	Y	
010010	000100	001101	000011	001100	001110	001101	000100	011000	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (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 ^ 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	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR

0 ^ 1 = 1

One-Time Pad Encryption

Encryption.

- Convert text message to N bits.
- Generate N random bits (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	base64
110010	010011	110110	111001	011010	111001	100010	111111	010010	random bits
100000	010111	111011	111010	010110	110111	101111	111011	001010	XOR
g	X	7	6	W	3	v	7	K	encrypted

One-Time Pad Decryption

Decryption.

- Convert encrypted message to binary.

g	X	7	6	W	3	v	7	K	
									encrypted

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

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	random bits

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 ^ 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	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR

1 ^ 1 = 0

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	random bits
010010	000100	001101	000011	001100	001110	001101	000100	011000	XOR
S	**E**	**N**	**D**	**M**	**O**	**N**	**E**	**Y**	message

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 ^ b) ^ b = a ^ (b ^ b) = a ^ 0 = a

associativity of ^ · always 0 · identity

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
101000	011100	110101	101111	010010	111001	100101	101010	001010	wrong bits
001000	001011	001110	010101	000100	001110	001010	010001	000000	XOR
I	**L**	**O**	**V**	**E**	**O**	**K**	**R**	**A**	wrong message

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]

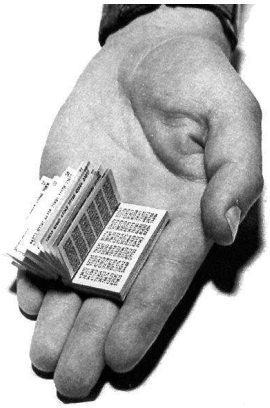
eavesdropper Eve sees only random bits

"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.

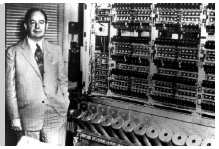
- Let's make a "random" bit generator gadget.
- Alice and Bob each get identical small gadgets.

instead of identical large one-time pads

How to make small gadget that produces pseudo-random numbers.

- Enigma.
- Linear feedback shift register.
- Linear congruential generator.
- Blum-Blum-Shub generator.
- ...

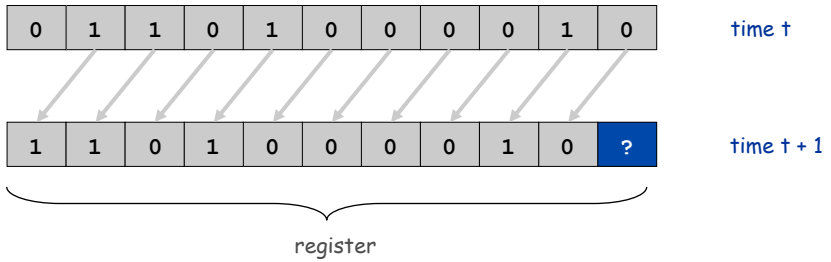
" Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. "
– Jon von Neumann (left)
– ENIAC (right)



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.

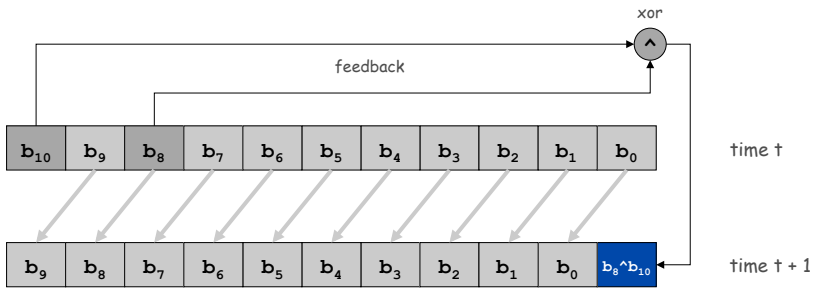


Linear Feedback Shift Register (LFSR)

{8, 10} linear feedback shift register.

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

LFSR demo



Random Numbers

Q. Are these 2000 numbers random? If not, what is the pattern?

```
11001001001111011011100101101011100110001011111101001000010011010010111100110010011111
10111000001010110001000011101010011010000111100100110011101111111010100000100001000101
01010100011000001011110001001001101101111000111010011011100111101011110010001001110101
01110100000101001000100011010101011100000001011000001001110001011101101001010110011000
11111110011000001111110001100001101111001110100111001001110111011010101010101000
00000001000000001010000001000100001010101001000000011010000011100100011011101011010100
01010000101000100100010101110101000011000010011110010111001110010111011100100101011011
00001010111001000010111010010010100111000111101110110010101011110000001001100001011111
10010010001110110101101100011000111011110110101001011000011001110011111101111000010100
01100100011111101011000010001011010100110000001111100001100011001111011111001010000
11100010011010111011100010010111010110010100011110001011001101001111100111000011110
11001100101111111001000001110100001101001001110011011011111010101000100000010101000
10000010010001011000101001110100011010010110100110011001111111111100000000011000000
11110000011001100011111111011000000101110000100101100101100111100111110011100011110011
01100111110111110001010001101000101110010100101110001100101101111100110100011110010110
00111001110110111101011010010001100110101111111000100000110101000111000010110110010110
11110111101001010010011000110111101110100010101001010000011000100011101010110010000001
11101000110010010111110110010010111101010010010000110110100111011001110100111101000100
01001010101011000000001110000001101100001110111001101010111100000100011000101011110
```

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

LFSR Encryption

LFSR encryption.

- Convert text message to N bits.
- Initialize LFSR with small 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

LFSR Decryption

LFSR Decryption.

- Convert encrypted message to binary.
- Initialize identical LFSR with same 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
…	…	…
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

Goods and Bads of LFSR Encryption

Goods.

- Easily computed with simple machine.
- Very simple encryption / decryption process.
- 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

Bads.

- 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.
  [more complicated machines needed]

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       ){int
                  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

                          +=y=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^"cr3sfw6v;*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

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.

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 final int tap;
   private final 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
0111001100010111111010010000 1
0011010010111100110010011 1...
```


A Profound Question

Q. What is a random number?

LFSR does not produce random numbers.

- It is a very simple deterministic machine.
- But not obvious how to distinguish the bits it produces from random.

Q. Are random processes found in nature?

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

Q. Or, is the natural world a (not-so-simple) deterministic machine?

" God does not play dice. "
– Albert Einstein