Security


Crypto

- Quantum Cryptography!
- Based on computationally infeasible problems
  - factoring products of large primes
  - discrete logarithms
- If P = NP most of this would break!
Symmetric Crypto

ciphertext = encrypt(plaintext, key)

plaintext = decrypt(ciphertext, key)

• Same (shared) secret key used for both encryption and decryption

• (same algorithm => bijection)

• DES (and variants like 3-DES) etc.
Diffie-Hellman Key Exchange

B ^ rand1

B ^ rand2

k1 = (B ^ rand2) ^ rand1  \quad k2 = (B ^ rand1) ^ rand2

- k1 = k2 computable by either participant
- Eavesdropper cannot compute k
  - (assuming discrete logarithm is difficult)
- Secure channel that is not authenticated!
Public Key Crypto

ciphertext = encrypt(plaintext, keya)

plaintext = decrypt(ciphertext, keyb)

- Public/private keys: keya, keyb
- Different keys used for encryption and decryption
- keya public => encryption mode
- keyb public => authentication mode
- RSA, ...
- Key Distribution -- PKI
Hash Function

digest = hash(message)

• Computationally feasible to compute hash of message
• Computationally infeasible to
  • given h, find m such that h = hash(m)
  • find m1, m2 where hash(m1) = hash(m2)
• Computationally cheaper than encryption
• SHA, (MD5)
Digital Signature

signature = encrypt(hash(message), privatekey)

send (message, (signer_id, signature))

OK = (decrypt(signature, publickey) = hash(message)

- Message is not secret
- Integrity checked
- Non-repudiation
Key Distribution

- Why do I believe it is your public key?
  - It’s in the New York Times ...
  - But maybe the bad buys have altered my copy of the Times
    - i.e. compromised the key server!
- Certificates!
X.509 Certificate

- Certifying Authority (CA) creates certificate and digitally signs it
- CA public key is well known
Certificate Revocation

- Certifying Authority (CA) creates revocation list and digitally signs it
- CRLs must be made widely available
Advanced Features

- Chaining of certificates
SSL Overview

- Provide authenticated secure channel
  - any TCP application
- Phases:
  - Handshake
    - authenticate
    - establish session key
- Data Transfer
Handshake (Overview)

Supported ciphers; Rand

Chosen cipher; Rand; Certificate

Encrypted PreMaster Secret

(compute keys both sides)

MAC of handshake messages

MAC of handshake messages
Key Derivation

Client Rand

PreMaster Secret

Server Rand

Master Secret

Client MAC

Server MAC

Client Write

Key Block
Record Protocol

Data

Data Fragment  MAC

Hdr  Encrypted Data and MAC

...
Client Authentication

Supported ciphers; Rand

Chosen cipher; Rand; Certificate

Client Certificate

Encrypted PreMaster Secret

(compute keys both sides)

MAC of handshake messages

MAC of handshake messages
Session Resumption

Supported ciphers; Rand, SessionID

Chosen cipher; Rand; Certificate

ChangeCipherSpec

ChangedCipherSpec

(recompute keys both sides)

MAC of handshake messages

MAC of handshake messages

MAC computation requires knowledge of original session master secret!
Choosing SSL

• Port selection
  • http://... uses standard port, unencrypted service
  • https://... SSL capable server on different port

• Negotiation
Password Client Authentication

• Problems:
  • external disclosure
  • guessing
  • eavesdropping
  • replay
  • host compromise
Password Client Authentication

• Problems:
  • external disclosure
  • Stealing it
  • Buying it for chocolate bars
  • guessing
  • eavesdropping
  • replay
  • host compromise
Password Client Authentication

- Problems:
  - external disclosure
  - guessing
    - a very common problem - who can remember all those passwords?
    - (obnoxious) site can require long non-word patterns
  - eavesdropping
  - replay
  - host compromise
Password Client Authentication

- Problems:
  - external disclosure
  - guessing
  - eavesdropping
  - replay
  - SSL is supposed to take care of these
  - host compromise
Password Client Authentication

• Problems:
  • external disclosure
  • guessing
  • eavesdropping
  • replay
  • host compromise
  • do what you can ...
Host Compromise

- User has same password at multiple sites
- DO NOT store passwords anywhere
  - not even encrypted!
- Use a one-way hash function
- Keep (encrypted) hashes in the database
- forgotten password requires reset
Host Compromise

- What about credit card numbers?
- Have to be stored somewhere
- Limit scope of damage?
  - e.g. encrypt with key derived from user password per session
  - (user changes password => credit card number must be re-entered)
- Verisign / PayPal style service
  - At what scale is this sensible?