CSE/EE 461 Lecture 24
Security Theory and Practice

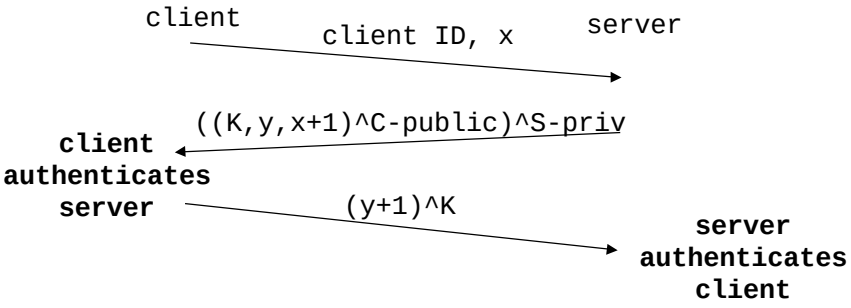
Tom Anderson
tom@cs.washington.edu
Peterson, Chapter 8

Secret Key, Public Key

- Encrypt messages for secrecy, authentication, message integrity
- Secret Key encryption
  - Single key (symmetric) is shared between parties, kept secret from everyone else
    - Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
- Public Key encryption
  - Keys come in pairs, public and private
    - Ciphertext = (M)^K-public; M = ((M)^K-public)^K-private
    - Ciphertext = (M)^K-private; M = ((M)^K-private)^K-public
  - Get both authentication and secrecy, by encrypting in private key of sender, public key of receiver

Public Key -> Session Key

- Public key encryption/decryption is slow; so can use public key to establish (shared) session key
  - assume both sides know each other's public key



Public Key Distribution

- How do we know public key of other side?
  - infeasible for every host to know everyone's key
  - need public key infrastructure (PKI)
- Certificates (X.509)
  - Distribute keys by trusted *certificate authority* (CA)
    - "I swear X's public key is Y", signed by CA (their private key)
  - Example CA's: Verisign, Microsoft, UW CS Dept., …
- How do we know public key of CA?
  - Can build chains of trust, e.g., given public key of UW CS's CA, who can sign for Verisign's public key, who can sign for xyz's public key

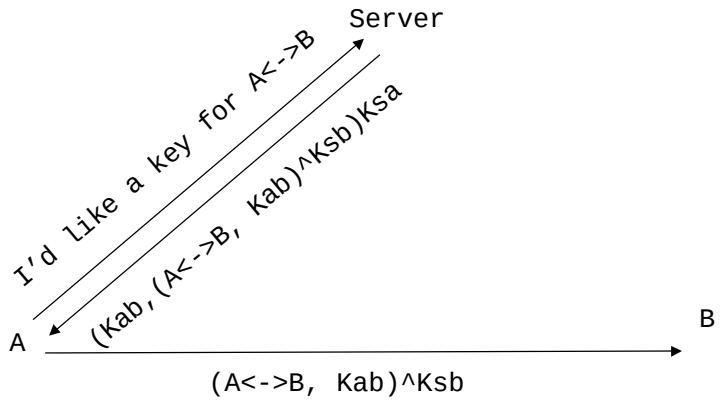
Public Key Revocation

- What if a private key is compromised?
  - need certificate revocation list (CRL)
    - and a CRL authority for serving the list
  - everyone using a certificate is responsible for checking to see if it is on CRL
  - ex: certificate can have two timestamps
    - one long term, when certificate times out
    - one short term, when CRL must be checked
    - CRL is online, CA can be offline

Shared Key -> Session Key

- In shared key systems, how do we gain a shared key with other side?
  - infeasible for everyone to share a secret with everyone else
  - solution: “authentication server” (Kerberos)
    - everyone shares (a separate) secret with server
    - server provides shared session key for A <-> B
  - everyone trusts authentication server
    - if compromise server, can do anything!

Kerberos Example

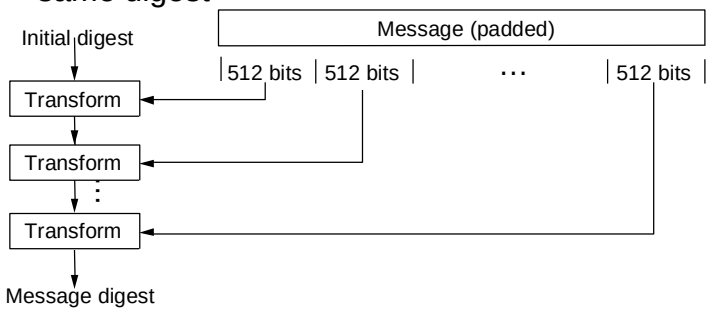


Kerberos Details

- Any key can be broken if given a long enough time
  - Use timestamps to ensure that keys were created recently
- Need to ensure attacker doesn't change messages in flight
  - ex: replace parts of message
  - use encrypted checksum on entire message
- Passwords are often easily broken
  - Derive Ksa from A's password
  - Use Ksa to establish temporary key, Ksa-temp

Message Digests (MD5, SHA)

- Cryptographic checksum: message integrity
  - Typically small compared to message (MD5 128 bits)
  - “One-way”: infeasible to find two messages with same digest



Example Systems

- Cryptography can be applied at multiple layers
- Pretty Good Privacy (PGP)
  - For authentic and confidential email
- Secure Sockets (SSL) and Secure HTTP (HTTPS)
  - For secure Web transactions
- IP Security (IPSEC)
  - Framework for encrypting/authenticating IP packets

PGP

- Application level system
- Based on public keys and a “grass roots” Web of trust
- Sign messages for integrity/authenticity
  - Encrypt with private key of sender
- Encrypt messages for privacy
  - Could just use public key of receiver …
  - But encrypt message with secret key, and secret key with public key of receiver to boost performance

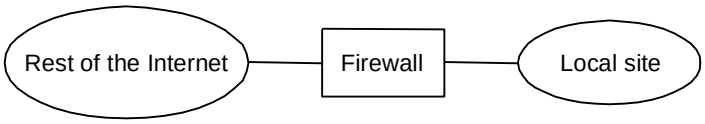
SSL/TLS and HTTPS

- Secure transport layer targeted at Web transactions
  - SSL/TLS inserted between TCP and HTTP to make secure HTTP
- Extra handshake phase to authenticate and exchange shared session keys
  - Client might authenticate Web server but not vice-versa
    - – Certificate Authority embedded in Web browser
- Performance optimization
  - Refer to shared state with session id
  - Can use same parameters across connections
    - – Client sends session id, allowing server to skip handshake

IPSEC

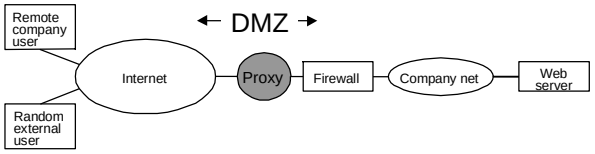
- Framework for encrypted IP packets
  - Choice of algorithms not specified
- Uses new protocol headers inside IPv4 packets
  - Authentication header
    - For message integrity and origin authenticity
    - Optionally “anti-replay” protection (via sequence number)
  - Encapsulating Security Payload
    - Adds encryption for privacy
- Depends on key distribution (ISAKAMP)
  - Sets up security associations
- Ex: secure tunnels between corporate offices

Filter-based Firewalls



- Sit between site and rest of Internet, filter packets
  - Enforce site policy in a manageable way
  - e.g. pass (*,*, 128.7.6.5, 80), then drop (*, *, *, 80)
  - Rules may be added dynamically to pass new connections
- Sometimes bundled with a router: “level 4” switch
  - Acts like a router (accepts and forwards packets)
  - Looks at information up to TCP port numbers (layer 4)

Proxy-Based Firewalls



- Problem: Filter ruleset can be complex/insufficient
  - Adequate filtering may require application knowledge
  - Example: email virus signature
- Run proxies for Web, mail, etc. just outside firewall
  - External requests go to proxies, only proxies connect inside
    - External user may or may not know this is happening
  - Proxies filter based on application semantics

Security Practice

- In practice, systems are not that secure
  - hackers can go after weakest link
    - any system with bugs is vulnerable
  - vulnerability often not anticipated
    - usually not a brute force attack against encryption system
  - often can't tell if system is compromised
    - hackers can hide their tracks
  - can be hard to resecure systems after a breakin
    - hackers can leave unknown backdoors

Two Old Examples

- Secure computer deep in Pentagon
  - Tiger team asked to see if they could break in
    - given all specs, source code, etc.
    - no physical access
  - Hacked into the system in < a week
- Secure communications channel: one time pad
  - paper tape of random #'s; same tape used at sender, receiver
  - system XOR random # to each bit before xmit
  - operational practice made system very insecure

Password Dictionary Attacks

- Moore's Law: brute force attacks become cheaper over time
- UNIX passwords: time to check all 5 letter passwords (lower case): 26^5 ~ 10M
  - in 75, 1 day
  - in 92, 10 seconds
  - in 02, 0.01 seconds
- Extend password to six letters, require upper, lower, number, control char: 70^6 ~ 600B
  - in 92, 6 days
  - in 02, with 100 PC's in parallel, < 1 minute (!)

Trojan Horse

- Can you trust your login prompt?
  - did the person before you really log out? how do you know?
- Can you trust your web browser?
  - what if someone modified the installed version to capture your password?
  - did you download the browser over the web? how do you know it didn't get modified in flight?
- Can you trust your email?
  - how do you know the sender sent the mail? that it wasn't modified?

Kerberos Weaknesses

- Early versions of Kerberos had several security flaws
  - block cipher: allowed encrypted blocks to be replaced
    - A -> B (transfer $10 to Tom's account)
    - A -> B (transfer $1M to Wells Fargo)
    - solution: add encrypted CRC over entire message
  - used timestamps to verify communication was recent
    - time server communication not encrypted
    - get time from authentication server
  - Kerberos login program downloaded over NFS
    - NFS authenticates requests, but data is unencrypted
    - disallow diskless operation

802.11 Weaknesses

- Ports often installed behind the firewall
  - anyone can listen, send packets on intranet
- Weak encryption method
  - uses 40 bit key, 32 bit initial #
  - most implementations use same initial #, allowing dictionary, replay attacks
- Key management overhead
  - single key used for all senders on a LAN; often disabled
- Uses parity instead of CRC for integrity
  - allows block replacements that maintain parity

Internet Worm

- Used the Internet to infect a large number of machines in 88
  - password dictionary
  - sendmail bug
    - default configuration allowed debug access
    - well known for several years, but not fixed
  - fingerd: finger tom@cs
    - fingerd allocated fixed size buffer on stack
    - copied string into buffer without checking length
    - encode virus into string!
- Used infected machines to find/infect others

Ping of Death

- IP packets can be fragmented, reordered in flight
- Reassembly at host
  - can get fragments out of order, so host allocates buffer to hold fragments
- Malformed IP fragment possible
  - offset + length > max packet size
  - Kernel implementation didn't check
- Was used for denial of service, but could have been used for virus propagation

TCP/DNS Hijacking

- Example: Mitnick
  - denial of service attack against system administrator
    - – open large number of TCP connections
  - scan for open, idle TCP connections (e.g., rlogin, xwindows)
    - – send bogus TCP packets to other end, e.g., to modify .rhosts to allow mitnick access
- Example: DNS cache poisoning
  - watch DNS cache for when it fetches new translation
    - – e.g., for cnn.com
  - spoof reply to poison cache to point to bogus server

Netscape

- Used time of day to pick session key
  - easy to predict, break
- Offered replacement browser code for download over Web
  - four byte change to executable made it use attacker's key
- Buggy helper applications (ex: ghostview)
  - if web site hosts infected content, can infect clients that browse to it

Microsoft

- Browser runs Java, supposedly "safe"
  - random byte code generation found numerous bugs that caused crashes
  - many could be used to covertly insert viruses
- Email viruses: Melissa, etc.
  - Attachments can run code that is poorly sandboxed

Code Red/Nimda

- Dictionary attack of known vulnerabilities
  - email attachments, Microsoft web server bugs, browser helper applications, …
  - used infected machines to infect new machines
- Code Red:
  - designed to cause all machines to access whitehouse.gov simultaneously
- Nimda:
  - Left open backdoor on infected machines for any use
  - Sysadmins could monitor virus propagation to located infected machines
  - Infected ~ 400K machines; approx ~30K still infected