Redes de Computadores II

Universidade do Algarve

Semana 11

https://github.com/ncatanoc/redes_algarve

Néstor Cataño

nestor.catano@gmail.com

Networking outlook

Goal:

to understand the basic underpinnings of TLS and HTTPS.

TLS Cryptography

- Client and server agree on the cryptographic primitives during the handshake.
- Cryptography is hard:
  - Weak/broken ciphersuites
  - Implementation flaws
  - Cryptographic oracles
  - Downgrade attacks

The complexity of TLS cryptography is mind-blowing
Open SSL contains 27 000 lines of Perl scripts
to generate assembly code for AES

Roadmap

1. **Transport Layer Security (TLS)**
2. Certificate Authority (CA)
3. Man-in-the-Middle attacks

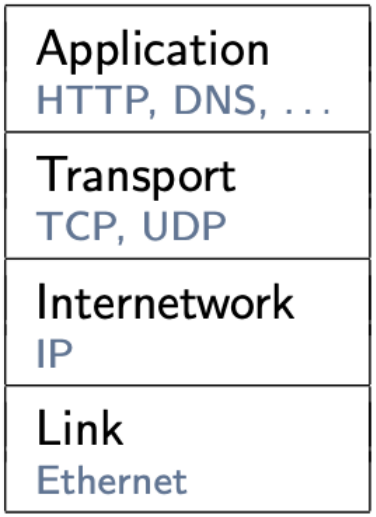
TLS - introduction

The Transport Layer Security Protocol (TLS)

Why a new protocol?

Protection Goals

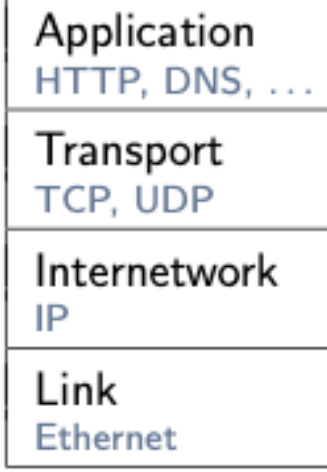
- Confidentiality
- Integrity
- Availability



The 3 first layers provide Availability but not Confidentiality and Integrity

⇒ TLS ensures confidentiality and integrity of message contents.

TLS - introduction

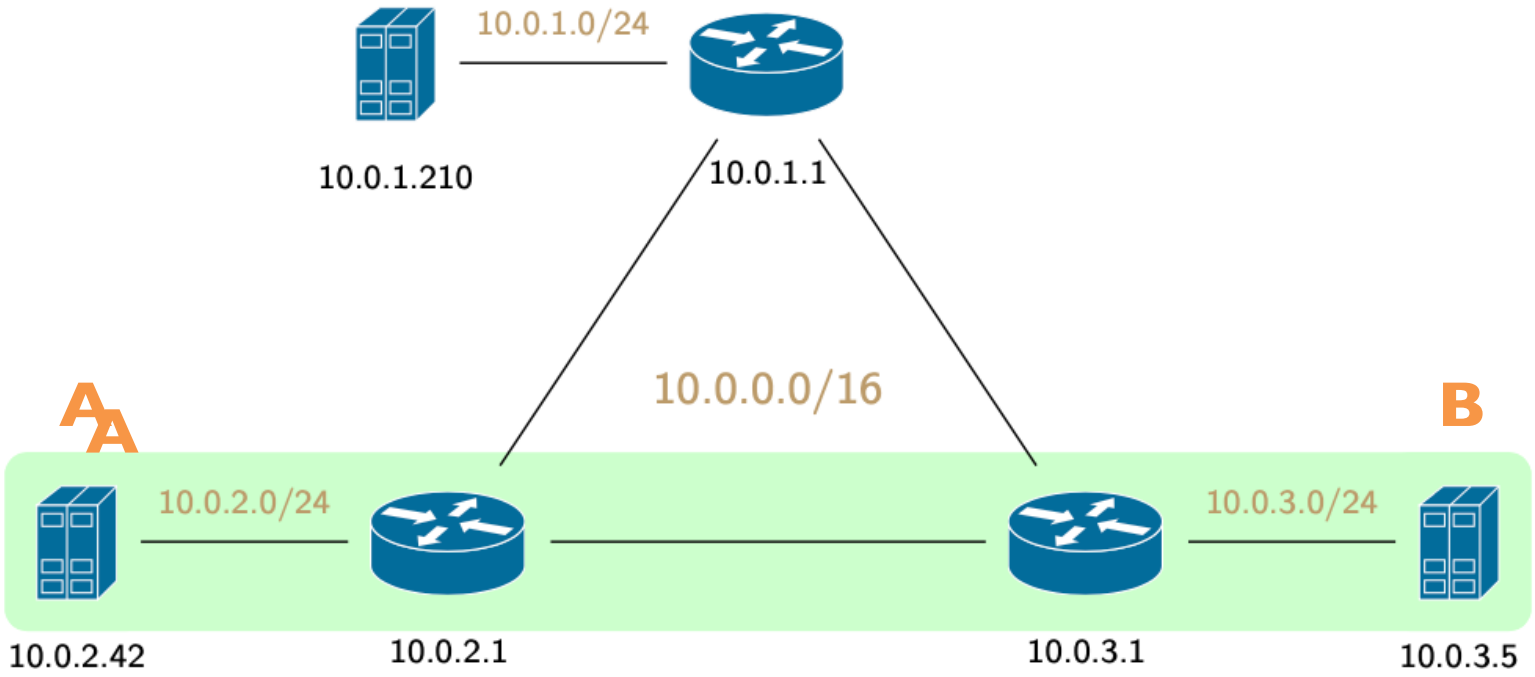


- The first 3 layers:
  - provide Availability
  - do not provide Confidentiality or Integrity

What do we need?

- Confidentiality: we use cryptography
  - We encrypt messages before passing them to TCP and UDP
- Integrity: we need to detect message modifications
- we need to make sure we are talking to the right person
  - e.g. Client browser talking to a server

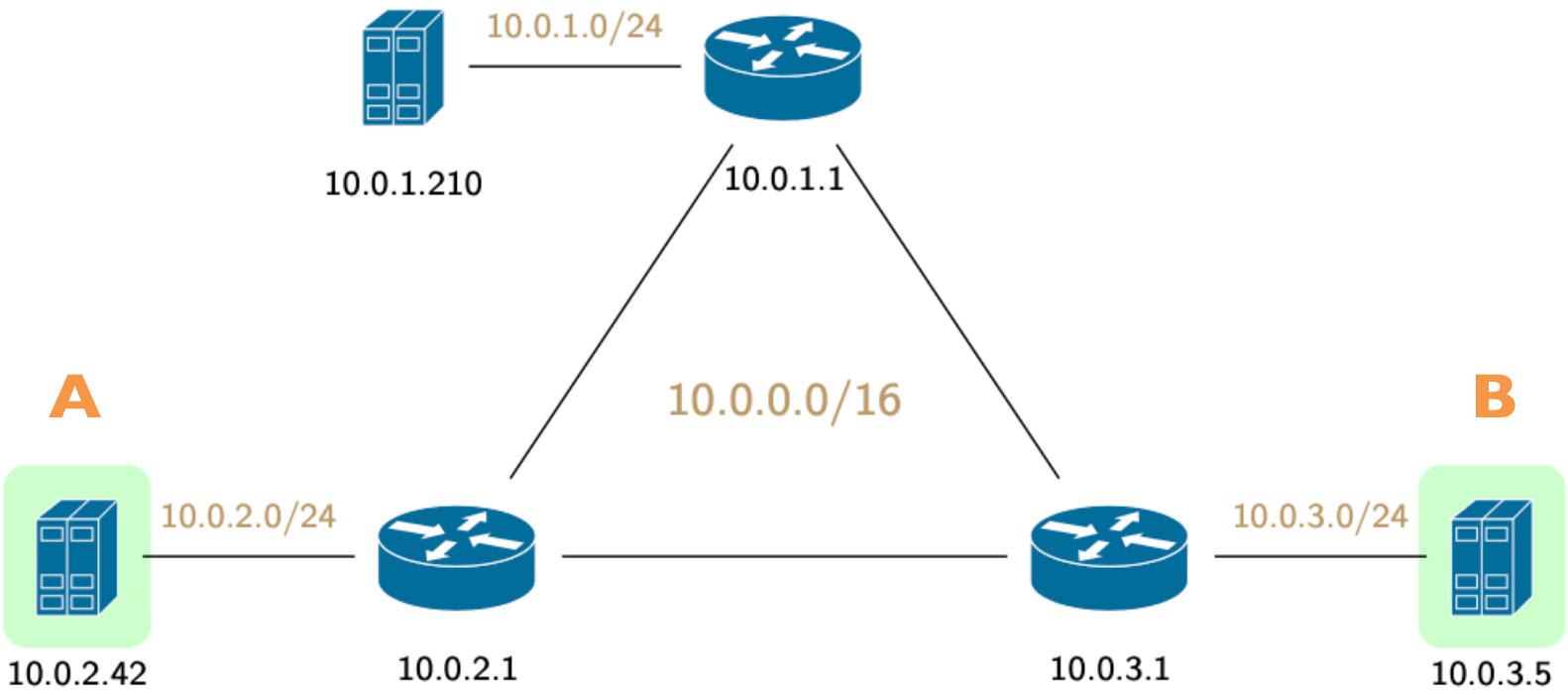
Dolev-Yao: Trusted Areas (No TLS)



Anyone between **A** and **B** can modify messages

We need to trust 4 individual parts

Dolev-Yao: Trusted Areas (TLS)

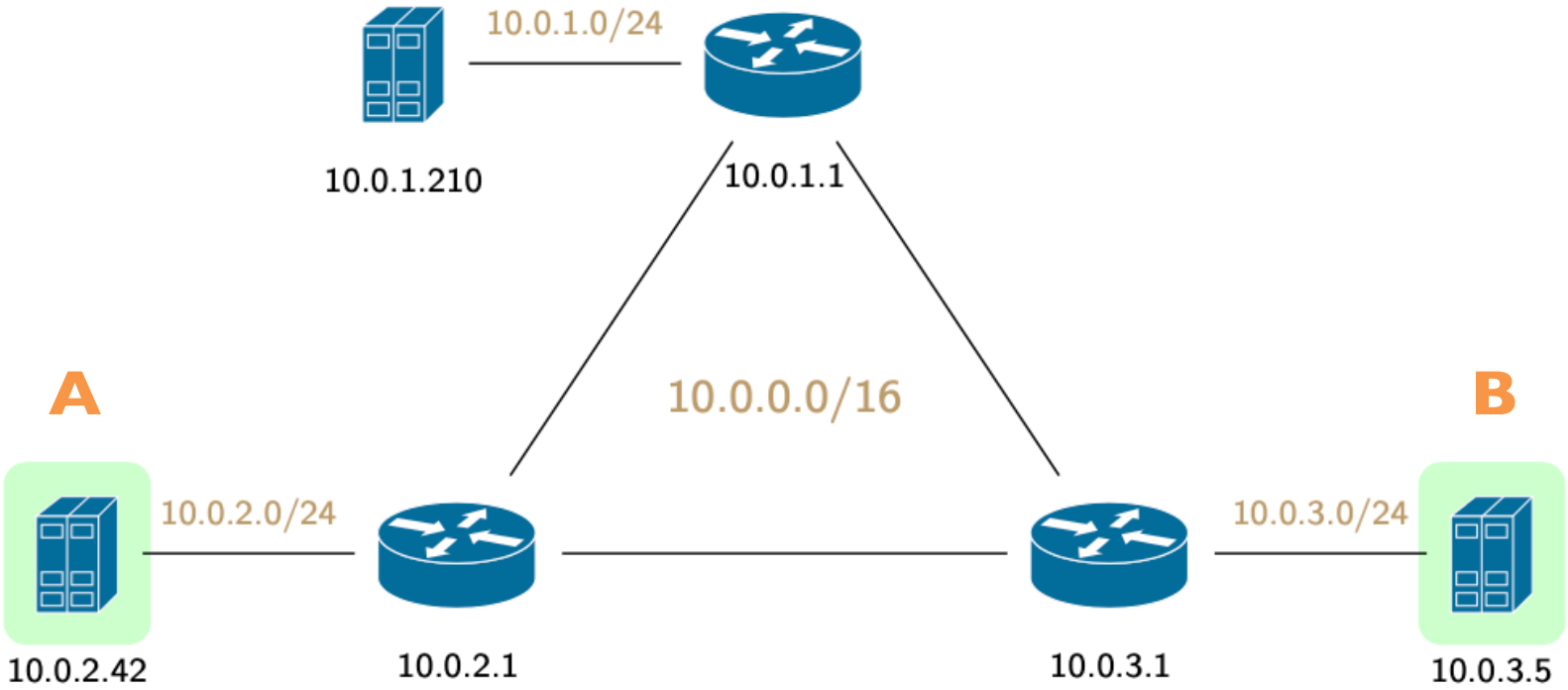


TLS' end-to-end principle:

We only need to trust **A** and **B**

routers between A and B cannot see their communication

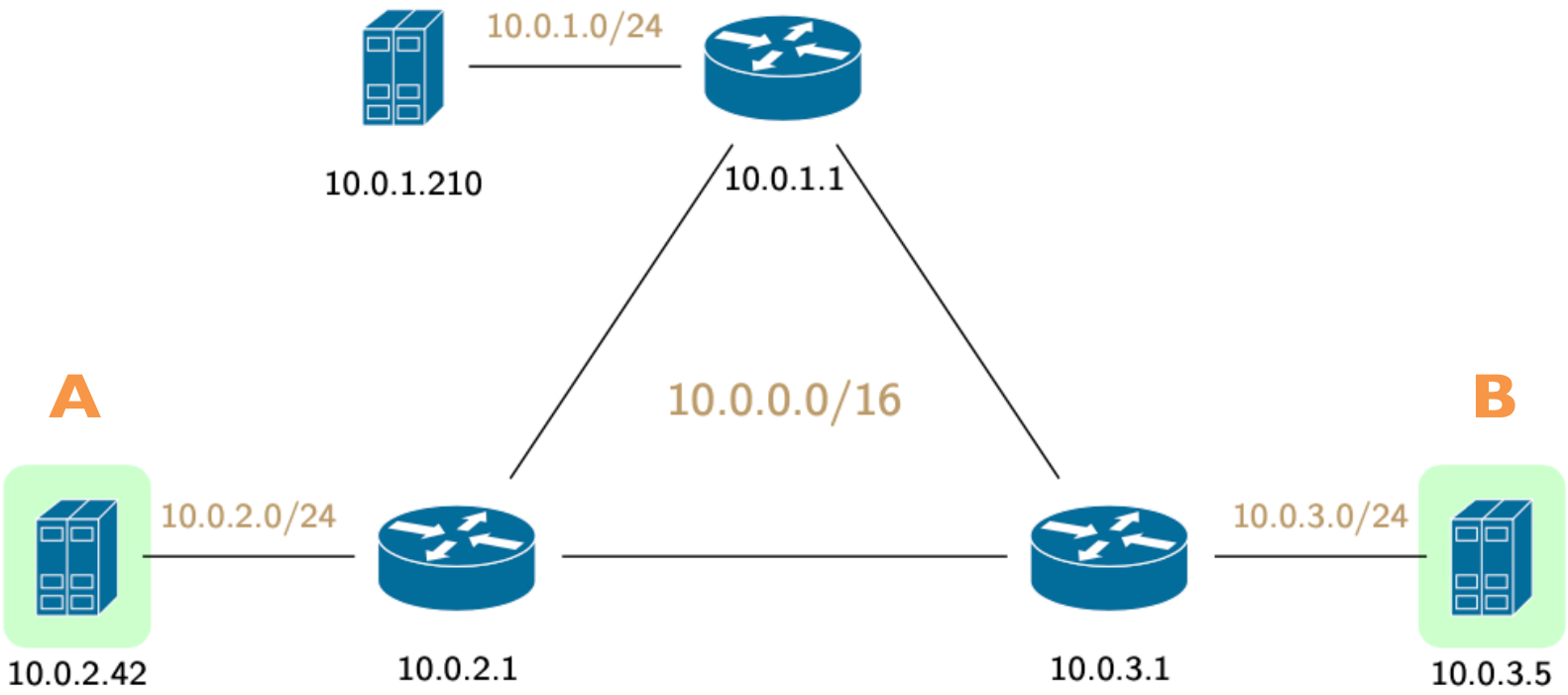
Dolev-Yao: Trusted Areas (TLS)



TLS' end-to-end principle:

Client **A:** If an attacker gains access to **A**'s machine, and can run arbitrary code then they can read and modify messages that A sends to **B**

Dolev-Yao: Trusted Areas (TLS)



TLS' end-to-end principle:

server **B:** if an attacker steals the server's certificate then the attacker can impersonate the server and bad things can happen

SSL vs. TLS

Ruins of Pompeii, Italy

Castle Neuschwanstein, Germany

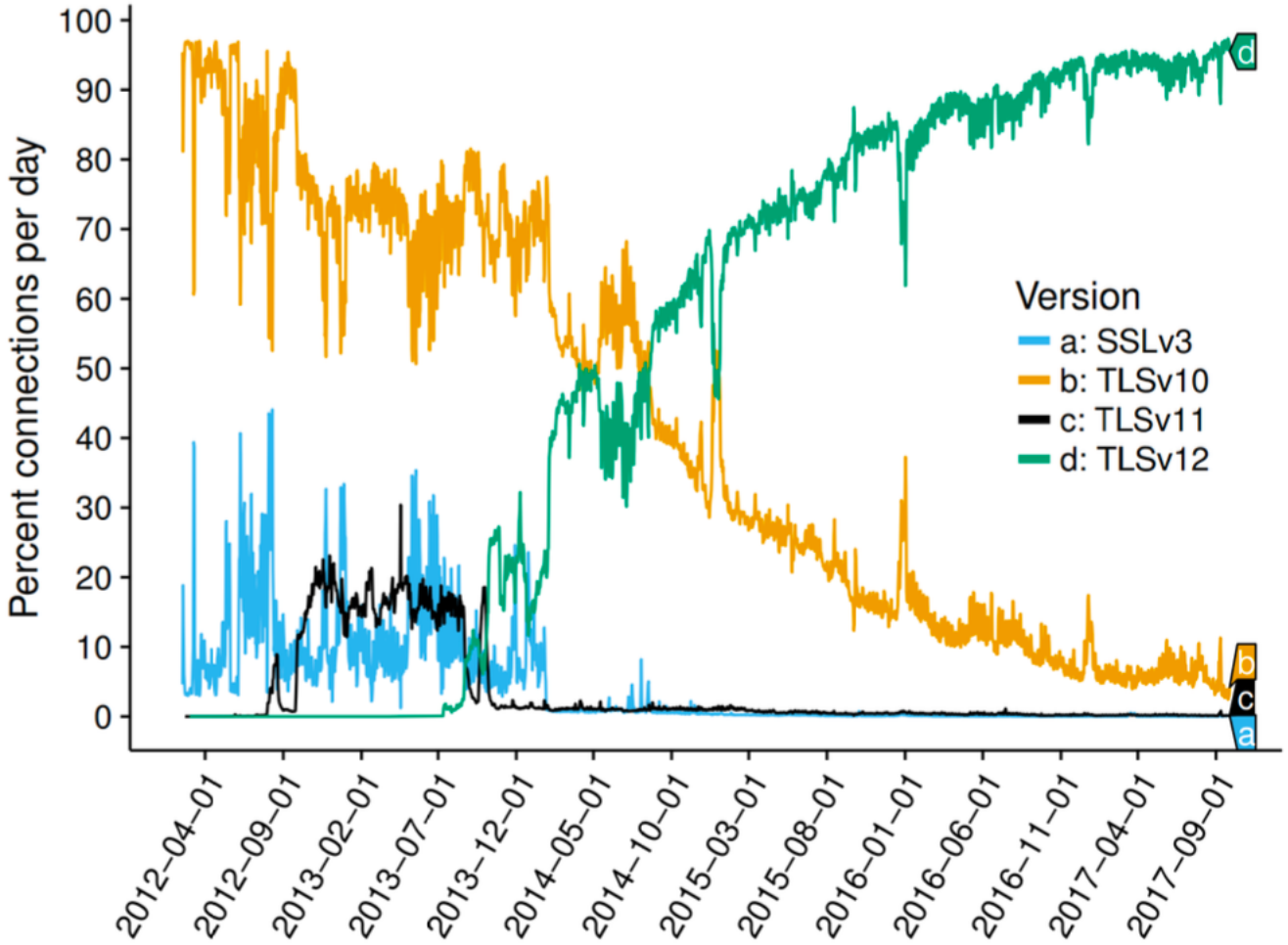
SSL

TLS

The "S" in HTTPS.

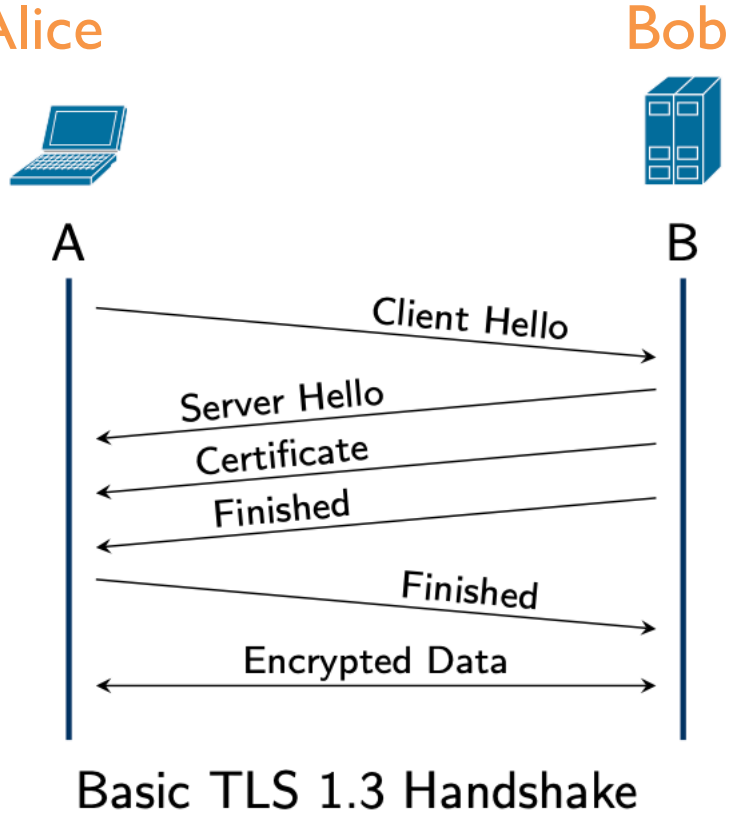
HTTPS means HTTP over TLS

TLS Versions



Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017.

TLS Handshake



Basic TLS 1.3 Handshake

Confidentiality: Client and server negotiate "master key" used to encrypt data.

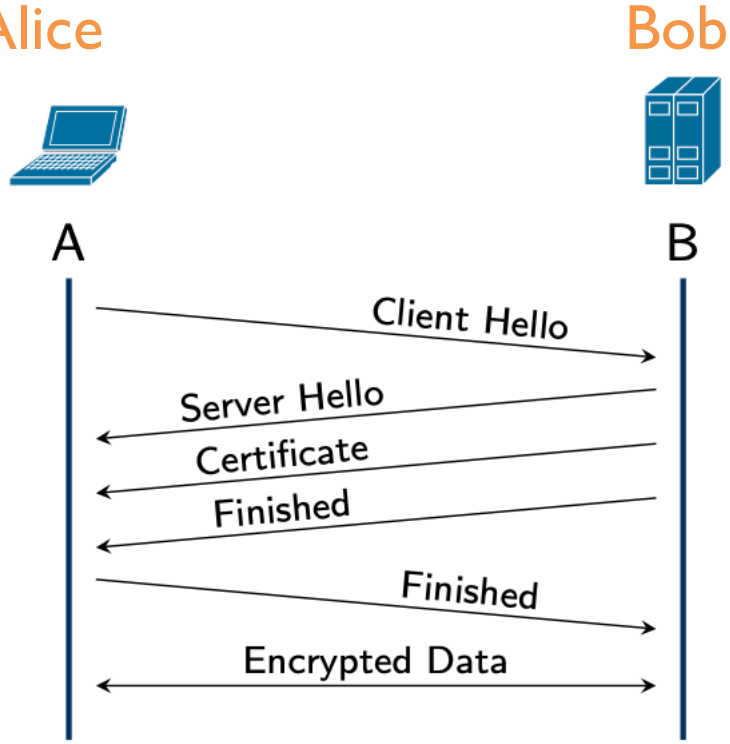
Integrity: Encryption primitives ensure that content modifications can be detected.

Authenticity: Server proves its identity by presenting certificate.
TLS allows for clients to do this, too.

Client's Hello

- TLS version
- cryptographic primitives supported by the client
- domain name of the web site
- everything required by the key-exchange

TLS Handshake



Basic TLS 1.3 Handshake

Confidentiality: Client and server negotiate "master key" used to encrypt data.

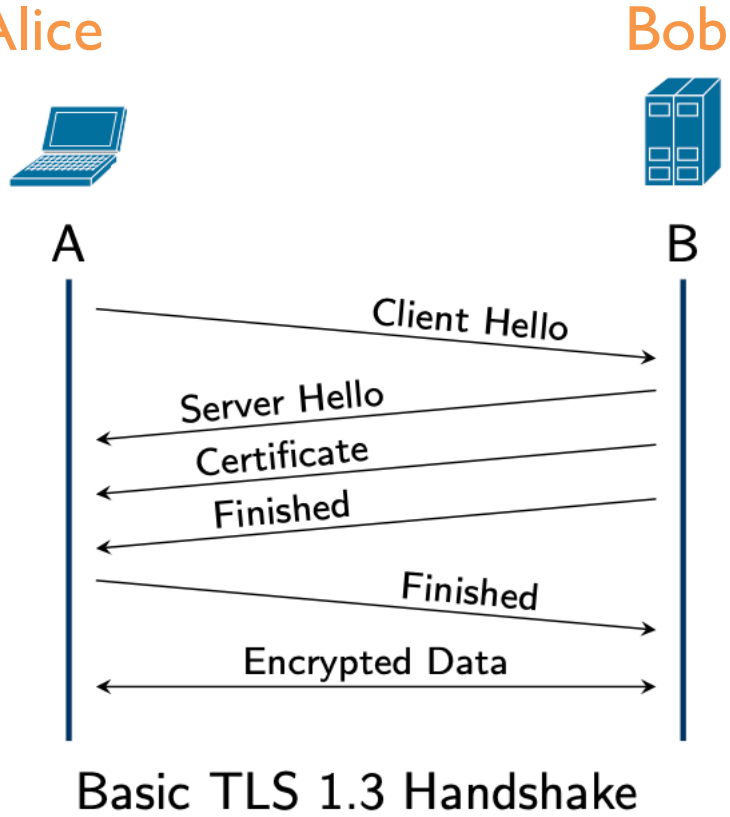
Integrity: Encryption primitives ensure that content modifications can be detected.

Authenticity: Server proves its identity by presenting certificate.
TLS allows for clients to do this, too.

Server's certificate

- The client uses the certificate to prove that the server is the right entity
- otherwise, the client can end up communicating with an attacker

TLS Handshake



Basic TLS 1.3 Handshake

Confidentiality: Client and server negotiate "master key" used to encrypt data.

Integrity: Encryption primitives ensure that content modifications can be detected.

Authenticity: Server proves its identity by presenting certificate.
TLS allows for clients to do this, too.

Client

- It does not prove its identity, so the server does not know to whom it's talking.
- There is no TLS client's authentication, but client authenticates on the application layer, e.g. servers sends client a cookie over a secure connection

TLS cryptography

TLS Cryptography

- Client and server agree on the cryptographic primitives during the handshake.
- Cryptography is hard:
  - Weak/broken ciphersuites
  - Implementation flaws
  - Cryptographic oracles
  - Downgrade attacks

Downgrade attacks take advantage of a system's backward compatibility to force it to a less secure mode of operation.

e.g. from HTTPS to HTTP

Roadmap

1. Transport Layer Security (TLS)
2. **Certificate Authority (CA)**
3. Man-in-the-Middle attacks

establishing trust

How does "asymmetric cryptography" work?

A. 2 keys: 1 private key and 1 public key
B. the private key is kept private
C. the public key is available to anyone
D. the private key is used to decrypt
E. the public key is used to encrypt
F. the private key is the only key that can decrypt data encrypted with the public

establishing trust

Establishing Trust

How does Alice know if she is talking to Bob?

Alice is the client
Bob is the server

Bob presents a certificate!

Establishing Trust

How does Alice know if she is talking to Bob?

Alice is the client
Bob is the server

we need a certified entity to sign this certificate!!

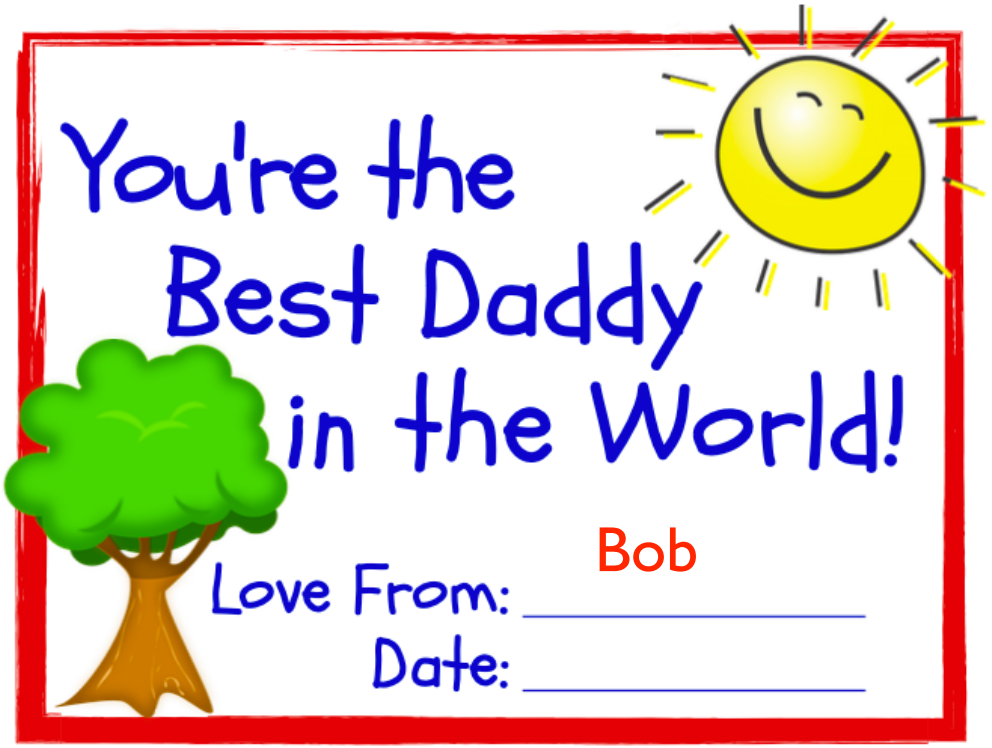


Illustration: Olivia Lasting

Establishing Trust

How does Alice know if she is talking to Bob?

Alice is the client
Bob is the server

copy of Bob's certificate

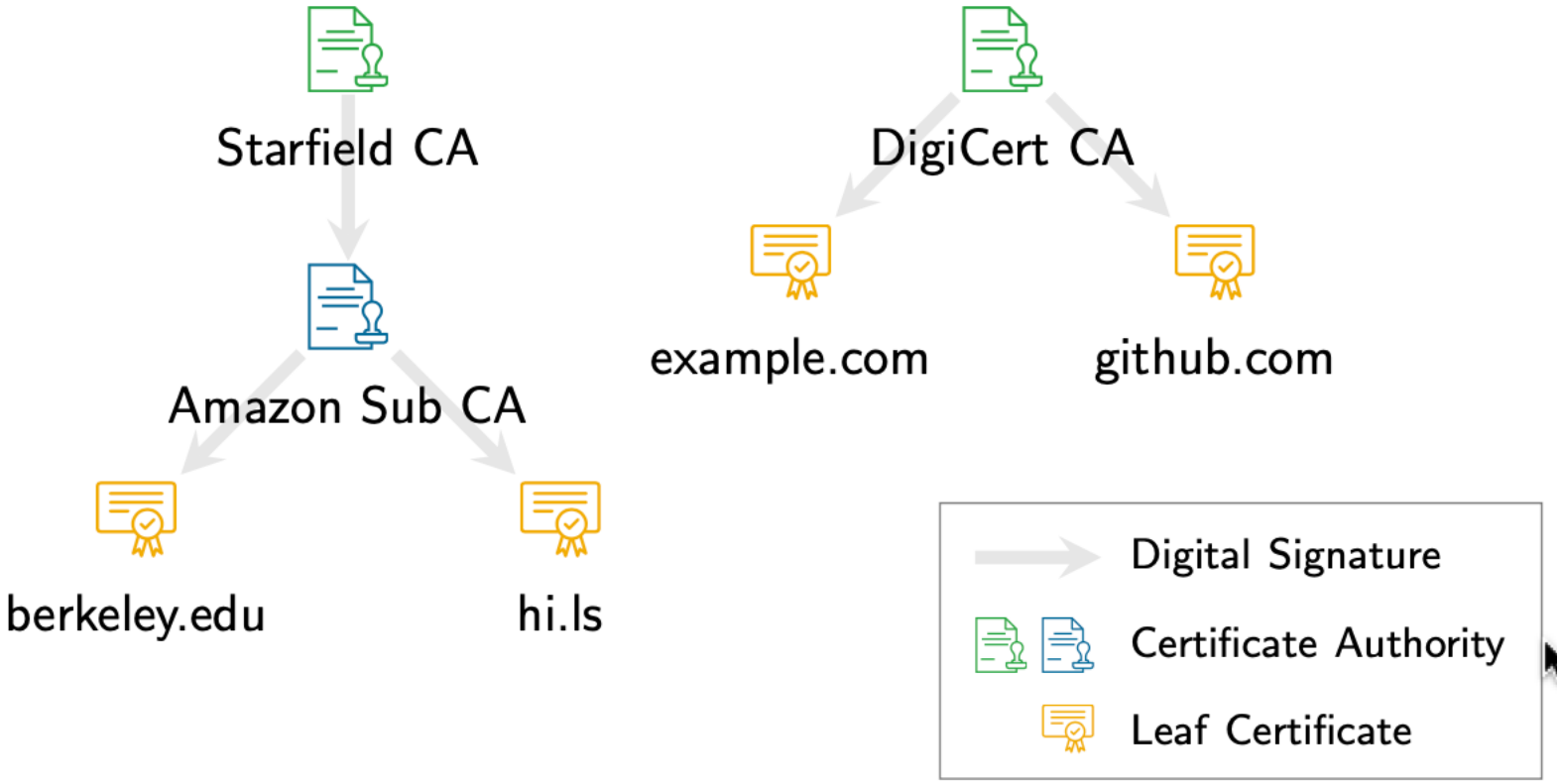
Website: `example.com`
Valid not before: May 4th, 2018
Valid not after: May 4th, 2019
Website Public Key: `c5 55 45 d6..`

Verified by: DigiCert Inc
Verifier Signature: `60 5f b3 66..`

CA

Internet Public Key Infrastructure

PKI - public key infrastructure



CA ~ Certification Authority

Your cell phone has a preloaded set of CAs that it trusts!!

TLS Trust Model Issues

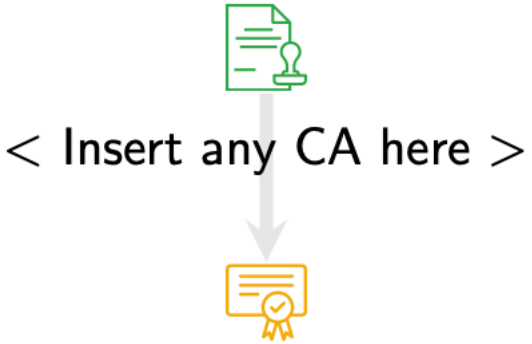
CAs

Certificates

Users

CA Compromise

Any CA can issue certificates for any domain and will be trusted.

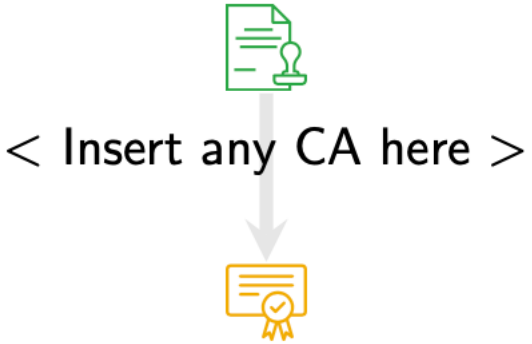


berkeley.edu

The compromise of a CA puts the whole ecosystem at risk.

CA Compromise

Any CA can issue certificates for any domain and will be trusted.



berkeley.edu

The compromise of a CA puts the whole ecosystem at risk.

If an attacker wants a certificate for Google.com,
the attacker can target the weakest CA in the ecosystem

certificate authority selection

- Every operating system and some browsers ship with their own set of trusted certificate authorities (CAs).
  - Windows: $\approx$ 130 trusted organizations.
- Root CAs can delegate trust to intermediary authorities.
  - $>$ 650 organizations
  - $>$ 50 jurisdictions
- Attacker can exploit the weakest link.

Certificate Authority Selection

certificate authority selection

Who can sign certificates?

- Trusted certificate authorities (CAs) are selected by operating system vendors and some browsers, most notably Firefox.
- CAs apply for inclusion.
- This gives operating systems and browsers some negotiation power on behalf of users.
- On the other side: CAs might be too big to fail.

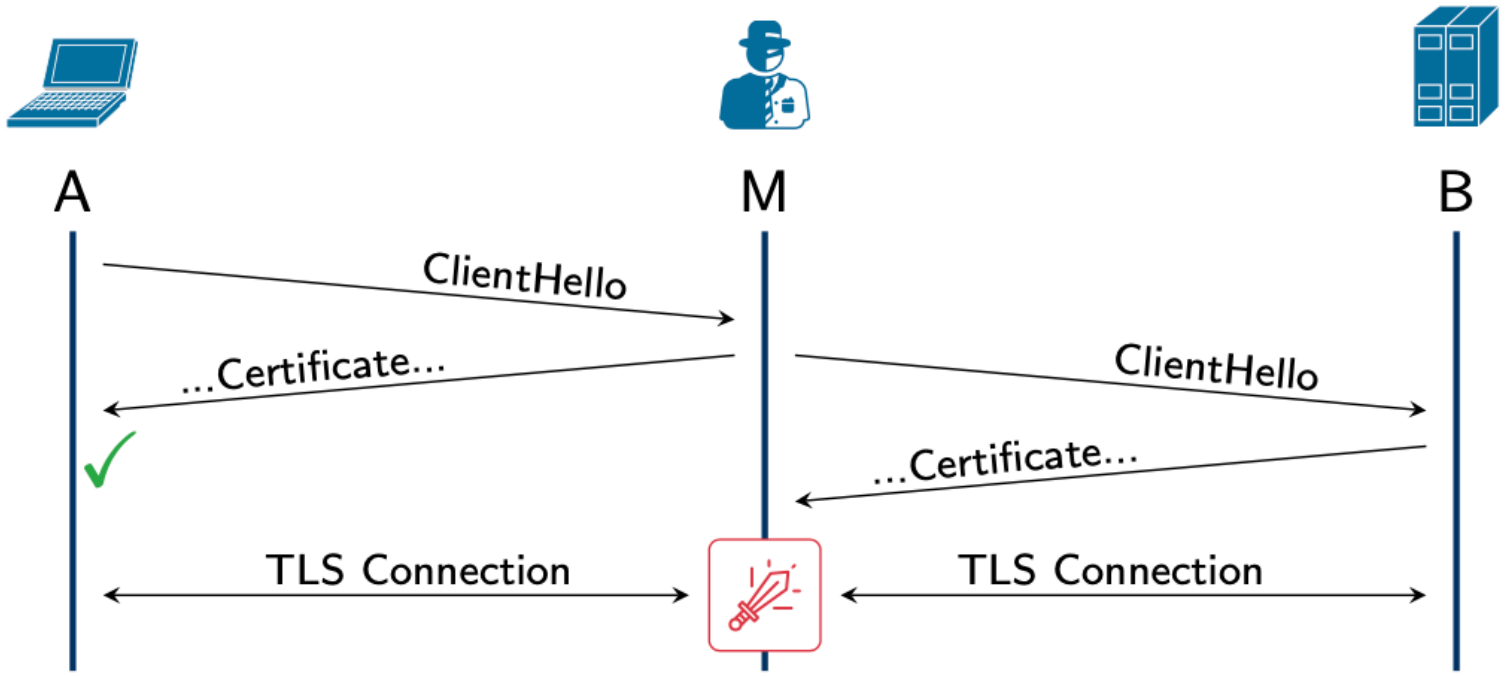
Roadmap

1. Transport Layer Security (TLS)
2. Certificate Authority (CA)
3. Man-in-the-Middle attacks

Man-in-the-middle attacks

How can we compromise protection goals?

Violate end-to-end principle with man-in-the-middle attack.



Attacker's Bucket List

1. Interception software
   - mitmproxy: `https://mitmproxy.org/`
2. A trusted certificate
   - Consent of client or server.
   - Trust model weakness.
   - Vulnerability in certificate verification.

TLS in Non-Browser Software (2012)

The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software

Martin Georgiev
The University of Texas at Austin

Subodh Iyengar
Stanford University

Suman Jana
The University of Texas at Austin

Rishita Anubhai
Stanford University

Dan Boneh
Stanford University

Vitaly Shmatikov
The University of Texas

We demonstrate that SSL certificate validation is completely broken in many security-critical applications and libraries.

based on it; Amazon's and PayPal's merchant SDKs responsible for transmitting payment details from e-commerce sites to payment gateways; integrated shopping carts such as osCommerce, ZenCart

shake, when the server presents its public-key certificate. In order for the SSL connection to be secure, the client must carefully verify that the certificate has been issued by a valid certificate authority,

The root causes of these vulnerabilities are badly designed APIs of SSL implementations (such as JSSE, OpenSSL, and GnuTLS) [...]

perils and pitfalls of SSL certificate validation in software based on these APIs and present our recommendations.

cessors such as PayPal and Amazon, (3) logging instant messenger clients into online services, and (4) authenticating servers to mobile

TLS in Non-Browser Software (2018)

Current State of OpenSSL:

- Certificate hostname is not verified by default.
- No functionality to use OS root certificate store.
- No automated revocation checks.
- TLS version specification is prone to error.

TLS outside of browsers is in a very sad state of affairs.