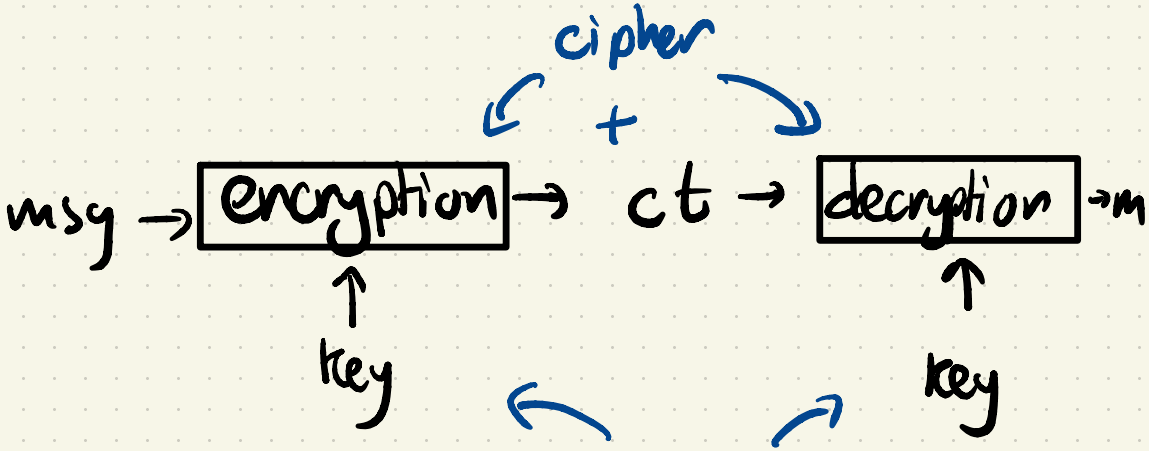
Day 2: Symmetric Encryption

Today

1. Cipher Syntax
2. One Time Pad
   a. Perfect Secrecy
   b. Limitations
3. Stream Ciphers
   → from Pseudo-Random Generators

1 What is a Cipher?



"symmetric": both algorithms use same key

Algorithms:

- Enc(msg, key) → ct
- Dec(ct, key) → msg

Domains:

msg ∈ M

ct ∈ C

key ∈ K

In this class, all will be byte sequences (or bitstrings)

2 First Cipher: One Time Pad (OTP)

First, the **XOR** function.

- defined over bits (0 or 1)
- outputs 1 if inputs differ
- input/output table:

x	y	XOR(x, y)
0	0	0
0	1	1
1	0	1
1	1	0

- written $x \oplus y$.
- operates on bit strings bit-by-bit
  - ex:

$$001 \oplus 101 = 100$$

- an "involution" (undoes itself):

$$x \oplus y \oplus y = x$$

OTP:

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

↑ length-n bit strings

key has same length as msg!

Algorithms:

- Enc(msg, key): ct ← msg ⊕ key
- Dec(ct, key): msg ← ct ⊕ key

Note: Dec(Enc(msg, key), key)

= Dec(msg ⊕ key, key)

= msg ⊕ key ⊕ key

= msg ✓

12a Perfect Secrecy.

OTP is <u>perfectly secret</u>:

a ciphertext c is equally likely to encode m_1 or m_2 (so long as k is random).

$$\Pr_K[Enc(m_1, k) = c] = \Pr_K[Enc(m_2, k) = c]$$

- Proved by Claude Shannon, father of "Information Theory"
  - → implies that c gives no info about m. Hence "perfect secrecy"

2b Limitations

- For OTP, len(k) = len(m).
- Hard to use - need big keys to be pre-shared

Russian OTP key book →
(Cold War)

- Unfortunately, big keys are unavailable

Thm (Shannon): correct &
If $\mathcal{E} = (Enc, Dec)$ is a perfectly secret cipher, then $|K| \geq |M|$

↗

can't have fewer keys than messages

→

can't have smaller keys than messages.

!! So are small keys impossible?

Idea: what if secrecy isn't perfect — but takes a long time to break?

3 Stream Ciphers from Pseudo-Random Generators (PRGs)

A PRG is just an algorithm
$G(seed \in \{0,1\}^\lambda) \to rand \in \{0,1\}^n$
with $\lambda < n$.

Ex: $G(0011) = 100011$

$(\lambda = 4,\ n = 6)$

It is secure if the output $G(s)$ "looks random" for random s.

Idea: use $G(s)$ as our key for OTP.

Stream Cipher:

$Enc(msg \in \{0,1\}^n, key \in \{0,1\}^\lambda)$

$ct \leftarrow G(key) \oplus msg$

$Dec(ct \in \{0,1\}^n, key \in \{0,1\}^\lambda)$

$msg \leftarrow G(key) \oplus msg.$

Now keys have length $\lambda < n$!

Moreover, we have secure PRGs with $\lambda = 256$ (32 bytes) for <u>any</u> # of bits n.

→ Ex: ChaCha20 (Daniel J Bernstein, U of Illinois)

For the problems:

cryptoy.prg(key: 32 bytes, n: integer)

returns n pseudo-random bytes.

Problems:

Required:
- bytes ↔ bit string
- OTP enc/dec
- stream cipher enc/dec

Bonus
- building an arbitrary-n PRG (from a PRG with n = 1+d)
- cryptanalysis of multi-time pad.

$$C_1 \oplus C_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$$
$$= m_1 \oplus m_2$$ ← leaky!