Today: Encryption

One-time security

One-time pad

Many-time security.

The assumption you should make:

Anyone can see the packets you are sending, everything is completely public!

Examples: HTTP, TCP/IP, Email,...

TCP dump: Dumps all the traffic sent on this WiFi.

Examples where encryption is used: HTTPS, messaging systems

Encryption scheme: Syntax

An encryption scheme consists of a key space $\mathcal{K}$, a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and two algorithms:

$\text{Enc}: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$

$\text{Dec}: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

Correctness: For every $m$ in $\mathcal{M}$, and every $k$ in $\mathcal{K}$,

$\text{Dec}(k, \text{Enc}(k,m)) = m$
Security: For every \( m, m' \) in \( M \),

\[
\text{Enc}(k,m) \equiv \text{Enc}(k,m')
\]

where \( k \) is uniformly distributed in \( K \).

Construction: One-Time Pad

Invented and patented by Gilbert Vernam 1917.

Analyzed and was proved secure by Shannon in 1945, but remained classified until 1949.

\[
M = K = C = \{0, 1\}
\]

\[
\text{Enc}(k,m) = k \oplus m
\]

\[
\text{Dec}(k,c) = k \oplus c
\]

Correctness:

\[
\text{Dec}(k, \text{Enc}(k,m)) = \text{Dec}(k, k \oplus m) = k \oplus (k \oplus m) = m
\]
Security: Fix any \( m \) in

If \( k \) is a random in \( \{0,1\}^n \) then

\[
\text{Enc}(k,m) = k \oplus m \text{ is random in } \{0,1\}^n:
\]

\[
\forall c \in \{0,1\}^n 
\]

\[
\Pr[\text{Enc}(k,m) = c] = \Pr[k \oplus m = c] = \Pr[k = c \oplus m] = 2^{-n} \checkmark
\]

One-time pad seems great, offers perfect security!

So, why not use one-time pad??

One-time pad only offers one-time security!

Note: Even though our definition of security seems to be so strong, it is not strong enough!

For example: Encryption of 0 reveals the secret key and then the key can no longer be used!
This seems like a contrived example, but is not as contrived as it seems. Often the beginning of the messages is known (say contains only meta-data). But then another message may contain secret information in the beginning.

**New definition:**

For any messages \( m_1, m_2, \ldots, m_t \) in \( M \), and messages \( m'_1, m'_2, \ldots, m'_t \) in \( M \)

\[
\text{Enc}(k, m'_1), \text{Enc}(k, m'_2), \ldots, \text{Enc}(k, m'_t) = \text{Enc}(k, m'_1), \text{Enc}(k, m'_2), \ldots, \text{Enc}(k, m'_t)
\]

**Impossible!**

Intuitively, \( \text{Enc}(k, m_1), \ldots, \text{Enc}(k, m_t) \) gives too much information about \( k \).

**Note:** A many-time secure scheme cannot be deterministic!

For any distinct \( m \) and \( m' \),

\((\text{Enc}(k, m), \text{Enc}(k, m))\) is distinguishable from \((\text{Enc}(k, m), \text{Enc}(k, m'))\)

**Conclusion:** A many-time secure encryption scheme must be randomized (or at least stateful)

**But the impossibility remains...**
Suppose we can generate as much randomness as we want from $k$ (like generating randomness "out of thin air").

Then we can use the one-time pad, while each time using newly generated randomness from $k$.

Seems like magic, right?

This is exactly what we will do! Generate randomness "out-of-thin air"
assuming hardness...

Namely, we will take a single key $k$, and use it to generate as many keys as we want: $F(k,1)$, $F(k,2)$, $\ldots$, $F(k,t)$ such that these keys are indistinguishable from random for a computationally bounded adversary!
Computationally bounded = polynomial time

Intuitively, computationally bounded means real world adversaries.

Definition: Indistinguishability against Chosen plaintext attacks

(Ind CPA, or CPA for short):

An encryption scheme \((Enc, Dec)\) is CPA secure if for any

\(m_1, m_2, \ldots, m_t\) in \(M\) and \(m'_1, m'_2, \ldots, m'_t\) in \(M\)

\[\left(Enc(k, m_1), \ldots, Enc(k, m_t)\right) \approx \left(Enc(k, m'_1), \ldots, Enc(k, m'_t)\right)\]

\[\uparrow\]

computational
indistinguishability

where \(k\) is random in \(\{0, 1\}^n\).

Intuitively, computationally indistinguishable means

indistinguishable in practice!