P561: Network Systems
Week 9: Network Security

Tom Anderson
Ratul Mahajan

TA: Colin Dixon

Administrivia

Fishnet Assignment #4
- Due next Monday, Dec 1

Final Exam
- Handed out next Monday night (and by email)
- Due Monday, 12/8, 11:59pm, no extensions

No extensions allowed for fishnet assignments/ homework, even for reduced credit, beyond 12/5

2

Security in Practice

Attackers have the advantage
- Get to think outside the box
- Can exploit any unanticipated weakness
- Obscurity hard to maintain

Defense
- Needs to anticipate all feasible attack vectors
- Hard to prove that no attack is possible
  - Even at the crypto level
- Hard to detect if an attack has been successful
- Hard to re-secure a system after an attack

3

Fundamental Tenet: *If lots of smart people have failed to break a system then it probably won't be broken*

4

To Publish or Not to Publish

If the good guys break your system, you'll hear about it

If you publish your system, the white hats provide free consulting by trying to crack it

The black hats will learn about your system anyway

Today, most (but not all) commercial systems are published; most military systems are not

To Publish or Not to Publish (Part 2)

If you discover a workable attack, what is your responsibility?

Gap between discovery of vulnerability, and exploiting the vulnerability can be seconds

If you publish your system, the white hats provide free consulting by trying to crack it

The black hats will learn about your system anyway

Today, most (but not all) commercial systems are published; most military systems are not

Some Old Examples

Western Digital
- Compromise went undetected for months

Thompson self-propagating back door login
- Reinstalls itself in every new version of UNIX

Tiger team attempt on Pentagon computer
- No physical access

Secure communications channel: one time pad
- paper tape of random #'s
- same tape used at sender, receiver
- system XORs to each bit before xmit/receive

Some Recent Examples

House Keys
ATM keypad
Pacemakers
Mifare transit smart cards
Washington State Driver's Licenses (EPC RFID)
Electronic car keys
Elevator controls
Voting machines
WEP

802.11 WEP Weaknesses

Firewall often only at the perimeter
- 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/config
- single key used for all senders on a LAN; often disabled

Uses parity instead of CRC for integrity
- allows block replacements that maintain parity

Network Security

Networks are shared
- each packet traverses many devices on path from source to receiver

Attacker might be in control of any of these devices
- Or other machines on the network
- Or administrative machines
- Or, ...

Network Security

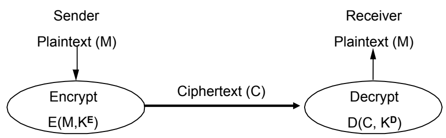
How do you know messages aren't:
- Copied
- Injected
- Replaced/modified
- Spoofed
- Inferred
- Prevented from being delivered
- ...

Network Security Goals

Despite the presence of malicious parties:

Privacy: messages can't be eavesdropped/inferred
Authentication: messages were sent by the right party
Integrity: messages can't be tampered with
Denial of Service: messages are delivered

Encryption



Cryptographer chooses E, D and keys K^E, K^D

- Suppose everything is known (E, D, M and C), should not be able to determine keys K^E, K^D and/or modify C without detection
- provides basis for authentication, privacy and integrity

How Secure is Encryption?

An attacker who knows the algorithm we're using could try all possible keys

Security of cryptography depends on the limited computational power of the attacker

A fairly small key (e.g. 128 bits) represents a formidable challenge to the attacker

Algorithms can also have weaknesses, independent of key size

How Practical is Encryption

Usability depends on being efficient for the good guys

Cost to the good guys tends to rise linearly with key length

Cost to search all keys rises exponentially with key length

How do we keep keys secret?

- Short keys: easy to remember, easy to break

How Secure are Passwords?

UNIX passwords: time to check all 5 letter passwords (lower case): 26^5 ~ 10M

- in 75, 1 day
- in 92, 10 seconds
- In 08, 0.001 seconds

Extend password to six letters, require upper, lower, number, control char: 70^6 ~ 600B

- in 92, 6 days
- in 08, with 1000 PC's in parallel, < 1 second (!)

Password Attack/Response

Moore's Law: enables large number of passwords to be checked very quickly

Countermeasure

- Delay password check for 1 second, so can't try them quickly
- Need to delay both successful and unsuccessful password checks!

Counter-countermeasure:

- Observe network traffic; extract any packet encrypted in password; check various passwords offline

Counter-counter-countermeasure:

- Kerberos: don't use password to encrypt packets; instead use password to encrypt file containing shared key; use shared key to encrypt packets

Counter-counter-counter-countermeasure: …

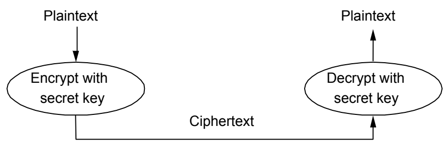
Cryptography

Secret Key Cryptography (DES, IDEA, RCx, AES)

Public Key Cryptography (RSA, Diffie-Hellman, DSS)

Message Digests (MD4, MD5, SHA-1)

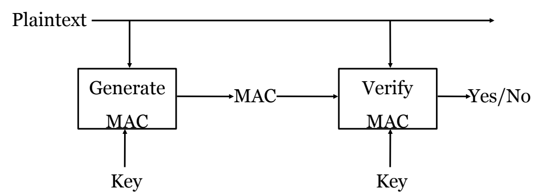
Secret Key



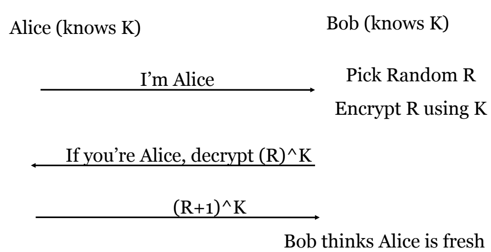
Single key (symmetric) is shared between parties, kept secret from everyone else

- Ciphertext = (M)^K; Plaintext = M = ((M)^K)^K
- if K kept secret, then both parties know M is authentic and secret

Secret Key Integrity: Message Authentication Codes



Challenge / Response Authentication



Secret Key Algorithms

DES (Data Encryption Standard)

- 56 bit key (+ 8 parity bits) => has become too small
- Input and output are 64 bit blocks
- slow in software, based on (gratuitous?) bit twiddling

IDEA (International Data Encryption Algorithm)

- 128 bit key
- Input and output are 64 bit blocks
- designed to be efficient in software

Secret Key Algorithms

Triple DES

- Apply DES three times (EDE) using K1, K2, K3 where K1 may equal K3
- Input and output 64 bit blocks
- Key is 112 or 168 bits

Advanced Encryption Standard (AES)

- New NIST standard to replace DES.
- Public Design and Selection Process. Rijndael.
- Key Sizes 128,192,256. Block size 128.

Secret Key Algorithms

RC2 (Rivest's Cipher #2)

- Variable key size
- Input and output are 64 bit blocks

RC4 (Rivest's Cipher #4)

- Variable key size
- Extremely efficient
- Stream cipher - one time use keys

Many other secret key algorithms exist

It is hard to invent secure ones!

No good reason to invent new ones

Encrypting Large Messages

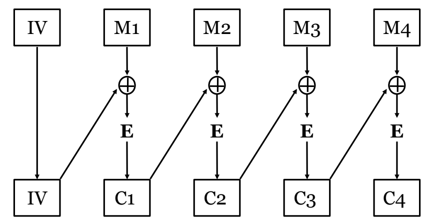
The basic algorithms encrypt a fixed size block

Obvious solution is to encrypt a block at a time. This is called Electronic Code Book (ECB)

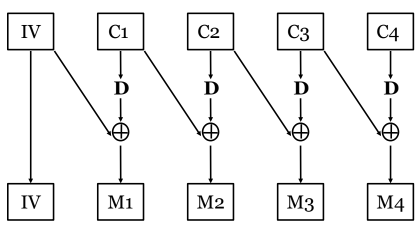
- Leaks data: repeated plaintext blocks yield repeated ciphertext blocks
- Does not guarantee integrity!

Other modes “chain” to avoid this (CBC, CFB, OFB)

CBC (Cipher Block Chaining)



CBC Decryption



XOR (Exclusive-OR)

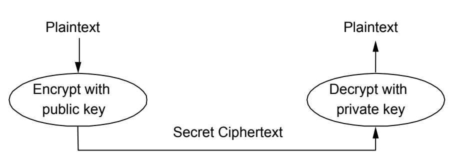
Bitwise operation with two inputs where the output bit is 1 if exactly one of the two input bits is one

(B XOR A) XOR A) = B

If A is a “one time pad”, very efficient and secure

Common encryption schemes (e.g. RC4) calculate a pseudo-random stream from a key

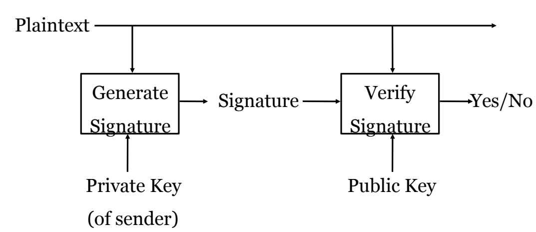
Public Key Encryption



Keys come in pairs, public and private

- Each entity (user, host, router,...) gets its own pair
- Public key can be published; private is secret to entity
  - can’t derive K-private from K-public, even given M, (M)^K-priv
- If encrypt with receiver’s public key, ensures can only be read by receiver

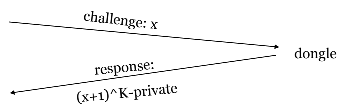
Public Key Integrity Protection



Zero Knowledge Authentication

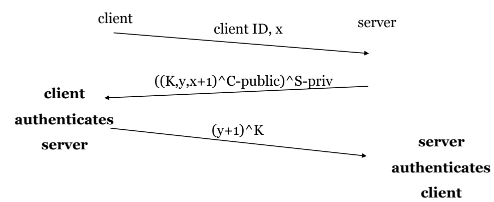
Where to keep your private key?
- keys that are easy to remember, are easier to break
- keys that aren't easy to break, can't be remembered!
- If stored online, can be captured

Instead, store private key inside a chip
- use challenge-response to authenticate user



Public Key -> Session Key

Public key encryption/decryption is slow; so can use public key to establish (shared) session key
- If 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., …
- But! Doesn't mean entity is trustworthy!

How do we know public key of CA?
- Typically, hard-coded into browsers
- Alternative: build chain of trust, e.g., from UW's CA to list of CA's that UW trusts

Public Key Revocation

What if a private key is compromised?
- Hope it never happens?

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

Secret Key -> Session Key

In secret key systems, how do we get a secret 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 session key for A <-> B
- everyone trusts authentication server
  - if compromise server, can do anything!

Kerberos

Developed at MIT

Based on secret key cryptography

Code is publicly available (for a long time not legally exportable from the U.S.)

Early version used block cipher
- Vulnerability caught and fixed

Embedded in a variety of commercial products
- Ex: in use by UW CSE

Kerberos Authentication (Basic)

Alice KDC Bob

Alice wants Bob →

← {“Bob”, Kab, {“Alice”,Kab}^Kb}^Ka

{“Alice”, Kab}^Kb, {timestamp}^Kab →

← {timestamp+1}^Kab

Ticket Granting Tickets

It is dangerous for the workstation to hold Alice’s secret for her entire login session

Instead, Alice uses her password to get a short lived “ticket” to the “Ticket Granting Service” which can be used to get tickets for a limited time

For a login session >8 hours, she must enter her password again

Ticket Granting Tickets

TGT looks just like ticket but encrypted with KDC’s key

WS keeps TGT = {“Alice”,S}K_{kdc} and S

Kerberos Authentication (with TGT={“Alice”,S}K_{kdc})

Alice KDC Bob

Alice wants Bob, TGT →

← {“Bob”, K_{ab}, {“Alice”,K_{ab}}^K_b}^ S

{“Alice”, K_{ab}}^K_b, {timestamp}^K_{ab} →

← {timestamp+1}^K_{ab}

Pre-authentication

Anyone can request a ticket on behalf of Alice, and the response will be encrypted under her password

This allows an off-line password guessing attack

Kerberos V5 requires an encrypted timestamp on the request

- Only an eavesdropper can guess passwords

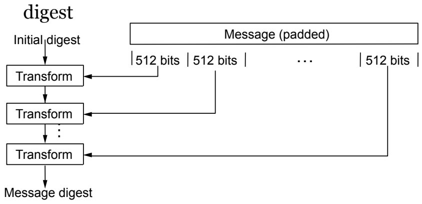
Kerberos Weaknesses

Early versions of Kerberos had several security flaws

- block cipher: allows encrypted blocks to be replaced
  - solution: add encrypted CRC over entire message
- uses 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?

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

TCP Hijacking

Example: Mitnick
- Denial of service attack against system administrator
  - open large number of TCP connections
- Followed by attack on user machines

Scan for open, idle TCP connections (e.g., rlogin, xwindows)

Send spoofed TCP packets to other end, e.g., to modify .rhosts to allow future access
- Requires ability to predict TCP sequence #

Fixed with SSL

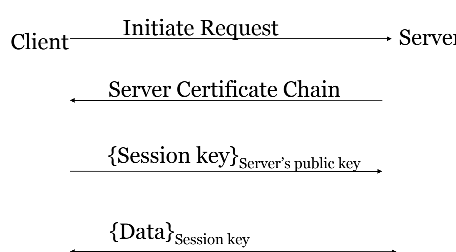
SSL/TLS and HTTPS

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

SSL/TLS



IPSEC

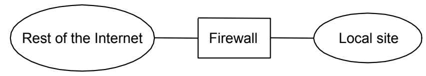
Framework for encrypted IP packets
- Choice of algorithms not specified

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

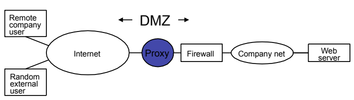
Filter-based Firewalls



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 allow 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

Trojan Horse

Can you trust your login prompt?
- did the sysadmin install the software correctly? 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?
- 20 minutes from BitTyrant release => virus at mirror

Can you trust your email?
- how do you know the sender sent the mail? that it wasn’t modified?

Phishing

Modern day trojan horse

Web page or email that appears to be from bank/ commercial entity
- Attacker inserts spoofed forms, links, executables
- Gathers login information, installs spyware, etc.

How do you protect yourself against phishing?
- Web pages at common misspellings (or unicode)
- Google ad listings
- Email alert from bank

Never trust anything on the web?

53

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

Morris 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

More Worms

Often use a dictionary of known vulnerabilities
- email attachments, Microsoft web server bugs, browser helper applications, …
- use infected machines to infect new machines
- Collateral damage: router DoS due to reverse ARP

Code Red (2000)
- designed to cause all infected machines to access whitehouse.gov at a defined date
- Brought down a large number of routers
- Short term fix: assign whitehouse a new IP address
- Still a substantial # of infected Code Red machines!

More Worms

Nimda: Code Red, but better engineered (2001)
- Left open backdoor on infected machines for any use
- Can monitor virus propagation to located infected machines
- Infected ~ 400K machines; approx ~30K still infected

SQL Slammer (2003)
- Exploited buffer overflow in SQL server
- Vulnerability had been identified, fixed and publicized six months earlier!
- Entire worm fit in one packet => rapid propagation

What are limits on virus propagation?
- Is automated response/quarantine even possible?

DNS Cache Poisoning

If attacker can know when DNS cache fetches a new translation
- spoof reply to poison cache to point to bogus server
- With a large TTL so it never refetches

Solution: DNS-SEC
- Digitally signed DNS records
- Need chain of signatures from root to leaf
- Not widely deployed

BGP Hijacking

BGP prefix origin announcements are not signed
- Easy to announce a new prefix
- Packets diverted to new origin (if closer to the source)
- Often done by mistake (1/2 of all new announcements done by mistake!)
- Ex: Cisco's prefix hijacked repeatedly
- Pakistan ISP hijacked YouTube intentionally

Solution: Secure BGP and variants
- Digitally signed BGP records
- Need chain of records from destination to source
- Not widely deployed

Denial of Service

Prevent access to a service by intended users
- Ex: Georgia
- Ex: extortion
- Ex: Root DNS servers

Any fixed resource can be overwhelmed
- Memory in the server (e.g., Mitnick)
  - Solution: SYN cookies, per-prefix connection limits
- CPU in the server
  - Solution: resource containers inside OS kernel
- DNS processing/bandwidth
  - Replication/longer TTLs

Denial of Service v2.0

What if DoS attack looks like a flash flood?
- Recruit large botnet (cf. viruses, worms)
  - 1M broadband nodes => 1Tb/s of traffic
- Activity could appear completely normal!
- Congestion can occur well upstream of destination

Solution: destination controls delivery
- Only deliver pre-approved packets
- How is connection set up in the first place?
- How does endpoint tell network what is ok?
- How does network implement filtering?
- What if partial deployment?

61

Thompson Virus

Ken Thompson self-replicating program
- installed itself silently on every UNIX machine, including new machines with new instruction sets

Aside: can you write a self-replicating C program?
- program that when run, outputs itself
  - without reading any input files!
- ex: main() { printf("main () { printf("main () …

Add backdoor to login.c

Step 1: modify login.c

A:
```
if (name == "ken") {
    don't check password;
    login ken as root;
}
```

Modification is too obvious; how do we hide it?

Hiding the change to login.c

Step 2: Modify the C compiler

B:
```
if see trigger {
    insert A into the input stream
}
```

Add trigger to login.c

/* gobblygook */

Now we don't need to include the code for the backdoor in login.c, just the trigger
- But still too obvious; how do we hide the modification to the C compiler?

Hiding the change to the compiler

Step 3: Modify the compiler

C:
```
if see trigger2 {
    insert B and C into the input stream
}
```

Compile the compiler with C present
- now in object code for compiler

Replace C in the compiler source with trigger2

Compiler compiles the compiler

Every new version of compiler has code for B,C included
- as long as trigger2 is not removed
- and compiled with an infected compiler
- if compiler is for a completely new machine: cross-compiled first on old machine using old compiler

Every new version of login.c has code for A included
- as long as trigger is not removed
- and compiled with an infected compiler