

CS2105

An *Awesome* Introduction to Computer Networks

Network Security



Department of Computer Science
School of Computing

Adapted from Prof Roger &

Some material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved

Network Security



Lecture goals:

- ❖ Understand principles of network security:
 - confidentiality
 - authentication
 - message integrity

Chapter 8 roadmap

8.1 What is network security?

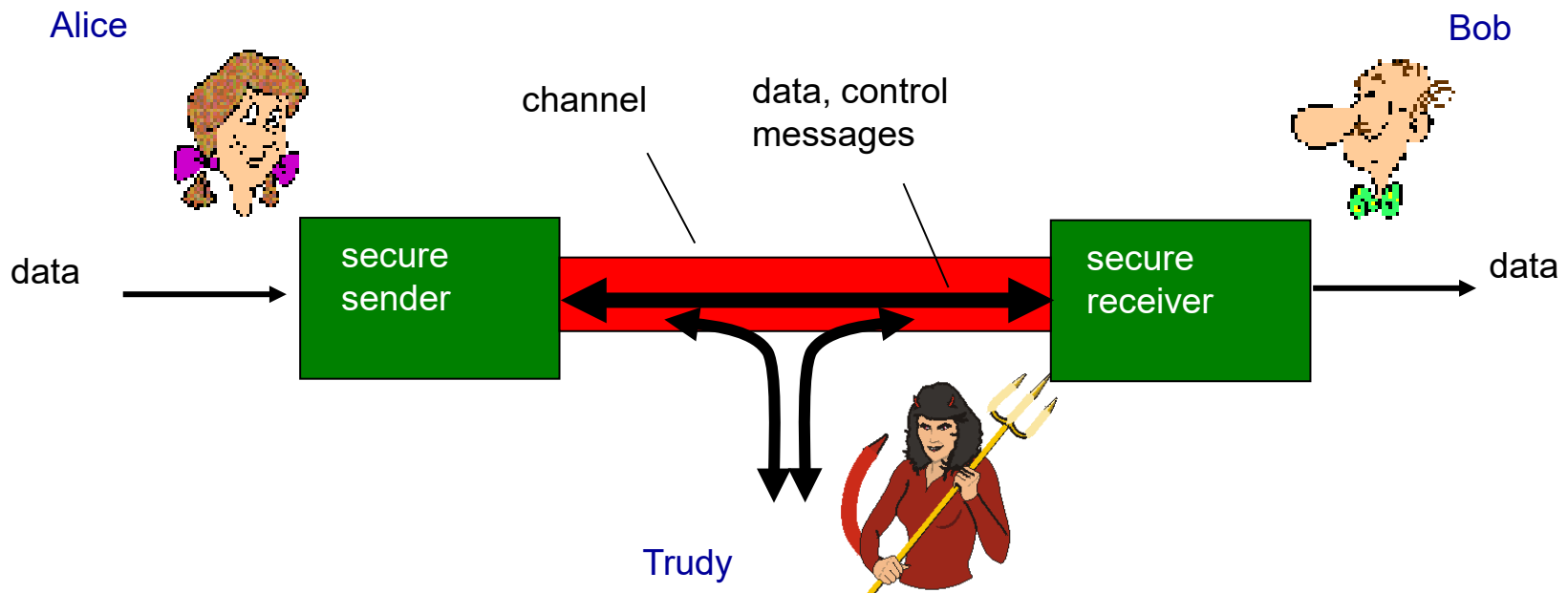
8.2 Principles of cryptography

8.3 Message integrity and digital signatures



8.9 Operational security: firewalls

Alice, Bob, Trudy

- Bob, Alice (lovers!) want to communicate
- Trudy (intruder) may
 - *Eavesdrop*
 - *Delete*
 - *Add messages*
- Bob (or Alice) may *repudiate*
- Bob, Alice (lovers!) want to communicate “*securely*”



Who might Bob, Alice be?

- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions
(e.g., on-line purchases) 
- On-line banking client/server 
- DNS servers
- Routers exchanging routing table updates

There are bad guys (and girls) out there!

Q: What can a “bad guy” do?

A: A lot!

- *eavesdrop*: intercept messages
- actively *insert* messages into connection
- *impersonation*: can fake (spoof) source address in packet (or any field in packet)
- *hijacking*: “take over” ongoing connection by removing sender or receiver, inserting oneself in place (Man in the middle)
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

What is network security?

❖ *Confidentiality:*

- ❖ only sender, intended receiver should “*understand*” message contents

❖ *Authentication:*

- ❖ sender, receiver want to *confirm identity* of each other

❖ *Message integrity:*

- ❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ *Access and availability:*

- ❖ services must be accessible and available to users

Motivation: Countering Trudy

- Bob, Alice (lovers!) want to communicate “*securely*”
- What would you, as Alice, do?
 - Make the Message Physically secure.
 - Use a language only know to you
 - Code Words
 - Code Language
 - Language of a gang of friends
 - Navajo Code Talkers in WWII
 - book cipher
 - Is it good?
 - *Confidentiality*: if the code language is secret, Trudy cannot understand.
 - *Authentication*: if the code language is secret, only Alice and Bob can write it.
 - *Message Integrity*: if the code language has an inviolable property.



cryptographic techniques are *inextricably* woven into authentication, message integrity, and confidentiality

Aim:

- Confidentiality
- Authentication
- Message Integrity

Chapter 8 roadmap

8.1 What is network security?

8.2 *Principles of cryptography*

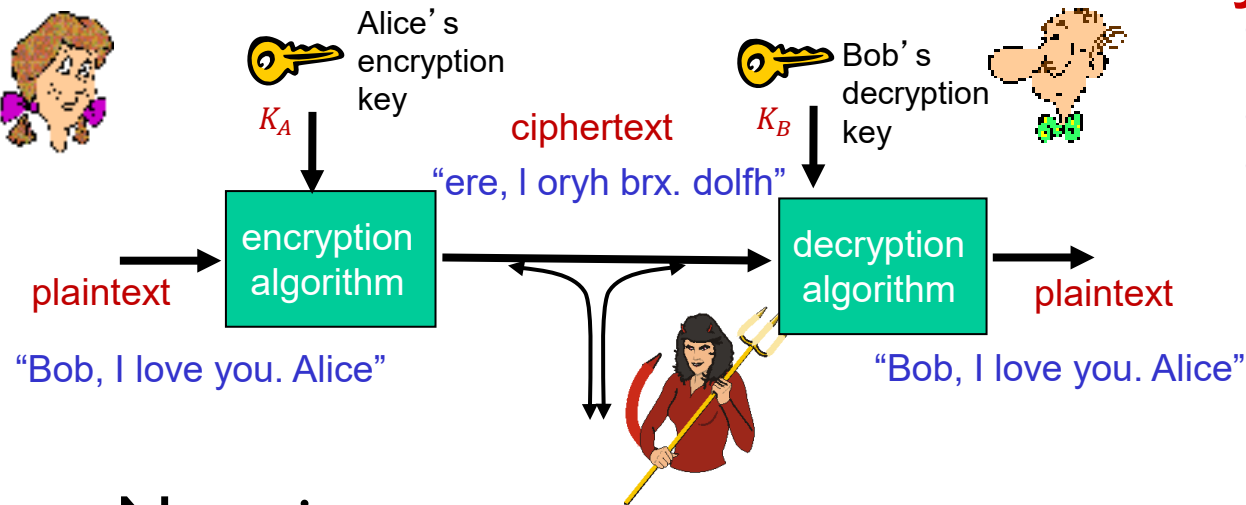
8.3 Message integrity and digital signatures

8.9 Operational security: firewalls

The language of cryptography

- Cryptography:
 - κρυπτός (kryptos) - *hidden*
 - γράφειν (graphia) – *writing*
- Cryptographic techniques
 - allow a sender to *disguise* data so that an intruder can gain no information from the intercepted data.
 - allow the receiver to *recover* the original data from the disguised data.

The language of cryptography



Jargon Alert:

- **Plaintext:** Message in the original form
- **Ciphertext:** Encrypted Message
- **Key:** A string of numbers or characters, provided as input parameter to the encryption/decryption algorithm

Notation

- ❖ m : plaintext message
- ❖ $K_A(.)$: Encryption algorithm, with key K_A
 - ❖ $K_A(m)$: ciphertext
- ❖ $K_B(.)$: Decryption algorithm, with key K_B
 - ❖ $K_B(K_A(m)) = m$

K_A is an abuse of notation, The Encryption and Decryption algorithm may not be same

Types of Cryptography

Based on the values of the keys, there are *two* types of encryption

❖ Symmetric Key Cryptography

- ❖ Sender and receiver use the *same key*

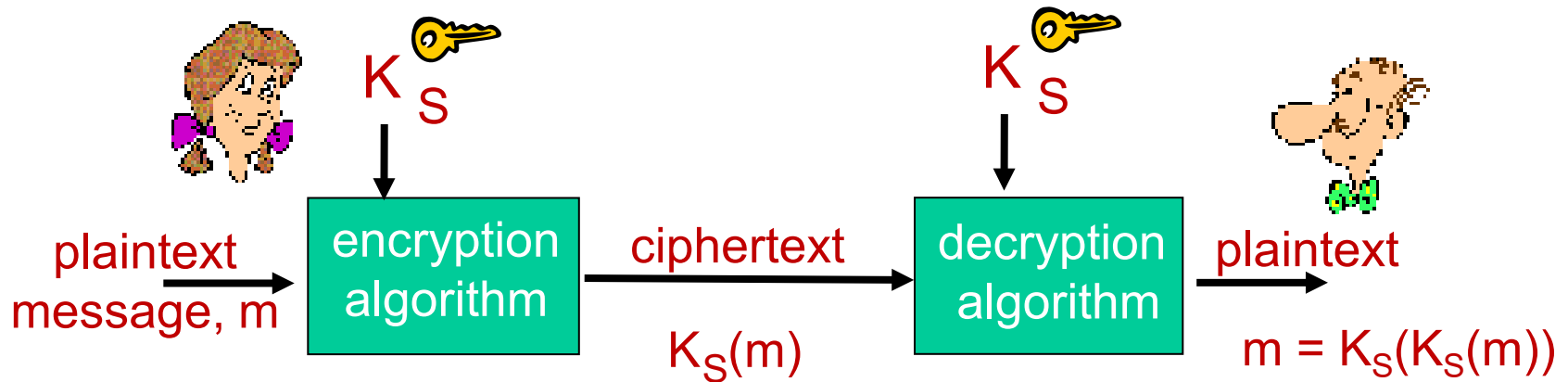
- ❖ $K_A = K_B$ (We are talking about the key not the algorithms)

❖ Asymmetric Key Cryptography AKA Public key Cryptography

- ❖ Sender and receiver use *different key*

- ❖ $K_A \neq K_B$ (We are talking about the key not the algorithms)

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

Q: how do Bob and Alice agree on key value?

Ans: Will need to decide on the common key prior to communication via some other secure means, like face to face meeting

Caesar's cipher

- This method is named after Julius Caesar, who used it in his private correspondence
- Is a form of *Substitution cipher*: substituting one thing for another
- Fixed shift of alphabet
 - e.g., right shift by 3:

abcdefghijklmnopqrstuvwxyz
↓ ↓
defghijklmnopqrstuvwxyzabc

e.g.: plaintext: the quick brown fox
 ciphertext: wkh txlfn eurzq ira



Encryption key: only need shift number, 25 possible values

Weakness: Easy to break with brute force search

Monoalphabetic cipher

- Substitute one letter for another
 - Is a *Substitution cipher*

abcdefghijklmnopqrstuvwxyz
↓ ↓
mnbvcxz asdfghjklpoiuytrewq

e.g.: Plaintext: bob, i love you. alice

ciphertext: nkn, s gktc wky. mgsbc



Encryption key: mapping from set of 26 letters to set of 26 letters, **26!** Mappings possible

Monoalphabetic cipher

- **26!** ($\sim 10^{26}$) mappings possible are not as much as they seem
- We can break it with *Statistical Analysis*
 - letters **e** (13%) and **t** (9%) are the most frequent letters
 - knowing that particular **two-and three-letter** occurrences of letters appear quite often together (for example, “in,” “it,” “the,” “ion,” “ing,”
 - If the intruder has some knowledge about the possible contents of the message, then it is even easier to break the code.
 - if Trudy might suspect that the names “**bob**” and “**alice**” appear in the text.
 - and had a copy of the example ciphertext message above, then she could immediately determine seven of the 26 letter pairings

Breaking an encryption scheme

- **Ciphertext only attack:** Trudy has ciphertext she can analyze
- **Known-plaintext attack:** Trudy has plaintext corresponding to ciphertext
 - e.g., in monoalphabetic cipher, Trudy determines pairings for *a,l,i,c,e,b,o*,
- **Chosen-plaintext attack:** Trudy can get ciphertext for chosen plaintext
 - e.g., in monoalphabetic cipher, Trudy determines gets Alice to send *“The quick brown fox jumps over the lazy dog,”*

Polyalphabetic encryption

- What is the fundamental weakness of Monoalphabetic Cipher?
 - Each letter has only one mapping
- Solution?
 - Use multiple mappings
- E.g.
 - Use n substitution ciphers, C_1, C_2, \dots, C_n
 - Define a cycling pattern:
 - e.g., $n=4$: C_1, C_3, C_4, C_3, C_2
 - for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
 - dog: d from C_1 , o from C_3 , g from C_4



Encryption key: n substitution ciphers, and cycling pattern

Polyalphabetic encryption

- Example:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
C_1	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
C_2	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
C_3	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i

Cycling Pattern: $C_1 C_3 C_3 C_2$

e.g.: plaintext: bob, i love you. alice
 ciphertext: **exk**, o **oxek** **bxd**. **durih**

Block Ciphers

- The message to be encrypted is processed in blocks of K bits.
- For example,
 - if $K = 64$, the message is broken into 64-bit blocks
 - each block is encrypted independently.
- To encode a block, the cipher uses a one-to-one mapping
- E.g.: $K = 3$
 - Input: 010110001111
 - Encrypted output: 101000111001
- Number of keys: $2^K!$
 - $2^{64}!$ is an astronomical value.

Input	Output
000	110
001	111
010	101
011	100
100	011
101	010
110	000
111	001

Block Cipher

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit block
- How secure is DES?
 - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in *less than a day*
- Making DES more secure:
 - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard

- Symmetric-key NIST standard, replaced DES (Nov 2001)
- 128 bit blocks; 128, 192, or 256 bit keys
- How secure is AES?
 - Machine capable of Brute force decryption DES in 1 sec on DES, takes *149 trillion years* for 128-AES

Public Key Cryptography



symmetric key crypto

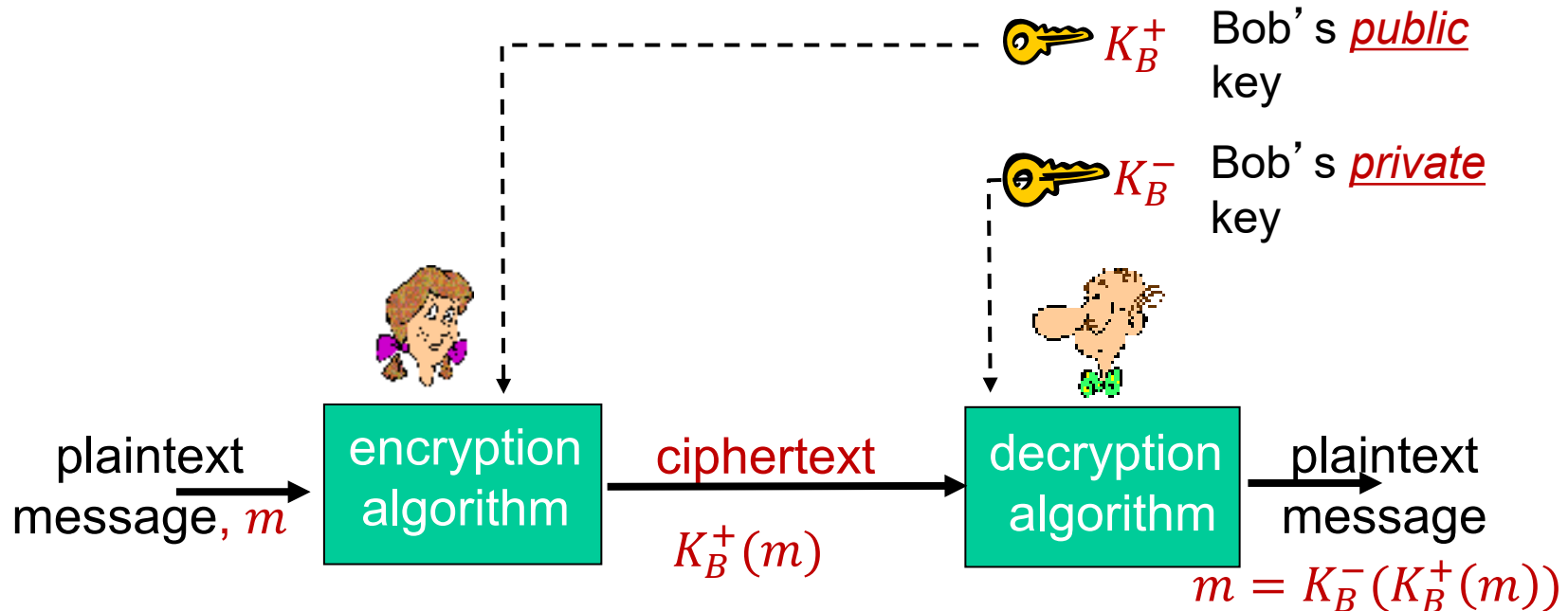
- *Drawback*: requires sender, receiver know shared secret key
- *Q*: how to agree on key in first place (particularly if never “met”)?



public key crypto

- sender, receiver *do not share* secret key
- Sender uses a *public* encryption key known to *all*
- receiver uses a *private* decryption key known *only to receiver*
- radically different approach [Diffie-Hellman’76, RSA’78]

Public key cryptography



Public key encryption algorithms

Requirements:

- ① need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$m = K_B^-(K_B^+(m))$$

- ② given public key K_B^+ , it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adleman algorithm



Prerequisite: modular arithmetic

- $x \bmod n$ = remainder of x when divide by n
- facts:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
 - $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$
- Thus
$$(a \bmod n)^d \bmod n = a^d \bmod n$$
- example:
 - $a = 14, n = 10, d = 2$
 - $a^d \bmod n = 14^2 \bmod 10 = 6$
 - $(a \bmod n)^d \bmod n = (14 \bmod 10)^2 \bmod 10 = 16 \bmod 10 = 6$

RSA: getting ready

- *message*: just a bit pattern
 - bit pattern can be uniquely represented by an *integer number*
 - thus, encrypting a message is equivalent to encrypting a number

example:

- $m = 10010001$.
- This message is uniquely represented by the decimal number *145*.
- to encrypt *m*, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers p, q .
2. compute $n = pq, z = (p - 1)(q - 1)$
3. choose e (with $e < n$) that has no common factors with z (i.e., e and z are “relatively prime”).
4. choose d such that $ed - 1$ is exactly divisible by z .
(in other words: $ed \bmod z = 1$).
5. public key is (n, e) . private key is (n, d) .

$\underbrace{\hspace{1.5cm}}$
 K_B^+

$\underbrace{\hspace{1.5cm}}$
 K_B^-

RSA: encryption, decryption

0. given (n, e) and (n, d) as computed above
1. to encrypt message m (Note: $m < n$), compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c , compute

$$c^d \bmod n$$

*magic
happens!*

$$\underbrace{(m^e \bmod n)}_c^d \bmod n = m$$

RSA example: Bob Chooses

1. $p = 5, q = 7.$

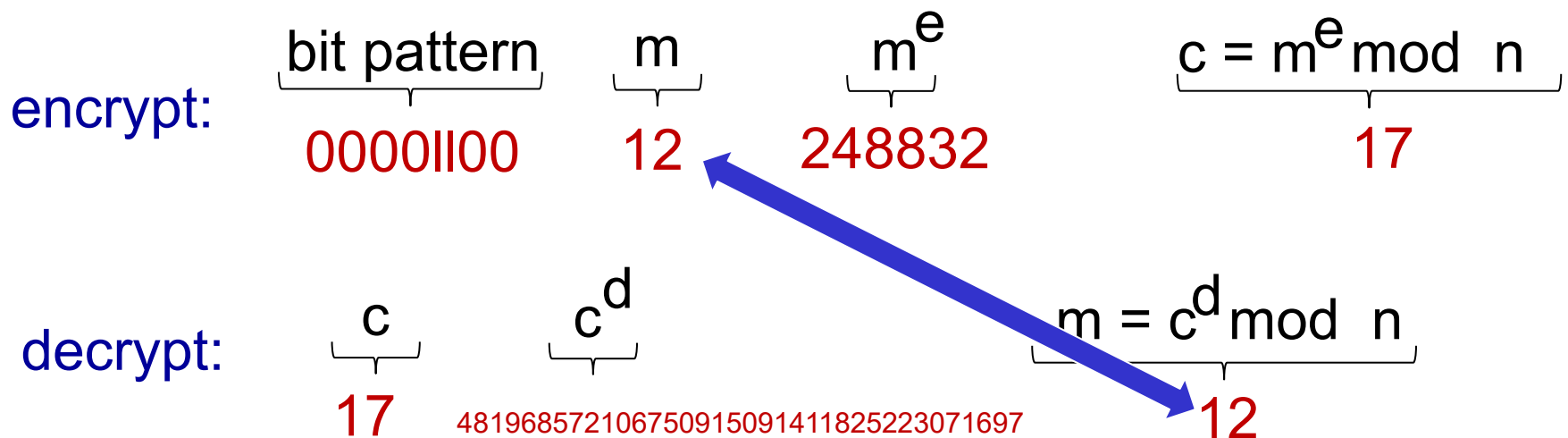
1. $n = pq = 35,$

2. $z = (p - 1)(q - 1) = 24$

2. $e = 5$ (with $e < n$ & e and z are “relatively prime”).

3. $d = 29$ such that $ed \bmod z = 1.$

encrypting 8-bit messages.



RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

use **public key** first,
followed by
private key

use **private key** first,
followed by
public key

result is the same!

Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

follows directly from modular arithmetic:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

RSA in practice: session keys

- Exponentiation in **RSA** is computationally intensive
- **DES** is at least 100 times faster than RSA, but needs prior knowledge of Key K_S
- Can we Combine them?
 - Select a Key K_S
 - Use RSA to transfer K_S
 - Use K_S as the symmetric key in DES for encrypting data for this session
- The symmetric key K_S , is called the **session key**.

What is network security?

❖ *Confidentiality:*

- ❖ only sender, intended receiver should “*understand*” message contents

❖ *Authentication:*

- ❖ sender, receiver want to *confirm identity* of each other

❖ *Message integrity:*

- ❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ *Access and availability:*

- ❖ services must be accessible and available to users

Chapter 8 roadmap

8.1 What is network security?

8.2 Principles of cryptography

8.3 *Message integrity and digital signatures*

8.9 Operational security: firewalls

Message Integrity

❖ *Message integrity:*

- ❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ Have we *seen* this requirement earlier?

- ❖ Does “*error detection*” fit the bill?

❖ Yes

- ❖ Checksum
- ❖ Parity
- ❖ CRC

Internet checksum

Internet checksum:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

Consider the given message: “IOU100.99BOB”

<u>message</u>	<u>ASCII format</u>		<u>message</u>	<u>ASCII format</u>
I O U 1	49 4F 55 31		I O U <u>9</u>	49 4F 55 <u>39</u>
0 0 . 9	30 30 2E 39		0 0 . <u>1</u>	30 30 2E <u>31</u>
9 B O B	39 42 D2 42		9 B O B	39 42 D2 42
<hr/>			<hr/>	
B2 C1 D2 AC		different messages but identical checksums!	B2 C1 D2 AC	

- It is easy to find another message with same checksum value
- checksum is designed to detect *accidental* errors not attacks!

CRC



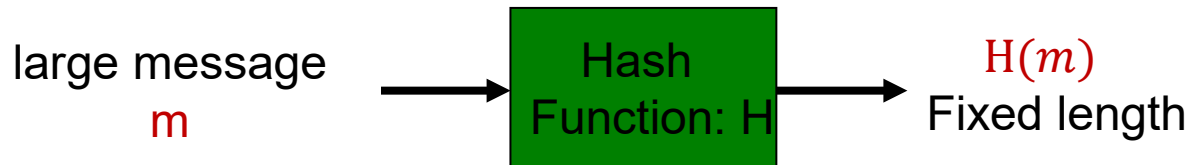
Better than Checksum

- Yet poor
- Output is biased to the input
 - Minor changes in input produce minor changes in output
- E.g.
 - “Steven has fifteen white tables.” and “Maria has nine red beds.”
 - Both have CRC32 checksum = **248210933**
 - “Joe has fourteen magenta things.” and “Lars has thirteen black balls.”
 - Both have CRC32 checksum = **93832682**

Cryptographic Hash Function

Hash function:

- If a function $H(.)$ that takes an input m and produces fixed-size msg digest (*fingerprint*)
 - many-to-1



Cryptographic Hash function:

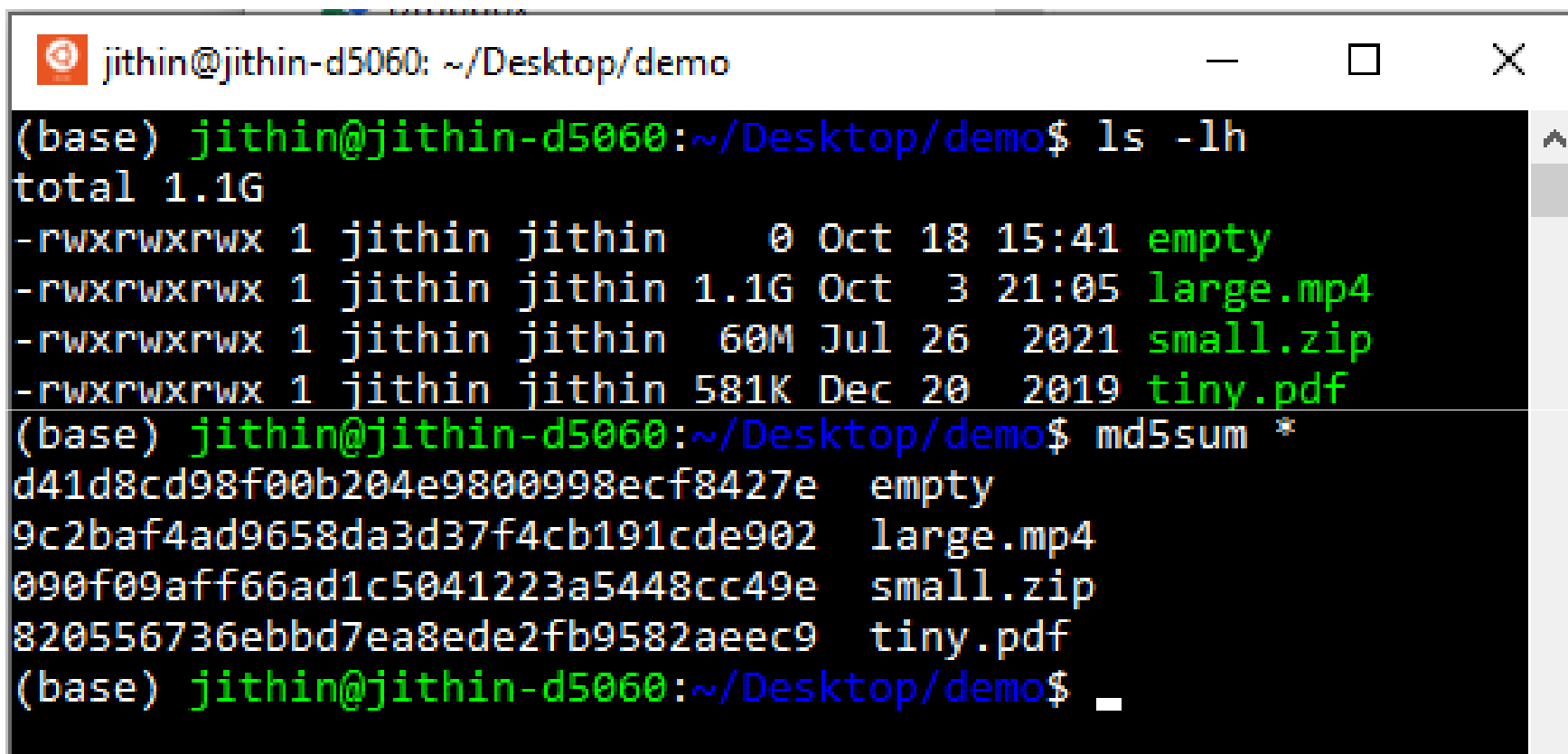
- Is a hash function such that it is computationally infeasible to find any two different messages x and y such that $H(x) = H(y)$
- Informally, this property means that it is computationally infeasible for an intruder to substitute one message for another message

Cryptographic Hash Function

- **MD5** hash function widely used (RFC 1321)
 - computes 128-bit message digest.
- **SHA-1** is also used
 - US standard [NIST]
 - 160-bit message digest
- Both SHA-1 and MD5 are **cryptographically broken**
 - NIST formally deprecated use of SHA-1 in 2011
 - Replaced by SHA-2, SHA-3

Hash function, e.g., md5sum

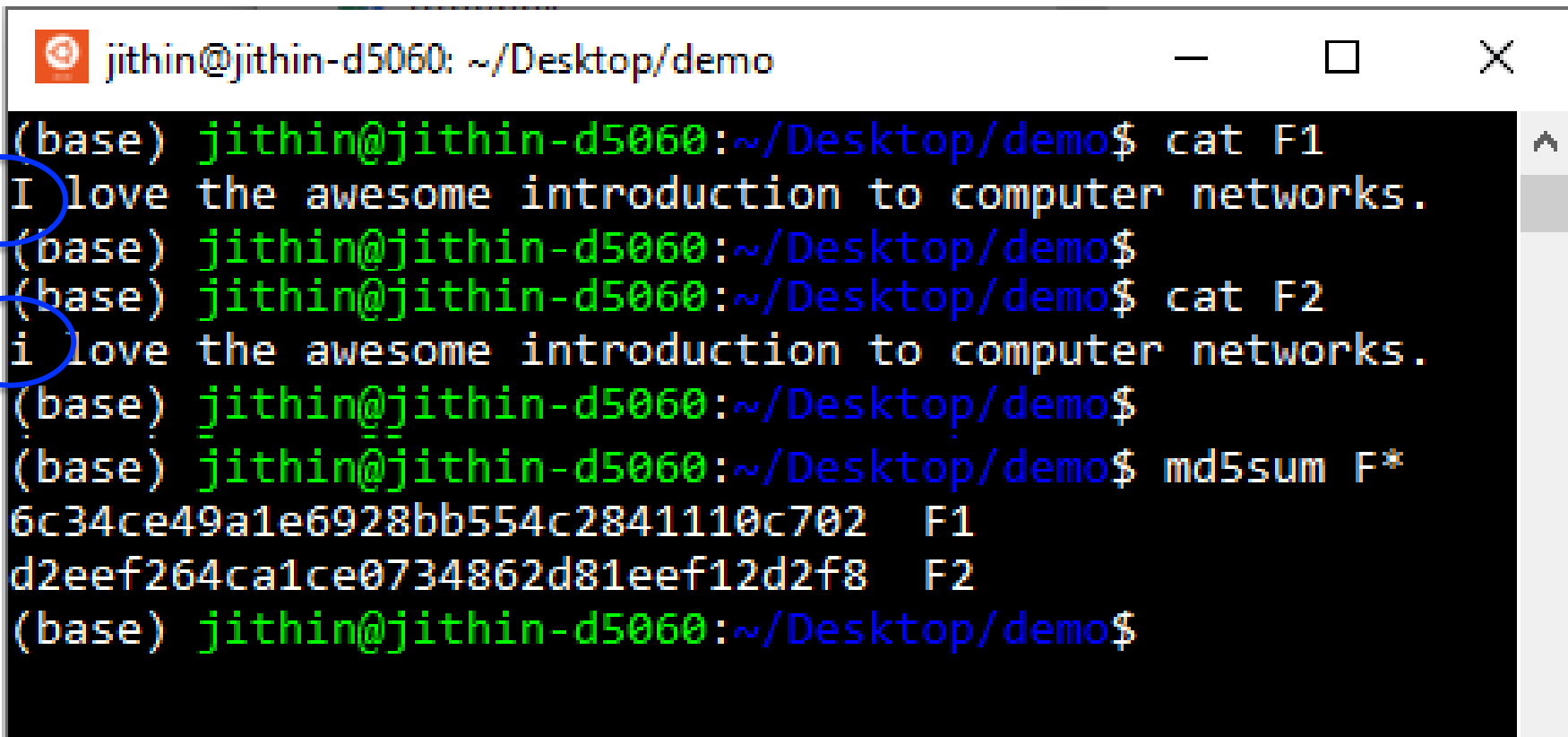
- Generate short, *fixed-length* outputs (or *digests*); *128 bits*
 - especially useful for longer inputs; “*fingerprint*”

A terminal window titled 'jithin@jithin-d5060: ~/Desktop/demo' with standard window controls. The terminal shows the execution of 'ls -lh' and 'md5sum *' commands. The 'ls' command lists four files: 'empty', 'large.mp4', 'small.zip', and 'tiny.pdf'. The 'md5sum' command outputs the corresponding MD5 hashes for each file.

```
jithin@jithin-d5060: ~/Desktop/demo
(base) jithin@jithin-d5060:~/Desktop/demo$ ls -lh
total 1.1G
-rwxrwxrwx 1 jithin jithin  0 Oct 18 15:41 empty
-rwxrwxrwx 1 jithin jithin 1.1G Oct  3 21:05 large.mp4
-rwxrwxrwx 1 jithin jithin 60M Jul 26  2021 small.zip
-rwxrwxrwx 1 jithin jithin 581K Dec 20  2019 tiny.pdf
(base) jithin@jithin-d5060:~/Desktop/demo$ md5sum *
d41d8cd98f00b204e9800998ecf8427e  empty
9c2baf4ad9658da3d37f4cb191cde902  large.mp4
090f09aff66ad1c5041223a5448cc49e  small.zip
820556736ebbd7ea8ede2fb9582aeec9  tiny.pdf
(base) jithin@jithin-d5060:~/Desktop/demo$
```


Hash function, e.g., md5sum

- A small change in the input should result in a large change in the hash output



```
jithin@jithin-d5060: ~/Desktop/demo

(base) jithin@jithin-d5060:~/Desktop/demo$ cat F1
I love the awesome introduction to computer networks.
(base) jithin@jithin-d5060:~/Desktop/demo$ 
(base) jithin@jithin-d5060:~/Desktop/demo$ cat F2
i love the awesome introduction to computer networks.
(base) jithin@jithin-d5060:~/Desktop/demo$ 
(base) jithin@jithin-d5060:~/Desktop/demo$ md5sum F*
6c34ce49a1e6928bb554c2841110c702  F1
d2eef264ca1ce0734862d81eef12d2f8  F2
(base) jithin@jithin-d5060:~/Desktop/demo$
```

Message Integrity

❖ *To ensure Message integrity:*

❖ Send $(m, H(m))$

❖ *Does this work?*

❖ *No!!!*

❖ *Recall: Message integrity:*

❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ What happens if the attacker replaces $(m, H(m))$ with $(m', H(m'))$?

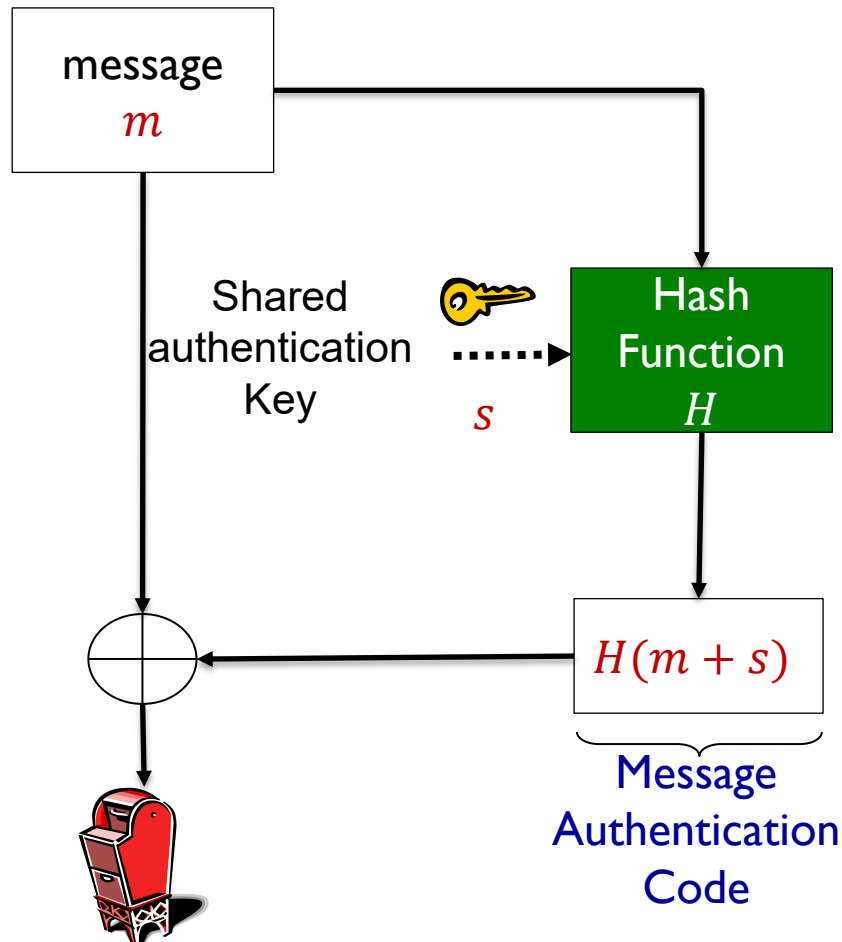
❖ Receiver has no way of detecting it

Message Authentication Code

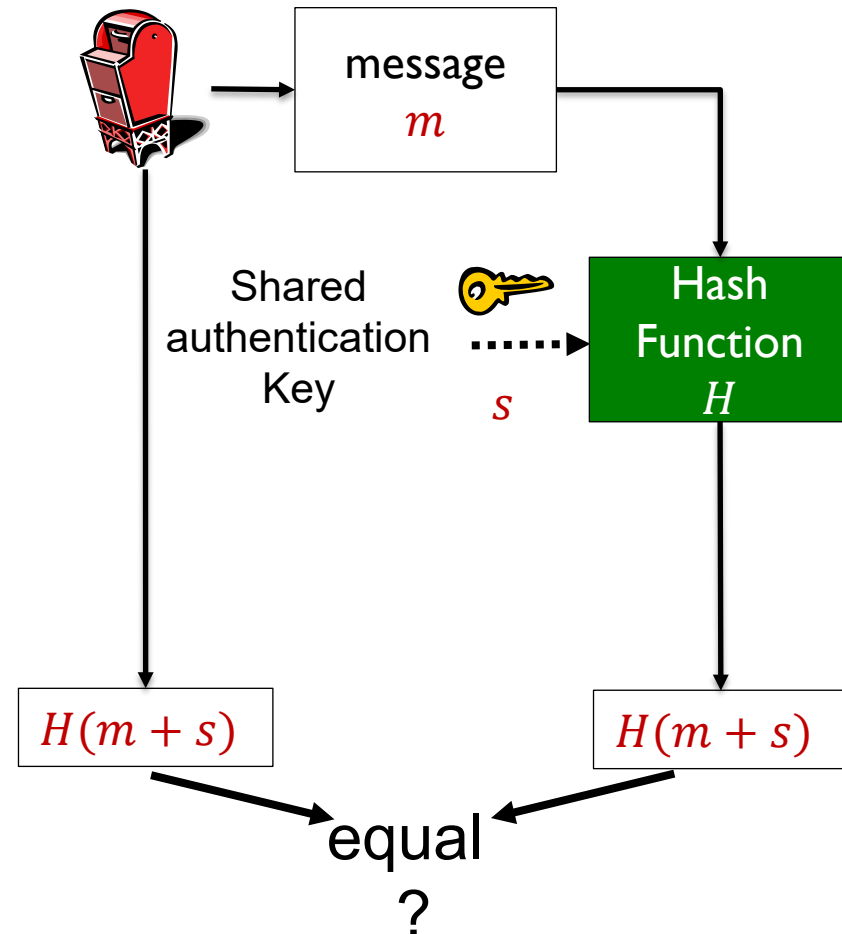
- ❖ *The sender and receiver share a “Authentication key” s*
- ❖ *To ensure Message integrity:*
 - ❖ *Send $(m, H(\underbrace{m + s}_{\text{Message Authentication Code}}))$*
- ❖ *Does This work?*
 - ❖ *Yes!!!*
 - ❖ *s is a secret key known to the receiver and no one else*
 - ❖ *Receiver can generate the authentication code directly from m and compare with the received code*

Message Authentication Code

Bob sends message:



Alice verifies the message:



What is network security?

❖ *Confidentiality:*

- ❖ only sender, intended receiver should “*understand*” message contents

❖ *Authentication:*

- ❖ sender, receiver want to *confirm identity* of each other

❖ *Message integrity:*

- ❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ *Access and availability:*

- ❖ services must be accessible and available to users

Digital signatures

- Cryptographic technique analogous to **hand-written signatures**
- Sender (Bob) digitally **signs** document, establishing he is document owner/creator.

Signature must be

- **Verifiable:**
 - Recipient (Alice) can check if the signature and the message was generated by Bob.
- **Unforgeable:**
 - No one, other than Bob should be able to generate the signature and the message.

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

use **public key** first,
followed by
private key

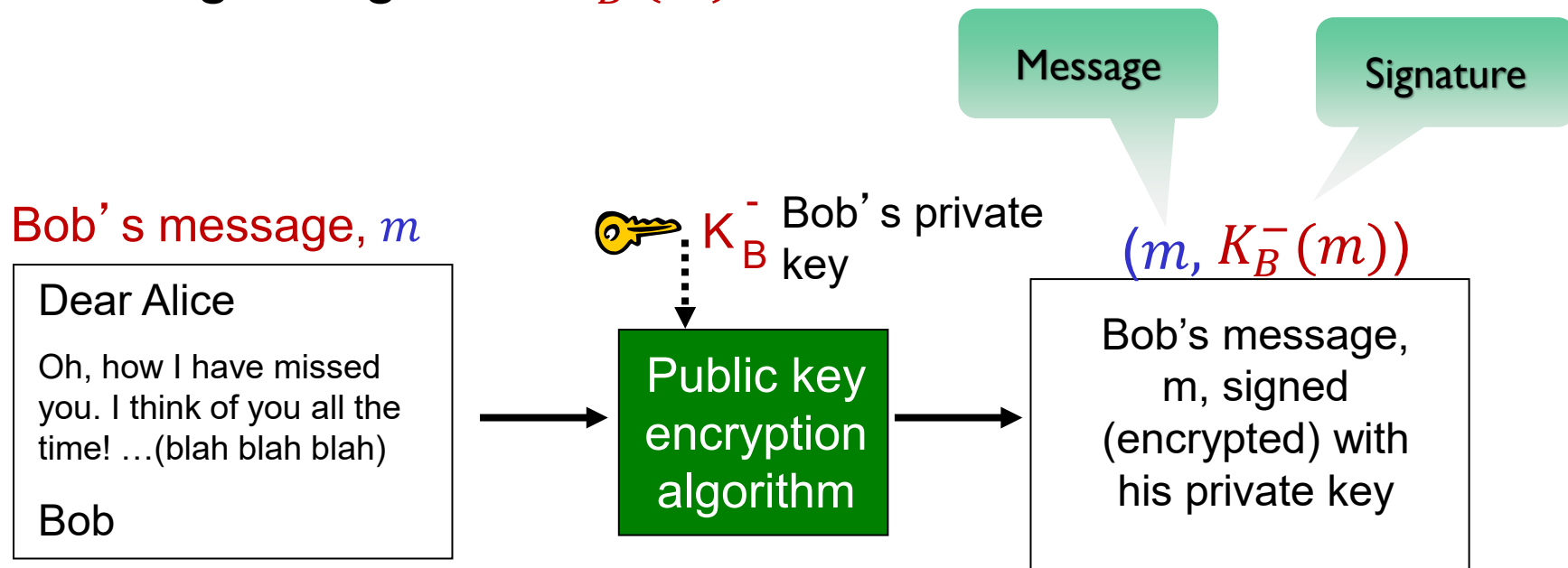
use **private key** first,
followed by
public key

result is the same!

Digital signatures

simple digital signature for message m :

- Bob signs m by encrypting with his private key K_B^- creating the signature $K_B^-(m)$



Aim:

- Verifiable
- Unforgeable

Digital signatures

- Suppose Alice receives msg m , with signature: $m, K_B^-(m)$.
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

- ✓ Alice can take m , and signature $K_B^-(m)$ to court and prove that Bob signed m

Aim:

- Verifiable
- Unforgeable

Digital signatures: Optimization

computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital “*fingerprint*”

- Apply hash function H to m , get fixed size message digest, $H(m)$.

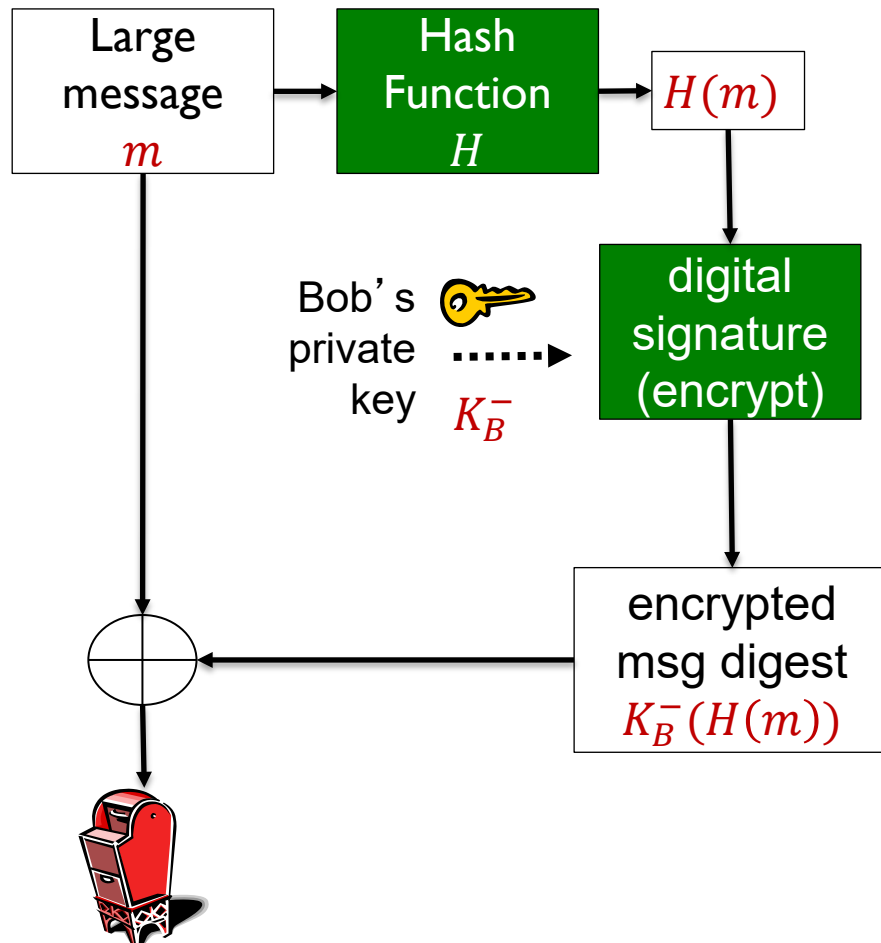


Hash function properties:

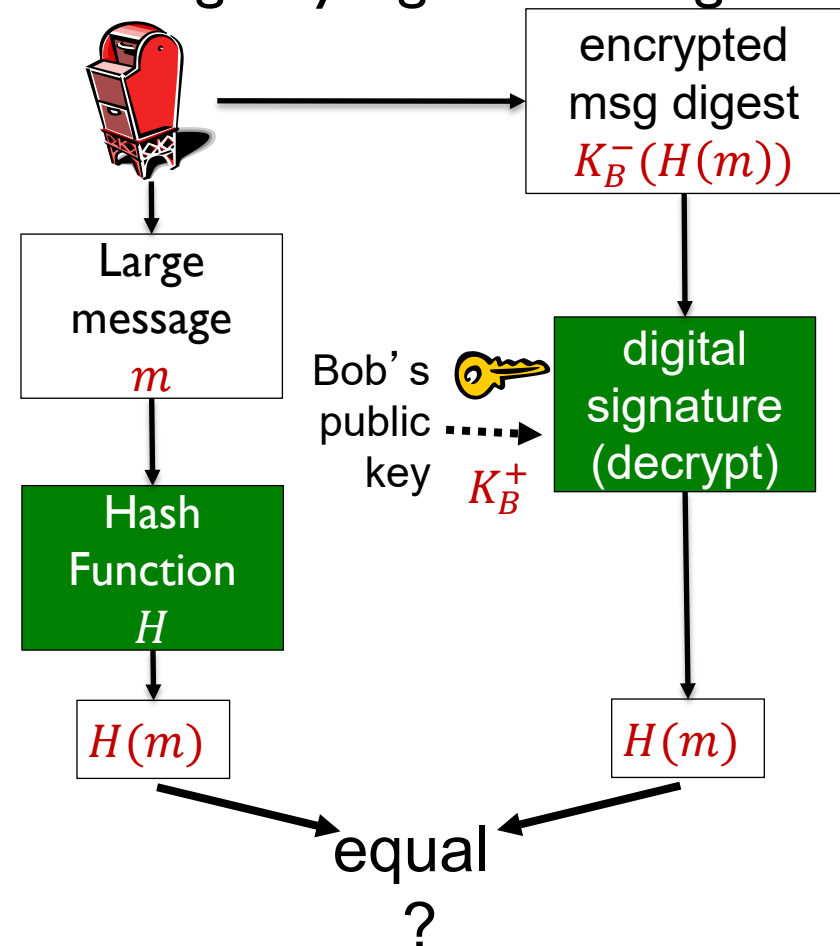
- produces fixed-size msg digest (fingerprint)
- given message digest x , computationally infeasible to find m such that $x = H(m)$

Digital signature = signed message digest

Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Public-key certification

- Trudy plays pizza prank on Bob
 - Trudy creates e-mail order:
*Dear PizzaHut, Please deliver to me four pepperoni pizzas.
Thank you, Bob*
 - Trudy **signs** order with her **private** key
 - Trudy sends order to Pizza Store
 - Trudy sends to Pizza Store **her public** key, but says it's Bob's public key
 - Pizza Store **verifies** signature; then delivers four pepperoni pizzas to Bob
 - Bob doesn't even like pepperoni
- Why did the prank succeed?



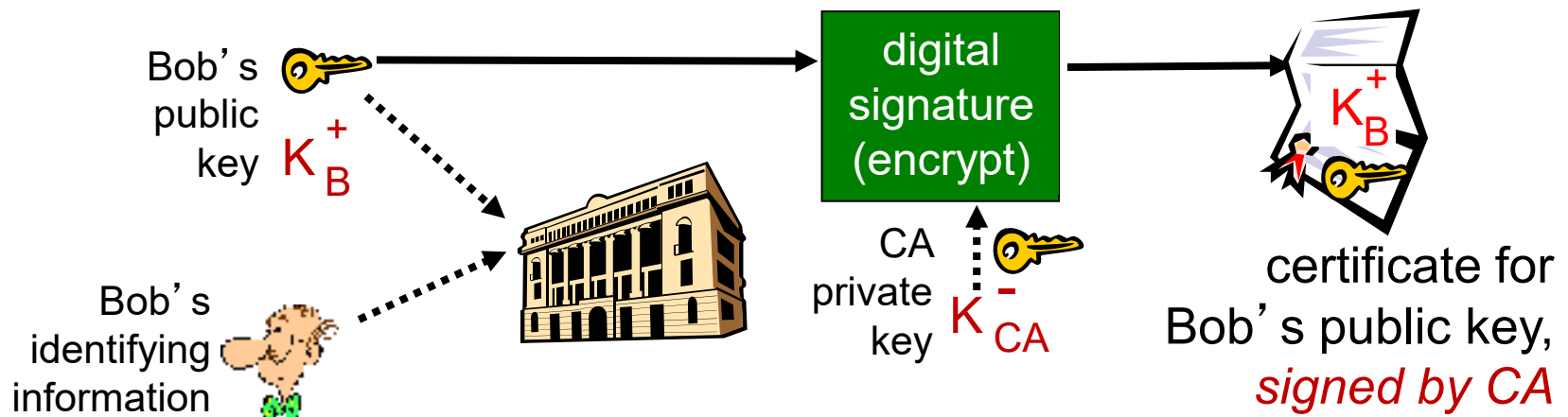
Point of Failure

Certification authorities

- **Problem:** *The reason we failed was, we did not know Bob's public key*
- **Solution:**
 - We create a Certification authority(CA) who maintain a public database of everyone's public key
 - Anyone who receives a message from "Bob" will access this database for K_B^+
- **Problem:** What if Trudy intercepts the communication with the CA and alters it?
- **Solution:**
 - CA signs it's messages.
- **Problem:** We do not know CA's public key
- **Solution:**
 - Let us make this a universal knowledge!!!!
 - We maintain a list of CAs trusted a priori.
 - Operating system has a list of "Trusted Root Certification Authorities"

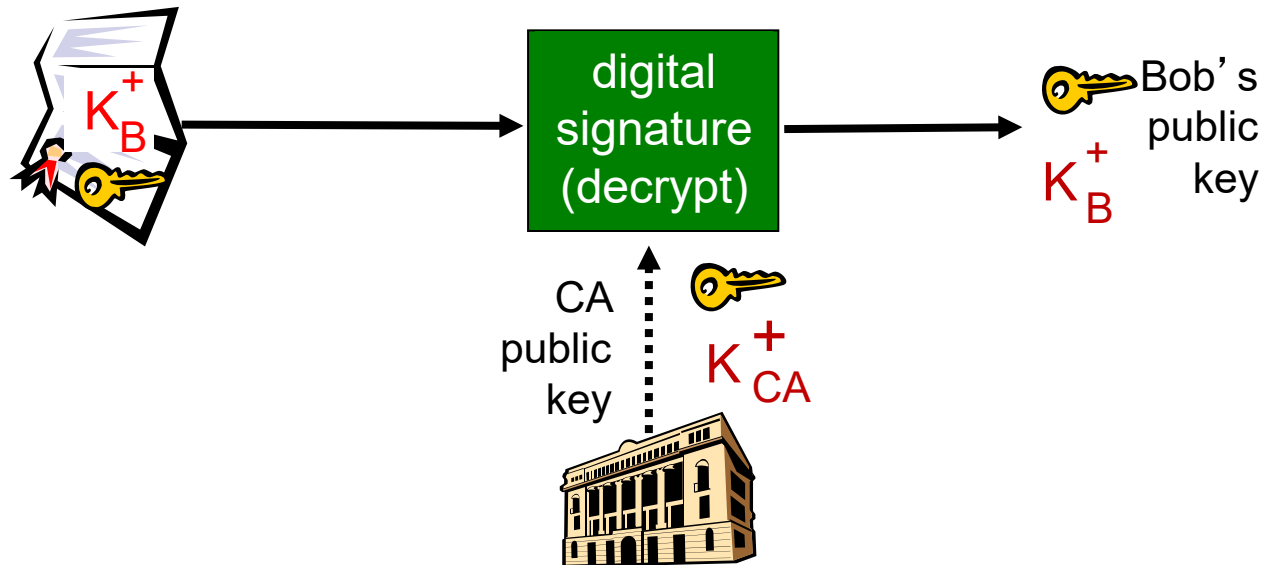
Certification authorities

- *certification authority (CA)*: binds public key to particular entity, *E*.
- *E* (person, router) registers its public key with CA.
 - *E* provides “proof of identity” to CA.
 - CA creates certificate binding *E* to its public key.
 - certificate containing *E*'s public key digitally signed by CA
 - CA says “this is *E*'s public key”



Certification authorities

- when Alice wants Bob's public key:
 - gets Bob's certificate (Bob or elsewhere).
 - apply CA's public key to Bob's certificate, get Bob's public key



What is network security?

❖ *Confidentiality:*

- ❖ only sender, intended receiver should “*understand*” message contents

❖ *Authentication:*

- ❖ sender, receiver want to confirm identity of each other

❖ *Message integrity:*

- ❖ sender, receiver want to ensure message *not altered* (in transit, or afterwards) *without detection*

❖ *Access and availability:*

- ❖ services must be accessible and available to users

Chapter 8 roadmap

8.1 What is network security?

8.2 Principles of cryptography

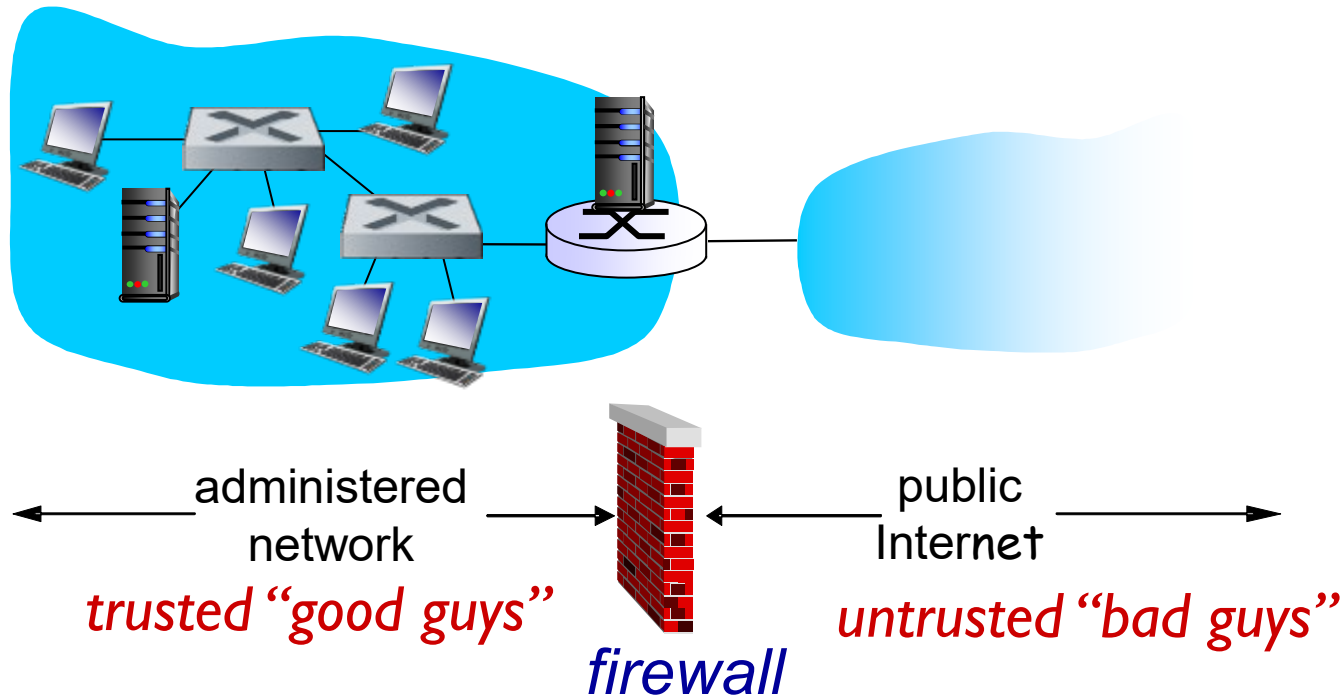
8.3 Message integrity and digital signatures

8.9 Operational security: firewalls

Firewalls

firewall

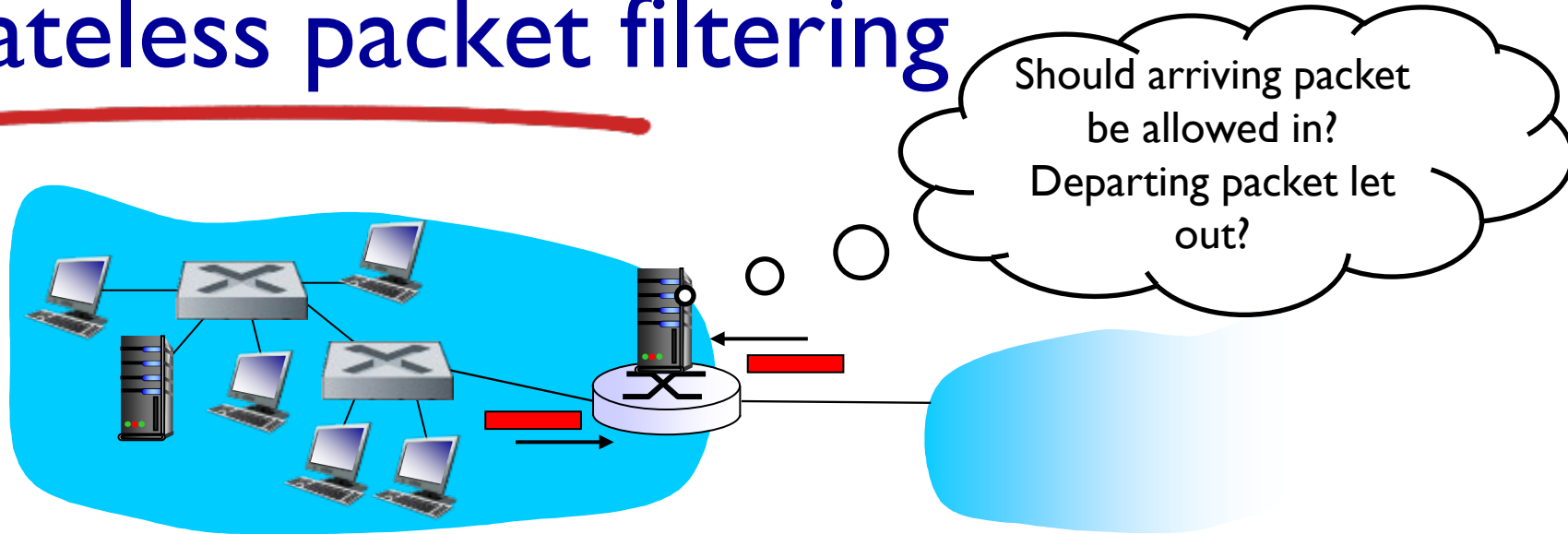
isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



Firewalls: why

- ❖ Prevent denial of service (DoS) attacks:
 - ❖ SYN flooding: attacker establishes many bogus TCP connections, no resources left for “real” connections
- ❖ Prevent illegal modification/access of internal data
 - ❖ e.g., attacker replaces CIA’s homepage with something else
- ❖ Allow only authorized access to inside network
 - ❖ set of authenticated users/hosts
- ❖ Three types of firewalls:
 - ❖ stateless packet filters
 - ❖ stateful packet filters
 - ❖ application gateways

Stateless packet filtering



- internal network connected to Internet via *router firewall*
- router *filters packet-by-packet*, decision to forward/drop packet based on:
 - source IP address, destination IP address
 - TCP/UDP source and destination port numbers
 - ICMP message type
 - TCP SYN and ACK bits

Stateless packet filtering: example

- *example 1:* block incoming and outgoing datagrams with IP protocol field = 17
 - *result:* all incoming, outgoing UDP flows are blocked

- *example 2:* block inbound TCP segments with ACK=0.
 - *result:* prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Stateless packet filtering: more examples

<i>Policy</i>	<i>Firewall Setting</i>
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

ACL: table of rules, applied top to bottom to incoming packets:
(action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Limitations of firewalls

- *IP spoofing*: router can't know if data “really” comes from claimed source
- Can become a bottleneck
- *tradeoff*: degree of communication with outside world, level of security
- Many highly protected sites still suffer from attacks

Summary

❖ *Confidentiality:*

- ❖ Substitution Cipher
- ❖ Symmetric Key
- ❖ Public Key

❖ *Authentication:*

- ❖ Signature

❖ *Message integrity:*

- ❖ Cryptographic Hash

❖ *Access and availability:*

- ❖ Firewall