Prologue: A Simple Machine

• Brief introduction
• Secure communication with a one-time pad
• Linear feedback shift registers
• Implications

What is this course about?

A broad introduction to computer science.

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

Topics
• Programming in Java.
• Design and architecture of computers.
• Theory of computation.
• Applications in science and engineering.

"Science is everything we understand well enough to explain to a computer." — Don Knuth

"Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate, and brilliant; together they are powerful beyond imagination." — Albert Einstein
Prologue: A Simple Machine

- Brief introduction
- Secure communication with a one-time pad
- Linear feedback shift registers
- Implications

Sending a secret message with a cryptographic key

Alice wants to send a secret message to Bob.
- Sometime in the past, they exchange a cryptographic key.
- Alice uses the key to encrypt the message.
- Bob uses the same key to decrypt the message.

Critical point: Without the key, Eve cannot understand the message.

Q. How does the system work?

Encrypt/decrypt methods

Goal. Design a method to encrypt and decrypt data.

<table>
<thead>
<tr>
<th>S</th>
<th>E</th>
<th>N</th>
<th>D</th>
<th>M</th>
<th>O</th>
<th>N</th>
<th>E</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
<td>N</td>
<td>D</td>
<td>M</td>
<td>O</td>
<td>N</td>
<td>E</td>
<td>Y</td>
</tr>
</tbody>
</table>

Example 1. Enigma encryption machine [German code, WWII]
- Broken by Turing bombe (one of the first uses of a computer).
- Broken code helped win Battle of Atlantic by providing U-boat locations.

Example 2. One-time pad [details to follow]

Example 3. Linear feedback shift register [later this lecture]

A digital world

A bit is a basic unit of information.
- Two possible values (0 or 1).
- Easy to represent in the physical world (on or off).

In modern computing and communications systems, we represent everything as a sequence of bits.

- Text [details to follow in this lecture]
- Numbers
- Sound [details to follow in this course]
- Pictures [details to follow in this course]
- ...
- Programs [profound implications, stay tuned]

Bottom line. If we can send and receive bits, we can send and receive anything.
**Encryption with a one-time pad**

**Preparation**
- Create a "random" sequence of bits (a one-time pad).
- Send one-time pad to intended recipient through a secure channel.

**Encryption**
- Encode text as a sequence of N bits.
- Use the first N bits of the pad.
- Compute a new sequence of N bits from the message and the pad.
- Decrypt to get a sequence of characters.

Result: A **ciphertext** (encrypted message).

![Encryption Diagram](image)

**One-Time Pads**

**What is a one-time pad?**
- A cryptographic key known only to the sender and receiver.
- Good choice: A **random** sequence of bits (stay tuned).
- Security depends on each sequence being used only once.

![One-Time Pad Diagram](image)

**Encryption**
- **Message** as a sequence of bits in message.
- **One-time pad** as a sequence of bits in the pad.
- **Ciphertext** as a sequence of bits in the encrypted message.

![Encryption Process](image)
Pop quiz on bitwise XOR encryption

Q. Encrypt the message E A S Y with the pad 0 1 2 3.

Decryption with a one-time pad

Sending a secret message with a cryptographic key

Alice wants to send a secret message to Bob.
- Sometimes in the past, they exchange a cryptographic key.
- Alice uses the key to encrypt the message.
- Bob uses the same key to decrypt the message.

Critical point: Without the key, Eve cannot understand the message.

A. Alice's device uses a "bitwise exclusive or" machine to encrypt the message.

Q. What kind of machine does Bob's device use to decrypt the message?

A. The same one (!)

A (very) simple machine for encryption and decryption

To compute a message from a ciphertext and a one-time pad
- Use binary encoding of ciphertext and pad.
- Each message bit is the bitwise exclusive or of corresponding bits in ciphertext and pad.

SENDMONEY

# SENDMONEY
Why does it work?

<table>
<thead>
<tr>
<th>S</th>
<th>E</th>
<th>N</th>
<th>D</th>
<th>M</th>
<th>O</th>
<th>N</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>010010000100001101000011001100001110001101000100011000</td>
<td>01000011011010110100001110011010001010011000111001100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ciphertext**

SENDMONEY

**message**

SENDMONEY

**one-time pad**

01000011011010110100001110011010001010011000111001100

**one-time pad**

010010000100001101000011001100001110001101000100011000

**XOR**

Crucial property: Decrypted message is the same as the original message. Let m be a bit of the message and k be the corresponding bit of the one-time pad. To prove: $(m \oplus k) \oplus k = m$ (Notation: $m \oplus k$ is equivalent to XOR(m, k))

### Approach 1: Truth tables

<table>
<thead>
<tr>
<th>m</th>
<th>k</th>
<th>m &amp; k</th>
<th>(m &amp; k) \oplus k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Approach 2: Boolean algebra

$(k \& k) = 0$

$m \oplus 0 = m$

$(m \& k) \oplus k = m \& (k \oplus k)$

$= m \& 0$

$= m$ ✓

### Decryption with the wrong pad

Eve cannot read a message without knowing the pad.

**ciphertext**

100000010111111011111010010110110111101111111011001010

**wrong pad**

001010100111111000011010001101110100000001100001100101

**gibberish**

100000010111111011111010010110110111101111111011001010

### One-time pad is provably secure [Shannon, 1940s]

- IF each pad is used only once,
- AND the pad bits are random,
- THEN Eve cannot distinguish ciphertext from random bits.

### Eve’s problem with one-time pads

Eve has a computer. Why not try all possibilities?

**Problem**

- 54 bits, so there are $2^{54}$ possible pad values.
- Suppose Eve could check a million values per second.
- It would still take 570+ years to check all possibilities.

**Much worse problem**

- There are also $2^{54}$ possible messages.
- If Eve were to check all the pads, she’d see all the messages.
- No way to distinguish the real one from any other.

**One-time pad is provably secure.**

### Goods and bads of one-time pads

**Goods.**

- Very simple encryption method.
- Decrypt with the same method.
- Provably unbreakable if bits are truly random.
- Widely used in practice.

**Bads.**

- Easily breakable if seed is re-used.
- Truly random bits are very hard to come by.
- Need separate secure channel to distribute key.
- Pad must be as long as the message.
Random bits are not so easy to find
You might look on the internet.

The randomness comes from atmospheric noise

... if you trust the internet.

Next: Creating a (long) sequence of “pseudo-random” bits from a (short) key.

A pseudo-random number generator
is a deterministic machine that produces a long sequence of pseudo random bits.

Examples
- Enigma.
- Linear feedback shift register (next).
- Blum-Blum-Shub generator.

... [an early application of computing]
[research still ongoing]

“Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.”

— John von Neumann
A pseudo-random number generator

is a deterministic machine that produces a long sequence of pseudo random bits.

**Deterministic:** Given the current state of the machine, we know the next bit.

An absolute requirement: Alice and Bob need the same sequence.

**Random:** We never know the next bit.

**Pseudo-random:** The sequence of bits appears to be random.

**Appears to be random??**

- A profound and elusive concept.
- For this lecture: "Has enough properties of a random sequence that Eve can’t tell the difference".

A pseudo-random number generator

is a deterministic machine that produces a long sequence of pseudo random bits.

**Deterministic:** Given the current state of the machine, we know the next bit.

An absolute requirement: Alice and Bob need the same sequence.

**Random:** We never know the next bit.

**Pseudo-random:** The sequence of bits appears to be random.

**Appears to be random??**

- A profound and elusive concept.
- For this lecture: "Has enough properties of a random sequence that Eve can’t tell the difference".

**Linear feedback shift register**

**Terminology**

- **Bit:** 0 or 1.
- **Cell:** storage element that holds one bit.
- **Register:** sequence of cells.
- **Seed:** initial sequence of bits.
- **Feedback:** Compute XOR of two bits and put result at right.

**An [11, 9] LFSR**

0 1 1 0 1 1 0 0 0 1 0 1

11 10 9 8 7 6 5 4 3 2 1

**More terminology**

- **Tap:** Bit positions used for XOR (one must be leftmost).
- **[N, k] LFSR:** N-bit register with taps at N and k.

### Linear feedback shift register simulation

<table>
<thead>
<tr>
<th>History of register contents</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1 0 0 0 0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1 0 0 0 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0 0 0 1 0 1 1</td>
<td>2</td>
</tr>
<tr>
<td>1 0 0 0 0 1 0 1 1 0</td>
<td>3</td>
</tr>
<tr>
<td>0 0 0 0 1 0 1 1 0 0</td>
<td>4</td>
</tr>
<tr>
<td>0 0 0 1 0 1 1 0 0 1</td>
<td>5</td>
</tr>
</tbody>
</table>

**History of register contents**

- **0 1 1 0 1 0 0 0 0 1 0**: Numbered from right, starting at 1.
- **Not all values of k give desired effect (stay tuned).**

**Time**

- **0 1 2 3 4 5**

**a pseudo-random bit sequence!**

**Linear feedback shift register simulation**

<table>
<thead>
<tr>
<th>History of register contents</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1 0 0 0 0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 0 1 0 0 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0 0 0 1 0 1 1</td>
<td>2</td>
</tr>
<tr>
<td>1 0 0 0 0 1 0 1 1 0</td>
<td>3</td>
</tr>
<tr>
<td>0 0 0 0 1 0 1 1 0 0</td>
<td>4</td>
</tr>
<tr>
<td>0 0 0 1 0 1 1 0 0 1</td>
<td>5</td>
</tr>
</tbody>
</table>

**History of register contents**

- **0 1 1 0 1 0 0 0 0 1 0**: Numbered from right, starting at 1.
- **Not all values of k give desired effect (stay tuned).**

**Time**

- **0 1 2 3 4 5**

**a pseudo-random bit sequence!**

**Linear feedback shift register simulation**

<table>
<thead>
<tr>
<th>History of register contents</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 1 0 0 0 0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 0 1 0 0 0 1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0 0 0 1 0 1 1</td>
<td>2</td>
</tr>
<tr>
<td>1 0 0 0 0 1 0 1 1 0</td>
<td>3</td>
</tr>
<tr>
<td>0 0 0 0 1 0 1 1 0 0</td>
<td>4</td>
</tr>
<tr>
<td>0 0 0 1 0 1 1 0 0 1</td>
<td>5</td>
</tr>
</tbody>
</table>

**History of register contents**

- **0 1 1 0 1 0 0 0 0 1 0**: Numbered from right, starting at 1.
- **Not all values of k give desired effect (stay tuned).**

**Time**

- **0 1 2 3 4 5**

**a pseudo-random bit sequence!**
A random bit sequence?

Q. Is this a random sequence?

Looks random to me.

It is pseudo-random (at least to some observers).

A. No. It is the output of an [11, 9] LFSR with seed 01101000010!

A random bit sequence?

one-time pad in our example

Pop quiz on LFSRs

Q. Give first 10 steps of [5, 4] LFSR with initial fill 00001.

Pop quiz on LFSRs

Q. Give first 10 steps of [5, 4] LFSR with initial fill 00001.

Encryption/decryption with an LFSR

Preparation
• Alice creates a book of “random” (short) seeds.
• Alice sends the book to Bob through a secure channel.

Encryption/decryption
• Alice sends Bob a description of which seed to use.
• They use the specified seed to initialize an LFSR and produce N bits.
  [and proceed in the same way as for one-time pads]

Encryption/decryption with an LFSR

message SENDMONEY

seed 0110000010

ciphertext YATSMYTTA

seed 0110000010

message SENDMONEY

KOR
Eve’s opportunity with LFSR encryption

Without the seed, Eve cannot read the message.

Eve has computers. 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.

Good news (for Eve): This approach can work.
• Ex: 11-bit register implies 2047 possibilities.
• Extremely likely that only one of those is not gibberish.
• After this course, you could write a program to check whether any of the 2047 messages have words in the dictionary.

Bad news (for Eve): It is easy for Alice and Bob to use a much longer LFSR.

Key properties of LFSRs

Property 1.
• Don’t use all 0s as a seed!
• Fill of all 0s will not otherwise occur.

Property 2. Bittstream must eventually cycle.
• $2^n - 1$ nonzero fills in an $N$-bit register.
• Future output completely determined by current fill.

Property 3. Cycle length in an $N$-bit register is at most $2^n - 1$.
• Could be smaller; cycle length depends on tap positions.
• Need theory of finite groups to know good tap positions.
Key properties of LFSRs

**Property 1.**
- Don’t use all 0s as a seed!
- Fill of all 0s will not otherwise occur.

**Property 2.** Bitstream must eventually cycle.
- \(2^n - 1\) nonzero fills in an \(N\)-bit register.
- Future output completely determined by current fill.

**Property 3.** Cycle length in an \(N\)-bit register is at most \(2^N - 1\).
- Could be smaller; cycle length depends on tap positions.
- Need theory of finite groups to know good tap positions.

Bottom line.
- [1, 9] register generates 2047 bits before repeating.
- [63, 62] register generates \(2^{63} - 1\) bits before repeating.

Definitely preferable: small cost, huge payoff.

---

Eve’s problem with LFSR encryption

Without the seed, Eve cannot read the message.

Eve has computers. 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 way too many possibilities.
- Ex: 63-bit register implies \(2^{63} - 1\) possibilities.
- If Eve could check 1 million seeds per second, it would take her 2923 centuries to try them all.

Bad news (for Alice and Bob): LFSR output is not random.

---

Goods and bads of LFSRs

**Goods.**
- Very simple encryption method.
- Decrypt with the same method.
- Scalable: 20 cells for 1 million bits; 30 cells for 1 billion bits.
- Widely used in practice. [Example: military cryptosystems.]

**Bads.**
- Easily breakable if seed is re-used.
- Still need secure key distribution.
- Experts can crack LFSR encryption.

**Example.**
- CSS encryption widely used for DVDs.
- Widely available DeCSS breaks it!
A Profound Idea

**Programming.** We can write a Java program to simulate the operation of any abstract machine.
- Basis for theoretical understanding of computation.
- Basis for bootstrapping real machines into existence. Stay tuned (we cover these sorts of issues in this course).

```java
public class LFSR {
    public static void main(String[] args)
    {
        int[] a = { 0, 0, 1, 0, 0, 0, 1, 0, 1, 0 }; 
        for (int t = 0; t < 2000; t++)
        { 
            a[0] = (a[1]) ^ a[9]);
            System.out.println(a[0]);
            for (int t = 11; i > 0; i--) 
            { 
                a[i] = a[i-1];
            }
            System.out.println();
        }
    }
}
```

Profound questions

**Q.** What is a random number?

- LFSRs do not produce random numbers.
  - They are deterministic. (von Neumann’s “state of sin”: we know that “deterministic” is incompatible with “random”)
  - It is not obvious how to distinguish the bits LFSRs produce from random,
    - BUT 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??

“No God does not play dice.”

— Albert Einstein
Image sources

http://commons.wikimedia.org/wiki/File:SpaceAndOpenContentAlliance.jpg
http://commons.wikimedia.org/wiki/File:Einstein-formal_portrait-35.jpg

Prologue:
A Simple Machine