PKI-101

Copyright, license, etc.

- This presentation is intellectual property of Unixerius.
  - Copyright and all rights remain with Unixerius.

- This presentation is offered to Unixerius' customer:
  - **For free**, to use for in-house training.
  - With the stipulation that it remains **unaltered**, as-is.
  - And that it is **not distributed** further.

Intro to PKI, crypto and certificates

aka "*When Alice met Bob*"

Instructional goals

- Understand **cryptography fundamentals**.
- Know why proper **key management** is important.
- Know what **certificates** are and their purpose.
- Understand the **infrastructure** for PKI.
- Have a small set of **best practices**.

Required knowledge

- Basic computer networking,
  - "*How do computers talk to each other?*"
  - "*What is a server? What is a client?*"

- DNS, hostnames, URLs
  - "*How do I get from google.com to 144.89.33.10?*"

Alice and Bob and the CIA

- Alice and Bob have come a long way
  - https://cryptocouple.com

- InfoSec says, things worth protecting:
  - Confidentiality
  - Integrity
  - Availability

Alice and Bob and the CIA

- Cryptography helps us with C and I.
  - Confidentiality by enciphering.
  - Integrity by hashing.

- … But it can hurt A!
  - Losing access to data.

Alice and Bob and the CIA

- Challenge: how to prove identity?
  - How does Alice know she's talking to Bob,
  - And not to some hacker?

- That's the point of PKI:
  - Assuring trust.

What could go wrong?

- If you don't follow guidelines and best practices:
  - Key compromise
  - Bad cryptographic implementations

- Leading to:
  - Data leaks or falsification
  - Impersonation and credential theft

Encryption basics

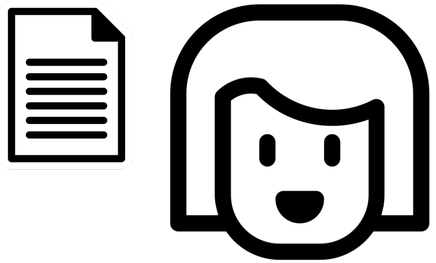
The dry definition

"[…] encryption is the process of ***encoding a message or information*** *in such a way that* ***only authorized parties can access it*** *and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to [interceptors]"*

- Wikipedia

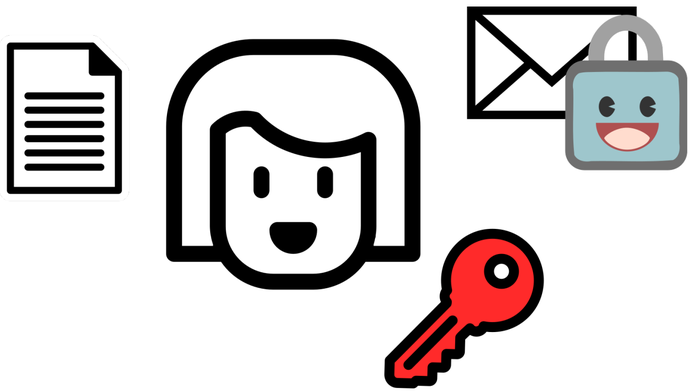
Symmetric key encryption

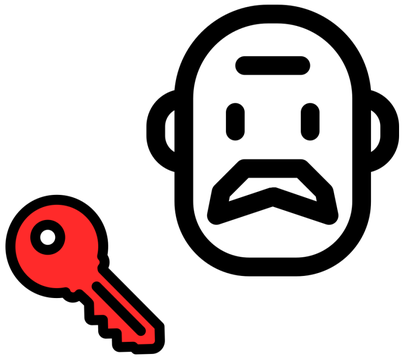
- Both parties use a shared secret.



Symmetric key encryption

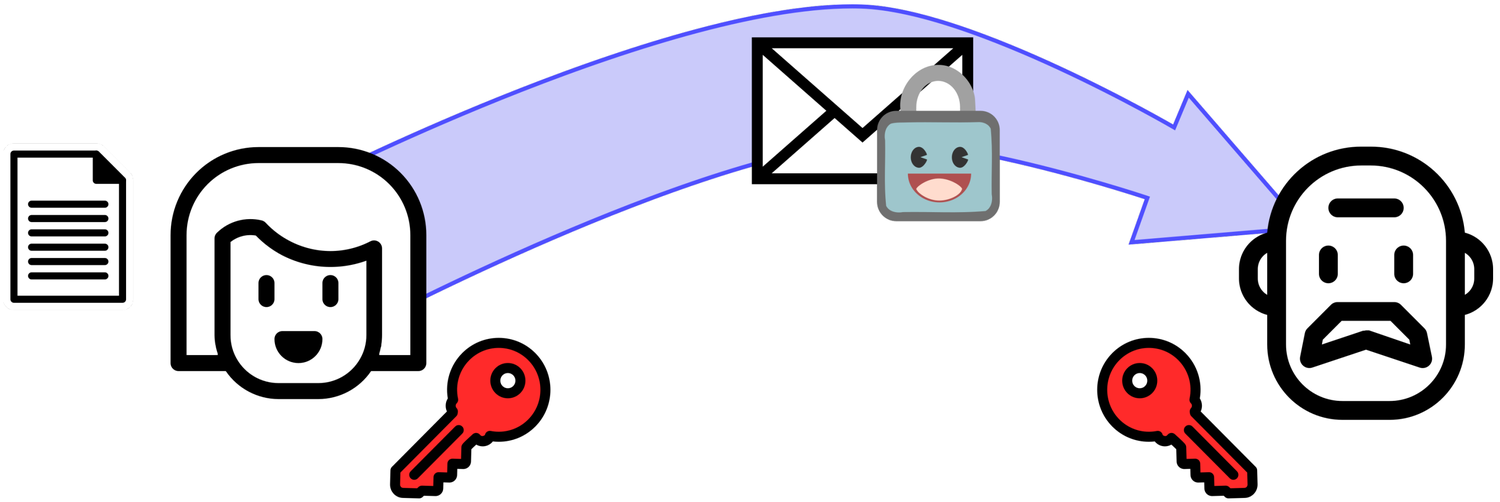
- Both parties use a shared secret.





Symmetric key encryption

- Both parties use a shared secret.



Symmetric key encryption

- Both parties use a shared secret.

- Confidentiality, but how to share the key?

Key exchange

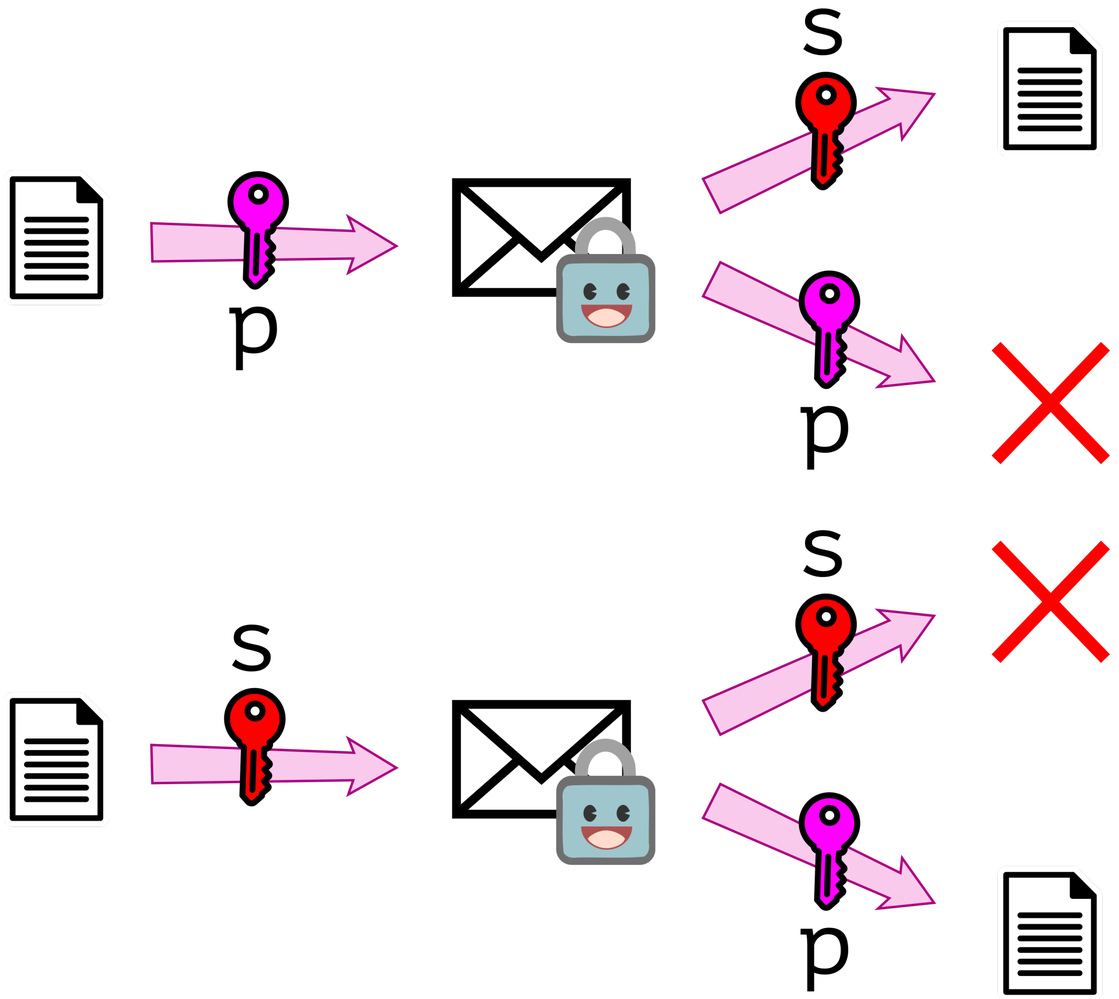
- Symmetric crypto is fast. We want that!
- But how to safely agree on a shared key?
  - Diffie Hellman, open-air mutual agreement
  - RSA / public-key encryption

See also: Venafi - Diffie-Hellman vs RSA

Asymmetric encryption

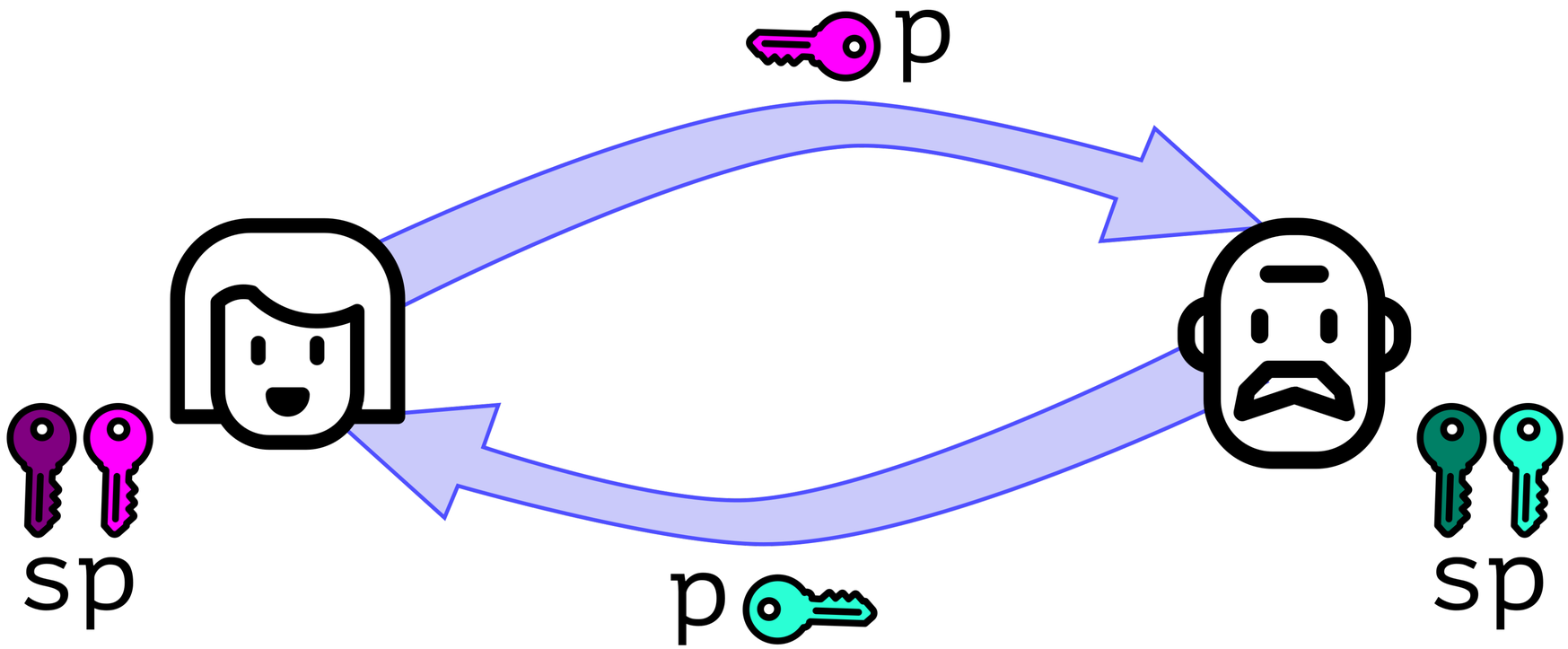
- Both parties have a pair of keys.
- One-way trapdoor math!
- What you encrypt with the one key,
  - Can only be recovered with the other.
  - And vice versa…

Asymmetric encryption



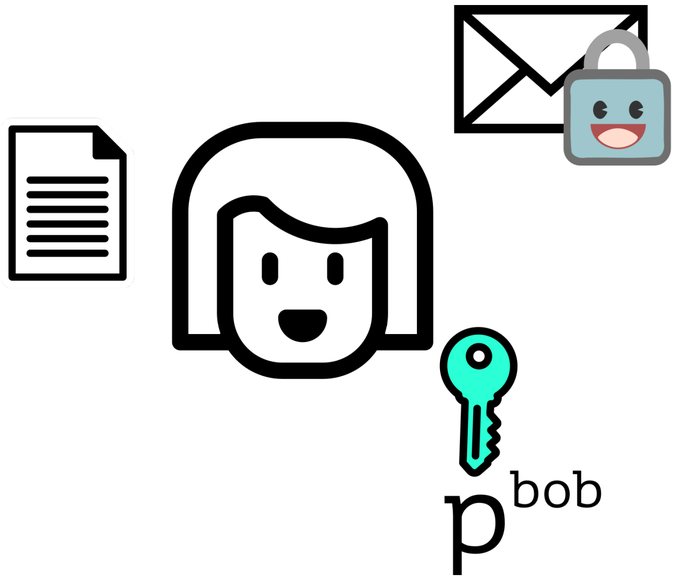
Asymmetric encryption

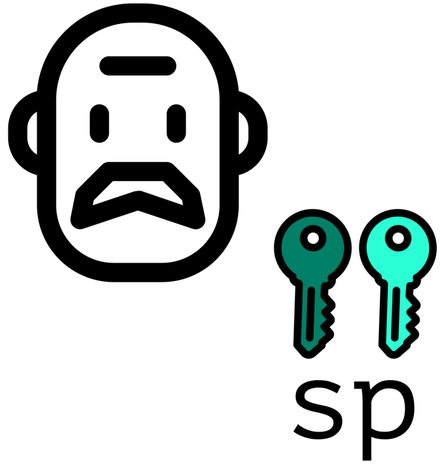
- Exchange public keys



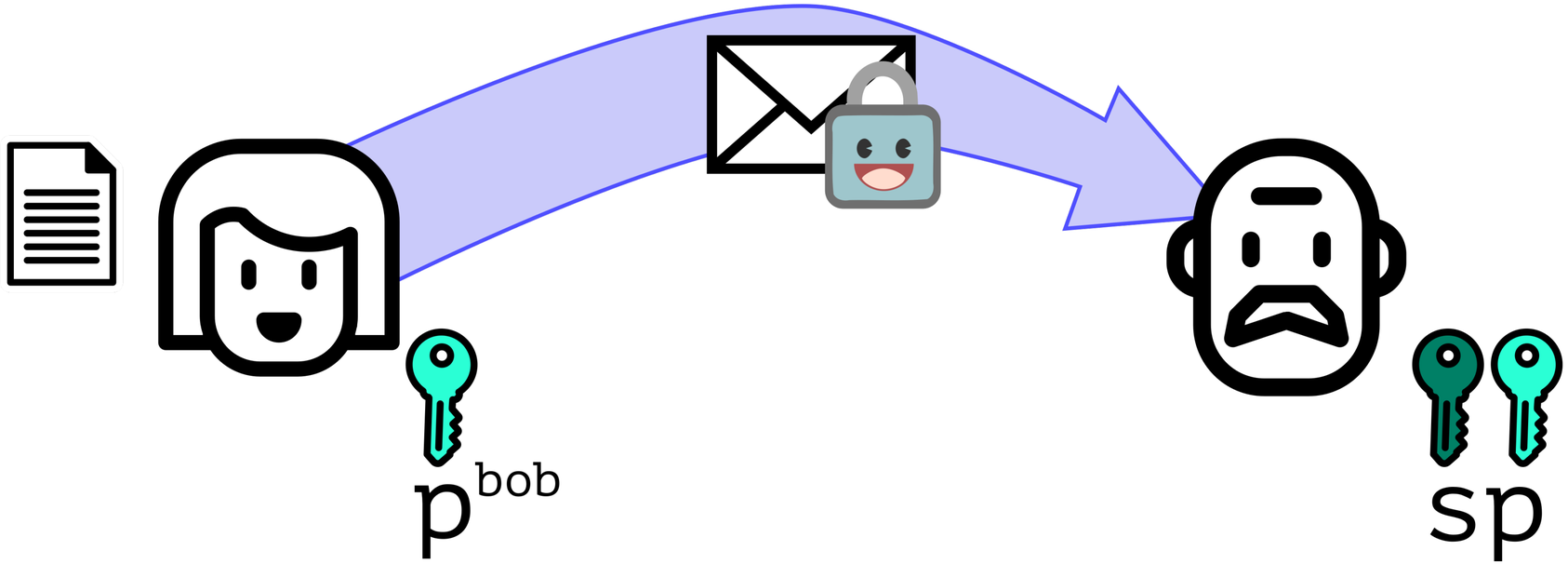
Asymmetric encryption

- Sender encrypts with recipient's public key.



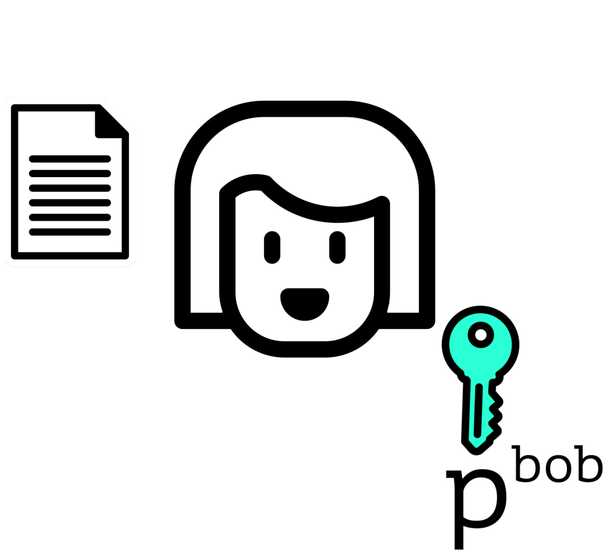


Asymmetric encryption



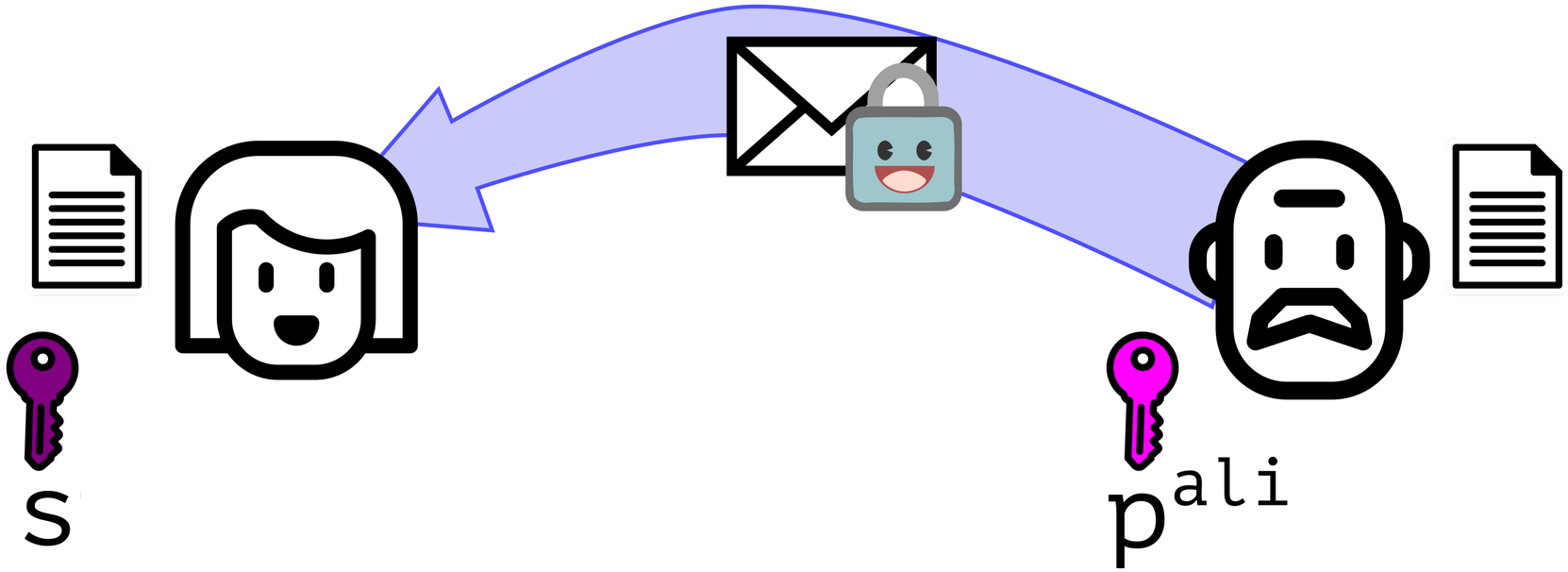
Asymmetric encryption

- Recipient decrypts with their private key.



Asymmetric encryption

- Sender encrypts with recipient's public key.

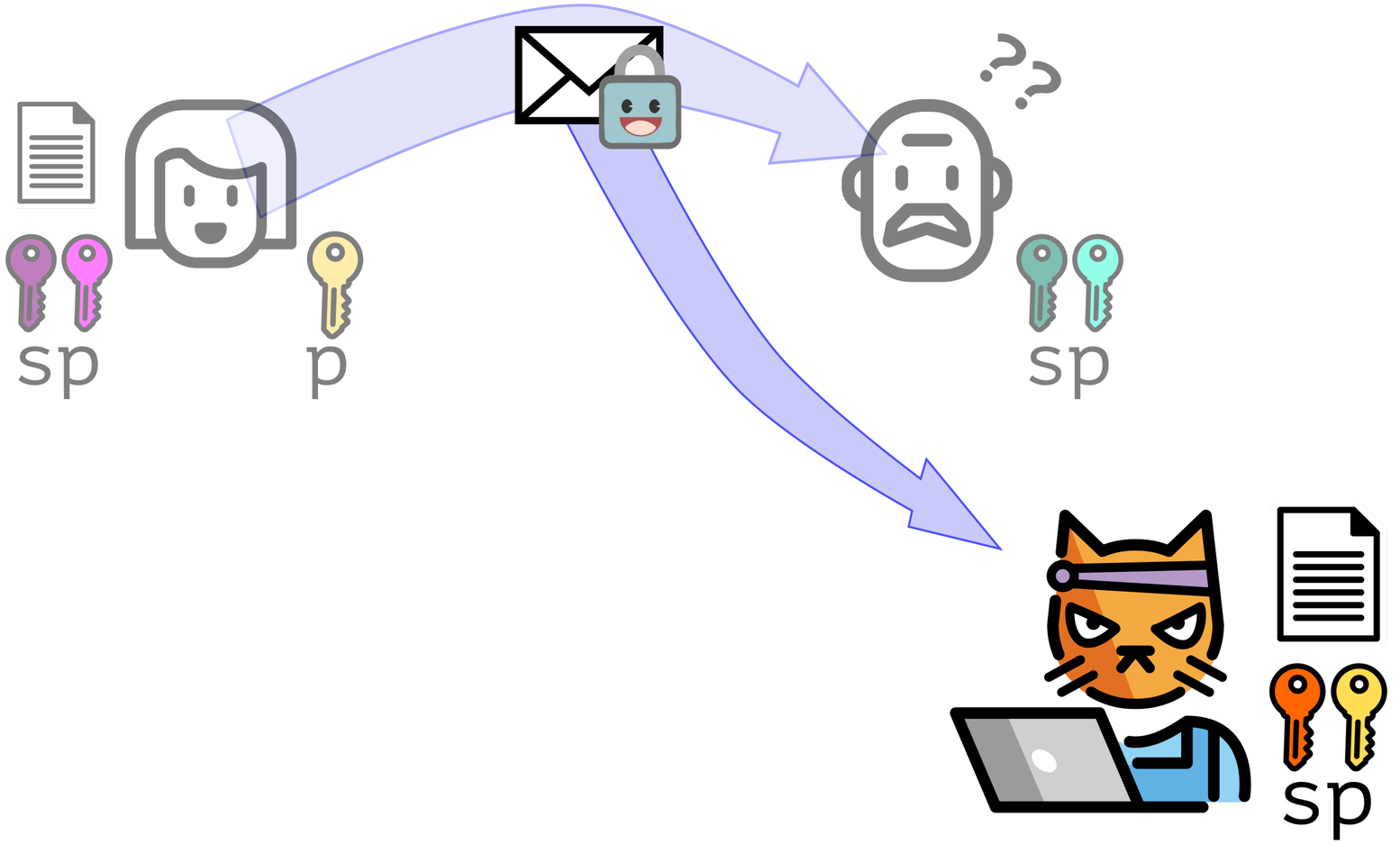


Asymmetric encryption

- BOOM! Confidentiality!
  - Without the worries of key-distribution.

- Daily usage:
  - SSH & TLS: key exchange
  - PGP: email encryption

The next challenge: identity



The next challenge: identity

- Let's add something!
  - *Hashing*, to guarantee integrity.
  - *Signing*, to enable non-repudiation.

- Non-*repudiwhatnow*? 😲
  - You can 100% prove someone did something.

Cryptographic hashing

- Maths which creates a quasi-unique string.
  - The string is derived from the input data.

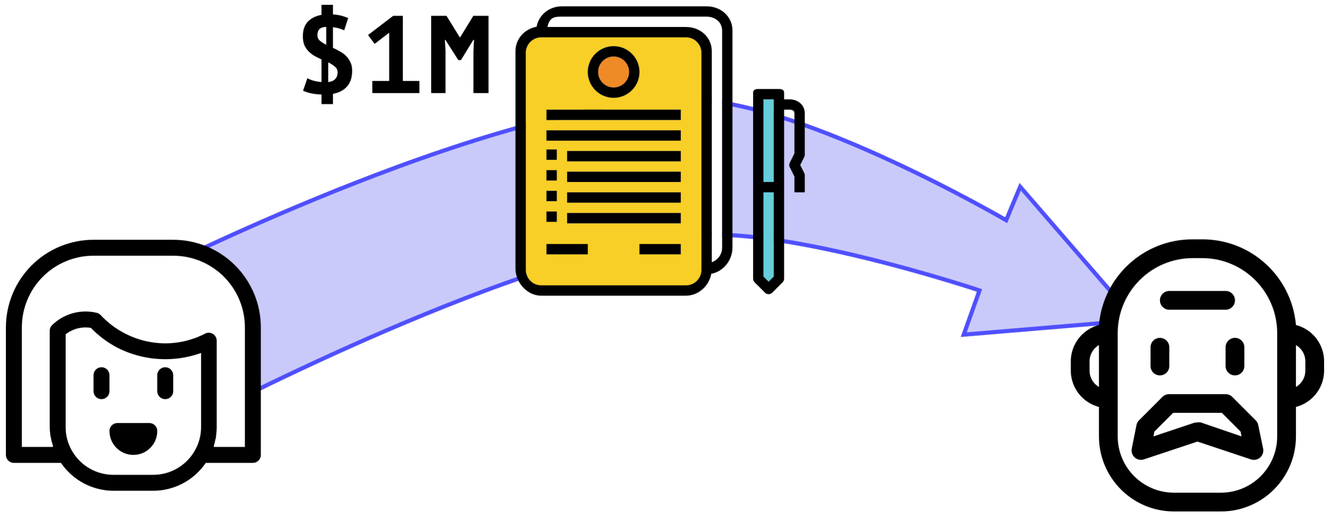
- The same input, will always make the same output.
  - The smallest change, will make big differences.

Signing

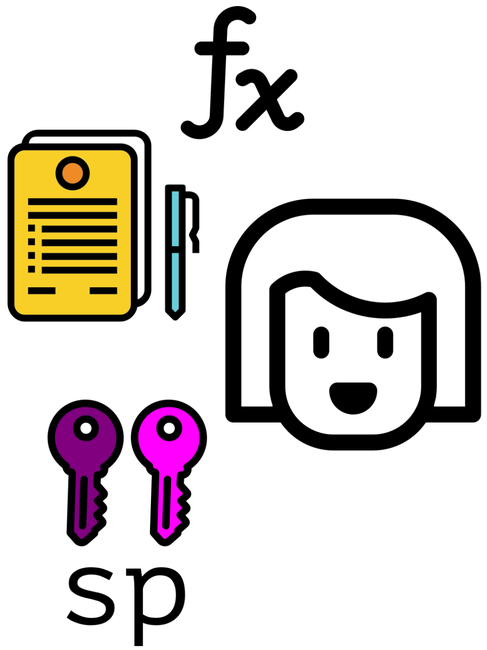
- Calculate your message’s hash.
- Encrypt the hash with your private key.

- Integrity and non-repudiation!
  - Recipient decrypts the hash with your public key.
  - And verifies message integrity.

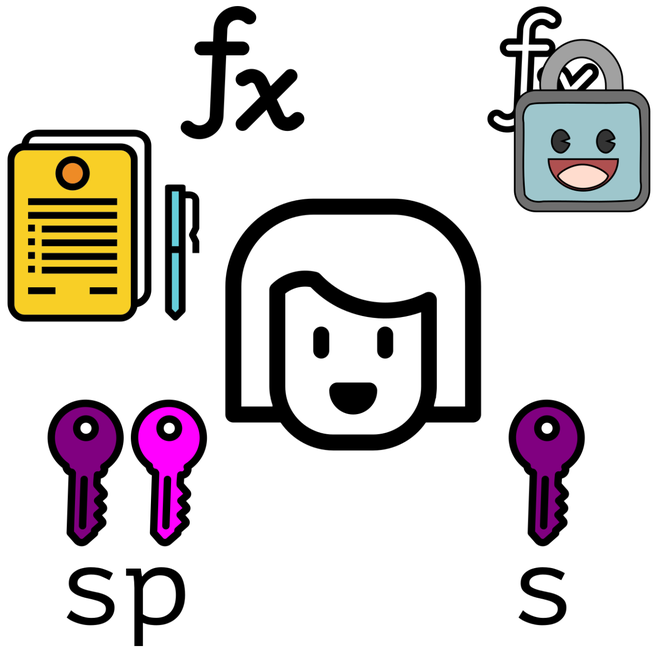
Hashing and signing



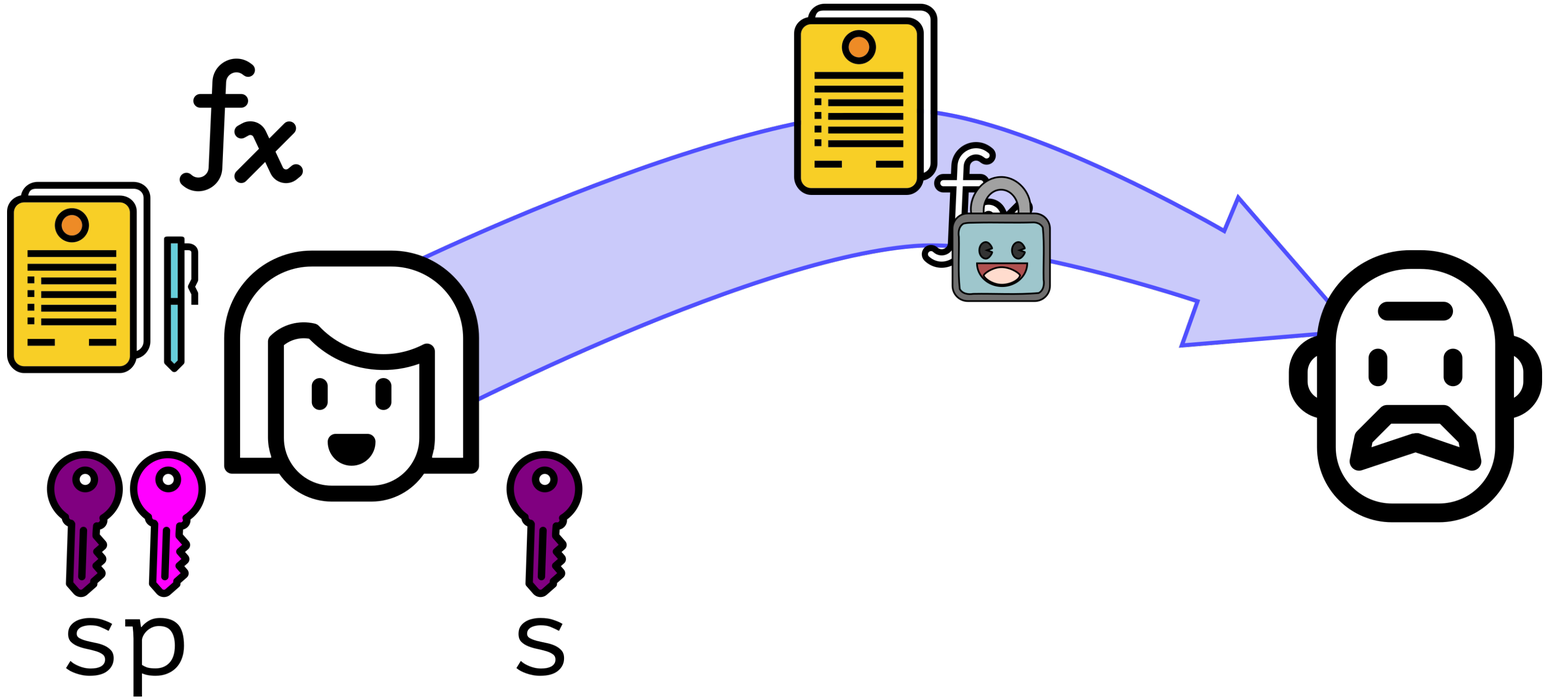
Hashing and signing



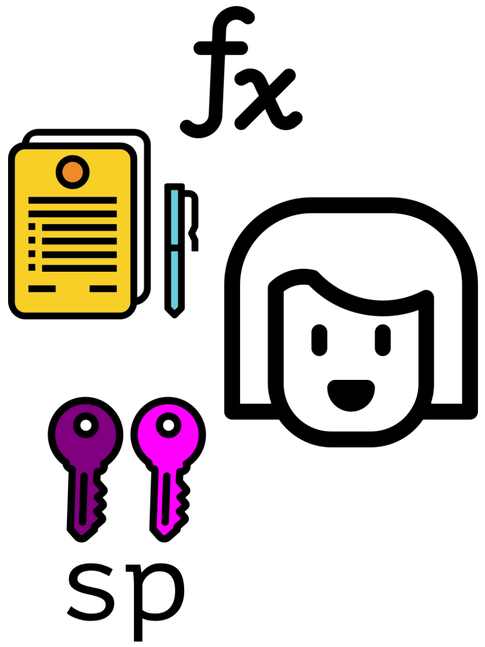
Hashing and signing

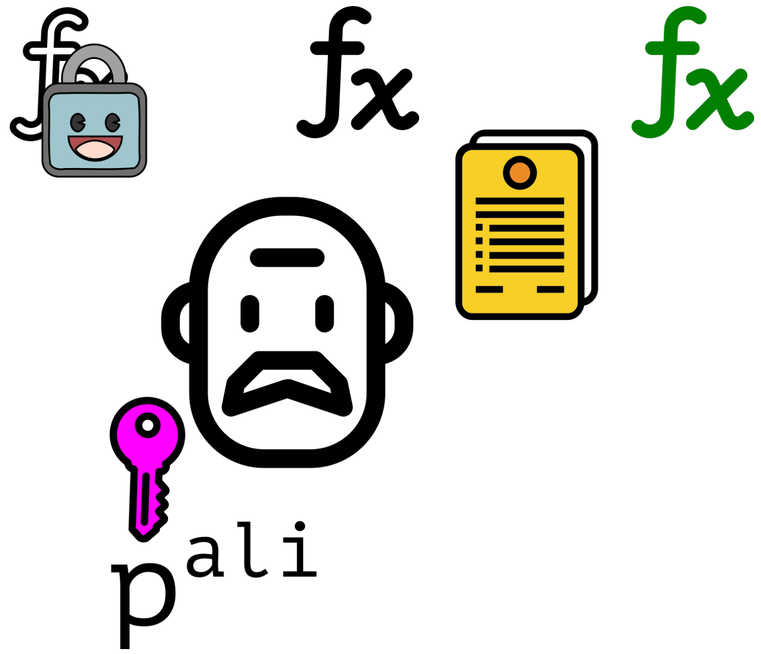


Hashing and signing



Hashing and signing





Certificates

Certificates

- What do we call it when:
  - Someone signs someone else's public key?

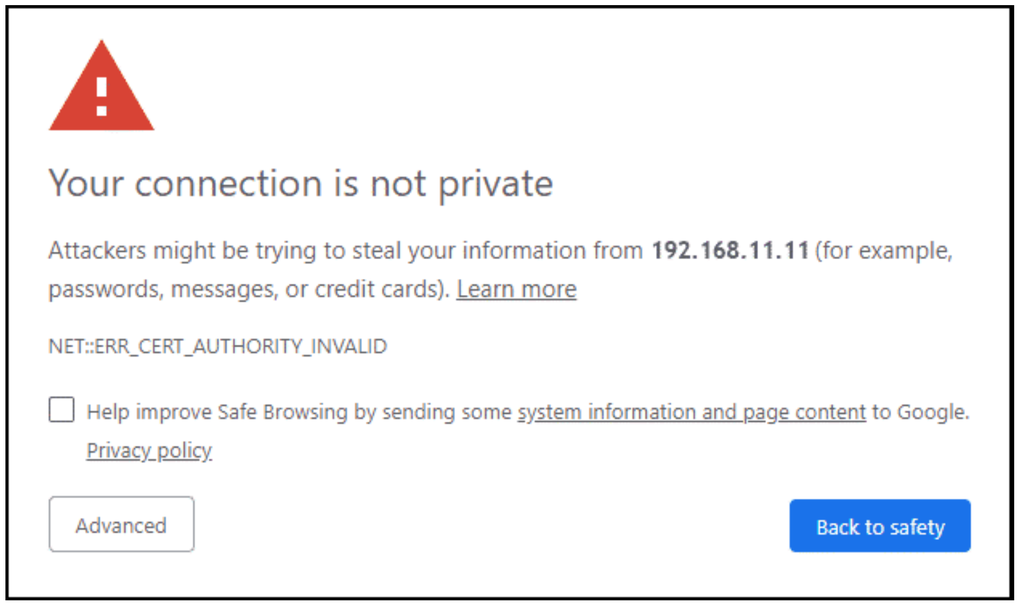
CERTIFICATION

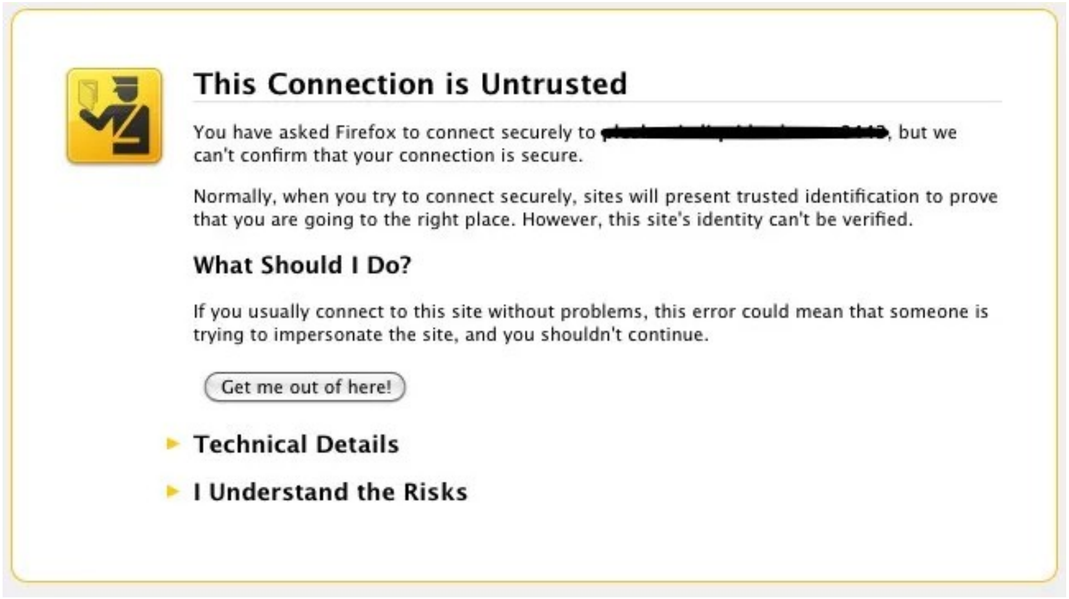
Certificates

- Self-signed certificate
  - Subject states that they are themselves
  - Anybody can make these!

Certificates

- Self-signed certificate





Certificates

- Someone else signs your certificate
  - Another party confirms the identity
  - Still, anybody can make these…

- How do we get to "trust"?

PKI: Public Key Infrastructure

Possible solutions to PKI

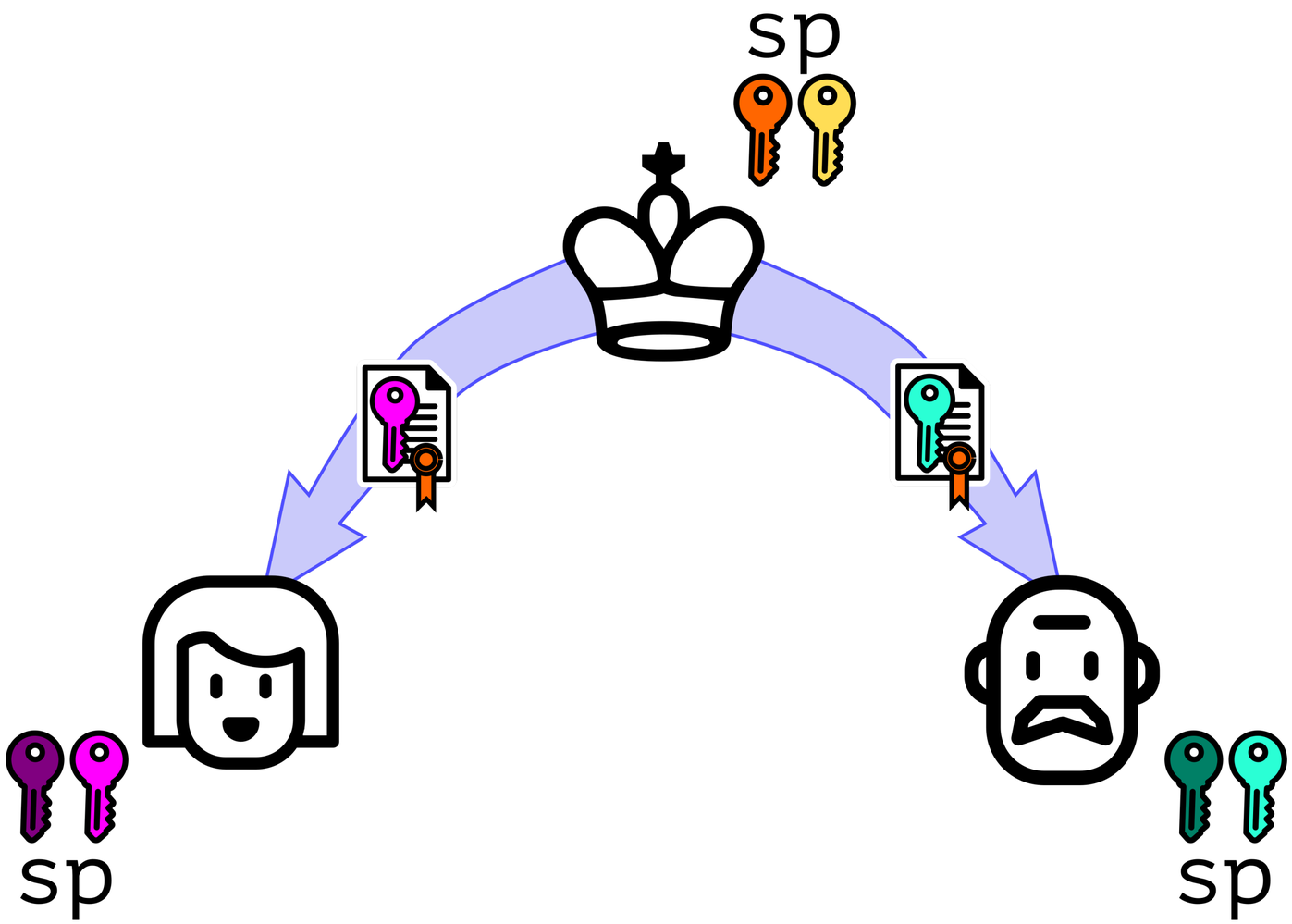
- Certificate Authority (syn. “PKI”)
- Web of trust (e.g. PGP)
- Blockchain

Certificates

- A trusted certificate consists of:
  - A public key,
  - Identifying information and
  - Meta-data

- And is signed by a trusted third party.

Certificate Authority



A Trusted Third Party

- Called a “Certificate Authority”.
- Lots of TTPs: commercial, government, open source
  - *VeriSign, GlobalSign, Let’s Encrypt!, KPN*
  - *Staat der Nederlanden, Department of Defense*

- But you can build your own as well!

A Trusted Third Party

- We trust TTPs because they:
  - Offer guarantees about security
  - Offer guarantees about vetting
  - Offer guarantees about availability

- https://cabforum.org
- https://casecurity.org

Does this ever go wrong?

- YES! Sometimes horribly:
  - *DigiNotar* (GMail attack by Iran)
  - *Comodo* (GlobalTrust)

- Rogue certificates:
  - "*Certificate transparency*" is a must,
  - And so is "*CAA*": CA Authorization

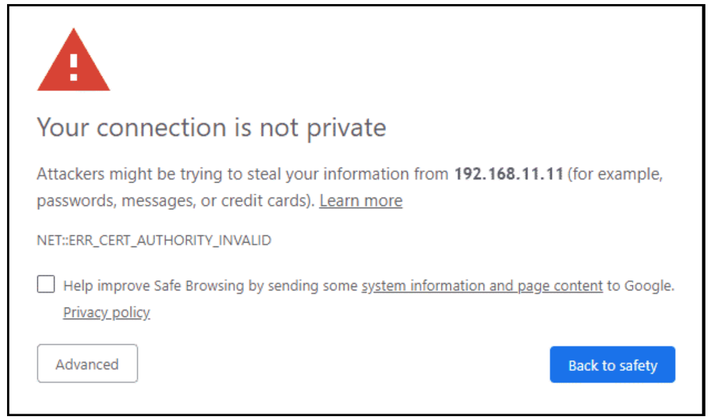
Assuring trust

Trust store

- There isn't ONE standard solution.
  - Most OSes offer a central trust store.
    - *Keychain Access*, *CertMgr.msc*, */etc/tls/certs*
  - Most browsers have their own trust store.
  - Many applications also have their own!

Trust store

- If your trust store isn't complete?
  - Then TLS will refuse to connect.




```
curl: (60) SSL certificate problem: unable to get local issuer
certificate

Exception: javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path validation
failed:
```

When to trust a certificate

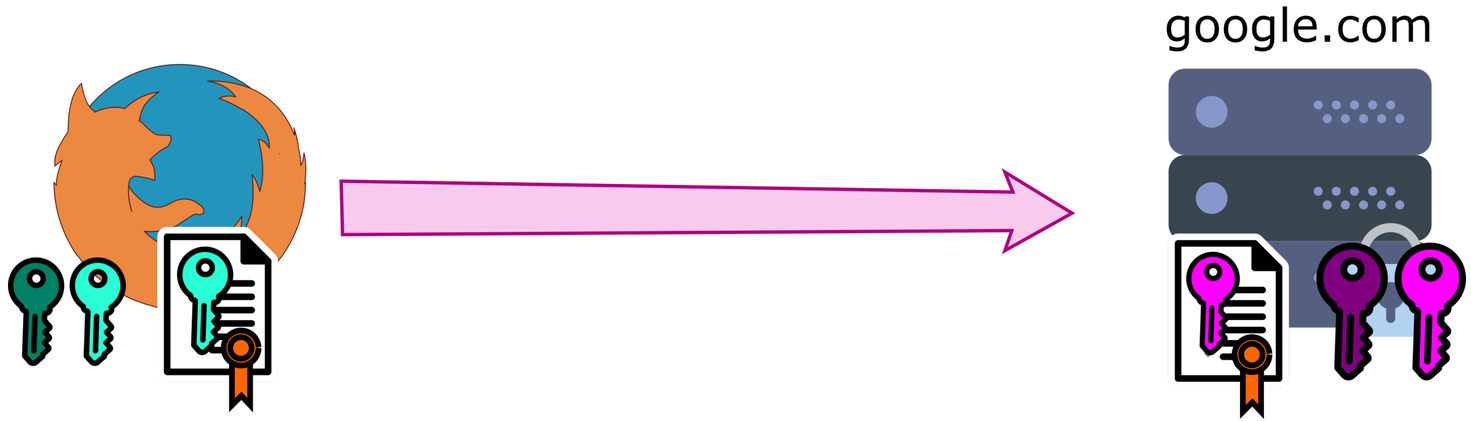
- When it matches our **counterpart**.
- When we actively **trust** the CA.
- When it hasn’t **expired**.
- When it hasn’t been **revoked**.
- When the “**chain**” is valid.

What is a chain?

- A series of valid, trusted certificates.
  - That link the subject,
  - Via the issuing CA(s),
  - To the "trust anchor", the "root CA".

Bringing it all together

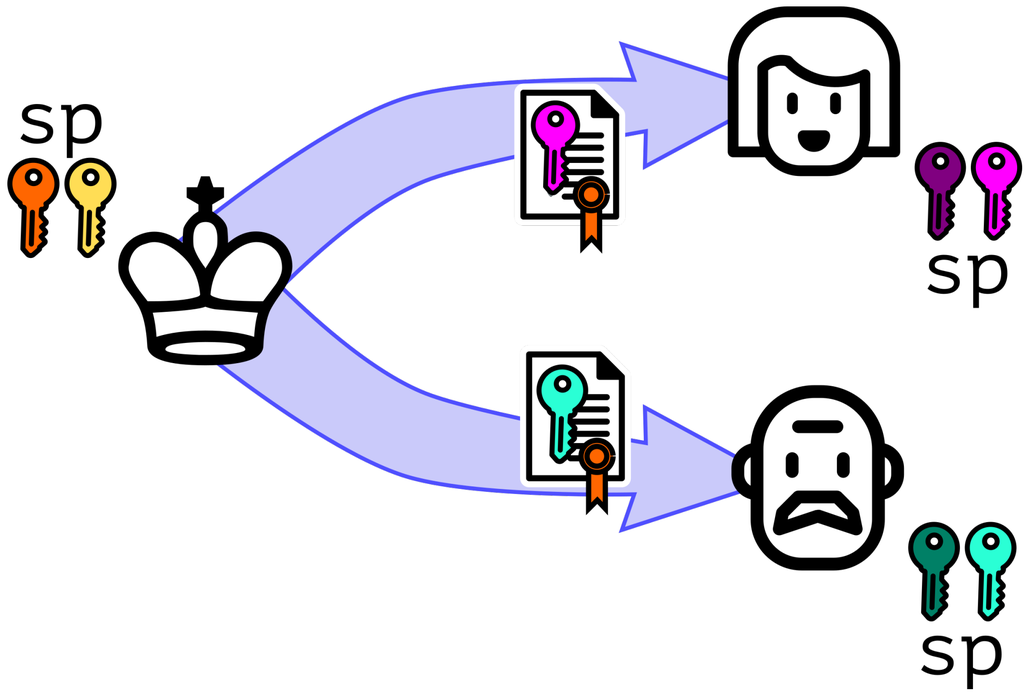
- HTTPS uses asymmetric crypto for key exchange.
  - Allows us to safely setup fast, symmetric crypto.



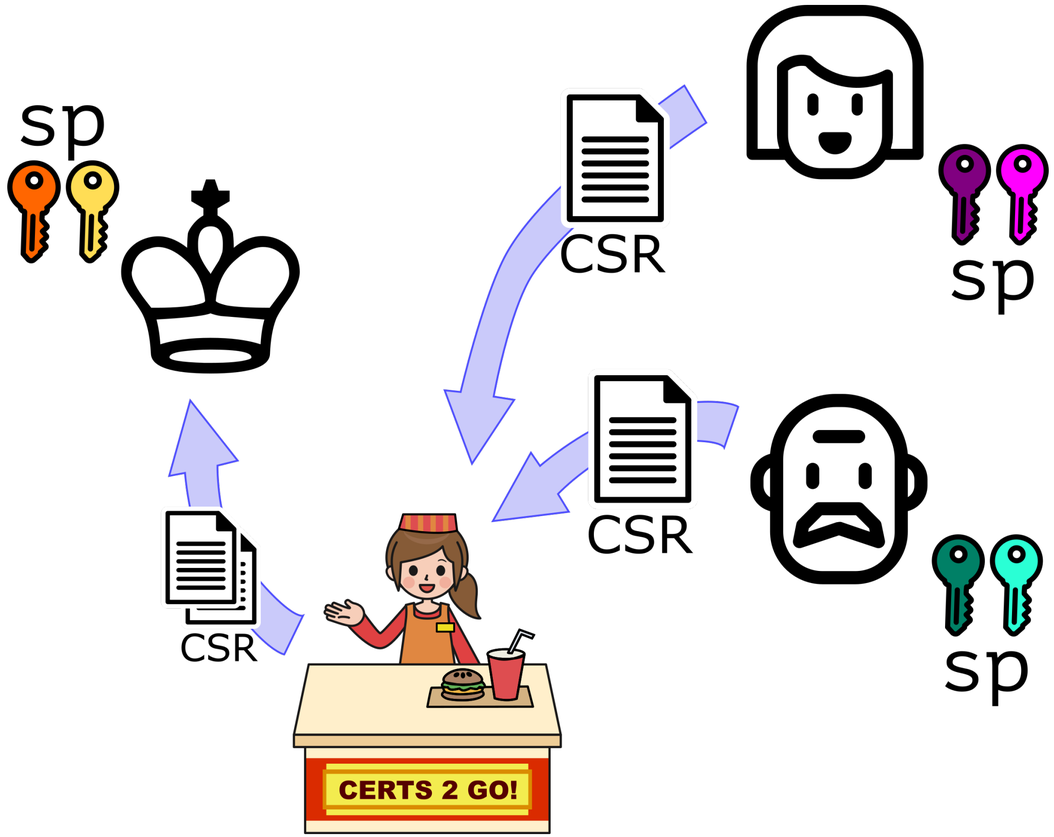
See: How does TLS work? and The illustrated TLS connection

PKI Anatomy

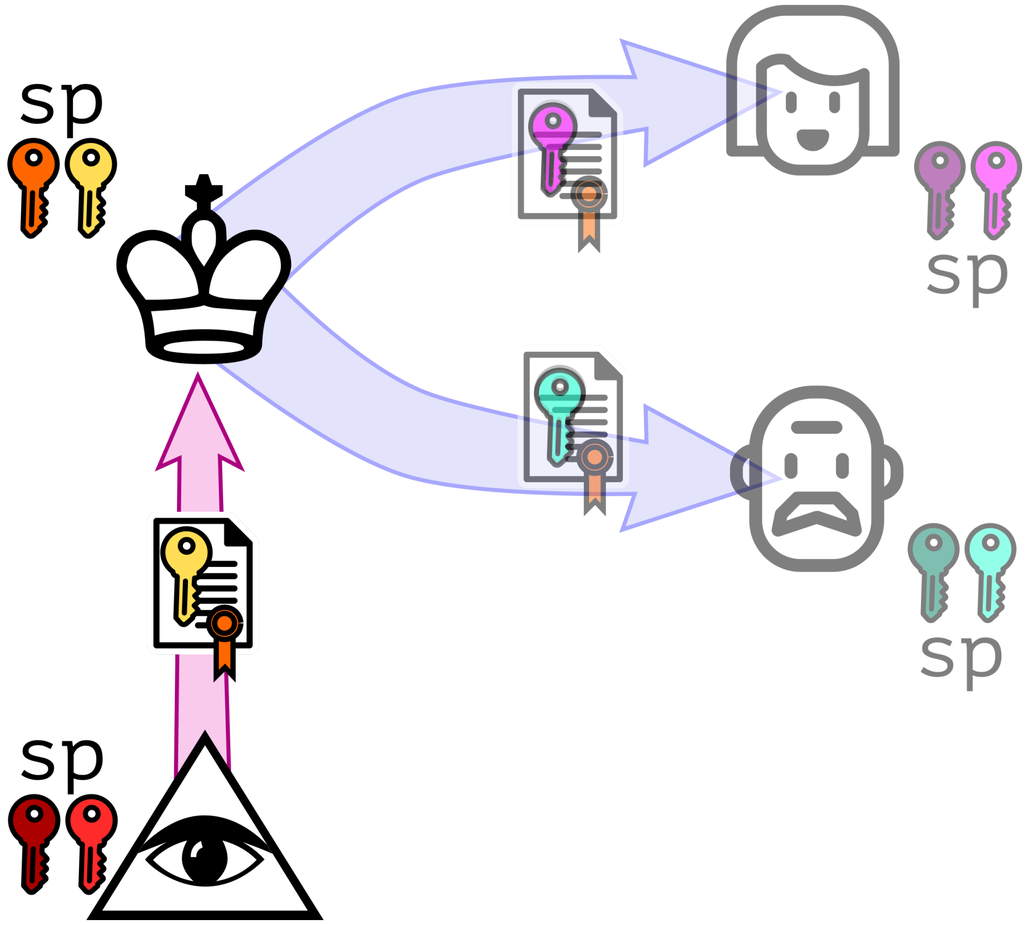
Subjects, Issuing CA



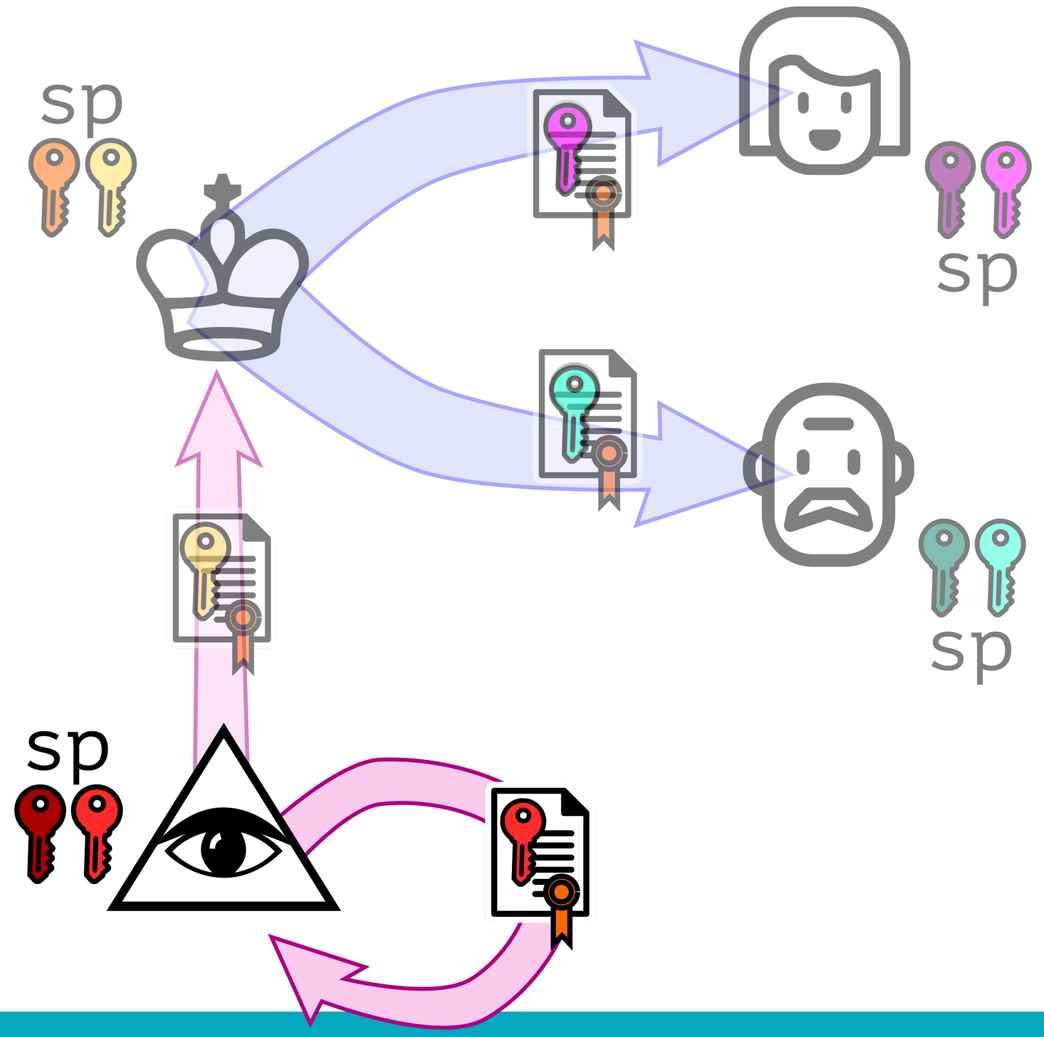
Registration Authority



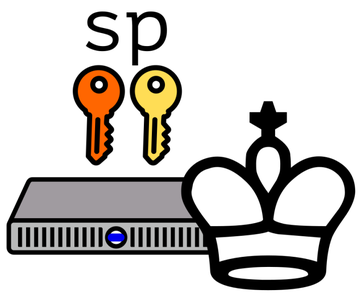
Root CA

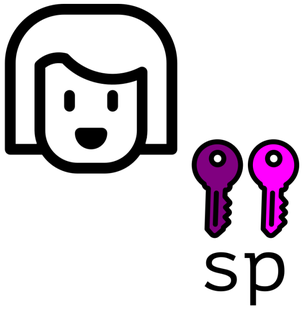


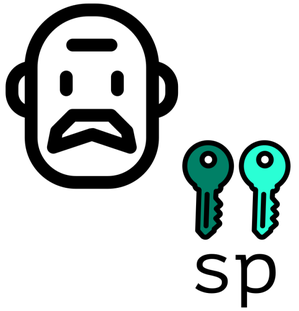
Root CA

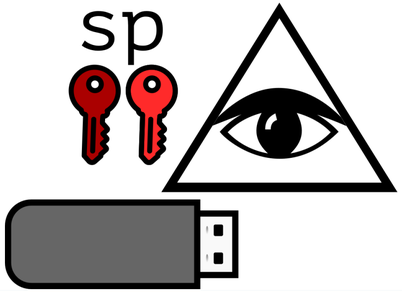


Hardware Security Module

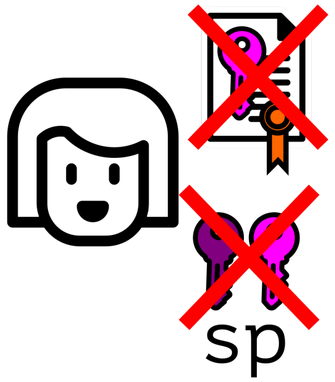


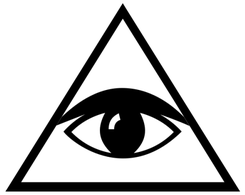




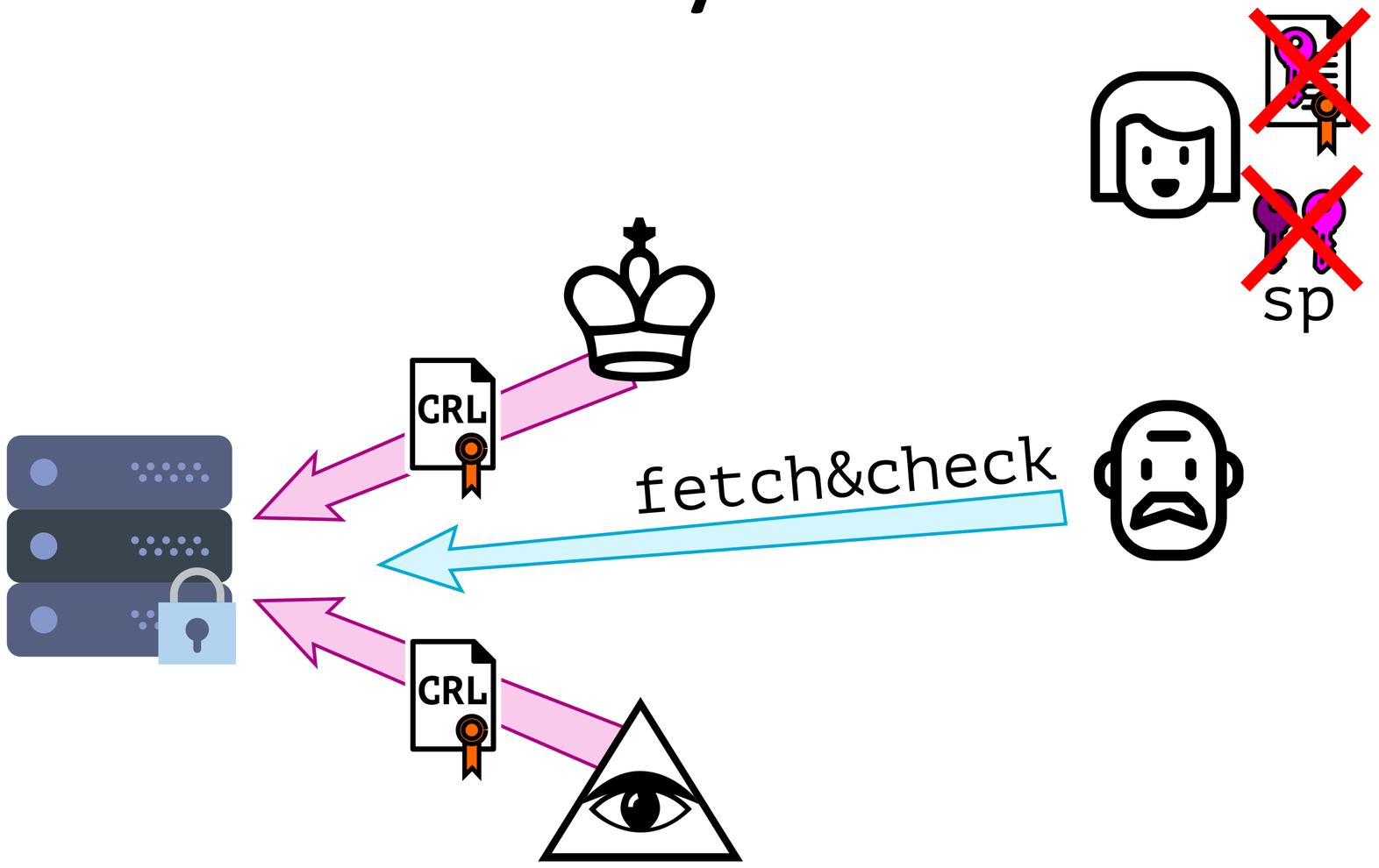


Validation Authority

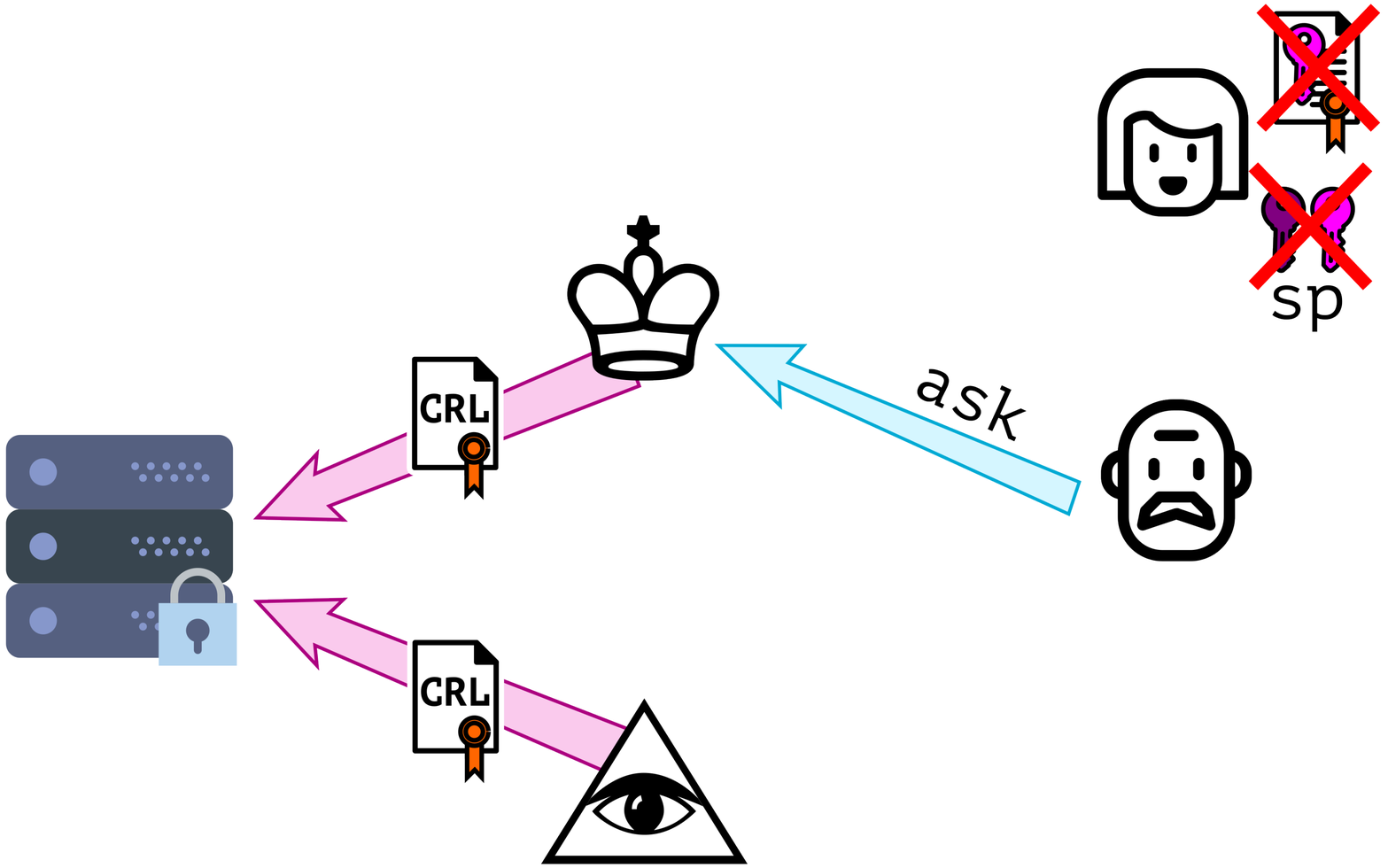




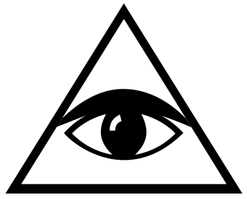
Validation Authority

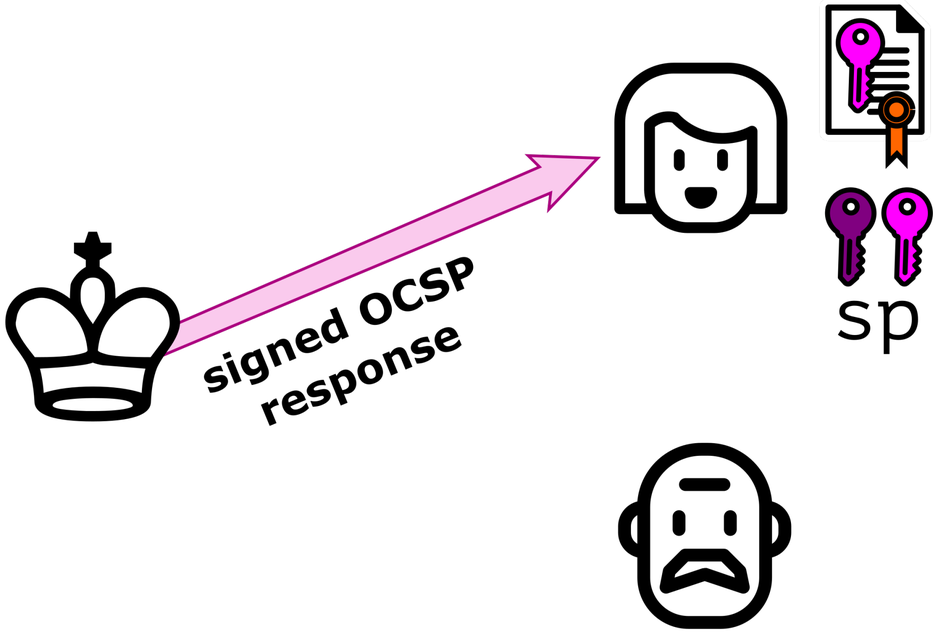


OCSP



OCSP stapling





How can it fail?

- Scott Helme: “*Revocation is broken.*”
  - OCSP breaks privacy.
  - CRL and OCSP can be blocked.
  - Many browsers soft-fail.
  - Many apps hard-fail.

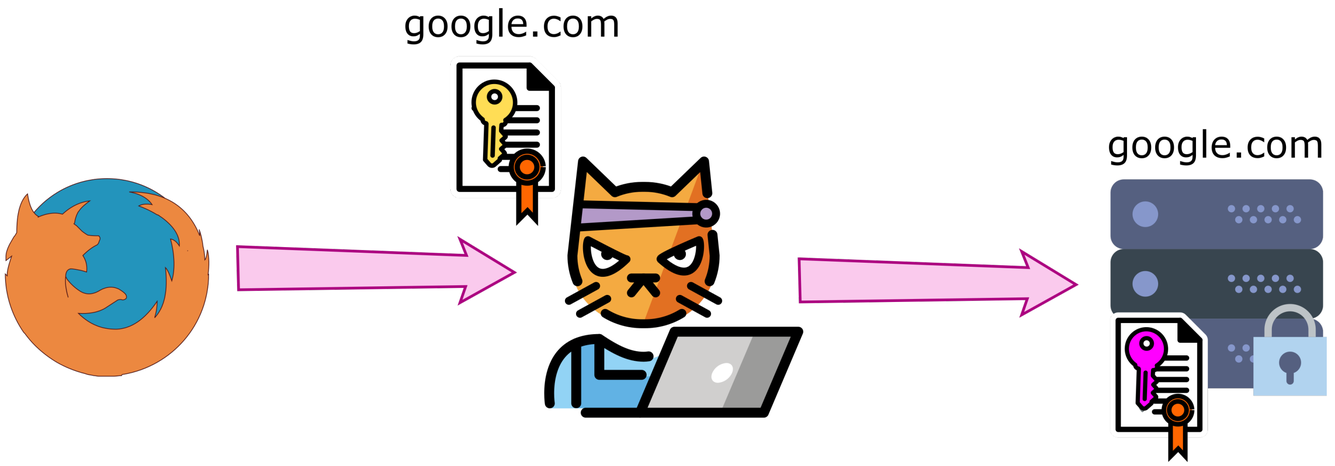
- There are workarounds and fixes.

And why do we trust these?

- Because the ones running the PKI:
  - Offer guarantees about security
  - Offer guarantees about vetting
  - Offer guarantees about availability

Attack Scenarios

On-path, no cert validation

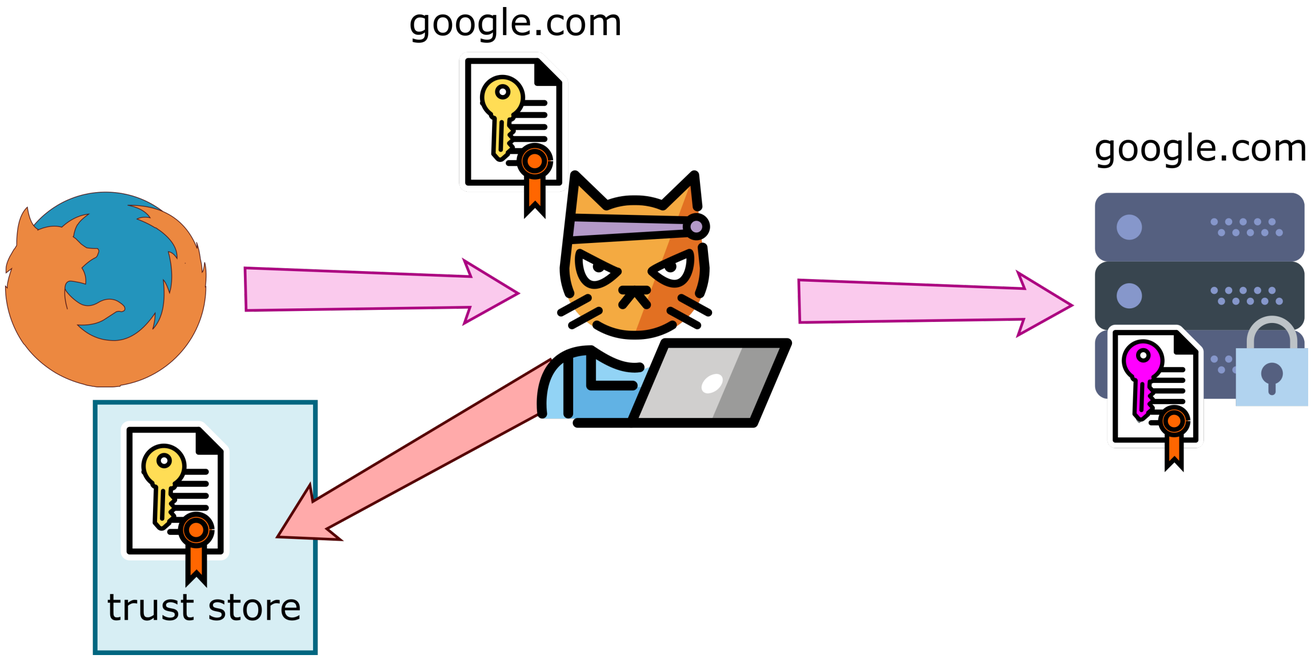



```
curl -k https://www.google.com
```

On-path, no cert validation

- When?
  - Every StackOverflow solution ...
  - Every tutorial ...
  - Every default demo project ...
  - That says: "*just disable TLS validation*".

On-path, breached trust store

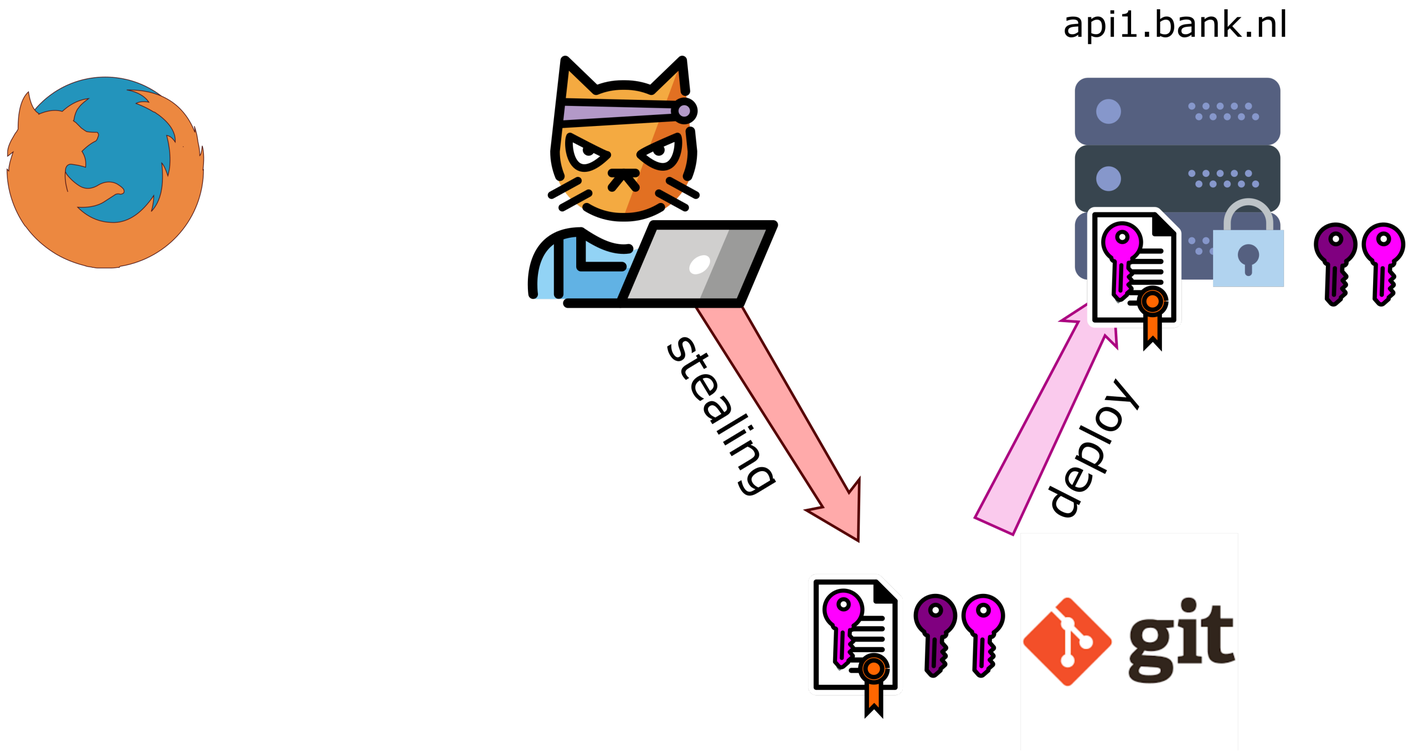


On-path, breached trust store

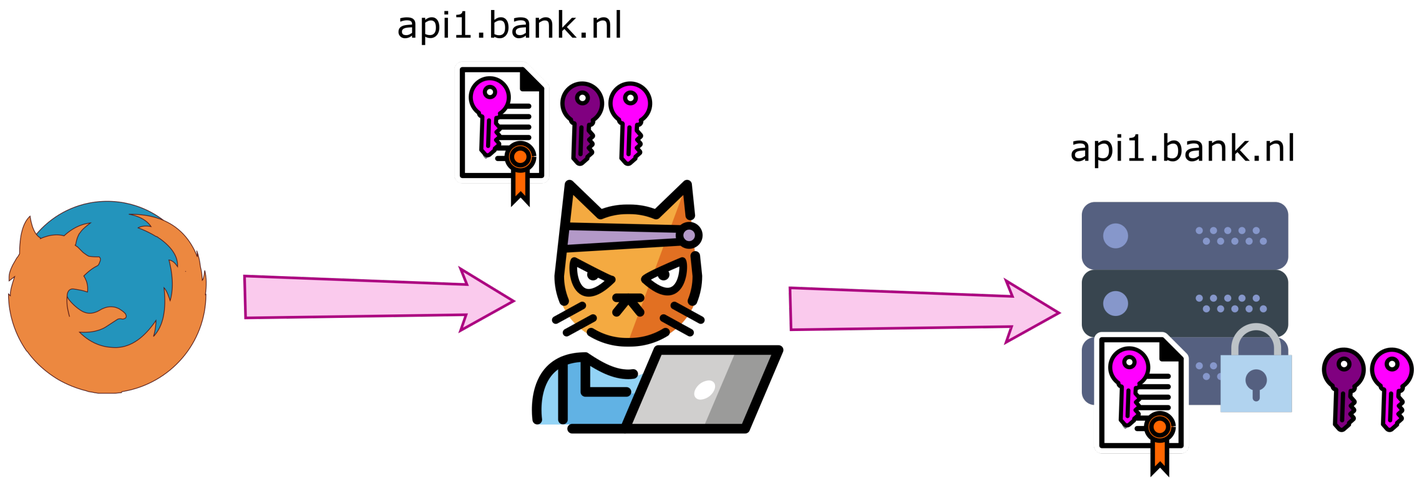
- Lenovo and Kazakhstan did this.

- When?
  - Malware
  - Internet inspection proxy servers
  - Government oversight

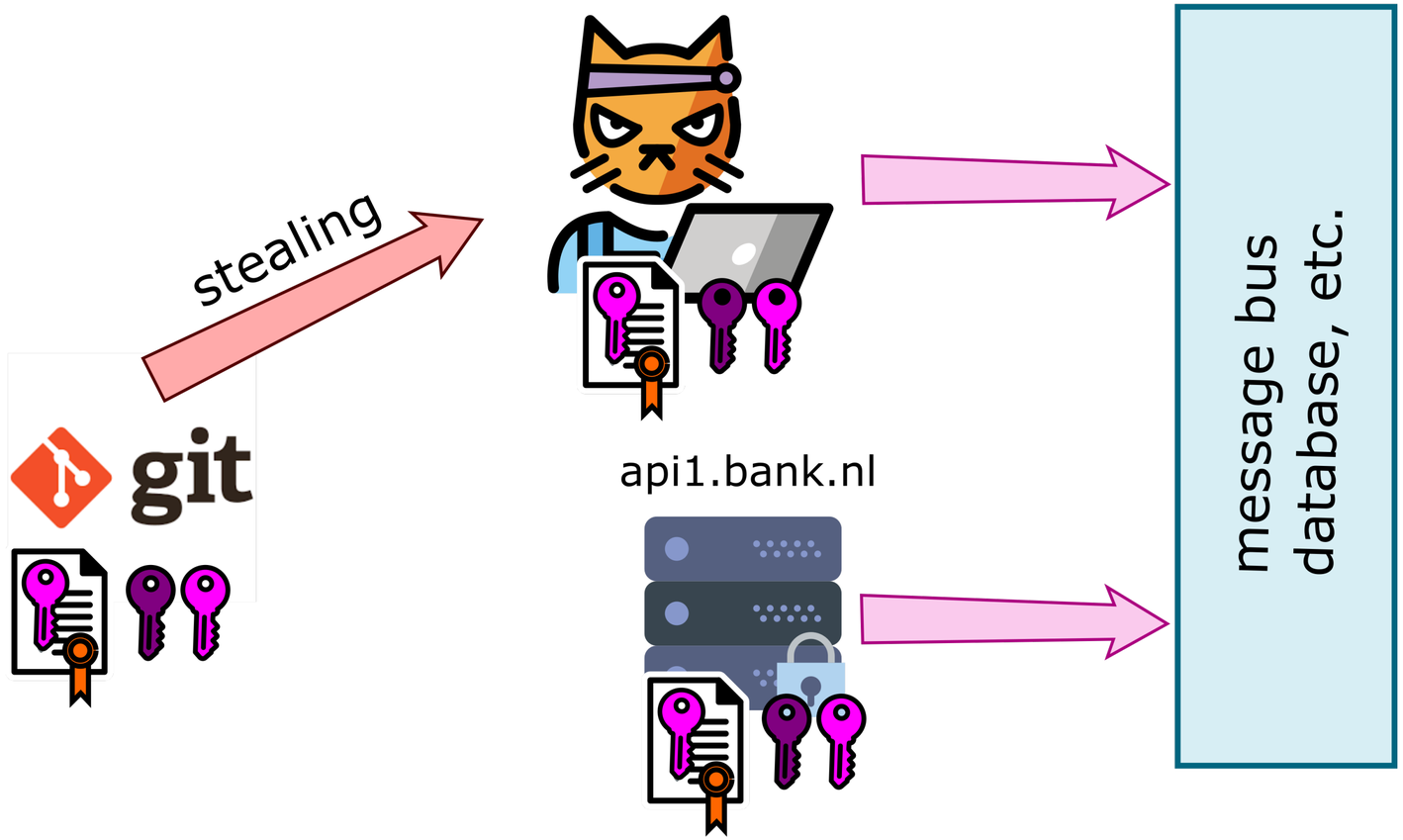
On-path, stolen keys+cert



On-path, stolen keys+cert



Impersonation

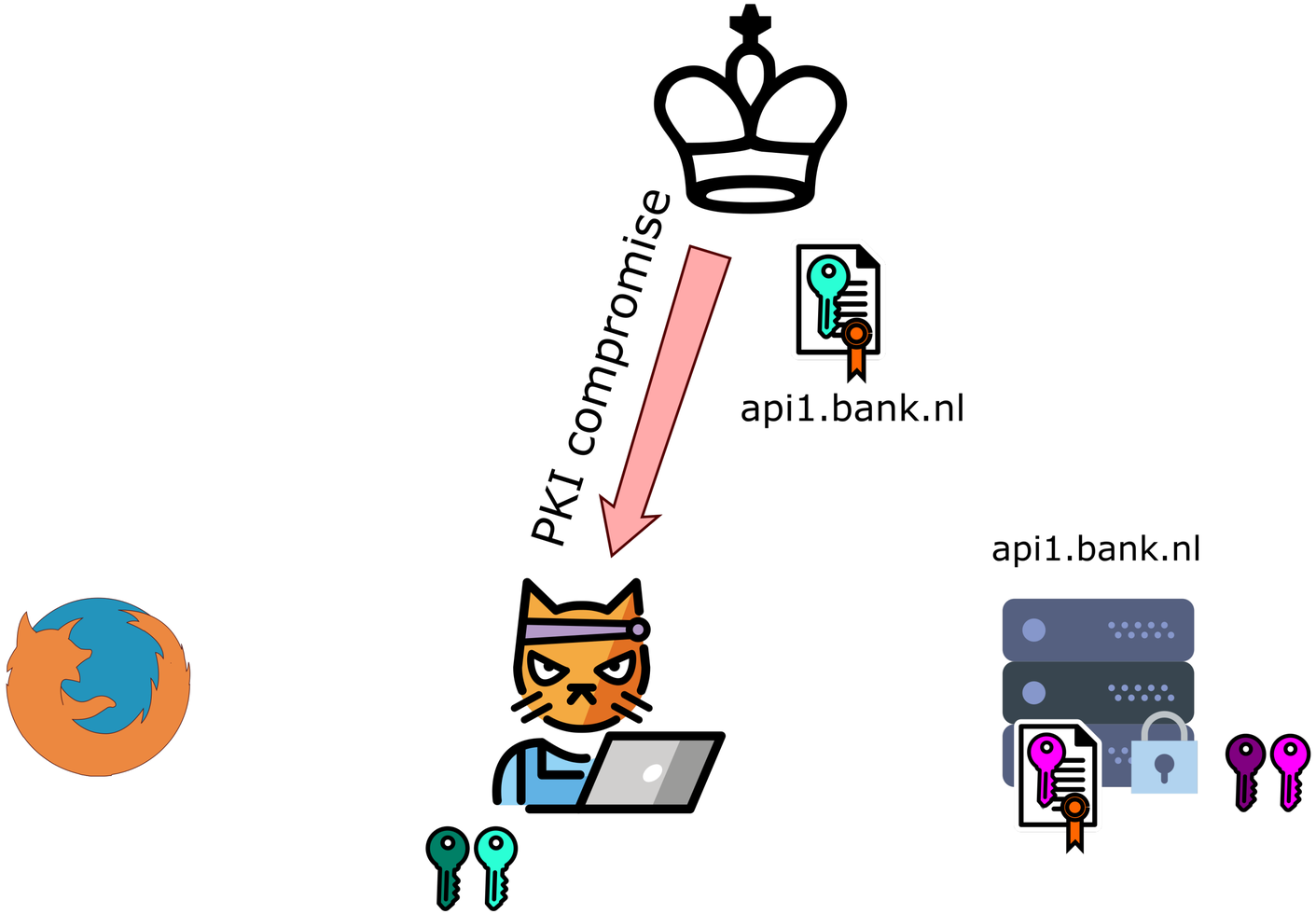


On-path, stolen keys+cert

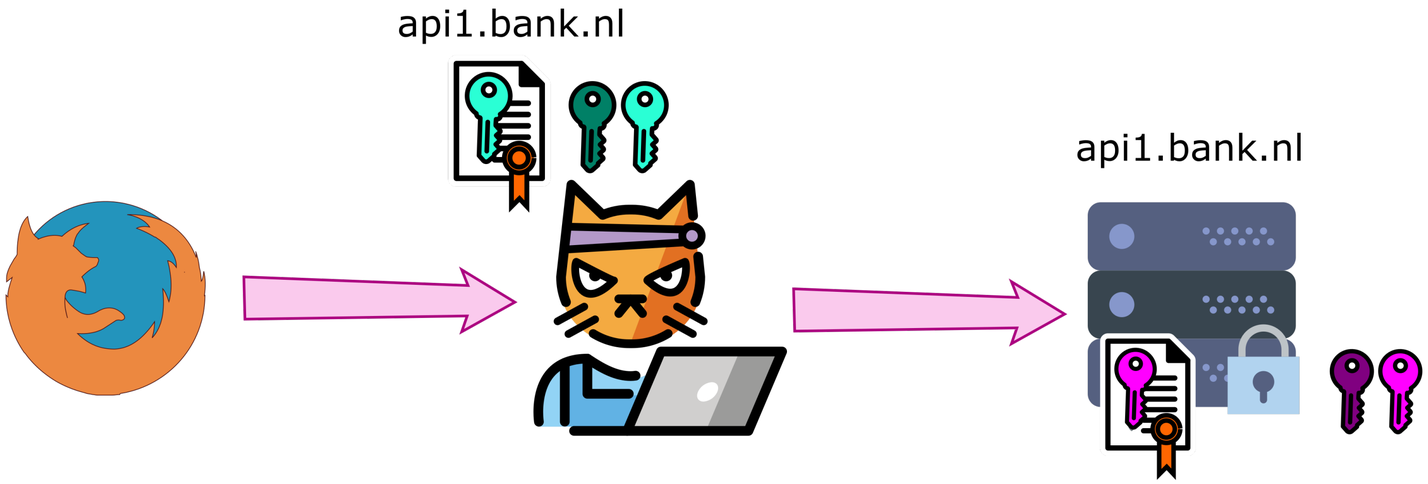
- This happened to Microsoft and Github!

- When?
  - Malware
  - APT: *Advanced Persistent Threat*
  - Criminal gangs

On-path, forged certificate



On-path, forged certificate



On-path, forged certificate

- This happened to Diginotar!

- When?
  - Malware
  - APT: *Advanced Persistent Threat*

Best practices

Your product MUST

- Trust your company's PKI.
- Validate and verify all certificates.
- Have its own, valid certificates.
- Store its keys securely.
- Use approved algorithms and libraries.

Your product must NOT

- Trust random CAs, have a writable trust store.
- Ignore certificate validation.
- Use self-signed or bogus certificates.
- Store keys in Git, file shares or outside the server.
- Use outdated crypto.
- "Roll your own" crypto. Never!