Cristina Nita-Rotaru

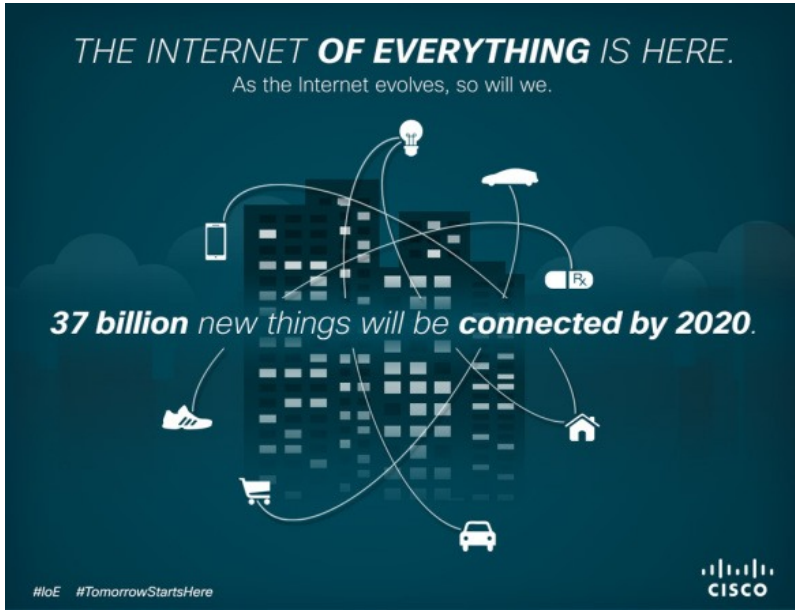
CS526: Information security

Network security

2: Background

Internet of Things

- Everything is connected
- Many types of devices
- Tremendous amount of data
- Available via cloud computing, accessed via personal devices



Lots of networks …

- Networks:
  - The Internet
  - Wifi
  - Cellular
  - Sensor
- Protocol: Defines rules of sending/receiving packets:
  - Format and type of the packets
  - Actions in response of receiving a certain type of packets …

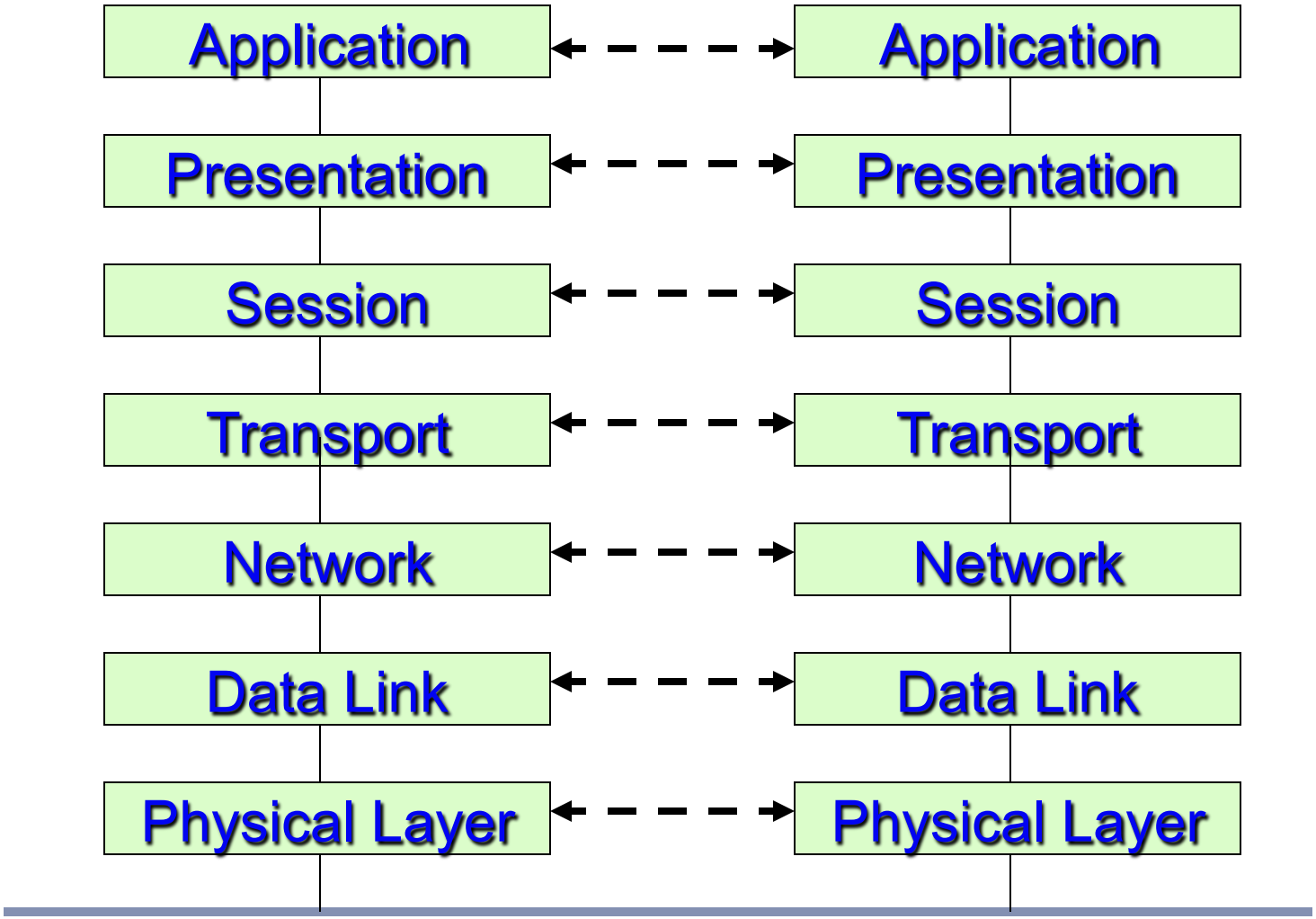
A network of networks

- Internet is a "network of networks"
- Autonomous System (AS): a network, a single administrative domain, can span more organizations
- How do those networks connect
  - Internet Exchange (IX)
  - Network Access Points (NAP)
  - Metropolitan Area Exchange (MAE)

Routing

- IP address identifies a computer
  - IPv4 - 32 bits, IPv6 - 128 bits
- Routing protocols: propagate information about routes to reach hosts (IP addresses) or networks (IP prefixes)
- Algorithms:
  - Distance vector protocols
  - Link-state protocols
  - Path vector protocols
- Relative to an AS
  - Inter-routing: RIP, OSPF
  - Intra-routing: BGP

OSI/ISO Model



Types of addresses

- Media Access Control (MAC) addresses in the network access layer
  - Associated with network interface card (NIC)
  - 48 bits or 64 bits
- IP addresses for the network layer
  - 32 bits for IPv4, and 128 bits for IPv6
  - E.g., 128.3.23.3
- IP addresses + ports for the transport layer
  - E.g., 128.3.23.3:80
- Domain names for the application/human layer
  - E.g., www.purdue.edu

Routing and translation of addresses

- Translation between IP addresses and MAC addresses
  - Address Resolution Protocol (ARP) for IPv4
  - Neighbor Discovery Protocol (NDP) for IPv6
- Routing with IP addresses
  - TCP, UDP, IP for routing packets, connections; IP communication between hosts, TCP and UDP between processes
  - Border Gateway Protocol for routing table updates
- Translation between IP addresses and domain names
  - Domain Name System (DNS)

NATs and their implications

- There are not enough IP addresses
- Solutions: IPv6 or ….Network Address Translation (NAT)
- NAT allows a single device, to act as an agent between the Internet (or "public network") and a local (or "private") network: only a single, unique IP address is required to represent an entire group of computers
- Computers can not communicate directly, STUN client-server protocol allows computers to discover each other behind a NAT (learn their public addresses), but requires presence of STUN server

Address Resolution Protocol (ARP)

- Interface between Link layer and Network Layer
- Allows hosts to query who owns an IP address on the same LAN
- Owner responds with hardware address
- Allows changes to link layer to be independent of IP addressing

Internet Protocol – IP

- IP is the current delivery protocol on the Internet, between hosts.
- IP provides ‘best effort’, unreliable delivery of packets.
  - IPv4 is the most used routing protocol on the Internet
  - IPv6, a newer version, still under adoption

https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption

User Datagram Protocol - UDP

- Runs on top of IP
- Connectionless protocol for a user process
- Minimal guarantee
  - No connection established
  - Unreliable transmission: no guarantee that the packets reach their destination
  - Error detection
  - No acknowledgment
  - No flow control

Transmission Control Protocol - TCP

- Connection oriented protocol for a user process:
  - Established a connection (channel) between two end-points
  - Reliable, full-duplex channel: acknowledgements, retransmissions, timeouts, flow-control
  - The packets are delivered in the same order in which they were sent.
  - Close the connection

Ports

- Remember:
  - Hardware addresses identity network cards
  - IP addresses identify hosts
  - Names identify hosts in a human friendly way.
- However, transport protocols (TCP and UDP) ensure communication between processes.
- How do computers differentiate what data is for which process?
- Port numbers
  - 16-bit numbers

Ports contd.

- In general servers use well-known ports, while clients use ephemeral ports
- Example: port 80 is assigned to web server (HTTP)
- Port numbers:
  - Well-known ports: 0 - 1023
  - Registered ports: 1024 – 49151
  - Dynamic/private ports: 49152 - 65535

IP Multicast

- Provides support for group communication: send to multiple parties
- Groups are specified by reserved IP multicast addresses 224.0.0.0 to 239.255.255.255.
- Unreliable communication
- IGMP is used to dynamically register individual hosts in a multicast group on a particular LAN.
- Network cards recognize IP multicast addresses: hosts that did not subscribe to a particular group will not process those packets (unlike broadcast that is processed by all hosts in a network segment)

Distance-vector routing - RIP

- Each node:
  - Maintains a vector with distances to all of the nodes.
  - Sends periodically its distance-vector to all its neighbors.
  - Updates its distance vector based on the information received from the neighbors (shortest path Bellman-Ford): for each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement.
- Examples: RIP uses distance-vector routing
  - Prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination.
  - The maximum number of hops allowed for RIP is 15
- RIP is a UDP-base protocol

Link-state routing - OSPF

- Each node:
  - Maintains global view of the network.
  - Sends periodically the current state of all links (link-state updates or advertisements) to all nodes (via flooding).
  - Notes the change and recompute its routes (use shortest-path – Dijkstra algorithm) to destination.
- Less bandwidth-intensive than Distance-Vector, but more complex and more computational and memory intensive.
- Examples: OSPF uses link-state routing.
- OSPF does not use TCP or UDP but uses IP directly

Path vectors - BGP

- Similar to distance vector protocols, but routing updates contain an ordered list of the path of traversed "nodes"
- Example: BGP
  - Routing updates contain an ordered list or AS path of traversed autonomous systems and a set of network prefixes belonging to the first AS in the list (UPDATE messages)
  - Uses TCP to exchange routing updates
  - Each BGP router receives UPDATEs from its neighbors and selects one path for each prefix as the "best" and reports that path to its neighbors (before that it has to withdraw the "old" path)
  - Selecting "best path": policies, local preference, shortest AS path, other metrics

Domain Name System (DNS)

- Distributed, hierarchical database that maps host names with IP addresses
- ICANN oversees the domain name assignments
- Uses UDP
- Tree structure
  - Divided into zones
  - Delegating responsibilities
- Name servers
  - Authoritative information (hints to whom might be able to answer the request)
  - Cached data updated periodically

Threats in networking

- Confidentiality
  - e.g. Packet sniffing
- Integrity
  - e.g. Session hijacking
- Availability
  - e.g. Denial of service attacks
- Common
  - e.g. Address translation poisoning attacks
  - e.g. Routing attacks

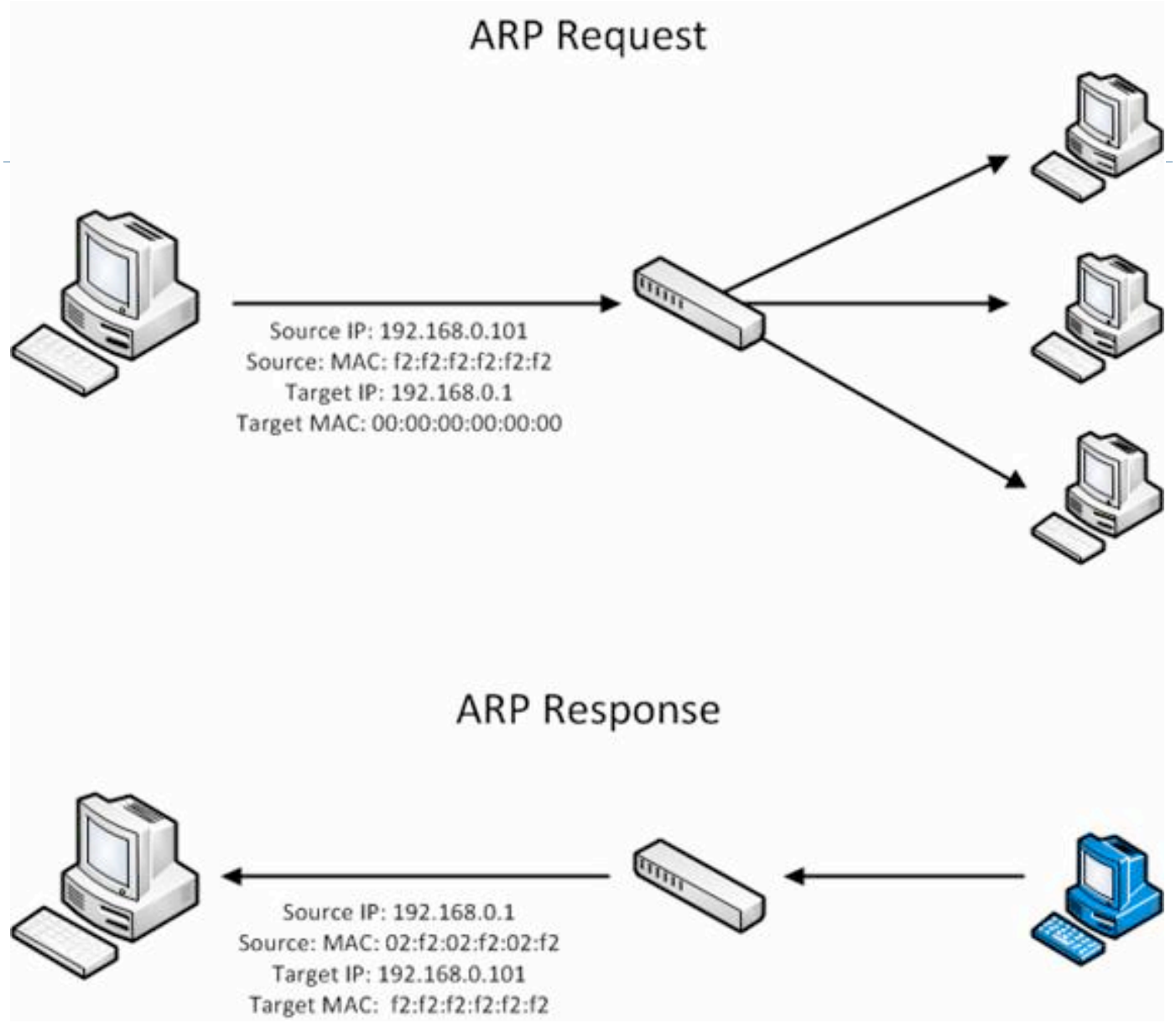
Examples security problems

- ARP is not authenticated
  - APR spoofing (or ARP poisoning)
- Network packets pass by untrusted hosts
  - Packet sniffing
- TCP state can be easy to guess
  - TCP spoofing attack
- Open access
  - Vulnerable to DoS attacks
- DNS is not authenticated
  - DNS poisoning attacks

2: Attacks against ARP

Address Resolution Protocol (ARP)

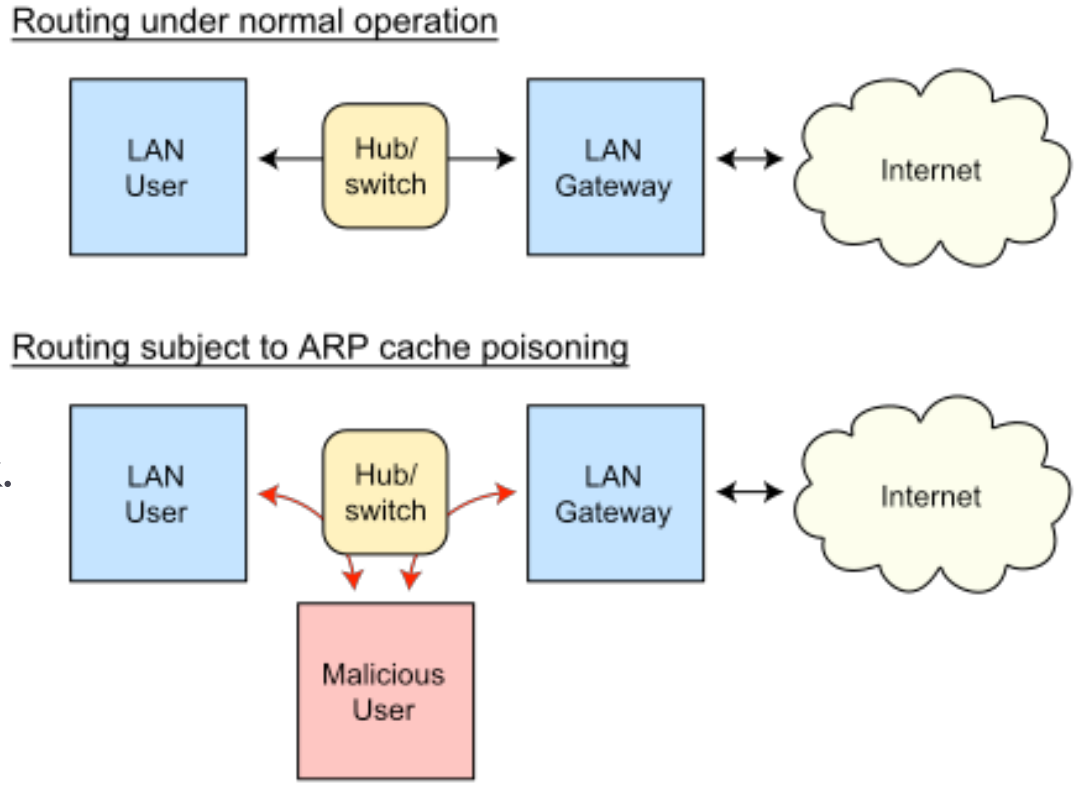
- Primarily used to translate IP addresses to Ethernet MAC addresses
  - The device drive for Ethernet NIC needs ARP to send a packet
- Also used for IP over other LAN technologies, e.g. IEEE 802.11
- Each host maintains a table of IP to MAC addresses
- Message types:
  - ARP request
  - ARP reply
  - ARP announcement



http://www.windowsecurity.com

ARP spoofing (ARP Poisoning)

- Send fake or 'spoofed' ARP messages to an Ethernet LAN
  - To have other machines associate IP addresses with the attacker's MAC
- Legitimate use
  - Redirect a user to a registration page before allow usage of the network.
  - Implementing redundancy and fault tolerance



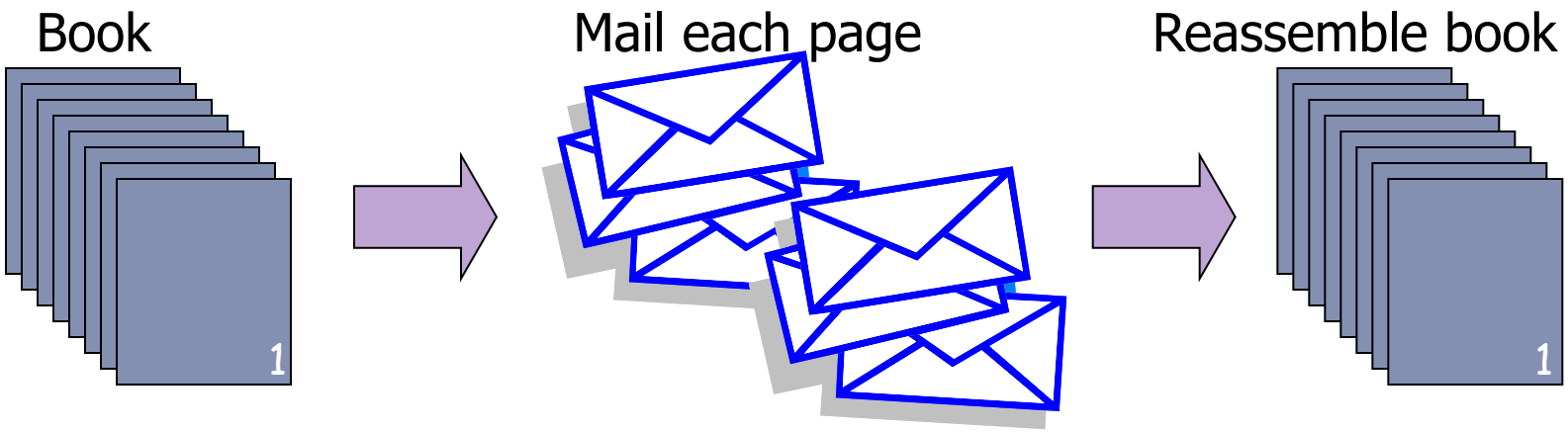
ARP spoofing defenses

- Static ARP table
- DHCP Certification (use access control to ensure that hosts only use the IP addresses assigned to them, and that only authorized DHCP servers are accessible).
- Detection: Arpwatch (sending email when updates occur)

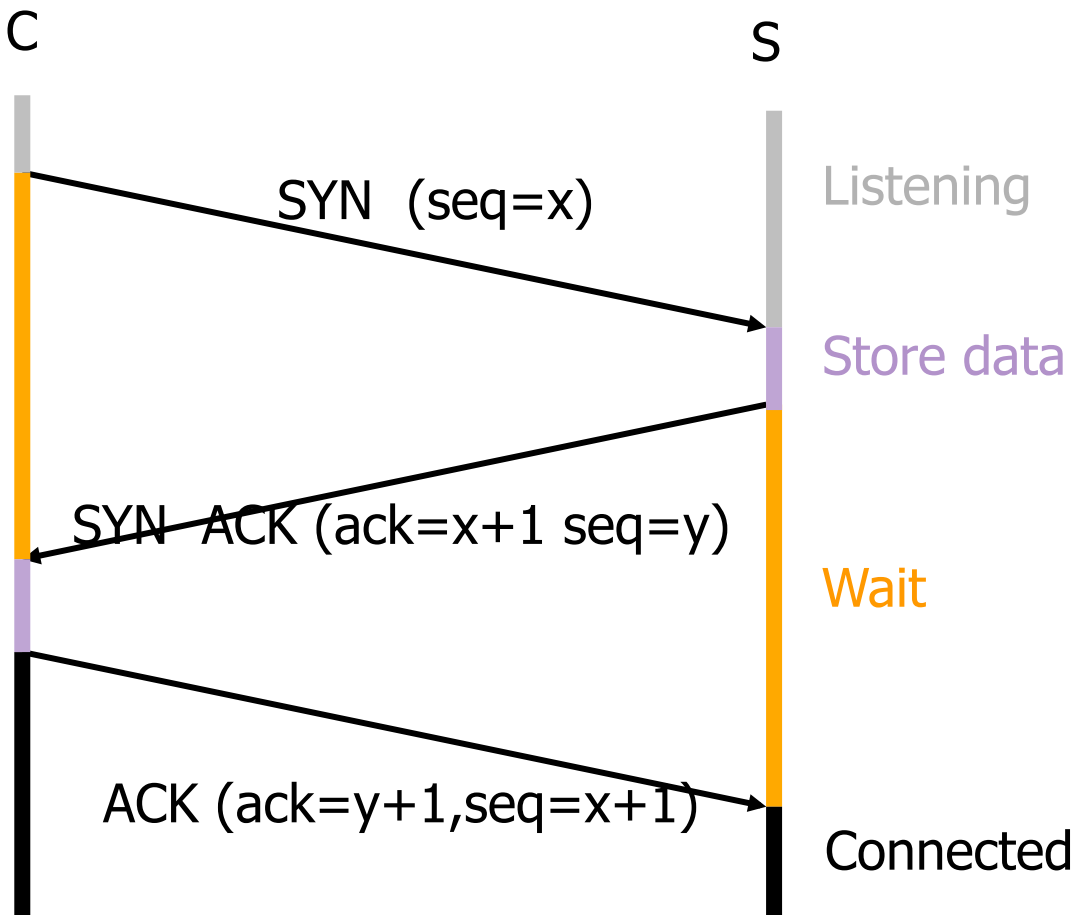
3: Attacks against TCP

Transmission Control Protocol - TCP

- Connection oriented protocol for a user process: establishes a connection (channel) between two end-points
  - Reliable, full-duplex channel: acknowledgements, retransmissions, timeouts
  - Messages broken in packets
  - Congestion control mechanisms
  - Packets are delivered in the same order in which they were sent



TCP handshake



- Resources allocated; There is a max. number of connections that can be in this state (SYN_RECVD state)
- Wait for the ACK (75 seconds)
- If timeout expires or RST received, data deallocated
- If ACK received, connection established, can also contain data.

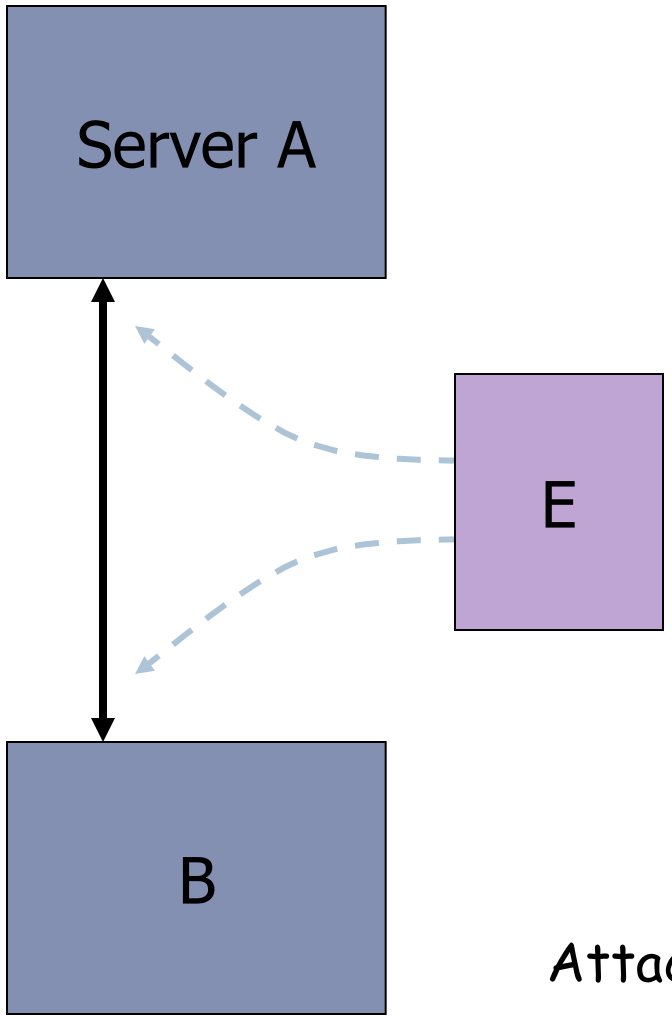
TCP sequence numbers

- Sequence number (32 bits) – has a dual role:
  - If the SYN flag is set, then this is the initial sequence number. The sequence number of the actual first data byte is this sequence number plus 1.
  - If the SYN flag is clear, then this is the accumulated sequence number of the first data byte of this packet for the current session.
- Acknowledgment number (32 bits) –
  - If the ACK flag is set then this the next sequence number that the receiver is expecting.
  - This acknowledges receipt of all prior bytes (if any).

TCP sequence prediction attack

- Predict the sequence number used to identify the packets in a TCP connection, and then counterfeit packets.
- Adversary: do not have full control over the network, but can inject packets with fake source IP addresses
  - E.g., control a computer on the local network
- TCP sequence numbers are used for authenticating packets
- Initial seq# needs high degree of unpredictability
  - If attacker knows initial seq # and amount of traffic sent, can estimate likely current values
  - Some implementations are vulnerable

Blind TCP session hijacking



- A, B trusted connection
  - Send packets with predictable seq numbers
- E impersonates B to A
  - Opens connection to A to get initial seq number
  - DoS B's queue
  - Sends packets to A that resemble B's transmission
  - E cannot receive, but may execute commands on A

Attack can be blocked if E is outside firewall.

Risks from session hijacking

- Inject data into an unencrypted server-to-server traffic, such as an e-mail exchange, DNS zone transfers, etc.
- Inject data into an unencrypted client-to-server traffic, such as ftp file downloads, http responses.
- Spoof IP addresses, which are often used for preliminary checks on firewalls or at the service level.
- Carry out MITM attacks on weak cryptographic protocols.
  - often result in warnings to users that get ignored
- Denial of service attacks, such as resetting the connection.

DoS vulnerability caused by session hijacking

- Suppose attacker can guess seq. number for an existing connection:
  - Attacker can send Reset packet to close connection. Results in DoS.
  - Naively, success prob. is 1/232 (32-bit seq. #'s).
  - Most systems allow for a large window of acceptable seq. #'s
    - Much higher success probability.
- Attack is most effective against long lived connections, e.g. BGP.

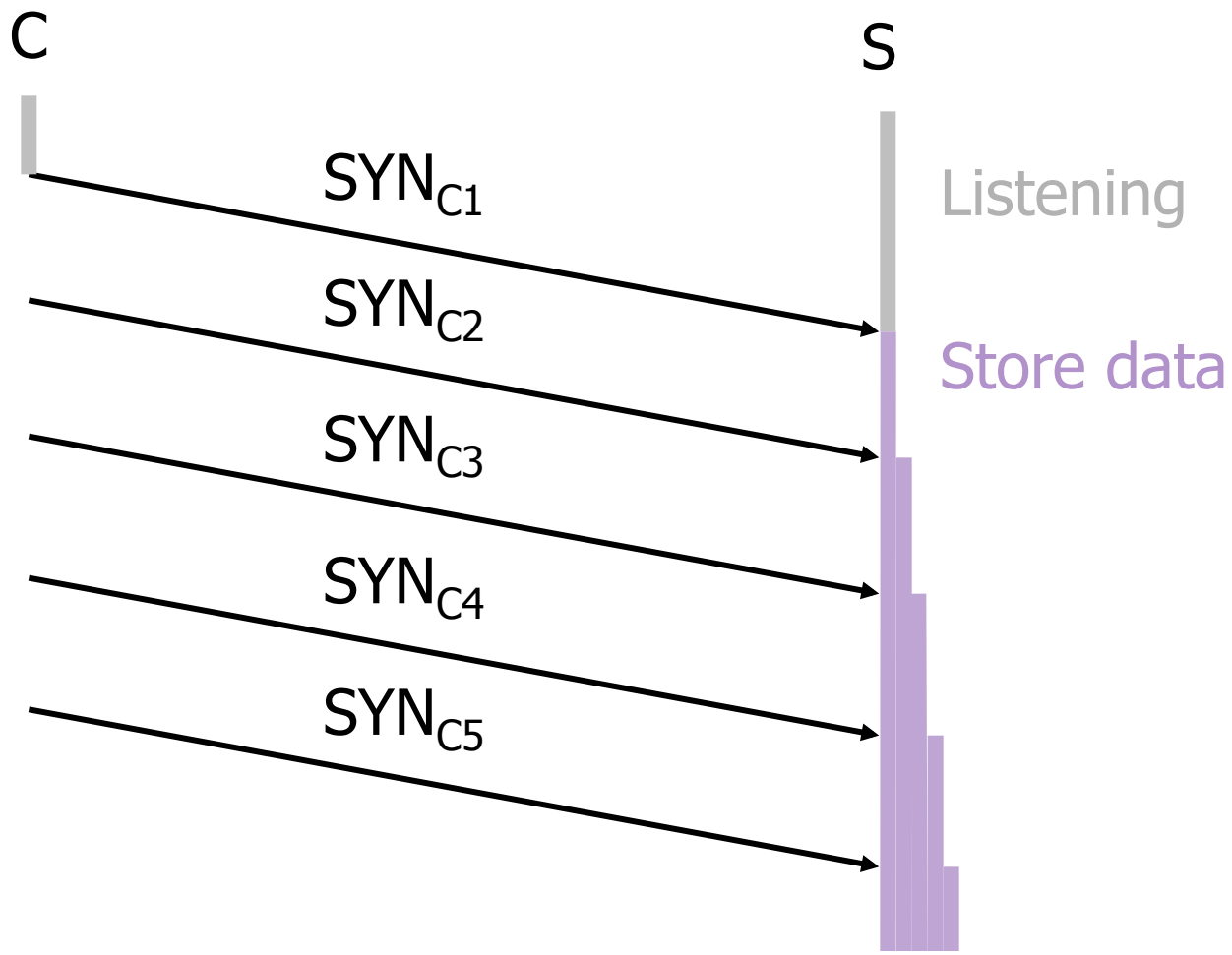
Categories of denial-of-service attacks

	Stopping services	Exhausting resources
Locally	• Process killing • Process crashing • System reconfiguration	• Spawning processes to fill the process table • Filling up the whole file system • Saturate comm bandwidth
Remotely	• Malformed packets to crash buggy services	• Packet floods (Smurf, SYN flood, DDoS, etc)

SYN flooding attack

- An attacker sends many SYN with source address spoofed packets to a target.
- If the limit is reached, target machine will refuse any incoming connections till the timeout expires.
- Spoofed address chosen to be a non-existent one (If the spoofed address belongs to a machine, then SYN+ACK packet will reach that machine and trigger a RST answer that will close the connection).

SYN flooding



Basis of the attack

- There is no authentication of the source of the packets
- Addresses can be spoofed
- The protocol requires asymmetric allocation of resources

Configuration optimizations

- System configuration
  - Reduce the timeout to 10 seconds
  - Increase the size of the queue
  - Disable non-essential services, reducing the number of ports to be attacked
- Router configuration
  - Block outside coming packets that have source addresses from the internal network
  - Block packets to the outside that have source addresses from outside the internal network

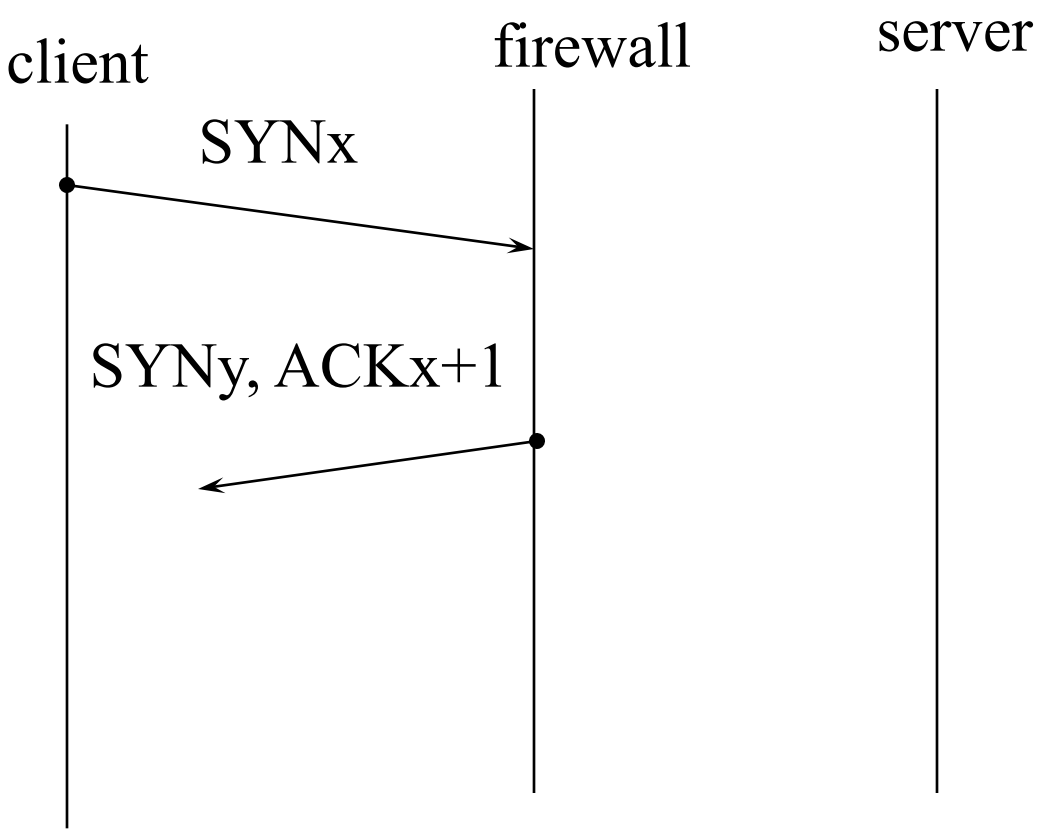
Infrastructure improvements

- If addresses prefixes separate clear the inside from the outside, then router configuration can be improved.
- Example: routers that attach an organization or an ISP to a backbone network.

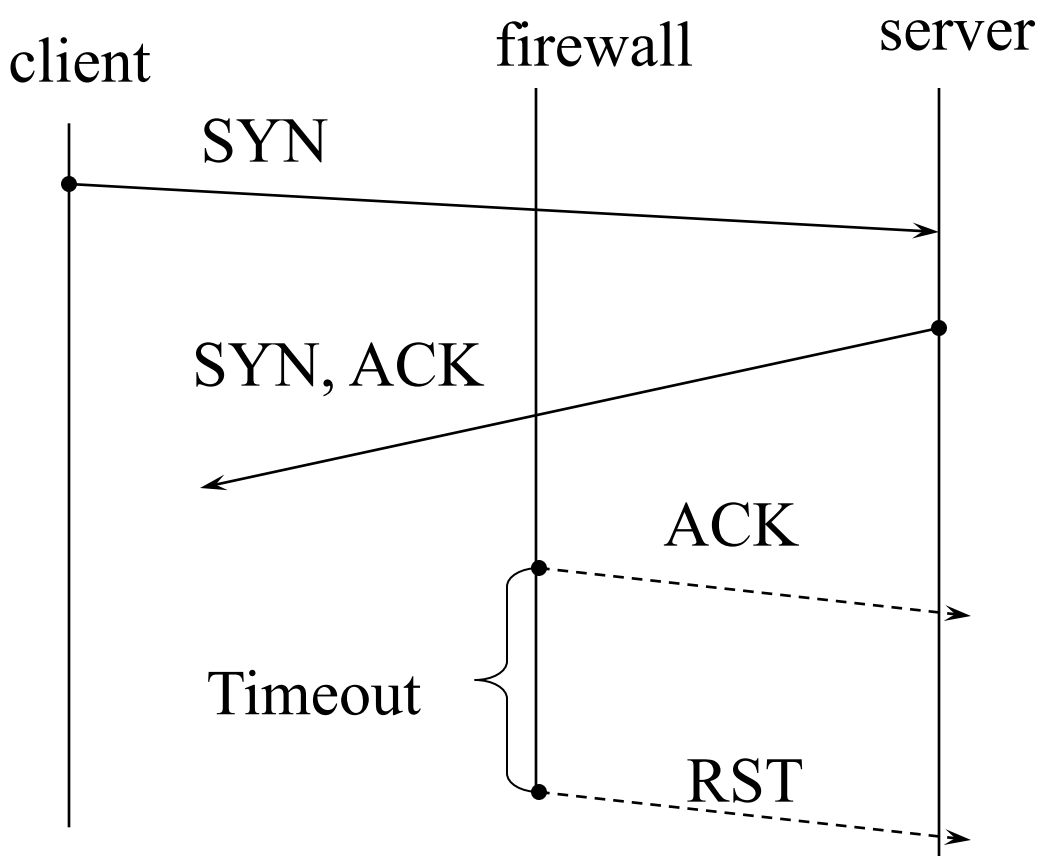
Firewall approach

- Main idea: each packet for inside network if first examined by the firewall
- Additional delays
- Two approaches:
  - Firewall as a relay
  - Firewall as a gateway

Firewall as a relay: Attack scenario



Firewall as a semi-transparent Gateway: attack scenario



Active monitoring

- Monitor the TCP traffic within a local area network and figure out which ones are illegitimate connections.
- Send RST for the illegitimate connections (this closes the connection).
- Does not require protocol stack modification.
- Monitor can be tricked to classify bad addresses as good addresses

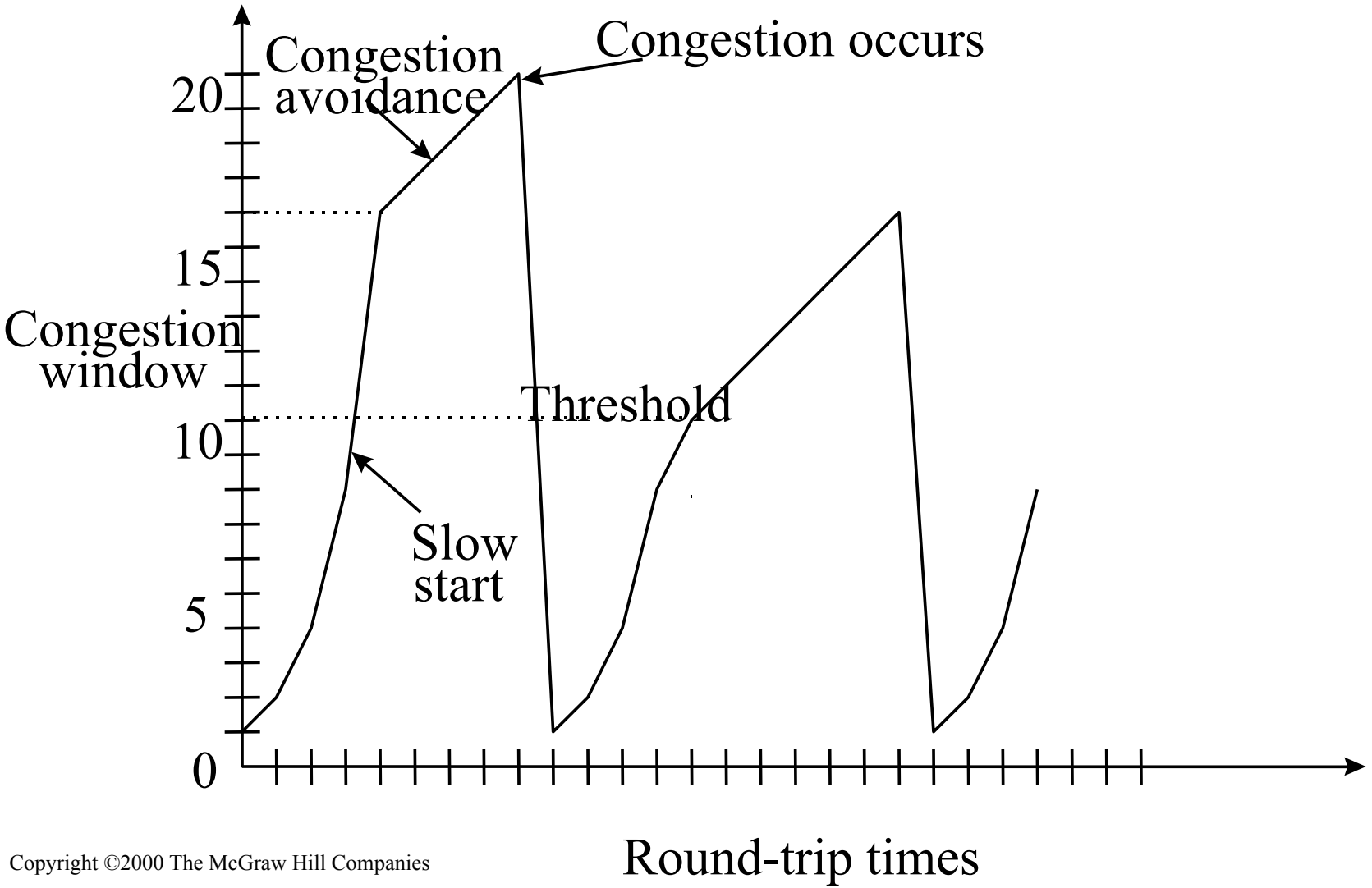
So far…

- Attacks require high-rate transmission (flood of SYN packets), unusual network traffic, attackers are relatively easy to detect and filter.
- However ….TCP can be attacked by using TCP friendly traffic (exploit congestion control mechanism), low rate, therefore it can cause significant damage without detection.

TCP congestion control

- Source determines how much bandwidth is available for it to send, it starts slow and increases the window of send packet based on ACKS.
- ACKS are also used to control the transmission of packets.
- Uses Additive Increase Multiplicative Decrease (AIMD)
- Uses Retransmission Timeout (RTO) to avoid congestion
- TCP Fairness: if k TCP sessions share same bottleneck link of bandwidth B, each should have average rate of B/k

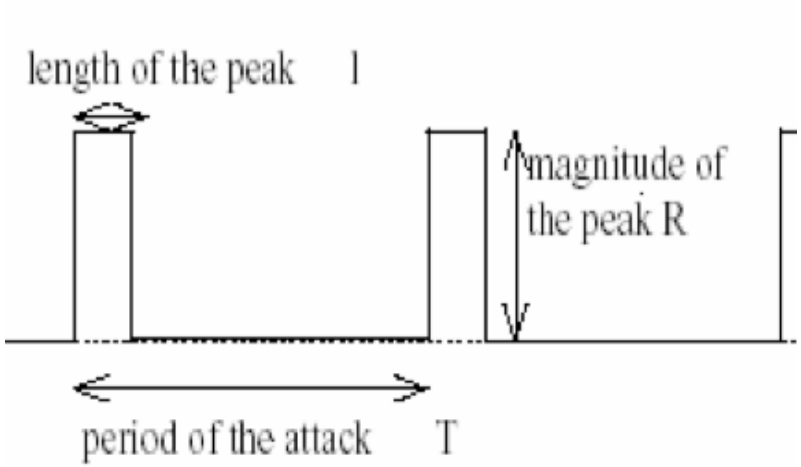
TCP congestion control



Leon-Garcia & Widjaja: *Communication Networks* Figure 7.63

The Attack

- All the attacker needs to do is generate a TCP flow to force the targeted TCP connection to repeatedly enter a retransmission timeout state
- Very effective, TCP throughput degrades significantly
- Sending high-rate, RTT scale short duration bursts and repeating periodically at RTO scale period.



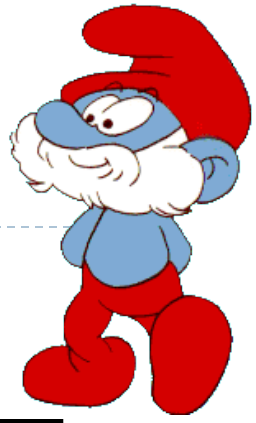
Basis of the Attack

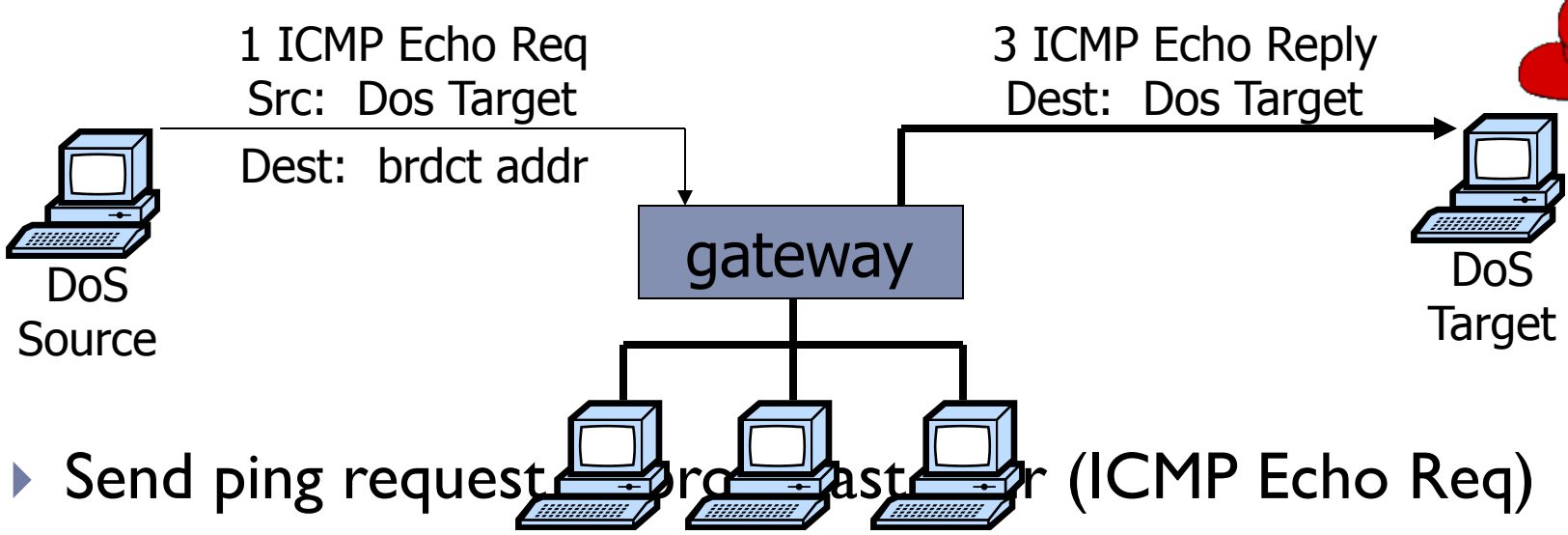
- Protocol is homogenous and deterministic
  - protocols react in a pre-defined way
  - tradeoff of vulnerability vs. predictability
- Periodic outages synchronize TCP flow states and deny their service
- Slow time scale protocol mechanisms enable low-rate attacks
  - outages at RTO scale, pulses at RTT scale imply low average rate

Proposed Solutions

- Factors: randomization, connectivity, accountability
- Router-Assisted Mechanisms: Routers identify and regulate the misbehaving flows
  - Router-Based algorithms
  - Random early detection with preferential dropping (queue management)
- End-point minRTO Randomization
- They mitigate the attack, but can not eliminate it

Smurf DoS Attack





- Send ping request to broadcast addr (ICMP Echo Req)
- Lots of responses:
  - Every host on target network generates a ping reply (ICMP Echo Reply) to victim
  - Ping reply stream can overload victim

Prevention: reject external packets to broadcast address

Internet Control Message Protocol

- Provides feedback about network operation
  - Error reporting
  - Reachability testing
  - Congestion Control
- Example message types
  - Destination unreachable
  - Time-to-live exceeded
  - Parameter problem
  - Redirect to better gateway
  - Echo/echo reply - reachability test

Distributed DoS (DDoS)

Hiding DDoS Attacks

- Reflection
  - Find big sites with lots of resources, send packets with spoofed source address, response to victim
    - PING => PING response
    - SYN => SYN-ACK
- Pulsing zombie floods
  - each zombie active briefly, then goes dormant;
  - zombies taking turns attacking
  - making tracing difficult

Cryptographic network protection

- Solutions above the transport layer
  - Examples: SSL and SSH
  - Protect against session hijacking and injected data
  - Do not protect against denial-of-service attacks caused by spoofed packets
- Solutions at network layer
  - Use cryptographically random ISNs [RFC 1948]
  - More generally: IPsec
  - Can protect against
    - session hijacking and injection of data.
    - denial-of-service attacks using session resets.

4: Protecting IP: IPSEC

IPSec Key management based on slides from B. LaMacchia

IPSec Overview

- Transparent to applications (below transport layer (TCP, UDP)
- Facilitate direct IP connectivity between sensitive hosts through untrusted networks
- Provides:
  - access control
  - integrity
  - data origin authentication
  - rejection of replayed packets
  - confidentiality
- IETF IPSEC Working Group
- Documented in RFCs and Internet drafts

Security Mechanism

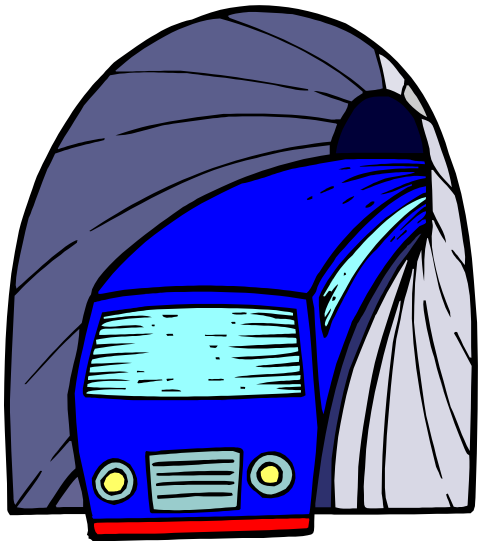
- Authentication Header (AH): provides integrity and authentication without confidentiality
- Encapsulating Security Payload (ESP): provides confidentiality and can also provide integrity and authentication
- Operates based on security associations
- Transport-mode: encapsulates an upper-layer
- protocol (e.g. TCP or UDP) and prepends an IP header in clear
- Tunnel-mode: encapsulates an entire IP datagram into new packet adding a new IP header

Transport Mode

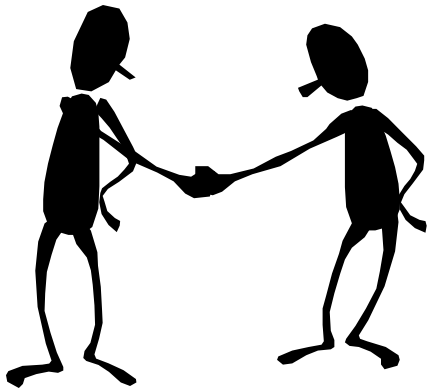
- ESP in Transport Mode: encrypts and optionally authenticates the IP payload (data), but not the IP header.
- AH in Transport Mode: authenticates the IP payload and selected portions of the IP header.

Tunnel Mode

- ESP in Tunnel Mode: encrypts and optionally authenticates the entire inner IP packet, including the inner IP header.
- AH in Tunnel Mode: authenticates the entire inner IP packet and selected portions of the outer IP header.



Security Associations (SA)



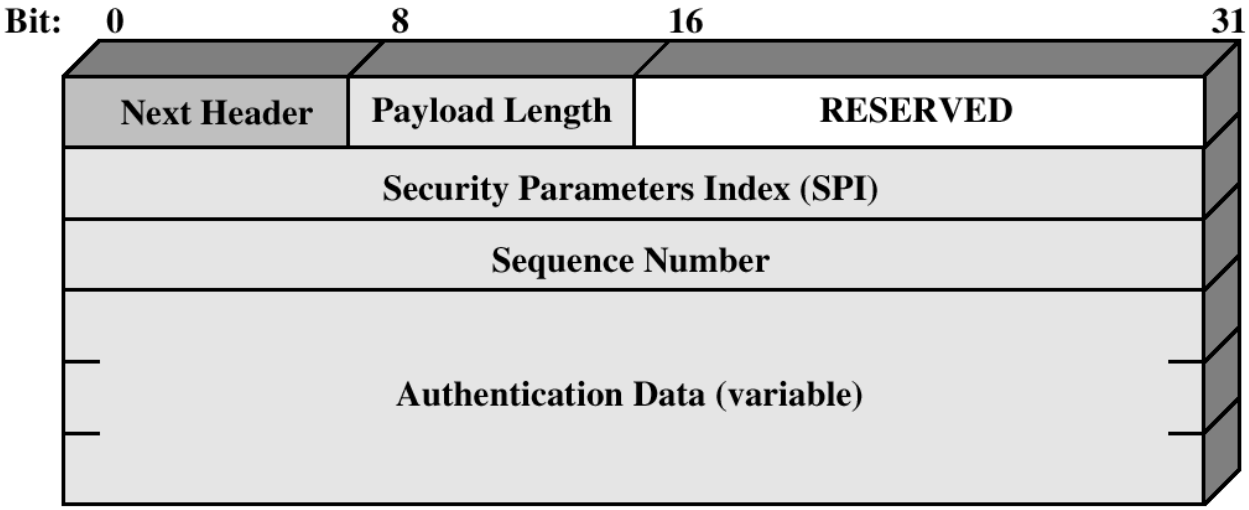
- A relationship between a
- sender and a receiver.
- Identified by three parameters:
  - Security Parameter Index (SPI)
  - IP Destination address (IP of the destination SA, can be a host, a firewall or a router)
  - Security Protocol Identifier (ESP or AH)
- SPI + IP destination address uniquely identifies a particular Security Association.
- SAs are unidirectional, sender supplies SPI to receiver.

Parameters of a Security Association

- Sequence number counter: used to generate a sequence number in AH and ESP headers
- Sequence counter overflow: how should sequence counter overflow be handled
- Anti-replay window: used to determine if an inbound AH or ESP packet is a replay
- AH information: auth. keys, key lifetime
- ESP information:encryption, auth., key, key lifetime, initial values
- Lifetime of the Security Association
- Protocol Mode: tunnel, transport

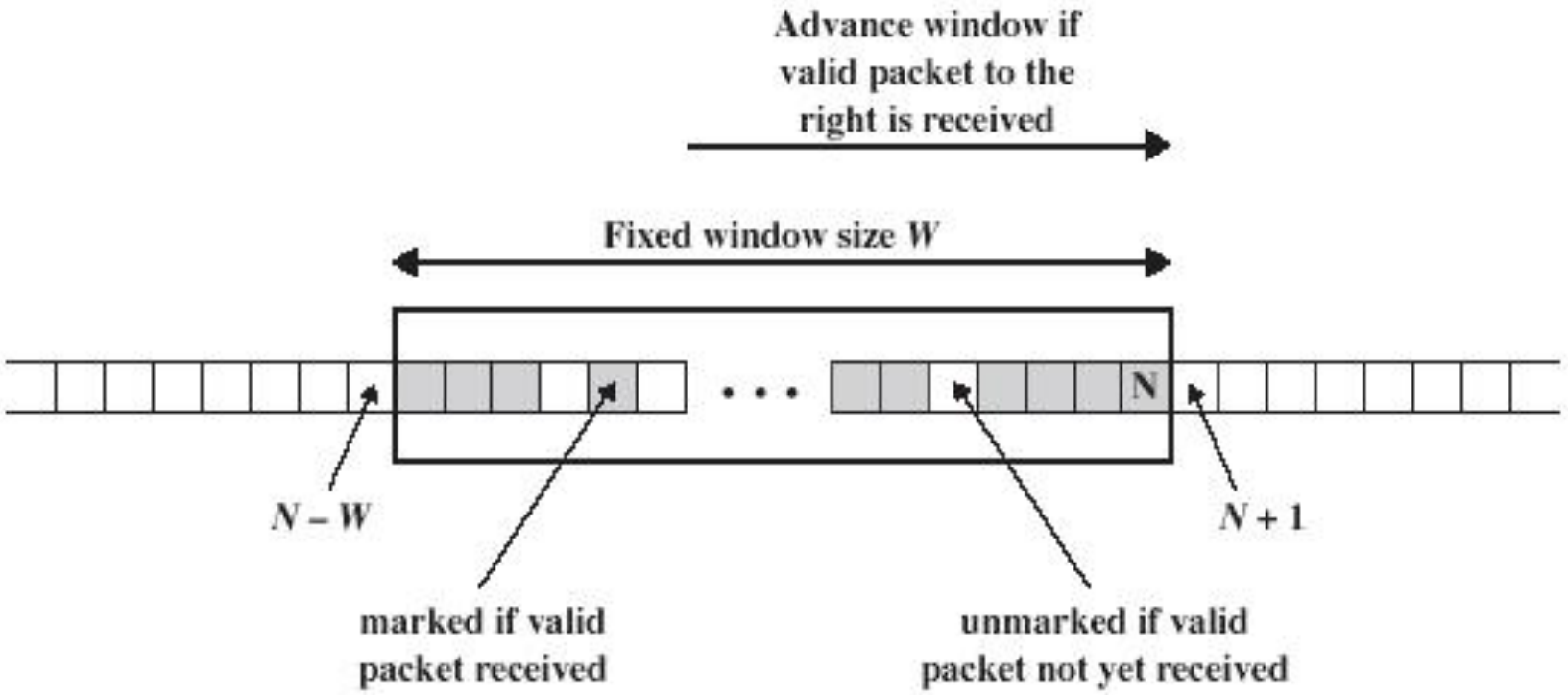
Authentication Header

- Provides support for data integrity and authentication (MAC) of IP packets, using HMAC based on MD5 or SHA1.
- Defends against replay attacks (sequence number).

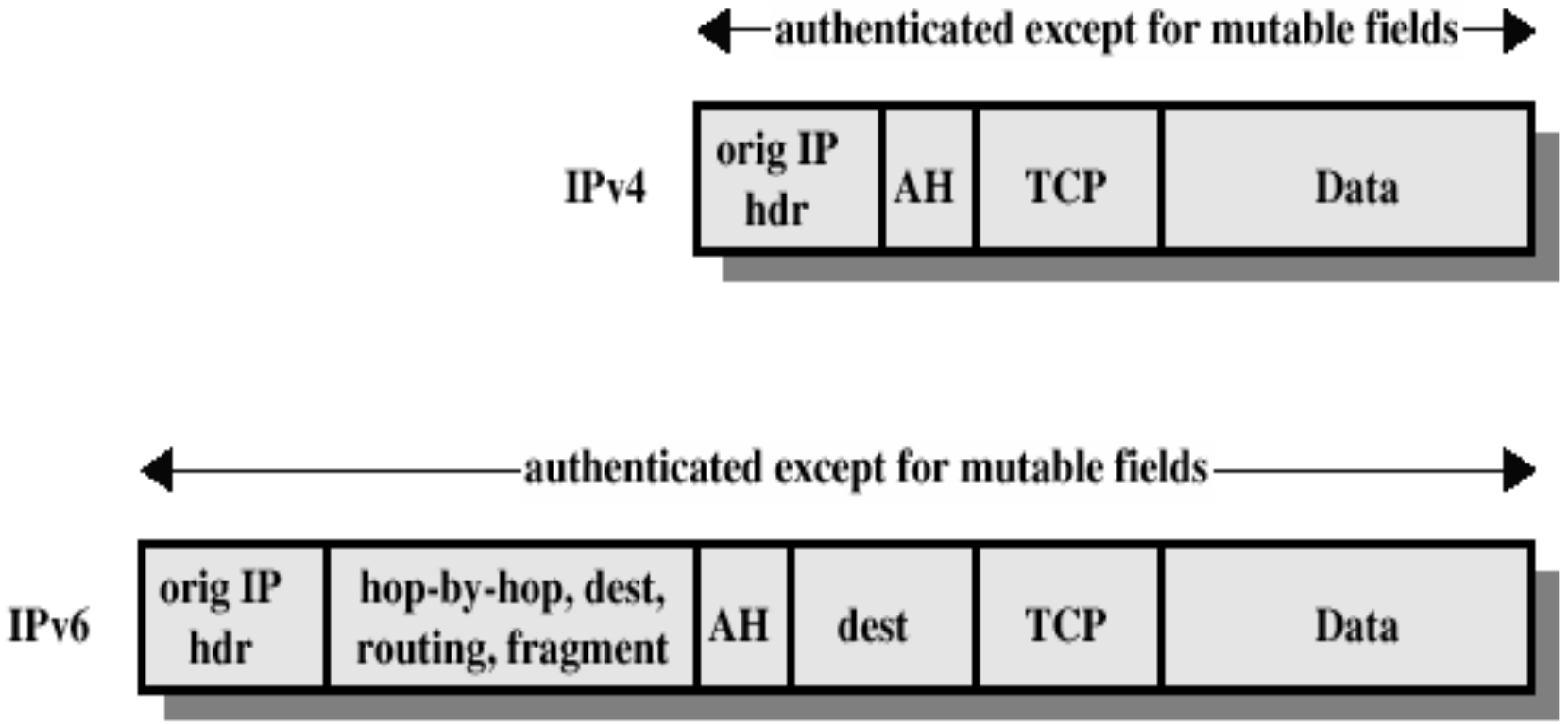


AH: Preventing Replay

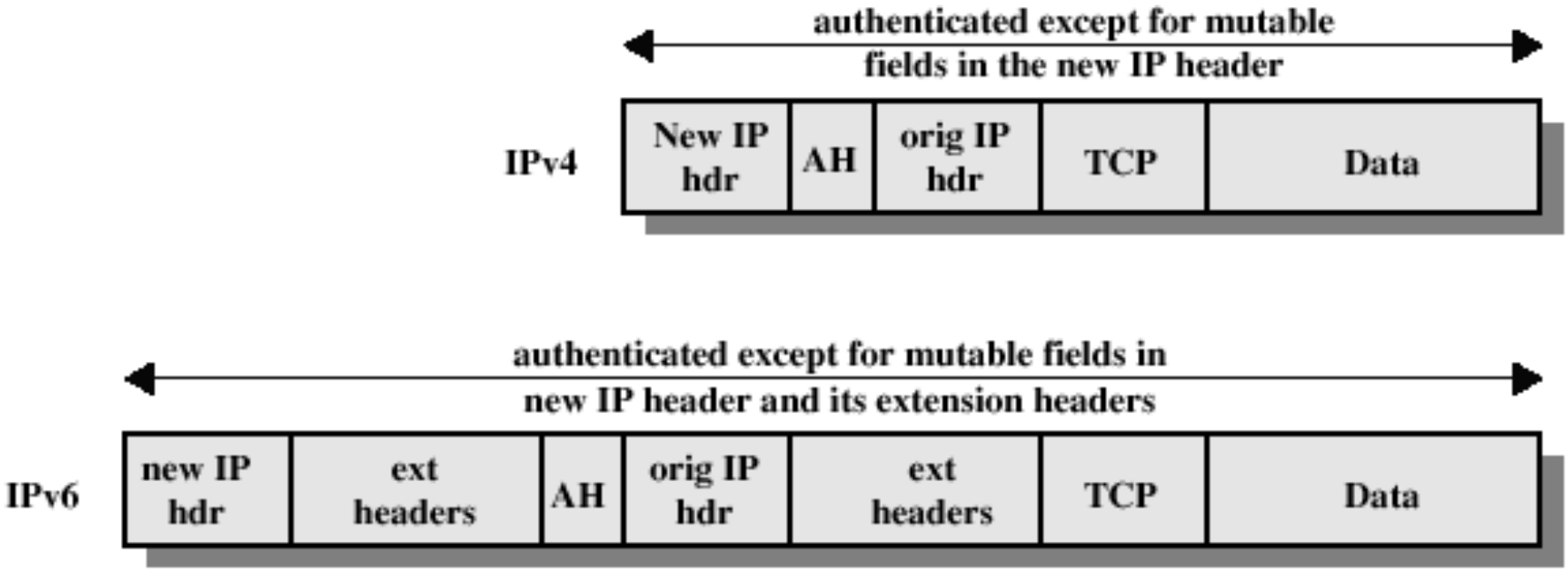
- When a SA is established, sender initializes sequence counter to 0.
- Every time a packet is sent the counter is incremented and is set in the sequnce number in the AH header.



AH Authentication: Transport Mode



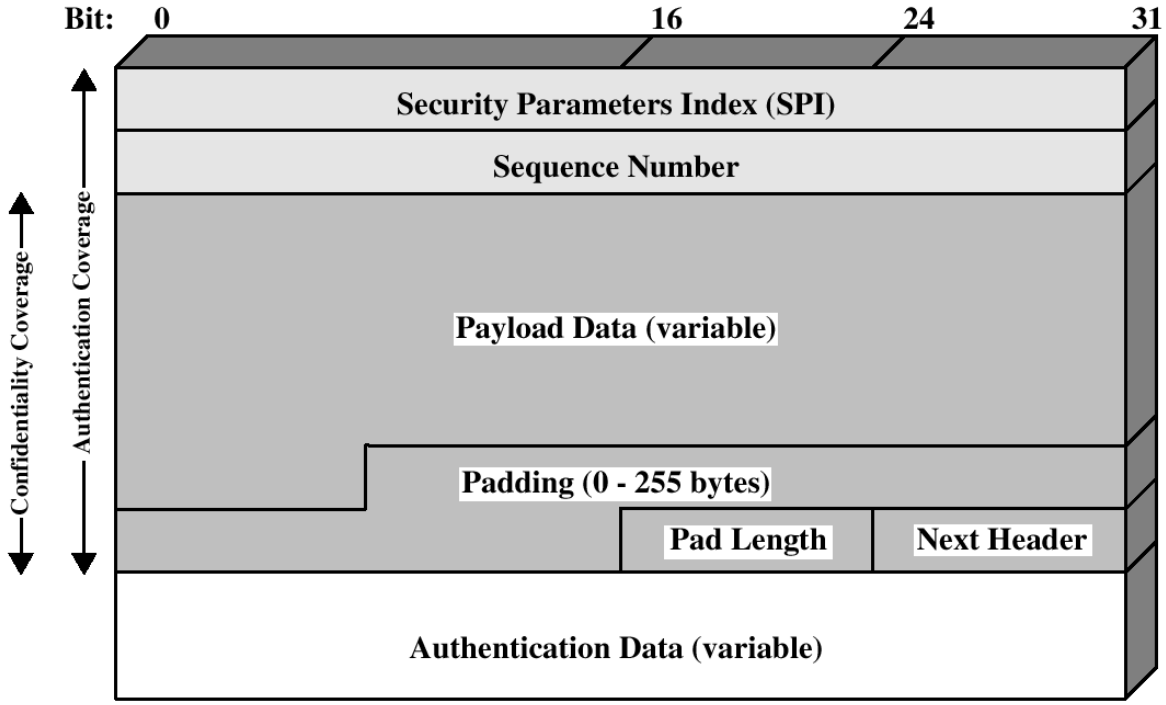
AH Authentication: Tunnel Mode



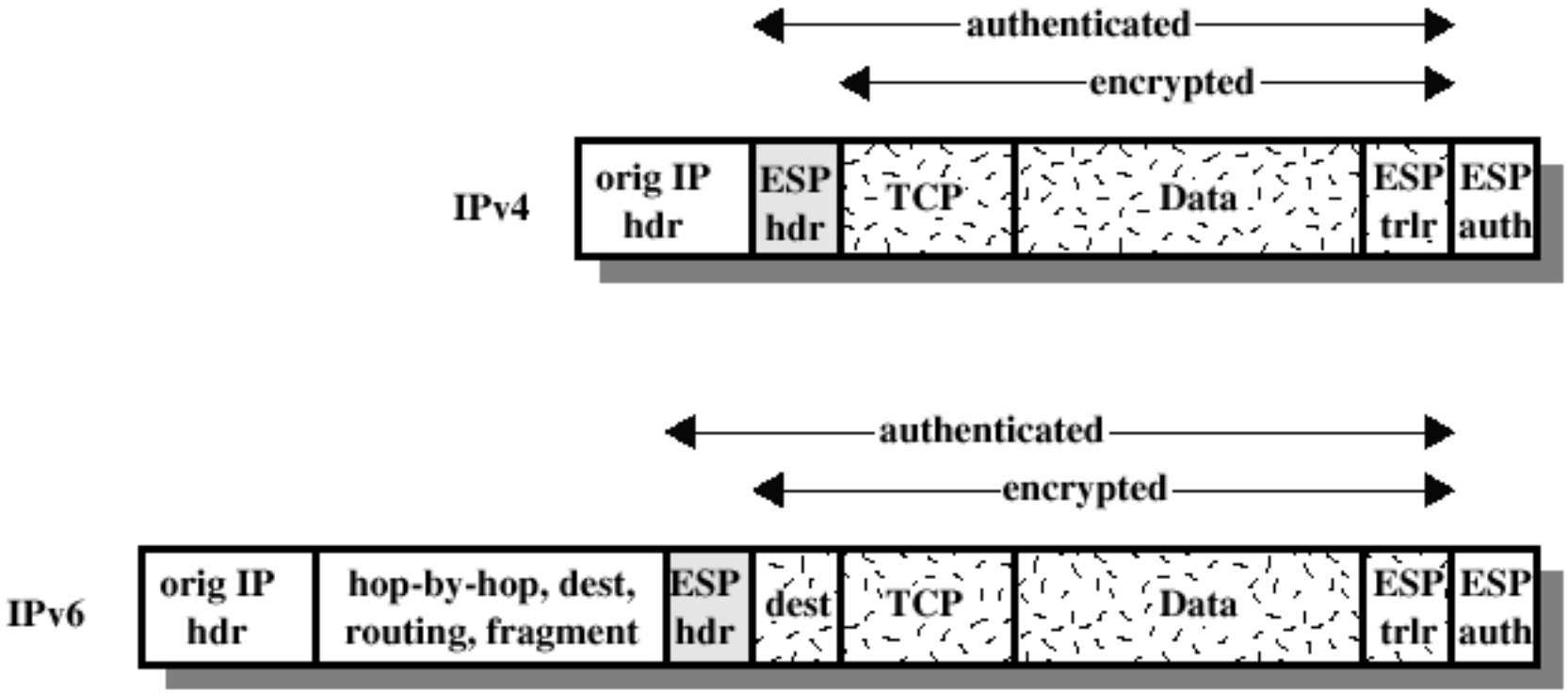
The new IP header contains different IP addresses than the ultimate destination and source

Encapsulating Security Payload

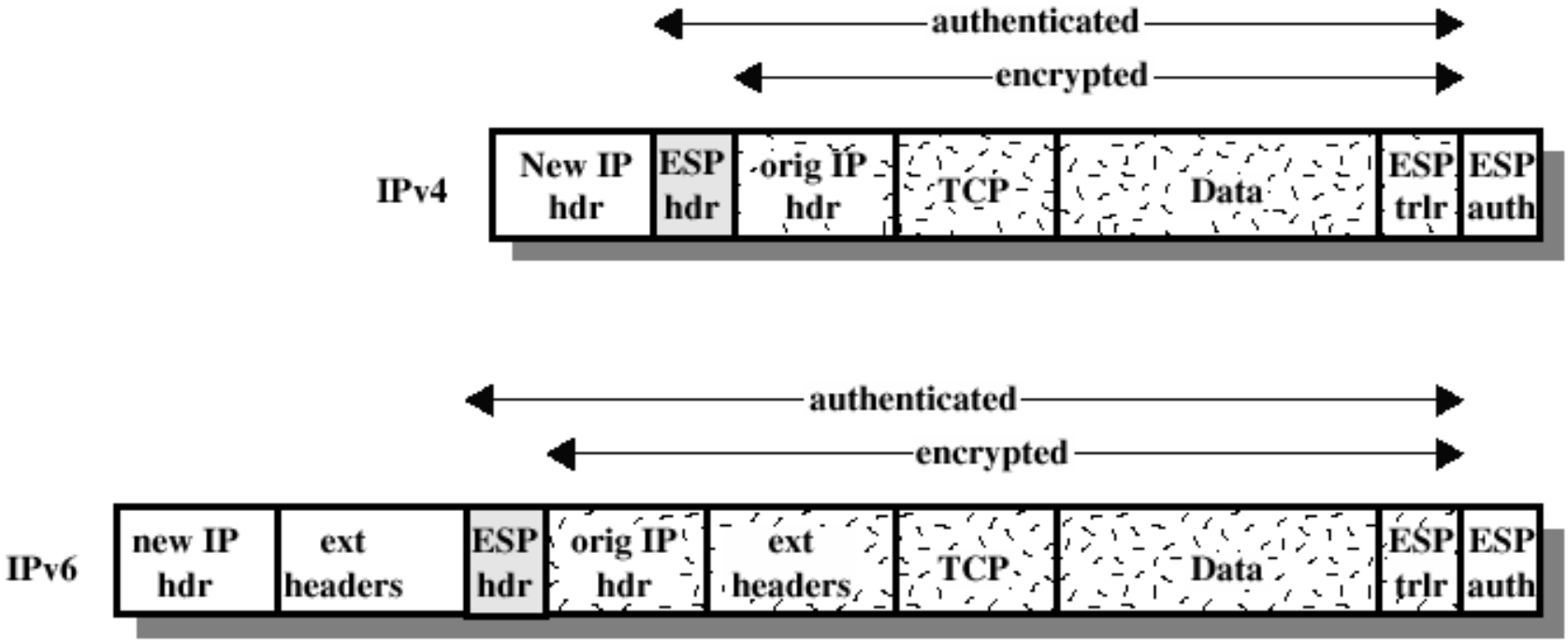
- ESP provides confidentiality services, optionally can provide the same services as AH
- Encryption: 3DES, Blowfish, CAST, IDEA, 3IDEA



ESP Encryption and Authentication: Transport Mode



ESP Encryption and Authentication: Tunnel Mode



Cryptographic Algorithms

- Encryption:
  - TripleDES-CBC
  - AES-CBC
  - AES-GCM: authentication and confidentiality
- Authentication:

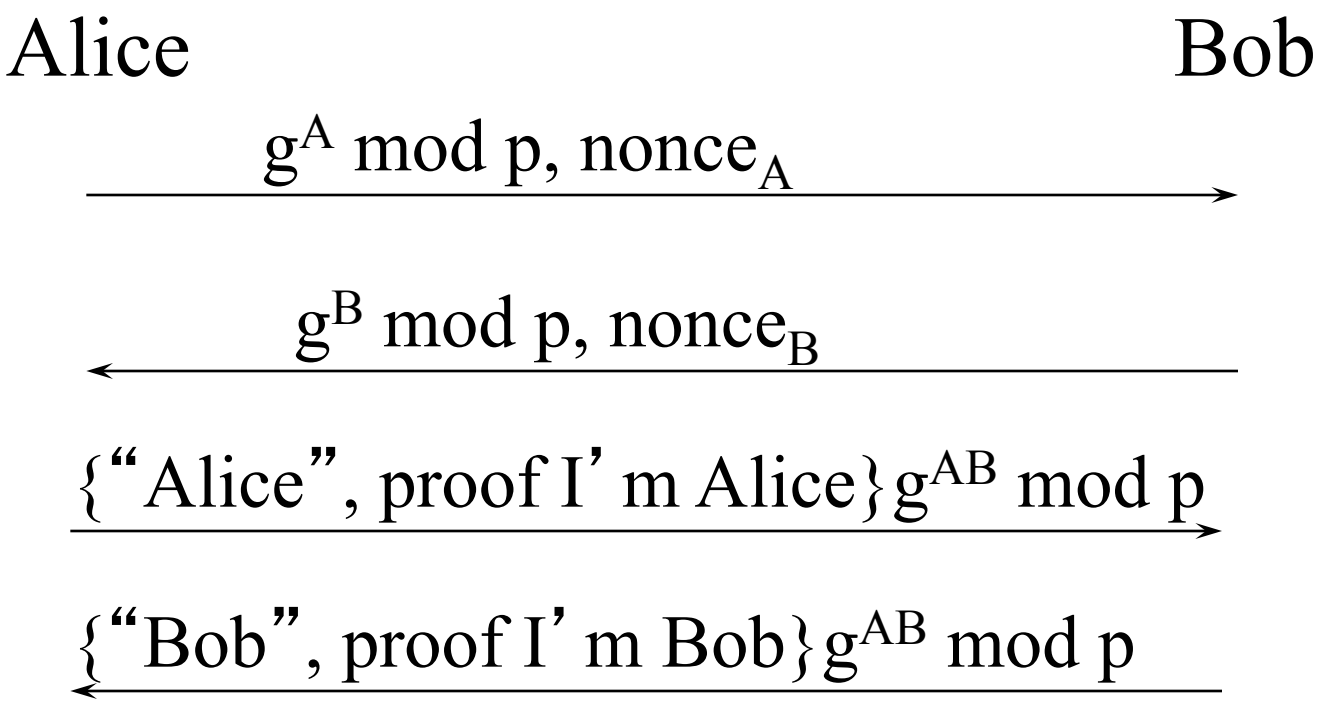
  HMAC-SHA1

  http://tools.ietf.org/html/rfc7321, August 2014

IPSec Key Management

- Goal: setting up SA and keys between pairs of communicating hosts
- Manual: system adminstrator configures the keys for hosts
- Automated: on-demand creation of keys
  - Oakley Key Determination Protocol (based on Diffie-Hellman): authenticated, prevents replays, negociates global parameters
  - Internet Security Association and Key Management Protocol (ISAKMP): Internet key management and negociation, defines procedures and packet formats to establish, negotiate, update, and destroy SAs
  - IKE: Resynchronize two ends of an IPsec SA: Choose cryptographic keys; Reset sequence numbers to zero; Authenticate endpoints

General idea of IKEv2



IKE Contenders

- Photuris: Signed Diffie-Hellman, stateless cookies, optional hiding endpoint IDs
- SKIP: Diffie-Hellman public keys, so if you know someone's public key g^B, you automatically know a shared secret g^{AB}. Each msg starts with per-msg key S encrypted with g^{AB}
- ISAKMP: "framework", not a protocol. Complex encodings. Flexible yet constraining.
  - Phase 1 expensive, establishes a session key with which to negotiate multiple phase 2 sessions

Internet Key Exchange (IKE)

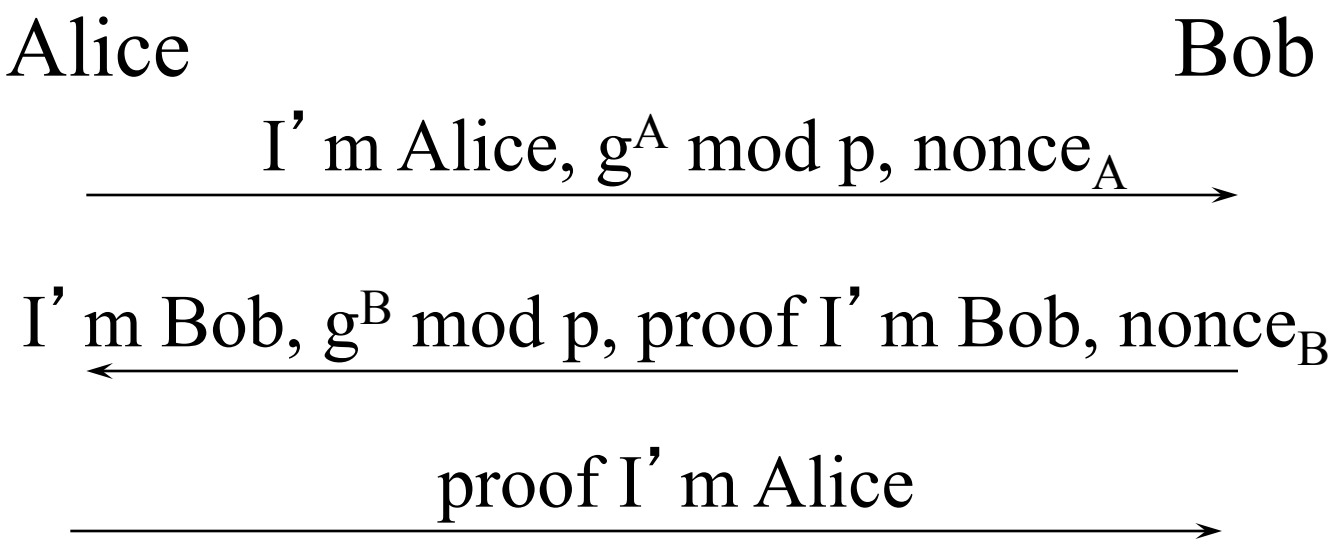
- Phase I
  - Establish a secure channel (ISAKMP SA)
  - Authenticate computer identity
- Phase II
  - Establishes a secure channel between computers intended for the transmission of data (IPSEC SA)

http://tools.ietf.org/html/rfc2409

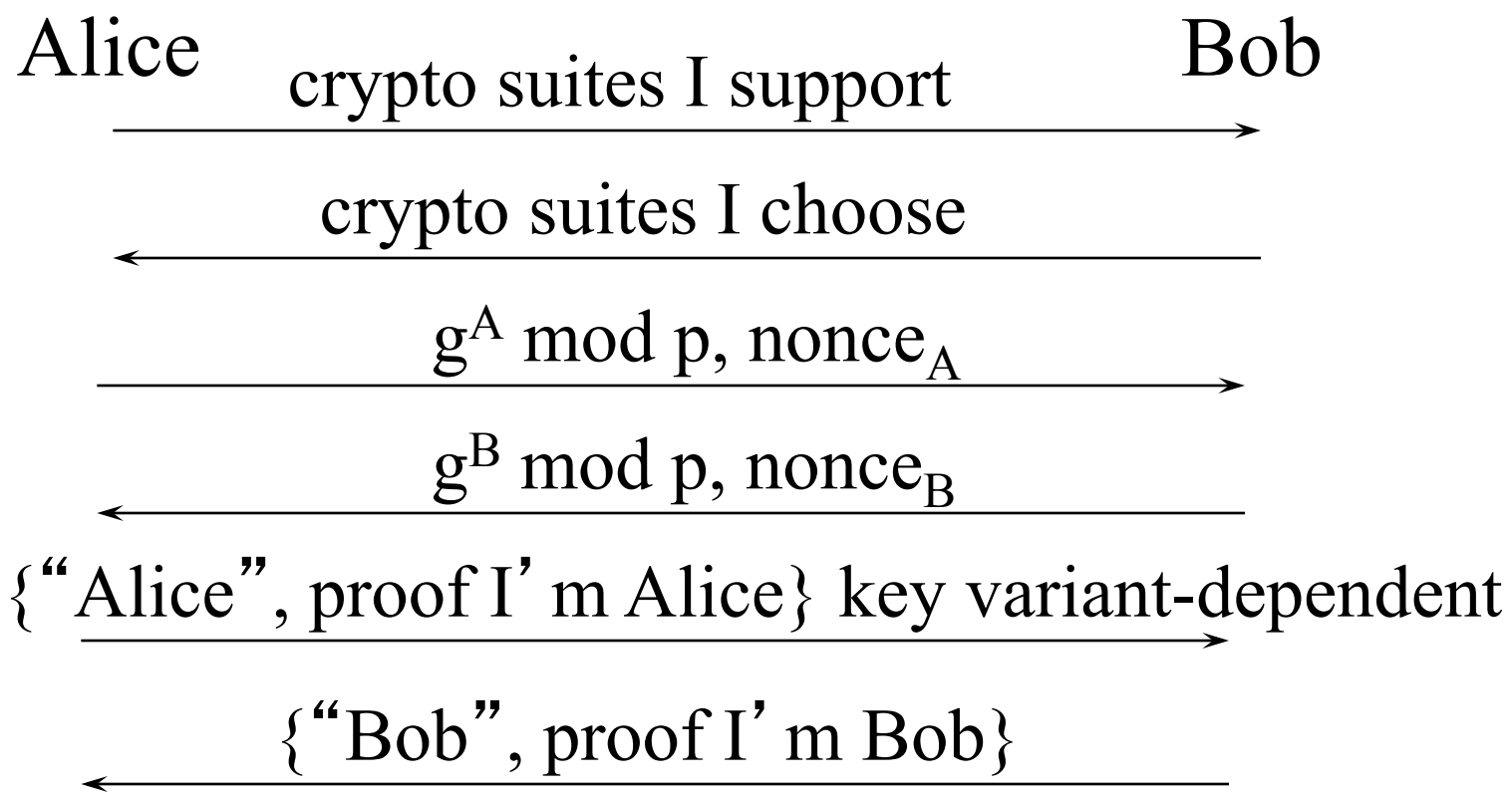
IKE

- IKEv1 authors tried to fit academic papers (SKEME, OAKLEY) into ISAKMP
- Mostly a rewriting of ISAKMP, but not self-contained. Uses ISAKMP
- Two phases, like ISAKMP
- Phase 1 is 8 protocols!
  - Two "modes": aggressive (3 msgs), and main (6 msgs)
  - Main does more, like hiding endpoint identifiers
- Phase 2 known as "quick mode"
- So 9 protocols (8 for phase 1, + phase 2)

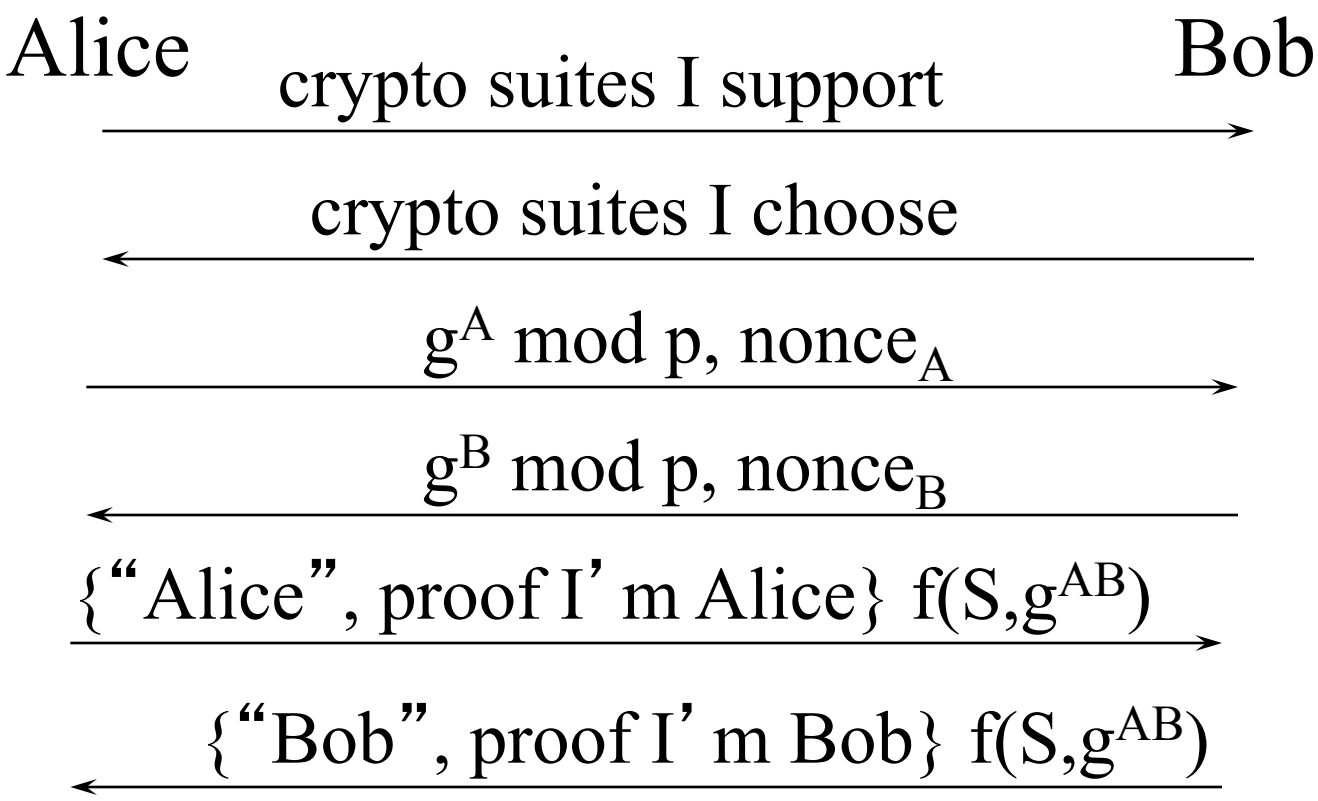
General Idea of Aggressive Mode



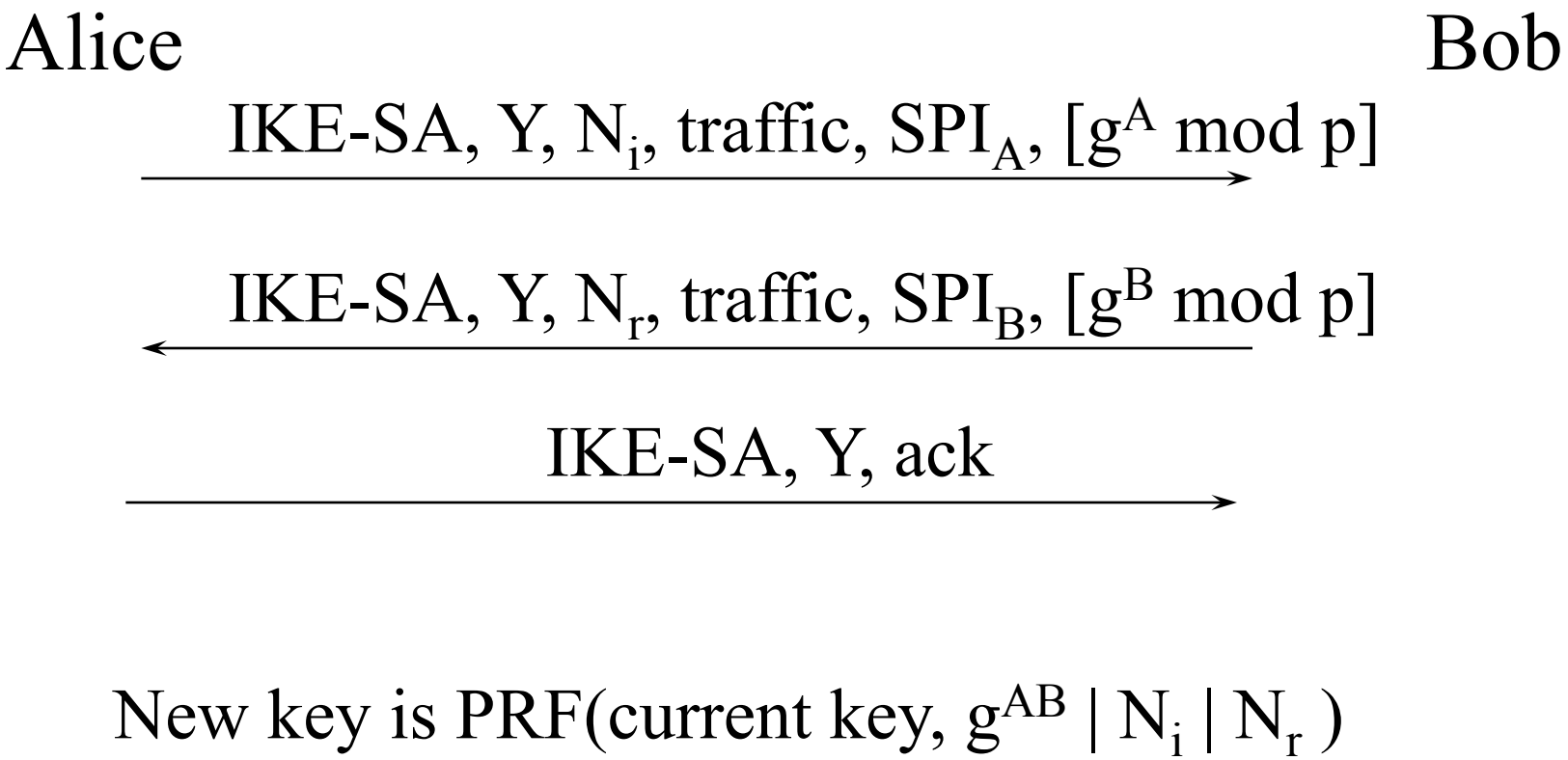
General Idea of Main Mode



Main-Mode-Preshared key S



General idea of Quick Mode



5: SSL/TLS

What is Transport Layer Security

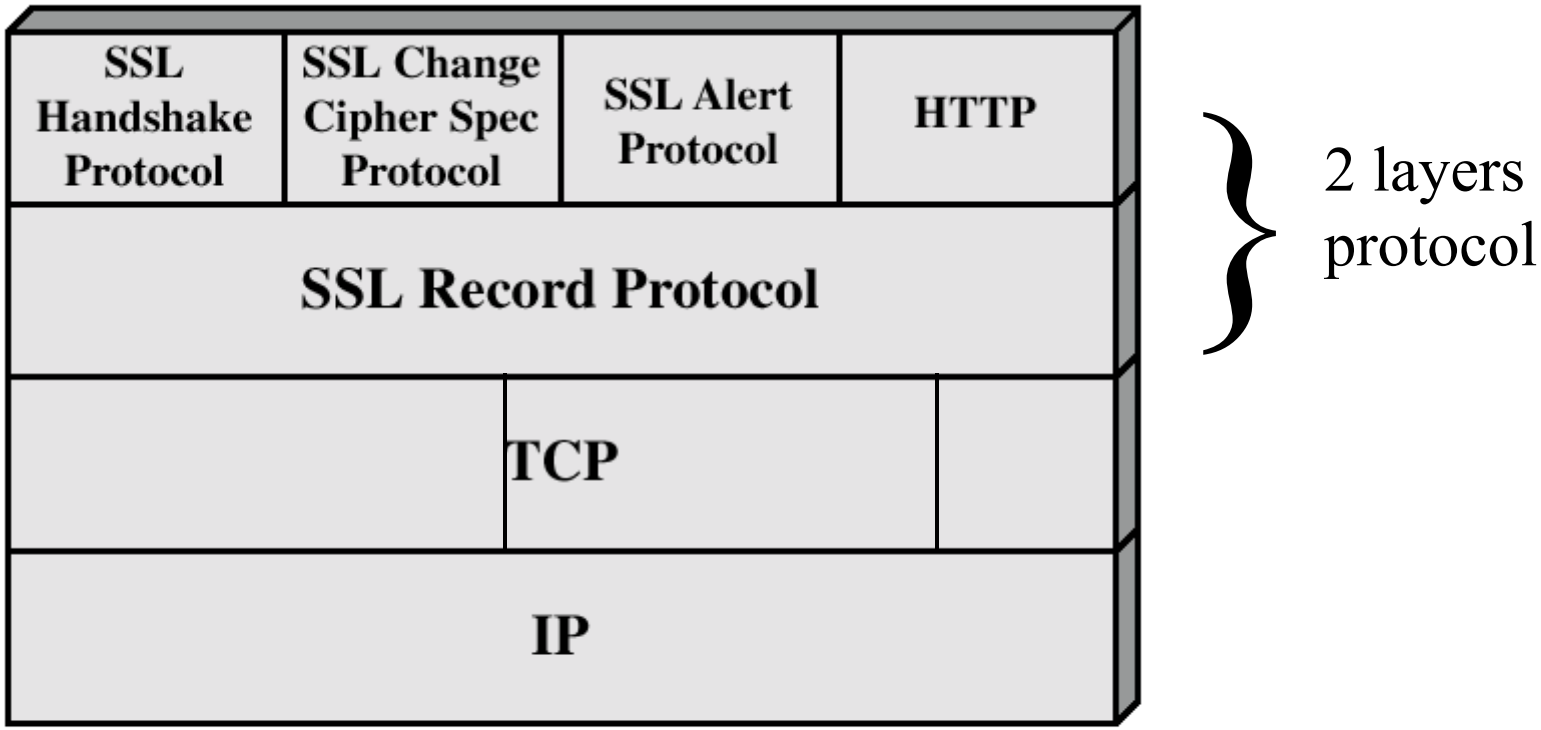
- Protocol that allows to establish an end-to-end secure channel, providing: confidentiality, integrity and authentication
- Defines how the characteristics of the channel are negotiated: key establishment, encryption cipher, authentication mechanism
- Requires reliable end-to-end protocol, so it runs on top of TCP
- It can be used by other session protocols (such as HTTPS)
- Several implementations: for example SSLeay, open source implementation (www.openssl.org)

TLS (cont.)

- **Confidentiality**: Achieved by encryption
- **Integrity**: Achieved by computing a MAC and send it with the message;
- **Key exchange**: relies on public key encryption for this.

- Several version algorithms changed with versions;
- TLS 1.2:
  - Replaced the use of MD5-SHA1 with SHA-256
  - AES, CCM and GCM modes
- TLS 1.3, draft
  - https://tools.ietf.org/html/draft-ietf-tls-rfc5246-bis-00

TLS: Protocol Architecture



Session and Connection

- **Session**:
  - association between a client and a server;
  - created by the Handshake Protocol;
  - defines secure cryptographic parameters that can be shared by multiple connections.
- **Connection**:
  - end-to-end reliable secure communication;
  - every connection is associated with a session.

Session

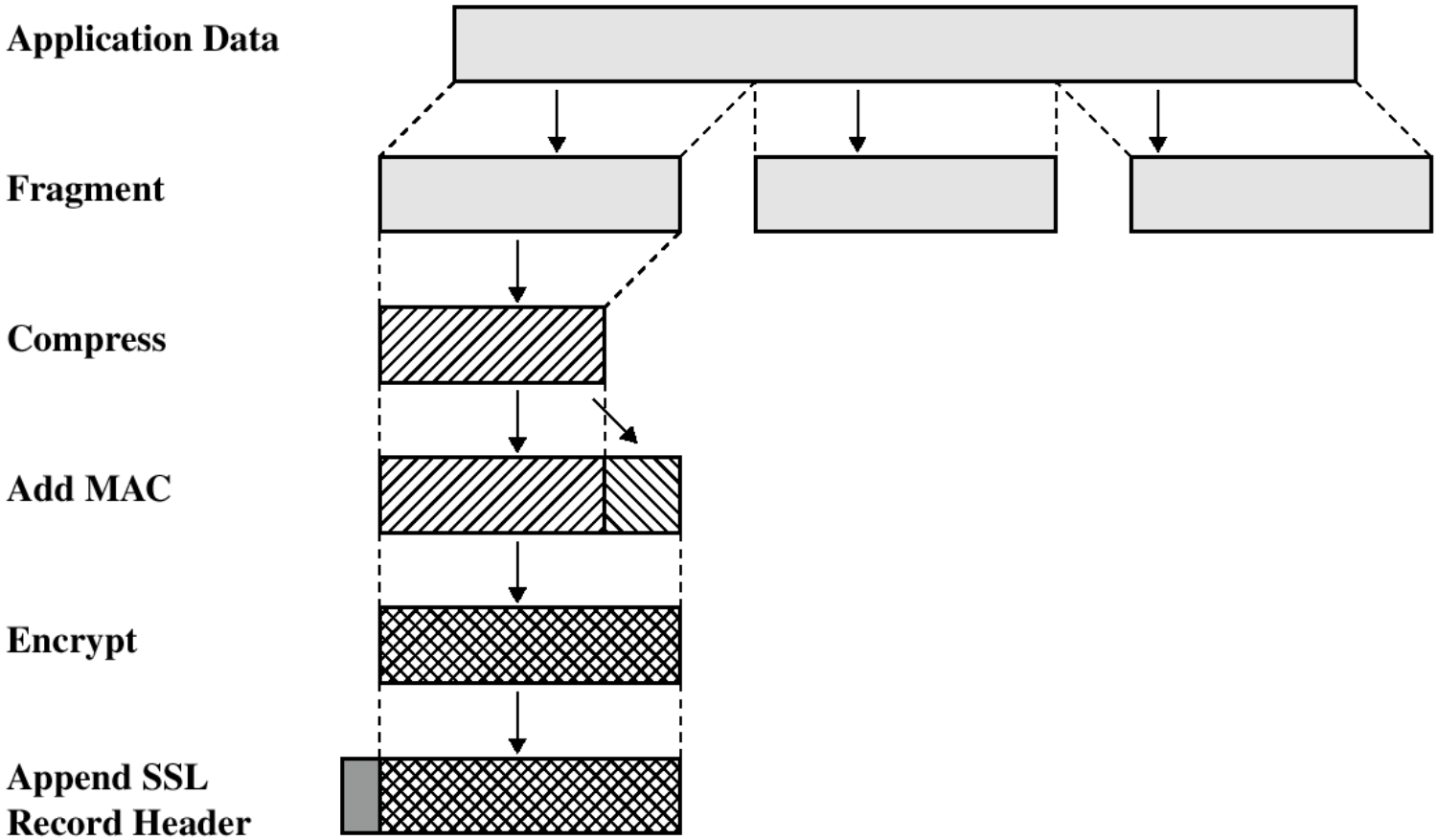
- Session identifier: generated by the server to identify an active or resumable session.
- Peer certificate: X 509v3 certificate.
- Compression method: algorithm used to compress the data before encryption.
- Cipher spec: encryption and hash algorithm, including hash size.
- Master secret: 48 byte secret shared between the client and server.
- Is resumable: indicates if the session can be used to initiate new connections.

Connection

- Server and client random: chosen for each connection.
- Server write MAC secret: shared key used to compute MAC on data sent by the server.
- Client write MAC secret: same as above for the client
- Server write key: shared key used by encryption when server sends data.
- Client write key: same as above for the client.
- Initialization vector: initialization vectors required by encryption.
- Sequence numbers: both server and client maintains such a counter to prevent replay, **cycle is $2^{64} - 1$**.

TLS: SSL Record Protocol

- Provides confidentiality and message integrity using shared keys established by the Handshake Protocol

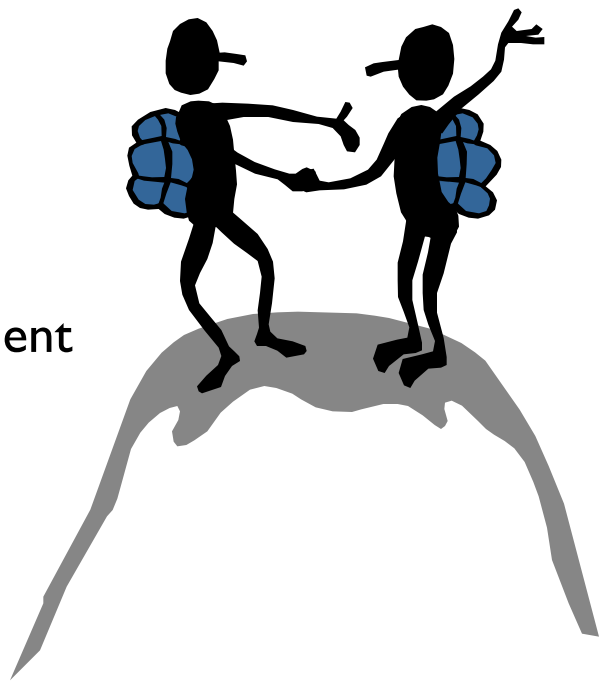


Alert Protocol

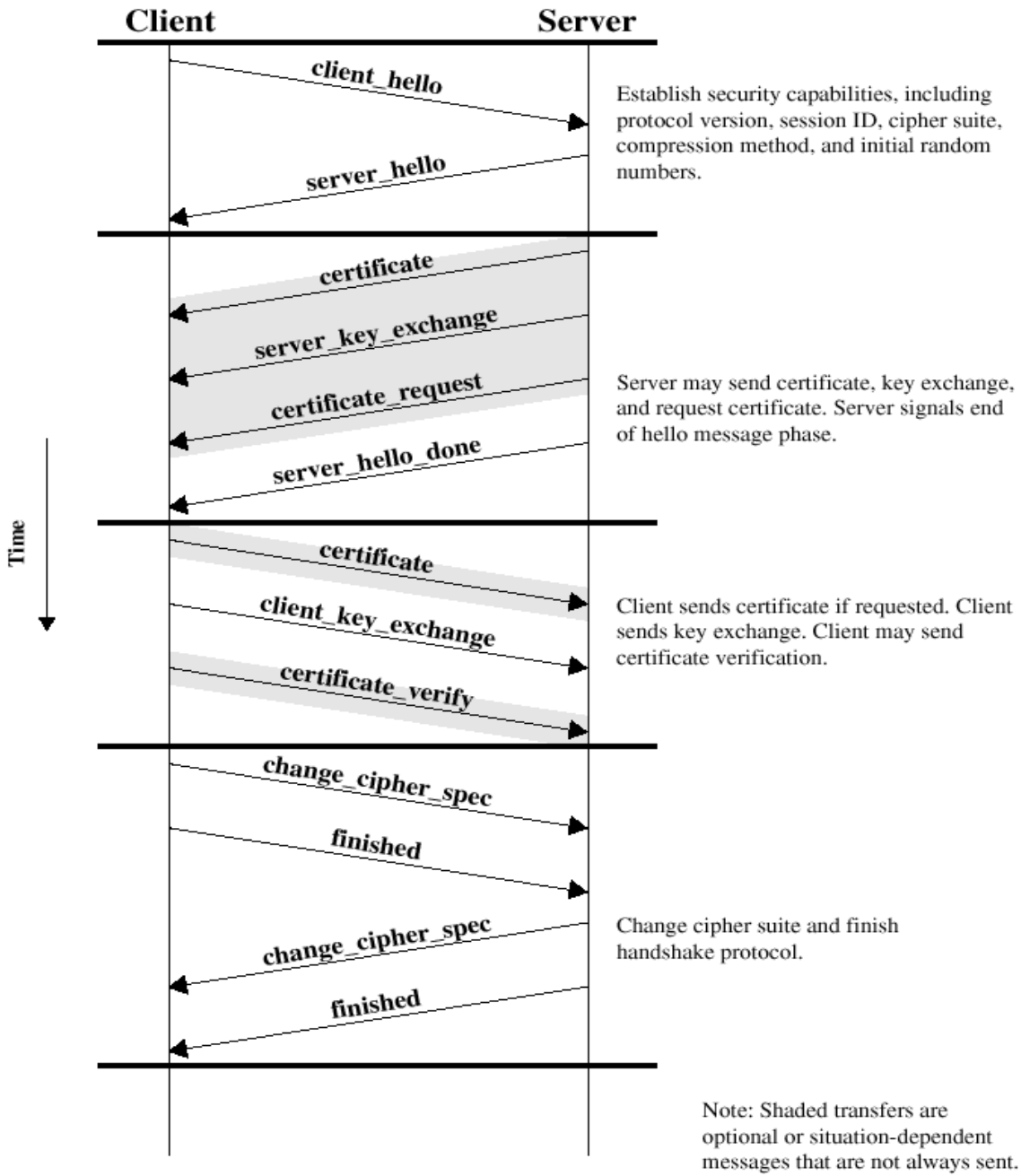
- Used to send TLS related alerts to peers
- **Alert messages are compressed and encrypted**
- Message: two bytes, one defines fatal/warnings, other defines the code of alert
- Fatal errors: decryption_failed, record_overflow, unknown_ca, access_denied, decode_error, export_restriction, protocol_version, insufficient_security, internal_error
- Other errors: decrypt_error, user_cancelled, no_renegotiation

TLS: Handshake Protocol

- Negotiate Cipher-Suite Algorithms
  - Symmetric cipher to use
  - Key exchange method
  - Message digest function
- Establish the shared master secret
- Optionally authenticate server and/or client



Handshake Protocol

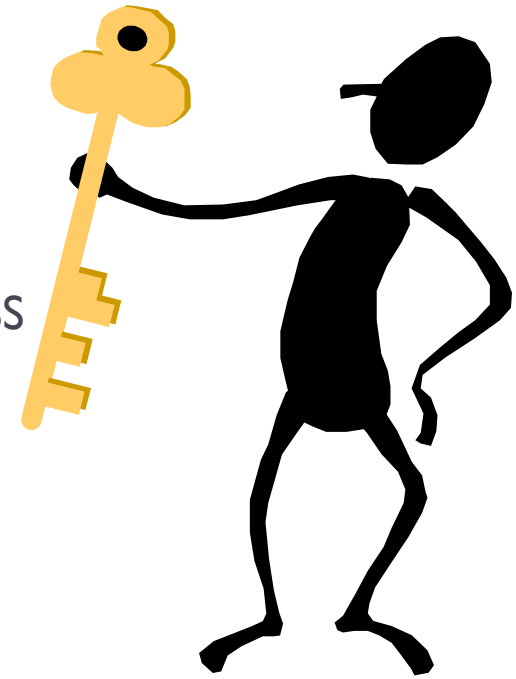


Handshake Protocol: Hello

- Client_hello_message has the following parameters:
  - Version
  - Random: timestamp + 28-bytes random
  - Session ID
  - CipherSuite: cipher algorithms supported by the client, first is key exchange
  - Compression method
- Server responds with the same
- Client may request use of cached session
  - Server chooses whether to accept or not

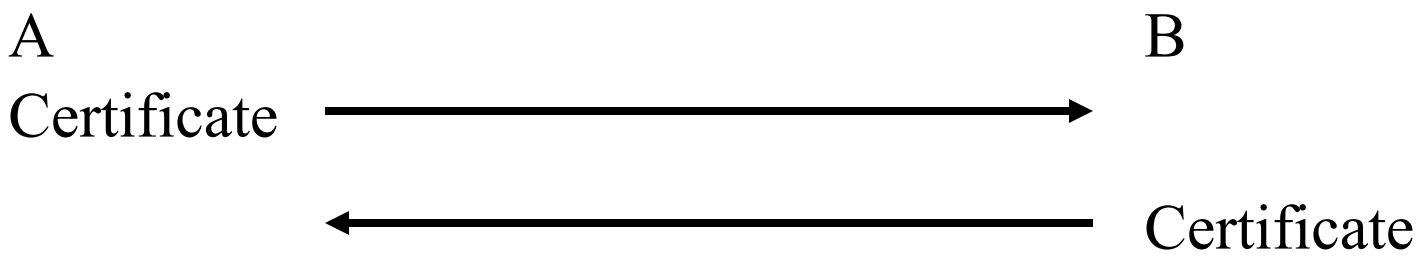
Handshake Protocol: Key Exchange

- Supported key exchange methods:
  - RSA: shared key encrypted with RSA public key
  - Fixed Diffie-Hellman; public parameters provided in a certificate
  - Ephemeral Diffie-Hellman: the best; Diffie-Hellman with temporary secret key, messages signed using RSA or DSS
  - Anonymous Diffie-Hellman: vulnerable to man-in-the-middle



TLS: Authentication

- Verify identities of participants
- Client authentication is optional
- Certificate is used to associate identity with public key and other attributes



TLS: Change Cipher Spec/Finished

- Change Cipher Spec completes the setup of the connections.
- Announce switch to negotiated algorithms and values
- The client sends a message under the new algorithms, allows verification of that the handshake was successful.

TLS vs. IPSEC

- Security goals are similar
- IPSec more flexible in services it provides, decouples authentication from encryption
- Different granularity: IPSec operates between hosts, TLS between processes
- Performance vs granularity

Cristina Nita-Rotaru

QUIC: Security Meets Performance

Slides by Samuel Jero

QUIC: Quick UDP Internet Connections

- New network protocol designed by Google
- Deployed in Chrome in 2013
- Encrypted, reliable, byte-stream protocol
  - Similar to TLS+TCP
- Goals:
  - Security similar to TLS
  - **Low latency--useful data in the first round trip**
  - Deployability

Low latency means users get data faster, which means happier users

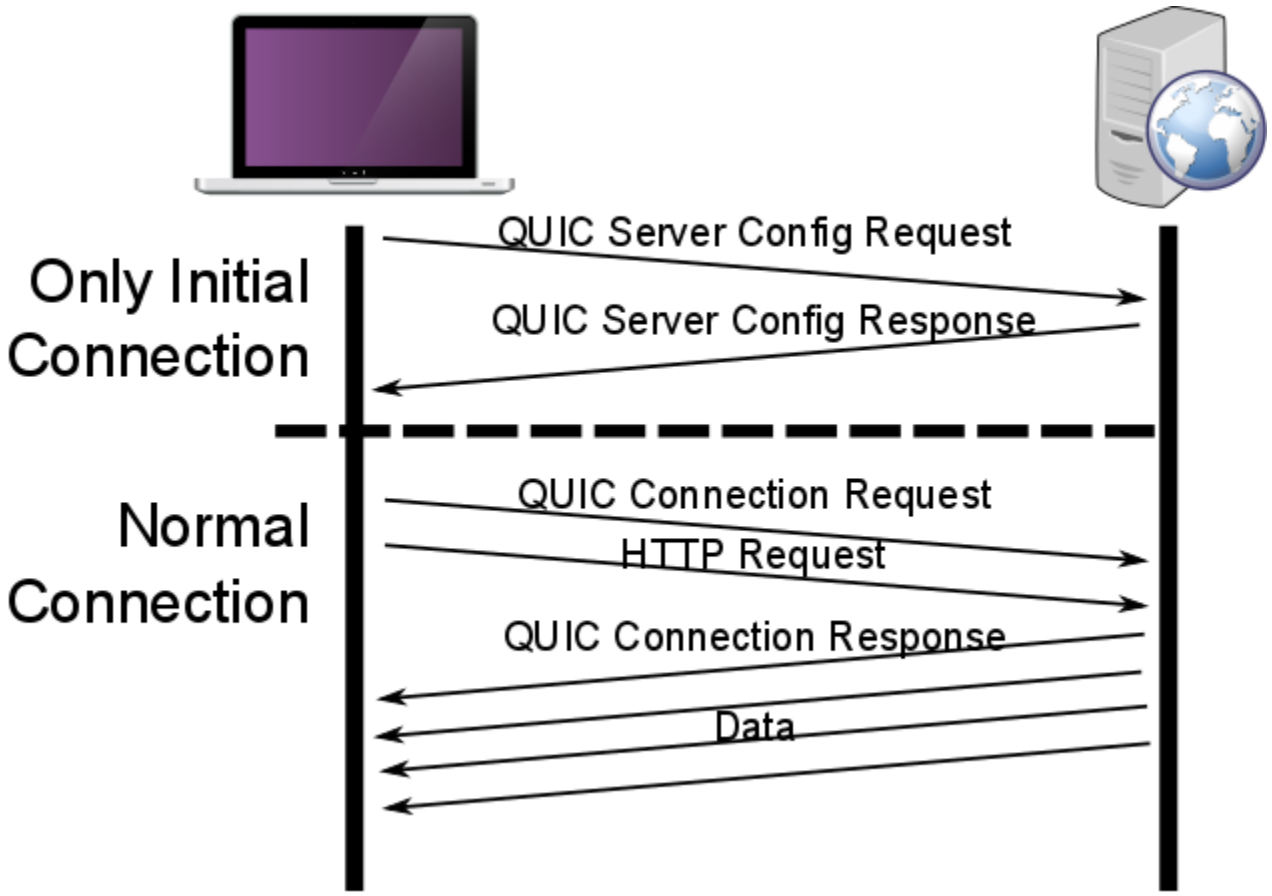
Low Latency

- QUIC is highly optimized for minimum connection establishment latency
- 0-RTT Connections---be able to send useful data in the first round trip
  - QUIC is built on top of UDP not TCP to eliminate the TCP handshake
  - QUIC makes a number of connection parameters cacheable
    - Encryption and key-exchange algorithms supported, Server's Diffie Hellman value, and an IP-bound token
    - Called the *server config*

QUIC Protocol

- On First Connection
  - Client requests the server config from the server
  - One round trip later, the client receives the server config
- Normal Connection
  - Client generates a connection request and computes encryption key using the server config
    - Server-side Diffie Hellman value is in server config
  - Client uses initial encryption key to send data
  - Server receives the connection request and sets up connection, computing the same initial encryption key
  - Server sends connection response containing new Diffie Hellman value to create a new, forward secure key
  - Server then sends data with this new key

QUIC Protocol



How Secure Is QUIC?

- QUIC is about 3 years old
- SSL/TLS is about 6 versions and 20 years old
- Existing security analyses of QUIC do not consider the protocol as actually specified or are formulated informally
- Solution: A provable security analysis

Provable Security:
A formal proof of a protocol under a specific security model specifying the security properties preserved, assumptions made, and the adversary's capabilities

Provable Security Analysis

- Existing provable security models cannot be reused for QUIC
  - Multiple encryption keys, lack of TCP, key exchange/data exchange overlap
- New Model: QACCE
  - Quick Authenticated and Confidential Channel Establishment
  - Designed for performance-optimized protocols like QUIC
  - Considers:
    - Confidentiality and authenticity of messages
    - Forward secrecy
    - Impersonation attacks
    - IP spoofing
- We proved QUIC secure under the QACCE model

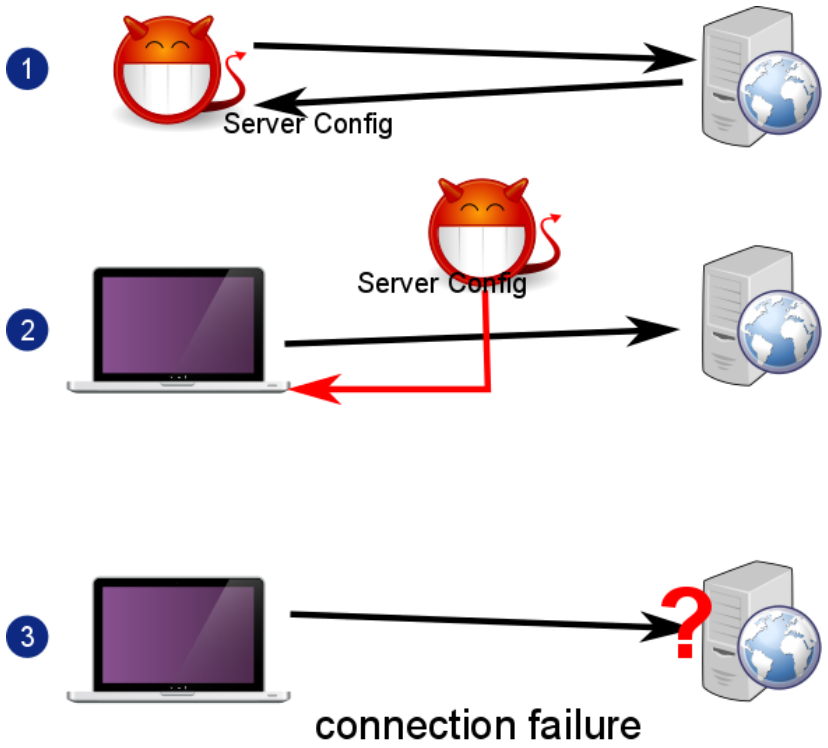
Performance Attacks

- QUIC is vulnerable to several attacks on its performance:
  - Client Denial of Service
    - Replay Attacks
    - Manipulation Attacks
  - Server Denial of Service
    - Replay Attacks

- These attacks do not compromise the security of QUIC
- Attacks on TLS's performance exist, it makes no performance claims

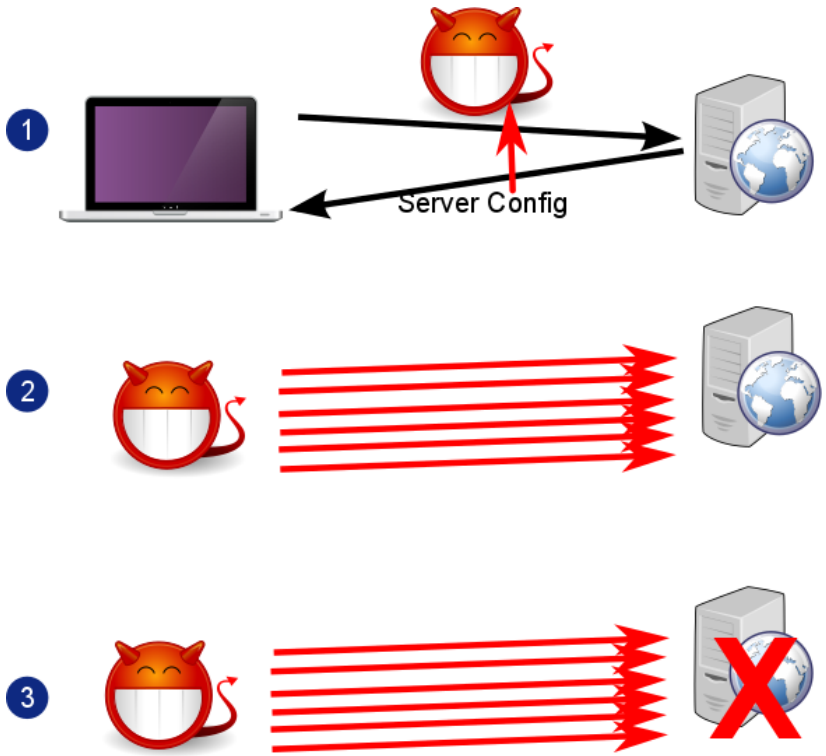
Replay Attacks (1)

- Attacker requests server config from server
- Attacker replays this config to an unsuspecting client
- Client connection fails due to bad IP token



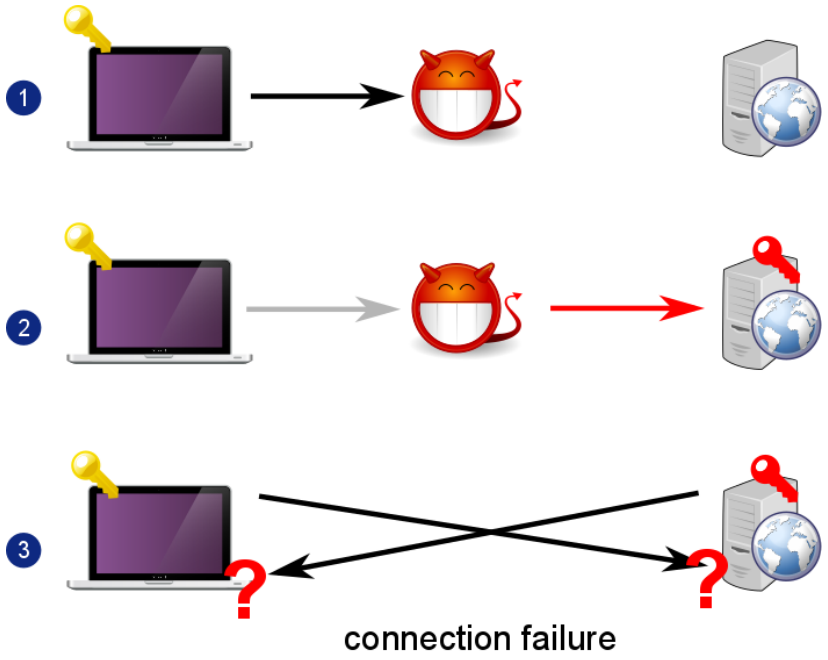
Replay Attacks (2)

- Attacker sniffs server config from legitimate connection
- Attacker forges many connection requests for this legitimate host
- Server becomes overloaded from keeping connection state



Manipulation Attacks

- Attacker is on the path between client and server
- Attacker modifies connection request
- Client and Server compute different keys causing connection failure
- Server continues to hold state for many minutes



Summary

- QUIC is a new protocol developed by Google
- Optimized for low latency using 0-RTT connections
  - Made possible by caching information about server
- We developed a provable security analysis showing that QUIC is secure
  - New model for performance-optimized protocols
  - Security guarantees comparable to TLS
- QUIC is vulnerable to several attacks on its performance
  - Resulting from its performance optimization
  - TLS is also vulnerable to attacks on its performance, but it makes no performance claims

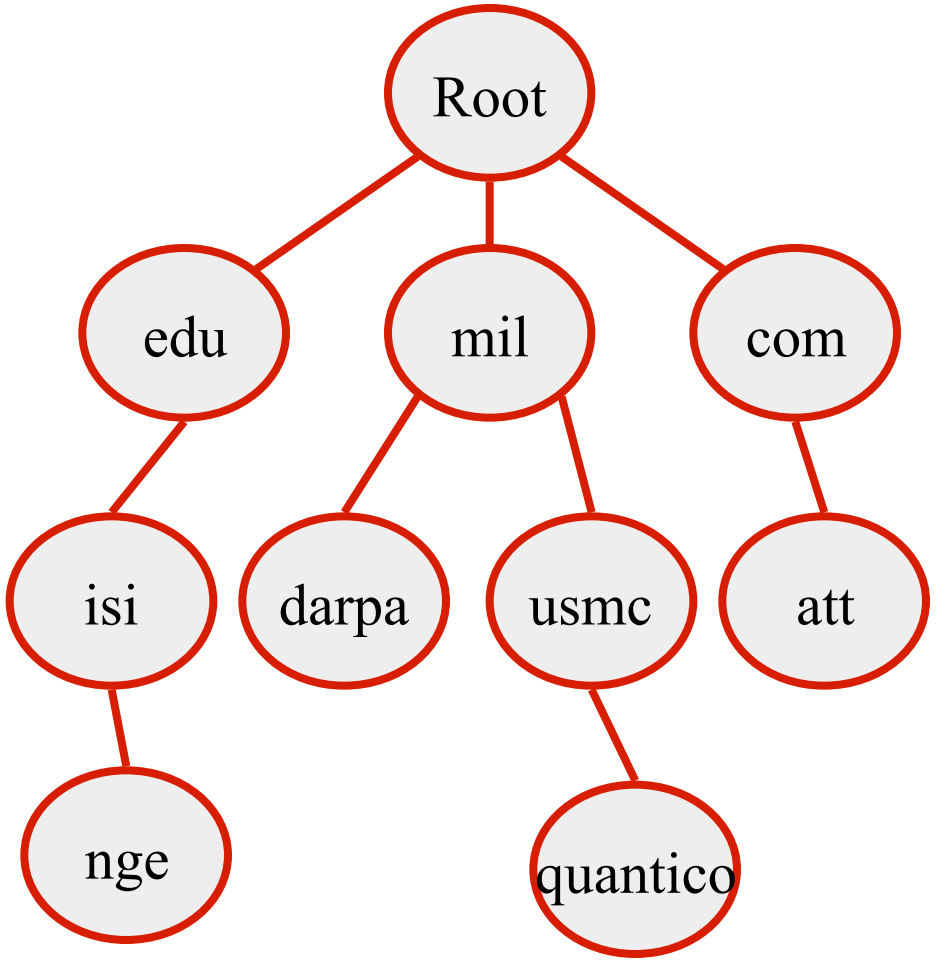
6: DNS Security

Domain Name System (DNS)

- People prefer names to identify computers instead of numbers
- DNS: Distributed, hierarchical database that maps host names with IP addresses
- Almost any application uses DNS
- If DNS is not working many applications will be crippled
- Uses UDP

DNS (cont.)

- Tree structure
  - Divided into zones
  - Delegating responsibilities
- ICANN oversees the domain name assignments
- Name servers
  - Authoritative information (hints to whom might be able to answer the request)
  - Cached data updated periodically



Domain Name System

Hierarchical Structure

- 13 root servers
- DNS root servers are in fact clusters of computers
- What kind of traffic? Oct 2002, 24 hours on f.root-servers.net root, 14GB, 152,744,325 queries, 1768 queries per second
- Top Level Domain (TLD) operate ".com", ".edu", etc
- Name servers

Name Server (NS)

- Resolver: client part that asks the questions about hostnames
- Name server: a server that can answer DNS queries.
- Authoritative name server:
  - Each zone has a name server that maintains database of host information for its zone
  - Information needs to be updated when host info changes in the zone
- Name servers cache answers, time for caching depends on the TTL (set by the administrator of the DNS server handing out the response), from seconds to days or even weeks.

Domain Name Servers

- Top-level domain (TLD) servers:
  - responsible for com, org, net, edu, etc, and all top-level country domains, e.g. uk, fr, ca, jp.
  - Network Solutions maintains servers for ".com"
- Authoritative DNS servers:
  - organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers.
  - can be maintained by organization or service provider.

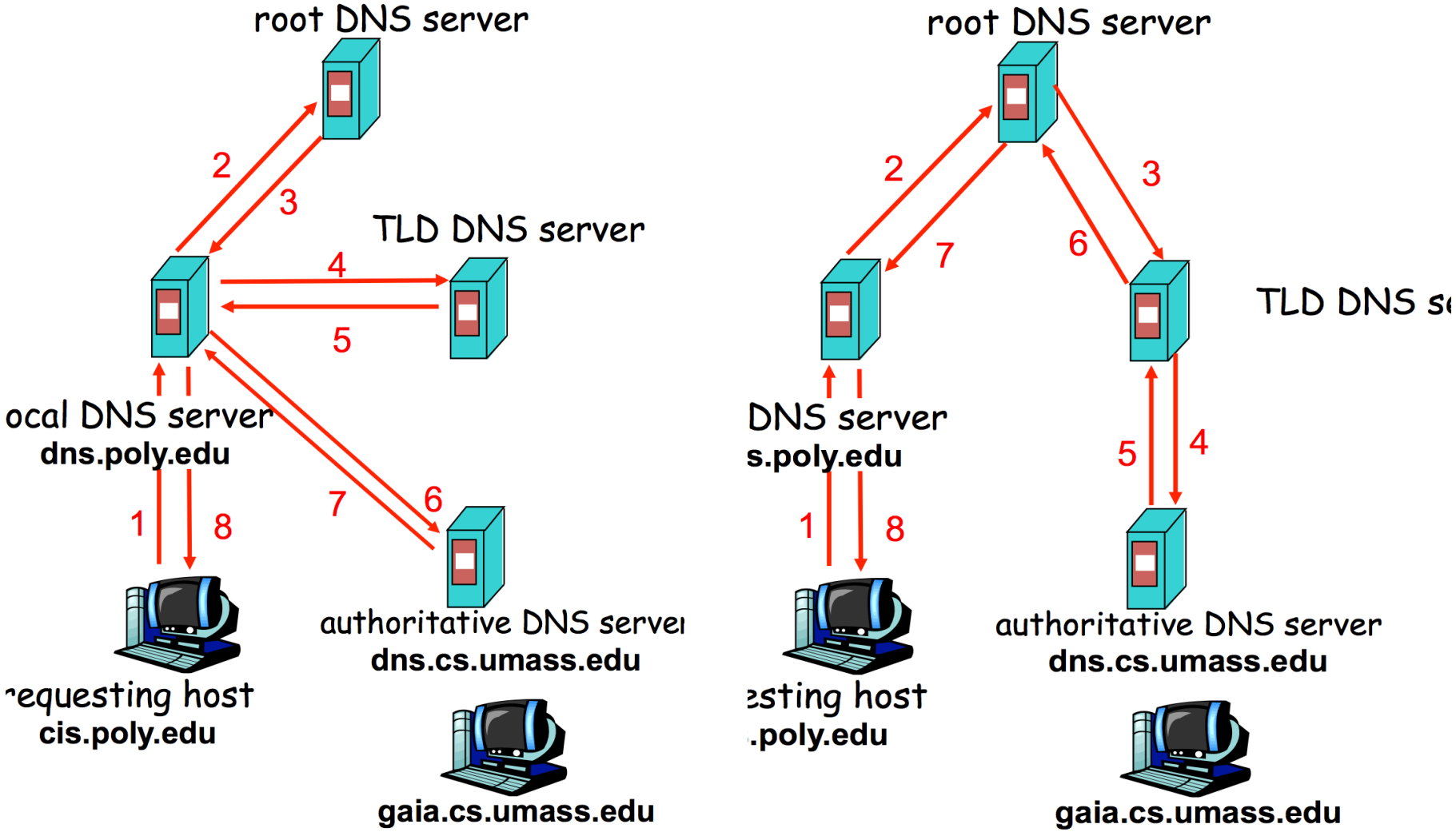
Domain Name Servers - 2

- Local Name Server
  - does not strictly belong to hierarchy
  - each ISP (residential ISP, company, university) has one.

DNS Resolving

- When host makes DNS query, query is sent to its local DNS server.
  - acts as proxy, forwards query into hierarchy.
- Two resolving schemes:
  - Iterative
  - Recursive

DNS Resolving - 2



Caching

- DNS responses are cached
  - Quick response for repeated translations
- Negative results are also cached
  - Save time for nonexistent sites, e.g. misspelling
- Cached data periodically times out
  - Each record has a TTL field

Inherent DNS Vulnerabilities

- Users/hosts typically trust the host-address mapping provided by DNS
  - What bad things can happen with wrong DNS info?
- DNS resolvers trust responses received after sending out queries.
  - How to attack?
- Obvious problem
  - No authentication for DNS responses

DNS Attacks

- Its estimated that 10 percent of servers in the network today are vulnerable to DNS attacks
- Denial of service: servers bombarded with requests
  - Defective implementations RFC1918 (private addresses) that propagate requests/updates that were not supposed to happen (blackhole servers now collect and drop this traffic)
  - Malicious attacks on the backbone: Oct. 2002, DDoS, 9 of the root servers were affected (about 1 hour, ICMP flooding); Second attacks in Feb. 2007, 2 of the root servers were affected

Mitigating Denial of Service

- Root servers are in fact clusters of machines
- Use load balancing
- Queries rate controlled, each source address is limited to a 10KBits/sec and queue size of 3 packets.

More DNS Attacks

- DNS Spoofing:
  - Guessing DNS queries Ids (man in the middle)
  - Compromise the DNS servers itself
- Cache Poisoning: False IP with a high TTL, which the DNS server will cache for a long time
- Email Spoofing: Registration with ICANN often done via email and authenticated by the email address. Return addresses can be falsified
- Mis-configuration: Administrator enters the DNS information incorrectly

A DNS Packet

- It's a UDP packet
- Query ID (16 bits): identifies each request
- Source/Destination IPs: machines that sent and should receive the packet.
- Source/Destination Ports:
  - DNS servers listen on port 53/udp for queries from the outside world;
  - The source port varies, 53, fixed port chosen at random by the operating system, or random port that changes every time.

User Side Attack - Pharming

- Exploit DNS poisoning attack
  - Change IP addresses to redirect URLs to fraudulent sites
  - Potentially more dangerous than phishing attacks
    - Why?
- DNS poisoning attacks have occurred:
  - January 2005, the domain name for a large New York ISP, Panix, was hijacked to a site in Australia.
  - In November 2004, Google and Amazon users were sent to Med Network Inc., an online pharmacy

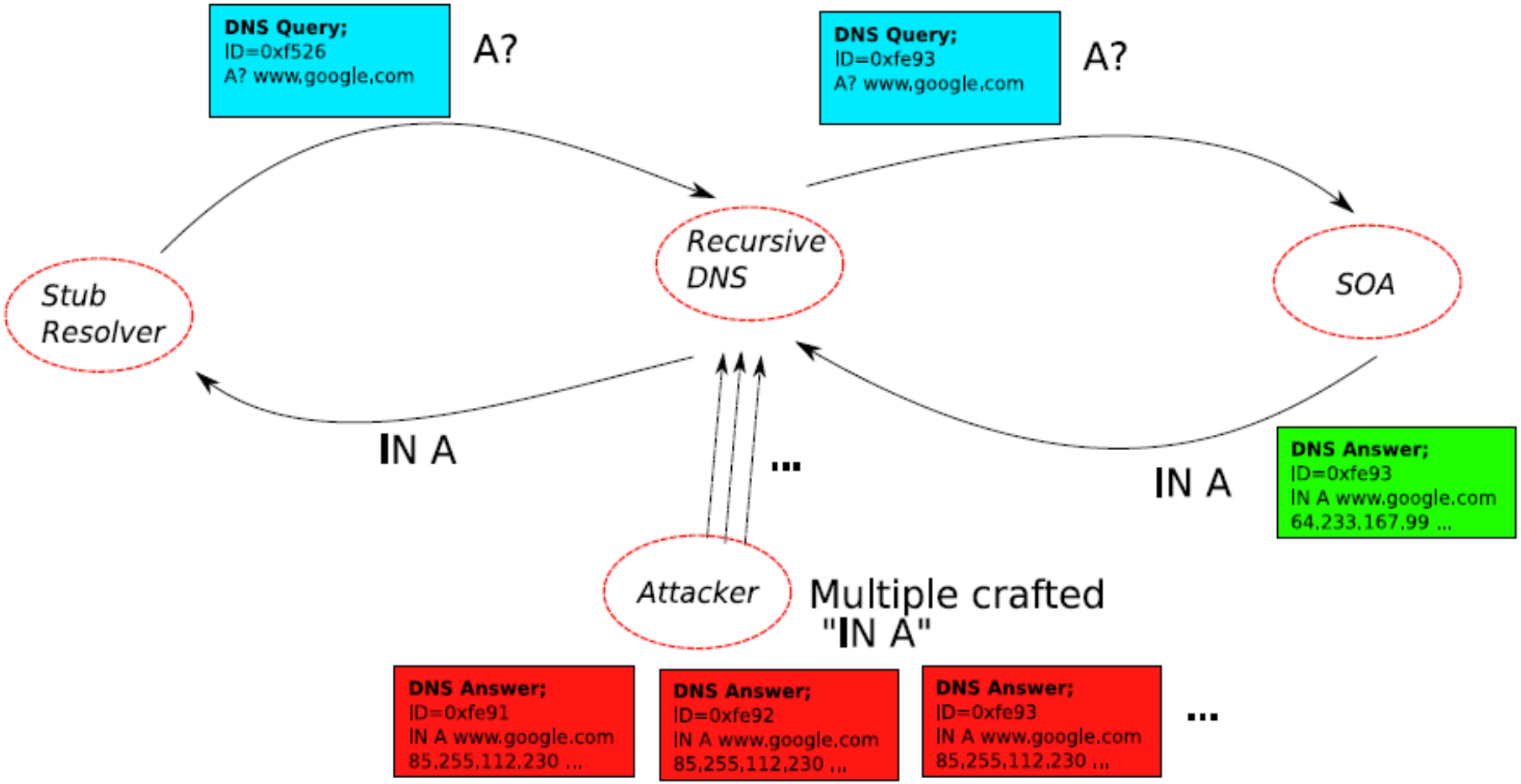
DNS Cache Poisoning

- Attacker wants his IP address returned for a DNS query
- When the resolver asks ns1.google.com for www.google.com, the attacker could reply first, with his own IP
- What is supposed to prevent this?
- Transaction ID
  - 16-bit random number
  - The real server knows the number, because it was contained in the query
  - The attacker has to guess

DNS cache poisoning - 2

- Responding before the real nameserver
  - An attacker can guess when a DNS cache entry times out and a query has been sent, and provide a fake response.
  - The fake response will be accepted only when its 16-bit transaction ID matches the query
  - CERT reported in 1997 that BIND uses sequential transaction ID and is easily predicted
    - fixed by using random transaction IDs

DNS cache poisoning: Racing to Respond First



DNS cache poisoning (Schuba and Spafford in 1993)

- DNS resource records (see RFC 1034)
  - An "A" record supplies a host IP address
  - A "NS" record supplies name server for domain
- First, guess query ID:
  - Ask (dns.target.com) for www.evil.org
  - Request is sent to dns.evil.org (get quid).
- Second, attack:
  - Ask (dns.target.com) for www.yahoo.com
  - Give responses from "dns.yahoo.com" to our chosen IP.

Defense Using The Bailiwicks Rules

- The bailiwick system prevents foo.com from declaring anything about "com", or some other new TLD, or www.google.com

- Using the bailiwicks rules
  - The root servers can return any record
  - The com servers can return any record for com
  - The google.com servers can return any record for google.com

DNS cache poisoning – Birthday attack

- Improve the chance of responding before the real nameserver (discovered by Vagner Sacramento in 2002)
  - Have many (say hundreds of) clients send the same DNS request to the name server
    - Each generates a query
  - Send hundreds of reply with random transaction IDs at the same time
  - Due to the Birthday Paradox, the success probability can be close to 1
    - 300 will give you 50%.
    - 700 will give you 1.07%

DNS poisoning – So far

- Early versions of DNS servers deterministically incremented the ID field
- Vulnerabilities were discovered in the random ID generation
  - Weak random number generator
  - The attacker is able to predict the ID if knowing several IDs in previous transactions
- Birthday attack
  - 16- bit (only 65,536 options).
  - Force the resolver to send many identical queries, with different IDs, at the same time
  - Increase the probability of making a correct guess

DNS cache poisoning - Kaminsky

- Kaminsky Attack
  - Big security news in summer of 2008
  - DNS servers worldwide were quickly patched to defend against the attack
- In previous attacks, when the attacker loses the race, the record is cached, with a TTL.
  - Before TTL expires, no attack can be carried out
  - Posining address for google.com in a DNS server is not easy.

How Does it Work?

- Attacker client requests a random name within the target domain (www12345678.bankofsteve.com), something unlikely to be in cache
- Attacker sends a stream of forged packets to the victim, delegates to another nameserver via Authority records. "I don't know the answer, but you can ask over there". The authority data may well contain the "real" bankofsteve.com nameserver hostnames, but points those nameservers at attacker IPs.
- Victim believes that attacker's nameservers are authoritative for bankofsteve.com. Attacker now owns the entire zone.
- **http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html**

What is New in the Kaminsky Attack?

- The bad guy does not need to wait to try again
- The bad guy asks the resolver to look up www.google.com
  - If the bad guy lost the race, the other race for www.google.com will be suppressed by the TTL
- If the bad guy asks the resolver to look up 1.google.com, 2.google.com, 3.google.com, and so on
  - Each new query starts a new race
- Eventually, the bad guy will win
  - he is able to spoof 183.google.com
  - So what? No one wants to visit 183.google.com

Kaminsky-Style Poisoning

- A bad guy who wins the race for "183.google.com" can end up stealing "www.google.com" as well

- Original malicious response:
  - google.com NS www.google.com
  - www.google.com A 6.6.6.6
- Killer response:
  - google.com NS ns.badguy.com

Kaminsky-Style Poisoning (cont')

- Why it succeeded:
  - Can start anytime; no waiting for old good cached entries to expire
  - No "wait penalty" for racing failure
  - The attack is only bandwidth limited

- Defense (alleviate, but not solve the problem)
  - Also randomize the UDP used to send the DNS query, the attacker has to guess that port correctly as well (increase the space of possible IDs).

DNS Poisoning Defenses

- Difficulty to change the protocol
  - Protocol stability (embedded devices)
  - Backward compatibility.
- Long-term
  - Cryptographic protections
    - E.g., DNSSEC, DNSCurve
  - Require changes to both recursive and authority servers
  - A multi-year process
- Short-term
  - Only change the recursive server (local DNS).
  - Easy to adopt

Short-Term Defenses

- Source port randomization
  - Add up to 16 bits of entropy
  - NAT could de-randomize the port
- DNS 0x20 encoding
  - From Georgia tech, CCS 2008
- Tighter logic for accepting responses

DNS-0x20 Bit Encoding

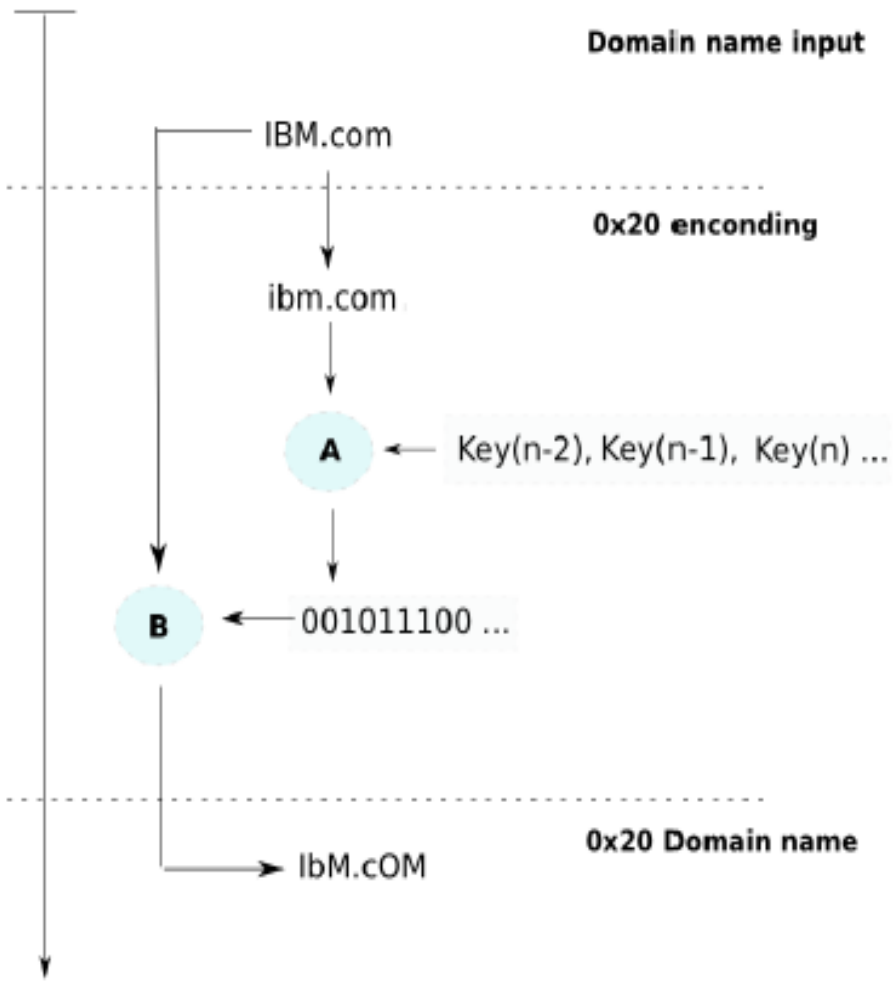
- DNS labels are case insensitive
- Matching and resolution is entirely case insensitive
- A resolver can query in any case pattern
  - E.g., WwW.ExAmpLe.cOM
  - It will get the answer for www.example.com

DNS-0x20 DNS Encoding (cont')

- A DNS response contains the query being asked
- When generating the response, the query is copied from the request exactly into the response
  - The case pattern of the query is preserved in the response
- Open source implementations exhibit this behavior
  - The DNS request is rewritten in place
- The mixed pattern of upper and lower case letters constitutes a channel, which can be used to improve DNS security
  - Only the real server knows the correct pattern

Query Encoding

- Transforms the query into all lowercase
- Encrypt the query with a key shared by all queries on the recursive server (A)
- The cipher text is used to encode the query
  - 0: buff[i] |= 0x20 (upper)
  - 1: buff[i] &= 0x20 (lower)



DNS-0x20 Encoding Analysis

- Do existing authority servers preserve the case pattern?
  - Scan 75 million name servers, 7 million domains
- Only 0.3% mismatch observed

Type	Mismatch	Mismatch pct.	Domain scanned
.com TLD	15451	0.327%	4786993
.net TLD	4437	0.204%	2168352

DNS-0x20 Encoding Analysis (cont')

- Not every character is 0x20 capable
- Improve the forgery resistance of DNS messages only in proportion to the number of upper or lower case characters
  - cia.gov 6-bit entropy
  - licensing.disney.com 18-bit entropy
  - 163.com 3-bit entropy
- TLDs are also vulnerable to Kaminsky-style attacks; but they have few 0x20-capable bits

Long Term Solution

- DNSSEC:
  - Authenticate responses.
  - Google DNS now is enabled by default.

- Challenges in deployment:
  - Response is large, might no linger fit in single UDP message.
  - Legacy software and machines.

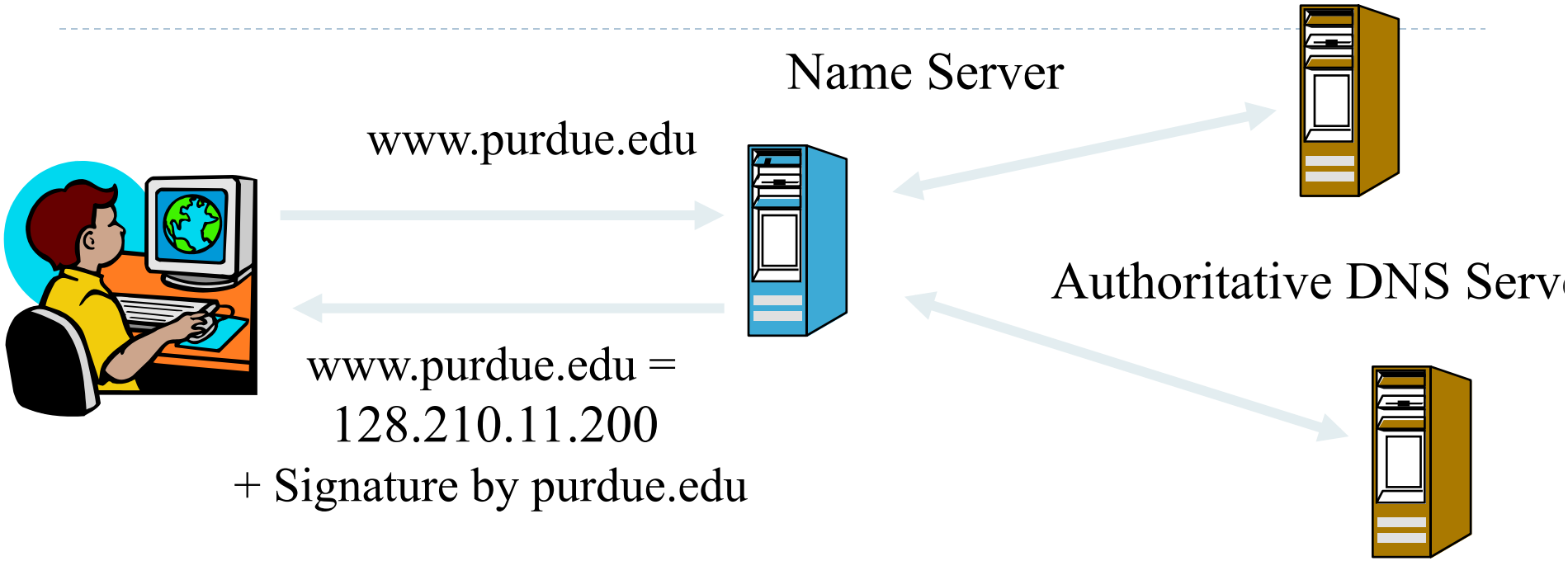
Non-cryptographic Defenses to Cache Poisoning

- Query ID is sequential: attacker floods with a window of future sequences (remember that everything is sent in clear)
- Proposed solution: **randomize the Query ID**. It's harder for the attacker to guess what the Query ID will be. (needs to inject thousands of packets)

DNSSEC

- Proposed solution: addressing authentication and integrity (digital signatures)
- Each DNS zone signs its data using its private key (signing can be done offline, in advance)
- Query for a record will return the requested resource and a digital signature of the requested resource record set
- Resolver will authenticate the response using the corresponding public key of the zone

Secure DNS


Name Server
www.purdue.edu
Authoritative DNS Serv
www.purdue.edu =
128.210.11.200
+ Signature by purdue.edu

What are the Issues?

- How to obtain the public key to verify the digital signature (chicken-and-egg problem)
- Key management is critical (connected with flexibility, original design (RFC 2535) was fatally flawed because did not consider carefully key management)
- Denial of existence : prove a domain for which a query was made, does not exist
- Incremental deployment, flexible to add new domains
- Cryptography alone adds new DoS due to crypto errors and attacks

Key Validation

- How to obtain certified public keys of zones, to verify the digital signatures
- New DNS records KEY, signed by servers in other zones
- Approaches
  - Tree structure: each parents signs the keys of children
  - PGP-style web of trust
  - Mesh: combination between the above, specifies how to find a path of trust

Tree-Based Key Validation

- *A key is valid if it is signed by the parent and the parent key is valid*
- Resolvers configured to trust a master key
- The good:
  - Fits the DNS structure
  - Efficient, no problem to find path of trust
- The bad:
  - Difficult to deploy incrementally
  - Single point of failure
  - Undesirable trust relationships

Web of Trust Key Validation

- *A key is valid if is signed by at least one other key, any chain of trust is possible*
- The good:
  - Ease of deployment
  - No single point of failure
  - Scalable key signing
- The bad:
  - Unauthorized signatures
  - Finding a chain of trust

Mesh-Based Key Validation

- Any zone may sign the public key of any other zone, the resolver decode which signatures are considered valid
- Each zone key has associated a routing record (lists the last link in a chain of trust);
- Default route record
- The good:
  - Ease of deployment
  - No single point of failure
  - Scalable key signing
- The bad ?

Signing the root

- **December 1, 2009:** Root zone signed for internal use by VeriSign and ICANN. ICANN and VeriSign exercise interaction protocols for signing the ZSK with the KSK.
- **January, 2010:** The first root server begins serving the signed root in the form of the DURZ (deliberately unvalidatable root zone). The DURZ contains unusable keys in place of the root KSK and ZSK to prevent these keys being used for validation.
- **Early May, 2010:** All root servers are now serving the DURZ. The effects of the larger responses from the signed root, if any, would now be encountered.
- **May and June, 2010:** The deployment results are studied and a final decision to deploy DNSSEC in the root zone is made.
- **June 16, 2010:** ICANN holds first KSK ceremony event in Culpeper, VA, USA
- **July 12, 2010:** ICANN holds second KSK ceremony event in El Segundo, CA, USA
- **July 15, 2010:** ICANN publishes the root zone trust anchor and root operators begin to serve the signed root zone with actual keys – **The signed root zone is available**.

DNSSec adoption