The University of Melbourne
School of Computing and Information Systems

# COMP30023
## Computer Systems

Semester 1, 2017

# Security

# Introduction

- Message Confidentiality: secret key encryption; public key encryption
- Message Authentication: certificates
- Security in practice: security in application, transport, network, link layers, firewalls, intrusion detection systems

# Definitions

- Confidentiality: only the sender and intended receiver should "understand" the message contents
- **Authentication:** the sender and receiver want to confirm the identity of each other
- Message integrity: the sender and receiver want to ensure that the message is not altered (in transit, or afterwards) without detection
- **Access and availability:** services must be accessible and available to users

# Security Issues

- **Eavesdrop:** intercept messages
- Actively insert messages into connection ( message corruption )
- **Impersonation:** can fake (spoof) source address in packet (or any field in packet)
- Hijacking: "take over" ongoing connection by removing sender or receiver
- **Denial of service:** prevent service from being used by others (e.g., by overloading resources)

# Cryptography

*"Encryption works. Properly implemented strong crypto systems are one of the few things that you can rely on. Unfortunately, endpoint security is so terrifically weak that NSA can frequently find ways around it."*
(Edward Snowden)

- Try to make something work (get the right answer, maintain privacy, etc.) even when a powerful adversary is deliberately trying to break it.
- Think evil and do good!
- The attacker's job is to find the attack you did not expect

# Cryptography

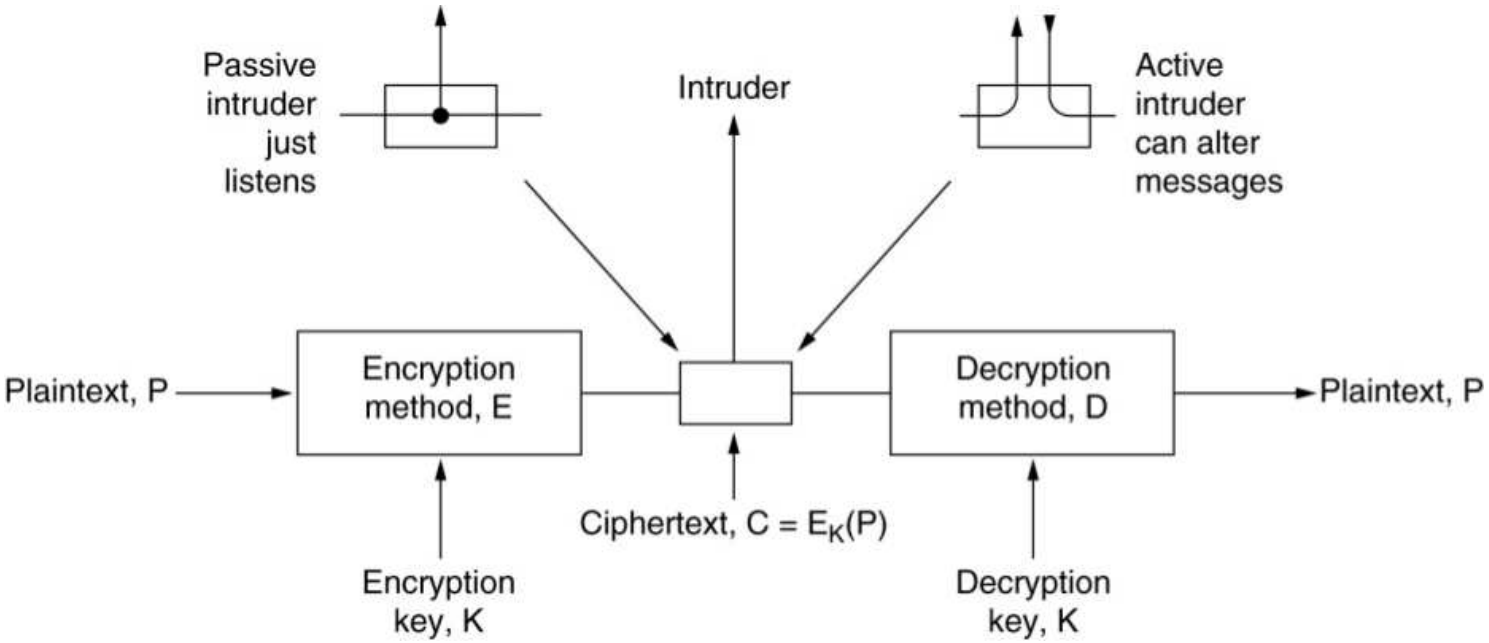
**Encrypting:**

- sending messages that are secret from everyone but the intended recipient
- sender has to hide the message for sending, so nobody else can understand it
- the ciphertext message then is sent over the network

**Decrypting:**

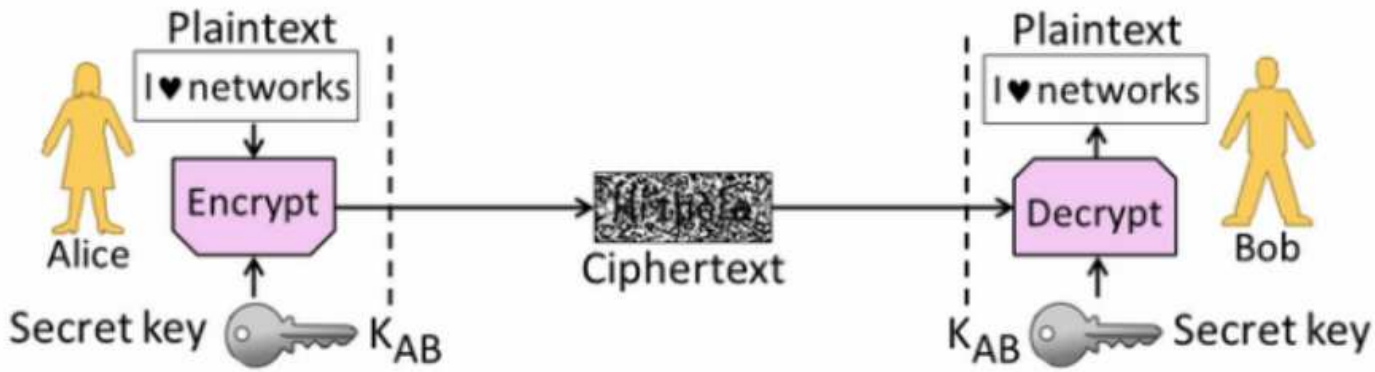
- receiver has to "un-hide" and recover the message
- the reverse process of encrypting

# Cryptography: Symmetric Key




e.g. substitution, transposition, block (AES)

# Secret-key Cryptography



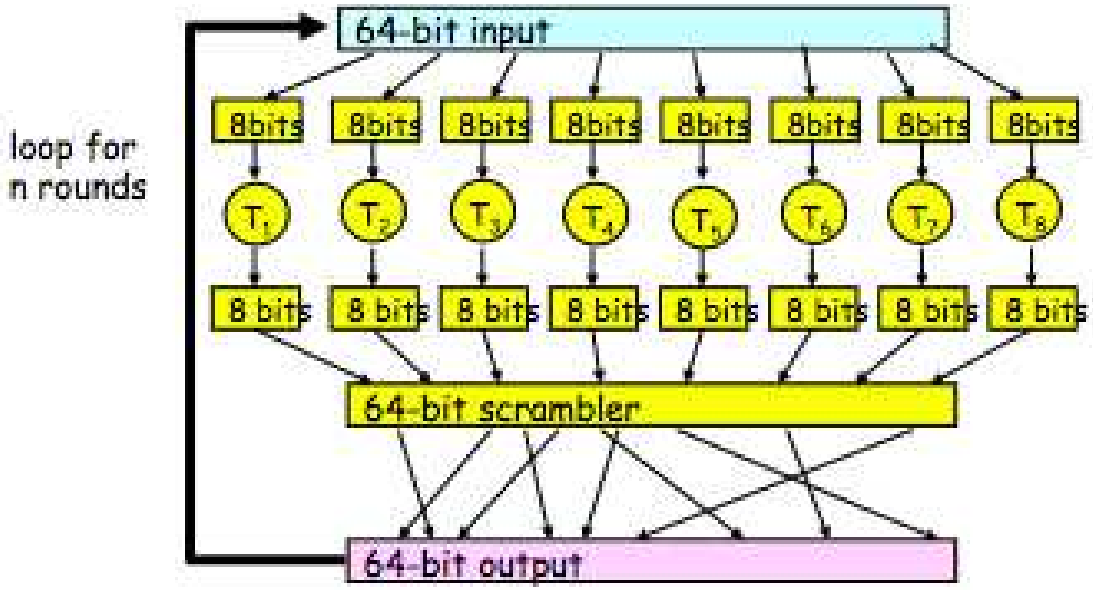

Both the sender and the receiver have to agree on the secret key in advance.

# AES: Advanced Encryption Standard *

The full design must be public.

- The algorithm must be a symmetric block cipher (see next slide)
- Key lengths of 128, 192, and 256 bits are supported.
- The algorithm must be public or licensed on nondiscriminatory terms.
- Brute force decryption (try each key) taking 1 sec on original DES, takes 149 trillion years for AES.

# AES: Block-cipher *



# Example . . .

# Public-key Cryptography

. . . or . . . *How to send secret messages to people you have not met*.

**The receiver generates two keys:**

- a public key E (for encrypting), and
- a private key D (for decrypting)

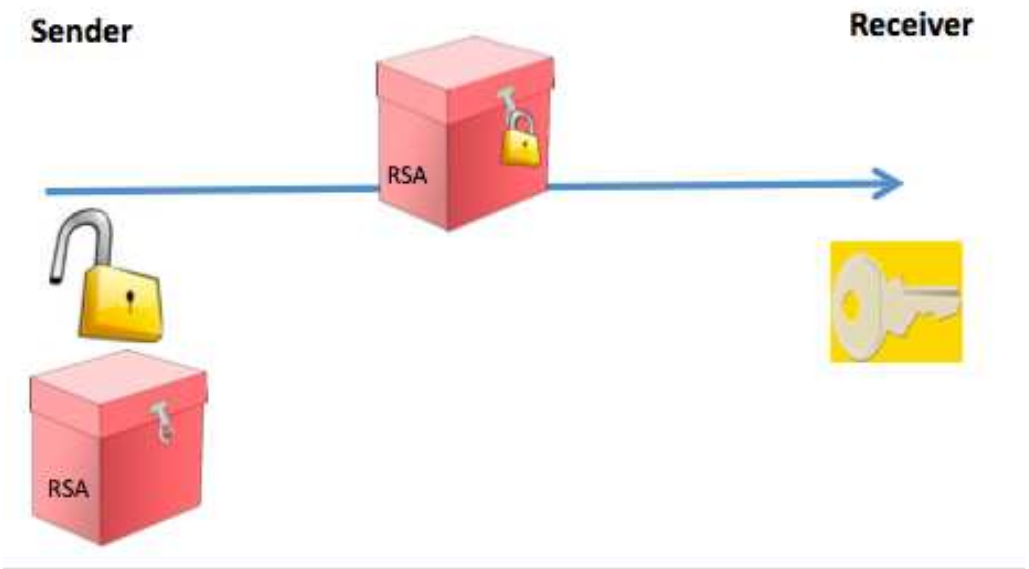
**Publicises the public key E**

- people use this for encrypting messages

**Keeps the private key D secret**

- receiver uses this for decrypting messages

# Public-key Cryptography: An example




Lots of people sending to the same receiver (e.g. in electronic voting, everyone sends their vote to the Electoral Commission)

*Encrypted with the Commission's public key.*

# Public-key Cryptography

**Sender**:

- generates a secret key,
- encrypts a message with the secret key,
- encrypts the secret key with the receiver's public key, and
- sends the encrypted message and the encrypted key

**Receiver**:

- uses her private key to decrypt the secret key
- uses the secret key to decrypt the message

# Public-key Encryption

This is (almost but not quite) how SSL/TLS works, when you get a comforting little lock at the bottom of your screen before you send your credit card number.

# RSA (Rivest, Shamir, Adleman) Algorithm *

Goal:

**D**( E(P) ) = P

It is exceedingly difficult to deduce **D** from E

E cannot be broken by a chosen plaintext attack

# RSA (Rivest, Shamir, Adleman) Algorithm *

| Plaintext (P) | | | Ciphertext (C) | | After decryption | |
|---|---|---|---|---|---|---|
| Symbolic | Numeric | $P^3$ | $P^3$ (mod 33) | $C^7$ | $C^7$ (mod 33) | Symbolic |
| S | 19 | 6859 | 28 | 13492928512 | 19 | S |
| U | 21 | 9261 | 21 | 1801088541 | 21 | U |
| Z | 26 | 17576 | 20 | 1280000000 | 26 | Z |
| A | 01 | 1 | 1 | 1 | 01 | A |
| N | 14 | 2744 | 5 | 78125 | 14 | N |
| N | 14 | 2744 | 5 | 78125 | 14 | N |
| E | 05 | 125 | 26 | 8031810176 | 05 | E |
| | | | Sender's computation | | Receiver's computation | |

(left) Encryption: $C = P^3 mod 33$
(right) Decryption: $P = C^7 mod 33$
parameters: p=3 q =11 n=33 z=20 d=7 e=3

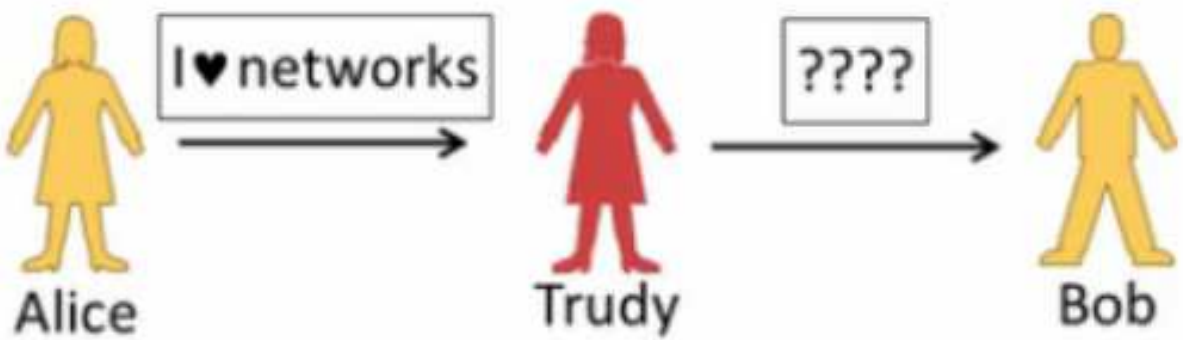
# Example

# Summary

| Property | Symmetric | Public Key |
|---|---|---|
| Key Distribution | Hard – share secret per pair of users | Easier – publish public key per user |
| Runtime Performance | Fast – good for high data rate | Slow – few, small, messages |

# Digital Signatures

. . . or . . . *How to check who sent the message*.



# Digital Signatures

A digital signature is mathematical link between a particular message and a particular public key.

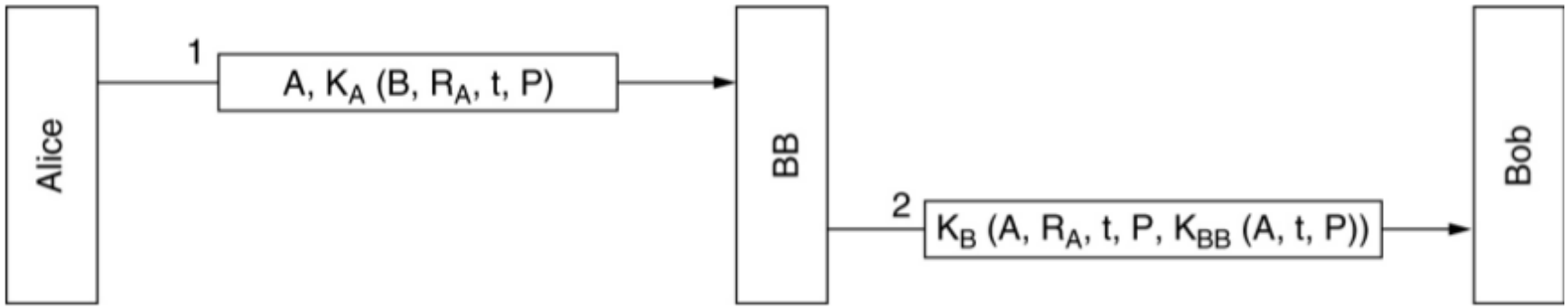
Signature: Sign(message, private_key) . A string of bits that Alice appends to her message.

**Verify(message, signature, public_key)**. Allows Bob to check that Alice's public key was used to sign.

Without Alice's public key, you cannot forge/modify/sign a different message if you try ie., verification will fail.

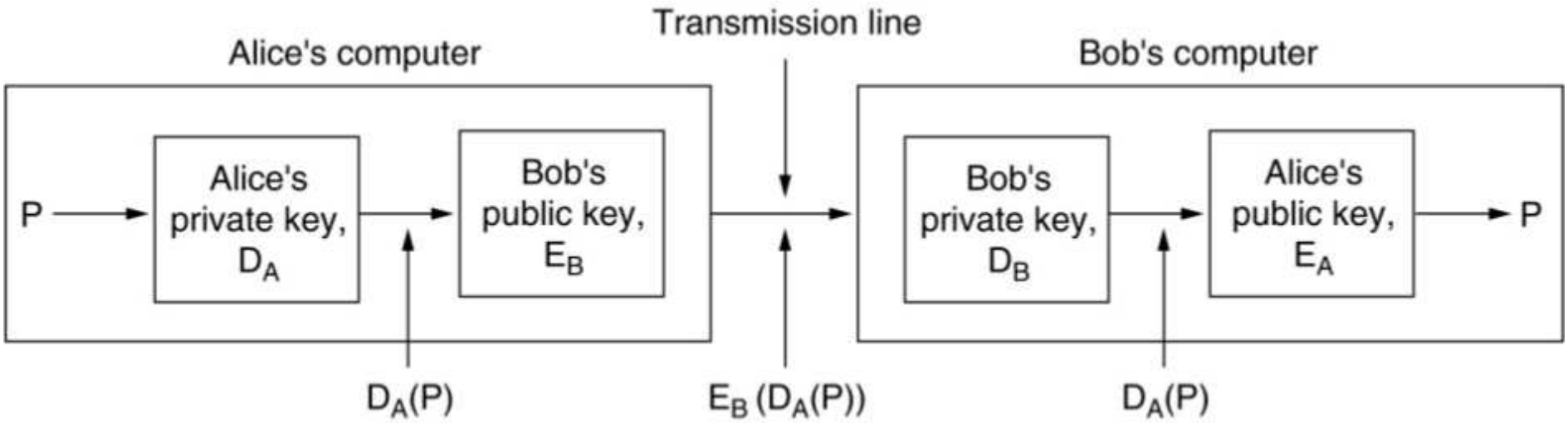
# Symmetric-key Signatures

Digital signatures with Big Brother (an intermediary)



# Public-key Signatures

Digital signatures using public-key cryptography.
(. . . no need to trust Big Brother)



# Using Digital Signatures

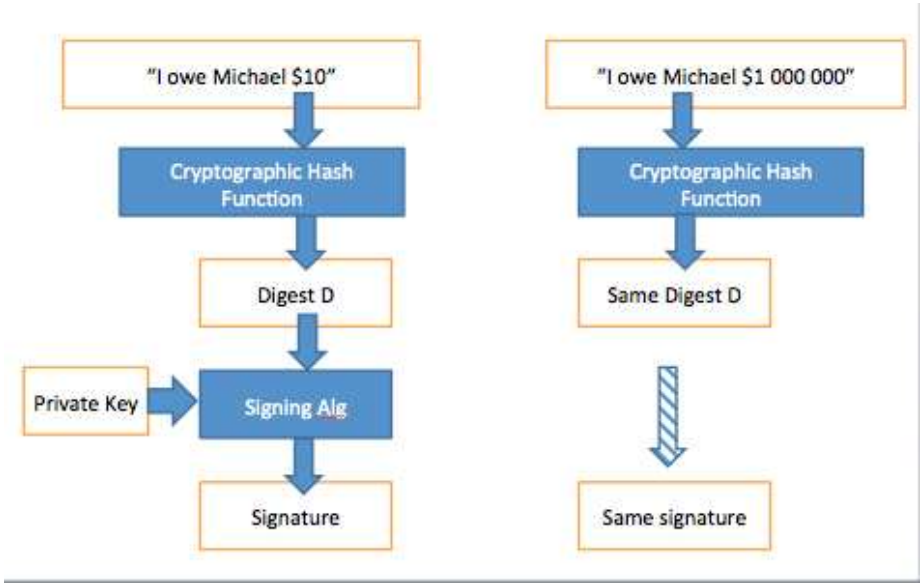
- Most digital signature algorithms (e.g. RSA, DSA) hash a message before signing.
- A hash algorithm takes a (possibly long) message, and produces a fixed-length digest (at least 160 bits).
- For crypto hashes, it should be infeasible to find two messages that hash to the same digest (this is called a "collision").

# Public-key Message Digest

# Example

When Alice obtains Bob's public key (from web site, e-mail, text message), how does she know it is Bob's public key, not Trudy's?

Solution: trusted certification authority (CA)



# Using Certificates

A certificate is just a special kind of signed message, where:

- the message says "XYZ " is the public key of Person A plus some other data, e.g. a date.
- the signer is someone (whose public key) you already know

A typical web browser has lots of certificate authorities' public keys installed when it ships. (Sometimes they are used in chains)

You can check them yourself e.g. using the *examine certificate* option in Windows or by using *openssl verify* on nutmeg/dimefox

# Management of Public keys: Certificates

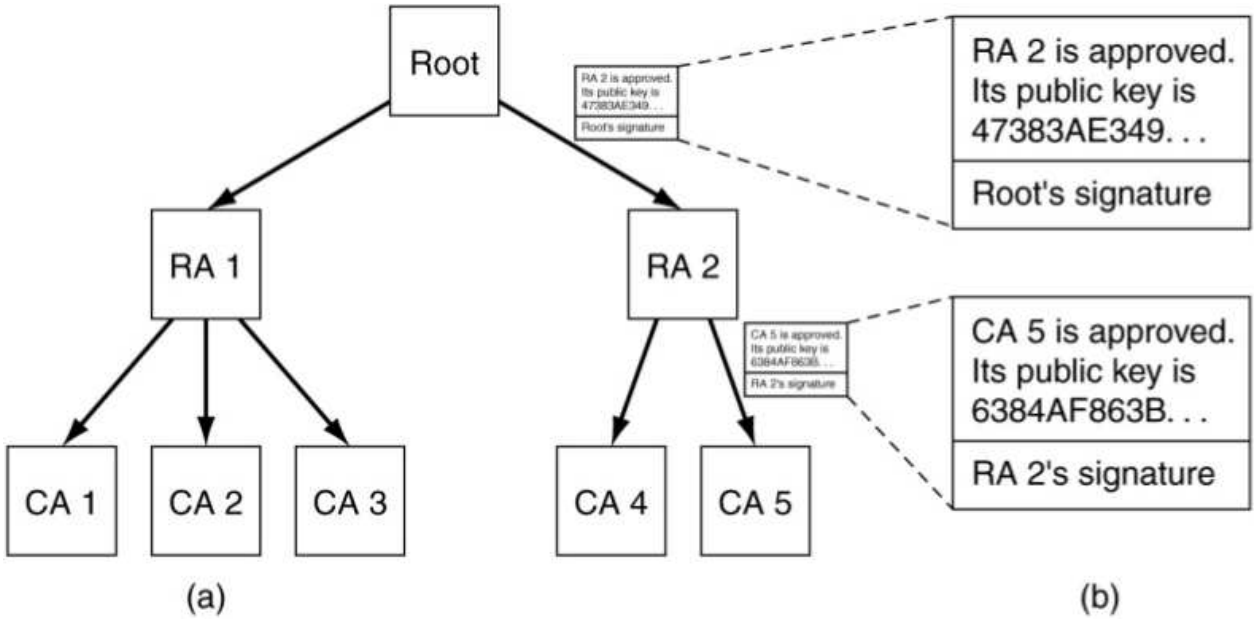
I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA's private key

# Management of Public keys: X.509

| Field | Meaning |
|---|---|
| Version | Which version of X.509 |
| Serial number | This number plus the CA's name uniquely identifies the certificate |
| Signature algorithm | The algorithm used to sign the certificate |
| Issuer | X.500 name of the CA |
| Validity period | The starting and ending times of the validity period |
| Subject name | The entity whose key is being certified |
| Public key | The subject's public key and the ID of the algorithm using it |
| Issuer ID | An optional ID uniquely identifying the certificate's issuer |
| Subject ID | An optional ID uniquely identifying the certificate's subject |
| Extensions | Many extensions have been defined |
| Signature | The certificate's signature (signed by the CA's private key) |

# Management of Public keys: Public Key Infrastructures




(a) A hierarchical PKI (b) A chain of certificates

# SSL/TLS Secure Transmissions

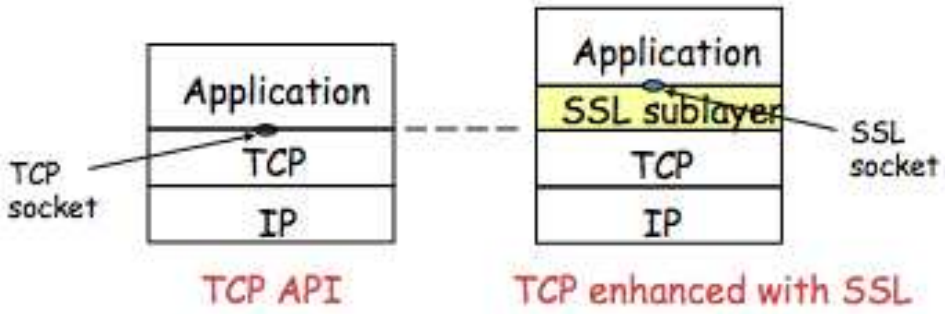
SSL/TLS is an almost-ubiquitous protocol for transmitting things securely over the Internet

- it checks the certificate of the server you are talking to
- it encrypts data (so nobody else can read it)
- it authenticates data (so nobody else can change it)

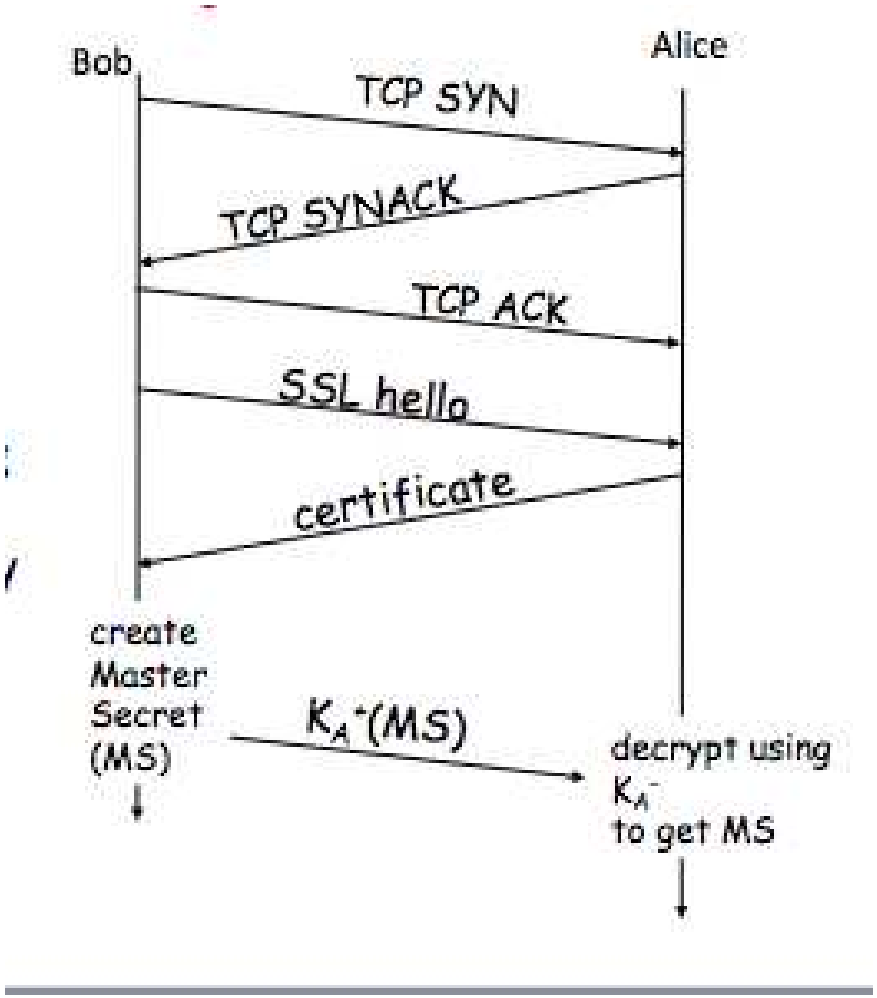
The comforting padlock in the corner of your web browser means you are using SSL/TLS

# Secure Sockets Layer (SSL)

Provides transport layer security to any TCP-based application using SSL services. e.g., between Web browsers, servers for e-commerce (https) 0.25cm Security services: server authentication, data encryption, client authentication (optional)



# SSL phases


Bob
Alice
TCP SYN
TCP SYNACK
TCP ACK
SSL hello
certificate
create
Master
Secret
(MS)
$K_A^+(MS)$
decrypt using
$K_A^-$
to get MS

# SSL phases

**Handshake:**

- Bob establishes TCP connection to Alice
- authenticates Alice via CA signed certificate
- creates, encrypts (using Alice's public key), sends master secret key to Alice nonce exchange not shown

## Key Derivation:

Alice, Bob use shared secret (MS) to generate 4 keys:

- $E_B$: Bob $\rightarrow$ Alice data encryption key
- $E_A$: Alice $\rightarrow$ Bob data encryption key
- $M_B$: Bob $\rightarrow$ Alice MAC key
- $M_A$: Alice $\rightarrow$Bob MAC key

Encryption and MAC algorithms negotiable between Bob, Alice

# Data transfer: