Introduction to Cryptography

m0leCon 2025 Workshops

What is cryptography?

- Where is cryptography?
- What is cryptography about?
- Kerchoff’s principle
- Classification of encryption / decryption algorithms
- An easy example of encryption: Caesar cipher

Where is cryptography?

Nowadays cryptography is found **anywhere**

- Internet communications (SSL, HTTPS...)
- Mobile networks (e.g. GSM)
- Messaging applications (e.g. Signal, WhatsApp)
- Legal documentations (digital signatures)
- Credit-card transactions over Internet
- Blockchains
- ... many more!

What is Cryptography about?

- Hiding data
  - Encryption: takes a secret key and the data to hide (plaintext) and returns a bunch of random looking bytes (ciphertext)
  - Decryption: takes the same secret key that was used to encrypt and the ciphertext, returns the original plaintext
- Authentication and integrity
  - Authentication: guarantees the “identity” of the sender (e.g. MACs, signatures)
  - Integrity: guarantees that the received message is the same as the sent message (e.g. hash functions)

A cryptographic system is usually made up of many fundamental cryptographic algorithms called primitives.

Kerchoff’s principle

“The cryptographic key should be the only secret: it would be foolish to rely on our enemies not to discover what algorithms we use because they most likely will. Instead, let’s be open about them.”

Classification of encryption / decryption algorithms

Symmetric cryptography

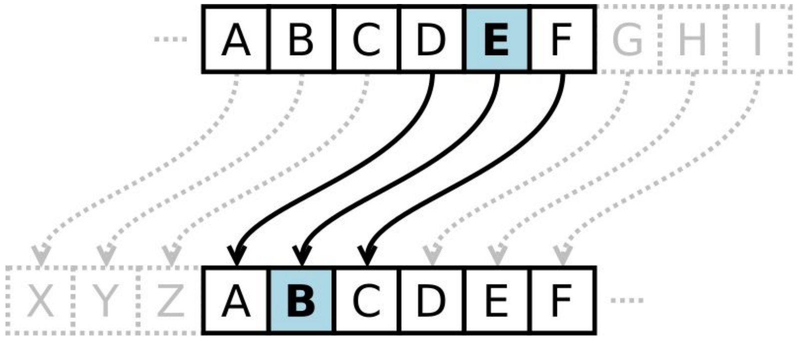
- The same key is used for encryption and decryption
- Basically making a lot of mess with bits
- Example: AES

Asymmetric cryptography

- Different keys are used for encryption and decryption
- Based on difficult mathematical problems
- Example: RSA

An easy example of encryption: Caesar cipher

- Encryption: shifting every letter in the message of x positions
- Decryption: shifting every letter in the message of x positions in the opposite direction
- Secret Key: the value of x



plaintext
"super secret message"

x = -3

ciphertext
"prmbo pbzobq jbppxdb"

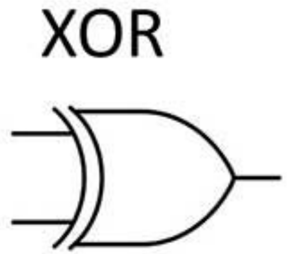
Challenge time

XOR

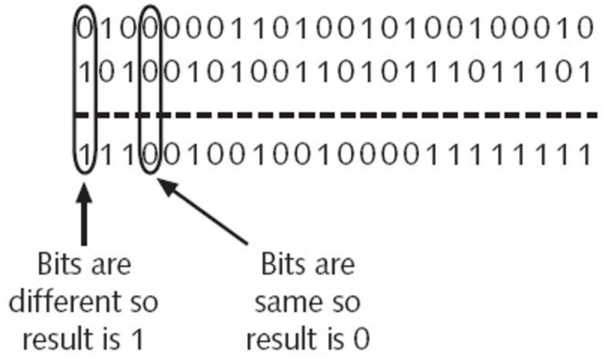
- Definition of the XOR operator
- The role of XOR in cryptography
- Why XOR?
- XOR Properties
- One time pad: how to encrypt with XOR

Definition of the XOR operator

- XOR takes two inputs and returns an output
- It is a bitwise operation, which means each bit of the two inputs is processed separately, producing one bit of output, then the different outputs are concatenated, producing the final output



A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0



The role of XOR in cryptography

Some examples of cryptographic primitives which rely on the XOR operation:

- Hash functions (sha2, sha3...)
- Symmetric key encryption / decryption
    - Block ciphers (AES-CBC...)
    - Stream ciphers (AES-CTR, ChaCha20...)

...and many more!

Why XOR?

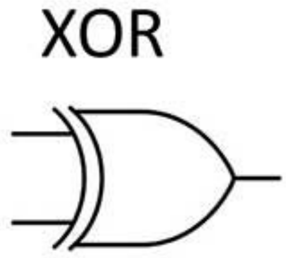
For a given plaintext bit (be it 0 or 1), the output is equally likely to be 0 or 1. So the ciphertext alone holds no information about the plaintext. This doesn't hold for other operators

Example: suppose we were using AND operator to encrypt a message, if a bit in the ciphertext is 1 we know for sure that the corresponding bit in the plaintext is 1 too.

Input		Output (Q)						
A	B	AND	OR	INH	XOR	NAND	NOR	XNOR
0	0	0	0	0	0	1	1	1
0	1	0	1	0	1	1	0	0
1	0	0	1	1	1	1	0	0
1	1	1	1	0	0	0	0	1

XOR properties

- $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- $a \oplus b = b \oplus a$
- $a \oplus a = 0$
- $a \oplus 0 = a$
- $a \oplus b \oplus a = b$
    - $a \oplus b \oplus a =$
    - $a \oplus a \oplus b =$
    - $0 \oplus b =$
    - b



A	B	Output
0	0	0
1	0	1
0	1	1
1	1	0

One Time Pad: how to encrypt with XOR

We have a plaintext p and a key k the same size of the plaintext, we compute the ciphertext c as:

$c = p \oplus k$

Since $a \oplus b \oplus a = b$, the decryption works as follows:

$c \oplus k = p \oplus k \oplus k = p$

Why do we need the key and the plaintext to be the same size?

Challenge time

Diffie-Hellman

- The problem of exchanging a shared secret key
- The Discrete Logarithm Problem
- DH Algorithm
- Attacks
    - Man in the Middle
    - Solving DLP

The Problem

- Alice and Bob want to exchange messages over an insecure channel, while preventing others from reading them

The Problem

- Alice and Bob want to exchange messages over an insecure channel, while preventing others from reading them

- They want to use a symmetric key cipher (AES), but they need a shared key

The Problem

- Alice and Bob want to exchange messages over an insecure channel, while preventing others from reading them

- They want to use a symmetric key cipher (AES), but they need a shared key

$\Rightarrow$ They have to find a way to share the secret key in a secure way

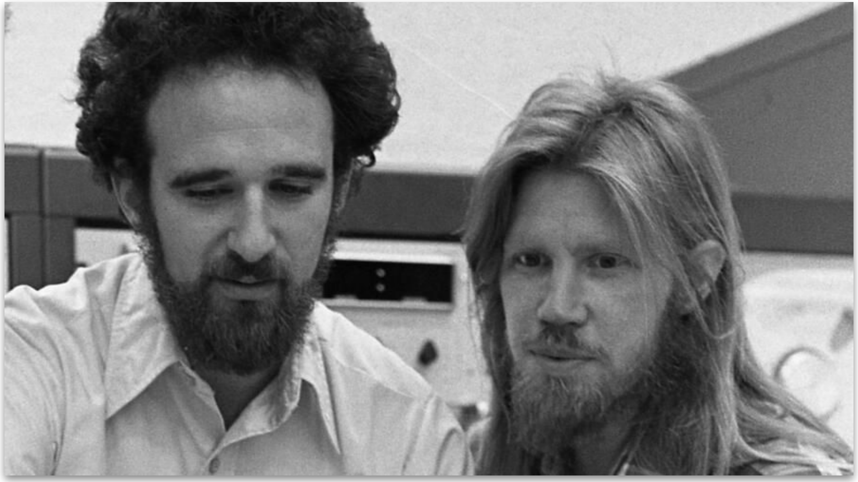
The Solution

There are 2 main solutions to this problem:

- Use physical means or meet each other
- Use the same insecure channel with some math tricks:
  - Diffie-Hellman
  - RSA

History

- The Diffie-Hellman key exchange is a cryptographic protocol that can securely generate a symmetric cryptographic key over a public channel
- It was published by Whitfield Diffie and Martin Hellman in 1976
- It was one of the first public key protocols



Applications

- TLS/SSL
- SSH
- IPSEC
- VPN
- Bluetooth
- WPA3
- IoT Pairing
- Smart TV

The Discrete Logarithm Problem

- Given three integers **g**, **c**, **p**, find an integer **x** that satisfies the following congruence:

$$\mathbf{g^x \equiv c \pmod{p}}$$

- If **g** and **p** are chosen properly, this problem is considered to be unsolvable with modern computational power

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**
- Then they both choose a secret number between 1 and p (we're going to refer to them as **a** and **b**)

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**
- Then they both choose a secret number between 1 and p (we’re going to refer to them as **a** and **b**)
- Alice sends Bob $\mathbf{A=g^a \pmod{p}}$

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**
- Then they both choose a secret number between 1 and p (we're going to refer to them as **a** and **b**)
- Alice sends Bob $\mathbf{A=g^a \pmod{p}}$
- At the same time Bob send Alice $\mathbf{B=g^b \pmod{p}}$

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**
- Then they both choose a secret number between 1 and p (we're going to refer to them as **a** and **b**)
- Alice sends Bob $\mathbf{A=g^a \pmod{p}}$
- At the same time Bob send Alice $\mathbf{B=g^b \pmod{p}}$
- Alice calculates $\mathbf{secret\_key_a = B^a \pmod{p} \equiv g^{ba} \pmod{p}}$
- Bob calculates $\mathbf{secret\_key_b = A^b \pmod{p} \equiv g^{ab} \pmod{p}}$

The Algorithm

- Alice and Bob agree on a prime number **p** and on a number **g** called generator modulo **p**
- Then they both choose a secret number between 1 and p (we're going to refer to them as **a** and **b**)
- Alice sends Bob $\mathbf{A=g^a \ (mod\ p)}$
- At the same time Bob send Alice $\mathbf{B=g^b \ (mod\ p)}$
- Alice calculates $\mathbf{secret\_key_a = B^a \ (mod\ p) \equiv g^{ba} \ (mod\ p)}$
- Bob calculates $\mathbf{secret\_key_b = A^b \ (mod\ p) \equiv g^{ab} \ (mod\ p)}$
- Alice and Bob obtained a shared key to use

An easy example

ALICE $\xrightarrow[\text{(Public)}]{p = 23,\ g = 5}$ BOB

$p = 23, g = 5$

ALICE

BOB

(Public)

$a = 4$ (private)

$b = 3$ (private)

ALICE $\xrightarrow[A = 5^4 \pmod{23} = 4]{p = 23,\ g = 5}$ BOB

$p = 23, g = 5$

ALICE → BOB

$A = 5^4 \pmod{23} = 4$

ALICE ← BOB

$B = 5^3 \pmod{23} = 10$

ALICE $\xrightarrow{p = 23,\ g = 5}$ BOB

$A = 5^4 \pmod{23} = 4$

ALICE $\longleftarrow$ BOB

$B = 5^3 \pmod{23} = 10$

ALICE $\longleftrightarrow$ BOB

$S_a = 10^4 \pmod{23} = 18$

$p = 23, g = 5$

ALICE → BOB

$A = 5^4 \pmod{23} = 4$

ALICE ← BOB

$B = 5^3 \pmod{23} = 10$

ALICE ↔ BOB

$S_a = 10^4 \pmod{23} = 18$

$S_b = 4^3 \pmod{23} = 18$

ALICE $\xrightarrow{p = 23,\ g = 5}$ BOB

$A = 5^4 \pmod{23} = 4$

ALICE $\xleftarrow{\hspace{2em}}$ BOB

$B = 5^3 \pmod{23} = 10$

Shared key

ALICE $\longleftrightarrow$ BOB

$S_a = 10^4 \pmod{23} = 18$

$S_b = 4^3 \pmod{23} = 18$

Challenge time

Attacks

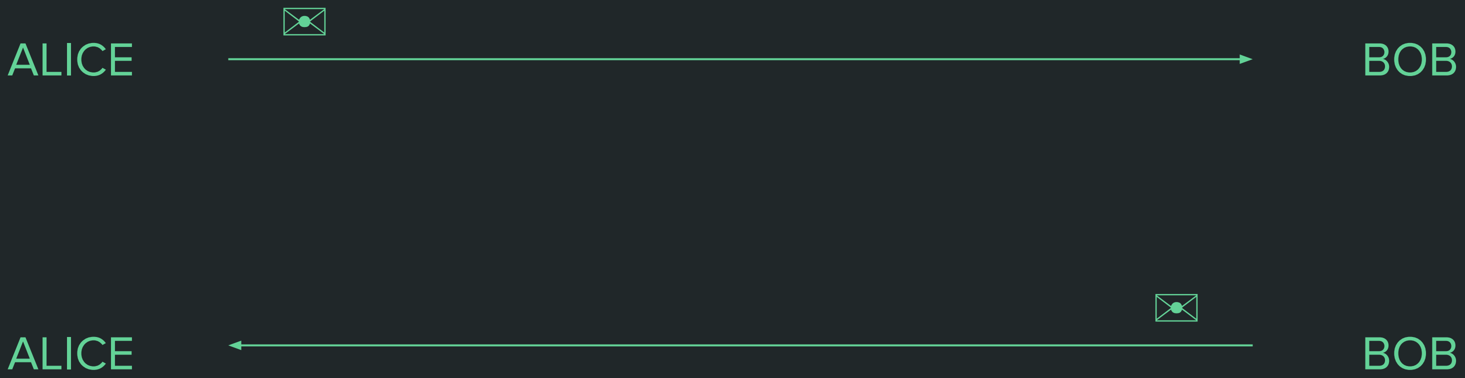
- Man in the Middle
- Solving DLP

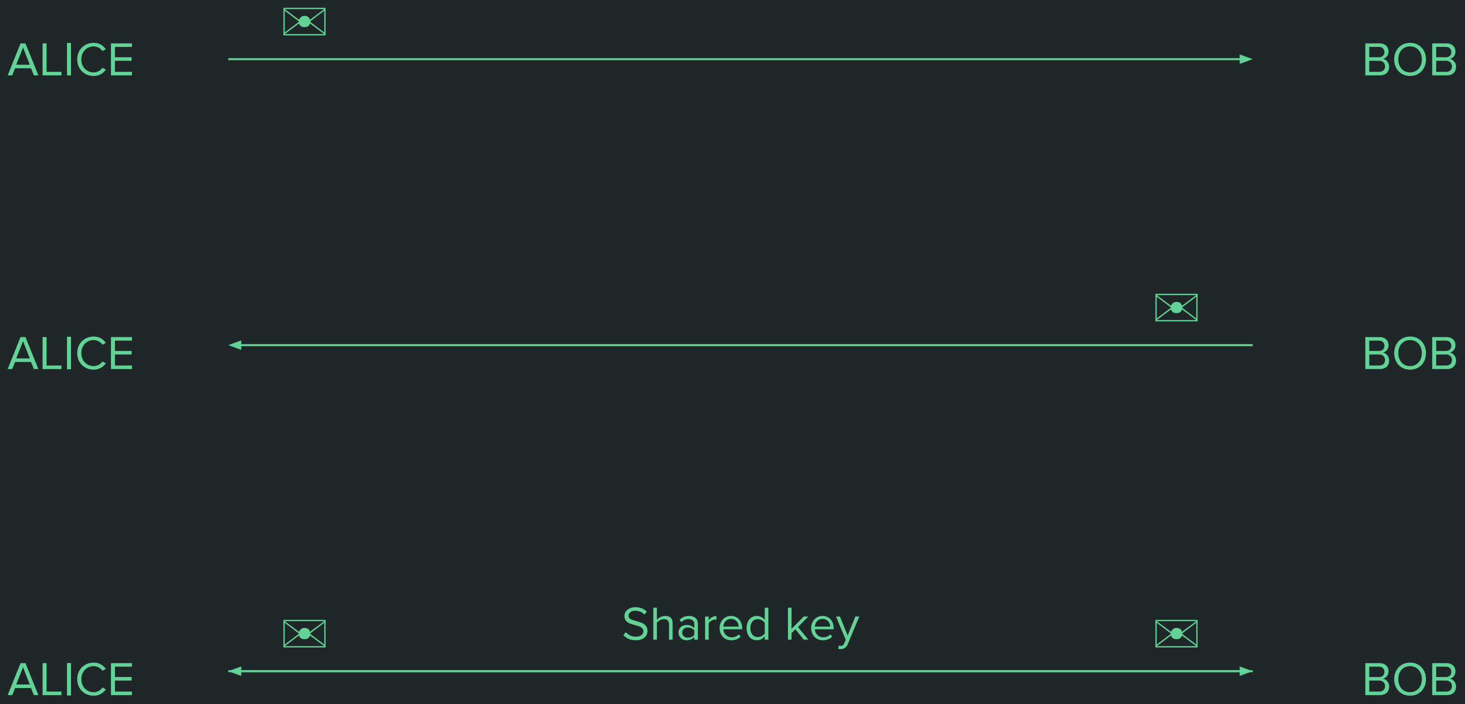
Man in the Middle

p, g

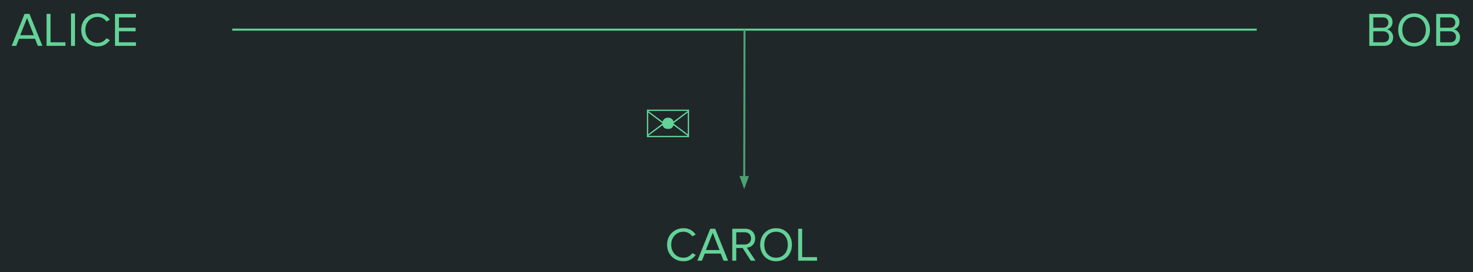
ALICE —————————————— BOB

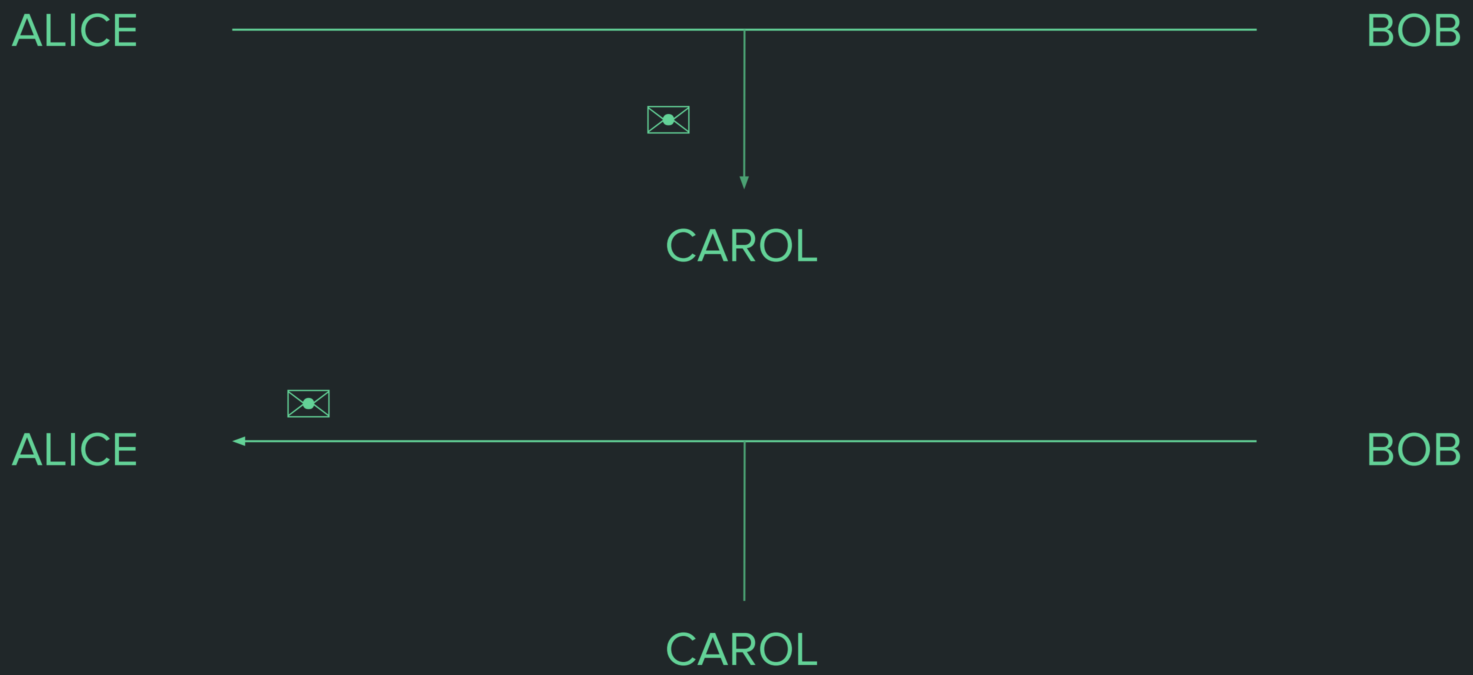

ALICE
BOB

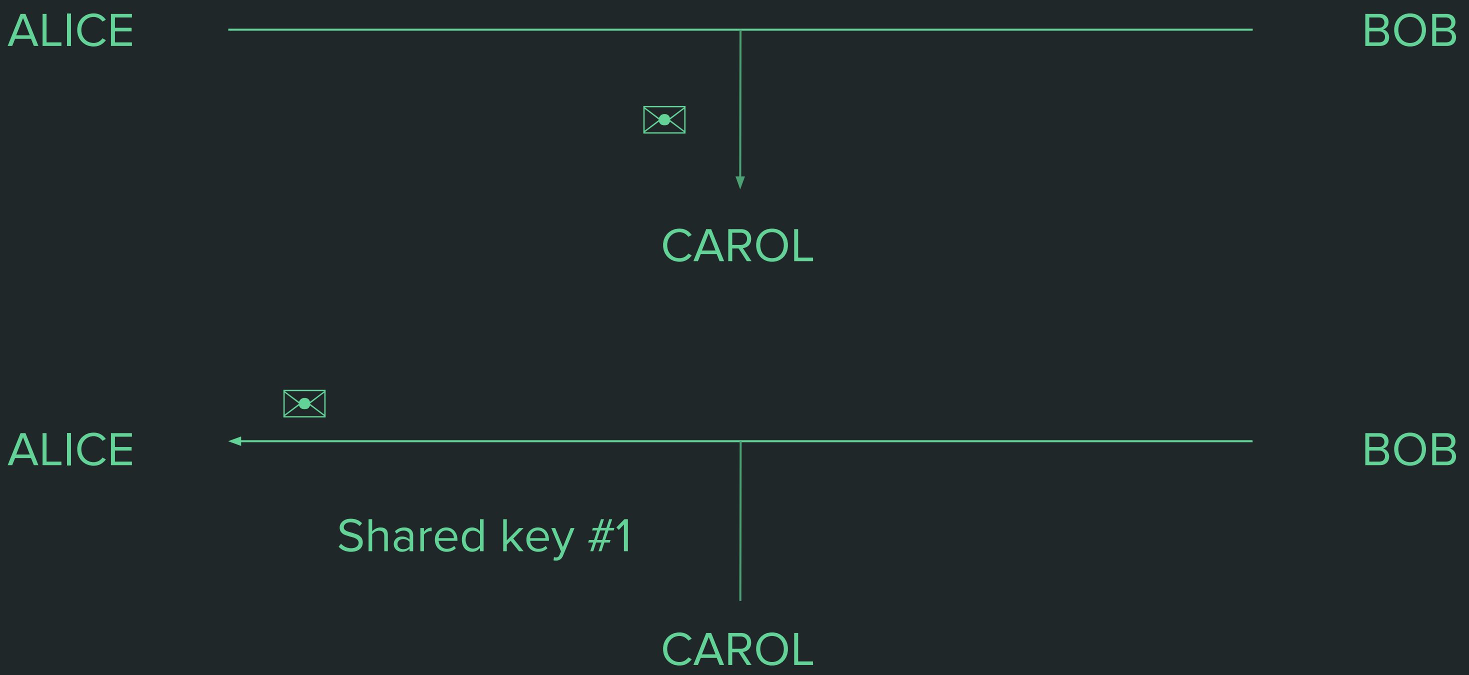

ALICE
BOB
ALICE
BOB

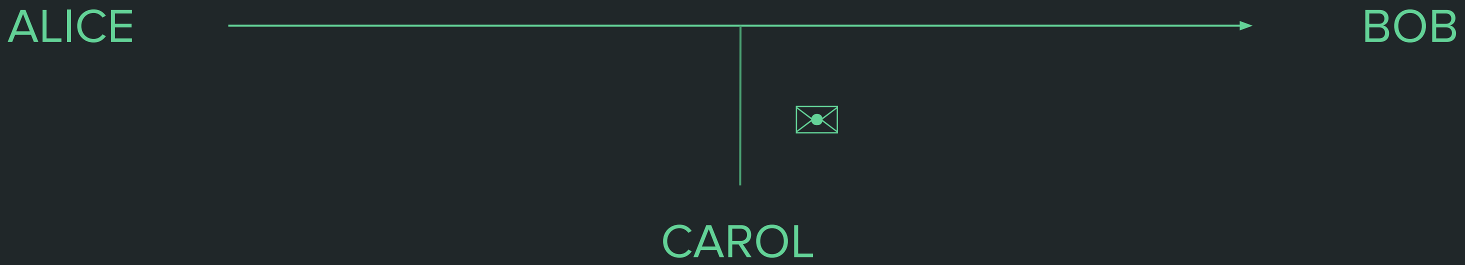

ALICE
BOB
ALICE
BOB
Shared key
ALICE
BOB

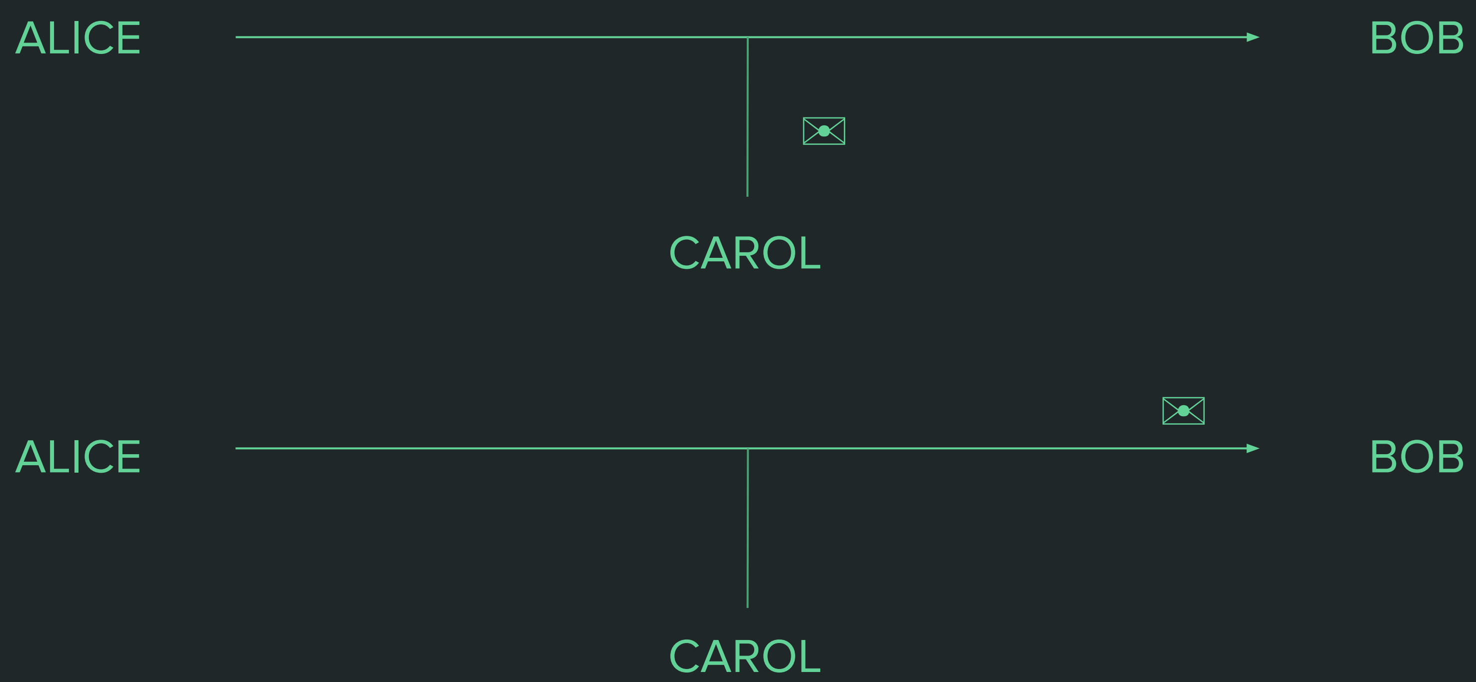

ALICE
BOB


ALICE
BOB
CAROL

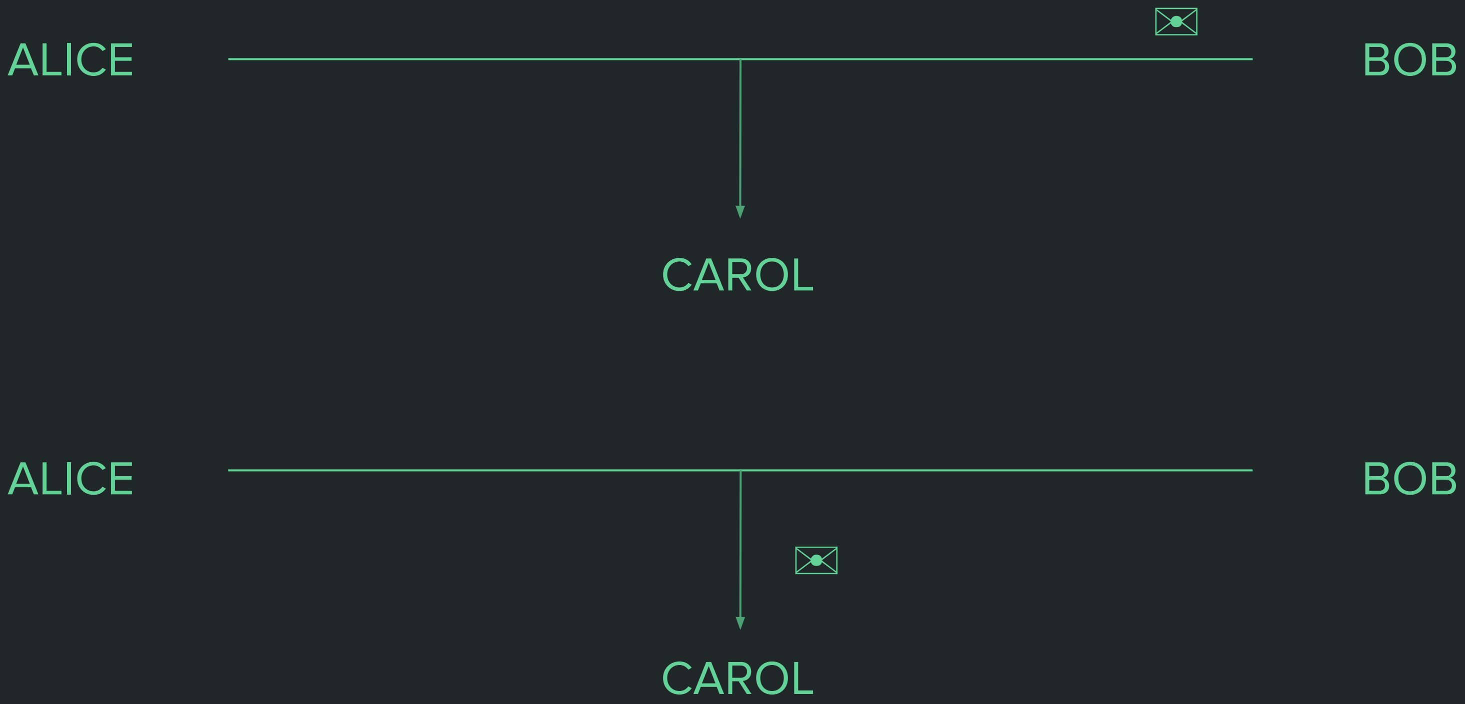

ALICE
BOB
CAROL
ALICE
BOB
CAROL

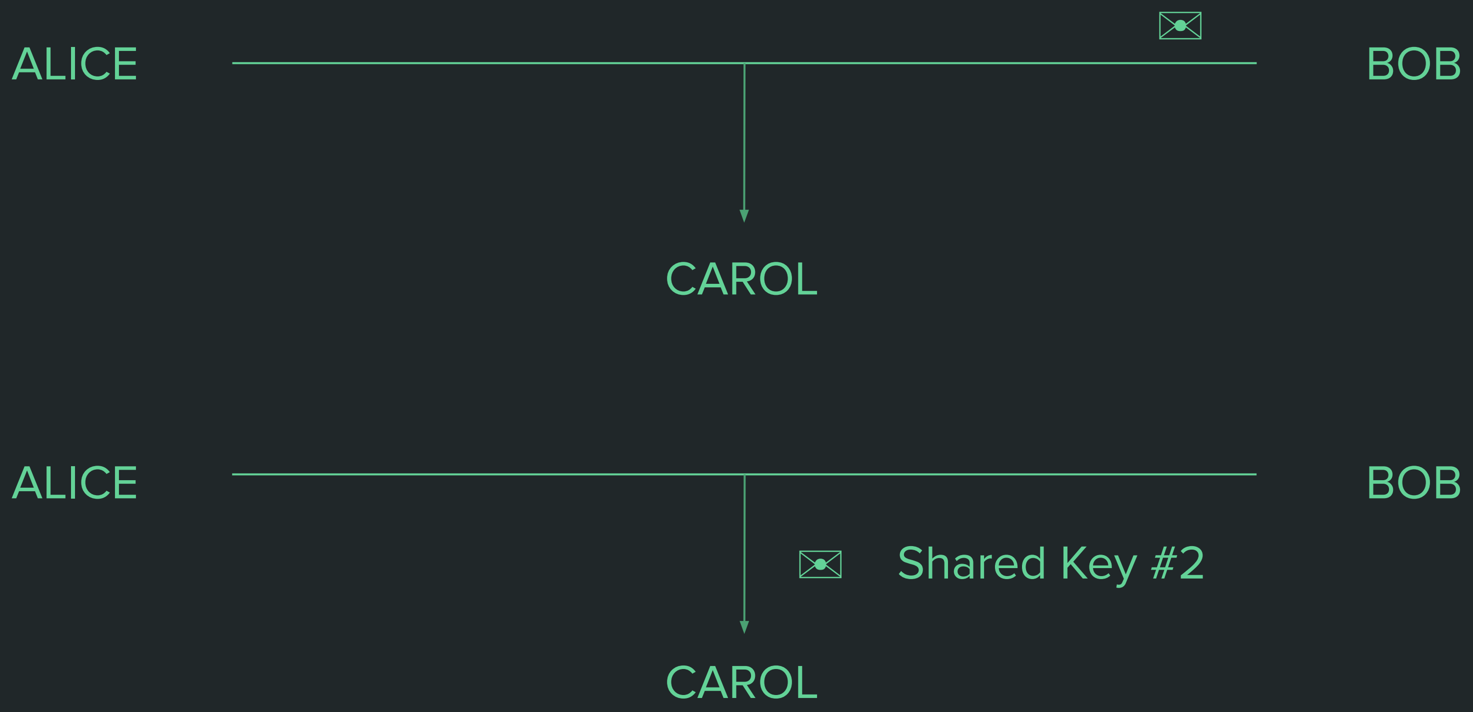

ALICE
BOB
CAROL
ALICE
BOB
Shared key #1
CAROL


ALICE
BOB
CAROL

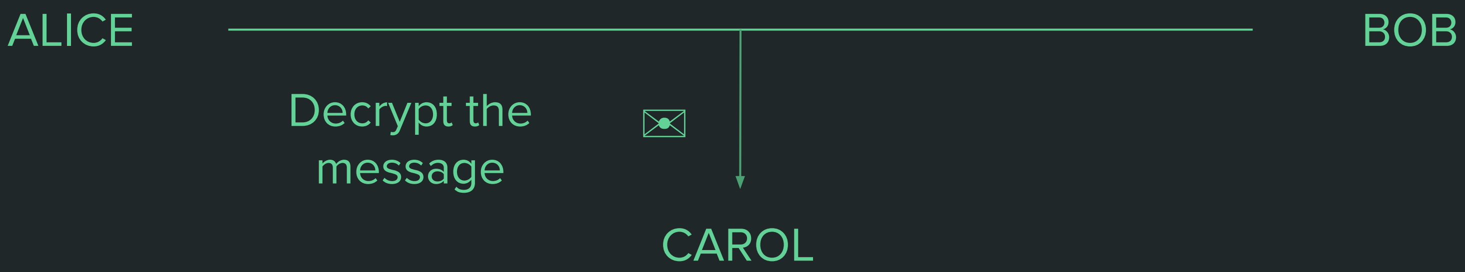

ALICE
BOB
CAROL
ALICE
BOB
CAROL


ALICE
BOB
CAROL


ALICE
BOB
CAROL
ALICE
BOB
CAROL


ALICE
BOB
CAROL
ALICE
BOB
Shared Key #2
CAROL


ALICE
BOB

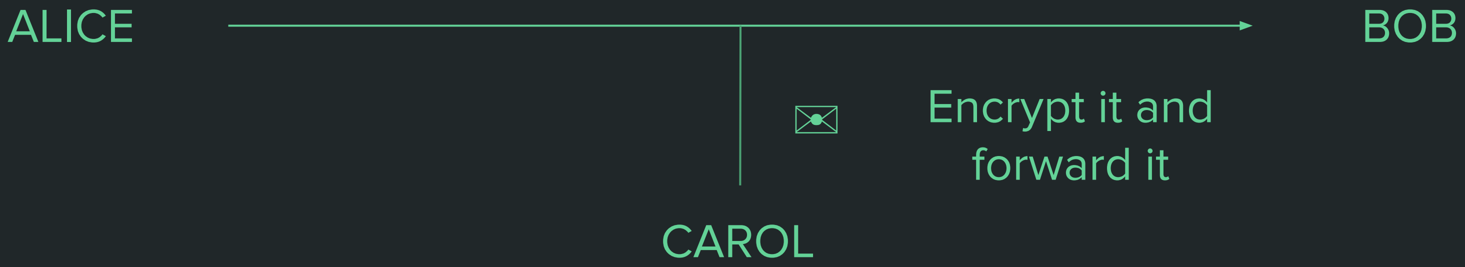

ALICE
BOB
Decrypt the message
CAROL

ALICE

BOB

CAROL

✉ Read the plaintext



Carol using the keys generated with Alice and Bob, can easily eavesdrop over the channel

Challenge time

Solving DLP

Solving DLP

There are algorithms that try to solve the DLP:

- Baby step - Giant step
- Pohlig - Hellman
- Pollard’s Rho
- Shor’s algorithm