

# Thierry Sans

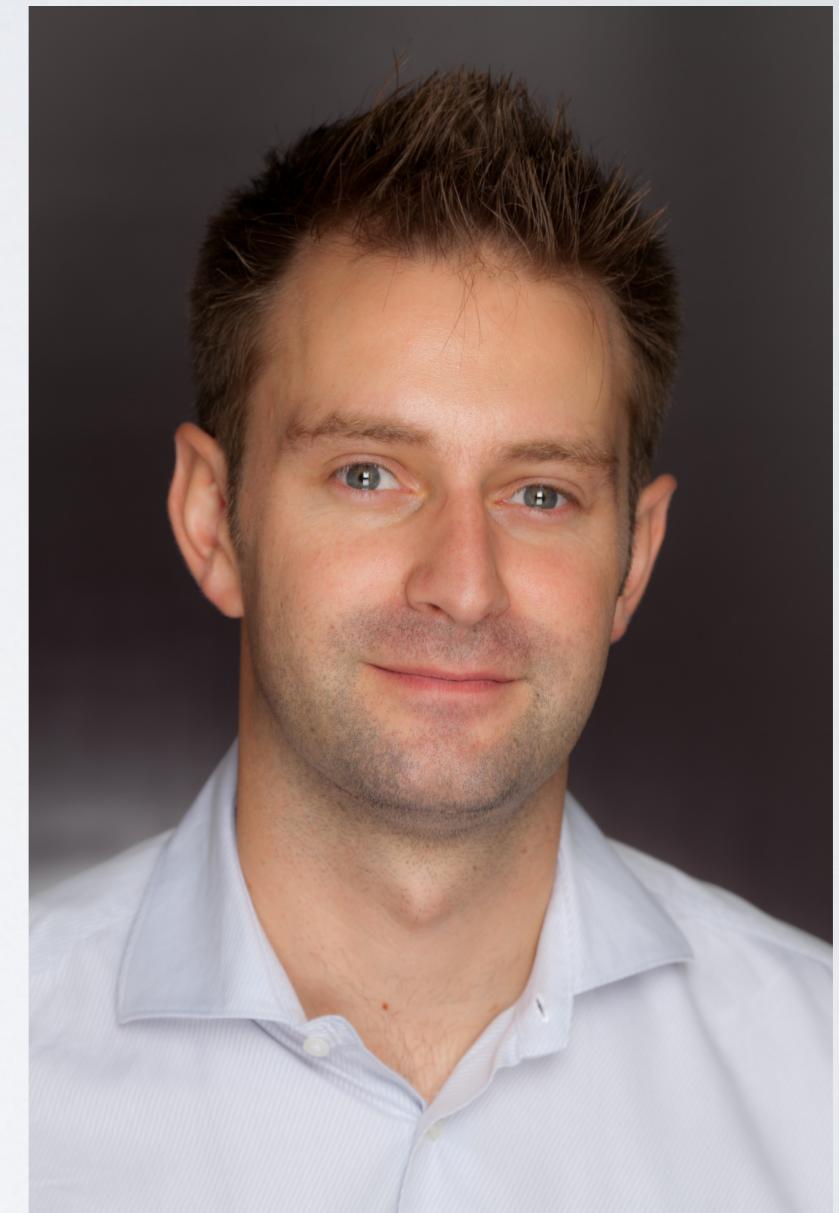
## Associate Professor - Teaching Stream

### Teaching

- Computer and Network Security
- Web Development
- Operating Systems

### Research

- Cryptography protocols
- Zero-knowledge proofs
- Blockchain protocols and applications



# **Understanding the "s" in "https://"**

An introduction to  
modern cryptography

Thierry Sans

# You are using cryptography everyday

- **Browsing the web** with HTTPS://
- **Text messaging** apps
- **Authentication** (passwords, biometrics, one-time code and so on)
- Connecting to a remote server **using SSH**
- **Bitcoin** and other cryptocurrencies

# The Internet

**1980's** - few hosts connected : government institutions and universities

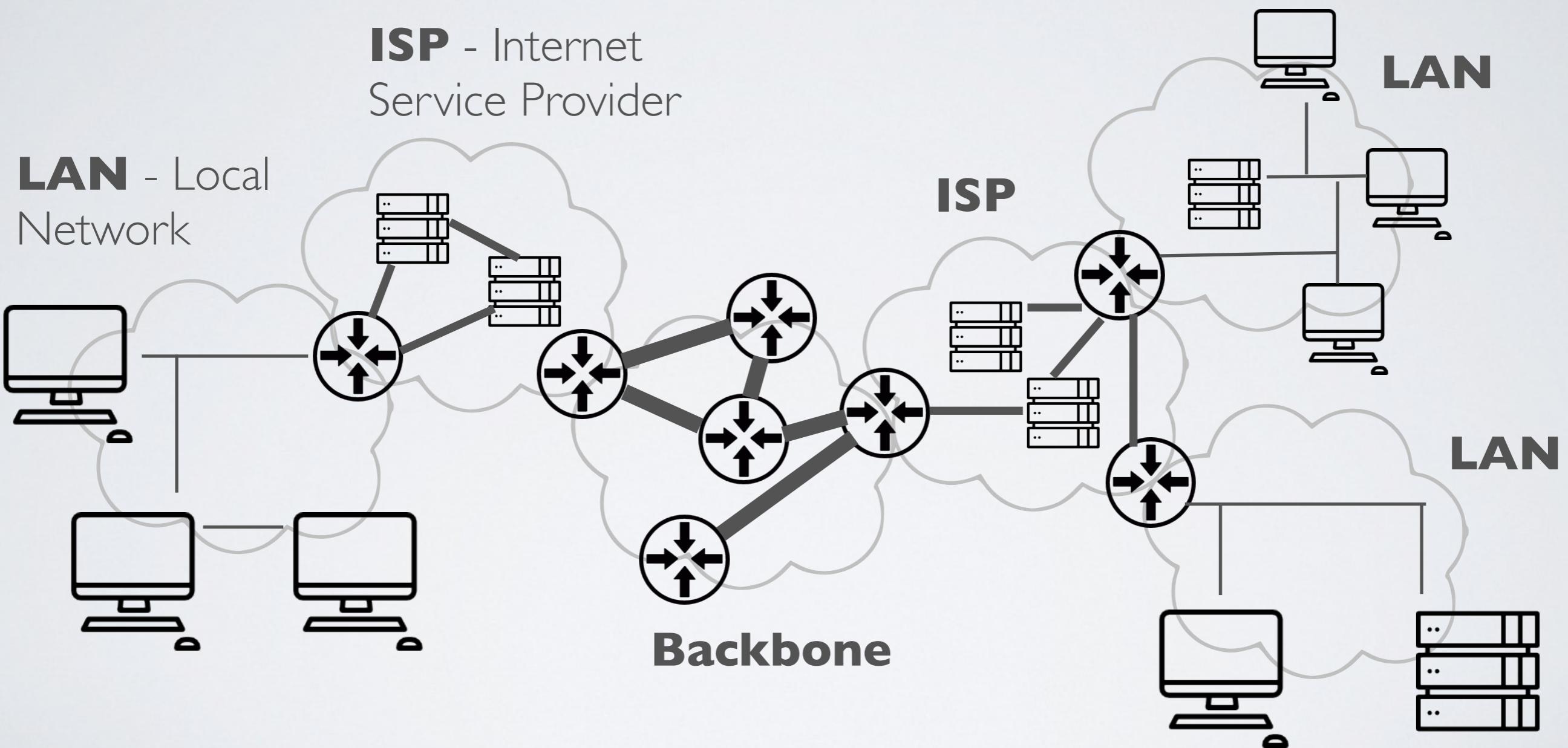
→ Trustworthy environment

**2025** - ~ 5.6 billion internet users : network of networks

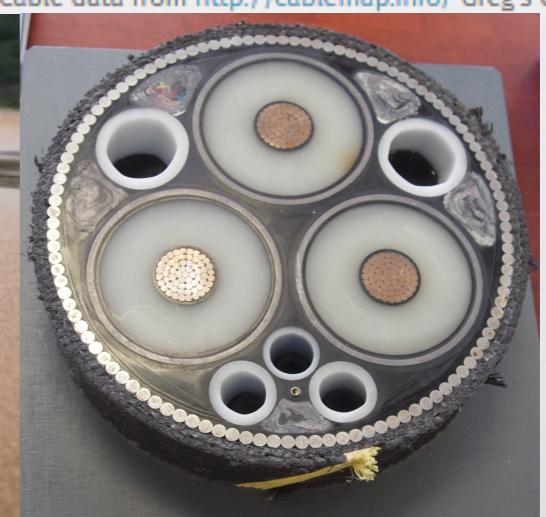
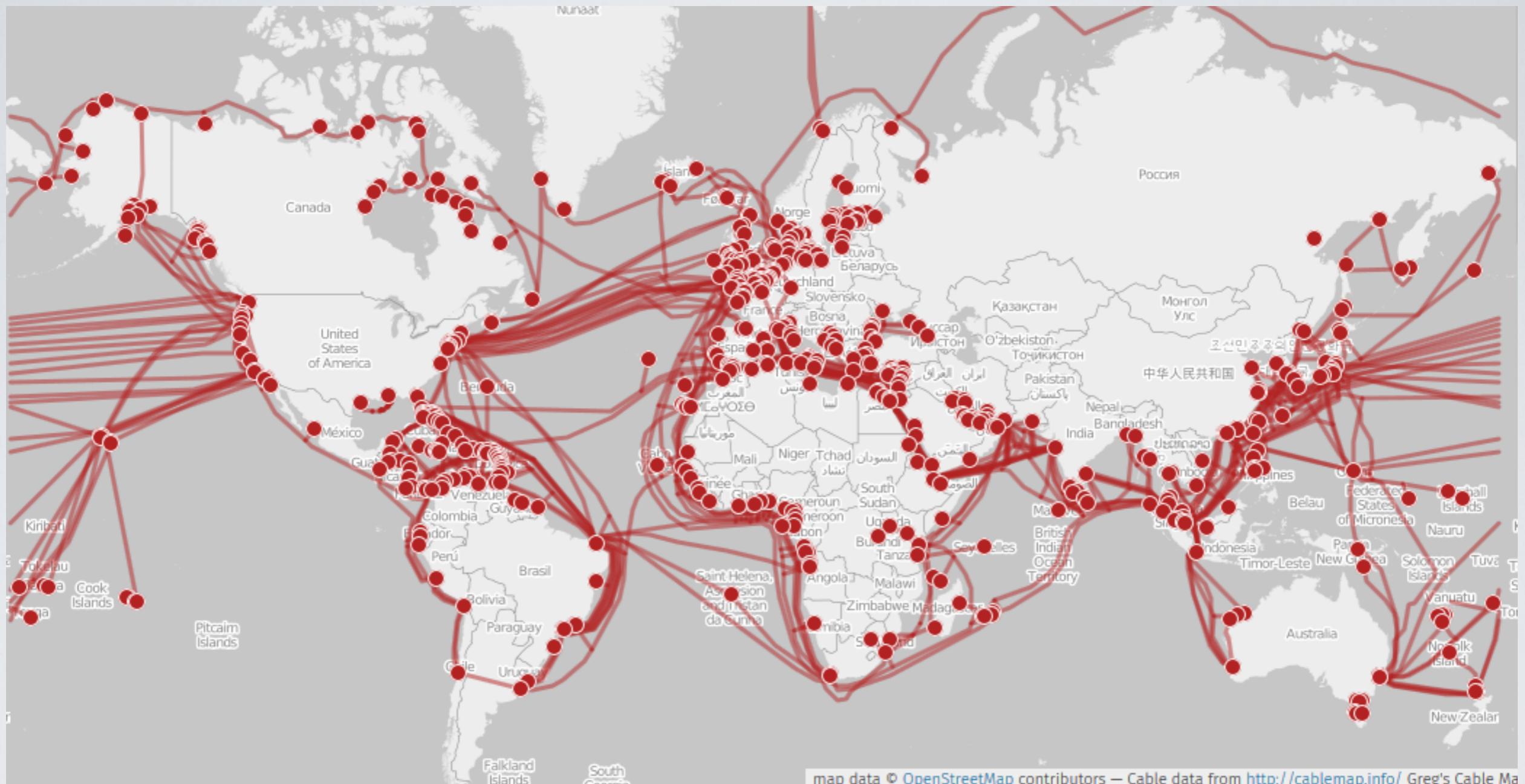
→ Untrustworthy environment

→ Internet (and its protocols) was  
**not designed for untrustworthy environment**

# A network of networks



# The Internet Backbone

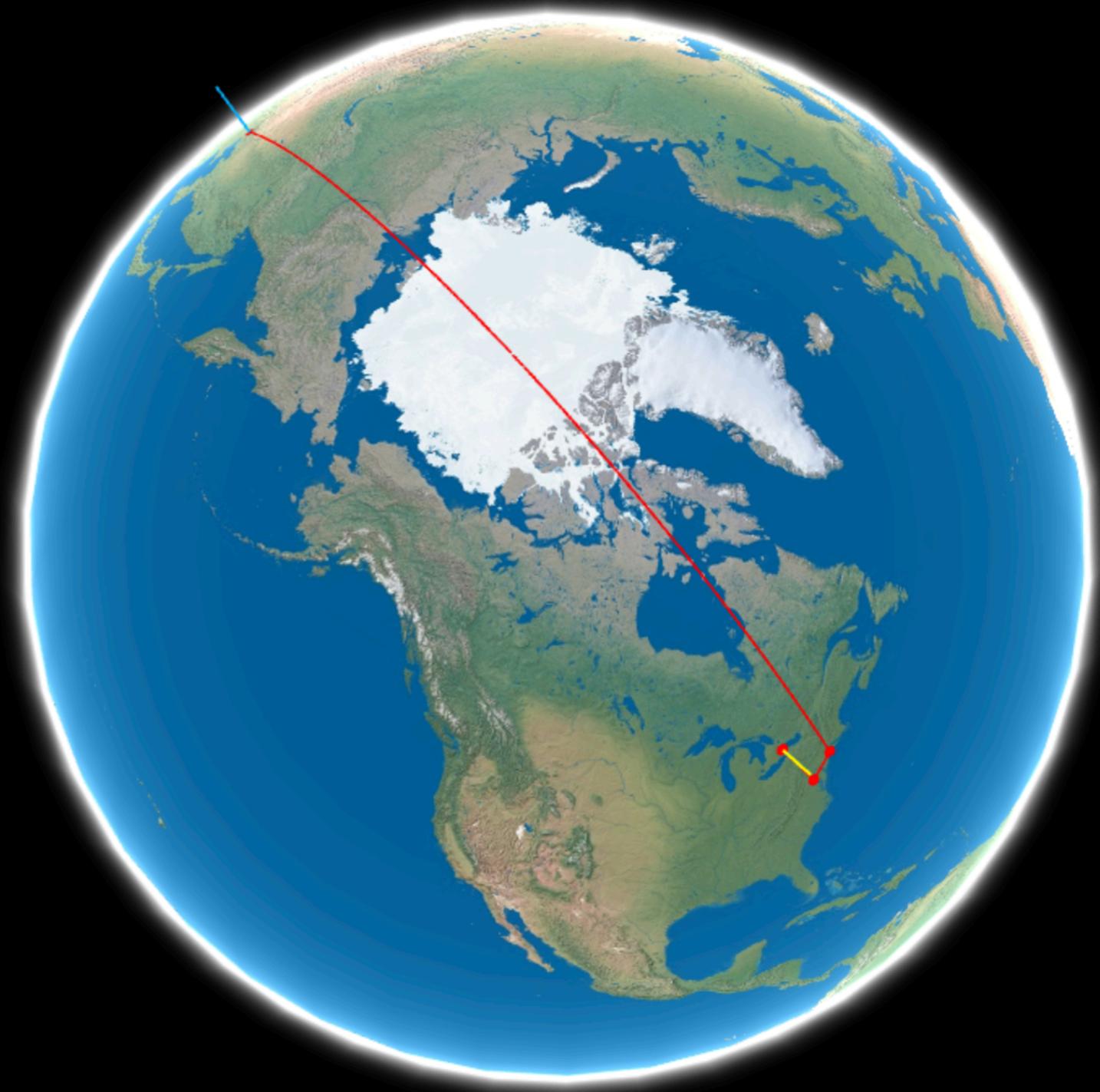


# GeoTraceroute to:

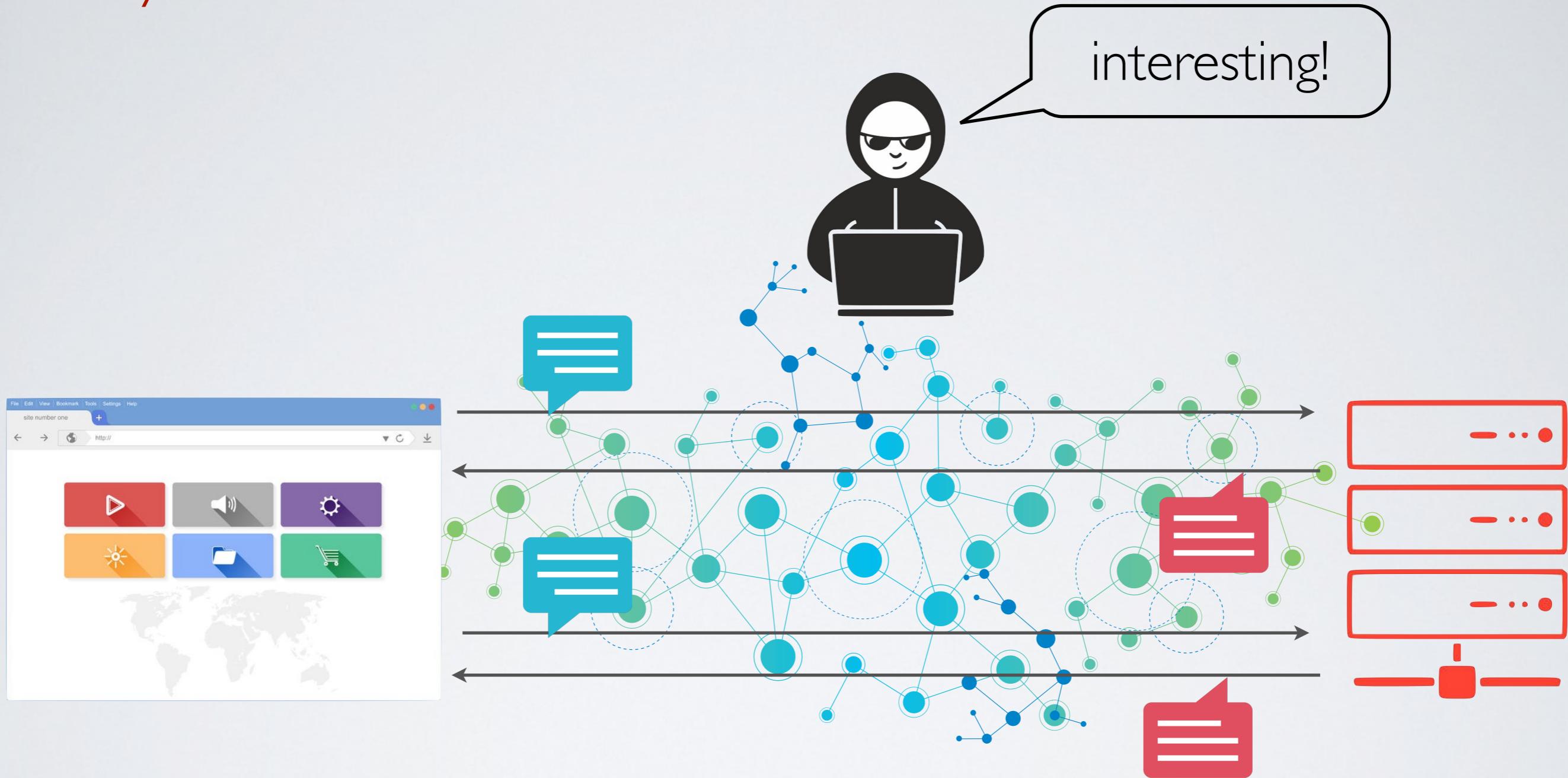


baidu.com

- 
- #1 🇨🇦 CA - Toronto (0 km)  
159.203.1.207 [AS14061] (0 ms)  
143.244.192.138 [AS0] (1 ms)  
143.244.224.37 [AS0] (1 ms)  
157.238.227.100 [AS2914] (1 ms)
  - #2 🇺🇸 US - Washington (564 km)  
129.250.2.141 [AS2914] (15 ms)
  - #3 🇺🇸 US - New York (340 km)  
129.250.3.213 [AS2914] (16 ms)  
129.250.9.222 [AS2914] (22 ms)
  - #4 🇨🇳 CN - Beijing (11012 km)  
202.97.63.98 [AS4134] (71 ms)  
202.97.41.49 [AS4134] (218 ms)  
202.97.12.49 [AS4134] (218 ms)  
36.110.251.78 [AS23724] (222 ms)  
106.38.196.190 [AS23724] (222 ms)  
182.61.255.66 [AS38365] (221 ms)  
182.61.255.45 [AS38365] (221 ms)  
182.61.201.211 [AS38365] (244 ms)

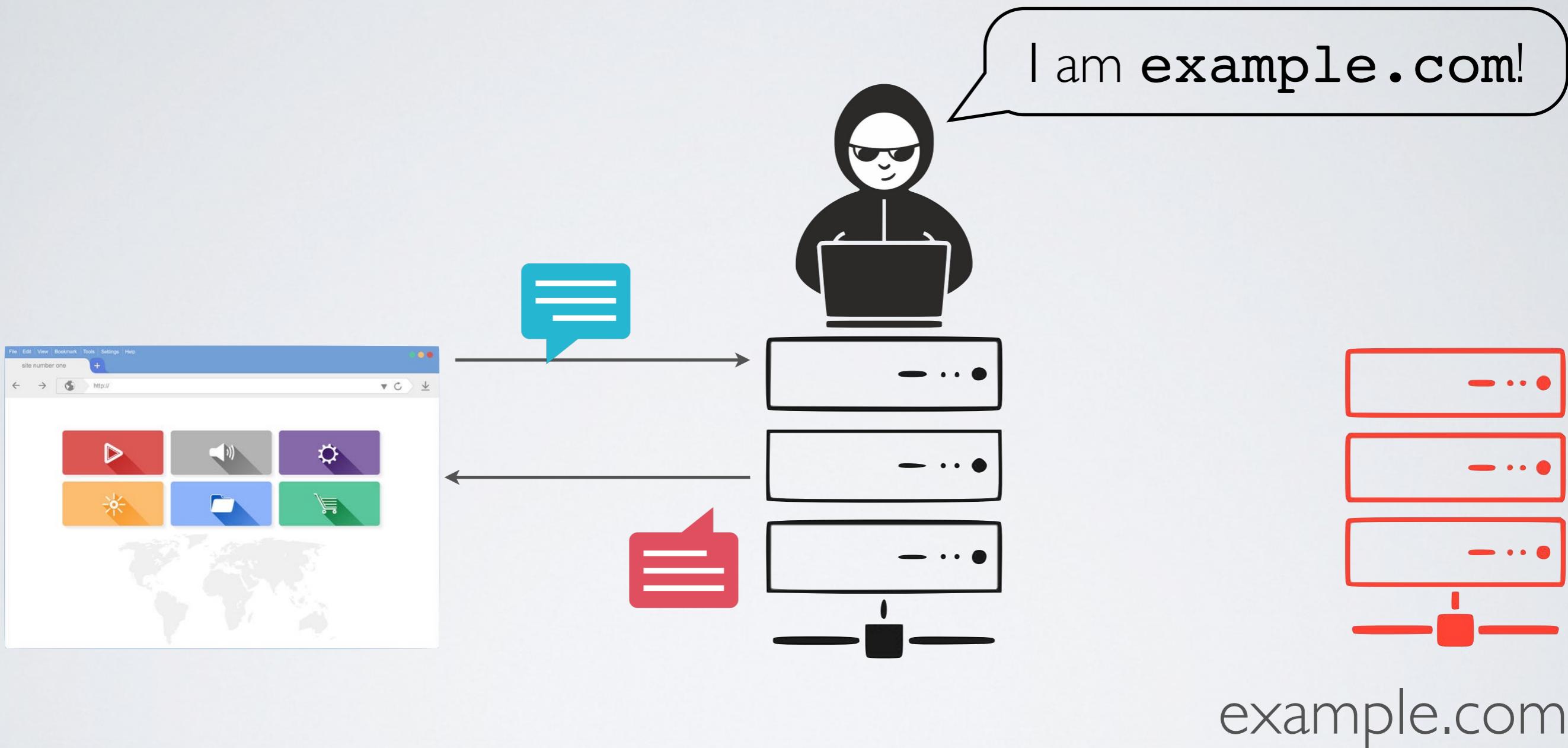


# Do you trust the network?



- Threat I : an attacker **can eavesdrop** messages sent back and forth

# Do you *really* trust the network?



- Threat 2: an attacker **can tamper with** messages sent back and forth

# Confidentiality and Integrity

- Threat 1: an attacker **can eavesdrop** messages sent back and forth

**Confidentiality:** How do we exchange information secretly?

- Threat 2: an attacker **can tamper** with messages sent back and forth

**Integrity:** How do we exchange information reliably?

# HTTPS = HTTP + TLS

- Transport Layer Security (formerly SSL) provides
  - **confidentiality:** end-to-end secure channel
  - **integrity:** authentication handshake
- ✓ Prevents all kinds of eavesdropping and tampering protecting many internet protocols

# An introduction to cryptography

## **Protecting confidentiality**

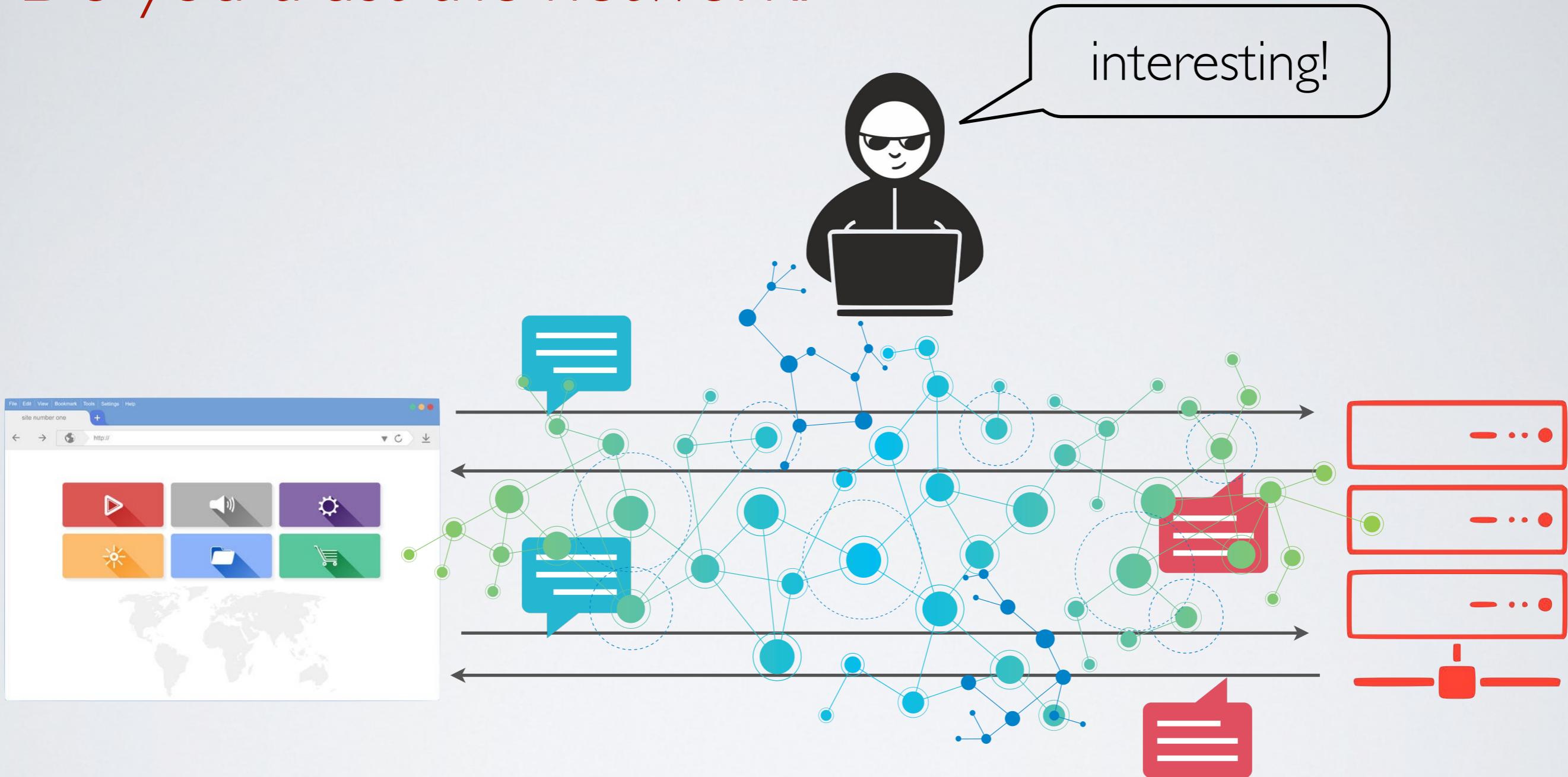
- Symmetric Encryption
- Asymmetric Encryption
- Key Exchange

## **Protecting Integrity**

- Digital Signatures
- Public Key Infrastructure

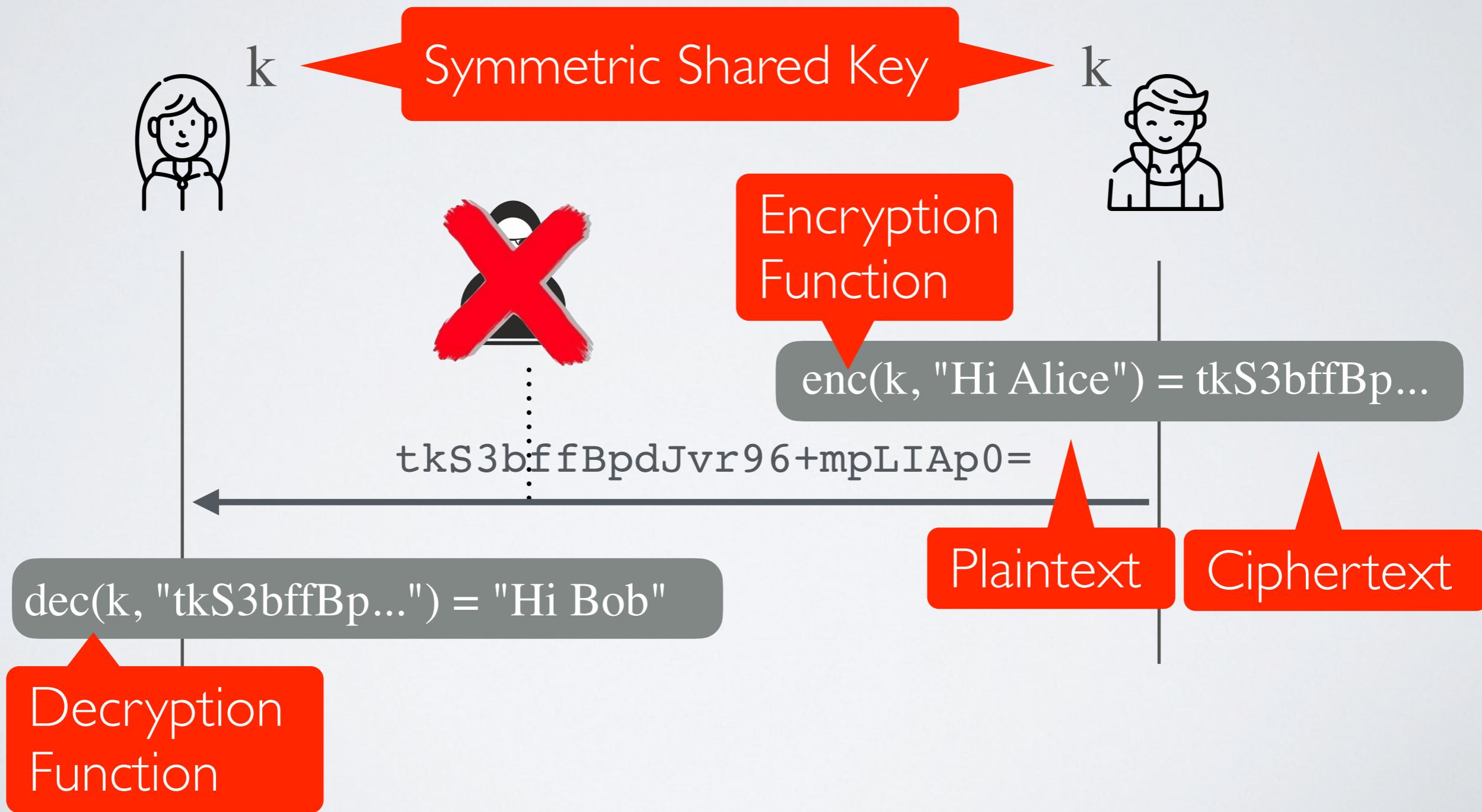
# Protecting Confidentiality

# Do you trust the network?



- Threat I : an attacker **can eavesdrop** messages sent back and forth

# Encryption to protect confidentiality



# Definitions

## **Plaintext**

The message in its original form (a.k.a "in clear")

## **Ciphertext**

The message in its encrypted form

## **Encryption Function**

Transform a plaintext into ciphertext

## **Decryption Function**

Transform a ciphertext into a plaintext

# Definitions

## **Cryptographic algorithm**

The method to do encryption and decryption

- ✓ The cryptographic algorithm is public

## **Cryptographic key**

An input variable used by the algorithm for the transformation

- The key must remain secret between the two parties

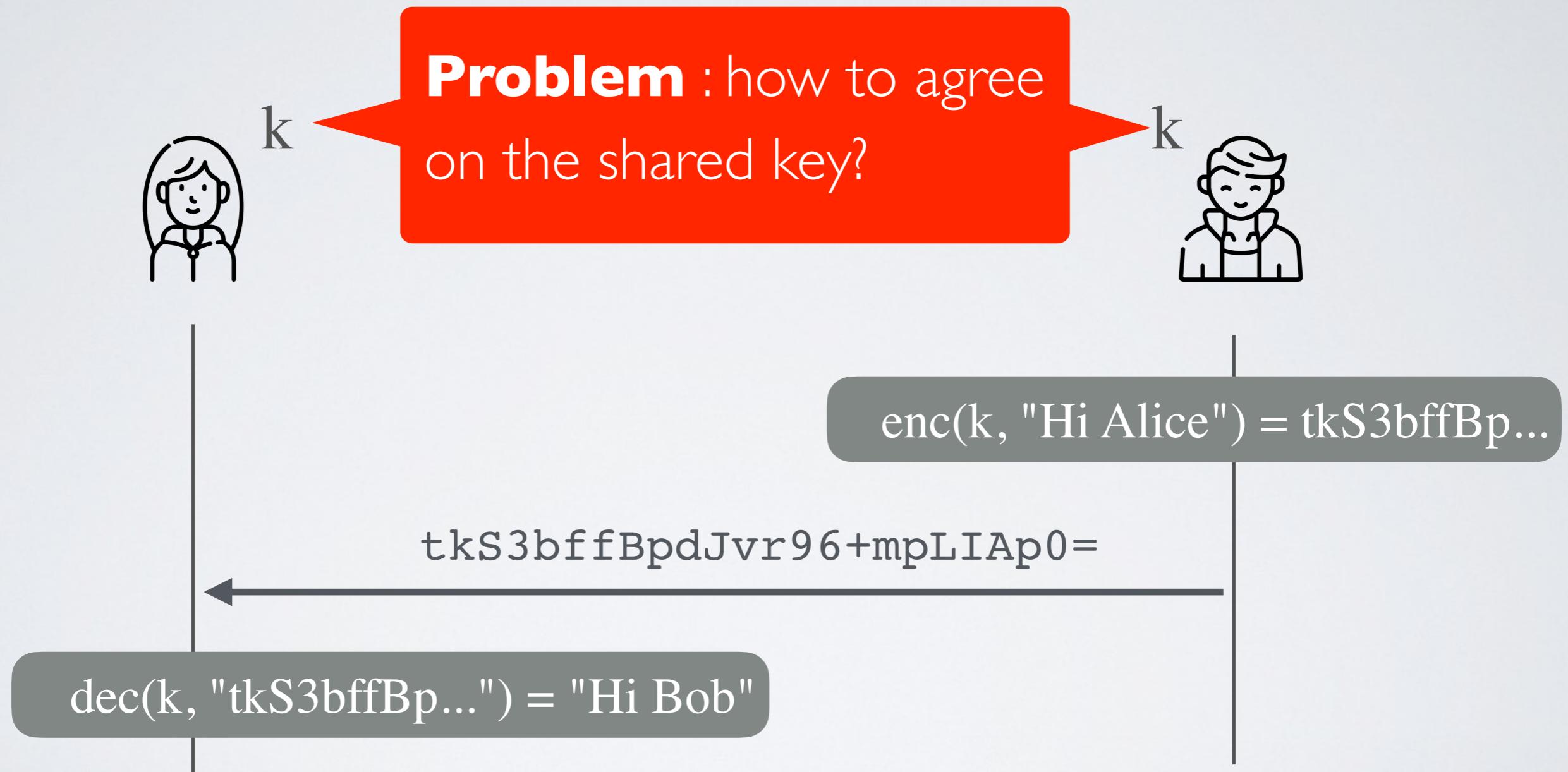
# The Kerckhoffs' principle (1883)

“The enemy knows the system” - the security of a communication should **not** rely on the fact that the algorithms are secrets

- A cryptosystem should be secure even if **everything about the system, except the key, is public knowledge**

**No security by obscurity**

# The big challenge with symmetric encryption



# Asymmetric encryption a.k.a Public Key Cryptography



$K_{sA}, K_{pA}$



$K_{pA}$



$K_{pA}$

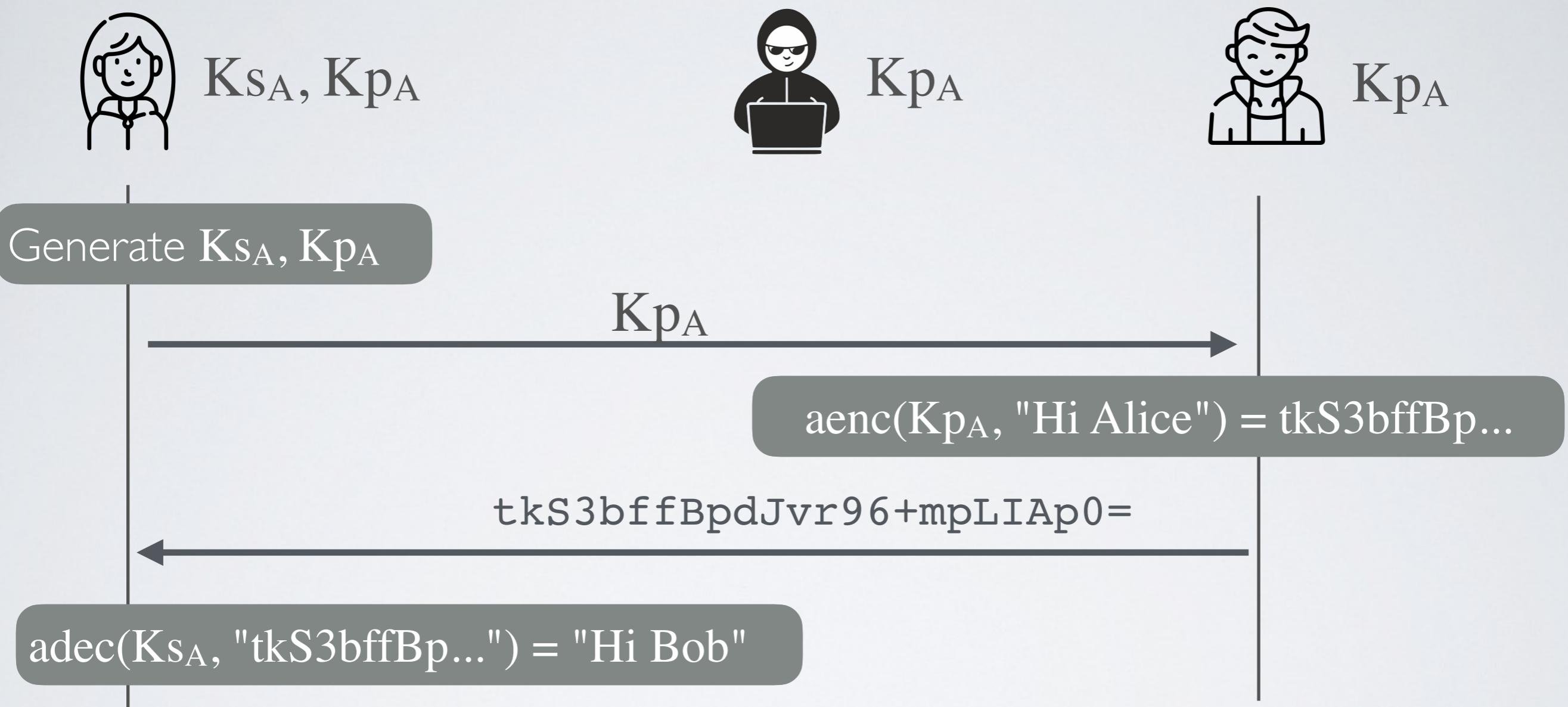
Generate  $K_{sA}, K_{pA}$

$K_{pA}$

Alice generates a pair of asymmetric keys

- $K_{sA}$  is the secret key that Alice keeps for herself
  - $K_{pA}$  is the public key that Alice gives to everyone (even Mallory)
- These two keys  $K_{sA}$  and  $K_{pA}$  work together

# Asymmetric encryption **for confidentiality**



- Nobody can decrypt m, except Alice with her private key  $K_{sA}$
- ✓ Confidentiality without the need to exchange a secret key

# Asymmetric vs Symmetric

	Symmetric	Asymmetric
pro	Faster	No key agreement
cons	Key agreement	Slower and works on small message only

The best of both worlds

- Use asymmetric encryption to encrypt a shared key
- Use symmetric encryption to encrypt message

# (naive) key exchange using asymmetric encryption



Generate  $K_{sA}, K_{pA}$

$K_{pA}$

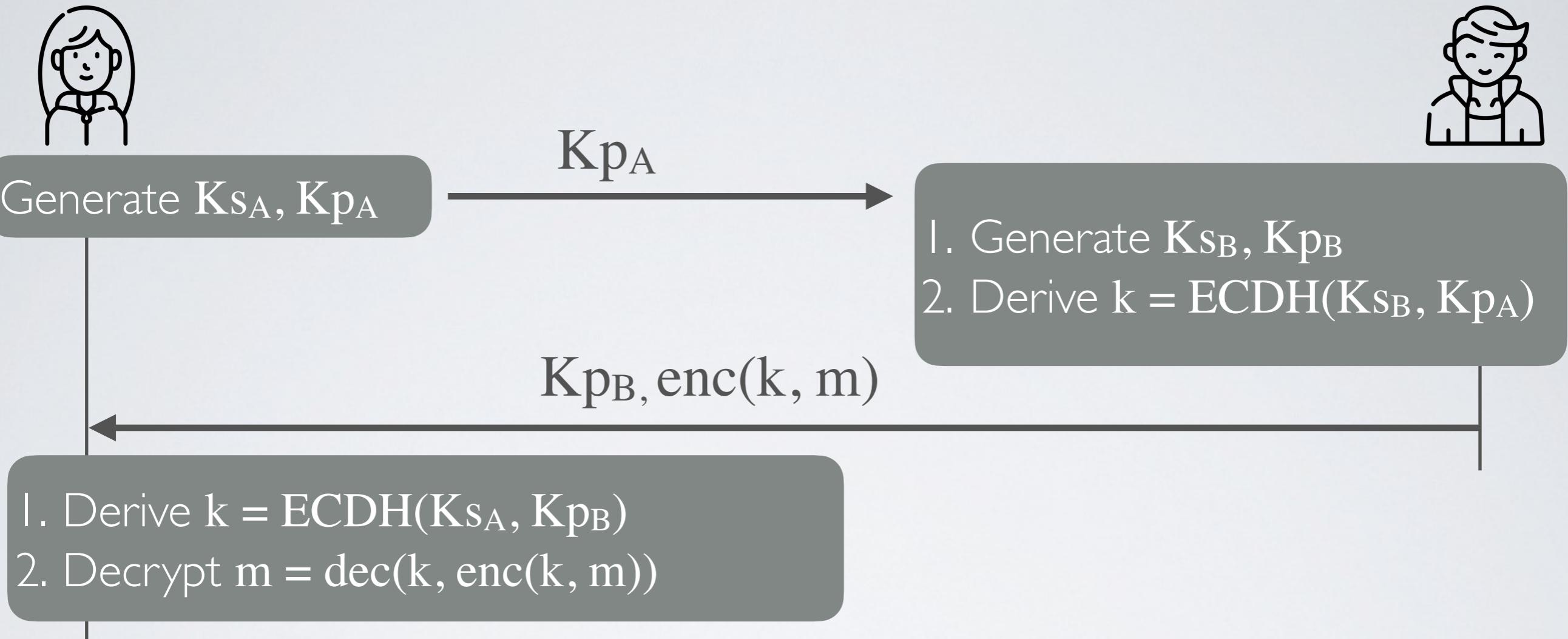
Generate  $k$

$aenc(K_{pA}, k), enc(k, m)$

1. Decrypt  $k = denc(K_{sA}, aenc(K_{pA}, k))$
2. Decrypt  $m = dec(k, enc(k, m))$

- Protecting the shared key is the **responsibility of Alice only**
- Generating the shared key is the **responsibility of Bob only**

# (better) key exchange using Diffie-Hellman-Merkle key exchange protocol

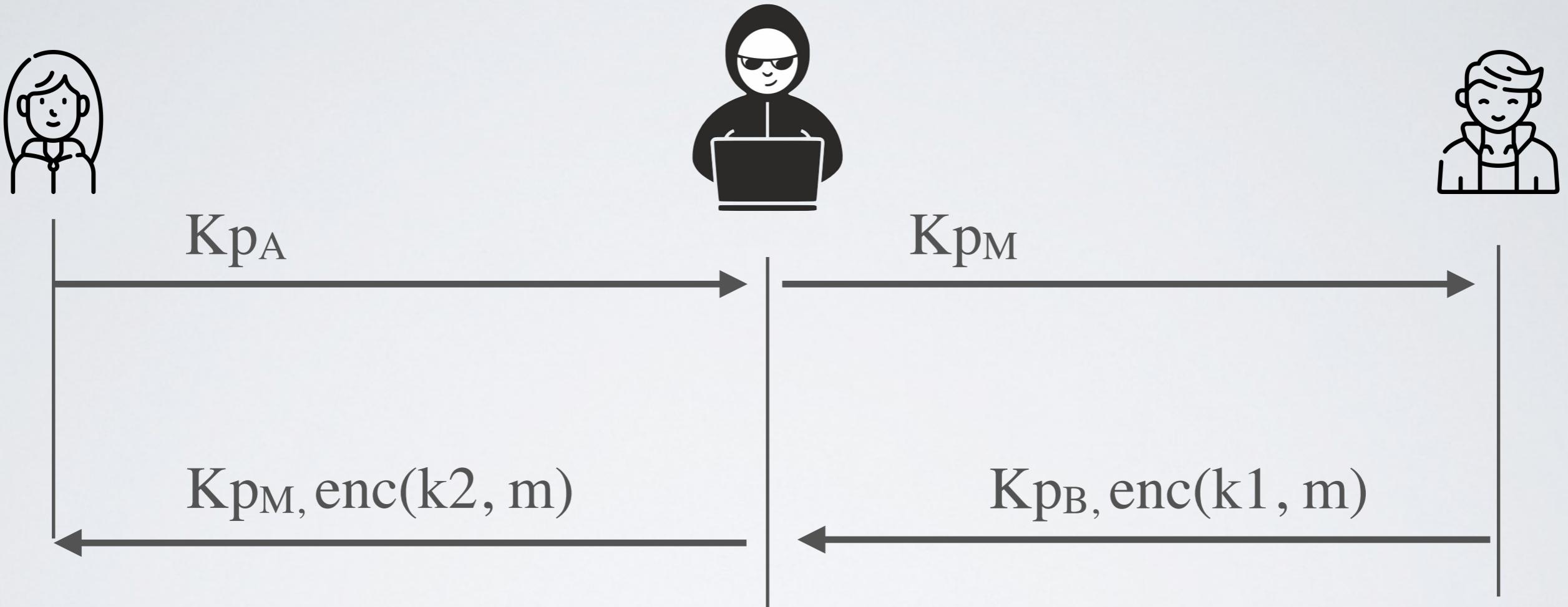


**Diffie-Hellman-Merkle** provides a way to generate a shared key from two asymmetric key pairs

$$\text{ECDH}(K_{sA}, K_{pB}) = \text{ECDH}(K_{sB}, K_{pA}) = k$$

- ✓ Mutual contribution to the key generation
- ✓ No need to send the encrypted shared key

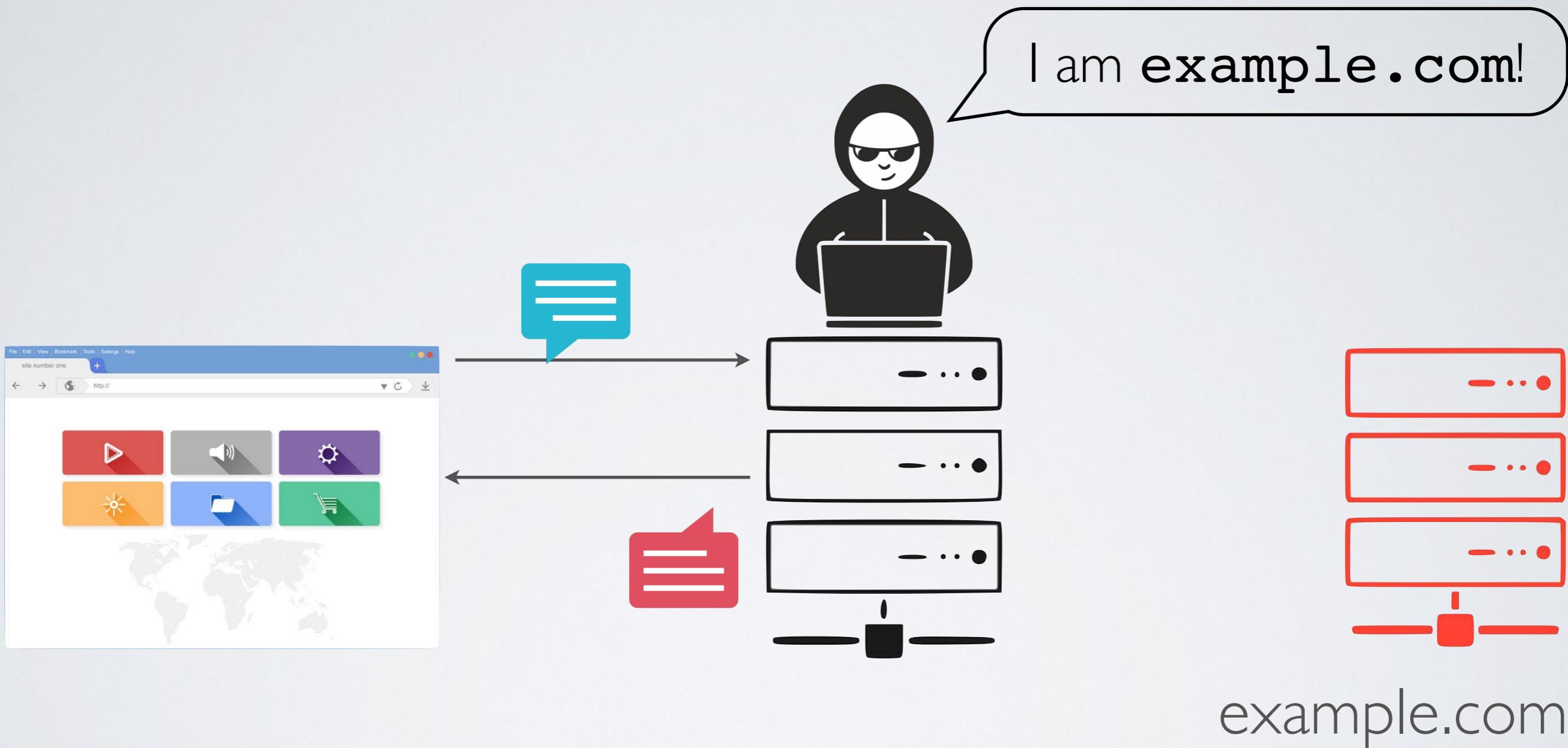
# Are we done yet?



- ✓ Encryption and key exchange protects against confidentiality ...
- ... but not **does not protect integrity**

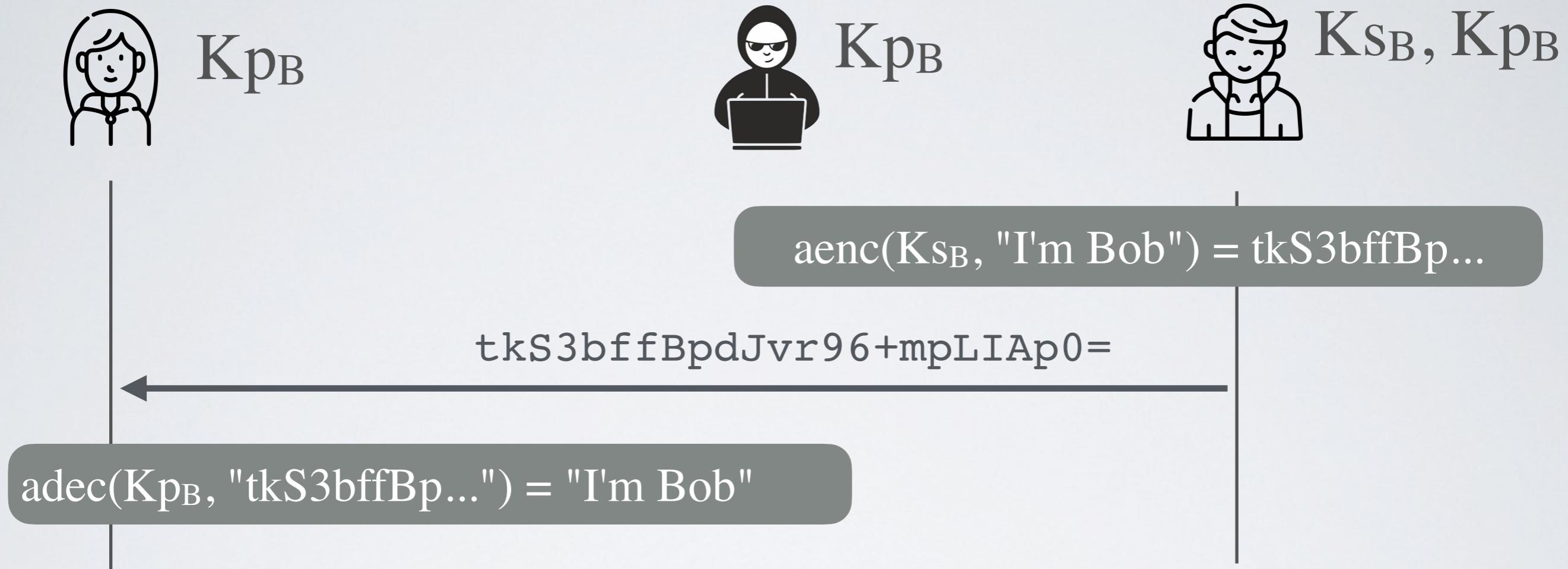
Protecting Integrity

# Do you *really* trust the network?



- Threat 2: an attacker **can tamper with** messages sent back and forth

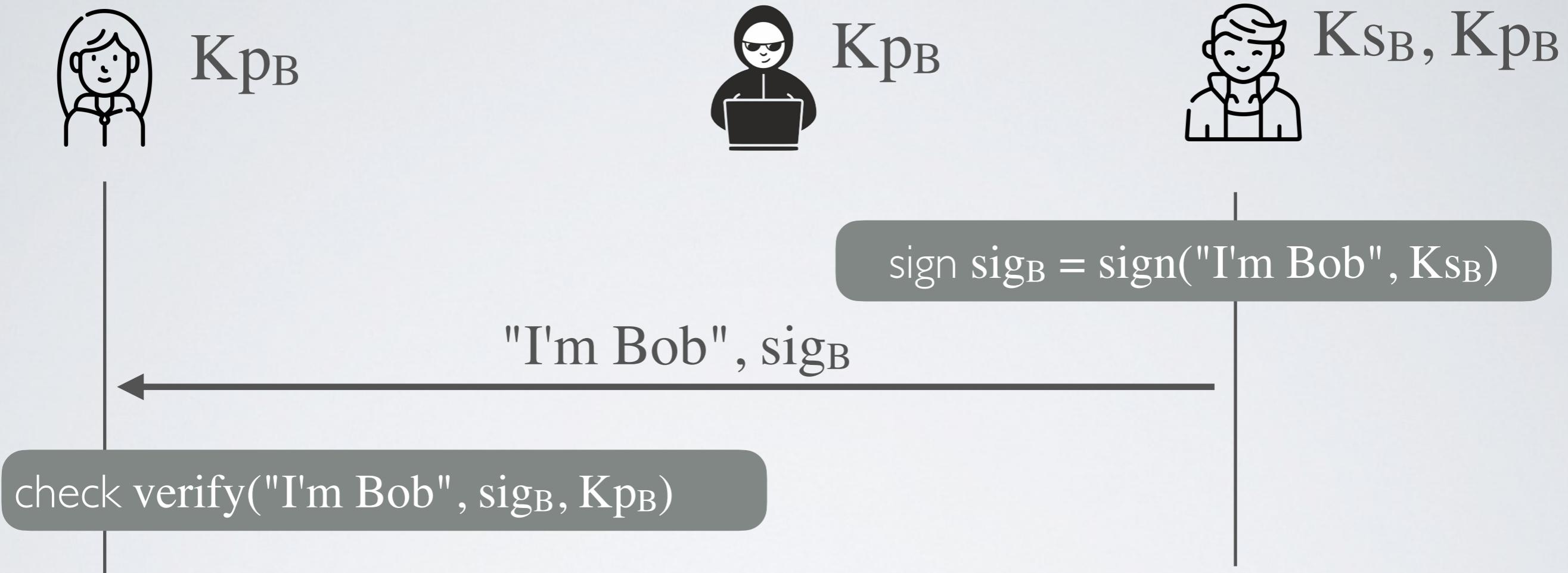
# Asymmetric encryption **for Integrity**



→ Everybody can decrypt  $m$  using Bob's public key  $K_{pB}$

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

# Digital Signatures

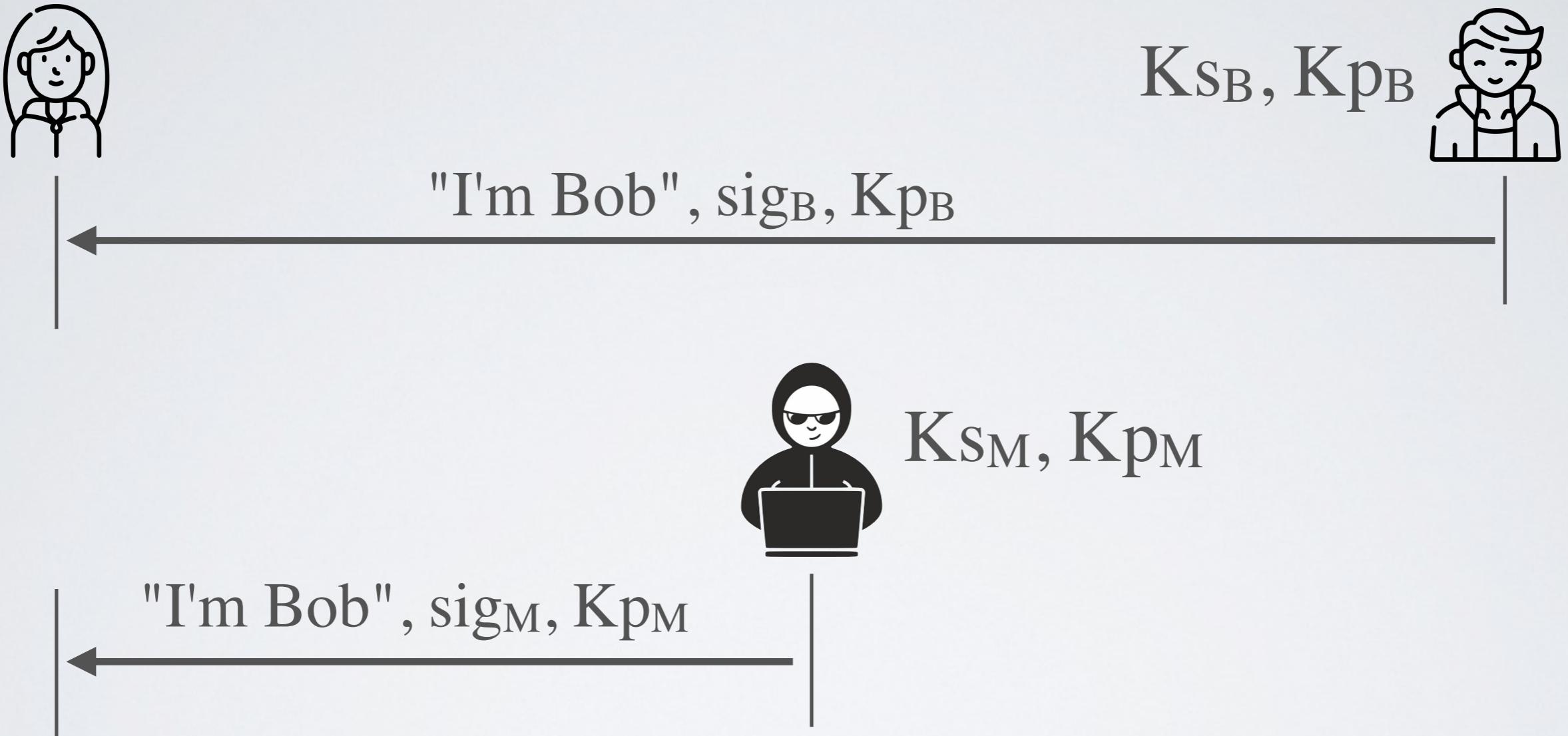


Only Bob can sign a message  $m$  with his secret key  $K_{sB}$

→ Everybody can verify  $m$  and its  $\text{sig}$  using Bob's public key  $K_{pB}$

# Problem :

How does everyone knows Bob's public key?



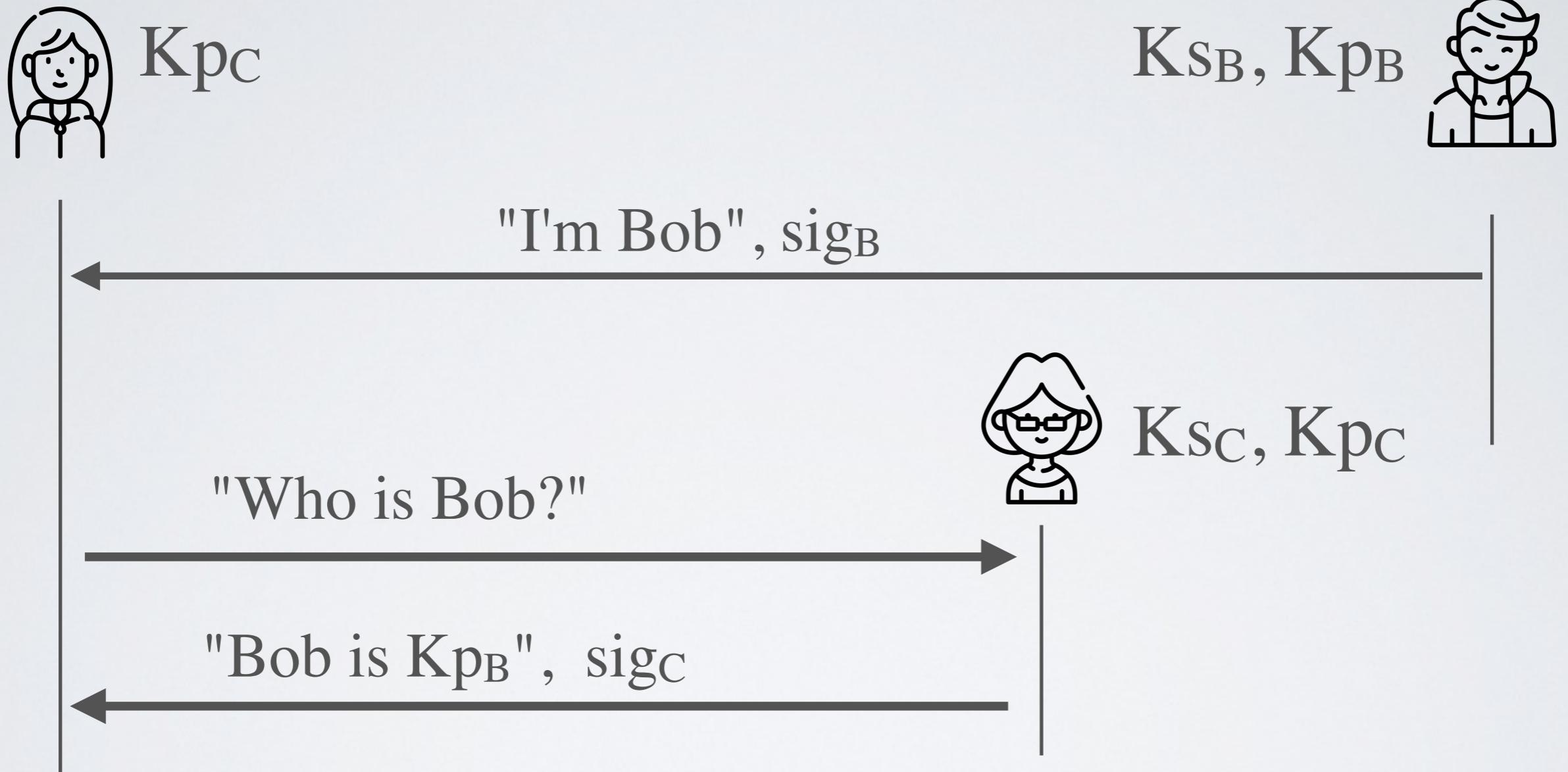
- If Alice does not know Bob in the first place, Bob cannot just send his key and ask Alice to trust it

# Transitive Trust

If Alice does not know Bob maybe **she knows Charlie that knows Bob** and can vouch for him

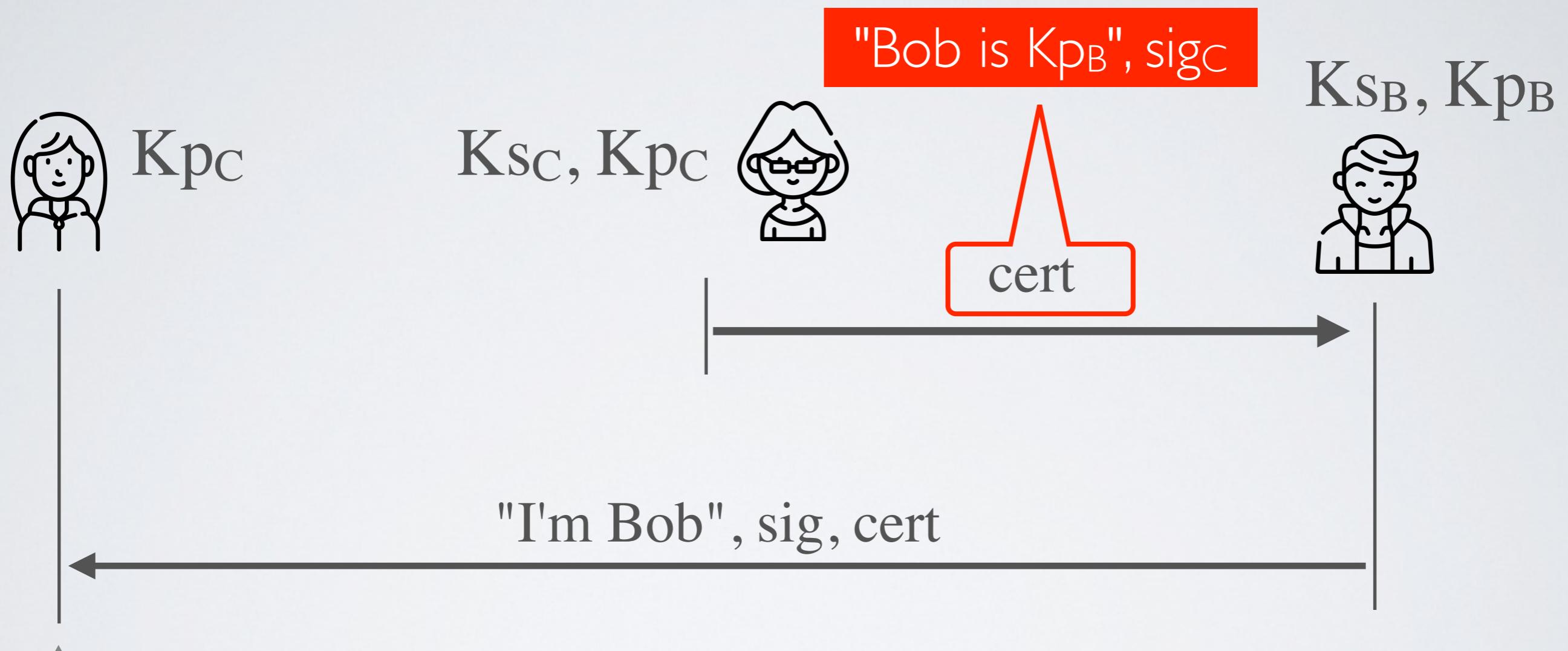
- ✓ Charlie is a Trusted-Third Party for Alice

# Solution I: Online Public Key Registry



1. check verify("Bob is K<sub>PB</sub>", sig<sub>C</sub>, K<sub>PC</sub>)
2. check verify("I'm Bob", sig<sub>B</sub>, K<sub>PB</sub>)

## Solution 2 : Offline Public Key Certification



1. check verify("Bob is  $K_{pb}$ ",  $\text{sig}_c$ ,  $K_{pc}$ )
2. check verify("I'm Bob",  $\text{sig}_b$ ,  $K_{pb}$ )

Putting everything together

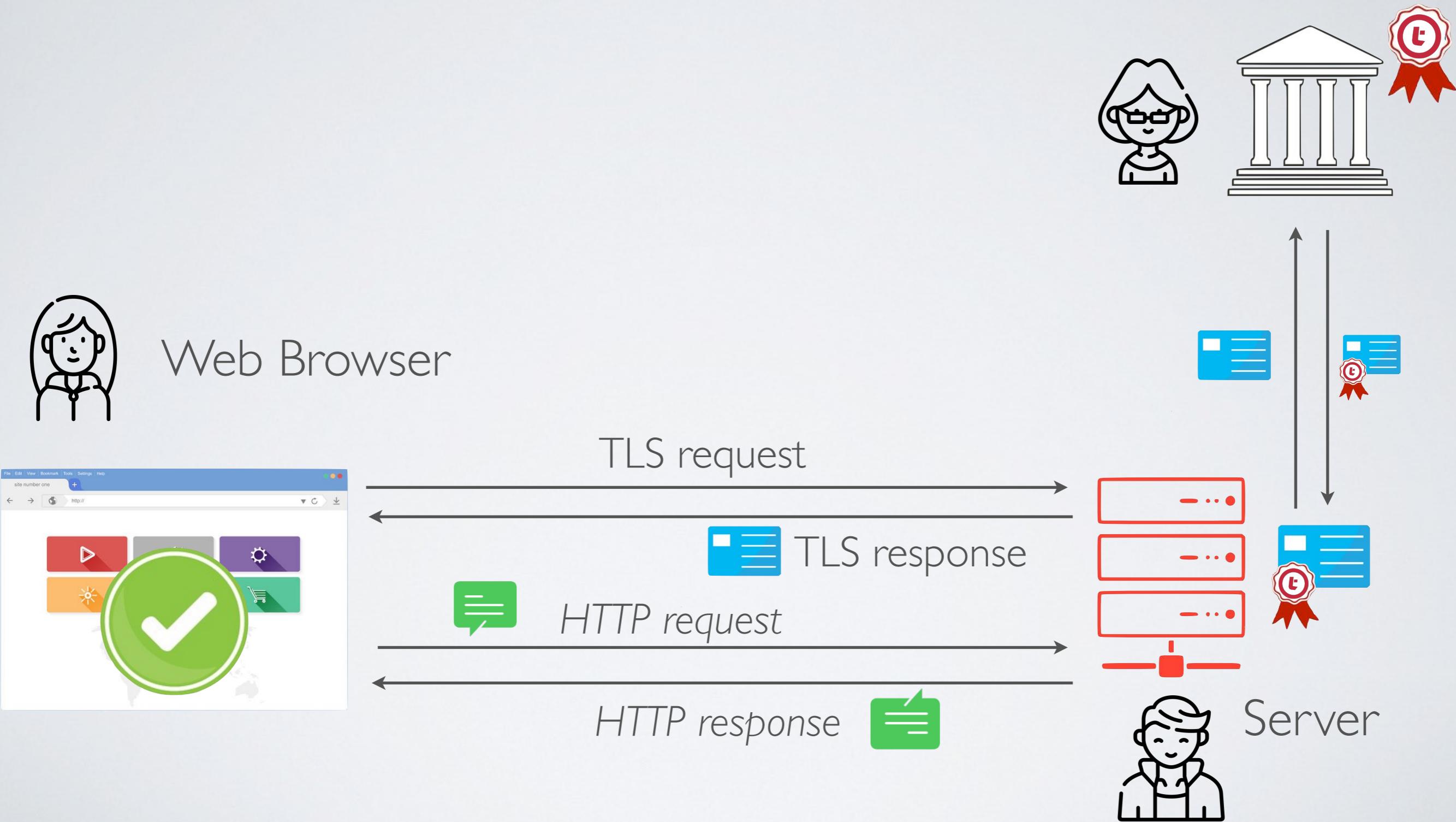
Understanding HTTPS

# HTTPS = HTTP + TLS

- Transport Layer Security (formerly SSL) provides
  - **confidentiality:** end-to-end secure channel
  - **integrity:** authentication handshake
- ✓ Prevents all kinds of eavesdropping and tampering protecting many internet protocols

# How HTTPS works

Certificate Authority (CA)



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

Keychain Access

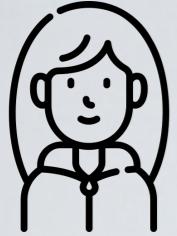
Click to unlock the System Roots keychain.

Keychains	Name	Kind	Expires	Keychain
login	Echoworx Root CA2	certificate	Oct 7, 2030, 1:49:13 PM	System Roots
Microsoft_Intermediate_Certificates	EE Certification Centre Root CA	certificate	Dec 18, 2030, 2:59:59 AM	System Roots
Local Items	Entrust Root Certification Authority	certificate	Nov 27, 2026, 11:53:42 PM	System Roots
System	Entrust Root Certification Authority - EC1	certificate	Dec 18, 2037, 6:55:36 PM	System Roots
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
Category	GeoTrust Global CA	certificate	May 21, 2022, 7:00:00 AM	System Roots
All Items	GeoTrust Primary Certification Authority	certificate	Jul 17, 2036, 2:59:59 AM	System Roots
Passwords	GeoTrust Primary Certification Authority - G2	certificate	Jan 19, 2038, 2:59:59 AM	System Roots
Secure Notes	GeoTrust Primary Certification Authority - G3	certificate	Dec 2, 2037, 2:59:59 AM	System Roots
My Certificates	Global Chambersign Root	certificate	Sep 30, 2037, 7:14:18 PM	System Roots
Keys	Global Chambersign Root - 2008	certificate	Jul 31, 2038, 3:31:40 PM	System Roots
Certificates	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

177 items

# Simplified TLS 1.3 (2018)

Key exchange and encryption only



Generate  $K_{s1}, K_{p1}$



$K_{p1}$

1. Generate  $K_{s2}, K_{p2}$
2. Derive  $k = \text{ECDH}(K_{s2}, K_{p1})$

$K_{p2}$

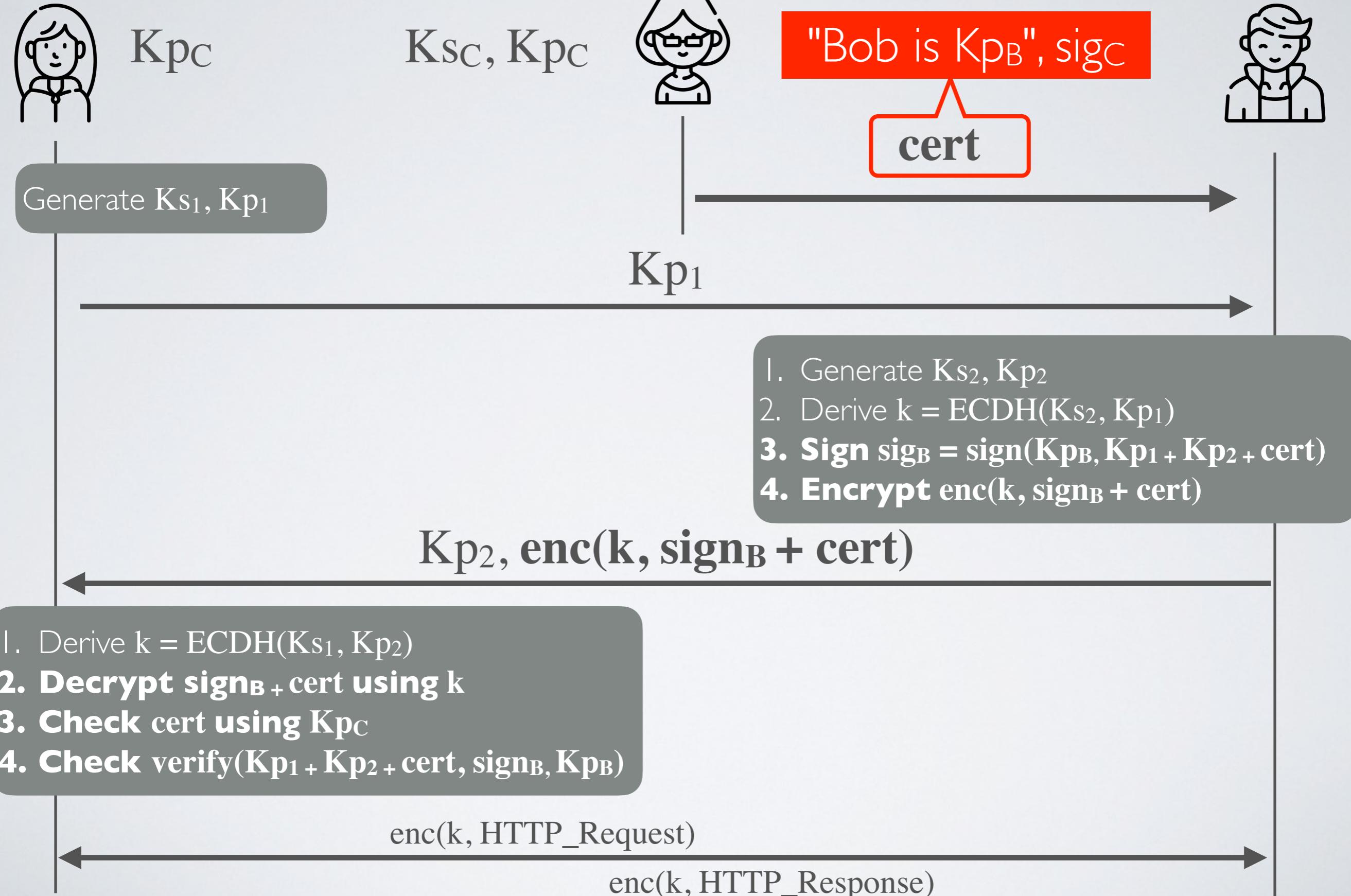
Derive  $k = \text{ECDH}(K_{s1}, K_{p2})$

$\text{enc}(k, \text{HTTP\_Request})$

$\text{enc}(k, \text{HTTP\_Response})$

# Simplified TLS 1.3 (2018)

## Adding one-way authentication



This afternoon,  
let's implement TLS in Python