Overarching goal of cryptography: securing communication over untrusted networks

Alice → Bob

third party should not be able to
1) eavesdrop of communication (confidentiality)
2) tamper with the communication (integrity)

Today: secure communication on web (https://...)
   TLS protocol (transport layer security)
   two components: handshake (key exchange)
   record layer (confidentiality + integrity)

Protecting data at rest: disk encryption

Most of this course: study mechanics for protecting confidentiality + data
- Encryption schemes for confidentiality
- Signature schemes for message integrity
- Key exchange for setting up shared secrets

End of this course: protecting communication ⇒ protecting computation
- Two users want to learn a joint function of their private inputs
  - training models on private (hidden) data
  - comparing two DNA sequences privately
  - private auction to determine winner without revealing bids
  - private voting mechanisms (can identify winner of election without revealing individual votes)
- We can show the following remarkable theorem:
  “Anything that can be computed with a trusted party can be computed without!”

Logistics and administration:
- Course website: https://www.cs.utexas.edu/~dwu4/courses/fa22
- See Ed Discussion for announcements; notes will be posted to course website (1-2 days after lecture)
- Homework submission via Gradescope (enroll via Course)
- Course consists of 5 homework assignments (worth 75%) and one take-home final (worth 25%)
- Five late days for the semester: use in 24-hour increments, max 72 hours (3 late days) for any single assignment
- Some office hours will also be available on Zoom

This semester: Lectures will be recorded using Lectures Online.
Please participate virtually if you are feeling unwell.
See protect.utexas.edu for suggested guidelines, vaccine information, etc.
A brief history of cryptography:

Original goal was to protect communication (in times of war)
Basic idea: Alice and Bob have a shared key k

Alice computes \( c \leftarrow \text{Encrypt} (k, m) \)

Bob computes \( m \leftarrow \text{Decrypt} (k, c) \) to recover the message

This tuple \((\text{Encrypt}, \text{Decrypt})\) is called a cipher

\( K, M, C \) are sets (e.g., \( K = M = C = \{0,1,\ldots,|K|\} \))

Definition. A cipher is defined over \((K, M, C)\) where \( K \) is a key-space, \( M \) is a message space and \( C \) is a ciphertext space, and consists of two algorithms \((\text{Encrypt}, \text{Decrypt})\):

\[
\begin{align*}
\text{Encrypt} : & K \times M \to C \quad \text{functions should be "efficiently computable"} \\
\text{Decrypt} : & K \times C \to M \quad \text{theory: runs in probabilistic polynomial time} \\
& \text{practice: fast on an actual computer (e.g., < 10ms on my laptop)}
\end{align*}
\]

Correctness: \( \forall k \in K, \forall m \in M: \)

\[
\text{Decrypt} (k, \text{Encrypt} (k, m)) = m
\]

"decrypting a ciphertext recovers the original message"

Early ciphers:

- Caesar cipher: "shift by 3"

\[
\begin{array}{c|c}
A & D \\
B & E \\
C & F \\
\vdots & \vdots \\
X & A \\
Y & B \\
Z & C \\
\end{array}
\]

Not a cipher! There is no key!
Anyone can decrypt!
Algorithm to encrypt is assumed to be public.
NEVER RELY ON SECURITY BY OBSCURITY!

- Caesar cipher ++: "shift by k" \((k = 13 : \text{ROT-13})\)

\( k \) is the key
Still totally broken since there are only 26 possible keys (simply via brute force guessing)

- Substitution cipher: the key defines a permutation of the alphabet (i.e., substitution)

\[
\begin{array}{c|c}
A & C \\
B & X \\
C & S \\
\vdots & \vdots \\
Z & T \\
\end{array}
\]

ABC \( \to \) CXJ

Substitution table is the key

How many keys? For English alphabet, \( 26! \approx 2^{88} \) possible keys

Very large value, cannot brute force the key
Still broken by frequency analysis
- e is the most frequent character (~12%)
- q is the least frequent character (~0.10%)

Can also look at digram, trigram frequencies

- Vigenère cipher (late 1500s) - "polyalphabetic substitution"
  - key is short phrase (used to determine substitution table):
    - \( m = \text{HELLO} \)
    - \( k = \text{CAT} \)
  - Encrypt \((k, m)\):
    \[
    \begin{align*}
    \text{HELLO} + \text{CATCA} \quad &\quad \text{repeat the key} \\
    &\quad \text{KFFPP} \quad \text{interpolate letters as number between 1 and 26}
    \end{align*}
    \]
    - addition is modulo 26

  - if we know the key length, can break using frequency analysis
  - otherwise, can try all possible key lengths \( l = 1, 2, \ldots \)

  \( \Rightarrow \) general assumption: keys will be much shorter than the message (otherwise if we have a good mechanism to deliver long keys securely, then can use that mechanism to share messages directly

- Fancier substitution ciphers: Enigma (based on rotor machines)
  - but: still breakable by frequency analysis

Today: encryption done using computers, lots of different ciphers
- AES (advanced encryption standard; 2000) - "block cipher"
- Salsa (2005) / ChaCha (2008) - "stream cipher"
Perfect secrecy is defined as follows: A perfect cipher (Encrypt, Decrypt) satisfies perfect secrecy if for all messages \( m_0, m \in M \), and all ciphertexts \( c \in C \):
\[
\Pr[k \in \mathbb{K} : \text{Encrypt}(k, m_0) = c] = \Pr[k \in \mathbb{K} : \text{Encrypt}(k, m) = c]
\]

where the probability is taken over the random choice of the key \( k \).

Perfect secrecy says that given a ciphertext, any two messages are equally likely.

\[ \Rightarrow \text{Cannot infer anything about underlying message given only the ciphertext (i.e., ciphertext-only attack)} \]

**Theorem.** The one-time pad satisfies perfect secrecy.

**Proof.** Take any message \( m \in \{0,1\}^n \) and ciphertext \( c \in \{0,1\}^n \). Then,
\[
\Pr[k \in \{0,1\}^n : \text{Encrypt}(k, m) = c] = \Pr[k \in \{0,1\}^n : k \oplus m = c] = \Pr[k \in \{0,1\}^n : k = m \oplus c] = \frac{1}{2^n}
\]

This holds for all messages \( m \) and ciphertexts \( c \), so one-time pad satisfies perfect secrecy.
Are we done? We now have a perfectly-secure cipher!

No! Keys are very long! In fact, as long as the message... if we can share keys of this length, can use same mechanism to share the message itself.

Malleable

Issues with the one-time pad:

- One-time: Very important. Never reuse the one-time pad to encrypt two messages. Completely broken!

Suppose $C_1 = k \oplus m_1$ and $C_2 = k \oplus m_2$

Then, $C_1 \oplus C_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2$ ← learn the xor of two messages!

One-time pad reuse:

- Project Verona (US. counter-intelligence operation against U.S. S.R. during Cold War)
  - Soviets reused some pages in codebook ~ led to decryption of ~3000 messages sent by Soviet intelligence over 37-year period [notably exposed espionage by Julius and Ethel Rosenberg]

- Microsoft Point-to-Point Tunneling (MS-PPTP) in Windows 98/NT (used for VPN)
  - Same key (in stream cipher) used for both server $\rightarrow$ client communication AND for client $\rightarrow$ server communication

- 802.11 WEP: both client and server use same key to encrypt traffic
  - Many problems just beyond one-time pad reuse (can even recover key after observing small number of frames!)

- Malleable: one-time pad provides no integrity; anyone can modify the ciphertext:

  $m \leftarrow k \oplus c$

  $\Rightarrow k \oplus (c \oplus m') = m \oplus m'$ ← adversary’s change now stored into original message