

CIS 110

Introduction to Computer Programming
University of Pennsylvania
Summer 2012

Benedict Brown

www.seas.upenn.edu/~cis110

Overview

What is CIS 110? Introduction to programming and computer science

Goals

- Demystify computer systems: what can computer do?
- Demystify computer science: what are the intellectual underpinnings?
- Demystify computer scientists: how do we think?

Topics

- Programming in Java
- Designing and understanding behavior of algorithms
- 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



2

The Basics

Lectures

 [Benedict Brown]

- Monday - Thursday, Towne 303, 10am - 11am
- Office hours: TBD in GRW 258 *and by appointment*
- Email: bjbzown@cis.upenn.edu

Recitations

- Monday - Thursday, Towne 311 and TBD, 11am - 12pm
- Attendance required
- Two TAs per section: twice the fun!
- Clarify lecture material, tips on assignments, group activities

Computer laboratory

- TA office hours in Moore computer lab (TBD)
- Help with debugging

Full details and office hours. See www.seas.upenn.edu/~cis110

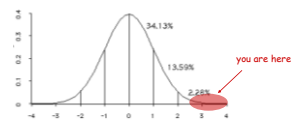
3

Grades

Course grades No preset curve or quota, but all work will be curved

~6 programming assignments	50%
Midterm	20%
Final Exam	30%
Extra credit and staff discretion	Adjust borderline cases.

Check grades Course web site



4

Course Materials

Course website

 [www.seas.upenn.edu/~cis110]

- Programming assignments and checklists
- Submit assignments
- Lecture slides *usually posted after lecture*
- Lecture videos *don't always record properly, and don't record everything!*
- Discussion board

Required text: Sedgewick and Wayne. *Intro to Programming in Java: An Interdisciplinary Approach.* *skim before lecture; read thoroughly afterwards*



5

Programming Assignments

Desiderata

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

Due

 9pm on due date via Web submission

- 3 hour grace period (no extra credit, won't get questions answered)
- No late days! No extensions! *Summer term is too compressed*

Computing equipment.

- Your laptop [OS X, Windows, Linux, ...]
- Moore computer labs

Advice.

- Start early; plan multiple sessions
- Seek help when needed *our job is to help you!*

7

What's Ahead?

Lecture 2 Intro to Java (11am)

First recitation Tomorrow

Change your section? Contact Dr. Brown [bjbrown@cis.upenn.edu]

Assignment 0. Due tomorrow, May 22, 9pm.

- Read Sections 1.1 and 1.2 in textbook.
- Install Java programming environment + a few exercises.
- Register for Piazza [http://piazza.com/upenn/summer2012/cis110]
- Lots of help available, don't be bashful.

END OF ADMINISTRATIVE STUFF

8

Languages

Machine languages. Tedious and error-prone.

Natural languages. Ambiguous and hard for computer to parse.

Kids Make Nutritious Snacks
Red Tape Holds Up New Bridge
Police Squad Helps Dog Bite Victim
Local High School Dropouts Cut in Half
[real newspaper headlines, compiled by Rich Pattis]

'At Last' Singer Etta James Terminally Ill
[New York Time Online, 16 December 2011]

"Santorium is an Unpleasant By-Product of Sex"
[dereadiche, 6 January 2012]

High-level programming languages. Acceptable tradeoff.

"Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do." – Donald Knuth



9

Hello, World



0. Prologue: A Simple Machine

Introduction to Programming in Java: An Interdisciplinary Approach Robert Sedgwick and Kevin Wayne · Copyright © 2002–2010 · 19 May 2012 1:43:52

11

Security on msnbc.com

Hidden flaw jeopardizes millions of online transactions

Mathematicians discover weakness in commonly used encryption schemes

Jump to discuss comments below

Recommend 38

Tweet 9

+7 3

Share 20

Below: Discuss Related

By Paul Wagenseil

security news

updated 2/15/2012 6:01:37 PM ET

Mathematicians based in Switzerland and the United States have discovered a small but the most commonly used digital encryption schemes, a flaw that could undermine the sec communication.

COMPUTERWORLD

Topics News In Depth Reviews Blogs Opti

Blogs



IT Blogwatch

A Daily Digest of IT Blogs from Richi Jennings

More posts | Read bio

February 16, 2012 - 9:00 A.M.

RSA crypto: 'flawed', 'risky', 'quagmire of vulnerabilities'

2 Comments

Like 5 +7 1

TAGS: certificate, crypto, cryptography, Diffie-Hellman, encryption, key management, RSA, SSL, IT TOPICS, Applications, Cyber crime & Hacking, Financial IT, Government & Regulation, Internet, Privacy, Security, Security Hardware & Software

A group of six academic researchers have concluded that real-world RSA encryption keys are riskier than Diffie-Hellman-based ones. It seems that some of the random numbers used to generate them weren't, erm, random. In [IT Blogwatch](#), bloggers wonder if the sky is falling.



The Register

Hardware Software Music & Media Networks Security Cloud Public Sector Business Science

Critics Malware Enterprise Security Spam D

Print Comment Tweet Like 1 Alert

'Predictably random' public keys can be cracked - crypto boffins
Battling researchers argue over whether you should panic
 By **John Leyden** - [Get more from this author](#)
 Posted in Enterprise Security, 16th February 2012 13:38 GMT


Analysis Cryptography researchers have discovered flaws in the key generation that underpins the security of important cryptography protocols, including SSL.

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

"Use y225a5y/8 if I ever send you an encrypted message"



Chat Client 1.0 (Alice)

[alice]: Hey Bob
 [bob]: Hi Alice!
 [alice]: gx76w3v7k


Encrypt SENDMONEY with y225a5y/8

Chat Client 1.0 (Bob)

[alice]: Hey Bob
 [bob]: Hi Alice!
 [alice]: gx76w3v7k

Decrypt gx76w3v7k with y225a5y/8

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

↓ encrypt

g X 7 6 W 3 v 7 K

↓ decrypt

S E N D M O N E Y

Encryption Machine

Goal Design a machine to encrypt and decrypt data

S E N D M O N E Y

↓ encrypt


g X 7 6 W 3 v 7 K

↓ decrypt

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]

- **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]

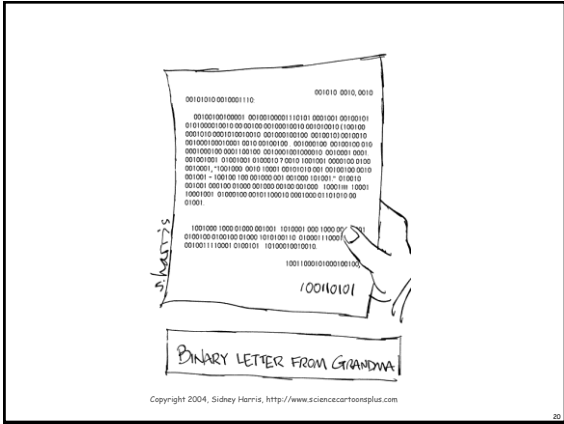
- **Text**
- Programs, executables
- Documents, pictures, sounds, movies, ...

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


computer with a cash dispenser


computer with a ballot box


computer with a heating element



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.

← very weak type of encryption

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

One-Time Pad Encryption

Encryption

- Convert text message to N bits

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

message
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

message
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

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
110010	010011	110110	111001	111001	100010	111111	010010	
100000	010111	111011	111010	010110	110111	111011	111011	001010

message
base64
random bits
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

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
110010	010011	110110	111001	011010	111001	100010	111111	010010
100000	010111	111011	111010	010110	110111	101111	111011	001010

message
base64
random bits
XOR

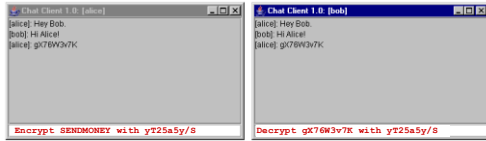
g	x	7	6	w	3	v	7	k
---	---	---	---	---	---	---	---	---

encrypted

Secure Chat (review)

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



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

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

XOR Truth Table

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

33

One-Time Pad Decryption (with the wrong pad)

Decryption

- Convert encrypted message to binary

g x 7 6 w 3 v 7 k encrypted

33

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

34

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**

35

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

36

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**

37



38

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

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

39

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 "random" bits

- Enigma machine
- **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)

40

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

register

41

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

LFSR demo

time t

time t + 1

42

Random Numbers

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

```

11001001001110110111001010101110010000101111101000001001101001011100110010011111
101100000101100000001110100011000000000000000000000000000000000000000000000000
01010100001000000101100010010101011100010100101110011010110100000100001001010
0111010000010100000100011010101110000000101100001001100001011010101010100100000
1111111001100000111000110000101110011000110001100110110101011100000000010000010111
000000000000000000010000000000000000000000000000000000000000000000000000000000
000000010000000010000000100000010101000000010100000111010001011010111011100001010
0101000010000001000010010101000011000010000000000000000000000000000000000000000
0101000010000000000101010100000000000000000000000000000000000000000000000000000
0110101001110000110001001111111101000000000000000000000000000000000000000000000
00001010110101011010101101010101010101010101010101010101010101010101010101010101
10010100011101010101010101010101010101010101010101010101010101010101010101010101
01100100011101010101010101010101010101010101010101010101010101010101010101010101
01001000110011000110001100011000110001100011000110001100011000110001100011000110001
00001000100110011100011000110011010101010101010101010101010101010101010101010101
100110001001101010101010101010101010101010101010101010101010101010101010101010101
01100100011101010101010101010101010101010101010101010101010101010101010101010101
100110001001101010101010101010101010101010101010101010101010101010101010101010101
1100100010011001000100110010001001100100010011001000100110010001001100100010011001
01100100010011001000100110010001001100100010011001000100110010001001100100010011001
0100100010011001000100110010001001100100010011001000100110010001001100100010011001
0100101010110000000011000000101100000110110011010101010101010101010101010101010101

```

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

43

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
010010	000100	001101	000011	001100	001110	001101	000100	011000
110010	010011	110110	111001	011010	111001	100010	111111	010010
100000	010111	111011	111010	010110	110111	101111	111011	001010
g	x	7	6	w	3	v	7	k

message
base64
LFSR bits
XOR
encrypted

44

LFSR Decryption

LFSR Decryption

- Convert encrypted message to N bits
- 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
100000	010111	111011	111010	010110	110111	101111	111011	001010
110010	010011	110110	111001	011010	111001	100010	111111	010010
010010	000100	001101	000011	001100	001110	001101	000100	011000
S	E	N	D	M	O	N	E	Y

encrypted
base64
LFSR bits
XOR
message

45

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]

46

Other LFSR Applications

What else can we do with a LFSR?

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



DVD Jon
(Norwegian hacker)

```
/* efdtt.c Author: Charles M. Rønne <root@hack.net> */
/* Usage is: cat title=key scrambled.vob | efdtt >clear.vob */
#define M(x) (x[(x)>4])<<

unsigned char x[5] , y,[2048],min(
n)for( read(0,x,5 );read(0,y,>=2048
); write(0 ,y,n) );
{y=x [13]^(x[20] /16)&&1 }list
j=1 [13] ^256 *M(0) &A ==2)
0,j= M(4) 17* M(3) 9*x^ 2-&A8
^B,A ==0,c ^2& for (i[y] --1;
--n; j +=2)w= a^7*1& 1,&w /2^1&2
<<2&f;for(j= 127; ++j;n;w>
9)
0
w+=1*(8^1>>4^1>>12,
1<>>8^y<17,a^w>>4,y^w^8^ack6,y^w
>>8^y<6,3&w[j],B ==7&w<121&1&
47)^2^<=c3&w<6;w>>n, [b>>4]^2^k^257/
8,w[j]^k^ (k&&^2&4) ^6^c^y
j)
}
http://www.cis.umd.edu/dst/DeCSS/Dallary
```

47

LFSR vs. 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	4 GB
input	seed	sequence of bits
computation	shift, XOR	logic, arithmetic, ...
output	pseudo-random bits	Sequence of bits

Critical difference. General-purpose computer can be programmed to simulate **any** abstract machine

48

A Profound Idea

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

- Basis for theoretical understanding of computation
- Basis for bootstrapping real machines into existence

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

    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
1100100100111011011011100101101
01110011000101011110100100001
001101001011110001100100111...
```

49

A Closing 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 truly 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



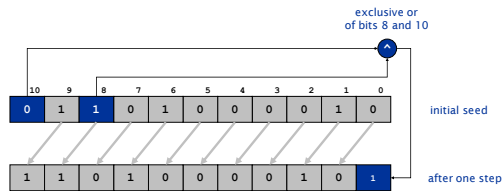
50

Extra Slides

Introduction to Programming in Java: An Interdisciplinary Approach · Robert Sedgwick and Kevin Wayne · Copyright © 2008 · *

51

Linear Feedback Shift Register



One step of an 11-bit LFSR with initial seed 01101000010 and tap at position 8

52