Course on

Computer networks

Application layer

Henk Corporaal

Leiden University
2000

Application Layer

Overview

- Network security
- DNS : Domain Name System
- SNMP : Simple Network Management Protocol
- Electronic Mail
  - SMTP : Simple Mail Transfer Protocol
- Usenet News
  - NNTP : Network News Transfer Protocol
- World Wide Web
  - HTTP : Hypertext Transfer Protocol
  - HTML : Hypertext Markup Language
  - Java
- Multi Media
  - MPEG
  - MBone

Network Security

Overview

- Introduction
- Traditional Cryptography
  - substitution ciphers
  - transposition ciphers
- Cryptographic principles
- Secret-key algorithms
  - DES: data encryption standard
- Public-key algorithm
  - RSA: Rivest, Shamir and Adleman
- Authentication protocols
- Digital signatures

Network Security : Introduction

Four security areas:

- Secrecy
  - keeping information from unauthorized users
- Authentication
  - determine origin of message (who am I talking to ?)
- Non repudiation
  - dealing with signature : how do I prove you wrote the letter, and made the deal ?
- Integrity control
  - Is your message not modified ?

Network Security : Introduction

Where to put security ?

- Secrecy can be put in transport layer, but not in lower layers (because decoding required at each router)
- authentication and signature checks has to be put in application layer

Four attack categories:

- Interruption
- Interception (= tapping)
- Modification
- Fabrication

Traditional Cryptography

Encryption model : fig. 7-2

Terminology

- plaintext: the message to be encrypted
- key : the encryption function is often parameterized by a key
- ciphertext : output of the encryption process
- cryptanalysis : art of breaking ciphers
- cryptology ; art of designing encryption methods (cryptography) and breaking them (cryptanalysis)

Traditional Cryptography

Plain text : P
Key : K
Encryption method : E
Decryption method : D
Cipher text : E_k (P) = C
It follows : D_k (E_k (P)) = P
Crypt analyst is assumed to know encryption method (but not the key)
Three decipher options:

- cipher text only C_i for some i
- known plaintext (P_i, C_i) for some i
- chosen plaintext (P_i, C_i) for any i

Traditional Cryptography

Substitution ciphers

- shift alphabet by k letters (Caesar cipher)
- permuted alphabet
  - mono alphabetic substitution
  - 26 ! ~ 4 x 10^{26} possible keys

Decipher

- use known frequencies of letters ; eg. e is most common letter
- use frequent patterns of 2 or 3 letters (digrams and trigrams)
- use likely word

Note : substitution preserves order of symbols

Traditional Cryptography

Transposition cipher

- reorders symbols, but do not change them
- example : the columnar transportation, fig 7-3
  - write rows
  - read out columns

How to decipher ?

- Look at frequencies of E, T, A, etc.
- If they match normal frequencies than it is indeed a transposition cipher
- Guess # columns (=k)
  - use probable word (out of message context)
    - eg. million dollars
    - observe (fig 7-3) diagrams : MO, IL, LL, LA, etc.
- Order columns
  - check k (k-1) column pairs
  - keep best pair, and try next column; etc

Traditional Cryptography

One time Pads : easy unbreakable cipher

- Consider plaintext P
- Choose random bitstring as key K, |K| = |P|
  - C = P $\oplus$ K

C can not be broken :

- it does not contain any information at all !
  - each letter occurs with some frequency

Problems:

- key is difficult to memorise
- |K| $\geq$ |P|
  - gigabit networks needs one CD per 5 seconds !!!
- sensitivity to lost/inserted bits => loss of synchronization

Cryptographic principles

Two fundamental principles of any cryptographic system:

- C needs redundancy
  i.e $\exists$ c $\in$ C for which D_k (C) is not valid
  - otherwise it would be easy to generate valid messages
  - however, redundancy makes deciphering easier
    - for any key K, D_K (C) has to make sense otherwise K cannot be valid
    - add random string as redundancy
- Re-playing messages must be avoided
  - use time stamp with time out

Secret-key Algorithms

Based on ***transition*** and ***substitution***

However:

- Traditionally: easy algorithm and very long keys
- Today: complex algorithm and shorter keys

Basic transposition and substitution circuits: fig. 7-4

- P box : implements a permutation (i.e. transition)
- S box : implements substitution of symbols
- S and P boxes can be arbitrarily combined to form very complex functions

Secret-key Algorithms

DES : data encryption standard

- IBM developed
- 1977: accepted as U.S. government standard
- Encryption and decryption use same key
  - Steps run in same order with keys reversed

DES algorithm is shown in fig. 7-5

- 16 iterations + initial and final transposition and final swap
- each iteration uses different key K_i (i=1..16)

DES details

For each iteration:

- split 64-bit value into 2 32-bit parts, L_{i-1} and R_{i-1}
- apply $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- f is a complex function involving both permutations and substitutions
  $f(R_{i-1}, K_i) = P_{32}S_{48\text{-}32}(K_i \oplus S_{32\text{-}48}R_{i-1})$
- keys: $K_0' = P_{56}K \Rightarrow C_0 + + D_0$
  $K_1' = R_1^a C_0 + + R_1^b D_0$
  $K_i' = R_i^a C_{i-1} + + R_i^b D_{i-1}$
  $K_i = P_{48}S_{56\text{-}48}P_{56}K_i^1$

DES Chaining

DES is basically a monoalphabetic substitution cipher using 64-bit characters

$\alpha \in P \rightarrow ß = E_k(a) \in C$

- easy to replace (modify) a cipher
  - see example in fig 7-6: electronic code book mode

Solution: chaining (fig 7-7):

- Characters (8 byte each): P_i
- $C_0 = E_\kappa(P_0 \oplus IV)$
  - IV = initialization vector of 64 bits
- $C_i = E_k(P_i \oplus C_{i-1})$

DES Chaining

Chaining: decryption similar to encryption (fig 7-7)

- $P_0 = IV \oplus D_K(C_0)$
- $P_i = C_{i-1} \oplus D_K(C_i)$

Check that decryption is inverse of encryption !

Disadvantage: entire block of 64-bit needed before decryption can begin

- Solution; cipher feedback mode (fig 7-8)
- Note that decryption is almost equal to encryption
- Disadvantage: 1-bit error messes up 64-bits

=> use output feedback mode

Breaking DES

DES concerns:

- Design criteria of S-boxes have never been published
  - are they designed to allow crypt analysis (by insiders) ?
- Short key length of 56 bits
  # keys = $2^{56} \sim 7.6 * 10^{16}$
  - brute force attack becomes possible in less than one day

Double DES solution:

$$C_i = E_{k2}(E_{k1}(P_i))$$

Breaking Double DES

Meet-in-the-middle attack to break Double DES

use: $D_{k2}(C_i) = E_{k1}(P_i)$

Steps (assuming some (C_i, P_i) pairs are known)

- compute $R_a = E_{Ka}(P_i)$ for some i, and all a = 1 .. 2^{56} keys
- compute $S_b = D_{Kb}(C_i)$ for same i, and all b = 1 .. 2^{56} keys
- sort R_a and S_b and search for the matches
- try found pairs of keys on other (C_i, P_i)
- Time complexity = O (2^{57})
  - not much higher than single DES

Triple DES

$C = E_{k1}\, D_{k2}\, E_{k1}\, (P)$

$P = D_{k2}\, E_{k1}\, D_{k2}\, (C)$

- Uses just two keys
  - compatible with Single DES: use $K_1 = K_2$
- No breaking method known to date
- Combine it with chaining

Alternative secret key algorithms

Many other protocols have been made

Most interesting: IDEA (Int. Data Encryption Alg.)

- 128-bit key
- free of NSA 'guidance'
- details in fig 7-10
- 128-bit key is transformed into 52 subkeys of 16 bits each
  - 6 for each of the 8 iterations
  - 4 for the final transformation
  - Decryption similar, but uses different subkeys

Public-Key algorithms

DES has key distribution problem

Alternative: work with two keys, one of which is public

- usually the public key is used for decryption, the private for encryption

Requirements:

- Deriving decryption key from encryption key and (known) cryptographic algorithm is impossible
- Decryption key can not be broken using plaintext approach

Public-key algorithm

Steps:

- Generate pair of keys for A and B
- Register public keys
- Sending P from A --> B : encrypt using B's public key
- Receiving C from A : decrypt using own private key

public keys A_{pub} B_{pub}

private keys A_{priv} B_{priv}

P : A $\xrightarrow{C}$ B

$C = E_{Bpub}\, (P)$

$P = D_{Bpriv}\, (C)$

Public-key Algorithm

Note that we can use same method for <u>authentication</u>

P: A --> B ; now we use A's keys

$C = E_{Apriv}\, (P)$

$P = D_{Apub}\, (C)$

- C can only come from source A
- C cannot be modified (Integrity)

Public-key Algorithm

RSA (Rival, Shamir, Adleman) algorithm (1978 MIT)

Key Generation	
Select p, q	p and q both prime
Calculate $n = p \times q$	
Select integer d	$gcd(\phi(n),d) = 1; 1 < d < \phi(n)$
Calculate e	$e = d^{-1} \bmod \phi(n)$
Public key	$KU = \{e,n\}$
Private key	$KR = \{d,n\}$

f (n) is Euler quotient of n

f (n) = $|\{m \in N \mid m<n \text{ and } gcd(m,n) = 1\}|$

= (p-1) * (q-1) (if n prime)

Public-key Algorithm

Encryption

Plaintext: $M < n$

Ciphertext: $C = M^e \pmod{n}$

Decryption

Ciphertext: C

Plaintext: $M = C^d \pmod{n}$

Example in fig 7-11

Authentication Protocols

Authentication = Verify that sender of message is really the one he says he is

General protocol model between A and B

- Use KDC: key distribution centre
  - KDC is always honest
  - KDC has shared keys with al its clients
  - KDC distributes session keys (K_s) used for direct communication between A and B
- Alternatively A and B may possess already a shared key
  - using this shared key they may establish a session key
- After exchanging several messages A and B should be convinced of each others identity

Authentication based on shared key

A 2-way authentication protocol is shown in fig 7-12

- R_i's are challenges
  - large random numbers to test the others identity
- An intruder T might falsely claim to be A, and successively reply messages
- this does not work because R_B is chosen different each time

Authentication based on Shared key

A shortened 2-way protocol is shown in fig 7-13

- However, this can be broken by the reflection attack (fig. 7-14) if B supports multiple sessions simultaneously

Moral of story:

- Initiator has to prove its identity before responder has to prove his identity
- Use different keys for proof
- Draw challenges from different sets (eg: even and odd sets)

Establishing a shared key

Diffie-Hellman key exchange (1976) : fig 7-15

- A selects two large (public) prime numbers: n and g
  - (n-1)/2 should also be prime
  - certain conditions on g exist
- A and B pick large (say 512 bit) private numbers: x and y
- A sends : (n, g, g^x mod n)
- B replies : (g^y mod n)
- A and B compute key : g^{xy} mod n

No practical algorithm exists for deriving x from g^x mod n

Establishing a shared key

The Diffie-Hellman alg. can be cheated by bucket brigade attack (also called Man-in-the-middle attack) : fig 7-16

- Intruder intercepts all messages
- Intruder sets up separate sessions with A and B

More complex algorithms can defend this attack

Authentication using KDC

Previous methods require key (eg. credit cards) for every 'person' to talk with

Key Distribution Centre solves this

Simple protocol : fig 7-17

- A selects session key and sends : A, K_A (B, K_S) to KDC
- KDC informs B with : K_B (A, K_S)
- Authentication guaranteed because of using keys K_A and K_B

However: intruder can copy and replay messages

Authentication using KDC

Solutions to replay:

- Include sequence # => old sequence # have to be remembered
- Combine sequence # and timestamp
- More elaborate protocol

Elaborate Authentication Protocol

1. Needham-Schroeder protocol (1978) ; fig. 7-18

   weakness:
   - if intruder knows old K_s it can replay step 3 with fresh random number

2. Otway-Rees protocol (1987) ; fig. 7-19
   - uses three random numbers:
     - R_A to challenge B
     - R_B to challenge A
     - R, used by KDC to verify intruder activity

Elaborate Authentication protocol

3. Kerberos protocol ; fig. 7-20

- variant of Needhorn-Schroeder used in real time systems
- assumes synchronized clocks
- uses 3 servers:
  - Authentication server: verifies users during login
  - Ticket server: supplying ID-tickets
  - Transaction server: performing the real work
- password used locally only (to decrypt message 2 with key K_A)
  - user can talk to many servers without needing password again

Authentication using Public-key cryptography

- use public-key system to exchange session key
- use 'easier' encryption (like DES) thereafter

Example: fig 7-21

- man-in-the-middle attack does not work
  - T can intercept E_B (A, R_A) ; it can not decipher it
  - E_B (T, R_x) can be send to B
  - T receives E_T (R_X, R_B, K_S)
  - T sends E_A (R_X, R_B, K_S) to A ; A recognizes wrong random #
  - T can not itself produce message 3

Authentication using Public-key cryptography

Problem: how to distribute public keys ?

- A and B exchange them first before starting the authentication
  - can be tackled by man-in-the-middle attack
  - A sends key to T, T sends its own key to B and vice versa
  - Now T can read all messages from A and B and fake them
- Putting all public keys in database
  - similar attack possible
  - how do A and B know replies come from real database?

Authentication using Public-key cryptography

Solution : Interlock protocol

- Exchange public keys
- A sends half of E_B (A, R_A)
- B sends half of its messages to A
- A sends, after receival, other half of its message
- B sends, after receival , its other half

Trick: intruder T can not decipher half of $E_T(A, R_A)$ and change it into half of $E_B(A, R_A)$

If T sends junk to B, B will discover that, after receiving the second half, the total message makes no sense.

Digital Signatures

Purpose signature:

- receiver can verify identity of sender
- sender can not deny message later on
- receiver can not change the message

Two methods:

- Secret-key signatures
  - uses trusted system (fig 7-22)
- Public-key signatures
  - using eg. RSA (fig 7-23)

Public-key Signatures

- A sends $E_B(D_A(P))$
  E_B public key of B
  D_A private key of A
- B stores $D_A(P)$ as signature

When A denies having sent the message:

- B shows P and $D_A(P)$
- judge checks $E_A D_A(P)$ against P

Public-key Signatures

Problems:

- A (quickly) discloses D_A (and claim it to be stolen)
  - now anyone can produce $D_A(P)$ (in particular B)
- A might have changed keys
  - authority needed recording these changes

New standard: DSS digital signature standard (to be proven yet)

Digital Signatures

Discussed methods so far combine security and signature. Often only signature is required.

- e.g. if message is public or broadcasted
- when encryption takes too much time
- authentication of computer programs

Example: signature protocol using public-key encryption is shown in figure 7-24

Digital signatures

Six signature (message digest) requirements:

- MD can be applied to arbitrary P (of any size)
- MD(P) has fixed length
- MD(P) is easy to compute
- Given MD(P) it is impossible to derive P
  - only required when MD is applied to message + secret key
- Given x it is impossible to find any such that MD(X) = MD(Y)
  - prevent forgery
- It is impossible to find a pair (x,y) with MD(x) = MD(y)
  - prevents birthday attack

First three properties are for practical purposes

Digital Signatures

Signature MD can be a hash function

Example:

- Fragment message in equal sized blocks of 128 bits. Take exor of all fragments

Problem: good measure for data integrity but attacker can make any P with same signature

- take MD(P), and add one fragment to P, such that signature matches

Much more complex hash function: MD5 (Rivest) (details in Stallings. pg.645)

- generates 128-bit signature

Birthday Attack

It looks like it takes 2^m trials to fool an m-bit hash function, but in practice its more like $2^{m/2}$ trials

Q: how many students are needed in a class such that prob. of finding a pair with same birthday > 0.5

A: ??

Solve: $1 - \frac{p!}{p^n (p-n)!} > 0.5$ (for p= 365, n #students)

Intuitively:

With n= 23 there are $n(n-1)/2$ = 253 pairs, each with hit probability = 1/365 ; so a fair hit chance

Birthday Attack

Generaly: for k options ($k=2^m$) we have a fair hit chance if

$$n(n-1)/2 > k$$

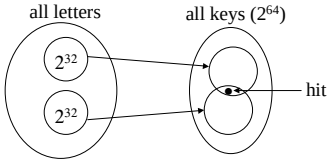
$$\Leftrightarrow n > \sqrt{k} = 2^{m/2}$$

Example

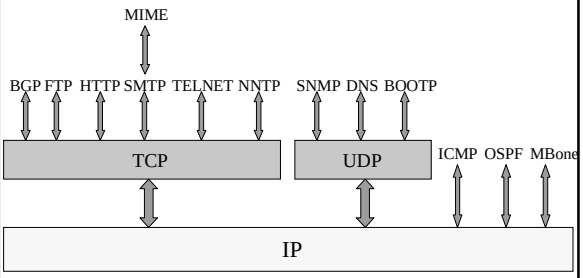
- write bad and good letter with same 64-bit signature
- each letter includes 32 binary options
- compare signatures of 2^{32} bad and 2^{32} good letters in order to find a matching pair.

Birthday Attack

Each letter maps to a particular key K=MD(letter)



TCP / IP Applications



TCP / IP Applications

Acronym	Protocol Name	Information	Section
BGP	Border Gateway Protocol	Exterior Routing	5.5.6
FTP	File Transfer Protocol		
HTTP	HyperText Transfer Protocol	WWW	7.6
SMTP	Simple Mail Transfer Protocol	Email	7.4
- MIME	Multipurpose Internet Mail Ext.		7.4
Telnet		Remote Login	
SNMP	Simple Network Management P		7.3
ICMP	Internet Control Message P		5.5.4
OSPF	Open Shortest Path First	Interior Routing	5.5.5
DNS	Domain Name System	Name resolution	7.2
BOOTP	Boot Protocol	Diskless boot	5.5.4
NNTP	Network News Transfer P	Net News	7.5
MBone	Multimedia Backbone	Video conferencing	7.7.5

Domain Name System

DNS is a hierarchical database used to map host names to IP addresses

- To map a name an application calls a (local) library procedure called resolver
- The resolver sends a UDP packet to a local DNS server
- This server returns, possible after consulting other DNS servers, the IP address to the application

DNS Name Space

- Naming scheme is hierarchical: see domain tree in fig 7-25
  - eg. cs.leidenuniv.nl
  - nl is top level domain
  - leidenuniv is subdomain : represents all the nodes under it
  - leaves of the tree represent domains which have no subdomain
- Generic (like com, edu, gov, org, net) and country (us, nl) top-level domains
- Naming follows organizational boundaries, not physical (network) ones

DNS Name Space

- Getting a name like ITuniv.nl requires registration with the nl *domain manager*
  - after that subdomains can be added freely: cs.ITuniv.nl
- Absolute names (like www.tudelft.nl.) end with period, whereas relatives do not
- Note: names are case insensitive

DNS Database

Every domain name has a set of corresponding ***resource records*** in the database

- For a single host: just record its IP address, but many other records exist
- When a resolver gives name to DNS, it gets back the associated resource records

DNS: resource record

- Format:

Domain-name	Time-to-live	Class	Type	Value

- Domain-name: search key
- Time-to-live: indicates how stable entry is; i.e. the number of seconds (needed for caching)
  - e.g. 86400 seconds (= 1 day)
- Class: always IN for internet information

DNS Database

- Type: specifies the meaning of the value field
  - A: address record specifying IP address in value field
  - MX: specifies email server which is willing to receive email for this domain name
    e.g.: newcompany.com 86400 IN MX 1 mailserver.wi.leidenuniv.nl
    newcompany.com may not have an IP connection itself
  - NS: specifies top-level name server
  - CNAME: alias (actually a macro)
  - HINFO and TXT: comments
- Value interpretation depends on Type: contains #, name, or ASCI string (see fig 7-26 / 7-27)

DNS Name Server

The DNS name space is split into non overlapping zones, as introduced in fig 7-28

- A zone usually has 1 primary name server, and multiple secondary ones, which get their information from the primary one.
  - note, cs.yale.edu and yale.edu have separate name servers
- A name server may return an authoritative record or a cached entry
  - authoritative record always correct
  - cached entry has timeout (Time_to_live field)

DNS Name Server

- A remote query is send to top-level domain server of requested name
  - from there the query goes recursively to the right zone server
  - see fig 7-29 on how a recursive query is handled
  - not all servers support recursive query; in that case the server just returns the name of the next server to contact directly by the application

World Wide Web

- Web is framework for accessing linked documents
  - Web servers provide documents (pages)
  - Web browsers view the documents
  - Pages are linked using hypertext techniques
- Web idea comes from CERN, Geneva (1989)
- WWW consortium was set up by CERN and MIT (1994) to organize and standardize further development
  - see their homepage http://www.w3.org

WWW Client Side

- Several browsers are available (e.g. Netscape, Explorer)
- Pages may include text, audio, pictures, video
  - to view these sometimes helper applications are started
- Smart image loading (get text first; then coarse grain image)
- Active pages
  - forms and active clickable maps
- Local caching of pages
- TCP connection support required (connecting to port 80)

Uniform Resource Locator: URL

WWW uses ULRs to locate pages (see fig 7-59)
URL contains 3 parts:

- protocol name:
  - eg. HTTP, FTP, news, mailto, or file; see fig 7-63
- server name, eg. : cs.et.tudelft.nl
- page (file) name, eg. MOVE

So we get eg. HTTP://cs.et.tudelft.nl/MOVE
Protocol steps:

- browser determines URL
- browser asks DNS to map server name to IP address
- browser makes TCP connection to IP adrress, port 80
- browser sends GET 'URL' command
- server sends page (file) MOVE.html

Uniform Resource Locator: URL

URL problem: combines location and page name => pages can not move!

- Note: access of each image requires new TCP connection

WWW Server Side

Hypertext transfer Protocol : HTTP

- HTTP is ASCII based protocol (see fig. 7-60)
- Two types of requests:
  - without or with protocol version
  - e.g.: *GET /hypertext/WWW/pages.html HTTP/1.0*

Several built-in HTTP request methods: fig 7-62

GET command can be followed by If-modified-Since header

- server only sends page if modified recently, otherwise local cached page will be used

WWW Pages

- Pages are written in HTML: HyperText Markup Language
- Not WYSIWYG
  - Markup commands are not hidden
  - of course, a page editor can be built on top of HTML
- Forms (see fig 7-67)
  - can be submitted in three ways:
    - submit button
    - clicking on active map
    - typing ENTER on one-item TEXT form
- The Action parameter can specify an executable (or script) which will be invoked

Active WWW pages and JAVA

CGI : Common Gateway Interface

- ASCII script is specified as URL
- When the script is invoked (either by clicking on the link, or via the FORM Action mechanism) it starts executing, and is assumed to return an HTML page on standard out, which is returned to browser
- Scripts run remote

JAVA applets:

- Downloaded and then run locally
- Allow rapid interaction
- Applets are written in JAVA bytecode and interpreted
- Interpreter has to guarantee security

Exercises

Make from chapter 7 the following exercises:

- 4, 6, 8, 10, 11, 17, 28, 41, 44, 52