Network Security

Acknowledgements

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

Goals

- understand principles of network security:
  - cryptography and its *many* uses
    - confidentiality
    - authentication
    - message integrity
    - digital signatures
- security in practice:
  - firewalls and intrusion detection systems
  - security in application, transport, network, link layers

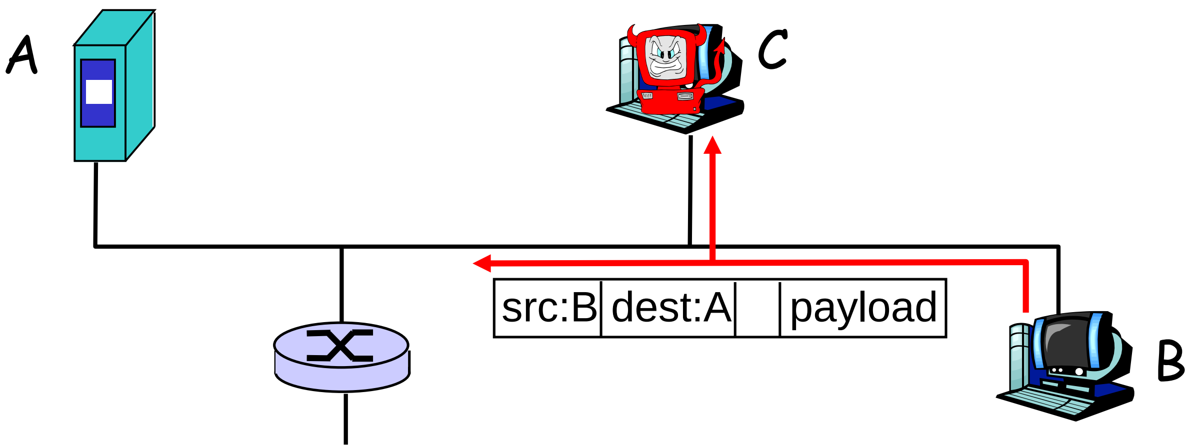
Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

The bad guys can sniff packets

Packet sniffing:

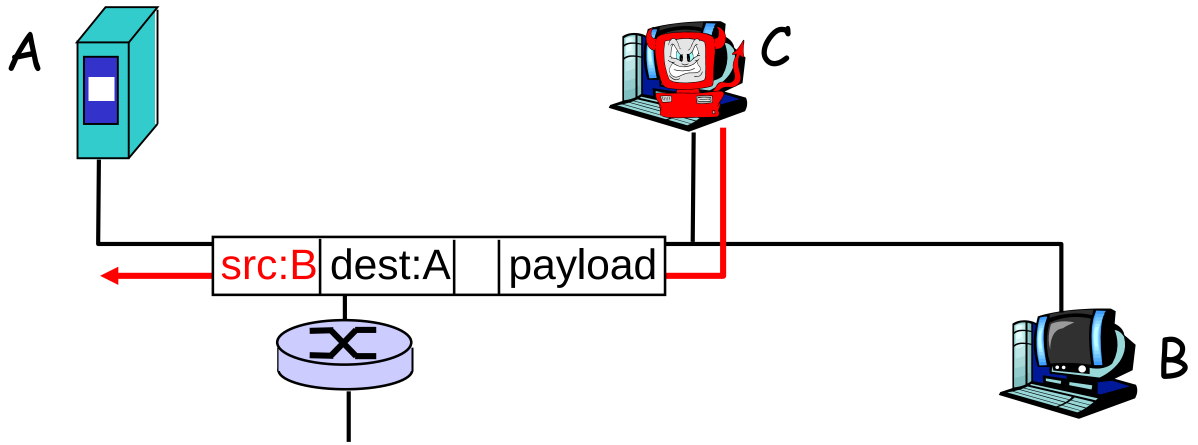
- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



- Wireshark software used for end-of-chapter labs is a (free) packet-sniffer

The bad guys can use false source addresses

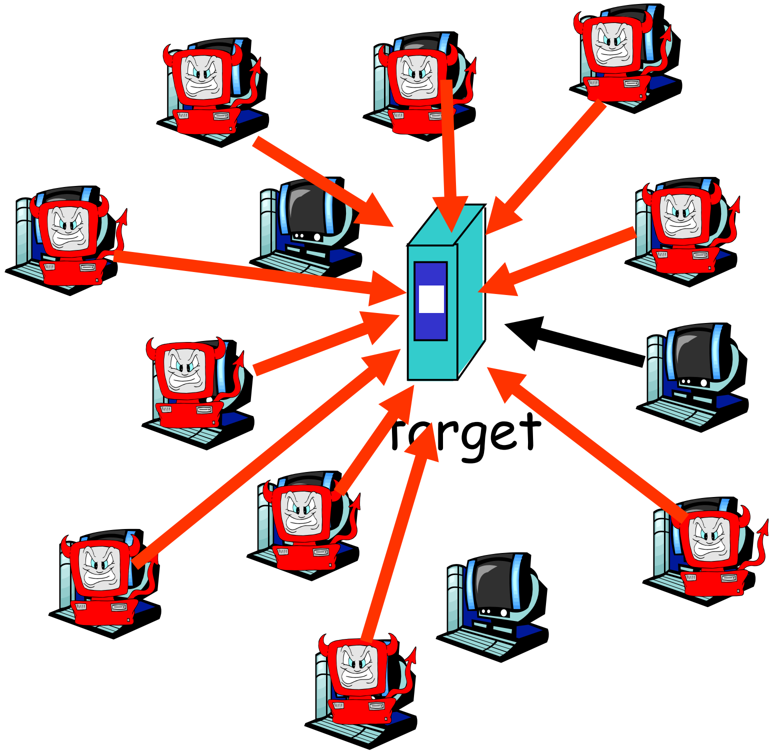
- *IP spoofing:* send packet with false source address



Bad guys can attack servers and network infrastructure

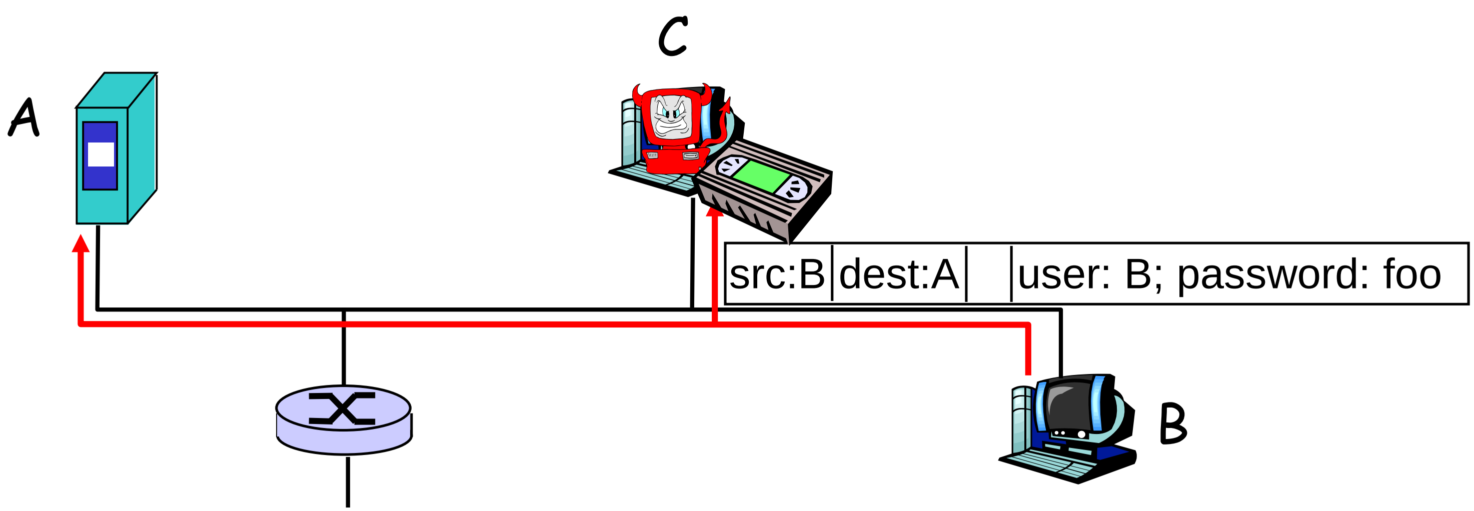
- Denial of service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets toward target from compromised hosts



The bad guys can record and playback

- *record-and-playback*: sniff sensitive info (e.g., password), and use later
  - password holder *is* that user from system point of view



Bad guys can put malware into hosts

- Malware can get in host from a virus, worm, or trojan horse.
- Spyware malware can record keystrokes, web sites visited, upload info to collection site.
- Infected host can be enrolled in a botnet, used for spam and DDoS attacks.
- Malware is often self-replicating: from an infected host, seeks entry into other hosts

Bad guys can put malware into hosts

- Trojan horse
  - Hidden part of some otherwise useful software
  - Today often on a Web page (Active-X, plugin)
- Virus
  - infection by receiving object (e.g., e-mail attachment), actively executing
  - self-replicating: propagate itself to other hosts, users
- Worm
  - infection by passively receiving object that gets itself executed
  - self- replicating: propagates to other hosts, users

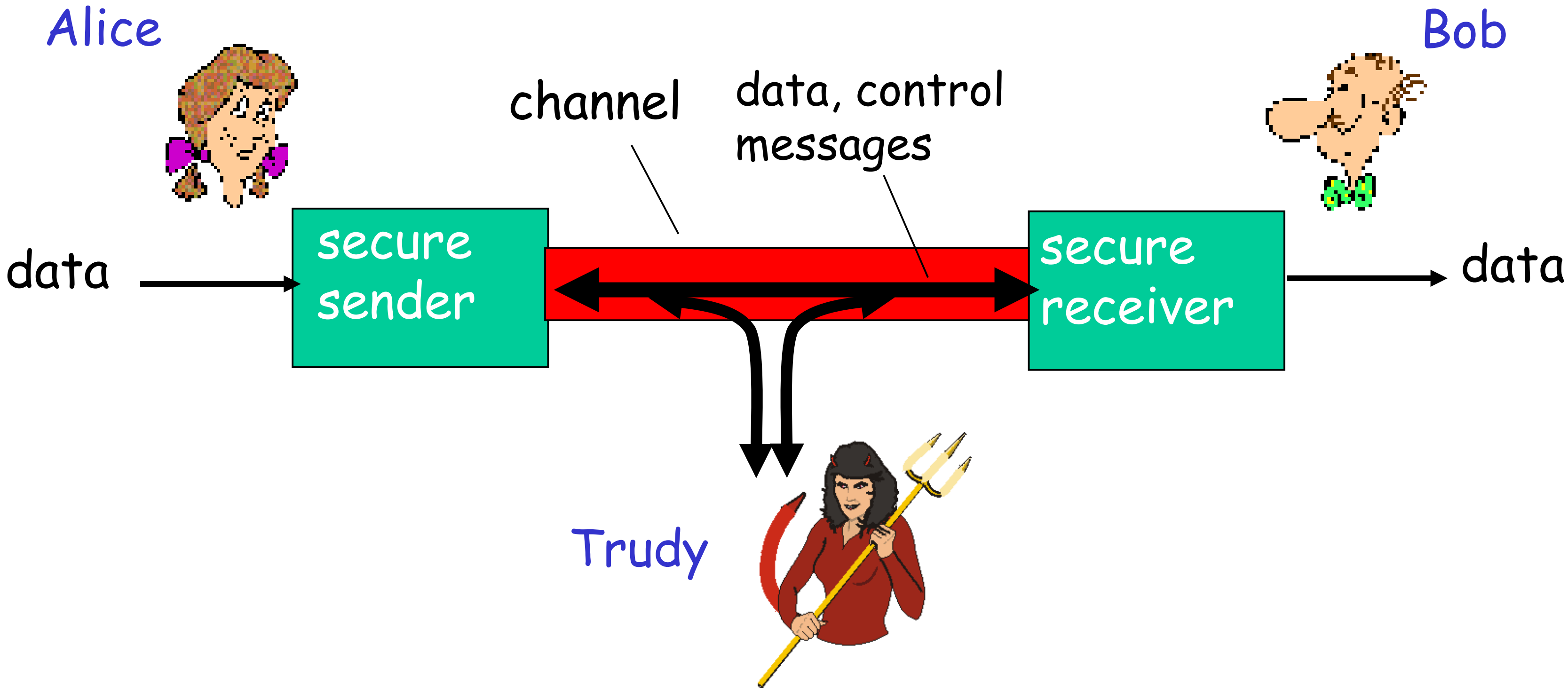
Key question

How to protect from bad guys?

Network Security!

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



Who might Bob, Alice be?

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

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

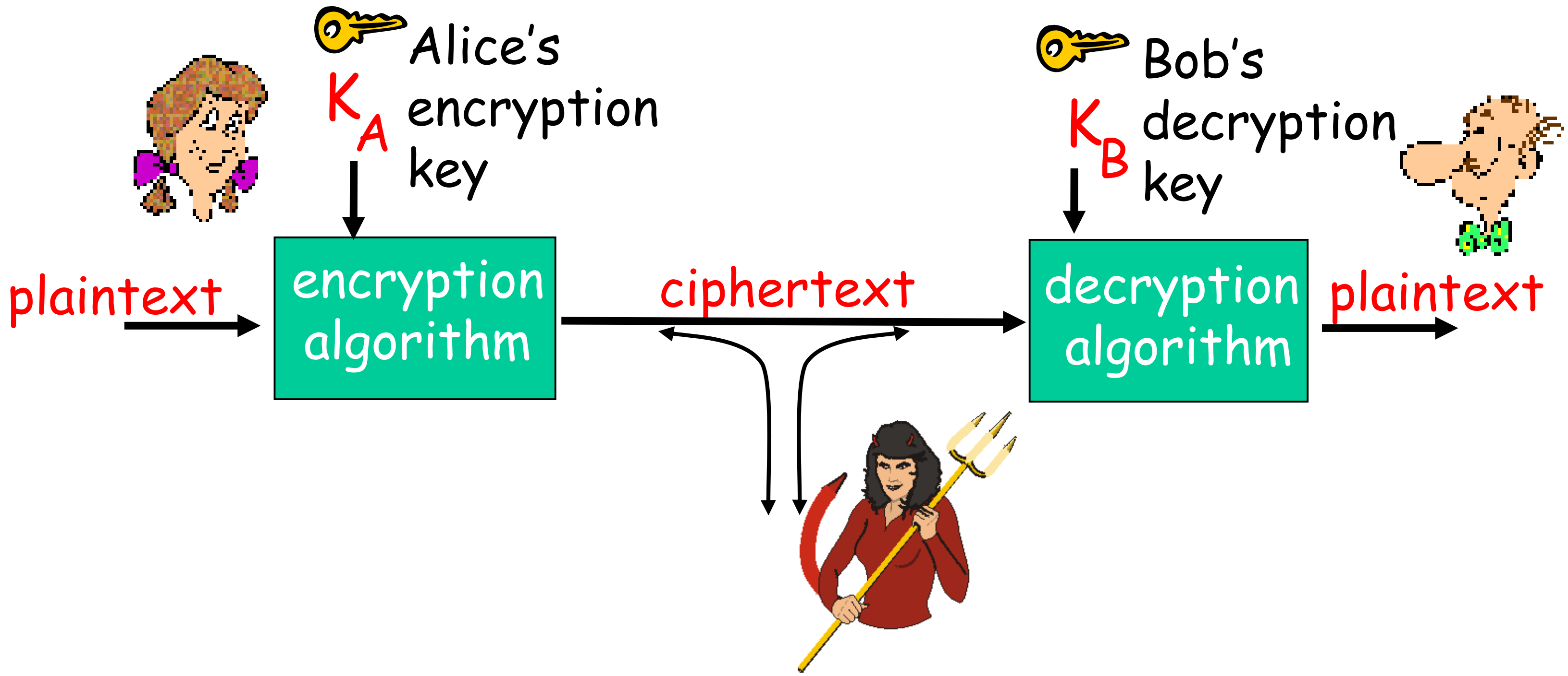
Message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

Access and availability: services must be accessible and available to users

Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

The language of cryptography



m plaintext message

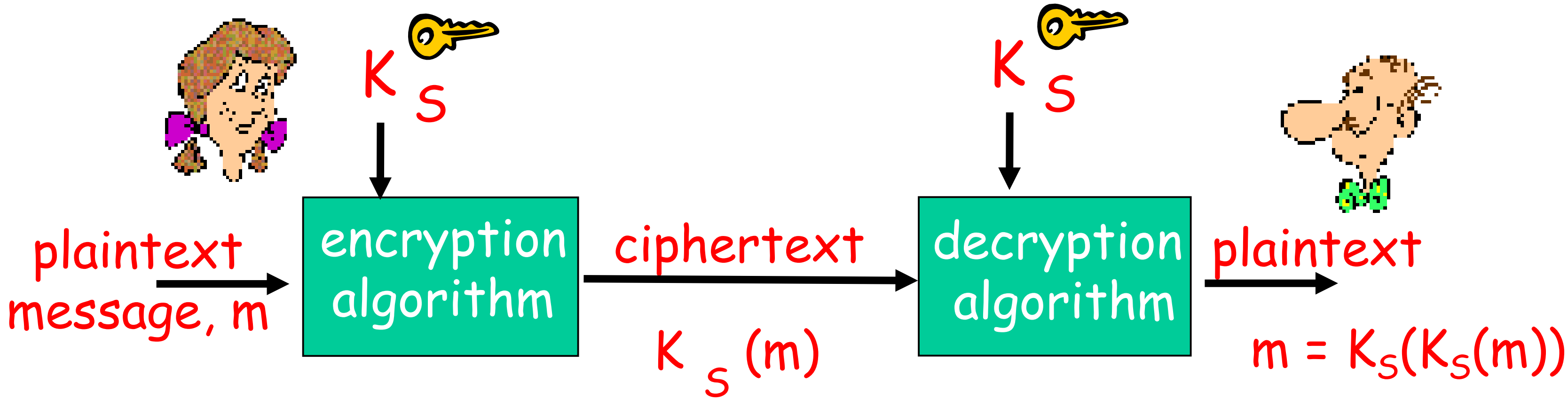
$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Types of Cryptography

- Crypto often uses keys:
  - Algorithm is known to everyone
  - Only "keys" are secret
- Public key cryptography
  - Involves the use of two keys
- Symmetric key cryptography
  - Involves the use of one key
- Hash functions
  - Involves the use of no keys
  - Nothing secret: How can this be useful?

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

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

<u>Q:</u> how do Bob and Alice agree on key value?

Cesar cypher

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  ghijklmnopqrstuvwxyzabcdef
```

E.g.:

```
 Plaintext: bob. i love you. alice
ciphertext: huh. o rubk eua. groik
```

Key: offset between the character in the pain text and the corresponding character in the cyphertext

Monoalphabetic cypher

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
```

Key: the mapping from the set of 26 letters to the set of 26 letters

Polyalphabetic encryption

- n monoalphabetic cyphers, $M_1, M_2, \ldots, M_n$
- 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 monoalphabetic pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4
- Key: the n ciphers and the cyclic pattern

Breaking an encryption scheme

- Cipher-text only attack:
  - Trudy has ciphertext that she can analyze
- Two approaches:
  - Search through all keys
  - Statistical analysis
- Known-plaintext attack:
  - trudy has some plaintext corresponding to some ciphertext
  - eg, in monoalphabetic cipher, trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack
  - trudy can get the cyphertext for some chosen plaintext

Two types of symmetric ciphers

- Stream ciphers
  - encrypt one bit at time
- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

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

  (actually encrypt, decrypt, encrypt)

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
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Key Question

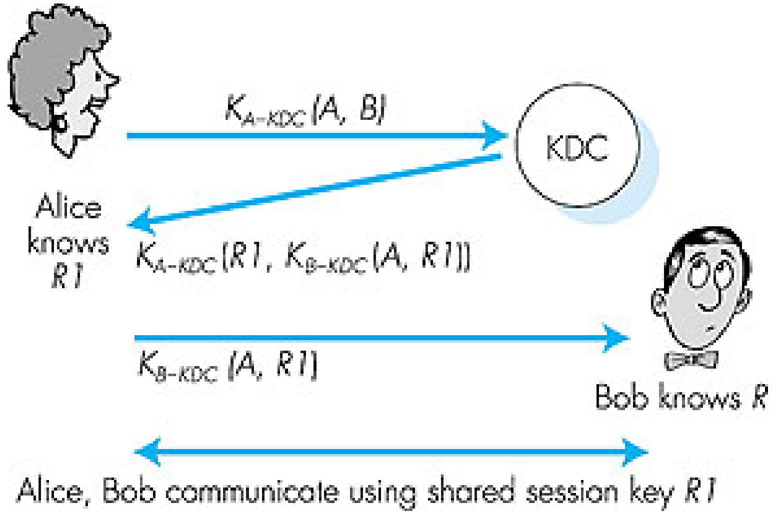
How do two entities establish shared secret key over network?

Solutions:

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

Key Distribution Center (KDC)

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



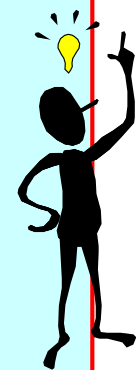
- Alice communicates with KDC, gets session key R1, and $K_{B\text{-}KDC}(A,R1)$
- Alice sends Bob $K_{B\text{-}KDC}(A,R1)$, Bob extracts R1
- Alice, Bob now share the symmetric key R1.

Public Key Cryptography

symmetric key crypto

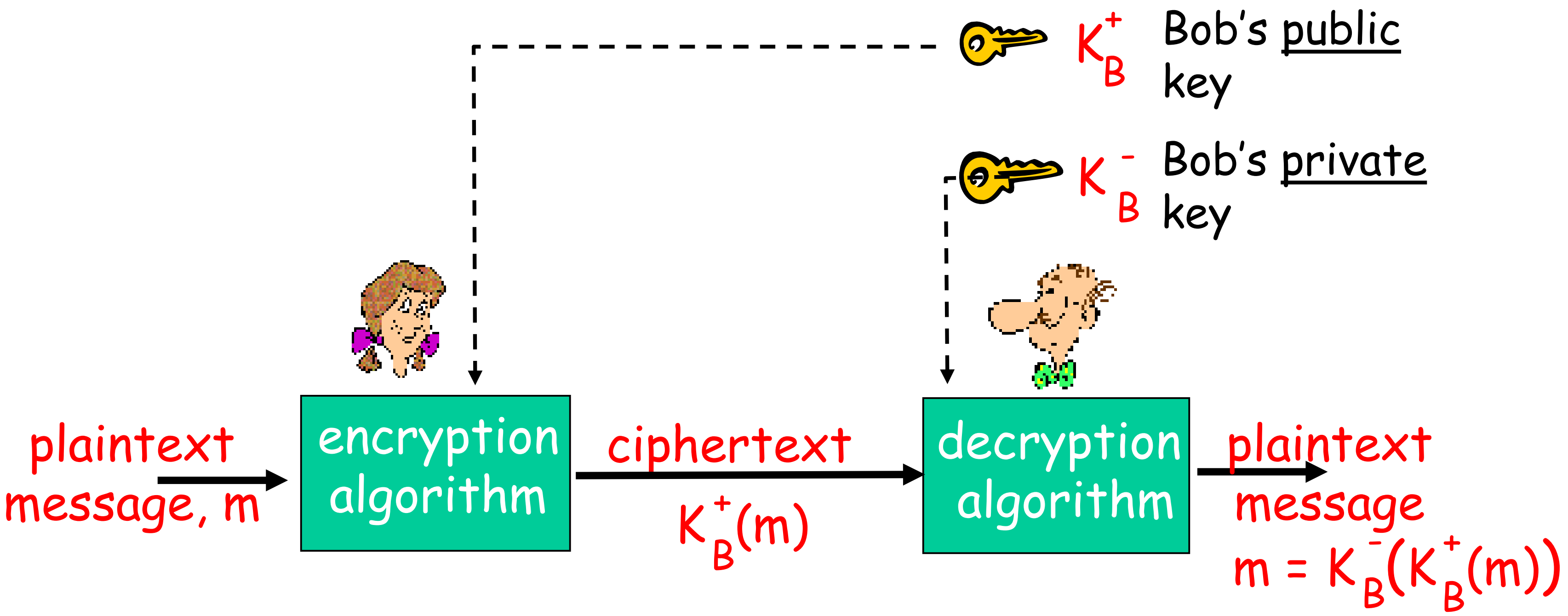
- requires sender, receiver know shared secret key
- How to agree on key in first place
  - particularly if never "met"?

public key cryptography



- 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 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, Adleman 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!

Session keys

- Public key cryptography is computationally intensive
- DES is at least 100 times faster than RSA

Session key, K_S

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

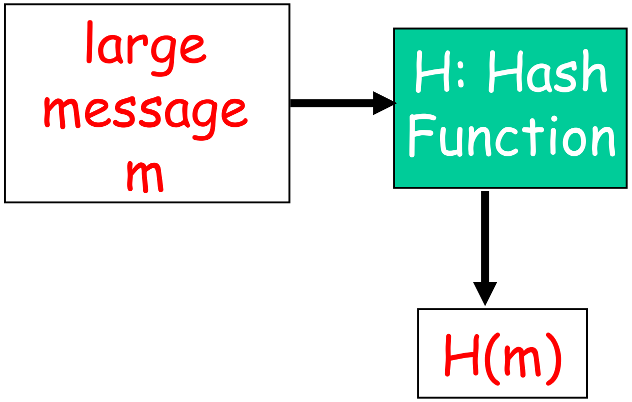
Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- 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



- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"

- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Given [m, H(m)], it must be computationally unfeasible to produce m' (with m<>m') such that H(m) = H(m')
  - Seemingly random output

Internet checksum

Internet checksum has some properties of hash function:

- ✓ produces fixed length digest (16-bit sum) of input
- ✓ is many-to-one
- ❒ But given message with given hash value, it is easy to find another message with same hash value.
- ❒ Example: Simplified checksum: add 4-byte chunks at a time:

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

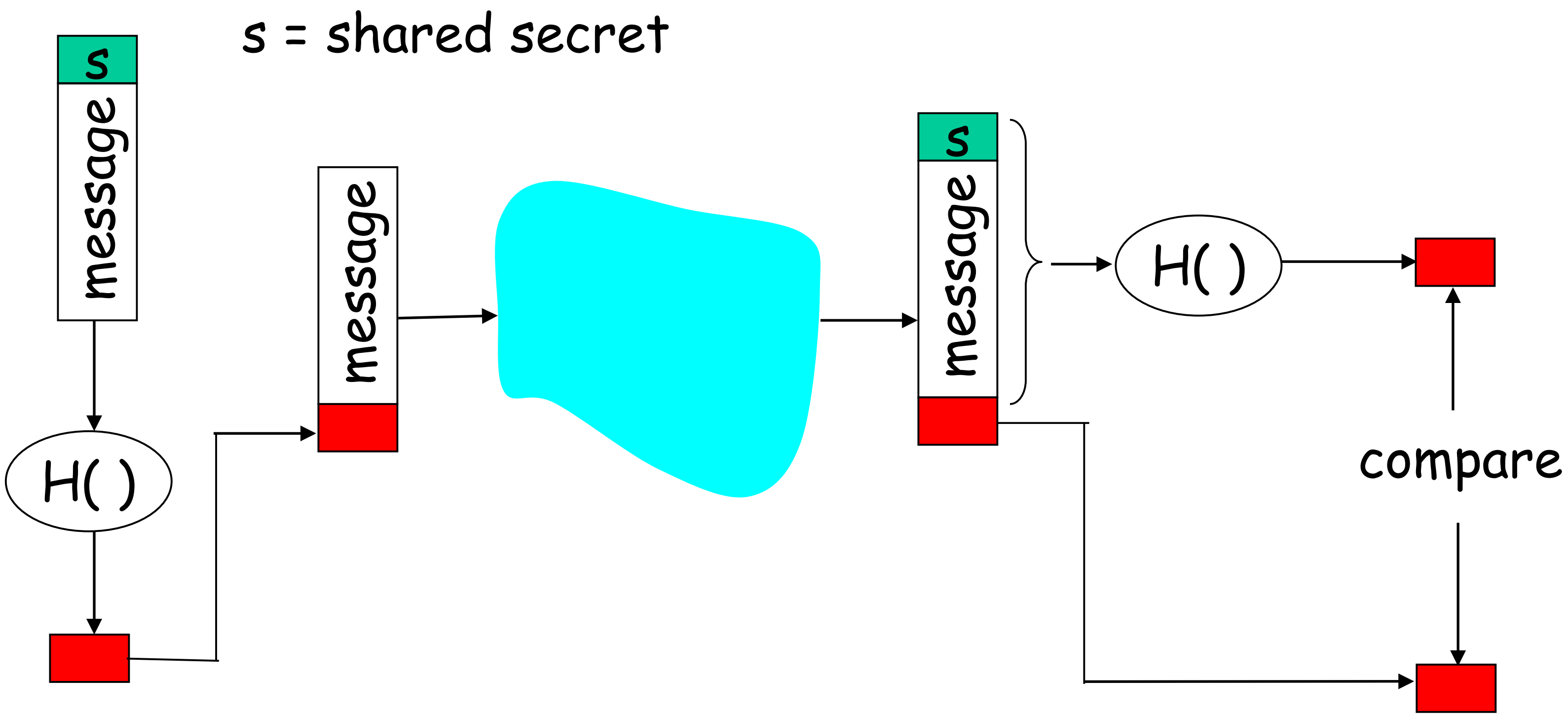
message	ASCII format
I O U 9	49 4F 55 31
0 0 . 1	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

Hash Function Algorithms

- MD5 hash function widely used [Rivest, RFC 1321]
  - computes 128-bit message digest in 4-step process.
  - C source code implementation available in RFC 1321
- SHA-1 is also used.
  - US standard [NIST]
  - 160-bit message digest

Message Authentication Code (MAC)



- ***Authenticates sender***
- ***Verifies message integrity***
- Sender:
  - calculates MAC: H(m||s) ;
  - send [m|| H(m||s)]
- 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])

Example: OSPF

- Recall that OSPF is an intra-AS routing protocol
- Each router creates map of entire AS (or area) and runs shortest path algorithm over map.
- Router receives link-state advertisements (LSAs) from all other routers in AS.

Attacks:

- Message insertion
- Message deletion
- Message modification

- How do we know if an OSPF message is authentic?

OSPF Authentication

- Within an Autonomous System, routers send OSPF messages to each other.
- OSPF provides authentication choices
  - No authentication
  - Shared password: inserted in clear in 64-bit authentication field in OSPF packet
  - Cryptographic hash
- Cryptographic hash with MD5
  - 64-bit authentication field includes 32-bit sequence number
  - MD5 is run over a concatenation of the OSPF packet and shared secret key
  - MD5 hash then appended to OSPF packet; encapsulated in IP datagram

Digital Signature

- Cryptographic technique analogous to hand-written signatures.
  - The sender (Bob) digitally signs document, establishing he is the 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'

Digital Signatures

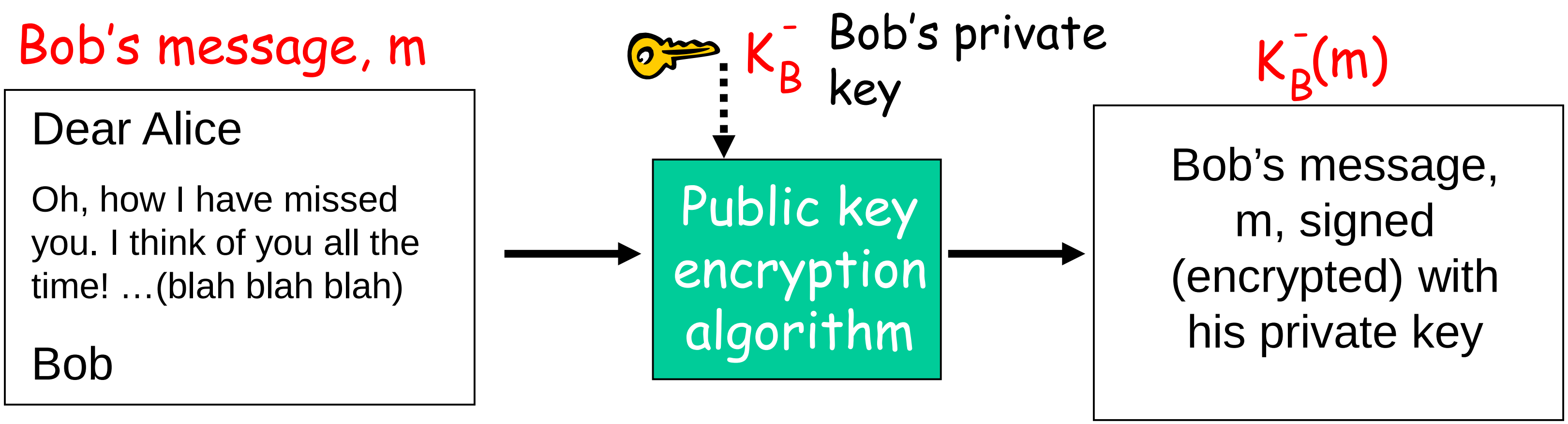
Could we use Message Authentication Code as a Digital Signature??

- 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

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.

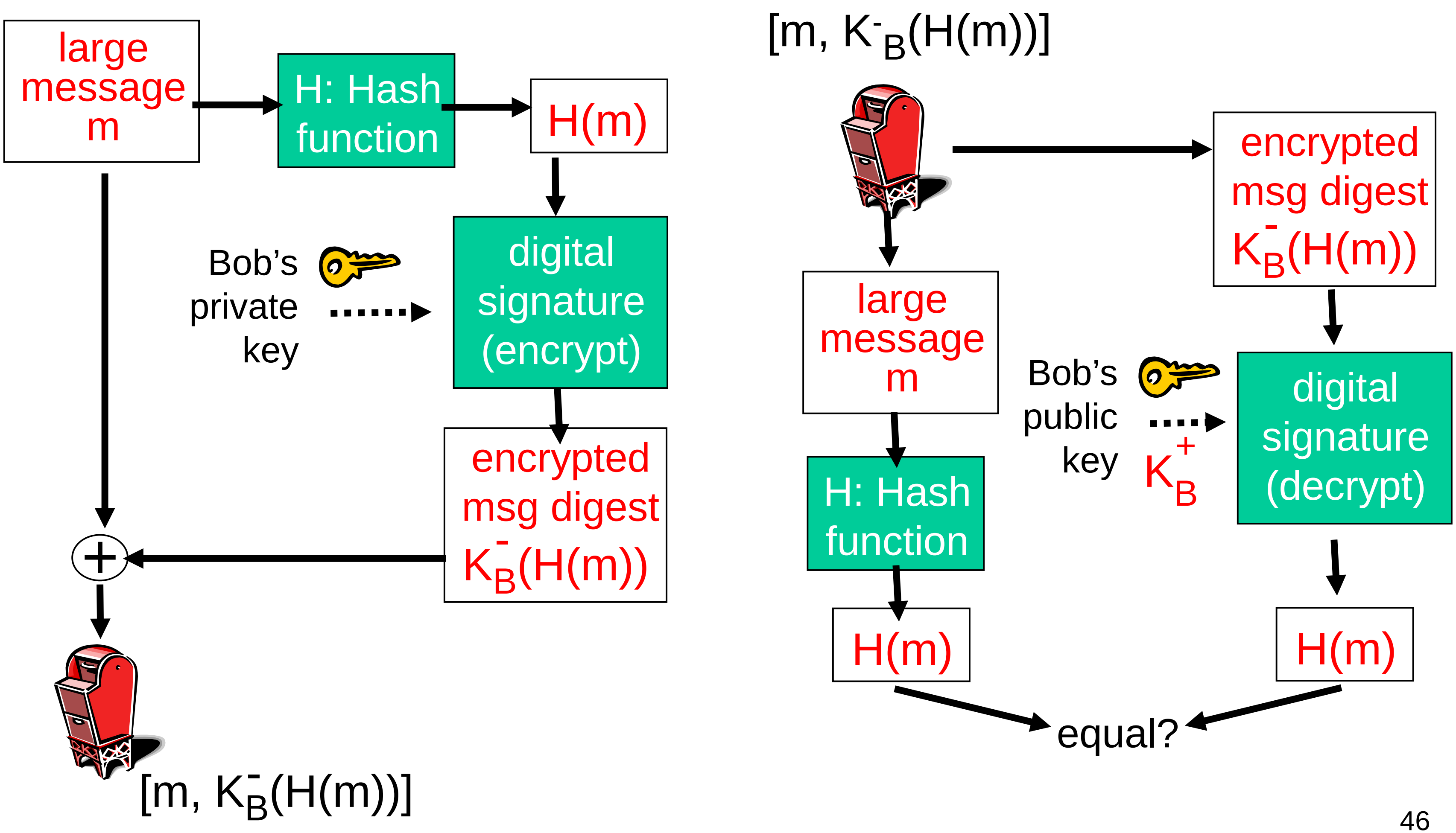
Are requirements satisfied?

- 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 Integrity
  - Bob can prove that he signed m and not *m'*.

Signed message digest

Bob sends digitally signed message:

Alice verifies signature and integrity of digitally signed message:



Authentication Code vs. Digital Signature

- MAC: m+s ➔ H(m+s) ➔ [m, H(m+s)]
- DS: m ➔ H(m) ➔ K^-(H(m)) ➔ [m, K^-(H(m))]
- Digital signature is a heavier technique
  - Requires a Public Key Infrastructure (PKI)
- In practice
  - MAC used in OSPF for message integrity
  - MAC also used for transport and network layer solutions
  - DS used in PGP for message integrity and non repudiation

Key Question

- How can Alice achieve Bob's public key?
  - E-mail?
  - Website?
  - ??

Motivation for public-key certification

- Trudy send a message to Alice
  - Trudy creates e-mail message:

    My loved Alice,

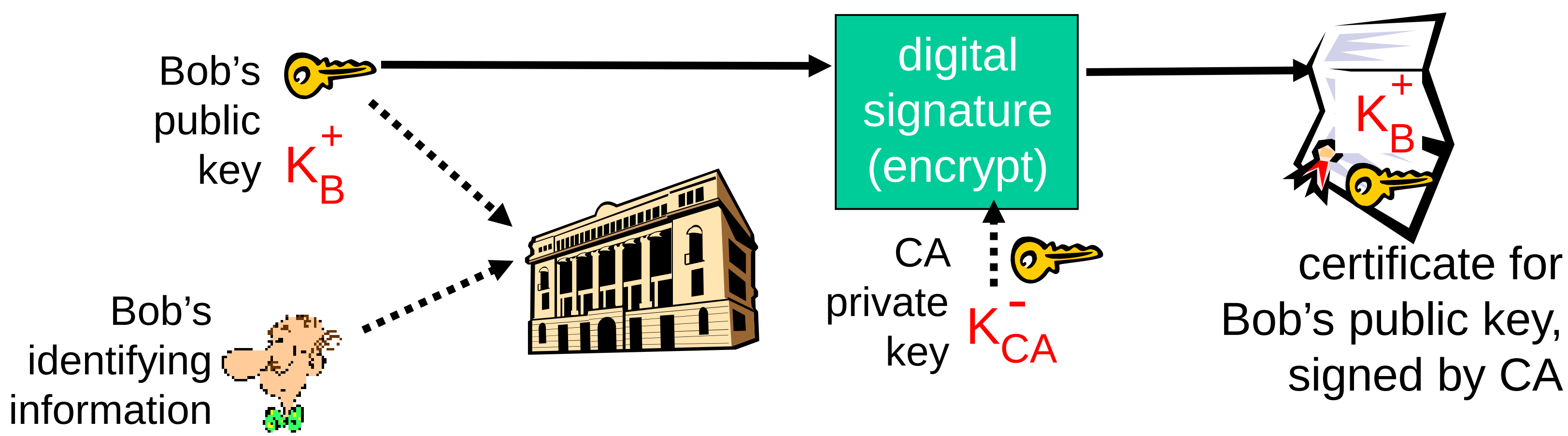
    I also think of you all the time!

    I want to take you in marriage soon!

    Bob
  - 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

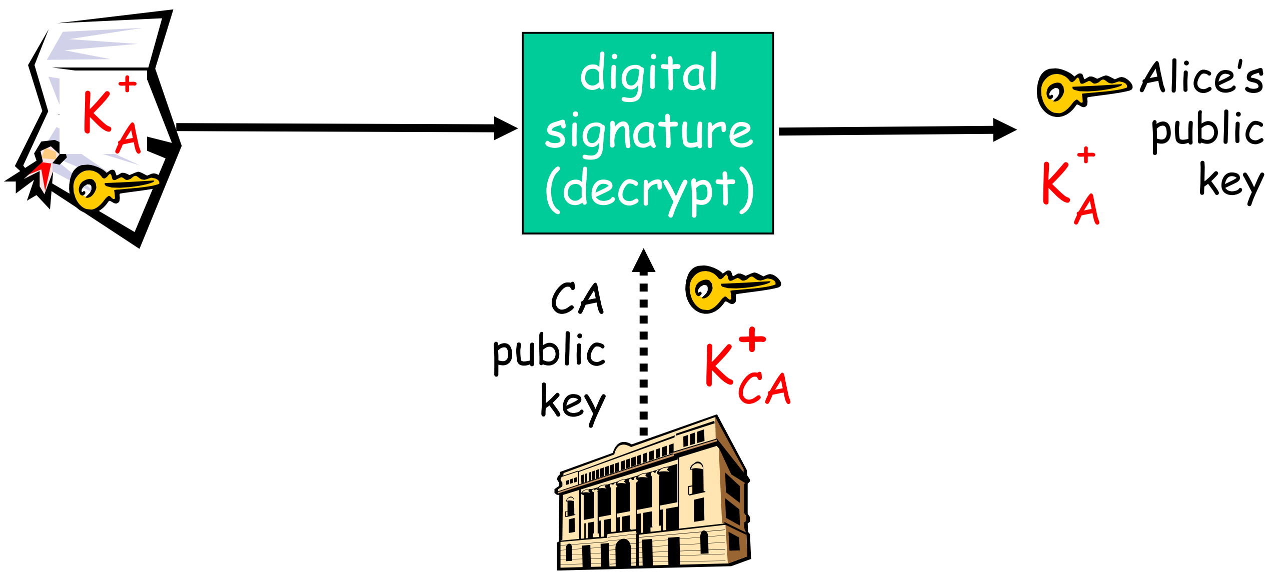
Certification Authorities

- Certification authority (CA):
  - binds public key to particular entity, E.
- E (person, router) 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 Bob wants Alice's public key:
  - gets Alice's certificate (even from Alice).
  - apply CA's public key to Alice's certificate, get Alice's public key



Certificates

- 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"

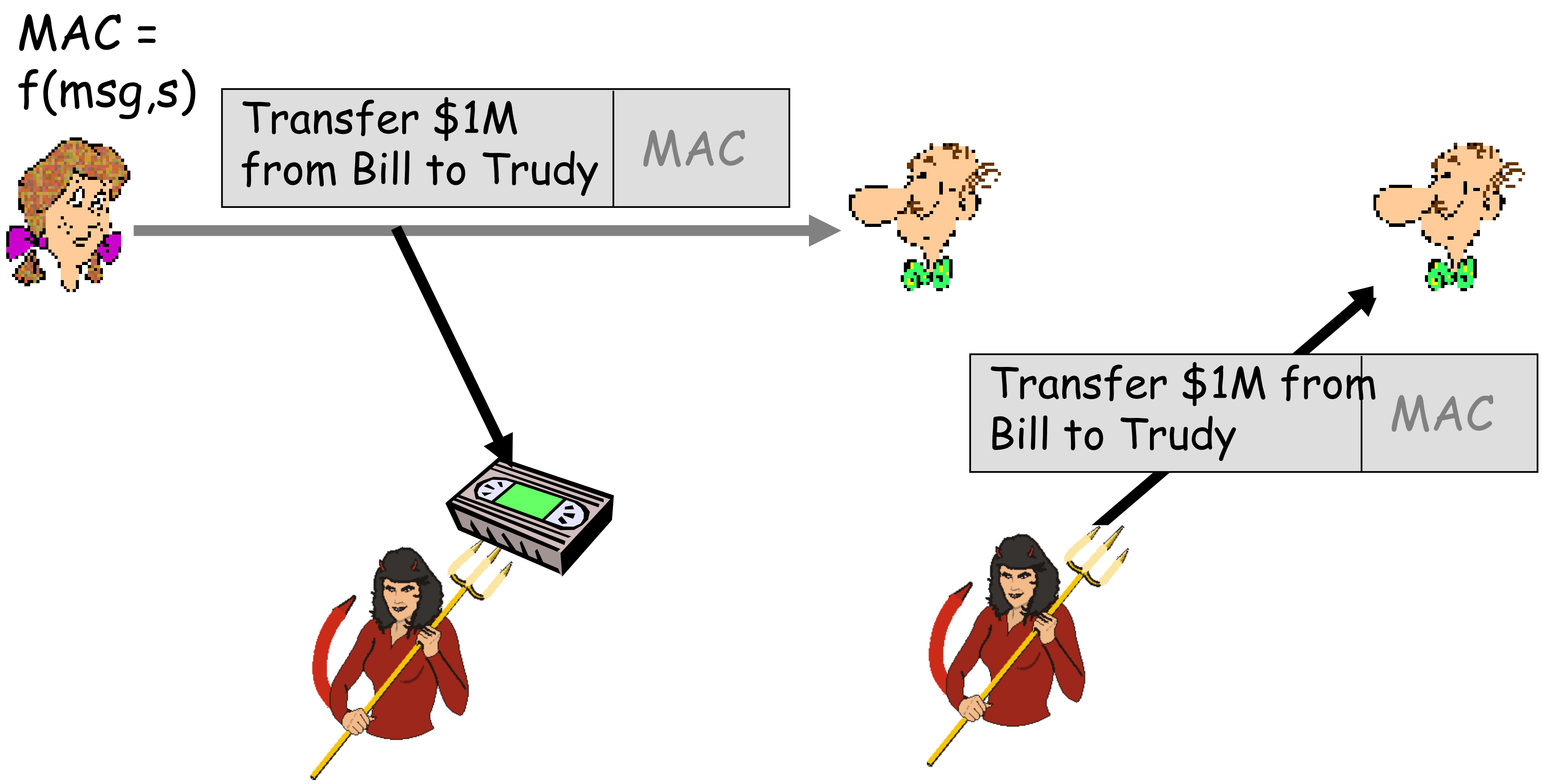
Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

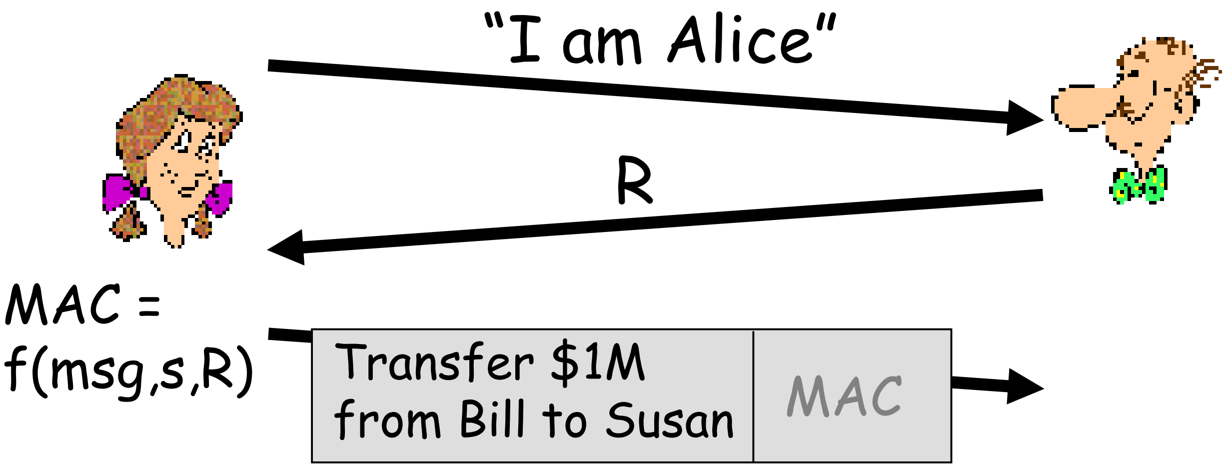
End-point authentication

- Want to be sure of the originator of the message – *end-point authentication.*
- Assuming Alice and Bob have a shared secret, will MAC provide end-point authentication?
  - We do know that Alice created the message.
  - But did she send it?

Playback attack



Defending against playback attack: nonce

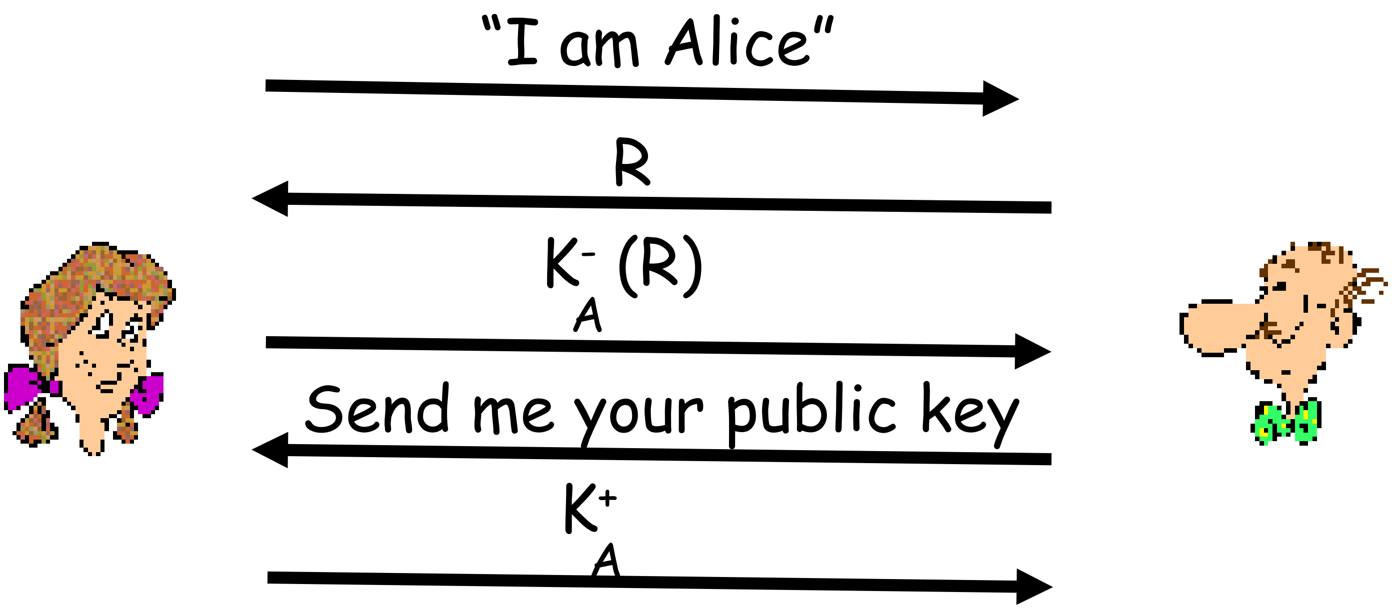


Authentication with public key

MAC requires shared symmetric key

- problem: how do Bob and Alice agree on key?
- can we authenticate using public key techniques?

Solution: use nonce, public key cryptography



A possible security hole

- If Bob does not require a certified public key from Alice
- Man (woman) in the middle attack
  - Trudy poses as Alice (to Bob) and as Bob (to Alice)
- Solution: always use certified public keys

Roadmap

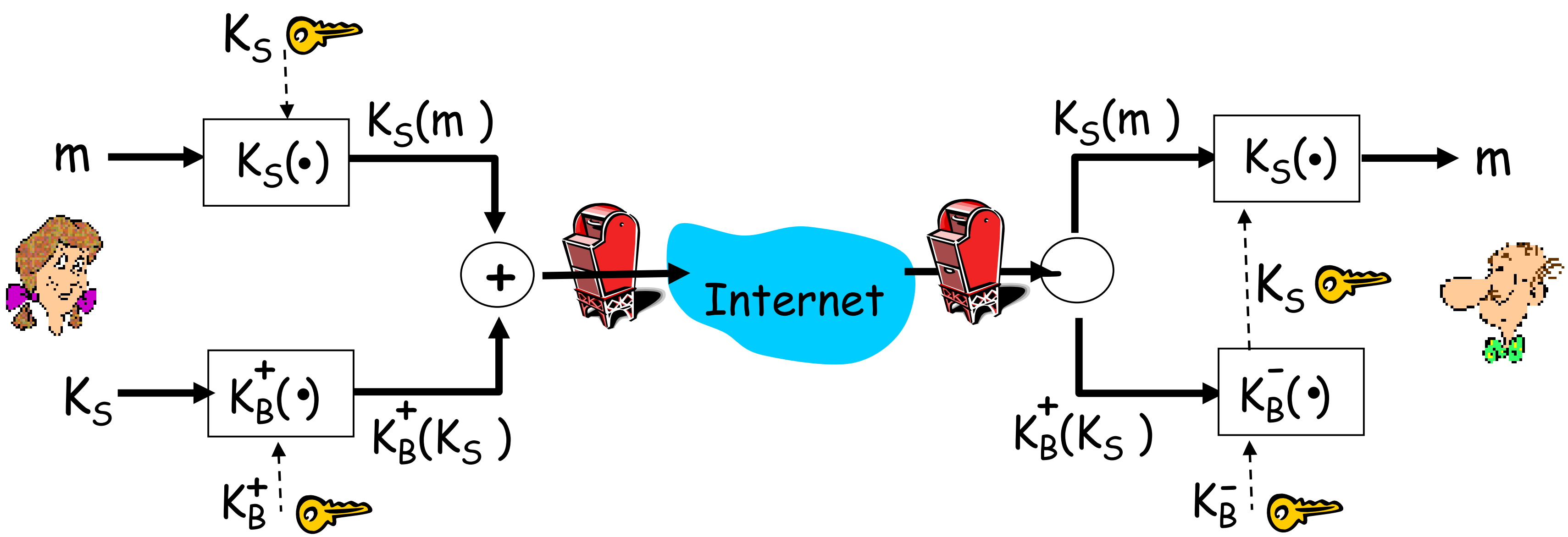
- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

Secure e-mail

- Requirements
  - Confidentiality
  - Sender Authentication
  - Receiver Authentication
  - Message Integrity

Secure e-mail

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

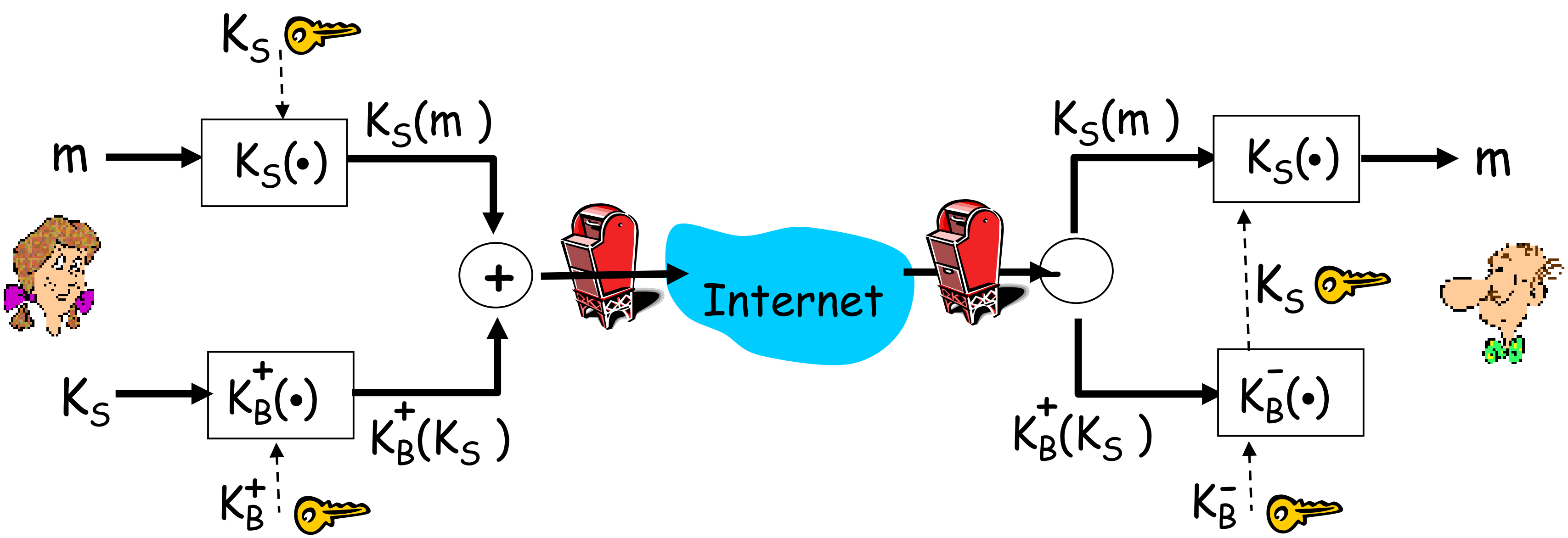


Alice:

- 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

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

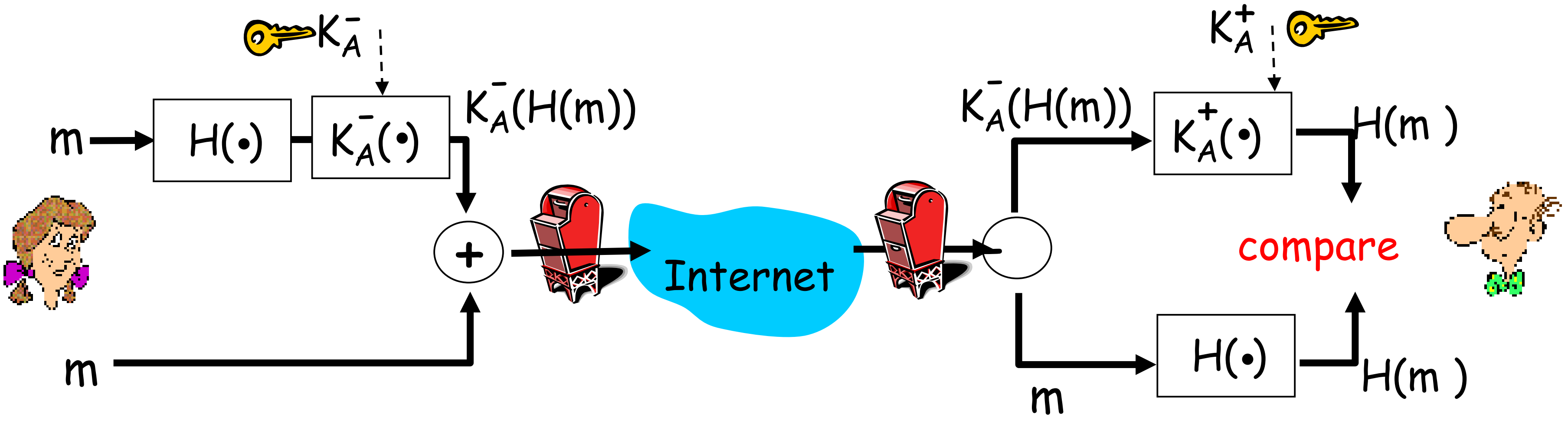


Bob:

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

Secure e-mail (continued)

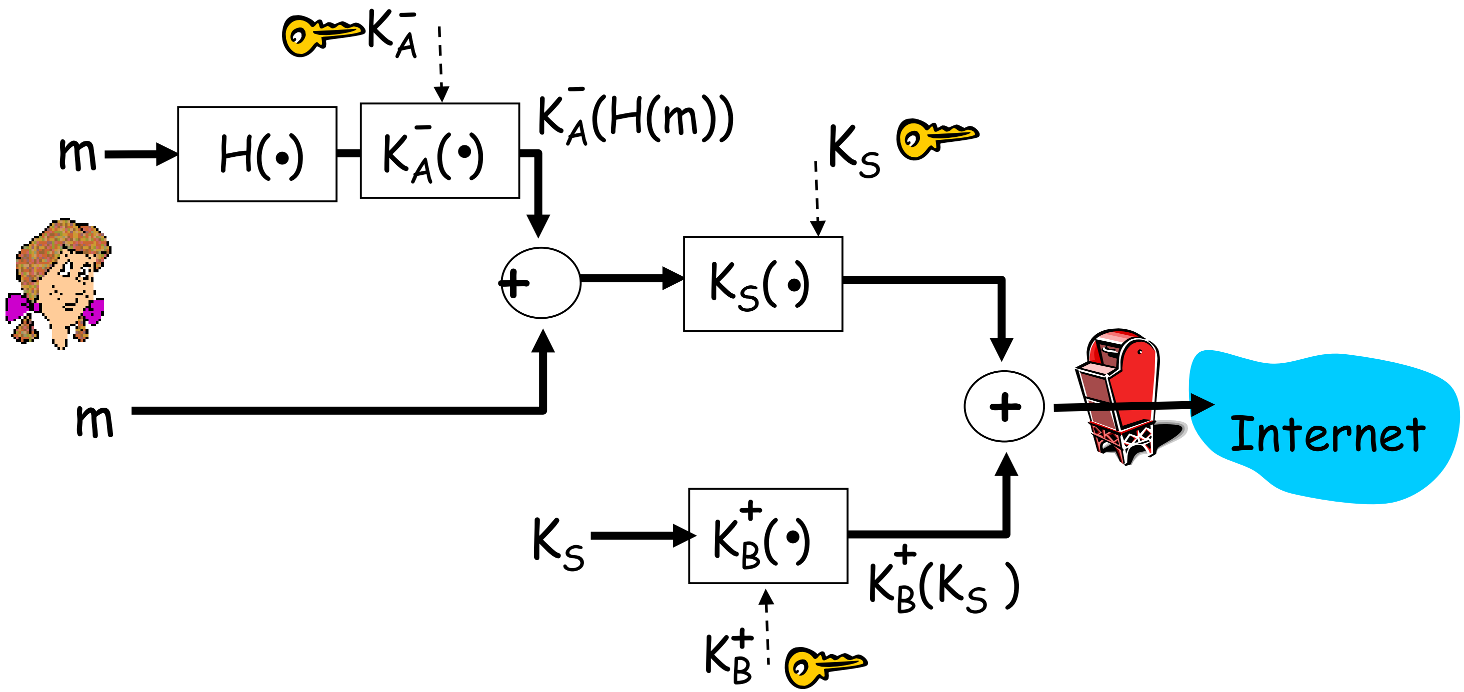
- Alice wants to provide sender authentication message integrity.



- Alice digitally signs message.
- sends both message (in the clear) and digital signature.

Secure e-mail (continued)

- Alice wants to provide *secrecy, sender authentication*, message integrity.



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

Pretty good privacy (PGP)

- Internet e-mail encryption scheme, a de-facto standard.
- Uses symmetric key cryptography, public key cryptography, hash function, and digital signature as described.
- Provides secrecy, sender authentication, integrity.
- Inventor, Phil Zimmerman, was target of 3-year federal investigation.

A PGP signed message:

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

   Bob:
   My husband is out of town
   tonight. Passionately yours,
   Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJ
   hFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

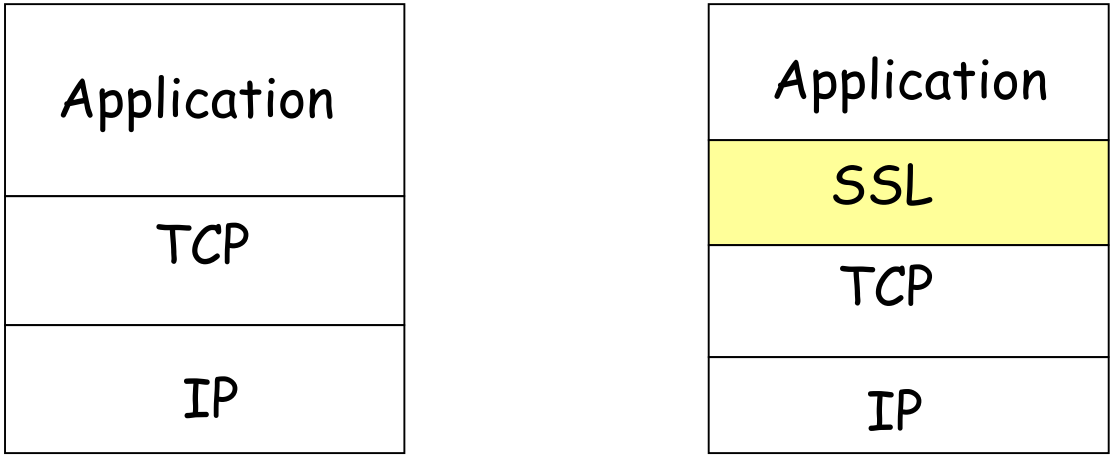
Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

SSL: Secure Sockets Layer

- Widely deployed security protocol
  - Originally designed by Netscape in 1993
  - Supported by almost all browsers and web servers (https)
  - Used by Amazon, eBay, Yahoo!, …
- Number of variations
  - TLS: transport layer security, RFC 2246
- Provides
  - Confidentiality
  - Data Integrity
  - End-point Authentication
- Original goals
  - Web e-commerce transactions in mind
  - Encryption (especially credit-card numbers)
  - Web-server authentication
  - Optional client authentication
- Available to all TCP applications
  - Secure socket interface

SSL and TCP/IP

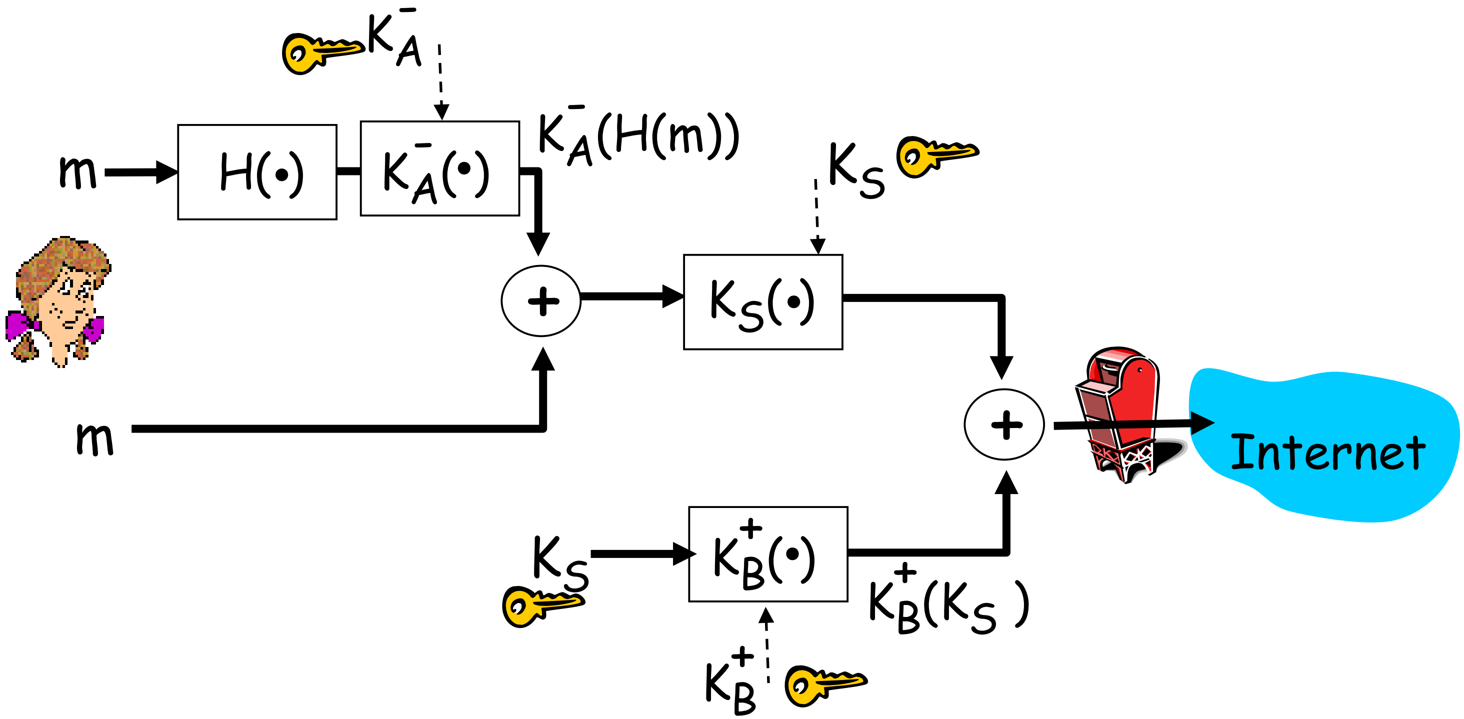


Normal Application

Application with SSL

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Could do something like PGP

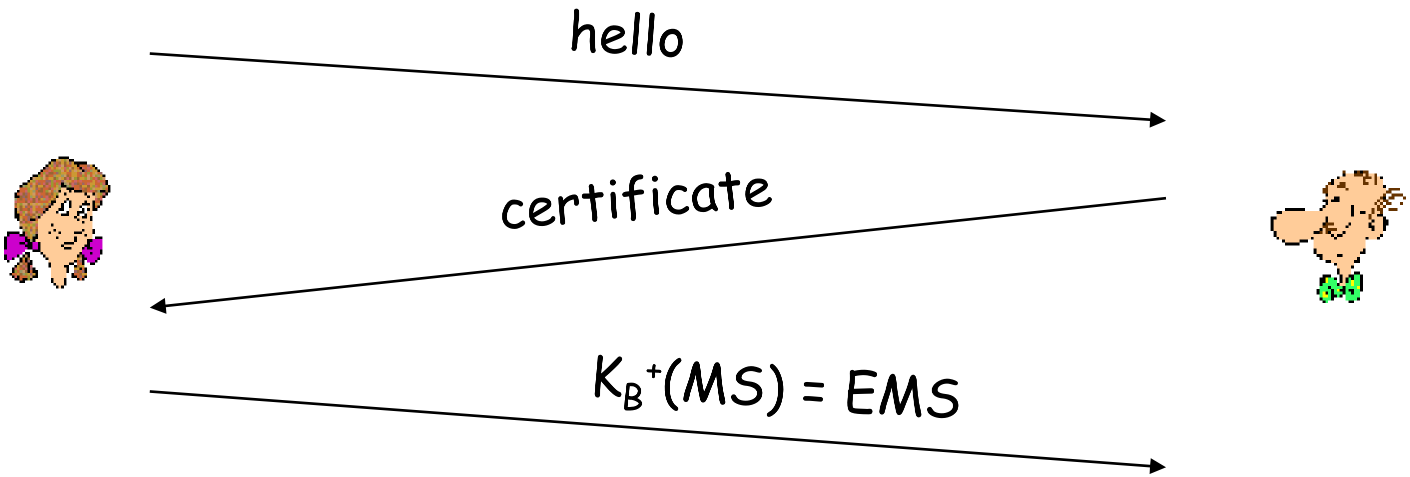


- But want to send byte streams & interactive data
- Want a set of secret keys for the entire connection
- Handshake phase for end-point authentication and keys derivation

Simplified SSL

- **Handshake:** Alice and Bob use their certificates and private keys to authenticate each other and exchange shared secret
- **Key Derivation:** Alice and Bob use shared secret to derive a set of session keys
- **Data Transfer:** Data to be transferred is broken up into a series of records
- **Connection Closure:** Special messages to securely close connection

Simplified SSL: handshake



- MS = master secret
- EMS = encrypted master secret

Simplified SSL: Key derivation

- Considered bad to use same key for more than one cryptographic operation
  - Use 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 and creates the keys

Simplified SSL: Data Records

- Where would we put the MAC?
  - If at end, no message integrity until all data processed.
  - For example, with instant messaging, how can we do integrity check over all bytes sent before displaying?
- Instead, break stream in series of records
  - Each record carries a MAC
  - Receiver can act on each record as it arrives
- Issue: in record, receiver needs to distinguish MAC from data
  - Want to use variable-length records

Simplified SSL: Sequence Numbers

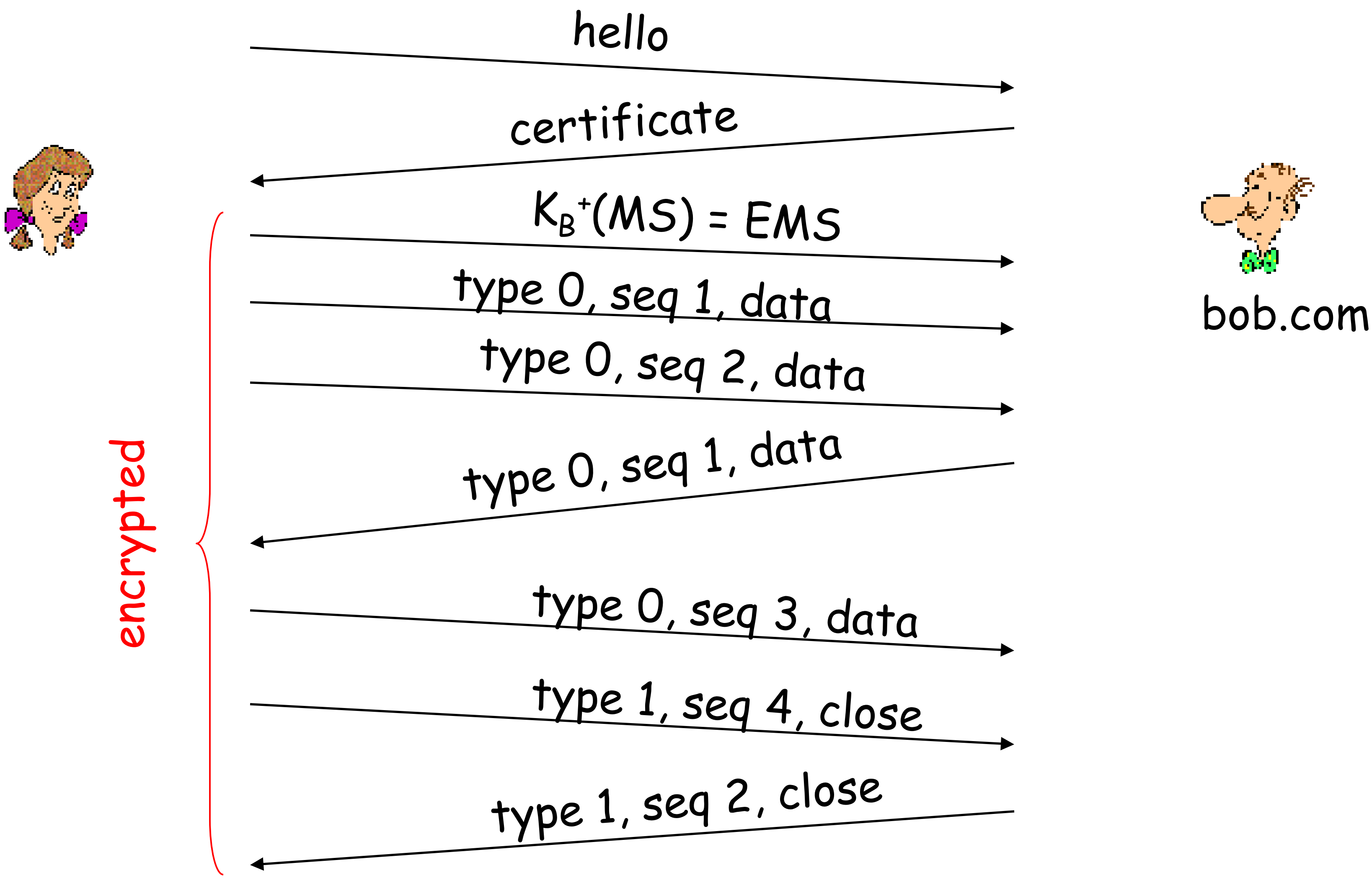
- Attacker can capture and replay record or re-order records
  - e.g., changing the seqnum in TCP segments
- Solution: put sequence number into MAC:
  - MAC = MAC(M_x, sequence||data)
  - Note: no sequence number field
- Attacker could still replay all of the records
  - Server sends a random nonce with its public key certificate (see Real SSL, later)

Simplified SSL: Control information

- 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 closure
- MAC = MAC(M_x, sequence||type||data)

vers	type	length	data	MAC

Simplified SSL: Summary



Simplified SSL isn't complete

- What encryption protocols?
- No negotiation
  - client and server should support different encryption algorithms
  - client and server should choose together specific algorithm before data transfer

Most common symmetric ciphers in SSL

- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- AES – Advanced Encryption Standard : block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

Public key encryption

- RSA

SSL Cipher Suite

- Cipher Suite
  - Public-key algorithm
  - Symmetric encryption algorithm
  - MAC algorithm
- Negotiation: client and server must agree on cipher suite
- Client offers choice; server picks one

Real SSL: Handshake (1)

Purpose

1. Server authentication
2. Negotiation: agree on crypto algorithms
3. Establish keys
4. Client authentication (optional)

Real SSL: Handshake (2)

1. Client sends list of algorithms it supports, along with client nonce
2. Server chooses algorithms from list; sends back: choice + certificate + server nonce
3. Client verifies certificate, extracts server's public key, generates Pre-Master-Secret, (PMS), encrypts PMS with server's public key, sends to server
4. Client and server independently compute encryption and MAC keys from PMS and nonces
5. Client sends a MAC of all the handshake messages
6. Server sends a MAC of all the handshake messages

Real SSL: Handshaking (3)

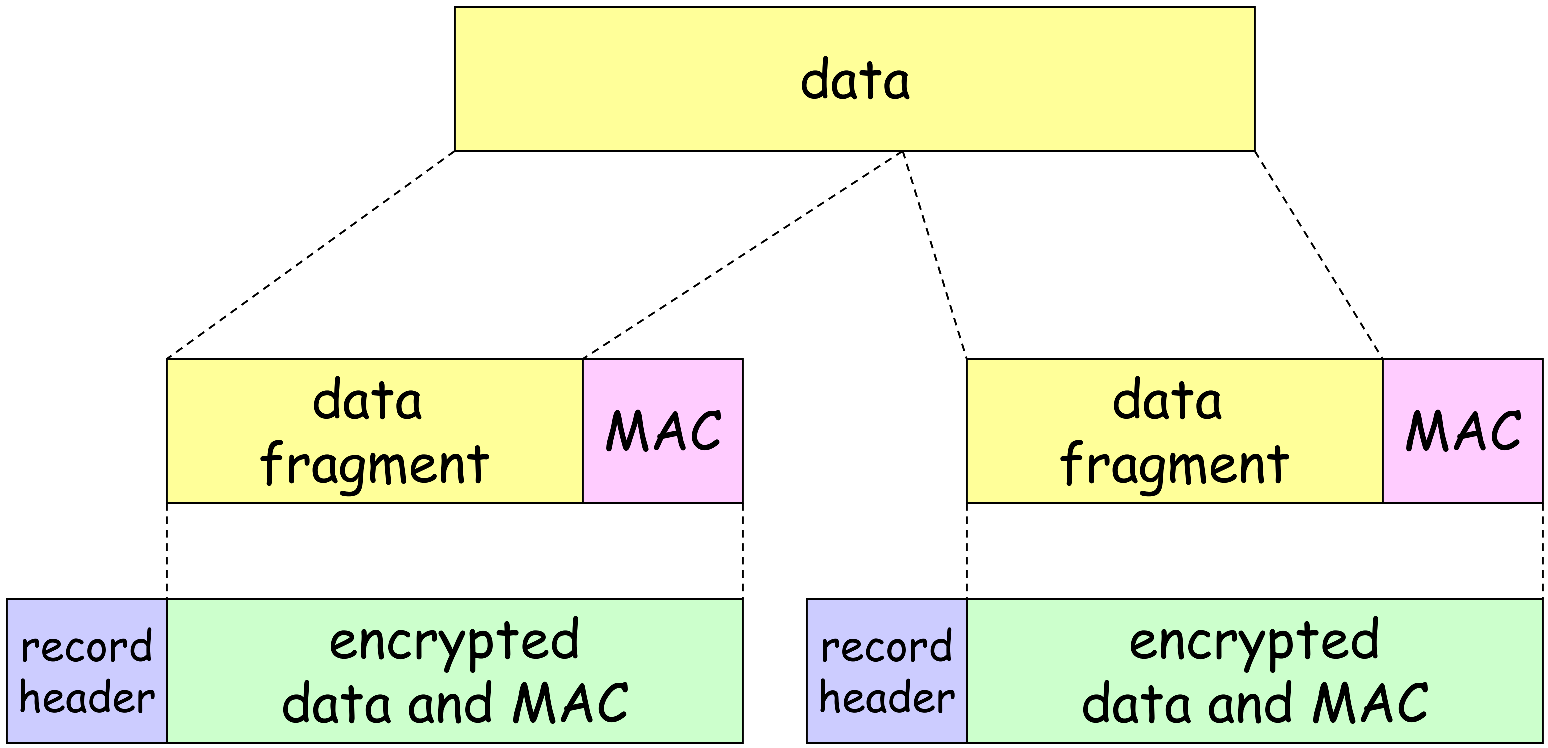
Last 2 steps protect handshake from tampering

- Client typically offers range of algorithms, some strong, some weak
- Man-in-the middle could delete the stronger algorithms from list
- Last 2 steps prevent this

Real SSL: Handshaking (4)

- Why the random nonces?
- Suppose Trudy sniffs all messages between Alice & Bob.
- Next day, Trudy sets up TCP connection with Bob, sends the exact same sequence of records (connection replay attack).
  - Bob (Amazon) thinks Alice made two separate orders for the same thing.
  - Solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days.
  - Trudy's messages will fail Bob's integrity check.

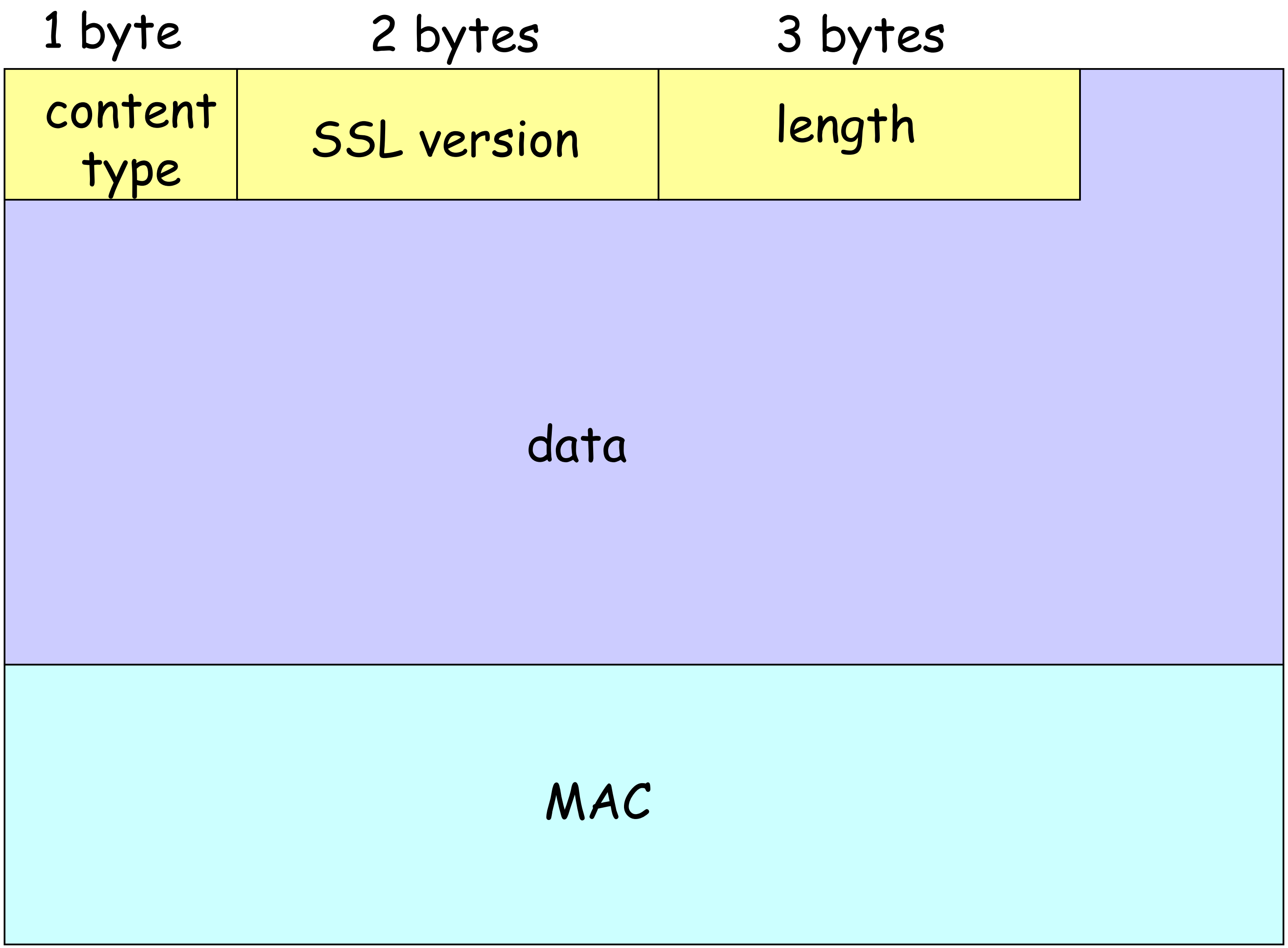
SSL Record Protocol



record header: content type; version; length

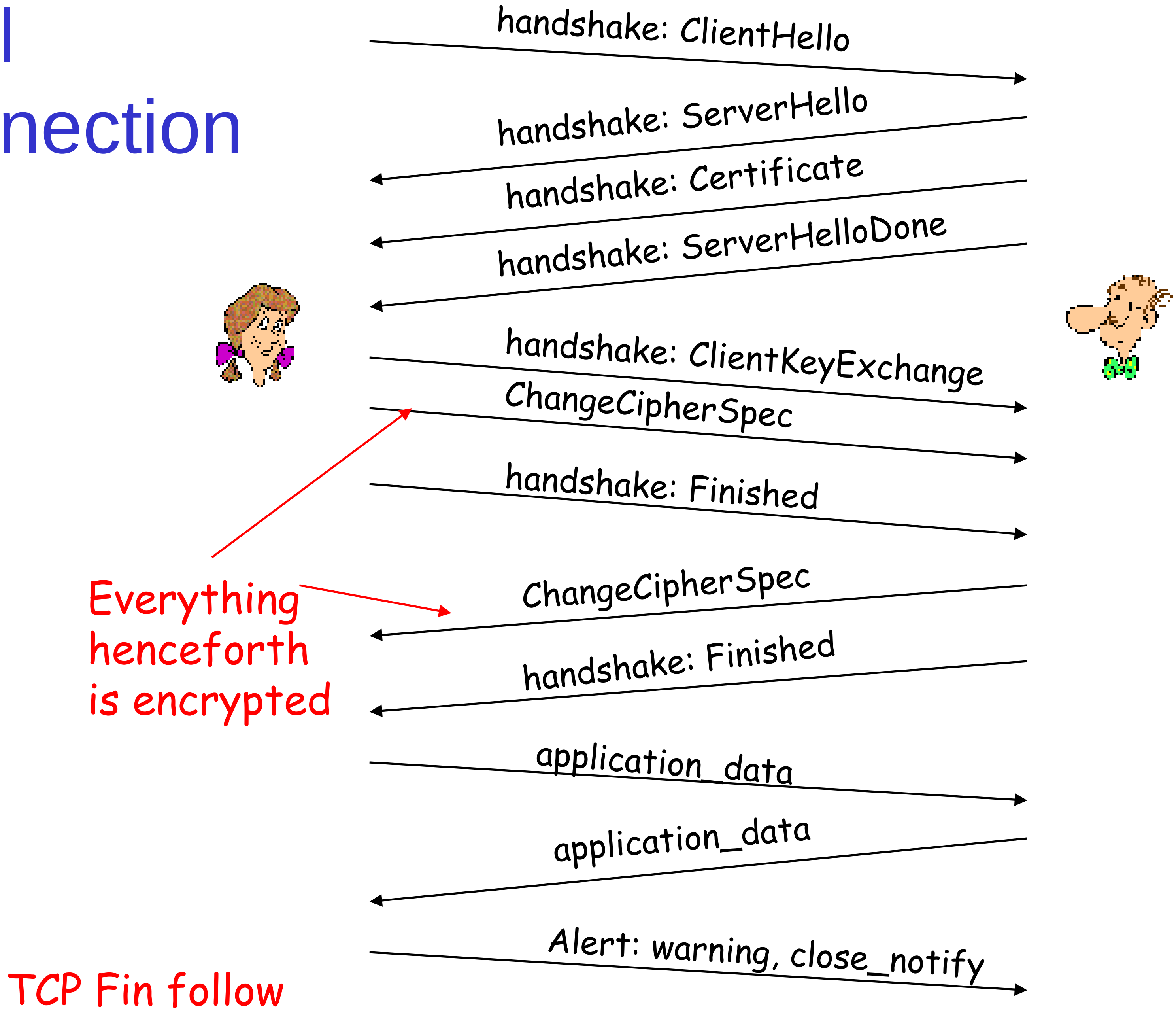
MAC: includes sequence number, MAC key M_x

SSL Record Format



Data and MAC encrypted (symmetric algo)

Real Connection



Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

What is confidentiality at the network-layer?

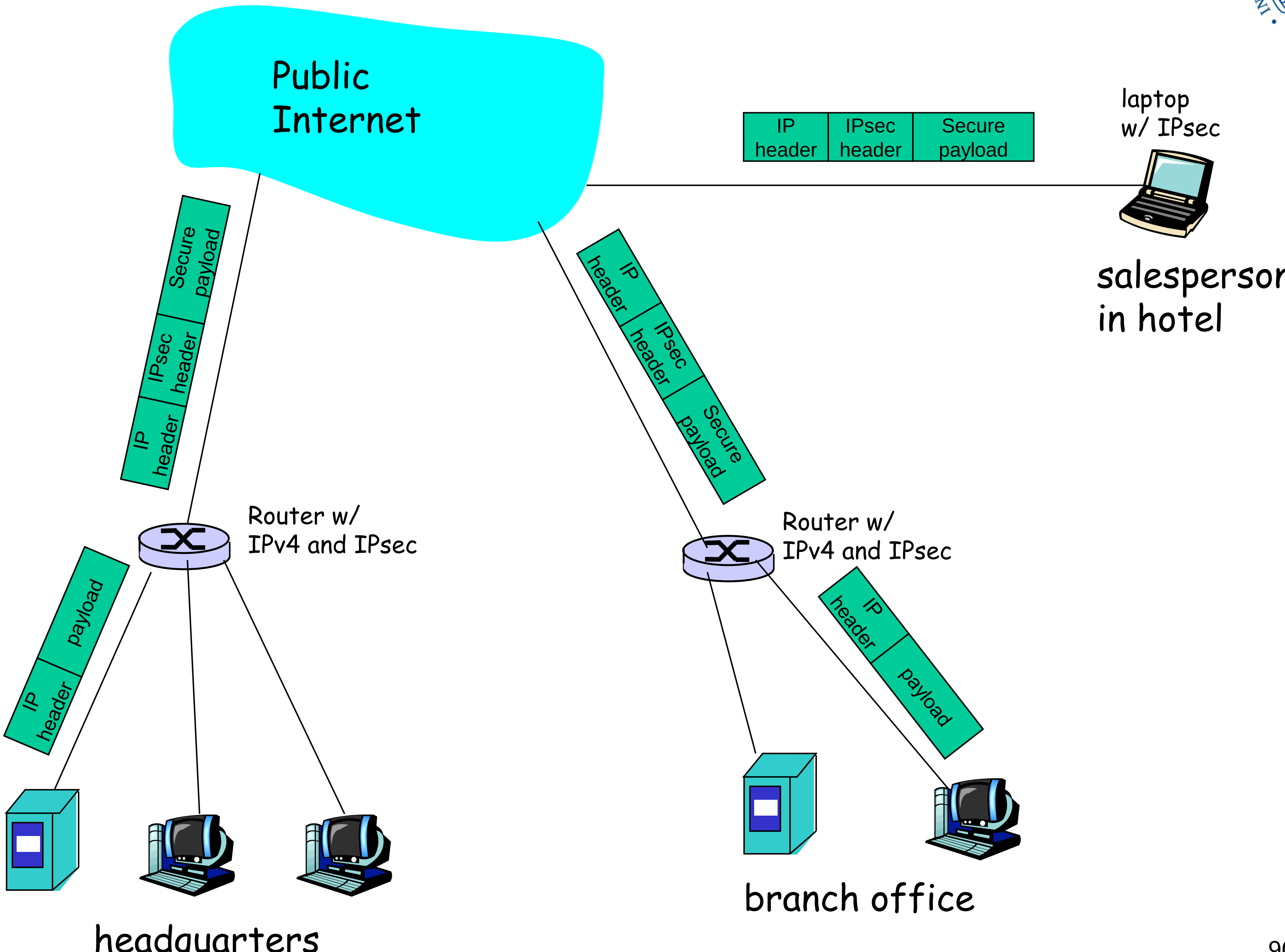
Between two network entities:

- Sending entity encrypts the payloads of datagrams. Payload could be:
  - TCP segment, UDP segment, ICMP message, OSPF message, and so on.
- All data sent from one entity to the other would be hidden:
  - Web pages, e-mail, P2P file transfers, TCP SYN packets, and so on.
  - That is, “blanket coverage”.
- Additional services
  - Source authentication, data integrity, replay attack prevention

Virtual Private Networks (VPNs)

- 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

Virtual Private Network (VPN)

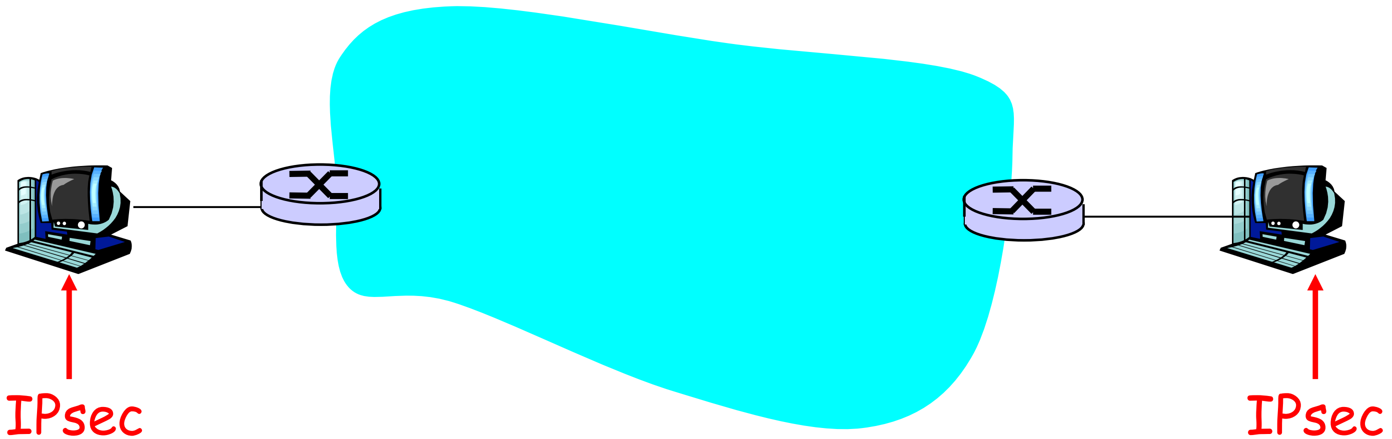


IPsec services

- Data integrity
- Origin authentication
- Replay attack prevention
- Confidentiality

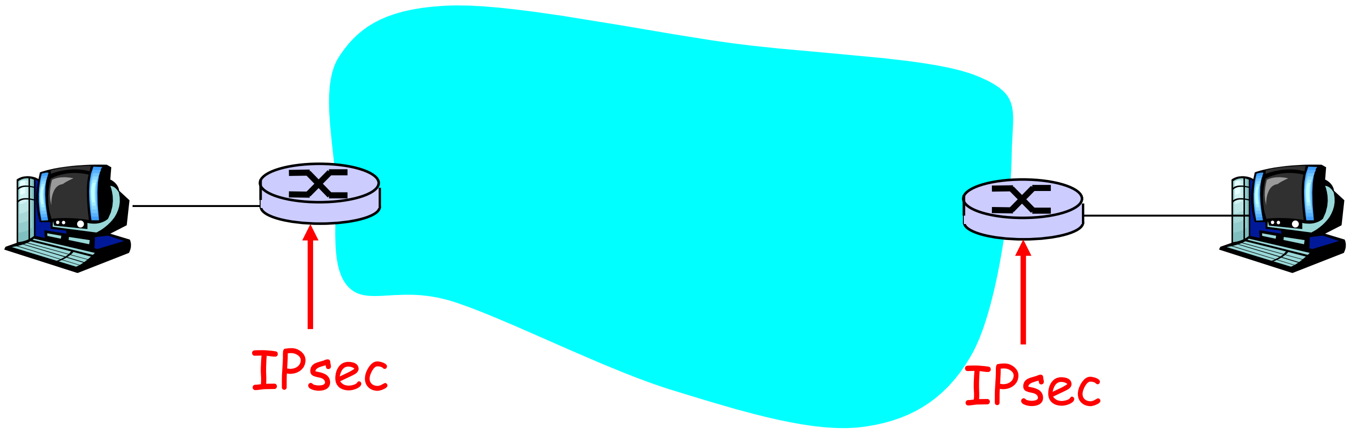
- Two protocols providing different service models:
  - Authentication Header (AH)
  - Encapsulation Security Protocol (ESP)

IPsec Transport Mode



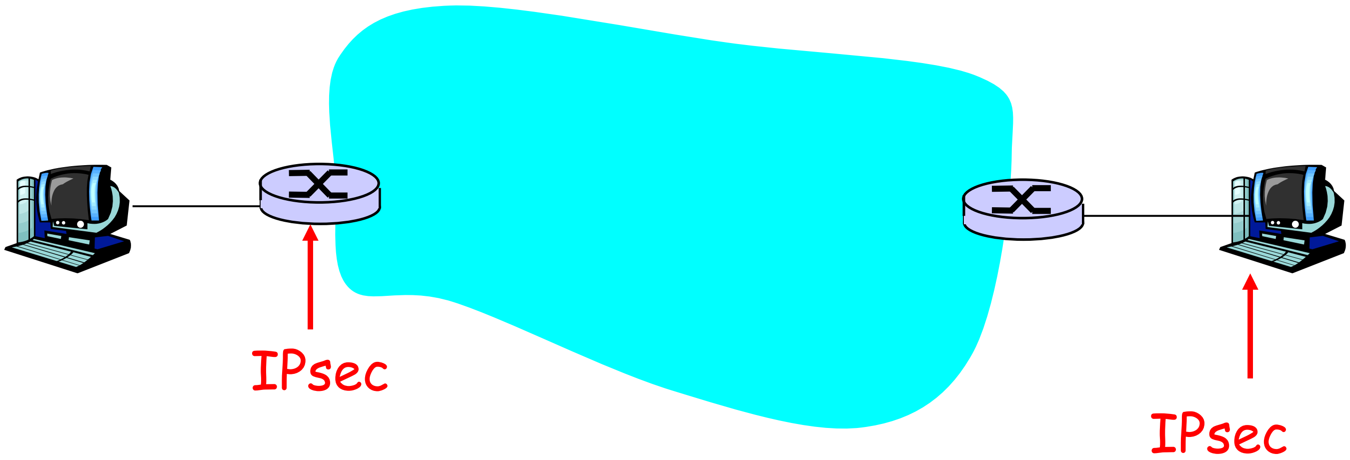
- IPsec datagram emitted and received by end-system.
- Protects upper level protocols

IPsec – tunneling mode (1)



- End routers are IPsec aware. Hosts need not be.

IPsec – tunneling mode (2)



- Also tunneling mode.

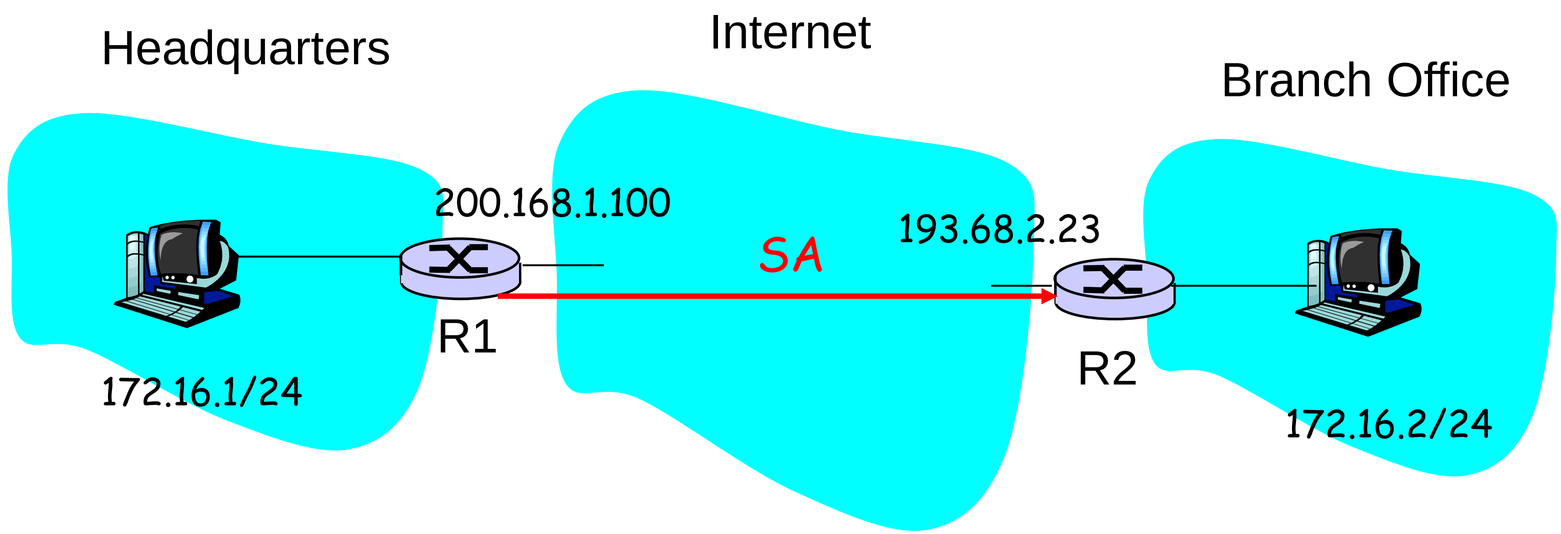
Two protocols

- 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
  - In the following we will focus on ESP

Security associations (SAs)

- Before sending data, a virtual connection is established from sending entity to receiving entity
- Called "security association (SA)"
  - SAs are simplex: for only one direction
- Both sending and receiving entities maintain *state information* about the SA
  - Recall that TCP endpoints also maintain state information.
  - IP is connectionless; IPsec is connection-oriented!
- How many SAs in VPN with headquarter, branch office, and n traveling salesperson?
  - 2+2n

Example: SA from R1 to R2



For each SA, R1 stores:

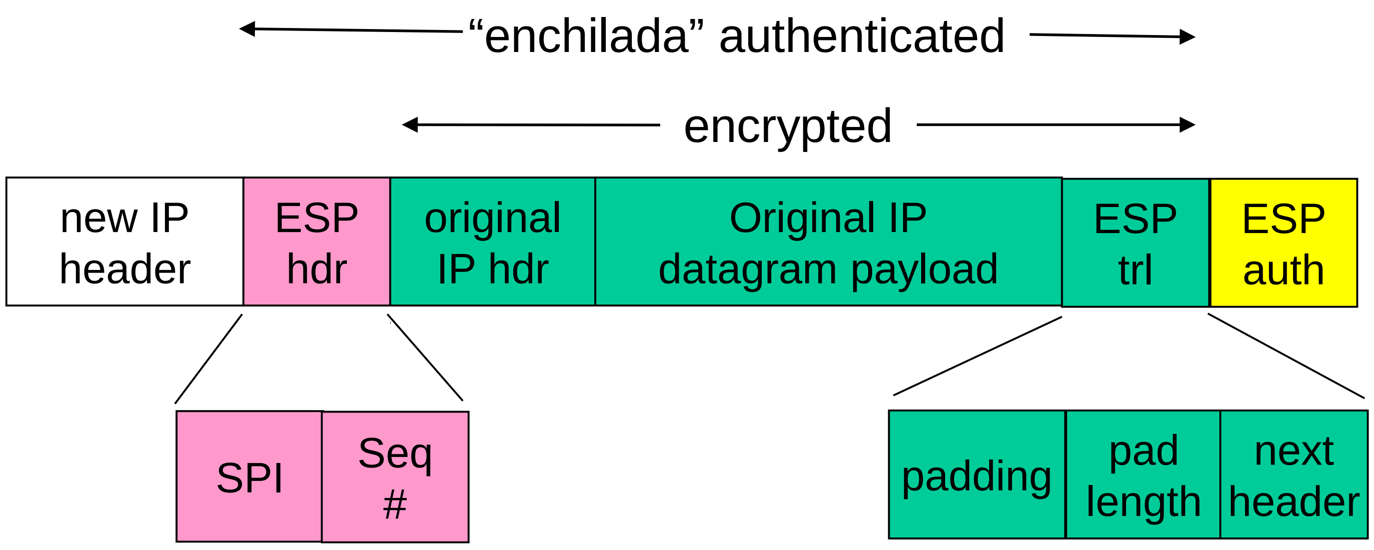
- 32-bit identifier for SA: *Security Parameter Index (SPI)*
- the origin interface of the SA (200.168.1.100)
- destination interface of the SA (193.68.2.23)
- type of encryption to be used (for example, 3DES with CBC)
- encryption key
- type of integrity check (for example, HMAC with MD5)
- authentication key

Security Association Database (SAD)

- Endpoint holds state of its SAs in a SAD, where it can locate them during processing.
- With n salespersons, 2 + 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

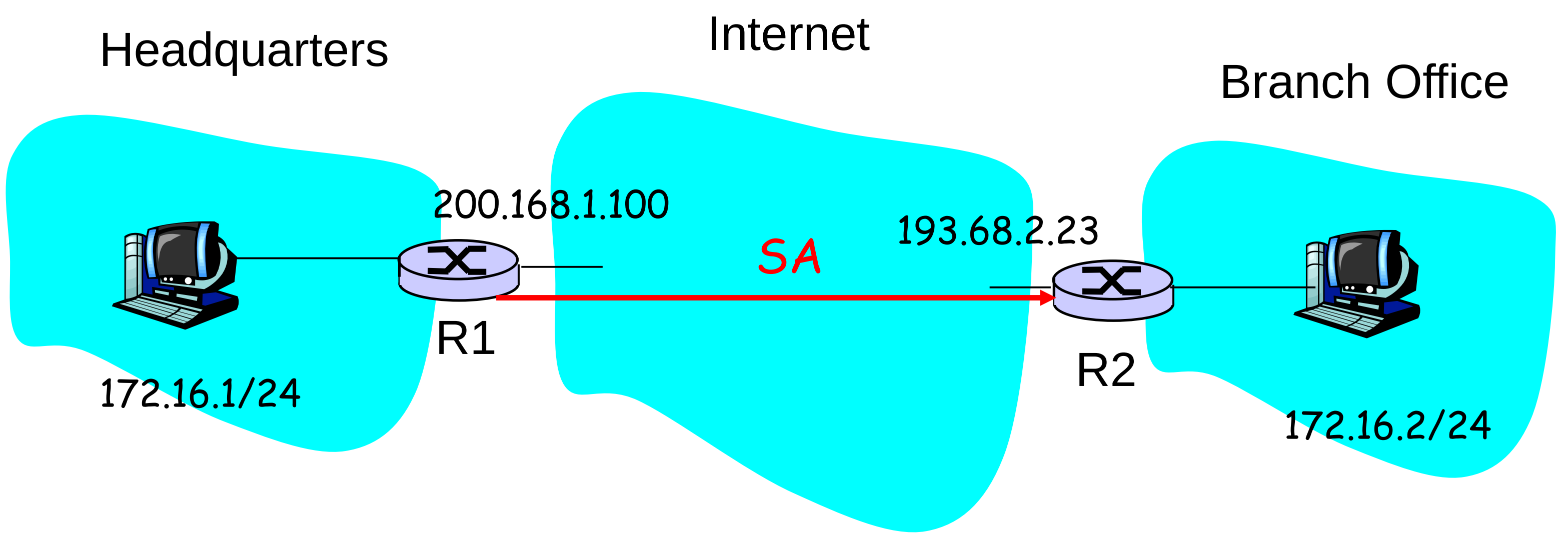
Focus for now on tunnel mode with ESP

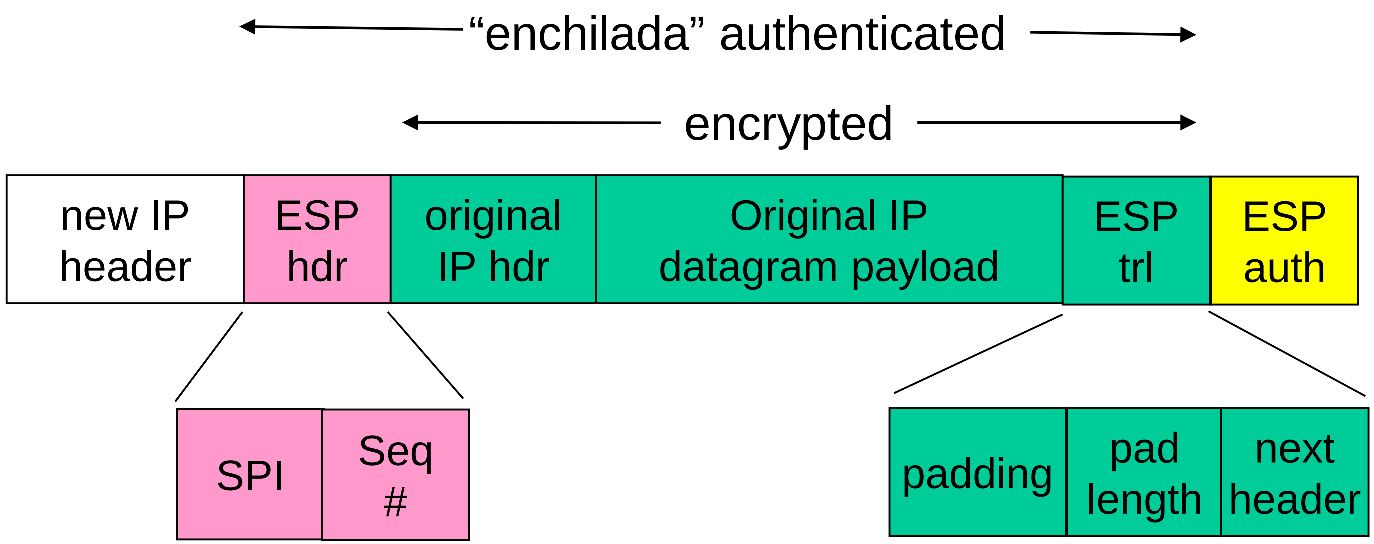


R1 converts original datagram into IPsec datagram

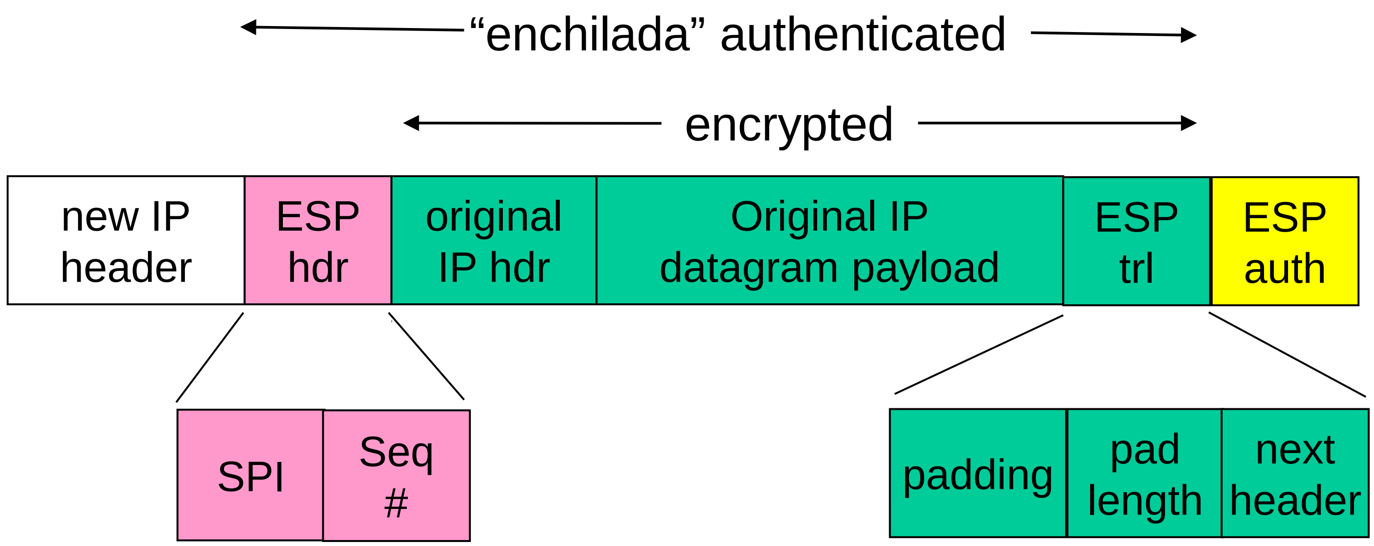
- Appends to back of original datagram (which includes original header fields!) an "ESP trailer" field.
- Encrypts result using algorithm & key specified by SA.
- Appends to front of this encrypted quantity the "ESP header, creating "enchilada".
- Creates authentication MAC over the *whole enchilada*, 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.

What happens?





Inside the enchilada:



- 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 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
  - But 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.
- 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 the SAD indicates "how" to do that.

IPsec: Some questions?

- Suppose Trudy sits somewhere between R1 and R2. She doesn't know the keys.
  - Will Trudy be able to see contents of original 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?

Internet Key Exchange (IKE)

- In previous examples, we manually established 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…

- Such manually keying is impractical for large VPN with, say, hundreds of sales people.
- Instead use *IPsec IKE (Internet Key Exchange)*

IKE Phases

- Similar to SSL
  - Only two phases
- Authentication Phase (proof who you are)
  - Pre-shared secret (PSK)
    - both sides start with a secret
  - with PKI (public keys and certificates).
- SA creations
  - Endpoints create SAs for both directions
  - message exchange for algorithms, secret keys, SPI numbers

Summary of IPsec

- IPsec peers can be two end systems, two routers/firewalls, or a router/firewall and an end system
- Either the AH or the ESP protocol (or both)
  - The AH protocol provides integrity and source authentication
  - The ESP protocol additionally provides encryption
- IPsec creates Security Associations (SAs)
- IKE used for establishing SAs
  - message exchange for algorithms, secret keys, SPI numbers

Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

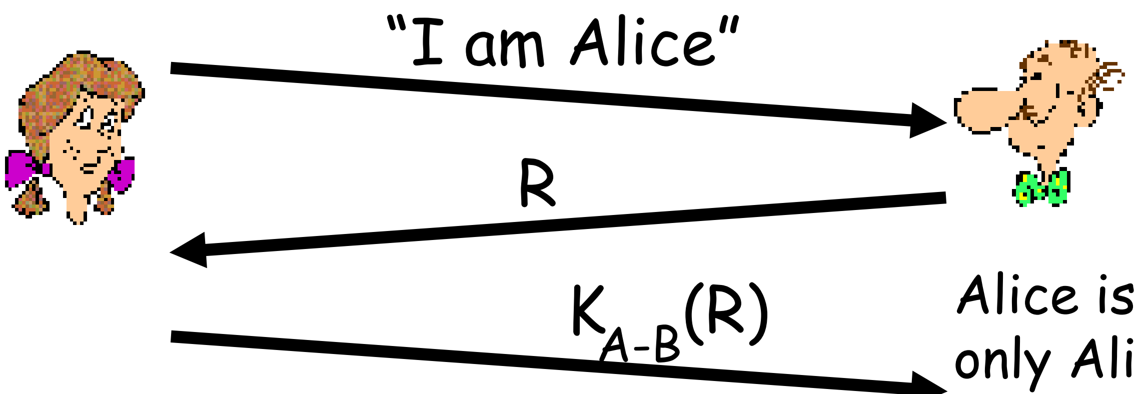
WEP Design Goals

- Symmetric key crypto
  - Confidentiality
  - Station authorization
  - Data integrity
- Self synchronizing: each packet separately encrypted
  - Given encrypted packet and key, can decrypt; can continue to decrypt packets when preceding packet was lost
  - Unlike Cipher Block Chaining (CBC) in block ciphers
- Efficient
  - Can be implemented in hardware or software

End-point authentication w/ nonce

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

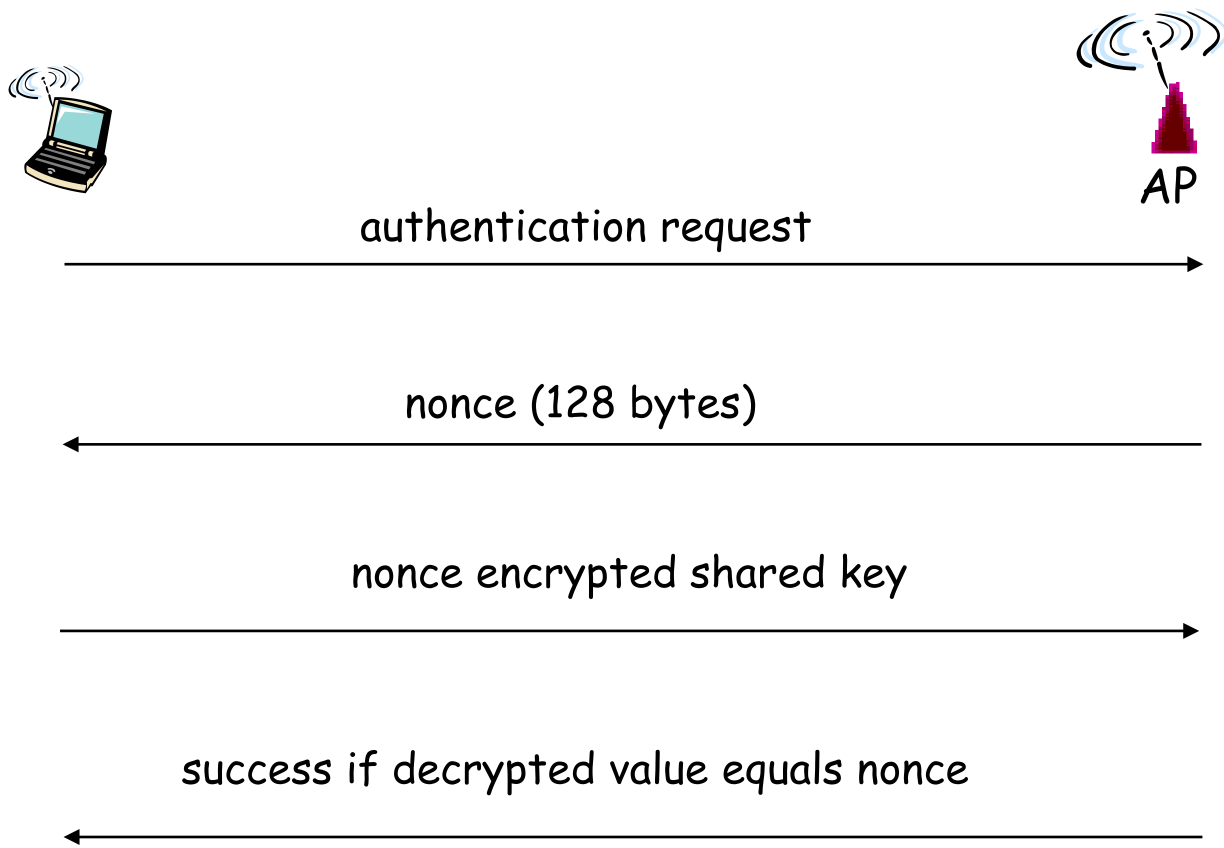
How: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



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

WEP Authentication

Not all APs do it, even if WEP is being used. AP indicates if authentication is necessary in beacon frame. Done before association.



Breaking 802.11 WEP encryption

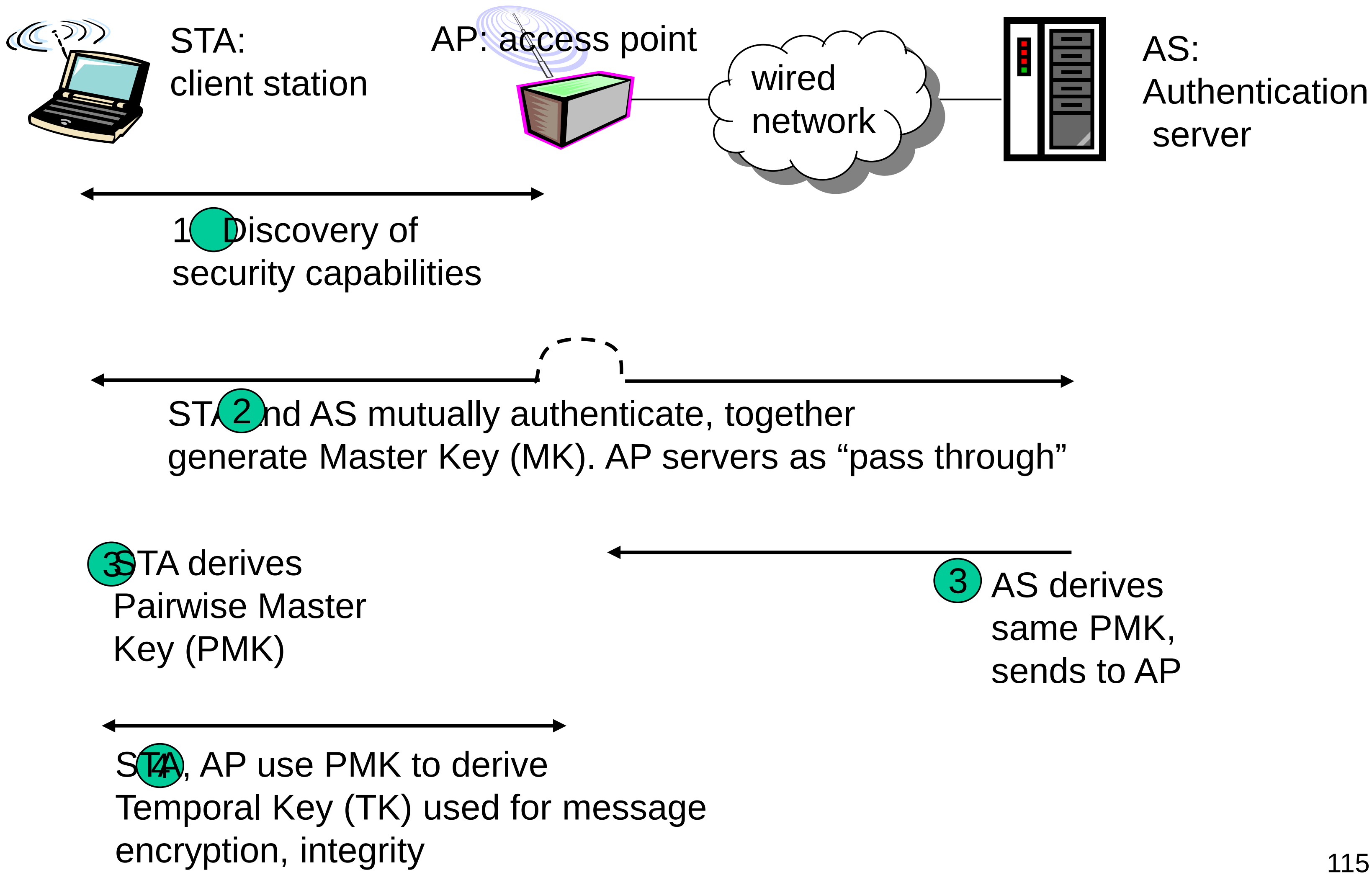
security hole:

- 24-bit IV, one IV per frame, -> IV's eventually reused
- IV transmitted in plaintext -> IV reuse detected
- attack:
  - Trudy causes Alice to encrypt known plaintext d_1 d_2 d_3 d_4 …
  - Trudy sees: $c_i = d_i$ XOR k_i^{IV}
  - Trudy knows c_i d_i, so can compute k_i^{IV}
  - Trudy knows encrypting key sequence k_1^{IV} k_2^{IV} k_3^{IV} …
  - Next time IV is used, Trudy can decrypt!

802.11i: improved security

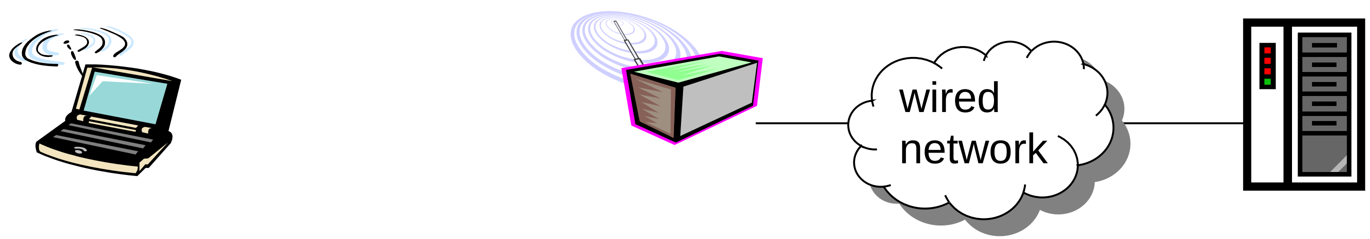
- numerous (stronger) forms of encryption possible
- provides key distribution
- uses authentication server separate from access point

802.11i: four phases of operation



EAP: extensible authentication protocol

- EAP: end-end client (mobile) to authentication server protocol
- EAP sent over separate "links"
  - mobile-to-AP (EAP over LAN)
  - AP to authentication server (RADIUS over UDP)



EAP TLS	
EAP	
EAP over LAN (EAPoL)	RADIUS
IEEE 802.11	UDP/IP

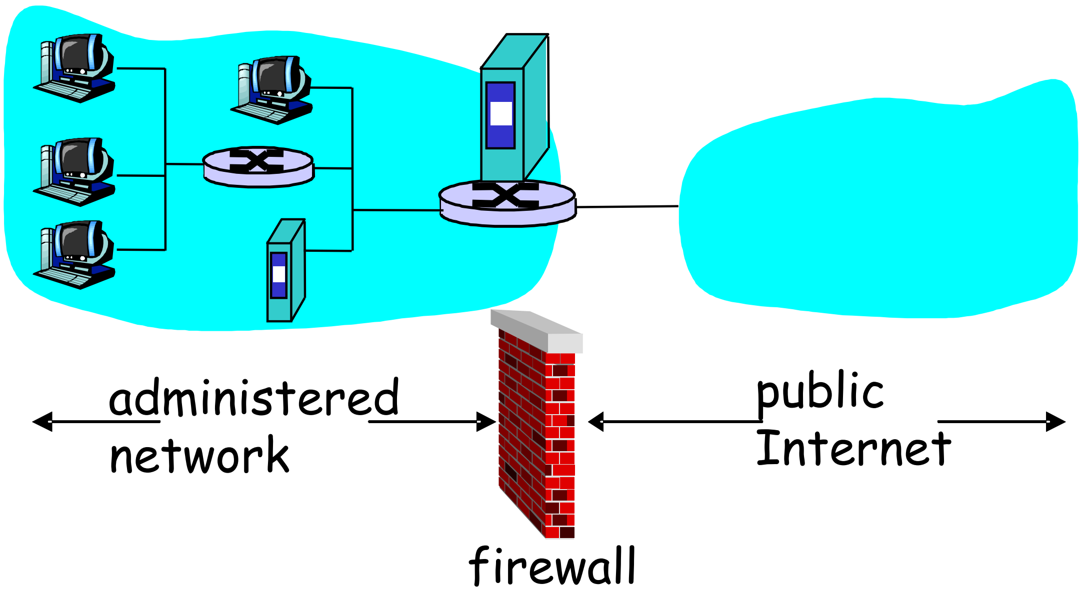
Roadmap

- Introduction
- Principles of cryptography
  - Confidentiality
  - Message integrity
  - End-point authentication
- Securing e-mail
- Securing TCP connections: SSL
- Network layer security: IPsec
- Securing wireless LANs
- Operational security: firewalls and IDS

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.

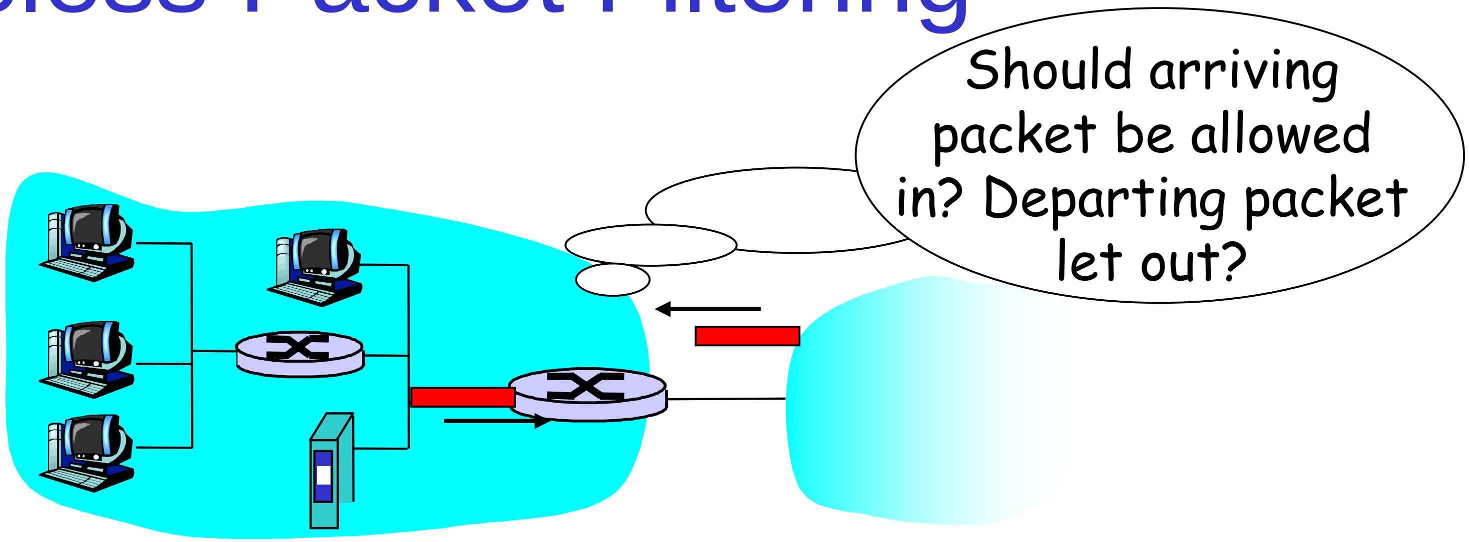
- e.g., attacker replaces CIA's homepage with something else

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

three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

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: Examples

- Example 1: Block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23.
  - all incoming, outgoing UDP flows and telnet connections are blocked.
- Example 2: Block incoming TCP segments with ACK=0.
  - 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 (eg 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 on *each* interface: (action, condition) pairs

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., 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): can determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets

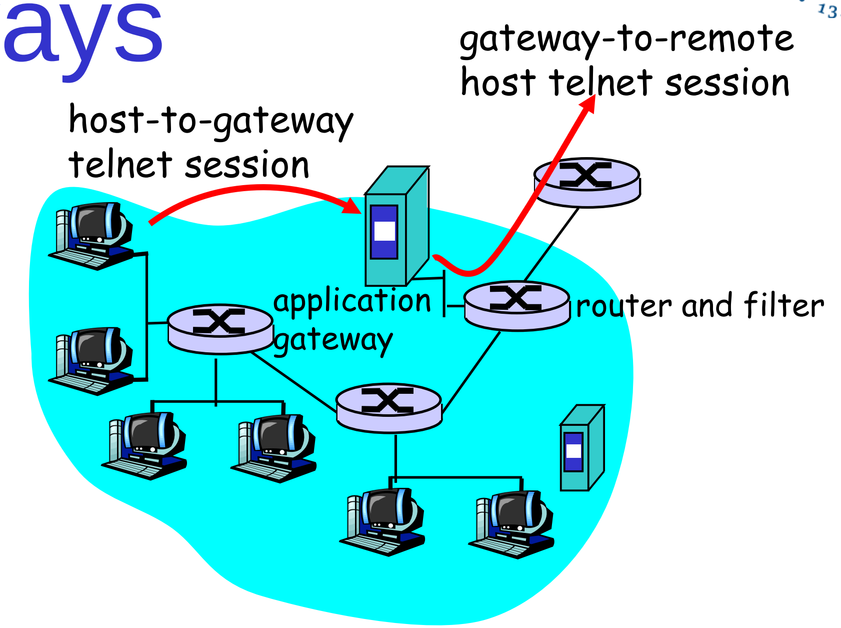
Stateful Packet Filtering

- 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 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	×
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	

Application Gateways

- filters packets on application data as well as on IP/TCP/UDP fields.
- <u>example:</u> 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 and Gateways

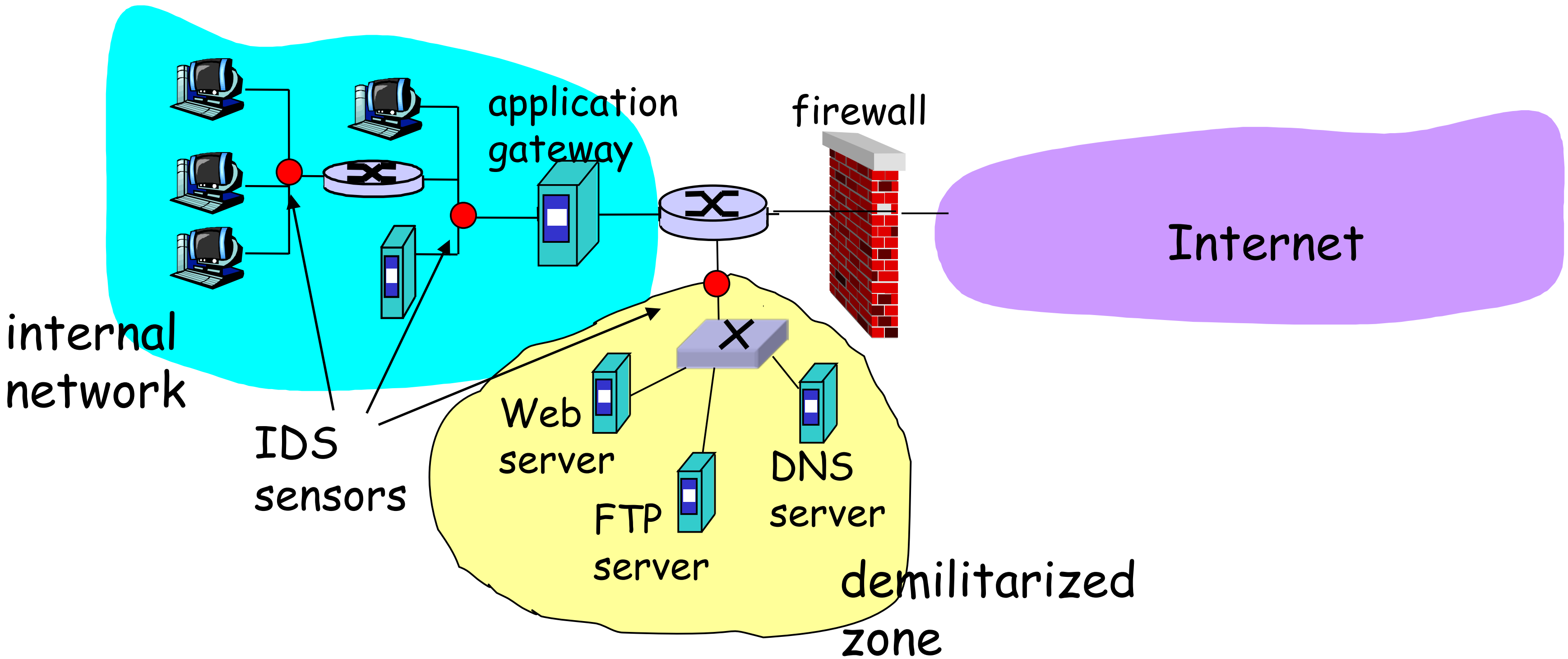
- if multiple app's. need special treatment, each has own app. gateway.
- client software must know how to contact gateway.
  - e.g., must set IP address of proxy in Web browser
- IP spoofing: router can't know if data "really" comes from claimed source
- filters often use all or nothing policy for UDP.
- 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)
- message integrity
- end-point authentication

…. used in many different security scenarios

- secure email
- secure transport (SSL)
- IP sec
- 802.11

Operational Security: firewalls and IDS