CSE 433S: Introduction to Computer Security

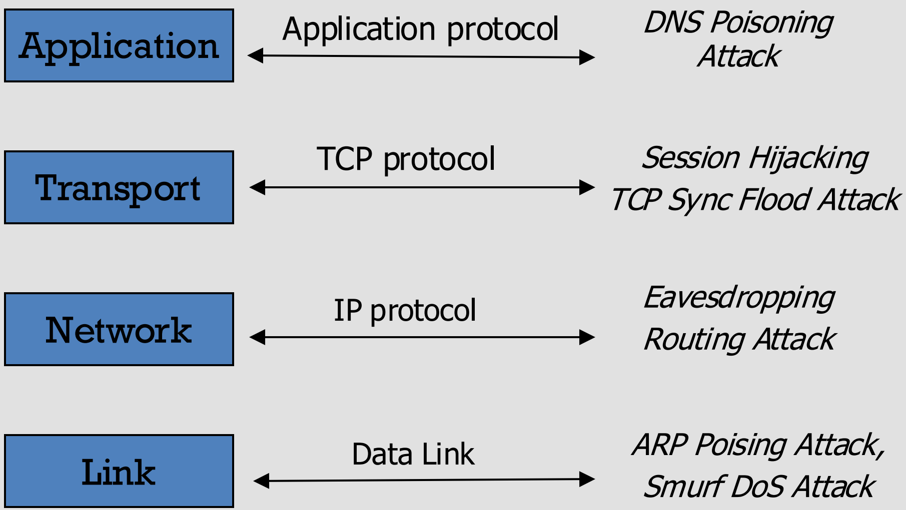
Network Defense Lecture

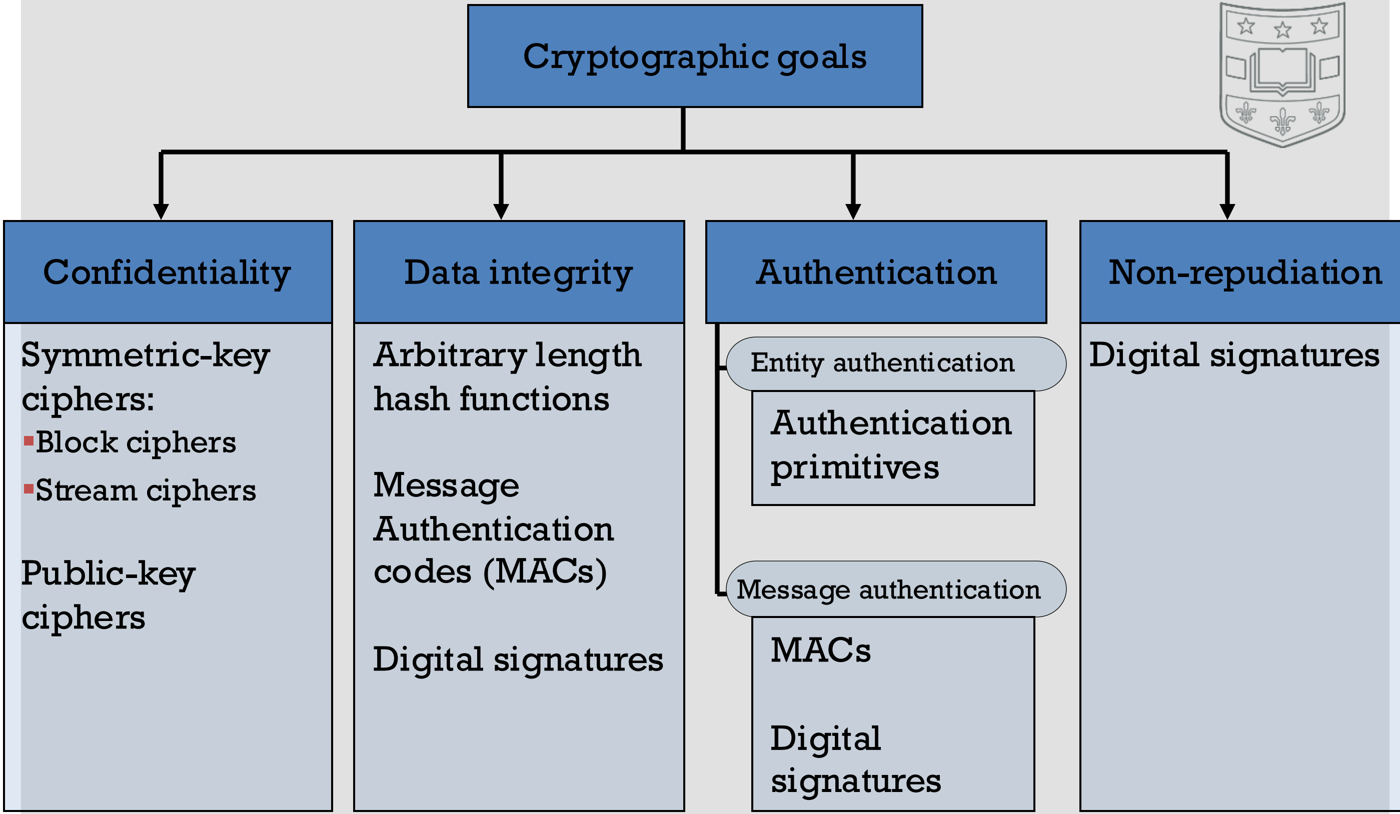
Washington University in St. Louis

Review Questions

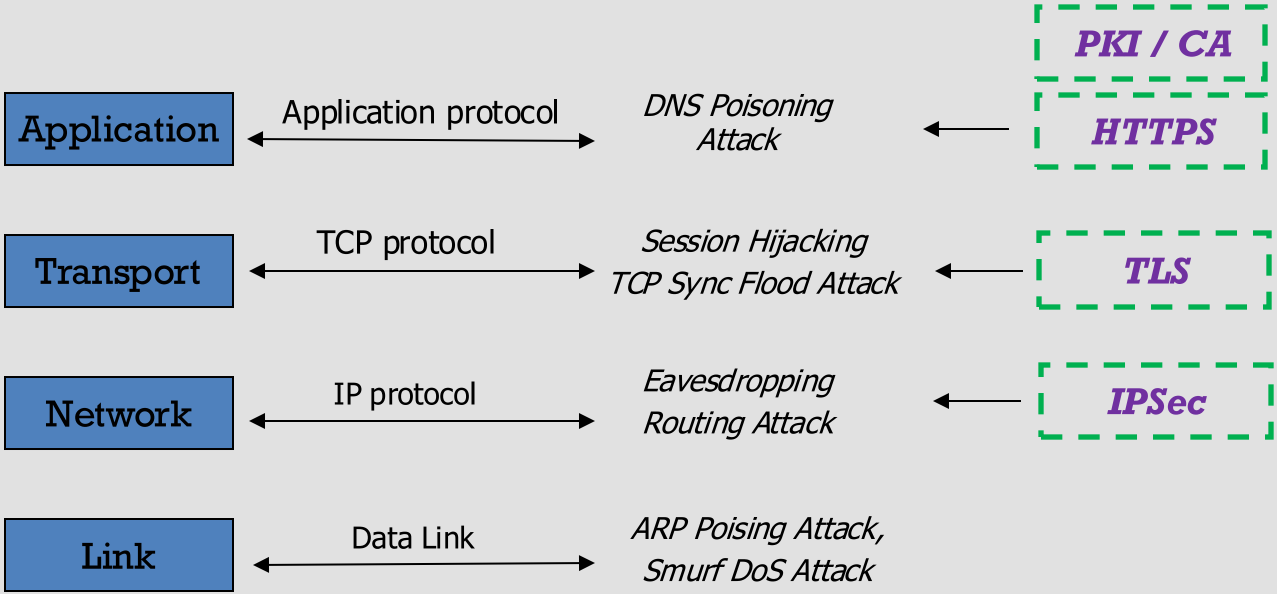
- How many symmetric keys does it take to support secure communications among 4 peers? What are the possible approaches to mitigate this issue?
- What is Diffie-Hellman key exchange, what attack is it vulnerable to, how do you launch that attack?
- What is public key crypto and how is different than symmetric key crypto? What key do Alice use if Alice wants to deliver a secret to Bob.
- In practice, can we use RSA to directly encrypt secret key for communication?
- What is digital signature? What key does Alice use to sign a file that she wants to authenticate and why?

Previously on Introduction to Computer Security – Network Attacks




Cryptographic goals
Confidentiality
Symmetric-key ciphers:
Block ciphers
Stream ciphers
Public-key ciphers
Data integrity
Arbitrary length hash functions
Message Authentication codes (MACs)
Digital signatures
Authentication
Entity authentication
Authentication primitives
Message authentication
MACs
Digital signatures
Non-repudiation
Digital signatures

This module - Network defense



Network Defense

- Public key infrastructure (PKI)
  - How do we know we are talking to the right entity on the web
- Certificate authority (CA)
  - Mechanisms to establish the web of trust
- IPSec
  - Enables confidentiality and integrity protection for communication between two network nodes at IP layer
- TLS (Transport Layer Security)
  - Enables security communication link over TCP

PKI/CA

“Web of trust”

- Obtain public keys from friends in person
  - “Key-signing parties” from the PGP time

- Obtain “certificates” on my public key from my friends

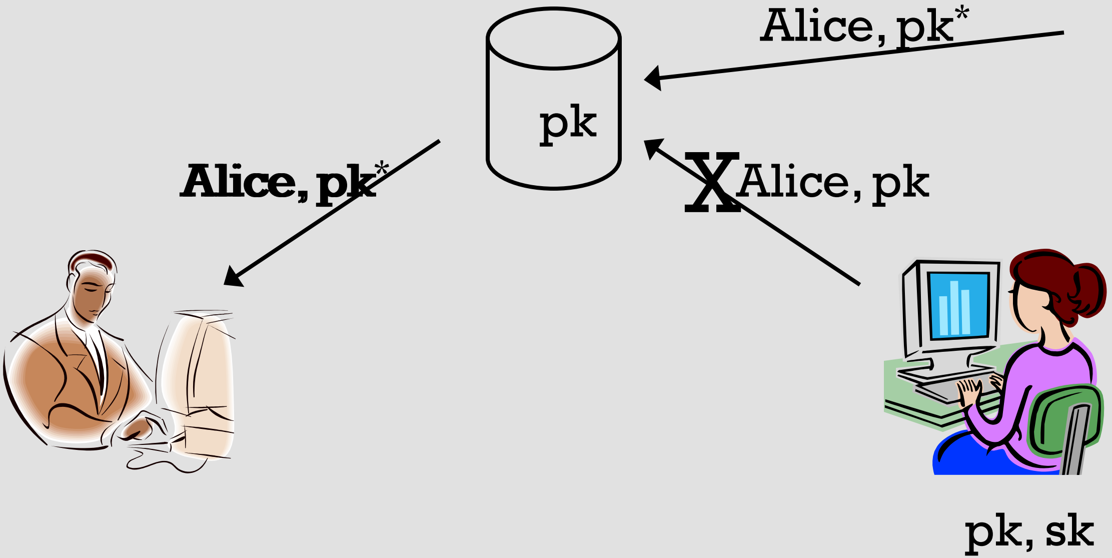
- If A knows pk_B, and B issued a certificate for C, then C can send that certificate to A
  - What trust assumptions are being made here?

Public-key repository

- Store certificates in a central repository
  - E.g., MIT PGP keyserver

- To find Alice’s public key
  - Get all public keys for “Alice,” along with certificates on those keys
  - Look for a certificate signed by someone you trust whose public key you already have

Challenges on Public Key Repository

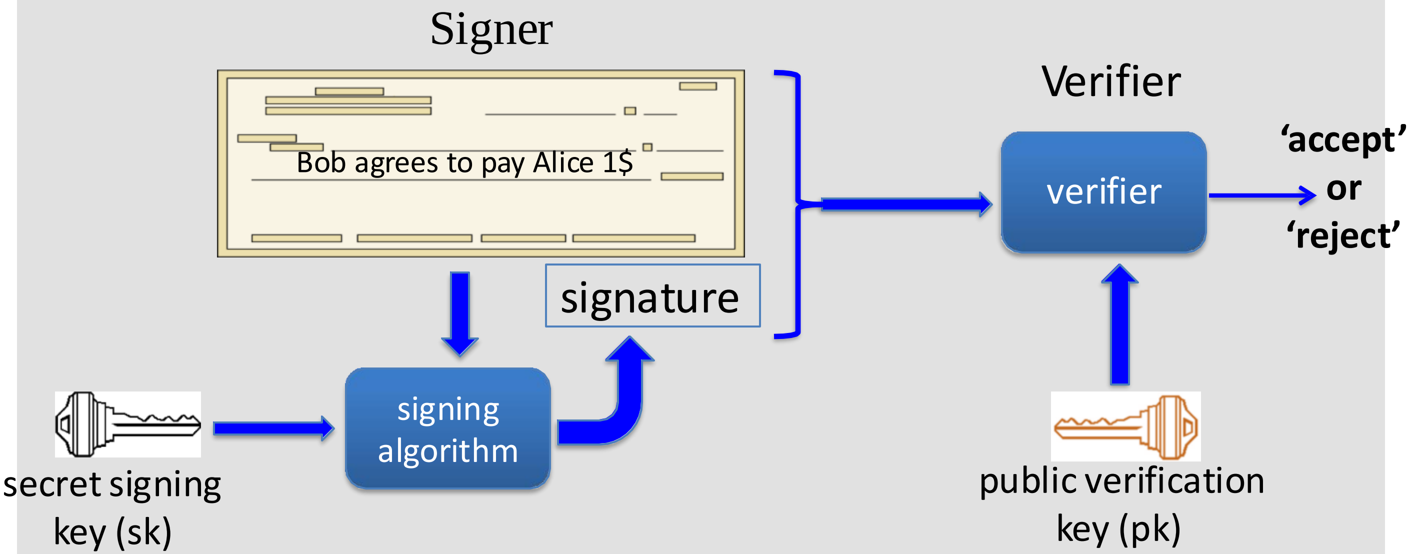


Public Key infrastructure

- Problem: How to determine the correct public key of a given entity
  - Binding between IDENTITY and PUBLIC KEY
- Possible attacks
  - Name spoofing: Eve associates Alice’s name with Eve’s public key
  - Key spoofing: Eve associates Alice’s key with Eve’s name
  - DoS: Eve associates Alice’s name with a nonsensical (bogus) key

Digital signatures

Solution: make signature depend on document



Use signatures for secure key distribution!

- Assume a trusted party with a public key known to everyone
  - CA = certificate authority
  - Public key pk_{CA}
  - Private key sk_{CA}

Use signatures for secure key distribution!

- Alice asks the CA to sign the *binding* (Alice, pk)
  - $\text{cert}_{\text{CA}\rightarrow\text{Alice}} = \text{Sign}_{sk_{\text{CA}}}(\text{Alice, pk})$

- (CA must verify Alice's identity out of band)

Use signatures for secure key distribution!

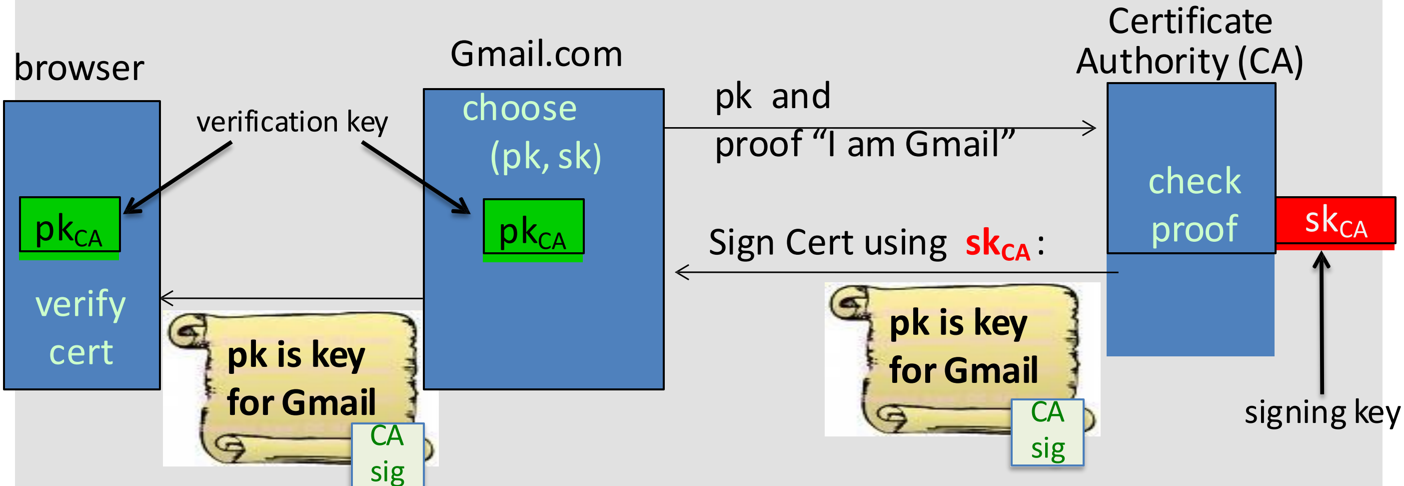
- Bob obtains Alice, pk, and the certificate $\text{cert}_{CA \to Alice}$ …
  - … verifies that $\text{Vrfy}_{pK_{CA}}((\text{Alice}, pk), \text{cert}_{CA \to Alice}) = 1$

- Bob is then assured that pk is Alice’s public key
  - As long as the CA is trustworthy…
    - Honest, and properly verifies Alice’s identity
  - …*and the CA’s private key has not been compromised*

Important application: Certificates

Problem: browser needs server's public-key to setup a session key

Solution: server asks trusted 3rd party (CA) to sign its public-key pk



Server uses Cert for an extended period (e.g. one year)

Chicken-and-egg problem?

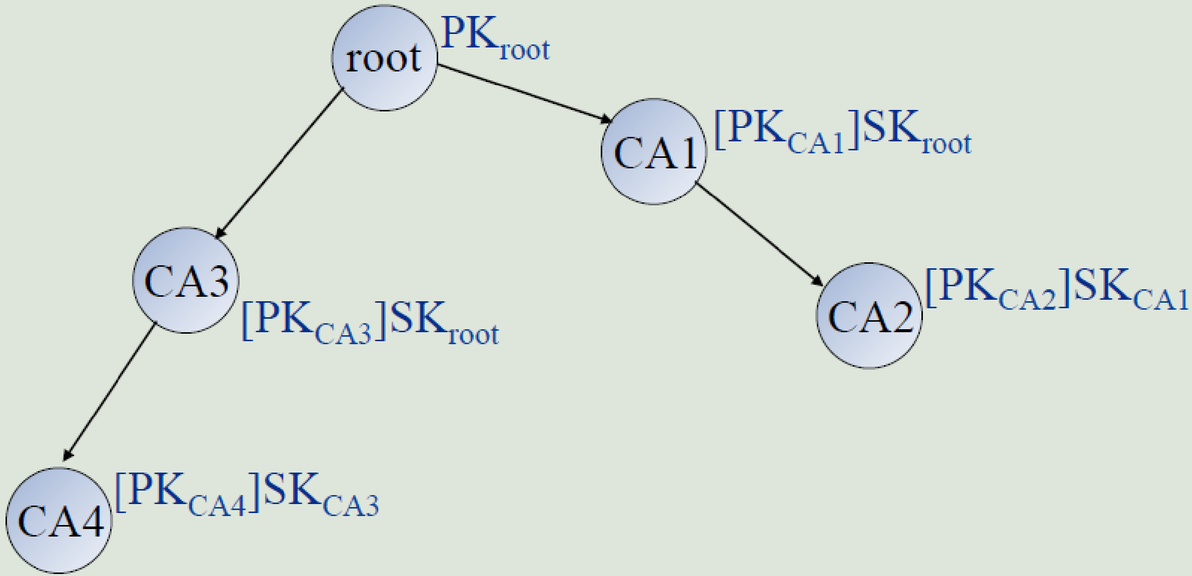
- How does Bob get pk_{CA} in the first place?
- Several possibilities…

“Roots of trust”

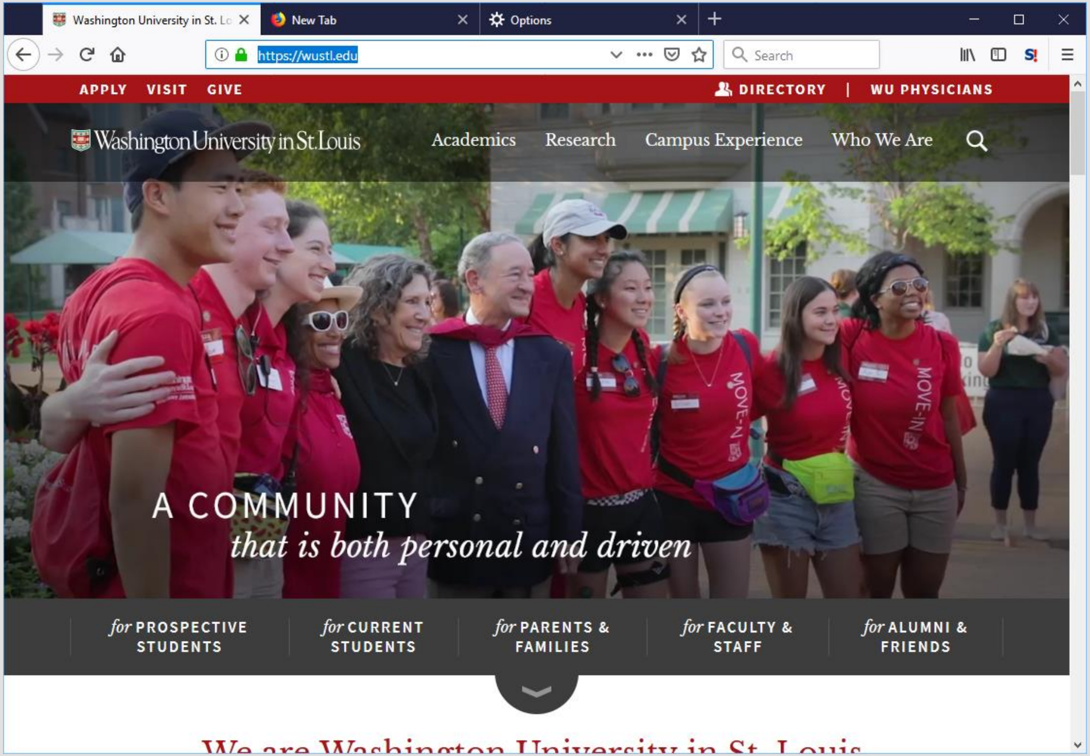
- Bob only needs to securely obtain a small number of CA’s public keys
  - Need to ensure secure distribution only for these few, initial public keys

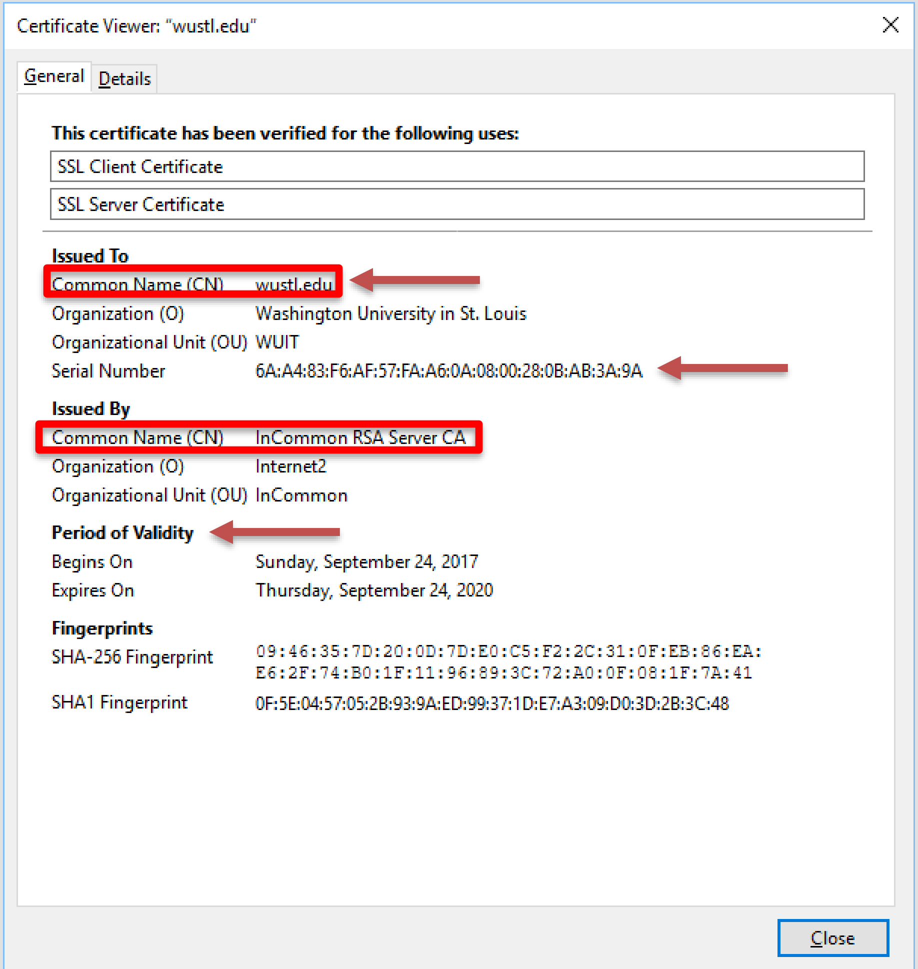
- E.g., distribute as part of an operating system, or web browser
  - Firefox: Tools->Options->Privacy & Security->View certificates->Authorities

Certificate Hierarchy

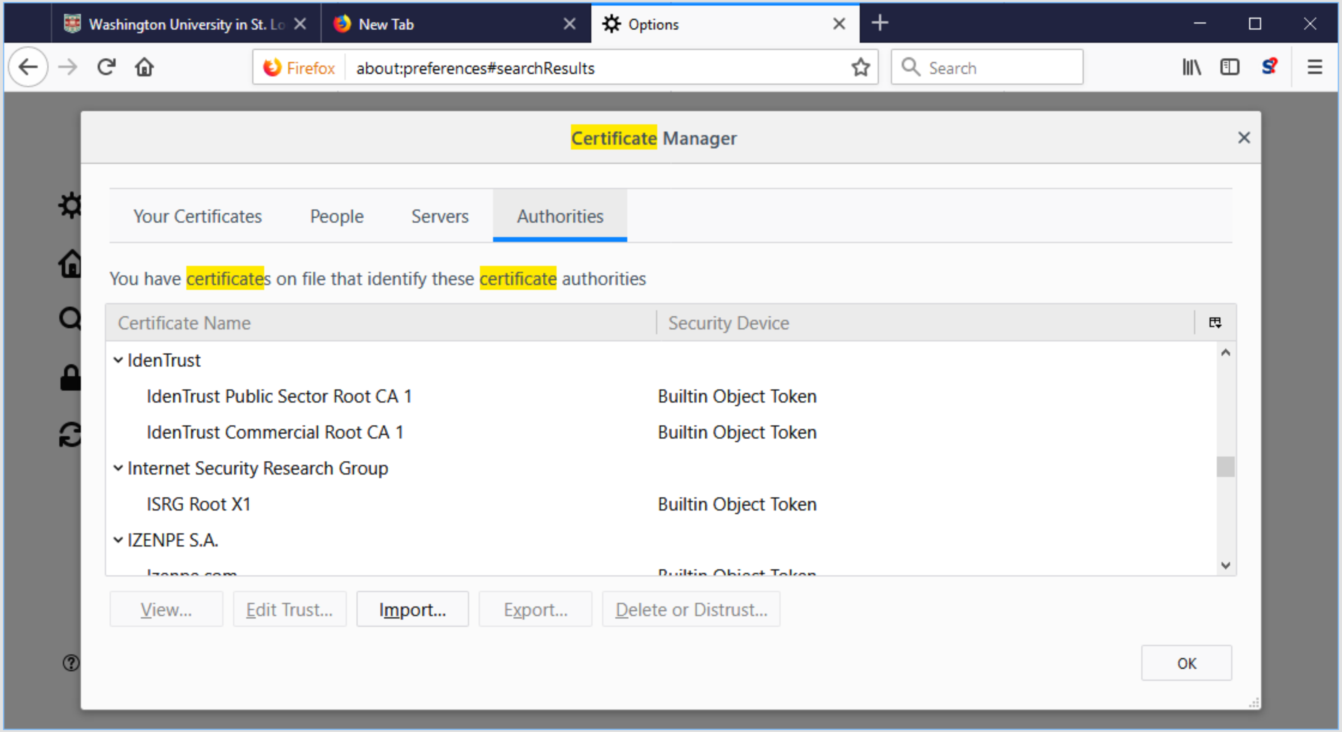


Is WUSTL secure?

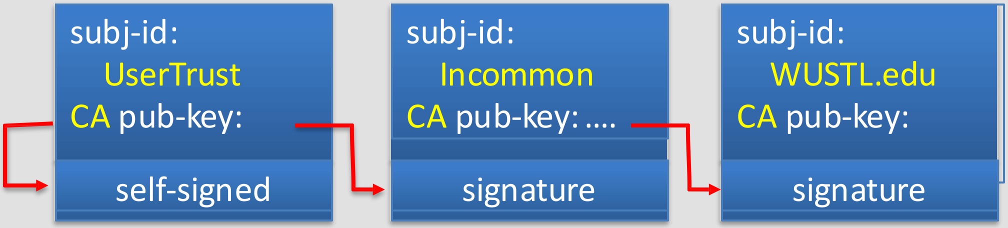


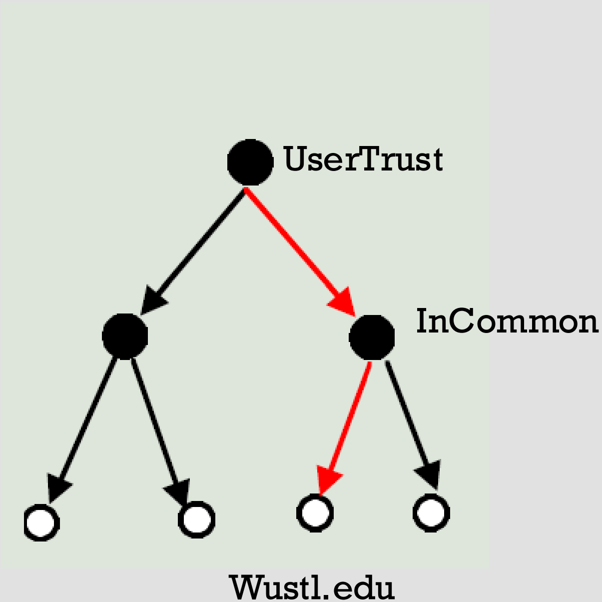

Certificate Viewer: "wustl.edu"
General
Details
This certificate has been verified for the following uses:
SSL Client Certificate
SSL Server Certificate
Issued To
Common Name (CN) wustl.edu
Organization (O) Washington University in St. Louis
Organizational Unit (OU) WUIT
Serial Number 6A:A4:83:F6:AF:57:FA:A6:0A:08:00:28:0B:AB:3A:9A
Issued By
Common Name (CN) InCommon RSA Server CA
Organization (O) Internet2
Organizational Unit (OU) InCommon
Period of Validity
Begins On Sunday, September 24, 2017
Expires On Thursday, September 24, 2020
Fingerprints
SHA-256 Fingerprint 09:46:35:7D:20:0D:7D:E0:C5:F2:2C:31:0F:EB:86:EA:E6:2F:74:B0:1F:11:96:89:3C:72:A0:0F:08:1F:7A:41
SHA1 Fingerprint 0F:5E:04:57:05:2B:93:9A:ED:99:37:1D:E7:A3:09:D0:3D:2B:3C:48
Close

What else are we trusting?



Certificate Path





- Verifier has to know the public key of root CA
- Other public keys can be discovered
- **All** CA on the path has be trusted by the verifier

Certificates: example

Important fields:

Serial Number	5814744488373890497
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, July 31, 2013 4:59:24 AM Pacific Daylight Time
Not Valid After	Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 71 6C DD E0 0A C9 76 ...
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 8A 38 FE D6 F5 E7 F6 59 ...

Equifax Secure Certificate Authority
↳ GeoTrust Global CA
↳ Google Internet Authority G2
↳ mail.google.com

mail.google.com
Issued by: Google Internet Authority G2
Expires: Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
This certificate is valid

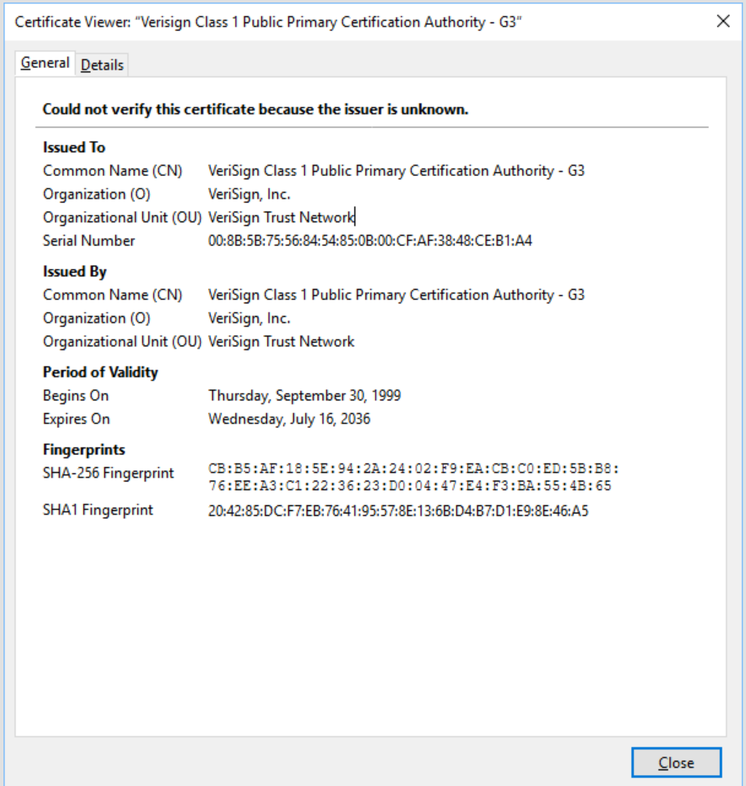
Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2

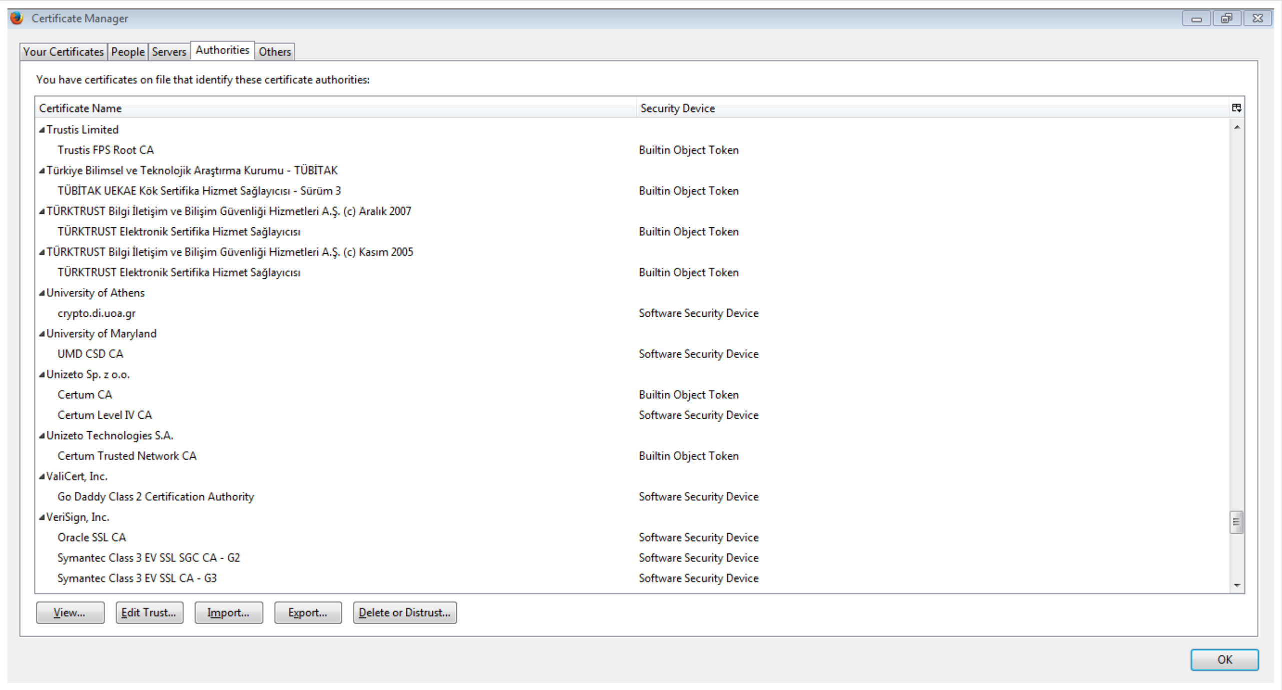
PKI in practice… Many problems

- Does not work quite as well as in theory…
  - Proliferation of root CAs
  - Revocation
  - Common Name Issue

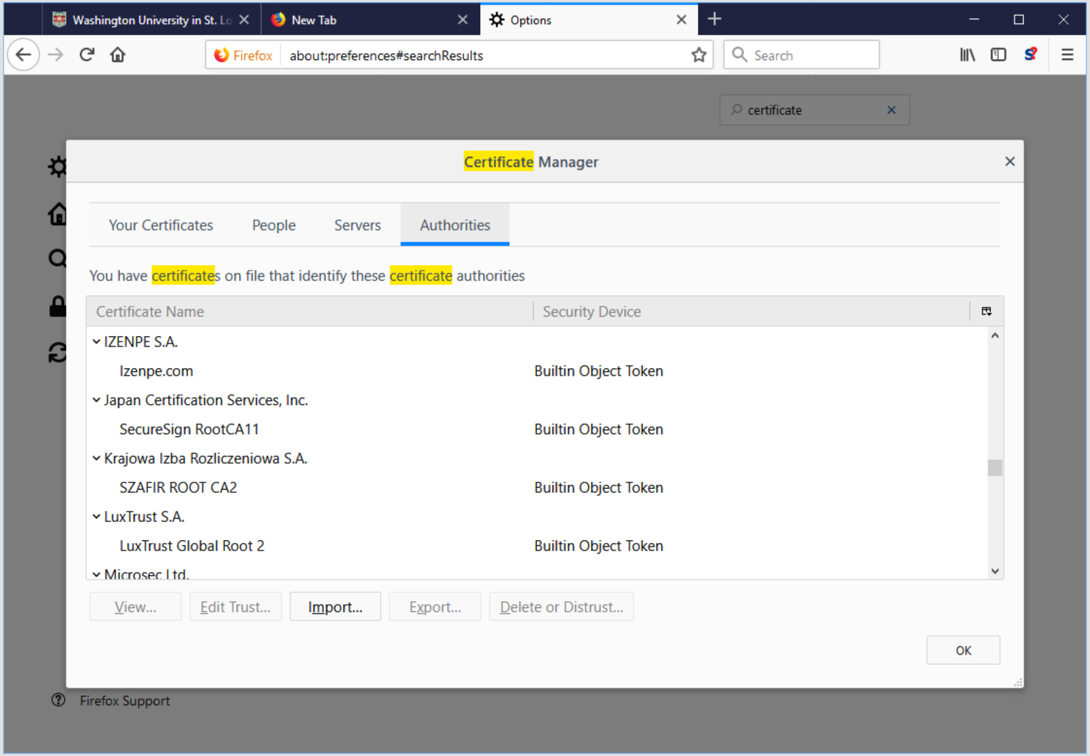
Something you trust



Do you trust them all ?



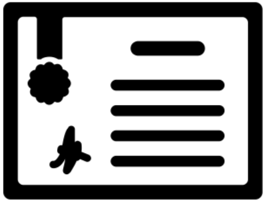
A lot of people can sign, and this is just top level



The numbers

☞ Browsers accept certificates from a large number of CAs:

- Top level CAs ≈ 60
- Intermediate CAs ≈ 1200



Revocation

Why do we need revocation?

- It could be
  - CA is compromised
  - Bob forget his private key
  - Bob’s private key is stolen
  - Bob disclose his private key

- Certificate revocation needs to occur when:
  - certificate holder key compromise/loss
  - CA key compromise
  - end of contract (e.g. certificates for employees)

General Requirement of Revocation

- Timeliness
  - Before using a certificate, must check most recent revocation status
- Efficiency
  - Computation
  - Bandwidth and storage
  - Availability
- Security

Revocation Type

- Implicit
  - Each certificate is periodically issued
  - Alice has a fresh certificate -> Alice not revoked
  - No need to distribute/publish revocation info
- Explicit
  - Only revoked certificates are periodically announced
  - Alice's certificate not listed among the revoked -> Alice not revoked
  - Need to distribute/publish revocation info

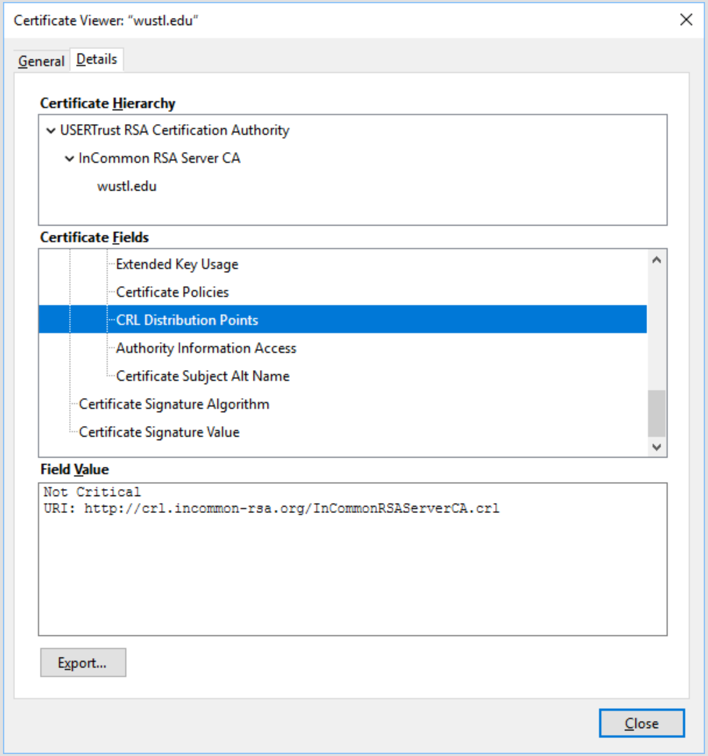
Revocation Method

- CRL - Certificate Revocation List
  - CRL-DP, indirect CRL, dynamic CRL-DP,
  - delta-CRL, windowed CRL, etc.
  - CRT and other Authenticated Data Structures

- OCSP – On-line Certificate Status Protocol

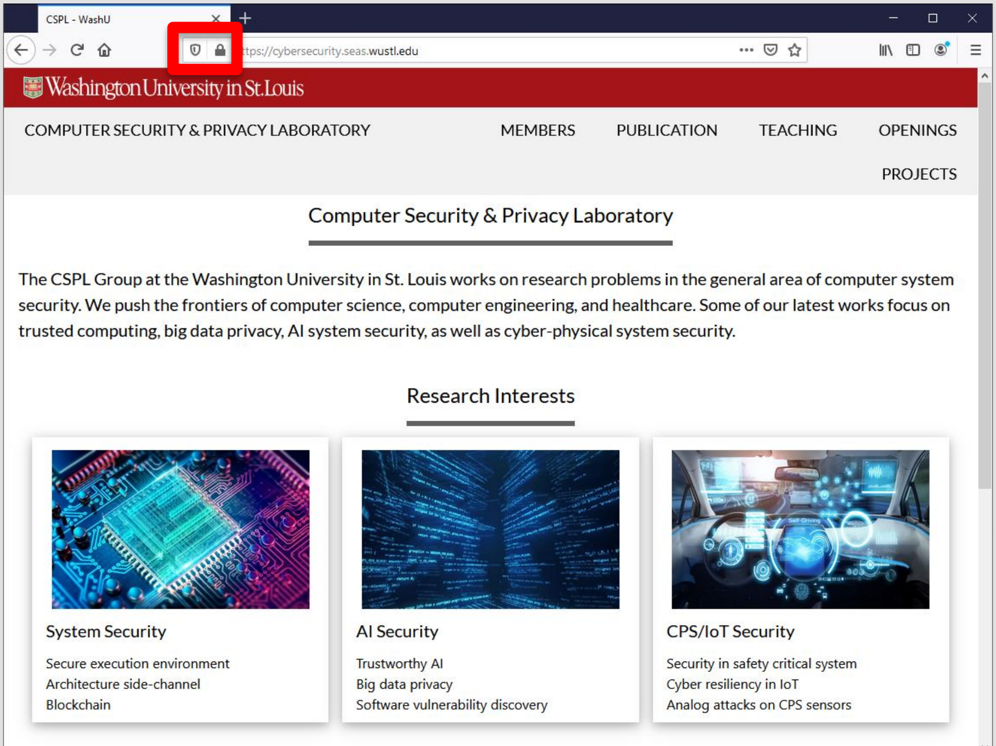
- CRS - Certificate Revocation System

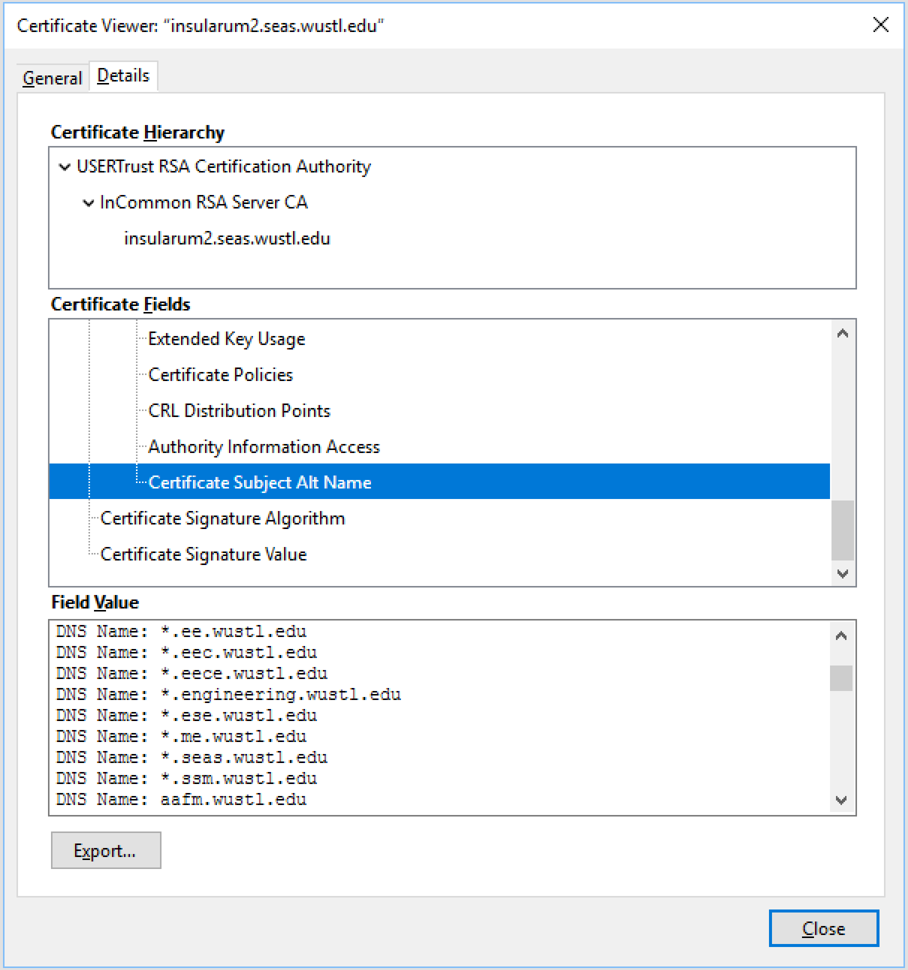
Revocation for WUSTL



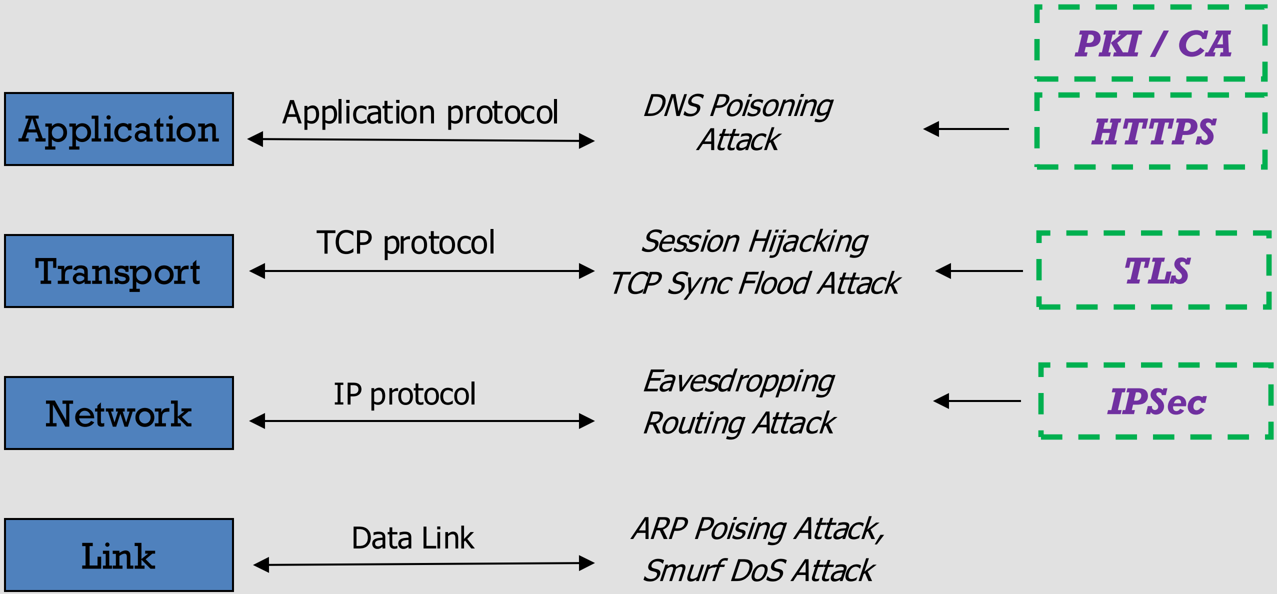
Alternative Names

My website is secure? How?



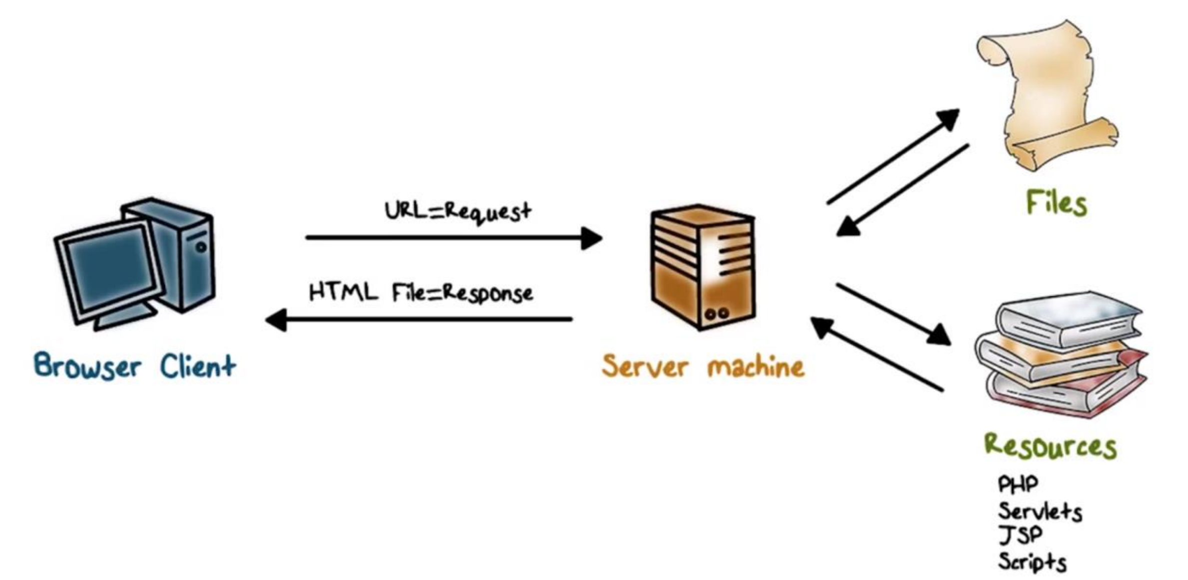

Certificate Viewer: "insularum2.seas.wustl.edu"
General
Details
Certificate Hierarchy
USERTrust RSA Certification Authority
InCommon RSA Server CA
insularum2.seas.wustl.edu
Certificate Fields
Extended Key Usage
Certificate Policies
CRL Distribution Points
Authority Information Access
Certificate Subject Alt Name
Certificate Signature Algorithm
Certificate Signature Value
Field Value
DNS Name: *.ee.wustl.edu
DNS Name: *.eec.wustl.edu
DNS Name: *.eece.wustl.edu
DNS Name: *.engineering.wustl.edu
DNS Name: *.ese.wustl.edu
DNS Name: *.me.wustl.edu
DNS Name: *.seas.wustl.edu
DNS Name: *.ssm.wustl.edu
DNS Name: aafm.wustl.edu
Export...
Close

This module - Network defense



HTTPS/TLS

HTTP



URL=Request
HTML File=Response
Browser Client
Server machine
Files
Resources
PHP
Servlets
JSP
Scripts

Threat Model

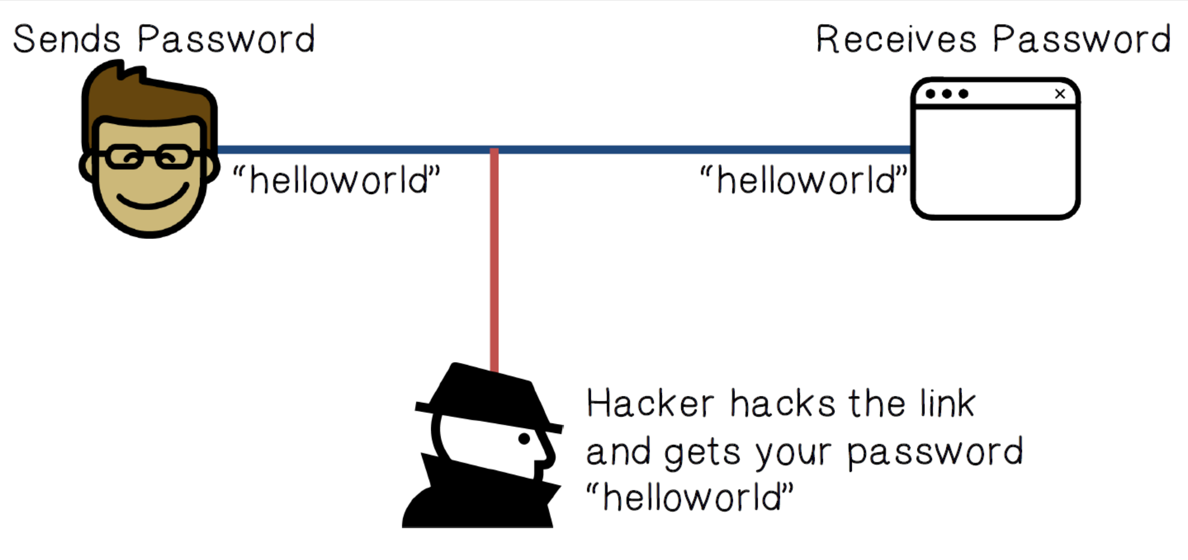
☞ Network Attacker:

- Controls network infrastructure: Routers, DNS
- Eavesdrops, injects, blocks, and modifies packets

Examples:

- Wireless network at Internet Café
- Internet access at hotels (untrusted ISP)

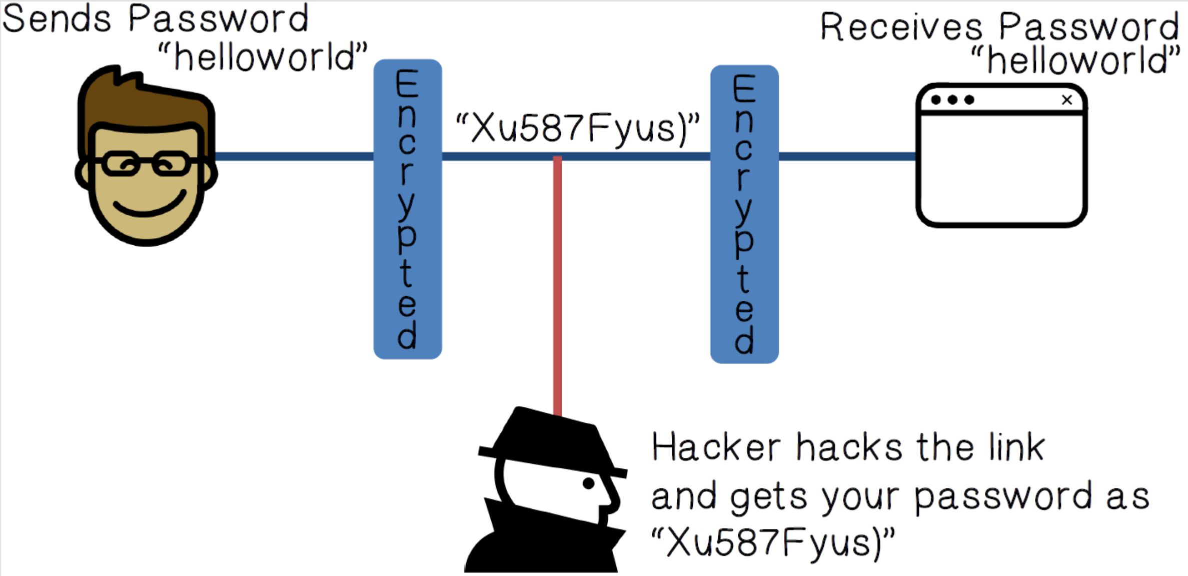
HTTP


Sends Password
Receives Password
“helloworld”
“helloworld”
Hacker hacks the link
and gets your password
“helloworld”

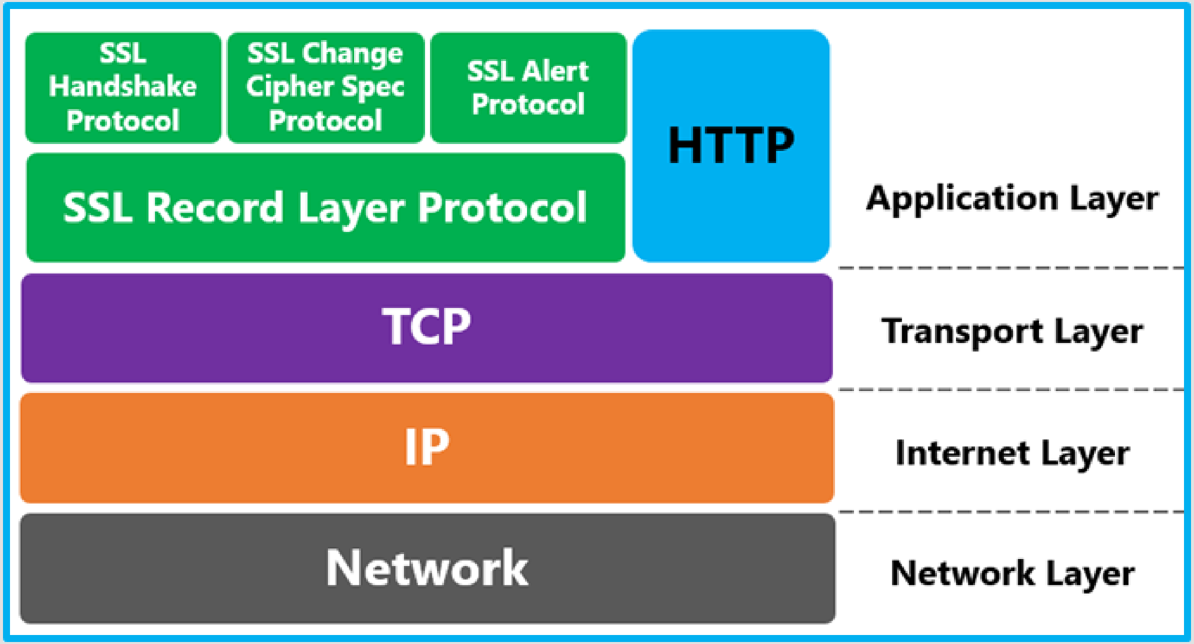
HTTPS

☞ HTTPS: Hyper Text Protocol Secure= all communication between your browser and a website is encrypted.

HTTPS

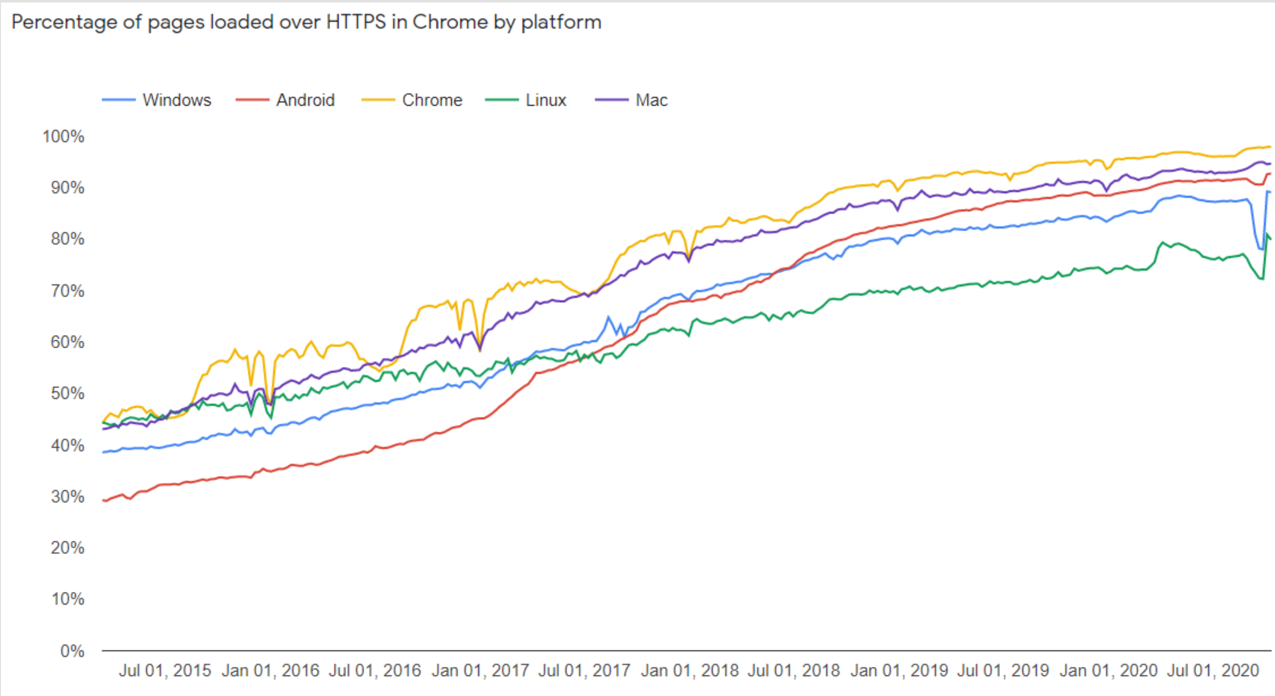

Sends Password
"helloworld"
Encrypted
"Xu587Fyus)"
Encrypted
Receives Password
"helloworld"
Hacker hacks the link
and gets your password as
"Xu587Fyus)"

Where is HTTPS

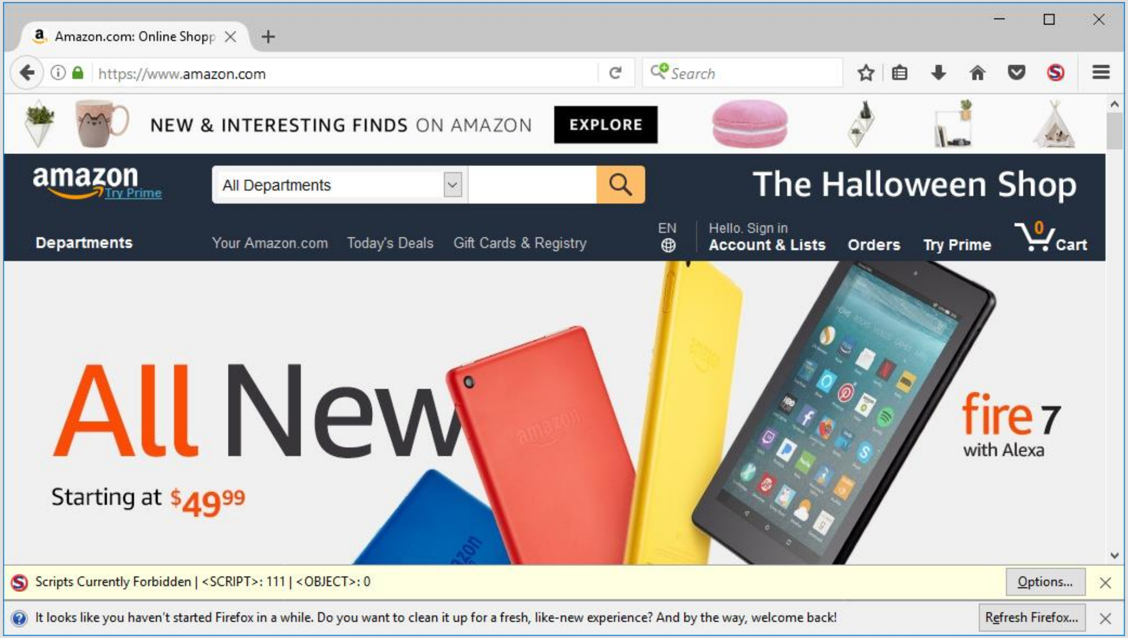


HTTPS

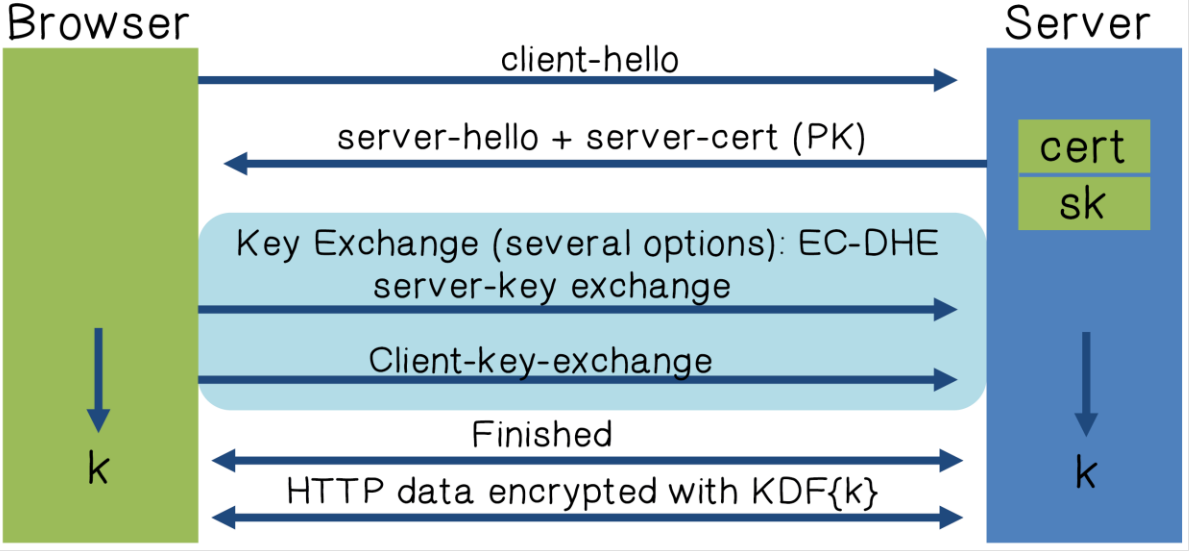
- Creates a secure channel over an insecure network
- Is an effective protection against man-in-the-middle attacks
- Can still provide security even when only one side of the communication is secure


Percentage of pages loaded over HTTPS in Chrome by platform
Windows
Android
Chrome
Linux
Mac
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Jul 01, 2015
Jan 01, 2016
Jul 01, 2016
Jan 01, 2017
Jul 01, 2017
Jan 01, 2018
Jul 01, 2018
Jan 01, 2019
Jul 01, 2019
Jan 01, 2020
Jul 01, 2020

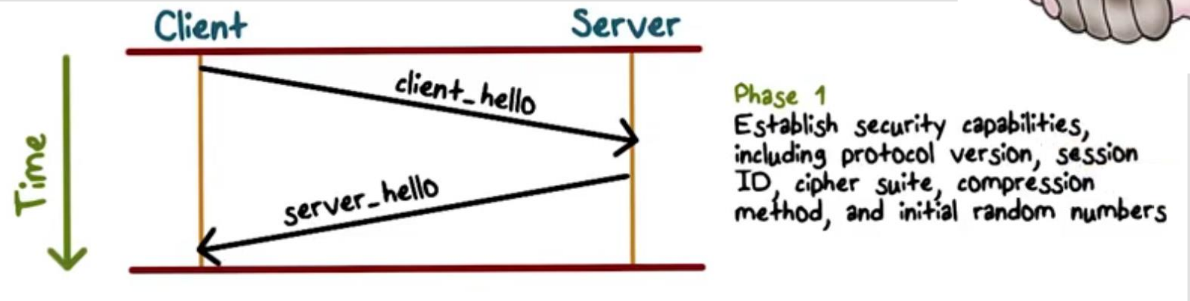
First few hundred millisecs when you go to amazon.com



TLS Summary


Browser
Server
client-hello
server-hello + server-cert (PK)
cert
sk
Key Exchange (several options): EC-DHE
server-key exchange
Client-key-exchange
k
k
Finished
HTTP data encrypted with KDF{k}

The Handshake Protocol



The Handshake Protocol

The Parameters:

- Version: the highest TLS version understood by the client
- Random: a 32-bit timestamp and 28 bytes generated by a secure random number generator
- Session ID: a variable-length session identifier
- CipherSuite: a list containing the combinations of cryptographic algorithms supported by the client
- Compression Method: a list of compression methods supported by the client

HTTPS -> Port 443

```
⊞ Internet Protocol, Src: 172.17.30.63 (172.17.30.63), Dst: 7
⊟ Transmission Control Protocol, Src Port: 50752 (50752), Dst
     Source port: 50752 (50752)
     Destination port: https (443)
     Sequence number: 1    (relative sequence number)
     [Next sequence number: 164    (relative sequence number)]
     Acknowledgement number: 1    (relative ack number)
     Header length: 20 bytes
   ⊞ Flags: 0x18 (PSH, ACK)
     Window size: 64860
```

Client Hello – Version

```
⊟ TLSv1 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 158
  ⊟ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 154
      Version: TLS 1.0 (0x0301)

   fd 5c e2 64 00 00 16 03  01 00 9e 01 00 00 9a 03   .\.d.. ........
   01 4a 2f 07 ca b9 4f b3  06 7a 06 56 7f ce c9 f7   .J/...O. .z.V....
```

Client Hello - Random

```
⊟ Random
    gmt_unix_time: Jun  9, 2009 21:09:30.000000000
    random_bytes: B94FB3067A06567FCEC9F737BD5270F7002BB0D6723E551A..
```

```
01 4a 2f 07 ca b9 4f b3  06 7a 06 56 7f ce c9 f7   .J/...O. .z.V....
37 bd 52 70 f7 00 2b b0  d6 72 3e 55 1a 0d 57 d9   7.Rp..+. .r>U..W.
82 00 00 44 c0 0a c0 14  00 88 00 87 00 39 00 38   ...D.... .....9.8
```

Why do we need this?

Client Hello – Session ID

```
   Session ID Length: 0

82 00 00 44 c0 0a c0 14  00 88 00 87 00 39 00 38   ...D.... .....9.8
```

Client Hello – Cipher Suite

```
   Cipher Suites Length: 68
⊟ Cipher Suites (34 suites)
     Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
     Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
     Cipher Suite: TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0088)

82 00 00 44 c0 0a c0 14  00 88 00 87 00 39 00 38   ...D.... .....9
c0 0f c0 05 00 84 00 35  c0 07 c0 09 c0 11 c0 13   .......5 .......
00 45 00 44 00 33 00 32  c0 0c c0 0e c0 02 c0 04   .E.D.3.2 .......
00 41 00 04 00 05 00 2f  c0 08 c0 12 00 16 00 13   .A...../ .......
c0 0d c0 03 fe ff 00 0a  01 00 00 2d 00 00 00 13   ........ ...-..
```

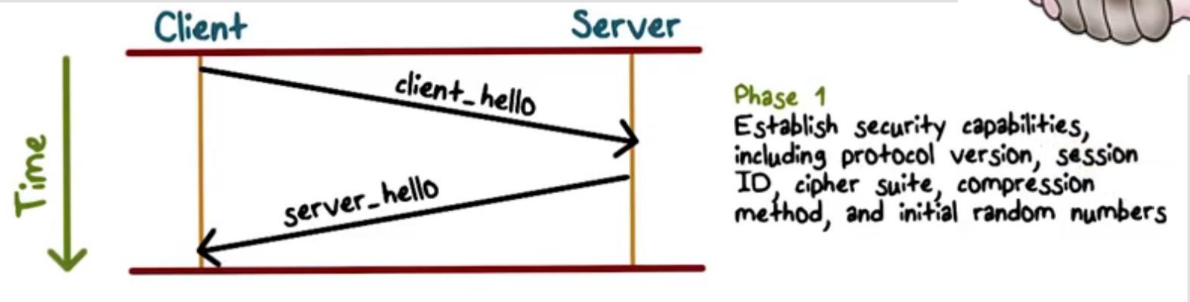
Client Hello –
Server Name Extension

```
⊞ Extension: server_name

c0 0d c0 03 fe ff 00 0a  01 00 00 2d 00 00 00 13   ........ ...-....
00 11 00 00 0e 77 77 77  2e 61 6d 61 7a 6f 6e 2e   .....www .amazon.
63 6f 6d 00 0a 00 08 00  06 00 17 00 18 00 19 00   com..... ........
```

Why do we need this?

The Handshake Protocol

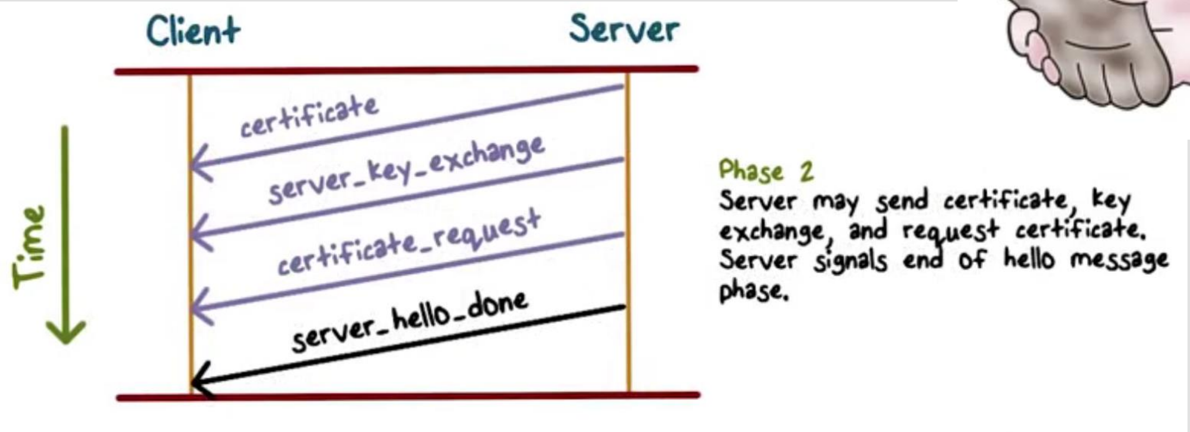


Server Hello

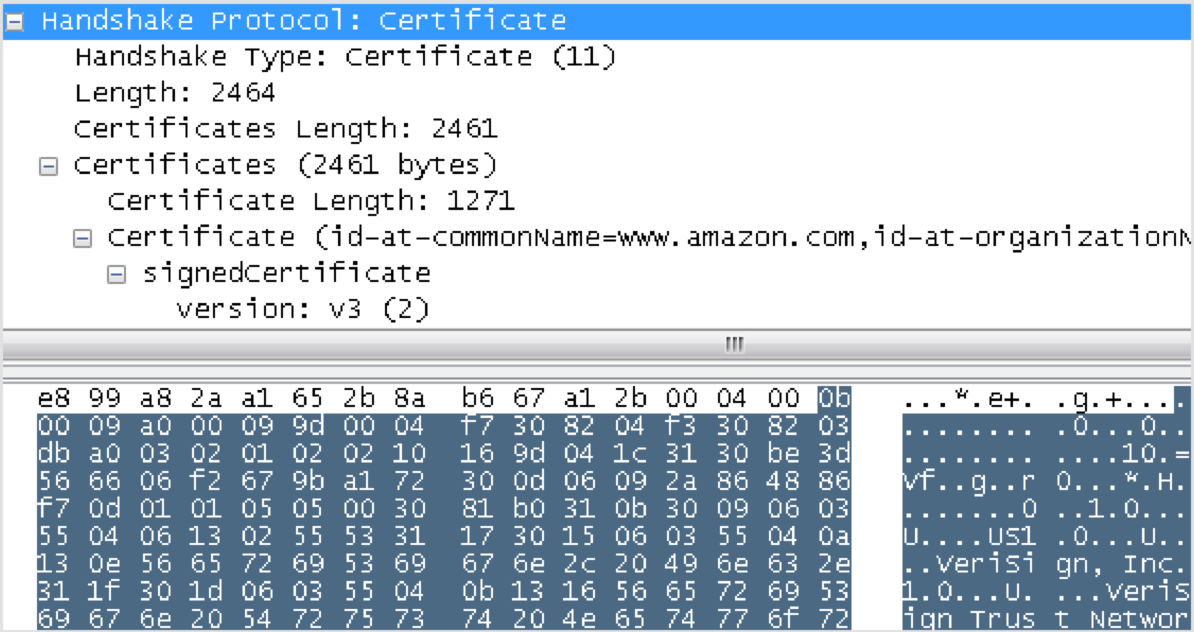
```
⊟ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 70
    Version: TLS 1.0 (0x0301)
  ⊟ Random
      gmt_unix_time: Jun  9, 2009 21:09:30.000000000
      random_bytes: 986BE7A3ACDD547D0382358 95ABA467EED6EB07CD46E5E8A..
    Session ID Length: 32
    Session ID: ACD0716799 6609E3584D15C2D43596D8AB93CFA9E899A82A...
    Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)

 16 03 01 09 f2 02 00 00  46 03 01 4a 2f 07 ca 98   ........ F..J/...
 6b e7 a3 ac dd 54 7d 03  82 35 89 5a ba 46 7e ed   k....T}. .5.Z.F~.
 6e b0 7c d4 6e 5e 8a 19  9e 0c 13 20 ac d0 71 67   n.|.n^.. ... ..qg
 99 66 09 e3 58 4d 15 c2  d4 35 96 d8 ab 93 cf a9   .f..XM.. .5......
 e8 99 a8 2a a1 65 2b 8a  b6 67 a1 2b 00 04 00 0b   ...*.e+. .g.+....
```

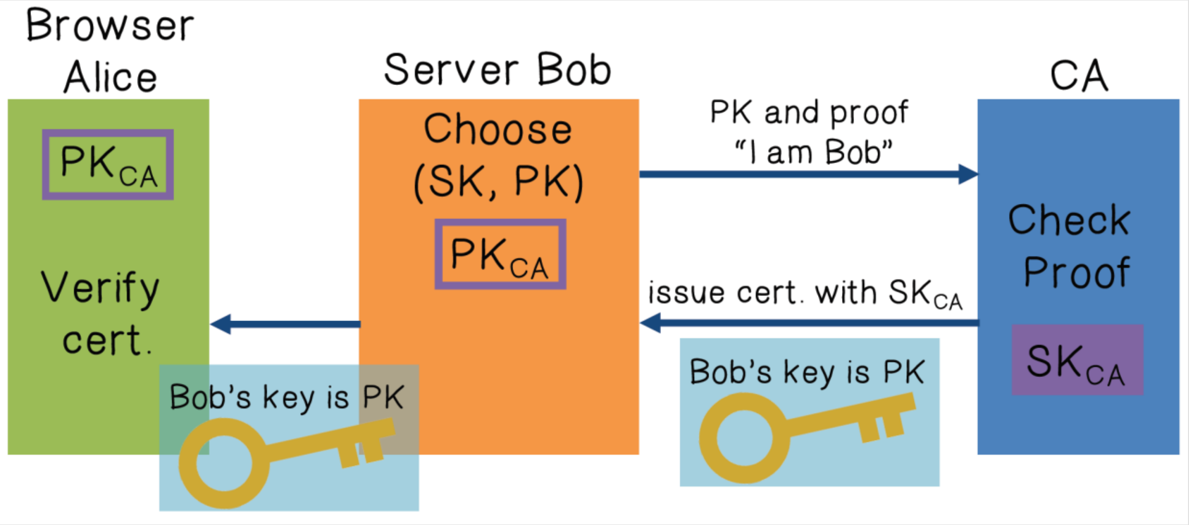
The Handshake Protocol



Certificate Message



What is a certificate ?

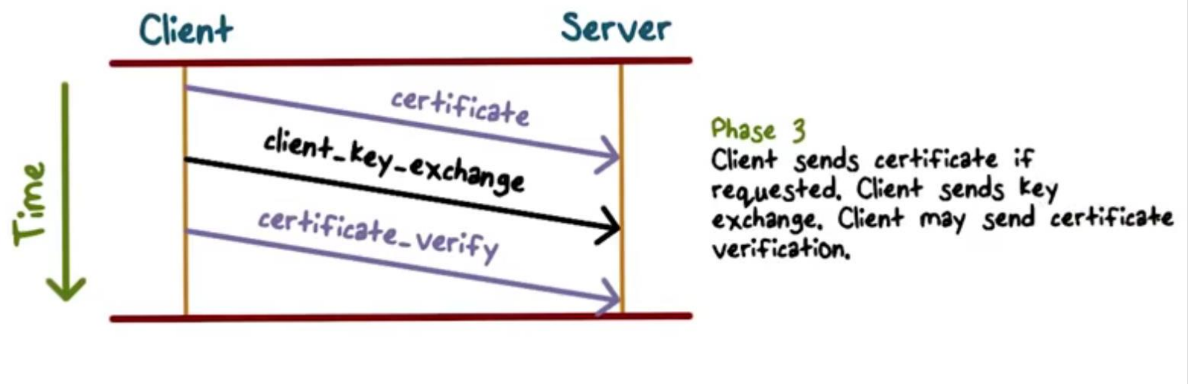


Server Hello Done

```
⊟ Handshake Protocol: Server Hello Done
    Handshake Type: Server Hello Done (14)
    Length: 0

  03 90 0c 0e 00 00 00                                      ........
```

The Handshake Protocol

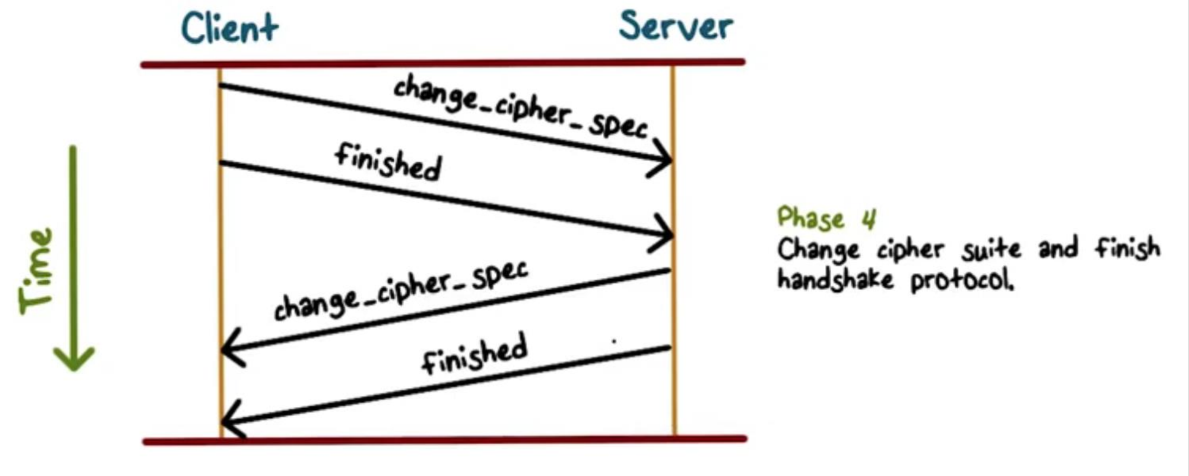


Key Exchange

```
⊟ Handshake Protocol: Client Key Exchange
    Handshake Type: Client Key Exchange (16)
    Length: 130

fd 5c e2 77 00 00 16 03  01 00 86 10 00 00 82 00   .\.w.... ........
80 8c 08 63 cc 01 6c d0  36 ef a2 28 89 e1 61 1a   ...c..l. 6..(..a.
8a 03 30 43 66 eb b0 4d  a1 cb 91 85 48 e0 24 f4   ..0Cf..M ....H.$.
88 a1 55 cd 85 03 0b 5b  d7 e6 7a 9d a2 01 90 89   ..U....[ ..z.....
ce ba 57 11 00 05 1c af  ce ef f4 2b 3f e6 30 f2   ..W..... ...+?.0.
6e 7d 55 e6 d6 c8 09 76  85 1d a2 51 7e 28 bf 83   n}U....v ...Q~(..
1e be 0b ed 60 1c 9b 00  c5 2e a6 22 38 ed a3 fb   ....`... ..."8...
7e dd 68 9a c2 ad dc 23  41 ba 30 f0 cd 95 60 48   ~.h....# A.0...`H
85 ed 44 8e 82 83 46 c3  b7 12 06 ae 48 11 4e b2   ..D...F. ....H.N.
da 14 03 01 00 01 01 16  03 01 00 20 6a 20 7a 41   ........ ... j zA
```

The Handshake Protocol

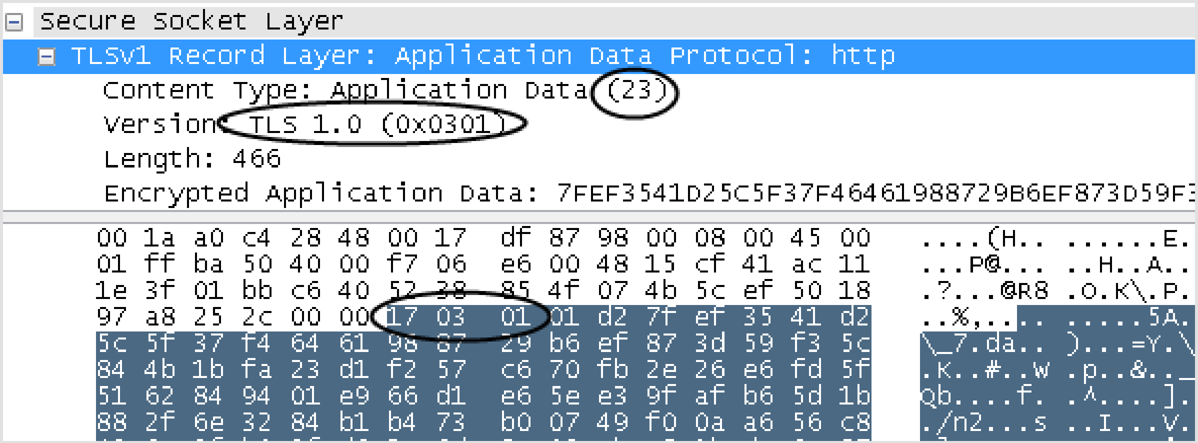


Client Change Cipher Spec

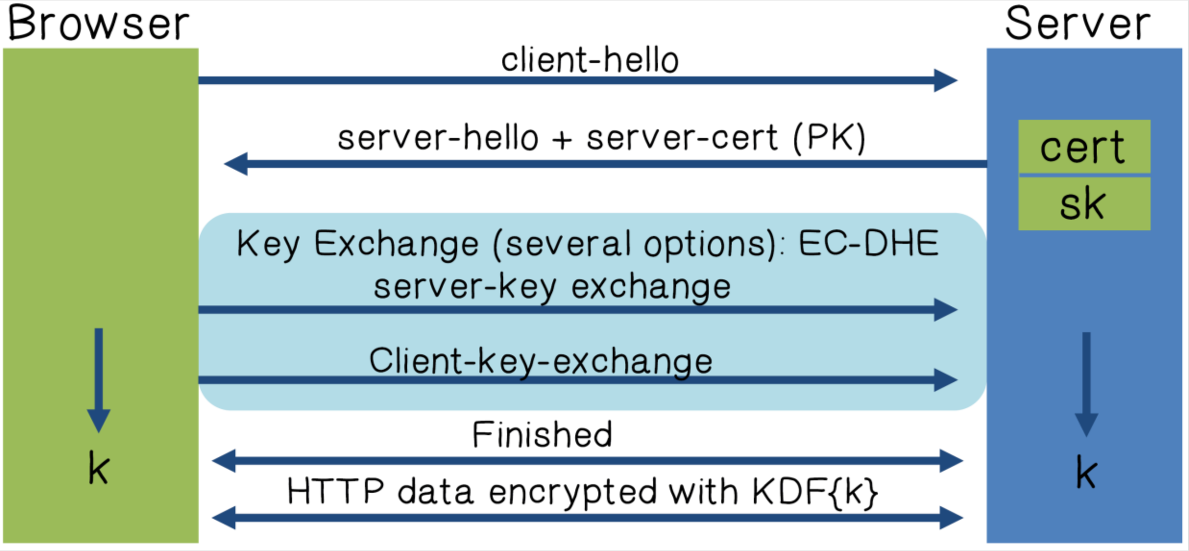
```
⊟ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.0 (0x0301)
      Length: 1
      Change Cipher Spec Message

c0   da 14 03 01 00 01 01 16   03 01 00 20 6a 20 7a 41    .......... ... j zA
```

Finally, Encrypted Traffic



TLS Summary


Browser
Server
client-hello
server-hello + server-cert (PK)
cert
sk
Key Exchange (several options): EC-DHE
server-key exchange
Client-key-exchange
Finished
HTTP data encrypted with KDF{k}
k
k

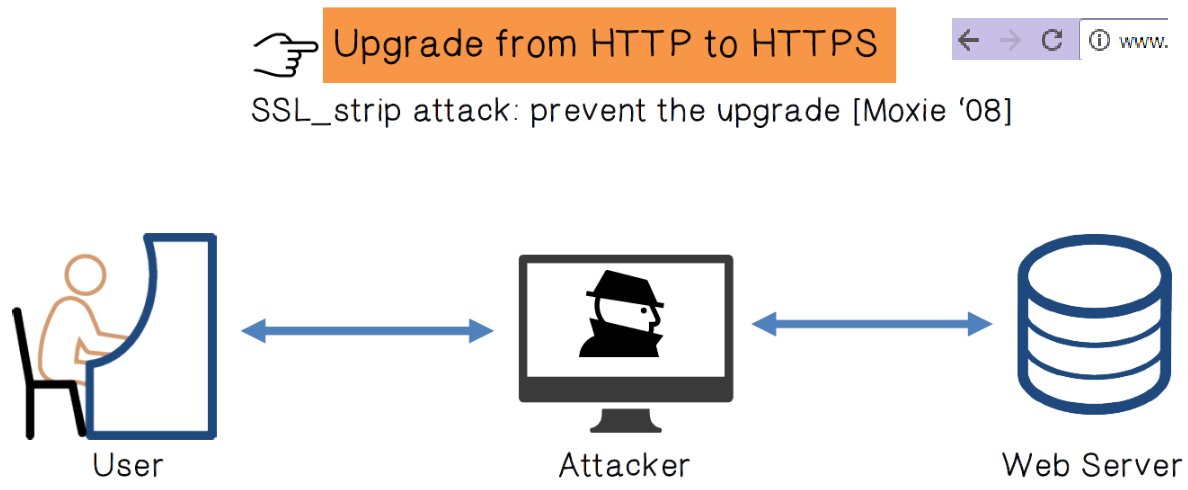
Problems with HTTPS

☞ Upgrade from HTTP to HTTPS

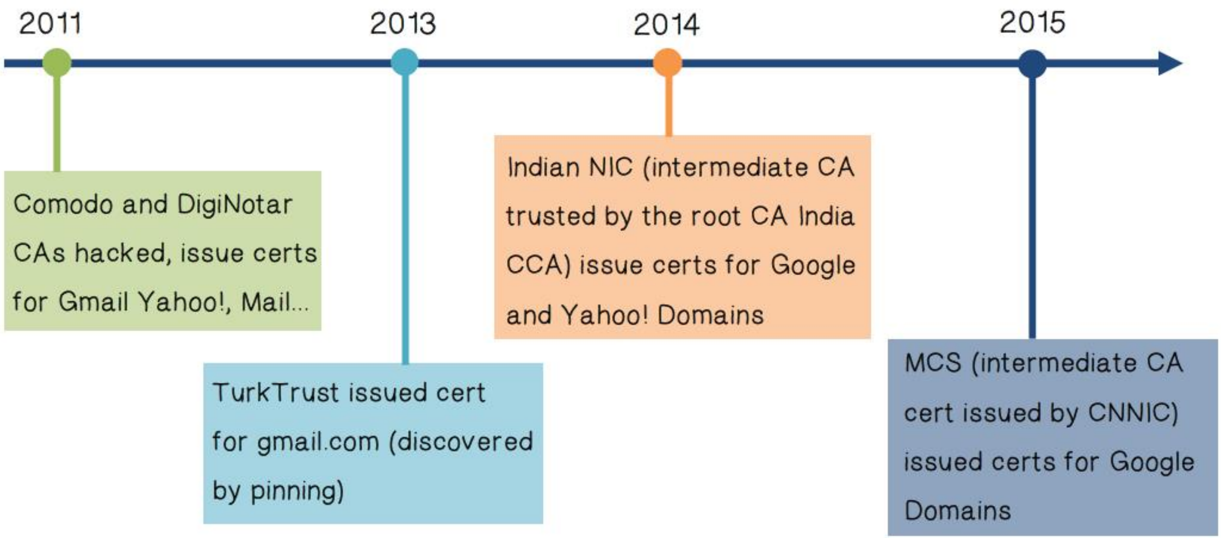
Forged certs

Mixed content: HTTP and HTTPS on the same page

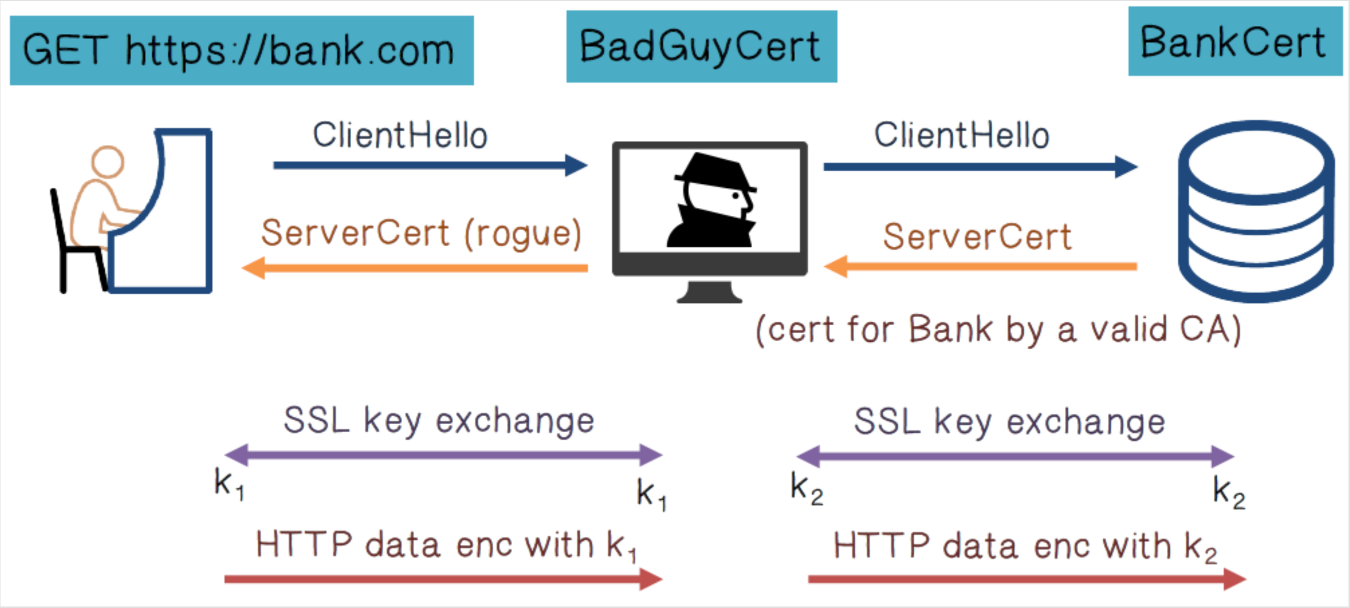
HTTPS downgrading attack



Certificates: wrong issuance



MIMT using rouge cert



Problems with HTTPS

Upgrade from HTTP to HTTPS

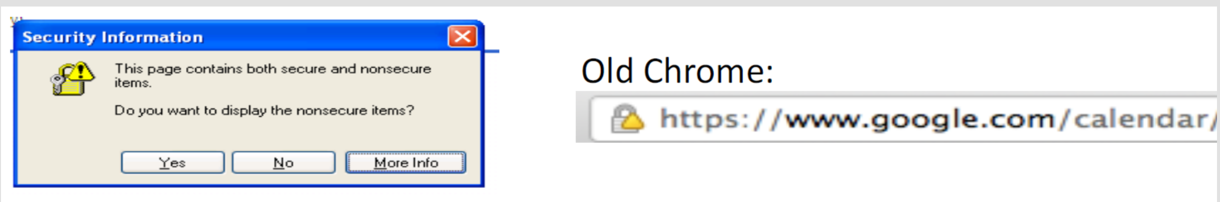
Forged certs

☞ Mixed content: HTTP and HTTPS on the same page

Mixed Contents: HTTP and HTTPS

- Page loads over HTTPS, but contains content over HTTP (e.g. <script src=“http://.../script.js>)

⇒ Active network attacker can hijack session by modifying script en-route to browser



Mostly ignored by users …

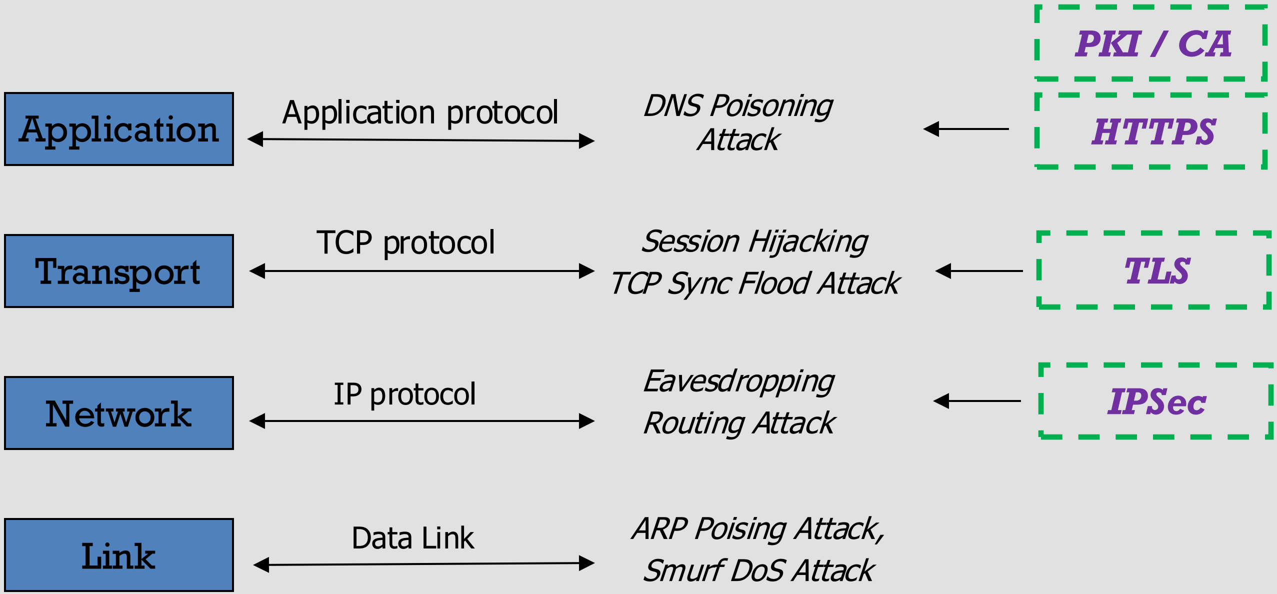
How do you attack TLS?

Padding Oracle Attack (BEAST)
Export Downgrade (FREAK)
Common Exponent (LogJam, CCS 15 Best Paper)
Side Channel Attack (Chen et al. SP 10)

Summary

- TLS is one of the most fundamental methods to secure the internet
  - TLS client hello / server hello
  - Key exchange
- HTTPS attacks
  - Downgrade attack
  - Rouge certificates
- TLS attack
  - Padding oracle (BEAST, Lucky 13)
  - Export downgrade (FREAK)
  - DH component reuse (Logjam)
  - Side channel information leakage

This module - Network defense



IPSec

IP is not Secure!

- IP protocol was designed in the late 70s to early 80s
- Part of DARPA Internet Project
- Very small network
  - All hosts are known!
  - So are the users!
  - Therefore, security was not an issue

Security Issues in IP

- Source spoofing
- Replay packets
- No data integrity or confidentiality

• DOS attacks
• Replay attacks
• MiTM attack
• Interleaving attacks
• Eavesdropping
• and more…

Fundamental Issue:

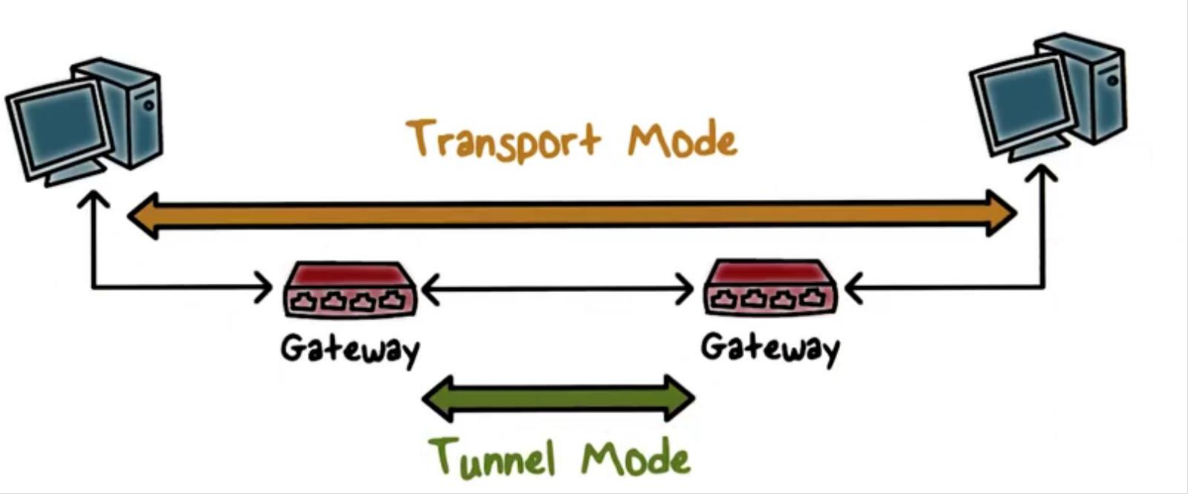
Networks are not fully secure

IPSec Architecture

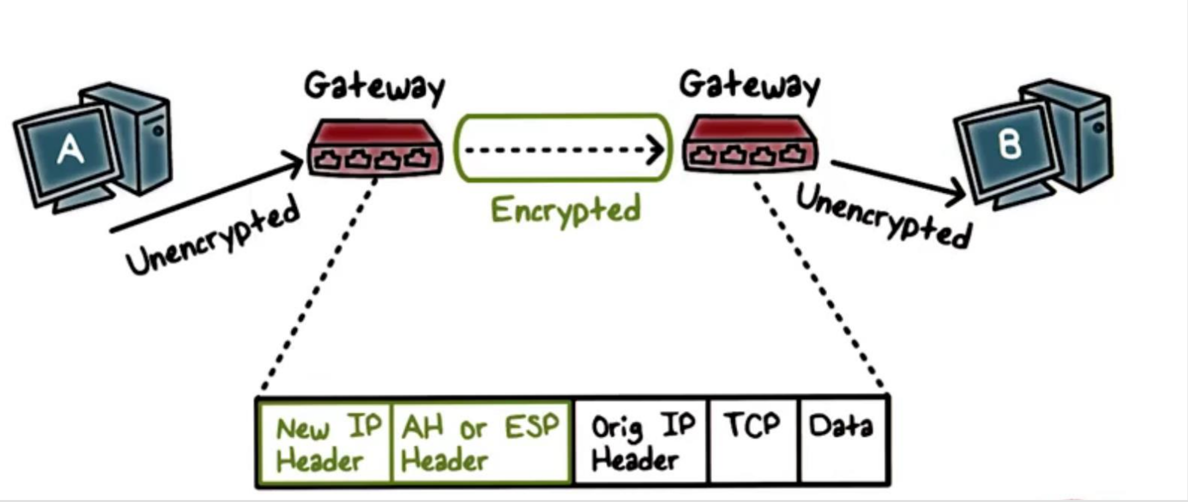
- IPSec provides security in three situations:
  - Host-to-host
  - Host-to-gateway
  - Gateway-to-gateway

- IPSec operates in two modes:
  - *Transport mode* (for end-to-end, host-to-host)
  - *Tunnel mode* (for VPN)

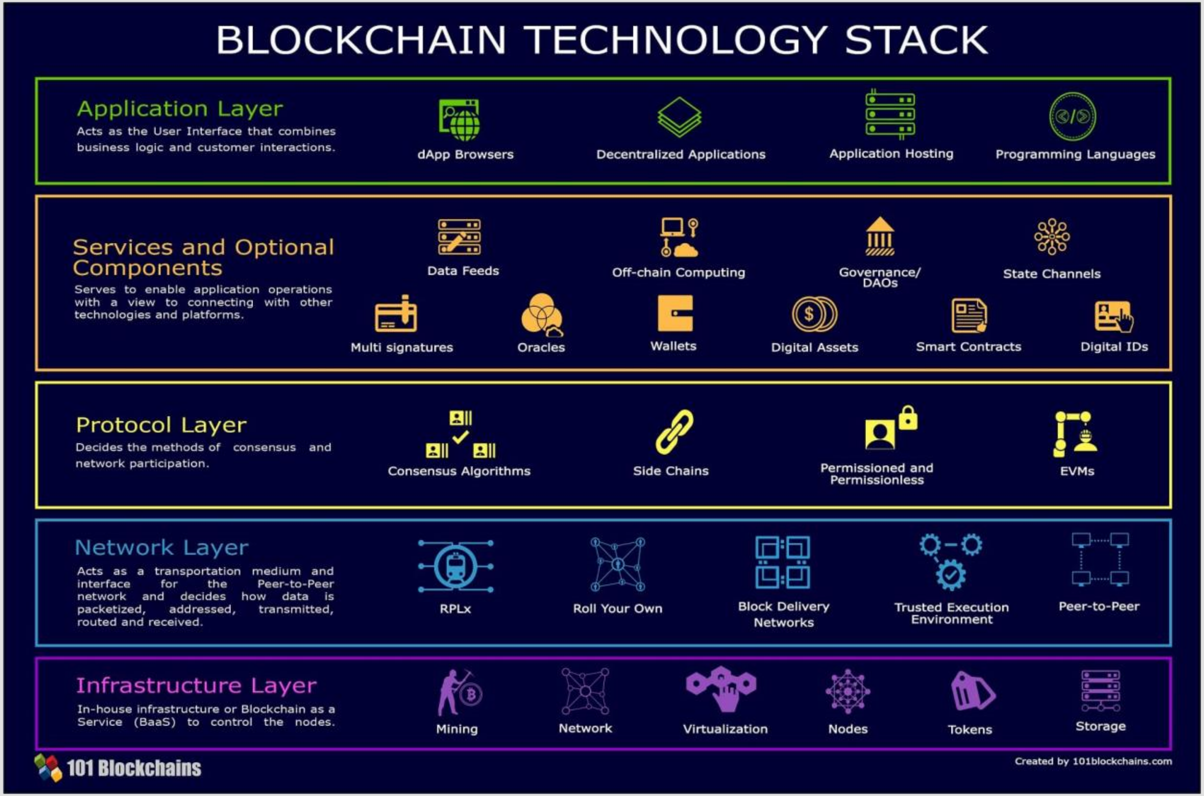
IPSec Modes - Tansport



IPSec Modes - Tunnel Mode



Blockchain


BLOCKCHAIN TECHNOLOGY STACK
Application Layer
Acts as the User Interface that combines business logic and customer interactions.
dApp Browsers
Decentralized Applications
Application Hosting
Programming Languages
Services and Optional Components
Serves to enable application operations with a view to connecting with other technologies and platforms.
Data Feeds
Off-chain Computing
Governance/ DAOs
State Channels
Multi signatures
Oracles
Wallets
Digital Assets
Smart Contracts
Digital IDs
Protocol Layer
Decides the methods of consensus and network participation.
Consensus Algorithms
Side Chains
Permissioned and Permissionless
EVMs
Network Layer
Acts as a transportation medium and interface for the Peer-to-Peer network and decides how data is packetized, addressed, transmitted, routed and received.
RPLx
Roll Your Own
Block Delivery Networks
Trusted Execution Environment
Peer-to-Peer
Infrastructure Layer
In-house infrastructure or Blockchain as a Service (BaaS) to control the nodes.
Mining
Network
Virtualization
Nodes
Tokens
Storage
101 Blockchains
Created by 101blockchains.com