Lecture 2

Cryptography:
History + Simple Encryption Methods and Preliminaries

Cryptography can be used at different levels

- **algorithms**: encryption, signatures, hashing, RNG
- **protocols** (2 or more parties): key distribution, authentication, identification, login, payment, etc.
- **systems**: electronic cash, secure filesystems, smartcards, VPNs, e-voting, etc.
- **attacks**: on all the above
Some applications of cryptography

- network, operating system security
- protect Internet, phone, space communication
- electronic payments (e-commerce)
- database security
- software/content piracy protection
- pay TV (e.g., satellite)
- military communications
- voting

Open vs. closed design model

- **Open design**: algorithm, protocol, system design (and even possible plaintext) are public information. Only key(s) are kept secret.

- **Closed design**: as much information as possible is kept secret.
Core issue in network security: how to communicate securely?

Looks simple…

But, the devil is in the details

Note: even storage is a form of communication

The biggest “headache” is that...

Good security must be

Effective

Yet

Unobtrusive

Because security is not a service in and of itself, but a burden!
Cryptography has been around...

• Most CS sub-fields are fairly new:
  - Graphics, compilers, software, CSCW, OS, architecture

• And, a few are quite old:
  - Cryptography, database, networking

Some history: Caesar’s cipher

Homo
Hominem
Lupus!

Krpr
Krplqhp
Oxsxv!
Some history: Rosetta Stone

Some history: Enigma
Historical (Primitive) Ciphers

- **Shift (e.g., Caesar):** \( \text{Enc}_k(x) = x + k \mod 26 \)
- **Affine:** \( \text{Enc}_{k_1,k_2}(x) = k_1 \times x + k_2 \mod 26 \)
- **Substitution:** \( \text{Enc}_{\text{perm}}(x) = \text{perm}(x) \)
- **Vigenere':** \( \text{Enc}_k(x) = ( X[0]+K[0], X[1]+K[1], \ldots ) \)
- **Vernam:** one-time pad (OTP)

---

**Shift (Caesar) Cipher**

**Example:**

\( K = 11 \)

\[
\begin{align*}
W &\rightarrow 22 & E &\rightarrow 4 & W &\rightarrow 22 & I &\rightarrow 8 & L &\rightarrow 11 & E &\rightarrow 4 \\
M &\rightarrow 12 & E &\rightarrow 4 & T &\rightarrow 19 & A &\rightarrow 0 & T &\rightarrow 19 & M &\rightarrow 12 \\
I &\rightarrow 8 & D &\rightarrow 3 & N &\rightarrow 13 & I &\rightarrow 6 & G &\rightarrow 7 & H &\rightarrow 19 \\
T &\rightarrow 15 & R &\rightarrow 4 & S &\rightarrow 19 \\
\end{align*}
\]

- How many keys are there?
- How many trials are needed to find the key?
**Substitution Cipher**

Example:

```
 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

**KEY**

```
 W E W I L L M E E T A T M I D N I G H T
 K H K Z B B T H H M X M T Z A S Z O G M
```

- How many keys are there?
- How many trials are needed to find the key?

**Substitution Cipher**

**Cryptanalysis**

Probabilities of Occurrence
Substitution Cipher

Cryptanalysis

Frequency of some common digrams

VERNAM One-Time Pad: world’s best cipher

Plaintext = \{p_0, ..., p_{n-1}\}

One-time pad stream = \{otp_0, ..., otp_{n-1}\}

Ciphertext = \{c_0, ..., c_{n-1}\}

where:

\[ c_i = p_i \oplus otp_i \forall 0 < i < n \]

\[ C = A \oplus B \]

\[ C \oplus B = A \]
VERNAM One-Time Pad: world’s best cipher

- Vernam offers perfect information-theoretic security, but:

- How long does the OTP keystream needs to be?

- How do Alice and Bob exchange the keystream?

Encryption Principles

- A cryptosystem has (at least) five ingredients:
  - Plaintext
  - Secret Key
  - Ciphertext
  - Encryption algorithm
  - Decryption algorithm

- Security usually depends on the secrecy of the key, not the secrecy of the algorithm
Crypto Basics

Crypto Attacks:
- ciphertext only
- known plaintext
- chosen plaintext
- chosen ciphertext

Cryptosystem:
- $P$ -- plaintext
- $C$ -- ciphertext
- $K$ -- keyspace
- $E$ -- encryption rules
- $D$ -- decryption rules

Average time required for exhaustive key search (for brute force attacks)

<table>
<thead>
<tr>
<th>Key Size (bits)</th>
<th>Number of Alternative Keys</th>
<th>Time required at $10^6$ Decr/$\mu$s</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>$2^{32} = 4.3 \times 10^9$</td>
<td>2.15 milliseconds</td>
</tr>
<tr>
<td>56</td>
<td>$2^{56} = 7.2 \times 10^{16}$</td>
<td>10 hours</td>
</tr>
<tr>
<td>128</td>
<td>$2^{128} = 3.4 \times 10^{38}$</td>
<td>$5.4 \times 10^{18}$ years</td>
</tr>
<tr>
<td>168</td>
<td>$2^{168} = 3.7 \times 10^{50}$</td>
<td>$5.9 \times 10^{30}$ years</td>
</tr>
</tbody>
</table>
Types of Attainable Security

• **Perfect, unconditional or “information theoretic”:** the security is evident free of any assumptions

• **Reducible or “provable”:** security can be shown to be based on some common (often unproven) assumptions, e.g., the conjectured difficulty of factoring large integers

• **Ad hoc:** the security seems good → often “snake oil”...

Take a look at:

http://www.ciphersbyritter.com/GLOSSARY.HTM

Computational Security

• Encryption scheme is *computationally secure* if
  – cost of breaking it (via brute force) exceeds the value of the encrypted information; or
  – time required to break it exceeds useful lifetime of the encrypted information

• Most modern schemes we will see are **considered** computationally secure
  – Usually rely on very large key-space, impregnable to brute force

• Most advanced schemes rely on lack of knowledge of effective algorithms for certain hard problems, not on a proven inexistence of such algorithms → reducible security!
  – Such as: factorization, discrete logarithms, etc.
## Conventional Encryption Principles

<table>
<thead>
<tr>
<th>Plaintext input</th>
<th>Encryption algorithm (e.g., DES)</th>
<th>Transmitted ciphertext</th>
<th>Decryption algorithm (reverse of encryption algorithm)</th>
<th>Plaintext output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secret key shared by sender and recipient</td>
<td>Secret key shared by sender and recipient</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Cryptosystems

Classified along three dimensions:

- **Type of operations used for transforming plaintext into ciphertext**
  - Binary arithmetic: shifts, XORs, ANDs, etc.
    - Typical for **conventional** encryption
  - Integer arithmetic
    - Typical for **public key** encryption

- **Number of keys used**
  - Symmetric or conventional (single key used)
  - Asymmetric or public-key (2 keys: 1 to encrypt, 1 to decrypt)

- **How plaintext is processed:**
  - One bit at a time
  - A string of any length
  - A block of bits
Complexity reminder/re-cap

- \( P \): problems that can be solved in polynomial time, i.e., problems that can be solved/decided "efficiently"
- \( \overline{NP} \): broad set of problems that includes \( P \):
  - answers can be verified "efficiently";
  - solutions can't always be efficiently found.
- \( NP\text{-complete} \): the believed-to-be-hard decision problems in \( NP \), they appear to have no efficient solution; answers are efficiently verifiable, solution to one is \textit{never much harder than a solution to another}
- \( NP\text{-hard} \): hardest; cannot be solved by a non-deterministic TM. Many computation version of \( NP\text{-complete} \) problems are \( NP\text{-hard} \).

Examples:
- Factoring, discrete log are in \( NP \), not know if in \( NP\text{-complete} \) or in \( P \)
- Primality testing was recently shown to be in \( P \)
- Knapsack is in \( NP\text{-complete} \)

For more info, see: http://www.nist.gov/dads/HTML

Suggested readings:

Chapters 1 and 2 in KPS book
Optional: Ch 1 in Stinson

Don't forget to check the website! Did you do it before this lecture?