Network Security

What is network security?

- Confidentiality: only sender, intended receiver should "understand" message contents
  - sender encrypts message
  - receiver decrypts message
  - Others cannot understand the message
  - The identities, timing or frequency should be secrets as well
- Authentication: sender, receiver want to confirm identity of each other
- Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- Access and Availability: services must be accessible and available to users

Outline

- Attacks and counter measures
- Security primer
- Security in different layers

Internet security threats

Mapping:

- before attacking: "case the joint" – find out what services are implemented on network
- Use `ping` to determine what hosts have addresses on network
- Port-scanning: try to establish TCP connection to each port in sequence (and see what happens)
- nmap (http://www.insecure.org/nmap/) mapper: "network exploration and security auditing"

Countermeasures?

Internet security threats

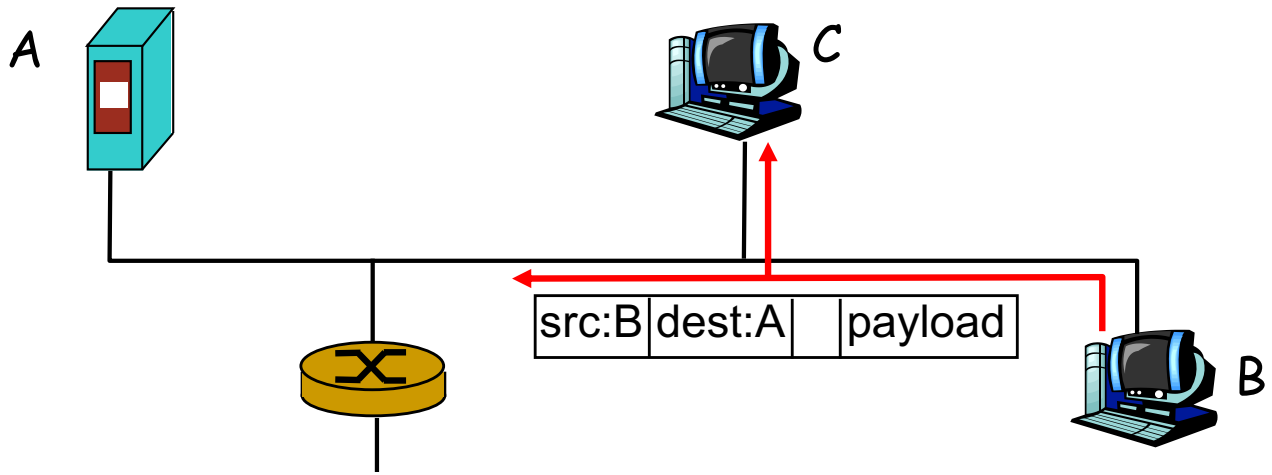
Mapping: countermeasures

- record traffic entering network
- look for suspicious activity (IP addresses, ports being scanned sequentially)

Internet security threats

Packet sniffing:

- broadcast media
- promiscuous NIC reads all packets passing by
- can read all unencrypted data (e.g. passwords)
- e.g.: C sniffs B's packets

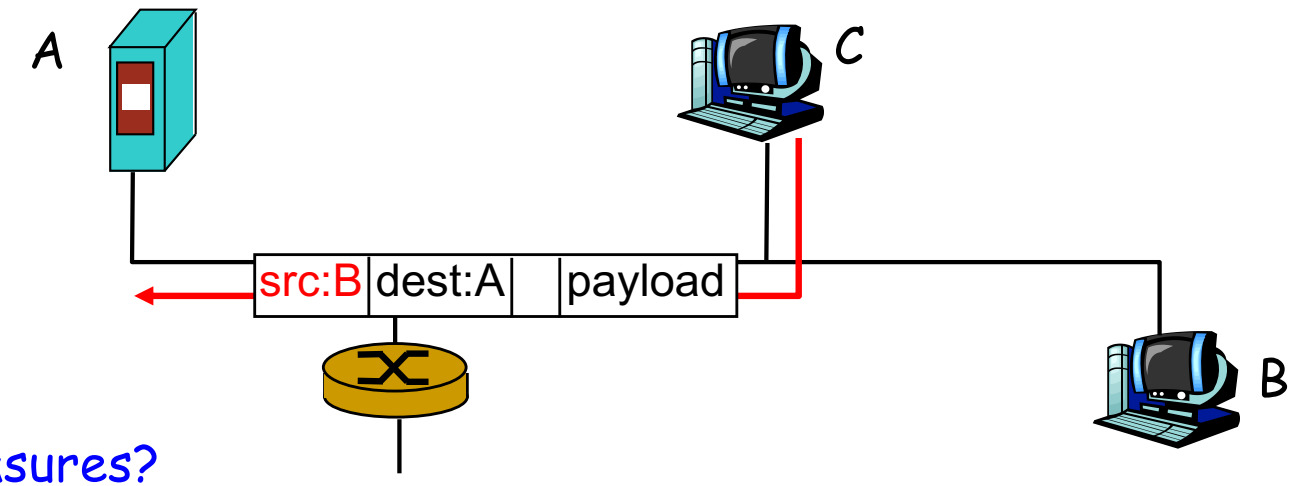


Countermeasures?

Internet security threats

IP Spoofing:

- can generate "raw" IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is spoofed
- e.g.: C pretends to be B

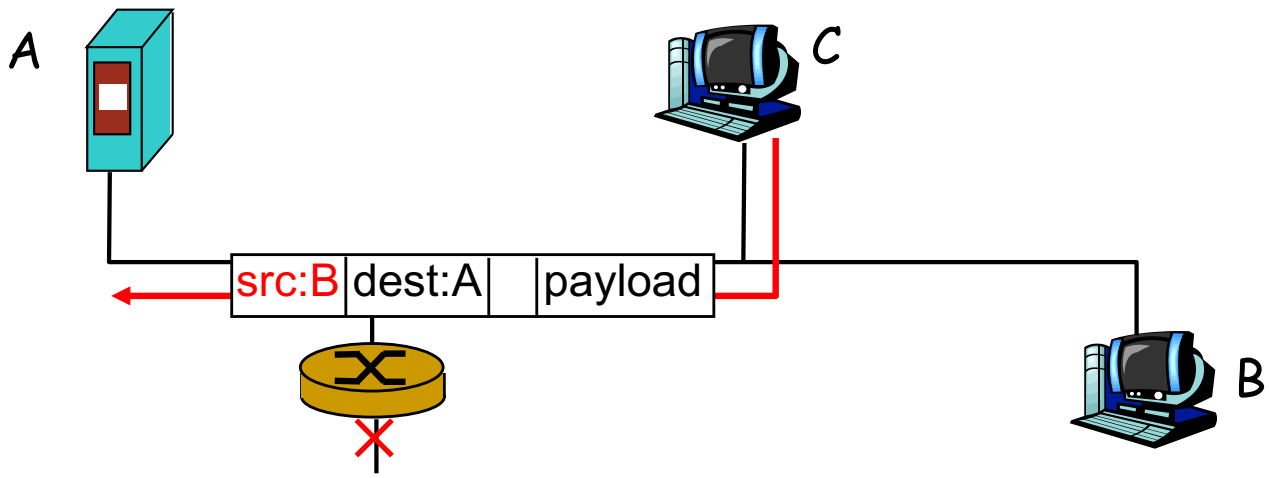


Countermeasures?

Internet security threats

IP Spoofing: ingress filtering

- routers should not forward outgoing packets with invalid source addresses (e.g., datagram source address not in router's network)
- great, but ingress filtering can not be mandated for all networks



Denial of Service (DOS)

- Huge problem in current Internet
- General form
  - Prevent legitimate users from gaining service by overloading or crashing a server

Amazon 'thwarts largest ever DDoS cyber-attack'

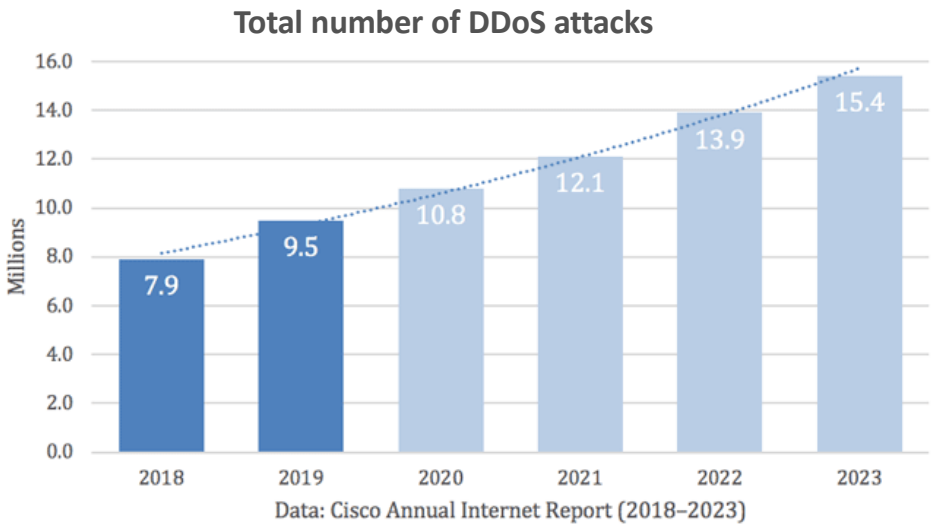
18 June 2020

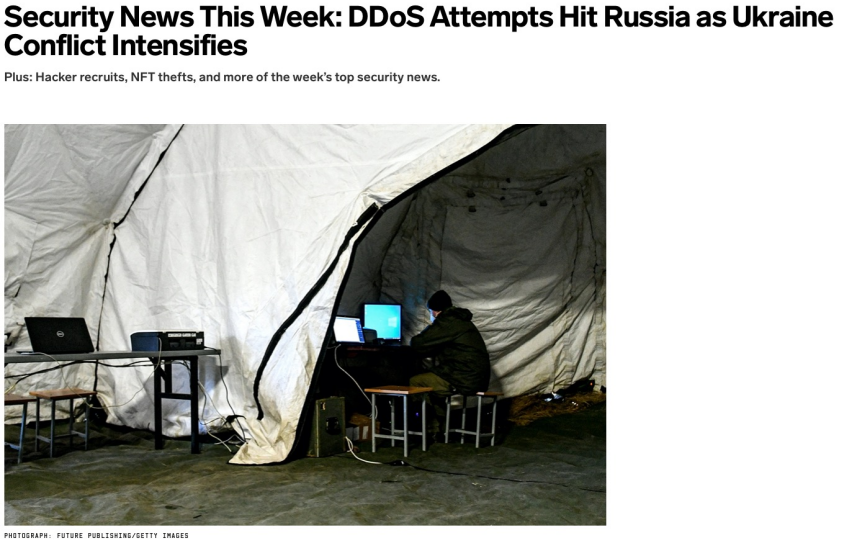
Amazon Web Services is an enormous cloud-services provider and a major money-maker for Amazon

Amazon says its online cloud, which provides the infrastructure on which many websites rely, has fended off the largest DDoS attack in history.

Distributed denial of service (DDoS) attacks are designed to knock a website offline by flooding it with huge amounts of requests until it crashes.

Amazon Web Services (AWS) said the February attack had fired 2.3Tbps.





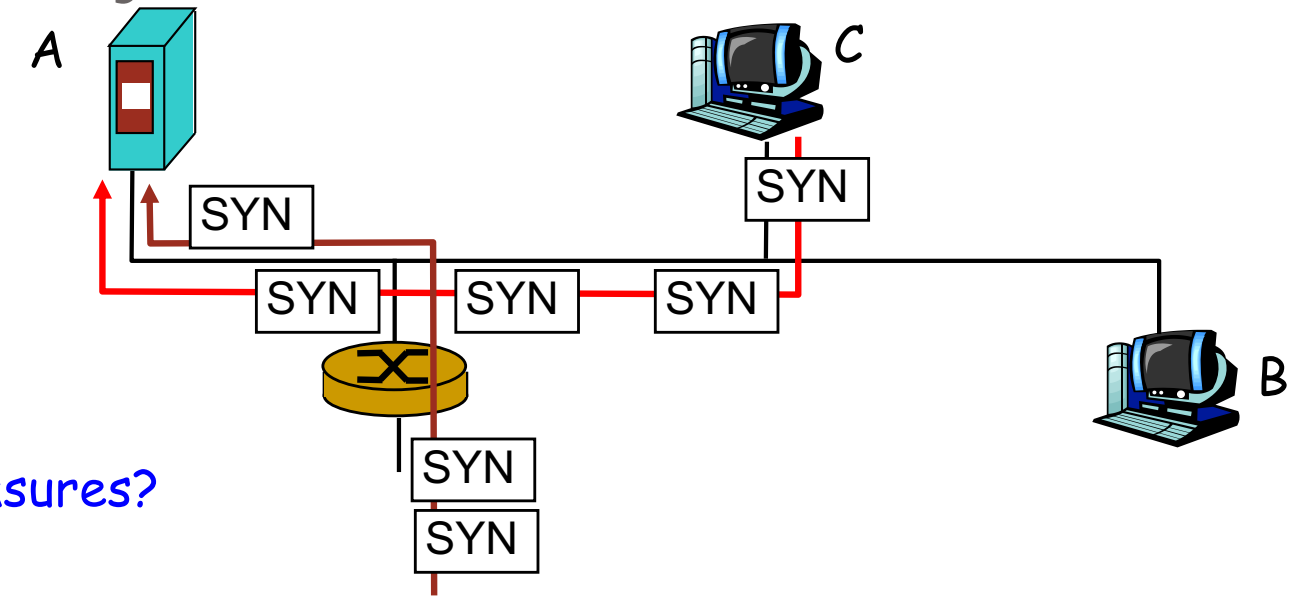
Security News This Week: DDoS Attempts Hit Russia as Ukraine Conflict Intensifies

Plus: Hacker recruits, NFT thefts, and more of the week's top security news.

PHOTOGRAPH: FUTURE PUBLISHING/GETTY IMAGES

In 2024, Cloudflare's autonomous DDoS defense systems blocked around 21.3 million DDoS attacks

DOS: Sync Attack

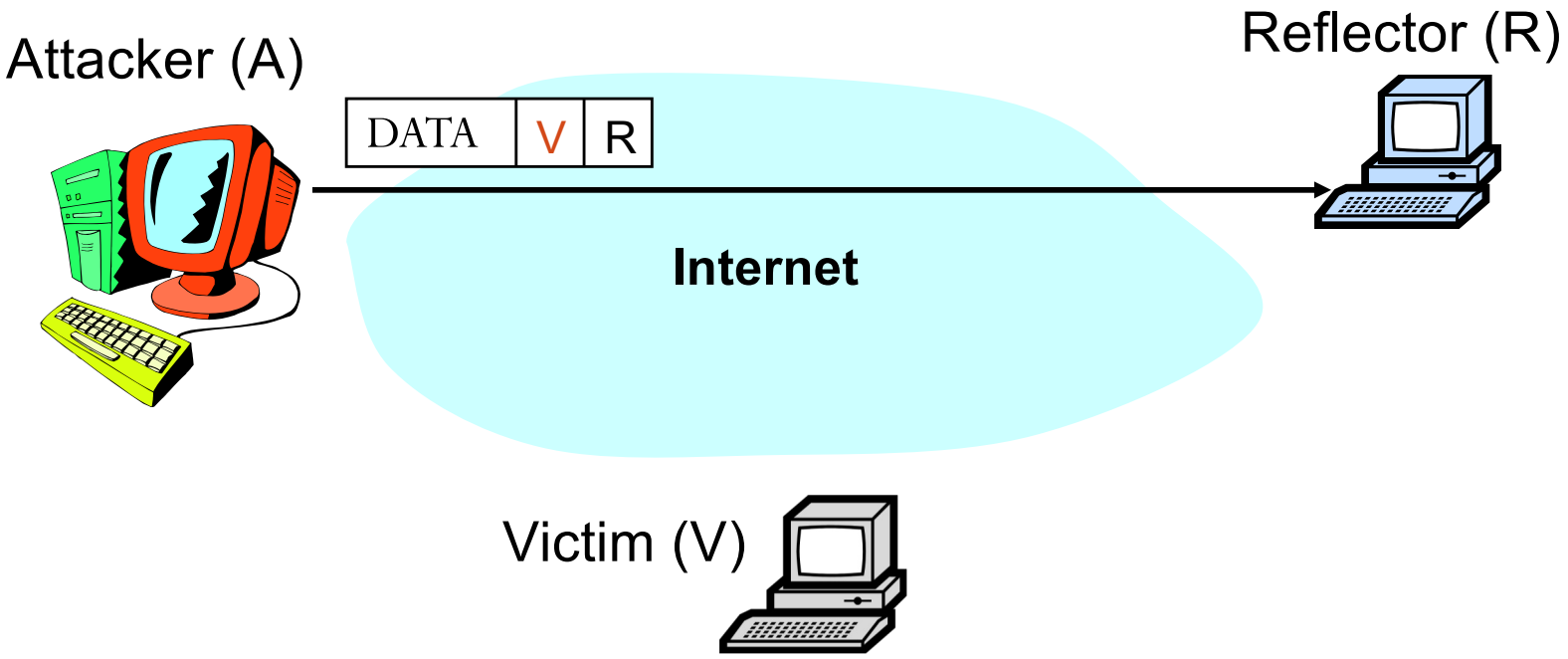


Countermeasures?

- Buggy implementations allow unfinished connections to eat all memory, leading to crash
- Better implementations limit the number of unfinished connections
  - Once limit reached, new SYNs are dropped

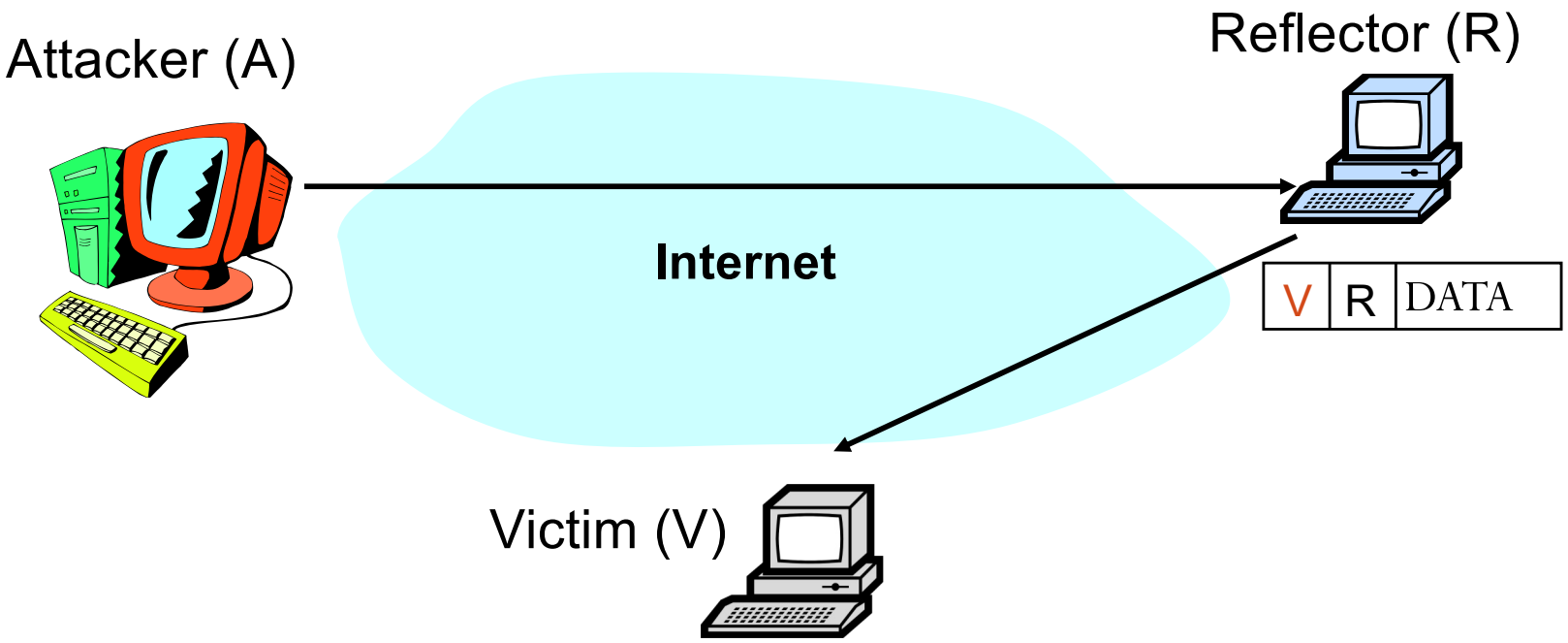
Denial-of-Service Attacks: Reflection

- Reflection
  - Cause one non-compromised host to attack another
  - E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V



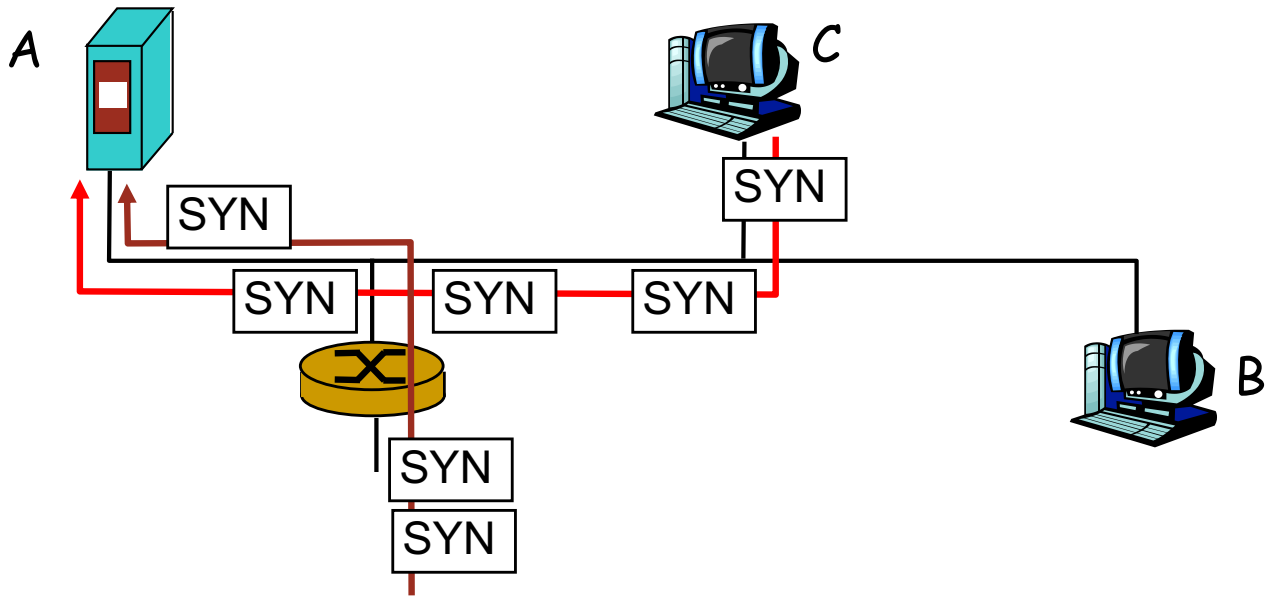
Denial-of-Service Attacks: Reflection

- Reflection
  - Cause one non-compromised host to attack another
  - E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V



Denial of service (DOS): countermeasures

- filter out flooded packets (e.g., SYN) before reaching host: throw out good with bad
- traceback to source of floods (most likely an innocent, compromised machine)



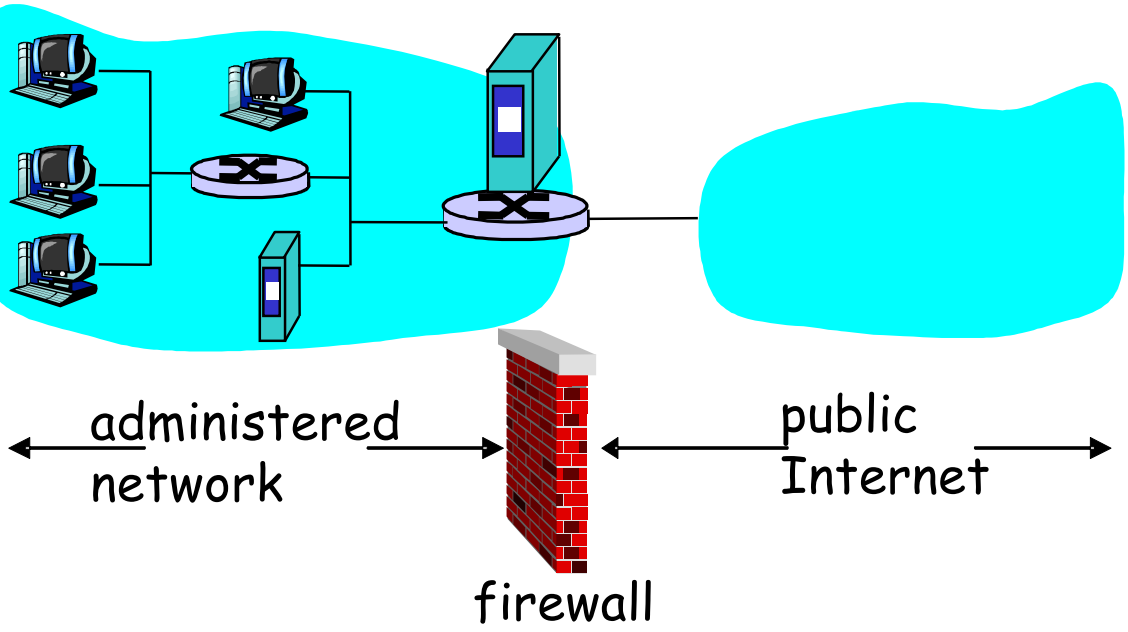
IP Traceback

- Routers probabilistically tag packets with an identifier
- Destination can infer path to true source after receiving enough packets

Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others.



Firewalls: Why

prevent denial of service attacks:

- SYN flooding: attacker establishes many bogus TCP connections, no resources left for "real" connections.

prevent illegal modification/access of internal data.

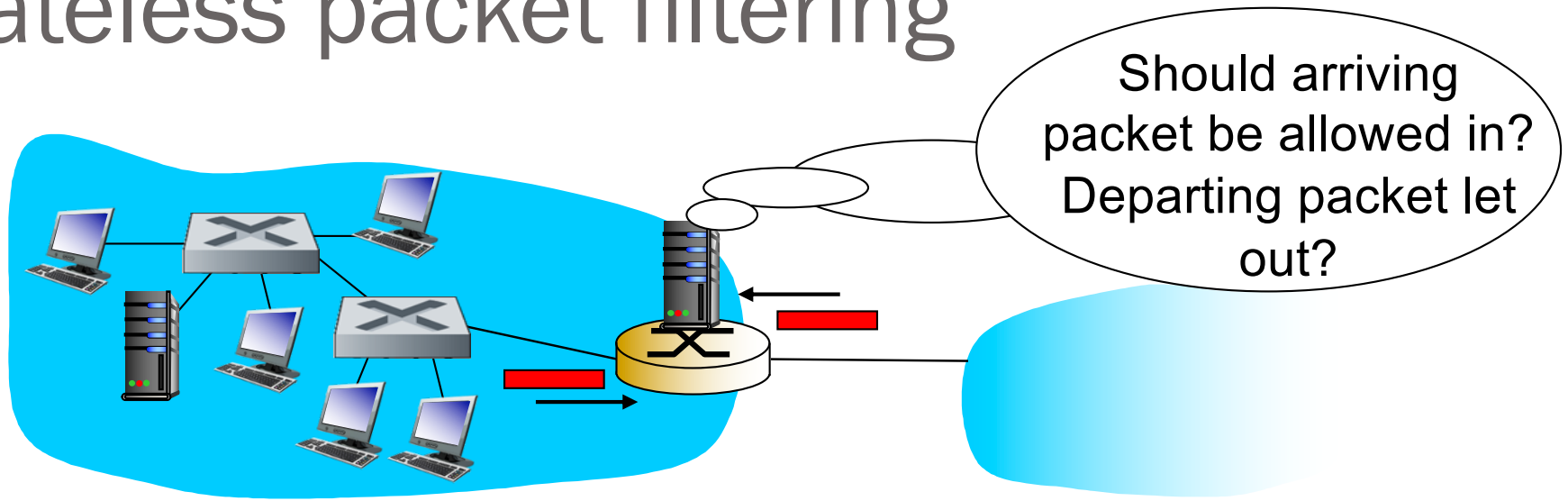
- e.g., attacker replaces CIA's homepage with something else

allow only authorized access to inside network (set of authenticated users/hosts)

two types of firewalls:

- packet-filtering: stateless vs stateful
- application-level

Stateless packet filtering



- internal network connected to Internet via *router firewall*
- router *filters packet-by-packet*, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

Stateless Packet Filtering Example

- Example 1: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - All incoming and outgoing UDP traffic and telnet connections are blocked.
- Example 2: Block inbound TCP segments with ACK=0.
  - Prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateful packet filtering

- stateless packet filter: heavy handed tool
  - admits packets that "make no sense," e.g., source port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- stateful packet filter: track status of every TCP connection
  - track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets
  - Can also be used for UDP datagrams to track request-response

Stateful packet filtering

- Access control list (ACL) augmented to indicate need to check connection state table before admitting packet

action	source address	dest address	proto	source port	dest port	flag bit	check conxion
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----	X
deny	all	all	all	all	all	all	

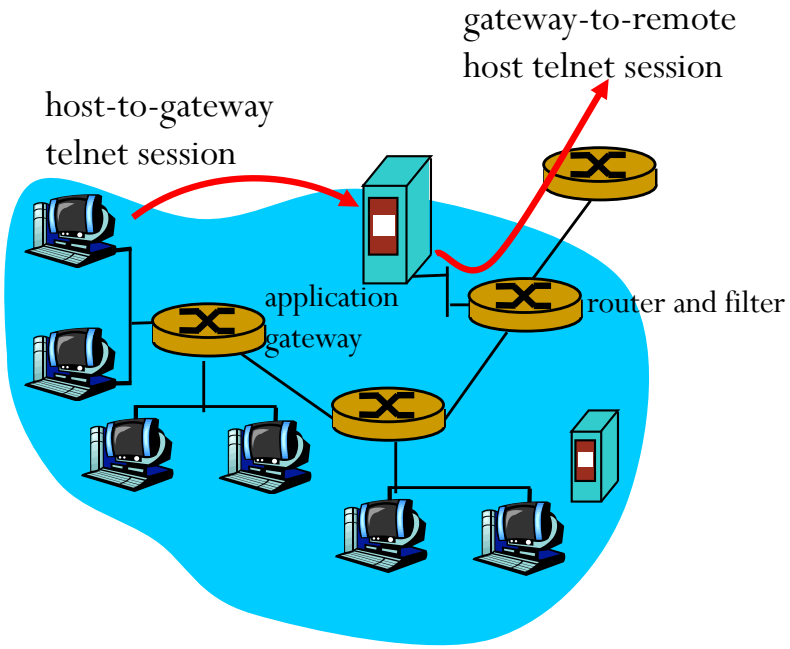
Firewall State Table

- Stores information of active "connections": protocol, IP addresses, port numbers, process ID, timestamp, timeout, direction, etc.
  - TCP states during connection setup and tear down

Source addr	Source port	Dest addr	Dest port	Connection state
222.22.1.100	1090	210.8.23.11	80	ESTABLISHED
222.22.1.20	33163	66.102.9.88	22	TIME_WAIT

Application gateways

- filters packets on application data as well as on IP/TCP/UDP fields.
- *example:* allow selected internal users to telnet outside.



1. require all telnet users to telnet through gateway.
2. for authorized users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway.

Limitations of firewalls and gateways

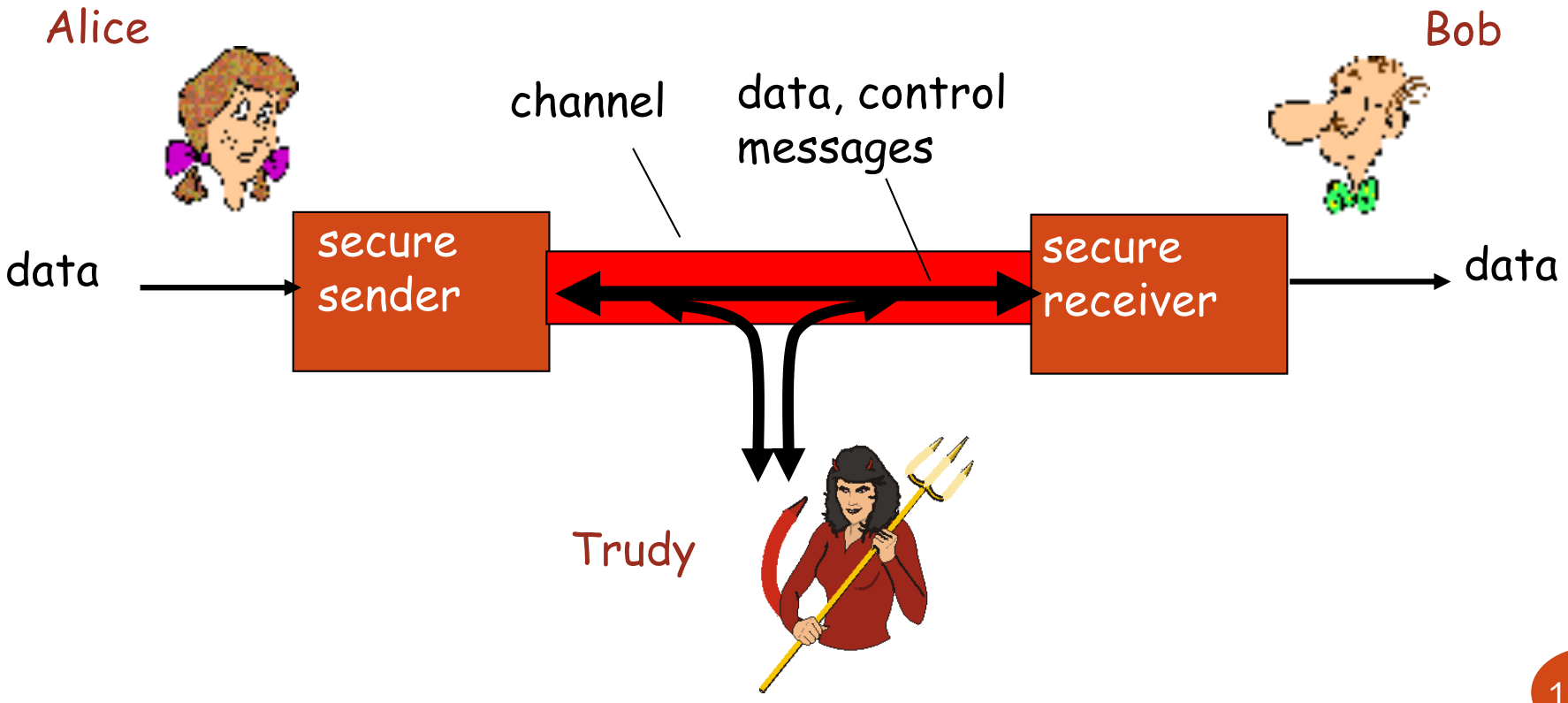
- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple applications need special treatment, each has own application gateway.
- For application gateways, client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP.
- tradeoff: degree of communication with outside world, level of security
- many highly protected sites still suffer from attacks.

Outline

- Attacks and counter measures
- Security primer
- Security in different layers

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) poses threats to the communication



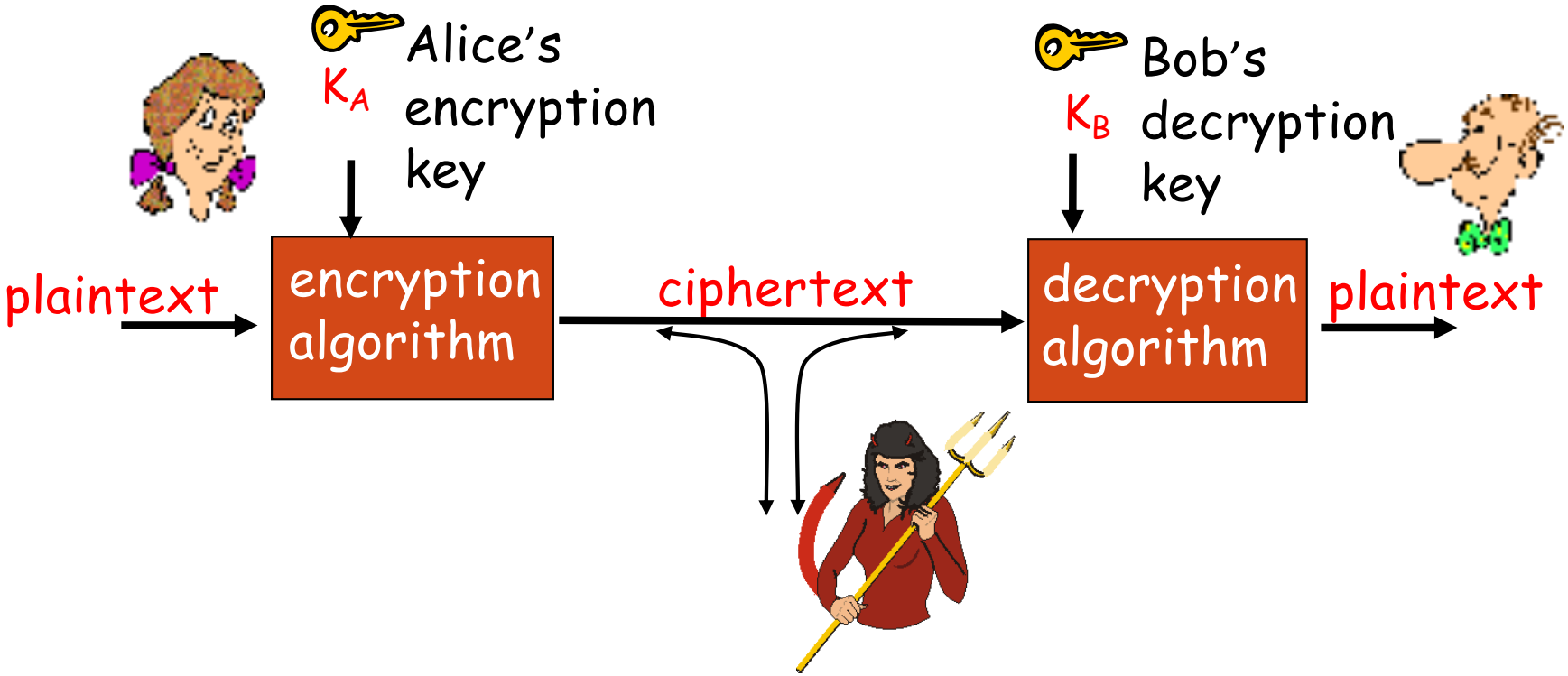
There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: a lot!

- *eavesdrop:* intercept messages
- actively *insert/modify* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

The language of cryptography



symmetric key crypto: sender, receiver keys *identical*

public-key crypto: encryption key *public*, decryption key *secret* (private)

Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz
                  ↓                 ↓
ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

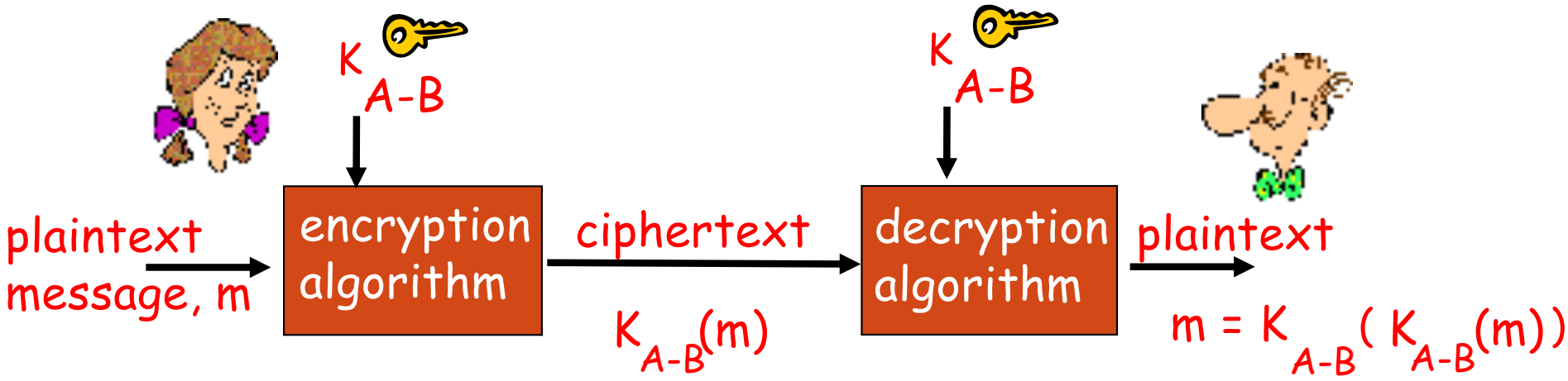
```
Plaintext: bob. i love you. alice

ciphertext: nkn. s gktc wky. mgsbc
```

Q: How hard to break this simple cipher?:

- brute force (how hard?)
- other?

Symmetric key cryptography



symmetric key crypto: Bob and Alice share the same (symmetric) key: K_{A-B}

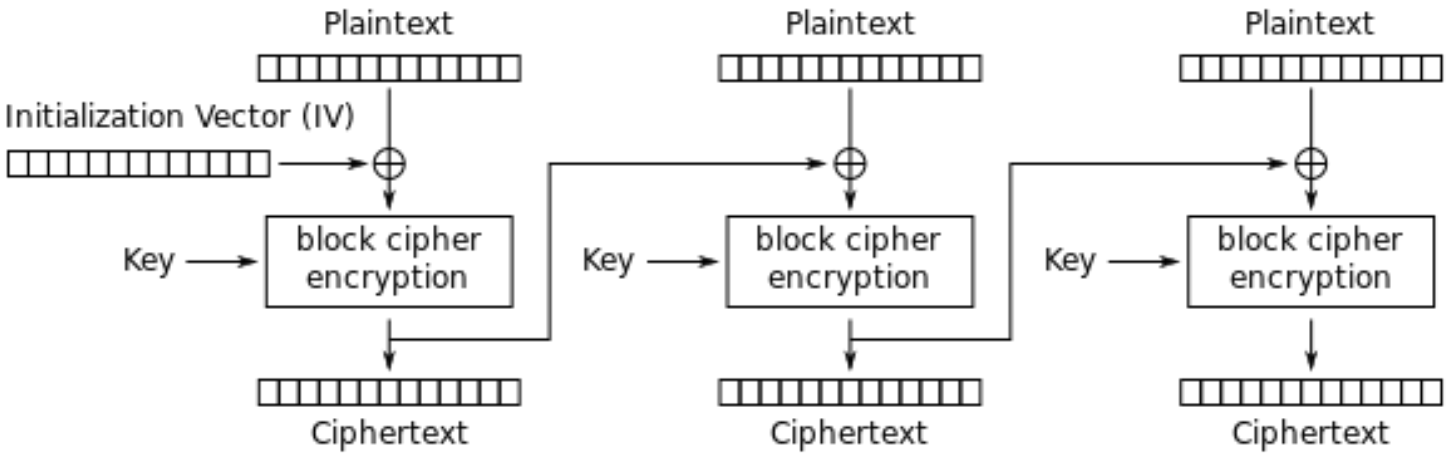
- e.g., key is a known substitution pattern in mono alphabetic substitution cipher

Symmetric key crypto: DES

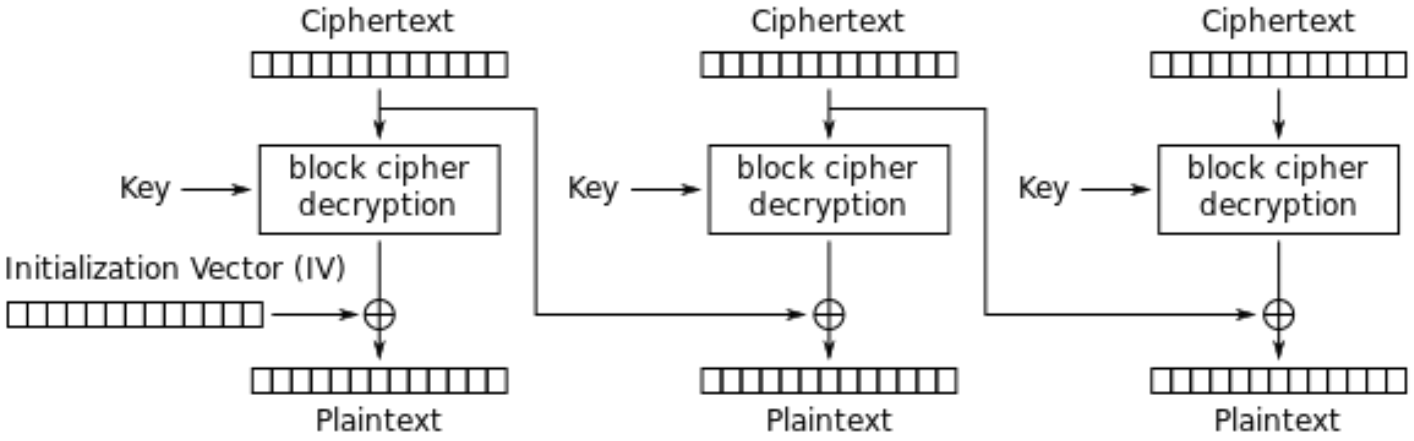
DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- How secure is DES?
  - DES Challenge III was a joint effort between distributed.net and Deep Crack. The key was found in just **22 hours 15 minutes** in January 1999, and the plaintext was "See you in Rome (second AES Conference, March 22-23, 1999)"
- making DES more secure:
  - use three keys sequentially (3-DES) on each datum
  - use cipher-block chaining

Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

AES: Advanced Encryption Standard

- new (Nov. 2001) symmetric-key NIST standard, replacing DES
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- If brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

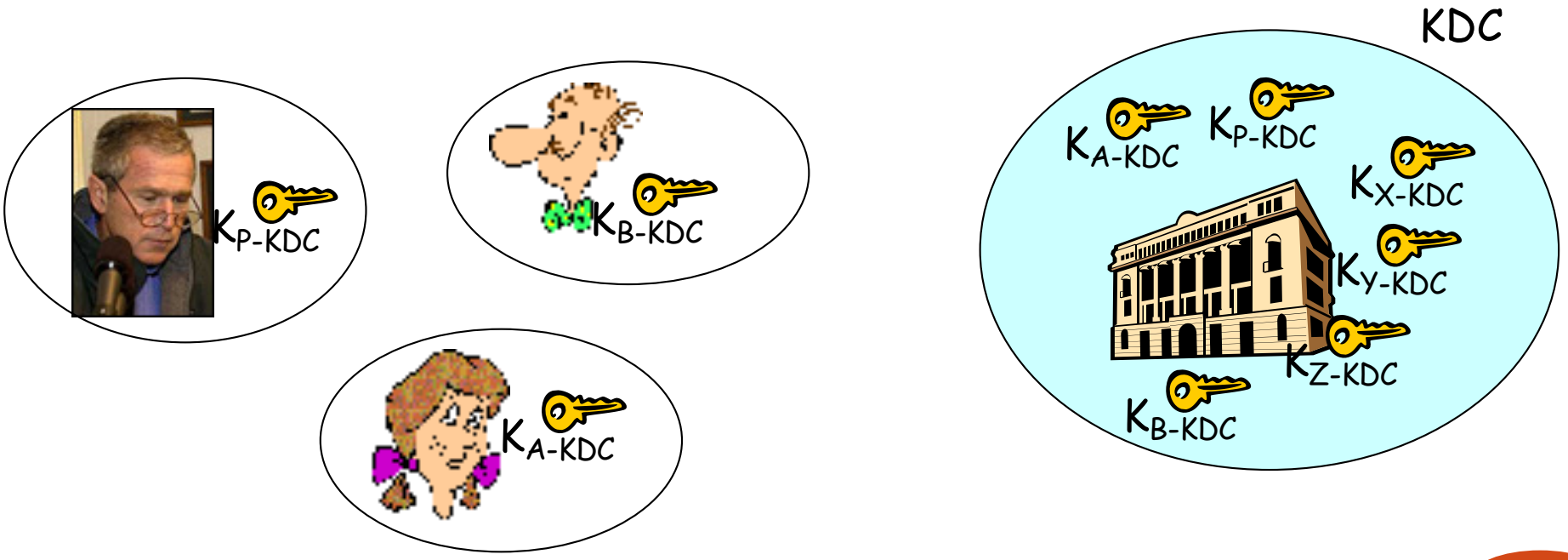
At present, there is no known practical attack that would allow someone without knowledge of the key to read data encrypted by AES when correctly implemented.

Q: how do Bob and Alice agree on key value?

- Key distribution center
- Diffe-Hellman key agreement protocol

Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$, $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?

$K_{A\text{-}KDC}(A,B)$

KDC generates R1

$K_{A\text{-}KDC}(R1, K_{B\text{-}KDC}(A,R1))$

Alice knows R1

$K_{B\text{-}KDC}(A,R1)$

Bob knows to use R1 to communicate with Alice

Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

Diffie-Hellman Key Agreement Protocol

Allow Alice and Bob to agree on a shared secret in a public channel (against passive, i.e., eavesdropping only adversaries)

Setup: a prime p and a base g, both public.

$g^a \bmod p$

$g^b \bmod p$

Pick random, secret a

Compute and send $g^a \bmod p$

Pick random, secret b

Compute and send $g^b \bmod p$

$K = (g^a \bmod p)^b \bmod p = \mathbf{g^{ab} \bmod p} = (g^b \bmod p)^a \bmod p$

Diffie-Hellman Example

- Alice and Bob agree on p = 23 and g = 5.
- Alice chooses a = 6 and sends $5^6 \bmod 23 = 8$
- Bob chooses b = 15 and sends $5^{15} \bmod 23 = 19$
- Alice computes $19^6 \bmod 23 = 2$.
- Bob computes $8^{15} \bmod 23 = 2$.
- Then 2 is the shared secret.

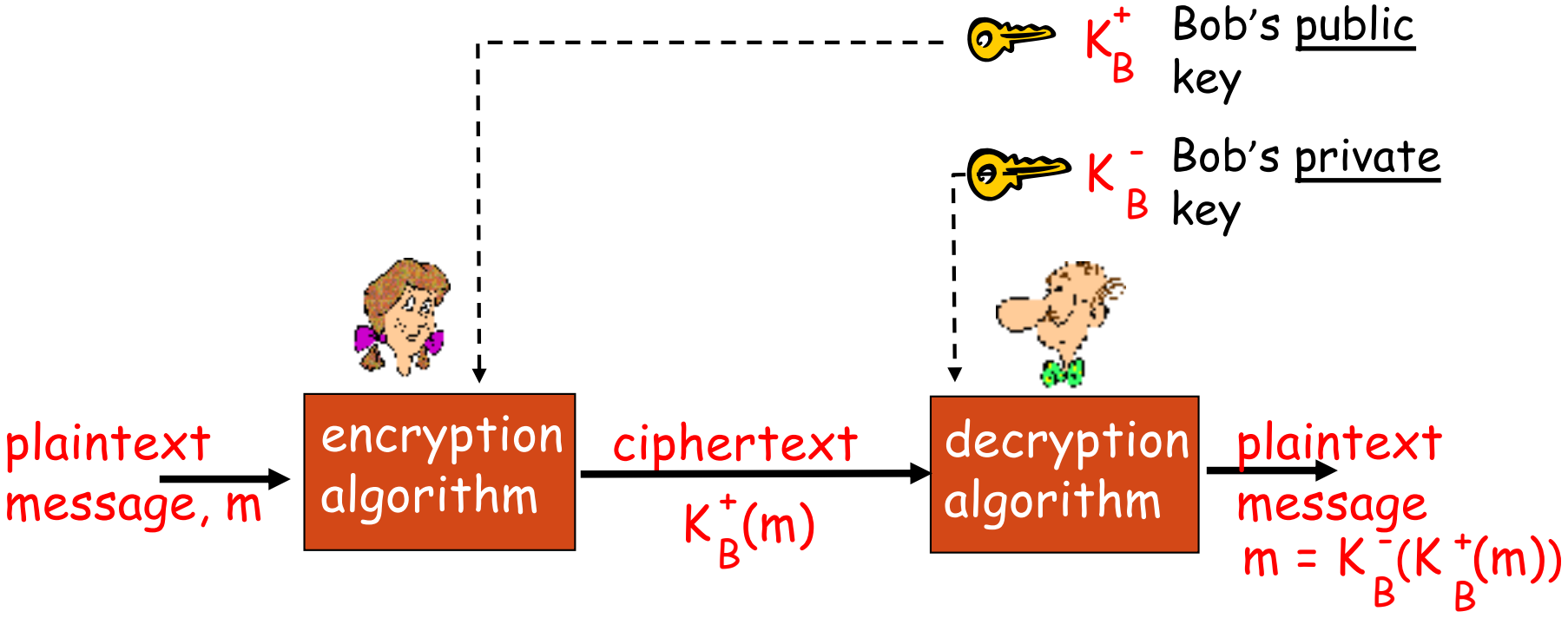
- Assumption: Finding $5^{??} \bmod 23 = 8$ and $5^{??} \bmod 23 = 19$ are hard

Public Key Cryptography

public key cryptography

- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

Requirements:

1. need $K_B^+( )$ and $K_B^-( )$ such that

$$K_B^-(K_B^+(m)) = m$$

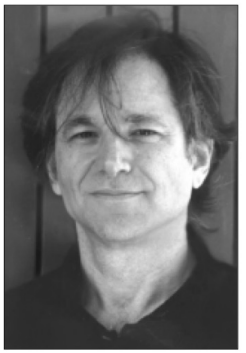
2. given public key K_B^+ , it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

Rivest, Shamir, and Adleman Receive 2002 Turing Award

Ronald L. Rivest

Adi Shamir



Leonard M. Adleman

RSA: Choosing keys

1. Choose two large prime numbers p, q. (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose e (with $e<n$) that has no common factors with z. (e, z are "relatively prime").

4. Choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).

5. *Public* key is (n,e). *Private* key is (n,d).

K_B^+ K_B^-

RSA: Encryption, decryption

0. Given (n,e) and (n,d) as computed above

K_B^+ K_B^-

1. To encrypt bit pattern, m, compute

$c = m^e \mod n$ (i.e., remainder when m^e is divided by n)

2. To decrypt received bit pattern, c, compute

$m = c^d \mod n$ (i.e., remainder when c^d is divided by n)

Magic happens!

$$m = \underbrace{(m^e \mod n)}_{c}{}^d \mod n$$

RSA example:

Bob chooses *p=5, q=7*. Then *n=p*q=35, z=(p-1)*(q-1)=24*.

e=5 (so *e, z* relatively prime).
d=29 (so *ed-1* exactly divisible by z)

encrypt:

letter	m	m^e	$c = m^e \bmod n$
L	12	1524832	17

decrypt:

c	c^d	$m = c^d \bmod n$	letter
17	481968572106750915091411825223071697	12	L

RSA: Why is that $m = (m^e \bmod n)^d \bmod n$

Useful number theory result: If p,q prime and $n = pq$, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

(using number theory result above)

$$= m^1 \bmod n$$

(since we chose ed to be divisible by $(p-1)(q-1)$ with remainder 1)

$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

Result is the same!

Digital Signatures

Application of public key crypto

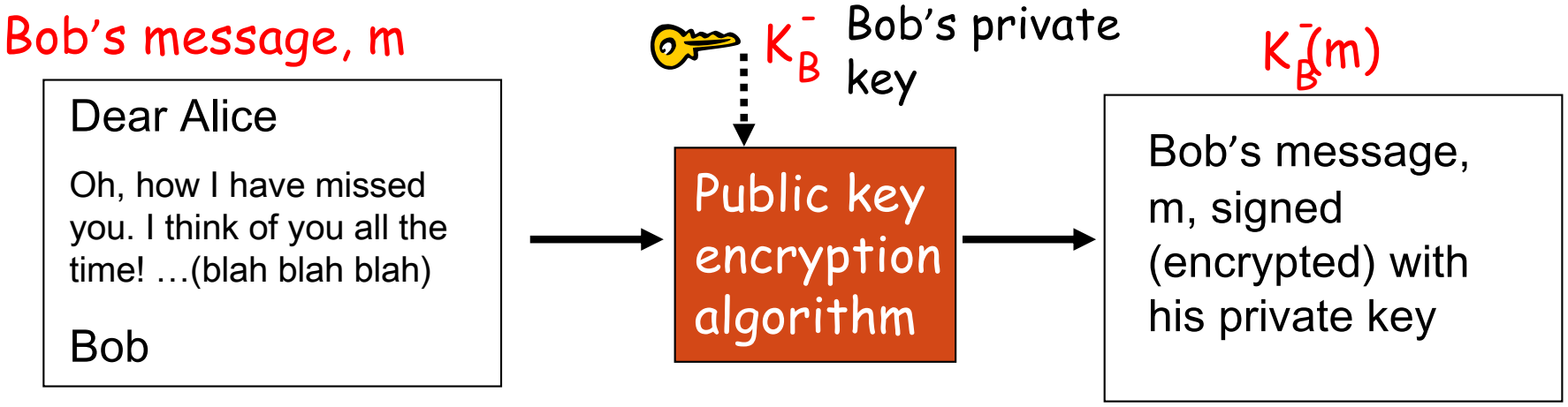
Cryptographic technique analogous to hand-written signatures.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital Signatures

Simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital Signatures (more)

- Suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

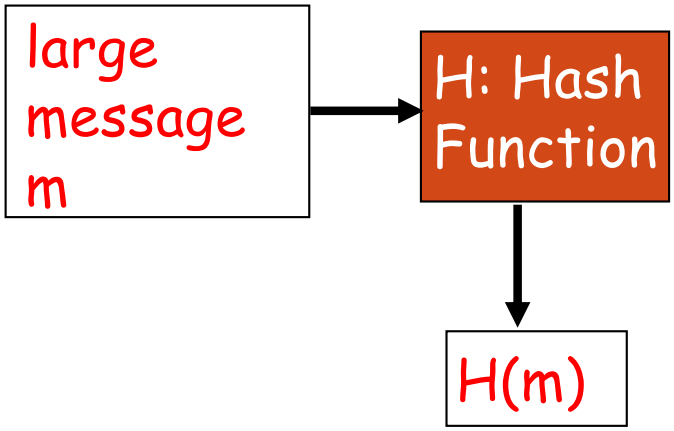
Alice thus verifies that:

- Bob signed m.
- No one else signed m.
- Bob signed m and not m'.

Non-repudiation:

✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

Message Digests



Computationally expensive to public-key-encrypt long messages

Goal: fixed-length, easy- to-compute digital "fingerprint"

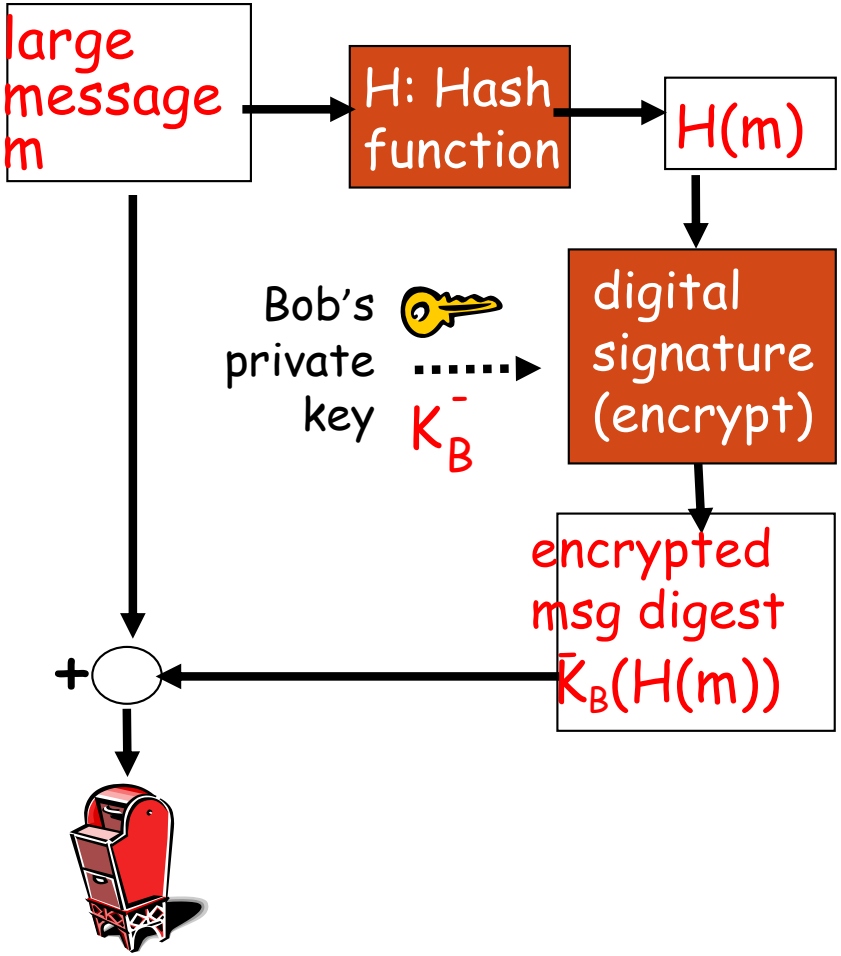
- apply hash function H to *m*, get fixed size message digest, *H(m)*.

Hash function properties:

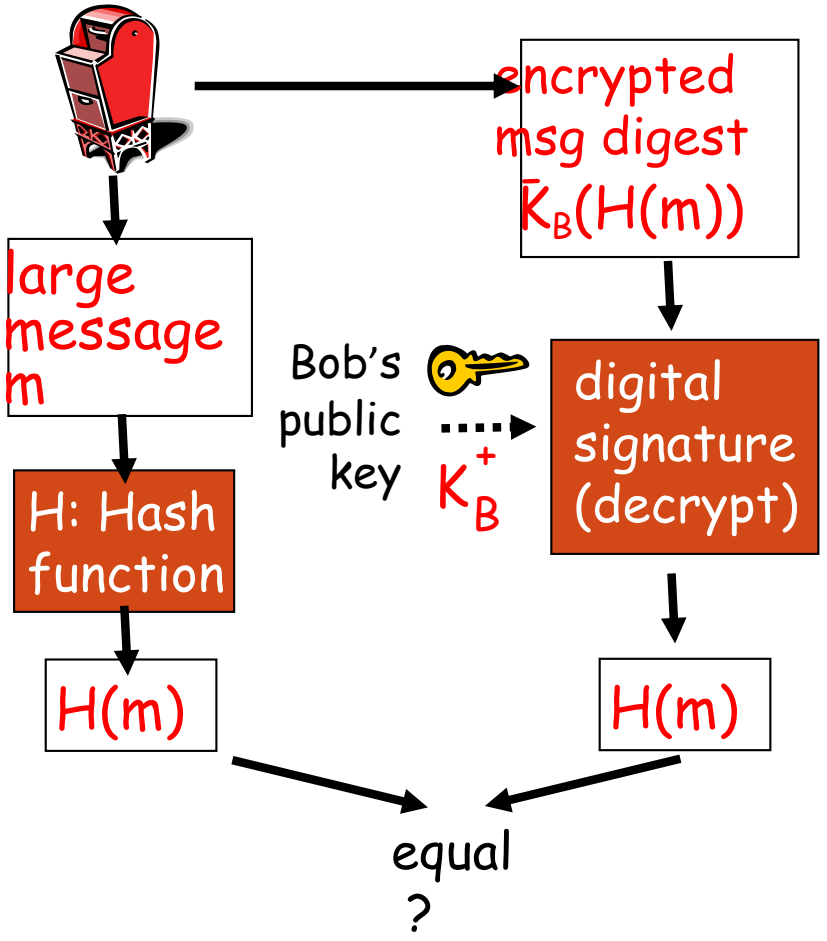
- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



X.509 certificate of VeriSign

VeriSign Trust Network, (c) 1998 VeriSign, Inc. - For authorized use only, Class 2 Public Primary Certification Authority - G2
Self-signed root certificate
Expires: Tuesday, August 1, 2028 at 7:59:59 PM Eastern Daylight Time
⚠ This certificate has not been verified by a third party

▶ **Trust**
▼ **Details**

Issuer Name	
Country or Region	US
Organization	VeriSign, Inc.
Organizational Unit	Class 2 Public Primary Certification Authority - G2
Organizational Unit	(c) 1998 VeriSign, Inc. - For authorized use only
Organizational Unit	VeriSign Trust Network
Serial Number	00 B9 2F 60 CC 88 9F A1 7A 46 09 B8 5B 70 6C 8A AF
Version	1
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	None
Subject Name	
Country or Region	US
Organization	VeriSign, Inc.
Organizational Unit	Class 2 Public Primary Certification Authority - G2
Organizational Unit	(c) 1998 VeriSign, Inc. - For authorized use only
Organizational Unit	VeriSign Trust Network
Not Valid Before	Sunday, May 17, 1998 at 8:00:00 PM Eastern Daylight Time
Not Valid After	Tuesday, August 1, 2028 at 7:59:59 PM Eastern Daylight Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	128 bytes : A7 88 01 21 74 2C E7 1A 03 F0 98 E1 97 3C 0F 21 08 F1 9C B8 97 E9 9A FC C2 04 06 13 BE 5F 52 C8 CC 1E 2C 12 56 2C B8 01 69 2C CC 99 1F AD B0 96 AE 79 04 F2 13 39 C1 7B 98 BA 08 2C E8 C2 84 13 2C AA 69 E9 09 F4 C7 A9 02 A4 42 C2 23 4F 4A D8 F0 0E A2 FB 31 6C C9 E6 6F 99 27 07 F5 E6 F4 4C 78 9E 6D EB 46 86 FA B9 86 C9 54 F2 B2 C4 AF D4 46 1C 5A C9 15 30 FF 0D 6C F5 2D 0E 6D CE 7F 77
Exponent	65537
Key Size	1,024 bits
Key Usage	Any
Signature	128 bytes : 72 2E F9 7F D1 F1 71 FB ...
Fingerprints	
SHA-256	3A 43 E2 20 FE 7F 3E A9 65 3D 1E 21 74 2E AC 2B 75 C2 0F D8 98 03 05 BC 50 2C AF 8C 2D 9B 41 A1
SHA-1	B3 EA C4 47 76 C9 C8 1C EA F2 9D 95 B6 CC A0 08 1B 67 EC 9D

Algo. for signing the certificate

Generated by hashing the certificate content and encrypting it with the CA's private key

Computed from hashing certificate data

Hash Function Algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - In 1996 a flaw was found in the design of MD5 ☹ -- "should be considered cryptographically broken and unsuitable for further use"
- SHA-2, SHA-3
  - 224, 256, 384 or 512 bits in digests

Certification for Public Key

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities
- DH

Public key problem:

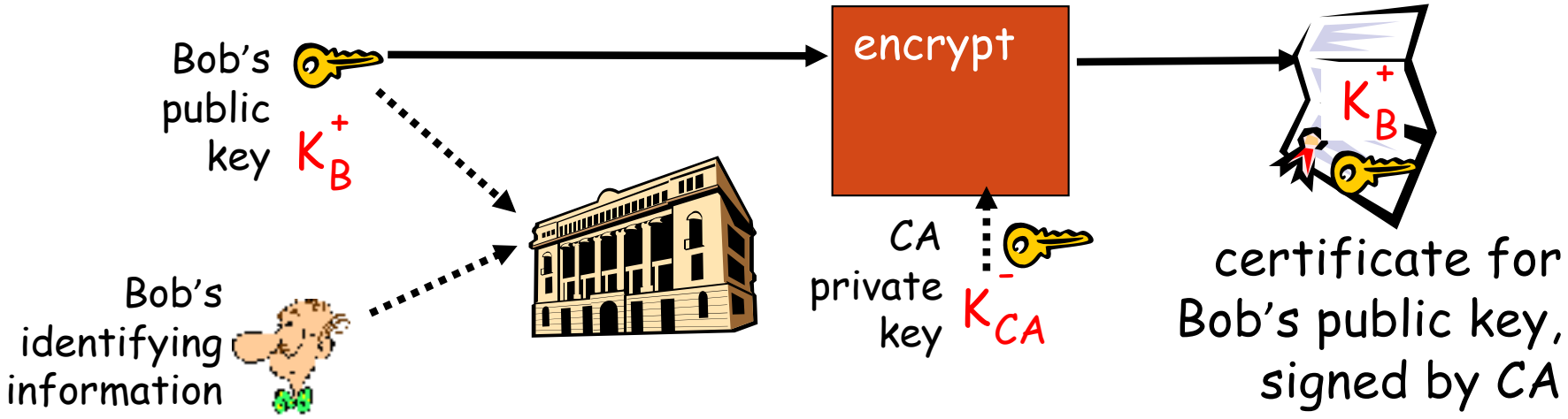
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- trusted certification authority (CA)

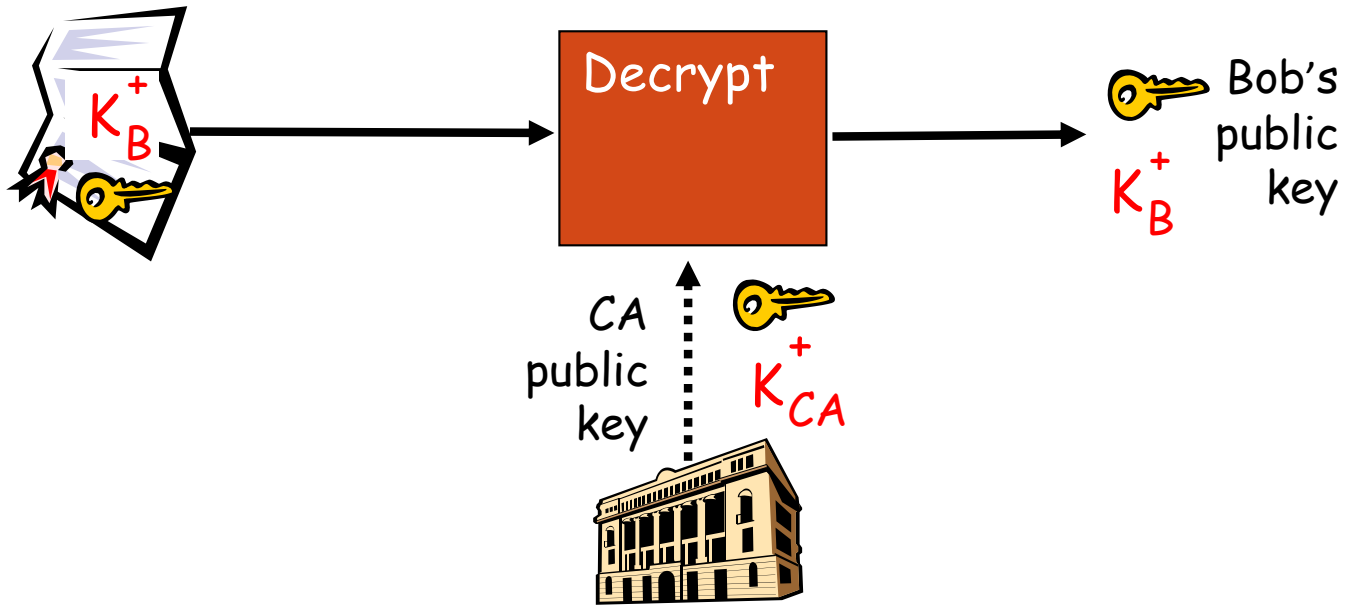
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, server) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key
  - Agree or not?



What have we learned so far?

- Message confidentiality: shared key or public key crypto
- Message integrity: hash
- Authenticity of a digital message: digital signature

What about authenticity of sender/receiver?

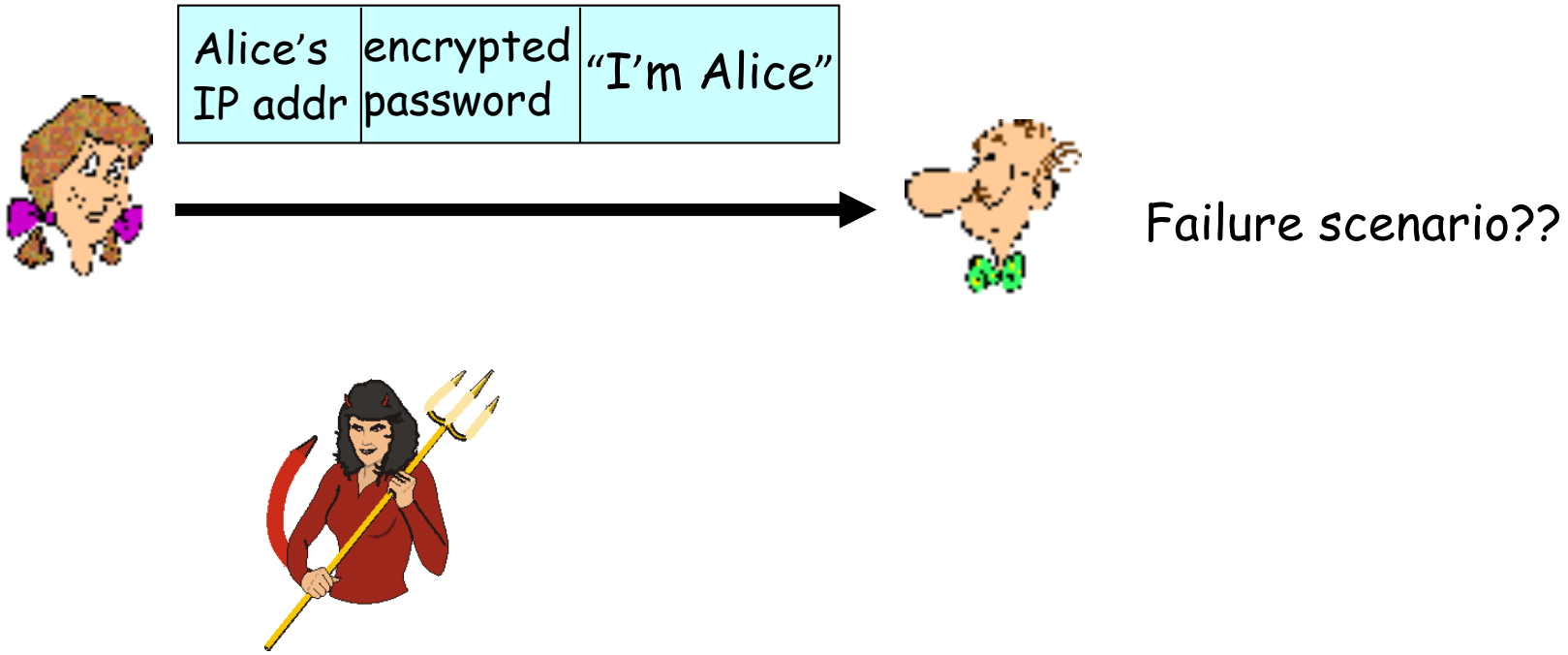
- ARP poisoning
- IP/MAC address spoofing
- phishing attacks

Need authentication

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol: assume pre-shared secret between Alice and Bob

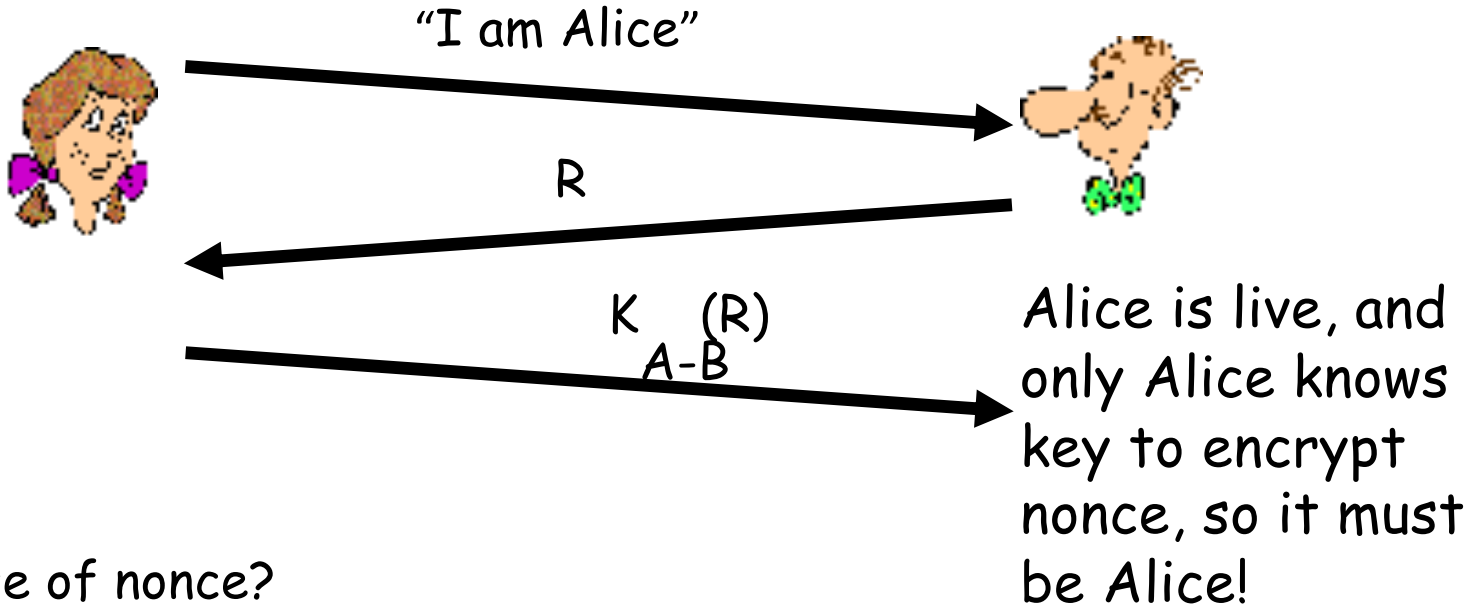


Authentication: Symmetric Key Crypto

Goal: avoid IP proofing, playback attack

Nonce: number (R) used only *once –in-a-lifetime*

ap: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

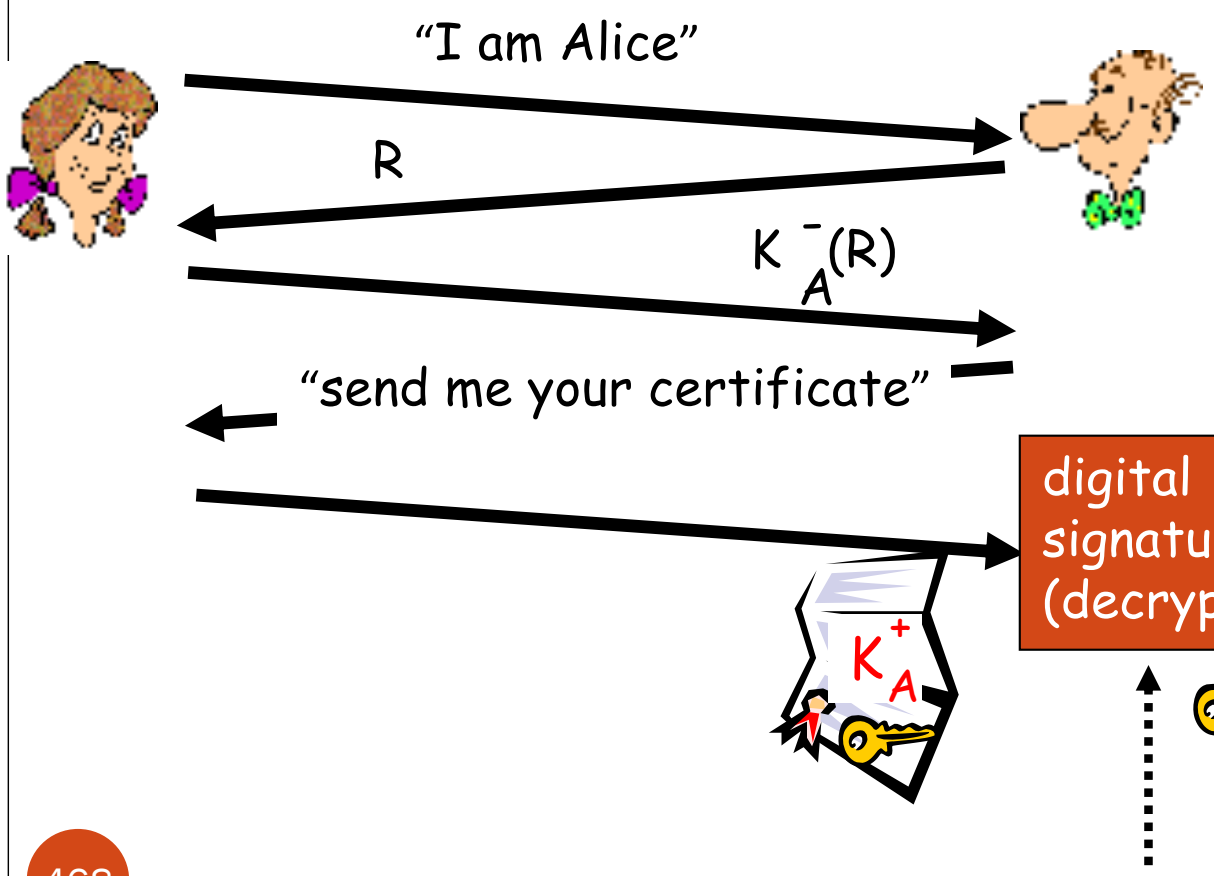


Why the use of nonce?

Authentication: Public Key Crypto

- can we authenticate using public key techniques?

use nonce, public key cryptography

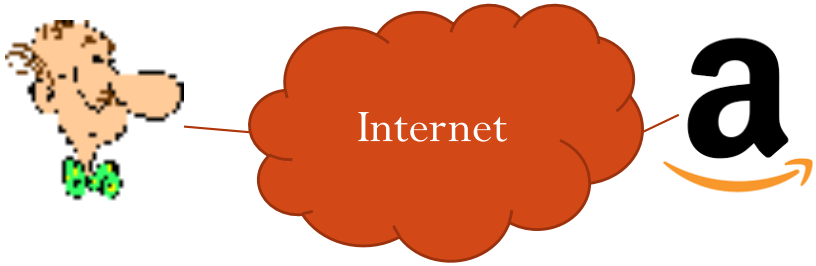


$K_A^+(K_A^-(R)) = R$

and knows only Alice could have the private key, that encrypted R such that

$K_A^+(K_A^-(R)) = R$

A real-life scenario



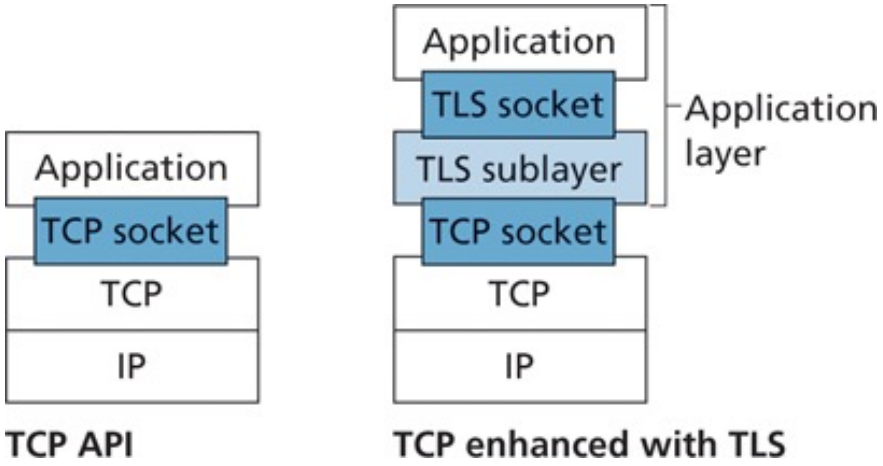
- Bob logs in to his amazon.ca account to purchase a bag of coffee beans
- What types of network security does Bob want?
  - Authentication of the website
  - Confidentiality of his account log-in information and credit card information
  - Message integrity of his on-line transactions (against alternation, insertion, replay, reordering)
  - Service availability
- What security primitives discussed so far can be used?
  - Certificate
  - Symmetric key crypto
  - Hash
  - …

Outline

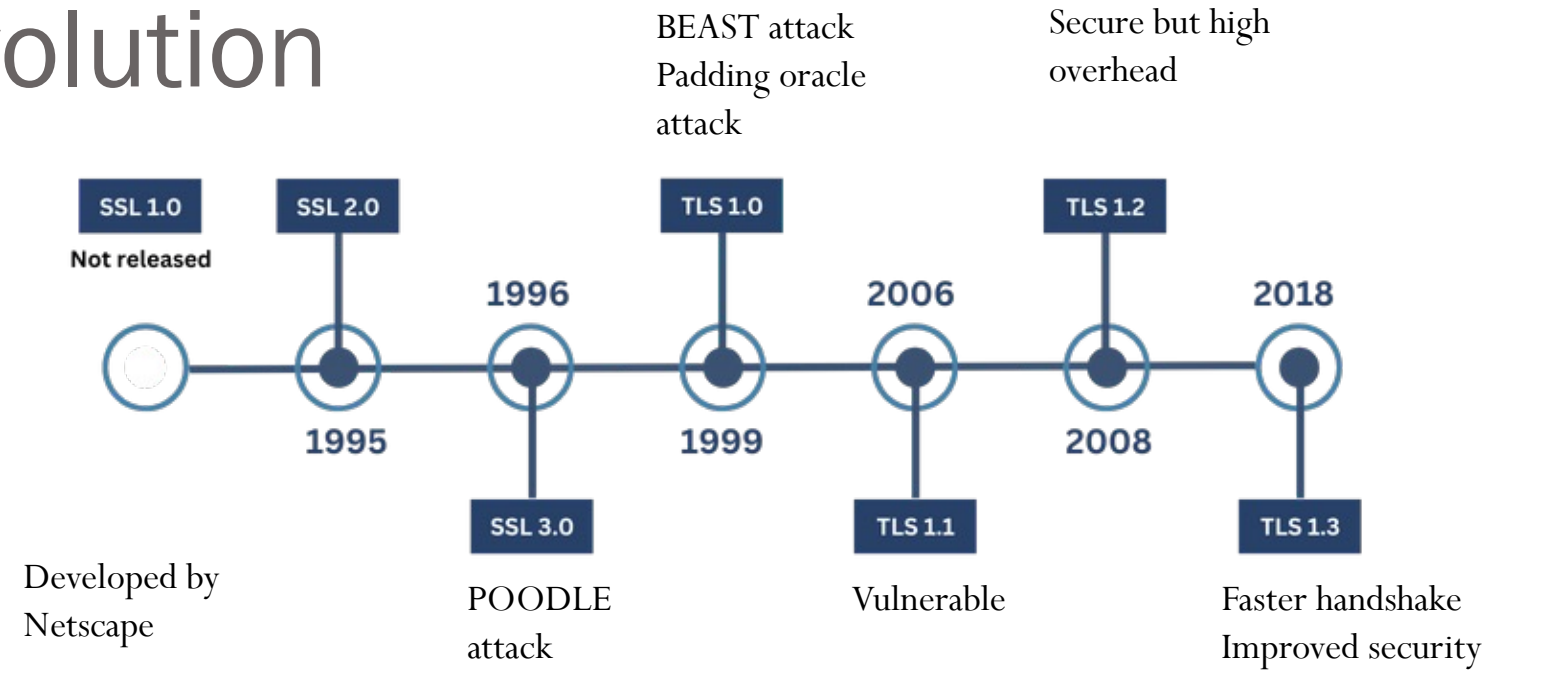
- Attacks and counter measures
- Security primer
- Security in different layers
  - Transport layer security – TLS
  - Network layer security – IPsec VPN
  - Data link layer security – 802.11i

Transport Layer Security (TLS)

- transport layer security to any TCP-based application
- Secure socket layer (SSL) is a predecessor of TLS. Often used equivalently.
- used between Web browsers, servers for e-commerce (https).
  - Port 443
  - TLS can be used for non-Web applications, e.g., IMAP.



Evolution

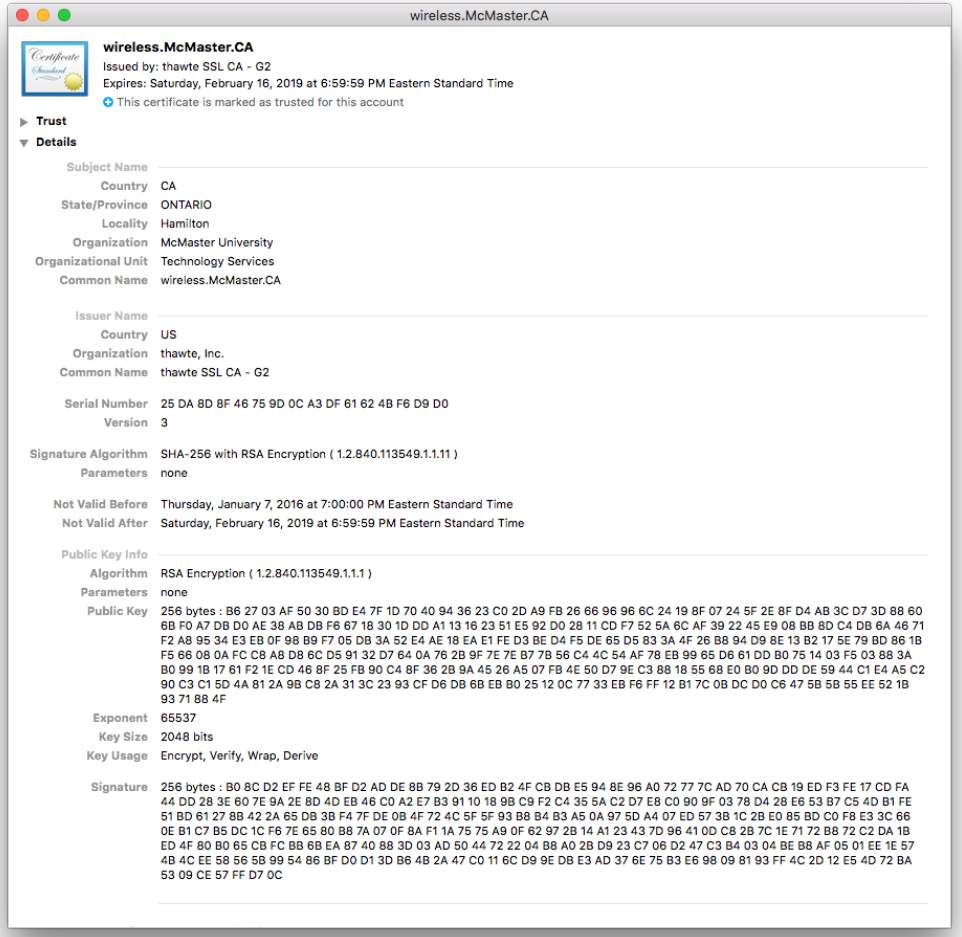


Source: https://scribbledtech.com/comprehensive_guide_to_tls/

- security services:
  - server authentication
  - data encryption
  - data integrity
  - client authentication (optional)

TLS (cont'd)

- server authentication:
  - TLS-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to verify server's public key from certificate.
- check your browser's security menu to see its trusted CAs or from local keychain



TLS (continued)

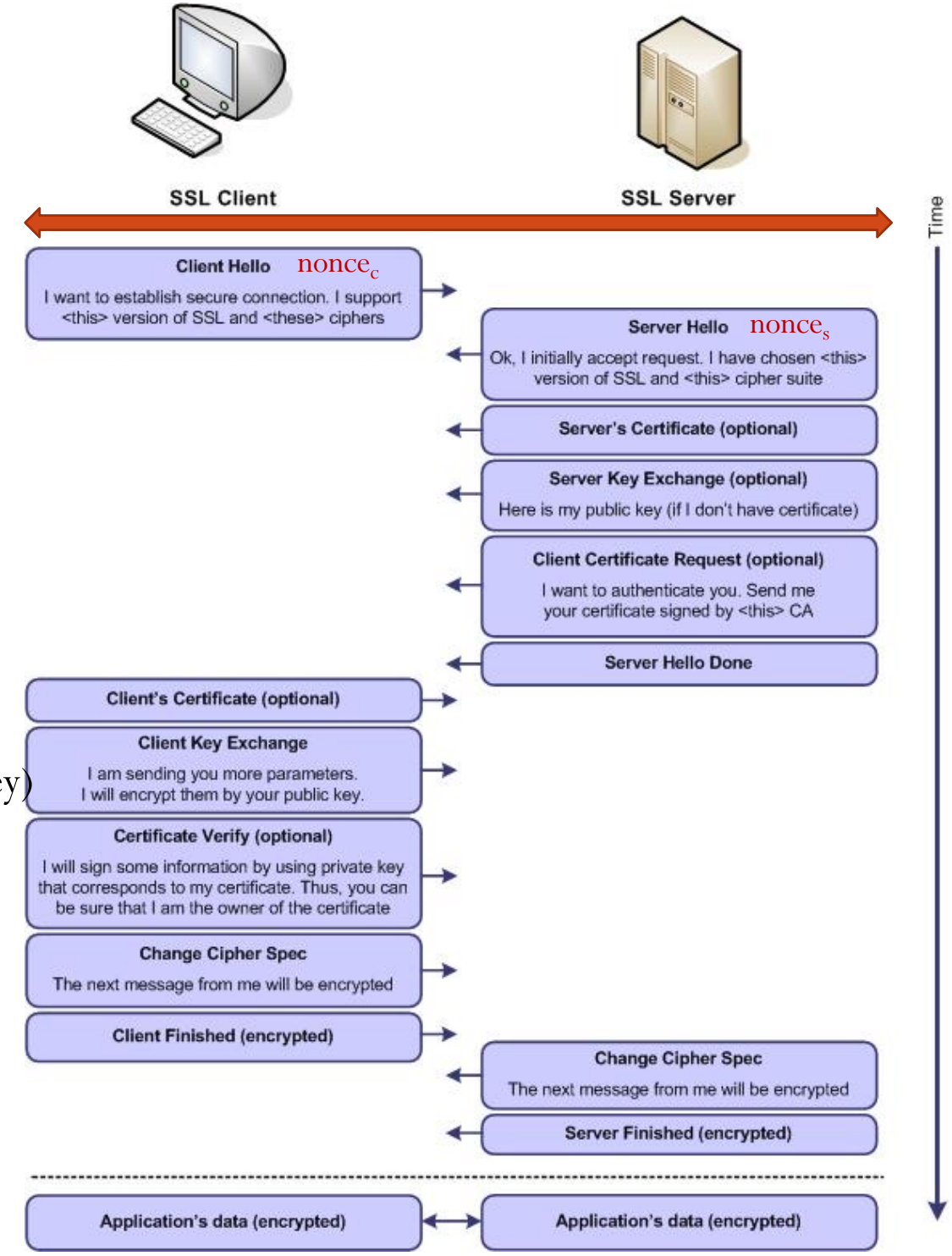
Encrypted TLS session:

- Generating symmetric session key depending on the key exchange method agreed
  - e.g., if RSA is used for key exchange,
    1. browser generates a *symmetric session key* locally, encrypts it with server's public key, sends encrypted key to server;
    2. using private key, server decrypts session key.
- Browser, server know the session key
  - All data sent into a TCP socket (by client or server) encrypted with session key.
- (optional) Client authentication can be done with client certificates.

TCP 3-way handshake

Handshake

K_S^+(Pre-master key)



Secure data transfer

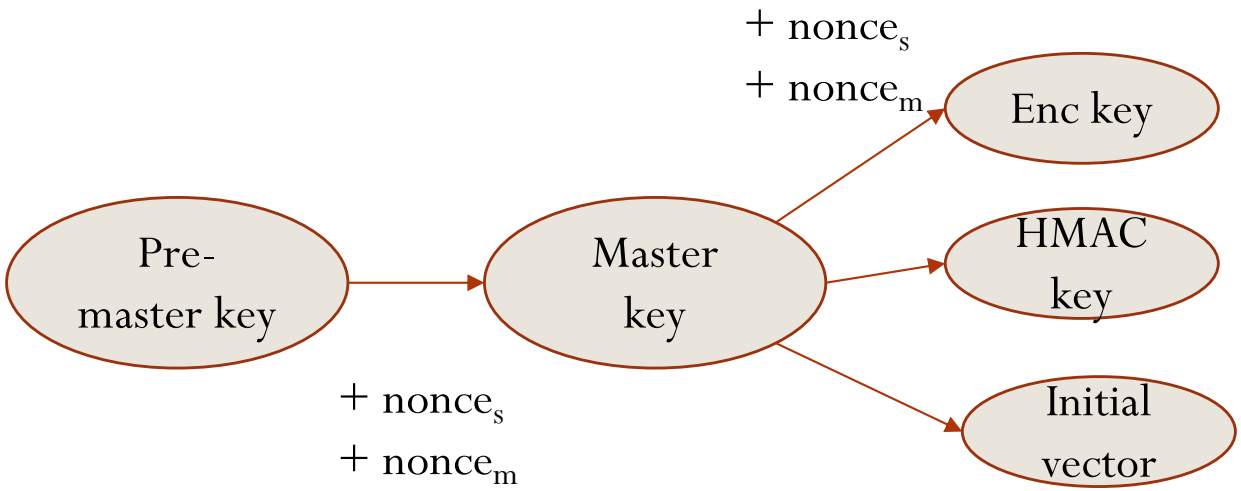
Key Exchange and Cipher Suite

Key exchange algorithm	Authentication	Cipher	Hash
Elliptic Curve Diffie-Hellman (ECDH) RSA Diffie-Hellman …	RSA DSA	3DES AES …	MD5 SHA …

Example:

- ECDHE_RSA_WITH_AES_256_GCM_SHA385
- RSA_WITH_AES_128_CBC_SHA256

Key Hierarchy



Identical for client and server for each connection

Master Key reusable between TCP connections

Session keys per session

Outline

- Attacks and counter measures
- Security primer
- Security protocols
  - Transport layer security – TLS
  - Network layer security – VPN
  - Data link layer security – 802.11i

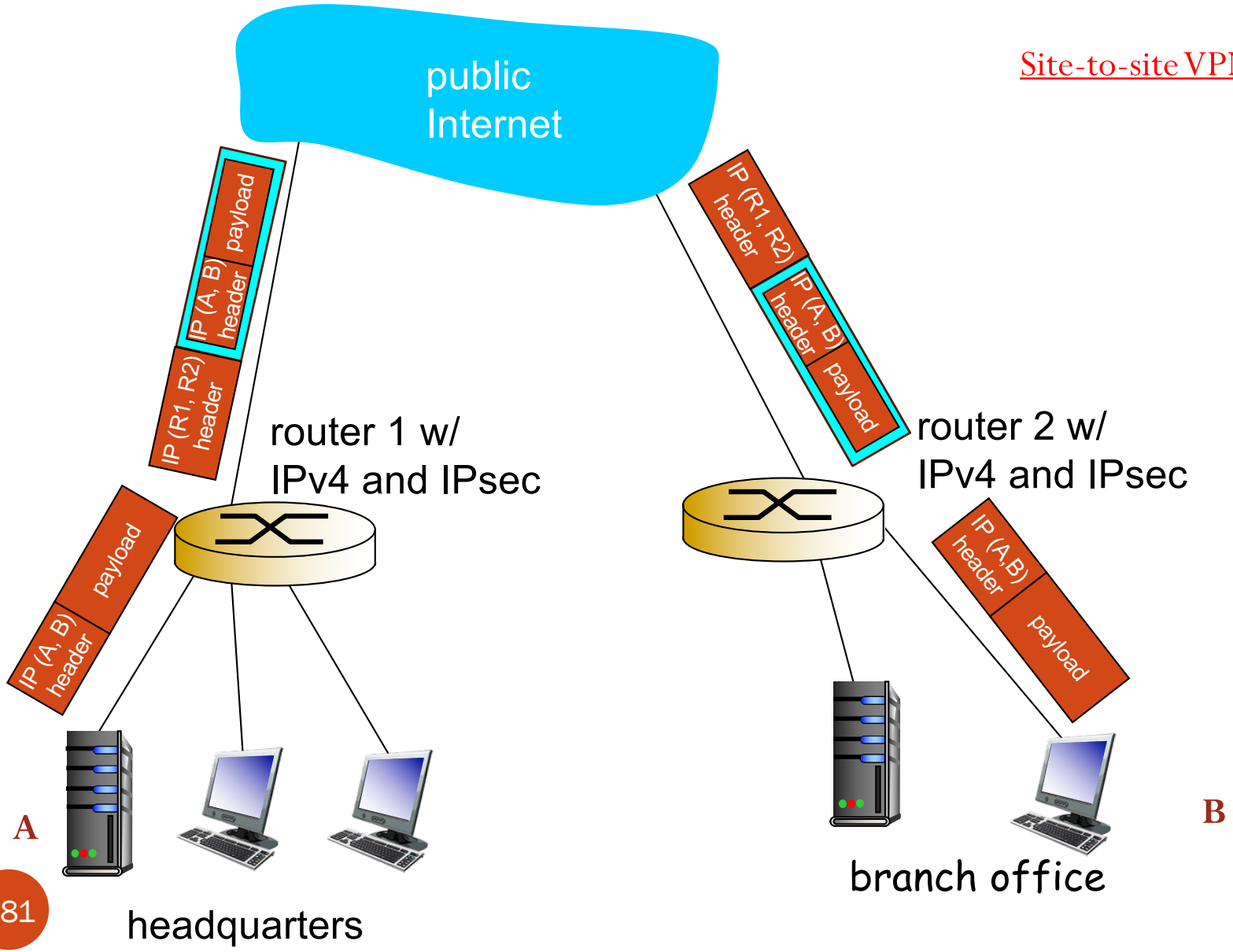
What is network-layer confidentiality ?

- between two network entities: sending entity encrypts datagram payload, payload could be:
  - Any IP datagrams including TCP or UDP segments, ICMP message, OSPF message ….
- all data sent from one entity to other would be hidden:
- "blanket coverage"

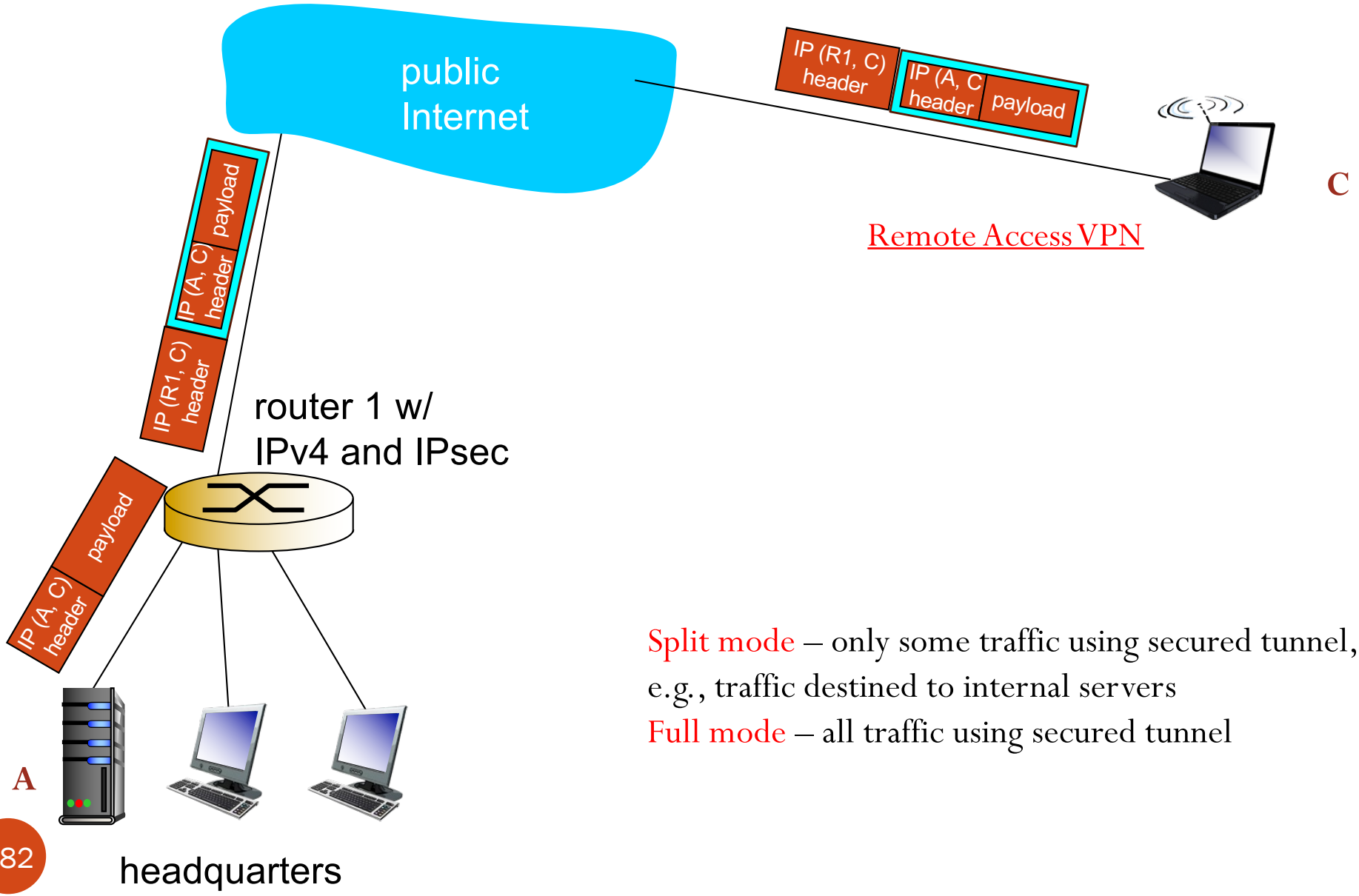
Virtual Private Networks (VPNs)

- institutions often want private networks for security.
  - costly: separate routers, links, DNS infrastructure.
- VPN: institution's inter-office traffic is sent over public Internet instead
  - encrypted before entering public Internet
  - logically separate from other traffic

Virtual Private Networks (VPNs)



Virtual Private Networks (VPNs)

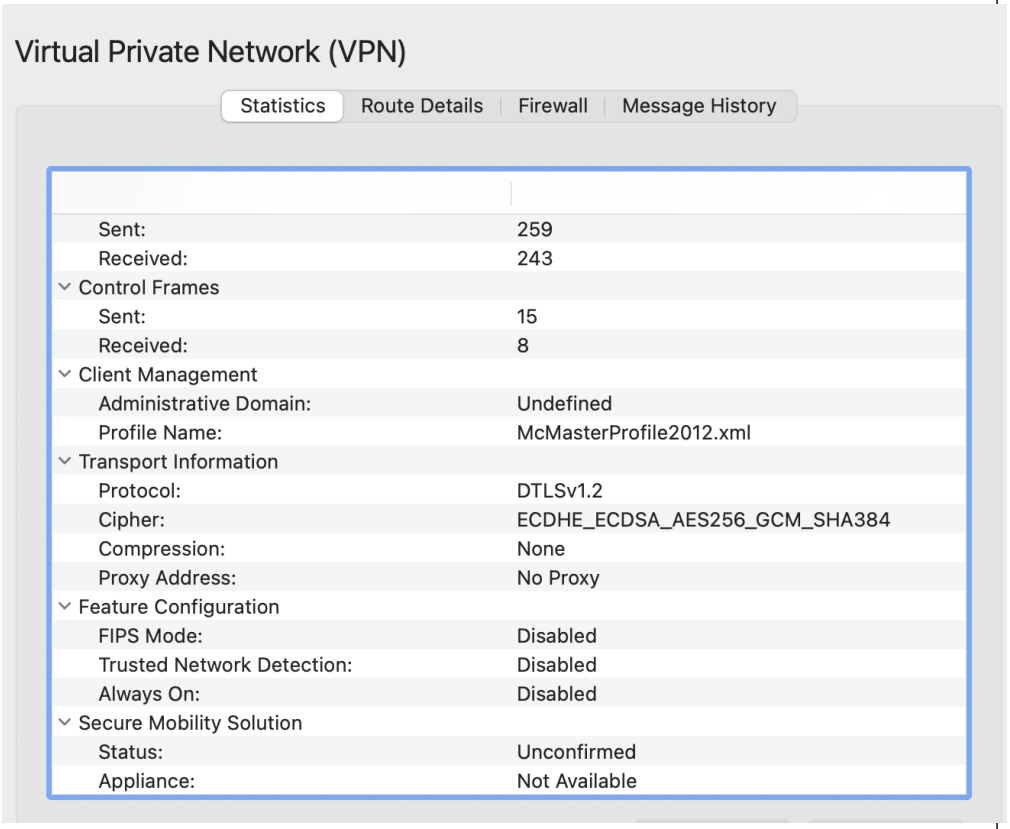


Split mode – only some traffic using secured tunnel, e.g., traffic destined to internal servers

Full mode – all traffic using secured tunnel

VPN Protocols

Protocol	"Layer" of encaputation	Firewall friendly
TLS/SSL VPN OpenVPN	Typically TCP port 443	Yes
WireGuard	UDP port 51820	No
IPSec	IP protocol 1701 for data	No



VPN services

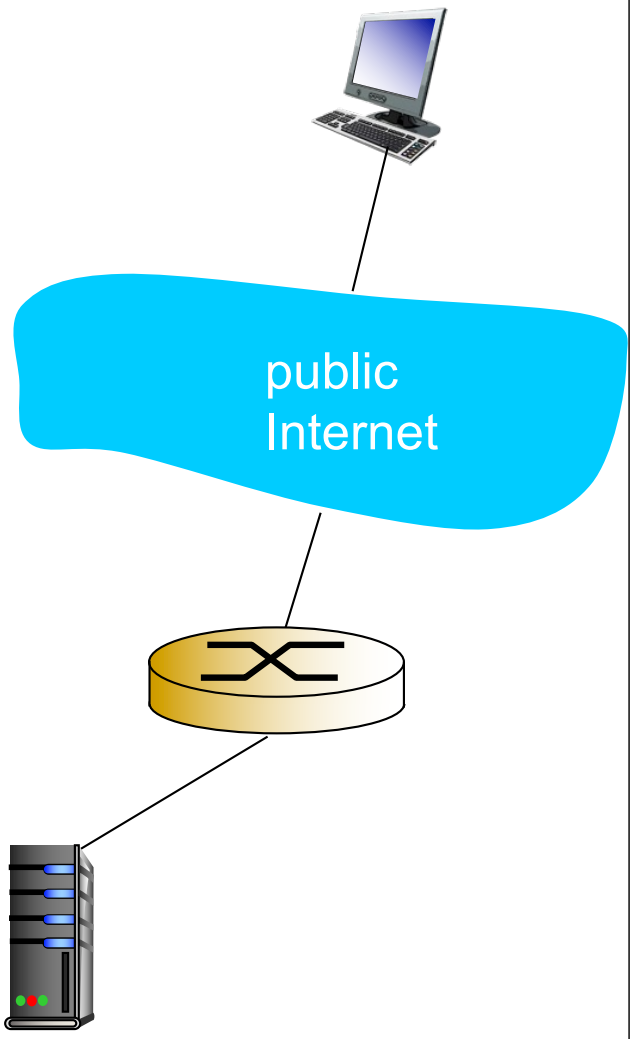
- confidentiality
- data integrity
- origin authentication
- replay attack prevention

TLS/SSL VPN

- Implementations: Cisco AnyConnect, OpenVPN
- Phase 1: establish a secured tunnel to a VPN server
  - Key exchange and derivation via TLS handshake
- Phase 2: encrypt and encapsulate an IP datagram in a TCP segment and send over a TCP socket to the VPN server, which forwards it to the destination

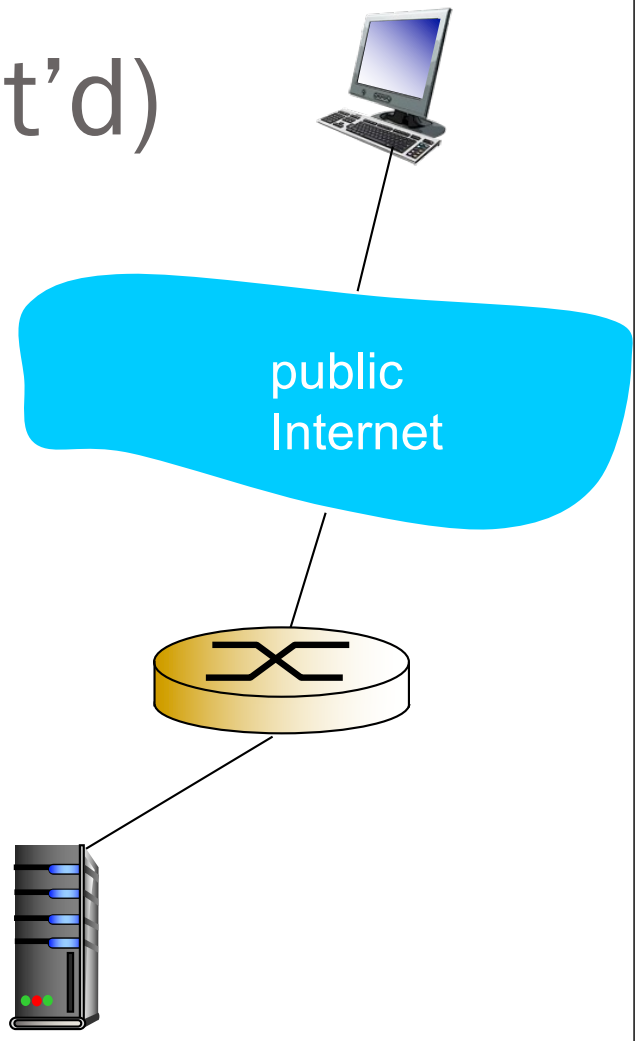
A TLS VPN Example

- Goal: ssh to a McMaster server 130.113.72.169 (hopper.cas.mcmaster.ca) from a laptop off campus
- Connect to campus VPN server 130.113.69.65 using Cisco AnyConnect
  - A new tun interface is created with IP address 172.18.205.252 on the laptop
  - Performs TLS handshake, negotiates keys (pre-master, master, session keys)
- A user ssh to hopper.cas.mcmaster.ca
  - 172.18.205.252 (tun) as source IP and 130.113.72.169 as destination IP, port 22
  - A TCP segment is sent (including SYN)
- The tun interface "intercepts" the IP datagram, passes it to the VPN client



A TLS VPN Example (Cont'd)

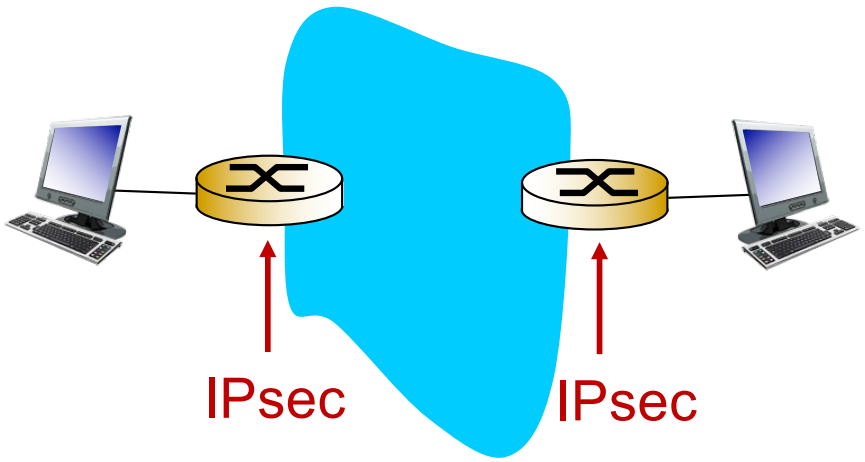
- The VPN client encrypts the IP datagram with TLS (and wrap in a TLS record) and sends over a TCP/UDP socket to the VPN server (dest IP 130.113.72.169)
  - The source IP address should be the client's public IP address (e.g., after NAT)
- VPN server receives TCP/UDP segments, extracts the original IP datagram and forwards to the ssh server
  - Note that source IP address is the IP of the tun interface (part of campus subnet)
- Responses from the ssh server will be routed through the VPN server, which encrypts them and sends via the VPN tunnel previously established with the laptop



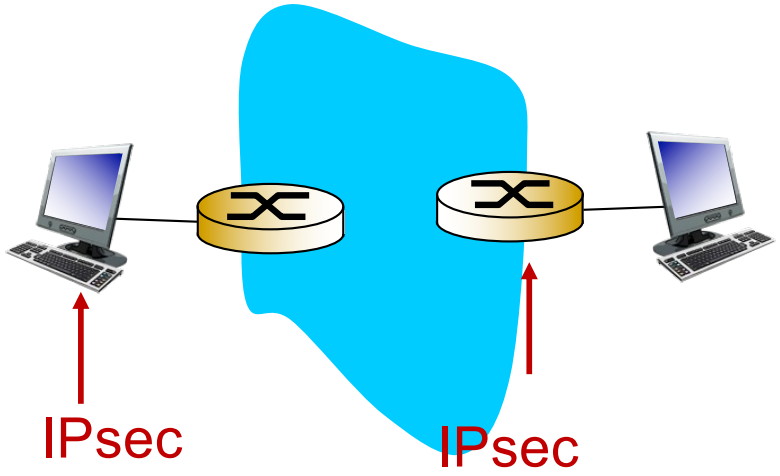
IPSec VPN

- Protocols:
  - IKE (internet key exchange): Negotiates security associations (SAs), algorithms, and keys
  - ESP (Encapsulating Security Payload): Provides encryption, integrity, and anti-replay protection
  - AH (Authentication Header): Provides integrity and authentication only

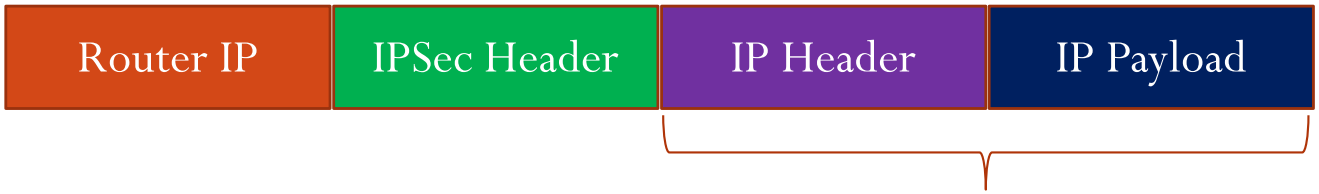
IPsec – tunneling mode



Site-to-site VPN

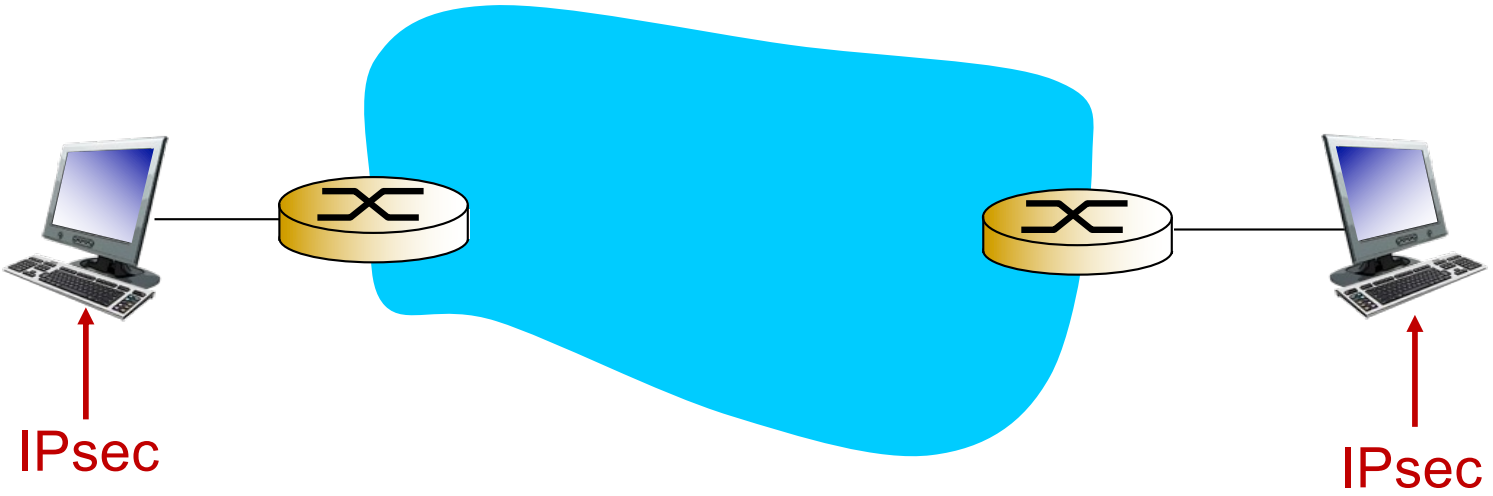


Remote Access VPN



IPsec transport mode

- IPsec datagram emitted and received by end-system
- protects upper level protocols



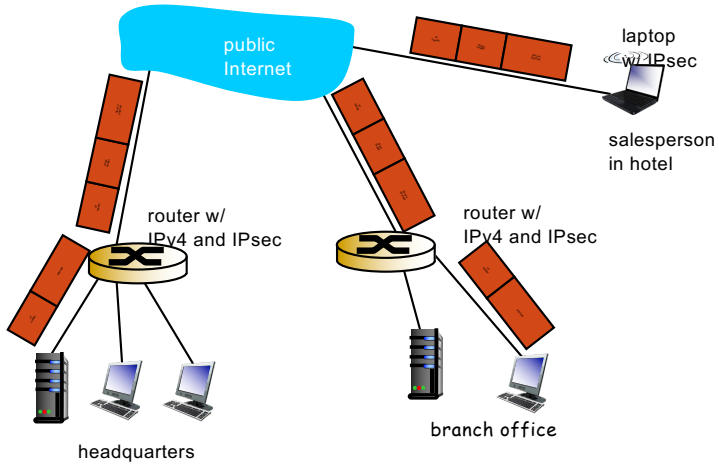
Modes and Protocols

- Authentication Header (AH) protocol
  - provides source authentication & data integrity but not confidentiality
- Encapsulation Security Protocol (ESP)
  - provides source authentication, data integrity, and confidentiality
  - more widely used than AH

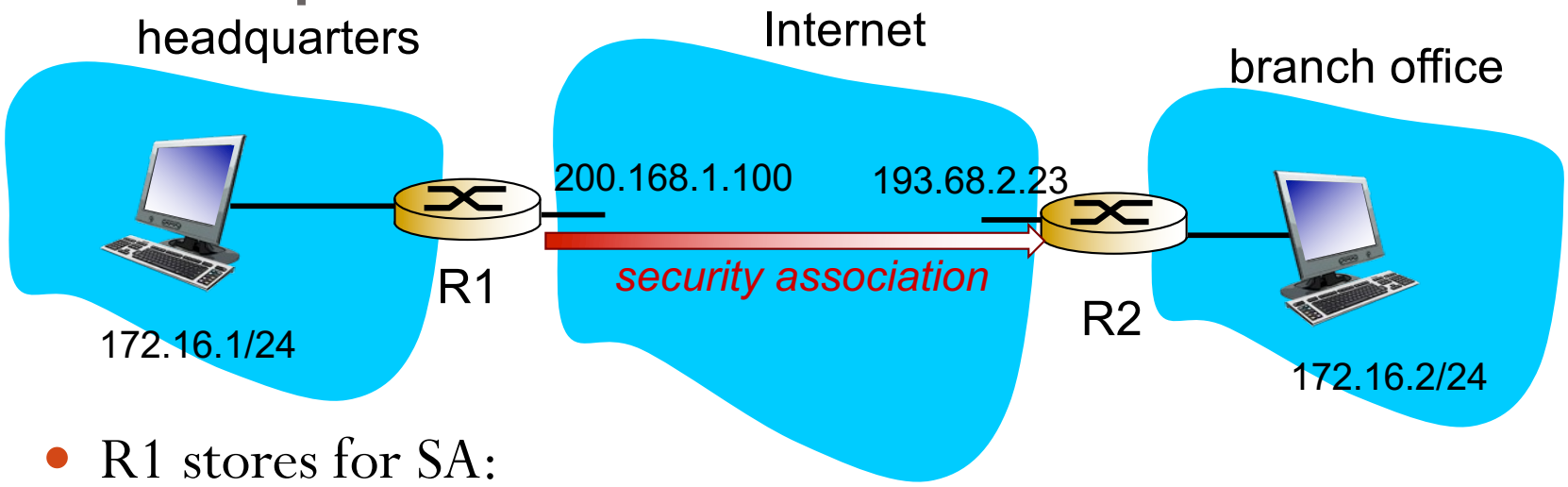
Transport mode with AH	Transport mode with ESP
Tunnel mode with AH	Tunnel mode with ESP

Security associations (SAs)

- before sending data, "security association (SA)" established from sending to receiving entity
  - SAs are simplex: for only one direction
- ending, receiving entitles maintain state information about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!

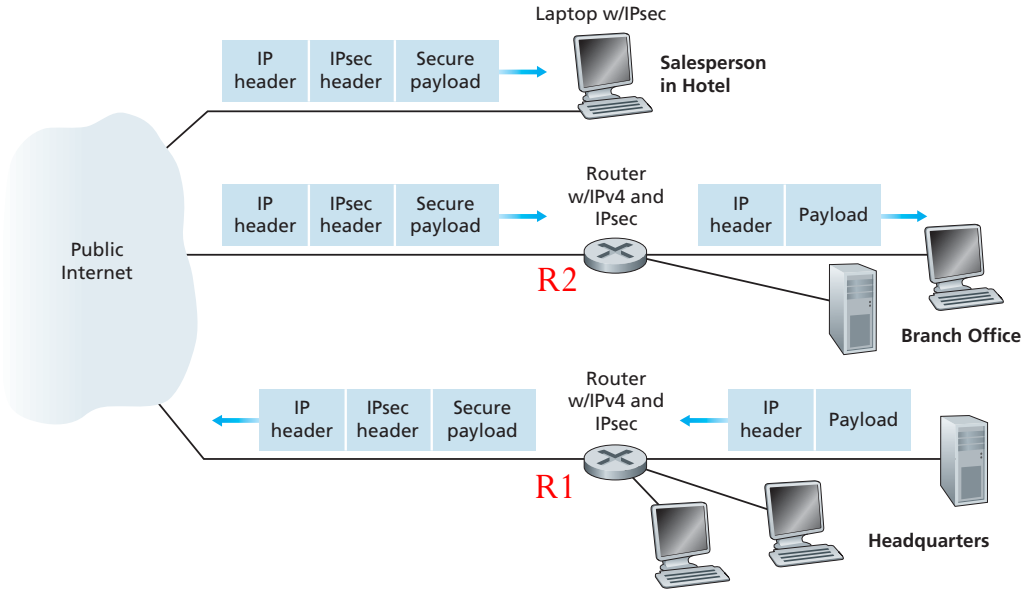


Example SA from R1 to R2



- R1 stores for SA:
  - 32-bit SA identifier: Security Parameter Index (SPI)
  - origin SA interface (200.168.1.100)
  - destination SA interface (193.68.2.23)
  - type of encryption used (e.g., 3DES with CBC)
  - encryption key
  - type of integrity check used (e.g., HMAC with MD5)
  - authentication key

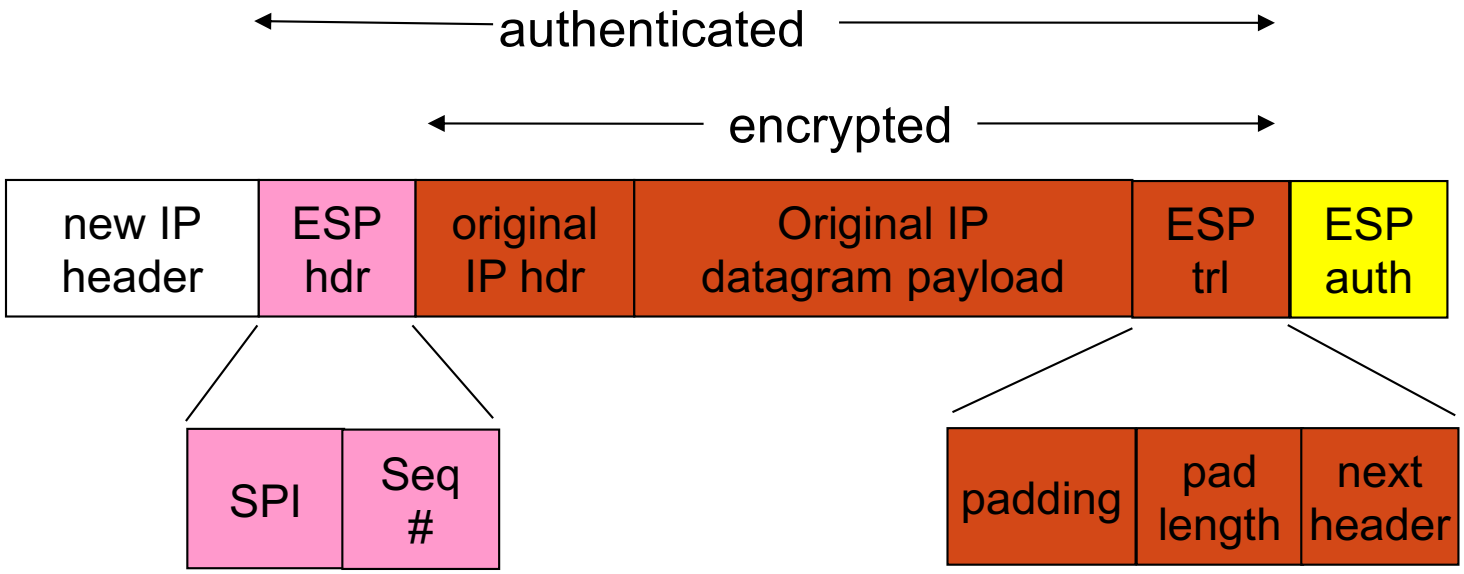
Security Association Database (SAD)



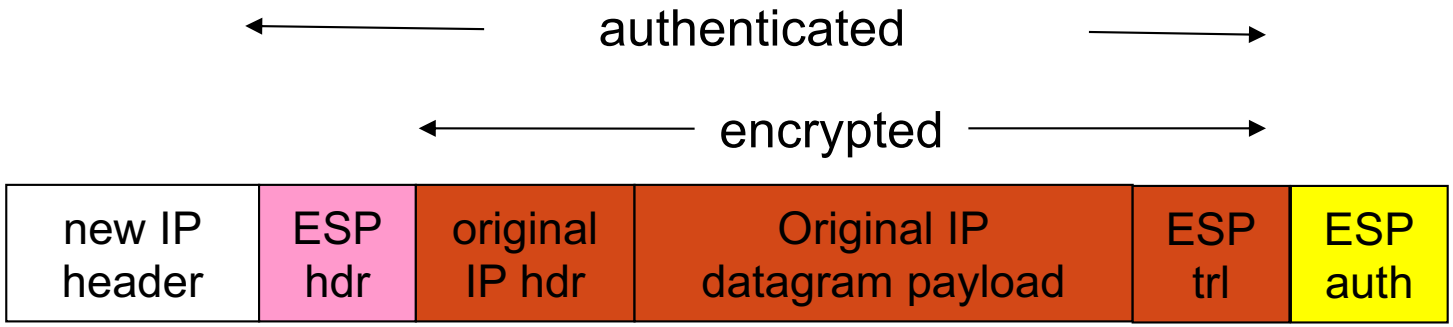
- endpoint holds SA state in security association database (SAD), where it can locate them during processing.
- One bi-directional IPsec traffic between headquarters and the branch office
  - 2 SAs
- One bi-directional IPsec traffic between headquarter and each salesperson
  - with n salespersons, 2n SAs in R1's SAD
- when sending IPsec datagram, R1 accesses SAD to determine how to process datagram.
- when IPsec datagram arrives to R2, R2 examines SPI in IPsec datagram, indexes SAD with SPI, and processes datagram accordingly.

IPsec datagram

- focus for now on tunnel mode with ESP

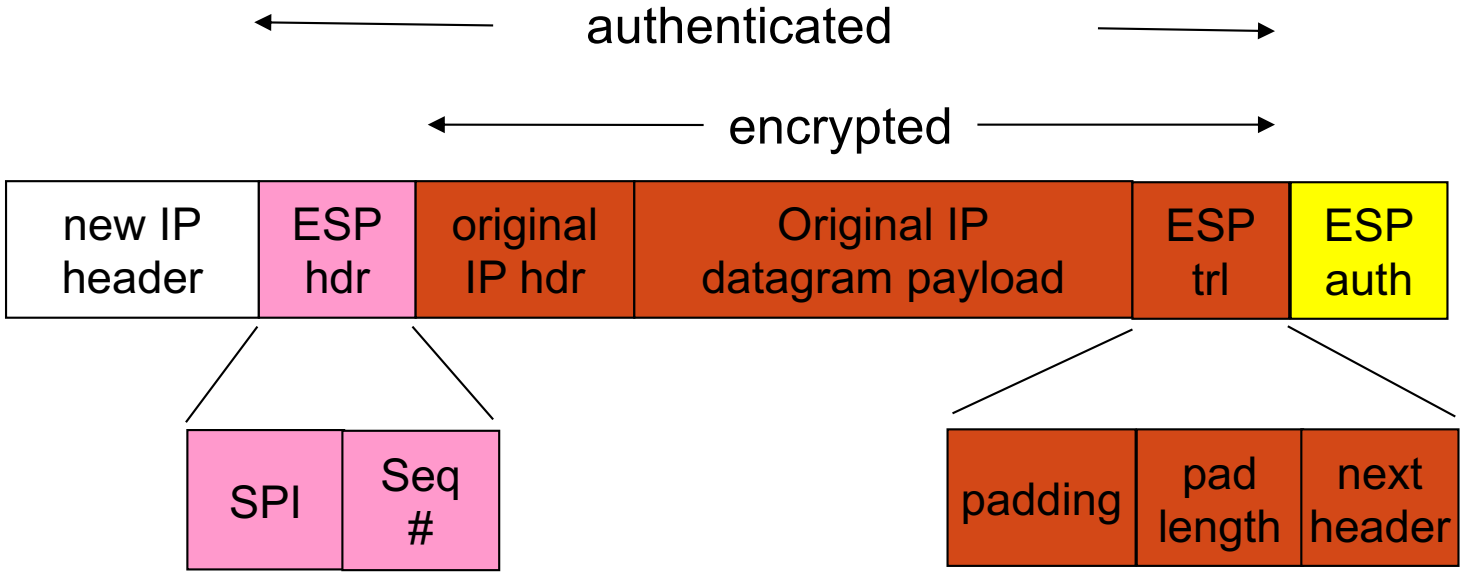


R1: convert original datagram to IPsec datagram



- appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- encrypts result using algorithm & key specified by SA.
- appends to front of this encrypted quantity the ESP header, creating "enchilada".
- creates authentication MAC over the whole enchilada, using algorithm and key specified in SA;
- appends MAC to back of enchilada, forming payload;
- creates brand new IP header, with all the classic IPv4 header fields, which it appends before payload.

Inside the encapsulation



- ESP trailer: Padding for block ciphers
- ESP header:
  - SPI, so receiving entity knows what to do
  - Sequence number, to thwart replay attacks
- MAC in ESP auth field is created with shared secret key

IPsec sequence numbers

- for a new SA, sender initializes seq. # to 0
- each time datagram is sent on SA:
  - sender increments seq # counter
  - places value in seq # field
- goal:
  - prevent attacker from sniffing and replaying a packet
  - receipt of duplicate, authenticated IP packets may disrupt service
- method:
  - destination checks for duplicates
  - doesn't keep track of all received packets; instead uses a window

Security Policy Database (SPD)

- policy: For a given datagram, sending entity needs to know if it should use IPsec – similar to firewall policies
- needs also to know which SA to use
  - may use: source and destination IP address; protocol number
- info in SPD indicates "what" to do with arriving datagram
- info in SAD indicates "how" to do it
- Example: based on the destination IP, SPD contains rules to discard, perform IPsec, or bypass IPsec processing

IPSec properties

- suppose Trudy sits somewhere between R1 and R2. she doesn't know the keys.
  - will Trudy be able to see original contents of datagram? How about source, dest IP address, transport protocol, application port?
  - flip bits without detection?
  - masquerade as R1 using R1's IP address?
  - replay a datagram?

IKE: Internet Key Exchange

- previous examples: manual establishment of IPsec SAs in IPsec endpoints:
  - Example SA
    - SPI: 12345
    - Source IP: 200.168.1.100
    - Dest IP: 193.68.2.23
    - Protocol: ESP
    - Encryption algorithm: 3DES-cbc
    - HMAC algorithm: MD5
    - Encryption key: 0x7aeaca…
    - HMAC key:0xc0291f…
- manual keying is impractical for VPN with 100s of endpoints
- instead use IPsec IKE (Internet Key Exchange)

IKE: PSK and PKI

- authentication (prove who you are) with either
  - pre-shared secret (PSK) or
  - with public key infrastructure (PKI).
- PSK: both sides start with secret
  - run IKE to authenticate each other and to generate IPsec SAs (one in each direction), including encryption, authentication keys
- PKI: both sides start with public/private key pair, certificate
  - run IKE to authenticate each other, obtain IPsec SAs (one in each direction).
  - similar with handshake in SSL.

IPsec summary

- Use Internet Key Exchange to derive keys and SPIs
- either AH or ESP protocol (or both)
  - AH provides integrity, source authentication
  - ESP protocol (with AH) additionally provides encryption
- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system

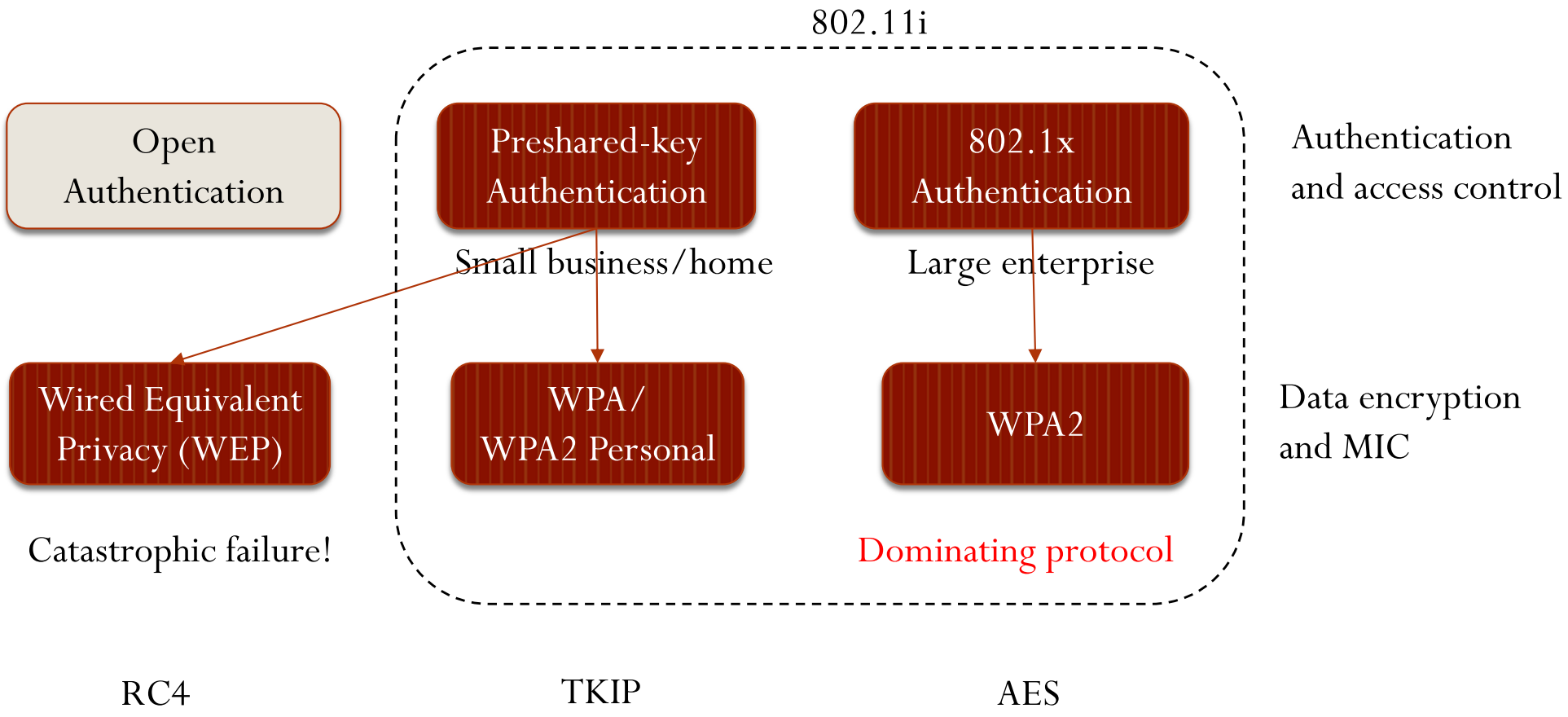
Outline

- Attacks and counter measures
- Security primer
- Security protocols
  - Transport layer security – TLS
  - Network layer security – TLS VPN and IPsec VPN
  - Data link layer security – 802.11i

IEEE 802.11 security

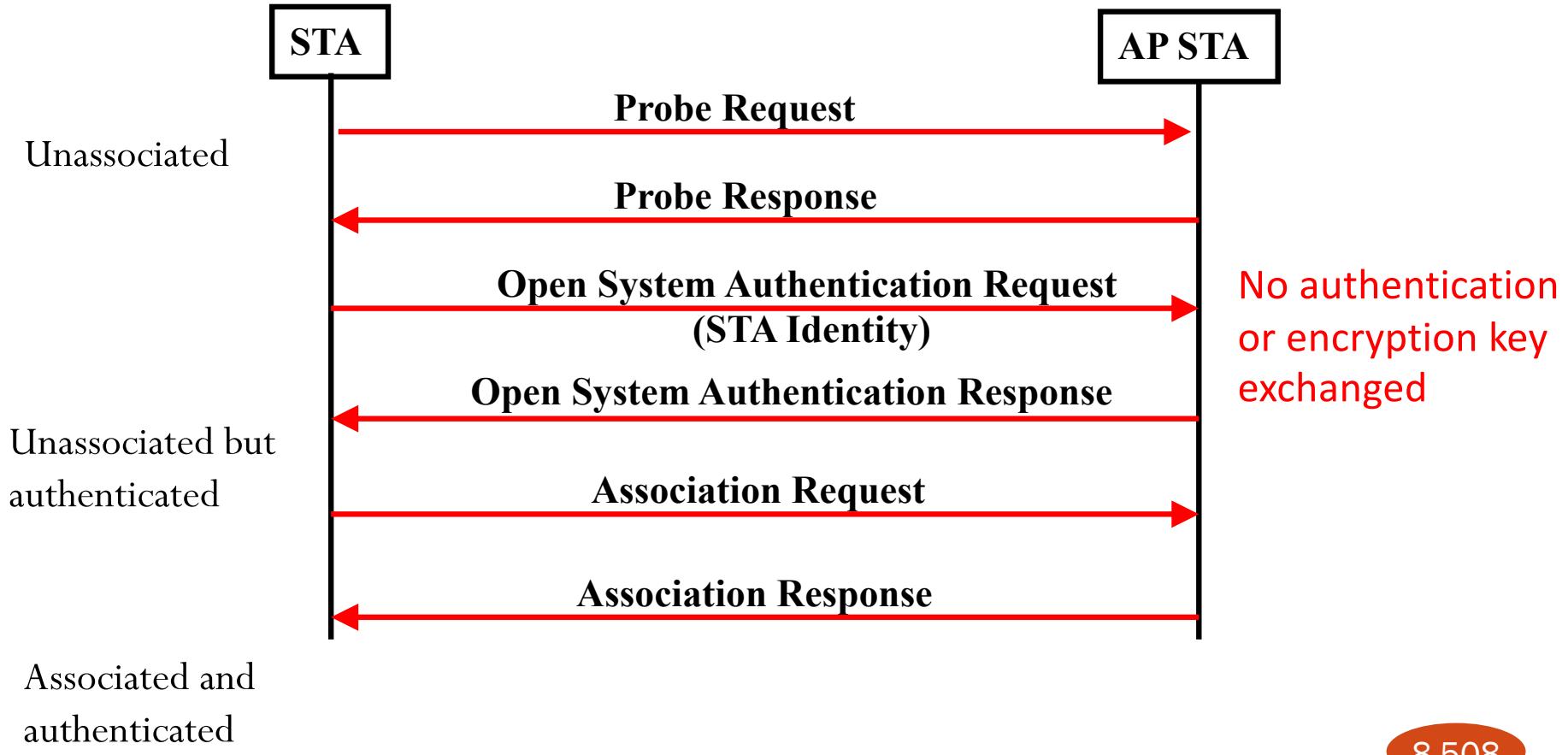
- *War-driving:* drive around your neighborhood, see what WiFi networks available?
  - Some uses no encryption/authentication
  - packet-sniffing and various attacks easy!
- Securing 802.11
  - encryption, authentication
  - first attempt at 802.11 security: Wired Equivalent Privacy (WEP): a failure
  - current attempt: 802.11i

802.11 Security Overview



Open System Authentication

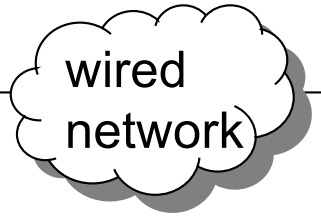
- Establishing the IEEE 802.11 association with no authentication

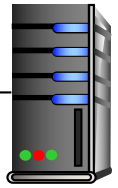


802.11i: four phases of operation

STA: client station

AP: access point





AS: Authentication server

1 Discovery of security capabilities

2 STA and AS mutually authenticate, together generate Master Key (MK). *AP serves as "pass through"*

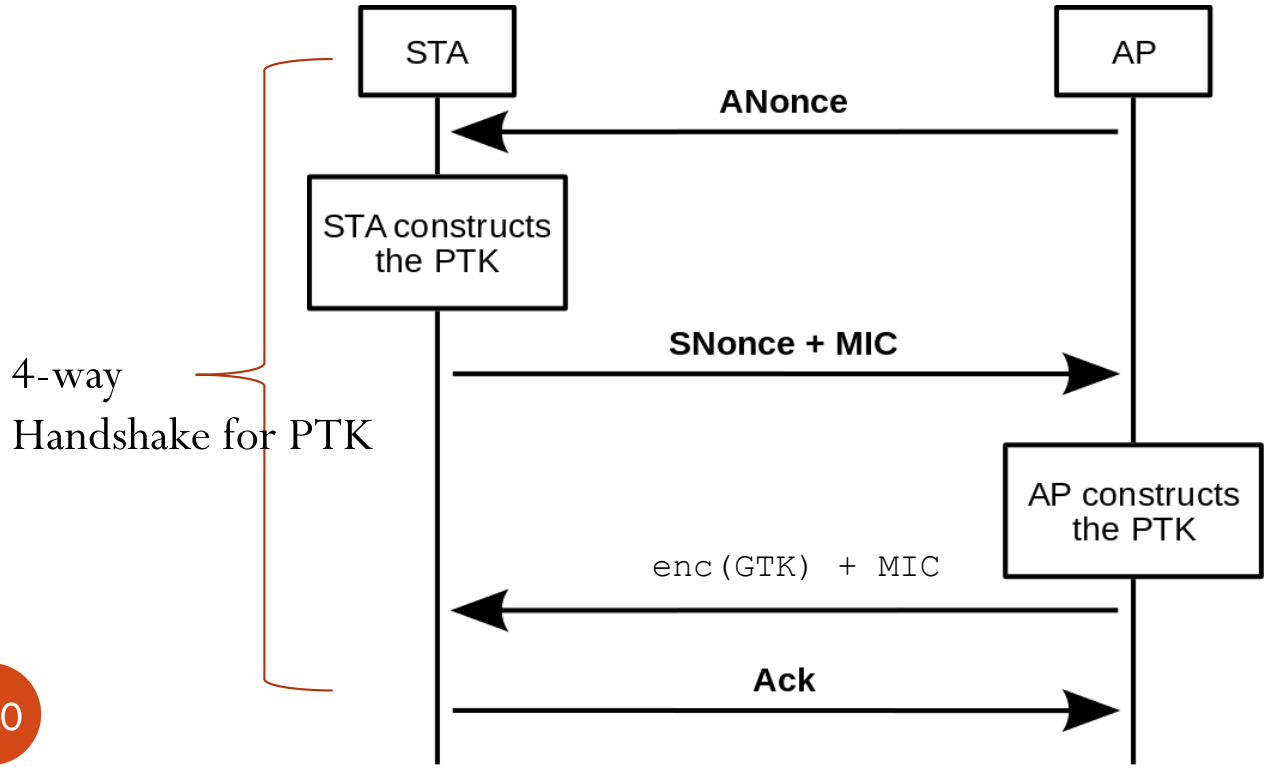
3 STA derives Pairwise Master Key (PMK)

3 AS derives same PMK, sends to AP

4 STA, AP use PMK to derive Temporal Key (PTK) used for message encryption, integrity

Pre-shared Key (PSK) Authentication

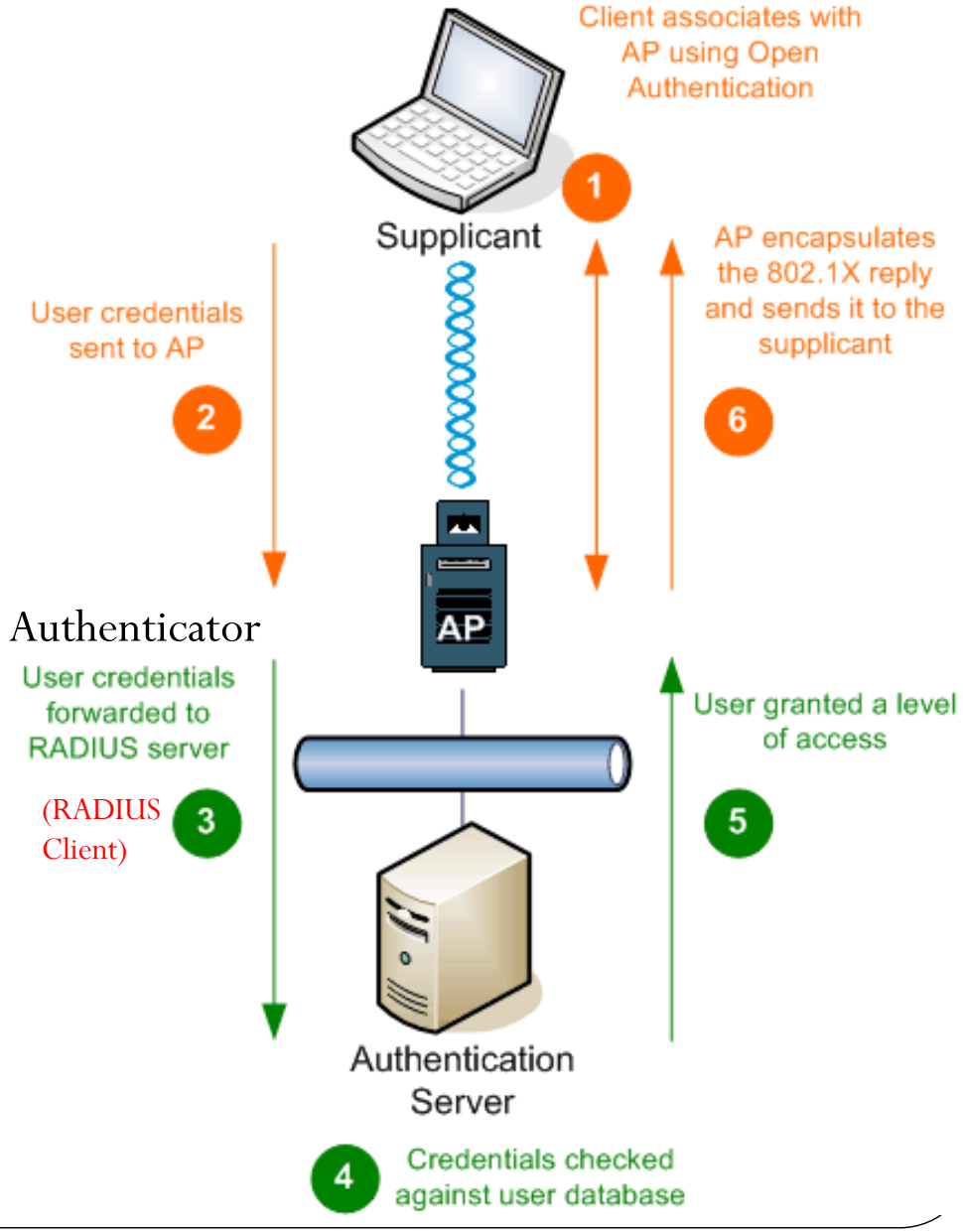
- AP co-locates with AS
- Uses a passphase (MK) to generate encryption key
- *PMK = PBKDF2(PassPhrase, ssid, ssidLength, 4096, 256)*
- *PTK = PRF512(*PMK, AMAC, SMAC, ANonce, SNonce)



PTK – Pairwise temporary key
MIC -- Message integrity check
GTK – Group template key

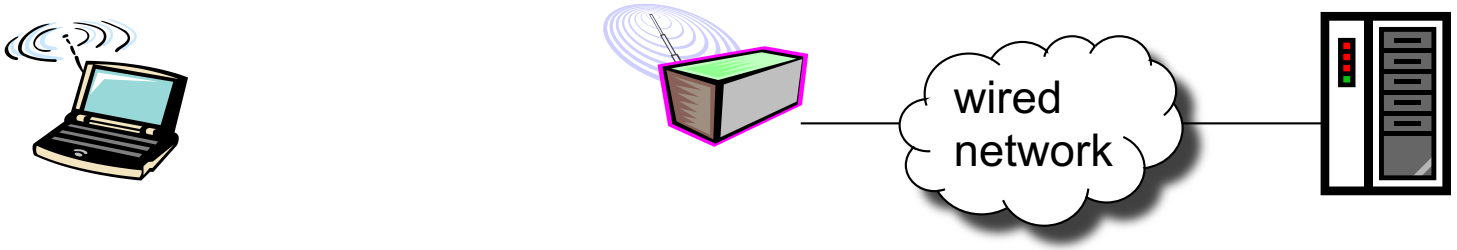
802.1x Authentication

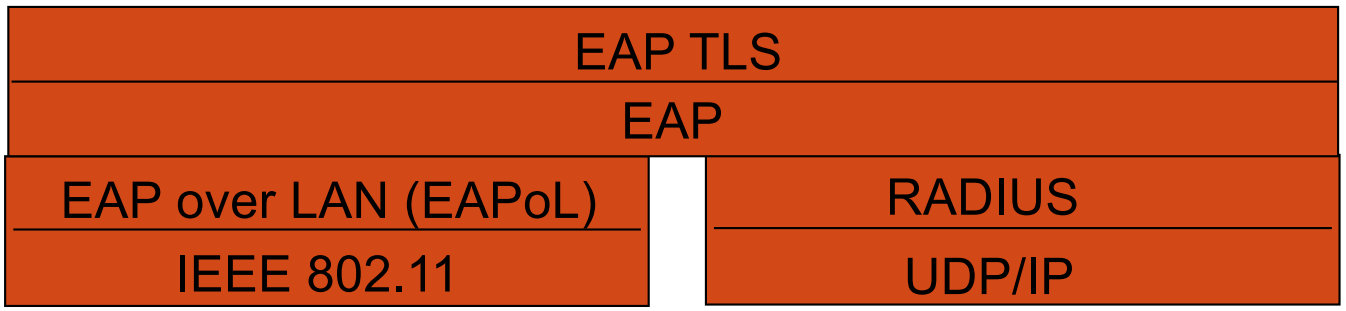
- An IEEE standard for port-based network access control
- Provide authentication for devices connected via LAN or WLAN
- RADIUS (Remote Authentication Dial-In User Service) implements the centralized backend authentication services



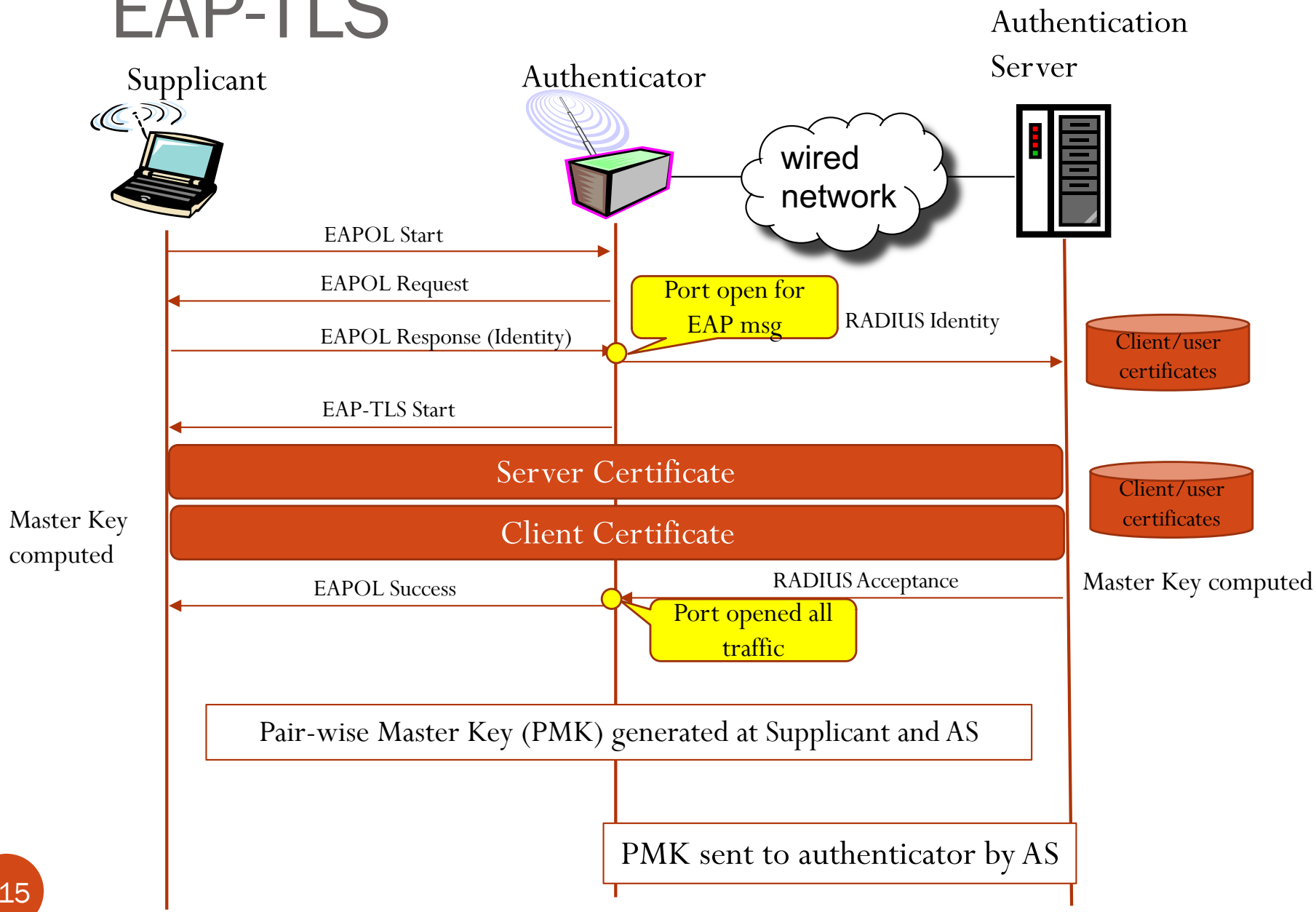
EAP: extensible authentication protocol

- EAP: end-end client (mobile) to authentication server protocol
  - Originally an extension of point-to-point protocol for dial-ups
- EAP sent over separate "links"
  - mobile-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)
- Support different authentication methods: MD5, TLS, PEAP ..



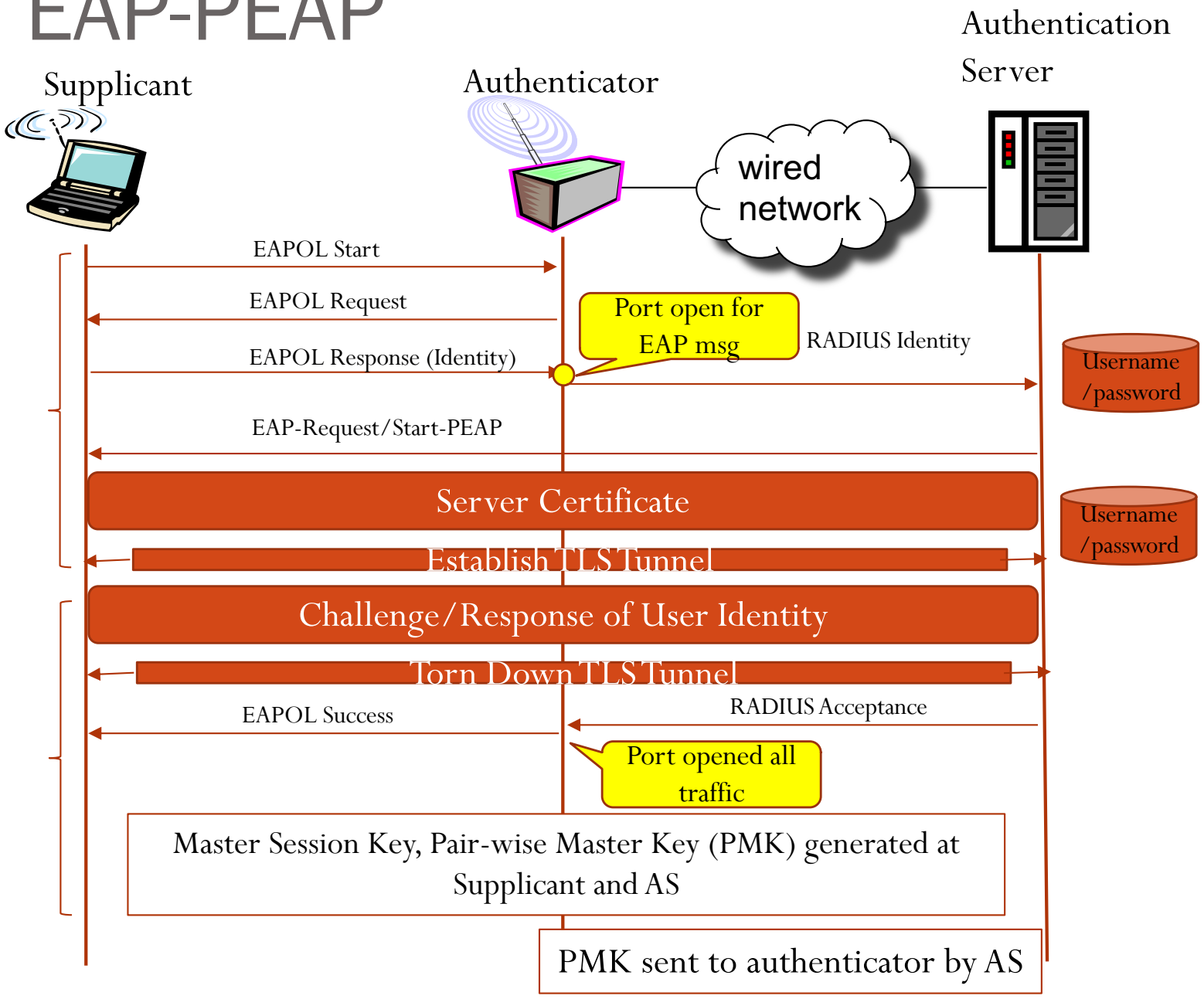


EAP-TLS



More at https://tools.ietf.org/html/rfc5216

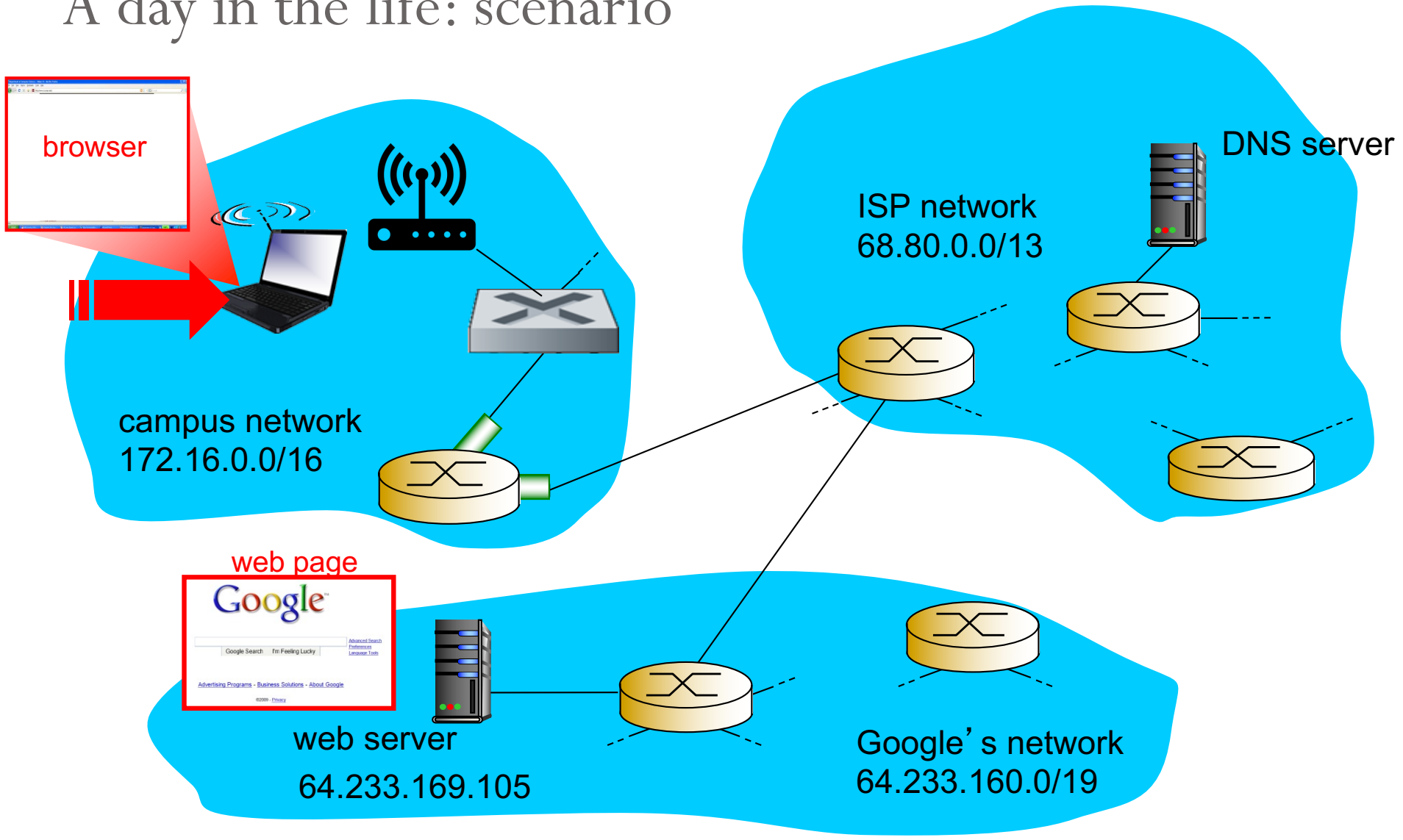
EAP-PEAP


Supplicant
Authenticator
Authentication Server
wired network
EAPOL Start
EAPOL Request
Port open for EAP msg
RADIUS Identity
EAPOL Response (Identity)
Username /password
Phase 1
EAP-Request/Start-PEAP
Server Certificate
Username /password
Establish TLS Tunnel
Challenge/Response of User Identity
Torn Down TLS Tunnel
RADIUS Acceptance
EAPOL Success
Phase 2
Port opened all traffic
Master Session Key, Pair-wise Master Key (PMK) generated at Supplicant and AS
PMK sent to authenticator by AS

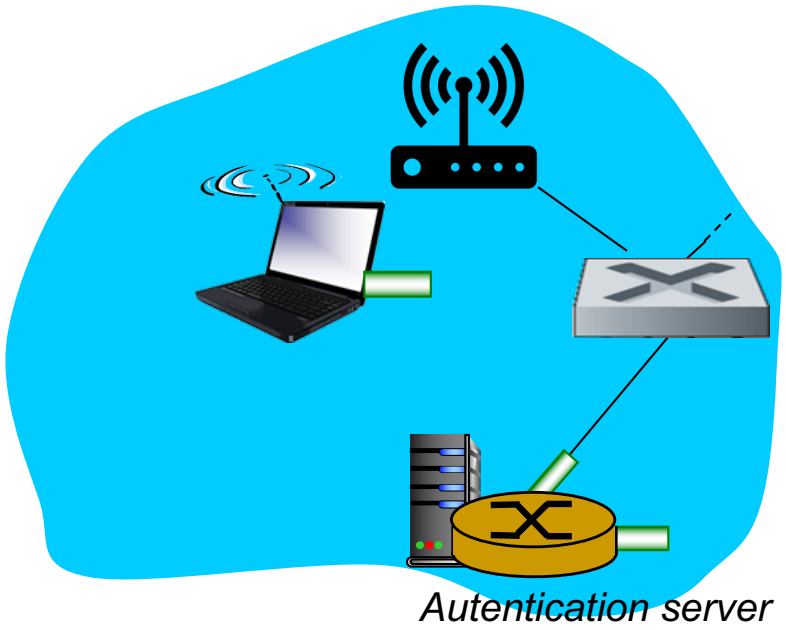
Revisit: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* one connects a laptop via Mac WiFi with WPA2, and requests/receives https://www.google.com

A day in the life: scenario



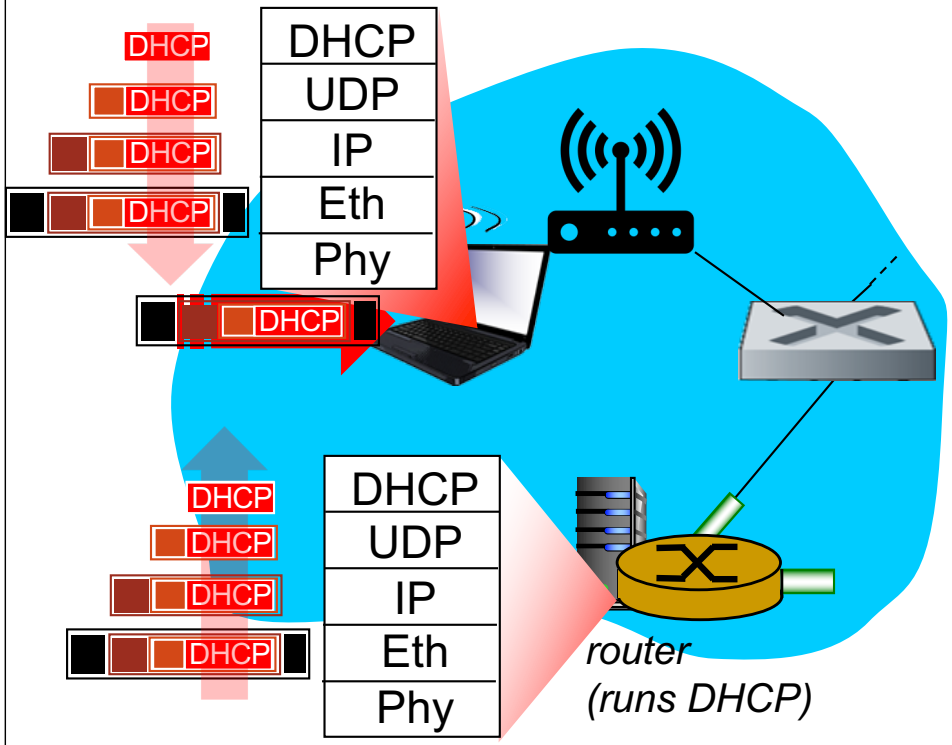
A day in the life… connecting to the Internet



Connect to Mac WiFi

- *Scan beacons from APs*
- *Open authentication*
- *Association request/response message (only 802.1x traffic allowed)*
- *802.1X/EAP authentication and derivation of keys*
- *Network access granted*

A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*

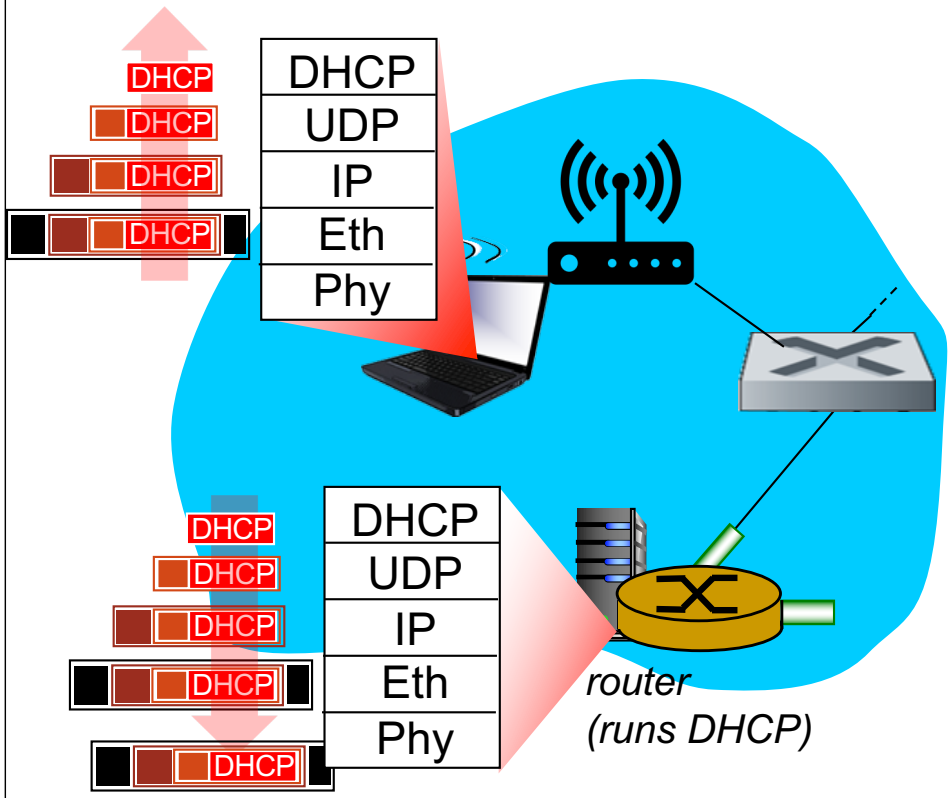
- DHCP request encapsulated in UDP, encapsulated in IP, encrypted in a 802.11 frame with dest MAC FFFFFFFFFFFF
- AP decrypts the 802.11 frame,

and AP converts 802.11 frame to 802.3 frame

- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running a DHCP server
- (Switch broadcast the frame to all LAN segments)

- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

A day in the life… connecting to the Internet



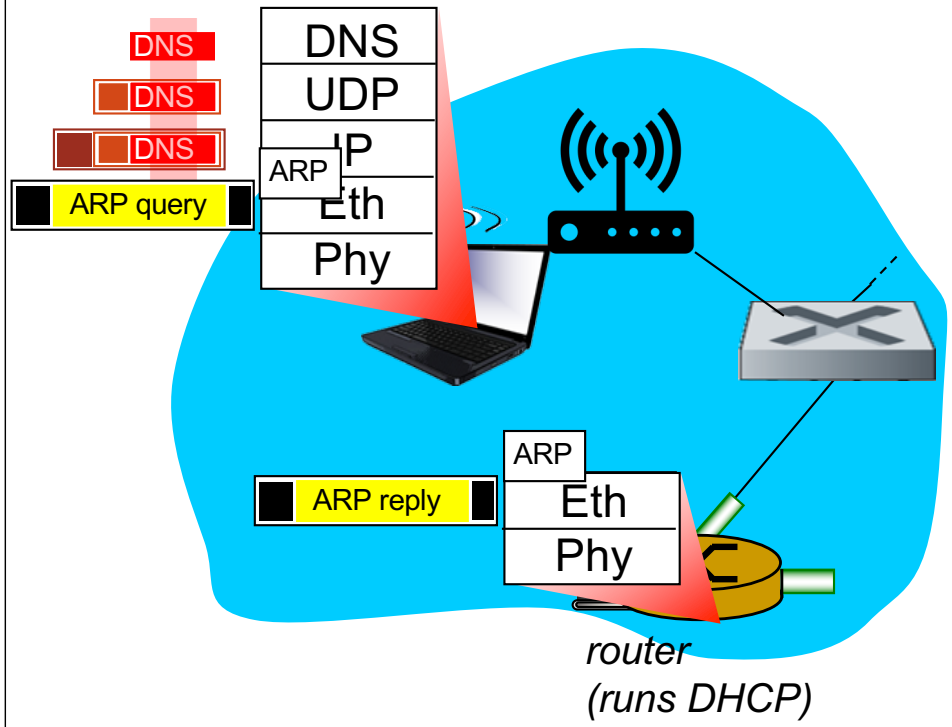
- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client

- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



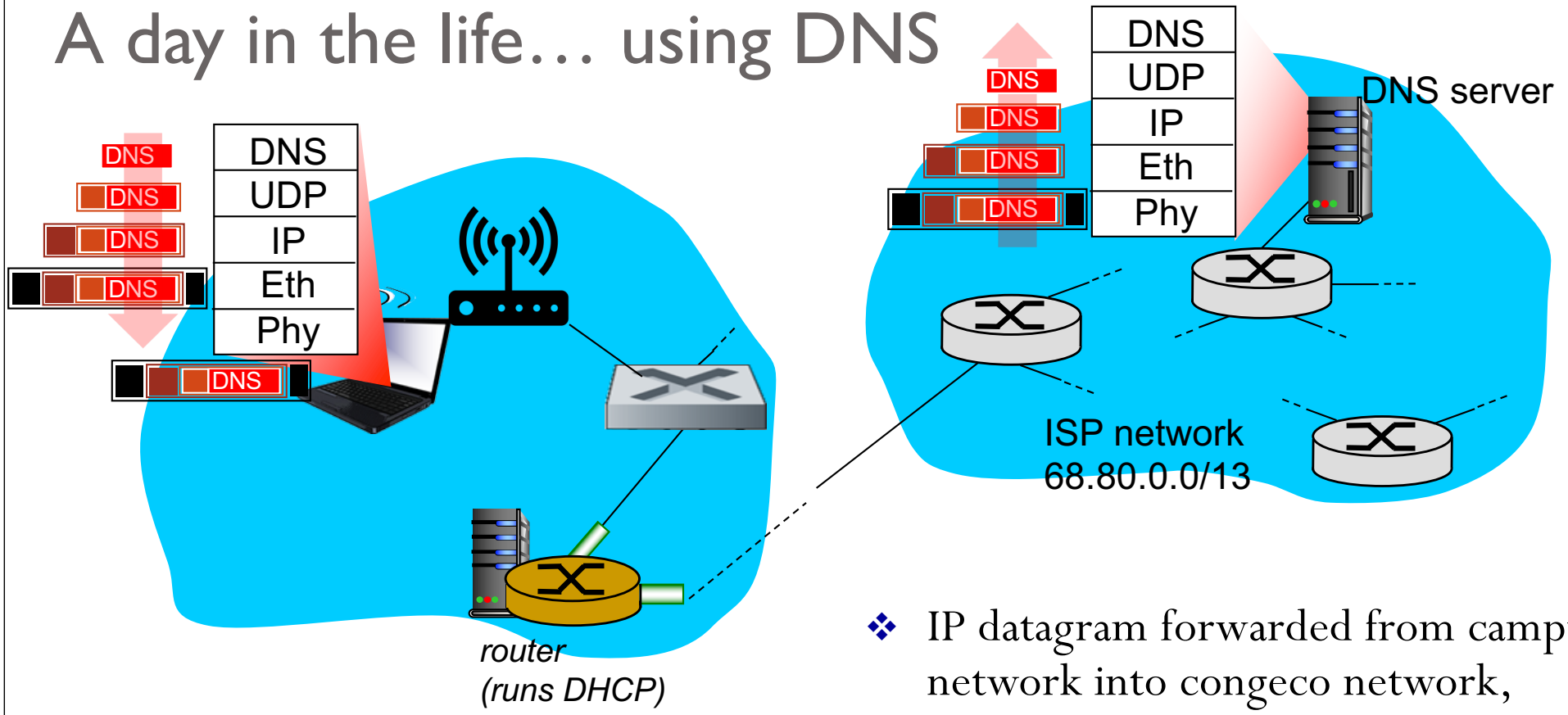
- before sending *HTTPs* request, need IP address of www.google.com: *DNS*

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP

- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface

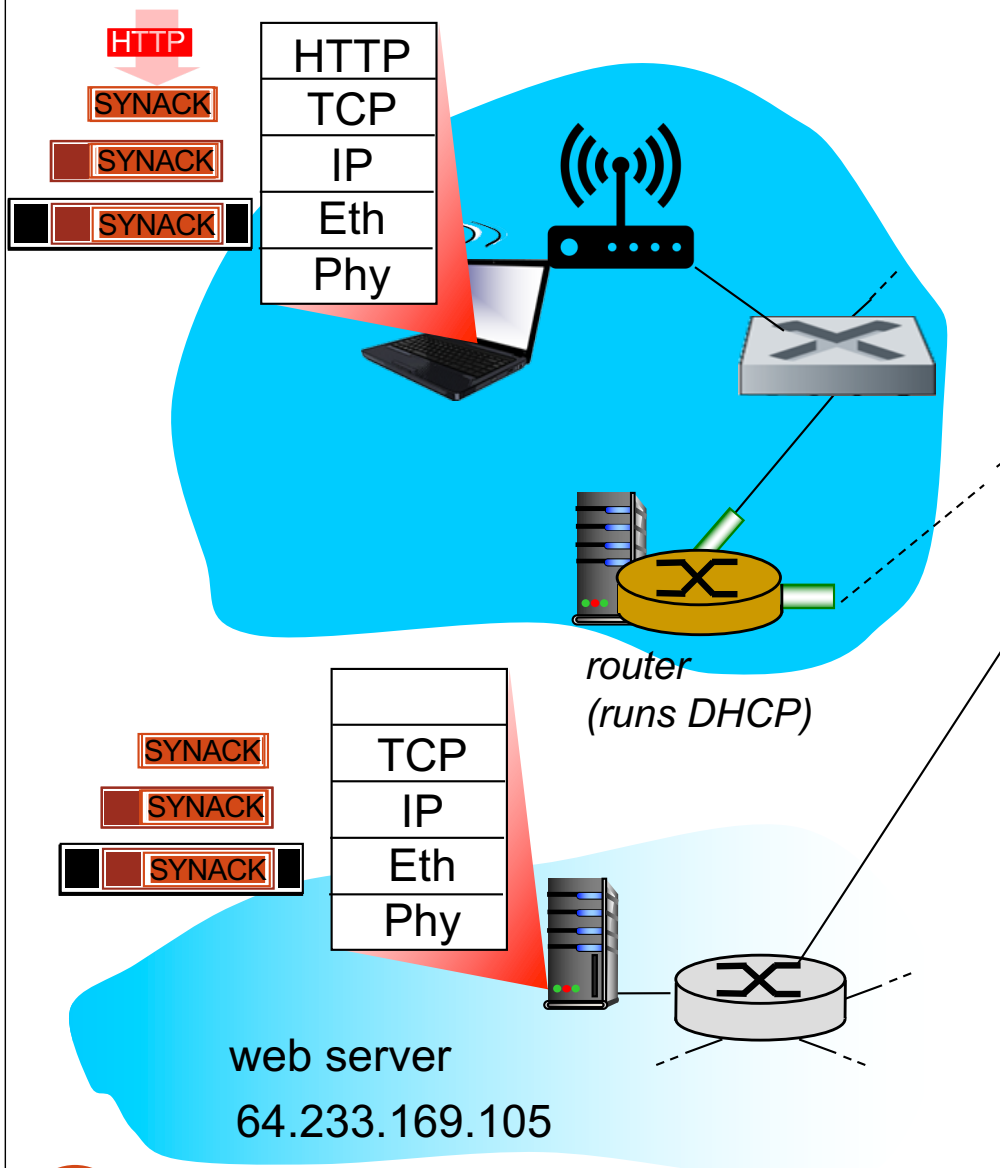
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS



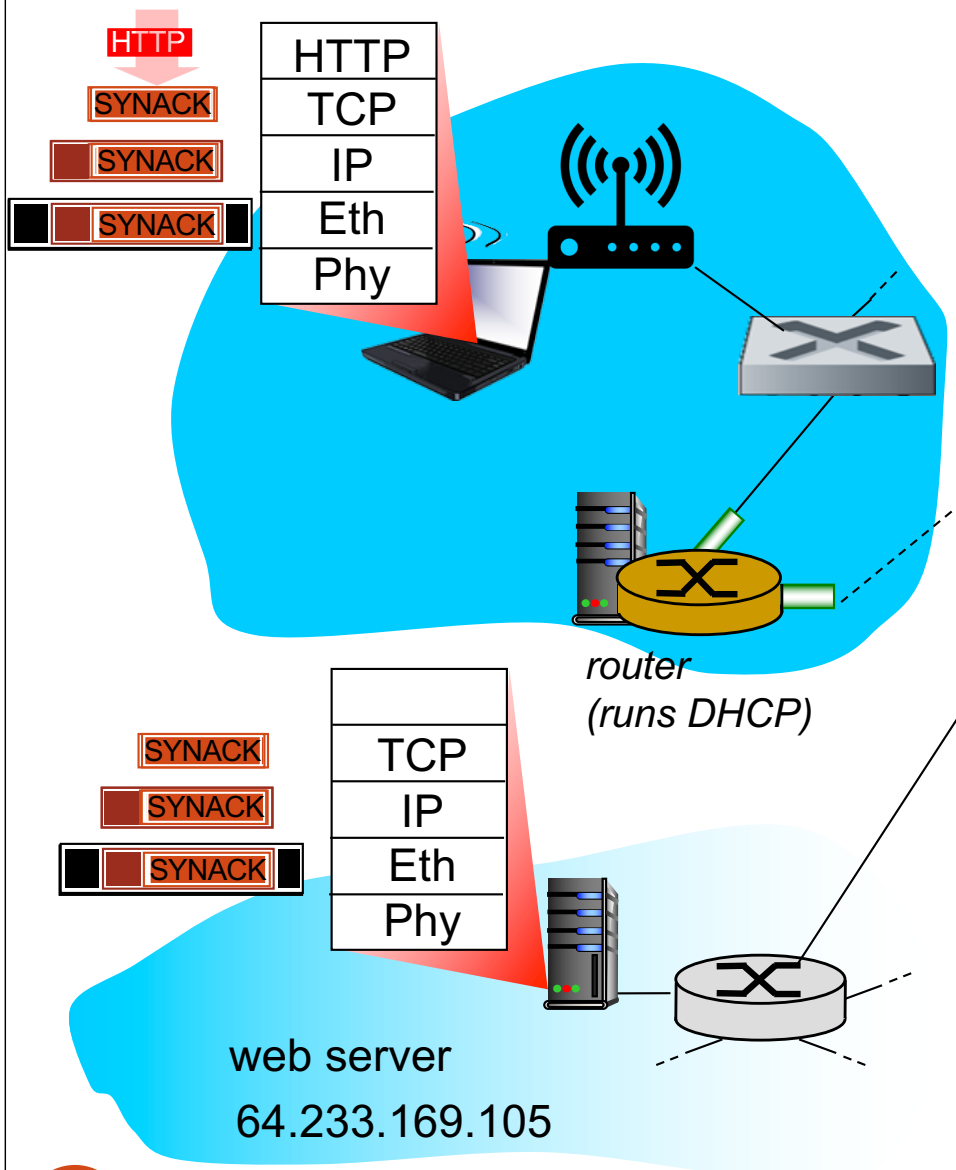
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into congeco network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



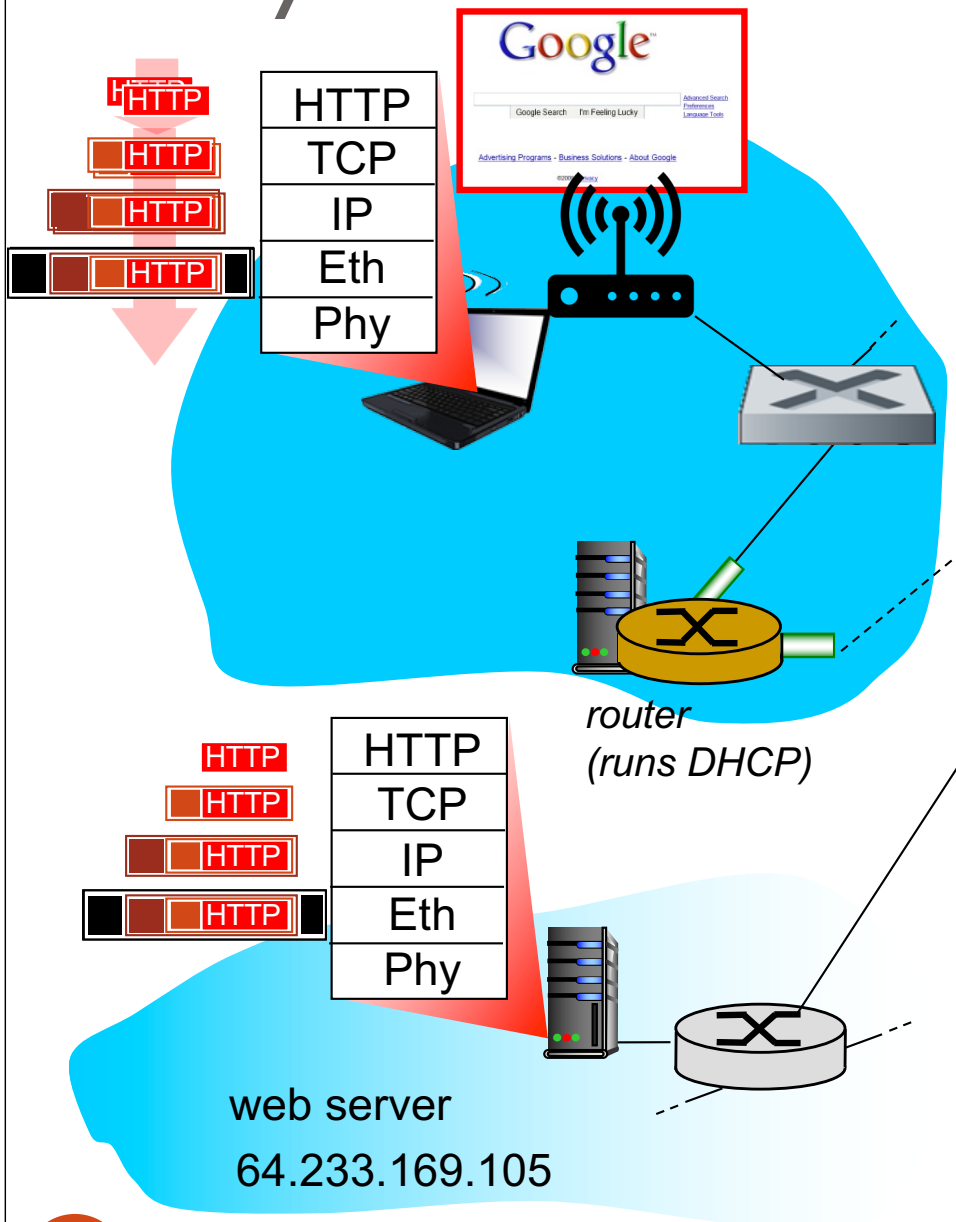
- to send HTTPs request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

A day in the life…TCP connection carrying HTTP



- TLS handshake between the client and server
  - Authenticate server via server certificate
  - Establish session keys for data encryption, MAC and IV

A day in the life… HTTP request/reply



- Encrypted HTTP request sent into TCP socket,
- IP datagram routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing encrypted HTTP reply routed back to client