not ideal

One-time pad [Vigenere cipher where key is as long as the message!]

$\mathcal{K} = \{0,1\}^n$ Encrypt(k, m): output $c = k \oplus m$

$\mathcal{M} = \{0,1\}^n$ Decrypt(k, c): output $m = k \oplus c$

$\mathcal{C} = \{0,1\}^n$ bitwise exclusive OR operation (addition mod 2)

Correctness: Take any $k \in \{0,1\}^n$, $m \in \{0,1\}^n$:

$$\text{Decrypt}(k, \text{Encrypt}(k, m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = m \quad (\text{since } k \oplus k = 0^n)$$

Is this secure? How do we define security?

- Given a ciphertext, cannot recover the key?

  NOT GOOD! Says nothing about hiding message. Encrypt$(k, m) = m$ would be secure under this definition, but this scheme is totally insecure intuitively!

- Given a ciphertext, cannot recover the message.

  NOT GOOD! Can leak part of the message. Encrypt$(k, (m_0, m_1)) = (m_0, m_1 \oplus k)$. This encryption might be considered secure but leaks half the message. [Imagine if message was "username: alice || password: 123456"

  this might be the string that is leaked!

- Given a ciphertext, cannot recover any bit of the message.

  NOT GOOD! Can still learn parity of the bits (or every pair of bits), etc. Information still leaked...

- Given a ciphertext, learn nothing about the message.

  GOOD! But how to define this?

Coming up with good definitions is difficult! Definitions have to rule out all adversarial behavior (i.e., capture broad enough class of attacks)

Big part of crypto is getting the definitions right. Pre-1970s: cryptography has relied on intuition, but intuition is often wrong! Just because I cannot break it does not mean someone else cannot...

How do we capture "learning nothing about the message"?

If the key is random, then ciphertext should not give information about the message.

Definition. A cipher (Encrypt, Decrypt) satisfies perfect secrecy if for all messages $m_0, m_1 \in \mathcal{M}$, and all ciphertexts $c \in \mathcal{C}$:

$$\underbrace{\Pr[k \xleftarrow{R} \mathcal{K} : \text{Encrypt}(k, m_0) = c]}_{\text{probability that encryption of } m_0 \text{ is } c\text{, where the probability is taken over the random choice of the key } k} = \Pr[k \xleftarrow{R} \mathcal{K} : \text{Encrypt}(k, m_1) = c]$$

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

$\Rightarrow$ 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,

$$\begin{aligned}\Pr[k \xleftarrow{R} \{0,1\}^n : \text{Encrypt}(k,m) = c] &= \Pr[k \xleftarrow{R} \{0,1\}^n : k \oplus m = c] \\ &= \Pr[k \xleftarrow{R} \{0,1\}^n : k = m \oplus c] \\ &= \frac{1}{2^n}\end{aligned}$$

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]

"One-time" restriction

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! ← can leverage this to recover messages

  One-time pad reuse:

  - Project Verona (U.S. 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 → client communication AND for client → server communication
    ↳ (RC4)
  - 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:

  $m \leftarrow k \oplus c$

  ↑ replace c with $c \oplus m'$

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

Theorem (Shannon). If a cipher satisfies perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$.

Intuition: Every ciphertext can decrypt to at most $|\mathcal{K}| < |\mathcal{M}|$ messages. This means that ciphertext leaks information about the message (not all messages equally likely). Cannot be perfectly secret.

Proof. We will use a "counting" argument. Suppose $|\mathcal{K}| < |\mathcal{M}|$. Take any ciphertext $c \leftarrow \text{Encrypt}(k,m)$ for some $k \in \mathcal{K}$, $m \in \mathcal{M}$. This ciphertext can only decrypt to at most $|\mathcal{K}|$ possible messages (one for each choice of key). Since $|\mathcal{K}| < |\mathcal{M}|$, there is some message $m' \in \mathcal{M}$ such that

$$\forall k \in \mathcal{K} : \text{Decrypt}(k,c) \neq m'$$

By correctness of the cipher,

$$\forall k \in \mathcal{K} : \text{Encrypt}(k, m') \neq c$$

This means that

$$\left.\begin{aligned} &\Pr[k \xleftarrow{R} \mathcal{K} : \text{Encrypt}(k,m') = c] = 0 \\ &\Pr[k \xleftarrow{R} \mathcal{K} : \text{Encrypt}(k,m) = c] > 0 \end{aligned}\right\} \text{Cannot be perfectly secret!}$$

Take-away: Perfect secrecy requires long keys. Very impractical (except in the most critical scenarios – exchanging daily codebooks)

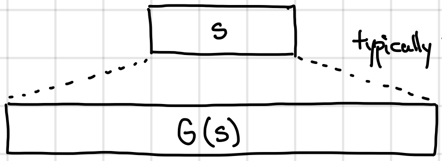
If we want something efficient / usable, we need to compromise somewhere.

- Observe: Perfect secrecy is an information-theoretic (i.e., a mathematical) property

  Even an infinitely-powerful (computationally-unbounded) adversary cannot break security

  We will relax this property and only require security against computationally-bounded (efficient) adversaries

<u>Idea</u>: "compress" the one-time pad: we will generate a long random-looking string from a <u>short</u> seed (e.g., $s \in \{0,1\}^{128}$).



typically: $s \in \{0,1\}^\lambda$ (λ is the seed length or <u>security parameter</u>)

$G(s) \in \{0,1\}^n$ where $n \gg \lambda$

n is the "stretch" of a PRG

<u>Stream cipher</u>: $\mathcal{K} = \{0,1\}^\lambda$

$\mathcal{M} = \mathcal{C} = \{0,1\}^n$

Encrypt(k, m): $c \leftarrow m \oplus \boxed{G(k)}$

Decrypt(k, c): $m \leftarrow c \oplus \boxed{G(k)}$

Instead of xor-ing with the key, we use the key to derive a "stream" of random-looking bits and use that in place of the one-time pad

If $\lambda < n$, then this scheme cannot be perfectly secure! So we need a <u>different</u> notion of security

<u>Intuitively</u>: Want a stream cipher to function "like" a one-time pad to any "reasonable" adversary.

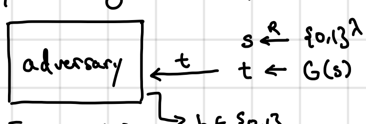
$\Rightarrow$ Equivalently: output of a PRG should "look" like uniformly-random string

What is a "reasonable" adversary?

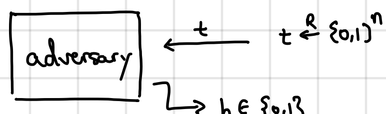
- Theoretical answer: algorithm runs in (probabilistic) polynomial time
- Practical answer: runs in time $< 2^{80}$ and space $< 2^{64}$ (can use larger numbers as well)

<u>Goal</u>: Construct a PRG so no efficient adversary can distinguish output from random.

Captured by defining two experiments or games:



Experiment 0



Experiment 1

the input to the adversary (t) is often called the <u>challenge</u>

Adversary's goal is to <u>distinguish</u> between Experiment 0 (pseudorandom string) and Experiment 1 (truly random string)

↳ It is given as input a string t of length n (either $t \leftarrow G(s)$ or $t \xleftarrow{R} \{0,1\}^n$)

↳ It outputs a guess (a single bit $b \in \{0,1\}$)

Remember: adversary knows the algorithm G; <u>only seed</u> is hidden!

Let $W_0 := \Pr[\text{adversary outputs } 1 \text{ in Experiment } 0]$

$W_1 := \Pr[\text{adversary outputs } 1 \text{ in Experiment } 1]$

define the distinguishing advantage of A as

$\text{PRGAdv}[A, G] := |W_0 - W_1|$

DO NOT RELY ON SECURITY BY OBSCURITY!

<u>Definition</u>. A PRG $G: \{0,1\}^\lambda \to \{0,1\}^n$ is <u>secure</u> if for all efficient adversaries A,

probabilistic polynomial time

$$\text{PRGAdv}[A, G] = \underline{\text{negl}}(\lambda)$$

↳ negligible function (in the input length)

smaller than any inverse polynomial

e.g., $\frac{1}{2^\lambda}$, $\lambda^{\log \lambda}$

- Theoretical definition: $f(\lambda)$ is negligible if $f \in o(\lambda^{-c})$ for all $c \in \mathbb{N}$
- Practical definition: quantity $\leq 2^{-80}$ or $\leq 2^{-128}$