Module M7

CPSC 317

November 22, 2022

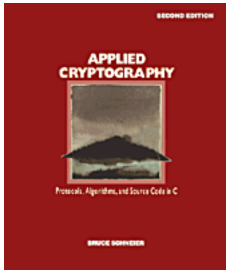
“Security is difficult!”

- Most of computer science is concerned with achieving desired behavior
- In some sense, security is concerned with preventing <u>un</u>desired behavior
  - Different way of thinking!
  - An enemy/opponent/hacker/adversary may be <u>actively</u> and <u>maliciously</u> trying to circumvent any protective measures you put in place

A Few Links

- www.elonka.com/UnsolvedCodes.html
- www.eff.org
- www.schneier.com/blog/about/

Secrets and Lies, Bruce Schneier



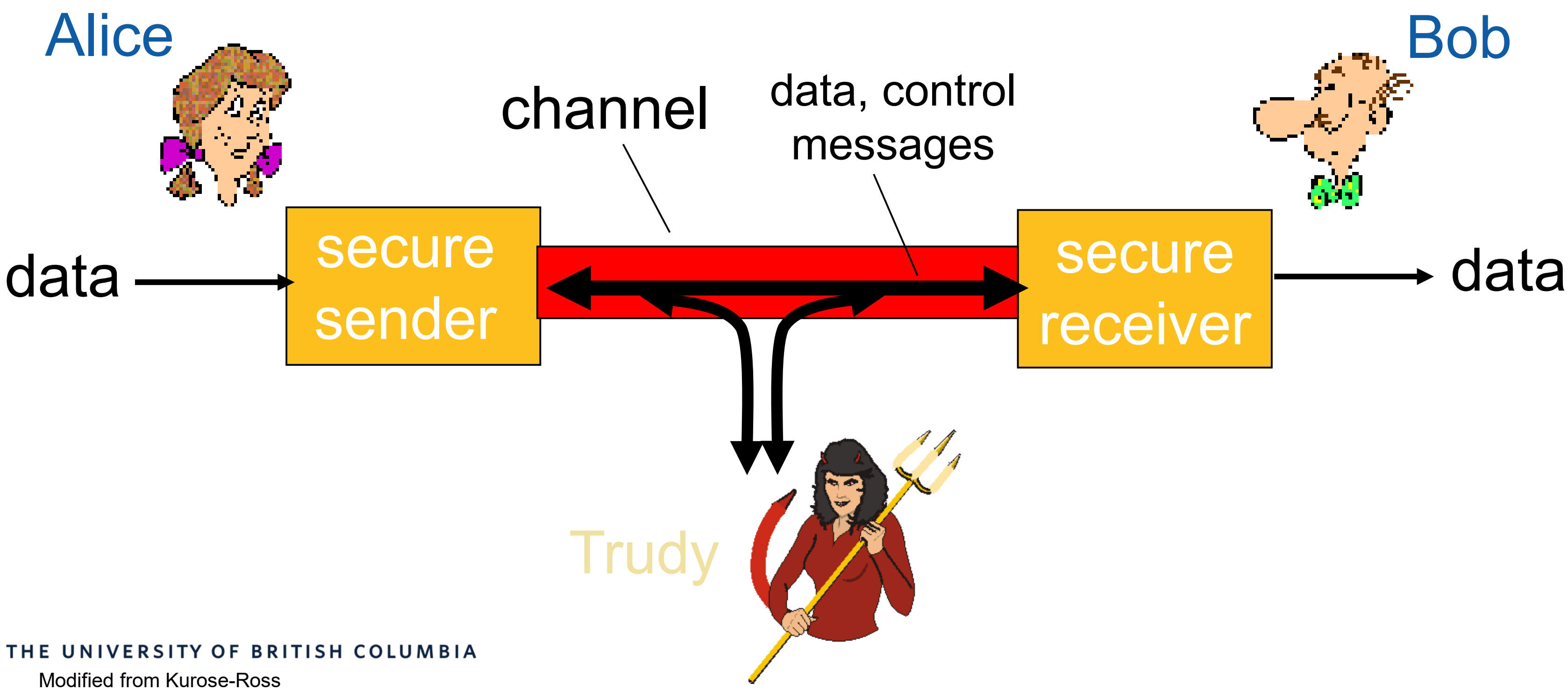
What are we trying to achieve?

- The ability to communicate securely

- What does that mean?

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- **A**lice and **B**ob 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
- DNS servers
- routers exchanging routing table updates
- other examples?

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: a lot!

- *eavesdrop:* intercept messages
- actively *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)

more on this later ……

What is network security?

Confidentiality

Authentication

Message Integrity

Access and Availability

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

Cryptography

- Derives from the Greek words '*kruptos*' meaning "secret" and '*graphia*' meaning "writing".

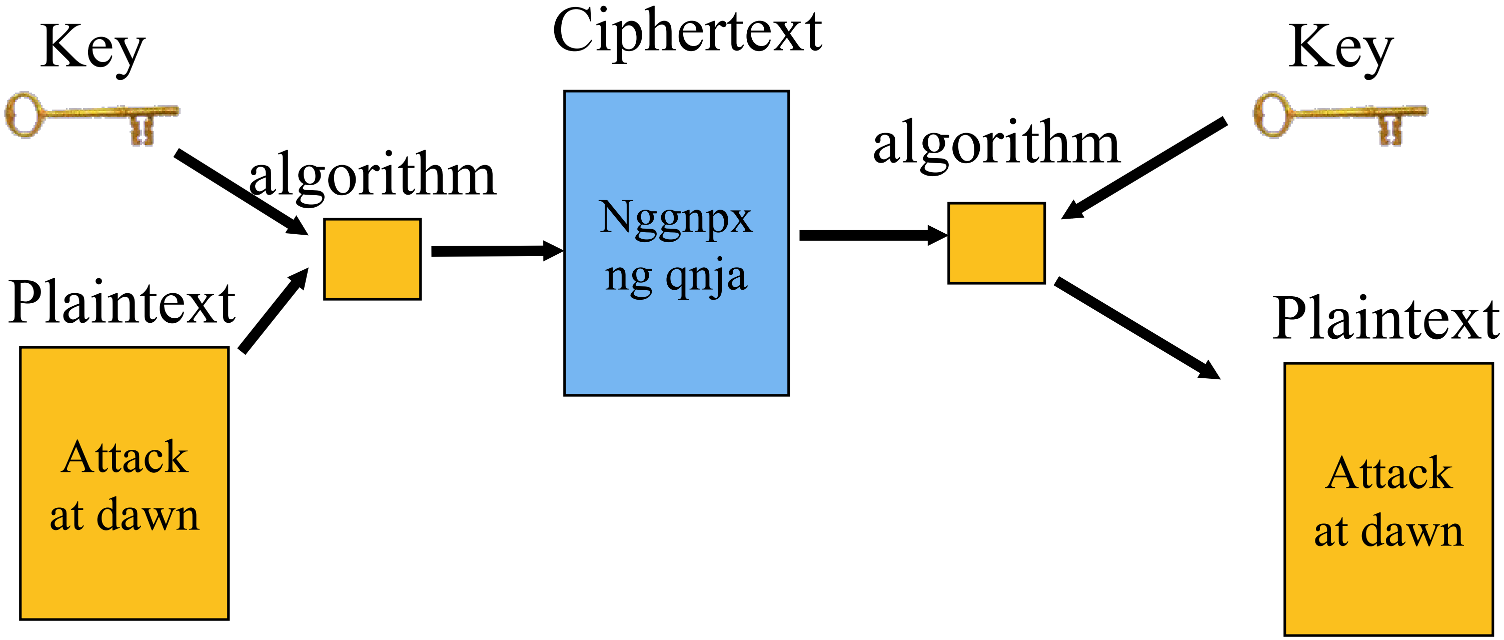
- Non-mathematical introduction

- Cryptographers toolkit of what is possible

- Basic: ensuring privacy

Algorithms and Keys

- All cryptography uses algorithms and keys, no matter how simple or complex.
  - E.g.. Rot13, Rot is the algorithm, 13 is the key



One Time Pads

- The only genuinely unbreakable cipher.
- Works when <u>one</u> unique key is used <u>once</u> to encipher/decipher <u>one</u> message.
- It works because, in this circumstance, all possible decryptions are equally as probable.
- Example: Two different meanings from the same ciphertext with two different keys.

ATTACKATDAWN

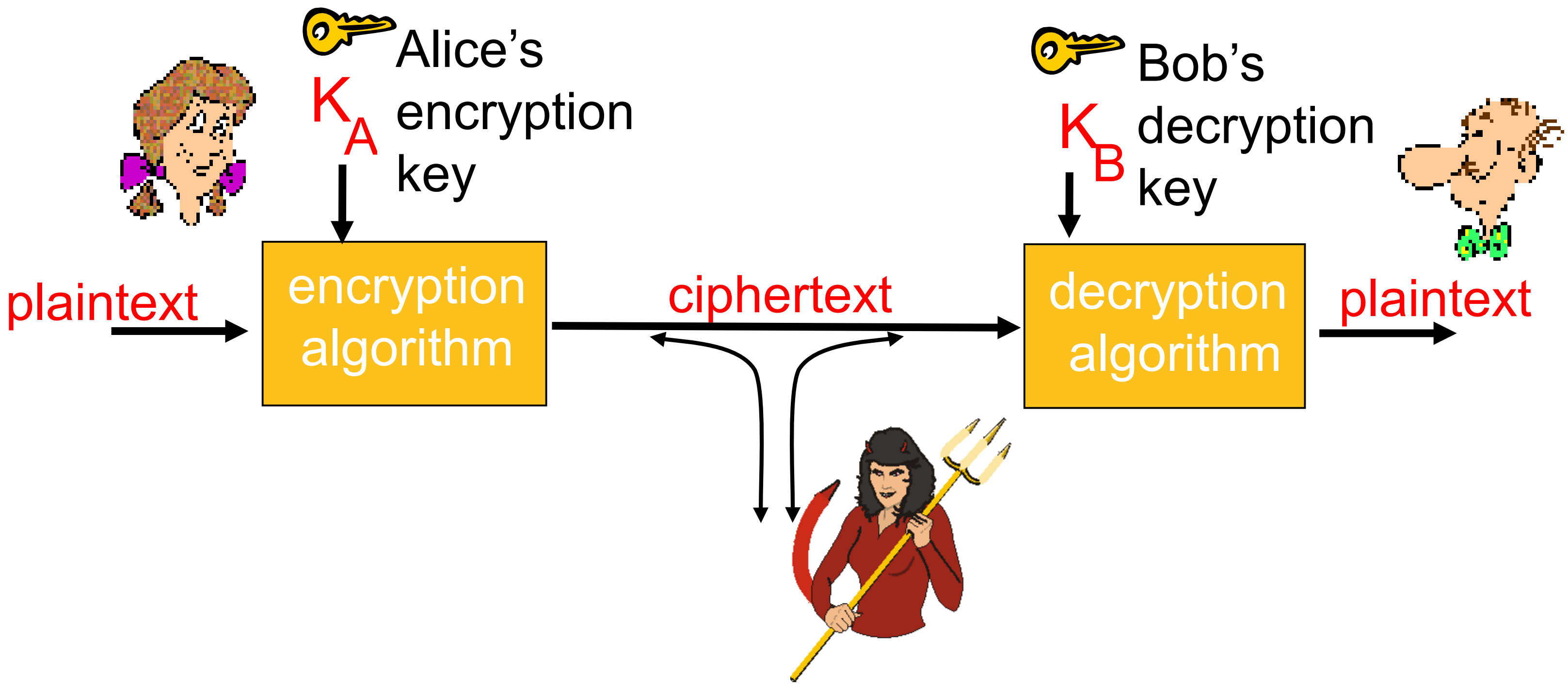
DXYCQXDXICKA

RETREATBYDAY

Can become either depending on key

- No way to know which key is correct. All keys are equally probable.

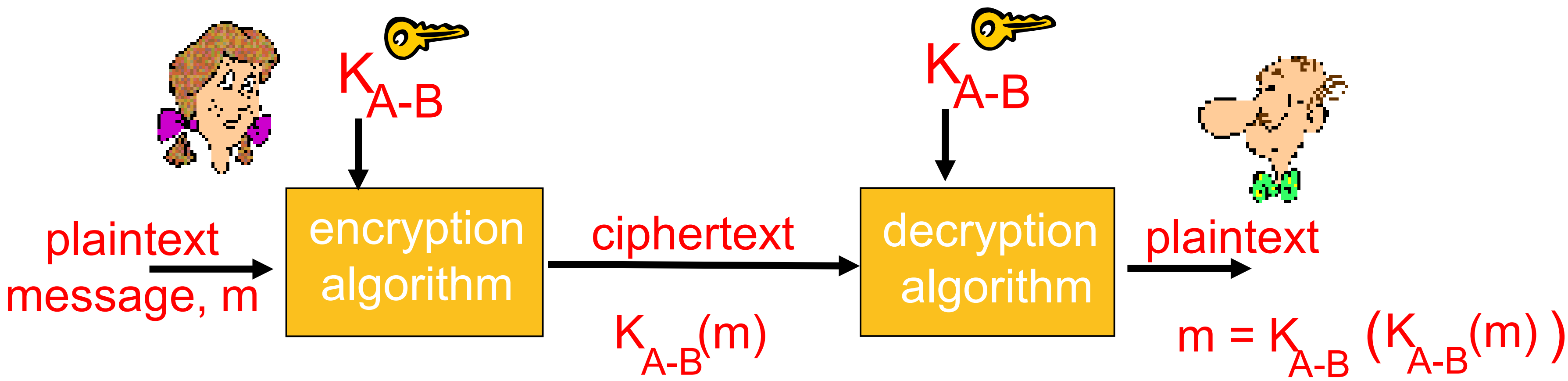
The language of cryptography



symmetric key crypto: sender, receiver keys *identical*

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

Symmetric key cryptography



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

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: how do Bob and Alice agree on key value?

Symmetric key cryptography

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
 plaintext:  abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

E.g.:

```
 Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Q: How hard to break this simple cipher?:

- ❑ brute force (how hard?)
- ❑ other?

Vigenère square

- Pick a keyword?
  - Repeated word
  - Could be text from a book
- Encode, letter in column and key letter in row, gives codeword letter
- Decode, look up the keyword letter in the row (e.g Q) and find the codeword letter, answer is the column letter.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

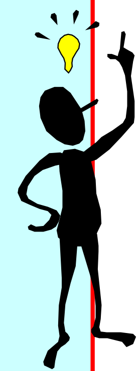
Public Key Cryptography

symmetric key crypto

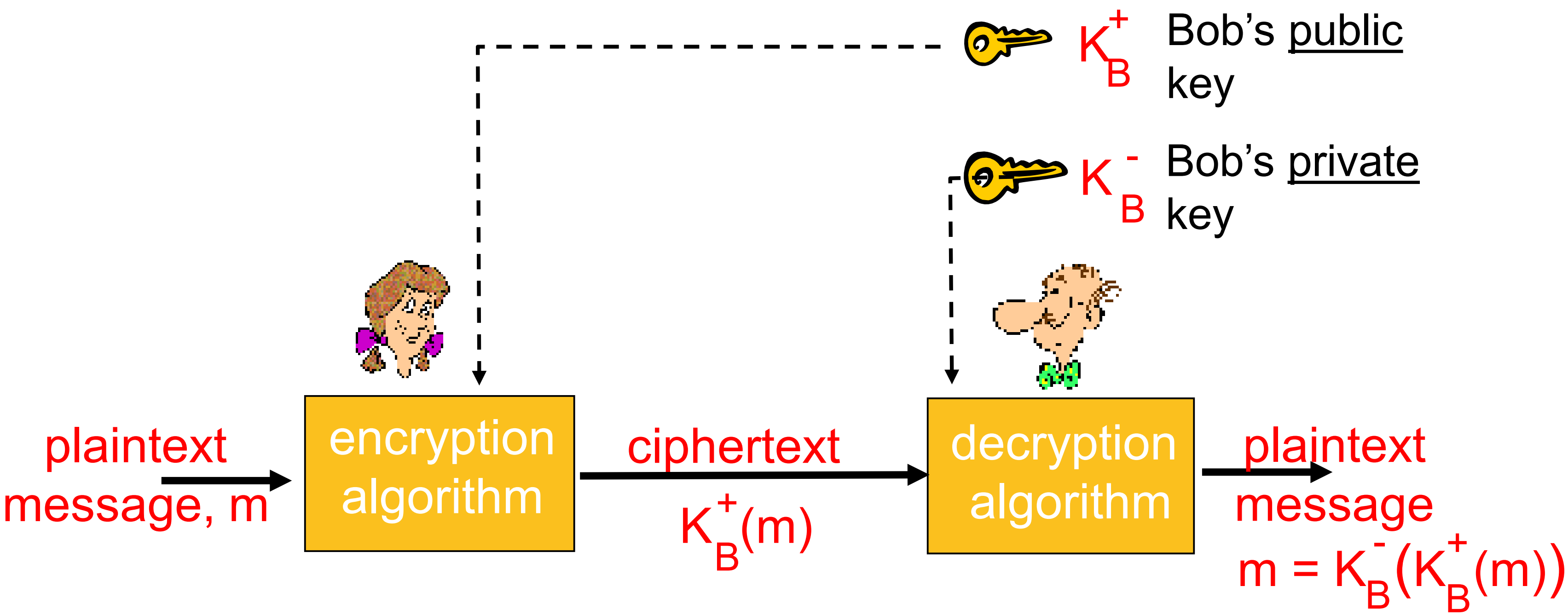
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key 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, Adelman algorithm

Digital Signatures

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

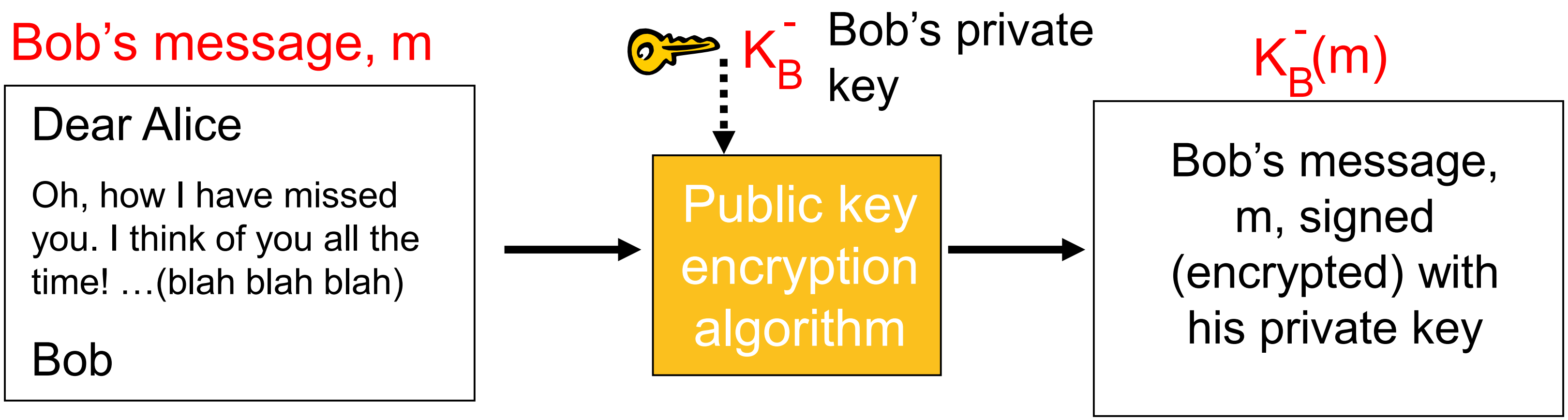
Cryptographic technique analogous to hand-written signatures.

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

Digital Signatures

Simple digital signature for message m:

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



Digital Signatures (more)

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

Alice thus verifies that:

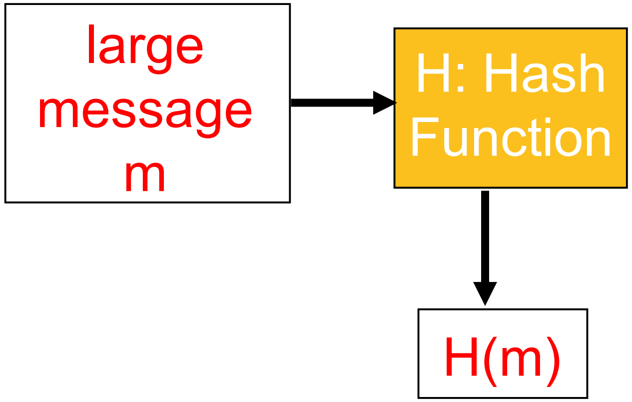
- ✓ Bob signed m.
- ✓ No one else signed m.
- ✓ Bob signed m and not m'.

Non-repudiation:

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

Digital Digests

Message Digests



Computationally expensive to public-key-encrypt long messages

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

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

Hash function properties:

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

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 4F 42
	B2 C1 D2 AC

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

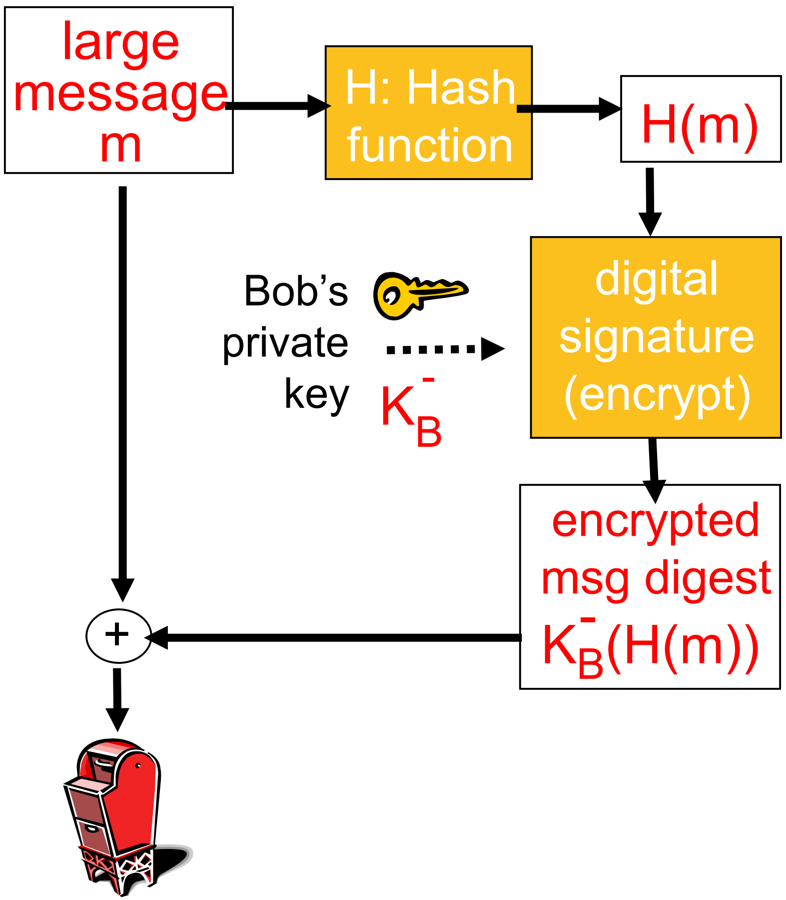
different messages but identical checksums!

Hash Function Algorithms

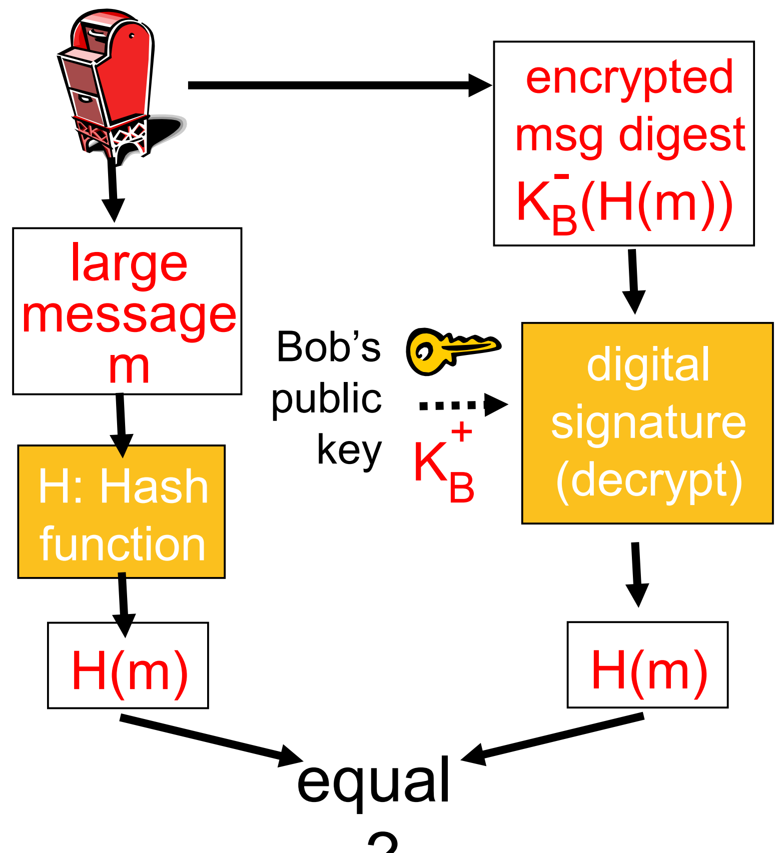
- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x.
- SHA-1 is also used. (SHA-2, SHA-3)
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature and integrity of digitally signed message:



Summary

- Encryption: (symmetric and asymmetric)
- Authentication: digitally sign documents
- Message Integrity: Digital Digests – Message Authentication Code (HMAC – Hash MAC).

Authentication

Play-back

Man-in-the-middle attacks

Authentication

Goal: Bob wants Alice to “prove” her identity to him

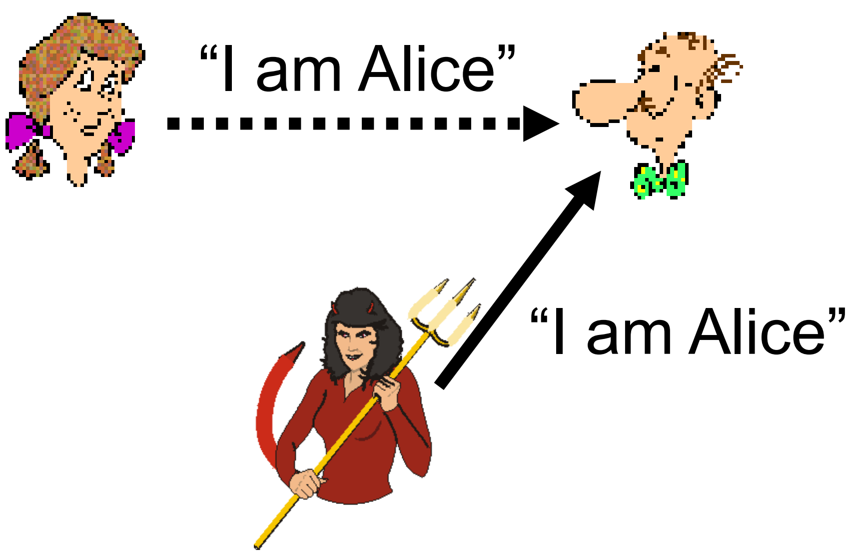
Protocol ap1.0: Alice says “I am Alice”

Failure scenario??

Authentication

<u>Goal:</u> Bob wants Alice to “prove” her identity to him

<u>Protocol ap1.0:</u> Alice says “I am Alice”



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

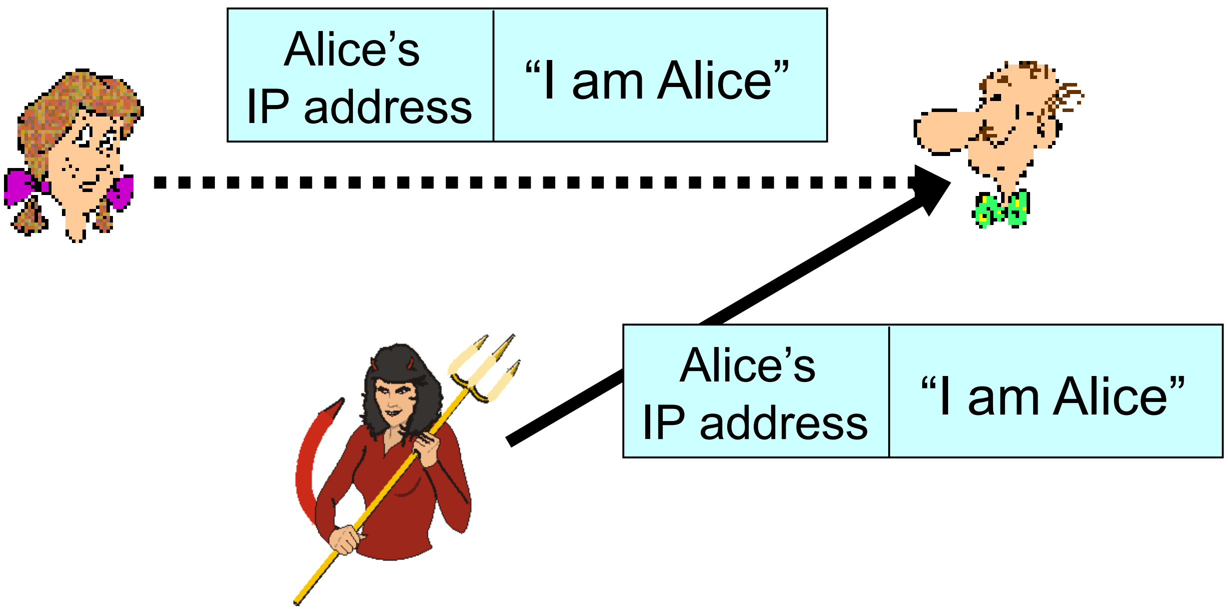
Authentication: another try

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

Failure scenario??

Authentication: another try

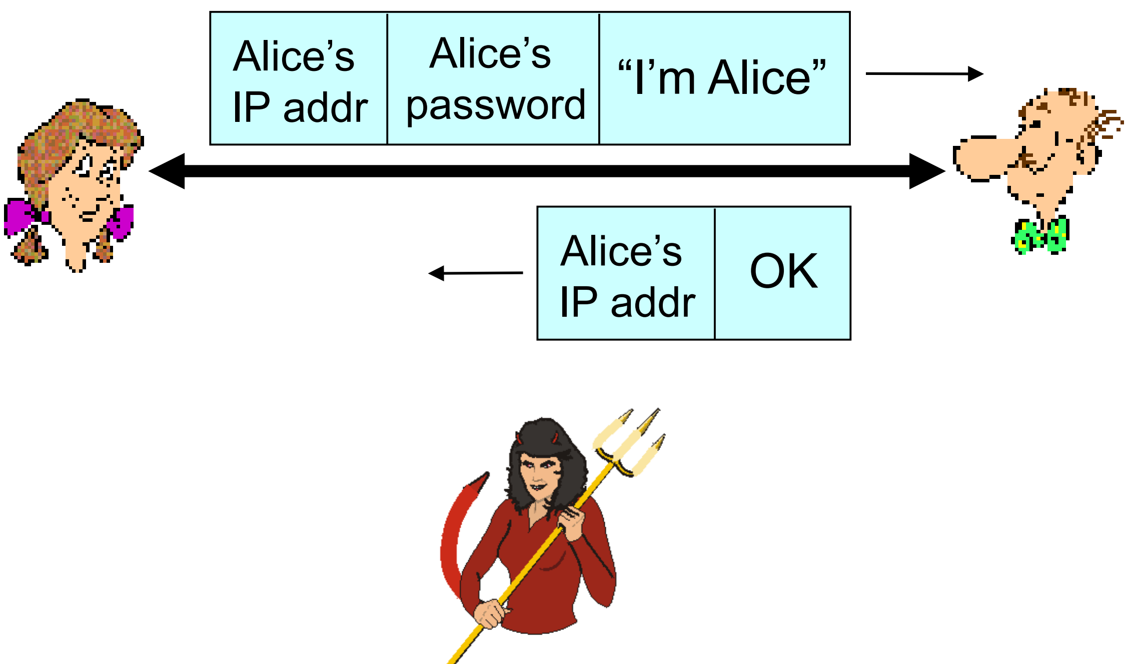
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



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

Authentication: another try

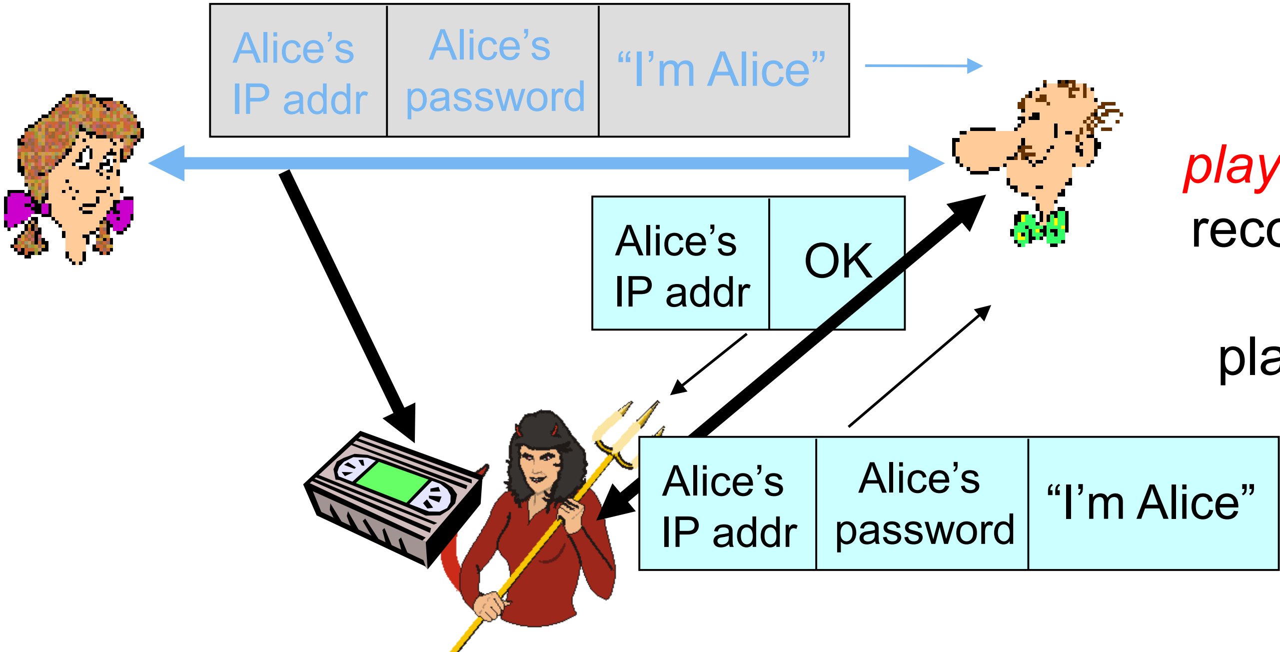
<u>Protocol ap3.0:</u> Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

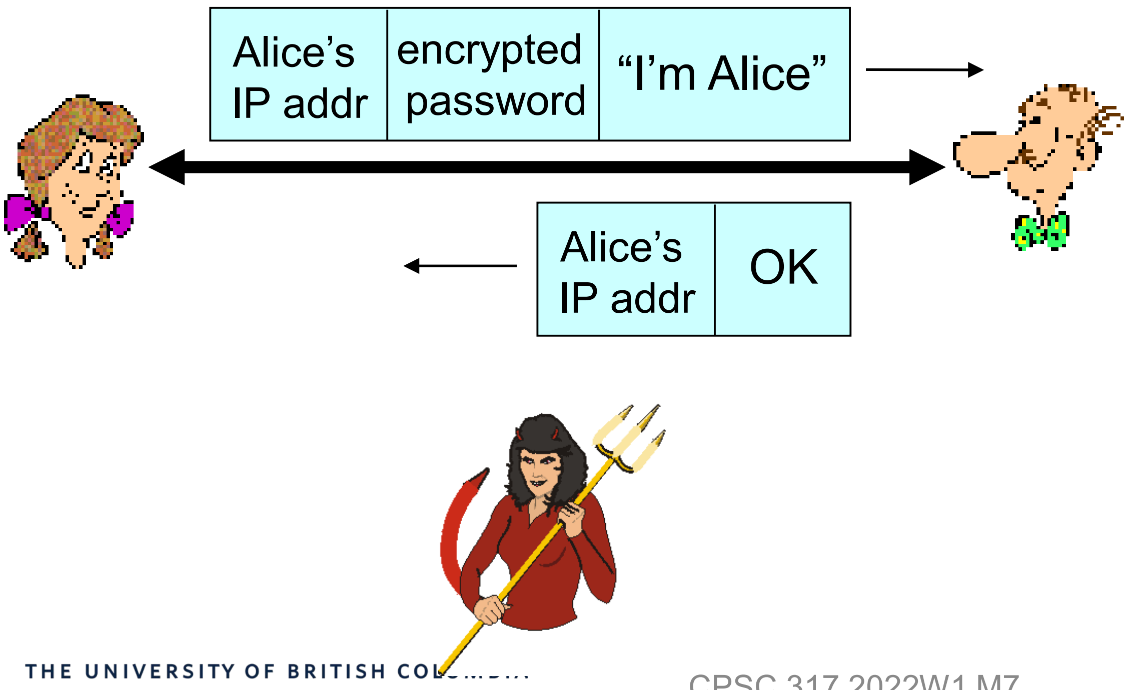
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



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

Authentication: yet another try

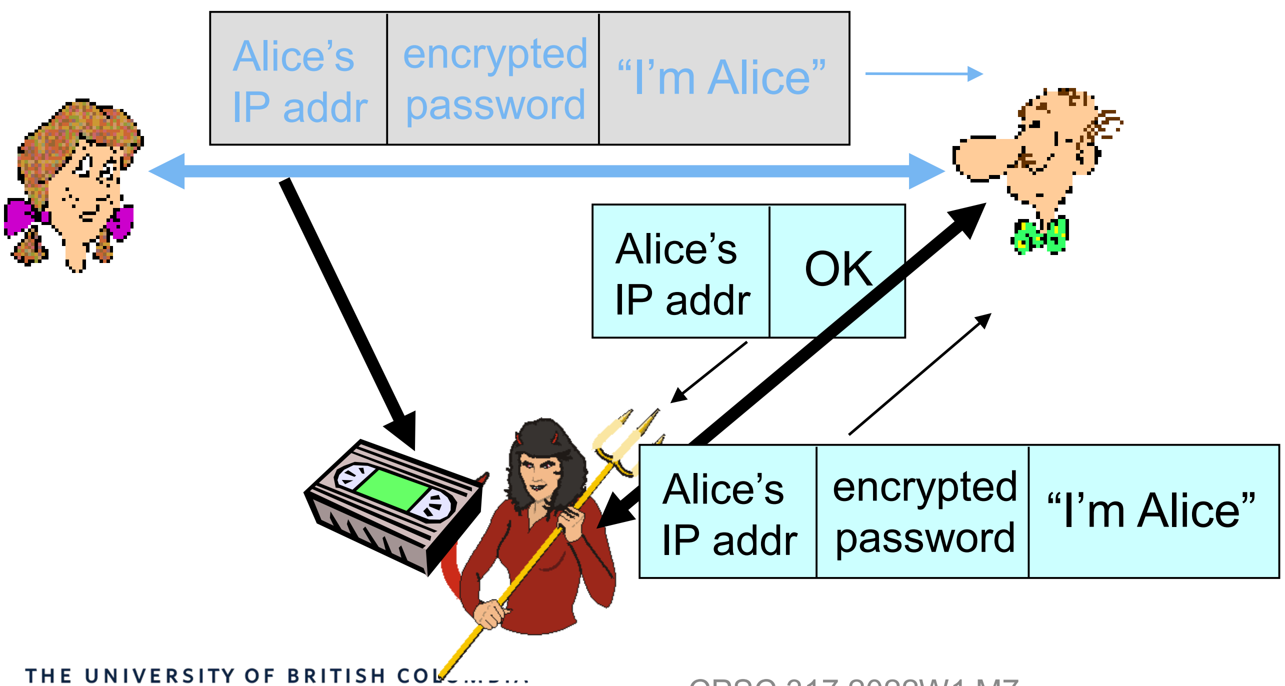
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



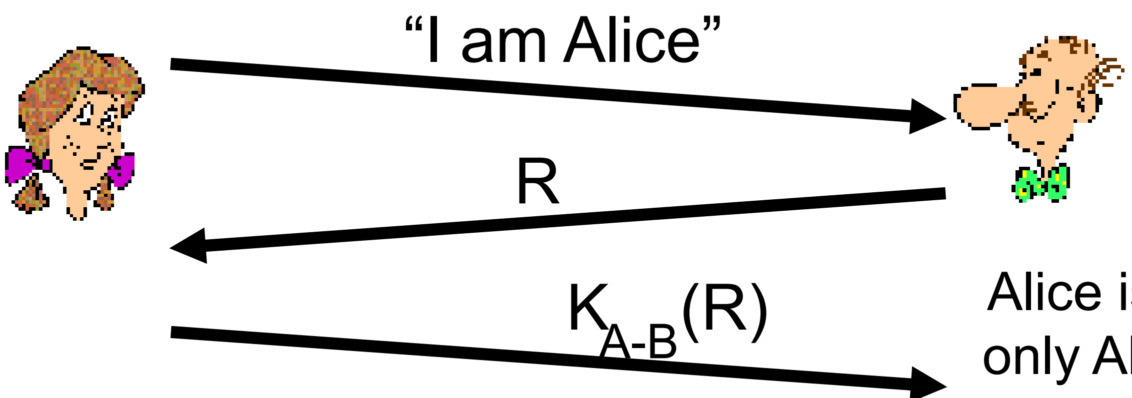
record
and
playback
still works!

Authentication: yet another try

Goal: avoid playback attack

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

ap4.0: 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!

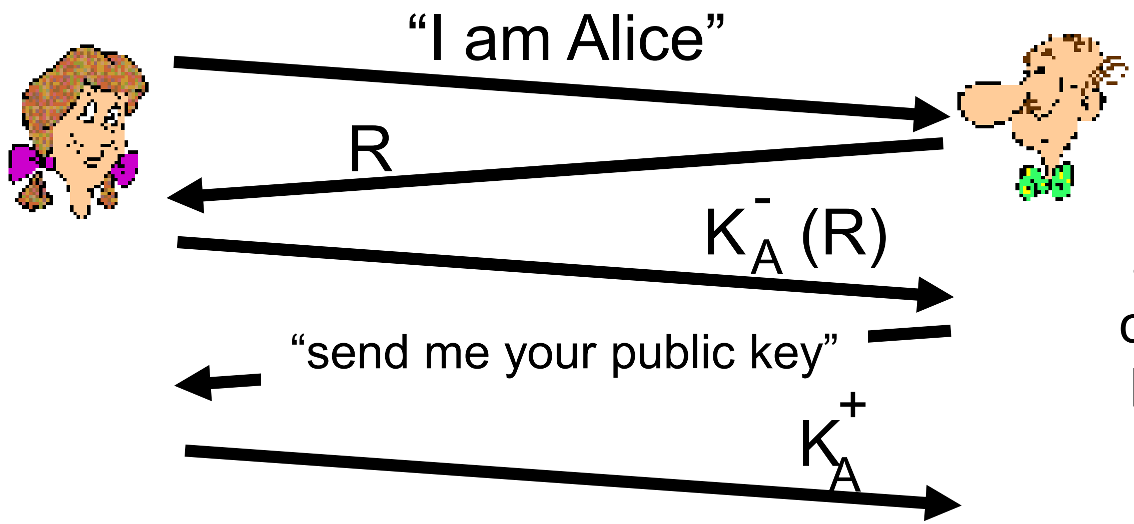
Failures, drawbacks?

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

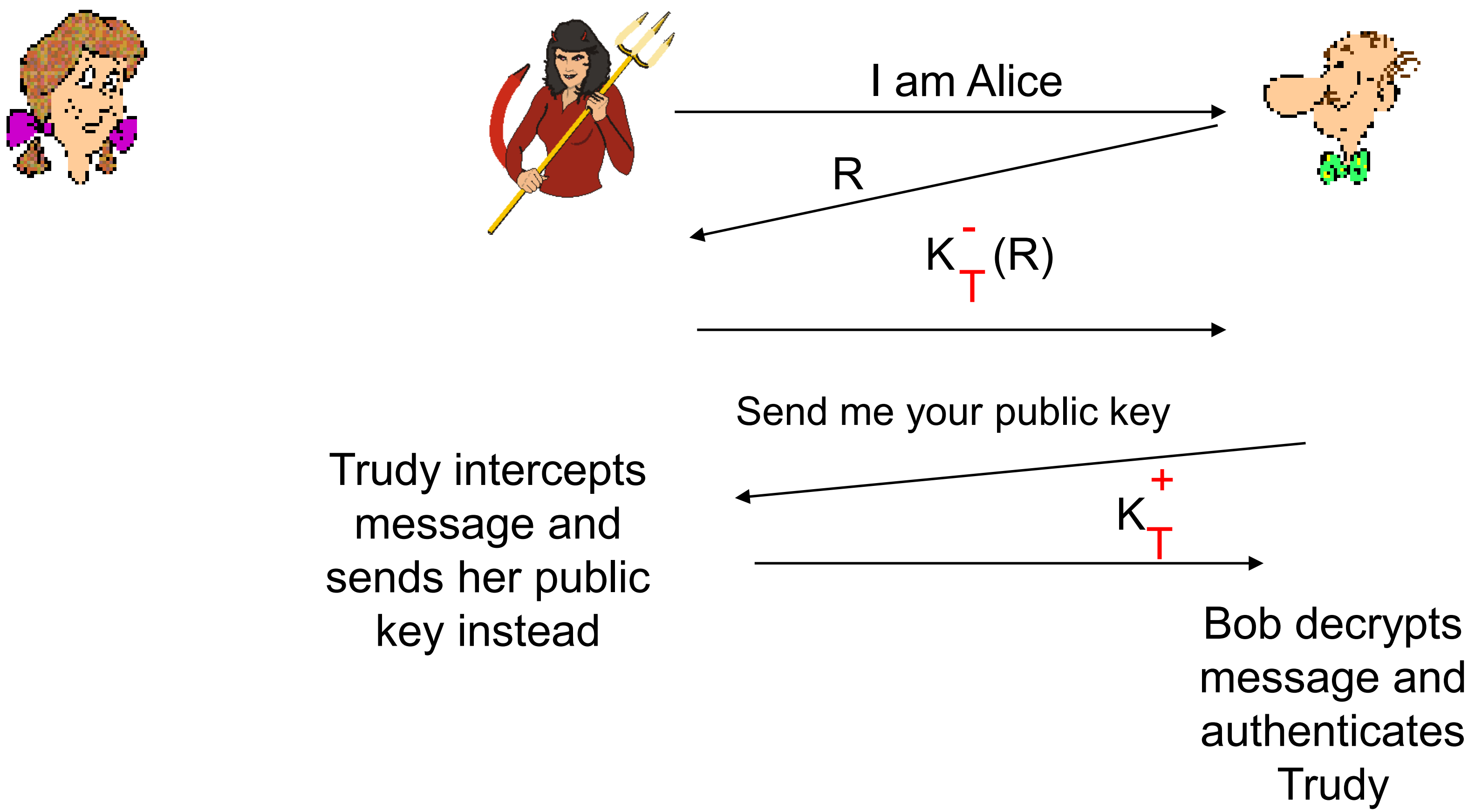
$$K_A^+(K_A^-(R)) = R$$

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

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

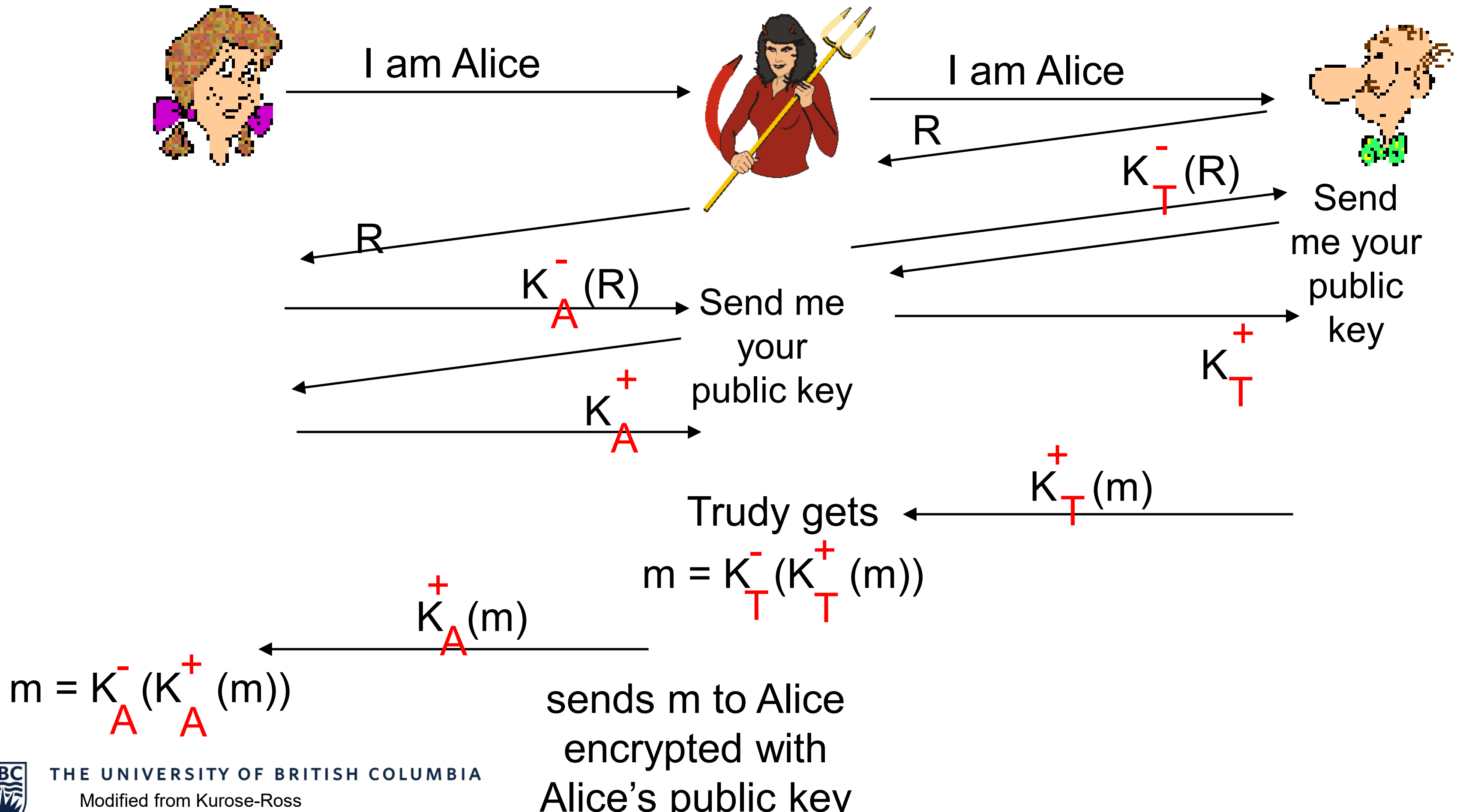
ap5.0: security hole

Trudy poses as Alice (to Bob) and as Bob (to Alice)



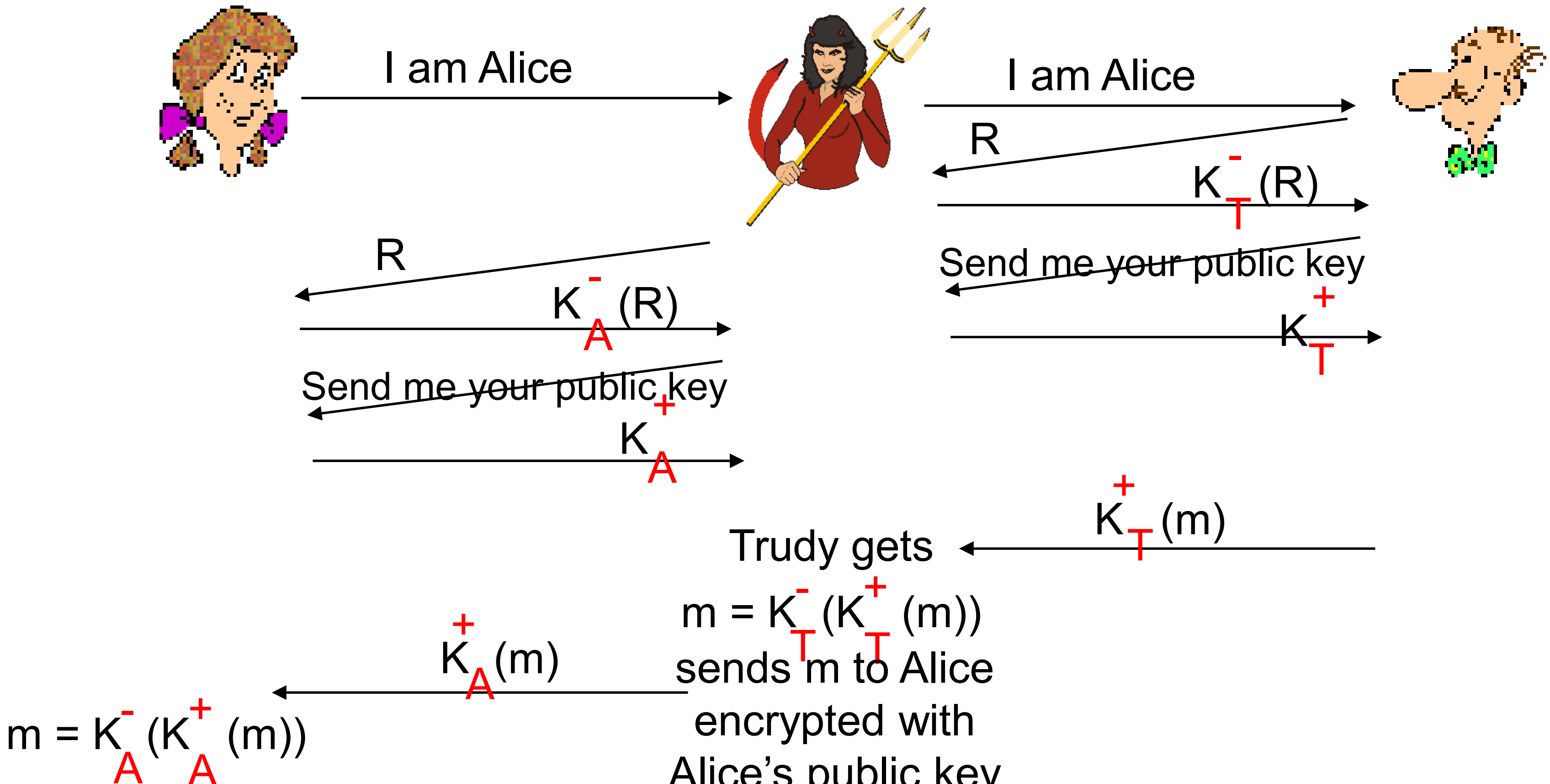
ap5.0: security hole

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



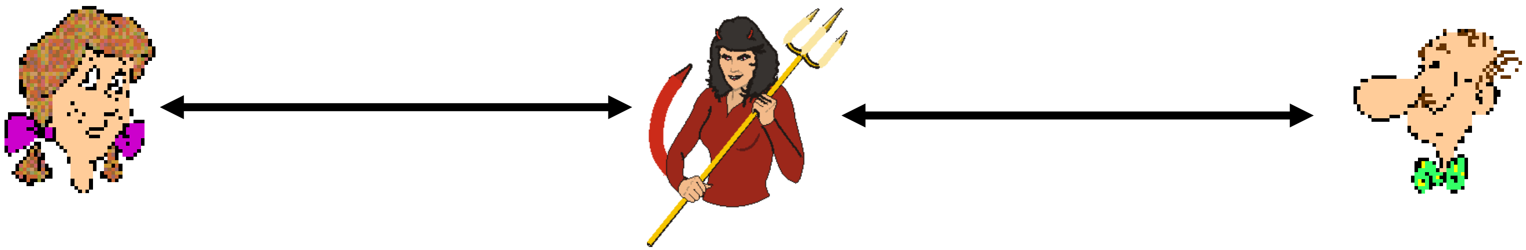
ap5.0: security hole

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



ap5.0: security hole

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



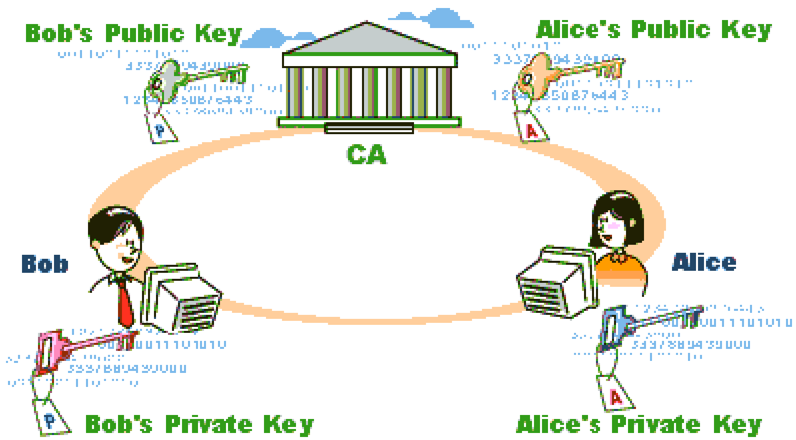
Difficult to detect:

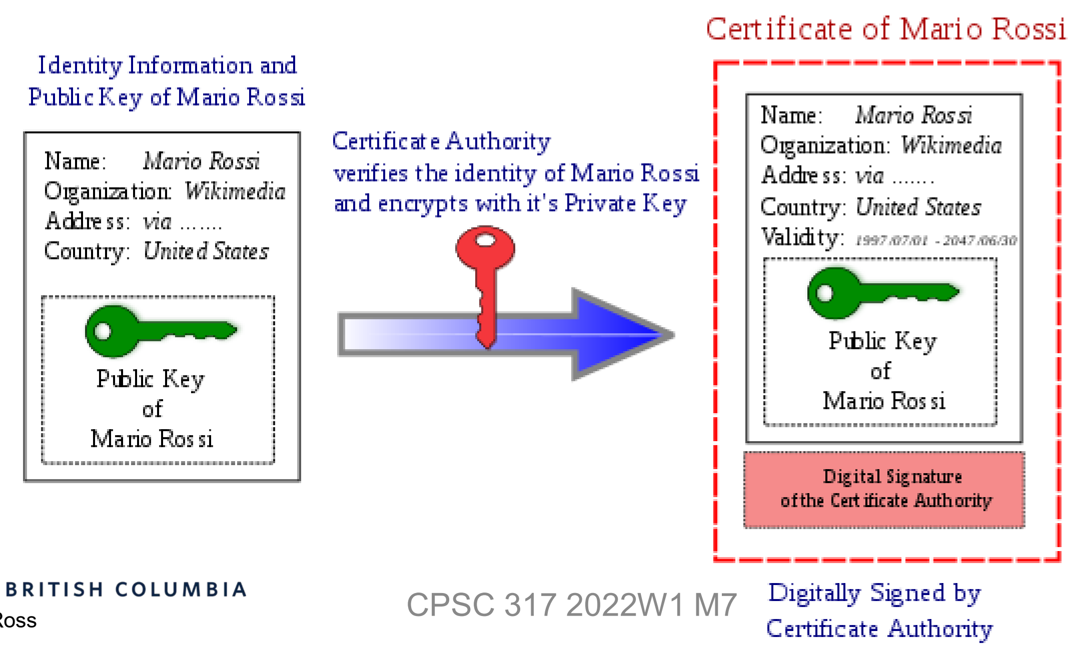
- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation)
- problem is that Trudy receives all messages as well!

SECURITY PROTOCOLS in PRACTICE

TRUST

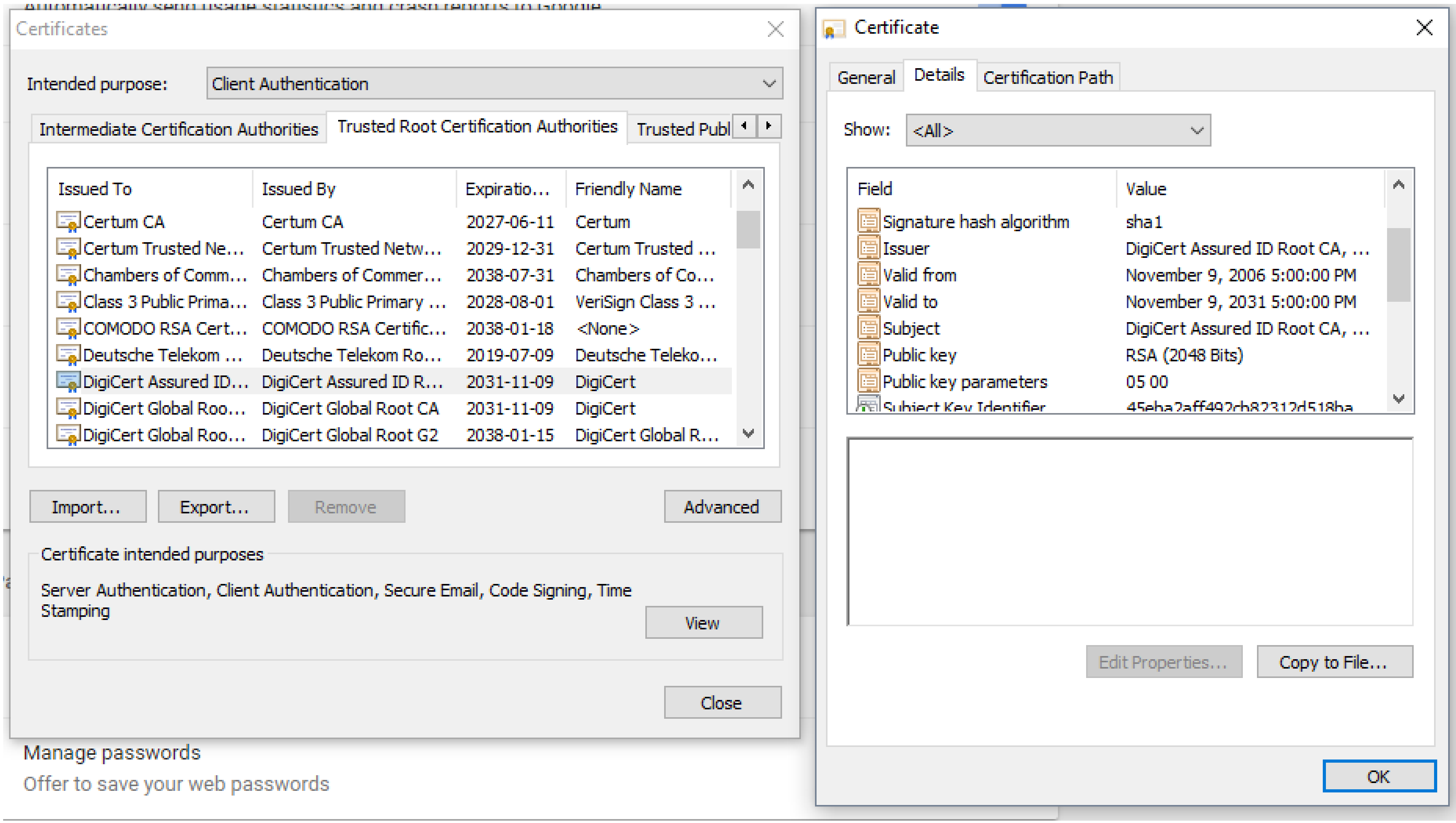
Public Key Intra-structure





Public Key Infrastructure Certificates

- Certificate authority (CA)
  - Provides "proof of identity"
  - Every host keeps a list of trusted authorities' public keys
- A server can present a certificate
  - Content is the server's identity and public key information (among others)
  - Encrypted using a CA's private key
  - Client can decrypt using that CA's public key
- Pretty Good Privacy (PGP) keyrings


Certificates
Intended purpose: Client Authentication
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Publ
Issued To | Issued By | Expiratio... | Friendly Name
Certum CA | Certum CA | 2027-06-11 | Certum
Certum Trusted Ne... | Certum Trusted Netw... | 2029-12-31 | Certum Trusted ...
Chambers of Comm... | Chambers of Commer... | 2038-07-31 | Chambers of Co...
Class 3 Public Prima... | Class 3 Public Primary ... | 2028-08-01 | VeriSign Class 3 ...
COMODO RSA Cert... | COMODO RSA Certific... | 2038-01-18 | <None>
Deutsche Telekom ... | Deutsche Telekom Ro... | 2019-07-09 | Deutsche Teleko...
DigiCert Assured ID... | DigiCert Assured ID R... | 2031-11-09 | DigiCert
DigiCert Global Roo... | DigiCert Global Root CA | 2031-11-09 | DigiCert
DigiCert Global Roo... | DigiCert Global Root G2 | 2038-01-15 | DigiCert Global R...
Import...
Export...
Remove
Advanced
Certificate intended purposes
Server Authentication, Client Authentication, Secure Email, Code Signing, Time Stamping
View
Close
Manage passwords
Offer to save your web passwords
Certificate
General
Details
Certification Path
Show: <All>
Field | Value
Signature hash algorithm | sha1
Issuer | DigiCert Assured ID Root CA, ...
Valid from | November 9, 2006 5:00:00 PM
Valid to | November 9, 2031 5:00:00 PM
Subject | DigiCert Assured ID Root CA, ...
Public key | RSA (2048 Bits)
Public key parameters | 05 00
Edit Properties...
Copy to File...
OK

APPLICATION LAYER
(TRANSPORT LAYER)

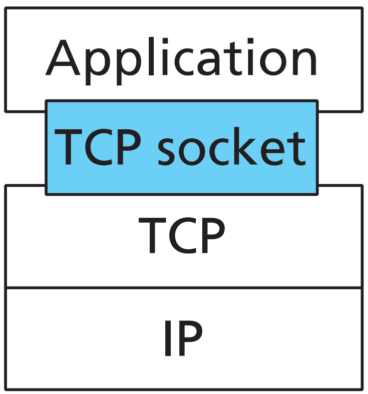
SSL: Secure Sockets Layer

- widely deployed security protocol
  - supported by almost all browsers, web servers
  - https
  - billions $/year over SSL
- mechanisms: [Woo 1994], implementation: Netscape
- variation - **TLS**: transport layer security, RFC 2246
- provides
  - *confidentiality*
  - *integrity*
  - *authentication*

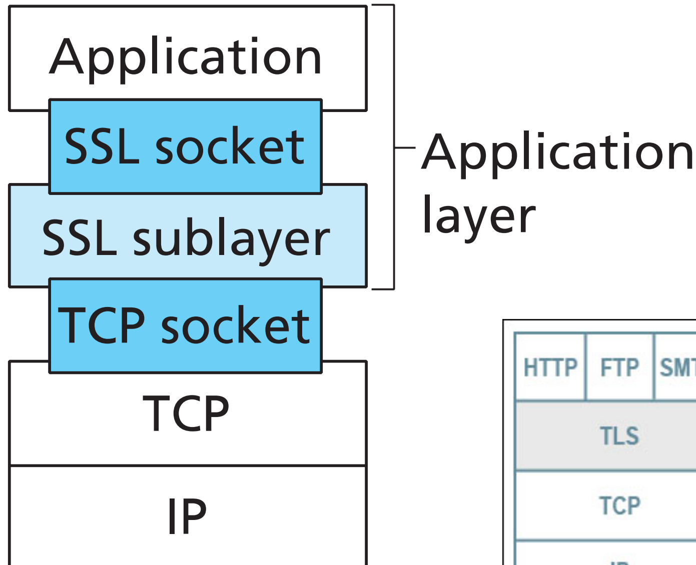
`https://......`

- 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

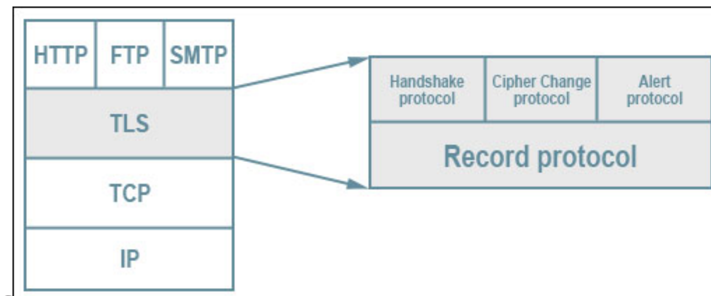
SSL (TLS) and TCP/IP



TCP API



TCP enhanced with SSL

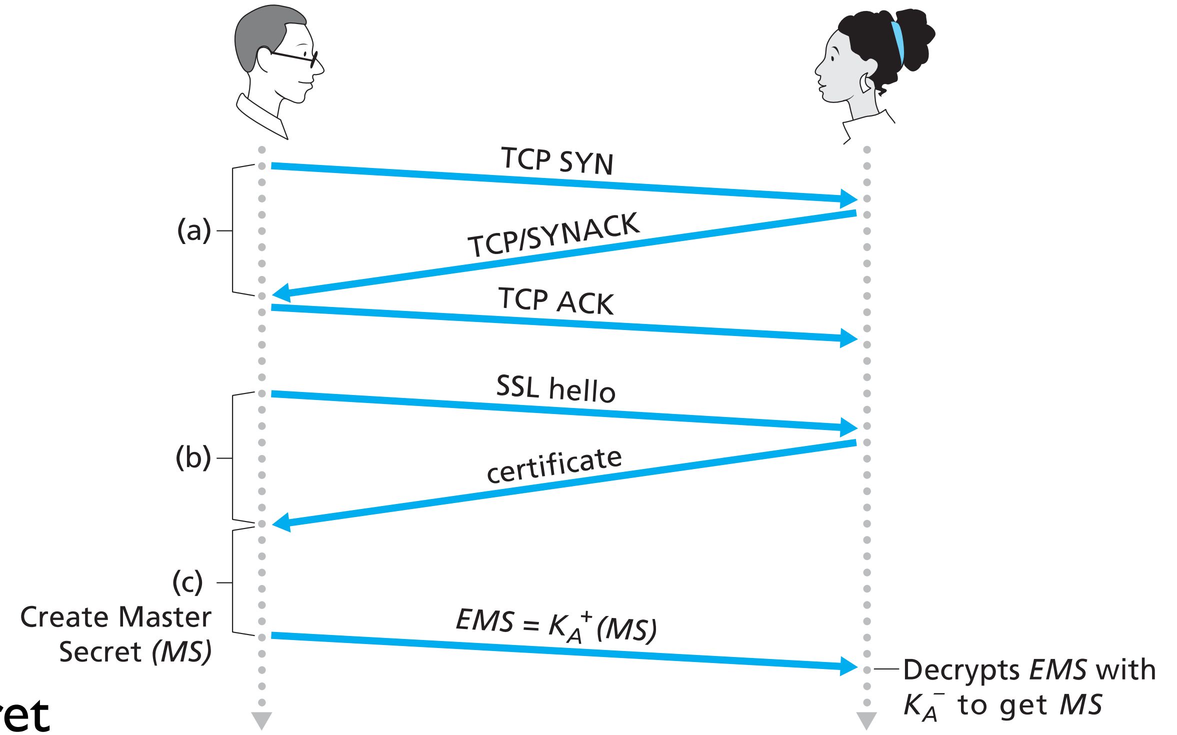


- SSL/TLS provides application programming interface (API) with socket to applications
- C and Java SSL libraries/classes readily available

Toy SSL: a simple secure channel

- *handshake:* after establishing TCP connection, Alice and Bob use their certificates (public keys), 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

Toy: a simple handshake



MS: master secret

EMS: encrypted master secret

- SSL hello: to verify that Alice is really Alice!
- (CA certified) certificate contains Alice's public key
- Send Alice a **encrypted master secret** key (EMS)

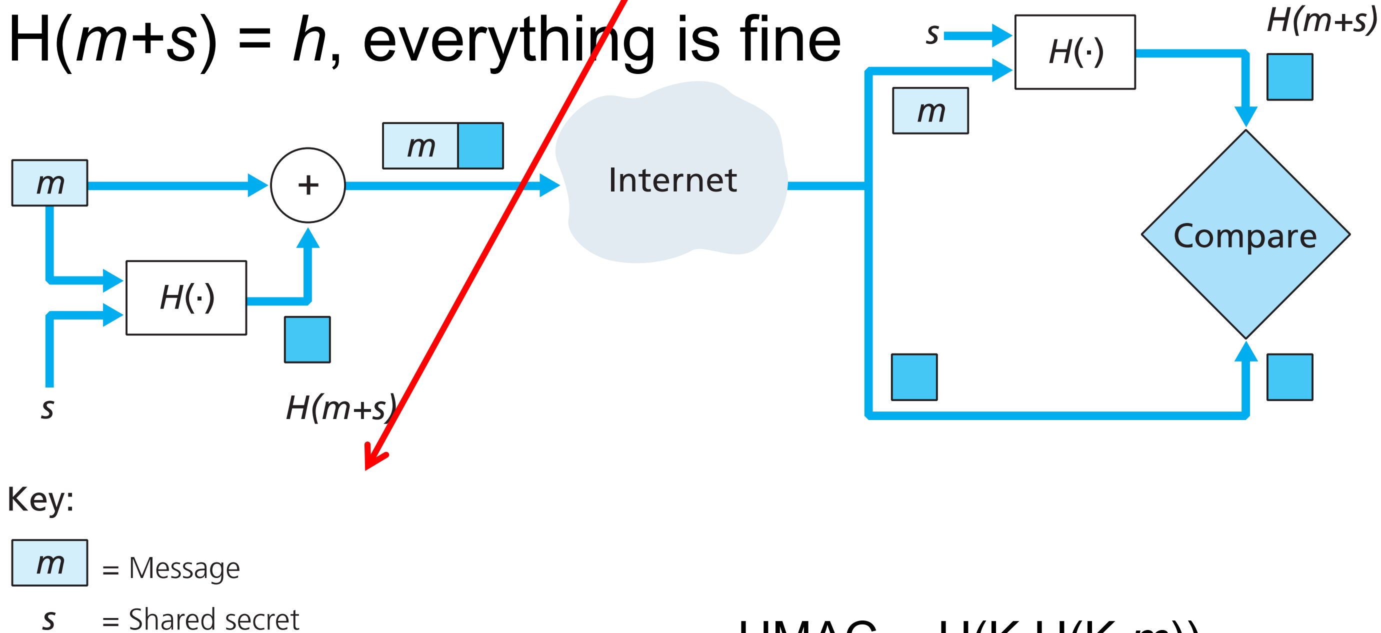
Both parties now share a **master secret** (for this SSL session)

Toy SSL: key derivation

- considered bad to use the same key for more than one cryptographic operation
  - use different keys for message authentication code (MAC) and encryption
- Client and server use MS to generate 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

Message Authentication Code (MAC)

- Alice and Bob share secret *s* (termed **authentication key**)
- Alice creates message *m*, create *m*+*s*, calculate H(*m*+*s*) [**Message Authentication Code (MAC)**]
- Alice send (*m*, H(*m*+*s*)) to Bob
- Bob receives (*m*, *h*); knowing s, Bob calculates MAC H(*m*+*s*); if H(*m*+*s*) = *h*, everything is fine

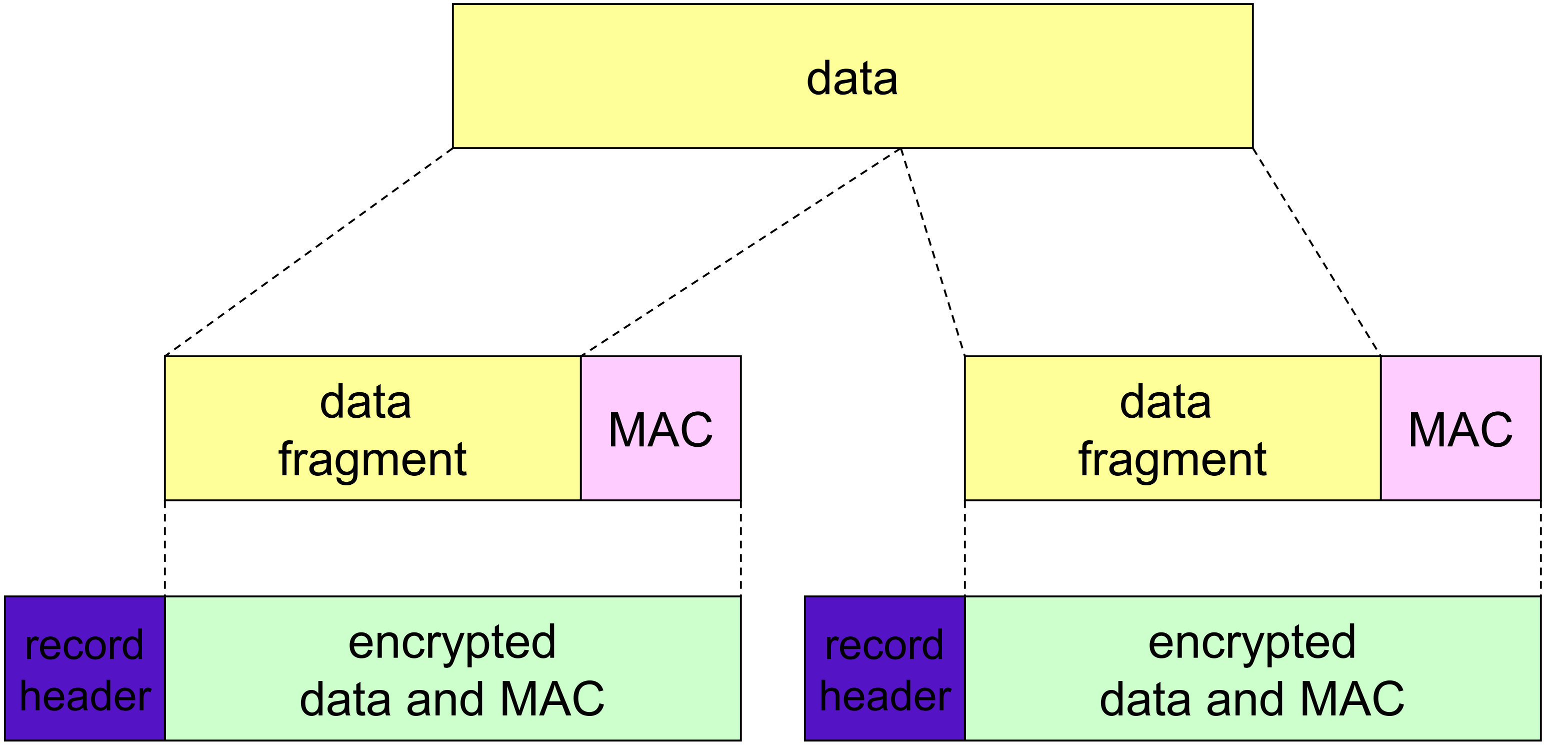


HMAC -- H(K,H(K,*m*))

Toy SSL: data records

- As TCP is a byte-stream protocol, how would you encrypt data?
- Would like to encrypt (application) data in constant stream, on the fly, and then pass the encrypted data, on the fly, to TCP
  - where would we put the MAC?
  - If at the end, no message integrity until all data processed
- instead, break stream in series of **records**
  - each record carries a MAC for integrity check
  - encrypt each [record + 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

Toy SSL: data records



Toy SSL: sequence numbers

- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = H(M_x, **sequence** + data)
  - note: no sequence number field

- *problem:* attacker could **replay** **all** records
- *solution:* use nonce

Toy: control information

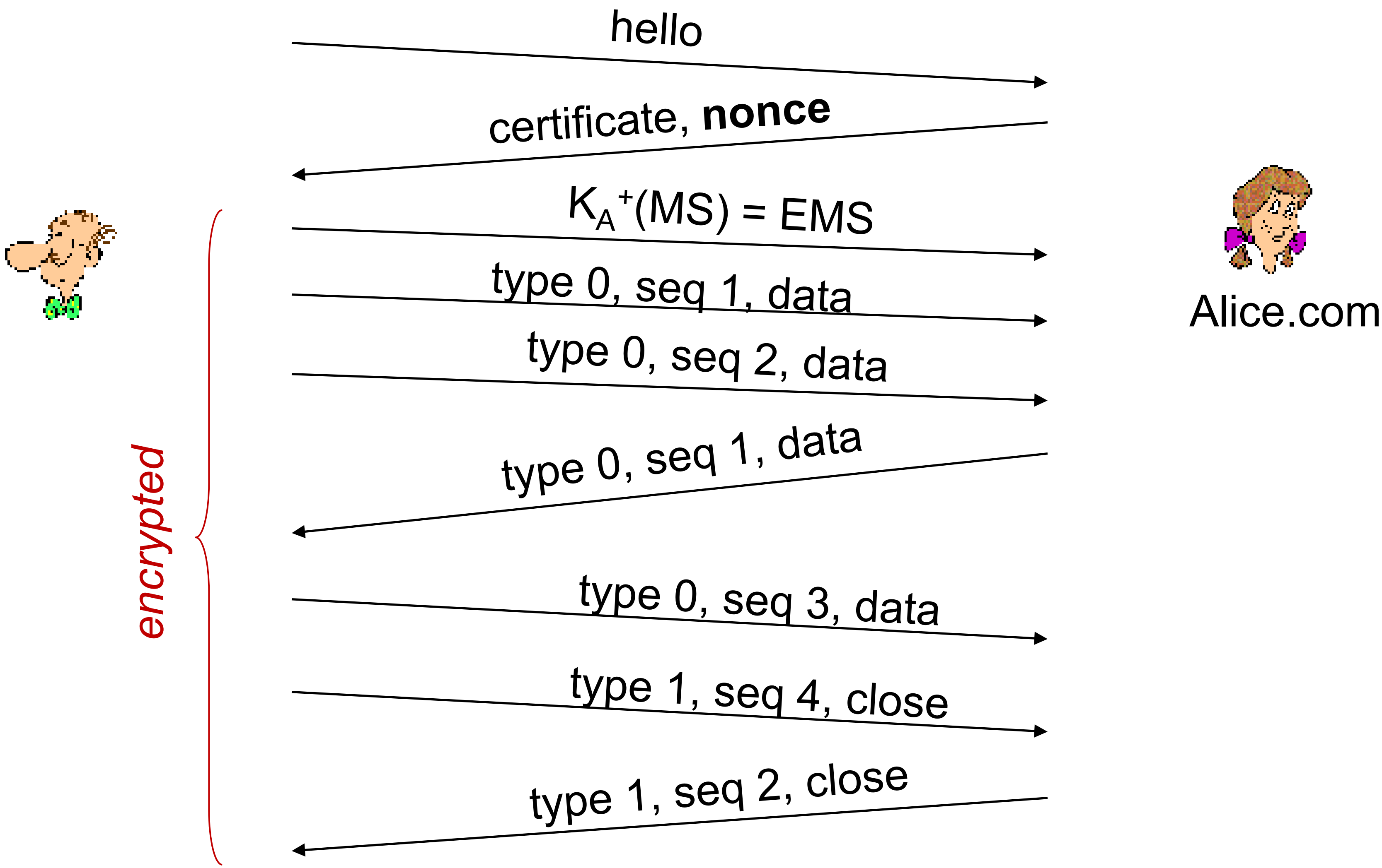
- *problem:* truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
  - type 0 for data; type 1 for closure
- MAC = MAC(M_x, sequence||type||data)

Type	Version	Length	Data	MAC

Type: indicates either handshake msg or data msg

$\underbrace{\text{Data} \quad \text{MAC}}_{\text{Encrypted with } E_B}$

Toy SSL: summary



SSL isn't complete

- cipher suite
  - public-key algorithm
  - symmetric encryption algorithm
  - MAC algorithm
- Want to support multiple ciphers: SSL supports several cipher suites
- Want 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
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL Public key encryption

- RSA

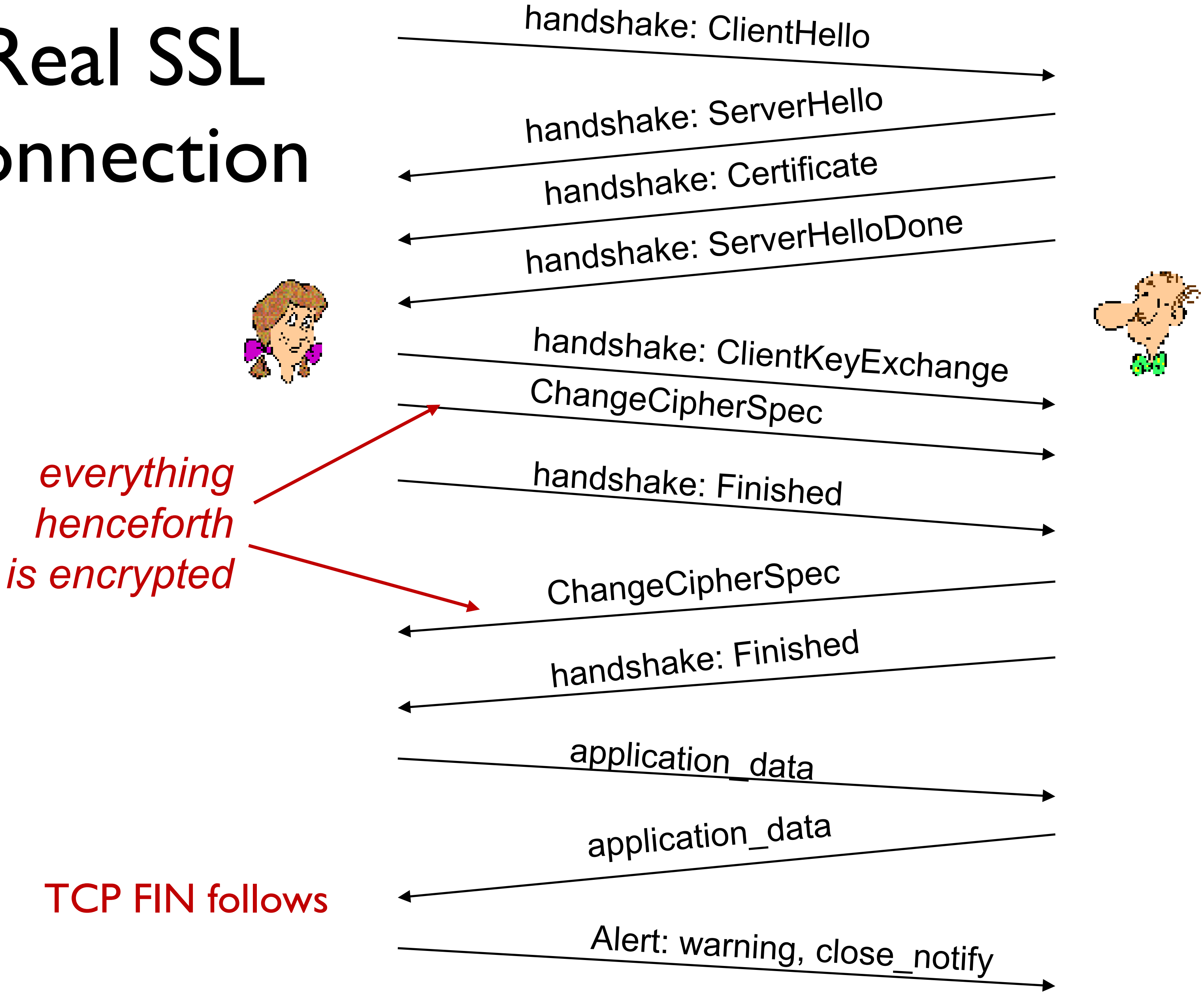
Real SSL: handshake

Why?

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
7. client authentication optional

Why?

Real SSL connection

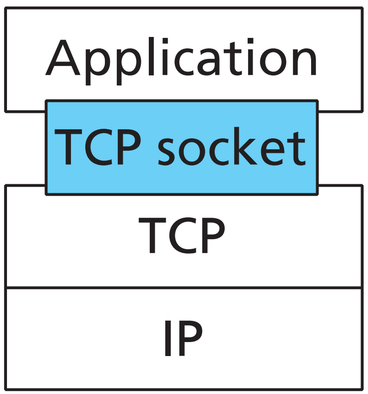


Connection Closure

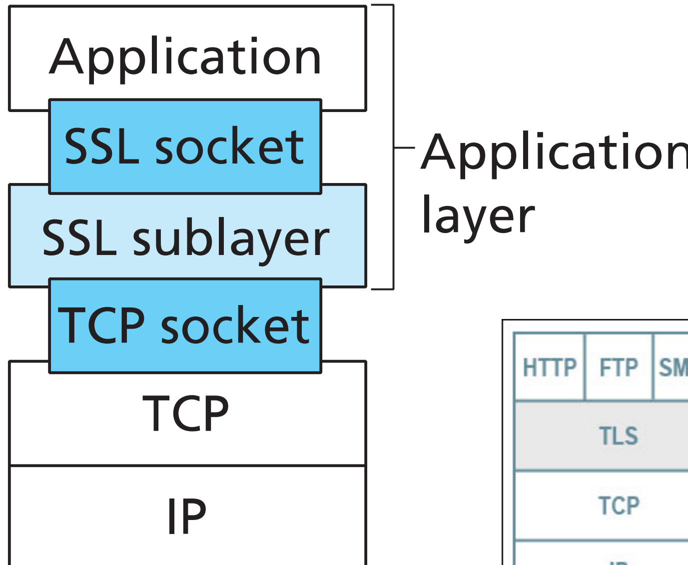
- Using TCP FIN alone is not secure
  - Allow **truncation attack** where anyone else could end the SSL session
- Indicate in the **type** field of record whether the record serves to terminate the SSL session (termed closure SSL record)
  - Although SSL type is sent in the clear, it is authenticated at receiver using the record's MAC
  - If a TCP FIN were received earlier, something funny is going on

TRANSPORT LAYER

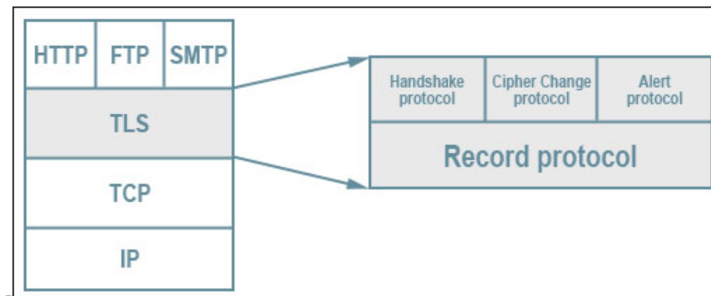
SSL (TLS) and TCP/IP



TCP API



TCP enhanced with SSL

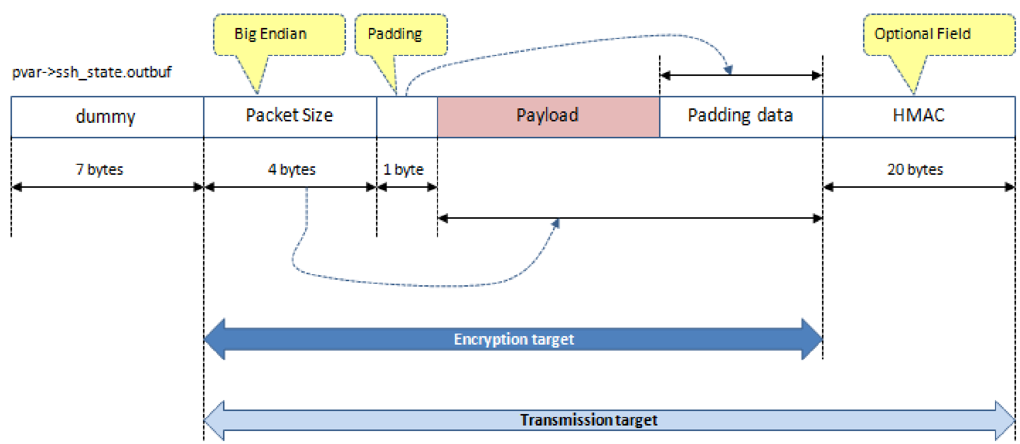


- SSL/TLS provides application programming interface (API) with socket to applications
- C and Java SSL libraries/classes readily available

Secure Shell (ssh,scp)

- Replaces telnet, rsh, rcp,
- Runs over a TCP connection
- Put into a TCP connection

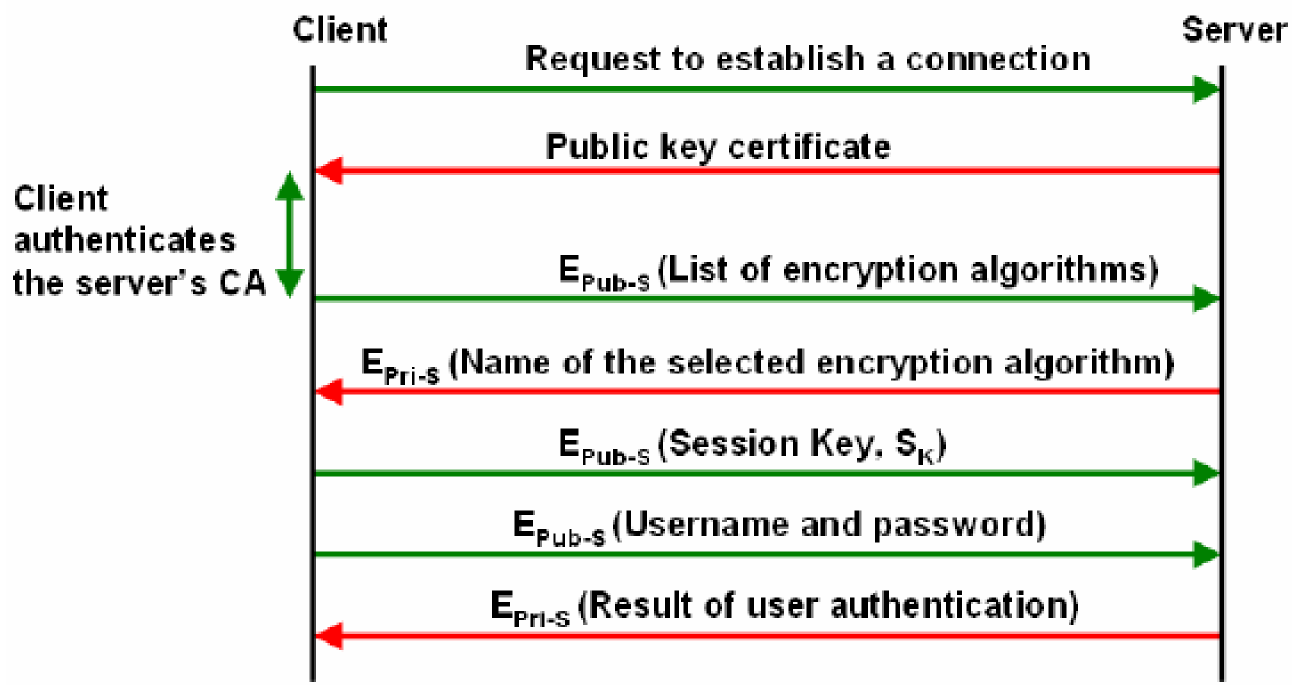
SSH2 Packet Format (without compression)



Once the packet construction and encryption has been completed, SSH2 packet is transmitted in **finish_send_packet_special ()**.
This is TCP communication with **send_packet_blocking ()**.

Secure Shell (ssh,scp)

- SSH-TRANS (C), SSH-AUTH (A), SSH-CONN



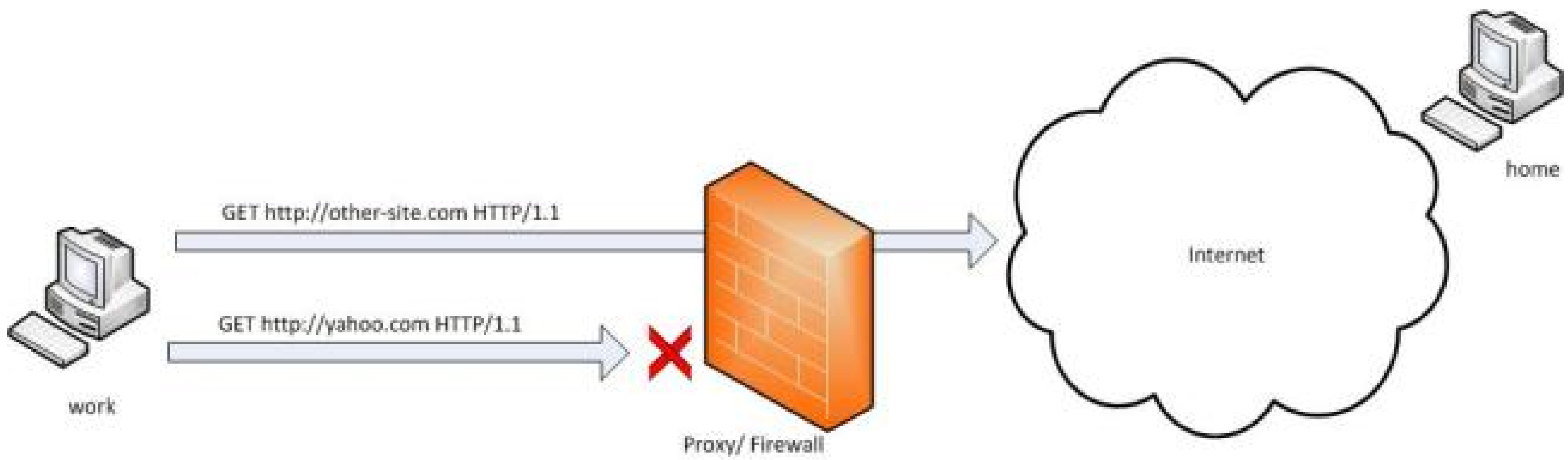
- Ssh-keygen, .ssh, known-hosts, authorized_keys

Tunnels

- Idea of putting a transport/network packet inside another transport/network packet (not exclusive to that level)
- Comes up in a variety of places:
  - IP inside IP
  - Tunnelling IPv6 inside IPv4 packets
  - Tunnelling IPsec over IP packets
  - MPLS tunnels, IP over tag-switching fabrics
- Used in a variety of VPN technologies.

Secure Shell

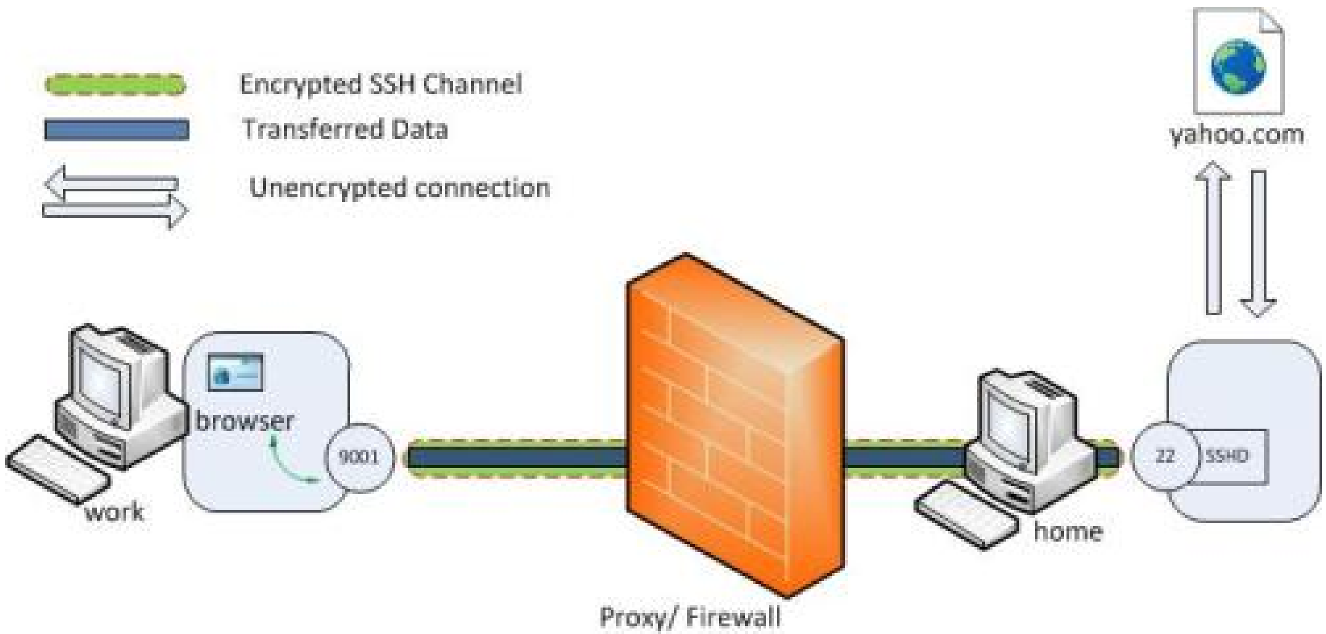
- ssh Tunneling and port forwarding



https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/

Secure Shell

- Tunneling and port forwarding



https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/

IP-LAYER

IPsec: Network Layer Security

- network-layer secrecy:
  - sending host encrypts the data in IP datagram
  - TCP and UDP segments; ICMP and SNMP messages. (one stream)
  - Lower level security for performance reasons: encapsulate many streams (gateways)
- network-layer authentication
  - Authenication: destination host can authenticate source IP address
  - Integrity
  - Confidentiality
  - Protect against replay and man-in-middle
- two principal protocols:
  - authentication header (AH) protocol
  - encapsulation security payload (ESP) protocol
- for both AH and ESP, source, destination handshake:
  - create network-layer logical channel called a security association (SA)
- each SA unidirectional.
- uniquely determined by:
  - security protocol (AH or ESP)
  - source IP address
  - 32-bit connection ID
- Transport or Tunnel Mode

IPSEC

- Provide at a lower level for performance, encapsulate many streams.
- Highly modular:
  - Access control
  - Integrity
  - Authenication
  - Replay protection
  - Confidentiality
  - Narrow or Wide streams (TCP) or (Gateways)

Authentication Header (AH) Protocol

- provides source authentication, data integrity, no confidentiality
- AH header inserted between IP header, data field.
- protocol field: 51
- intermediate routers process datagrams as usual

AH header includes:

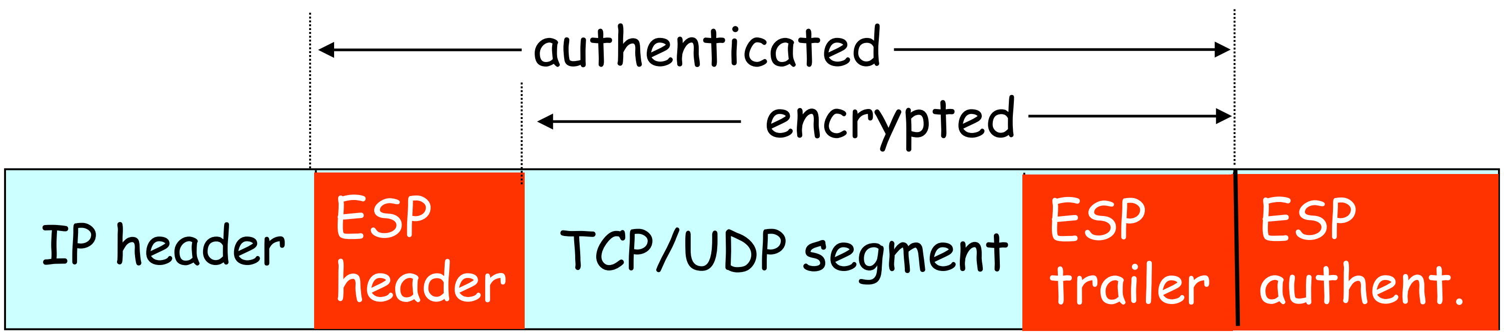
- connection identifier
- authentication data: source-signed message digest calculated over original IP datagram.
- next header field: specifies type of data (e.g., TCP, UDP, ICMP)

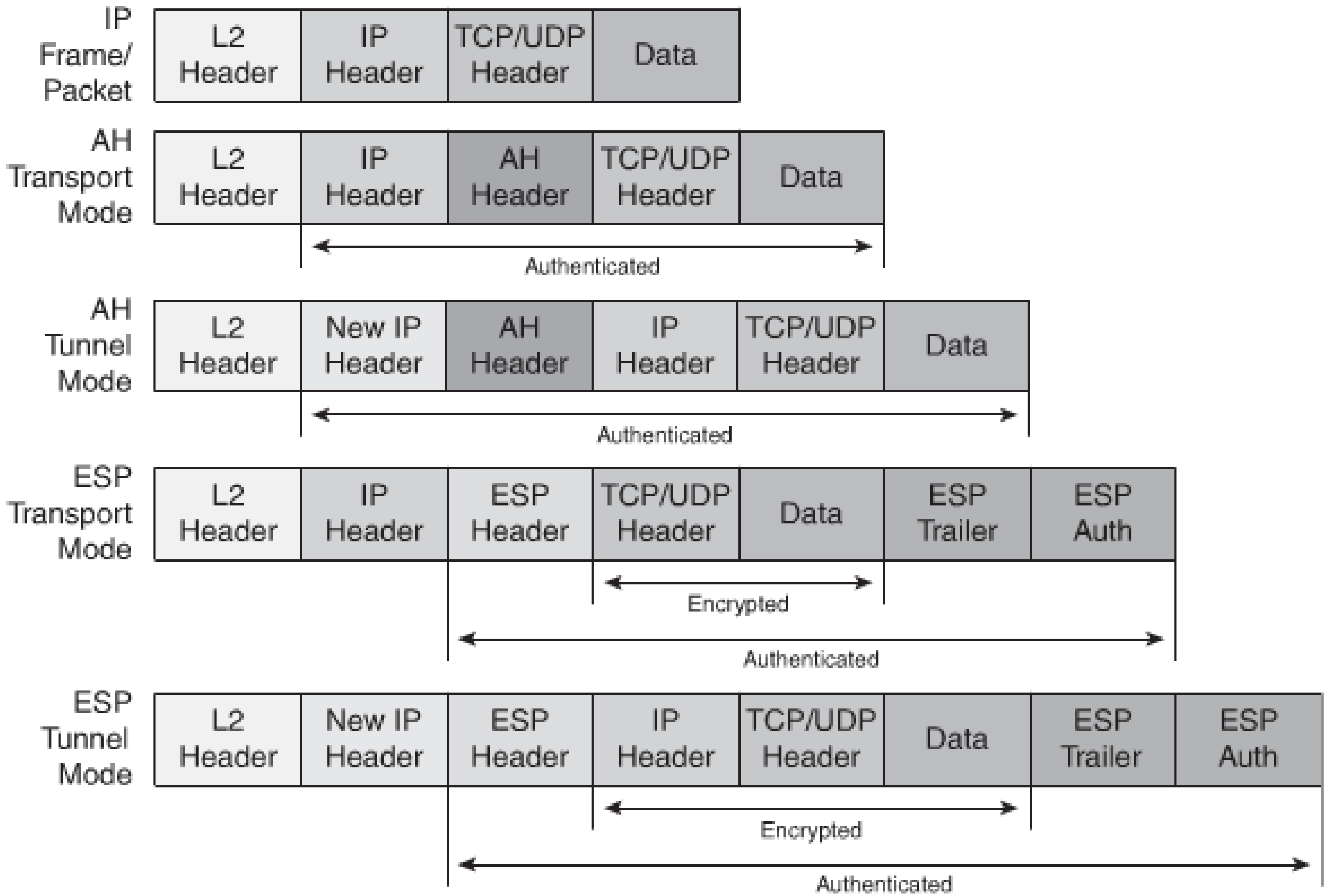
IP header	AH header	data (e.g., TCP, UDP segment)

ESP Protocol

More relevant than AH protocol

- provides secrecy, host authentication, data integrity.
- data, ESP trailer encrypted.
- next header field is in ESP trailer.
- ESP authentication field is similar to AH authentication field.
- Protocol = 50.



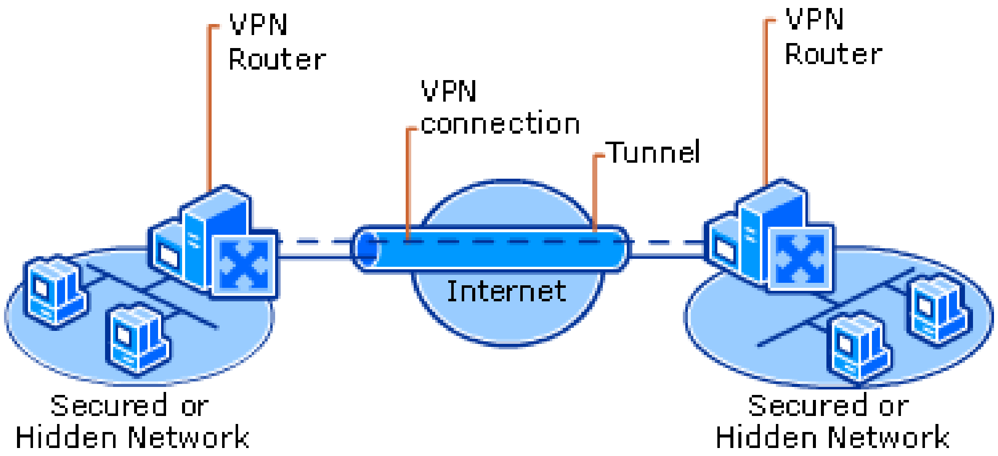


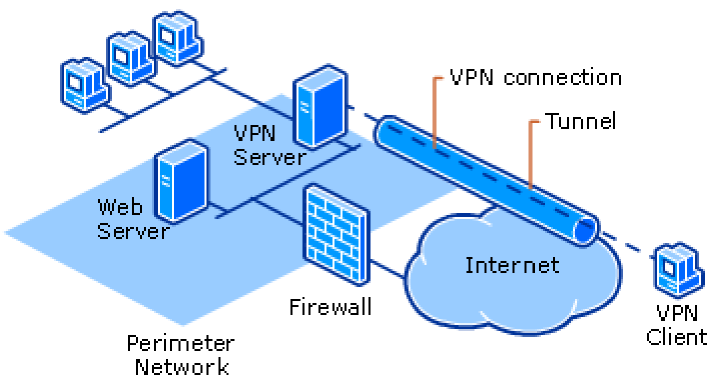
IP Frame/ Packet
L2 Header
IP Header
TCP/UDP Header
Data
AH Transport Mode
L2 Header
IP Header
AH Header
TCP/UDP Header
Data
Authenticated
AH Tunnel Mode
L2 Header
New IP Header
AH Header
IP Header
TCP/UDP Header
Data
Authenticated
ESP Transport Mode
L2 Header
IP Header
ESP Header
TCP/UDP Header
Data
ESP Trailer
ESP Auth
Encrypted
Authenticated
ESP Tunnel Mode
L2 Header
New IP Header
ESP Header
IP Header
TCP/UDP Header
Data
ESP Trailer
ESP Auth
Encrypted
Authenticated

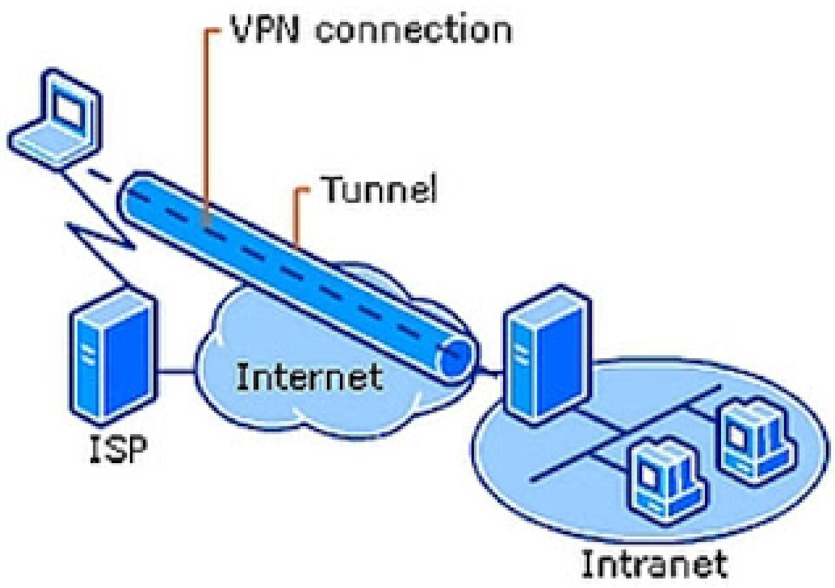
VPN

Virtual Private Network (VPN)

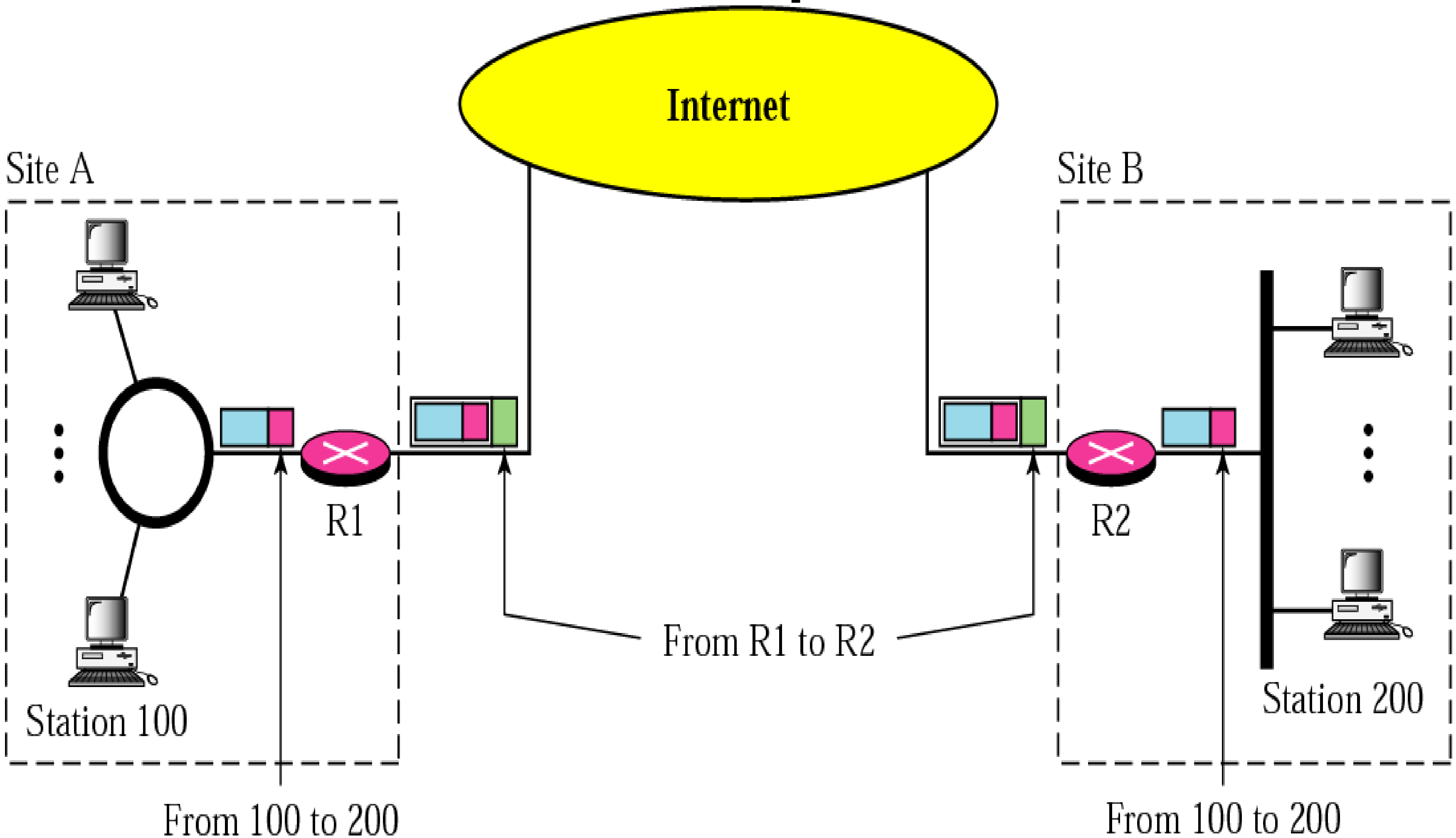
- Motivation
  - Company with multiple locations wants everything to appear as one big network
  - Workers want access to resources restricted to company internal network (e.g., hardware, restricted content, etc.)
  - Students want access to restricted material inside UBC network
  - Users want to bypass regional blocks (e.g., Netflix)
- Solution: pretend you are somewhere else


VPN
Router
VPN
connection
Tunnel
VPN
Router
Internet
Secured or
Hidden Network
Secured or
Hidden Network


VPN connection
Tunnel
VPN
Server
Web
Server
Firewall
Internet
VPN
Client
Perimeter
Network


VPN connection
Tunnel
Internet
ISP
Intranet

VPN Encapsulation



VPN Encapsulation (IP Layer)

- Virtual network interface on two end point systems
- Virtual end points establish a software association between them
  - e.g., a TCP connection
- Routing rules send traffic to virtual interface
- Virtual interface encapsulates IP message and sends it into the virtual connection
- Receiver receives this IP message and sends it through its own network

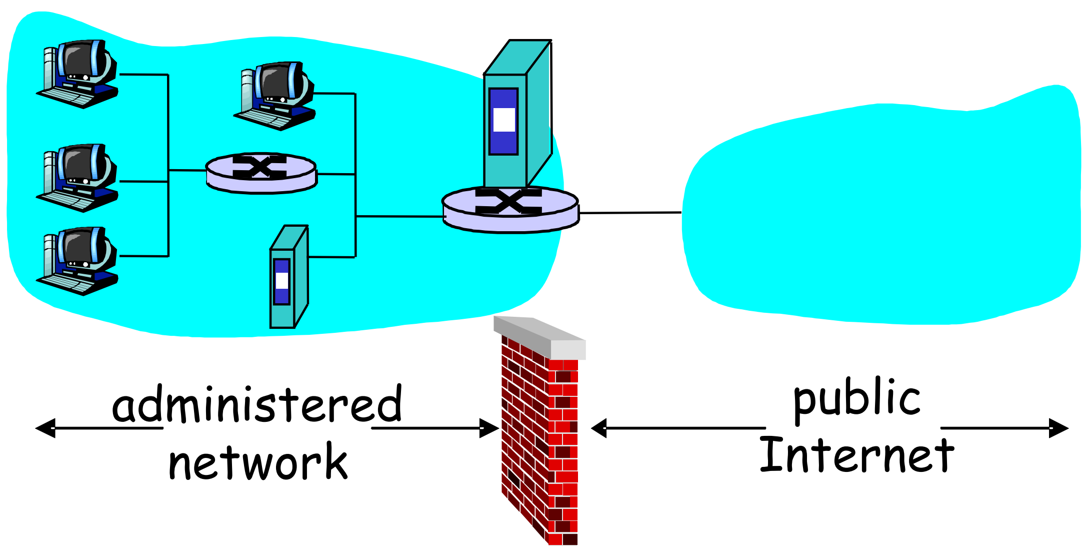
FIREWALLS

Firewalls

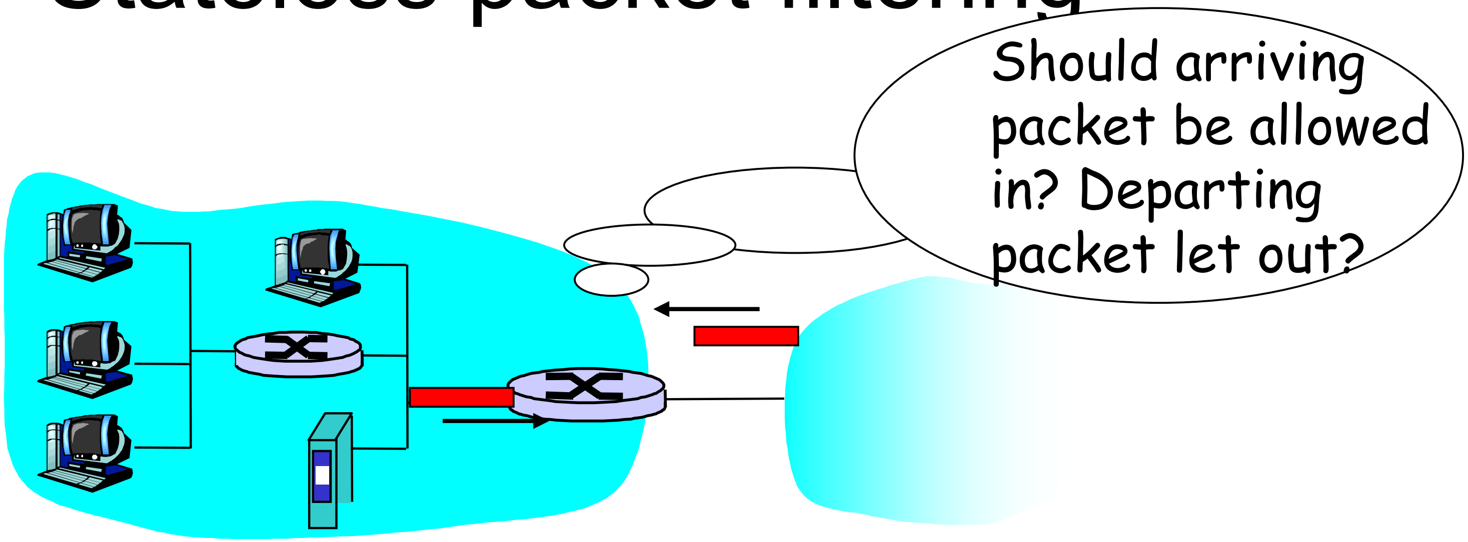
- Firewalls can evaluate messages against rules
  - In router: messages going through in each “direction”
  - In final destination: incoming and outgoing messages
  - Only allow authorized traffic
  - Close to hardware, cannot be penetrated
- prevent denial of service attacks:
  - SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connection
- Types of firewalls:
  - Packet filters
  - Stateful filters
  - Application gateways

Firewalls

- Firewalls can evaluate messages against rules
  - In router: messages going through in each “direction”
  - In final destination: incoming and outgoing messages



Stateless packet filtering



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

Stateless packet filtering: example

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

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

Firewalls

- Messages can be blocked based on:
  - Blacklists (block messages with certain rules)
  - Whitelists (only allow messages with certain rules)
  - IP address, port number, protocols, characteristics

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