IERG 4130 – 2026 Fall

香港中文大學
CUHK

One-Time Pad

Yajin ZHOU

https://yajin.org

Credits: the slides are from CS161 of UC Berkeley,

Roadmap

	Symmetric-key	Asymmetric-key
Confidentiality	• **One-time pads** • Block ciphers with chaining modes (e.g. AES-CBC) • Stream ciphers	• RSA encryption • ElGamal encryption
Integrity, Authentication	• MACs (e.g. HMAC)	• Digital signatures (e.g. RSA signatures)

- Hash functions
- Pseudorandom number generators
- Public key exchange (e.g. Diffie-Hellman)
- Key management (certificates)
- Password management

Review: XOR

The XOR operator takes two bits and outputs one bit:

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Useful properties of XOR

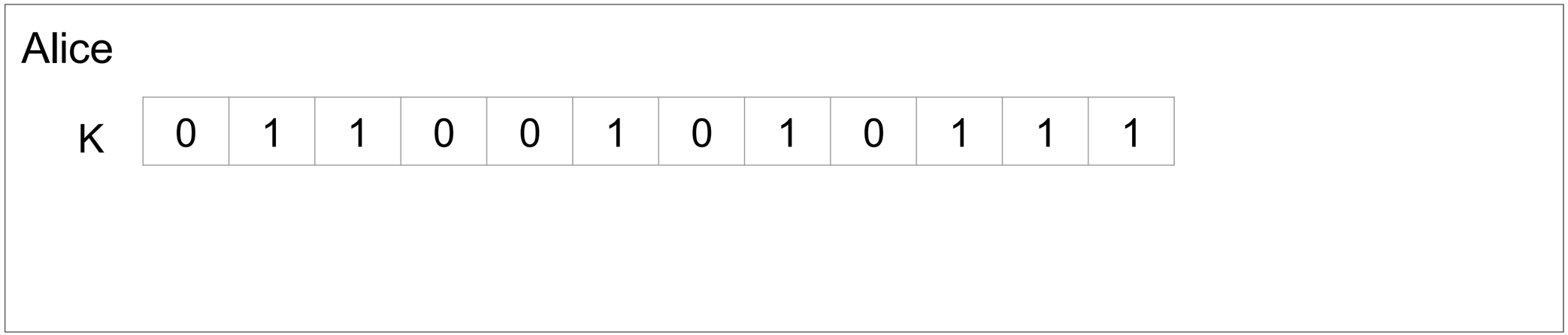
$x \oplus 0 = x$
$x \oplus x = 0$
$x \oplus y = y \oplus x$
$(x \oplus y) \oplus z = x \oplus (y \oplus z)$
$(x \oplus y) \oplus x = y$

Review: XOR Algebra

Algebra works on XOR too

$y \oplus 1 = 0$	Goal: Solve for y
$y \oplus 1 \oplus 1 = 0 \oplus 1$	XOR both sides by 1
$y = 1$	Simplify with identities

One-Time Pads: Key Generation



The key *K* is a randomly-chosen bitstring.

Recall: We are in the symmetric-key setting, so we'll assume Alice and Bob both know this key.

One-Time Pads: Encryption

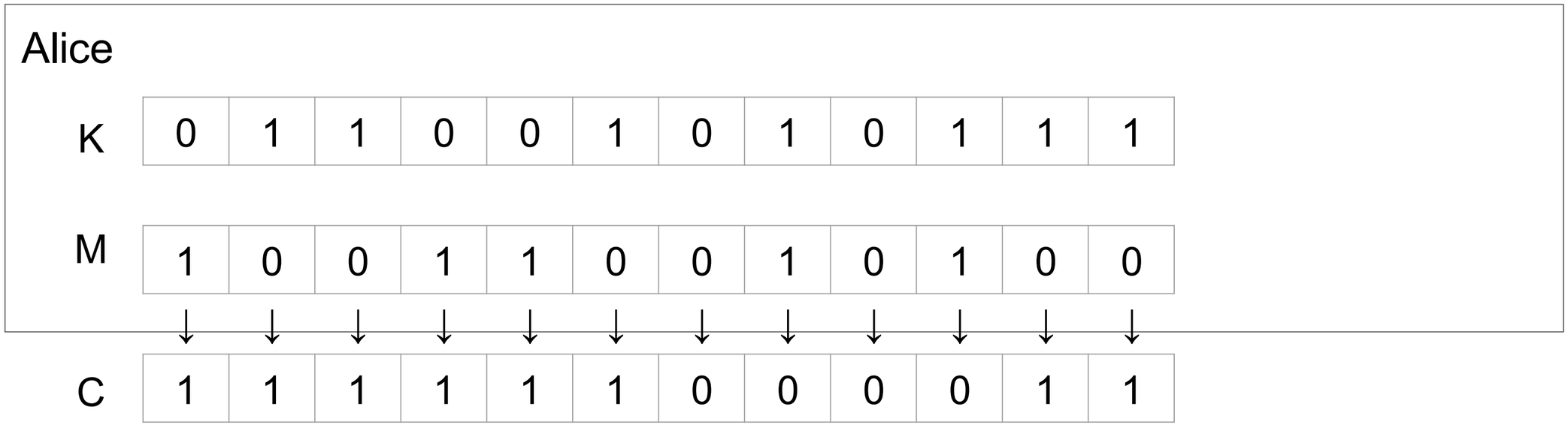
Alice

K	0	1	1	0	0	1	0	1	0	1	1	1
M	1	0	0	1	1	0	0	1	0	1	0	0

The plaintext *M* is the bitstring that Alice wants to encrypt.

Idea: Use XOR to scramble up *M* with the bits of *K*.

One-Time Pads: Encryption



Encryption algorithm: XOR each bit of *K* with the matching bit in *M*.

The ciphertext *C* is the encrypted bitstring that Alice sends to Bob over the insecure channel.

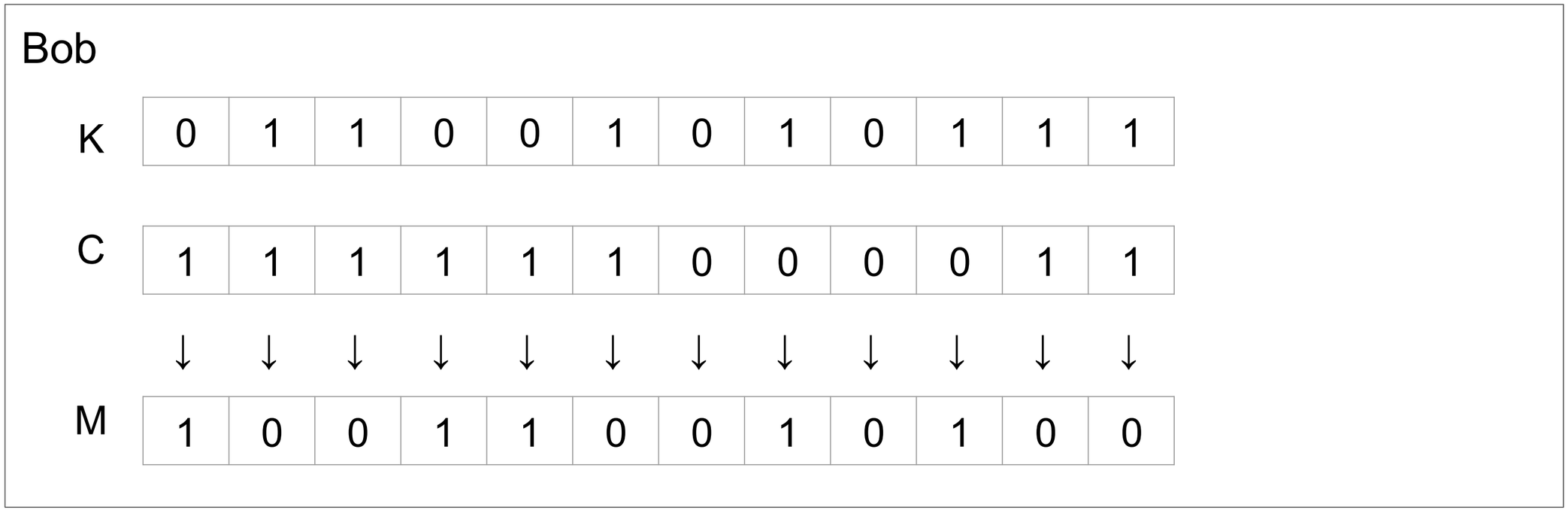
One-Time Pads: Decryption

Bob

K	0	1	1	0	0	1	0	1	0	1	1	1
C	1	1	1	1	1	1	0	0	0	0	1	1

Bob receives the ciphertext *C*. Bob knows the key *K*. How does Bob recover *M*?

One-Time Pads: Decryption



Decryption algorithm: XOR each bit of *K* with the matching bit in *C*.

One-Time Pad



- **KeyGen()**
  - Randomly generate an *n*-bit key, where *n* is the length of your message
    - Recall: For today, we assume that Alice and Bob can securely share this key
    - For one-time pads, we generate a *new* key for every message
- **Enc(*K*, *M*) = *K* ⊕ *M***
  - Bitwise XOR *M* and *K* to produce *C*
    - In other words: XOR the *i*th bit of the plaintext with the *i*th bit of the key.
    - $C_i = K_i \oplus M_i$
  - Alice and Bob use a different key for each encryption (this is the “one-time” in one-time pad).
- **Dec(*K*, *C*) = *K* ⊕ *C***
  - Bitwise XOR *C* and *K* to produce *M*
    - $M_i = K_i \oplus C_i$

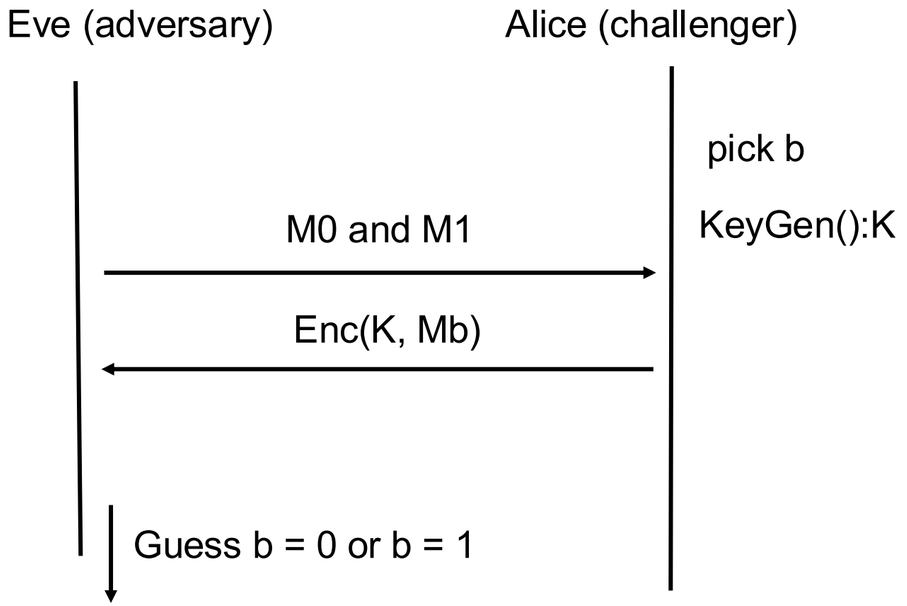
One-Time Pad: Correctness

- Correctness: If we encrypt and then decrypt, we should get the original message back

$$
\begin{aligned}
\mathrm{Enc}(K, M) &= K \oplus M && \text{Definition of encryption} \\
\mathrm{Dec}(K, \mathrm{Enc}(K, M)) &= \mathrm{Dec}(K, K \oplus M) && \text{Decrypting the ciphertext} \\
&= K \oplus (K \oplus M) && \text{Definition of decryption} \\
&= M && \text{XOR property}
\end{aligned}
$$

One-Time Pad: Security

- Recall our definition of confidentiality: The ciphertext should not give the attacker any additional information about the plaintext
- Recall our preliminary experiment (before IND-CPA) to test confidentiality from earlier:
  - If the probability that Eve correctly guesses **which message was sent is 1/2, then the encryption scheme is confidential**



One-Time Pad: Security

Possibility 0: Alice sends Enc(K, M_0)

The ciphertext is $C = K \oplus M_0$

Therefore, $K = C \oplus M_0$

Possibility 1: Alice sends Enc(K, M_1)

The ciphertext is $C = K \oplus M_1$

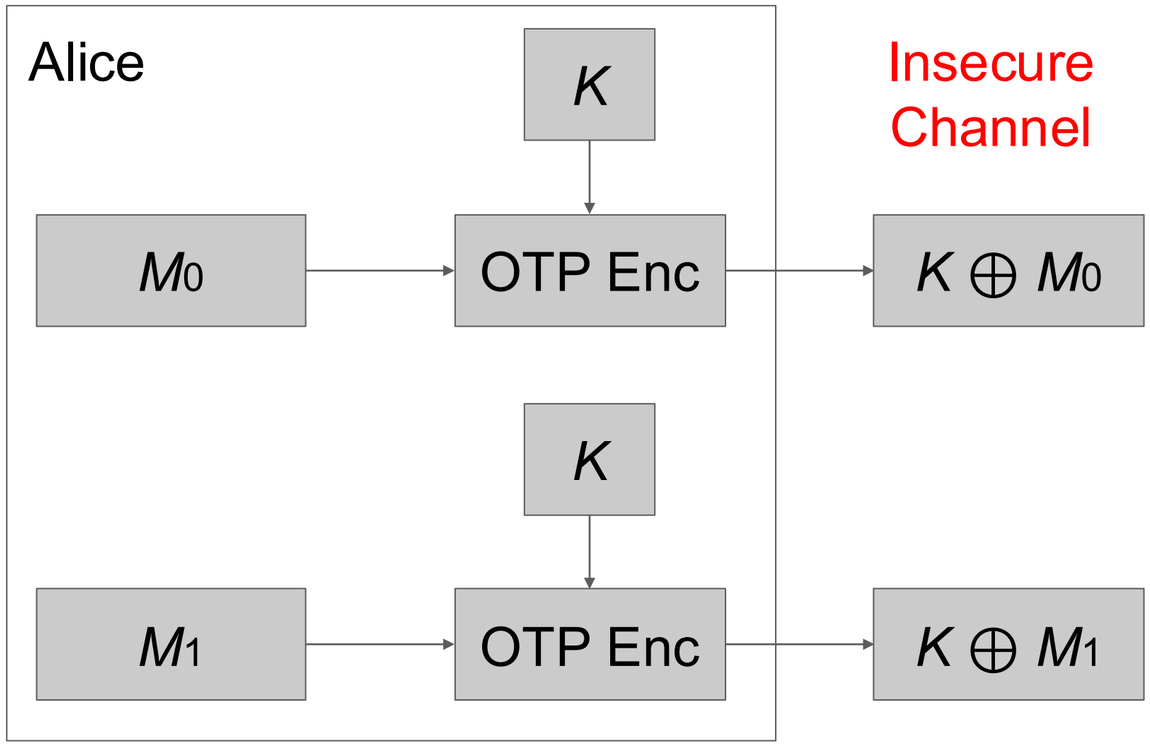
Therefore, $K = C \oplus M_1$

K was chosen randomly, so both possibilities are equally possible

Eve has learned no new information, so the scheme is *perfectly secure*

By “perfectly” we mean that any attacker has chance of winning the security game exactly ½ (not ½+epsilon)

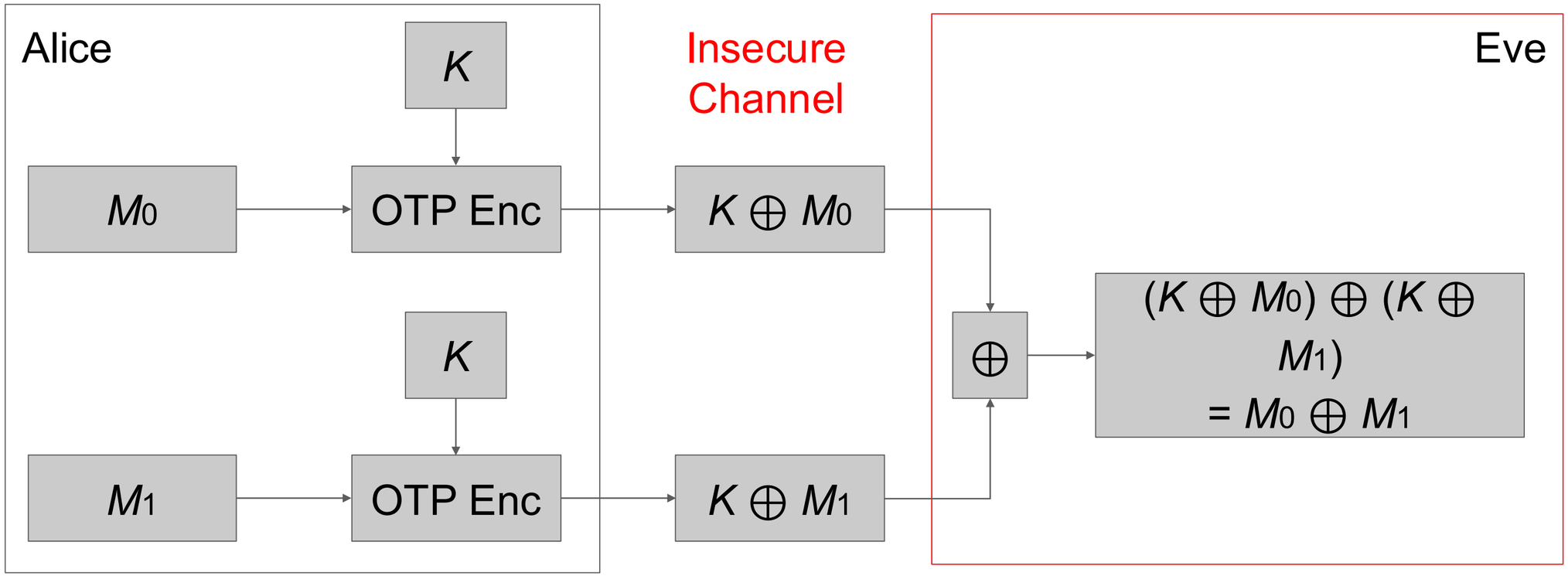
Two-Time Pads?



Eve sees two ciphertexts over the insecure channel.

What if we use the same key *K* to encrypt two different messages?

Two-Time Pads?



If Eve XORs the two ciphertexts, she learns $M_0 \oplus M_1$!

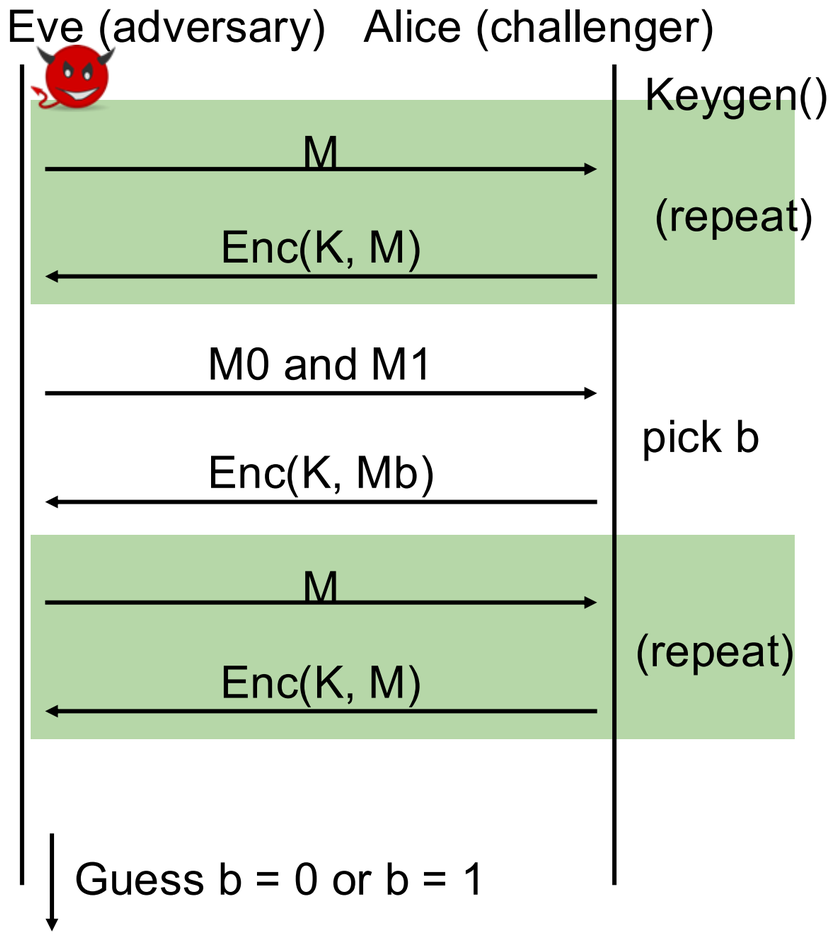
Two-Time Pads?



- **What if we use the same key *twice*?**
  - Alice encrypts M_0 and M_1 with the same key
  - Eve observes $K \oplus M_0$ and $K \oplus M_1$
  - Eve computes $(K \oplus M_0) \oplus (K \oplus M_1) = M_0 \oplus M_1$
    - Recall the XOR property: the K's cancel out
- **Eve has learned $M_0 \oplus M_1$. This is partial information about the messages!**
  - In other words, Eve knows which bits in M_0 match bits in M_1
  - If Eve knows M_0, she can deduce M_1 (and vice-versa)
  - Eve can also guess M_0 and check that M_1 matches her guess for M_0
- Result: One-time pads are not secure if the key is reused
  - Alice and Bob must use a different key for every message!

OTP does not offer IND-CPA

- **What is an attack strategy?**
- **Many possible, e.g.**
  - In the trial phase and the challenge phase, ask for “00” and “11”, it is deterministic.



Impracticality of One-Time Pads



- **Problem #1: Key generation**
  - Keys must be randomly generated for every message, and **never reused**
  - Randomness is expensive, as we'll see later
- **Problem #2: Key distribution**
  - To communicate an *n*-bit message, we need to securely communicate an *n*-bit key first
  - But if we have a way to securely communicate an *n*-bit key, we could have communicated the message directly!
- **Communicate keys in advance**
  - You have a secure channel now, but you won't have it later
  - Use the secure channel now to communicate keys in advance
  - Use one-time pad later to communicate over the insecure channel
  - And people can compute this by hand without computers!

Summary



- Symmetric encryption scheme: Alice and Bob share a secret key.
- Encryption and decryption: Bitwise XOR with the key.
- No information leakage if the key is never reused.
- Information leaks if the key is reused.
- Impractical for real-world usage