

4CS401

Cryptography and Network Security

By,

Manik K. Chavan

Assistant Professor, CSE Department

Walchand College Of Engineering, Sangli.

manik.chavan@walchandsangli.ac.in

Mob. 7507760777/9545413443

Prerequisite

Computer Networks

Mathematical background

Syllabus

Module-I: INTRODUCTION

Module-II: SYMMETRIC KEY CRYPTOGRAPHY

Module-III: PUBLIC KEY CRYPTOGRAPHY

Module-IV: MESSAGE AUTHENTICATION AND INTEGRITY

Module-V: NETWORK SECURITY

Module-VI: SYSTEM SECURITY

Course Learning Outcomes



CO1: apply different encryption and decryption techniques to solve problems related to confidentiality and authentication.



CO2: Analyze and apply system security concept to recognize malicious code.



CO3: analyze different attacks on networks and evaluate the performance of firewalls and security protocols like SSL, IPSec, and PGP



CO4: Apply the knowledge of cryptographic checksums and evaluate the performance of different message digest algorithms for verifying the integrity of varying message sizes.

Agenda



Why Cryptography and Network Security?



What is Cryptography



Classification of cryptography

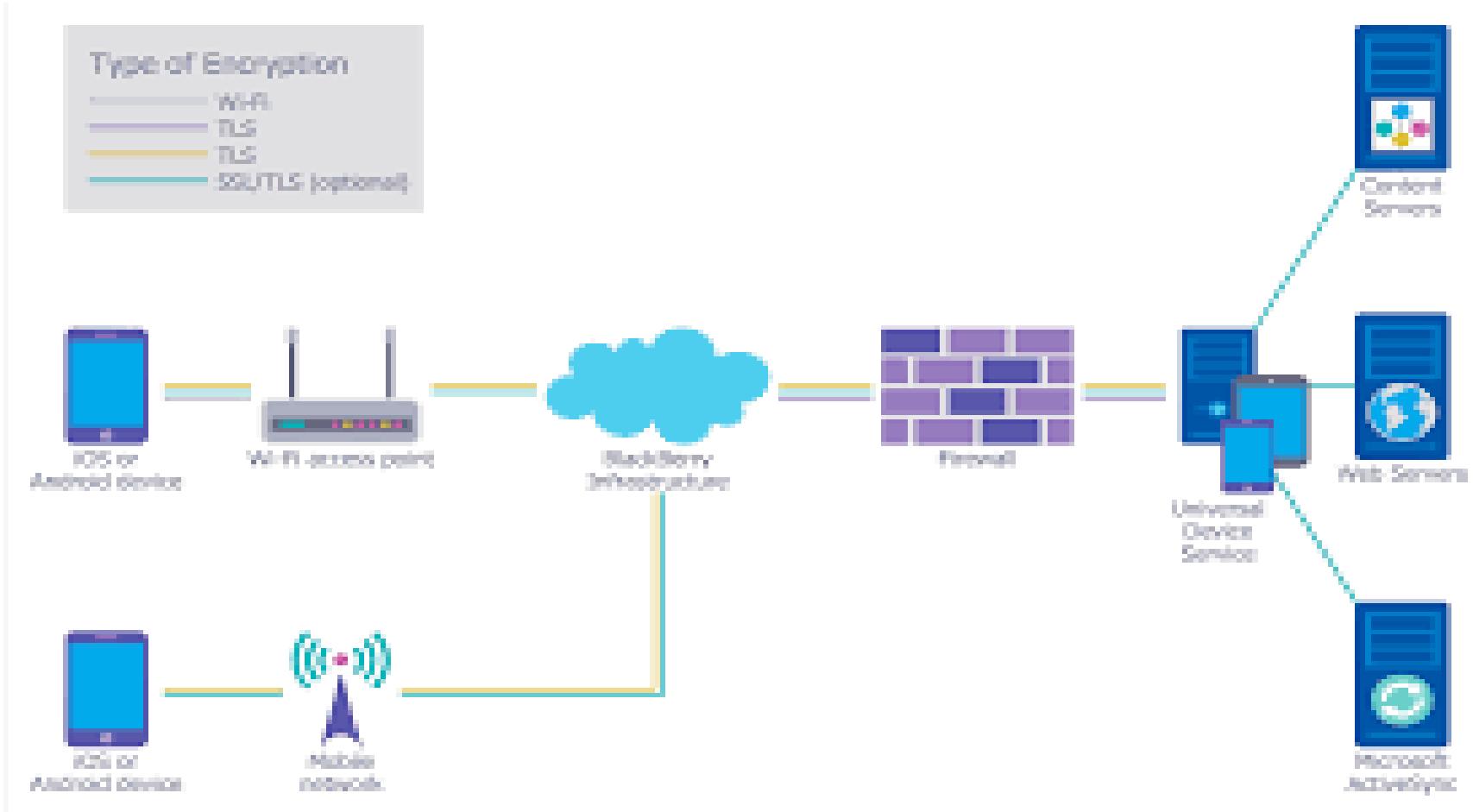


How various cryptographic algorithm works

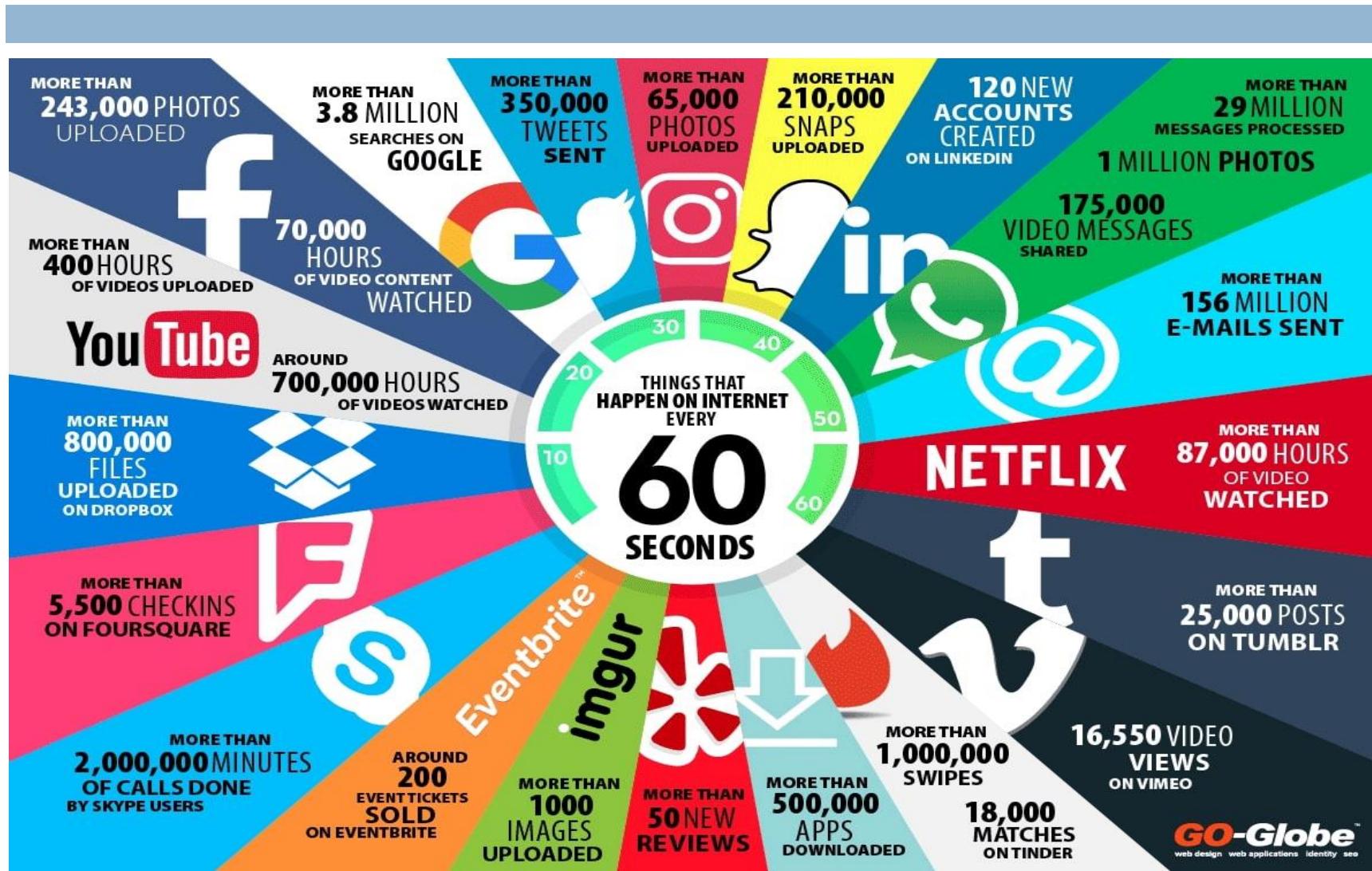
Connected world



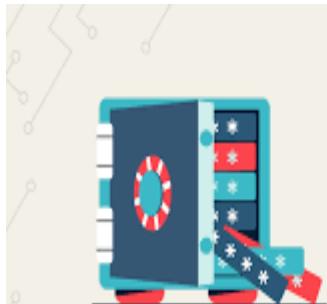
Need for Network Security



Internet statistics-2020!



Increased Security Breaches



On March 21, 2019, **FACEBOOK** admitted that since 2012 it has not properly secured the passwords of as many as **600 MILLION USERS.**

IdentityForce

VARONIS



There was an **80%** **INCREASE** in the number of people affected by **HEALTH DATA BREACHES** from 2017 to 2019.

Statista



VARONIS



YAHOO holds the record for the largest data breach of all time with **3 BILLION COMPROMISED ACCOUNTS.**

Statista

VARONIS



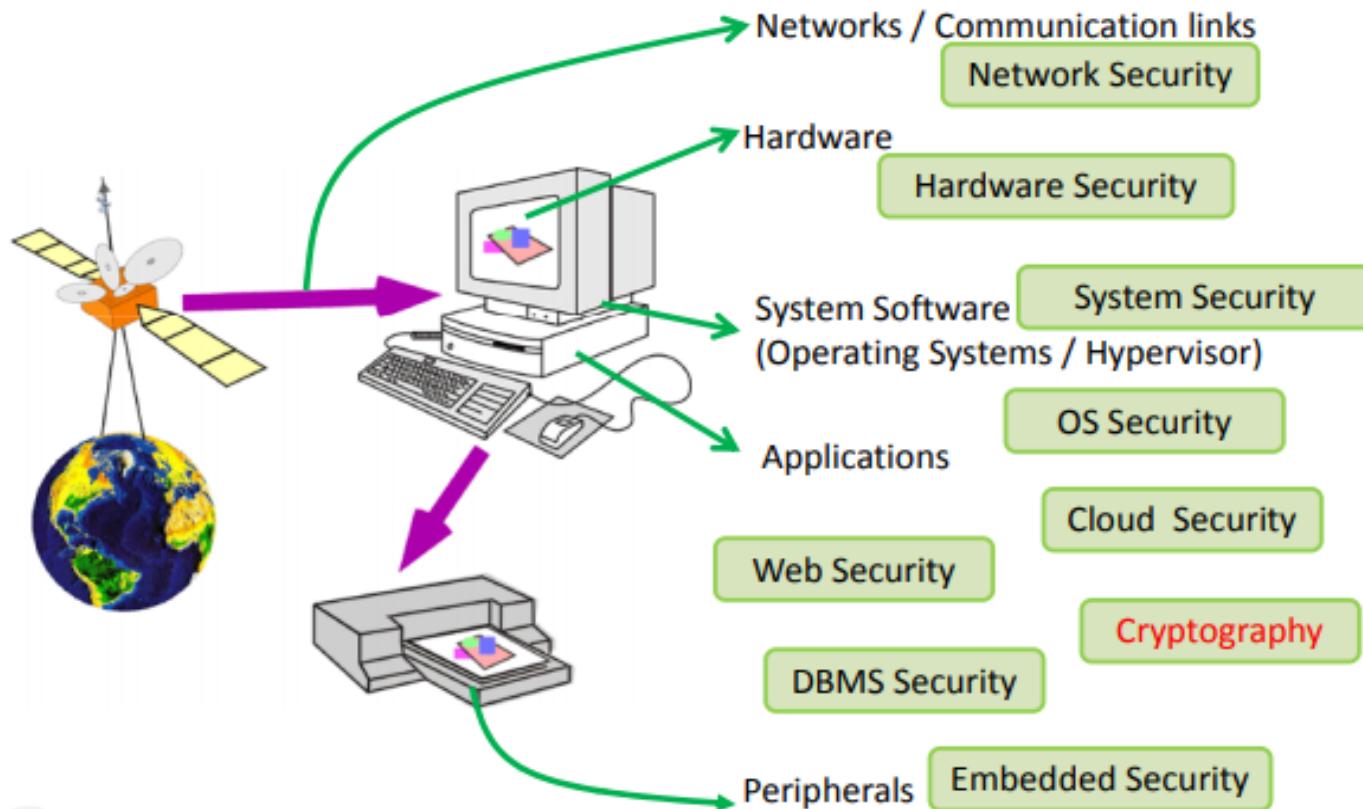
It is predicted that **GLOBAL CYBERSECURITY** spending will exceed **\$1 TRILLION** cumulatively from 2017 to 2021.

Cybersecurity Ventures



VARONIS

Security Studies(an ocean)



Information Security vs. Cyber Security vs. Network Security

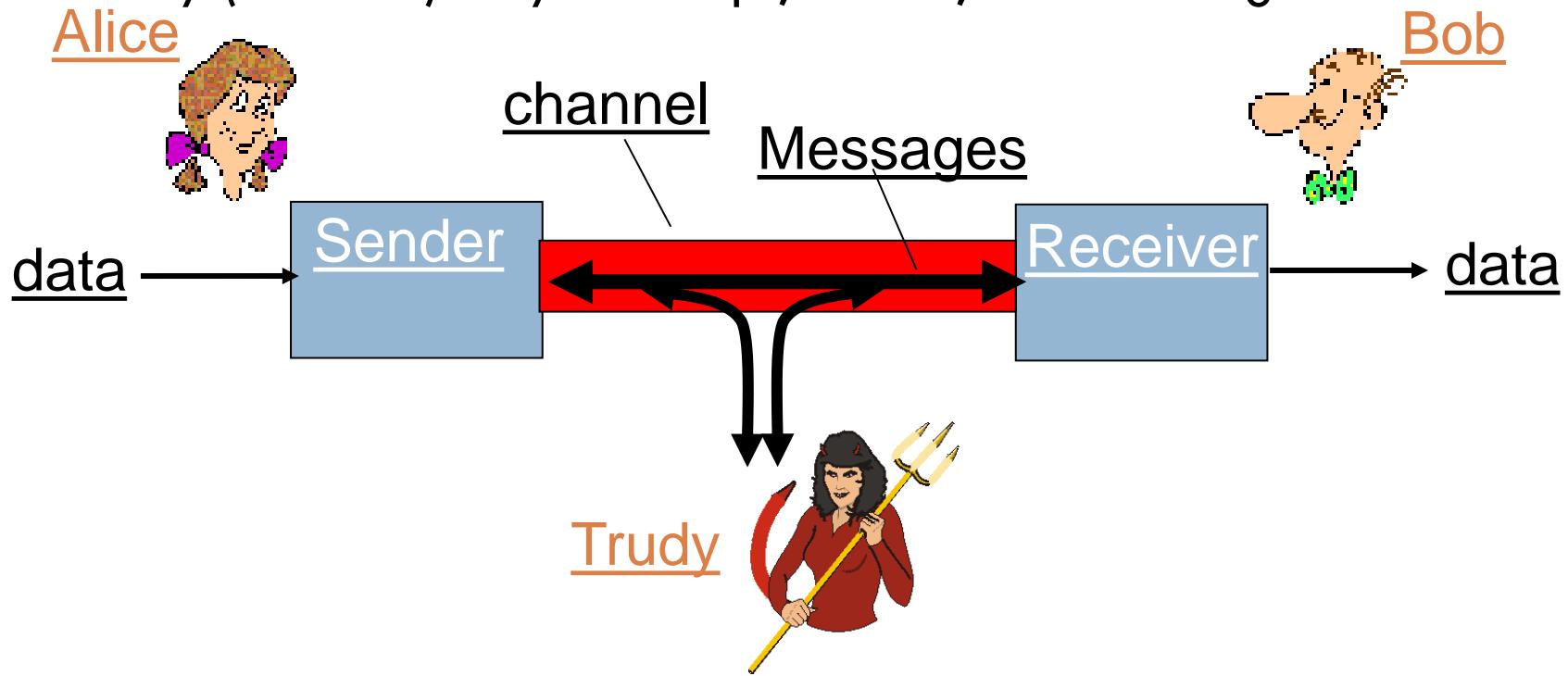
- What is Information Security?
 - Information security (also known as InfoSec) ensures that both physical and digital data is protected from unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction.
 - Information security differs from cybersecurity in that InfoSec aims to keep data in any form secure, whereas cybersecurity protects only digital data.

Information Security vs. Cyber Security vs. Network Security cont....

- What is Cybersecurity?
 - Cybersecurity, is a process or measures taken by organizations or experts to protect devices, computer networks, or data from malicious activities.
- What is Network Security?
 - a subset of cybersecurity, aims to protect any data that is being sent through devices in your network to ensure that the information is not changed or intercepted.
 - role of network security is to protect the organization's IT infrastructure from all types of cyber threats.

Communication over the Internet

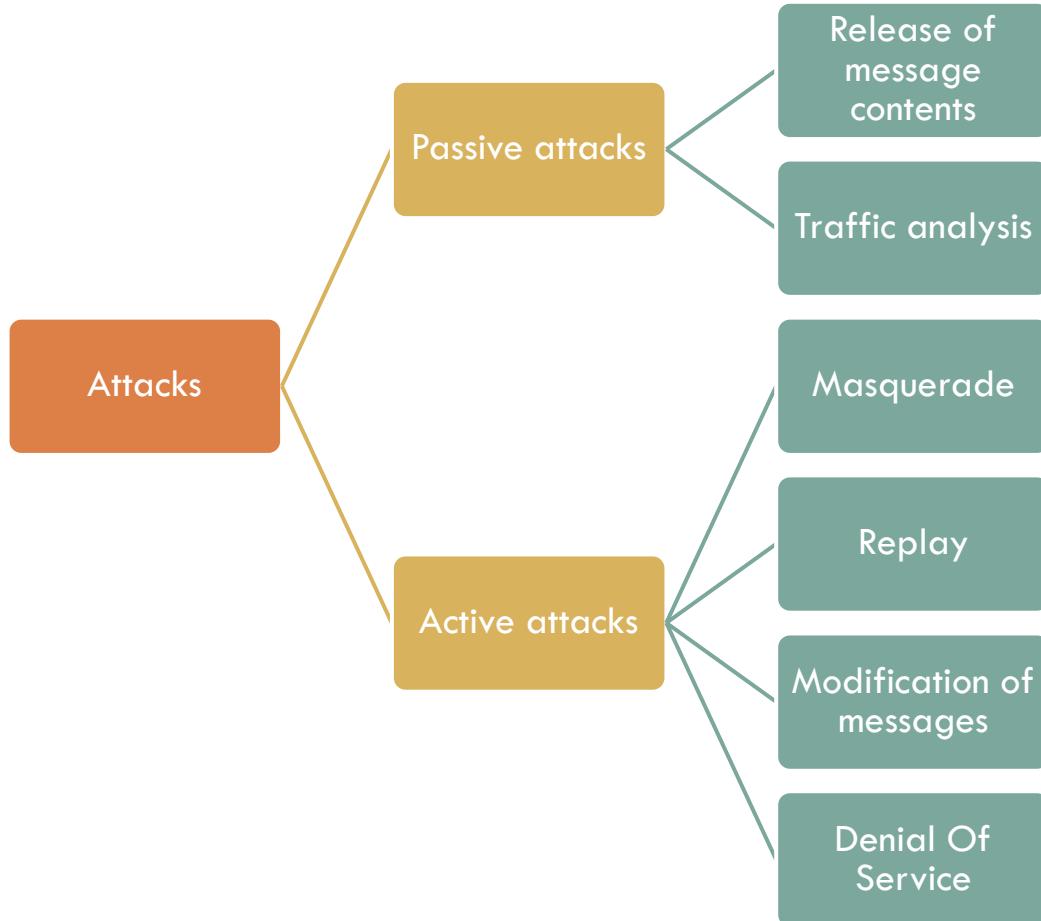
- Friends and enemies: Alice, Bob, Trudy
- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



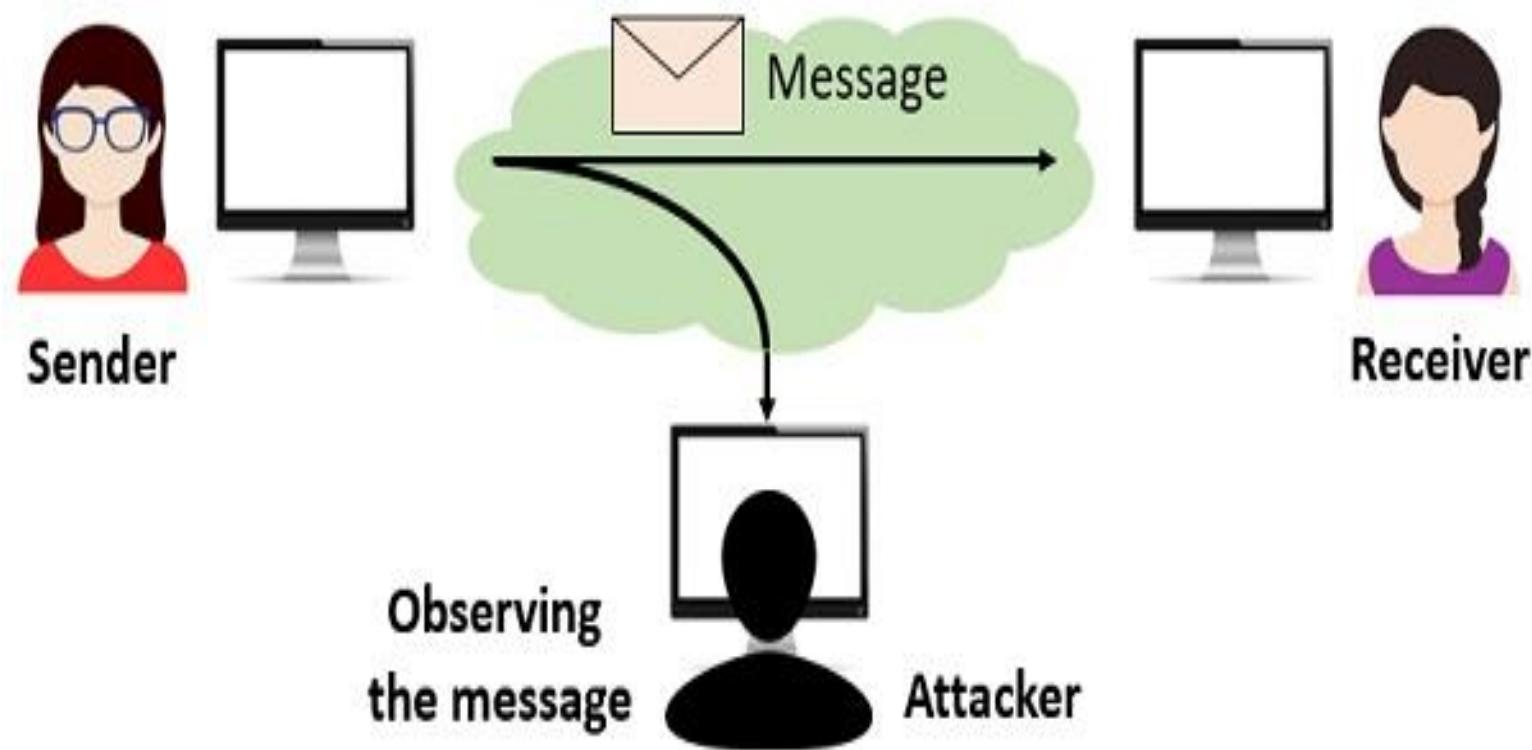
Security attacks



Action that compromises the security of an organization or an individual

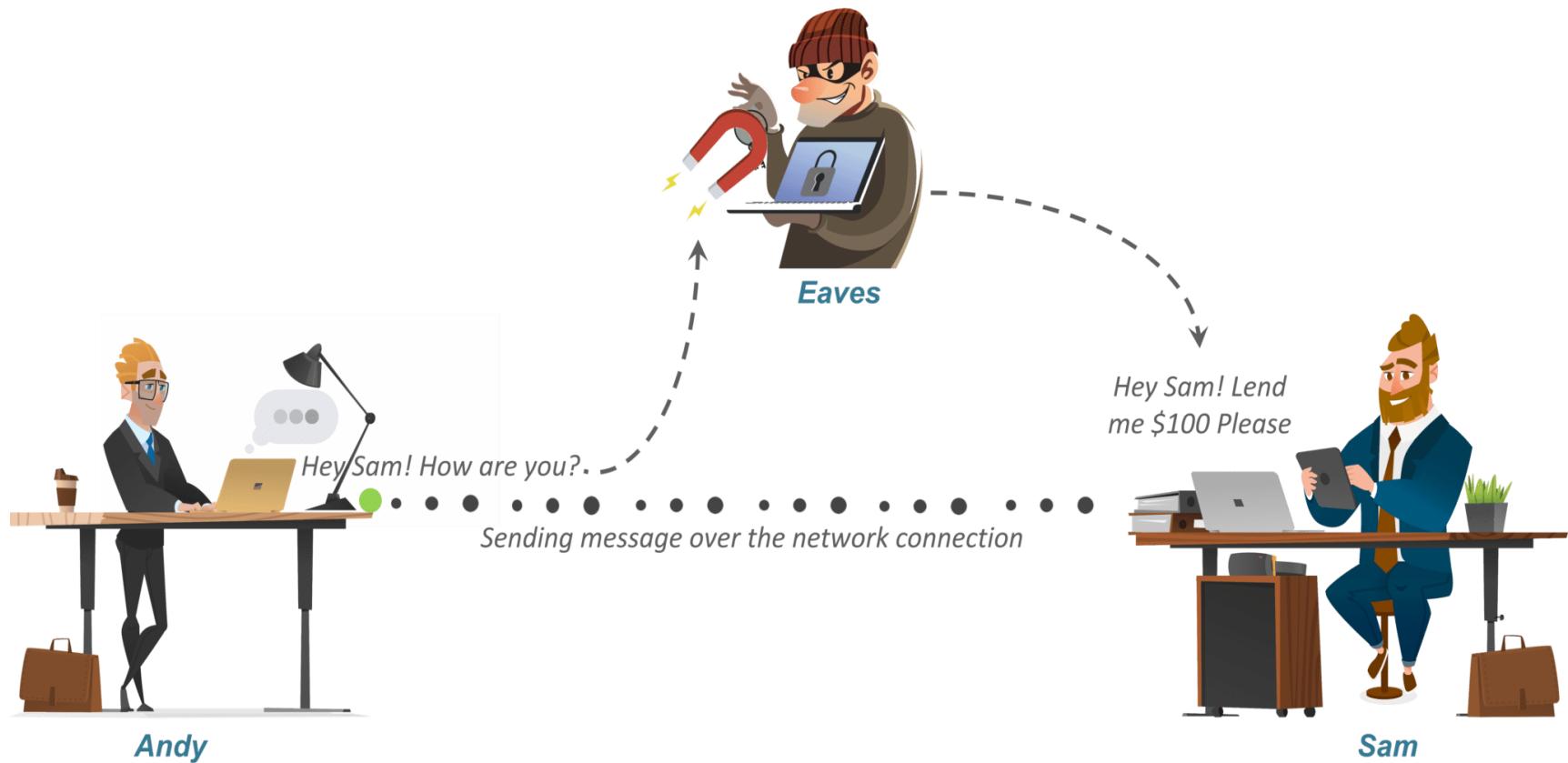


Passive Attacks



Passive Attack

Active attack



Security Attacks

A means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of **passive attacks** and **active attacks**

- A **passive attack** attempts to learn or make use of information from the system but does not affect system resources
- An **active attack** attempts to alter system resources or affect their operation

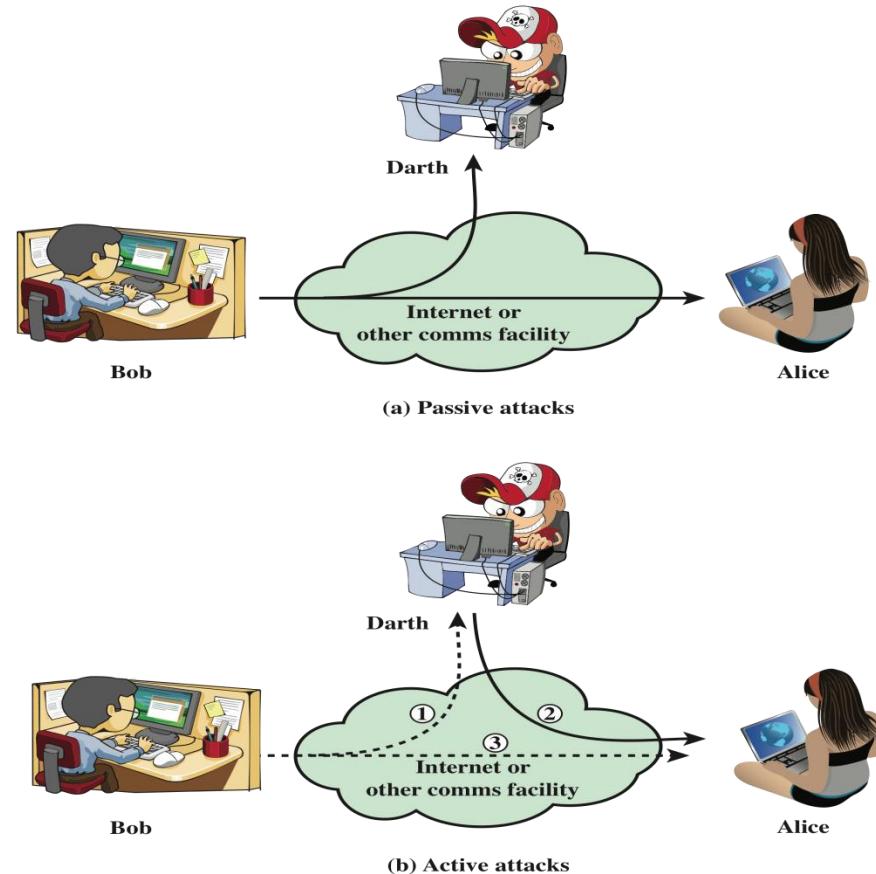


Figure 1.1 Security Attacks

Passive Attacks

- the nature of eavesdropping on, or monitoring of, transmissions
- Goal of the opponent is to obtain information that is being transmitted



- Two types of passive attacks are:
 - The release of message contents
 - Traffic analysis

Active Attacks

- ❑ Involve some modification of the data stream or the creation of a false stream
- ❑ Difficult to prevent because of the wide variety of potential physical, software, and network vulnerabilities
- ❑ Goal is to detect attacks and to recover from any disruption or delays caused by them



Masquerade

- Takes place when one entity pretends to be a different entity
- Usually includes one of the other forms of active attack

Replay

- Involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect

Modification of messages

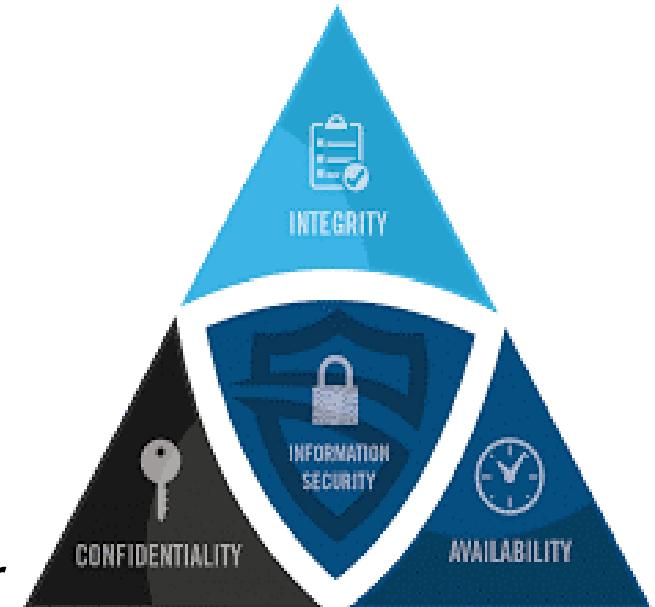
- Some portion of a legitimate message is altered, or messages are delayed or reordered to produce an unauthorized effect

Denial of service

- Prevents or inhibits the normal use or management of communications facilities

Security goals-CIA Triad

- Crucial component in all security system: **CIA triad**
- Cryptography used to achieve:
 - Confidentiality
Only authorized users access information
 - Integrity
Ensure completeness, accuracy and an absence of unauthorized modifications
 - Availability
Available and operational when required by users



need to define some other security objectives!

- **Authentication** - a mechanism (a protocol) by which a user is identified and uses some token to prove who they are.
- **Non-repudiation**- a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Cryptography

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it.

Thus preventing unauthorized access to information.

The prefix “**crypt**” means “**hidden**” and suffix “**graphy**” means “**writing**”.

Sample Encrypted Message

-----BEGIN PGP MESSAGE-----

Version: GnuPG v2.0.20 (MingW32)

hQEMA5EOTKIA1RLeAQgAk0+I4DzLmyygCxWs/f+R0XjVtJIFp5010gjhCZC5+h58
9vqKZF511d/+2vi/Jzt6vfSW2ORqPRfkeVcWCzz4FS6RchX6d9IkBHEnf6/US+o
IZnqTOx/QIcEvhVqpgEs0i01yjQ5GyPpUhPwiNchoWcEyjp6p6OjdXSvXpR8Kw1
6ssbTFIZOx7b0500VNH6dExhV9D86OqkGnhE7ap8IH5J8uzUJD1pPdNiRQRTu+Qv
vb30kBQ34egGY5avJKBk88ybtXEbfawKREbGtZaC1kAOXNPjfAEmar/ENx6ceKUF
UzEbJ17j520JFcHGEGdpQufc8IrPAzfW2XnxzZOMdLpAVzyKMr3+SENCsere+vN
K48dKwosb0gIWFPVtWZh7swEtTRiMnyP7NkHB3PlQ3gtx7N04a5yPQJq0JBUoq0E
S1Dv5K2q2gSL+HiCj311DIltMHkbNGtJDP+/4ETgSciId91AKvr6FK9mzLYrp09gz
+Y8g6Lgz+Ib6YWhQuwyG4ObqkIywZeBvtQ5yWLk9HdrOiqpBFhzLcKfs60NzWUND
cZIELVn7cqsS1IYBw0CtqAb80vrX6zxIS6MTjNzIwQGwcbH0uaA3ctgGbnF7/E0n
sx8jBCA/8+nACuR3ZEmDqrhCZRvHEUWgo7tBa4Hi8oJ3JaxiO3xMJmulSN2PDyg/
dj+AG6hvJidNBdvBQmFOCdcDTAaBSMPxHeZQeEKoHXG617QQr9ZsHuN+1+tRPTZY
1dXWZwcZ9Ei55+vO2xwIjVpYfjQT11qofHbVOftn61LSVuLtTnLDP+pVW6tmalai
zGrqrBK9gm0A7XqIpIF7LqurDVODH0+NvYC75xQwHPOQ7An9P8JUvjmWUbPEZsBJ
1mwb3weQ9WorCYHX2SC16gTLFaKAvyRZyCkivdy2HZQJFnOuCtxN49Kwr37zcam2
Ic9+8IwQcEzwcmo+0W1VumPsTTglNWEXu5JQ1OZC2Oa+6laa5XxbmV0b059P250+
gKpfQUgVNUF0IicVYCEzH+cZjZ8+JtL+Wil07gsQAYa4w/eP/nRQXKVgA==
=ZZ4U

-----END PGP MESSAGE-----

German Telegram

JAN 19 1917

Fast Day Message	<input checked="" type="checkbox"/>	Day Letter		Night Message	
Please check the boxes above X opposite the class of service desired: OTHERWISE THIS TELEGRAM WILL BE TRANSMITTED AS A FAST DAY MESSAGE.					

WESTERN UNION
TELEGRAM

NEWCOMB CARLTON, PRESIDENT

H C
5500

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to:

GERMAN LEGATION
MEXICO CITY

via Galveston

JAN 19 1917

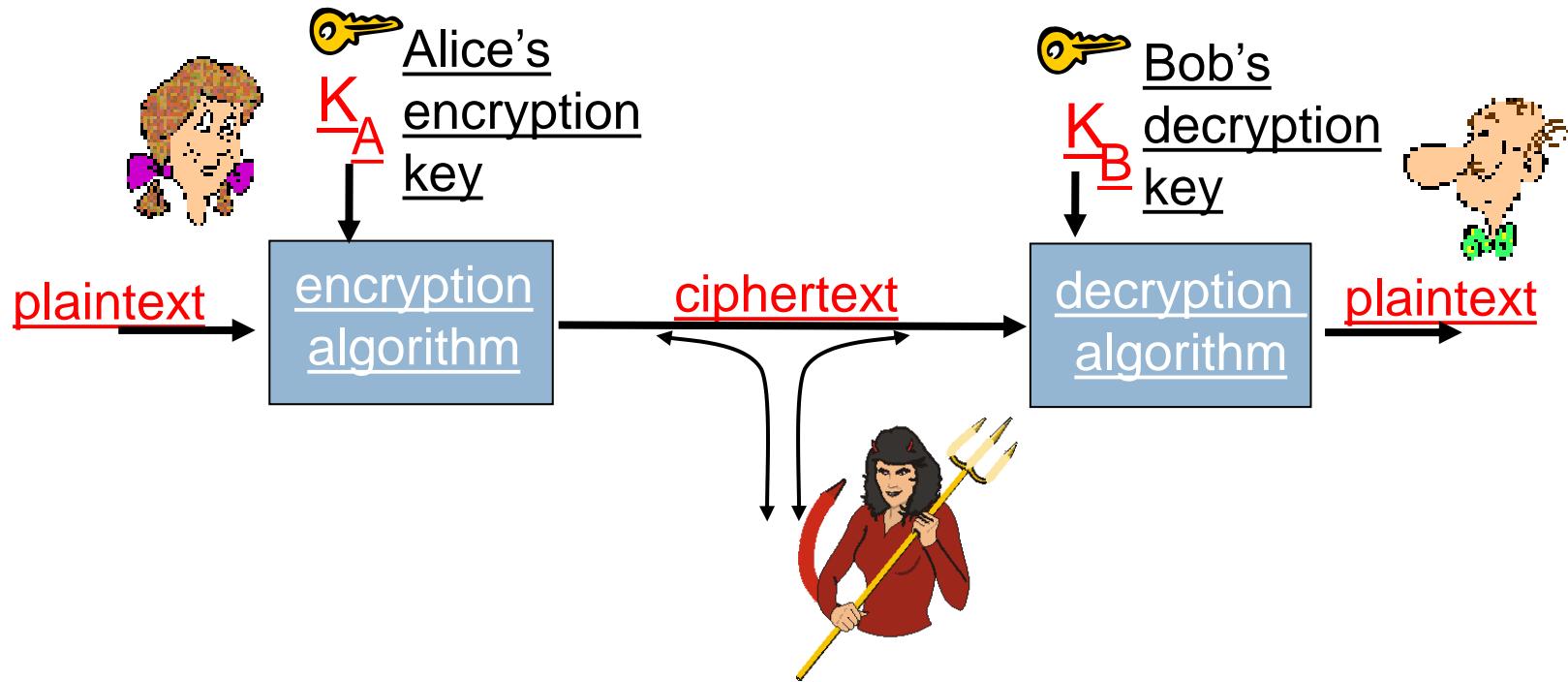
81/2117-798

130	13042	13401	8501	115	3528	416	17214	6491	11310
18147	18222	21560	10247	11518	23677	13605	3494	14936	
98092	5905	11311	10392	10371	0302	21290	5161	39695	
23571	17504	11269	18276	18101	0317	0228	17694	4473	
22284	22200	19452	21589	67893	5569	13918	8958	12137	
1333	4725	4458	5905	17166	13851	4458	17149	14471	6706
13850	12224	6929	14991	7382	15857	67893	14218	36477	
5870	17553	67893	5870	5454	16102	15217	22801	17138	
21001	17388	7416	23638	18222	6719	14331	15021	23845	
3156	23552	22096	21604	4797	9497	22464	20855	4377	
23610	18140	22260	5905	13347	20420	39689	13732	20667	
6929	5275	18507	52262	1340	22049	13339	11265	22295	
10439	14814	4178	6992	8784	7632	7357	6926	52262	11267
21100	21272	9346	9559	22464	15874	18502	18500	15857	
2188	5376	7381	98092	16127	13486	9350	9220	76036	14219
5144	2831	17920	11347	17142	11264	7667	7762	15099	9110
10482	97556	3569	3670						

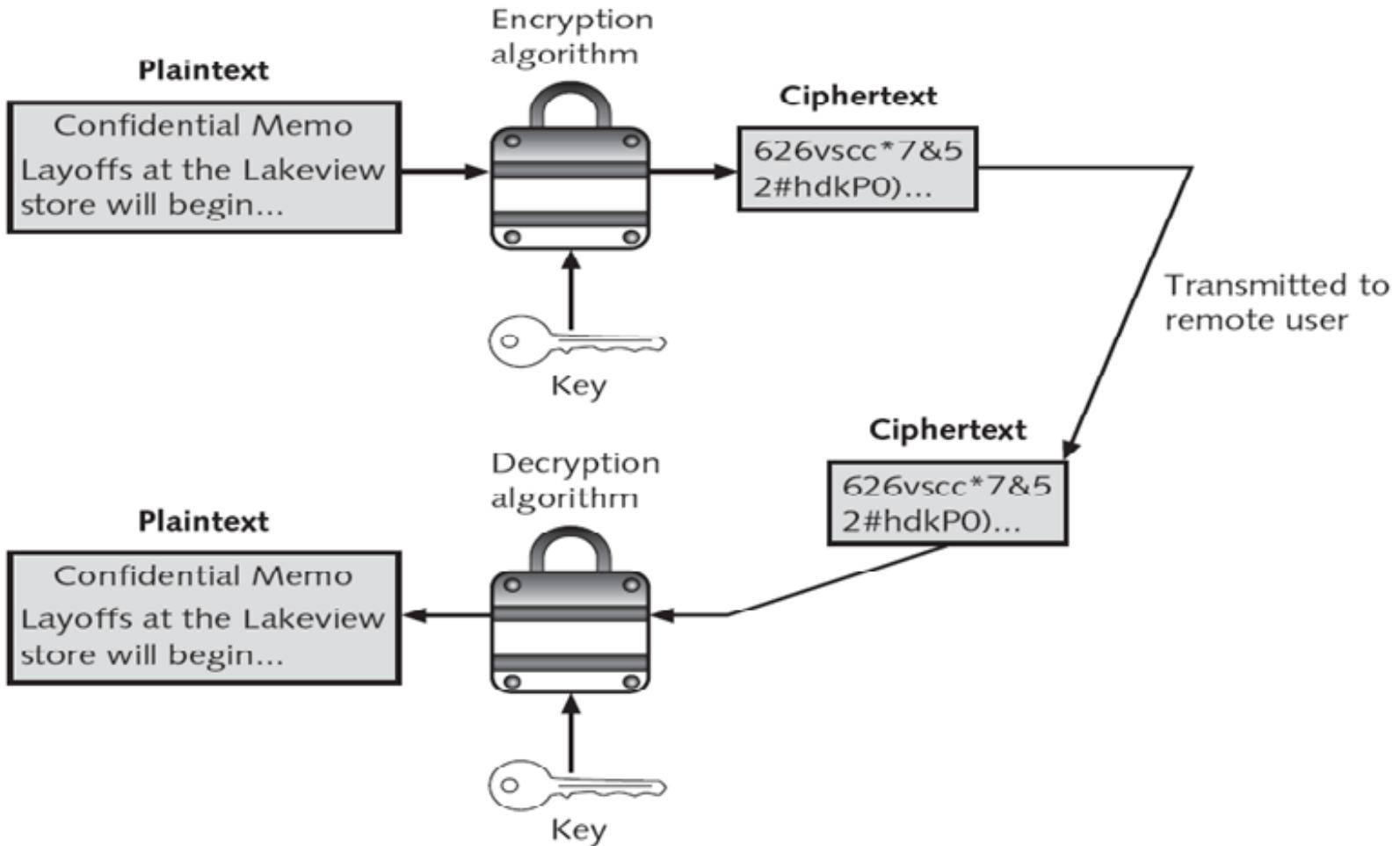
BEPNSTOPFF.

Charge German Embassy.

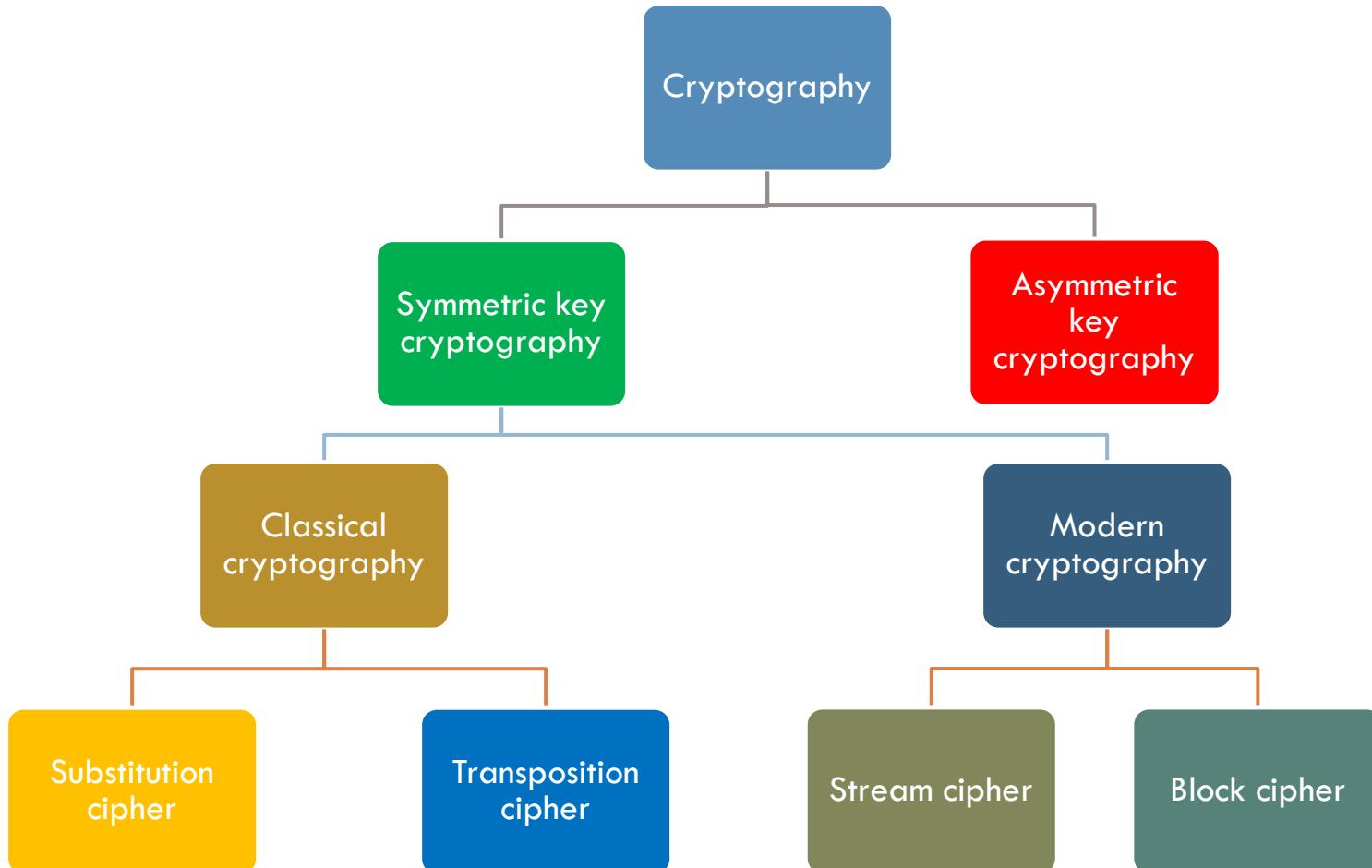
The language of cryptography



Cryptographic process



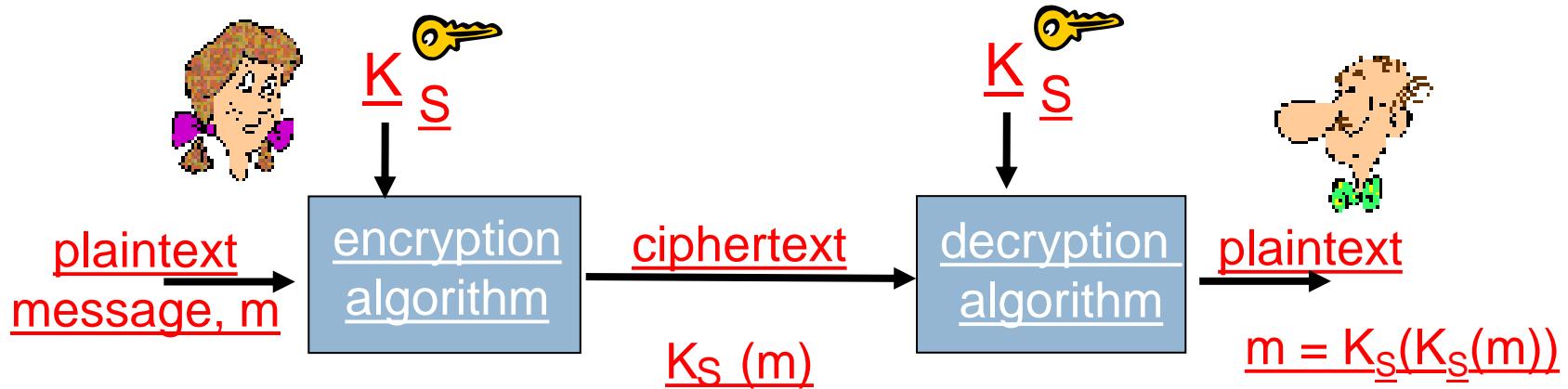
Classification of Cryptography



Types of Cryptography

- Crypto often uses keys:
 - Algorithm is known to everyone
 - Only “keys” are secret
- Symmetric key cryptography
 - Involves the use one key
- Asymmetric key/Public key cryptography
 - Involves the use of two keys

Symmetric key cryptography



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

The most common symmetric key system is the Data Encryption Standard (DES)

Substitution Cipher: Caesar Cipher

MESSAGE FROM MARY STUART KILL THE QUEEN

Substitution Table - Caesar's Cipher

ABCDEFGHIJKLMNOPQRSTUVWXYZ
↓ ↓ ↓ ↓ ↓
DEFGHIJKLMNOPQRSTUVWXYZABC



key = 3 cyclic shifts

PHVVD JHIUR PPDUB VVXDU WNLOO WKHTX HHQ

General Substitution Table

ABCDEFGHIJKLMNOPQRSTUVWXYZ
EYUOBMDXVTHIJPRCNAKQLSGZFW

Courtesy:
Andreas
Steffen

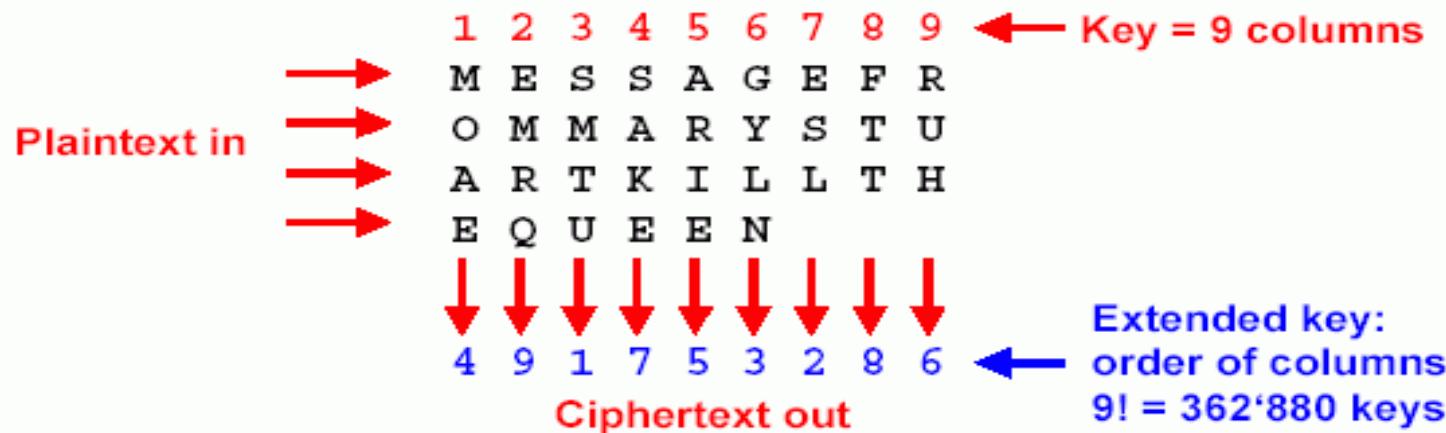
26! possible keys

JBKKE DBMAR JJEAJ KQLEA QHVII QXBNL BBP

Modern substitution ciphers take in N bits and substitute N bits using lookup table: called S-Box

Transposition cipher

MESSAGE FROM MARY STUART KILL THE QUEEN



MOAEE MRQSM TUSAK EARIE GYLNE SLFTT RUH
SMTUE SLGYL NMOAE ARIER UHSAK EFTTE MRQ

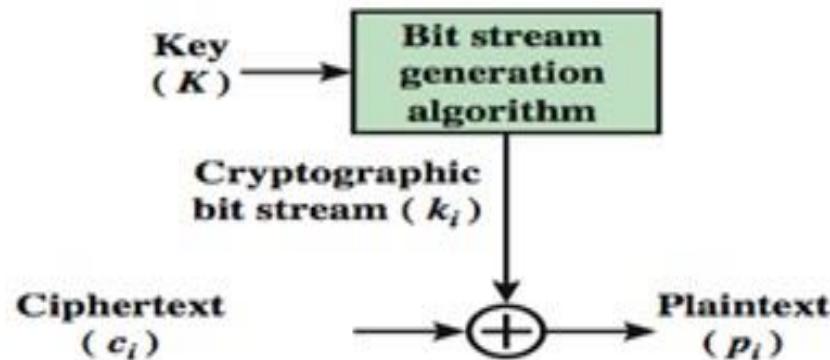
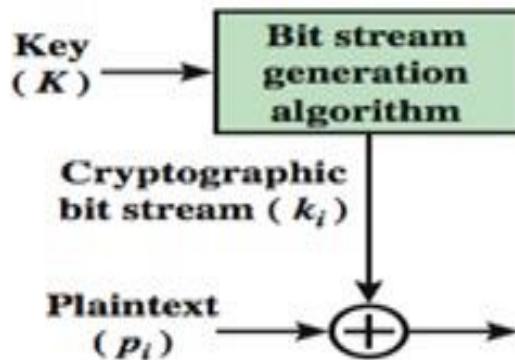
Courtesy:
Andreas
Steffen

Diffusion means permutation of bit or byte positions !

Modern Transposition ciphers take in N bits and permute using lookup table : called P-Boxes

Stream Cipher

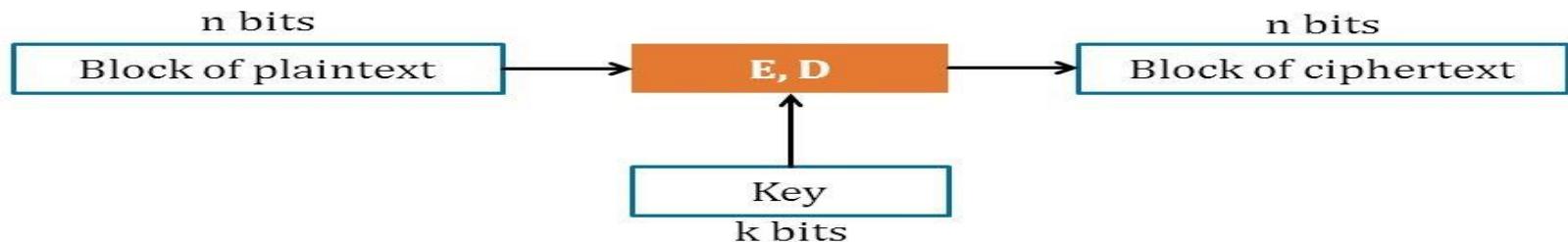
- Combine each bit of keystream with bit of plaintext to get bit of ciphertext
 - $c(i) = ks(i) \oplus m(i)$
 - $m(i) = ks(i) \oplus c(i)$



(a) Stream Cipher Using Algorithmic Bit Stream Generator

Block ciphers

- Message to be encrypted is processed in blocks of k bits (e.g., 64-bit blocks).



1. 3DES: $n = 64$ bits, $k = 168$ bits
2. AES: $n = 128$ bits, $k = 128, 192, 256$ bits



Thank You...

Any queries...?

manik.chavan@walchandsangli.ac.in

Classical Encryption Techniques

4CS401

M. K. Chavan,

Asst. Professor, CSE Department,
Walchand College of Engineering, Sangli

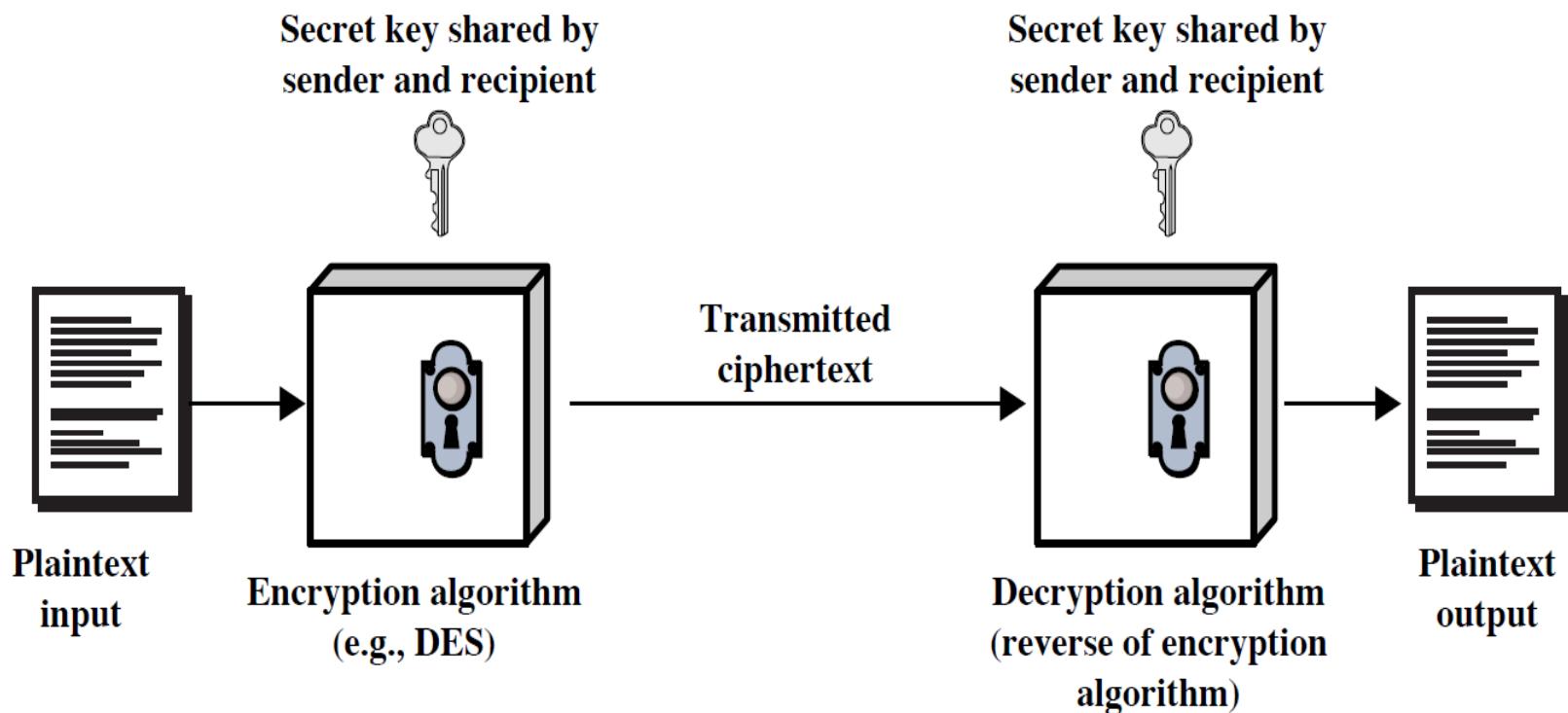
Classical encryption techniques

- As opposed to **modern cryptography**
- Goals:
 - to introduce basic concepts & terminology of encryption
 - to prepare us for studying modern cryptography

Basic terminology

- **Plaintext:** original message to be encrypted
- **Ciphertext:** the encrypted message
- **Enciphering or encryption:** the process of converting plaintext into ciphertext
- **Encryption algorithm:** performs encryption
 - Two inputs: a **plaintext** and a **secret key**

Symmetric Cipher Model



- Deciphering or decryption: recovering plaintext from ciphertext
- Decryption algorithm: performs decryption
 - Two inputs: ciphertext and secret key
- Secret key: same key used for encryption and decryption
 - Also referred to as a symmetric key

- **Cipher or cryptographic system** : a scheme for encryption and decryption
- **Cryptography**: science of studying ciphers
- **Cryptanalysis**: science of studying attacks against cryptographic systems
- **Cryptology**: cryptography + cryptanalysis

Ciphers

- **Symmetric cipher:** same key used for encryption and decryption
 - **Block cipher:** encrypts a block of plaintext at a time (typically 64 or 128 bits)
 - **Stream cipher:** encrypts data one bit or one byte at a time
- **Asymmetric cipher:** different keys used for encryption and decryption

Symmetric Encryption

- or conventional / secret-key / single-key
- sender and recipient share a common key
- all classical encryption algorithms are symmetric
- The only type of ciphers prior to the invention of asymmetric-key ciphers in 1970's
- by far most widely used

Symmetric Encryption

- Mathematically:

$$Y = E_K(X) \quad \text{or} \quad Y = E(K, X)$$

$$X = D_K(Y) \quad \text{or} \quad X = D(K, Y)$$

- X = plaintext
- Y = ciphertext
- K = secret key
- E = encryption algorithm
- D = decryption algorithm
- Both E and D are known to public

Cryptanalysis

- Objective: to recover the plaintext of a ciphertext or, more typically, to recover the secret key.
- Two general approaches:
 - **brute-force** attack
 - **non-brute-force** attack (cryptanalytic attack)

Brute-Force Attack

- Try every key to decipher the ciphertext.
- On average, need to try half of all possible keys
- Time needed proportional to size of **key space**

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/ μ s	Time required at 10^6 decryptions/ μ s
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6.4 \times 10^{12}$ years	6.4×10^6 years

Classical Cryptography

- Sender, receiver share common key
 - Keys may be the same, or trivial to derive from one another
 - Sometimes called *symmetric cryptography*
- Two basic types
 - Transposition ciphers
 - Substitution ciphers
 - Combinations are called *product ciphers*

Classical Ciphers

- Plaintext is viewed as a sequence of elements (e.g., bits or characters)
- **Substitution cipher:** replacing each element of the plaintext with another element.
- **Transposition (or permutation) cipher:** rearranging the order of the elements of the plaintext.
- **Product cipher:** using multiple stages of substitutions and transpositions

Transposition Cipher

- Rearrange letters in plaintext to produce ciphertext
- Example (Rail-Fence Cipher or 2-columnar transposition)
 - Plaintext is HELLO WORLD
 - HE
LL
OW
OR
LD
 - Ciphertext is HLOOL ELWRD

Transposition Cipher

- Generalize to n-columnar transpositions
- Example 3-columnar
 - HEL
LOW
ORL
DXX
 - HLODEORXLWLX

Caesar Cipher

- Earliest known substitution cipher
- Invented by Julius Caesar
- Each letter is replaced by the letter three positions further down the alphabet.
- Plain: 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
Cipher: 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
- Example: ohio state → RKL R VWD W H

Caesar Cipher

- Mathematically, map letters to numbers:

a, b, c, . . . , x, y, z

0, 1, 2, . . . , 23, 24, 25

- Then the general Caesar cipher is:

$$c = E_K(p) = (p + k) \bmod 26$$

$$p = D_K(c) = (c - k) \bmod 26$$

- Can be generalized with any alphabet.

Cryptanalysis of Caesar Cipher

- Key space: {0, 1, ..., 25}
- Vulnerable to brute-force attacks.
- E.g., break ciphertext “DWWDFN“

Monoalphabetic Substitution Cipher

- Shuffle the letters and map each plaintext letter to a different random ciphertext letter:

Plain letters: abcdefghijklmnopqrstuvwxyz

Cipher letters: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext: if we wish to replace letters

Ciphertext: WIRFRWAJUHYFTSDVFSUUUFYA

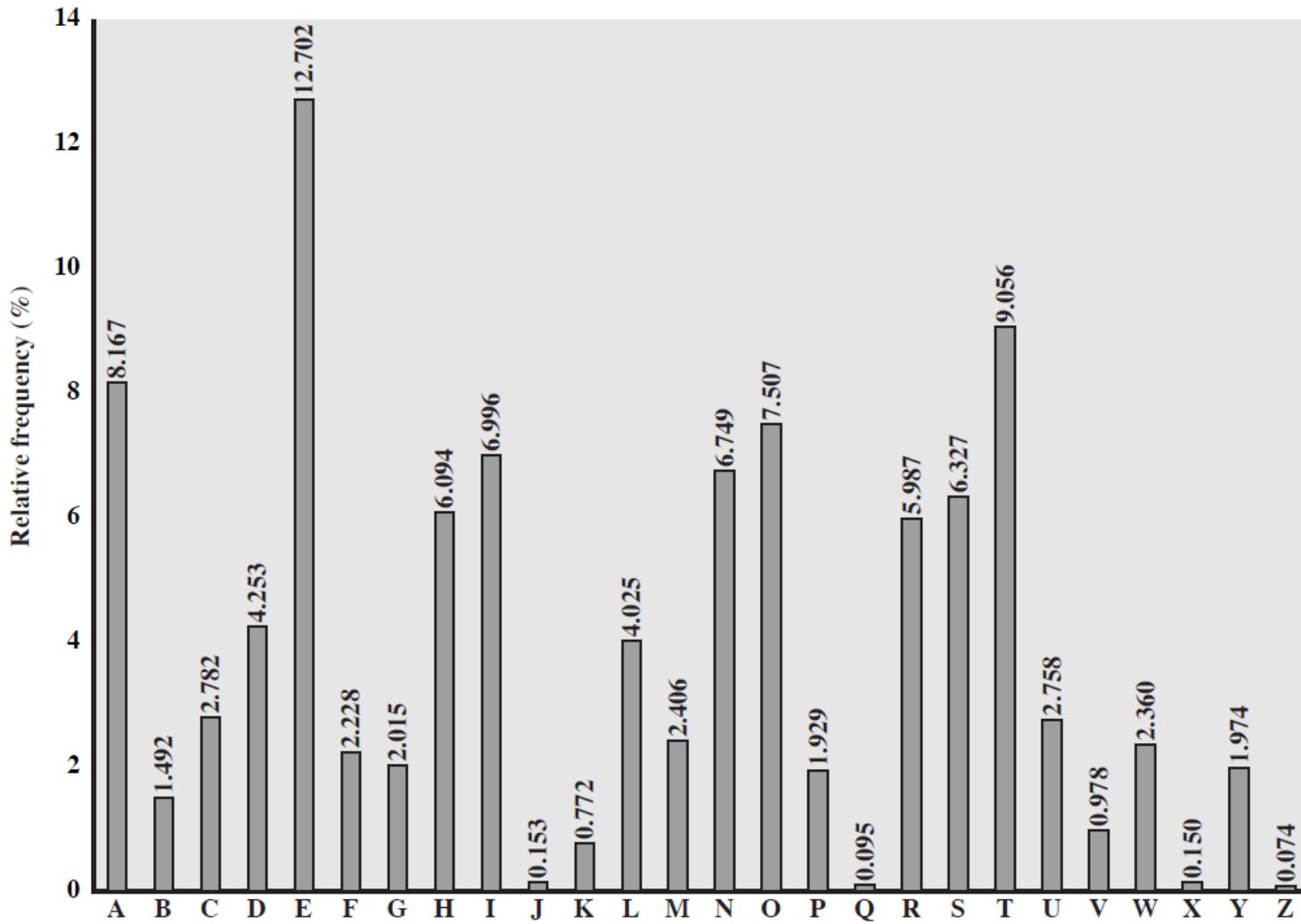
Monoalphabetic Cipher Security

- With so many keys, it is secure against brute-force attacks.
- But not secure against some cryptanalytic attacks.
- Problem is language characteristics.

Language Statistics and Cryptanalysis

- Human languages are not random.
- Letters are not equally frequently used.
- In English, E is by far the most common letter, followed by T, R, N, I, O, A, S.
- Other letters like Z, J, K, Q, X are fairly rare.
- There are tables of single, double & triple letter frequencies for various languages

English Letter Frequencies



Statistics for double & triple letters

- In decreasing order of frequency

- Double letters:

th he an in er re es on, ...

- Triple letters:

the and ent ion tio for nde, ...

Use in Cryptanalysis

- Key concept: monoalphabetic substitution does not change relative letter frequencies
- To attack, we
 - calculate letter frequencies for ciphertext
 - compare this distribution against the known one

Example Cryptanalysis

- Given ciphertext:

UZQSOVUOHHMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- Count relative letter frequencies (see next page)
- Guess $\{P, Z\} = \{e, t\}$
- Of double letters, ZW has highest frequency, so guess ZW = th and hence ZWP = the
- Proceeding with trial and error finally get:

it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow

Letter frequencies in ciphertext

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

Playfair Cipher

- Not even the large number of keys in a monoalphabetic cipher provides security.
- One approach to improving security is to **encrypt multiple letters at a time**.
- The **Playfair Cipher** is the best known such cipher.
- Invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair.

Playfair Key Matrix

- Use a 5×5 matrix.
- Fill in letters of the key (w/o duplicates).
- Fill the rest of matrix with other letters.
- E.g., key = **MONARCHY**.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Encrypting and Decrypting

Plaintext is encrypted two letters at a time.

1. If a pair is a repeated letter, insert filler like 'X'.
2. If both letters fall in the same row, replace each with the letter to its right (circularly).
3. If both letters fall in the same column, replace each with the letter below it (circularly).
4. Otherwise, each letter is replaced by the letter in the same row but in the column of the other letter of the pair.

Polyalphabetic Substitution Ciphers

- A sequence of monoalphabetic ciphers ($M_1, M_2, M_3, \dots, M_k$) is used in turn to encrypt letters.
- A key determines which sequence of ciphers to use.
- Each plaintext letter has multiple corresponding ciphertext letters.
- This makes cryptanalysis harder since the letter frequency distribution will be flatter.

Vigenère Cipher

- Simplest polyalphabetic substitution cipher
- Consider the set of all Caesar ciphers:
 $\{ C_a, C_b, C_c, \dots, C_z \}$
- Key: e.g. **security**
- Encrypt each letter using $C_s, C_e, C_c, C_u, C_r, C_i, C_t, C_y$ in turn.
- Repeat from start after C_y .
- Decryption simply works in reverse.

Example of Vigenère Cipher

- Keyword: *deceptive*

key: deceptive deceptive deceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCOYGLMGJ

$$C = (p+k) \bmod 26$$

Transposition Ciphers

- Also called **permutation** ciphers.
- Shuffle the plaintext, without altering the actual letters used.
- Example: Row Transposition Ciphers

Rail Fence cipher

- write message letters out diagonally over a number of rows
- then read off cipher row by row
- eg. write message out as:

m e m a t r h t g p r y
e t e f e t e o a a t

- giving ciphertext

MEMATRHTGPRYETEFETOAAAT

Row Transposition Ciphers

- Plaintext is written row by row in a rectangle.
- Ciphertext: write out the **columns** in an order specified by a key.

Key: 4 3 1 2 5 6 7

Plaintext:

a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
w	o	a	m	x	y	z

Ciphertext: TTNAAPMTSUOAODWCOIXKNLYPETZ

Product Ciphers

- Uses a sequence of substitutions and transpositions
 - Harder to break than just substitutions or transpositions
- This is a bridge from classical to modern ciphers.

Unconditional & Computational Security

- A cipher is **unconditionally secure** if it is secure no matter how much resources (time, space) the attacker has.
- A cipher is **computationally secure** if the best algorithm for breaking it will require so much resources (e.g., 1000 years) that practically the cryptosystem is secure.
- All the ciphers we have examined are not unconditionally secure.

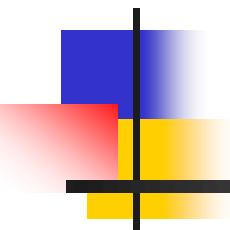
An unconditionally Secure Cipher

Vernam's one-time pad cipher

- Key = $k_1 k_2 k_3 k_4 \dots$ (random, used one-time only)
- Plaintext = $m_1 m_2 m_3 m_4 \dots$
- Ciphertext = $c_1 c_2 c_3 c_4 \dots$
where $c_i = m_i \oplus k_i$
- Can be proved to be unconditionally secure.

Summary

- Have considered:
 - classical cipher techniques and terminology
 - monoalphabetic substitution ciphers
 - cryptanalysis using letter frequencies
 - Playfair cipher
 - polyalphabetic ciphers
 - transposition ciphers
 - product ciphers



Module-II

Symmetric Key Cryptography

Symmetric Key Cryptography

Module-II

Contents:

SYMMETRIC KEY CIPHERS:

Block cipher Principles of DES

Strength of DES

Block cipher design principles

Block cipher mode of operation

Evaluation criteria for AES – Advanced Encryption Standard

RC4

29-3 MODERN CIPHERS

The traditional symmetric-key ciphers that we have studied so far are character-oriented ciphers. With the advent of the computer, we need bit-oriented ciphers. This is because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream. A modern block cipher can be either a block cipher or a stream cipher.

Modern Block Ciphers

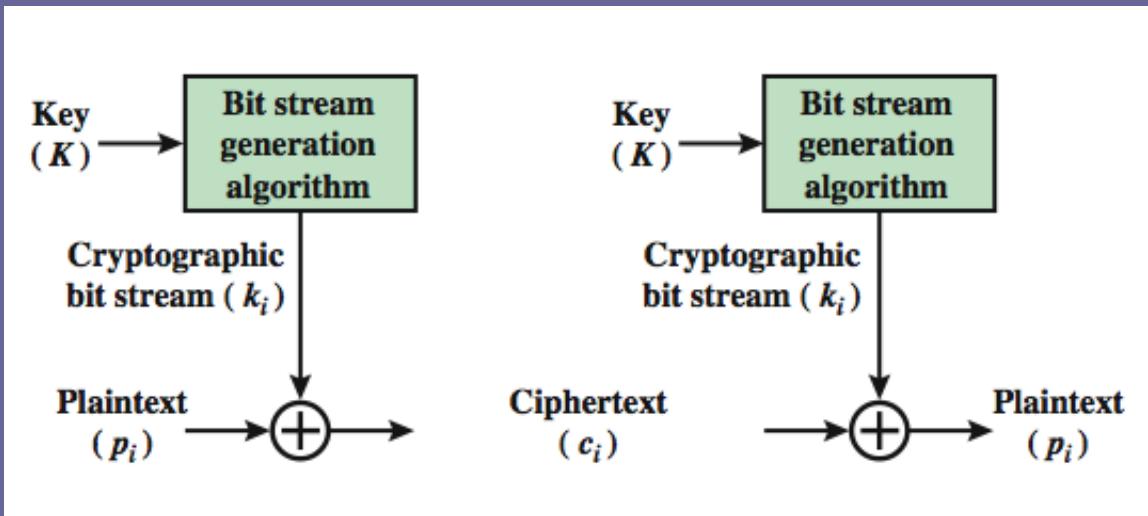
- now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy /authentication services
- focus on DES (Data Encryption Standard)
- to illustrate block cipher design principles



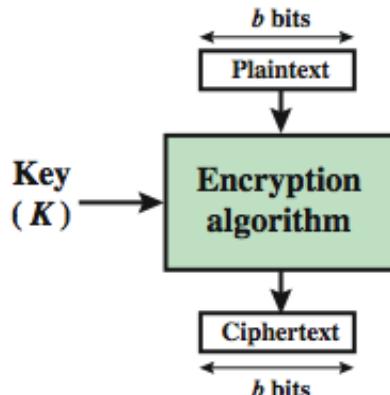
Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
 - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
 - better analysed
 - broader range of applications

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator

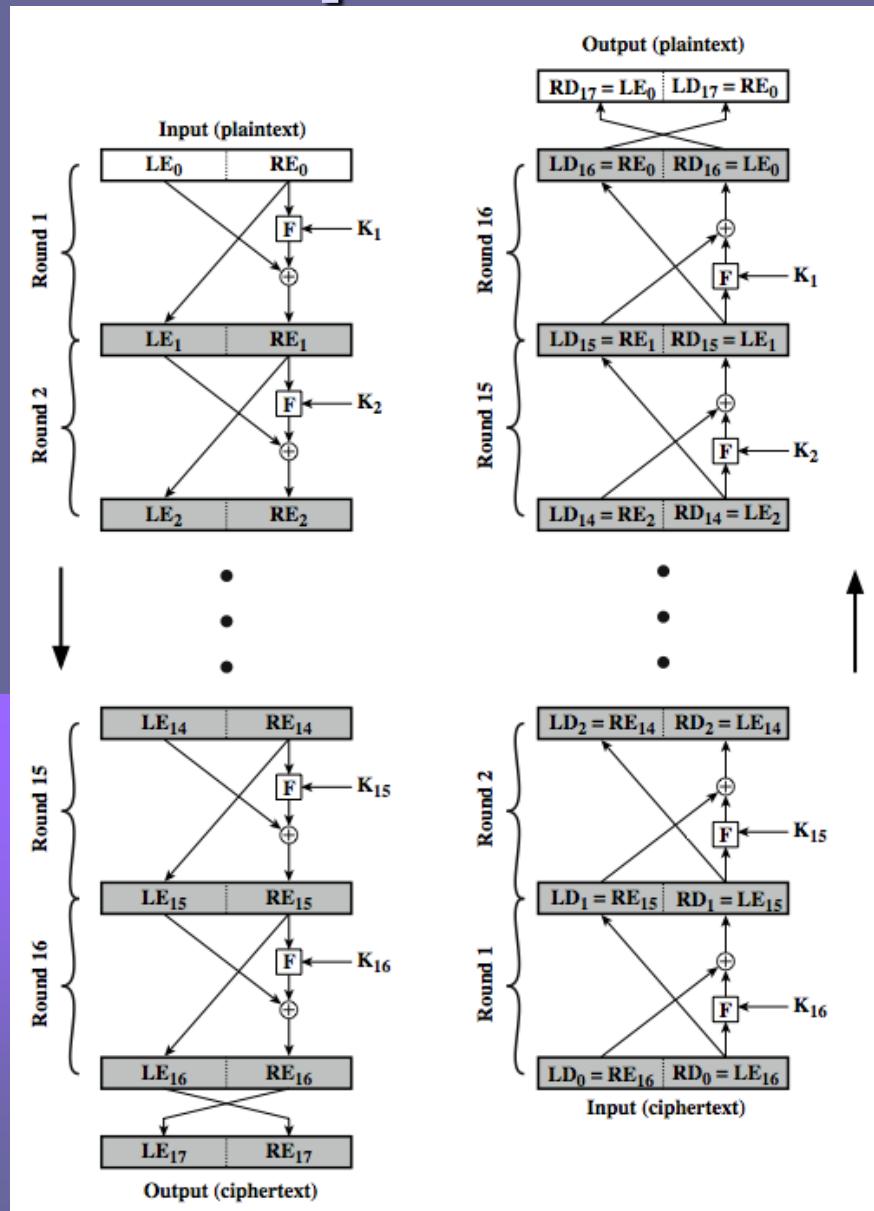


(b) Block Cipher

Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
 - based on concept of invertible product cipher
- partitions input block into two halves
 - process through multiple rounds which
 - perform a substitution on left data half
 - based on round function of right half & subkey
 - then have permutation swapping halves

Feistel Cipher Structure



Feistel Cipher Design Elements

- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis

Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
 - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security



DES History

- IBM developed Lucifer cipher
 - by team led by Feistel in late 60's
 - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
 - in choice of 56-bit key (vs Lucifer 128-bit)
 - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
 - especially in financial applications
 - still standardised for legacy application use

Figure 29.7 A modern block cipher

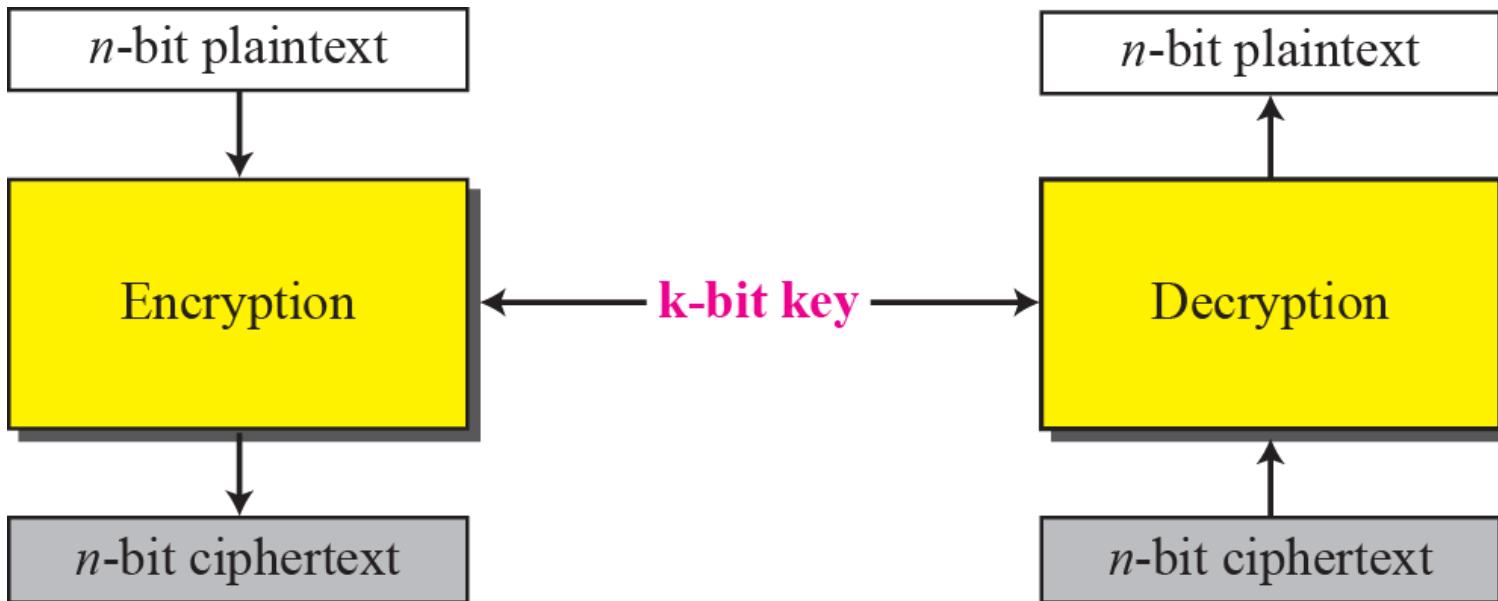


Figure 29.8 Components of a modern block cipher

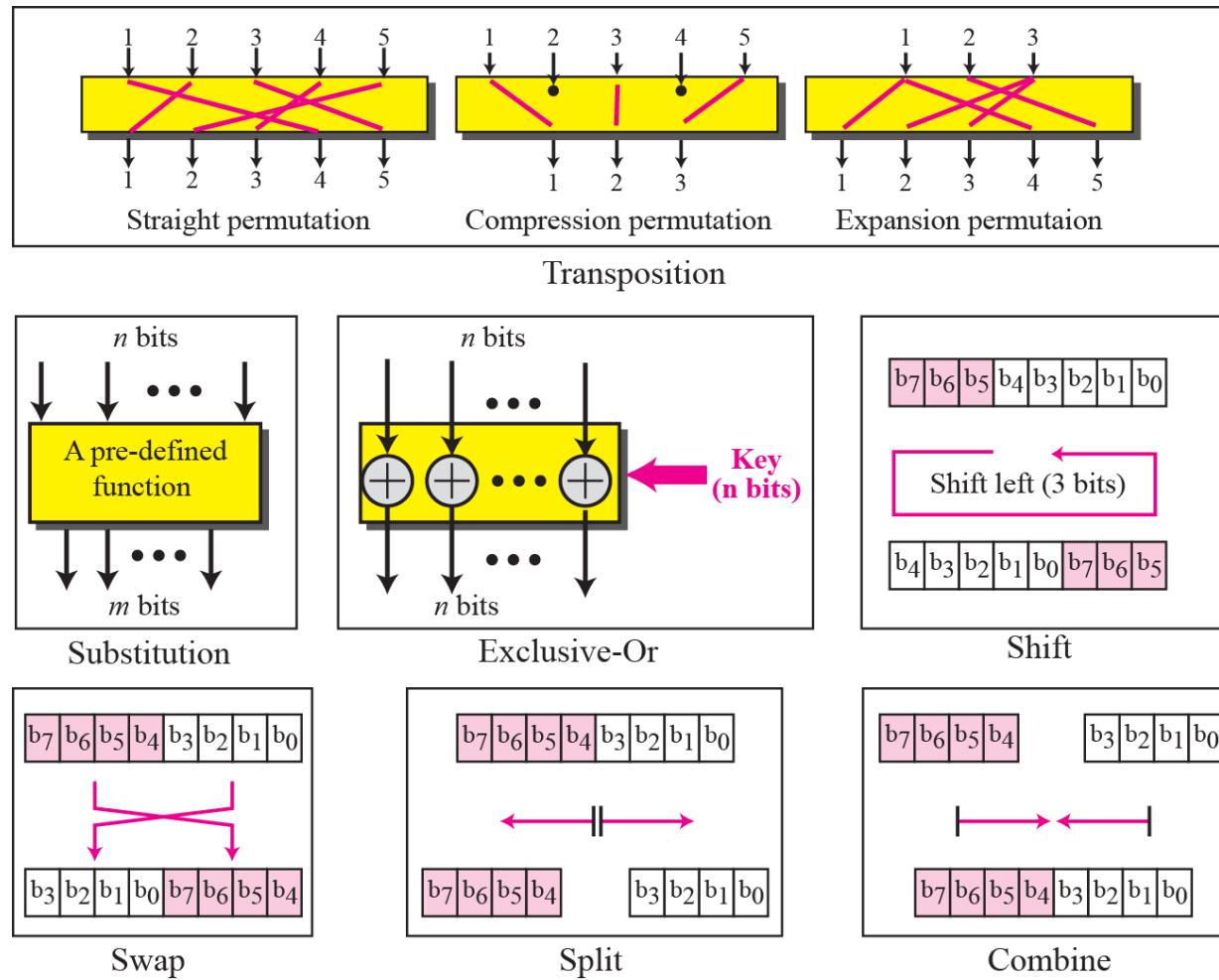


Figure 29.9 General structure of DES

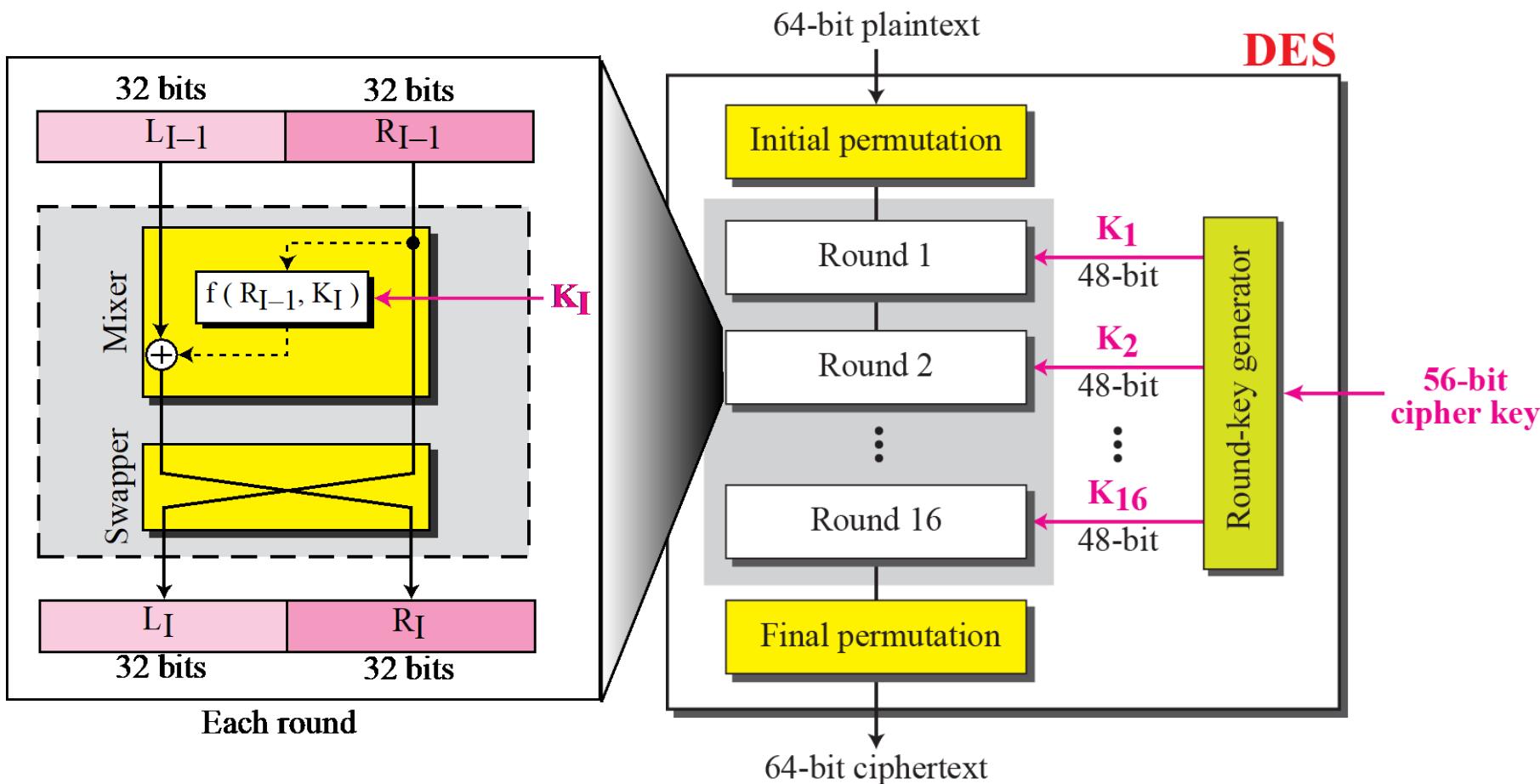
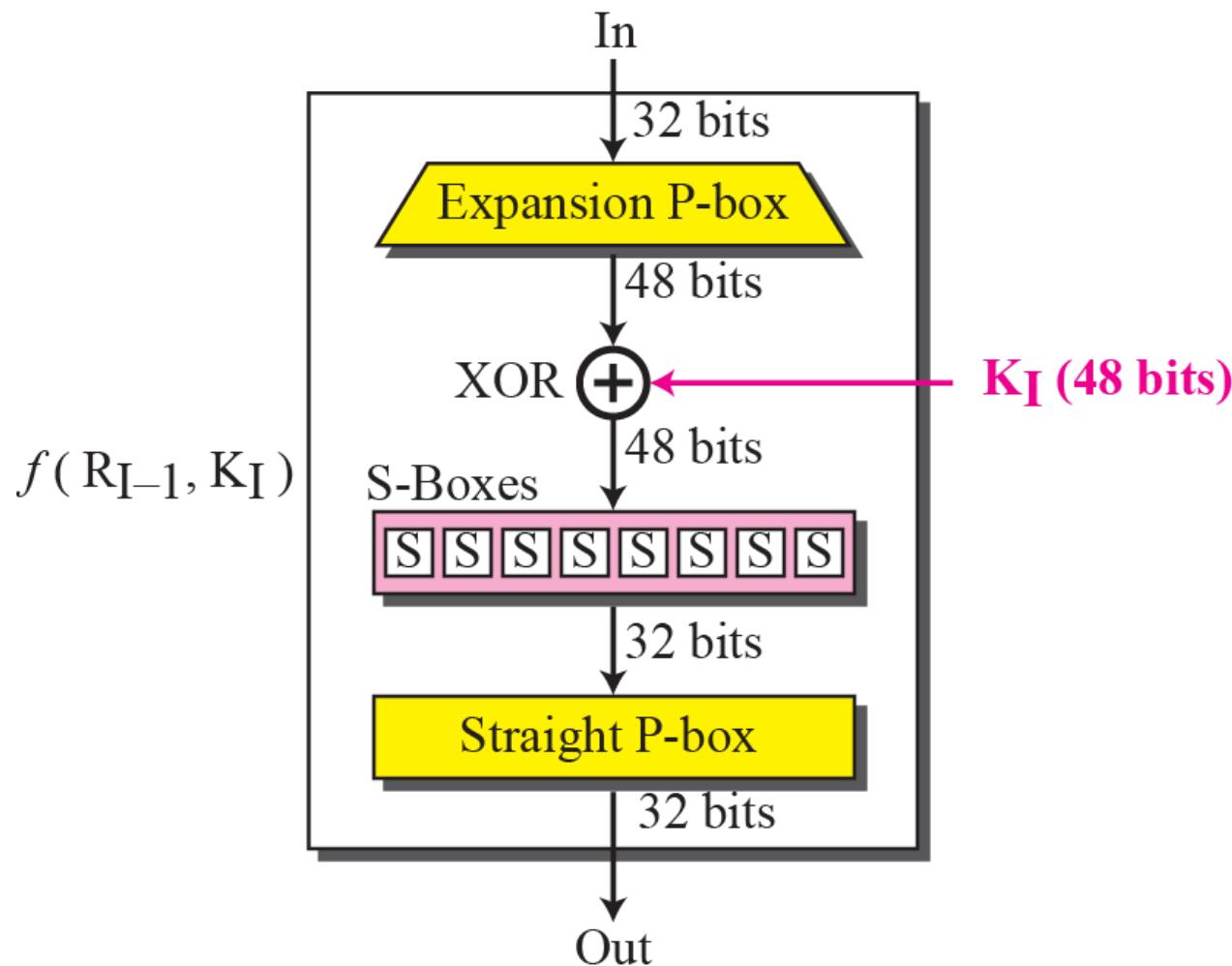


Figure 29.10 DES function



DES Round in Full

Right Half i-1

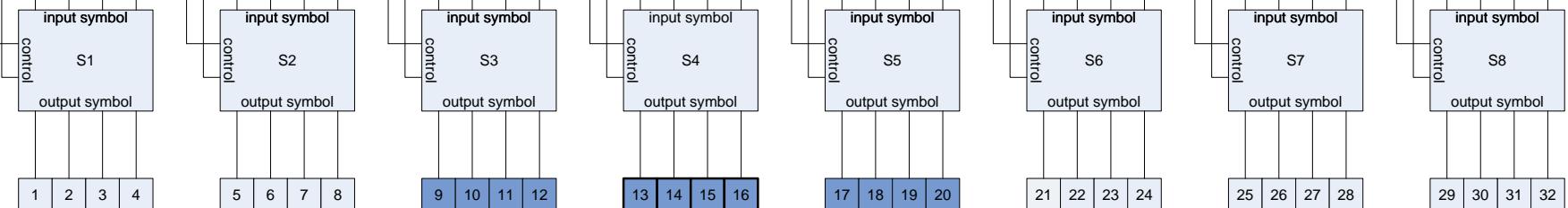
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

32	1	2	3	4	5	4	5	6	7	8	9	10	11	12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21	22	23	24	25	26	27	28	29	30	31	32	1
----	---	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---

Round Key i

⊕	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10	2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25
----	---	----	----	----	----	----	----	---	----	----	----	---	----	----	----	---	---	----	----	----	----	---	---	----	----	----	---	----	----	---	----

Left Half i-1

⊕	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Right Half i

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Figure 29.11 Key generation



Example 29.4

We choose a random plaintext block, a random key, and a computer program to determine what the ciphertext block would be (all in hexadecimal):

Plaintext:
123456ABCD132536

Key:
AABB09182736CCDD

CipherText:
C0B7A8D05F3A829C

Example 29.5

To check the effectiveness of DES, when a single bit is changed in the input, let us use two different plaintexts with only one single bit difference. The two ciphertexts are completely different without even changing the key:

Plaintext :	Key :	Ciphertext :
0000000000000000	22234512987ABB23	4789FD476E82A5F1
Plaintext :	Key :	Ciphertext :
00000000000000001	22234512987ABB23	0A4ED5C15A63FEA3

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits.



Advance Encryption Standard

Topics

- ▶ **Origin of AES**
- ▶ Basic AES
- ▶ Inside Algorithm
- ▶ Final Notes



Origins

- ▶ A replacement for DES was needed
 - ▶ Key size is too small
- ▶ Can use Triple-DES – but slow, small block
- ▶ US NIST issued call for ciphers in 1997
- ▶ 15 candidates accepted in Jun 98
- ▶ 5 were shortlisted in Aug 99



AES Competition Requirements

- ▶ symmetric key block cipher
- ▶ 128-bit data, 128/192/256-bit keys
- ▶ Stronger & faster than Triple-DES
- ▶ Provide full specification & design details



AES Evaluation Criteria

- ▶ criteria
 - ▶ general security
 - ▶ ease of software & hardware implementation
 - ▶ implementation attacks
 - ▶ flexibility (in en/decrypt, keying, other factors)



AES Shortlist

- ▶ After testing and evaluation, shortlist in Aug-99
 - ▶ MARS (IBM) - complex, fast, high security margin
 - ▶ RC6 (USA) - v. simple, v. fast, low security margin
 - ▶ Rijndael (Belgium) - clean, fast, good security margin
 - ▶ Serpent (Euro) - slow, clean, v. high security margin
 - ▶ Twofish (USA) - complex, v. fast, high security margin

Rijndae: pronounce “Rain-Dahl”



The AES Cipher - Rijndael

- ▶ Rijndael was selected as the AES in Oct-2000
 - ▶ Designed by Vincent Rijmen and Joan Daemen in Belgium
 - ▶ Issued as FIPS PUB 197 standard in Nov-2001
- ▶ An **iterative** rather than **Feistel** cipher
 - ▶ processes data as block of 4 columns of 4 bytes (128 bits)
 - ▶ operates on entire data block in every round
- ▶ Rijndael design:
 - ▶ simplicity
 - ▶ has 128/192/256 bit keys, 128 bits data
 - ▶ resistant against known attacks
 - ▶ speed and code compactness on many CPUs



V. Rijmen



J. Daemen

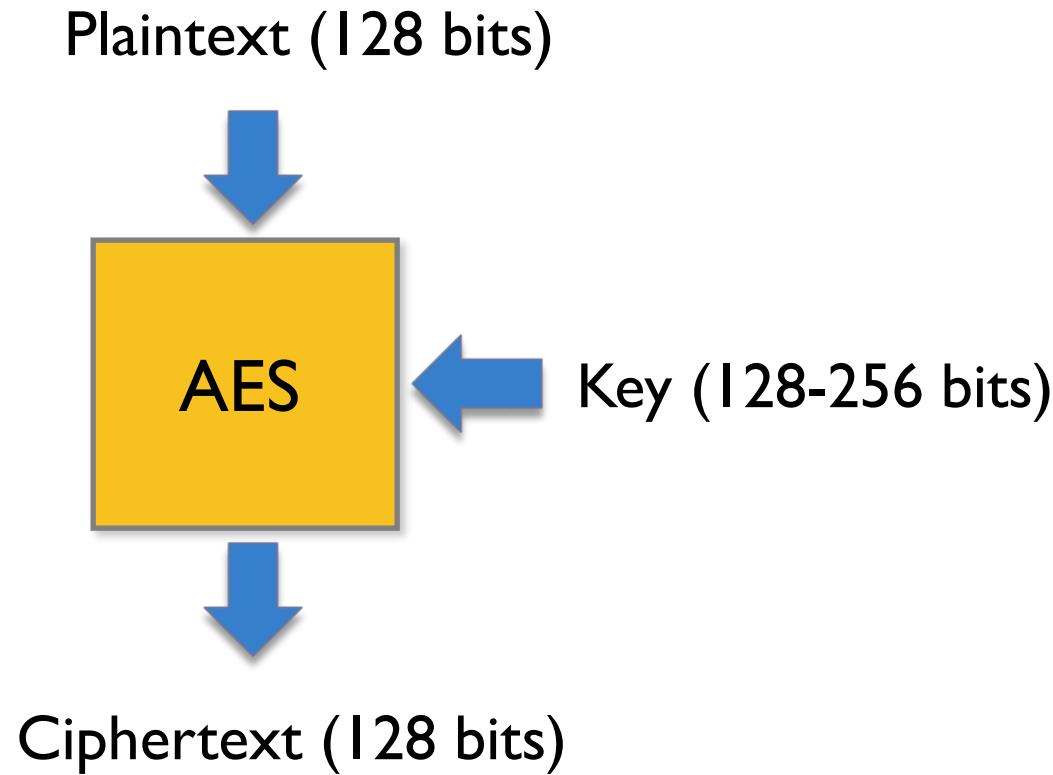


Topics

- ▶ Origin of AES
- ▶ **Basic AES**
- ▶ Inside Algorithm
- ▶ Final Notes

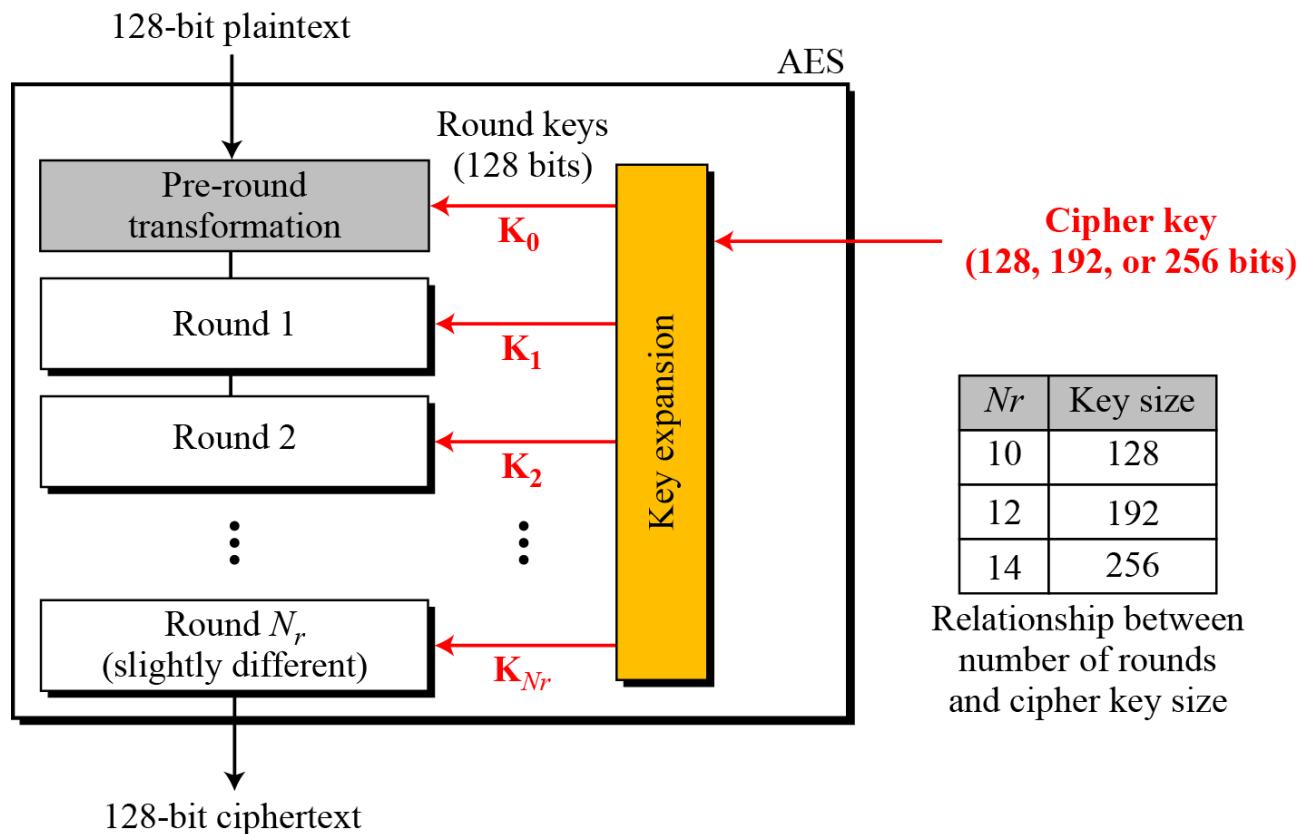


AES Conceptual Scheme



Multiple rounds

- ▶ Rounds are (almost) identical
 - ▶ First and last round are a little different



High Level Description

Key Expansion

- Round keys are derived from the cipher key using Rijndael's key schedule

Initial Round

- AddRoundKey : Each byte of the state is combined with the round key using bitwise xor

Rounds

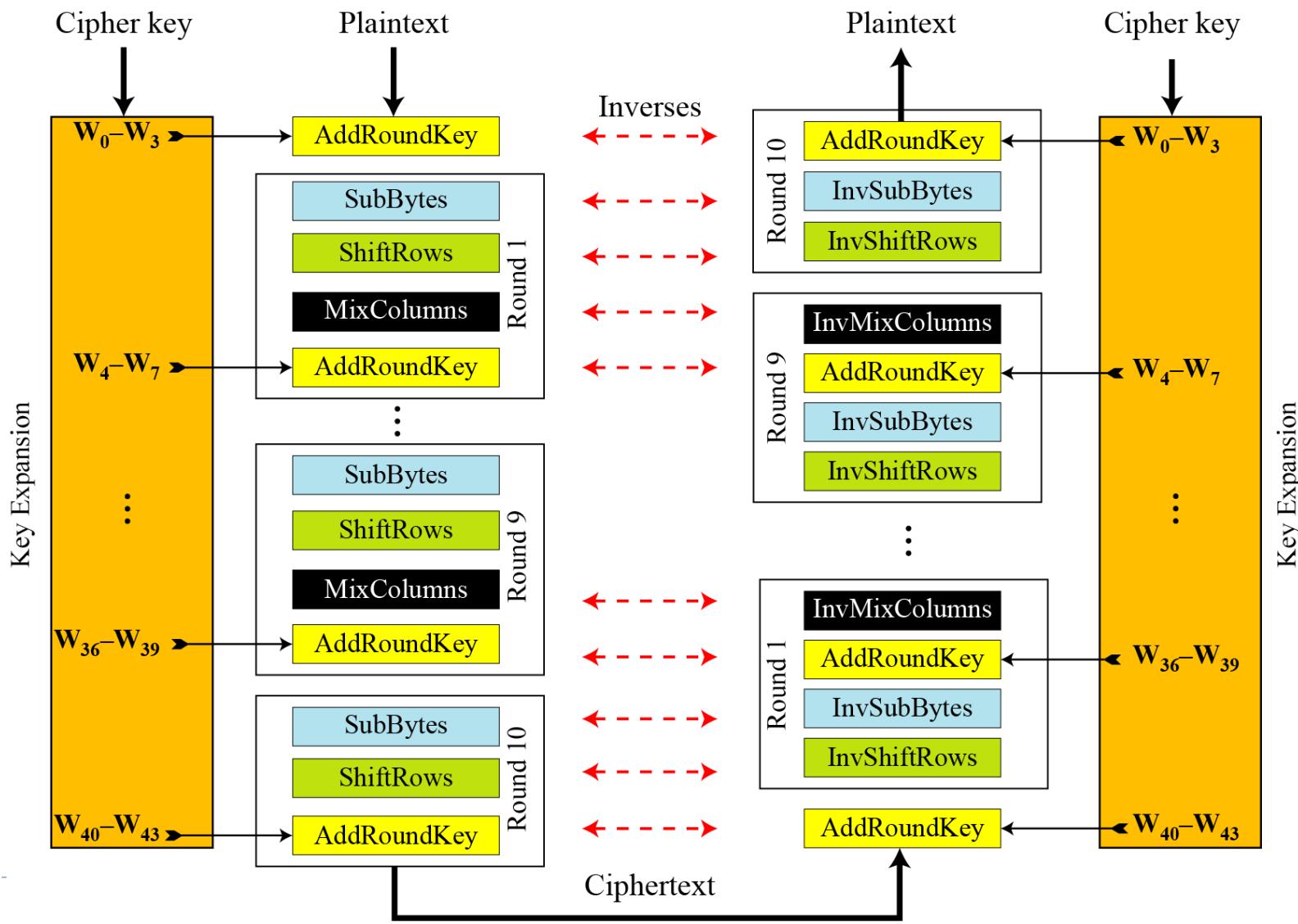
- SubBytes : non-linear substitution step
- ShiftRows : transposition step
- MixColumns : mixing operation of each column.
- AddRoundKey

Final Round

- SubBytes
- ShiftRows
- AddRoundKey

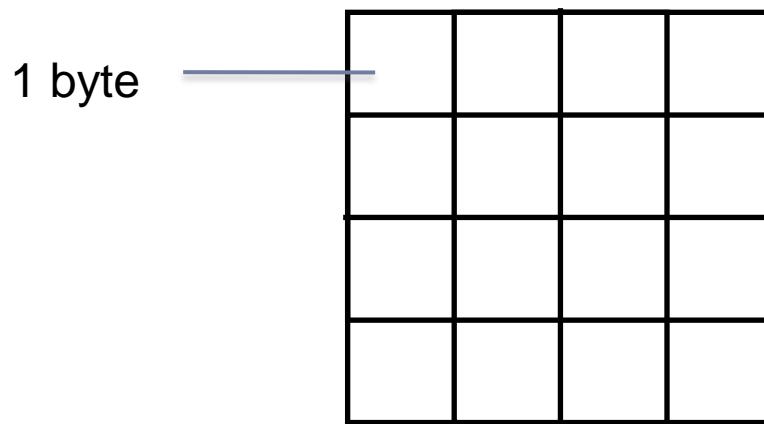
No MixColumns

Overall Structure

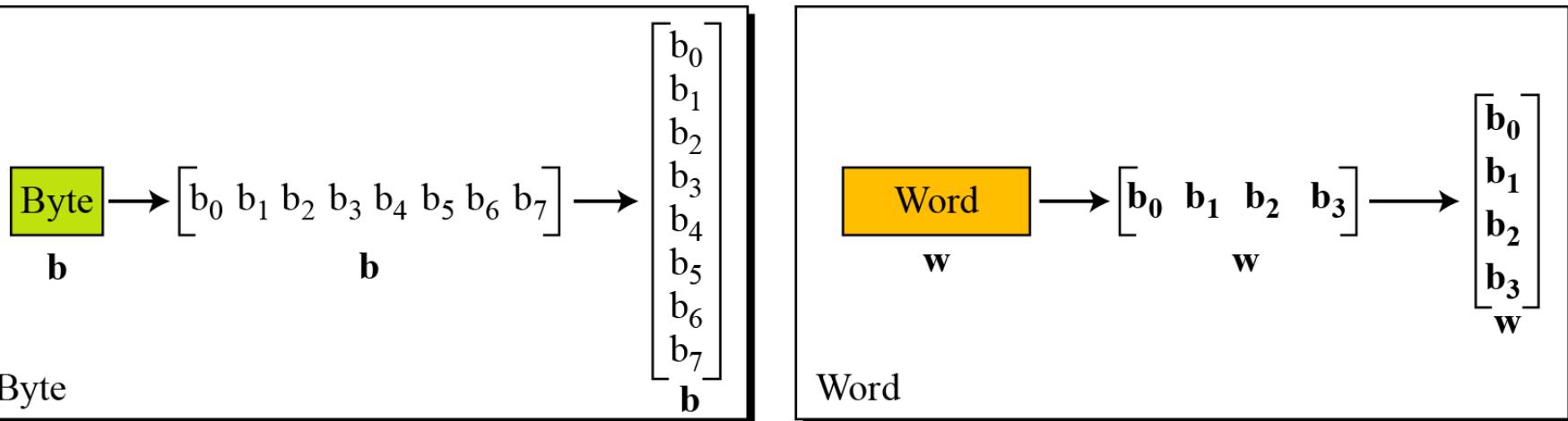


128-bit values

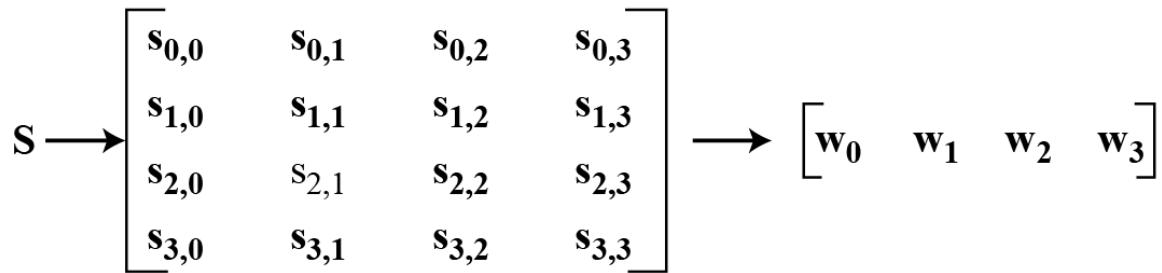
- ▶ Data block viewed as 4-by-4 table of bytes
- ▶ Represented as 4 by 4 matrix of 8-bit bytes.
- ▶ Key is expanded to array of 32 bits words



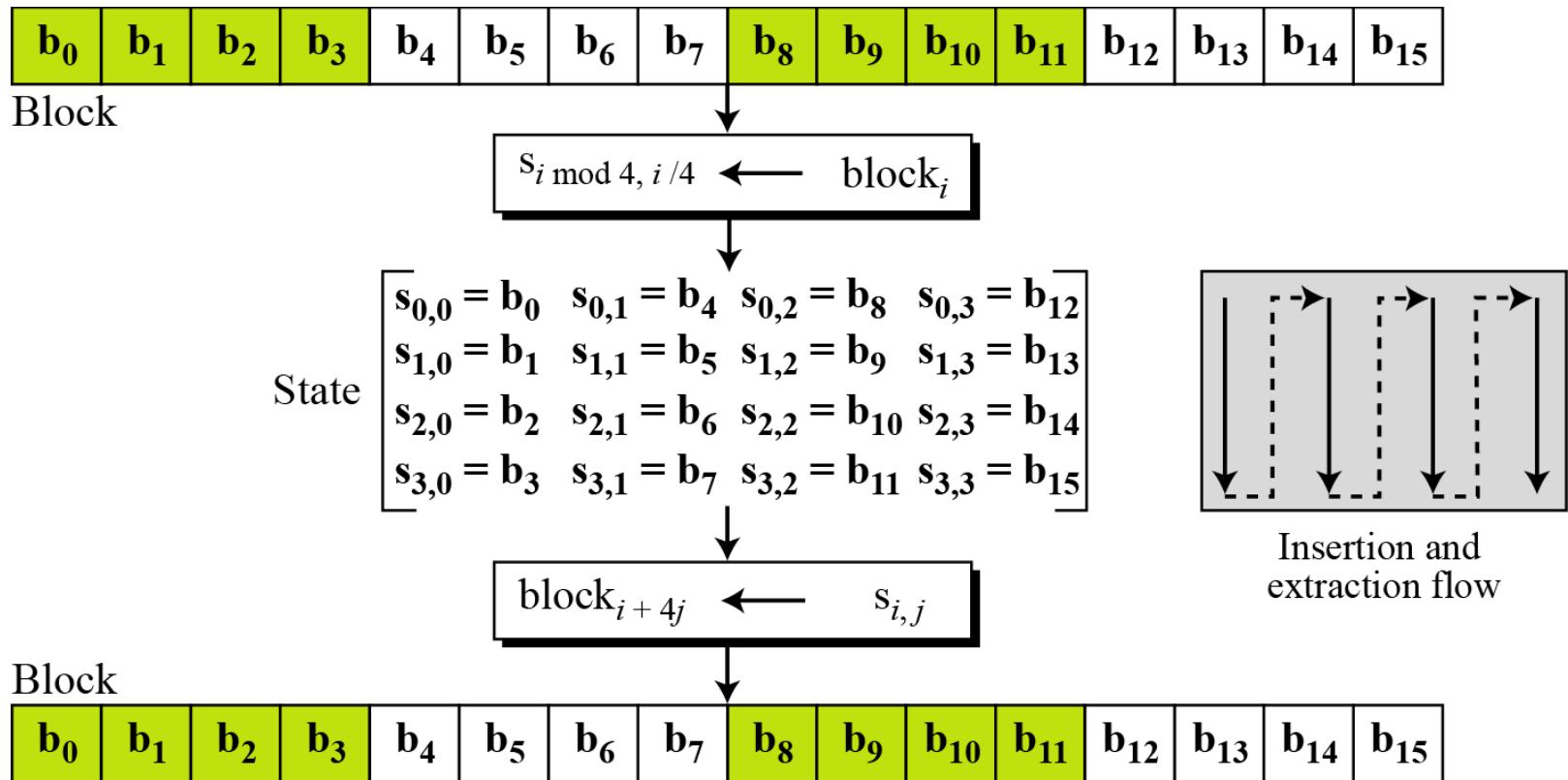
Data Unit



Block



Unit Transformation



Changing Plaintext to State

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19
$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$ State																

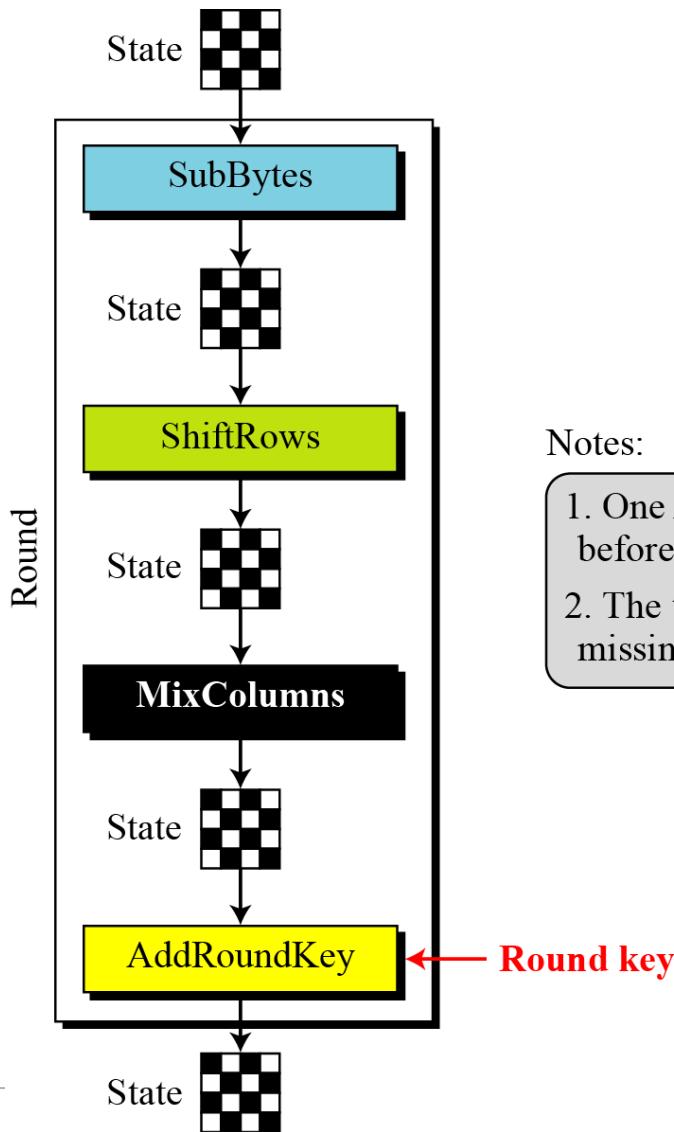


Topics

- ▶ Origin of AES
- ▶ Basic AES
- ▶ **Inside Algorithm**
- ▶ Final Notes



Details of Each Round



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

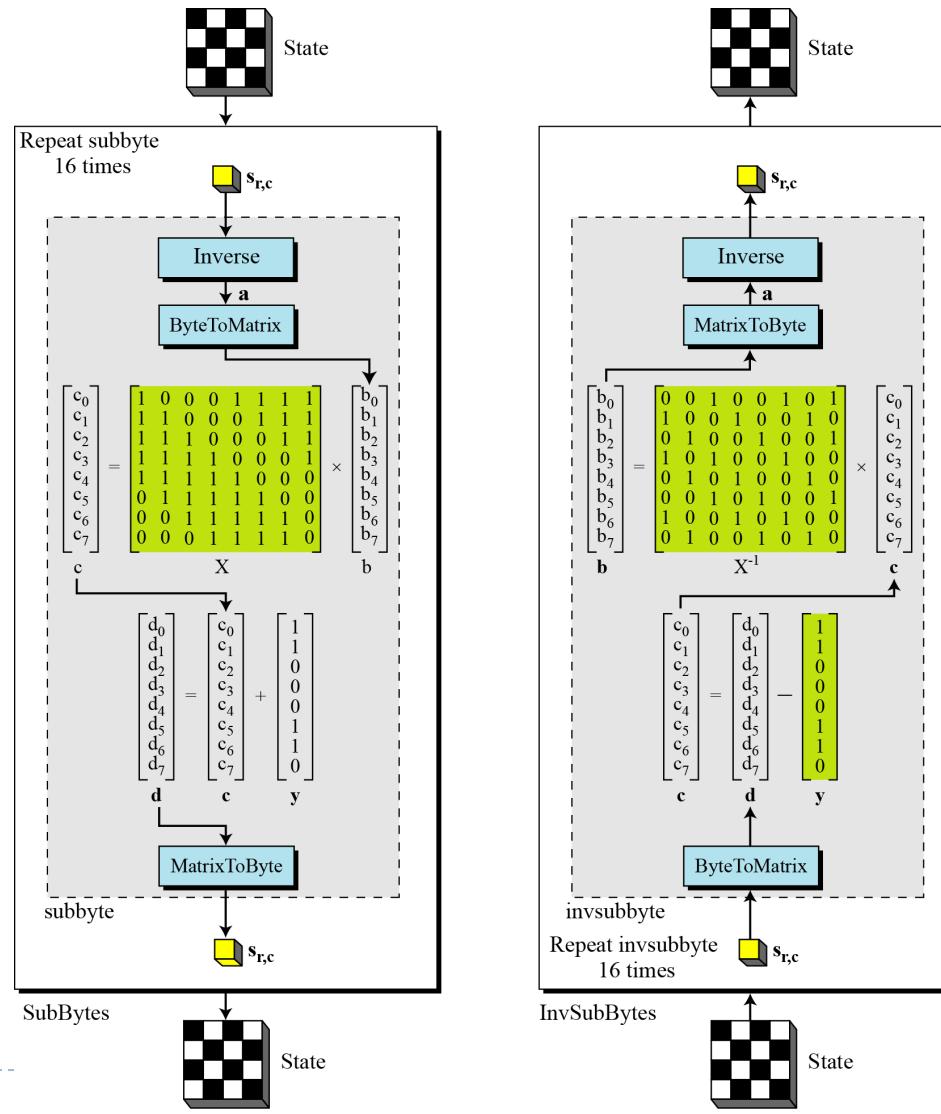
SubBytes: Byte Substitution

- ▶ A simple substitution of each byte
 - ▶ provide a confusion
- ▶ Uses one S-box of 16×16 bytes containing a permutation of all 256 8-bit values
- ▶ Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
 - ▶ eg. byte {95} is replaced by byte in row 9 column 5
 - ▶ which has value {2A}
- ▶ S-box constructed using defined transformation of values in Galois Field- $GF(2^8)$

Galois : pronounce “Gal-Wa”

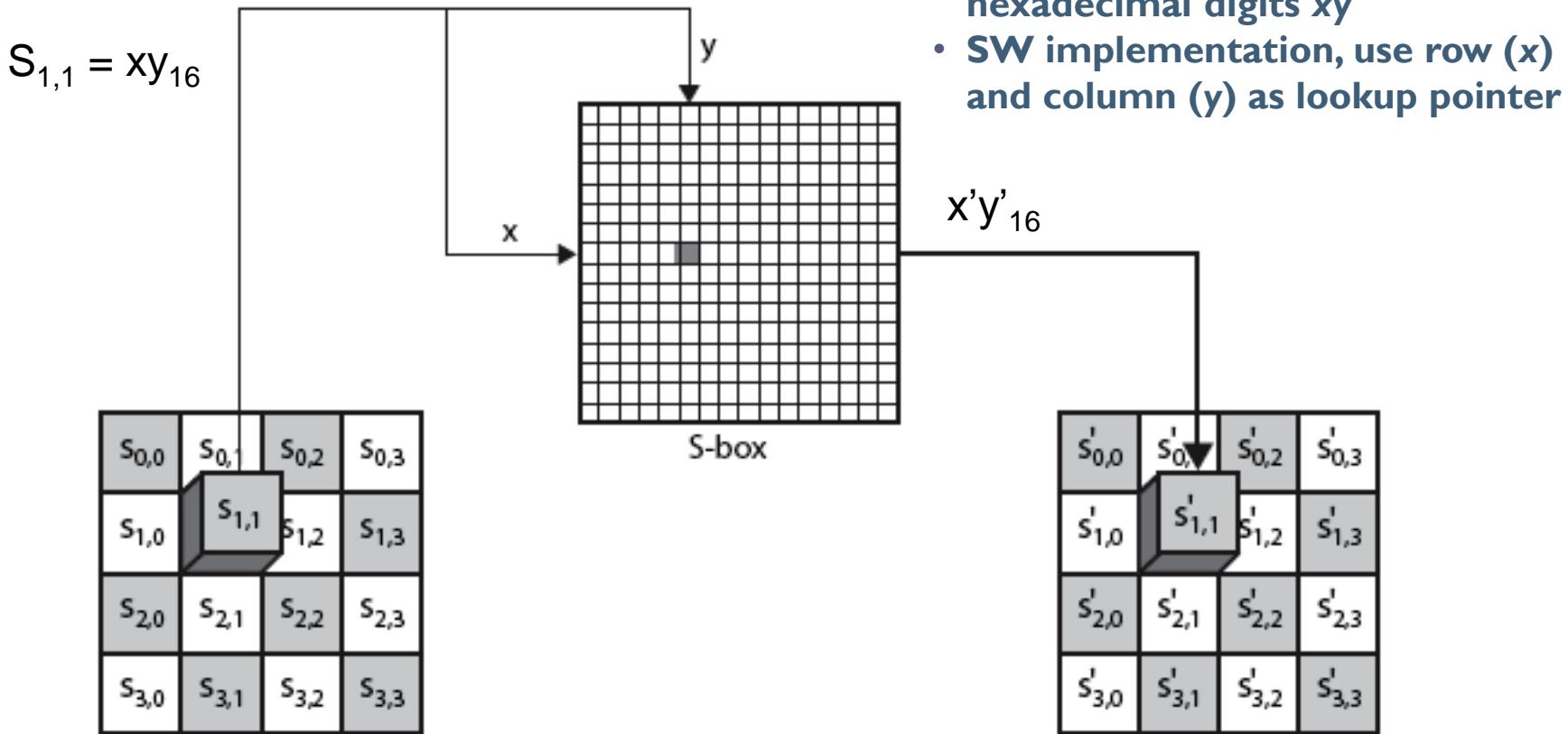


SubBytes and InvSubBytes



SubBytes Operation

- The SubBytes operation involves 16 independent byte-to-byte transformations.



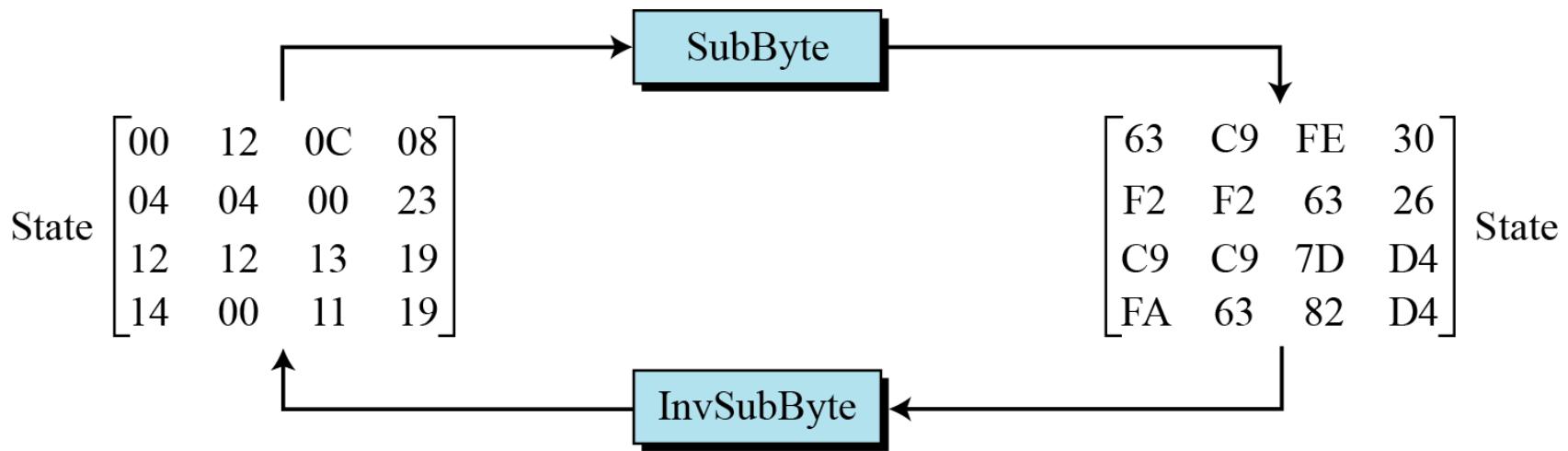
SubBytes Table

► Implement by Table Lookup

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

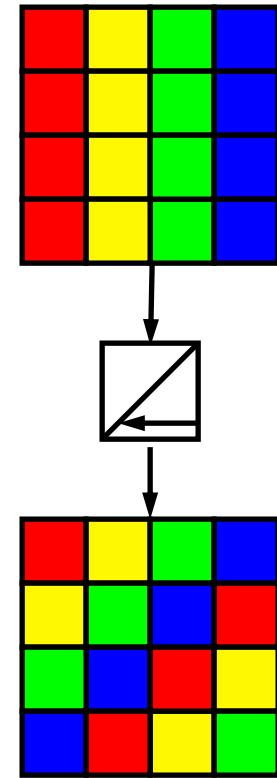
Sample SubByte Transformation

- The SubBytes and InvSubBytes transformations are inverses of each other.

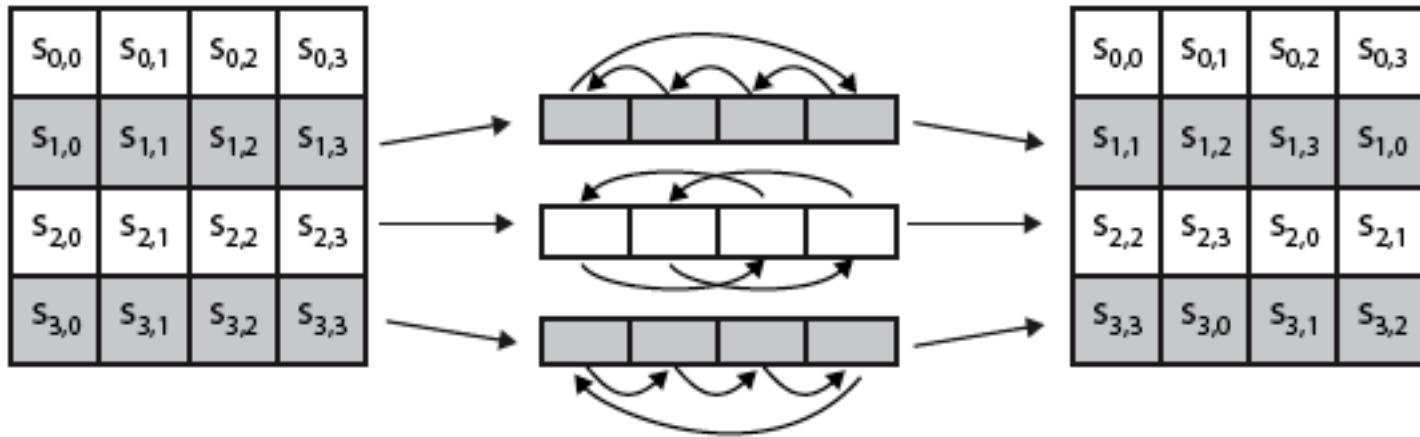


ShiftRows

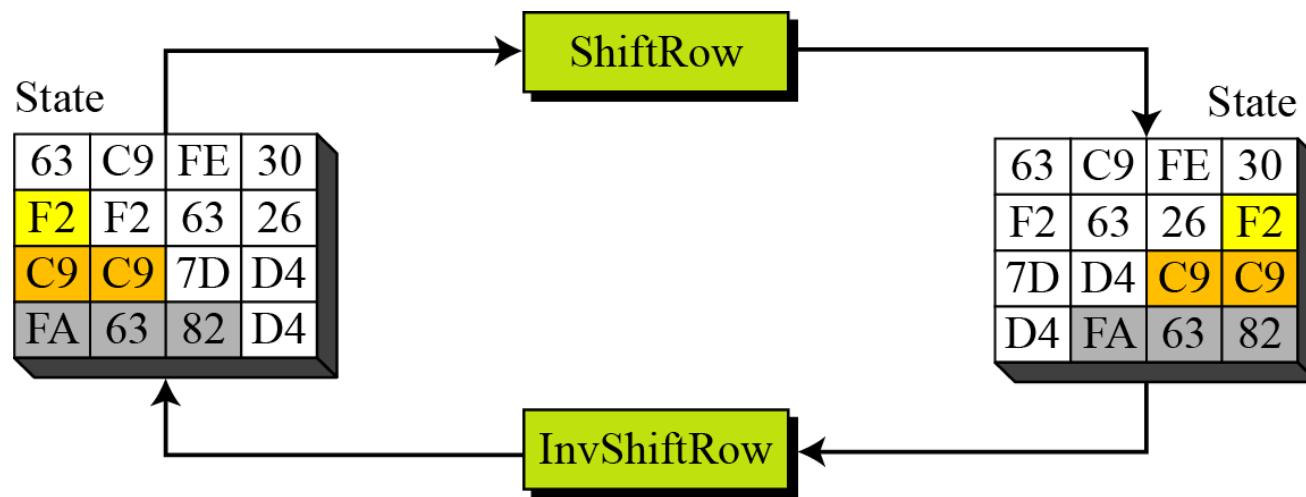
- ▶ Shifting, which permutes the bytes.
- ▶ A circular byte shift in each row
 - ▶ 1st row is unchanged
 - ▶ 2nd row does 1 byte circular shift to left
 - ▶ 3rd row does 2 byte circular shift to left
 - ▶ 4th row does 3 byte circular shift to left
- ▶ In the encryption, the transformation is called ShiftRows
- ▶ In the decryption, the transformation is called InvShiftRows and the shifting is to the right



ShiftRows Scheme



ShiftRows and InvShiftRows



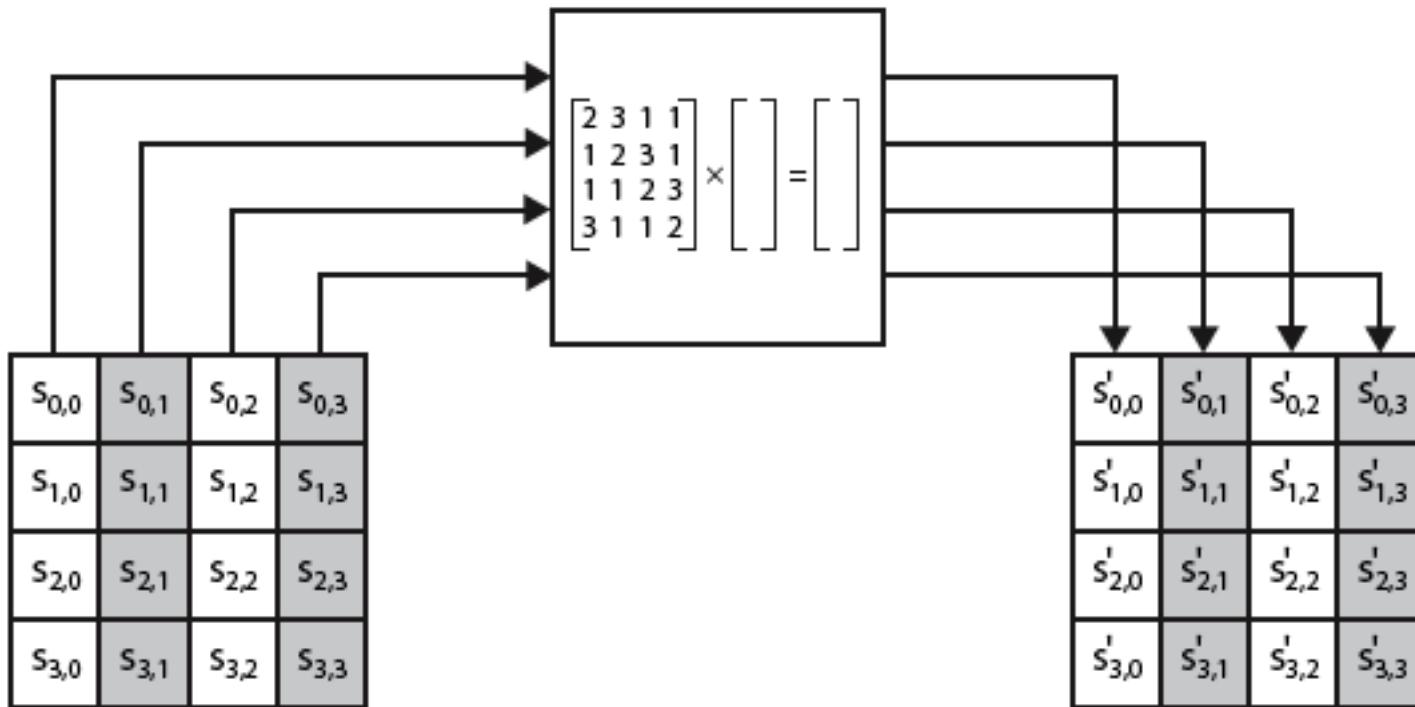
MixColumns

- ▶ ShiftRows and MixColumns provide diffusion to the cipher
- ▶ Each column is processed separately
- ▶ Each byte is replaced by a value dependent on all 4 bytes in the column
- ▶ Effectively a matrix multiplication in $\text{GF}(2^8)$

$$\begin{array}{l} ax + by + cz + dt \\ ex + fy + gz + ht \\ ix + jy + kz + lt \\ mx + ny + oz + pt \end{array} \xrightarrow{\left[\begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array} \right]} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix}$$

New matrix **Constant matrix** Old matrix

MixClumns Scheme



The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

AddRoundKey

- ▶ XOR state with 128-bits of the round key
- ▶ AddRoundKey proceeds one column at a time.
 - ▶ adds a round key word with each state column matrix
 - ▶ the operation is matrix addition
- ▶ Inverse for decryption identical
 - ▶ since XOR own inverse, with reversed keys
- ▶ Designed to be as simple as possible

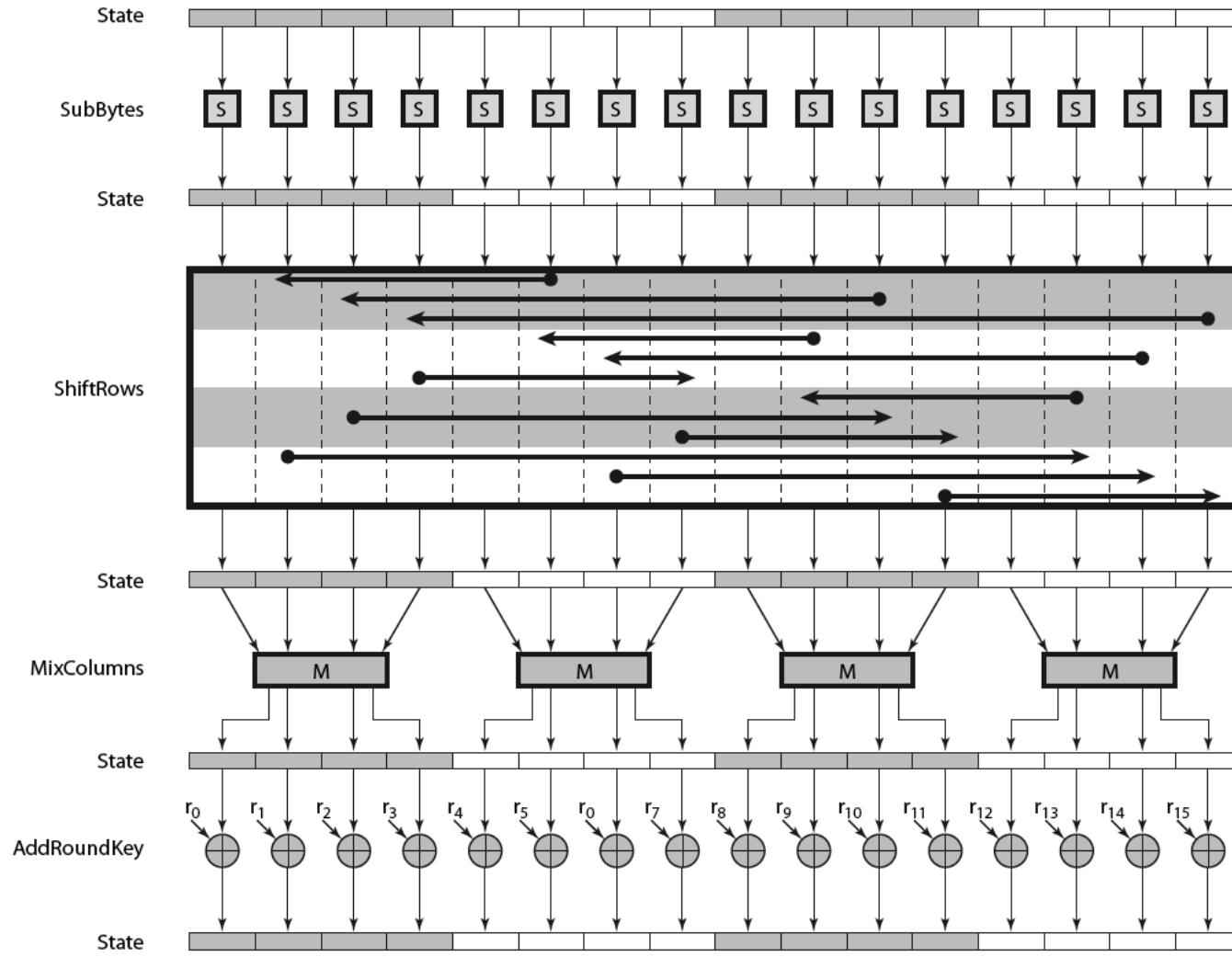


AddRoundKey Scheme

$$\begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline w_i & w_{i+1} & w_{i+2} & w_{i+3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ \hline s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ \hline s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ \hline s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \\ \hline \end{array}$$



AES Round



Topics

- ▶ Origin of AES
- ▶ Basic AES
- ▶ Inside Algorithm
- ▶ Final Notes



AES Security

- ▶ AES was designed after DES.
- ▶ Most of the known attacks on DES were already tested on AES.
- ▶ Brute-Force Attack
 - ▶ AES is definitely more secure than DES due to the larger-size key.



Implementation Aspects

- ▶ The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.
- ▶ Very efficient
- ▶ Implementation was a key factor in its selection as the AES cipher
- ▶ AES animation:
 - ▶ http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf





Modes of Operation



Topics

- ▶ **Overview of Modes of Operation**
- ▶ ECB, CBC, CFB, OFB, CTR
- ▶ Notes and Remarks on each modes



Modes of Operation

- ▶ Block ciphers encrypt fixed size blocks
 - ▶ eg. DES encrypts 64-bit blocks, with 56-bit key
- ▶ Need way to use in practise, given usually have arbitrary amount of information to encrypt
 - ▶ Partition message into separate block for ciphering
- ▶
- ▶ A **mode of operation** describes the process of encrypting each of these blocks **under a single key**
- ▶ Some modes may use randomized addition input value



Quick History

1981

- ▶ Early modes of operation: **ECB, CBC, CFB, OFB**
 - ▶ DES Modes of operation
<http://www.itl.nist.gov/fipspubs/fip81.htm>

2001

- ▶ Revised and including **CTR** mode and AES
 - ▶ Recommendation for Block Cipher Modes of Operation
<http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>

2010

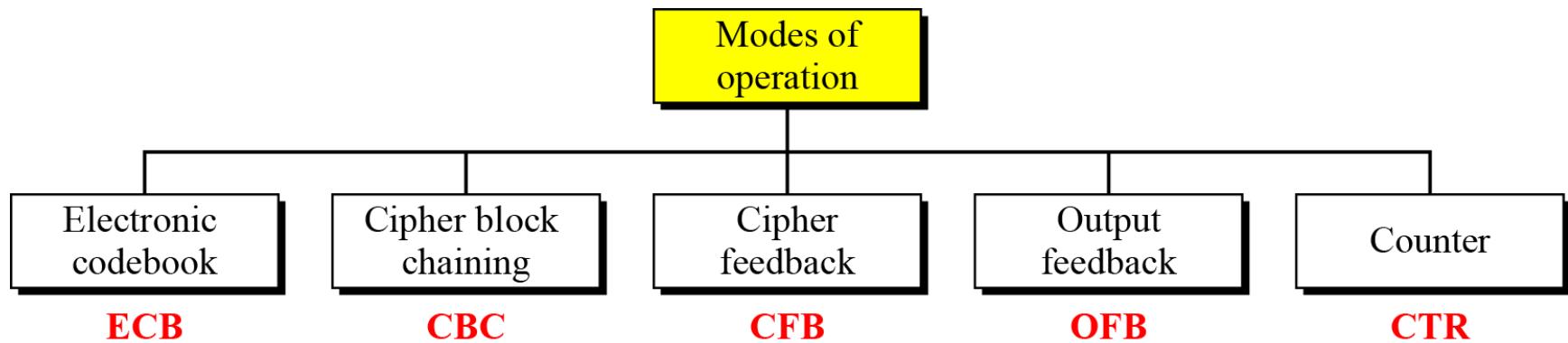
- ▶ New Mode : **XTS-AES**
 - ▶ Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices
<http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf>

Modes of operation are nowadays defined by a number of national and internationally recognized standards bodies such as ISO, IEEE, ANSI and IETF. The most influential source is the US NIST



Modes of Operation Taxonomy

- ▶ Current well-known modes of operation



Moe Technical Notes

- ▶ Initialize Vector (IV)
 - ▶ a block of bits to randomize the encryption and hence to produce distinct ciphertext
- ▶ Nonce : Number (used) Once
 - ▶ Random or pseudorandom number to ensure that past communications can not be reused in replay attacks
 - ▶ Some also refer to initialize vector as nonce
- ▶ Padding
 - ▶ final block may require a padding to fit a block size
 - ▶ Method
 - ▶ Add null Bytes
 - ▶ Add 0x80 and many 0x00
 - ▶ Add the n bytes with value n



Electronic Codebook Book (ECB)

- ▶ Message is broken into independent blocks which are encrypted
- ▶ Each block is a value which is substituted, like a codebook, hence name
- ▶ Each block is encoded independently of the other blocks
$$C_i = E_K (P_i)$$
- ▶ Uses: secure transmission of single values



Topics

- ▶ Overview of Modes of Operation
- ▶ **EBC, CBC, CFB, OFB, CTR**
- ▶ Notes and Remarks on each modes



ECB Scheme

Encryption: $C_i = E_K(P_i)$

Decryption: $P_i = D_K(C_i)$

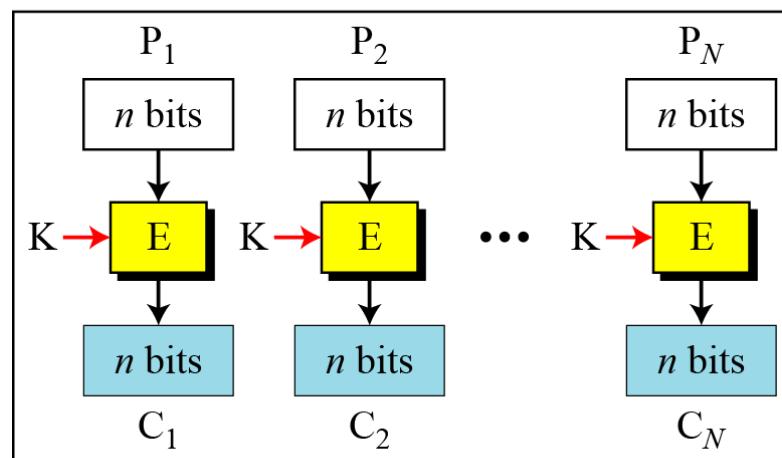
E: Encryption

P_i : Plaintext block i

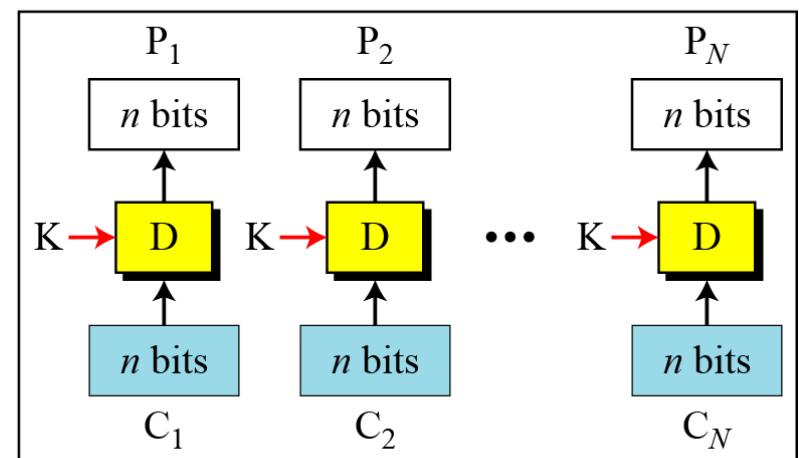
K: Secret key

D: Decryption

C_i : Ciphertext block i



Encryption



Decryption

Remarks on ECB

- ▶ Strength: it's simple.
- ▶ Weakness:
 - ▶ Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.
 - ▶ If the same message is encrypted (with the same key) and sent twice, their ciphertext are the same.
- ▶ Typical application:
 - ▶ secure transmission of short pieces of information (e.g. a temporary encryption key)

Cipher Block Chaining (CBC)

- ▶ Solve security deficiencies in ECB
 - ▶ Repeated same plaintext block result different ciphertext block
- ▶ Each previous cipher blocks is chained to be input with current plaintext block, hence name
- ▶ Use Initial Vector (IV) to start process
$$C_i = E_K (P_i \text{ XOR } C_{i-1})$$
$$C_0 = IV$$
- ▶ Uses: bulk data encryption, authentication



CBC scheme

E: Encryption

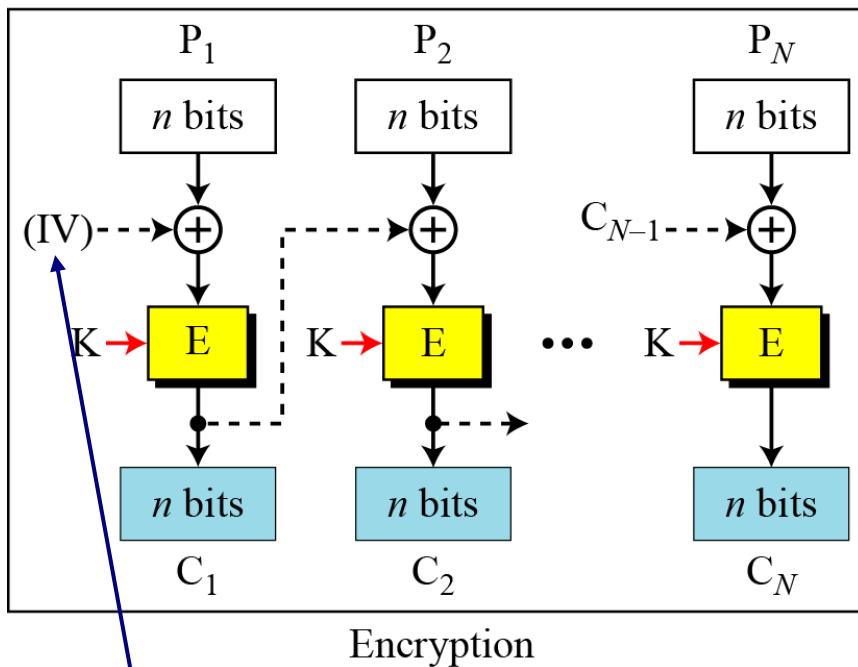
P_i : Plaintext block i

K: Secret key

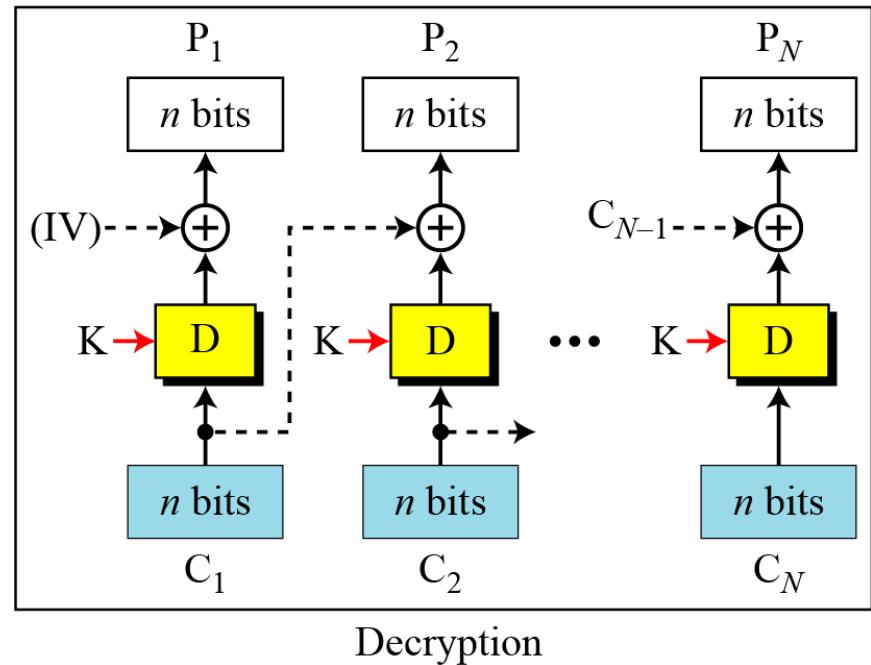
D : Decryption

C_i : Ciphertext block i

IV: Initial vector (C_0)



Encryption



Decryption

Encryption:

$$C_0 = \text{IV}$$

$$C_i = E_K(P_i \oplus C_{i-1})$$

Decryption:

$$C_0 = \text{IV}$$

$$P_i = D_K(C_i) \oplus C_{i-1}$$



Remarks on CBC

- ▶ The encryption of a block depends on the current and **all** blocks before it.
- ▶ So, repeated plaintext blocks are encrypted differently.
- ▶ Initialization Vector (IV)
 - ▶ May sent encrypted in ECB mode before the rest of ciphertext

Cipher FeedBack (CFB)

- ▶ Use Initial Vector to start process
- ▶
- ▶ Encrypt previous ciphertext , then combined with the plaintext block using X-OR to produce the current ciphertext
- ▶ Cipher is fed back (hence name) to concatenate with the rest of IV
- ▶ Plaintext is treated as a stream of bits
 - ▶ Any number of bit (1, 8 or 64 or whatever) to be feed back (denoted CFB-1, CFB-8, CFB-64)
- ▶ Relation between plaintext and ciphertext
$$C_i = P_i \text{ XOR } \text{SelectLeft}(E_K(\text{ShiftLeft}(C_{i-1})))$$
$$C_0 = \text{IV}$$
- ▶ Uses: stream data encryption, authentication



CFB Scheme

Encryption: $C_i = P_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1}]\}$

Decryption: $P_i = C_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1}]\}$

E : Encryption

P_i : Plaintext block i

K: Secret key

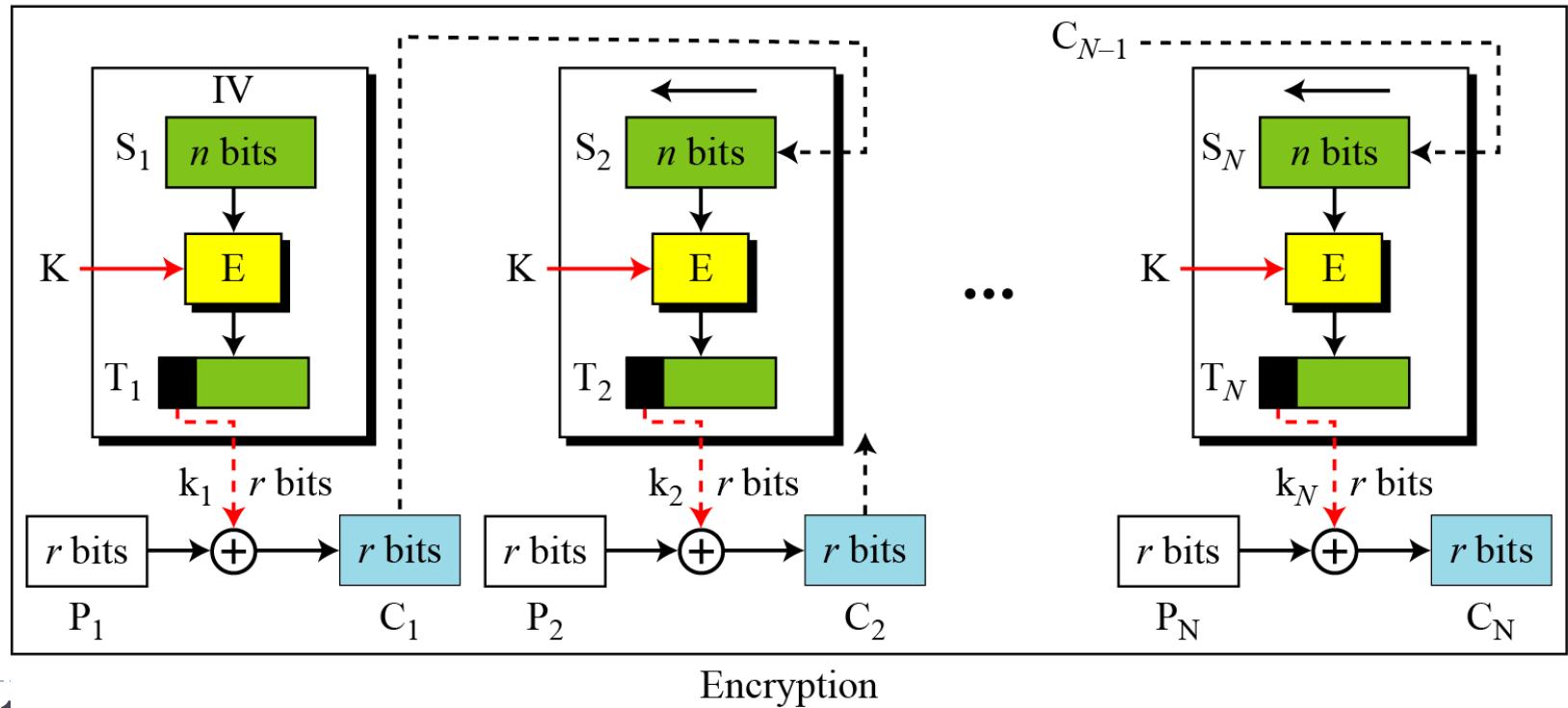
D : Decryption

C_i : Ciphertext block i

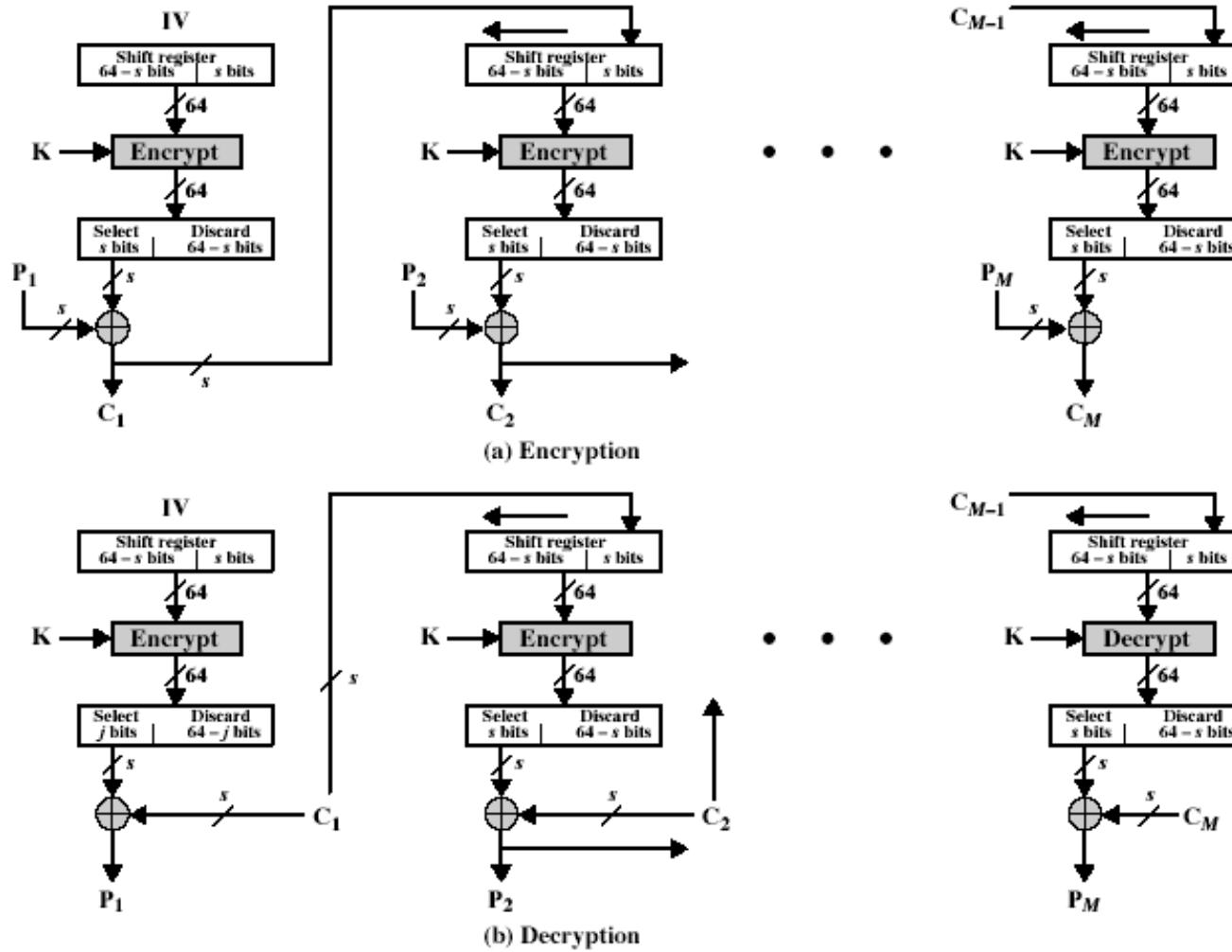
IV: Initial vector (S_1)

S_i : Shift register

T_i : Temporary register

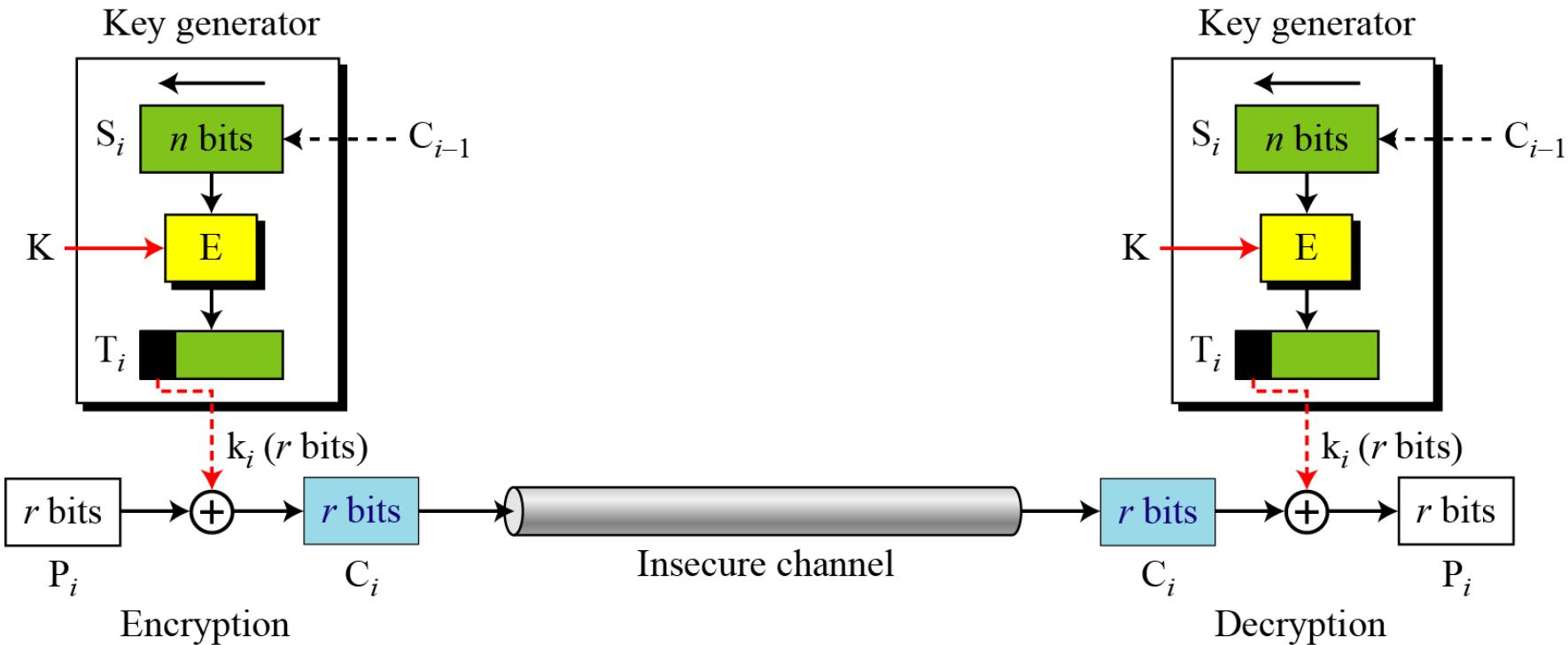


CFB Encryption/Decryption



CFB as a Stream Cipher

- In CFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.



Remark on CFB

- ▶ The block cipher is used as a stream cipher.
 - enable to encrypt any number of bits e.g. single bits or single characters (bytes)
 - S=1 : bit stream cipher
 - S=8 : character stream cipher)
- ▶ A ciphertext segment depends on the current and all preceding plaintext segments.
- ▶ A corrupted ciphertext segment during transmission will affect the current and next several plaintext segments.

Output FeedBack (OFB)

- ▶ Very similar to CFB
- ▶ But output of the encryption function output of cipher is fed back (hence name), instead of ciphertext
- ▶ Feedback is independent of message
- ▶ Relation between plaintext and ciphertext

$$C_i = P_i \text{ XOR } O_i$$

$$O_i = E_K(O_{i-1})$$

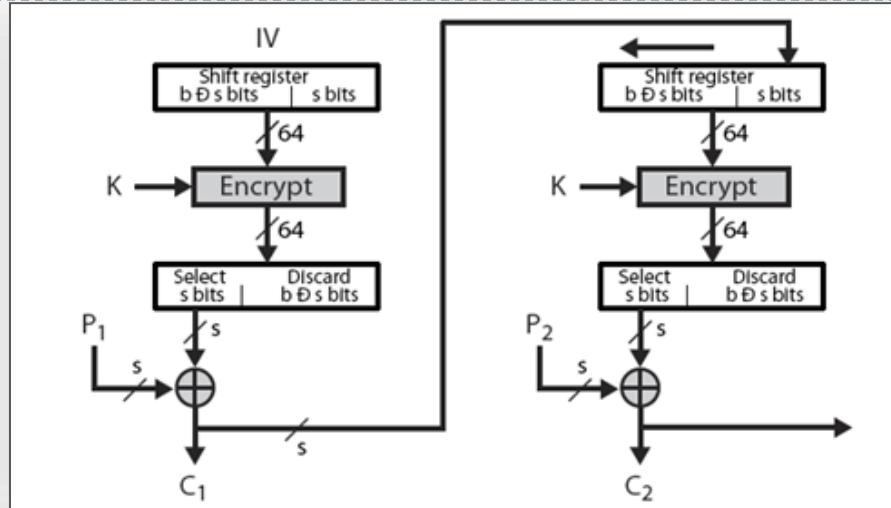
$$O_0 = IV$$

- ▶ Uses: stream encryption over noisy channels

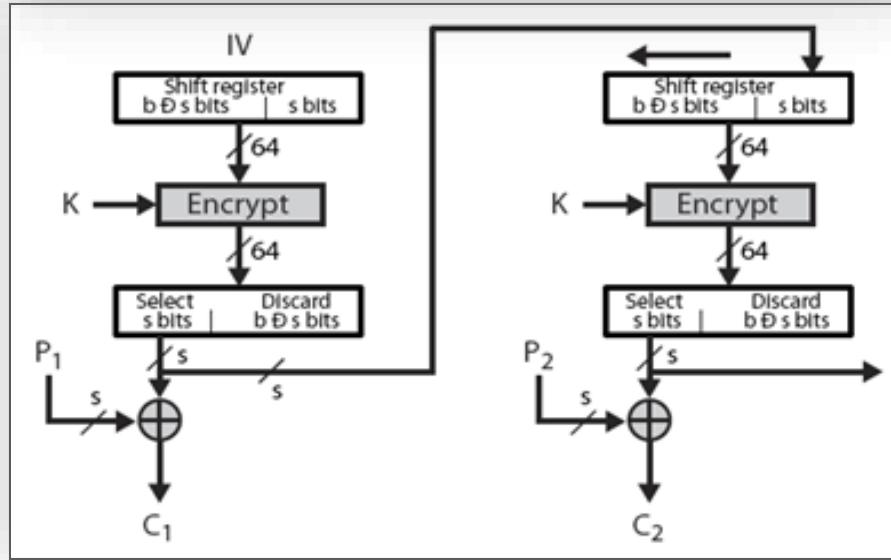


CFB V.S. OFB

Cipher Feedback



Output Feedback



OFB Scheme

E : Encryption

P_i : Plaintext block i

K: Secret key

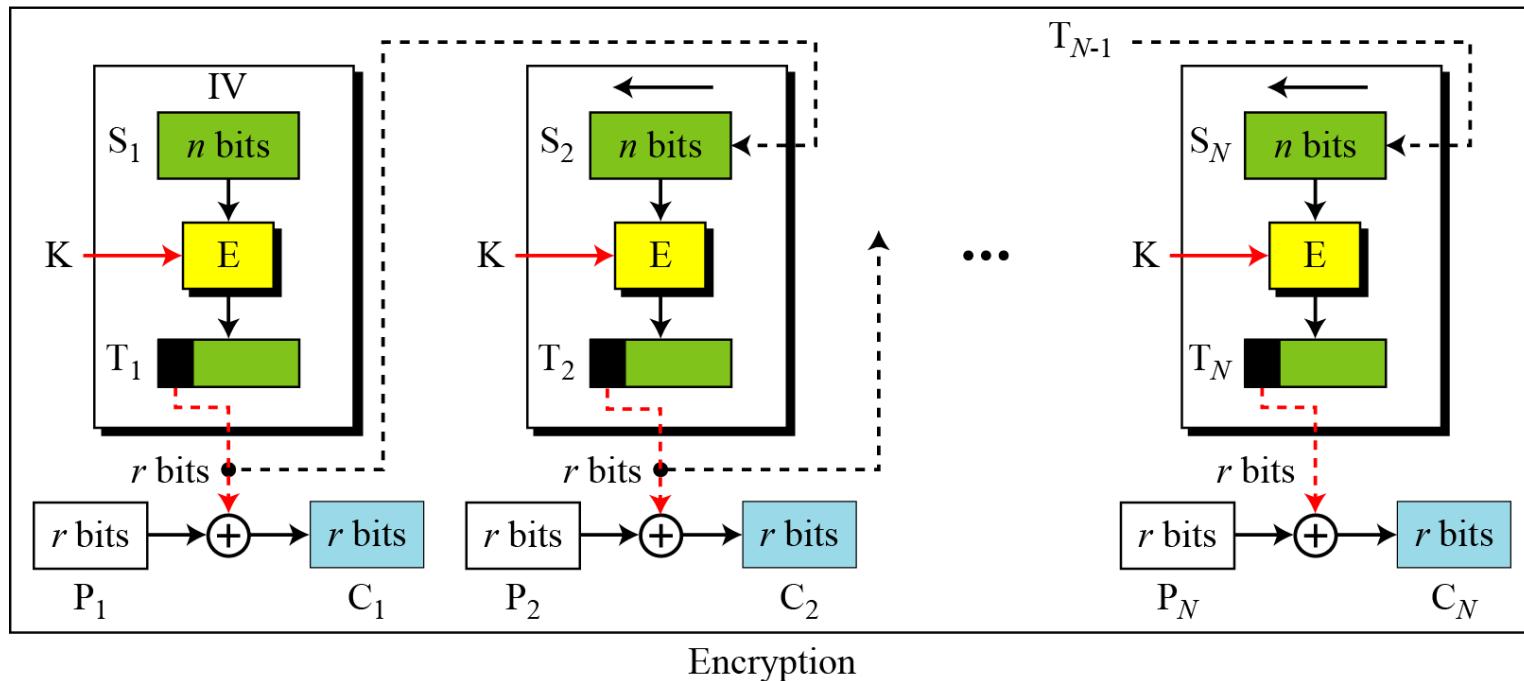
D : Decryption

C_i : Ciphertext block i

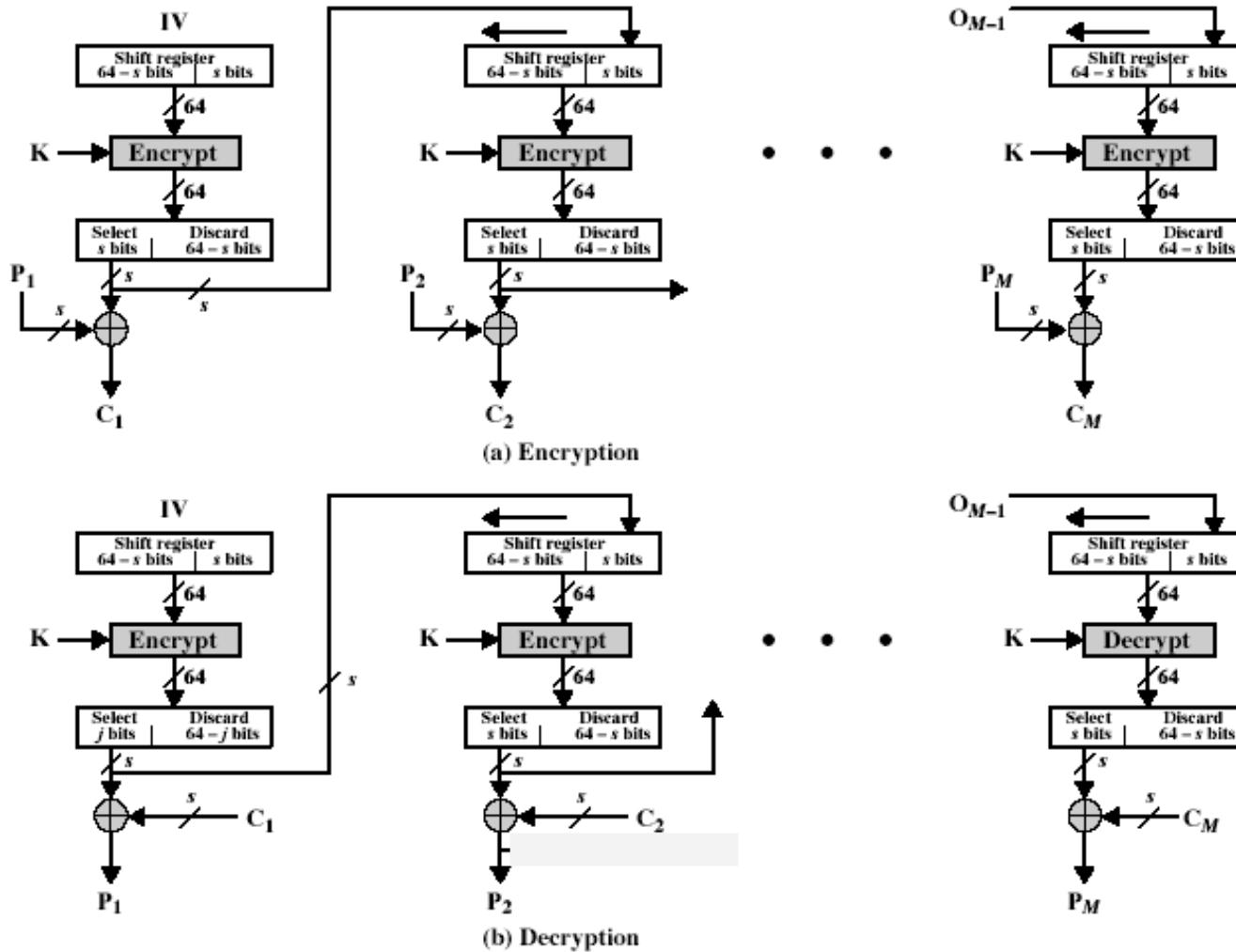
IV: Initial vector (S_1)

S_i : Shift register

T_i : Temporary register

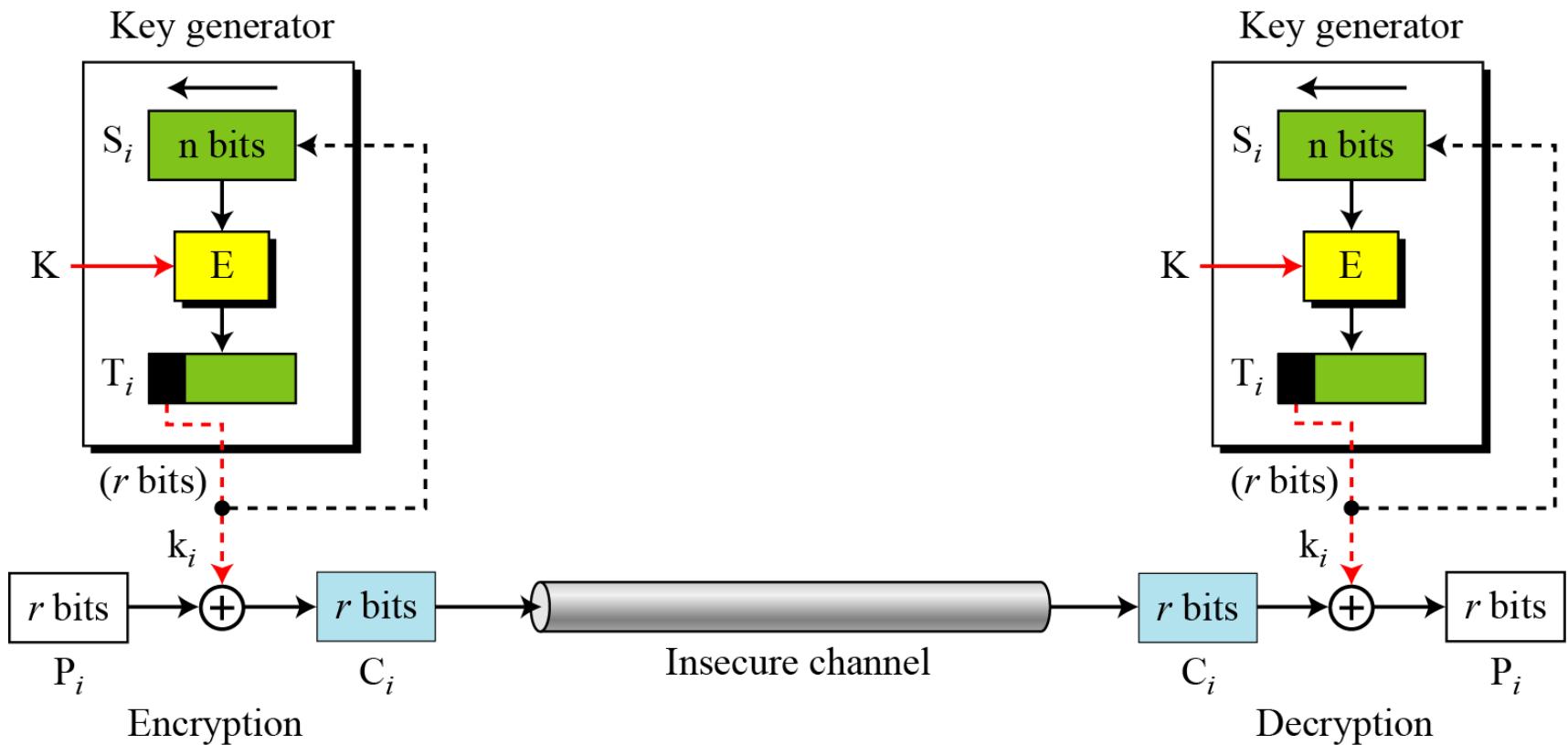


OFB Encryption and Decryption



OFB as a Stream Cipher

- In OFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.



Remarks on OFB

- ▶ Each bit in the ciphertext is independent of the previous bit or bits. This avoids error propagation
- ▶ Pre-compute of forward cipher is possible
- ▶ Security issue
 - ▶ when j^{th} plaintext is known, the j^{th} output of the forward cipher function will be known
 - ▶ Easily cover j^{th} plaintext block of other message with the same IV
- ▶ Require that the IV is a nonce



Counter (CTR)

- ▶ Encrypts counter value with the key rather than any feedback value (no feedback)
- ▶ Counter for each plaintext will be different
 - ▶ can be any function which produces a sequence which is guaranteed not to repeat for a long time
- ▶ Relation

$$\begin{aligned} C_i &= P_i \text{ XOR } O_i \\ O_i &= E_K(i) \end{aligned}$$

- ▶ Uses: high-speed network encryptions



CTR Scheme

E : Encryption

P_i : Plaintext block i

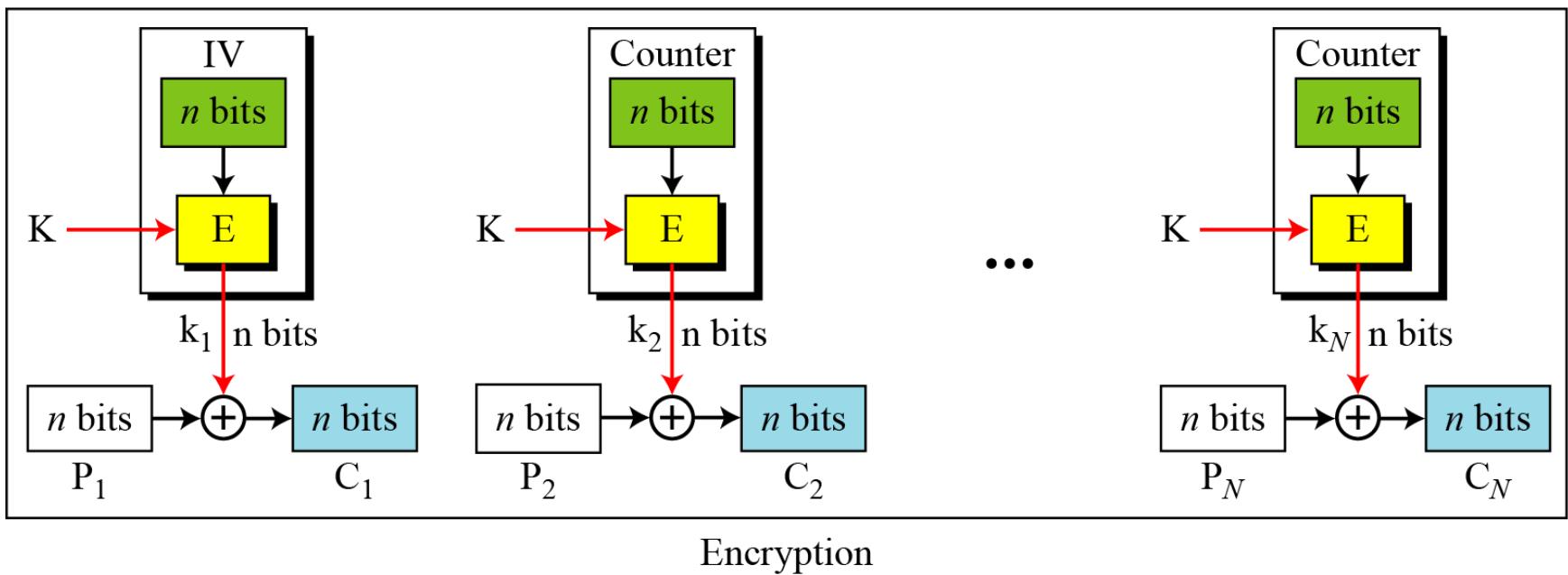
K : Secret key

IV: Initialization vector

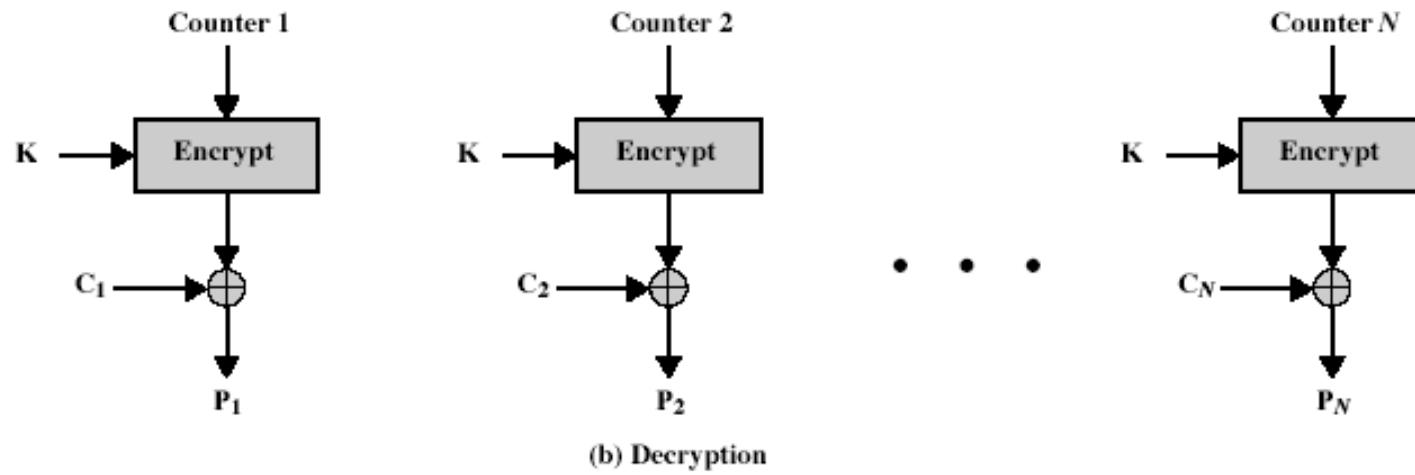
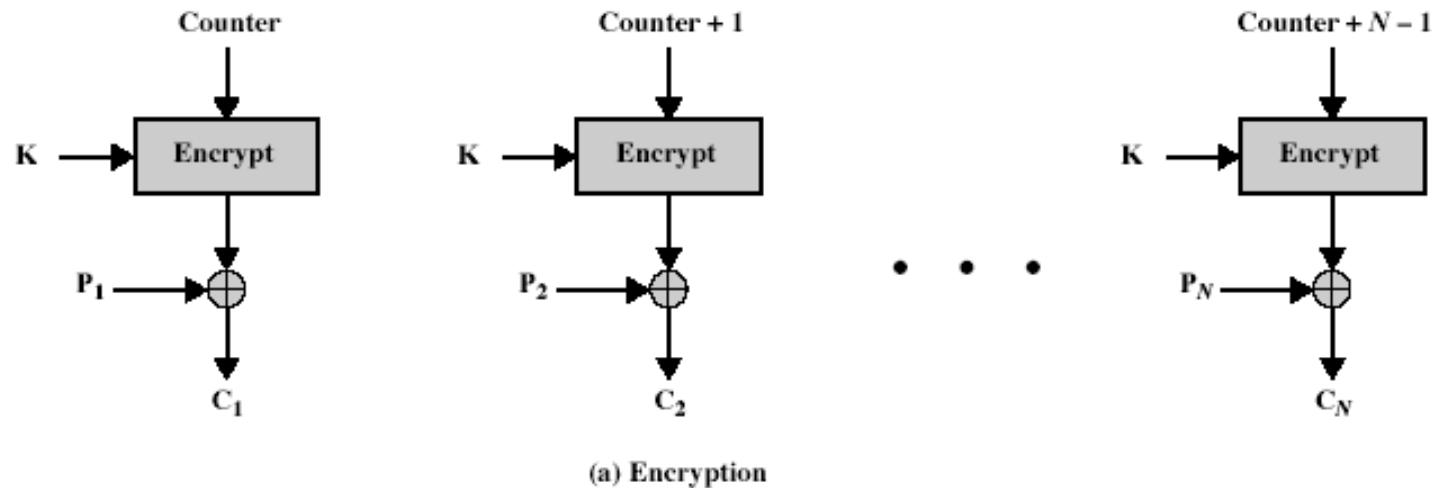
C_i : Ciphertext block i

k_i : Encryption key i

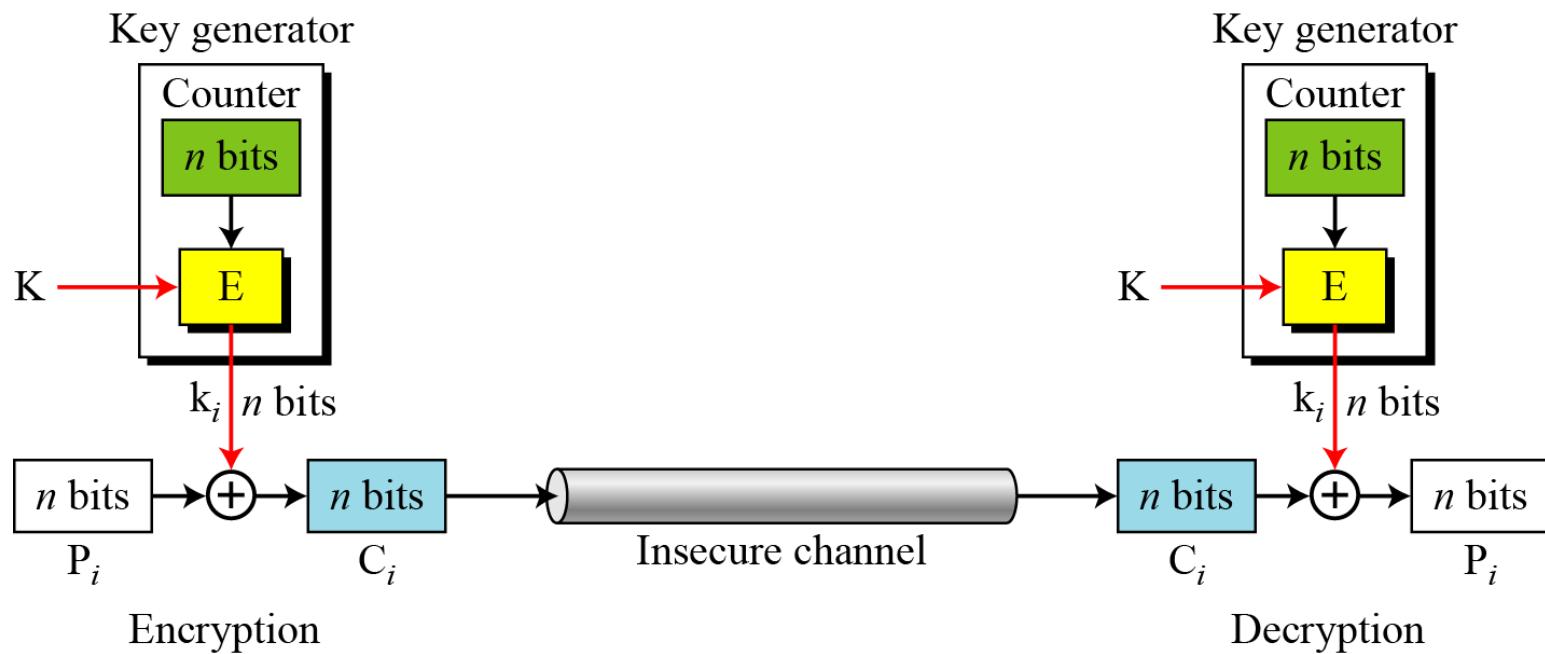
The counter is incremented for each block.



CTR Encryption and Decryption



OFB as a Stream Cipher



Remark on CTR

- ▶ **Strengthes:**
 - ▶ Needs only the encryption algorithm
 - ▶ Random access to encrypted data blocks
 - ▶ blocks can be processed (encrypted or decrypted) in parallel
 - ▶ Simple; fast encryption/decryption
- ▶ **Counter must be**
 - ▶ Must be unknown and unpredictable
 - ▶ pseudo-randomness in the key stream is a goal

Topics

- ▶ Overview of Modes of Operation
- ▶ EBC, CBC, CFB, OFB, CTR
- ▶ **Notes and Remarks on each modes**



Remark on each mode

- ▶ Basically two types:
 - ▶ block cipher
 - ▶ stream cipher
- ▶ CBC is an excellent block cipher
- ▶ CFB, OFB, and CTR are stream ciphers
- ▶ CTR is faster because simpler and it allows parallel processing

Modes and IV

- ▶ An IV has different security requirements than a key
- ▶ Generally, an IV will not be reused under the same key
- ▶ CBC and CFB
 - ▶ reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages
- ▶ OFB and CTR
 - ▶ reusing an IV completely destroys security



CBC and CTR comparison

CBC	CTR
Padding needed	No padding
No parallel processing	Parallel processing
Separate encryption and decryption functions	Encryption function alone is enough
Random IV or a nonce	Unique nonce
Nonce reuse leaks some information about initial plaintext block	Nonce reuse will leak information about the entire message

Comparison of Different Modes

<i>Operation Mode</i>	<i>Description</i>	<i>Type of Result</i>	<i>Data Unit Size</i>
ECB	Each n -bit block is encrypted independently with the same cipher key.	Block cipher	n
CBC	Same as ECB, but each block is first exclusive-ored with the previous ciphertext.	Block cipher	n
CFB	Each r -bit block is exclusive-ored with an r -bit key, which is part of previous cipher text	Stream cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous r -bit key.	Stream cipher	$r \leq n$
CTR	Same as OFB, but a counter is used instead of a shift register.	Stream cipher	n

Comparison of Modes

Mode	Description	Application
ECB	64-bit plaintext block encoded separately	Secure transmission of encryption key
CBC	64-bit plaintext blocks are XORed with preceding 64-bit ciphertext	Commonly used method. Used for authentication
CFB	s bits are processed at a time and used similar to CBC	Primary stream cipher. Used for authentication

Comparison of Modes

Mode	Description	Application
OFB	Similar to CFB except that the output is fed back	Stream cipher well suited for transmission over noisy channels
CTR	Key calculated using the nonce and the counter value. Counter is incremented for each block	General purpose block oriented transmission. Used for high-speed communications

Final Notes

- ▶ ECB, CBC, OFB, CFB, CTR, and XTS modes only provide confidentiality
- ▶ To ensure an encrypted message is not accidentally modified or maliciously tampered requires a separate Message Authentication Code (MAC)
- ▶ Several MAC schemes
 - ▶ HMAC, CMAC and GMAC
- ▶ But.. compositing a confidentiality mode with an authenticity mode could be difficult and error prone
- ▶ New modes combined confidentiality and data integrity into a single cryptographic primitive
 - ▶ CCM, GCM, CWC, EAX, IAPM and OCB

Q&A



Module-III

PUBLIC KEY CRYPTOGRAPHY

PUBLIC KEY CRYPTOGRAPHY

MATHEMATICS
OF ASYMMETRIC
KEY
CRYPTOGRAPHY

- Primes, Primality Testing, Factorization, Euclid's Algorithm, Euler's totient function, Fermat's Theorem, Chinese Remainder Theorem, Exponentiation and logarithm

ASYMMETRIC
KEY CIPHERS

- RSA cryptosystem
- Key distribution & Key management
- Diffie Hellman key exchange
- ElGamal cryptosystem
- Elliptic curve cryptography

Chapter 2

Objectives

- To review integer arithmetic, concentrating on divisibility and finding the greatest common divisor using the Euclidean algorithm
- To understand how the extended Euclidean algorithm can be used to solve linear congruent equations, and to find the multiplicative inverses
- To emphasize the importance of modular arithmetic and the modulo operator, because they are extensively used in cryptography
- To emphasize and review matrices and operations on residue matrices that are extensively used in cryptography
- To solve a set of congruent equations using residue matrices



Part-I

Mathematics of Cryptography

Modular Arithmetic, Congruence,
and Matrices

2-1 INTEGER ARITHMETIC

In integer arithmetic, we use a set and a few operations. You are familiar with this set and the corresponding operations, but they are reviewed here to create a background for modular arithmetic.

Topics discussed in this section:

- 2.1.1 Set of Integers**
- 2.1.2 Binary Operations**
- 2.1.3 Integer Division**
- 2.1.4 Divisibility**
- 2.1.5 Linear Diophantine Equations**

2.1.1 Set of Integers

The set of integers, denoted by Z, contains all integral numbers (with no fraction) from negative infinity to positive infinity (Figure 2.1).

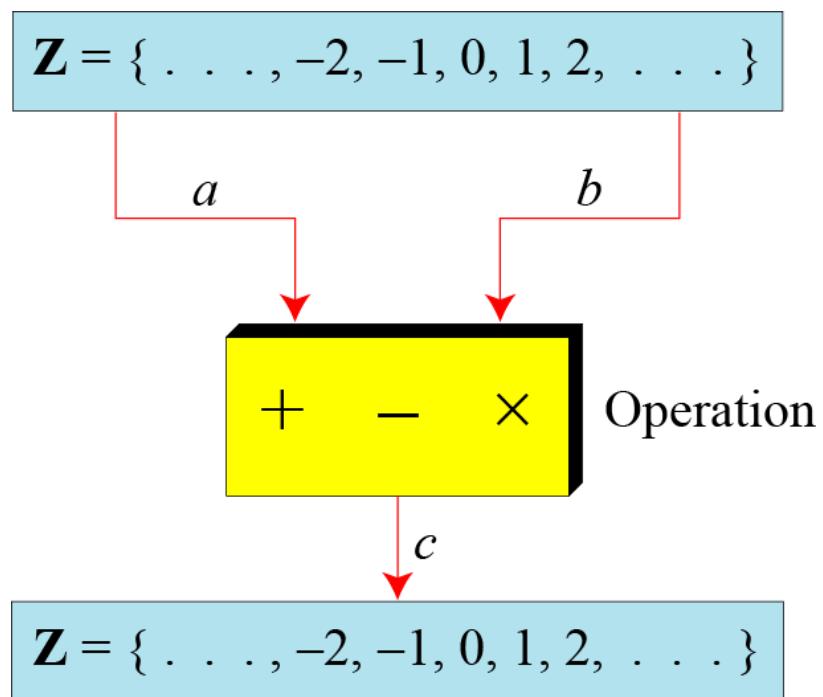
Figure 2.1 The set of integers

$$\mathbf{Z} = \{ \dots, -2, -1, 0, 1, 2, \dots \}$$

2.1.2 Binary Operations

In cryptography, we are interested in three binary operations applied to the set of integers. A binary operation takes two inputs and creates one output.

Figure 2.2 Three binary operations for the set of integers



2.1.2 *Continued*

Example 2.1

The following shows the results of the three binary operations on two integers. Because each input can be either positive or negative, we can have four cases for each operation.

Add:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
Subtract:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
Multiply:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

2.1.3 Integer Division

In integer arithmetic, if we divide a by n , we can get q And r . The relationship between these four integers can be shown as

$$a = q \times n + r$$

2.1.3 Continued

Example 2.2

Assume that $a = 255$ and $n = 11$. We can find $q = 23$ and $R = 2$ using the division algorithm.

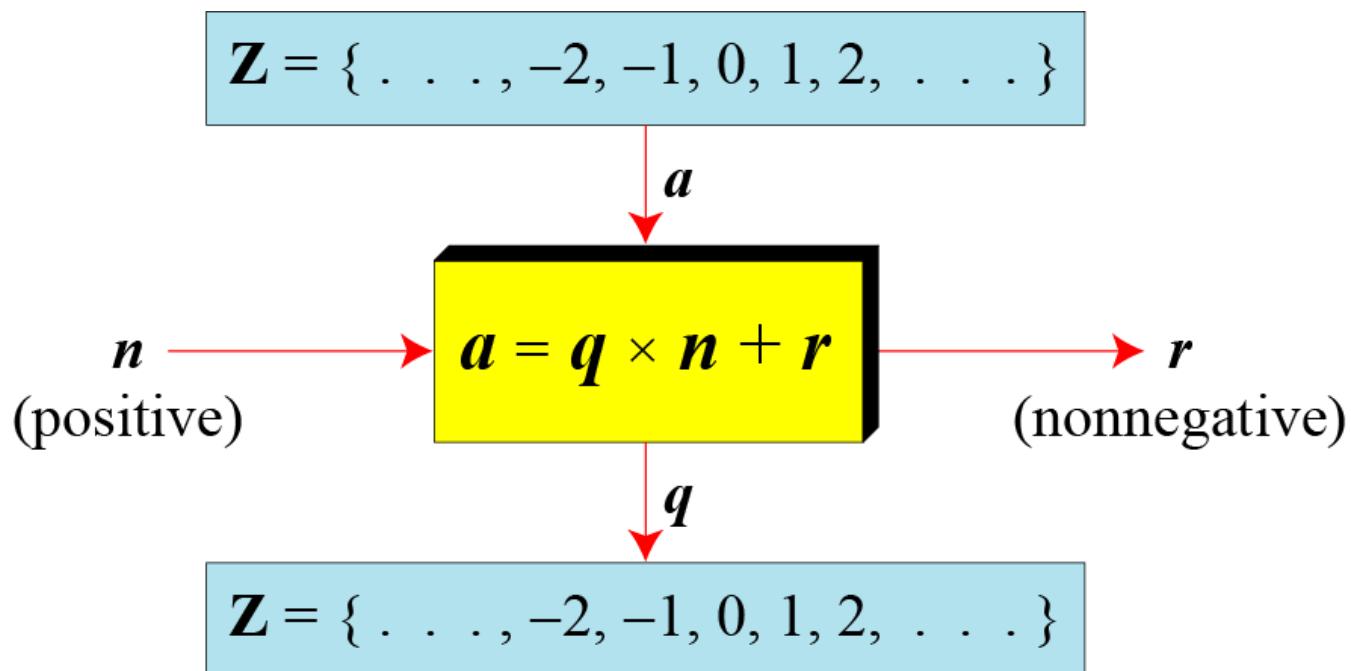
Figure 2.3 Example 2.2, finding the quotient and the remainder

$$\begin{array}{r} 2 \ 3 \quad \longleftarrow q \\ \hline 2 \ 5 \ 5 \quad \longleftarrow a \\ 2 \ 2 \\ \hline 3 \ 5 \\ 3 \ 3 \\ \hline 2 \quad \longleftarrow r \end{array}$$

The diagram shows the division algorithm for $a = 255$ by $n = 11$. The quotient $q = 23$ is written above the division bar, with a red arrow pointing from the variable q to the digit 3. The dividend $a = 255$ is written below the division bar, with a red arrow pointing from the variable a to the digit 5. The remainder $r = 2$ is written at the bottom, with a red arrow pointing from the variable r to the digit 2. The divisor $n = 11$ is written to the left of the division bar, with a red arrow pointing from the variable n to the digit 1.

2.1.3 Continued

Figure 2.4 Division algorithm for integers



2.1.3 *Continued*

Example 2.3

When we use a computer or a calculator, r and q are negative when a is negative. How can we apply the restriction that r needs to be positive? The solution is simple, we decrement the value of q by 1 and we add the value of n to r to make it positive.

$$-255 = (-23 \times 11) + (-2) \quad \leftrightarrow \quad -255 = (-24 \times 11) + 9$$

2.1.4 *Continued*

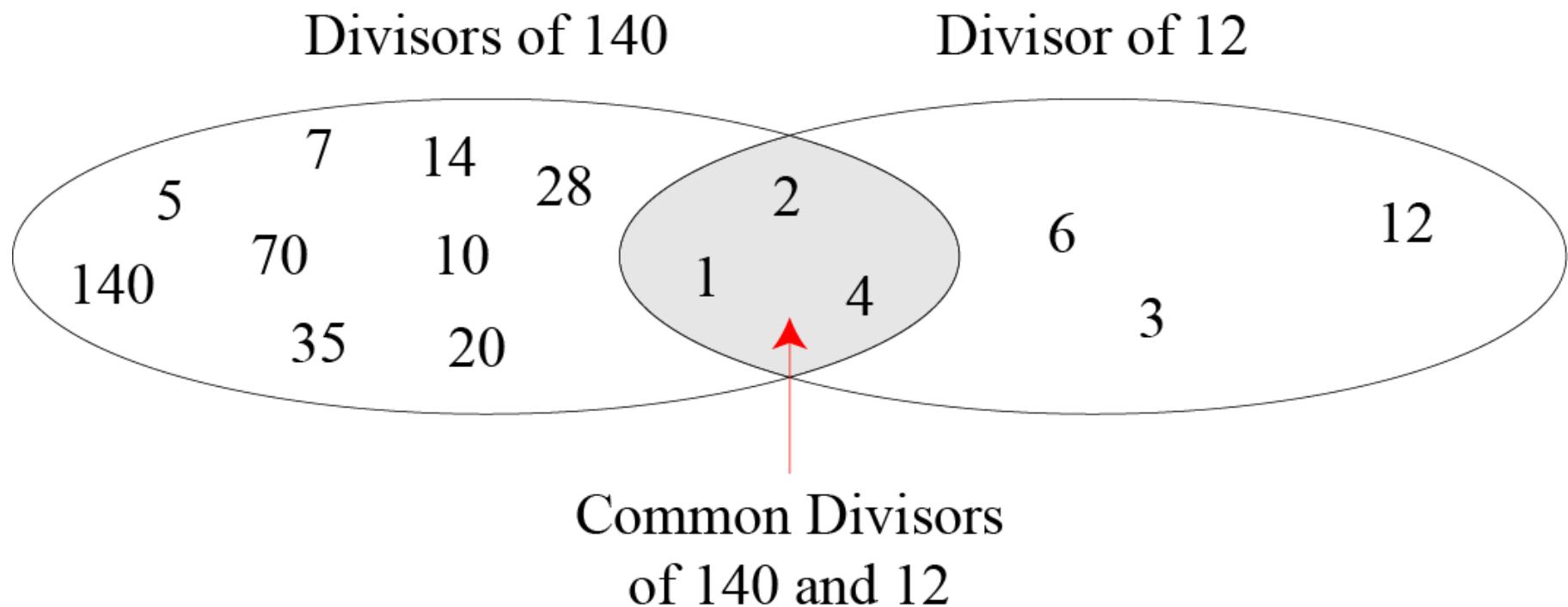
Note

Fact 1: The integer 1 has only one divisor, itself.

Fact 2: Any positive integer has at least two divisors, 1 and itself (but it can have more).

2.1.4 *Continued*

Figure 2.6 Common divisors of two integers



2.1.4 *Continued*

Note

Greatest Common Divisor

The greatest common divisor of two positive integers is the largest integer that can divide both integers.

Note

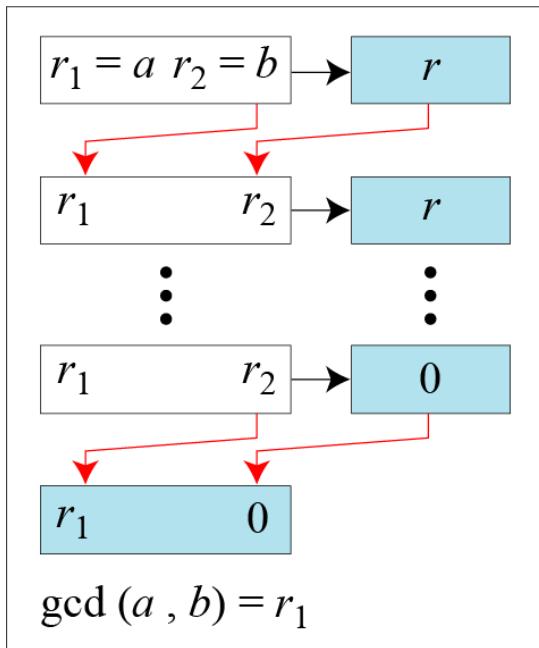
Euclidean Algorithm

Fact 1: $\gcd(a, 0) = a$

Fact 2: $\gcd(a, b) = \gcd(b, r)$, where r is the remainder of dividing a by b

2.1.4 Continued

Figure 2.7 Euclidean Algorithm



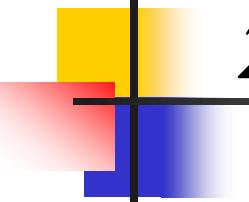
a. Process

```
r1 ← a;      r2 ← b;    (Initialization)  
while (r2 > 0)  
{  
    q ← r1 / r2;  
    r ← r1 - q × r2;  
    r1 ← r2; r2 ← r;  
}  
gcd (a, b) ← r1
```

b. Algorithm

Note

When $\gcd(a, b) = 1$, we say that a and b are relatively prime.



2.1.4 Continued

Note

When $\gcd(a, b) = 1$, we say that a and b are relatively prime.

2.1.4 *Continued*

Example 2.7

Find the greatest common divisor of 2740 and 1760.

Solution

We have $\gcd(2740, 1760) = 20$.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

2.1.4 *Continued*

Example 2.8

Find the greatest common divisor of 25 and 60.

Solution

We have $\gcd(25, 60) = 5$.

q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

2.1.4 *Continued*

Extended Euclidean Algorithm

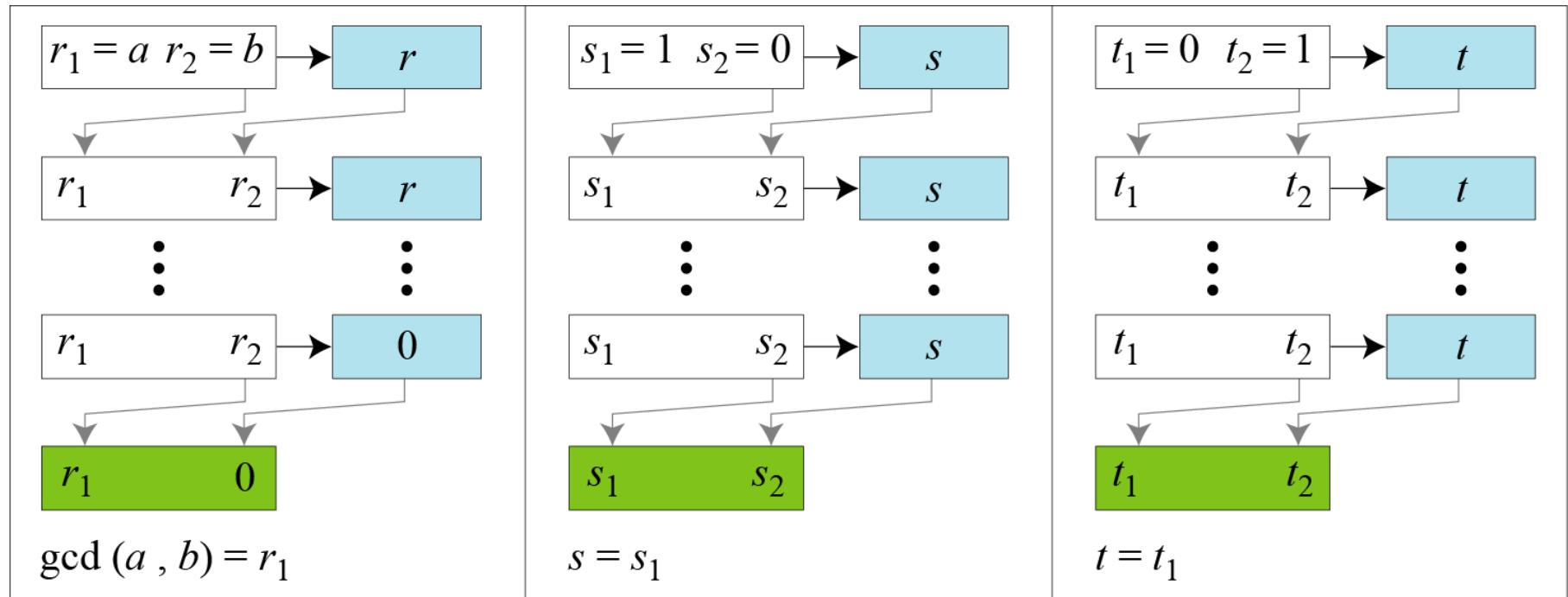
Given two integers a and b , we often need to find other two integers, s and t , such that

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm can calculate the $\gcd(a, b)$ and at the same time calculate the value of s and t .

2.1.4 Continued

Figure 2.8.a Extended Euclidean algorithm, part a



a. Process

2.1.4 Continued

Figure 2.8.b Extended Euclidean algorithm, part b

```
r1 ← a;      r2 ← b;  
s1 ← 1;      s2 ← 0;  
t1 ← 0;      t2 ← 1;
```

(Initialization)

while ($r_2 > 0$)

{

$q \leftarrow r_1 / r_2$;

```
  r ← r1 - q × r2;  
  r1 ← r2; r2 ← r;
```

(Updating r 's)

```
  s ← s1 - q × s2;  
  s1 ← s2; s2 ← s;
```

(Updating s 's)

```
  t ← t1 - q × t2;  
  t1 ← t2; t2 ← t;
```

(Updating t 's)

}

gcd(a, b) ← r₁; s ← s₁; t ← t₁

b. Algorithm

2.1.4 Continued

Example 2.9

Given $a = 161$ and $b = 28$, find $\gcd(a, b)$ and the values of s and t .

Solution

We get $\gcd(161, 28) = 7$, $s = -1$ and $t = 6$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

2.1.4 Continued

Example 2.10

Given $a = 17$ and $b = 0$, find $\gcd(a, b)$ and the values of s and t .

Solution

We get $\gcd(17, 0) = 17$, $s = 1$, and $t = 0$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
17	0			1	0		0	1	

2.1.4 Continued

Example 2.11

Given $a = 0$ and $b = 45$, find $\gcd(a, b)$ and the values of s and t .

Solution

We get $\gcd(0, 45) = 45$, $s = 0$, and $t = 1$.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
0	0	45	0	1	0	1	0	1	0
	45	0		0	1		1	0	

2-2 MODULAR ARITHMETIC

The division relationship ($a = q \times n + r$) discussed in the previous section has two inputs (a and n) and two outputs (q and r). In modular arithmetic, we are interested in only one of the outputs, the remainder r .

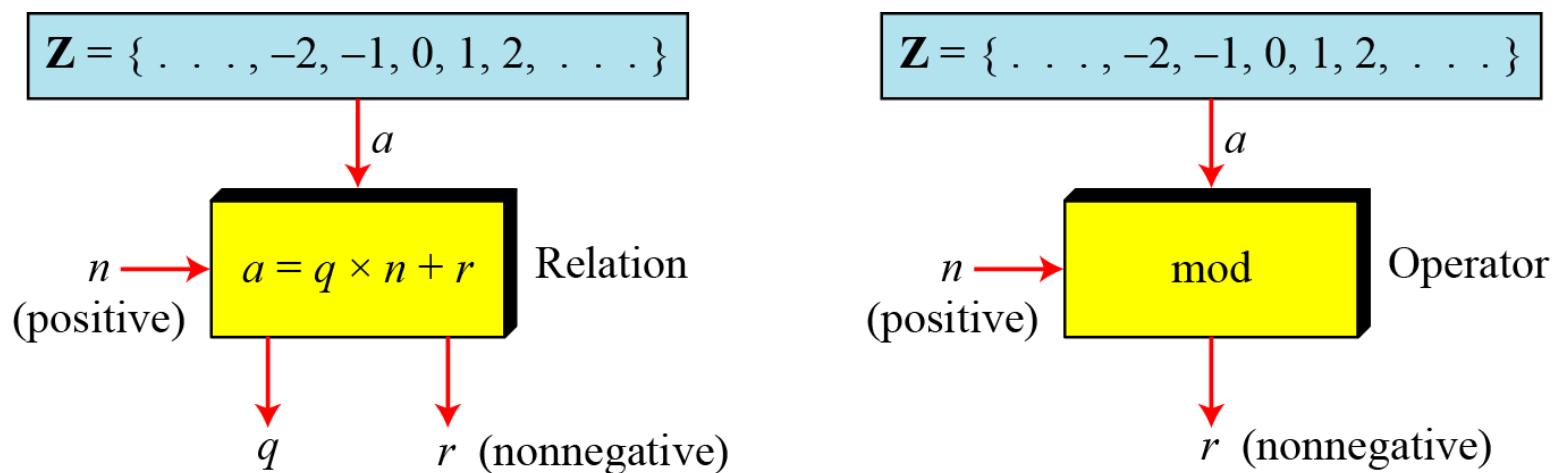
Topics discussed in this section:

- 2.2.1 Modular Operator**
- 2.2.2 Set of Residues**
- 2.2.3 Congruence**
- 2.2.4 Operations in Z_n**
- 2.2.5 Addition and Multiplication Tables**
- 2.2.6 Different Sets**

2.2.1 Modulo Operator

The modulo operator is shown as **mod**. The second input (n) is called the modulus. The output r is called the residue.

Figure 2.9 Division algorithm and modulo operator



2.1.4 *Continued*

Example 2.14

Find the result of the following operations:

- a. $27 \bmod 5$
- b. $36 \bmod 12$
- c. $-18 \bmod 14$
- d. $-7 \bmod 10$

Solution

- a. Dividing 27 by 5 results in $r = 2$
- b. Dividing 36 by 12 results in $r = 0$.
- c. Dividing -18 by 14 results in $r = -4$. After adding the modulus $r = 10$
- d. Dividing -7 by 10 results in $r = -7$. After adding the modulus to -7 , $r = 3$.

2.2.2 Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as **the set of least residues modulo n, or Z_n** .

Figure 2.10 Some Z_n sets

$$Z_n = \{ 0, 1, 2, 3, \dots, (n - 1) \}$$

$$Z_2 = \{ 0, 1 \}$$

$$Z_6 = \{ 0, 1, 2, 3, 4, 5 \}$$

$$Z_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

2.2.3 Congruence

To show that two integers are congruent, we use the congruence operator (\equiv). For example, we write:

$$2 \equiv 12 \pmod{10}$$

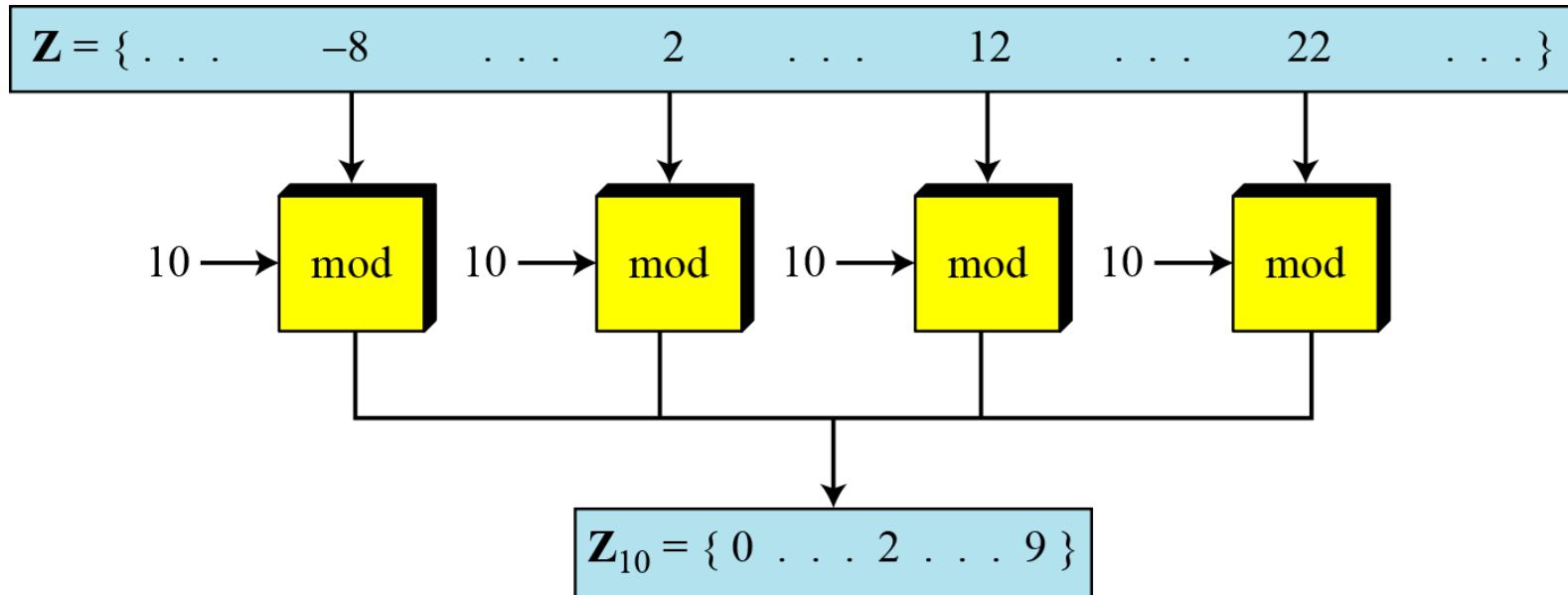
$$3 \equiv 8 \pmod{5}$$

$$13 \equiv 23 \pmod{10}$$

$$8 \equiv 13 \pmod{5}$$

2.2.3 *Continued*

Figure 2.11 *Concept of congruence*



$$-8 \equiv 2 \equiv 12 \equiv 22 \pmod{10}$$

Congruence Relationship

2.2.3 *Continued*

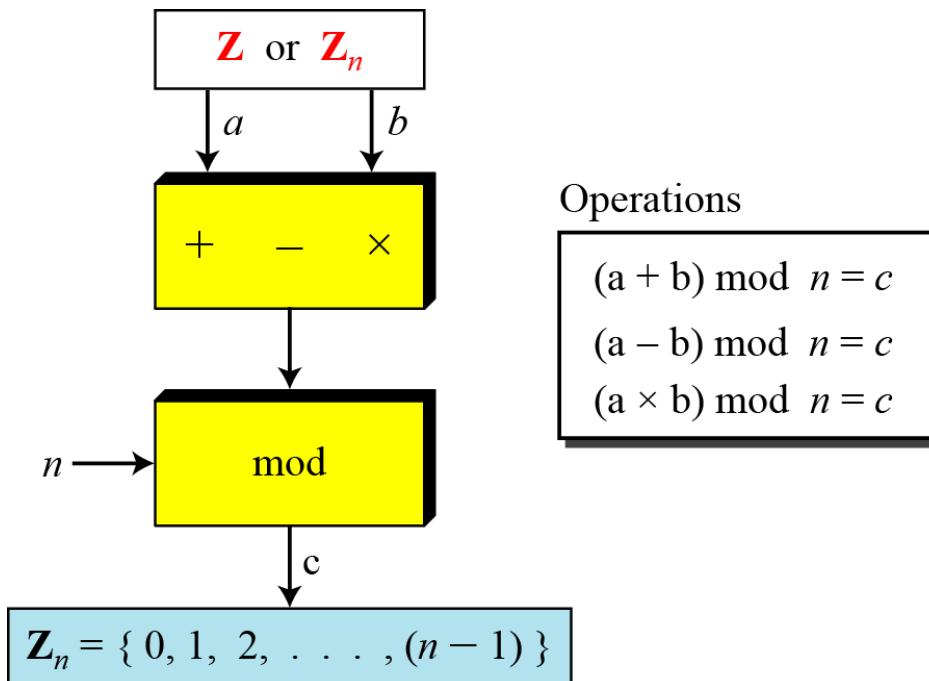
Example 2.15

We use modular arithmetic in our daily life; for example, we use a clock to measure time. Our clock system uses modulo 12 arithmetic. However, instead of a 0 we use the number 12.

2.2.4 Operation in Z_n

The three binary operations that we discussed for the set Z can also be defined for the set Z_n . The result may need to be mapped to Z_n using the mod operator.

Figure 2.13 Binary operations in Z_n



2.2.4 *Continued*

Example 2.16

Perform the following operations (the inputs come from Z_n):

- Add 7 to 14 in Z_{15} .
- Subtract 11 from 7 in Z_{13} .
- Multiply 11 by 7 in Z_{20} .

Solution

$$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$$

$$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$$

$$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$$

2.2.4 *Continued*

Example 2.17

Perform the following operations (the inputs come from either Z or Z_n):

- a. Add 17 to 27 in Z_{14} .
- b. Subtract 43 from 12 in Z_{13} .
- c. Multiply 123 by -10 in Z_{19} .

Solution

2.2.4 *Continued*

Properties

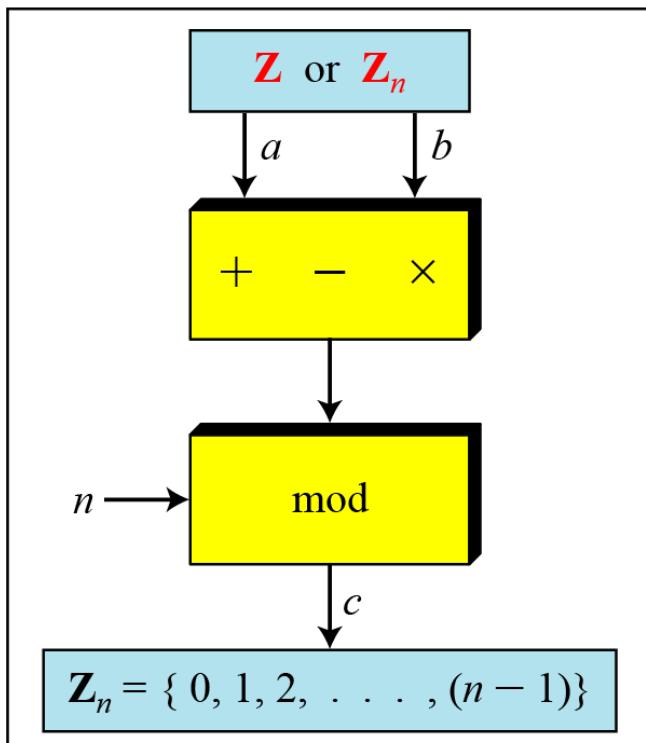
First Property: $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$

Second Property: $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$

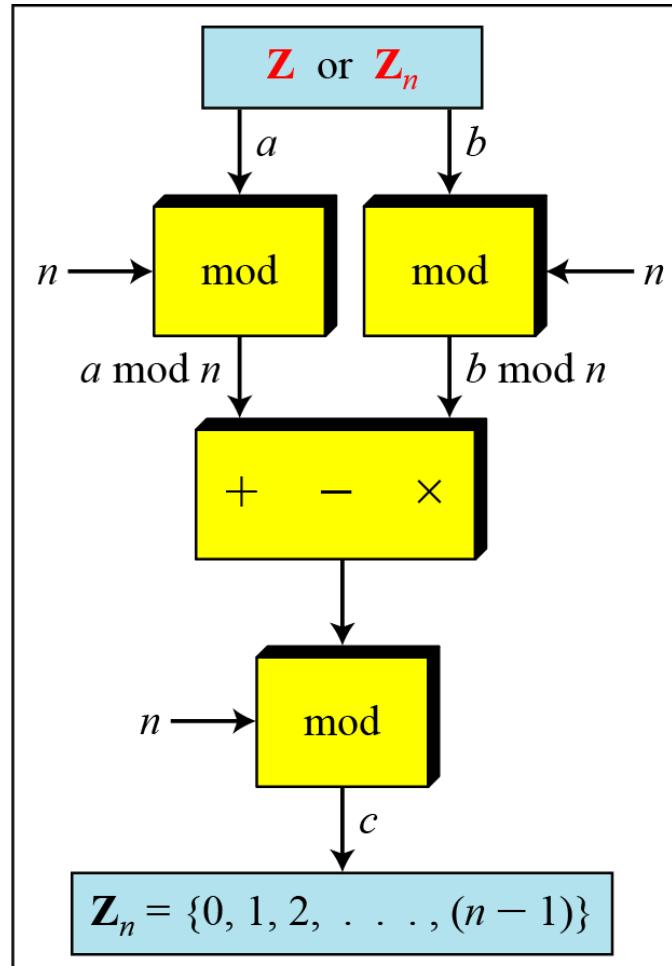
Third Property: $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

2.2.4 Continued

Figure 2.14 Properties of mode operator



a. Original process



b. Applying properties

2.2.4 *Continued*

Example 2.18

The following shows the application of the above properties:

1. $(1,723,345 + 2,124,945) \text{ mod } 11 = (8 + 9) \text{ mod } 11 = 6$
2. $(1,723,345 - 2,124,945) \text{ mod } 16 = (8 - 9) \text{ mod } 11 = 10$
3. $(1,723,345 \times 2,124,945) \text{ mod } 16 = (8 \times 9) \text{ mod } 11 = 6$

2.2.5 Continue

Multiplicative Inverse

In Z_n , two numbers a and b are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

Note

In modular arithmetic, an integer may or may not have a multiplicative inverse. When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo n.

2.2.2 Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as **the set of least residues modulo n, or Z_n** .

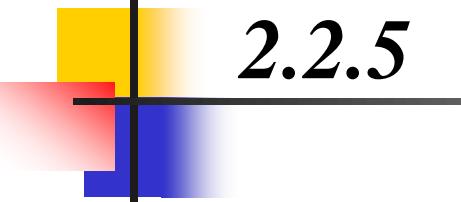
Figure 2.10 Some Z_n sets

$$Z_n = \{ 0, 1, 2, 3, \dots, (n - 1) \}$$

$$Z_2 = \{ 0, 1 \}$$

$$Z_6 = \{ 0, 1, 2, 3, 4, 5 \}$$

$$Z_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$



2.2.5 *Continued*

Example 2.22

Find the multiplicative inverse of 8 in \mathbf{Z}_{10} .

Solution

There is no multiplicative inverse because $\gcd(10, 8) = 2 \neq 1$. In other words, we cannot find any number between 0 and 9 such that when multiplied by 8, the result is congruent to 1.

Example 2.23

Find all multiplicative inverses in \mathbf{Z}_{10} .

Solution

There are only three pairs: $(1, 1)$, $(3, 7)$ and $(9, 9)$. The numbers $0, 2, 4, 5, 6$, and 8 do not have a multiplicative inverse.

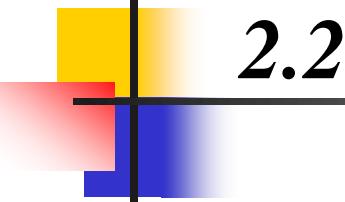
2.2.5 *Continued*

Example 2.24

Find all multiplicative inverse pairs in \mathbf{Z}_{11} .

Solution

We have seven pairs: $(1, 1)$, $(2, 6)$, $(3, 4)$, $(5, 9)$, $(7, 8)$, $(9, 9)$, and $(10, 10)$.



2.2.5 *Continued*

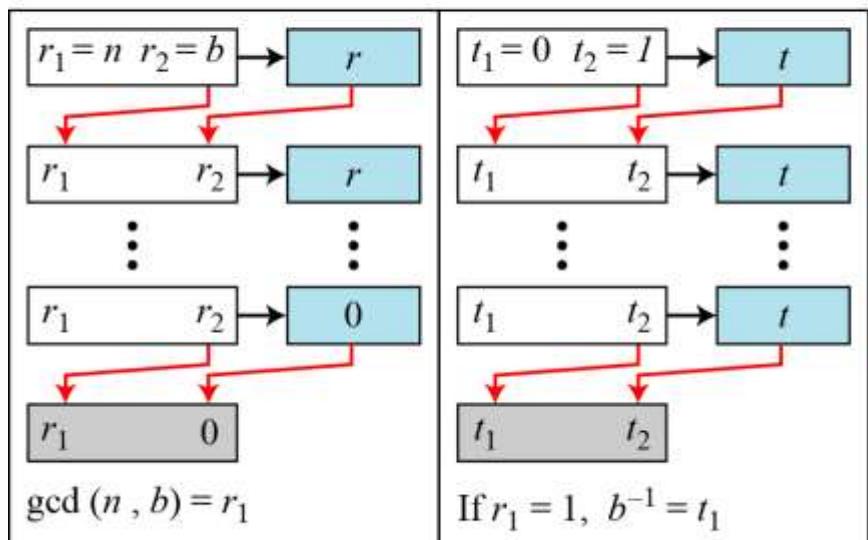
Note

The extended Euclidean algorithm finds the multiplicative inverses of b in Z_n when n and b are given and $\gcd(n, b) = 1$.

The multiplicative inverse of b is the value of t after being mapped to Z_n .

2.2.5 Continued

Figure 2.15 Using extended Euclidean algorithm to find multiplicative inverse



a. Process

```
r1 ← n;      r2 ← b;  
t1 ← 0;      t2 ← 1;  
  
while (r2 > 0)  
{  
    q ← r1 / r2;  
  
    r ← r1 - q × r2;  
    r1 ← r2;      r2 ← r;  
  
    t ← t1 - q × t2;  
    t1 ← t2;      t2 ← t;  
}  
if (r1 = 1) then b-1 ← t1
```

b. Algorithm

2.2.5 *Continued*

Example 2.25

Find the multiplicative inverse of 11 in \mathbf{Z}_{26} .

Find the multiplicative inverse of 11 in \mathbf{Z}_{26} .

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

The gcd (26, 11) is 1; the inverse of 11 is -7 or 19.

2.2.5 *Continued*

Example 2.26

Find the multiplicative inverse of 23 in \mathbb{Z}_{100} .

2.2.5 *Continued*

Example 2.26

Find the multiplicative inverse of 23 in \mathbf{Z}_{100} .

Solution

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	-4
2	23	8	7	1	-4	19
1	8	7	1	-4	9	-13
7	7	1	0	9	-13	100
	1	0		-13	100	

The gcd (100, 23) is 1; the inverse of 23 is -13 or 87.

2.2.5 *Continued*

Example 2.27

Find the inverse of 12 in \mathbb{Z}_{26} .

Solution

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	-2
6	12	2	0	1	-2	13
	2	0		-2	13	

Module-III

Public Key Cryptography and RSA

Private-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one key**
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

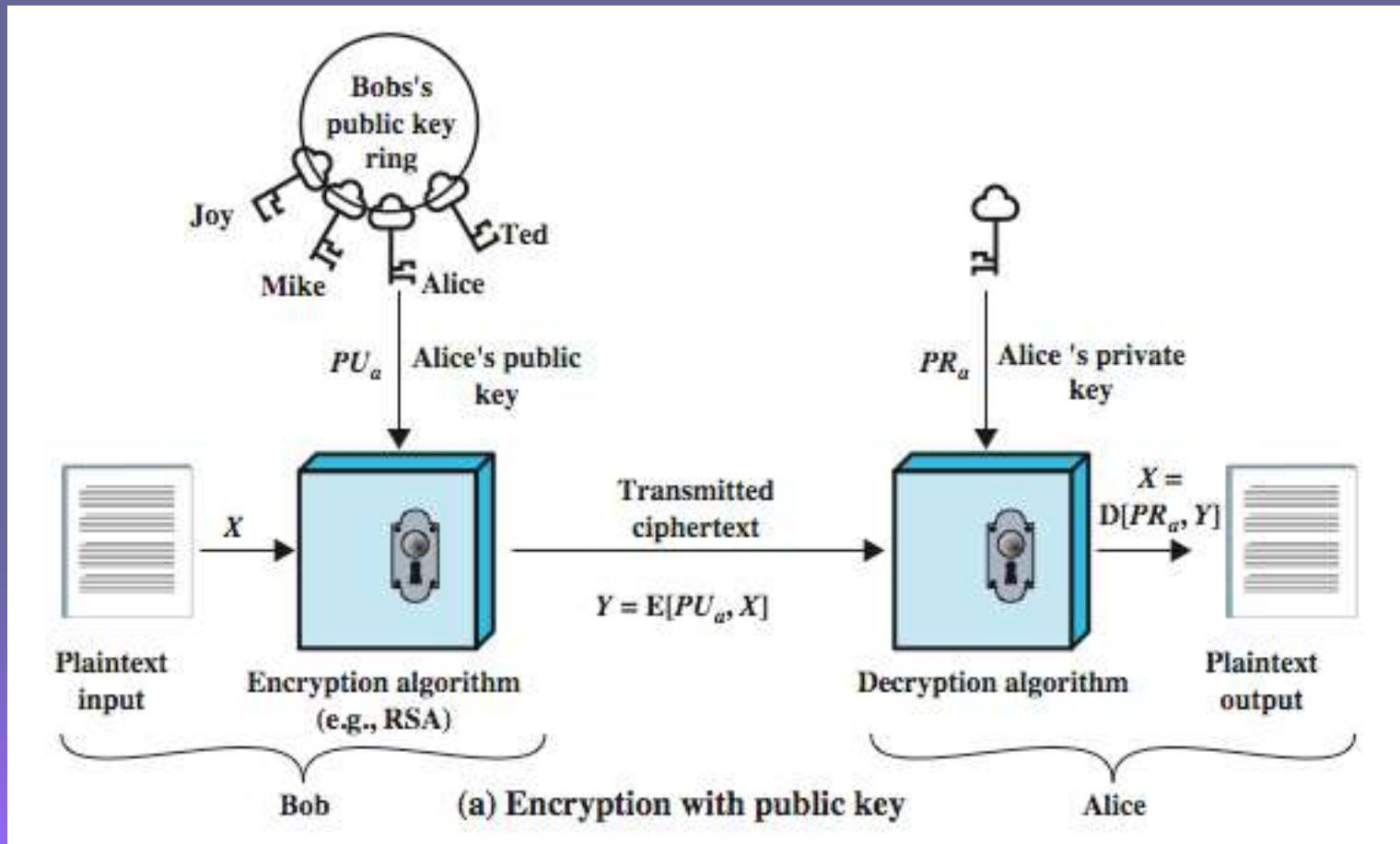
Why Public-Key Cryptography?

- developed to address two key issues:
 - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
 - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
 - known earlier in classified community

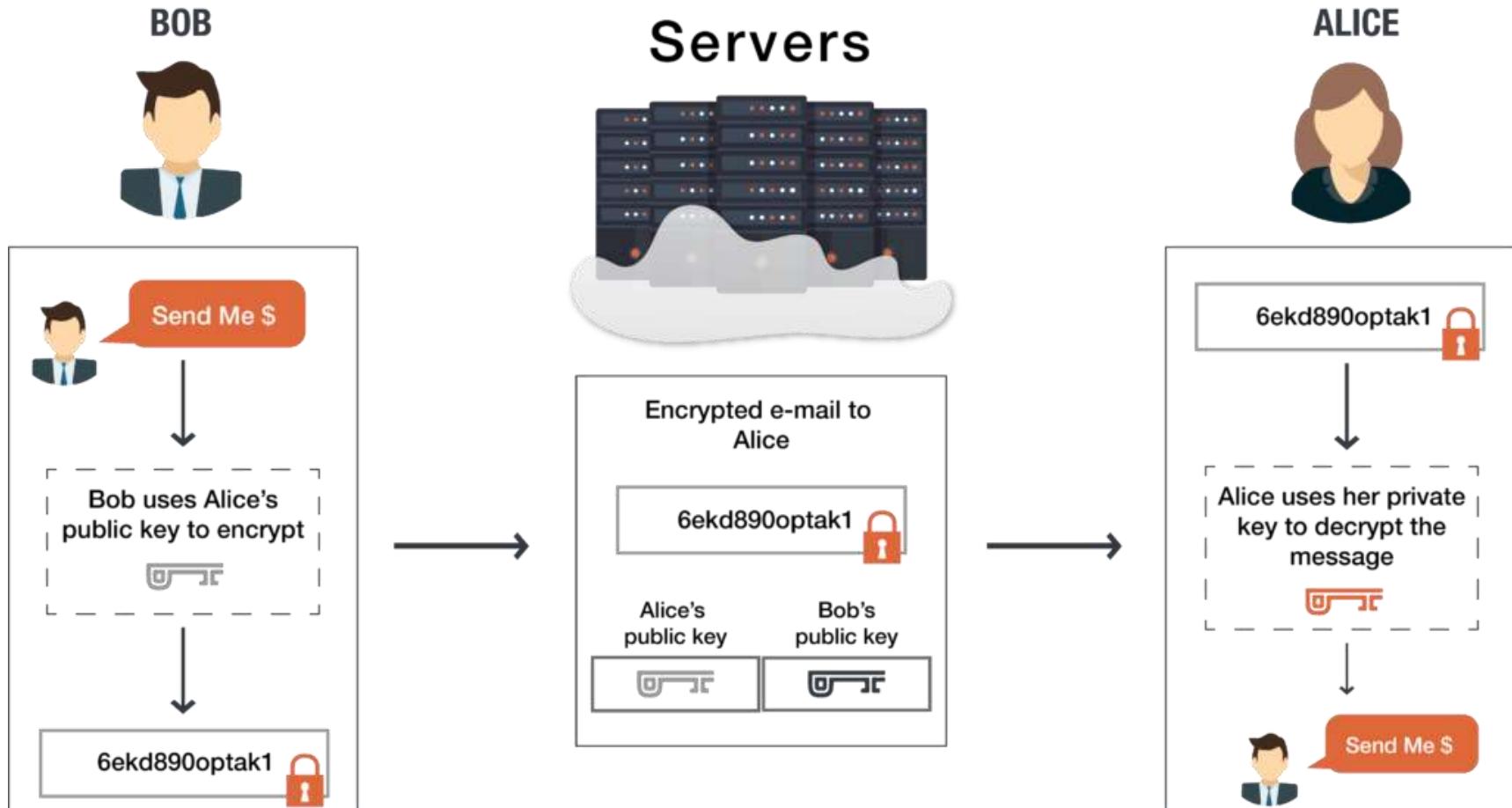
Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
 - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
 - a related **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- **infeasible to determine private key from public**
- **is asymmetric** because
 - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



Asymmetric key cryptography



Public-Key Applications

- can classify uses into 3 categories:
 - **encryption/decryption** (provide secrecy)
 - **digital signatures** (provide authentication)
 - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Public-Key Requirements

- Public-Key algorithms rely on two keys where:
 - it is computationally infeasible to find decryption key knowing only algorithm & encryption key
 - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
 - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
 - factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA En/decryption

- to encrypt a message M the sender:
 - obtains **public key** of recipient $PU = \{e, n\}$
 - computes: $C = M^e \text{ mod } n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
 - uses their private key $PR = \{d, n\}$
 - computes: $M = C^d \text{ mod } n$
- note that the message M must be smaller than the modulus n (block if needed)

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random: p, q
- computing their system modulus $n=p \cdot q$
 - note $\phi(n) = (p-1)(q-1)$
- selecting at random the encryption key e
 - where $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$
- solve following equation to find decryption key d
 - $e \cdot d \equiv 1 \pmod{\phi(n)}$ and $0 \leq d \leq n$
- publish their public encryption key: $PU=\{e,n\}$
- keep secret private decryption key: $PR=\{d,n\}$

RSA Example - Key Setup

1. Select primes: $p=17 \text{ & } q=11$
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e : $\gcd(e, 160) = 1$; choose $e=7$
5. Determine d : $de \equiv 1 \pmod{160}$ and $d < 160$
Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key PU={7, 187}
7. Keep secret private key PR={23, 187}

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88 < 187$)
- encryption:

$$C = 88^7 \bmod 187 = 11$$

- decryption:

$$M = 11^{23} \bmod 187 = 88$$



RSA Key Generation

- users of RSA must:
 - determine two primes at random - p, q
 - select either e or d and compute the other
- primes p, q must not be easily derived from modulus $n=p \cdot q$
 - means must be sufficiently large
 - typically guess and use probabilistic test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

Progress in Factoring

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-years	Algorithm
100	332	April 1991	7	quadratic sieve
110	365	April 1992	75	quadratic sieve
120	398	June 1993	830	quadratic sieve
129	428	April 1994	5000	quadratic sieve
130	431	April 1996	1000	generalized number field sieve
140	465	February 1999	2000	generalized number field sieve
155	512	August 1999	8000	generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

Summary

- have considered:
 - principles of public-key cryptography
 - RSA algorithm, implementation, security

Key Management

Diffie-Hellman Key Exchange Algorithm

by
M.K.Chavan
Asst. Professor
Dept. of Computer Science

The Problem of Key Exchange

- One of the main problems of symmetric key encryption is it requires a secure & reliable channel for the shared key exchange.
- The Diffie-Hellman Key Exchange protocol offers a way in which a public channel can be used to create a confidential shared key.

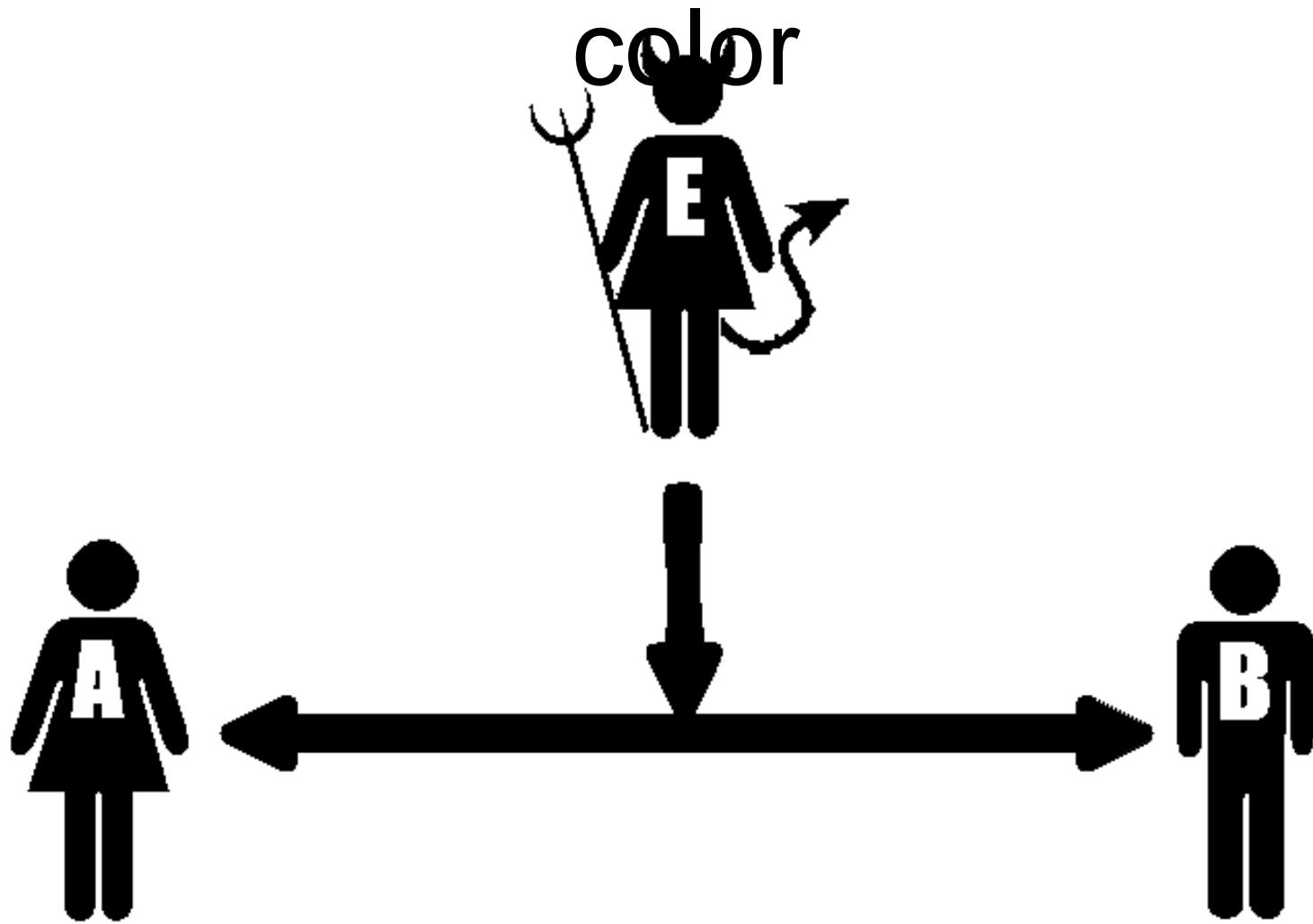
Modular what?

- In practice the shared encryption key relies on such complex concepts as *Modular Exponentiation*, *Primitive Roots* and *Discrete Logarithm Problems*.
- Let's see though is we can explain the Diffie-Hellman algorithm with no complex mathematics.

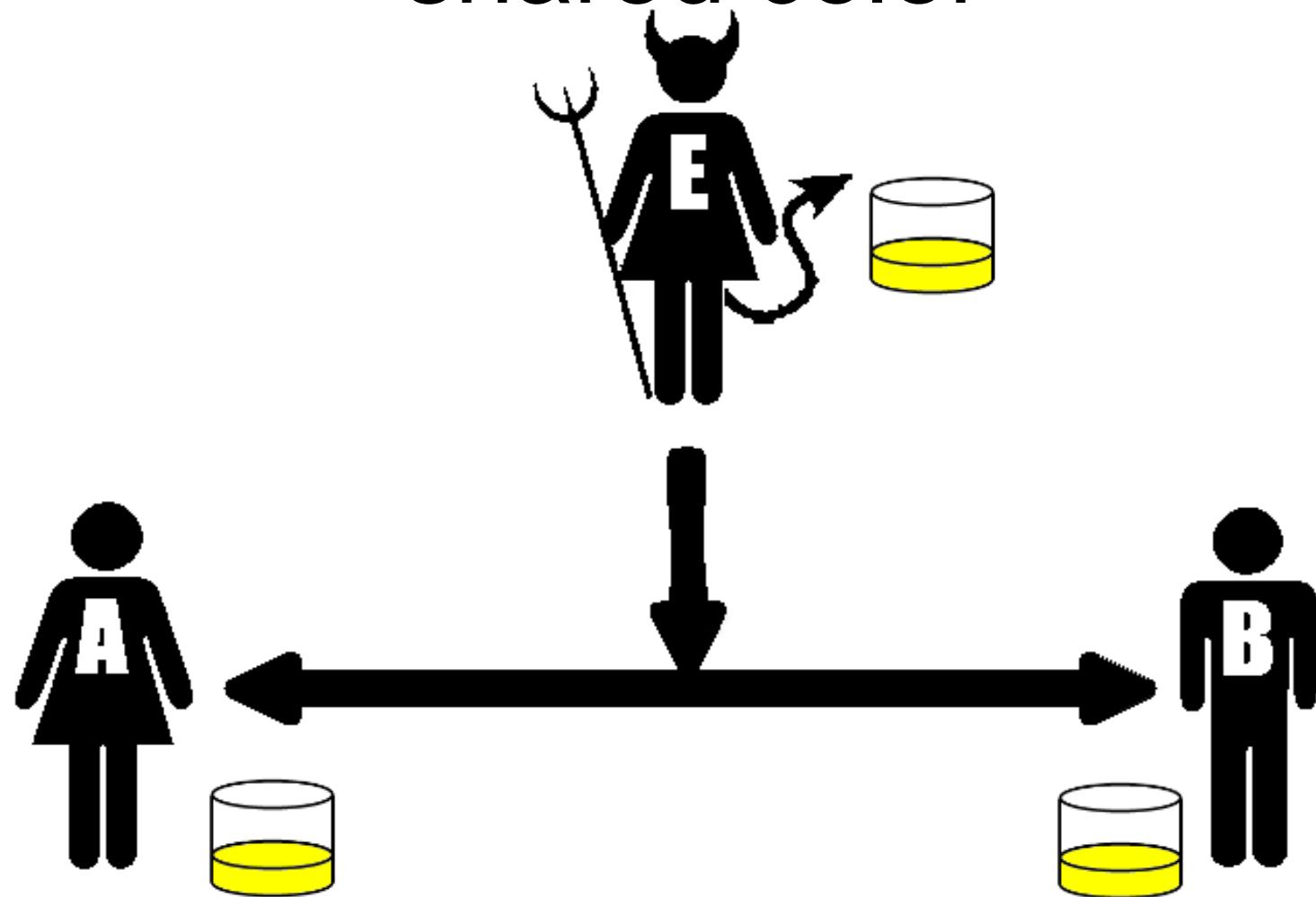
A Difficult One-Way Problem

- The first thing we require is a simple real-world operation that is easy to **Do** but hard to **Undo**.
 - You can ring a bell but not unring one.
 - Toothpaste is easy to squeeze out of a tube but famously hard to put back in.
- In our example we will use *Mixing Colors*.
 - Easy to mix 2 colors, hard to unmix

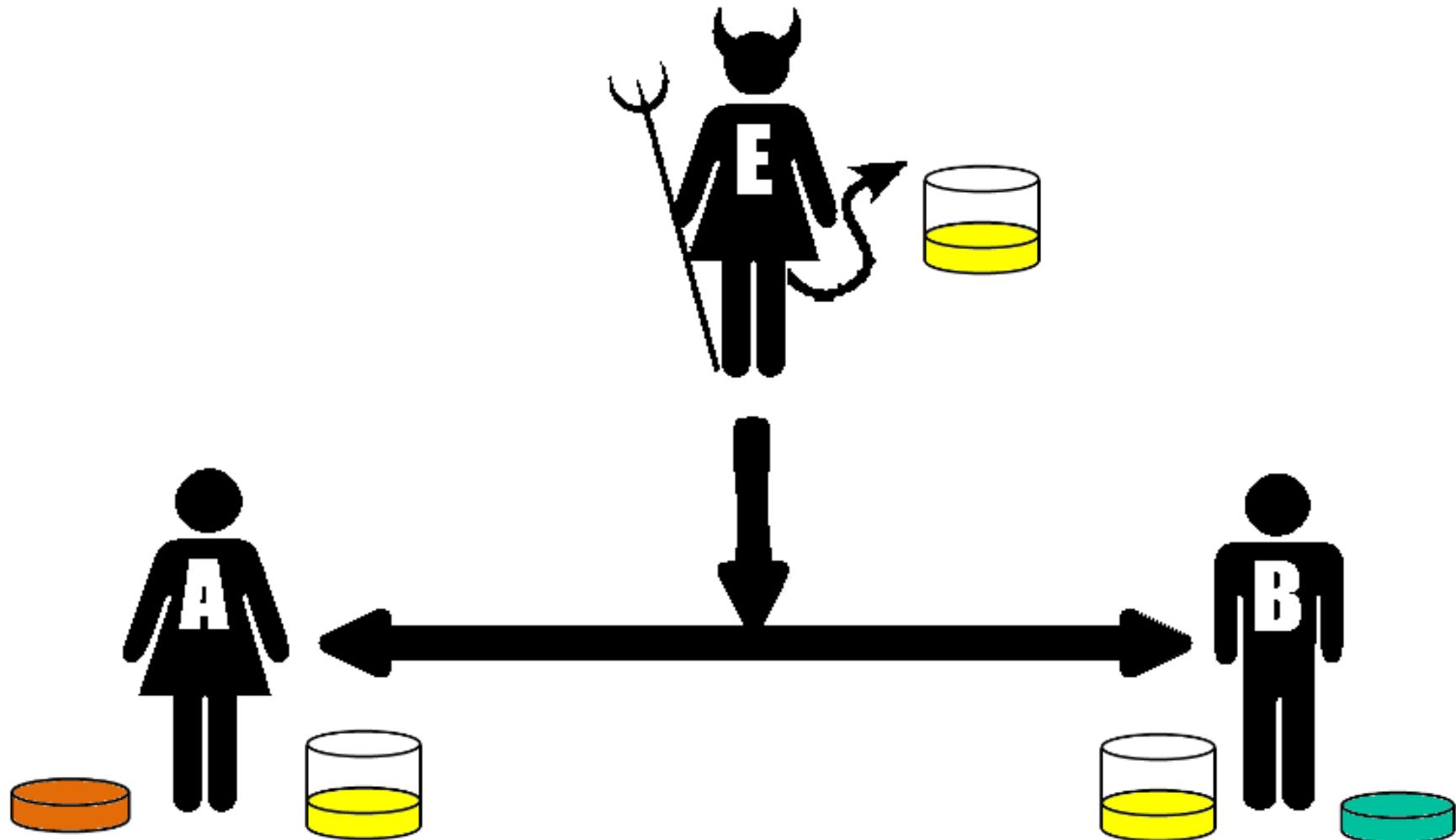
Alice & Bob with Eve listening
wish to make a secret shared



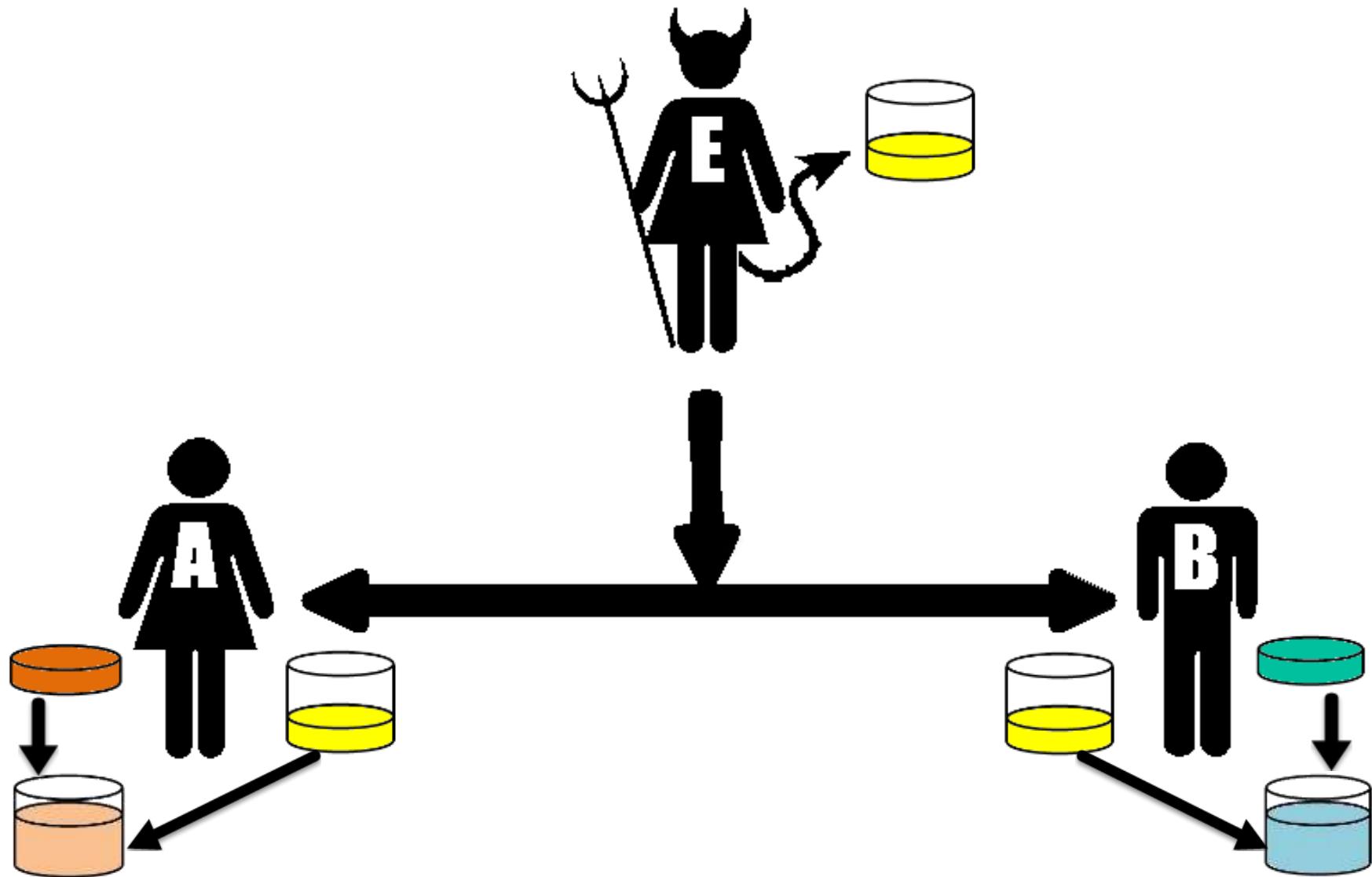
Step 1 - Both publicly agree to a shared color



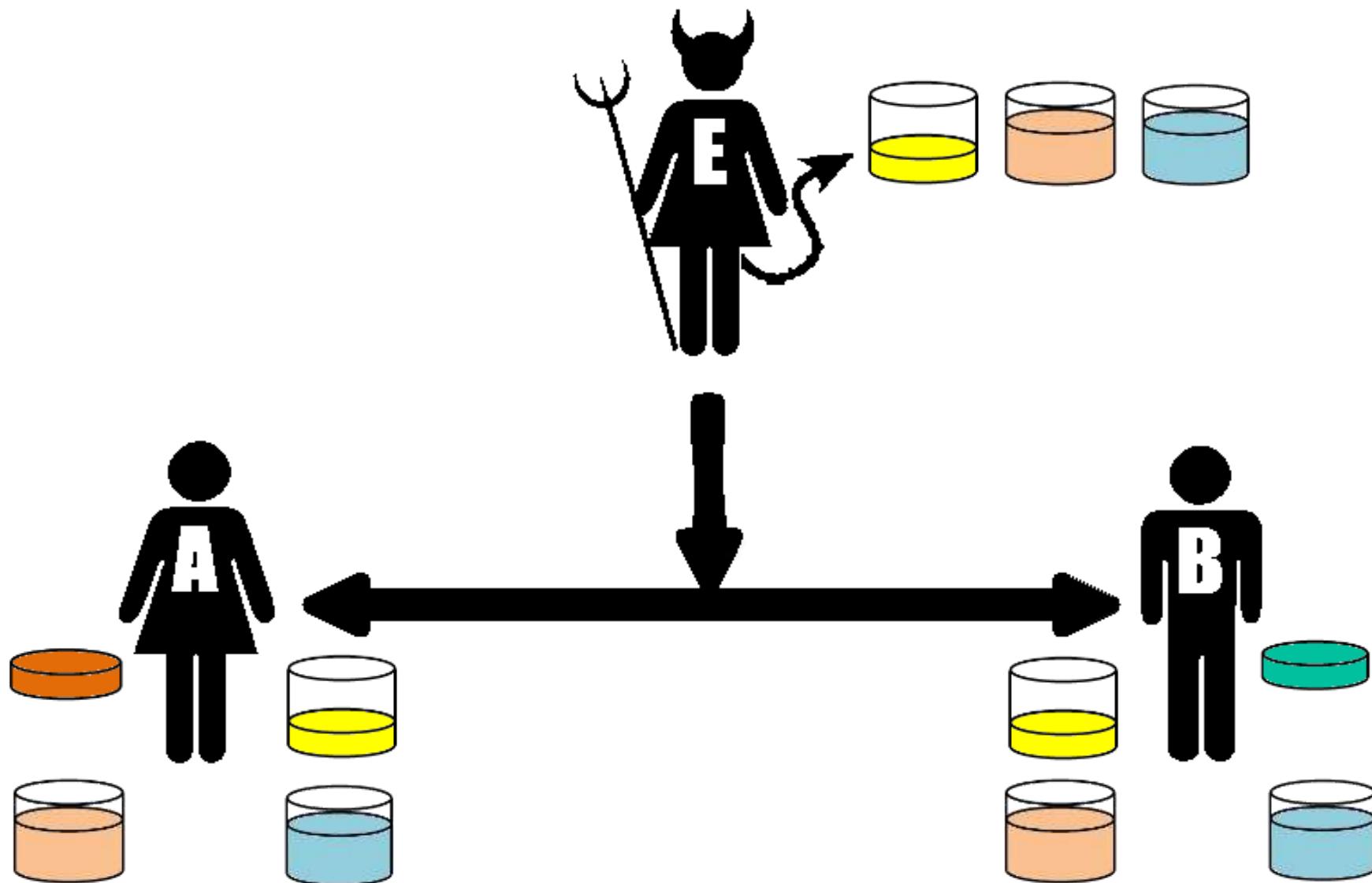
Step 2 - Each picks a secret color



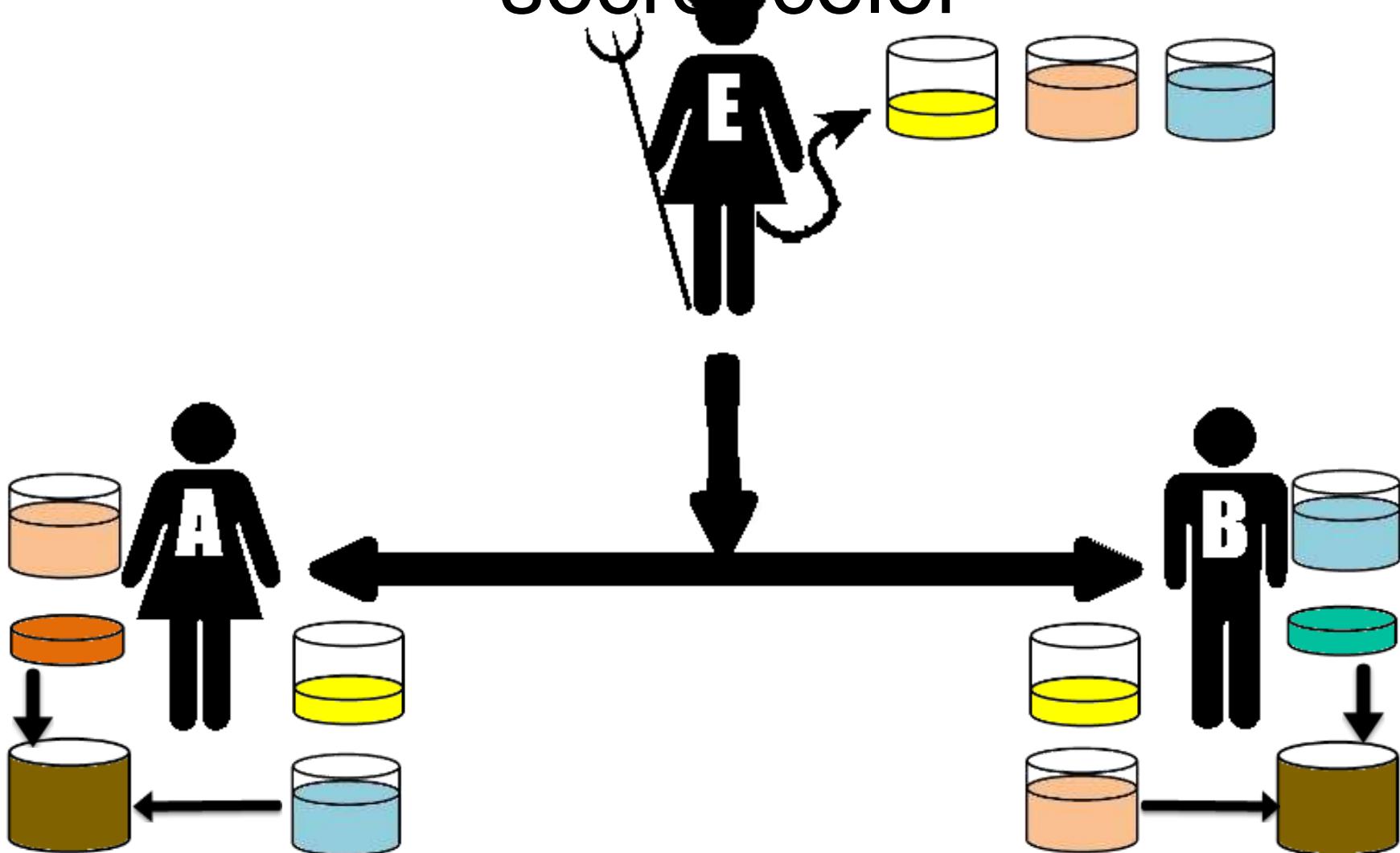
Step 3 - Each adds their secret color to the shared color



Step 4 - Each sends the other their new mixed color



Each combines the shared color from the other with their own secret color



Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixtures are identical?
- Alice mixed
 - $[(\text{Yellow} + \text{Teal}) \text{ from Bob}] + \text{Orange}$
- Bob mixed
 - $[(\text{Yellow} + \text{Orange}) \text{ from Alice}] + \text{Teal}$

Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixture is secret?
- Eve never has knowledge of the secret colors of either Alice or Bob
- Unmixing a color into its component colors is a hard problem

Alice & Bob have agreed to a shared color unknown to Eve

- How is it that Alice & Bob's final mixture is secret?
- Eve never has knowledge of the secret colors of either Alice or Bob
- Unmixing a color into its component colors is a hard problem

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
 - For key distribution only
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
 - note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Setup

- all users agree on global parameters:
 - large prime integer or polynomial q
 - α a primitive root mod q
- each user (eg. A) generates their key
 - chooses a secret key (number): $x_A < q$
 - compute their **public key**: $y_A = \alpha^{x_A} \text{ mod } q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K:

$$K = y_A^{x_B} \bmod q \quad (\text{which } \mathbf{B} \text{ can compute})$$

$$K = y_B^{x_A} \bmod q \quad (\text{which } \mathbf{A} \text{ can compute})$$

(example)

- K is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
 - A chooses $x_A=97$, B chooses $x_B=233$
- compute public keys:
 - $y_A = 3^{97} \text{ mod } 353 = 40 \quad (\text{Alice})$
 - $y_B = 3^{233} \text{ mod } 353 = 248 \quad (\text{Bob})$
- compute shared session key as:
$$K_{AB} = y_B^{x_A} \text{ mod } 353 = 248^{97} \text{ mod } 353 = 160 \quad (\text{Alice})$$
$$K_{AB} = y_A^{x_B} \text{ mod } 353 = 40^{233} \text{ mod } 353 = 160 \quad (\text{Bob})$$

Reference

- Cryptography and Network Security Principles and Practices, William Stallings, 4th Edition.



Cryptographic Hash Functions

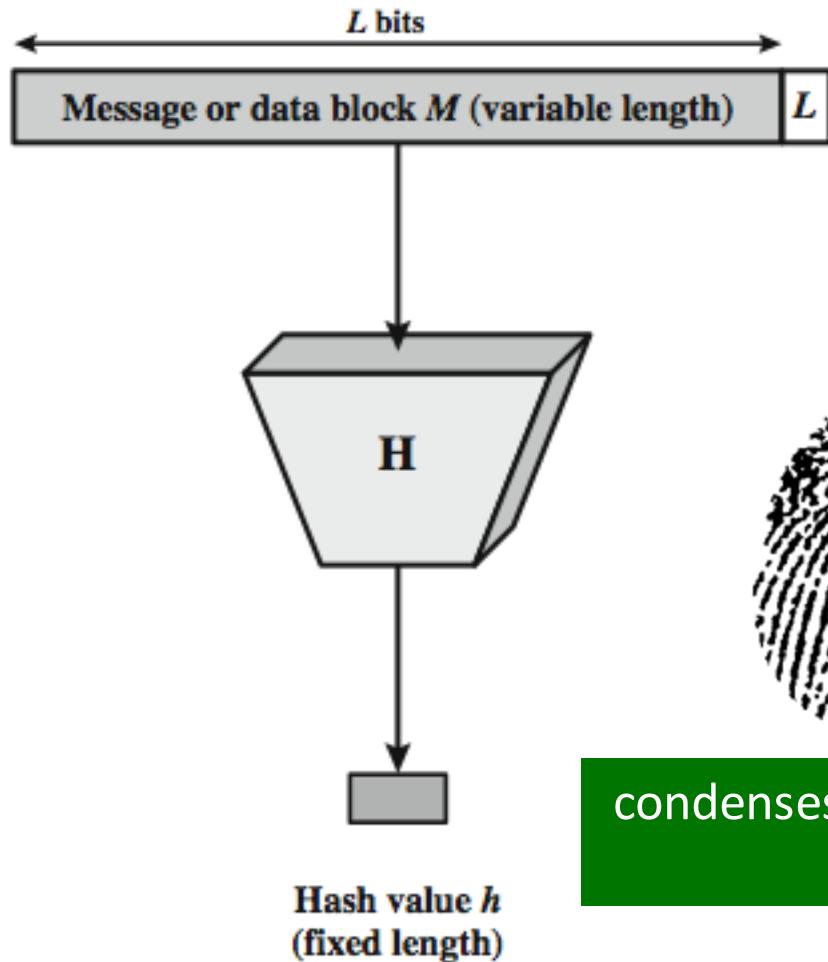


by M.K.Chavan

Topics

- ▶ **Overview of Cryptography Hash Function**
- ▶ Usages
- ▶ Properties
- ▶ Hashing Function Structure
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Hash Function



- ▶ The hash value represents concisely the longer message
 - ▶ may called the *message digest*

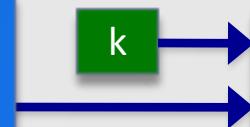


- ▶ A message digest is as a "digital fingerprint" of the original document

condenses arbitrary message to fixed size
$$h = H(M)$$

Hashing V.S. Encryption

Hello, world.
A sample sentence to show encryption.



E

NhbXBsZSBzZW50ZW5jZS
B0byBzaG93IEVuY3J5cHR
pb24KsZSBzZ

Hello, world.
A sample sentence to show encryption.

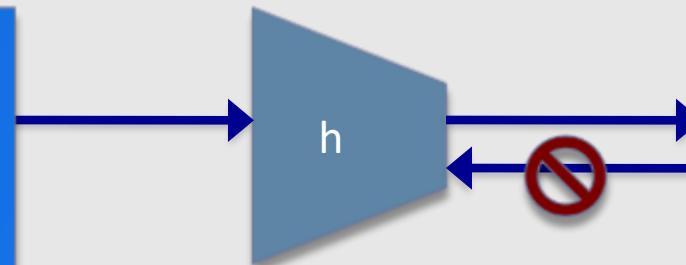


D

NhbXBsZSBzZW50ZW5jZS
B0byBzaG93IEVuY3J5cHR
pb24KsZSBzZ

- ▶ Encryption is two way, and requires a key to encrypt/decrypt

This is a clear text that can easily read without using the key. The sentence is longer than the text above.



h

52f21cf7c7034a20
17a21e17e061a863

- ▶ Hashing is one-way. There is no 'de-hashing'

Motivation for Hash Algorithms

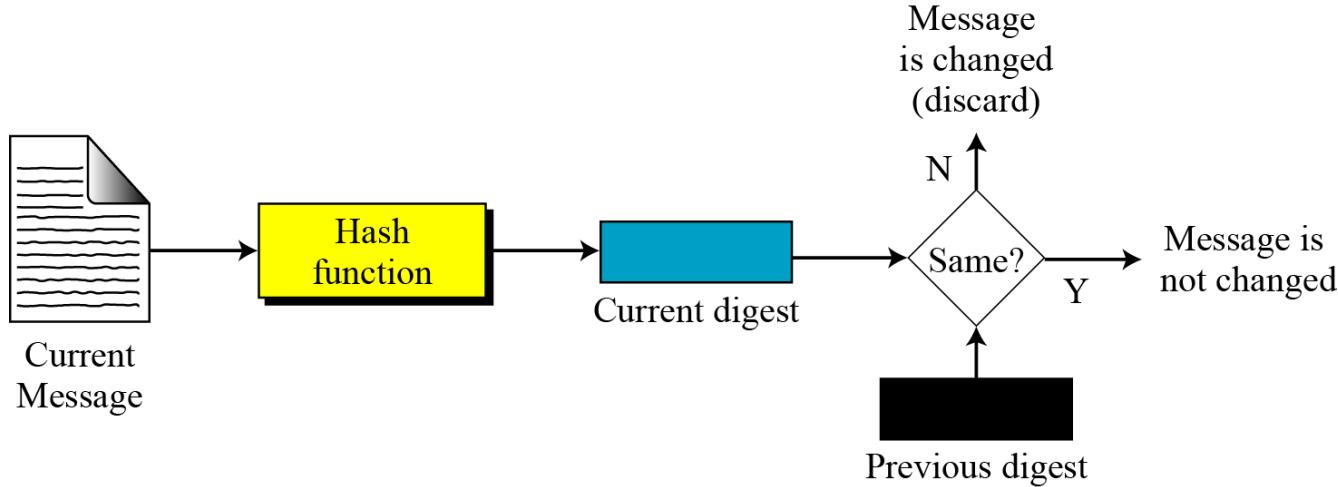
▶ Goal

- ▶ Design a code where the original message can not be inferred based on its checksum
- ▶ such that an accidental or intentional change to the message will change the hash value

Hash Function Applications

- ▶ Used Alone
 - ▶ Fingerprint -- file integrity verification
 - ▶ Password storage (one-way encryption)
- ▶ Combined with encryption functions
 - ▶ Hash based Message Authentication Code (HMAC)
 - ▶ protects both a message's integrity and confidentiality
 - ▶ Digital signature
 - ▶ Ensuring Non-repudiation
 - ▶ Encrypt hash with private (signing) key and verify with public (verification) key

Integrity



- ▶ to create a one-way password file
 - ▶ store hash of password not actual password
- ▶ for intrusion detection and virus detection
 - ▶ keep & check hash of files on system

Password Verification

Store Hashing Password

Iam#4VKU

h

661dce0da2bcb2d8
2884e0162acf8194

Password store

Verification an input password against the stored hash

Iam#4VKU

h

661dce0da2bcb2d8
2884e0162acf8194

Password store

661dce0da2bcb2d8
2884e0162acf8194

Hash Matching
Exactly?

Yes

No

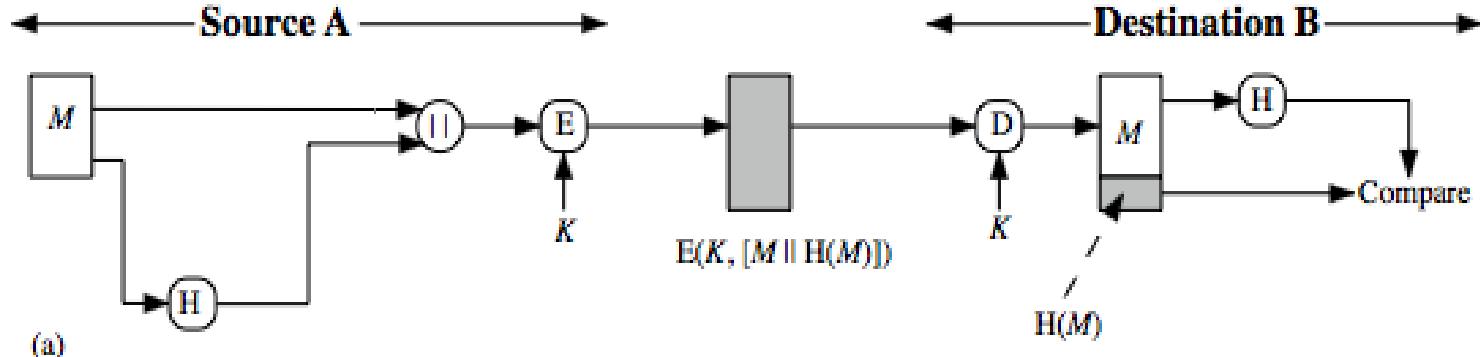
Grant

Deny

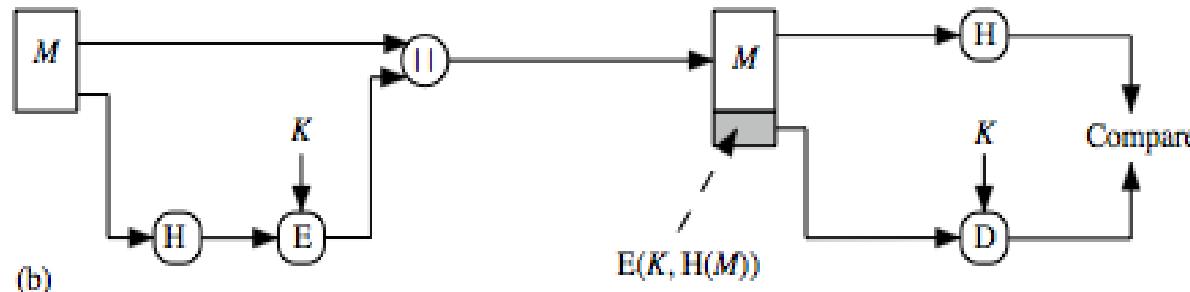
Topics

- ▶ Overview of Cryptography Hash Function
- ▶ **Usages**
- ▶ Properties
- ▶ Hashing Function Structure
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Hash Function Usages (I)

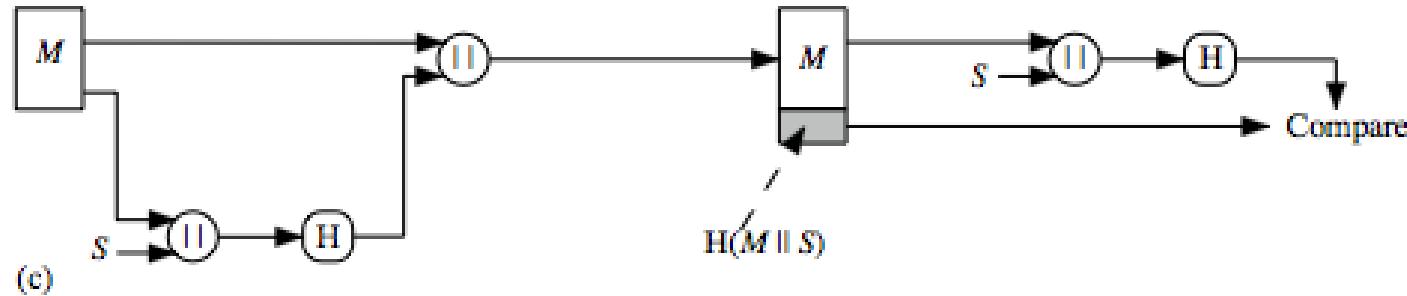


Message encrypted : Confidentiality and authentication

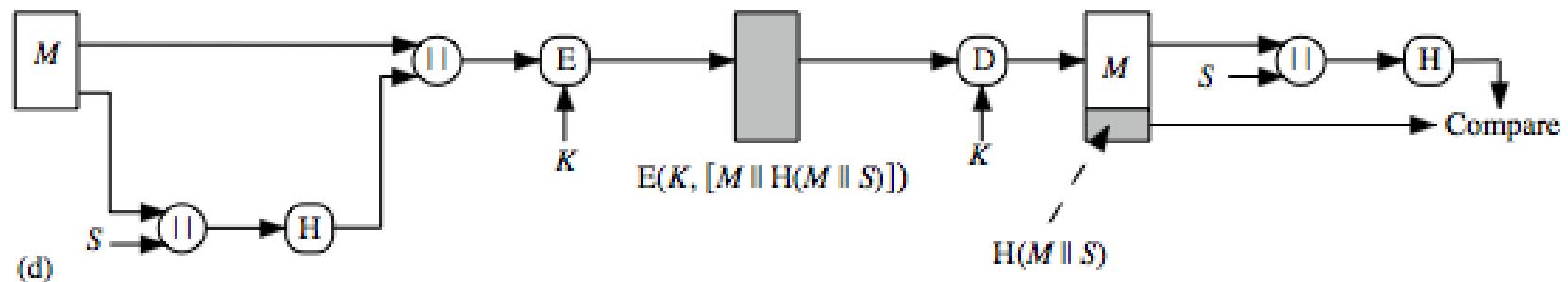


Message unencrypted: Authentication

Hash Function Usages (II)

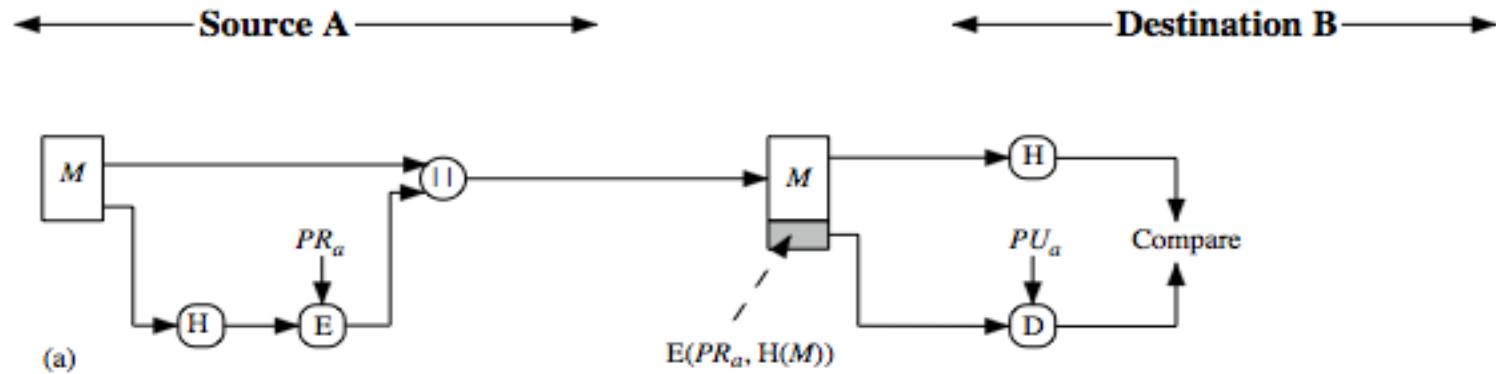


Message encrypted : Authentication (no encryption needed!)

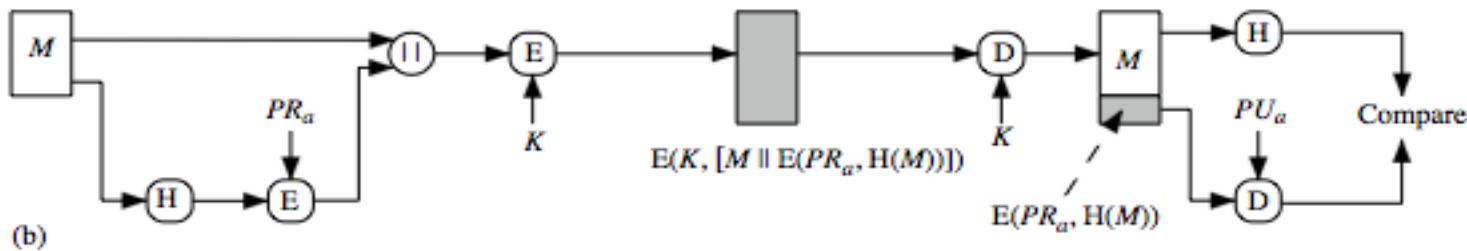


Message unencrypted: Authentication, confidentiality

Hash Function Usages (III)



Authentication, digital signature



Authentication, digital signature, confidentiality

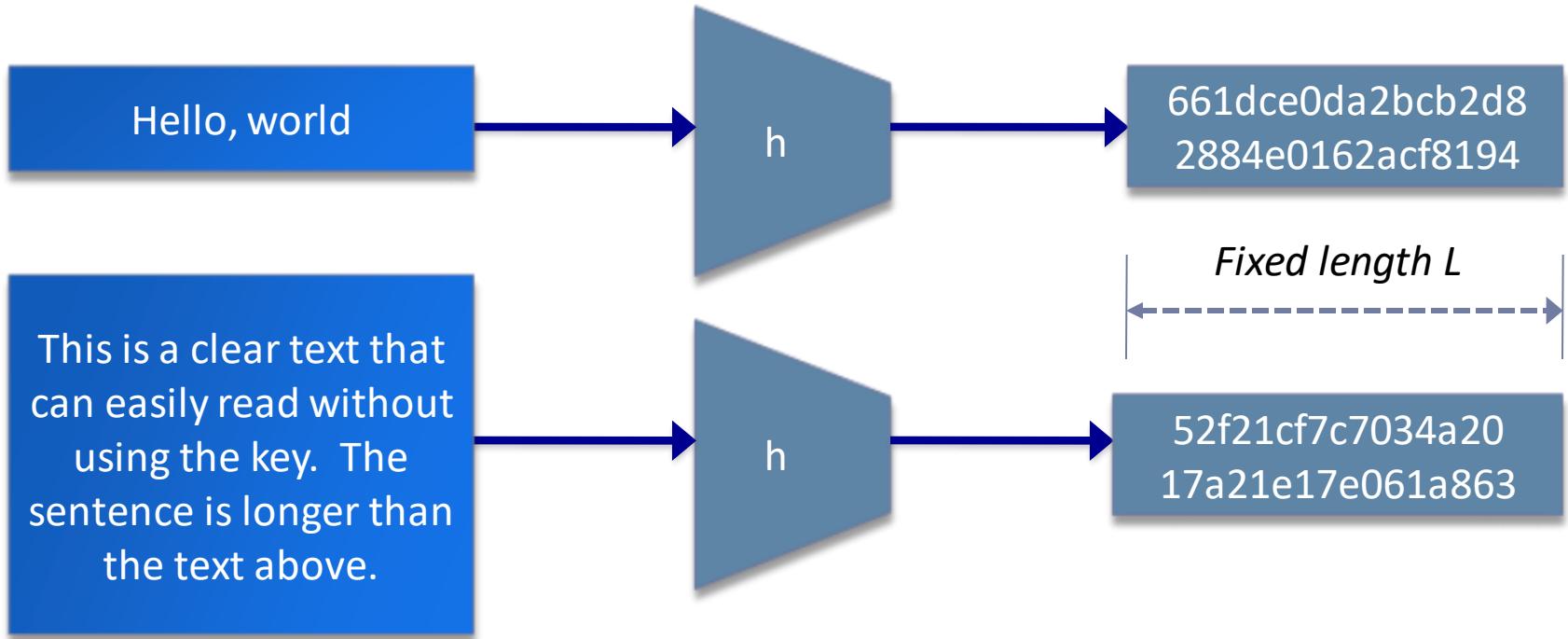
Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ **Properties**
- ▶ Hashing Function Structure
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Hash Function Properties

- ▶ Arbitrary-length message to fixed-length digest
- ▶ Preimage resistant (**One-way property**)
- ▶ Collision resistant (**Strong collision resistance**)

Properties : Fixed length



- ▶ Arbitrary-length message to fixed-length digest

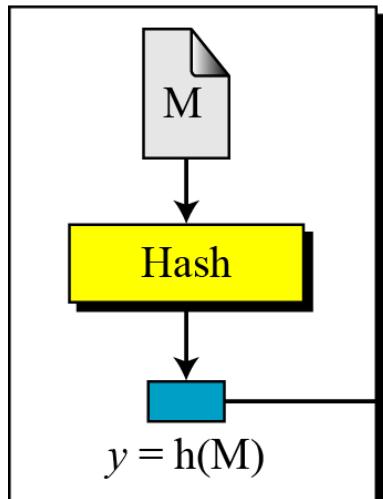
Preimage resistant

- ▶ This measures how difficult to devise a message which hashes to the known digest
- ▶ Roughly speaking, the hash function must be one-way.

Preimage Attack

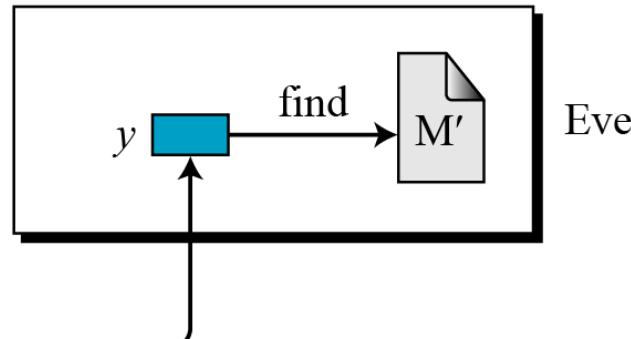
Given: $y = h(M)$

M: Message
Hash: Hash function
 $h(M)$: Digest



Find: M' such that $y = h(M')$

Given: y
Find: any M' such that
 $y = h(M')$



Given only a message digest, can't find any message (or preimage) that generates that digest.

Collision Resistant

Given: none

Collision Attack

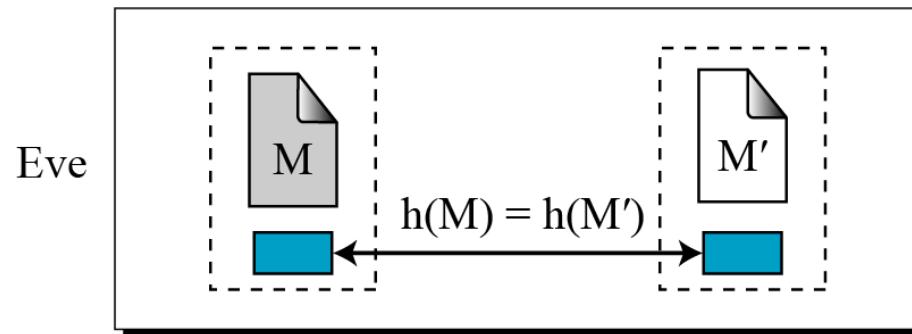
Find: $M' \neq M$ such that $h(M) = h(M')$

M: Message

Hash: Hash function

$h(M)$: Digest

Find: M and M' such that $M \neq M'$, but $h(M) = h(M')$



- ▶ Can't find any two different messages with the same message digest
 - ▶ Collision resistance implies second preimage resistance
 - ▶ Collisions, if we could find them, would give signatories a way to repudiate their signatures

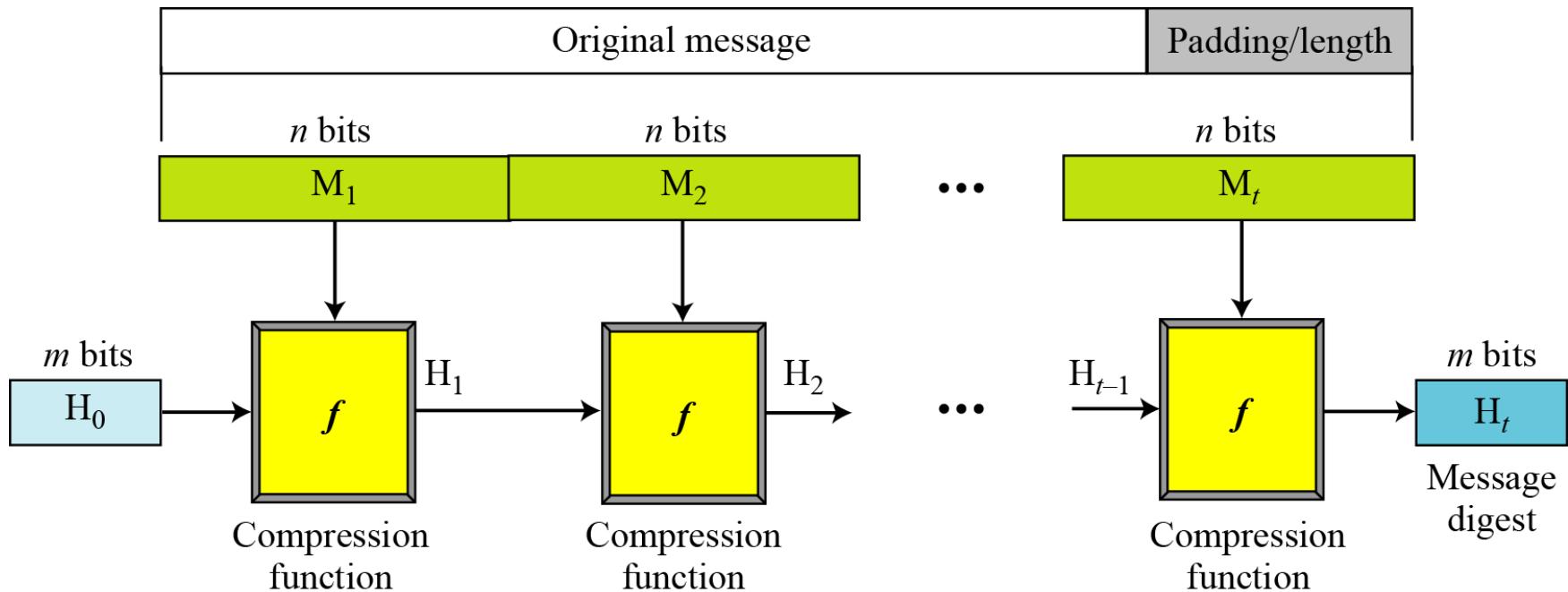
Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ Properties
- ▶ **Hashing Function Structure**
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Two Group of Compression Functions

- ▶ The compression function is made from scratch
 - ▶ Message Digest
- ▶ A symmetric-key block cipher serves as a compression function
 - ▶ Whirlpool

Merkle-Damgard Scheme



- ▶ Well-known method to build cryptographic hash function
- ▶ A message of arbitrary length is broken into blocks
 - ▶ length depends on the compression function f
 - ▶ padding the size of the message into a multiple of the block size.
 - ▶ sequentially process blocks , taking as input the result of the hash so far and the current message block, with the final fixed length output

Hash Functions Family

- ▶ **MD (Message Digest)**
 - ▶ Designed by Ron Rivest
 - ▶ Family: MD2, MD4, MD5
- ▶ **SHA (Secure Hash Algorithm)**
 - ▶ Designed by NIST
 - ▶ Family: SHA-0, SHA-1, and SHA-2
 - ▶ SHA-2: SHA-224, SHA-256, SHA-384, SHA-512
 - ▶ SHA-3: New standard in competition
- ▶ **RIPEMD (Race Integrity Primitive Evaluation Message Digest)**
 - ▶ Developed by Katholieke University Leuven Team
 - ▶ Family : RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320

MD5, SHA-1, and RIPEMD-160

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

MD2, MD4 and MD5

- ▶ Family of one-way hash functions by Ronald Rivest
 - ▶ All produces 128 bits hash value
- ▶ **MD2: 1989**
 - ▶ Optimized for 8 bit computer
 - ▶ Collision found in 1995
- ▶ **MD4: 1990**
 - ▶ Full round collision attack found in 1995
- ▶ **MD5: 1992**
 - ▶ Specified as Internet standard in RFC 1321
 - ▶ Practical Collision MD5 has been broken since 2004
 - ▶ CA attack published in 2007

Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ Properties
- ▶ **Hashing Function Structure**
 - ▶ MD5
 - ▶ SHA
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ Properties
- ▶ **Hashing Function Structure**
 - ▶ MD5
 - ▶ SHA
- ▶ Attack on Hash Function
- ▶ The Road to new Secure Hash Standard

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
 - revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
 - standard is FIPS 180-1 1995, also Internet RFC3174
 - based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

Revised SHA

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
 - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

SHA Versions

	MD5	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Digest size	128	160	160	224	256	384	512
Message size	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Block size	512	512	512	512	512	1024	1024
Word size	32	32	32	32	32	64	64
# of steps	64	64	80	64	64	80	80

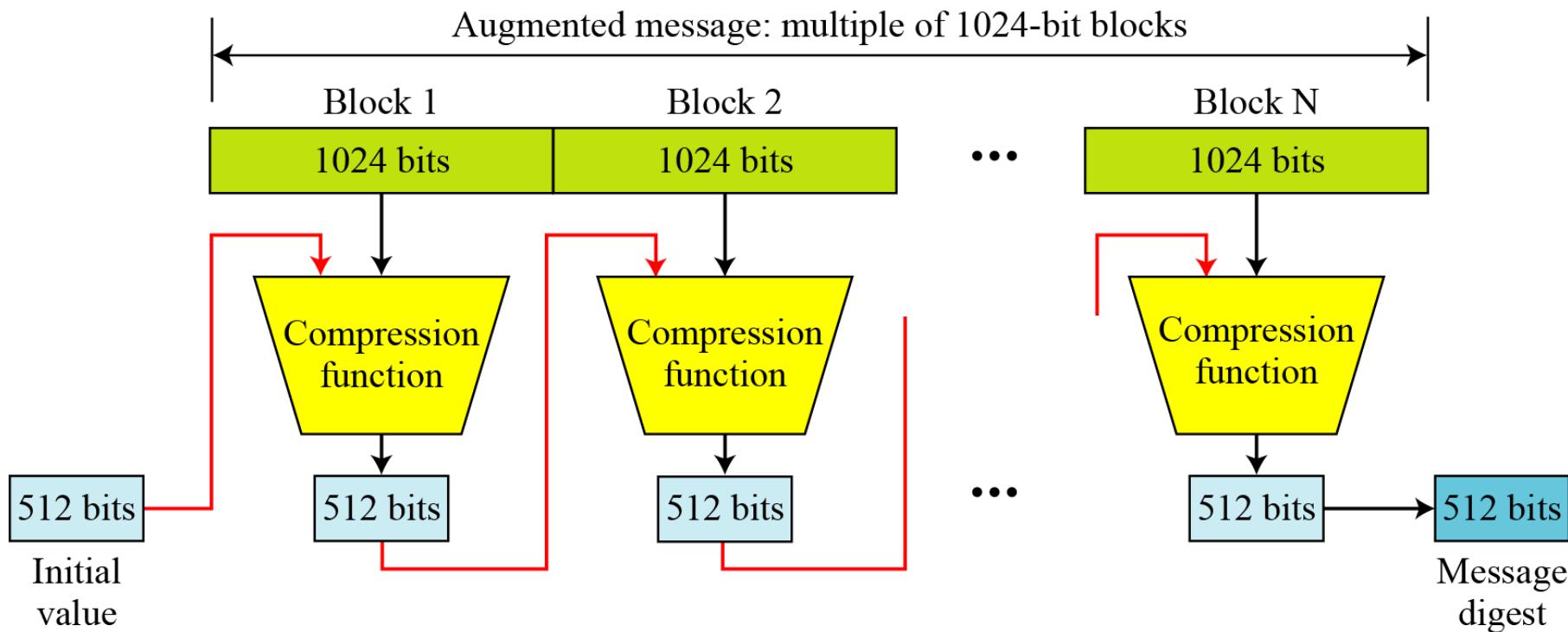
Full collision found

Sample Processing

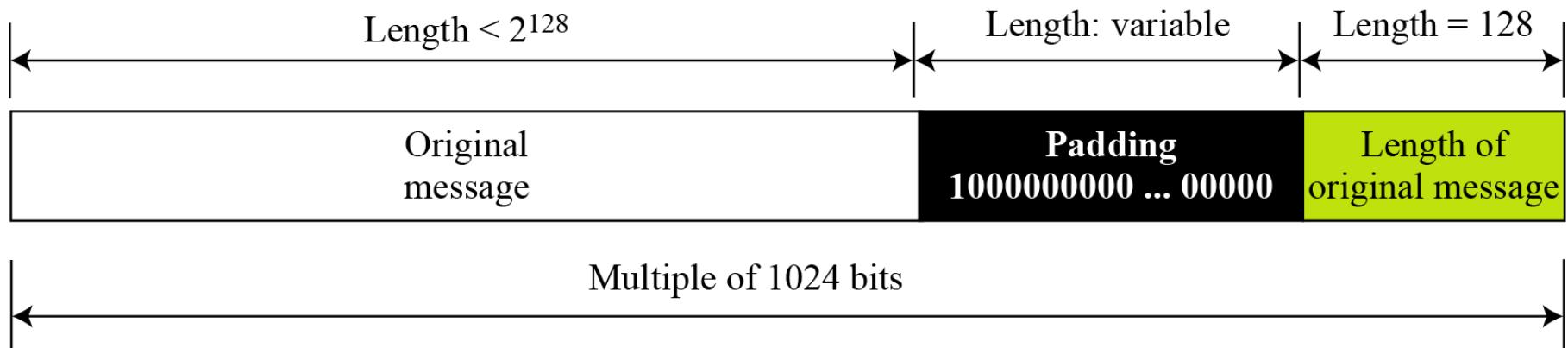
Type	bits	data processed
MD5	128	469.7 MB/s
SHA-1	160	339.4 MB/s
SHA-512	512	177.7 MB/s

- ▶ Mac Intel 2.66 Ghz core i7
- ▶ 1024 bytes block of data

SHA-512 Overview



Padding and length field in SHA-512

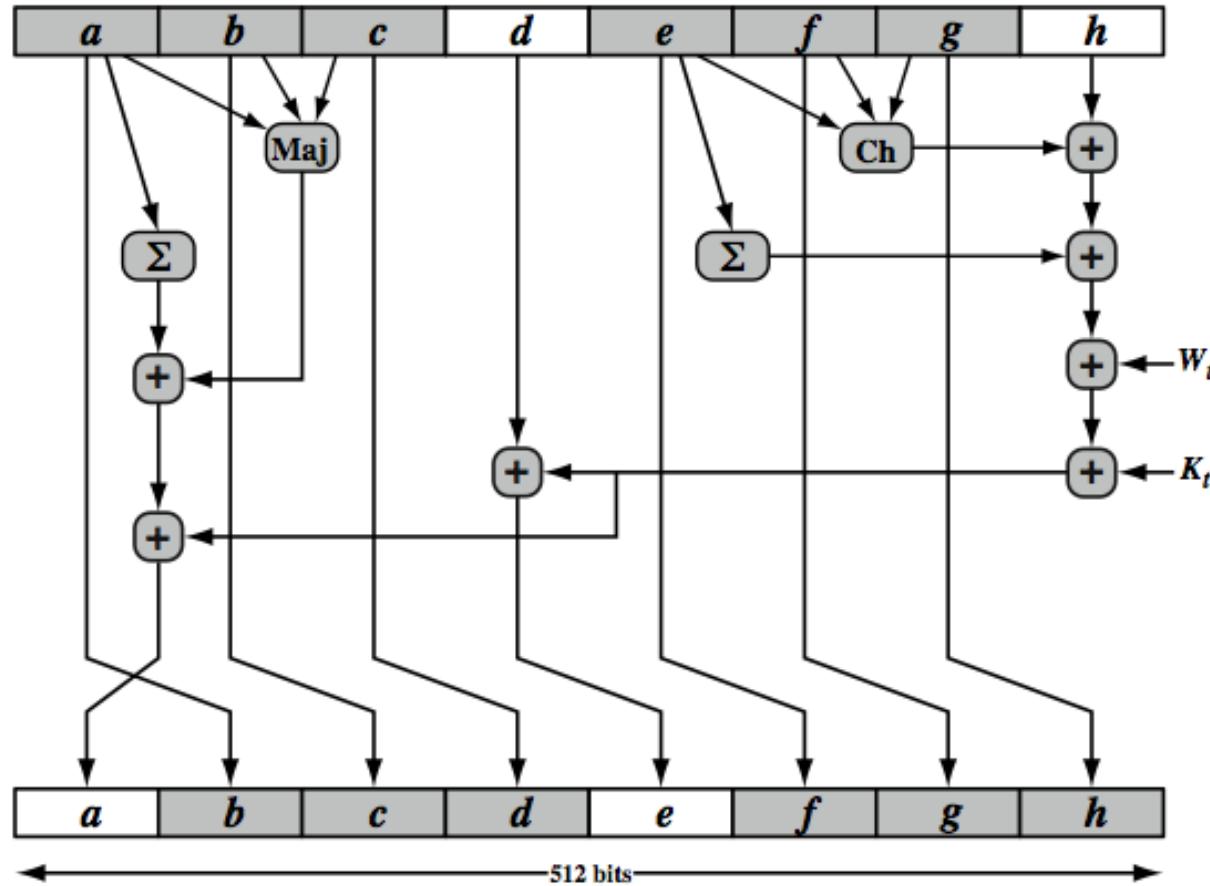


- ▶ **What is the number of padding bits if the length of the original message is 2590 bits?**
- ▶ We can calculate the number of padding bits as follows:

$$|P| = (-2590 - 128) \bmod 1024 = -2718 \bmod 1024 = 354$$

- ▶ The padding consists of one 1 followed by 353 0's.

SHA-512 Round Function

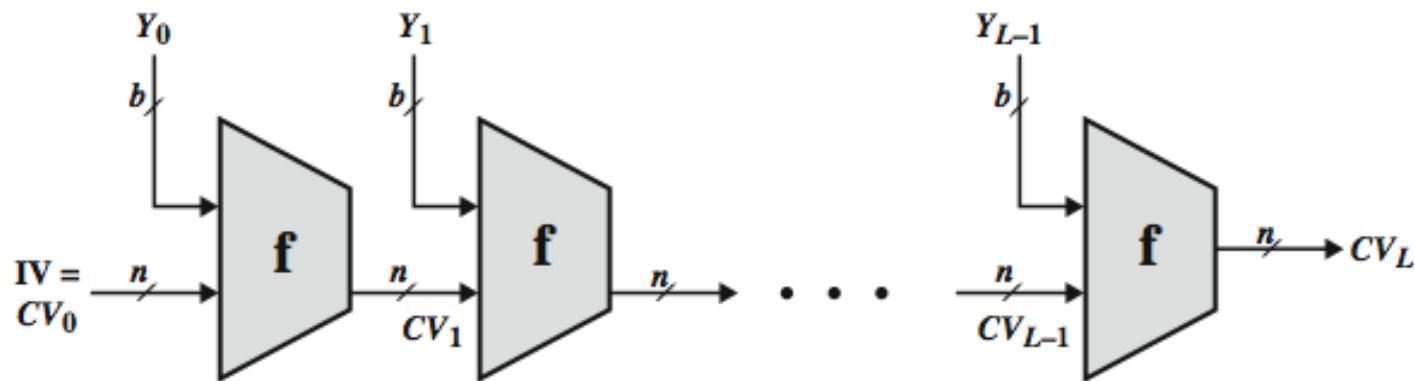


Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ Properties
- ▶ Hashing Function Structure
 - ▶ MD5
 - ▶ SHA

Hash Function Cryptanalysis

- cryptanalytic attacks exploit some property of algorithm so faster than exhaustive search
- hash functions use iterative structure
 - process message in blocks (incl length)
- attacks focus on collisions in function f



Attacks on Hash Functions

- brute-force attacks and cryptanalysis
 - cryptanalytic attacks exploit some property of algorithm so faster than brute-force
- a preimage or second preimage attack
 - find y such that $H(y)$ equals a given hash value
- collision resistance
 - find two messages x & y with same hash so $H(x) = H(y)$

"md5 and sha1 are both clearly broken (in terms of collision-resistance)"

Ron Rivest

<http://mail.python.org/pipermail/python-dev/2005-December/058850.html>

Topics

- ▶ Overview of Cryptography Hash Function
- ▶ Usages
- ▶ Properties
- ▶ Hashing Function Structure
 - ▶ MD5
 - ▶ SHA
- ▶ Attack on Hash Function
- ▶ **The Road to new Secure Hash Standard**

The need of new Hash standard

- *MD5 should be considered cryptographically broken and unsuitable for further use, US CERT 2010*
- In 2004, a collision for the full SHA-0 algorithm was announced
- SHA-1 not yet fully “**broken**”
 - but similar to the broken MD5 & SHA-0
 - so considered insecure and be fade out
- SHA-2 (esp. SHA-512) seems secure
 - shares same structure and mathematical operations as predecessors so have concern

SHA-3 Requirements

- NIST announced in 2007 a competition for the SHA-3 next gen hash function
- Replace SHA-2 with SHA-3 in any use
 - so use same hash sizes
- preserve the nature of SHA-2
 - so must process small blocks (512 / 1024 bits)
- evaluation criteria
 - security close to theoretical max for hash sizes
 - cost in time & memory
 - characteristics: such as flexibility & simplicity

Timeline Competition

- ▶ **Nov 2007:** Announce public competition
- ▶ **Oct 2008:** 64 Entries
- ▶ **Dec 2008:** 51 Entries as 1st Round
- ▶ **Jul 2009:** 14 Entries as 2nd Round
- ▶ **Dec 2010:** 5 Entries as 3rd Round
- ▶ **Jan 2011:** Final packages submission and enter public comments
- ▶ **2012:** *SHA-3 winner announcement (Still in progress)*

Five SHA-3 Finalists

- ▶ BLAKE
- ▶ Grøstl
- ▶ JH
- ▶ Keccak
- ▶ Skien

http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html

Secure Hash Algorithm-512 (SHA-512)

M.K.Chavan

Outlines

- SHA-512 overview
- processing of SHA-512
- word expansion
- compression function
- round function
- additive constants
- example using SHA-512
- applications
- cryptanalysis



Secure Hash Algorithm-512 (SHA-512)

Outlines

- SHA-512 overview
- processing of SHA-521
- word expansion
- compression function
- round function
- additive constants
- example using SHA-512
- applications
- cryptanalysis

Overview

- - developed by National Institute of Standards and Technology
 - member of SHA-2 family
 - latest version of Secure Hash Algorithm
 - based on the Merkle-Damgard scheme
 - maximum message size $2^{128}-1$ bits
 - block size 1024 bits
 - message digest size 512 bits
 - number of rounds 80
 - word size 64 bits

Processing of SHA-512

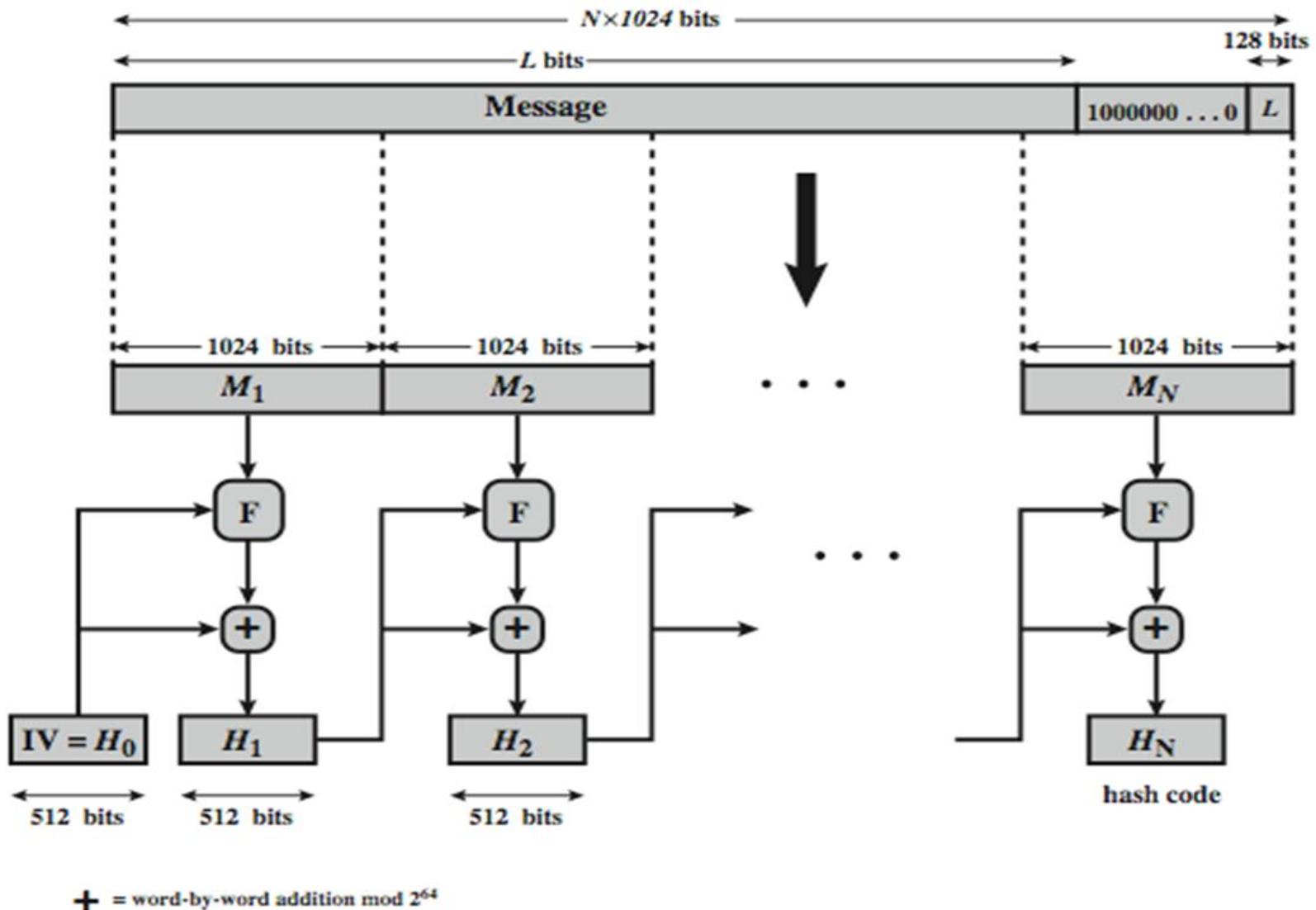


Figure: SHA-512 Processing of a Single 1024-Bit Block 46

Message block and digest word

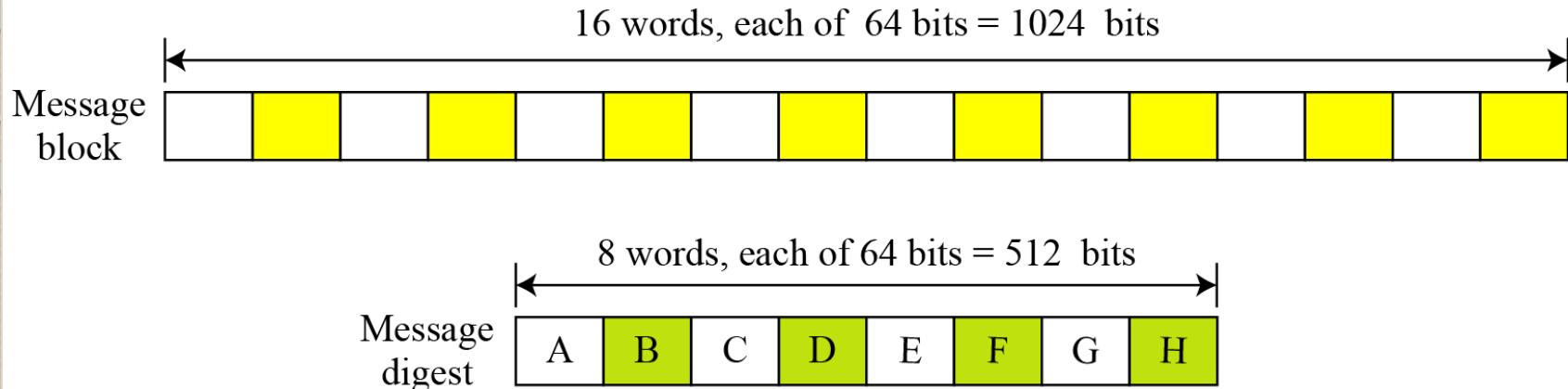


Figure: A message block and the digest as words

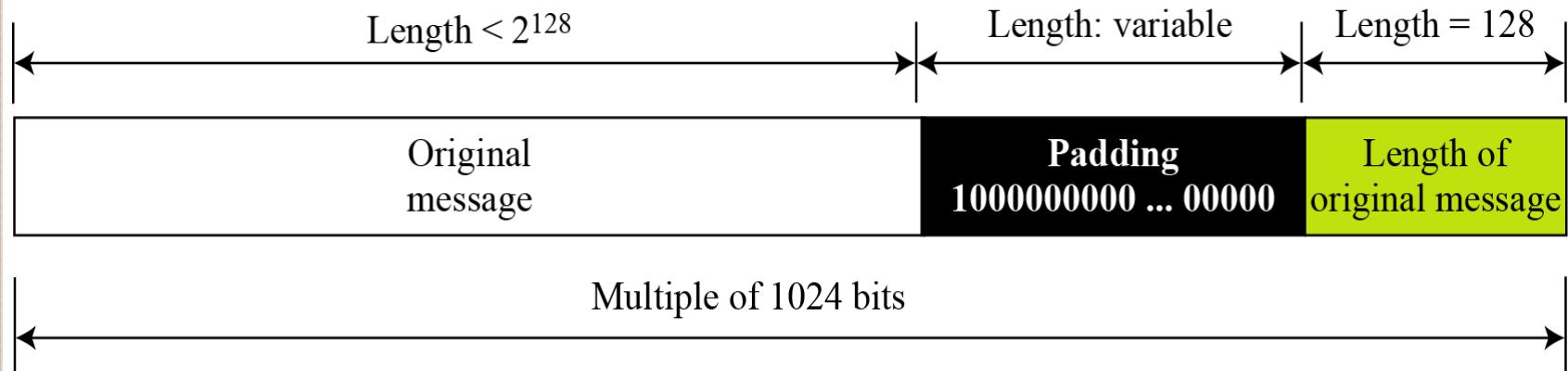


Figure: Padding and length field

Processing of SHA-512

- ❑ appending padding and fixed 128 bit length field
- ❑ dividing the augmented message into blocks
- ❑ using a 64-bit word derived from the current message block
- ❑ using 8 constants based on square root of first 8 prime numbers (2-19)
- ❑ updating a 512-bit buffer
- ❑ using a round constant based on cube root of first 80 prime numbers (2-409)

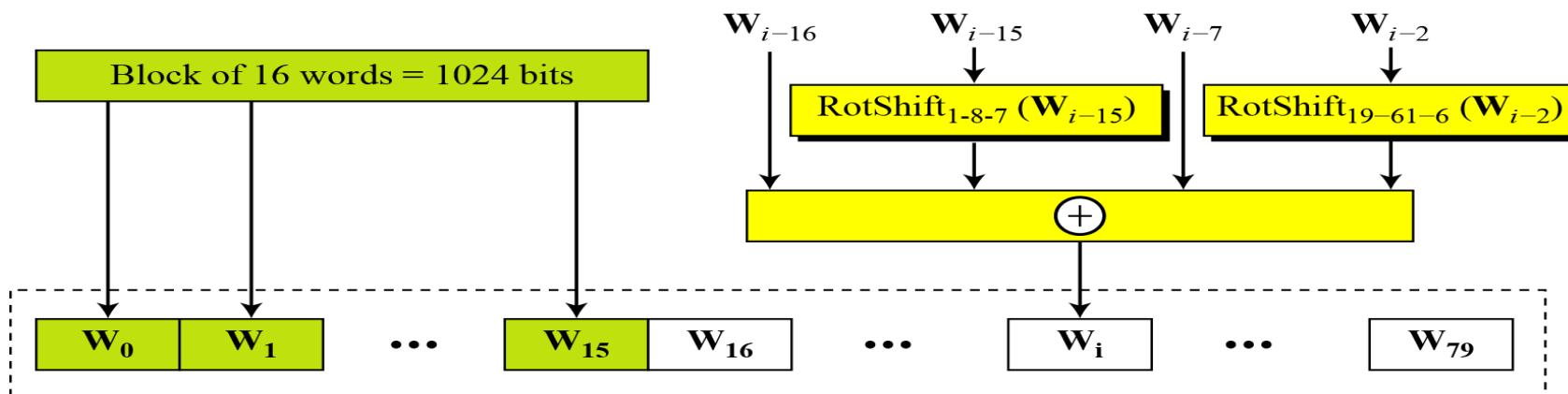
Message block and digest word

- operates on words
- each block consists of sixteen 64 bits(1024 bits) words
- message digest has eight 64 bits (512 bits) words named A,B,C,D,E,F,G,H
- expanding 80 words from sixteen 64 bits words

SHA-512 Initial Values & Word Expansion

Buffer	Value (in Hexadecimal)	Buffer	Value (in Hexadecimal)
A ₀	6A09E667F3BCC908	E ₀	510E527FADE682D1
B ₀	BB67AE8584CAA73B	F ₀	9B05688C2B3E6C1F
C ₀	3C6EF372FE94F82B	G ₀	1F83D9ABFB41BD6B
D ₀	A54FF53A5F1D36F1	H ₀	5BE0CD19137E2179

Table : Values of constants in message digest initialization of SHA-512



$\text{RotShift}_{1-m-n}(x) : \text{RotR}_l(x) \oplus \text{RotR}_m(x) \oplus \text{ShL}_n(x)$

$\text{RotR}_i(x)$: Right-rotation of the argument x by i bits

$\text{ShL}_i(x)$: Shift-left of the argument x by i bits and padding the left by 0's.

Figure: Word expansion in SHA-512

Calculation of constants

For example,

The 8th prime is 19, with the square root $(19)^{1/2} = 4.35889894354$ Converting this number to binary with only 64 bits in the fraction part, we get,

$$(100.0101\ 1011\ 1110 \dots 1001)_2 \longrightarrow (4.5BE0CD19137E2179)_{16}$$

The fraction part : $(5BE0CD19137E2179)_{16}$

The 80th prime is 409, with the cubic root $(409)^{1/3} = 7.42291412044$. Converting this number to binary with only 64 bits in the fraction part, we get

$$(111.0110\ 1100\ 0100\ 0100 \dots 0111)_2 \rightarrow (7.6C44198C4A475817)_{16}$$

The fraction part: $(6C44198C4A475817)_{16}$

SHA-512 Compression Function

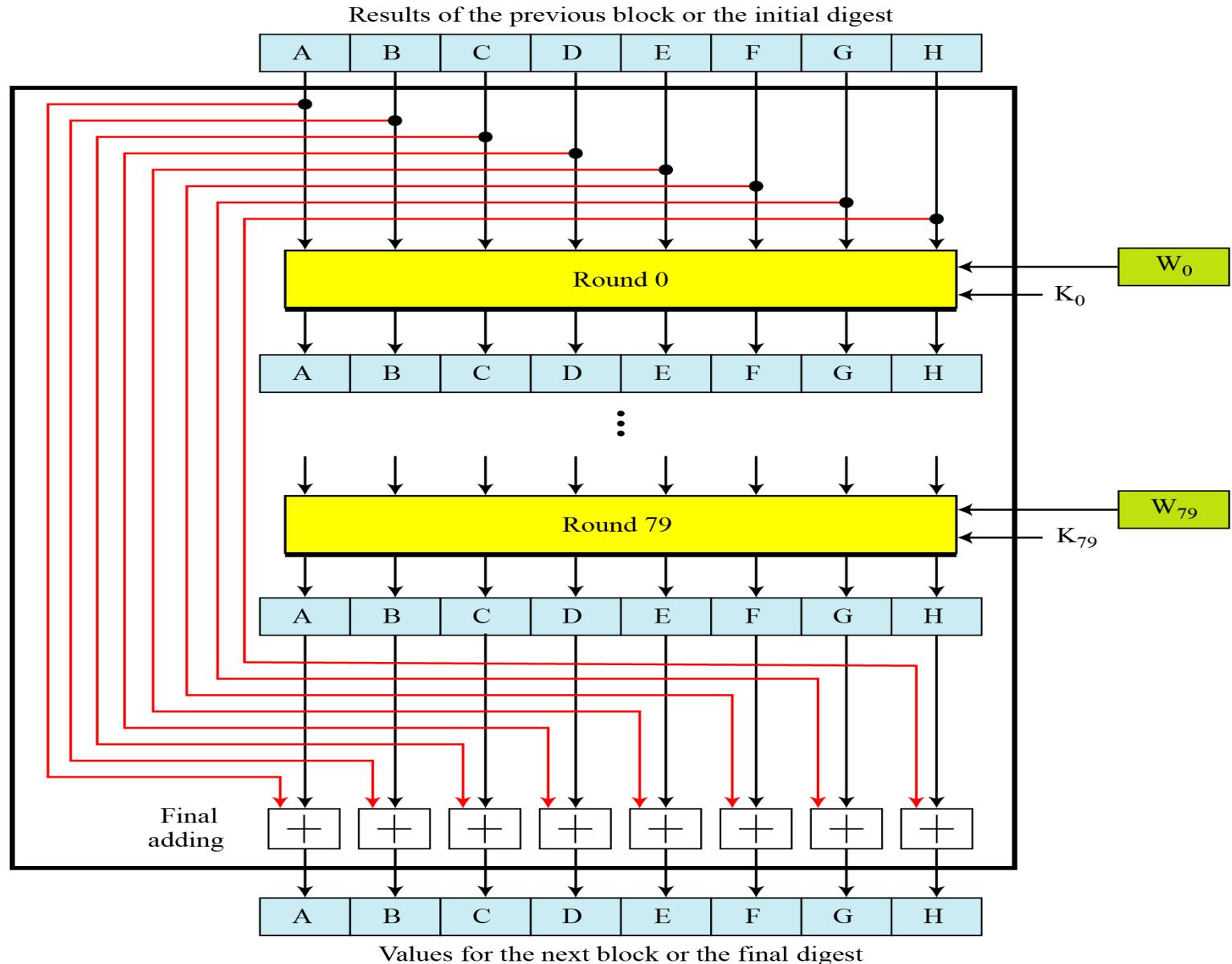


Figure: Compression Function in SHA-512

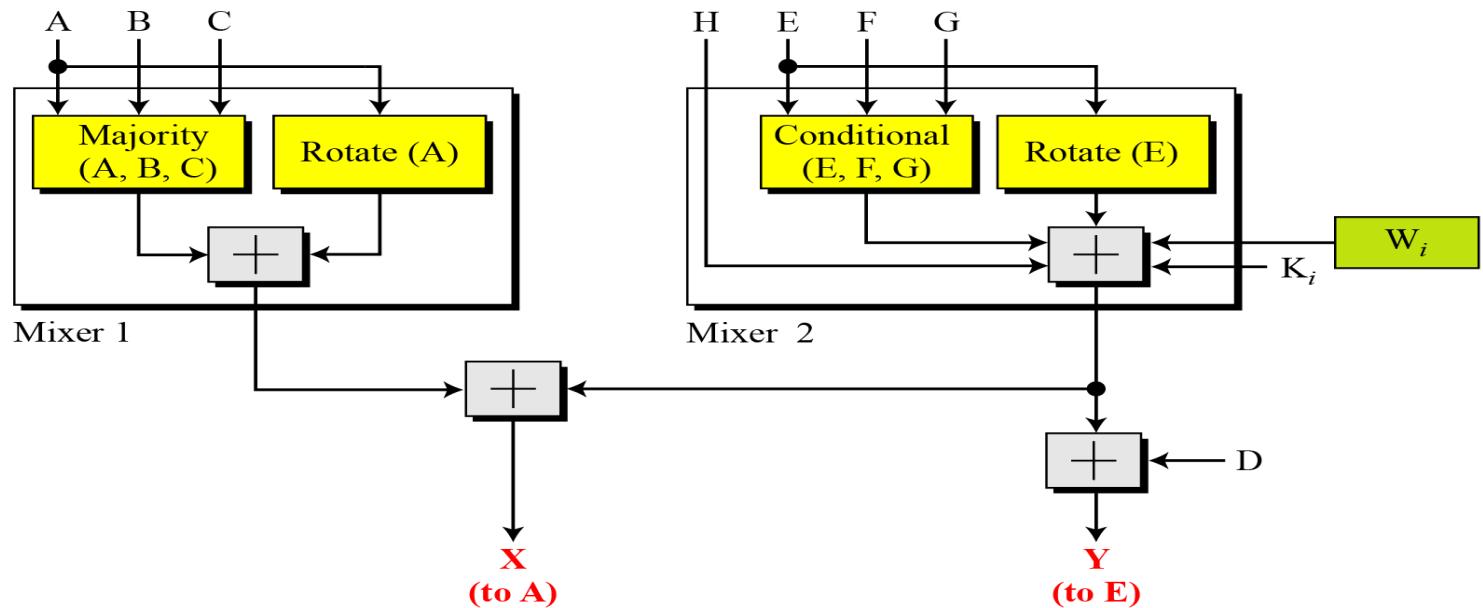
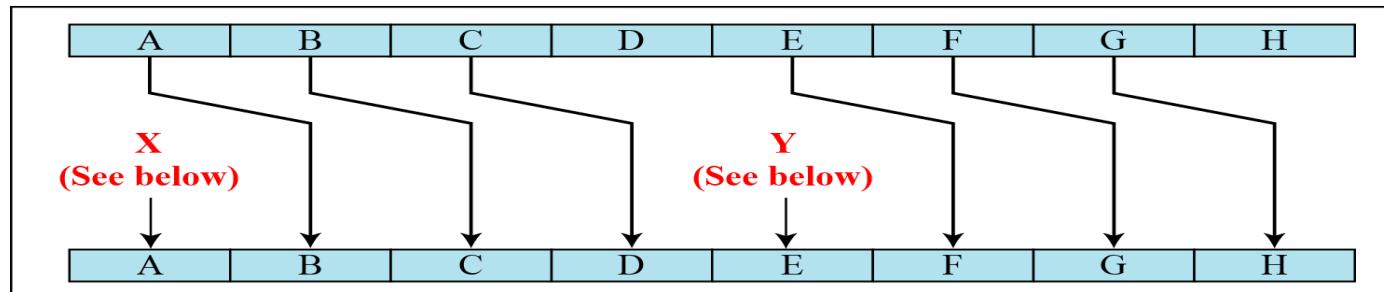
SHA-512 Round constants (K)

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

Figure: List of round constants used in SHA-512

SHA-512 Round Function

Round



Majority (x, y, z)

$$(x \text{ AND } y) \oplus (y \text{ AND } z) \oplus (z \text{ AND } x)$$

Conditional (x, y, z)

$$(x \text{ AND } y) \oplus (\text{NOT } x \text{ AND } z)$$

Rotate (x)

$$\text{RotR}_{28}(x) \oplus \text{RotR}_{34}(x) \oplus \text{RotR}_{39}(x)$$

\oplus addition modulo 2^{64}

$\text{RotR}_i(x)$: Right-rotation of the argument x by i bits

Figure: Structure of each round in SHA-512

SHA-512 Round Function

Majority Function

$$(A_j \text{AND } B_j) \oplus (B_j \text{AND } C_j) \oplus (C_j \text{AND } A_j)$$

Conditional Function

$$(E_j \text{AND } F_j) \oplus (\text{NOT } E_j \text{AND } G_j)$$

Rotate Functions

Rotate (A): $\text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$

Rotate (E): $\text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$

SHA-512 Majority Function calculation

- We apply the Majority function on buffers A, B, and C. If the leftmost hexadecimal digits of these buffers are 0x7, 0xA, and 0xE, respectively, what is the leftmost digit of the result?

Solution

$$(0 \text{ AND } 1) \oplus (1 \text{ AND } 1) \oplus (1 \text{ AND } 0) = 0 \oplus 1 \oplus 0 = 1$$

The digits in binary are 0111, 1010, and 1110.

- a. The first bits are 0, 1, and 1. The majority is 1.
- b. The second bits are 1, 0, and 1. The majority is 1.
- c. The third bits are 1, 1, and 1. The majority is 1.
- d. The fourth bits are 1, 0, and 0. The majority is 0.

The result is **1110**, or **0xE** in hexadecimal.

SHA-512 Conditional Function calculation

- We apply the Conditional function on E, F, and G buffers. If the leftmost hexadecimal digits of these buffers are 0x9, 0xA, and 0xF respectively, what is the leftmost digit of the result?

Solution

$$(1 \text{ AND } 1) \oplus (\text{NOT } 1 \text{ AND } 1) = 1 \oplus 0 = 1$$

The digits in binary are 1001, 1010, and 1111.

- a. The first bits are 1, 1, and 1. The result is F_1 , which is 1.
 - b. The second bits are 0, 0, and 1. The result is G_2 , which is 1.
 - c. The third bits are 0, 1, and 1. The result is G_3 , which is 1.
 - d. The fourth bits are 1, 0, and 1. The result is F_4 , which is 0.
- The result is **1110**, or **0xE** in hexadecimal.

Example using SHA-512

ASCII characters: “abc”, which is equivalent to the following 24-bit binary string:

01100001 01100010 01100011 = 616263 in Hexadecimal

The original length is 24 bits, or a hexadecimal value of 18.
the 1024-bit message block, in hexadecimal, is

6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018

$W_0 = 6162638000000000$
 $W_1 = 0000000000000000$
 $W_2 = 0000000000000000$
 $W_3 = 0000000000000000$
 $W_4 = 0000000000000000$
 $W_{10} = 0000000000000000$
 $W_{11} = 0000000000000000$
 $W_{12} = 0000000000000000$

$W_5 = 0000000000000000$
 $W_6 = 0000000000000000$
 $W_7 = 0000000000000000$
 $W_8 = 0000000000000000$
 $W_9 = 0000000000000000$
 $W_{13} = 0000000000000000$
 $W_{14} = 0000000000000000$
 $W_{15} = 0000000000000018$

Example using SHA-512

The following table shows the initial values of these variables and their values after each of the first two rounds.

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

The process continues through 80 rounds. The output of the final round is

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9
d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

Example using SHA-512

The hash value is then calculated as

$$\begin{aligned} H_{1,7} &= 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f \\ H_{1,6} &= 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e \\ H_{1,5} &= 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd \\ H_{1,4} &= 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8 \\ H_{1,3} &= a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a \\ H_{1,2} &= 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2 \\ H_{1,1} &= bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131 \\ H_{1,0} &= 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba \end{aligned}$$

The resulting 512-bit message digest is

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

SHA-512 Applications

- Used as part of a system to authenticate archival video from the International Criminal Tribunal of the Rwandan genocide.
- Proposed for use in DNSSEC
- are moving to 512-bit SHA-2 for secure password hashing by Unix and Linux vendors

SHA-512 Cryptanalysis

Published in	Year	Attack method	Rounds
New Collision Attacks Against Up To 24-step SHA-2	2008	Deterministic	24/80
Preimages for step-reduced SHA-2	2009	Meet-in-the-middle	42/80
			46/80
Advanced meet-in-the-middle preimage attacks	2010	Meet-in-the-middle	42/80
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family	2011	Biclique	50/80
			57/80
Branching Heuristics in Differential Collision Search with Applications to SHA-512	2014	Heuristic differential	38/80

Summary

- ▶ Hash functions are keyless
 - ▶ Applications for digital signatures and in message authentication codes
- ▶ The three security requirements for hash functions are
 - ▶ one-wayness and collision resistance
- ▶ MD5 and SHA-0 is insecure
- ▶ Serious security weaknesses have been found in SHA-1
 - ▶ should be phased out
 - ▶ SHA-2 appears to be secure
 - ▶ May use SHA-512 and use the first 256 bytes
- ▶ SHA-3 (Secure Hash Algorithm 3) is the latest member of the [Secure Hash Algorithm](#) family of standards, released by [NIST](#) on August 5, 2015

Authentication Applications

Outline

- Security Concerns
- Kerberos
- X.509 Authentication Service
- Recommended reading and Web Sites

KERBEROS



In Greek mythology, a many headed dog, the guardian of the entrance of Hades

KERBEROS

- Users wish to access services on servers.
- Three threats exist:
 - User pretend to be another user.
 - User alter the network address of a workstation.
 - User eavesdrop on exchanges and use a replay attack.

KERBEROS

- Provides a centralized authentication server to authenticate users to servers and servers to users.
- Relies on conventional encryption, making no use of public-key encryption
- Two versions: version 4 and 5
- Version 4 makes use of DES

Kerberos Version 4

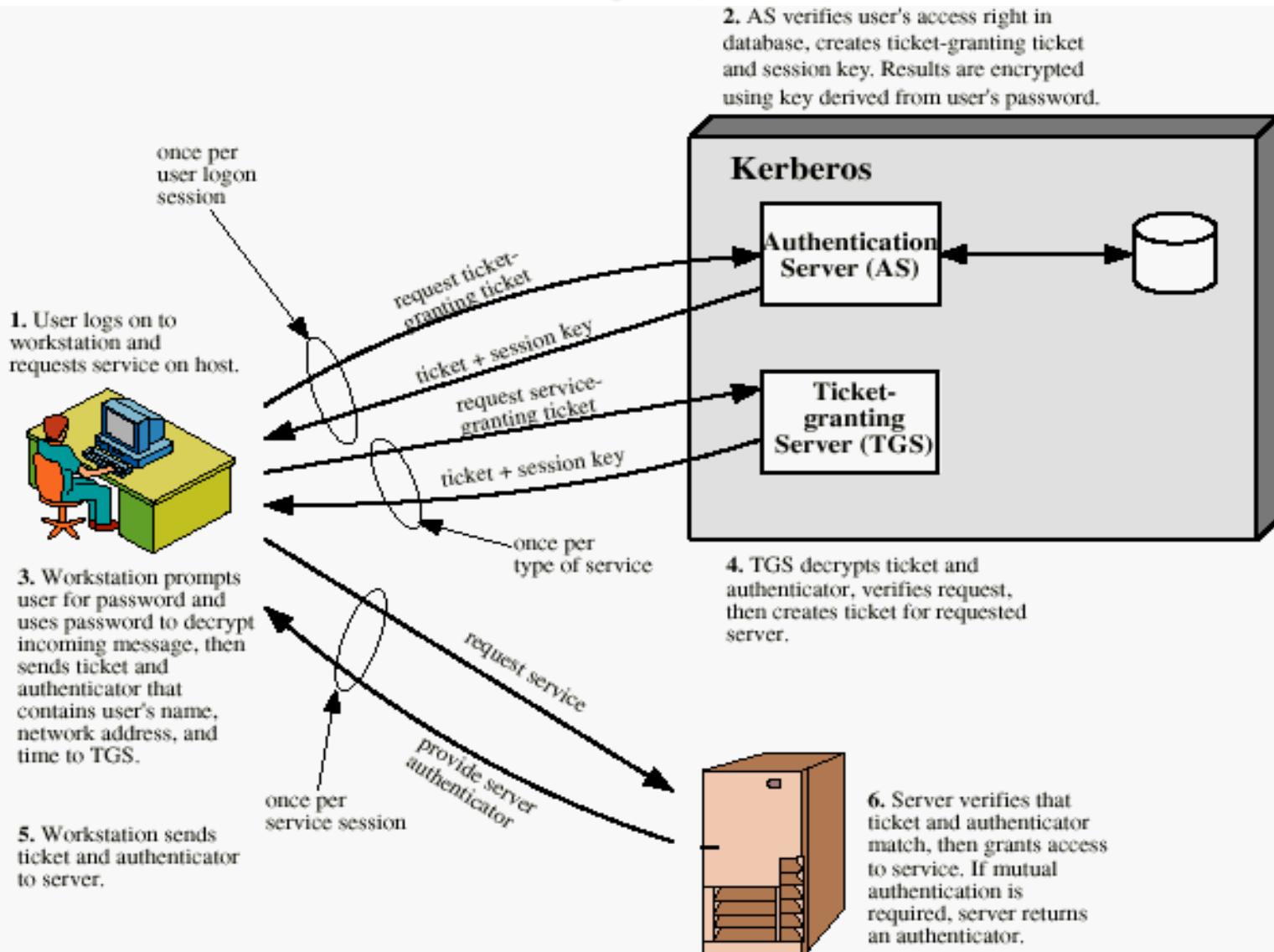
- Terms:
 - C = Client
 - AS = authentication server
 - V = server
 - ID_c = identifier of user on C
 - ID_v = identifier of V
 - P_c = password of user on C
 - AD_c = network address of C
 - K_v = secret encryption key shared by AS and V
 - TS = timestamp
 - $\|$ = concatenation

A Simple Authentication Dialogue

- | | |
|-------------------------|-------------------------------------|
| (1) $C \rightarrow AS:$ | $ID_c \parallel P_c \parallel ID_v$ |
| (2) $AS \rightarrow C:$ | Ticket |
| (3) $C \rightarrow V:$ | $ID_c \parallel \text{Ticket}$ |

$\text{Ticket} = E_{K_v}[ID_c \parallel P_c \parallel ID_v]$

Overview of Kerberos



Request for Service in Another Realm

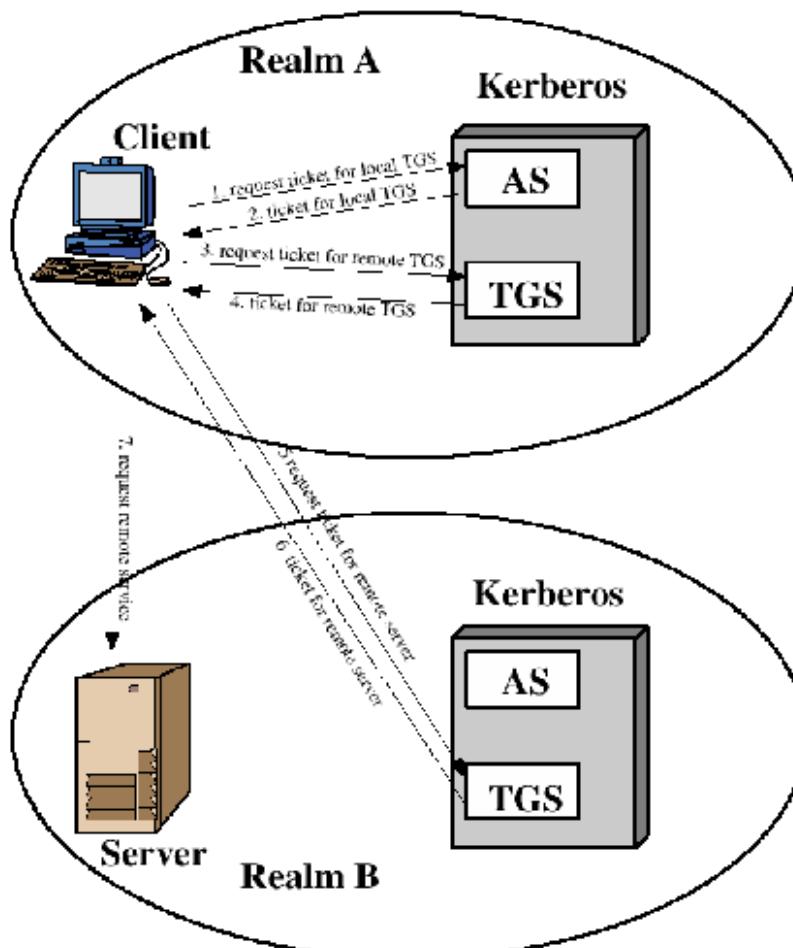


Figure 4.2 Request for Service in Another Realm

Difference Between Version 4 and 5

- Encryption system dependence (V.4 DES)
- Internet protocol dependence
- Message byte ordering
- Ticket lifetime
- Authentication forwarding
- Interrealm authentication

Kerberos Encryption Techniques

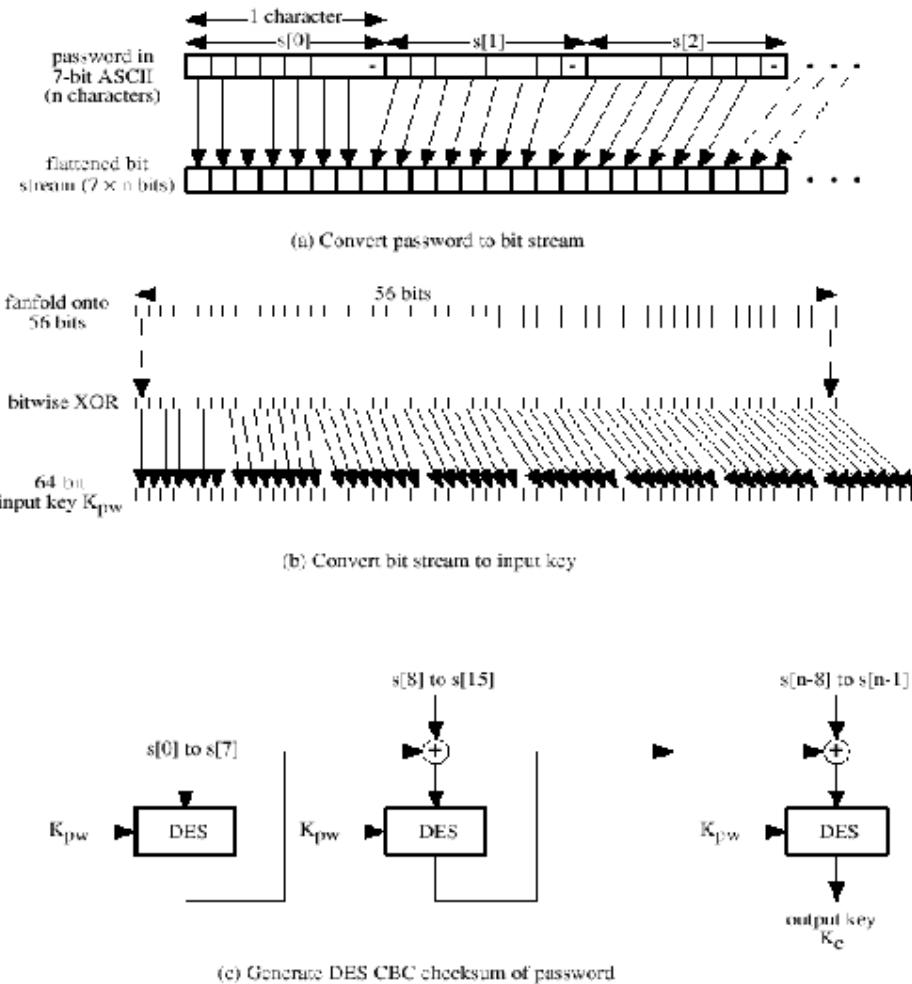


Figure 4.6 Generation of Encryption Key from Password

PCBC Mode

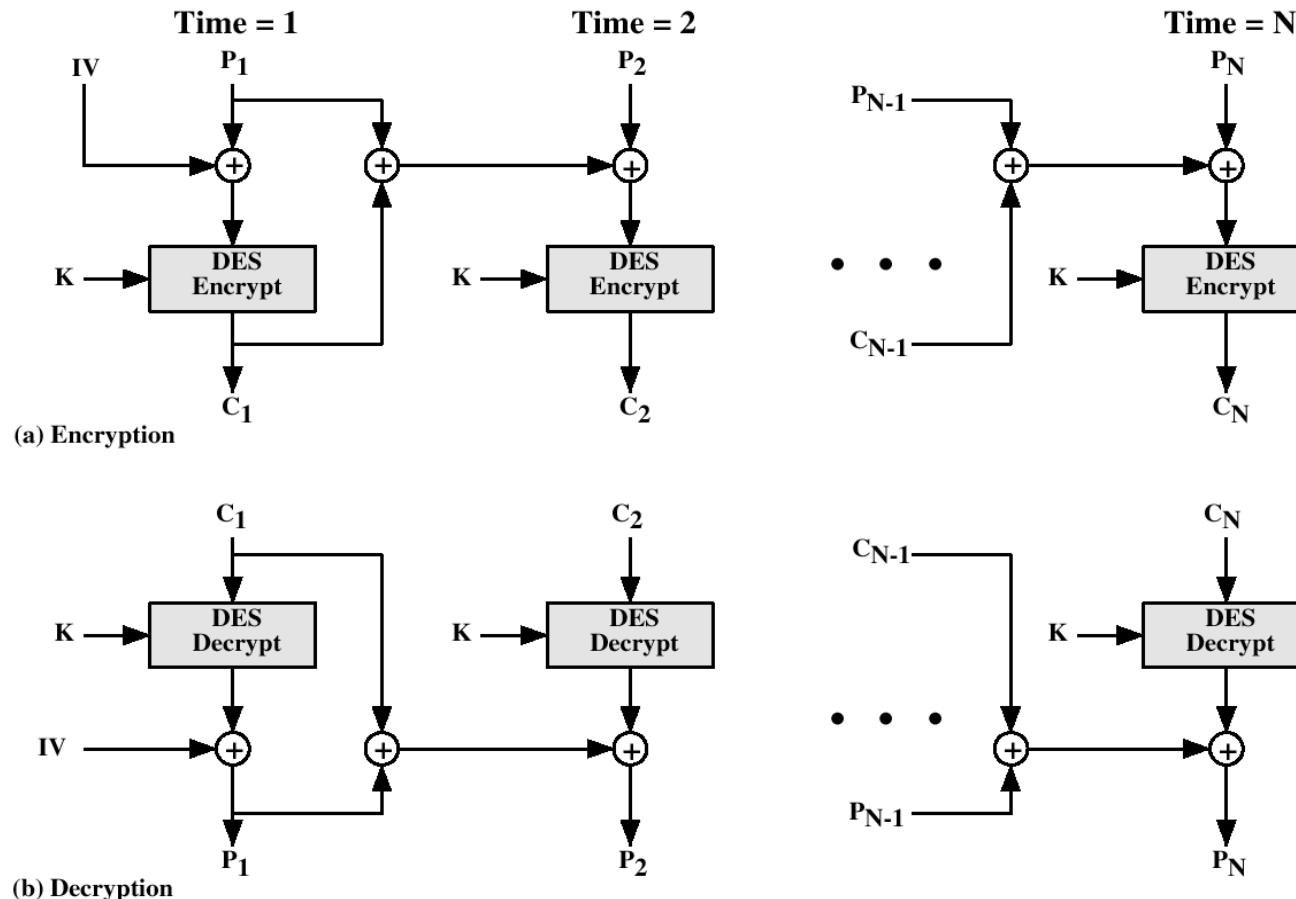


Figure 4.7 Propagating Cipher Block Chaining (PCBC) Mode

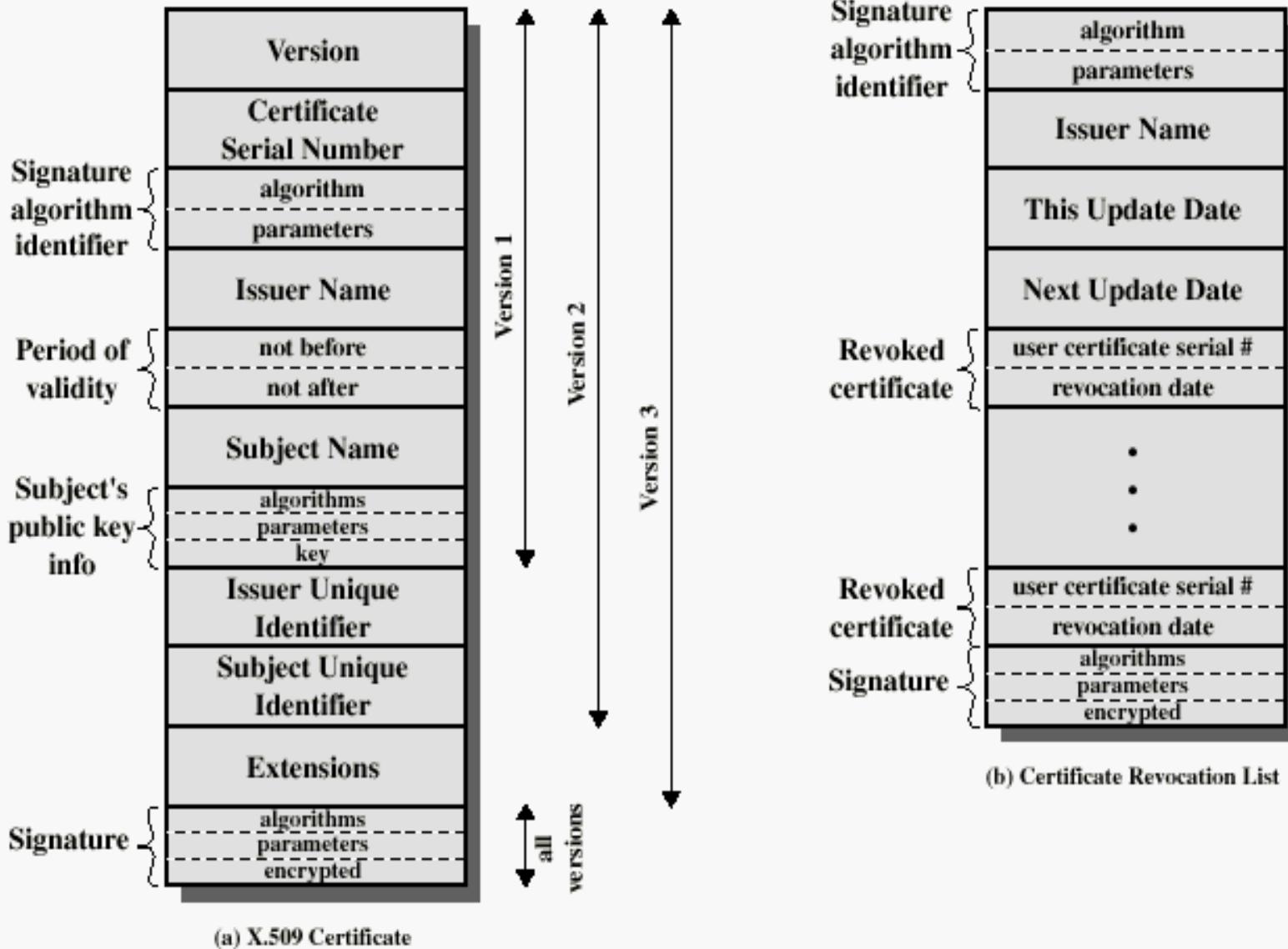
Kerberos - in practice

- Currently have two Kerberos versions:
- 4 : restricted to a single realm
- 5 : allows inter-realm authentication, in beta test
- Kerberos v5 is an Internet standard
- specified in RFC1510, and used by many utilities
- To use Kerberos:
 - need to have a KDC on your network
 - need to have Kerberised applications running on all participating systems
 - major problem - US export restrictions
 - Kerberos cannot be directly distributed outside the US in source format (& binary versions must obscure crypto routine entry points and have no encryption)
 - else crypto libraries must be reimplemented locally

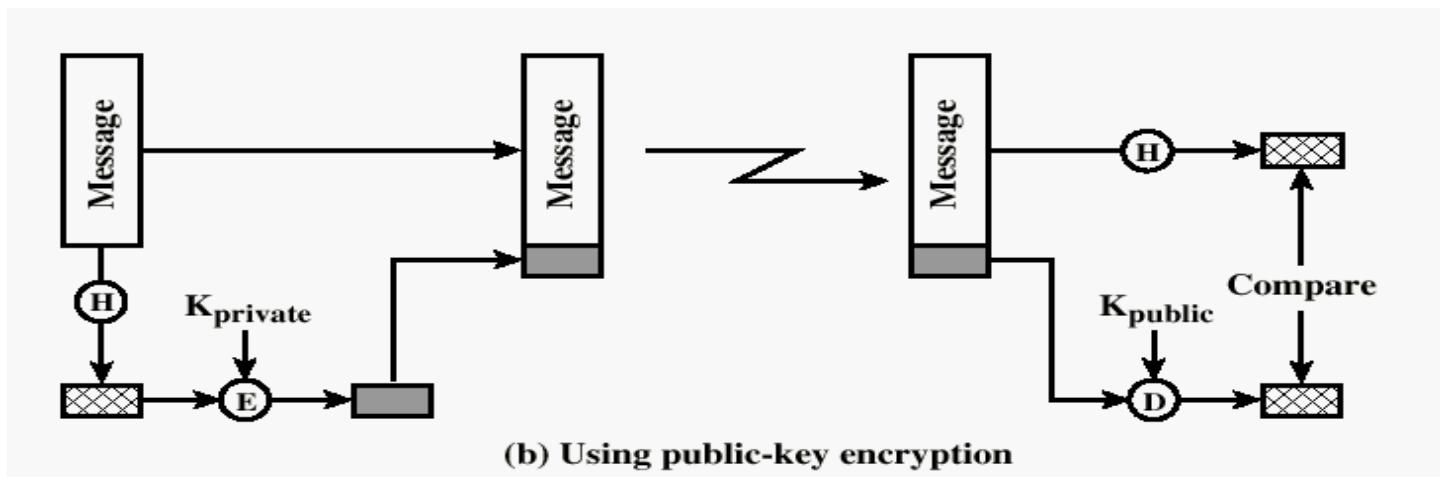
X.509 Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in S/MIME, IP Security, SSL/TLS and SET.
- RSA is recommended to use.

X.509 Formats



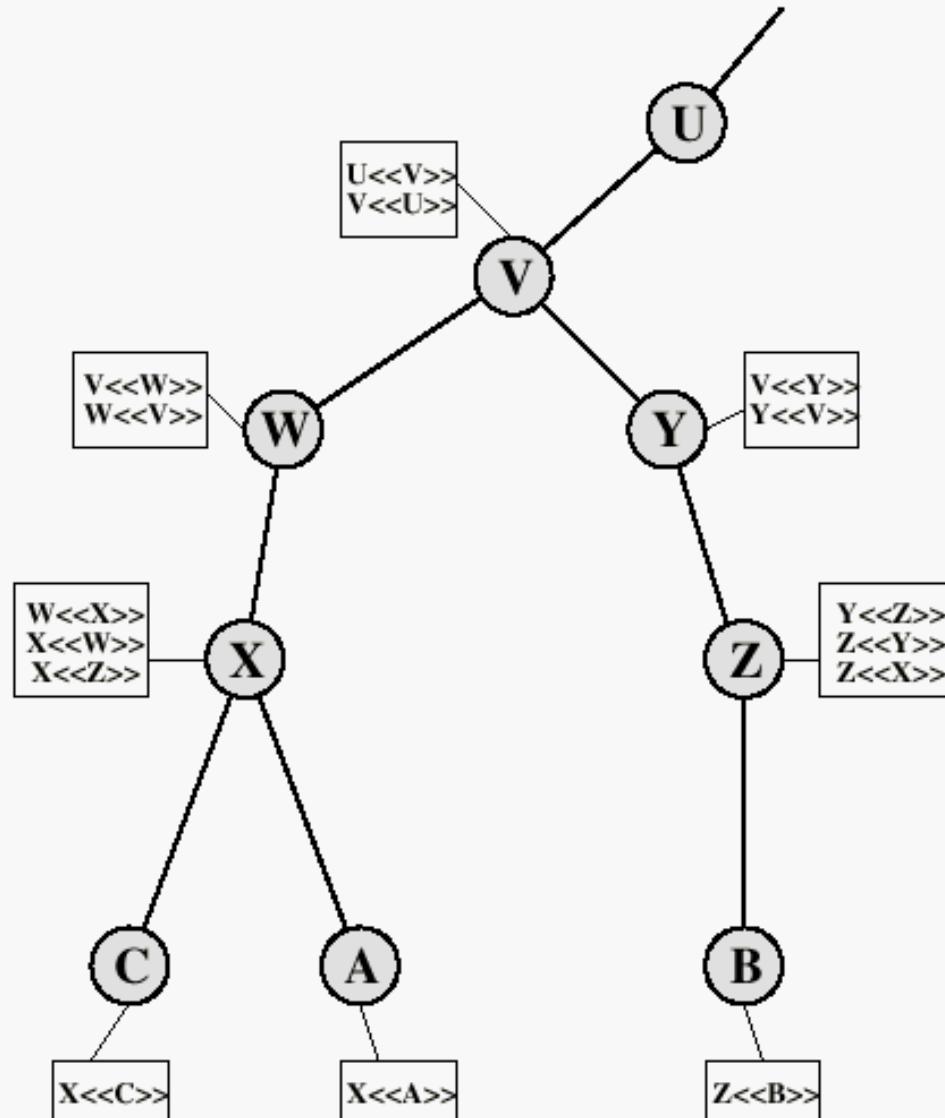
Typical Digital Signature Approach



Obtaining a User's Certificate

- Characteristics of certificates generated by CA:
 - Any user with access to the public key of the CA can recover the user public key that was certified.
 - No part other than the CA can modify the certificate without this being detected.

X.509 CA Hierarchy



Revocation of Certificates

- Reasons for revocation:
 - The user's secret key is assumed to be compromised.
 - The user is no longer certified by this CA.
 - The CA's certificate is assumed to be compromised.

Authentication Procedures

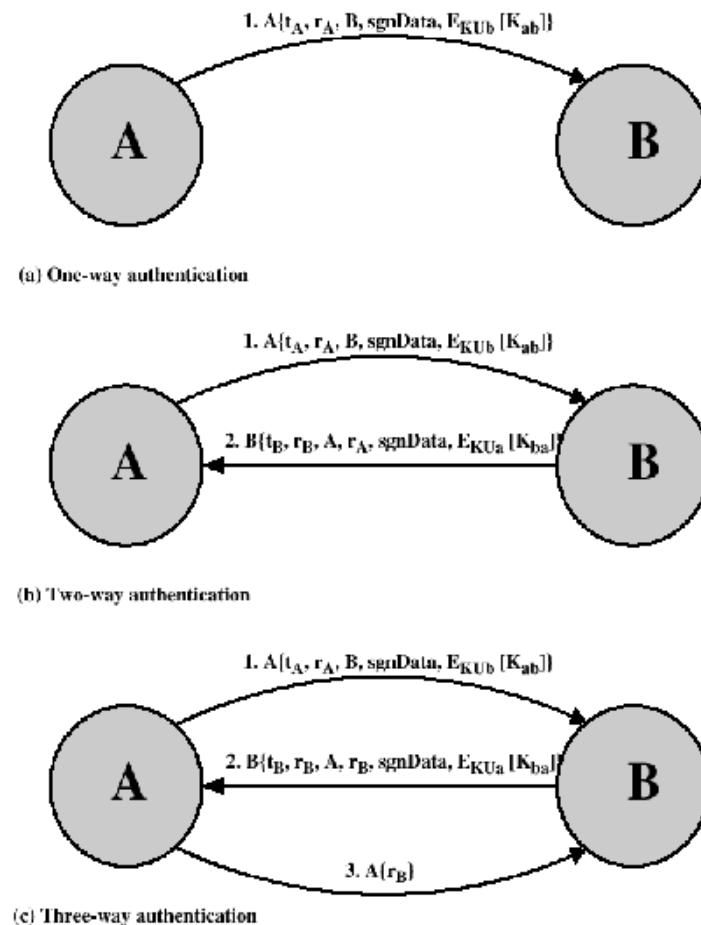


Figure 4.5 X.509 Strong Authentication Procedures

Recommended Reading and WEB Sites

- www.whatis.com (search for kerberos)
- Bryant, W. Designing an Authentication System: A Dialogue in Four Scenes.
<http://web.mit.edu/kerberos/www/dialogue.html>
- Kohl, J.; Neuman, B. "The Evolution of the Kerberos Authentication Service"
<http://web.mit.edu/kerberos/www/papers.html>
- <http://www.isi.edu/gost/info/kerberos/>

Cryptography & Network Security

M.K.Chavan

Computer Science & Engineering Department
WCE, Sangli.

Module-V Network Security

Internet Security Protocols: SSL/TLS

Outline

- Web Security and SSL

Web Security

- Web now widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
 - integrity
 - confidentiality
 - denial of service
 - authentication
- need added security mechanisms

Two protocols are dominant today for providing security at the transport layer: the Secure Sockets Layer (SSL) Protocol and the Transport Layer Security (TLS) Protocol. The latter is actually an IETF version of the former.

Topics discussed in this section:

SSL Services

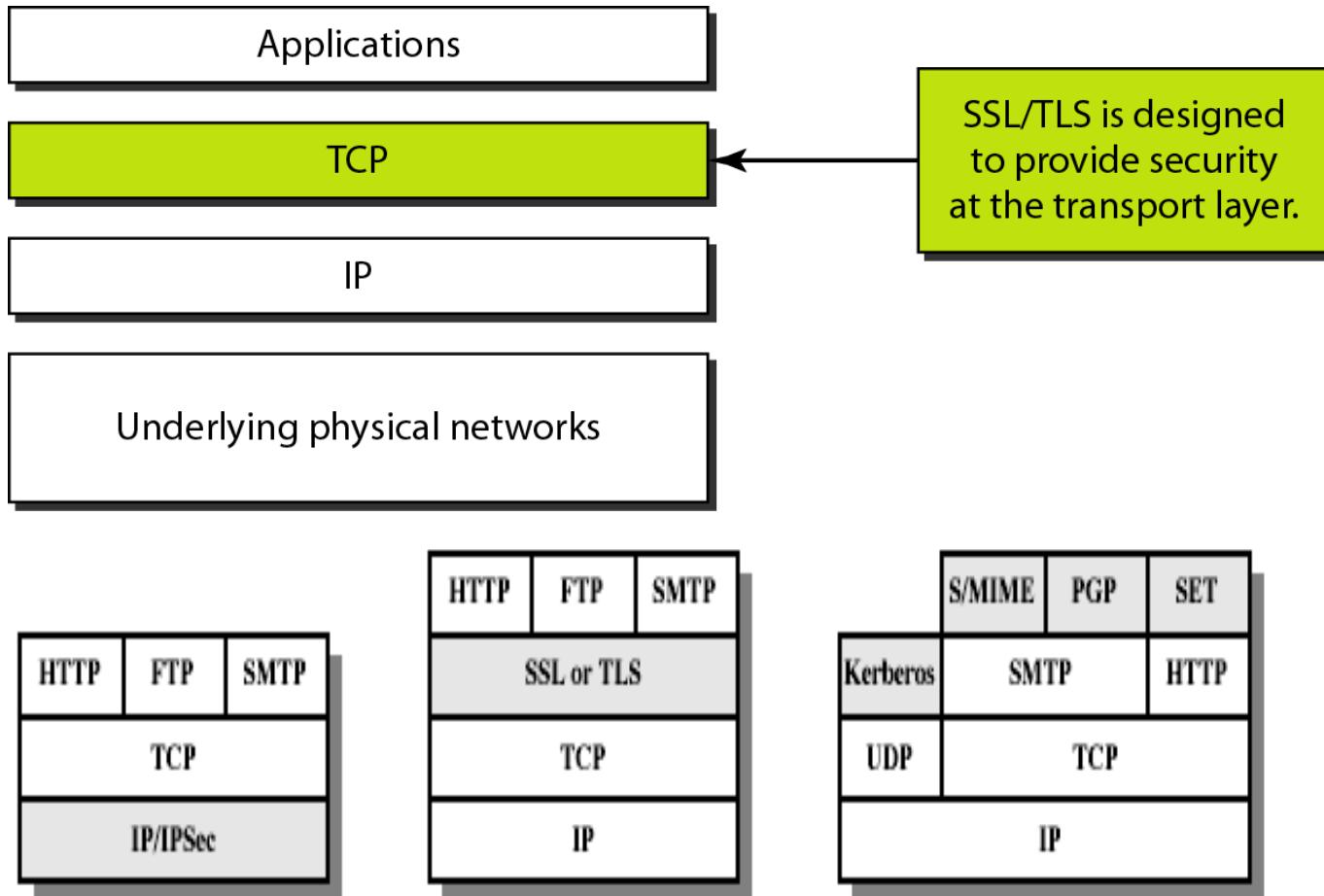
Security Parameters

Sessions and Connections

Four Protocols

Transport Layer Security

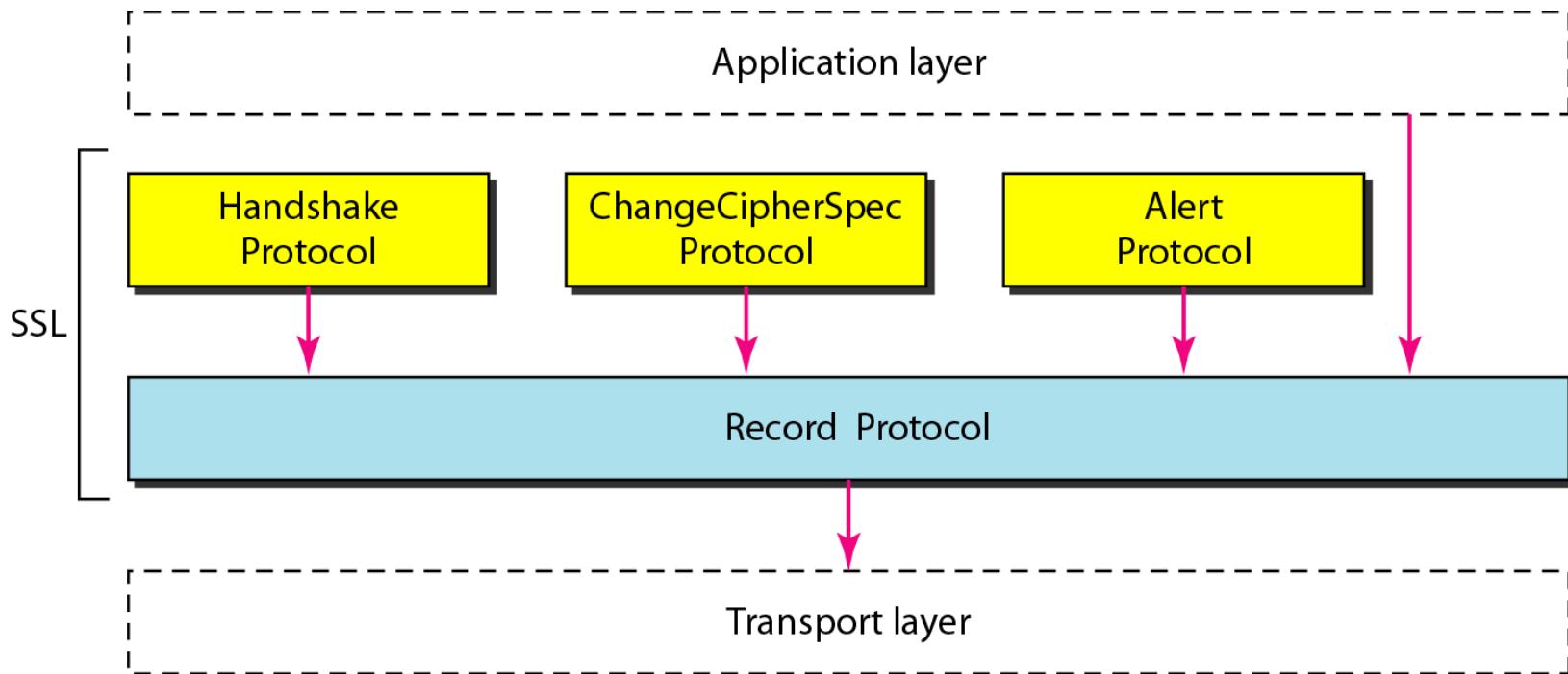
Web Traffic Security Approaches



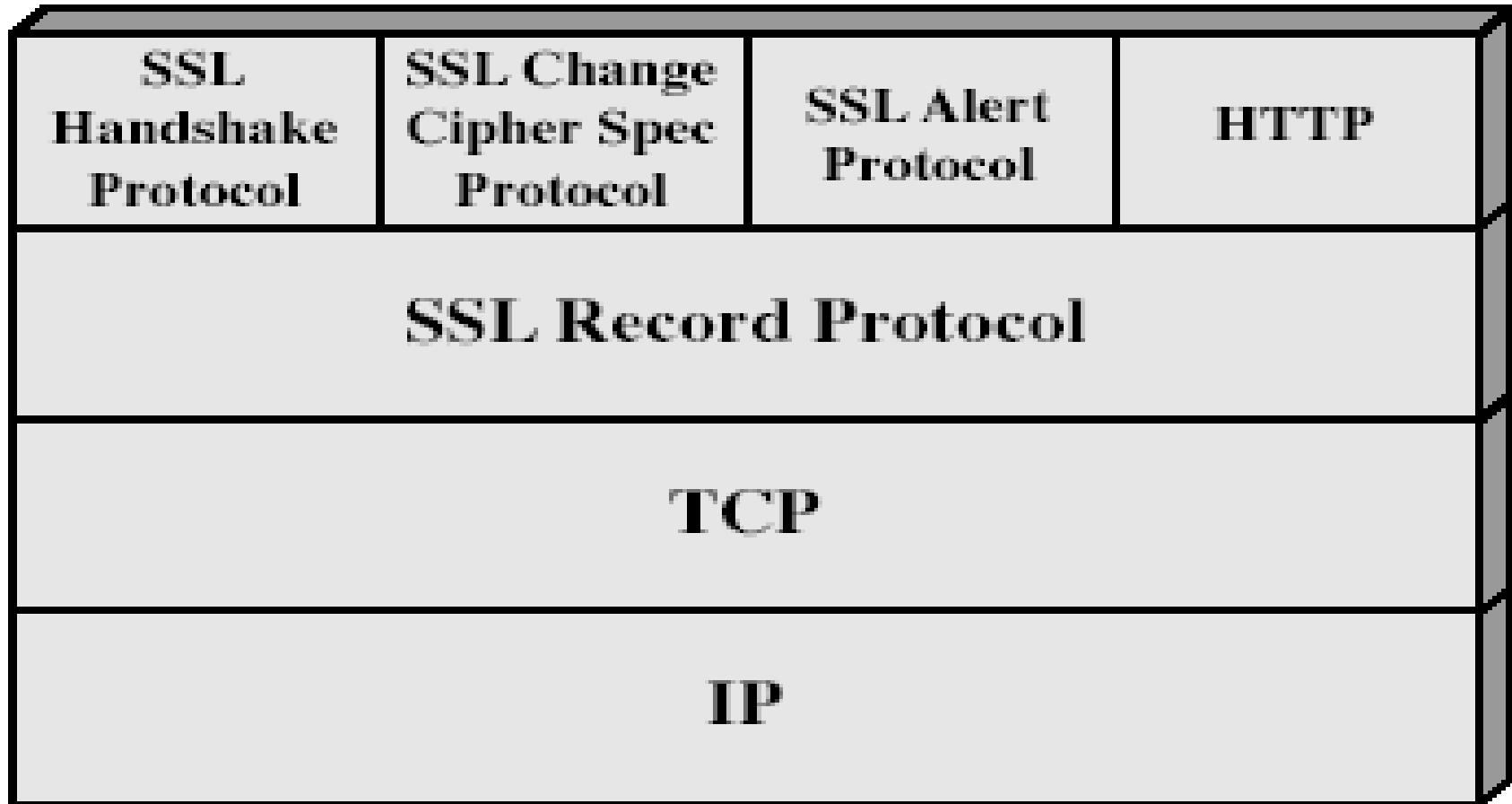
SSL (Secure Socket Layer)

- transport layer security service
- originally developed by Netscape
- version 3 designed with public input
- subsequently became Internet standard known as TLS (Transport Layer Security) (viewed as SSLv3.1)
- uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

Figure 32.16 Four SSL protocols



SSL Architecture



SSL (Secure Socket Layer)

- SSL Record Protocol: basic security services to higher-layer protocols (HTTP)
- Handshake, Change Cipher Spec, Alert: management of SSL exchanges

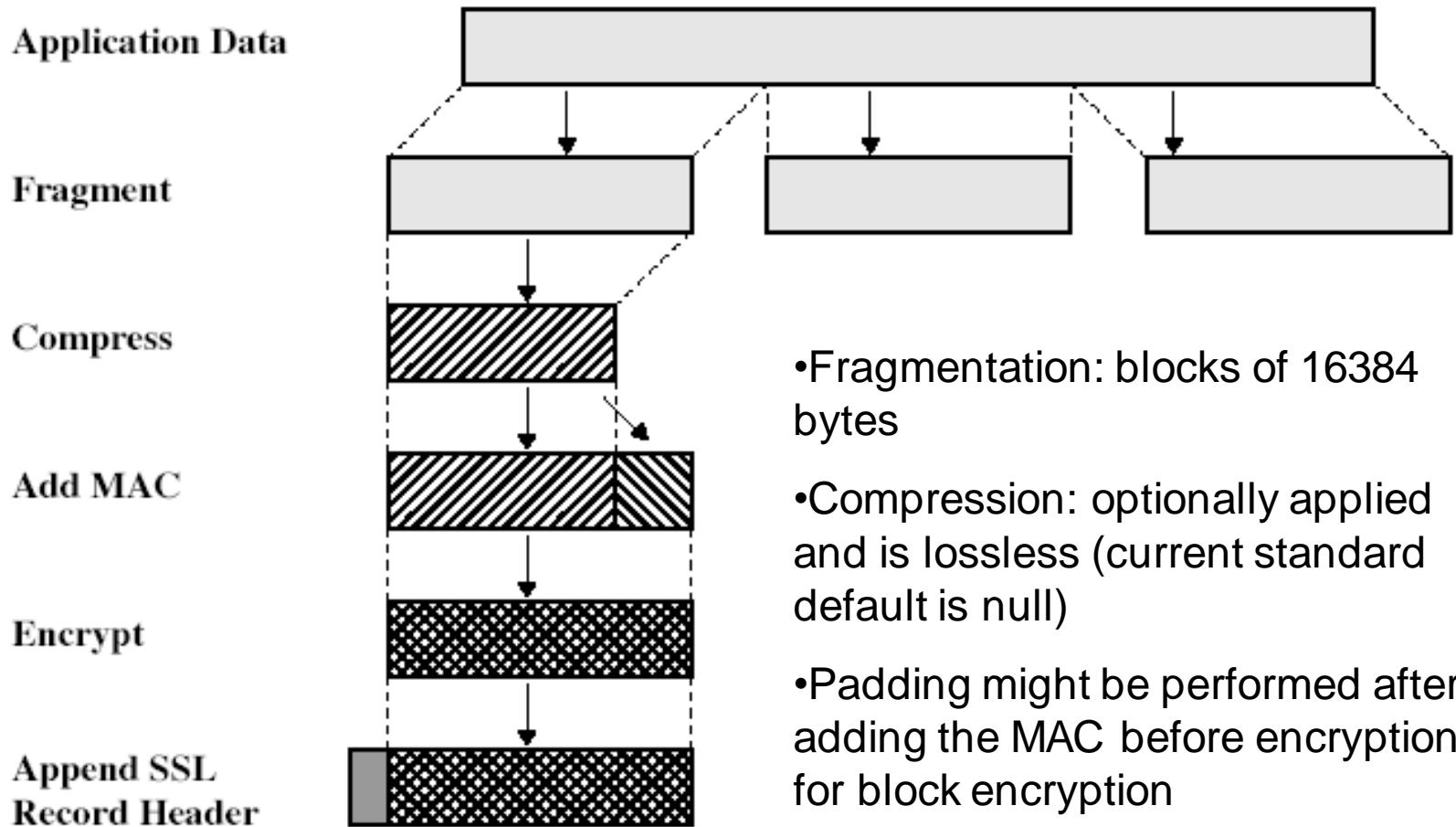
SSL Session state

- **Session Identifier**: byte sequence chosen by server
- **Peer certificate** X509.v3 certificate
- **Compression method**: compress prior to encryption
- **Cipher Spec**: data encryption algorithm and hash algorithm
- **Master Secret**: 48 byte secret
- **Is Resumable**: can session be used to initiate new connections

SSL Record Protocol

- **confidentiality**
 - using symmetric encryption with a shared secret key defined by Handshake Protocol
 - IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
 - message is compressed before encryption
- **message integrity**
 - using a MAC with shared secret key
 - similar to HMAC but with different padding

SSL Record Protocol



SSL Change Cipher Spec Protocol

- one of 3 SSL specific protocols which use the SSL Record protocol
- a single message
- causes pending state to become current
- hence updating the cipher suite in use

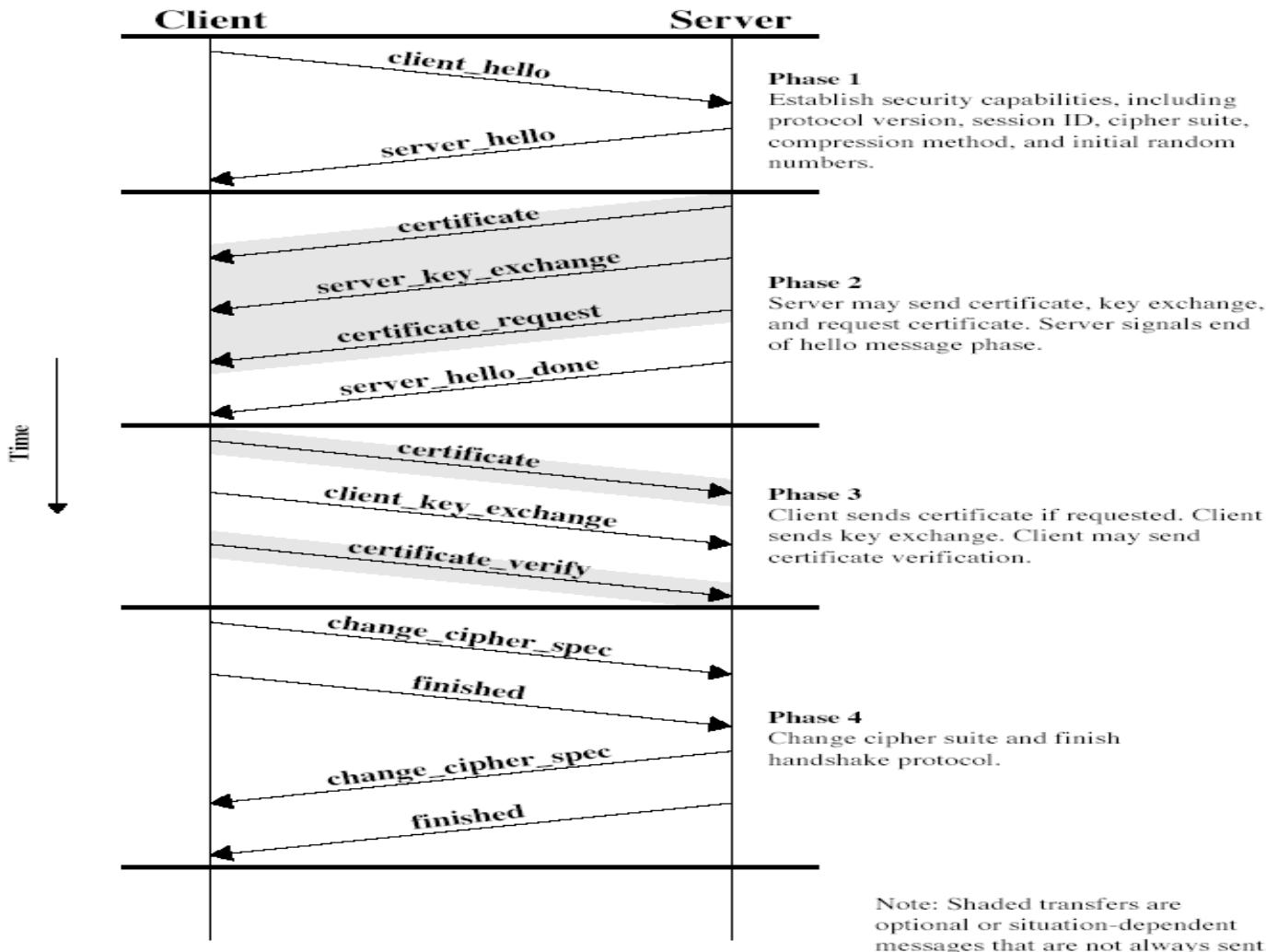
SSL Alert Protocol

- conveys SSL-related alerts to peer entity
- severity
 - warning or fatal
- specific alert
 - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
 - close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Alert messages are compressed & encrypted like all SSL data

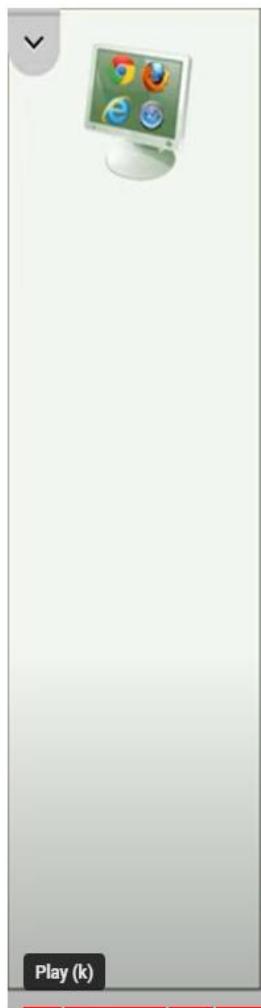
SSL Handshake Protocol

- allows server & client to:
 - authenticate each other
 - to negotiate encryption & MAC algorithms
 - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
 - Establish Security Capabilities
 - Server Authentication and Key Exchange
 - Client Authentication and Key Exchange
 - Finish

SSL Handshake Protocol



Client Hello Message



- Handshake: **Client Hello**
 - Version
 - Highest version of TLS/SSL **Client** Supports
 - **Random Number** – 32 bytes / 256 bits
 - Timestamp encoded in first four bytes
 - **Session ID** – 8 bytes / 32 bits
 - 00000... All 0's in initial Client Hello
 - **Cipher Suites**
 - List of Cipher Suites **Client** supports
 - **Extensions**
 - Optional additional features added to TLS/SSL



Server Hello Message

- Version
 - Lower of version suggested by client and highest supported by server
- Random
 - 32-bit timestamp and 28 bytes by a secure random number generator
- Session ID
 - Nonzero by client: same used by server
 - Zero: value for a new session
- CipherSuite
 - Single cipher suite selected by server from sent by client
- Compression Method
 - Compression method selected by server

Key Exchange Methods

- RSA
 - Secret key encrypted with receiver's RSA public key
- Fixed Diffie-Hellman
 - Public key certificate contains Diffie-Hellman public key parameters signed by certificate authority (CA)
 - Fixed secret key between two peers
- Ephemeral Diffie-Hellman
 - Diffie-Hellman public keys are exchanged, signed using sender's private RSA or DSS key
 - Certificates used to authenticate public keys
 - Temporary authenticated secret key
- Anonymous Diffie-Hellman
- Fortezza

Table 32.3 *SSL cipher suite list*

<i>Cipher Suite</i>	<i>Key Exchange Algorithm</i>	<i>Encryption Algorithm</i>	<i>Hash Algorithm</i>
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4_128	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4_128	SHA
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA_CBC	SHA
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4_128	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES_CBC	SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES_EDE_CBC	SHA

Table 32.3 SSL cipher suite list (*continued*)

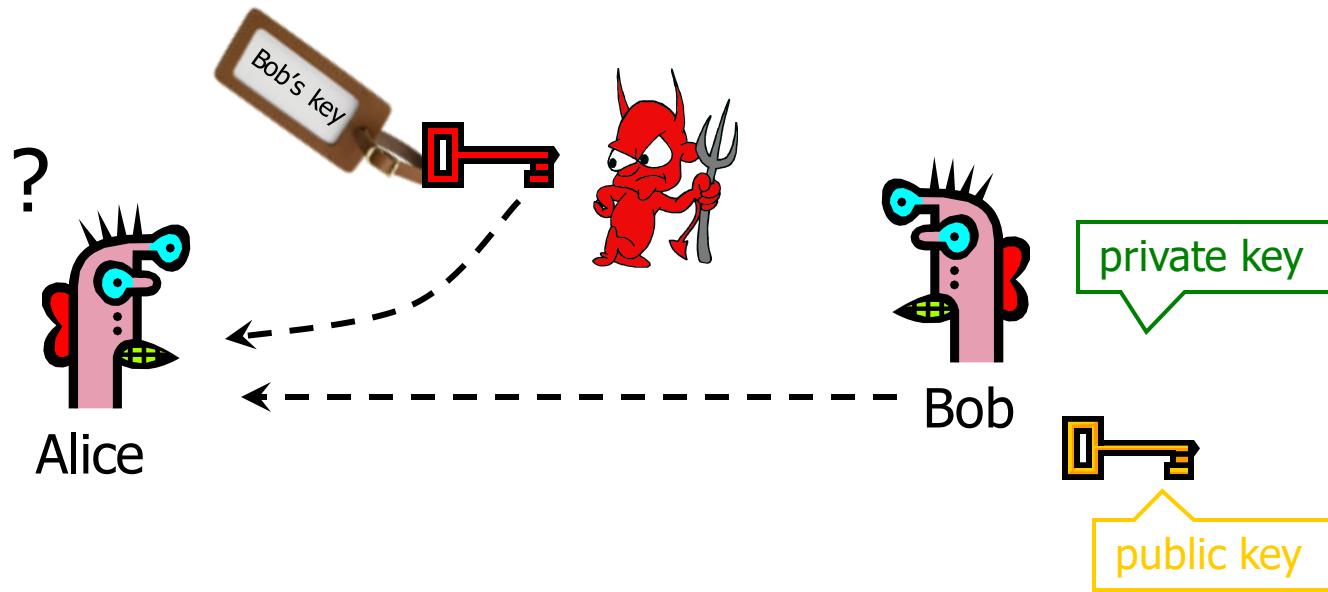
<i>Cipher Suite</i>	<i>Key Exchange Algorithm</i>	<i>Encryption Algorithm</i>	<i>Hash Algorithm</i>
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES_CBC	SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE_CBC	SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES_CBC	SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES_EDE_CBC	SHA
SSL_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES_CBC	SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE_CBC	SHA
SSL_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES_CBC	SHA
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE_CBC	SHA
SSL_FORTEZZA_DMS_WITH_NULL_SHA	FORTEZZA_DMS	NULL	SHA
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	FORTEZZA_DMS	FORTEZZA_CBC	SHA
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	FORTEZZA_DMS	RC4_128	SHA

TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- with minor differences
 - in record format version number
 - uses HMAC for MAC
 - a pseudo-random function expands secrets
 - has additional alert codes
 - some changes in supported ciphers
 - changes in certificate negotiations
 - changes in use of padding

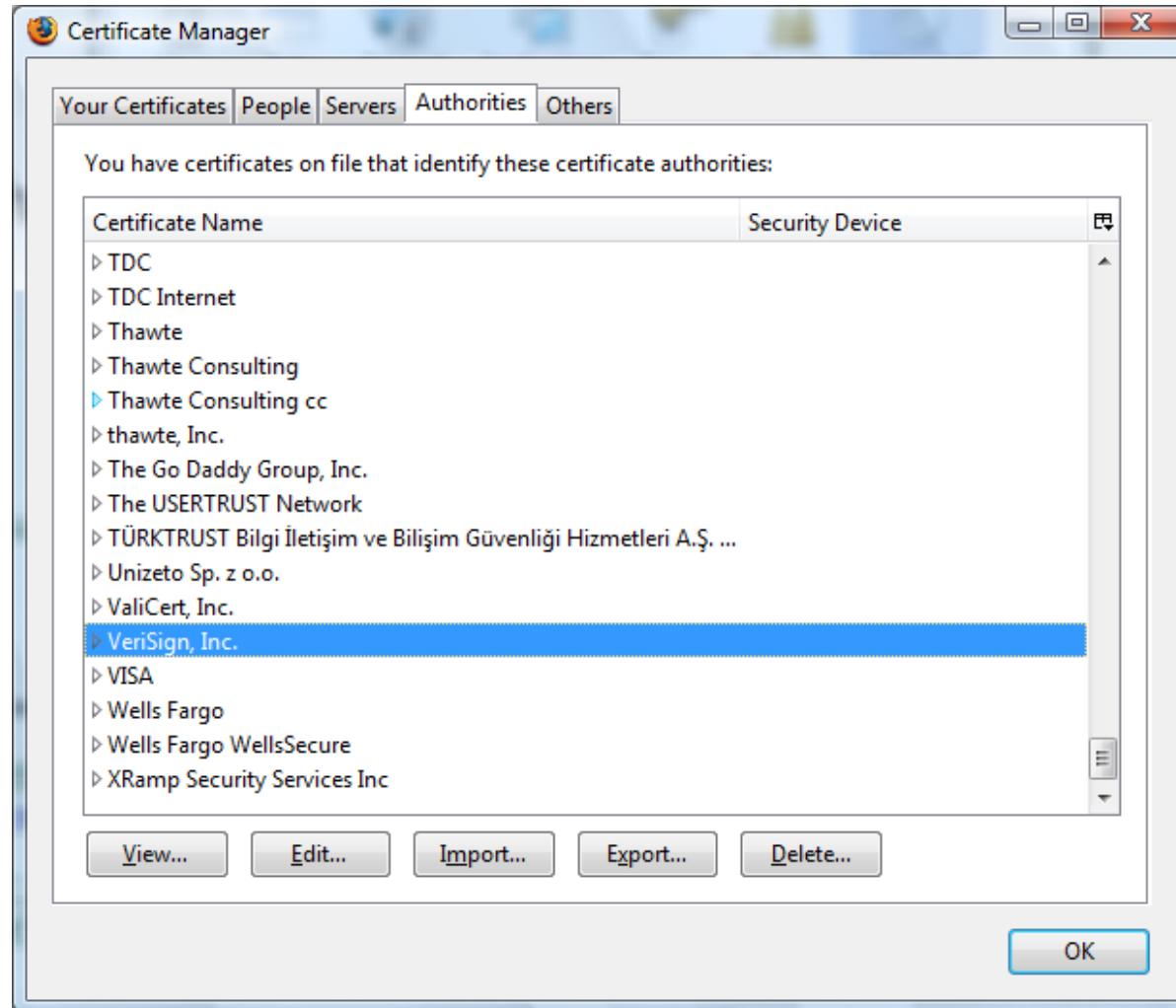
Digital Certificate

Authenticity of Public Keys



Problem: How does Alice know that the public key she received is really Bob's public key?

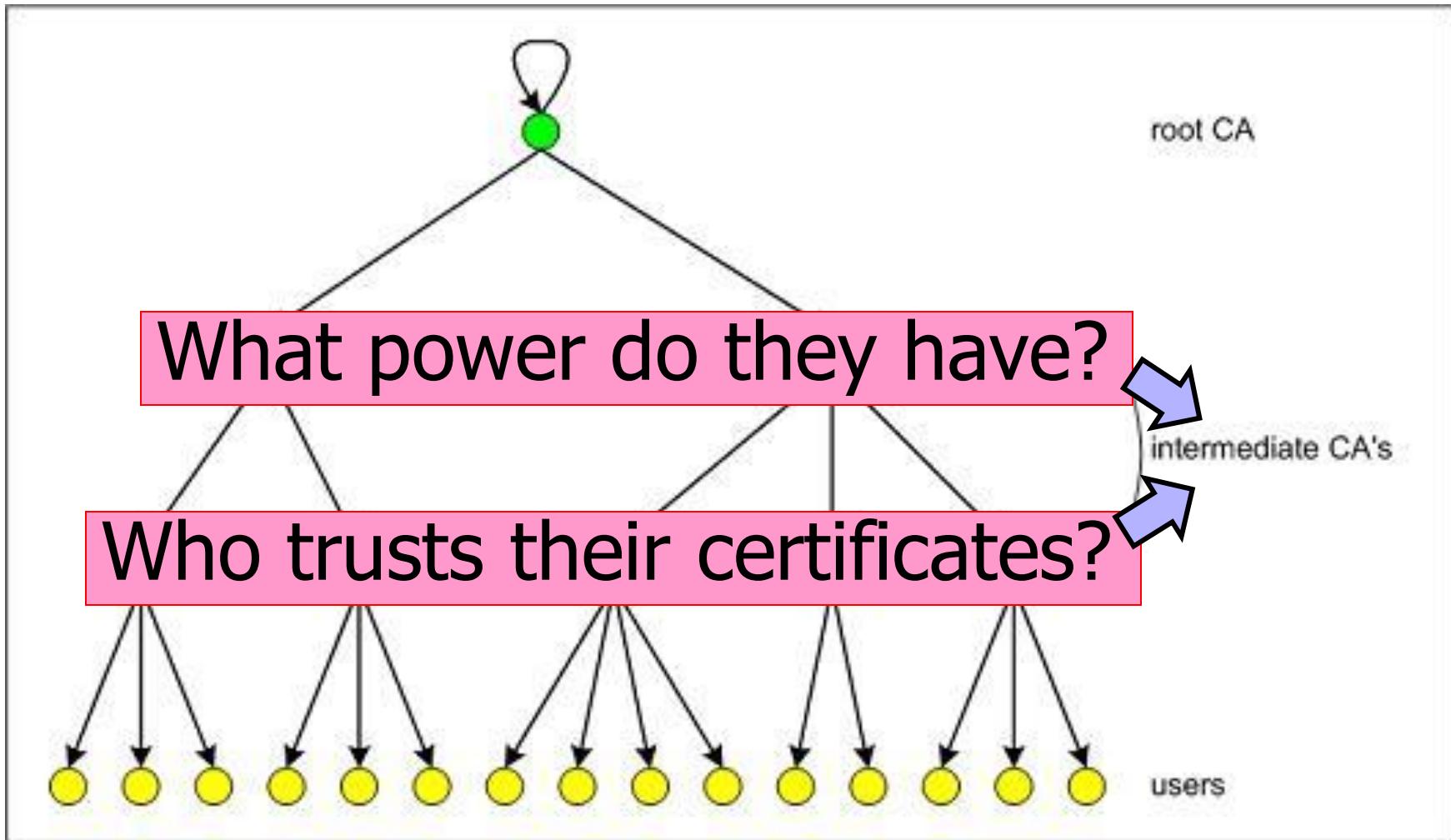
Trusted Certificate Authorities



CA Hierarchy

- Browsers, operating systems, etc. have trusted root certificate authorities
 - Firefox 3 includes certificates of 135 trusted root CAs
- A Root CA signs certificates for intermediate CAs, they sign certificates for lower-level CAs, etc.
 - Certificate “chain of trust”
 - $\text{sig}_{\text{Verisign}}(\text{"UT Austin"}, \text{PK}_{\text{UT}})$, $\text{sig}_{\text{UT}}(\text{"Vitaly S."}, \text{PK}_{\text{Vitaly}})$
- CA is responsible for verifying the identities of certificate requestors, domain ownership

Certificate Hierarchy



Example of a Certificate

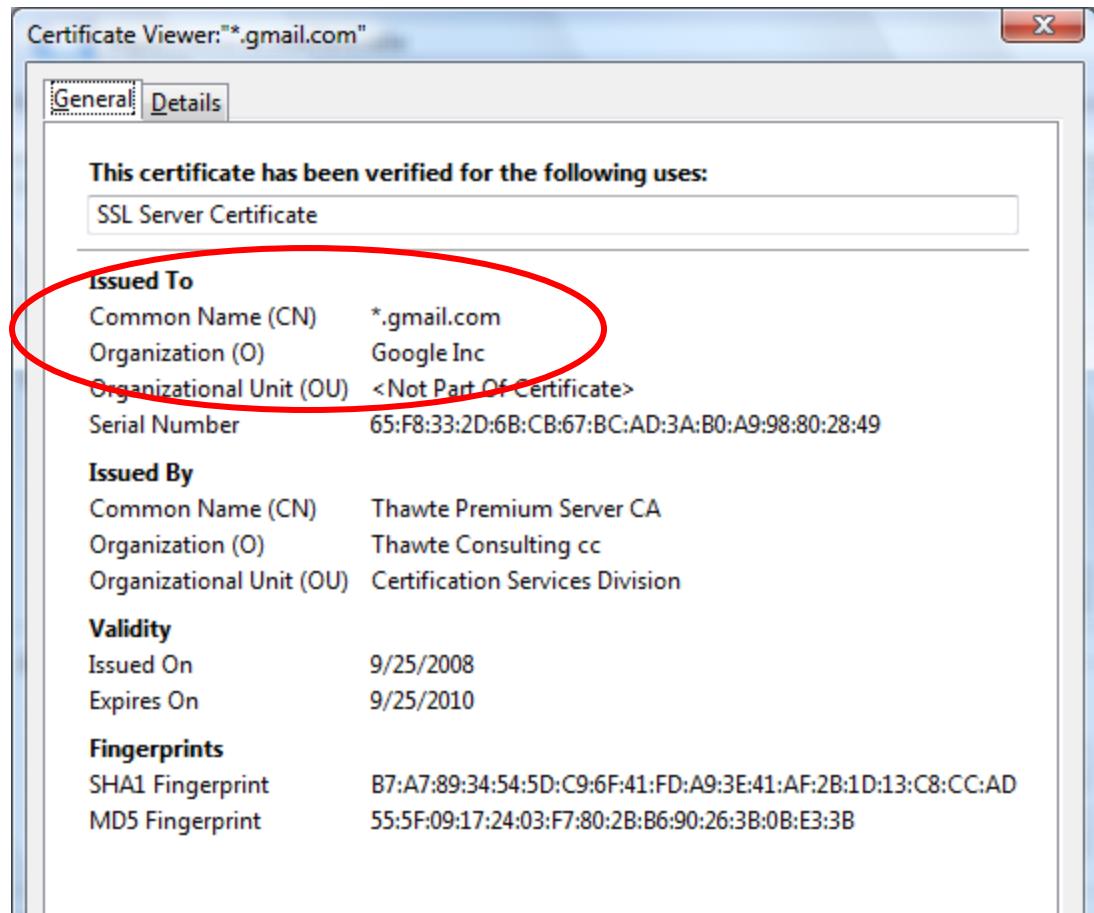
Important fields

- Certificate Signature Algorithm
- Issuer
- Validity
 - Not Before
 - Not After
- Subject
- Subject Public Key Info
 - Subject Public Key Algorithm
 - Subject's Public Key
- Extensions

Field Value

Modulus (1024 bits):

```
ac 73 14 97 b4 10 a3 aa f4 c1 15 ed cf 92 f3 9a  
97 26 9a cf 1b e4 1b dc d2 c9 37 2f d2 e6 07 1d  
ad b2 3e f7 8c 2f fa a1 b7 9e e3 54 40 34 3f b9  
e2 1c 12 8a 30 6b 0c fa 30 6a 01 61 e9 7c b1 98  
2d 0d c6 38 03 b4 55 33 7f 10 40 45 c5 c3 e4 d6  
6b 9c 0d d0 8e 4f 39 0d 2b d2 e9 88 cb 2d 21 a3  
f1 84 61 3c 3a aa 80 18 27 e6 7e f7 b8 6a 0a 75  
e1 bb 14 72 95 cb 64 78 06 84 81 eb 7b 07 8d 49
```

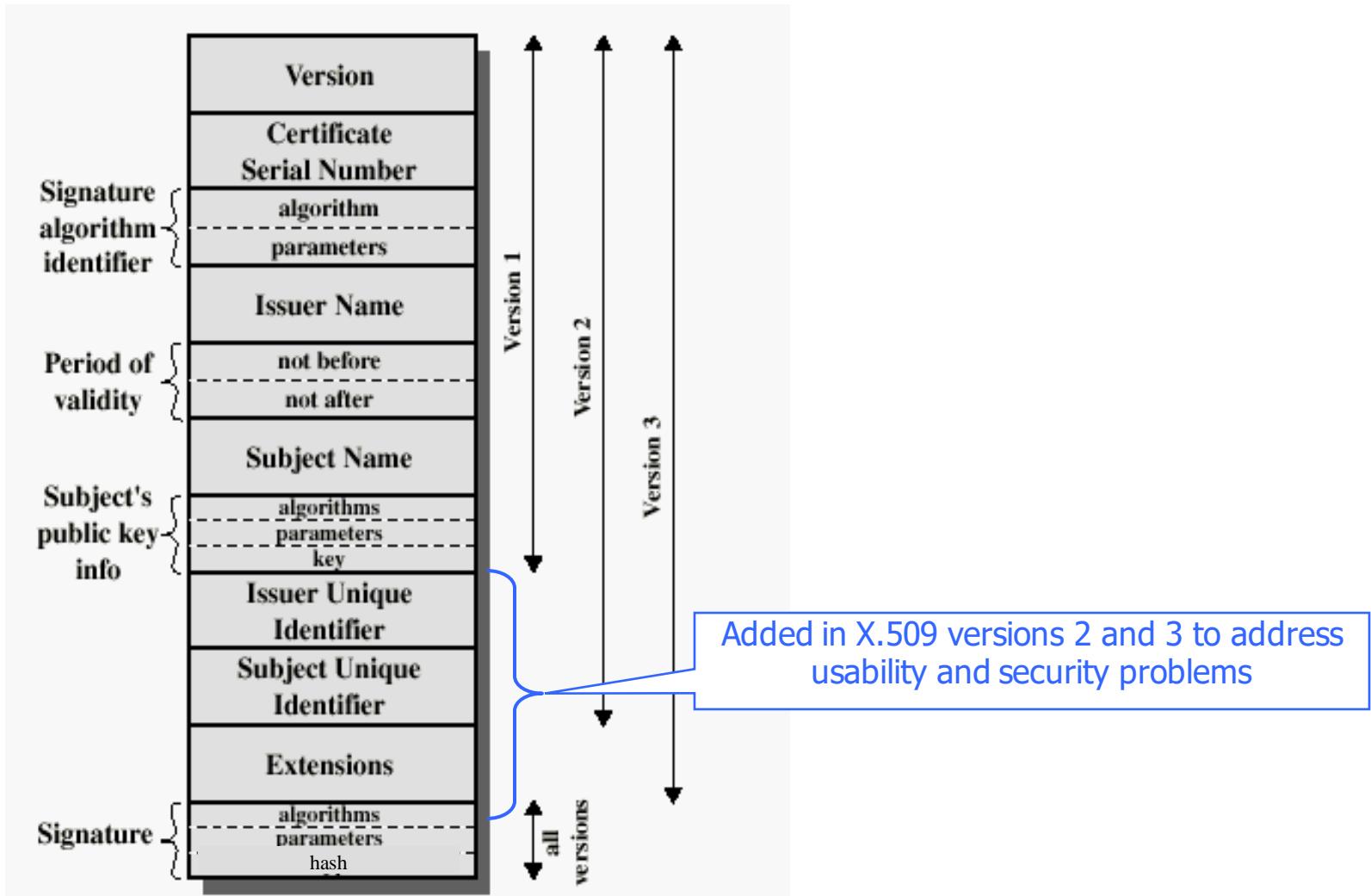


X.509 Authentication Service

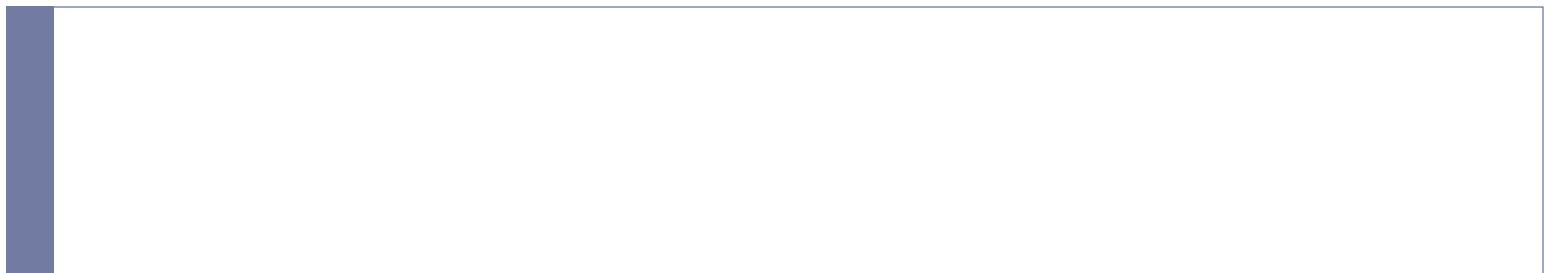
- Internet standard (1988-2000)
- Specifies certificate format
 - X.509 certificates are used in IPsec and SSL/TLS
- Specifies certificate directory service
 - For retrieving other users' CA-certified public keys
- Specifies a set of authentication protocols
 - For proving identity using public-key signatures
- Can use with any digital signature scheme and hash function, but must hash before signing

Remember MD5?

X.509 Certificate



Network Security- IPSec



By,
M.K.Chavan

Introduction

- ▶ Internet was tiny and relatively private. Today it is enormous and truly public.
 - ▶ A number of methods have evolved over the years to address the need for security. Most of them are focused on the higher layers of the OSI model.
 - ▶ For example, SSL (secure sockets layer) can be used for certain applications like world wide web or file transfer protocol (FTP).
 - ▶ IPSec is not a single protocol. It is a set of services and protocols that provide a complete security solution for an IP network.
- 

IPSec

- ▶ **IP services and functions**
 - ▶ Encryption of user data and privacy.
 - ▶ Authentication of the integrity of a message to ensure that it is not changed.
 - ▶ Protection against certain types of security attacks such as replay attacks.
 - ▶ Ability for devices to negotiate the security algorithm and the required keys.

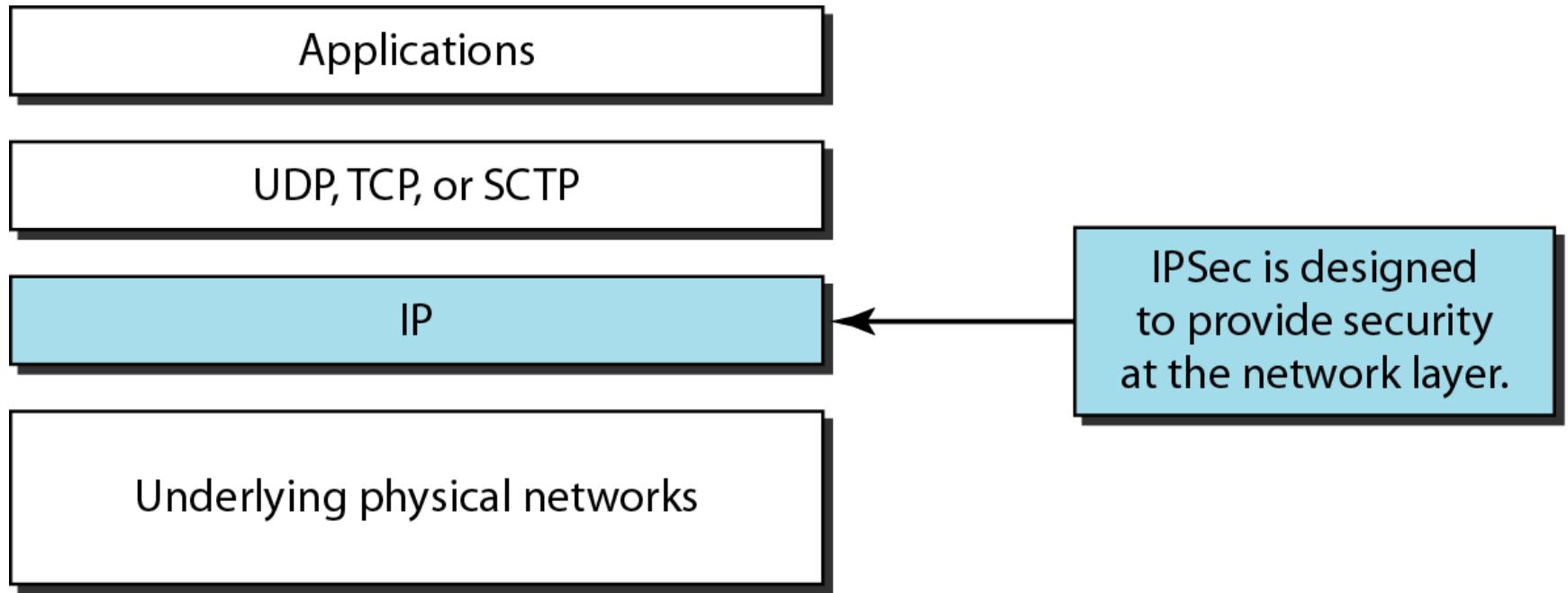


IPSec

- ▶ **IPSec operation**
- ▶ When two devices (user hosts, or intermediate devices such as routers and firewalls) want to engage in a secure communication, they set up a secure path between themselves that may traverse across many insecure intermediate systems.
- ▶ Devices must agree on a set of security protocols such that each one sends data in a format that the other can understand.
- ▶ Devices must decide on an encryption algorithm.
- ▶ Devices must exchange keys.
- ▶ **IPSec provide confidentiality and authentication to the IP layer.**



TCP/IP protocol suite and IPSec



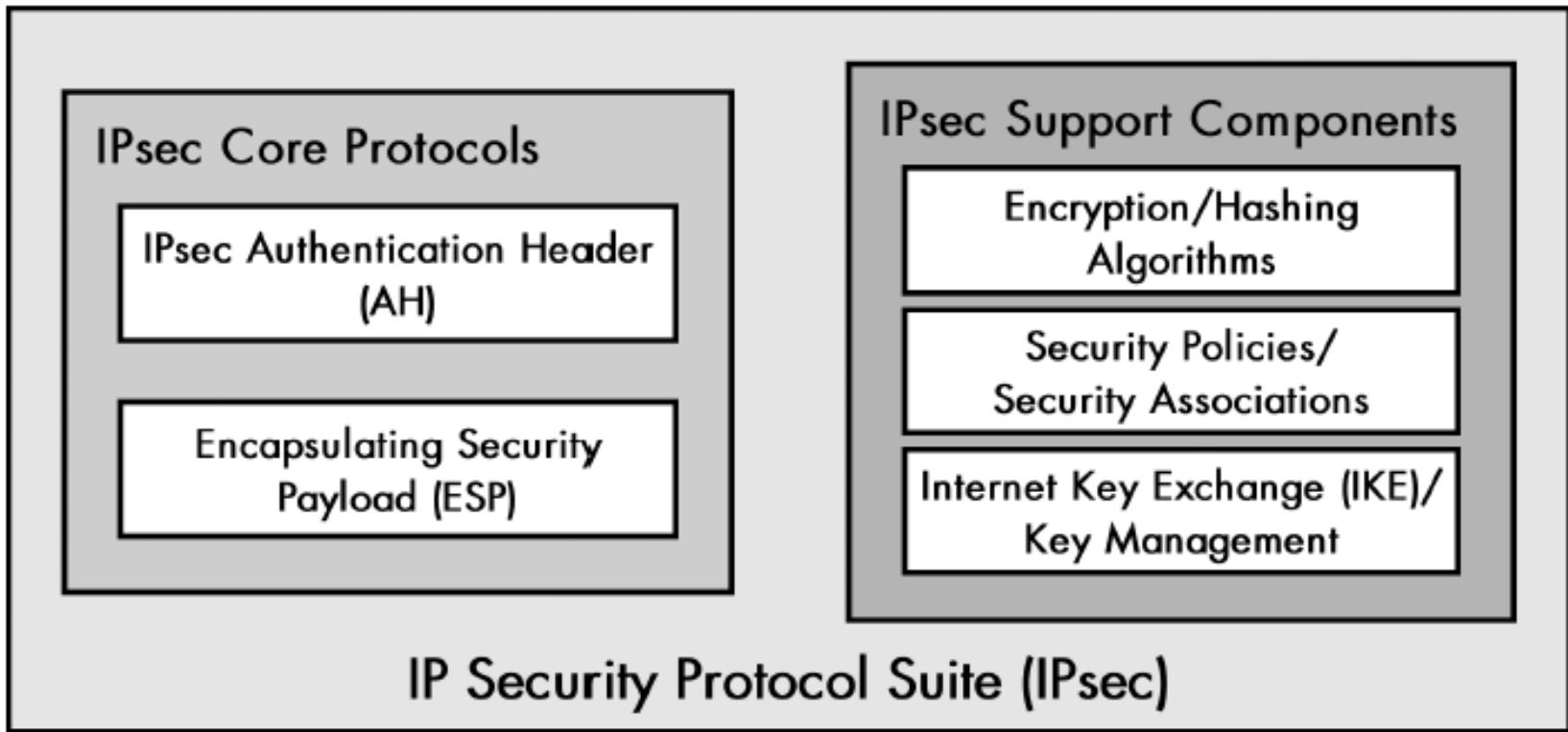
IPsec

- ▶ **IPSec core protocols:**
- ▶ A number of different components make up the total package known as IPSec.

- 1- IPSec authentication header (AH): allows to verify that the intermediate devices have not changed any of the data in the datagram.
- 2- Encapsulated security payload (ESP): AH ensures the integrity of the data in a datagram, but not its privacy. ESP allows encryption to ensure privacy of a message.



IPSec



IPSec

▶ **IPSec architecture:**

1. Host-host implementation:

- ▶ Putting all IPSec into all hosts devices.
- ▶ Enables end to end security between any two devices on the network.

2- Router implementation:

- Is much less work. You make changes to only a few routers instead of hundreds of clients. It provides protection only between pairs of routers.



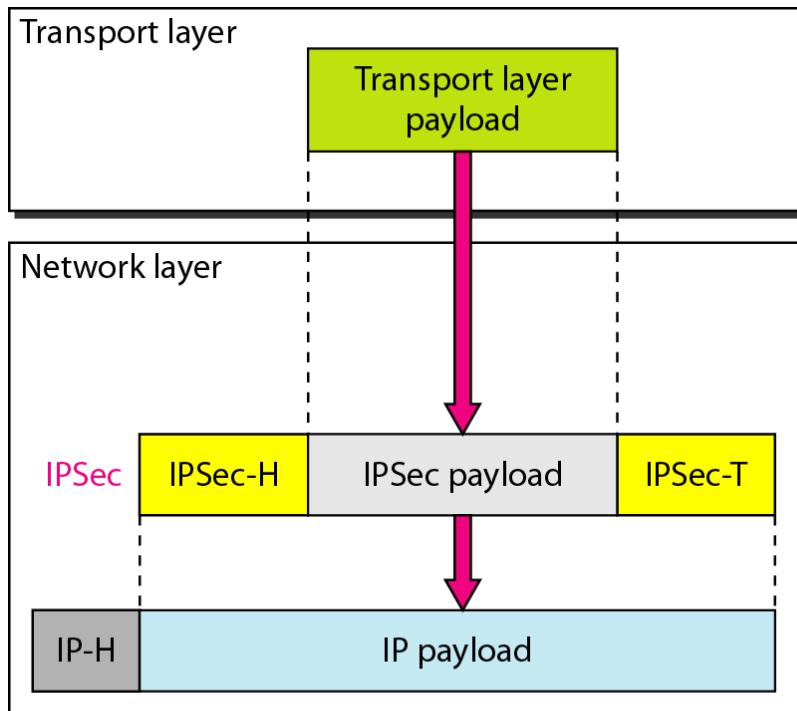
IPSec Modes

I- Transport mode:

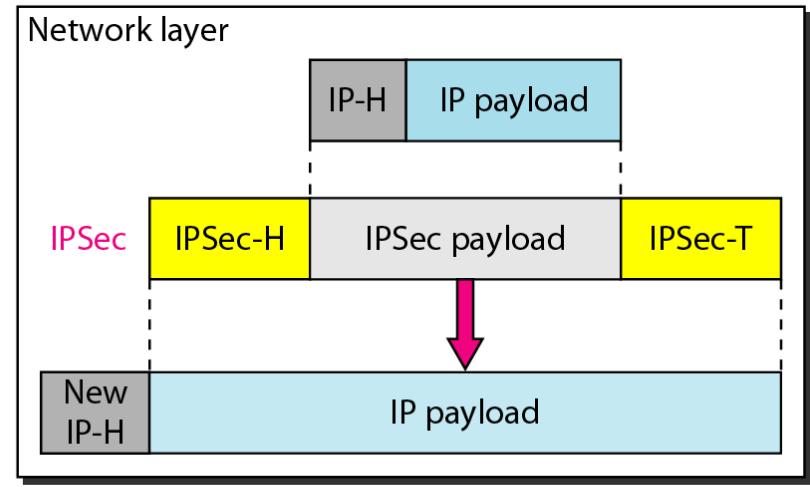
- ▶ IPSec protects the message passed down to IP from the transport layer. The message is processed by AH and /or ESP and the appropriate headers are added.
- ▶ IPSec in the transport mode does not protect the IP header; it only protects the information coming from the transport layer.
- ▶ The transport mode is normally used when we need host-to-host protection of data.



IPSec



a. Transport mode



b. Tunnel mode

Transport Mode

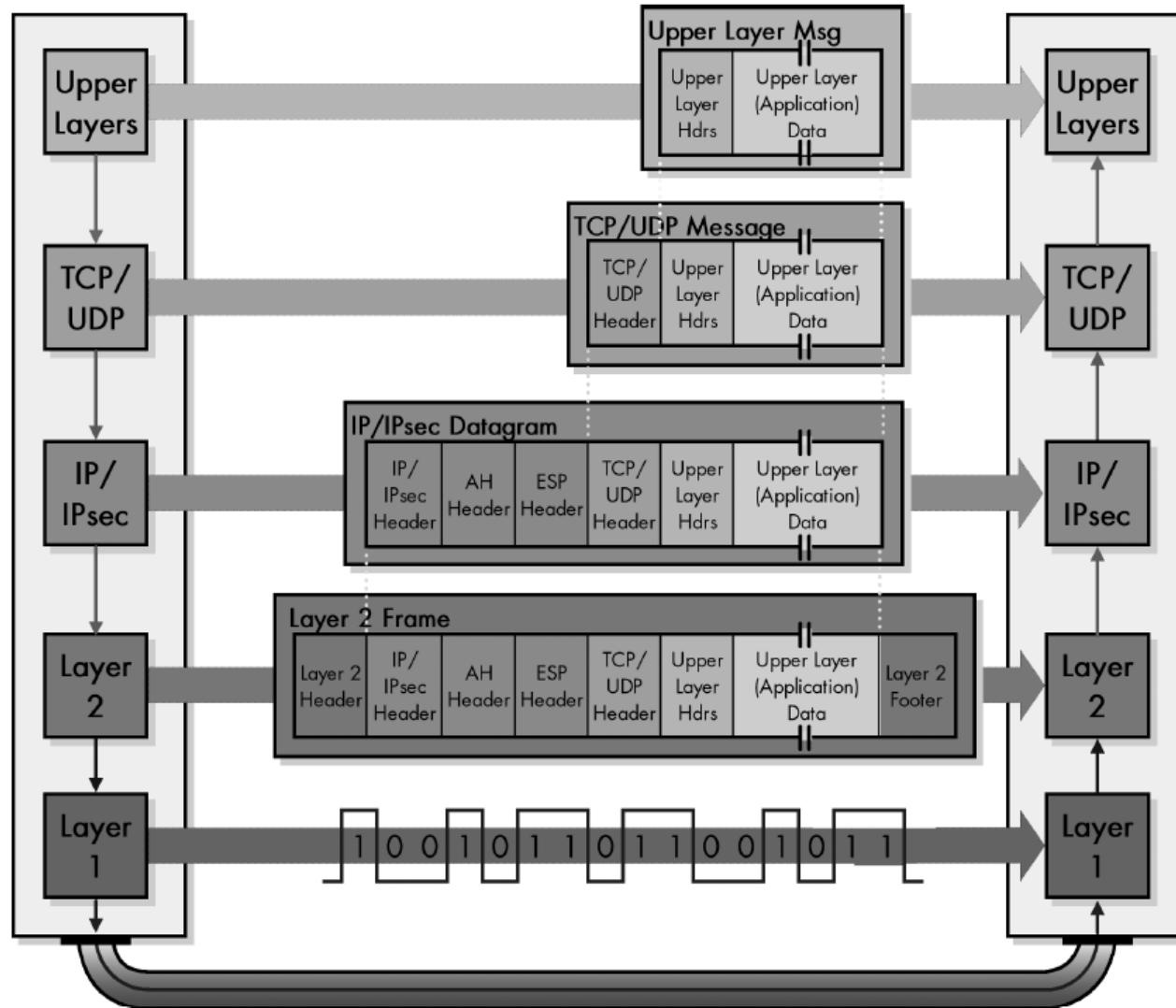


Figure 22-4 IPsec Transport Mode: A Message Flow Diagram

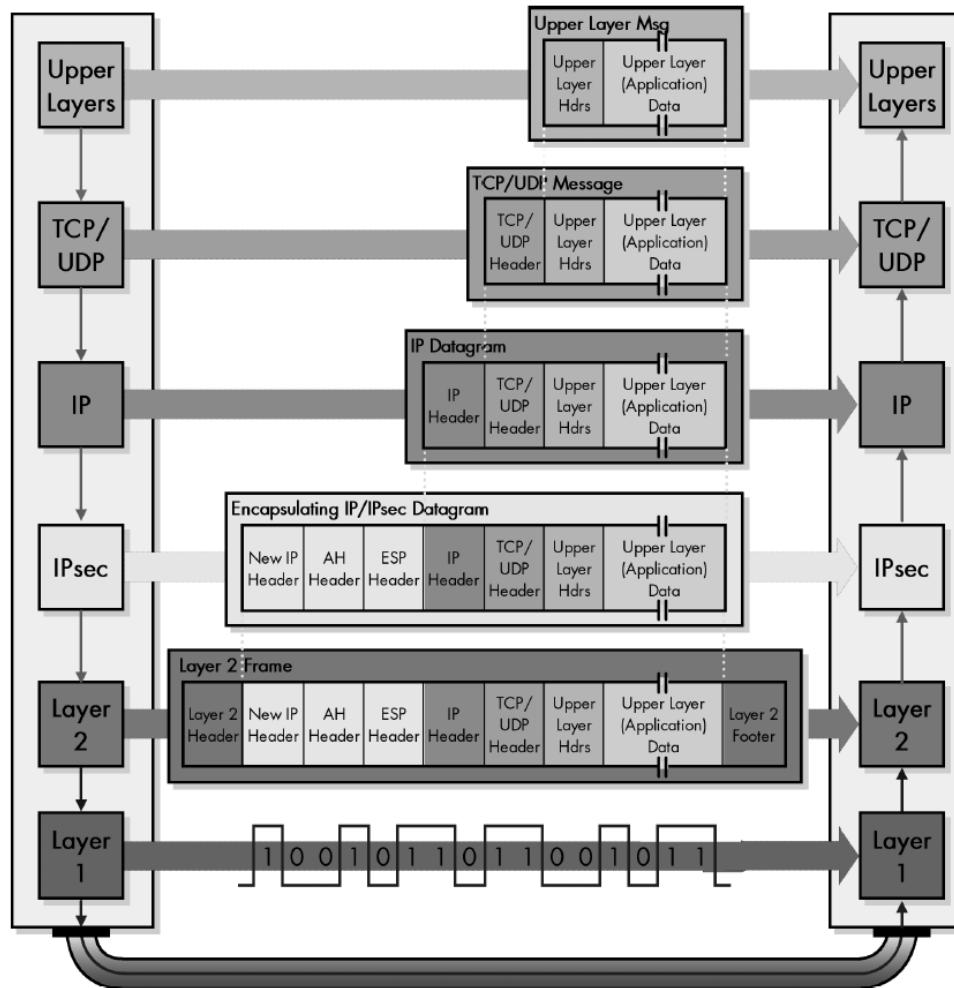
Tunnel Mode

2- *Tunnel mode:*

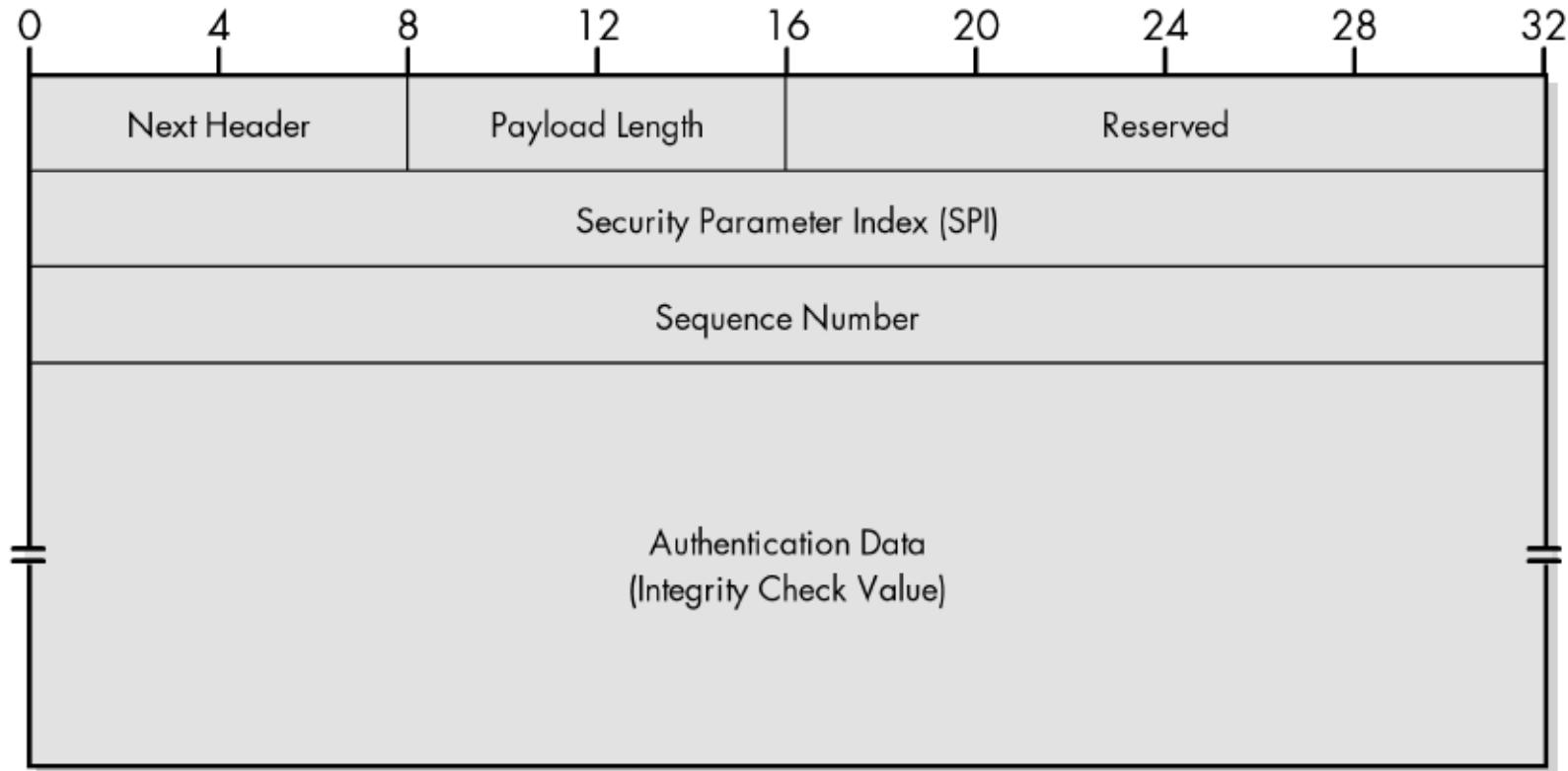
- ▶ IPSec is used to protect a completely encapsulated IP datagram after the IP header has already been applied to it.
- ▶ IPSec in tunnel mode protects the original IP header.
- ▶ It takes an IP packet, including the header, applies IPSec security methods to the entire packet, and then adds a new IP header.
- ▶ It is used when either the sender or the receiver is not a host.



Tunnel Mode



IPSec Authentication Header (AH)



IPSec Authentication Header (AH)

- ▶ **Next header:** the 8-bit next-header field defines the type of payload carried by the IP datagram (such as TCP, UDP, ICMP,...).
- ▶ **Payload length:** it defines the length of the authentication header
- ▶ **Security Parameter index:** the 32-bit security parameter index (SPI) is same for all packets sent during a connection called a security association.
- ▶ **Sequence number:** the 32-bit sequence number provides ordering information for a sequence of datagram.



Authentication Header (AH) Protocol in transport mode

- ▶ **Authentication data:** Authentication data field is the result of applying a hash function to the entire IP datagram except for the field that are changed during transit e.g. time-to-live.

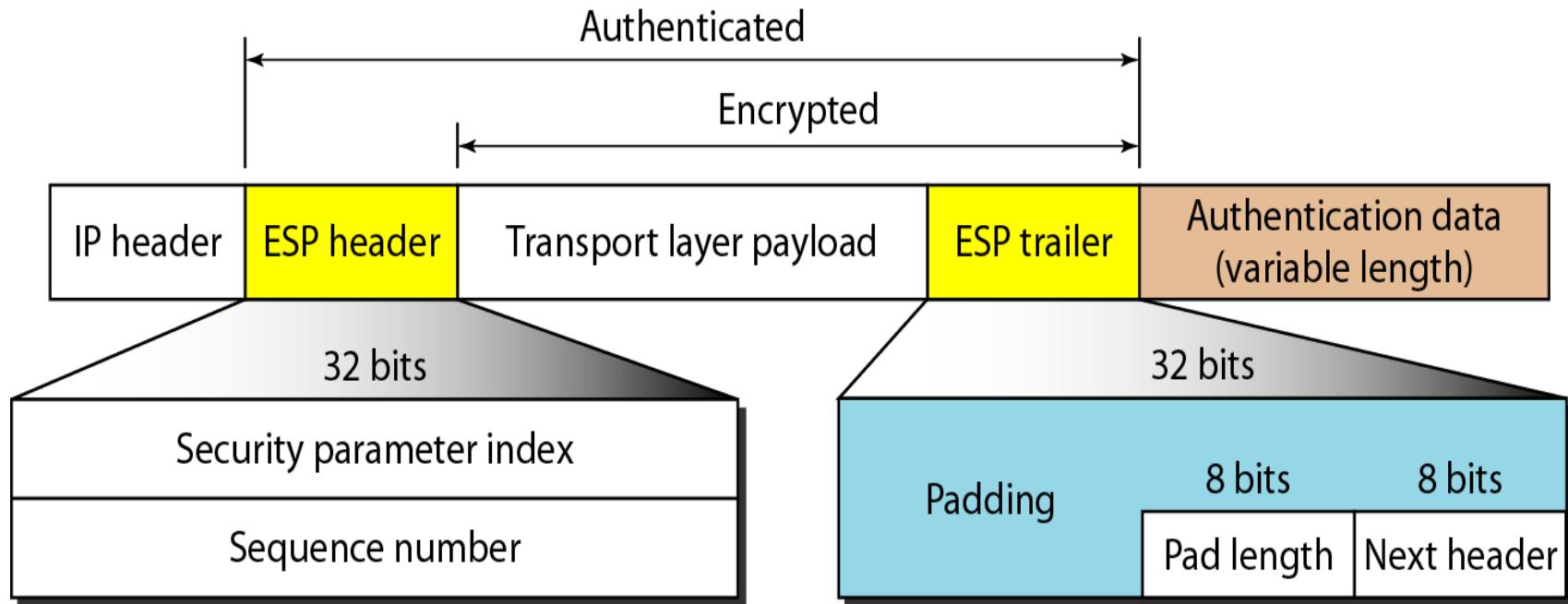


Encapsulating Security Payload (ESP) Protocol

- Since AH does not provide privacy, IPSec later define an alternative protocol that provides source authentication, integrity, and privacy called ***Encapsulating Security Payload (ESP) Protocol.***
- ESP adds a header and trailer.



Encapsulating Security Payload (ESP)



Encapsulating Security Payload (ESP) Protocol in transport mode

- ▶ **Security parameter index:** the 32-bit security parameter index field is similar to that defined for the AH protocol.
 - ▶ **Sequence number:** the 32-bit sequence number field is similar to that defined for the AH protocol.
 - ▶ **Padding:** this variable-length field (0 to 255 bytes) of 0s serves as padding.
 - ▶ **Pad length:** the 8-bit pad length field defines the number of padding bytes.
-

AH Versus ESP

- ▶ The ESP Protocol was designed after AH Protocol was already in use.
- ▶ ESP does whatever AH does with additional functionality (privacy).
- ▶ Why do we need AH ?
 - ▶ We don't, but the implementation of AH is already included in some commercial products.



Services Provided by IPSec

- ▶ The two protocols AH and ESP can provide several security services for packets at the network layer as shown in the table below:

<i>Services</i>	<i>AH</i>	<i>ESP</i>
Access control	Yes	Yes
Message authentication (message integrity)	Yes	Yes
Entity authentication (data source authentication)	Yes	Yes
Confidentiality	No	Yes
Replay attack protection	Yes	Yes



Services Provided by IPSec

- ▶ **Access Control:** IPSec provides access control indirectly by using a Security Association Database (SADB).
- ▶ **Message Authentication:** the integrity of the message is preserved in both AH and ESP by using the authentication data.
- ▶ **Entity Authentication:** The security association and the keyed-hashed digest of the data sent by the sender authenticate the sender in both AH and ESP.



Services Provided by IPSec

- ▶ **Confidentiality:** The encryption of the message in ESP provides confidentiality. AH doesn't provide confidentiality.
- ▶ **Replay Attack Protection:** both protocols prevent replay attack by using sequence numbers.



Security Association

- ▶ It is a mechanism that IPSec used to establish the security parameters.
- ▶ IP is connectionless protocol (each datagram is independent of others).



Security Association

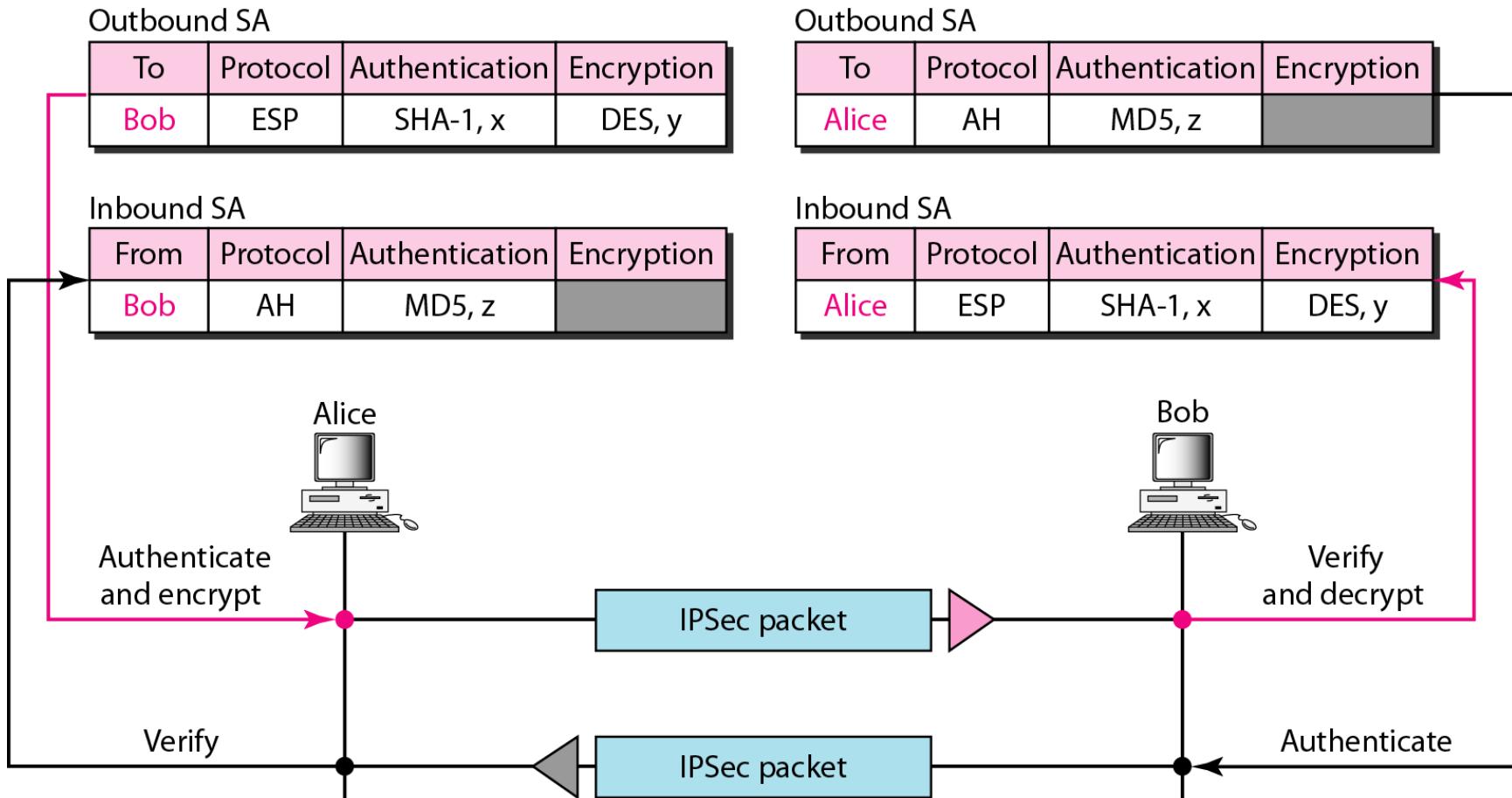
- ▶ A set of security parameters can be established between a sender and a particular receiver the first time they have communication.

- ▶ It is called Security Association

- ▶ Using Security Association , IPSec changes a connectionless protocol (IP) to a connection- oriented protocol.



Simple inbound and outbound security associations



Security Association Database (SADB)

- ▶ What if Alice needs to send to many people and receive from many people too.
- ▶ She needs to have multiple inbound and outbound SAs.
- ▶ Thus, SADB is needed to collect those sets of SAs.
- ▶ SADB it is a two-dimensional table with each row defining a single SA.
- ▶ Normally, there are two SADBs one inbound and one outbound.



Security Parameter Index (SPI)

- ▶ It is used to distinguish one association from the other.
- ▶ Each association is defined by a parameter called the Security Parameter Index (SPI).
- ▶ SPI contains the destination address (outbound) or source address (inbound) and protocol (AH or ESP). → uniquely identifies an association!

module 5

Electronic mail security

mkc

Outline

- Pretty good privacy
- S/MIME
- Recommended web sites

Pretty Good Privacy

- Philip R. Zimmerman is the creator of PGP.
- PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

Figure 32.19 *Position of PGP in the TCP/IP protocolsuite*

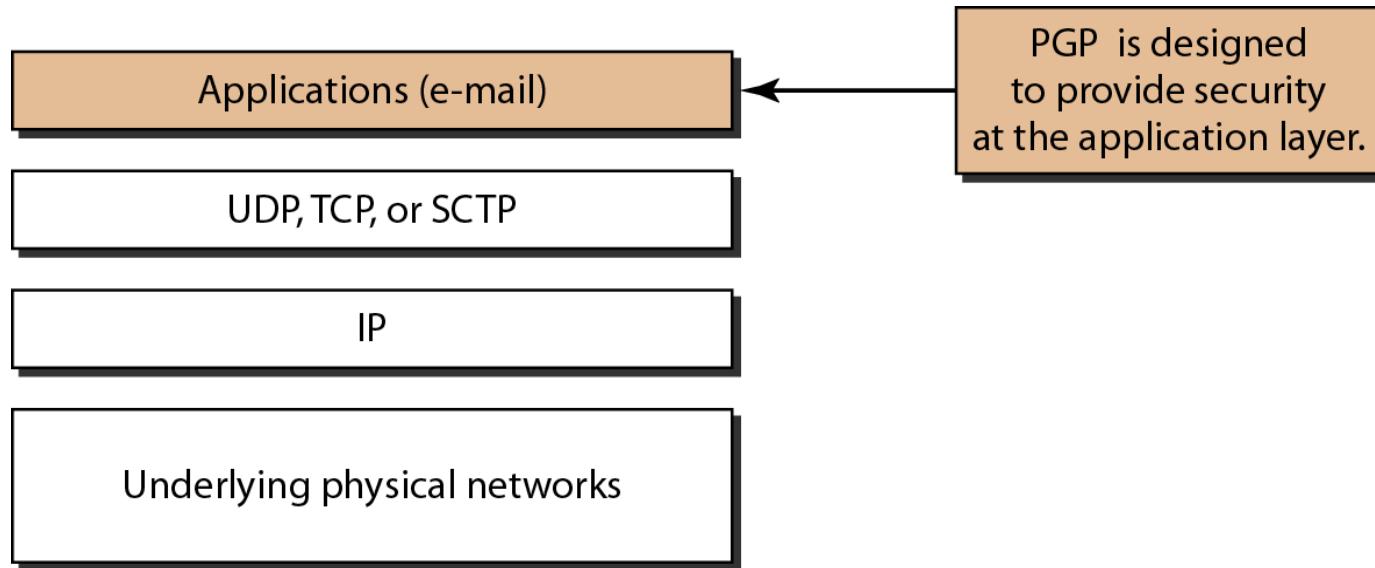
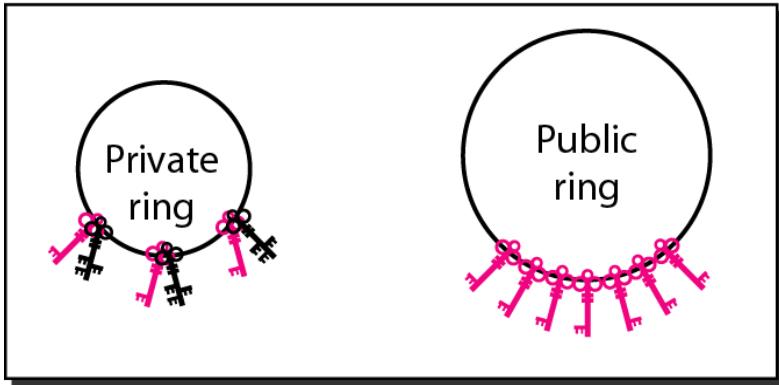


Table 32.4 PGP Algorithms

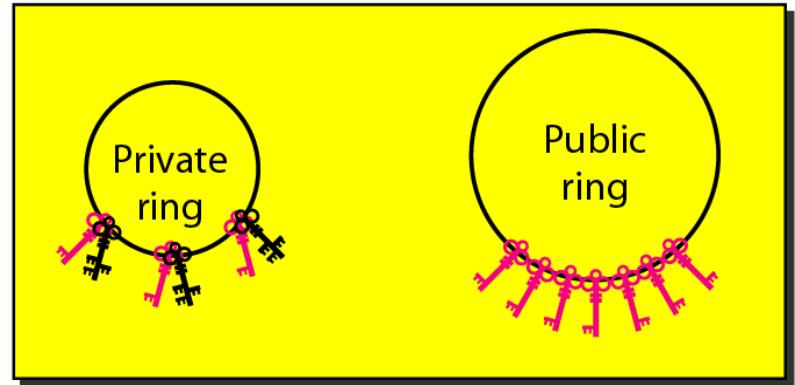
<i>Algorithm</i>	<i>ID</i>	<i>Description</i>
Public key	1	RSA (encryption or signing)
	2	RSA (for encryption only)
	3	RSA (for signing only)
	17	DSS (for signing)
Hash algorithm	1	MD5
	2	SHA-1
	3	RIPE-MD
Encryption	0	No encryption
	1	IDEA
	2	Triple DES
	9	AES

Figure 32.21 *Rings*

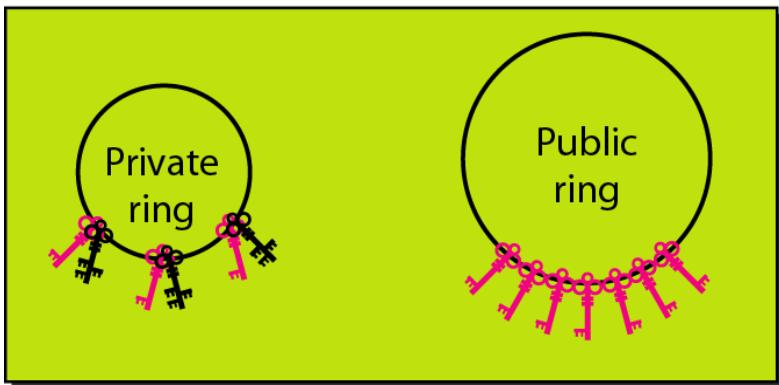
Alice's rings



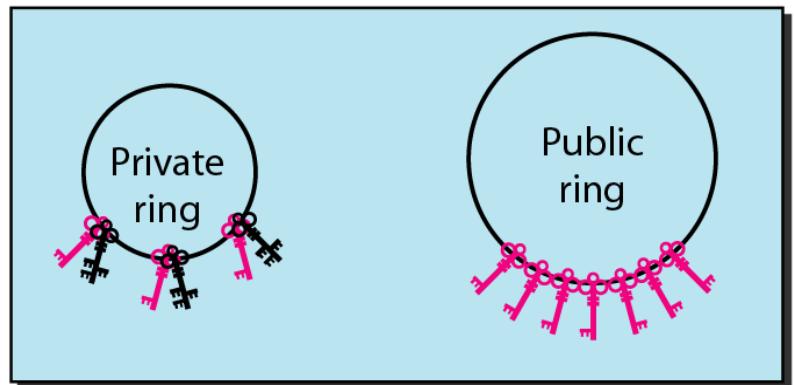
Bob's rings

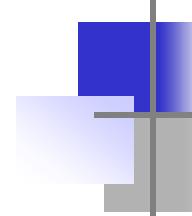


Ted's rings



John's rings





Note

In PGP, there can be multiple paths from fully or partially trusted authorities to any subject.

Why Is PGP Popular?

- It is available free on a variety of platforms.
- Based on well known algorithms.
- Wide range of applicability
- Not developed or controlled by governmental or standards organizations

Operational Description

- Consist of five services:
 - Authentication
 - Confidentiality
 - Compression
 - E-mail compatibility
 - Segmentation

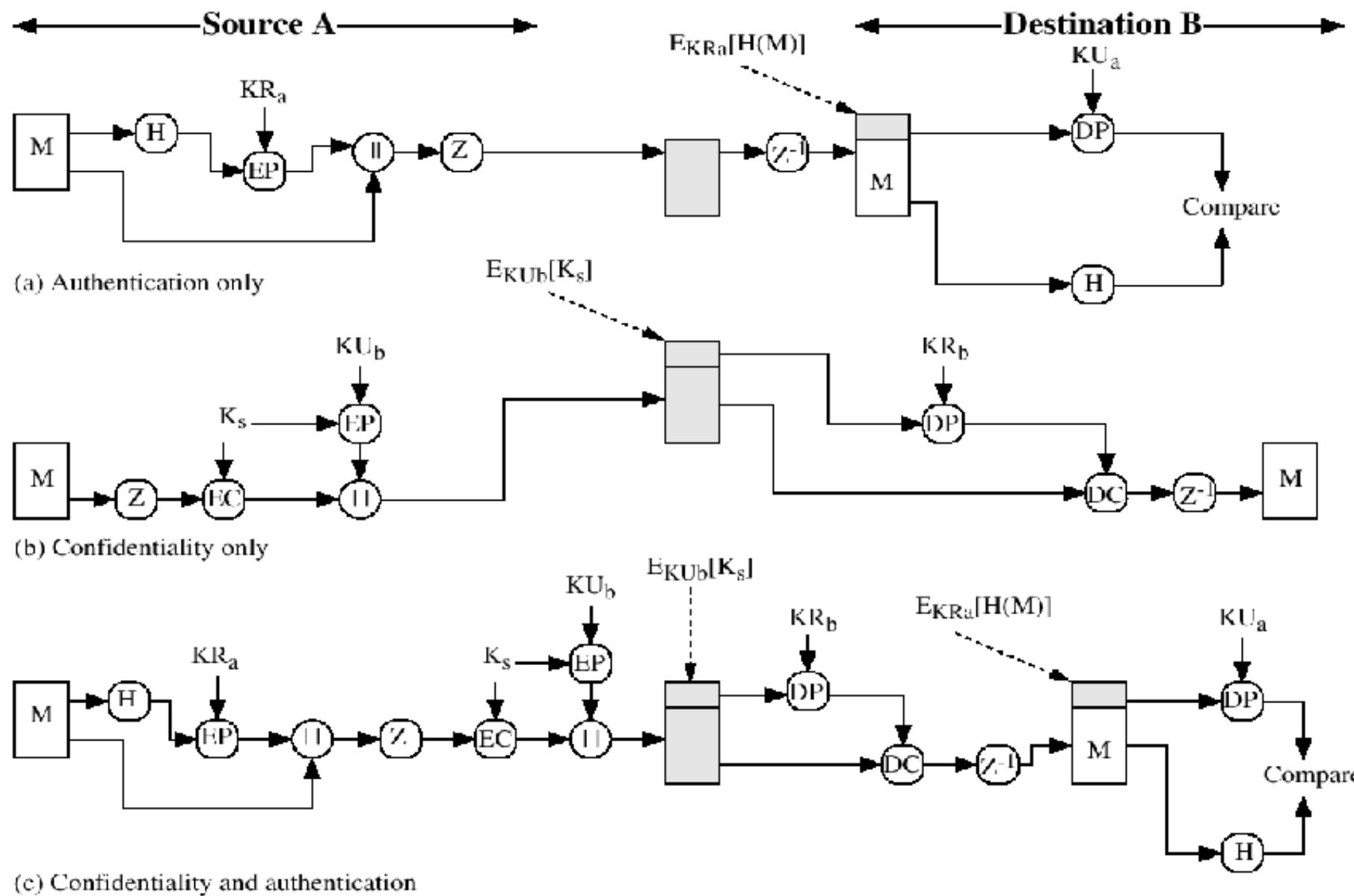


Figure 5.1 PGP Cryptographic Functions

Compression

- PGP compresses the message after applying the signature but before encryption
- The placement of the compression algorithm is critical.
- The compression algorithm used is ZIP (described in appendix 5A)

E-mail Compatibility

- The scheme used is radix-64 conversion (see appendix 5B).
- The use of radix-64 expands the message by 33%

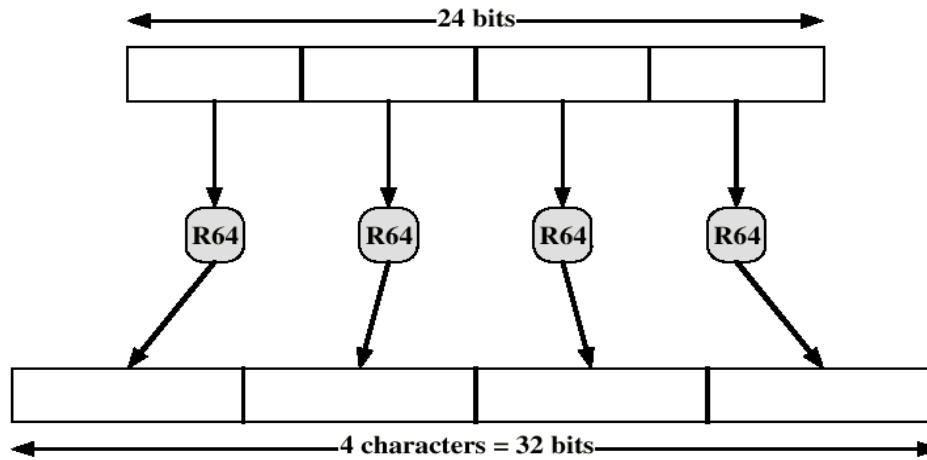


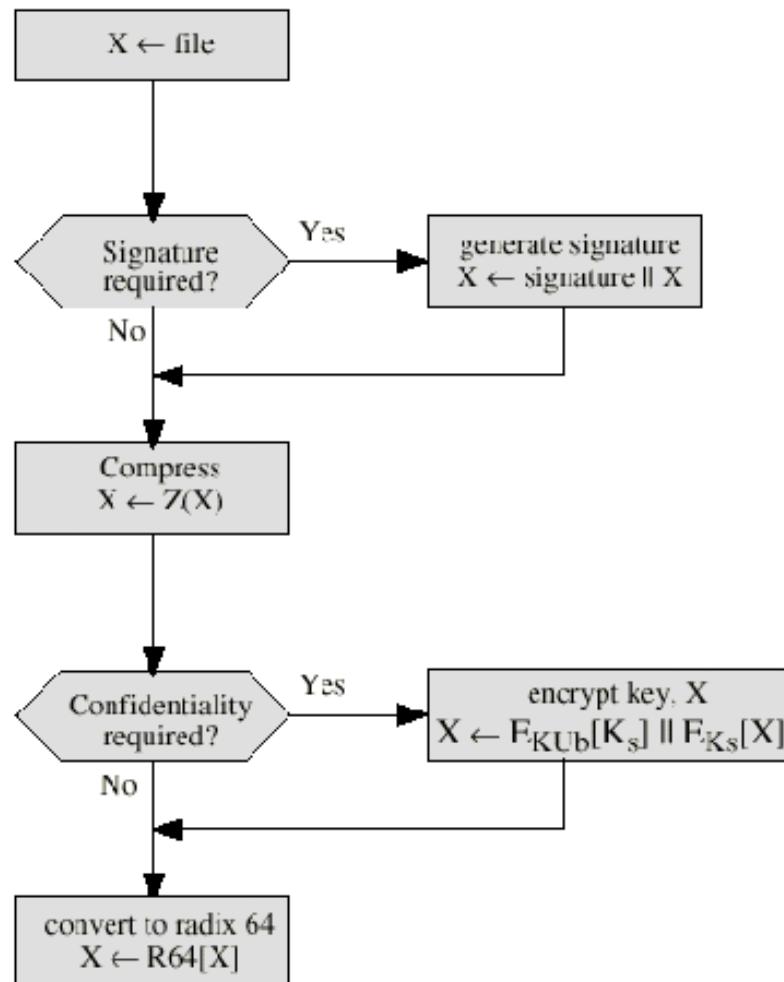
Figure 5.11 Printable Encoding of Binary Data into Radix-64 Format

Segmentation and Reassembly

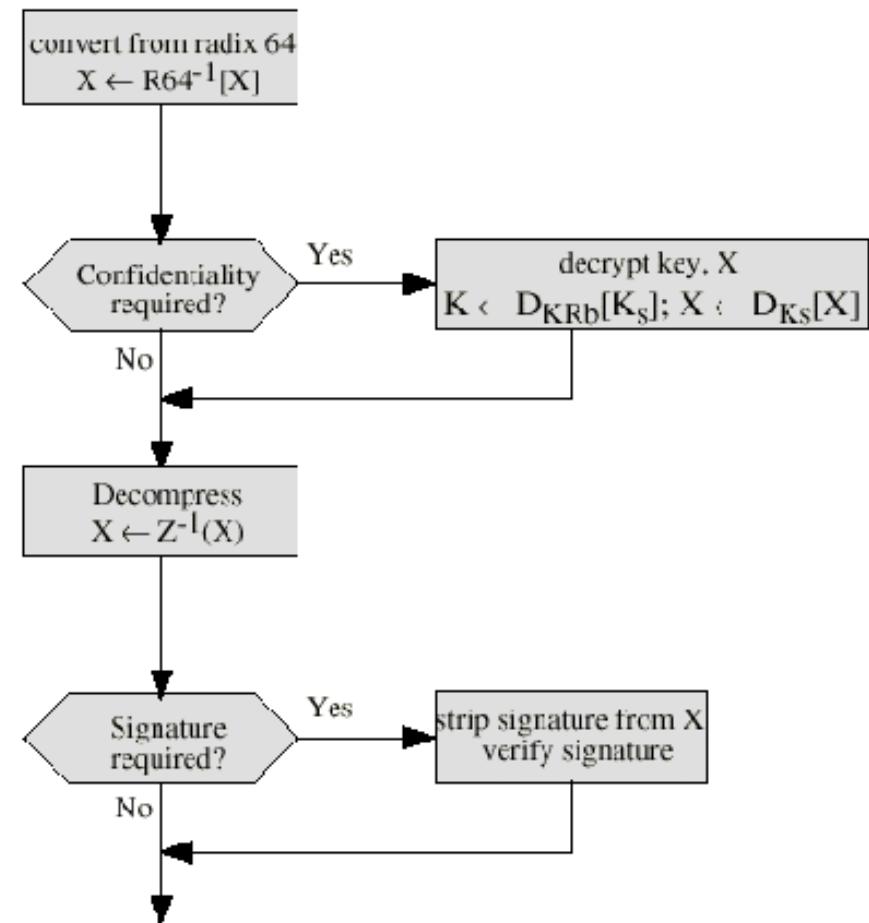
- Often restricted to a maximum message length of 50,000 octets.
- Longer messages must be broken up into segments.
- PGP automatically subdivides a message that is too large.
- The receiver strips off all e-mail headers and reassemble the block.

Summary of PGP Services

Function	Algorithm Used
Digital Signature	DSS/SHA or RSA/SHA
Message Encryption	CAST or IDEA or three-key triple DES with Diffie-Hellman or RSA
Compression	ZIP
E-mail Compatibility	Radix-64 conversion
Segmentation	



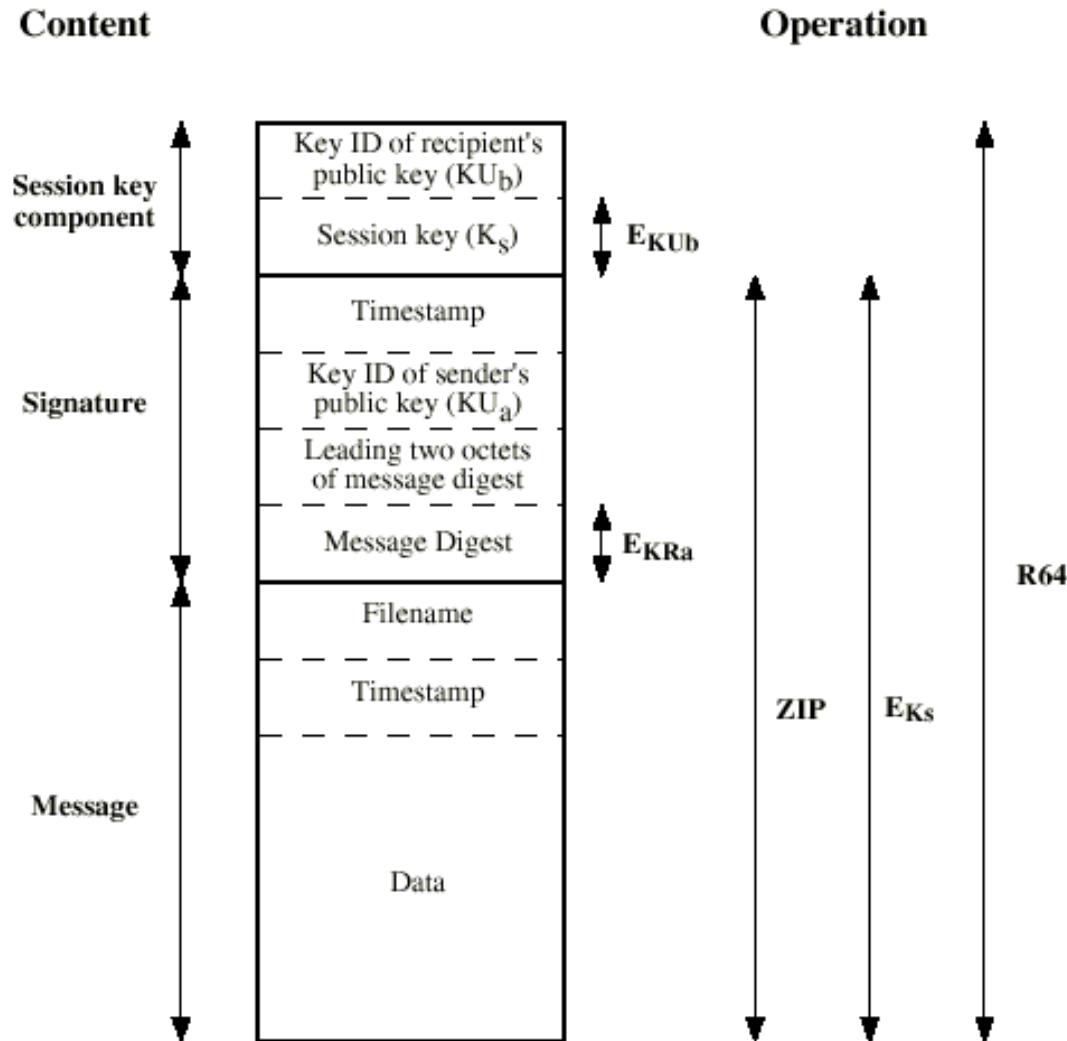
(a) Generic Transmission Diagram (from A)



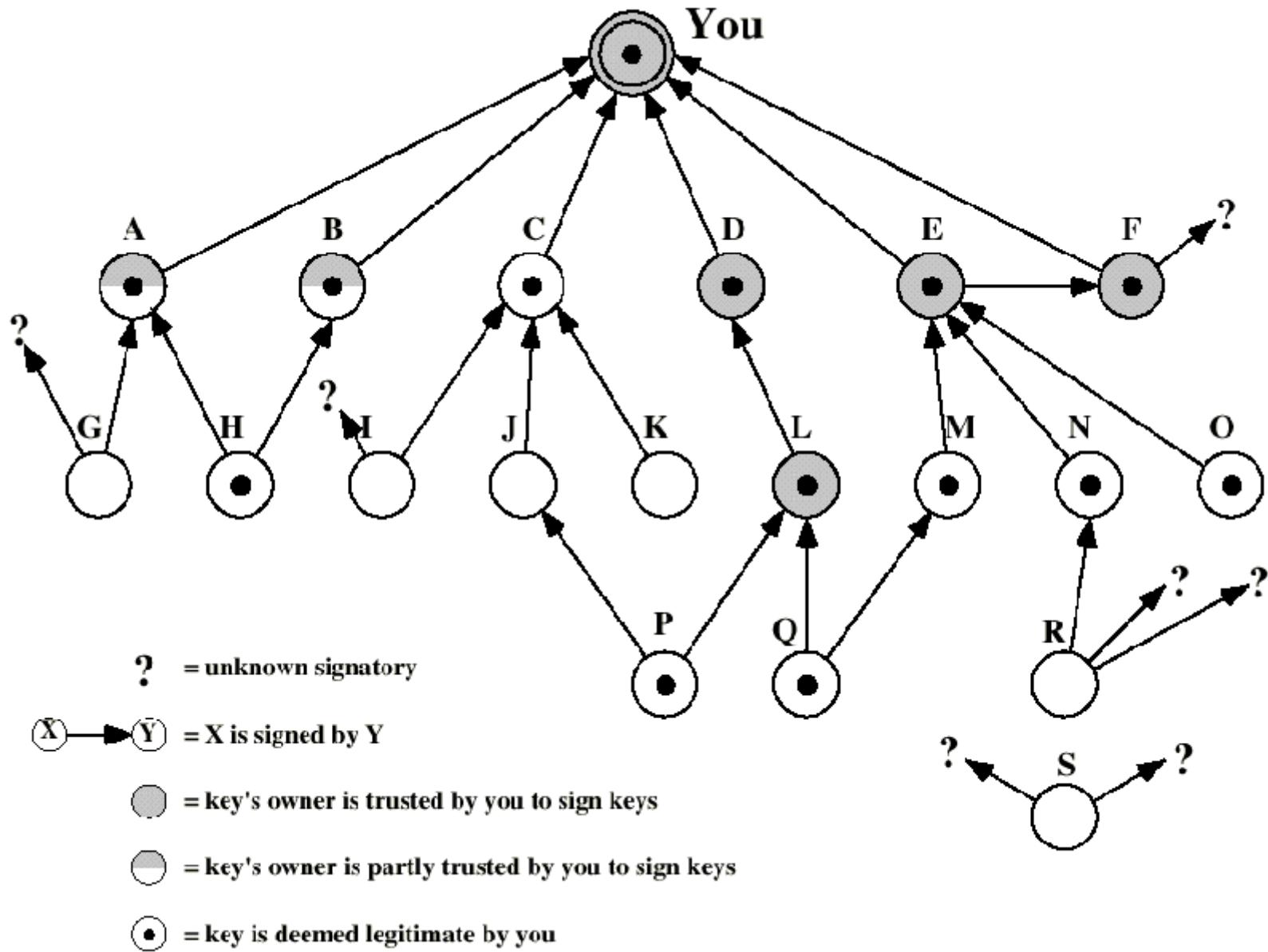
(b) Generic Reception Diagram (to B)

Figure 5.2 Transmission and Reception of PGP Messages

Format of PGP Message



The Use of Trust



Revoking Public Keys

- The owner issue a key revocation certificate.
- Normal signature certificate with a revote indicator.
- Corresponding private key is used to sign the certificate.

S/MIME

- Secure/Multipurpose Internet Mail Extension
- S/MIME will probably emerge as the industry standard.
- PGP for personal e-mail security

Simple Mail Transfer Protocol (SMTP, RFC 822)

- **SMTP Limitations** - Can not transmit, or has a problem with:
 - executable files, or other binary files (jpeg image)
 - “national language” characters (non-ASCII)
 - messages over a certain size
 - ASCII to EBCDIC translation problems
 - lines longer than a certain length (72 to 254 characters)

Header fields in MIME

- **MIME-Version:** Must be "1.0" -> RFC 2045, RFC 2046
- **Content-Type:** More types being added by developers (application/word)
- **Content-Transfer-Encoding:** How message has been encoded (radix-64)
- **Content-ID:** Unique identifying character string.
- **Content Description:** Needed when content is not readable text (e.g.,mpeg)

S/MIME Functions

- **Enveloped Data:** Encrypted content and encrypted session keys for recipients.
- **Signed Data:** Message Digest encrypted with private key of "signer."
- **Clear-Signed Data:** Signed but not encrypted.
- **Signed and Enveloped Data:** Various orderings for encrypting and signing.

Algorithms Used

- **Message Digesting:** SHA-1 and MDS
- **Digital Signatures:** DSS
- **Secret-Key Encryption:** Triple-DES, RC2/40 (exportable)
- **Public-Private Key Encryption:** RSA with key sizes of 512 and 1024 bits, and Diffie-Hellman (for session keys).

User Agent Role

- S/MIME uses Public-Key Certificates - X.509 version 3 signed by Certification Authority
- Functions:
 - **Key Generation** - Diffie-Hellman, DSS, and RSA key-pairs.
 - **Registration** - Public keys must be registered with X.509 CA.
 - **Certificate Storage** - Local (as in browser application) for different services.
 - **Signed and Enveloped Data** - Various orderings for encrypting and signing.

User Agent Role

- Example: Verisign (www.verisign.com)
 - Class-1: Buyer's email address confirmed by emailing vital info.
 - Class-2: Postal address is confirmed as well, and data checked against directories.
 - Class-3: Buyer must appear in person, or send notarized documents.

Recommended Web Sites

- PGP home page: www.pgp.com
- MIT distribution site for PGP
- S/MIME Charter
- S/MIME Central: RSA Inc.'s Web Site

Radix-64 Conversion in PGP

Cunsheng Ding

Department of CSE

HKUST

PGP E-Mail Compatibility

Many electronic mail systems can only transmit blocks of ASCII text. This can cause a problem when sending encrypted data since ciphertext blocks might not correspond to ASCII characters which can be transmitted.

PGP overcomes this problem by using radix-64 conversion.

PGP E-Mail Compatibility: Example

- Suppose the email message is: new
- ASCII format: 01101110 01100101 01110111
- After encryption: 10010001 10011010 10001000
- The problem after encryption:
 - the three bytes do not represent any keyboard ASCII characters.
 - Most email systems cannot transmit and process such a piece of ciphertext.

Radix-64 Conversion

Suppose the text to be encrypted has been converted into binary using ASCII coding and encrypted to give a ciphertext stream of binary.

Radix-64 conversion maps arbitrary binary into printable characters as follows:

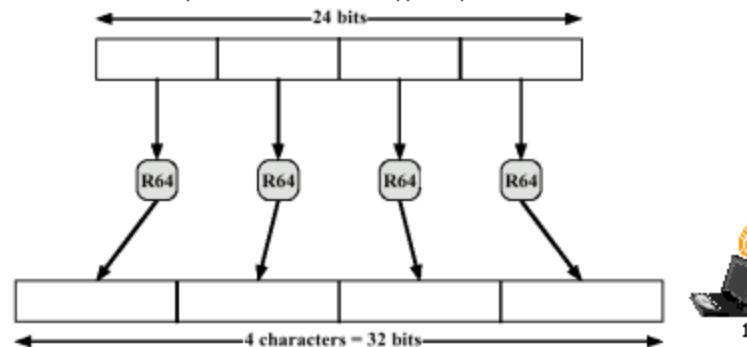
Radix-64 Conversion

1. The binary input is split into blocks of 24 bits (3 bytes).
2. Each 24 block is then split into four sets each of 6-bits.
3. Each 6-bit set will then have a value between 0 and 2^6-1 (=63).
4. This value is encoded into a printable character.

Pictorial Description

Radix-64 Conversion

- To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion
- Radix-64 expands a message by 33%



6 bit value	Character encoding						
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

Radix-64 Conversion: Example

- Suppose the email message is: new
- ASCII format: 01101110 01100101 01110111
- After encryption: 10010001 10011010 10001000
- The Radix-64 conversion:
 - The 24-bit block: 10010001 10011010 10001000
 - Four 6-bit blocks: 100100 011001 101010 001000
 - Integer version: 36 25 38 8
 - Printable version: k Z m I

Computer Networks and Vulnerabilities

Vulnerabilities in Computer Networks

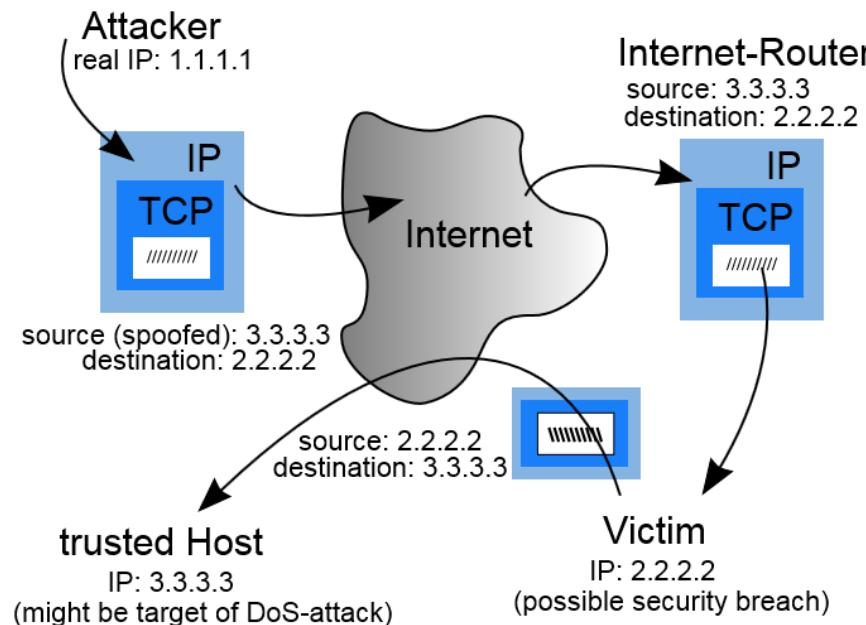
	Threats	Consequences	Countermeasures
Integrity	<ul style="list-style-type: none">• Modification of user data• Trojan horse browser• Modification of memory• Modification of message traffic in transit	<ul style="list-style-type: none">• Loss of information• Compromise of machine• Vulnerability to all other threats	Cryptographic checksums
Confidentiality	<ul style="list-style-type: none">• Eavesdropping on the net• Theft of info from server• Theft of data from client• Info about network configuration• Info about which client talks to server	<ul style="list-style-type: none">• Loss of information• Loss of privacy	Encryption, Web proxies
Denial of Service	<ul style="list-style-type: none">• Killing of user threads• Flooding machine with bogus requests• Filling up disk or memory• Isolating machine by DNS attacks	<ul style="list-style-type: none">• Disruptive• Annoying• Prevent user from getting work done	Difficult to prevent
Authentication	<ul style="list-style-type: none">• Impersonation of legitimate users• Data forgery	<ul style="list-style-type: none">• Misrepresentation of user• Belief that false information is valid	Cryptographic techniques

Protection of confidentiality, integrity and authentication: cryptography
(see a separate module – cryptography and SSL in transport layer)

Attacks in network layer

IP Spoofing

IP spoofing is sending IP packets with a buggy source address with intent to conceal the sender's identity. IP spoofing may be used in denial-of-service (DoS) attacks.



IP Spoofing - Continue

- IP spoofing is most frequently used in denial-of-service attacks. In such attacks, the goal is to flood the victim with overwhelming amounts of traffic, and the attacker does not care about receiving responses to the attack packets. Packets with spoofed addresses are thus suitable for such attacks. They are more difficult to filter if each spoofed packet appears to come from a different address, and they hide the true source of the attack.
- Denial of service attacks that use spoofing typically randomly choose addresses from the entire IP address space, though more sophisticated spoofing mechanisms might avoid unrouteable addresses or unused portions of the IP address space. Attackers typically have a spoofing available tool, so defenses against denial-of-service attacks that rely on the validity of the source IP address in attack packets might have trouble with spoofed packets.

IP Spoofing - Continue

- IP spoofing can also be a method of attack used by network intruders to defeat network security measures, such as authentication based on IP addresses. This method of attack on a remote system can be extremely difficult, as it involves modifying thousands of packets at a time. This type of attack is most effective where trust relationships exist between machines.

For example, it is common on some corporate networks to have internal systems trust each other, so that users can log in without a username or password provided they are connecting from another machine on the internal network (and so must already be logged in). By spoofing a connection from a trusted machine, an attacker may be able to access the target machine without authentication.

How to prevent IP Spoofing

1. Use an access control list to deny private IP addresses on your downstream interface.
2. Implement filtering of both inbound and outbound traffic.
3. Configure your routers and switches if they support such configuration, to reject packets originating from outside your local network that claim to originate from within.
4. Use authentication based on key exchange between the machines on your network; something like IPsec will significantly cut down on the risk of spoofing.
5. Enable encryption sessions on your router so that trusted hosts that are outside your network can securely communicate with your local hosts.

Other Attacks in Network Layer

Routing (RIP) Attack:

Routing Information Protocol (RIP) is used to distribute routing information within networks, such as shortest-paths, and advertising routes out from the local network. The original version of RIP has no built in authentication, and the information provided in a RIP packet is often used without verifying it.

- An attacker could forge a RIP packet, claiming his host "X" has the fastest path out of the network. All packets sent out from that network would then be routed through X, where they could be modified or examined.
- An attacker could also use RIP to effectively impersonate any host, by causing all traffic sent to that host to be sent to the attacker's machine instead.

Mitigations:

- New version of RIP was enhanced with a simple password authentication algorithm, which makes RIP attack harder to happen.
- Internet Protocol Security (Ipsec) provides a protocol suite for secure Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session.

ICMP Attack:

The **Internet Control Message Protocol (ICMP)** is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. There is no authentication in ICMP, which leads to attacks using ICMP that can result in a denial of service, or allowing the attacker to intercept packets.

An attacker sends forged ICMP echo packets to vulnerable networks' broadcast addresses. All the systems on those networks send ICMP echo replies to the victim, consuming the target system's available bandwidth and creating a denial of service (DoS) to legitimate traffic.

Mitigations:

- Most ICMP attacks can be effectively reduced by deploying Firewalls at critical locations of a network to filter un-wanted traffic and from any destinations.
- Configure your ICMP parameters in your network devices as follows:
 - Allow ping ICMP Echo-Request outbound and Echo-Reply messages inbound.
 - Allow traceroute TTL(Time to Live)-Exceeded and Port-Unreachable messages inbound.
 - Blocking other types of ICMP traffic.

Ping Flood (ICMP Flood)

PING is one of the most common uses of ICMP which sends an ICMP "Echo Request" to a host, and waits for that host to send back an ICMP "Echo Reply" message. Attacker simply sends a huge number of "ICMP Echo Requests" typically overloading its victim that it expends all its resources responding until it can no longer process valid network traffic.

Mitigations:

- Block ICMP altogether at perimeter of your network via firewall filters.
- Limit the rate at which a single source can send ICMP Packets.

Packet Sniffing:

Most network applications distribute network packets in clear/plain text. A packet sniffing tool can exploit information passed in clear text providing the hacker with sensitive information such as user account names and passwords.

Mitigations:

- Authentication - Using strong authentication, such as one-time passwords.
- Cryptography - The most effective method for countering packet sniffers does renders them irrelevant.
- Anti-sniffer tools - Use these tools to employ software and hardware designed to detect the use of sniffers on a network.

Attacks in Transport Layer

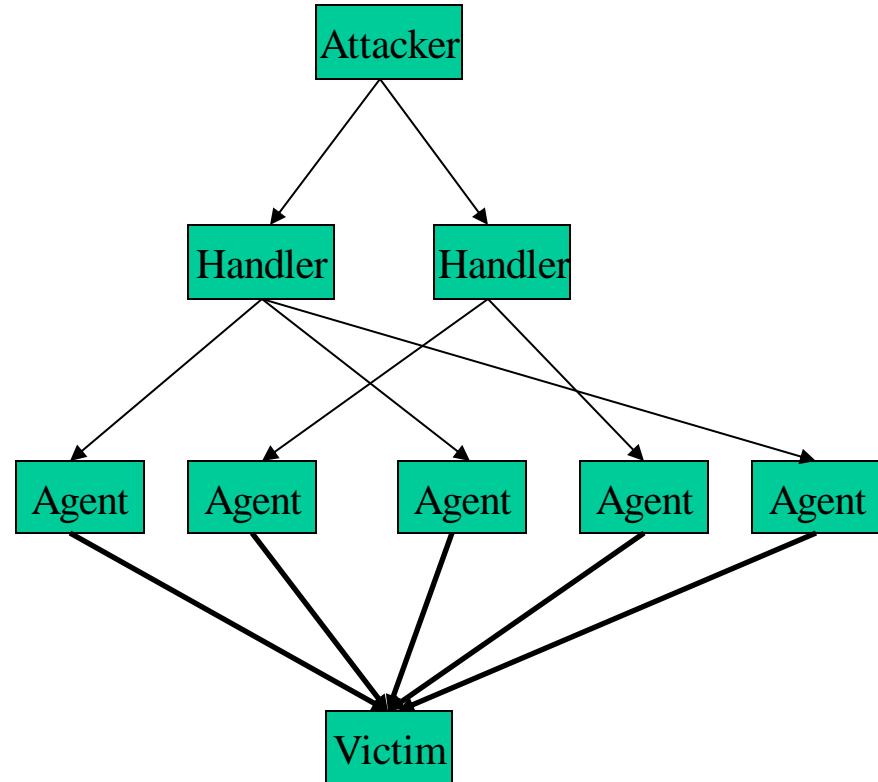
Denial of Service in Transport Layer

Make a service unusable, usually by overloading the server or network

- Consume host resources
 - TCP SYN floods
- Consume bandwidth
 - UDP floods
- Crashing the victim
 - TCP options (unused, or used incorrectly)
- Forcing more computation
 - Taking long path in processing of packets

Distributed DoS

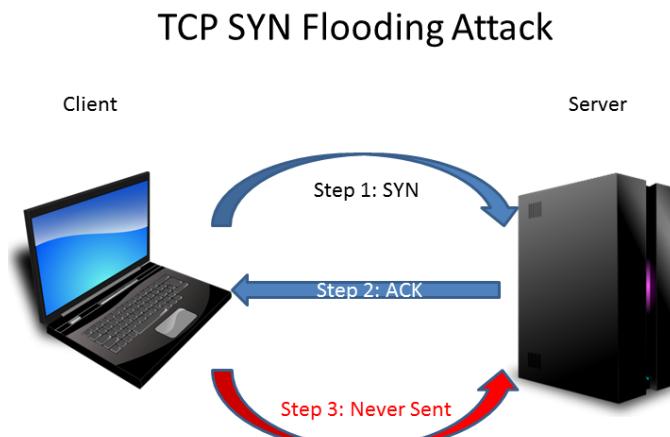
- The handlers are usually very high volume servers
 - Easy to hide the attack packets
- The agents are usually home users with DSL/Cable
 - Already infected and the agent installed
- Very difficult to track down the attacker
- How to differentiate between DDoS and Flash Crowd?
 - Flash Crowd: Many clients using a service legitimately
 - Slashdot Effect
 - Victoria Secret Webcast
 - Generally the flash crowd disappears when the network is flooded
 - Sources in flash crowd are clustered



TCP SYN Flooding

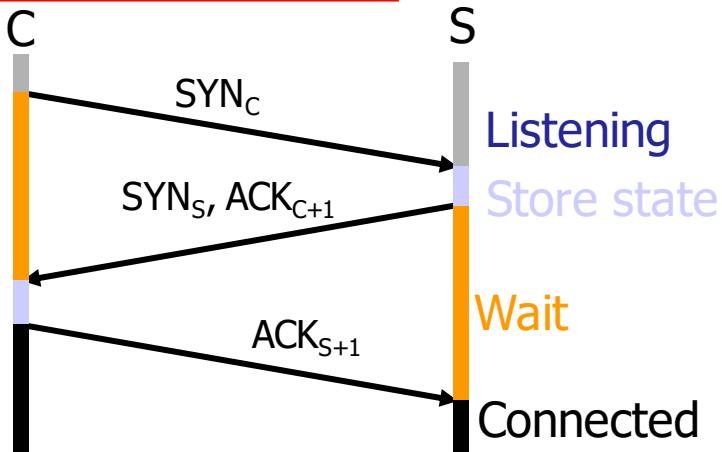
In a server that provides TCP connections for services such as Telnet, Web, Email, etc., lots of half-open TCP connections will cause a problem known as TCP SYN Flooding attack. This problem is due to the TCP 3-way hand-shaking protocol:

A client initiates a TCP connection by sending a TCP SYN packet to the server in Step 1. The server upon receiving the TCP SYN packet replies with an ACK packet in Step 2. However, the client may not send an ACK packet to the server to complete the TCP 3-way hand-shaking protocol. If the client keeps sending the SYN packets, the server will eventually run out of resource to other TCP connection requests.

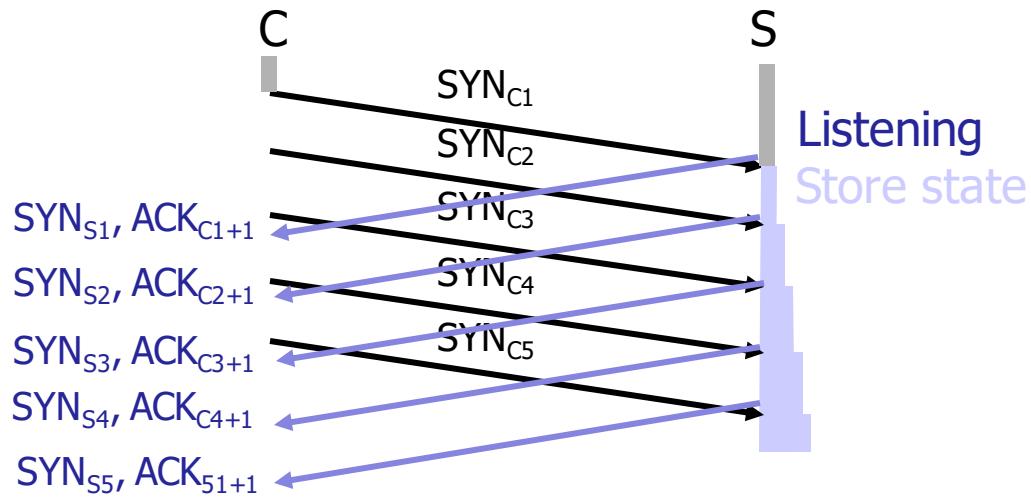


TCP SYN Flooding - Continue

TCP Three Way Handshake for establish connection



SYN Flooding



- The backlog queue is a large memory structure used to handle incoming packets with the SYN flag set until the moment the three-way handshake process is completed.
- An operating system allocates part of the system memory for every incoming connection. Every TCP port can handle a defined number of incoming requests. The backlog queue controls how many half-open connections can be handled by the operating system at the same time.
- When a maximum number of incoming connections is reached, subsequent requests are silently dropped by the operating system.

TCP SYN Flooding - Continue

Identifying the Attacker

The IP address of an attacking system is hidden because the source addresses in the SYN packets are often falsified. When the packet arrives at the server, there is no way to determine its true source IP address. Since the network forwards packets based on destination address, the only way to validate the source of a packet is to use input source filtering in the client side.

Attack Detection

Most of the operating systems provide a command line tool “netstat” to display protocol statistics and current TCP/IP network connections. [The following command running on a Window 7 machine lists network connections.](#) Pay attention to the state column. If there are lots of “SYN_RECEIVED” connections, the system is under attack. The SYN_RECEIVE state indicates that a connection request has been received from the network.

How to detect a TCP SYN attack

- The netstat command shows how many connections are currently in the half-open state. The half-open state is described as SYN_RECEIVED in Windows and as SYN_RECV in Unix systems.

```
# netstat -n -p TCP
```

- tcp	0	0	10.100.0.200:21	237.177.154.8:25882	SYN_RECV
- tcp	0	0	10.100.0.200:21	236.15.133.204:2577	SYN_RECV
- tcp	0	0	10.100.0.200:21	127.160.6.129:51748	SYN_RECV
- tcp	0	0	10.100.0.200:21	230.220.13.25:47393	SYN_RECV
- tcp	0	0	10.100.0.200:21	227.200.204.182:60427	SYN_RECV
- tcp	0	0	10.100.0.200:21	232.115.18.38:278	SYN_RECV
- tcp	0	0	10.100.0.200:21	229.116.95.96:5122	SYN_RECV
- tcp	0	0	10.100.0.200:21	236.219.139.207:49162	SYN_RECV
- tcp	0	0	10.100.0.200:21	238.100.72.228:37899	SYN_RECV - ...

- How many half-open connections are in the backlog queue at the moment can be counted. In the example below, 769 connections (for TELNET) in the SYN RECEIVED state are kept in the backlog queue.

```
# netstat -n -p TCP | grep SYN_RECV | grep :23 | wc -l 769
```

- The other method for detecting SYN attacks is to print TCP statistics and look at the TCP parameters which count dropped connection requests.
- [TcpHalfOpenDrop parameter on a Sun Solaris machine.](#)

```
# netstat -s -P tcp | grep tcpHalfOpenDrop      tcpHalfOpenDrop = 473
```
- It is important to note that every TCP port has its own backlog queue, but only one variable of the TCP/IP stack controls the size of backlog queues for all ports.

Built-in Protection for SYN Flooding

The most important parameter in Windows 2000 and also in Windows Server 2003 is SynAttackProtect. Enabling this parameter allows the operating system to handle incoming connections more efficiently. The protection can be set by adding a SynAttackProtect DWORD value to the following registry key:

HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

- When a SYN attack is detected the SynAttackProtect parameter changes the behavior of the TCP/IP stack. This allows the operating system to handle more SYN requests. It works by disabling some socket options, adding additional delays to connection indications and changing the timeout for connection requests. When the value of SynAttackProtect is set to 1, the number of retransmissions is reduced. The recommended value of SynAttackProtect is 2, which additionally delays the indication of a connection to the Windows Socket until the three-way handshake is completed.
- By enabling the SynAttackProtect parameter we don't change the TCP/IP stack behavior until under a SYN attack. But even then, when SynAttackProtect starts to operate, the operating system can handle legitimate incoming connections.

Built-in Protection for SYN Flooding - Continue

The operating system enables protection against SYN attacks automatically when it detects that values of the following three parameters are exceeded. These parameters are TcpMaxHalfOpen, TcpMaxHalfOpenRetried and TcpMaxPortsExhausted. To change the values of these parameters, first we have to add them to the same registry key as we made for SynAttackProtect.

- ❖ **TcpMaxHalfOpen** registry entry defines the maximum number of SYN RECEIVED states which can be handled concurrently before SYN protection starts working. The recommended value of this parameter is 100 for Windows 2000 Server and 500 for Windows 2000 Advanced Server.
- ❖ **TcpMaxHalfOpenRetried** defines the maximum number of half-open connections, for which the operating system has performed at least one retransmission, before SYN protection begins to operate. The recommended value is 80 for Windows 2000 Server, and 400 for Advanced Server.
- ❖ **TcpMaxPortsExhausted** registry entry defines the number of dropped SYN requests, after which the protection against

Other Attacks in Transport Layer

1. Session hijacking

This kind of attack occurs after a source and destination computer have established a communications link. A third computer disables the ability of one the computers to communicate, and then imitates that computer. Because the connection has already been established, the third computer can disrupt the C-I-A (confidentiality integrity and availability) triad.

Protection against session hijacking

- Use SSL/HTTPS encryption for the entire web site, and you have the best guarantee that no man in the middle attacks will be able to sniff an existing client session cookie
- use some sort of encryption on the session value itself that is stored in your session cookie

Protection against session hijacking Using HTTPS/SSL

HTTPS is a combination of the standard HTTP protocol and the cryptographic security of the SSL protocol. The HTTPS protocol contains mechanisms for secure identification of the server and encryption of the client-server communication.

- Obtain an SSL certificate from a Certificate Authority (CA). A CA is third party that the client trusts to verify that the site using the certificate is indeed the owner of the certificate. There are many CAs to choose from, Google provides a list of popular CAs. Then configure Internet Information Service (IIS) so that the site uses the certificate.
- Make Sure That the Session Cookie is Sent Over an Encrypted Connection
- In order to fully safeguard against session hijacking, make sure that all communication where the session cookie is sent is encrypted. There are two options to achieve this:

Option 1 - Force SSL at All Times

Option 2 – Only Send Session Cookies Over SSL

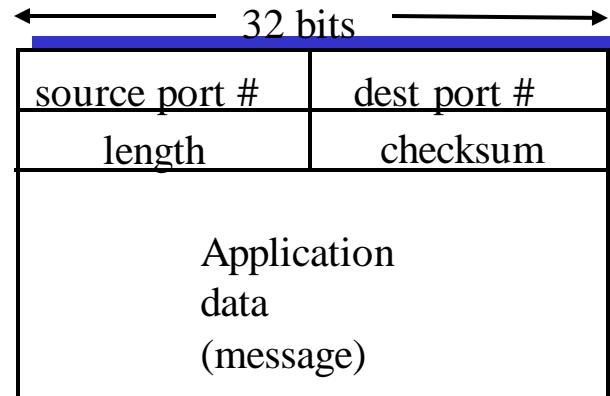
By setting requireSSL="true" on the forms-element in web.config, specify that the session cookie should only be sent when using the HTTPS protocol.

This approach enables to use SSL only on parts of the site (edit/admin for example) and allow non-encrypted communication when browsing the public parts of the site.

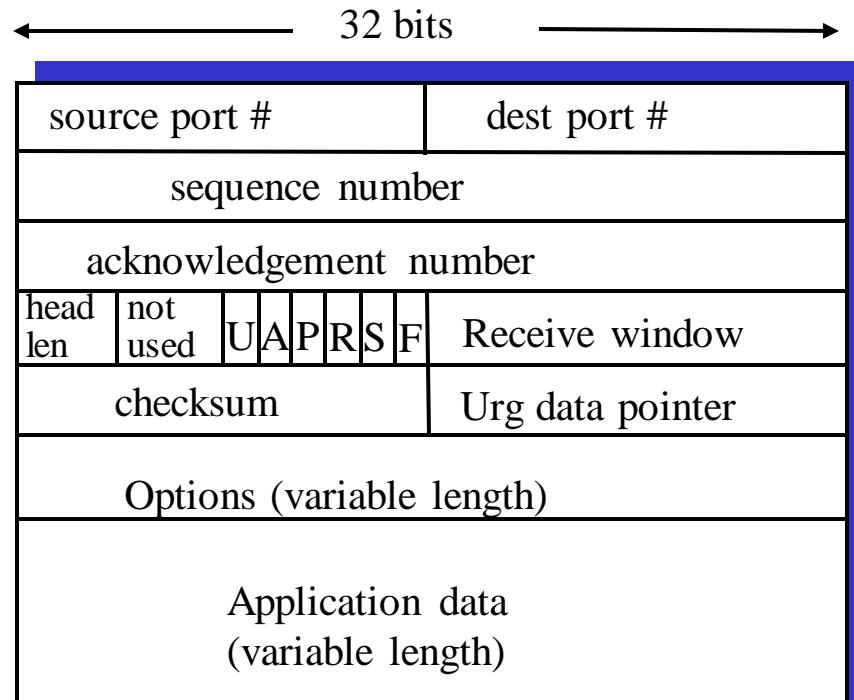
2. TCP Connection Spoofing

At the Transport layer, either a UDP or TCP header is added to the message. By knowing the UDP or TCP header fields and lengths, the ports that are used for communications between a source and destination computer can be identified, and that information can be corrupted or exploited.

- If attacker knows initial seq # and amount of traffic sent, it can estimate likely current values
- Send a flood of packets with likely seq numbers
- Attacker can inject packets into existing connection



UDP segment format



TCP segment format

How TCP connection spoofing works

Injecting IP packets which seems to originate from another host is insufficient to impersonate that host during a TCP connection, because every TCP segment has a 32bit sequence number. A segment with a sequence number which is out of line will be ignored. That means **in order to successfully insert a TCP segment into an existing transmission it needs to guess the next sequence number**, otherwise the segment will be discarded.

- **This isn't so hard when the attacker can eavesdrop at least on the client; otherwise, it can only brute-force the sequence number.**
- With more simple transport protocols, like UDP for example, the attacker doesn't have problem. UDP has no sequence numbers, so unless an upper protocol layer replicates the functionality similar to sequence numbers, it can just insert additional segments which will then be treated as if they were coming from the real host.
- **If the attacker doesn't know existing connection and instead want to establish a new TCP connection which appears to originate from another host, a 3-way handshake is required** (client sends SYN, server sends ACK, client sends SYNACK). The ACK by the server includes a random number which the attack needs for an acceptable SYNACK. So when it can only send IP packets but not receive any of the packets intended for the spoofed host, **attacker will have to guess this random number.**

Transport Layer Solution

The transport layer represents the last place that segments can be authenticated before they affect connection management.

TCP has a variety of current and proposed mechanisms to increase the authentication of segments, protecting against both off-path and on-path third-party spoofing attacks. Other transport protocols, such as SCTP and DCCP, also have limited antispoofing mechanisms.

1. TCP MD5 Authentication
2. TCH RST window attenuation
3. TCP timestamp authentication

References

- B. A. Forouzan, “Cryptography and Network Security,” Mc Graw Hill, 2006
- W. Stallings, “Cryptography and Network Security,” Pearson, 2014
- J. F. Kurose, K. W. Ross, “Computer Networking,” Addison Wesley,
- Ahmad Al-Ghoul, “TCP/IP layers and vulnerabilities (module 9),” Philadelphia University, 2011.
- J. Mitchell, “Network Protocols and Vulnerabilities,”
<http://www.slideserve.com/larya/network-protocols-and-vulnerabilities>, Stanford University.

System Security

Module-VI

Contents

Intrusion Detection System

Firewalls

Malicious software

Wireless Security, Blockchain Cryptocurrencies and the
Dark Web

Intrusion Detection

Intrusion and Intrusion Detection

- Intrusion : Attempting to break into or misuse your system.
- Intruders may be from outside the network or legitimate users of the network.
- Intrusion can be a physical, system or remote intrusion.

Intrusion Detection Systems (IDS)

Intrusion Detection Systems look for attack signatures, which are specific patterns that usually indicate malicious or suspicious intent.

Intrusion Detection Systems (IDS)

- Different ways of classifying an IDS
 - IDS based on
 - anomaly detection
 - signature based misuse
 - host based
 - network based

Anomaly based IDS

- This IDS models the normal usage of the network as a noise characterization.
- Anything distinct from the noise is assumed to be an intrusion activity.
 - E.g flooding a host with lots of packet.
- The primary strength is its ability to recognize novel attacks.

Drawbacks of Anomaly detection IDS

- Assumes that intrusions will be accompanied by manifestations that are sufficiently unusual so as to permit detection.
- These generate many false alarms and hence compromise the effectiveness of the IDS.

Signature based IDS

- This IDS possess an attacked description that can be matched to sensed attack manifestations.
- The question of what information is relevant to an IDS depends upon what it is trying to detect.
 - E.g DNS, FTP etc.

Signature based IDS (contd.)

- ID system is programmed to interpret a certain series of packets, or a certain piece of data contained in those packets, as an attack. For example, an IDS that watches web servers might be programmed to look for the string “phf” as an indicator of a CGI program attack.
- Most signature analysis systems are based off of simple pattern matching algorithms. In most cases, the IDS simply looks for a sub string within a stream of data carried by network packets. When it finds this sub string (for example, the ``phf" in ``GET /cgi-bin/phf?"), it identifies those network packets as vehicles of an attack.

Drawbacks of Signature based IDS

- They are unable to detect novel attacks.
- Suffer from false alarms
- Have to programmed again for every new pattern to be detected.

Host/Applications based IDS

- The host operating system or the application logs in the audit information.
- These audit information includes events like the use of identification and authentication mechanisms (logins etc.) , file opens and program executions, admin activities etc.
- This audit is then analyzed to detect trails of intrusion.

Drawbacks of the host based IDS

- The kind of information needed to be logged in is a matter of experience.
- Unselective logging of messages may greatly increase the audit and analysis burdens.
- Selective logging runs the risk that attack manifestations could be missed.

Strengths of the host based IDS

- Attack verification
- System specific activity
- Encrypted and switch environments
- Monitoring key components
- Near Real-Time detection and response.
- No additional hardware

Stack based IDS

- They are integrated closely with the TCP/IP stack, allowing packets to be watched as they traverse their way up the OSI layers.
- This allows the IDS to pull the packets from the stack before the OS or the application have a chance to process the packets.

Network based IDS

- This IDS looks for attack signatures in network traffic via a promiscuous interface.
- A filter is usually applied to determine which traffic will be discarded or passed on to an attack recognition module. This helps to filter out known un-malicious traffic.

Strengths of Network based IDS

- Cost of ownership reduced
- Packet analysis
- Evidence removal
- Real time detection and response
- Malicious intent detection
- Complement and verification
- Operating system independence

Commercial ID Systems

- ISS – Real Secure from Internet Security Systems:
 - Real time IDS.
 - Contains both host and network based IDS.
- Tripwire – File integrity assessment tool.
- Bro and Snort – open source public-domain system.

Future of IDS

- To integrate the network and host based IDS for better detection.
- Developing IDS schemes for detecting novel attacks rather than individual instantiations.

Firewalls

Outline

- Firewall Design Principles
 - Firewall Characteristics
 - Types of Firewalls
 - Firewall Configurations

Firewalls

- Effective means of protection a local system or network of systems from network-based security threats while affording access to the outside world via WAN's or the Internet

Firewall Design Principles

- The firewall is inserted between the premises network and the Internet
- Aims:
 - Establish a controlled link
 - Protect the premises network from Internet-based attacks
 - Provide a single choke point

Firewall Characteristics

- Design goals:
 - All traffic from inside to outside must pass through the firewall (physically blocking all access to the local network except via the firewall)
 - Only authorized traffic (defined by the local security police) will be allowed to pass

Firewall Characteristics

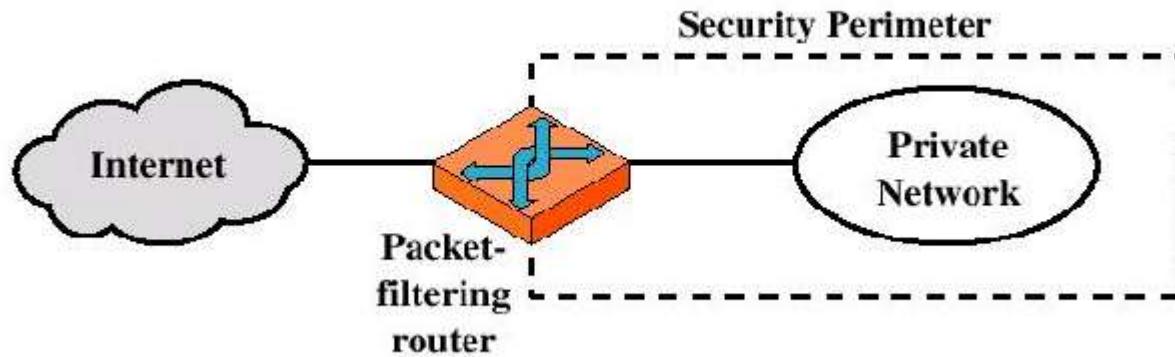
- Design goals:
 - The firewall itself is immune to penetration (use of trusted system with a secure operating system)

Types of Firewalls

- Three common types of Firewalls:
 - Packet-filtering routers
 - Application-level gateways
 - Circuit-level gateways

Types of Firewalls

- Packet-filtering Router

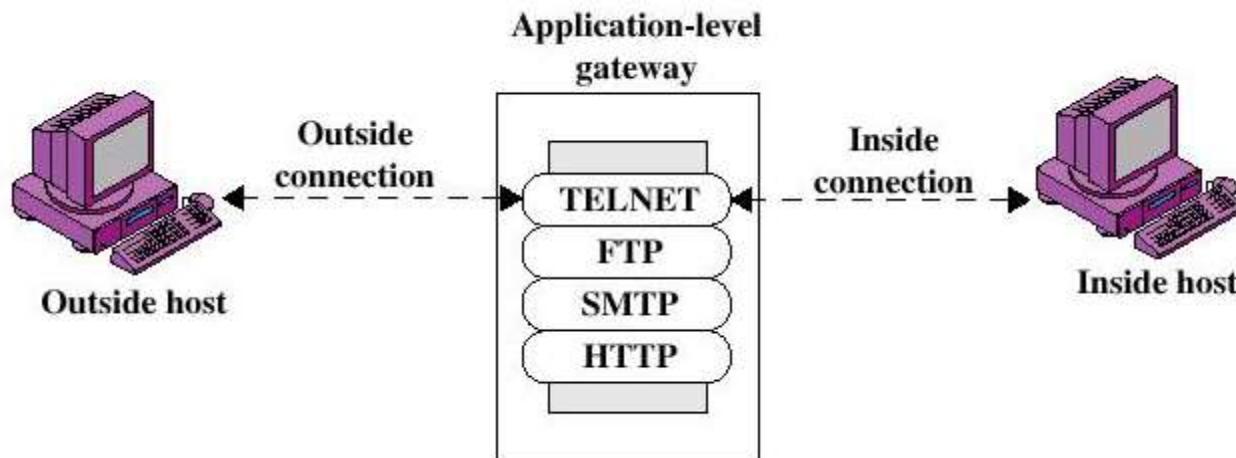


Types of Firewalls

- Packet-filtering Router
 - Applies a set of rules to each incoming IP packet and then forwards or discards the packet
 - Filter packets going in both directions
 - The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header
 - Two default policies (discard or forward)

Types of Firewalls

- Application-level Gateway



Types of Firewalls

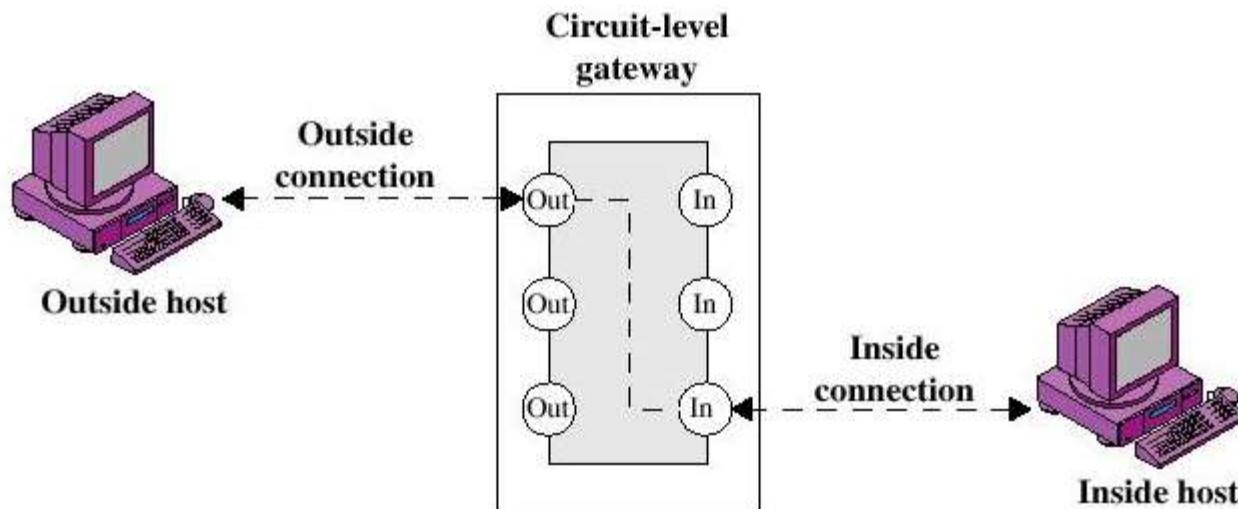
- Application-level Gateway
 - Also called proxy server
 - Acts as a relay of application-level traffic

Types of Firewalls

- Advantages:
 - Higher security than packet filters
 - Only need to scrutinize a few allowable applications
 - Easy to log and audit all incoming traffic
- Disadvantages:
 - Additional processing overhead on each connection (gateway as splice point)

Types of Firewalls

- Circuit-level Gateway



Types of Firewalls

- Circuit-level Gateway
 - Stand-alone system or
 - Specialized function performed by an Application-level Gateway
 - Sets up two TCP connections
 - The gateway typically relays TCP segments from one connection to the other without examining the contents

Malicious Software

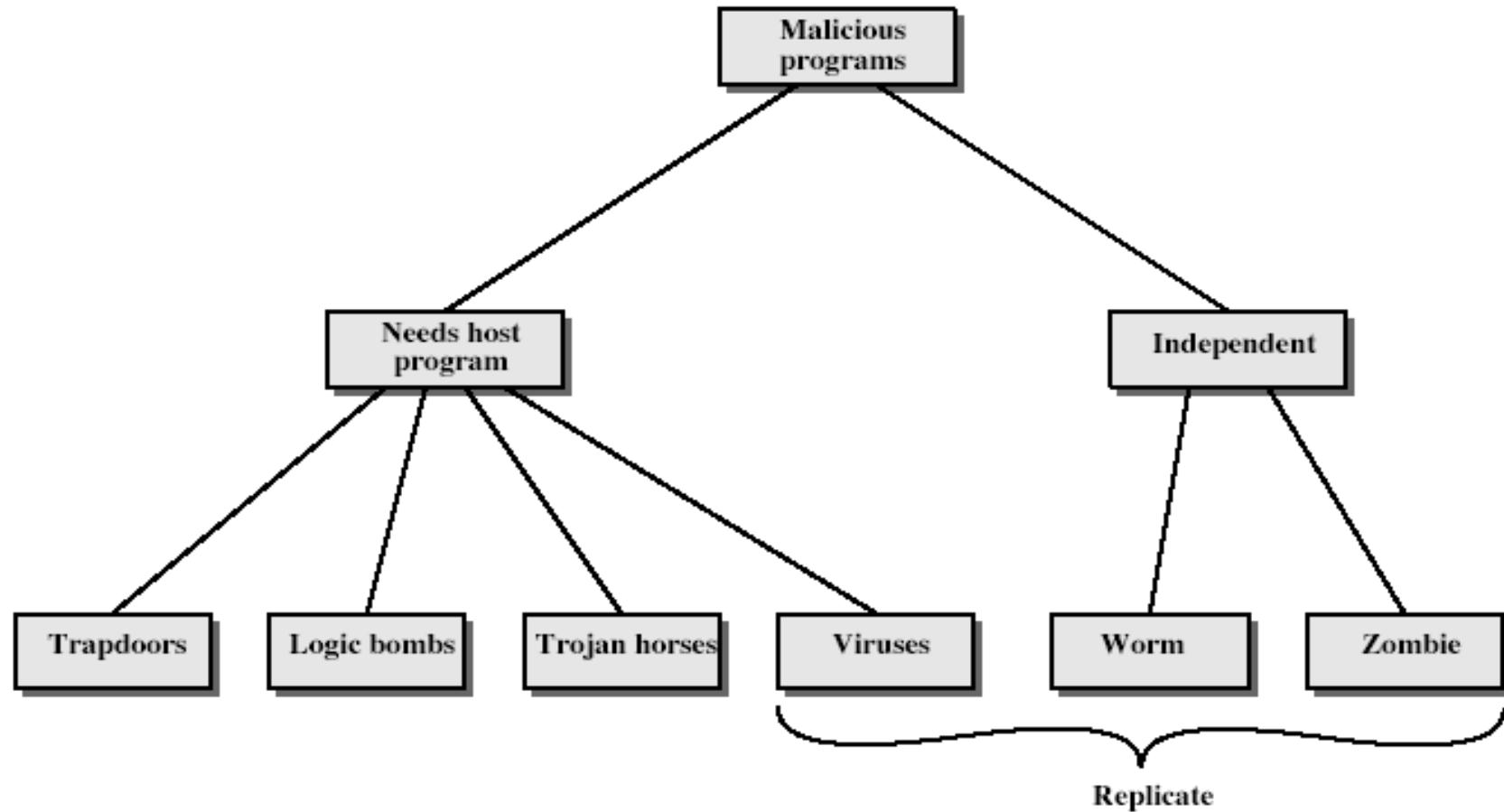
What is the concept of defense: The parrying of a blow. What is its characteristic feature: Awaiting the blow.

—On War, Carl Von Clausewitz

Viruses and Other Malicious Content

- computer viruses have got a lot of publicity
- one of a family of **malicious software**
- effects usually obvious
- have figured in news reports, fiction, movies (often exaggerated)
- getting more attention than deserve
- are a concern though

Malicious Software



Trapdoors

- secret entry point into a program
- allows those who know access bypassing usual security procedures
- have been commonly used by developers
- a threat when left in production programs allowing exploited by attackers
- very hard to block in O/S
- requires good s/w development & update

Logic Bomb

- one of oldest types of malicious software
- code embedded in legitimate program
- activated when specified conditions met
 - eg presence/absence of some file
 - particular date/time
 - particular user
- when triggered typically damage system
 - modify/delete files/disks

Trojan Horse

- program with hidden side-effects
- which is usually superficially attractive
 - eg game, s/w upgrade etc
- when run performs some additional tasks
 - allows attacker to indirectly gain access they do not have directly
- often used to propagate a virus/worm or install a backdoor
- or simply to destroy data

Zombie

- program which secretly takes over another networked computer
- then uses it to indirectly launch attacks
- often used to launch distributed denial of service (DDoS) attacks
- exploits known flaws in network systems

Viruses

- a piece of self-replicating code attached to some other code
 - cf biological virus
- both propagates itself & carries a payload
 - carries code to make copies of itself
 - as well as code to perform some covert task

Virus Operation

- virus phases:
 - dormant – waiting on trigger event
 - propagation – replicating to programs/disks
 - triggering – by event to execute payload
 - execution – of payload
- details usually machine/OS specific
 - exploiting features/weaknesses

Types of Viruses

- can classify on basis of how they attack
- parasitic virus
- memory-resident virus
- boot sector virus
- stealth
- polymorphic virus
- macro virus

Macro Virus

- **macro code** attached to some **data file**
- interpreted by program using file
 - eg Word/Excel macros
 - esp. using auto command & command macros
- code is now platform independent
- is a major source of new viral infections
- blurs distinction between data and program files making task of detection much harder
- classic trade-off: "ease of use" vs "security"

Email Virus

- spread using email with attachment containing a macro virus
 - cf Melissa
- triggered when user opens attachment
- or worse even when mail viewed by using scripting features in mail agent
- usually targeted at Microsoft Outlook mail agent & Word/Excel documents

Worms

- replicating but not infecting program
- typically spreads over a network
 - cf Morris Internet Worm in 1988
 - led to creation of CERTs
- using users distributed privileges or by exploiting system vulnerabilities
- widely used by hackers to create **zombie PC's**, subsequently used for further attacks, esp DoS
- major issue is lack of security of permanently connected systems, esp PC's

Worm Operation

- worm phases like those of viruses:
 - dormant
 - propagation
 - search for other systems to infect
 - establish connection to target remote system
 - replicate self onto remote system
 - triggering
 - execution

Morris Worm

- best known classic worm
- released by Robert Morris in 1988
- targeted Unix systems
- using several propagation techniques
 - simple password cracking of local pw file
 - exploit bug in finger daemon
 - exploit debug trapdoor in sendmail daemon
- if any attack succeeds then replicated self

Recent Worm Attacks

- new spate of attacks from mid-2001
- **Code Red**
 - exploited bug in MS IIS to penetrate & spread
 - probes random IPs for systems running IIS
 - had trigger time for denial-of-service attack
 - 2nd wave infected 360000 servers in 14 hours
- **Code Red 2**
 - had backdoor installed to allow remote control
- **Nimda**
 - used multiple infection mechanisms
 - email, shares, web client, IIS, Code Red 2 backdoor

Virus Countermeasures

- viral attacks exploit lack of integrity control on systems
- to defend need to add such controls
- typically by one or more of:
 - **prevention** - block virus infection mechanism
 - **detection** - of viruses in infected system
 - **reaction** - restoring system to clean state

Thank you

Module-VI: Malware: Malicious Software

Viruses, Worms, Trojans, Rootkits

- **Malware** can be classified into several categories, depending on propagation and concealment
- Propagation
 - **Virus**: human-assisted propagation (e.g., open email attachment)
 - **Worm**: automatic propagation without human assistance
- Concealment
 - **Rootkit**: modifies operating system to hide its existence
 - **Trojan**: provides desirable functionality but hides malicious operation
- Various types of payloads, ranging from annoyance to crime

Insider Attacks

- An **insider attack** is a security breach that is caused or facilitated by someone who is a part of the very organization that controls or builds the asset that should be protected.
- In the case of malware, an insider attack refers to a security hole that is created in a software system by one of its programmers.

Backdoors

- A **backdoor**, which is also sometimes called a **trapdoor**, is a hidden feature or command in a program that allows a user to perform actions he or she would not normally be allowed to do.
- When used in a normal way, this program performs completely as expected and advertised.
- But if the hidden feature is activated, the program does something unexpected, often in violation of security policies, such as performing a privilege escalation.

Non-malicious Backdoors

- Some backdoors are put into a program by its programmers
 - Debugging purpose (bypass some tedious steps to speed up debugging)
 - Many computer games have backdoors
 - Secret key code to change gaming role (full health, full map, invincible)

Malicious Backdoors

- Deliberate backdoors inserted by malicious programmers
 - Blackmail, secret privilege
- Backdoor created by malware on compromised machines
 - Open a TCP listening service, anyone can have a shell connection to the machine without account and password
 - Example: Code Red II

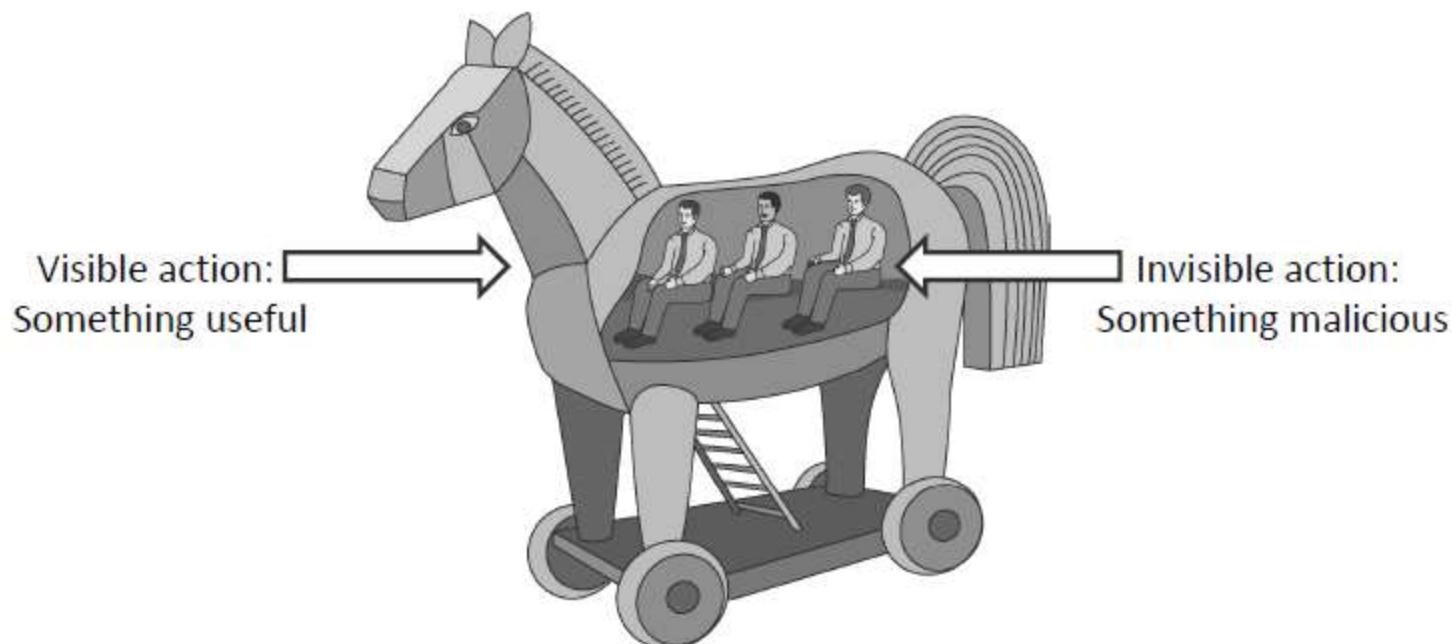
Logic Bombs

- A **logic bomb** is a program that performs a malicious action as a result of a certain logic condition.
- The classic example of a logic bomb is a programmer coding up the software for the payroll system who puts in code that makes the program crash should it ever process two consecutive payrolls without paying him.
- Another classic example combines a logic bomb with a backdoor, where a programmer puts in a logic bomb that will crash the program on a certain date.



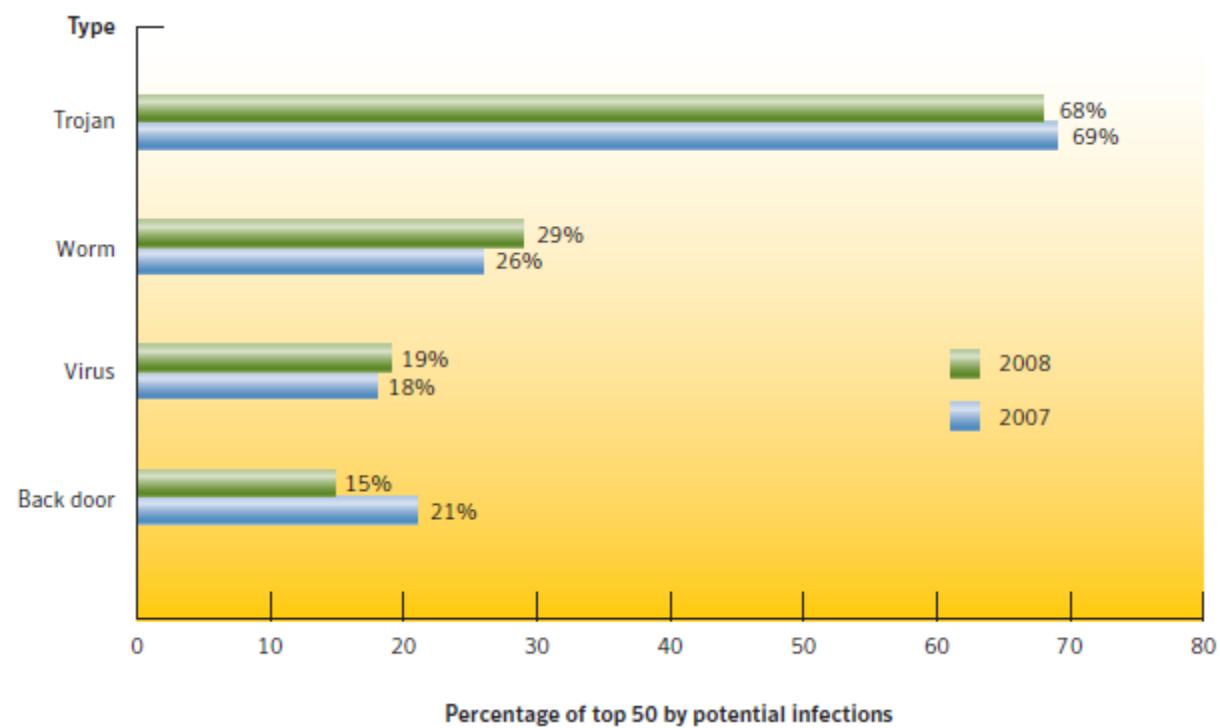
Trojan Horses

- A **Trojan horse (or Trojan)** is a malware program that appears to perform some useful task, but which also does something with negative consequences (e.g., launches a keylogger).
- Trojan horses can be installed as part of the payload of other malware but are often installed by a user or administrator, either deliberately or accidentally.



Current Trends

- Trojans currently have largest infection potential
 - Often exploit browser vulnerabilities
 - Typically used to download other malware in multi-stage attacks

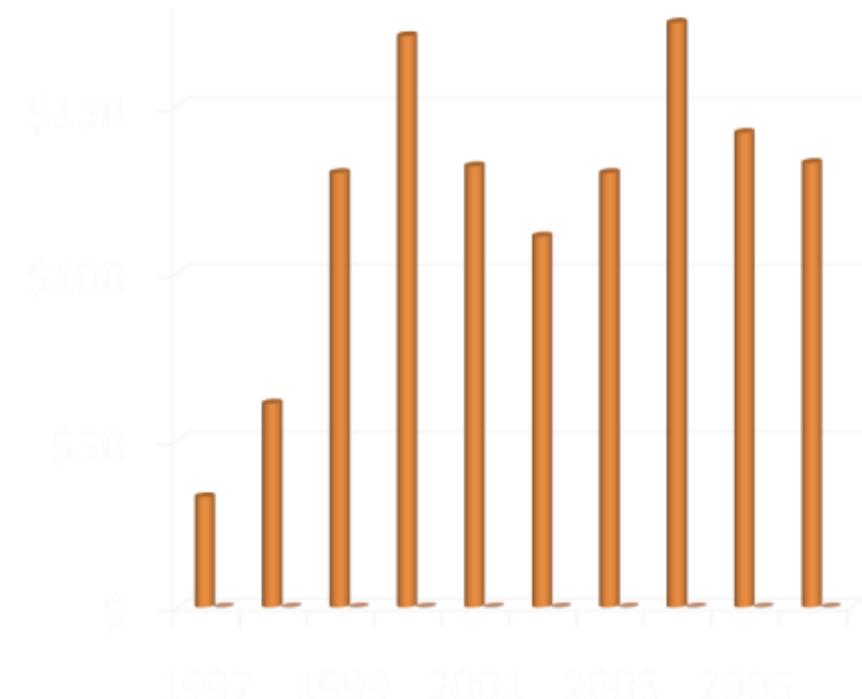


Source:
Symantec Internet
Security Threat
Report, April 2009

Financial Impact

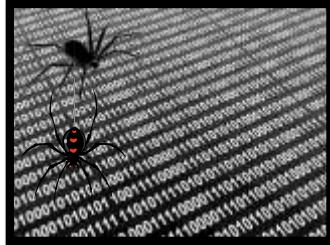
- Malware often affects a large user population
- Significant financial impact, though estimates vary widely, up to \$100B per year (mi2g)
- Examples
 - LoveBug (2000) caused \$8.75B in damages and shut down the British parliament
 - In 2004, 8% of emails infected by W32/MyDoom.A at its peak
 - In February 2006, the Russian Stock Exchange was taken down by a virus.

Source: Computer Economics



Adware

Adware software payload

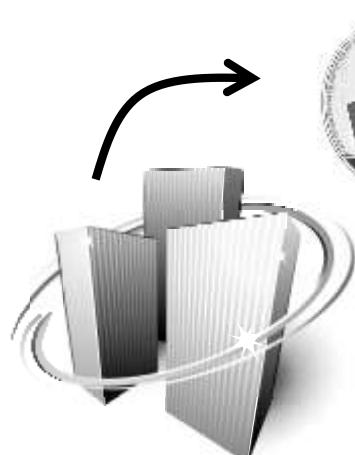


Adware engine infects
a user's computer

Computer user



Advertisers contract with
adware agent for content



Adware engine requests
advertisements
from adware agent



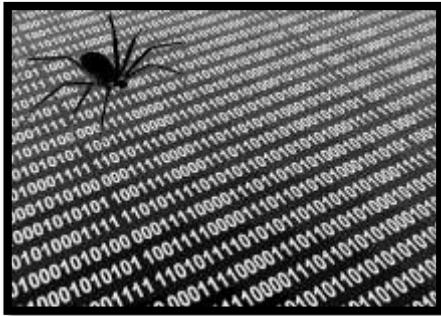
Adware agent



Adware agent delivers
ad content to user

Spyware

Spyware software payload



1. Spyware engine infects a user's computer.

Computer user



2. Spyware process collects keystrokes, passwords, and screen captures.



3. Spyware process periodically sends collected data to spyware data collection agent.

Spyware data collection agent

Malware

Signatures: A Malware Countermeasure

- Scan compare the analyzed object with a database of signatures
- A **signature** is a virus fingerprint
 - E.g., a string with a sequence of instructions specific for each virus
 - Different from a digital signature
- A file is infected if there is a signature inside its code
 - Fast **pattern matching** techniques to search for signatures
- All the signatures together create the malware database that usually is proprietary

White/Black Listing

- Maintain database of cryptographic hashes for
 - Operating system files
 - Popular applications
 - Known infected files
- Compute hash of each file in hard drives
- Look up into database to compare
- Needs to protect the integrity of the database
- Example: TripWire software

Quarantine

- A suspicious file can be isolated in a folder called **quarantine**:
 - E.g., if the result of the heuristic analysis is positive and you are waiting for db signatures update
- The suspicious file is not deleted but made harmless: the user can decide when to remove it or eventually restore for a false positive
 - Interacting with a file in quarantine it is possible only through the antivirus program
- The file in quarantine is harmless because it is encrypted
- Usually the quarantine technique is proprietary and the details are kept secret