

CS290i - Lecture 9

Secured servers

Scalable Internet Services and Systems, Winter 2002

Thorsten von Eicken
Department of Computer Science
University of California at Santa Barbara

Schedule

- Fri 2/8: project #2 due
- Fri 2/15
- Fri 2/22: project #3 due
- Fri 3/1:
- Fri 3/8:
- Fri 3/15: project #4 benchmarking (last day of classes)
- Fri 3/22: project #4: final results due

2

Overview

- Symmetric cryptography
- Asymmetric cryptography
- Certificates
- TLS1.0/SSL3.1
- Securing servers
- Firewalls & routers

3

Goals of a Cryptosystem

- Maintaining security
 - by restricting knowledge of the key(s) to trusted entities
 - never "by obscurity", i.e. algorithms/protocols used are assumed public
- Foundation for security
 - algorithms that are computationally intractable to reverse without knowledge of the key(s)
 - e.g., it would take the fastest super computer 100 million years to break the algorithm
- Level of encryption must be in proportion to data lifetime
 - e.g. if currently sensitive data is useless in five minutes, then weak encryption may be OK

4

Basic vocabulary

- Plaintext (P) – Unencrypted data
- Ciphertext (C) – Encrypted data
- Encryption (E) – Converting plaintext to ciphertext
- Decryption (D) – Converting ciphertext back to plaintext
- Key – A fixed length sequence of bits
 - ♦ The meaning of the bits depends on the algorithm
 - ♦ In general
 - | plaintext + key + algorithm => ciphertext
 - | ciphertext + key + algorithm => plaintext
 - ♦ Symmetric algorithms use the same key to both encrypt and decrypt data
 - ♦ Asymmetric algorithms use one key to encrypt and a different key to decrypt
- Authentication – The ability of the receiver of a message to ascertain its origin. The sender “signs” the message

5

A Simple Example

- Bob encrypts a message, M, with key, K1, and sends it Alice
 - ♦ Notation: $E_{K1}\{M\}$
- Alice decrypts the encrypted message she received from Bob with key, K2, and ends up with the original message
 - ♦ Notation: $D_{K2}\{E_{K1}\{M\}\} = M$
- Depending on the algorithm, it may or may not be the case that K1 equals K2
- Essentially, the functions E and D are inverses of each other. In some cases, E and D are the same function (e.g. DES)

6

Symmetric Cryptography

- A single key is used to both encrypt and decrypt data
- Operations: substitution, transposition, and bitwise operations
- Through a series of rounds, the key and data are “scrambled” together to either encrypt or decrypt
- Why use these simple operations?
 - ♦ They are very fast
 - ♦ They are simple to implement in both hardware and software
 - ♦ Enough repetition (number of rounds) with sufficient key length can provide excellent security
- Famous algorithms: DES, AES (Rijndael), Blowfish, RC4, PKZIP, Enigma, etc...

7

Symmetric Alg. Types

■ Two basic types:

- Block ciphers
- Stream ciphers

■ Block ciphers

- operate on blocks of plaintext and ciphertext
- blocks usually are 64 bits or longer.

■ Stream ciphers

- operate on streams of plaintext and ciphertext
- conceptually, they operate on one bit at a time
- usually they are implemented to operate one byte or word at a time

8

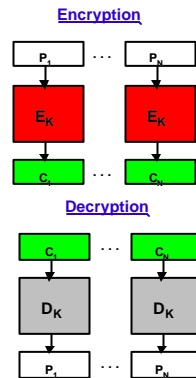
Symmetric Block Ciphers

Simple approach

- data is divided into equal sized blocks
- each is encrypted, forming the encrypted data stream

Problem

- messages usually have a regular structure or pattern
- given the plaintext and ciphertext for several messages, their structure can be exploited to more quickly break the cipher

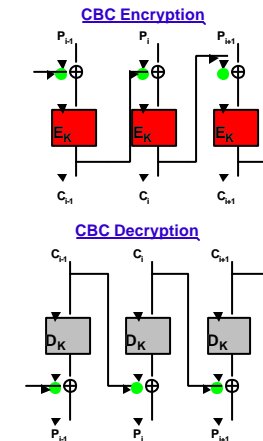


9

Symmetric Block Ciphers

Solution

- Cipher Block Chaining Mode (CBC)
- each plaintext block is XORed with the previous ciphertext block before it is encrypted.



10

Cipher Block Chaining

- Each ciphertext block depends on:
 - the plaintext block that generated it
 - and all previous plaintext blocks
 - this eliminates patterns.
- The encrypted data stream becomes
 - $E_K\{P1\} = C1$ followed by
 - $E_K\{P2 \text{ XOR } C1\} = C2$ followed by
 - $E_K\{P3 \text{ XOR } C2\} = C3$ followed by
 - ...
 - $E_K\{PN \text{ XOR } CN-1\} = CN$
- What about the first block?
 - use an Initialization Vector (IV): a random block-sized series of bits chosen by the sender
 - $E_K\{P1 \text{ XOR } IV\} = C1$ followed by
 - added benefit: identical messages encrypt differently per IV
 - IV can be transmitted in the clear to the receiver

11

Asymmetric Cryptography

- Often referred to as public-key cryptography
- Two different keys
 - one public
 - one private
 - computationally hard, if not impossible, to determine the private key given the public key
- For instance, RSA key pairs are based around large prime numbers
 - strength: it is computationally infeasible to factor these numbers in a reasonable time frame
- Each key pair is associated with an entity
 - E.g., a person, a company, a server, etc.
 - An entity distributes its public key, but keeps its private key secret

12

Properties of PK Crypto

- Properties:
 - ◆ $D_{K_{priv}} \{ E_{K_{pub}} \{ M \} \} = M$
 - ◆ $D_{K_{pub}} \{ E_{K_{priv}} \{ M \} \} = M$
- PK cryptography can be used for privacy and authentication
 - ◆ Bob and Alice exchange their public keys
 - ◆ Consider this message from Bob to Alice:
 - ! $C = E_{K_{Alice_pub}} \{ E_{K_{Bob_priv}} \{ P \} \}$
 - ◆ Alice is the only one that can decrypt the "outer layer"
 - ! $D_{K_{Alice_priv}} \{ C \} \Rightarrow P1 = E_{K_{Bob_priv}} \{ P \}$
 - ◆ Alice then verifies the message must have come from Bob
 - ! $D_{K_{Bob_pub}} \{ P1 \} \Rightarrow P$

13

Why not just PK Crypto?

- **Speed**
 - PK cryptography is much slower than symmetric encryption
 - it is based on arithmetic operations on very large integers
 - extremely expensive next to the operations typical symmetric algorithms use
- **Solution: hybrid cryptosystem**
 - public-key cryptography
 - ◆ Identify correspondents
 - ◆ Securely exchange session keys
 - symmetric key cryptography
 - ◆ Secure message traffic

14

Certificates

■ How to distribute public keys safely?

- Idea: a known entity introduces a foreign one
- Consider the message:
 - ◆ Foreigner: $E_{K_{friend_priv}}(\text{foreigner}, K_{foreigner_pub})$
 - ◆ Provides the foreigner's name and public key
 - ◆ Signed by a friend, thus establishing authenticity
- Certificate chains:
 - ◆ Friend certifies key of guy1, guy1 certifies key of guy2, guy2 certifies key of guy3, etc...
 - ◆ PGP parties where friends exchange certificates
 - ! Allows me to receive signed email from my friends' friends
 - ◆ Verisign certifies Sun's key, Sun certifies Solaris division's key
 - ! Verisign's key is "well-known"
- Embed expiration dates into certificates

■ Revocation: how?

15

One-Way Hash Functions

- Operation
 - ◆ operate on a message of arbitrary length, M
 - ◆ return a fixed-length hash value, $H(M) = h$
- "One-wayness" provides security:
 - ◆ Given M, it is easy to compute h
 - ◆ Given h, it is hard to compute M_i such that $H(M_i) = h$
 - ◆ Given M, it is hard to find another message, M' , such that $H(M) = H(M')$
 - ! This is called the "collision-free" property
- Uses:
 - ◆ Message checksum, message integrity check
 - ◆ Verify that someone has the same file as you without sending the whole file

16

TLS1.0 / SSL3

Idea:

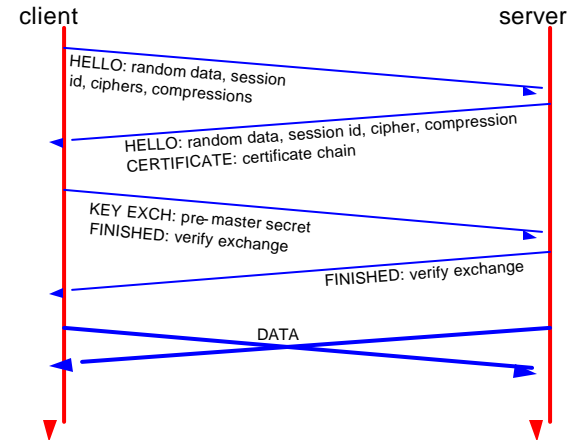
- Majority of protocols use a byte stream
- Provide encrypted byte stream transparently
- "Secure Socket Layer" initially developed by Netscape

TLS 1.0 = SSL 3.1

- IETF "blessed" standard, RFC 2246
- Socket interface:
 - connect, write, read, close
 - record boundaries not preserved
- TLS interfaces:
 - Provides socket interface
 - Uses socket interface: sends and receives records over a reliable byte stream

17

TLS Overview



18

Securing servers

- Tcpwrapper
- Yassp
- Tripwire
- Firewalls
- Intrusion detection

19

Minimal processes

Run only required processes:

```
#ps -ef
UID      PID  PPID  C   STIME TTY          TIME CMD
root     0    0    0   Jan 22 ?           0:00 sched
root     1    0    0   Jan 22 ?           0:02 /etc/init -
root     2    0    0   Jan 22 ?           0:00 pageout
root     3    0    0   Jan 22 ?           66:15 fsflush
root 17633  1    0   Jan 22 console 0:00 /usr/lib/saf/ttymon
root    178  1    0   Jan 22 ?           0:07 /usr/lib/inet/xntpd
root    247  1    0   Jan 22 ?           0:11 /opt/local/sbin/sshd
root    160  1    0   Jan 22 ?           0:00 /usr/sbin/inetd -t -s
root   1235  1    0 23:58:04 ?           0:00 /usr/sbin/syslogd -t
root    174  1    0   Jan 22 ?           0:03 /usr/sbin/cron
root    333  1    0   Jan 22 ?           87:52 /opt/EMPsysedge/bin/sysedge
root   1518 247  0 08:59:36 ?           0:00 /opt/local/sbin/sshd
root   1536 1518 0 08:59:38 pts/1       0:00 -ksh
root   1549 1536 0 08:59:49 pts/1       0:00 ps -ef
logsrfr 234  1    0   Jan 22 ?           0:00 /usr/local/bin/logsurfer
broker  1338  1    0 02:50:06 ?           0:00 perl ./bin/wrapper.pl
broker  1343 1338 0 02:50:06 ?           0:00 /usr/java/bin/java
broker  1345 1343 0 02:50:06 ?           399:56 /usr/java/bin/..
```

Only allow access via SSH

20

Some Tools

Tcpwrapper (linux : tcpd)

- Inetd -> tcpwrapper -> server process
- Logs accesses
- Simple access control rules
 - ◆ Source IP address
 - ◆ Reverse DNS lookups

Yassp

- Yet another secure solaris package
- Configures Solaris as appropriate for an external, "hardened" host

Cfengine

- Periodic upload of config info & scripts to apply them

21

Some Tools (cont.)

Tripwire

- tool that checks to see what has changed on your system
 - ◆ a form of intrusion detection
- monitors key attributes of files that should not change
 - ◆ binary signature, size, expected change of size, etc.
- Mean installation — periodically:
 - ◆ Open SSH connection to server
 - ◆ Upload tripwire executable
 - ◆ Upload tripwire config & state
 - ◆ Run tripwire & gather results
 - ◆ Delete all "evidence"

22

Firewalls & Routers

Access lists

- Example:
 - ◆ `permit tcp 10.16.2.0 0.0.1.255 10.0.0.0 0.255.255.255 eq smtp`
- Items:
 - ◆ Protocol (ip, tcp,udp, icmp ...)
 - ◆ Source & destination (address, address mask, port)
 - ◆ SYN/ACK/FIN/RST/ACK/established
- Typical settings:
 - ◆ Deny incoming with your source addresses
 - ◆ Deny outgoing with other than your source addresses
 - ◆ Deny incoming for protos/ports other than public services
 - ◆ Deny outgoing to protos/ports other than admin stuff

23

Firewalls & Routers (Cont.)

Content-based access control

- Keep track of TCP connections
 - ◆ Initially only accept SYN
 - ◆ Watch handshake & enter into table
 - ◆ Only accept "data" packets per table
 - ◆ Drop connection from table when it ends
- Simulate UDP "connections"
 - ◆ Watch outgoing UDP packet
 - ◆ Then allow incoming packets with same addr/port pairs
 - ◆ Timeout after inactivity
- Keep track of higher-level protocols
 - ◆ FTP uses multiple sockets
 - ◆ Video conferencing uses control & data streams
- Authentication & authorization
- VPN

24

Intrusion Detection

■ Watch the network for suspicious activity

- Configured rules
 - ◆ E.g. to match known attacks (buffer overflow, SMB probes, ...)
- Different from past observed activity
- Statistical profiling
- Honey-pots

■ And/or watch hosts

- E.g. tripwire