Cryptography

Lecture 01
Welcome!

- Crypto is amazing
  - Can do things that initially seem impossible

- Crypto is important
  - It impacts us every day

- Crypto is fun!
  - Deep theory
  - Attackers’ mindset
Necessary administrative stuff

- Course webpage:
  - Prerequisites/information posted here
  - Syllabus posted there
  - HWs posted there
  - Announcements posted there
  - Midterm already scheduled
Necessary administrative stuff

- Canvas/ELMS
  - Used only to submit homework electronically-Must be Typed
  - Let me know if unable to access

- Piazza
  - Useful for discussions/questions
  - Preferable to email if you think others will have the same question
TAs

- Nathan Grammel
- Jeremy Klein
- Dan McVicker
- Jacob Prinz
- Jake Yamada
This is a tough class

- Mathematical prerequisites
  - Discrete math, probability, modular arithmetic
- Requires mathematical maturity
  - Proofs, abstraction
This is a tough class

- CS prerequisites
  - Binary, hex, pseudocode, algorithms, big-O notation

- Programming assignments
  - Hard part should not be the programming, but the thought behind it
  - Flexibility in choice of language
“Active learning”

- Read textbook before class
  - See course syllabus

- Can also view Jon Katz’s videos on Coursera

- Ask questions in advance on Piazza and/or bring questions to class
HWs/exams

- HWs every week.
- Due Monday on Elms before class begins.
- **Sick Cat Policy:** can post to Elms Wed BEFORE CLASS without penalty
- **WARNING:** YOU have already been given an extension, HW solutions will be posted on Wed, so NO extensions past that.
- We will keep track of your lateness NOT for grade, but for letters.
- In-class midterm and final

Can buy on Amazon used.

Don’t tell Katz I said so.
Laptops/electronics

- No laptops/electronics policy
  - Distracting to you
  - Distracting to others

- If you feel you need an exception, talk to me
How to reach me

- Best way is by email: gasarch@cs.umd.edu
- Please put “CMSC456” in subject line
- Office hours before class
- Can also email TA’s (email addresses on syll)
Course goals

▶ Understand real-world crypto via a rigorous approach

▶ When you encounter crypto in your career
  ▶ Understand the key terms
  ▶ Understand the security guarantees provided
  ▶ Know how to use crypto
  ▶ Understand what goes on “under the hood”

▶ “Crypto mindset”
Course non-goals

- Designing your own crypto-schemes
- Implementing your own crypto for real-world use
- Course goal: Realize when to consult an expert!
Classical VS Modern cryptography

Classical: (1900 BCE?–1975)

2. They recruited WW II crypto analysts by asking for people that were good at crossword puzzles or linguistics.
3. Alan Turing and others brought more math into it, but not much math compared compared to Modern

Modern: (1976-today)

1. Lots of Math.
2. Lots of Rigor.
3. The notion of Provably Secure important.

Note: The cutoff of 1975–1976 is approximate.
## Rough course outline

<table>
<thead>
<tr>
<th></th>
<th><strong>Secrecy</strong></th>
<th><strong>Integrity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private-key setting</strong></td>
<td>Private-key encryption</td>
<td>Message authentication codes</td>
</tr>
<tr>
<td><strong>Public-key setting</strong></td>
<td>Public-key encryption</td>
<td>Digital signatures</td>
</tr>
</tbody>
</table>
Classical Cryptography
Motivation

- Allows us to “ease into things...”,
- Shows why unprincipled approaches are dangerous
- Illustrates why things are more difficult than they may appear
Alice, Bob, and Eve

- Alice sends a message to Bob in code.

- Eve overhears it.

- We want Eve to not be able to decode it.

This can mean one of two things:

- Eve does not have enough information to decode it. So even if Eve had unlimited computing power she could not decode.

- Assuming Eve can’t Factor quickly (or some other function) then Eve cannot break the code.
The First Step in Any Cipher-Spaces

I want to encode

*Cryptography is an important part of security*

Spaces give away information! For example, if I do SHIFT-BY-1 I get

*Dszuphsbqiz jt bo jnqpsubou qbsu pg tfdvsjuz*

Without any fancy math Eve knows that the second and third word are two letters long. Thats information she can use!

What to do?
The First Step in Any Cipher-Blocks of Five

I want to encode

*Cryptography is an important part of security*

Break it up into blocks of 5:

*Cryptography is an important part of security*

However you code it, spaces will not give anything away.
The First Step in Any Cipher—Other Issues

I want to encode

*Are my TAs for CMSC 456 awesome? All but Alex!*
The First Step in Any Cipher - Other Issues

I want to encode

Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.
The First Step in Any Cipher-Other Issues

I want to encode

Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.  
   Map everything to Capitols.
The First Step in Any Cipher-Other Issues

I want to encode

\[ \text{Are my TAs for CMSC 456 awesome? All but Alex!} \]

1. Capitol and small letters leak information.
   Map everything to Capitols.
2. Punctuation leaks information.
I want to encode

Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.
   Map everything to Capitols.
2. Punctuation leaks information.
   Get rid of all punctuation.
I want to encode

_Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.
   Map everything to Capitols.
2. Punctuation leaks information.
   Get rid of all punctuation.
3. What to do about numbers?

Note: In this class we will use 26-letter English only.
The First Step in Any Cipher—Other Issues

I want to encode

Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.
   Map everything to Capitols.
2. Punctuation leaks information.
   Get rid of all punctuation.
3. What to do about numbers?
   Use Mod 36
I want to encode

*Are my TAs for CMSC 456 awesome? All but Alex!*

1. Capitol and small letters leak information.
   Map everything to Capitols.
2. Punctuation leaks information.
   Get rid of all punctuation.
3. What to do about numbers?
   Use Mod 36
   More generally, set your mod equal to your alphabet size.
The First Step in Any Cipher—Other Issues

I want to encode

Are my TAs for CMSC 456 awesome? All but Alex!

1. Capitol and small letters leak information.
   Map everything to Capitals.

2. Punctuation leaks information.
   Get rid of all punctuation.

3. What to do about numbers?
   Use Mod 36
   More generally, set your mod equal to your alphabet size.

Note: In this class we will use 26-letter English only.
The Shift Cipher
The Shift Cipher

- Consider encrypting English text
- Associate ‘a’ with 0; ‘b’ with ‘; . . . ; ‘z’ with 25
- \( k \in \mathcal{K} = \{0, \ldots, 25\} \) (or could think of \( k \in \{a, \ldots, z\} \))
- To encrypt using key \( k \), shift every letter of the plaintext by \( k \) positions (with wraparound)
- Decryption just does the reverse

\[
\begin{align*}
\text{hello world} &+ 22222 22222 \\
\text{= jgnnq yqtnf}
\end{align*}
\]
Modular arithmetic

- $x = y \mod N$ if and only if $N$ divides $x-y$

- $[x \mod N] = \text{the remainder when } x \text{ is divided by } N$
  - i.e. the unique value $y \in \{0, \ldots, N-1\}$ such that $x = y \mod N$

- $25 = 35 \mod 10$

- $25 \neq [35 \mod 10]$

- $5 = [35 \mod 10]$
The Shift Cipher, Formally

- $\mathcal{M} = \{\text{strings over lowercase English alphabet}\}$
  All arithmetic mod 26.

- Choose uniform $k \in \{0, \ldots, 25\}$

- Encode $(m_1 \ldots m_t)$ as $(m_1 + k, \ldots m_t + k)$

- Decode $(c_1 \ldots c_t)$ as $(c_1 - k, \ldots c_t - k)$

- Can verify that correctness holds.
Is the Shift Cipher Secure?

- No – only 26 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - Only one possibility will “make sense”

- Example of a “brute-force” or “exhaustive-search” attack
Example

- Ciphertext uryyb jbeyq
- Try every possible key...
  - tqxxa iadxp
  - spwwz hzcwo
  - ...
  - hello world

Question: We can tell that hello world is correct but how can a computer do that. Can we mechanize the process of picking out the right one?
Letter Frequencies

<table>
<thead>
<tr>
<th>Letter</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>8.2</td>
</tr>
<tr>
<td>b</td>
<td>1.5</td>
</tr>
<tr>
<td>c</td>
<td>2.8</td>
</tr>
<tr>
<td>d</td>
<td>4.3</td>
</tr>
<tr>
<td>e</td>
<td>12.7</td>
</tr>
<tr>
<td>f</td>
<td>2.2</td>
</tr>
<tr>
<td>g</td>
<td>2.0</td>
</tr>
<tr>
<td>h</td>
<td>6.1</td>
</tr>
<tr>
<td>i</td>
<td>7.0</td>
</tr>
<tr>
<td>j</td>
<td>0.2</td>
</tr>
<tr>
<td>k</td>
<td>0.8</td>
</tr>
<tr>
<td>l</td>
<td>4.0</td>
</tr>
<tr>
<td>m</td>
<td>2.4</td>
</tr>
<tr>
<td>n</td>
<td>6.7</td>
</tr>
<tr>
<td>o</td>
<td>1.5</td>
</tr>
<tr>
<td>p</td>
<td>1.9</td>
</tr>
<tr>
<td>q</td>
<td>0.1</td>
</tr>
<tr>
<td>r</td>
<td>6.0</td>
</tr>
<tr>
<td>s</td>
<td>6.3</td>
</tr>
<tr>
<td>t</td>
<td>9.1</td>
</tr>
<tr>
<td>u</td>
<td>2.8</td>
</tr>
<tr>
<td>v</td>
<td>2.4</td>
</tr>
<tr>
<td>w</td>
<td>2.4</td>
</tr>
<tr>
<td>x</td>
<td>2.0</td>
</tr>
<tr>
<td>y</td>
<td>0.2</td>
</tr>
<tr>
<td>z</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Use Letter Frequencies to Tell What ”Looks Like English”

Let $T$ be a long test of normal English. Let $\vec{f}$ be the freq vector of English. The components are all between 0 and 1 and add up to 1. We assume freq vector of $T$ is approx $\vec{f}$.

- One can compute that
  \[
  \vec{f} \cdot \vec{f} \approx 0.065
  \]

- Let $s \in \{1, \ldots, 25\}$. Let $T_s$ be the text shifted by $s$. Let $\vec{g}$ be the freq vector for $T_s$. One can compute that
  \[
  \vec{f} \cdot \vec{g} \leq \approx 0.038
  \]
Is English

We describe a way to tell if a text Is English that we will use throughout this course.
Let $\vec{f}$ be the freq vector for English.

1. Input($T$) a text
2. Compute $\vec{g}$, the freq vector for $T$
3. Compute $\vec{g} \cdot \vec{f}$. If $\approx 0.065$ then output YES, else NO
Given $T$ a long text that you KNOW was coded by shift.

For $s = 0$ to 25

▶ Create $T_s$ which is $T$ shifted by $s$.
▶ If $\text{Is English}(T) = \text{YES}$ then output $T_s$ and stop. Else try next value of $s$.

Key: No Near Misses. There will not be two values of $s$ that are both close to 0.065.
A Note on Cracking Shift Cipher

In the last slide we tried all shifts in order. Can do better:

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$
- Sort vector. So most common letter is $\sigma_1$, next is $\sigma_2$, etc.
- For $i = 0$ to $25$
  - Create $T_s$ which is $T$ shifted as if $\sigma_i$ maps to $e$.
  - Compute $\vec{g}$, the freq vector for $T_s$
  - Compute $\vec{g} \cdot \vec{f}$. If $\approx 0.065$ then stop: $T_s$ is your text. Else try next value of $s$.

**Key:** Quite likely to succeed in the first try, or at least very early.
Private-key encryption

\[ c := \text{Enc}_k(m) \]

encryption

\[ m := \text{Dec}_k(c) \]

decryption

message/plaintext
Private-key encryption

$k \quad m \quad c := \text{Enc}_k(m) \quad c \quad k \quad m := \text{Dec}_k(c)$
A private-key encryption scheme is defined by a message space \( \mathcal{M} \) and algorithms (Gen, Enc, Dec):

- **Gen** (key generation algorithm): outputs \( k \in K \) (For SHIFT this is \( k \in \{0, \ldots, 25\} \). Should 0 be included?)
- **Enc** (encryption algorithm): takes key \( k \) and message \( m \in \mathcal{M} \) as input; outputs ciphertext \( c \)

\[
    c \leftarrow Enc_k(m)
\]

(For SHIFT this is \( Enc(m_1, \ldots, m_n) = (m_1 + k, \ldots, m_n + k) \).)
- **Dec** (decryption algorithm): takes key \( k \) and ciphertext \( c \) as input; outputs \( m \) or “error”

\[
    m := Dec_k(c)
\]

(For SHIFT this is \( Dec(c_1, \ldots, c_n) = (c_1 - k, \ldots, c_n - k) \).)

\( \forall k \) output by Gen \( \forall m \in \mathcal{M}, Dec_k(Enc_k(m)) = m \)

(For SHIFT this is \( (m + k) - k = m \))
Kerchoffs’s principle

We made the comment We KNOW that SHIFT was used. More generally we use this principle.

- The encryption scheme is not secret
  - The attacker knows the encryption scheme
  - The only secret is the key
  - The key must be chosen at random; kept secret

- Some arguments in favor of this principle
  - Easier to keep key secret than algorithm
  - Easier to change key than to change algorithm
  - Standardization
    - Ease of deployment
    - Public validation