Lecture 14
Security and Protection

Prof. Yinqian Zhang

Fall 2024

Goals of Protection

- Computer consists of a collection of objects (hardware objects or software objects)

- 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 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
  - Limits damage if entity has a bug, gets abused
  - Can be static (during life of system, during life of process)
  - Or dynamic (changed by process as needed) – **privilege escalation**

Access Matrix

- View protection as a matrix (**access matrix**)

- Rows represent domains (e.g. users or processes)

- Columns represent objects

- `Access(i, j)` is the set of operations that a process executing in Domain$_i$ can invoke on Object$_j$

object / domain	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

- If a process in Domain D_i tries to do "op" on object O_j, then "op" must be in the 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

- **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

Implementation of Access Matrix

- Generally, 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 $M \in R_k$
  - 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 (Cont.)

- Option 2 – Access-control lists for objects
  - Each column implemented as an access-control 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

Implementation of Access Matrix (Cont.)

- 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 (Cont.)

- Each column = Access-control list for one object
Defines who can perform what operation

  Domain 1 = Read, Write
  Domain 2 = Read
  Domain 3 = Read

- Each Row = Capability List (like a key)
For each domain, what operations allowed on what objects

  Object F1 – Read

  Object F4 – Read, Write, Execute

  Object F5 – Read, Write, Delete, Copy

Implementation of Access Matrix (Cont.)

- Most systems use combination of access-control lists and capabilities
  - First access to an object -> access-control list searched
    - If allowed, capability created and attached to process
      - Additional accesses need not be checked
    - After last access, capability destroyed

The Security Problem

- System **secure** if resources used and accessed as intended under all circumstances

- **Intruders** (**crackers or hackers**) attempt to breach security

- **Threat** is potential security violation

- **Attack** is attempt to breach security

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 Measure Levels

■ Impossible to have absolute security, but make cost to perpetrator sufficiently high to deter most intruders

■ 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

Malware

- ## Trojan Horse
  - Code segment that misuses its environment

- ## Trap Door
  - Specific user identifier or password that circumvents normal security procedures

- ## Virus
  - Code fragment embedded in legitimate program
  - Self-replicating, designed to infect other computers

- ## Logic Bomb
  - Program that initiates a security incident under certain circumstances

Program Security

- ## Stack and Buffer Overflow
  - Exploits a bug in a program (overflow either the 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
  - Unauthorized user or privilege escalation

System and Network Threats (Cont.)

- **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
  - `nmap` scans all ports in a given IP range for a response
  - `nessus` has a database of protocols and bugs (and exploits) to apply against a system

System and Network Threats (Cont.)

- **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 traffic to a web site

    - How can you tell the difference between being a target and being really popular?

  - Accidental – CS students writing bad `fork()` code

  - Purposeful – extortion, punishment

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

Implementing Security Defenses

- Intrusion detection endeavors to detect attempted or successful intrusions
  - **Signature-based** detection spots known bad patterns
  - **Anomaly detection** spots differences from normal behavior
    - Can detect **zero-day** attacks
  - **False-positives** and **false-negatives** a problem
- Virus protection
  - Searching all programs or programs at execution for known virus patterns
  - Or run in **sandbox** so can't damage system

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**
- Can be tunneled or spoofed
  - Tunneling allows disallowed protocol to travel within allowed protocol (i.e., telnet inside of HTTP)
  - 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)

Attacks against Network Communication

- **Eavesdropping**
  - Stealing the content of network communication

- **Replay attack**
  - Resend a previously intercepted message

- **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

Cryptography

- Encryption

- Authentication

- Key distribution

- These topics are covered in "Network security"

Encryption

- Protect **confidentiality** 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
    - Both E and 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
    - Both D and D_k for any k should be efficiently computable functions

Encryption (Cont.)

- An encryption algorithm must provide this essential property: 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
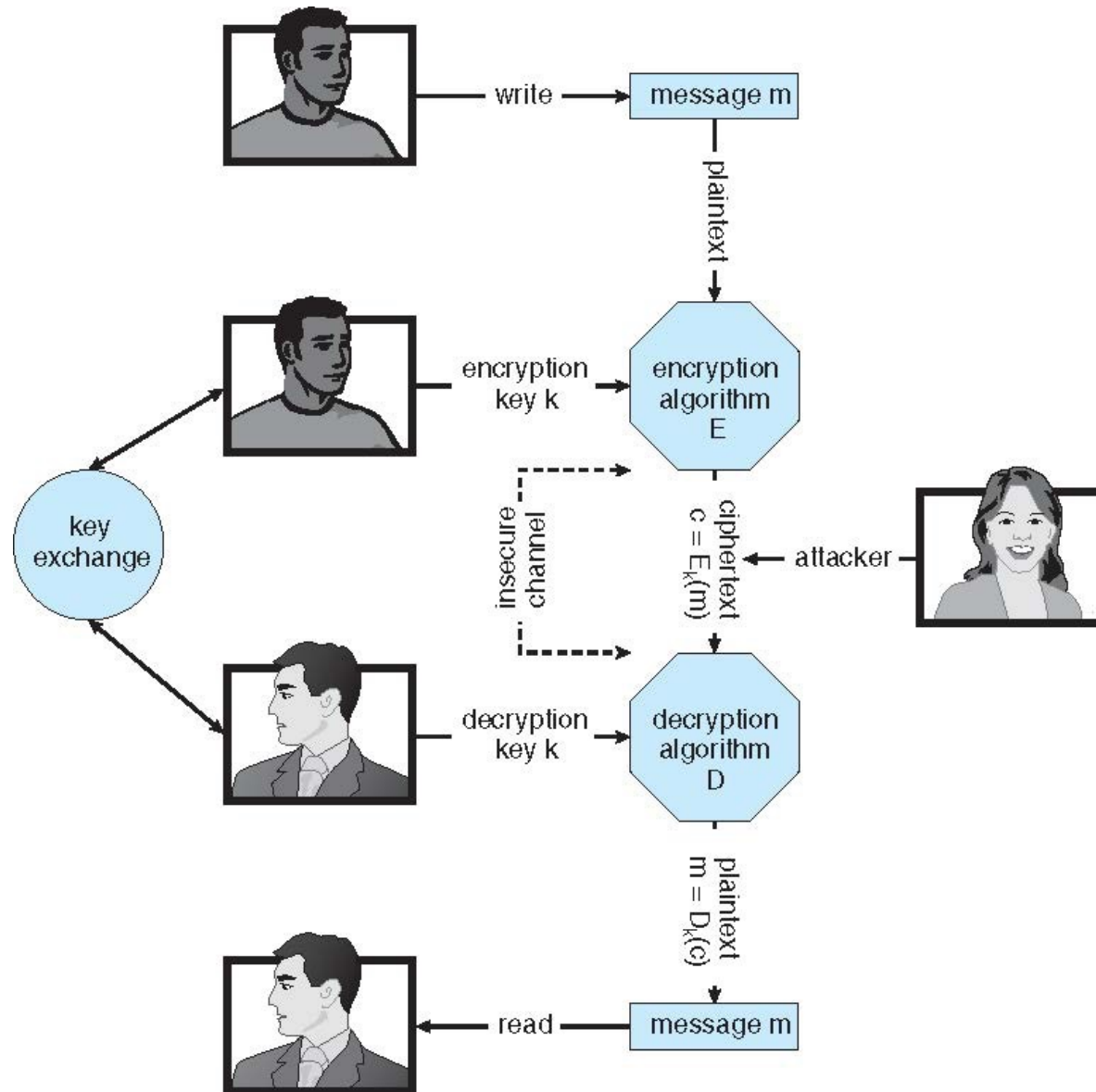
Symmetric Encryption

- Same key used to encrypt and decrypt
  - Therefore *k* must be kept secret

- Block cipher (messages encrypted block-by-block)
  - DES was most commonly used symmetric block-encryption algorithm
    - Keys too short so now considered insecure
  - 2001 NIST adopted new block cipher - Advanced Encryption Standard (**AES**)
    - Keys of 128, 192, or 256 bits, works on 128 bit blocks

- Stream cipher (message encrypted bit-by-bit or byte-by-byte)
  - RC4 is most common symmetric stream cipher, but known to have vulnerabilities

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** cipher

  - Efficient algorithm for testing whether or not a number is prime

  - No efficient algorithm is know for finding the prime factors of a number

Encryption using Asymmetric Cryptography
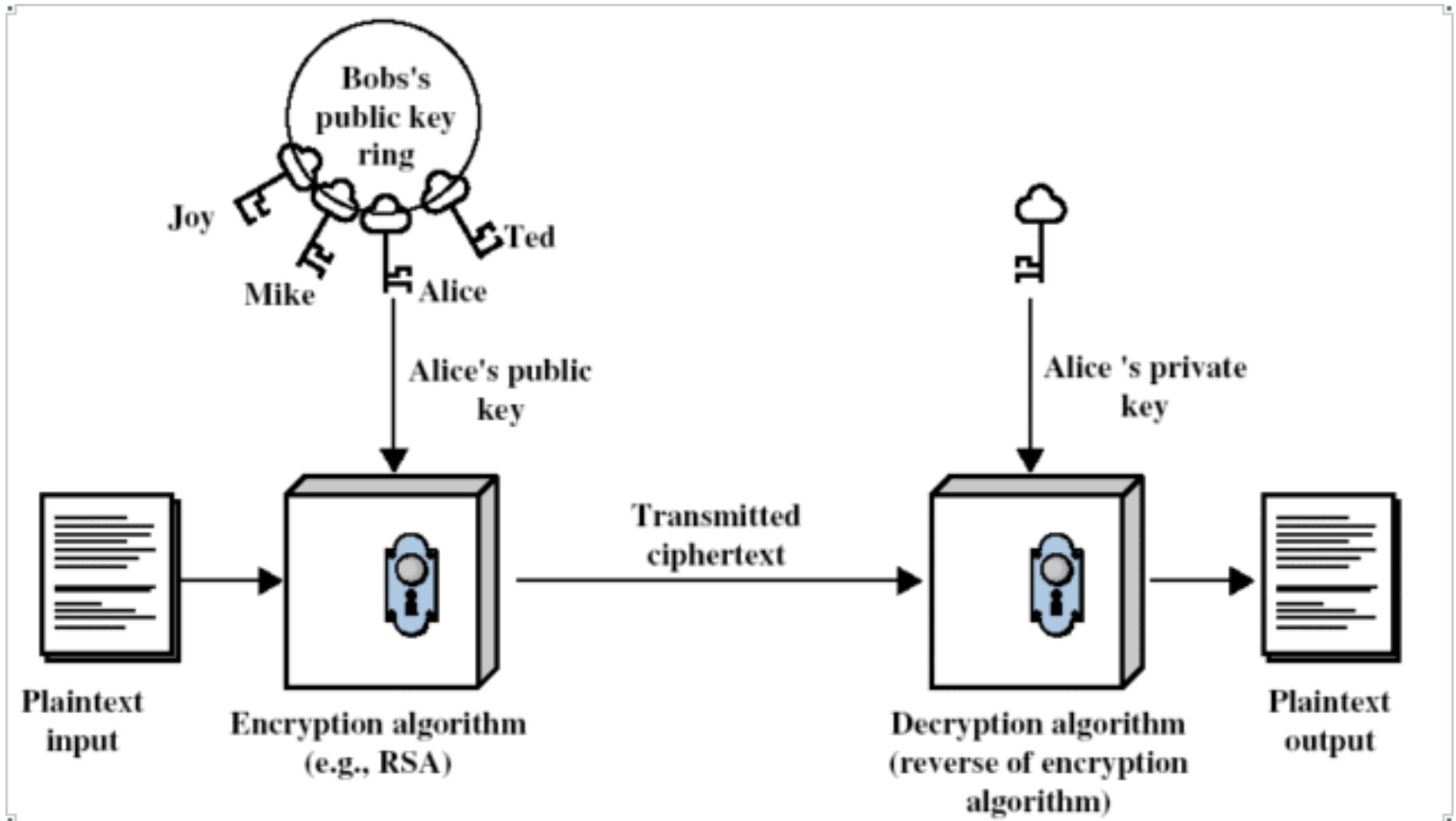
Authentication

- Protect integrity of a message
- Algorithm components
  - A set K of keys
  - A set M of messages
  - A set A of authenticators
  - A function $S : K \rightarrow (M \rightarrow A)$
    - 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 $V : K \rightarrow (M \times A \rightarrow \{true, false\})$. 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 (Cont.)

- 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

- **Message-authentication code** (**MAC**) authentication algorithm
  - Based on symmetric encryption
  - Both parties share secret keys

- **Digital signatures** authentication algorithm
  - Based on asymmetric encryption
  - *anyone* can verify authenticity of a message using the public key

Key Distribution

- Delivery of symmetric key is huge challenge
  - Sometimes done **out-of-band**

- Asymmetric keys distribution – public key
  - 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

Encryption Example – SSL/TLS

- 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

Thank you!