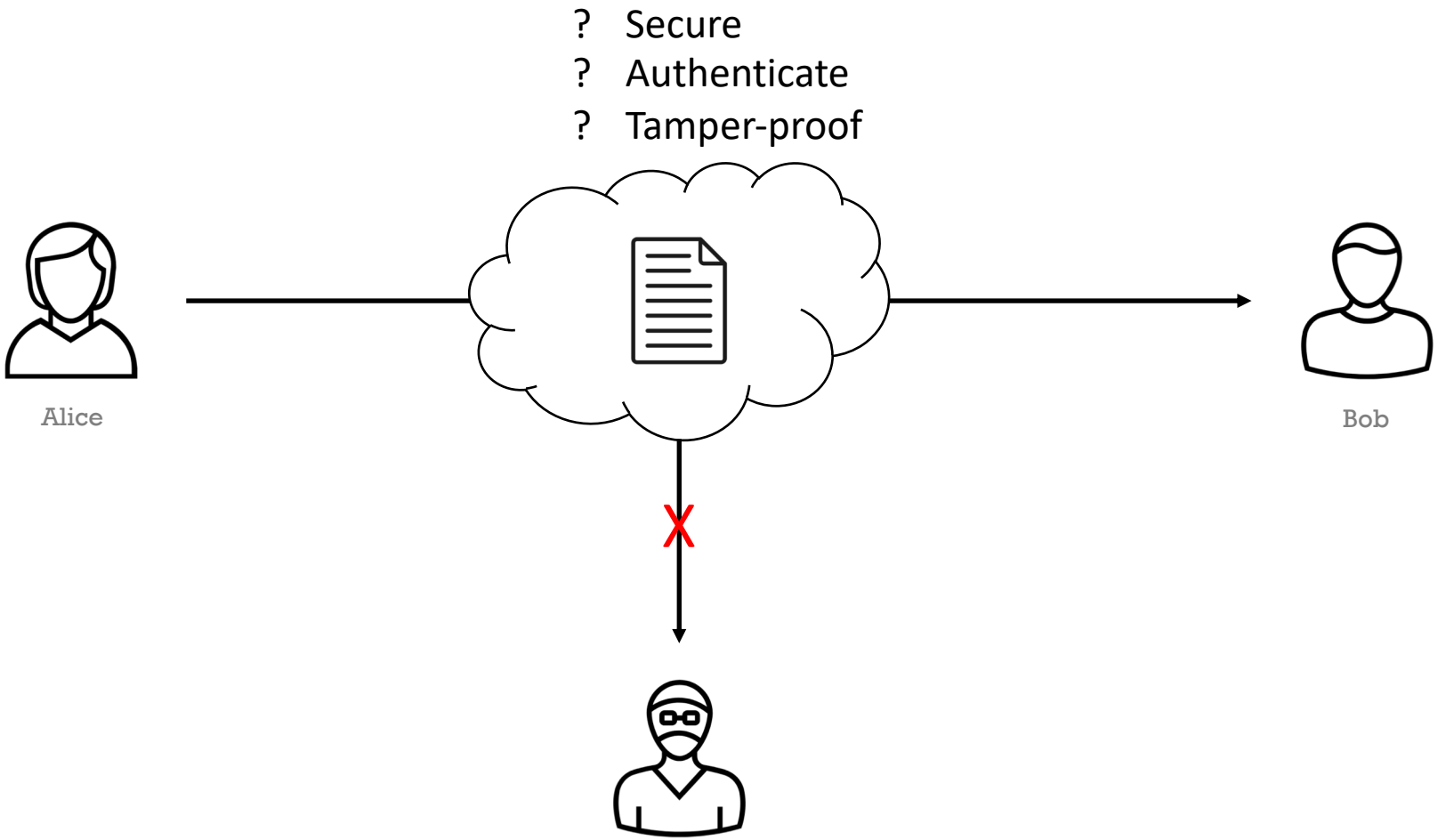
Crypto-101

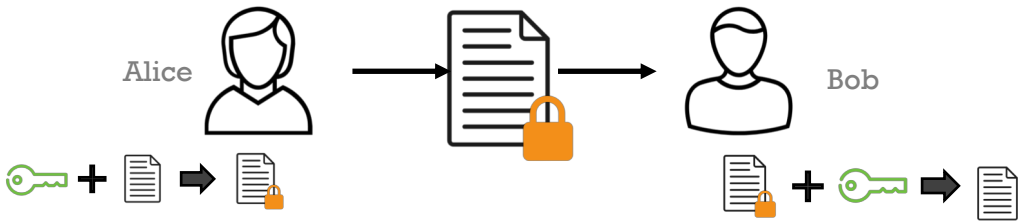
and some more…

Why?



Encryption

Symmetric-key encryption

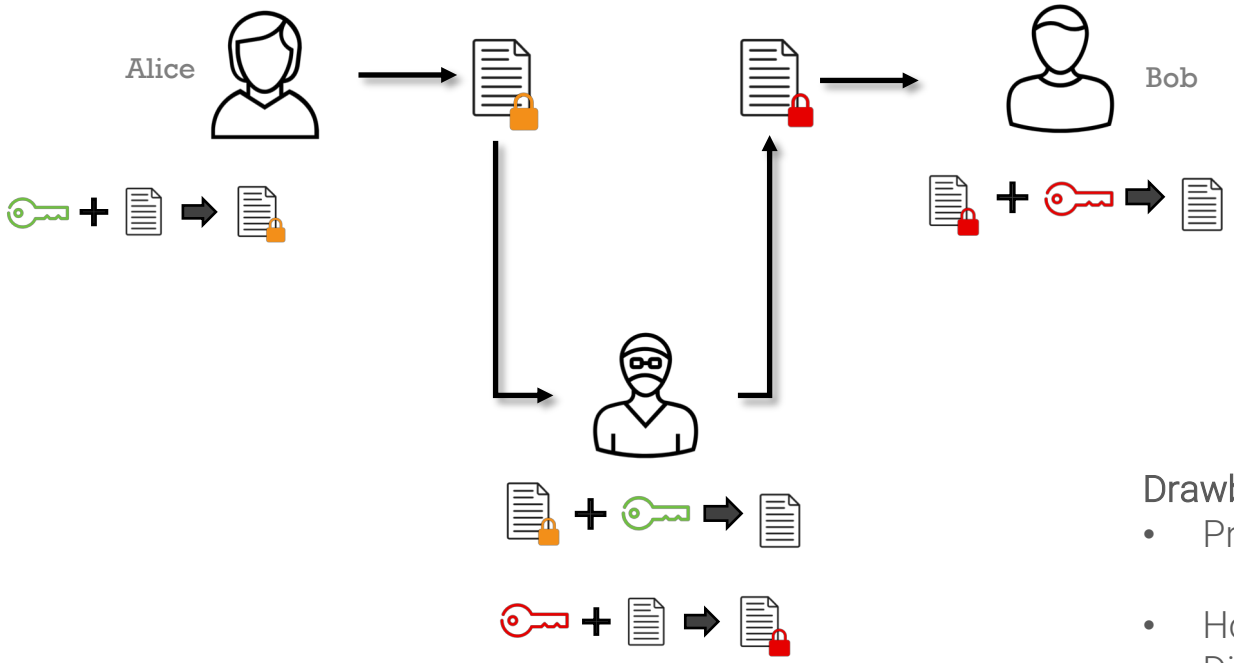


Same cryptographic key used for encryption and decryption

e.g. Caesar cipher

Encryption

Symmetric-key encryption

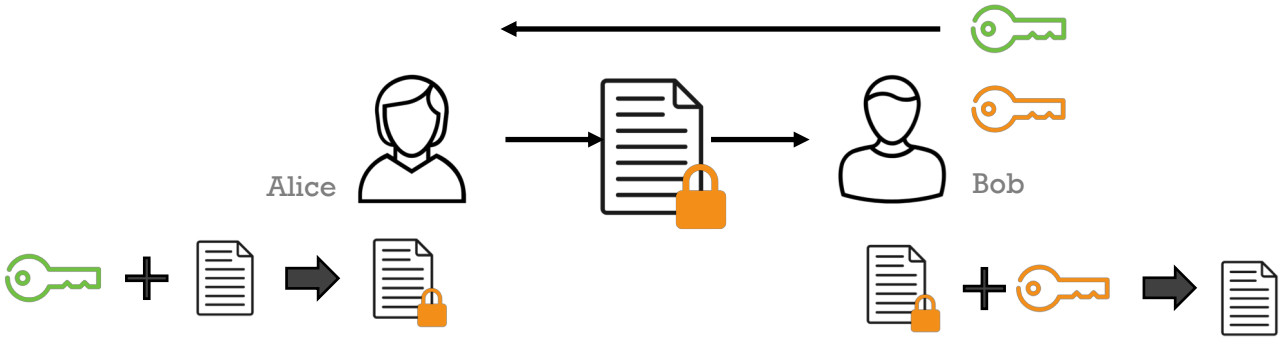


Drawbacks

- Prior knowledge of keys is required.
  - How to transmit the keys?
- How to authenticate the sender
- Difficult to change the keys
- Vulnerable to brute-force attack

Encryption

Asymmetric-key or Public-key encryption



Cryptographic key-pair is used for encryption

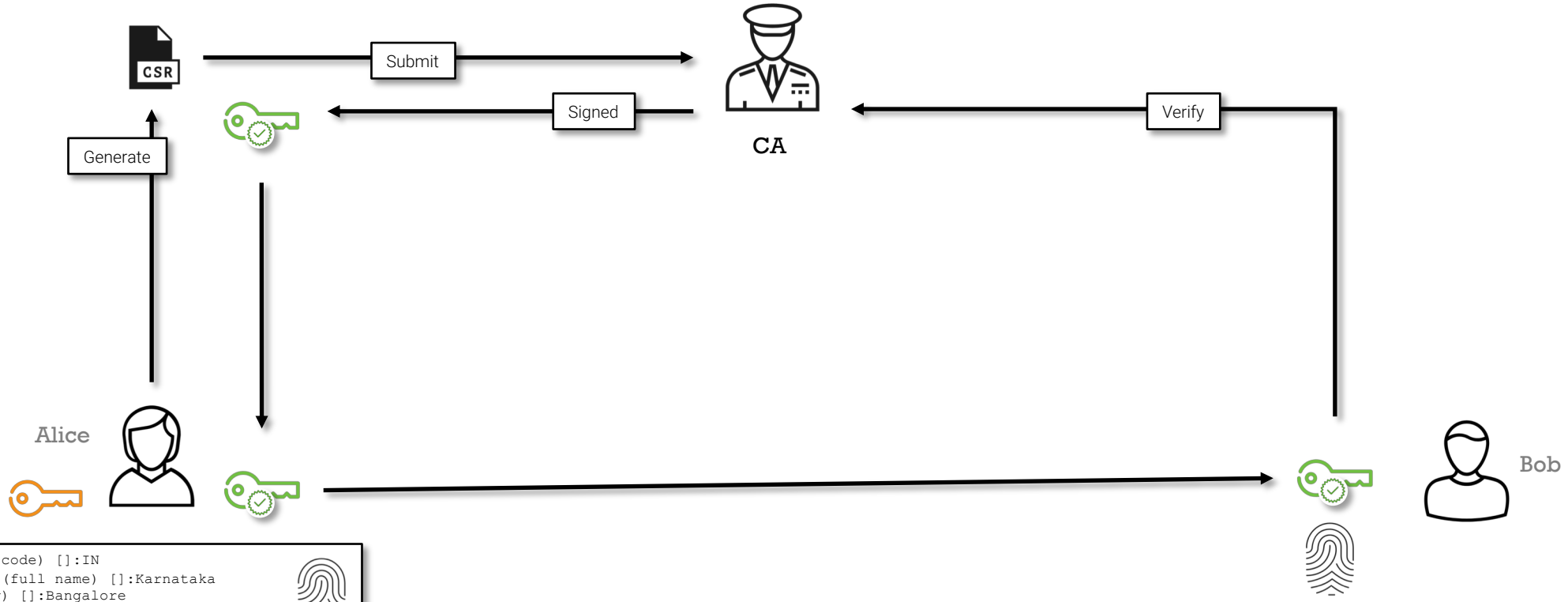
- **Public-key** or **Certificate** is shared widely
- **Private-key** or **Key** kept safe

Drawbacks

- Computation is intensive
- How can you be sure that Bob's certificate is really Bob's certificate?

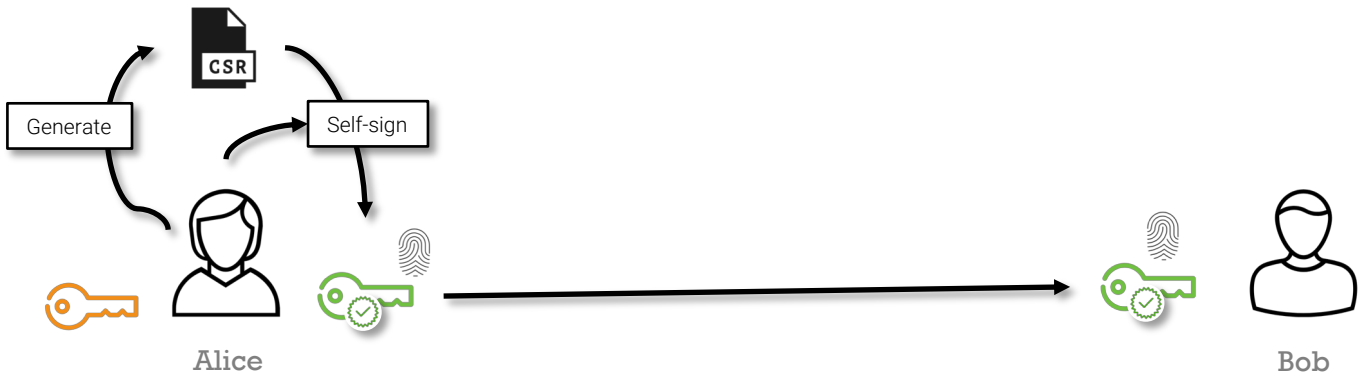
Certificate Authorities (CA)

Solving the Identity Crisis

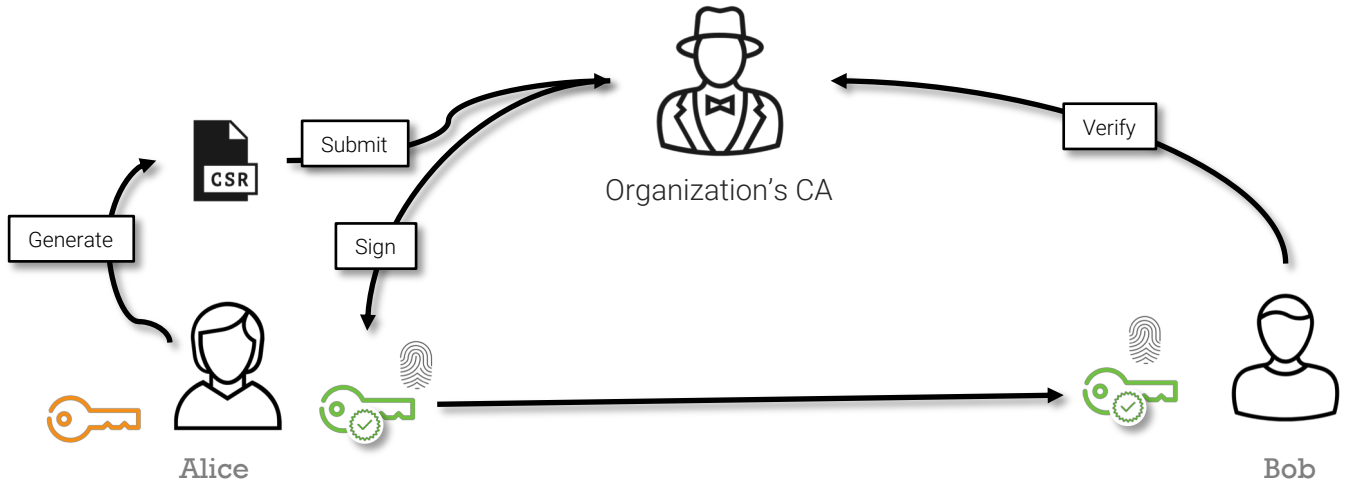



```
Country Name (2 letter code) []:IN
State or Province Name (full name) []:Karnataka
Locality Name (eg, city) []:Bangalore
Organization Name (eg, company) []:CAPIOT
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, fully qualified host name) []:odp.capiot.com
Email Address []:it@capiot.com
```

Self-signed Certs. / Org. Level CAs



Trust the certificate at your own risk



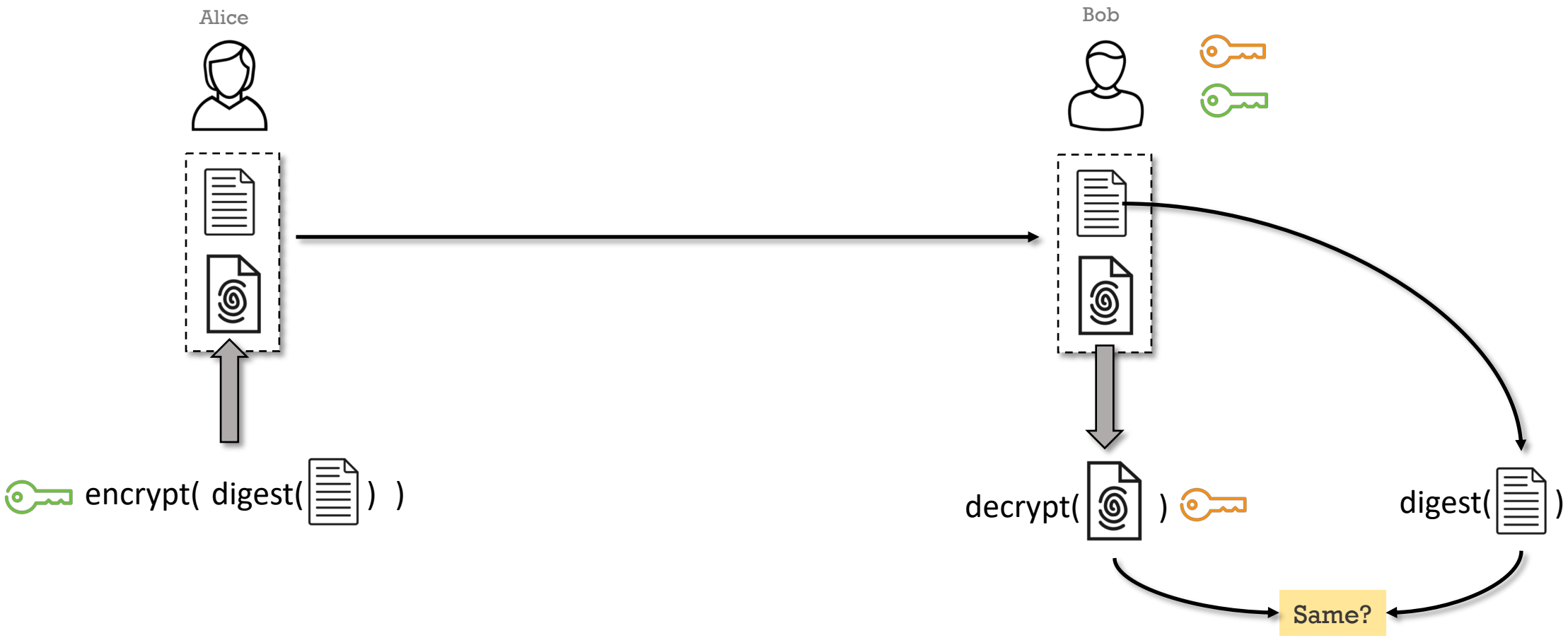
Everyone trusts the Organization's CA

Digital Signatures

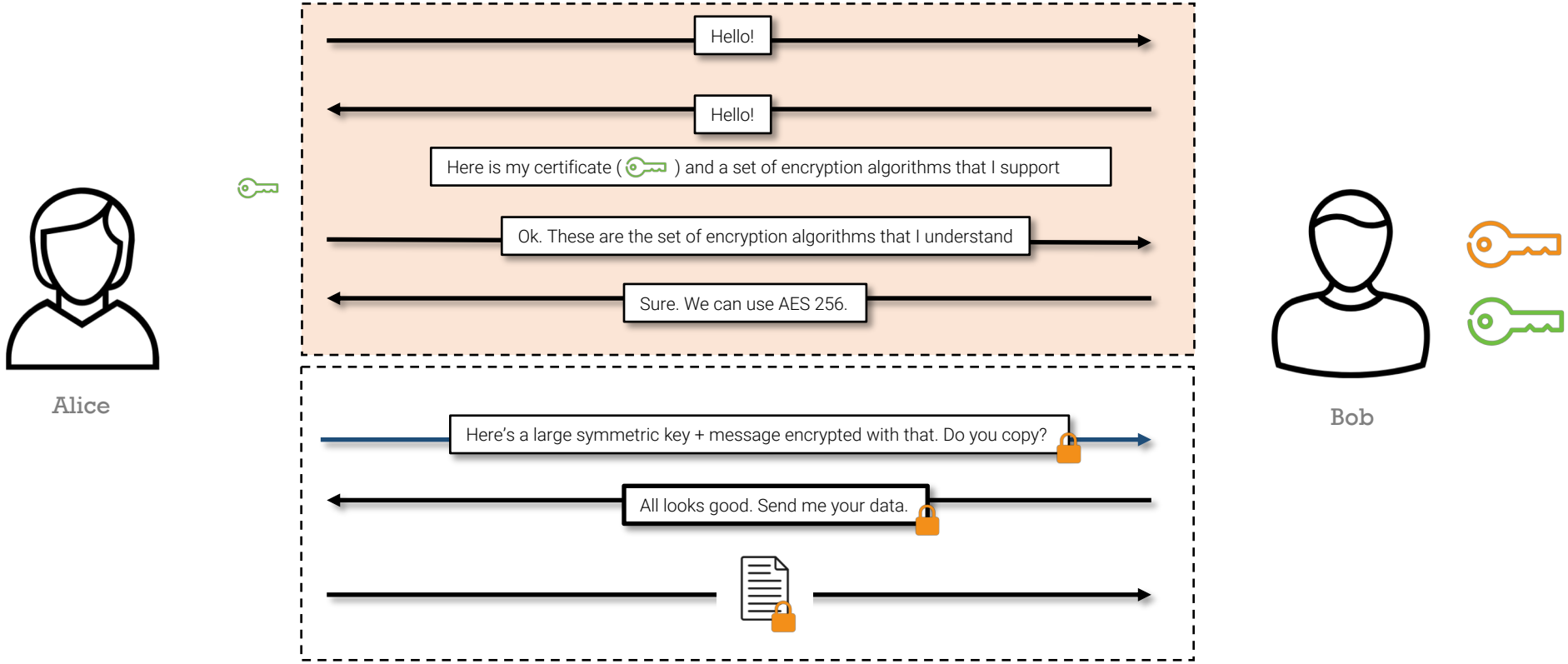
Sending tamper-proof messages

Digital Signatures

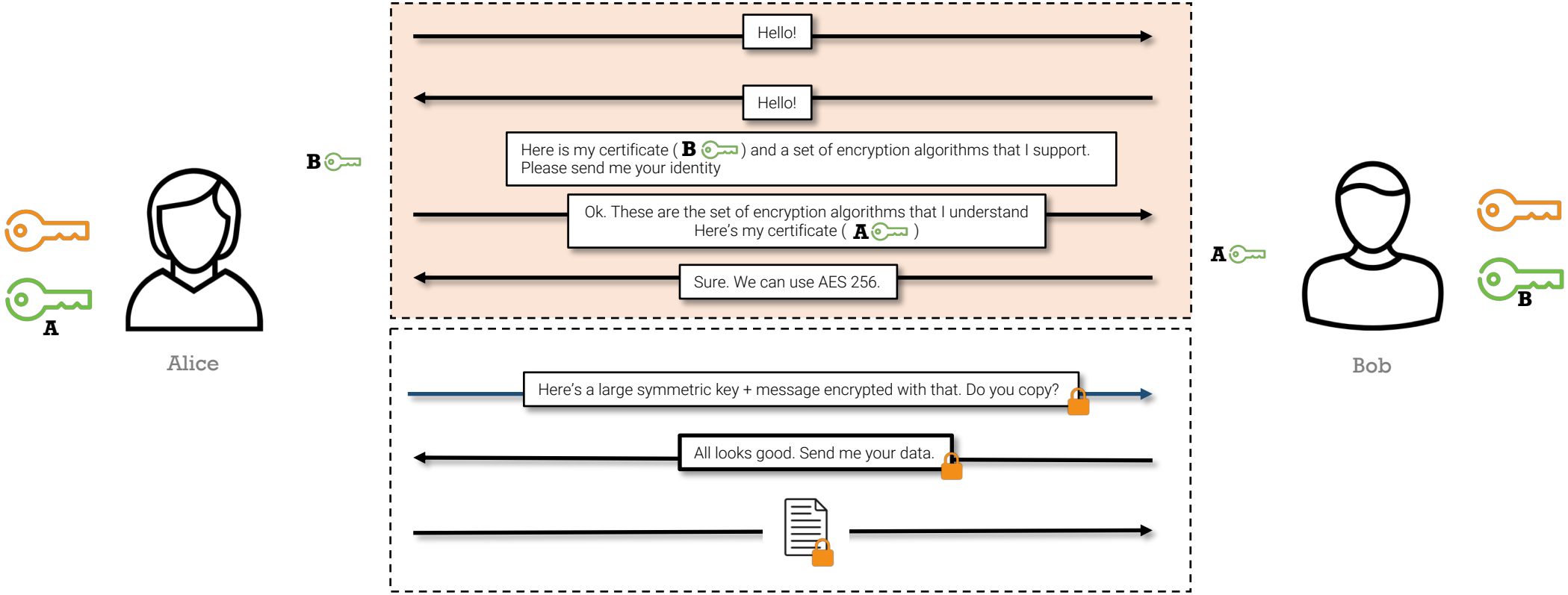
Sending tamper-proof messages



Secure-socket Layer (SSL)



2-way SSL


Hello!
Hello!
Here is my certificate (B) and a set of encryption algorithms that I support.
Please send me your identity
Ok. These are the set of encryption algorithms that I understand
Here's my certificate (A)
Sure. We can use AES 256.
Here's a large symmetric key + message encrypted with that. Do you copy?
All looks good. Send me your data.
A
Alice
B
A
Bob
B

Saving passwords

Never save passwords as clear text in the DB. **Why?**

What's the alternate approach?

Saving passwords

Hashing!

```
md5
sha256
sha512
```

checksum(**** + ) = `6926f9e52fbebdcf1ef16c14fe77d2fa`

Password

Salt
(saved in the db)

Same?

User authentication

checksum(**** + ) = `6926f9e52fbebdcf1ef16c14fe77d2fa`

User entered password

Salt (from db)

Generating a KEY

```
openssl genrsa -out server_2048.key 2048
openssl genrsa -out server_4096.key 4096
openssl genrsa -out server.key 2048
openssl genrsa -out ca.key 2048
```

Generate a CSR from KEY

```
openssl req -out server.csr -key server.key —new
```

Generate CSR and KEY (single command)

```
openssl req -out server.csr -new -newkey rsa:2048 -nodes -keyout server.key
openssl req -out ca.csr -new -newkey rsa:2048 -nodes -keyout ca.key
```

Display information within CSR

```
openssl req -in server.csr -verify -noout -text
openssl req -in ca.csr -noout -text
```

Generate a CERT from CSR and self-sign it with the key

```
openssl x509 -signkey server.key -in server.csr -req -days 365 -out server.crt
openssl x509 -signkey ca.key -in ca.csr -req -days 365 -out ca.crt
```

Generate a CERT from CSR and CA sign it

```
openssl ca -config openssl.cnf -policy signing_policy -extensions signing_req -
out server_ca.crt -infiles server.csr
```

* The *openssl.cnf* file can be found in the repository

Display information within CERT

```
openssl x509 -in server.crt -text -noout
openssl x509 -in server_ca.crt -text -noout
```

Encrypting a file

Generate a random key to encrypt

```
openssl rand -base64 32 > key.bin
```

Encrypting a file

Encrypt the file

```
openssl enc -aes-256-cbc -salt -in README.md -out README.md.enc -pass
file:./key.bin
```

Encrypting a file

Get the public key

```
openssl rsa -in server.key -out server.pub.pem -outform PEM -pubout
```

Encrypting a file

Encrypt the random key with the public keyfile

```
openssl rsautl -encrypt -inkey server.pub.pem -pubin -in key.bin -out
key.bin.enc
```

Decrypting a file

Decrypt the random key with our private key file

```
openssl rsautl -decrypt -inkey server.key -in key.bin.enc -out key.bin.out
```

Decrypting a file

Decrypt the large file with the random key

```
openssl enc -d -aes-256-cbc -in README.md.enc -out README.md.out -pass
file:./key.bin.out
```

Hashing

```
md5sum README.md
shasum -a 256 README.md
```

Commands

https://github.com/jerrymannel/crypto101

Thankyou!