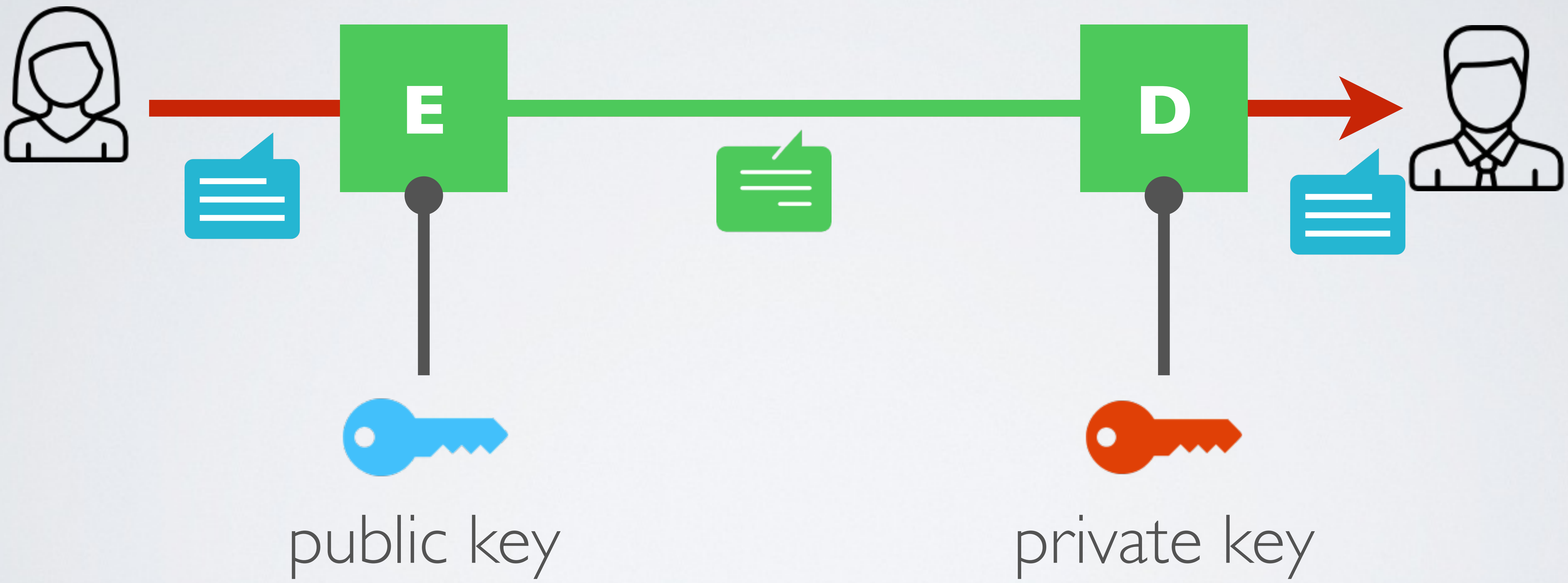
Applied Asymmetric Cryptography

Protocols, Attacks, Implementation Flaws

Kc Udonsi

Refresher

Asymmetric encryption
a.k.a Public Key Cryptography



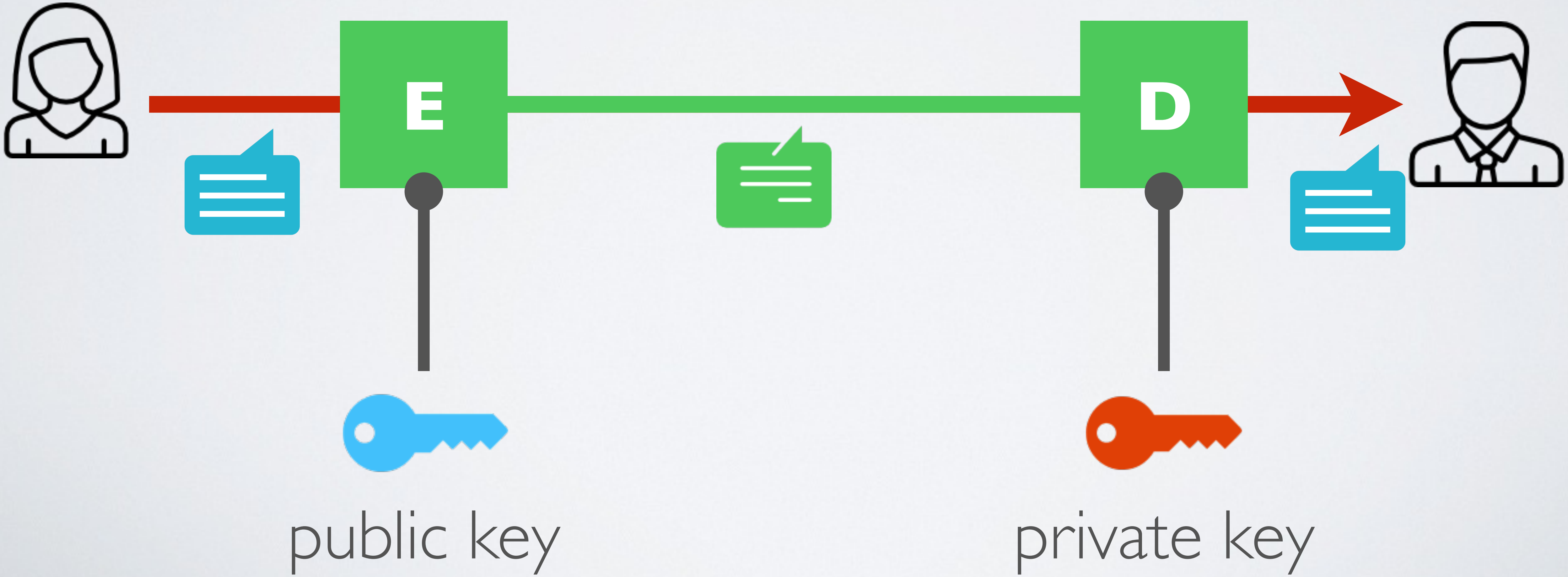
Asymmetric Keys - Functional Requirements

$D_{Ks}(E_{Kp}(m)) = m$ and $D_{Kp}(E_{Ks}(m)) = m$ for every pair (Kp, Ks)

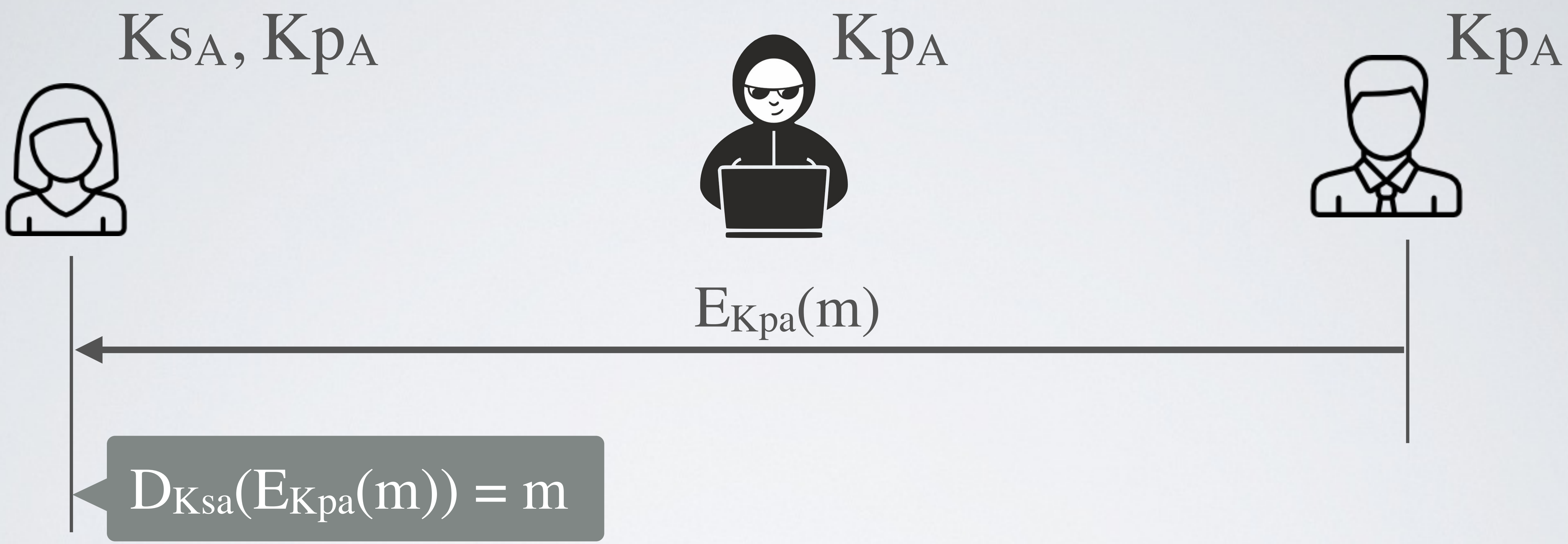
- ✓ Generating a pair (Kp, Ks) is easy to compute (polynomial)
- ✓ Encryption is easy to compute (either polynomial or linear)
- ✓ Decryption is easy to compute (either polynomial or linear)
- ◉ Finding a matching key Ks for a given Kp is hard (exponential)
- ◉ Decryption without knowing the corresponding key is hard (exponential)

Asymmetric encryption
a.k.a Public Key Cryptography

- ➡ The public key for encryption
- ➡ The private key for decryption



Asymmetric encryption for **confidentiality**



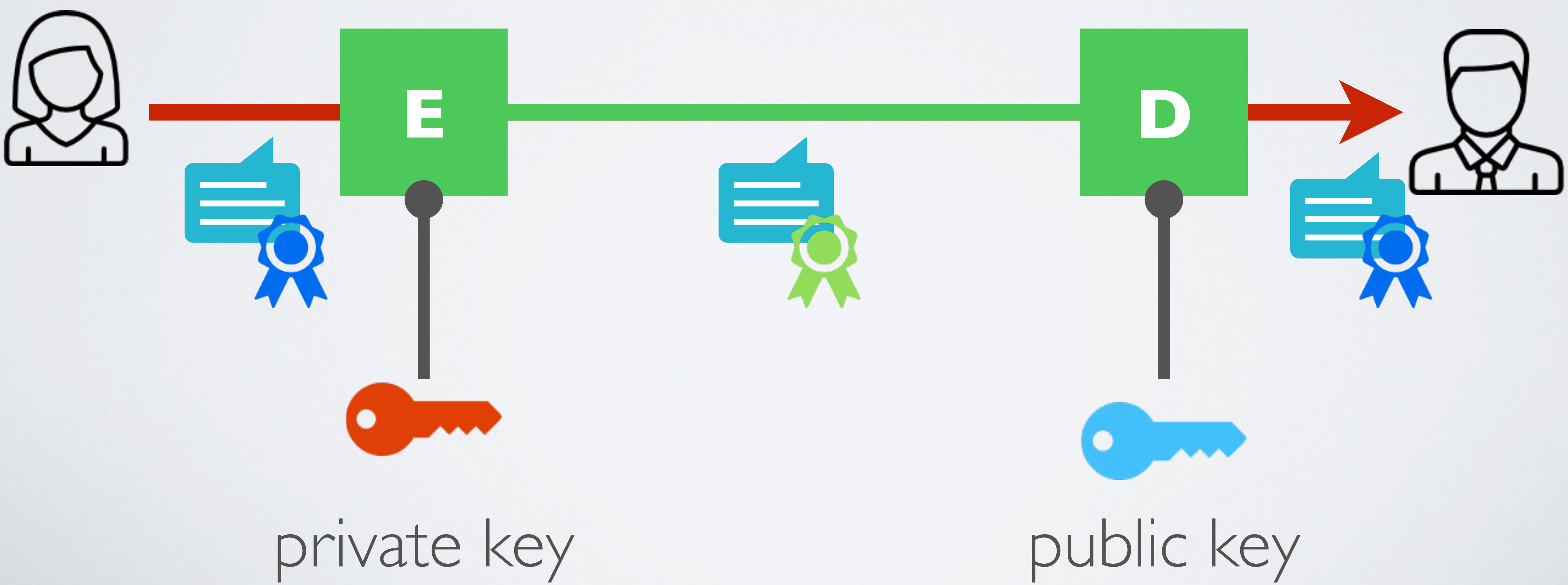
Bob encrypts a message **m** with Alice's public key $\mathbf{Kp_A}$

➡ Nobody can decrypt **m**, except Alice with her private key $\mathbf{Ks_A}$

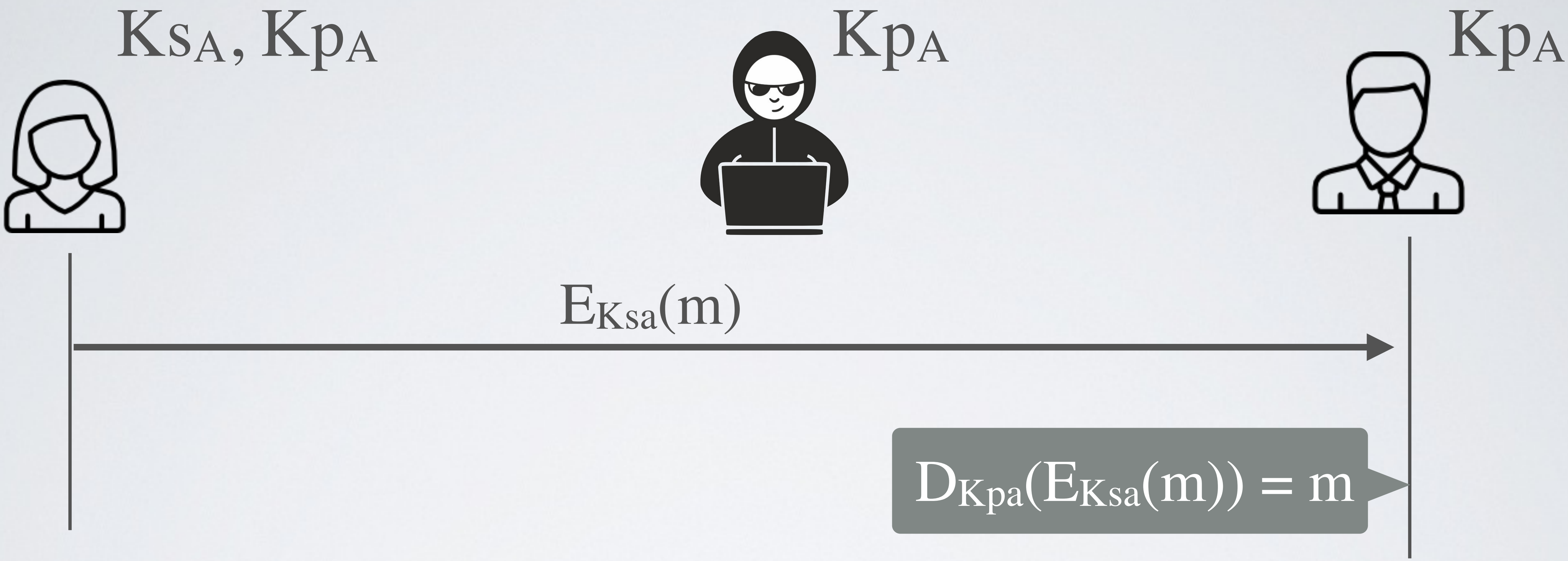
✓ Confidentiality without the need to exchange a secret key

Asymmetric encryption: Digital Signature

- ➡ The private key for encryption
- ➡ The public key for decryption



Asymmetric encryption for **integrity**

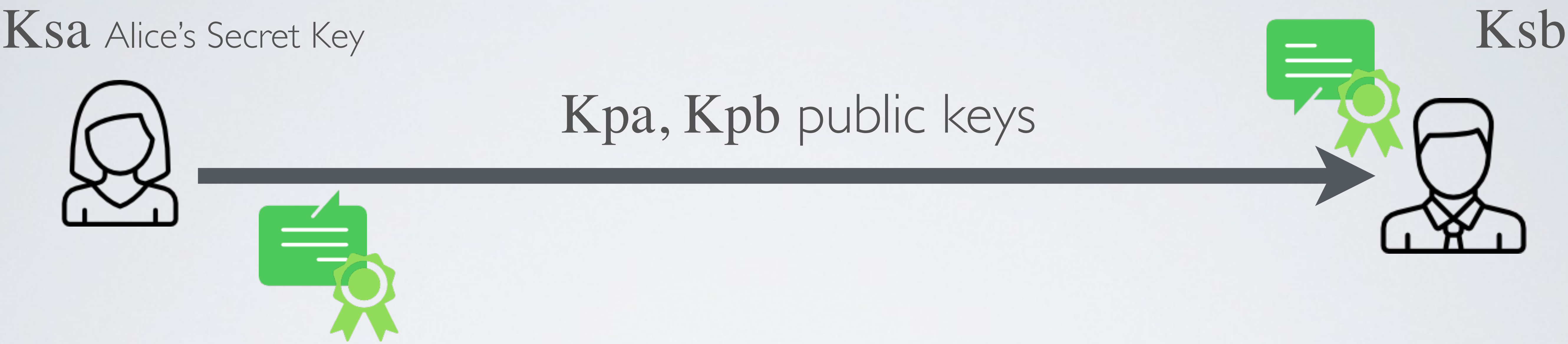


Alice encrypts a message **m** with her private key $\mathbf{Ks_A}$

➡ Everybody can decrypt **m** using Alice's public key $\mathbf{Kp_A}$

✓ Authentication with non-repudiation (a.k.a Digital Signature)

Digital Signature



- ➡ Use public cryptography to **sign and verify**

$$m \mathbin{\|} SIG_{Ksa}(m)$$

$$SIG_{Ksa}(m) = E_{Ksa}(H(m))$$

Non-repudiation as a special case of integrity

	MAC	Digital Signature
Integrity	✓	✓
Non-repudiation	✗	✓

Digital Signatures and Confidentiality

1. Alice generates a symmetric session key k
2. Use both symmetric and asymmetric cryptography to **encrypt, sign and verify** the message and the key

$$E_{Kpb}(k) \,||\, E_{k}(m \,||\, E_{Ksa}(H(m))$$

Goals

1. Establish a session key to exchange data while ensuring Perfect Forward Secrecy

   ✓ Use the Diffie-Hellman key exchange protocol

2. Ensure one-way or mutual authentication

   ✓ Use asymmetric encryption

Protocols

The Needham-Schroeder public-key protocol for mutual authentication

Assumptions and Goals

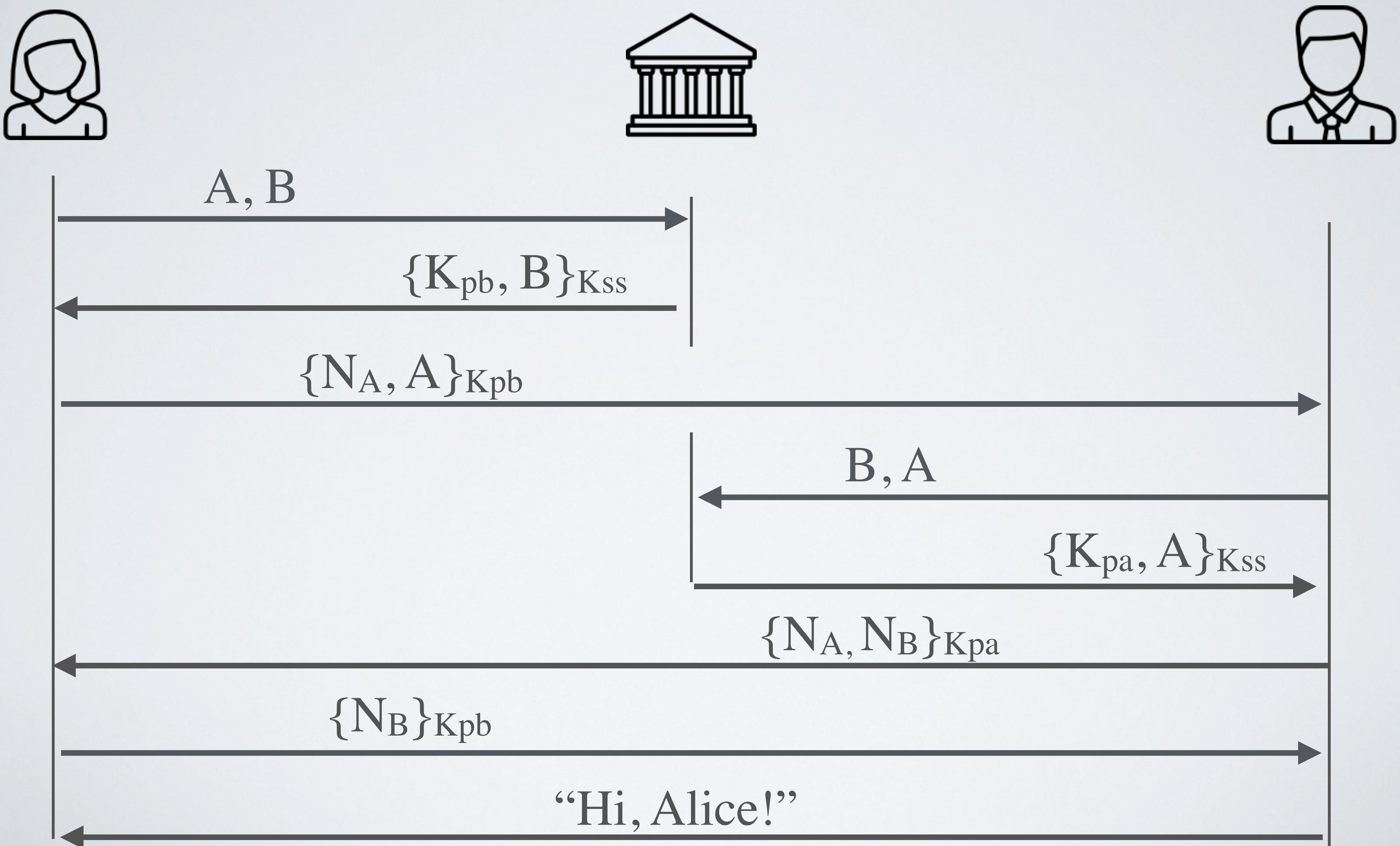
Assumptions

- 4 principals : **A**lice, **B**ob, **M**allory and a Public-Key **S**erver
- Alice, Bob, Mallory and the Server have generated their own public/private key pair
- Alice, Bob and Mallory know the Server's public key **Kps**
- **A**, **B** , **M** and **S** talk to each other using the same protocol

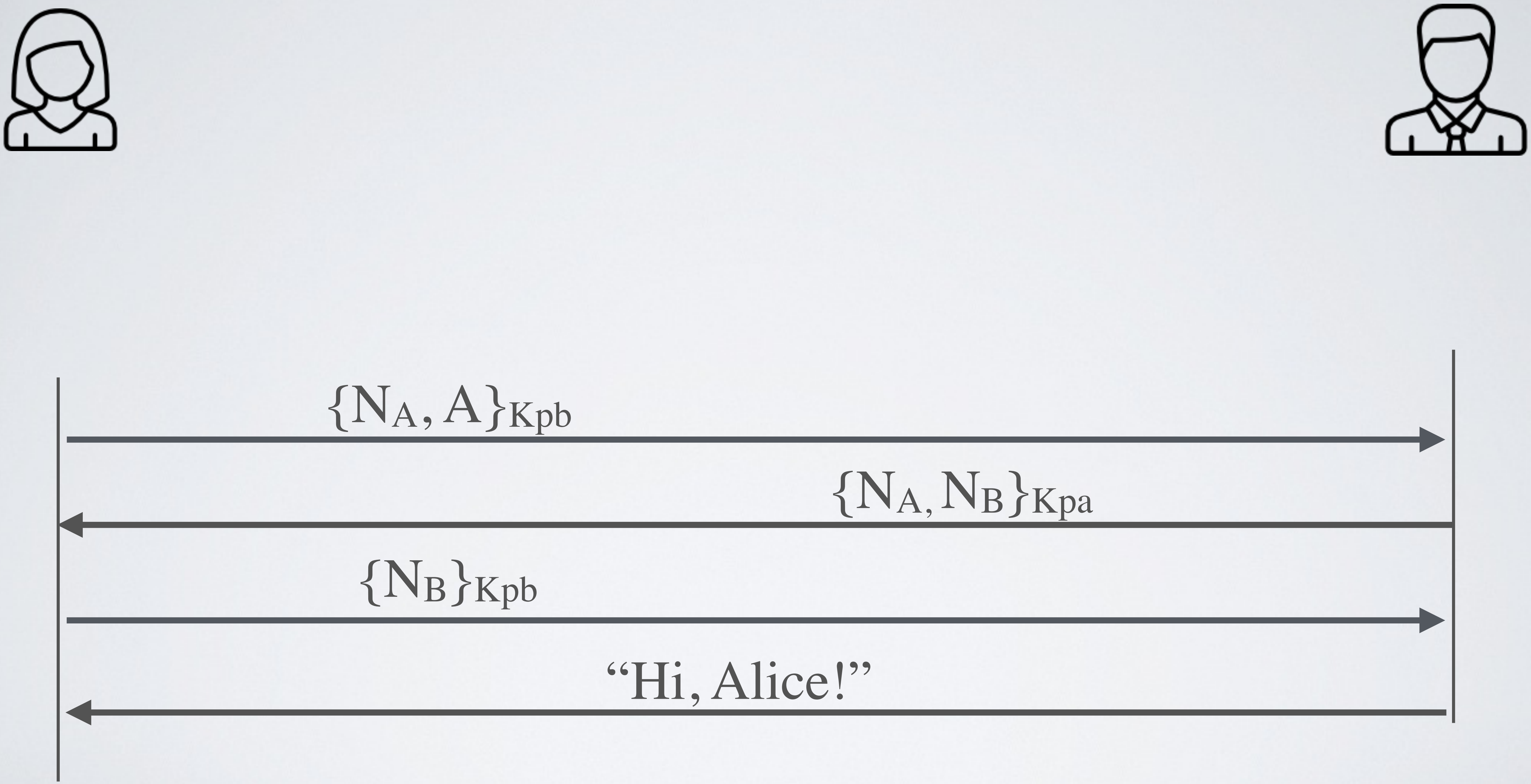
Goals

When two parties want to engage in the communication, they want to make sure that they talk to the right person (authentication)

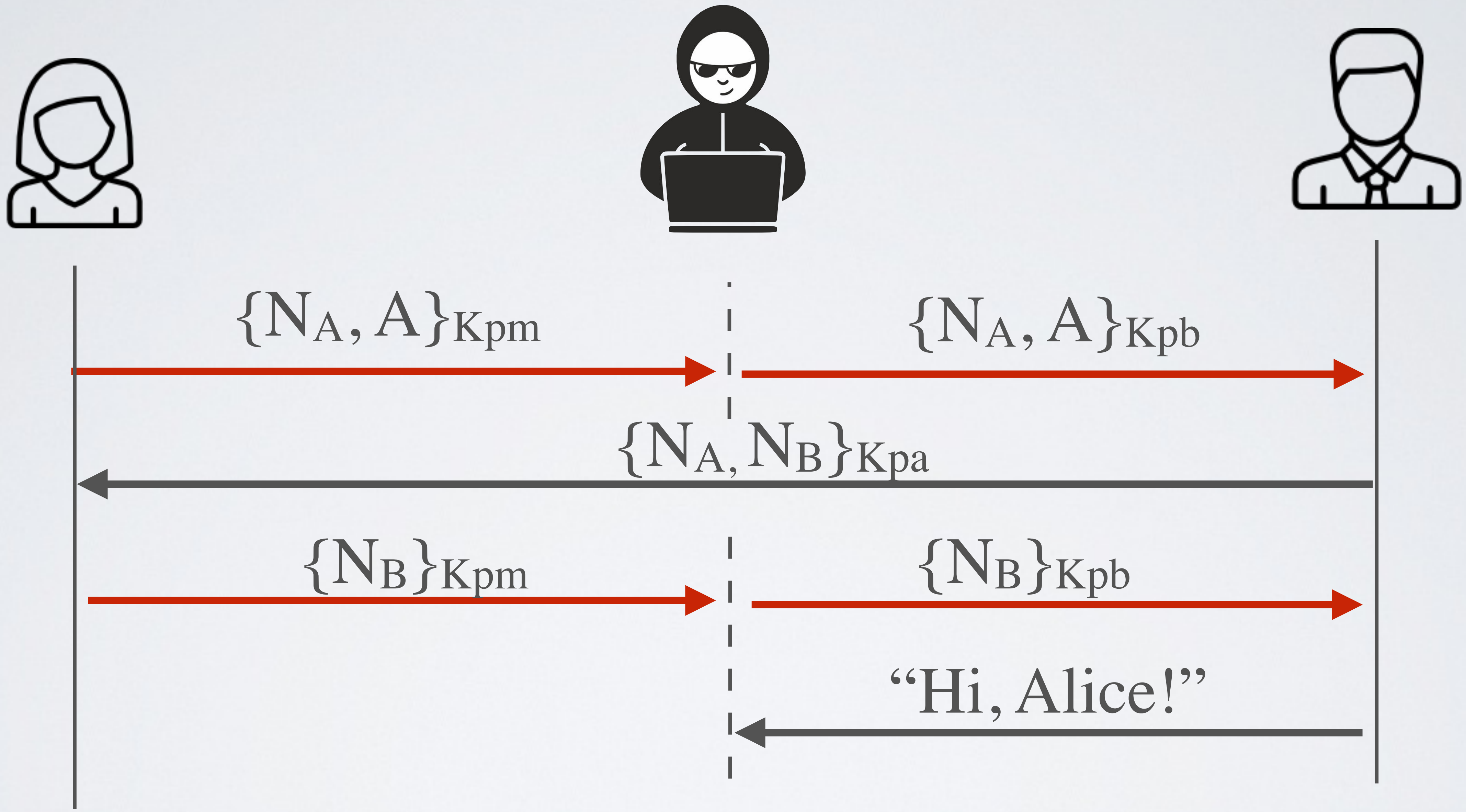
The vulnerable version (1978)



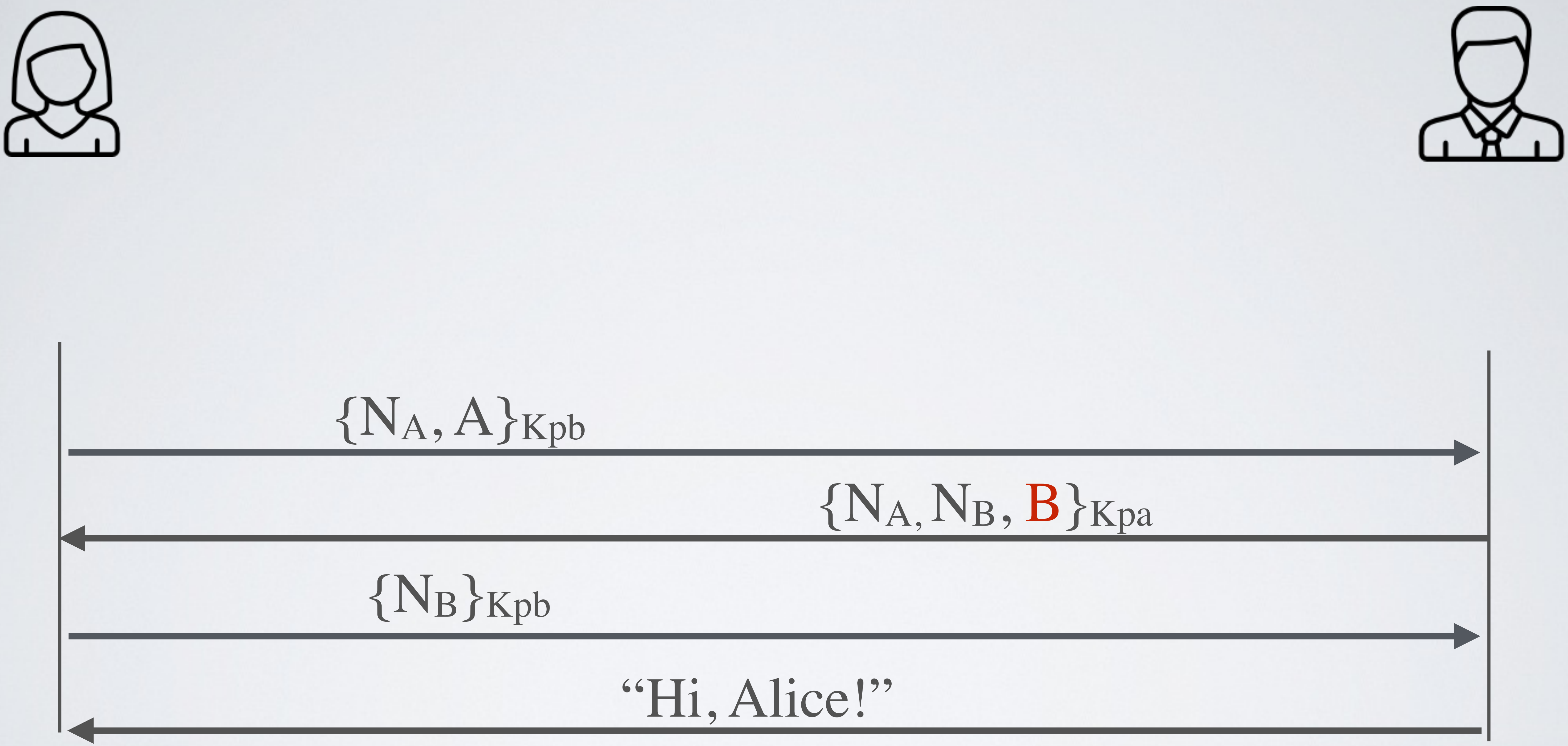
Simplified (but still vulnerable) version (1978)



Man-in-the-middle attack (Lowe's 1995)



Lowe's fix (1995)

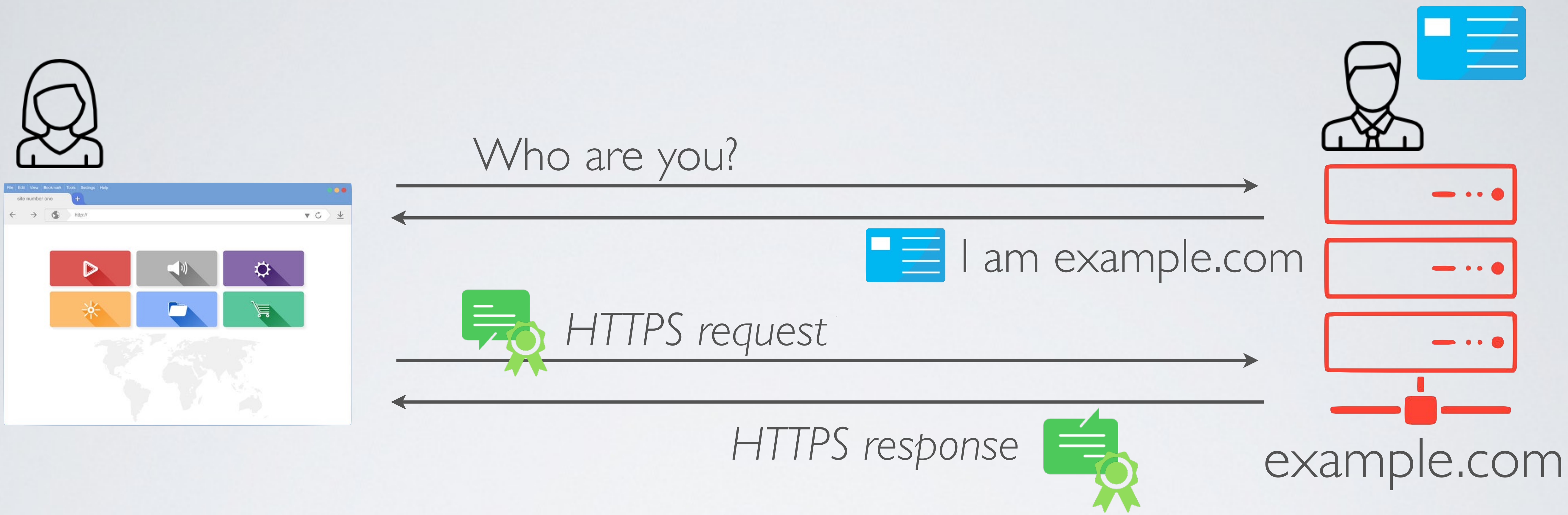


Not a perfect protocol yet

- ✓ Does authenticate Alice and bob
- ✓ Does prevent replay attacks
- ✓ Does ensure the authenticity of the public keys
- ◉ But the Public Key Server is a single point of failure

TLS - Transport Layer Security
a.k.a SSL - Secure Sockets Layer

This how HTTPS works

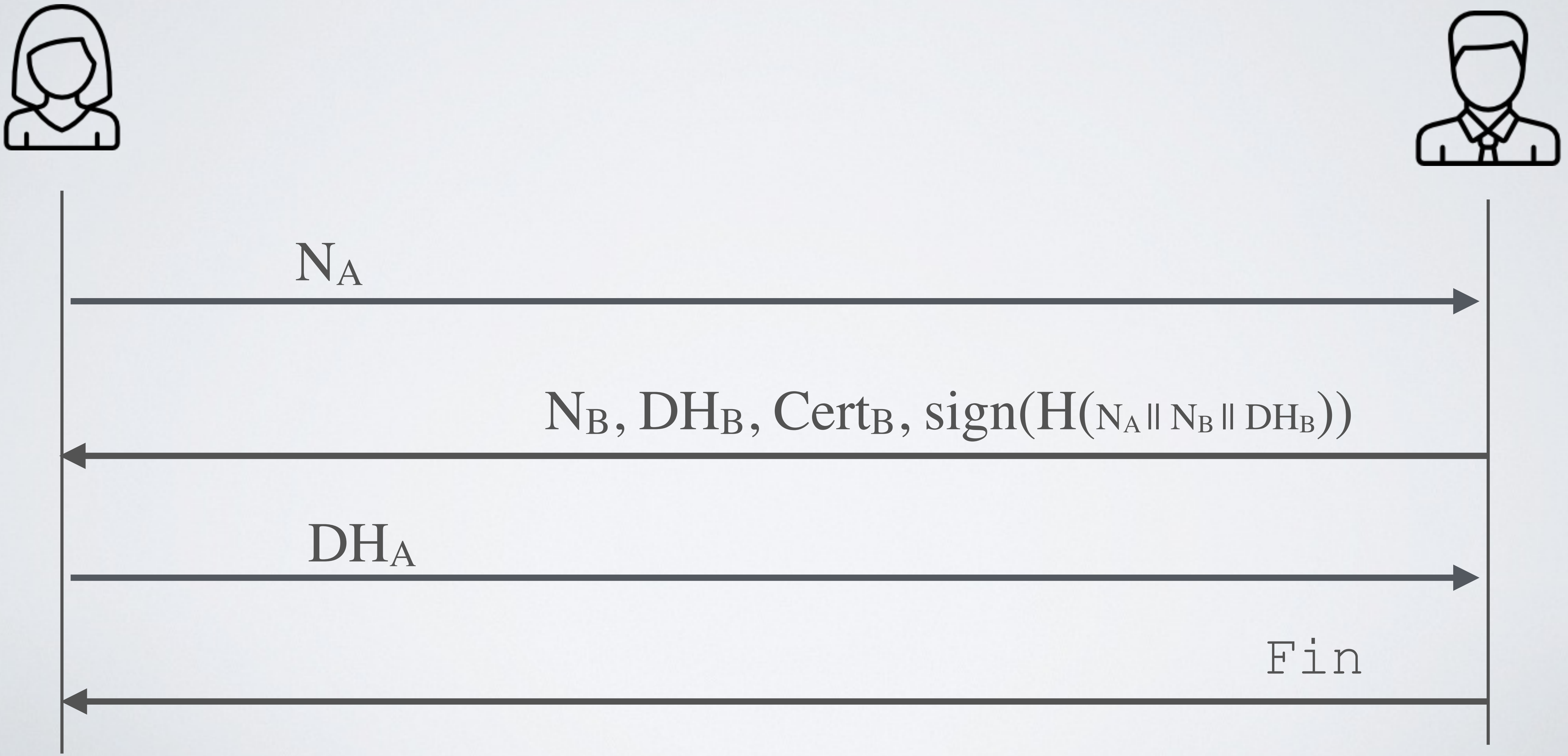


✓ **HTTPS = HTTP + TLS**

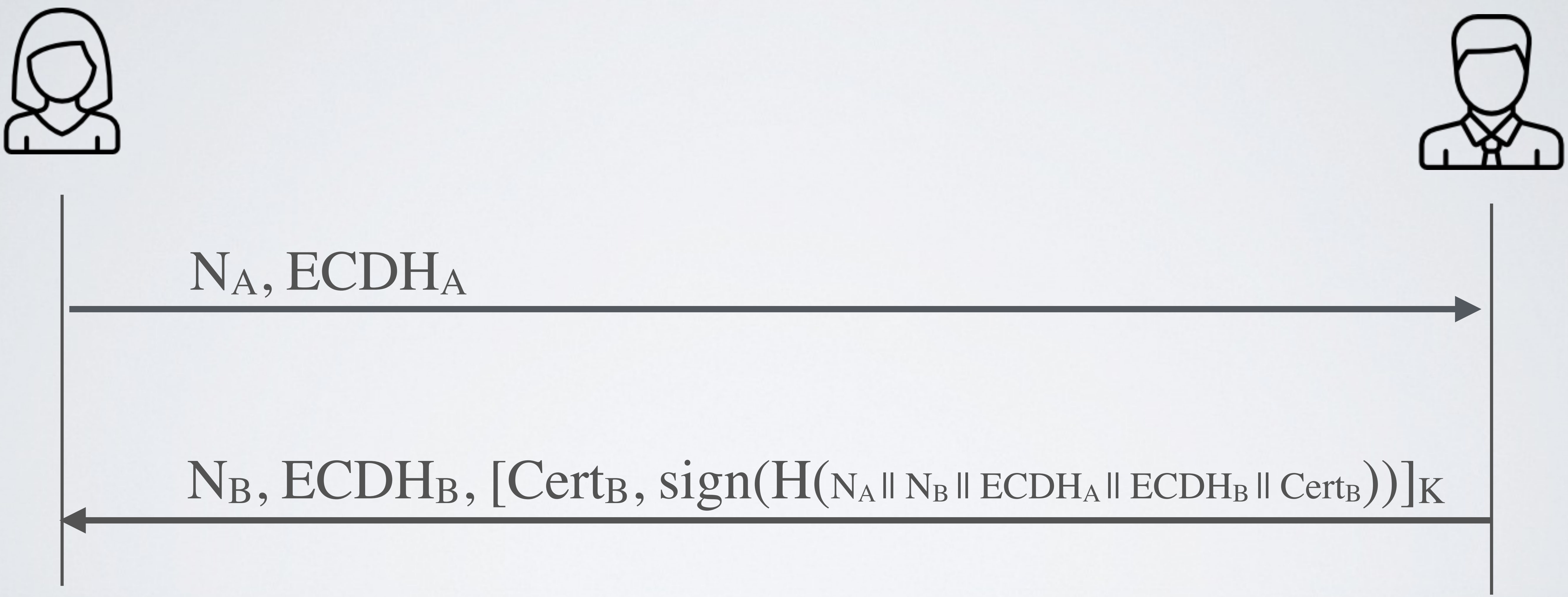
➡ TLS - Transport Layer Security (a.k.a SSL) provides

- **confidentiality :** end-to-end secure channel
- **integrity :** one-way authentication handshake

simplified and one-way authentication TLS 1.2 (2008)



simplified and one-way authentication
TLS 1.3 (2018)



TLS 1.3 is much better than TLS 1.2

✓ Only one round in the handshake (vs 2 with TLS 1.2)

✓ Faster (use of elliptic curves)

✓ Certificate is encrypted (better confidentiality)

✓ Protocol has been formally proven
(dos not prevent from implementation bugs)

Almost there ...

- ✓ Does ensure the confidentiality of the communication
- ✓ Does authenticate Alice and bob
- ✓ Does prevent replay attacks
- ➡ But how to ensure the authenticity of the public keys without using a Public Key Server ?

Trust Models

Two trust models

How to establish the authenticity of the binding between someone and its public key ?

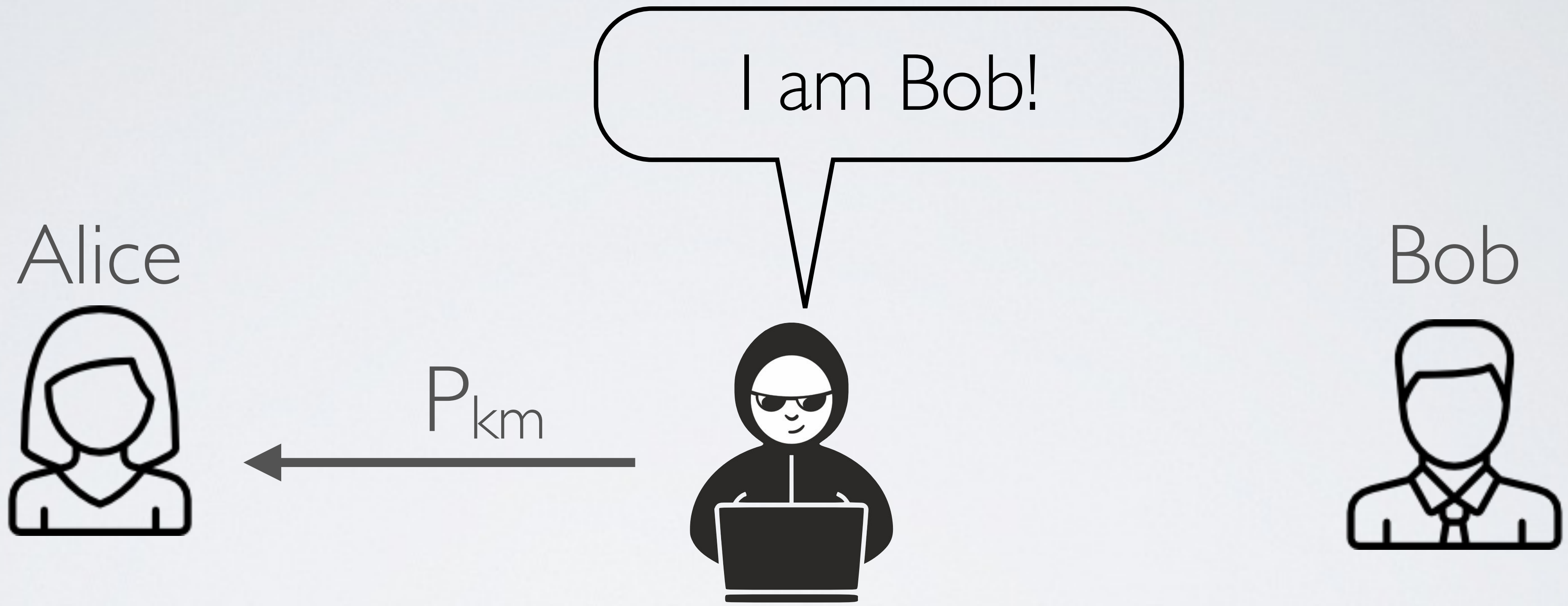
Decentralized trust model

➡ **Web of Trust**

Centralized trust model

➡ **PKI - Public Key Infrastructure**

Do you trust the GPG key ?

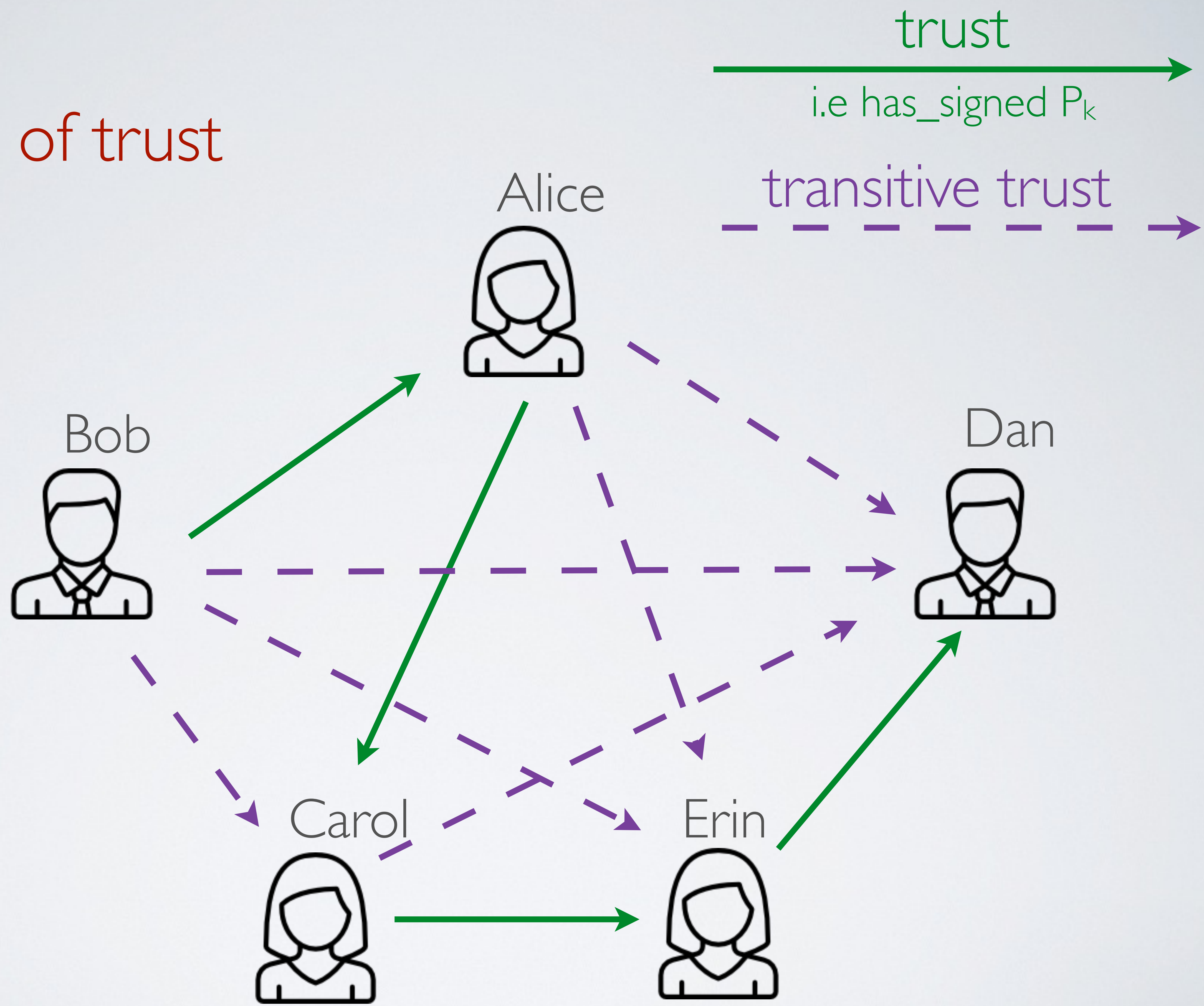


Alice should verify Bob's public key fingerprint

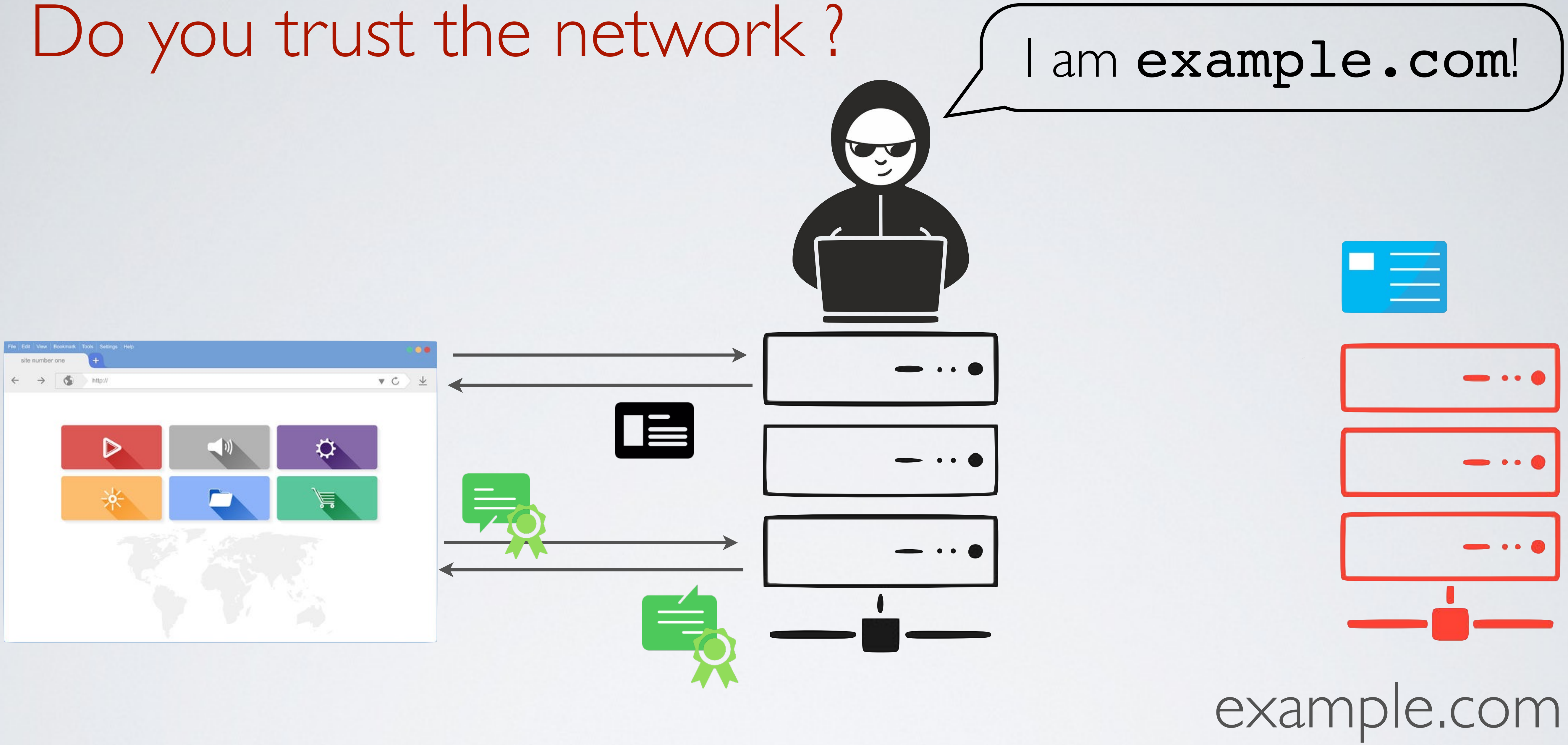
- either by communicating with Bob over another channel
- or by trusting someone that already trusts Bob

➡ **the web of trust**

The web of trust



Do you trust the network ?

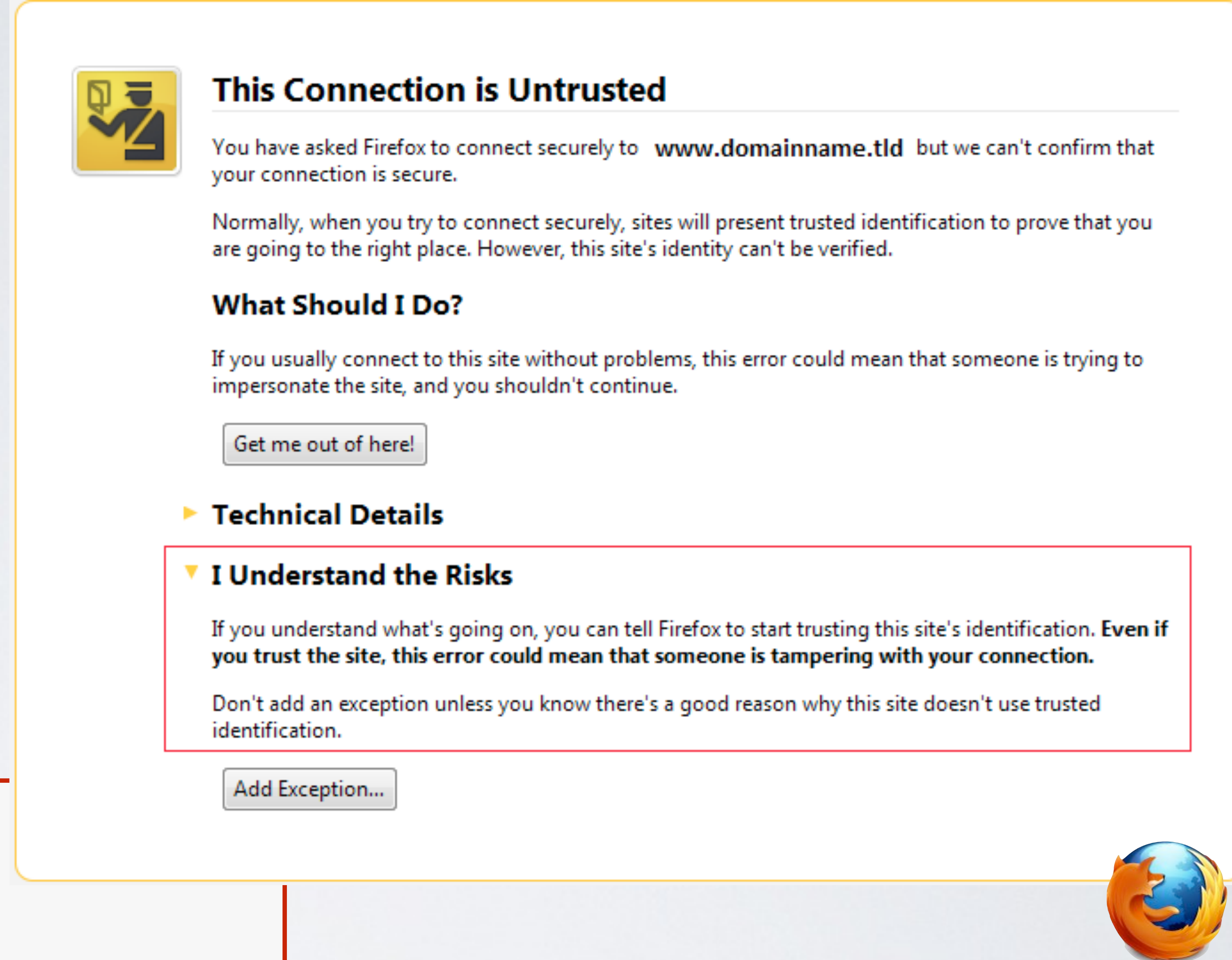


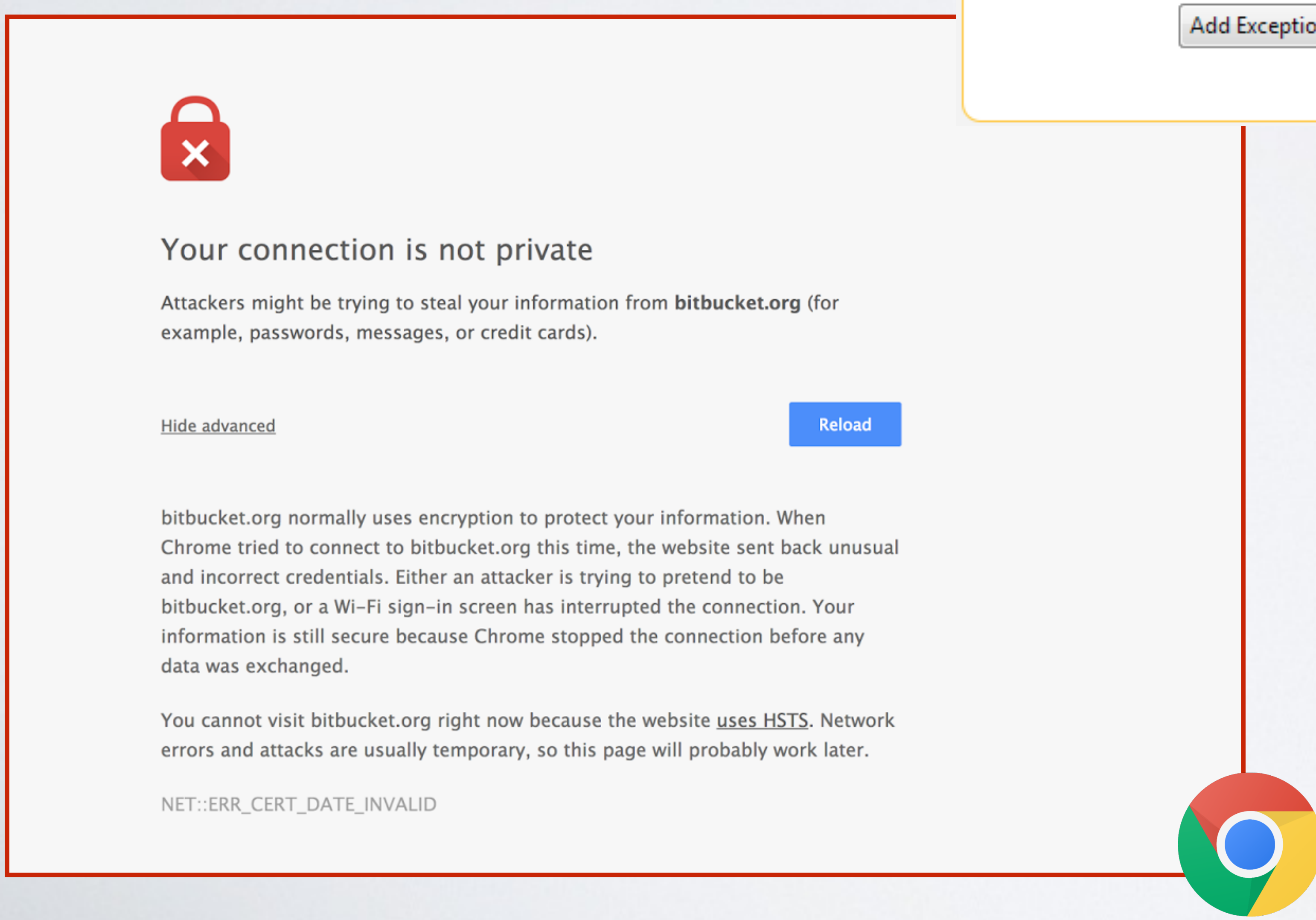
The browser should verify the certificate

➡ **PKI - Public Key Infrastructure**

Generating and using (self-signed) certificates

Self-signed certificates are not trusted by your browser

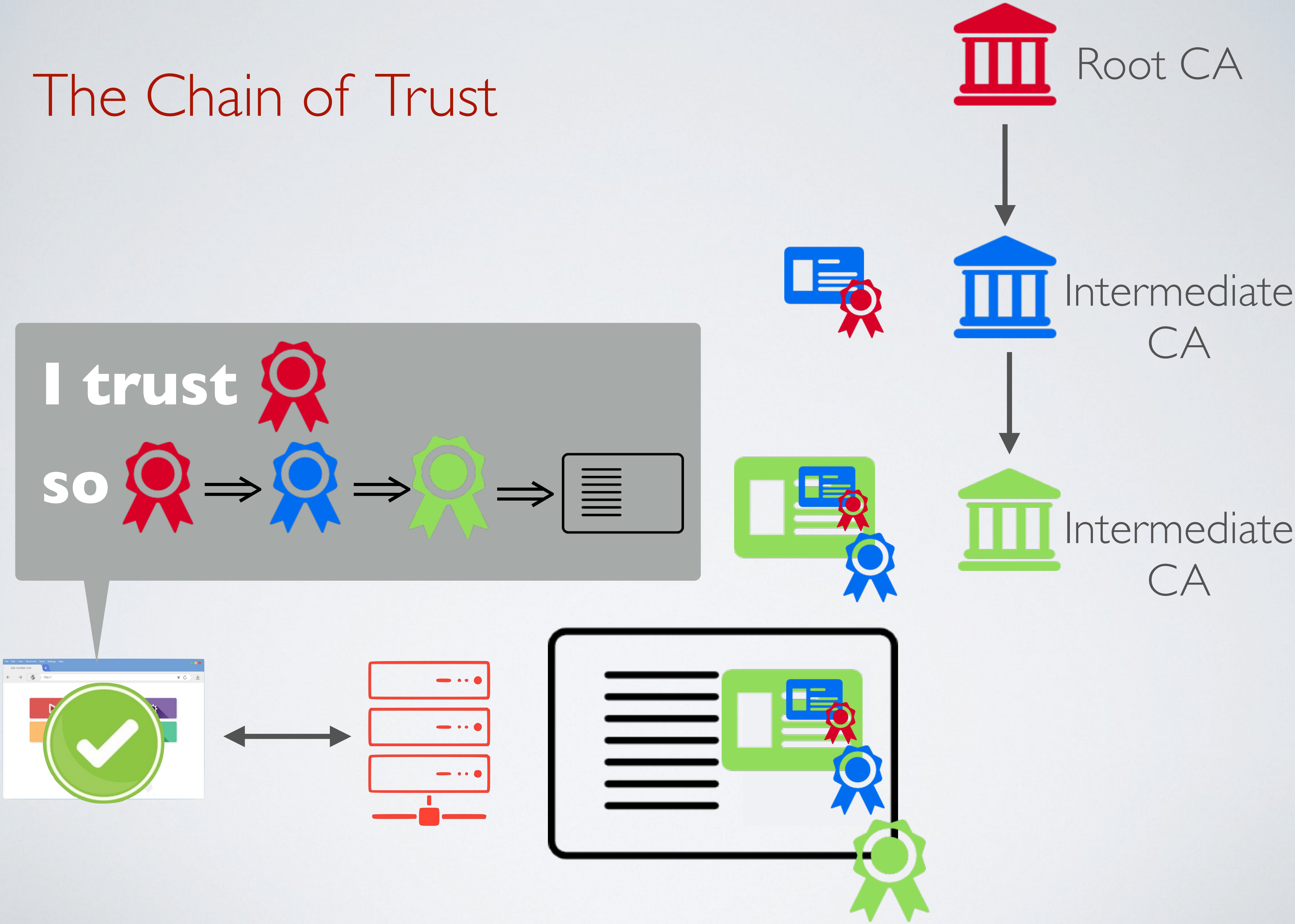




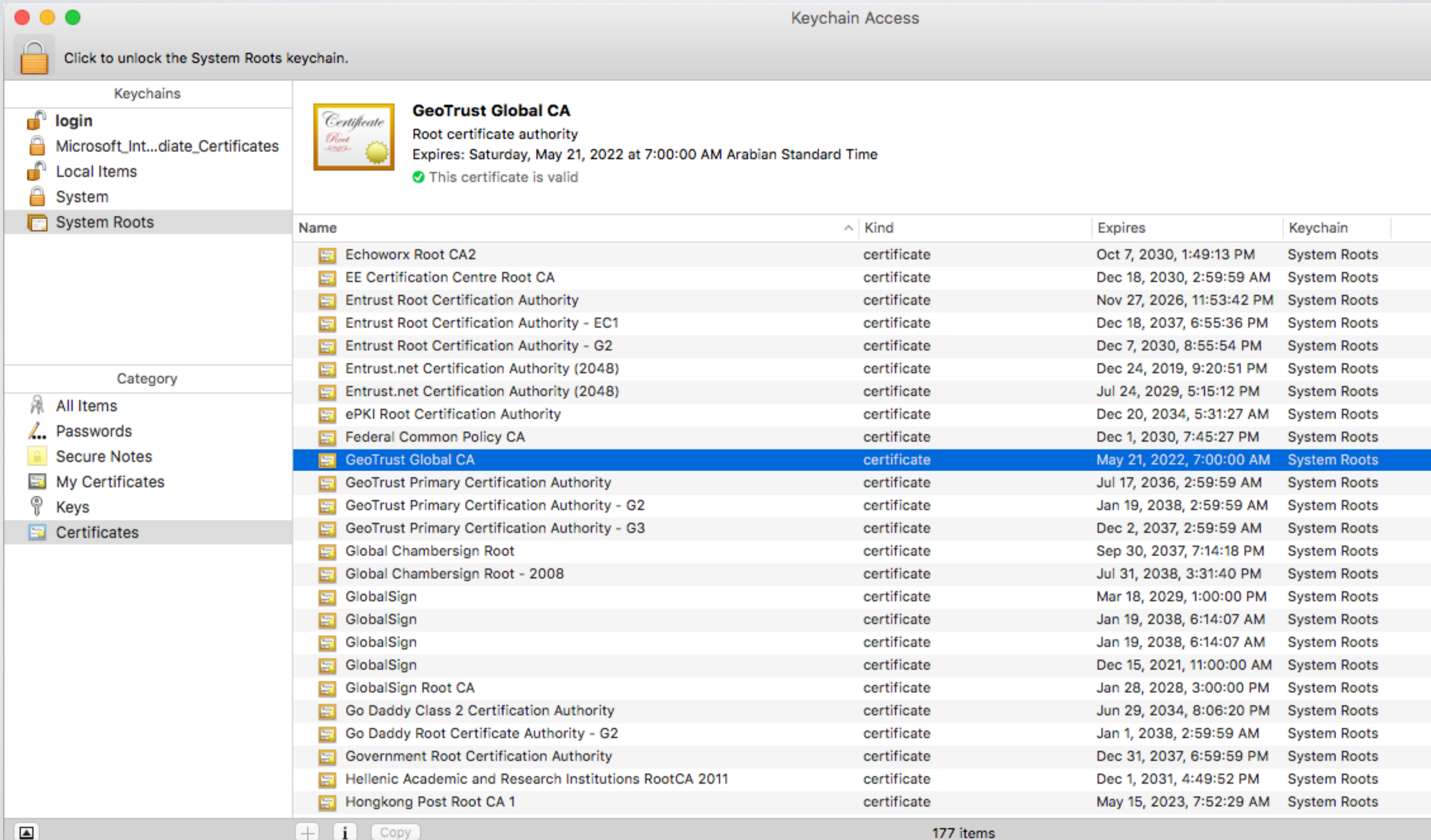
Signed Certificate

Certificate Authority (CA)

The Chain of Trust



Your browser trusts many root CAs **by default**



Real attacks

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

An update on attempted man-in-the-middle attacks

August 29, 2011

Posted by Heather Adkins, Information Security Manager

Today we received reports of attempted SSL man-in-the-middle (MITM) attacks against Google users, whereby someone tried to get between them and encrypted Google services. The people affected were primarily located in Iran. The attacker used a fraudulent SSL certificate issued by DigiNotar, a root certificate authority that should not issue certificates for Google (and has since revoked it).

Google Chrome users were protected from this attack because Chrome was able to detect the fraudulent certificate.

Google Security Blog

The latest news and insights from Google on security and safety on

Enhancing digital certificate security

January 3, 2013

Posted by Adam Langley, Software Engineer

Late on December 24, Chrome detected and blocked an unauthorized digital certificate for the "*.google.com" domain. We investigated immediately and found the certificate was issued by an intermediate certificate authority (CA) linking back to TURKTRUST, a Turkish certificate authority. Intermediate CA certificates carry the full authority of the CA, so anyone who has one can use it to create a certificate for any website they wish to impersonate.

Real attacks

threatpost | Podcasts / Malware / Vulnerabilities / InfoSec Insiders / Webinars

Study: Password Security Improves with Age | Researchers Find Methods fo

Flame Malware Uses Forged Microsoft Certificate to Validate Components

Author:
Dennis Fisher
June 4, 2012 / 12:00 pm

2 minute read

Share this article:

Microsoft has found that some components of the Flame malware were signed using a forged digital certificate that the attackers were able to create by exploiting a weakness in the way that Microsoft's Terminal Services allows customers to sign code with Microsoft certificates. The company has sent out an update that will remove three untrusted certificates from the Microsoft Trusted Certificate Store and has made a change to the way Terminal Services handles code signing.

Microsoft Security Response Center | Report an iss

Flame malware collision attack explained

Security Research & Defense / By swiat / June 6, 2012 / malware, PKI

Since our last MSRC blog post, we've received questions on the nature of the cryptographic attack we saw in the complex, targeted malware known as Flame. This blog summarizes what our research revealed and why we made the decision to release Security Advisory 2718704 on Sunday night PDT. In short, by default the attacker's certificate would not work on Windows Vista or more recent versions of Windows. They had to perform a collision attack to forge a certificate that would be valid for code signing on Windows Vista or more recent versions of Windows. On systems that pre-date Windows Vista, an attack is possible without an MD5 hash collision. This certificate and all certificates from the involved certificate authorities were invalidated in Security Advisory 2718704. We continue to encourage all customers who are not installing updates automatically to do so immediately.

Mysterious Missing Extensions

Limitation of secure channels