# ECE568 Lecture 07: Network Security 02

Wei Huang
Department of Electrical and Computer Engineering
University of Toronto

# Outline

- ~~**Spoofing**~~
  - ~~DNS cache poisoning~~
  - BGP spoofing
  - TCP connection Spoofing
  - ARP Spoofing
- **Denial of Service**
  - Syn-flooding
  - TCP Reset
- **Defenses**
  - TLS
  - IPSec
  - Firewalls

# What the Bad Guys (and girls) Could Do?

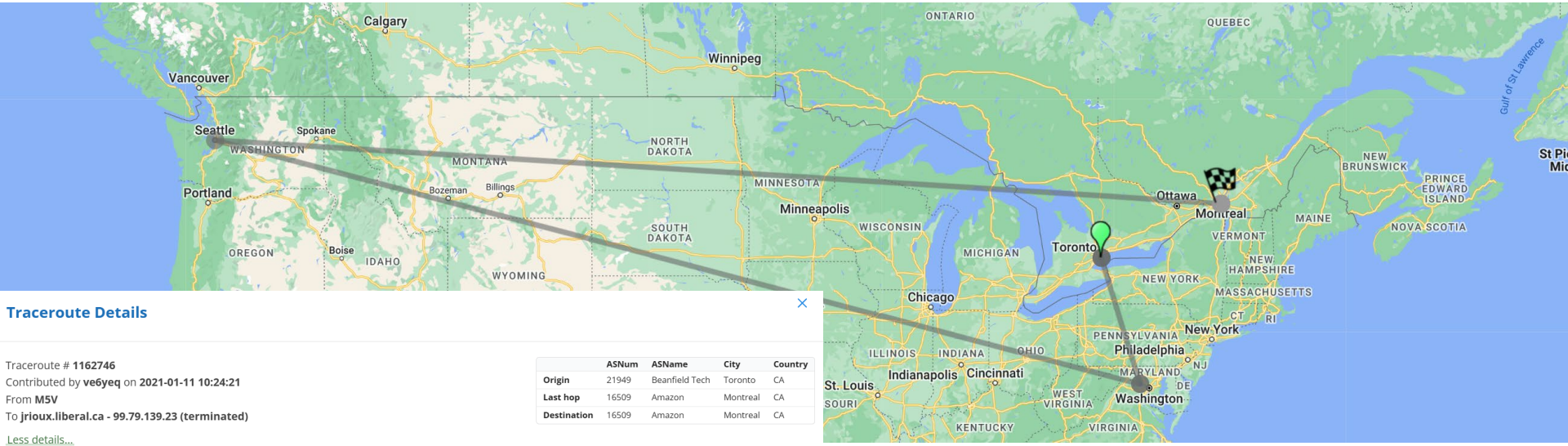
- Eavesdrop
  - Intercepting packets
- Impersonation
  - Fake identity
  - Spoof source address
- Hijacking
  - Take over on-going traffic
  - Redirect packets to elsewhere
- Denial of Service
  - Prevent service from being available

# Border Gateway Protocol (BGP)

The Internet is broken up into **Autonomous Systems** (AS) that are independently managed and connect to each other via gateways (*e.g.*, an ISP)

- These gateway routers communicate using the **Border Gateway Protocol** (BGP) protocol to update routing information
  - *e.g.*, if a router goes down, BGP detects and updates routing information with alternate routes to destinations
  - Routes are updated in peer-to-peer manner, by asking neighbors if they have a route to a certain destination

# Some Traceroute Cases…




**Traceroute Details**

Traceroute # **1162746**
Contributed by **ve6yeq** on **2021-01-11 10:24:21**
From **M5V**
To **jrioux.liberal.ca - 99.79.139.23 (terminated)**

| | ASNum | ASName | City | Country |
|---|---|---|---|---|
| **Origin** | 21949 | Beanfield Tech | Toronto | CA |
| **Last hop** | 16509 | Amazon | Montreal | CA |
| **Destination** | 16509 | Amazon | Montreal | CA |

Less details...

| Hop | IP | Hostname | ASnum | Latencies | Lat | Long | Geocorrection |
|---|---|---|---|---|---|---|---|
| 3 | 72.15.49.67 | te3-2.pe01.151front711.beanfield.com | 21949 | 4\|2\|2\|4 | 43.644671 | -79.384241 | 2 |
| 4 | 206.108.34.31 | gw-beanfield.torontointernetxchange.net | -1 | 2\|2\|2\|11 | 43.644671 | -79.384241 | 2 |
| 5 | 206.108.35.37 | amazon-b.ip4.torontointernetxchange.net | -1 | 2\|2\|2\|2 | 43.644671 | -79.384241 | 2 |
| 6 | 52.93.3.134 | 52.93.3.134 | -1 | 2\|2\|2\|7 | 39.0481 | -77.4728 | |
| 7 | 52.93.3.141 | 52.93.3.141 | -1 | 2\|2\|2\|2 | 39.0481 | -77.4728 | |
| 9 | 54.239.43.29 | 54.239.43.29 | -1 | 9\|9\|9\|9 | 47.6103 | -122.3341 | |
| 10 | 52.94.82.124 | 52.94.82.124 | 16509 | 12\|10\|10\|9 | 45.5 | -73.5833 | |
| 11 | 52.94.83.149 | 52.94.83.149 | 16509 | 13\|9\|9\|9 | 45.5 | -73.5833 | |
| 12 | 52.94.83.196 | 52.94.83.196 | 16509 | 12\|10\|10\|16 | 45.5 | -73.5833 | |
| 13 | 52.94.82.249 | 52.94.82.249 | 16509 | 10\|10\|10\|10 | 45.5 | -73.5833 | |
| 14 | 52.94.81.14 | 52.94.81.14 | 16509 | 9\|9\|9\|10 | 45.5 | -73.5833 | |
| 20 | 99.79.139.23 | ec2-99-79-139-23.ca-central-1.compute.amazonaws.com | 16509 | 9\|10\|10\|10 | 45.4995 | -73.5848 | |

Data from IXMAPS

# Attacks on BGP

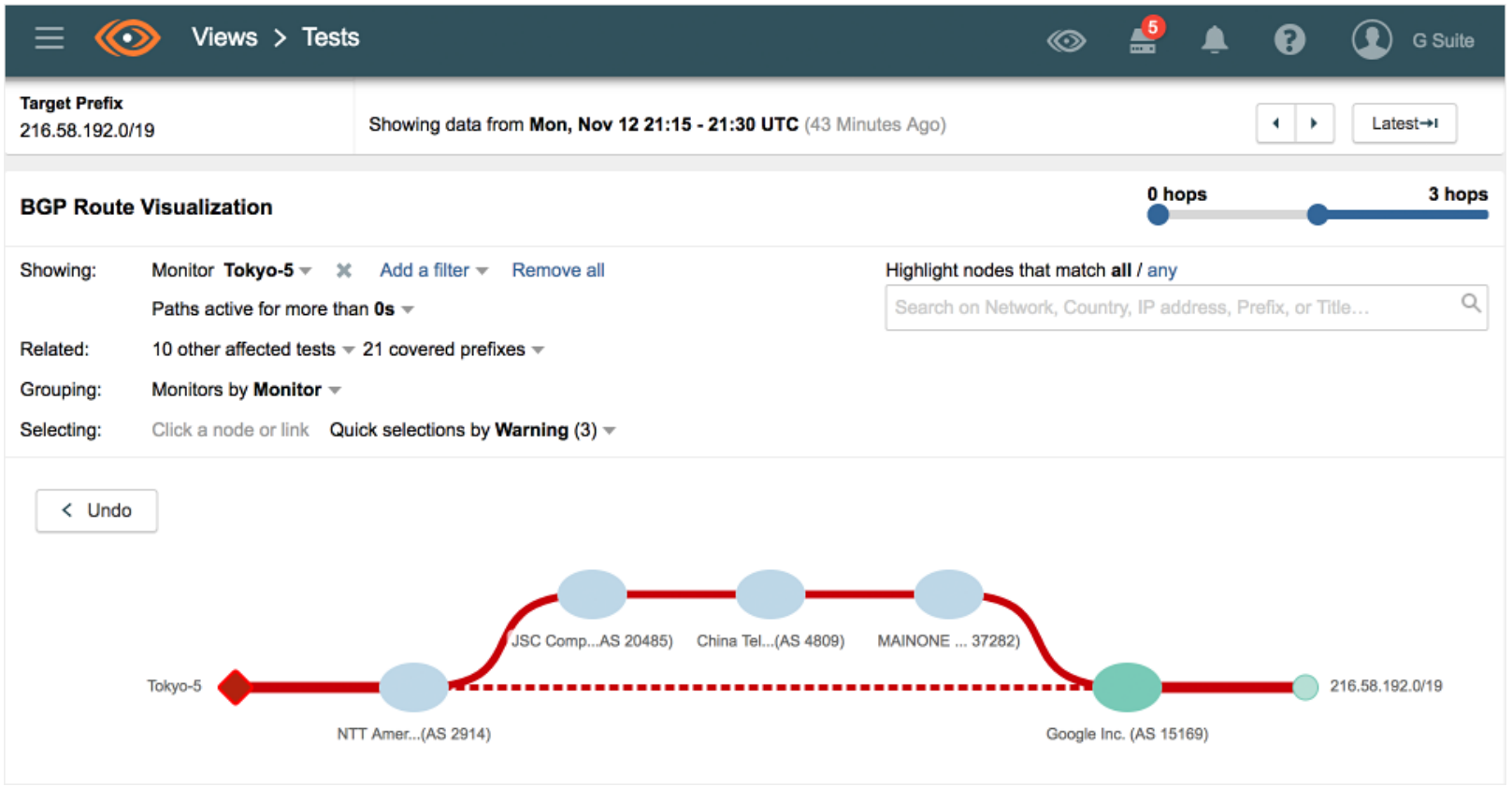
Attackers can compromise the protocol by advertising "good" routes to destinations

- Traffic then gets directed through attacker's gateway
- Routers trust each other and don't authenticate their peers, the BGP packets, or the routes advertised by the peers
- It is assumed that all BGP routers are configured correctly and are not malicious

There have been proposals to secure and include authentication for BGP requests

- Resistance to adoption
- Doesn't really solve the fundamental problem that the routers are all trusted

# BGP Hijack Example



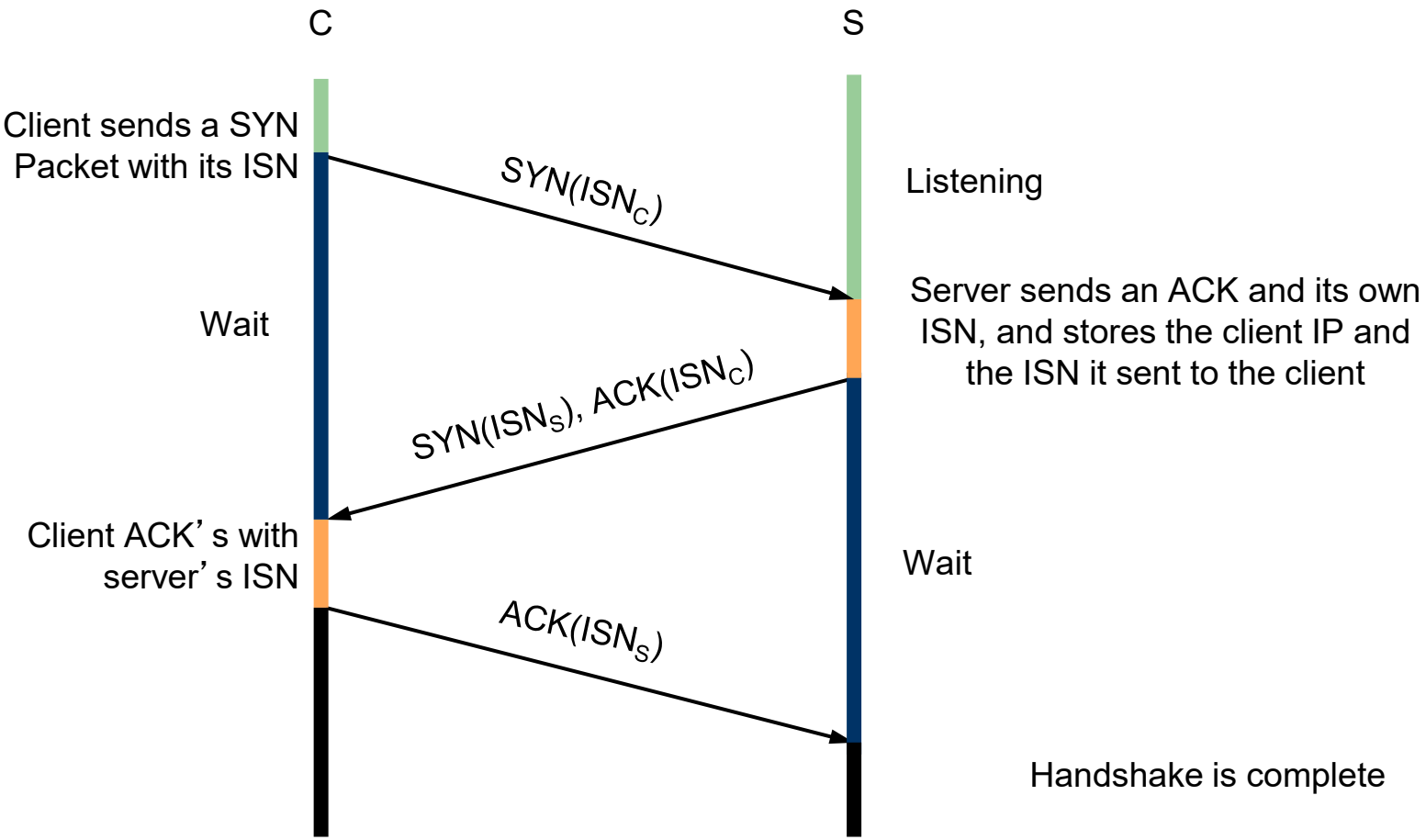

(Source: Thousandeyes *Suspicious announcement for 216.58.192.0/19 showing the best path to Google via Russia, China and Nigeria.*

# BGP Hijack Example


```
traceroute from Google (Ashburn, VA) to ACOnet (Vienna, Austria) at 09:57 Jun 06, 2019
1  *                                                                                0.0
2  195.219.50.2    if-ae-7-2.tcore1.fnm-frankfurt.as6453.net Frankfurt    Germany  86.451
3  *                                                                                0.0
4  *                                                                                0.0
5  118.85.205.233  CHINANET BACKBONE NETWORK          Amsterdam          Netherlands 265.544
6  *                                                                                0.0
7  202.97.52.65    CHINANET backbone network          Frankfurt am Main  Germany     387.218
8  118.85.205.90   CHINANET BACKBONE NETWORK                             China       340.859
9  80.80.225.142   vlan24.cs2.gva.safehost.net        Genève             Switzerland 297.579
10 80.80.225.211   Safe Host Network Geneva           Genève             Switzerland 309.027
11 80.80.225.193   ge-3-1.ds4.gva.safehost.net        Genève             Switzerland 308.962
12 83.137.83.1     euNetworks GmbH                    Vevey              Switzerland 222.019
13 80.86.163.17    Loopbacks and PtP links Switzerl   Braunschweig       Germany     219.624
14 217.71.96.37    ae6.irt1.fra44.de.as13237.net      Frankfurt am Main  Germany     218.007
15 217.71.96.6     ae4.irt1.mun02.de.as13237.net      Munich             Germany     213.565
16 217.71.96.110   ae1.400.irt1.vie08.at.as13237.net  Vienna             Austria     222.668
17 193.171.255.33  ACOnet Services Network            Vienna             Austria     221.573
```

(Source: Arstechnica: BGP event sends European mobile traffic through China Telecom for 2 hours)

# TCP Handshake




$ISN_C$ = Client Initial Sequence Number

$ISN_S$ = Server Initial Sequence Number

# TCP Connection Spoofing

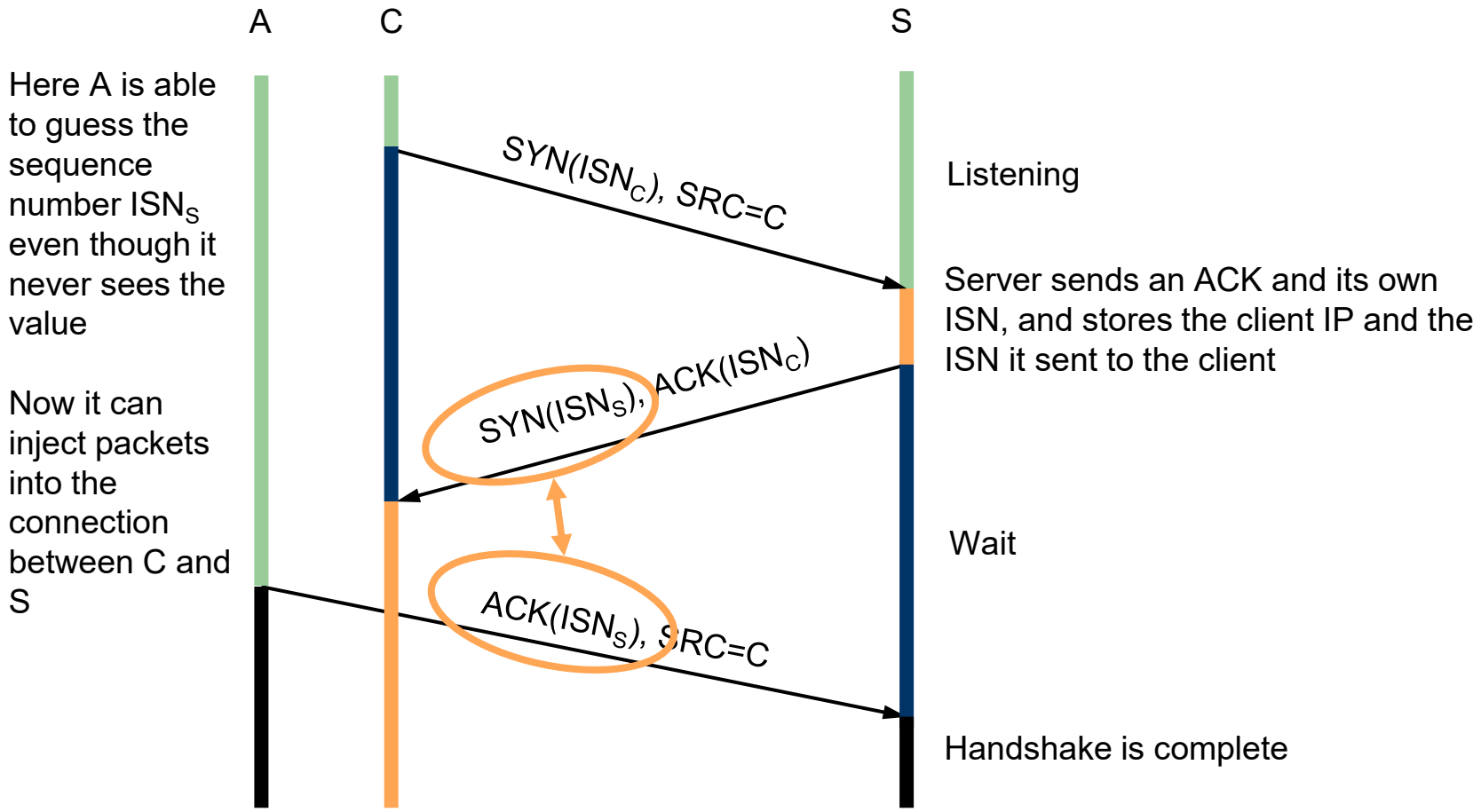
The TCP handshake uses the initial sequence numbers as a weak authenticator

- If these numbers are wrong, the connection is not set up

An attacker can forge a packet with the source IP address set to the client's address

- Forging source IP addresses doesn't allow the attacker to receive packets (still get routed to real source)
- However, if the server's initial sequence number can be guessed, then the attacker can insert new messages into the connection
- Older systems still use sequential sequence numbers, so these numbers are easy to guess

# TCP Connection Spoofing




$ISN_C$ = Client Initial Sequence Number
$ISN_S$ = Server Initial Sequence Number

# Address Resolution Protocol (ARP)

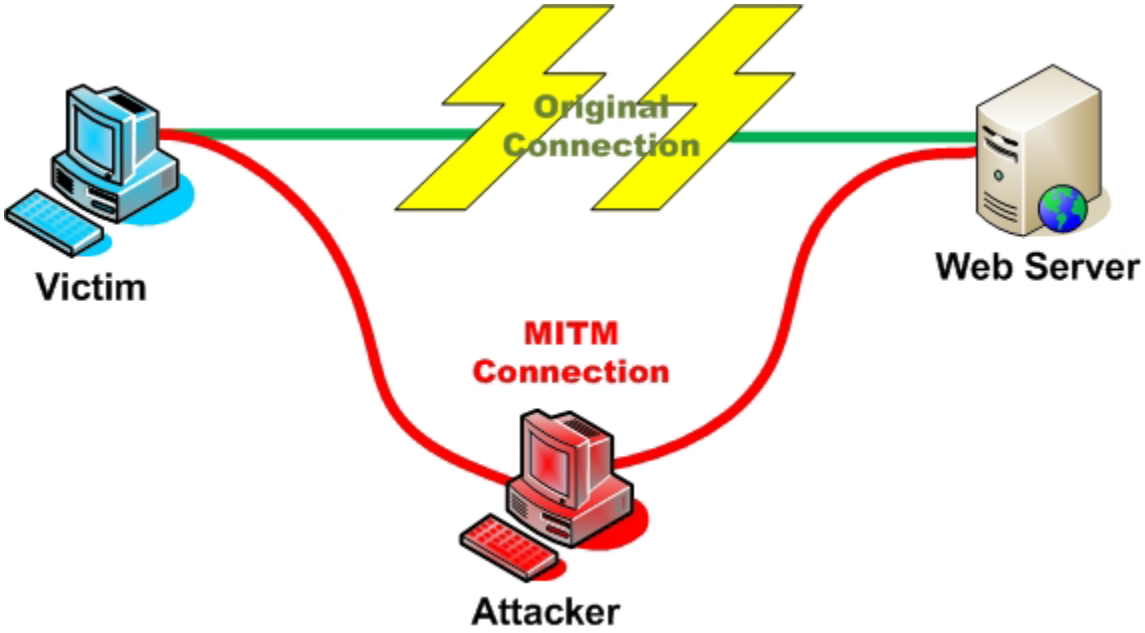
Address Resolution Protocol (ARP)

- The IP layer uses **IP addresses** for routing packets
- ARP is used to map IP addresses to **MAC addresses** so that IP packets can be sent to the link layer
- Packets are sent to the next hop using MAC (*e.g.*, Ethernet) addresses

ARP is simple and uses broadcasting:

- A host that wants to send a packet to IP address **A**, it will perform an **ARP broadcast** to all devices asking which device owns that IP address
- All hosts ignore the broadcast except the host that has IP address **A**, who will respond with its MAC address
- All packets sent to IP Address **A** are sent using the given MAC address

# ARP Posioning




**Reference:** http://www.art0.org/security/man-in-the-middle-attacks-with-ettercap

# Address Resolution Protocol (ARP)

With attacker privileged full access to a host on the network, they can **spoof ARP requests**

- The attacker makes the hacked machine redirect all traffic to itself by responding to all ARP broadcasts
- Every machine starts to think that the hacked machine owns every IP address
- All LAN traffic is redirected to hacked machine which can start faking services, stealing passwords, etc.

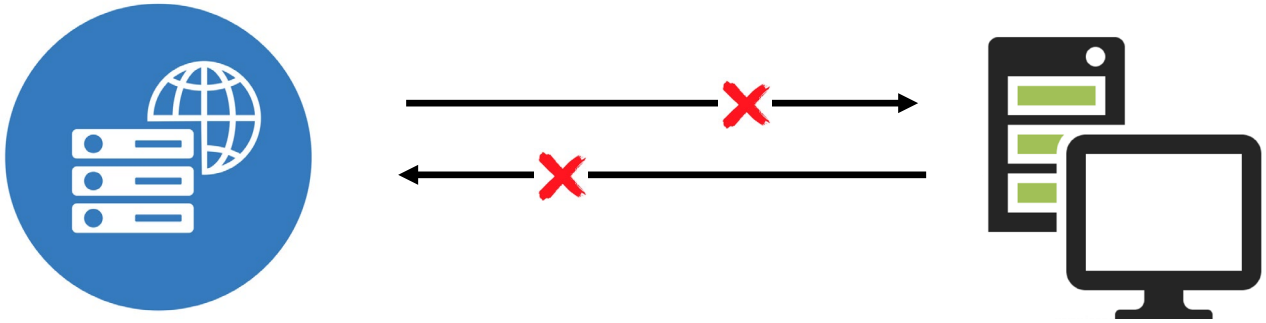
- ARP broadcasts are never forwarded outside of a subnet, so the attacker must control a machine on the same subnet (i.e. LAN)

# Outline

- ~~**Spoofing**~~
  - ~~DNS cache poisoning~~
  - ~~BGP spoofing~~
  - ~~TCP connection Spoofing~~
  - ~~ARP Spoofing~~
- **Denial of Service**
  - Syn-flooding
  - TCP Reset
- **Defenses**
  - TLS
  - IPSec
  - Firewalls

# Denial of Service

- In a denial of service attack, adversary's goal is to prevent user from accessing service:
  - Cause network connections to drop information or terminate early
  - Overwhelm servers with fake requests so that legitimate requests cannot get through



# TCP Reset Attack

A variant of a TCP connection spoofing attack is a **TCP reset attack**

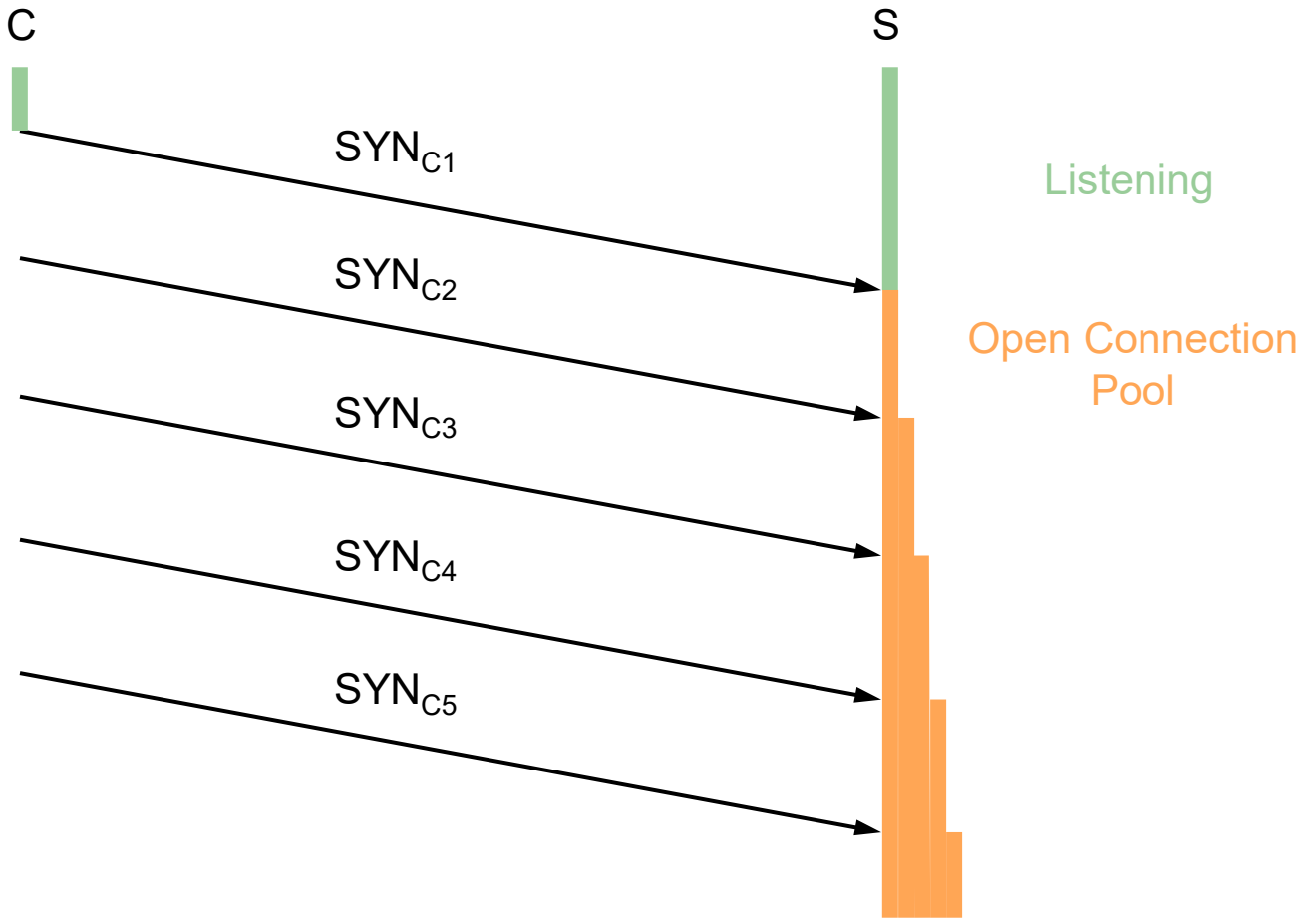
- A TCP reset attack falsely terminates a established TCP connection resulting in denial of service
- The attacker spoofs the sender's connection and sends a RST packet to the receiver
  - Requires IP spoofing and guessing the current packet sequence number
- On receiving a 'valid' RST packet, the receiver immediately terminates the connection
- Defenses are not obvious
  - Requires ignoring bogus RST packets, *e.g.*, multiple packets

# TCP SYN Flooding

Attacker sends many connection requests with spoofed IP source addresses

- Victim allocates resources for each request until some timeout
- Typically, OSs have a fixed bound on these **half-open** connections
- Eventually, the half-open connection queue **resource is exhausted**
- Then, no more requests are accepted, leading to denial of service

# TCP SYN Flooding



# TCP SYN Flooding Defenses

- Reduce half-open connection timeout
- Drop half-open connections randomly
- Use SYN-ACK cookies
  - Client sends SYN
  - Server responds to client with SYN-ACK cookie
    - ISNs = H(src addr, src port, dest addr, dest port, rand)
    - This is a normal response, but server does not save state
  - Honest client responds with ACK(ISNs)
    - No changes required at client
  - Server regenerates ISNs and checks that the client's response matches ISNs
    - rand is derived from a 32-bit time counter
    - Server uses some recent time counter values

# Distributed Denial of Service

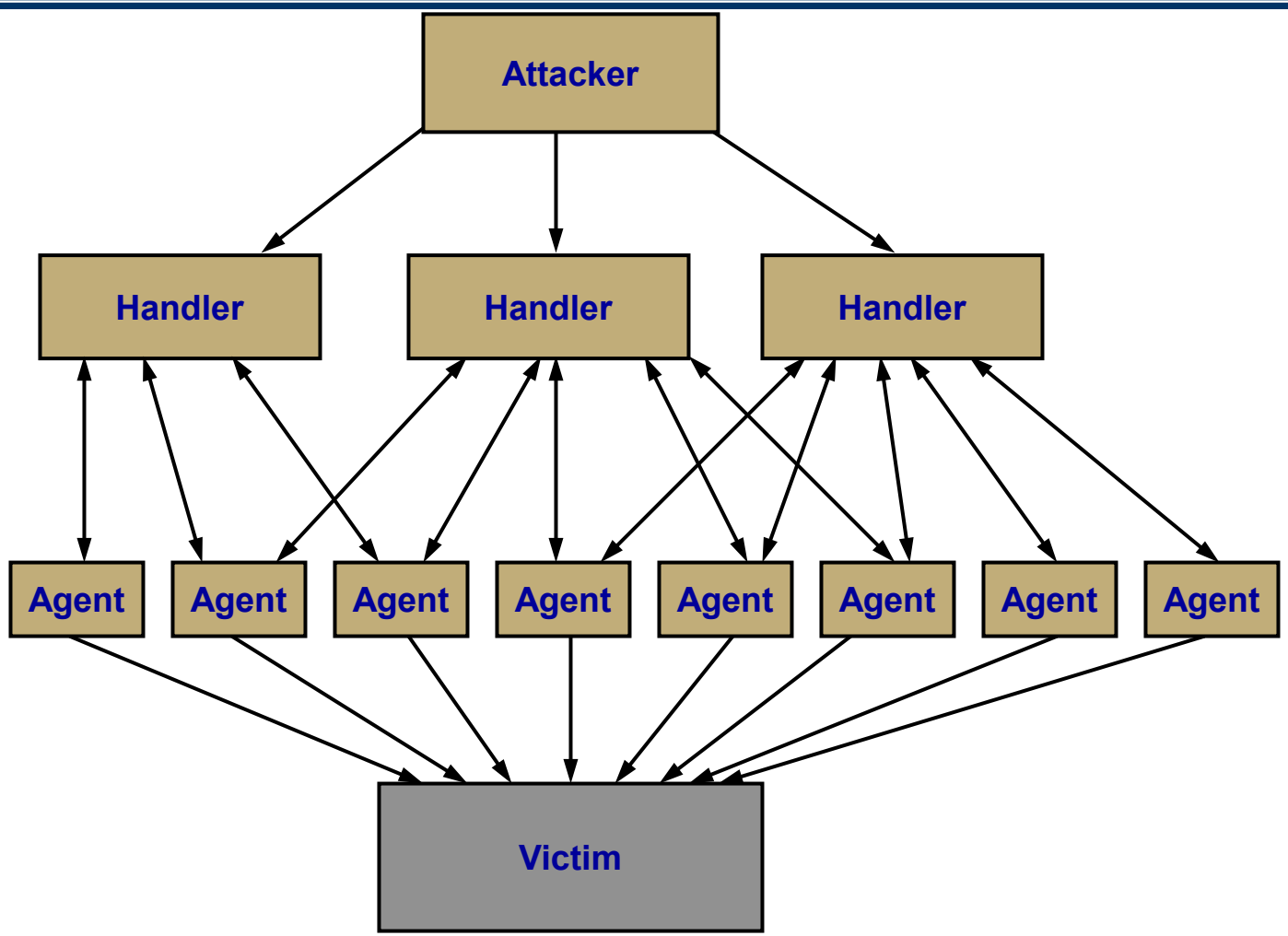
**Denial of Service** (DoS) simply means making a system inaccessible to legitimate users

- Usually, the attacker's goal is to consume as many resources as possible, so that others can't get service
- Typically, the attacker targets bandwidth or memory
  - What resource did the DoS attacks discussed earlier (TCP syn flooding) target?

A bandwidth attack requires flooding the server with packets, so the attacker needs more bandwidth than the victim

- One method used by an attacker is to build up a large number of compromised hosts, and then use them to simultaneously attack a single target
- This is called **Distributed Denial of Service (DDOS)**

# Distributed Denial of Service



# Outline

- ~~**Spoofing**~~
  - ~~DNS cache poisoning~~
  - ~~BGP spoofing~~
  - ~~TCP connection Spoofing~~
  - ~~ARP Spoofing~~
- ~~**Denial of Service**~~
  - ~~Syn-flooding~~
  - ~~TCP Reset~~
- **Defenses**
  - TLS
  - IPSec
  - Firewalls

# Defenses: Use of Cryptography

Cryptographic protocols can be used to defend against many of the attacks on Internet protocols

- Protect against spoofing attacks and injected data
- Generally, don't protect against DoS attacks

- **Application layer**
  - SSL/TLS, SSH
- **Transport layer**
  - Use cryptographically random ISNs for TCP/IP
- **Network layer**
  - IPSec

# TLS – Transport Layer Security

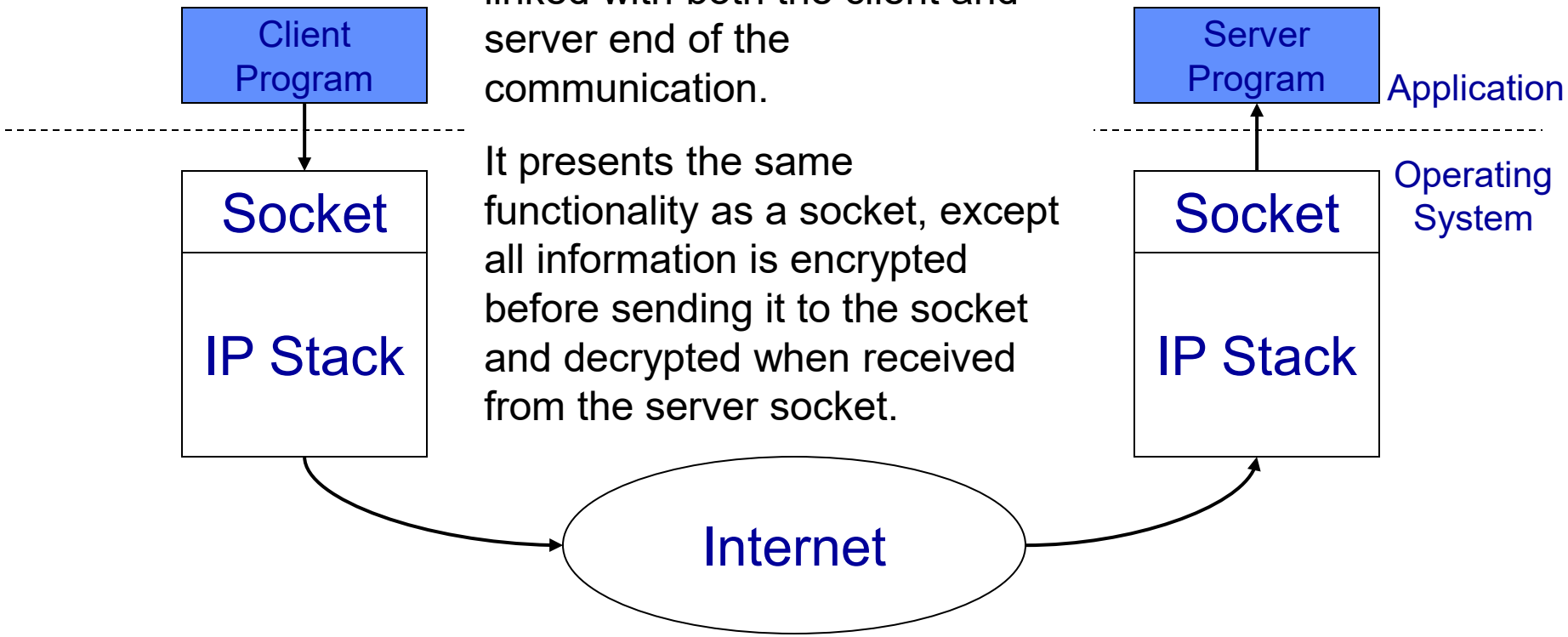
- Security properties:
  - Confidentiality: via symmetric encryption
  - Integrity: via cryptographic hashing
  - Authentication: via public-key cryptography

- Implementation:
  - Supported by most browsers and web servers
  - Can secure web sessions and any other application
  - End-to-end security
  - Secure socket layer (SSL) deprecated [RFC 6101]
  - TLS 1.3 [RFC 8846]

# The SSL/TLS Protocol

SSL/TLS is implemented as a application level library that is linked with both the client and server end of the communication.

It presents the same functionality as a socket, except all information is encrypted before sending it to the socket and decrypted when received from the server socket.



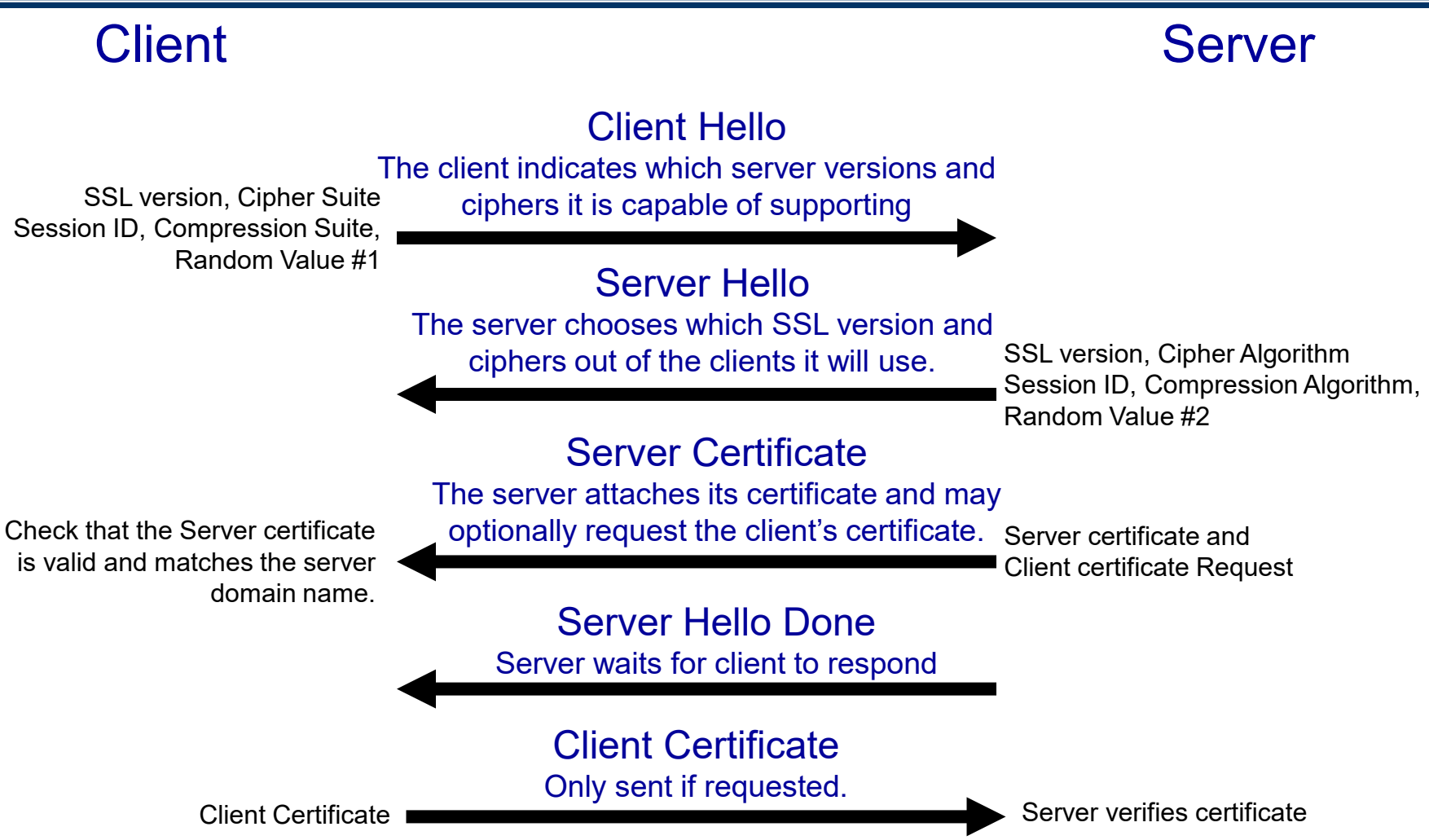
# SSL/TLS Mechanics

- SSL/TLS has 2 phases:
  1. **Key Exchange or Handshake**: The initial phase establishes a shared secret key between the sender and the receiver. Any authentication is also performed. Compatibility between different versions is also handled here. Since this happens only once for any exchange, it can be relatively slow
  2. **Communication**: Once the keys are setup, an arbitrary number of messages can be exchanged between the two parties in both directions. This could involve a large amount of data, so this phase is pretty efficient.

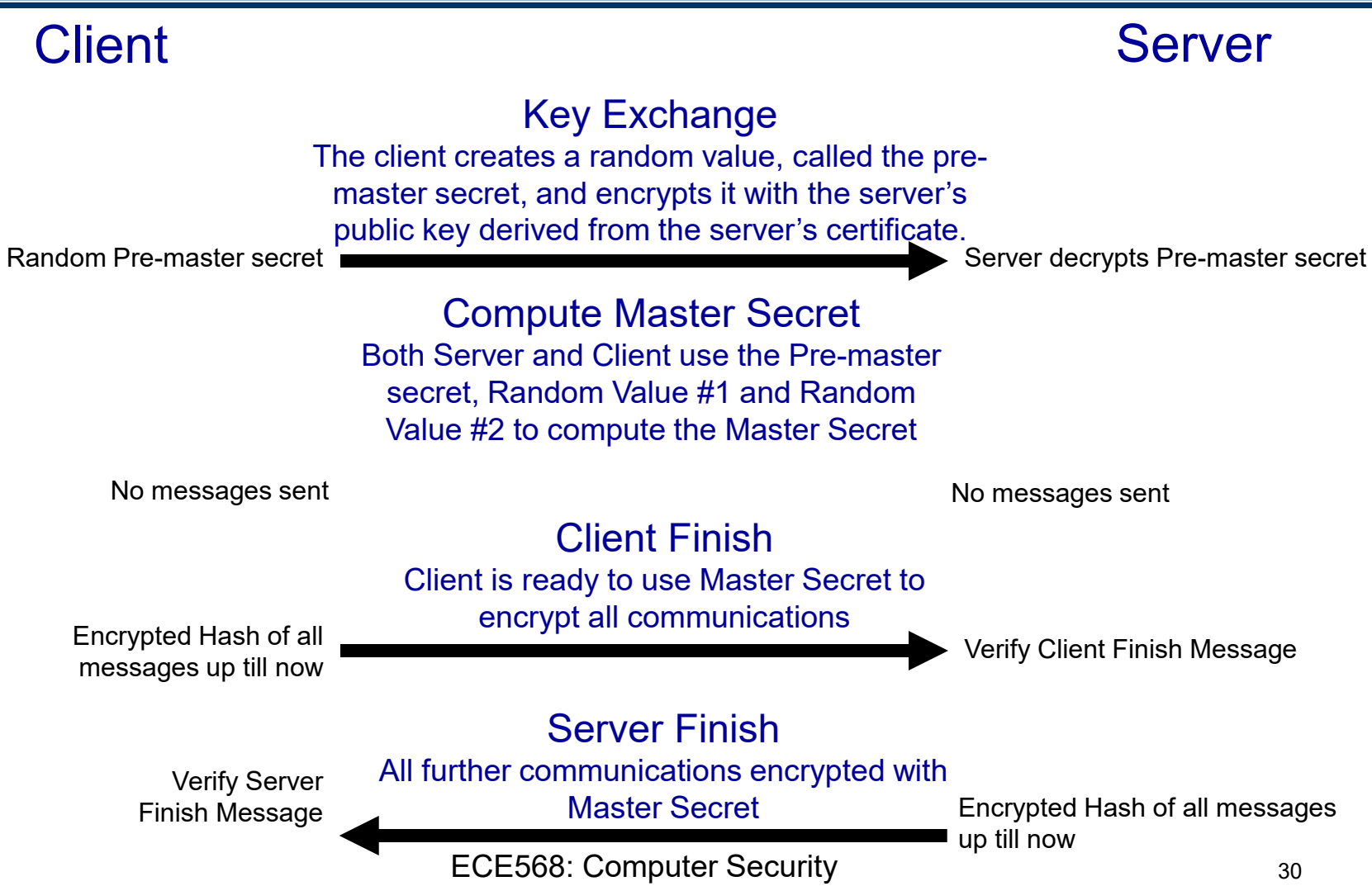
# SSL/TLS Handshake

- The SSL/TLS Handshake has 3 purposes:
  1. Establish the suite of ciphers each side supports and what version of the protocol is being used.
  2. Securely establish a shared secret that can be used as a session key (for symmetric encryption).
  3. Authenticate each other's identities via certificates. Note this authenticates the identities of the machines, not the users (i.e. people) making the requests.
     - User authentication (i.e. a website asking you for a username/password) is not done by the SSL/ TLS protocol, but by scripts or servlets on the web server.
     - Note that client machine authentication is optional and usually not done (since web servers will connect to any client).

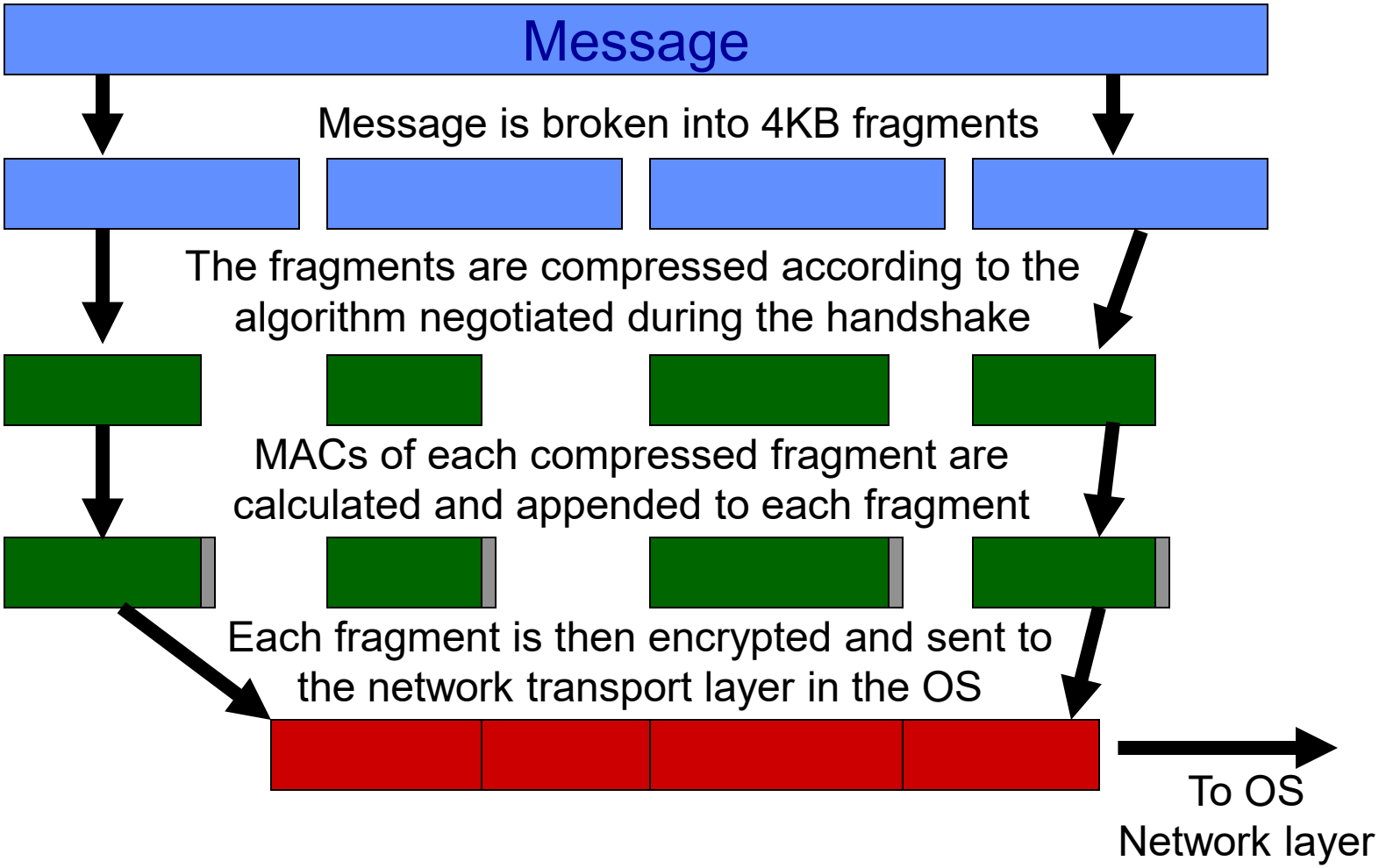
# SSL/TLS Handshake Detail



# SSL/TLS Handshake Detail



# SSL/TLS Communication



# IPSec

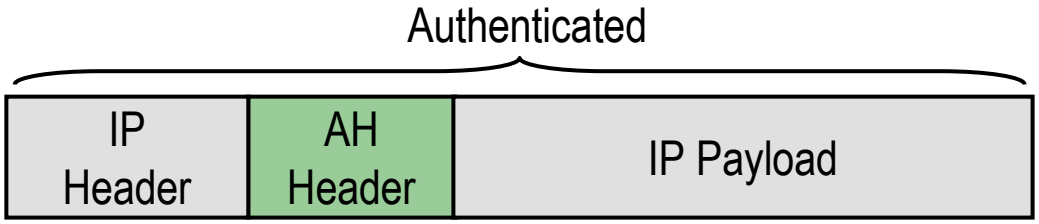
IPSec was developed by the IETF

- Designed to provide the following at IP layer:
  - Message confidentiality
  - Message integrity
  - Source authentication
  - Protection against replay
- IPSec is composed of two protocols:
  - **Authenticated Headers** (AH)
    - [RFC 4302]
    - Provides all of the above properties except confidentiality
  - **Encapsulating Security Payload** (ESP)
    - [RFC 4303]
    - Provides all of the above properties

# IPSec Protocols

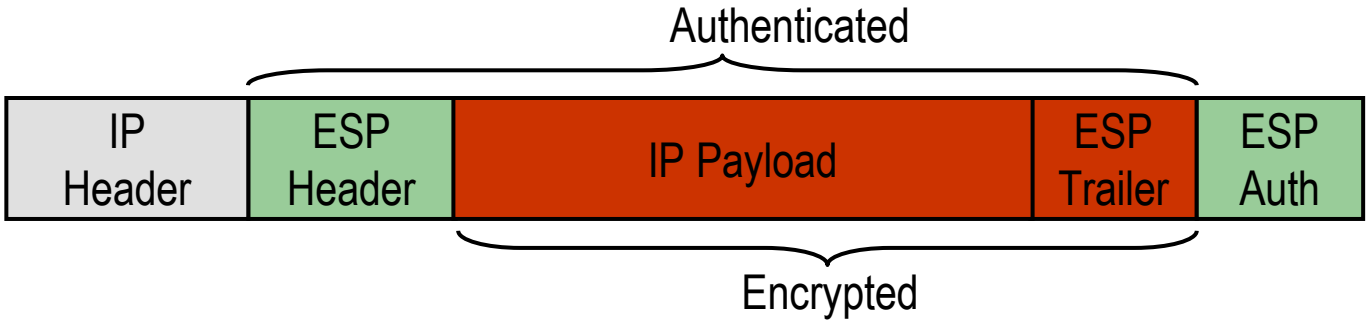
**Authenticated Headers**

- Protects the IP packet (except IP header fields that are altered during transit) using a MAC stored in AH Header




**Encapsulating Security Payload**

- Payload is encrypted to protect contents



# IPSec Modes

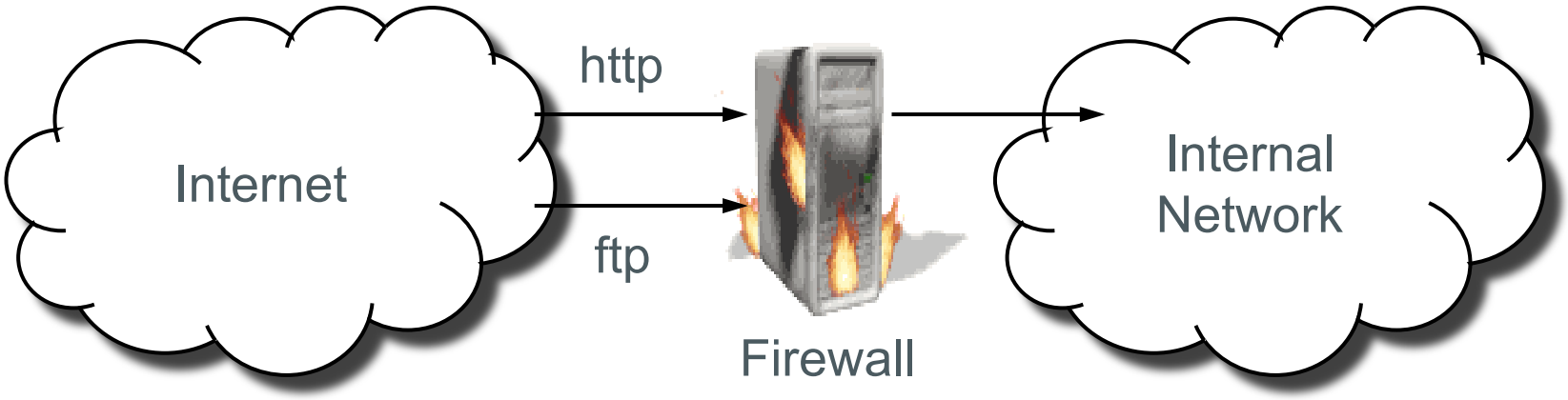
IPSec can be used even when all routers on the Internet are not IPSec enabled. IPSec has two modes:

- **Transport mode**
  - Both endpoints support IPSec, but intermediary routers do not
  - This mode encrypts/authenticates the packet **payload**
  - Similar to SSL, SSH, etc.
- **Tunnel mode**
  - Endpoints do not support IPSec, but endpoint gateways do
  - This mode encrypts/authenticates the packet **header and payload** and encapsulates it in another regular IP packet
  - Similar to SSH tunneling or VPN software

# Firewalls

A firewall is a machine whose function is to control access to an internal network

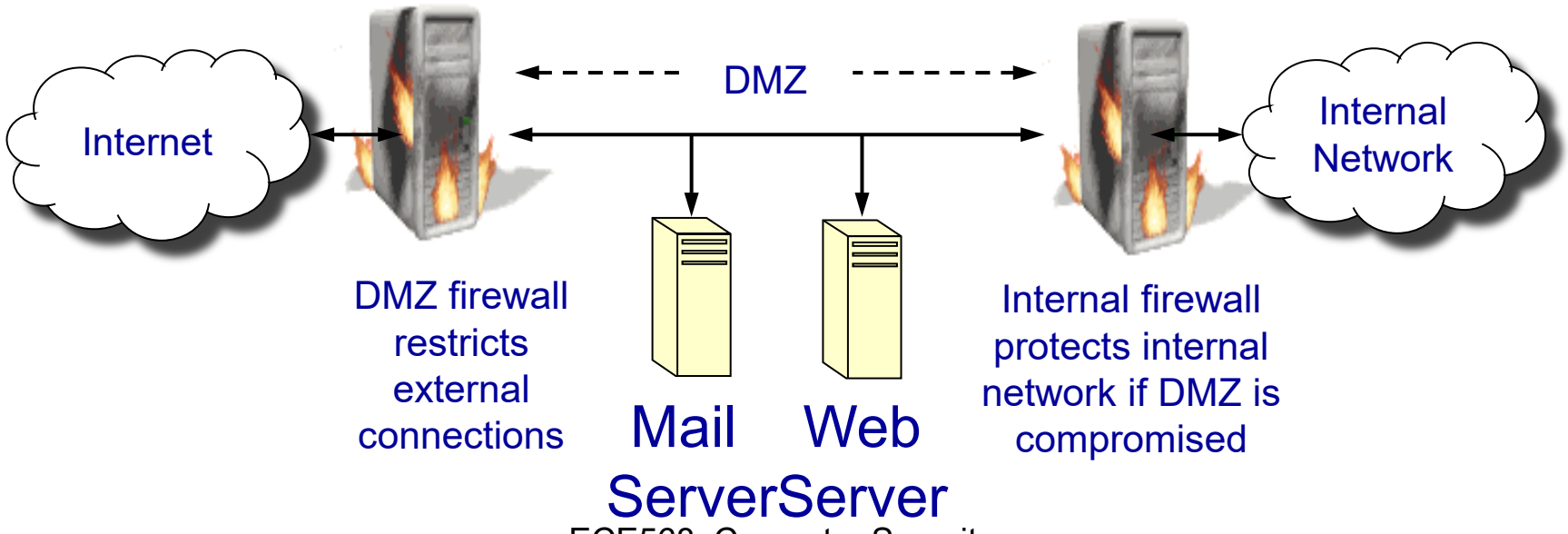
- Some types of connections are allowed while other are not
- Access policy is usually determined by the port number which indicates the type of service being accessed



# Firewall Deployment

A firewall is normally placed at entry points between an internal and an external network

- Sometimes certain machines need to be accessible both externally and internally, requiring an internal firewall and a Demilitarized Zone (DMZ) firewall



# Firewall Example

Home routers provide simple firewall functionality

- Filter based on port number, and possibly source address

Sophisticated firewalls filter on numerous criteria:

- Protocol, frequency of packets, etc.
- Allowing incoming packets only when an initial outgoing connection has been established, etc.

Example: a Linux system can be configured with firewall rules with the **iptables** command

- Each rule has a set of criteria that the kernel matches with each packet (*i.e.*, src addr, dest addr, protocol, etc.)
- Each rule also has an target (accept, drop, log, etc.)
- The firewall tables are composed of chains of rules
- The OS tries each rule in the chain until one matches
- If a match is found, the target is executed

# IPTables Example

```
% iptables -L
…
Chain INPUT (policy DROP)
target     prot opt source          destination
ACCEPT     all  --  anywhere        anywhere     state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere        anywhere     state NEW tcp dpt:http
ACCEPT     tcp  --  anywhere        anywhere     state NEW tcp dpt:ssh
ACCEPT     tcp  --  128.100.8.51    anywhere     state NEW tcp dpt:smtp
```

- The INPUT chain determines whether hosts outside of the firewall can make new connections
  - Rule 1 says that all packets on already established connections are allowed (to come in)
  - Rules 2-4 allow http, ssh and smtp connections to be initiated from the outside, but smtp connections can only come from 128.100.8.51