

---

## DISTRIBUTED SYSTEMS (COMP9243)

### Lecture 8: Security

#### Slide 1

- ① Introduction
- ② Cryptography
- ③ Secure communication
- ④ Authentication
- ⑤ Authorisation

---

### SECURITY IN DISTRIBUTED SYSTEMS

Related to dependability:

**Confidentiality:** information disclosed/services provided only to authorised parties

**Integrity:** alterations can only be made in an authorised way

**Availability:** system is ready to be used by authorised parties

#### Slide 2

Two aspects:

- Data security
  - Communication security
  - Protection
- System security

Protect the services and data that a system offers against security threats

---

---

### SECURITY POLICY

Security is a question of tradeoffs

Security Policy:

- A statement of security requirements
- Describes which actions entities in a system are allowed to take and which ones are prohibited
  - Entities: users, services, data, machines, etc.

Example:

- Everyone (staff and students) has an account
  - Access to course accounts must be approved
  - Only course accounts can modify grades
- 

### SECURITY THREATS

**Interception:** unauthorised party has gained access to a service or data

**Interruption:** service or data become unavailable, unusable, destroyed, etc.

#### Slide 4

**Modification:** unauthorised changing of data or tampering with a service (so that it no longer adheres to its specifications)

**Fabrication:** additional data or activity are generated that would normally not exist

---

---

## ATTACKS

Passive vs Active Attacks:

**Passive:** observe data without altering it

**Active:** alter data

**Slide 5** Attacking the Communication Channel:

- Eavesdropping
- Masquerading
- Message tampering
  - Man-in-the-middle
- Replay
- Denial of service

---

## SECURITY MECHANISMS

Good Mechanisms:

**Encryption:** transform data into something an attacker cannot understand

- Slide 6**
- A means to implement confidentiality
  - Support for integrity checks (check if data has been modified)

**Authentication:** verify the claimed identity of an entity

**Authorisation:** determine what actions an authenticated entity is authorised to perform

**Auditing:** trace which entities access what

Less Good Mechanisms:

**Slide 7** **Obscurity:** count on system details being unknown

**Intimidation:** count on fear to keep you safe

---

## DESIGNING A SECURE SYSTEM

Basic steps:

- Slide 8**
- ① Specify security policy
  - ② List threats: how can security policy be violated
  - ③ Choose mechanisms to prevent successful attacks
  - ④ Verify (formally or informally) that mechanisms foil threats

And just in case...

- Implement auditing to detect security violations due to unanticipated threats

## WHY SECURITY IS HARD

### Weakest Link:

- Security of a system is only as strong as its weakest link
- Need to make sure all weak links are removed
- One bug is enough
- People are often the weakest link

### Complexity:

#### Slide 9

- Security involves many separate subsystems
- Complex to set up and use
- People won't use complex systems

### Pervasiveness:

- Application level
- Middleware level
- Network level
- OS level

### Distribution of Mechanisms:

#### Slide 10

- Trusted Computing Base (TCB): *those parts of the system that are able to compromise security*
- The smaller the TCB the better.
- May have to implement key services yourself
- Physically separate security services from other services

### Simplicity:

- Simplicity contributes to trust
- Very difficult to make a simple secure system

## COMMUNICATION SECURITY

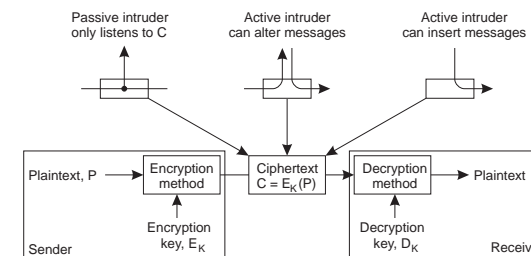
#### Slide 11

- Cryptography
- Signatures and Digests
- Protocols and Key Distribution
- Secure Communication

## CRYPTOGRAPHY

### The Basic Idea:

#### Slide 12



- Map **cleartext** (or plaintext)  $T$  to **ciphertext** (or *cryptogram*)  $C$
- Mapping is by a **well-known** function parameterised by a **key**  $K$
- $T$  infeasible to reconstruct from  $C$  without knowledge of key

Slide 13

More formally:

- $E(K_E, T) = E_{K_E}(T) = \{T\}_{K_E}$ : encryption of  $T$  with key  $K_E$
- $D(K_D, C) = D_{K_D}(C) = \{C\}_{K_D}$ : decryption of  $C$  with key  $K_D$
- $D(K_D, E(K_E, T)) = T$  for **cognate** (matching) keys  $K_D, K_E$

THE CAST

The Good Guys:

- Alice, Bob
- Want to communicate securely

Slide 14

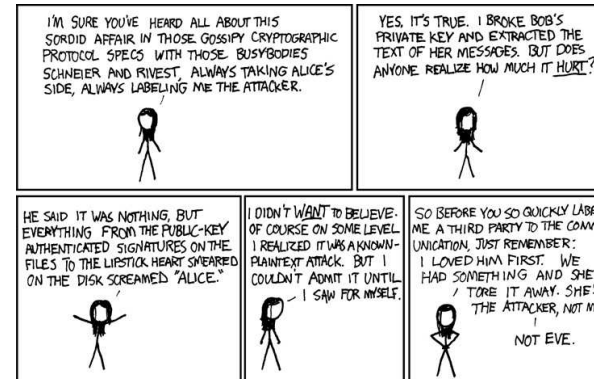
The Bad Guys:

- Eve
- The eavesdropper — tries to thwart Alice and Bob's plans

The Alice and Bob After Dinner Speech:

- google it more about Alice and Bob

Slide 15



Cryptographer:

- Uses cryptography to convert plaintext into ciphertext

Cryptanalyst:

Slide 16

- Uses cryptanalysis to attempt to turn ciphertext back into plaintext
- Cryptanalysis: the science of making encrypted data unencrypted

---

## ENCRYPTION

The essence of encryption functions:

Find a function  $E$  that is easy to compute, but for which it is hard to compute  $T$  from  $\{T\}_{K_E}$  without a matching decryption key  $K_D$  for  $K_E$ .

Slide 17

- "Hard to compute" means that it must take at least hundreds of years to reverse  $E$  without knowledge of  $K_D$  or to compute  $K_D$
- Such functions are known as **one-way functions**.

Cipher must be resilient to:

- Ciphertext only attacks
  - Known plaintext attacks
  - Chosen plaintext attacks
  - Brute-force attacks
- 

What properties should a good cipher possess?

Slide 18

- Confusion: every bit of key influences large number of ciphertext bits
  - Diffusion: every bit of plaintext influences large number of ciphertext bits
  - Fast to compute, ideally in hardware
  - Easy to use
  - Not critically depend on users selecting "good" keys
  - Have been heavily scrutinised by experts
  - Based on operations which are provably "hard" to invert
- 

In practice, keys are of finite length. Consequences?

Slide 19

- Finite key space  $\Rightarrow$  susceptible to exhaustive search
  - Longer keys  $\Rightarrow$  more time needed for brute-force attack
    - Time to guess a key is exponential in the number of bits of the key)
  - ✗ Longer keys also make  $E$  and  $D$  more expensive
  - Cipher must be secure against any systematic attack significantly faster than exhaustive search of key space
- 

## BASIC CIPHERS

Substitution Ciphers:

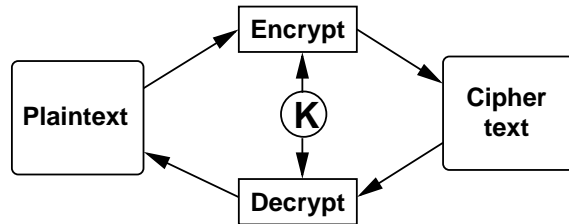
- Each plaintext character replaced by a ciphertext character
- Caesar cipher: shift alphabet  $x$  positions
  - Easy to break using statistical properties of language
- Book cipher: replace words by location of word in book
  - Knowledge of book is the key

Slide 20

One Time Pads:

- Random string XORed with plaintext
  - Information theoretically secure
  - Random string must:
    - Have no pattern or be predictable
    - Not be reused
    - Not be known by cryptanalyst
  - Key distribution problem
-

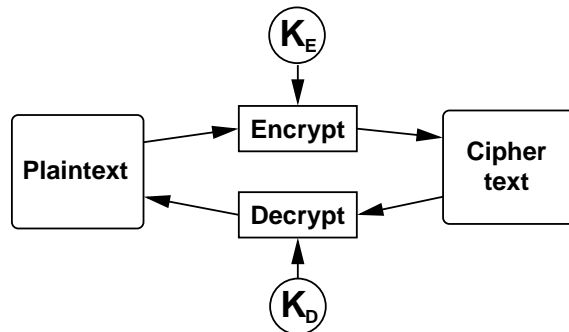
Symmetric ciphers:



Slide 21

- Secret key:  $K_E = K_D$
- ✓ fast ⇒ suited for large data volumes
- ✗ Secure channel is needed to establish the shared, secret key
- How many keys needed for  $N$  agents?
  - ➔ For any two agents, one key is needed
- Examples: Enigma, UNIX crypt, DES (data encryption standard), IDEA, AES

Asymmetric ciphers:



Slide 22

- Due to Diffie & Hellman & Merkle (1976)
- Instead of one secret key per pair of agents, one public/private key pair per agent
- $K_E \neq K_D$ ,  $K_D$  infeasible to compute from  $K_E$

- each agent can publish public key  $K_E =: K_P$ , keep private key  $K_D =: K_p$  secret
- Too slow to encrypt large volumes of data
- Examples: RSA and variants of Diffie & Hellman's original algorithm, such as ElGamal

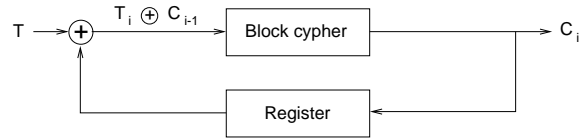
Slide 23

How they work:

- Trap-door functions: one-way functions with a secret exit
- Easy to compute in one direction, but infeasible to invert unless a secret (secret key) is known
- Key pair is usually derived from a common root (such as large prime numbers) such that it is infeasible to reconstruct the root from the public key

Slide 24

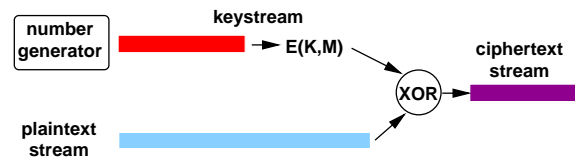
Block ciphers:



Slide 25

- Encrypt **fixed-size blocks** of data (e.g., 64 bits), one at a time
- Requires some padding in the last block (weakness?)
- Blocks of ciphertext are independent (weakness?)
  - Attacker may spot repeating patterns and infer relationship to plaintext

Stream cipher:

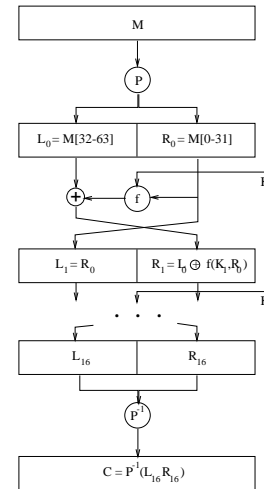


Slide 26

- Encode a given plaintext **bit by bit** (e.g., voice)
- Xor a **keystream** (sequence of 'random' bits) with the plaintext
- Security of ciphertext same as keystream
- Keystream: Output of a random number generator encoded with a block cipher algorithm
- How does the receiver reconstruct the plaintext?
  - Generate the same keystream and xor it with the ciphertext
  - requires starting value of RNG and the secret key
- Under which conditions can partial message loss be tolerated?

Note: This is not the same as a One Time Pad

DATA ENCRYPTION STANDARD (DES)



Slide 27

- Developed by IBM for US government
- Encode a 64-bit block  $M$  with a 56-bit key  $K$ 
  - 16 rounds
  - initial permutation  $P$
  - 48-bit key  $K_i$ :
    - obtained from  $K$  by rotating 1-2 bits each step
  - encryption step function  $f$ :
    - transpose and partially duplicate 32 bits of  $R$  yielding 48 bits
    - XOR with key
    - substitute/reduce number bits to 32 (S-box)

Properties of DES:

- $1.2 \times 10^9$  bits/s in hardware (Eberle & Thacker)
- kilobits/s in software
- Heavily used in banks, government institutions, etc.
- But vanilla DES no longer considered safe against brute-force attack:

Slide 28

- 06/1997 (DES-I), Verser: 12 weeks
- 02/1998 (DES-II-1), distributed.net: 39 days
- 07/1998 (DES-II-2), Electronic Frontier Foundation (EFF): 3 days
- 1999 (DES-III), distributed.net with EFF: 23h!
- "triple DES" uses  $112 = 2 \times 56$  bit key to encrypt-decrypt-encrypt with standard DES (very slow)
- effective key length shorter than 112 bits (Merkle & Hellman)

---

## OTHER SYMMETRIC CIPHERS

International Data Encryption Algorithm (IDEA):

- Uses 128-bit key to encrypt 64-bit blocks
- Approximately three times as fast as DES
- Same function for encryption and decryption (like DES)

**Slide 29** Advanced Encryption Standard (AES):

- Defined in 2001, to replace DES
  - The algorithm *Rijndael* (Daemen & Rijmen) was selected from a set of submissions and publicly reviewed
  - Variable block and key length; specification 128, 192, or 256 bit keys and 128, 192 or 256 bit blocks
  - Block and key length can be extended to multiples of 32 bit
- 

## RSA

Asymmetric (public key) cipher by Rivest, Shamir and Adelman Uses very large primes.

**Slide 30**

- ① Choose two primes  $P, Q > 10^{100}$
  - ②  $N = PQ; Z = (P - 1)(Q - 1)$
  - ③ Choose  $K_E$  relatively prime with  $Z$
  - ④ Determine  $K_D$  such that  $K_E K_D = 1 \pmod Z$
  - ⑤ Encrypt  $k$  bits,  $2^k < N$ :  $\{T\}_{K_E} = T^{K_E} \pmod N$
  - ⑥ Decrypt  $k$  bits:  $\{C\}_{K_D} = C^{K_D} \pmod N$
- 

Properties of RSA:

- Slow (kilobits/s even in hardware)
  - Easy to establish secure channel (distribute keys)
  - $\{\{T\}_{K_E}\}_{K_D} = \{\{T\}_{K_D}\}_{K_E} = T \forall T : 0 \leq T \leq N$ 
    - Can be used for digital signatures
  - Secure because factoring large numbers is computationally hard:
    - No proof of this
    - Recently key of  $\approx 500$  bits broken by brute force
    - Factoring shown to be polynomial on quantum computers
- 

**Slide 31**

---

## DIGITAL SIGNATURES & DIGESTS

Cryptographically ensure message integrity and authenticate originator.

How can we check whether a message has been altered?

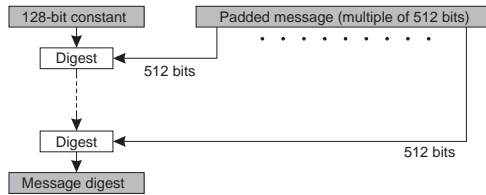
- **Secure digest** or **hash**
- Fixed-length value condensing information in the message
- Given message  $M$  and hash  $H(M)$ , it must be very hard to find  $M'$  with  $H(M) = H(M')$
- If hash  $H(M)$  is the same after transmission, message is unaltered with very high likelihood

**Slide 32**

Must be resilient against birthday attacks:

- Birthday paradox: chance that out of 23 people two have same birthday is 50%
  - **Birthday attack**: find two documents with matching hashes
-

Hash functions:



Slide 33

- Not unlike encryption functions, but not information preserving
- Most widely used algorithms: MD5 and SHA
- Rivest's MD5 algorithm: 128-bit digest; more efficient than SHA
- SHA is standardised, more secure (produces 160-bit digest)
- Any symmetric encryption algorithm could be used as hashing function with cipher block chaining, but
  - less efficient and
  - requires use of a key

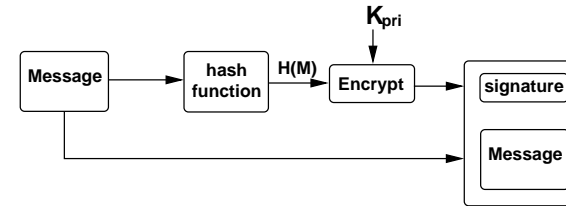
How can we make the digest tamper resistant?

Slide 34

- Transmitting the digest alongside the message doesn't prevent tampering
- Encrypt the digest using public key cryptography
- Sender encrypts the digest with private key

Digital Signature:

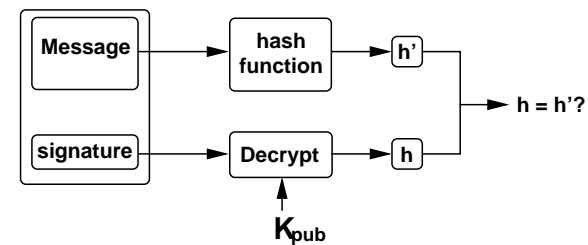
→ How to verify who sent the message



Slide 35

→ Given a message  $M$  and sender private key  $K_{pri}$ , signed message:

$$(M, \{H(M)\}_{K_{pri}})$$



Slide 36

- Recipient uses matching public key  $K_{pub}$  to recover digest
- Compare recovered digest to result of computing  $H(M)$
- If same, sent message must be unaltered and sender the owner of  $K_{pri}$

Slide 37

What guarantees are provided by a signed document in a sealed envelope?

- **Authentication**: receiver wants to be sure of the sender's identity
- **Confidentiality**: transmitted information is kept private
- **Integrity**: message could not have been altered
- **Non-repudiation**: sender cannot credibly deny having signed the message

## CRYPTOGRAPHIC PROTOCOLS

Build solutions to actual problems using:

- encryption
- signatures
- secure digest
- random number generators

Protocol mechanisms:

- Challenge-Response
  - nonce – used to uniquely relate two messages together
- Ticket – secured information to be passed to another party
- Session keys

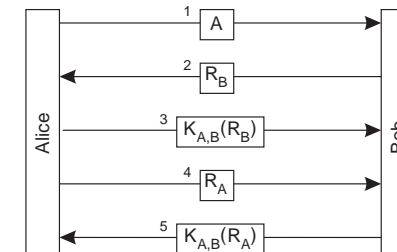
Principles:

- A message must contain all relevant information
- Don't allow parties to do things identically
- Don't give away valuable information to strangers

## A SIMPLE PROTOCOL

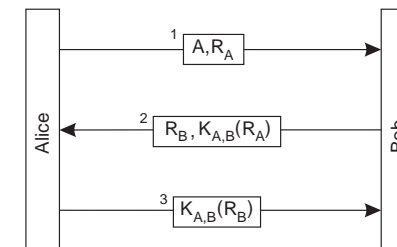
Authentication

Slide 39



## OPTIMISING THE PROTOCOL

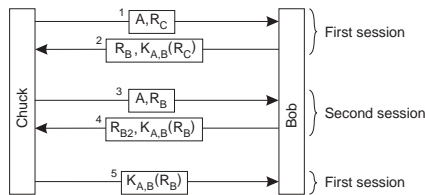
Slide 40



Oops!

- Vulnerable to reflection attack

Slide 41



### KEY DISTRIBUTION

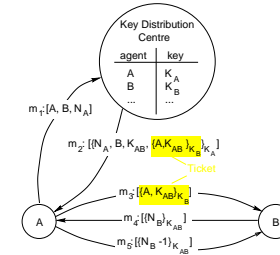
A set of keys provides a **secure channel** for communication.

How does the secure channel get established in the first place?

Slide 42

- Use separate channel to establish keys
- Use key distribution protocols
- Protocols vary depending on whether symmetric or asymmetric encryption is used
- Often symmetric keys are communicated over a channel using an asymmetric cipher

### DISTRIBUTION OF SYMMETRIC KEYS (NEEDHAM-SCHROEDER)



Slide 43

- Central **key distribution centre**  $D$
- Each agent  $A$  shares a (symmetric) key  $K_A$  with  $D$
- $A$  wants to communicate with  $B$ , asks  $D$  for **session key**  $K_{AB}$
- After key distribution protocol, both  $A$  and  $B$  know that they share a key provided by  $D$ .

Properties of the symmetric key distribution protocol:

Slide 44

- Ticket and challenge implicitly **authenticate**  $A$  and  $B$ .
- Nonce and challenge protect against replay attacks.
- $D$  is centralised resource (hierarchical scheme possible).
- Every agent must **trust**  $D$ .
- $D$  maintains highly sensitive information (secret keys), compromising  $D$  compromises all communication.
- Large number of keys required (one per pair of agents), manufactured by  $D$  on-the-fly.
- $D$  must take care to make key sequence non-predictable.

## DISTRIBUTION OF PUBLIC KEYS

Major weakness of Needham-Schroeder:

- Key distribution centre as a central authority
- Compromised keys can be used to decrypt past communication

Public Key Infrastructure (PKI):

- Public keys can be exposed without risk
- Distribution centre only establishes link between identities and public keys

Certificates and certification authorities:

- A **certificate** links an identity with a public key
- Distribution centres are called **certificate servers** or **certificate directories**

Slide 45

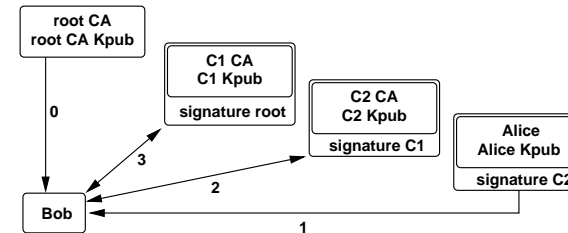
How to communicate certificates to clients?

- Secure channel between certificates server and client?
- Digital signatures establish the validity of certificates
- Formatted according to X509.1 standard or PGP format

Slide 46

Whose signature?

- **Certification authorities** sell certification as a service
- Alternatively, **web of trust** avoids any central authority



Slide 47

Checking of certificates is recursive:

- To establish trust in Alice's certificate signed by  $C_2$ , Bob may need to obtain  $C_2$ 's certificate
- Bob uses the public key of  $C_2$  to validate Alice's certificate
- $C_2$  is signed by  $C_1$
- This may lead to a **chain of certificates**
- Terminated by self-signed certificate of a **root certification authority** (who Bob trusts)

Are certificates valid forever?

- Certificates may have an expiry date to reduce risk of security breach
- After a certificate expires, a new one must be generated and signed
- Alternatively, certificates may be revoked
- Revocation is only effective if receiver regularly checks the certificate server

Slide 48

## SECURE COMMUNICATION

Properties of a Secure Channel:

Slide 49

- Authentication
- Message confidentiality
- Message integrity

## EXAMPLE: SSL (AND TLS)

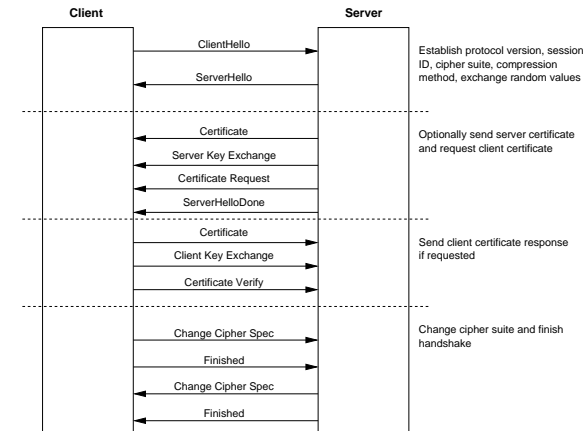
Secure Socket Layer:

Slide 50

- Application level protocol for secure channel
- Handshake protocol: establish and maintain session
- Record protocol: secure channel
- Flexible: can choose ciphers to use
- Most widely used to secure HTTP (https: URLs)
- TLS (Transport Layer security): IETF standard based on SSL 3.0
- TLS 1.0: RFC 2246

SSL Handshake Protocol:

Slide 51



## SECURE GROUP COMMUNICATION

Two types:

Confidential group communication:

- All group members share the same secret key  
Need to trust all members
- Separate keys for each pair  
Scalability problem
- Public key cryptography  
Everyone knows each others keys

Slide 52

Secure replicated servers:

- Collect responses from all servers and authenticate each  
Not transparent
- Secret sharing: All group members know part of a secret.  
Recipient combines answers from  $k$  members, decrypts with special decryption function  $D$ . If successful: these  $k$  members are honest. If not: try other combination of answers.

---

## PROTECTION

- Slide 53**
- Authentication
  - Authorisation
  - Access Control

---

## AUTHENTICATION

Verify the claimed identity of an entity (principal)

### Authentication Requires:

- Slide 54**
- Representation of identity
    - Unix user id, email address, student number, bank account
  - Some way to verify the identity
    - Password, reply to email, student card, PIN
  - Different levels of authentication

### Credentials:

- *Speaks for* a principal
- Example: certificate stating identity of a principal
- Combine credentials
- Role-based credentials

---

### Delegation:

- Slide 55**
- Entitles one principal to perform an action with the authority of another
  - Associated with right restriction
  - Delegation certificate, capabilities

---

### Approaches to Authentication:

- Slide 56**
- Shared secret key:** challenge and response encoded with shared secret key
  - Key distribution centre:** keys stored at KDC, never sent over network
  - Public key:** exchange session key encoded with public keys
  - Hybrid:** use public keys to set up a secure channel and then authenticate

## KERBEROS

- Commercial authentication system developed at MIT
- Based on Needham and Schroeder protocol
- Integrates symmetric key encryption, distribution and authentication into commercial computer systems.
- Assumptions:

Slide 57

- secure central server
- insecure network
  - never transmit cleartext passwords
- insecure workstations (shared between users)
  - hold user passwords on workstations for very short periods only
  - hold no system keys on workstations

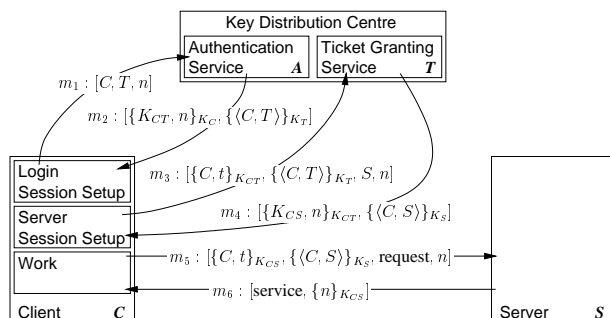
- Central KDC contains

- Authentication service  $A$ ,  
knows all user logins and their passwords (secret keys) as well as identity and key of  $T$ ;
- Ticket granting service  $T$ ,  
knows all servers and their secret keys

Slide 59

- Kerberos protocol has three phases:
  - ① login session setup (user authentication)
  - ② server session setup (establishing secure channel to server)
  - ③ client-server RPC
- Uses time-limited tickets

Kerberos Authentication:



Slide 58

## HYBRID: SSH1 AUTHENTICATION

Client	Server
Connect to server	
	SSH version
SSH Version	
	public host key: H public server key: S bulk data ciphers: 3DES, Blowfish Authentication methods: RSA, password
data cipher: 3DES session key: H(S(K))	
	encrypted with K: OK

Slide 60

Slide 61

Client:encrypted with K	Server:encrypted with K
login:alice	
	need authentication for alice
RSA authentication: public key P	
	<i>server checks authorized_keys</i> <i>not found</i> NO
password	
	<i>server checks password</i> <i>matches</i> SUCCESS

## AUTHORISATION AND ACCESS CONTROL

Determine what actions an authenticated entity is authorised to perform

Access Rights:

Slide 62 → The rights required to access (perform an operation on) a given resource

Two aspects:

**Access Control:** verify access rights

**Authorisation:** grant access rights

Ensuring that authorisation and access control are respected

Non-distributed Protection:

- Global mechanisms
- Global policies
- Examples:
  - Users
  - File permissions
  - Separate address spaces

Slide 63

Distributed Protection:

- Service specific
  - Web servers and .htaccess: authentication, access control
- Application specific

Design considerations in a protection system:

- Propagation of rights:
  - Can someone act as an agent's proxy?
- Restriction of rights:
  - Can an agent propagate a subset of their rights?
- Amplification of rights:
  - Can an unprivileged agent perform some privileged operations?
- Revocation of rights:
  - Can a right, once granted, be remove from an agent?
- Determination of **object accessibility**
  - Who has which rights on an object?
- Determination of **agent's protection domain**
  - What is the set of objects an agent can access?

Slide 64

## ACCESS CONTROL MATRIX

	Objects			
Subjects	$O_1$	$O_2$	$O_3$	$O_4$
$S_1$	terminate	wait, signal, send	read	
$S_2$	wait, signal, terminate			read, execute write, control
$S_3$		wait, signal, receive		
$S_4$	control		execute	write

Slide 65

- Access permissions of a given subject to a given object
- Specifies allowed operations

### Properties of the access matrix:

- Rows define subjects' **protection domains**
- Columns define objects' **accessibility**
- Dynamic data structure: frequently changes
  - permanent changes (e.g. `chmod`)
  - temporary changes (e.g. `setuid` flag)
- Matrix is very **sparse** with many repeated entries
  - ➔ usually not stored explicitly

Slide 66

### Access control lists (ACLs):

Object	Subjects			
	$S_1$	$S_2$	$S_3$	$S_4$
<code>/etc/passwd</code>	read	read, write	-	read

Slide 67

- Column-wise representation of the access matrix
- Each object associated with a list of (**subject, rights**) pairs
  - ➔ requires explicit authentication
- Usually supports concept of **group rights** (domain classes) (granted to each agent belonging to the group)
- Often simplified to a simple fixed-size list (e.g., UNIX *user-group-others* or VMS *system-owner-group-world*)
- Can have negative rights as well (e.g., to simplify exclusion from groups)

### Properties of ACLs:

- **Propagation**: (e.g., owner can `chmod`)
- **Restriction**: meta-right
- **Amplification**: (e.g., `setuid`)
- **Revocation**: remove from ACL
- **Object accessibility**: explicit in ACL
- **Protection domain**: hard (if not impossible)

Slide 68

Capabilities:

- An element of access matrix
- Capabilities list (C-list) associated with each subject, which defines a protection domain
- Each capability can confer a single or a set of rights
- Capabilities can confer negative rights
- Capabilities must be protected against forgery and theft
- Capability implies an **object name**:
  - **prima facie evidence** of access permission
  - independent of authentication
  - don't need to trust intermediary

Slide 69

Properties of capabilities:

- **Propagation**: copy capability (but need to be careful about confinement)
- **Restriction**: may be supported by **derived** capabilities
- **Amplification**: may have amplification capabilities
- **Revocation**: difficult, requires invalidation
- **Object accessibility**: hard (if not impossible)
- **Protection domain**: explicit in C-list

Slide 70

Three basic approaches to making caps tamper-proof:

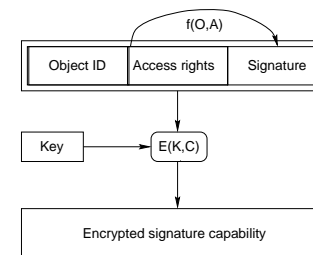
- **Tagged capabilities**:
  - protected by hardware (tag bit)
  - controlled by OS (only kernel can turn on tag bit)
  - used in most historical capability systems (Plessey 250, CAP, Hydra, System/38)
- **Partitioned (segregated) capabilities**:
  - protected by OS: Capabilities kept in kernel space
  - used in Mach, Grasshopper, EROS
- **Sparse capabilities**:
  - protected by sparseness (obscurity)
  - used in Monash Password Capability System, Amoeba, Mungi

Slide 71

Signature capabilities:

- "First Migration Scheme": Designed for distributing tagged capabilities

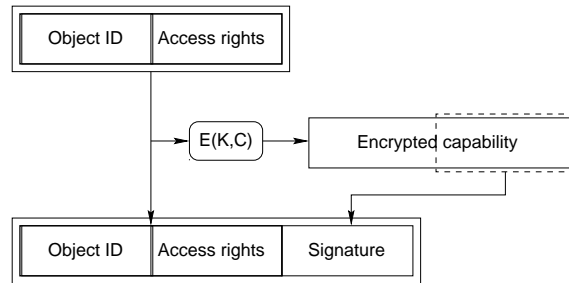
Slide 72



- ✓ tamper proof via encryption with **secret kernel key**
- ✓ can be freely passed around
- ✗ need to decrypt on each validation
- ✗ users do not know which object a capability refers to
- ✗ secret kernel key

Signature capabilities:

→ "Second Migration Scheme"



Slide 73

→ makes object ID visible, yet still tamper proof

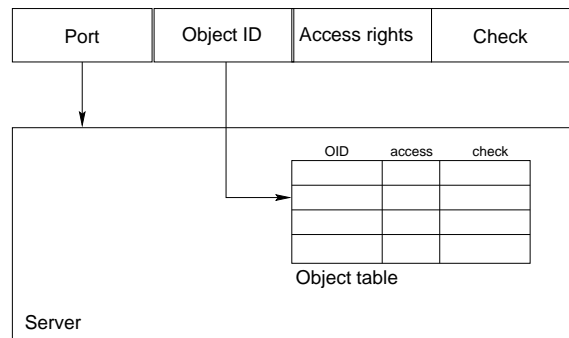
Properties of Amoeba's capabilities:

- Port identifies server (kernel caches server location)
- Port IDs are large (48-bit) sparse numbers (knowing port implies send rights)
- Server uses OID to look up rights, check fields to compare
- Original ("owner") capability has all rights.
  - **Restriction** by asking server to derive lesser capability
  - **Revocation** requires asking server to change check field ⇒ revokes access for *all* holders
  - **Amplification** by server

Slide 75

Amoeba's capabilities:

→ Amoeba is a server-based distributed OS using sparse capabilities

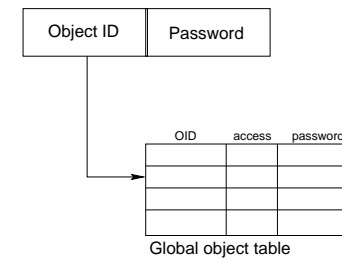


Slide 74

Password capabilities:

- Invented for Monash U's **Password Capability System**
- "Random" bitstring is password, not derived from other parts of capability.
- Validation requires checking against global **object table**.

Slide 76



---

## FIREWALLS

### Properties:

- When communicating with untrusted clients/servers `
- Disconnects part of system from outside world
- Incoming communication inspected and filtered

### Two types:

- Packet-filtering gateway
- Application-level gateway

### Three Myths of Firewalls:

- ① We've got the place surrounded
- ② Nobody here but us chickens
- ③ Sticks and Stones may break my bones, but words will never hurt me

Slide 77

---

## HOW TO BREAK SECURITY?

### Encryption:

- find weaknesses in algorithms
- find weaknesses in implementations
- attack underlying intractable problem
- brute force

### Protocols:

- try man-in-the-middle, reflection attacks
- find vulnerability in implementation

### Authentication:

- find keys or passwords
- social engineering

### Authorisation and Access Control:

- find and exploit bugs to escalate privileges

Slide 78