Just the cryptography you need to know for TLS

Lerna Ekmekcioglu

Sr Solutions Architect, AWS

Thoughts and opinions are my own and do not represent that of my employer

TLS 1.3 🐋

Image by Bianca Van Dijk from Pixabay

Origins of cryptography

Ancient Egypt
4000 years ago

First code writing with unusual hieroglyphs

Hieroglyphs (CC0 1.0)

Mesopotamia
3500 years ago

Phonetic encryption on clay tablets

Clay tablets (CC0 1.0)

Spartans
2600 years ago

First use for correspondance

Let’s travel back in time — in style

DeLorean Time Machine by JMortonPhoto.com & OtoGodfrey.com (CC BY-SA 4.0)

American revolutionary war: 1780

🧥 *British (redcoats)*

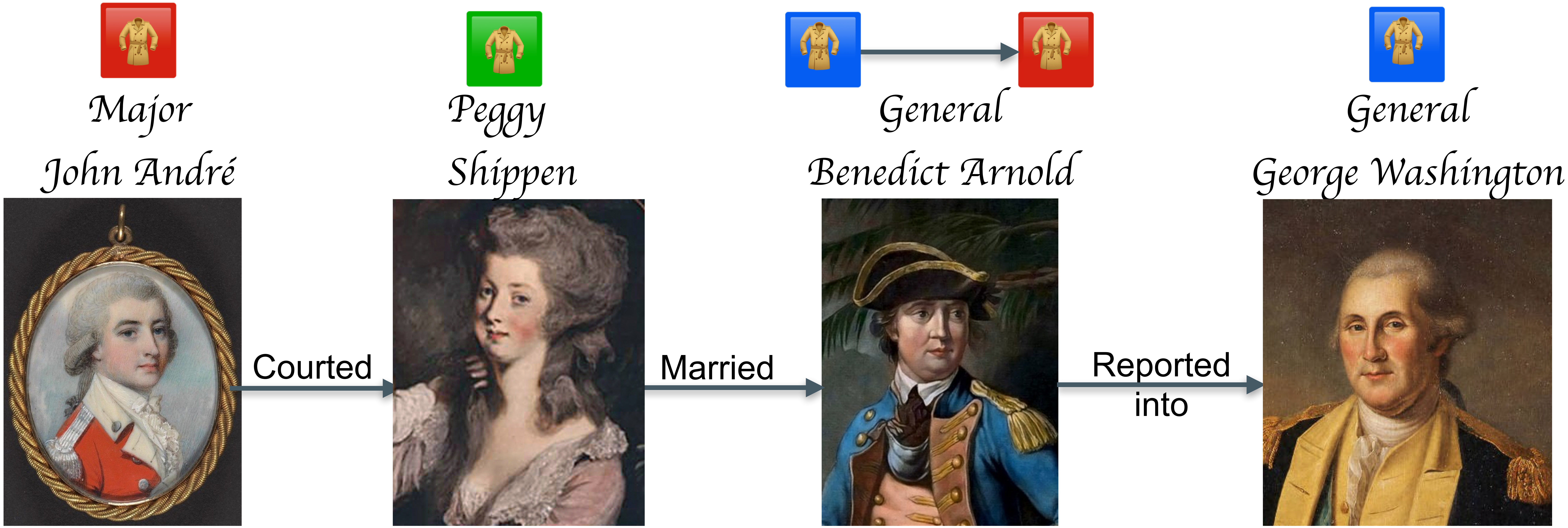
British old infantry uniforms (CC0 1.0)

⚔

🧥 *American (bluecoats)*

Continental infantry uniform (CC0 1.0)

Key characters



🟥 Major John André → Courted → 🟩 Peggy Shippen → Married → 🟦 → 🟥 General Benedict Arnold → Reported into → 🟦 General George Washington

By George Engleheart, Yale Center for British Art (CC0 1.0)

By Daniel Gardner (CC0 1.0)

By Thomas Hart (CC0 1.0)

"George Washington after the Battle of Princeton (Charles Willson Peale), PP218," Princeton University Art Museums collections online, https://artmuseum.princeton.edu/collections/objects/45234

Why cryptography?

🟥 *John*

Securely share sensitive data

over public channels

🟦 → 🟥 *Ben*

🟥 🗡 *American revolutionary war, 1780* 🗡 🟦

Why cryptography?

🟥🧥 John

PLAN
des Forts, Batteries
et Poste de West-Point
1780.
Nord
Sud
PARTIE
de l'Isle de
la Constitution.
Batterie et Bastion
HUDSON ou RIVIÈRE DU NORD
WEST-POINT
Fort Putnam
A. Magazin détruit.
B. Vieilles Casernes.
C. Magazin militaire
D. Muraille.
E. Embarcaderes.
F. Hôpital.
G. Batteries.
H. Horn-Point.
I. Colline de Bunker.
K. Duck-Point.
L. Jardin de Kosciusko
M. Prisons. Casernes.
N. Ecole du Génie.
O. Atteliers.
P. Etang.
R. Magazin milit.re
S. Bibliothèque.
T. Quartier-Général.
U. Laboratoire.
100
200
300 T. Françaises

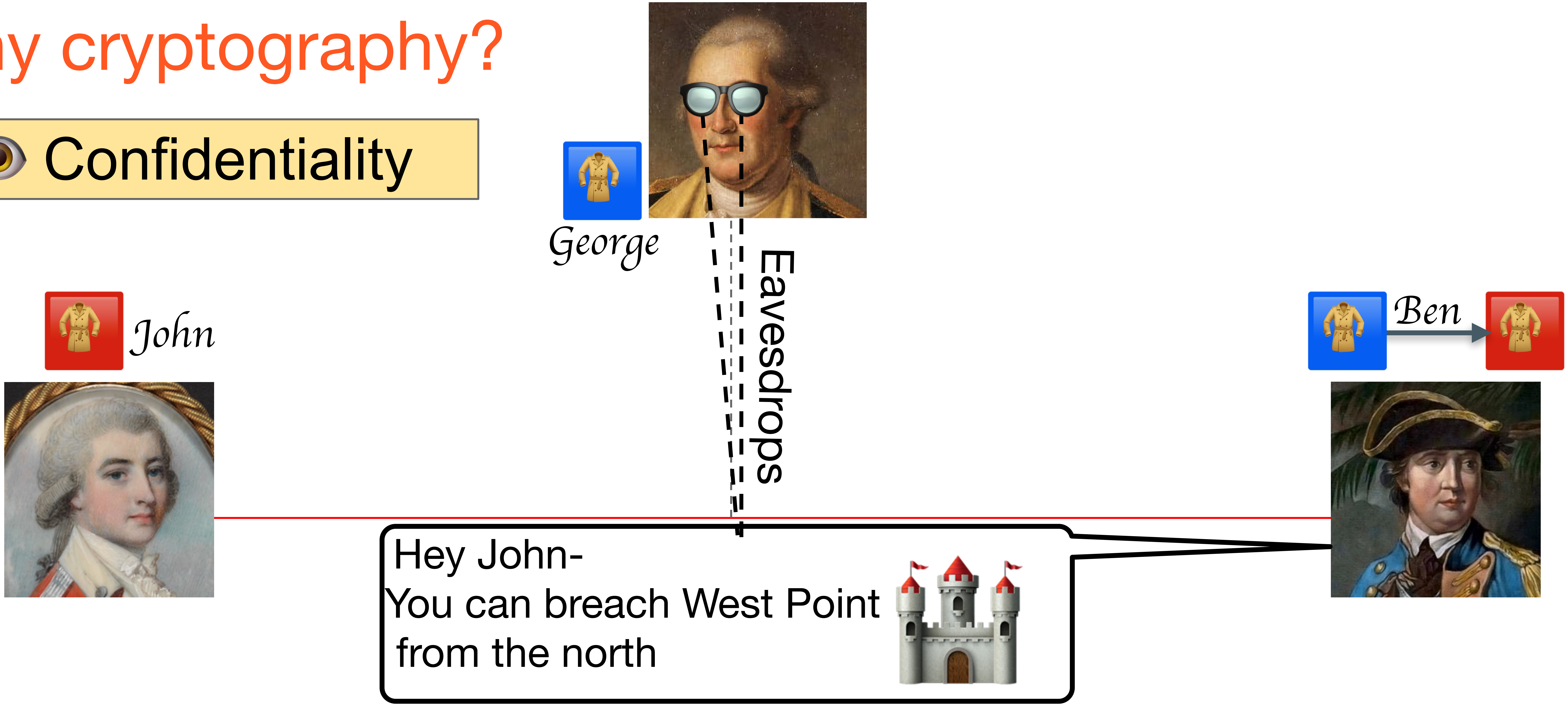
🟦🧥 → 🟥🧥 Ben

West Point by Pierre Didot, Boston Public Library Digital Map Collection (CC0 1.0)

🟥🧥 ⚔ *American revolutionary war, 1780* ⚔ 🟦🧥

Why cryptography?

👁 Confidentiality

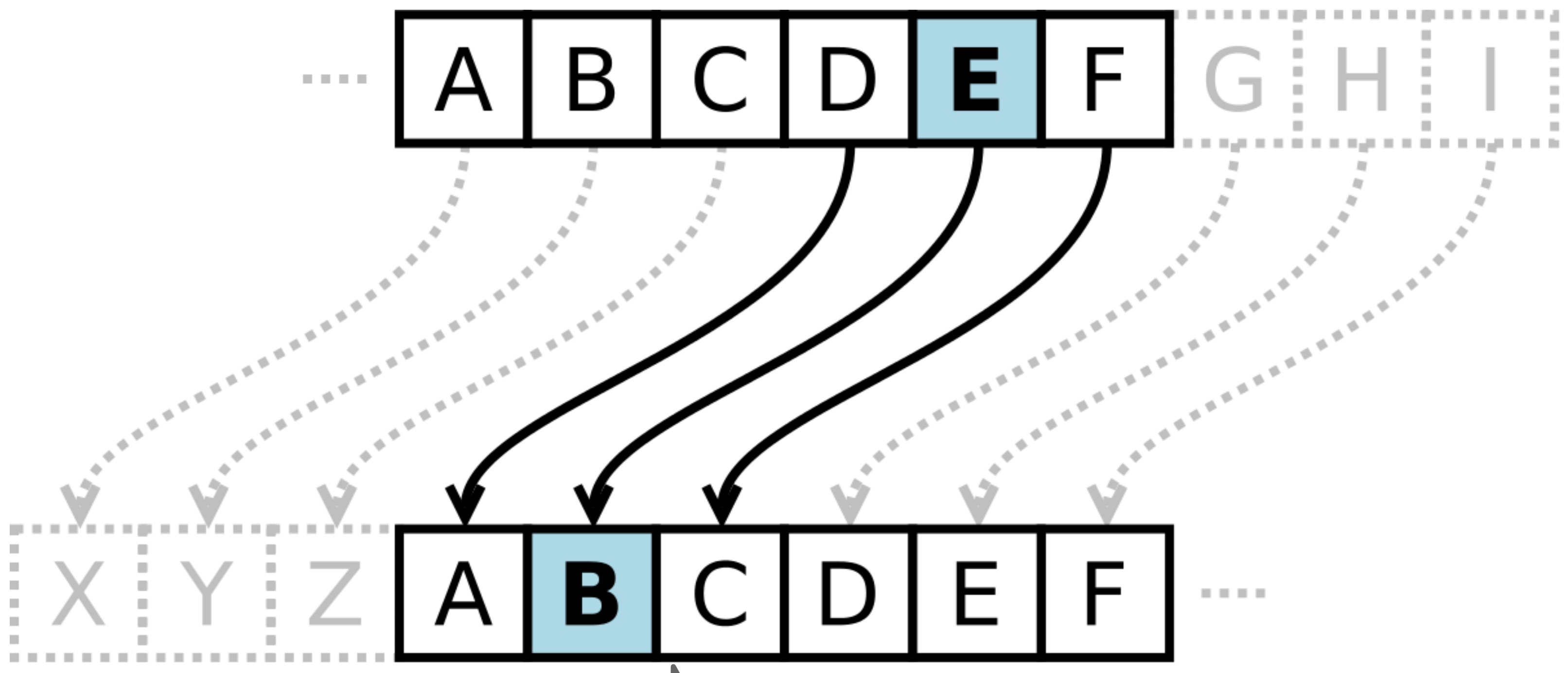


Hey John-
You can breach West Point
from the north

Confidentiality with symmetric key cryptography

Symmetric key = Encrypt and decrypt with the same key

```
Caesar(key=-3).encipher('Gauls are advancing to the west')
                        'DXRIPXOBXASXKZFKDQLQEBTBPQ'
Caesar(key=-3).decipher('DXRIPXOBXASXKZFKDQLQEBTBPQ')
                        'GAULSAREADVANCINGTOTHEWEST'
```



Caeser's substitution cipher

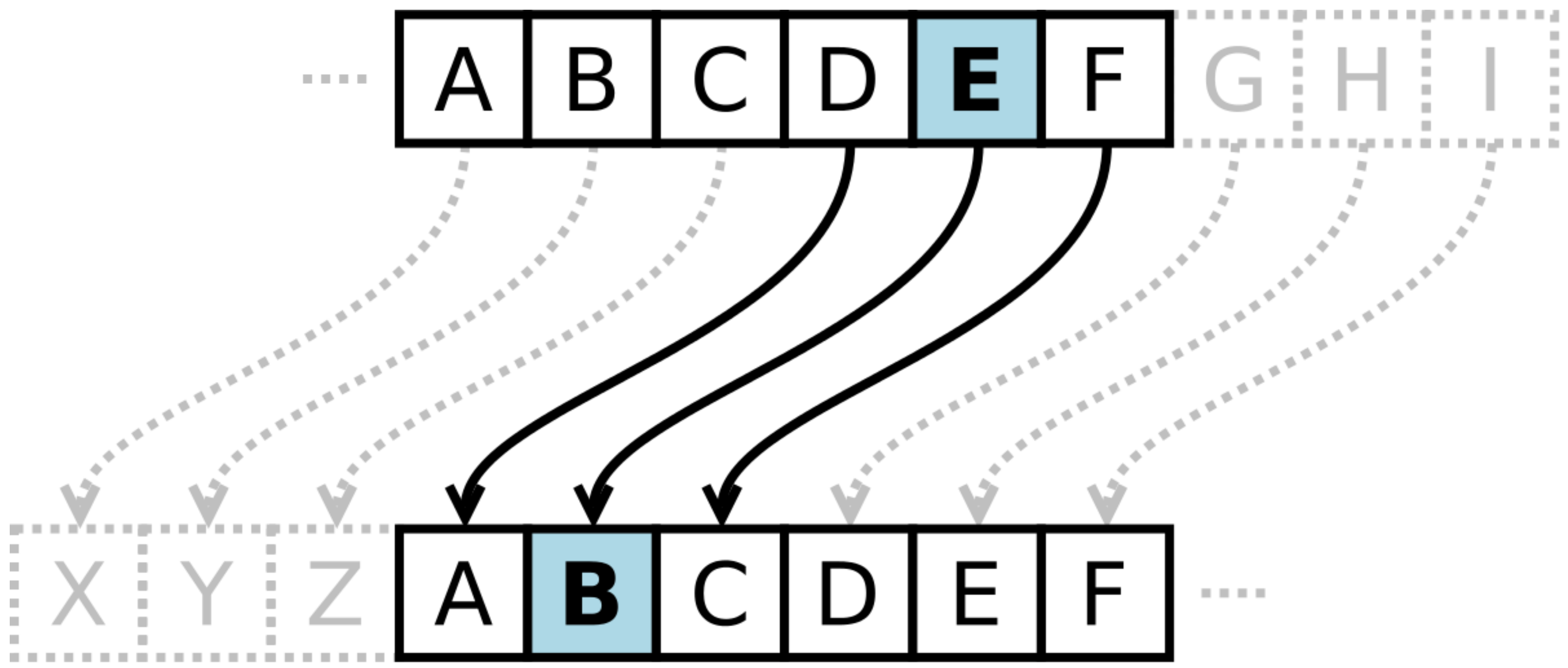
Caesar cipher with a shift of 3, CC0 1.0

Confidentiality with symmetric key cryptography

AES 🧱 and **ChaCha20** 💦 are common symmetric key algorithms.

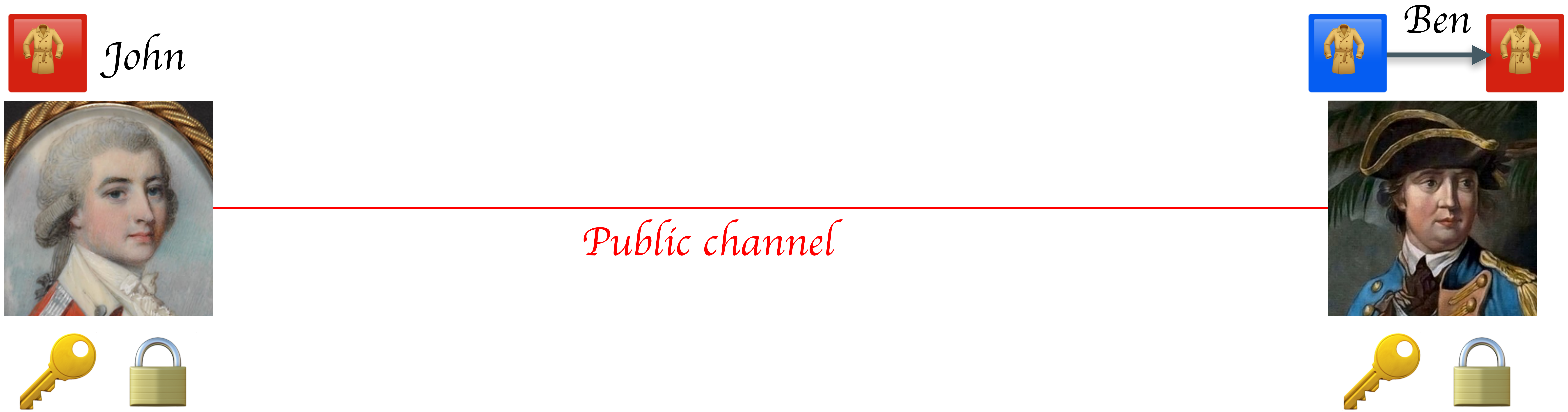
```
Caesar(key=-3).encipher('Gauls are advancing to the west')
                         ↓↓
                        ‘DXRIPXOBXASXKZFKDQLQEBTBPQ'

Caesar(key=-3).decipher('DXRIPXOBXASXKZFKDQLQEBTBPQ')
                         ↓↓
                        'GAULSAREADVANCINGTOTHEWEST'
```



Caesar cipher with a shift of 3, CC0 1.0

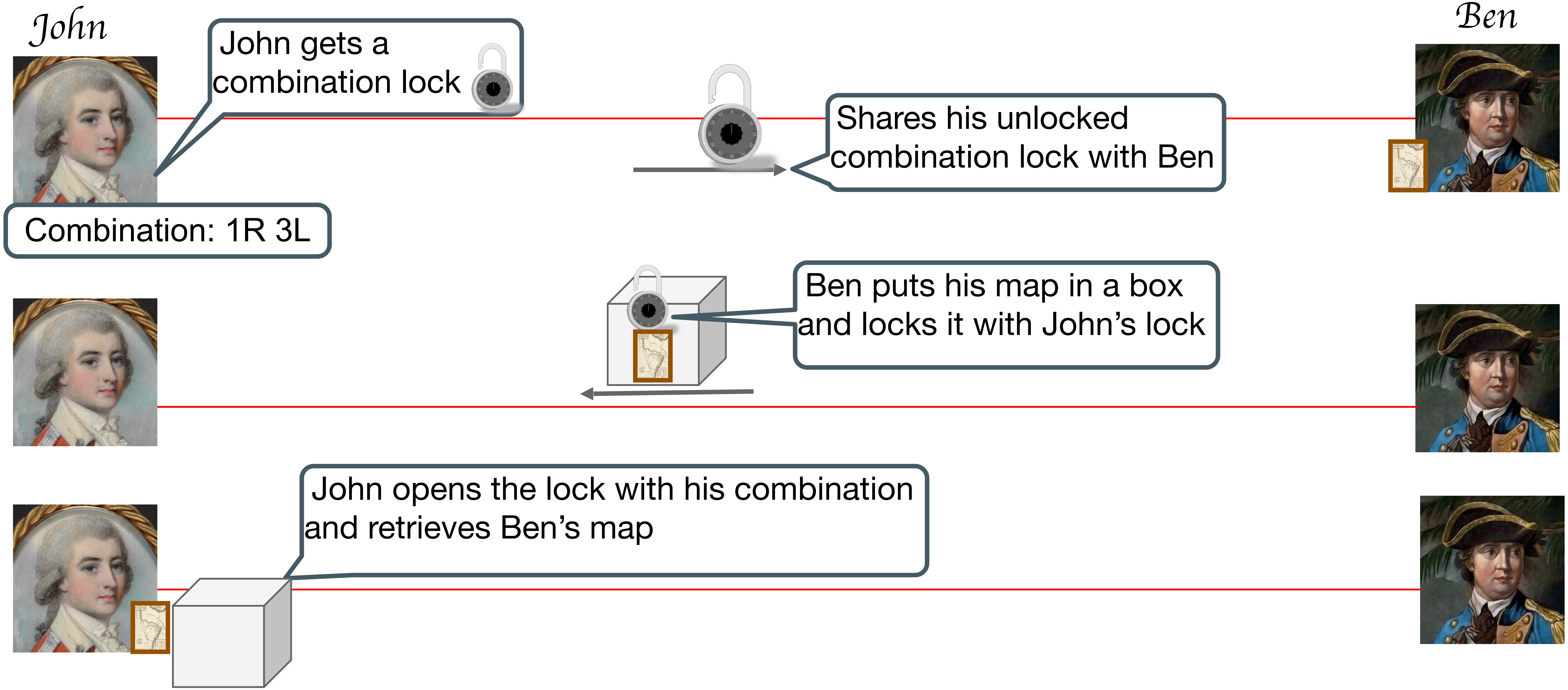
How to securely establish the symmetric key?



Use public key cryptography to establish symmetric key

🔒 ➡️⛔ 🔑

Public key cryptography analogy



John

Ben

John gets a combination lock

Combination: 1R 3L

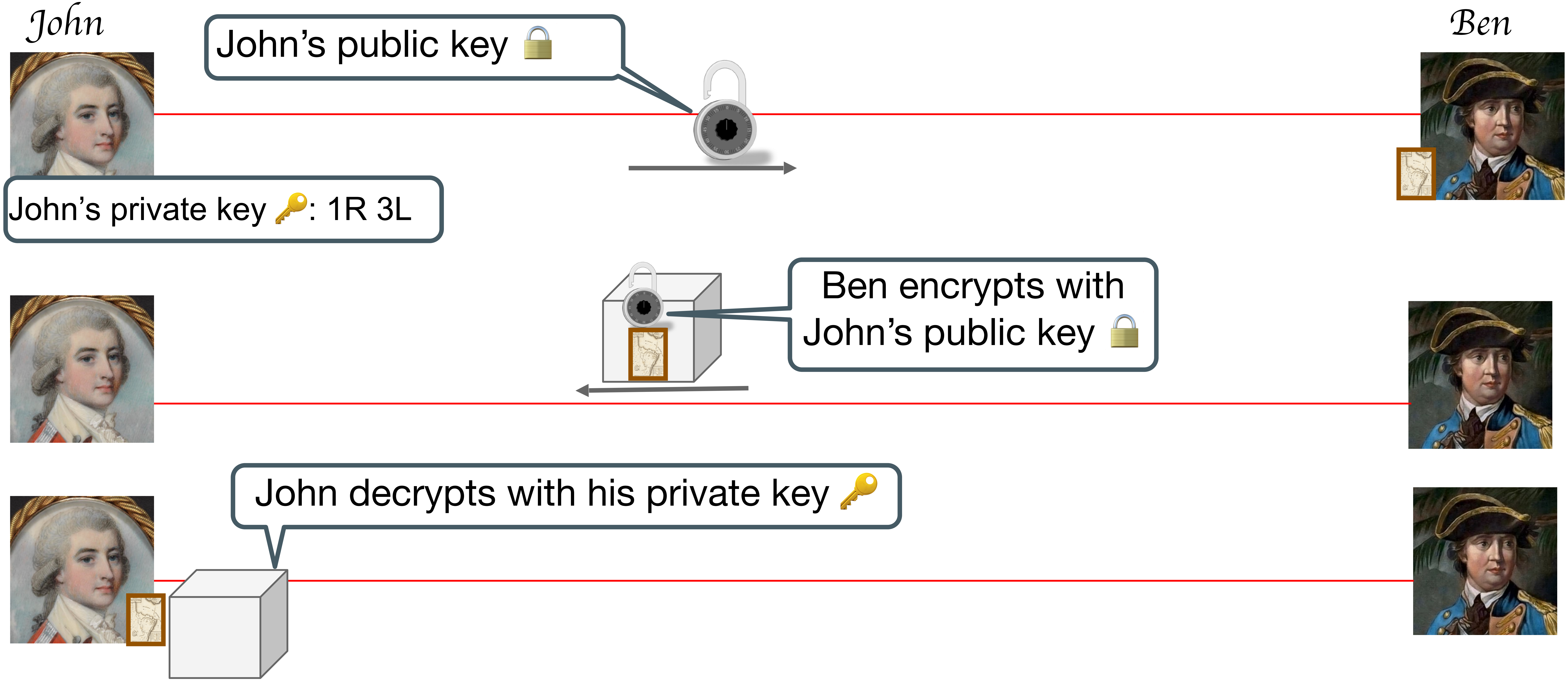
Shares his unlocked combination lock with Ben

Ben puts his map in a box and locks it with John’s lock

John opens the lock with his combination and retrieves Ben’s map

Combination lock (CC0 1.0)

Public key cryptography analogy



John

Ben

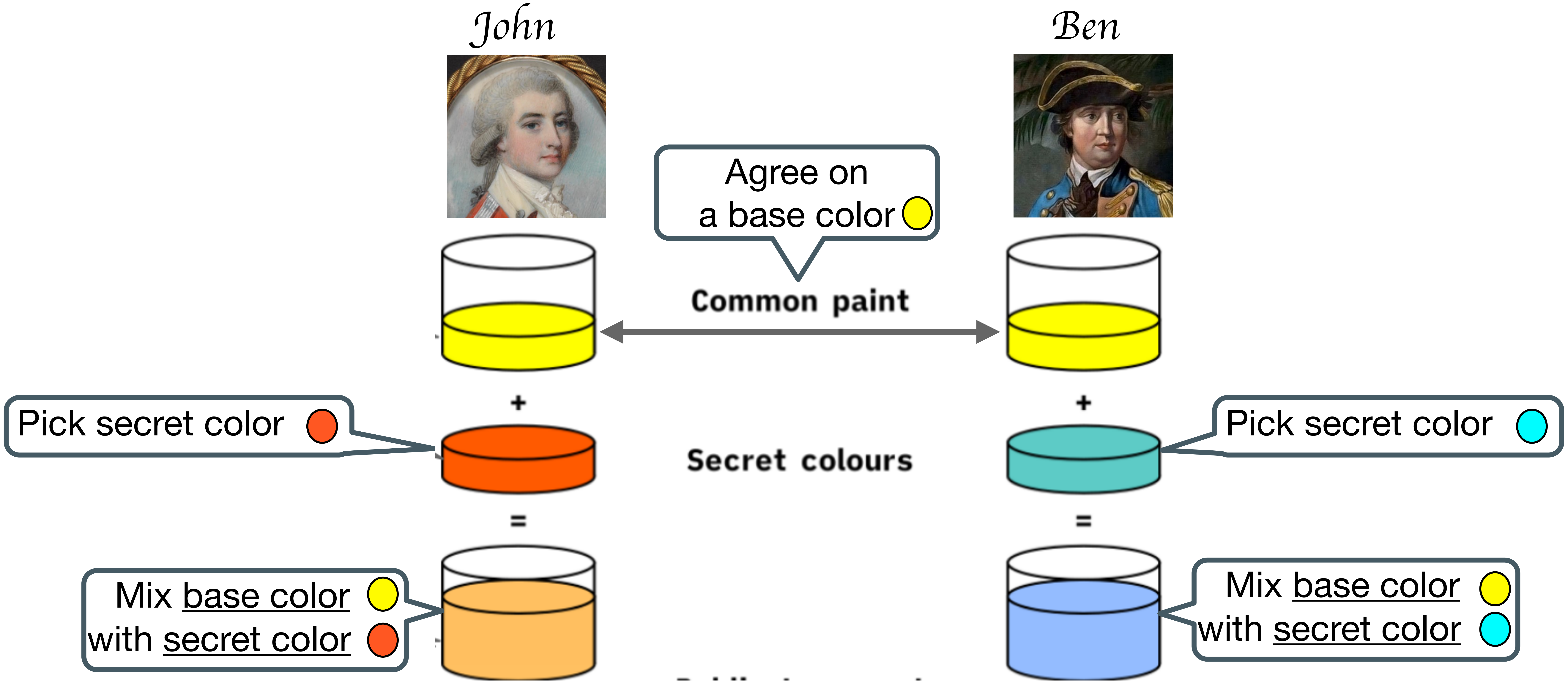
John’s public key 🔒

John’s private key 🔑: 1R 3L

Ben encrypts with John’s public key 🔒

John decrypts with his private key 🔑

Diffie-Hellman Key Exchange Analogy
Establish a shared key with public key cryptography



Agree on a base color ●

Pick secret color ●

Pick secret color ●

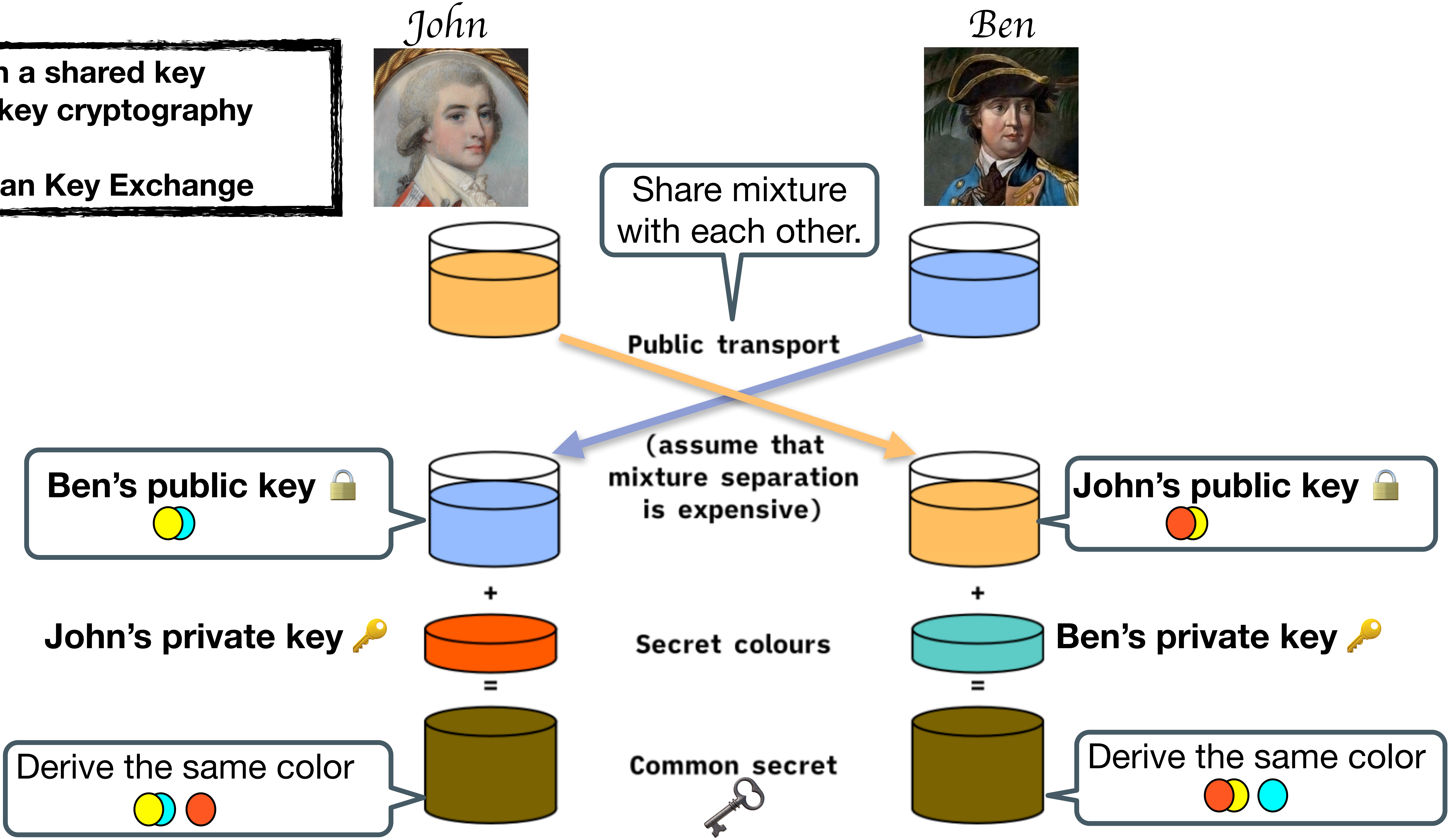
Mix base color ● with secret color ●

Mix base color ● with secret color ●

Diffie-Hellman Key Exchange Analogy (CC0 1.0)

Establish a shared key with public key cryptography

Diffie-Hellman Key Exchange



John

Ben

Share mixture with each other.

Public transport

(assume that mixture separation is expensive)

Ben’s public key 🔒
🟡🔵

John’s public key 🔒
🔴🟡

+

John’s private key 🔑

Secret colours

Ben’s private key 🔑

=

Derive the same color
🟡🔵 🔴

Common secret 🗝️

Derive the same color
🔴🟡 🔵

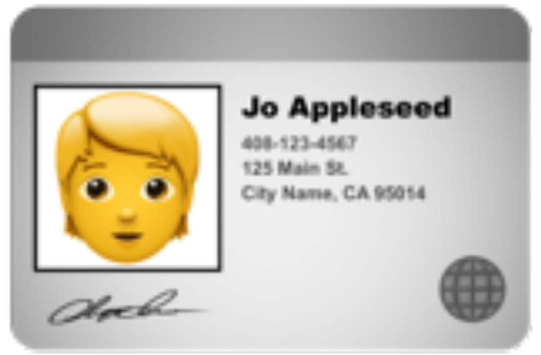
[Diffie-Hellman Key Exchange Analogy](#) ([CC0 1.0](#))

Why cryptography?

John

Am I really talking to Ben?

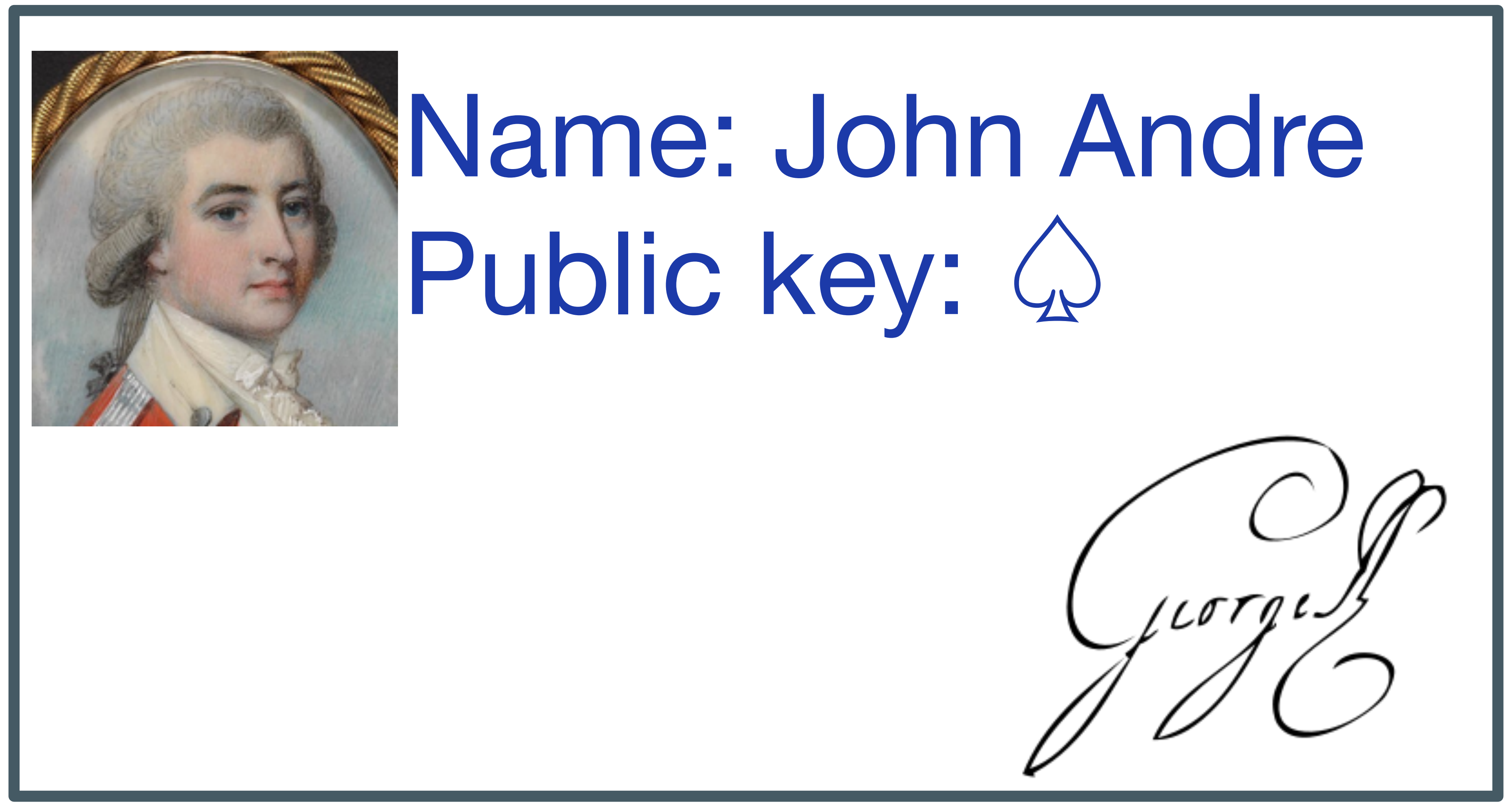
Authentication



Ben

Am I really talking to John?

How can John show proof of his identity?



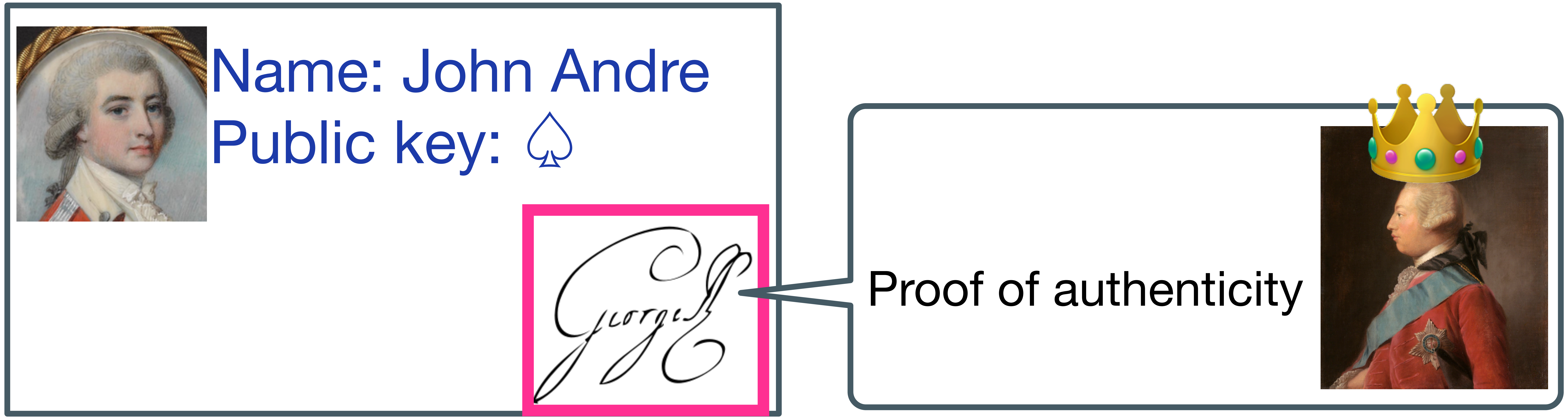
Name: John Andre
Public key: ♤

George R

Certificate Authority (CA)
issues a certificate
to John

Certificate binds John’s public key ♤ to a unique name identifying John

How can John show proof of his identity?



Name: John Andre

Public key: ♤

Proof of authenticity

CA *digitally signs cert* =

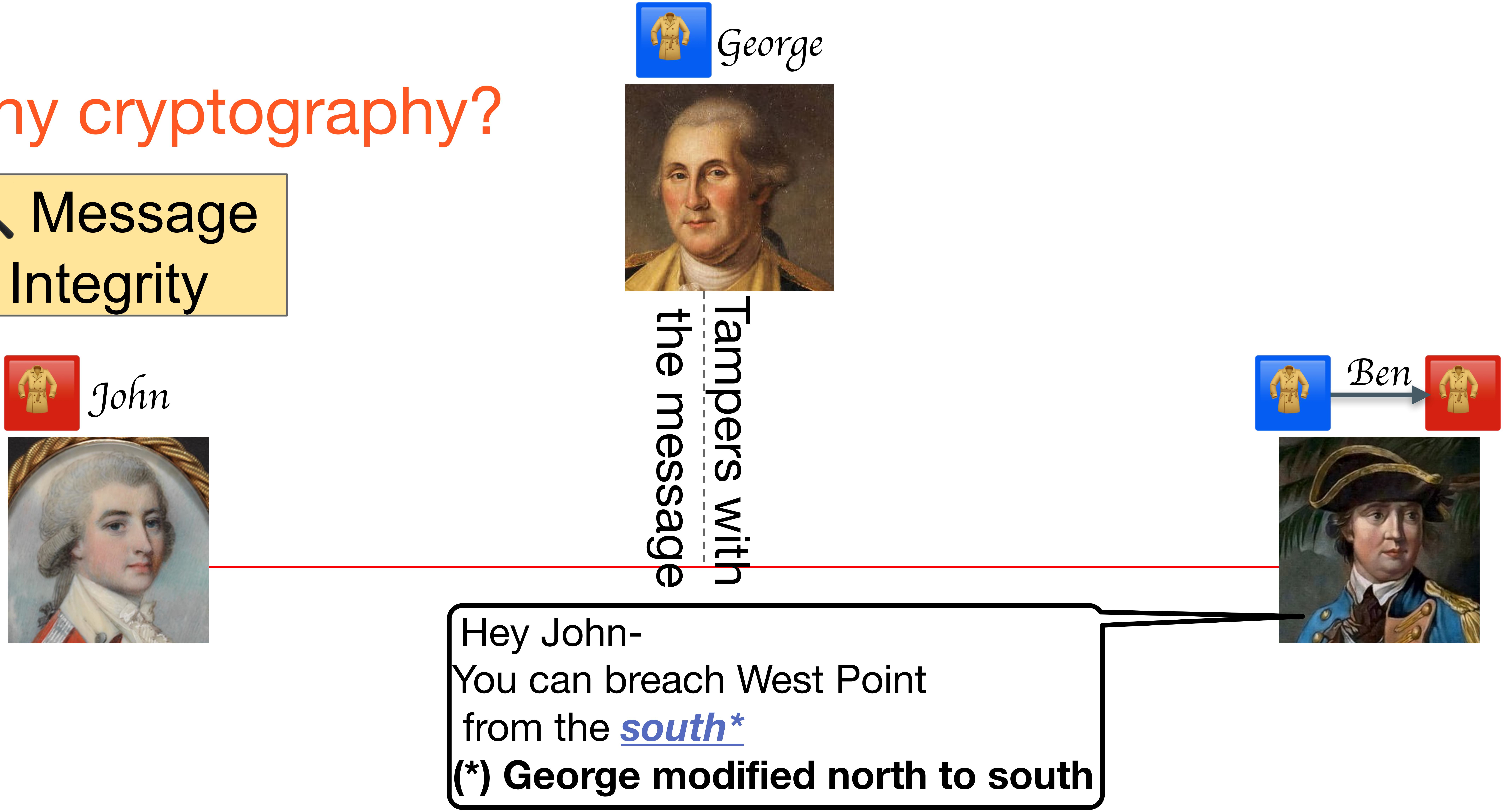
CA *encrypts hash of cert data with* (CA's) *own private key*

Portrait of George III by Allan Ramsey, CC0 1.0

King George III signature (CC0 1.0)

Why cryptography?

🔨 Message Integrity



Hey John-
You can breach West Point
from the ***south****

(*) George modified north to south

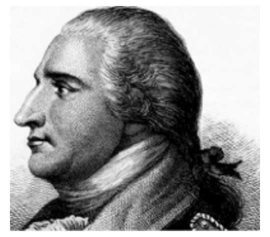
TLS 1.3

1. Handshake ❦

❦ The first time John and Ben meet

2. Application data

CLIENT (JOHN)



SERVER (BEN)

by H.B. Hall, National Archives and Records Administration, CC0 1.0

TLS 1.3 🤝

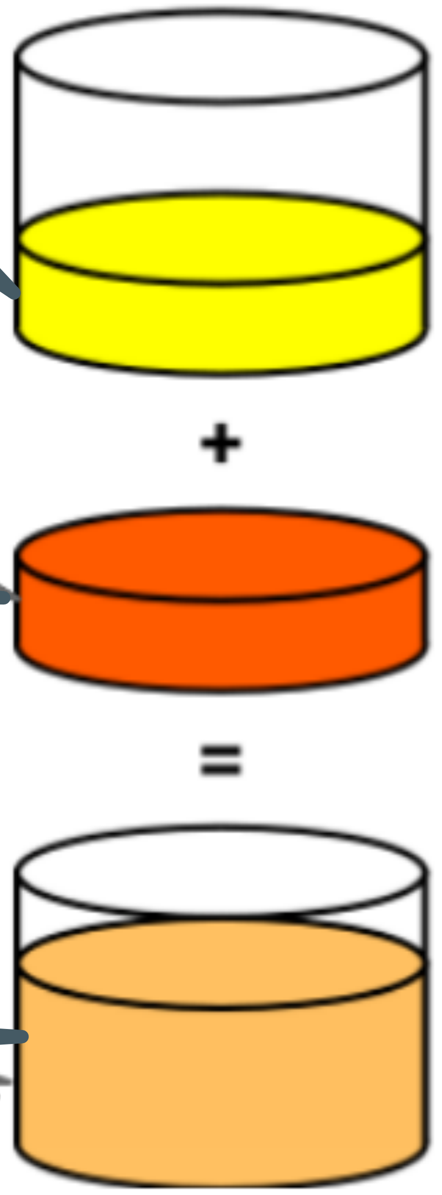
🩳🔑💪

generates a ECDHE key pair 🔐

Diffie **H**ellman
Ephemeral

Forward Secrecy 🐋

Assumes common "popular" params ⭐⭐⭐⭐⭐



Elliptic curve 25519 ⭐⭐⭐⭐⭐
with base point G=9

🔑 Picks private key

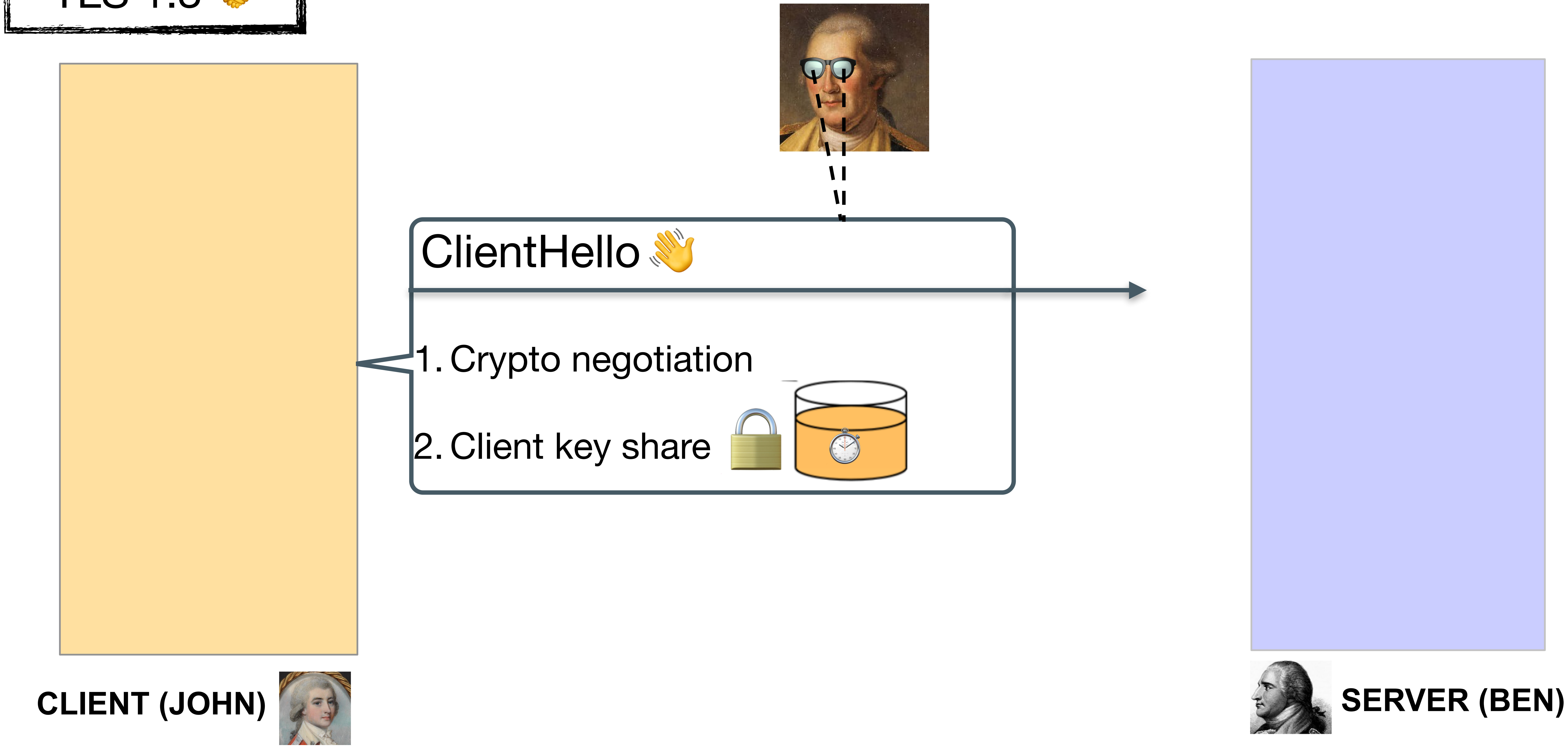
A random number (256 bit)

🔒 Calculates public key

$john\_private\_key * G$

Diffie-Hellman Key Exchange Analogy (CC0 1.0)

TLS 1.3 🤝



ClientHello 👋

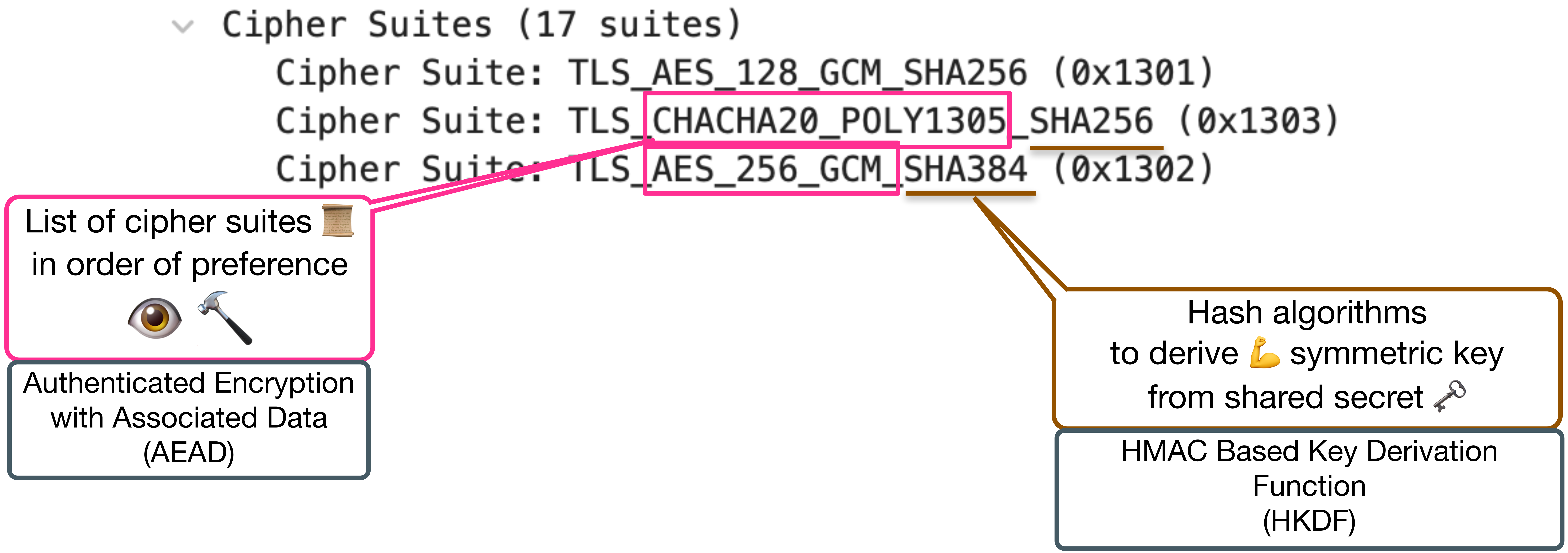
1. Crypto negotiation
2. Client key share 🔒

CLIENT (JOHN)

SERVER (BEN)

TLS 1.3 🤝

👋 Crypto negotiation



```
⌄ Cipher Suites (17 suites)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
```

List of cipher suites 📜 in order of preference 👁️ 🔨

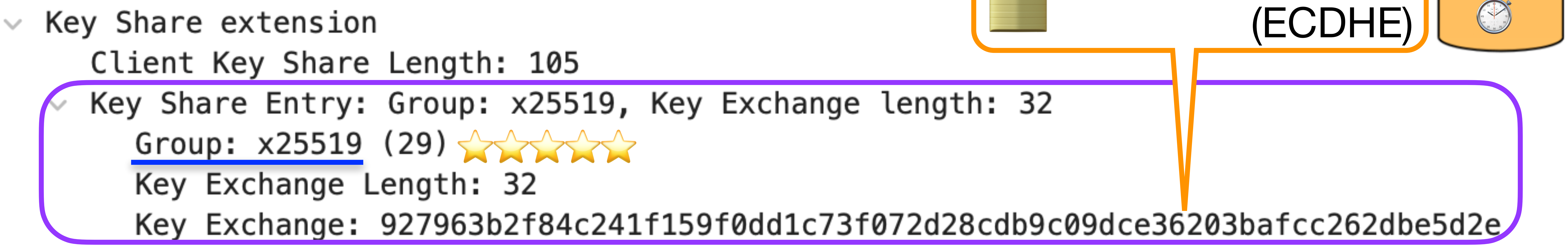
Authenticated Encryption with Associated Data (AEAD)

Hash algorithms to derive 💪 symmetric key from shared secret 🗝️

HMAC Based Key Derivation Function (HKDF)

👋 Client key share

🐋 Shorter 🤝



```
⌄ Key Share extension
    Client Key Share Length: 105
  ⌄ Key Share Entry: Group: x25519, Key Exchange length: 32
      Group: x25519 (29) ⭐⭐⭐⭐⭐
      Key Exchange Length: 32
      Key Exchange: 927963b2f84c241f159f0dd1c73f072d28cdb9c09dce36203bafcc262dbe5d2e
```

🔒 Client's public key (ECDHE) ⏱️

TLS 1.3 🤝

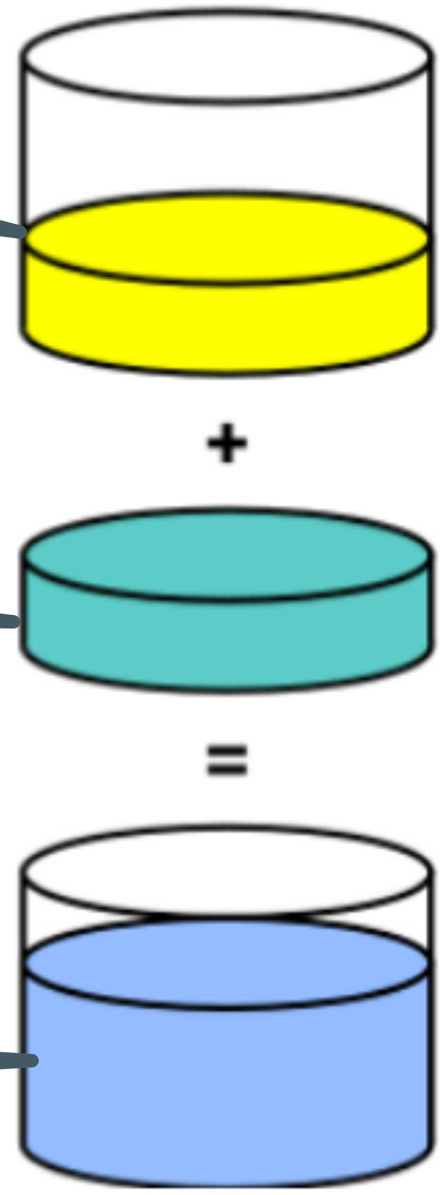
generates a key pair 🔐

ECDHE

Elliptic **C**urve **D**iffie **H**ellman

Ephemeral

Forward Secrecy 🐋



Agrees on common params ⭐⭐⭐⭐⭐

Elliptic curve 25519 ⭐⭐⭐⭐⭐
with base point G=9

🔑 Picks private key

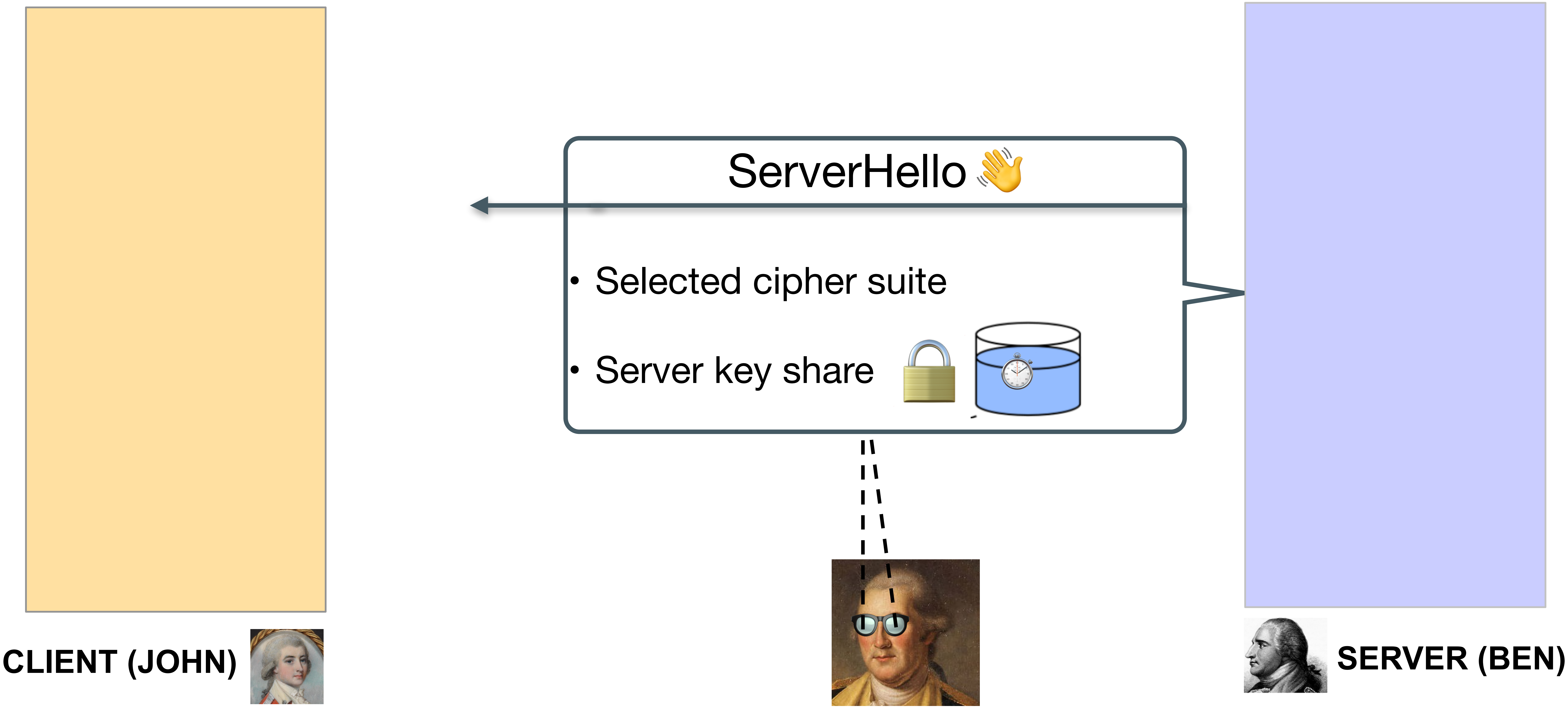
A random number (256 bit)

🔒 Calculates public key

$ben\_private\_key * G$

Diffie-Hellman Key Exchange Analogy (CC0 1.0)

TLS 1.3 🤝

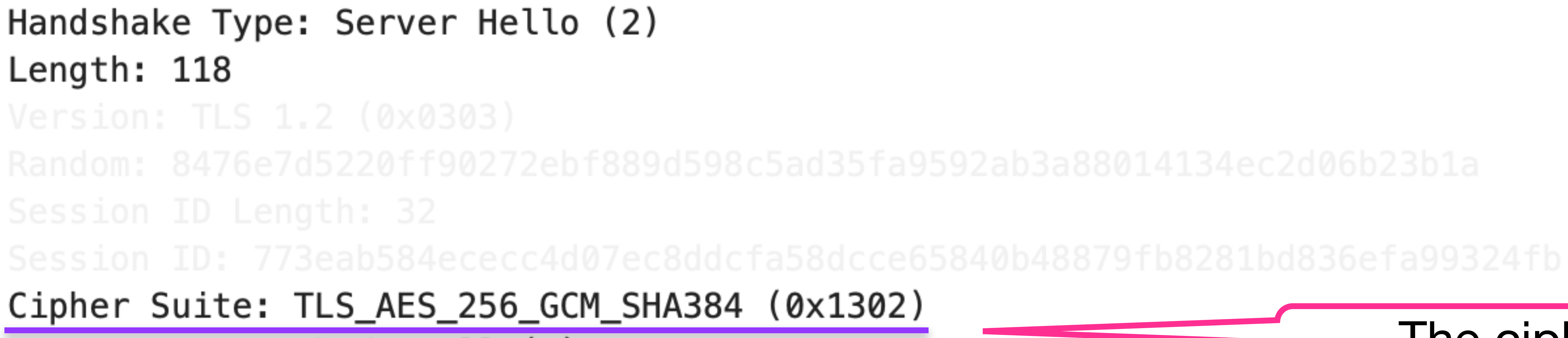


ServerHello 👋

- Selected cipher suite
- Server key share 🔒

TLS 1.3 🤝

👋 Selected cipher suite



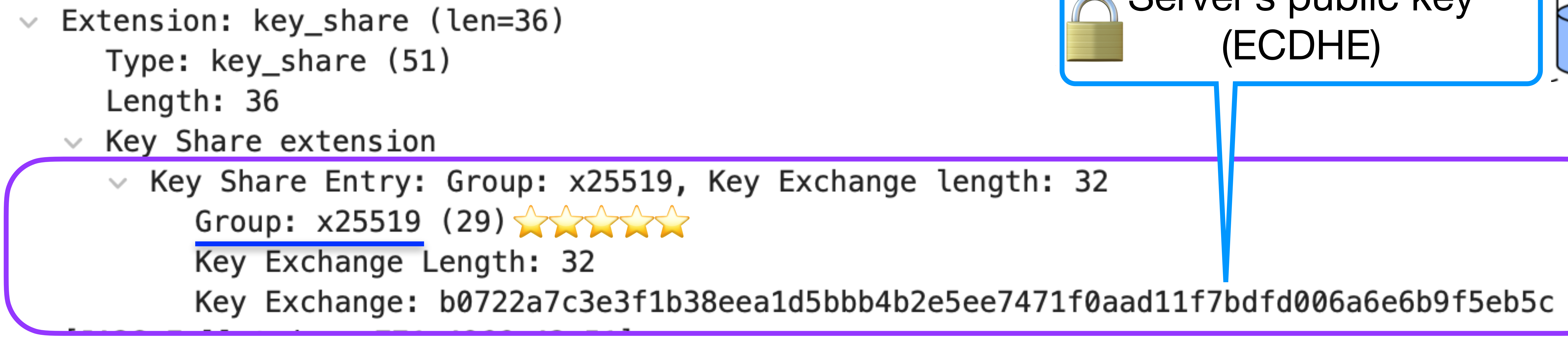
```
Handshake Type: Server Hello (2)
Length: 118
Version: TLS 1.2 (0x0303)
Random: 8476e7d5220ff90272ebf889d598c5ad35fa9592ab3a88014134ec2d06b23b1a
Session ID Length: 32
Session ID: 773eab584ececc4d07ec8ddcfa58dcce65840b48879fb8281bd836efa99324fb
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
```

The cipher suite selected by the server out of client's list 📜

```
Cipher Suites (17 suites)
```

TLS 1.3 🤝

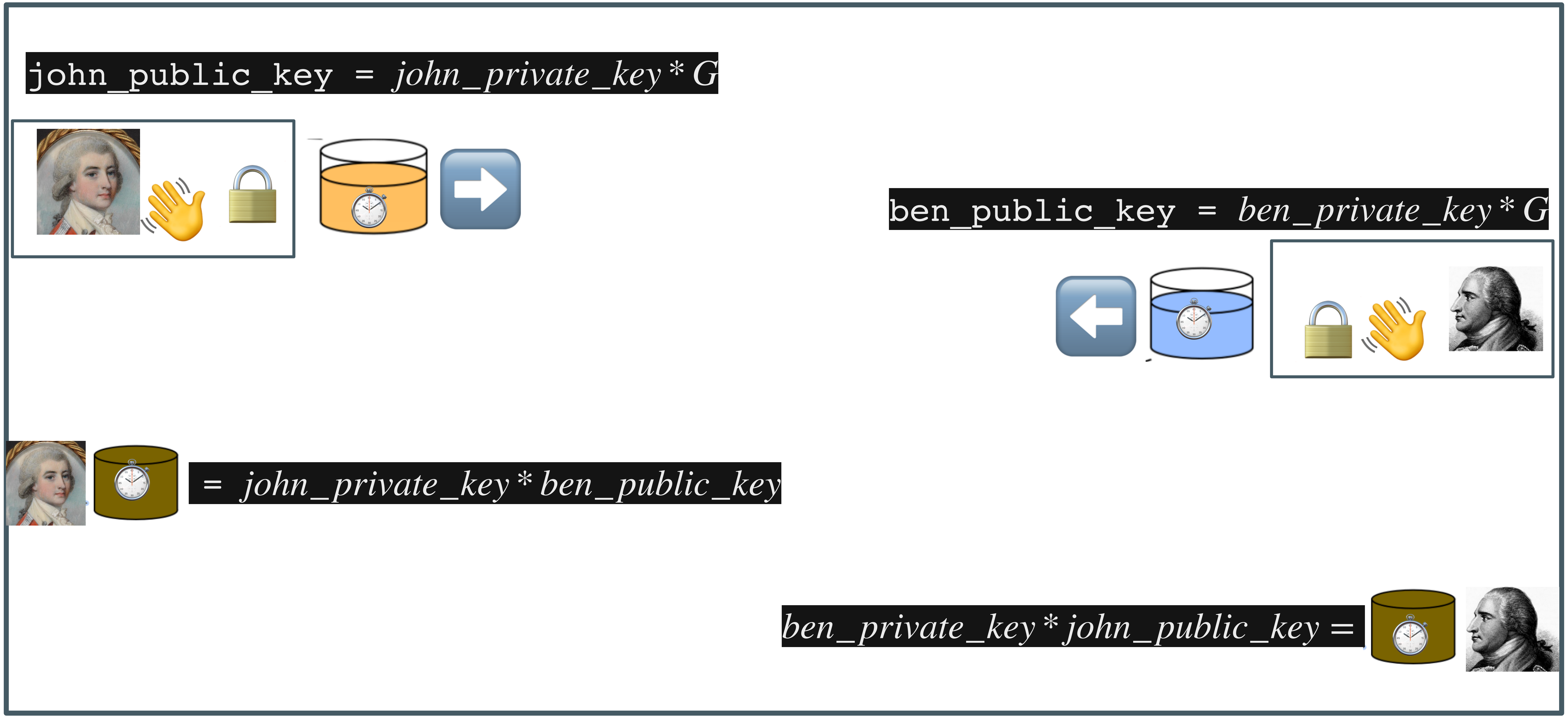
Server key share



```
⌄ Extension: key_share (len=36)
    Type: key_share (51)
    Length: 36
  ⌄ Key Share extension
    ⌄ Key Share Entry: Group: x25519, Key Exchange length: 32
        Group: x25519 (29)⭐⭐⭐⭐⭐
        Key Exchange Length: 32
        Key Exchange: b0722a7c3e3f1b38eea1d5bbb4b2e5ee7471f0aad11f7bdfd006a6e6b9f5eb5c
```

🔒 Server's public key (ECDHE)

ECDHE Key exchange



john_public_key = $john\_private\_key * G$

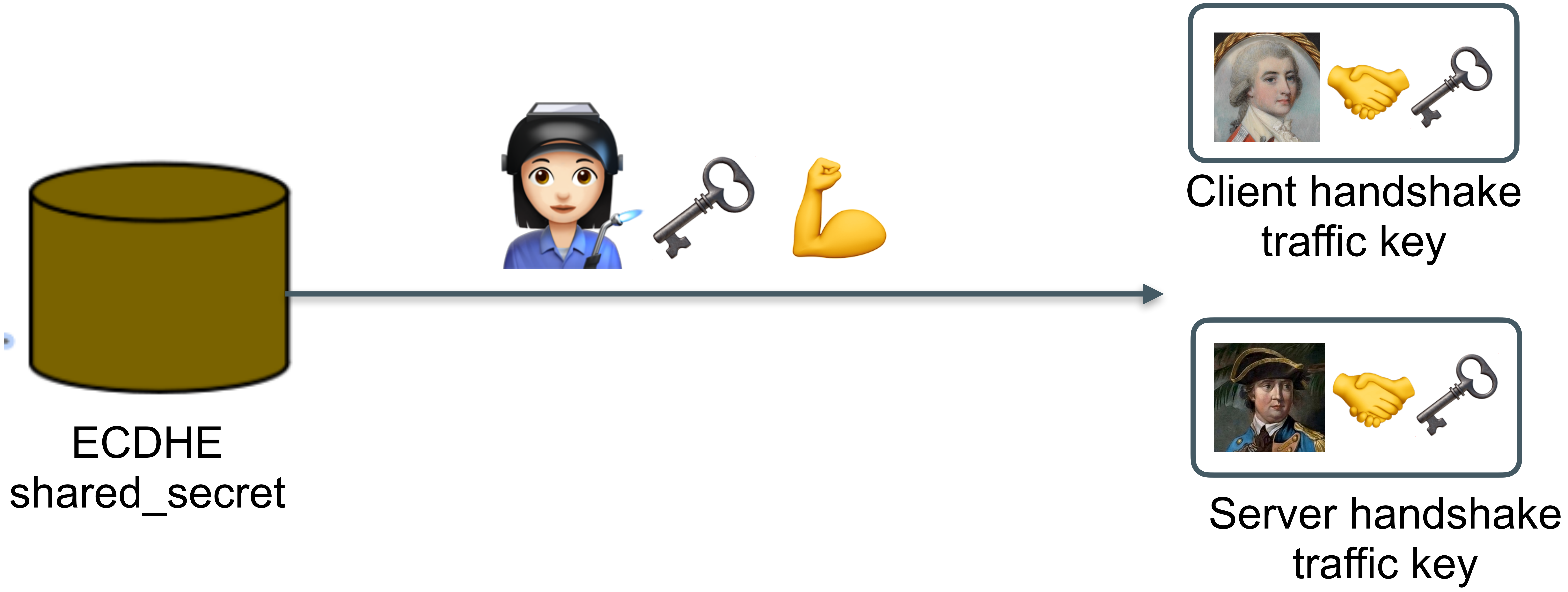
ben_public_key = $ben\_private\_key * G$

= $john\_private\_key * ben\_public\_key$

$ben\_private\_key * john\_public\_key =$

TLS 1.3 🤝

🤝 🗝️ Handshake keys derivation



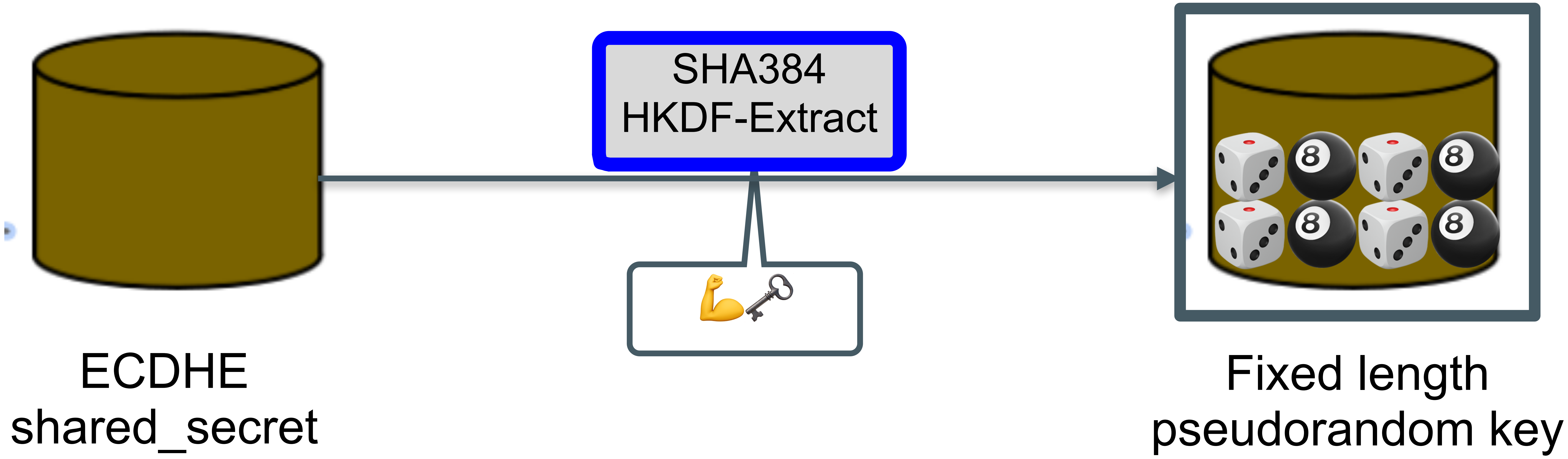
ECDHE
shared_secret

Client handshake
traffic key

Server handshake
traffic key

TLS 1.3 🤝

🤝 🗝️ derivation: HKDF-Extract



SHA384
HKDF-Extract

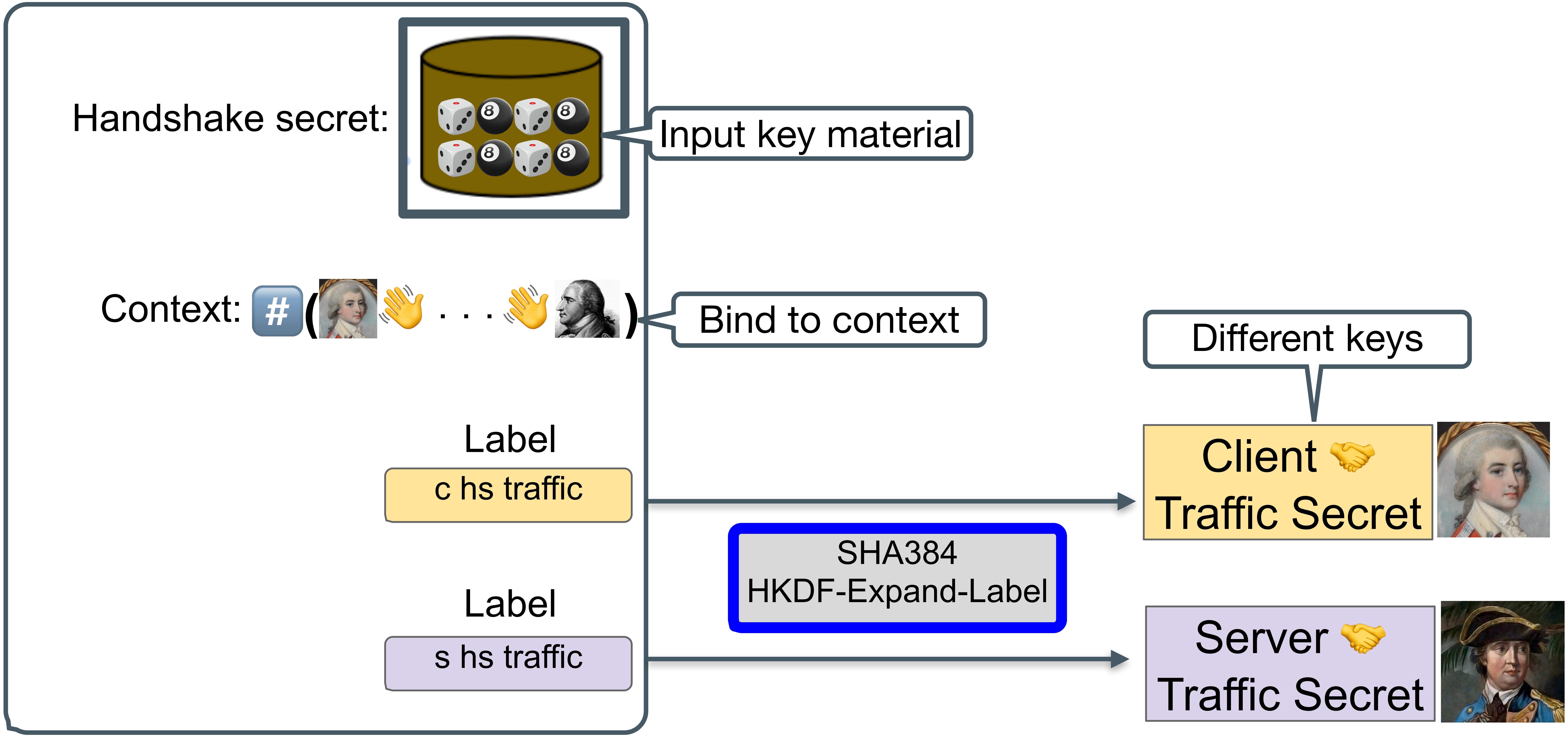
💪🗝️

ECDHE
shared_secret

Fixed length
pseudorandom key

TLS 1.3 🤝

🤝 🗝️ derivation: HKDF-Expand-Label



Handshake secret:

Input key material

Context: #️⃣(👋 . . . 👋)

Bind to context

Label

c hs traffic

Label

s hs traffic

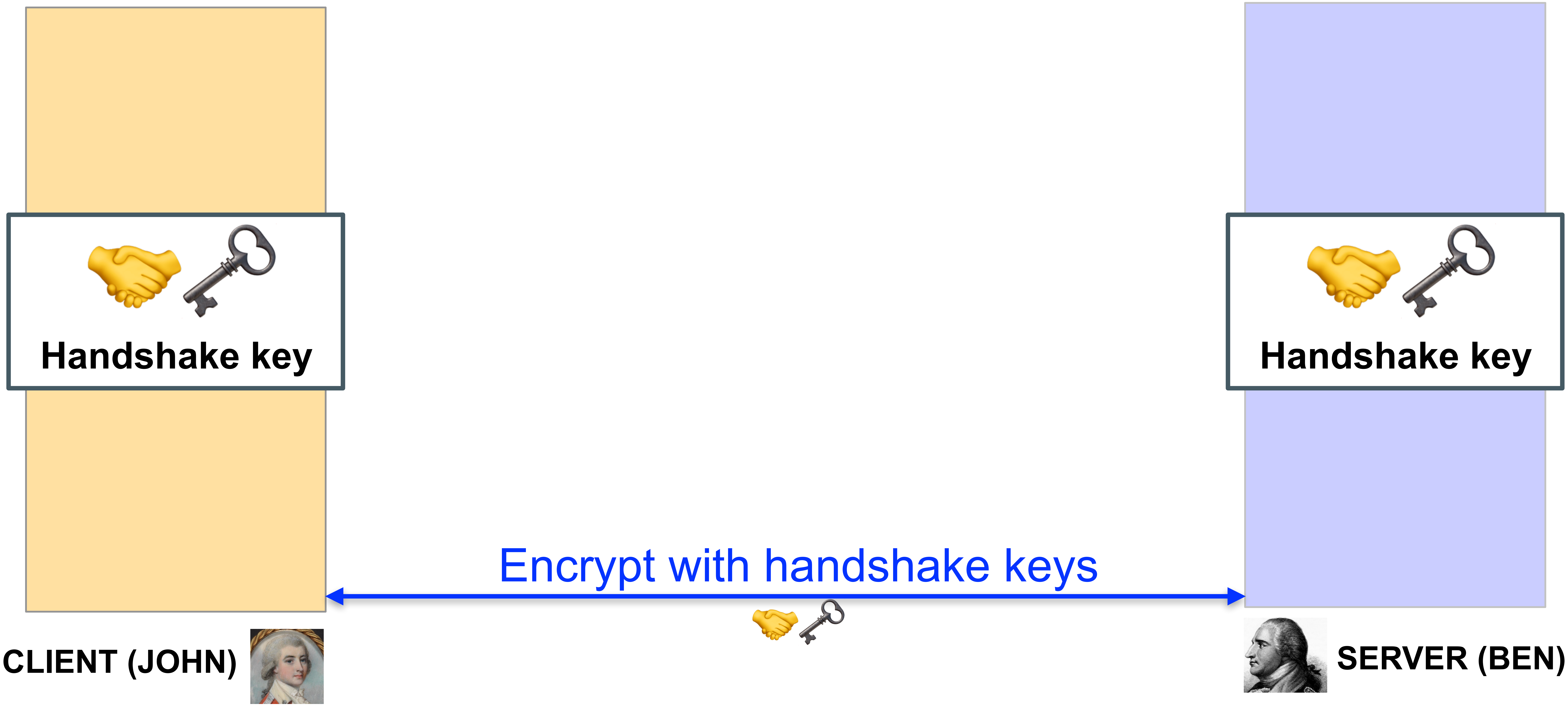
SHA384
HKDF-Expand-Label

Different keys

Client 🤝
Traffic Secret

Server 🤝
Traffic Secret

TLS 1.3 🤝



🤝🗝️ Handshake key

🤝🗝️ Handshake key

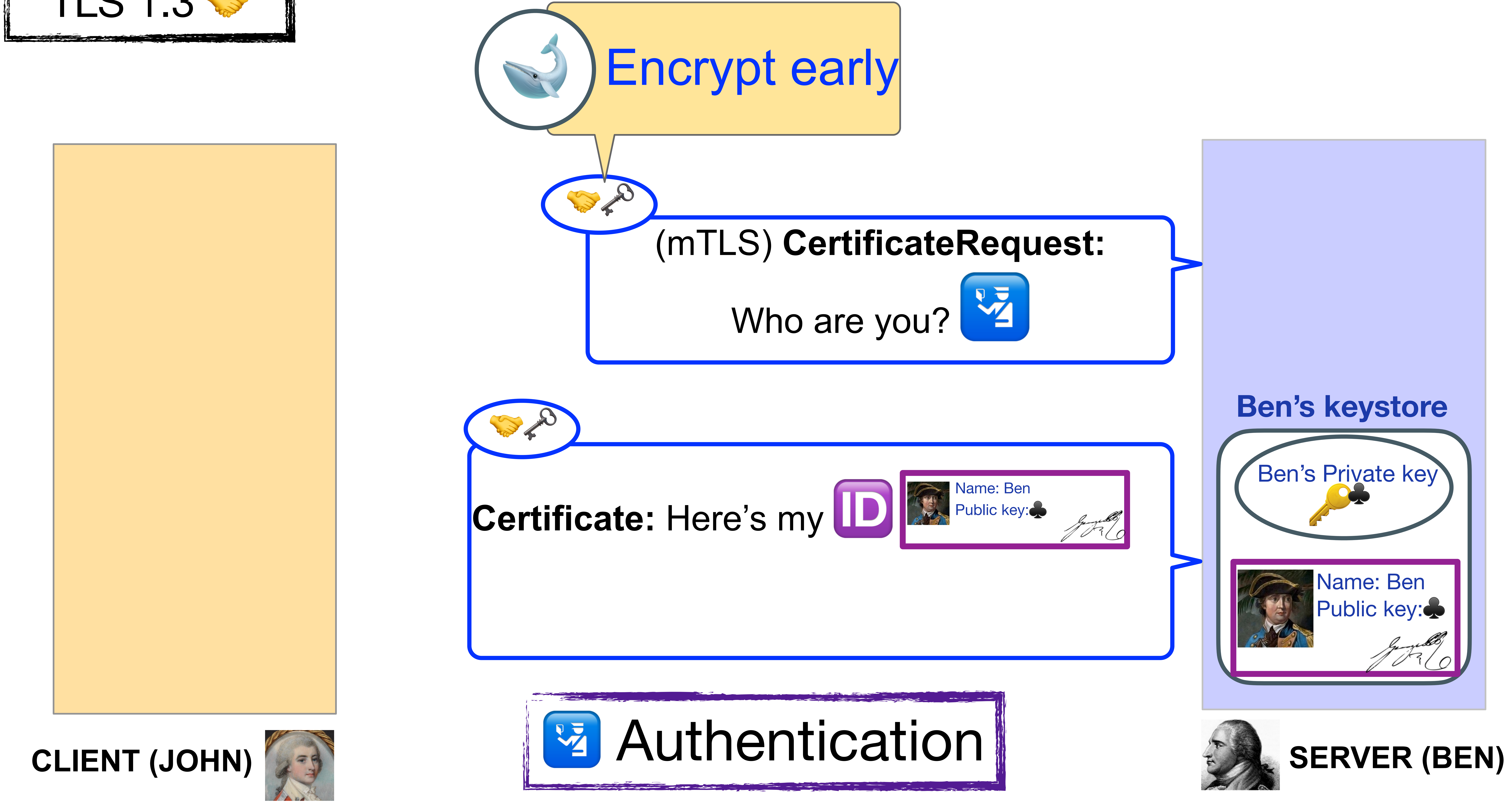
Encrypt with handshake keys

🤝🗝️

CLIENT (JOHN)

SERVER (BEN)

TLS 1.3 🤝



🐳 Encrypt early

🤝🗝

(mTLS) **CertificateRequest:**

Who are you? 🛂

🤝🗝

Certificate: Here's my 🆔

Name: Ben
Public key:♣

Ben's keystore

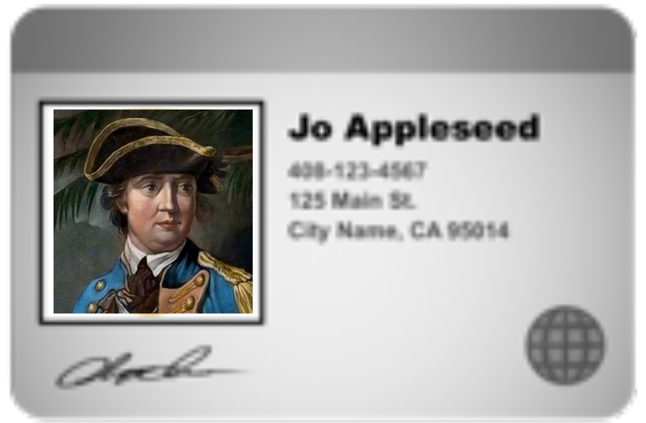
Ben's Private key 🔑♣

Name: Ben
Public key:♣

🛂 Authentication

CLIENT (JOHN)

SERVER (BEN)



Server Certificate

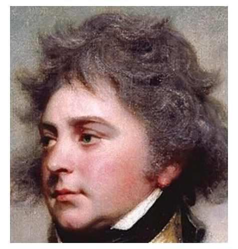
```
⌄ Certificate: 3082072e30820616a00302010202101524b1355353544a35ba9821f898655f300d06092a… (id-at-commonName=www.foo.com,id-at-serialNumber=691011,id-at-businessCatego…
  ⌄ signedCertificate
      version: v3 (2)
      serialNumber: 0x1524b1355353544a35ba9821f898655f
    › signature (sha256WithRSAEncryption)
    ⌄ issuer: rdnSequence (0)
      › rdnSequence: 5 items (id-at-commonName=Entrust Certification Authority - L1M,id-at-organizationalUnitName=(c) 2014 Entrust, Inc. - for authorized use only,id-…
    › validity
    › subject: rdnSequence (0)
    › subjectPublicKeyInfo
    › extensions: 10 items
```

Issued by Intermediate CA

Prince of Wales (later George IV) by Sir William Beechey R.A. (CC0 1.0)

Certificate Chain

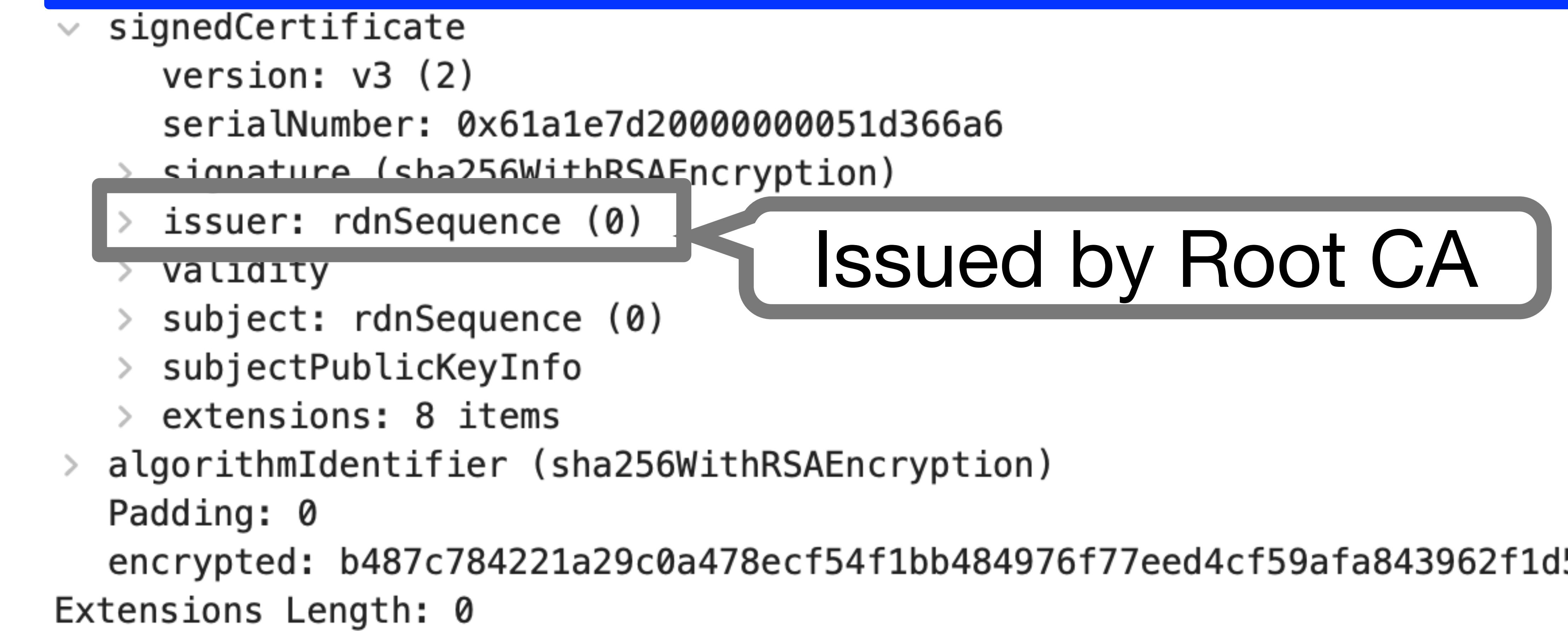
Intermediate CA cert



Name: Prince

Public key:🔴

Issuer: the King



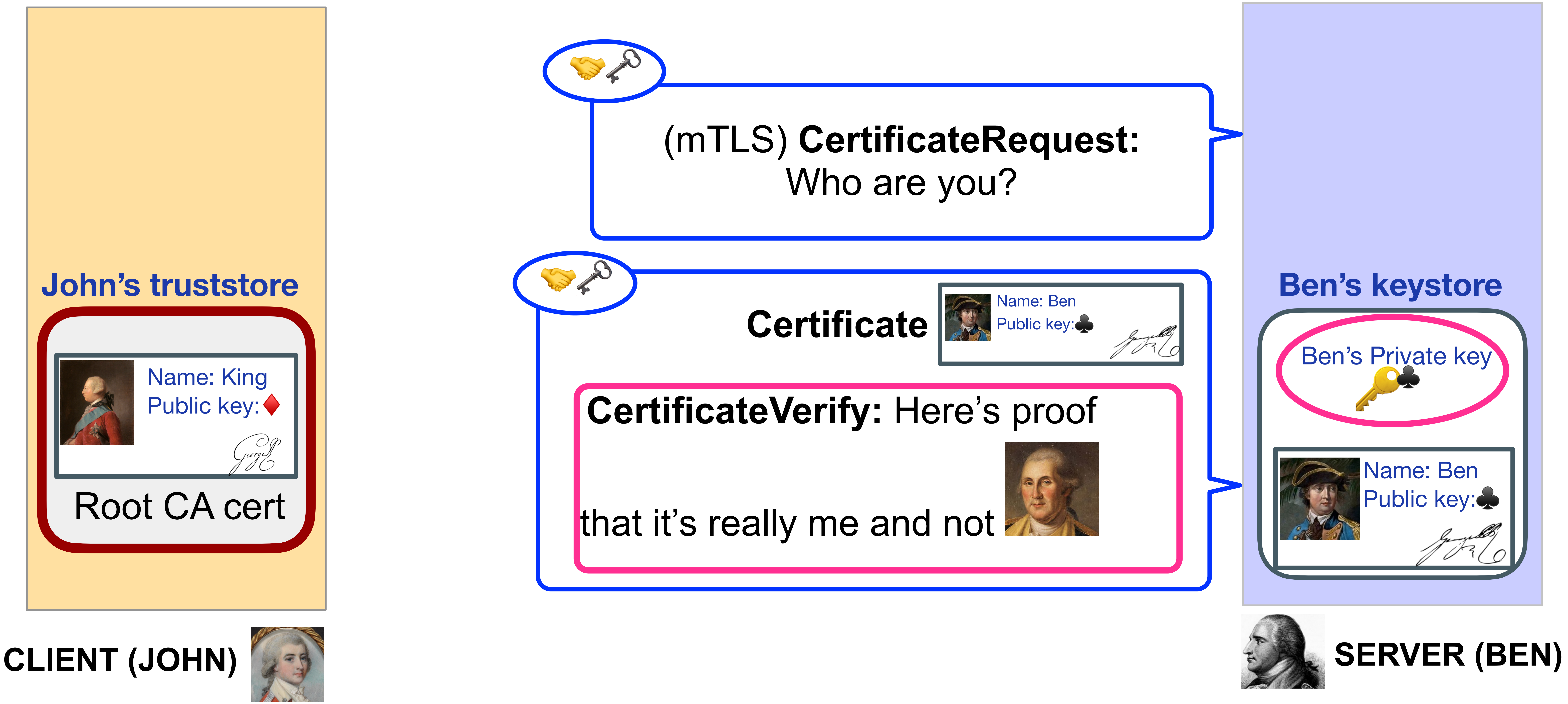
```
Certificate: 3082052d30820415a003020102020c61a1e7d20000000051d366a6300d06092a864886f7… (id-at-commonName=Entrust Certification Authority - L1M,id-at-organizationalUn…
  signedCertificate
    version: v3 (2)
    serialNumber: 0x61a1e7d20000000051d366a6
  > signature (sha256WithRSAEncryption)
  > issuer: rdnSequence (0)
  > validity
  > subject: rdnSequence (0)
  > subjectPublicKeyInfo
  > extensions: 8 items
> algorithmIdentifier (sha256WithRSAEncryption)
  Padding: 0
  encrypted: b487c784221a29c0a478ecf54f1bb484976f77eed4cf59afa843962f1d58dea6f3155b2e…
Extensions Length: 0
```

Issued by Root CA

Prince of Wales (later George IV) by Sir William Beechey R.A. (CC0 1.0)

TLS 1.3 🤝

🛂 Authentication



🤝🔑

(mTLS) **CertificateRequest:** Who are you?

🤝🔑

Certificate

Name: Ben
Public key:♣

CertificateVerify: Here's proof that it's really me and not

John's truststore

Name: King
Public key:♦

Root CA cert

Ben's keystore

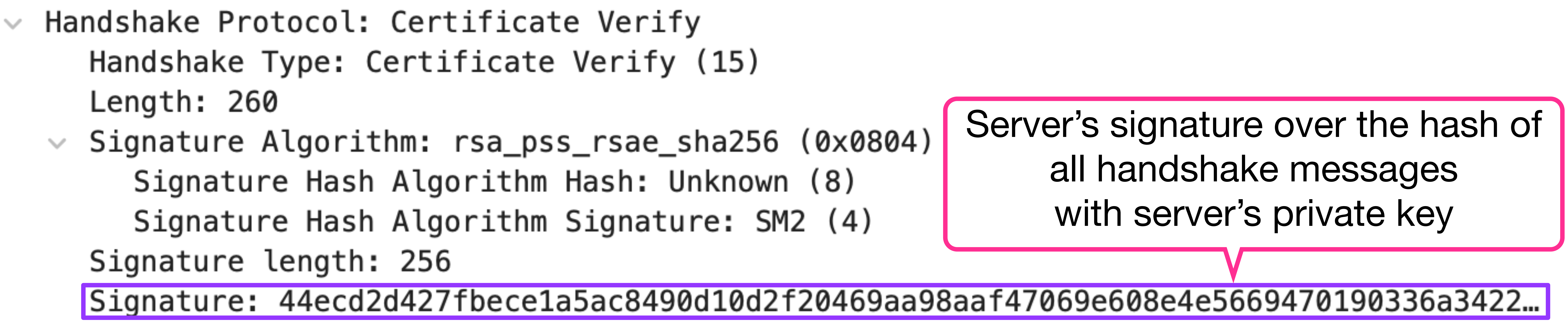
Ben's Private key 🔑♣

Name: Ben
Public key:♣

CLIENT (JOHN)

SERVER (BEN)

🛂 Authentication: CertificateVerify



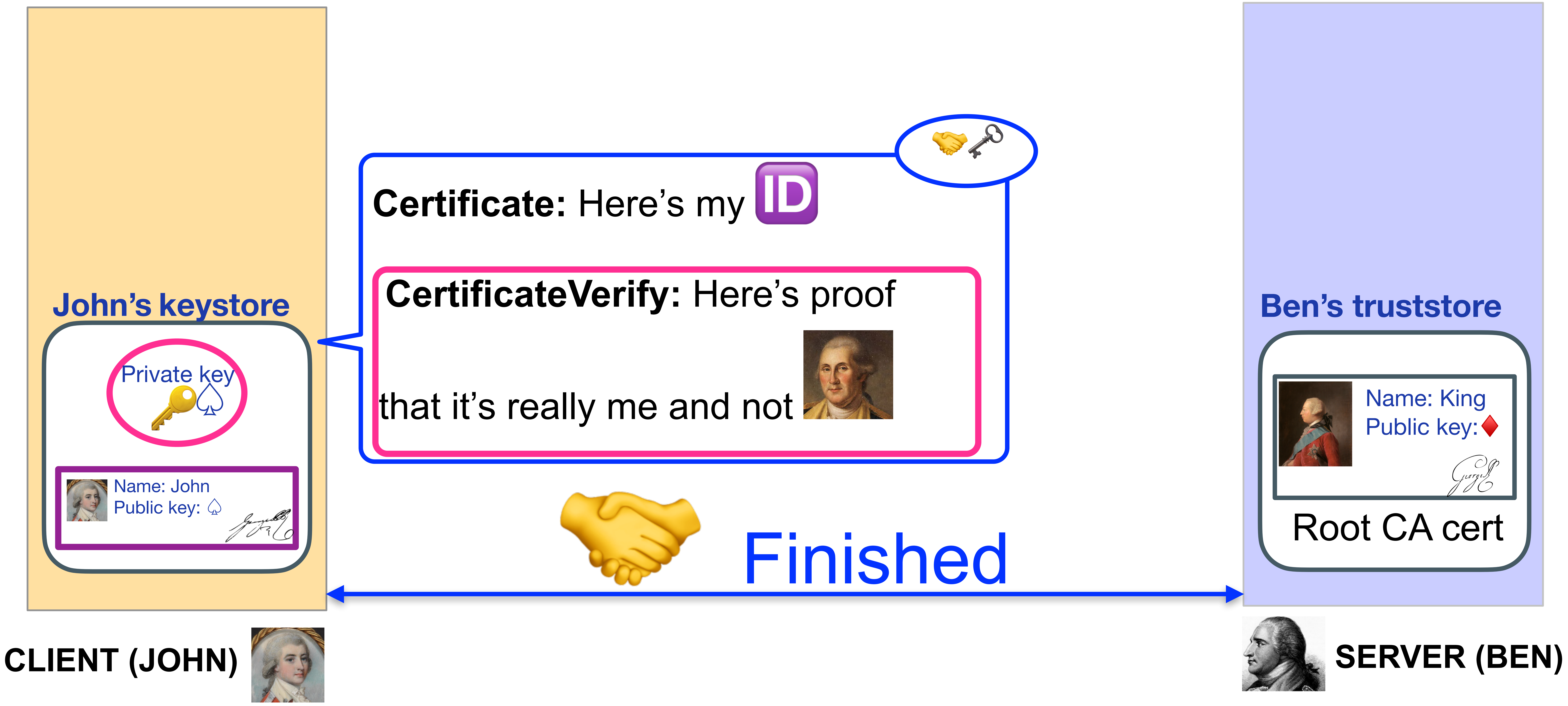
```
⌄ Handshake Protocol: Certificate Verify
    Handshake Type: Certificate Verify (15)
    Length: 260
  ⌄ Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
      Signature Hash Algorithm Hash: Unknown (8)
      Signature Hash Algorithm Signature: SM2 (4)
    Signature length: 256
    Signature: 44ecd2d427fbece1a5ac8490d10d2f20469aa98aaf47069e608e4e5669470190336a3422…
```

Server’s signature over the hash of all handshake messages with server’s private key

Benedict Arnold signature (CC0 1.0)

TLS 1.3 🤝

🛂 Mutual Authentication



🤝🗝️

Certificate: Here's my 🆔

CertificateVerify: Here's proof that it's really me and not

John's keystore

Private key 🔑♤

Name: John
Public key: ♤

Ben's truststore

Name: King
Public key: ♦

Root CA cert

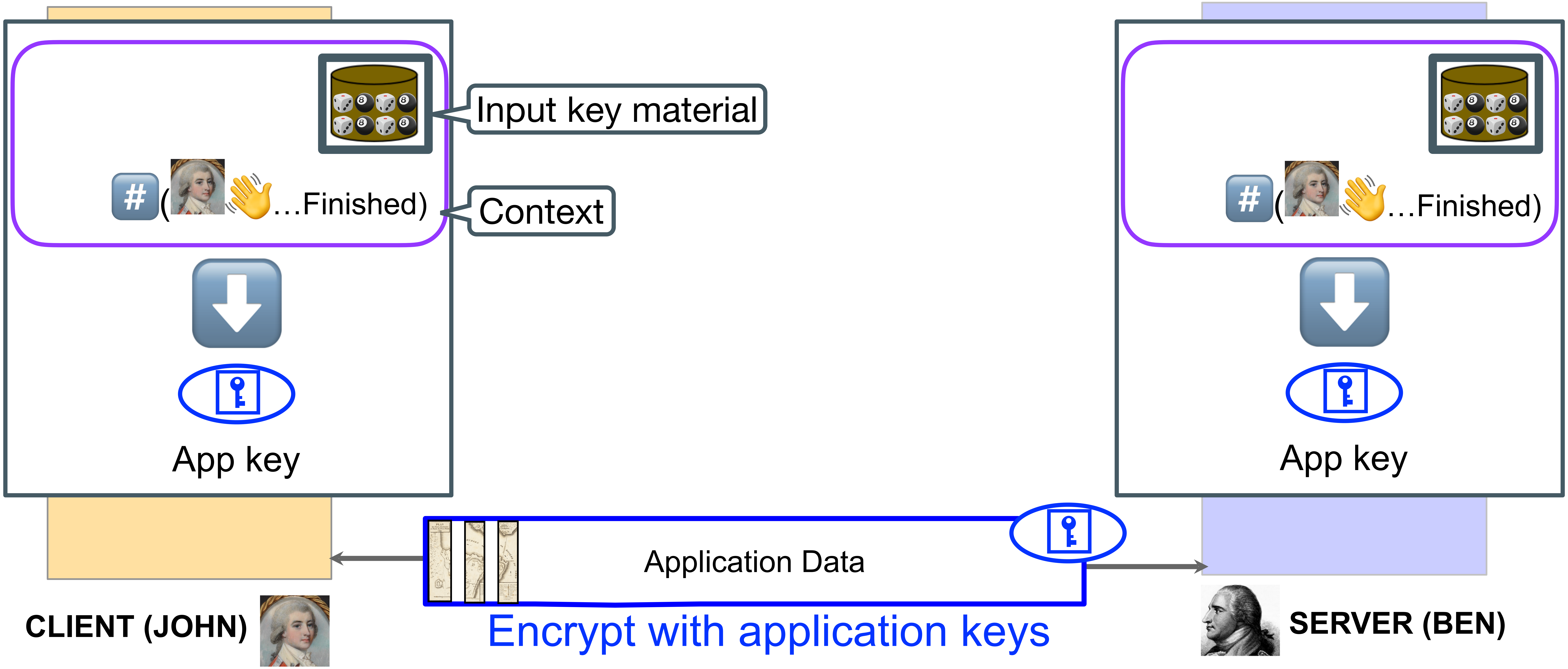
🤝 Finished

CLIENT (JOHN)

SERVER (BEN)

TLS 1.3

Application Traffic



Input key material

Context

#(…Finished)

App key

#(…Finished)

App key

Application Data

Encrypt with application keys

CLIENT (JOHN)

SERVER (BEN)

Thank You

Lerna Ekmekcioglu
🐳 https://www.linkedin.com/in/lerna

Humpback whale image by HANSUAN FABREGAS from Pixabay

References

Content:

- The Illustrated TLS 1.3 Connection
- A Readable Specification of TLS 1.3 by David Wong
- How does Transport Layer Security work? from Real-World Cryptography by David Wong
- Implementing a toy version of TLS 1.3 by Julia Evans
- Practical Cryptography for Developers by Svetlin Nakov, PhD
- A Detailed Look at RFC 8446 (a.k.a. TLS 1.3) by Nick Sullivan
- RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3
- Why Benedict Arnold Turned Traitor Against the American Revolution, Smithsonian Magazine, May 2016

Images:

- Hieroglyphs (CC0 1.0)
- Clay tablets (CC0 1.0)
- DeLorean Time Machine by JMortonPhoto.com & OtoGodfrey.com (CC BY-SA 4.0)
- British old infantry uniforms (CC0 1.0)
- Continental infantry uniform (CC0 1.0)
- Major Andre by George Engleheart, Yale Center for British Art (CC0 1.0)
- Peggy Shippen by Daniel Gardner (CC0 1.0)
- General Benedict Arnold by Thomas Hart (CC0 1.0)
- "George Washington after the Battle of Princeton (Charles Willson Peale), PP218," Princeton University Art Museums collections online, https://artmuseum.princeton.edu/collections/objects/45234
- West Point by Pierre Didot, Boston Public Library Digital Map Collection (CC0 1.0)
- Caesar cipher with a shift of 3, CC0 1.0
- Combination lock (CC0 1.0)
- Diffie-Hellman Key Exchange Analogy (CC0 1.0)
- Portrait of George III by Allan Ramsey, CC0 1.0
- King George III signature (CC0 1.0)
- Benedict Arnold by H.B. Hall, National Archives and Records Administration, CC0 1.0
- Prince of Wales (later George IV) by Sir William Beechey R.A. (CC0 1.0)
- Benedict Arnold signature (CC0 1.0)
- Image by Bianca Van Dijk from Pixabay
- Humpback whale image by HANSUAN FABREGAS from Pixabay