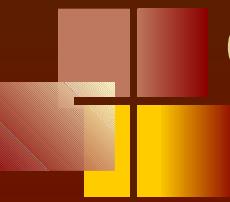


Cryptography and Network Security

Chapter 1

Introduction



Chapter 1

Objectives

- To define three security goals
- To define security attacks that threaten security goals
- To define security services and how they are related to the three security goals
- To define security mechanisms to provide security services
- To introduce two techniques, cryptography and steganography, to implement security mechanisms.

Definitions

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** - measures to protect data during their transmission
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

1-1 SECURITY GOALS

This section defines three security goals.

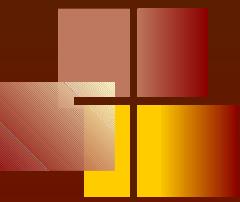
Topics discussed in this section:

1. Confidentiality
2. Integrity
3. Availability

1.1 *Continued*

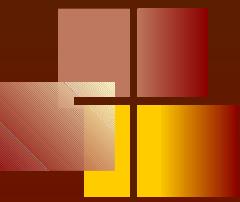
Figure 1.1 *Taxonomy of security goals*





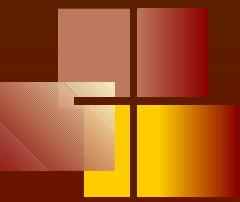
1.1.1 Confidentiality

Confidentiality is probably the most common aspect of information security. We need to protect our confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information.



1.1.2 Integrity

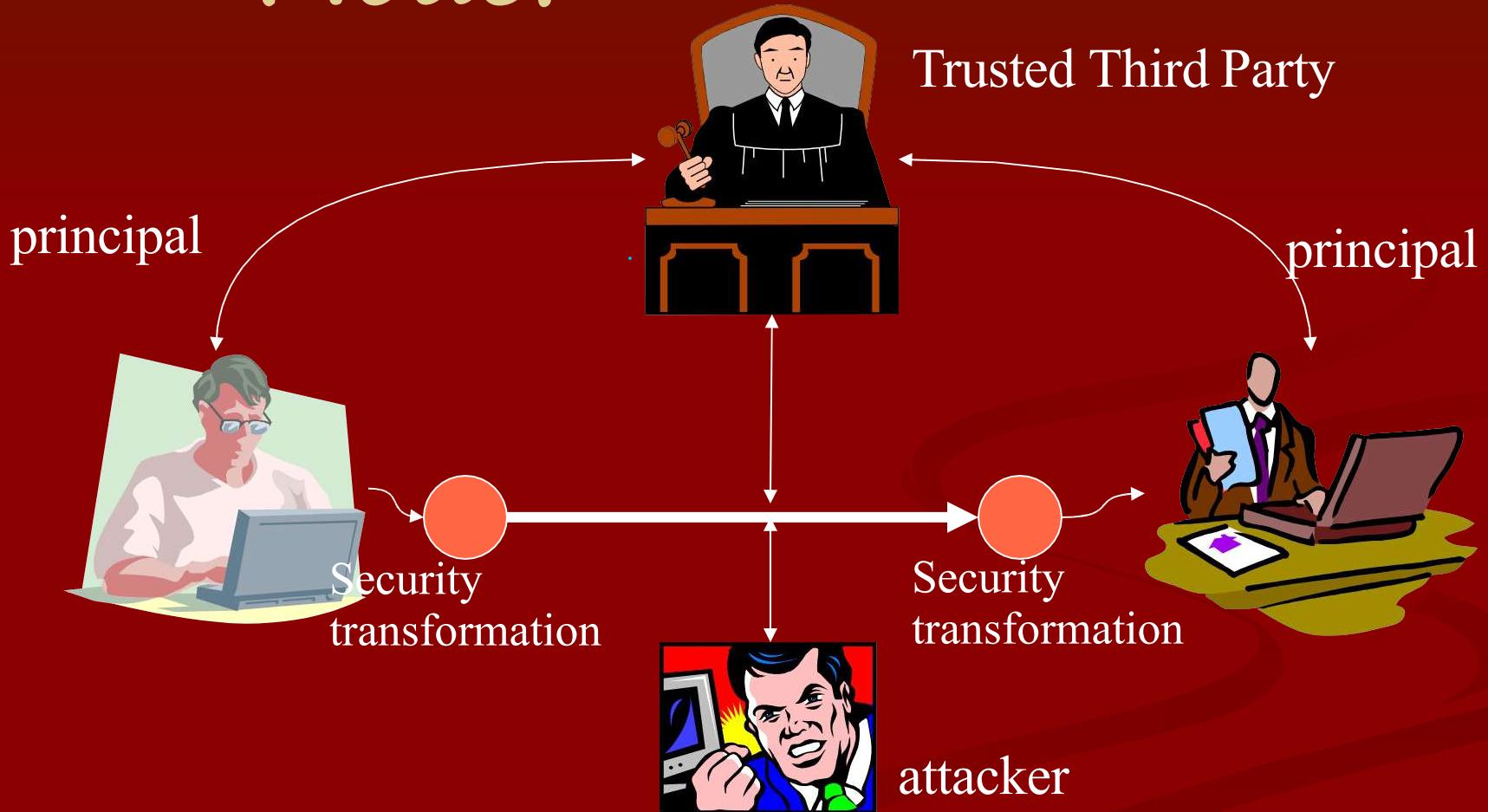
Information needs to be changed constantly. Integrity means that changes need to be done only by authorized entities and through authorized mechanisms.



1.1.3 Availability

The information created and stored by an organization needs to be available to authorized entities. Information needs to be constantly changed, which means it must be accessible to authorized entities.

Network Security Model



1-2 ATTACKS

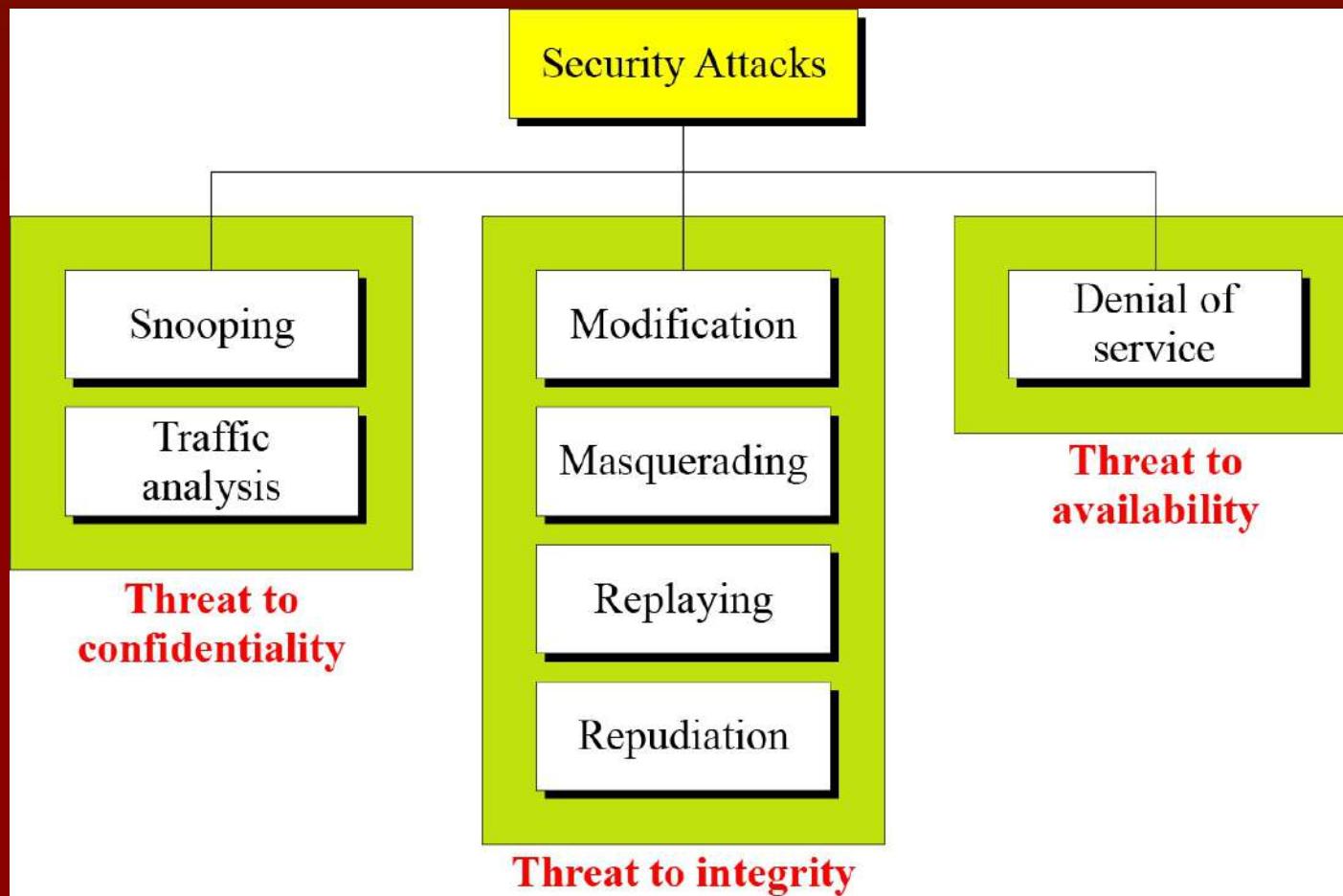
The three goals of security—confidentiality, integrity, and availability—can be threatened by security attacks.

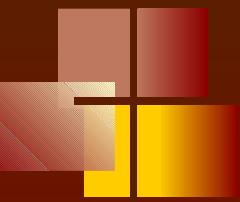
Topics discussed in this section:

1. Attacks Threatening Confidentiality
2. Attacks Threatening Integrity
3. Attacks Threatening Availability
4. Passive versus Active Attacks

1.2 Continued

Figure 1.2 Taxonomy of attacks with relation to security goals





1.2.1 Attacks Threatening Confidentiality

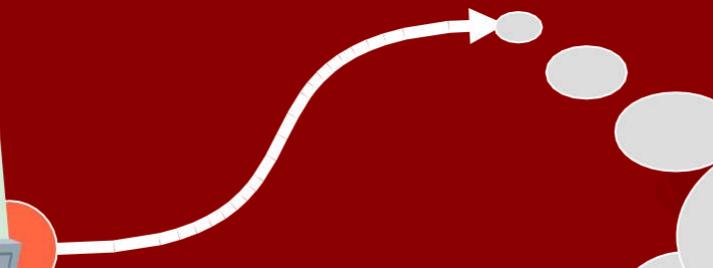
Snooping refers to unauthorized access to or interception of data.

Traffic analysis refers to obtaining some other type of information by monitoring online traffic.

Information Transferring

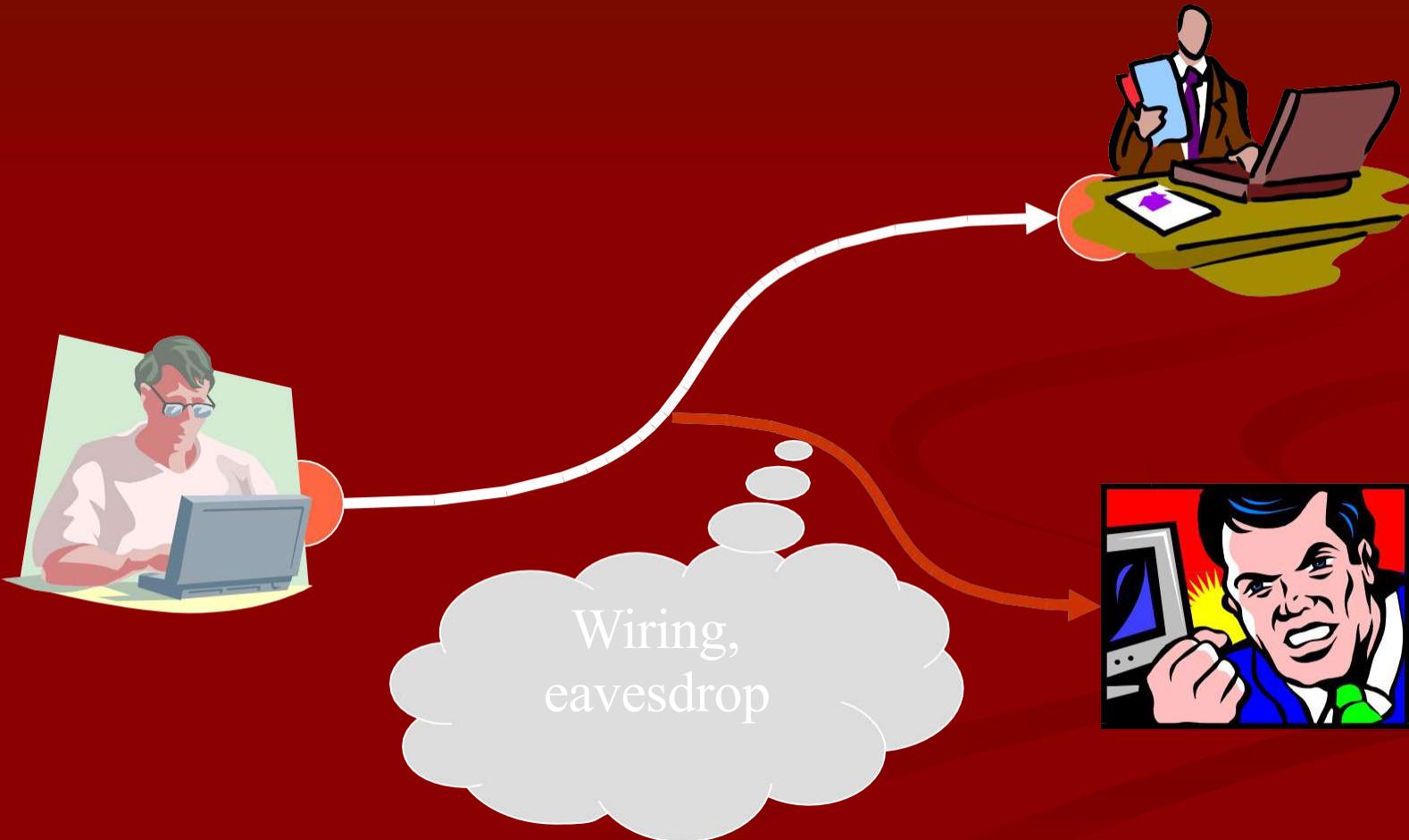


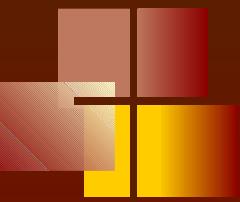
Attack: Interruption



Cut wire lines,
Jam wireless
signals,
Drop packets,

Attack: Interception





1.2.2 Attacks Threatening Integrity

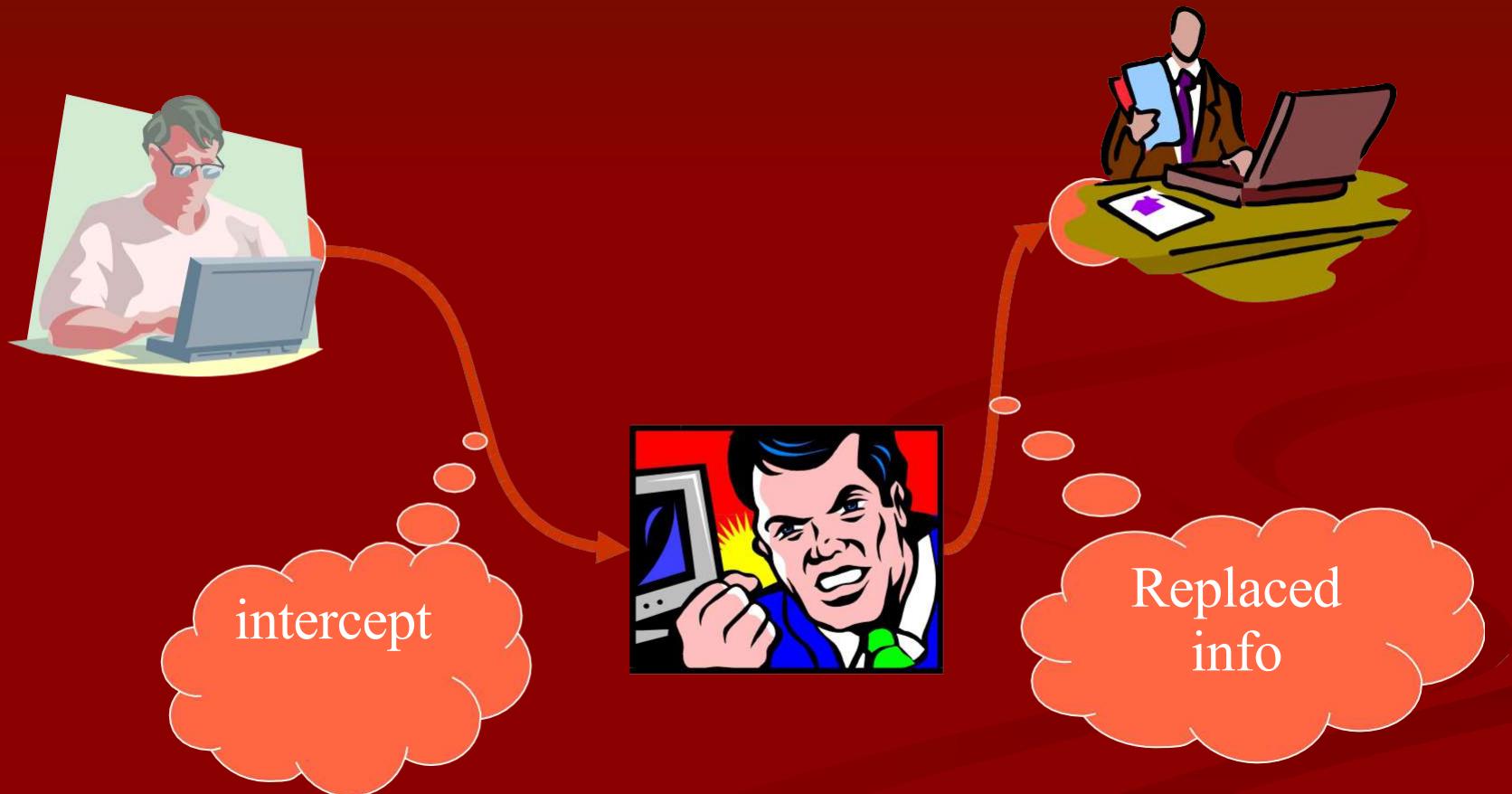
Modification means that the attacker intercepts the message and changes it.

Masquerading or spoofing happens when the attacker impersonates somebody else.

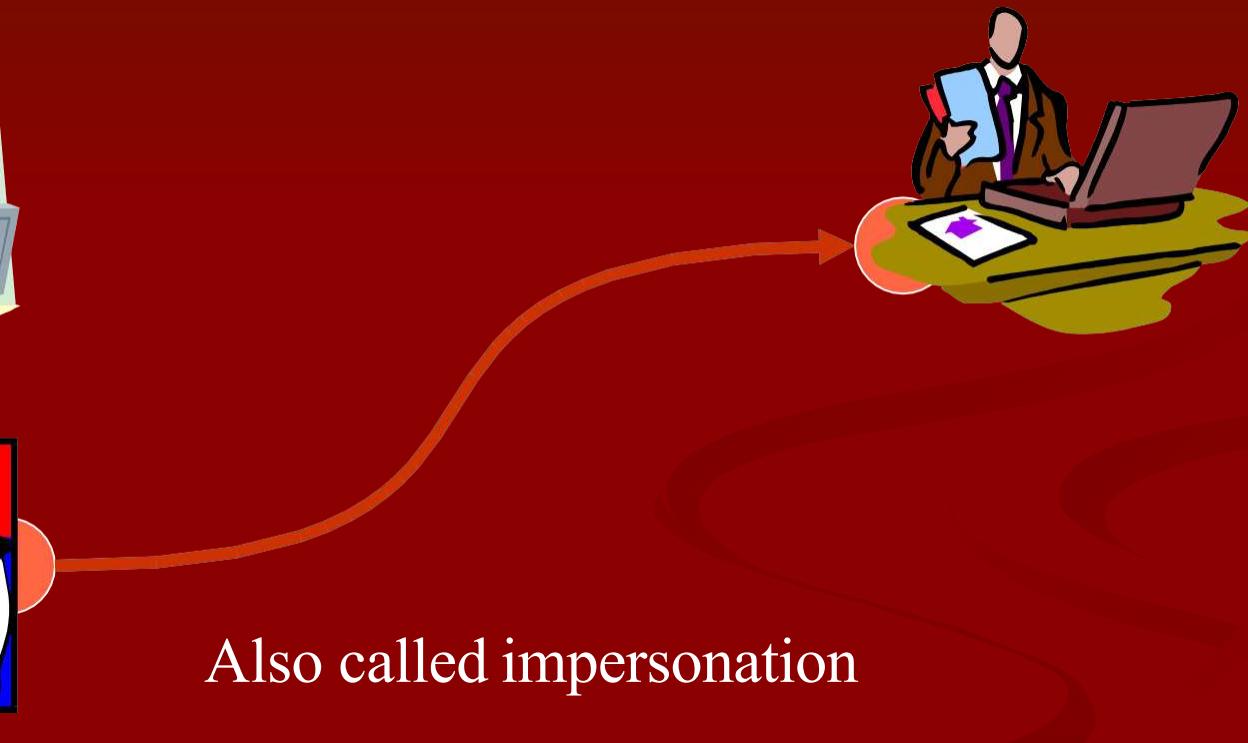
Replaying means the attacker obtains a copy of a message sent by a user and later tries to replay it.

Repudiation means that sender of the message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.

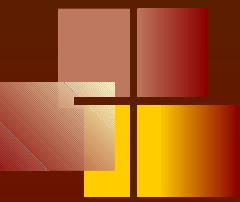
Attack: Modification



Attack: Fabrication



Also called impersonation



1.2.3 Attacks Threatening Availability

Denial of service (DoS) is a very common attack. It may slow down or totally interrupt the service of a system.

1.2.4 *Passive Versus Active Attacks*

Table 1.1 *Categorization of passive and active attacks*

Attacks	Passive/Active	Threatening
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

1-3 SERVICES AND MECHANISMS

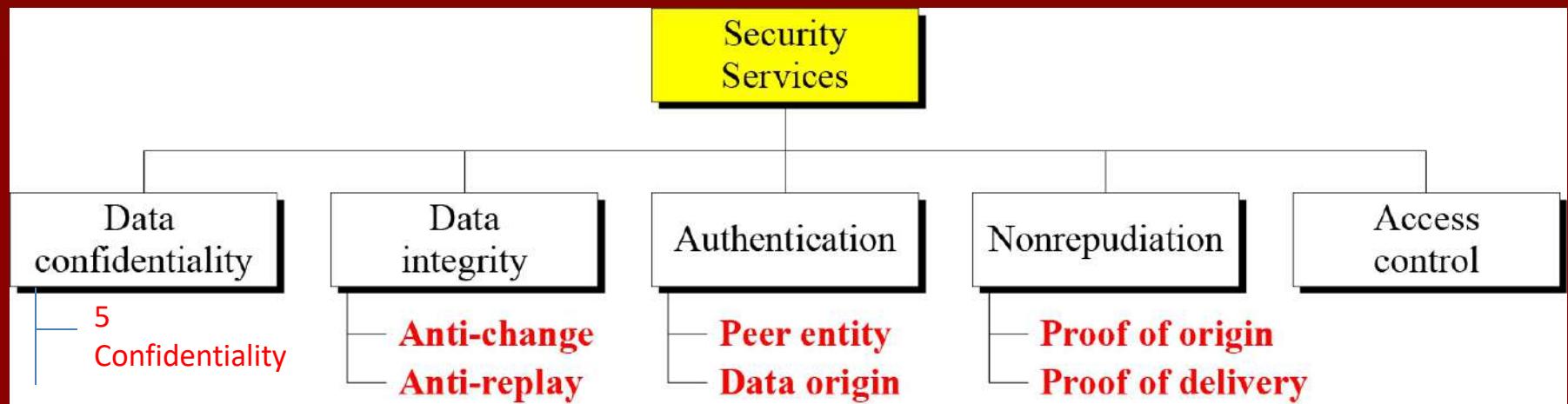
ITU-T provides some security services and some mechanisms to implement those services. Security services and mechanisms are closely related because a mechanism or combination of mechanisms are used to provide a service..

Topics discussed in this section:

1. **Security Services**
2. **Security Mechanism**
3. **Relation between Services and Mechanisms**

1.3.1 Security Services

Figure 1.3 Security services



Security Services (X.800)

1) Authentication -

- Peer Entity authentication.

- Data Origin authentication.

2) Data Confidentiality -

- Connection Confidentiality.

- Connectionless confidentiality.

Security Services (X.800)

3) Data Integrity.

- Connection integrity with recovery.
- Connection integrity without recovery.
- Connectionless integrity.
- Selected field connection Integrity.
- Selected field connectionless Integrity.

Security Services (X.800)

4) Nonrepuditation

- nonrepudiation

Origin.

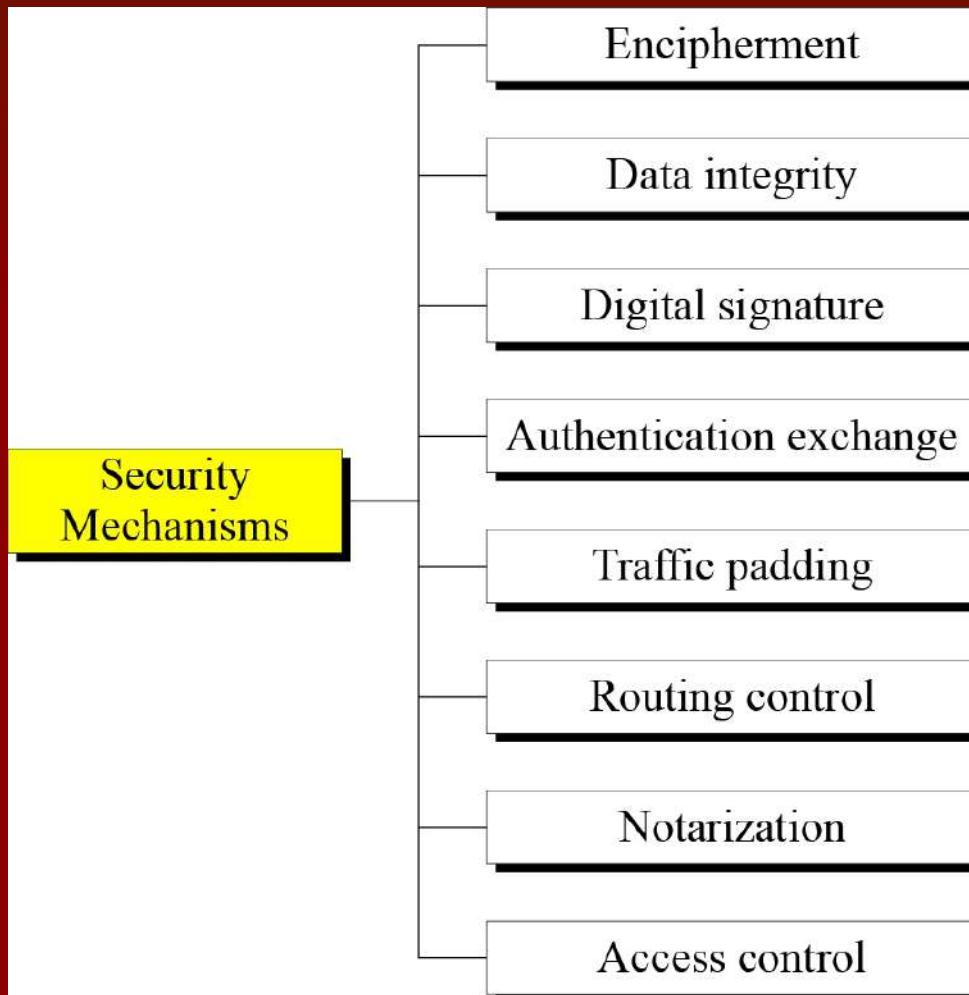
- nonrepudiation

destination.

5) Access Control

1.3.2 Security Mechanism

Figure 1.4 Security mechanisms

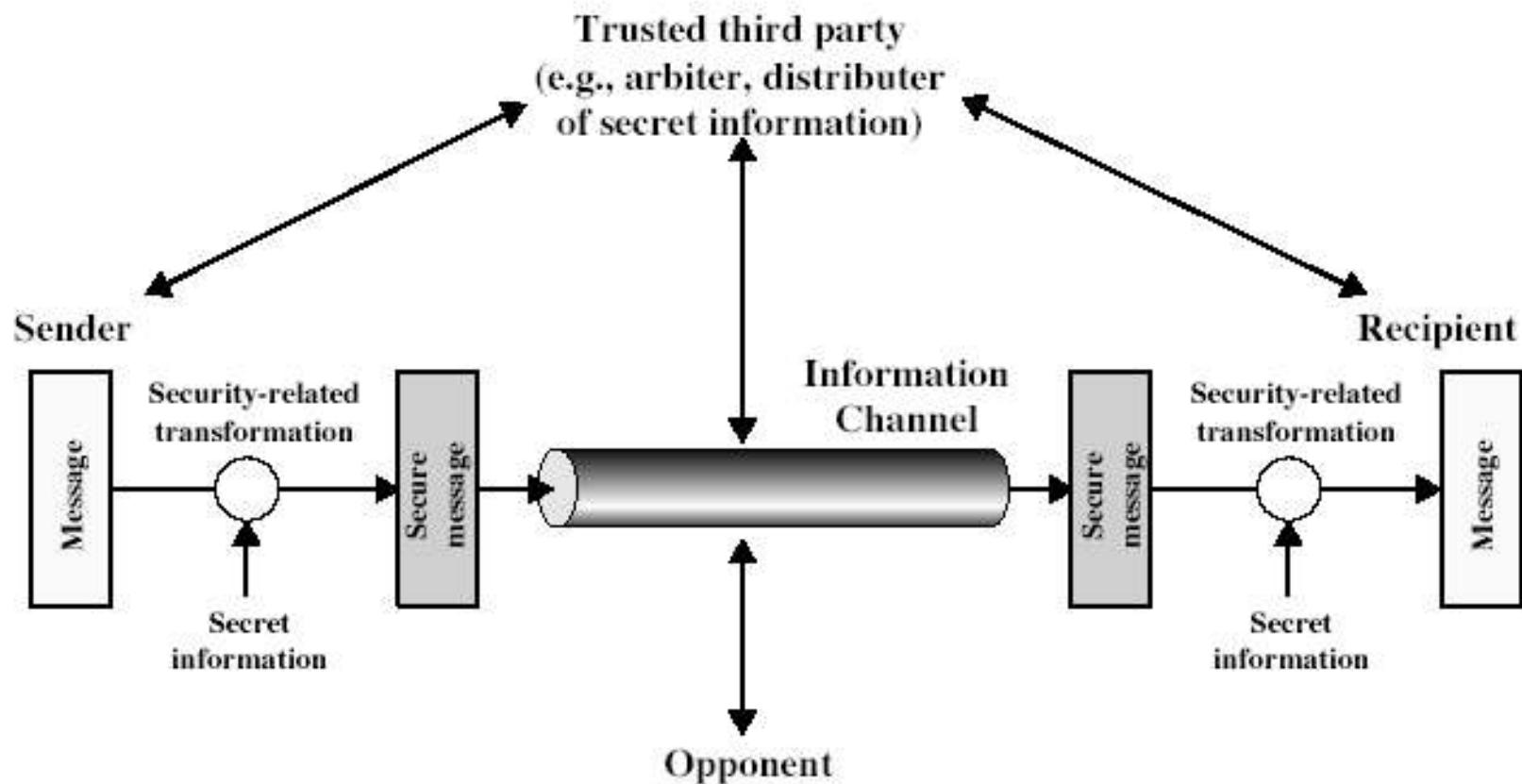


1.3.3 *Relation between Services and Mechanisms*

Table 1.2 *Relation between security services and mechanisms*

<i>Security Service</i>	<i>Security Mechanism</i>
Data confidentiality	Encipherment and routing control
Data integrity	Encipherment, digital signature, data integrity
Authentication	Encipherment, digital signature, authentication exchanges
Nonrepudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

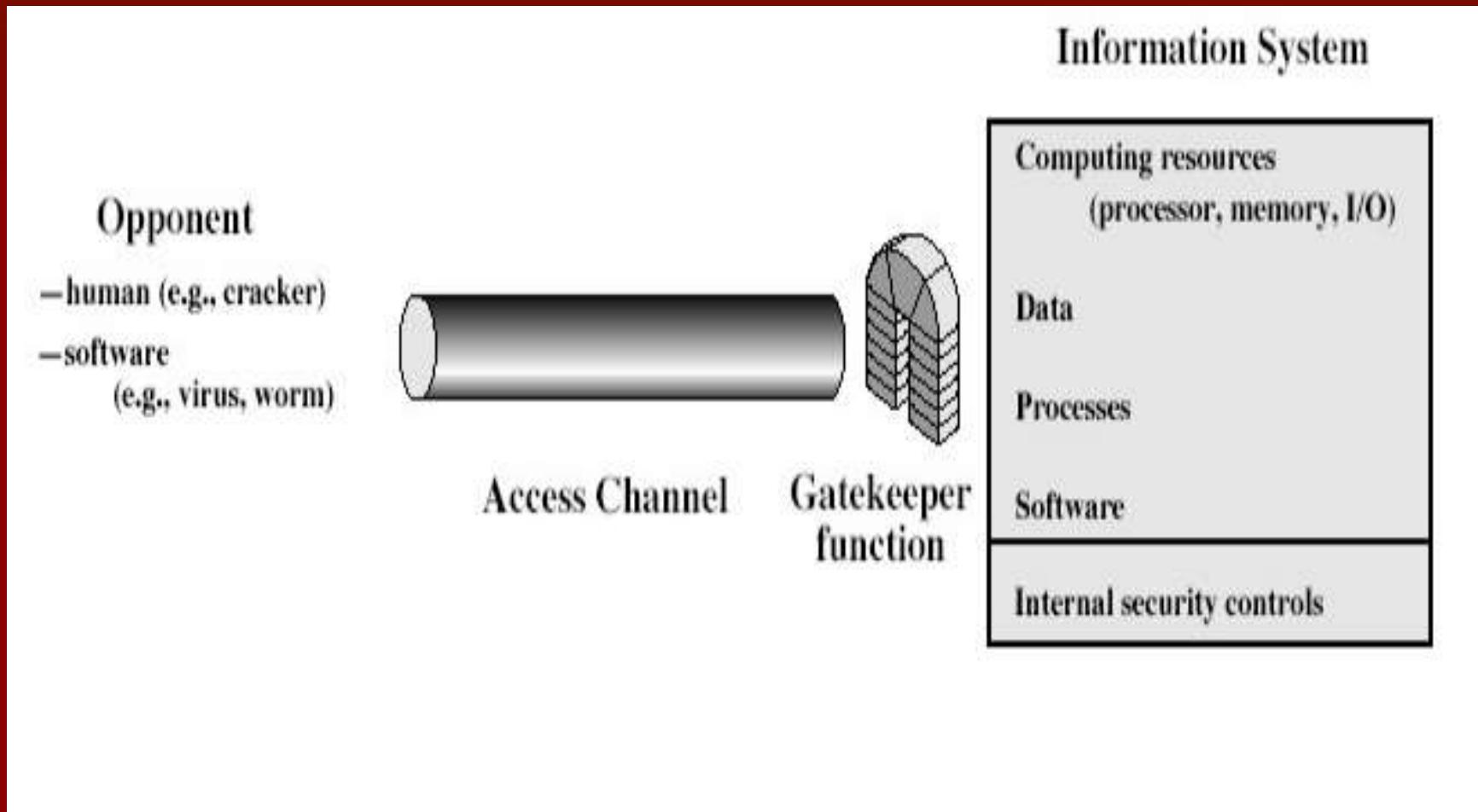
Model for Network Security



Model for Network Security

- using this model requires us to:
 - design a suitable algorithm for the security transformation
 - generate the secret information (keys) used by the algorithm
 - develop methods to distribute and share the secret information
 - specify a protocol enabling the principals to use the transformation and secret information for a security service

Model for Network Access Security



Model for Network Access Security

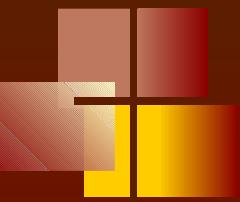
- using this model requires us to:
 - select appropriate gatekeeper functions to identify users
 - implement security controls to ensure only authorised users access designated information or resources
- trusted computer systems can be used to implement this model

1-4 TECHNIQUES

Mechanisms discussed in the previous sections are only theoretical recipes to implement security. The actual implementation of security goals needs some techniques. Two techniques are prevalent today: cryptography and steganography.

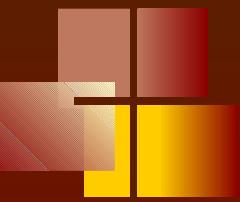
Topics discussed in this section:

1. Cryptography
2. Steganography



1.4.1 Cryptography

Cryptography, a word with Greek origins, means “secret writing.” However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks.



1.4.2 Steganography

The word steganography, with origin in Greek, means “covered writing,” in contrast with cryptography, which means “secret writing.”

Example: covering data with text

This book is mostly about cryptography, not steganography.

<input type="checkbox"/>						
0	1	0	0	0	0	1

1.4.2 *Continued*

Example: using dictionary

A	friend	called	a	doctor.
0	10010	0001	0	01001

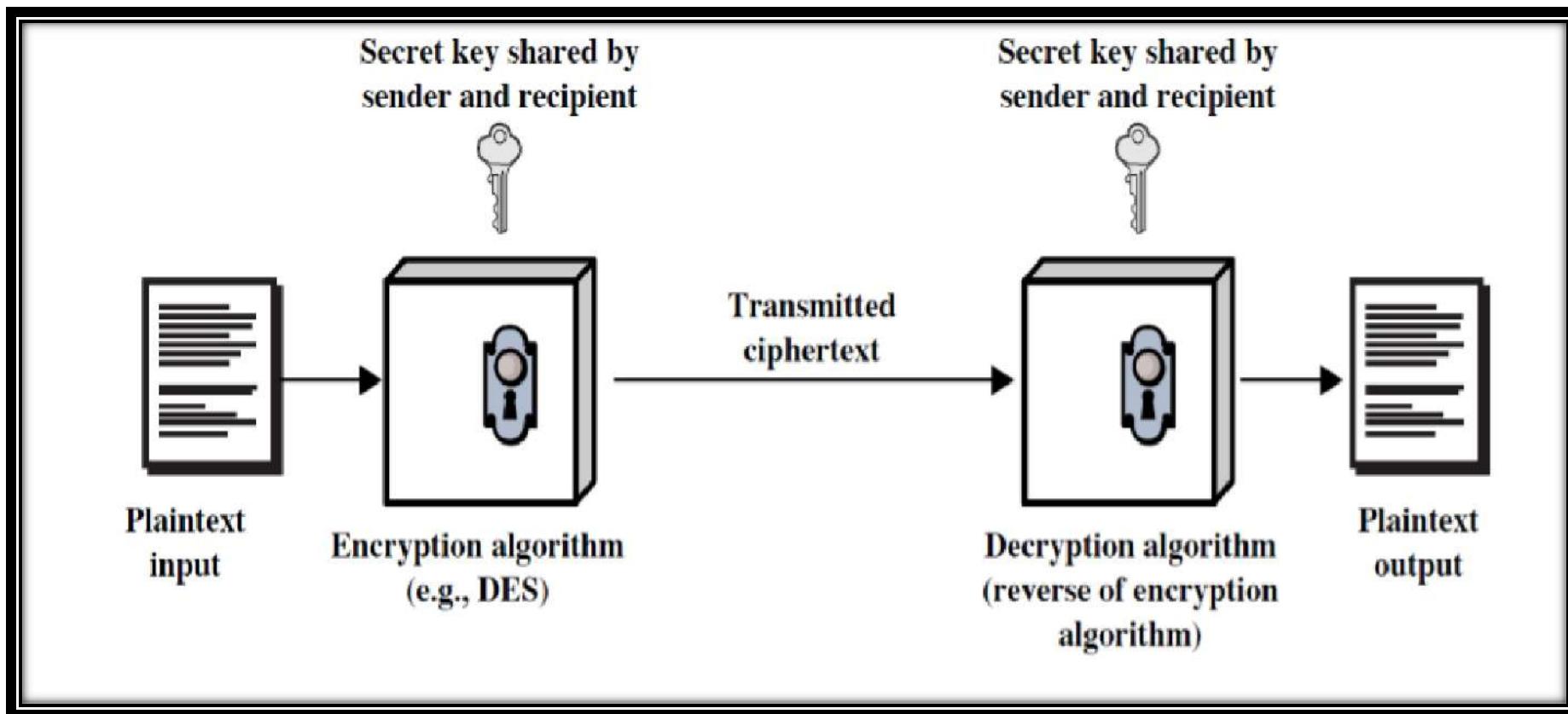
Example: covering data under color image

0101001 <u>1</u>	1011110 <u>0</u>	0101010 <u>1</u>
0101111 <u>0</u>	1011110 <u>0</u>	0110010 <u>1</u>
0111111 <u>0</u>	0100101 <u>0</u>	0001010 <u>1</u>

Topic:- Symmetric Cipher Model, Substitution techniques,
Transposition techniques, Steganography

Symmetric Cipher Model

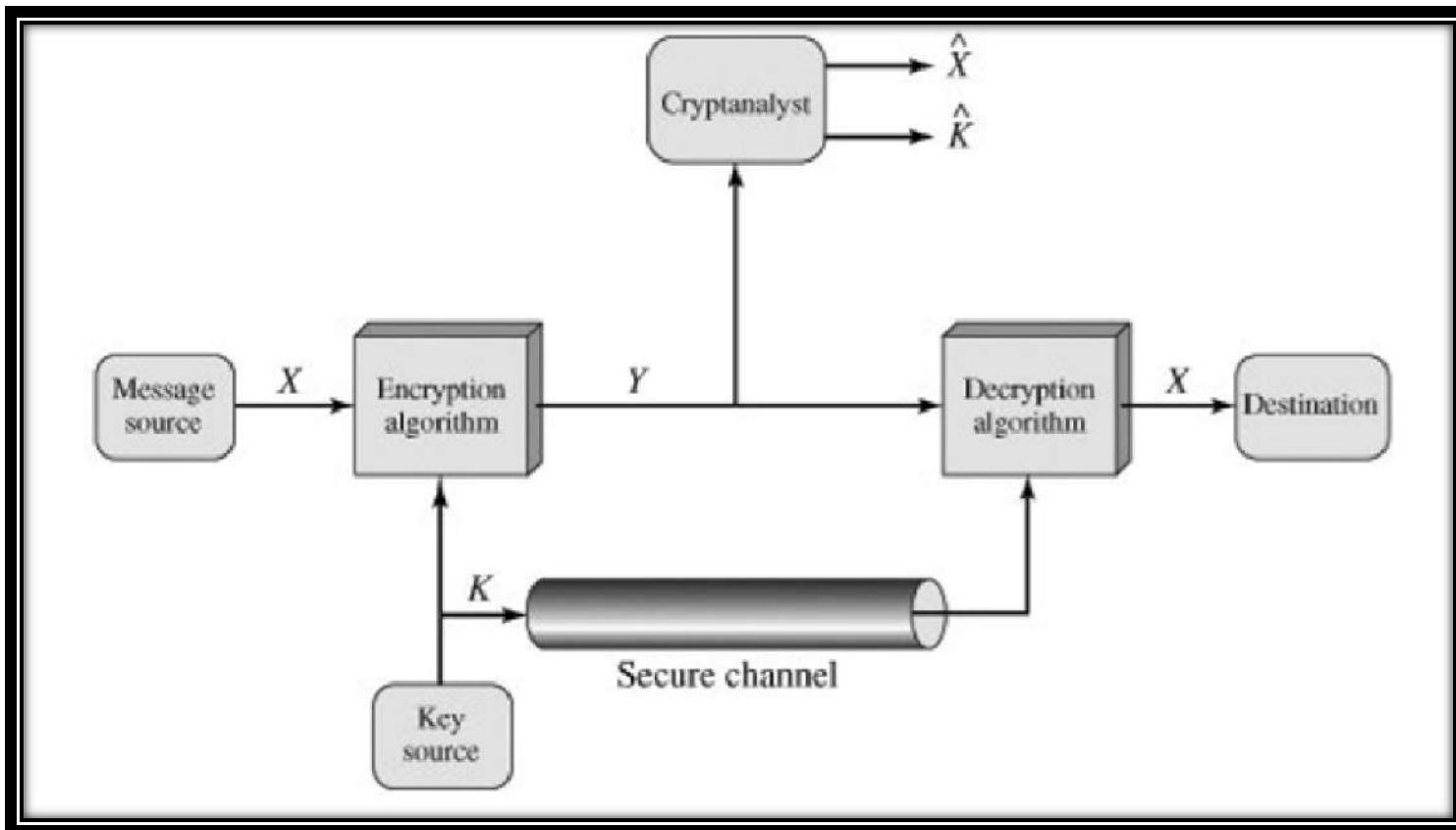
- ❑ A symmetric encryption scheme has five ingredients:
 - Plaintext: original message to be encrypted.
 - Ciphertext: the encrypted message.
 - Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.
 - Decryption algorithm: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.
 - Secret key: A secret key is the piece of information or parameter that is used to encrypt and decrypt messages in a symmetric, or secret-key, encryption.



Simplified Model of Symmetric encryption

There are two requirements for secure use of symmetric encryption:

- a strong encryption algorithm.
- a secret key known only to sender / receiver.



Model of Conventional Cryptosystem

Symmetric Encryption

Mathematically:

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

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

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

Cryptography

Cryptographic systems are characterized along three independent dimensions:

- The type of operations used for transforming plaintext to ciphertext.
- The number of keys used.
- The way in which the plaintext is processed.

Cryptanalysis

Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

Kirchhoff's principle: the adversary knows all details about a cryptosystem except the secret key.

Two general approaches:

- brute-force attack
- non-brute-force attack (cryptanalytic attack)

Brute-force attack:

The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
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	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s}$ = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s}$ = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s}$ = 6.4×10^{12} years	6.4×10^6 years

Cryptanalytic Attacks

Type of Attack	Known to Cryptanalyst
Ciphertext Only	<ul style="list-style-type: none">• Encryption algorithm• Ciphertext
Known Plaintext	<ul style="list-style-type: none">• Encryption algorithm• Ciphertext• One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	<ul style="list-style-type: none">• Encryption algorithm• Ciphertext• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	<ul style="list-style-type: none">• Encryption algorithm• Ciphertext• Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	<ul style="list-style-type: none">• Encryption algorithm• Ciphertext• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key• Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Unconditional Security

- An encryption scheme is unconditionally secure if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available.
- Therefore, all that the users of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria:
 - (1)The cost of breaking the cipher exceeds the value of the encrypted information.
 - (2)The time required to break the cipher exceeds the useful lifetime of the information.
- An encryption scheme is said to be computationally secure if either of the foregoing two criteria are met.

Basic Terminology

- **plaintext** - the original message
- **ciphertext** - the coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

Cryptosystem

A *cryptosystem* is a five-tuple (P, C, E, D) , where the following are satisfied:

1. p is a finite set of possible *plaintexts*.
2. C is a finite set of possible *ciphertexts*.
3. K , the *key space*, is a finite set of possible *keys*

E_K (encryption rule),

D_K (decryption rule).

Each $E_K: P \rightarrow C$ and $D_K: C \rightarrow P$ are functions

such that, $D_K(E_K(x)) = x$.

The two basic building blocks of all encryption techniques are :

1. Substitution techniques: A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

Different types of Substitution techniques are:

1. Caesar Cipher
2. Monoalphabetic Ciphers
3. Playfair Cipher
4. Hill Cipher
5. Polyalphabetic Ciphers
6. One-Time Pad

Caesar Cipher

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

➤ Example, key=3

plaintext: hello how are you

ciphertext: KHOOR KRZ DUH BRX

Caesar Cipher

- λ can define transformation as:

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
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

- λ mathematically give each letter a number

a b c d e f g h i j k l m
0 1 2 3 4 5 6 7 8 9 10 11 12
n o p q r s t u v w x y z
13 14 15 16 17 18 19 20 21 22 23 24 25

- λ then have Caesar cipher as:

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

$$p = D(C) = (C - k) \bmod (26)$$

Monoalphabetic Ciphers

- Better than Caeser Cipher
- For each character of alphabet, assign different-abrupt concerned character
- Example:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

- Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.
- A countermeasure is to provide multiple substitutes, known as **homophones**, for a single letter. If two letters considered for substitutes it is called **as digrams**.

• For Example: A-

Plaintext: goodmorning

Ciphertext: TLLWNLIMRMT

Playfair Cipher

- The best-known multiple-letter encryption cipher is the Playfair, which treats diagrams in the plaintext as single units and translates these units into ciphertext diagrams.
- The Playfair algorithm is based on the use of a $5 * 5$ *matrix* of letters constructed using a keyword.

➤ For Example,

 Keyword: security

 (monarchy)

 Plaintext: pattern

- In this case, the keyword is security. *The matrix is constructed by filling* in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order.
- Note: The letters I and J count as one letter.

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

Plaintext is encrypted two letters at a time, according to the following rules: **BALLOON=> BA LX LO ON**

- 1) Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that **pattern** would be treated as *pa tx te rn*.
- 2) Two plaintext letters that fall in the **same row** of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, **op** is encrypted as **PL**.
- 3) Two plaintext letters that fall in the **same column** are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, **mv** is encrypted as **VE**.

4) Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, **pa** becomes **OB**.

P I a i n t e x t : p a t x t e r n

Ciphertext: OB VA FT CP

- The Playfair cipher is a great advance over simple monoalphabetic ciphers.
- For one thing, whereas there are only 26 letters, there are $26 * 26 = 676$ diagrams, so that identification of individual diagrams is more difficult.

Hill Cipher

- This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters.
- The substitution is determined by m linear equations in which each character is assigned a numerical value.

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
0	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

In general terms , the hill system can be expressed as,

$$C = E(P, K) = PK \bmod 26$$

$$P = D(C, K) = CK^{-1} \bmod 26$$

Example

Plaintext: abcd

Keyword: $\begin{bmatrix} 3 & 1 \\ 5 & 2 \end{bmatrix}$

$$C = E(P, K) = PK \bmod 26$$

$$C = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 5 & 2 \end{bmatrix} \bmod 26$$

$$C = \begin{bmatrix} 5 & 2 \\ 21 & 8 \end{bmatrix} \bmod 26$$

$$C = \begin{bmatrix} 5 & 2 \\ 21 & 8 \end{bmatrix}$$

$$C = fcvi$$

$$k^{-1} = \frac{1}{ac - bd} \begin{bmatrix} 2 & -1 \\ 5 & 3 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 5 & 3 \end{bmatrix}$$

$$P = D(C, K) = CK^{-1} \bmod 26$$

$$P = \begin{bmatrix} 5 & 2 \\ 21 & 8 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 5 & 3 \end{bmatrix} \bmod 26$$

$$P = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \bmod 26$$

$$P = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

$$P = abcd$$

Polyalphabetic Ciphers

- Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message.
- The general name for this approach is polyalphabetic substitution cipher.
- All these techniques have the following features in common:
 1. A set of related monoalphabetic substitution rules is used.
 2. A key determines which particular rule is chosen for a given transformation.

For Example,

key: deceptivewearediscoveredsav

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

$$C_{i\wedge} \equiv (P_i + k_i \bmod m) \bmod 26$$

$$P_{i\wedge} \equiv (C_i - k_i \bmod m) \bmod 26$$

	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
a	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
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z			
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z				
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z					
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z						
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z							
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z								
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z									
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z										
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z											
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z												
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z													
o	O	P	Q	R	S	T	U	V	W	X	Y	Z														
p	P	Q	R	S	T	U	V	W	X	Y	Z															
q	Q	R	S	T	U	V	W	X	Y	Z																
r	R	S	T	U	V	W	X	Y	Z																	
s	S	T	U	V	W	X	Y	Z																		
t	T	U	V	W	X	Y	Z																			
u	U	V	W	X	Y	Z																				
v	V	W	X	Y	Z																					
w	W	X	Y	Z																						
x	X	Y	Z																							
y	Y	Z																								
z	Z																									

Key

One-Time Pad (Vernam cipher)

We now show two different decryptions using two different keys:

ciphertext:

key: pxlmvmsydoфuyrvzwc tnlebnecvgdupahfzzlmnyih

plaintext: mr mustard with the candlestick in the hall

ciphertext:

key: mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyouuhwt

plaintext: miss scarlet with the knife in the library

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
0	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

- In theory, we need look no further for a cipher. The one-time pad offers complete security but, in practice, has two fundamental difficulties:
 1. There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.
 2. Even more discouraging is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.
- Because of these difficulties, the one-time pad is of limited utility and is useful primarily for low-bandwidth channels requiring very high security.
- The one-time pad is the only cryptosystem that exhibits what is referred to as **perfect secrecy**.

Transposition Techniques

- A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.
- The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

For Example,

Plaintext: meet me after the toga party

rail fence of depth: 2

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

The encrypted message is: MEMATRHTGPRYETEFETEOAAT

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm.

Key: 4 3 1 2 5 6 7 (ATTACK POSTPONEDUNTILTWOAMXYZ)

Plaintext:

a	t	t	a	c	k	p
o	s	t	p	o	n	E
d	u	n	t	I	I	T
w	o	a	m	x	y	z

Ciphertext: TTNAAPMTSUOAODWCOIXKNLYPETZ

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm,

Key: 4 3 1 2 5 6 7

Plaintext: tt naap t
m t suoao
dw co i x k
n l ype t z

TTNAAPMTSUOAODWCOIXKNLYPETZ

Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

- To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is,

01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28

- After the first transposition we have,

03 10 17 24 04 11 18 25 02 09 16 23 01 08

15 22 05 12 19 26 06 13 20 27 07 14 21 28

- which has a somewhat regular structure. But after the second transposition, we have,

17 09 05 27 24 16 12 07 10 02 22 20 03 25

15 13 04 23 19 14 11 01 26 21 18 08 06 28

- This is a much less structured permutation and is much more difficult to cryptanalyze.

Steganography

- A plaintext message may be hidden in one of two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

Various other techniques have been used historically; some examples are the following :

- Character marking: Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.
- Invisible ink: A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.
- Pin punctures: Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

- Typewriter correction ribbon: Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.
- Steganography has a number of drawbacks when compared to encryption. It requires a lot of overhead to hide a relatively few bits of information, although using a scheme like that proposed in the preceding paragraph may make it more effective.
- Also, once the system is discovered, it becomes virtually worthless. This problem, too, can be overcome if the insertion method depends on some sort of key.
- Alternatively, a message can be first encrypted and then hidden using steganography.

- The advantage of steganography is that it can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered.
- Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide.

THANK YOU

Cryptography and Network Security

Third Edition
by William Stallings

Chapter 8 – Introduction to Number Theory

Prime Numbers

- prime numbers only have divisors of 1 and self
 - they cannot be written as a product of other numbers
 - note: 1 is prime, but is generally not of interest
 - eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
 - prime numbers are central to number theory
 - list of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
61 67 71 73 79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179 181 191
193 197 199

Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: $n=a \times b \times c$
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
 - eg. $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers & GCD

- two numbers a , b are **relatively prime** if have **no common divisors** apart from 1
 - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
 - eg. $300 = 2^1 \times 3^1 \times 5^2$ $18 = 2^1 \times 3^2$ hence $\text{GCD}(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$

Fermat's Theorem

- $a^{p-1} \bmod p = 1$
 - where p is prime and $\gcd(a, p) = 1$
- also known as Fermat's Little Theorem
- useful in public key and primality testing

Euler Totient Function $\phi(n)$

- when doing arithmetic modulo n
- **complete set of residues** is: $0 \dots n-1$
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
 - eg for $n=10$,
 - complete set of residues is $\{0,1,2,3,4,5,6,7,8,9\}$
 - reduced set of residues is $\{1,3,7,9\}$
- number of elements in reduced set of residues is called the **Euler Totient Function $\phi(n)$**

Euler Totient Function $\phi(n)$

- to compute $\phi(n)$ need to count number of elements to be excluded
- in general need prime factorization, but
 - for p (p prime) $\phi(p) = p-1$
 - for $p \cdot q$ (p, q prime) $\phi(p \cdot q) = (p-1)(q-1)$
- eg.
 - $\phi(37) = 36$
 - $\phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\phi(n)} \text{mod } N = 1$
 - where $\gcd(a, N) = 1$
- eg.
 - $a=3; n=10; \phi(10)=4;$
 - hence $3^4 = 81 = 1 \text{ mod } 10$
 - $a=2; n=11; \phi(11)=10;$
 - hence $2^{10} = 1024 = 1 \text{ mod } 11$

Primality Testing

- often need to find large prime numbers
- traditionally **sieve** using **trial division**
 - ie. divide by all numbers (primes) in turn less than the square root of the number
 - only works for small numbers
- alternatively can use statistical primality tests based on properties of primes
 - for which all prime numbers satisfy property
 - but some composite numbers, called pseudo-primes, also satisfy the property

Miller Rabin Algorithm

- a test based on Fermat's Theorem
- algorithm is:

TEST (n) is:

1. Find integers k, q , $k > 0$, q odd, so that $(n-1) = 2^k q$
2. Select a random integer a , $1 < a < n-1$
3. **if** $a^q \text{ mod } n = 1$ **then** return ("maybe prime");
4. **for** $j = 0$ **to** $k - 1$ **do**
 5. **if** $(a^{2^j q} \text{ mod } n = n-1)$
then return(" maybe prime ")
6. return ("composite")

Probabilistic Considerations

- if Miller-Rabin returns “composite” the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is $< \frac{1}{4}$
- hence if repeat test with different random a then chance n is prime after t tests is:
 - $\Pr(n \text{ prime after } t \text{ tests}) = 1 - 4^{-t}$
 - eg. for $t=10$ this probability is > 0.99999

Prime Distribution

- prime number theorem states that primes occur roughly every $(\ln n)$ integers
- since can immediately ignore evens and multiples of 5, in practice only need test $0.4 \ln(n)$ numbers of size n before locate a prime
 - note this is only the “average” sometimes primes are close together, at other times are quite far apart

Chinese Remainder Theorem

- used to speed up modulo computations
- working modulo a product of numbers
 - eg. $\text{mod } M = m_1m_2..m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- can implement CRT in several ways
- to compute $(A \bmod M)$ can firstly compute all $(a_i \bmod m_i)$ separately and then combine results to get answer using:

$$A \equiv \left(\sum_{i=1}^k a_i c_i \right) \bmod M$$

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{for } 1 \leq i \leq k$$

Primitive Roots

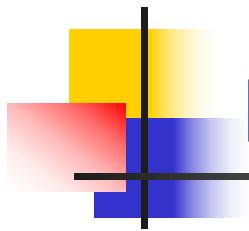
- from Euler's theorem have $a^{\phi(n)} \pmod{n} = 1$
- consider $a^m \pmod{n} = 1$, $\text{GCD}(a, n) = 1$
 - must exist for $m = \phi(n)$ but may be smaller
 - once powers reach m , cycle will repeat
- if smallest is $m = \phi(n)$ then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group \pmod{p}
- these are useful but relatively hard to find

Discrete Logarithms or Indices

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- that is to find x where $a^x = b \text{ mod } p$
- written as $x = \log_a b \text{ mod } p$ or $x = \text{ind}_{a,p}(b)$
- if a is a primitive root then always exists,
otherwise may not
 - $x = \log_3 4 \text{ mod } 13$ (x st $3^x = 4 \text{ mod } 13$) has no answer
 - $x = \log_2 3 \text{ mod } 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Summary

- have considered:
 - prime numbers
 - Fermat's and Euler's Theorems
 - Primality Testing
 - Chinese Remainder Theorem
 - Discrete Logarithms



Module: 2

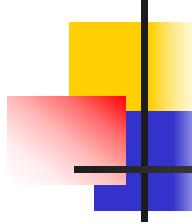
“Block Cipher and Stream Ciphers”

--Dr.S.P.Anandaraj

'Associate Professor'

Department-CSE'

Presidency University, Bangalore

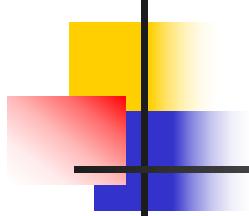


Plaintext to Ciphertext

Objectives:

- one 'character' in ciphertext
 - = function of a large number of 'characters' in the plaintext.
- Thus if e is the most commonly used character in English plaintext, it may not be so in the ciphertext.

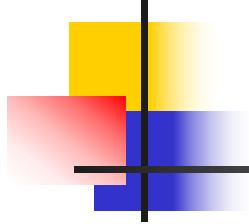
In ciphertext all the characters should have ideally an equal frequency of occurrence.



Frequency of Patterns

For every language: frequency of **characters**, **digrams** (two letter sequences) and **trigrams** are known. → statistical analysis to decipher encrypted information.

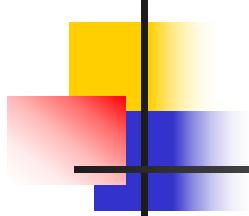
- English: **e**: the character with highest frequency
- C: **#define** and **#include** in the beginning
- **Protocols** and **tcpdump**: repetitive, fixed sized fields



Block Ciphers

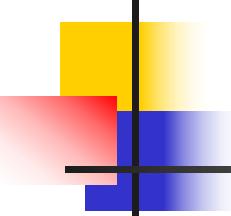
used for

- Fast encryption of large amount of data
- Creating a cryptographic checksum for guaranteeing the integrity of data
- Secrecy and authentication service



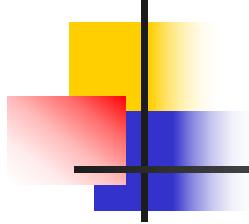
Modern Block Ciphers

- will now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy and/or authentication services
- in particular will introduce DES (Data Encryption Standard)



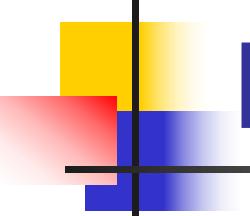
Types of Cipher Algorithms

- Streaming Cipher: encrypts data unit by unit, where a unit is of certain number of bits
(Example: If the unit be a bit, a stream cipher encrypts data unit by unit. Or if the unit be a byte, it encrypts byte by byte)
- Block cipher: encrypts a fixed- sized block of data at a time:
 - For a 64 bit block of plaintext, for encryption to a 64-bit ciphertext, may need a table of $2^{64} = 150$ million terabytes = 15×10^{19} bytes
 - For a block size of 128 bits, the table would require a memory of 5×10^{39} bytes.



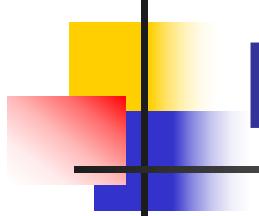
Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
 - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers



Block Cipher Principles

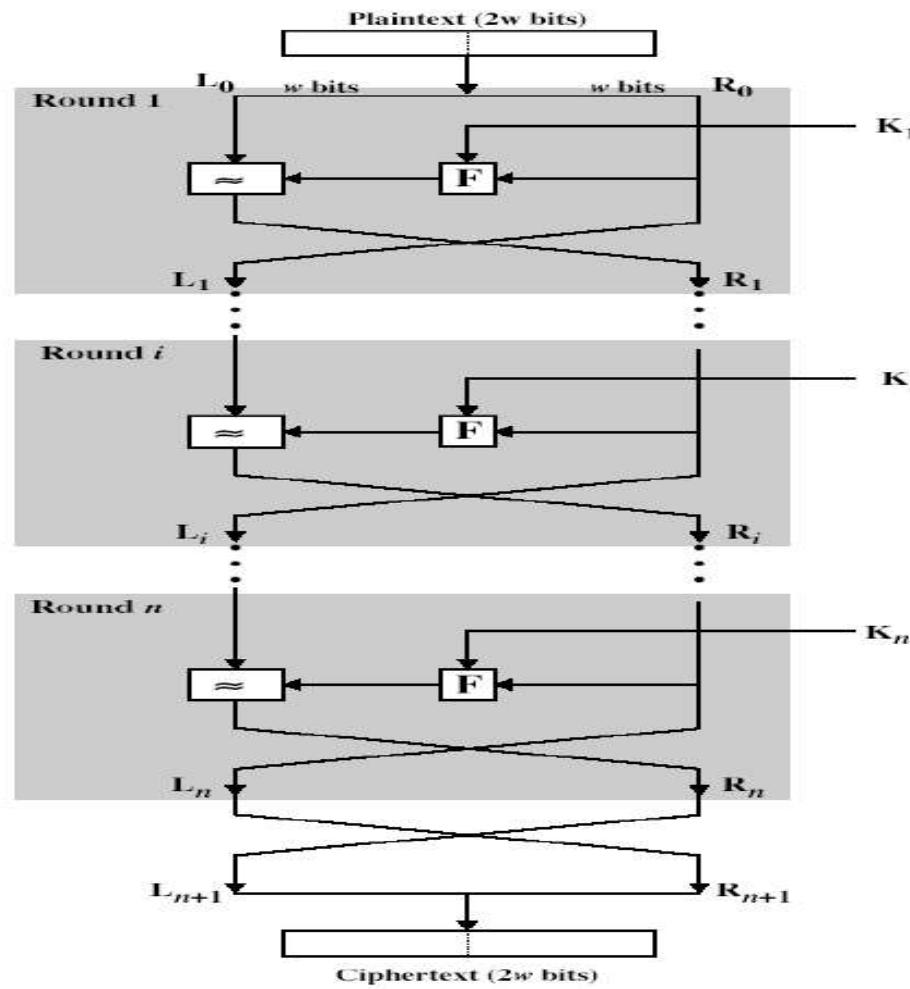
- most symmetric block ciphers are based on a **Feistel Cipher Structure**
- needed since must be able to **decrypt** ciphertext to recover messages efficiently
- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
- instead create from smaller building blocks
- using idea of a product 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
- implements Shannon's substitution-permutation network concept

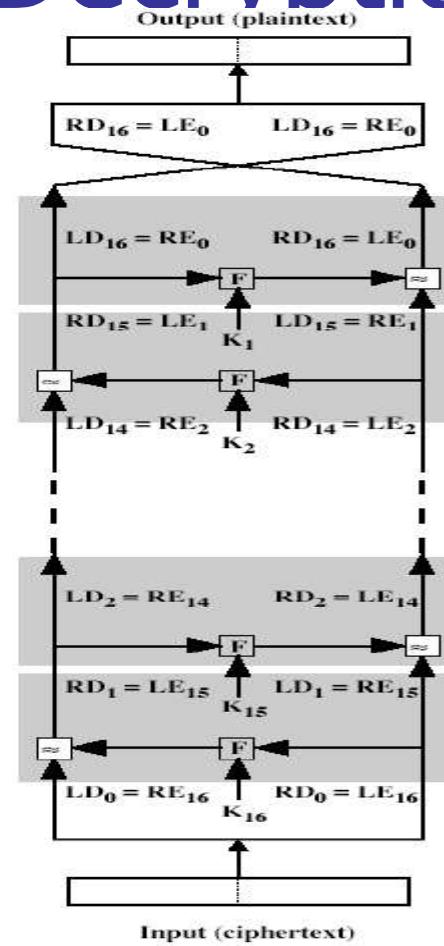
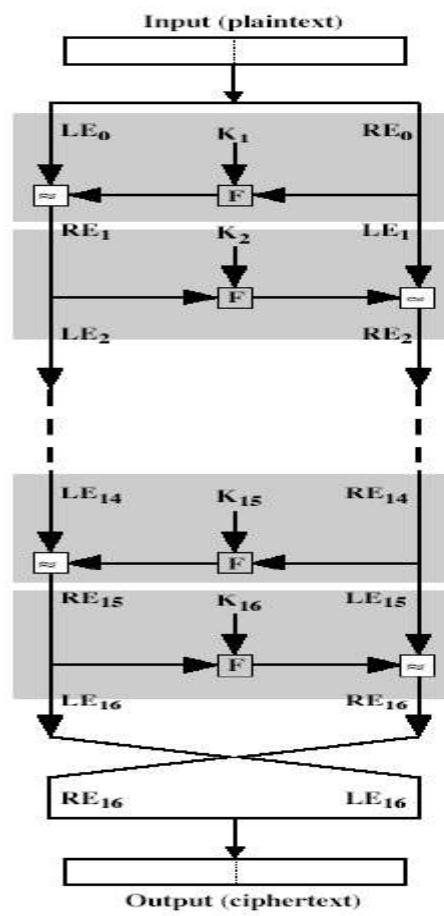
Feistel Cipher Structure

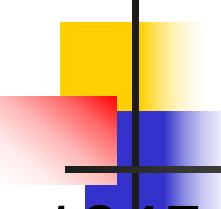


Feistel Cipher Design Principles

- **block size**
 - increasing size improves security, but slows cipher
- **key size**
 - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **number of rounds**
 - increasing number improves security, but slows cipher
- **subkey generation**
 - greater complexity can make analysis harder, but slows cipher
- **round function**
 - greater complexity can make analysis harder, but slows cipher
- **fast software en/decryption & ease of analysis**
 - are more recent concerns for practical use and testing

Feistel Cipher Decryption



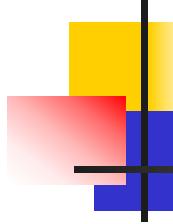


Diffusion & Confusion :

1945: "Introduce diffusion and confusion through cryptographic algorithms", said CLAUDE SHANNON.

DIFFUSION:

- Use **permutation** followed by some **functional transformation**.
- seeks to make statistical **relationship between the plaintext and ciphertext** as complex as possible.
- Diffuses the structure of the plaintext over a large part of the ciphertext.

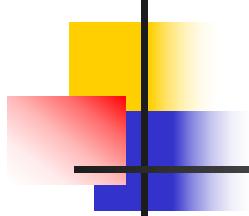


CONFUSION

CONFUSION:

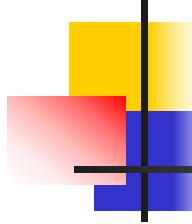
- makes the **relationship between the statistics of the ciphertext and the encryption key** as complex as possible.
- Achieved by using a complex **substitution algorithm**.

IMPORTANT: Substitution or Permutation: easy to break by using statistical analysis; strength due to non-linear functional transformation.¹⁴



Diffusion and Confusion

- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this
- more practically Shannon suggested combining elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible



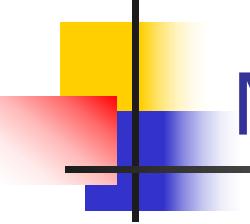
Kerckhoff's Rule

The strength of an encryption algorithm depends upon:

1. Design of the algorithm
2. Key length
3. Secrecy of the key (requires proper management of key distribution)

1883: Jean Guillaumen Hubert Victor Francois Alexandre Auguste Kerckhoff von Nieuwenhof: " Cryptosystems should rely on the secrecy of the key, but not of algorithm."

Advantages of Openness: 1994: A hacker published the source code of RC4, a secret encryption algorithm, designed by RSA Data security Inc. → attacks, that exposed several weaknesses of RC4

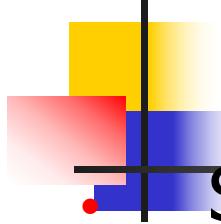


Modern Encryption Techniques:

- DES: A rather complex encryption scheme.
- Simplified DES:
 - A teaching tool
 - Designed by Prof. Edward Schaeter, Santa Clara University, 1996
 - http://www.cs.binghamton.edu/~steflik/cs455/Simplified_DES.ppt, as of 29th Sept 2009

Given: plaintext 8-bit, Key 10-bit

Output: ciphertext 8-bit



Simplified DES:

ciphertext = IP⁻¹ (f_{k2} (SW (f_{k1} (IP (plaintext)))))

- SDES 's five steps:

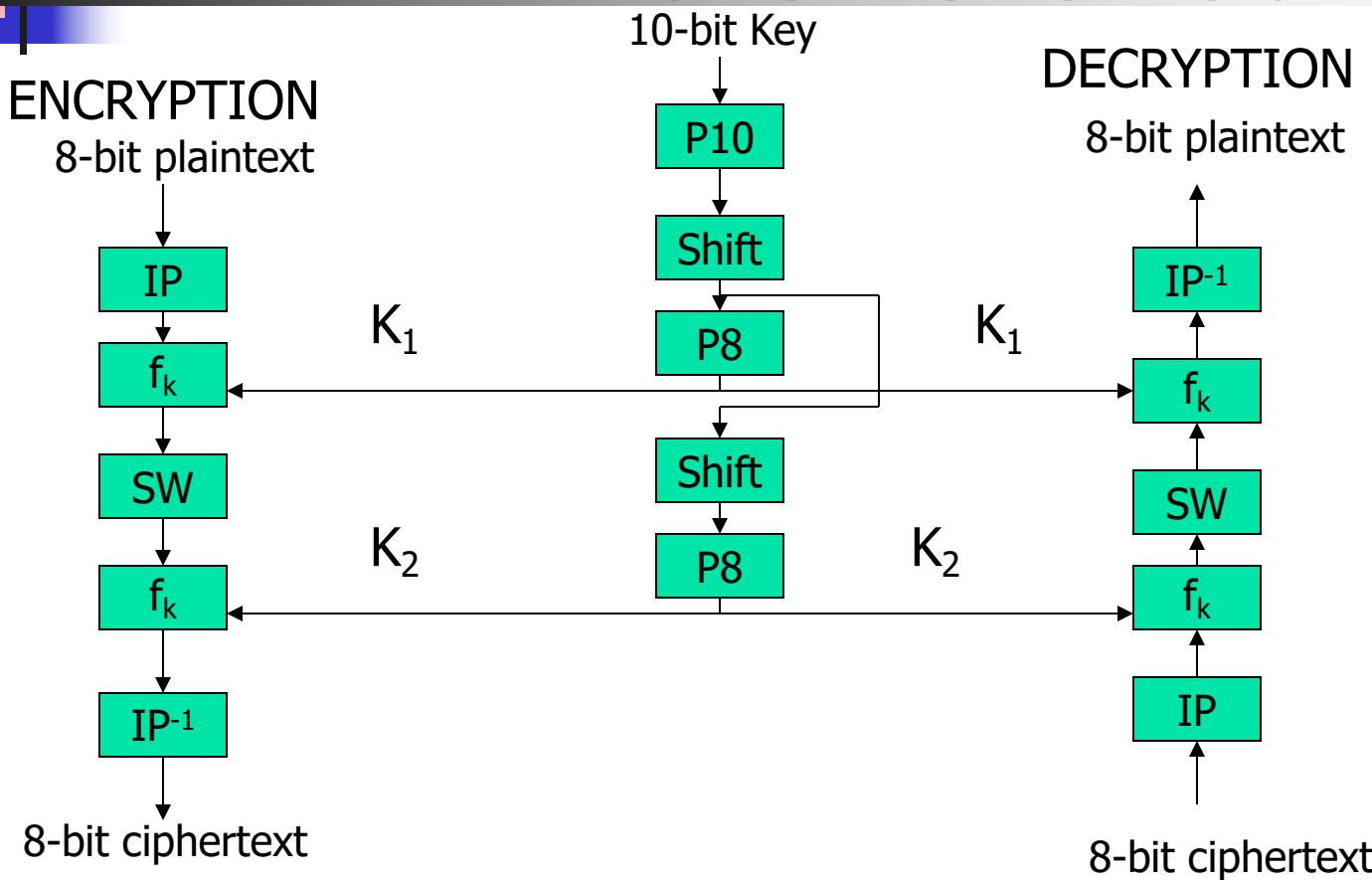
1. Initial Permutation IP.
2. A complex function f_k which requires key K₁.
3. A switch function SW which switches the left half and the right half of a data string.
4. The function f_k again with a different key K₂.
5. A permutation function that is the inverse of IP –called IP⁻¹.

$$(\text{IP}^{-1} (\text{IP} (X))) = X.$$

SDES may be said to have two ROUNDS of the function f_k.

Simplified DES scheme:

ciphertext = IP⁻¹ (f_{k2} (SW (f_{k1} (IP (plaintext)))))
Plaintext = IP⁻¹ (f_{k1} (SW (f_{k2} (IP (ciphertext)))))



SDES (continued)

$$K_1 = P8 (\text{Shift} (P10 (\text{Key})))$$

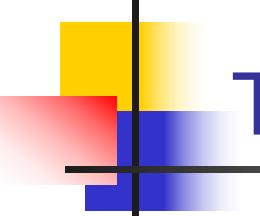
Plaintext = IP⁻¹ (f_{k1} (SW (f_{k2} (IP (ciphertext))))).

- To obtain K₁ and K₂:
- Given: K = (k₁ k₂ k₃ k₄ k₅ k₆ k₇ k₈ k₉ k₁₀)
- Step1: Permutation P10

P10 :

3	5	2	7	4	10	1	9	8	6
---	---	---	---	---	----	---	---	---	---

- Step2: Left shift (circular) by one bit
 - for the left half and
 - for the right half separately.



To obtain K₁ and K₂

- Step3: Permutation for producing an 8 bit key K₁ from a 10 bit input.

P8:

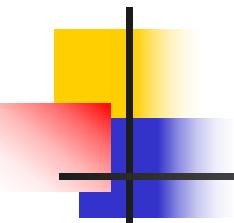
6	3	7	4	8	5	10	9
---	---	---	---	---	---	----	---

- Step4: Take the result of step2. On it use Left Shift (circular) by 2 bits
 - for the left half and
 - for the right half separately.
- Step5: Another instance of P8 is used to produce the second 8 bit key K₂.

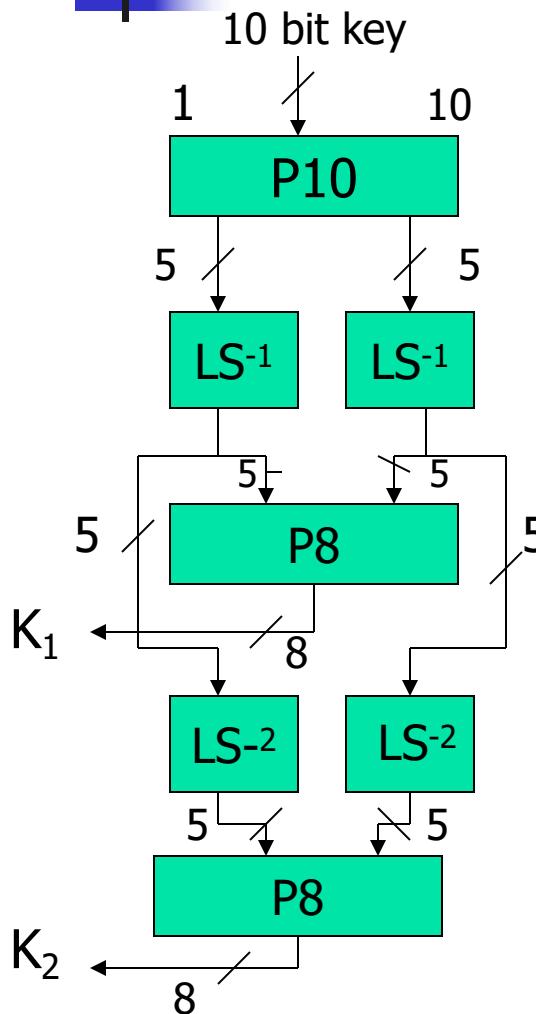
$$K_1 = P8 (\text{Shift} (P10 (\text{Key})))$$

$$K_2 = P8 (\text{Shift} (\text{Shift} (P10 (\text{Key})))).$$

Key generation for simplified DES:



$$K_1 = P8 (\text{Shift} (P10 (\text{Key})))$$
$$K_2 = P8 (\text{Shift} (\text{Shift} (P10 (\text{Key}))))$$



Circular left shift by 1, separately on the left and the right halves



Circular left shift by 2 , separately on the left and the right halves



Example: Key Generation

3 5 2 7 4 10 1 9 8 6

P10

6 3 7 4 8 5 10 9

P8

10-bit key = 1 0 1 0 0 0 0 0 1 0

1 0 0 0 0 0 1 1 0 0 P10

0 0 0 0 1 1 1 0 0 0 LS⁻¹ LS⁻¹

1 0 1 0 0 1 0 0 = K₁ P8

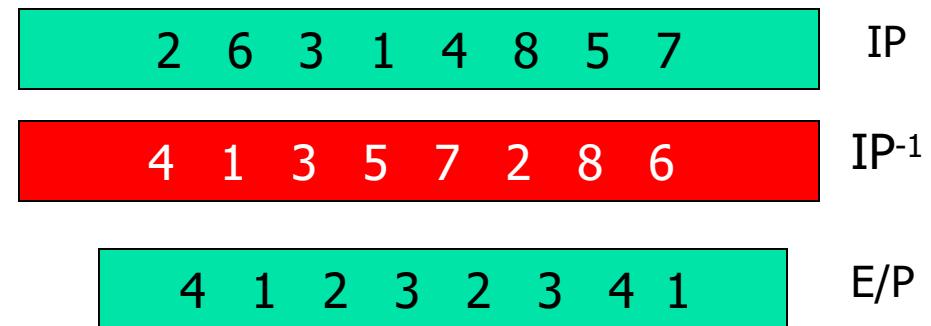
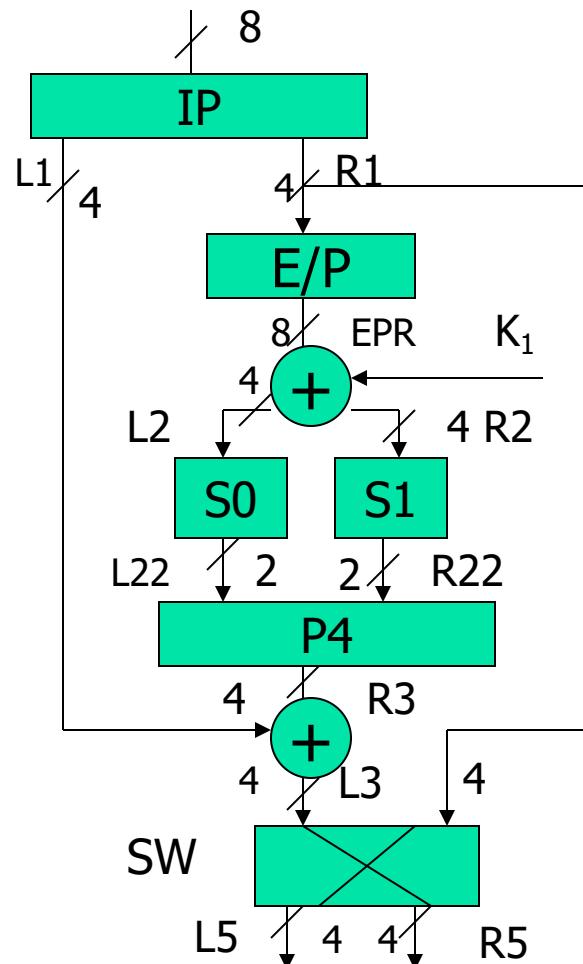
0 0 1 0 0 0 0 0 1 1 LS⁻² LS⁻²

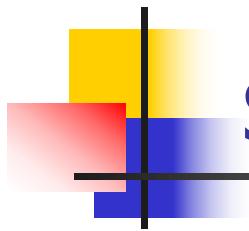
0 1 0 0 0 0 1 1 = K₂ P8

Simplified DES Encryption:

$$\text{ciphertext} = \text{IP}^{-1} (f_{k2} (\text{SW} (f_{k1} (\text{IP} (\text{plaintext}))))))$$

8-bit plaintext





S0 and S1 boxes:

$$S0 = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \left[\begin{matrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{matrix} \right] \end{matrix}$$

$$S1 = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} & \left[\begin{matrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{matrix} \right] \end{matrix}$$

2 4 3 1

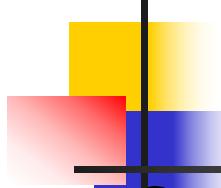
P4

The function f_k :

Permutation IP is applied to the 8-bit plaintext to generate L1 and R1, the left and the right halves.

INPUT to f_k :

- 4-bit left half (L1) and
- 4-bit Right half (R1) of a data string.
- Step1: E/P: Expansion/Permutation on R1 to produce an 8 bit data string called EPR.
- Step2: XOR of EPR with key K_1 for f_{k1} to produce the left half (L2) and right half (R2).
- Step3 (a): $L2 \xrightarrow{4} \text{S0 box} \xrightarrow{2} L22$



The function f_k (continued):

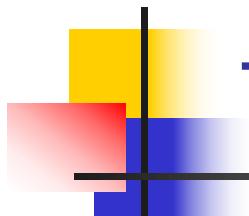
- Step3(b): $R2 \xrightarrow{4} S1 \text{ box} \xrightarrow{2} R22$

Given the 4 bits of $L2$ (or $R2$) part. Pick up the ij th element of $S0$ (or of $S1$), where

$i = 1^{\text{st}}$ and 4^{th} bits; $j = 2^{\text{nd}}$ and 3^{rd} bits.

Then convert this element to a 2-bit binary number.

- AUTOKEYING (also called autoclaving): In step 3, the selection of the element in the S -box depends on both data & key. This feature is called autokeying.



The function f_k

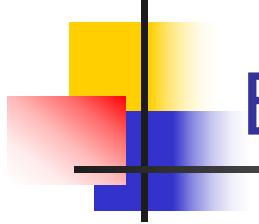
(continued)

Step4: $(L_{22} : R_{22})$ goes through a permutation P_4 to produce a 4-bit R_3 .

P_4

2	4	3	1
---	---	---	---

- Step5: $L_3 = L_1 \oplus R_3$
- Step6: $L_3 : R_1$ is then the input to SW .
- The second instance of f_k is similar to the first, except that the key K_2 is used.



Example: SDES Encryption

- Example:

Plaintext = 1 0 1 1 1 1 0 1

0 1 1 1 1 1 1 0

L1 = 0 1 1 1

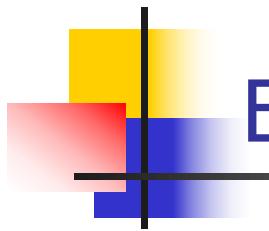
R1 = 1 1 1 0

EPR = 0 1 1 1 1 1 0 1

EPR \oplus K1 = 1 1 0 1 1 0 0 1

Row : first and fourth bit

Column : 2nd and 3rd bit



Example: SDES Encryption (continued)

For S0:

$$L_2 = 1101$$

Therefore Row = 3 Column = 2

$$L_{22} = 3 \Rightarrow 11$$

For S1:

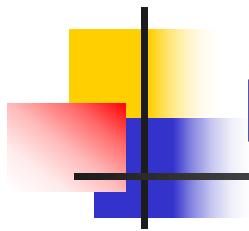
$$R_2 = 1001$$

Row = 3 Column = 0

$$R_{22} = 2 \Rightarrow 10$$

$L_{22} : R_{22} \quad 1110$

$$R_3 = 1011$$



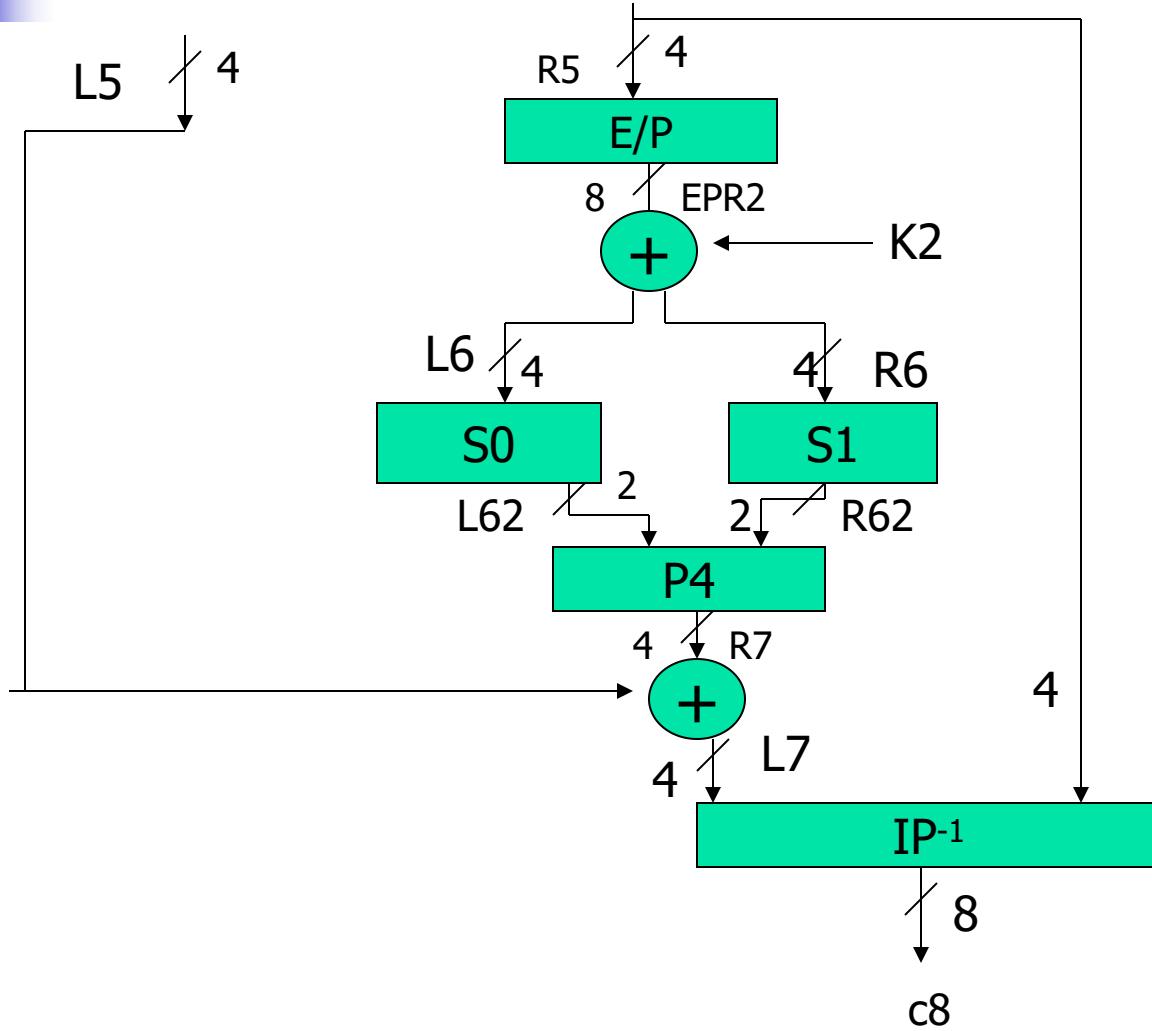
Example: SDES Encryption (continued)

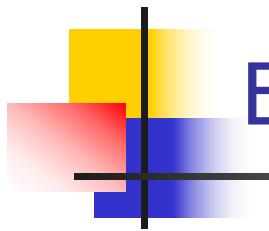
$$L_3 = R_3 \oplus L_1$$

$$= 1100$$

$$L_5 = 1110$$

$$R_5 = 1100$$





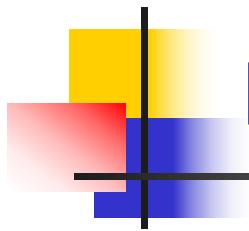
Example: SDES Encryption: f_{k2}

- EPR2 = 0 1 1 0 1 0 0 1
 $EPR2 \oplus K2 = 0 0 1 0 1 0 1 0$
 $L6 = 0 0 1 0$
 $R6 = 1 0 1 0$

For S0:

Row = 0 Column = 1

$L62 = 0 \Rightarrow 0 0$



Example: SDES Encryption: f_{k_2} (continued)

For S1:

Row = 2 Column = 1

$$R62 = 0 \Rightarrow 00$$

$$R7 = 0000$$

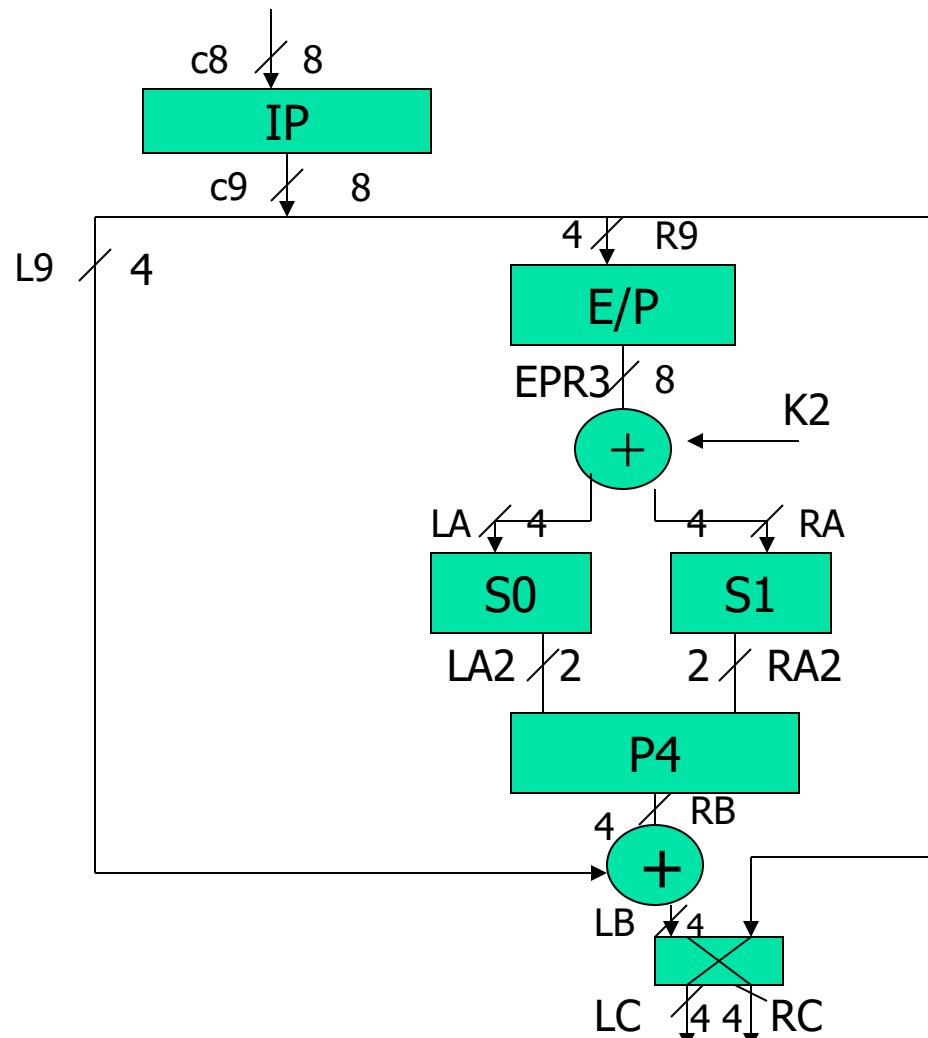
$$L7 = R7 \oplus L5$$

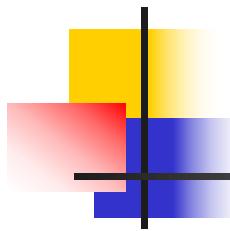
$$= 1110$$

$$L7 : L5 = 1110 \quad 1100$$

$$C8 = 01110101$$

SDES Decryption:





Decryption: Example

C9 = 1 1 1 0 1 1 0 0

EPR3 = 0 1 1 0 1 0 0 1

LA : RA = EPR3 \oplus K2

= 0 0 1 0 1 0 1 0

S0: Row = 0 Column = 1

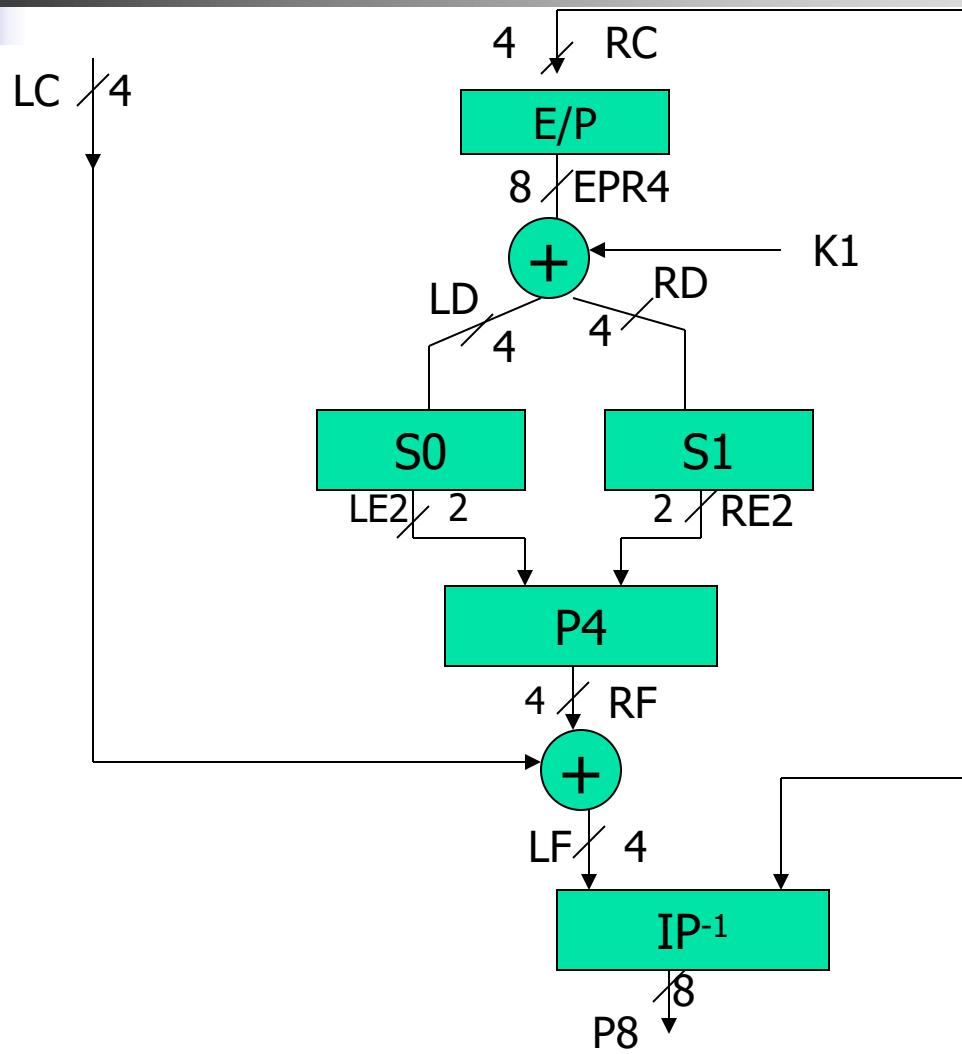
LA2 = 0 => 0 0

S1: Row = 2 Column = 1

RA2 = 0 => 0 0

R7 = 0000 LB = 1110

LC = 1100 RC = 1110



Decryption: Example (cont'd)

■ EPR4 = 0 1 1 1 1 1 0 1

$$\text{LD : RD} = \text{EPR4} \oplus \text{K1}$$

$$= \begin{matrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}$$

S0: Row = 3 Column = 2

LE2 = 3 => 1 1

S1: Row = 3 Column = 0

$$RE_2 = 2 \Rightarrow 10$$

$$\text{LE2 : RE2} = 1 \ 1 \ 1 \ 0$$

RF = 1 0 1 1

$$k_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$E|PR_1 = 0001 \quad 0100$$

17

ALL GOOD

101 - 000

15 05

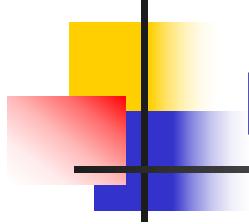
16

$$\text{لے } \frac{4}{\boxed{50}} \times 2 \rightarrow 100 \Rightarrow 101 \boxed{50}$$

$$R_5 \xrightarrow{4} \boxed{s} \xrightarrow{2e_{53}} \begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}$$

W₅₅ R₅₅ = 0.100

b - 1



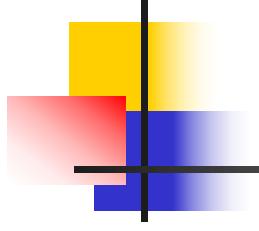
Decryption: Example (continued)

$$LF = LC \oplus RF$$

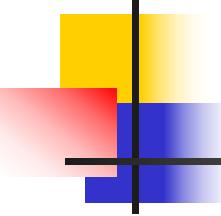
$$= 0\ 1\ 1\ 1$$

$$LF : RC = 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0$$

$$P8 = 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1$$



“As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality.”



DES Encryption:

DES: a public standard. But its design criterion has not been published.

64 bit plaintext goes through

- an Initial Permutation (IP).
 - 16 Rounds of a complex function f_k as follows:
 - Round 1 of a complex function f_k with sub key K_1 .
 - Round 2 of a complex function f_k with sub key K_2 .
 - |
• Round 16 of a complex function f_k with sub key K_{16}
 - At the end of 16 rounds, the Left-half and Right-half are swapped..
 - an Inverse Initial Permutation (IP^{-1})
- to produce 64 bit ciphertext.**

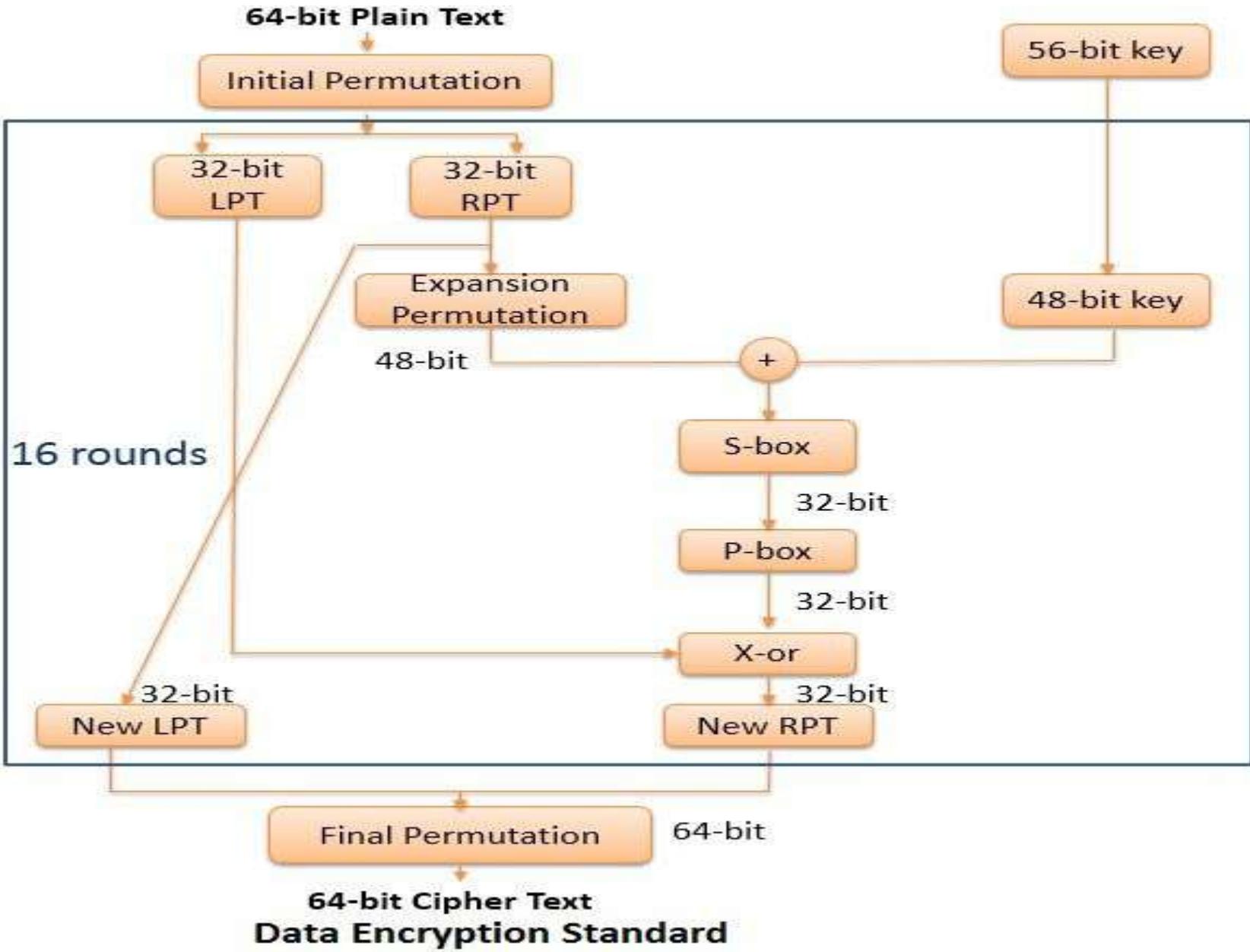
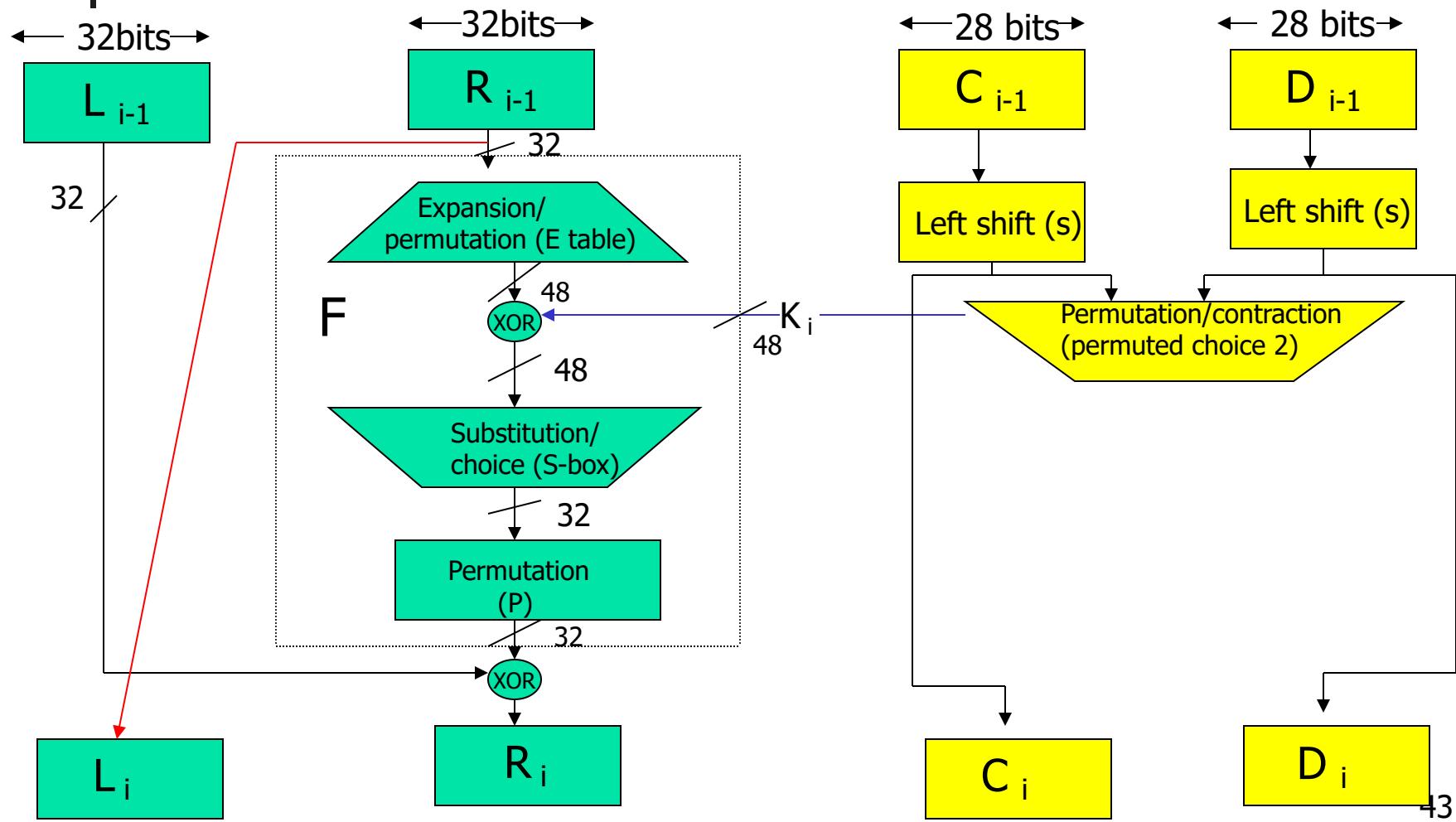
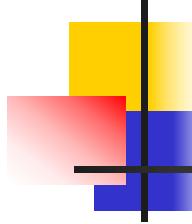


Fig : single Round of DES Algorithm:



- Each round contains following functions:
- **Expansion Permutation**: Here the 32-bit right portion is expanded to form 48-bit right portion.
- **Xor**: The 48-bit right portion is Xor with 48-bit subkey obtained from the 56-bit key, which results in the 48-bit output.
- **S-box**: The 48-bit output obtained by Xor step is reduced to 32 bit again.
- **P-box**: Here the 32-bit result obtained from S-box is again permuted, which result in 32-bit permuted output.



DES Round

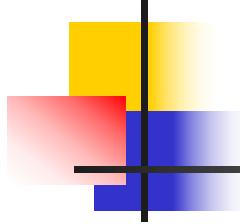
- x : block of plaintext
- let $x_0 = \text{IP}(x) = L_0:R_0$
- 16 rounds with f : cipher function
 K_i : sub-key for the i th round

While $i \leq 16$,

$$x_i = L_i:R_i$$

$$L_i = R_{i-1}$$

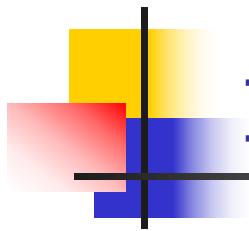
$$R_i = L_i \oplus F(R_{i-1}, K_i)$$



DES Encryption

Recapitulation:

- IP
- 16 rounds with 16 sub-keys
- Swapping
- Inverse Initial Permutation

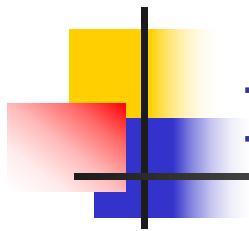


Initial Permutation (IP):

- IP and IP^{-1} are defined by 8×8 tables T1 and T2.

Table T1: IP

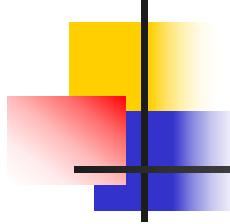
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	
1							
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



Inverse Initial Permutation(IP⁻¹)

Table T2: IP⁻¹

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



IP and IP⁻¹

Input Permutation permutes the position of bits as follows:

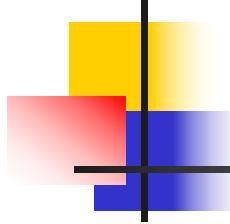
Input	1	2	3	4	61	62	63	64
Position	40	8	48	16	49	17	57	25
Output	40	8	48	16	49	17	57	25

IP(plaintext) = L(0): R(0),

where L(0) and R(0) are the left –half and the right-half of the output after the permutation done through IP.

Inverse Input Permutation reverses it back.

Thus $IP^{-1}(IP(X)) = X$



IP

Regular in structure, so that hardware is simple

- Even bits allocated to L(0)
- Odd bits allocated to R(0)

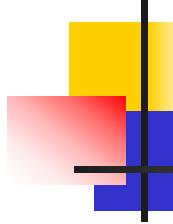
IP does not add to security. It makes the function a little complex.

- Example:

$$IP(675a6967\ 5e5a6b5a) = (ffb2194d\ 004df6fb)$$

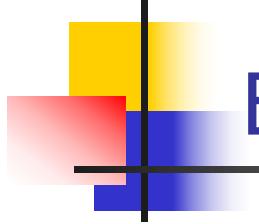
$$IP^{-1}(068dddcd\ 1d4cceb) = 974affbf,\ 86022d1f$$

Reference: Lawrie Brown, "Cryptography: lecture 8". available at <http://www.cs.adfa.edu.au/archive/teaching/studinfo/ccs3/lectures/less08.html>



Function

- E/P: to get 48 bits from 32 bits of R_i : each input block of 4 bits *contributes* 2 bits to each output block
→ **Avalanche Effect**: A small difference in plaintext *causes* quite different ciphertext
- $E(R_{i-1}) \oplus K_i$
- Eight 4×16 S-boxes for converting 48 bits to 32 bits output: **Non-linear**; provide major part of the strength of the cipher: Divide 48 bits to 8 units of 6 bits each; the first and the last bit determine the row #; the middle 4 bits determine the column #; The element value is between 0 and 15 → 4 bits
- Straight permutation
- XOR with left half
- Switch the left half and the right half



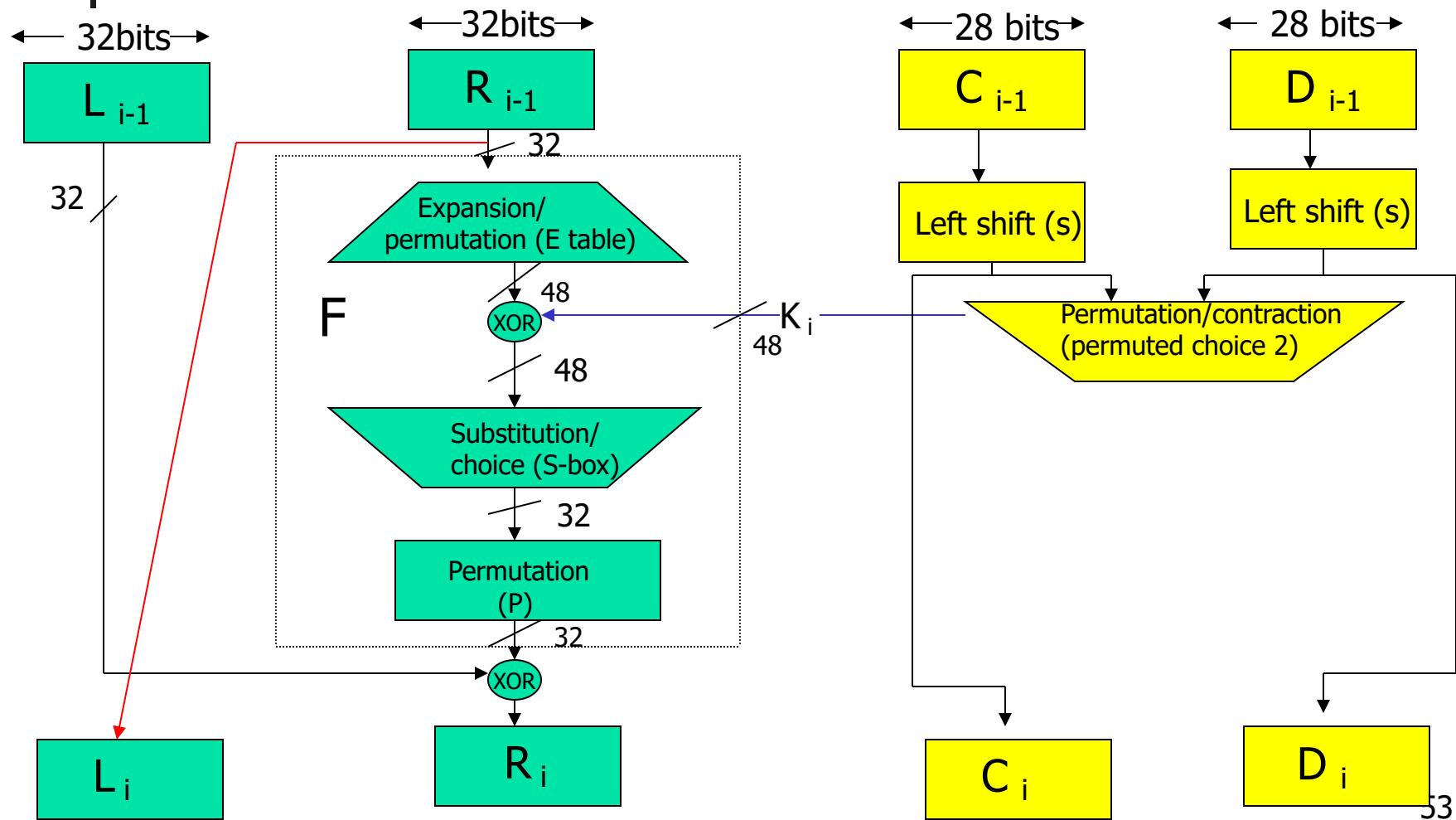
Each round of encryption:

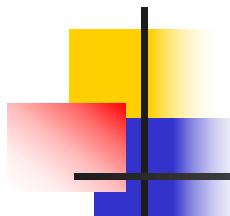
After IP, 64 bits are divided into left-half ($L(0)$) and Right-half ($R(0)$).

- $L(0)$: $R(0)$ is the input to Round 1 of encryption.
During Round1, $L(0):R(0)$ will be operated by F_{k1} to produce $L(1):R(1)$, where F_{k1} is the function F_k with subkey K1.
- .
- .
- Similarly for Round i , $L_{i-1}:R_{i-1}$ would be the input and $L_i:R_i$ will be the output.

Figure 2 shows the function F_{Ki} .

Fig : single Round of DES Algorithm:





i-th Round

The part **in yellow, in the previous slide**, shows the sub key generation. After PC1, the circular rotations are independent for the left half and the right-half.

ENCRYPTION: In the i-th round,

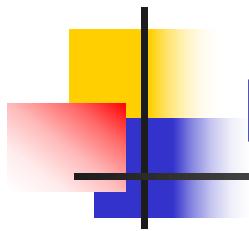
$$L_i = R_{i-1}$$

$$\begin{aligned} R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \\ &= L_{i-1} \oplus P(S(E(R_{i-1}) \oplus K_i)) \end{aligned}$$

Where E: expansion from 32 bits to 48

S: Using 8 S-boxes to convert 48 bits to 32 bits – each S box converts 6 bits to 4 bits

P: permutation

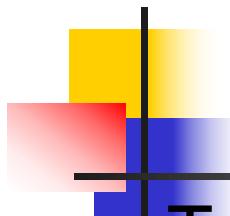


Expansion-Permutation (E/P):

- In figure 2, the E-table generates 48-bit output from 32 bit input by expansion-permutation by using table T6.

Table T6: E/P

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

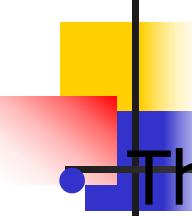


Expansion-Permutation (continued)

- Table T6 in fact divides the input of 32 bits into 8 units of 4 bit each.
- Each unit is converted to 6 bits by borrowing the two adjoining bits of each unit of 4 bits.
- Thus efgh i j k l m n o p
is converted by Expansion/permutation through table T6 to defghi h i j k l m l m n o p q

AUTOKEYING/ AUTOCLAVING: The duplicated outside bits in each group of 6 act as a key to the following S-box, allowing the data (as well as the key) to influence the S-box operation.

- This results in a 48 bit output from E/P module.

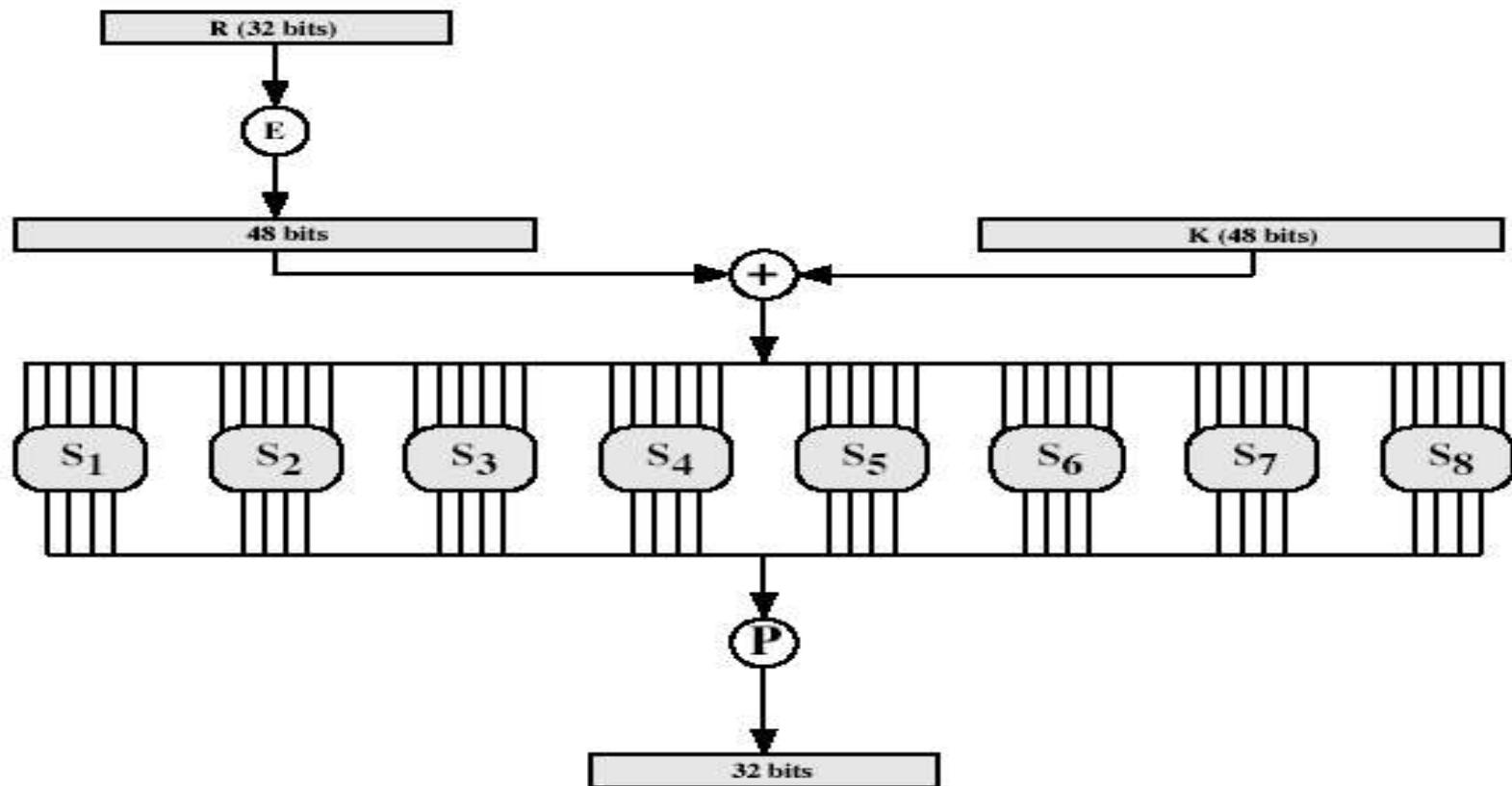


Output of E/P and S-Boxes

- The output from the E/P module is divided into 8 groups of 6 bits each.
- Using 8 4×16 S-boxes, each group of 6 bits is reduced to 4 bits as follows:
- For each S box: Row Number = Outermost 2 bits; Column Number = Inner 4 bits.
- Using the row and column number, the S-box yields a decimal number (lying between 0 and 15). Its 4 bit binary equivalent is the output of the S-box.

Reference: A F Webster & S E Tavares "On the Design of S-boxes", in Advances in Cryptology - Crypto 85, Lecture Notes in Computer Science, No 218, Springer-Verlag, 1985, pp 523-534

DES Round Structure



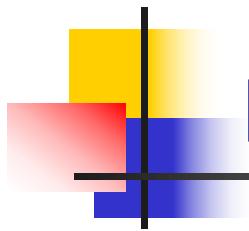
Definition of DES S-boxes:

- 8 S-boxes are shown in table T-7.

Table T7: S-Boxes

	0	1	2													15	
S ₁	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	0
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	1
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	2
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	3

S ₂	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	



Definition of DES S-boxes: S3 and S4

s_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

s_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Definition of DES S-boxes: S5 and S6

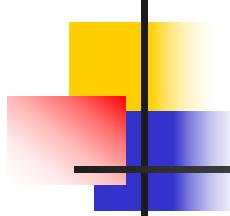
S ₅	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S ₆	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Definition of DES S-boxes: S7 and S8

S ₇	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S ₈	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11



Permutation Function (P):

- 8 S-boxes give 32 bit output, which is passed through Permutation (P).
- P is shown in table T-8.

Table T8: P

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

Example: Evaluation of F:

Given R_{i-1} and K_i : E, S boxes and P

$$E(R_{i-1}) = E(004df6fb) = 20\ 00\ 09\ 1b\ 3e\ 2d\ 1f\ 36$$

Converts 32 bits to 48 bits. The output is in 8 groups of 6bits – the first digit = 2bits; the second digit = 4 bits

$$K_i = 38\ 09\ 1b\ 26\ 2f\ 3a\ 27\ 0f$$

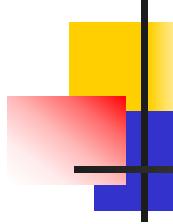
-- K1 from Slide 44

$$E(R_{i-1}) \oplus K_i = 18\ 09\ 12\ 3d\ 11\ 17\ 38\ 39$$

Input to S boxes: 48bits divided into 8 groups of 6 bits each.

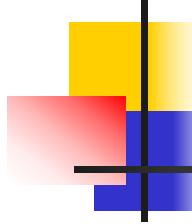
$$S(18\ 09\ 12\ 3d\ 11\ 17\ 38\ 39) = 5fd25e03$$

$$P(5fd25e03) = 746fc91a$$



Key Schedule Algorithm

- Each sub-key K_i : 48 bits: obtained from a 56 bit key K
- Fixed Permutation: $PC1(K) = C_0:D_0$
- A left circular shift (of 1 or 2 bits) on the Left-half (C_0) and Right-half (D_0) *separately*
(Output: C_1 of 28 bits and D_1 of 28 bits)
- 2 bits: for rounds 3-8 and 10-15
- Compression permutation $PC2$ to get 48 bit key K_i from $C_i:D_i$
- Round-dependent left shifts → different parts of initial key create each sub-key



Sub Key Generation

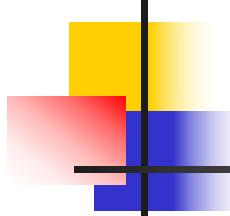
The input key: 56 bits

Hardware Design: the 8, 16, 24, 32, 40, 48, 56 and 64th bit is always the **odd parity bit**. → 64 bit key

Software design: the key is stated in ASCII code. Each character of 8 bits, with the first bit being zero plus 7 bits of code. (!)

Since DES was designed with the viewpoint of hardware implementation, the conversion to 56 bits is done by neglecting every 8th bit.

PC1 converts to 56 bits and permutes.



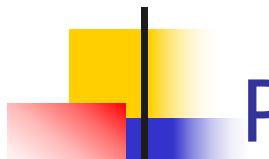
Key Schedule

- K: 64 bit key
- $C_0: D_0 = PC1(K)$, 56 bit key
- 16 steps for $i = 1-15$: A left circular shift (of 1 or 2 bits) on the Left-half (C_{i-1}) and Right-half (D_{i-1}) *separately* (Output: C_i of 28 bits and D_i of 28 bits)
- 16 Subkeys for $i = 1-15$: $K_i = PC2(C_i : D_i)$ of 48 bits each

Input Key

odd parity bit: 8, 16, 24, 32, 40, 48, 56, 64

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
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

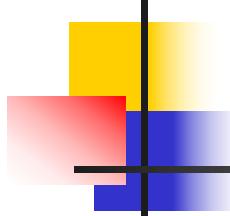


Permuted Choice One (PC-1):

- Tables 8×7 T3 show PC-1.
- Table 6×8 T4 (slide 39) shows PC-2. Table T5 (slide 40) gives the number of shifts

Table T3: PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4



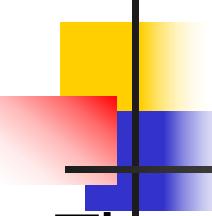
PC1: Obtaining C₀ and D₀

PC1 generates C₀ and D₀, the left and the right halves respectively.

C₀ Read the first column of the input 64-bit key from bottom up. Write it row-wise from left to right.
Repeat for the second, the third and the lower-half of the fourth column respectively.

D₀ Read the seventh column of the input 64-bit key from bottom up. Write it row-wise from left to right.
Repeat for the sixth, the fifth and the upper-half of the fourth column respectively.

Probably the conversion to the two halves was done due to the limitation of the hardware of seventies.

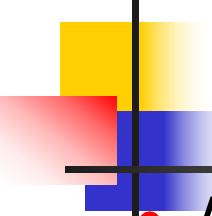


Sub Key Generation: continued

Thus DES has a 56 bit key K consisting of C_0 and D_0 . All the sub keys K_1 to K_{16} are of 48 bits.

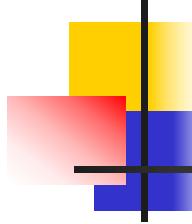
To generate these keys, K goes through

- A Permuted Choice (PC-1) (output C_0 of 28 bits and D_0 of 28 bits).
 - A left circular shift (of 1 or 2 bits) on the Left-half (C_0) and Right-half (D_0) *separately* (Output: C_1 of 28 bits and D_1 of 28 bits)
- followed by a Permuted Choice (PC-2) which permutes as well as 'contracts' to produce a sub-key K_1 of 48 bits.



Sub Key Generation (continued)

- A left circular shift (of 1 or 2 bits) on the Left-half (C_1) and Right-half (D_1) *separately* (Output: C_2 of 28 bits and D_2 of 28 bits)
followed by a Permuted Choice (PC-2) which permutes as well as 'contracts' to produce a sub-key K_2 of 48 bits.
 - .
 - .
 - .
- A left circular shift (of 1 or 2 bits) on the Left-half (C_{15}) and Right-half (D_{15}) *separately* (Output: C_{16} of 28 bits and D_{16} of 28 bits)
followed by a Permuted Choice (PC-2) which permutes as well as 'contracts' to produce a sub-key K_{16} of 48 bits.



Key Schedule

- $KA = PC1(K)$
- $KB1 = LS-j(KA);$

$LS-j$ is left circular shift by j bits, on the two halves of the 56 bits separately. j is given by Table 5.

$$KB2 = LS-j(KB1)$$

$$KB3 = LS-j(KB2)$$

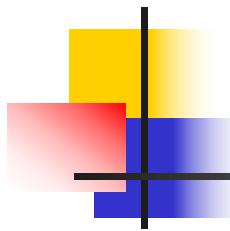
.

$$KB_i = LS-j(KB_{i-1})$$

.

$$KB16 = LS-j(KB15)$$

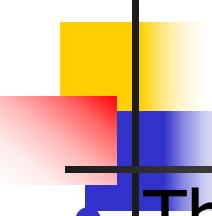
- $Ki = PC2(KB_i)$



Permuted Choice Two (PC-2):

Table T-4: PC-2: The upper 3 rows (24bits) refer to the left half Ci. (It affects S-boxes 1 to 4.) Similarly the remaining 24 bits refer to Di (and affect S-boxes 5 to 8).

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

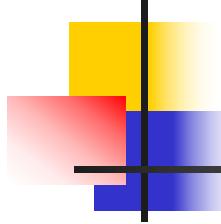


Schedule of left shifts:

- The number of circular left shift in each round is given in Table T5.
- 2 bits: for rounds 3-8 and 10-15:
- 1 bit: for rounds 1, 2, 9, 16 only. Total = 28

Table T-5 The number of circular Left Shifts

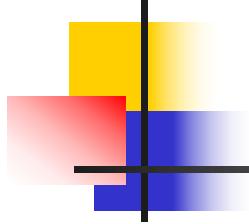
Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



Schedule of left shifts:

- $C_0 \rightarrow C_1$ with 1 shift
- $C_1 \rightarrow C_2$ with 1 shift
- $C_2 \rightarrow C_3$ with 2 shifts
- .
- .
- $C_8 \rightarrow C_9$ with 2 shifts
- .
- .
- $C_{15} \rightarrow C_{16}$ with 1 shift

D_i follows the same shifts as C_i .



Left Shifts in Key schedule

- The effect of left-shifts:
 - First, any bit would affect a different S-box in successive rounds.
 - Secondly if a bit is not used in one sub-key, it would be in the next.
 - Thirdly after 16 rounds it comes back to the original value so that computing the sub-keys for decryption becomes easy.

Example Trace of DES Key Schedule:

Reference: Lawrie Brown, "Cryptography: lecture 8". available at <http://www.cs.adfa.edu.au/archive/teaching/studinfo/ccs3/lectures/less08.html>

- keyinit(5b5a5767, 6a56676e)
- PC1(Key) C=00ffd82, D=f fec937

Key = PC2(C,D): Given in 16 HEX digits, in 8 pairs. The first digit is of 2bits (a value lying between 0 and 3), the second digit is of 4bits.

- KeyRnd01 C=01ffb04, D=ffd926f,
PC2(C,D)=(38 09 1b 26 2f 3a 27 0f)
- KeyRnd02 C=03ff608, D=ffb24df,
PC2(C,D)=(28 09 19 32 1d 32 1f 2f)
- KeyRnd03 C=0ffd820, D=fec937f,
PC2(C,D)=(39 05 29 32 3f 2b 27 0b)
- KeyRnd04 C=3ff6080, D=fb24dff,
PC2(C,D)=(29 2f 0d 10 19 2f 1d 3f)
- KeyRnd05 C=ffd8200, D=ec937ff,
PC2(C,D)=(03 25 1d 13 1f 3b 37 2a)

Example Trace of DES Key Schedule:

(continued-2)

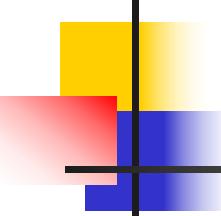
- KeyRnd06 C=ff60803, D=b24ffff,
PC2(C,D)=(1b 35 05 19 3b 0d 35 3b)
- KeyRnd07 C=fd8200f, D=c937ffe,
PC2(C,D)=(03 3c 07 09 13 3f 39 3e)
- KeyRnd08 C=f60803f, D=24dfffb,
PC2(C,D)=(06 34 26 1b 3f 1d 37 38)
- KeyRnd09 C=ec1007f, D=49bfff6,
PC2(C,D)=(07 34 2a 09 37 3f 38 3c)
- KeyRnd10 C=b0401ff, D=26ffffd9,
PC2(C,D)=(06 33 26 0c 3e 15 3f 38)
- KeyRnd11 C=c1007fe, D=9bfff64,
PC2(C,D)=(06 02 33 0d 26 1f 28 3f)

Example Trace of DES Key Schedule:

(continued-3)

- KeyRnd12 C=0401ffb, D=6ffffd92,
PC2(C,D)=(14 16 30 2c 3d 37 3a 34)
- KeyRnd13 C=1007fec, D=bfff649,
PC2(C,D)=(30 0a 36 24 2e 12 2f 3f)
- KeyRnd14 C=401ffb0, D=ffffd926,
PC2(C,D)=(34 0a 38 27 2d 3f 2a 17)
- KeyRnd15 C=007fec1, D=fff649b,
PC2(C,D)=(38 1b 18 22 1d 32 1f 37)
- KeyRnd16 C=00ffd82, D=ffec937,
PC2(C,D)=(38 0b 08 2e 3d 2f 0e 17)

Note that $C_{16} = C_0$ and $D_{16} = D_0$

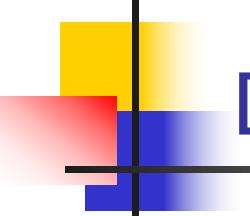


Example:

- Given: Slide 32: plaintext and $IP(\text{plaintext}) = L0:R0$
- $F_{k1}(R0) = 746fc91a$ -- Slide 59
- $L0 = ffb2194d,$
- $R1 = (F_{k1}(R0) \oplus L0) = 8bddd057$
- $L1 = R0 = 004df6fb$

After 16 rounds, IP^{-1} is the last step, for getting the encrypted block.

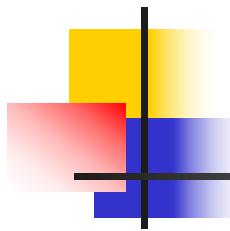
*Reference: Lawrie Brown, "Cryptography: lecture 8". available at
<http://www.cs.adfa.edu.au/archive/teaching/studinfo/ccs3/lectures/less08.html>*



DES Decryption:

Decryption uses the same algorithm as encryption except that the application of the sub-keys is reversed.:

- In the first round of decryption, sub-key K16 is used.
- .
- .
- .
- In the 16th round of decryption, sub-key K1 is used .



Decryption Relations

ENCRYPTION: (from slide 49)

$$L_i = R_{i-1}$$

$$\begin{aligned} R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \\ &= L_{i-1} \oplus P(S(E(R_{i-1}) \oplus K_i)) \end{aligned}$$

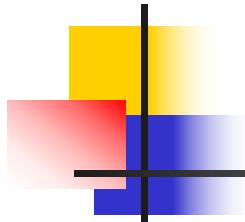
Rewriting: **DECRYPTION** relations are:

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$

On substituting the value of R_{i-1} from the first decryption relation,

$$L_{i-1} = R_i \oplus F(L_i, K_i)$$



Decryption Process

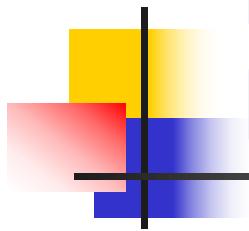
- First: IP on ciphertext: undoes the final IP⁻¹ step of encryption
 - 16 Rounds: First round with subkey 16 undoes 16th round of encryption

.

.

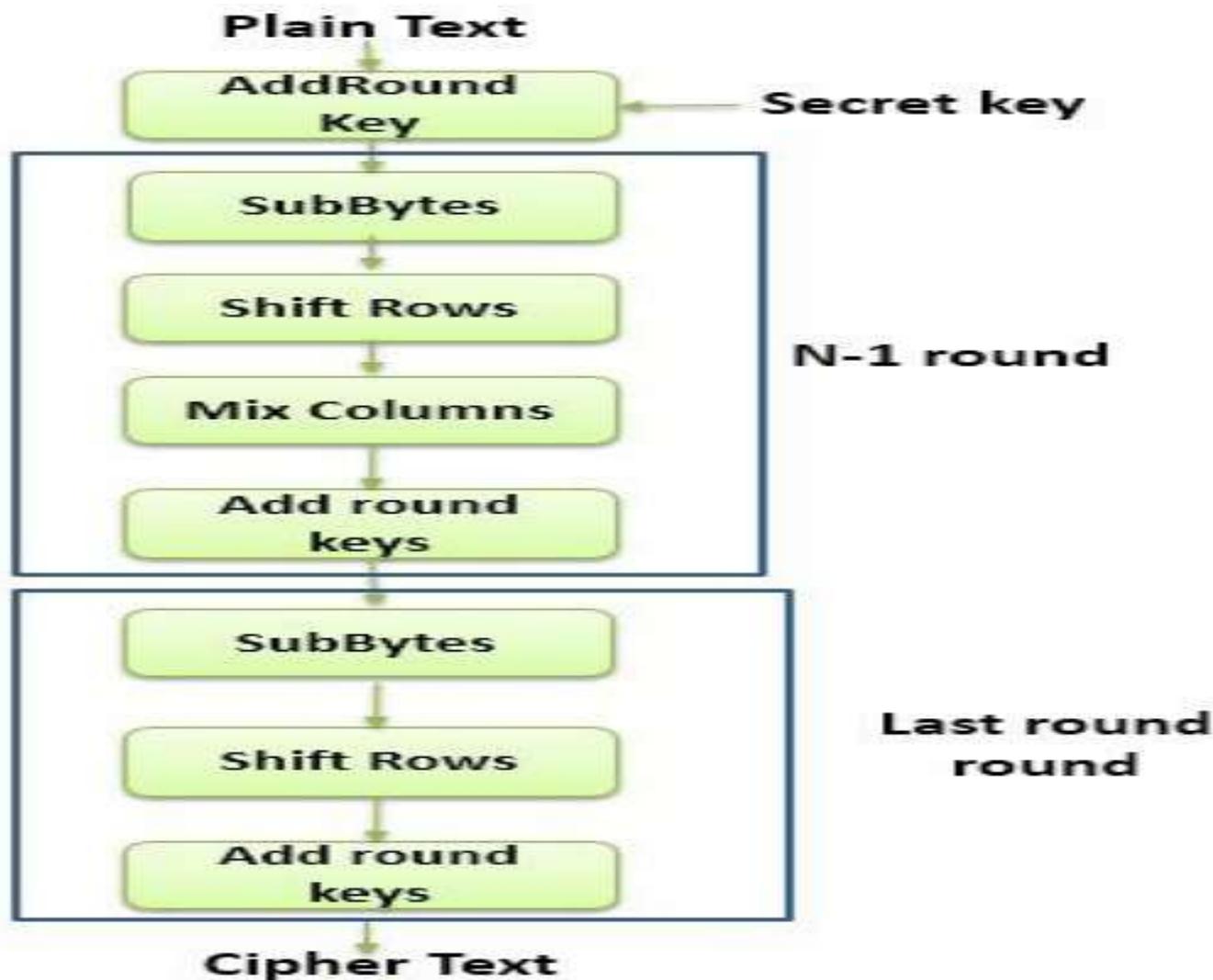
.

- Sixteenth round with subkey 1 undoes 1st encryption round
- Last: IP⁻¹ undoes the initial encryption IP

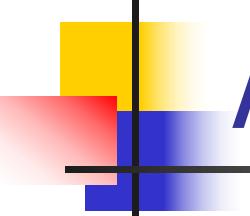


Definition of AES (Advanced Encryption Standard)

- Advanced Encryption Standard (AES) is also a **symmetric key block cipher**. AES was published in **2001** by the **National Institute of Standards and Technology**.
- AES was introduced to replace DES as DES uses very small cipher key and the algorithm was quite slower.



Advanced Encryption Standard



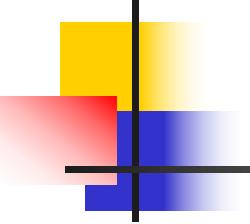
AES algorithm

- AES algorithm takes 128-bit plaintext and 128-bit secret key which together forms a 128-bit block which is depicted as 4 X 4 square matrix. This 4 X 4 square matrix undergoes an initial transformation. This step is followed by the 10 rounds. Among which 9 round contain following stages:
- **Sub-bytes:** It uses S-box by which it performs byte by byte substitution of the entire block (matrix).
- **Shift Rows:** Rows of the matrix are shifted.
- **Mix Columns:** Columns are of the matrix are shuffled from right to left.
- **Add round keys:** Here, the Xor of the current block and the expanded key is performed.
- And the last 10th round involves Subbytes, Shift Rows, and Add round keys stages only and provides 16 bytes (128-bit) ciphertext.

Difference Between DES (Data Encryption Standard) and AES (Advanced Encryption Standard)

BASIS FOR COMPARISON	DES (DATA ENCRYPTION STANDARD)	AES (ADVANCED ENCRYPTION STANDARD)
Basic	In DES the data block is divided into two halves.	In AES the entire data block is processed as a single matrix.
Principle	DES work on Feistel Cipher structure.	AES works on Substitution and Permutation Principle.
Plaintext	Plaintext is of 64 bits	Plaintext can be of 128,192, or 256 bits
Key size	DES in comparison to AES has smaller key size.	AES has larger key size as compared to DES.
Rounds	16 rounds	10 rounds for 128-bit algo 12 rounds for 192-bit algo 14 rounds for 256-bit algo
Rounds Names	Expansion Permutation, Xor, S-box, P-box, Xor and Swap.	Subbytes, Shiftrows, Mix columns, Addroundkeys.
Security	DES has a smaller key which is less secure.	AES has large secret key comparatively hence, more secure.
Speed	DES is comparatively slower.	AES is faster.

Key Differences Between DES and AES

- 
1. The basic difference between DES and AES is that the block in DES is divided into two halves before further processing whereas, in AES entire block is processed to obtain ciphertext.
 2. The DES algorithm works on the Feistel Cipher principle, and the AES algorithm works on substitution and permutation principle.
 3. The key size of DES is 56 bit which is comparatively smaller than AES which has 128,192, or 256-bit secret key.
 4. The rounds in DES include Expansion Permutation, Xor, S-box, P-box, Xor and Swap. On the other hands, rounds in AES include Subbytes, Shiftrows, Mix columns, Addroundkeys.
 5. DES is less secure than AES because of the small key size.
 6. AES is comparatively faster than DES

Modular Arithmetic

We begin by defining how to perform basic arithmetic modulo n , where n is a positive integer. Addition, subtraction, and multiplication follow naturally from their integer counterparts, but we have complications with division.

Euclid's Algorithm

We will need this algorithm to fix our problems with division. It was originally designed to find the greatest common divisor of two numbers.

Division

Once armed with Euclid's algorithm, we can easily compute divisions modulo n .

The Chinese Remainder Theorem

We find we only need to study \mathbb{Z}_{pk} where p is a prime, because once we have a result about the prime powers, we can use the Chinese Remainder Theorem to generalize for all n .

Units

While studying division, we encounter the problem of inversion. Units are numbers with inverses.

Exponentiation

The behaviour of units when they are exponentiated is difficult to study. Modern cryptography exploits this.

Order of a Unit

If we start with a unit and keep multiplying it by itself, we wind up with 1 eventually. The order of a unit is the number of steps this takes.

The Miller-Rabin Test

We discuss a fast way of telling if a given number is prime that works with high probability.

Generators

Sometimes powering up a unit will generate all the other units.

Cyclic Groups

We focus only on multiplication and see if we can still say anything interesting.

Quadratic Residues

Elements of Z_n that are perfect squares are called quadratic residues.

Bonus Material

Euclid's Algorithm

Given three integers a, b, c , can you write c in the form

$$c=ax+by$$

for integers x and y ? If so, is there more than one solution? Can you find them all? Before answering this, let us answer a seemingly unrelated question:

How do you find the greatest common divisor (gcd) of two integers a, b ?

We denote the greatest common divisor of a and b by $\text{gcd}(a, b)$, or sometimes even just (a, b) . If $(a, b)=1$ we say a and b are *coprime*.

The obvious answer is to list all the divisors a and b , and look for the greatest one they have in common. However, this requires a and b to be factorized, and no one knows how to do this efficiently.

A few simple observations lead to a far superior method: Euclid's algorithm, or the Euclidean algorithm. First, if d divides a and d divides b , then d divides their difference, $a - b$, where a is the larger of the two. But this means we've shrunk the original problem: now we just need to find $\text{gcd}(a, a-b)$. We repeat until we reach a trivial case.

Hence we can find $\text{gcd}(a, b)$ by doing something that most people learn in primary school: division and remainder. We give an example and leave the proof of the general case to the reader.

Suppose we wish to compute $\text{gcd}(27, 33)$. First, we divide the bigger one by the smaller one:

$$33 = 1 \times 27 + 6$$

Thus $\gcd(33,27)=\gcd(27,6)$. Repeating this trick:

$$27=4 \times 6 + 3$$

and we see $\gcd(27,6)=\gcd(6,3)$. Lastly,

$$6=2 \times 3 + 0$$

Since 6 is a perfect multiple of 3, $\gcd(6,3)=3$, and we have found that $\gcd(33,27)=3$.

This algorithm does not require factorizing numbers, and is fast. We obtain a crude bound for the number of steps required by observing that if we divide a by b to get $a=bq+r$, and $r>b/2$, then in the next step we get a remainder $r' \leq b/2$. Thus every two steps, the numbers shrink by at least one bit.

Extended Euclidean Algorithm

The above equations actually reveal more than the gcd of two numbers. We can use them to find integers m,n such that

$$3=33m+27n$$

First rearrange all the equations so that the remainders are the subjects:

$$6=33-1 \times 27$$

$$3=27-4 \times 6$$

Then we start from the last equation, and substitute the next equation into it:

$$3=27-4 \times (33-1 \times 27)=(-4) \times 33+5 \times 27$$

And we are done: $m=-4, n=5$.

If there were more equations, we would repeat until we have used them all to find m and n.

Thus in general, given integers a and b, let $d=\gcd(a,b)$. Then we can find integer m and n such that

$$d=ma+nb$$

Using the extended Euclidean algorithm.

The General Solution

We can now answer the question posed at the start of this page, that is, given integers a, b, c find all integers x, y such that

$$c = xa + yb.$$

Let $d = \gcd(a, b)$, and let $b = b'd, a = a'd$. Since $xa + yb$ is a multiple of d for any integers x, y , solutions exist only when d divides c .

So say $c = kd$. Using the extended Euclidean algorithm we can find m, n such that $d = ma + nb$, thus we have a solution $x = km, y = kn$.

Suppose x', y' is another solution. Then

$$c = xa + yb = x'a + y'b$$

Rearranging,

$$(x' - x)a = (y' - y)b$$

Dividing by d gives:

$$(x' - x)a' = (y' - y)b'$$

The numbers a' and b' are coprime since d is the greatest common divisor, hence $(x' - x)$ is some multiple of b' , that is:

$$x' - x = tb/d$$

for some integer t . Then solving for $(y' - y)$ gives

$$y' - y = ta/d$$

Thus $x' = x + tb/d$ and $y' = y - ta/d$ for some integer t .

But if we replace t with any integer, x' and y' still satisfy $c = x'a + y'b$. Thus there are infinitely many solutions, and they are given by

$$x = km + tb/d, y = kn - ta/d.$$

for all integers t .

Later, we shall often wish to solve $1 = xp + yq$ for coprime integers p and q . In this case, the above becomes

$$x=m+tq, y=n+tp.$$

Division

Intuitively, [to divide x by y means to find a number z such that y times z is x](#), but we had trouble adopting this definition of division because sometimes there is more than one possibility for z modulo n.

We solved this by only defining division when the answer is unique. We stated without proof that when division defined in this way, one can divide by y if and only if y^{-1} , the inverse of y exists.

We shall now show why this is the case. We wish to find all z such that $yz=x(\text{mod } n)$, which by definition means

$$x=zy+kn$$

for some integer k. But this is precisely the problem we encountered when discussing [Euclid's algorithm](#)! Let $d=\gcd(y, n)$.

Suppose $d>1$. Then no solutions exist if x is not a multiple of d. Otherwise the solutions for z,k are

$$z=r+tn/d, k=s-ty/d$$

for some integers r,s (that we get from Euclid's algorithm) and for all integers t. But this means z does not have a unique solution modulo n since $n/d < n$. (Instead z has a unique solution modulo n/d .)

On the other hand, if $d=1$, that is if y and n are coprime, then x is always a multiple of d so solutions exist. Recall we find them by using Euclid's algorithm to find r,s such that

$$1=ry+sn$$

Then the solutions for z,k are given by

$$z=xr+tn, k=zs-ty$$

for all integers t. Thus z has a unique solution modulo n, and division makes sense for this case.

Also, r satisfies $ry=1(\text{mod } n)$ so in fact $y^{-1}=r$. Thus our claims are correct: we can divide x by y if and only if y has an inverse.

We can also see that y has an inverse if and only if $\gcd(y,n)=1$. (Actually, we have only proved some of these statements in one direction, but the other direction is easy.)

Note: even though we cannot always divide x by y modulo n , sometimes we still need to find all z such that $yz=x$.

Computing Inverses

We previously asked: given $y \in \mathbb{Z}_n$, does y^{-1} exist, and if so, what is it?

Our answer before was that since \mathbb{Z}_n is finite, we can try every possibility. But if n is large, say a 256-bit number, this cannot be done even if we use the fastest computers available today.

A better way is to use what we just proved: y^{-1} exists if and only if $\gcd(y,n)=1$ (which we can check using Euclid's algorithm), and y^{-1} can be computed efficiently using the extended Euclidean algorithm.

Example: does $7^{-1}(\bmod 19)$ exist, and if so, what is it? Euclid's algorithm gives

$$19 = 2 \times 7 + 5$$

$$7 = 1 \times 5 + 2$$

$$5 = 2 \times 2 + 1$$

Thus an inverse exists since $\gcd(7, 19) = 1$. To find the inverse we rearrange these equations so that the remainders are the subjects. Then starting from the third equation, and substituting in the second one gives

$$1 = 5 - 2 \times 2 = 5 - 2 \times (7 - 1 \times 5) = (-2) \times 7 + 3 \times 5$$

Now substituting in the first equation gives

$$1 = (-2) \times 7 + 3 \times (19 - 2 \times 7) = (-8) \times 7 + 3 \times 19$$

from which we see that $7^{-1} = -8 = 11(\bmod 19)$.

The Chinese Remainder Theorem

Suppose we wish to solve

$$x = 2(\bmod 5)$$

$$x \equiv 3 \pmod{7}$$

for x . If we have a solution y , then $y+35$ is also a solution. So we only need to look for solutions modulo 35. By brute force, we find the only solution is $x \equiv 17 \pmod{35}$.

For any system of equations like this, the Chinese Remainder Theorem tells us there is always a unique solution up to a certain modulus, and describes how to find the solution efficiently.

Theorem: Let p, q be coprime. Then the system of equations

$$x \equiv a \pmod{p}$$

$$x \equiv b \pmod{q}$$

has a unique solution for x modulo pq .

The reverse direction is trivial: given $x \in \mathbb{Z}_{pq}$, we can reduce x modulo p and x modulo q to obtain two equations of the above form.

Proof: Let $p_1 = p - 1 \pmod{q}$ and $q_1 = q - 1 \pmod{p}$. These must exist since p, q are coprime. Then we claim that if y is an integer such that

$$y \equiv aq_1 + bp_1 \pmod{pq}$$

then y satisfies both equations:

Modulo p , we have $y \equiv aq_1 \equiv a \pmod{p}$ since $q_1 \equiv 1 \pmod{p}$. Similarly $y \equiv b \pmod{q}$. Thus y is a solution for x .

It remains to show no other solutions exist modulo pq .

If $z \equiv a \pmod{p}$ then $z - y$ is a multiple of p . If $z \equiv b \pmod{q}$ as well, then $z - y$ is also a multiple of q . Since p and q are coprime, this implies $z - y$ is a multiple of pq , hence $z \equiv y \pmod{pq}$. ■

This theorem implies we can represent an element of \mathbb{Z}_{pq} by one element of \mathbb{Z}_p and one element of \mathbb{Z}_q , and vice versa. In other words, we have a bijection between \mathbb{Z}_{pq} and $\mathbb{Z}_p \times \mathbb{Z}_q$.

Examples: We can write $17 \in \mathbb{Z}_{35}$ as $(2, 3) \in \mathbb{Z}_5 \times \mathbb{Z}_7$. We can write $1 \in \mathbb{Z}_{pq}$ as $(1, 1) \in \mathbb{Z}_p \times \mathbb{Z}_q$.

In fact, this correspondence goes further than a simple relabelling. Suppose $x, y \in \mathbb{Z}_{pq}$ correspond to $(a, b), (c, d) \in \mathbb{Z}_p \times \mathbb{Z}_q$ respectively. Then a little thought shows $x+y$ corresponds to $(a+c, b+d)$, and similarly xy corresponds to (ac, bd) .

A practical application: if we have many computations to perform on $x \in \mathbb{Z}_{pq}$ (e.g. RSA signing and decryption), we can convert x to $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_q$ and do all the computations on a and b instead before converting back. This is often cheaper because for many algorithms, doubling the size of the input more than doubles the running time.

Example: To compute $17 \times 17 \pmod{35}$, we can compute $(2 \times 2, 3 \times 3) = (4, 2)$ in $\mathbb{Z}_5 \times \mathbb{Z}_7$, and then apply the Chinese Remainder Theorem to find that $(4, 2)$ is $9 \pmod{35}$.

Let us restate the Chinese Remainder Theorem in the form it is usually presented.

For Several Equations

Theorem: Let m_1, \dots, m_n be pairwise coprime (that is $\gcd(m_i, m_j) = 1$ whenever $i \neq j$). Then the system of n equations

$$x \equiv a_1 \pmod{m_1}$$

...

$$x \equiv a_n \pmod{m_n}$$

has a unique solution for x modulo M where $M = m_1 \dots m_n$.

Proof: This is an easy induction from the previous form of the theorem, or we can write down the solution directly.

Define $b_i = M/m_i$ (the product of all the moduli except for m_i) and $b'_i = b_i - 1 \pmod{m_i}$. Then by a similar argument to before,

$$x \equiv \sum_{i=1}^n a_i b'_i b_i \pmod{M}$$

is the unique solution. ■

Prime Powers First

An important consequence of the theorem is that when studying modular arithmetic in general, we can first study modular arithmetic at prime power and

then appeal to the Chinese Remainder Theorem to generalize any results. For any integer n , we factorize n into primes $n=p_1k_1\dots p_m k_m$ and then use the Chinese Remainder Theorem to get

$$Z_n = Z_{p_1}^{k_1} \times \dots \times Z_{p_m}^{k_m}$$

To prove statements in Z_{pk} , one starts from Z_p , and inductively works up to Z_{pk} . Thus the most important case to study is Z_p .

Cryptography and Network Security

Third Edition
by William Stallings

Chapter 9 – 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

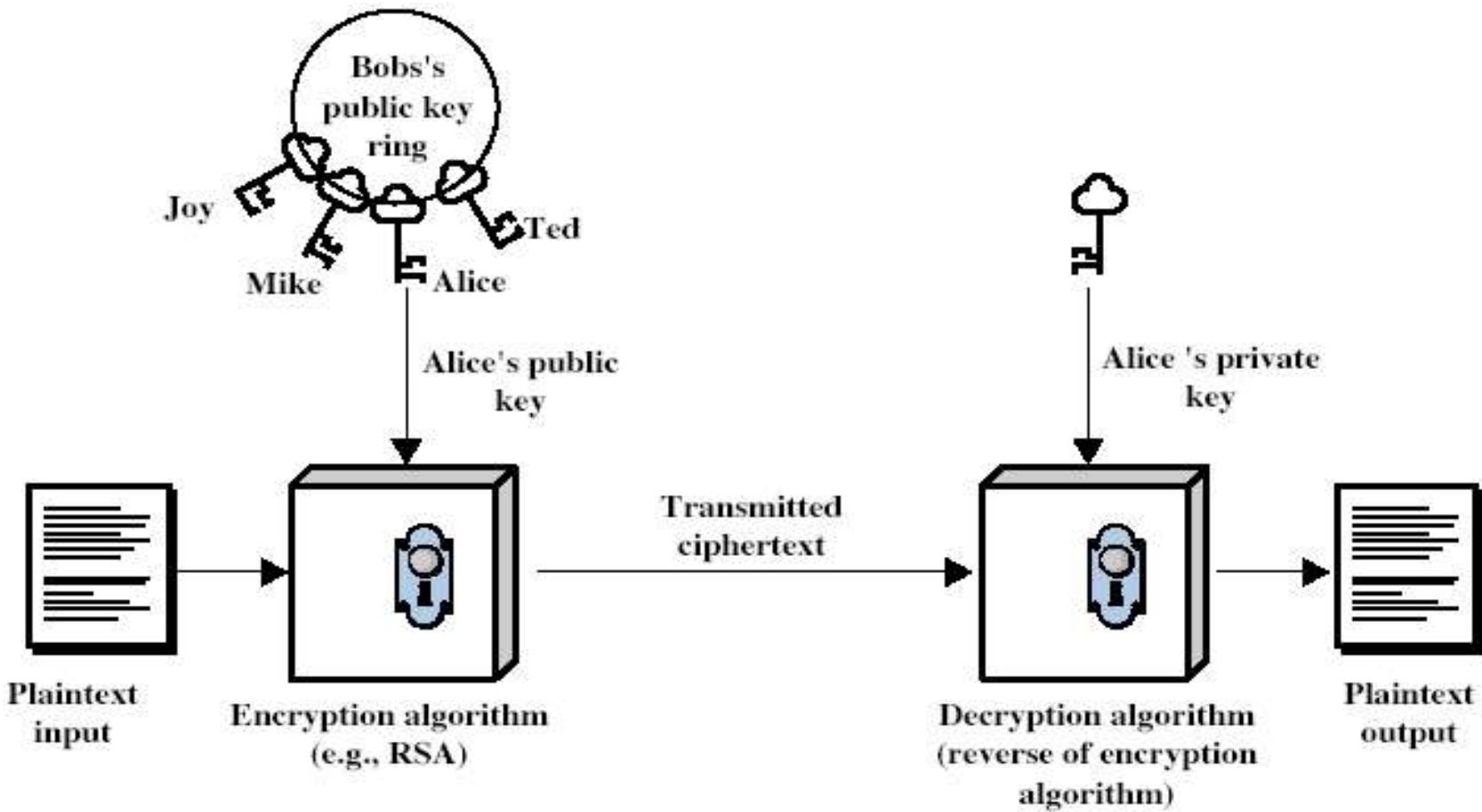
Public-Key Cryptography

- 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

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 **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because
 - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



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 Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
 - computationally infeasible to find decryption key knowing only algorithm & encryption key
 - 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 (in some schemes)

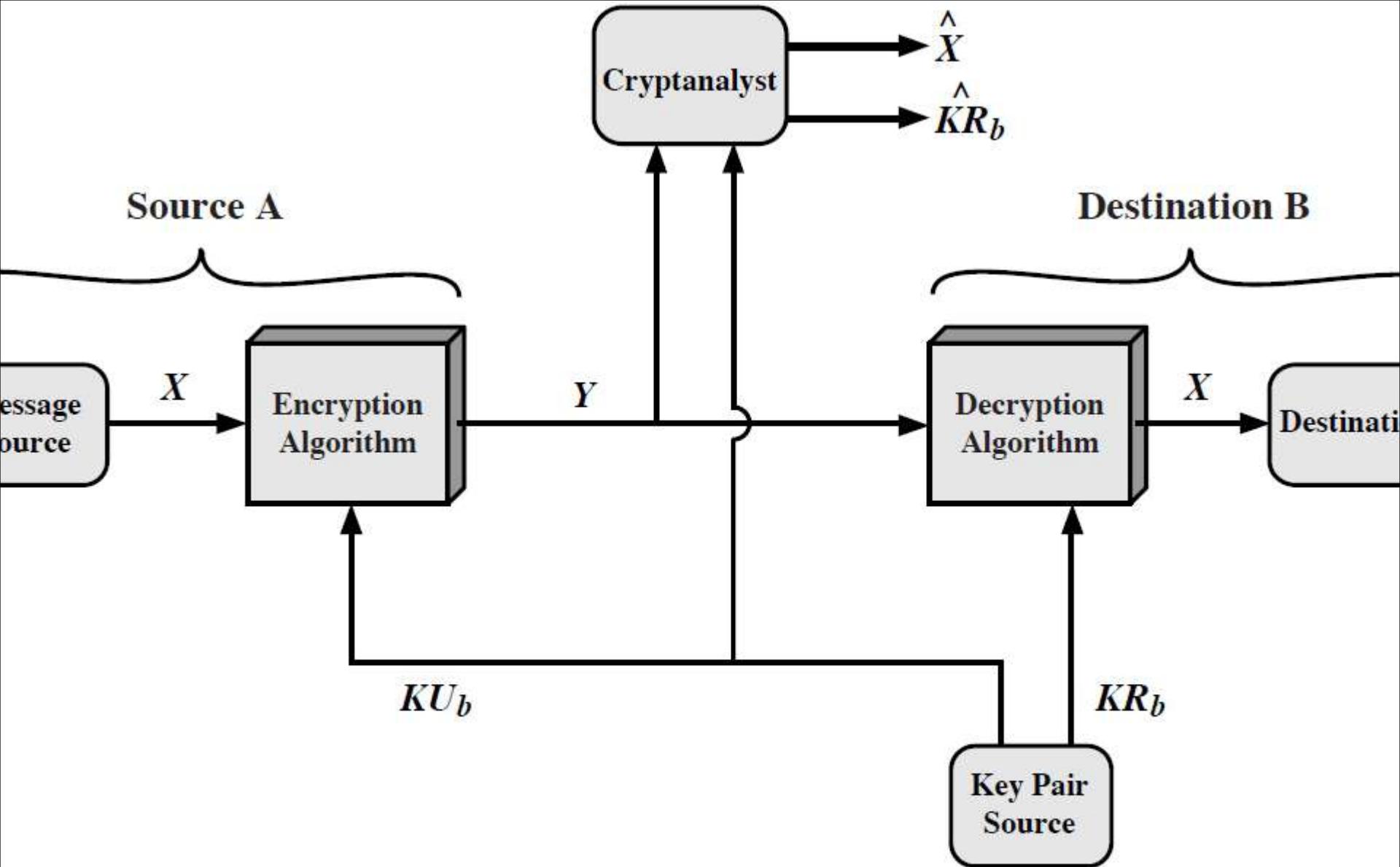


Figure 9.2 Public-Key Cryptosystem: Secrecy

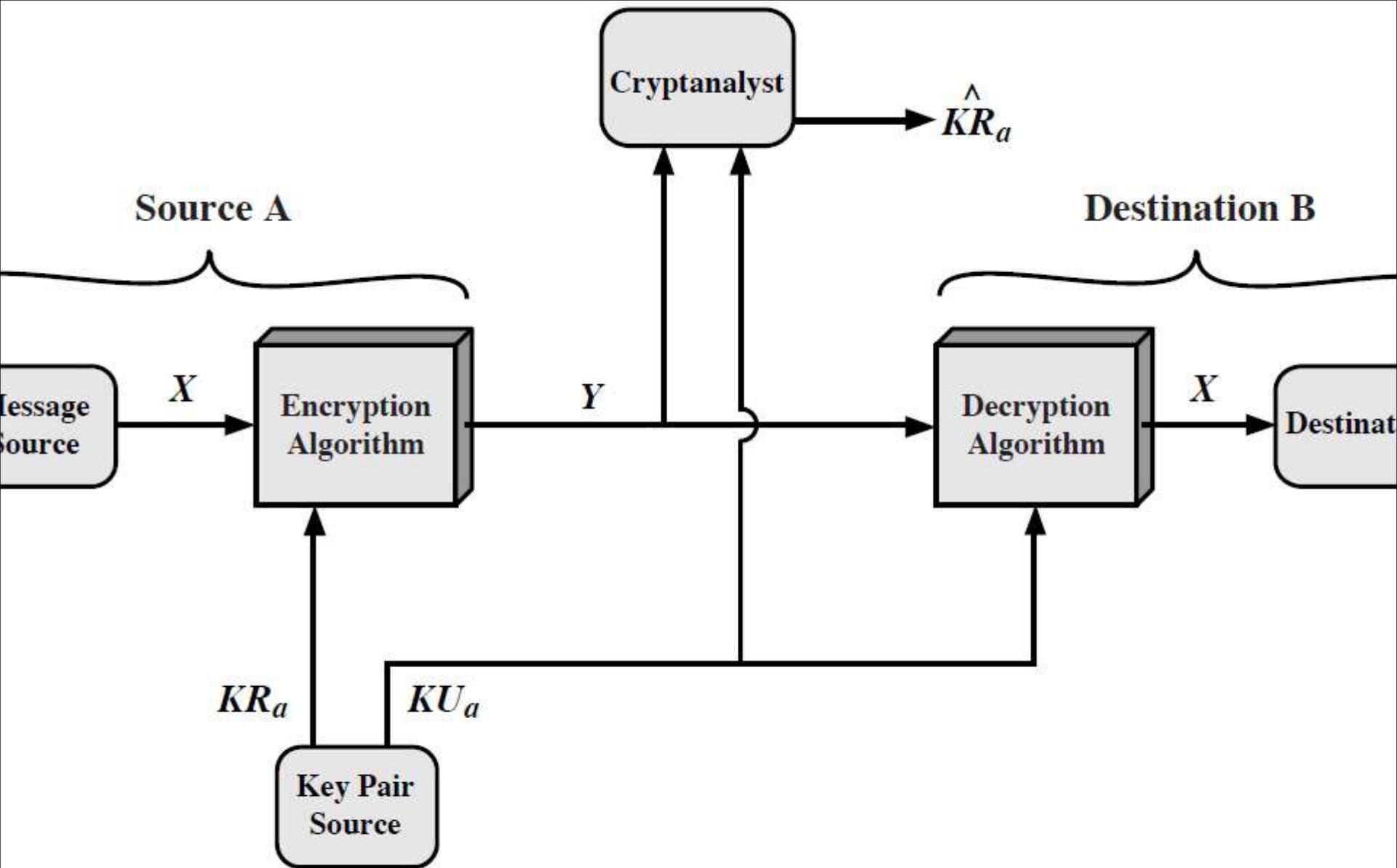


Figure 9.3 Public-Key Cryptosystem: Authentication

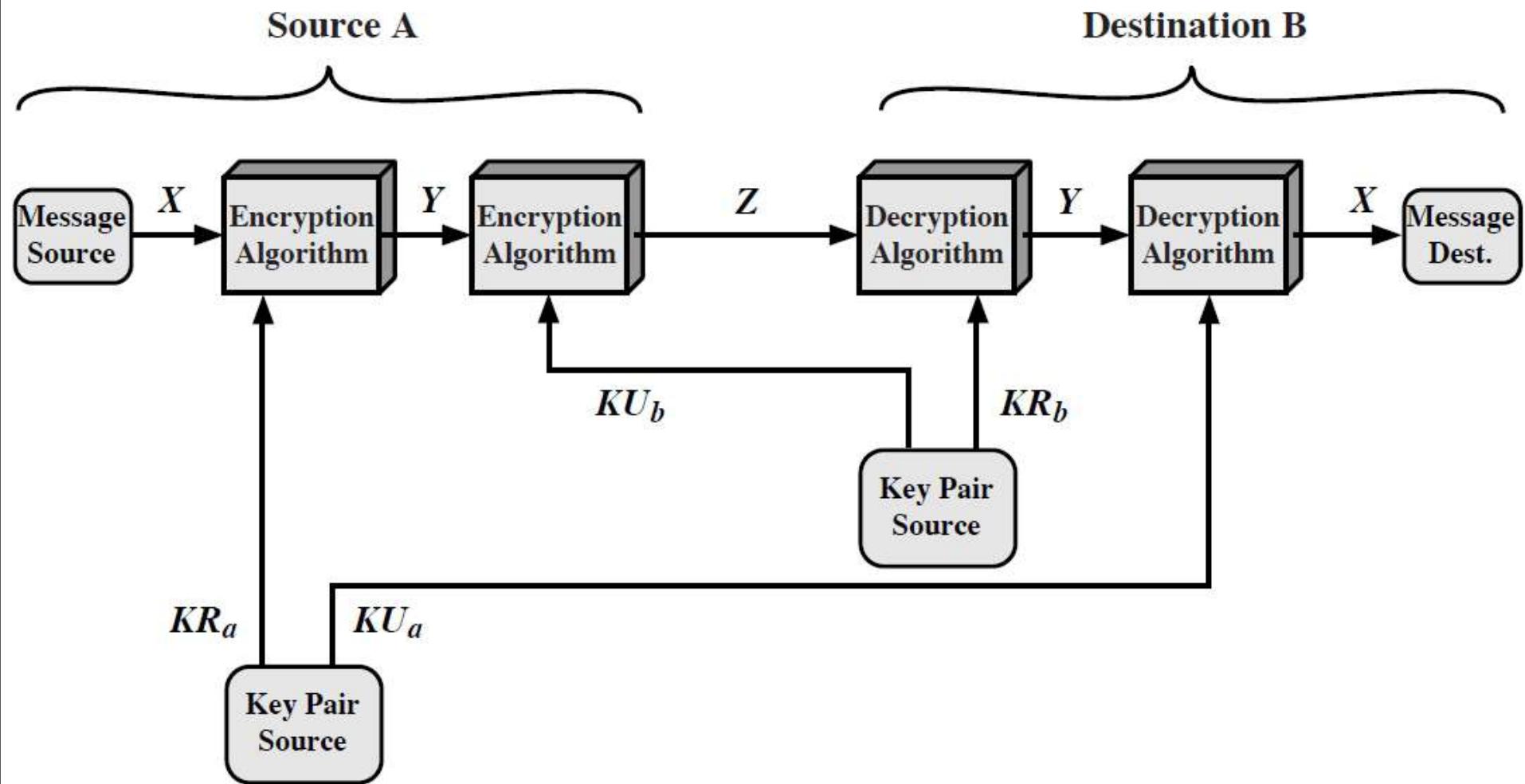


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

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

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, its just made too hard to do in practise
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
 - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- **security due to cost of factoring large numbers**
 - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

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: $KU=\{e, N\}$
- keep secret private decryption key: $KR=\{d, p, q\}$

RSA Use

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

Why RSA Works

- because of Euler's Theorem:
- $a^{\phi(n)} \mod N = 1$
 - where $\gcd(a, N) = 1$
- in RSA have:
 - $N = p \cdot q$
 - $\phi(N) = (p-1)(q-1)$
 - carefully chosen e & d to be inverses $\mod \phi(N)$
 - hence $e \cdot d = 1 + k \cdot \phi(N)$ for some k
- hence :
$$C^d = (M^e)^d = M^{1+k \cdot \phi(N)} = M^1 \cdot (M^{\phi(N)})^q = M^1 \cdot (1)^q = M^1 = M \mod N$$

RSA Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
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 $KU = \{7, 187\}$
7. Keep secret private key $KR = \{23, 187\}$

RSA Example cont

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

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

- decryption:

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

Computational Aspects

- The complexity of the computation required boils down to two aspects, the actual encryption/decryption process and the key generation process.
- 1. Encryption and Decryption: Both involve raising a (large) integer to a (large) integer power modulo n. If the exponentiation was done over the integers and then reduced modulo n, the intermediate values would be gigantic.

2. Key Generation: Before two parties can use a public key system, each must generate a pair of keys. This involves the following tasks:

- Determining two prime numbers p , q .
 - Selecting either e or d and calculating the other.

Firstly, considering selection of p and q . Because the value $n = pq$ will be known to any opponent, to prevent the discovery of p , q through exhaustive methods, these primes must be chosen from a sufficiently large set (must be large numbers)

- On the other hand the method used for finding large primes must be reasonably efficient.
- At present there are no useful techniques that yield arbitrarily large primes.
- The procedure is to pick at random an odd number of the desired magnitude and test that it is prime.
- If not, repeat until a prime is found.

- A variety of tests for primality have been developed, all of which are statistical in nature. The tests however may be run in such a way as to attain a probability, of as near 1 as is desired, that a particular number is prime.
- One of the more efficient algorithms is the Miller-Rabin scheme, which performs calculations on n , the candidate prime and a randomly chosen integer a .

- This procedure may be repeated as required.
- In summary the procedure for picking a prime is as follows:
 - (a) Pick an odd integer n at random (e.g., using a pseudorandom number generator).
 - (b) Pick an integer $a < n$ at random.
 - (c) Perform the probabilistic primality test, (such as Miller-Rabin). If n fails the test then go to step a.
 - (d) If n passes a sufficient number of tests then accept it, otherwise go to step b.

The Security of RSA

- RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits) prime numbers.
- Recovering the plaintext from one key and the cipher text is equivalent to factoring the product of two primes Taking a first look at cryptographic considerations.

- Three possible approaches include:
 1. Brute Force: Try all possible keys. Standard defense is a large key space. The larger e and d are the better, so we have the following:

	5 years ago	Today
Casual use	384 bits	768 bits
Commercial use	512 bits	1024
Military Spec.	1024 bits	4096 bits

- where the military specification is only an estimate due to this information being classified. For comparison, 512 bits is about 150 decimal digits.

- 2. Mathematical attacks:
 - Factor n into its 2 primes thus enabling calculation of $\varphi(n)$ and the private key $e \equiv d - 1 \pmod{\varphi(n)}$. The best known algorithm used in factoring an integer n is time proportional to:
- as discussed in the section on complexity theory. For a 200 digit number this would take about 1000 years on a large machine.
- However, there has been a lot of progress made in factorisation over the last number of years.

- Determining $\varphi(n)$, given n or determining d given n and e . These are at least as time consuming as factoring n so the factorizing performance of algorithms is used as the benchmark to evaluate the security of RSA.

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

3. Timing attacks: These are an implementation attack that depends on the running time of an algorithm. We will look at them in more detail when we study attacks on cryptosystems

Diffie Hellman Key Exchange

- Discovered by Whitfield Diffie and Martin Hellman
 - “New Directions in Cryptography”
- Diffie-Hellman key agreement protocol
 - Exponential key agreement
 - Allows two users to exchange a secret key
 - Requires no prior secrets
 - Real-time over an untrusted network

Introduction

- Security of transmission is critical for many network and Internet applications
- Requires users to share information in a way that others can't decipher the flow of information

“It is insufficient to protect ourselves with laws; we need to protect ourselves with mathematics.”

-Bruce Schneier

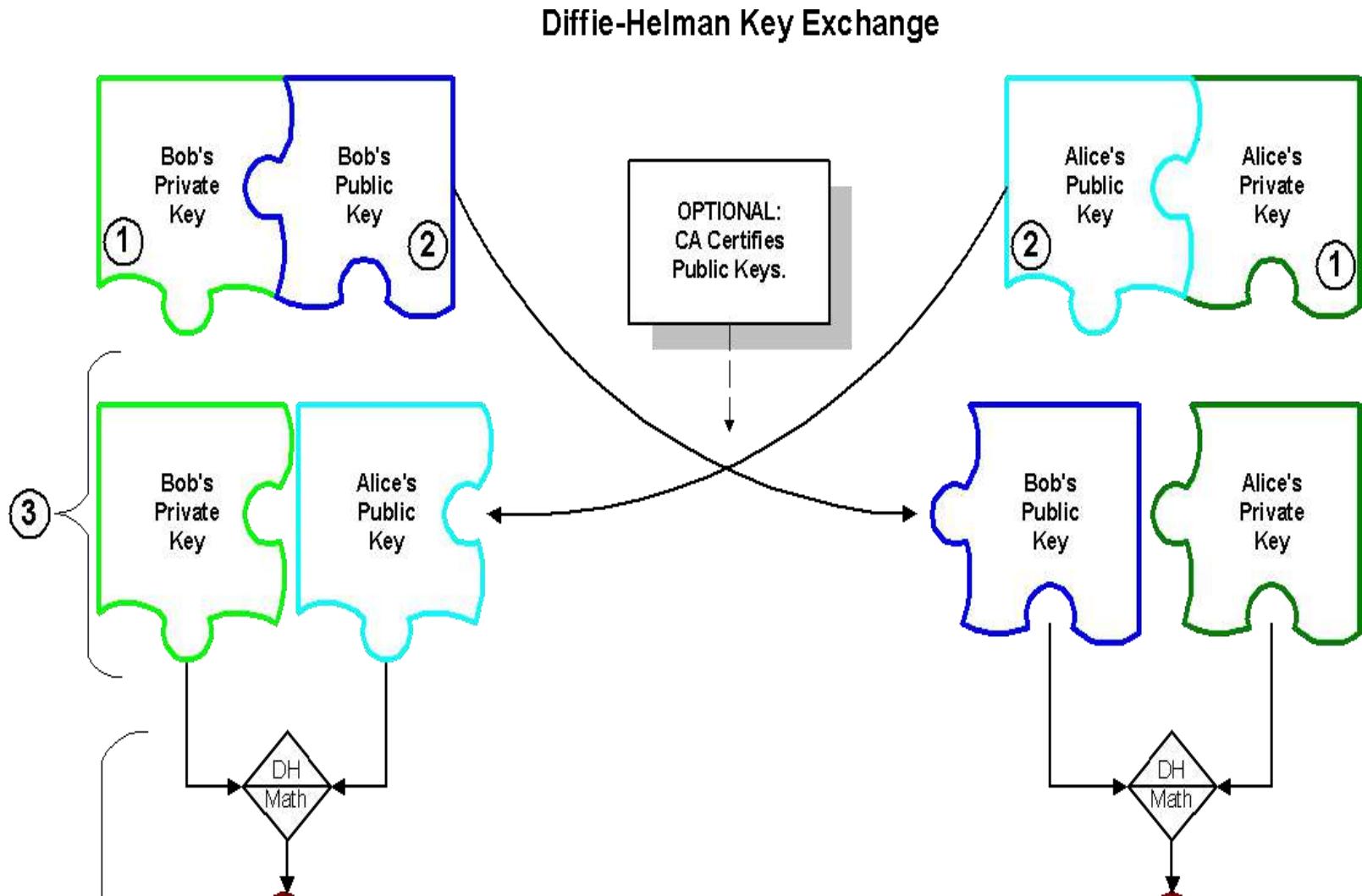
Introduction

- Based on the difficulty of computing discrete logarithms of large numbers.
- No known successful attack strategies*
- Requires two large numbers, one prime (P), and (G), a primitive root of P

Implementation

- P and G are both publicly available numbers
 - P is at least 512 bits
- Users pick private values a and b
- Compute public values
 - $x = g^a \text{ mod } p$
 - $y = g^b \text{ mod } p$
- Public values x and y are exchanged

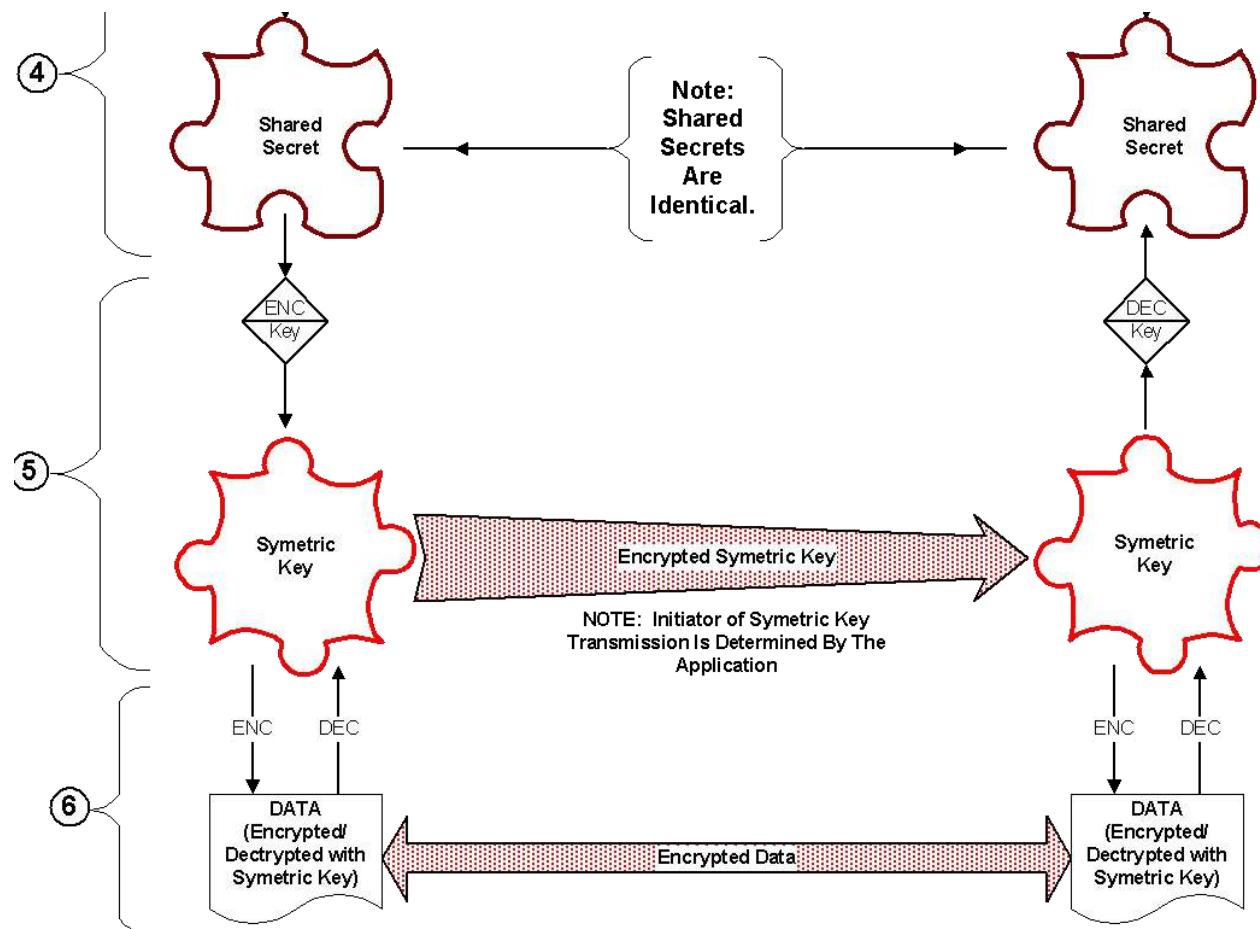
Implementation



Implementation

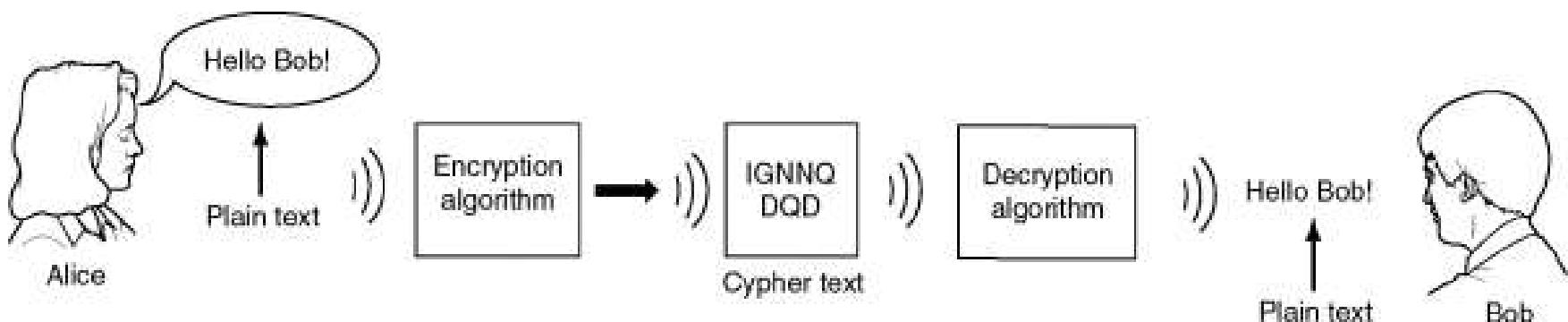
- Compute shared, private key
 - $k_a = y^a \text{ mod } p$
 - $k_b = x^b \text{ mod } p$
- Algebraically it can be shown that $k_a = k_b$
 - Users now have a symmetric secret key to encrypt

Implementation

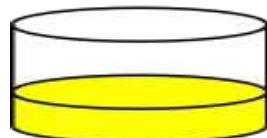


Example

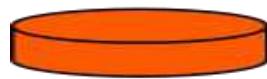
- Two Internet users, Alice and Bob wish to have a secure conversation.
 - They decide to use the Diffie-Hellman protocol



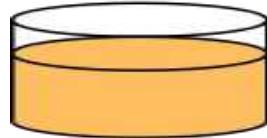
Alice



+

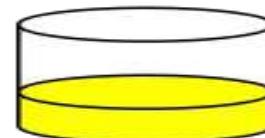


=



Common paint

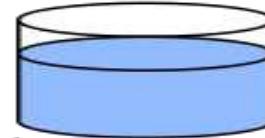
Bob



+



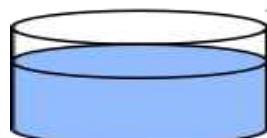
=



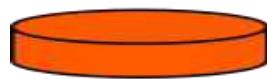
Secret colours

Public transport

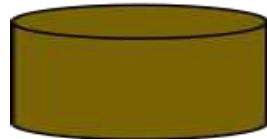
(assume
that mixture separation
is expensive)



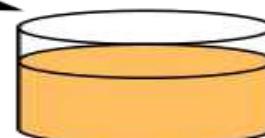
+



=



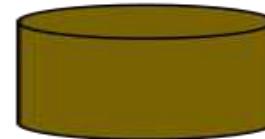
Secret colours



+



=



Common secret

Example

- Bob and Alice are unable to talk on the untrusted network.
 - Who knows who's listening?



B
A
D
G
U
Y

A vertical graphic featuring the words "BAD GUY" in yellow, bold, sans-serif letters, with each letter on its own separate line. The letters are positioned against a black background.

Diffie-Hellman Setup

- all users agree on global parameters:
 - large prime integer or polynomial q
 - a , which is a primitive root mod q
- each user (e.g. A) generates their key
 - chooses a secret key (number): $x_A < q$
 - computes their public key: $y_A = a^{x_A} \text{ mod } q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB} :

$$K_{AB} = a^{x_A \cdot x_B} \bmod q$$

$$= y_A^{x_B} \bmod q$$

(which B can compute)

$$= y_B^{x_A} \bmod q$$

(which A can compute)

Each principal has the other's public key and their own secret, along with a and q.

Diffie-Hellman Key Exchange

Agree on a and q

$$y_B = a^{x_B} \bmod q$$

$$y_A = a^{x_A} \bmod q$$

$$K_{AB} = y_A^{x_B} \bmod q$$

$$K_{AB} = y_B^{x_A} \bmod q$$

Both Alice and Bob have

$$K_{AB} = a^{x_A \cdot x_B} \bmod q$$

Diffie-Hellman Key Exchange

- K_{AB} is used as session key (or pre-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 (can use nonces and pre-key to make session key different)
- attacker needs a private key x , must solve discrete log base a modulo q to get it

Diffie-Hellman Example

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

Key Exchange Protocols

- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consult and use them to securely communicate with them
- both of these are vulnerable to a Man-in-the-Middle Attack
- authentication of the keys is needed

Man-in-the-Middle Attack

1. Darth prepares by creating two private / public keys
 2. Alice transmits her public key to Bob
 3. Darth intercepts this and transmits his first public key to Bob. Darth also calculates a shared key with Alice
 4. Bob receives the public key and calculates the shared key (with Darth instead of Alice)
 5. Bob transmits his public key to Alice
 6. Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
 7. Alice receives the key and calculates the shared key (with Darth instead of Bob)
- Darth can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

Man-in-the-Middle Attack

$$y_A = a^{x_A} \bmod q$$

$$y'_A = a^{x_{DA}} \bmod q$$

$$y_B = a^{x_B} \bmod q$$

$$y'_B = a^{x_{DB}} \bmod q$$

$$K_{DAB} = {y'_A}^{x_B} \bmod q \quad K_{ADB} = {y'_B}^{x_A} \bmod q$$

Darth has a private, unauthenticated channel with each of Alice and Bob

Man-in-the-Middle Attack

- Also known as “Bucket Brigade” Attack
- Need reliable way to associate public key with principal
- Public key infrastructure (PKI) is one way
- PGP web of trust is another
- In some circumstances, may be possible to use scheduling/timing to prevent MITM

Example

- Alice and Bob get public numbers
 - $P = 23$, $G = 9$
- Alice and Bob compute public values
 - $X = 9^4 \text{ mod } 23 = 6561 \text{ mod } 23 = 6$
 - $Y = 9^3 \text{ mod } 23 = 729 \text{ mod } 23 = 16$
- Alice and Bob exchange public numbers

Example



Alice and Bob compute symmetric keys

$$= 16$$

$$= 6^3$$

now



Applications

- Diffie-Hellman is currently used in many protocols, namely:
 - Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
 - Secure Shell (SSH)
 - Internet Protocol Security (IPSec)
 - Public Key Infrastructure (PKI)

ElGamal Cryptography

- public-key cryptosystem related to D-H
- uses exponentiation in a finite field
- with security based difficulty of computing discrete logarithms, as in D-H
- each user (e.g. A) generates their key
 - chooses a secret key (number): $1 < x_A < q-1$
 - computes their **public key**: $y_A = a^{x_A} \text{ mod } q$

ElGamal Message Exchange

- Bob encrypts a message to send to A computing
 - message M in range $0 \leq M \leq q-1$
 - longer messages must be sent as blocks
 - chose random integer k , $1 \leq k \leq q-1$
 - compute one-time key $K = y_A^k \text{ mod } q$
 - encrypt M as a pair of integers (C_1, C_2) where
 - $C_1 = a^k \text{ mod } q$ // like D-H public key
 - $C_2 = KM \text{ mod } q$ // encrypted msg

ElGamal Message Exchange

- encrypt M as a pair of integers (C_1, C_2) where
 - $C_1 = a^k \text{ mod } q ; C_2 = KM \text{ mod } q$
- A then recovers message by
 - recovering key K as $K = C_1^{x_A} \text{ mod } q$
 - computing M as $M = C_2 K^{-1} \text{ mod } q$
- a unique K must be used each time
 - otherwise result is insecure

ElGamal Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
 - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11, 5)$ by
 - choosing random $k=6$
 - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
 - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11;$
 $C_2 = KM \bmod q = 7 \cdot 17 \bmod 19 = 5$
- Alice recovers original message by computing:
 - recover $K = C_1^{x_A} \bmod q = 11^5 \bmod 19 = 7$
 - compute inverse $K^{-1} = 7^{-1} = 11$
 - recover $M = C_2 K^{-1} \bmod q = 5 \cdot 11 \bmod 19 = 17$

Additional Sources

- <http://www.sans.org/rr/encryption/algoritm.php>
- <http://www.hack.gr/users/dij/crypto/overview/index.html>

Summary

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

Cryptography and Network Security

Module-04
Part-II

Digital Signatures

- have looked at message authentication
 - but does not address issues of lack of trust
- digital signatures provide the ability to:
 - verify author, date & time of signature
 - authenticate message contents
 - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
 - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
 - with new message for existing digital signature
 - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

Arbitrated Digital Signatures

- involves use of arbiter A
 - validates any signed message
 - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- may be one-way or mutual
- key issues are
 - confidentiality – to protect session keys
 - timeliness – to prevent replay attacks

Replay Attacks

- where a valid signed message is copied and later resent
 - simple replay
 - repetition that can be logged
 - repetition that cannot be detected
 - backward replay without modification
- countermeasures include
 - use of sequence numbers (generally impractical)
 - timestamps (needs synchronized clocks)
 - challenge/response (using unique nonce)

Using Symmetric Encryption

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
 - each party shares own master key with KDC
 - KDC generates session keys used for connections between parties
 - master keys used to distribute these to them

Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:
 1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks||ID_A]]$
 3. A→B: $E_{Kb}[Ks||ID_A]$
 4. B→A: $E_{Ks}[N_2]$
 5. A→B: $E_{Ks}[f(N_2)]$

Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
 - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
 - timestamps (Denning 81)
 - using an extra nonce (Neuman 93)

Using Public-Key Encryption

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning 81 presented the following:
 1. $A \rightarrow AS: ID_A \parallel ID_B$
 2. $AS \rightarrow A: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T]$
 3. $A \rightarrow B: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T] \parallel E_{KUb}[E_{KRas}[K_s \parallel T]]$
- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces, vis:
 1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks||ID_A]]$
 3. A→B: $E_{Kb}[Ks||ID_A] \parallel E_{Ks}[M]$
- does not protect against replays
 - could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:
 $A \rightarrow B: E_{KUb}[Ks] \parallel E_{Ks}[M]$
 - has encrypted session key, encrypted message
- if authentication needed use a digital signature with a digital certificate:
 $A \rightarrow B: M \parallel E_{KRa}[H(M)] \parallel E_{KRas}[T \parallel ID_A \parallel KU_a]$
 - with message, signature, certificate

Digital Signature Standard (DSS)

- US Govt approved signature scheme FIPS 186
- uses the SHA hash algorithm
- designed by NIST & NSA in early 90's
- DSS is the standard, DSA is the algorithm
- a variant on ElGamal and Schnorr schemes
- creates a 320 bit signature, but with 512-1024 bit security
- security depends on difficulty of computing discrete logarithms

DSA Key Generation

- have shared global public key values (p, q, g):
 - a large prime $p = 2^L$
 - where $L = 512$ to 1024 bits and is a multiple of 64
 - choose q , a 160 bit prime factor of $p-1$
 - choose $g = h^{(p-1)/q}$
 - where $h < p-1$, $h^{(p-1)/q} \pmod{p} > 1$
- users choose private & compute public key:
 - choose $x < q$
 - compute $y = g^x \pmod{p}$

DSA Signature Creation

- to **sign** a message M the sender:
 - generates a random signature key k , $k < q$
 - nb. k must be random, be destroyed after use, and never be reused
- then computes signature pair:
$$r = (g^k \pmod p) \pmod q$$
$$s = (k^{-1} \cdot \text{SHA}(M) + x \cdot r) \pmod q$$
- sends signature (r, s) with message M

DSA Signature Verification

- having received M & signature (r, s)
- to **verify** a signature, recipient computes:

$$w = s^{-1} \pmod{q}$$

$$u1 = (\text{SHA}(M) \cdot w) \pmod{q}$$

$$u2 = (r \cdot w) \pmod{q}$$

$$v = (g^{u1} \cdot y^{u2} \pmod{p}) \pmod{q}$$

- if $v=r$ then signature is verified
- see book web site for details of proof why

Summary

- have considered:
 - digital signatures
 - authentication protocols (mutual & one-way)
 - digital signature standard

Module-04

Cryptographic and hash Functions

By

Dr.S.P.Anandaraj



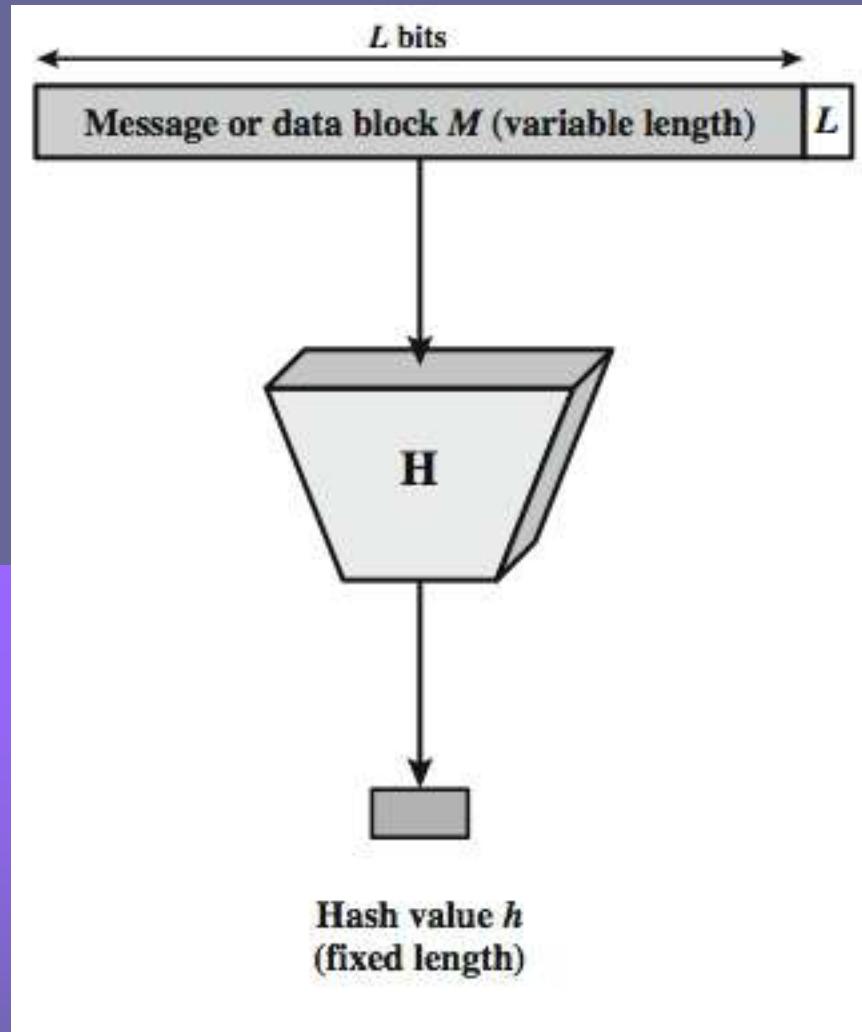
Hash Functions

- condenses arbitrary message to fixed size

$$h = H(M)$$

- usually assume hash function is public
- hash used to detect changes to message
- want a cryptographic hash function
 - computationally infeasible to find data mapping to specific hash (one-way property)
 - computationally infeasible to find two data to same hash (collision-free property)

Cryptographic Hash Function



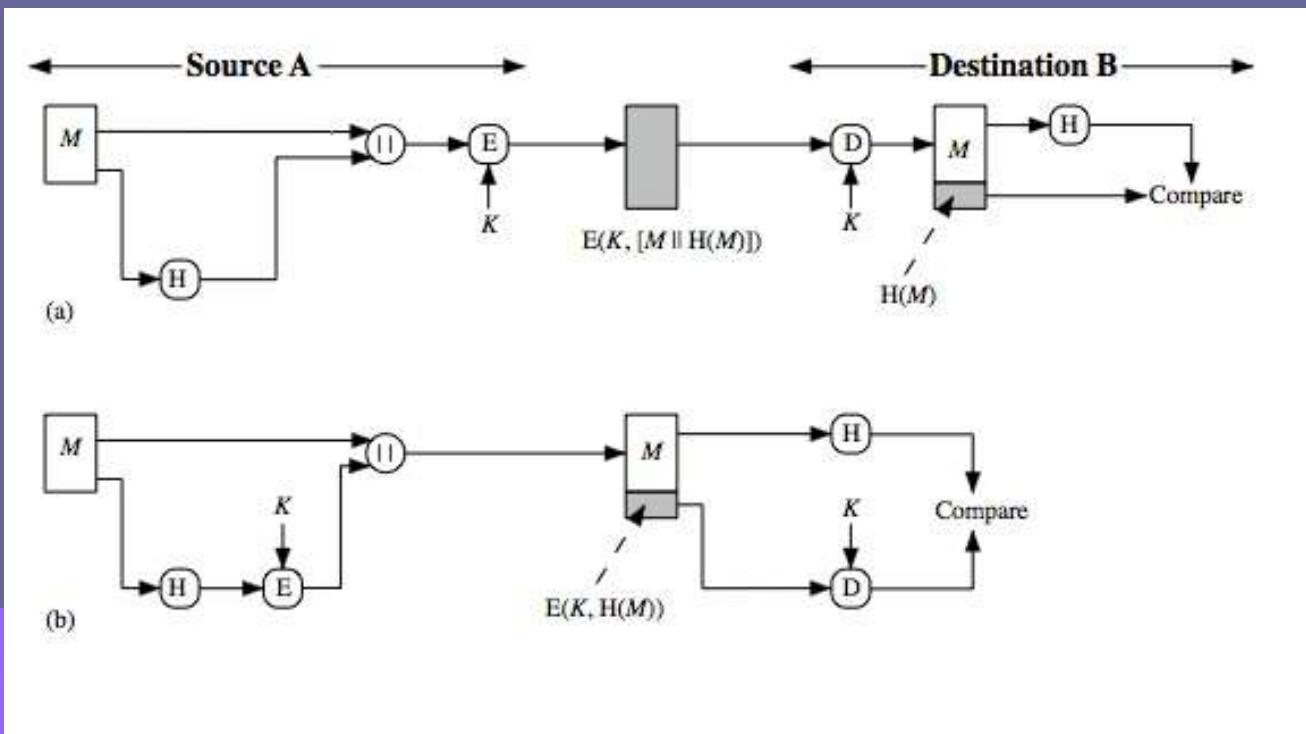
Hash Function Uses

- Message Integrity Check (MIC)
 - send hash of message (digest)
 - MIC always encrypted, message optionally
- Message Authentication Code (MAC)
 - send keyed hash of message
 - MAC, message optionally encrypted
- Digital Signature (non-repudiation)
 - Encrypt hash with private (signing) key
 - Verify with public (verification) key

Hash Functions & Message Authentication

Symmetric Key
Unkeyed Hash

a) Message encrypted



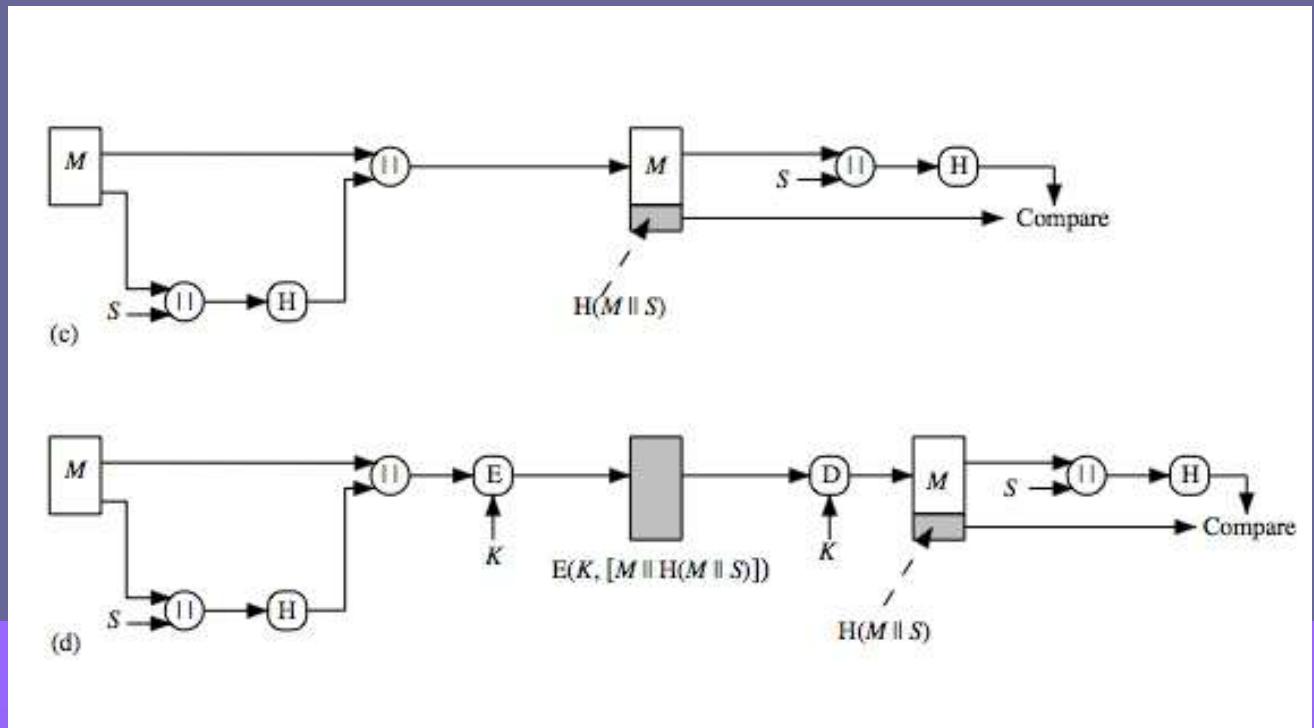
b) Message
unencrypted

Hash Functions & Message Authentication

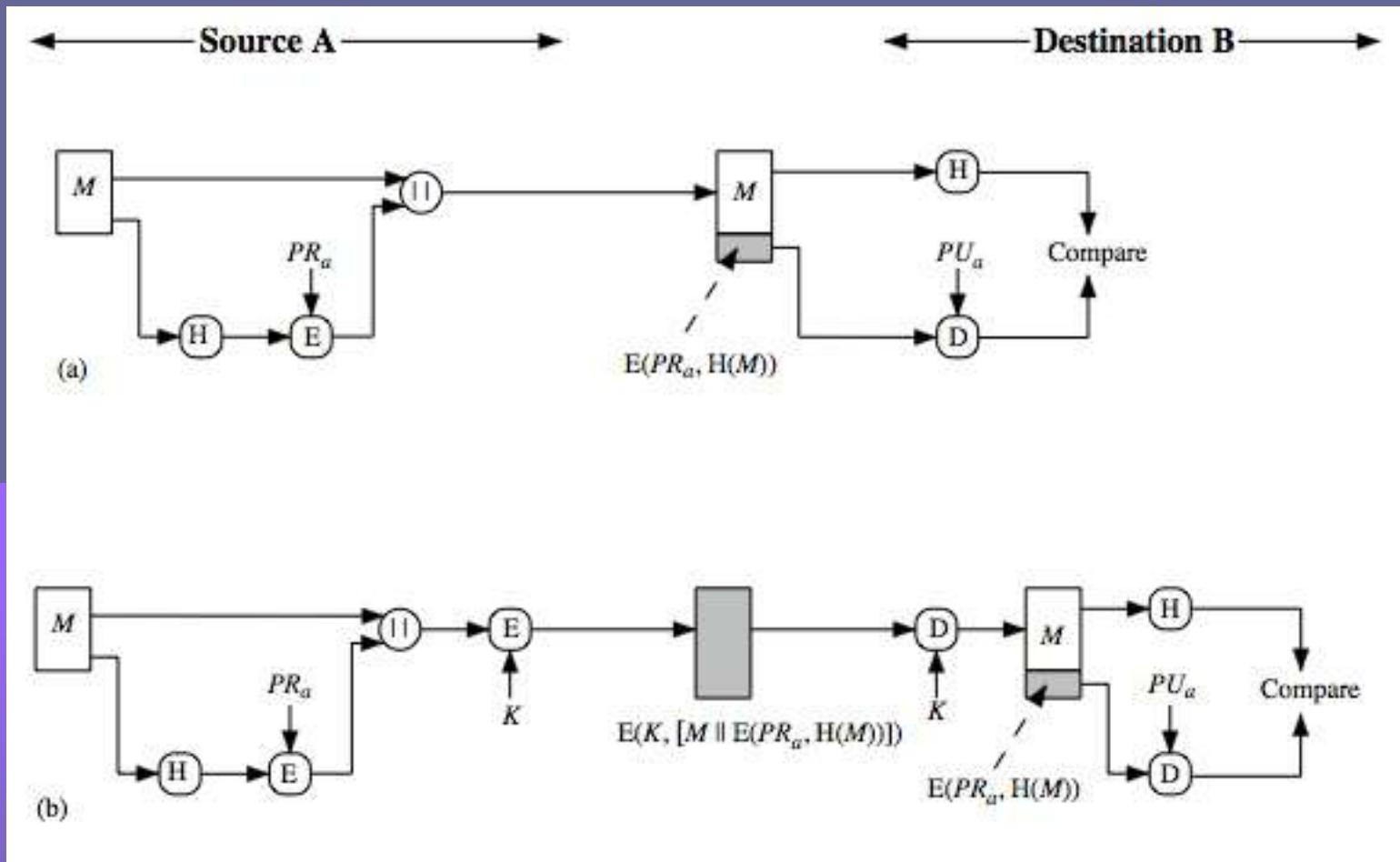
Symmetric Key
Keyed Hash

a) Message
unencrypted

d) Message
encrypted



Hash Functions & Digital Signatures - PKCS



Other Hash Function Uses

- pseudorandom function (PRF)
 - Generate session keys, nonces
 - Produce key from password
 - Derive keys from master key cooperatively
- pseudorandom number generator (PRNG)
 - Vernam Cipher/OTP
 - S/Key, proof of “what you have” via messages

More Hash Function Uses

- to create a one-way password file
 - store hash of password not actual password
 - e.g., Unix, Windows NT, etc.
 - salt to deter precomputation attacks
 - Rainbow tables
- for intrusion detection and virus detection
 - keep & check hash of files on system
 - e.g., Tripwire

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
 - using $H_0=0$ and zero-pad of final block
 - compute: $H_i = E_{M_i}[H_{i-1}]$
 - and use final block as the hash value
 - similar to CBC but without a key
- resulting hash is too small (64-bit)
 - both due to direct birthday attack
 - and to “meet-in-the-middle” attack
- other variants also susceptible to attack

Secure Hash Algorithm

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

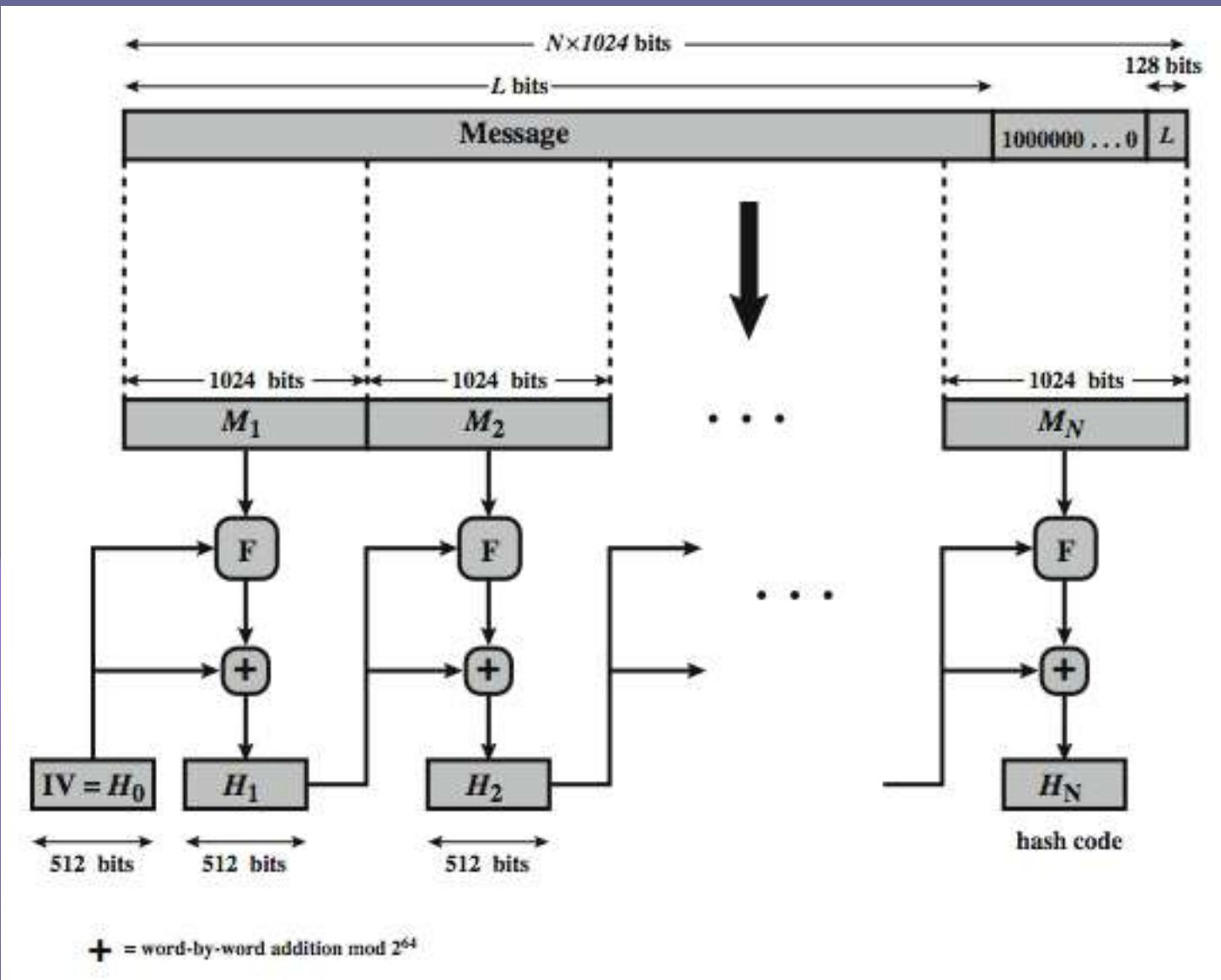
Revised Secure Hash Standard

- 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

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

SHA-512 Overview



SHA-512 (1024 bit message) Algorithm Framework

- ✓ Step 1: Append Padding Bits....

Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits fewer than an even multiple of 1024.

- ✓ Step 2: Append Length....

128 bits are appended to the end of the padded message. These bits hold the binary format of 128 bits indicating the length of the original message.

SHA-1 Framework Continued

✓ Step 3: Prepare Processing Functions....

SHA1 requires 80 processing functions defined as:

$$\begin{aligned} f(t;B,C,D) &= (B \text{ AND } C) \text{ OR } ((\text{NOT } B) \text{ AND } D) \quad (0 \leq t \leq 19) & f(t;B,C,D) &= B \text{ XOR } C \\ &\text{XOR } D \quad (20 \leq t \leq 39) & f(t;B,C,D) &= (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \\ &(40 \leq t \leq 59) & f(t;B,C,D) &= B \text{ XOR } C \text{ XOR } D \quad (60 \leq t \leq 79) \end{aligned}$$

✓ Step 4: Prepare Processing Constants....

SHA1 requires 80 processing constant words defined as:

$K(t) = 0x5A827999$	$(0 \leq t \leq 19)$
$K(t) = 0x6ED9EBA1$	$(20 \leq t \leq 39)$
$K(t) = 0x8F1BBCDC$	$(40 \leq t \leq 59)$
$K(t) = 0xCA62C1D6$	$(60 \leq t \leq 79)$

SHA-1 Framework Continued

✓ Step 5: Initialize Buffers....

SHA1 requires 160 bits or 5 buffers of words (32 bits):

H0 = 0x67452301 H1 = 0xEFCDAB89 H2 = 0x98BADCFE H3 =
0x10325476 H4 = 0xC3D2E1F0



SHA-1 Framework

Final Step

- ✓ Step 6: Processing Message in 1024-bit blocks (L blocks in total message)....

This is the main task of SHA1 algorithm which loops through the padded and appended message in 512-bit blocks.

Input and predefined functions:

$M[1, 2, \dots, L]$: Blocks of the padded and appended message $K(0)$,

$f(0;B,C,D)$, $f(1,B,C,D)$, ..., $f(79,B,C,D)$: 80 Processing Functions $K(1), \dots, K(79)$: 80 Processing Constant Words

✓ $H_0, H_1, H_2, H_3, H_4, H_5$: 5 Word buffers with initial values



SHA-1 Framework Continued

v Step 6: Pseudo Code....

For loop on k = 1 to L

(W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */

For t = 16 to 79 do:

 W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1

 A = H0, B = H1, C = H2, D = H3, E = H4

 For t = 0 to 79 do:

 TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t) E = D, D = C, C = B<<<30,

 B = A, A = TEMP

 End of for loop

 H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E

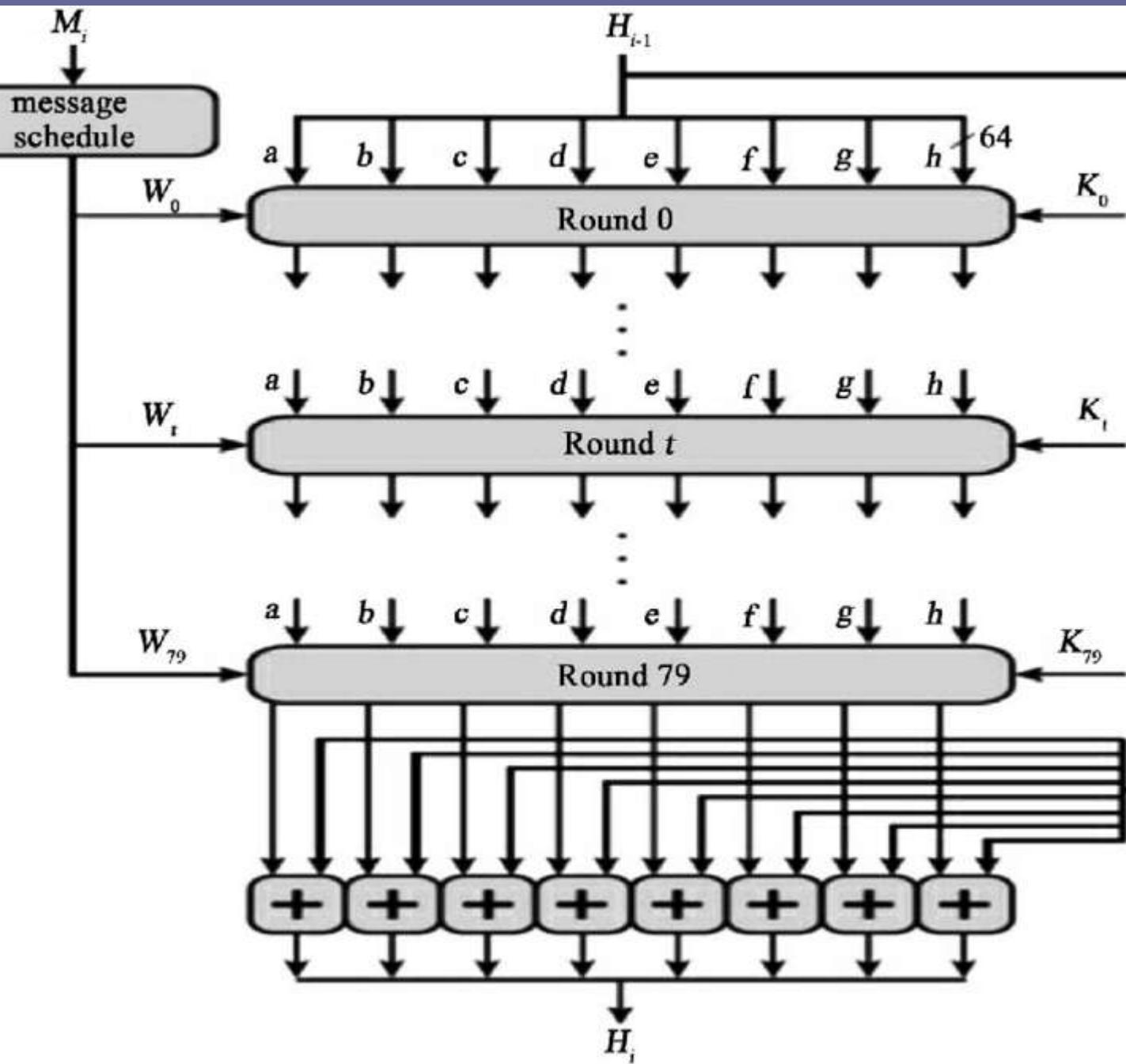
End of for loop

Output:

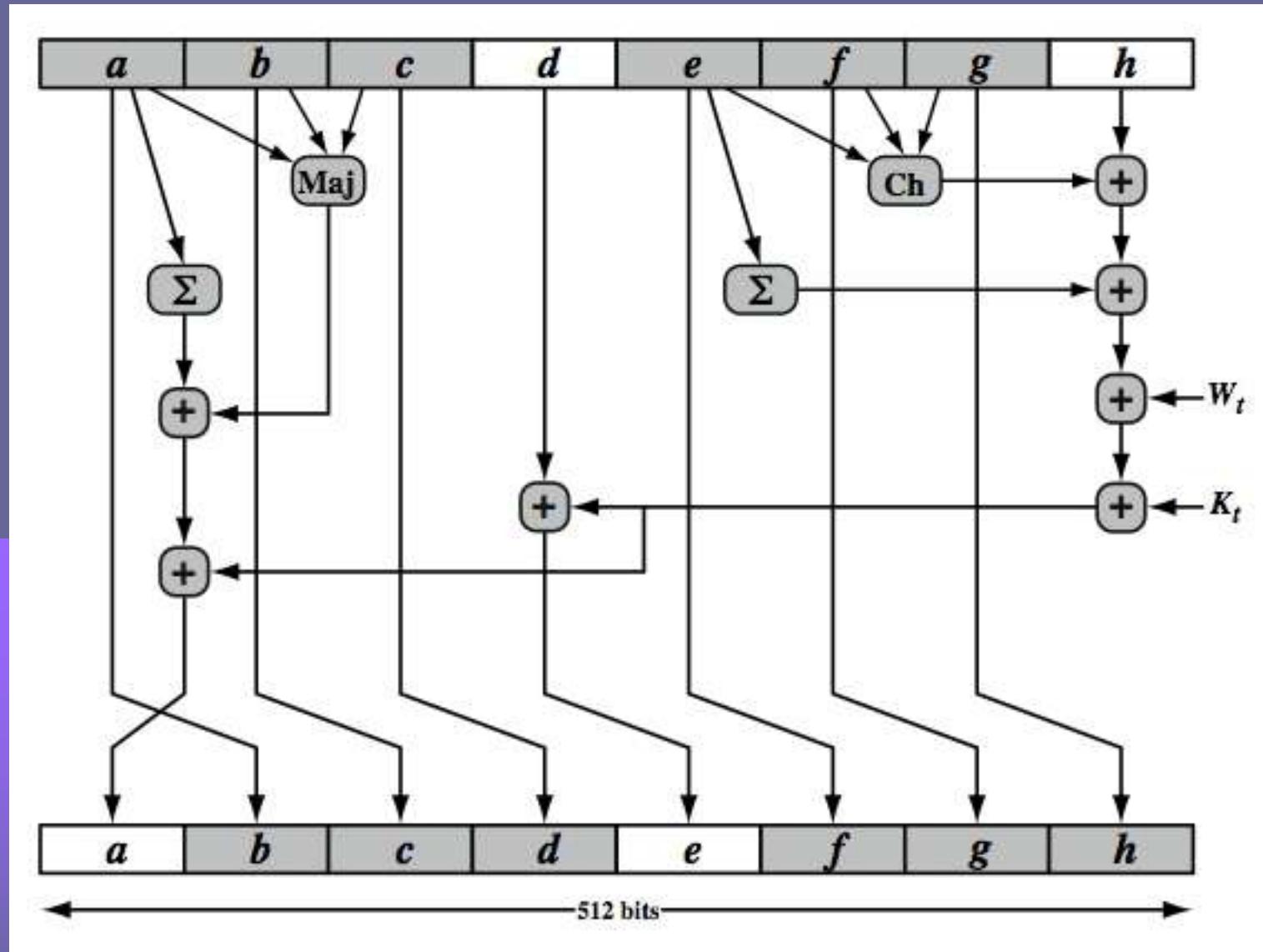
H0, H1, H2, H3, H4, H5: Word buffers with final message digest

SHA-512 Compression Function

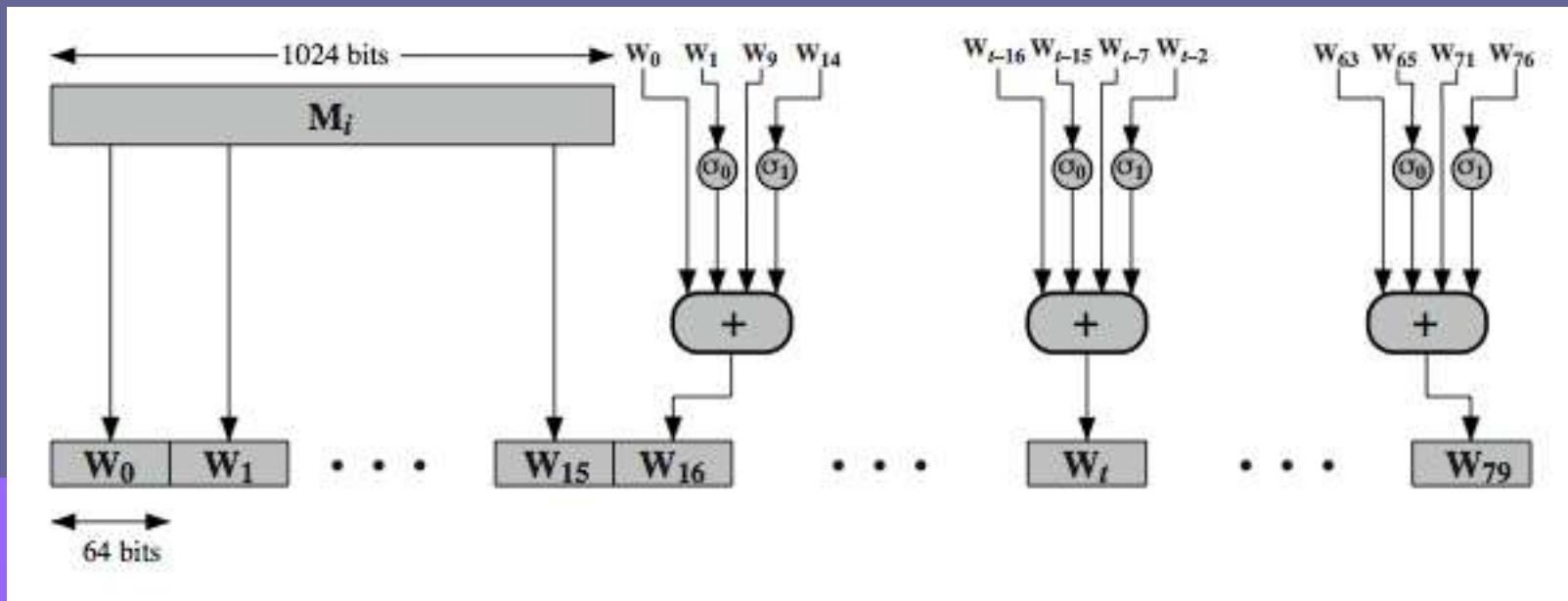
- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
 - updating a 512-bit buffer
 - using a 64-bit value W_t derived from the current message block
 - and a round constant based on cube root of first 80 prime numbers



SHA-512 Round Function



SHA-512 Round Function



SHA-3

- SHA-1 not yet "broken"
 - but similar to broken MD5 & SHA-0
 - so considered insecure
- SHA-2 (esp. SHA-512) seems secure
 - shares same structure and mathematical operations as predecessors so have concern
- NIST announced in 2007 a competition for the SHA-3 next gen NIST hash function
- Keccak winner Oct 2012 – std in Q2,2014

SHA-3 Requirements

- replace SHA-2 with SHA-3 in any use
 - so use same hash sizes
- preserve the online 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

Message Authentication

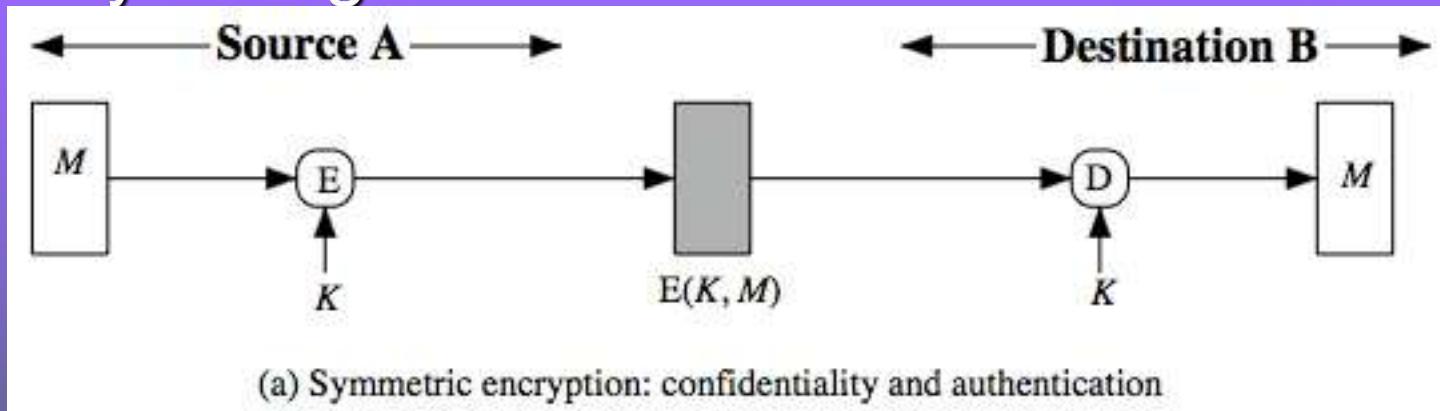
- message authentication is concerned with:
 - protecting the integrity of a message
 - validating identity of originator
 - non-repudiation of origin (dispute resolution)
- will consider the security requirements
- then three alternative functions used:
 - hash function (see Ch 11)
 - message encryption
 - message authentication code (MAC)(see ch12)

Message Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

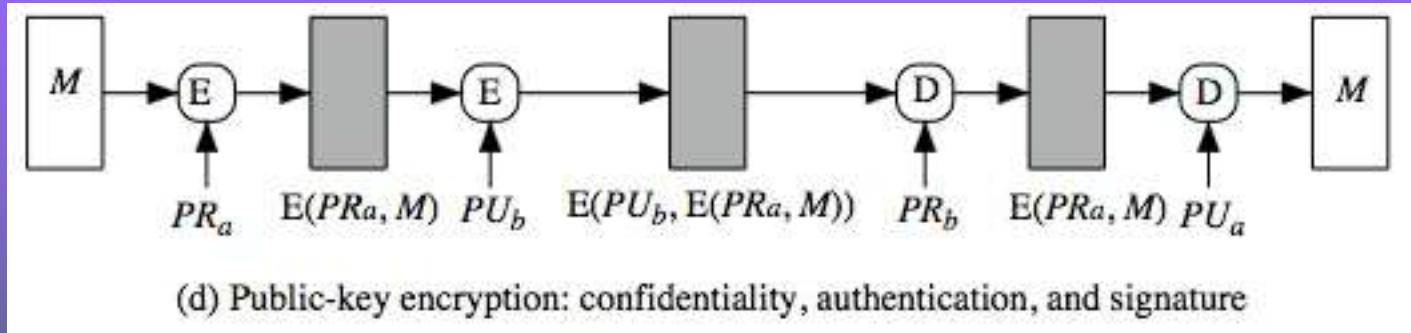
Symmetric Message Encryption

- encryption can also provides authentication
- if symmetric encryption is used then:
 - receiver know sender must have created it
 - since only sender and receiver know key used
 - know content cannot have been altered...
 - ... if message has suitable structure, redundancy or a suitable checksum to detect any changes



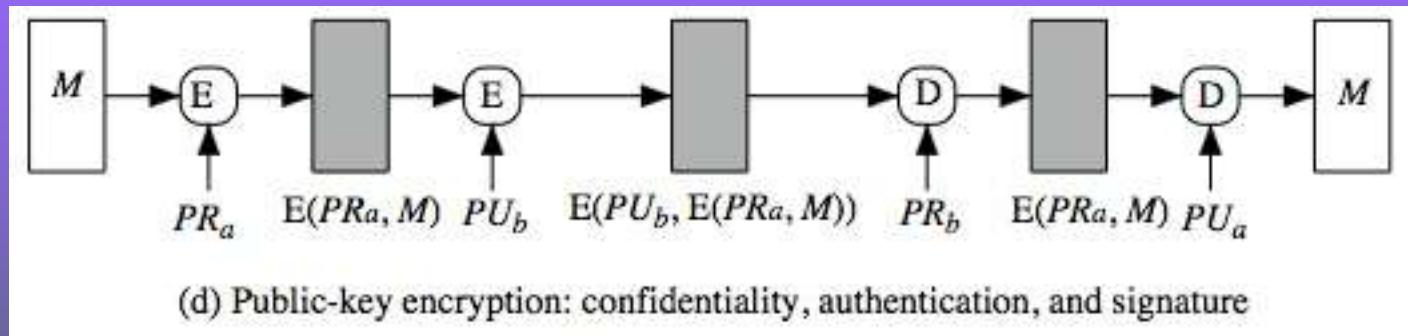
Public-Key Message Encryption

- if public-key encryption is used:
 - encryption provides no confidence of sender
 - since anyone potentially knows public-key
 - however if
 - sender **signs** message using their private-key
 - then encrypts with recipients public key
 - have both secrecy and authentication
 - again need to recognize corrupted messages
 - but at cost of two public-key uses on message



Public-Key Message Encryption

- Dirty little detail on PKCS
 - Every time you encrypt, size expands
 - Due to protections in PKCS#1
- So signing (by encryption) then encrypting, the size is more than doubled!

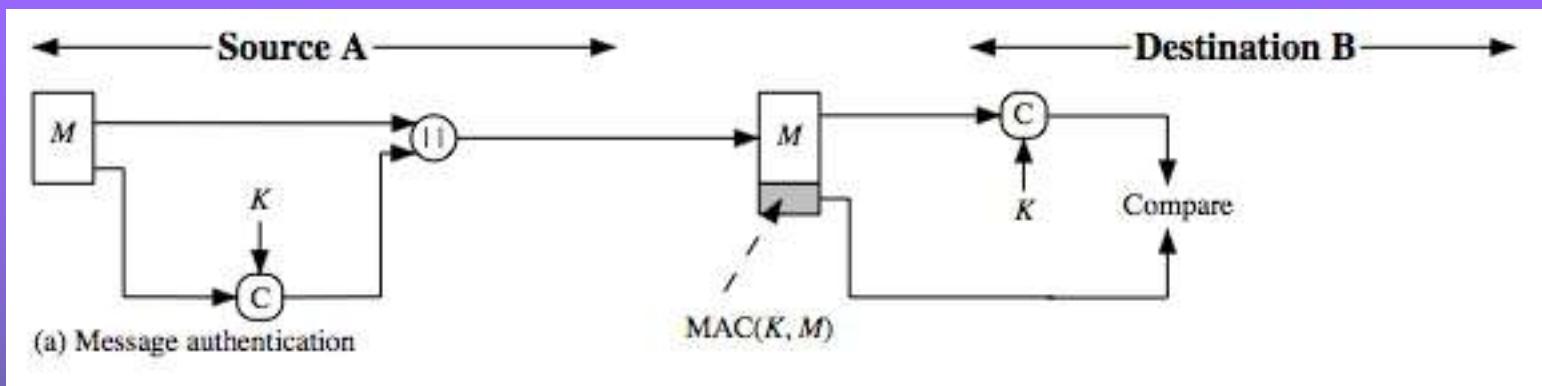


Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
 - depending on both message and secret key
 - like encryption though need not be reversible
- appended to message as a “signature”
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender

Message Authentication Code

- a small fixed-sized block of data
- generated from message + secret key
- $\text{MAC} = \text{C}(K, M)$
- appended to message when sent



Message Authentication Codes

- as shown the MAC provides **authentication**
- can also use encryption for secrecy
 - generally use **separate keys** for each
 - can compute MAC either before or after encryption
 - is generally regarded as better done before, but see Generic Composition

Message Authentication Codes

- why use a MAC?
 - sometimes only authentication is needed
 - sometimes need authentication to persist longer than the encryption (e.g. archival use)
- note that a MAC is **not a digital signature**
 - Does NOT provide non-repudiation

MAC Properties

- a MAC is a cryptographic checksum

$$\text{MAC} = C_K(M)$$

- condenses a variable-length message M
- using a secret key K
- to a fixed-sized authenticator

- is a many-to-one function

- potentially many messages have same MAC
- but finding these needs to be very difficult

Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
 1. knowing a message and MAC, is infeasible to find another message with same MAC
 2. MACs should be uniformly distributed
 3. MAC should depend equally on all bits of the message

Security of MACs

- like block ciphers have:
- **brute-force** attacks exploiting
 - strong collision resistance hash have cost $2^{m/2}$
 - 128-bit hash looks vulnerable, 160-bits better
 - MACs with known message-MAC pairs
 - can either attack keyspace (cf. key search) or MAC
 - at least 128-bit MAC is needed for security

Security of MACs

- **cryptanalytic attacks** exploit structure
 - like block ciphers want brute-force attacks to be the best alternative
- more variety of MACs so harder to generalize about cryptanalysis

HMAC Design Objectives

- use, without modifications, hash functions
- allow for easy replacement of embedded hash function
- preserve original performance of hash function without significant degradation
- use and handle keys in a simple way.
- have well understood cryptographic analysis of authentication mechanism strength

HMAC

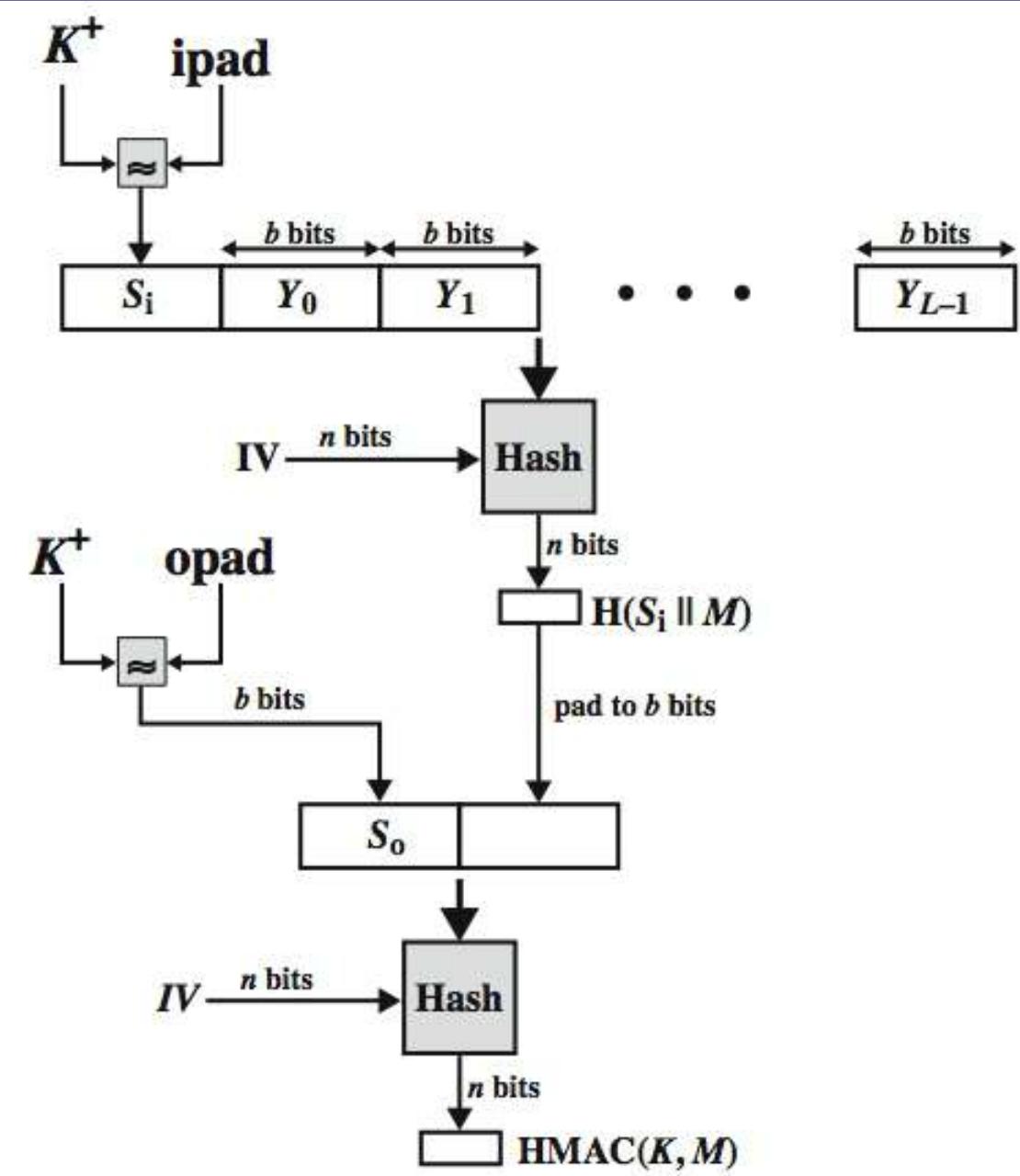
- specified as Internet standard RFC2104
- uses hash function on the message:

$$\text{HMAC}_K(M) = \text{Hash} [(K^+ \text{ XOR } opad) \parallel \\ \text{Hash} [(K^+ \text{ XOR } ipad) \parallel M]]$$

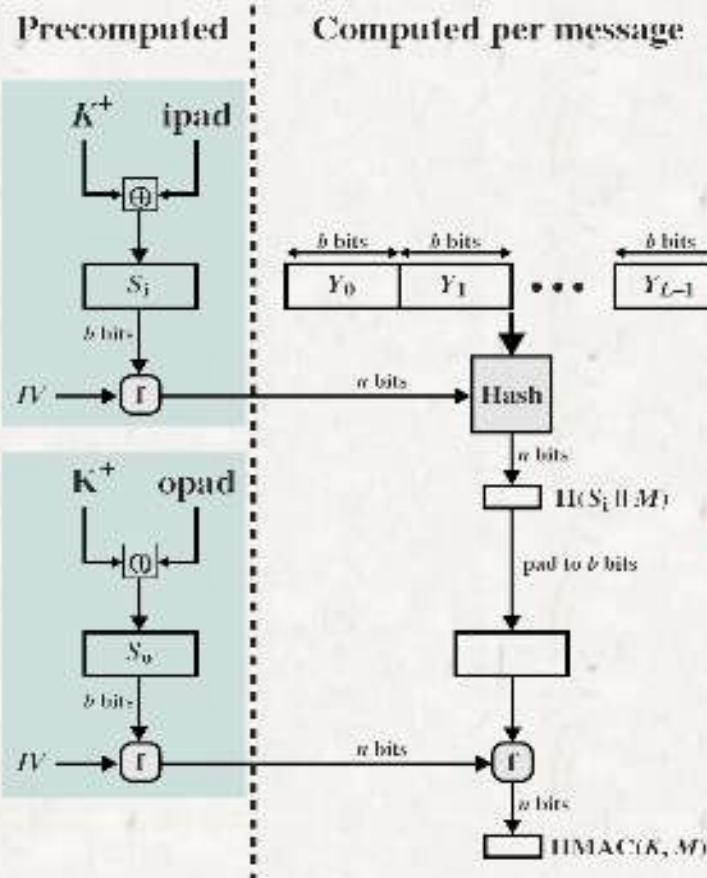
- where K^+ is the key padded out to block size
- $opad$, $ipad$ are specified padding constants

- overhead is just 3 more hash block calculations than the message needs alone
- any hash function can be used
 - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

HMAC Overview



HMAC



- **HMAC should execute in approximately the same time as the embedded hash function**

- for a long message.
- HMAC adds 3 executions of the hash compression function.

- **A more efficient implement is possible by precomputing**

$$f(IV, (K^+ \oplus \text{ipad}))$$

$$f(IV, (K^+ \oplus \text{opad}))$$

HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
 - brute force attack on key used
 - birthday attack (but since keyed would need to observe a very large number of messages)
- choose hash function used based on speed versus security constraints

Digital Signatures

have looked at message authentication
but does not address issues of lack of trust

digital signatures provide the ability to:

verify author, date & time of signature

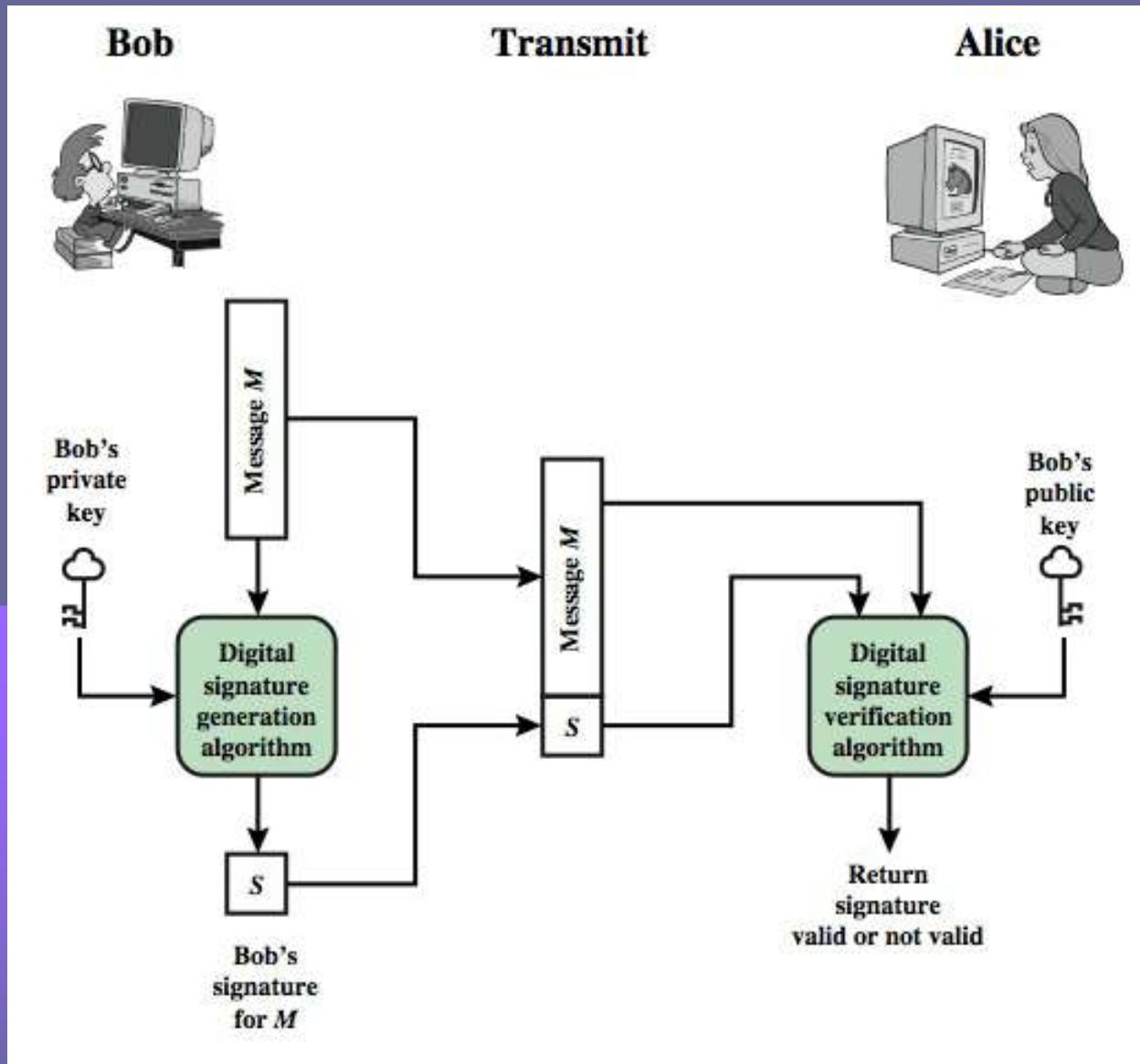
authenticate message contents

be verified by third parties to resolve disputes

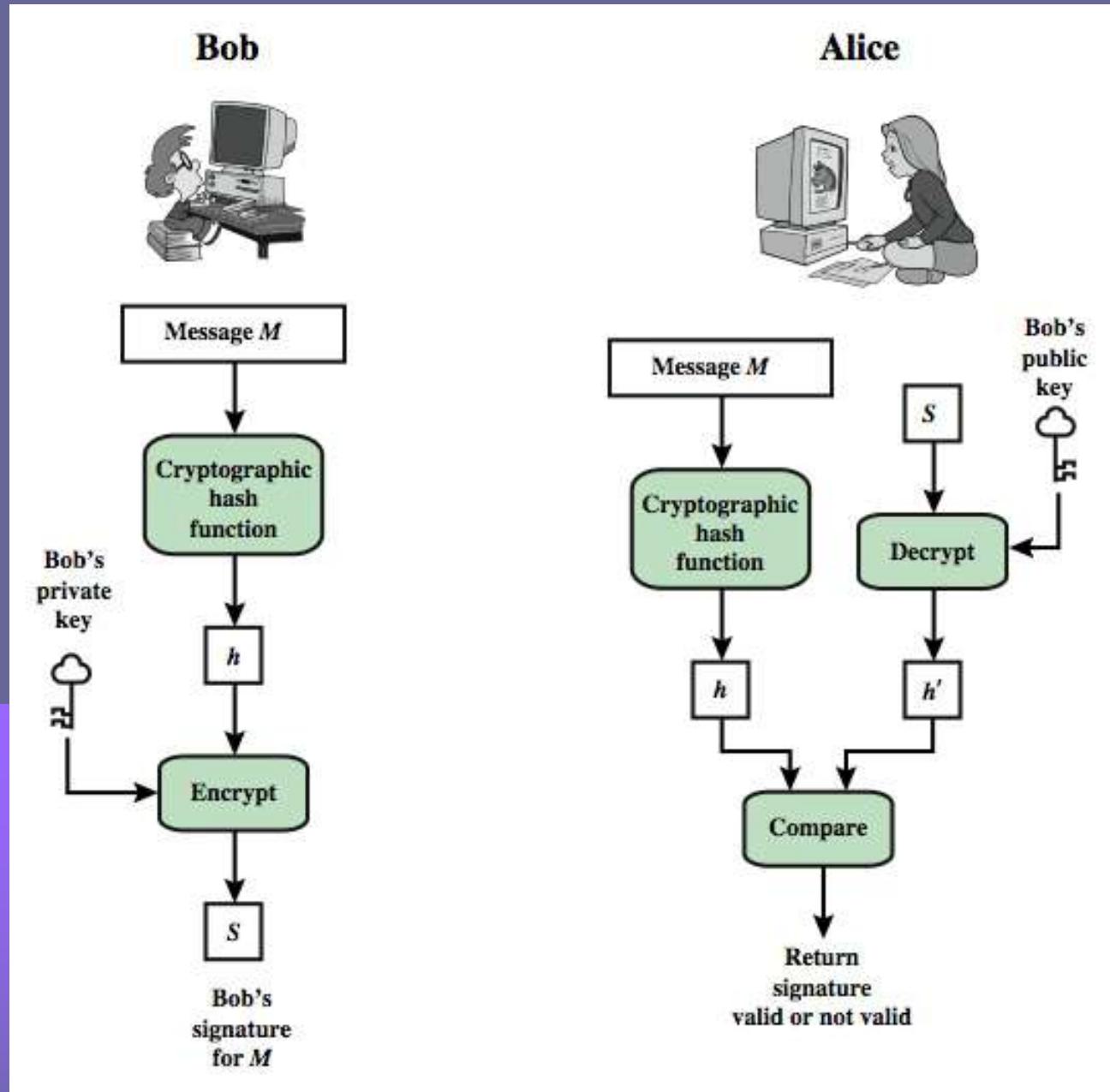
hence include authentication function with
additional capabilities



Digital Signature Model



Digital Signature Model



Attacks and Forgeries

attacks

key-only attack

known message attack

generic chosen message attack

directed chosen message attack

adaptive chosen message attack

break success levels

total break

selective forgery

existential forgery

Digital Signature Requirements

- must depend on the message signed
- must use information unique to sender
 - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
 - with new message for existing digital signature
 - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

involve only sender & receiver
assumed receiver has sender's public-key
digital signature made by sender signing
entire message or hash with private-key
can encrypt using receivers public-key
important that sign first then encrypt
message & signature
security depends on sender's private-key

Cryptography and Network Security

Module-05

Authentication Applications

- will consider authentication functions
- developed to support application-level authentication & digital signatures
- will consider Kerberos – a private-key authentication service
- then X.509 directory authentication service

Kerberos

- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
 - allows users access to services distributed through network
 - without needing to trust all workstations
 - rather all trust a central authentication server
- two versions in use: 4 & 5

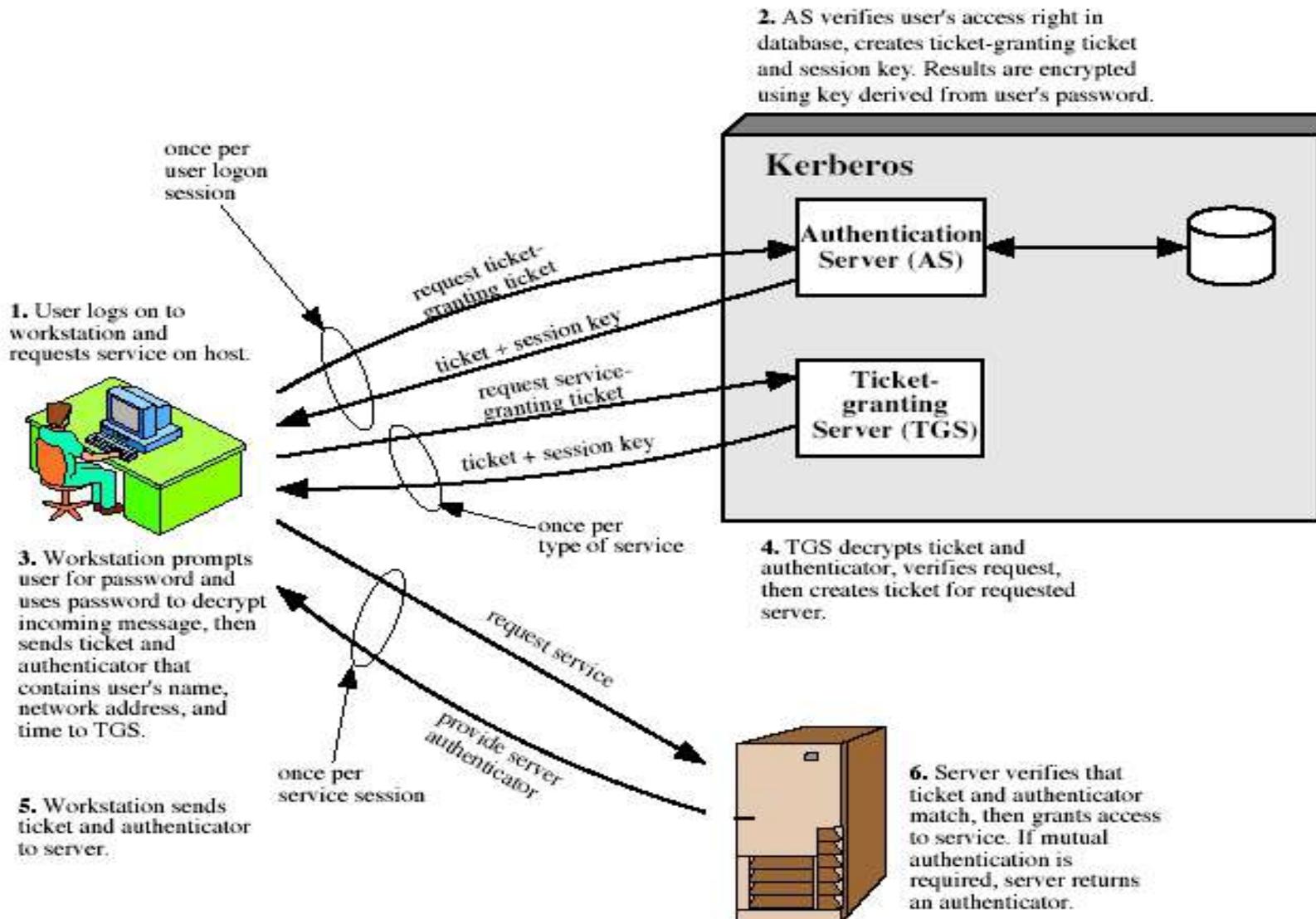
Kerberos Requirements

- first published report identified its requirements as:
 - security
 - reliability
 - transparency
 - scalability
- implemented using an authentication protocol based on Needham-Schroeder

Kerberos 4 Overview

- a basic third-party authentication scheme
- have an Authentication Server (AS)
 - users initially negotiate with AS to identify self
 - AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
 - users subsequently request access to other services from TGS on basis of users TGT

Kerberos 4 Overview



Kerberos Realms

- a Kerberos environment consists of:
 - a Kerberos server
 - a number of clients, all registered with server
 - application servers, sharing keys with server
- this is termed a realm
 - typically a single administrative domain
- if have multiple realms, their Kerberos servers must share keys and trust

Kerberos Version 5

- developed in mid 1990's
- provides improvements over v4
 - addresses environmental shortcomings
 - encryption alg, network protocol, byte order, ticket lifetime, authentication forwarding, interrealm auth
 - and technical deficiencies
 - double encryption, non-std mode of use, session keys, password attacks
- specified as Internet standard RFC 1510

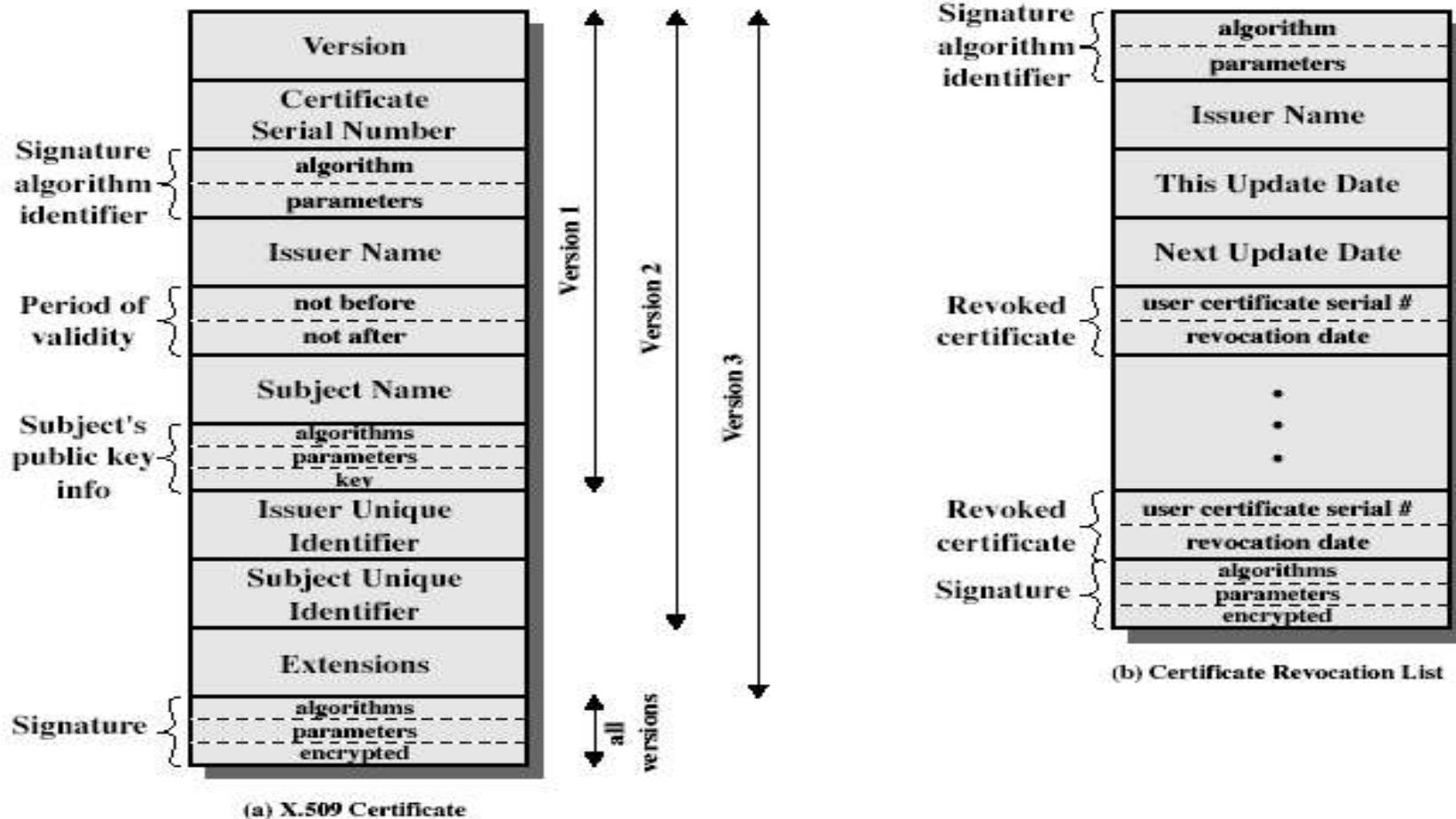
X.509 Authentication Service

- part of CCITT X.500 directory service standards
 - distributed servers maintaining some info database
- defines framework for authentication services
 - directory may store public-key certificates
 - with public key of user
 - signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
 - algorithms not standardised, but RSA recommended

X.509 Certificates

- issued by a Certification Authority (CA), containing:
 - version (1, 2, or 3)
 - serial number (unique within CA) identifying certificate
 - signature algorithm identifier
 - issuer X.500 name (CA)
 - period of validity (from - to dates)
 - subject X.500 name (name of owner)
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)
 - signature (of hash of all fields in certificate)
- notation CA<<A>> denotes certificate for A signed by CA

X.509 Certificates



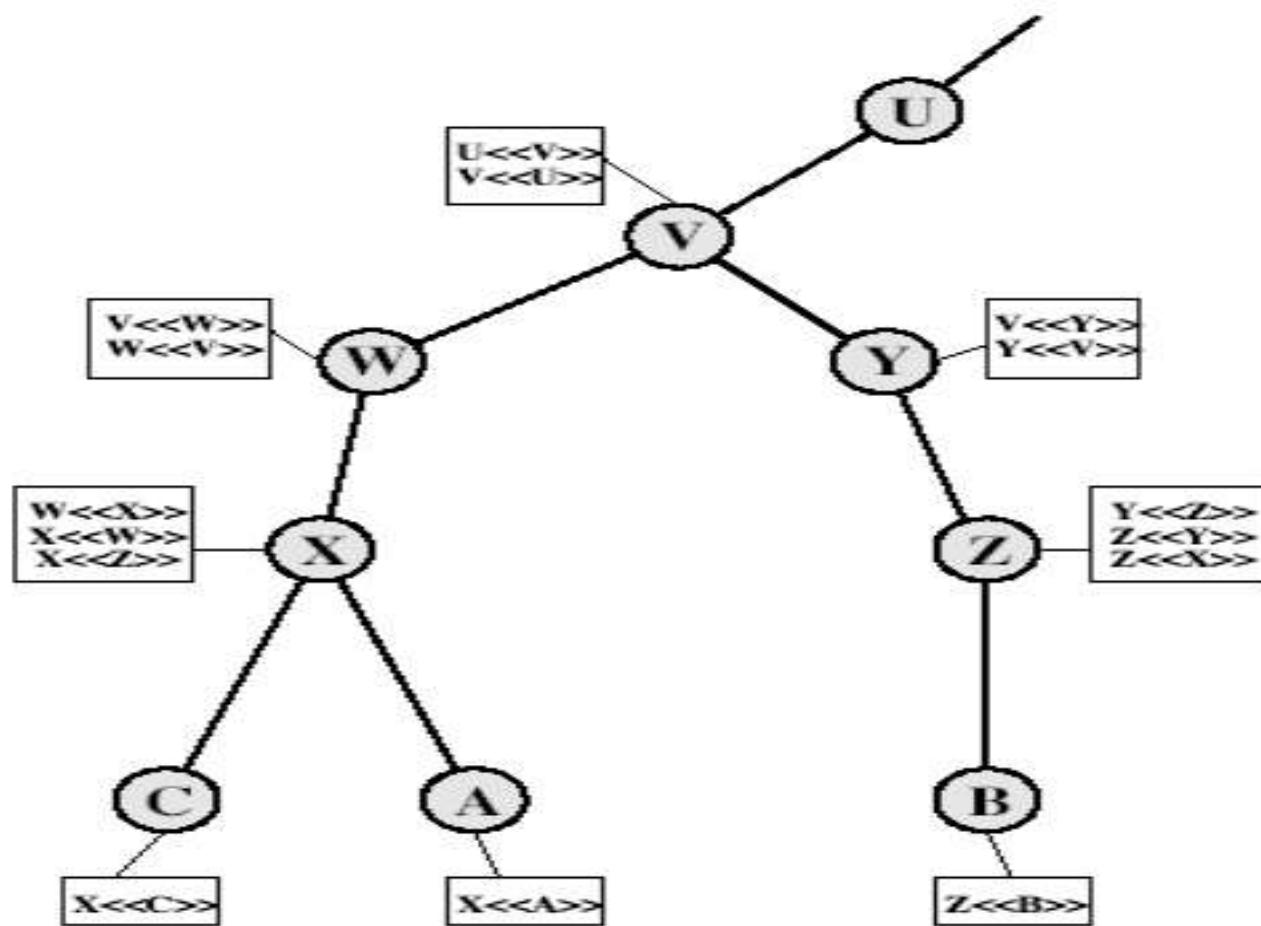
Obtaining a Certificate

- any user with access to CA can get any certificate from it
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy
- use certificates linking members of hierarchy to validate other CA's
 - each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

CA Hierarchy Use



Certificate Revocation

- certificates have a period of validity
- may need to revoke before expiry, eg:
 1. user's private key is compromised
 2. user is no longer certified by this CA
 3. CA's certificate is compromised
- CA's maintain list of revoked certificates
 - the Certificate Revocation List (CRL)
- users should check certs with CA's CRL

Authentication Procedures

- X.509 includes three alternative authentication procedures:
- One-Way Authentication
- Two-Way Authentication
- Three-Way Authentication
- all use public-key signatures

One-Way Authentication

- 1 message (A->B) used to establish
 - the identity of A and that message is from A
 - message was intended for B
 - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A

Two-Way Authentication

- 2 messages ($A \rightarrow B$, $B \rightarrow A$) which also establishes in addition:
 - the identity of B and that reply is from B
 - that reply is intended for A
 - integrity & originality of reply
- reply includes original nonce from A, also timestamp and nonce from B

Three-Way Authentication

- 3 messages ($A \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$) which enables above authentication without synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

X.509 Version 3

- has been recognised that additional information is needed in a certificate
 - email/URL, policy details, usage constraints
- rather than explicitly naming new fields defined a general extension method
- extensions consist of:
 - extension identifier
 - criticality indicator
 - extension value

Certificate Extensions

- key and policy information
 - convey info about subject & issuer keys, plus indicators of certificate policy
- certificate subject and issuer attributes
 - support alternative names, in alternative formats for certificate subject and/or issuer
- certificate path constraints
 - allow constraints on use of certificates by other CA's

Summary

- have considered:
 - Kerberos trusted key server system
 - X.509 authentication and certificates

Cryptography and Network Security

E-mail-Security, PGP and S/MIME

Email Security

- email is one of the most widely used and regarded network services
- currently message contents are not secure
 - may be inspected either in transit
 - or by suitably privileged users on destination system

E-mail Components

- Message User Agents(MUA)
- Message Transfer Agents(MTA)
- Message Handling Service(MHS)

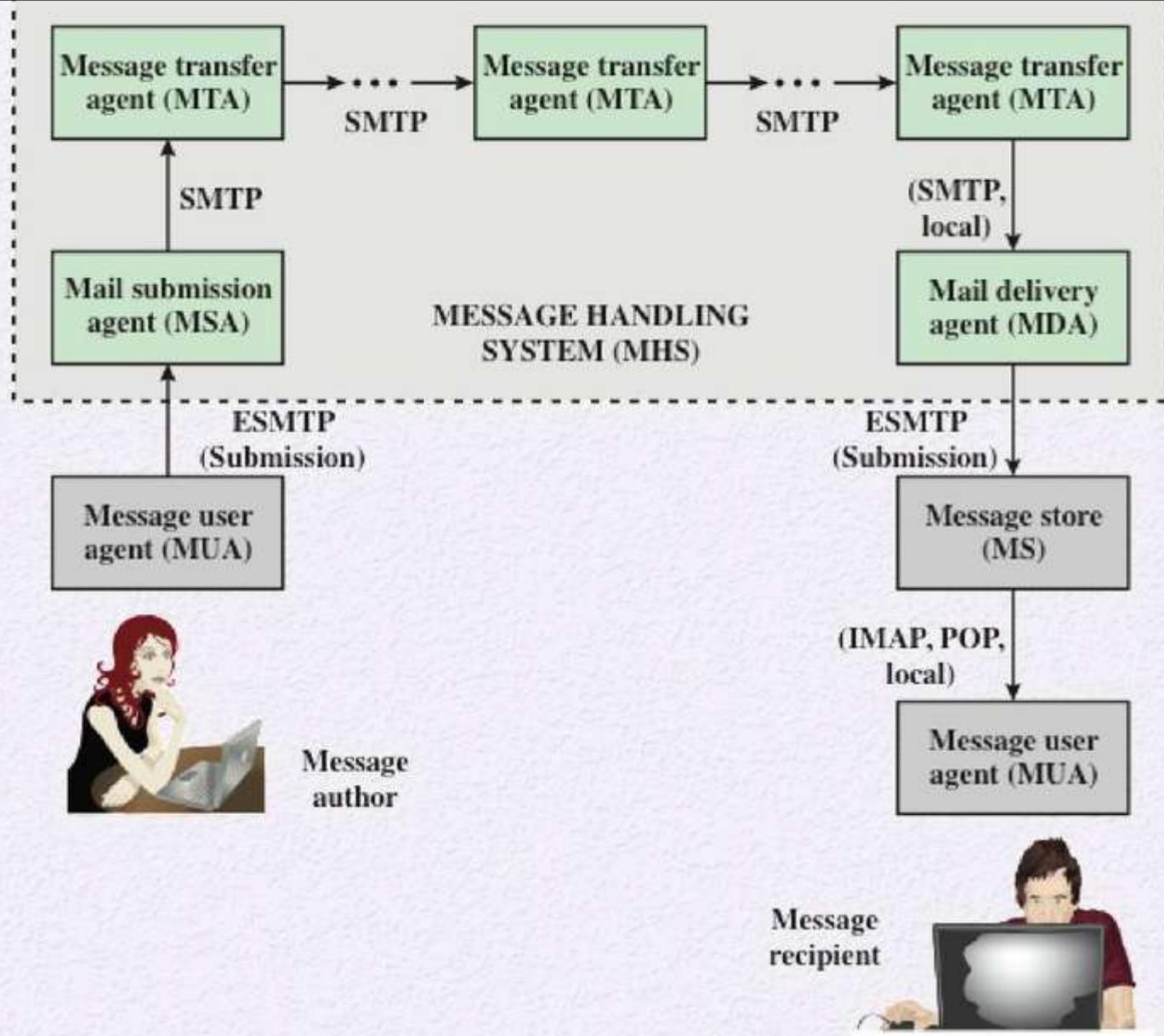


Figure 19.1 Function Modules and Standardized Protocols Used Between Them in the Internet Mail Architecture

Key Components:

- Message User Agent(MUA)
- Mail Submission Agent(MSA)
- Message Transfer Agent(MTA)
- Mail Delivery Agent(MDA)
- Message Store(MS)

E-mail Protocols:

- Two types of protocols are used for transferring email:
 - SMTP(Move message through the Internet from source to destination).
 - IMAP,POP(Transfer messages between mail servers)

Example SMTP Transaction Scenario

```
mail from: sender@domain.com
250 2.1.0 Ok
rcpt to: recipient@domain.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
From: "Sender" <sender@domain.com>
To: "Recipient" <recipient@domain.com>
Subject: Message Sample
```

Hello, this is a message sample.

Thanks.

```
250 2.0.0 Ok: queued as AF290244C4
```

E-mail formats

- RFC 5322
- MIME(Multipurpose Internet Mail Extensions)-Five Header fields:
 - MIME-Version
 - Content-type
 - Content transfer Encoding
 - Content-ID
 - Content-Description

MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted 7-bit ASCII text; no transformation by MIME is needed
	HTML	HTML format
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Body contains no-ordered parts of different data types
	Digest	Body contains ordered parts of different data types, but the default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

GIF = Graphics Interchange Format; HTML = Hypertext Markup Language; JPEG = Joint Photographic Experts Group; MPEG = Motion Picture Experts Group.

MIME Transfer Encoding

Type	Description
7-bit	The body contains The 7-bit ASCII Characters With maximum length of 1000 characters
8-bit	There can be non-ASCII 8-bit characters but the maximum length of the body is limited to 1000 characters.
Binary	Binary 8-bit characters without limitation of 1000 characters in the body.
Quoted-printable	This is useful when data consists of largely printable characters. Characters in the rang decimal equivalent 33 to 61 in ASCII are represented in ASCII. Others are represented as two-digit hex representation preceded by '=' sign, Non-text characters are replaced with six-digit hex sequence
Base 64	6-bit block of input data is encoded into 8-bit block of output.

Email Security Enhancements

- confidentiality
 - protection from disclosure
- authentication
 - of sender of message
- message integrity
 - protection from modification
- non-repudiation of origin
 - protection from denial by sender

S.N.	Service and Description
1.	Gmail Gmail is an email service that allows users to collect all the messages. It also offers approx 7 GB of free storage.
2.	Hotmail Hotmail offers free email and practically unlimited storage accessible on web.
3.	Yahoo Mail Yahoo Mail offers unlimited storage, SMS texting, social networking and instant messaging to boot.
4.	iCloud Mail iCloud Mail offers ample storage, IMAP access, and an elegantly functional web application.
5.	ATM Mail ATM Mail is a free email service with good spam protection.
6.	Mail.com and GMX Mail Mail.com and GMX Mail offers reliable mail service with unlimited online storage.
7.	Shortmail Shortmail offers easy and fast email service but with limited 500 characters per message.
8.	Inbox.com Inbox.com offers 5 GB of free online storage. IMAP is not supported by Inbox.com
9.	Facebook Messages Facebook Messages includes the message conversation.
10.	My Way Mail My Way Mail offers clean and fast free email service but lacks in secure messaging.

Pretty Good Privacy (PGP)

- widely used de facto secure email
- developed by Phil Zimmermann
- selected best available crypto algs to use
- integrated into a single program
- available on Unix, PC, Macintosh and Amiga systems
- originally free, now have commercial versions available also

Pretty Good Privacy-PGP

- PGP is a remarkable phenomenon that provides confidentiality, authentication, and compression for email and data storage.
- Its building blocks are made of the best available cryptographic algorithms: RSA, DSS, Diffie-Hellman.
- It is independent of operating system and processor.
- It has a small set of easy-to-use commands

- available on Unix, Windows, Macintosh and Amiga systems
- originally free, now have commercial versions available also

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

Notations

- K_s = session key used in Symmetric encryption scheme
- KR_a = private key of User A (public key encryption)
- KU_a = public key of User A (public key encryption)
- EP = public key encryption
- DP = public key decryption
- EC = symmetric encryption
- DC = symmetric decryption
- H = hash function
- \parallel = concatenation
- Z = compression
- R64 = conversion to radix 64 ASCII format

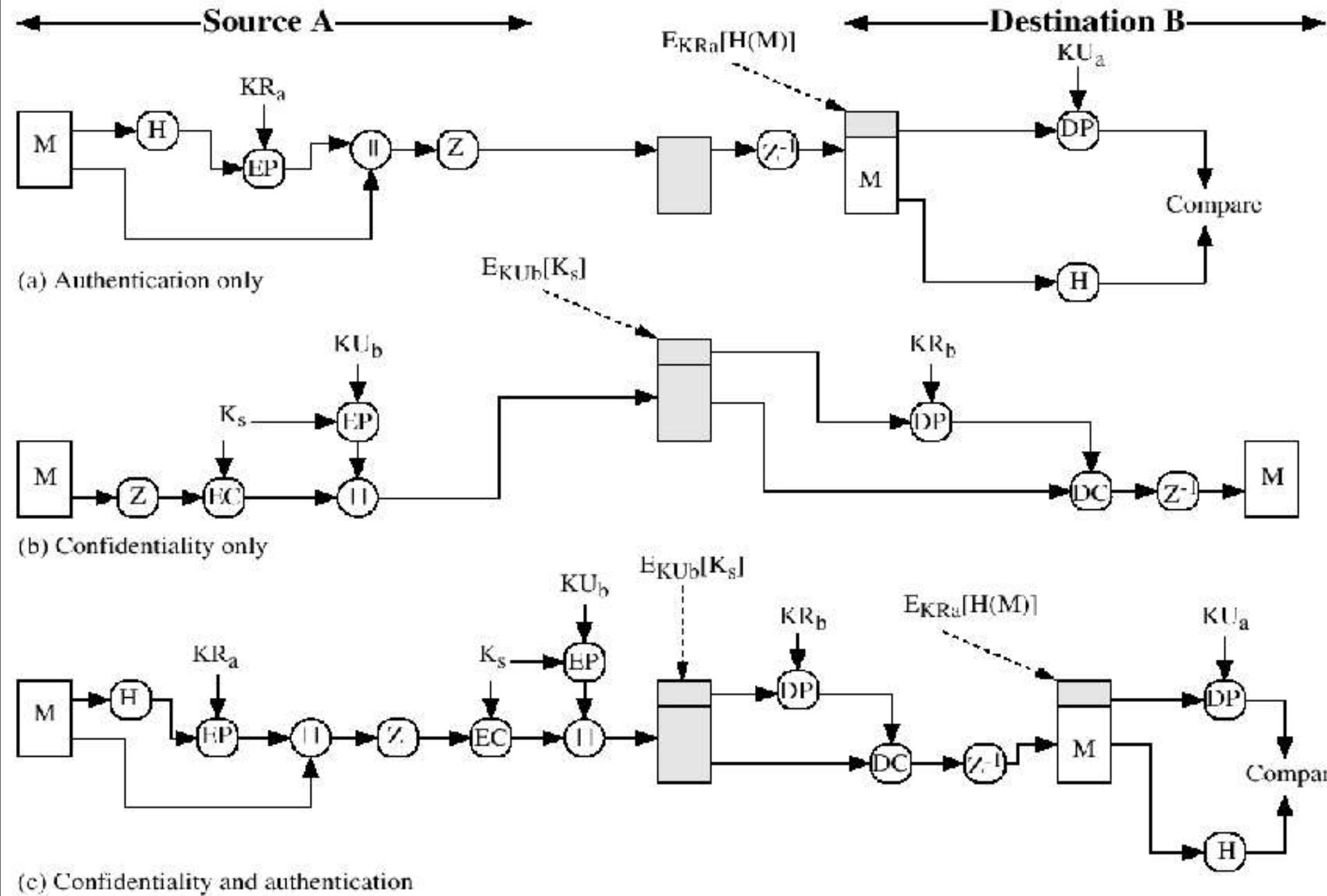


Figure 5.1 PGP Cryptographic Functions

PGP Operation – Authentication

1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

PGP Operation – Authentication

1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

PGP Operation – Confidentiality

1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message

PGP Operation – Confidentiality & Authentication

- uses both services on same message
 - create signature & attach to message
 - encrypt both message & signature
 - attach RSA encrypted session key

PGP Operation – Compression

- by default PGP compresses message after signing but before encrypting
 - so can store uncompressed message & signature for later verification
 - & because compression is non deterministic
- Msg encryption done after-strengthen security
- The placement of the compression algorithm is critical.
- uses ZIP compression algorithm

PGP Operation – Email Compatibility

- when using PGP will have binary data to send (encrypted message etc)
- however email was designed only for text
- hence PGP must encode raw binary data into printable ASCII characters
- The scheme uses radix-64 algorithm
 - maps 3 bytes to 4 printable chars
 - also appends a CRC

E-mail Compatibility

- The use of radix-64 expands the message by 33%.

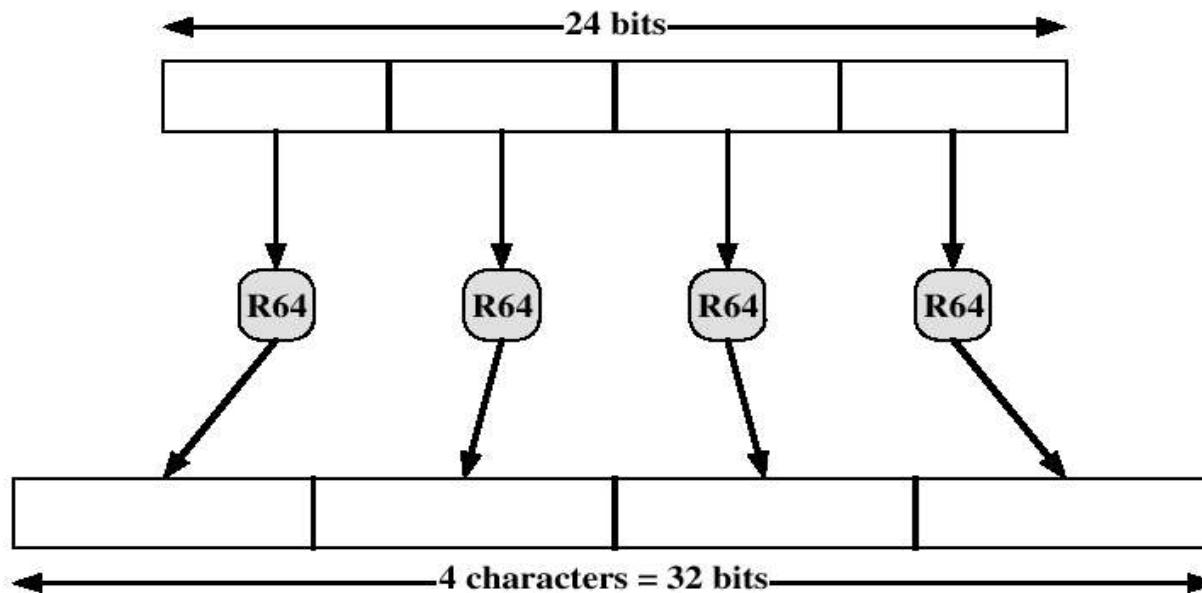


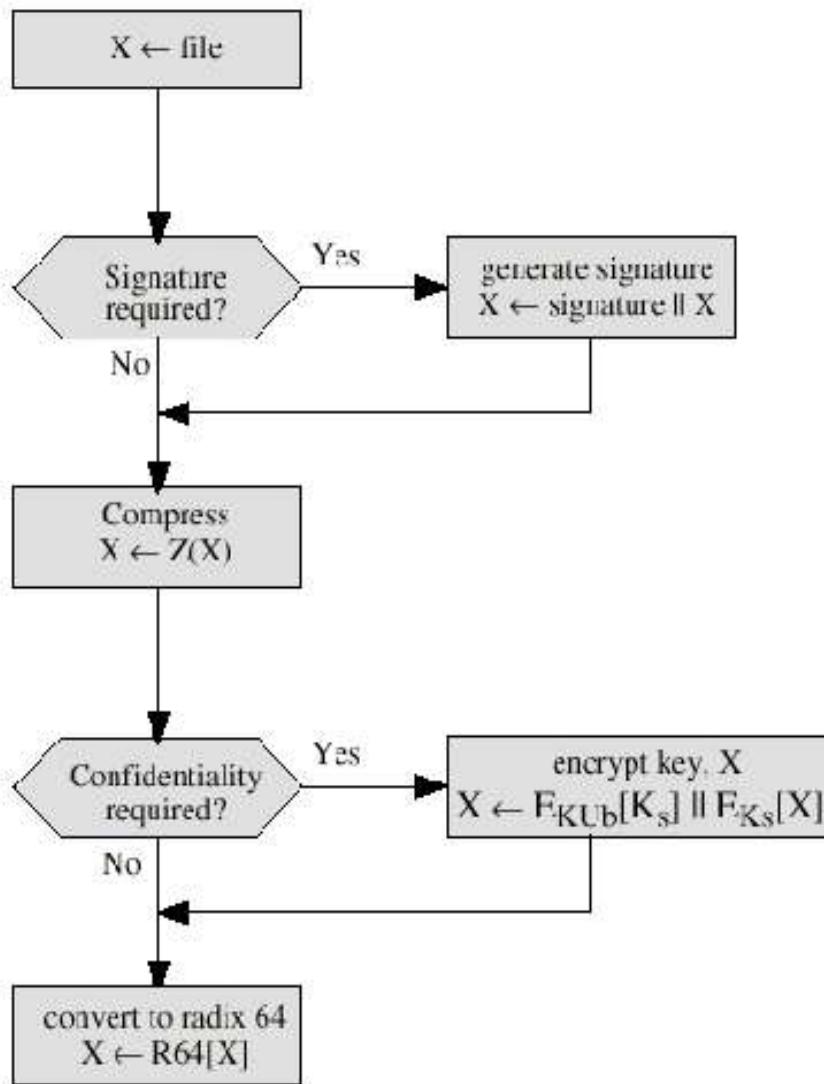
Figure 5.11 Printable Encoding of Binary Data into Radix-64 Format

Segmentation and Reassembly

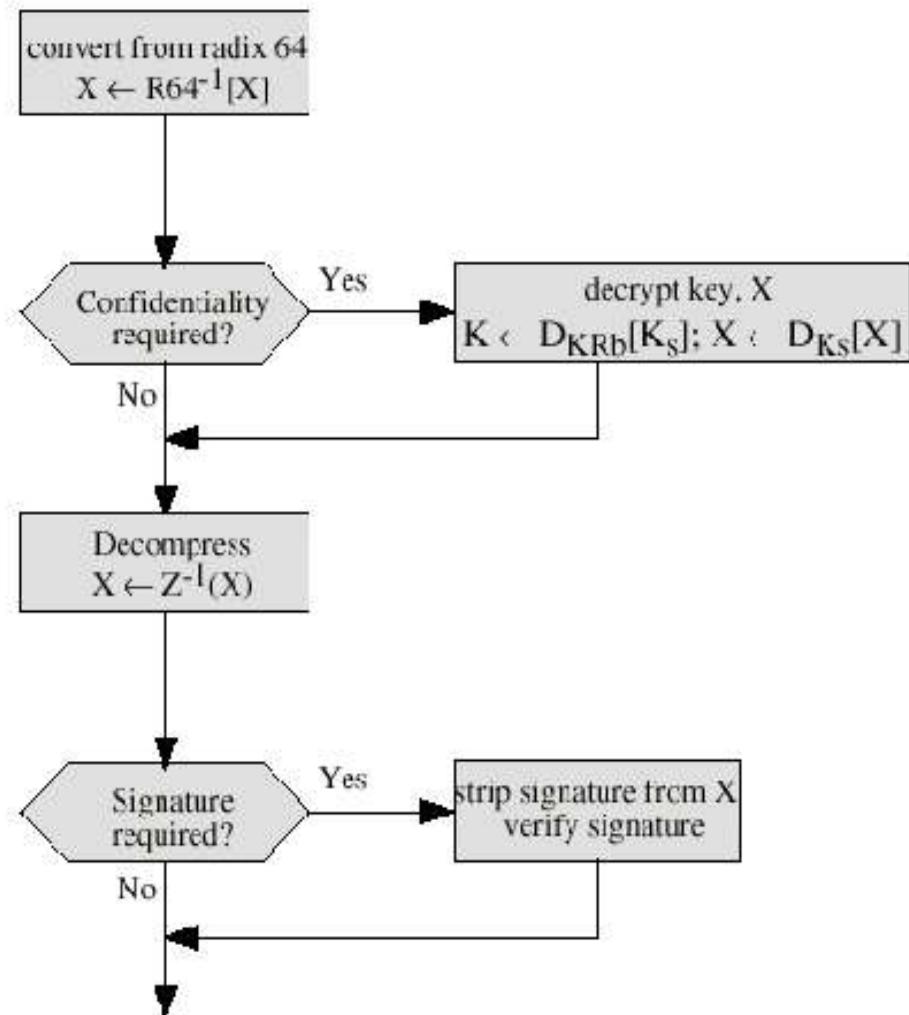
- E-mail facilities are often restricted to a maximum message length of 50,000 octets.
- Longer messages must be broken up into segments.
- PGP automatically subdivides a message that is too large.
- The receiver strips off all e-mail headers and reassemble the block.

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

PGP Session Keys

- need a session key for each message
 - of varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES
- generated using ANSI X12.17 mode
- uses random inputs taken from previous uses and from keystroke timing of user

PGP Public & Private Keys

- since many public/private keys may be in use, need to identify which is actually used to encrypt session key in a message
 - could send full public-key with every message
 - but this is inefficient
- rather use a key identifier based on key
 - is least significant 64-bits of the key
 - Key ID of K_{Ua} public key is $K_{Ua} \bmod 2^{64}$
 - will very likely be unique
- also use key ID in signatures

Content

Operation

Session key component

Key ID of recipient's public key (KU_b)

Session key (K_s)

E_{KUb}

Timestamp

Key ID of sender's public key (KU_a)

Leading two octets of message digest

Message Digest

E_{KRa}

Filename

Timestamp

ZIP

E_{Ks}

Message

Data

R64

Format of PGP Message

PGP Key Rings

- each PGP user has a pair of keyrings:
 - public-key ring contains all the public-keys of other PGP users known to this user, indexed by key ID
 - private-key ring contains the public/private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$KU_i \bmod 2^{64}$	KU_i	$EH(P_i)[KR_i]$	User i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$KU_i \bmod 2^{64}$	KU_i	trust_flag i	User i	trust_flag i		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = field used to index table

Figure 5.4 General Structure of Private and Public Key Rings

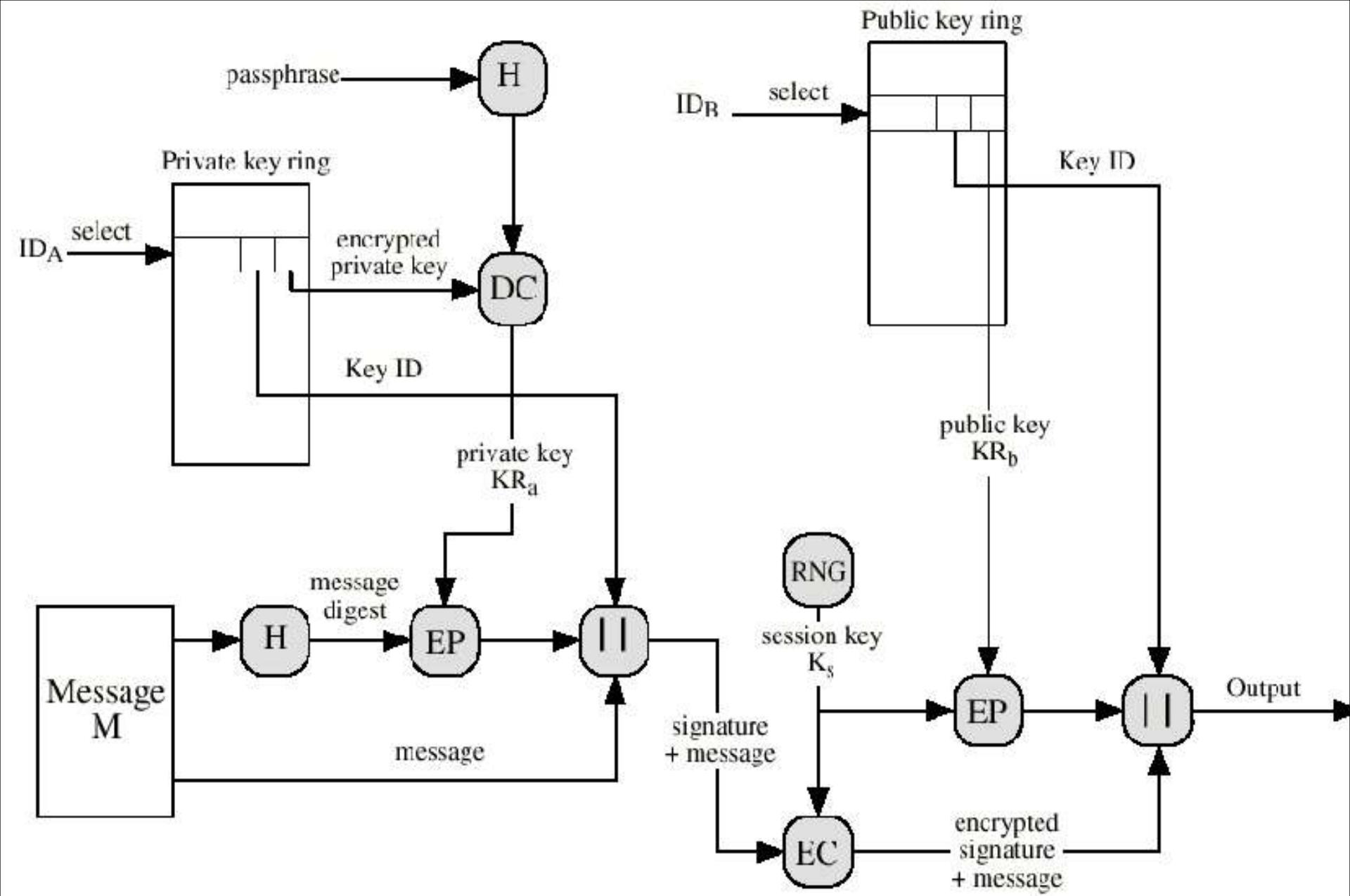


Figure 5.5 PGP Message Generation (from User A to User B; no compression or radix 64 conversion)

S/MIME (Secure/Multipurpose Internet Mail Extensions)

- security enhancement to MIME email
 - original Internet RFC822 email was text only
 - MIME provided support for varying content types and multi-part messages
 - with encoding of binary data to textual form
 - S/MIME added security enhancements
- have S/MIME support in various modern mail agents: MS Outlook, Netscape etc

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 (76 characters)

Header fields in MIME

- **MIME-Version:** Must be “1.0” -> RFC 2045, RFC 2046
- **Content-Type:** Describes the data contained in the body . 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:** Text description of the object with the body. Needed when content is not readable text (e.g.,audio data)

S/MIME: Signed Mail

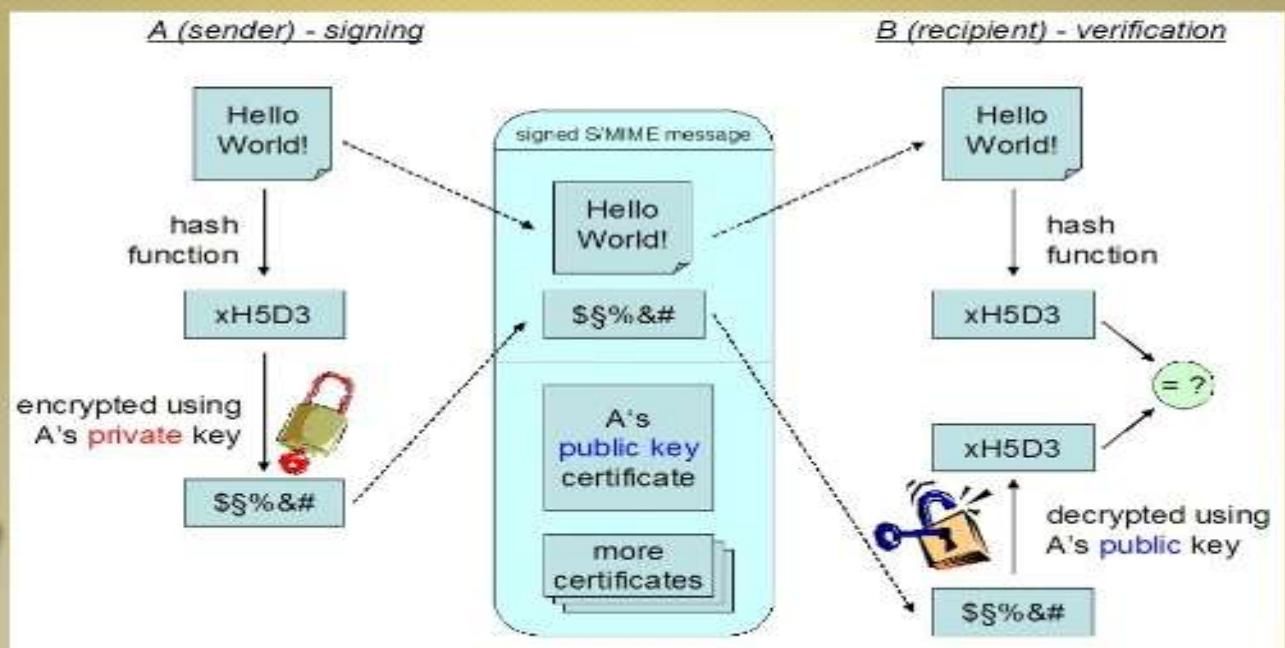
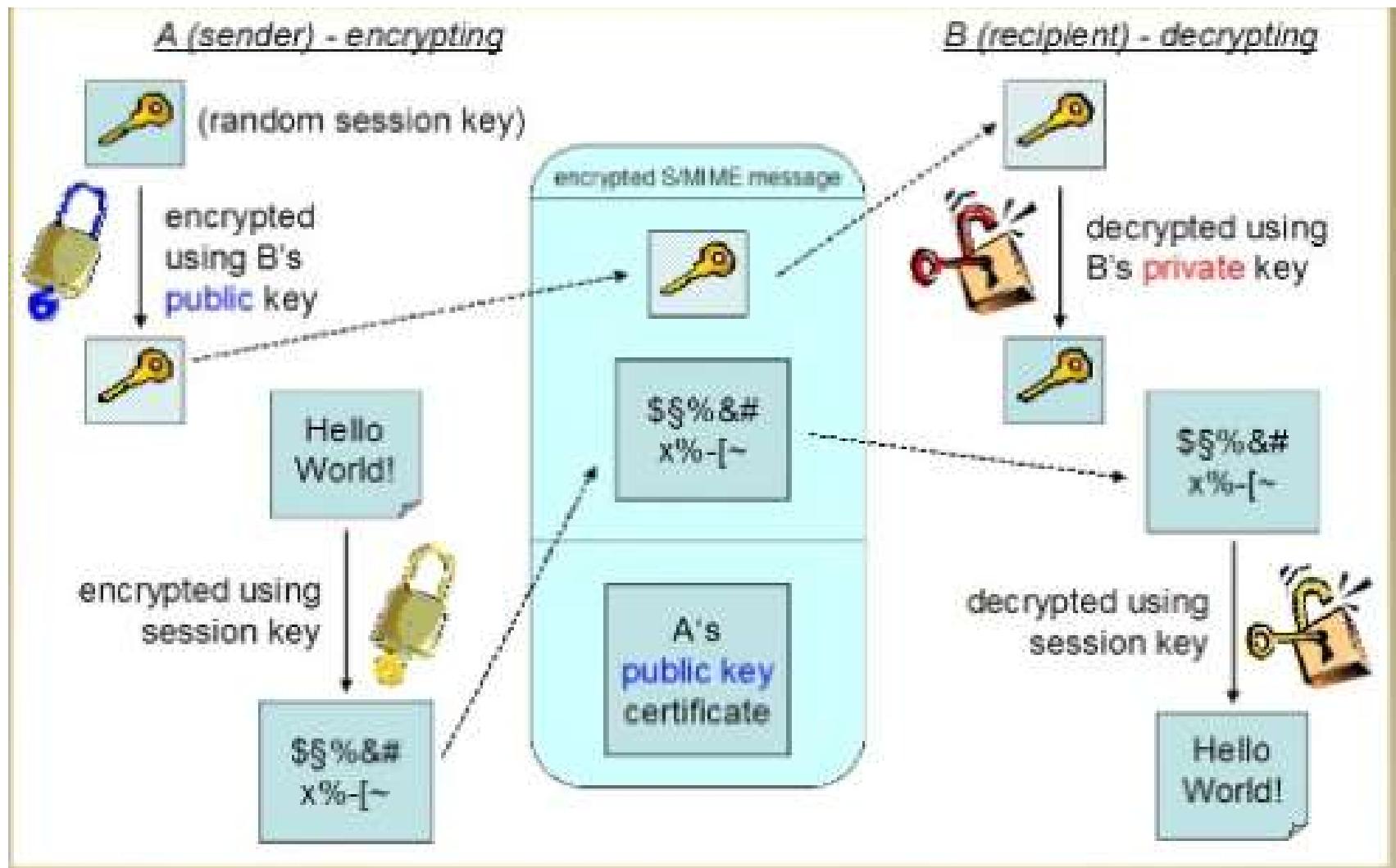


Image copied from internet (Not the complete PPT :P)

fppt.com

S/MIME Encrypted mail



S/MIME Functions

- enveloped data
 - encrypted content and associated keys
- signed data
 - encoded message + signed digest
- clear-signed data
 - cleartext message + encoded signed digest
- signed & enveloped data
 - nesting of signed & encrypted entities

S/MIME Cryptographic Algorithms

- hash functions: SHA-1 & MD5
- digital signatures: DSS & RSA
- session key encryption: ElGamal & RSA
- message encryption: Triple-DES, RC2/40 and others
- have a procedure to decide which algorithms to use

S/MIME Certificate Processing

- managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust
- each client has a list of trusted CA's certs
- and own public/private key pairs & certs
- certificates must be signed by trusted CA's

Contd..

- User Agent Role
 - Key Generation
 - Registration
 - Certificate storage and retrieval.
- Enhanced Security Services
 - Signed receipts
 - Security labels
 - Secure mailing lists.
 - Signing certificates

Certificate Authorities

- have several well-known CA's
- Verisign one of most widely used
- Verisign issues several types of Digital IDs
- with increasing levels of checks & hence trust

Class	Identity Checks	Usage
1	name/email check	web browsing/email
2+	enroll/addr check	email, subs, s/w validate
3+	ID documents	e-banking/service access

Summary

- have considered:
 - secure email
 - PGP
 - S/MIME

Cryptography and Network Security

IP Security

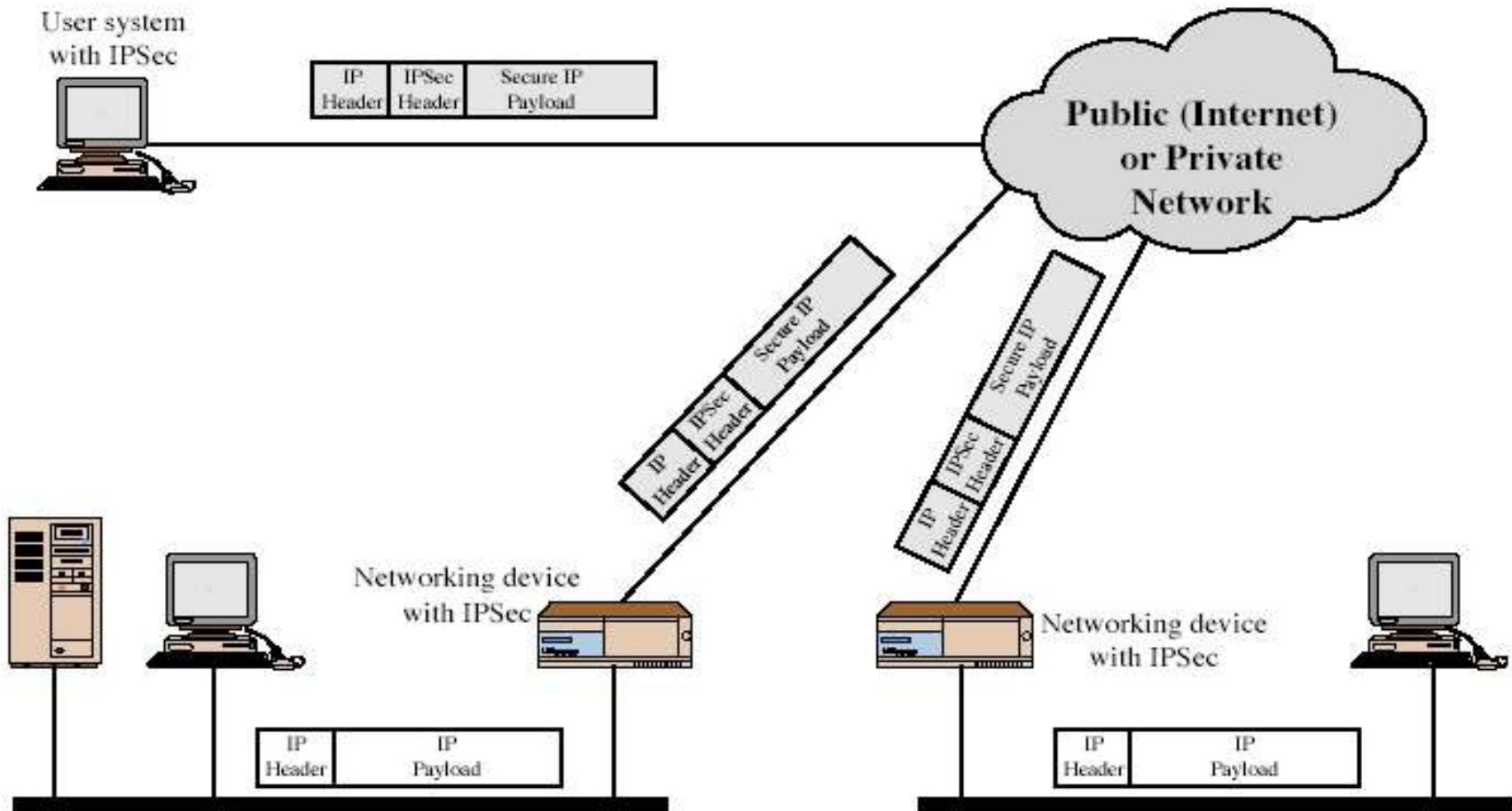
IP Security

- have considered some application specific security mechanisms
 - eg. S/MIME, PGP, Kerberos, SSL/HTTPS
- however there are security concerns that cut across protocol layers
- would like security implemented by the network for all applications

IPSec

- general IP Security mechanisms
- provides
 - authentication
 - confidentiality
 - key management
- applicable to use over LANs, across public & private WANs, & for the Internet

IPSec Uses



Benefits of IPSec

- in a firewall/router provides strong security to all traffic crossing the perimeter
- is resistant to bypass
- is below transport layer, hence transparent to applications
- can be transparent to end users
- can provide security for individual users if desired

IP Security Architecture

- specification is quite complex
- defined in numerous RFC's
 - incl. RFC 2401/2402/2406/2408
 - many others, grouped by category
- mandatory in IPv6, optional in IPv4

IPSec Services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
 - a form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

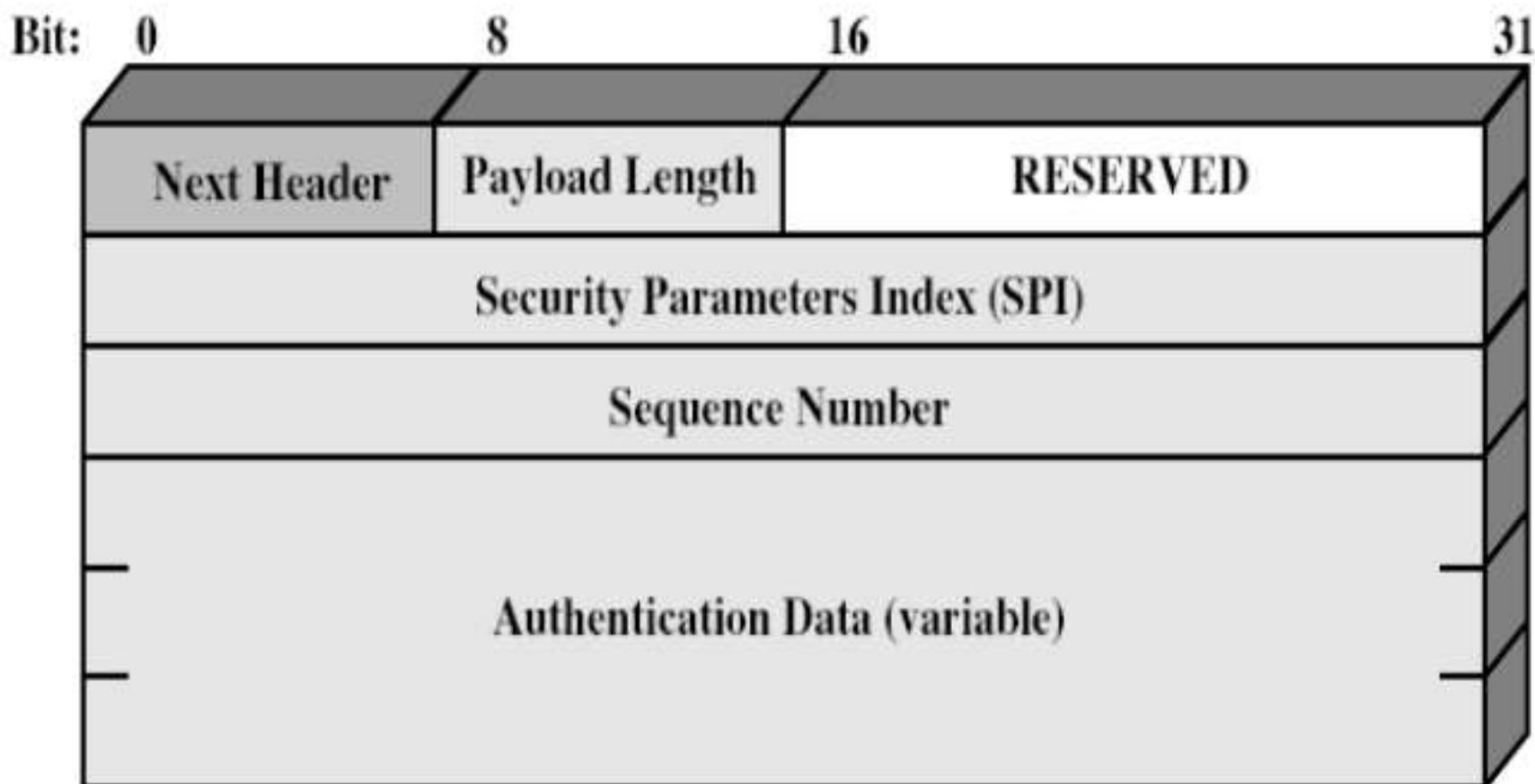
Security Associations

- a one-way relationship between sender & receiver that affords security for traffic flow
- defined by 3 parameters:
 - Security Parameters Index (SPI)
 - IP Destination Address
 - Security Protocol Identifier
- has a number of other parameters
 - seq no, AH & EH info, lifetime etc
- have a database of Security Associations

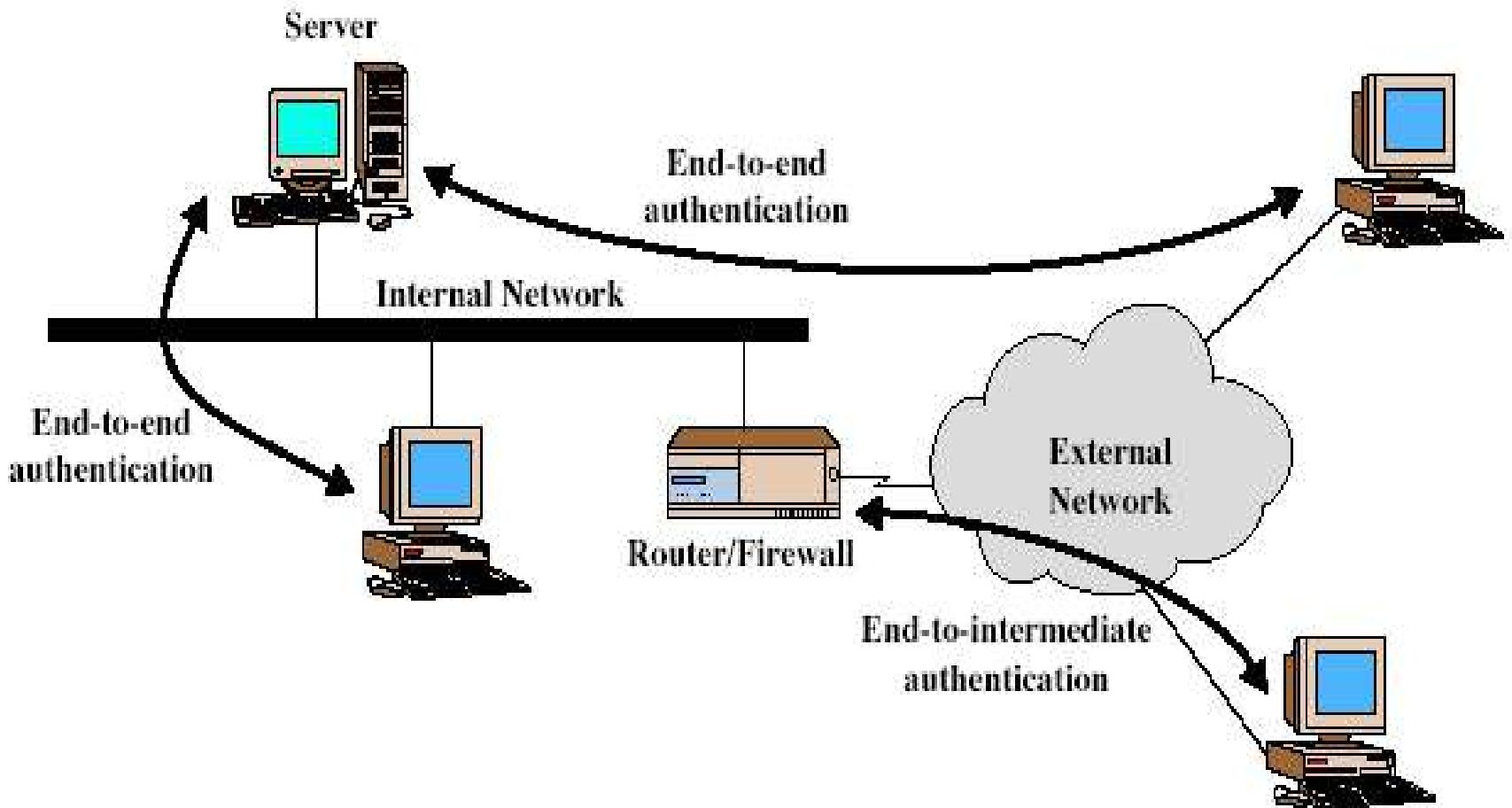
Authentication Header (AH)

- provides support for data integrity & authentication of IP packets
 - end system/router can authenticate user/app
 - prevents address spoofing attacks by tracking sequence numbers
- based on use of a MAC
 - HMAC-MD5-96 or HMAC-SHA-1-96
- parties must share a secret key

Authentication Header



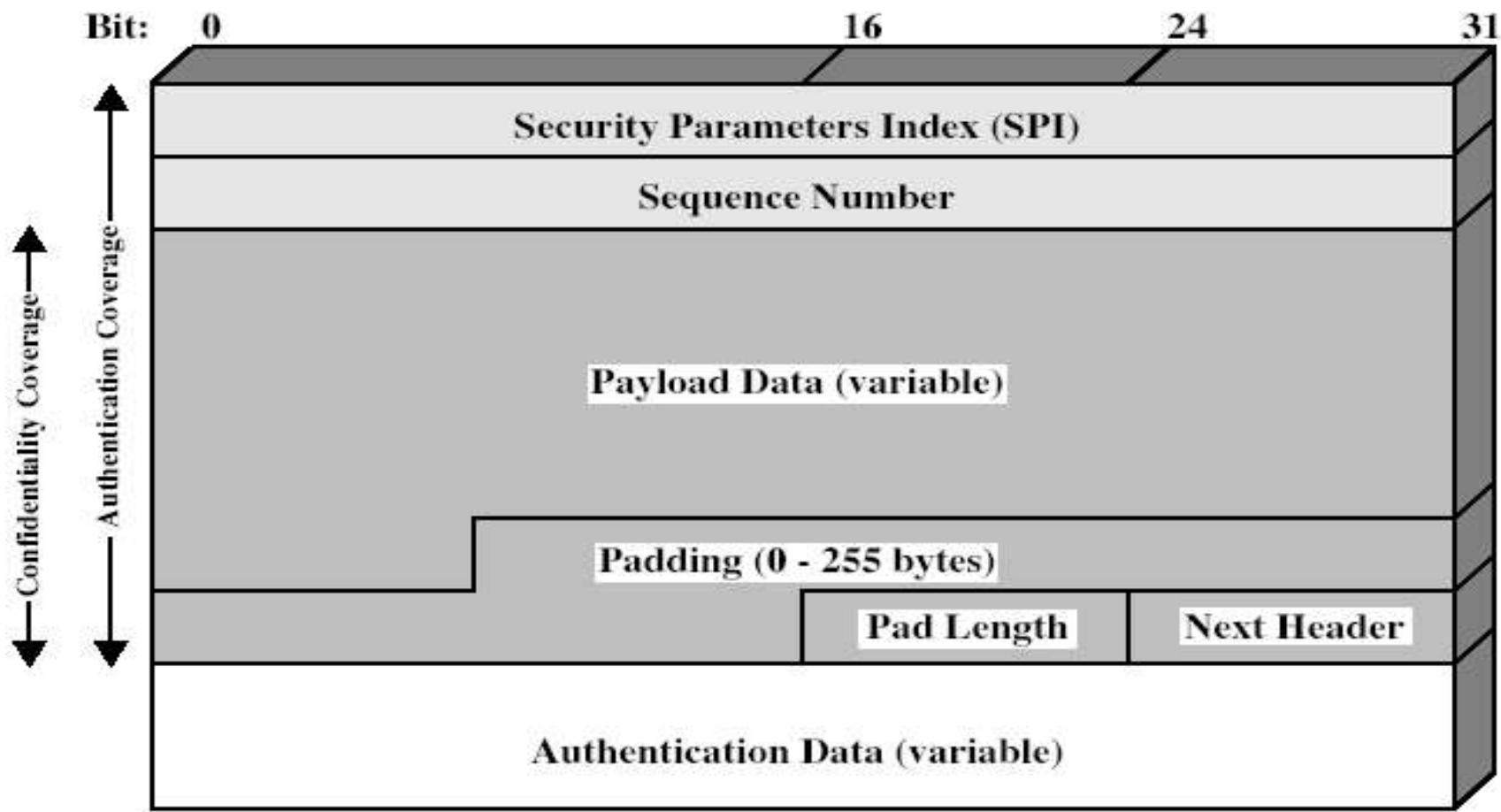
Transport & Tunnel Modes



Encapsulating Security Payload (ESP)

- provides message content confidentiality & limited traffic flow confidentiality
- can optionally provide the same authentication services as AH
- supports range of ciphers, modes, padding
 - incl. DES, Triple-DES, RC5, IDEA, CAST etc
 - CBC most common
 - pad to meet blocksize, for traffic flow

Encapsulating Security Payload



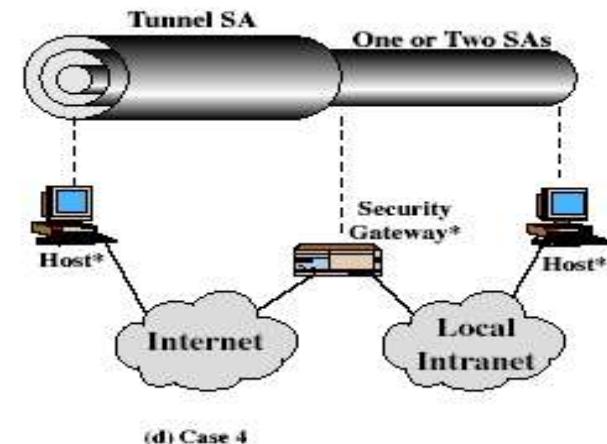
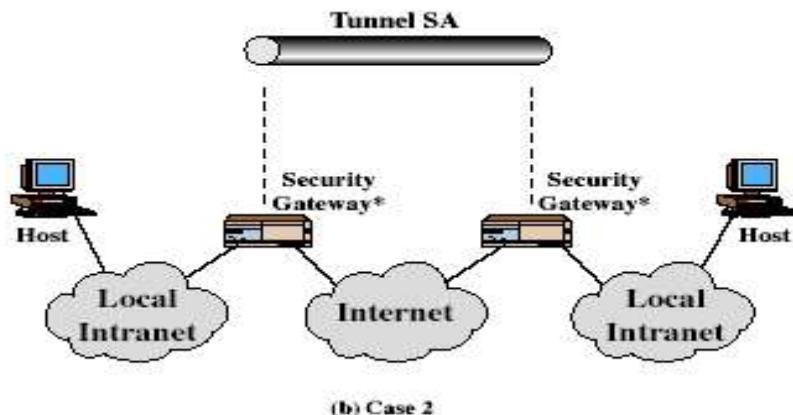
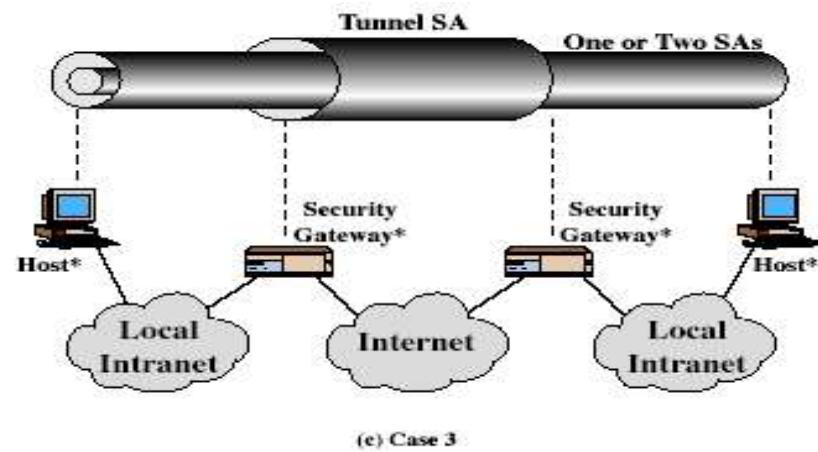
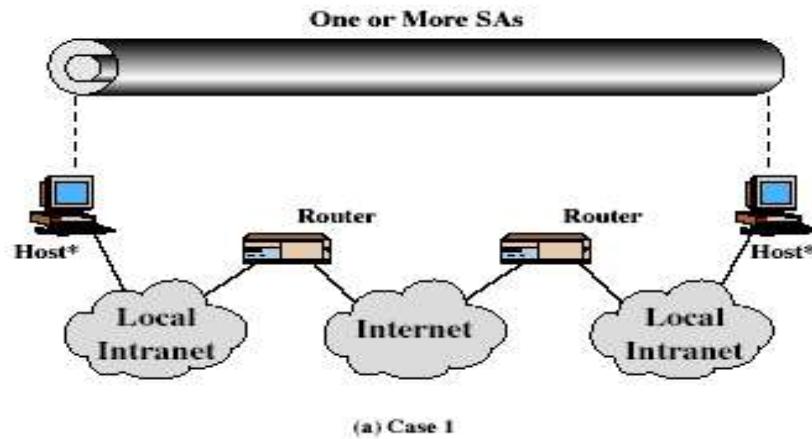
Transport vs Tunnel Mode ESP

- transport mode is used to encrypt & optionally authenticate IP data
 - data protected but header left in clear
 - can do traffic analysis but is efficient
 - good for ESP host to host traffic
- tunnel mode encrypts entire IP packet
 - add new header for next hop
 - good for VPNs, gateway to gateway security

Combining Security Associations

- SA's can implement either AH or ESP
- to implement both need to combine SA's
 - form a security bundle
- have 4 cases (see next)

Combining Security Associations



Key Management

- handles key generation & distribution
- typically need 2 pairs of keys
 - 2 per direction for AH & ESP
- manual key management
 - sysadmin manually configures every system
- automated key management
 - automated system for on demand creation of keys for SA's in large systems
 - has Oakley & ISAKMP elements

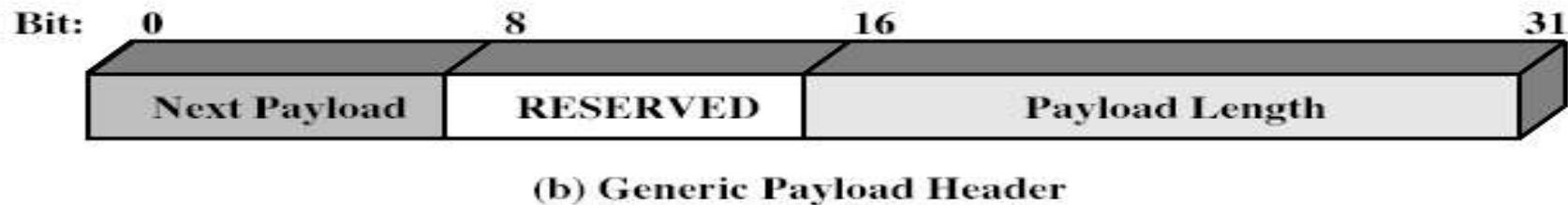
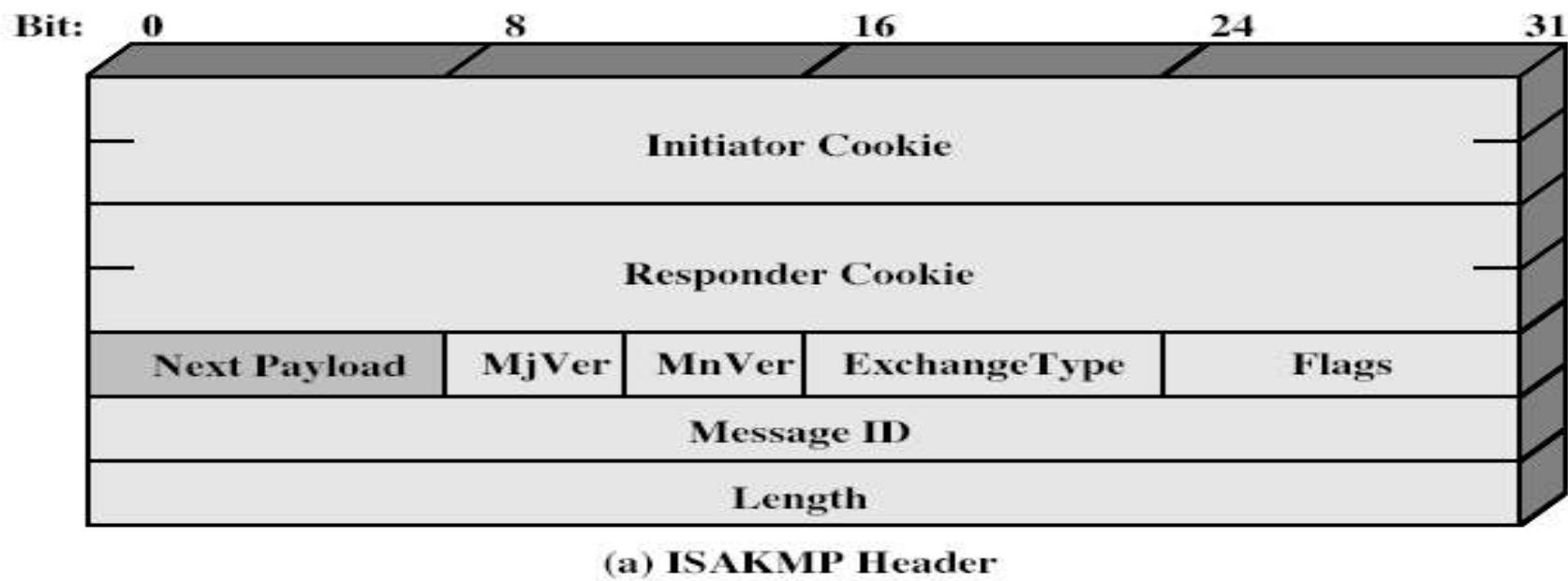
Oakley

- a key exchange protocol
- based on Diffie-Hellman key exchange
- adds features to address weaknesses
 - cookies, groups (global params), nonces, DH key exchange with authentication
- can use arithmetic in prime fields or elliptic curve fields

ISAKMP

- Internet Security Association and Key Management Protocol
- provides framework for key management
- defines procedures and packet formats to establish, negotiate, modify, & delete SAs
- independent of key exchange protocol, encryption alg, & authentication method

ISAKMP



Summary

- have considered:
 - IPSec security framework
 - AH
 - ESP
 - key management & Oakley/ISAKMP

Cryptography and Network Security

Web Security

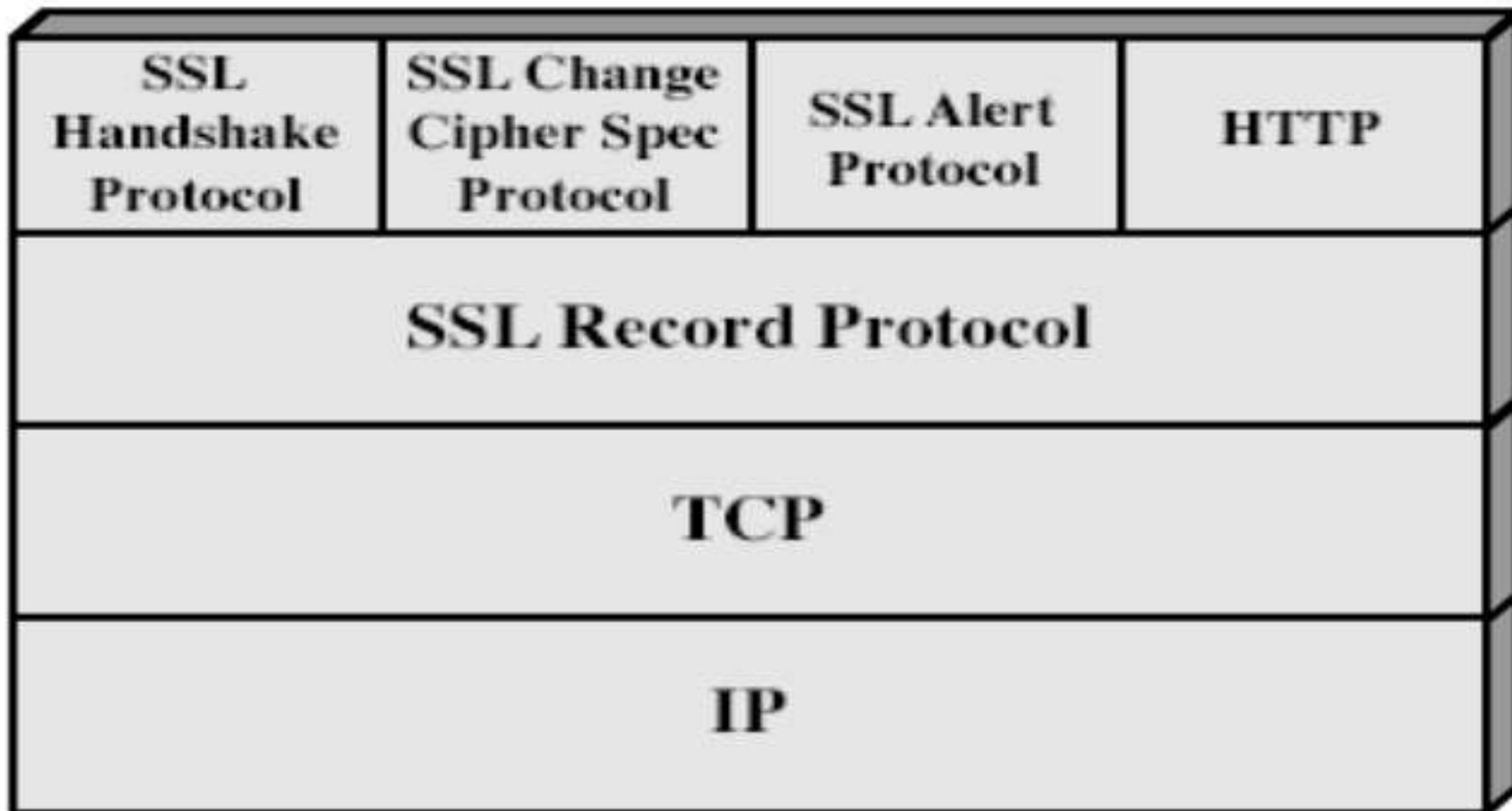
Web Security

- 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

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)
- uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

SSL Architecture



SSL Architecture

- **SSL session**
 - an association between client & server
 - created by the Handshake Protocol
 - define a set of cryptographic parameters
 - may be shared by multiple SSL connections
- **SSL connection**
 - a transient, peer-to-peer, communications link
 - associated with 1 SSL session

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 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

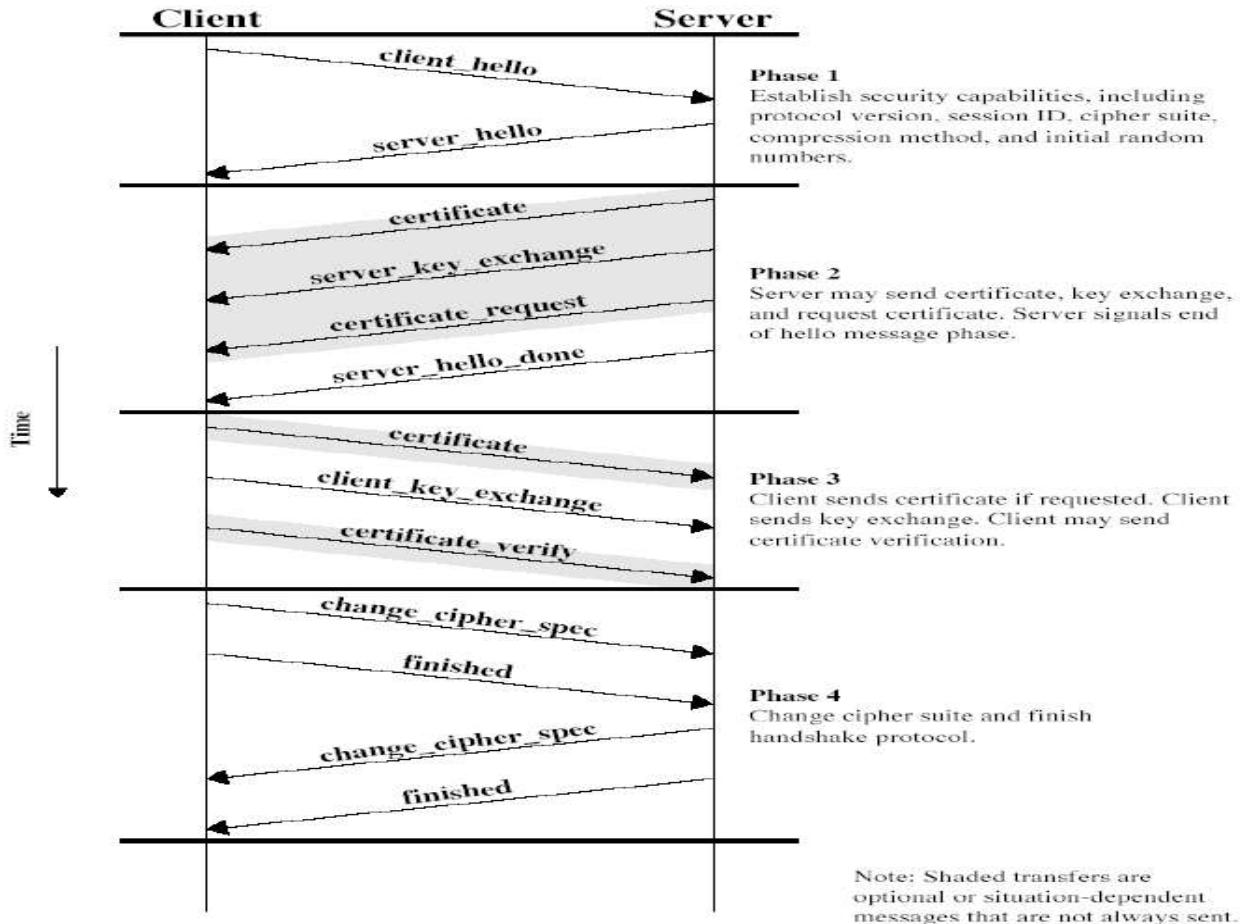
SSL Alert Protocol

- 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
- 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



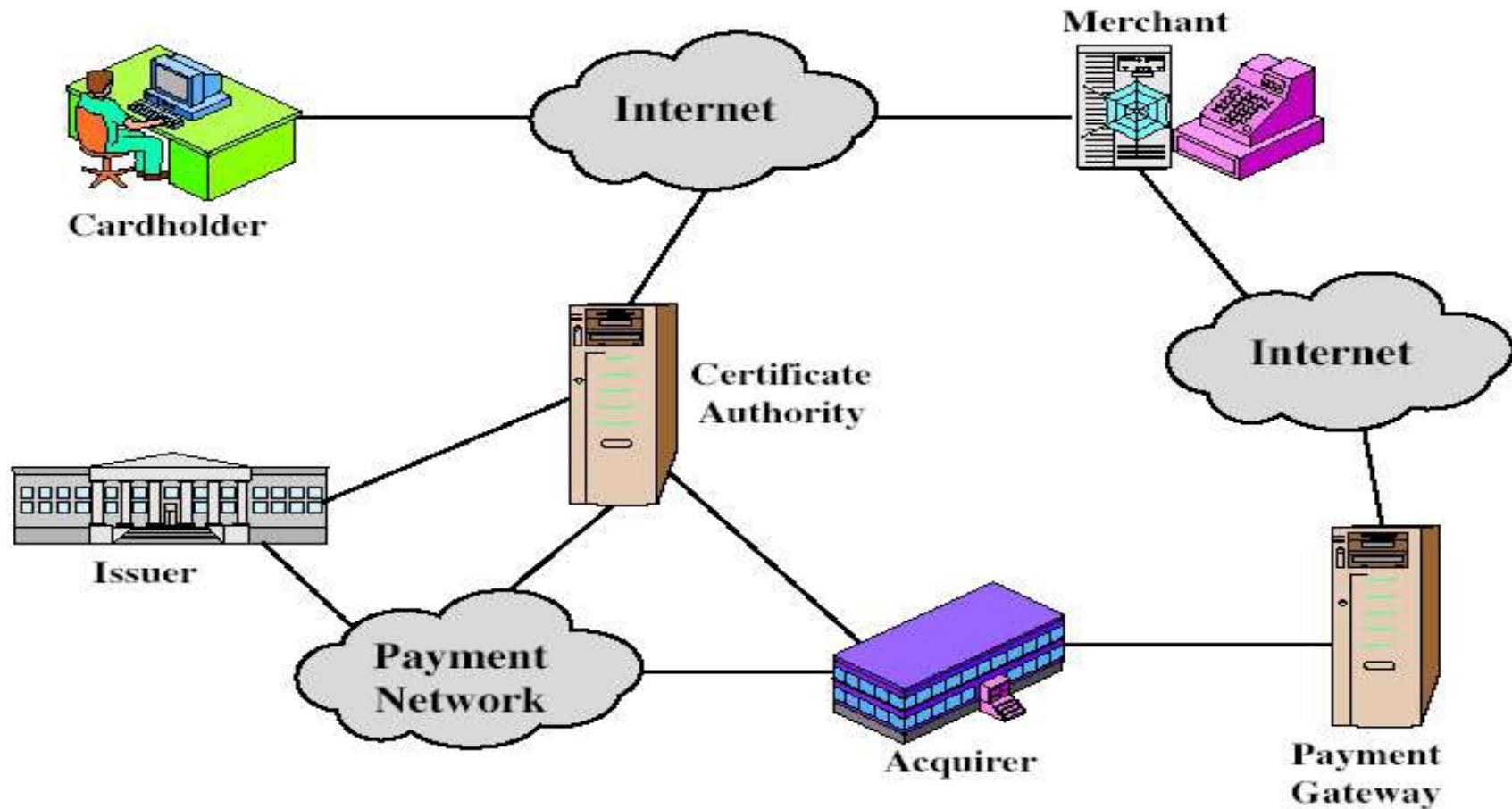
TLS (Transport Layer Security)

- 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

Secure Electronic Transactions (SET)

- open encryption & security specification
- to protect Internet credit card transactions
- developed in 1996 by Mastercard, Visa etc
- not a payment system
- rather a set of security protocols & formats
 - secure communications amongst parties
 - trust from use of X.509v3 certificates
 - privacy by restricted info to those who need it

SET Components



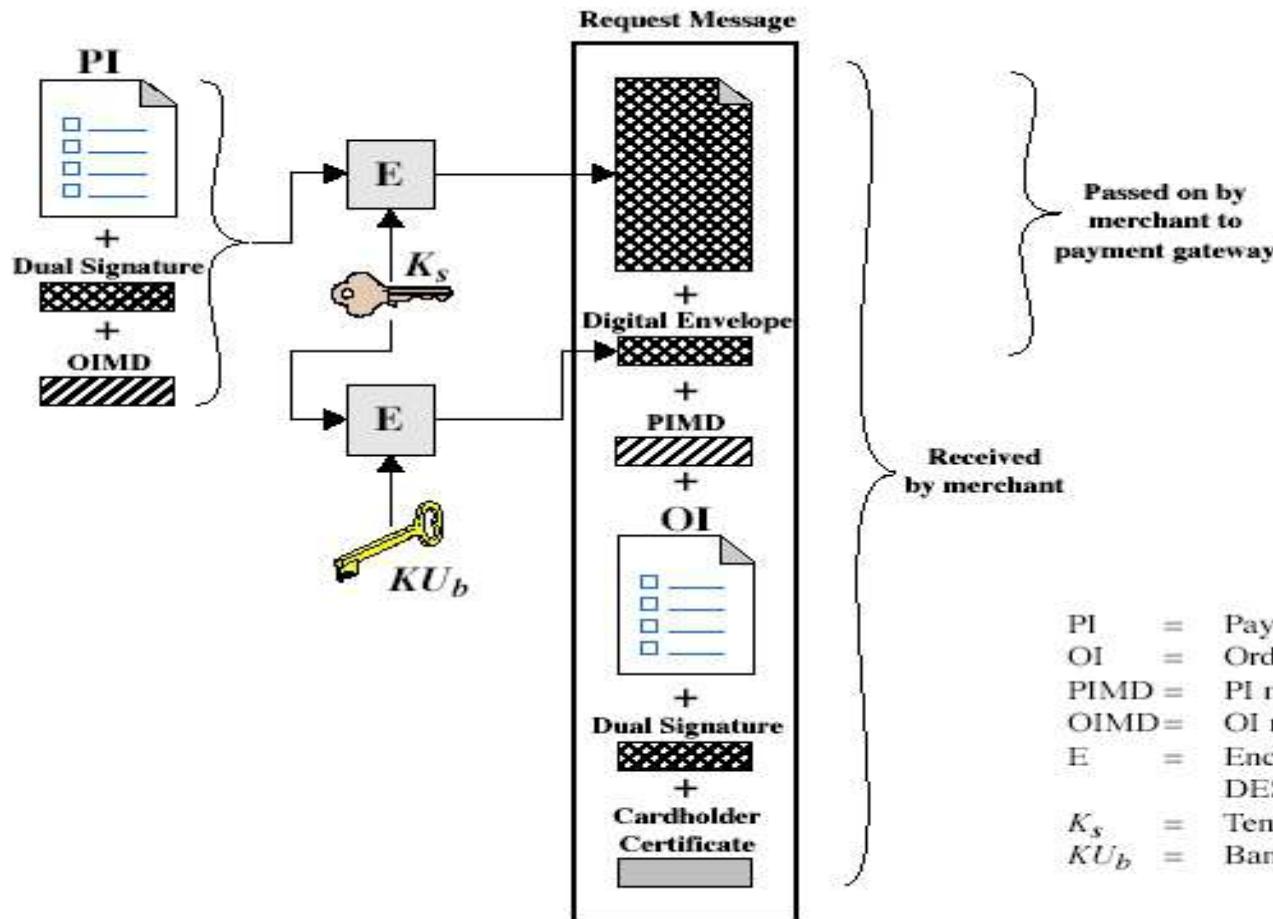
SET Transaction

1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. merchant requests payment

Dual Signature

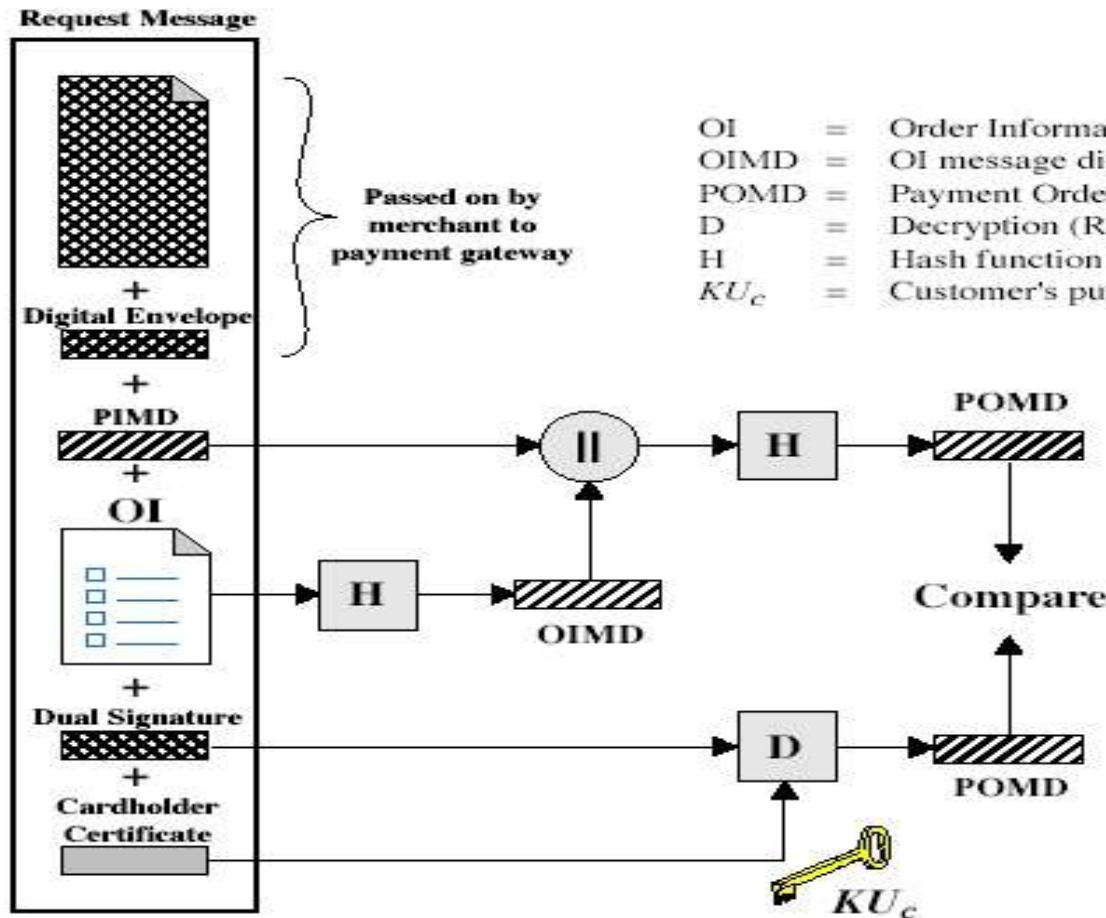
- customer creates dual messages
 - order information (OI) for merchant
 - payment information (PI) for bank
- neither party needs details of other
- but **must** know they are linked
- use a dual signature for this
 - signed concatenated hashes of OI & PI

Purchase Request – Customer



PI	=	Payment Information
OI	=	Order Information
PIMD	=	PI message digest
OIMD	=	OI message digest
E	=	Encryption (RSA for asymmetric; DES for symmetric)
K_s	=	Temporary symmetric key
KU_b	=	Bank's public key-exchange key

Purchase Request – Merchant



OI = Order Information
OIMD = OI message digest
POMD = Payment Order message digest
D = Decryption (RSA)
H = Hash function (SHA-1)
 KU_c = Customer's public signature key

Purchase Request – Merchant

1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization (described later)
4. sends a purchase response to cardholder

Payment Gateway Authorization

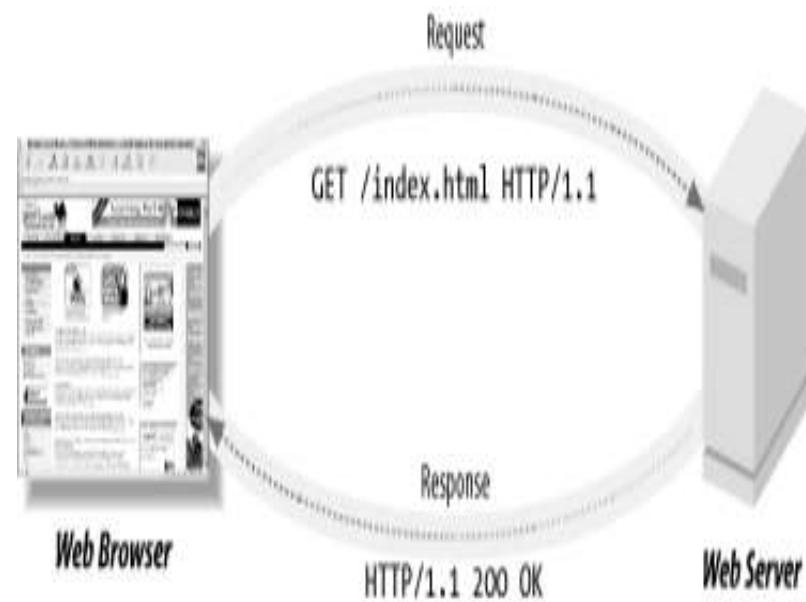
1. verifies all certificates
2. decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. verifies dual signature on payment block
6. verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. requests & receives an authorization from issuer
8. sends authorization response back to merchant

Payment Capture

- merchant sends payment gateway a payment capture request
- gateway checks request
- then causes funds to be transferred to merchants account
- notifies merchant using capture response

HTTP

- HTTP stands for **Hypertext Transfer Protocol.**
- HTTP provides a set of rules and standards that govern how information is transmitted on the World Wide Web.
- Computers on the World Wide Web use the HyperText Transfer Protocol to talk with each other
- <http://www.google.co.in>
- The first part of an address (URL) of a site on the Internet, signifying a document written in Hypertext Markup Language (HTML).



HTTP

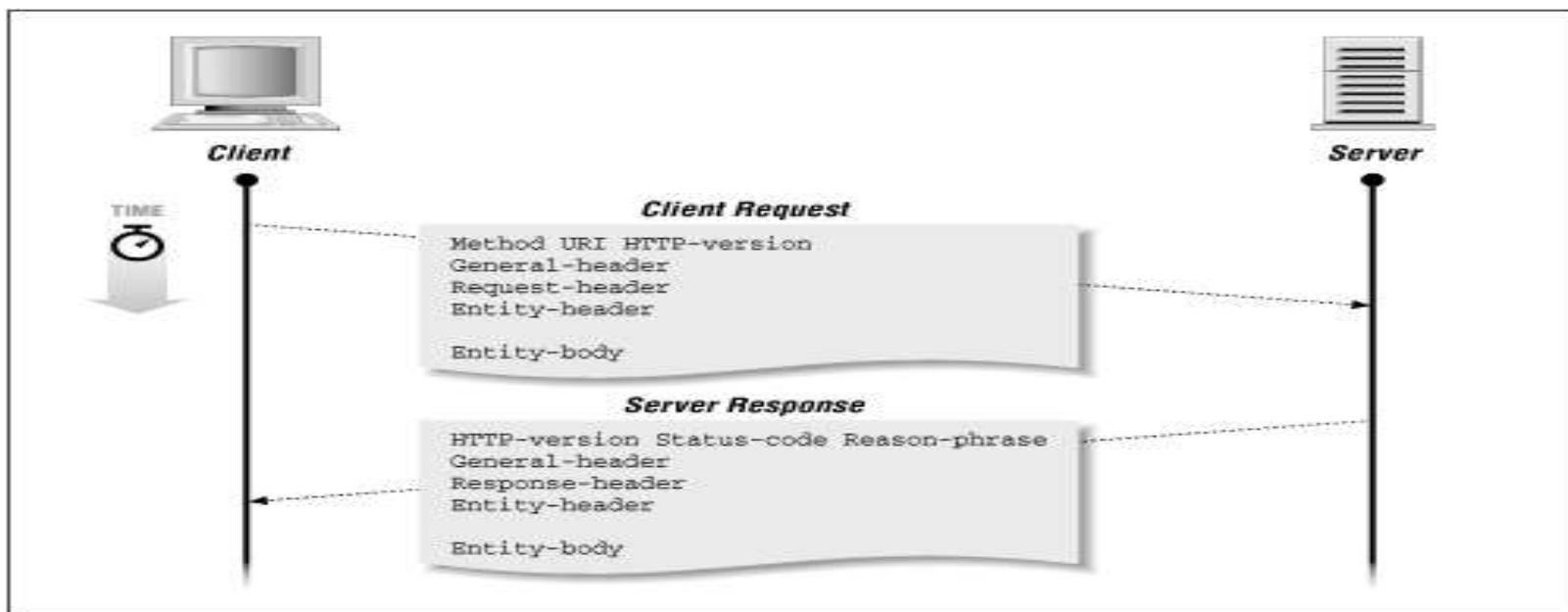
- ❖ HTTP is a client-server protocol by which two machines communicate using a reliable, connection-oriented transport service such as the TCP.
 - ❖ A browser is an *HTTP client* because it sends requests to an *HTTP server* (Web server), which then sends responses back to the client
 - ❖ An HTTP server is a program that sits listening on a machine's port for HTTP requests.
 - ❖ The standard (and default) port for HTTP servers to listen on is 80, though they can use any port.
- ❖ HTTP can be "implemented on top of any other protocol on the Internet, or on other networks."
- ❖ HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used." e.g. TCP.

HTTP

- ❖ HTTP is stateless. The lifetime of a connection corresponds to a single request-response sequence
 - ❖ An HTTP client opens a TCP/IP connection to the server via a socket, transmits a request for a document, then waits for a reply from the server. Once the request-response sequence is completed, the socket is closed.
 - ❖ There is no "memory" between client connections.
 - ❖ The pure HTTP server implementation treats every request as if it was brand-new.

How HTTP Works

- HTTP Server is implemented by Apache HTTP Server · Microsoft IIS · Jigsaw · Zope etc.
- Each client-server transaction, whether a request or a response, consists of three main parts
 - A response or request line
 - Header information
 - The body



Advantages of HTTP

- ❖ Platform independent- Allows Straight cross platform porting.
- ❖ No Runtime support required to run properly.
- ❖ Usable over Firewalls! Global applications possible.
- ❖ Not Connection Oriented- No network overhead to create and maintain session state and information.

HTTP Limitations

Security Concerns

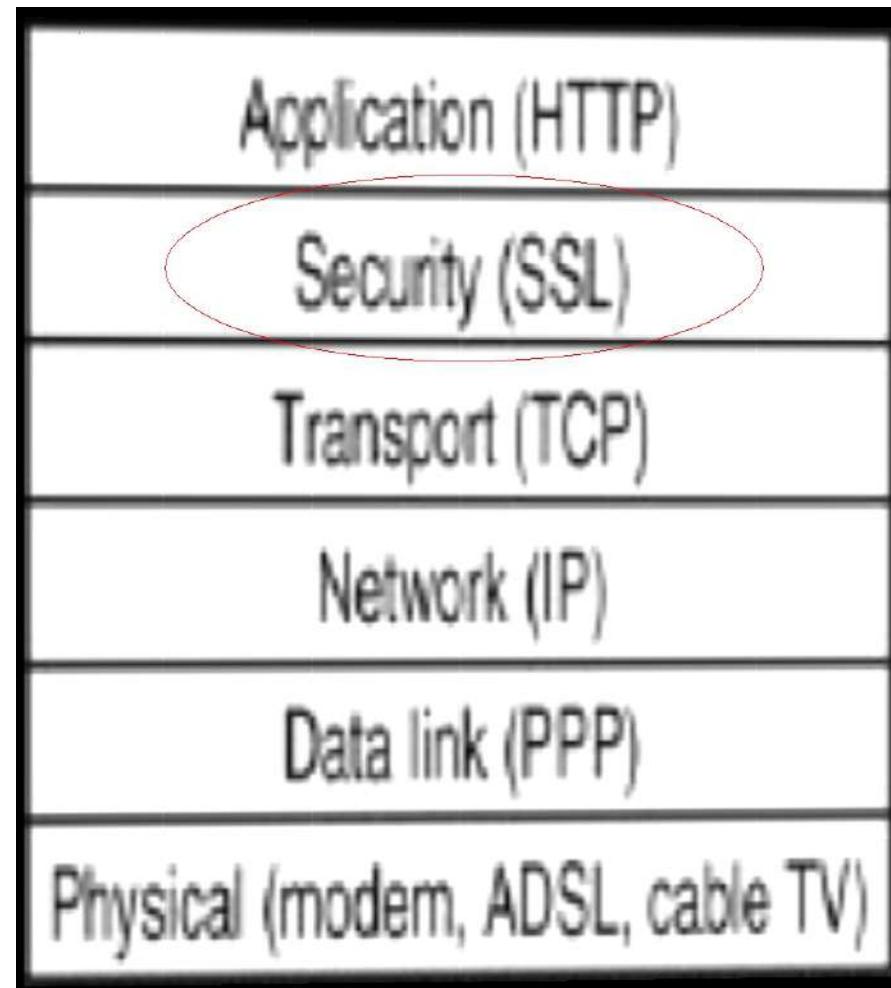
- **Privacy** -Anyone can see content
- **Integrity** -Someone might alter content. HTTP is insecure since no encryption methods are used. Hence is subject to man in the middle and eavesdropping of sensitive information.
- **Authentication** -Not clear who you are talking with. Anyone who intercepts the request can determine the username and password being used.
- **Stateless** - Need State management techniques to maintain the information across multiple request-response cycles.

HTTPS

☞ HTTPS stands for Hypertext Transfer Protocol over Secure Socket Layer, or HTTP over SSL.

- SSL acts like a sub layer under regular HTTP application layering.

☞ HTTPS encrypts an HTTP message prior to transmission and decrypts a message upon arrival.



HTTPS

HISTORY

- ❑ Netscape Communications created HTTPS in 1994 for its Netscape Navigator web browser.
- ❑ Originally, HTTPS was used with SSL protocol. As SSL evolved into Transport Layer Security (TLS), the current version of HTTPS was formally specified by RFC 2818 in May 2000.

HTTPS

- ☞ HTTPS by default uses port 443 as opposed to the standard HTTP port of 80.
- ☞ URL's beginning with HTTPS indicate that the connection between client and browser is encrypted using SSL
e.g.: https://login.yahoo.com/config/login_verify2?&.src=ym
- ☞ SSL transactions are negotiated by means of a key based encryption algorithm between the client and the server, this key is usually either 40 or 128 bits in strength (**the higher the number of bits the more secure the transaction**).

HTTPS

❖ Need SSL if...

- ❖ you have an online store or accept online orders and credit cards
- ❖ you offer a login or sign in on your site
- ❖ you process sensitive data such as address, birth date, license, or ID numbers
- ❖ you need to comply with privacy and security requirements

Certification Authority (CA) is an entity that issues digital certificates for use by other parties. It is an example of a trusted third party.

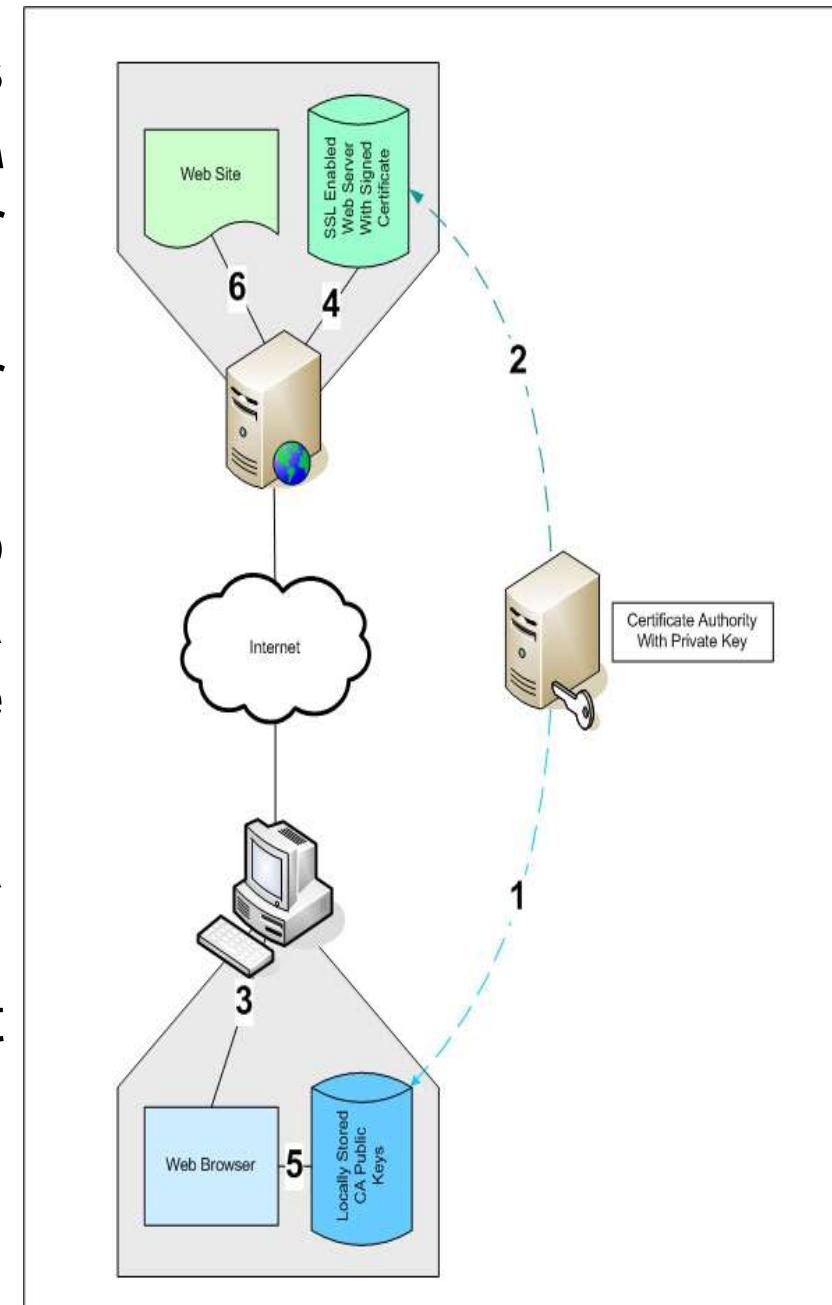
e.g. VeriSign, Thwate, Geotrust etc

HTTPS

- ☞ Ability to connect to server via HTTP
secure consists of:
 - ☞ Generating key
 - ☞ Generating certificate signing request
 - ☞ Generating self signed certificate
 - ☞ Certificate Authority signed certificate
 - ☞ Configuring web server.

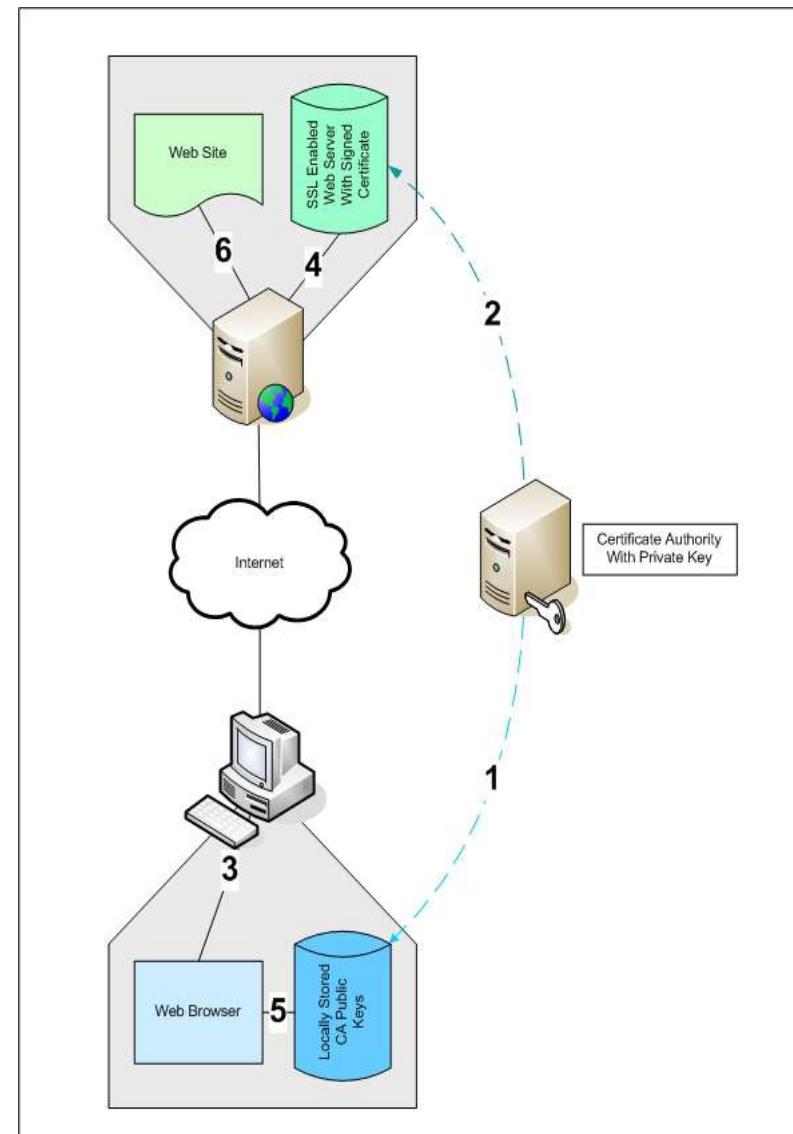
SSL Diagram

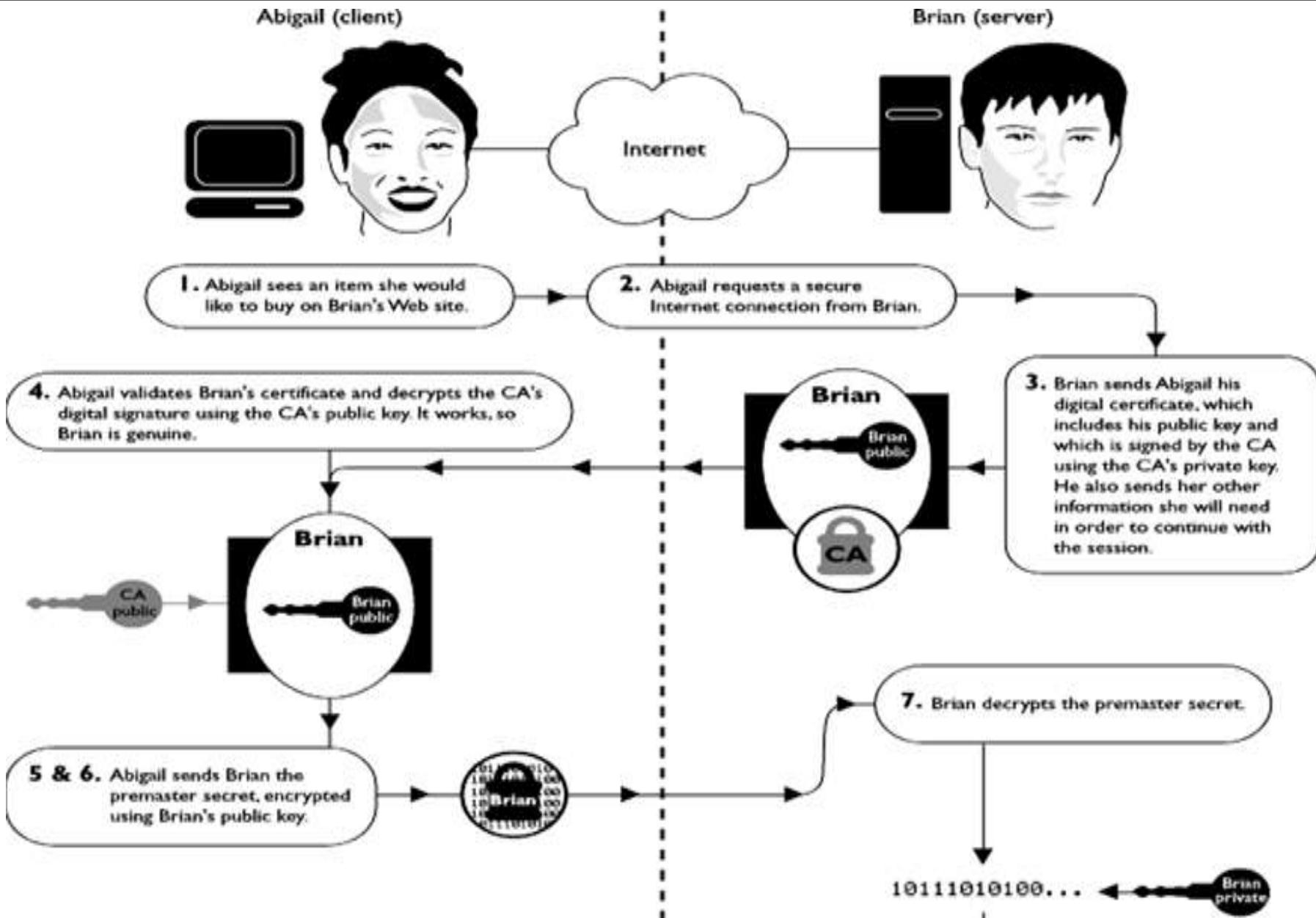
- When any modern browser is installed, it is sent with several CA *issuer certificates*. These issuer certificates contain a public key for the issuer, among other information.
- When a web designer decides to use SSL he needs to purchase a *certificate that is signed* using the CA's private key.
- The web browser starts a connection to an HTTPS site. Along with this request the client sends all supported encryption schemes.



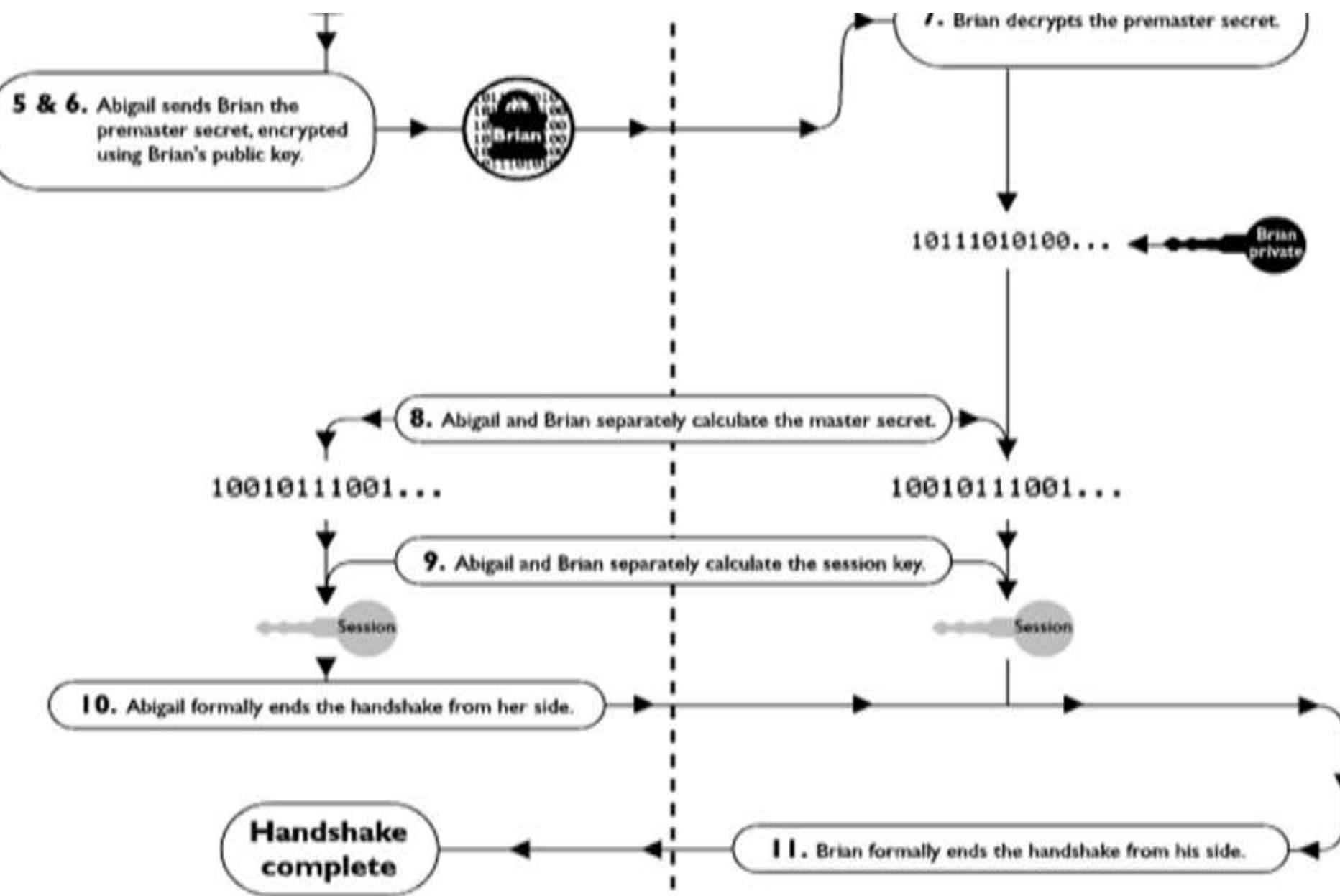
SSL Diagram

- As a response to the browser's connection request, the Server sends a copy of the certificate from step 2. Along with this transmission is the server's answer to the encryption negotiation.
- Once a certificate is downloaded, the signature of the certificate (*that was signed using the CA's private key*) is checked using the CA's public key (installed in the browser in step 1).
- The connection succeeds, the client can now download and upload to the web site with the security of encryption.

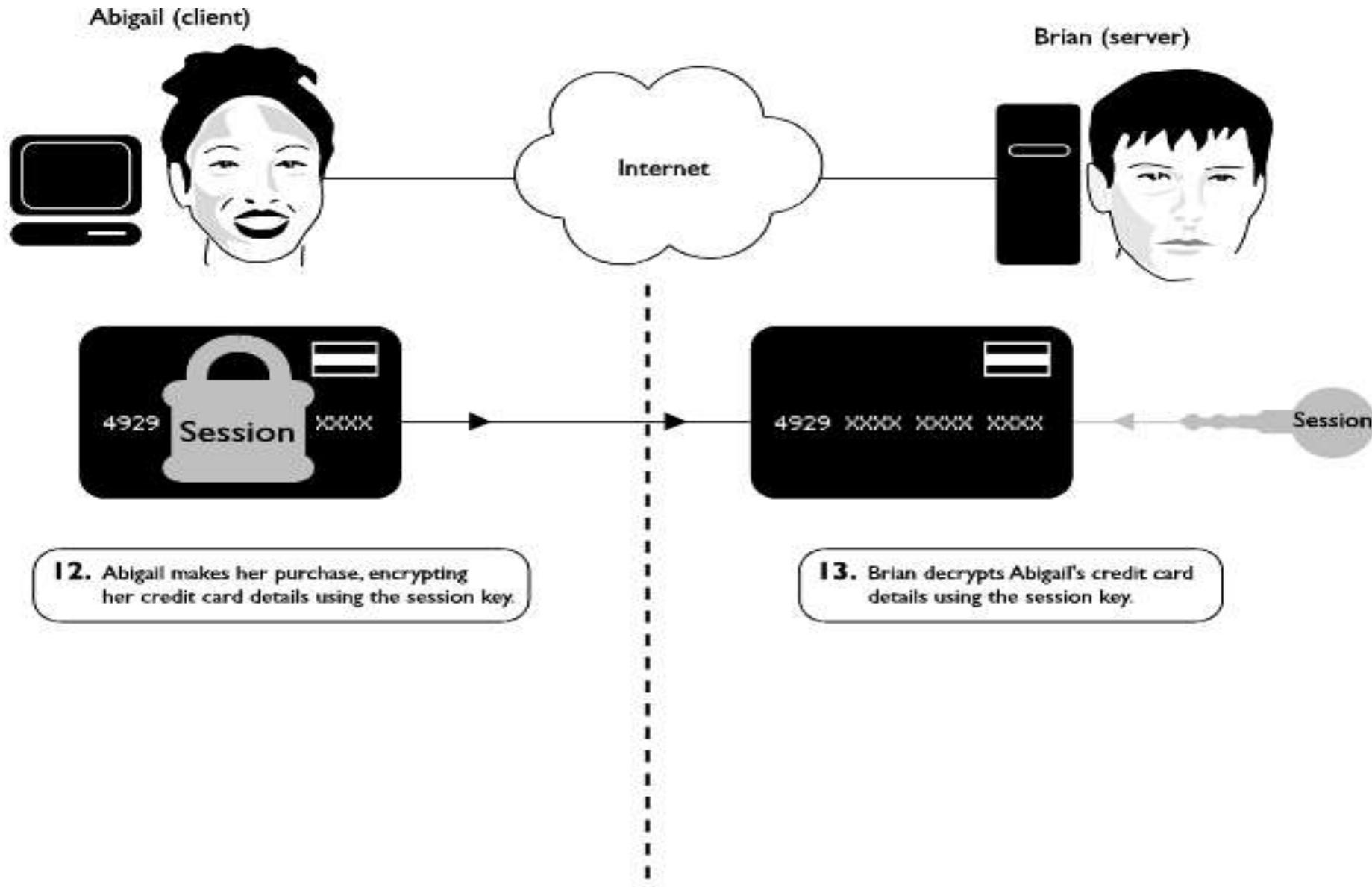




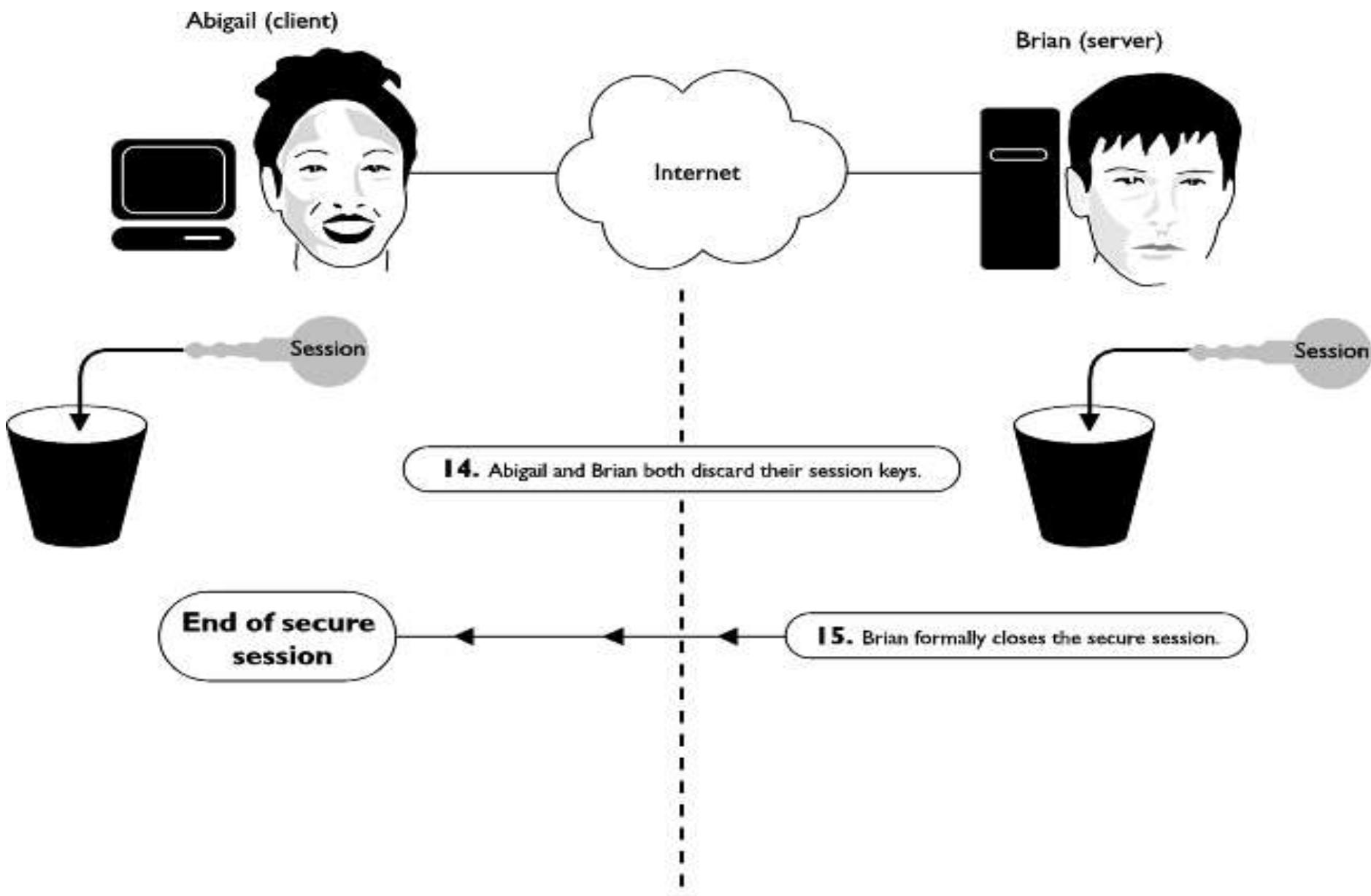
Part One: SSL Handshake



Part One: SSL Handshake



Part Two: Transfer of Confidential Information



Part Three: Secure Session Closure

How SSL Overcomes HTTP Security Concerns

❖ Secure Sockets Layer technology protects your Web site and makes it easy for your Web site visitors to trust you in three essential ways:

❖ Privacy

❖ An SSL Certificate enables **encryption** of sensitive information during online transactions.

❖ Integrity.

❖ A Certificate Authority **verifies** the identity of the certificate owner when it is issued.

❖ Authentication.

❖ Each SSL Certificate contains unique, **authenticated** information about the certificate owner.

Limitations of HTTPS

- ❖ HTTPS cannot prevent stealing confidential information from the pages cached on the browser.
- ❖ Since in SSL data is encrypted only during transmission on the network, it is in clear text in the browser memory

- ❖ HTTPS is slightly slower than HTTP.
- ❖ HTTPS adds computational overhead as well as network overhead.

Summary

- have considered:
 - need for web security
 - SSL/TLS transport layer security protocols
 - SET secure credit card payment protocols
 - HTTP and HTTPS

Authentication Applications

Outline

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

Security Concerns

- key concerns are **confidentiality** and **timeliness**
- to provide confidentiality must encrypt identification and session key info
- which requires the use of previously shared private or public keys
- need timeliness to prevent **replay attacks**
- provided by using sequence numbers or timestamps or challenge/response

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
 - \parallel = concatenation

A Simple Authentication Dialogue

- | | |
|------------------|-------------------------------------|
| (1) $C \vee AS:$ | $ID_C \parallel P_C \parallel ID_V$ |
| (2) $AS \vee C:$ | Ticket |
| (3) $C \vee V:$ | $ID_C \parallel \text{Ticket}$ |

$\text{Ticket} = E_{K_V}[ID_C \parallel P_C \parallel ID_V]$

Version 4 Authentication Dialogue

- Problems:
 - Lifetime associated with the ticket-granting ticket
 - If too short \vee repeatedly asked for password
 - If too long \vee greater opportunity to replay
- The threat is that an opponent will steal the ticket and use it before it expires

Version 4 Authentication Dialogue

Authentication Service Exchange: To obtain Ticket-Granting Ticket

- (1) $C \vee AS:$ $ID_c \parallel ID_{tgs} \parallel TS_1$
- (2) $AS \vee C:$ $E_{K_c}[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel \text{Lifetime}_2 \parallel \text{Ticket}_{tgs}]$

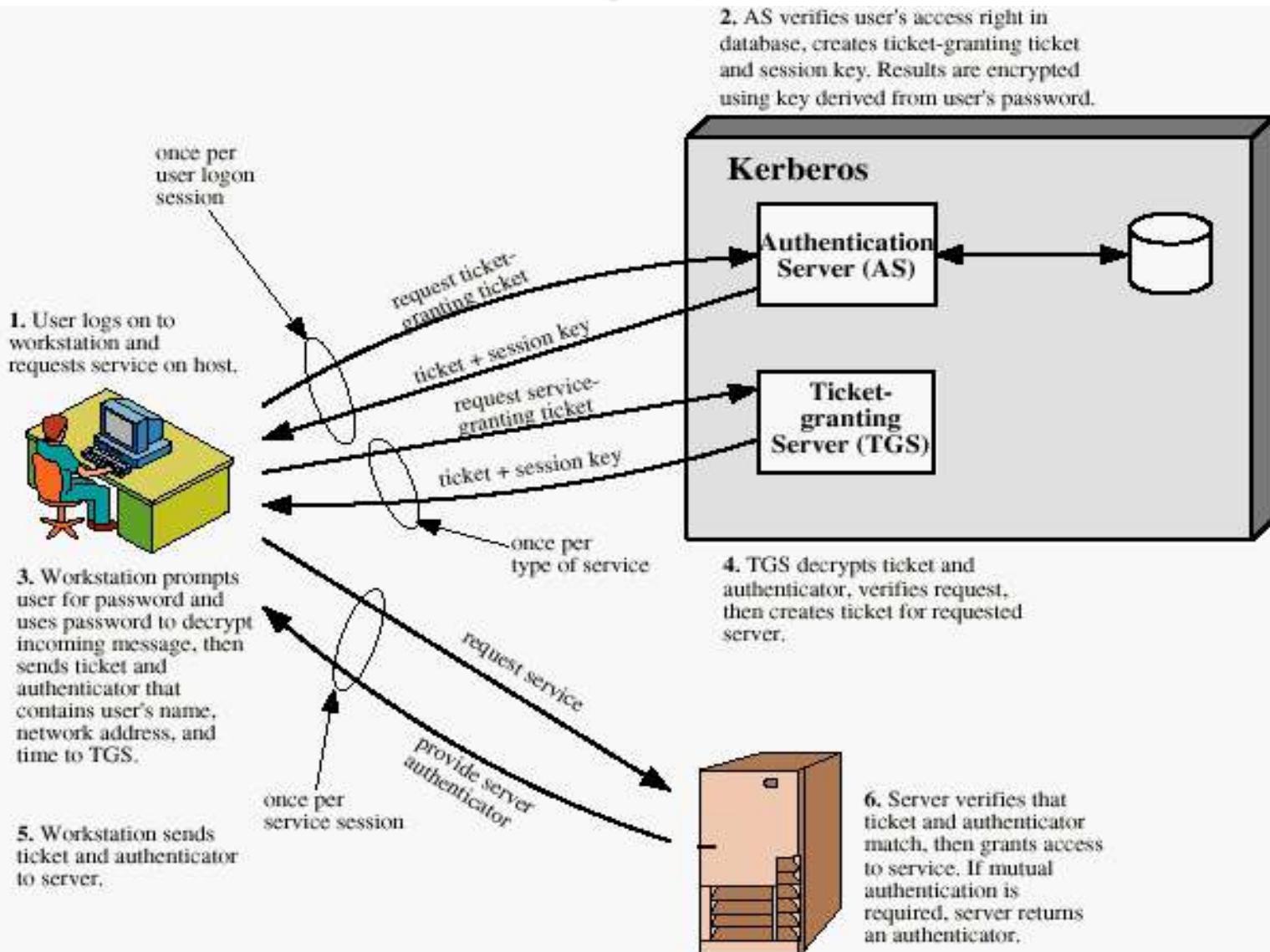
Ticket-Granting Service Exchange: To obtain Service-Granting Ticket

- (3) $C \vee TGS:$ $ID_v \parallel \text{Ticket}_{tgs} \parallel \text{Authenticator}_c$
- (4) $TGS \vee C:$ $E_{K_c}[K_{c,v} \parallel ID_v \parallel TS_4 \parallel \text{Ticket}_v]$

Client/Server Authentication Exchange: To Obtain Service

- (5) $C \vee V:$ $\text{Ticket}_v \parallel \text{Authenticator}_c$
- (6) $V \vee C:$ $E_{K_c,v}[TS_5 + 1]$

Overview of Kerberos



Request for Service in Another Realm

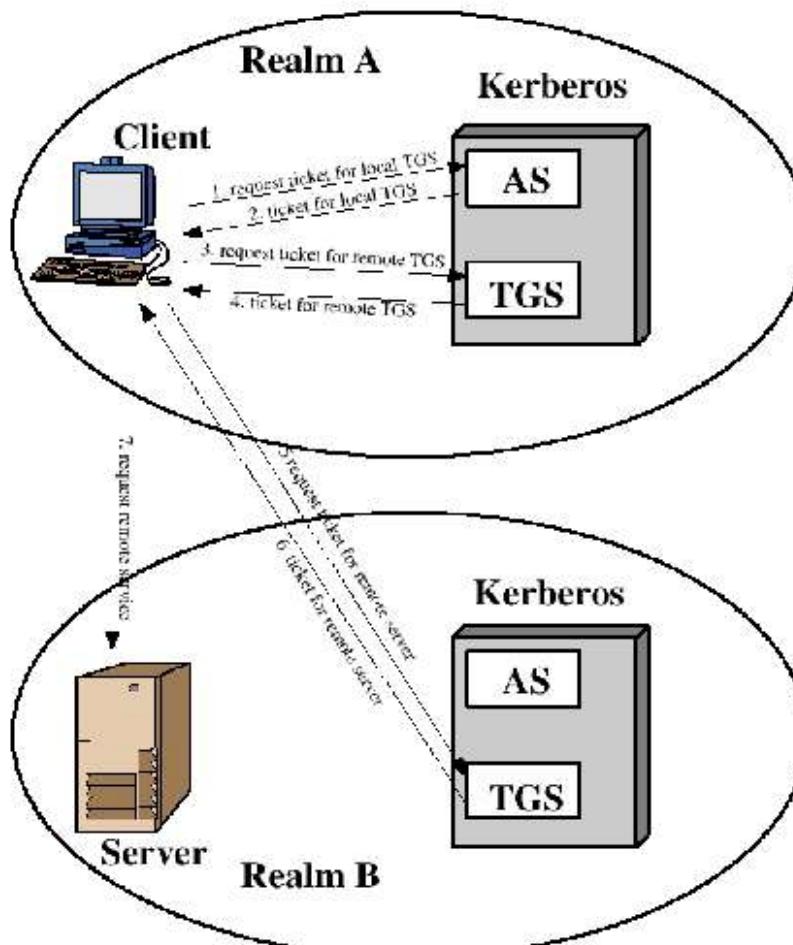


Figure 4.2 Request for Service in Another Realm

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

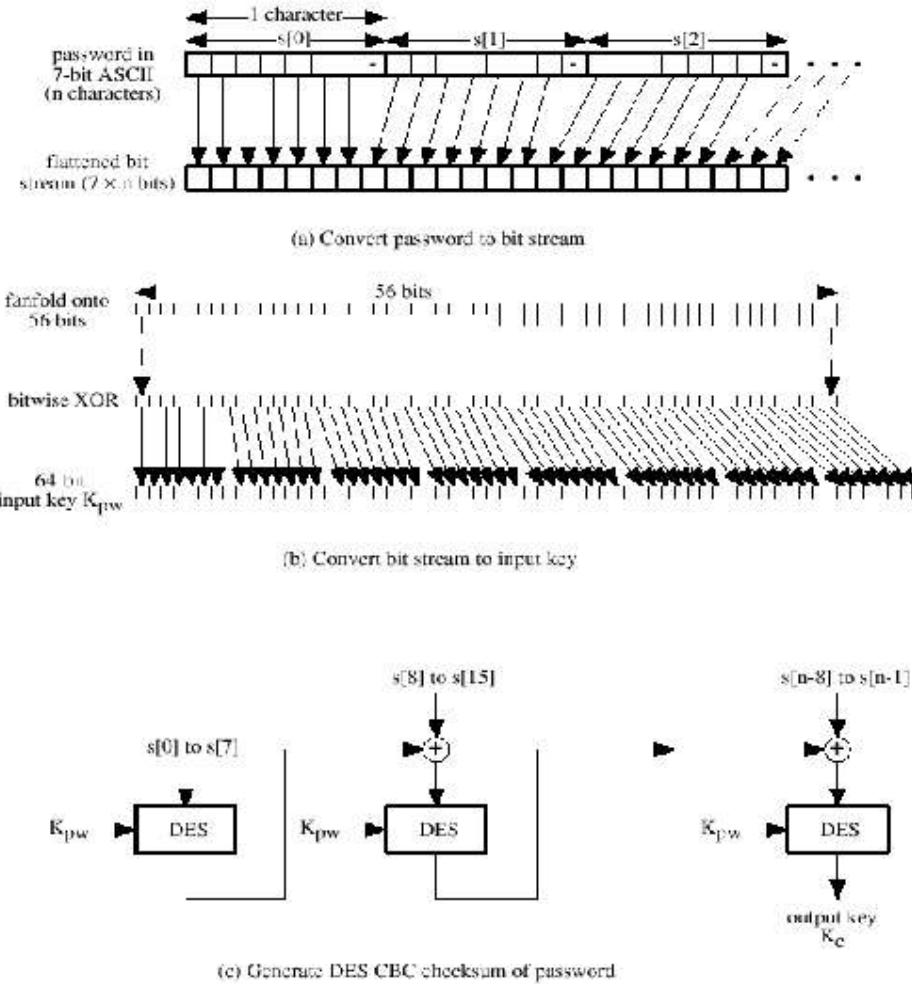


Figure 4.6 Generation of Encryption Key from Password

PCBC Mode

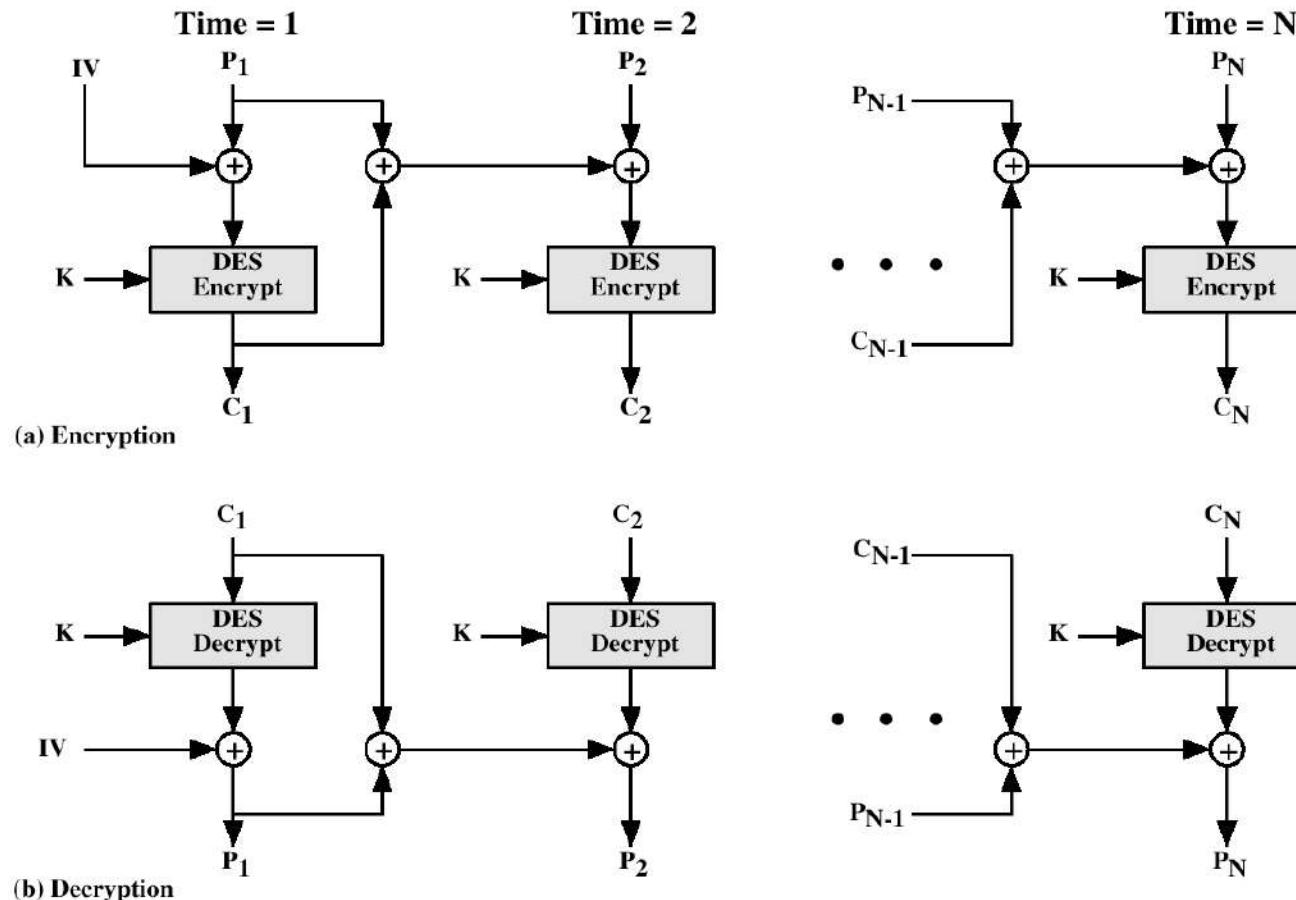


Figure 4.7 Propagating Cipher Block Chaining (PCBC) Mode

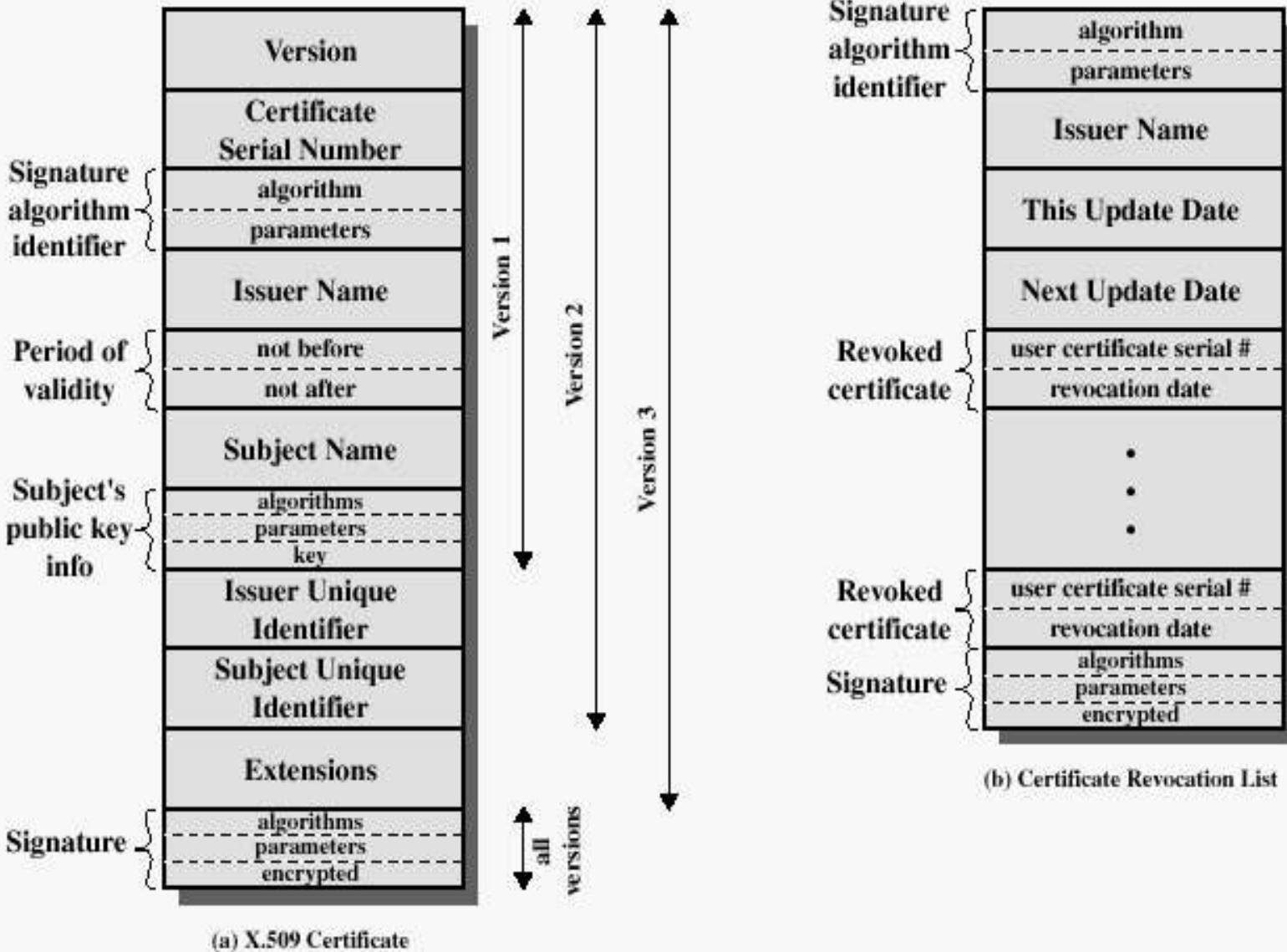
Kerberos - in practice

- 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

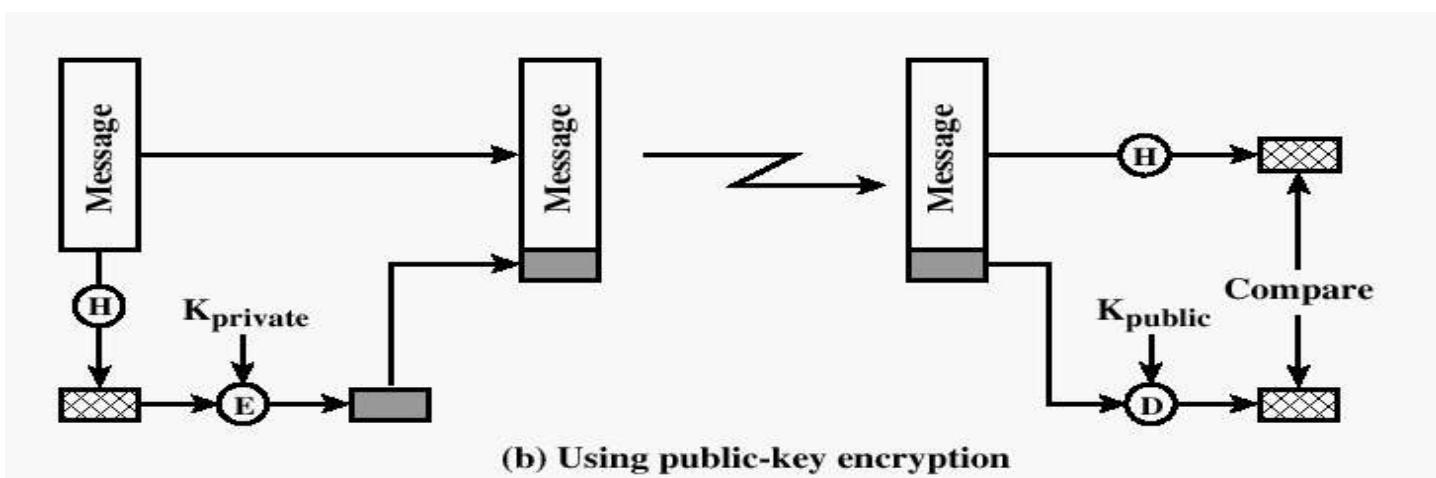
X.509 Authentication Service

- 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



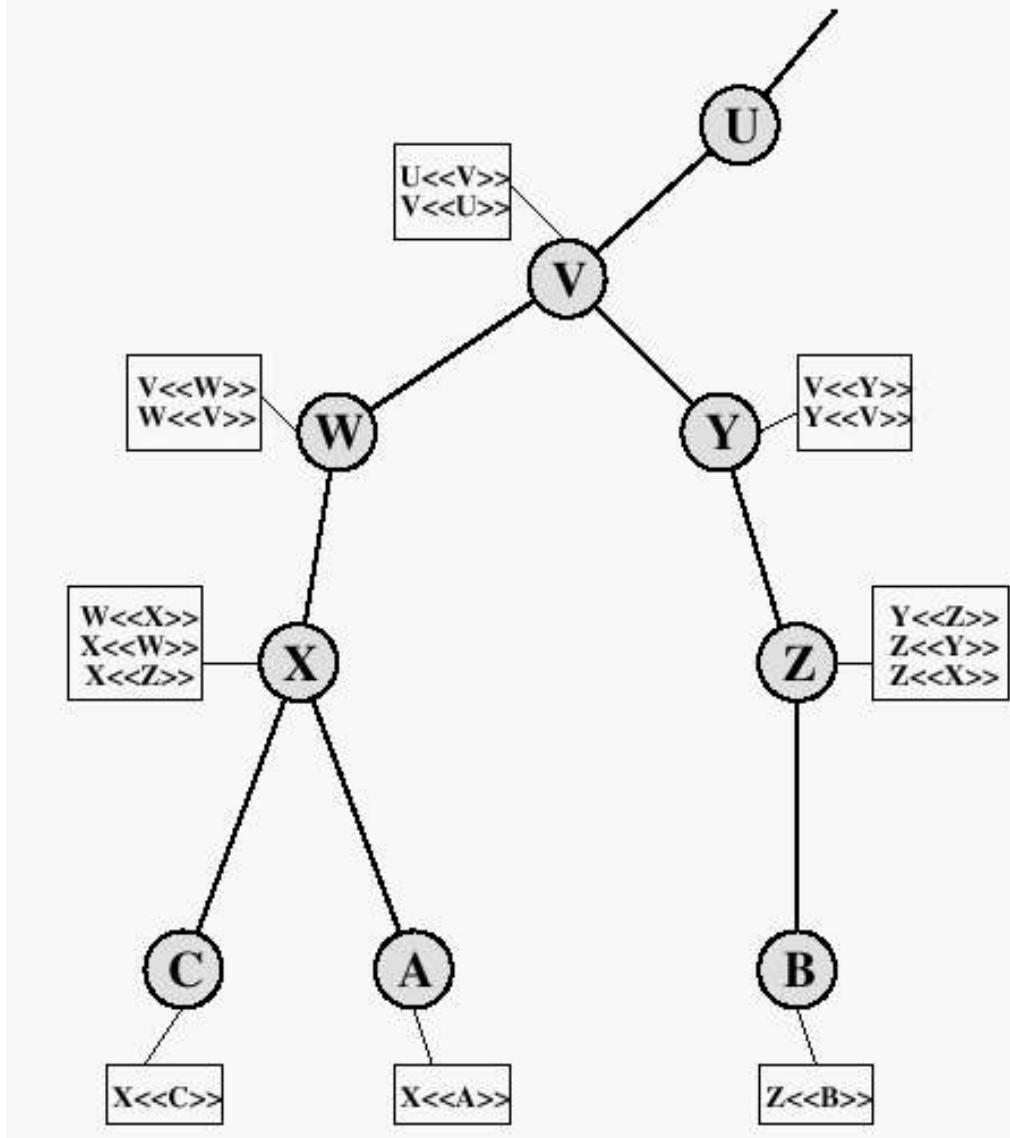
Typical Digital Signature Approach



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

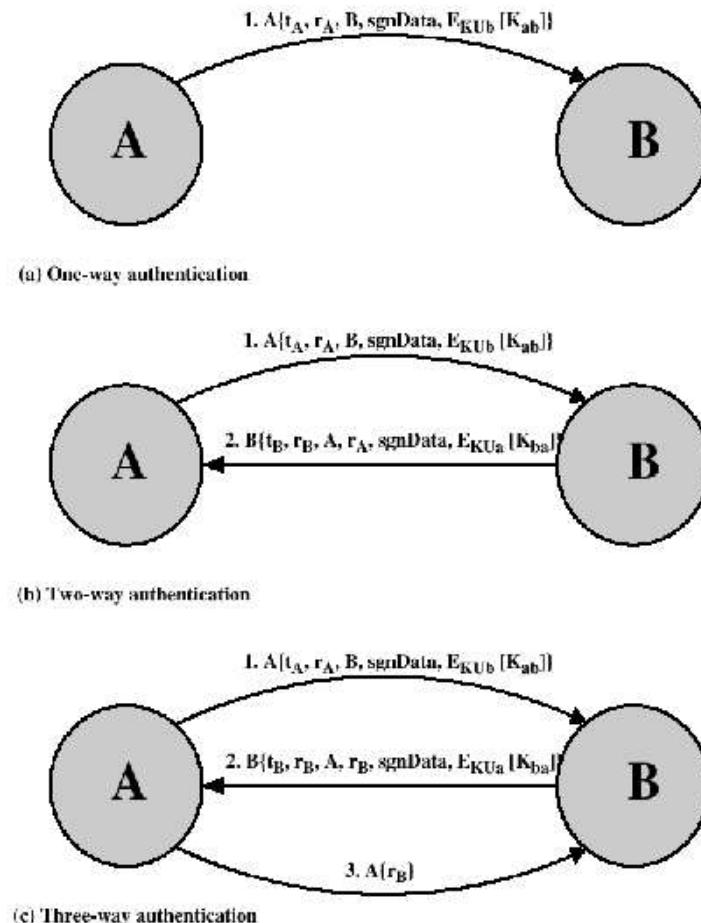


Figure 4.5 X.509 Strong Authentication Procedures

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/>

Lecture 10: Public Key Cryptography

Lecturer: Kurt Mehlhorn & He Sun

Today's lecture is about the Public Key Cryptography. We first discuss general ideas in designing cryptography protocols. Typically, the encryption scheme is a pair of algorithms, **encryption** algorithm and **decryption** algorithm. When sending a message, the sender first uses the **encryption** algorithm to encode the message, and send the encoded messages, called *ciphertext*, over the channel. Upon receiving a ciphertext, the receiver applies the decryption algorithm to decode the message, and receive the original message.

Some possible approaches in designing cryptographic protocols:

- Use a dictionary as the key
- Use a random sequence as the key
- Use a pseudorandom generator and a random seed as the key

1 Modular Arithmetic

Definition 1. *The number a is equivalent (congruent) to the number b modulo n , expressed by $a \equiv b \pmod{n}$, if a differs from b by an exact multiple of n .*

Lemma 2. *The following statements hold:*

- If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $a + c \equiv b + d \pmod{n}$.
- If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $ac \equiv bd \pmod{n}$.

Example. $321 \times 741 \equiv 1 \times 1 \equiv 1 \pmod{5}$.

Example. $715^{10000} \equiv 1 \pmod{7}$.

Example. $321^3 \equiv 6^3 \pmod{7} = 36 \times 6 \pmod{7} \equiv 6 \pmod{7}$.

Example. $320^{984} \equiv 1 \pmod{7}$

Let us look at the solutions of the example above. We first write down the binary expression of 984, i.e.

$$\begin{aligned} 984 &= 512 + 256 + 128 + 64 + 16 + 8 \\ &= 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3. \end{aligned}$$

Note that $320^{984} \equiv 5^{984} \pmod{7}$. Moreover, we have the following:

- $5^2 = 25 \equiv 4 \pmod{7}$
- $5^4 = 4 \times 4 \pmod{7} \equiv 2 \pmod{7}$

- $5^8 = 2 \times 2 \pmod{7} \equiv 4 \pmod{7}$
- $5^{16} = 4 \times 4 \pmod{7} = 2 \pmod{7}$
- $5^{32} \equiv 4 \pmod{7}$
- $5^{64} \equiv 2 \pmod{7}$
- $5^{128} \equiv 4 \pmod{7}$
- $5^{256} \equiv 2 \pmod{7}$
- $5^{512} \equiv 4 \pmod{7}$

Hence

$$\begin{aligned} 5^{984} &= 5^{512+256+128+64+16+8} \pmod{7} \\ &\equiv 4 \times 2 \times 4 \times 2 \times 2 \times 2 \times 4 \pmod{7} \\ &\equiv 1 \pmod{7} \end{aligned}$$

2 Fermat's Little Theorem

We call that n is divisible by m if $n = km$.

Theorem 3 (Fermat's Little Theorem). *If p is a prime number, then $a^p \equiv a \pmod{p}$ for all a .*

An alternative formulation of the Fermat's Little Theorem is as follows: If p is a prime number and a is any integer not divisible by p , then $a^{p-1} \equiv 1 \pmod{p}$.

Definition 4. *If $ab \equiv 1 \pmod{m}$, then b is called the multiplicative inverse of a modulo m .*

3 The Euclidean Algorithm

Given two integers r_0 and r_1 , the Euclidean Algorithm finds the greatest common divisor of r_0 and r_1 , denoted by $\gcd(r_0, r_1)$.

Before present the algorithm, we first look at the following lemma.

Lemma 5. $\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$

Theorem 6 (The Euclidean Algorithm). *Given two integers $0 < b < a$, we make a repeated application of the division algorithm to obtain a series of division equations, which eventually terminate with a zero remainder:*

$$\begin{aligned} a &= bq_1 + r_1, 0 < r_1 < b \\ b &= r_1q_2 + r_2, 0 < r_2 < r_1 \\ &\dots \\ r_{j-2} &= r_{j-1}q_j + r_j, 0 < r_j < r_{j-1} \\ r_{j-1} &= r_jq_{j+1} \end{aligned}$$

The greatest common divisor $\gcd(a, b)$ of a and b is r_j , the last nonzero remainder in the division process.

Now we show that the Euclidean Algorithm can be used to compute a multiplicative inverse.

Definition 7. If $ab \equiv 1 \pmod{p}$, then b is called the multiplicative inverse of a module p .

Theorem 8 (Multiplicative Inverse Algorithm). Given two integers $0 < b < a$, consider the Euclidean Algorithm equations which yield $\gcd(a, b) = r_j$. Rewrite all of these equations except the last one, by solving for the remainders:

$$\begin{aligned} r_1 &= a - bq_1 \\ r_2 &= b - r_1q_2 \\ r_3 &= r_1 - r_2q_3 \\ &\dots \\ r_{j-1} &= r_{j-3} - r_{j-2}q_{j-1} \\ r_j &= r_{j-2} - r_{j-1}q_j \end{aligned}$$

Then, in the last of these equations, $r_j = r_{j-2} - r_{j-1}q_j$, replace r_{j-1} with its expression in terms of r_{j-3} and r_{j-2} from the equation immediately above it. Continue this process successively, replacing r_{j-2}, r_{j-3}, \dots , until we obtain the final equation

$$r_j = ax + by,$$

where x and y are integers. In the special case that $\gcd(a, b) = 1$, the integer equation reads

$$1 = ax + by.$$

Therefore we deduce

$$1 \equiv by \pmod{a}$$

so that the residue of y is the multiplicative inverse of b , mod a .

4 The RSA Algorithm

- p, q are two prime numbers
- Let $n = p \cdot q$
- Pick a positive integer r that has no common factor with $(p - 1) \cdot (q - 1)$
- Find a multiplicative inverse of r modulo $(p - 1) \cdot (q - 1)$, i.e. we find a number s such that $rs \equiv 1 \pmod{(p - 1) \cdot (q - 1)}$

Encryption: We need to know n, r . Assume that the message is $x \leq n$. Let

$$y \triangleq x^r \pmod{n}.$$

The pair of values n, r are **public encryption key**. This information is publicly available, and anyone can compute y if they are given x .

Decryption: To decrypt, you need to know s , the **private decryption key**. With the value of s , you simply compute

$$z \triangleq y^s \pmod{n} \equiv x^{rs} \equiv x \pmod{n}.$$

That is, you need to know s to decrypt. Now s is the multiplicative inverse of r modulo $(p-1)(q-1)$. The outsiders know r , and if they knew $(p-1)(q-1)$, then it would be easy (with the Euclidean Algorithm) to compute s . But they do not know $(p-1)(q-1)$. They know n , which is equal to pq , but they do not have n factored into p and q . To find $(p-1)(q-1)$, they need to know the prime factors p and q of n , and factoring large numbers is not easy.