

HTTPS, TLS, and the CA Ecosystem

How do we translate?

Cryptographic Primitives

Symmetric Encryption

RSA

PKI

HMAC

Certificate

Public Key

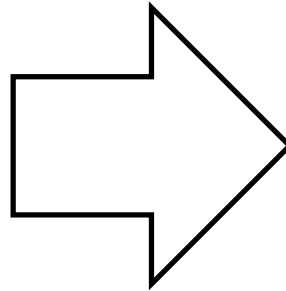
RC4

Diffie-Hellman

DSA

ECDSA

Asymmetric Encryption



Objectives:

Message Integrity

Confidentiality

Authentication

for Websites

How do we translate?

Cryptographic Primitives

Symmetric
Encryption

RSA

PKI

HMAC

Certificate

Public Key

RC4

Diffie-Hellman

DSA

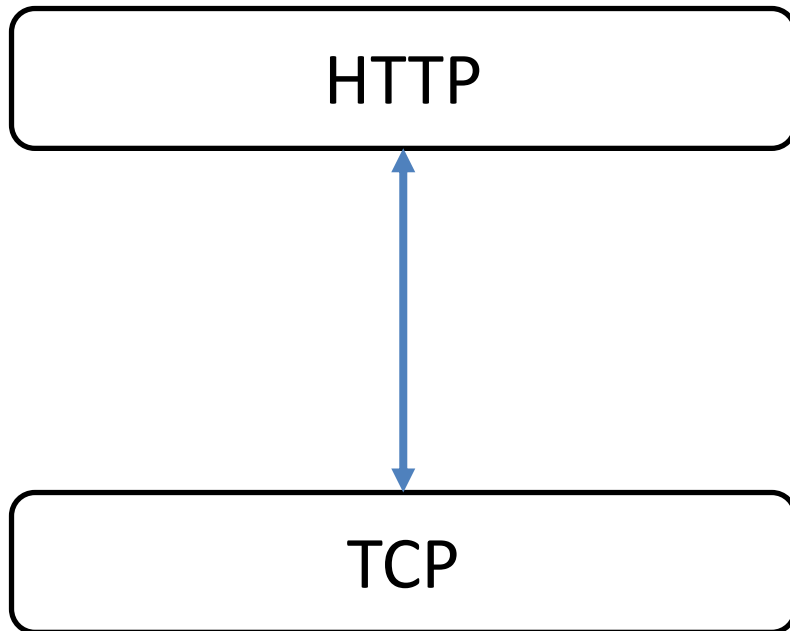
ECDSA

Asymmetric
Encryption

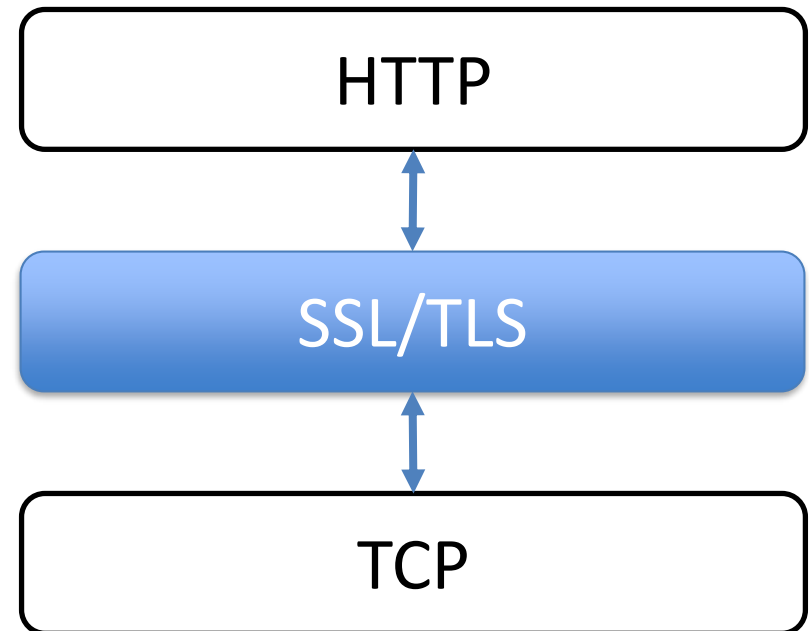
HTTPS
Protocol

Adding Crypto to HTTP

Normal HTTP Transaction



HTTPS Transaction



SSL/TLS

Arguably the most important (and widely used) cryptographic protocol on the Internet

Almost all popular encrypted protocols (except SSH) use SSL/TLS for transport encryption

HTTPS, POP3, IMAP, SMTP, FTP, NNTP, XMPP (Jabber),
OpenVPN, SIP (VoIP), ...

When you need an encrypted socket for your application, use SSL/TLS

SSL/TLS

SSL (Secure Socket Layer) – Netscape, late 1990s

- Version 2.0: Broken, don't use
(disabled by default in modern browsers)
- Version 3.0: Broken, don't use
(starting to be disabled by browsers)

TLS (Transport Layer Security) – IETF Standard

- 1.0, 1.1: Outdated, prefer not to use
- 1.2: Commonly used
- 1.3: Standard being defined now

TLS Threat Model



Adversarial Network

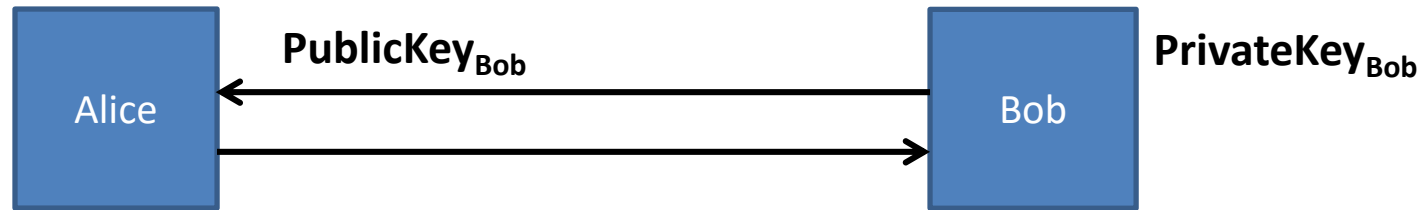
- Attacker controls infrastructure (routers, DNS, wireless access points)
- Passive attacker: only eavesdrops
- Active attacker: eavesdrops, injects, blocks, and modifies packets

Examples: Internet Café, Hotel, CSE

Does not protect against:

- Intruder on server
- Malware on client

Review: Public-key Crypto



Bob generates **$\text{PrivateKey}_{\text{Bob}}$** , **$\text{PublicKey}_{\text{Bob}}$** and distributes public key to Alice.

Alice can encrypt messages to Bob:

She uses **$\text{PublicKey}_{\text{Bob}}$** to encrypt message,

Bob can decrypt using **$\text{PrivateKey}_{\text{Bob}}$**

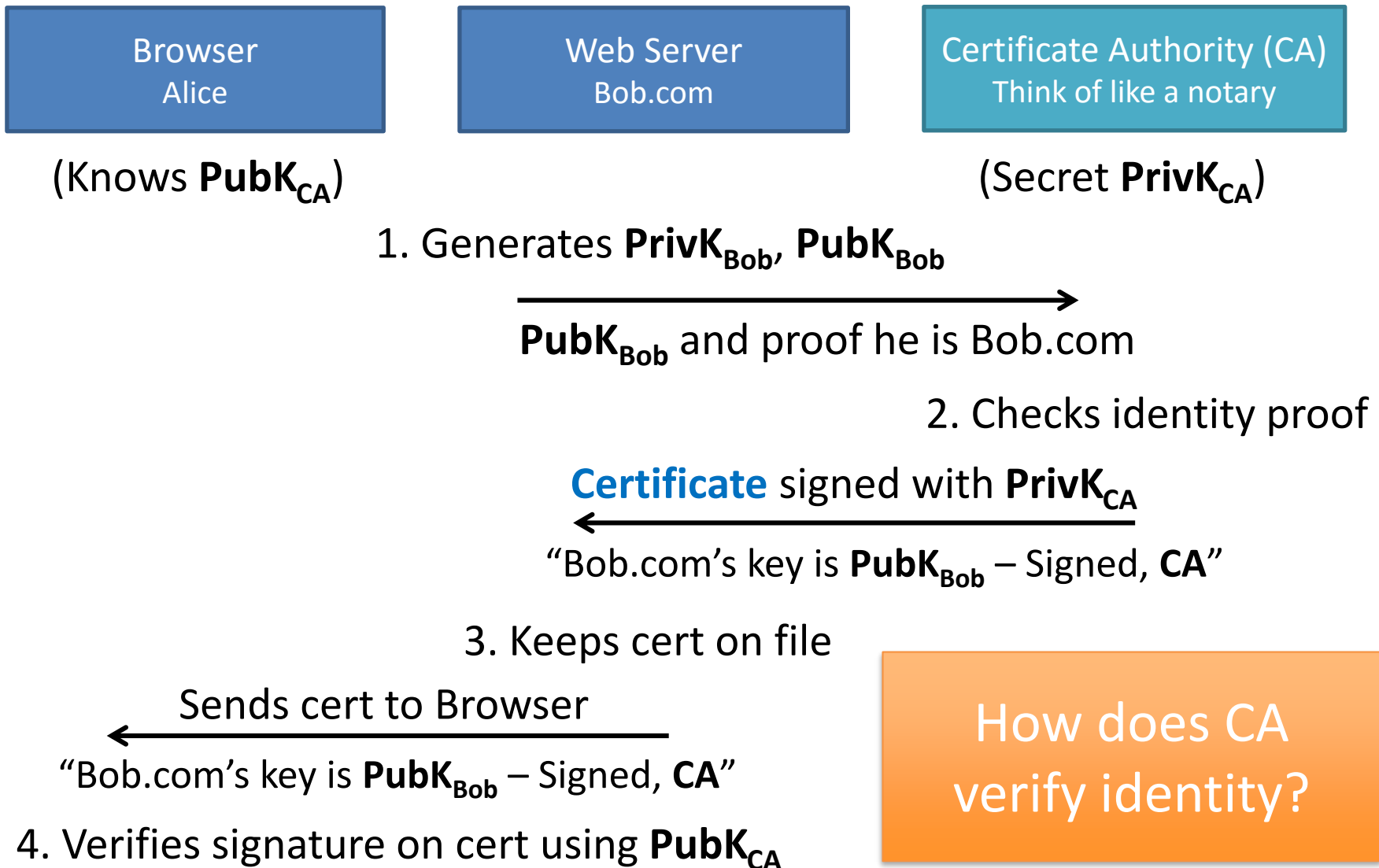
Bob can sign messages that Alice can verify:

He uses **$\text{PrivateKey}_{\text{Bob}}$** to generate signature,

Alice can verify using **$\text{PublicKey}_{\text{Bob}}$**

Certificates

How does the browser obtain the server's public key?



x.509 Certificates

Subject: CN=www.google.com

Issuer: C=US/O=Google Inc/CN=Google Internet Authority

Serial Number: 01:b1:04:17:be:22:48:b4:8e:1e:8b:a0:73:c9:ac:83

Validity Period: Jul 20 2015 - [Jul 19 2016](#)

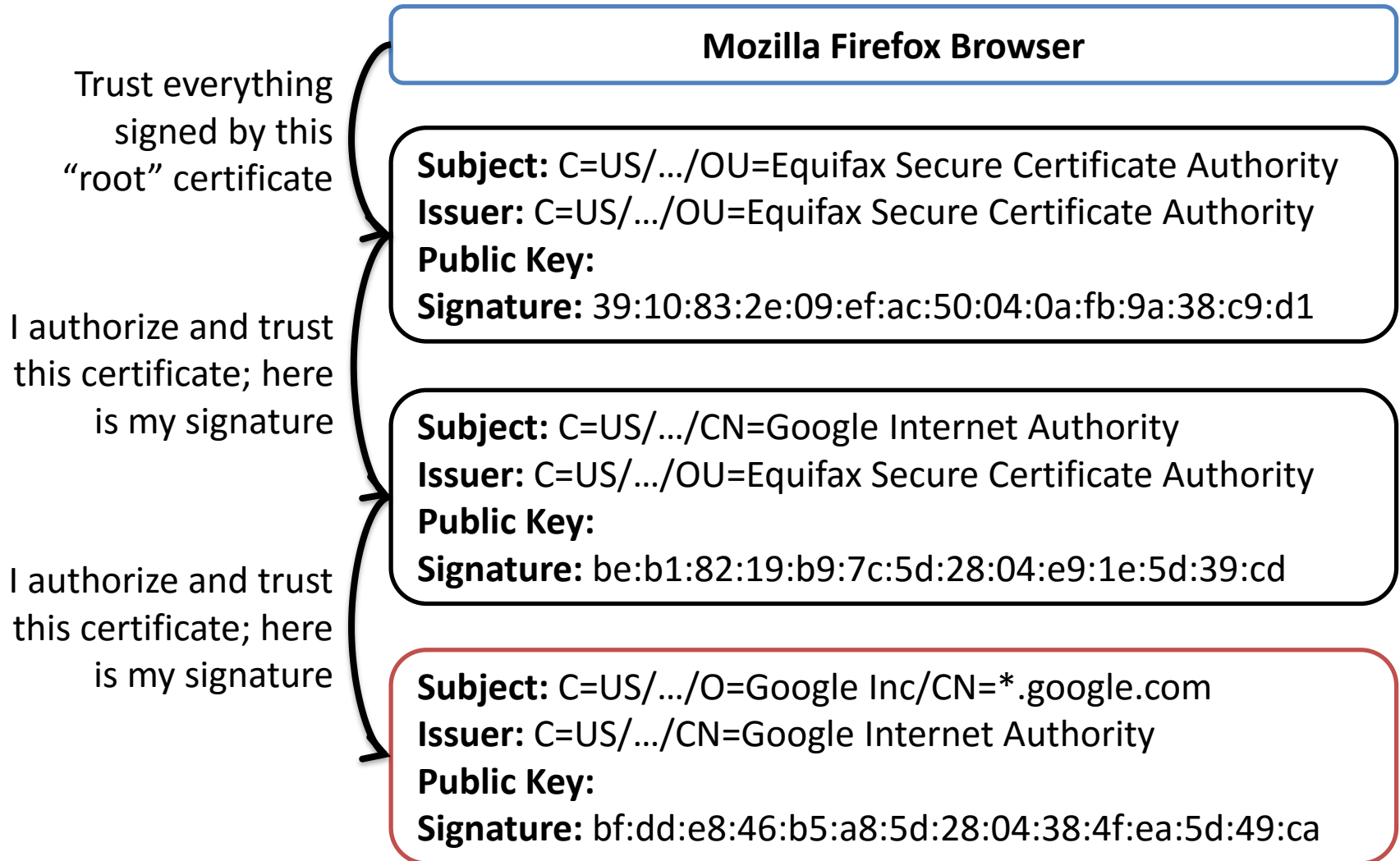
Public Key Algorithm: rsaEncryption

Public Key: 43:1d:53:2e:09:ef:dc:50:54:0a:fb:9a:f0:fa:14:58:ad:a0:81:b0:3d
7c:be:b1:82:19:b9:7c3:8:04:e9:1e5d:b5:80:af:d4:a0:81:b0:b0:68:5b:a4:a4
:ff:b5:8a:3a:a2:29:e2:6c:7c3:8:04:e9:1e5d:b5:7c3:8:04:e9:39:23:46

Signature Algorithm: sha1WithRSAEncryption

Signature: 39:10:83:2e:09:ef:ac:50:04:0a:fb:9a:f0:fa:14:58:ad:a0:81:b0:3d
7c:be:b1:82:19:b9:7c3:8:04:e9:1e5d:b5:80:af:d4:a0:81:b0:b0:68:5b:a4:a4
:ff:b5:8a:3a:a2:29:e2:6c:7c3:8:04:e9:1e5d:b5:7c3:8:04:e9:1e5d:b5

Certificate Chains



Certificate Authority Ecosystem

Each browser trusts a set of CAs

CAs can sign certificates for new CAs

CAs can sign certificates for *any* web site

If a single CA is compromised, then the entire system is compromised

We ultimately place our complete trust of the Internet in the weakest CA

Getting a Certificate



Client

Server

The TLS “handshake”

Client

Server

Client Hello: Here's what I support and a *random*



Client

Server

Client Hello: Here's what I support and a *random*

```
graph LR; Client -- "Client Hello: Here's what I support and a random" --> Server;
```

The diagram shows a single arrow pointing from the Client to the Server, representing the initial 'Client Hello' message.

Server Hello: Chosen Cipher

```
graph LR; Server -- "Server Hello: Chosen Cipher" --> Client;
```

The diagram shows a single arrow pointing from the Server back to the Client, representing the 'Server Hello' message.

Certificate: Here is my "X509 Certificate"

```
graph LR; Server -- "Certificate: Here is my X509 Certificate" --> Client;
```

The diagram shows a single arrow pointing from the Server back to the Client, representing the transmission of the X509 Certificate.

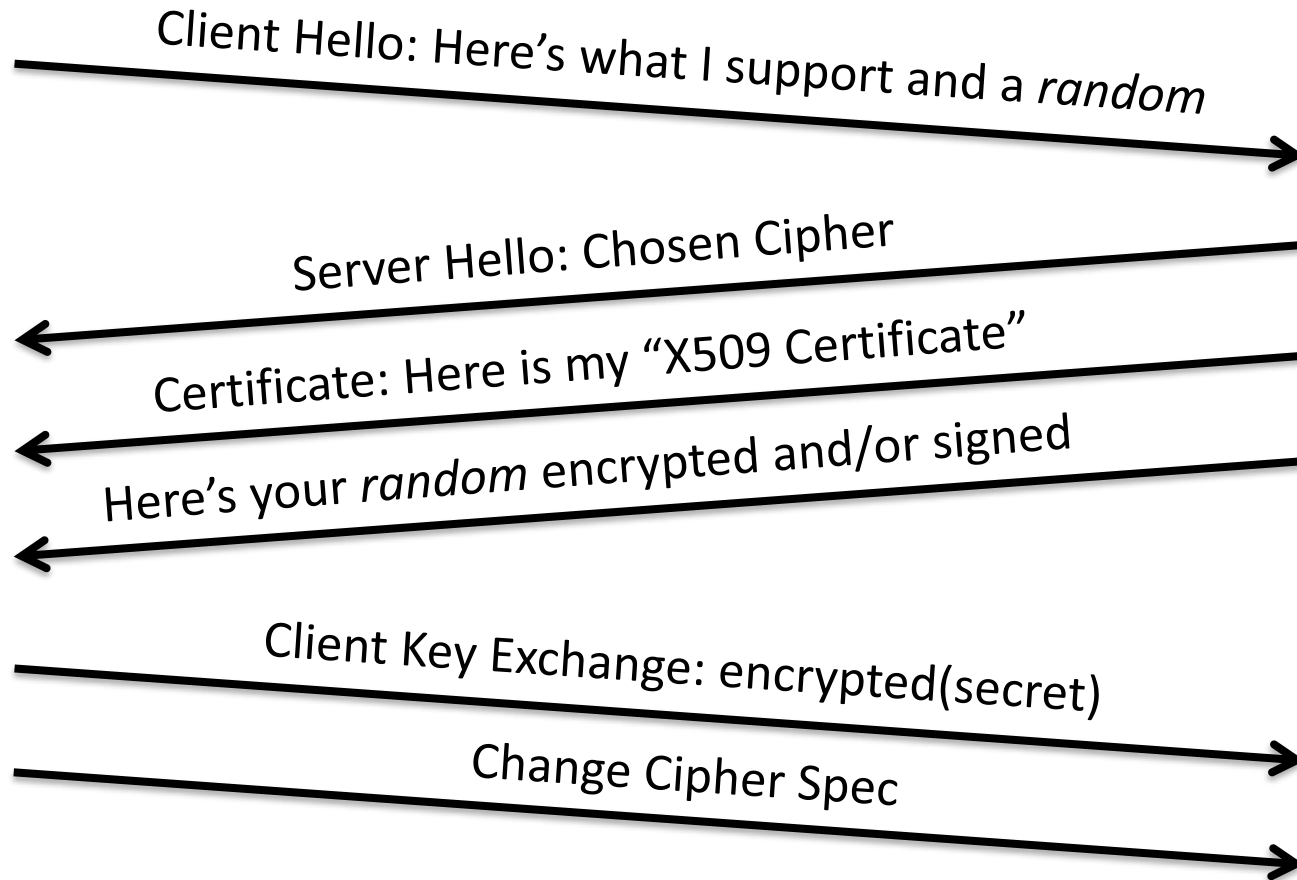
Here's your *random* encrypted and/or signed

```
graph LR; Server -- "Here's your random encrypted and/or signed" --> Client;
```

The diagram shows a single arrow pointing from the Server back to the Client, representing the encrypted random value and the signature.

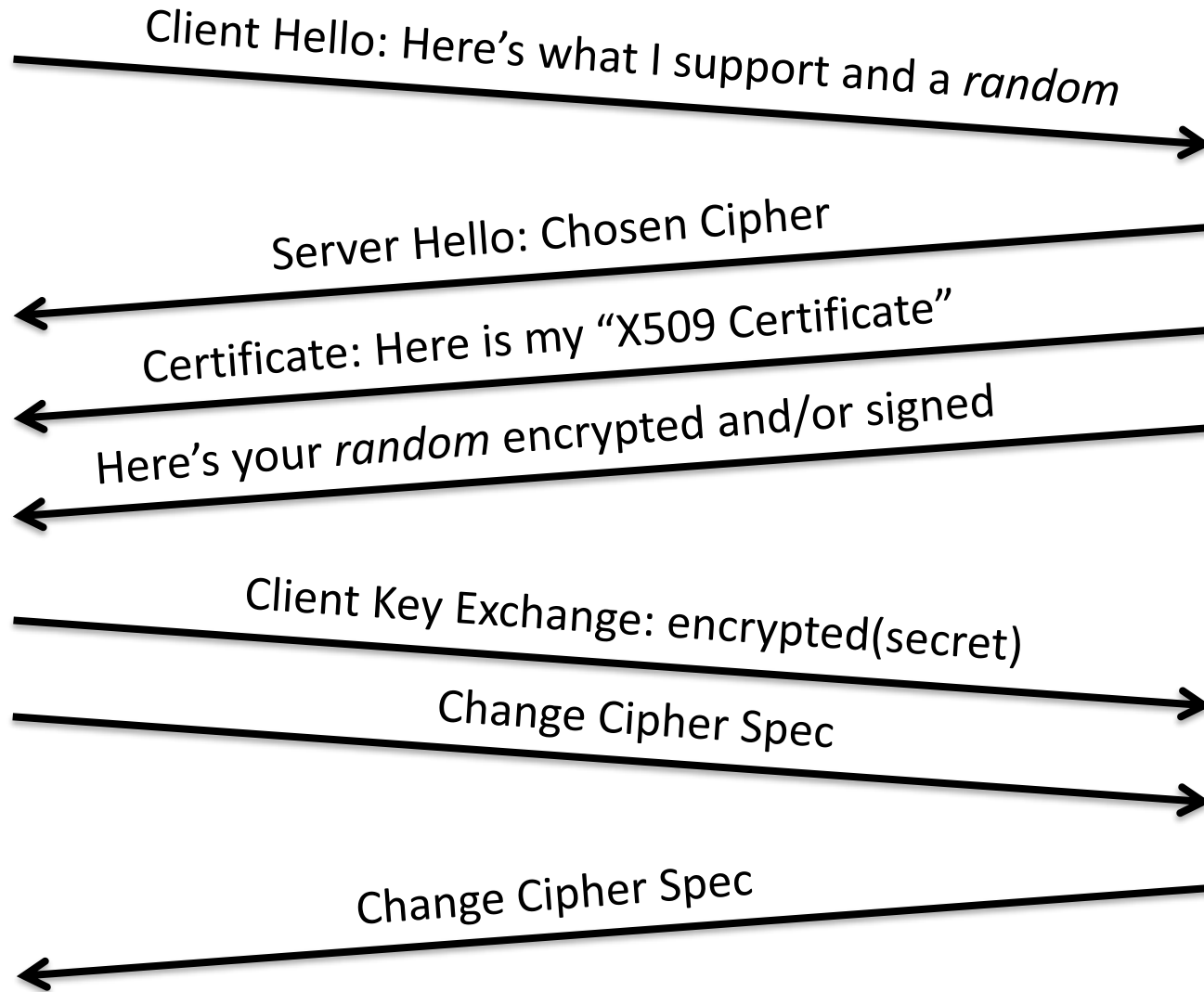
Client

Server



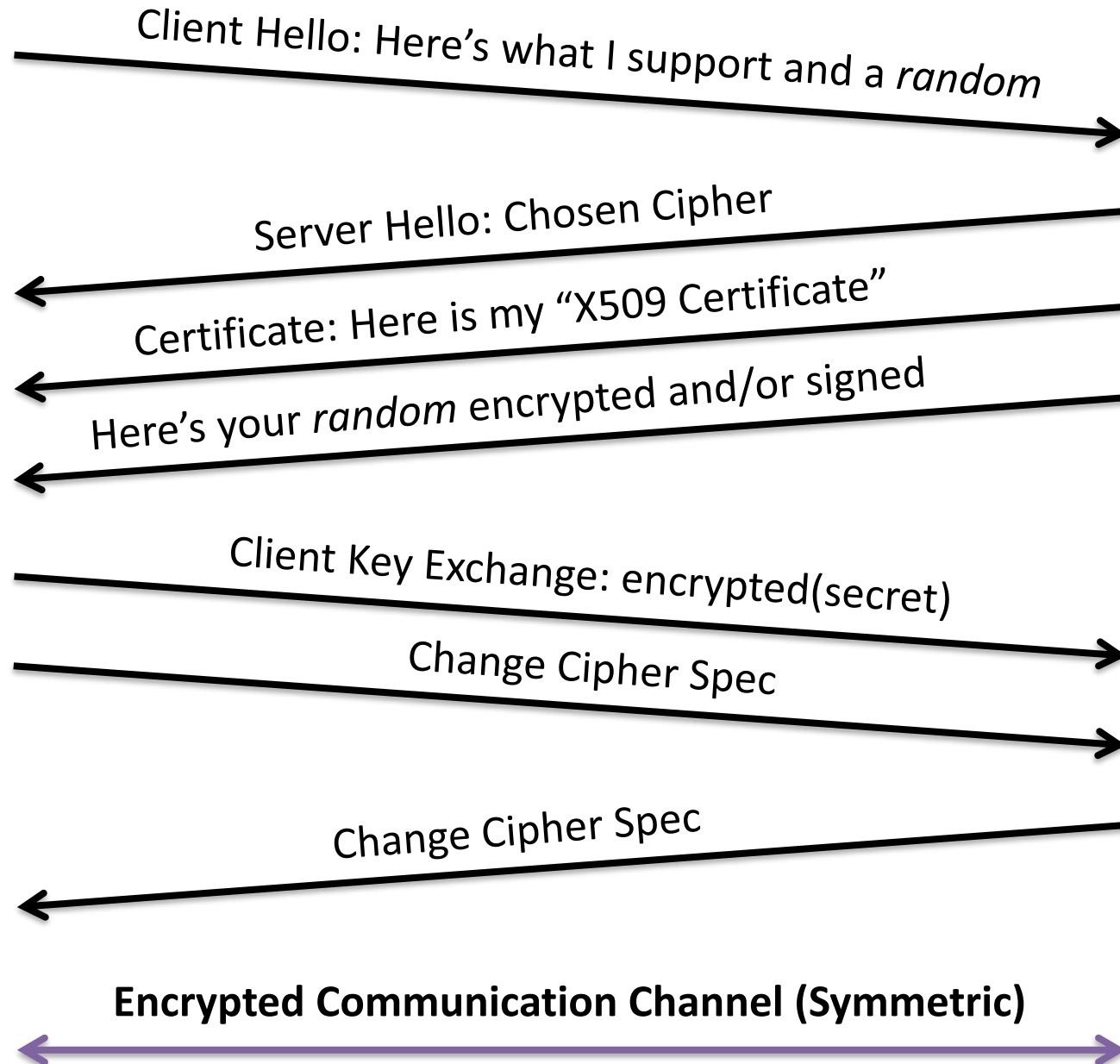
Client

Server



Client


Server



Cipher Suites

DHE - RSA - AES256 - SHA

Ephemeral
Key Exchange



```
graph BT; A[Ephemeral Key Exchange] --> C[DHE - RSA - AES256 - SHA]; B[Key Exchange] --> C; D[Data Transfer Cipher] --> C; E[Message Digest] --> C;
```

Key Exchange

Data Transfer
Cipher

Message Digest

HTTPS User Interface (Tricky!)

Goal: Help users authenticate site

Lock icon — Displayed when all elements of page fetched using HTTPS
HTTPS cert must be issued by a CA trusted by browser
HTTPS cert is valid (e.g., not expired or revoked)
CommonName in cert matches domain in URL
Must check all of these or else a problem!

Extended Validation (EV) certificates

Green Bar in Firefox with name of the organization.
(Mostly for banks and large e-commerce sites)

CA does extra work to verify identity — expensive, more secure?

Invalid certificate warnings

(Deliberately hard to override, users do anyway)

Goals



Confidentiality (Symmetric Crypto)



Message Integrity (HMACs)



Authentication (Public Key Crypto)

Attacks Against HTTPS

1. Attacking the Browser UI

Picture-in-picture Attack

Spoof the user interface

Attacker page draws fake browser window with lock icon

Semantic Attacks

Example: micros0ft.com

Example: International character sets contains chars that look similar to English letters

Example: Hiding domain later in long URL

"www.bank.com/accounts/login.php@attacker.com"

Invalid Certs

Expired, Common Name != URL, unknown CA (e.g., self-signed)

Warning overload — Many users will click through

Accepting enables man-in-the-middle attack (active adversary)

Defense: Make it hard for users to click through (Firefox takes 4 clicks!)

2. Attacking Site Design

ssl_strip attack

Many sites:

- browse via HTTP, switch to HTTPS for checkout
- connect via HTTP, switch to HTTPS for login
- Simple attack: Transparent proxy strips out redirects, relays HTTP to HTTPS on server

Defenses?

Mixed Content attack

Page loads over HTTPS but contains content over HTTP (common)

e.g. JavaScript, Flash

Active attacker can tamper with HTTP content to hijack session

Defense: Browser warnings, ("This page contains insecure content"), but inconsistent and often ignored

3. Attacking the CA Ecosystem

Distributed architecture: *Nobody knows*
complete set of trusted intermediate CAs...
(1,733 visible in UMich Internet-wide scans CAs)

History of CAs being hacked (e.g., **DigiNotar**)

Oops! Korea gave every elementary school,
library, and agency a CA certificate (1,324)

Luckily, were invalid due to a higher-up constraint

DigiNotar

- DigiNotar ***was*** a Dutch Certificate Authority
- On June 10, 2011, *.**google.com** cert was issued to an attacker and subsequently used to orchestrate MITM attacks in Iran
- Nobody noticed the attack until someone found the certificate in the wild...

DigiNotar Contd.

- DigiNotar later admitted that dozens of fraudulent certificates were created
- Google, Microsoft, Apple and Mozilla all revoked the root DigiNotar certificate
- Dutch Government took over DigiBotar
- DigiBotar went bankrupt

Search

About 274,000 results (0.24 seconds)

Everything

Images

Maps

Videos

News

Shopping

More

All results

Related searches

More search tools

[-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...](#)
[pastebin.com/TbaeU93m](#)

19 Apr 2010 - ... the difference. Copied. -----BEGIN RSA PRIVATE KEY-----
MIICXwIBAAKBpemis1ePqHkVN9KaGBESjV6zBrlsZc+XQYT1S/Va9R/4SAXoYpl ...

[-----BEGIN RSA PRIVATE KEY - Pastebin.com - #1 paste tool since ...](#)
[pastebin.com/sC7bGw00](#)

18 Apr 2010 - ... difference. Copied. -----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvxBalHzKMewLvmlr1ptID1gO7EWGFyudzOAHLqm3+0+gpPbk ...

[site:pastebin.com "-----BEGIN RSA PRIVATE KEY-----" - Posterous](#)
[cdeviers.posterous.com/sitepastebincom-begin-rsa-private-key-google](#)

20 Apr 2010 - Apr 19, 2010 ... -----BEGIN RSA PRIVATE KEY-----
MIICXwIBAAKBpemis1ePqHkVN9KaGBESjV6zBrlsZc+XQYT1S/Va9R/4SAXoYpl ...

[help/en/howto/ftp - Cyberduck](#)
[trac.cyberduck.ch/wiki/help/en/howto/ftp](#)

Private keys containing a DSA or RSA private key in PEM format are supported (look
for -----BEGIN DSA PRIVATE KEY----- or -----BEGIN RSA PRIVATE KEY----- ...

[SSH access with a private RSA key \[Archive\] - VanDyke Software For ...](#)
[forums.vandyke.com/archive/index.php/t-2185.html](#)

2 Sep 2011 - -----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQBujdbxyIX4KaQPETTSF/
aOSBwSpZNMjTixUZYq8JkipjM/YpYwpNj1TODzRJf ...

4. Attacking Implementations

Null Prefix Attack, 2009

(x.509 uses Pascal-style strings, browsers use C strings; what if a common name contains “\0”?)

`gmail.com\0.badguy.com`

Apple Goto Fail, Feb. 2014

(Apple SSL bug; skipped certificate check for almost a year!)

OpenSSL Heartbleed, April 2014

(OpenSSL bug; leaked data, including private key!)

Mozilla BERsek, October 2014

(Bug in verifying cert signatures, allowed spoofing certs, probably since the beginning....!)

Takeaways

- Use HTTPS! It's so much better than nothing.
- TLS will keep breaking.
Use it, but don't rely on it exclusively.
- Have a backup plan for times when it's broken.