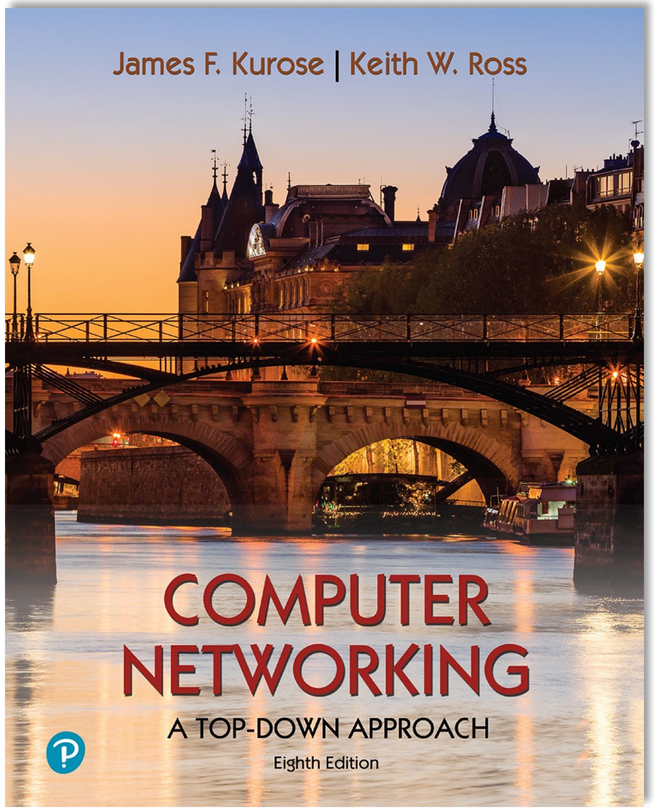
Network Security



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Acknowledgements
These Slides have been adapted from the originals made available by J. Kurose and K. Ross

Goals

- Understand principles of network security
  - confidentiality
  - authentication
  - message integrity
  - digital signature
- Security in practice
  - security in application, transport, network, link layers
  - firewalls and intrusion detection systems

Roadmap

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Operational security: firewalls and IDS

What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

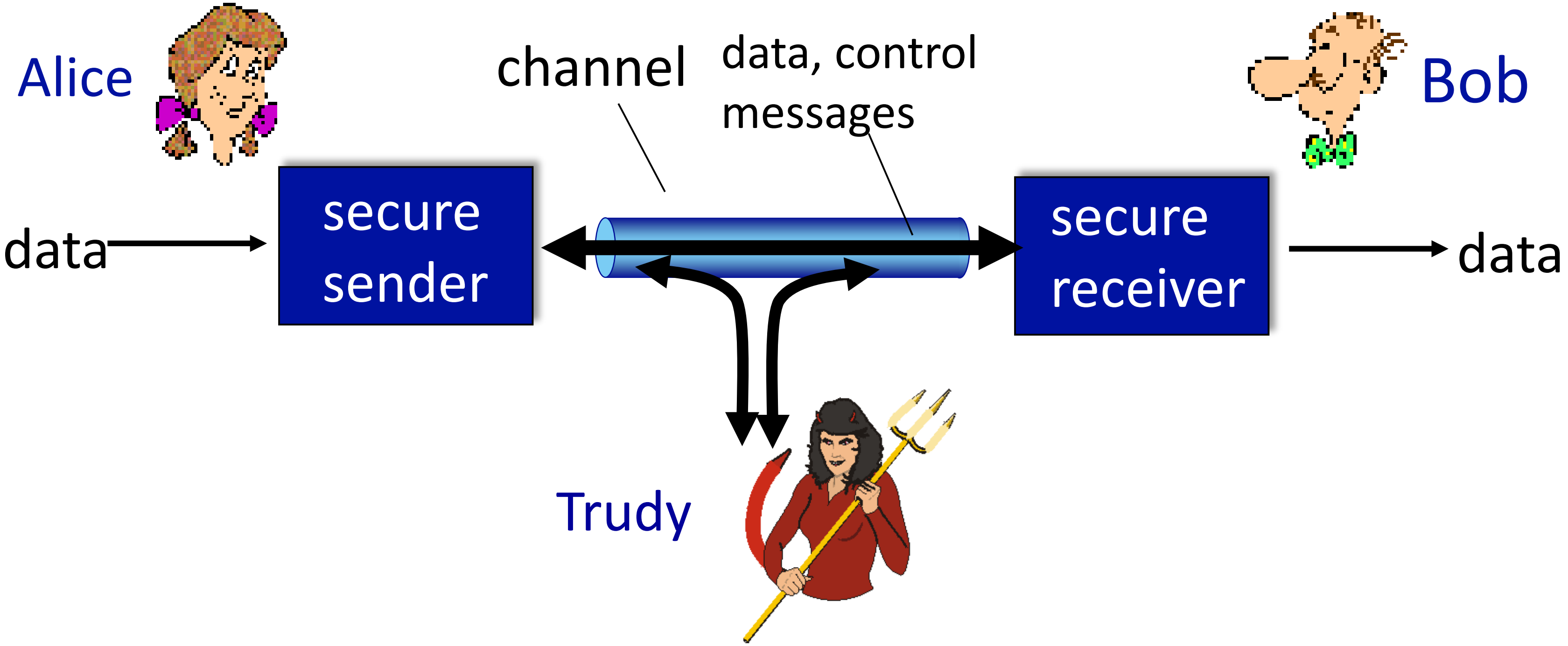
authentication: sender, receiver want to confirm identity of each other

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

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



Friends and enemies: Alice, Bob, Trudy

Who might Bob and Alice be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- BGP routers exchanging routing table updates
- other examples?

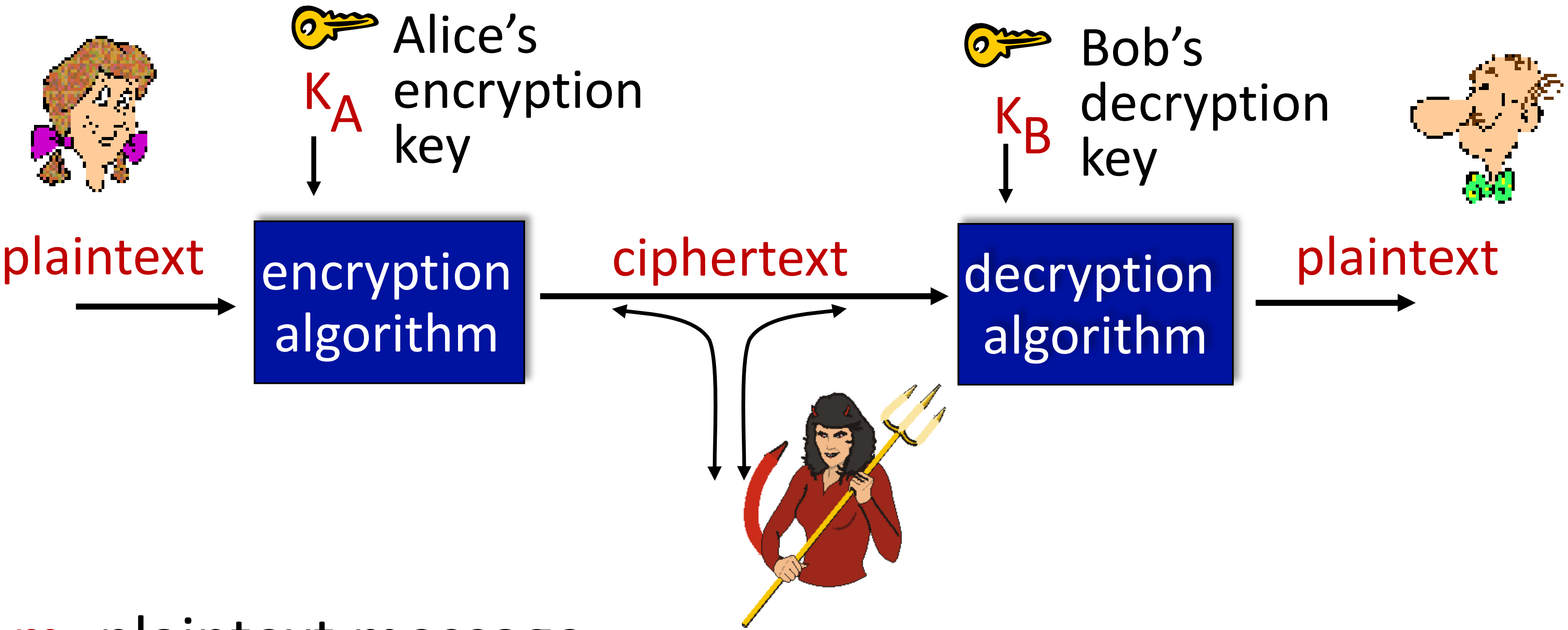
There are bad guys (and girls) out there!

- eavesdrop: intercept messages
- insert 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

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Operational security: firewalls and IDS

The language of cryptography



m: plaintext message

$K_A(m)$: ciphertext, encrypted with key K_A

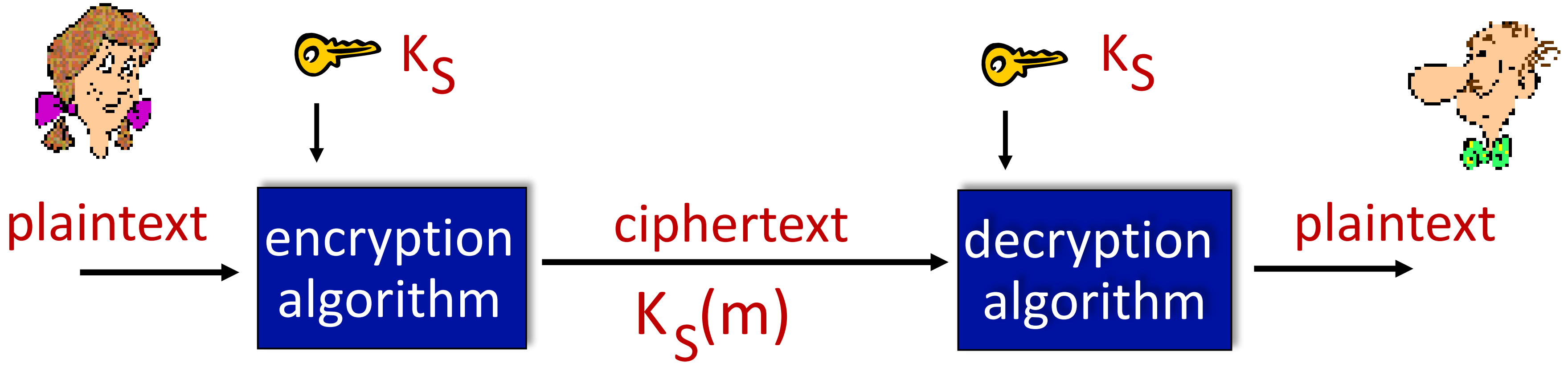
$m = K_B(K_A(m))$

Breaking an encryption scheme

- **cipher-text only attack:** Trudy has ciphertext she can analyze
- **two approaches:**
  - brute force
    - search through all keys
  - statistical analysis

- **known-plaintext attack:** Trudy has plaintext corresponding to ciphertext
  - *e.g.,* in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- **chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext

Symmetric key cryptography



symmetric key crypto

- Bob and Alice share same (symmetric) key: K

How do Bob and Alice agree on key value?

Simple encryption scheme

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
```

Encryption key: mapping from set of 26 letters to set of 26 letters

A more sophisticated encryption approach

- n substitution ciphers, $M_1,M_2,\dots,M_n$
- cycling pattern:
  - e.g., n=4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern

- key need not be just n-bit pattern

Two types of symmetric ciphers

- **Stream ciphers**
  - encrypt one bit at time

  - SSL/TLS connections
  - Bluetooth connections
  - Cellular and 4G connections

- **Block ciphers**
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

  - Data Encryption Standard (**DES**)
  - Triple DES (**3DES**)
  - Advanced Encryption Standard (**AES**)
  - International Data Encryption Algorithm (**IDEA**)
  - …

Symmetric key crypto: DES

DES: Data Encryption Standard

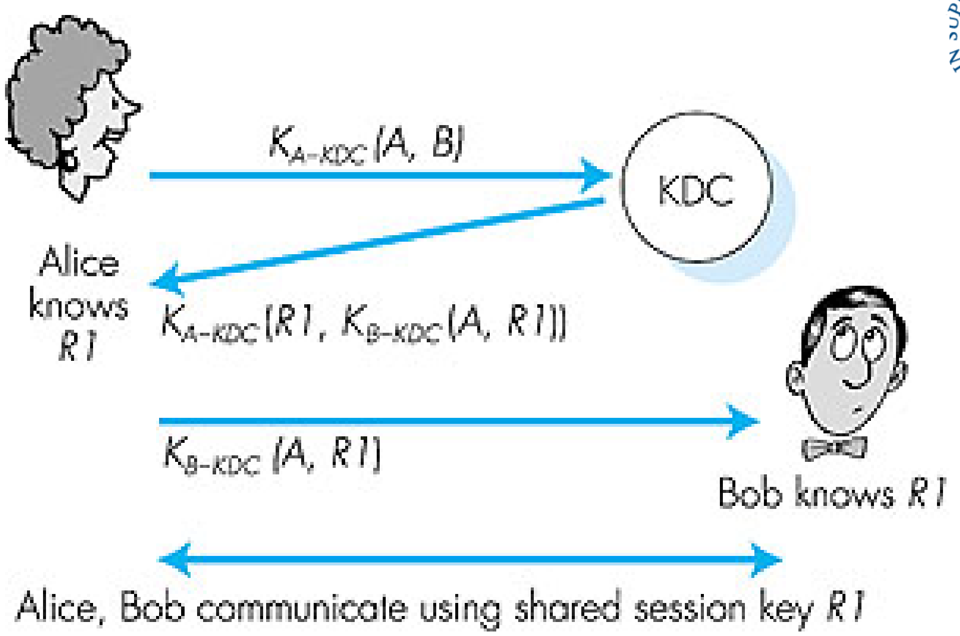
- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard

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

Key Question

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



Solutions

- Direct exchange (in person)
- Key Distribution Center (KDC)
  - Trusted entity acting as intermediary between entities
- Using public key cryptography

- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC
- Alice communicates with KDC, gets session key R1, and $K_{B\text{-}KDC}(A,R1)$
- Alice sends Bob $K_{B\text{-}KDC}(A,R1)$, Bob extracts R1
- Alice, Bob now share the symmetric key R1

Public Key Cryptography

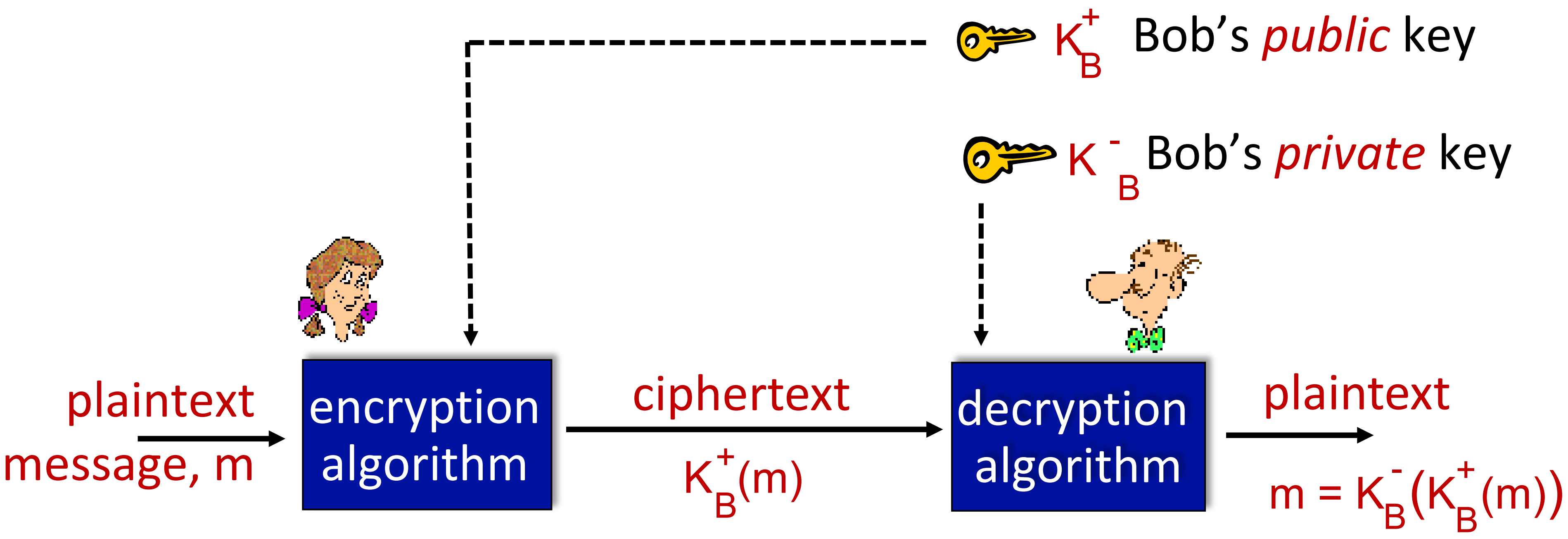
symmetric key crypto:

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto

- *radically* different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public Key Cryptography



Public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ 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

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

result is the same!

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use RSA to exchange a symmetric session key K_S
- once both have K_S, they use symmetric key cryptography

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS

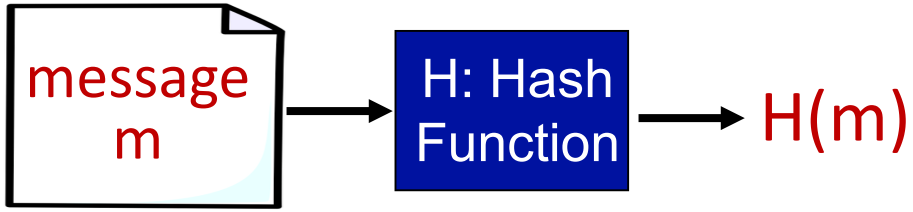
Message Integrity

- Allows communicating parties to verify that received messages are authentic
  - Source of message is who/what you think it is
  - Content of message has not been altered
  - Message has not been replayed
  - Sequence of messages is maintained
- Let’s first talk about message digests

Message digests

Goal: fixed-length, easy- to-compute digital “fingerprint”

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



Hash function properties

- **Easy** to compute
- **Irreversibility**: given message digest *x*, computationally infeasible to find *m* such that *x = H(m)*
- **Collision resistance**: Given [m, H(m)], it must be computationally infeasible to produce m’ (with m<>m’) such that H(m) = H(m’)
- Seemingly **random** output

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

But, given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

message	ASCII format
I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

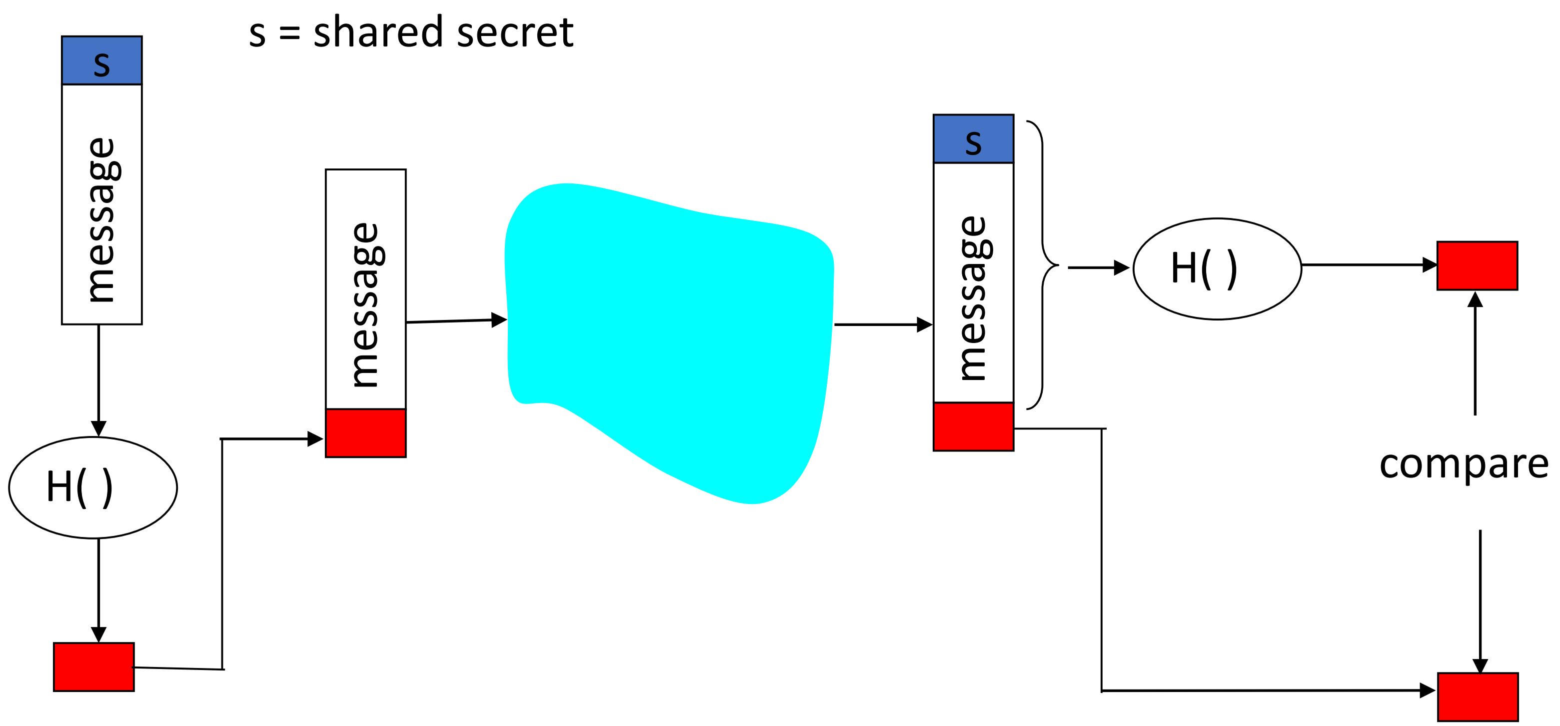
Hash function algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process
  - given arbitrary 128-bit string *x*, appears difficult to construct msg *m* whose MD5 hash is equal to *x*

- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Message Authentication Code (MAC)

- Authenticates sender
- Verifies message integrity
- No encryption! Also called "keyed hash"

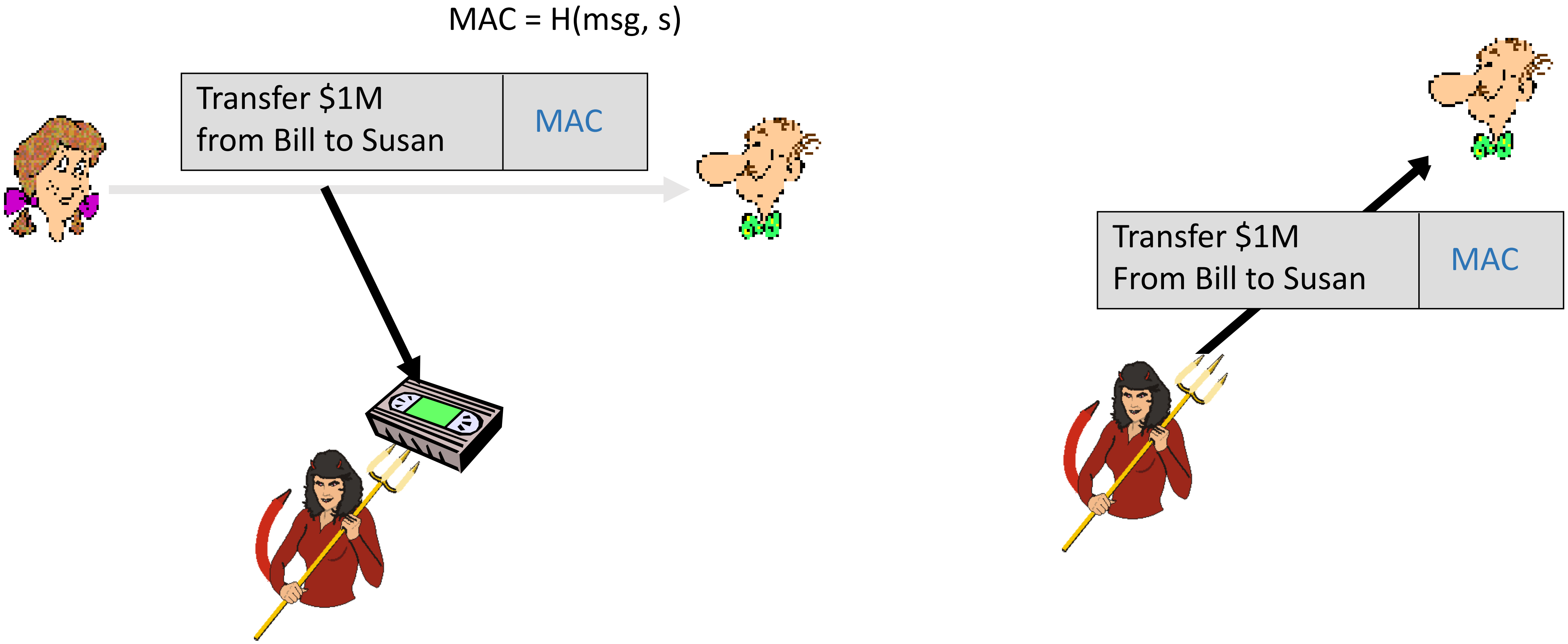


HMAC [RFC 2104]

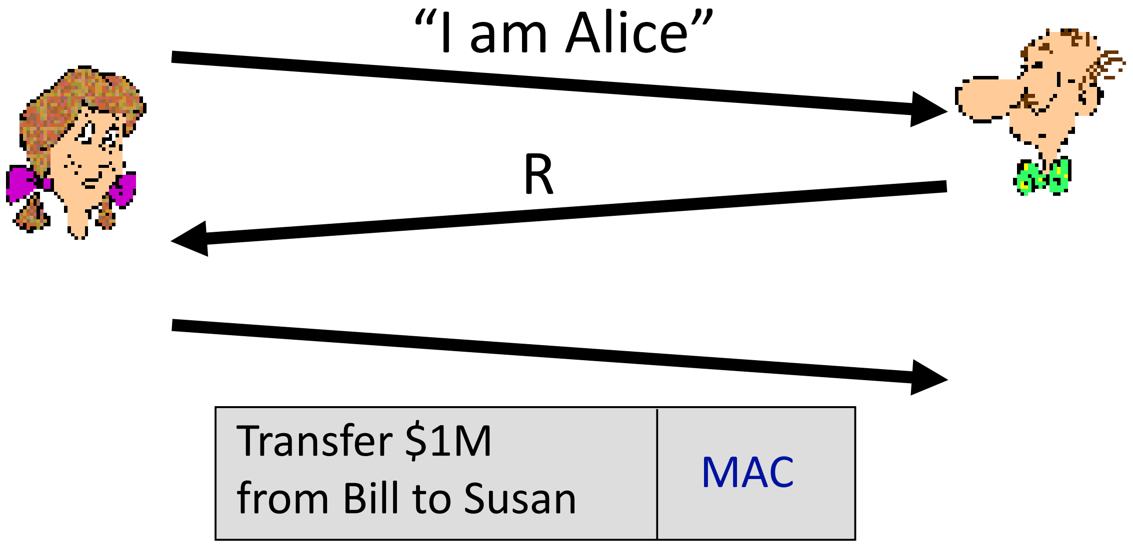
- Popular MAC standard
- Can use both MD5 and SHA-1

1. Concatenates secret to front of message: [s||m]
2. Hashes concatenated message: H([s||m])
3. Concatenates the secret to front of digest: [H([s||m])||m]
4. Hashes the combination again: H([H([s||m])||m])

Playback attack



Protection against playback attack



MAC =H(msg, s, R)

Digital signatures

- Cryptographic technique analogous to hand-written signatures
  - sender (Bob) digitally signs document: he is document owner/creator
- Verifiable
  - The recipient (Alice) can verify and prove that Bob, and no one else, signed the document
- Non-forgeable
  - The sender (Bob) can prove that someone else has signed a message
- Non repudiation
  - The recipient (Alice) can prove that Bob signed m and not m’
- Message integrity
  - The sender (Bob) can prove that he signed m and not m’

Relevant Question

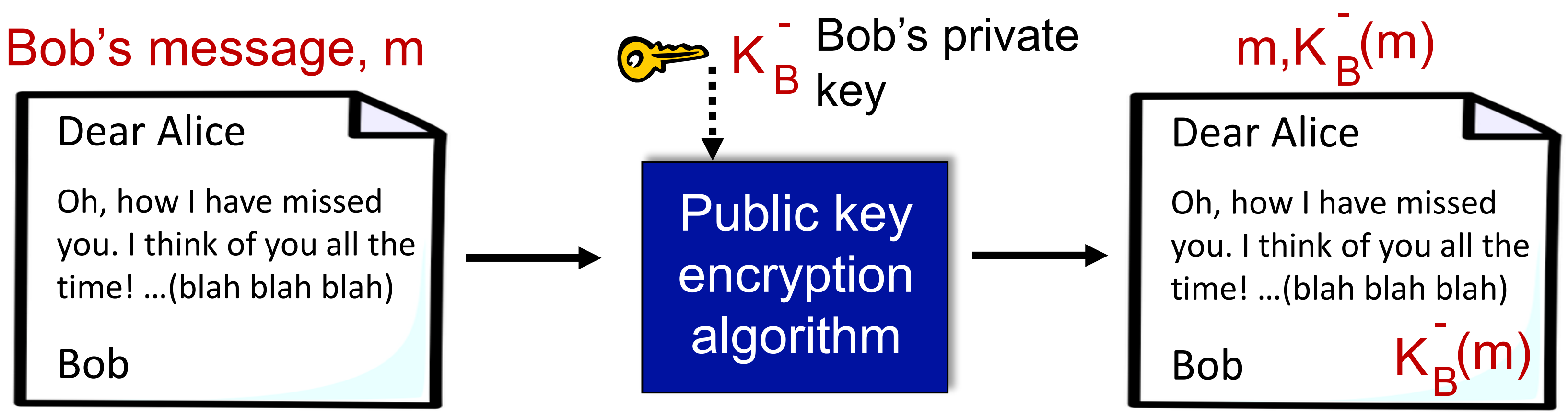
May we use MAC as a Digital Signature??

- Goal is similar to that of a MAC
  - MAC guarantees message integrity

- MAC does not guarantee
  - Verifiability
  - Non forgeability
  - Non repudiation
- Solution: use public key cryptography

Digital signatures

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



RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

result is the same!

Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- 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:

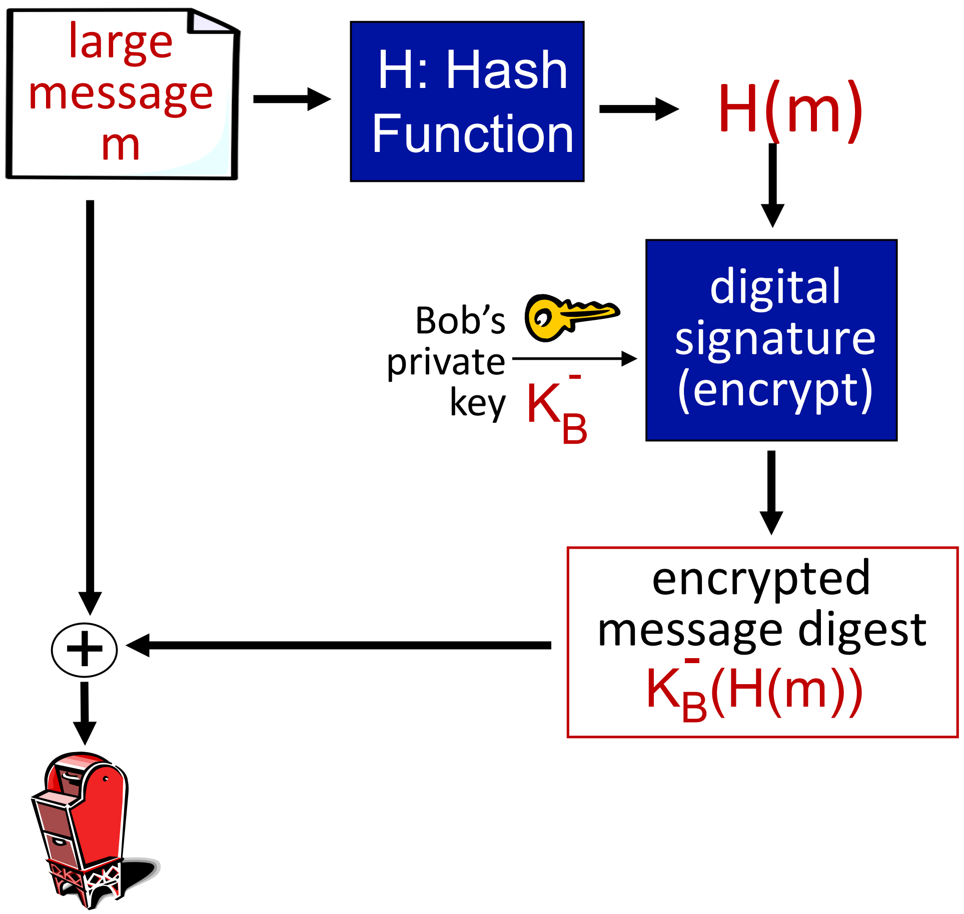
- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

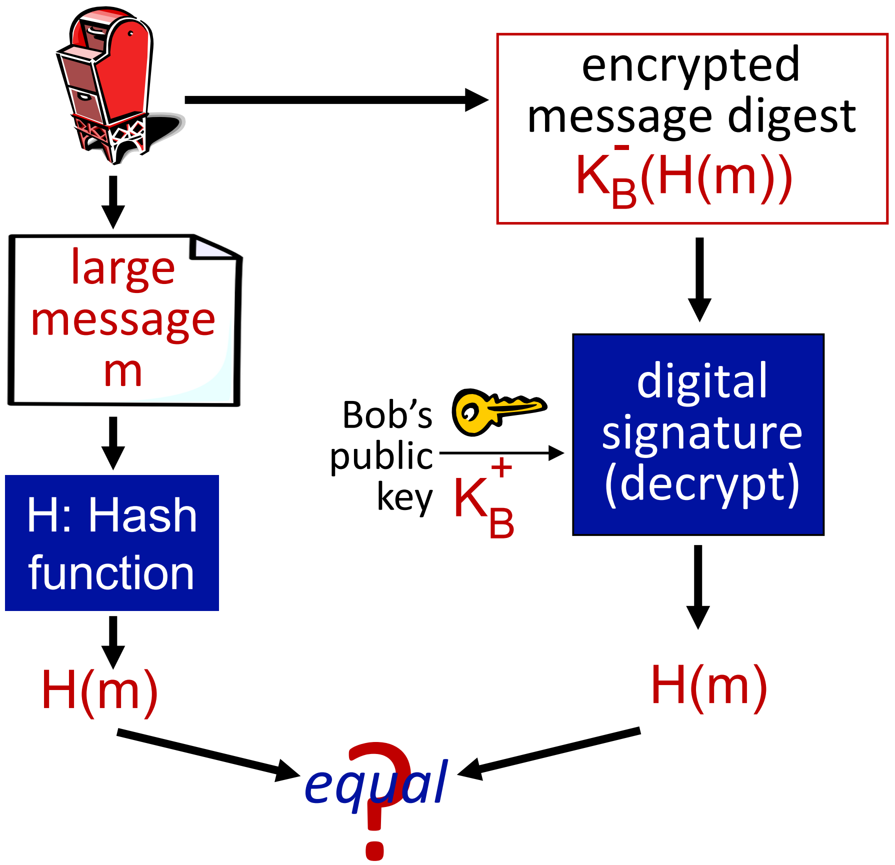
✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Motivation for public key certification

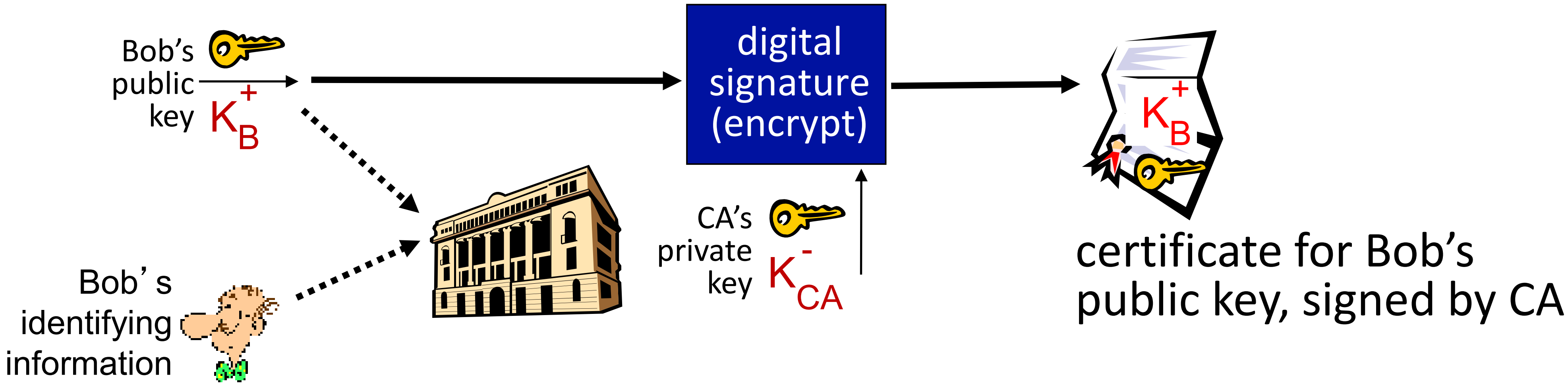
- Trudy send a message to Alice
  - Trudy creates e-mail message:
    Trudy signs message with her private key
  - Trudy sends message to Alice
- Trudy sends Alice her public key, but says it's Bob's public key
- Alice verifies signature
- Alice assumes that message is authentic

My loved Alice,
I also think of you all the time!
I want to take you in marriage soon!

Bob

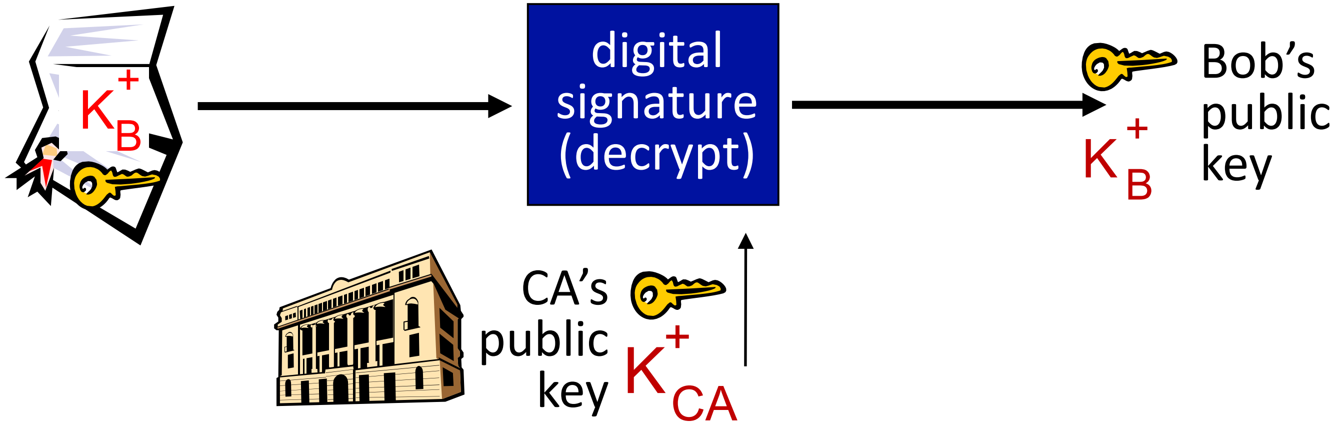
Public key Certification Authorities (CA)

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



Public key Certification Authorities (CA)

- 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



Public Key certificate

- Primary standard ITU X.509 (RFC 2459)
- Certificate includes:
  - Issuer name
  - Entity’s name, address, domain name, etc.
  - Entity’s public key
  - Digital signature
    - signed with issuer’s private key
- Public-Key Infrastructure (PKI)
  - Certificates and certification authorities
  - Often considered “heavy”

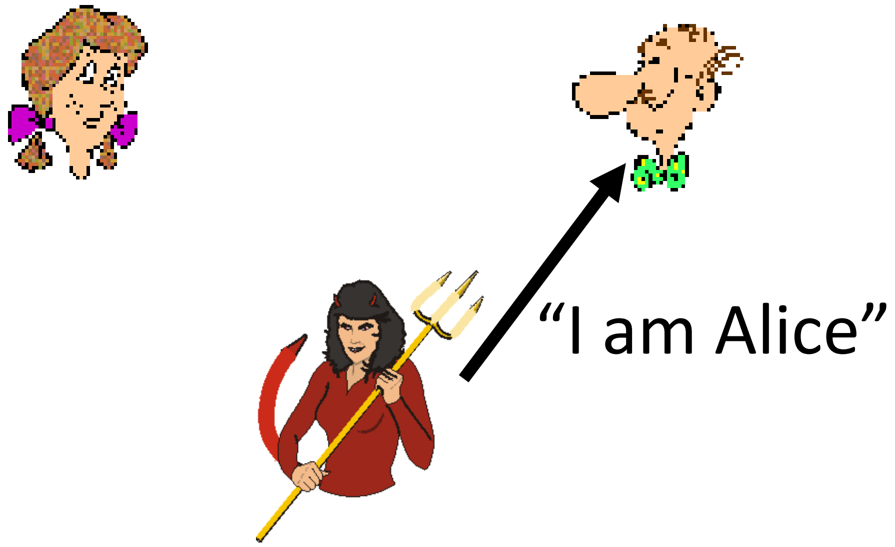
Chapter 8 outline

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS

Authentication

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

Protocol ap1.0: Alice says "I am Alice"

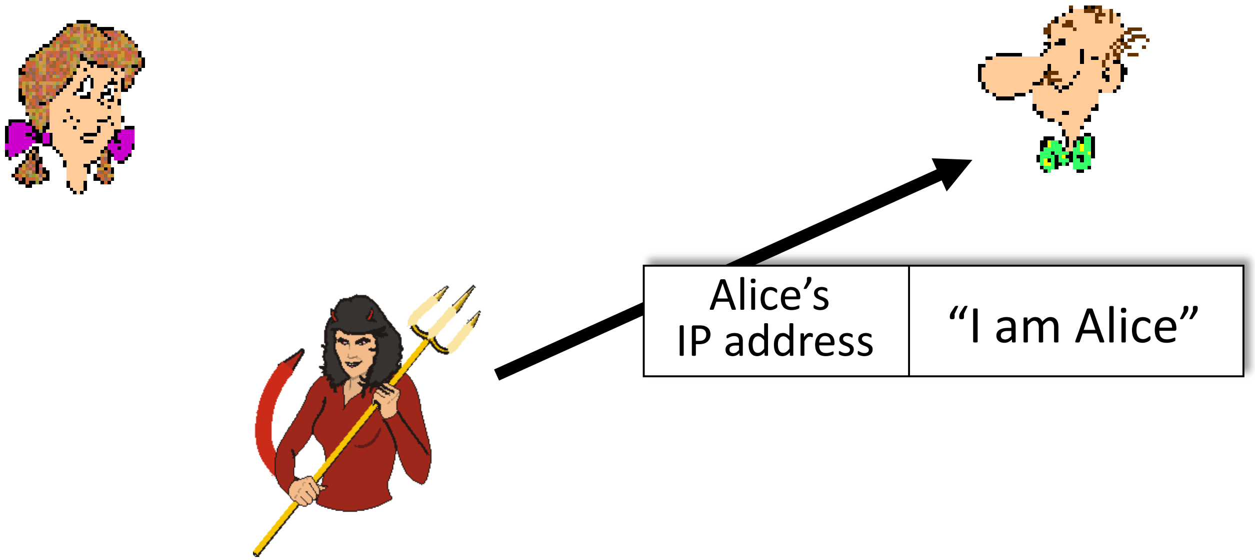


in a network, Bob can not "see" Alice, so Trudy simply declares herself to be Alice

Authentication: another try

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

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

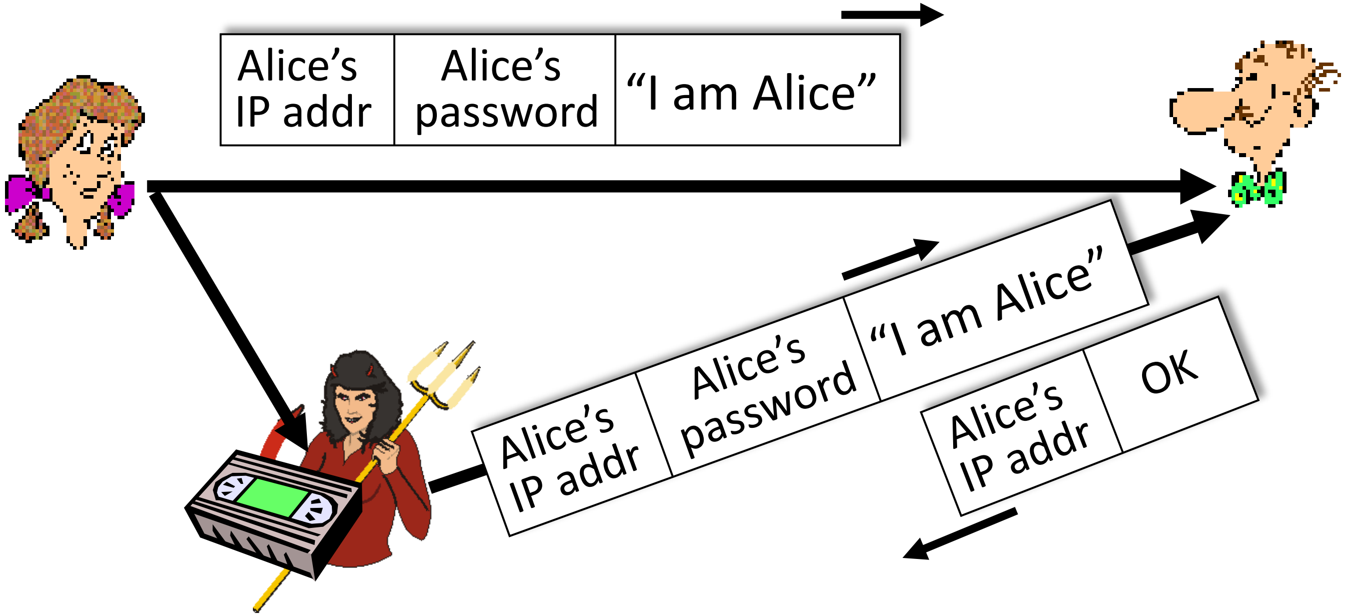


Trudy can create a packet "spoofing" Alice's address

Authentication: a third try

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

Protocol ap3.0: Alice says "I am Alice" Alice says "I am Alice" and sends her secret password to "prove" it.

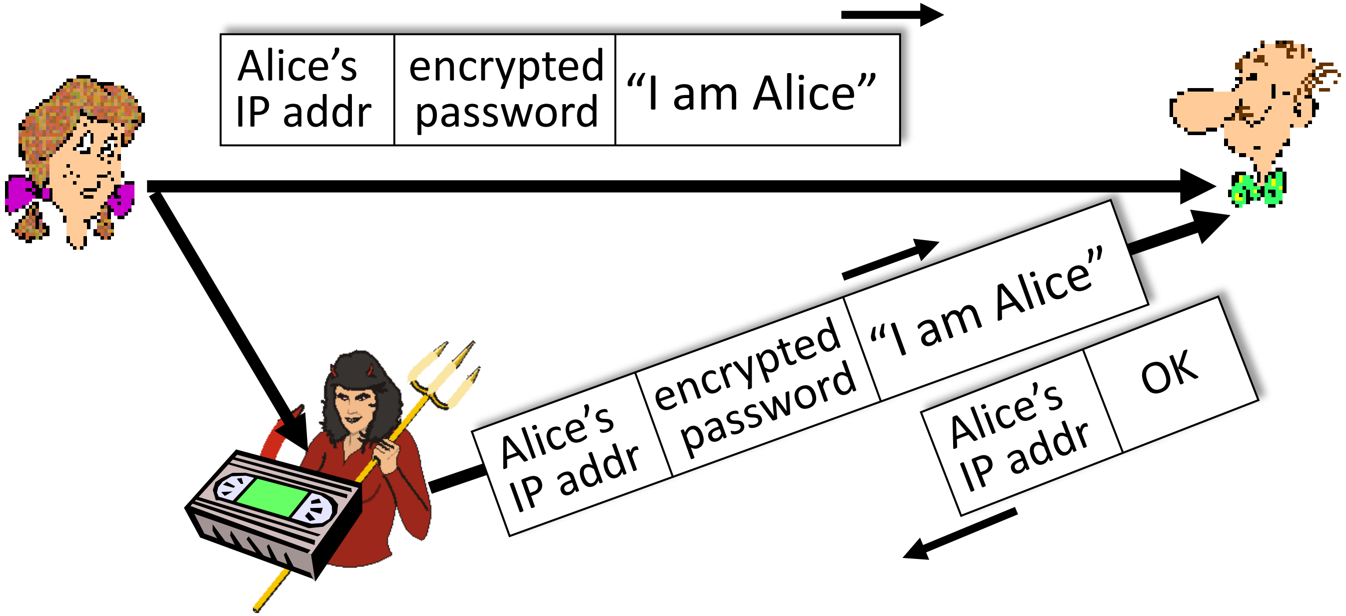


playback attack: *Trudy records Alice's packet and later plays it back to Bob*

Authentication: a modified third try

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

Protocol ap3.0: Alice says "I am Alice" Alice says "I am Alice" and sends her encrypted secret password to "prove" it.



playback attack still works: Trudy records Alice's packet and later plays it back to Bob

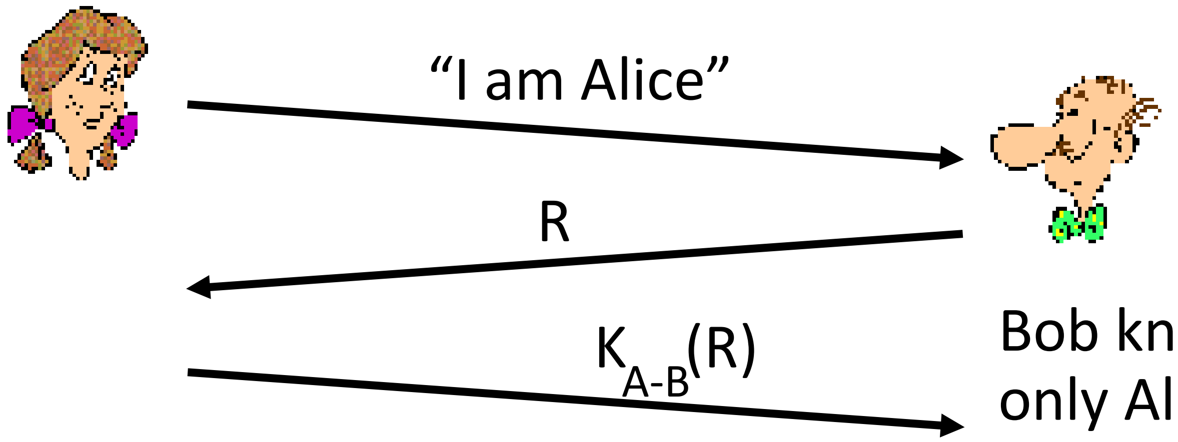
Authentication: a fourth try

Goal: avoid playback attack

nonce: number (R) used only once-in-a-lifetime

protocol ap4.0: to prove Alice "live", Bob sends Alice nonce, R

- Alice must return R, encrypted with shared secret key



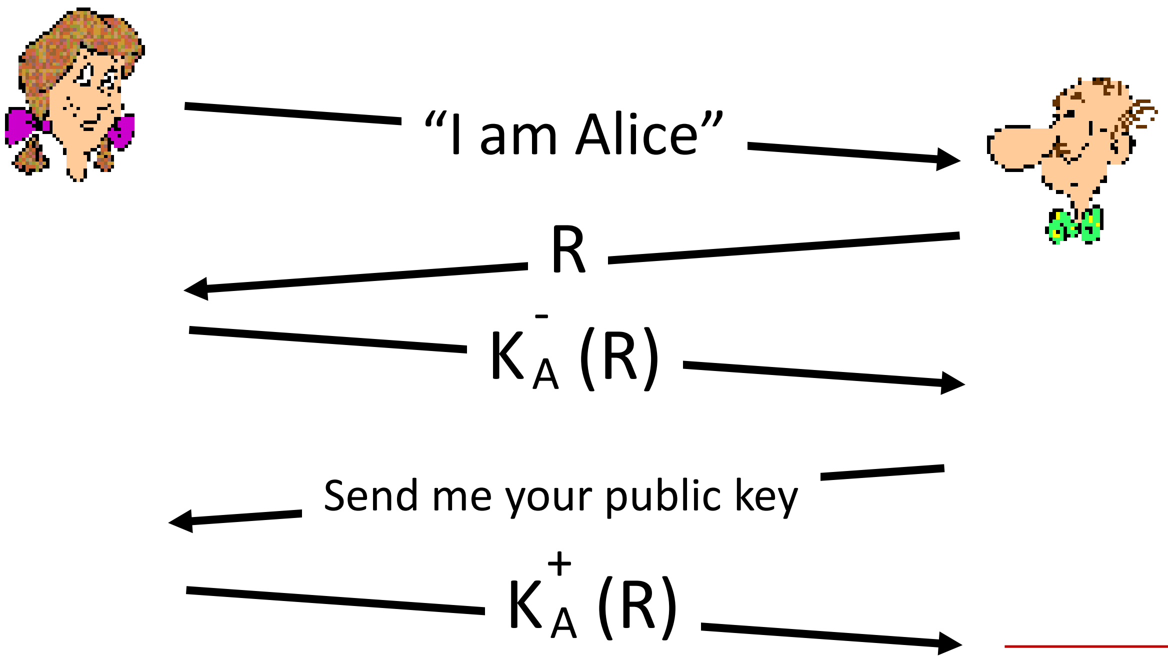
Bob know Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key - can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

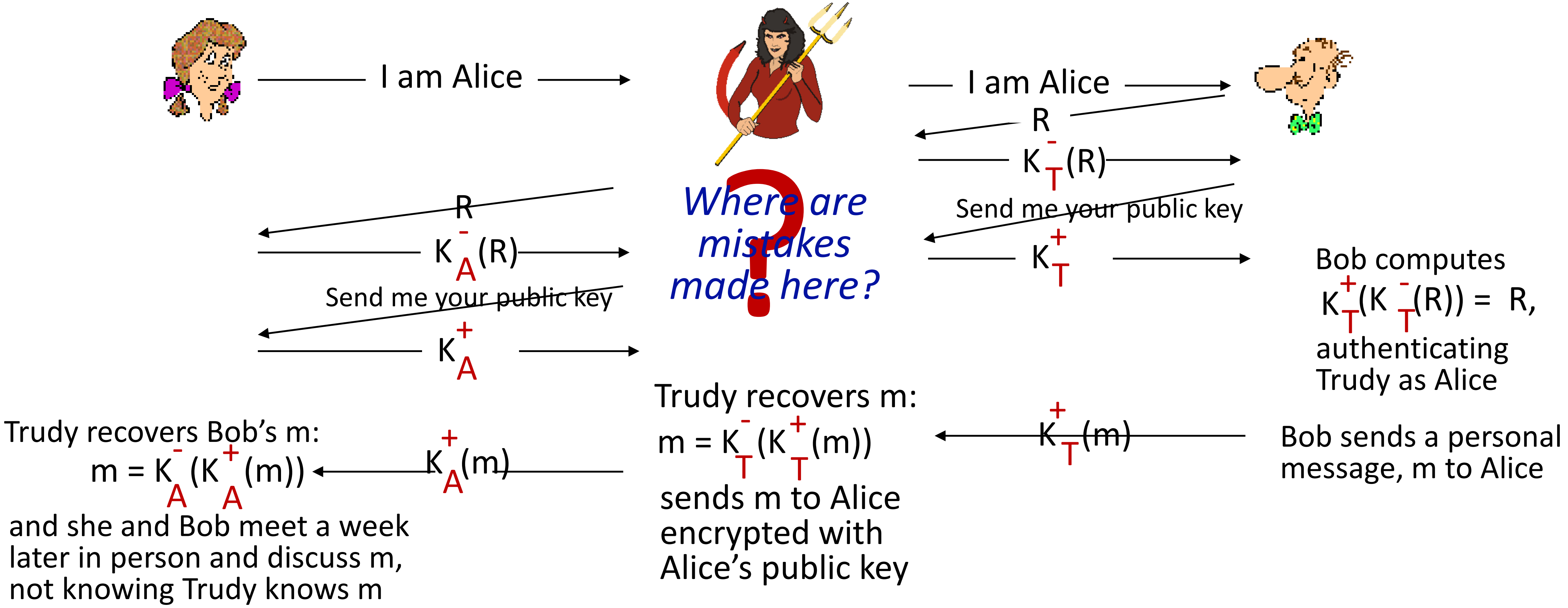
$K_A^+(K_A^-(R)) = R$

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

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

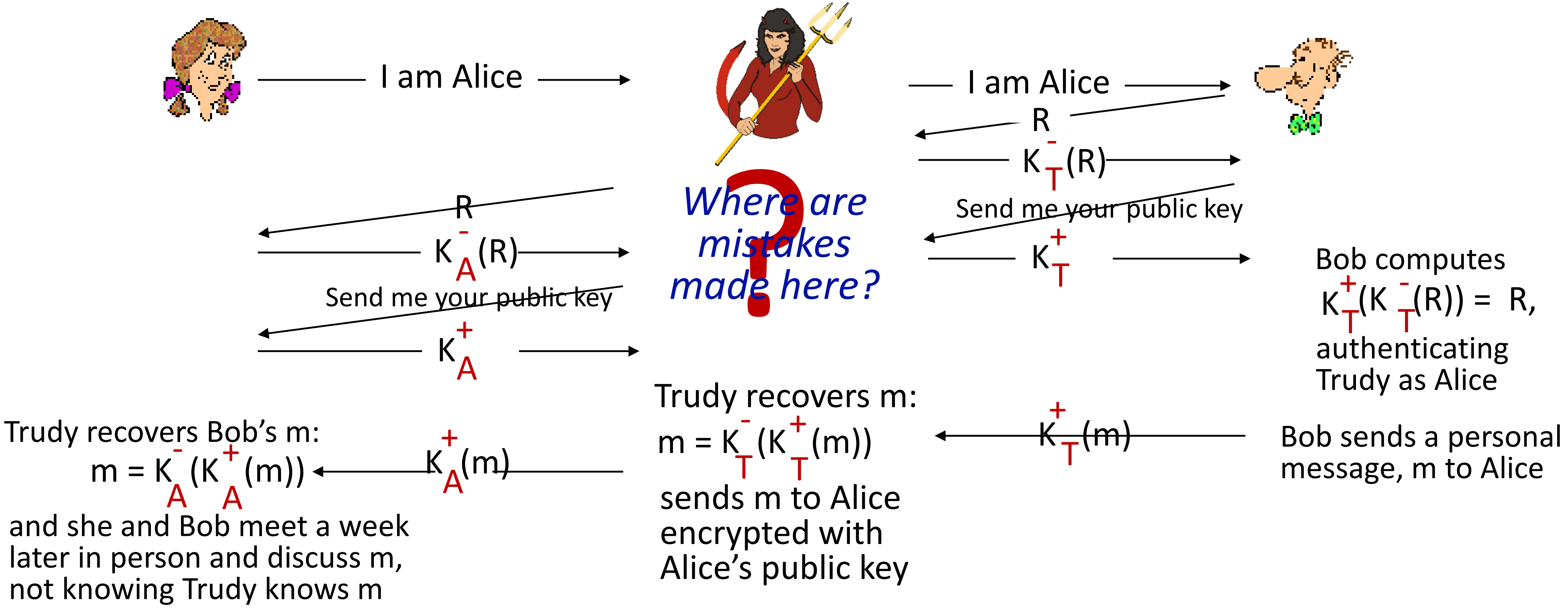
Authentication: ap5.0 – there's still a flaw!

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Authentication: ap5.0 – let's fix it!!

Recall the problem: Trudy poses as Alice (to Bob) and as Bob (to Alice)

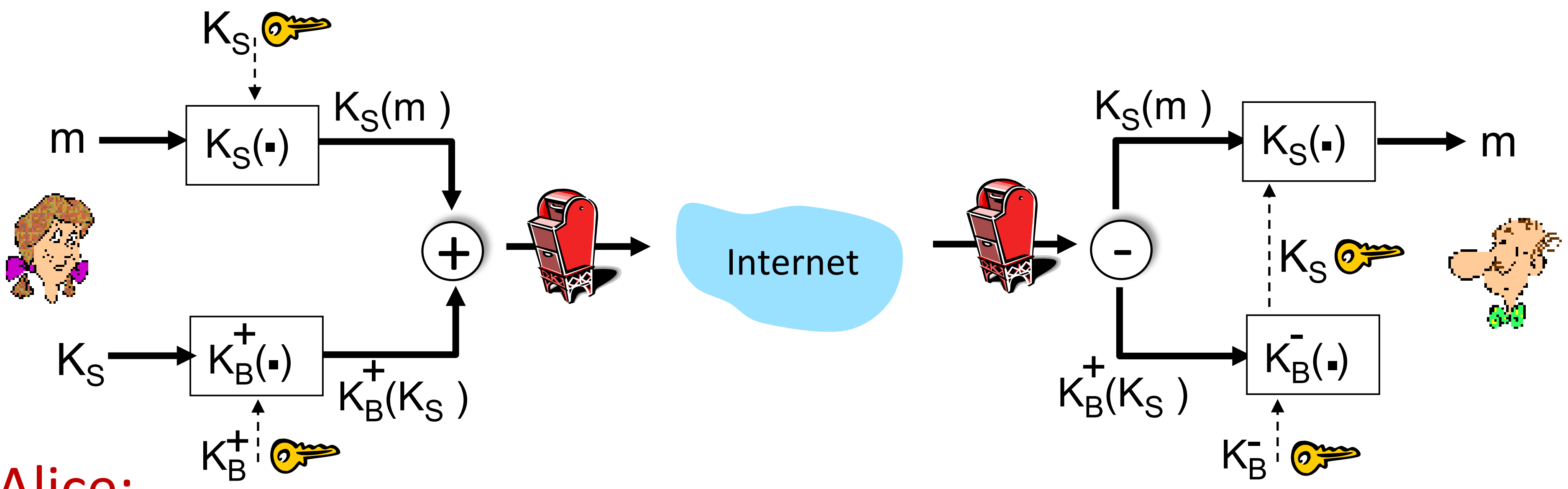


Chapter 8 outline

- What is network security?
- Principles of cryptography
- Message integrity
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Security in wireless and mobile networks
- Operational security: firewalls and IDS

Secure e-mail: confidentiality

Alice wants to send *confidential* e-mail, m, to Bob.

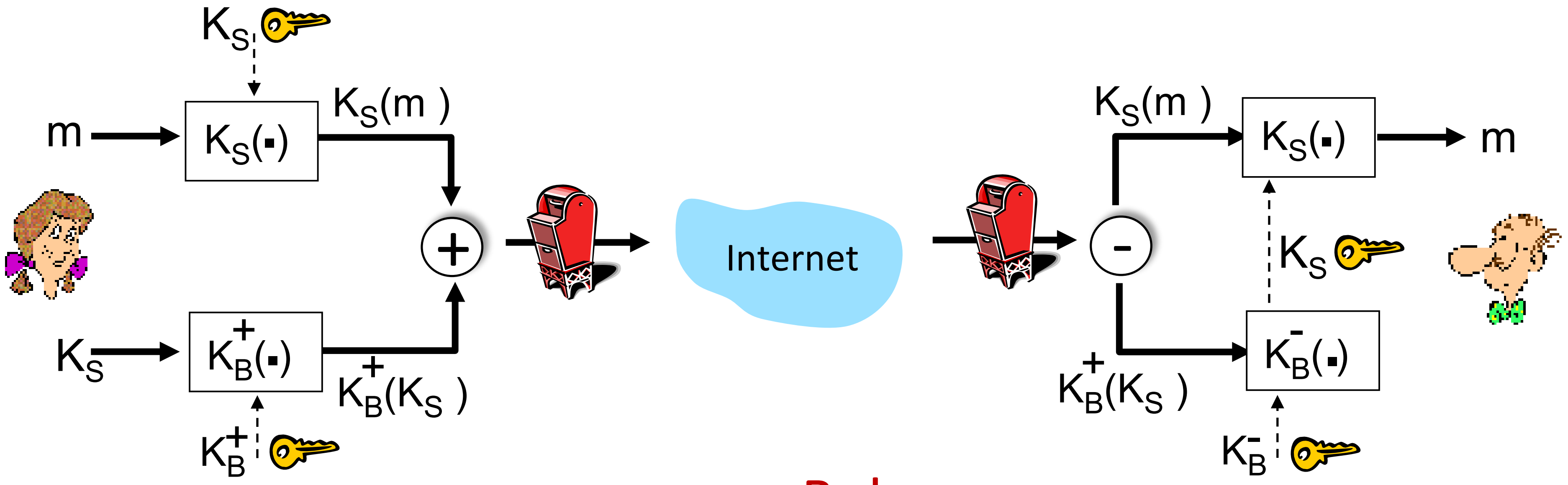


Alice:

- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $K^+_B(K_S)$ to Bob

Secure e-mail: confidentiality (more)

Alice wants to send *confidential* e-mail, m, to Bob.

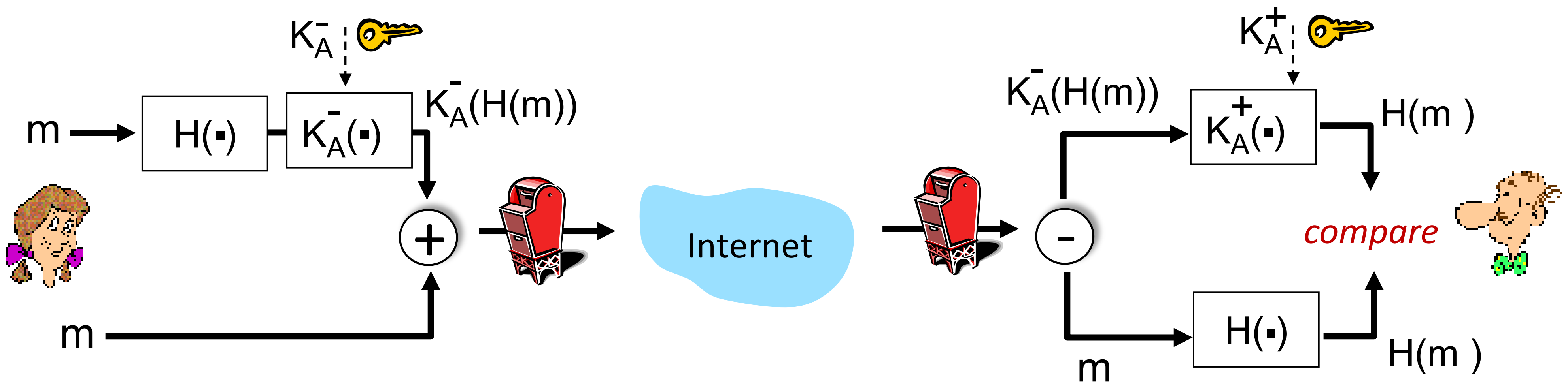


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail: integrity, authentication

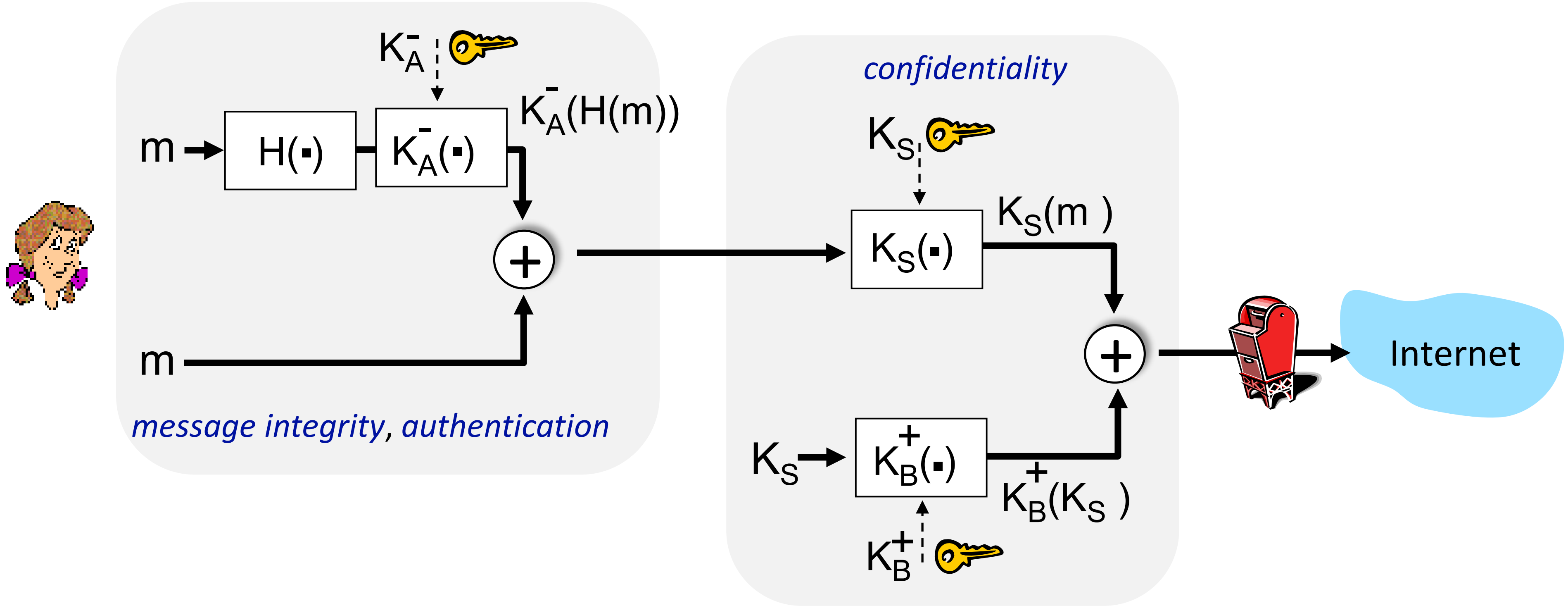
Alice wants to send m to Bob, with *message integrity*, *authentication*



- Alice digitally signs hash of her message with her private key, providing integrity and authentication
- sends both message (in the clear) and digital signature

Secure e-mail: integrity, authentication

Alice sends m to Bob, with *confidentiality, message integrity, authentication*



Alice uses three keys: her private key, Bob's public key, new symmetric key

What are Bob's complementary actions?

Chapter 8 outline

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Operational security: firewalls and IDS

Transport-layer security (TLS)

- widely deployed security protocol above the transport layer
  - supported by almost all browsers, web servers: https (port 443)
- provides:
  - confidentiality: via *symmetric encryption*
  - integrity: via *cryptographic hashing*
  - authentication: via *public key cryptography*

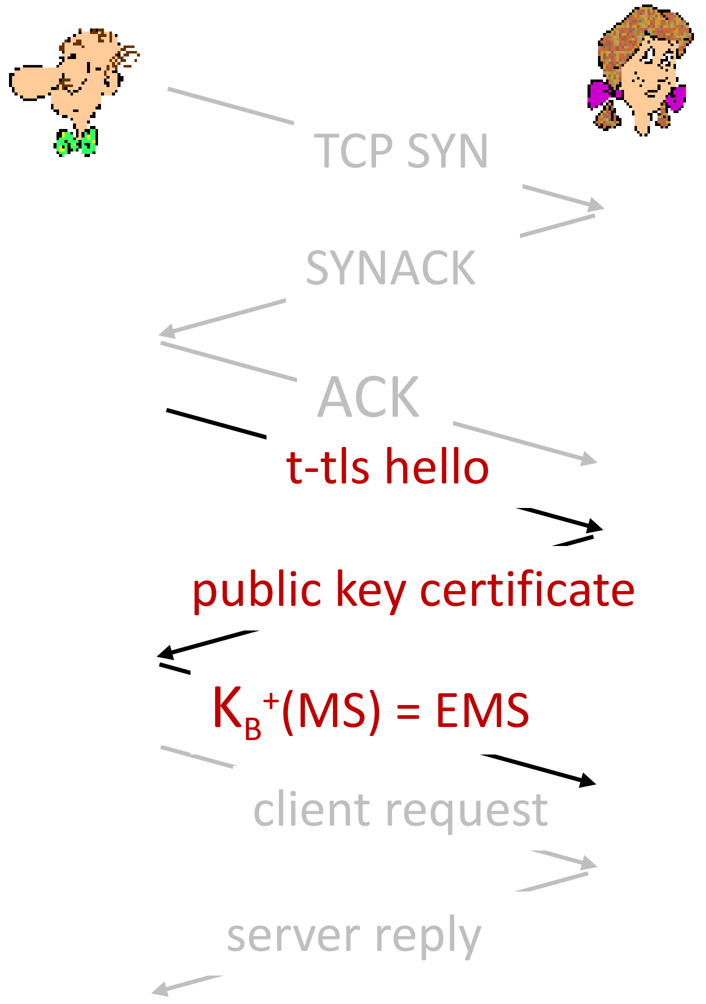
all techniques we have studied!

- history:
  - early research, implementation: secure network programming, secure sockets
  - secure socket layer (SSL) deprecated [2015]
  - TLS 1.3: RFC 8846 [2018]

Transport-layer security: what's needed?

- let's *build* a toy TLS protocol, *t-tls,* to see what's needed!
- we've seen the "pieces" already:
  - handshake: Alice, Bob use their certificates, private keys to authenticate each other, exchange or create shared secret
  - key derivation: Alice, Bob use shared secret to derive set of keys
  - data transfer: stream data transfer: data as a series of records
    - not just one-time transactions
  - connection closure: special messages to securely close connection

t-tls: initial handshake



t-tls handshake phase:

- Bob establishes TCP connection with Alice
- Bob verifies that Alice is really Alice
- Bob sends Alice a master secret key (MS), used to generate all other keys for TLS session
- potential issues:
  - 3 RTT before client can start receiving data (including TCP handshake)

t-tls: cryptographic keys

- considered bad to use same key for more than one cryptographic function
  - different keys for message authentication code (MAC) and encryption
- four keys:
  - K_c : encryption key for data sent from client to server
  - M_c : MAC key for data sent from client to server
  - K_s : encryption key for data sent from server to client
  - M_s : MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
  - takes master secret and (possibly) some additional random data to create new keys

t-tls: encrypting data

- recall: TCP provides data *byte stream* abstraction
- Q: can we encrypt data in-stream as written into TCP socket?
  - *A:* where would MAC go? If at end, no message integrity until all data received and connection closed!
  - *solution:* break stream in series of "records"
    - each client-to-server record carries a MAC, created using M_c
    - receiver can act on each record as it arrives
- t-tls record encrypted using symmetric key, $K_{c,}$ passed to TCP:

K_c(

length	*data*	*MAC*

)

t-tls: encrypting data (more)

- possible attacks on data stream?
  - *re-ordering:* man-in middle intercepts TCP segments and reorders (manipulating sequence #s in unencrypted TCP header)
  - *replay*
- solutions:
  - use TLS sequence numbers (data, TLS-seq-# incorporated into MAC)
  - use nonce

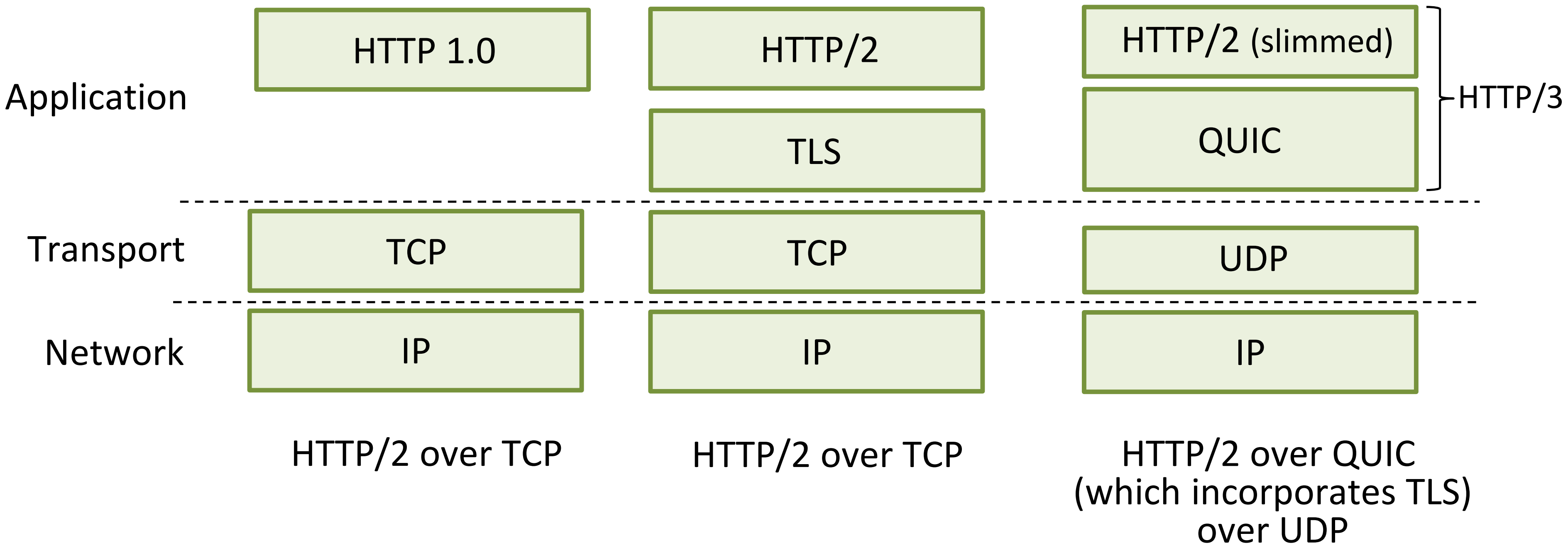
t-tls: connection close

- truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is
- solution: record types, with one type for closure
  - type 0 for data; type 1 for close
- MAC now computed using data, type, sequence #

K_c(| *length* | *type* | *data* | *MAC* |)

Transport-layer security (TLS)

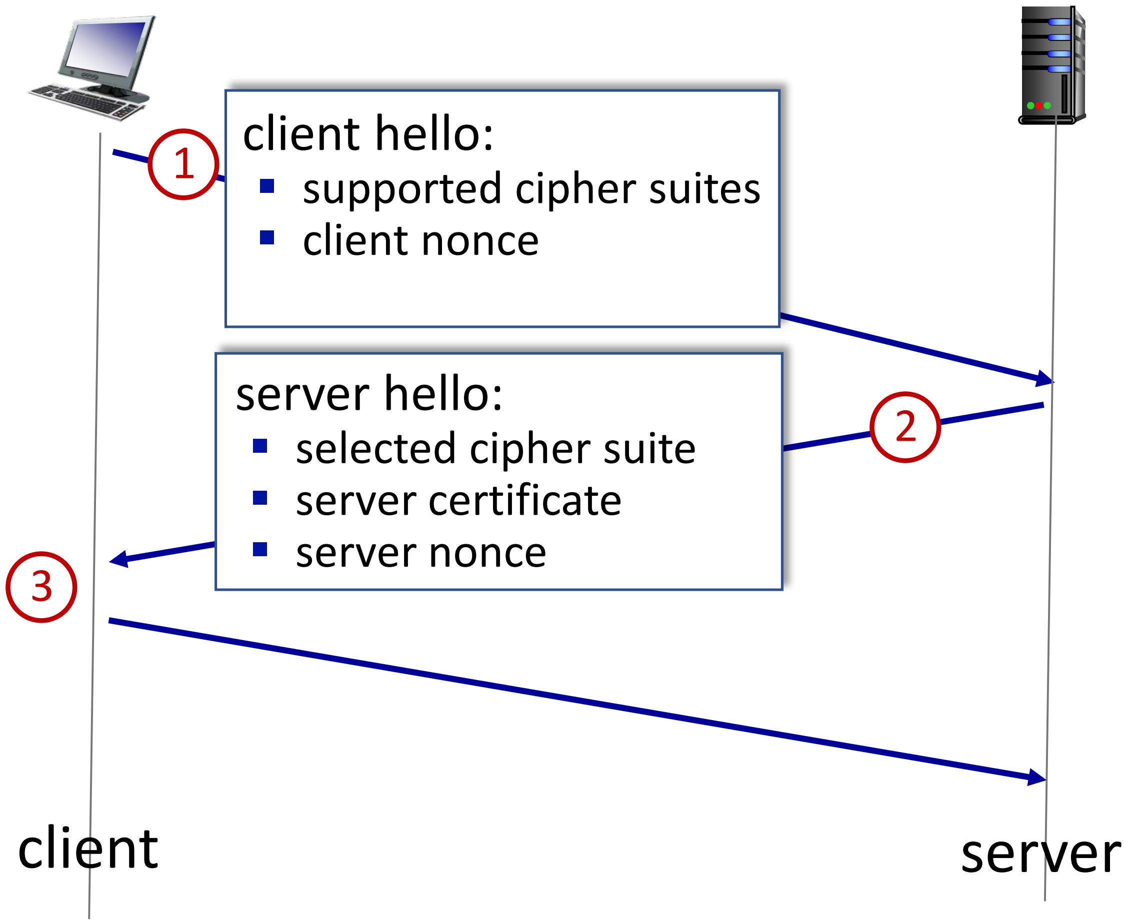
- TLS provides an API that *any* application can use
- an HTTP view of TLS:



TLS: 1.3 cipher suite

- "cipher suite":
  - Algorithm for key generation
  - Public-key encryption algorithm
  - Symmetric-key encryption algorithm
  - MAC algorithm
- TLS: 1.3 (2018): more limited cipher suite choice than TLS 1.2 (2008)
  - only 5 choices, rather than 37 choices

TLS 1.3 handshake: 1 RTT



1. client TLS hello msg:
   - *guesses* key agreement protocol, parameters
   - indicates cipher suites it supports

2. server TLS hello msg chooses
   - key agreement protocol, parameters
   - cipher suite
   - server-signed certificate

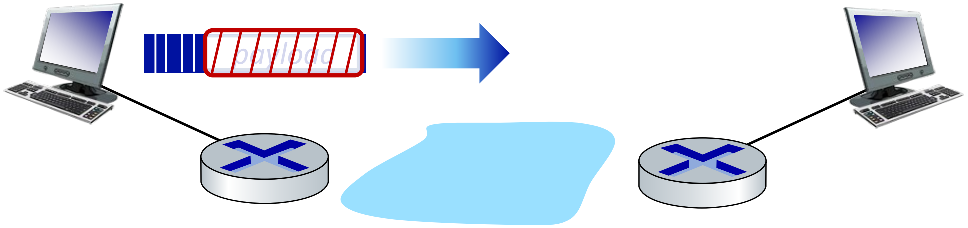
3. client:
   - checks server certificate
   - generates key
   - can now make application request (e.g.., HTTPS GET)

Roadmap

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Operational security: firewalls and IDS

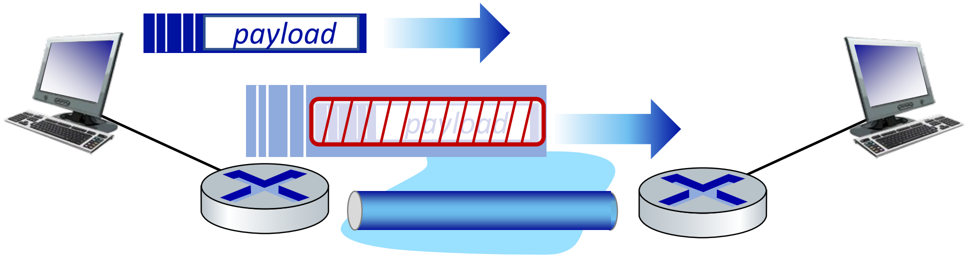
IP Sec

- provides datagram-level encryption, authentication, integrity
  - for both user traffic and control traffic (e.g., BGP, DNS messages)
- two “modes”:



transport mode:

- *only* datagram *payload* is encrypted, authenticated

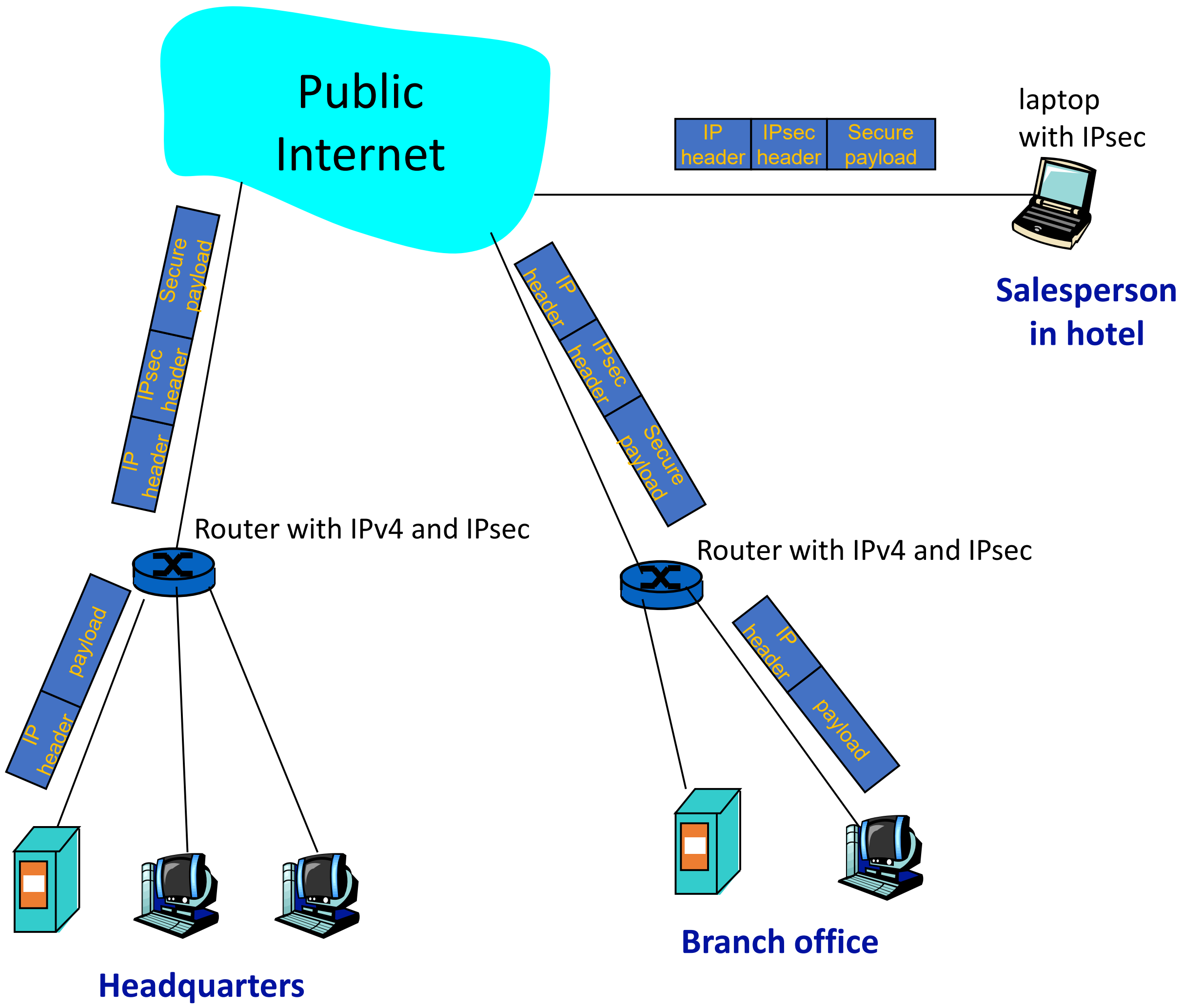


tunnel mode:

- entire datagram is encrypted, authenticated
- encrypted datagram encapsulated in new datagram with new IP header, tunneled to destination

Virtual Private Networks (VPN)

- Institutions often want private networks for security
  - Costly! Separate routers, links, DNS infrastructure
- With a VPN, institution's inter-office traffic is sent over public Internet instead
  - But inter-office traffic is encrypted before entering public Internet

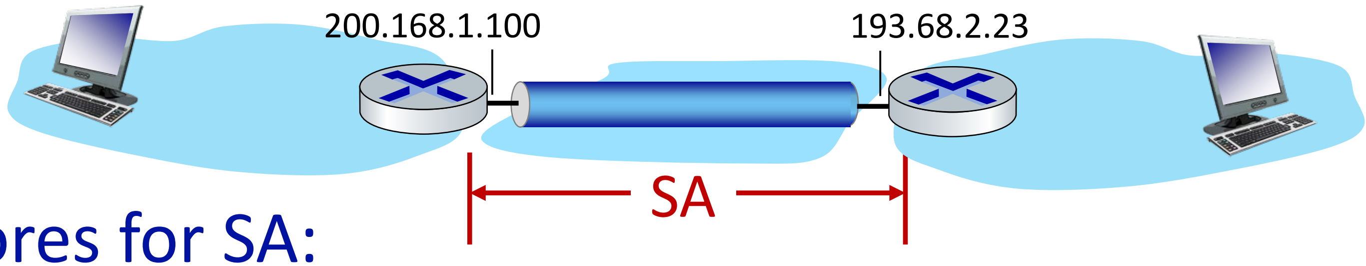


Two IPsec protocols

- Authentication Header (AH) protocol [RFC 4302]
  - provides source authentication & data integrity but *not* confidentiality
- Encapsulation Security Protocol (ESP) [RFC 4303]
  - provides source authentication, data integrity, *and confidentiality*
  - more widely used than AH

Security associations (SAs)

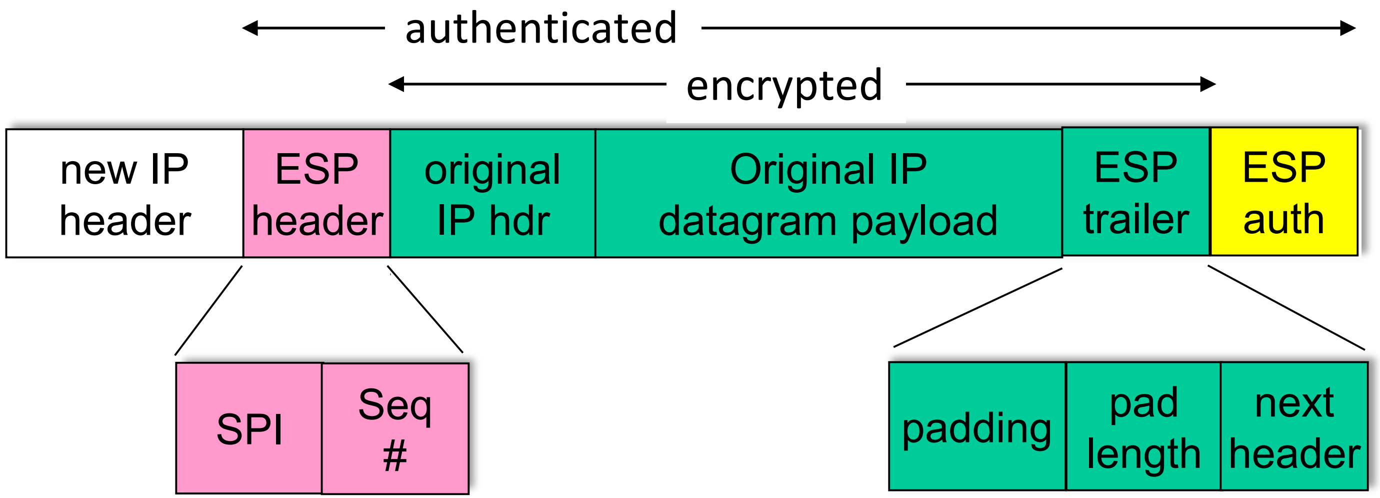
- before sending data, security association (SA) established from sending to receiving entity (directional)
- ending, receiving entitles maintain *state information* about SA
  - recall: TCP endpoints also maintain state info
  - IP is connectionless; IPsec is connection-oriented!



R1 stores for SA:

- 32-bit identifier: *Security Parameter Index (SPI)*
- origin SA interface (200.168.1.100)
- destination SA interface (193.68.2.23)
- type of encryption used
- encryption key
- type of integrity check used
- authentication key

IPsec datagram



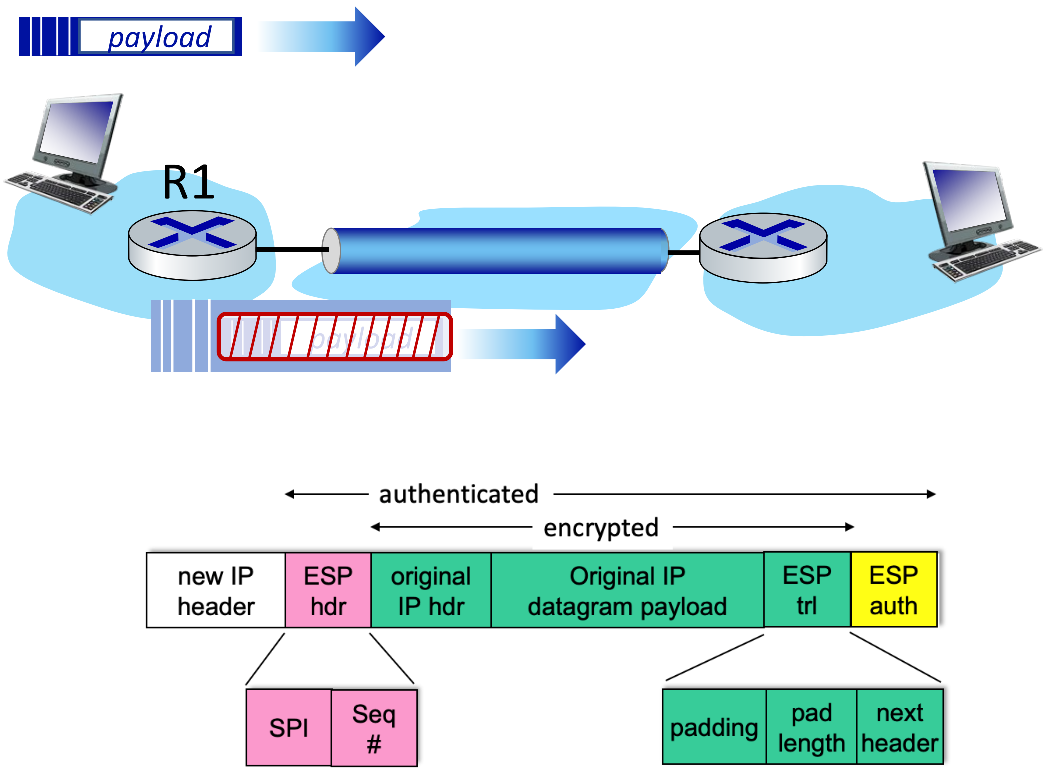
tunnel mode ESP

- 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 created with shared secret key

ESP tunnel mode: actions

at R1:

- appends ESP trailer to original datagram (which includes original header fields!)
- encrypts result using algorithm & key specified by SA
- appends ESP header to front of this encrypted quantity
- creates authentication MAC using algorithm and key specified in SA
- appends MAC forming *payload*
- creates new IP header, new IP header fields, addresses to tunnel endpoint



IPsec sequence numbers

- for 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

IPsec security databases

Security Policy Database (SPD)

- policy: for given datagram, sender needs to know if it should use IP sec
- policy stored in security policy database (SPD)
- needs to know which SA to use
  - may use: source and destination IP address; protocol number

SAD: "how" to do it

Security Assoc. Database (SAD)

- endpoint holds SA state in security association database (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, processing
- datagram accordingly.

SPD: "what" to do

Relevant Questions

Trudy sits somewhere between R1, 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?

IPsec summary

- IPsec protocol for datagram-level security
- 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

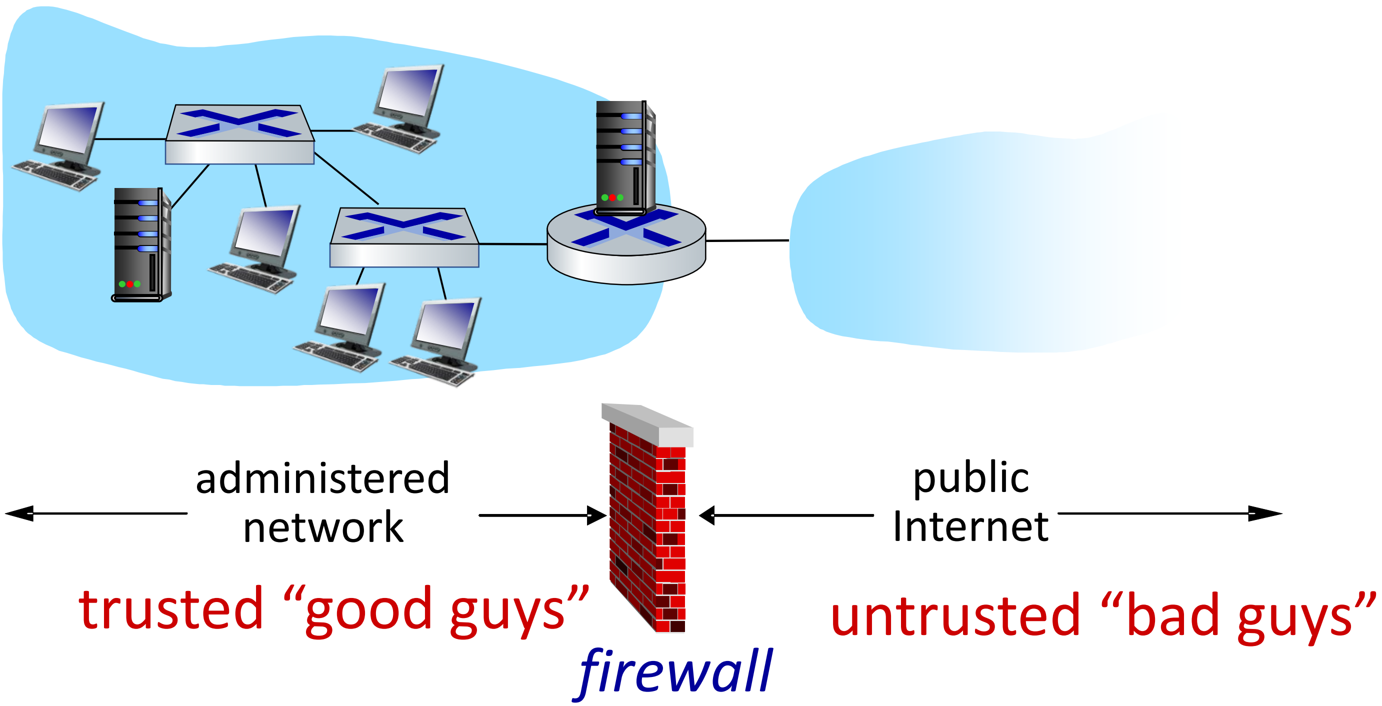
Roadmap

- What is network security?
- Principles of cryptography
- Message integrity
- Authentication
- Securing e-mail
- Securing TCP connections: TLS
- Network layer security: IPsec
- Operational security: firewalls and IDS

Firewalls

firewall

isolates organization's internal network 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

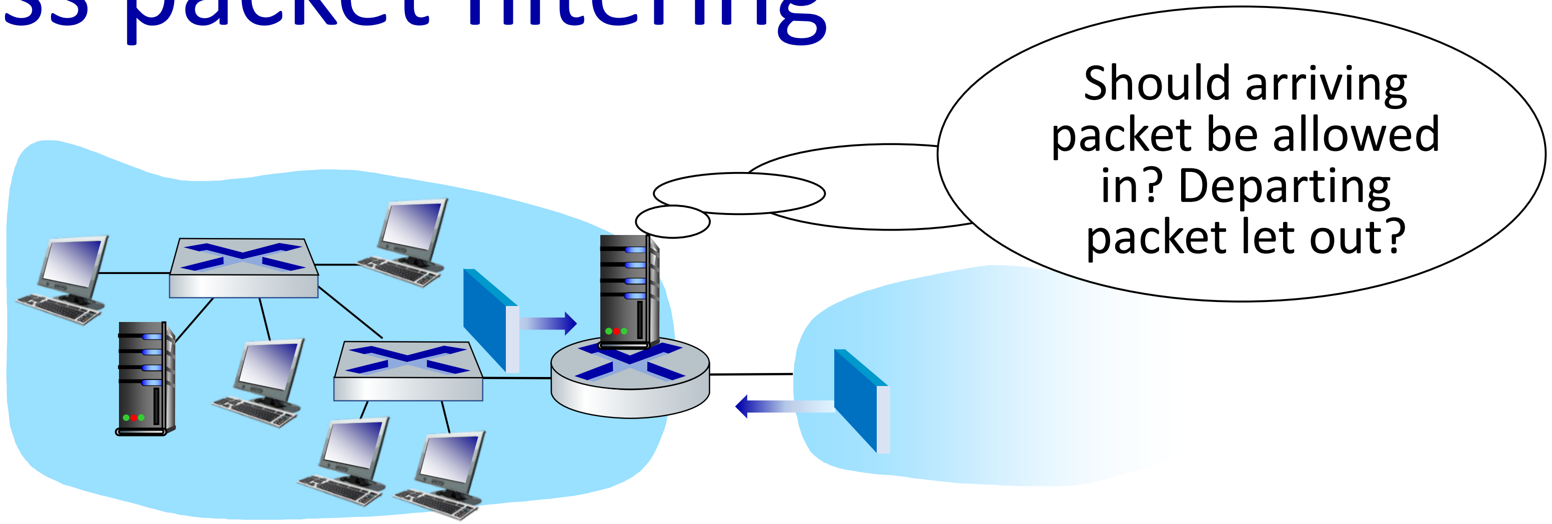
allow only authorized access to inside network

- set of authenticated users/hosts

three types of firewalls:

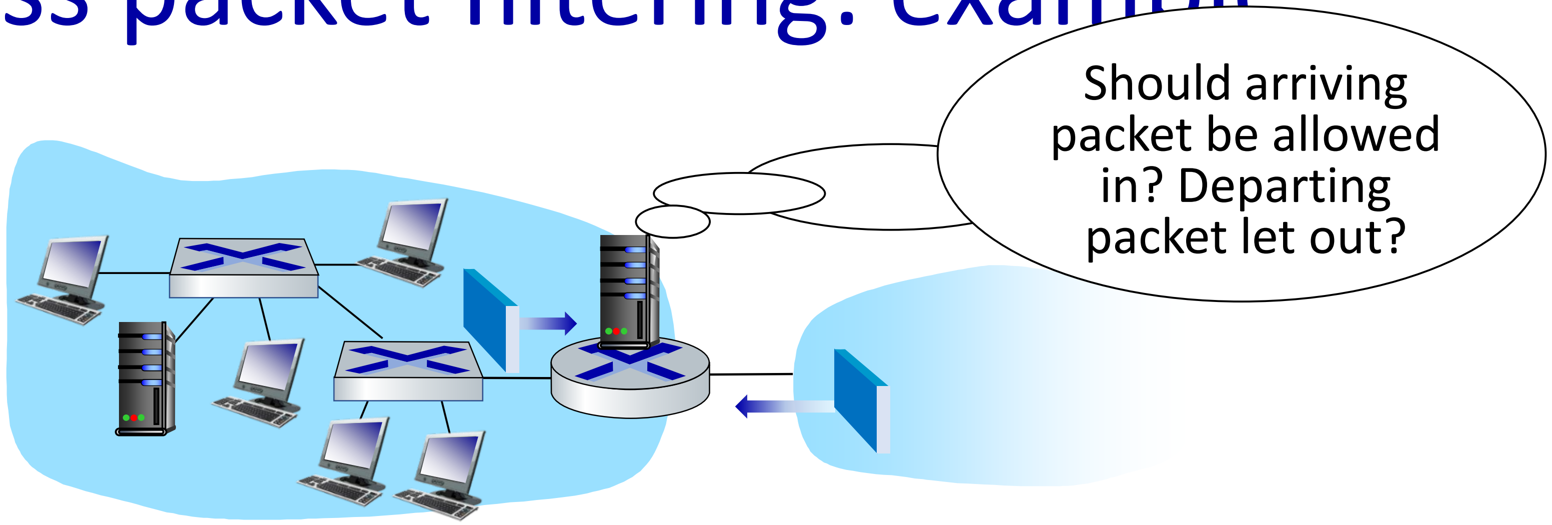
- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- internal network connected to Internet via router firewall
- filters packet-by-packet, decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source, destination port numbers
  - ICMP message type
  - TCP SYN, 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
  - result: all incoming, outgoing UDP flows and telnet connections are blocked
- example 2: block inbound TCP segments with ACK=0
  - result: prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside

Stateless packet filtering: more examples

Policy	Firewall Setting
no outside Web access	drop all outgoing packets to any IP address, port 80
no incoming TCP connections, except those for institution's public Web server only.	drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
prevent Web-radios from eating up the available bandwidth.	drop all incoming UDP packets - except DNS and router broadcasts.
prevent your network from being used for a smurf DoS attack	drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255)
prevent your network from being tracerouted	drop all outgoing ICMP TTL expired traffic

Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs: looks like OpenFlow forwarding

action	source address	dest address	protocol	source port	dest port	flag bit
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
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Stateful packet filtering

- *stateless packet filter:* heavy handed tool
  - admits packets that "make no sense," e.g., dest 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

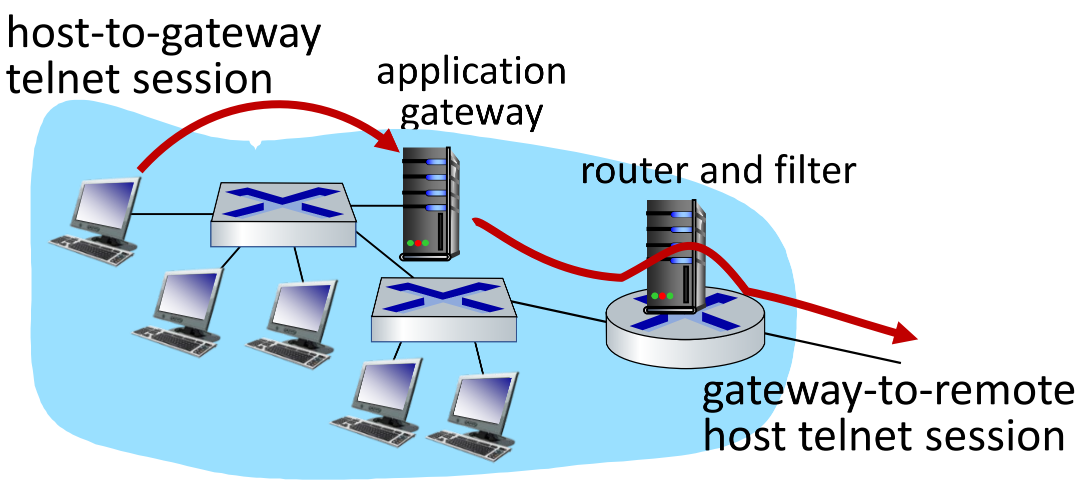
Stateful packet filtering

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 connection
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	

Application gateways

- filter packets on application data as well as on IP/TCP/UDP fields.
- *example:* allow select 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, gateways

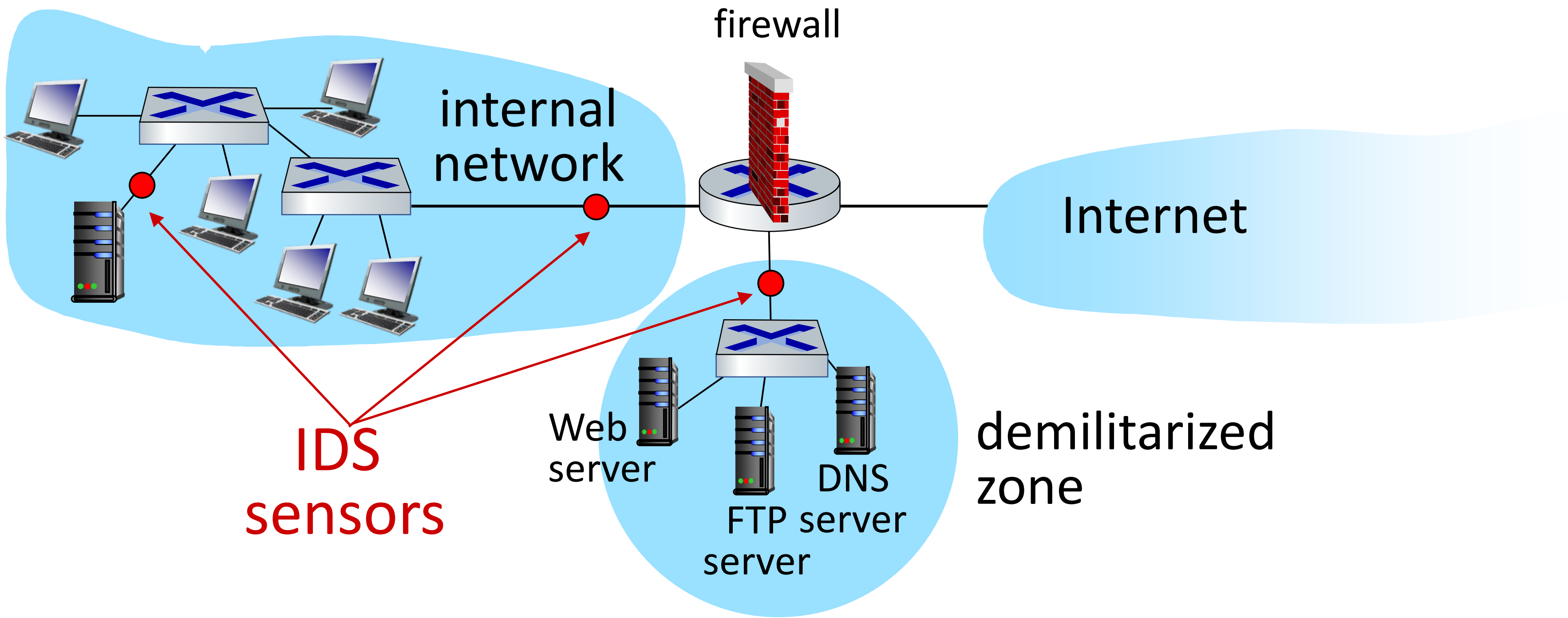
- IP spoofing: router can't know if data "really" comes from claimed source
- if multiple apps need special treatment, each has own app. gateway
- client software must know how to contact gateway
  - e.g., must set IP address of proxy in Web browser
- filters often use all or nothing policy for UDP
-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

Intrusion detection systems

- packet filtering:
  - operates on TCP/IP headers only
  - no correlation check among sessions
- IDS: intrusion detection system
  - deep packet inspection: look at packet contents (e.g., check character strings in packet against database of known virus, attack strings)
  - examine correlation among multiple packets
    - port scanning
    - network mapping
    - DoS attack

Intrusion detection systems

multiple IDSs: different types of checking at different locations



Network Security (summary)

basic techniques…...

- cryptography (symmetric and public key)
- message integrity
- end-point authentication

…. used in many different security scenarios

- secure email
- secure transport (TLS)
- IP sec
- …

operational security: firewalls and IDS