CS343 - Operating Systems

Module-8A

Protection Services by Operating Systems

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

http://www.iitg.ac.in/johnjose/

Overview

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Access Control
- Revocation of Access Rights
- Capability-Based Systems

Objectives

- Discuss the goals and principles of protection in a modern computer system
- Explain how protection domains combined with an access matrix are used to specify the resources a process may access
- Examine capability based protection systems

Goals of Protection

- Computer consists of a collection of objects, hardware or software
- Each object has a unique name and can be accessed through a well-defined set of operations
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so

Principles of Protection

- Guiding principle – **principle of least privilege**
  - Programs, users and systems should be given just enough **privileges** to perform their tasks
  - Can be static (during life of system, during life of process)
  - Or dynamic (changed by process as needed) – **domain switching**, **privilege escalation**

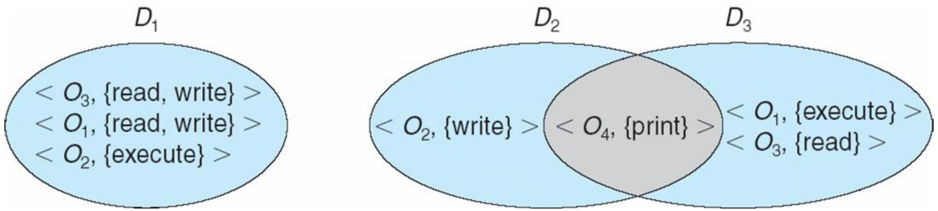
Principles of Protection

- Rough-grained privilege management easier, simpler, but least privilege now done in large chunks
  - For example, traditional Unix processes either have abilities of the associated user, or of root
- Fine-grained management more complex, more overhead, but more protective
  - File Access Control List (ACL), Roll Based Access Control (RBAC)
- Domain can be user, process, procedure

Domain Structure

- Access-right = <object-name, rights-set>
  where rights-set is a subset of all valid operations that can be performed on the object

- Domain = set of access-rights



Domain Implementation (UNIX)

- Domain (user-id) switch accomplished via file system
    - Each file has associated with it a domain bit (setuid bit)
    - When file is executed and setuid = on, then user-id is set to owner of the file being executed
    - When execution completes user-id is reset
- Domain switch accomplished via passwords
    - `su` command temporarily switches to another user’s domain when other domain’s password provided
- Domain switching via commands
    - `sudo` command prefix executes specified command in another domain (if original domain has privilege or password given)

Access Matrix

- View protection as a matrix (**access matrix**)
- Rows represent domains & columns represent objects
- **Access(i, j)** is the set of operations that a process executing in Domain$_i$ can invoke on Object$_j$
- If a process in Domain D_i tries to do ***op*** on object O_j, then ***op*** must be in the access matrix

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Use of Access Matrix

- User who creates object can define access column for that object
- Can be expanded to dynamic protection
  - Operations to add, delete access rights
  - Special access rights:
    - *owner of* O_i
    - *copy op from* O_i *to* O_j
    - *control* – D_i *can modify* D_j *access rights*
    - *transfer* – *switch from domain* D_i *to* D_j
  - *Copy* and *Owner* applicable to an object
  - *Control* applicable to domain object

Use of Access Matrix

- **Access matrix** design separates mechanism from policy
  - Mechanism
    - Operating system provides access-matrix + rules
    - It ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced
  - Policy
    - User dictates policy
    - Who can access what object and in what mode

Access Matrix of Figure A with Domains as Objects

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Access Matrix with Copy Rights

- A right is copied from access(i, j) to access(k, j); it is then removed from access(i, j). This action is a transfer of a right, rather than a copy.

- Propagation of the copy R* is copied from access(i,j) to access(k,j), only the right R (not R*) is created. A process executing in domain Dk cannot further copy the right R.

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Access Matrix With Owner Rights

- If access(i, j) includes the owner right, then a process executing in domain Di can add and remove any right in any entry in column j.

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Modified Access Matrix with Control Rights

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

- The control right is applicable only to domain objects.
- If access(i, j) includes the control right, then a process executing in domain Di can remove any access right from row j.

Implementation of Access Matrix

- Generally, access matrix is a sparse matrix
- **Option 1 – Global table**
  - Store ordered triples **<domain, object, rights-set>** in table
  - A requested operation M on object O_j within domain D_i →search table for $< D_i, O_j, R_k >$ with $M \in R_k$.
  - If this triple is found, the operation is allowed to continue; otherwise, an exception (or error) condition is raised
  - But table could be large → won't fit in main memory
  - Difficult to group objects (consider an object that all domains can read)

Implementation of Access Matrix

- **Option 2 – Access lists for objects**
  - Each column implemented as an access list for one object
  - Resulting per-object list consists of ordered pairs **<domain, rights-set>** defining all domains with non-empty set of access rights for the object
  - When an operation M on an object O_i is attempted in domain D, we search the access list for object O_i, looking for an entry $< D; R_k >$ with $M \in R_k$. If the entry is found, we allow the operation;
  - if it is not, we check the default set. If M is in the default set, we allow the access. Otherwise, access is denied

Implementation of Access Matrix

- Each column = Access-control list for one object
  Defines who can perform what operation
  - Domain 1 = Read, Write
  - Domain 2 = Read
  - Domain 3 = Read

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

- Each Row = Capability List (like a key)
  For each domain, what operations allowed on what objects
  - Object F1 – Read
  - Object F2 – Read, Write, Execute
  - Object F3 – Read, Write, Delete, Copy

Implementation of Access Matrix

- **Option 3 – Capability list for domains**
  - Instead of object-based, list is domain based
  - **Capability list** for domain is list of objects together with operations allows on them
  - Object represented by its name or address, called a **capability**
  - Execute operation M on object O_j, process requests operation and specifies capability as parameter
    - Possession of capability means access is allowed

Implementation of Access Matrix

- **Option 4 – Lock-key**
  - Compromise between access lists and capability lists
  - Each object has list of unique bit patterns, called **locks**
  - Each domain has list of unique bit patterns called **keys**
  - Process in a domain can only access object if domain has key that matches one of the locks

Comparison of Implementations

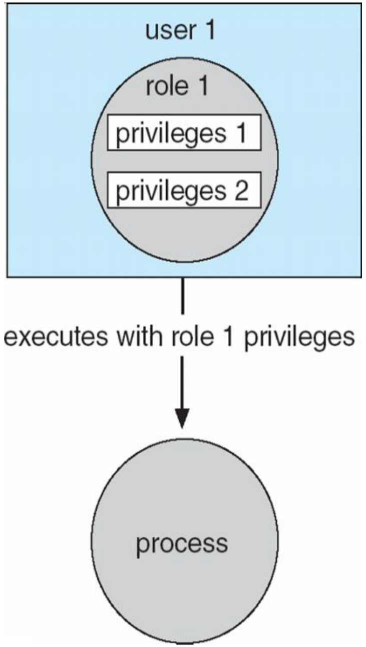
- Global table is simple, but can be large
- Access lists correspond to needs of users
  - Determining set of access rights for domain non-localized so difficult
  - Every access to an object must be checked
  - Many objects and access rights -> slow
- Capability lists useful for localizing information for a given process
  - But revocation capabilities can be inefficient
- Lock-key effective and flexible, keys can be passed freely from domain to domain, easy revocation

Comparison of Implementations

- Most systems use combination of access lists and capabilities
  - First access to an object → access list searched
    - If allowed, capability created and attached to process
    - Additional accesses need not be checked
    - After last access, capability destroyed
  - Consider file system with ACLs per file

Access Control

- Protection can be applied to non-file resources
- Oracle Solaris 10 provides **role-based access control** (**RBAC**) to implement least privilege
  - ***Privilege*** is right to execute system call or use an option within a system call
  - Can be assigned to processes
  - Users assigned ***roles*** granting access to privileges and programs
    - Enable role via password to gain its privileges
  - Similar to access matrix



Revocation of Access Rights

- Various options to remove the access right of a domain to an object
  - **Immediate vs. delayed**
  - **Selective vs. general**
  - **Partial vs. total**
  - **Temporary vs. permanent**
- **Access List** – Delete access rights from access list
  - **Simple** – search access list and remove entry
  - **Immediate**, **general** or **selective**, **total** or **partial**, **permanent** or **temporary**

Revocation of Access Rights

- **Capability List** – Scheme required to locate capability in the system before capability can be revoked
  - **Reacquisition** – periodic delete, with require and denial if revoked
  - **Back-pointers** – set of pointers from each object to all capabilities of that object
  - **Indirection** – capability points to global table entry which points to object – delete entry from global table, not selective (CAL)
  - **Keys** – unique bits associated with capability, generated when capability created
    - Master key associated with object, key matches master key for access
    - Revocation – create new master key

Capability-Based Systems

- **Hydra - A capability based protection system**
  - Fixed set of access rights known to and interpreted by the system
    - i.e. read, write, or execute each memory segment
    - User can declare other **auxiliary rights** and register those with protection system
    - Accessing process must hold capability and know name of operation
    - **Rights amplification** allowed by trustworthy procedures for a specific type
  - Includes library of prewritten security routines

Capability-Based Systems

- **Cambridge CAP System**
  - Simpler but powerful
  - **Data capability** - provides standard read, write, execute of individual storage segments associated with object – implemented in microcode
  - **Software capability** -interpretation left to the subsystem, through its protected procedures
    - Only has access to its own subsystem
    - Programmers must learn principles and techniques of protection

Thank you

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/

CS343 - Operating Systems

Module-8B

System Security and Threat Categories

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

http://www.iitg.ac.in/johnjose/

Objectives

- To discuss security threats and attacks
- To explain the fundamentals of encryption, authentication, and hashing
- To examine the uses of cryptography in computing
- To describe the various countermeasures to security attacks

Overview

- The Security Problem
- Program Threats
- System and Network Threats

The Security Problem

- Protection is strictly an internal problem →provide controlled access to programs and data stored in a computer
- A protection system is ineffective if user authentication is compromised or a program is run by an unauthorized user.
- System is **secure** if resources used and accessed as intended under all circumstances
- **Threat** is the potential for security violation
- **Attack** is attempt to break security
- **Intruders** (**crackers**) attempt to breach security
- Security violations can be accidental or malicious (intentional)
- Easier to protect against accidental than malicious misuse

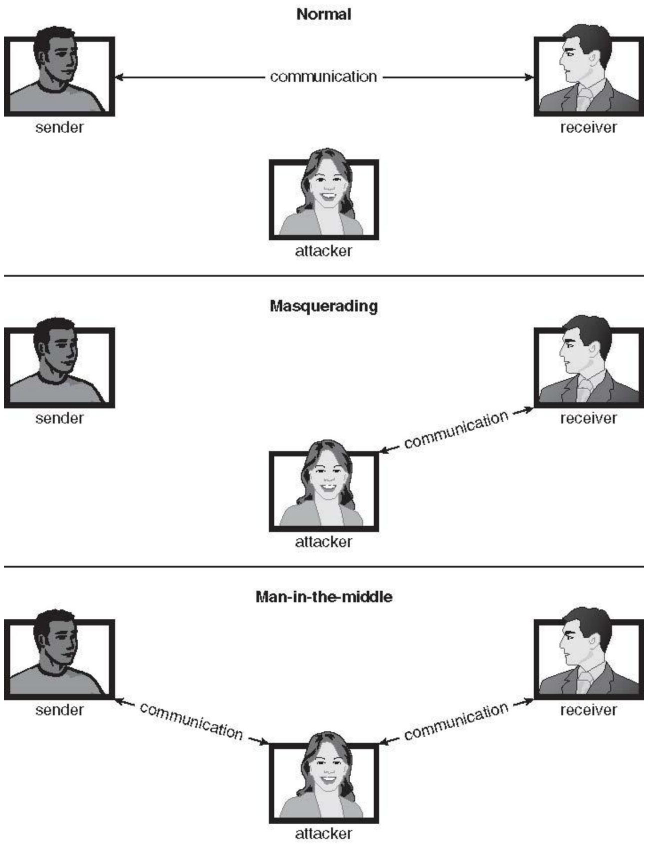
Security Violation Categories

- **Breach of confidentiality**
  - Unauthorized reading of data
- **Breach of integrity**
  - Unauthorized modification of data
- **Breach of availability**
  - Unauthorized destruction of data
- **Theft of service**
  - Unauthorized use of resources
- **Denial of service (DOS)**
  - Prevention of legitimate use

Security Violation Methods

- **Masquerading** (breach **authentication**)
  - Pretending to be an authorized user to escalate privileges
- **Replay attack**
  - Fraudulent repeat of a valid data transmission.
- **Man-in-the-middle attack**
  - Intruder sits in data flow, masquerading as sender to receiver and vice versa
- **Session hijacking**
  - Intercept an already-established session to bypass authentication

Standard Security Attacks



Security Measure Levels

- Security must occur at four levels to be effective:
  - **Physical :** Data centers, servers, connected terminals
  - **Human :** Avoid **social engineering**, **phishing**, **dumpster diving**
  - **Operating System :** Protection mechanisms, debugging
  - **Network :** Intercepted communications, interruption, DOS
- Security is as weak as the weakest link in the chain
- But can too much security be a problem?

Program Threats

- **Trojan Horse**
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
  - **Spyware**, **pop-up browser windows**, **covert channels**
  - Up to 80% of spam delivered by spyware-infected systems
- **Trap Door**
  - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler

Program Threats

- **Logic Bomb**
  - Program that initiates a security incident under certain circumstances
- **Stack** and **Buffer Overflow**
  - Exploits a bug in a program (overflow in stack or memory buffers)
  - Failure to check bounds on inputs, arguments
  - Write past arguments on the stack into the return address on stack
  - When routine returns from call, returns to hacked address
  - Pointed to code loaded onto stack that executes malicious code

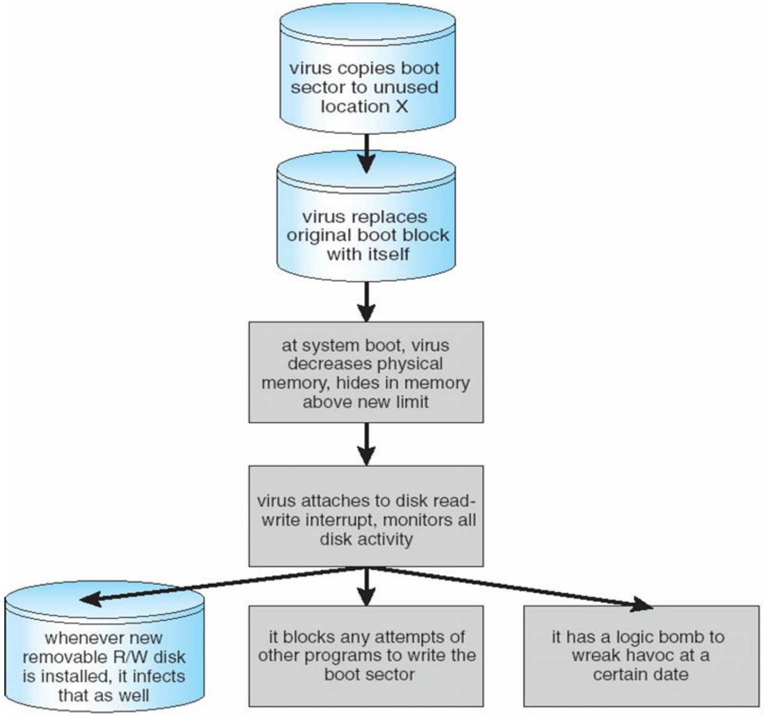
Program Threats

- **Viruses**
  - Code fragment embedded in legitimate program
  - Self-replicating, designed to infect other computers
  - Very specific to CPU architecture, operating system, applications
  - Usually borne via email or as a macro
  - **Virus dropper** inserts virus onto the system

Program Threats – Virus categories

- File / parasitic
- Boot / memory
- Macro
- Source code
- Polymorphic
- Encrypted
- Stealth
- Tunneling
- Multipartite
- Armored

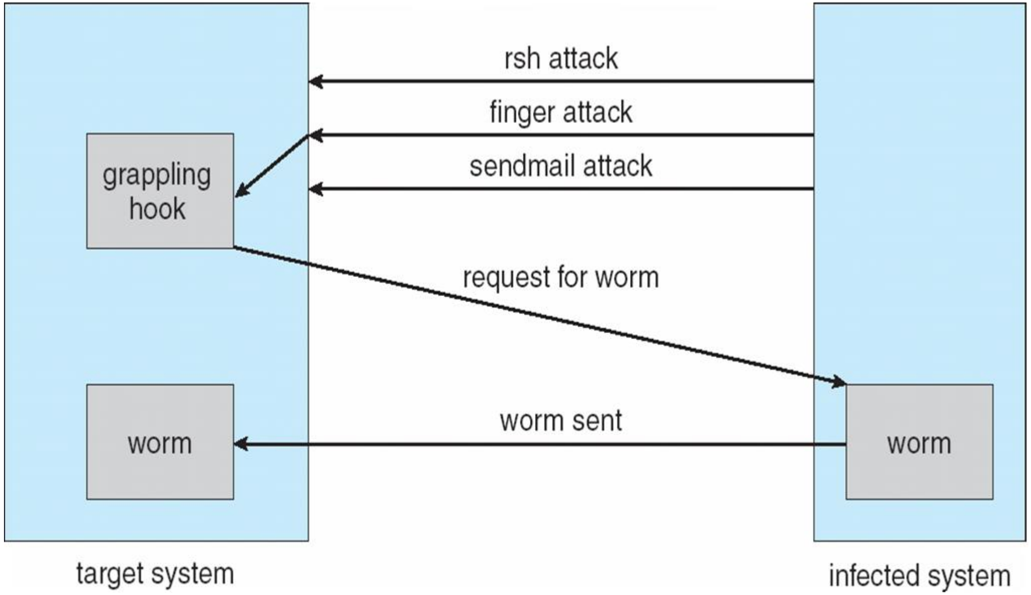
A Boot-sector Computer Virus



System and Network Threats

- **Worms** – use **spawn** mechanism; standalone program
- Internet worm (Morris worm)
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs
  - Exploited trust-relationship mechanism used by *rsh* to access friendly systems without use of password
  - **Grappling hook (bootstrap/ vector)** program uploaded main worm program – few lines of C code
  - Hooked system then uploaded main code, tried to attack connected systems
  - Also tried to break into other users accounts on local system via password guessing / rsh

The Morris Internet Worm



System and Network Threats

- **Port scanning**
  - Automated attempt to connect to a range of ports on one or a range of IP addresses
  - Detection of answering service protocol
  - Detection of OS and version running on system
  - Frequently launched from **zombie systems** to decrease trace-ability

System and Network Threats

- **Denial of Service**
  - Overload the targeted computer preventing it from doing any useful work
  - **Distributed denial-of-service** (**DDOS**) come from multiple sites at once
  - Consider the start of the IP-connection handshake (SYN)
    - How many started-connections can the OS handle?
  - Consider traffic to a web site - being a target and being really popular?

Thank you

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/

CS343 - Operating Systems

Module-8C

Security Mechanisms in Operating Systems

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

http://www.iitg.ac.in/johnjose/

Overview

- Cryptography
- User Authentication
- Implementing Security Defenses
- Firewalling to Protect Systems and Networks

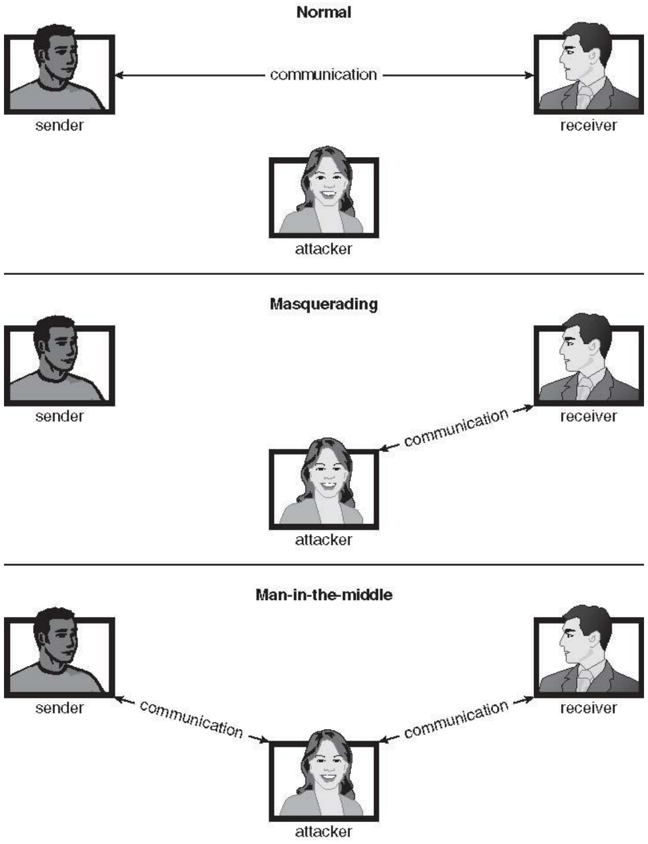
Objectives

- To explain the fundamentals of encryption, authentication, and hashing
- To examine the uses of cryptography in computing
- To describe the various countermeasures to security attacks

Security Violation Categories

- **Breach of confidentiality**
  - Unauthorized reading of data
- **Breach of integrity**
  - Unauthorized modification of data
- **Breach of availability**
  - Unauthorized destruction of data
- **Theft of service**
  - Unauthorized use of resources
- **Denial of service (DOS)**
  - Prevention of legitimate use

Standard Security Attacks



Cryptography as a Security Tool

- ❖ Broadest security tool - Art of secret writing
- ❖ Source and destination of messages can be known and protected
  - ❖ OS creates, manages, protects process IDs, communication ports
- ❖ Source and destination of messages on network cannot be trusted without cryptography
  - ❖ Local network – IP address?
    - ❖ Consider unauthorized host added
  - ❖ WAN / Internet – how to establish authenticity
    - ❖ Not via IP address

Cryptography

- Means to constrain potential senders (sources) and / or receivers (destinations) of messages
  - Based on secrets (**keys**)
  - Confirmation of source
  - Receipt only by certain destination
  - Trust relationship between sender and receiver
- Symmetric cryptography based on transformations
- Asymmetric cryptography based on mathematical functions
  - Asymmetric much more compute intensive
  - Typically not used for bulk data encryption

Encryption

- Constrains the set of possible receivers of a message
- **Encryption** algorithm consists of
  - Set K of keys
  - Set M of Messages
  - Set C of ciphertexts (encrypted messages)
  - A function $E : K \rightarrow (M \rightarrow C)$. That is, for each $k \in K$, E_k is a function for generating ciphertexts from messages
    - E_k for any k should be efficiently computable functions
  - A function $D : K \rightarrow (C \rightarrow M)$. That is, for each $k \in K$, D_k is a function for generating messages from ciphertexts
    - D_k for any k should be efficiently computable functions

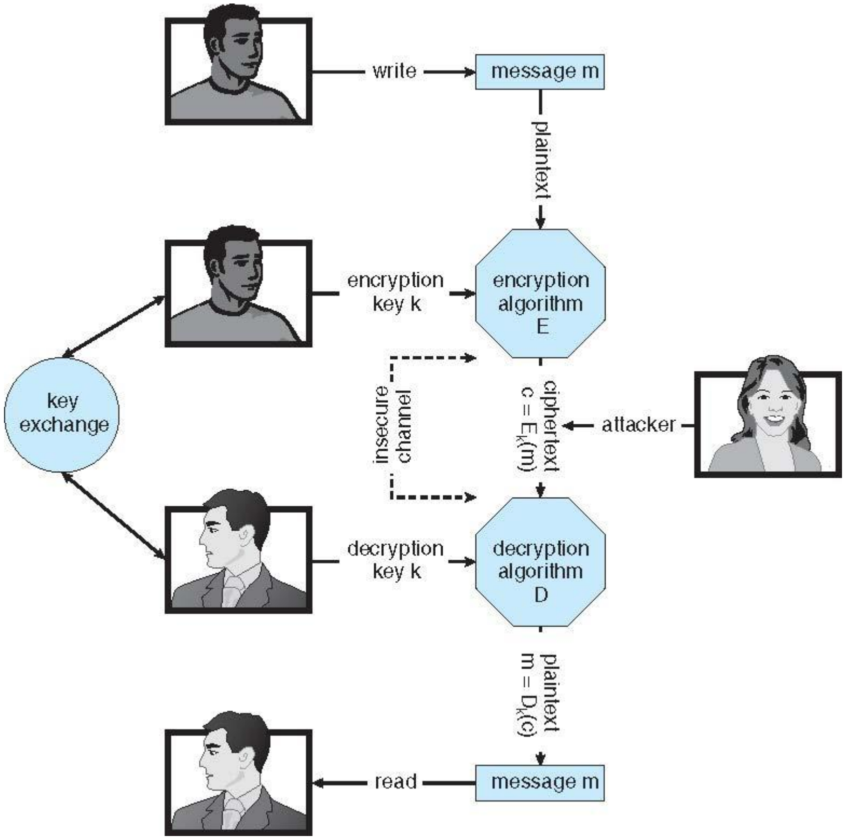
Encryption

- **Essential property of encryption algorithm**
- Given a ciphertext $c \in C$, a computer can compute m such that $E_k(m) = c$ only if it possesses k
  - Thus, a computer holding k can decrypt ciphertexts to the plaintexts used to produce them, but a computer not holding k cannot decrypt ciphertexts
  - Since ciphertexts are generally exposed (for example, sent on the network), it is important that it be infeasible to derive k from the ciphertexts

Symmetric Encryption

- Same key used to encrypt and decrypt
  - Therefore *k* must be kept secret
- DES was most commonly used symmetric block-encryption algorithm
- Triple-DES considered more secure $c = E_{k3}(D_{k2}(E_{k1}(m)))$
- Block cipher - Advanced Encryption Standard (**AES**)
  - Keys of 128, 192, or 256 bits, works on 128 bit blocks

Secure Communication over Insecure Medium



Asymmetric Encryption

- **Public-key encryption** based on each user having two keys:
  - **public key** – published key used to encrypt data
  - **private key** – key known only to individual user used to decrypt data
- Must be an encryption scheme that can be made public without making it easy to figure out the decryption scheme
  - Most common is **RSA** block cipher

Asymmetric Encryption

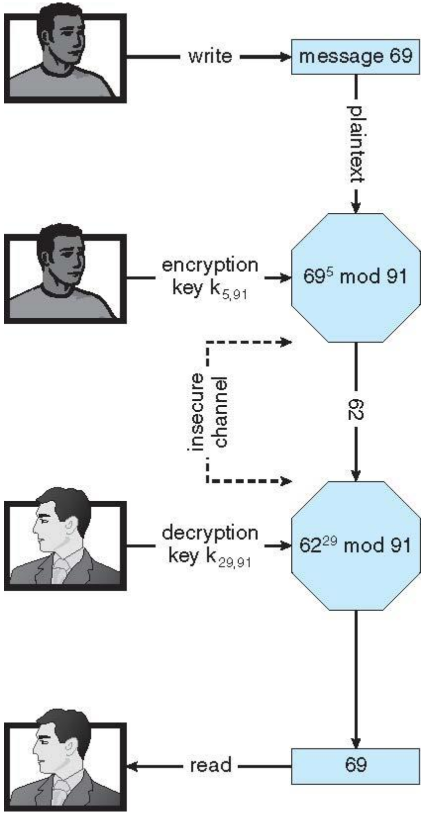
- Formally, it is computationally infeasible to derive $k_{d,N}$ from $k_{e,N}$, and so k_e need not be kept secret and can be widely disseminated
  - k_e is the **public key**
  - k_d is the **private key**
  - N is the product of two large, randomly chosen prime numbers p and q (for example, p and q are 512 bits each)
  - Encryption algorithm is $E_{ke,N}(m) = m^{k_e} \bmod N$, where k_e satisfies $k_e k_d \bmod (p-1)(q-1) = 1$
  - The decryption algorithm is then $D_{kd,N}(c) = c^{k_d} \bmod N$

Asymmetric Encryption Example

- Assume, $p = 7$ and $q = 13$
- We then calculate $N = 7*13 = 91$ and $(p-1)(q-1) = 72$
- We next select k_e relatively prime to 72 and< 72, yielding 5
- Finally, we calculate k_d such that $k_e k_d \bmod 72 = 1$, yielding 29
  - Public key, $k_{e,N} = 5, 91$
  - Private key, $k_{d,N} = 29, 91$
- Encrypting the message 69 with the public key results in the cyphertext 62
- Cyphertext can be decoded with the private key
  - Public key can be distributed in cleartext to anyone who wants to communicate with holder of public key

Encryption using RSA Asymmetric Cryptography

- ❖Public key, $k_{e,N}$ = 5, 91
- ❖Private key, $k_{d,N}$ = 29, 91
- ❖Encryption algorithm: $E_{ke,N}(m) = m^{k_e} \bmod N$.
- ❖Decryption algorithm: $D_{kd,N}(c) = c^{k_d} \bmod N$



Authentication

- Constraining set of potential senders of a message
  - Complementary to encryption
  - Also can prove message unmodified
- A set K of keys, set M of messages, A set A of authenticators
  - A function $S : K \to (M \to A)$
    - That is, for each $k \in K$, S_k is a function for generating authenticators from messages
    - Both S and S_k for any k should be efficiently computable functions
  - A function $V : K \to (M \times A \to \{true, false\})$. That is, for each $k \in K$, V_k is a function for verifying authenticators on messages
    - Both V and V_k for any k should be efficiently computable functions

Authentication

- For a message m, a computer can generate an authenticator $a \in A$ such that $V_k(m, a)$ = true only if it possesses k
- Thus, computer holding k can generate authenticators on messages so that any other computer possessing k can verify them
- Computer not holding k cannot generate authenticators on messages that can be verified using V_k
- Since authenticators are generally exposed (for example, they are sent on the network with the messages themselves), it must not be feasible to derive k from the authenticators
- Practically, if $V_k(m,a)$ = **true** then we know m has not been modified and that send of message has k
    - If we share k with only one entity, know where the message originated

Authentication – Hash Functions

- Basis of authentication
- Creates small, fixed-size block of data **message digest** (**hash value)** from m
- Hash Function H must be collision resistant on m
  - Must be infeasible to find an $m' \neq m$ such that $H(m) = H(m')$
- If $H(m) = H(m')$, then $m = m'$
  - The message has not been modified

Authentication – Hash Functions

- Common message-digest functions include **MD5**, which produces a 128-bit hash, and **SHA-1**, which outputs a 160-bit hash
- Not useful as authenticators
  - For example H(m) can be sent with a message
    - But if H is known someone could modify m to m’ and recompute H(m’) and modification not detected
    - So must authenticate H(m)

Authentication - MAC

- Symmetric encryption used in **message-authentication code** (**MAC**) authentication algorithm
- Cryptographic checksum generated from message using secret key
  - Can securely authenticate short values
- If used to authenticate H(m) for an H that is collision resistant, then obtain a way to securely authenticate long message by hashing them first
- Note that k is needed to compute both S_k and V_k, so anyone able to compute one can compute the other

Authentication – Digital Signature

- **Digital signatures -** based on asymmetric keys to verify authenticity of m.
- Similar to the RSA encryption algorithm, but the key use is reversed
- k_v is the public key and k_s is the private key
- Computationally infeasible to derive k_s from k_v
- RSA digital-signature algorithm
  - Digital signature of message $S_{ks}(m) = H(m)^{ks} \bmod N$
  - The key k_s again is a pair (d, N), where N is the product of two large, randomly chosen prime numbers p and q
  - Verification algorithm is $V_{kv}(m, a)$ $(a^{kv} \bmod N = H(m))$
    - Where k_v satisfies $k_v k_s \bmod (p - 1)(q - 1) = 1$

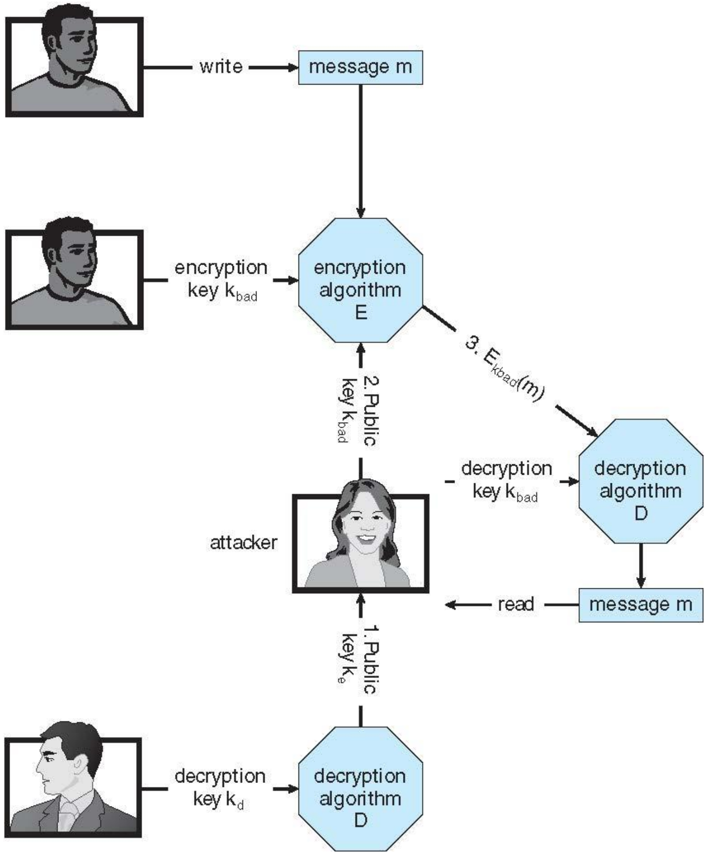
Key Distribution

- Delivery of symmetric key is huge challenge
  - Sometimes done **out-of-band**
- Asymmetric keys can proliferate – stored on **key ring**
  - Even asymmetric key distribution needs care – man-in-the-middle attack

Digital Certificates

- Proof of who or what owns a public key
- Public key digitally signed a trusted party
- Trusted party receives proof of identification from entity and certifies that public key belongs to entity
- **Certificate authority** are trusted party – their public keys included with web browser distributions
    - They vouch for other authorities via digitally signing their keys, and so on

Man-in-the-middle Attack - Asymmetric Cryptography



Implementation of Cryptography

- Can be done at various **layers** of ISO Reference Model
  - SSL at the Transport layer
  - Network layer is typically **IPSec**
    - **IKE** for key exchange
    - Basis of **Virtual Private Networks (VPNs)**

OSI model
7. Application Layer
NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · SMPP · SMTP · SNMP · Telnet · Netconf · (more)
6. Presentation Layer
MIME · XDR · TLS · SSL
5. Session Layer
Named Pipes · NetBIOS · SAP · L2TP · PPTP · SPDY
4. Transport Layer
TCP · UDP · SCTP · DCCP · SPX
3. Network Layer
IP (IPv4, IPv6) · ICMP · IPsec · IGMP · IPX · AppleTalk
2. Data Link Layer
ATM · SDLC · HDLC · ARP · CSLIP · SLIP · GFP · PLIP · IEEE 802.3 · Frame Relay · ITU-T G.hn DLL · PPP · X.25 · Network Switch · DHCP
1. Physical Layer
EIA/TIA-232 · EIA/TIA-449 · ITU-T V-Series · I.430 · I.431 · POTS · PDH · SONET/SDH · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · ITU-T G.hn PHY · USB · Bluetooth · Hubs
This box: **view** · talk · edit

OSI Model			
	Data unit	**Layer**	**Function**
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Encryption Example - SSL

- Insertion of cryptography at one layer of network model (transport layer)
- SSL – Secure Socket Layer (also called TLS)
- Cryptographic protocol that limits two computers to only exchange messages with each other
- Used between web servers and browsers for secure communication (credit card numbers)
- The server is verified with a **certificate** assuring client is talking to correct server
- Asymmetric cryptography used to establish a secure **session key** (symmetric encryption) for bulk of communication during session
- Communication between each computer then uses symmetric key cryptography

User Authentication

- Crucial to identify user correctly, as protection systems depend on user ID
- User identity most often established through **passwords**, can be considered a special case of either keys or capabilities
- Passwords must be kept secret
  - Frequent change of passwords
  - History to avoid repeats
  - Use of "non-guessable" passwords
  - Log all invalid access attempts (but not the passwords themselves)
  - Unauthorized transfer
- Passwords may also either be encrypted or allowed to be used only once

Passwords

- Encrypt to avoid having to keep secret
  - But keep secret anyway
  - Use algorithm easy to compute but difficult to invert
  - Only encrypted password stored, never decrypted
- One-time passwords
  - Use a function based on a seed to compute a password, both user and computer
- Biometrics
  - Some physical attribute (fingerprint, hand scan)
- Multi-factor authentication

Implementing Security Defenses

- **Defense in depth** – multiple layers of security
- **Security policy** describes what is being secured
- Vulnerability assessment compares real state of system / network compared to security policy
- Intrusion detection endeavors to detect attempted or successful intrusions
  - **Signature-based** detection spots known bad patterns
  - **Anomaly detection** spots differences from normal behavior
  - **False-positives** and **false-negatives** a problem

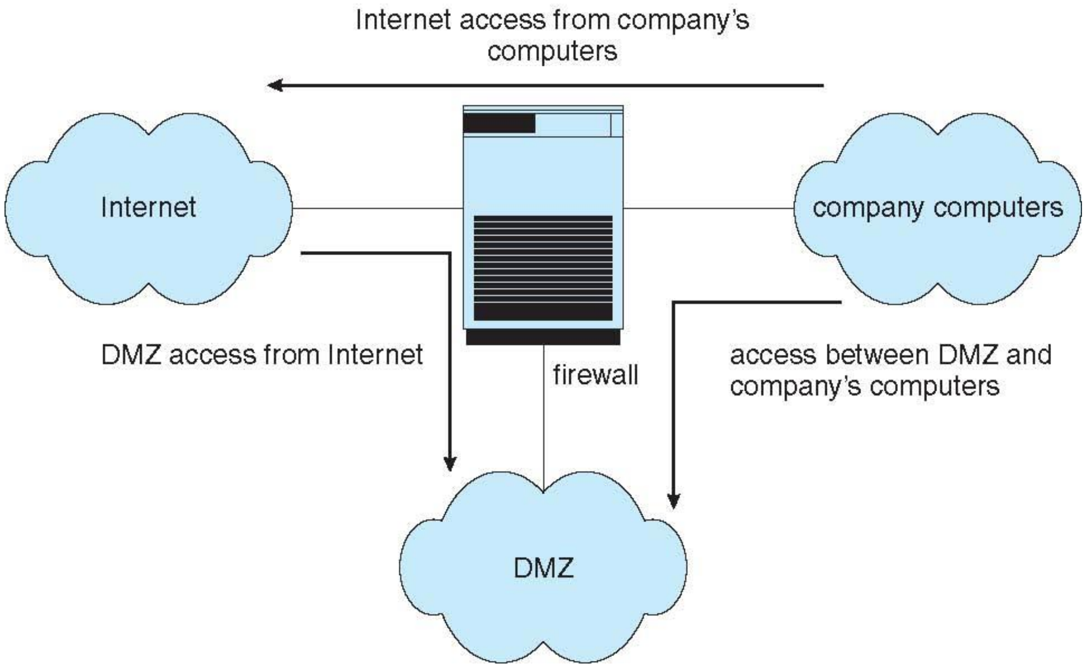
Implementing Security Defenses

- Virus protection
  - Searching all programs or programs at execution for known virus patterns
- Auditing, accounting, and logging of all or specific system or network activities
- Practice **safe computing** – avoid sources of infection, download from only good sites, etc

Firewalling to Protect Systems and Networks

- A network **firewall** is placed between trusted and untrusted hosts
- The firewall limits network access between these two **security domains**
- Firewall rules typically based on host name or IP address which can be spoofed
- **Personal firewall** is software layer on given host
  - Can monitor / limit traffic to and from the host
- **Application proxy firewall** understands application protocol and can control them (i.e., SMTP)
- **System-call firewall** monitors all important system calls and apply rules to them (i.e., this program can execute that system call)

Network Security Through Firewall



Thank you

johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/