Romain Fontugne

2023 Fall Semester

Information Network Systems

Security, cryptography

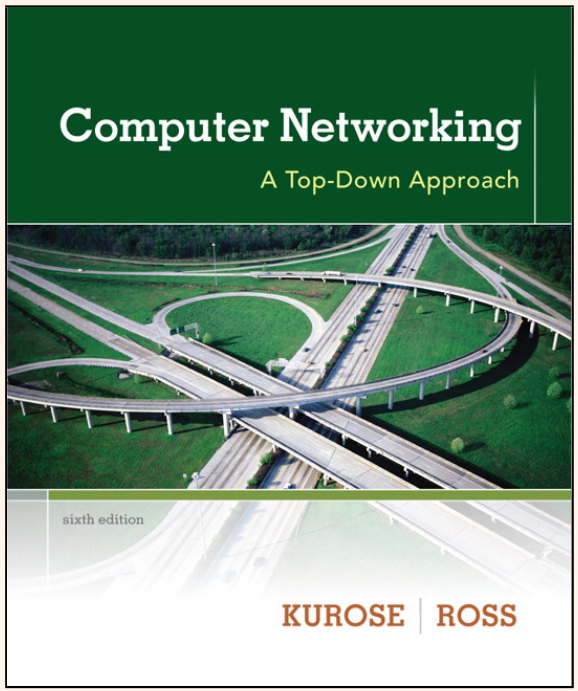
Last lecture

We've seen the fundamentals of cryptography

- Symmetric Key Cryptography
- Public Key Cryptography
- Message Integrity
- Digital signature

Today's Lecture: Security Cryptography

1. Securing e-mail
2. Securing TCP connections: SSL
3. Operational Security: Firewall and IDS



- Chapter 8

Secure e-mail

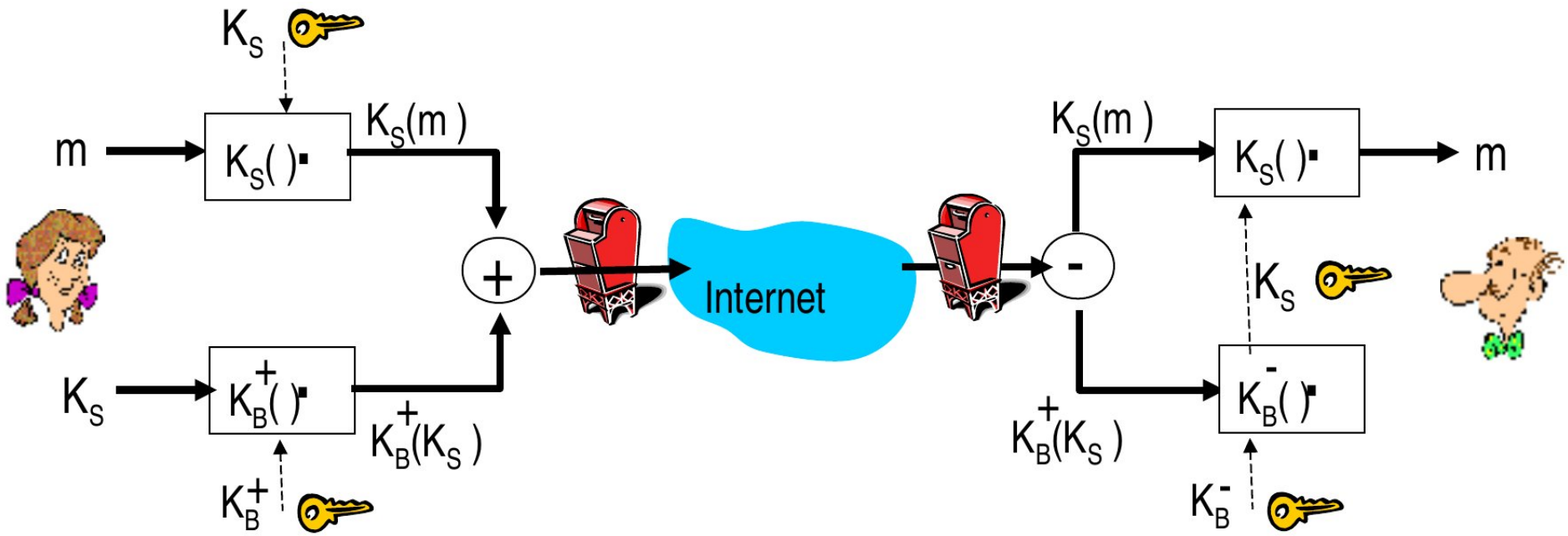
PGP: Pretty Good Privacy

- Most used e-mail encryption software
- Written in 1991
- Based on different encryption algorithms:
  - message digest: MD5 or SHA
  - symmetric key: 3DES, CAST or IDEA
  - public key: RSA
- Free implementation: GnuPG

Secure e-mail

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

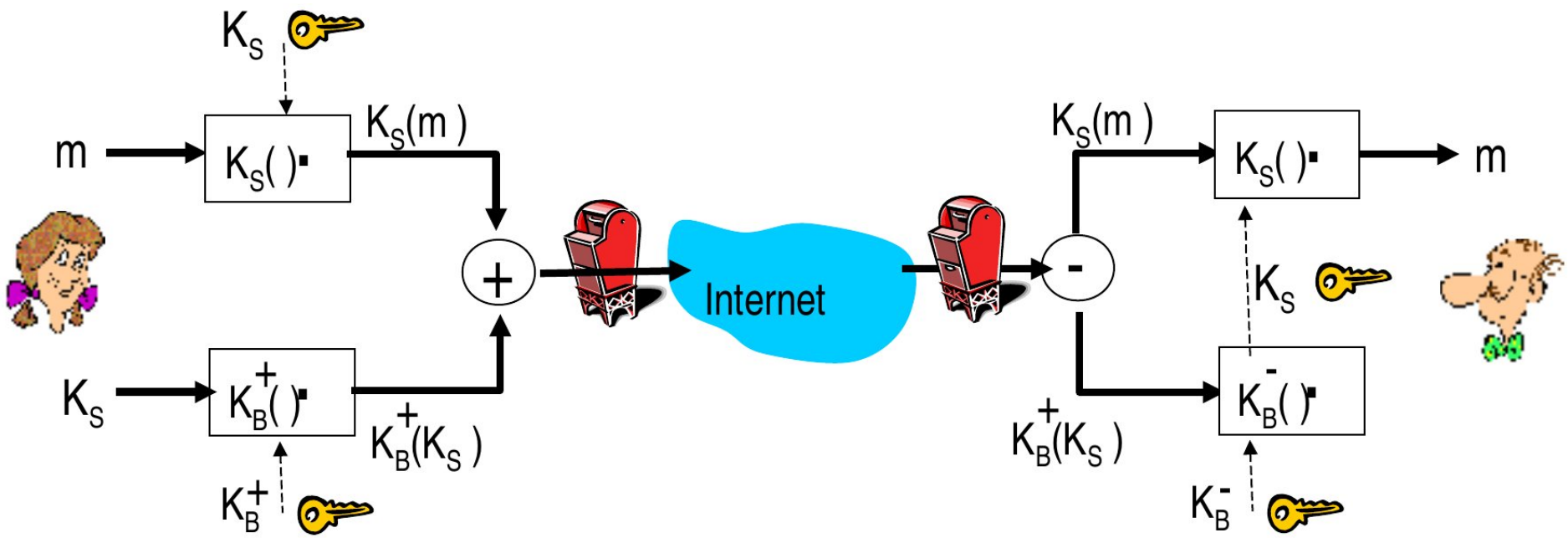
- **Alice**:
  - generates random symmetric private key, K_S
  - encrypts message with K_S (for efficiency)
  - also encrypts K_S with Bob's public key
  - sends both $K_S(m)$ and $K_B(K_S)$ to Bob



Secure e-mail

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

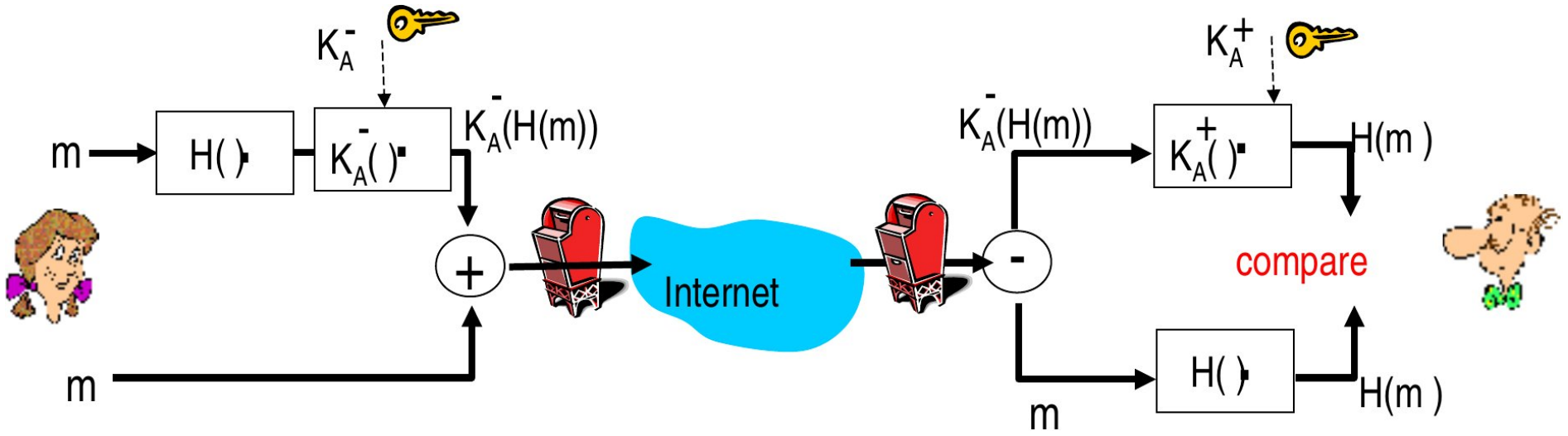
- **Bob**:
  - uses his private key to decrypt and recover K_S
  - uses K_S to decrypt $K_S(m)$ to recover m



Secure e-mail (continued)

Alice wants to provide sender authentication message integrity

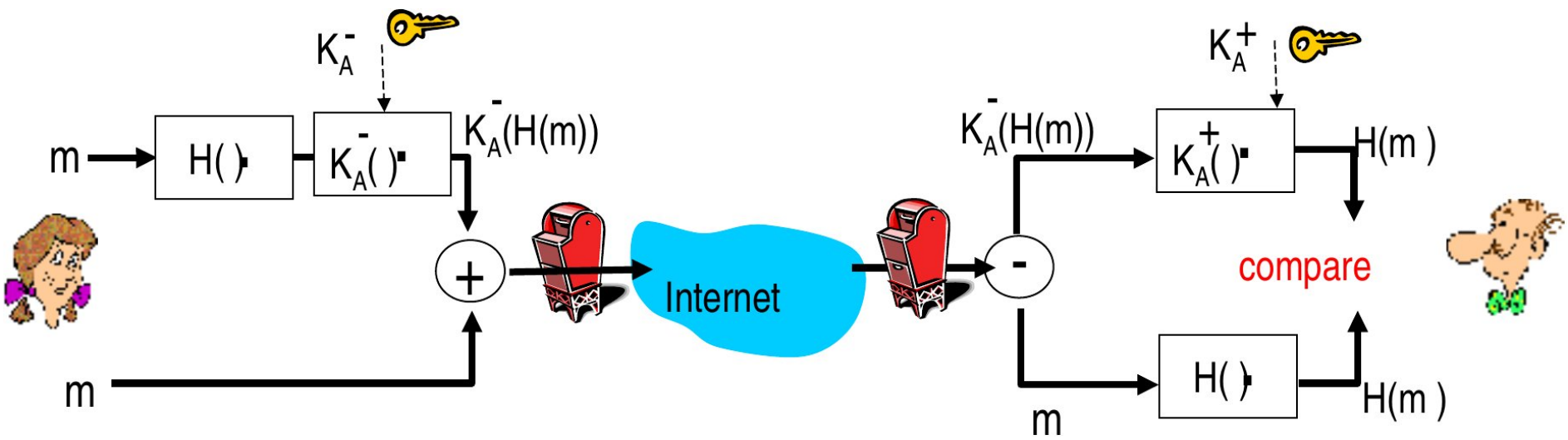
- **Alice**:
  - digitally signs message (using hash and her private key)
  - sends both message (in the clear) and digital signature



Secure e-mail (continued)

Alice wants to provide sender authentication message integrity

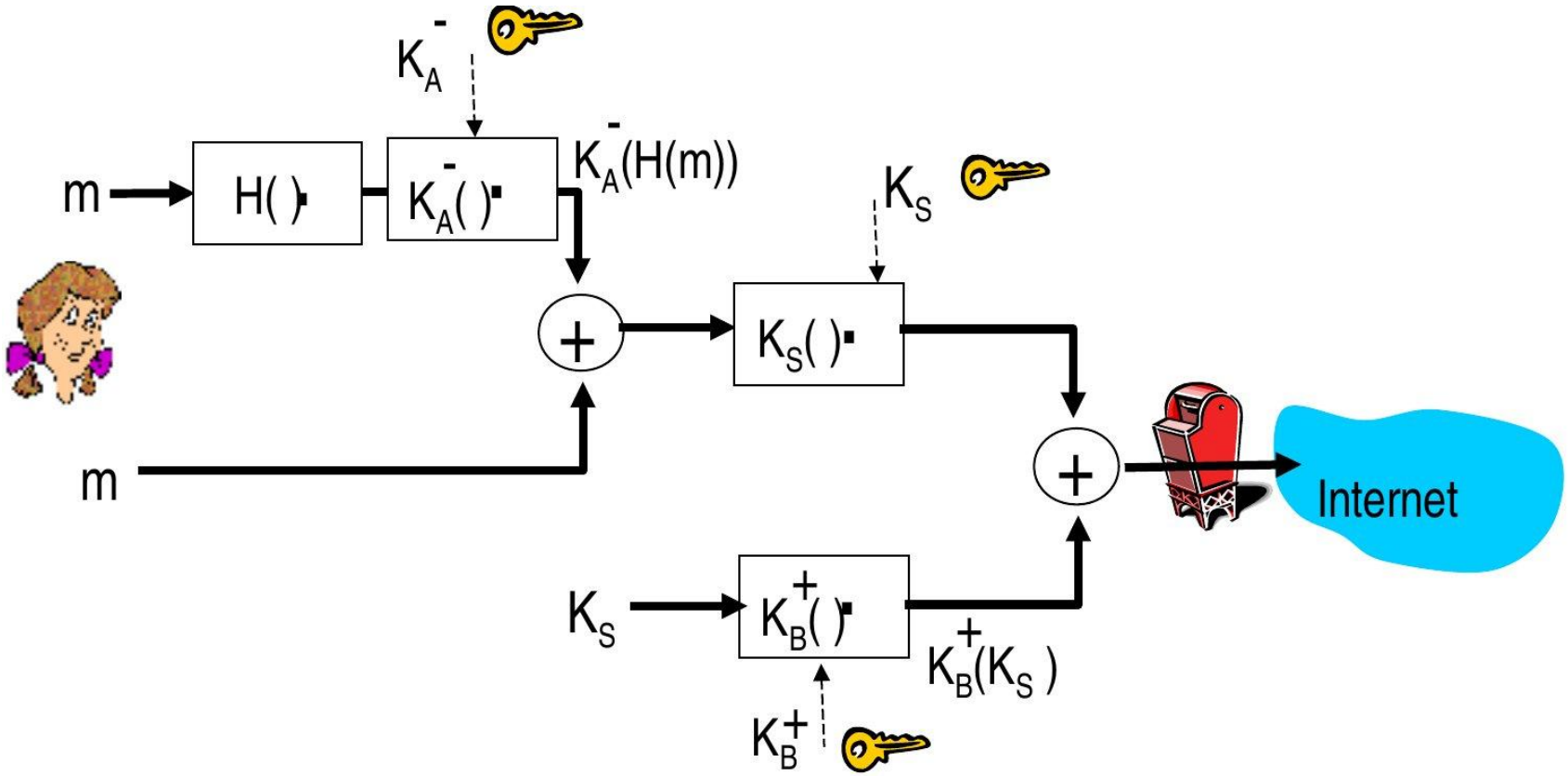
- **Bob**:
  - computes the message digest: H(m')
  - decrypts the digital signature with Alice public key: H(m)
  - compares H(m') and H(m)



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



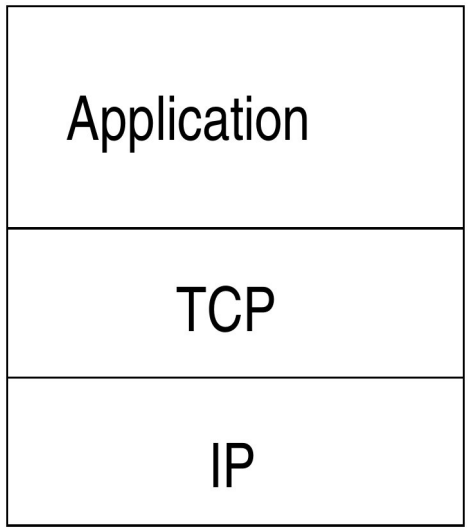
SSL: Secure Sockets Layer

SSL: Widely deployed security protocol

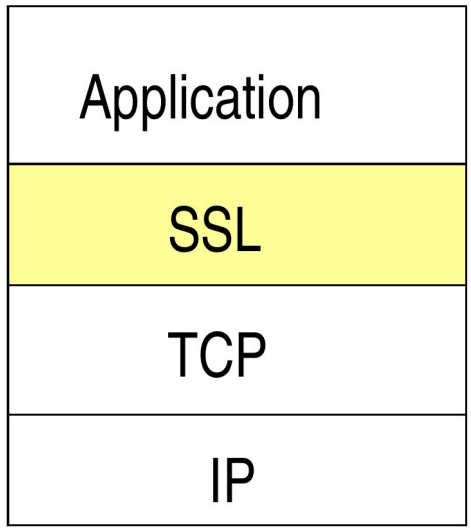
- **supported by almost all browsers, web servers**
  - https
  - billions $/year over SSL
- variation -TLS: transport layer security
- provides
  - confidentiality
  - integrity
  - authentication
- **original goals:**
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant
- **available to all TCP applications (secure socket interface)**

SSL and TCP/IP

- SSL provides application programming interface (API) to applications
- C, Java, Python... SSL libraries/classes easily available

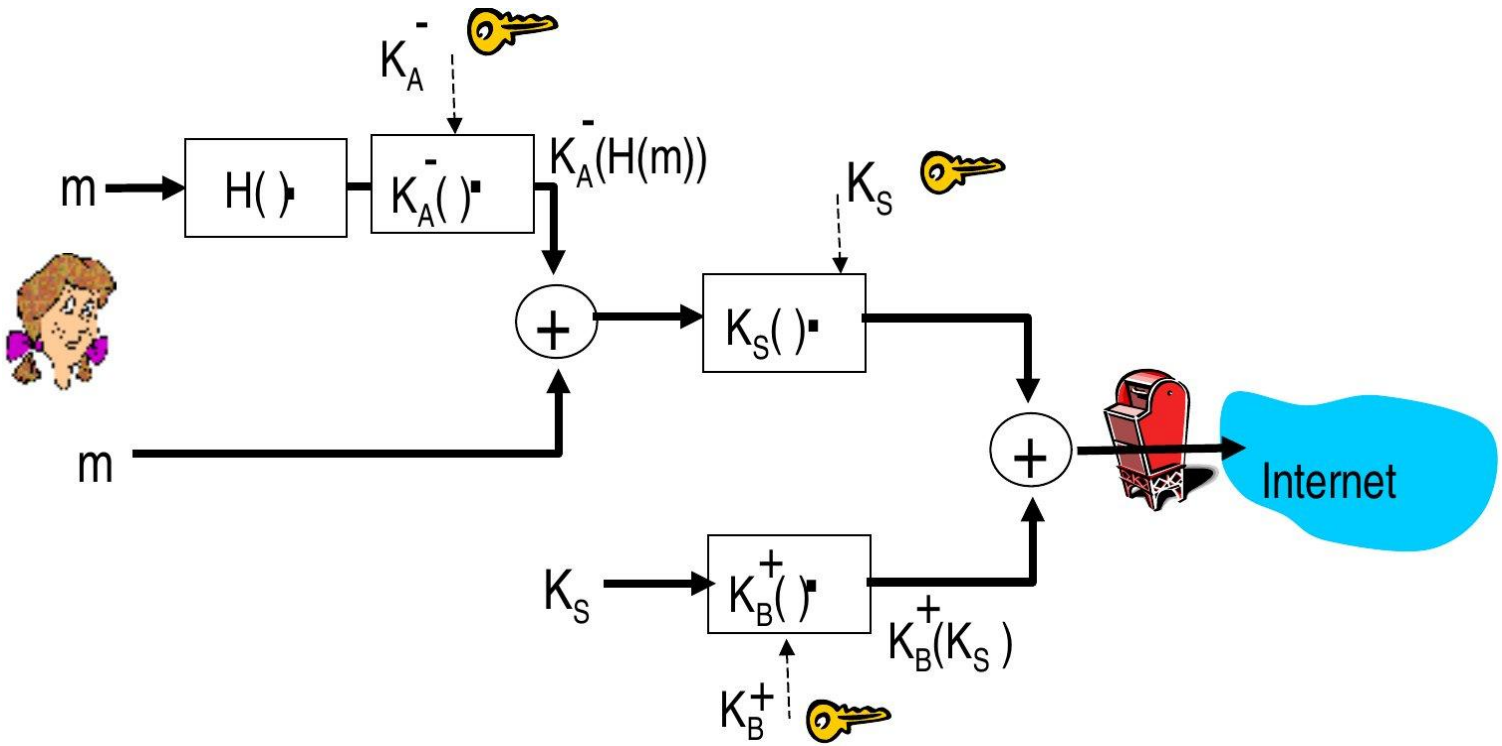


normal application



application with SSL

Could do something like PGP?



but...

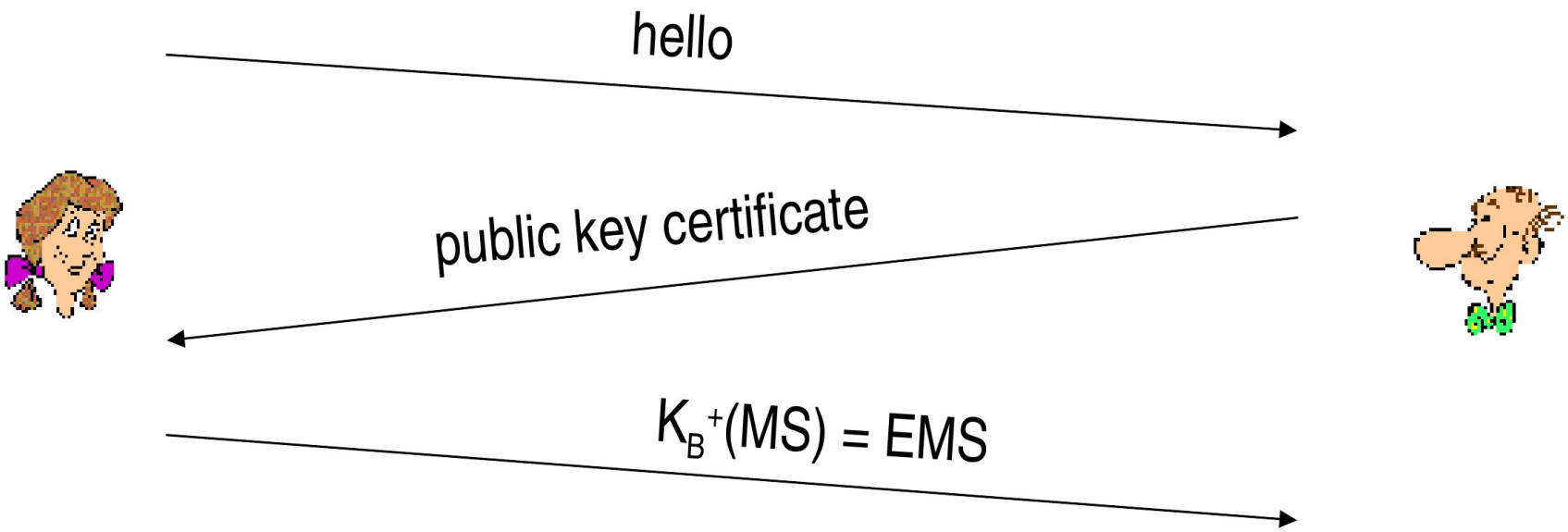
- want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Simplified SSL: a simple secure channel

Simplified SSL: Features

- handshake: Alice and Bob use their certificates, private keys to authenticate each other and exchange **shared secret**
- key derivation: Alice and Bob use shared secret to derive **set of keys**
- data transfer: data to be transferred is broken up into series of records
- connection closure: special messages to securely close connection

Simplified SSL: a simple 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:

- Kc = encryption key for data sent from client to server
- Mc = MAC key for data sent from client to server
- Ks = encryption key for data sent from server to client
- Ms = 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

Why not encrypt data in constant stream as we write it to TCP?

- where would we put the MAC? If at end, no message integrity until all data processed
- e.g., 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

Problem:

- attacker can capture and replay record or re-order records

Solution:

- $MAC = MAC(Mx, sequence||data)$
- note: no sequence number field

Problem:

- attacker could replay all records!

Solution:

- use nonce

Simplified SSL: control information

Problem:truncation attack

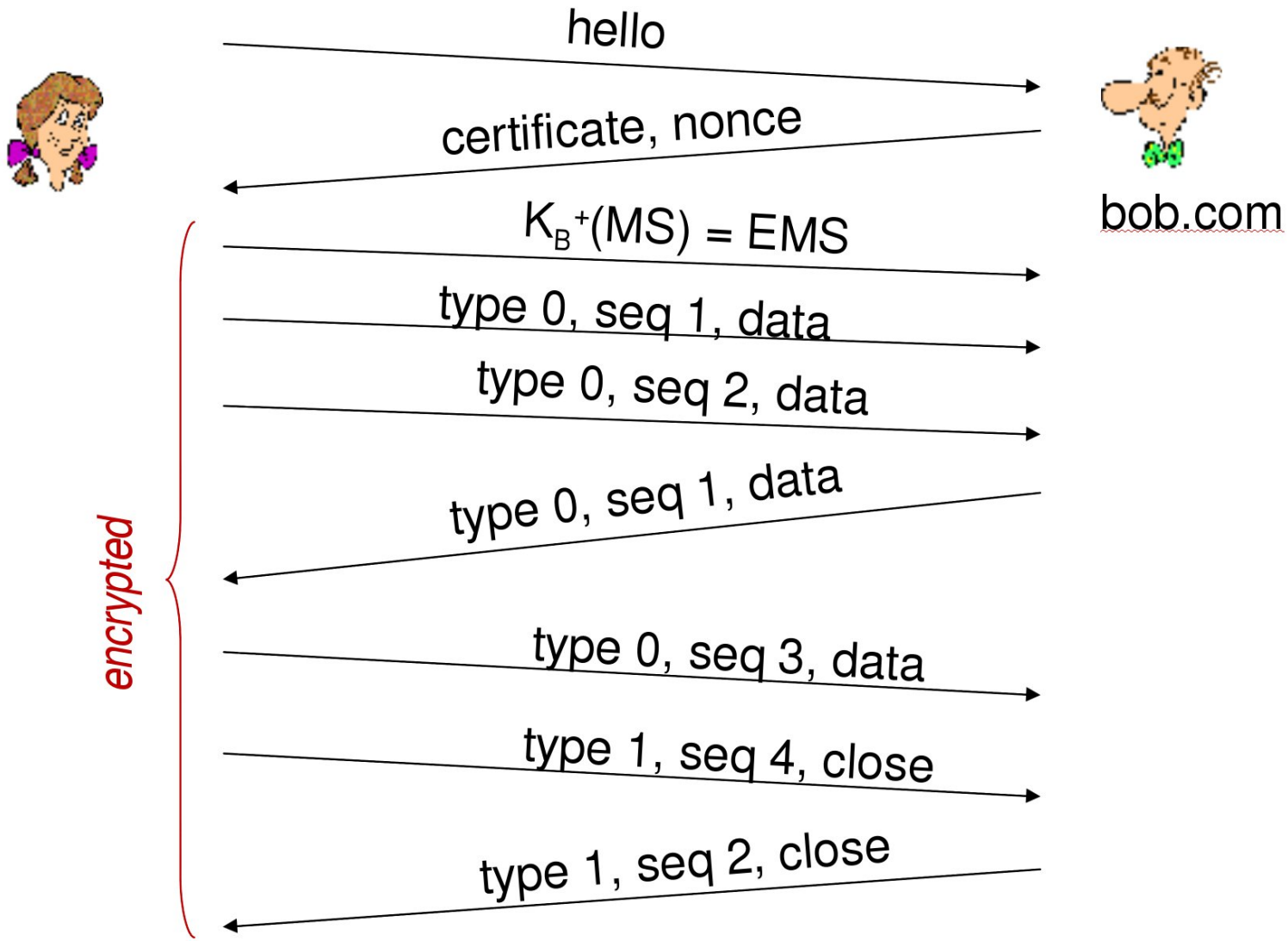
- attacker forges TCP-connection-close segment
- one or both sides thinks there is less data than there actually is

Solution: record types, with one type for closure

- type 0 for data; type 1 for closure

$$MAC = MAC(Mx, sequence||type||data)$$

Simplified SSL: summary



Simplified SSL isn't complete

A few missing things...

- how long are fields?
- which encryption protocols?
- want negotiation?
  - allow client and server to support different encryption algorithms
  - allow client and server to choose together specific algorithm before data transfer

SSL cipher suite

Cipher suite

- public-key algorithm
- symmetric encryption algorithm
- MAC algorithm

SSL supports several cipher suites

Negotiation: client, server agree on cipher suite

- client offers choice
- server picks one

Common SSL symmetric ciphers

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

SSL Public key encryption

- RSA
- Diffie–Hellman

Real SSL: handshake (1)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

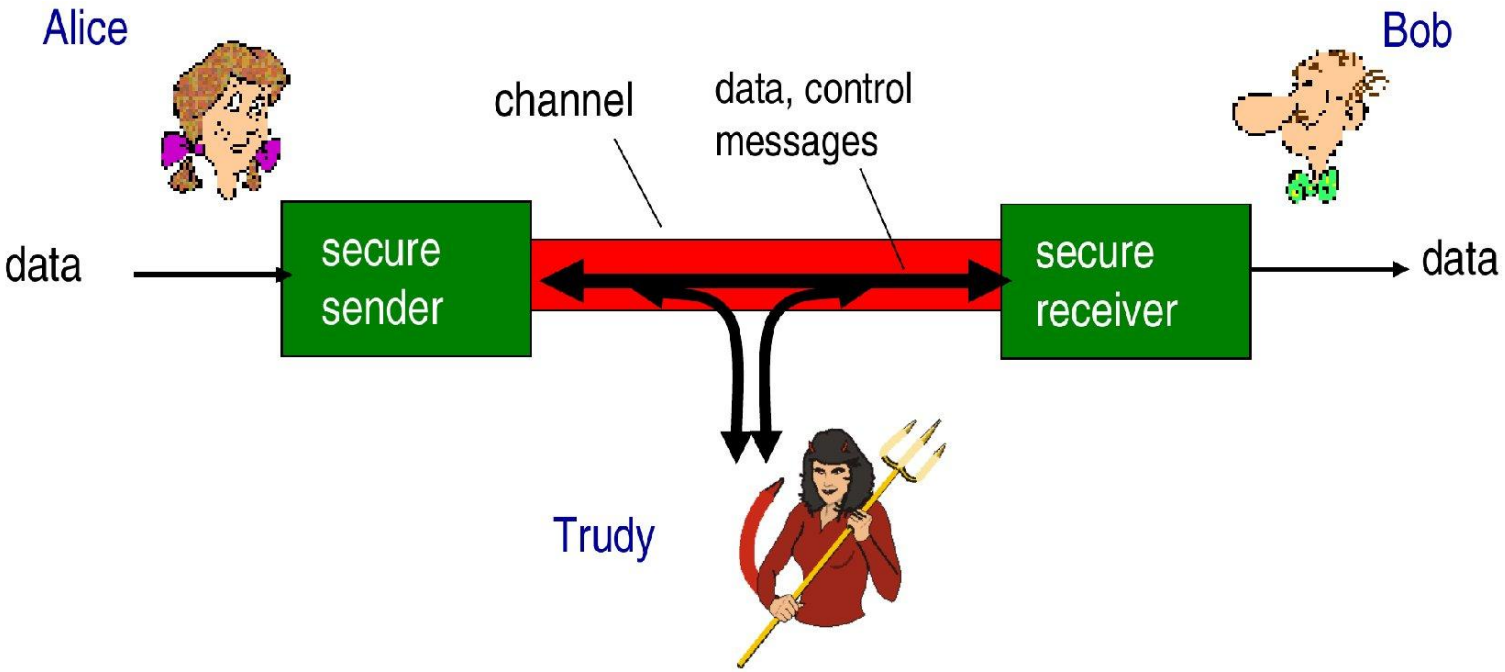
Real SSL: handshake (2)

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, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret 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 stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

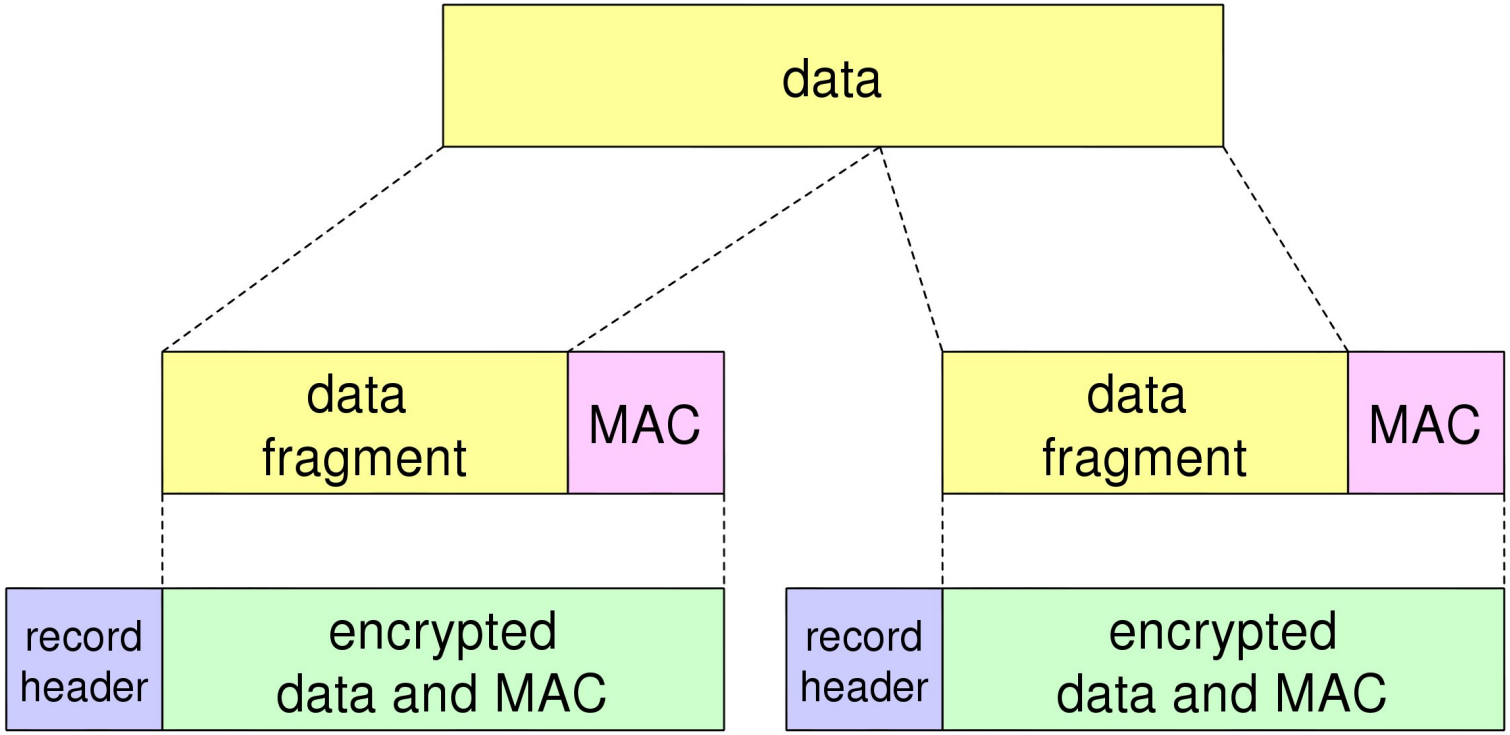


Real SSL: handshaking (4)

Why two random nonces?

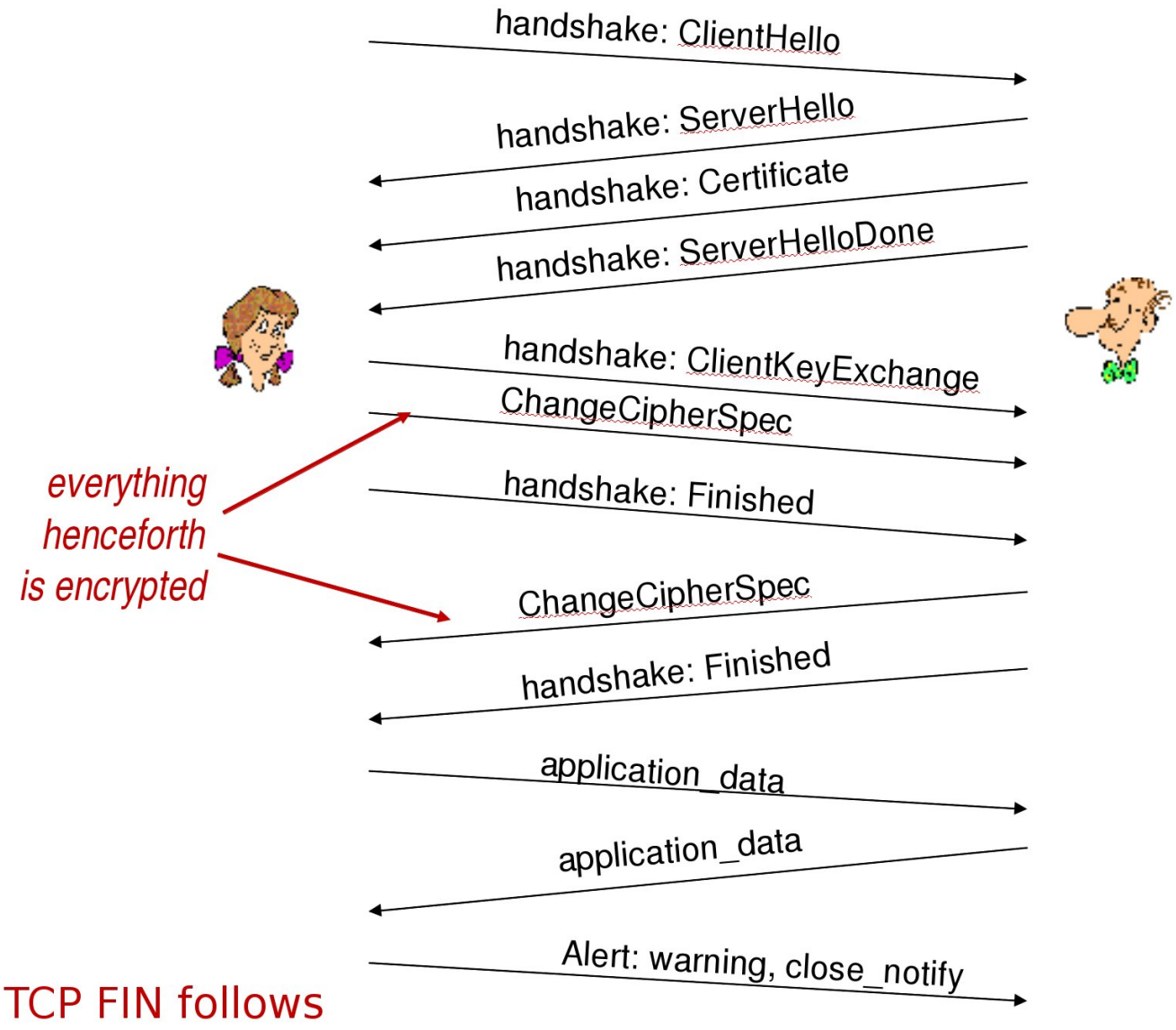
- Suppose Trudy sniffs all messages between Alice & Bob
- Next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - Solution: Bob sends different random nonce for each connection. This causes encryption 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: also includes sequence number and MAC key Mx
- Data & MAC: encrypted with symmetric algorithm
- Fragment: each SSL fragment 2^{14} bytes (16 Kbytes)

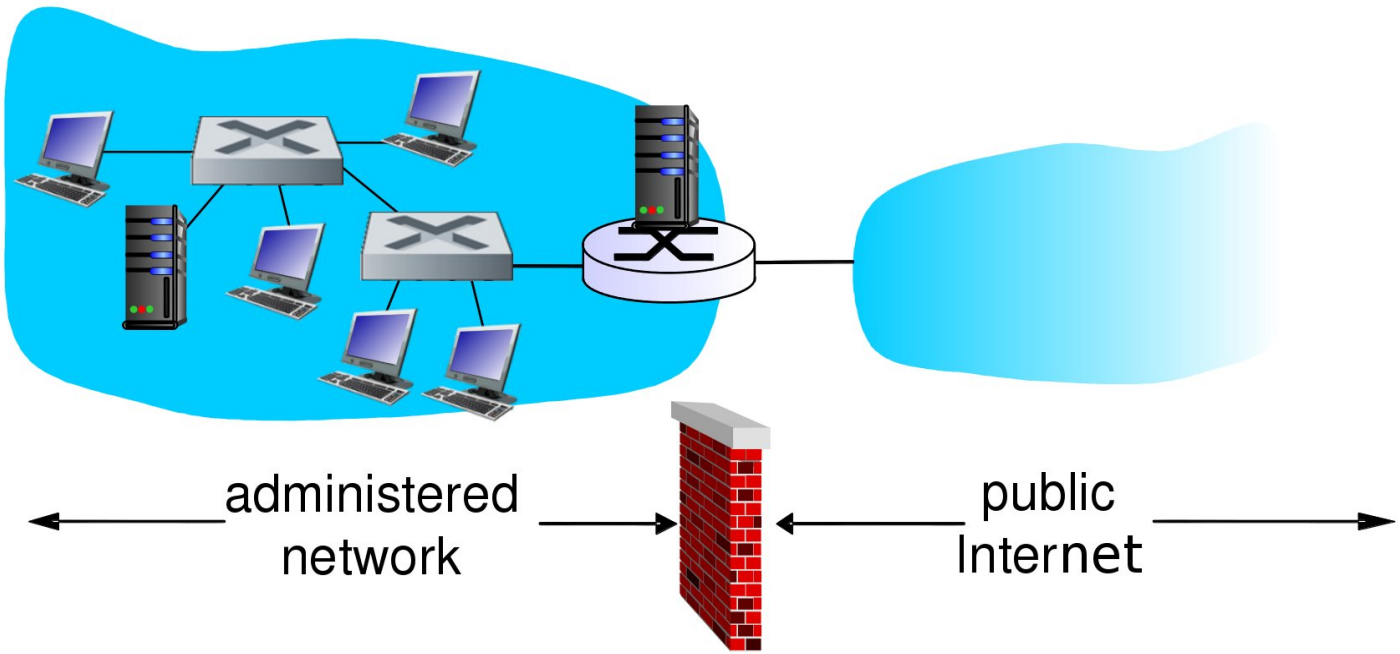
Real SSL connection



Firewalls

Firewall: definition

- Isolates organization's internal net from Internet, allowing some packets to pass, blocking others



trusted "good guys" *firewall* *untrusted "bad guys"*

Firewalls: why?

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

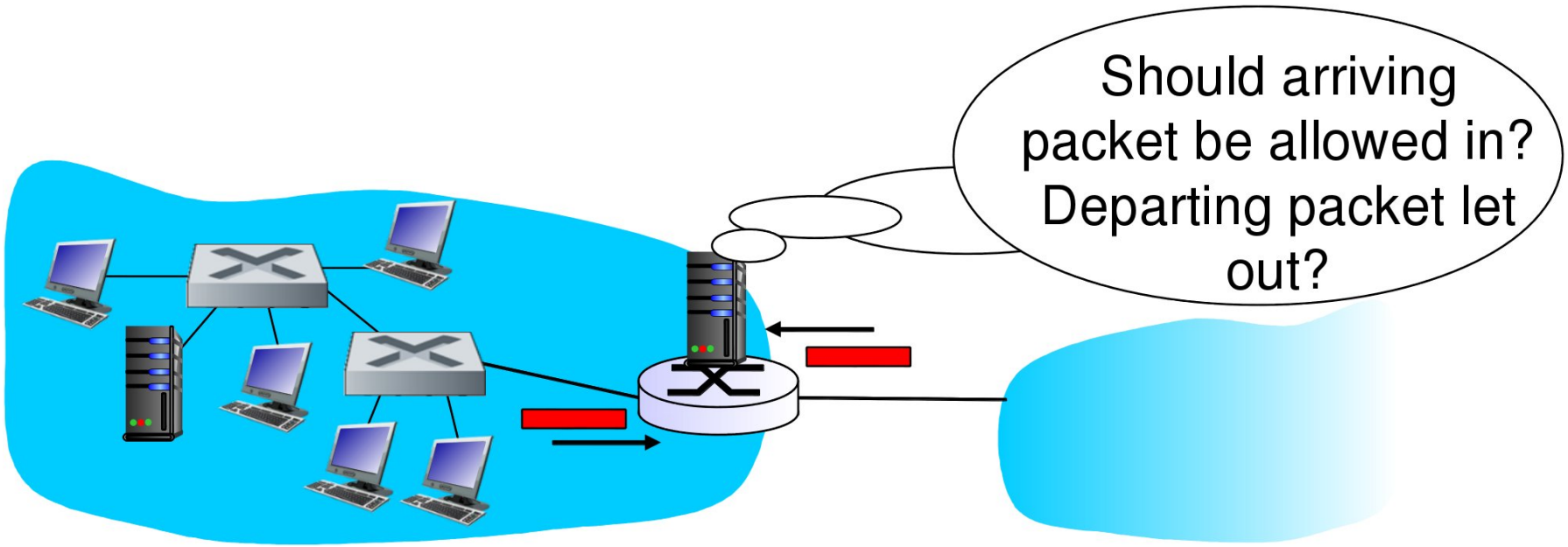
Prevent denial of service attacks:

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

Three types of firewalls:

- stateless packet filters
- stateful packet filters
- application gateways

Stateless packet filtering



- Internal network connected to Internet via router firewall
- Router firewall filters packet-by-packet, and decides 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 UDP packets 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

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 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	

Intrusion detection systems

Packet filtering:

- operates on TCP/IP headers only
- no correlation check among sessions

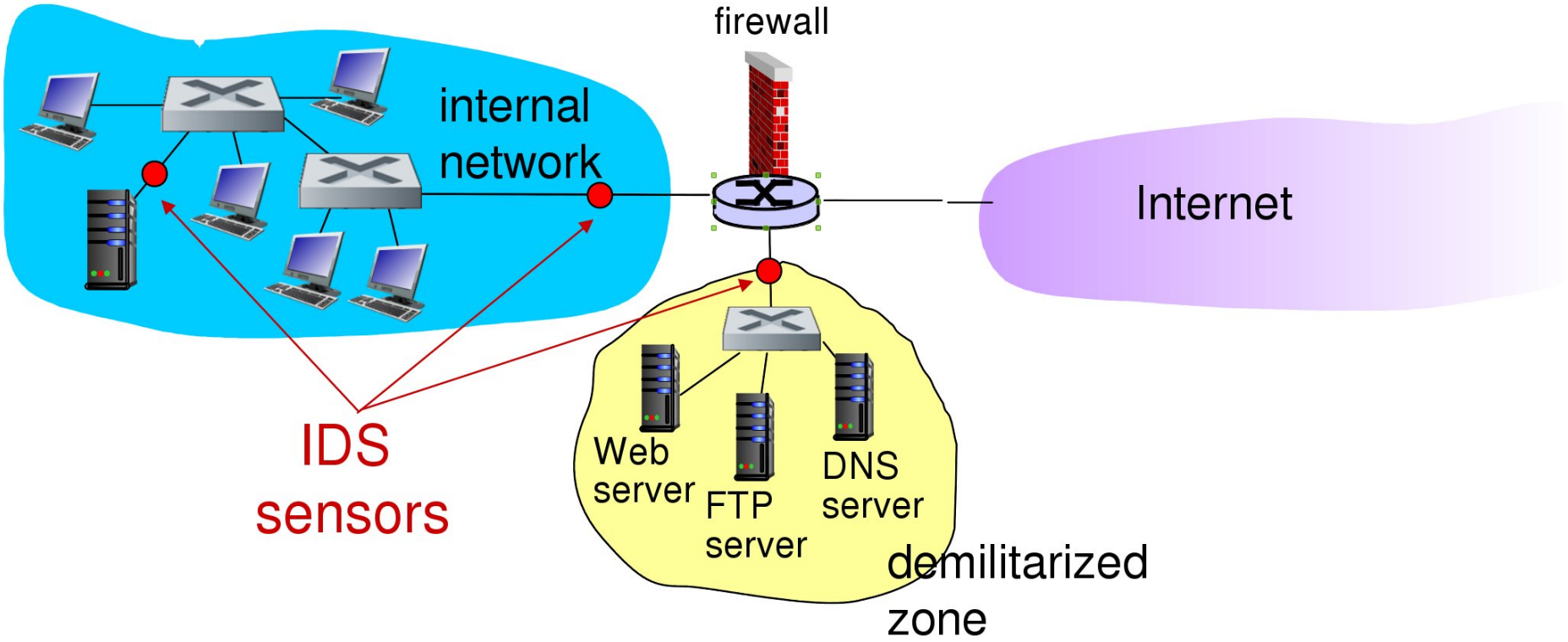
IDS: intrusion detection system

- 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
- can detect:
  - port scanning
  - network mapping
  - DoS attack

Intrusion detection systems

Multiple IDSs:

- different types of checking at different locations



Network Security: Summary

The last 2 lectures covered:

- basic techniques...
  - cryptography (symmetric and public)
  - message integrity
  - end-point authentication
- ... used in many different security scenarios
  - secure email
  - secure transport (SSL)
    - HTTPS
- operational security: firewalls and IDS

Next time

We'll build our own network and play with it:

- Setup a local area network
- Write simple network application
- Monitor the traffic
- Understand how it works
- ...

→Bring your laptop! (and any device you wanna connect)

Get ready:

- Install python (if it is not already done)
- Install wireshark