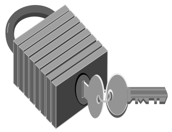
# *Security*

- Threat Analysis
- Military vs. Commercial Security
- Access Control
- Authentication
- Digests and Signatures



# What is Security

- Security is the set of mechanisms and techniques used to protect a system and prevent unauthorised access to resources or disclosure of information.

Security Services include:

***Access Control:*** control over who has access to services or resources within the system

***Confidentiality:*** prevent disclosure of information to unauthorised users + prevent analysis of traffic characteristics.

***Authentication:*** validity of the identity of sender or server.

***Integrity:*** prevent modification of information by unauthorised users – includes no duplication, replays, insertions or reordering.

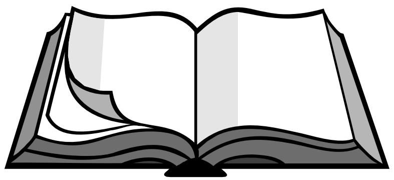
***Non repudiation:*** prevent denial of sending or receipt of a message

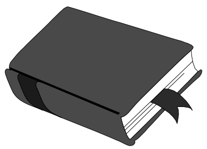
***Availability:*** prevent denial of service e.g. by disruption



# Security References

- Network Security Essentials 2nd ed. W. Stallings Prentice Hall 2003.
- Crytography and Network Security 3rd ed., W. Stallings 2003
- Coulouris et al Distributed Systems 4th ed. Ch. 7
- D. Gollmann, Computer Security, Wiley, 1999

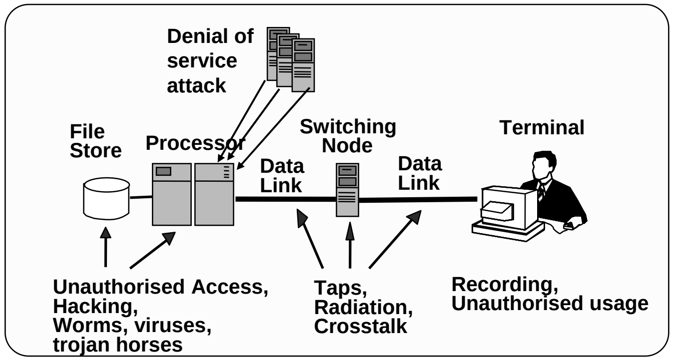






# Threats: Identifying Potential Attacks

- **Identify:**
  - What security breaches can occur
  - Where they can occur
  - How they occur



# Threat Analysis 1

- Information or Resources
  - Theft / copying, disclosure
  - Modification, corruption or fabrication
  - Destruction
- Services
  - Unauthorised utilisation of resources
  - Disruption of service
  - Denial of access to authorised users
- Users
  - Abuse of privilege by legitimate user
  - Masquerading – impersonation of the identity of another authorised user.

*Large scale networks (and hence distributed systems) cannot be made physically secure.*



# Threat Analysis 2

- **Passive Attack**
  - Observe information in network without interference
  - Message content – break confidentiality
  - Message traffic analysis – frequency, length, source, destination
    Could have military significance
- **Active Attack**
  - Modify message contents or message stream
  - Delete, delay, reorder, replay, insert valid or invalid messages
  - Masquerade as authorised user
  - Denial of service by flooding servers with valid requests
  - Passwords gained through passive attack can be used for active attack



# Threat Analysis 3

**Require:**

- Vulnerability Analysis: identify potential weak elements within system – What is critical to the organisation?
- Threat Assessment: likelihood of a threat occurring which exploits the vulnerability detected
- Risk Analysis: analyse the potential consequences of problems arising from security breach + estimate cost of a successful attack
  e.g. loss of revenue.
- Prevention techniques: what can be done to prevent security breaches and what are their cost?
- Cost benefit analysis: do the consequences of security breaches justify the cost of protection?
- If security controls cause too much inconvenience or loss in performance they will be bypassed.
- Recovery: may be less costly than prevention??



# Security Threats 1

- **Virus**
  - Incorporates a copy of itself into other programs and performs a malicious function when the program is invoked.
  - Virus detection and protection software is available – not foolproof.
- **Trojan Horse**
  - Performs a useful function, but contains hidden code to perform a malicious function e.g. via compiler. Often produced by in-house programmer.
  - May be triggered at a particular time or particular conditions
    e.g. programmer no longer on payroll – “Logic Bomb”
  - Require management of development process – code inspection
- **Worm**
  - Invades a workstation via the network, using a security loophole (e.g. in remote login or Email) and implants a virus or Trojan horse.
  - Require monitoring to prevent, and patching known security bugs

# Security Threats 2

- **Unattended Terminal**
  - Passer-by can gain access to resources accessible by user
  - Solution – forced log off after timeout
    - Terminal freezing, random or after timeout – released by password
    - Educate users not to leave terminals logged in
- **Man in the Middle attack**
  - Relays all interactions between client and server pretending to be the client to the server, and the server to the client.
  - Substitutes his own encryption keys for that of the server.
  - Make sure DNS server is not compromised and only accept certified encryption keys
- **Hacker Detection**
  - Statistical anomaly
  - Rule based expert systems



# Password Cracking

- Techniques
  - Try default passwords for standard accounts shipped with system – may not have been changed.
  - Password guessing – copy password file; use known system encryption algorithm with dictionary, names of users' family, phone numbers, sports etc; compare with stored encrypted password
  - Line tapping to capture passwords sent unencrypted or above methods with encrypted passwords
  - Trojan horse e.g. in login program to capture passwords
- Solution
  - Controlled format: $\geq$ 6 characters, Mixed upper and lower case + some non-alphabetic characters.
    No dictionary words or names.
  - Regular change by user ➔ validity time
  - One way encrypted storage, with date and time of last change also encrypted.



# Worst Case Assumptions & Guidelines

- Interfaces are exposed – an attacker can send message to any interface
- Networks are insecure – message can be monitored, altered spoofed etc
- Limit lifetime and scope of secret – time limit and restrict sharing of secrets
- Algorithms and code are available to attackers – publish and rely on secrecy of cryptographic keys ➔ third party scrutiny
- Attackers have access to powerful resources – computation time is not a reliable protection mechanism
- Minimize trusted base – parts of system implementing security + h/w and s/w on which they depend



# Security Policies

**Mandatory Policy**

The security rules which are built into the system and cannot be changed by users i.e. they cannot be trusted to enforce security policy.
E.g. no information from a high security level can flow to a low security level.
Subjects labelled with clearances defining their privileges
Targets labelled with classification defining sensitivity

**Discretionary Policy**


The security rules which a user or manager may choose to apply.
E.g. who can access your personal files.
Typically defined by identity based rules where access decided on identity of subject

# Military Security Policy

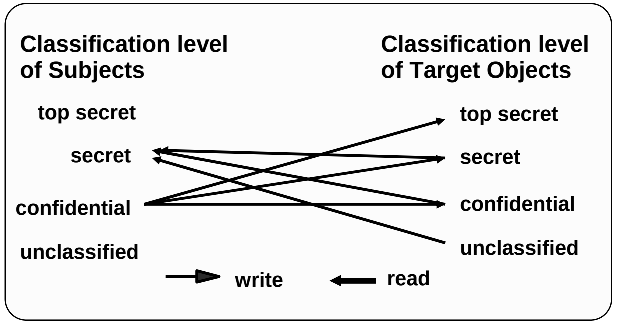
- Based on Security level classifications (labels)
- Mandatory labelling of all documents with classification level
- Regulate control of classified information and prevent access by unauthorised people
  - ➔ emphasis on confidentiality & controlling flow of information
- Mandatory controls constrain user so that any action taken conforms to security policy – very little flexibility
- Discretionary aspect: user can further restrict access, but cannot re-label to a lower classification.
- Authority vested in officer hierarchy

Rigid security "compartments"



# Bell-Lapadula Security Model

- Typical military security levels:
  unclassified, confidential, secret, top secret
- Categories of information e.g. nuclear, naval, planes
- Subject Label (clearance) =
  (Level {categories}) e.g. secret {nuclear, planes}
- Target Label = (level {1 category})




Write up
Read down



# Bell-Lapadula Security Model

**1) Simple Security condition**

Subject s may only read from target t

- if (s.level $\geq$ t.level) & (t.category $\in$ {s. categories})
- i.e. read information of permitted category at same or lower classification eg officer cleared for secret cannot read top secret, but can read all lower classified documents

2) * Property

- Subject s can write to a target t
  - if (s.level $\leq$ t.level of t) & (t.category $\in$ {s. categories})
  - ie. write to documents at higher classification
    ➔ prevents copying information from higher to lower levels
  - Can only have read/write access to documents at same level
- ***Anomaly ?***



# Commercial Security Policy

- More emphasis on integrity of data to prevent fraud and errors – making sure people authorised for some access do not perform unauthorised operations.
- Well formed transactions – no arbitrary manipulation of data, only predefined, constrained operations – not just read & write.
- Auditability – log all data accesses so that actions can be audited later c.f. writing in ink and making correcting entries rather than erasing
- Accountability – log which subject performed a transaction, + when and where it occurred
- Separation of responsibility e.g. transaction approved by one person but carried out by another cf. 2 keys to a safe.
- Security administrator is responsible for giving access rights.
- More emphasis on discretionary policies to model
  - Authority – legitimate power to perform actions
  - Delegation of authority within organisational managerial structure

➔ Controlled sharing of resources

# Access Control

- Ensure that the operations which users or processes can perform on computer resources are done in a controlled and authorised manner.

- Needed for
  - confidentiality: protect resources from being read
  - integrity: protect resources from being modified
  - availability: prevent denial of authorised access

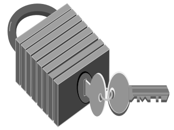


# Access Control

**Physical Access Control**

- Secure computer rooms – no windows, locked doors, guards etc.
- Secure terminals – within same building as computer,
- Dedicated terminal to users
- Lock up PCs and discs
- No remote access

**Insufficient for networked or distributed systems**



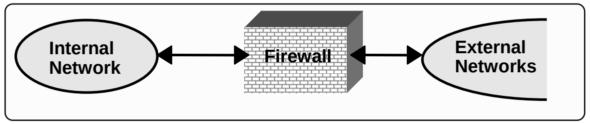
**Logical Access Control**

- Protection of resources from users who have already gained physical access to the computer system
  - Identification – who are you?
  - Authentication – are you who you say you are you?
  - Authorisation – are you permitted access?
- Essential for multi-user and distributed systems



# Firewalls

A security gateway between internal and external networks




- Based on packet filters – user defined filtering rules for both incoming and outgoing messages.
- Filtering criteria
  - **Address** – only permit access from selected sites or hosts e.g. remote sites of the same organisation, or collaborators.
    Could be based on combinations of IP port address
  - **Message type** – permit incoming Email & HTML (Web) messages but prevent Telnet or FTP



# Example Firewall Rules

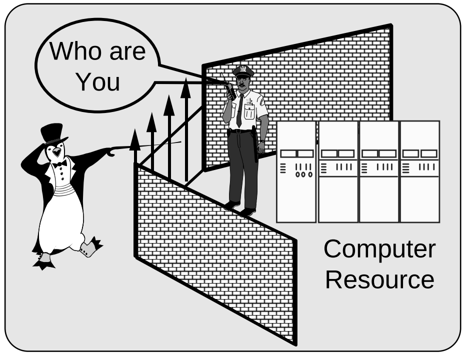
| Rule | Action | In/ Out | Source IP | Source port | Dest IP | Dest Port | Description |
|---|---|---|---|---|---|---|---|
| 1 | allow | in | 8.2.3.1 - 8.2.3.255 | * | * | * | Trust this set of hosts |
| 2 | block | in | 8.2.3.177 | * | * | 513 | Block from this specific source from login access |
| 3 | allow | out | * | * | * | 80 | Allow access to remote web servers |
| 4 | allow | in | 8.5.3.2 | 3024 | 6.5.2.1 | 1500 | Very specific rule |
| 10 | block | * | * | * | * | * | Default blocking of all other in or out traffic |

- Top down evaluation
- Default policy must be last
- *What is wrong in the above specification?*

# Identification

**Establishing Identity of Subject**

**Identification can use:**
- Name or ID provided by user
- Workstation address
- Magnetic card e.g. bank card
- Smart card

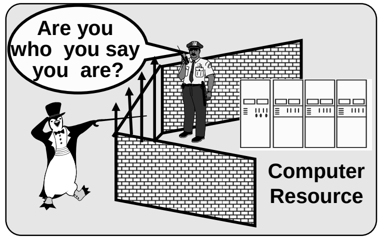



**Recommendations:**
- Use name/initials not numbers
- Same ID for ALL systems & email
- Individual accountability ➔ unique ID for each user
  Identify users in logs and audit trails
- No sharing of ID



# Authentication

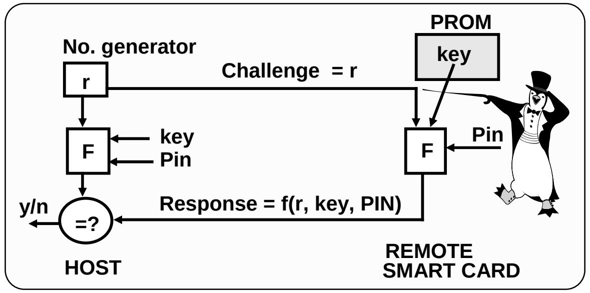
Personal Authentication
= Verification of identity of subject
- Personal identification number (PIN) for magnetic or smart cards
- Passwords
  - Controlled format: ≥ 6 characters, Mixed upper and lower case + some non-alphabetic characters.
  - No dictionary words – prevent dictionary based hacking.
  - Regular change by user ➔ validity time
  - One way encrypted storage, with date and time of last change also encrypted.
  - Password list: use password once and discard (e.g. SWIFT) Distribute password lists in 2 halves.
- Retina or Fingerprint scanning
- Maximum number of authentication attempts (e.g.3)
- Logging and investigation of all authentication failures






# Remote Access Authentication

## Smart Cards
- Embedded microprocessor + storage
- Preprogrammed with Key in PROM
- Require possession + knowledge of PIN




Challenge / Response Protocol

Host generates random number r which is sent to Smart Card. Card uses one way function F, seeded with cryptographic key K and PIN number to calculate response which is sent back to host. Host uses same function to check response.
Not susceptible to replay



# Trust Issues

- Users may run OS which subverts security mechanisms
- Easy to connect unauthorised machine to LAN
- Workstations have ***promiscuous*** mode where they listen to all messages on a LAN ➔ easy tapping of communications.
- Communications with remote servers should NOT be trusted
- Need to authenticate users to servers for access to personal resources (files), accounting etc.
- Need to authenticate servers to users so that they can be sure it is not a malicious user masquerading as a server.
- Where should users be authenticated?
  - When logging into own workstation – but servers will not trust it.
  - Re-authenticate for each server used – inconvenient if many servers are used.

➔ Use trusted Authentication Service

# Authorisation

Specification of security policy in terms of what resources a subject can access and what operations can be performed upon them.

- Authorisation should be in terms of user roles rather than user identities i.e. Manager of Personnel rather than Joe Bloggs
  ➔ simpler when people move jobs.
- Default to no access i.e. require explicit authorisation to permit access
- ***Principle of least privilege***
  Any subject or user object should be given the minimum access rights required to permit it to carry out its assigned task. This principle is violated if a server has to take on the user's identity in order to perform a function on its behalf.



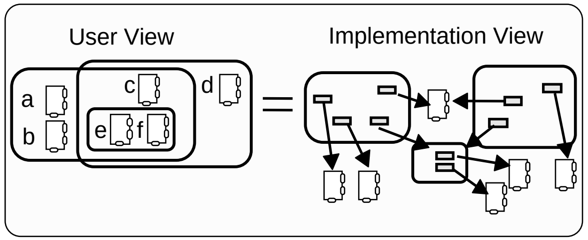
# Access Matrix

| User/Subject \ object | F1 | F2 | Printer | O1 | O2 |
|---|---|---|---|---|---|
| mss | read, write | | print | a, b c | i, j |
| jnd | read | execute | print | b, c | k |
| jk | | read | print | c | i, j, k |
| xyz | append | read, write | | a, b | i |



# Domains and Policies

- Impractical to specify policy for individual objects in a large distributed systems with millions of objects
- Use domains to group objects for specifying policy c.f. directory
- Reflect organisational, geographical or network structure
- Policy propagates to subdomains
- Policy about an object is determined from the set of domains of which it is a member




Domain holds a set of object references:
- Local name for object
- Object Identifier (OID)
  = Unique identifier
  + Address or
  distinguished name

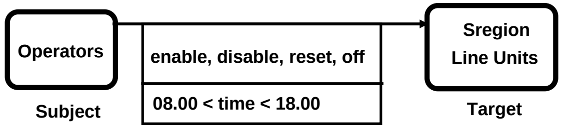


# Authorisation Policy

- Defines what activities a subject is permitted or not permitted (prohibited) to do to a target.
- Specify:
  - Modality – permit or forbid
  - Subject Domain e.g. section, department or manager
  - Target Domain e.g. directory
  - Set of permitted operations
  - Constraints – time of day, day of week,
    or a predicate on the value of a subject or target attribute
    eg location of subject.

# Authorisation Policy

## Examples




**inst auth+** { **subjec**t Operators; **target** <line_unit> Sregion;
  **action** enable, disable, reset, off;
  **when** time.between ( “08.00”, “18.00”) ; }
**inst auth- {subjec**t students; **action** reboot**;**
  **target** teaching_workstations ; }
**inst auth+** { **subjec**t s= managers; **action** read**, write;**
  **target** development_directory ;
  **when** s.location = planning office; }

**Implementation**

- Access Control List (ACL) attached to target domain.
  cf. Column of Access Matrix
- Set of Access Tickets or capabilities held by user domain
  c.f. row of Access Matrix.



# Access Control List (ACL)

**An ACL is a list of {user domain, permitted operations} held with the target domain or target object.**

- In many current implementations, there is no real concept of domain – only individual users or classes of users.
  The ACLs are held with servers.
- Unix only 3 user domains – owner, group, others
  Operations are read, write and execute
  A file can be set to switch user ID to that of owner when it is executed ➔ increase privileges when using system programs.
  - Unix approach gives very coarse ACL
    Cannot easily specify access to individual operations on an object



# Access Control List (ACL)

- windows has more flexible subject groups (domains) and can specify access to up to 32 different operations on an object.

| | | | | | | |
|---|---|---|---|---|---|---|
| Owner | r | w | x | opa | opb | opc |
| GroupA | - | - | - | opa | - | opc |
| GroupB | r | - | - | opa | - | - |
| fred | r | - | x | - | - | - |
| World | - | - | - | - | - | opc |



# Capabilities

- A capability is a token or ticket which refers to a particular object and specifies what operations can be performed on the object. A capability is like a protected name for an object. Possession of the capability permits a subject to perform the specified operations on the object. The capability can be passed in parameters or messages.

| Object Reference | | | Operations | Check Field |
|---|---|---|---|---|
| Server Address | Object Type | Object Instance | | |
| 64 | 16 | 16 | 32 | 64 bits |

- The check field is an encrypted version of the permitted operations field and a suitable random constant. It protects the capability and prevents it being forged.

# Capabilities

- A server or manager issues capabilities for the objects it controls. When the object is created the random number is generated and is stored in an internal reference table to the object. When an access request is received, the object field is used as an index into the table. Decrypting the check field must yield the correct constant for the request to be permitted.
- Creation of capabilities, copying or modification of access rights are protected operations of the OS. The operations field will indicate which of these operations is permitted on the capability itself. Modifying the operations field (usually to reduce access rights, before passing on the capability) can be performed by requesting the originator to issue a new one or by special commutative one way functions.
- Capabilities permit very limited access rights to be passed to other user.
- Checking capabilities can be more efficient than ACLs so are used when checking is needed on a per operation basis.
- Cannot easily check who has access to a resource

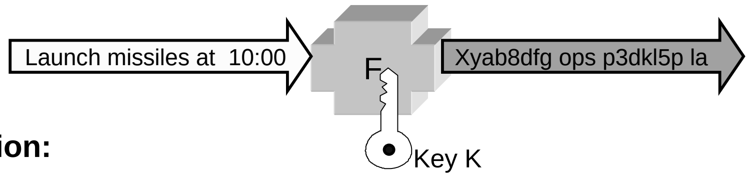


# Revocation of Capabilities

- Hard to revoke access rights
  Capabilities are distributed throughout the system – must be found and be destroyed
- **Expiry Time:** capabilities expire after a time and new capability must be requested – this is refused if rights have been revoked.
- **Keys:** capability contains encrypted key, checked by object. Change key in object to revoke capability.
  No selective revocation.



# Cryptography



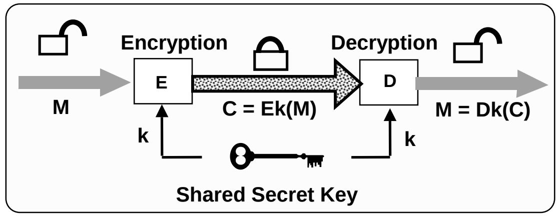

**Encryption:**
Transformation of information based on:
- Substitution e.g. table look up
- Transposition e.g. exchange bytes 1 & 3, 2 & 4 etc.
- Combine with a key which specifies what substitution or transposition to use
  - ➔ Encryption *Function E + Key k*

**Decryption:**
- Inverse of encryption to obtain original information
- Computation time required to decrypt without the key makes it impractical but not impossible.



# Secret Key Cryptography

**Single Secret Key**




- Basis for Data Encryption Standard (DES)
- Same algorithm applied for Encryption and Decryption
  i.e. E = D
- 56 bit key applied to blocks of 64 bits of data – easily cracked
- Advanced Encryption Standard (AES) selected in March 2001 after public competition: Rijndael from Belgium
- 128, 192 or 256 bit keys
- Problems of key management

# Key Management & Escrow

**Key Management**

- Secret key required for each partner or even session
  ➔ Key distribution problem
- Hierarchical Key Structure
  - Distribute master key by courier e.g. monthly
  - Use master key to encrypt new keys sent out e.g. daily
  - Use current day key to encrypt session key.
- Use session key distribution server e.g. Kerberos

**Key Escrow**

US Govt designed a chip for en/decryption called clipper
Secret keys should be held (in 2 halves) by "trusted" authorities so that they can tap conversations & messages between criminal, terrorists ( and political opponents??)
Very controversial

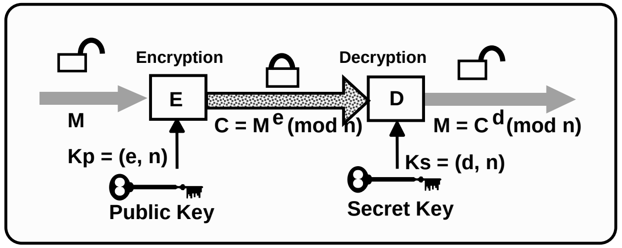


# Public Key Encryption

- **Rivest, Shamir, Adleman (RSA) Algorithm**
  - Use two keys:
    - *Public Key* Kp sent out over network and stored with name servers – used by sender for encryption
    - *Secret Key* Ks used by recipient for decryption
    - Cannot deduce secret key from public key
  - DKs(EKp(M)) = M i.e. decryption of encrypted message yields the original message – symmetric encryption & decryption
- **Key Generation**
  1 Choose 2 prime numbers p, q > $10^{100}$
  2 n = p x q
  3 choose prime number d
  which has no common factor with (p-1) x (q-1)
  4 e x d = 1 (mod ((p-1) x (q-1)))
  Kp = (e, n) Ks = (d, n)



# Public Key Encryption

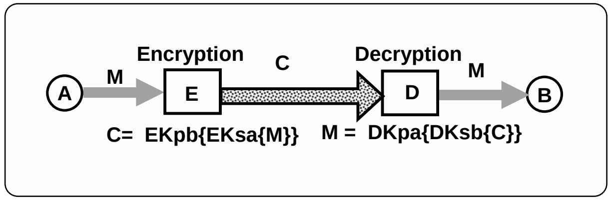



- Simple Key management - no secret keys to distribute
- Computation intensive ➔ poor performance

➔ Use public key system for distribution of secret session keys, and session keys for encrypting messages



# Public Key Signatures




- Use public key encryption where:
  DKp(EKs(M)) = M and DKs(EKp(M)) = M
- A uses a secret key to encrypt a message
- B receives the message and decrypts it with A's public key.
- If the decryption is successful it must have been encrypted by A, i.e. "signed" by A ➔ authenticates sender.
- *How can you tell it was successfully decrypted?*
- Can be decrypted by anyone with A's public key
  ➔ Encrypt with B's public key – only B can decrypt.

# Public Key Signatures

- No verification of time or prevention of replay:

  X could capture message and repeat it e.g. if message was from point of sale terminal in Company X to A's Bank to transfer £1000 to X's account.
- A can pretend secret key Ksa was stolen and deny sending message.
- Need to store complete encrypted message for later verification.

- *What is needed?*



# Message Digest

- A one way function on a message ➔ a code which can be used to check its integrity e.g. checksum or hash function.
- Detects message modification but does not provide secrecy.
- Sender computes digest, appends to message and receiver re-computes to check
  - Given message m it is easy to compute H(m)
  - Given H(m) it is impossible to compute m
  - No 2 messages can generate same H(m)

- Use cryptographic hashing or encrypt digest.
- Hashing is faster than encryption eg MD5 or Secure Hashing Algorithm (SHA)
- Pad message to multiple of 512 bits & mangle blocks of 512 bits to produce 128 bit (MD5) or 160 bit code (SHA).



# Certification Problem

Alice has generated a new electronic work of art in the form of a large 50MB file. She wants to be able to prove that she is the originator of the work so requires a certificate from a certification authority (CA).

Using public key cryptography and message digests show

1) what information Alice sends to the CA

2) the response from the CA.

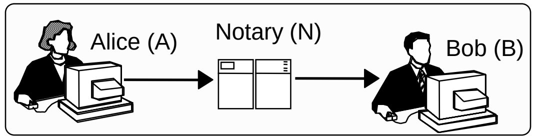


# Digital Signature

- Needed for legally binding documents.
  E.g. Joe sends message to stockbroker to buy 10,000 shares in HAL Plc which goes bust next day.
- Joe may deny sending message or stockbroker may change number to 100,000 to offload other shares he had bought for himself.
  - Verify author, date and time of signature
    ➔ non-repudiation of origin
  - Verify contents at time of signature
    ➔ receiver or someone else cannot modify contents or forge message claiming it came from sender.
  - Signature must be verifiable by third party to resolve dispute.
  - Signature must be unforgeable
  - Must be practical to retain copy of digital signature in storage.

- Need Notarisation Service (Arbiter) to prevent non-repudiation by sender or subsequent claims of loss of secret keys.

# Notarised Signatures




- Assume source name and address is included in every message – *why?*

A → N: A, B, Kab{m}, Kan{ A, H(Kab{m}), Ta}

- Message m is protected by Kab known only to Alice and Bob so cannot be read by Notary. Ta is A's timestamp.
- Notary validates encrypted message by checking hash digest.
- Notary inserts its timestamp Tn in message.

N → B: N, Kbn{ A, Ta,Tn}, Kab{m}, Kan{ A, Ta, Tn, H(Kab{m}) }

- Bob knows timestamp of generation and signing so can check for freshness.
- The notary's signature Kan{ A, Ta, Tn, H(Kab{m}) } is stored – Bob cannot decrypt it.
  Only m need be stored as Kab{m} can be recreated.



# Notarised Signatures

- In case of dispute Bob sends the following to the Notary

  B → N: B, Kbn { A , Ta, Kab{m}}, Kan{ A, Ta, Tn, H(Kab{m}) }

  - The notary can then verify the time, the sender and that the digest of the encrypted message is valid so m has not been changed.

- Proof of receipt

  B → N: B, Kbn{ A, H(Kab{m}), Tb}

  - N holds the original message until the receipt is received and checks the digest H (Kab{m} ), adds it's timestamp and B's identity, then forwards the receipt to A.

  N → A: B, Kbn{ A, B, H(Kab{m}), Tb, Tn}

  - A holds the receipt which it cannot decode or modify, but could be sent to N in case of dispute.



# X 509 Certificates

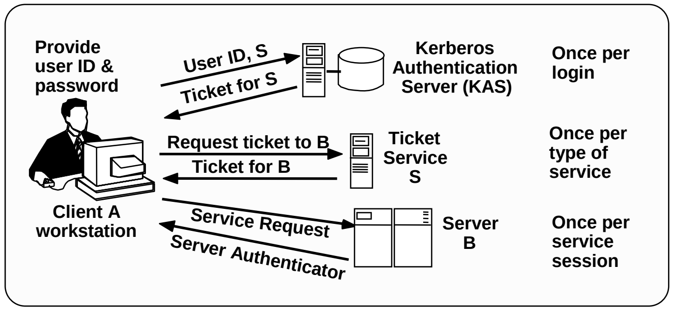
- Associate public key with user or service
- Signed by a certification authority (CA)
- Not forgeable – can be stored in a directory
- Issuer can revoke specific certificates – hold revocation list.
  - Version of certificate format
  - Serial number – unique within CA
  - Signature algorithm
  - Issuer name + unique identifier
  - Period of validity
  - Subject name + unique identifier – to whom the certificate was issued
  - Extension fields for additional info about subject or issuer eg public key to be used to verify signature
  - Signature – hash code of all other fields encrypted with issuer's private key.



# Kerberos Authentication Service (V4)

Separate authentication and ticket granting service

Users and Service providers must register with Kerberos



# Kerberos Authentication Service (V4)

- KAS stores identities, server private keys and encrypted user passwords used to generate private user keys
  Must be secure + master/slave redundancy
- Ticket service can be replicated – does not hold secret data
- Use time stamps to detect replay
- Realm (c.f. domain) managed by autonomous administration & has 1 or more authentication servers.
- Principal name = (primary name, instance, realm)
- Lifetime of tickets issued by ticket server must be less than the lifetime of ticket originally issued by Kerberos
- Servers accept tickets within limited window of timestamp to prevent replay or masquerading.
- Require clock synchronisation (within minutes)



# Tickets and Authentication

**Ticket for Server X**

Secure transfer of authenticated identity of user to server, plus optional authorisation data that can be used for access control.

***Tax*** = Kx {Aname, Arealm, Aadr, Kax, TS, TL, Xname}

TS = ticket timestamp

TL = ticket lifetime

Can be reused

Generated by KAS for ticket service, or by ticket service for Server B

**Authenticator for Server X**

Proves identity of user presenting ticket is same as that to whom ticket was issued.

***Xax*** = Kax{Aname, Arealm, Appl. data, TS}

Can only be used once when session with B is established
Generated by client A



# Kerberos Interactions

- A logs in and provides userid & password.
  Workstation uses password to generate Ka then discards password.
  - A → KAS: Options, Aname, Arealm, Sname, TS
    - KAS generates session key Kas & ticket Tas for ticket server S
    - KAS uses A's password to generate Key Ka
  - KAS → A: Ka{Kas,TS2, TL, ***Tas***}
    - Only client A can decrypt this to obtain ticket ***Tas***
    - Use Kas to create authenticator ***Xas*** for S
  - A → S: Bname, ***Tas, Xas***
    - Decrypt ***Tas*** to get session key Kas. Check authenticator & ticket.
    - Generate session key Kab and ticket ***Tab***
  - S → A: Kas{ Kab, Bname, TS, *Tab*}
    - Decrypt to get ticket and session key for B. Create authenticator for B
  - A → B**: *Tab, Xab***
    - Decrypt ticket to get session key, check authenticator, generate response by adding 1 to TS in authenticator. This authenticates Server B to A.
  - B → A**:** Kab{TS +1}. Only server B can generate this response.



# Kerberos (Contd.)

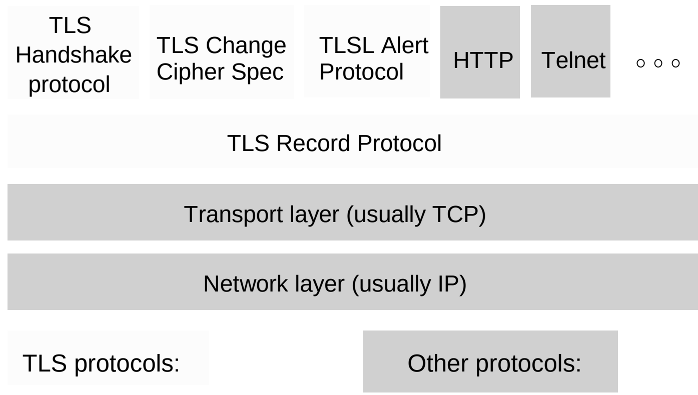
- Kerberos database can be replicated with master and multiple slaves kept in physically secure place.
- Manager updating (master) database must be authenticated by KAS i.e. get ticket from KAS not ticket service. Master propagates changes to slaves.
- **Multiple Realms**
  - Remote realm ticket service (TS) must be registered with local TS.
  - Local TS gives tickets to remote TS which then gives ticket to remote server.
  - →does not scale to large numbers of realms
- **Kerberos V5**
  - Tickets have initial authentication time, ticket start-time, end time and renew until time.
  - Better inter-realm authentication
  - Authenticators can include authorisation data

# Case Study: Secure Socket Layer SSL

- Originally developed by Netscape now IETF standard Transport Layer Security TLS
- Negotiable encryption and authentication algorithms
  - Key exchange method eg RSA & public key certificates
  - X509 certificates for authentication
  - Cipher for data transfer eg DES or IDEA
  - Message digest for creating authentication codes (MACs) eg SHA
- Bootstrapped secure communication
  - Unencrypted initial exchanges
  - Public key encryption used to exchange data to generate shared secret keys
  - Secret keys used as session keys for encryption
- Usually above TCP/IP

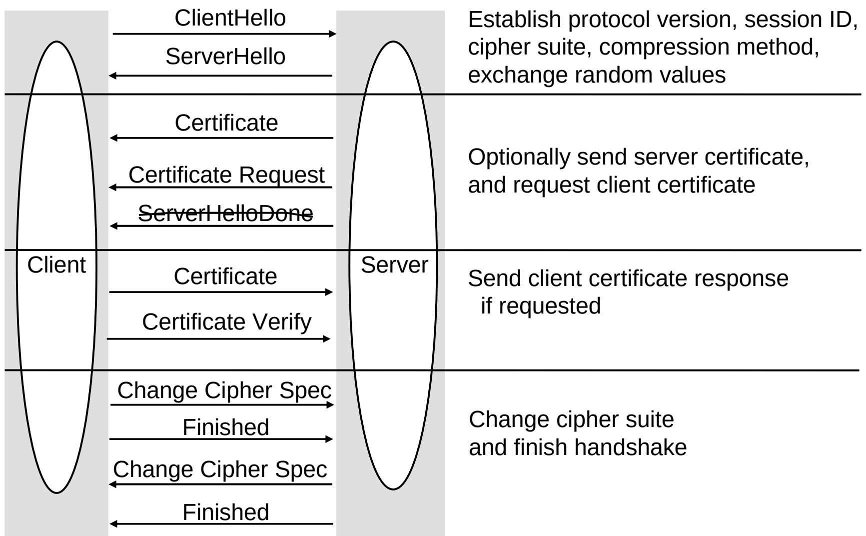


# TLS Architecture






# TLS handshake protocol

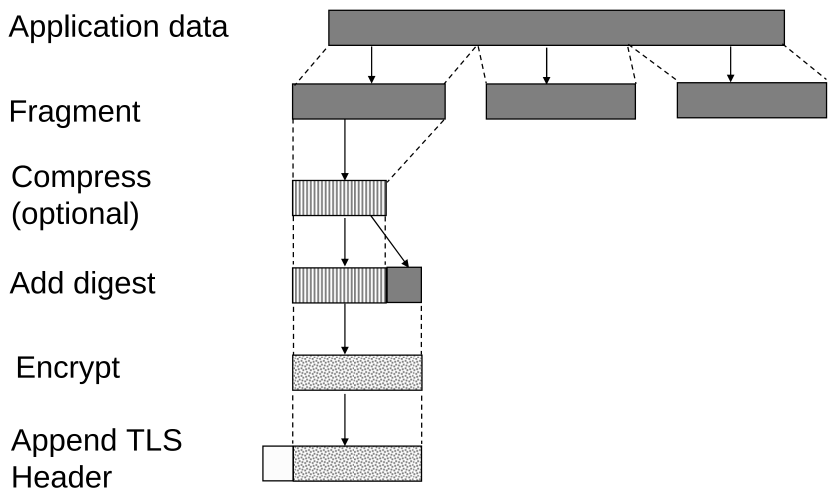





# Handshake Protocol

- Server offers a selection of well known cipher suites, from which client selects one.
- Exchange X509 public keys & possibly others for encryption, if X509 can only be used for signing.
  issuer, subject, subject's public key,
  issuer's signature (encrypted hash of other fields)
- Client generates a 48 byte premaster secret which it sends encrypted to server encrypted with server's public key. Used to generate different session write keys for client and server for encrypting transmissions and message authentication secrets.
- Note: TLS authentication is not strong – susceptible to "*man in the middle*" attack – public key used to verify first certificate often preloaded.

# SSL Record Protocol






# Certification Solution

Alice → CA : Alice, $K_{as}${Alice, H(file)}

Alice sends a digest of the file to obtain the digital signature, encrypted with her secret key. CA uses Alice's public key to decrypt and prove it came from Alice

CA → Alice : $K_{cs}${Alice, H(file), t}

Digital signature includes Alice's ID and time stamp of when it was generated. This can be check with the CA's public key.



# Summary

- Physical security impossible in Distributed Systems
- Networks inherently less secure than centralised computer systems as they are more susceptible to wire tapping etc.
- Cannot trust distributed workstations
- Dependence on distributed systems and value of information they manipulate make security an essential aspect
- **Require:**
  - Authentication
  - Access Control
  - Data confidentiality and integrity Service
  - Notarisation Service
  - Notation for specification of Authorisation policy
- Encryption is an important mechanism for these services.
- Security features attract "hackers" to try to break them
- Commonsense in choice of passwords and careful management may be adequate for many non-military applications
- May be able to isolate critical components from network or use firewalls