

# Network Security

## Concepts: Review

Raj Jain

Washington University in Saint Louis

Saint Louis, MO 63130

Jain@cse.wustl.edu

These slides are available on-line at:

<http://www.cse.wustl.edu/~jain/cse574-06/>



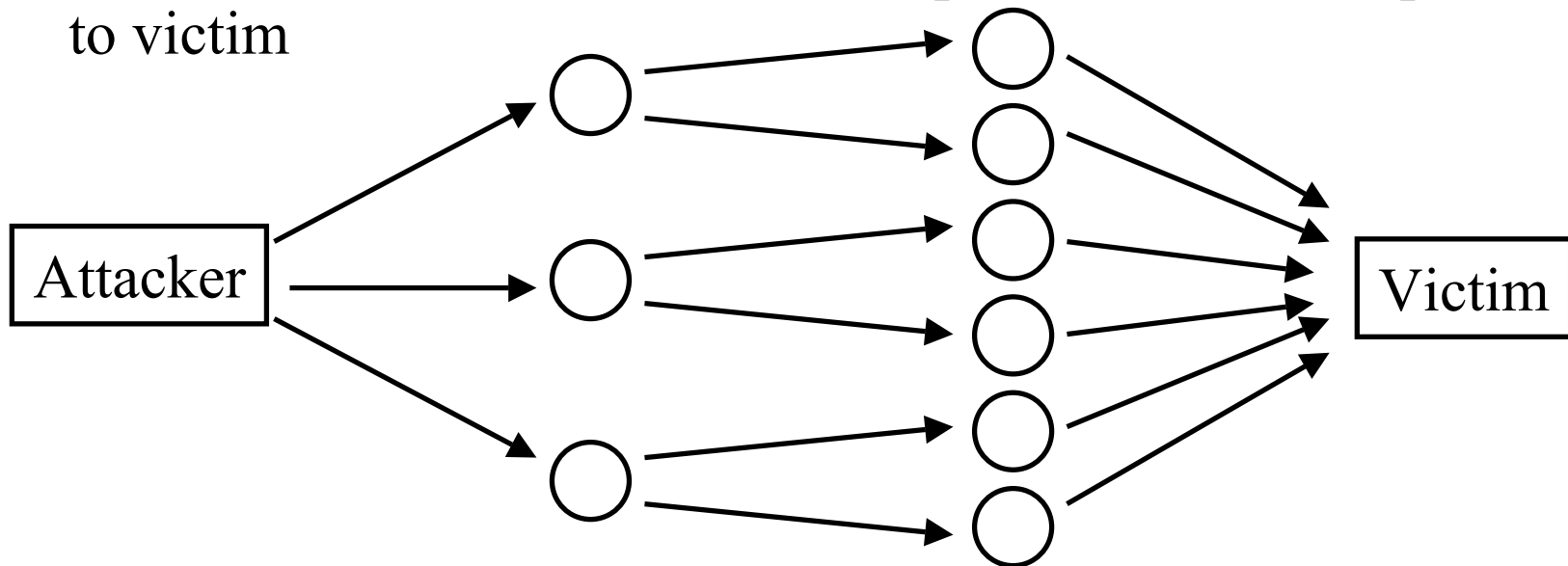
- ❑ Types of security attacks and solutions
- ❑ Secret Key and Public Key Encryption
- ❑ Hash Functions
- ❑ Message Authentication Code (MAC)
- ❑ Digital Signature and Digital Certificates
- ❑ RSA Public Key Encryption

# Types of Security Attacks

## ❑ Denial of Service (DoS)

- ❑ DoS by Flooding: Lots of packets from one node to victim. DoS on DNS or root name servers. ARP flooding, ping broadcasts, TCP SYN flooding.
- ❑ DoS by Forging: Send incorrect routing message

## ❑ Distributed DoS (DDoS): Lots of packets from multiple nodes to victim



# Security Attacks (Cont)

- ❑ **Sniffing**: Listen to unencrypted traffic
- ❑ **Replay**: Record and reuse messages later
- ❑ **Traffic Redirection**: Poison ARP tables in routers.
- ❑ **Reaction**: Send spurious packets; monitor the response.  
Challenge-response authentication.
- ❑ **Jamming**: RF interference.
- ❑ **Rogue AP**: Man-in-the-middle attacks.  
Easily deployed in public areas.  
Fake SSID
- ❑ **Fraud**: Criminal deception. E.g., identity theft
  - ❑ DNS query and responses are in clear. Can be spoofed by a man-in-the-middle. DNS cache poisoning.
  - ❑ BGP routing messages can be spoofed..

# Security Attacks (Cont)

- ❑ **Trojan Horse:** Programs with hidden functionality. Could be triggered when a specific time or condition.
- ❑ **Trap Doors:** Backdoor. Code segment to circumvent access control.
- ❑ **Virus:** A program that reproduces by introducing a copy of itself in other programs. Jump to Viral code and return to beginning.
- ❑ **Worms:** Creates copies of itself on other machines. Unlike virus, worms do not require user action. Morris worm spread by finding IP addresses on the machine. Slammer worm sent UDP packets to cause buffer overflow.
- ❑ **Buffer Overflow:** Overwrite code segments and execute code in data space. Many programming languages do enforce bound checking.

# Security Attacks (Cont)

- ❑ **Covert Communications Channel:** Hidden channel.
  - ❑ Capture electromagnetic radiations from keyboards, screens, and processors.
  - ❑ Pizza deliveries to White House
- ❑ **Steganography** or Information Hiding: Lower bits of pictures or music files.
- ❑ **Reverse Engineering:** dismantling and inspecting to infer internal function and structure. Code dumping and decompiling
- ❑ **Scavenging:** Acquisition of data from residue. Searching through rubbish bins. Buffer space in memory, deleted files on disks, bad blocks on disks
- ❑ **Cryptanalysis:** Find encryption key, encryption method, or clear text. Get plain-text and cipher text pairs.

# Security Solutions

- ❑ **Audits:** May including testing by a red team.  
Keep good system logs.
- ❑ **Formal methods:** Used to verify no human errors in the code and protocols.
- ❑ **Attack Graphs:** Show paths that an attacker can take to get access
- ❑ **Security Automata:** Security policies expressed as finite state machines.
- ❑ **Encryption:** Secret key and public key
- ❑ **Steganography:** Digital water marking. Information hidden in images, sound, or video can be used to find the origin of data.

# Security Solutions (Cont)

- ❑ **Obfuscation**: Make a concept confusing and difficult to understand. Common in politics. Write programs so that they can not be reverse engineered.
- ❑ **Virus Scanners**
- ❑ **Proof Carrying Code**: Mobile code contains a proof that it is safe.
- ❑ **Sandboxing**: Limiting access
- ❑ **Firewalls**: Scan and filter network traffic.
- ❑ **Red/black separation**: Handle sensitive and insensitive data on different machines.
- ❑ **Secure Hardware**: Temperproof. Physical security.

# Security Requirements

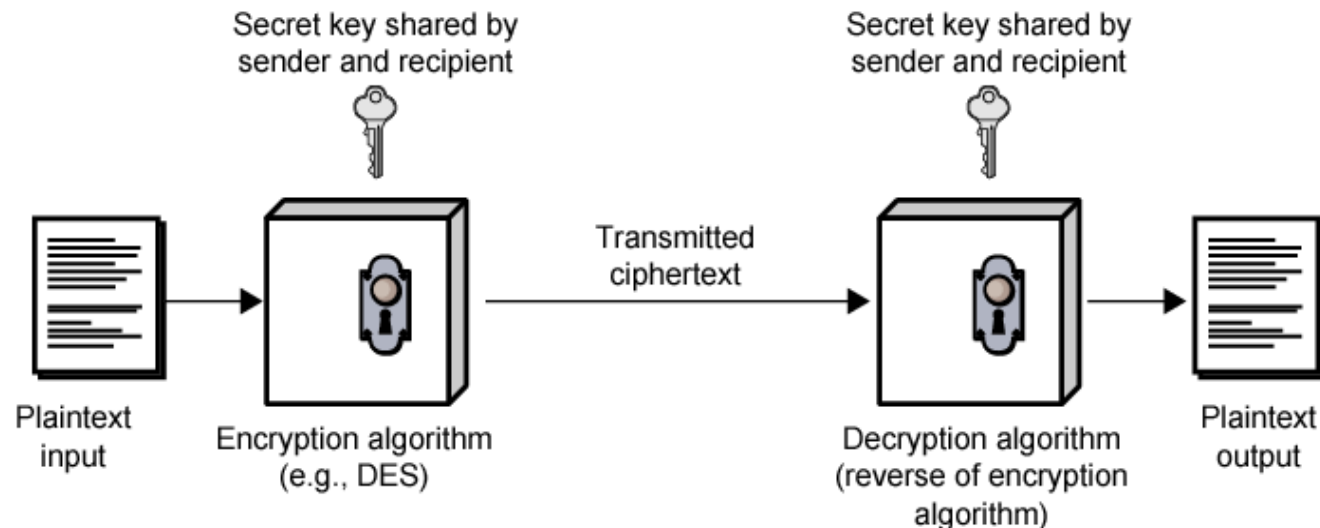


- ❑ **Integrity:** Received = sent?
- ❑ **Availability:** Legal users should be able to use.  
Ping continuously  $\Rightarrow$  No useful work gets done.
- ❑ **Confidentiality and Privacy:**  
No snooping or wiretapping
- ❑ **Authentication:** You are who you say you are.  
A student at Dartmouth posing as a professor canceled the exam.
- ❑ **Authorization** = Access Control  
Only authorized users get to the data
- ❑ **No repudiation:** Neither sender nor receiver can deny the existence of a message

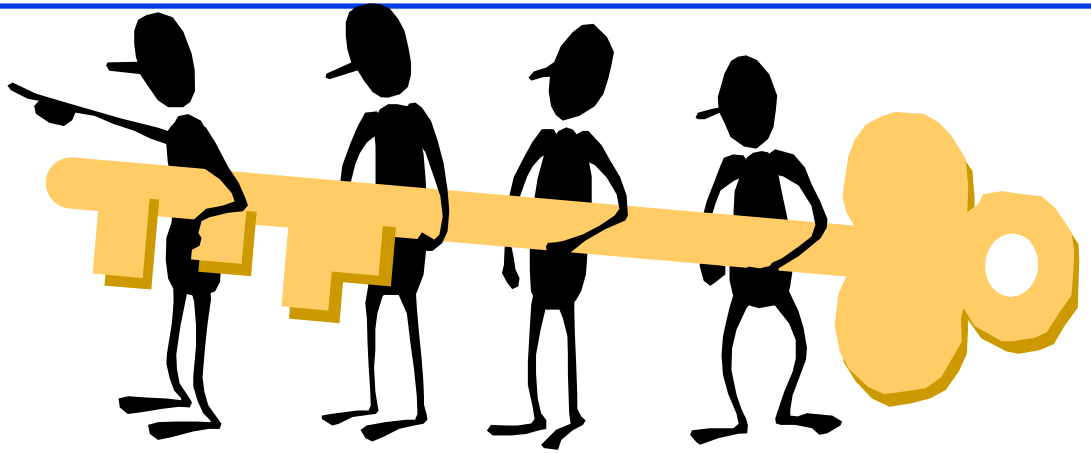
# Secret Key Encryption



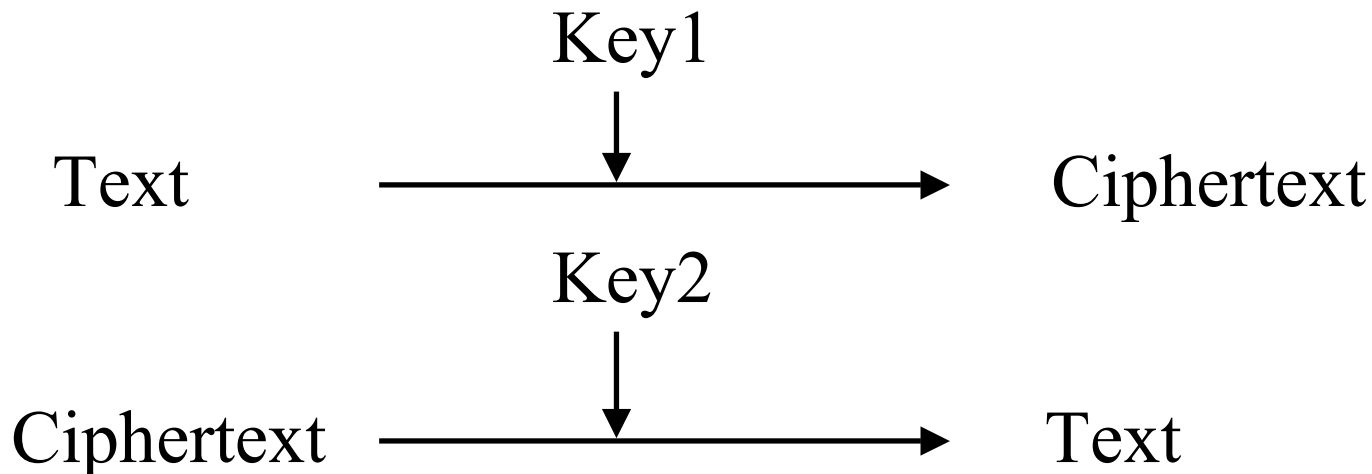
- ❑ Also known as symmetric encryption
- ❑  $\text{Encrypted\_Message} = \text{Encrypt}(\text{Key}, \text{Message})$
- ❑  $\text{Message} = \text{Decrypt}(\text{Key}, \text{Encrypted\_Message})$
- ❑ Example: Encrypt = division
- ❑  $433 = 48 \text{ R } 1$  (using divisor of 9)



# Public Key Encryption



- ❑ Invented in 1975 by Diffie and Hellman
- ❑  $\text{Encrypted\_Message} = \text{Encrypt}(\text{Key1}, \text{Message})$
- ❑  $\text{Message} = \text{Decrypt}(\text{Key2}, \text{Encrypted\_Message})$



# Public Key Encryption

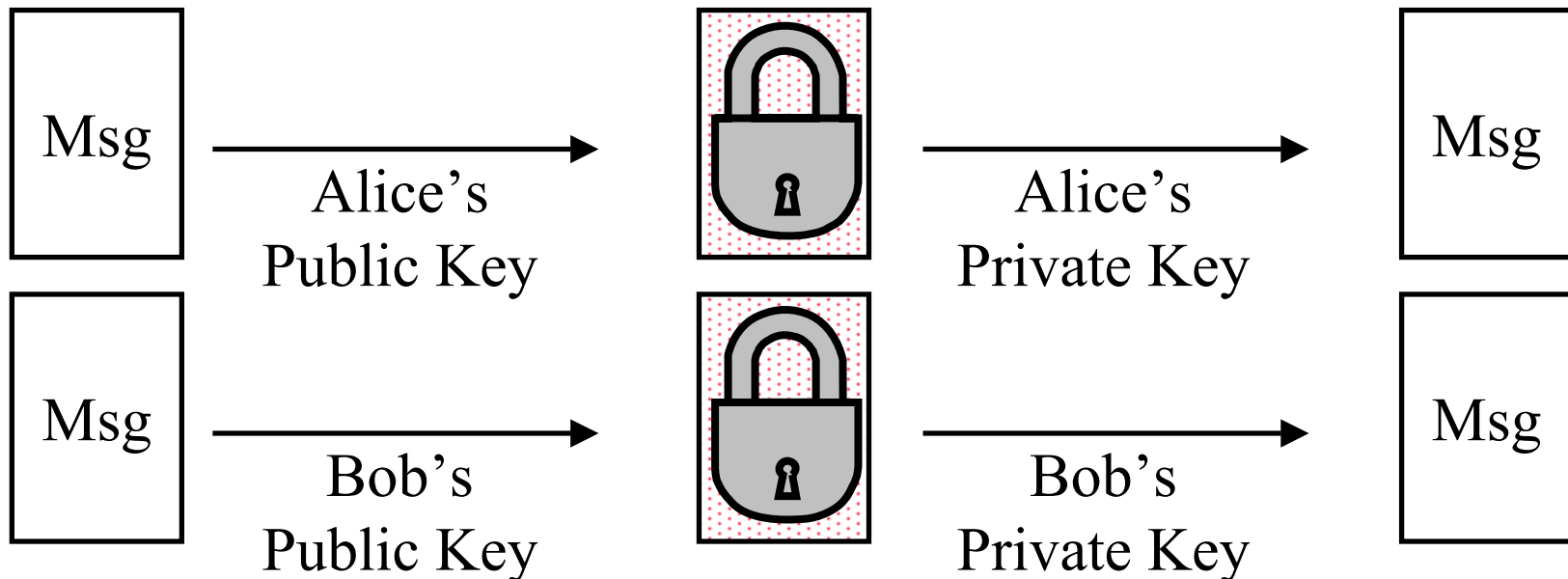
- ❑ RSA: Encrypted\_Message =  $m^3 \bmod 187$
- ❑ Message = Encrypted\_Message<sup>107</sup> mod 187
- ❑ Key1 = <3,187>, Key2 = <107,187>
- ❑ Message = 5
- ❑ Encrypted Message =  $5^3 = 125$
- ❑ Message =  $125^{107} \bmod 187 = 5$   
=  $125^{(64+32+8+2+1)} \bmod 187$   
=  $\{(125^{64} \bmod 187)(125^{32} \bmod 187) \dots$   
 $(125^2 \bmod 187)(125 \bmod 187)\} \bmod 187$

# Modular Arithmetic

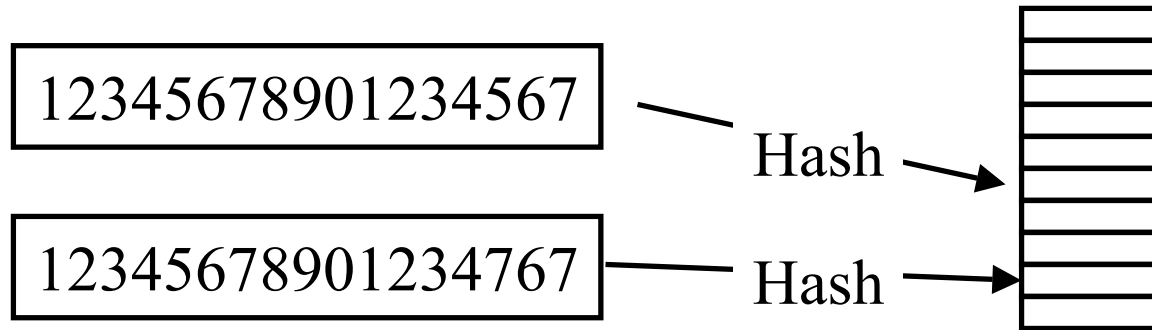
- $xy \bmod m = (x \bmod m)(y \bmod m) \bmod m$
- $x^4 \bmod m = (x^2 \bmod m)(x^2 \bmod m) \bmod m$
- $x^{ij} \bmod m = (x^i \bmod m)^j \bmod m$
- $125 \bmod 187 = 125$
- $125^2 \bmod 187 = 15625 \bmod 187 = 104$
- $125^4 \bmod 187 = (125^2 \bmod 187)^2 \bmod 187$   
 $= 104^2 \bmod 187 = 10816 \bmod 187 = 157$
- $125^8 \bmod 187 = 157^2 \bmod 187 = 152$
- $125^{16} \bmod 187 = 152^2 \bmod 187 = 103$
- $125^{32} \bmod 187 = 103^2 \bmod 187 = 137$
- $125^{64} \bmod 187 = 137^2 \bmod 187 = 69$
- $125^{64+32+8+2+1} \bmod 187 = 69 \times 137 \times 152 \times 104 \times 125 \bmod 187$   
 $= 18679128000 \bmod 187 = 5$

# Public Key (Cont)

- ❑ One key is private and the other is public
- ❑  $\text{Message} = \text{Decrypt}(\text{Public\_Key}, \text{Encrypt}(\text{Private\_Key}, \text{Message}))$
- ❑  $\text{Message} = \text{Decrypt}(\text{Private\_Key}, \text{Encrypt}(\text{Public\_Key}, \text{Message}))$



# Hash Functions



**Example:** CRC can be used as a hash  
(not recommended for security applications)

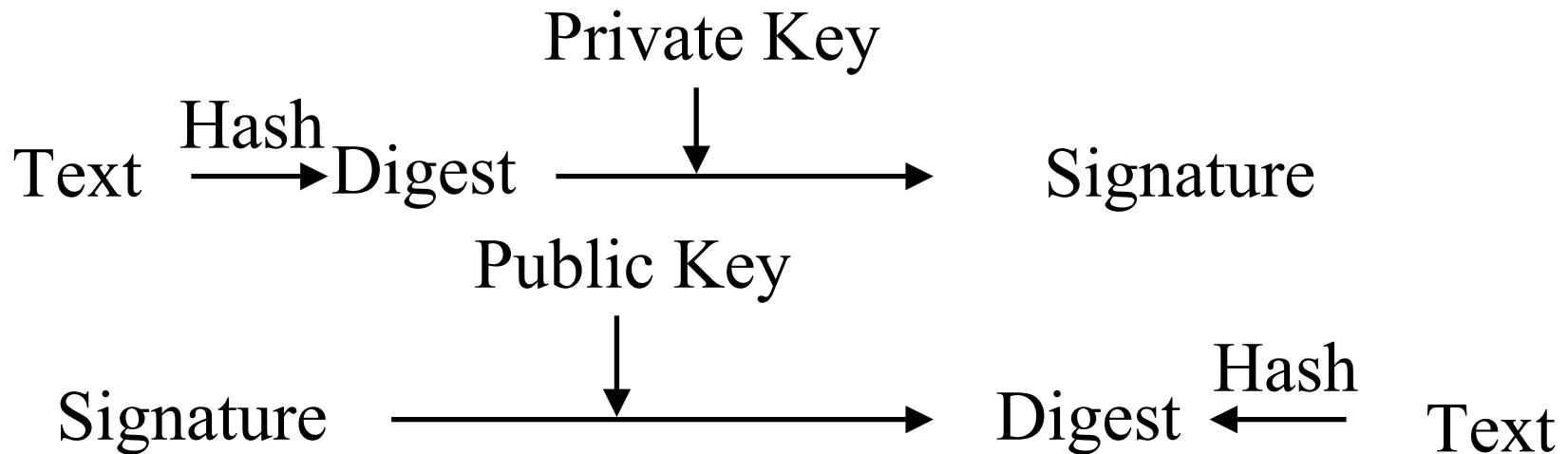
## Requirements:

1. Applicable to any size message
2. Fixed length output
3. Easy to compute
4. Difficult to Invert  $\Rightarrow$  Can't find  $x$  given  $H(x) \Rightarrow$  One-way
5. Difficult to find  $y$ , such that  $H(x) = H(y) \Rightarrow$  Can't change msg
6. Difficult to find *any* pair  $(x, y)$  such that  $H(x) = H(y)$   
 $\Rightarrow$  Strong hash

# Digital Signature

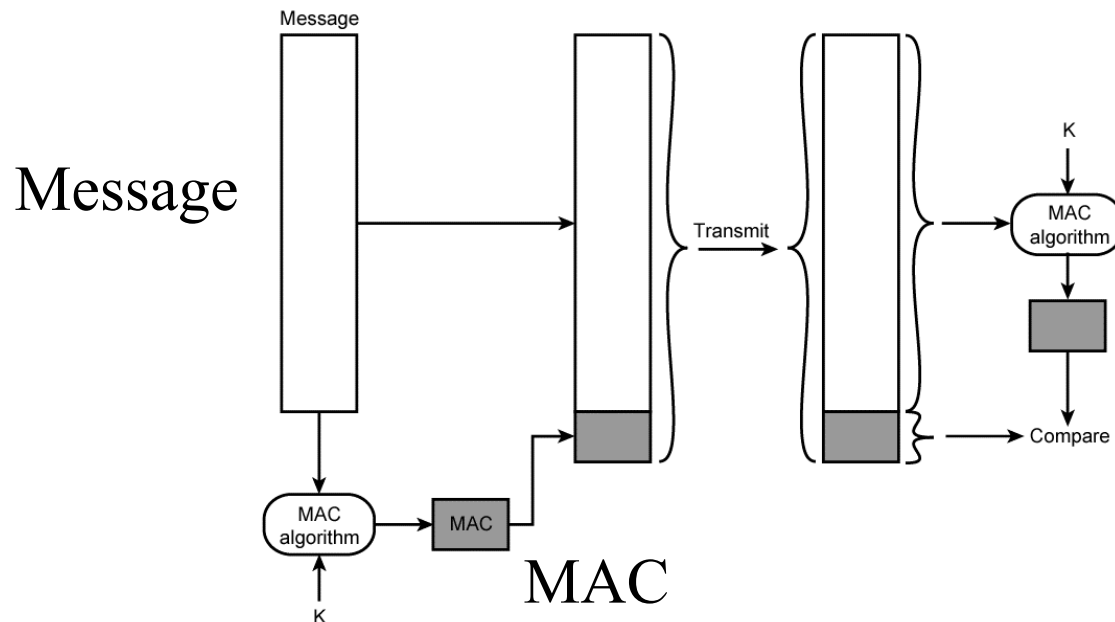


- ❑ Message Digest = Hash(Message)
- ❑ Signature = Encrypt(Private\_Key, Hash)
- ❑ Hash(Message) = Decrypt(Public\_Key, Signature)  
⇒ Authentic
- ❑ Also known as Message *authentication* code (MAC)



# Message Authentication Code (MAC)

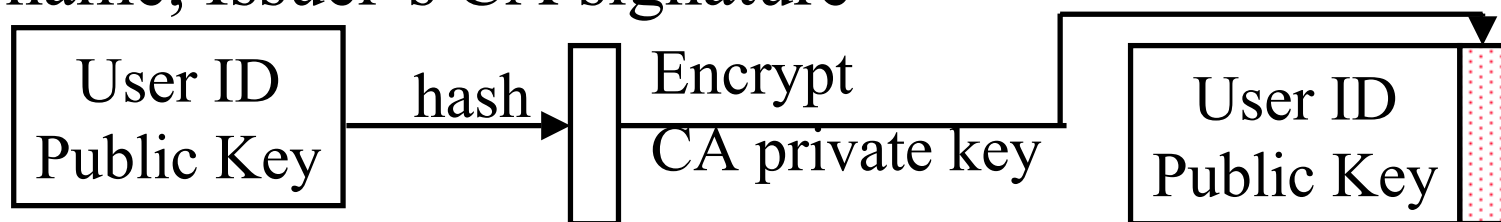
- ❑ Authentic Message = Contents unchanged + Source Verified
- ❑ May also want to ensure that the time of the message is correct
- ❑ Encrypt( {Message, CRC, Time Stamp}, Source's secret key)
- ❑ Message + Encrypt(Hash, Source's secret key)
- ❑ Message + Encrypt(Hash, Source's private key)



# Digital Certificates

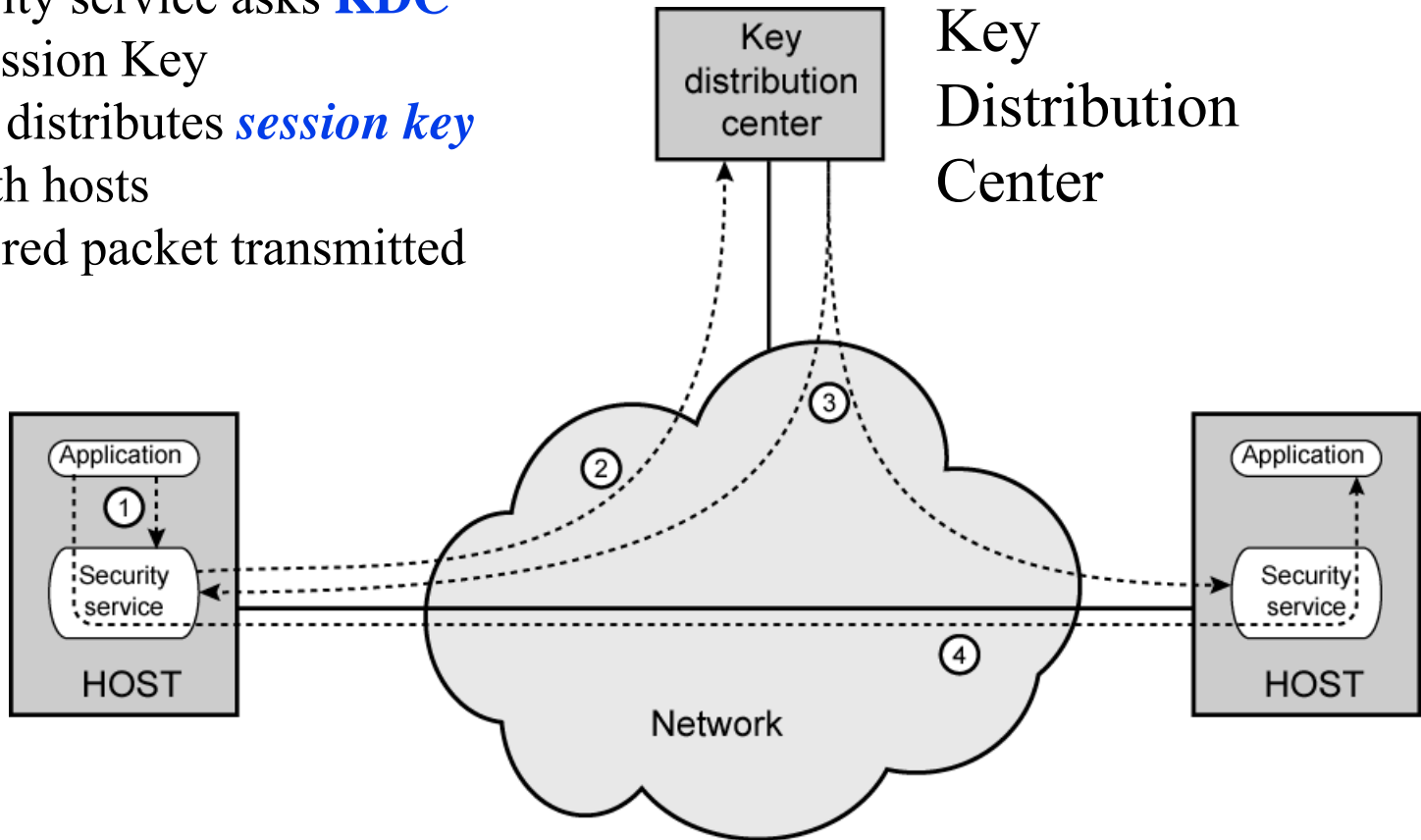


- ❑ Like driver license or passport
- ❑ Digitally signed by Certificate authority (CA) - a trusted organization
- ❑ Public keys are distributed with certificates
- ❑ CA uses its private key to sign the certificate  
⇒ Hierarchy of trusted authorities
- ❑ X.509 Certificate includes: Name, organization, effective date, expiration date, public key, issuer's CA name, Issuer's CA signature



# Key Distribution

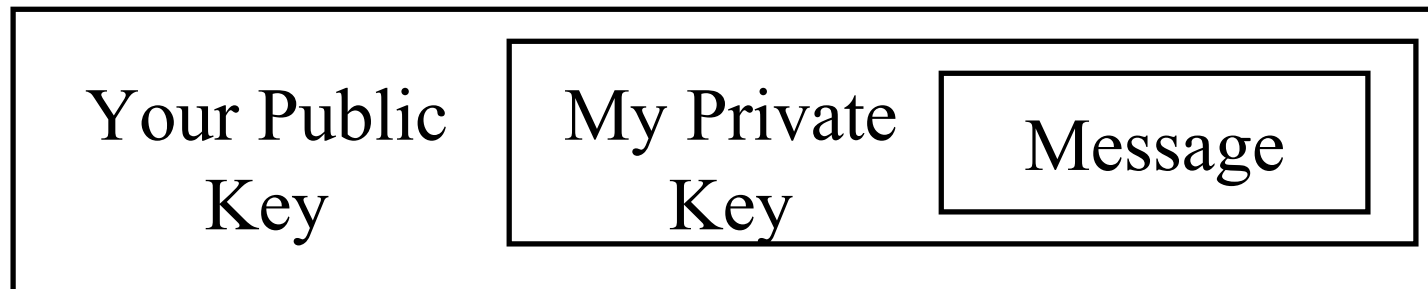
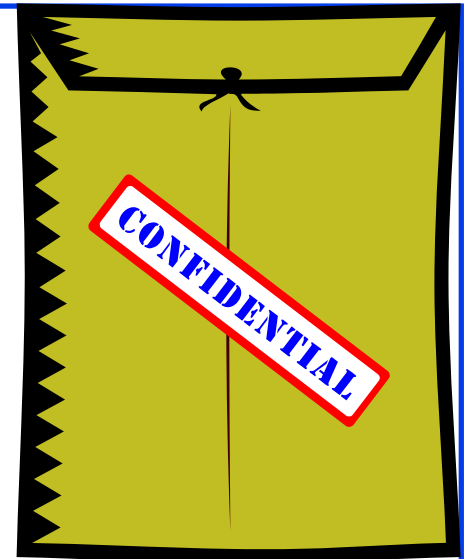
1. Application requests connection
2. Security service asks **KDC** for session Key
3. KDC distributes *session key* to both hosts
4. Buffered packet transmitted



KDC shares a secret key with each Host.

# Confidentiality

- ❑ User 1 to User 2:
- ❑ Encrypted\_Message  
= Encrypt(Public\_Key2,  
Encrypt(Private\_Key1, Message))
- ❑ Message = Decrypt(Public\_Key1,  
Decrypt(Private\_Key2, Encrypted\_Message)  
⇒ Authentic and Private



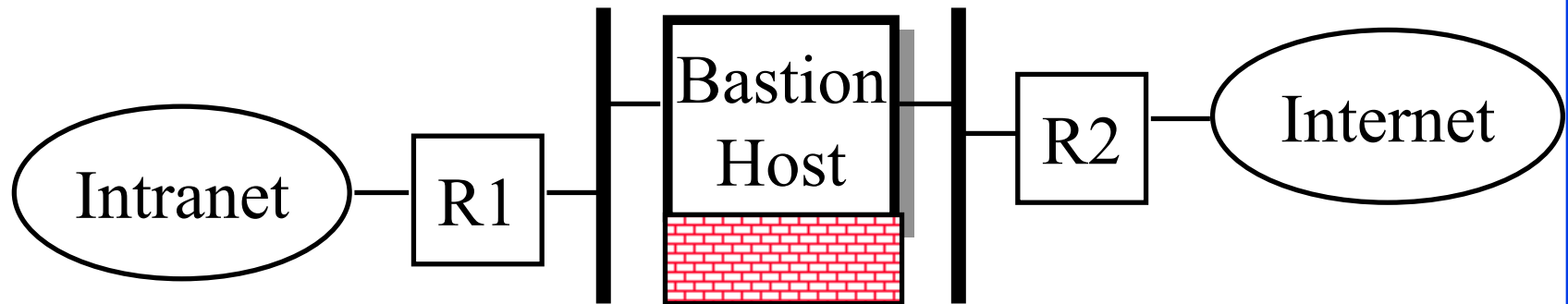
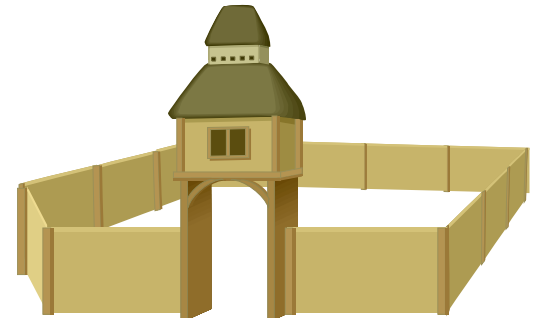
# RSA Public Key Encryption

- ❑ Ron Rivest, Adi Shamir, and Len Adleman at MIT 1978
- ❑ Both plain text  $M$  and cipher text  $C$  are integers between 0 and  $n-1$ .
- ❑ Key 1 =  $\{e, n\}$ ,  
Key 2 =  $\{d, n\}$
- ❑  $C = M^e \bmod n$   
 $M = C^d \bmod n$
- ❑ How to construct keys:
  - ❑ Select two large primes:  $p, q, p \neq q$
  - ❑  $n = p \times q$
  - ❑ Calculate  $\Phi = (p-1)(q-1)$
  - ❑ Select  $e$ , such that  $\text{lcd}(\Phi, e) = 1; 0 < e < \Phi$
  - ❑ Calculate  $d$  such that  $de \bmod \Phi = 1$

# RSA Algorithm: Example

- ❑ Select two large primes:  $p, q, p \neq q$   
 $p = 17, q = 11$
- ❑  $n = p \times q = 17 \times 11 = 187$
- ❑ Calculate  $\Phi = (p-1)(q-1) = 16 \times 10 = 160$
- ❑ Select  $e$ , such that  $\text{lcd}(\Phi, e) = 1; 0 < e < \Phi$   
say,  $e = 7$
- ❑ Calculate  $d$  such that  $de \text{ mod } \Phi = 1$ 
  - ❑  $160k+1 = 161, 321, 481, 641$
  - ❑ Check which of these is divisible by 7
  - ❑ 161 is divisible by 7 giving  $d = 161/7 = 23$
- ❑ Key 1 =  $\{7, 187\}$ , Key 2 =  $\{23, 187\}$

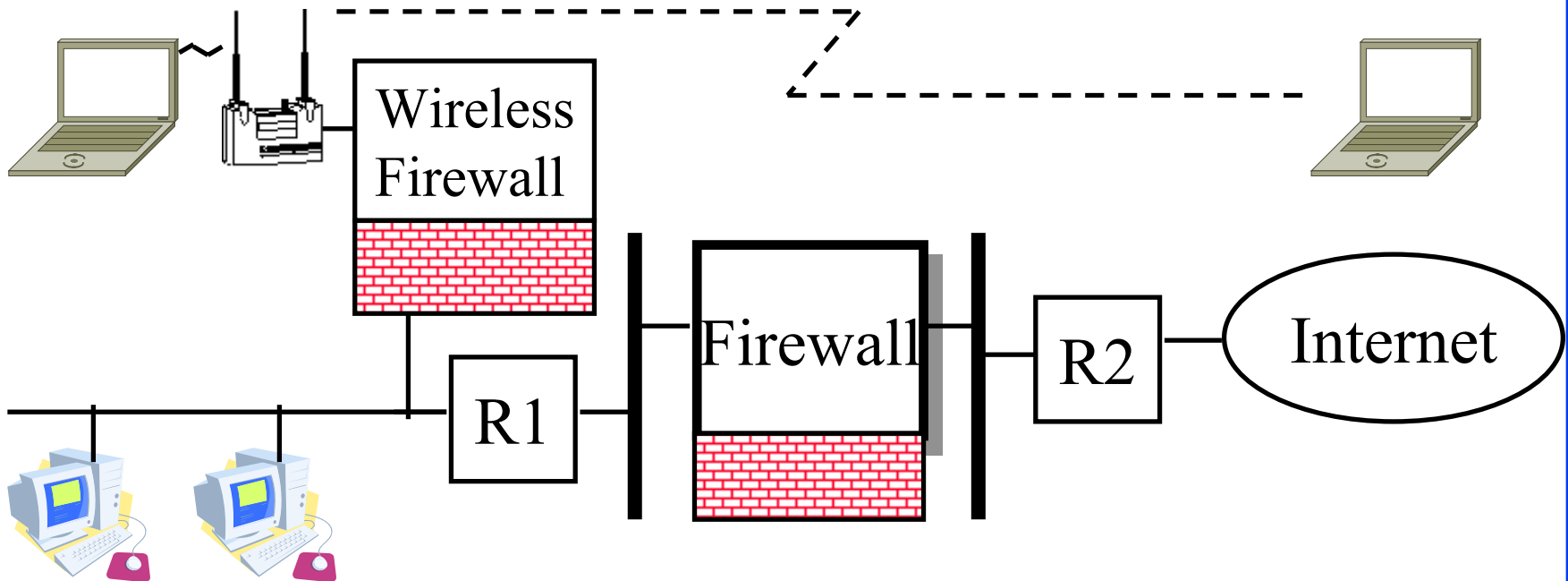
# Firewall: Bastion Host



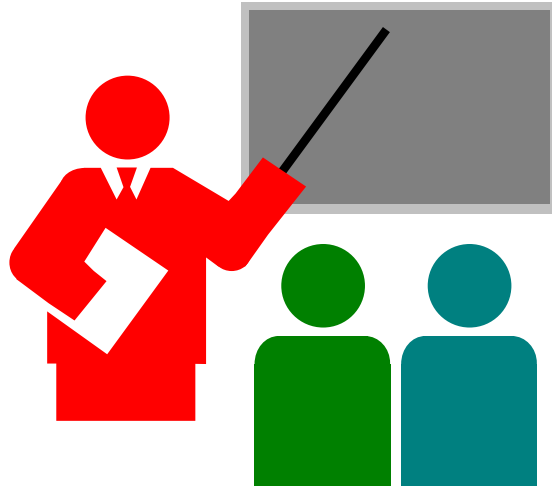
- ❑ Bastions overlook critical areas of defense, usually having stronger walls
- ❑ Inside users log on the Bastion Host and use outside services.
- ❑ Later they pull the results inside.
- ❑ One point of entry. Easier to manage security.

# Wireless Firewall

- ❑ Wireless Access point allows access to inside resources from outside
  - ⇒ Wireless hosts should be behind a “wireless firewall”



# Summary



- ❑ Types of attacks: DoS, DDoS, Virus, Worm, ...
- ❑ Types of solutions: Security audit, Encryption, firewalls, ...
- ❑ Secret Key and Public Key Encryption
- ❑ Secure Hash Functions
- ❑ Message Authentication Code (MAC)
- ❑ Digital Signature and Digital Certificates
- ❑ RSA Public Key Encryption based on exponentiation

# Homework

- **Exercise 1:** If  $125^{2048} \bmod 187$  is 86, what is  $125^{4096} \bmod 187$ ?
- **Exercise 2:** In a public key system using RSA, you intercept the cipher text  $C=10$  sent to a user whose public key is  $e=5$ ,  $n=35$ . What is the plain text  $M$ ?

# References

- ❑ W. Stallings, “Data and Computer Communications,” 7<sup>th</sup> Ed, Prentice Hall, 2004, Chapter 21,
- ❑ R. R. Brooks, “Disruptive Security Technologies with Mobile Code and Peer-to-Peer Networks,” CRC Press, 2004, pp. 5-55, ISBN:0849322723
- ❑ Sudhir Dixit and R. Prasad, (Eds), “Wireless IP and Building the Mobile Internet,” Artech House, 2002, pp. 587-617
- ❑ Frank Ohrtman, “Voice over 802.11,” Artech House, 2004, pp. 97-126, ISBN: 1580536778