Overview

Today:
- Cryptography concepts and classical crypto
  - Textbook sections 3.1, 3.2 (except Hill cipher), 3.5
- Working in Binary

To do before Thursday:
- Study for quiz on HW1!
- Read Sections 4.1, 4.2, 4.4
- Start talking to project team members to solidify project ideas

Introduction to Cryptography
Confidentiality Protection for Messages

Plaintext: The message the sender wants to send.
Ciphertext: The data that is transmitted.
Goal: Ciphertext should reveal no information about plaintext to anyone who doesn't have the decryption key.
Introduction to Cryptography
Confidentiality Protection for Messages

Keys the same or different?
The same: "Symmetric cipher"
- Some algorithms: DES, AES, Blowfish, RC4 (probably best to avoid...)
- Best feature: fast!
  (AES-NI gives 5-10 Gbps)
- Worst feature: difficult key management.

Different: "Public-key crypto"
- Some algorithms: RSA, ElGamal, ECC, ...
- Best feature: Simpler key management (can send to a stranger!)
- Worst feature: Slow (1-2 Mbps for 2048-bit RSA - others are a little faster...)

Some Terminology

Cryptography: Making codes
Cryptanalysis: Breaking codes
Cryptology: The science of both (generally "cryptography" now)

Participants traditionally given names:
- Alice and Bob are legitimate users
- Trent is a "trusted third party"
- Eve is a passive adversary (an eavesdropper)
- Mallory is an active adversary (malicious...)

Encipher and encrypt are synonyms (also decipher/decrypt)

Written as functions:
- \( C = E(K_e, P) \)
- \( P = D(K_d, C) \)

- \( K_e \): “Keyspace”
- \( P \): “Plaintext space”
- \( C \): “Ciphertext space”
Kerckhoff's Principle

The book (section 3.1) talks about "two requirements for secure use of conventional encryption" - these requirements are from:

**Kerckhoff's Principle (1883):** The security of a cryptosystem depends on the **strength** of the algorithm and the **secrecy** of the key.

Trying to keep algorithms secret ("security through obscurity") almost never works.
- DVD Content Scrambling System (CSS)
- Mobil Speedpass
- Every digital rights management system ever… (a slightly different issue)

Remember design principles: Open Design
- Better to use a system that experts have pounded on (and failed to break)

---

Block vs Stream Ciphers

**Block Ciphers**
- Must be given a minimum amount of data
- Typical symmetric cipher blocks: 64 or 128 bits
- If not enough data to fill a block, must either
  - Wait for more data, or
  - Pad the block with extra bits

**Stream Ciphers**
- Work in small units - bits or bytes
- Bit-oriented stream cipher: one bit in, one bit out
- Consider interactive terminal session...

---

Attacker Information/Access

What information/access does the attacker have?

Increasing Attacker Power

- Ciphertext Only
- Known Plaintext
- Chosen Plaintext
- Chosen Ciphertext

Real-world examples for all models

Interesting point: In the 2014 movie The Imitation Game, "breakthrough" in cracking German code was basically shifting model from "ciphertext only" to "known plaintext"
Types of Attacks

Cryptanalysis
- Analyzes ciphertext/algorithm for patterns or structural properties to get information
- Example: If most keys used by a cipher result in “a” being replaced by “M”, then that’s a big clue!
- Can lead to very fast attacks on weak encryption algorithms!

Brute Force
- Try every possible key to see which produces a “sensible” plaintext
  - Need to distinguish sensible plaintext from non-sensible
- Average tests required to break: \(|K|/2\) (half the keyspace size)

Question: Given a baseline of 1 billion tests/second, how big does the keyspace need to be for brute force to be impractical (use powers of 2).

Classical Cryptography

Generalized Caesar Cipher

Generalized Caesar Cipher: Shift by \(k\) places

Example: Shift \(k = 5\) places

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>...</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cipher</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>...</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

Keyspace size: \(|K| = 26\)

Trivial size to brute force, looking for sensible English.

Classical Cryptography

Arbitrary Monoalphabetic Substitution

Arbitrary substitute: Any one-to-one mapping can be used

Example:

<table>
<thead>
<tr>
<th>Plaintext</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>...</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cipher</td>
<td>P</td>
<td>F</td>
<td>J</td>
<td>Y</td>
<td>N</td>
<td>L</td>
<td>...</td>
<td>O</td>
<td>D</td>
<td>M</td>
</tr>
</tbody>
</table>

Keyspace size: \(|K| = 26! = 403,291,461,126,605,635,584,000,000,000\)

\(= 4 \times 10^{26}\)

Testing 1 billion keys / second takes \(4 \times 10^{20}\) sec = 128 million centuries

And yet…. People solve these all the time for fun (Cryptograms) - how?

Cryptanalysis! Letter frequencies, patterns, ...
Classical Cryptography

Vigenère Polyalphabetic Substitution

Idea: Have a sequence of shifts \((k_1, k_2, \ldots, k_p)\) as key
- After all \(p\) are used, start over with \(k_1\)
- \(p\) is the period of the cipher
- Since different positions use different substitutions, evens out frequencies

Example with key \((4,1,22,12)\):

| Plaintext: secret i phone plans |
| Shift: 4 1 22 12 4 1 22 12 4 1 22 12 4 |
| Ciphertext: W F Y D I U E B L P J Q T M W Z k |

Questions for the class to answer:
- If our alphabet has 64 values (26 upper case, 26 lower, 10 digits, 2 punctuation), what is keyspace size a given \(p\)?
- How large does \(p\) have to be for this to be out of range of brute force attacks?

Important: Don’t use, even with large \(p\) - not stuck with brute force, as there are good cryptanalytic attacks.

Classical Cryptography

One-Time Pad - On Letters

Idea: Vigenère key repeats after \(p\) positions. So don’t repeat!
- Requires key to be as long as plaintext
- Key should be picked randomly (uniform distribution)


Ciphertext: GIFKMB
Key test 1: G0QKBXK
Key test 2: F0NSTXI

Question: What is the probability that test key 1 is used by sender? What about test key 2? Any reason to believe, as the attacker, that one is more probable than the other?

Recall from brute-force: “Need to distinguish sensible plaintext from non-sensible”

More on one-time pad security after talking about binary operators...

Binary Operations

AND and OR

Recall basic bitwise operations
(Operands are really symmetric, but often thought of as “data” and “mask”)

| 1001101 \(\text{(data)}\) | 1001101 \(\text{(data)}\) |
| 00001111 \(\text{(mask)}\) | OR 00001111 \(\text{(mask)}\) |
| 00001101 | 10011111 |

AND operation:
- “0” position in mask are cleared
- “1” position in mask are copied

OR operation:
- “0” position in mask are copied
- “1” position in mask are set

Widely used (with shift operators) for manipulating individual bits or packing small data fields into single bytes/words.
Binary Operations

Exclusive OR

<table>
<thead>
<tr>
<th>1001101 (data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR 0101010 (mask)</td>
</tr>
<tr>
<td>1100100</td>
</tr>
</tbody>
</table>

XOR operation:
- "0" position in mask are copied
- "1" position in mask are flipped

Writing as a formula: for bytes/words/bitvectors x and y, use "x ⊕ y"

Question 1: What do you think ((x ⊕ y) ⊕ y) is?

Question 2: If y is chosen as a completely random bitvector:
- What is the probability that the first bit of x ⊕ y is 0? Is 1?
- What is the probability that the last bit of x ⊕ y is 0? Is 1?

One-Time Pad On Bytes

Idea: Same as with letters, but use XOR instead of alphabet shift

- Let m be a b-bit long plaintext message
- Let k be a b-bit long random bitvector (uniformly distributed)
- Calculate ciphertext \( c = m \oplus k \)

Consider captured ciphertext \( c \) and to possible plaintext messages \( m_1 \) and \( m_2 \)

- No a priori reason to think \( m_1 \) or \( m_2 \) is more likely
- Possibility 1: \( m_1 \) was the message - key is \( k_1 = c \oplus m_1 \)
- Possibility 2: \( m_2 \) was the message - key is \( k_2 = c \oplus m_2 \)
- \( \text{Prob}(k_1 \text{ chosen}) = \text{Prob}(k_2 \text{ chosen}) = 1/2^b \)

Bottom line: Every b-bit long message is possible, each with equally likely keys

Perfect confidentiality - as long as you never re-use any portion of the key!

Example of failure to use properly: Venona

One-Time Pad

Is perfect confidentiality perfect security?

Scenario of an instructor sending a grade to registrar using OTP:

Alice (instructor) sends a message containing grade 'F': char value 0x46
Uses OTP key 0x0D9 → ciphertext is 0x9F

Mallory intercepts message (0x9F) and XORs with 'F'\( \oplus \)A = 0x46\( \oplus \)0x41 = 0x07
→ 0x9F\( \oplus \)0x07 = 0x88

Bob (registrar) receives message 0x88 and XORs with OTP key 0x0D9
→ 0x88\( \oplus \)0x0D9 = 0x41 = 'A'

OTP is a malleable cipher: An active attacker can make a change to the ciphertext that will make a predictable change in the plaintext recovered by the receiver.

Bottom line: OTP has perfect confidentiality, but is very hard to use (key management) and is very weak with respect to message integrity.
Steganography

Hiding the existence of a message

This picture has a secret message embedded.

Steganography

Hiding the existence of a message

The message was “On the Internet, nobody knows you’re a dog.”

It was embedded using the “outguess” steganography software.