Security

Fuzzy Stuff – Cryptosystems, Passwords, Trust

1. Holes in security will be found
    a. Simplifying assumptions reduce security
    b. Difficult systems unlikely to be properly used or maintained
2. Improper focus on security problem
    a. Underlying system is not at risk, rather the higher level is the weak link

Users are dumb

1. Easy passwords to decode
2. Fix
    a. System assigned passwords
    b. Disallow dictionary words

Trust

1. Trust nobody but yourself
2. Basic idea
    a. Program that prints itself
    b. Modify login prompt to accept special password
        i. Likely to be noticed
    c. Modify compiler to compile login with special password acceptance
    d. End result -- you're hosed

Security Mechanisms

1. Formal Authentication
2. Develop language for authentication procedures
3. Derive systems of authentication
4. Prove or disprove security of existing systems

In Practice

1. Boot securely
2. OS secure
3. ACL – channels for principals
4. Derive legitimate requests through ACL and deny requests not derivable

Confinement Problem

Prevent leaking of confidential data

a. All possible channels of leakage must be known and controlled

1. Storage
    a. Copying a file
    b. Dumping memory
    c. Messages
2. Legitimate
    a. Billing – batch system
3. Covert
    a. Power usage
    b. I/O rate
    c. Memory usage
4. Solution
    a. Only call other confined programs
    b. Memoryless
    c. Masking – caller determines all inputs into legitimate and covert channels
    d. Supervisor enforces paging rates, I/O rates, etc.

Encryption algorithms

General

Two principals wish to communicate securely

1. How do they set up a secure communications channel?
    a. Secret key – known to both principals
    b. Public key – each principal has own private key and can look up the others private key

How to establish keys

1. Secret key – faster than public key
    a. Manual setup
    b. Index set of known keys
        i. Snoopers may have same list of keys
    c. Public key to establish private key
2. Public key
    a. Lookup recipient public key at a trusted server (Kpb)
    b. Encrypt message with Kpb and send to B
    c. Only B can decrypt message using its private key Kb
    d. Similar for B to send back to A

Private key must be kept private

Key Concerns

Changing key

1. Why?
    a. Snoopers get too much data to decrypt
    b. Key is compromised
    c. Algorithms change
2. How? (public key)
    a. Principal communicates trusted server of its new private key
    b. Server must track key changes
3. How? (secret key)
    a. Same as setup of secret key

Freshness concerns

Freshness – how to tell if a message is new or is being replayed

Nonce – used once identifier

Still can be replayed

Timestamp – aid in replay detection

Freshness based on clock

Clock skew is a problem

Requires synchronization – hard

Digital signatures

You are responsible for messages you've signed

Signature must be verifiable

Server tracks old keys

Storing data with old encryption enables cracking on more data

Store time of key change

Kerberos

Tickets and time-stamp