Real-world Cryptography

Eleanor McMurtry (@noneuclideangrl)

About me

- M.Sc. (Computer Science) student at University of Melbourne — about to graduate!
- *Published Author* (very fancy)
- I really like cryptography and also breaking stuff

What is cryptography?

- Keeping information in the hands of the good guys (and out of the hands of the bad guys)
- **Confidentiality:** encrypting messages
- **Integrity:** detecting if someone alters a message
- **Authentication:** confirming who sent a message

Why cryptography?

- An extra layer of defence to go with other security policies in AppSec etc.
- It’s the foundation of many security systems you use every day!

Goals

- Understand the idea behind:
  - symmetric cryptography and key agreement
  - digital signatures
  - message authentication
- Write a very simple end-to-end encrypted message system

Non-goals

- Understand the maths of crypto (that’s a whole other workshop)
- Certificate chains, Trusting Trust (https://www.win.tue.nl/~aeb/linux/hh/thompson/trust.html)
- Anything related to cryptocurrency

Non-goals

- Understand the maths of crypto (that’s a whole other workshop)
- Certificate chains, Trusting Trust (https://www.win.tue.nl/~aeb/linux/hh/thompson/trust.html)
- Anything related to cryptocurrency
    - Just don’t.

Symmetric-key cryptography

Alice and Bob share a **secret key** that can be used to **encrypt** and **decrypt** messages.

Classic example: the **Caesar cipher**.

A B C D E F G …

A B C D E F G …

hello world

jgnnq yqtnf

Here, the key is **2**: shift **2** places to the right.

Symmetric-key cryptography

Real-world ciphers (AES, XChaCha20, …) are more complex:

1. Convert message to numbers (ASCII) — hello world → 68 65 6c 6c 6f …
2. Apply the cipher to numbers — 68 65 6c 6c 6f … → 6b 95 d3 b3 1d …
3. Convert numbers back to text (base 64) — 6b 95 d3 … → a5XTsx2zYDeBjCNBwlgNhQ

See **Exercise 1**.

Key agreement

How are Alice and Bob going to **securely** come up with a secret key?

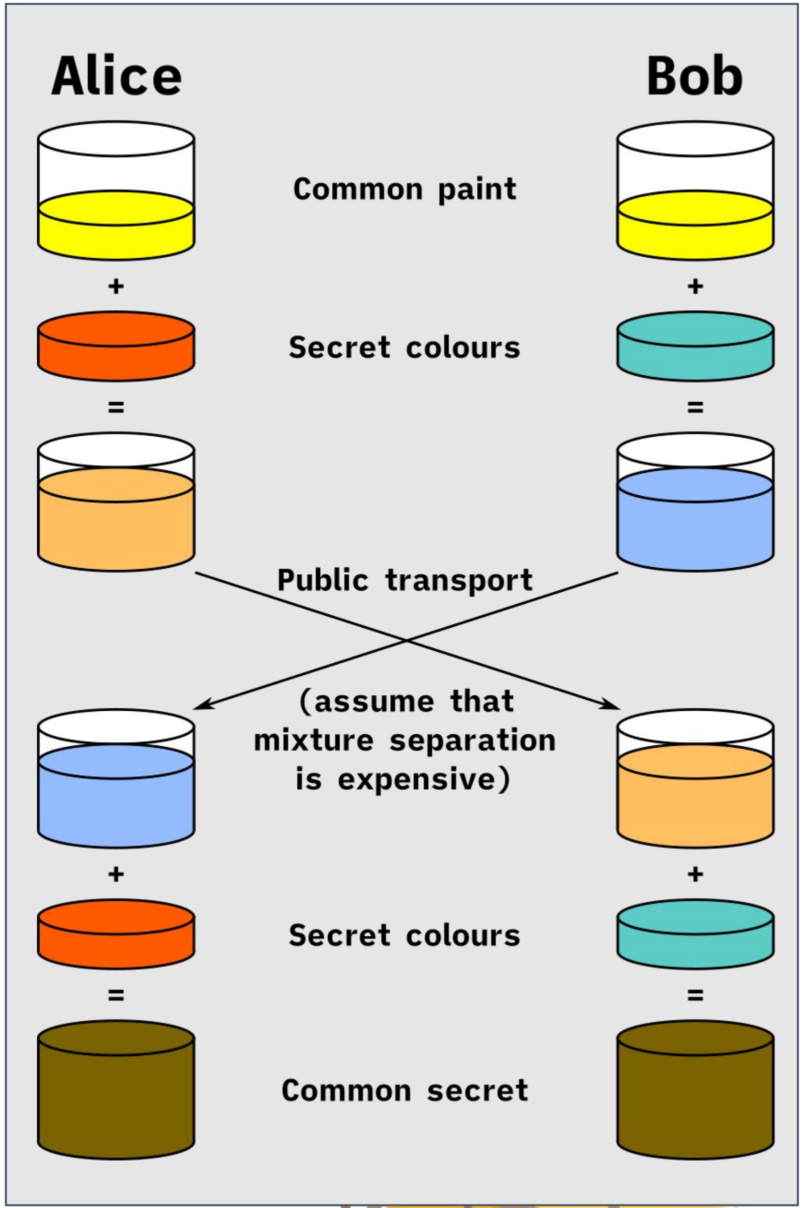
- Alice emails it to Bob?
- Alice tells Bob over a phone call?
- Alice gives Bob a copy of the key in person?

Key agreement

How are Alice and Bob going to **securely** come up with a secret key?

In 1976, Diffie & Hellman came up with a **computationally** secure method. Alice:

1. Agrees on a **generator** with Bob
2. Combines her **secret** with the generator, and sends it to Bob
3. Combines her secret with the **result** from Bob

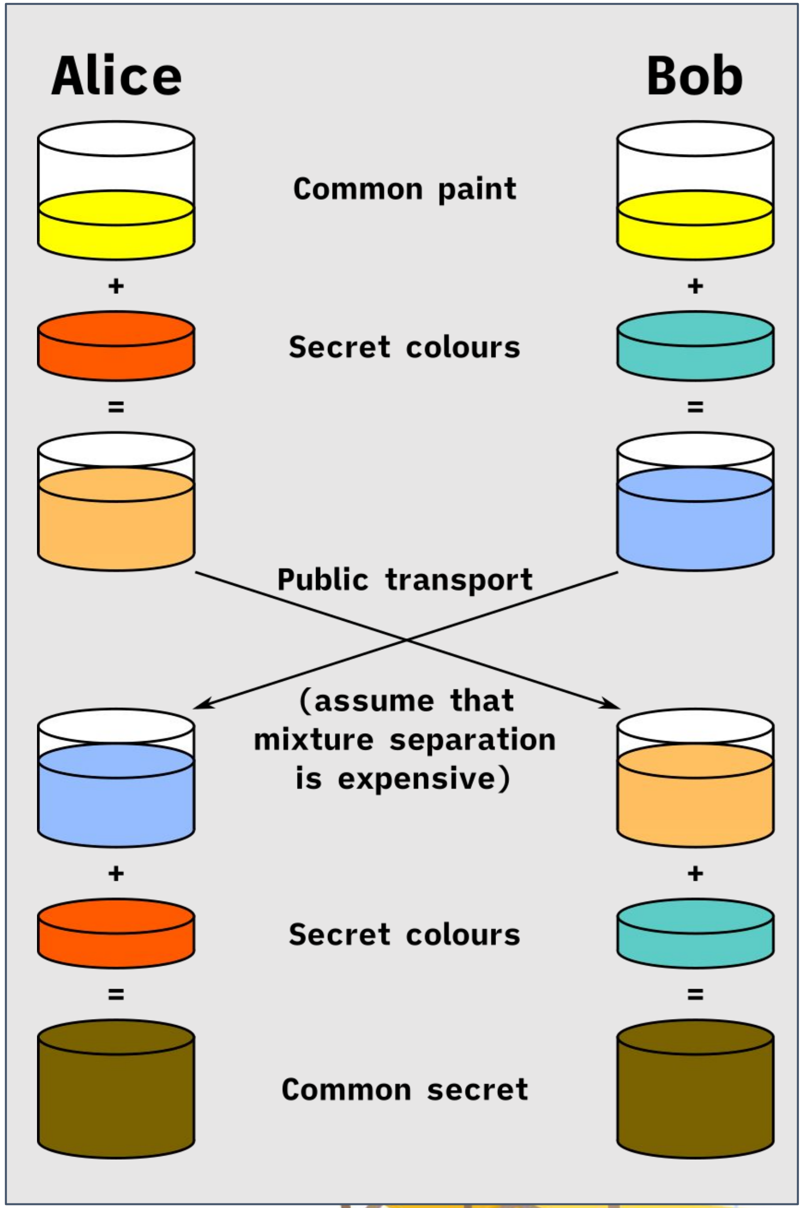


Key agreement

This is the **Diffie-Hellman (DH) protocol**. Modern variants use elliptic curves, so it's called **ECDH**.

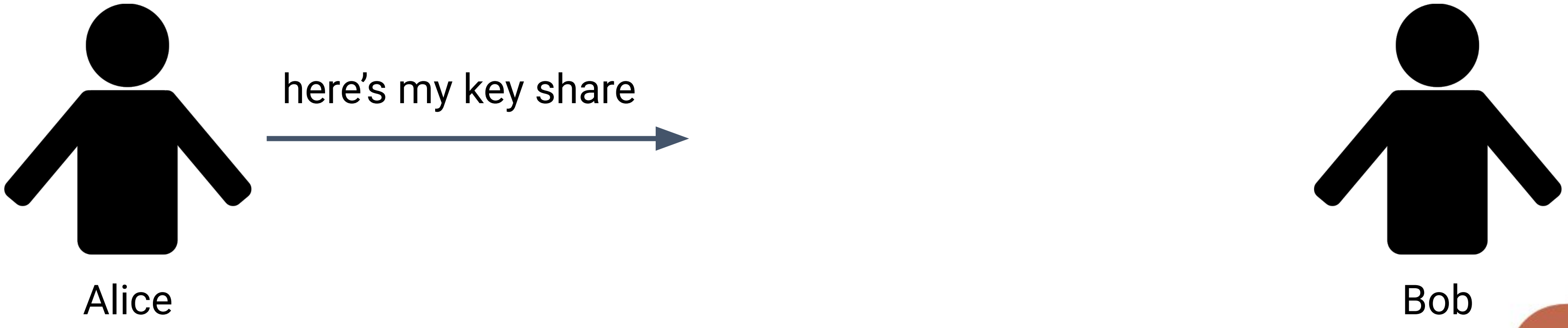
See **Exercise 2**.

For the mathematically-inclined: choose a group with a generator g. Alice and Bob choose powers a, b. Alice sends Bob g^a and Bob sends Alice g^b. Alice calculates $(g^b)^a$ and Bob calculates $(g^a)^b$.



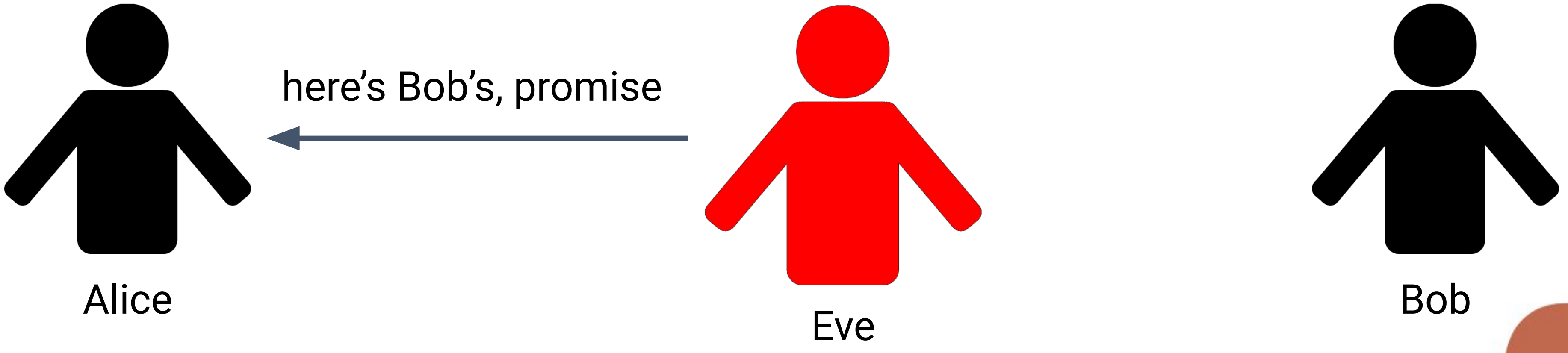
The story so far

Alice and Bob can share a secret key, and can communicate securely with the key. Are we done?



The story so far

Alice and Bob can share a secret key, and can communicate securely with the key. Are we done?



Digital signatures

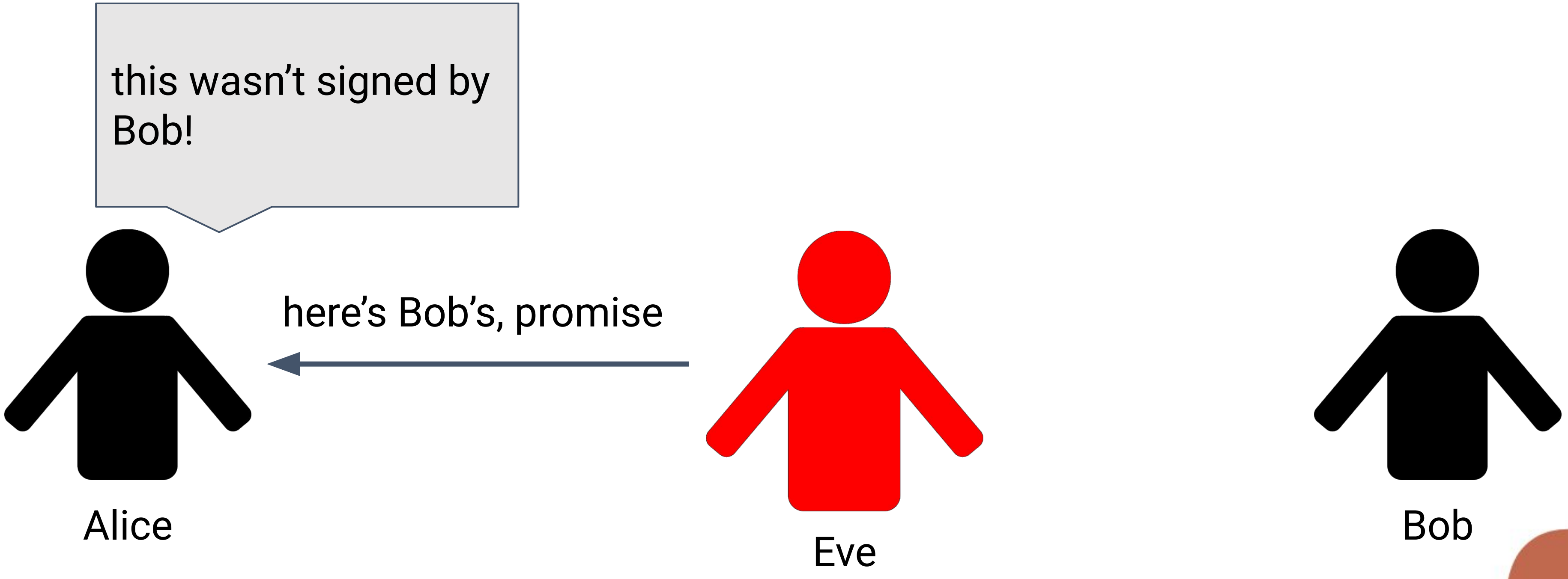
We need to **authenticate** messages.

Alice creates two keys: the **signing key** (kept secret) and the **verifying key** (made public).

Sign(message, signing key) = (message, signature)

Verify(message, signature, verifying key) = true or false

Digital signatures



Digital signatures

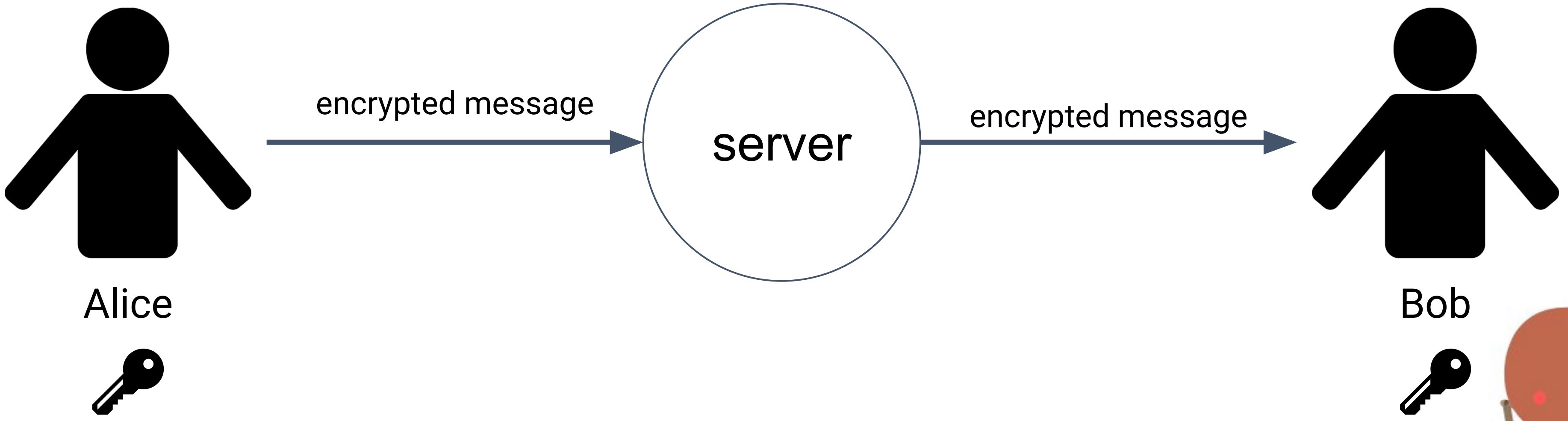
Signatures also provide **integrity**: a signature of one message is not a signature of another.

(This is done behind the scenes using a hash function. Ask me if you're curious!)

See **Exercise 3**.

End-to-end encryption

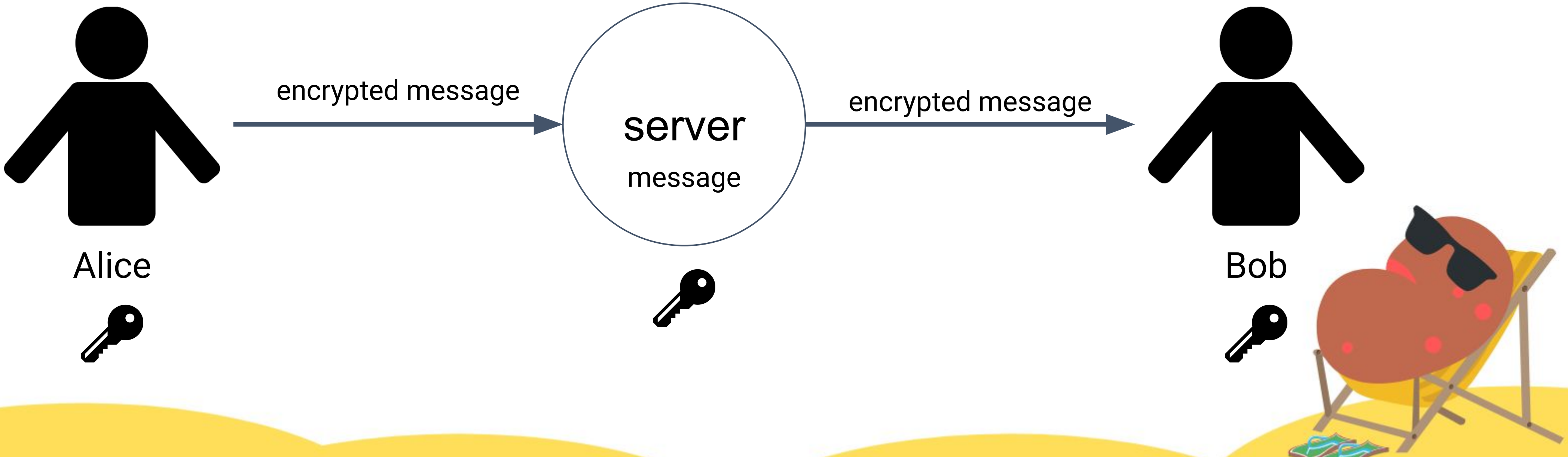
In a peer-to-peer system, **only the peers** can decrypt messages (not a federating server).



End-to-end encryption

In a peer-to-peer system, **only the peers** can decrypt messages (not a federating server).

NOT end-to-end encryption: (looking at you, Zoom)



End-to-end encryption

In a peer-to-peer system, **only the peers** can decrypt messages (not a federating server).

Game plan:

1. Alice and Bob share a key with ECDH, signing their messages
2. Alice and Bob send encrypted and signed messages

End-to-end encryption

In a peer-to-peer system, **only the peers** can decrypt messages (not a federating server).

Game plan:

1. Alice and Bob share a key with ECDH, signing their messages
2. Alice and Bob send encrypted and signed messages

Problem: digital signatures are really slow.

Message authentication codes (MACs)

- Similar to signatures, but use a **shared secret key**
- Add an **authentication tag** to the message

Sign(message, MAC key) = (message, auth tag)

Verify(message, auth tag, MAC key) = true or false

Case study: COVIDSafe app.

AES-256-GCM (Exercise 1) actually does this automatically. :)

Summary

- You’ve learnt the most important pieces of crypto technology used today.
- TLS, HTTPS, etc. all use these building blocks.
- **Remember:** this knowledge is an **extra layer** on top of usual AppSec.
- Some further reading using fancier techniques:
  https://soatok.blog/2020/11/14/going-bark-a-furrys-guide-to-end-to-end-encryption/