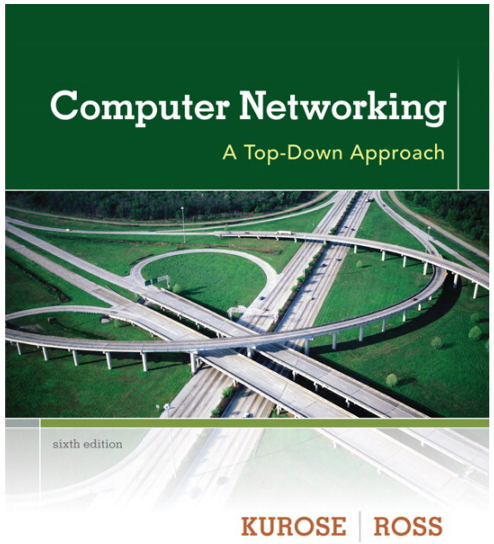
Chapter 5
Security



Bibliography:
Computer Networking: A top Down Approach, Chapter 8.

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chapter 5: Network Security

Chapter goals:

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

Chapter 5 roadmap

5.1 What is network security?

5.2 Principles of cryptography

5.3 Message integrity, authentication

5.5 Securing TCP connections: SSL

What is network security?

confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

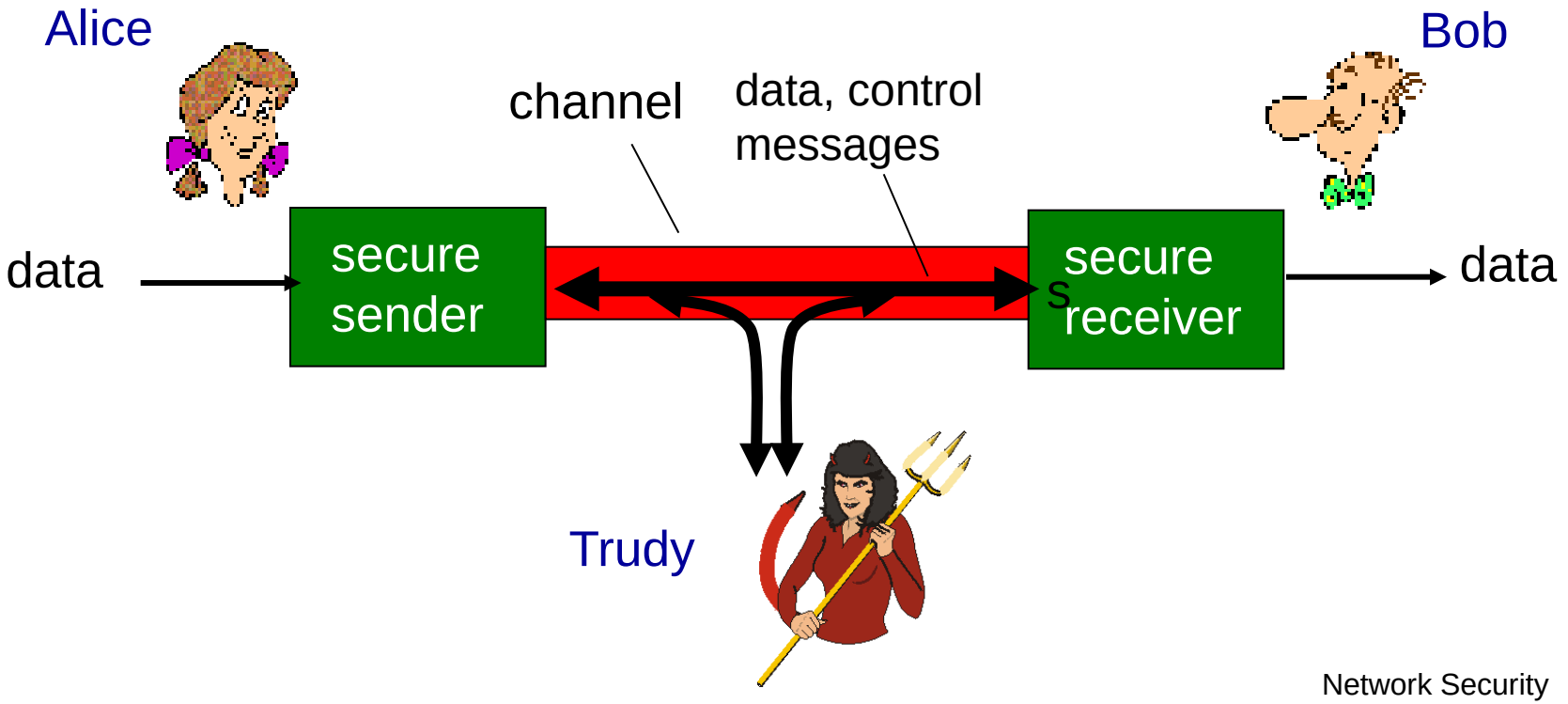
authentication: sender, receiver want to confirm identity of each other

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

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

Friends and enemies: Alice, Bob, Trudy

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



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?

Intruders : Passive vs Active

- **Passive Intruder**
  - Attempt to learn or make use of information from the system but does not affect system resources
    - Monitor transmission to obtain message contents or traffic analysis
    - Eavesdropping
    - Difficult to detect because there is no alteration of data
- **Active Intruder**
  - Attempt to alter system resources or affect their operation
    - Modification of messages in transit
    - Denial of service
    - A typical active attack is one in which an intruder impersonates one end of the conversation, or acts as a man-in-the-middle

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

A: A lot! See section 1.6

- *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)

Chapter 5 roadmap

5.1 What is network security?

5.2 Principles of cryptography

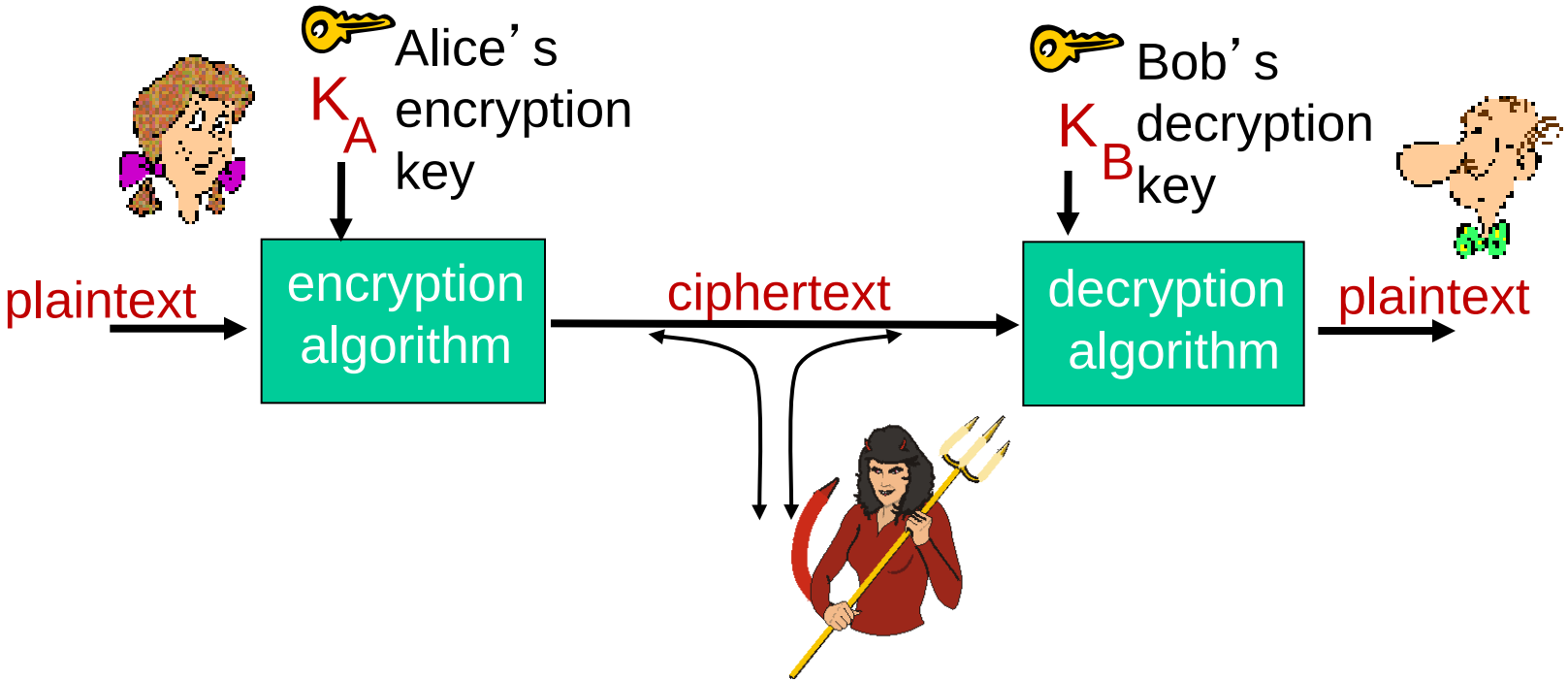
5.3 Message integrity, authentication

5.4 Securing TCP connections: SSL

What is Cryptography?

- Cryptography derived its name from a Greek word called “Kryptos” which means “Hidden Secrets”.
- Cryptography is the practice and study of hiding information. It is the Art or Science of converting a plain intelligible data into an unintelligible data and again retransforming that message into its original form.
- It provides Confidentiality, Integrity, Accuracy.

The language of cryptography



m plaintext message

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

$m = K_B(K_A(m))$

Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

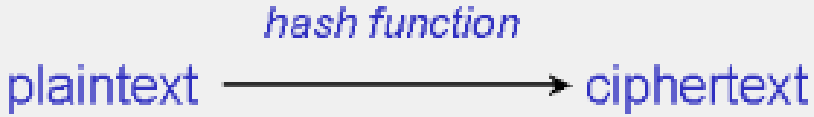
Cryptographic algorithms

- The cryptographic algorithms, based on the number of keys that are employed for encryption and decryption, can be categorized in three types of algorithms
  - Symmetric Key Cryptography (SKC):
    - Uses a single key for both encryption and decryption. Primarily used for privacy and confidentiality.
  - Public Key Cryptography (PKC):
    - Uses one key for encryption and another for decryption; also called *asymmetric encryption*. Primarily used for authentication, non-repudiation, and key exchange.
  - Hash Functions:
    - Uses a mathematical transformation to irreversibly "encrypt" information, providing a digital fingerprint. Primarily used for message integrity.

Cryptographic algorithms

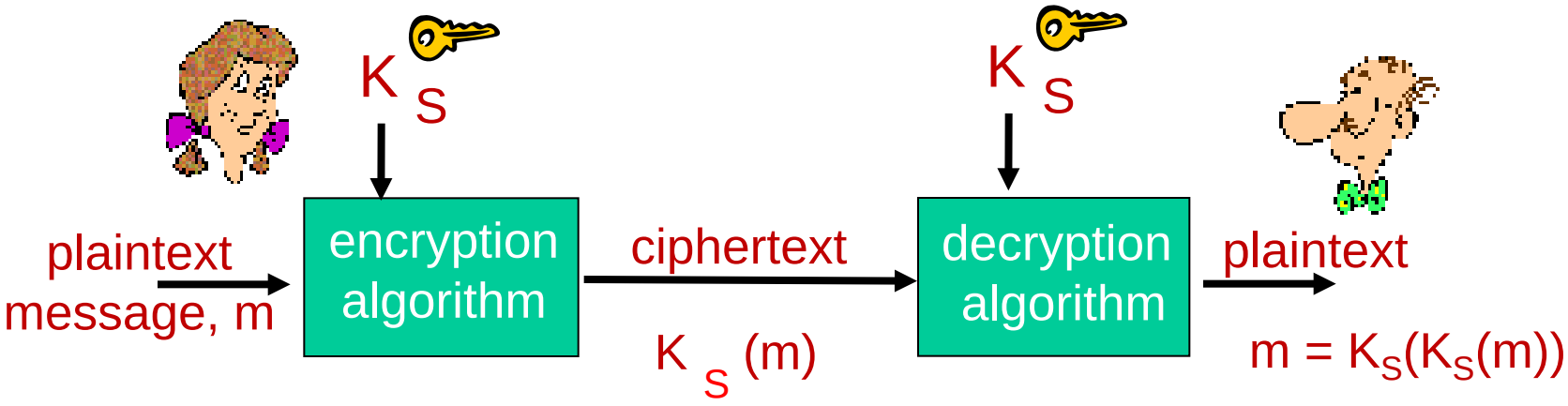
A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.

B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.



C) Hash function (one-way cryptography). Hash functions have no key since the plaintext is not recoverable from the ciphertext.

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
- Only one key is used to encrypt and decrypt messages
- The key is kept private from everyone else

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

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:  abcdefghijklmnopqrstuvwxyz
            |                       |
            v                       v
ciphertext: mnbvcxzasdfghjklpoiuytrewq
```

e.g.: **Plaintext: bob. i love you. alice**

ciphertext: nkn. s gktc wky. mgsbc

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

A more sophisticated encryption approach

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- 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 subsitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern

- key need not be just n-bit pattern

Two schemes of symmetric key cryptography

- Symmetric key cryptography schemes are generally categorized as being either ***stream ciphers*** **or** ***block ciphers***
  - **Stream ciphers** operate on a single bit (byte or computer word) at a time
  - A **block cipher** is so-called because the text message is fragments into blocks of identical size, and then the scheme encrypts one block of data at a time using the same key on each block

Symmetric key crypto: DES

DES: Data Encryption Standard

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

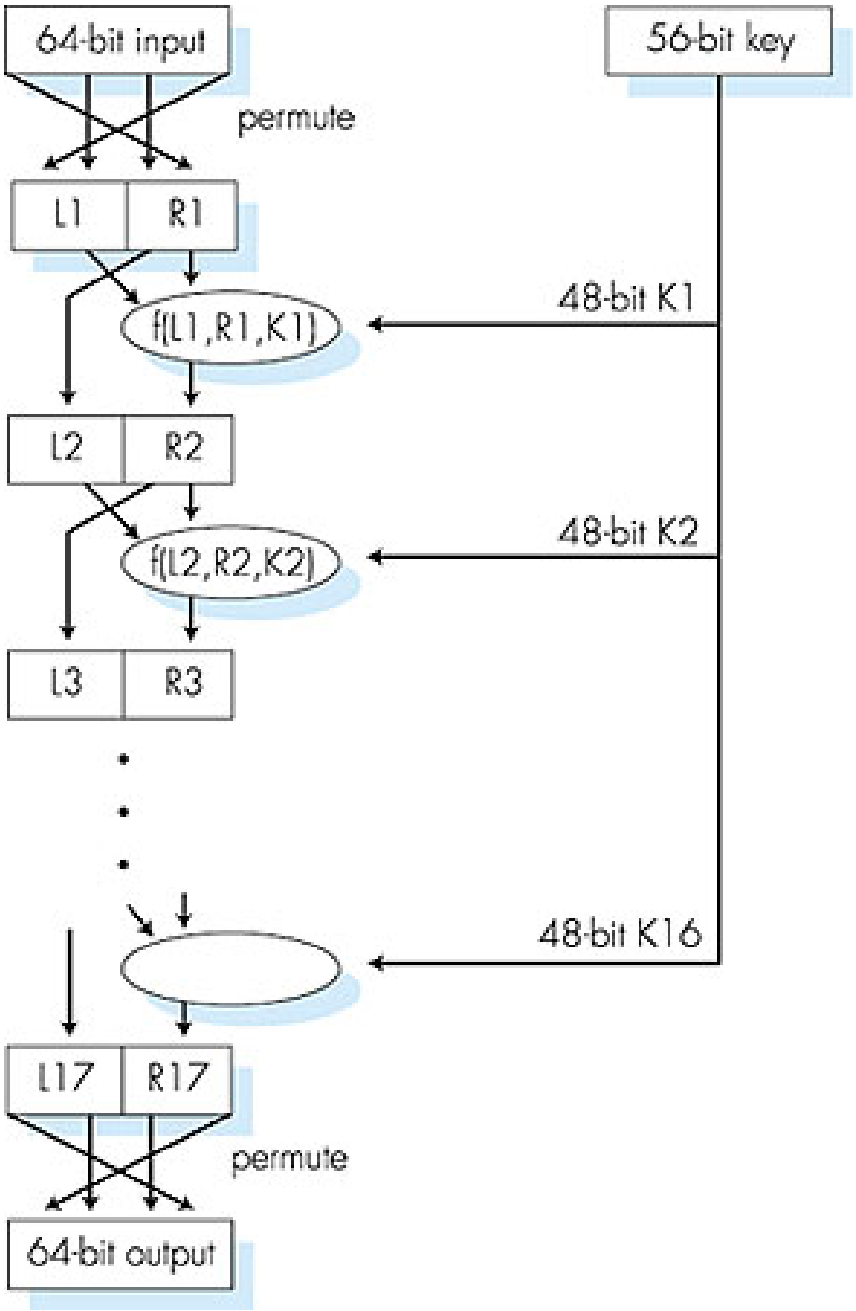
Symmetric key crypto: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



AES: Advanced Encryption Standard

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

Public Key Cryptography

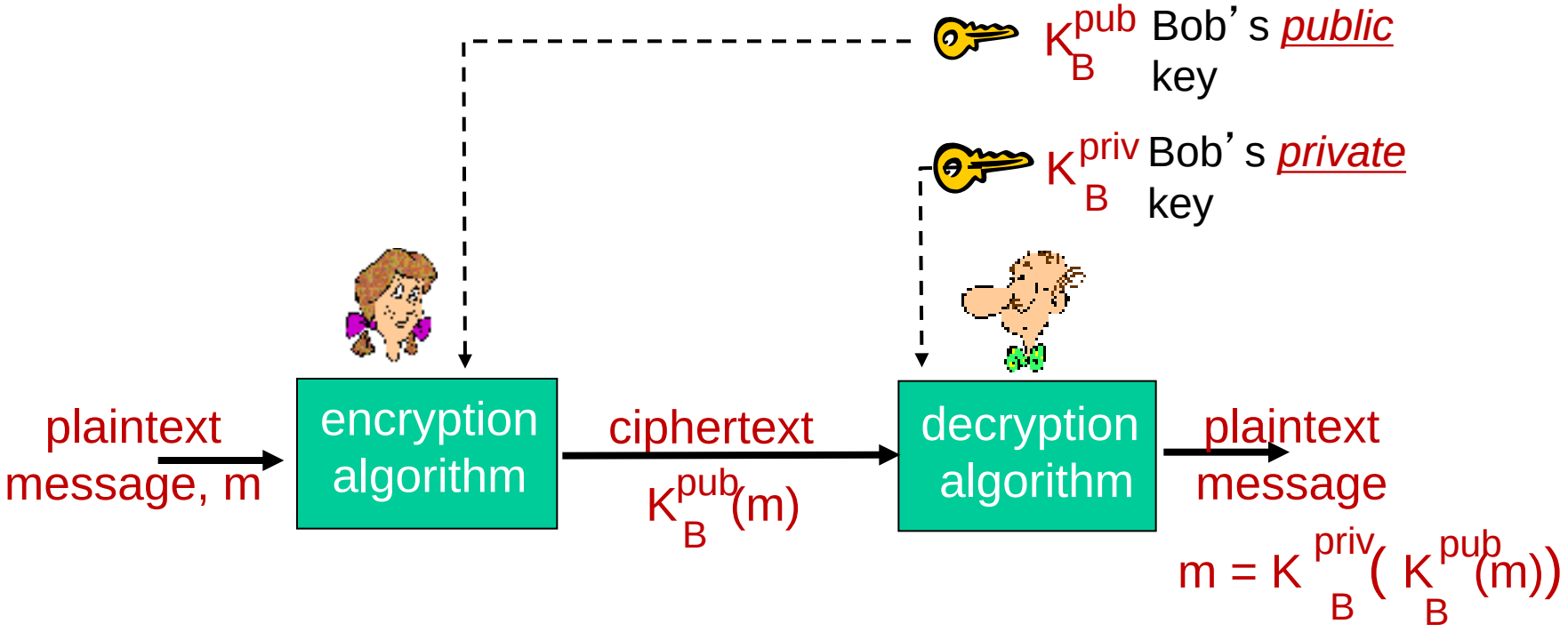
symmetric key crypto

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

public key crypto

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

Public key cryptography



Public key encryption algorithms

requirements:

1. need $K_B^{pub}(\cdot)$ and $K_B^{priv}(\cdot)$ such that

$$K_B^{priv}(K_B^{pub}(m)) = m$$

2. given public key K_B^{pub}, it should be impossible to compute private key K_B^{priv}

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$

  $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

  $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers *p*, *q*. (e.g., 1024 bits each)
2. compute *n* = *pq*, *z* = *(p-1)(q-1)*
3. choose *e* (with *e<n*) that has no common factors with z (e, *z* are "relatively prime").
4. choose *d* such that *ed-1* is exactly divisible by *z*. (in other words: *ed* mod *z* = *1*).
5. *public* key is $\underbrace{(n,e)}_{K_B^{pub}}$. *private* key is $\underbrace{(n,d)}_{K_B^{priv}}$.

RSA: encryption, decryption

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

1. to encrypt message m ($<n$), compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c, compute

$$m = c^d \bmod n$$

magic happens!

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

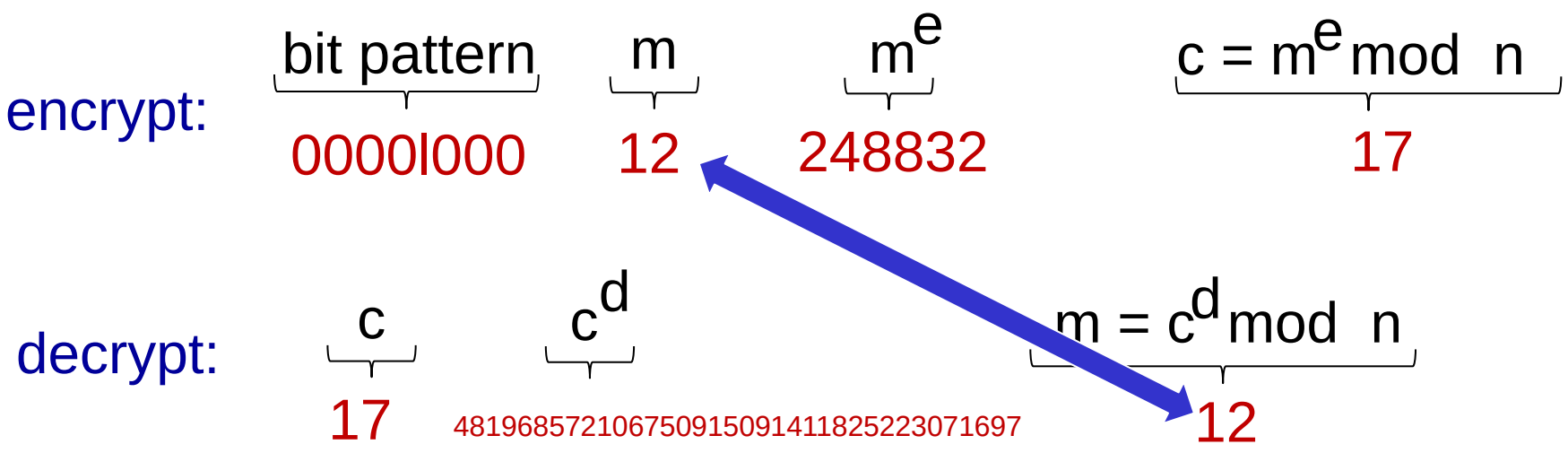
RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.



Why does RSA work?

- must show that $c^d \bmod n = m$ where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where n= pq and z = (p-1)(q-1)
- thus,

$$c^d \bmod n = (m^e \bmod n)^d \bmod n$$
$$= m^{ed} \bmod n$$
$$= m^{(ed \bmod z)} \bmod n$$
$$= m^1 \bmod n$$
$$= m$$

RSA: another important property

The following property will be *very* useful later:

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

result is the same!

Why $K_B^{priv}(K_B^{pub}(m)) = m = K_B^{pub}(K_B^{priv}(m))$?

follows directly from modular arithmetic:

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

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

RSA in practice: session keys

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

session key, K_S

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

Chapter 5 roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 Message integrity, authentication

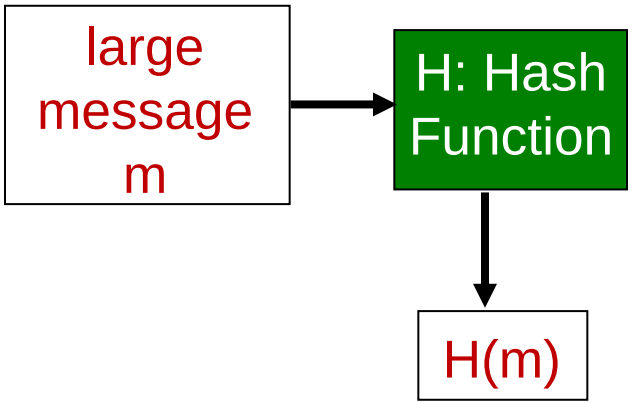
8.4 Securing TCP connections: SSL

Message integrity

- Cryptography ensures the authenticity and integrity of messages received through the network
  - the message content has not been altered
  - origin of the message is what you think it
  - message has not been reproduced
  - Sequence message has remained
- However, encrypting the entire message can be costly (in terms of computational and storage)
  - Efficient solution: encrypt a block of small bits obtained from the document (message digest) by a function (hash function)

Hash function H()

- Hash function H () takes an input message m of arbitrary size, apply hash function H to m, and produces fixed size message digest (fingerprint), H(m).



- Hash function properties:
  - easy to calculate (computationally)
  - given message digest x, computationally infeasible to find m such that x = H(m)
- Collision resistance: it must be computationally difficult to produce m and m‘ such that

$$H(m) = H(m')$$

  - it must seem random output
  - Similar idea to the checksum or CRC

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

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

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

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

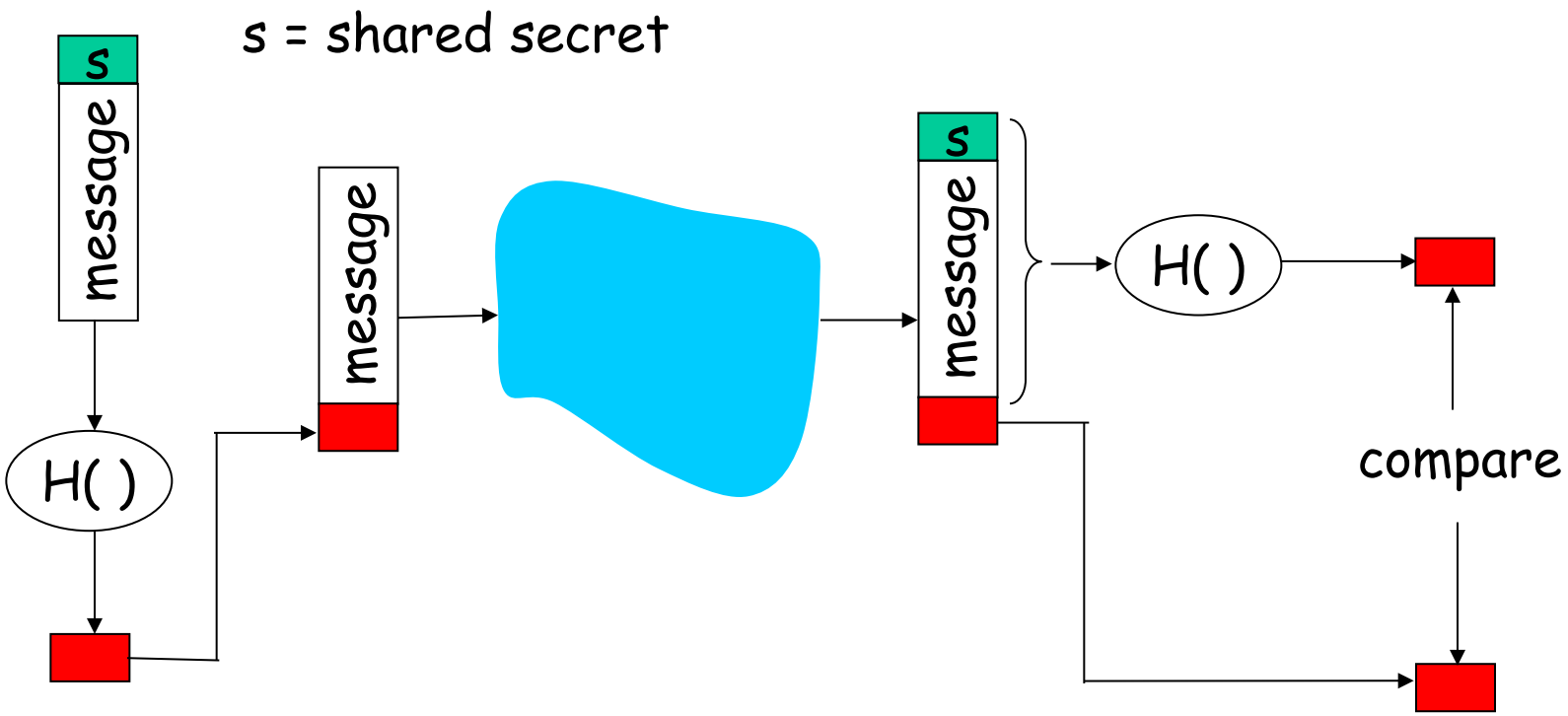
different messages but identical checksums!

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

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
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest
- Ensures the integrity of the message but does not provide authentication

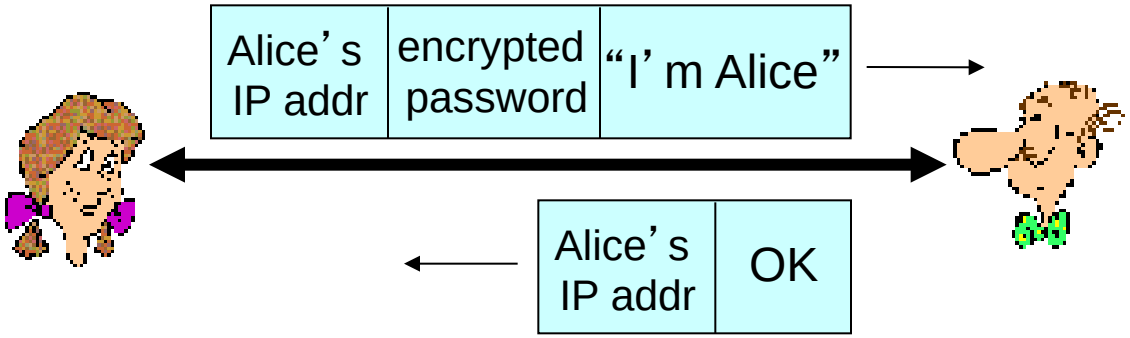
Message Authentication using HMAC



- Hashed Message Authentication Code (HMAC)
  - authenticates the sender
  - verifies the integrity of messages
  - no encryption!
  - also called "hash keys"
  - The shared secret is not sent!!

Authentication: playback attack

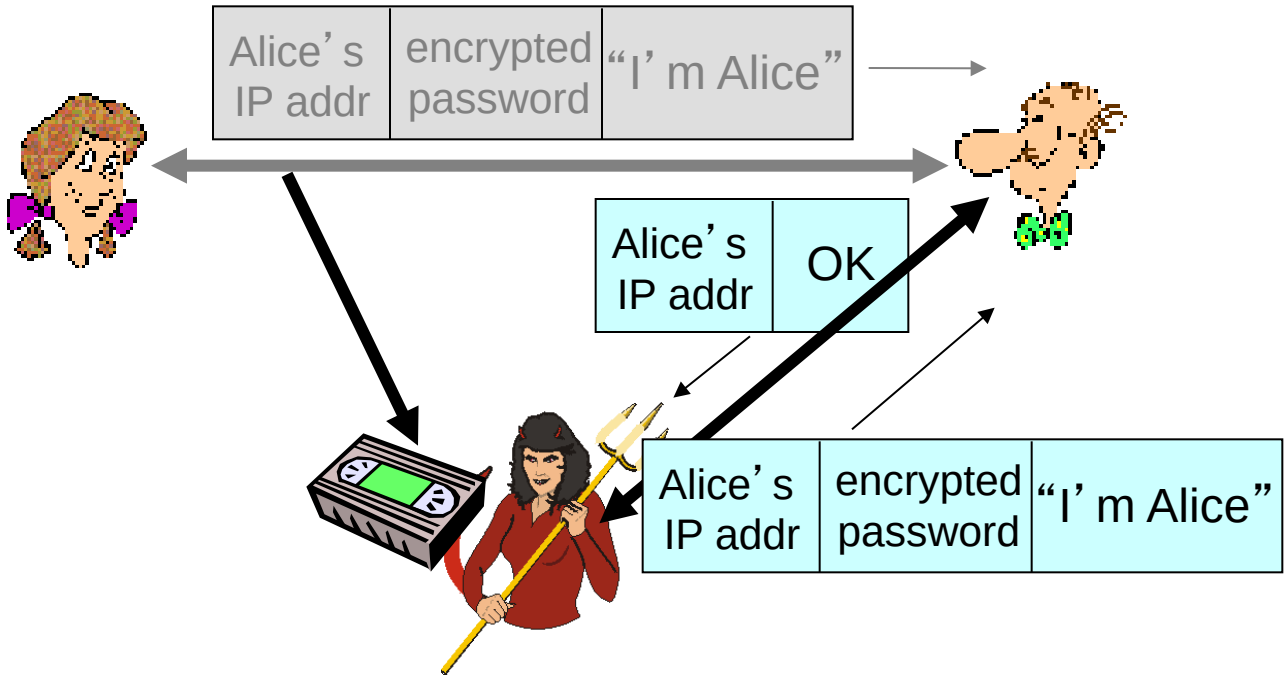
Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: playback attack

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



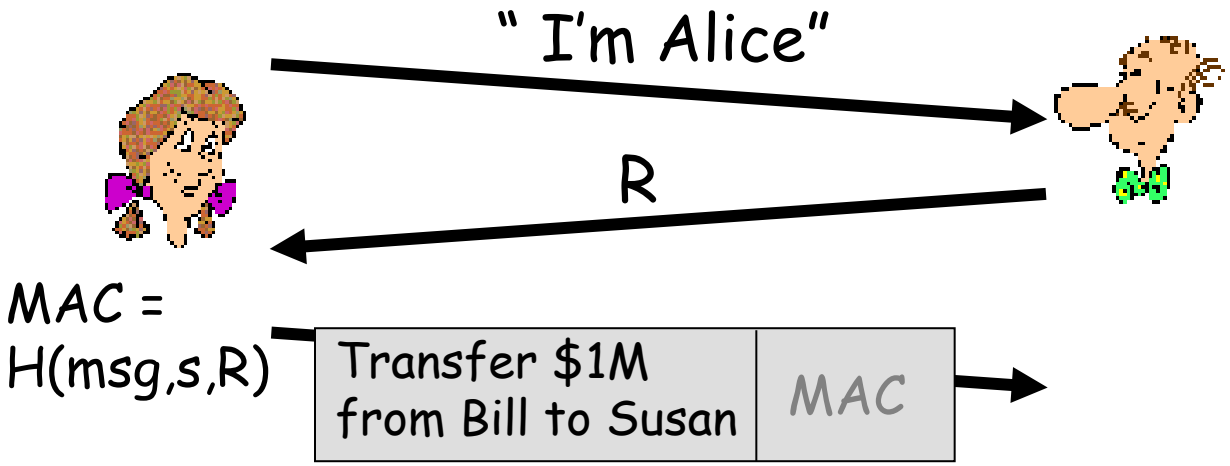
record and playback works!

Solution to playback attack

Goal: avoid playback attack

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

To prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



The MAC is generated as Hash of the message, the secret key and number R

Digital signatures

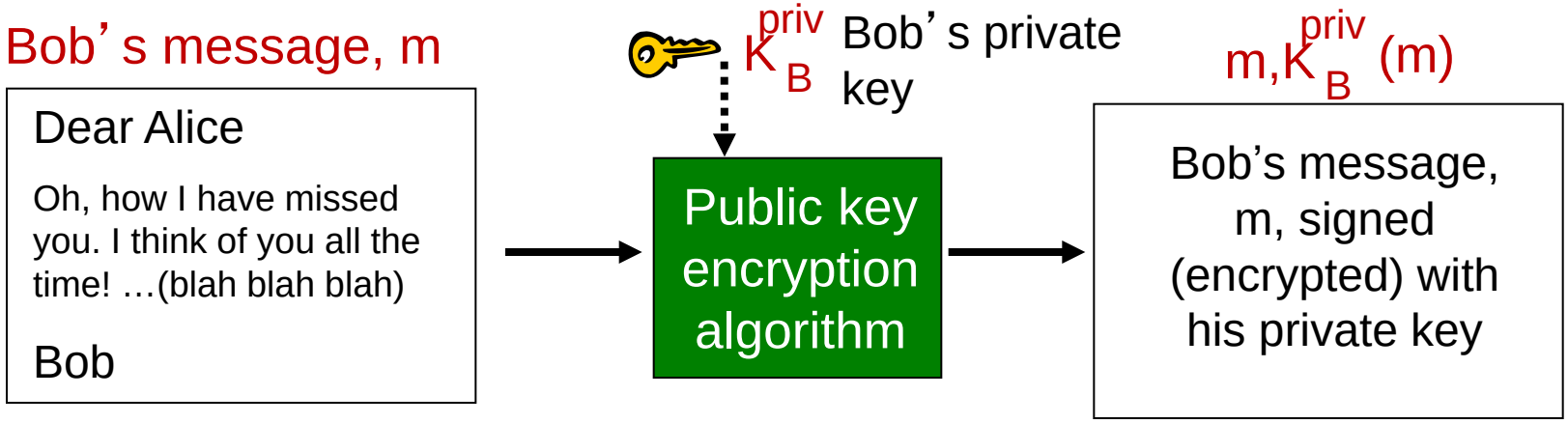
cryptographic technique analogous to hand-written signatures:

- 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
- Similar goal than MAC (Message Authentication Code), but using public key cryptography

Digital signatures

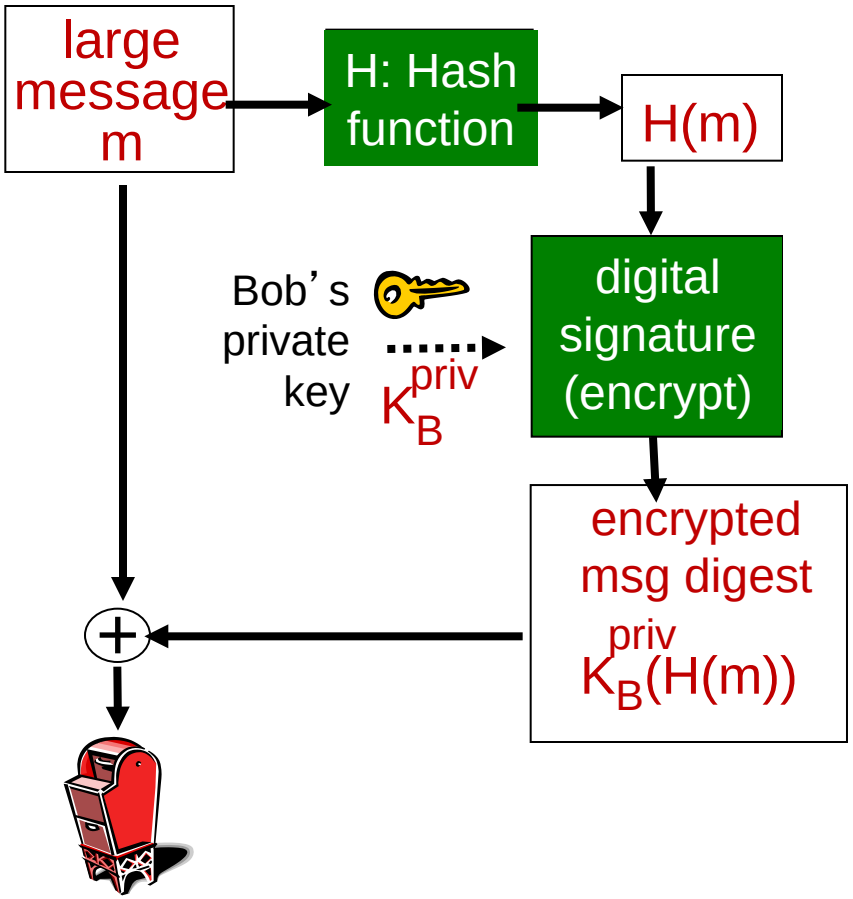
simple digital signature for message m:

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

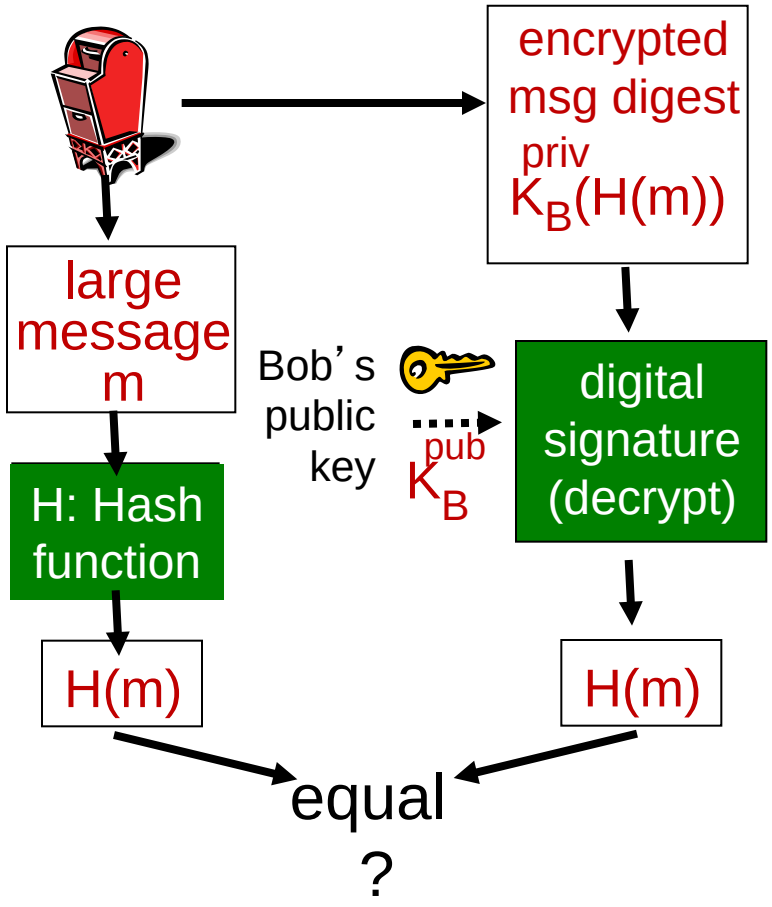


Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^{priv}(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^{pub} to $K_B^{priv}(m)$ then checks $K_B^{pub}(K_B^{priv}(m)) = m$.
- If $K_B^{pub}(K_B^{priv}(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

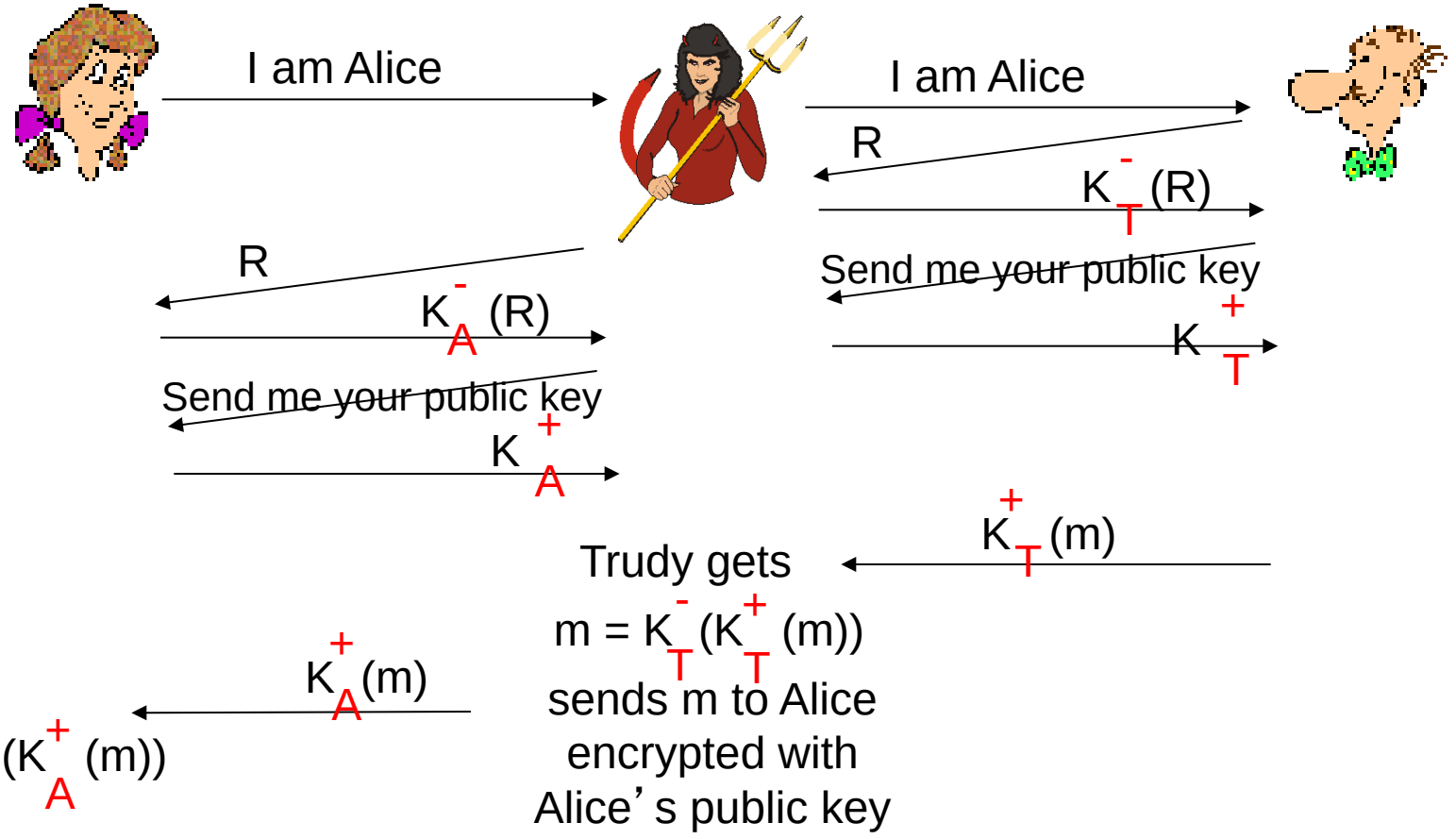
- ✓ Bob signed m
- ✓ no one else signed m
- ✓ Bob signed m and not m '

non-repudiation:

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

man (or woman) in the middle attack

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

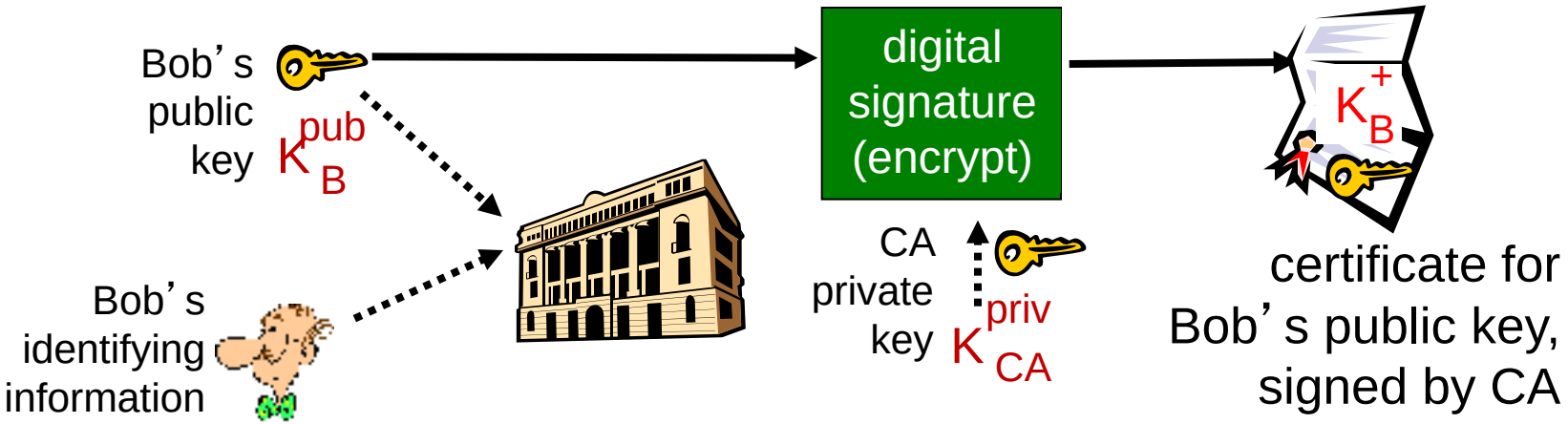


Public-key certification

- motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
  - Bob doesn't even like pepperoni

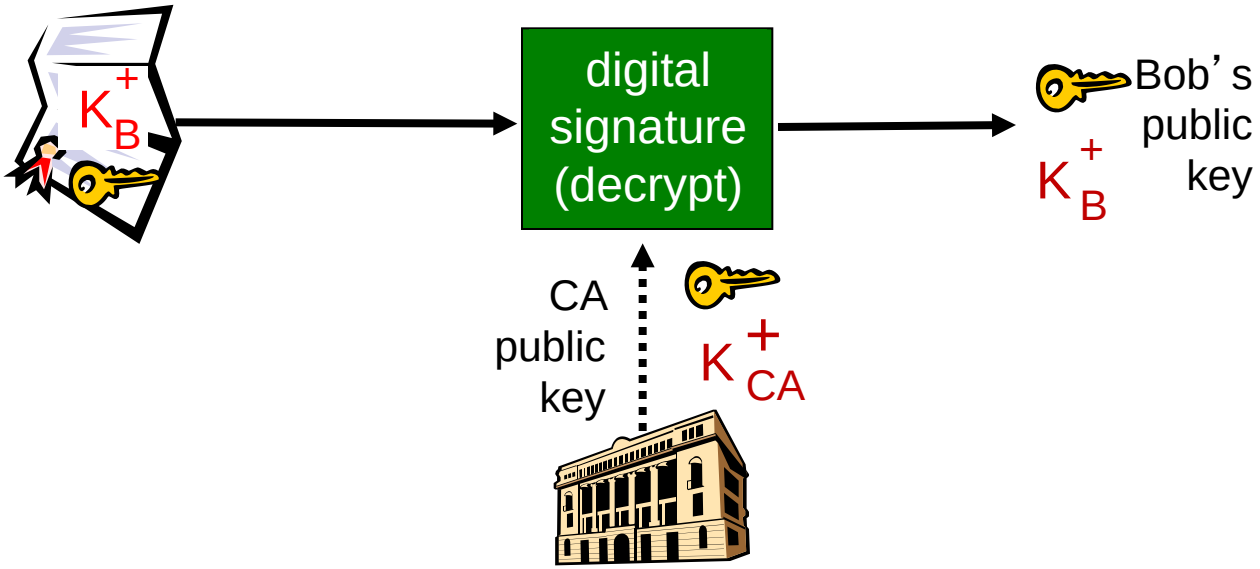
Certification authorities

- *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 Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



Certification authorities

- Certificate authority business is fragmented, with national or regional providers, because many uses of digital certificates, such as for legally binding digital signatures, are linked to local law, regulations, and accreditation schemes for certificate authorities
- X.509 standard (RFC 2459)
- The certificate contains:
  - name of the issuer (the certificate authority) , entity name, address, domain name, etc., public key of the entity, digital signature (signed with the private key the issuer)
- Public Key Infrastructure (PKI) is the most commonly encountered schemes used to implement https on the world-wide-web

Chapter 5 roadmap

5.1 What is network security?

5.2 Principles of cryptography

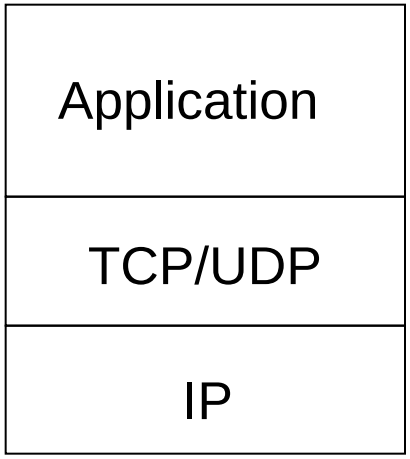
5.3 Message integrity, *authentication*

5.4 Securing TCP connections: SSL

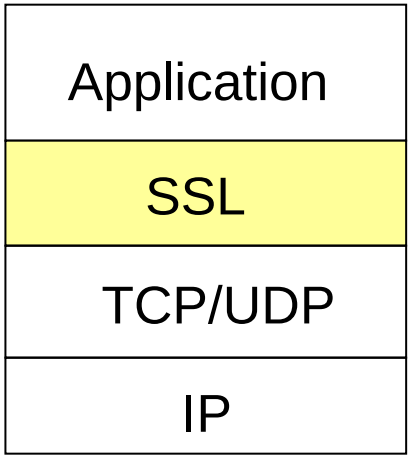
SSL: Secure Sockets Layer

- 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*
- 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 and UDP applications
  - secure socket interface

SSL and TCP/IP



normal application



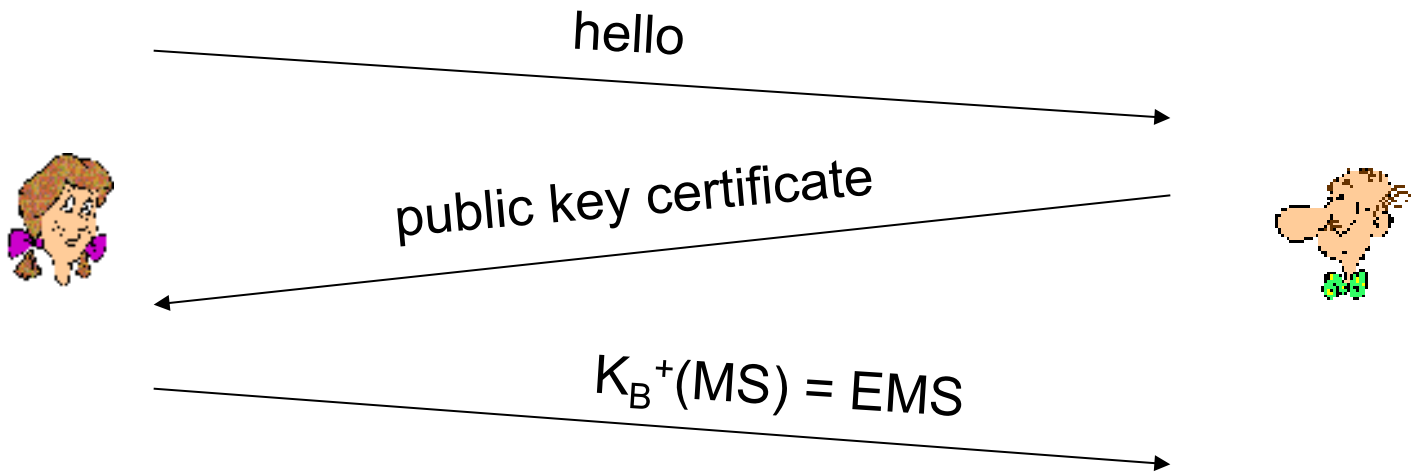
application with SSL

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

Toy SSL: a simple secure channel

- *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

Toy: a simple handshake



MS: master secret

EMS: encrypted master secret

Toy: 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
  - K_c and K_s : to encrypt data
  - M_c and M_s : to provide message integrity
- keys derived from key derivation function (KDF)
  - takes master secret and (possibly) some additional random data and creates the keys

Toy: 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

Toy: sequence numbers

- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = MAC(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)

Toy SSL: summary

bob.com

hello

certificate, nonce

$K_B^+(MS) = EMS$

type 0, seq 1, data

type 0, seq 2, data

type 0, seq 1, data

type 0, seq 3, data

type 1, seq 4, close

type 1, seq 2, close

encrypted

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

Toy SSL isn't complete

- 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