

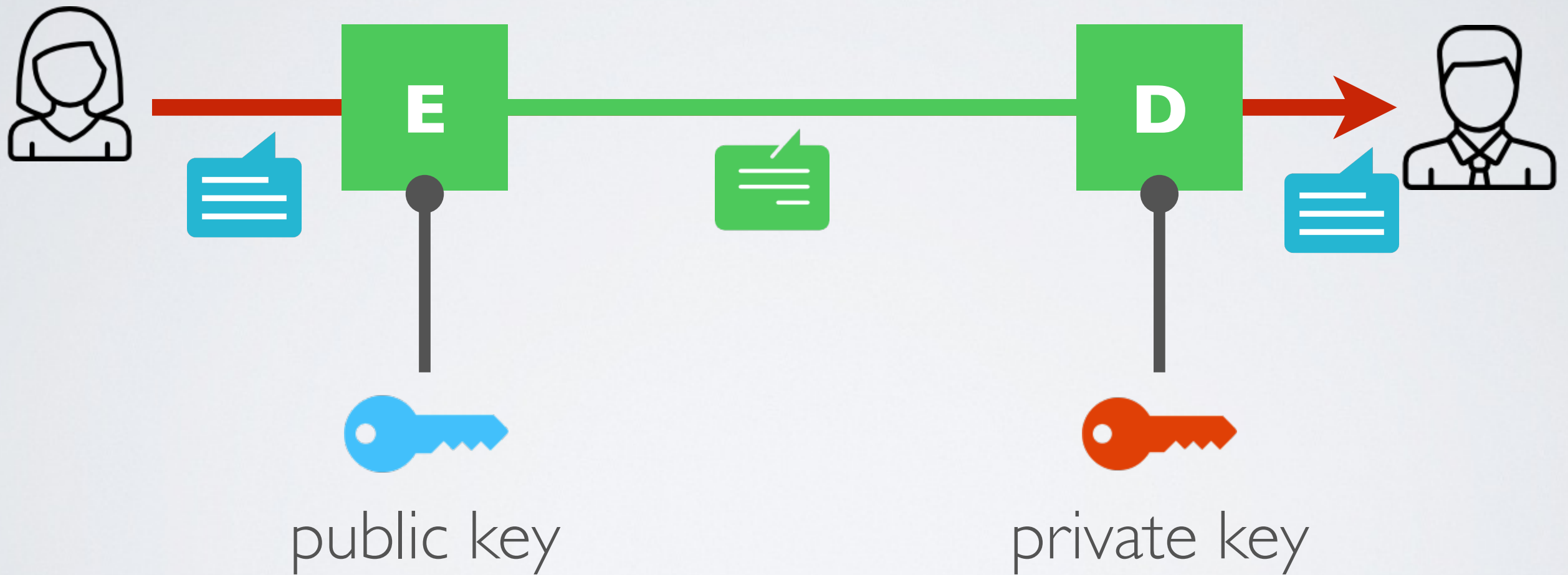
Applied Asymmetric Cryptography

Protocols, Attacks, Implementation Flaws

Kc Udonsi

Refresher

Asymmetric encryption a.k.a Public Key Cryptography



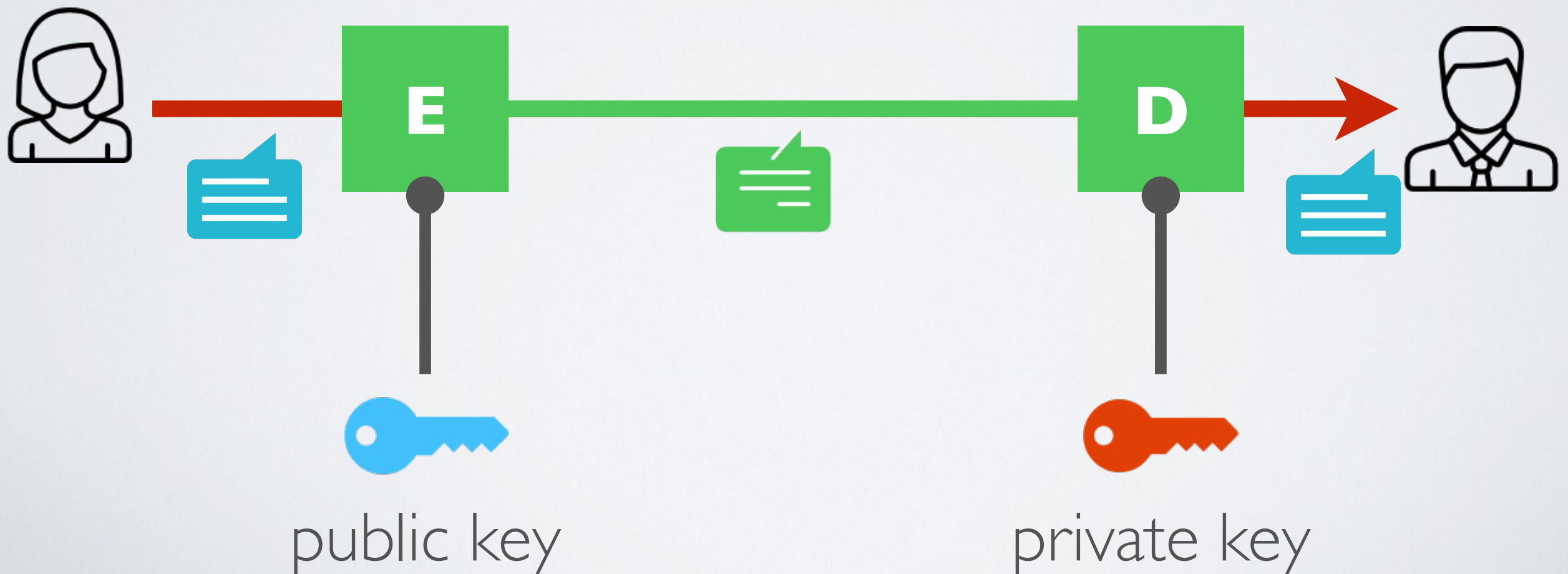
Asymmetric Keys - Functional Requirements

$D_{K_s}(E_{K_p}(m)) = m$ and $D_{K_p}(E_{K_s}(m)) = m$ for every pair (K_p, K_s)

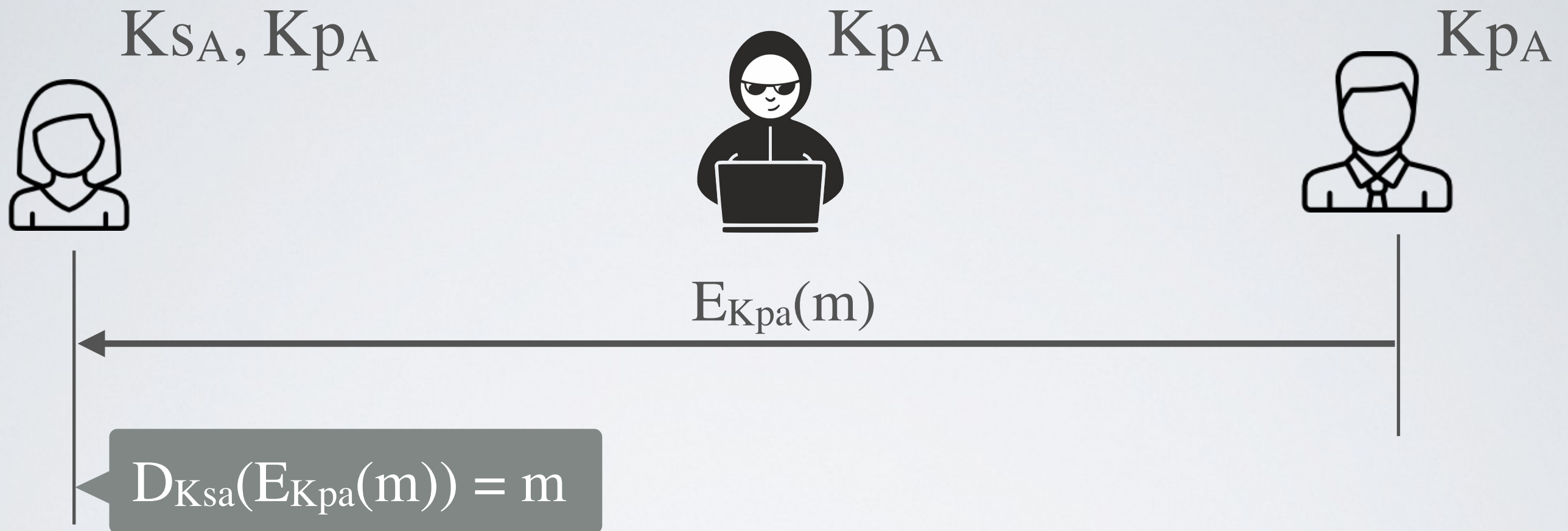
- ✓ Generating a pair (K_p, K_s) is easy to compute (polynomial)
- ✓ Encryption is easy to compute (either polynomial or linear)
- ✓ Decryption is easy to compute (either polynomial or linear)
- Finding a matching key K_s for a given K_p is hard (exponential)
- Decryption without knowing the corresponding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



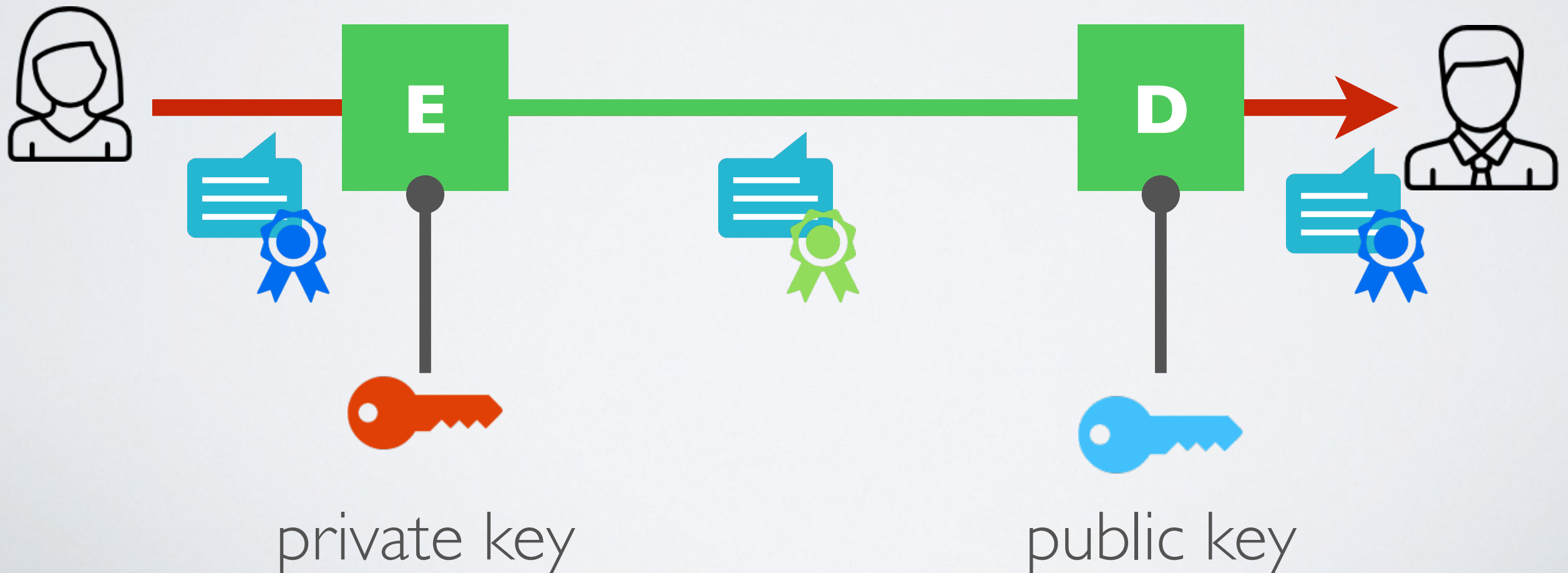
Bob encrypts a message m with Alice's public key K_{PA}

➔ Nobody can decrypt m , except Alice with her private key K_{SA}

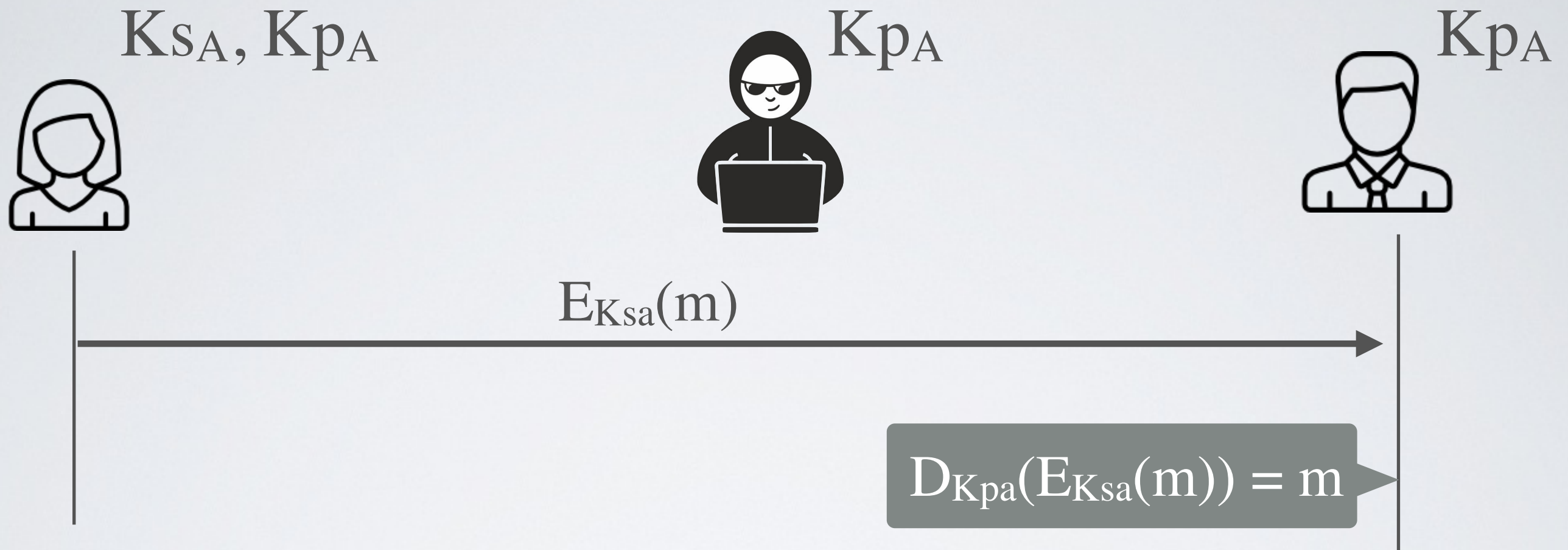
✓ Confidentiality without the need to exchange a secret key

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 K_{sA}

➔ Everybody can decrypt m using Alice's public key K_{pA}

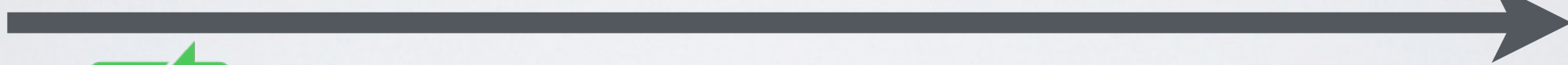
✓ Authentication with non-repudiation (a.k.a Digital Signature)

Digital Signature

K_{sa} Alice's Secret Key



K_{pa}, K_{pb} public keys



K_{sb}







➔ Use public cryptography to **sign and verify**

$$m \parallel \text{SIG}_{K_{sa}}(m)$$

$$\text{SIG}_{K_{sa}}(m) = E_{K_{sa}}(H(m))$$

Non-repudiation as a special case of integrity

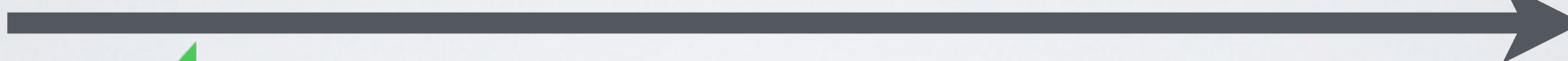
	MAC	Digital Signature
Integrity		
Non-repudiation		

Digital Signatures and Confidentiality

K_{sa} Alice's Secret Key



K_{pa}, K_{pb} public keys



K_{sb}

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

$$E_{K_{pb}}(k) \parallel E_k(m \parallel E_{K_{sa}}(H(m)))$$

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

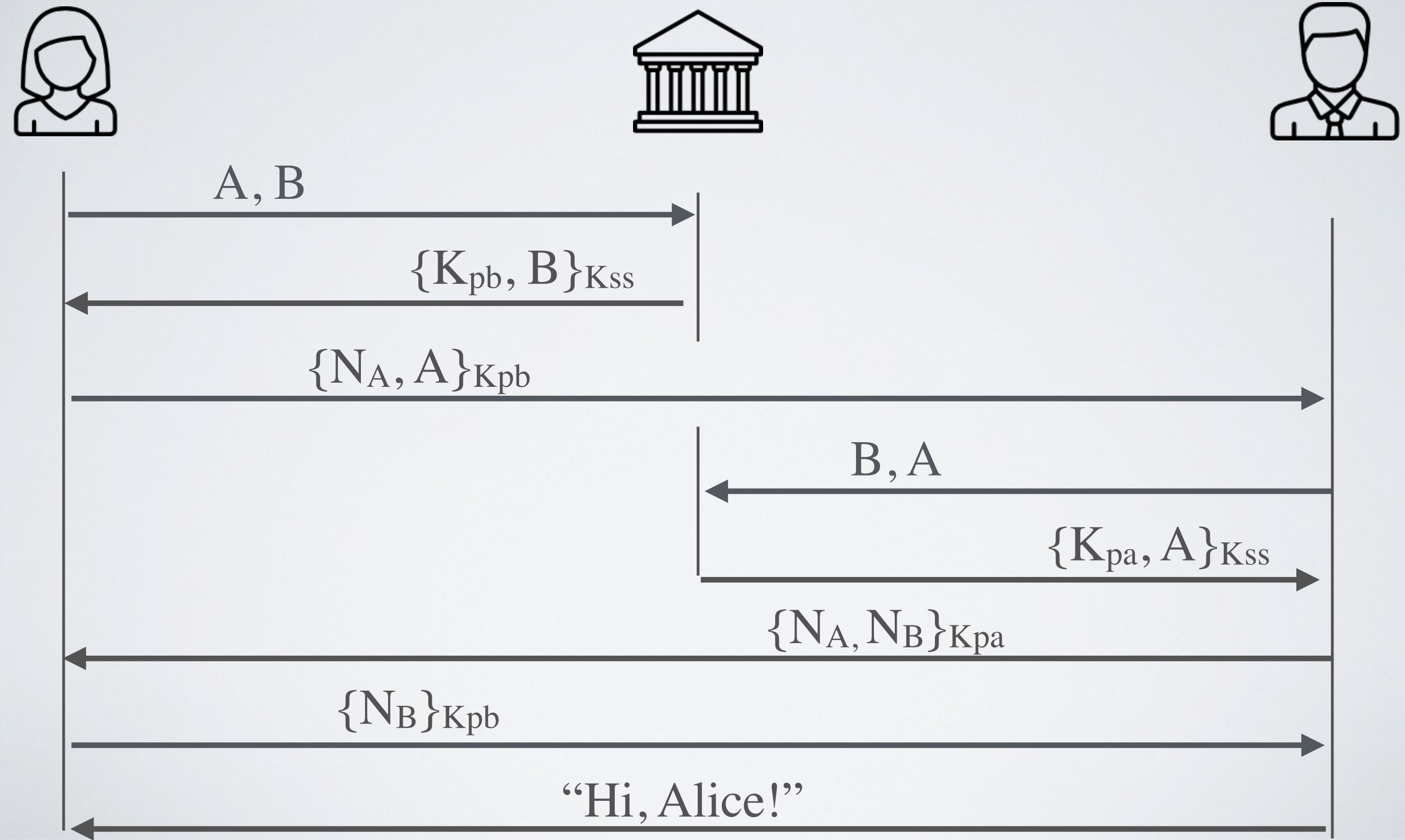
Assumptions

- 4 principals : Alice, Bob, Mallory and a Public-Key Server
- Alice, Bob, Mallory and the Server have generated their own public/private key pair
- Alice, Bob and Mallory know the Server's public key K_{ps}
- A, B, M and S talk to each other using the same protocol

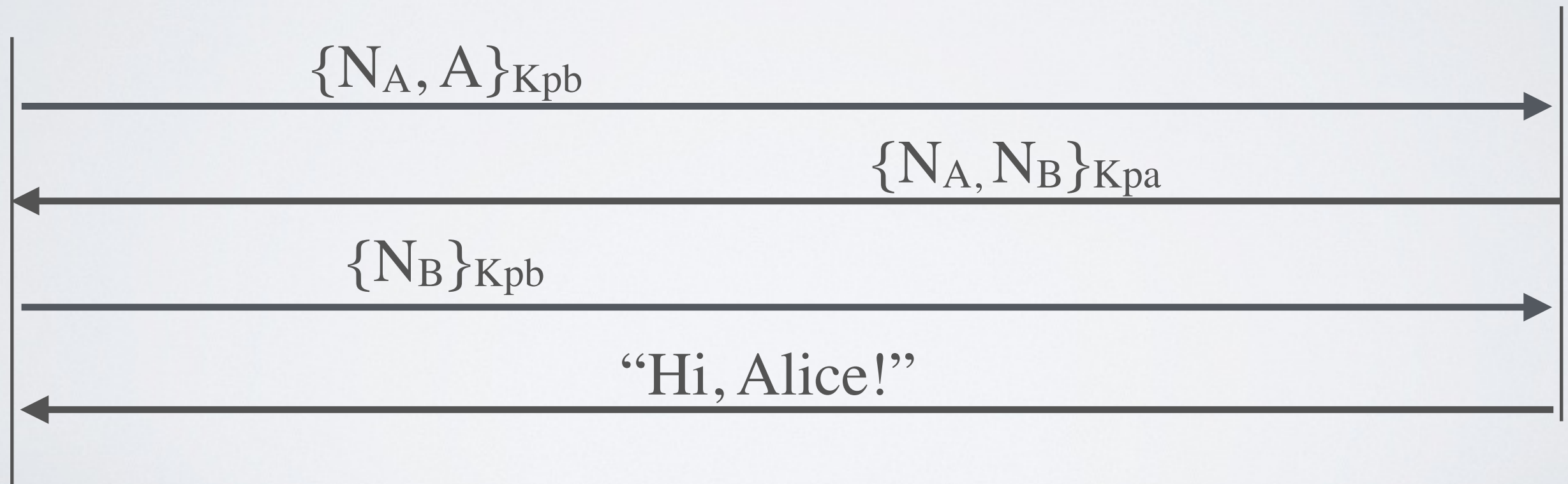
Goals

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

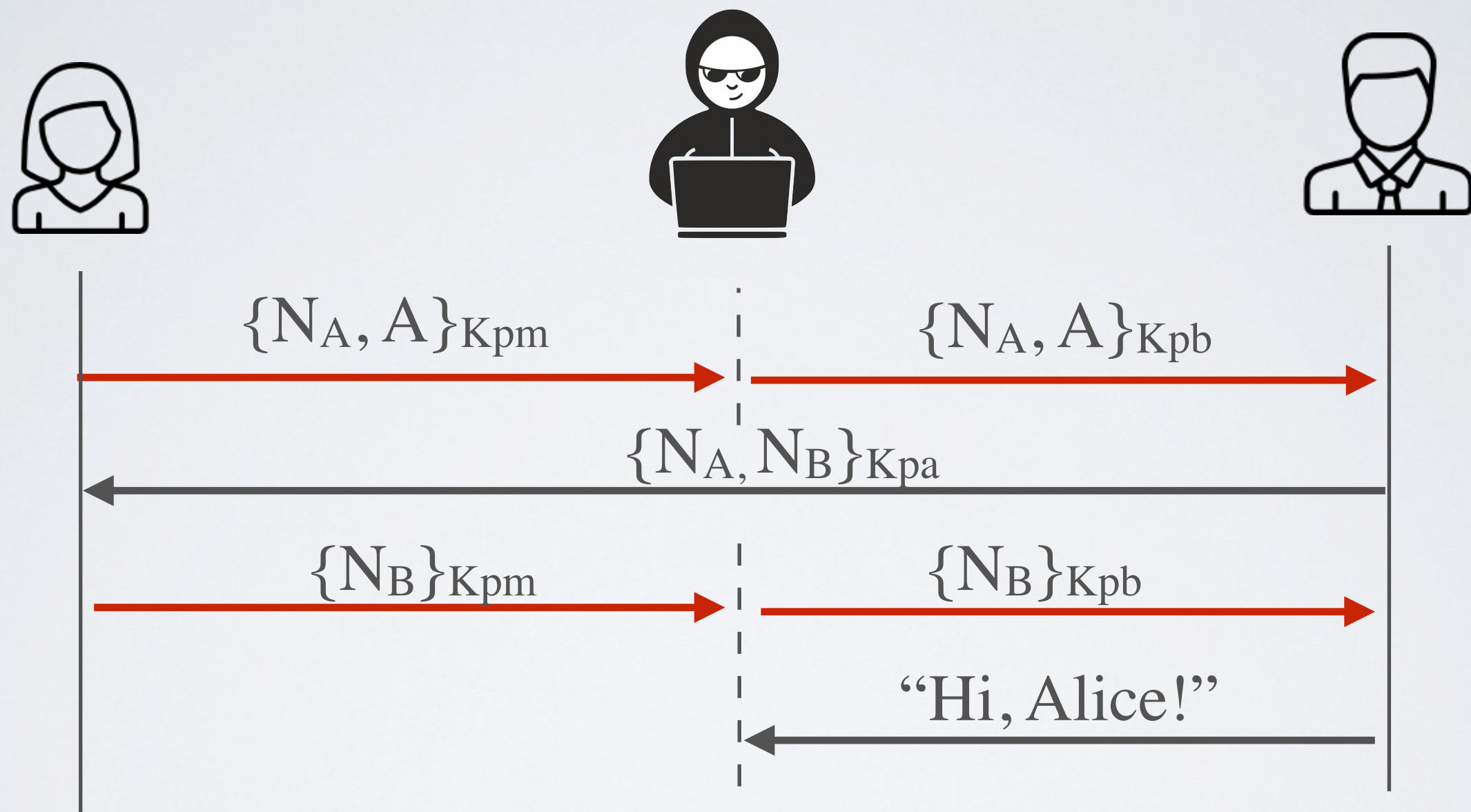
The vulnerable version (1978)



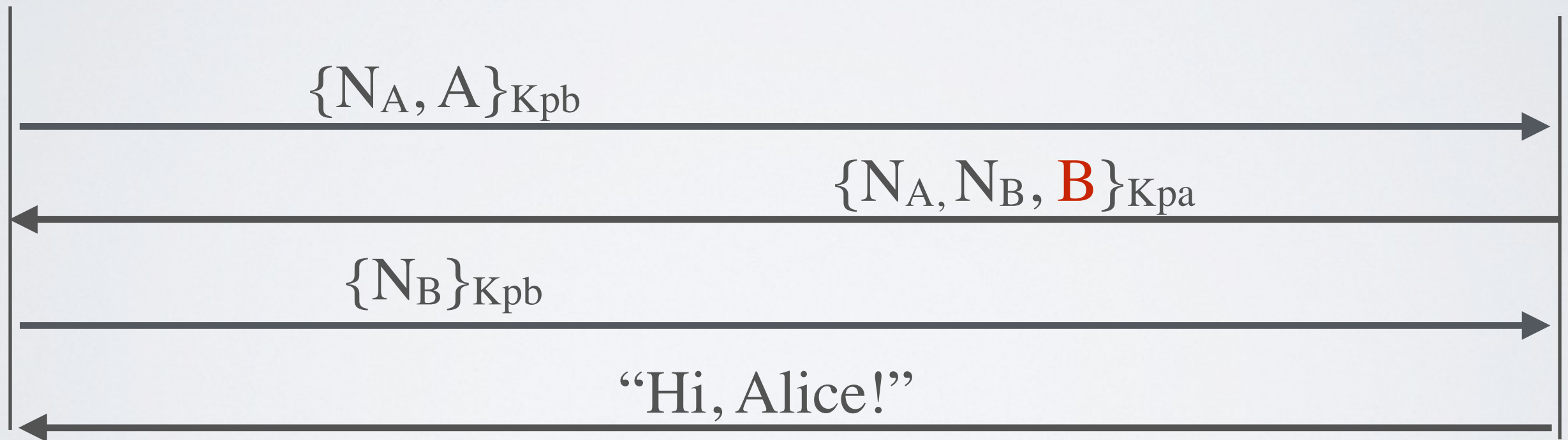
Simplified (but still vulnerable) version (1978)



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



Lowe's fix (1995)

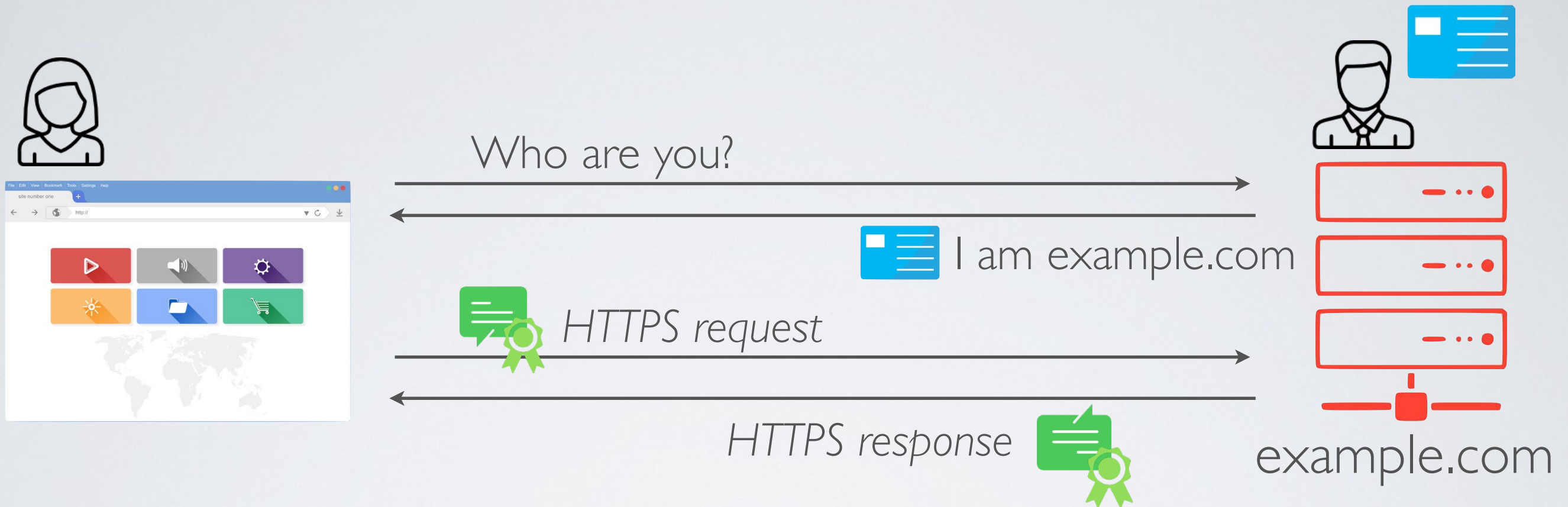


Not a perfect protocol yet

- ✓ 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

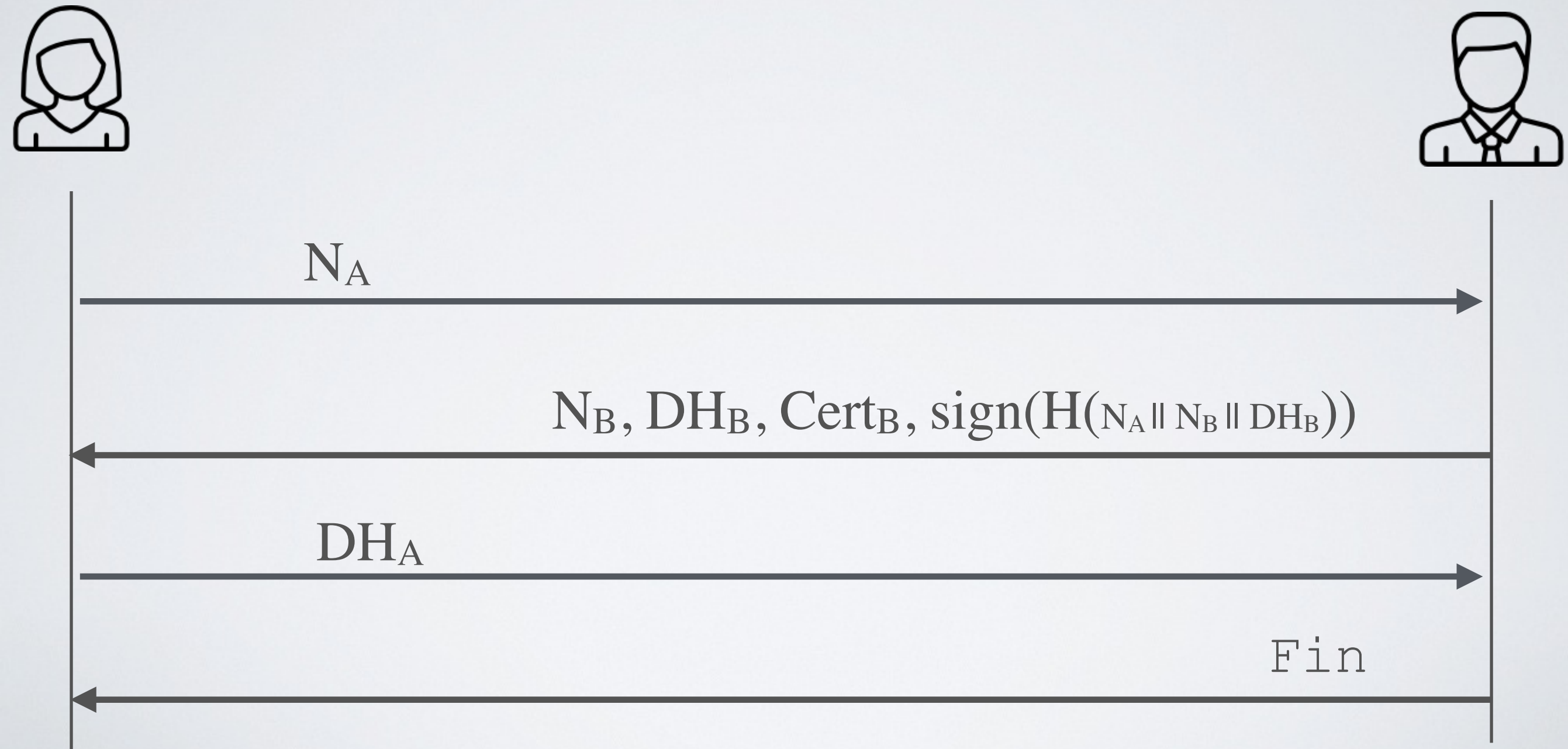


✓ **HTTPS = HTTP + TLS**

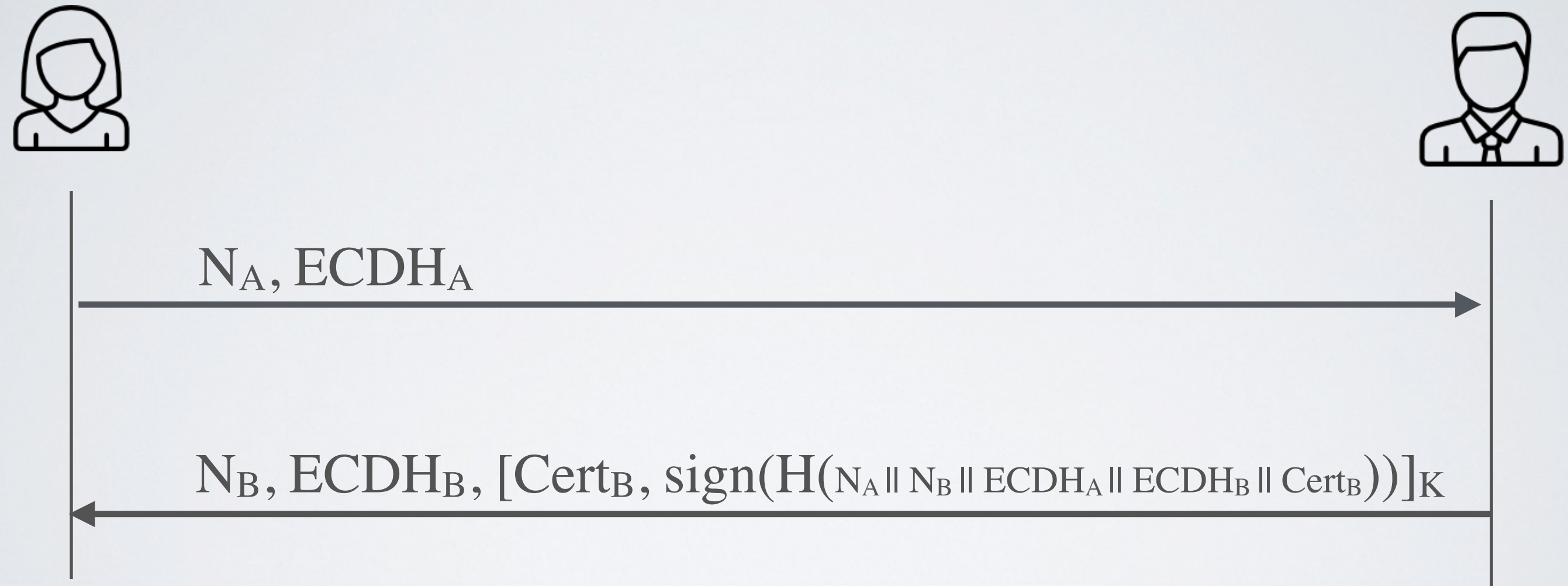
➔ TLS - Transport Layer Security (a.k.a SSL) provides

- **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
(does 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**

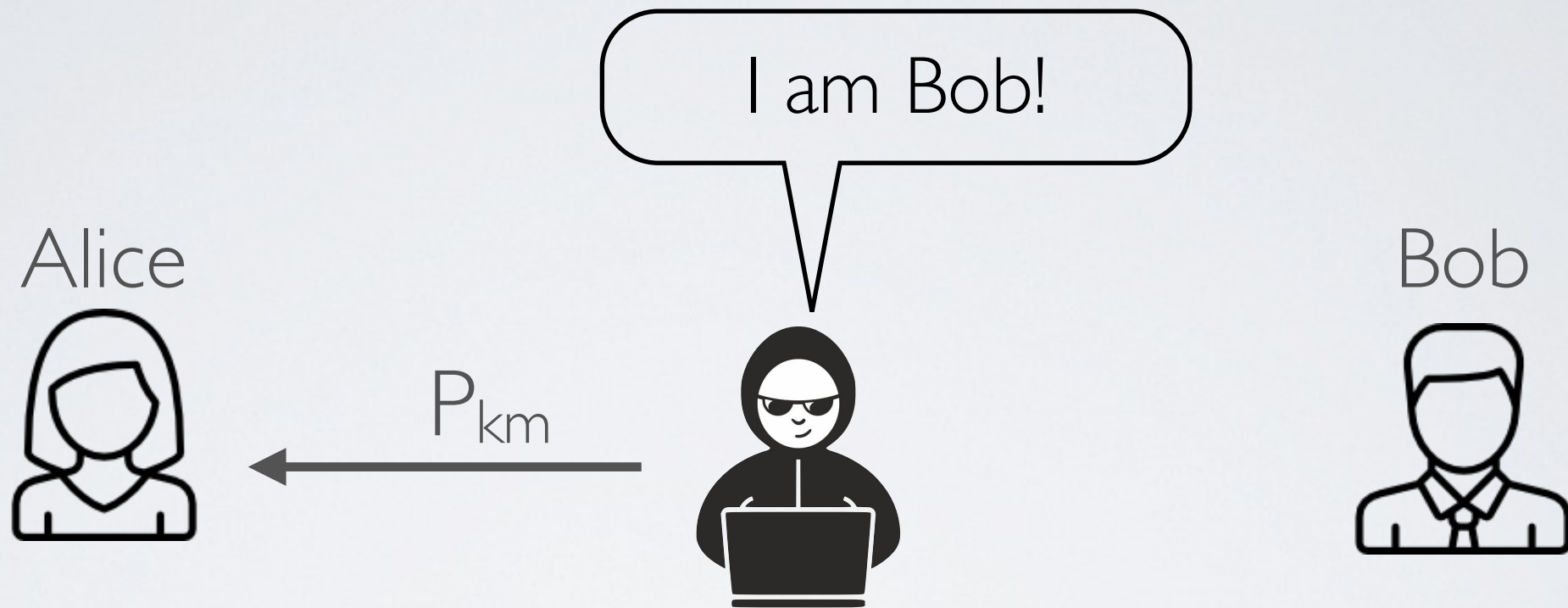


Centralized trust model

➔ **PKI - Public Key Infrastructure**



Do you trust the GPG key ?

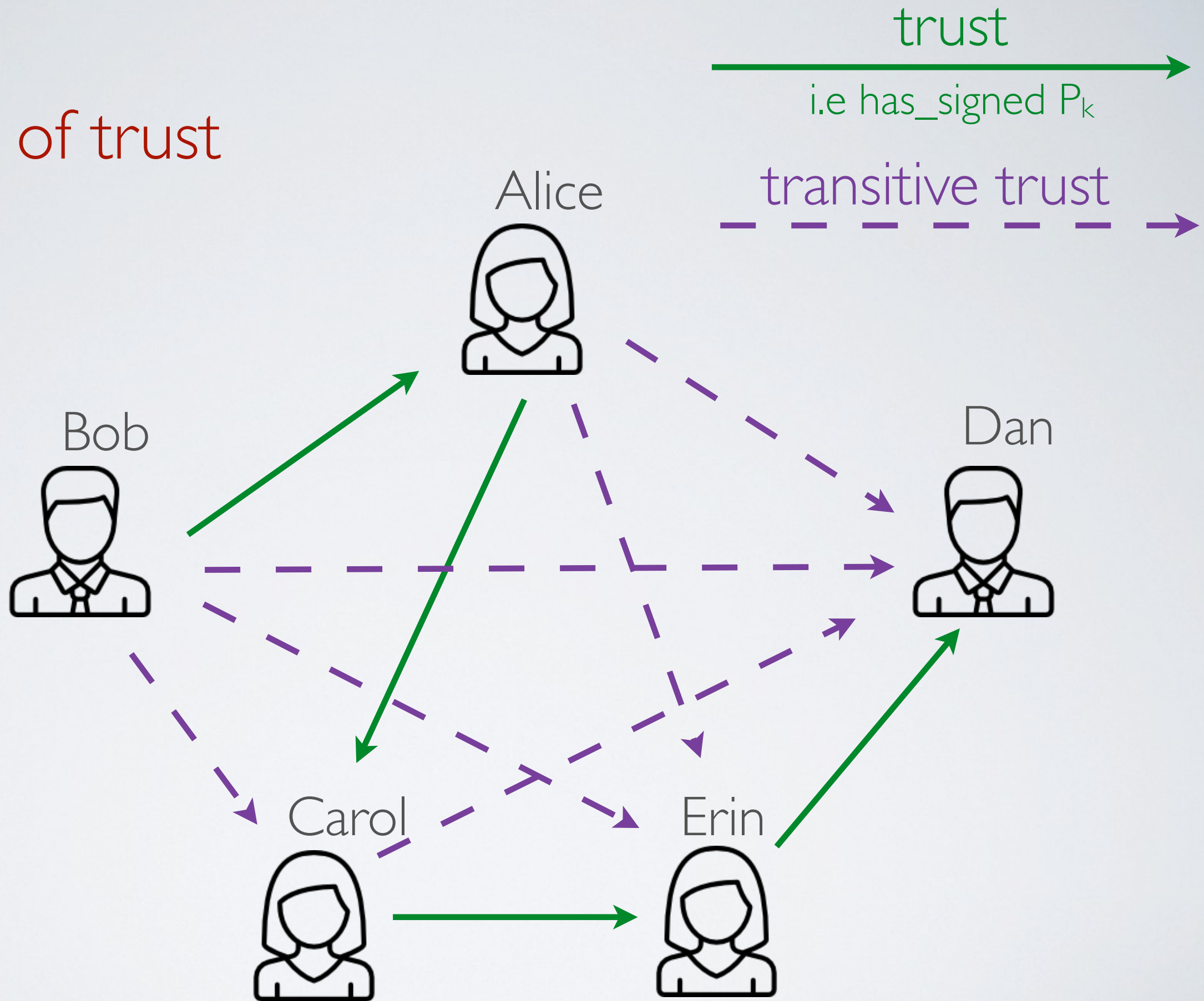


Alice should verify Bob's public key fingerprint

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

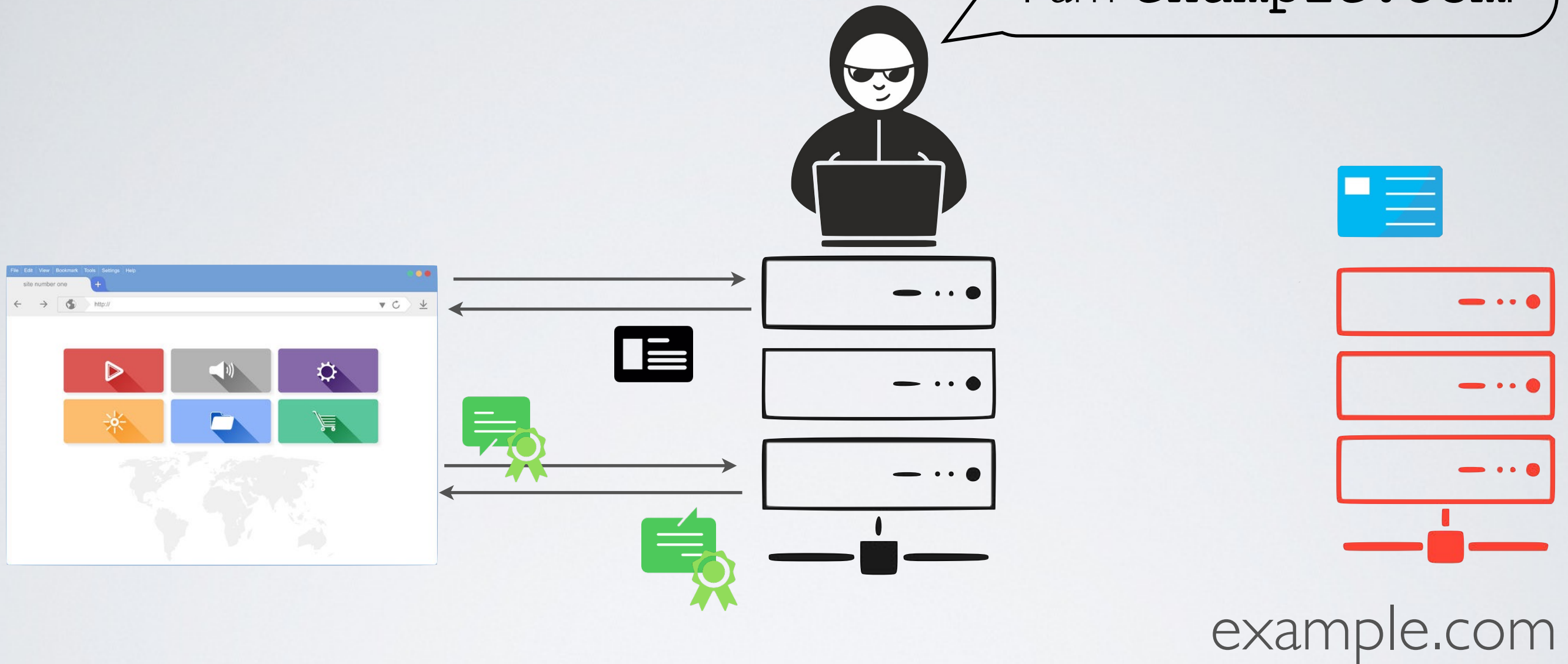
➔ **the web of trust**

The web of trust



Do you trust the network ?

I am `example.com`!

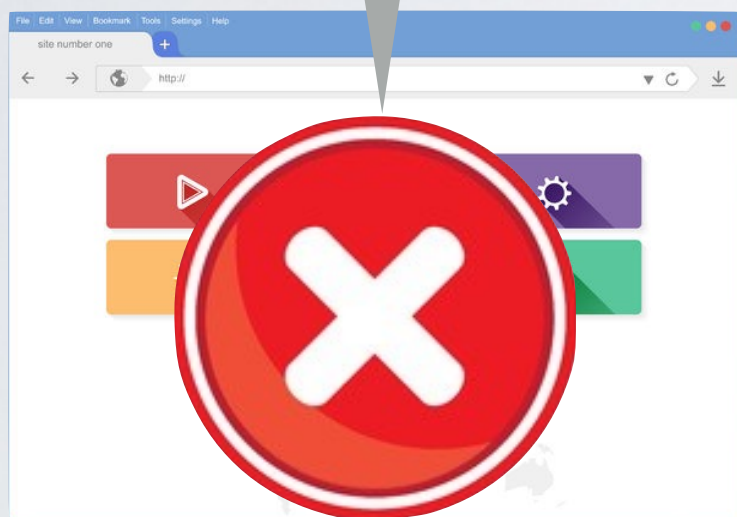


The browser should verify the certificate

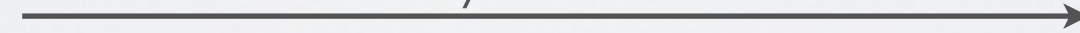
➔ **PKI - Public Key Infrastructure**

Generating and using (self-signed) certificates

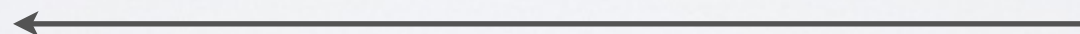
I don't know 



Who are you?



I am example.com



Self-signed certificates are not trusted by your browser



Your connection is not private

Attackers might be trying to steal your information from **bitbucket.org** (for example, passwords, messages, or credit cards).

[Hide advanced](#)

[Reload](#)

bitbucket.org normally uses encryption to protect your information. When Chrome tried to connect to bitbucket.org this time, the website sent back unusual and incorrect credentials. Either an attacker is trying to pretend to be bitbucket.org, or a Wi-Fi sign-in screen has interrupted the connection. Your information is still secure because Chrome stopped the connection before any data was exchanged.

You cannot visit bitbucket.org right now because the website [uses HSTS](#). Network errors and attacks are usually temporary, so this page will probably work later.

NET::ERR_CERT_DATE_INVALID



This Connection is Untrusted

You have asked Firefox to connect securely to **www.domainname.tld** but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

[Get me out of here!](#)

▶ Technical Details

▼ I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

[Add Exception...](#)



Signed Certificate


Certificate Authority (CA)

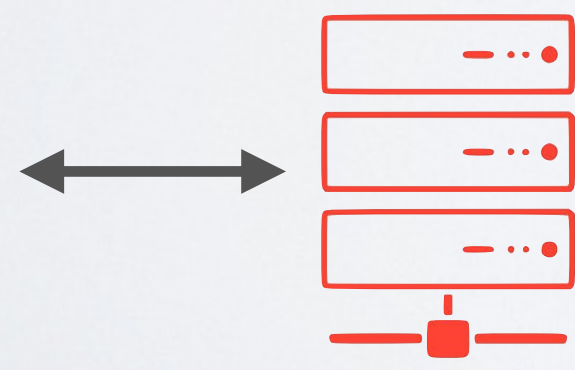
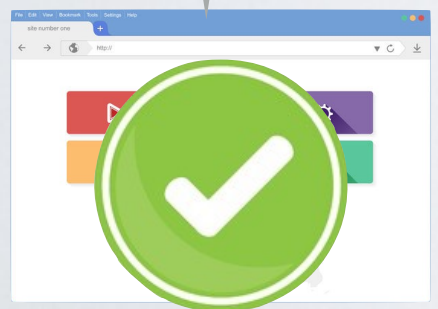
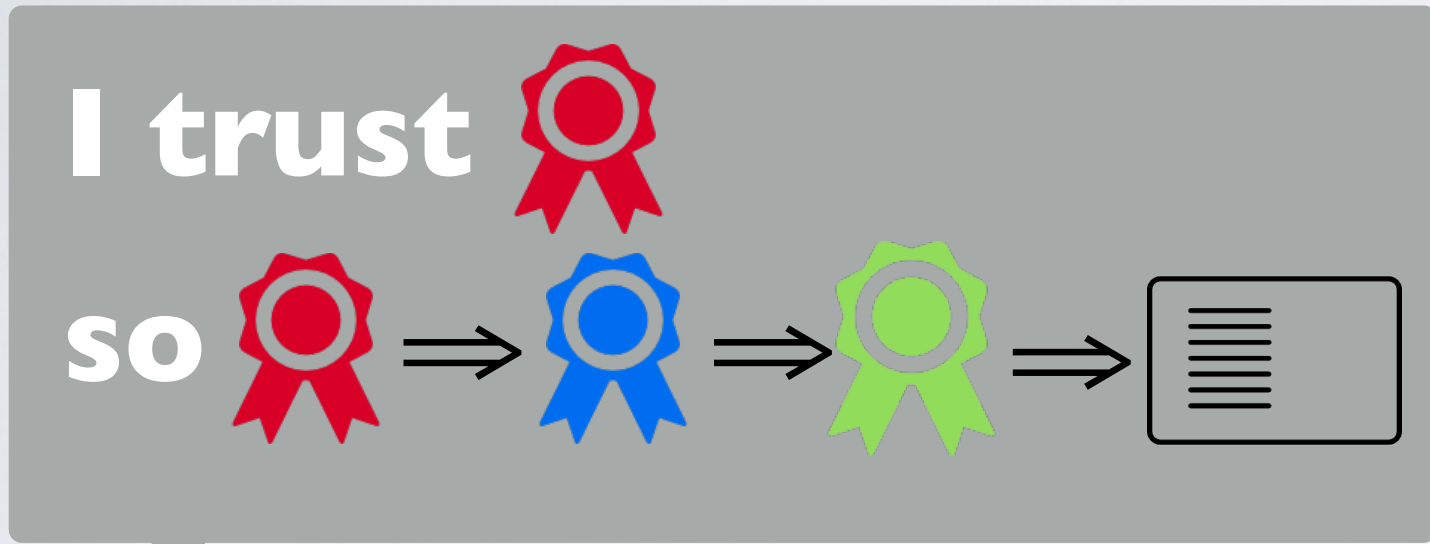


The Chain of Trust

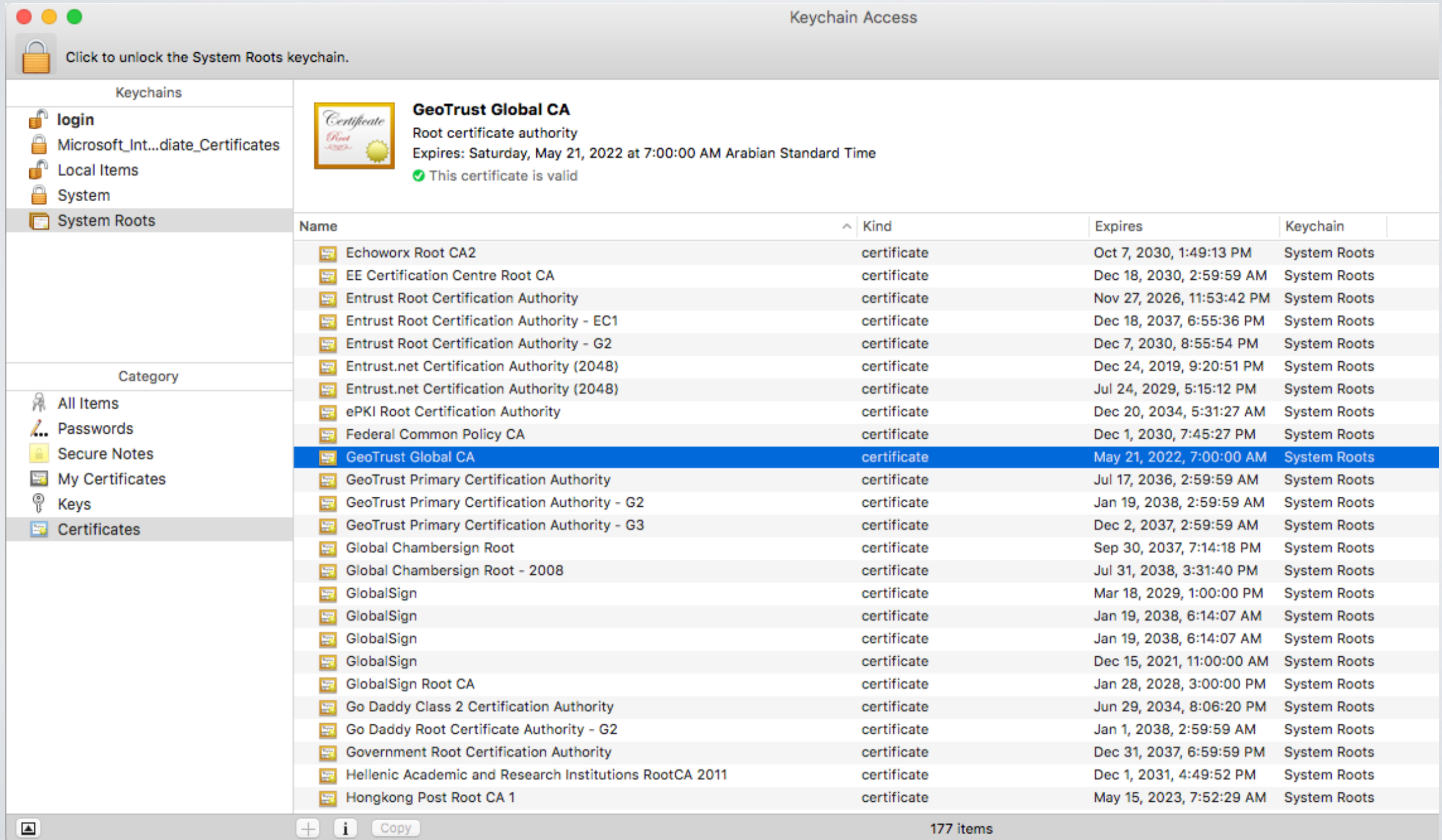
 Root CA

 Intermediate CA

 Intermediate CA



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



The screenshot shows the macOS Keychain Access application. The 'System Roots' keychain is selected, displaying a list of 177 root certificates. The 'GeoTrust Global CA' certificate is highlighted in blue. A detailed view of this certificate is shown above the table, indicating it is valid and expires on May 21, 2022.

GeoTrust Global CA
Root certificate authority
Expires: Saturday, May 21, 2022 at 7:00:00 AM Arabian Standard Time
This certificate is valid

Name	Kind	Expires	Keychain
Echoworx Root CA2	certificate	Oct 7, 2030, 1:49:13 PM	System Roots
EE Certification Centre Root CA	certificate	Dec 18, 2030, 2:59:59 AM	System Roots
Entrust Root Certification Authority	certificate	Nov 27, 2026, 11:53:42 PM	System Roots
Entrust Root Certification Authority - EC1	certificate	Dec 18, 2037, 6:55:36 PM	System Roots
Entrust Root Certification Authority - G2	certificate	Dec 7, 2030, 8:55:54 PM	System Roots
Entrust.net Certification Authority (2048)	certificate	Dec 24, 2019, 9:20:51 PM	System Roots
Entrust.net Certification Authority (2048)	certificate	Jul 24, 2029, 5:15:12 PM	System Roots
ePKI Root Certification Authority	certificate	Dec 20, 2034, 5:31:27 AM	System Roots
Federal Common Policy CA	certificate	Dec 1, 2030, 7:45:27 PM	System Roots
GeoTrust Global CA	certificate	May 21, 2022, 7:00:00 AM	System Roots
GeoTrust Primary Certification Authority	certificate	Jul 17, 2036, 2:59:59 AM	System Roots
GeoTrust Primary Certification Authority - G2	certificate	Jan 19, 2038, 2:59:59 AM	System Roots
GeoTrust Primary Certification Authority - G3	certificate	Dec 2, 2037, 2:59:59 AM	System Roots
Global Chambersign Root	certificate	Sep 30, 2037, 7:14:18 PM	System Roots
Global Chambersign Root - 2008	certificate	Jul 31, 2038, 3:31:40 PM	System Roots
GlobalSign	certificate	Mar 18, 2029, 1:00:00 PM	System Roots
GlobalSign	certificate	Jan 19, 2038, 6:14:07 AM	System Roots
GlobalSign	certificate	Jan 19, 2038, 6:14:07 AM	System Roots
GlobalSign	certificate	Dec 15, 2021, 11:00:00 AM	System Roots
GlobalSign Root CA	certificate	Jan 28, 2028, 3:00:00 PM	System Roots
Go Daddy Class 2 Certification Authority	certificate	Jun 29, 2034, 8:06:20 PM	System Roots
Go Daddy Root Certificate Authority - G2	certificate	Jan 1, 2038, 2:59:59 AM	System Roots
Government Root Certification Authority	certificate	Dec 31, 2037, 6:59:59 PM	System Roots
Hellenic Academic and Research Institutions RootCA 2011	certificate	Dec 1, 2031, 4:49:52 PM	System Roots
Hongkong Post Root CA 1	certificate	May 15, 2023, 7:52:29 AM	System Roots

Real attacks

Google Security Blog

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

Google Security Blog

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

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.

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.

Real attacks

≡ **threatpost** Podcasts / Malware / Vulnerabilities / InfoSec Insiders / Webinars

← Study: Password Security Improves with Age Researchers Find Methods for

Flame Malware Uses Forged Microsoft Certificate to Validate Components

Microsoft Security Response Center

Report an issue



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.

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 Missina Extensions

Limitation of secure channels

