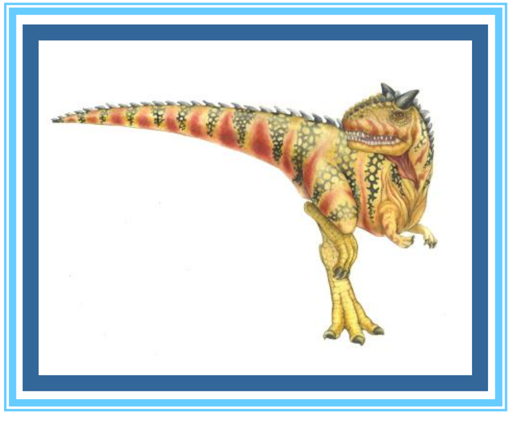
# Chapter 16: Security

Section 16.1-16.3, 16.4.1



# The Security Problem

- Goal of security is to protect
  - the integrity of the information stored in the system (both data and code)
  - and the physical resources of the computer system
- System is **secure** if resources are used and accessed as intended under all circumstances
  - Unachievable
- The security system prevents unauthorized access, malicious destruction or alteration of data, and accidental introduction of inconsistency
- **Intruders** are those who attempt to breach security
- A **threat** is anything that leads to loss or corruption of data or physical damage to the hardware and/or infrastructure
  - Theft, fire, virus, spyware
- An **attack** is an attempt to breach security
  - Attack can be accidental or malicious

# Requirements of Security Mechanisms

- **Confidentiality**: information maintained by a computer system is accessible only by authorized parties (users and the processes that run as/represent those users).

- **Integrity:** a computer system’s resources can be modified only by authorized parties.

- **Availability**: a computer system be accessible at required times by authorized parties.

- **Authenticity**: a computer system can verify the identity of a user

# 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

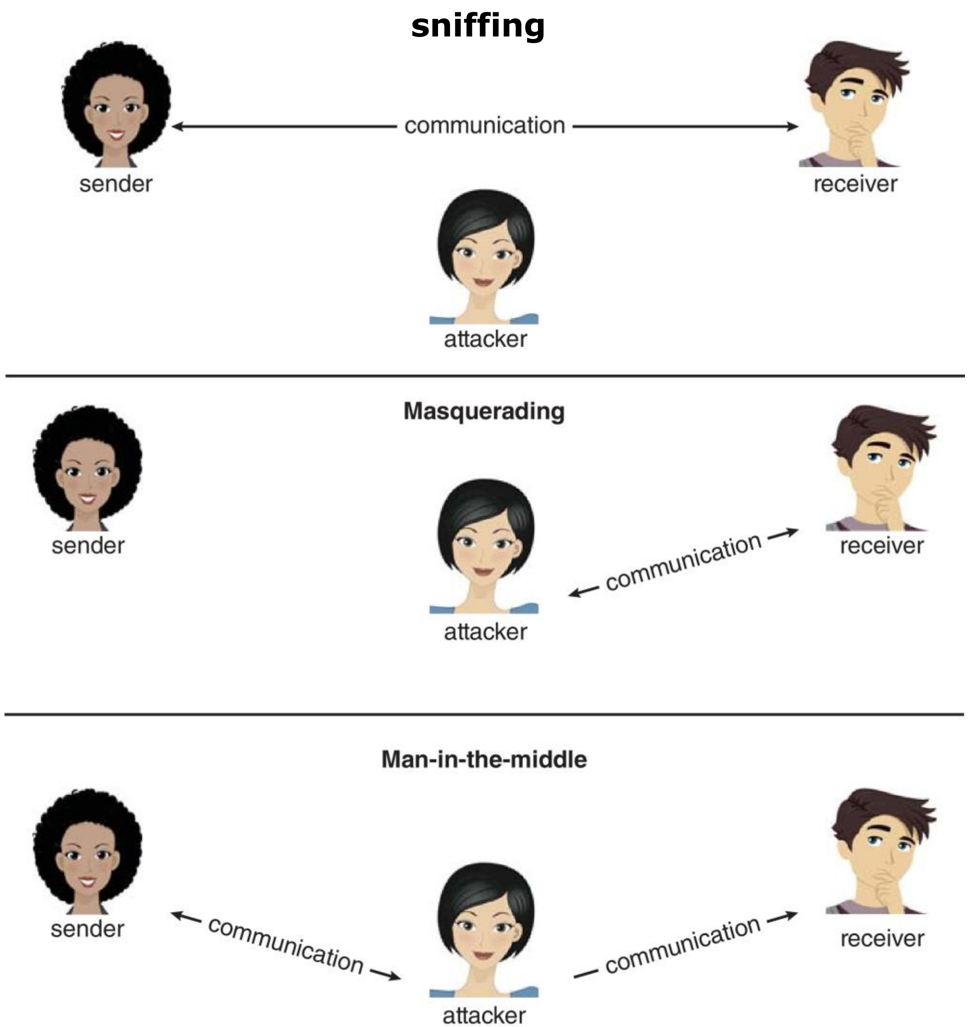
# Program Threats

- **Malware -** Software designed to exploit, disable, or damage computer systems
- **Trojan Horse** – Program that looks legitimate but can take control of your computer.
  - **Spyware** – Program frequently installed with legitimate software to display ads, capture user data (Up to 90% of spam delivered by spyware-infected systems)
- **Ransomware** – Locks up data via encryption, demanding payment to unlock it
- Malware thrive when there is a violation of the Principle of Least Privilege

***THE PRINCIPLE OF LEAST PRIVILEGE***

"The principle of least privilege. Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job. The purpose of this principle is to reduce the number of potential interactions among privileged programs to the minimum necessary to operate correctly, so that one may develop confidence that unintentional, unwanted, or improper uses of privilege do not occur."—Jerome H. Saltzer, describing a design principle of the Multics operating system in 1974: https://pdfs.semanticscholar.org/1c8d/06510ad449ad24fbdd164f8008cc730cab47.pdf.

# System and Network Threats



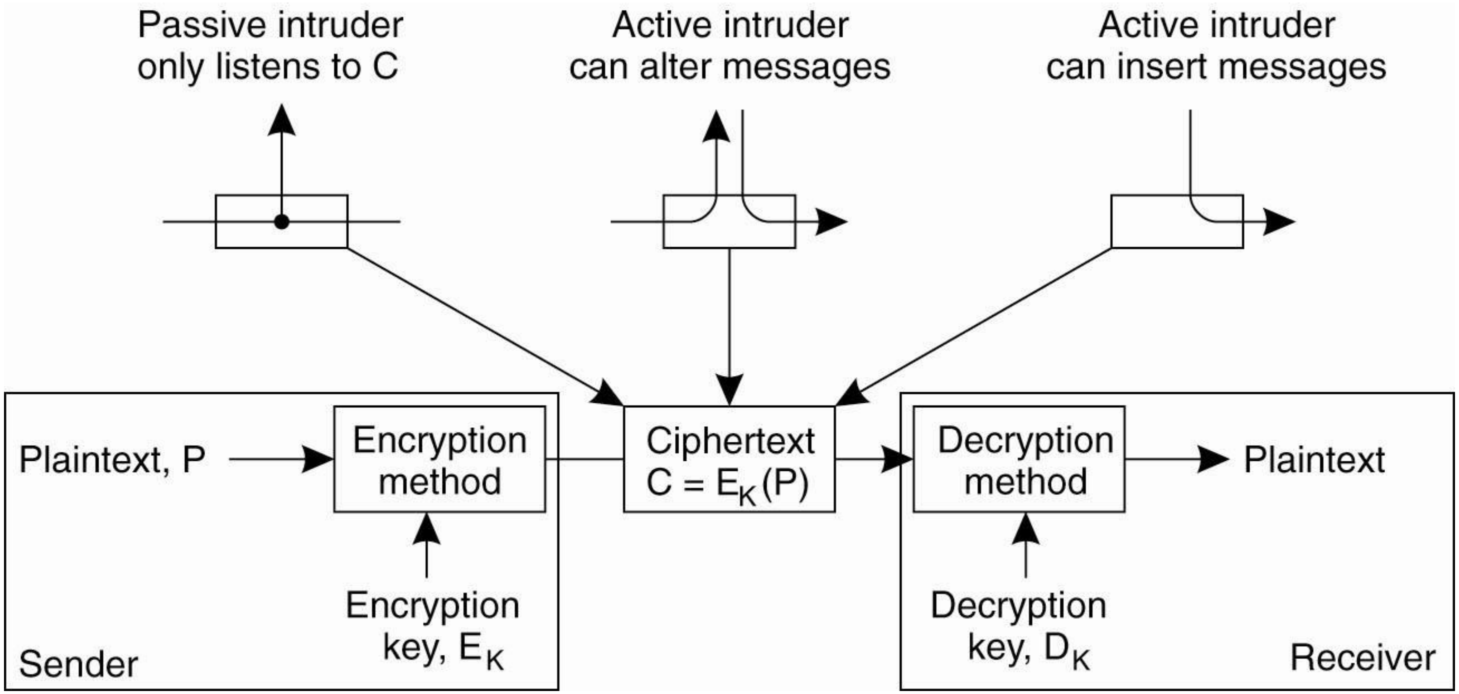
# System and Network Threats (Cont.)

- **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 the TCP-connection handshake
    - How many connections can the OS handle?
  - Consider traffic to a web site
    - How can you tell the difference between being a target and being really popular?
- **Port scanning**
  - Automated attempt to connect to a range of ports on one or a range of IP addresses
  - Detection of running services in order to identify vulnerabilities
  - Detection of OS and version running on system

# Cryptography

- Goal: keep information from those who aren't supposed to see it
  - Do this by encrypting the data
  - Encryption constrains the set of possible receivers of a message
- Use a well-known algorithm to encrypt data
  - Algorithm has two inputs: data and key
  - Key is known to authorized users

# Basics of Cryptography




- plaintext: unencrypted message
- ciphertext: encrypted form of message

# Cryptosystems

- Cryptosystems are either symmetric or asymmetric
- **Symmetric system**: $E_k = D_k$, so the key must be kept secret
- **Asymmetric system** (aka **public-key system**): $E_k \neq D_k$ , $E_k$ can be made public; $D_k$ is secret and can't easily be derived from $E_k$

# Symmetric Encryption Algorithms

- Same key is used to encrypt and decrypt
  - Therefore key *k* must be kept secret
- Data Encryption Standard (DES) was most commonly used symmetric block-encryption algorithm (created by US Government)
  - 56-bit keys
  - Encrypts a block of data at a time, 64-bit block size
  - Keys too short so now considered insecure
- Triple-DES considered more secure
  - Algorithm used 3 times using 3 keys

$$c = E_{k3}(D_{k2}(E_{k1}(m)))$$

- In 2001 NIST adopted a new block cipher - Advanced Encryption Standard (**AES**)
  - Keys of 128, 192, or 256 bits, works on 128-bit blocks
  - A machine that could crack 56-bit DES in one second would take 149 trillion years to crack a 128-bit AES key!

# 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
- Most common is **RSA** block cipher
  - No efficient algorithm is know for finding the prime factors of a number

# RSA

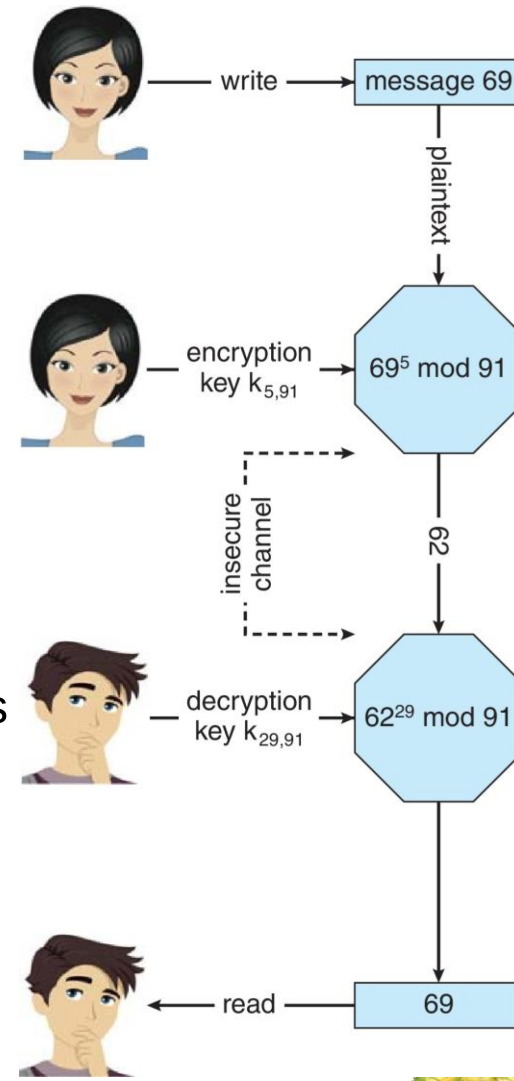
- $k_e$ is the **public key**
- $k_d$ is the **private key**
- $N$ is the product of two large, randomly chosen prime numbers $p$ and $q$ (for example, $p$ and $q$ are 512 bits each)
- Choose $k_e$ which has no common factors with $z=(p-1)(q-1)$
- Compute $k_d$ such that $k_e k_d = 1 \bmod z$
- Encryption algorithm is $E_{ke,N}(m) = m^{k_e} \bmod N$
- Decryption algorithm is $D_{kd,N}(c) = c^{k_d} \bmod N$
- *$K_e$ and N made public*
- *$K_d$ kept secret*

**To break RSA:**

- Need to know $p$, $q$, given $n=pq$ ($n$ known)
- Factoring 200 digit $n$ into primes takes 4 billion years using known methods

# RSA Example

- Make $p = 7$, $q = 13$
- 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
- We how have our keys
  - 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 ciphertext 62
- Decrypting the ciphertext 62 with the private key results in the message 69



# Why Does RSA Work?

- Number theory result:

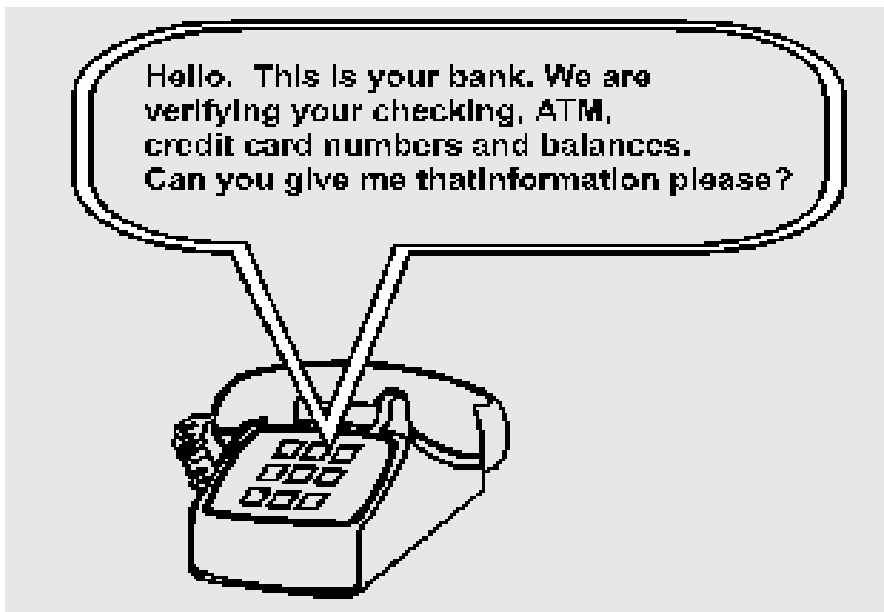
  if $p$, $q$ prime, then $b^{(p-1)(q-1)} \bmod pq = 1$

- Using mod $pq$ arithmetic:

$$
\begin{aligned}
(b^e)^d &= b^{ed} \\
&= b^{k(p-1)(q-1)+1} \text{ for some } k \\
&= b\, b^{(p-1)(q-1)}\, b^{(p-1)(q-1)} \ldots b^{(p-1)(q-1)} \\
&= b\, 1\, 1 \ldots 1 \\
&= b
\end{aligned}
$$

- **Note: we can also encrypt with *d* and decrypt with *e*.**

# Authentication




- Question: how does the receiver know that remote communicating entity is who it is claimed to be?

# Authentication Protocol (AP)

- AP 1.0
  - Alice to Bob: “I am Alice”
  - Problem: intruder “Trudy” can also send such a message
- AP 2.0
  - Authenticate source IP address is from Alice’s machine
  - Problem: IP Spoofing (send IP packets with a false address)
- AP 3.0: use a secret password
  - Alice to Bob: “I am Alice, here is my password” (e.g., telnet)
  - Problem: Trudy can intercept Alice’s password by sniffing packets

# Authentication Protocol

AP 3.1: encrypt the password

- Use a symmetric key known to Alice and Bob

  A to B: “I am A”, and A’s encrypted password

  B: if decrypted password is correct

  then A is verified

  else A is fraudulent

- Failure scenario: playback attack
  - Trudy can intercept Alice’s message and masquerade as Alice at a later time

# Authentication Using Nonces

- Problem with AP 3.1: same password is used for all sessions
- **Solution:** pick a "once-in-a-lifetime" number (nonce) for each session
- **AP 4.0**

  A to B: msg1 = "I am A" /* note: unencrypted message! */

  B to A: once-in-a-lifetime value, n

  A to B: msg2 = encrypt(n) /* use symmetric keys */

  B computes: if decrypt(msg2)==n

      then A is verified

      else A is fraudulent

# Authentication Using Public Keys

AP 4.0 uses symmetric keys for authentication

Question: can we use public keys?

**Symmetry: DA(EA(n)) = EA(DA(n)),** DA=private key of A, EA=public key of A

**AP 5.0**

A to B: msg = "I am A"

B to A: once-in-a-lifetime value, n

A to B: msg2 = DA(n)

B computes: if EA (DA(n))== n

then A is verified

else A is fraudulent

# Problems with AP 5.0

- Bob needs Alice's public key for authentication
  - Trudy can impersonate Alice to Bob
    - Trudy to Bob: msg1 = "I am Alice"
    - Bob to Alice: nonce n (Trudy intercepts this message)
    - Trudy to Bob: msg2= DT(n)
    - Bob to Alice: send me your public key (Trudy intercepts)
    - Trudy to Bob: send ET (claiming it is EA)
    - Bob: verify ET(DT(n)) == n and authenticates Trudy as Alice!!
- AP 5.0 is only as "secure" as public key distribution!

# Digital Signatures Using Public Keys

**Goals of digital signatures:**

- Sender cannot repudiate message ("I never sent that")
- Receiver cannot fake a received message

Suppose A wants B to "sign" a message M:

B sends M and DB(M) to A

A checks if EB ( DB(M)) == M

If yes, then B has signed M

# Message Digests

- Encrypting and decrypting entire messages using digital signatures is computationally expensive
- Message digests: like a checksum
  - Hash function H: converts variable length string to fixed length message digest
  - Digitally sign H(M)
  - A sends M and DA(H(M))
  - B computes: if H(M) == EA(DA(H(M))) then A sent the message and the **message hasn't been changed!**
- Property of H
  - It is infeasible to find any two messages x and y such that H(x) = H(y)