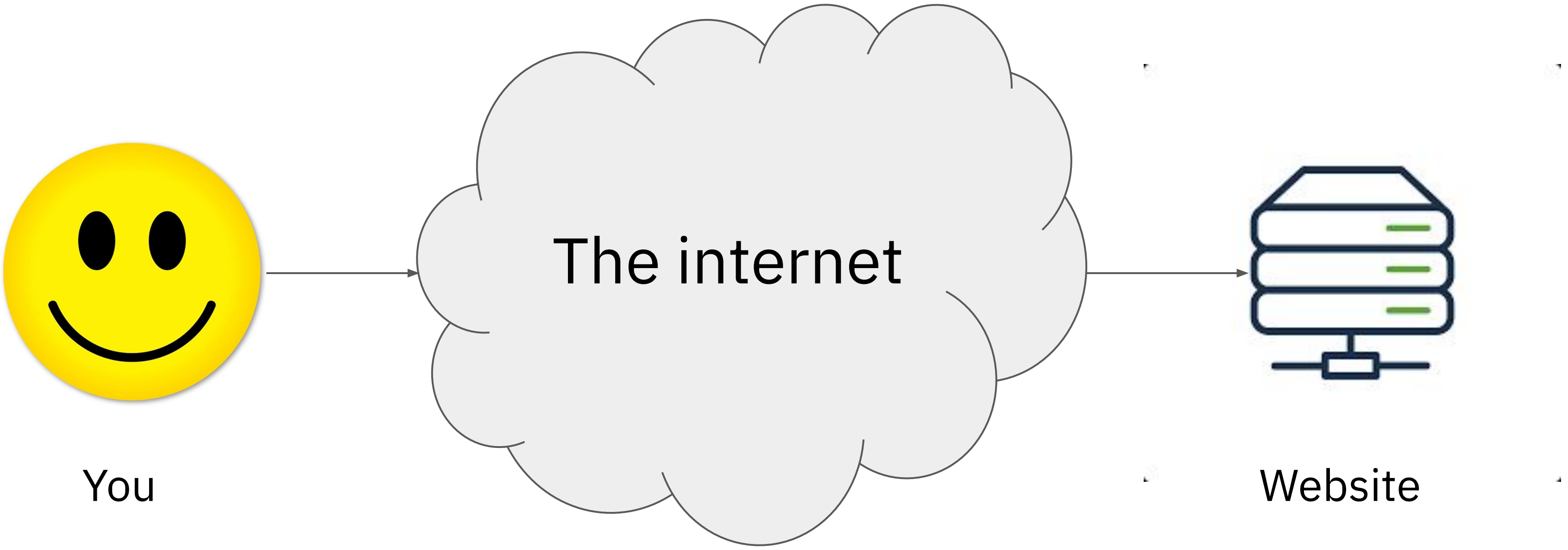
Networking Crash Course

CS 40 | January 10, 2024

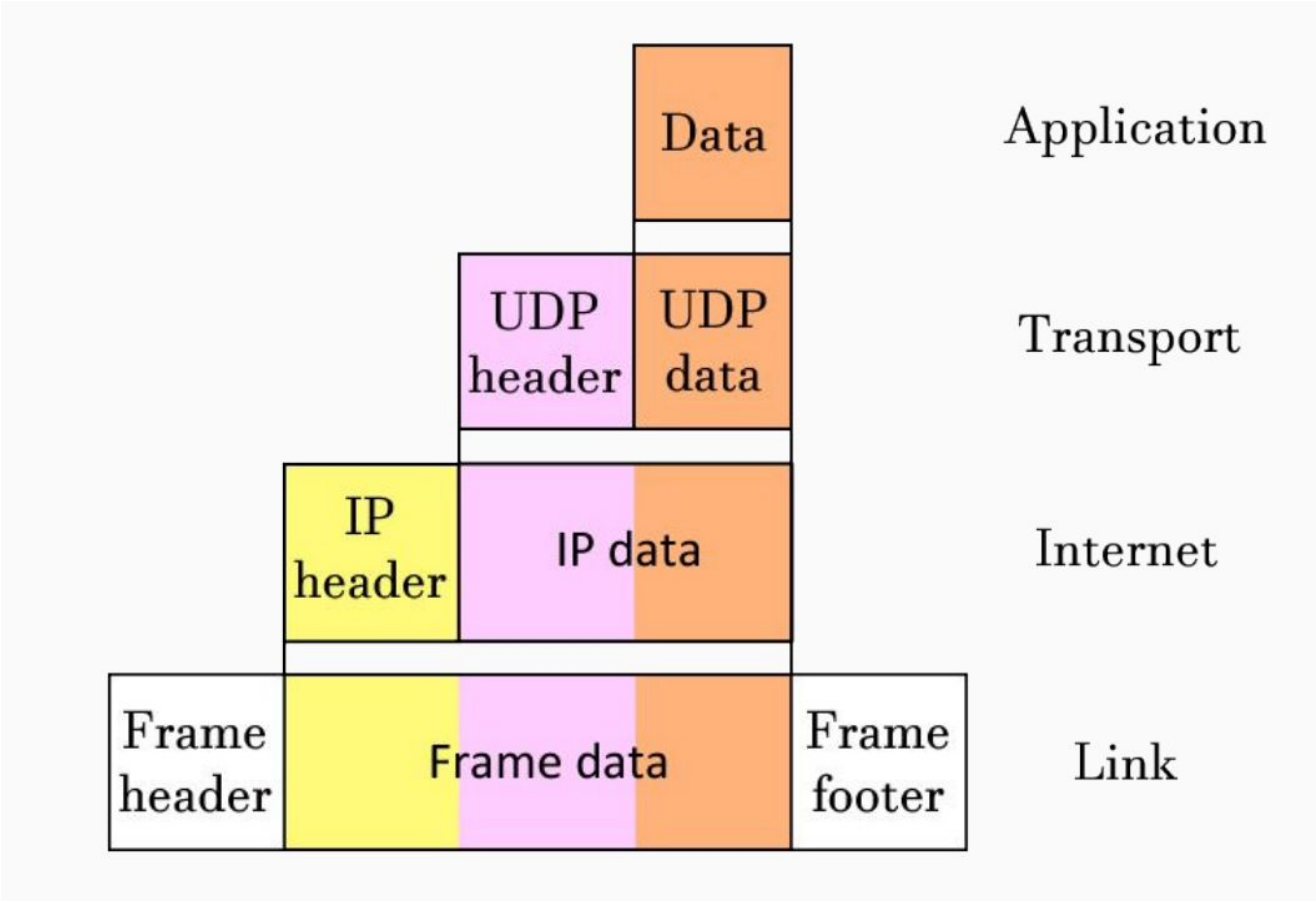
Agenda

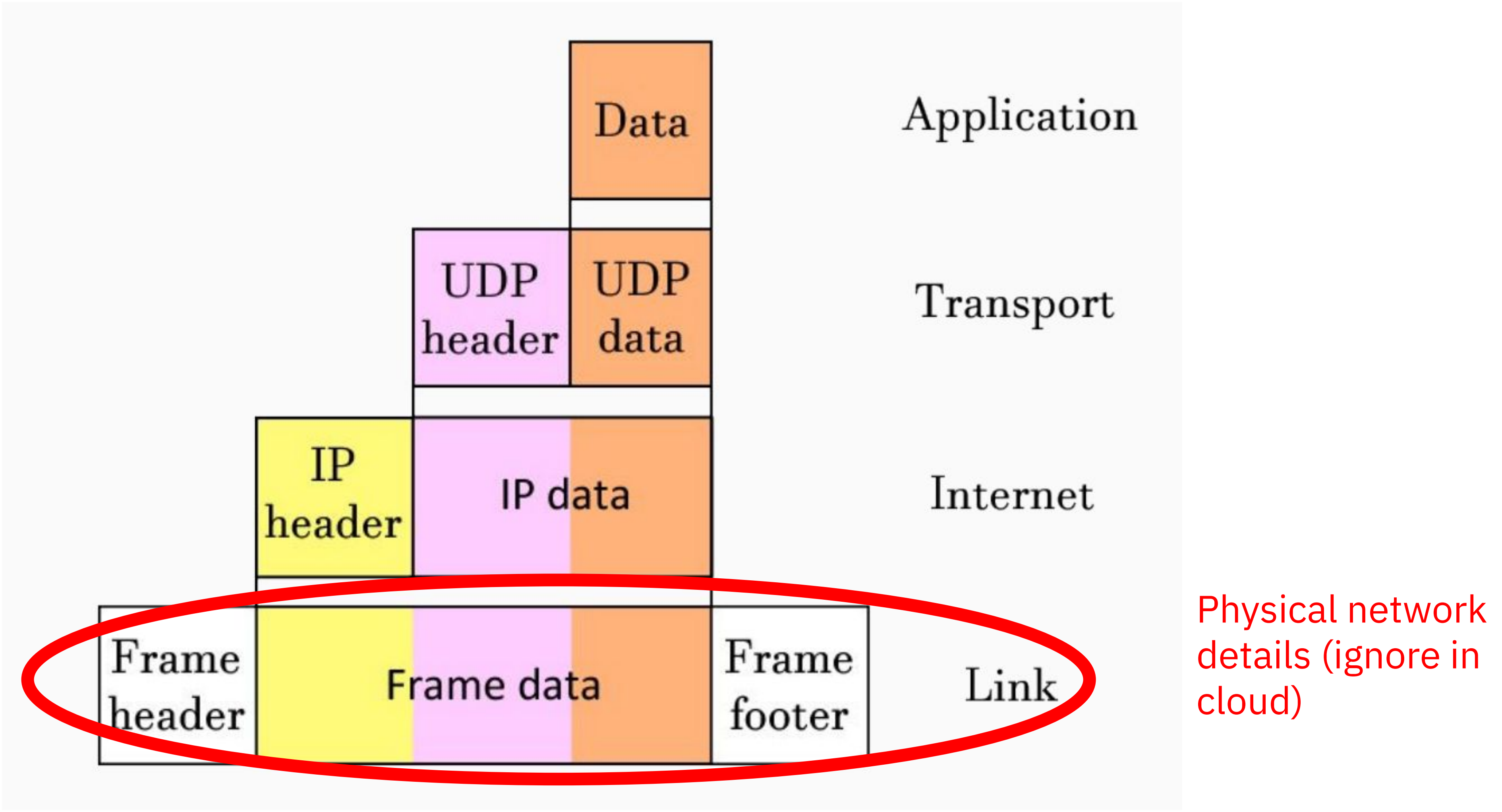
1. Brief TCP/IP Overview
2. IP Addresses and Routing
3. Networks, Subnetworks, LAN vs WAN
4. NAT
5. Firewalls
6. DNS
7. TLS

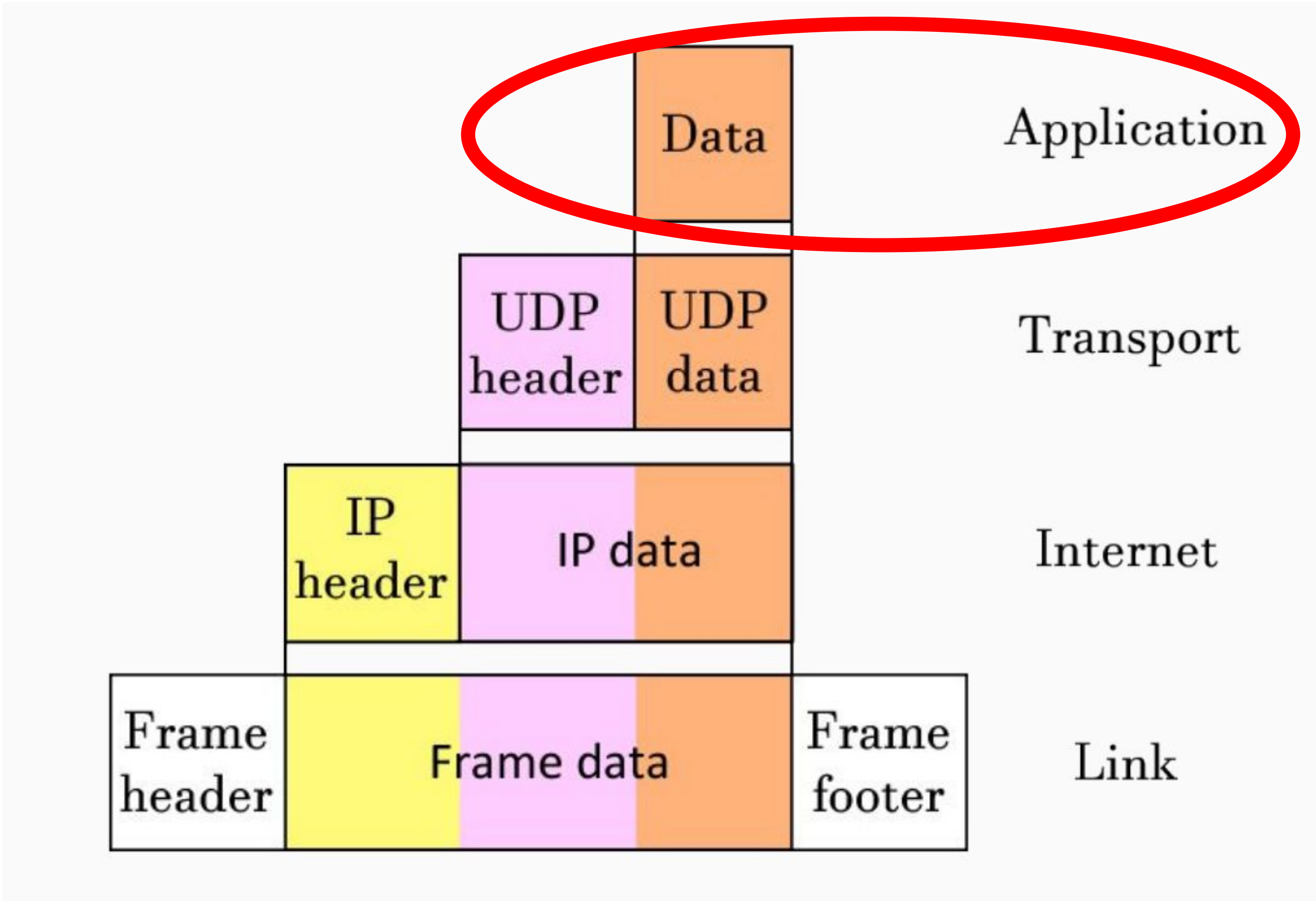
Understanding the pieces of how this works:



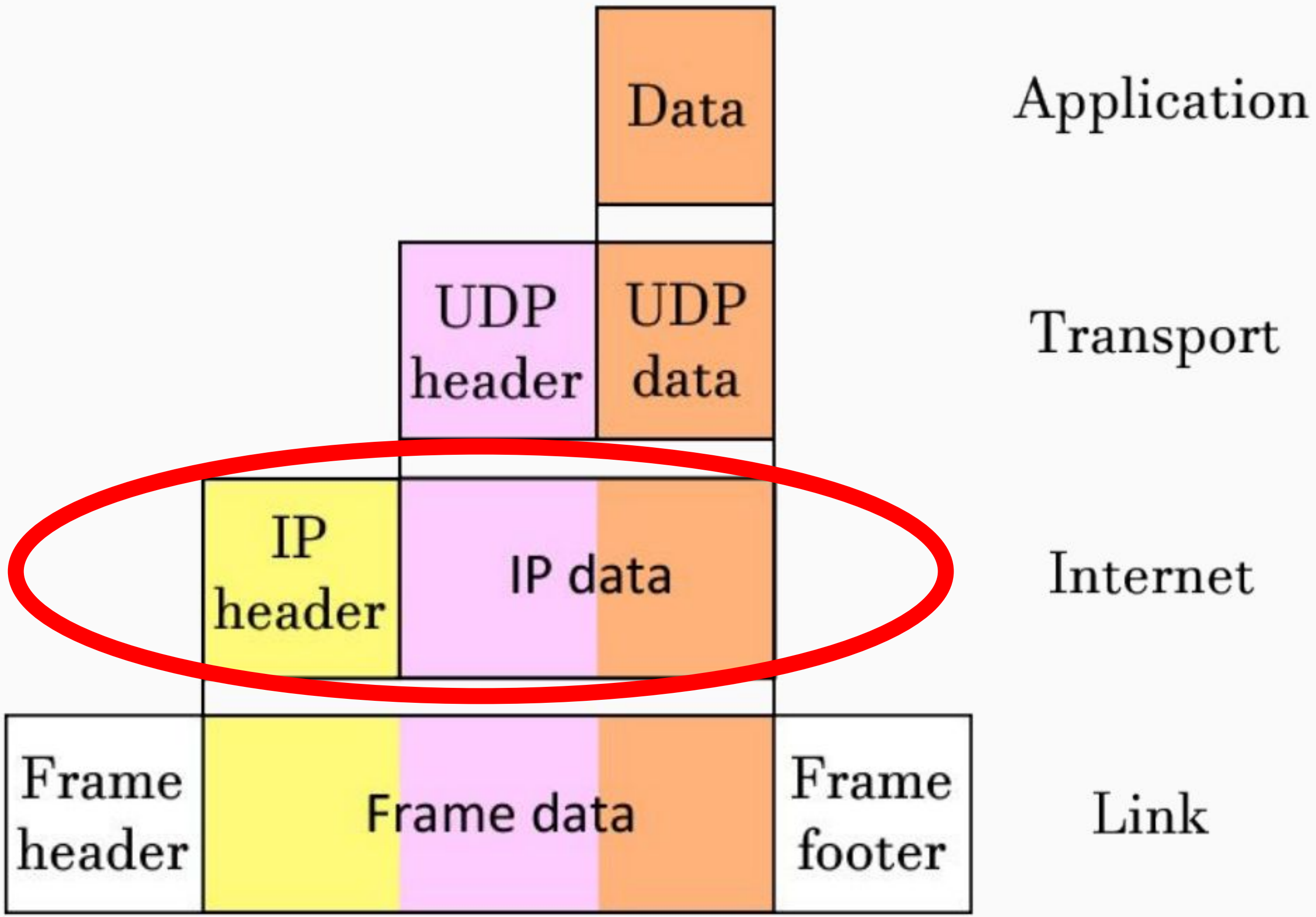
Brief TCP/IP Overview


Data
Application
UDP header
UDP data
Transport
IP header
IP data
Internet
Frame header
Frame data
Frame footer
Link


Data
Application
UDP header
UDP data
Transport
IP header
IP data
Internet
Frame header
Frame data
Frame footer
Link
Physical network details (ignore in cloud)



Depends on your product



This is our focus

IP Addresses and Routing

IP addresses

Every routable *host* has one or more IP addresses

IPv4

`192.168.171.222`

`10.34.182.62`

`127.0.0.1` (**localhost**)

`26.29.163.96`

IPv6

`6923:0db8:85a3:0000:0000:8a2e:0370:7334`

`fe80::0202:b3ff:fe1e:8329`

`2001:dead:beef:cafe::0001`

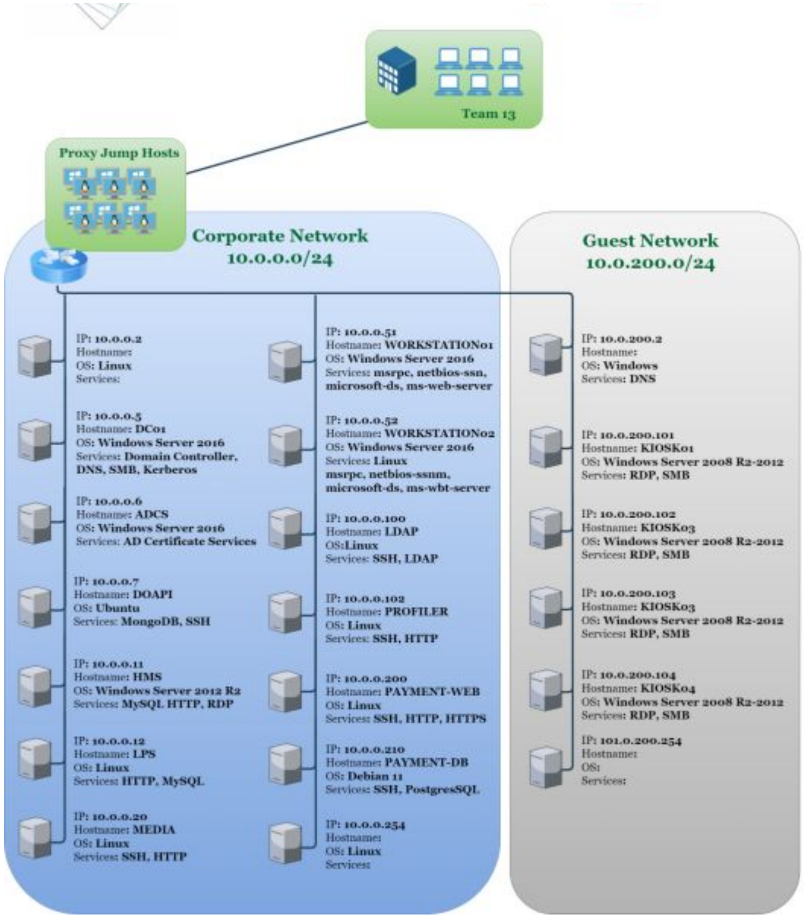
`::1` (**loopback address**)

Ports

- Every host has 65536 ports
- An application (ie, nginx) can *listen* on a port
- Important ports: 22 (SSH) 53 (DNS) 80 (HTTP) 443 (HTTPS)
- Specify both an IP and port when connecting to a host
  - ie, 192.168.0.1:53, 34.62.95.226:443, etc

TCP and UDP

- Different ways to send a packet between destinations
- TCP
  - 100% reliable, as long as the connection continues to exist
  - less bandwidth
  - higher latency
  - Used for most protocols historically
- UDP
  - packet isn't guaranteed to reach destination (but networks are usually reliable these days)
  - higher bandwidth
  - lower latency
  - DNS, modern HTTP versions


Team 13
Proxy Jump Hosts
Corporate Network
10.0.0.0/24
Guest Network
10.0.200.0/24
IP: 10.0.0.2
Hostname:
OS: Linux
Services:
IP: 10.0.0.5
Hostname: DC01
OS: Windows Server 2016
Services: Domain Controller, DNS, SMB, Kerberos
IP: 10.0.0.6
Hostname: ADCS
OS: Windows Server 2016
Services: AD Certificate Services
IP: 10.0.0.7
Hostname: DOAPI
OS: Ubuntu
Services: MongoDB, SSH
IP: 10.0.0.11
Hostname: HMS
OS: Windows Server 2012 R2
Services: MySQL HTTP, RDP
IP: 10.0.0.12
Hostname: LPS
OS: Linux
Services: HTTP, MySQL
IP: 10.0.0.20
Hostname: MEDIA
OS: Linux
Services: SSH, HTTP
IP: 10.0.0.51
Hostname: WORKSTATION01
OS: Windows Server 2016
Services: msrpc, netbios-ssn, microsoft-ds, ms-web-server
IP: 10.0.0.52
Hostname: WORKSTATION02
OS: Windows Server 2016
Services: Linux
msrpc, netbios-ssnm, microsoft-ds, ms-wbt-server
IP: 10.0.0.100
Hostname: LDAP
OS:Linux
Services: SSH, LDAP
IP: 10.0.0.102
Hostname: PROFILER
OS: Linux
Services: SSH, HTTP
IP: 10.0.0.200
Hostname: PAYMENT-WEB
OS: Linux
Services: SSH, HTTP, HTTPS
IP: 10.0.0.210
Hostname: PAYMENT-DB
OS: Debian 11
Services: SSH, PostgresSQL
IP: 10.0.0.254
Hostname:
OS: Linux
Services:
IP: 10.0.200.2
Hostname:
OS: Windows
Services: DNS
IP: 10.0.200.101
Hostname: KIOSK01
OS: Windows Server 2008 R2-2012
Services: RDP, SMB
IP: 10.0.200.102
Hostname: KIOSK03
OS: Windows Server 2008 R2-2012
Services: RDP, SMB
IP: 10.0.200.103
Hostname: KIOSK03
OS: Windows Server 2008 R2-2012
Services: RDP, SMB
IP: 10.0.200.104
Hostname: KIOSK04
OS: Windows Server 2008 R2-2012
Services: RDP, SMB
IP: 101.0.200.254
Hostname:
OS:
Services:

Routing

- Motivating Issue: How to get from one host to another, potentially over different networks across the group

- Define *routes*: paths to various destinations
    - *default route*: where to send a packet that doesn't match any other route
    - for most hosts, there are is only two routes, one route to localhost and the default route to the router

- Routers forward packets to the next router until it reaches the destination
    - General path: your network → your ISP → their ISP → their network

Example Path to Destination

```
traceroute to google.com (142.250.189.206), 30 hops max, 60 byte packets
 1  _gateway (10.138.2.251)  0.752 ms  0.717 ms  0.704 ms
 2  10.137.0.7 (10.137.0.7)  0.887 ms  0.874 ms  0.863 ms
 3  10.5.64.1 (10.5.64.1)  3.628 ms *  3.603 ms
 4  10.5.1.3 (10.5.1.3)  3.591 ms  3.580 ms 10.5.1.2 (10.5.1.2)  3.569 ms
 5  10.5.0.1 (10.5.0.1)  2.480 ms  2.469 ms  2.459 ms
 6  csee-rtf-rtr-vl3877.SUNet (171.64.77.3)  3.519 ms csee-west-rtr-vl3877.SUNet (171.64.77.2)  5.567
 7  dc-sf-rtr-vl12.SUNet (171.66.0.207)  8.656 ms  8.621 ms  8.608 ms
 8  dc-sfo-agg4--stanford-100g.cenic.net (137.164.23.178)  5.454 ms  3.634 ms  3.603 ms
 9  dc-svl-agg8--sfo-agg4-100gbe.cenic.net (137.164.11.92)  6.954 ms  6.943 ms  6.932 ms
10  dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80)  6.908 ms  6.891 ms  6.881 ms
11  142.250.175.184 (142.250.175.184)  6.870 ms 74.125.50.18 (74.125.50.18)  6.873 ms 142.250.175.184
12  * * *
13  * 142.251.228.228 (142.251.228.228)  10.815 ms 142.250.238.166 (142.250.238.166)  7.063 ms
14  192.178.46.196 (192.178.46.196)  5.117 ms  5.102 ms 142.251.224.175 (142.251.224.175)  5.091 ms
15  sfo03s25-in-f14.1e100.net (142.250.189.206)  5.071 ms  5.059 ms  5.044 ms
```

Example Path to Destination

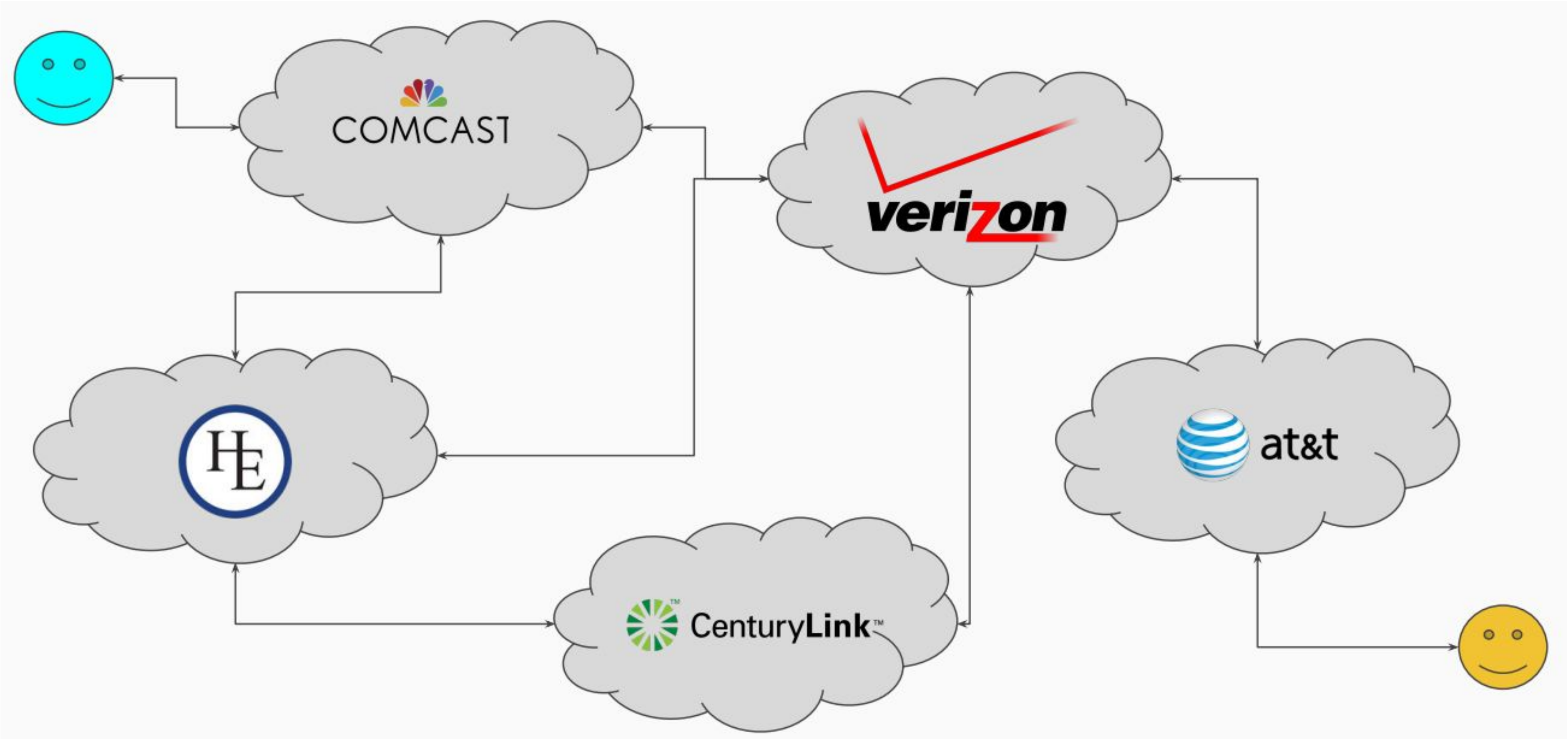
```
traceroute to google.com (142.250.189.206), 30 hops max, 60 byte packets
 1  _gateway (10.138.2.251)  0.752 ms  0.717 ms  0.704 ms
 2  10.137.0.7 (10.137.0.7)  0.887 ms  0.874 ms  0.863 ms
 3  10.5.64.1 (10.5.64.1)  3.628 ms *  3.603 ms
 4  10.5.1.3 (10.5.1.3)  3.591 ms  3.580 ms 10.5.1.2 (10.5.1.2)  3.569 ms
 5  10.5.0.1 (10.5.0.1)  2.480 ms  2.469 ms  2.459 ms
 6  csee-rtf-rtr-vl3877.SUNet (171.64.77.3)  3.519 ms csee-west-rtr-vl3877.SUNet (171.64.77.2)  5.567
 7  dc-sf-rtr-vl12.SUNet (171.66.0.207)  8.656 ms  8.621 ms  8.608 ms
 8  dc-sfo-agg4--stanford-100g.cenic.net (137.164.23.178)  5.454 ms  3.634 ms  3.603 ms
 9  [illegible]
10  dc-svl-agg10--svl-agg8-300g.cenic.net (137.164.11.80)  6.908 ms  6.891 ms  6.881 ms
11  142.250.175.184 (142.250.175.184)  6.870 ms 74.125.50.18 (74.125.50.18)  6.873 ms 142.250.175.184
12  * * *
13  * 142.251.228.228 (142.251.228.228)  10.815 ms 142.250.238.166 (142.250.238.166)  7.063 ms
14  192.178.46.196 (192.178.46.196)  5.117 ms  5.102 ms 142.251.224.175 (142.251.224.175)  5.091 ms
15  sfo03s25-in-f14.1e100.net (142.250.189.206)  5.071 ms  5.059 ms  5.044 ms
```

Stanford internal network and ISP

Google's ISP

Google

Demo: Routing


COMCAST
verizon
at&t
CenturyLink

Subnets

- **Definition**: A *subnet* is a range of IP addresses behind a router
  - Hosts can communicate with each other through an internal router

- Subnets help efficiently manage groups of hosts
  - Network level firewalls
  - Unified DNS
  - Manage IP address assignments

Subnet Notation

Prefix length

172.16.0.0/12

Prefix

Special Subnets

- **`127.0.0.1`**: localhost
- Three internal subnets
  - **`10.0.0.0/8`**
  - **`192.168.0.0/16`**
  - **`172.16.0.0/12`**

Aside: IPv4 shortage

- 2^{32} = ~4 billion possible IP addresses

- Large chunks already owned by ISPs and other corporations, or reserved

- Implication: *scarce resource → you have to pay for static IPv4 addresses*
  - *Elastic IP is AWS's method of doing this*

- Solution: IPv6, except it's a disaster
  - for CS40 purposes, we're only going to talk about IPv4

NAT

Demo: NAT

NAT

- Core Idea: **Abstract away the details of the internal network**
  - Fewer IPv4 addresses used
  - Security
  - Flexibility

- Replace the source IP of every packet with the IP of the router
  - Every host behind the router shares the same public IP
  - Endpoint can't determine which host on a network made what request (without other information)

Firewalls

Firewalls

- Firewalls can regulate network access to:
  - IPs
  - Subnets
  - Ports

- Two directions: *inbound* and *outbound*
  - Generally, inbound rules are very strict, but more difficult to write strict outbound rules
  - Applications usually need a way of updating themselves, or communicating with external resources

- Two modes of operation: default allow or default deny

Two types of firewalls

Network level

- Protects the *entire* subnet
- Often implemented in the router
- **AWS term**: *VPC ACL* (next lecture)

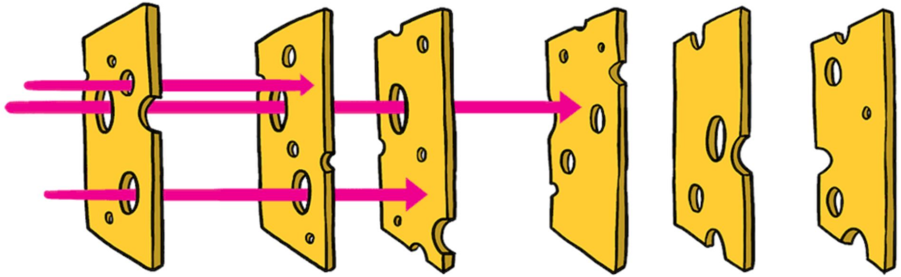
Host level

- Protects a single host
- Depending on implementation, can be disabled by an administrator on the host
- **AWS term**: *Security Group*

Demo: Firewalls

Why two layers of firewalls?

- **Defense in depth**: multiple overlapping layers of security are better than one
  - "Swiss Cheese Model"
  - One layer of compromise should not be sufficient to defeat security



- **Principle of least privilege**: only give {machines, people, everything} access to what they need to function
  - prevents accidental errors as well as improves security

DNS

It's not DNS

There's no way it's DNS

It was DNS

D(omain) N(ame) S(ystem)

- Motivating issue:
  - IP protocol and routing works using *IP addresses*
  - Humans don't work using IP addresses

- Solution: Provide a way to *resolve* human readable names (ie, `google.com`) to an IP address
  - Abstracts away more hosting details
  - Flexibility
  - In practice: query a DNS server to resolve a domain name to an IP, then connect to the IP

DNS Architecture

- Hierarchical tree of DNS servers
  - **Root DNS servers** maintained by ICANN
- Router (usually) hosts a **local DNS server**, provides internal network DNS
- Router specifies an **authoritative DNS server** for machines outside of the domain
  - Performs a *reverse DNS lookup* for all queries it cannot resolve internally

Example DNS Query Path

You

↓

Domain Controller

↓

Local Network DNS

↓

ISP DNS

↓

Google DNS

Important DNS Servers

- Cloudflare: **`1.1.1.1`**
- Google: **`8.8.8.8, 8.8.4.4`**
- Others: **`9.9.9.9`**, various ISP DNS servers
  - e.g. Stanford: **`171.67.64.53, 171.64.69.53`**

DNS Record Types

- **`A`** (alias): simplest record type, maps domain name to IPv4 address
  - e.g. **`a1.codyho.infracourse.cloud`** → **`34.212.146.53`**
  - **`AAAA`**: Same thing for IPv6 domains

- **`CNAME`** (canonical name): used to create aliases, mapping domain names to domain names
  - e.g. **`provisiondns.infracourse.cloud`** → **`infracourse-dns-provisioner.pages.dev`**

- **`NS`** (nameserver): used to designate an authoritative nameserver
  - Output: another DNS server to query the domain name against
  - e.g. **`codyho.infracourse.cloud`** → **`ns-1573.awsdns-04.co.uk`**

https://toolbox.googleapps.com/apps/dig

DNS Replication

- DNS servers communicate with each other to **continuously update** their records
    - e.g. when someone buys a new domain name

- Replication introduces **synchronization issues** in DNS
    - Records take time to propagate
    - Different DNS servers may have outdated entries

Route53

Aside: DNS Implications

- **Privacy**: Without additional protections, anyone on the network path between you and the DNS server can see all the websites you visit
  - e.g. ISPs often mine this data (or even host their own DNS) to sell, often for ad targeting
  - Solution: **DNS over HTTPS**: encrypted DNS queries and responses

- **Censorship**: A small number of DNS providers can effectively blacklist a website

Transport Layer Security (TLS)

Privacy error

Not secure | https://expired.badssl.com

Your connection is not private

Attackers might be trying to steal your information from **expired.badssl.com** (for example, passwords, messages, or credit cards). Learn more

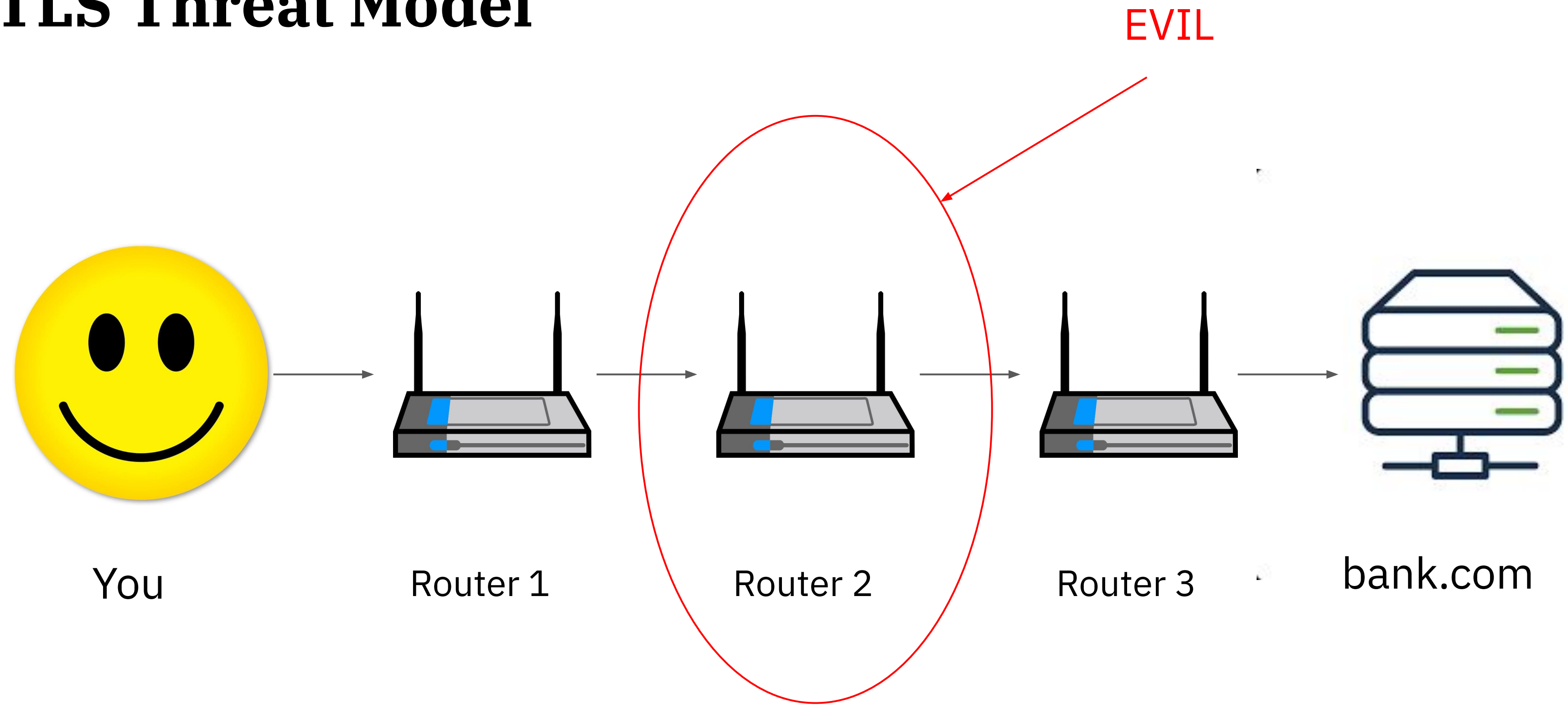
NET::ERR_CERT_DATE_INVALID

To get Chrome's highest level of security, turn on enhanced protection

Advanced

Back to safety

TLS Threat Model



Solution: Asymmetric Cryptography

- Idea: Server sends you a *public key* that can **only be used to encrypt data**, such that only the corresponding *private key* can decrypt the data
  - Because the public key can't decrypt the data, an attacker has no way of viewing the plaintext data

- This helps create a **secure channel** that can be used to communicate
  - Server public and private key used to **agree on a shared key** used to encrypt all traffic

- Issue: How do you verify the public key is **legitimately owned** by the website you are trying to connect to?
  - Solution: have some way of **choosing which keys to trust**
    - Designate a number of **trusted providers** (*certificate authorities*) which can verify other keys
    - Only accept a key if someone you already trust has verified it's legitimate

T(ransport) L(ayer) S(ecurity)

- Idea: have a hierarchical tree structure of trusted providers
  - The root of this tree is stored on your hard drive and is controlled by your OS and browser
  - Any certificate with a chain of trust ending at one of these certificates is trusted
  - Certificates are given out by *certificate authorities*

- Based on X.509 certificates, contain:
  - Public key
  - *Signature*
    - from a trusted CA, not self signed
  - Domain name(s) the certificate is valid for
  - Time a certificate is valid
  - Cryptographic information

https://letsencrypt.org/certs/isrgrootx1.txt

What happens when you navigate to google.com?

1. Resolve the domain name `google.com`
2. Connect to `google.com`, first through a NAT, then a network of public routers
3. Receive a certificate from `google.com`
4. Verify the certificate from `google.com`
5. Establish the connection

Further exploration

- **CS 144** Introduction to Computer Networking
- **CS 155** Computer and Network Security
- **CS 255** Introduction to Cryptography
- **CS 249I** The Modern Internet
- **CS 349D** Cloud Computing Technology
    - Taught by our faculty sponsor Prof. Kozyrakis