Encryption

Secure Chat

Alice wants to send a secret message to Bob?

- Sometimes 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.

```
| E | N | C | R | Y | P | T | O | N |
```

```
| D | E | C | R | Y | P | T | O | N |
```

```
encrypt
```

```
decrypt
```
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. \([\text{bit} = 0 \text{ or } 1]\)
- Text.
- Programs, executables.
- Documents, pictures, sounds, movies, ...

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

One-Time Pad Encryption

Encryption.
- Convert text message to \(N\) bits.

<table>
<thead>
<tr>
<th>Base64 Encoding</th>
<th>Char</th>
<th>hex</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0100</td>
<td>0000</td>
<td>001000</td>
</tr>
<tr>
<td>K</td>
<td>0101</td>
<td>0001</td>
<td>001001</td>
</tr>
<tr>
<td>X</td>
<td>0110</td>
<td>0010</td>
<td>001010</td>
</tr>
<tr>
<td>N</td>
<td>0111</td>
<td>0011</td>
<td>001011</td>
</tr>
</tbody>
</table>

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

Very weak type of encryption.

Binary letters from grandma.
One-Time Pad Encryption

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

<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>010010</td>
<td>000100</td>
<td>001101</td>
<td>000011</td>
<td>001100</td>
<td>001110</td>
<td>001101</td>
<td>000100</td>
<td>011000</td>
</tr>
</tbody>
</table>

Message: SEND MONEY

Base64: S3NkMDQkO3MkMDI=

Random Bits:

<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>011010</td>
<td>001101</td>
<td>001110</td>
<td>001101</td>
<td>000100</td>
<td>010011</td>
<td>010010</td>
<td>011000</td>
<td>010010</td>
</tr>
</tbody>
</table>

XOR:

<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>010000</td>
<td>010111</td>
<td>111011</td>
<td>111010</td>
<td>010110</td>
<td>110111</td>
<td>101111</td>
<td>111011</td>
<td>001010</td>
</tr>
</tbody>
</table>

Encrypted:

gX76W3v7K

Base64: S3NkMDQkO3MkMDI=

One-Time Pad Decryption

Decryption.
- Convert encrypted message to binary.

| g | X | 7 | 6 | W | 3 | v | 7 | K |

Encrypted: gX76W3v7K

Base64: S3NkMDQkO3MkMDI=
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.

<table>
<thead>
<tr>
<th>g</th>
<th>X</th>
<th>7</th>
<th>6</th>
<th>W</th>
<th>3</th>
<th>v</th>
<th>7</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>110010</td>
<td>010011</td>
<td>110110</td>
<td>111001</td>
<td>011010</td>
<td>111001</td>
<td>100010</td>
<td>111111</td>
<td>010010</td>
</tr>
</tbody>
</table>

**Crucial property.** Decrypted message = original message.

**Why is crucial property true?**
- Use properties of XOR:
  - \((a \oplus b) \oplus b = a \oplus (b \oplus b) = a \oplus 0 = a\)

**XOR Truth Table**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>x \oplus y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Why Does It Work?

One-Time Pad Decryption (with the wrong pad)

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

<table>
<thead>
<tr>
<th>g</th>
<th>X</th>
<th>7</th>
<th>6</th>
<th>W</th>
<th>3</th>
<th>v</th>
<th>7</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>110010</td>
<td>010011</td>
<td>110110</td>
<td>111001</td>
<td>011010</td>
<td>111001</td>
<td>100010</td>
<td>111111</td>
<td>010010</td>
</tr>
</tbody>
</table>

Decryption (with the wrong pad)

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

<table>
<thead>
<tr>
<th>g</th>
<th>X</th>
<th>7</th>
<th>6</th>
<th>W</th>
<th>3</th>
<th>v</th>
<th>7</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>110010</td>
<td>010011</td>
<td>110110</td>
<td>111001</td>
<td>011010</td>
<td>111001</td>
<td>100010</td>
<td>111111</td>
<td>010010</td>
</tr>
</tbody>
</table>

Decryption (with the wrong pad)

**Decryption**
- Convert encrypted message to binary.
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.
  - encrypted
    - g X 7 6 W 3 v 7 R
      - 0000 0101 1001 1111 0110 0000 0010 0111
      - 1101 1111 1110 0101 0110 1110 1011 0000
  - xor
    - 0010 0000 0010 1110 0011 1101 0100 0011

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]

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.

Goods and Bads of One-Time Pads

Pseudo-Random Bit Generator

Practical middle-ground.
- Let’s make a “random”-bit generator gadget.
- Alice and Bob each get identical small gadgets.

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."
- von Neumann (left)
- ENIAC (right)
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.

0 1 1 0 1 0 0 0 0 1 0

Time t

0 1 0 0 0 0 1 0

Time t + 1

Linear Feedback Shift Register (LFSR)

(8, 10) linear feedback shift register.
- Shift register with 11 cells.
- Bit b_9 is XOR of previous bits b_6 and b_10.
- Pseudo-random bit = b_6.

Random Numbers

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

A. No. This is output of (8, 10) LFSR with seed 0101000010!

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.

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.

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]

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.

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?

http://www.cs.cmu.edu/~dst/DeCSS/Gallery

DVD Jon (Norwegian hacker)

Extra Slides

Linear Feedback Shift Register

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

exclusive or of bits 8 and 10