Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

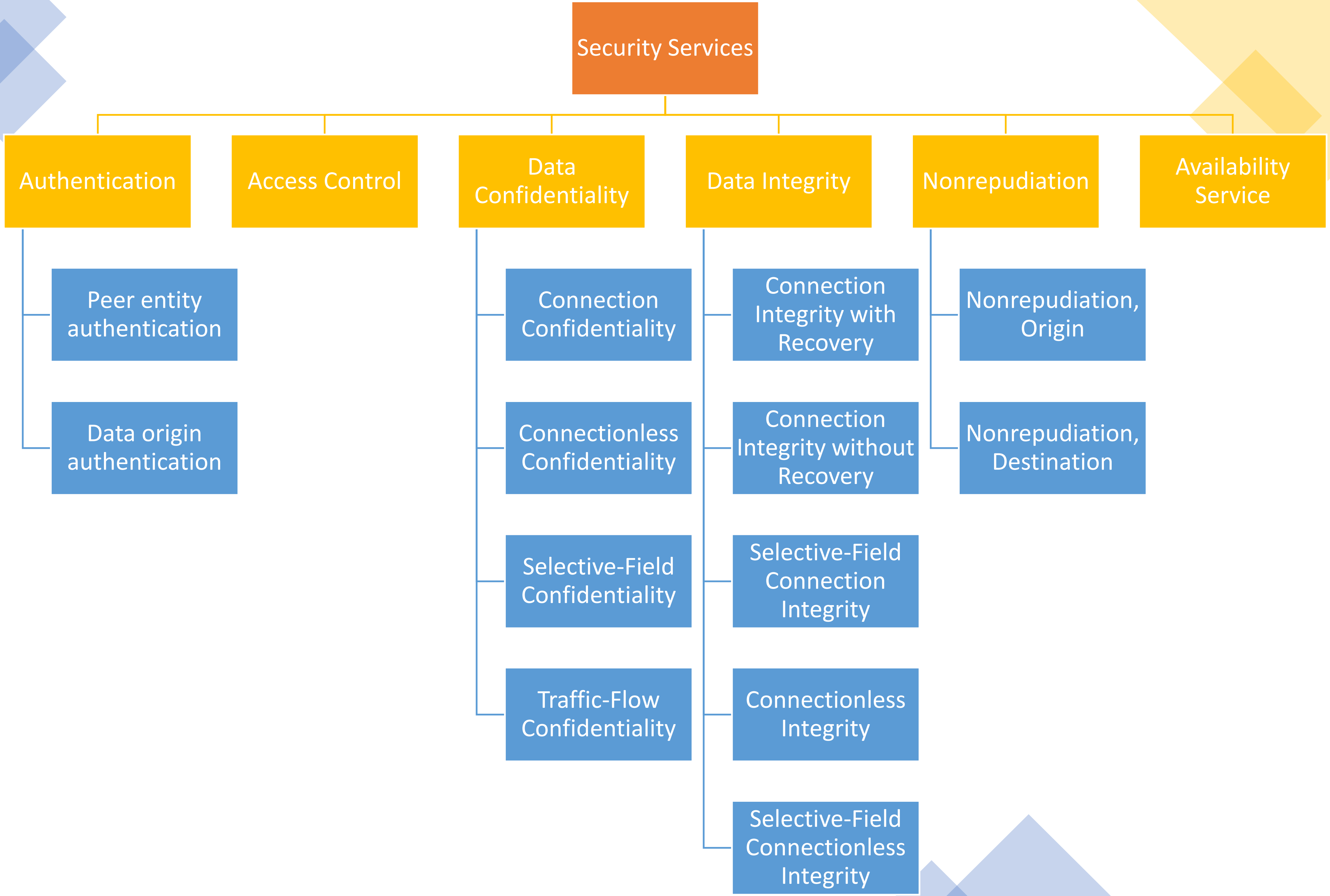
Topics

Security Services

Security Mechanisms

A Model For Network Security

Security definitions


Security Services
Authentication
Access Control
Data Confidentiality
Data Integrity
Nonrepudiation
Availability Service
Peer entity authentication
Data origin authentication
Connection Confidentiality
Connectionless Confidentiality
Selective-Field Confidentiality
Traffic-Flow Confidentiality
Connection Integrity with Recovery
Connection Integrity without Recovery
Selective-Field Connection Integrity
Connectionless Integrity
Selective-Field Connectionless Integrity
Nonrepudiation, Origin
Nonrepudiation, Destination

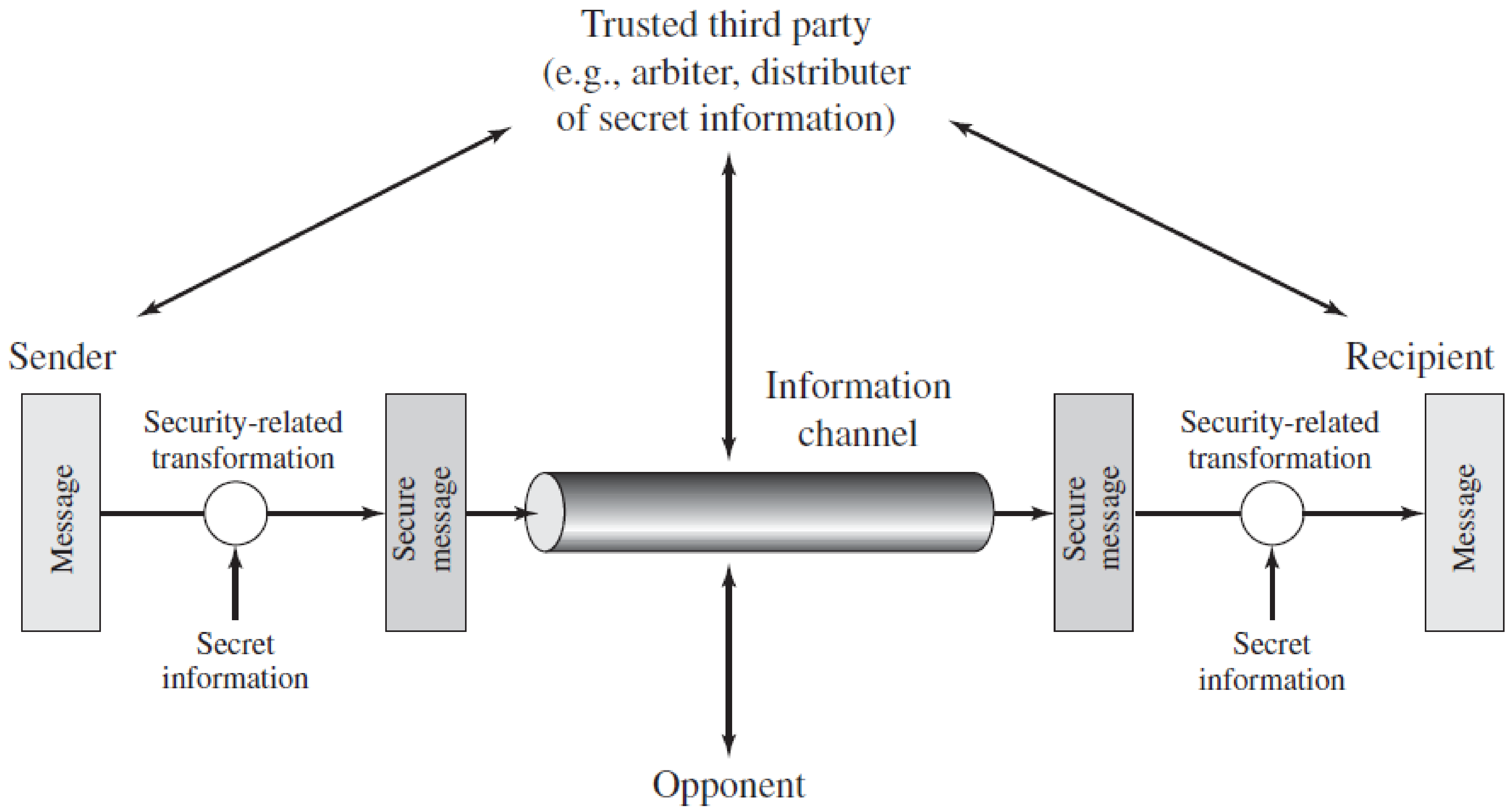
Security Mechanisms

Specific Security Mechanisms	Encipherment
	Digital Signature
	Access Control
	Data Integrity
	Authentication Exchange
	Traffic Padding
	Routing Control
	Notarization
Pervasive Security Mechanisms	Trusted Functionality
	Security Label
	Event Detection
	Security Audit Trail
	Security Recovery

Table 1.4 Relationship Between Security Services and Mechanisms

	Mechanism							
Service	**Encipherment**	**Digital Signature**	**Access Control**	**Data Integrity**	**Authentication Exchange**	**Traffic Padding**	**Routing Control**	**Notarization**
Peer Entity Authentication	Y	Y			Y			
Data Origin Authentication	Y	Y						
Access Control			Y					
Confidentiality	Y						Y	
Traffic Flow Confidentiality	Y					Y	Y	
Data Integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

A Model For Network Security



Model for Network Security

A Model For Network Security

This general model shows that there are four basic tasks in designing a particular security service:

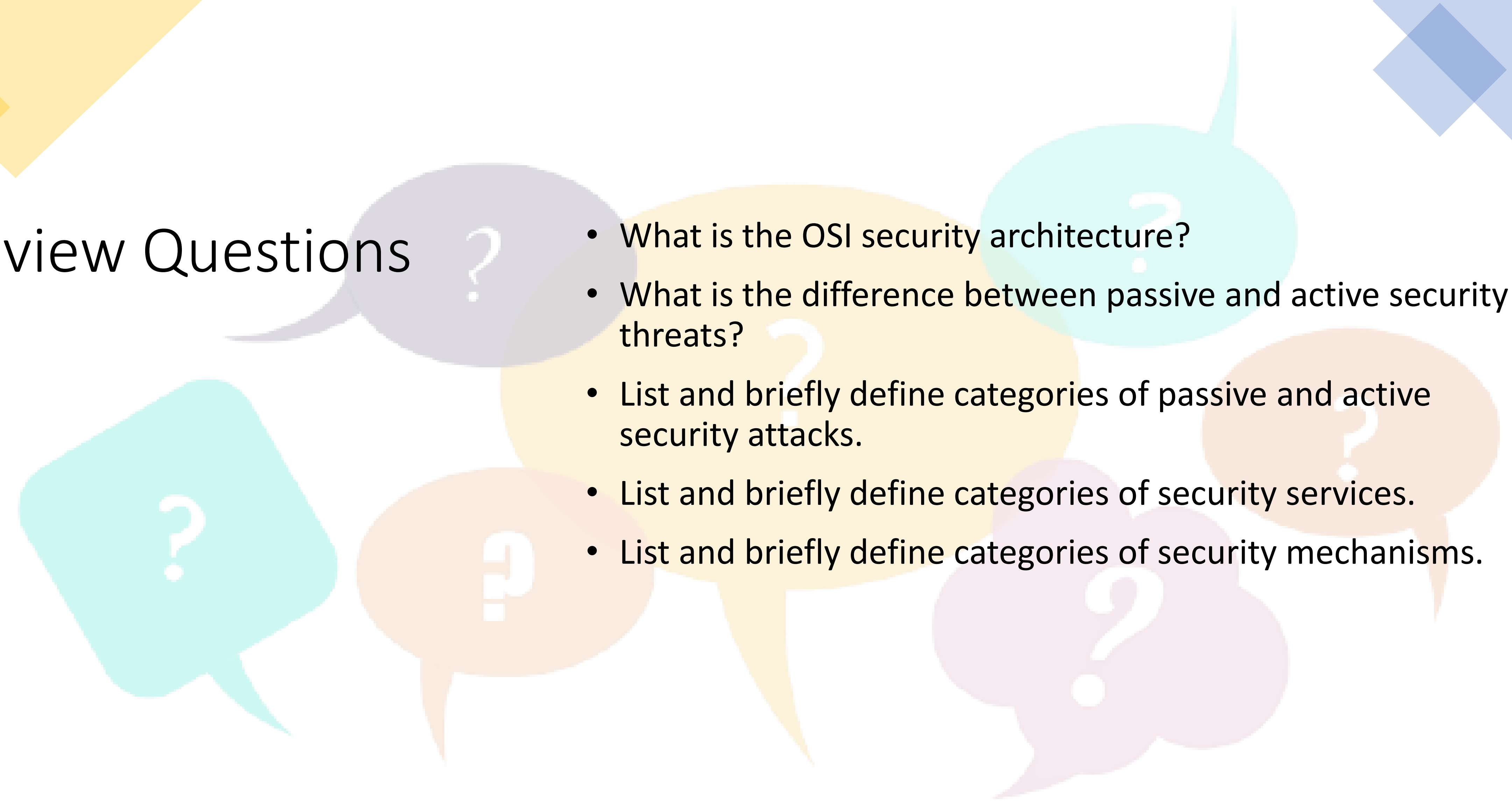
1- Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

2- Generate the secret information to be used with the algorithm.

3- Develop methods for the distribution and sharing of the secret information.

4- Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Review Questions



- What is the OSI security architecture?
- What is the difference between passive and active security threats?
- List and briefly define categories of passive and active security attacks.
- List and briefly define categories of security services.
- List and briefly define categories of security mechanisms.

Symmetric Ciphers

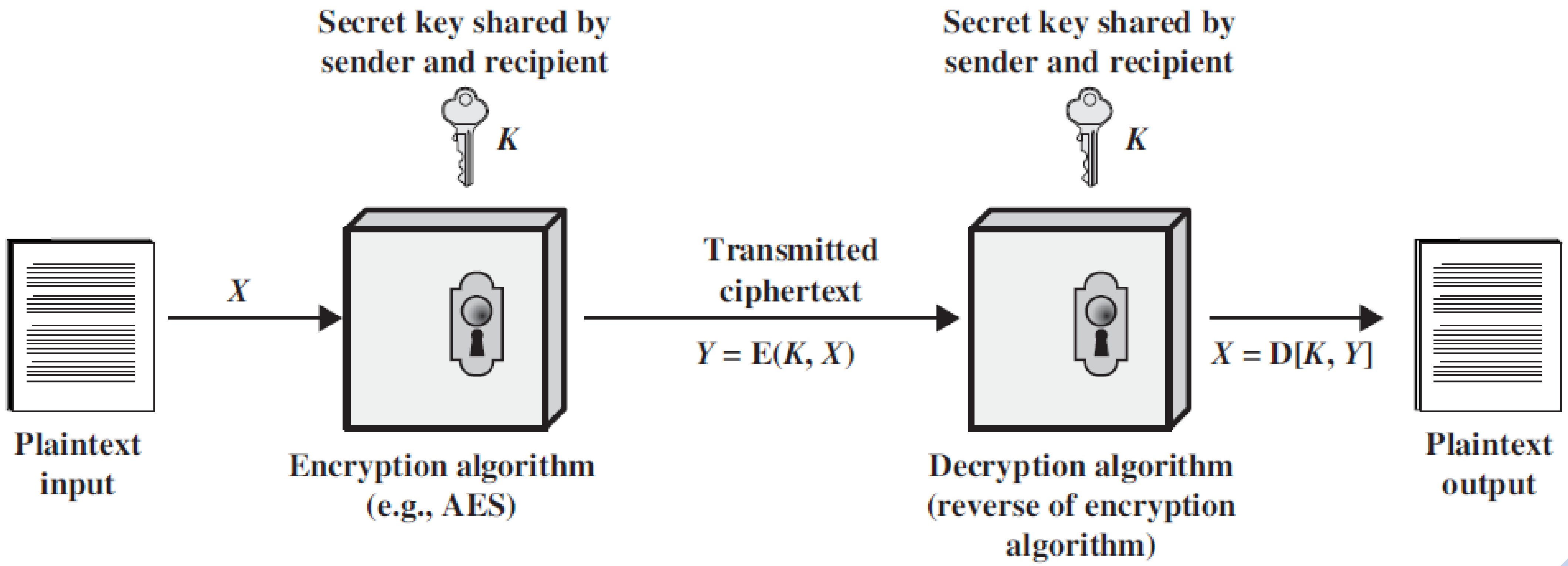
Classical Encryption Techniques

Modern Encryption Techniques

Terms

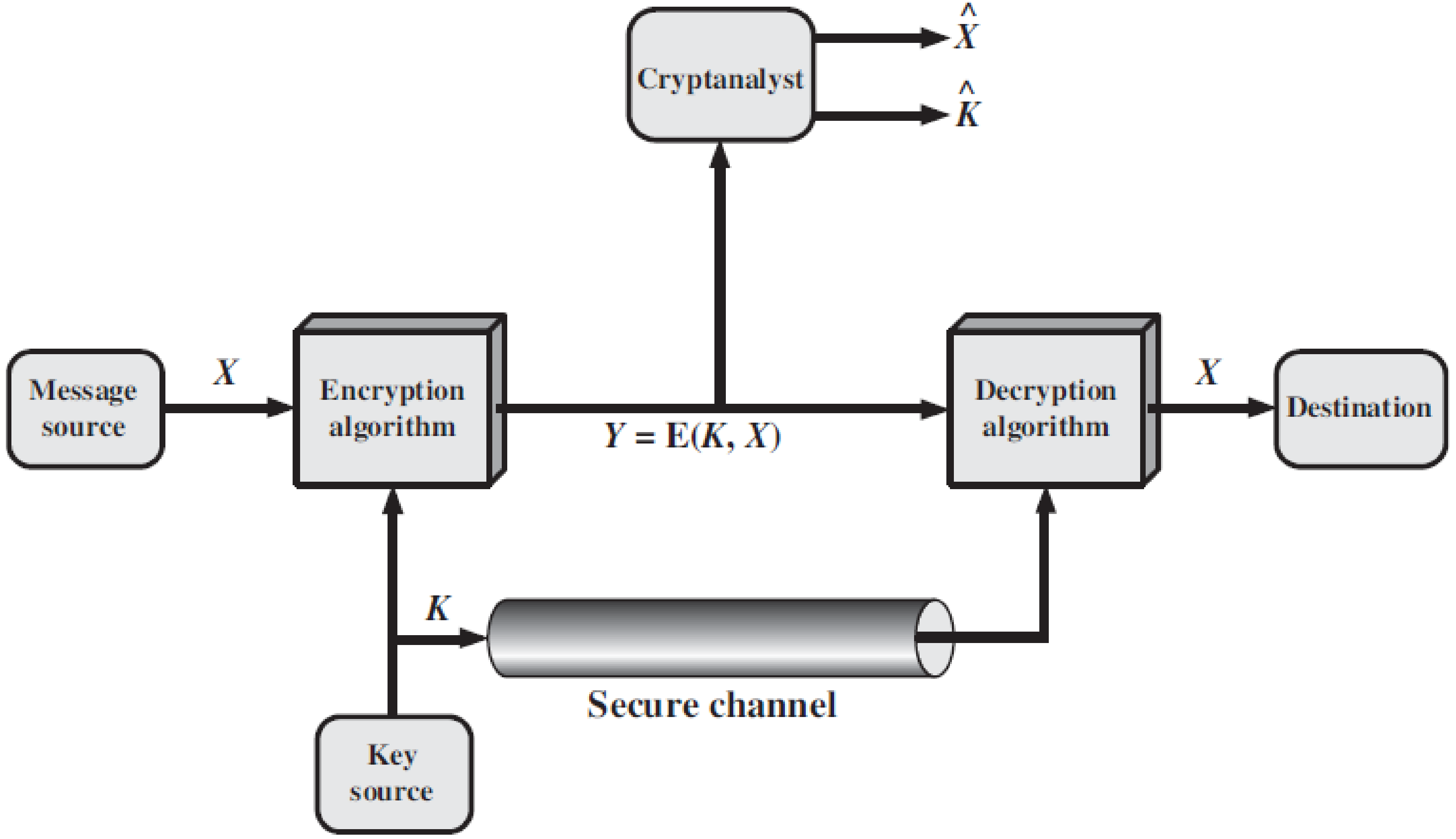
- **Plaintext:** the original message
- **Ciphertext**: the coded message
- **Enciphering** or **Encryption**: The process of converting from plaintext to ciphertext
- **Deciphering** or **Decryption**: The process of restoring the plaintext from the ciphertext
- **Cryptography**: The area of studying all schemes used for encryption constitute
- **Cryptographic system** or a **Cipher**: Cryptography scheme
- **Cryptanalysis**: Techniques used for deciphering a message without any knowledge of the enciphering details.
- **Cryptology**: The areas of cryptography and cryptanalysis

Simplified Model of Symmetric Encryption



Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
$Y = E(K, X)$
$X = D[K, Y]$
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Model of Symmetric Cryptosystem



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

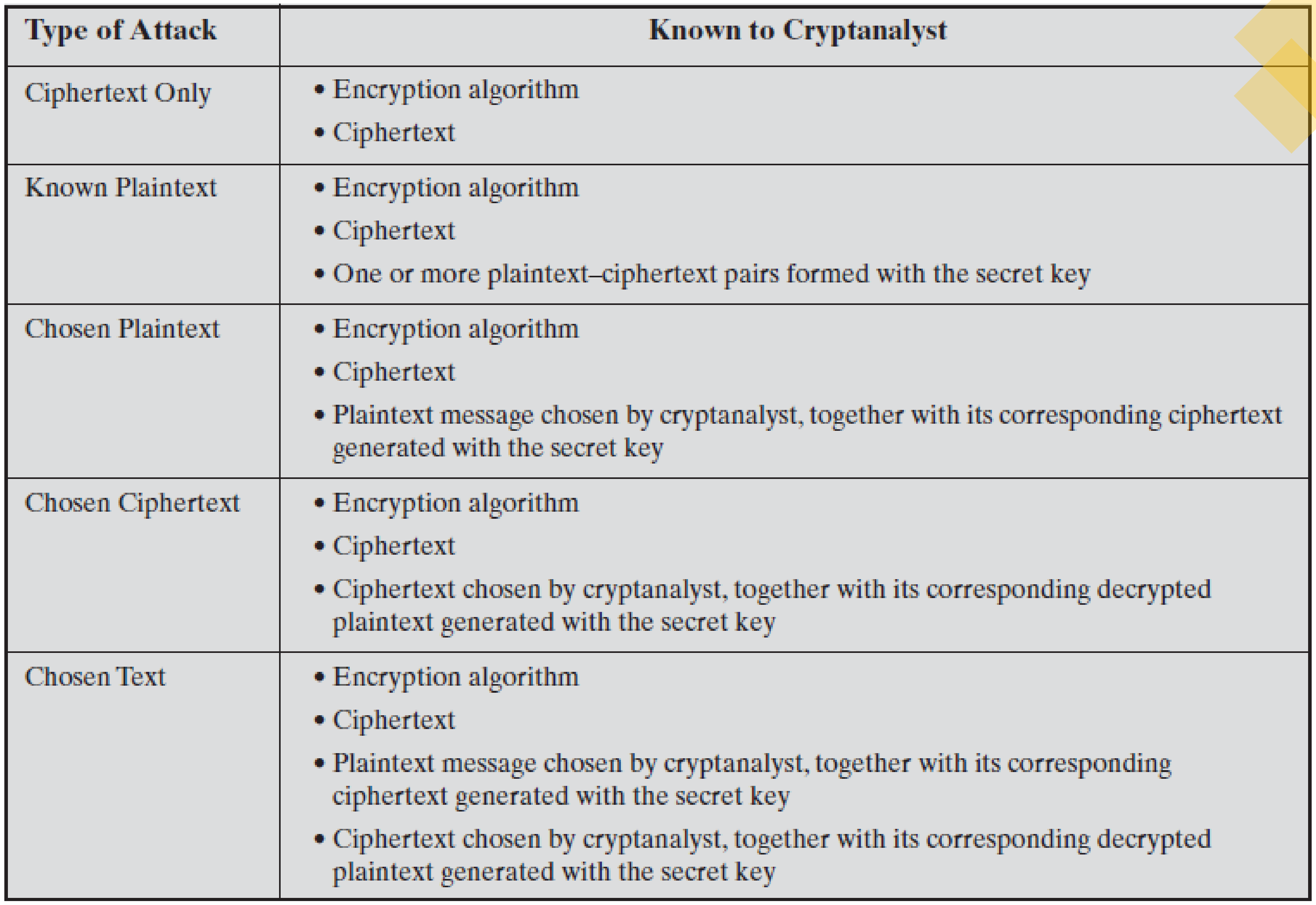
Cryptanalysis and Brute-Force Attack

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.

Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained

Cryptanalysis

Table 2.1 Types of Attacks on Encrypted Messages



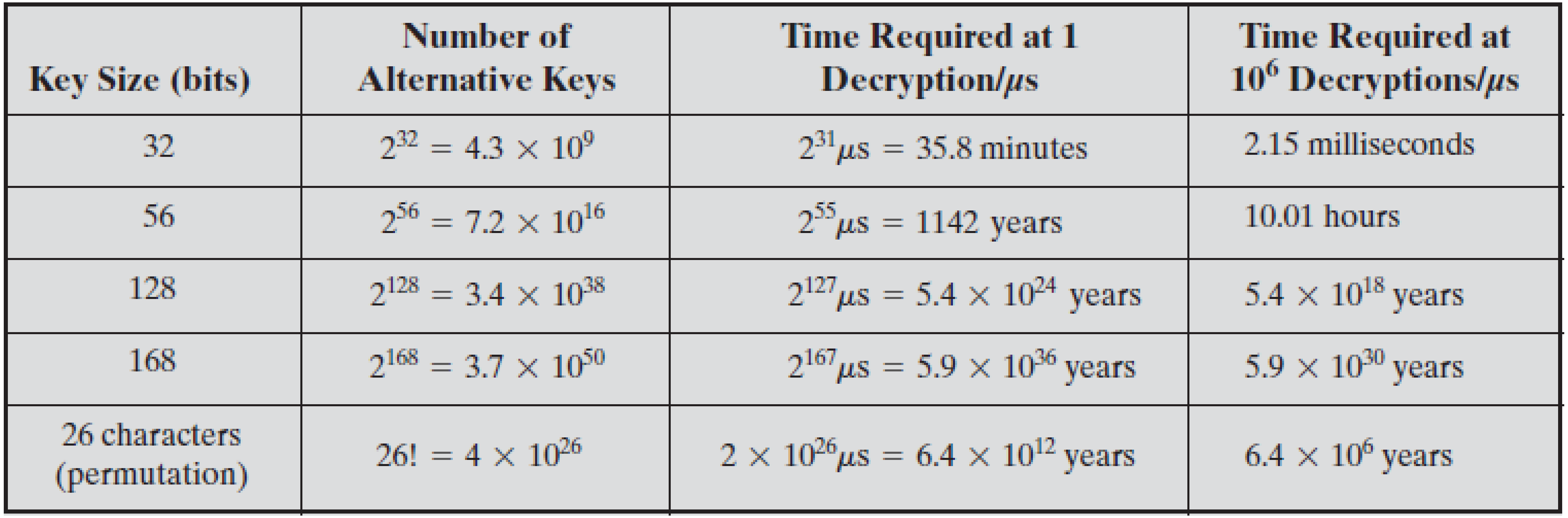
Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• 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

Cryptanalysis

- 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
- An encryption scheme is said to be **computationally secure** if either of the two criteria are met:
  - The **<u>cost</u>** of breaking the cipher exceeds the value of the encrypted information.
  - The **<u>time</u>** required to break the cipher exceeds the useful lifetime of the information.

Brute-force attack

Table 2.2 Average Time Required for Exhaustive Key Search



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

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

- Caesar Cipher
- Playfair Cipher
- Vigen`ere Cipher
- One-Time Pad

Classical encryption techniques

- Substitution
- Transposition
- Substitution and transposition

Caesar Cipher

- The earliest known, and the simplest, use of a substitution cipher was by Julius Caesar

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

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

Caesar Cipher

caser cipher considered to be a special type of the monoalphbatic cipher where:

$$C = \mathrm{E}(k, p) = (p + k) \bmod 26$$

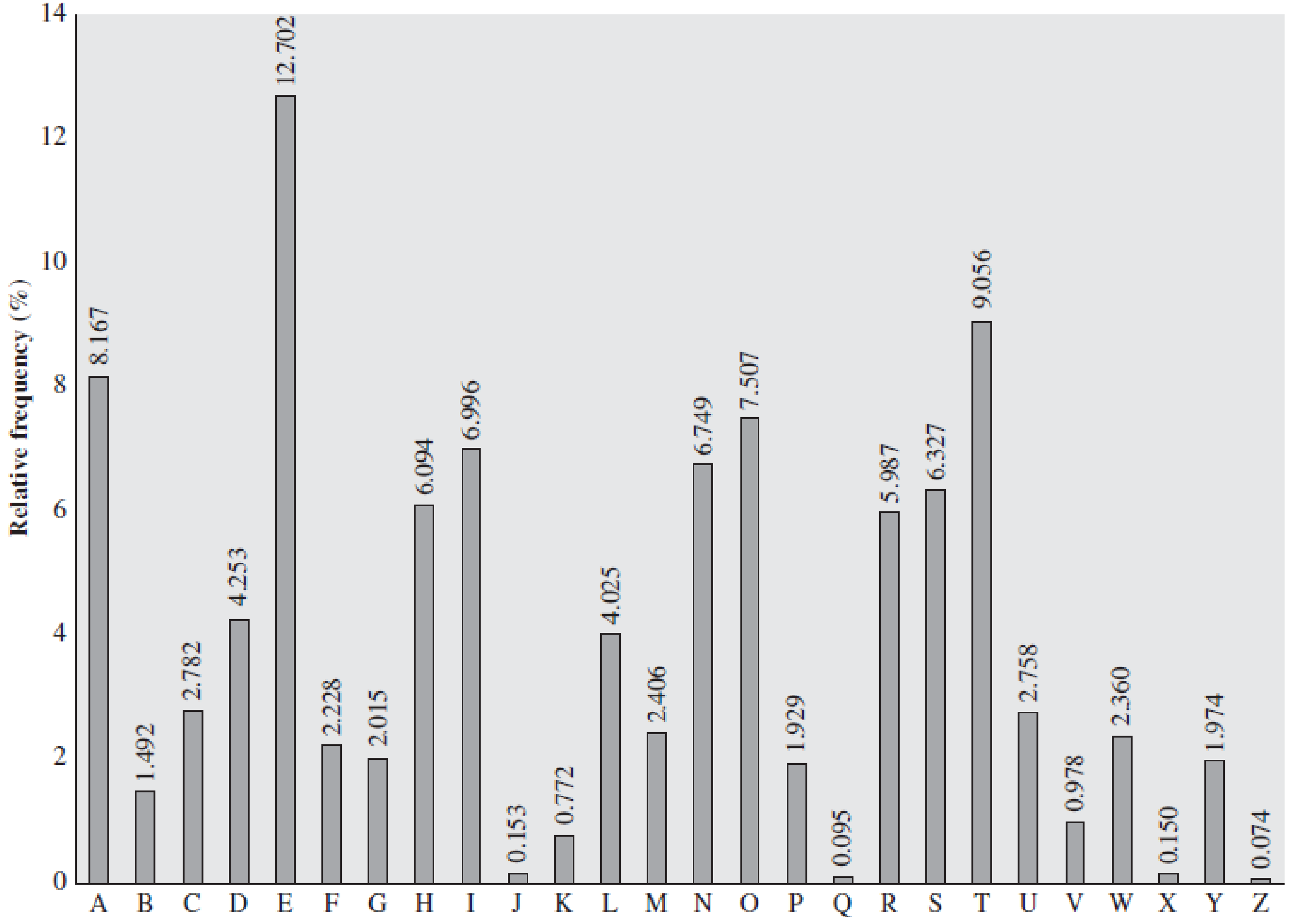
And

$$p = \mathrm{D}(k, C) = (C - k) \bmod 26$$

Three important characteristics of this problem enabled us to use a bruteforce cryptanalysis:

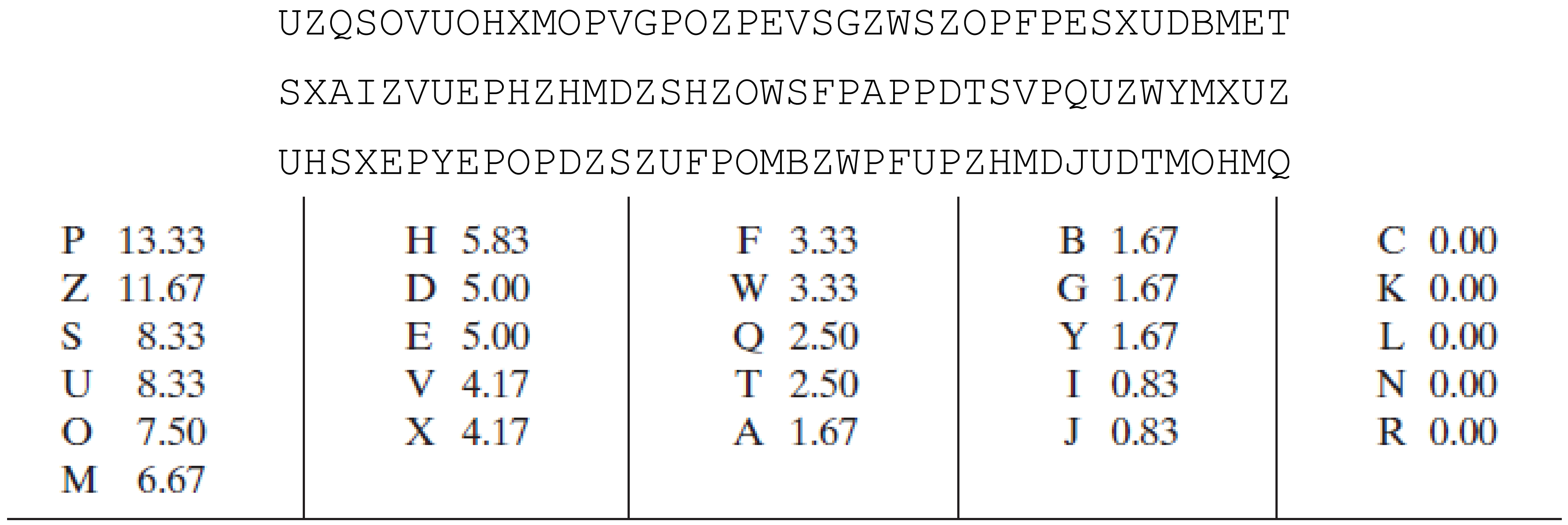
1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

Relative Frequency of Letters in English Text



Monoalphabetic Ciphers

- **Permutation**
- **digrams**



```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMET
SXAIZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZ
UHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

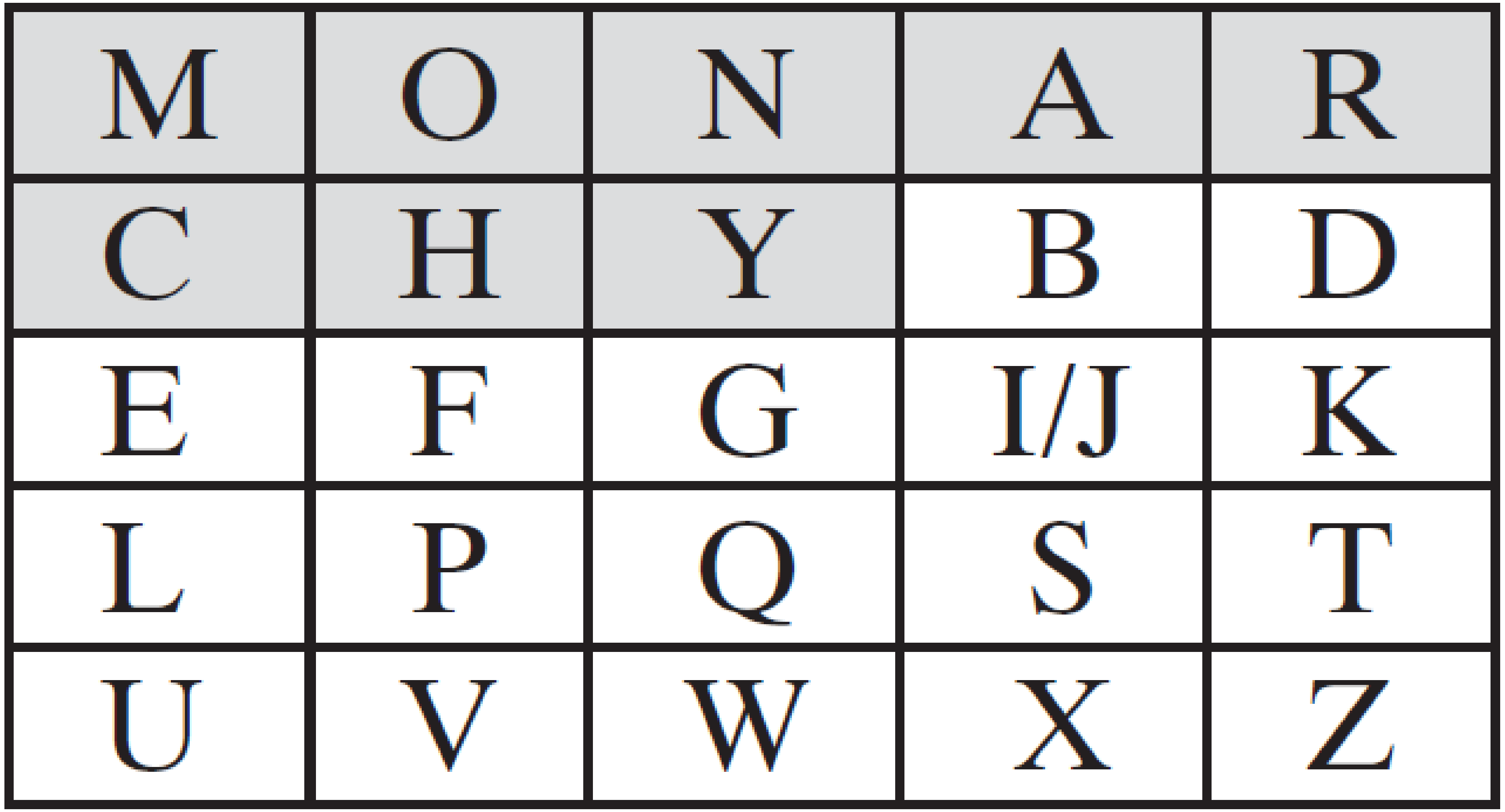
P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

Monoalphabetic Ciphers

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMET
SXAIZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZ
UHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

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

Playfair Cipher



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

Playfair Cipher

Plaintext is encrypted two letters at a time, according to the following rules:
1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
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, ar is encrypted as RM.
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, mu is encrypted as CM.
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, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

Polyalphabetic Ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message

- A set of related monoalphabetic substitution rules is used.
- A key determines which particular rule is chosen for a given transformation.

Vigen`ere Cipher

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is ***deceptive***, the message **“we are discovered save yourself”** is encrypted as

key: **deceptivedeceptivedeceptive**

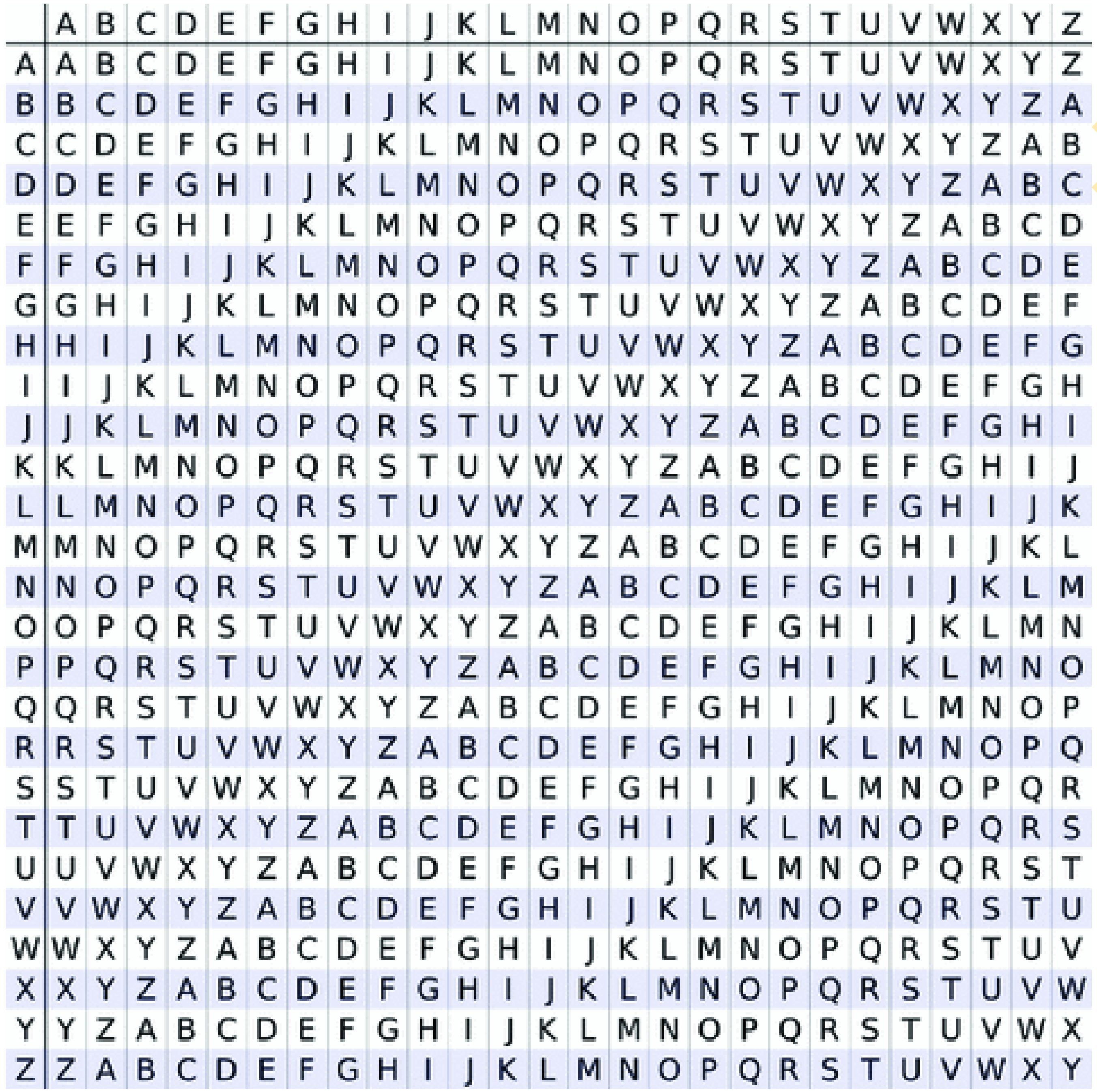
plaintext: **wearediscoveredsaveyourself**

ciphertext: **ZIC$\underline{\text{VTW}}$QNGRZG$\underline{\text{VTW}}$AVZHCQYGLMGJ**

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

Vigen`ere Cipher



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

Vigen`ere Cipher

The periodic nature of the keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself. Vigenère proposed what is referred to as an **autokey system**, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

```
key:        deceptivewearediscoveredsav
plaintext:  wearediscoveredsaveyourself
ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA
```

One-Time Pad

using a random key that is **as long as the message**, so that the key need not be repeated.

Each new message requires a new key of the same length as the new message.

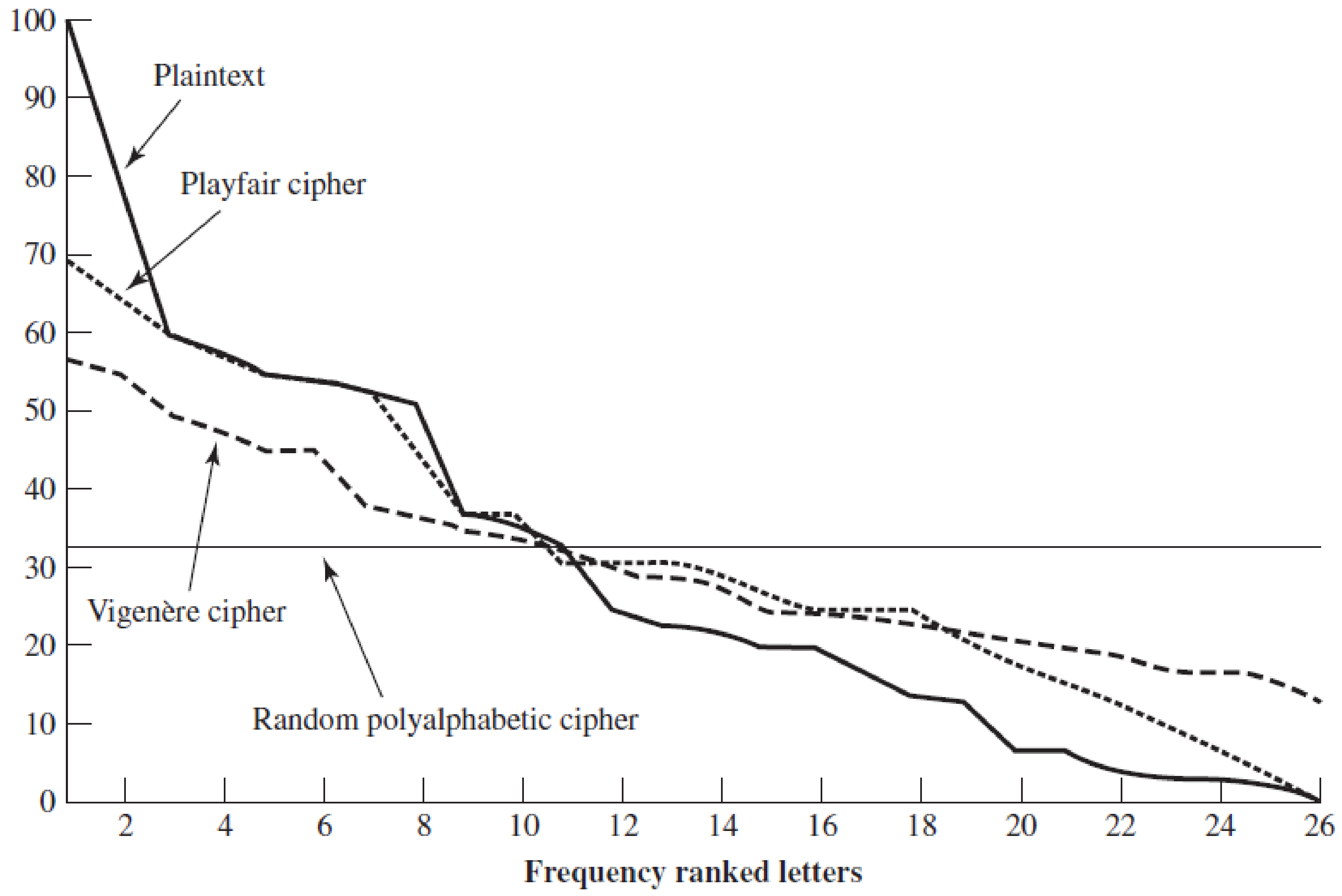
It produces **random output** that bears no statistical relationship to the plaintext. Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.

One-Time Pad

We now show two different decryptions using two different keys:

```
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plaintext:  mr mustard with the candlestick in the hall


ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        mfugpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt
 plaintext:  miss scarlet with the knife in the library
```

Relative Frequency of Occurrence of Letters



Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Transposition Techniques

Rotor Machines

Steganography

Transposition Techniques (rail fence)

- A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters:

“meet me after the toga party”

`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`

Transposition Techniques (rail fence)

- 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. For example,

```
Key:        4 3 1 2 5 6 7
Plaintext:  a t t a c k p
            o s t p o n e
            d u n t i l t
            w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Transposition Techniques (rail fence)

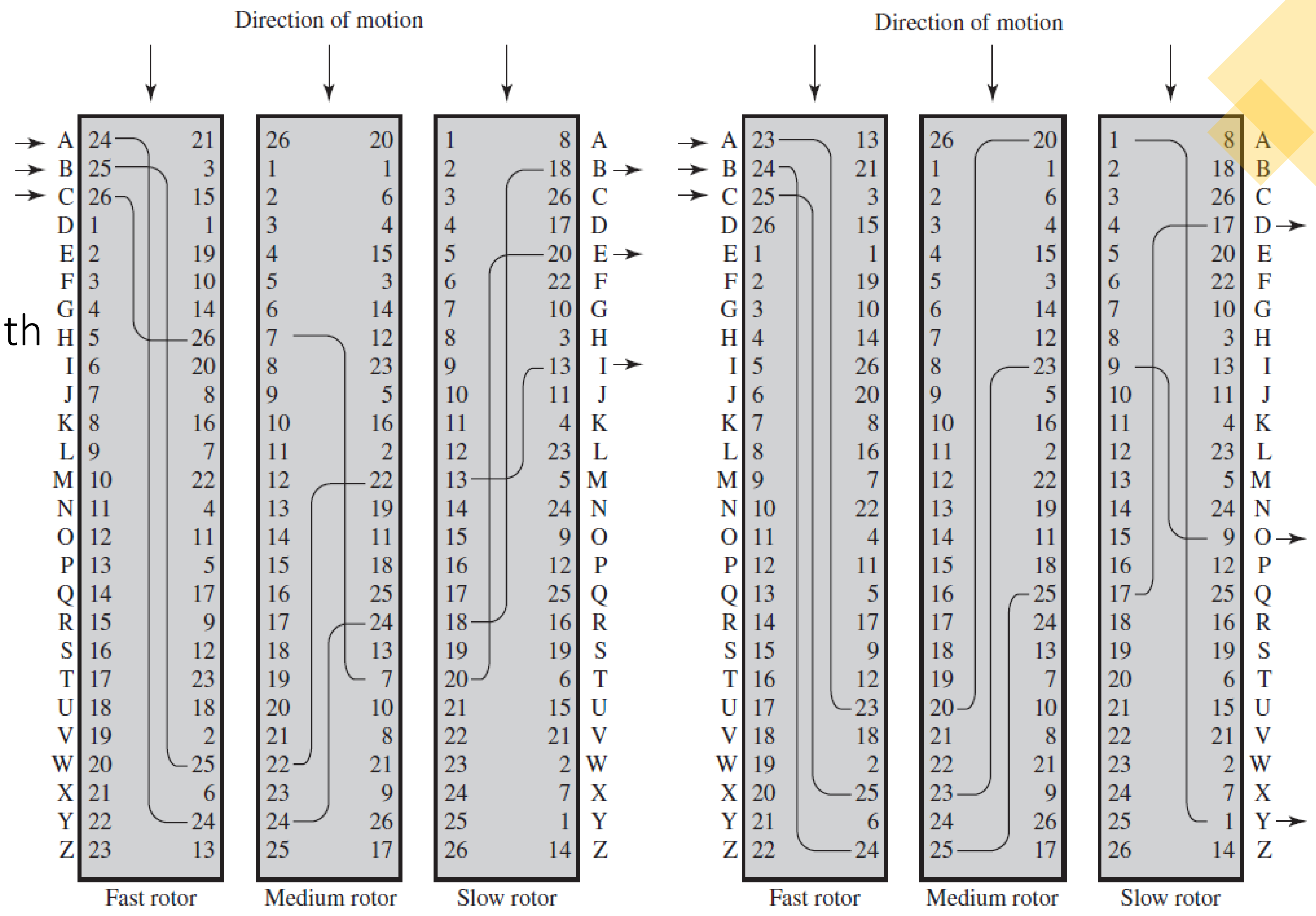
- The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm

```
        Key:    4 3 1 2 5 6 7
        Input:  t t n a a p t
                m t s u o a o
                d w c o i x k
                n l y p e t z
    Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

Rotor Machines

- The machine consists of a set of independently rotating cylinders through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin.

Three-Rotor Machine with Wiring Represented by Numbered Contacts

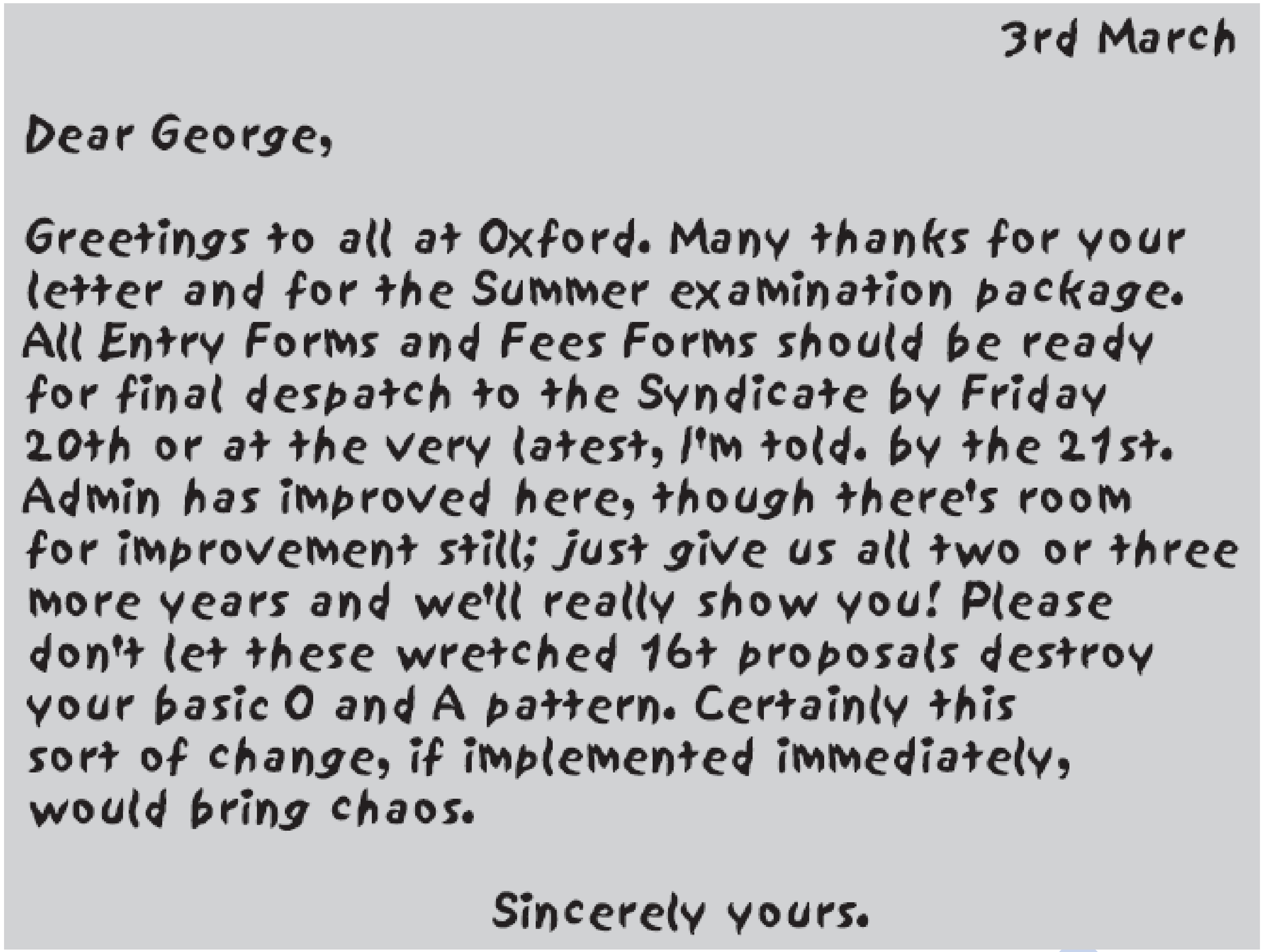


(a) Initial setting

(b) Setting after one keystroke

Steganography

- 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.
- The sequence of first letters of each word of the overall message spells out the hidden message
- **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.



3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told. by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours.

Steganography

- the Kodak Photo CD format’s maximum resolution is 2048 X 3072 pixels, with each pixel containing 24 bits of RGB color information. The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image. The result is that you can hide a 2.3-megabyte message in a single digital snapshot. There are now a number of software packages available that take this type of approach to steganography.

Steganography

- Steganography has a number of drawbacks when compared to encryption:
  - It requires a lot of overhead to hide a relatively few bits of information
  - If once the system is discovered, it becomes virtually worthless.
- Benefits:
  - Alternatively, a message can be first encrypted and then hidden using steganography
  - it can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered.

Steganography

- CMD
- Echo “secret” >> “image_name.jpg”

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

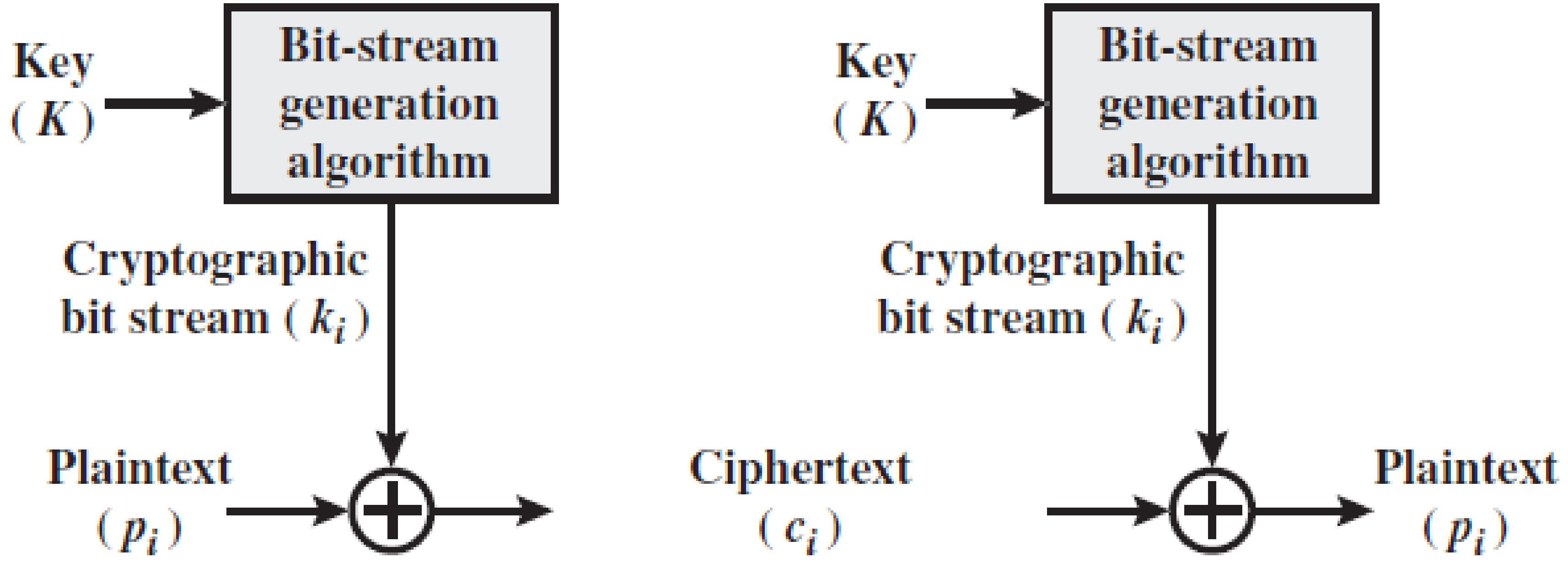
Topics

The Feistel Cipher

Data Encryption Standard (DES)

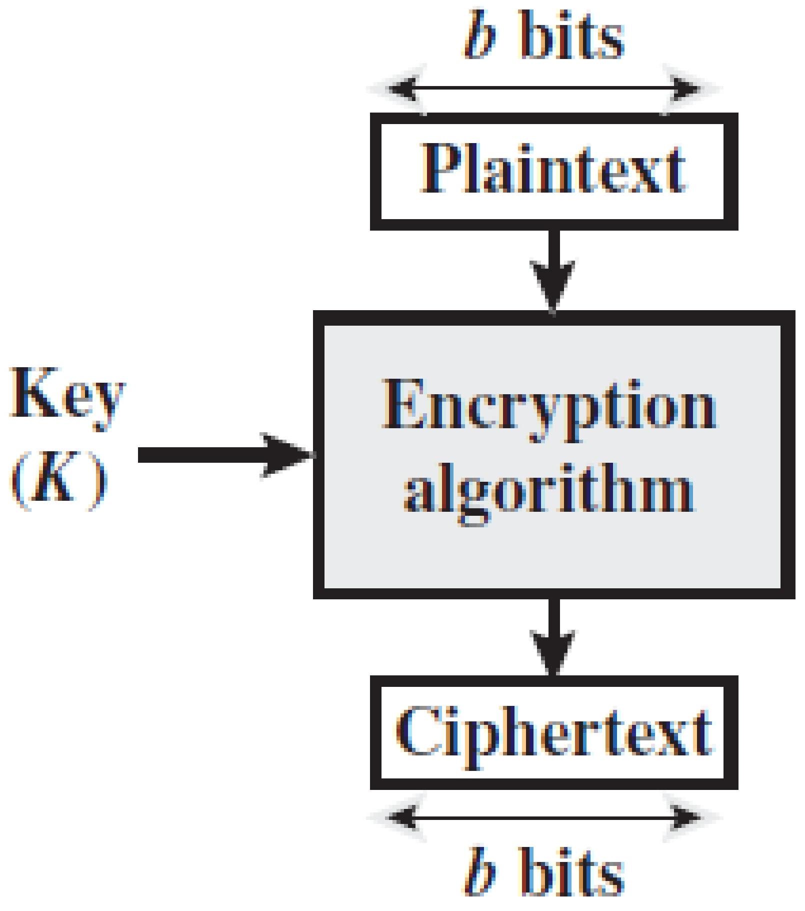
Stream Ciphers and Block Ciphers

- A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher. If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream


Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext
(p_i)
$\oplus$
Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext
(c_i)
$\oplus$
Plaintext
(p_i)

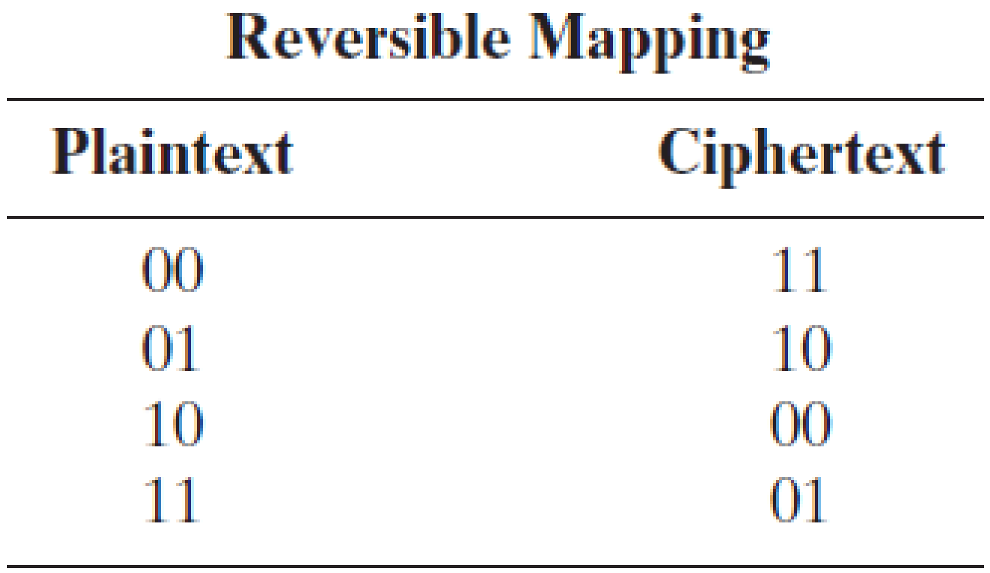
Stream Ciphers and Block Ciphers

- A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used



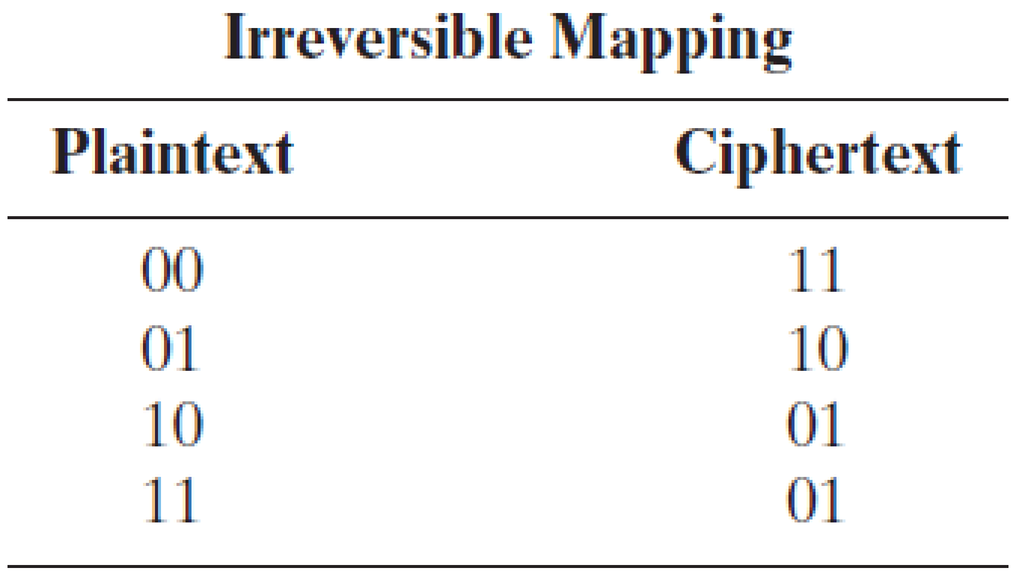
Motivation for the Feistel Cipher Structure

- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits
- There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block



Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

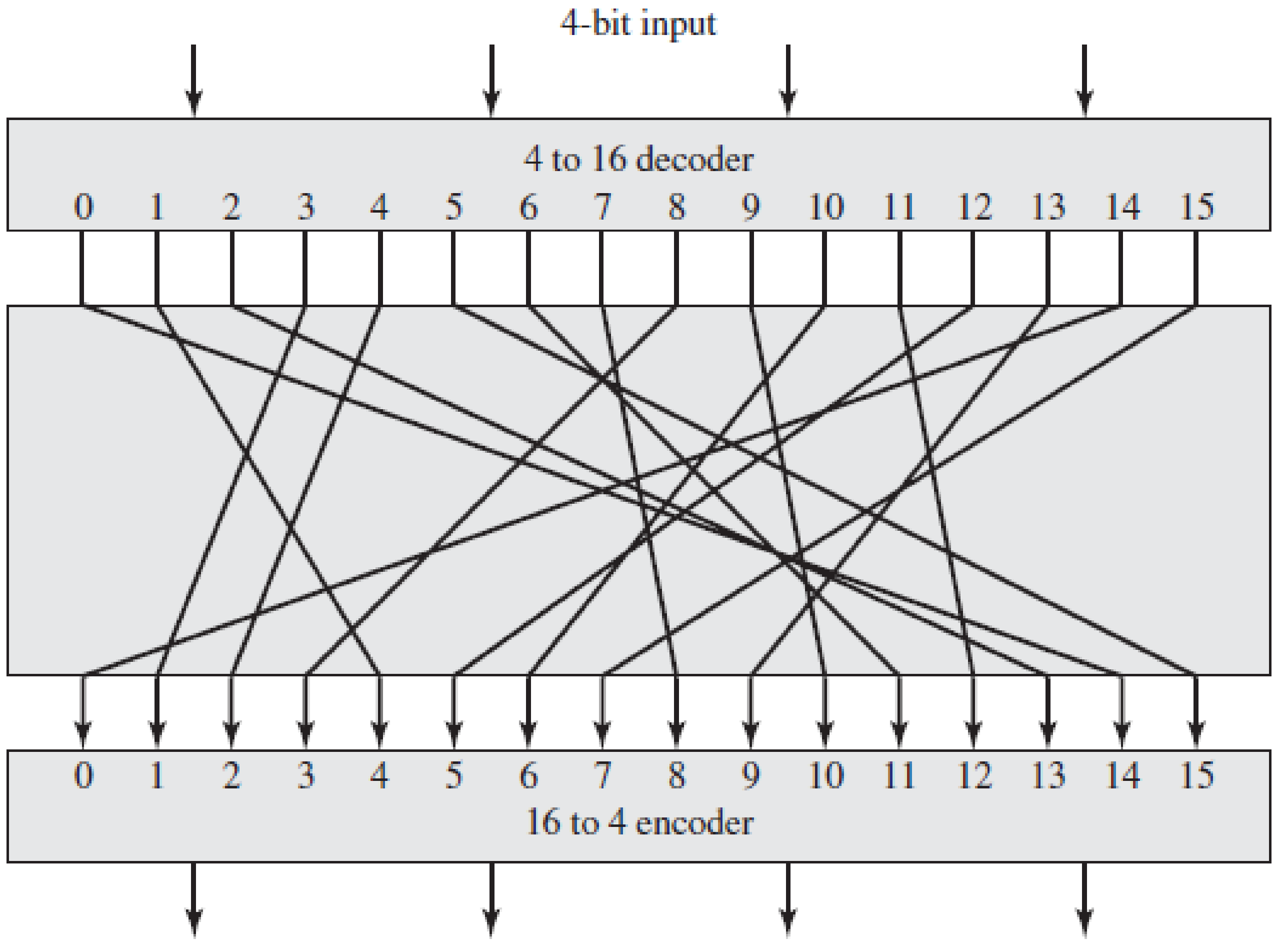


Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

Motivation for the Feistel Cipher Structure

- A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits
- An 64 bit key will be required!!



General n-bit-n-bit Block Substitution (shown with $n = 4$)

The Feistel Cipher

- Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers
- In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations:
  - Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
  - Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

The Feistel Cipher

- In **diffusion**, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext
- **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, to thwart attempts to discover the key

The Feistel Cipher

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size**: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- **Key size**: Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds**: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm**: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function** F: Again, greater complexity generally means greater resistance to cryptanalysis.

The Feistel Cipher

There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption**: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- **Ease of analysis**: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality

Data Encryption Standard (DES)

- The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46
- The algorithm itself is referred to as the Data Encryption Algorithm (DEA)
- A 56 bit key, 64 bit input block, 64 bit output block symmetric block cipher

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Data Encryption Standard (DES)

Simplified-DES

Advanced Encryption Standard (AES)

Other Symmetric Encryption Algorithms

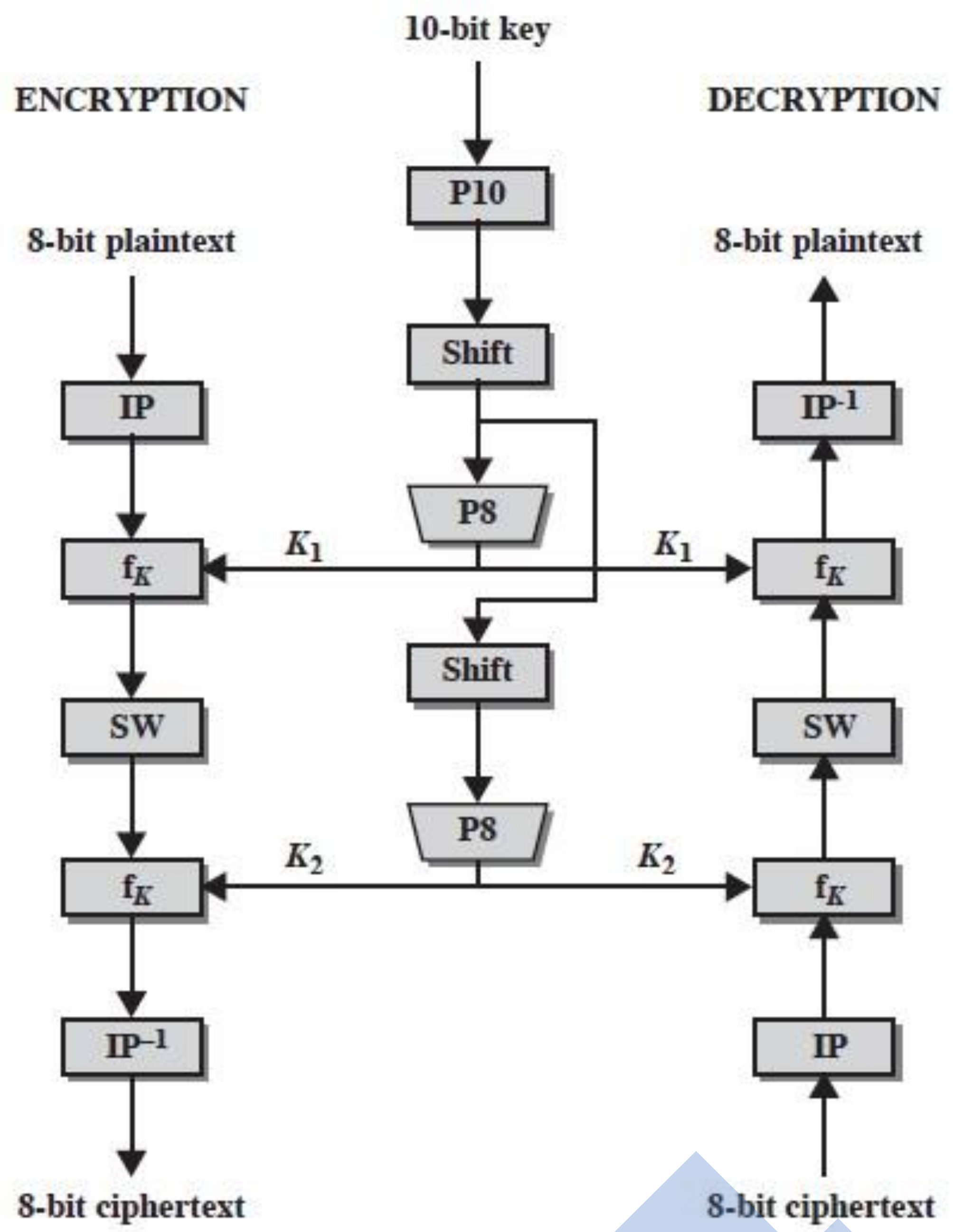
Data Encryption Standard (DES)

- The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46
- The algorithm itself is referred to as the Data Encryption Algorithm (DEA)
- A 56 bit key, 64 bit input block, 64 bit output block symmetric block cipher

Simplified-DES

- Educational encryption algorithm
- Input (plaintext) block: 8 bits
- Output (ciphertext) block: 8 bits
- Key: 10 bits
- Rounds: 2
- Round keys generated using permutations and left shifts
- Encryption: initial permutation, round function, switch halves
- Decryption: Same as encryption, except round keys used in opposite order

Simplified-DES



Simplified-DES

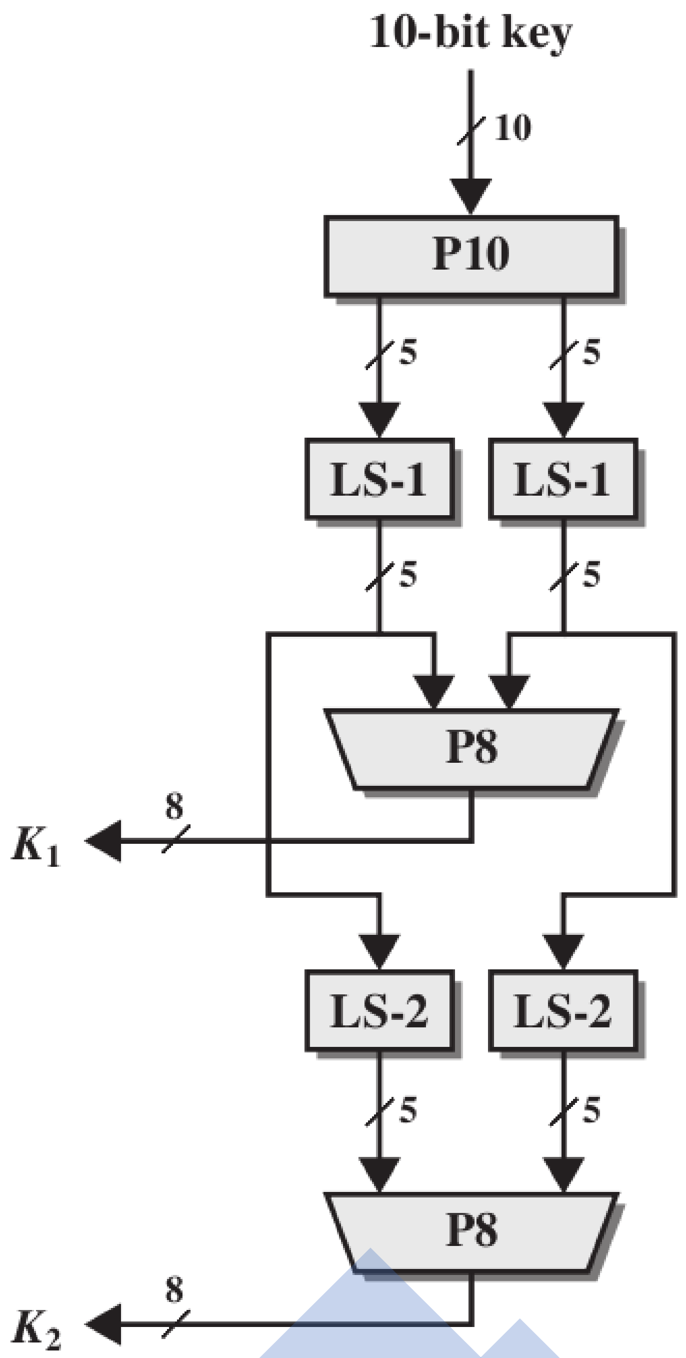
- Plaintext: 01110010
- Key: 1010000010
- Ciphertext: 01110111

Simplified-DES

P10 (permutate)
Input   : 1 2 3 4 5 6 7 8 9 10
Output: 3 5 2 7 4 10 1 9 8 6

P8 (select and permutate)
Input   :1 2 3 4 5 6 7 8 9 10
Output: 6 3 7 4 8 5 10 9

LS-1 (left shift 1 position)
LS-2 (left shift 2 positions)



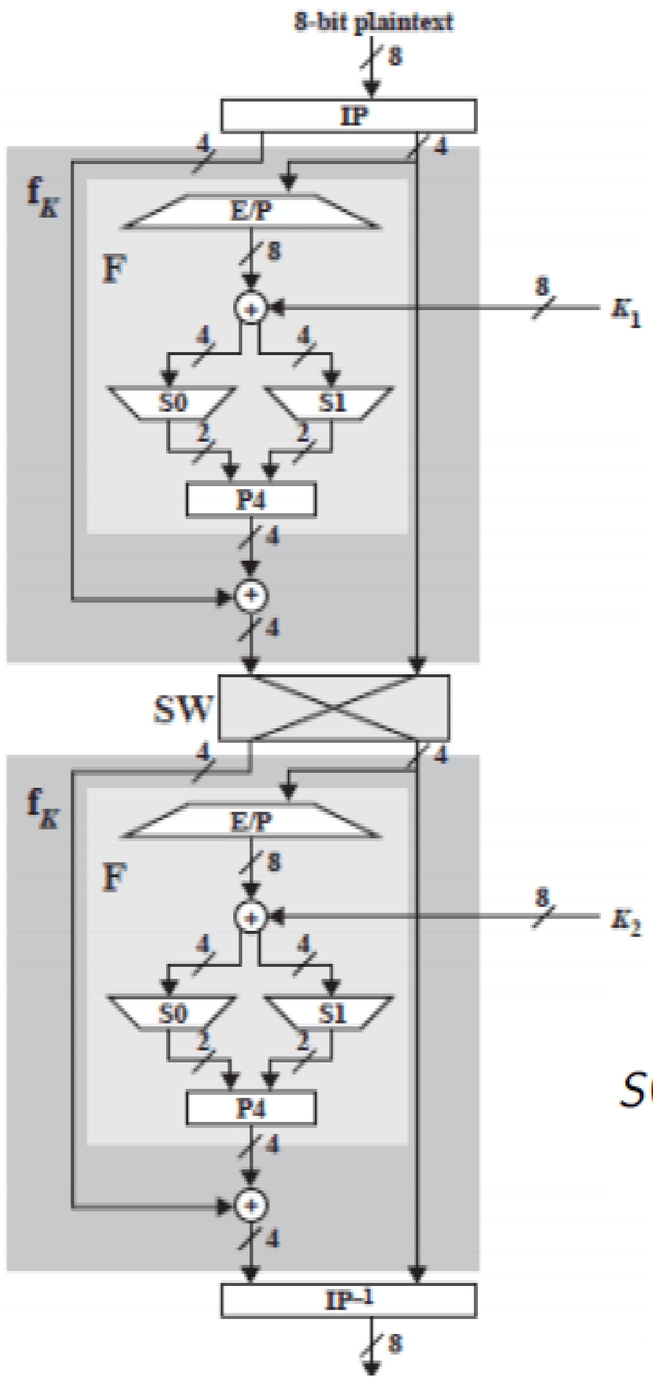
Simplified-DES

IP (initial permutation)
Input :1 2 3 4 5 6 7 8
Output:2 6 3 1 4 8 5 7

EP(expand and permutate)
Input :1 2 3 4
Output:4 1 2 3 2 3 4 1

P4 (permutate)
Input :1 2 3 4
Output:2 4 3 1

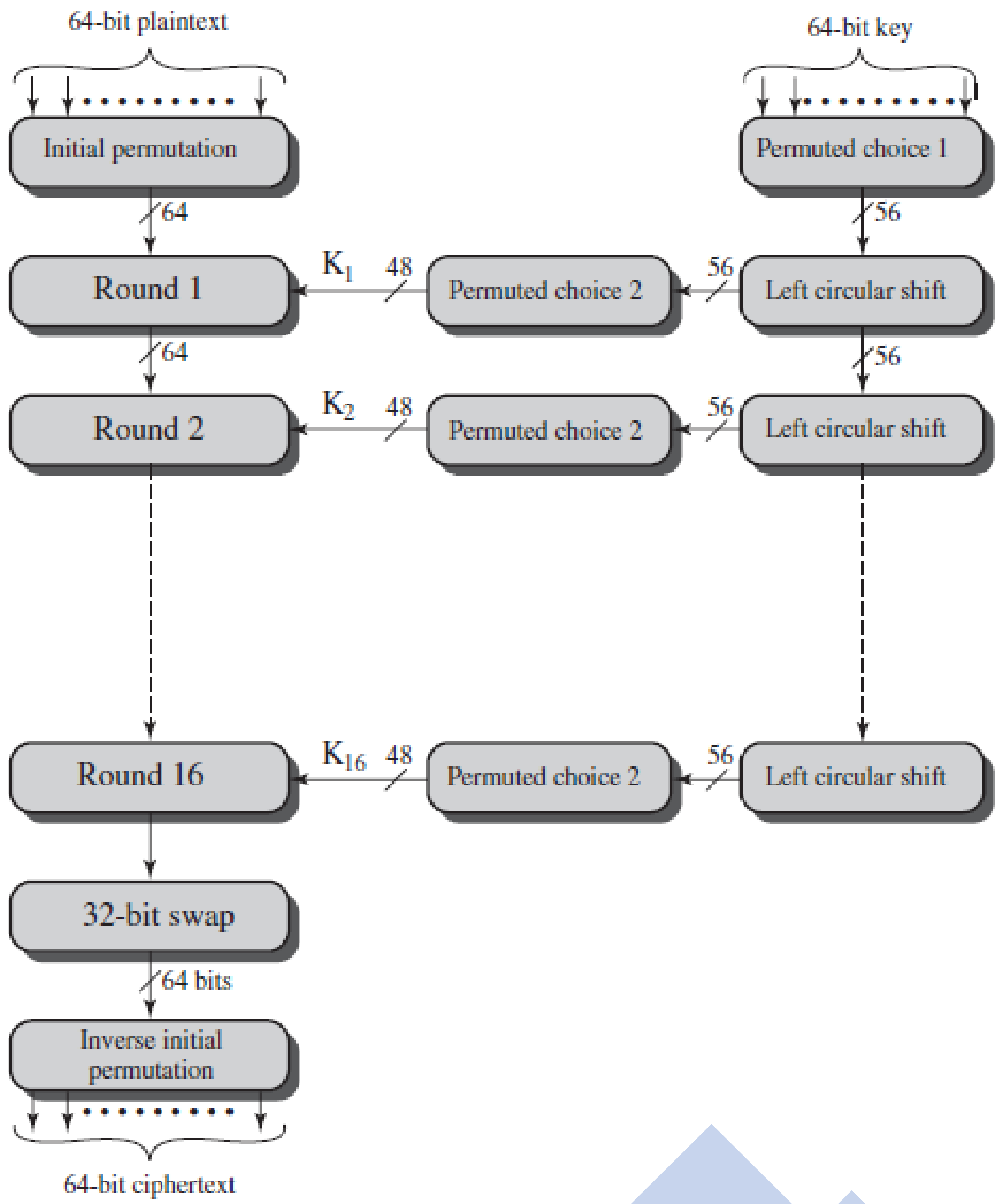
$\mathbf{IP^{-1}}$(inverse of IP)



- **S-DES** (and DES) perform substitutions using S-Boxes
- **S-Box** considered as a matrix: input used to select row/column; selected element is output
- **4 bit input**: bit1, bit2, bit3, bit4
- **bit1, bit4 specifies row** (0,1, 2 or 3 in decimal)
- **bit2, bit3 specifies column**
- **2-bit output**

$$S0 = \begin{array}{cccc} 01 & 00 & 11 & 10 \\ 11 & 10 & 01 & 00 \\ 00 & 10 & 01 & 11 \\ 11 & 01 & 11 & 10 \end{array} \qquad S1 = \begin{array}{cccc} 00 & 01 & 10 & 11 \\ 10 & 00 & 01 & 11 \\ 11 & 00 & 01 & 00 \\ 10 & 01 & 00 & 11 \end{array}$$

DES Encryption Algorithm



DES Encryption Algorithm

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

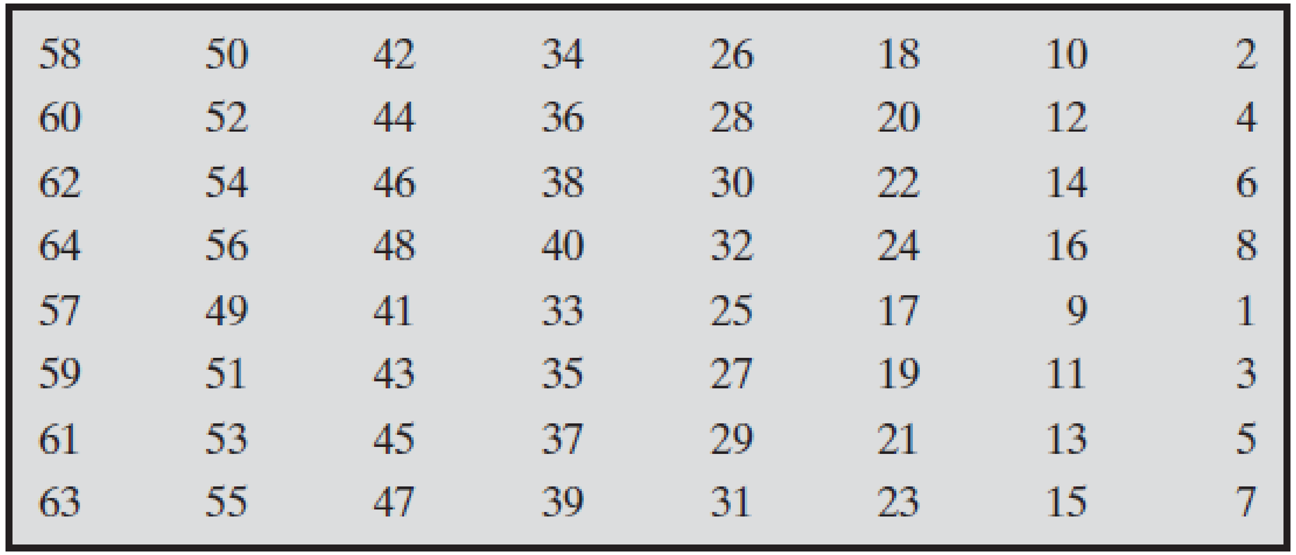
DES Encryption Algorithm

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

DES Encryption Algorithm

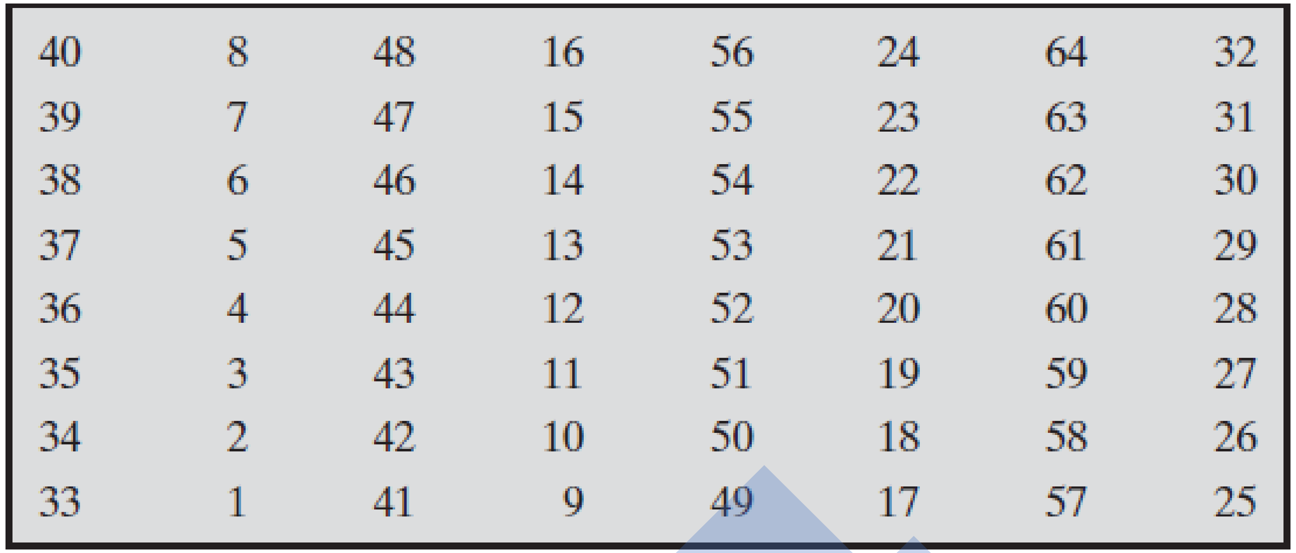
Permutation Tables for DES

(a) Initial Permutation (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

(b) Inverse Initial Permutation (IP^{-1})

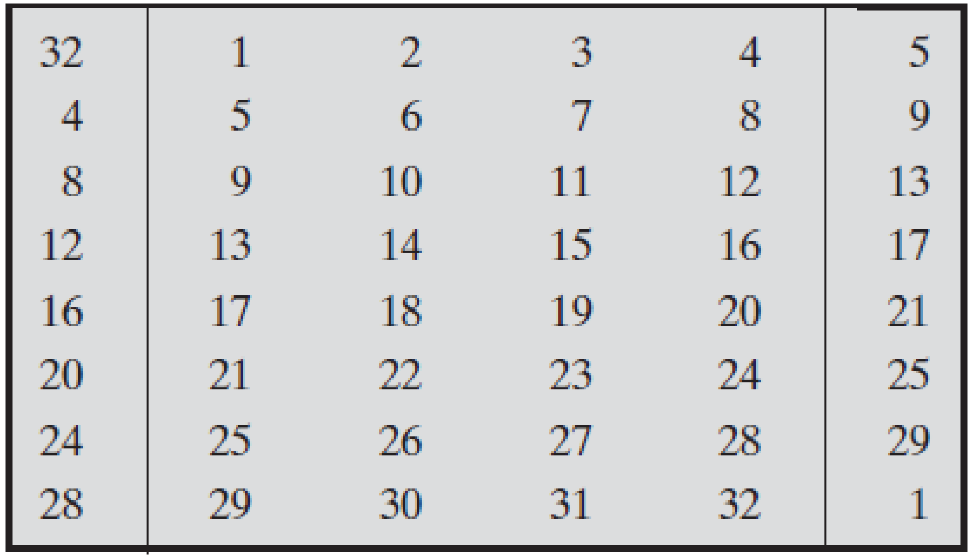


40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES Encryption Algorithm

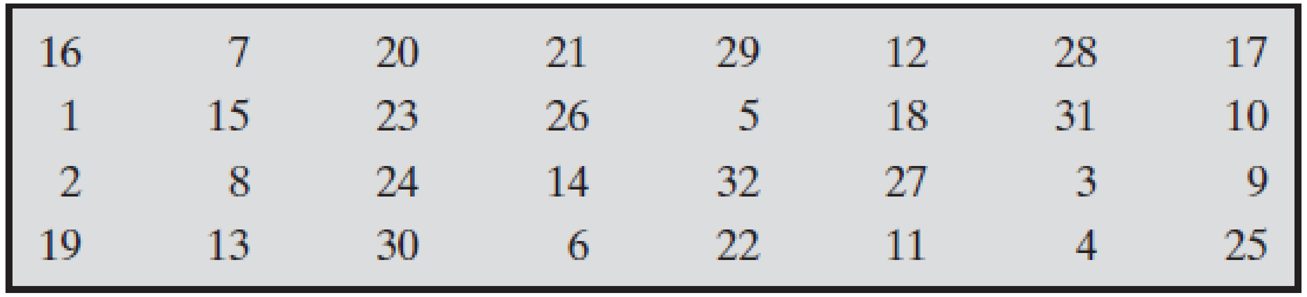
Permutation Tables for DES

(c) Expansion Permutation (E)



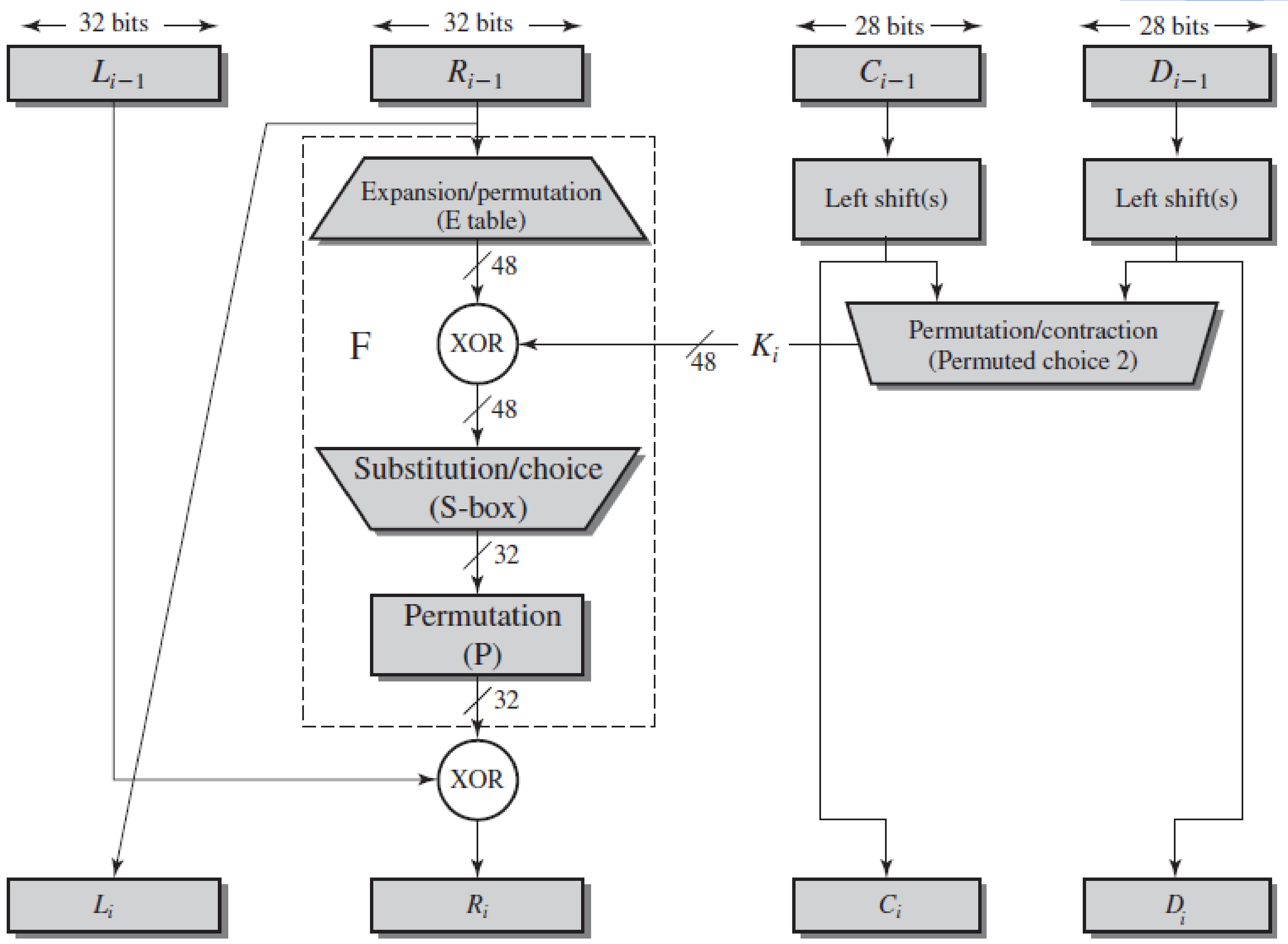
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

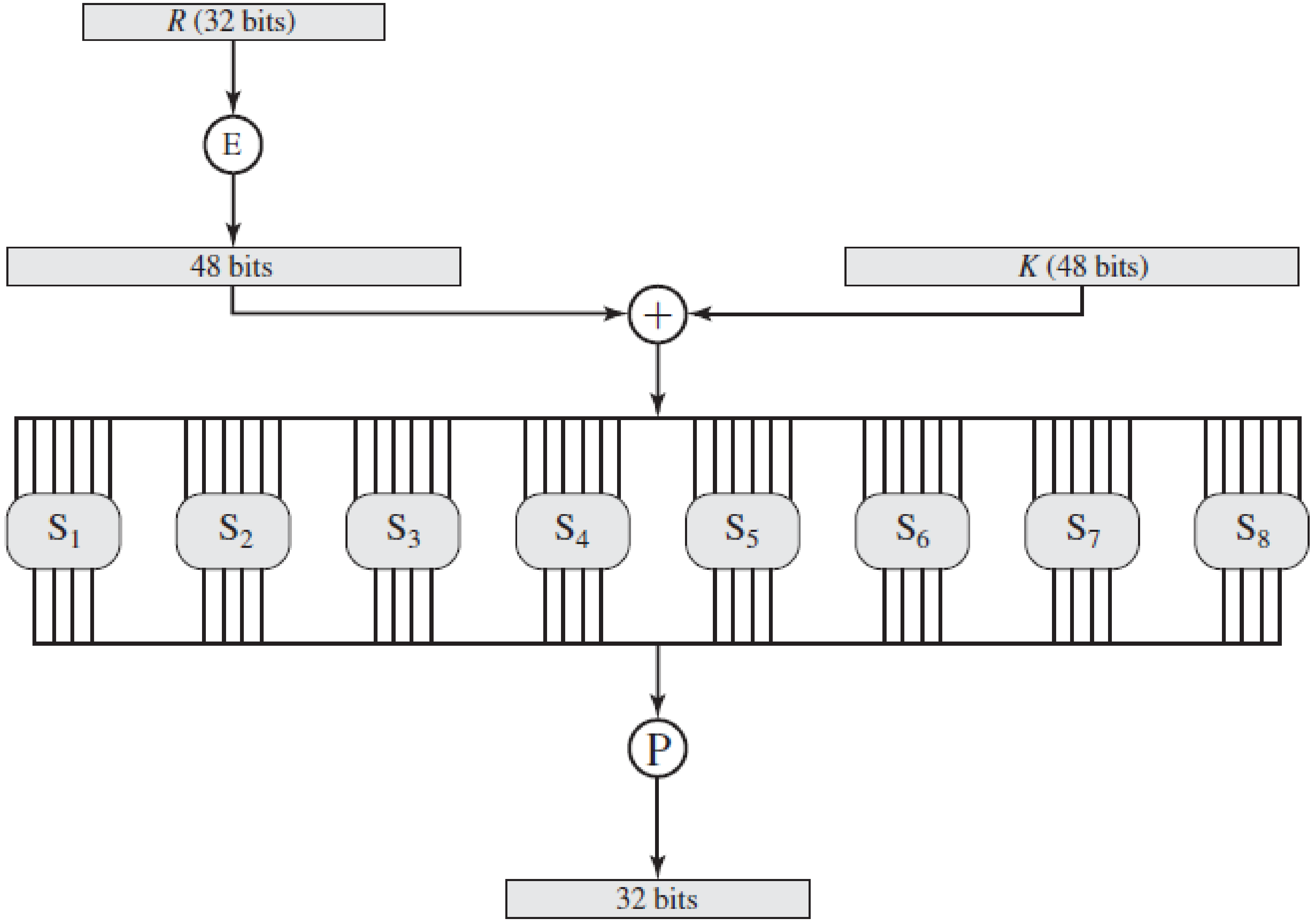


16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

DES Encryption Algorithm



DES Encryption Algorithm



DES Encryption Algorithm

Definition of DESS-Boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

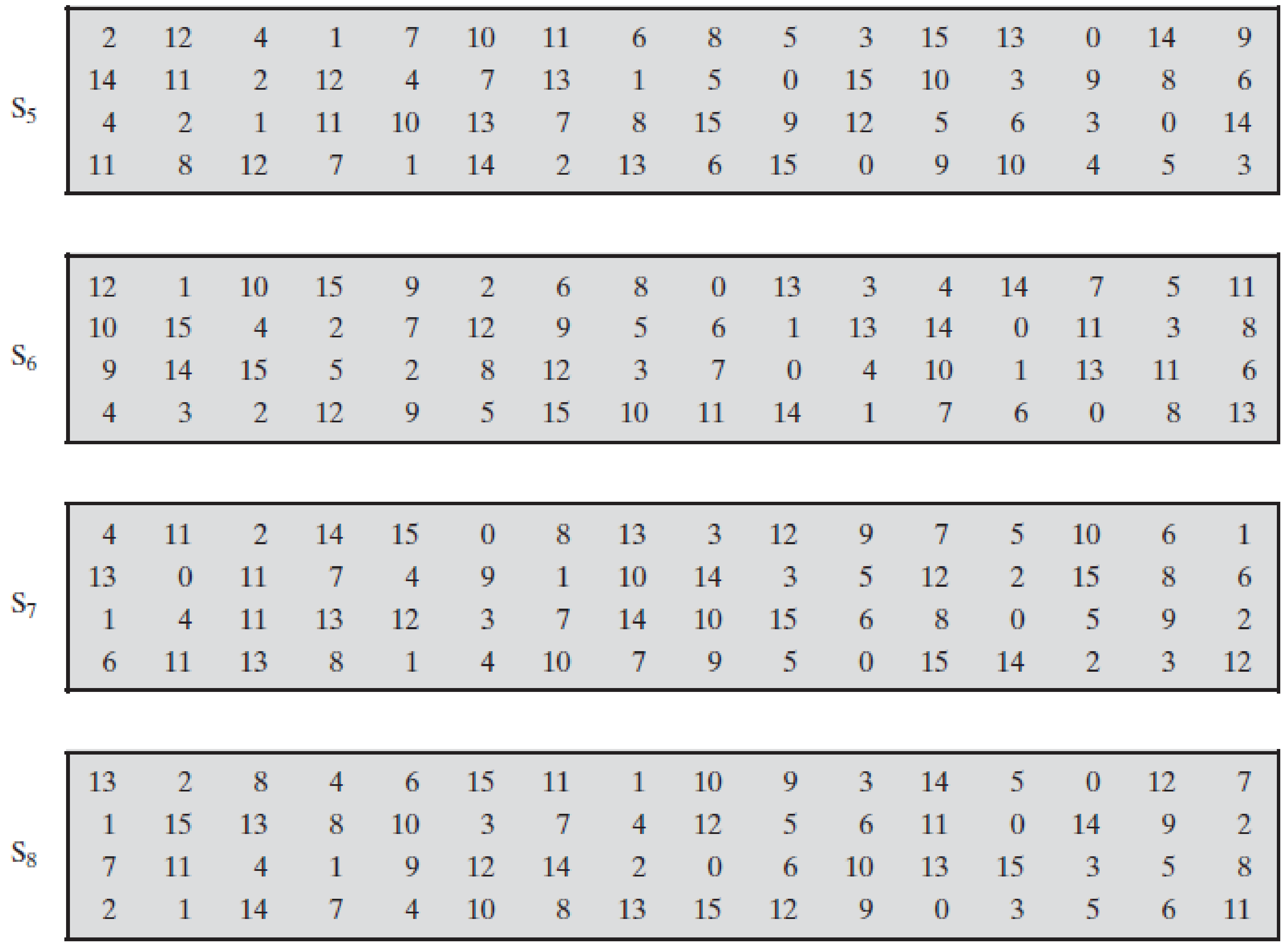
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

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

DES Encryption Algorithm

Definition of DESS-Boxes



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

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

DES Encryption Algorithm

DES Key Schedule Calculation

(a) Input Key

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

(b) Permuted Choice One (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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of 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

Data Encryption Standard (DES)

DES Algorithm Design

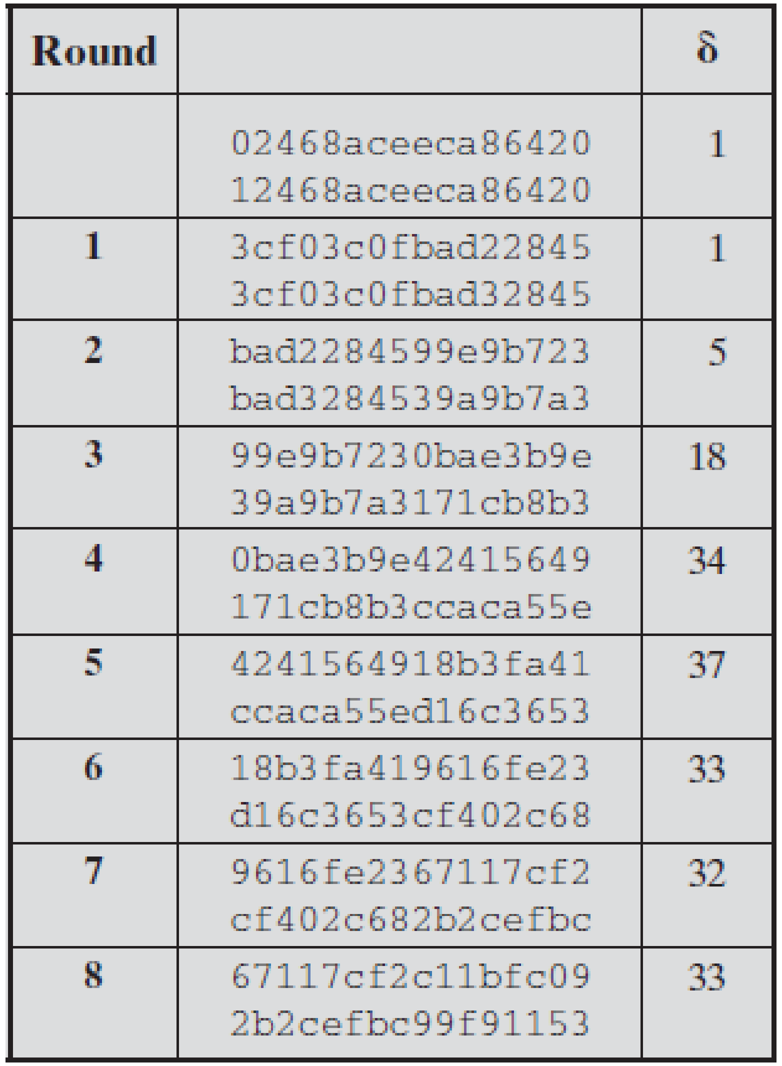
- DES was designed in private; questions about the motivation of the design
- S-Boxes provide non linearity: important part of DES, generally considered to be secure
- S-Boxes provide increased confusion
- Permutation P chosen to increase diffusion

The Avalanche Effect

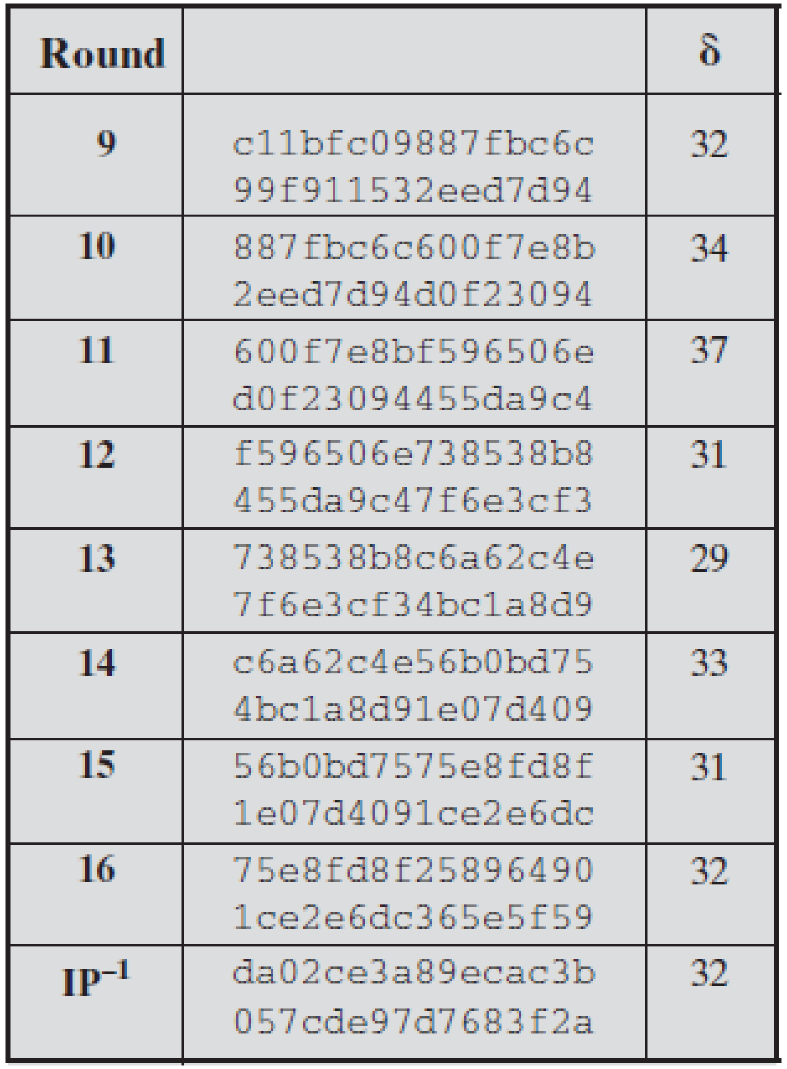
Aim: small change in key (or plaintext) produces large change in ciphertext

- Avalanche effect is present in DES (good for security)
- Following examples show the number of bits that change in output when two different inputs are used, differing by 1 bit
  - Plaintext 1: 02468aceeca86420
  - Plaintext 2: 12468aceeca86420
  - Ciphertext difference: 32 bits
  - Key 1: 0f1571c947d9e859
  - Key 2: 1f1571c947d9e859
  - Ciphertext difference: 30

Avalanche Effect in DES: Change in Plaintext



Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33



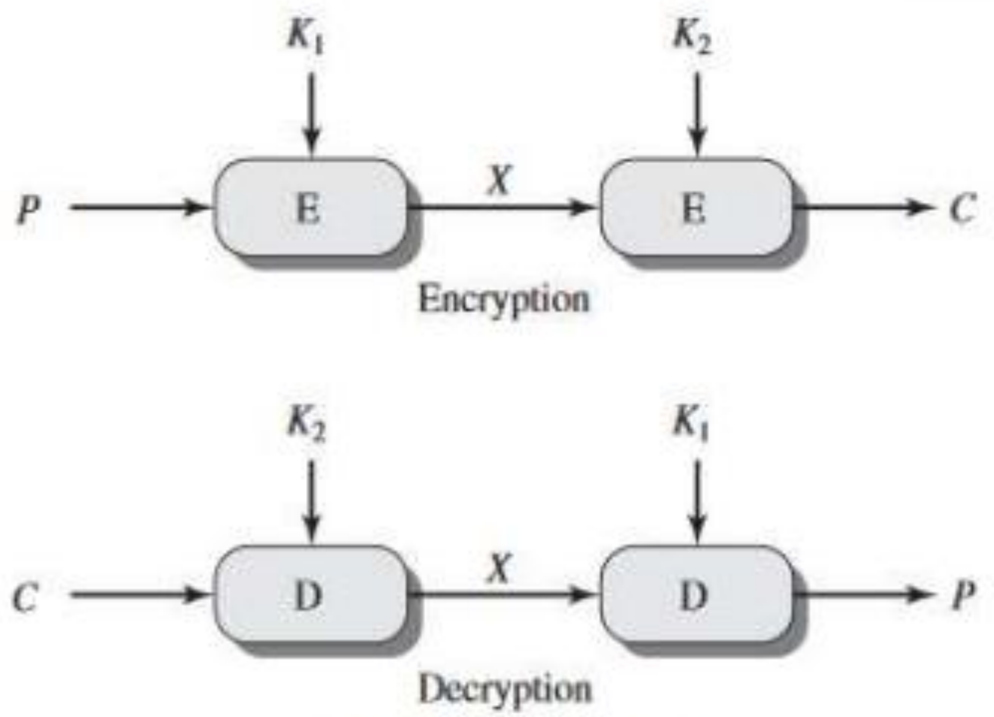
Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Avalanche Effect in DES: Change in Key

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Double Encryption

- For DES, 2×56 bit keys, meaning 112 bit key length
- Requires 2^{111} operations for brute force?
- Meet in the middle attack makes it easier



(a) Double encryption

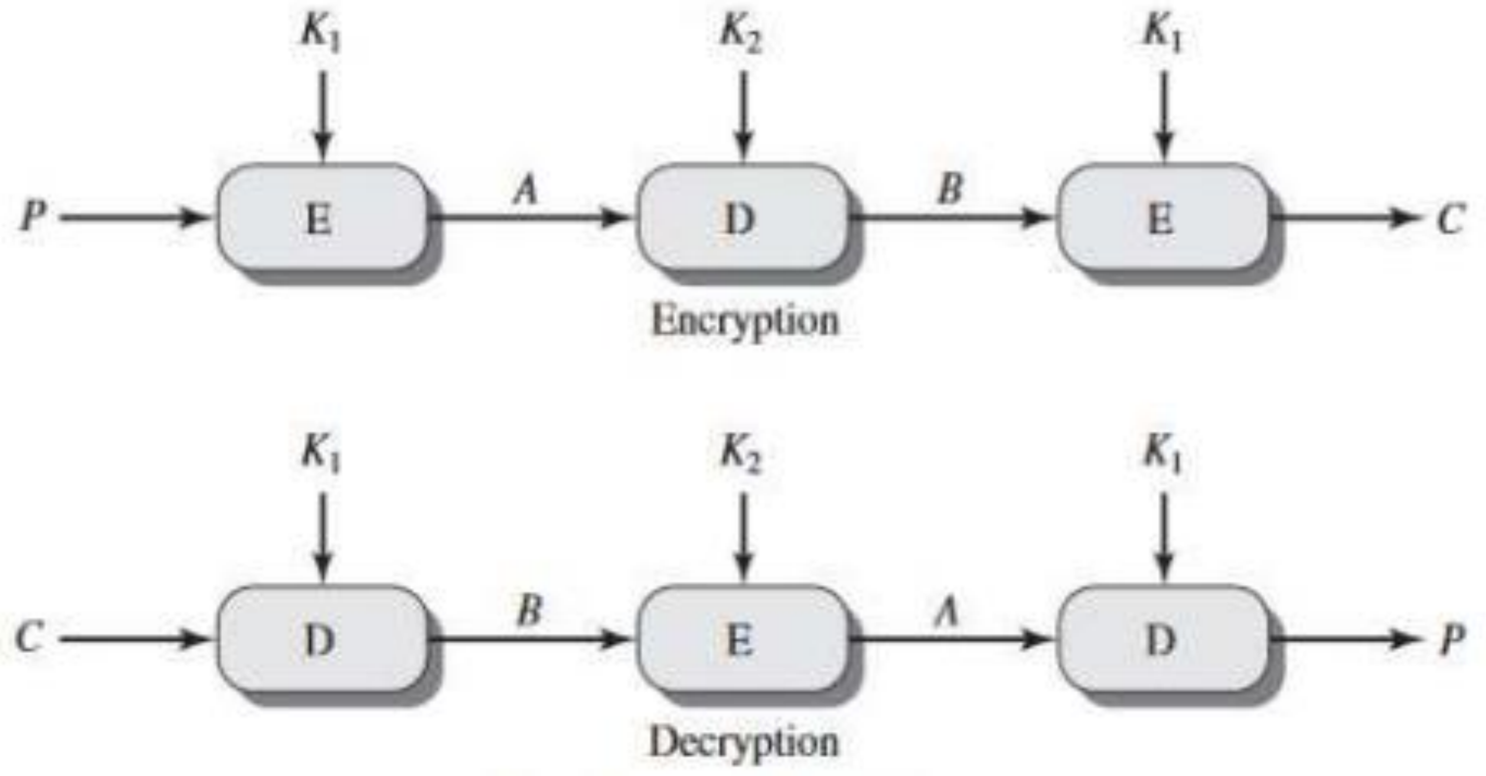
Meet in the Middle Attack

Double DES Encryptions: C=E(K2,E(K1,P))
Say X=E(K1,P)=D(K2,C)
Attacker knows two plaintext, ciphertext pairs (Pa, Ca) and (Pb, Cb)
1. Encrypt P using all 2^{56} values of K1 to get multiple values of X
2. Store results in table and sort by X
3. Decrypt Ca using all 2^{56} values of K2
4. As each decryption result produced, check against table
5. If match, check current K1, K2 on Cb. If Pb obtained, then accept the keys

With two known plaintext, ciphertext pairs, probability of successful attack is almost 1
Encrypt/decrypt operations required: 2^{56} (twice as many as single DES)

Triple Encryption

- 2 keys, 112 bits
- 3 keys, 168 bits
- Why E-D-E? To be compatible with single DES:



Advanced Encryption Standard

NIST called for proposals for new standard in 1997

- Aims: security, efficient software/hardware implementations, low memory requirements, parallel processing
- Candidate algorithms from around the world
- Rijndael chosen, standard called AES created in 2001

AES:

- Block size: 128 bits (others possible)
- Key size: 128, 192, 256 bits
- Rounds: 10, 12, 14 (depending on key)
- Operations: XOR with round key, substitutions using
- S-Boxes, mixing using Galois Field arithmetic

Widely used in file encryption, network communications

Generally considered secure

See textbook for details (including S AES)

Other Symmetric Encryption Algorithms

Blowfish (1993): 64 bit blocks/32 448 bit keys; Feistel structure

Twofish (Schneier et al, 1998): 128/128, 192, 256; Feistel structure

Serpent (Anderson et al, 1998): 128/128, 192, 256; Substitution permutation network

Camellia (Mitsubishi/ 2000): 128/128, 192, 256; Feistel structure

CAST 128 (Adams and Tavares, 1996): 64/40 128; Feistel structure

CAST 256 (Adams and Tavares, 1998): 128/up to 256; Feistel structure

RC5 (1994): 32, 64 or 128/up to 2040; Feistel like structure

RC6 (Rivest et al, 1998): 128/128, 192, 256; Feistel structure

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

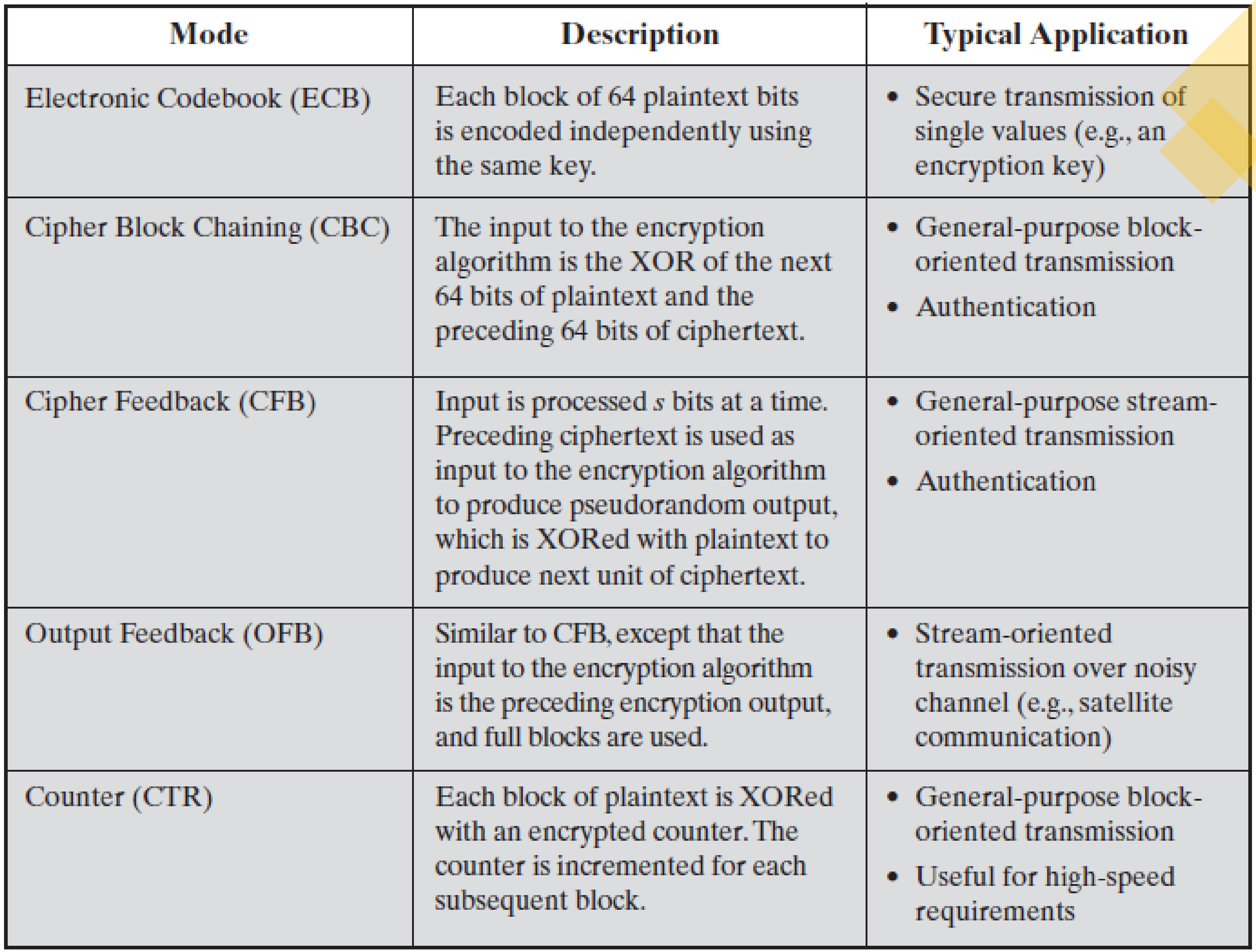
Topics

Block Cipher Modes

Block Cipher Modes

- A block cipher takes a fixed-length block of text of length b-bits and a key as input and produces a b-bit block of ciphertext. If the amount of plaintext to be encrypted is greater than b-bits, then the block cipher can still be used by breaking the plaintext up into b-bit blocks
- When multiple blocks of plaintext are encrypted using the same key, several security issues arise
- five *modes of operation* have been defined by NIST:
  - Electronic Codebook (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)

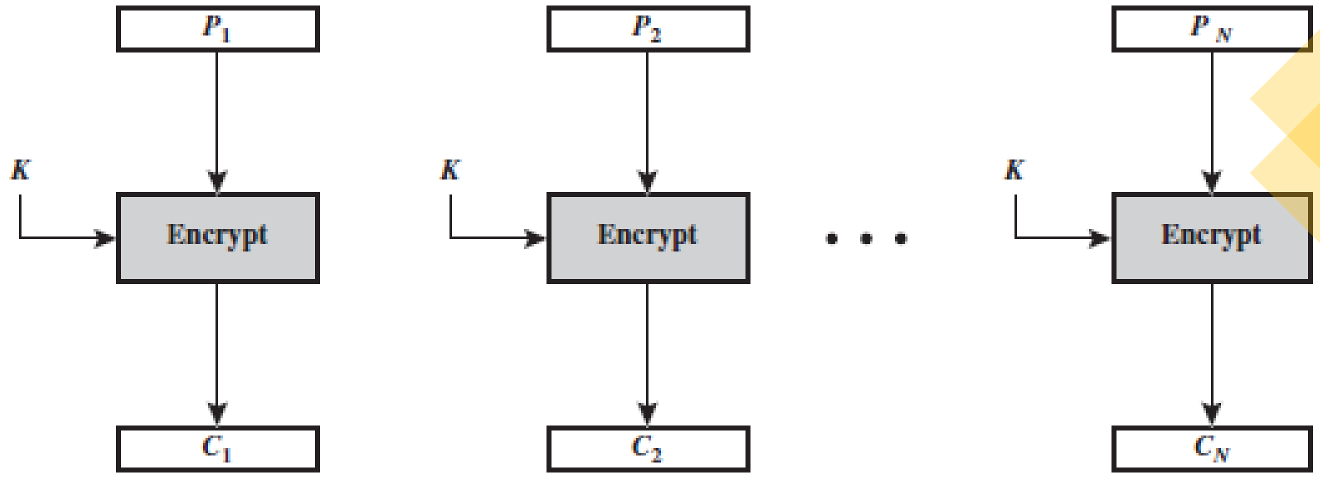
Block Cipher Modes



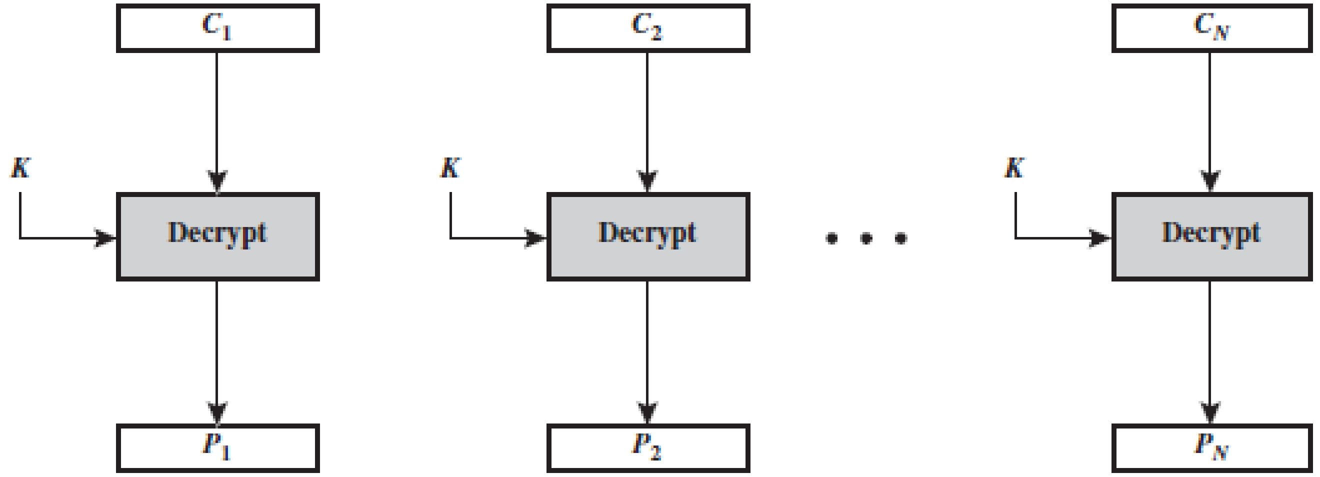
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

Electronic Codebook (ECB)

- simplest mode
- Plaintext is handled one block at a time and each block of plaintext is encrypted using the same key
- For a message longer than b-bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary
- The ECB method is ideal for a short amount of data, such as an encryption key.
- The most significant characteristic of ECB is that if the same b-bit block of plaintext appears more than once in the message, it always produces the same ciphertext.



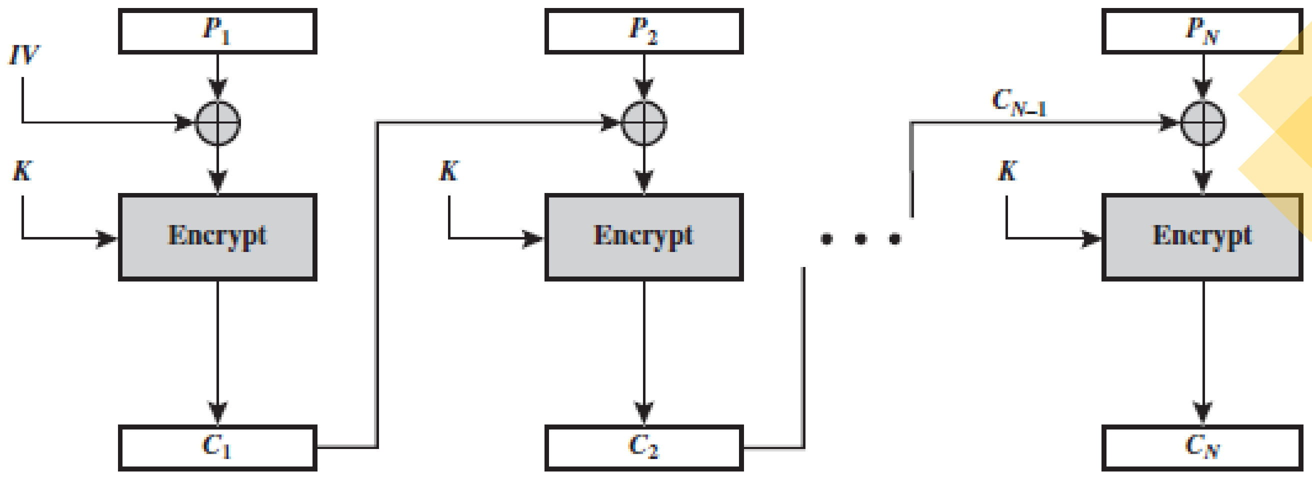
(a) Encryption



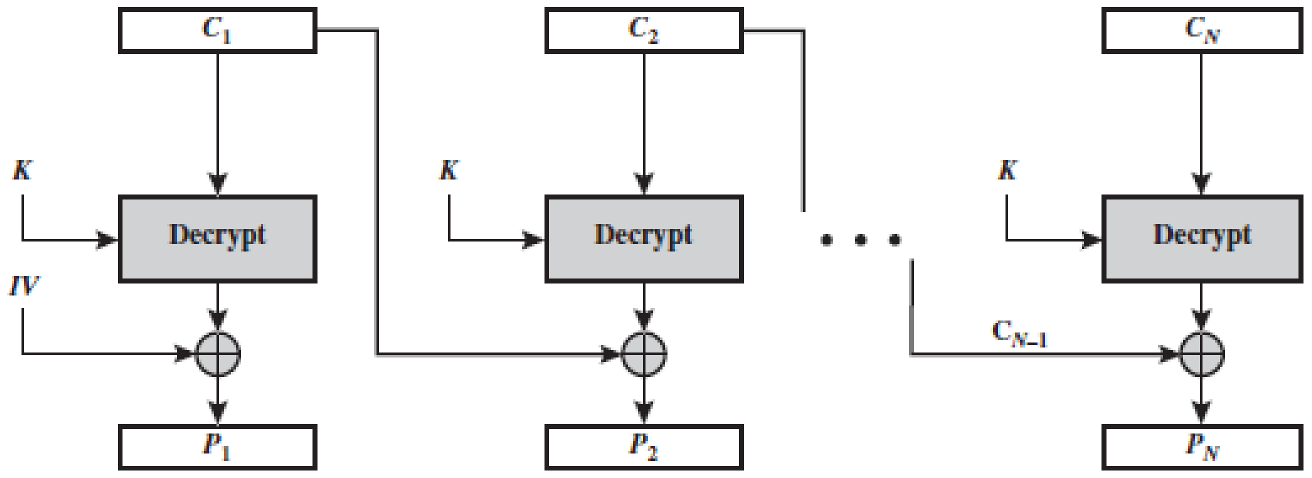
(b) Decryption

Cipher Block Chaining (CBC)

- Produces different ciphertext blocks for the same plaintext block
- The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block
- The last block should be padded to a full b-bits if it is a partial block
- For decryption, each cipher block is passed through the decryption algorithm
- To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext
- On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext



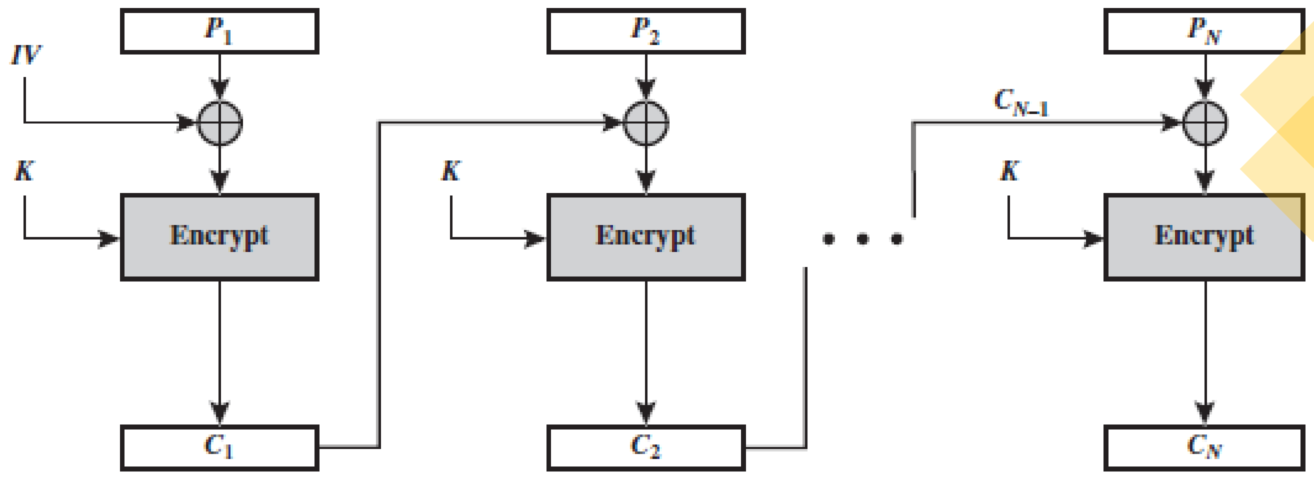
(a) Encryption



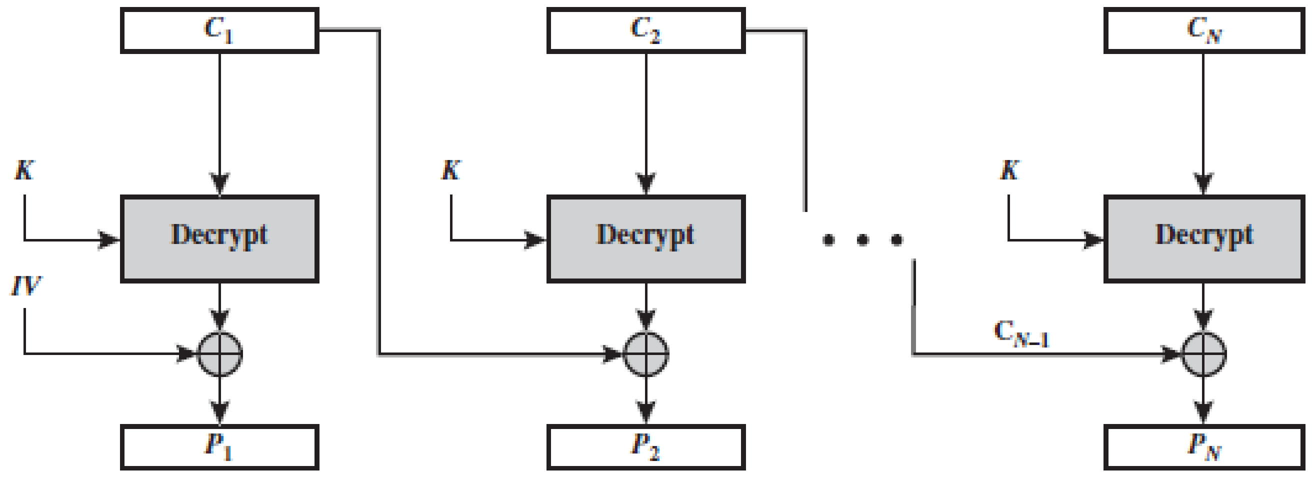
(b) Decryption

Cipher Block Chaining (CBC)

- The IV must be known to both the sender and receiver but be unpredictable by a third party
- If an opponent is able to fool the receiver into using a different value for IV, then the opponent can invert selected bits in the first block of plaintext
- The first method to hide IV is to apply the encryption function, under the same key that is used for the encryption of the plaintext, to a **nonce**
- The second method is to generate a random data block using a random number generator
- its use to achieve **confidentiality** and can be used for **authentication**



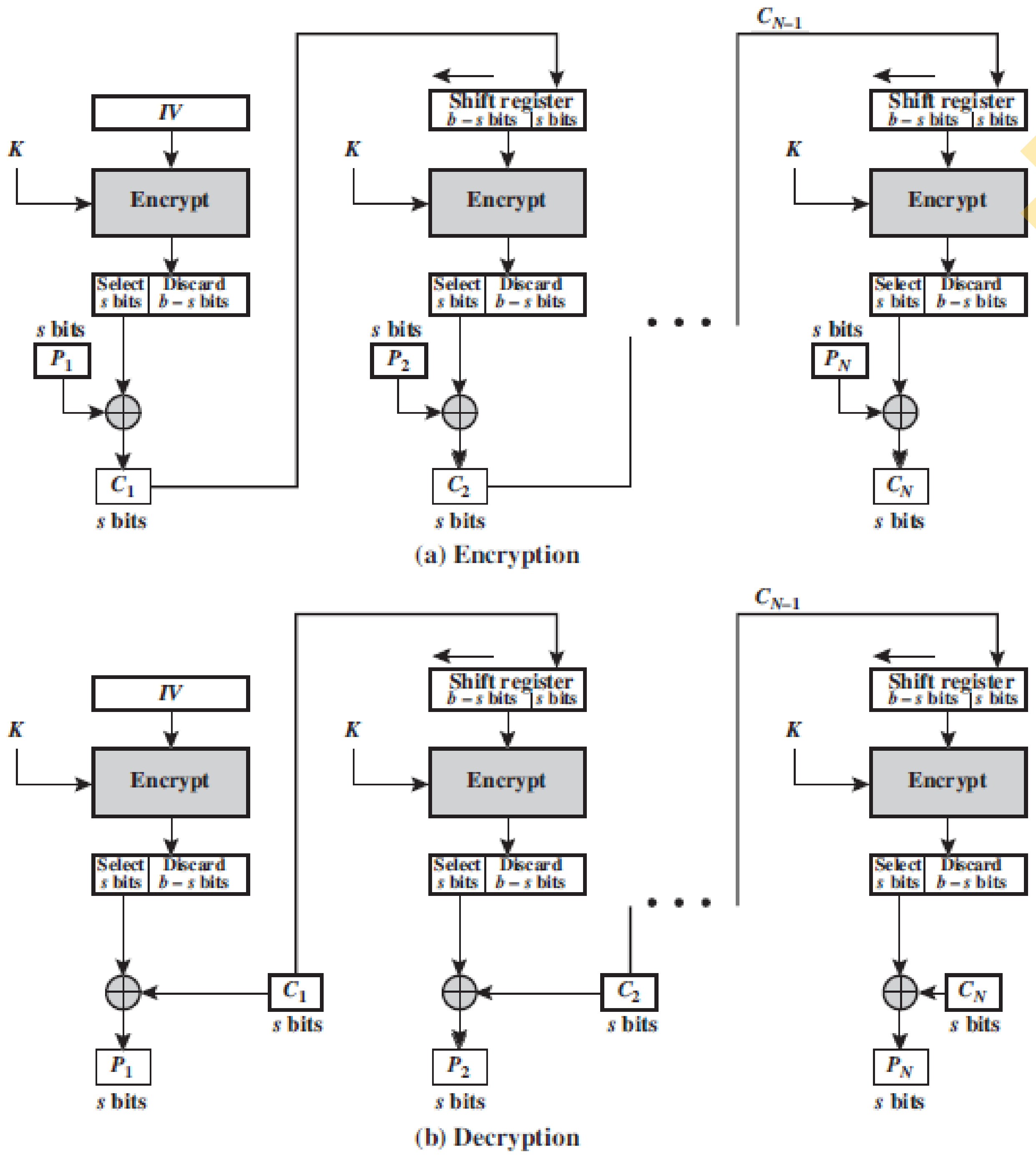
(a) Encryption



(b) Decryption

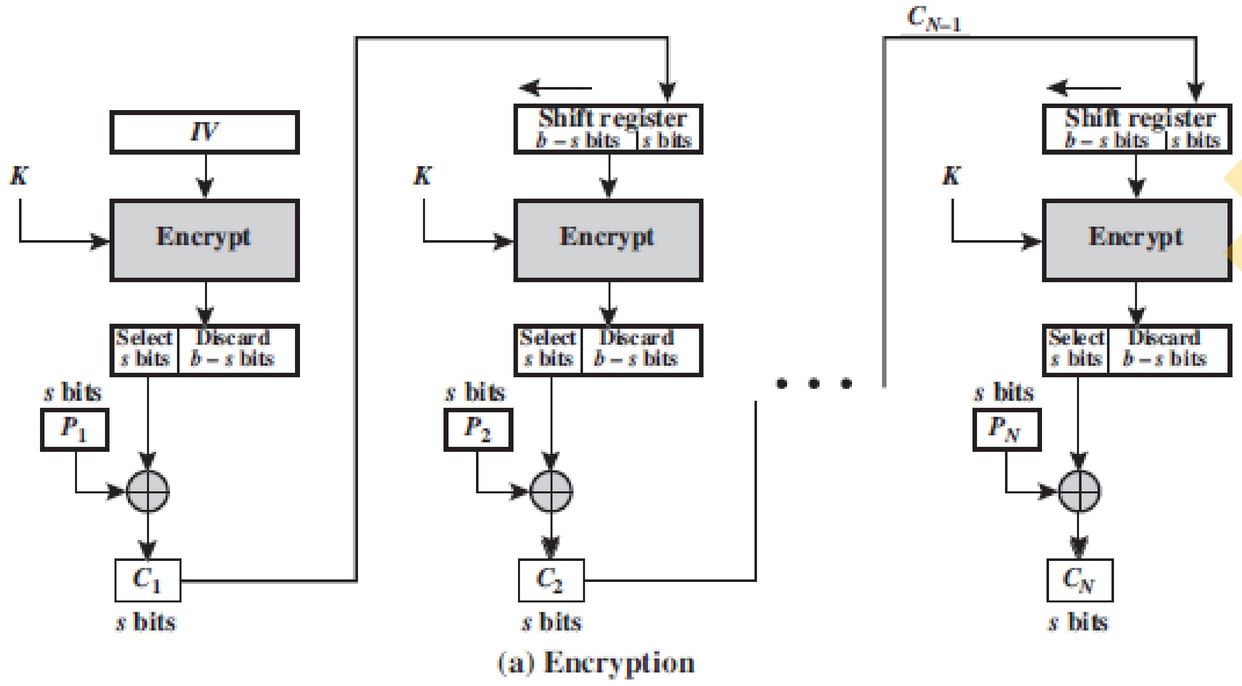
Cipher Feedback (CFB)

- A stream cipher eliminates the need to pad a message to be an integral number of blocks
- operate in real time
- One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext
- the plaintext is divided into **segments** of bits
- The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV)
- The MS output of the encryption function is XORed with the segment of s-bit size of the plaintext

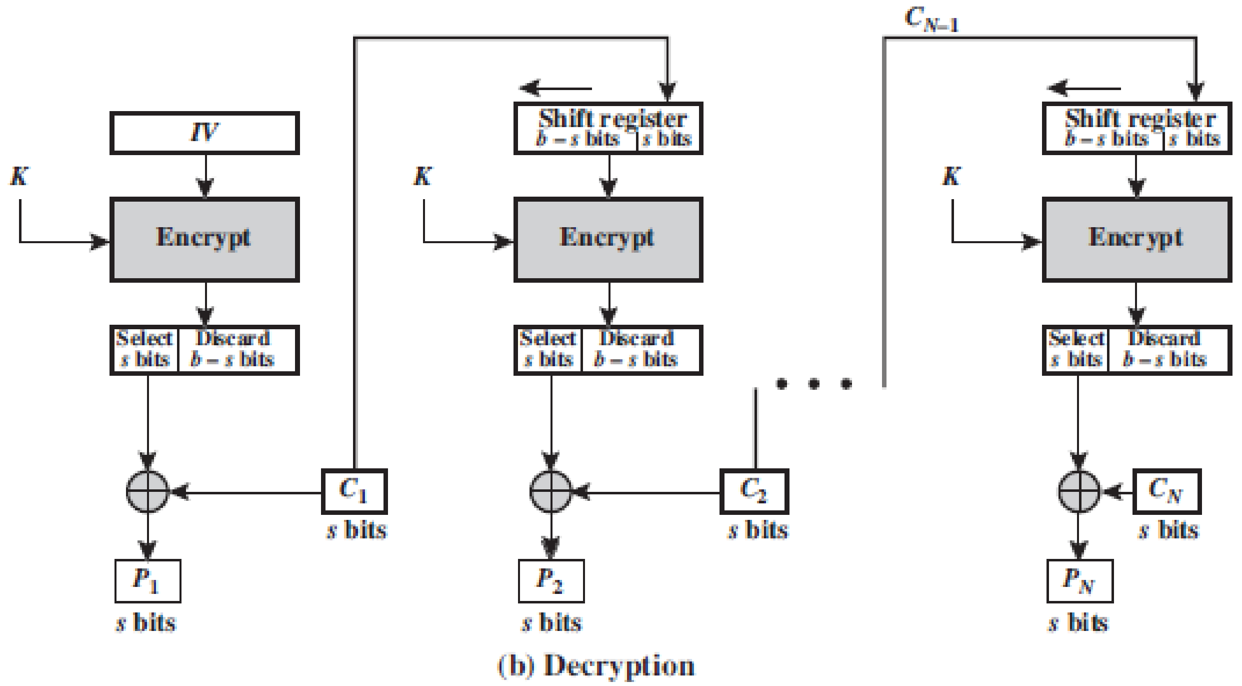


Cipher Feedback (CFB)

- The same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit
- Note that it is the encryption function that is used, not the decryption function.
- In a typical stream cipher, the cipher takes as input some initial value and a key and generates a stream of bits, which is then XORed with the plaintext bits
- In the case of CFB, the stream of bits that is XORed with the plaintext also depends on the plaintext



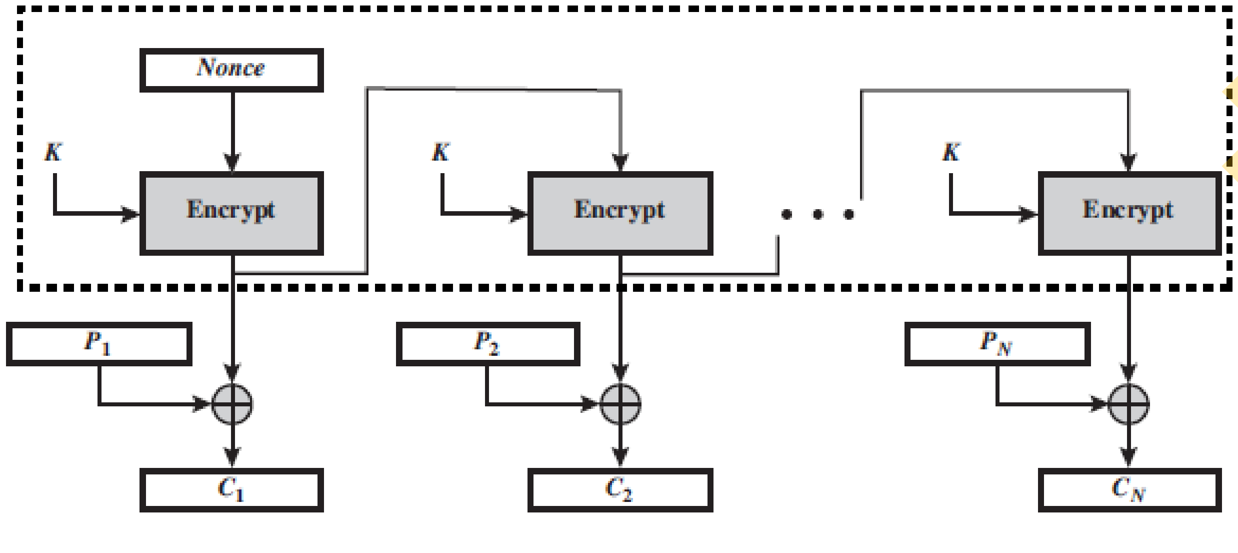
(a) Encryption



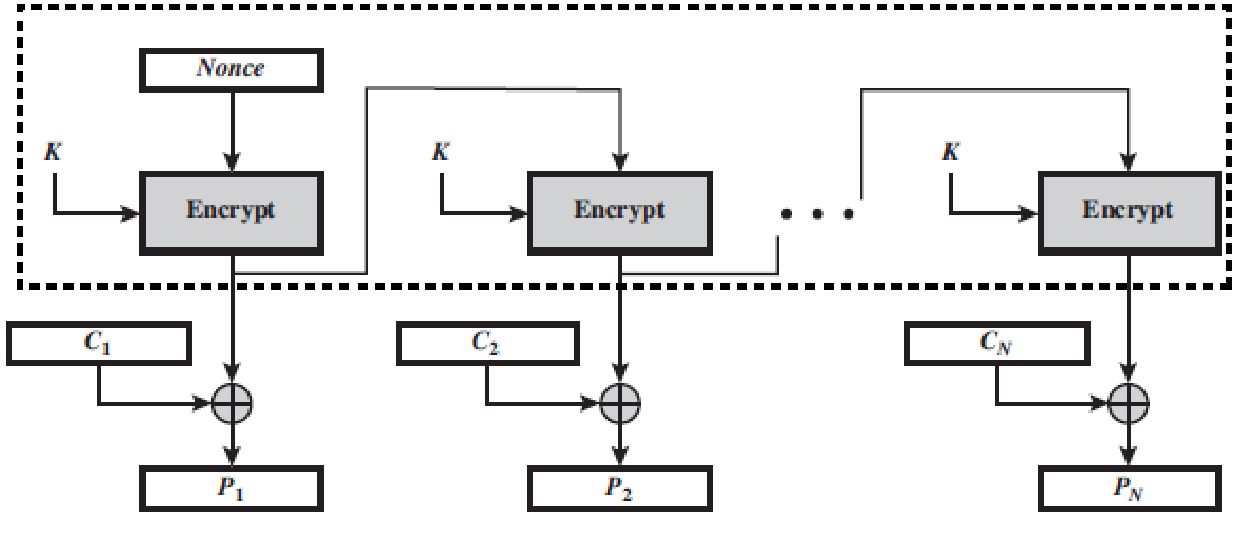
(b) Decryption

Output Feedback (OFB)

- The output of the encryption function that is fed back to the shift register in OFB
- OFB mode operates on full blocks of plaintext and ciphertext
- If the last block of plaintext contains u-bits where u<b the most significant bits of the last output block are used for the XOR operation
- the IV must be a **nonce** for each message
- One advantage of the OFB method is that bit errors in transmission do not propagate
- The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB
- OFB has the structure of a typical stream cipher



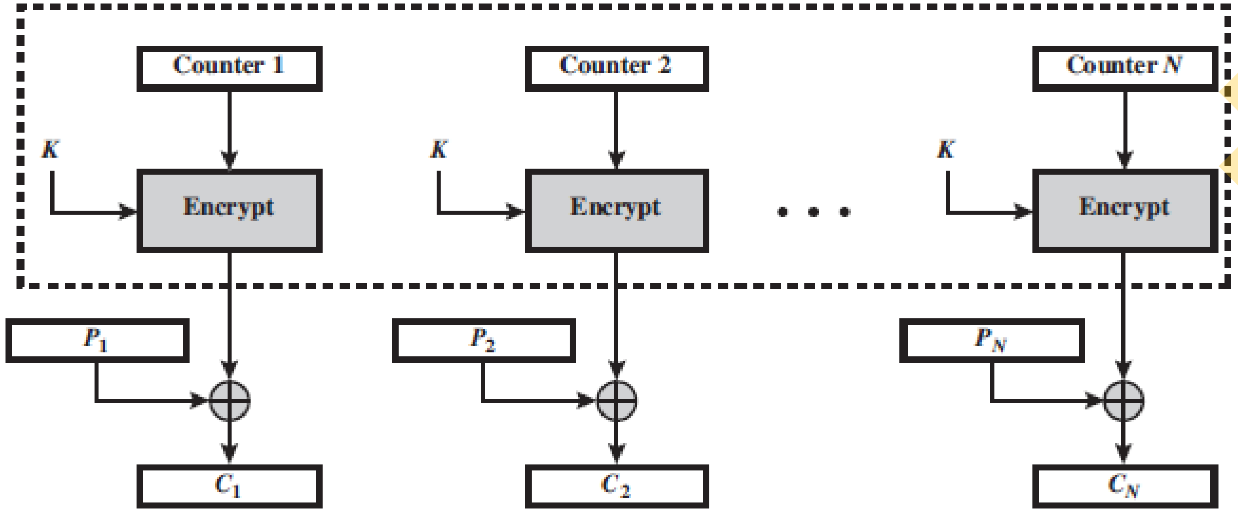
(a) Encryption



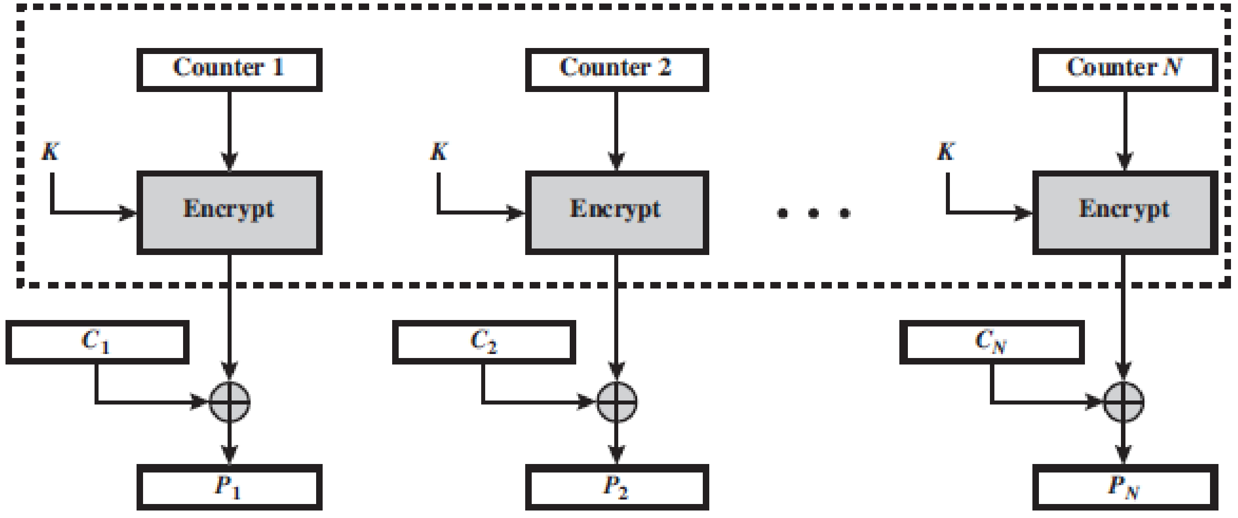
(b) Decryption

Counter (CTR)

- A counter equal to the plaintext block size is used
- The counter value must be different for each plaintext block that is encrypted
- For the last plaintext block, which may be a partial block of bits, the most significant bits of the last output block are used for the XOR operation; the remaining bits are discarded
- Unlike the ECB, CBC, and CFB modes, we do not need to use padding because of the structure of the CTR mode
- The initial counter value must be a nonce



(a) Encryption

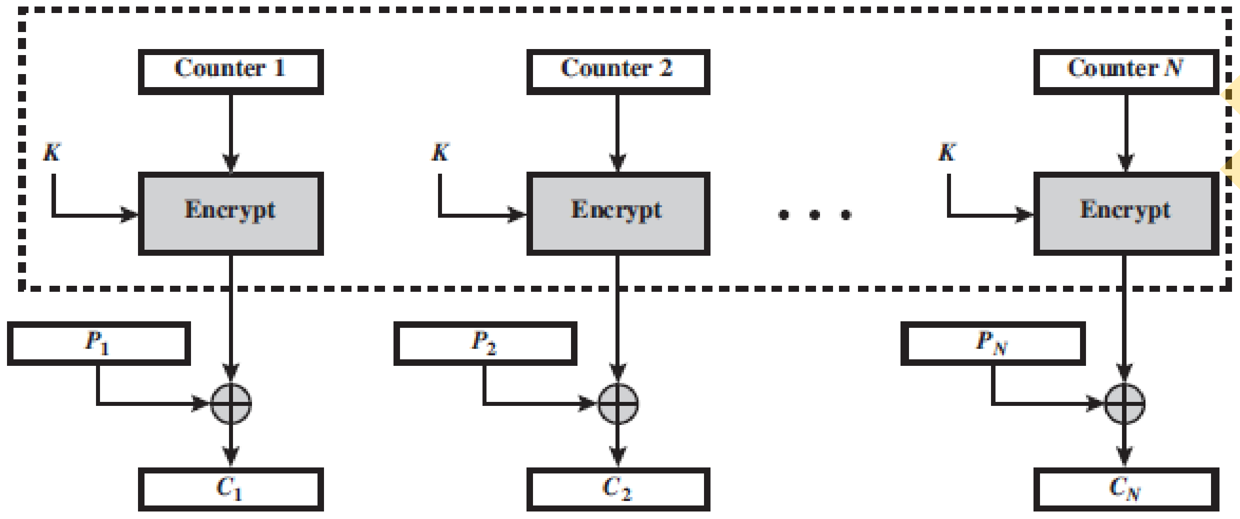


(b) Decryption

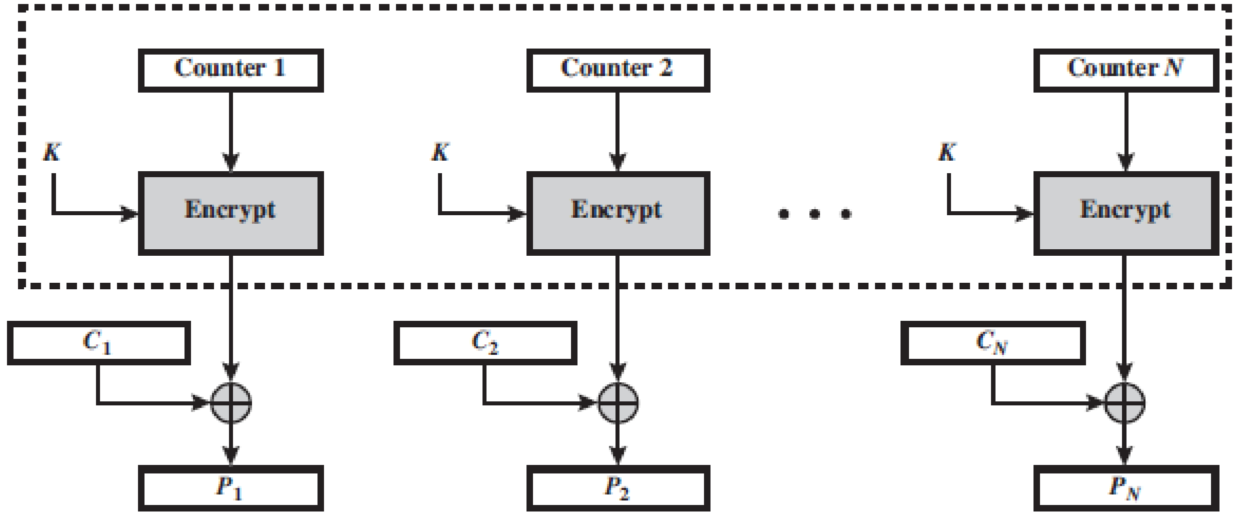
Counter (CTR)

Advantages of CTR mode:

- Hardware efficiency
- Software efficiency
- Preprocessing
- Random access
- Provable security
- Simplicity



(a) Encryption



(b) Decryption

Review Questions

- Why is it important to study the Feistel cipher?
- What is the difference between a block cipher and a stream cipher?
- What is the difference between diffusion and confusion?
- What is the purpose of the S-boxes in DES?
- Explain the avalanche effect.
- What is triple encryption?
- What is a meet-in-the-middle attack?
- How many keys are used in triple encryption?
- Why is the middle portion of 3DES a decryption rather than an encryption?
- Why do some block cipher modes of operation only use encryption while others use both encryption and decryption?

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES)

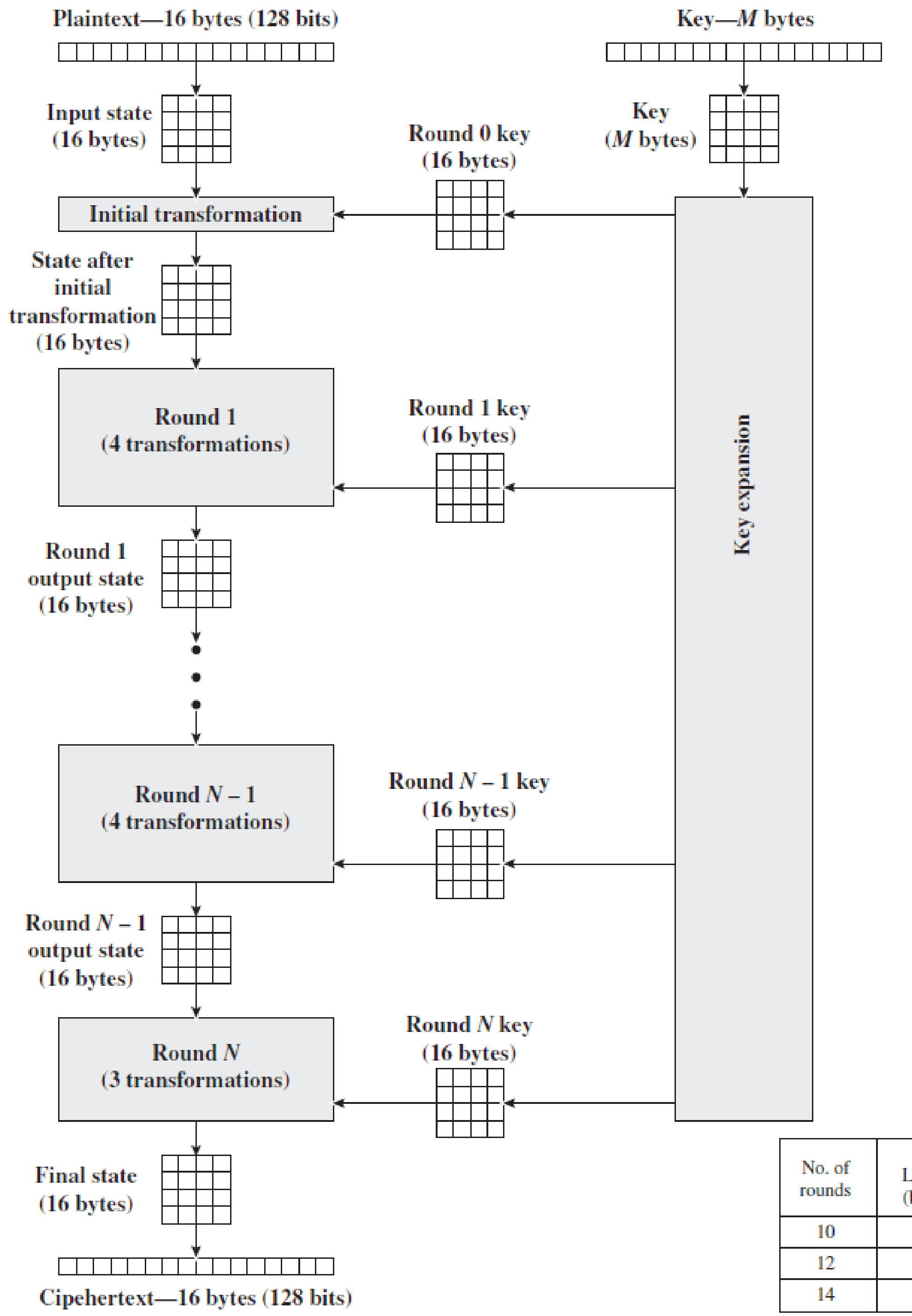
- AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications.
- Can use Triple DES but slow, has small blocks
- US NIST issued call for ciphers in 1997 and 15 candidates accepted in Jun 98, then 5 were shortlisted in Aug 99
- Rijndael was selected as the AES in Oct 2000
- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001

Advanced Encryption Standard (AES)

- AES encryption process takes a plaintext block size of 128 bits, or 16 bytes
- The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits)
- The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length
- an **iterative** rather than **Feistel** cipher
- Processes data as block of 4 columns of 4 bytes
- Operates on entire data block in every round
- Designed to have:
    - Resistance against known attacks
    - Speed and code compactness on many CPUs
    - Design simplicity

AES Encryption Process

- The cipher consists of N rounds and initial transformation, where the number of rounds depends on the key length
- The first N-1 rounds consist of four distinct transformation functions: **Sub-Bytes, Shift Rows, Mix Columns, and Add Round Key**.
- The final round contains only three transformations

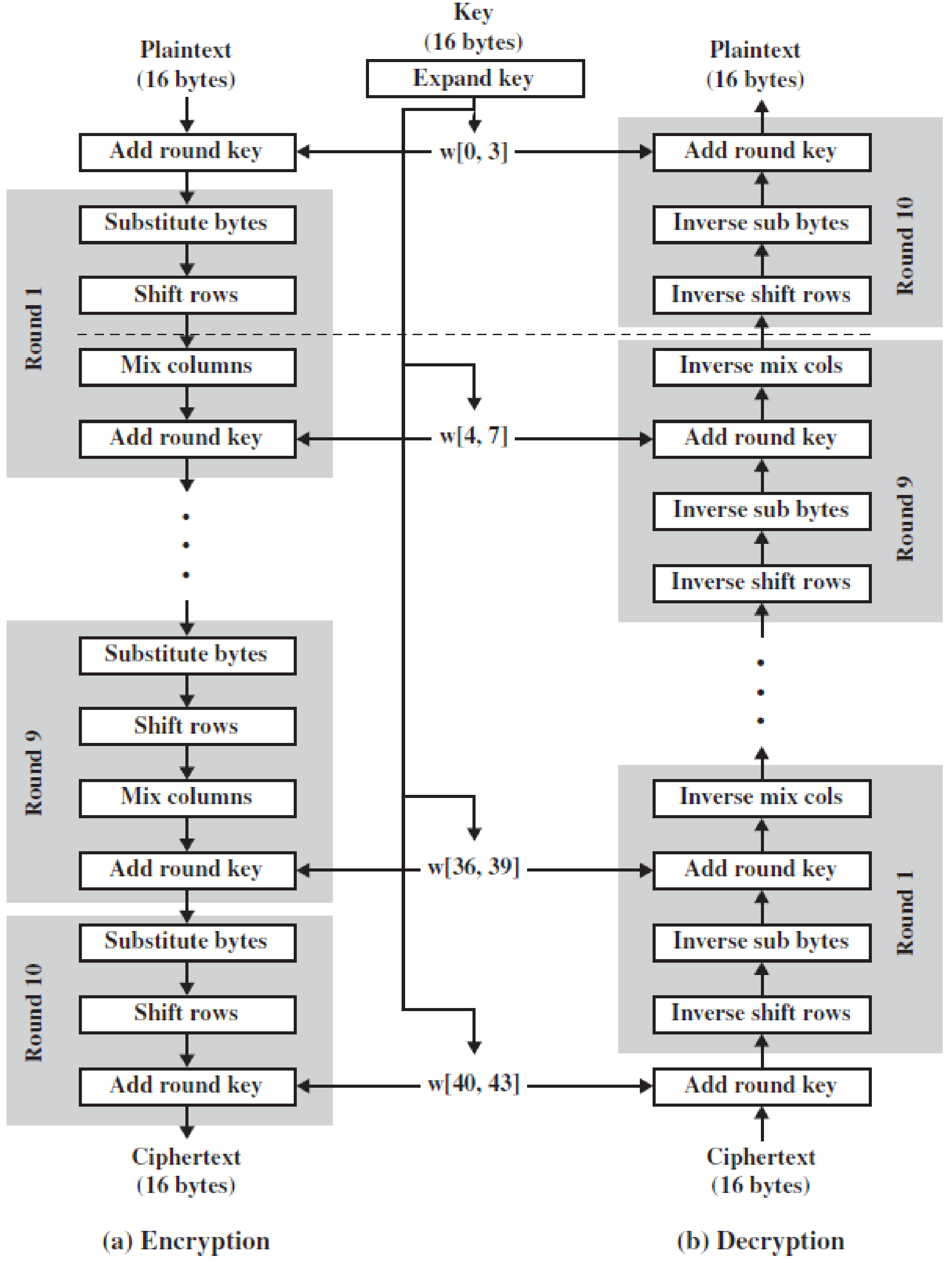

Plaintext—16 bytes (128 bits)
Key—M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Encryption and Decryption

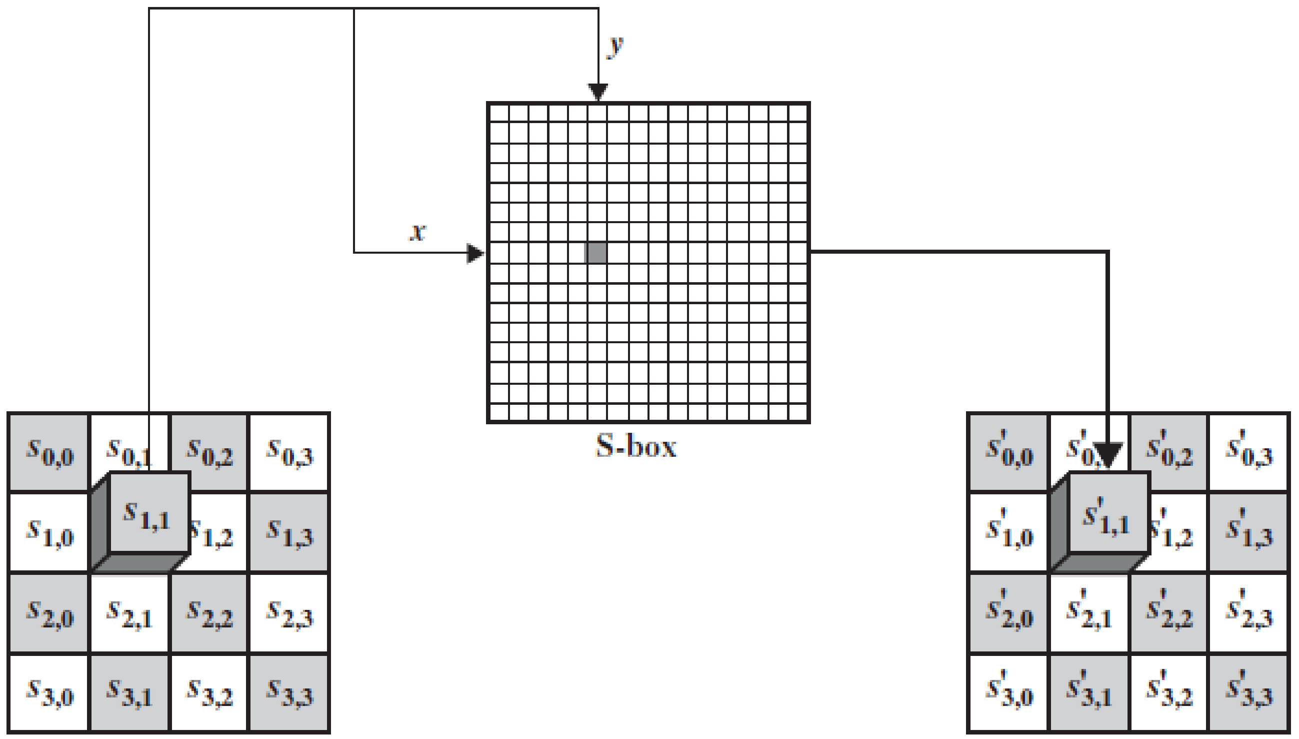
Four different stages are used, one of permutation and three of substitution:

- **Substitute bytes**: Uses an S-box to perform a byte-by-byte substitution of the block
- **Shift Rows**: A simple permutation
- **Mix Columns**: A substitution that makes use of arithmetic over $GF(2^8)$
- **Add Round Key:** A simple bitwise XOR of the current block with a portion of the expanded key



Substitute Bytes Transformation

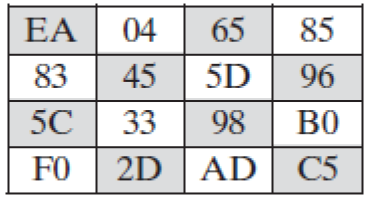
- Simple substitution of each byte
- Uses one table of 16 x 16 bytes containing a permutation of all 256 8-bit values
- Each individual byte of State is mapped into a new byte
- The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value



y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

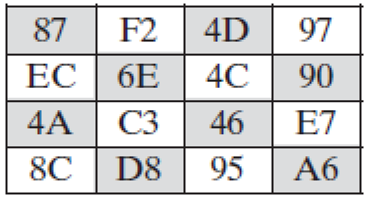
Substitute Bytes Transformation

- These row and column values serve as indexes into the S-box to select a unique 8-bit output value
- For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}
- Designed to be resistant to all known attacks
- Example:

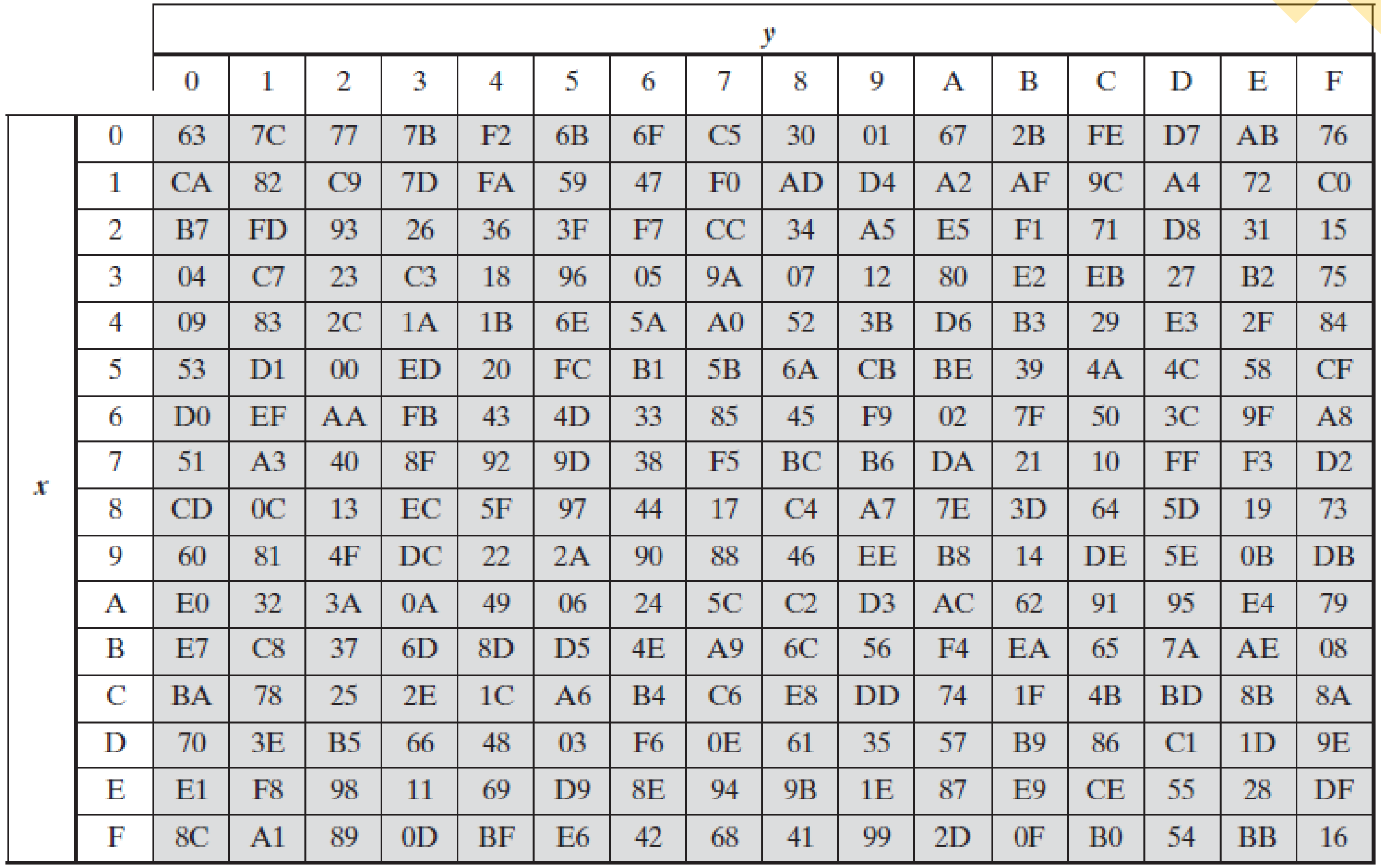


EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→



87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6



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

S-box

Substitute Bytes Transformation

- These row and column values serve as indexes into the S-box to select a unique 8-bit output value
- For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}
- Designed to be resistant to all known attacks
- Example:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

$\rightarrow$

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

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

Inverse S-box

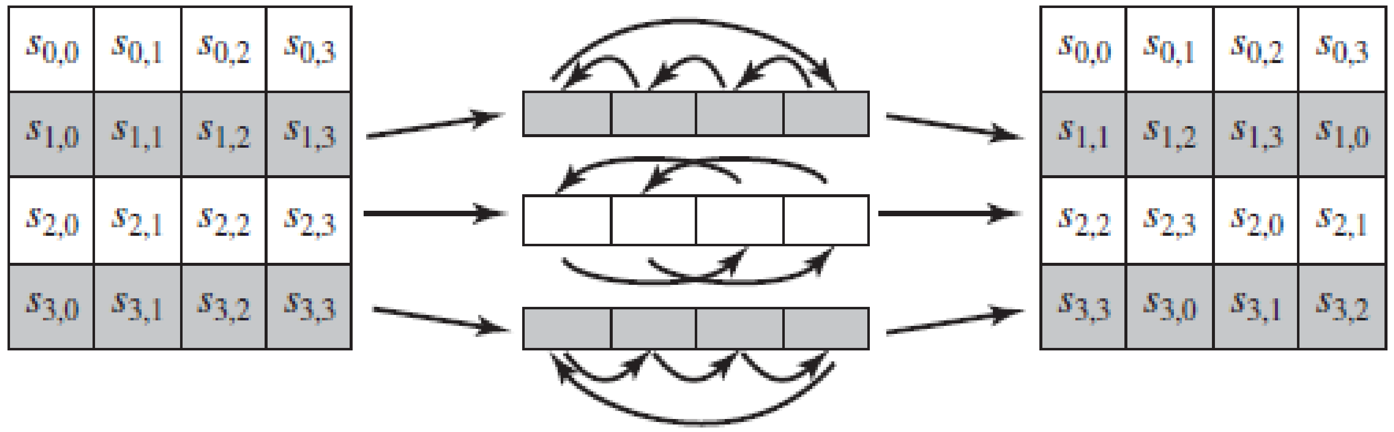
Shift Rows Transformation

- A circular byte shift in each row
- 1st row is unchanged
- 2nd row does 1 byte circular shift to left
- 3rd row does 2 byte circular shift to left
- 4th row does 3 byte circular shift to left
- Decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns
- Example:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

$\rightarrow$

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

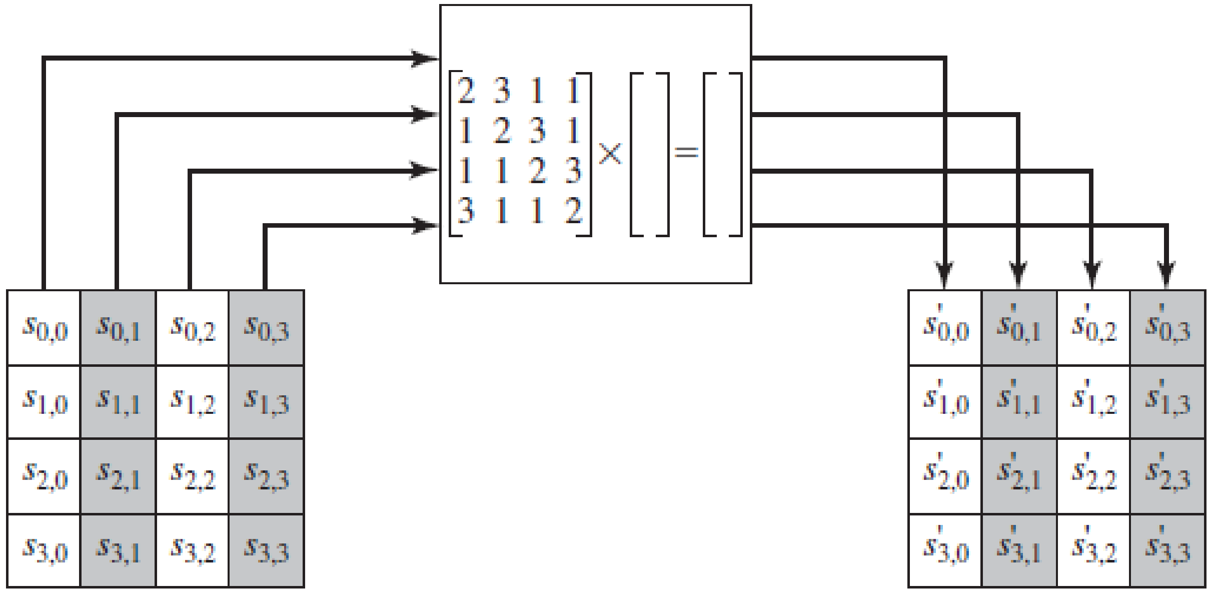
Mix Columns Transformation

- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column
- Effectively a matrix multiplication in $GF(2^8)$ using prime poly

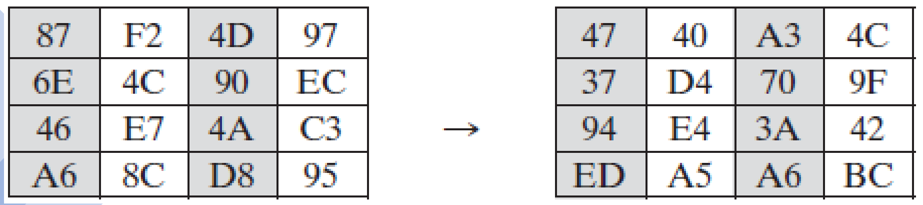
$$m(x) = x^8 + x^4 + x^3 + x + 1$$

- Example:

{02} •{87 } mod {11 b} = (1 0000 1110)mod {11 b}
= (1 0000 1110) xor (1 0001 1011) = 0001 0101



$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix} \times \begin{bmatrix}\;\end{bmatrix} = \begin{bmatrix}\;\end{bmatrix}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$



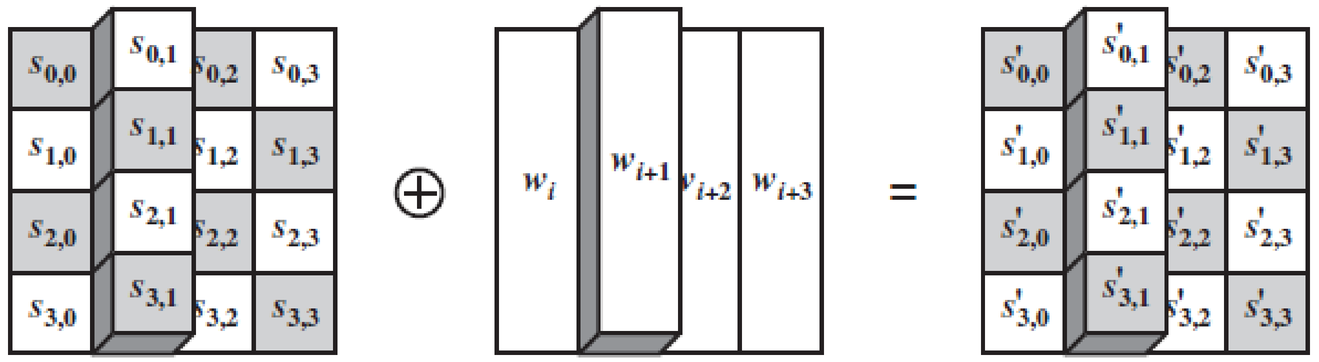
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Add Round Key Transformation

- XOR state with 128 bits of the round key
- Processed by column (though effectively a series of byte operations)
- Inverse for decryption identical
  - Since XOR own inverse, with reversed keys
- Designed to be as simple as possible
  - A form of Vernam cipher on expanded key
  - Requires other stages for complexity / security

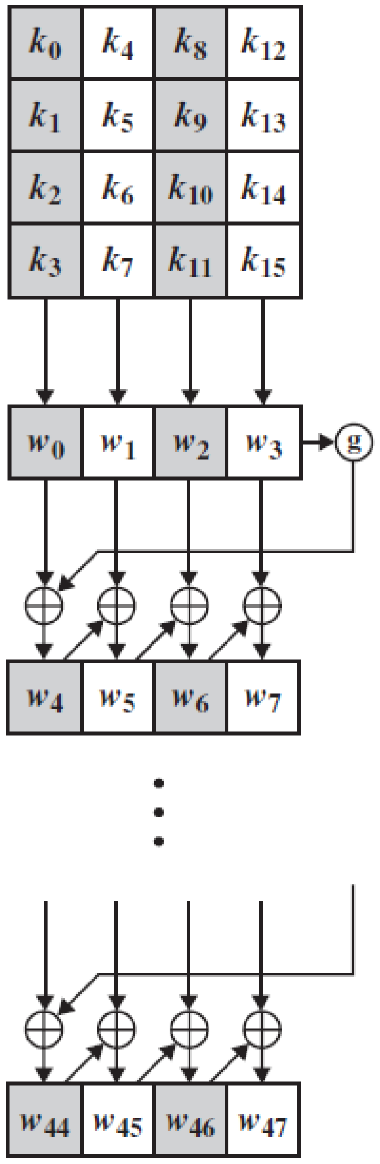


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$\oplus$
w_i w_{i+1} w_{i+2} w_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

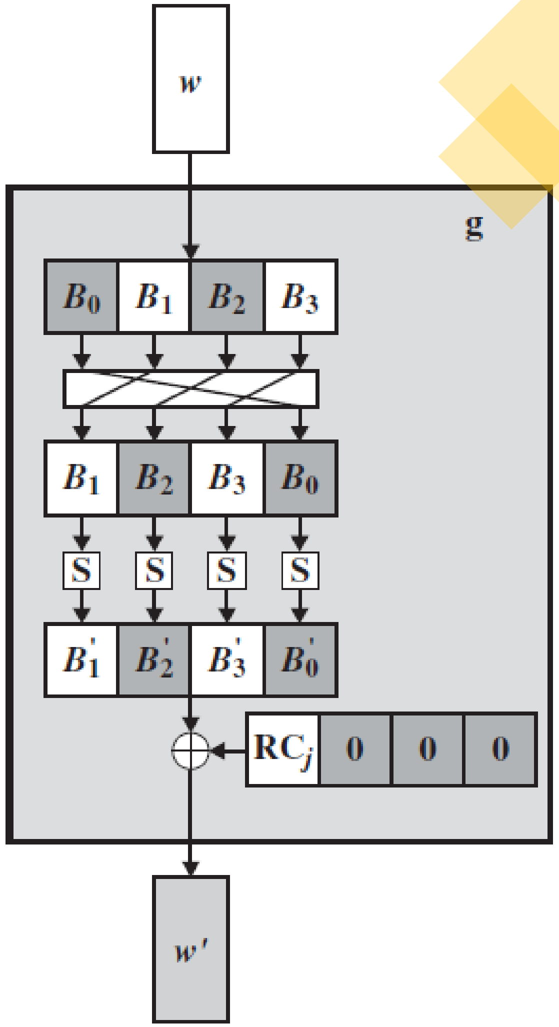
$$\begin{array}{|c|c|c|c|}\hline 47 & 40 & A3 & 4C \\ \hline 37 & D4 & 70 & 9F \\ \hline 94 & E4 & 3A & 42 \\ \hline ED & A5 & A6 & BC \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline AC & 19 & 28 & 57 \\ \hline 77 & FA & D1 & 5C \\ \hline 66 & DC & 29 & 00 \\ \hline F3 & 21 & 41 & 6A \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline EB & 59 & 8B & 1B \\ \hline 40 & 2E & A1 & C3 \\ \hline F2 & 38 & 13 & 42 \\ \hline 1E & 84 & E7 & D6 \\ \hline \end{array}$$

Key Expansion

- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- Start by copying key into first 4 words, then loop creating words that depend on values in previous & 4 places back
    - in 3 of 4 cases just XOR these together
    - 1st word in 4 has rotate + S box + XOR round constant on previous, before XOR 4th back

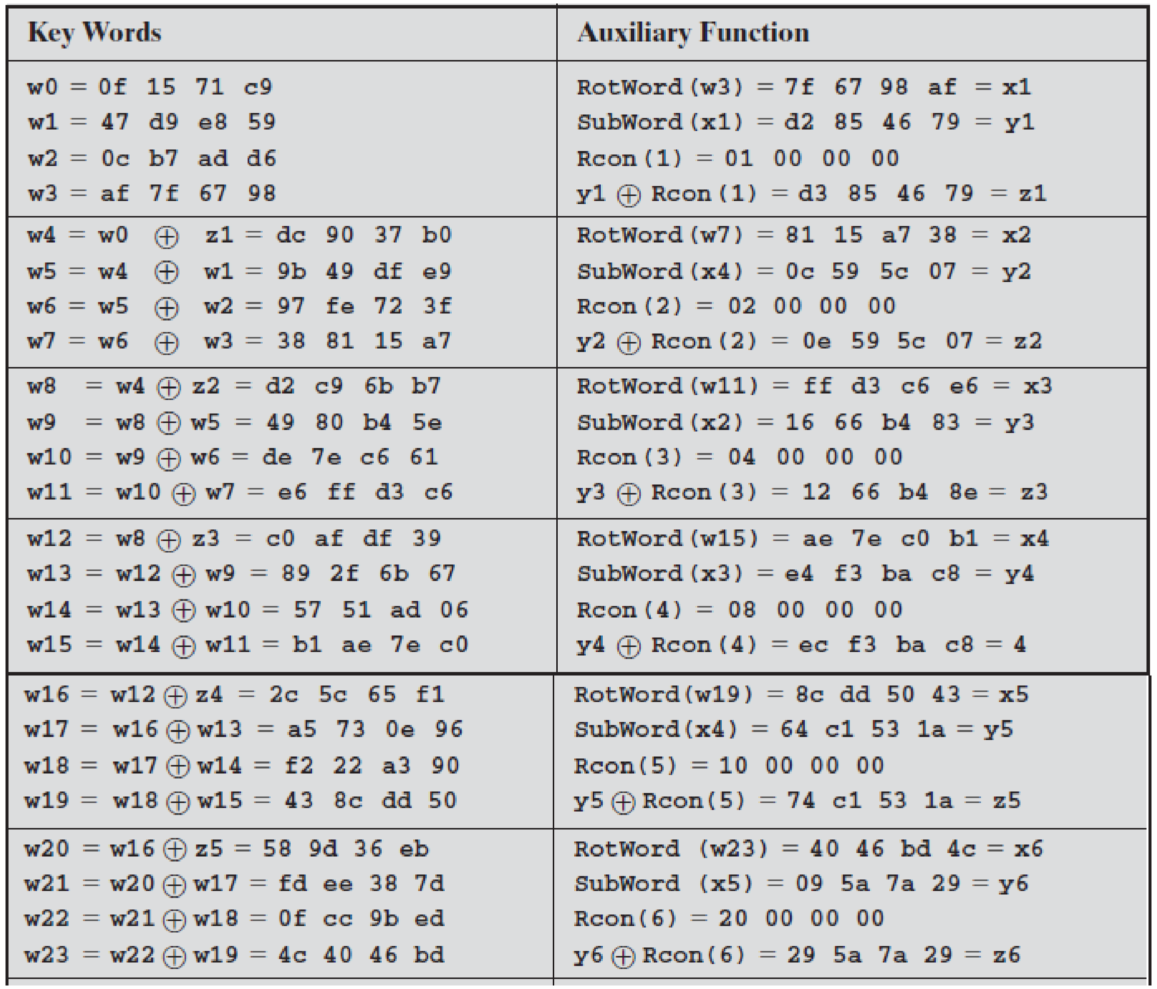

k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{44} w_{45} w_{46} w_{47}

(a) Overall algorithm

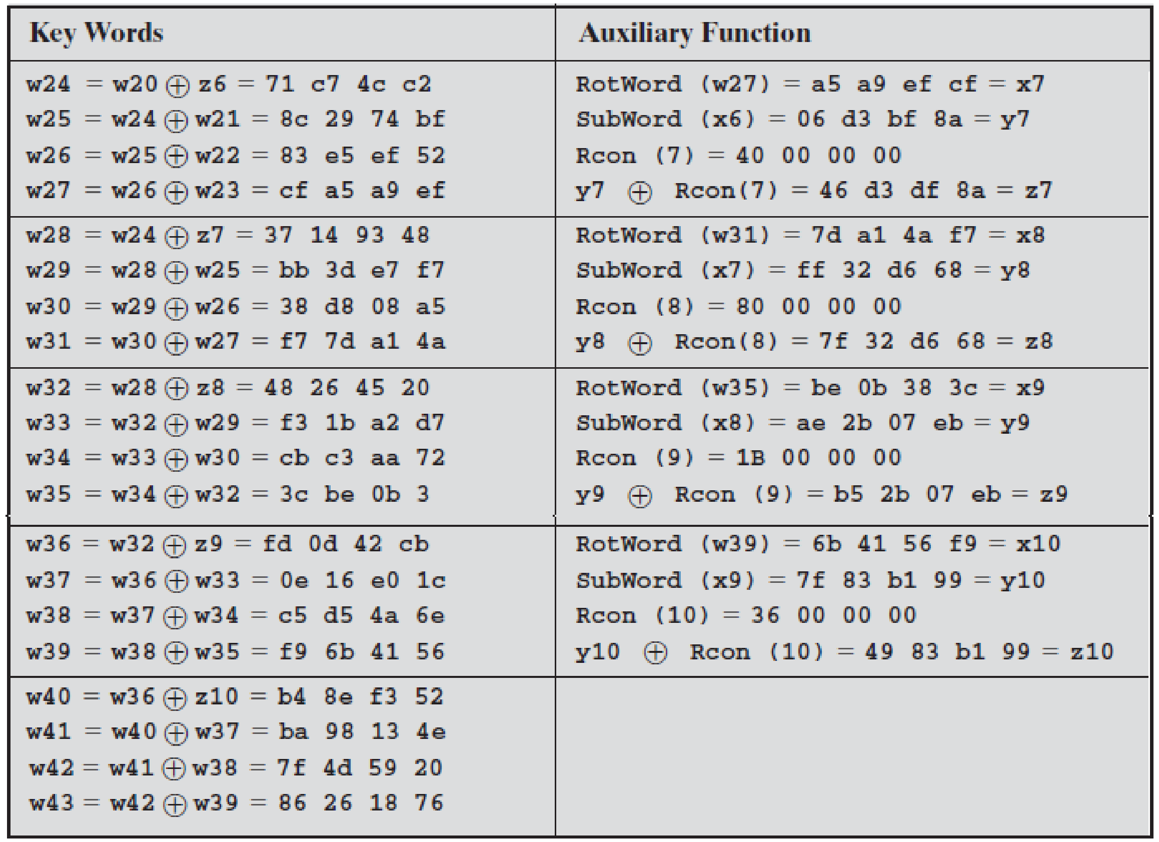


(b) Function g

Key Expansion Example



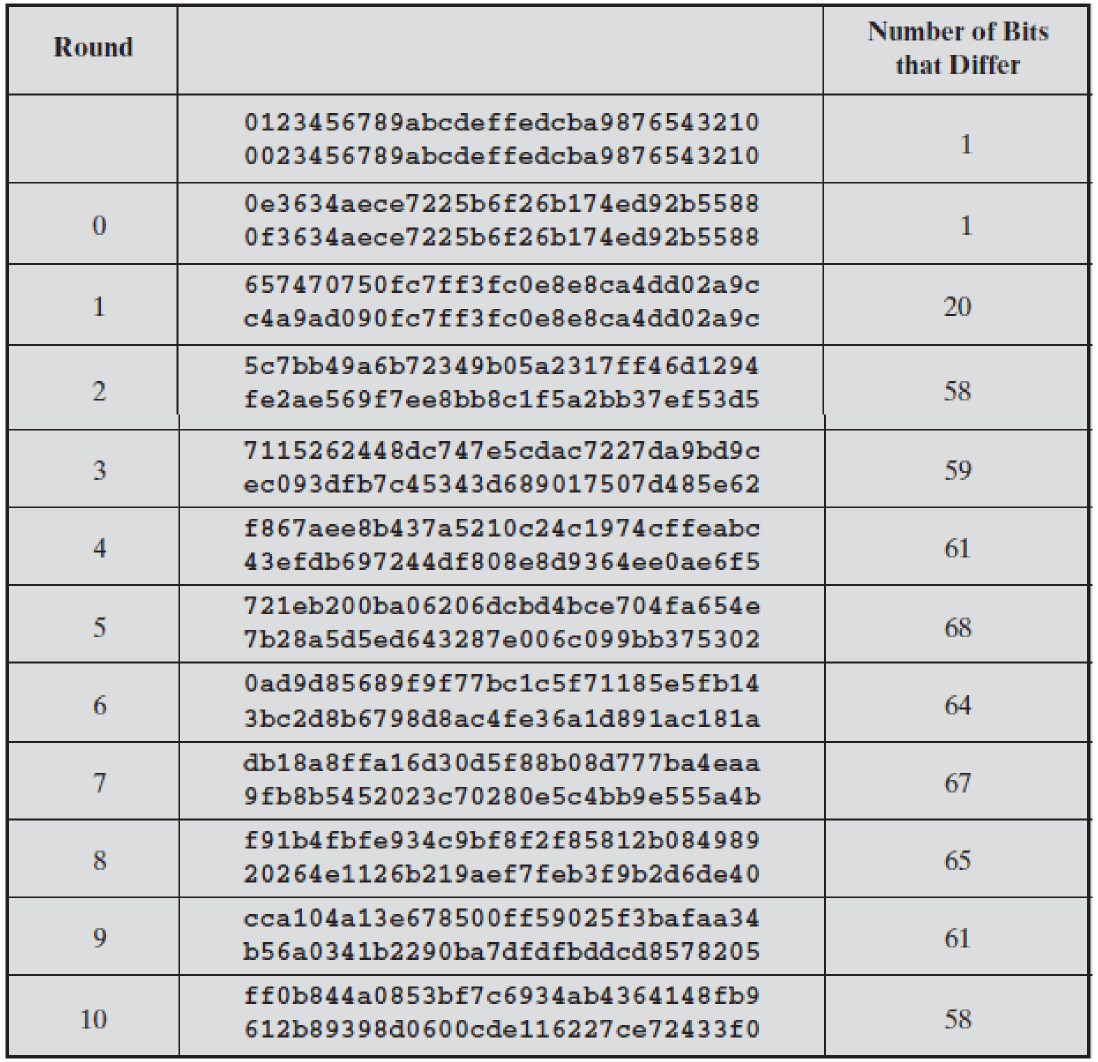
Key Words	Auxiliary Function
w0 = 0f 15 71 c9	RotWord(w3) = 7f 67 98 af = x1
w1 = 47 d9 e8 59	SubWord(x1) = d2 85 46 79 = y1
w2 = 0c b7 ad d6	Rcon(1) = 01 00 00 00
w3 = af 7f 67 98	y1 $\oplus$ Rcon(1) = d3 85 46 79 = z1
w4 = w0 $\oplus$ z1 = dc 90 37 b0	RotWord(w7) = 81 15 a7 38 = x2
w5 = w4 $\oplus$ w1 = 9b 49 df e9	SubWord(x4) = 0c 59 5c 07 = y2
w6 = w5 $\oplus$ w2 = 97 fe 72 3f	Rcon(2) = 02 00 00 00
w7 = w6 $\oplus$ w3 = 38 81 15 a7	y2 $\oplus$ Rcon(2) = 0e 59 5c 07 = z2
w8 = w4 $\oplus$ z2 = d2 c9 6b b7	RotWord(w11) = ff d3 c6 e6 = x3
w9 = w8 $\oplus$ w5 = 49 80 b4 5e	SubWord(x2) = 16 66 b4 83 = y3
w10 = w9 $\oplus$ w6 = de 7e c6 61	Rcon(3) = 04 00 00 00
w11 = w10 $\oplus$ w7 = e6 ff d3 c6	y3 $\oplus$ Rcon(3) = 12 66 b4 8e = z3
w12 = w8 $\oplus$ z3 = c0 af df 39	RotWord(w15) = ae 7e c0 b1 = x4
w13 = w12 $\oplus$ w9 = 89 2f 6b 67	SubWord(x3) = e4 f3 ba c8 = y4
w14 = w13 $\oplus$ w10 = 57 51 ad 06	Rcon(4) = 08 00 00 00
w15 = w14 $\oplus$ w11 = b1 ae 7e c0	y4 $\oplus$ Rcon(4) = ec f3 ba c8 = 4
w16 = w12 $\oplus$ z4 = 2c 5c 65 f1	RotWord(w19) = 8c dd 50 43 = x5
w17 = w16 $\oplus$ w13 = a5 73 0e 96	SubWord(x4) = 64 c1 53 1a = y5
w18 = w17 $\oplus$ w14 = f2 22 a3 90	Rcon(5) = 10 00 00 00
w19 = w18 $\oplus$ w15 = 43 8c dd 50	y5 $\oplus$ Rcon(5) = 74 c1 53 1a = z5
w20 = w16 $\oplus$ z5 = 58 9d 36 eb	RotWord (w23) = 40 46 bd 4c = x6
w21 = w20 $\oplus$ w17 = fd ee 38 7d	SubWord (x5) = 09 5a 7a 29 = y6
w22 = w21 $\oplus$ w18 = 0f cc 9b ed	Rcon(6) = 20 00 00 00
w23 = w22 $\oplus$ w19 = 4c 40 46 bd	y6 $\oplus$ Rcon(6) = 29 5a 7a 29 = z6



Key Words	Auxiliary Function
w24 = w20 $\oplus$ z6 = 71 c7 4c c2	RotWord (w27) = a5 a9 ef cf = x7
w25 = w24 $\oplus$ w21 = 8c 29 74 bf	SubWord (x6) = 06 d3 bf 8a = y7
w26 = w25 $\oplus$ w22 = 83 e5 ef 52	Rcon (7) = 40 00 00 00
w27 = w26 $\oplus$ w23 = cf a5 a9 ef	y7 $\oplus$ Rcon(7) = 46 d3 df 8a = z7
w28 = w24 $\oplus$ z7 = 37 14 93 48	RotWord (w31) = 7d a1 4a f7 = x8
w29 = w28 $\oplus$ w25 = bb 3d e7 f7	SubWord (x7) = ff 32 d6 68 = y8
w30 = w29 $\oplus$ w26 = 38 d8 08 a5	Rcon (8) = 80 00 00 00
w31 = w30 $\oplus$ w27 = f7 7d a1 4a	y8 $\oplus$ Rcon(8) = 7f 32 d6 68 = z8
w32 = w28 $\oplus$ z8 = 48 26 45 20	RotWord (w35) = be 0b 38 3c = x9
w33 = w32 $\oplus$ w29 = f3 1b a2 d7	SubWord (x8) = ae 2b 07 eb = y9
w34 = w33 $\oplus$ w30 = cb c3 aa 72	Rcon (9) = 1B 00 00 00
w35 = w34 $\oplus$ w32 = 3c be 0b 3	y9 $\oplus$ Rcon (9) = b5 2b 07 eb = z9
w36 = w32 $\oplus$ z9 = fd 0d 42 cb	RotWord (w39) = 6b 41 56 f9 = x10
w37 = w36 $\oplus$ w33 = 0e 16 e0 1c	SubWord (x9) = 7f 83 b1 99 = y10
w38 = w37 $\oplus$ w34 = c5 d5 4a 6e	Rcon (10) = 36 00 00 00
w39 = w38 $\oplus$ w35 = f9 6b 41 56	y10 $\oplus$ Rcon (10) = 49 83 b1 99 = z10
w40 = w36 $\oplus$ z10 = b4 8e f3 52	
w41 = w40 $\oplus$ w37 = ba 98 13 4e	
w42 = w41 $\oplus$ w38 = 7f 4d 59 20	
w43 = w42 $\oplus$ w39 = 86 26 18 76	

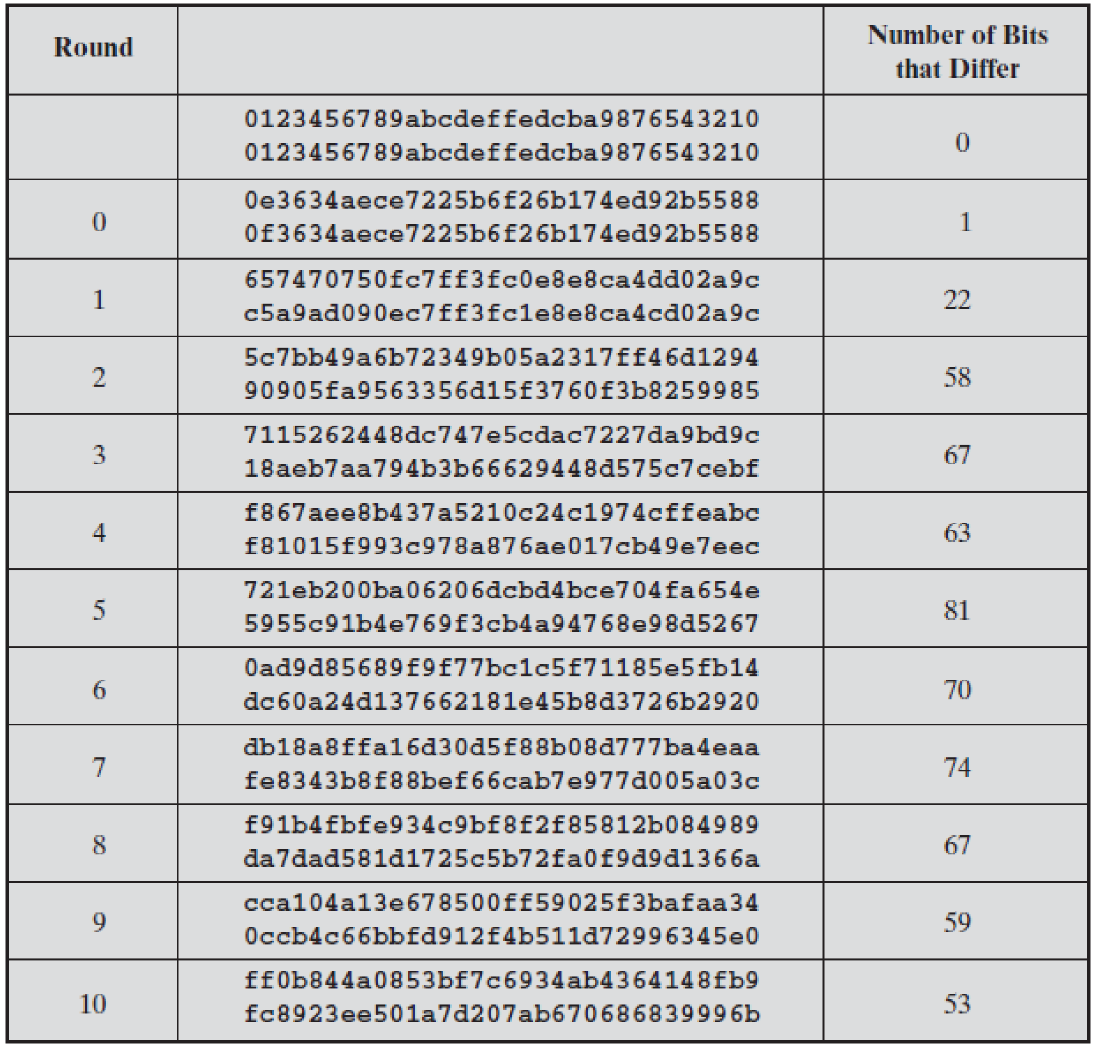
Avalanche Effect in AES

Table 5.5 Avalanche Effect in AES: Change in Plaintext



Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Table 5.6 Avalanche Effect in AES: Change in Key



Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Principles Of Public-Key Cryptosystems

The RSA Algorithm

Diffie-Hellman key exchange

ElGamal cryptograpy system

Terminology Related to Asymmetric Encryption

Asymmetric Keys

- Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate

- A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm

- A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)

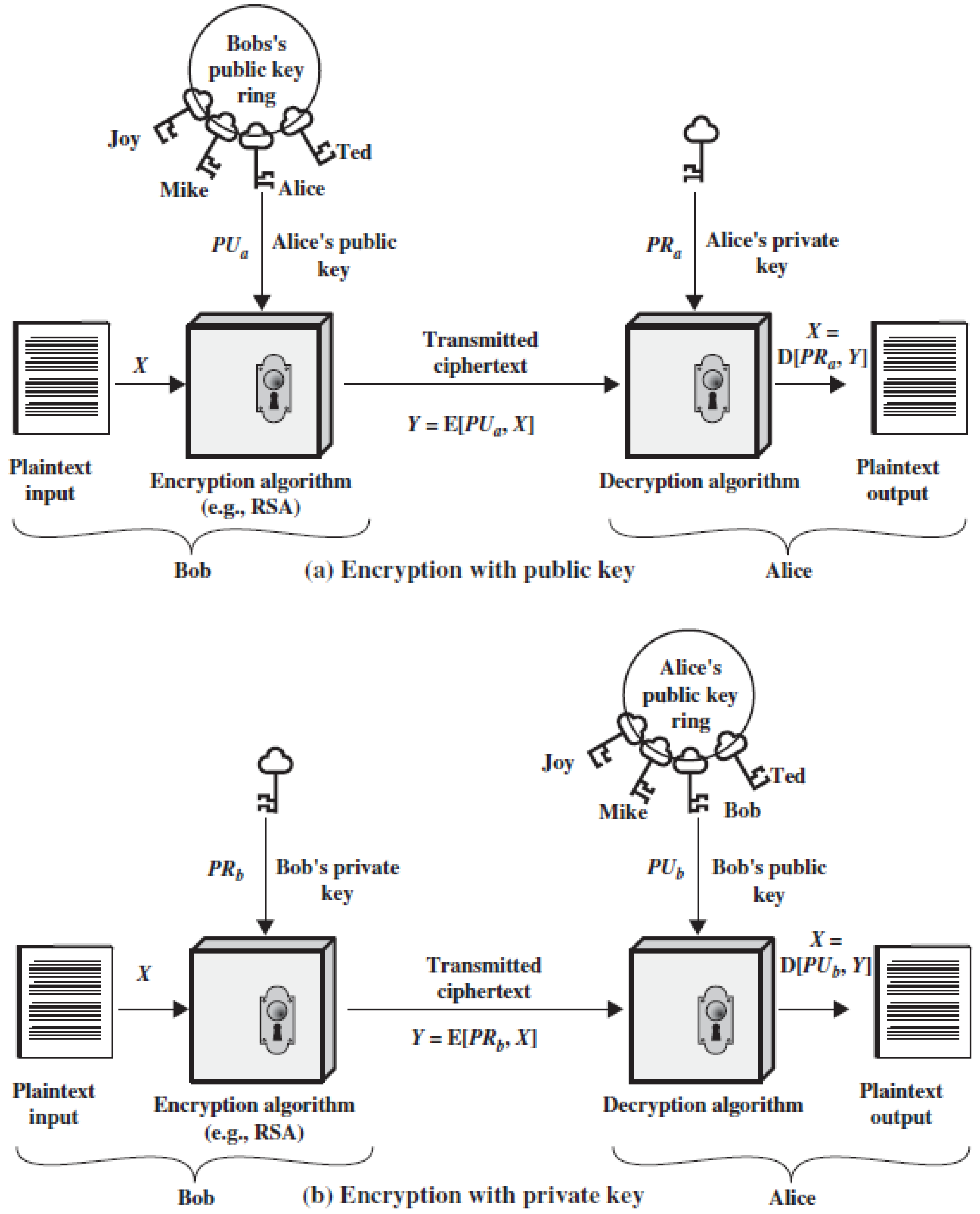
- A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Principles Of Public-key Cryptosystems

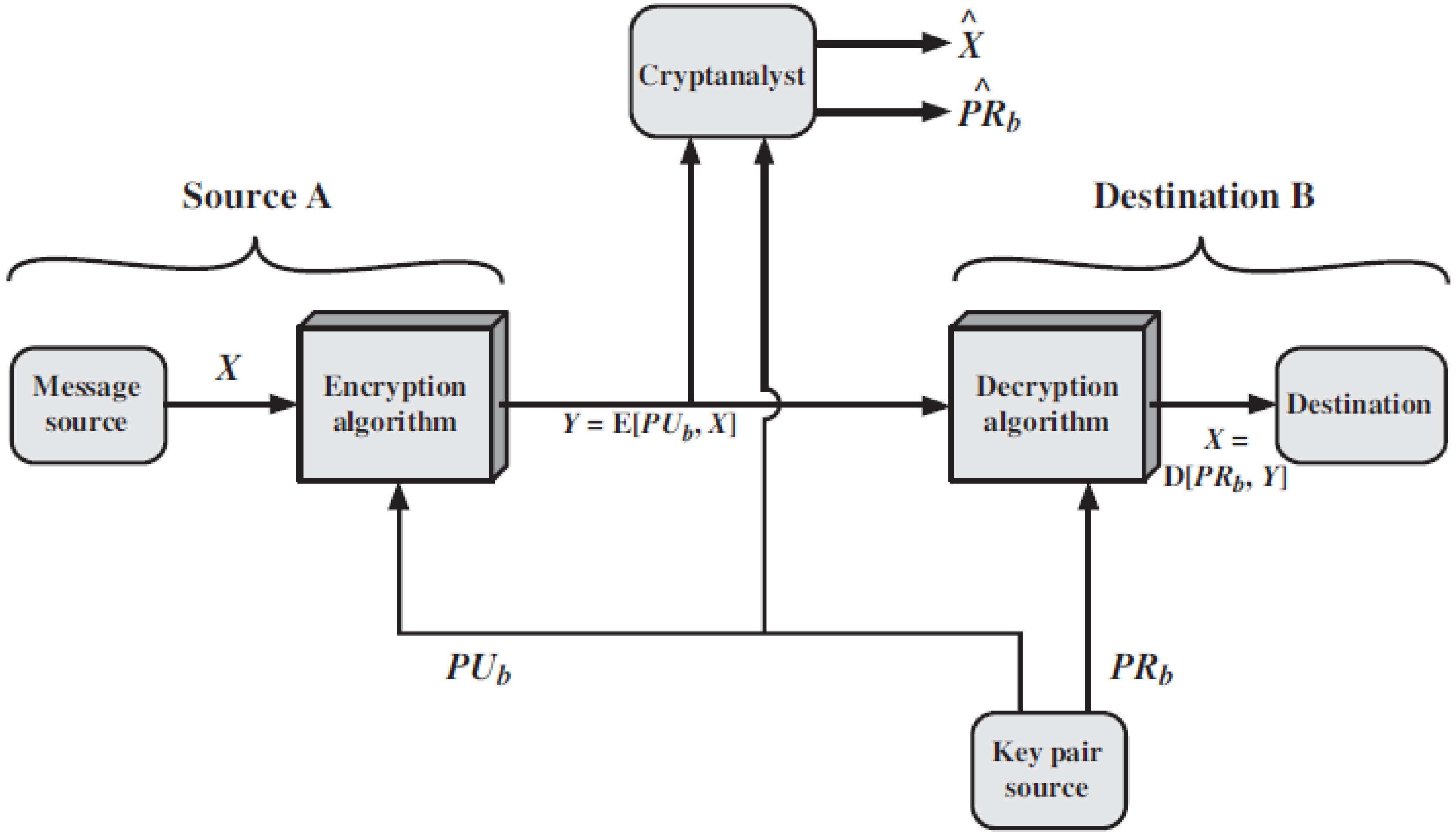
- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption
  - Key distribution
  - Digital signatures
- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption
  - It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
  - Either of the two related keys can be used for encryption, with the other used for decryption.

Public-Key Cryptosystems

- A public-key encryption scheme has five ingredients
- Plaintext
- Encryption algorithm
- Public and private keys
- Ciphertext
- Decryption algorithm

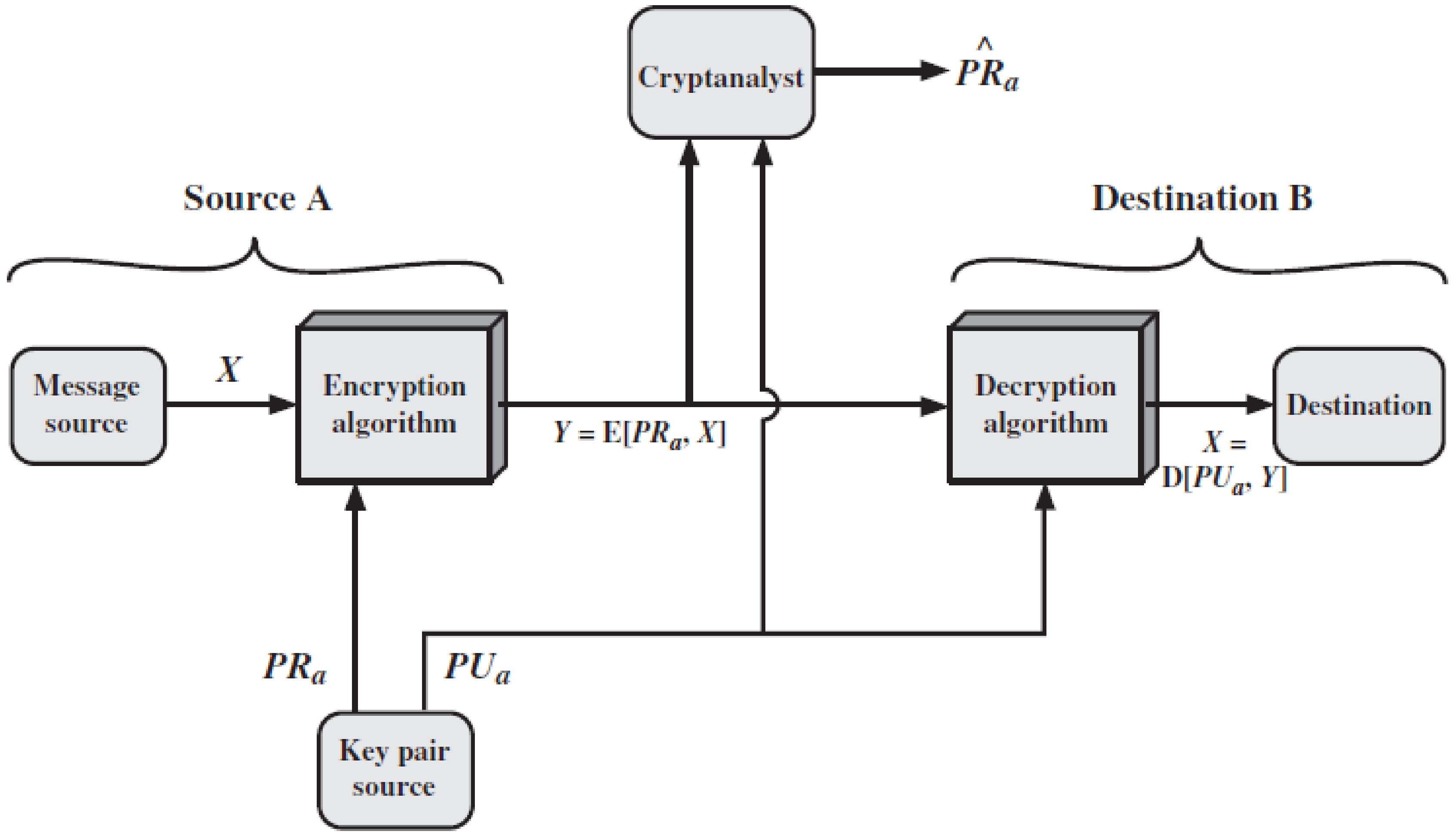


Public-Key Cryptosystems



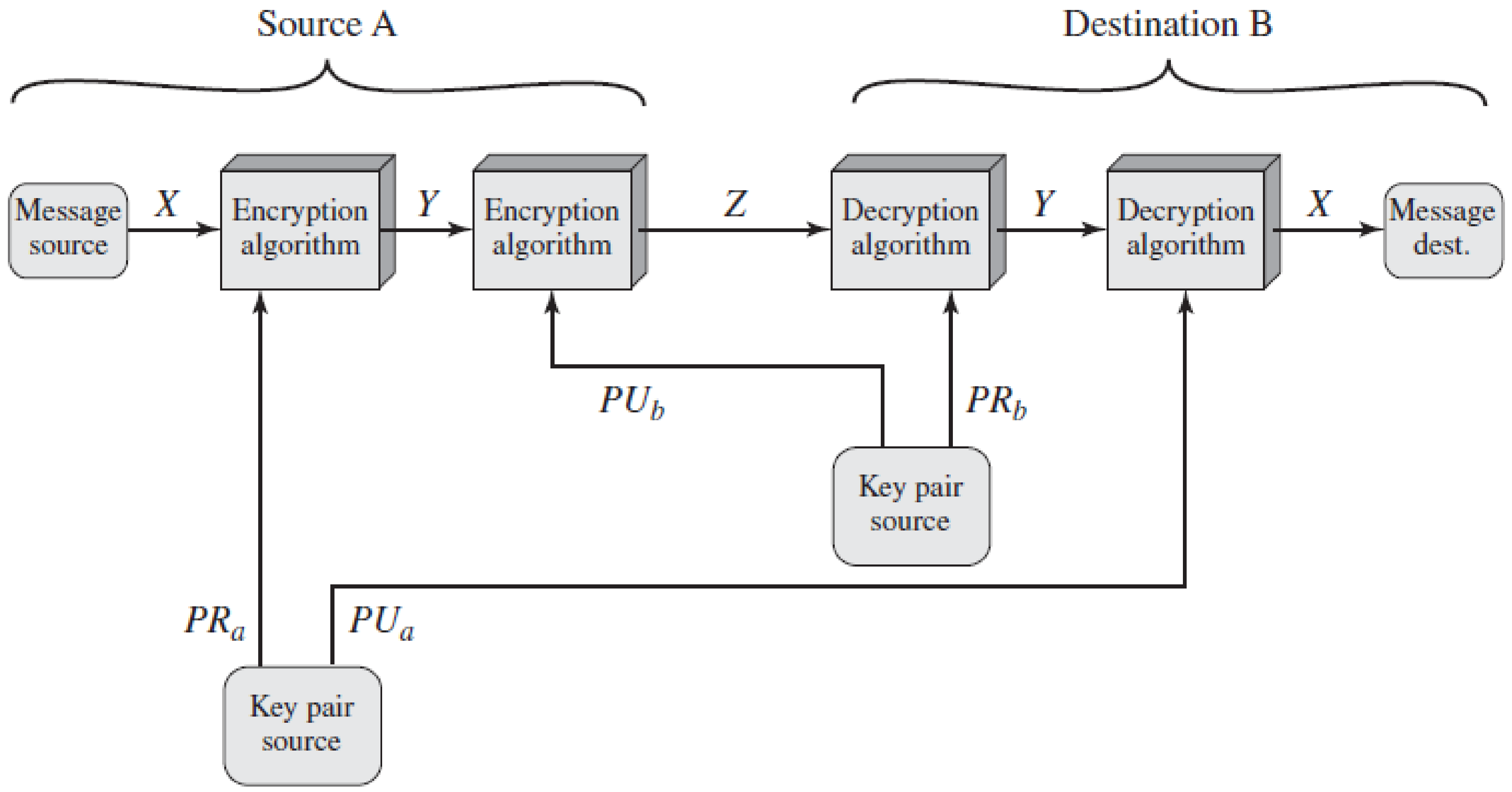
Confidentiality

Public-Key Cryptosystems



Authentication

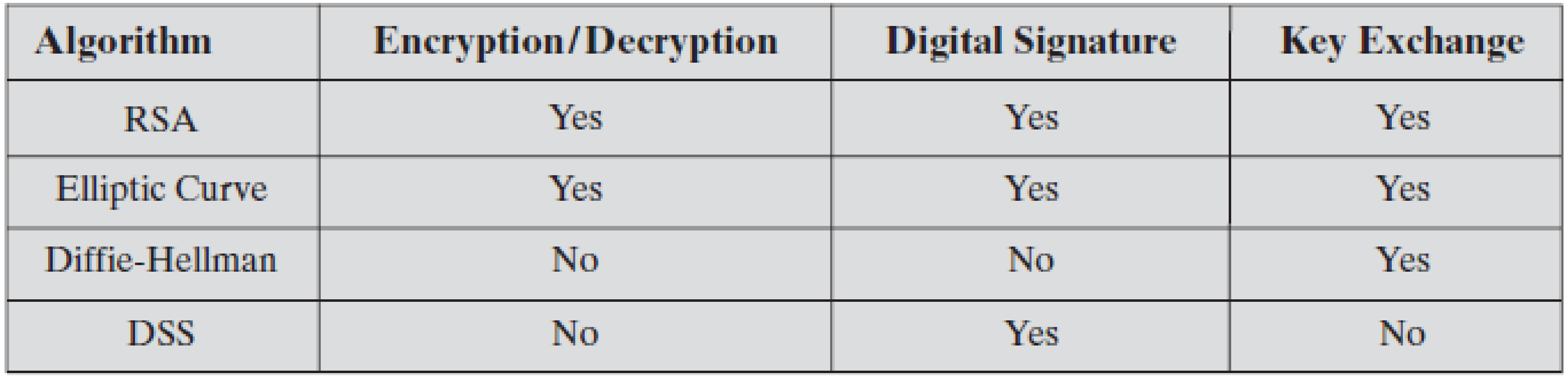
Public-Key Cryptosystems



Authentication and Confidentiality

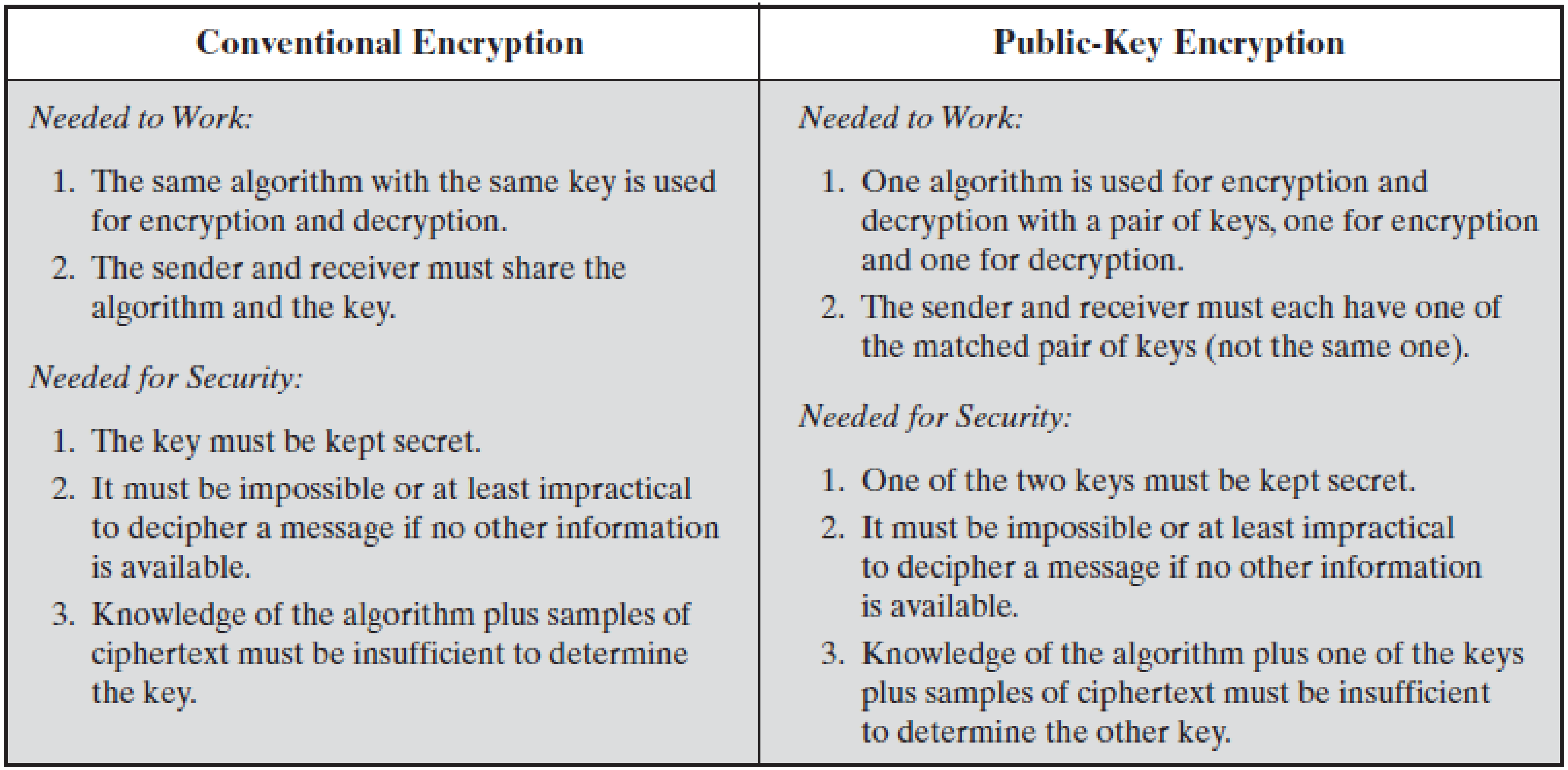
Applications for Public-Key Cryptosystems

- **Encryption /decryption**: The sender encrypts a message with the recipient’s public key.
- **Digital signature**: The sender “signs” a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange**: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.



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

Conventional and Public-Key Encryption



Conventional Encryption	**Public-Key Encryption**
Needed to Work:	*Needed to Work:*
1. The same algorithm with the same key is used for encryption and decryption.	1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.
2. The sender and receiver must share the algorithm and the key.	2. The sender and receiver must each have one of the matched pair of keys (not the same one).
Needed for Security:	*Needed for Security:*
1. The key must be kept secret.	1. One of the two keys must be kept secret.
2. It must be impossible or at least impractical to decipher a message if no other information is available.	2. It must be impossible or at least impractical to decipher a message if no other information is available.
3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted,M, to generate the corresponding ciphertext:

   $$C = E(PUb,M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

   $$M = D(PRb, C) = D[PRb, E(PUb,M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PUb, to determine the private key, PRb
5. It is computationally infeasible for an adversary, knowing the public key, PUb, and a ciphertext, C, to recover the original message,M.
6. The two keys can be applied in either order (optional)

Requirements for Public-Key Cryptography

- Those 6 requirements lead to need **for trap door one way function**
- is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known
- The development of a practical public-key scheme depends on discovery of a suitable trap-door one-way function

Public-Key Cryptanalysis

- public-key encryption scheme is vulnerable to a brute-force attack
  - Public-key systems depend on the use of some sort of invertible mathematical function.
  - The key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption
- Another form of attack is to find some way to compute the private key given the public key
- A probable-message attack

The RSA Algorithm

- One of the first successful responses to the challenge was developed in 1977 by Ron **R**ivest, Adi **S**hamir, and Len **A**dleman at MIT.
- The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n
- Encryption of plaintext M where M < n

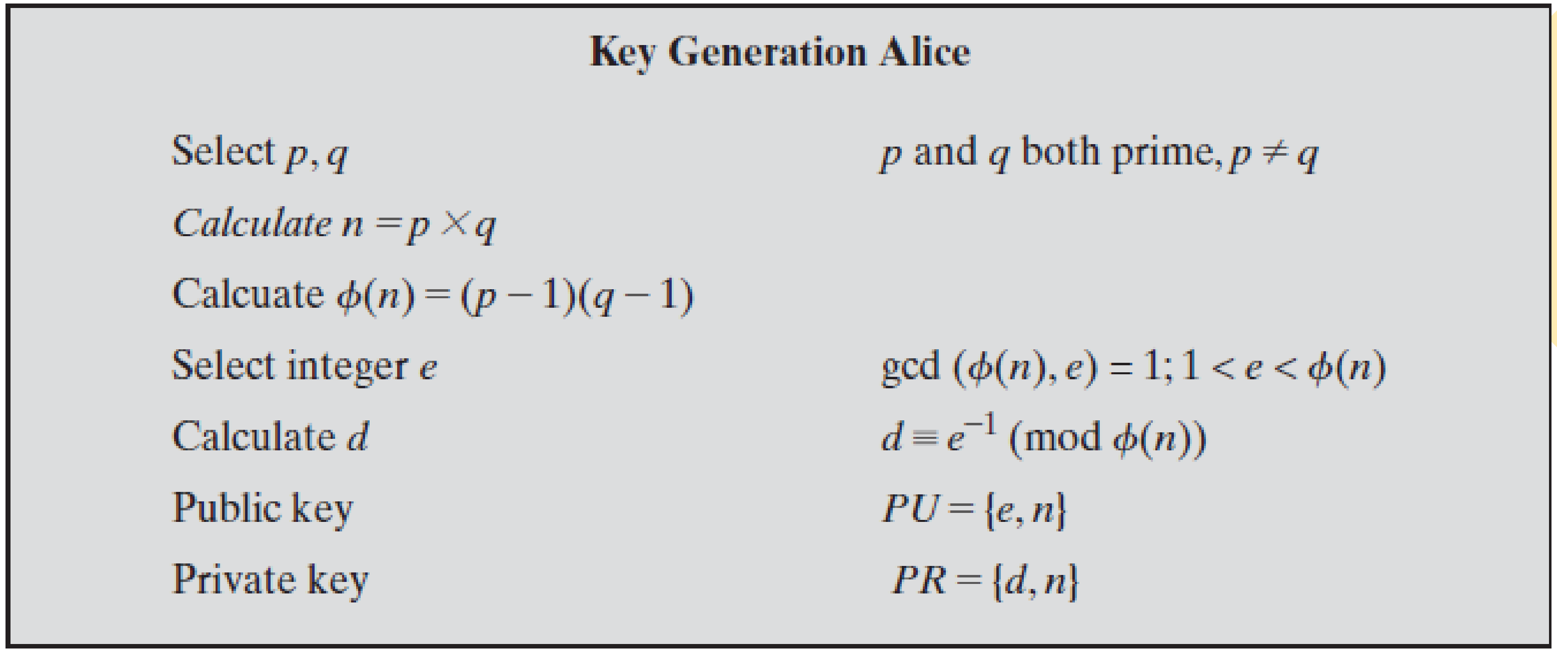
$$C = M^e \bmod n$$

- Decryption of ciphertext C

$$M = C^d \bmod n$$

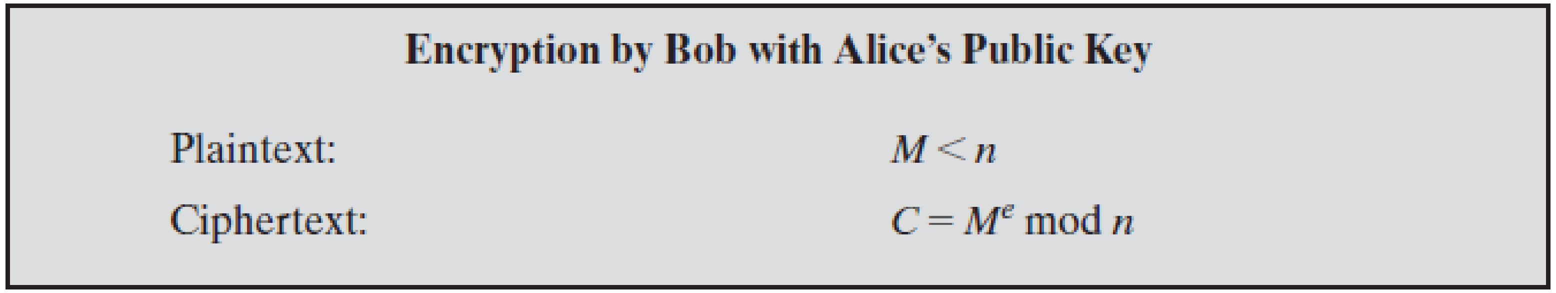
The RSA Algorithm

If p= 17 and q= 11
n=187
$\emptyset(n) = 160$
If e=7
Then d = 23



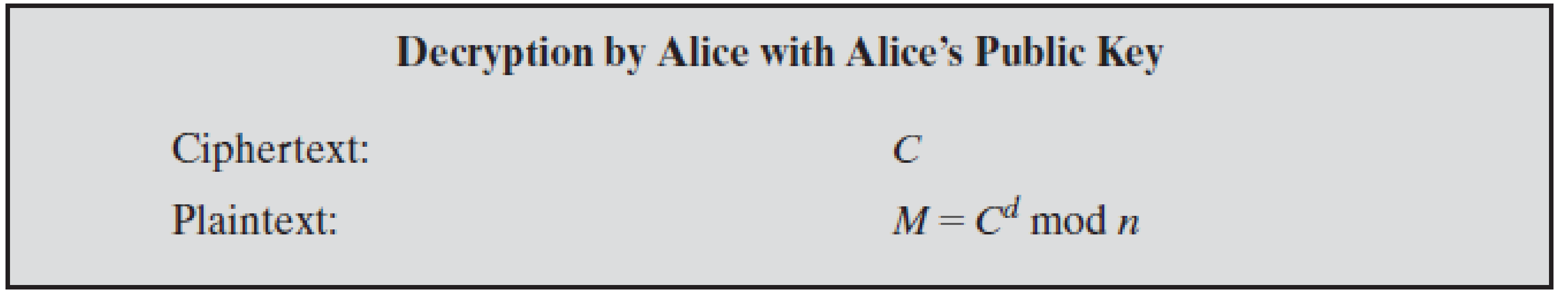
Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$



Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$



Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

The RSA Algorithm

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

$$ed \bmod \phi(n) = 1$$

$$\phi(pq) = (p - 1)(q - 1)$$

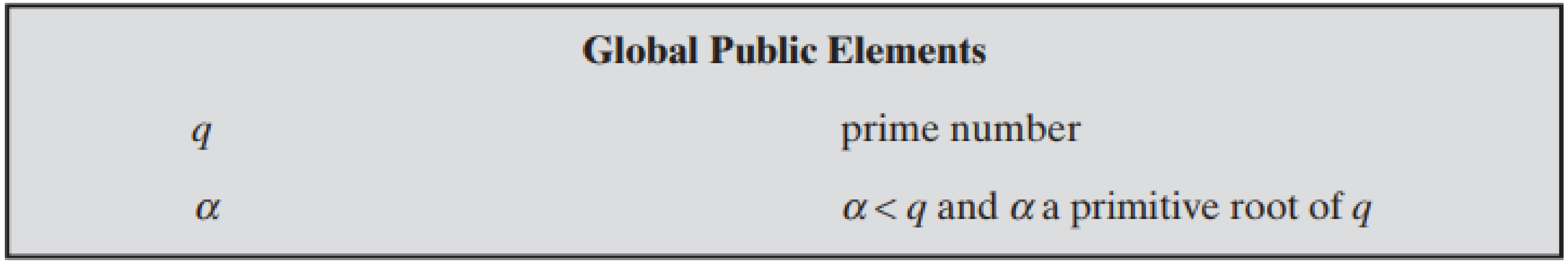
The Security of RSA

- Four possible approaches to attacking the RSA algorithm are
- **Brute force**: This involves trying all possible private keys.
- **Mathematical attacks**: There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks**: These depend on the running time of the decryption algorithm.
- **Chosen ciphertext attacks**: This type of attack exploits properties of the RSA algorithm.

Diffie-Hellman key exchange

- The first published public-key algorithm appeared
- The purpose of the algorithm is to enable two users to **securely exchange a key** that can then be used for subsequent encryption of messages.
- **Limited** to the exchange of secret values.
- Uses two publicly known numbers:

$$\boldsymbol{(q) and\ (\alpha)}$$



Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Diffie-Hellman key exchange

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

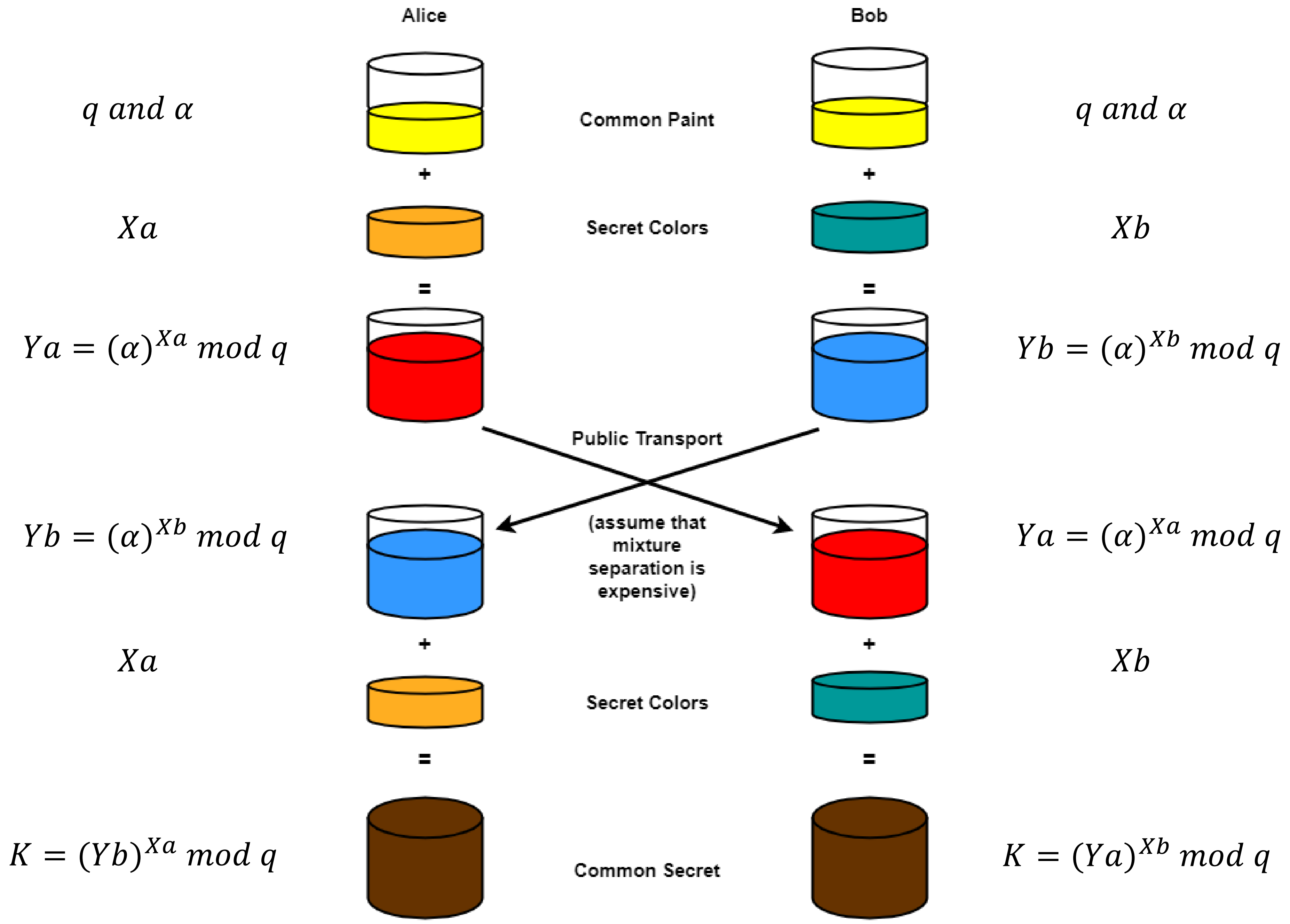
Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

Ex.
$q=353$ and $\alpha=3$
$XA = 97$ and $XB = 233$


Alice
Bob
q and α
Common Paint
q and α
+
+
Xa
Secret Colors
Xb
=
=
$Ya = (\alpha)^{Xa} \bmod q$
$Yb = (\alpha)^{Xb} \bmod q$
Public Transport
$Yb = (\alpha)^{Xb} \bmod q$
(assume that mixture separation is expensive)
$Ya = (\alpha)^{Xa} \bmod q$
+
+
Xa
Xb
Secret Colors
=
=
$K = (Yb)^{Xa} \bmod q$
Common Secret
$K = (Ya)^{Xb} \bmod q$

Man-in-the-Middle Attack

- The protocol that relies on Diffie-Hellman key exchange is insecure against a man-in-the-middle attack.
- It is vulnerable to such an attack because it does not authenticate the participants.
- This vulnerability can be overcome with the use of digital signatures and public-key certificates

ElGamal cryptography system

- In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique
- Used in some form in a number of standards including the digital signature standard (DSS), and the S/MIME e-mail standard.

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

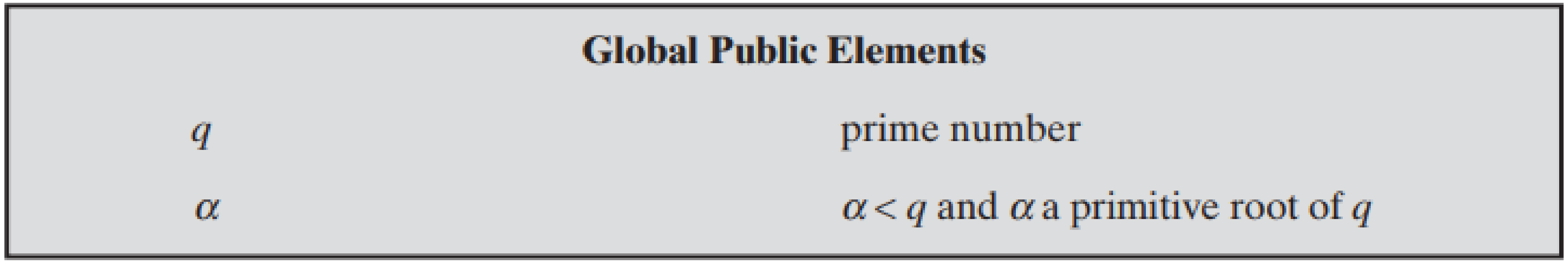
Diffie-Hellman key exchange

ElGamal cryptograpy system

Diffie-Hellman key exchange

- The first published public-key algorithm appeared
- The purpose of the algorithm is to enable two users to **securely exchange a key** that can then be used for subsequent encryption of messages.
- **Limited** to the exchange of secret values.
- Uses two publicly known numbers:

$$(\boldsymbol{q}) \boldsymbol{and} \ (\boldsymbol{\alpha})$$



Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Diffie-Hellman key exchange

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

Ex.
q=353 and α=3
XA = 97 and XB = 233

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

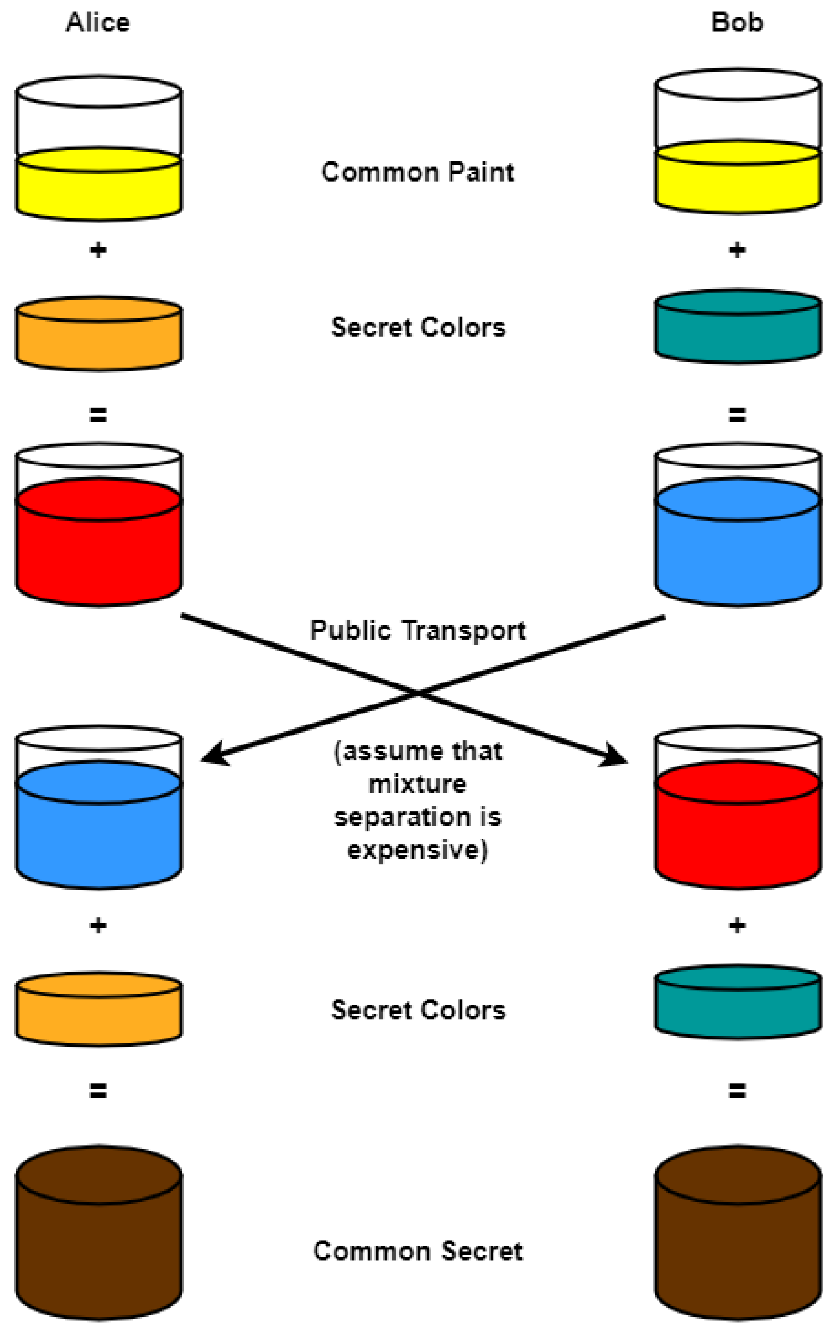
Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

$q \ and \ \alpha$ $\qquad$ $q \ and \ \alpha$

Xa $\qquad$ Xb

$Ya = (\alpha)^{Xa} \ mod \ q$ $\qquad$ $Yb = (\alpha)^{Xb} \ mod \ q$



$Yb = (\alpha)^{Xb} \ mod \ q$ $\qquad$ $Ya = (\alpha)^{Xa} \ mod \ q$

Xa $\qquad$ Xb

$K = (Yb)^{Xa} \ mod \ q$ $\qquad$ $K = (Ya)^{Xb} \ mod \ q$

Man-in-the-Middle Attack

- The protocol that relies on Diffie-Hellman key exchange is insecure against a man-in-the-middle attack.
- It is vulnerable to such an attack because it does not authenticate the participants.
- This vulnerability can be overcome with the use of digital signatures and public-key certificates

ElGamal cryptography system

- In 1984, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique
- Used in some form in a number of standards including the digital signature standard (DSS), and the S/MIME e-mail standard.

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

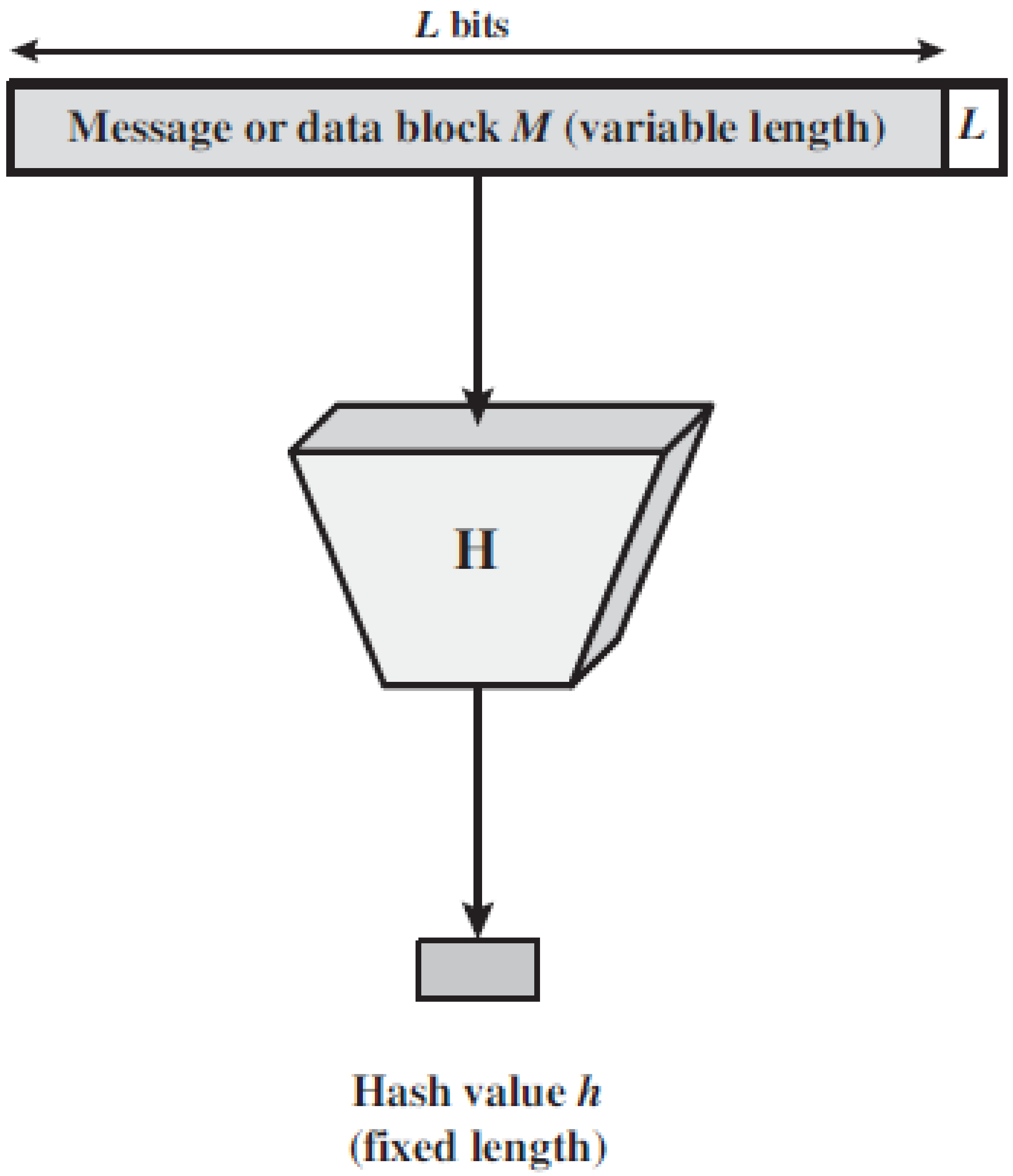
Cryptographic Hash Functions

Introduction

- A **hash function** H accepts a variable-length block of data M as input and produces a fixed-size hash value h
- the principal object of a hash function is data integrity
- A change to any bit or bits in M results, with high probability, in a change to the hash code.
- The kind of hash function needed for security applications is referred to as a **cryptographic hash function**
- it is computationally infeasible
- Because of these characteristics, hash functions are often used to determine whether or not data has changed

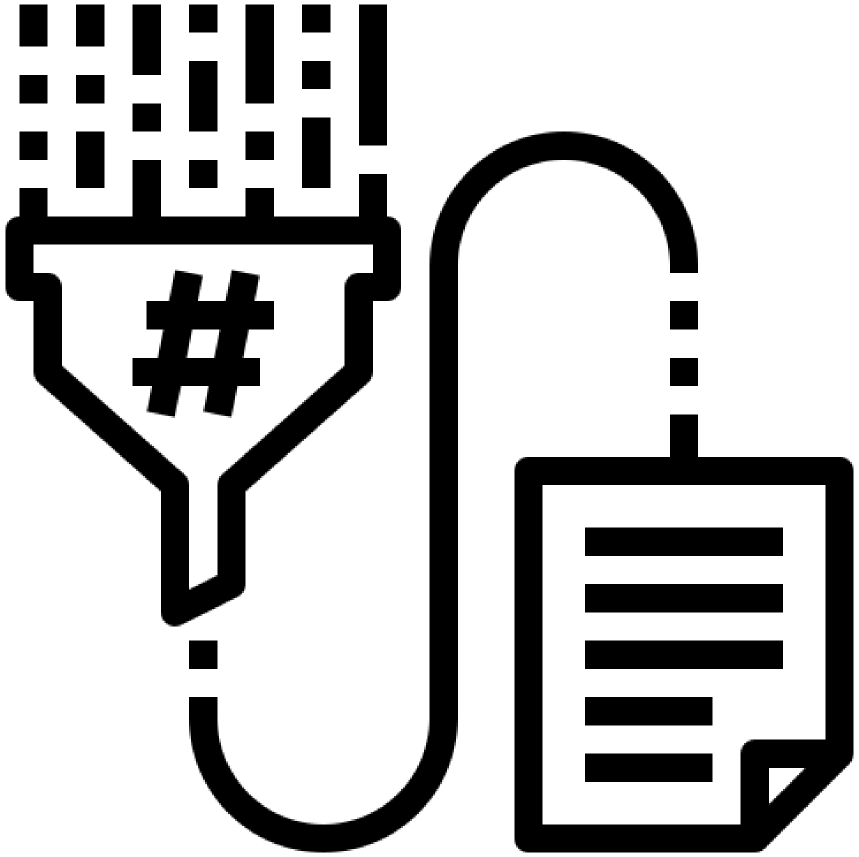
Introduction

- Typically, the input is padded out to an integer multiple of some fixed length and the padding includes the value of the length of the original message in bits.
- The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.



Applications Of Cryptographic Hash Functions

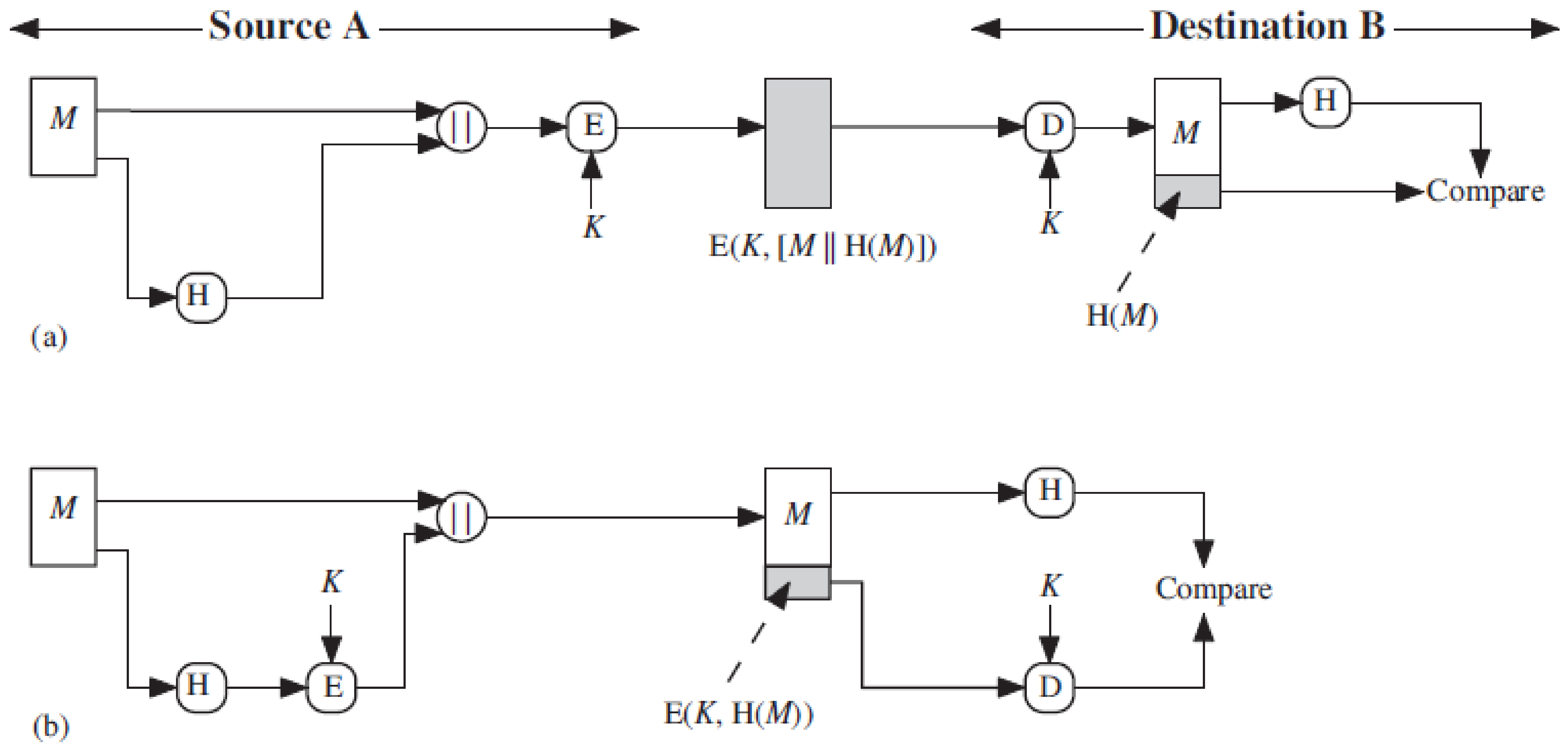
- Message Authentication
- Digital Signatures
- One-way password file
- Intrusion detection
- Virus detection



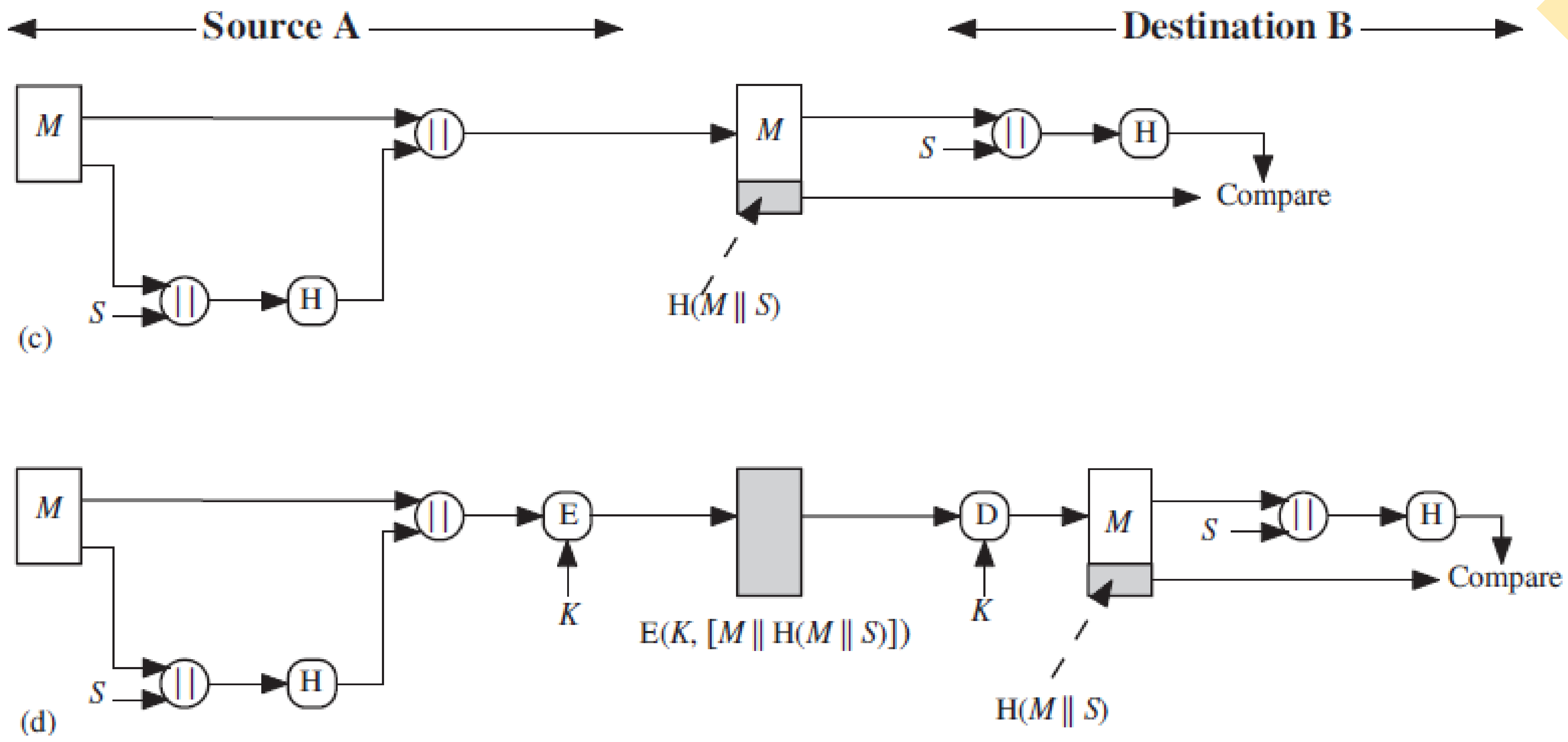
Message Authentication

- Message authentication is a mechanism or service used to verify the integrity of a message
- Message authentication assures that data received are exactly as sent
- The authentication mechanism assures that purported identity of the sender is valid
- When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**

Message Authentication



Message Authentication

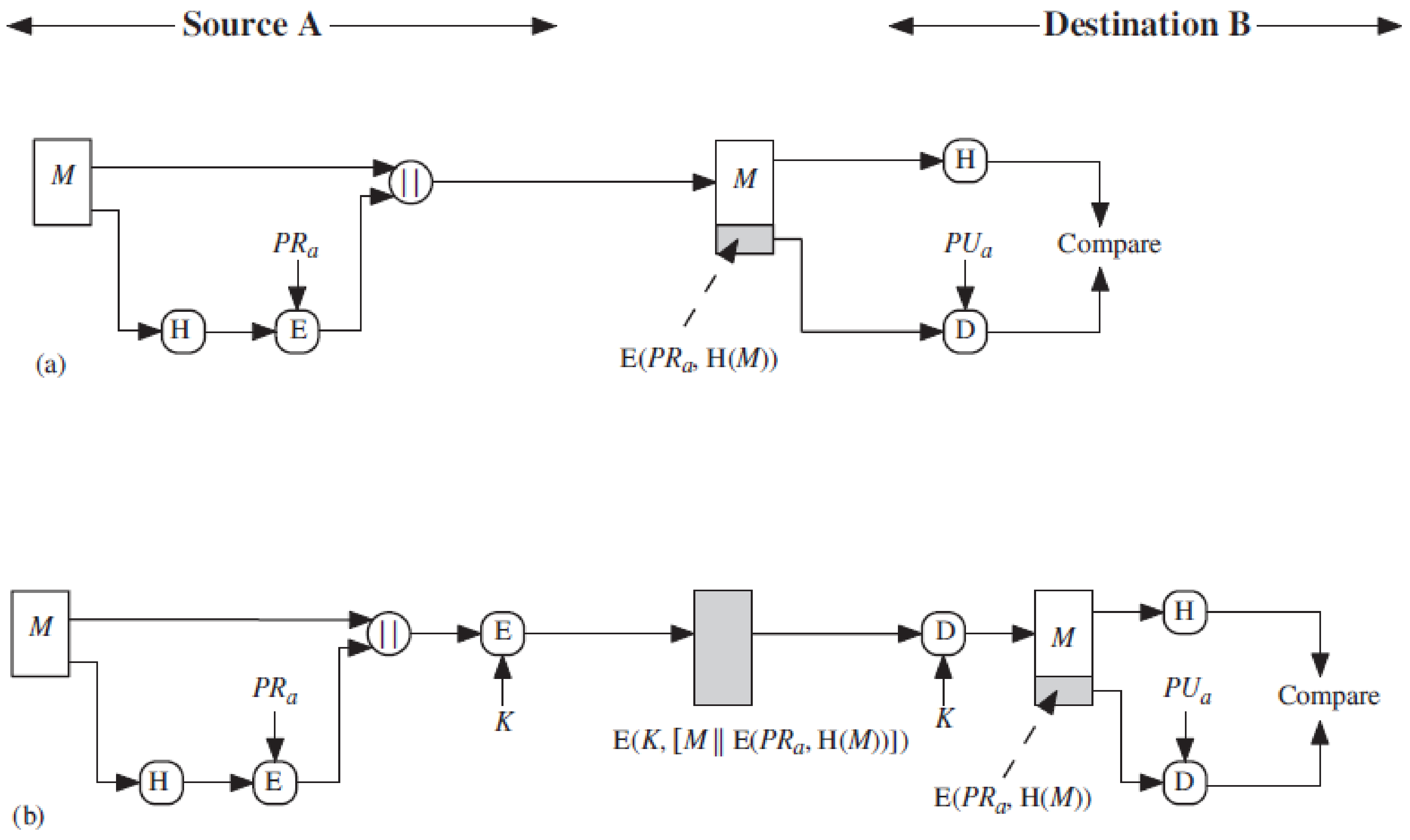


Message Authentication

- More commonly, message authentication is achieved using a **message authentication code (MAC)**, also known as a **keyed hash function.**
- A MAC function takes as input a secret key and a data block and produces a hash value
- If the **integrity** of the message needs to be checked, the MAC function can be applied to the message and the result compared with the stored MAC value
- In practice, specific MAC algorithms are designed that are generally more efficient than an encryption algorithm.

Digital Signatures

- The hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature

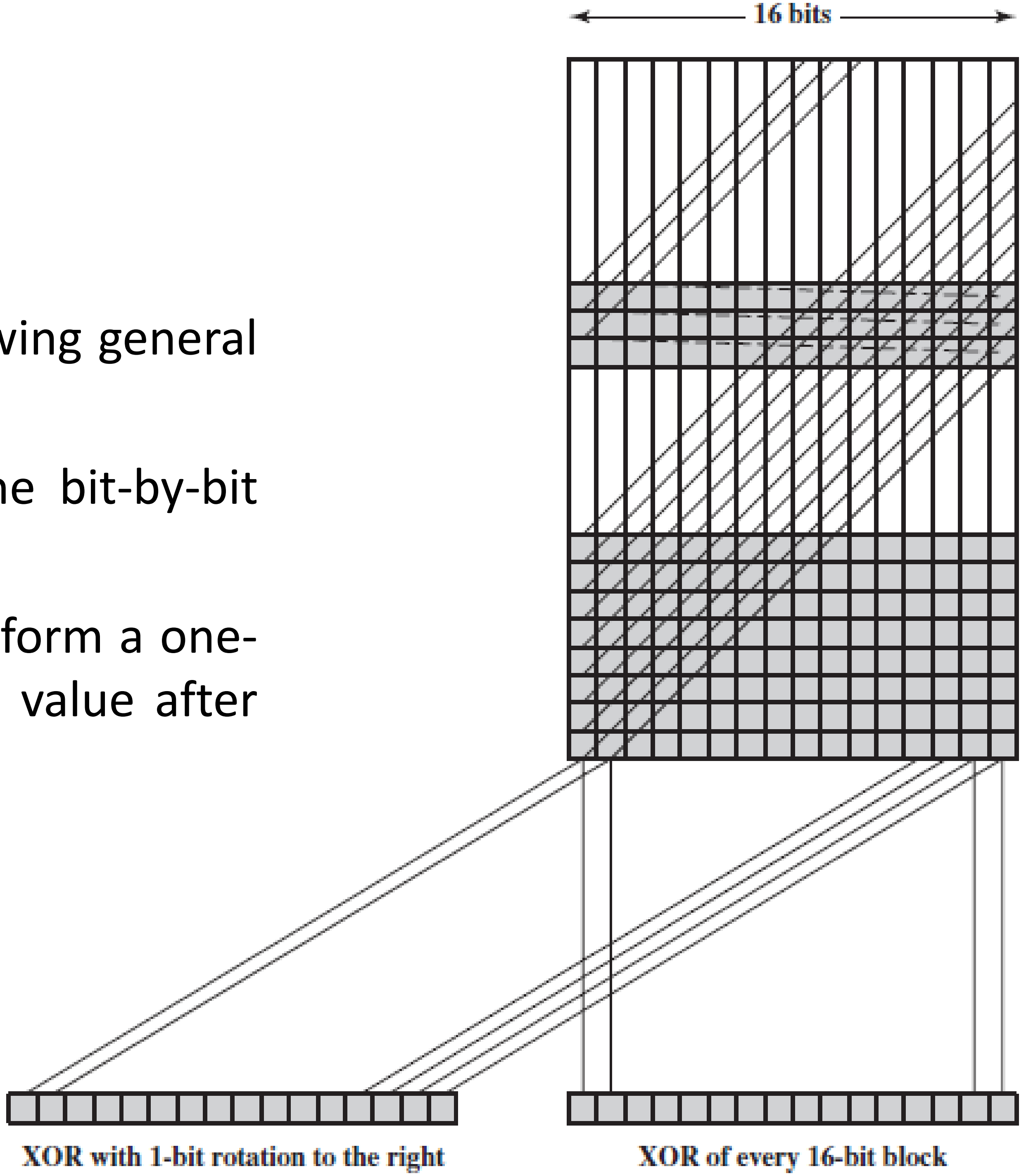


Other Applications

- One-way password file
- Intrusion detection
- Virus detection
- Pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

Two Simple Hash Functions

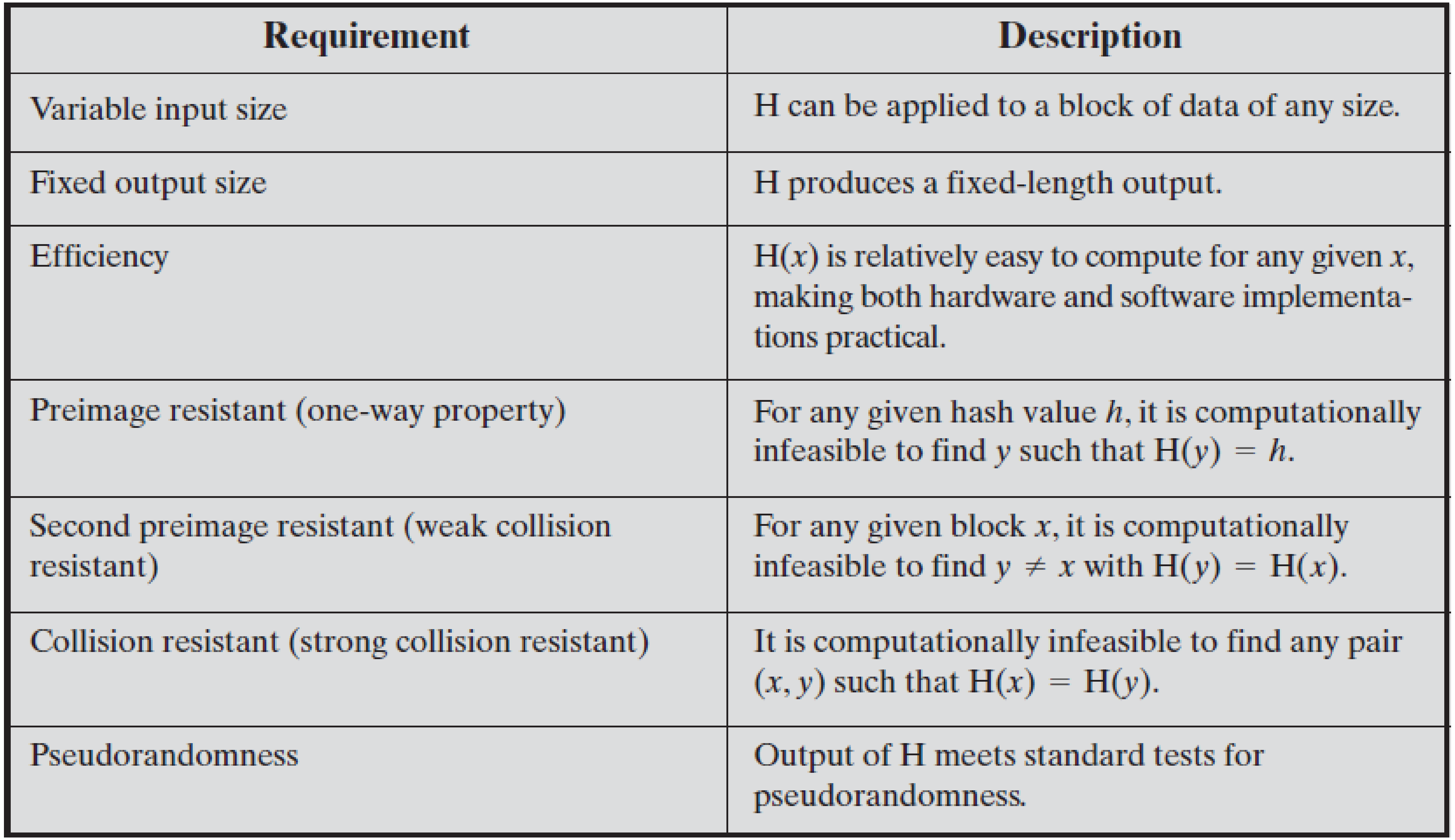
- All hash functions operate using the following general principles.
- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block.
- A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed


16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

Requirements And Security

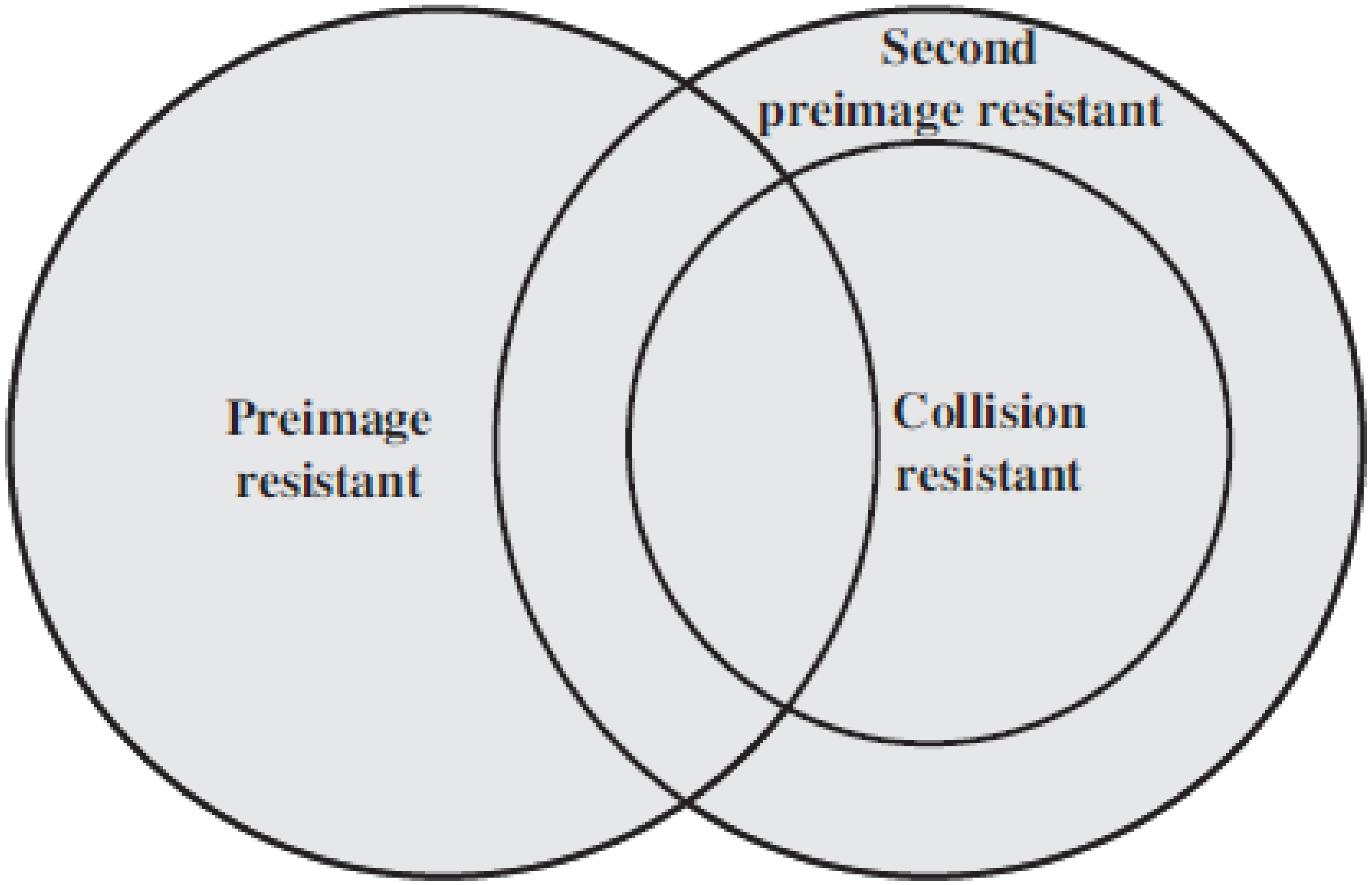
- For a hash value $h=H(x)$, we say that x is the **preimage** of h.
- Because H is a many-to-one mapping, for any given hash value h, there will in general be multiple preimages
- A **collision** occurs if we have $x \neq y$ and $H(x)=H(y)$.
- Because we are using hash functions for data integrity, collisions are clearly undesirable.

Security Requirements for Cryptographic Hash Functions



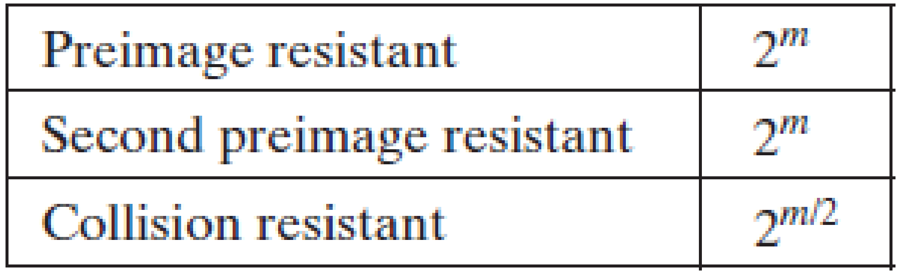
Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

Security Requirements for Cryptographic Hash Functions



Brute-Force Attacks

- A brute-force attack does not depend on the specific algorithm but depends only on bit length
- **Preimage or second preimage attack**
- **Collision resistant attack**
- The effort required is explained by a mathematical result referred to as the **birthday paradox**



Preimage resistant	2^m
Second preimage resistant	2^m
Collision resistant	$2^{m/2}$

Cryptanalysis

- A cryptanalysis, in contrast, is an attack based on weaknesses in a particular cryptographic algorithm
- An ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

- Hash Functions Based On Cipher Block Chaining
- Secure Hash Algorithm (SHA)

Hash Functions Based On Cipher Block Chaining

- A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without using the secret key
- Divide a message M into fixed-size blocks M1,M2,…….,Mn and use a symmetric encryption system such as DES to compute the hash code as

$$
\begin{aligned}
H_0 &= \text{initial value} \\
H_i &= \mathrm{E}(M_i, H_{i-1}) \\
G &= H_N
\end{aligned}
$$

- This is similar to the CBC technique, but in this case, there is no secret key.
- This form of hash can easily be cracked using a **meet-in-the-middle-attack**

Secure Hash Algorithm (SHA)

- The most widely used hash function has been the Secure Hash Algorithm (SHA).
- **SHA-0** was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- A 160 bits revised version was issued as FIPS 180-1 in 1995 and is referred to as **SHA-1**
- A revised version was issued as FIPS 180-2 in 2002 and is referred to as **SHA-2, known as SHA-256, SHA-384, and SHA-512.**
- In 2008 another subversion **SHA-224**
- The actual standards document is entitled **“Secure Hash Standard.”**

Secure Hash Algorithm (SHA)

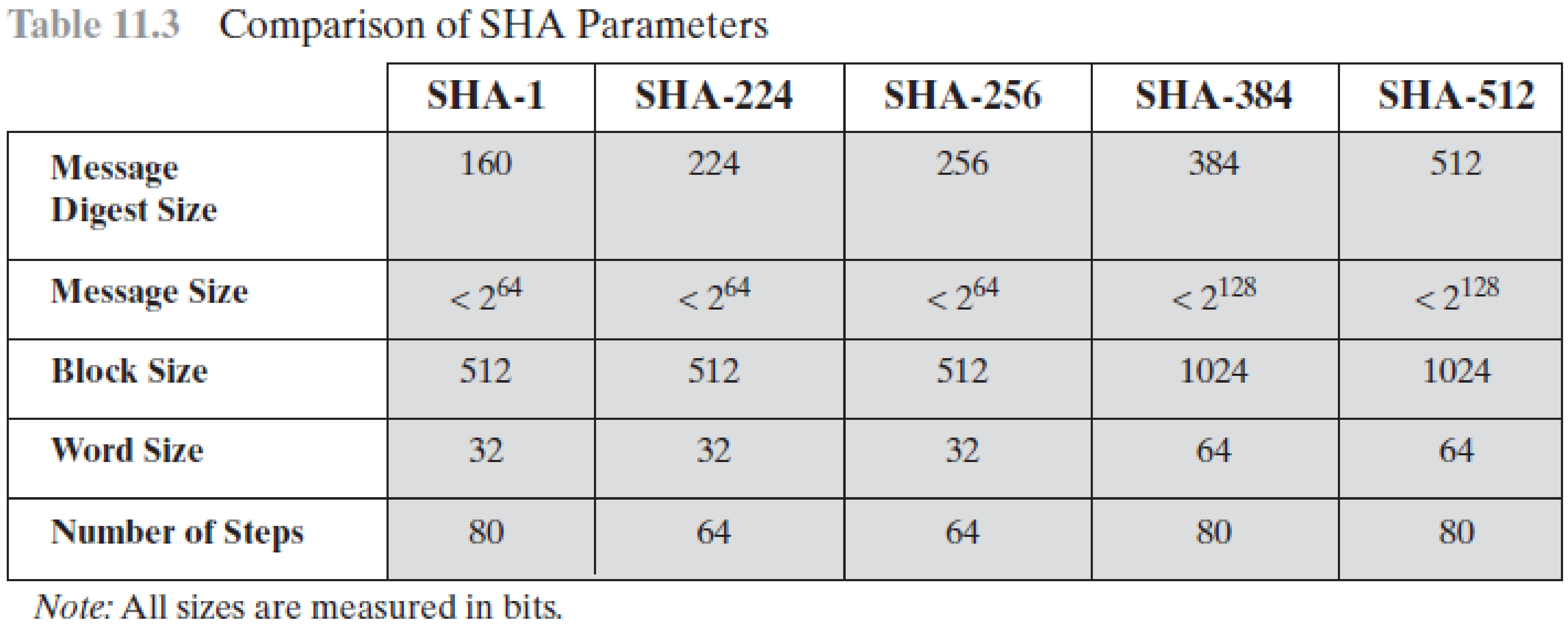


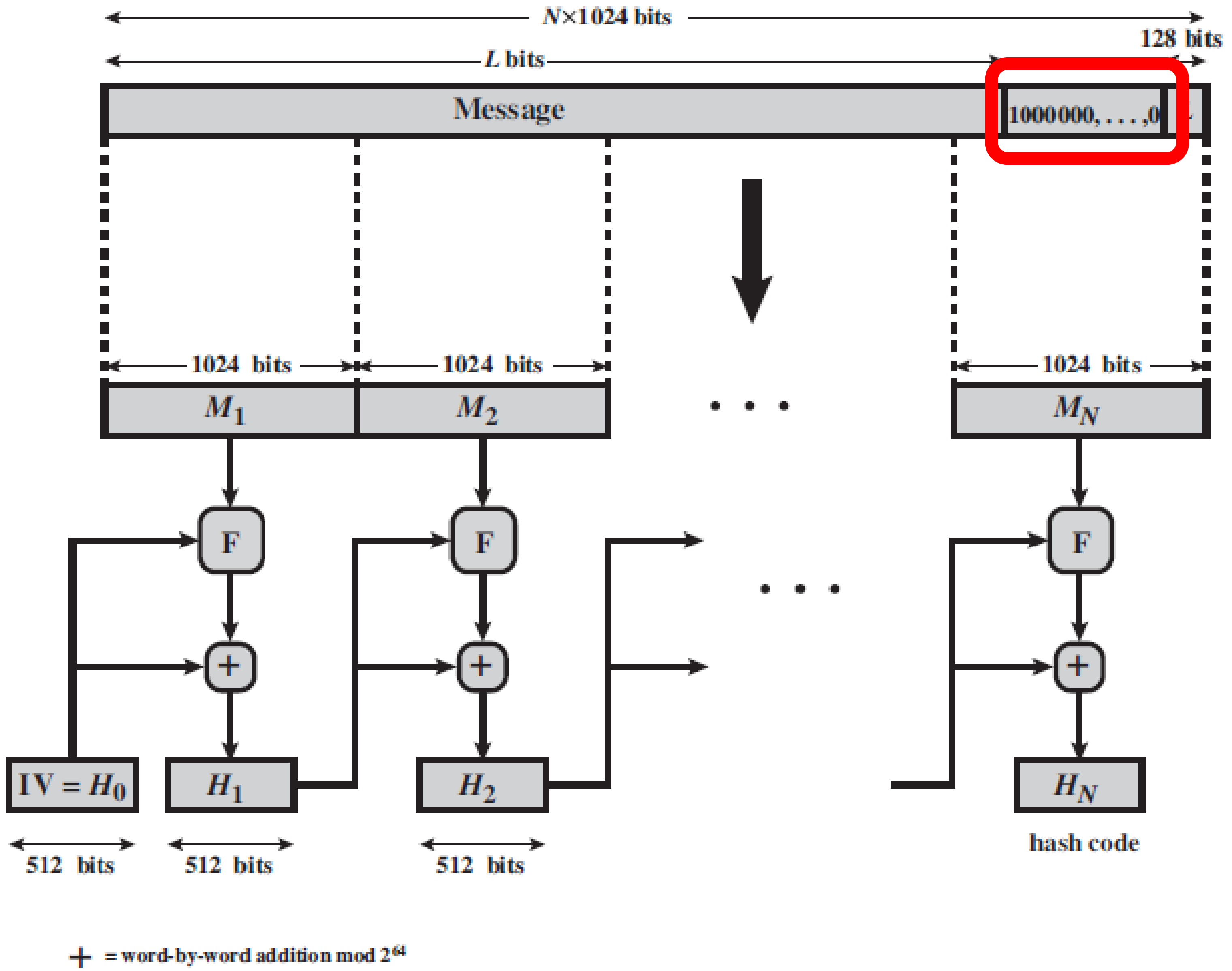
Table 11.3 Comparison of SHA Parameters

	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

Note: All sizes are measured in bits.

SHA-512 Logic

- **Step 1 Append padding bits**. The message is padded so that its length is congruent to 896 modulo 1024 .
- Padding is always added, even if the message is already of the desired length.
- The number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

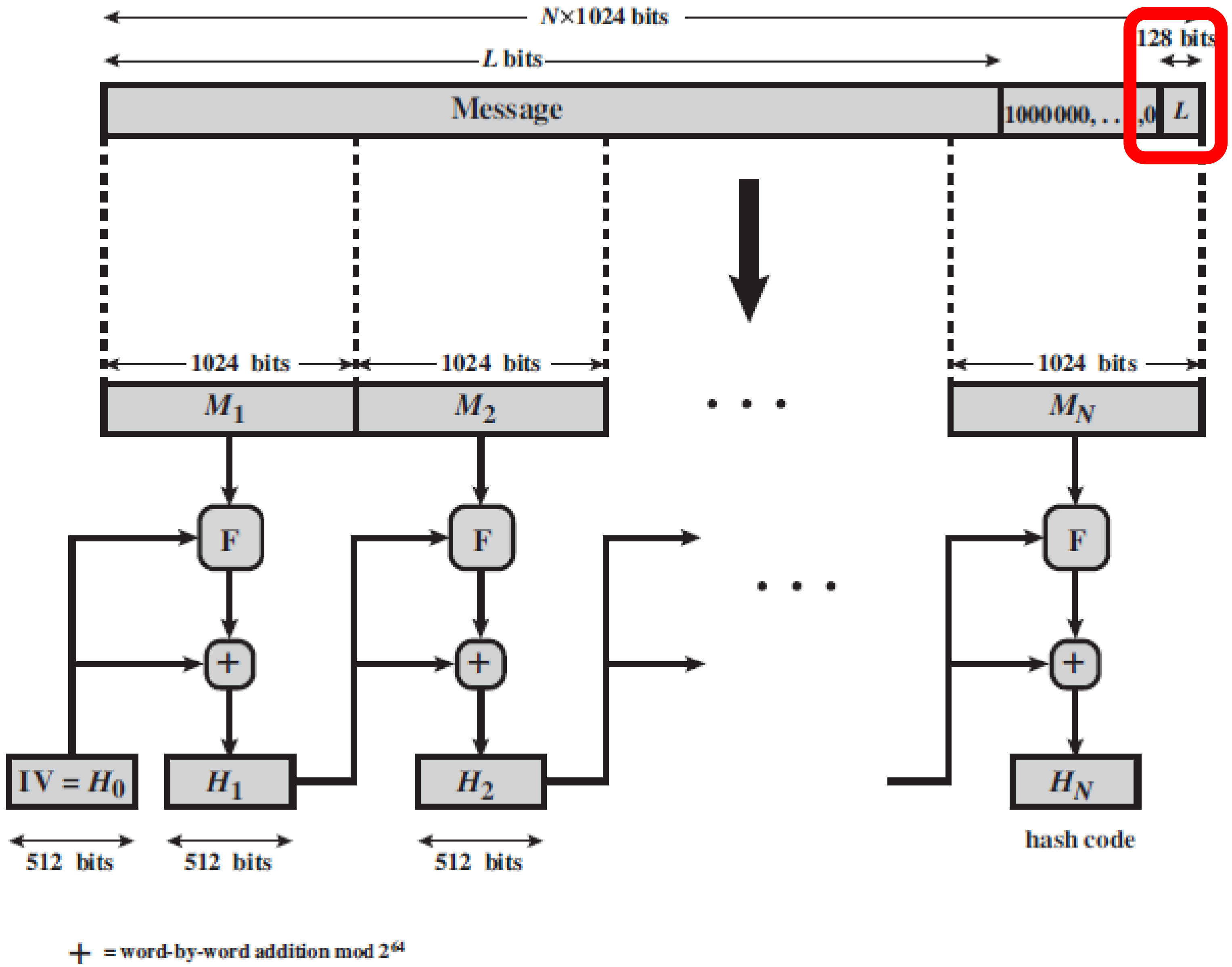


SHA-512 Logic

- **Step 2 Append length**.

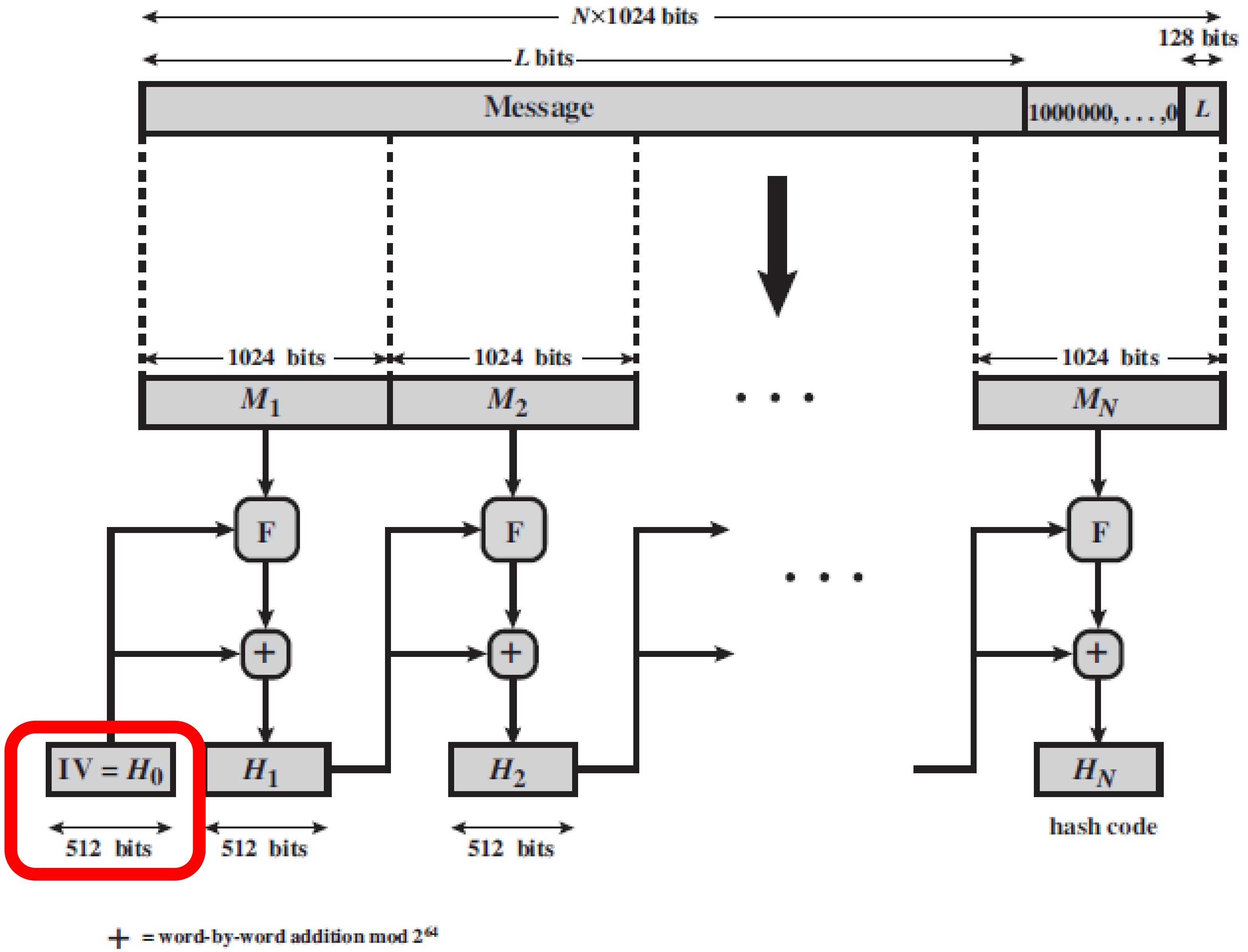
A block of 128 bits is appended to the message.

- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the **length of the original message**



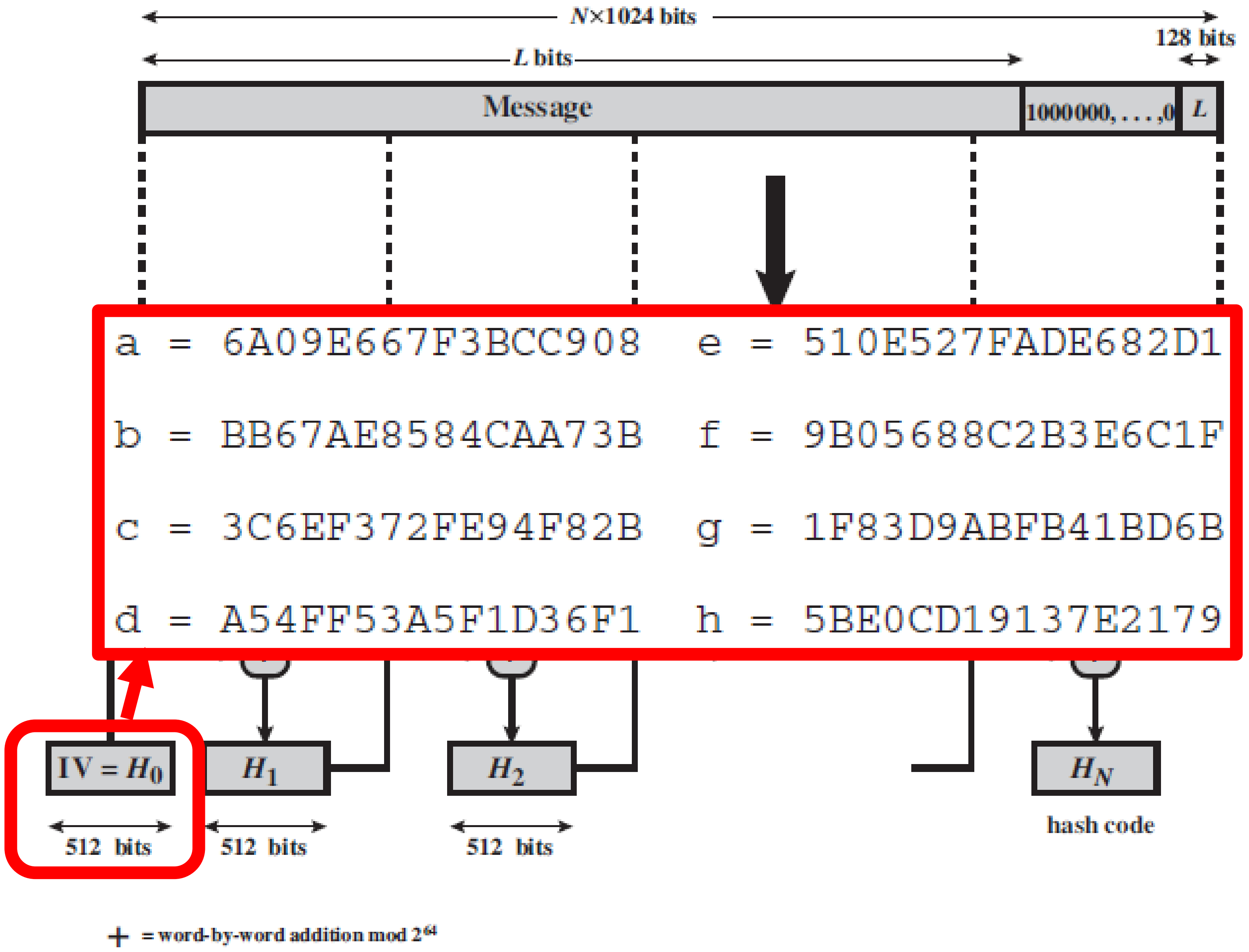
SHA-512 Logic

- **Step 3 Initialize hash buffer**. A 512-bit buffer is used to hold intermediate and final results of the hash function.
- The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).



SHA-512 Logic

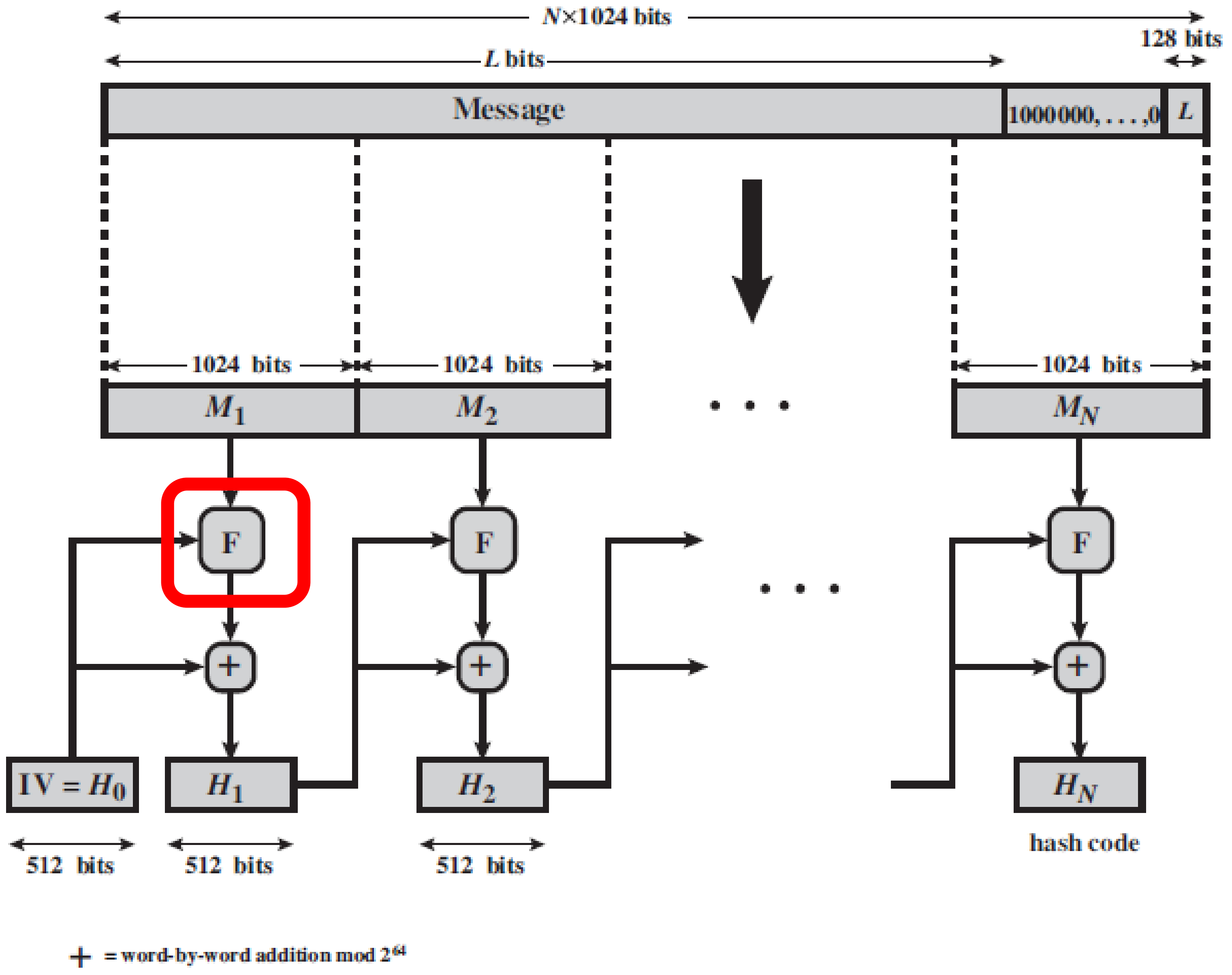
- These eight registers are initialized to 64-bit
- These values are stored in **big-endian** format, which is the most significant byte of a word in the low-address (leftmost) byte position.
- These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.



```
a = 6A09E667F3BCC908   e = 510E527FADE682D1

b = BB67AE8584CAA73B   f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B   g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1   h = 5BE0CD19137E2179
```

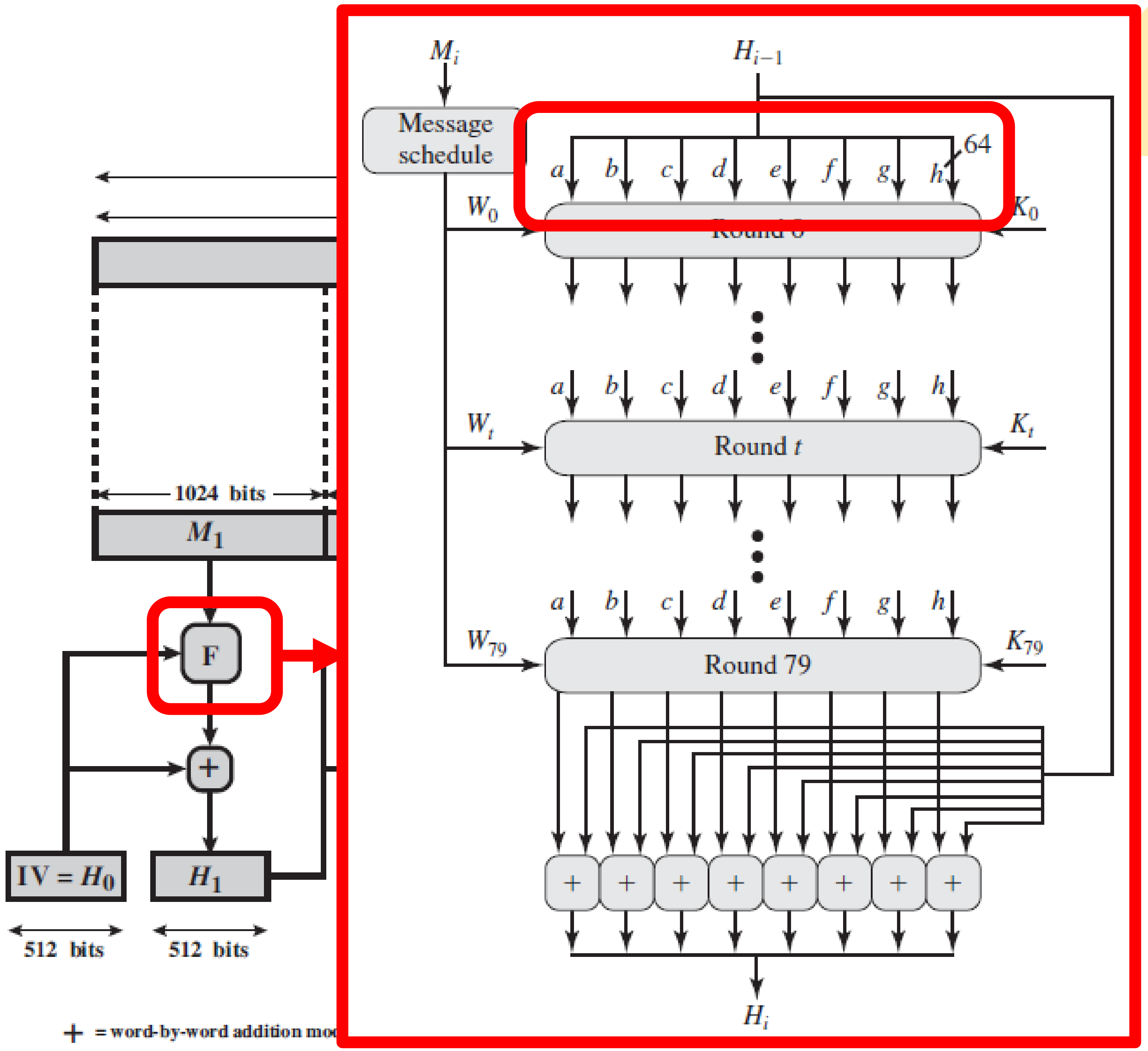
SHA-512 Logic

- **Step 4 Process message in 1024-bit (128-word) blocks.** The heart of the algorithm is a module that consists of 80 rounds



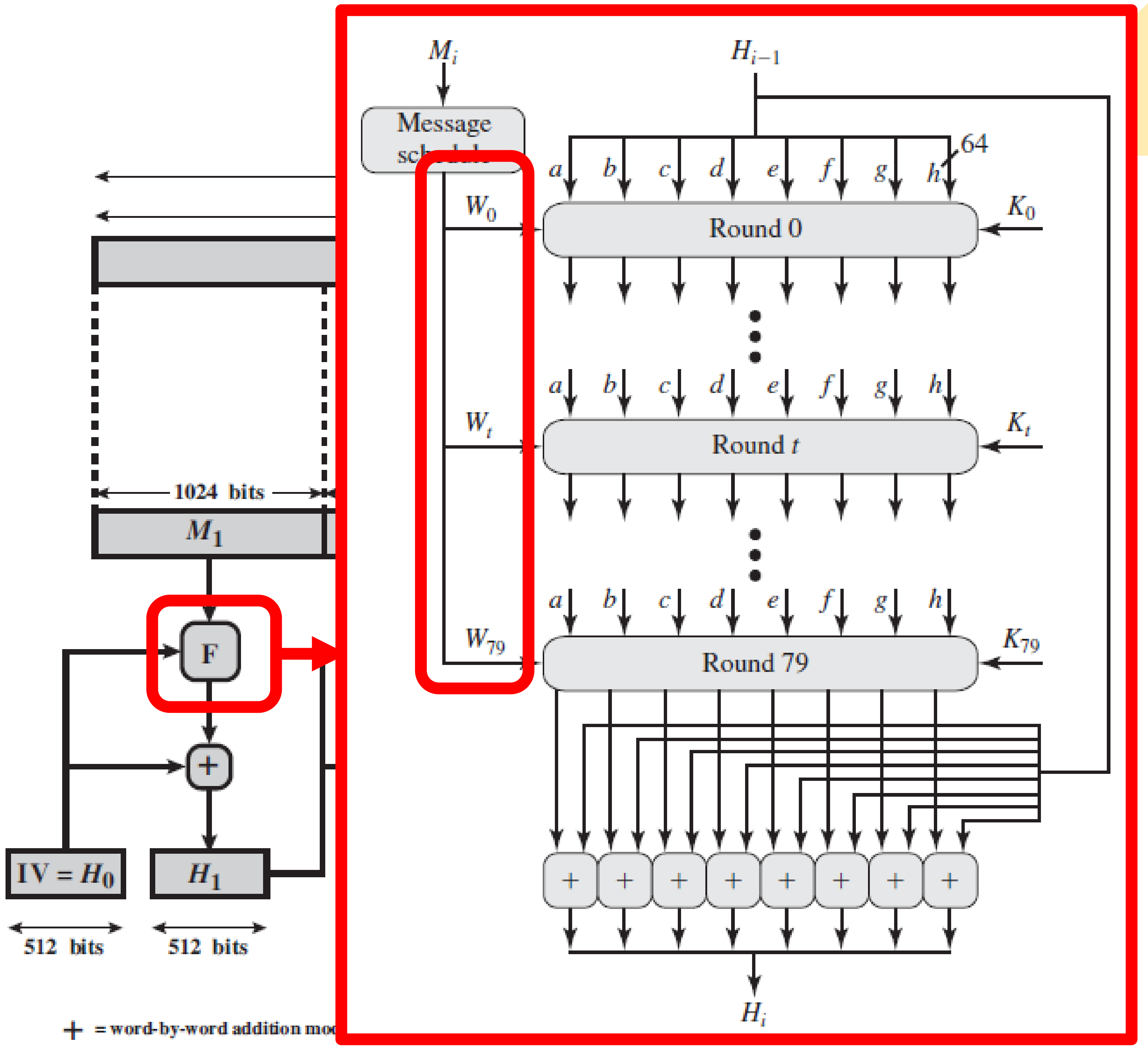
SHA-512 Logic

- Each round takes as input the 512-bit buffer value, **abcdefgh**, and updates the contents of the buffer



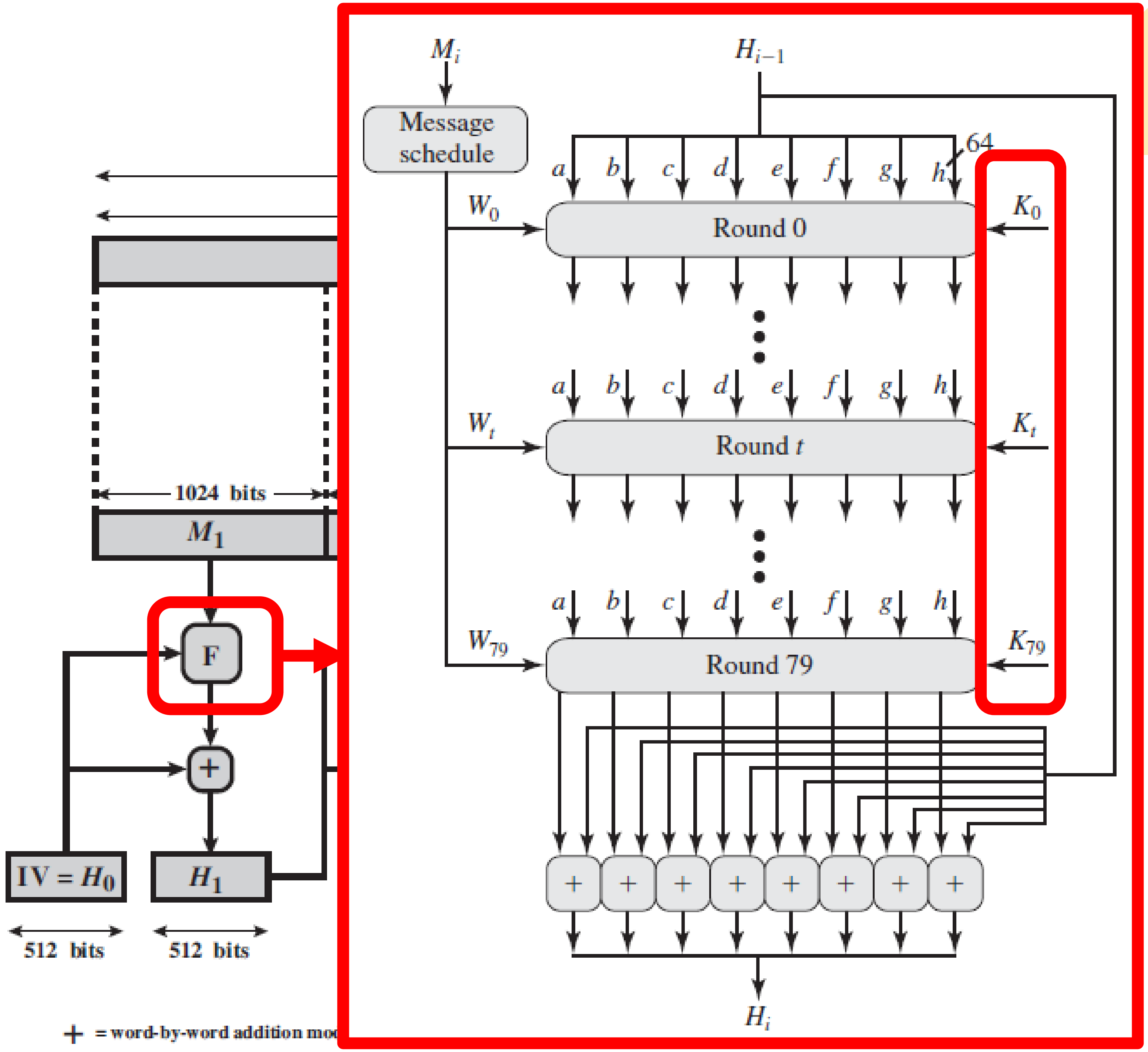
SHA-512 Logic

- At input to the first round, the buffer has the value of the intermediate hash value
- Each round makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed Mi



SHA-512 Logic

- Each round also makes use of an additive constant Kt
- These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers
- The constants provide a “randomized” set of 64-bit patterns



SHA-512 Logic

$$T_1 = h + \text{Ch}(e, f, g) + \left(\textstyle\sum_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\textstyle\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$

$$h = g$$

$$g = f$$

$$f = e$$

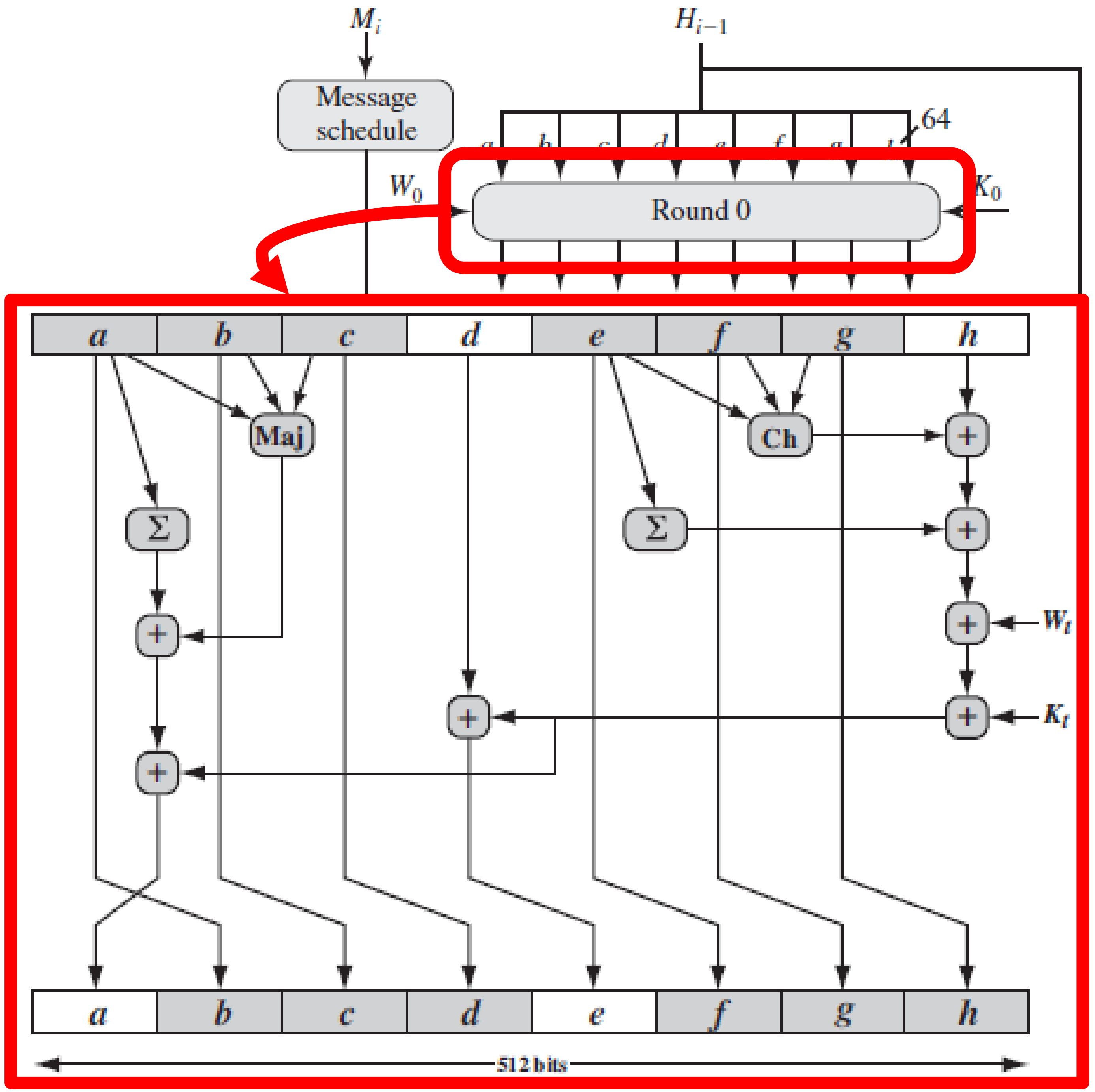
$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$



SHA-512 Logic

t	$=$ step number; $0 \leq t \leq 79$
$\text{Ch}(e, f, g)$	$= (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$ *the conditional function: If e then f else g*
$\text{Maj}(a, b, c)$	$= (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$ *the function is true only of the majority (two or three) of the arguments are true*
$\left(\sum_0^{512} a\right)$	$= \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$
$\left(\sum_1^{512} e\right)$	$= \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$
$\text{ROTR}^n(x)$	$=$ circular right shift (rotation) of the 64-bit argument x by n bits
W_t	$=$ a 64-bit word derived from the current 512-bit input block
K_t	$=$ a 64-bit additive constant
$+$	$=$ addition modulo 2^{64}

SHA-512 Logic

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

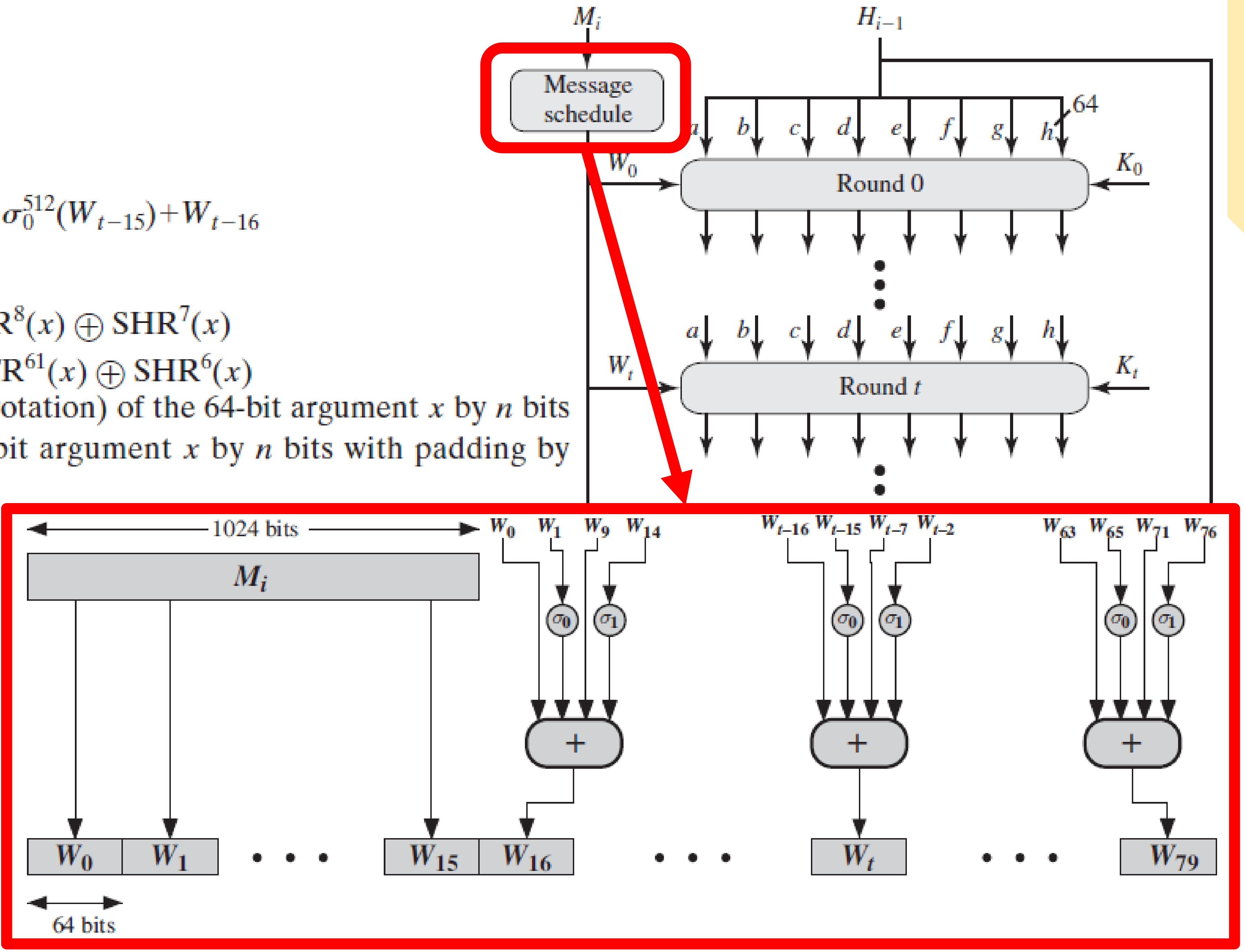
$$\sigma_0^{512}(x) = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x)$$

$$\sigma_1^{512}(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)$$

$\mathrm{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument x by n bits

$\mathrm{SHR}^n(x)$ = left shift of the 64-bit argument x by n bits with padding by zeros on the right

$+$ = addition modulo 2^{64}

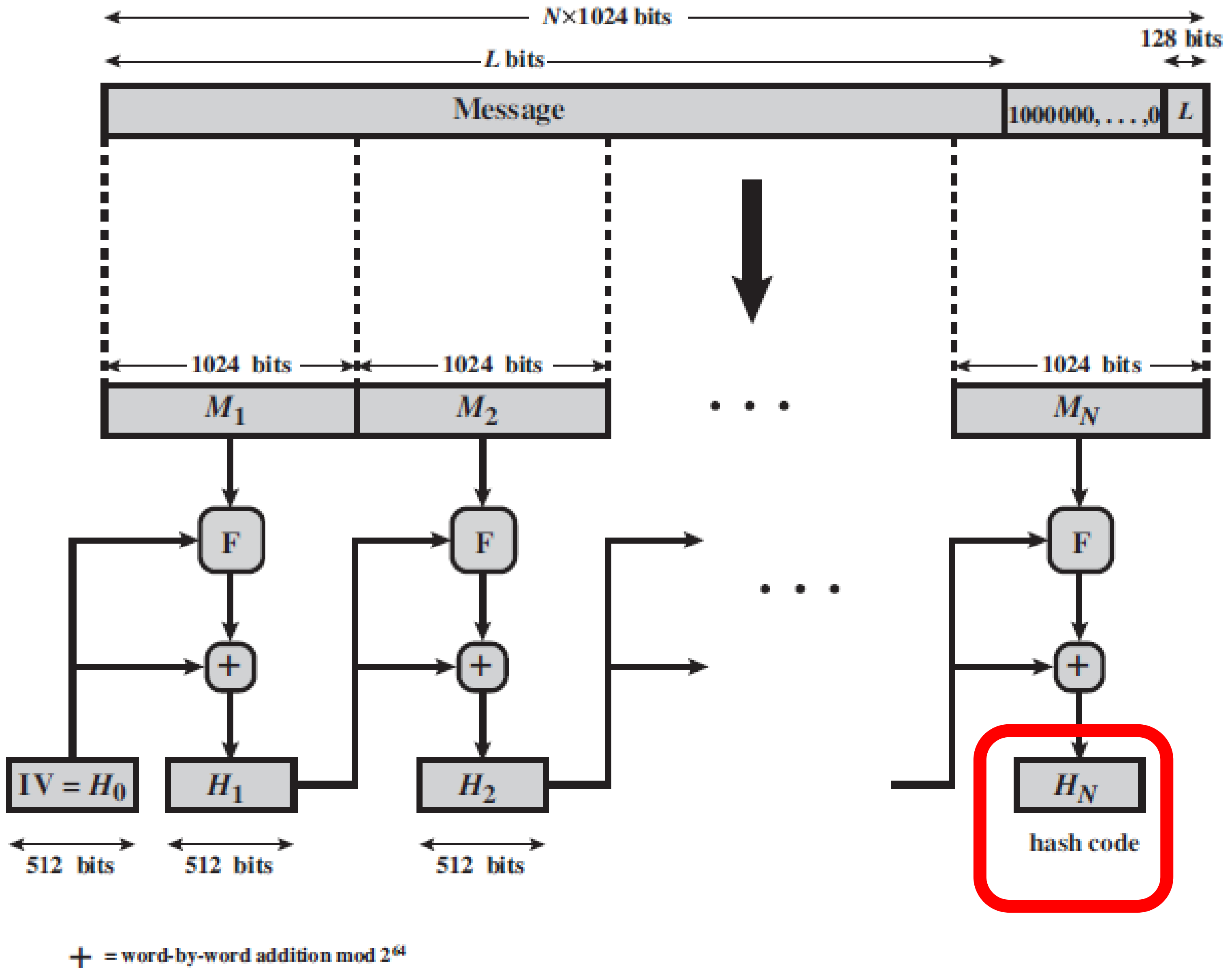


M_i
H_{i-1}
Message schedule
64
a b c d e f g h
W_0
Round 0
K_0
W_t
Round t
K_t
1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0 σ_1
+
W_0 W_1 W_{15} W_{16} W_t W_{79}
64 bits

SHA-512 Logic

- **Step 5 Output**.

After all N 1024-bit blocks have been processed, the output from the th stage is the 512-bit message digest



SHA-3

NIST announced in 2007 a competition to produce the next generation NIST hash function, to be called SHA-3.

The basic requirements that must be satisfied by any candidate for SHA-3 are the following.

1. It must be possible to replace SHA-2 with SHA-3 in any application by a simple drop-in substitution. Therefore, SHA-3 must support hash value lengths of 224, 256, 384, and 512 bits.

2. SHA-3 must preserve the online nature of SHA-2.That is, the algorithm must process comparatively small blocks (512 or 1024 bits) at a time instead of requiring that the entire message be buffered in memory before processing it.

SHA-3

The evaluation criteria for the new hash function, in decreasing order of importance, are as follows:

- **Security**: The security strength of SHA-3 should be close to the theoretical maximum for the different required hash sizes and for both preimage resistance and collision resistance. SHA-3 algorithms must be designed to resist any potentially successful attack on SHA-2 functions. In practice, this probably means that SHA-3 must be fundamentally different than the SHA-1, SHA-2, and MD5 algorithms in either structure, mathematical functions, or both.
- **Cost**: SHA-3 should be both time and memory efficient over a range of hardware platforms.
- **Algorithm and implementation characteristics**: Consideration will be given to such characteristics as flexibility (e.g., tunable parameters for security/ performance tradeoffs, opportunity for parallelization, and so on) and simplicity. The latter characteristic makes it easier to analyze the security properties of the algorithm

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

- Principles of Pseudorandom Number Generation
- Pseudorandom Number Generators
- PRNG Using a Block Cipher
- PRNG Using a Stream Cipher

Principles Of Pseudorandom Number Generation

- Random numbers play an important role in the use of encryption for various network security applications
- These applications give rise to two distinct
  - **RANDOMNESS**
    - **Uniform distribution**
    - **Independence**
  - **UNPREDICTABILITY**

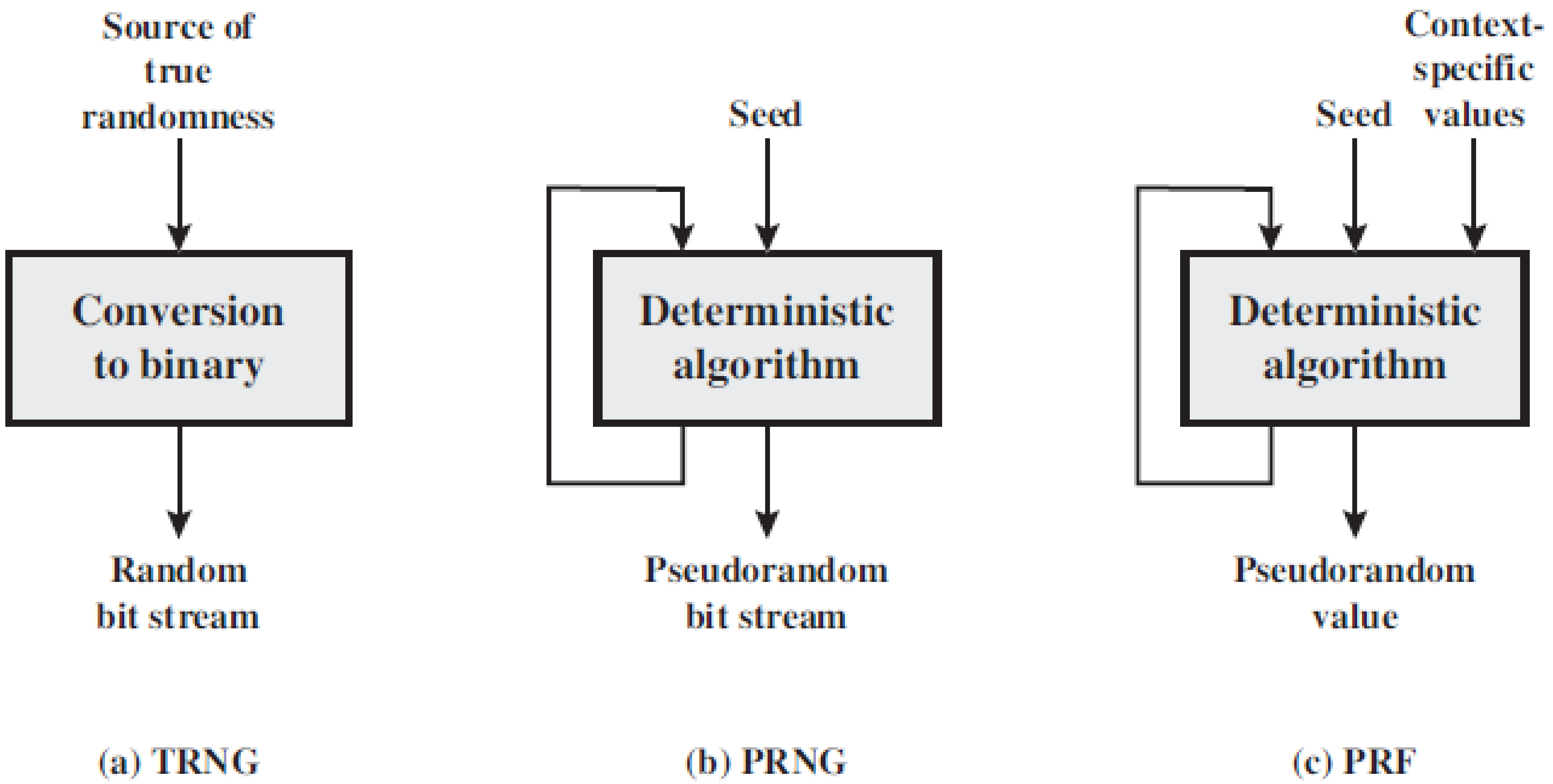
Principles Of Pseudorandom Number Generation

- Cryptographic applications typically make use of algorithmic techniques for random number generation
- These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random
- if the algorithm is good, the resulting sequences will pass many reasonable tests of randomness

TRNGs, PRNGs, and PRFs

- True Random Number Generator:
    - Non-deterministic source, physical environnent
    - Detect ionizing radiation events, leaky capacitors, Thermal noise from resistors or audio inputs
    - Mouse/keyboard activity, I/O operations, interrupts
    - Inconvenient, small number of values
- Pseudo Random Number Generator
    - Deterministic algorithms to calculate numbers in “relatively random” sequence
    - Seed is algorithm input
    - Produces continuous stream of random bits
- Pseudo Random Function
    - Same as PRNG but produces string of bits of some fixed length

TRNGs, PRNGs, and PRFs



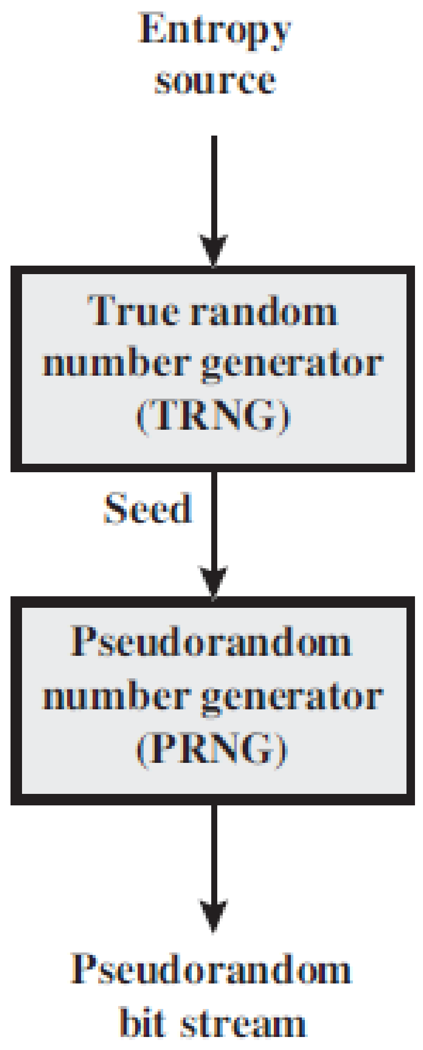
TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

PRNG Requirements

- When a PRNG or PRF is used for a cryptographic application, then the basic requirement is that an adversary who does not know the seed is unable to determine the pseudorandom string
- The general requirement for secrecy of the output of a PRNG or PRF leads to specific requirements :
  - RANDOMNESS
  - UNPREDICTABILITY
  - SEED REQUIREMENTS

PRNG Requirements

- RANDOMNESS:
  - Uniformity
  - Scalability
  - Consistency
- UNPREDICTABILITY:
  - Forward unpredictability
  - Backward unpredictability
- SEED:
  - Seed must be secure
  - Use TRNG to generate seed


Entropy source
True random number generator (TRNG)
Seed
Pseudorandom number generator (PRNG)
Pseudorandom bit stream

PRNG Tests

- **Frequency test:** This is the most basic test and must be included in any test suite. The purpose of this test is to determine whether the number of ones and zeros in a sequence is approximately the same as would be expected for a truly random sequence
- **Runs test:** The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence.
- **Maurer's universal statistical test**: The focus of this test is the number of bits between matching patterns (a measure that is related to the length of a compressed sequence). The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random

Algorithm Design

- **Purpose-built algorithms**
- **Algorithms based on existing cryptographic algorithms**

Three broad categories of cryptographic algorithms are commonly used to create PRNGs:

  - Symmetric block ciphers
  - Asymmetric ciphers
  - Hash functions and message authentication codes

Linear Congruential Generators

- **A widely used technique for pseudorandom number generation**

m	the modulus	m > 0
a	the multiplier	0 < a < m
c	the increment	0 … c < m
X0	the starting value, or seed	0 … X0 < m

$$X_{n+1} = (aX_n + c) mod\ (m)$$

- The selection of values for a ,c and m is critical in developing a good random number generator
- a= 16807
- c=0
- m = 2^{31} - 1

Linear Congruential Generators

- **tests to be used in evaluating a random number generator:**
- T1: The function should be a full-period generating function. That is, the function should generate all the numbers between 0 and m before repeating.
- T2: The generated sequence should appear random.
- T3: The function should implement efficiently with 32-bit arithmetic.

- Suppose that the opponent can determine values for X0,X1,X2 and X3

Blum Blum Shub (BBM) Generator

- It has perhaps the strongest public proof of its cryptographic strength of any purpose-built algorithm
- choose two large prime numbers p and q

$$p \equiv q \equiv 3 \; mod \; 4$$

- n=p x q
- choose a random number s , such that is relatively prime to n ; this is equivalent to saying that neither p nor q is a factor of s

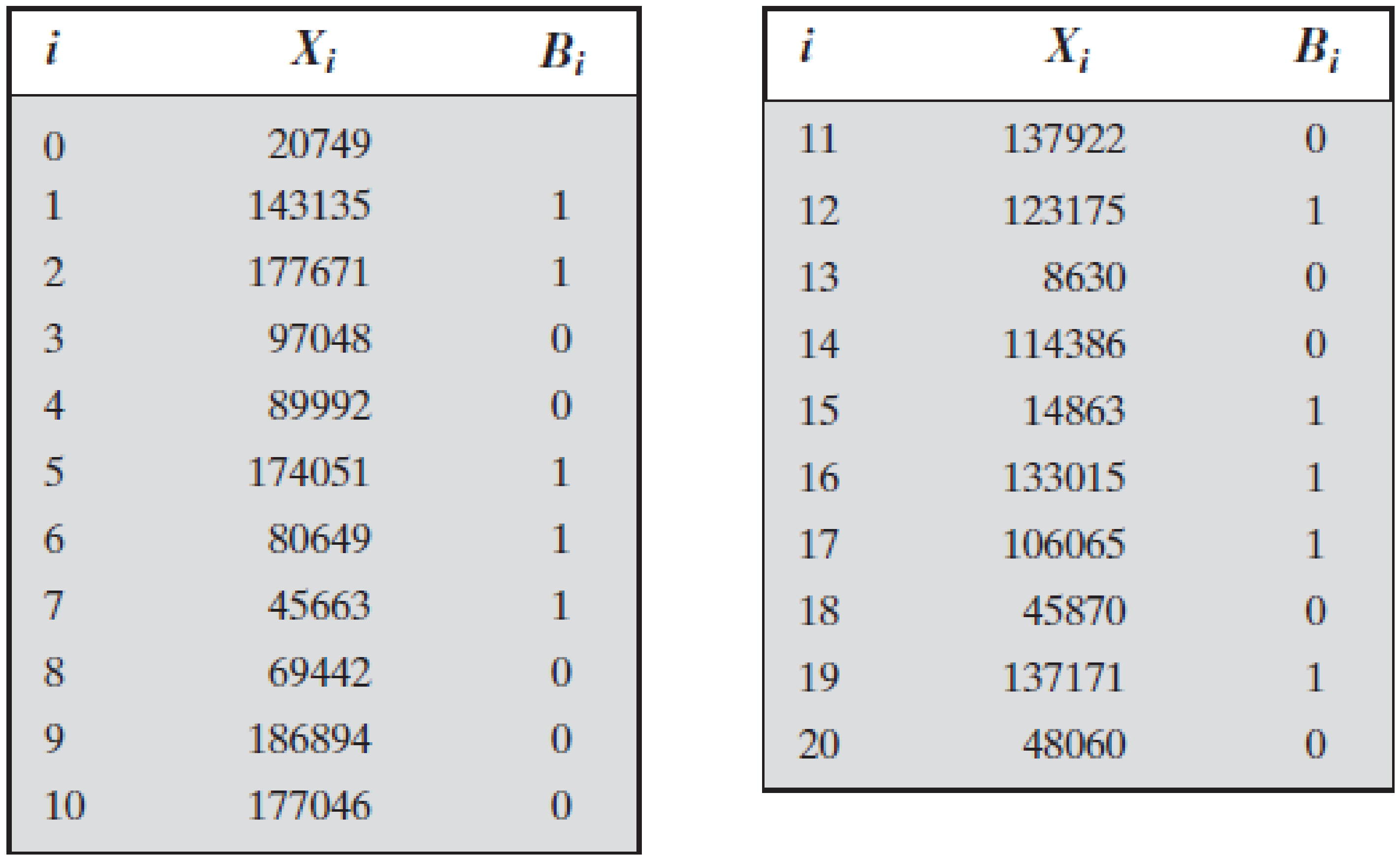
```
$X_0 = s^2 \bmod n$
for $i = 1$ to $\infty$
    $X_i = (X_{i-1})^2 \bmod n$
    $B_i = X_i \bmod 2$
```

- The BBS is referred to as a cryptographically secure pseudorandom bit generator (CSPRBG).
- The security of BBS is based on the difficulty of factoring n. That is, given n, we need to determine its two prime factors p and q.

Blum Blum Shub Generator

- n = 192649 = 383 * 503
- s = 101355

Example Operation of BBS Generator



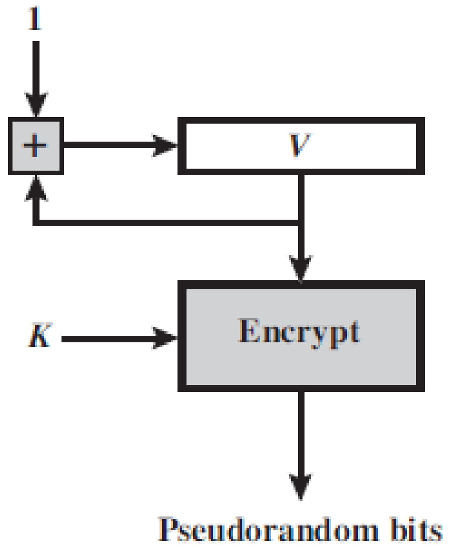
i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

PRNG Mechanisms Using a Block Ciphers

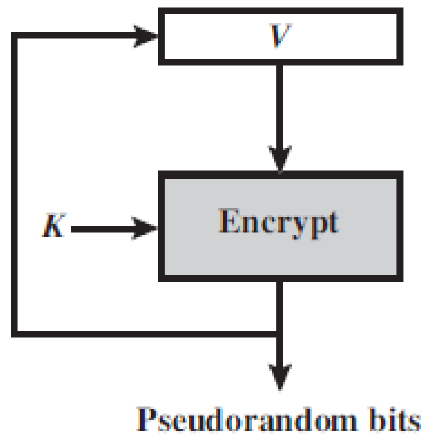
- A popular approach to PRNG construction is to use a symmetric block cipher as the heart of the PRNG mechanism
- If an established, standardized block cipher is used, such as DES or AES, then the security characteristics of the PRNG can be established.
- Further, many applications already make use of DES or AES, so the inclusion of the block cipher as part of the PRNG algorithm is straightforward.

PRNG Using Block Cipher Modes of Operation

- Use symmetric block ciphers (ex. AES, DES) to produce pseudo random bits
- The seed consists of two parts: the encryption key value K and a value V
- Pseudorandom bits are produced one block at a time



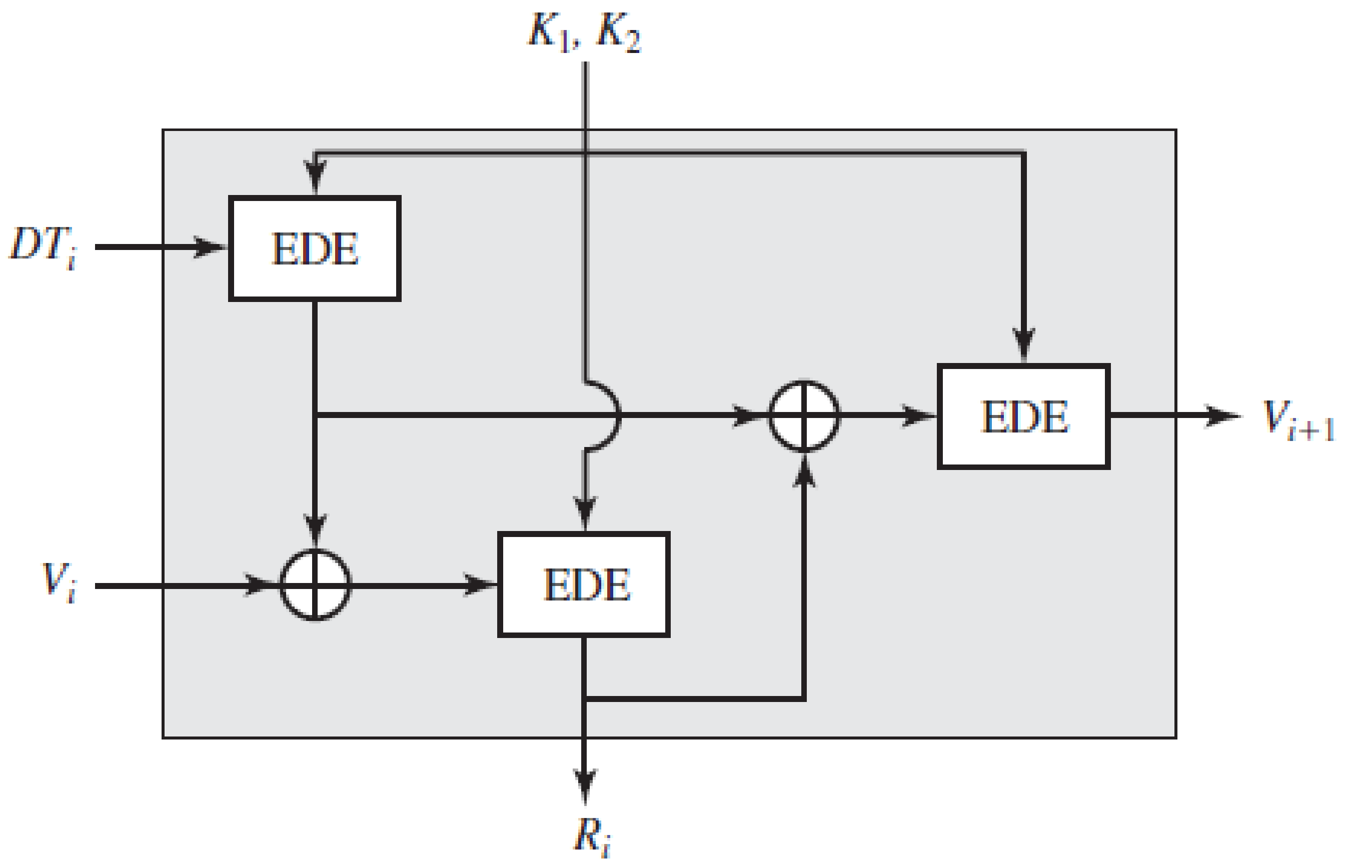
(a) CTR mode



(b) OFB mode

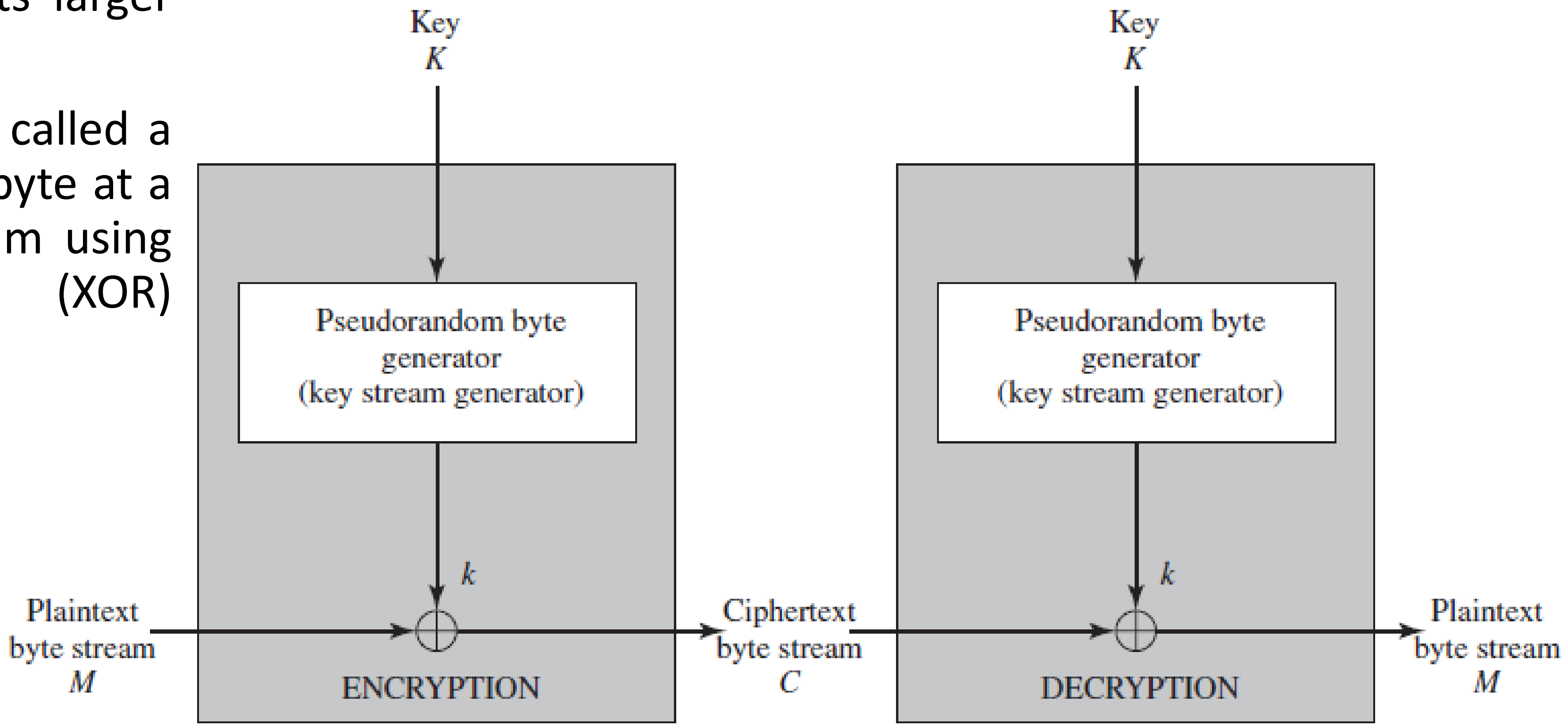
ANSI X9.17 PRNG

- One of the strongest (cryptographically speaking) PRNGs
- A number of applications employ this technique, including financial security applications and PGP
- The technique involves a 112-bit key and **three** EDE encryptions for a total of **nine** DES encryptions.
- The scheme is driven by two pseudorandom inputs, the date and time value, and a seed



Stream Ciphers

- Stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time
- The output of the generator, called a **keystream**, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation


Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

Stream Ciphers

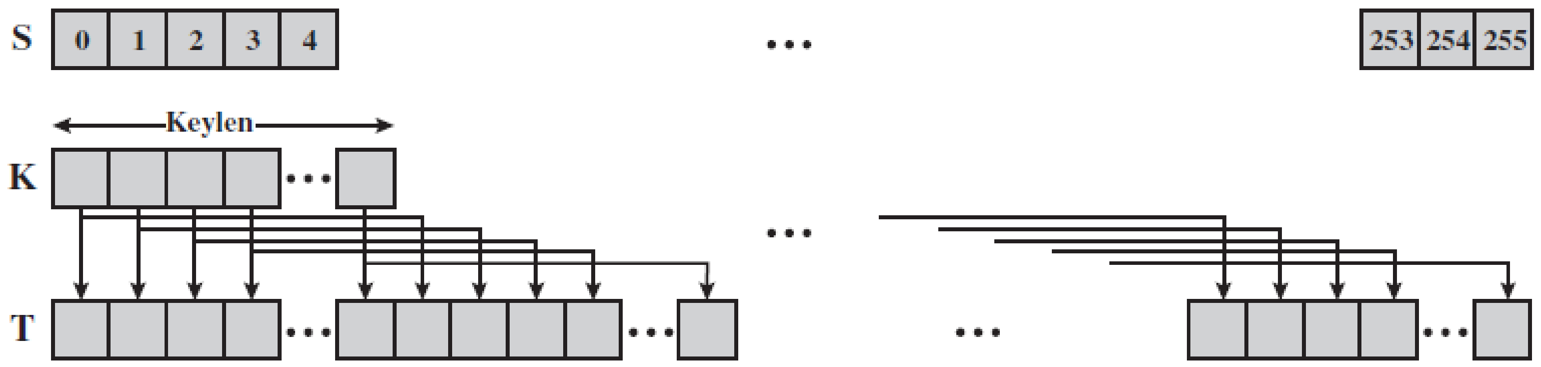
- Design considerations for a stream cipher:
  - The encryption sequence should have a large period
  - The keystream should approximate the properties of a true random number stream as close as possible
  - The key needs to be sufficiently long
- stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than do block ciphers

Stream Ciphers / RC4

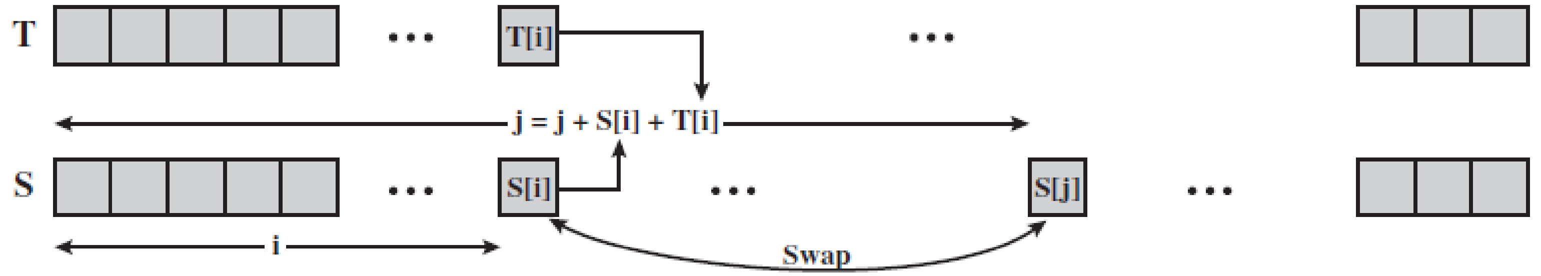
- RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security
- It is a variable key size stream cipher with byte-oriented operations
- Very long period of the cipher
- Used in:
  - The Secure Sockets Layer/Transport Layer Security (SSL/TLS)
  - The Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA)
- No known attacks if use 128 bit key and discard initial values of stream

RC4

```
/*Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

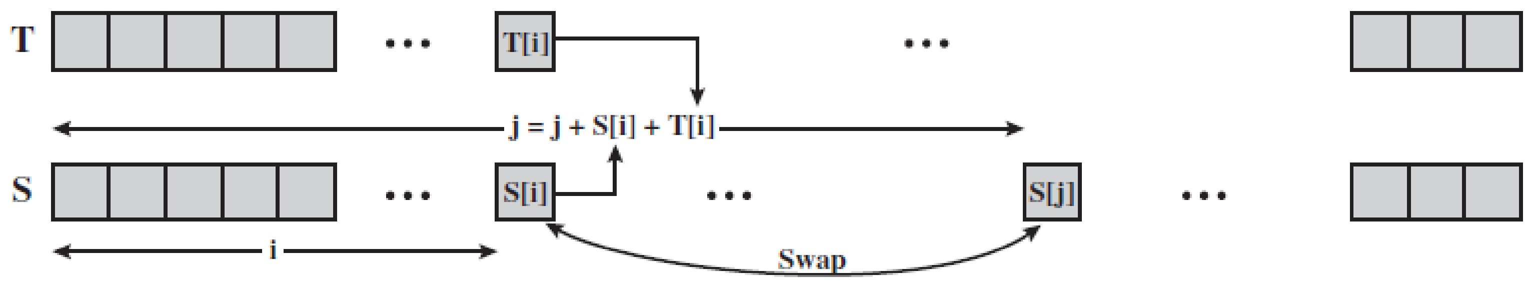


```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
j=(j + S[i] + T[i])mod 256;
Swap (S[i], S[j]);
```

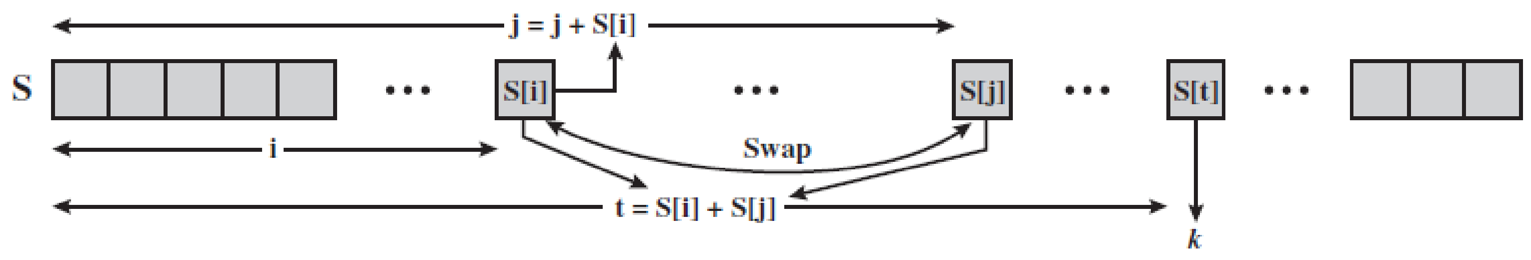


RC4

```
/* Stream Generation */
i = 0 , j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```



(b) Initial permutation of S



RC4

Steps:

1. Initialise S to values 0 to 255; initialise T with repeating values of key, K
2. Use T to create initial permutation of S
3. Permutate S and generate keystream, k from S
4. Encrypt a byte of plaintext, p by XOR with k

Stream Vs. Block Ciphers

- For applications that require encryption/decryption of a stream of data, such as over a data communications channel or a browser/Web link, a stream cipher might be the better alternative.
- For applications that deal with blocks of data, such as file transfer, e-mail, and database, block ciphers may be more appropriate.
- However, either type of cipher can be used in virtually any application.

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Spread Spectrum

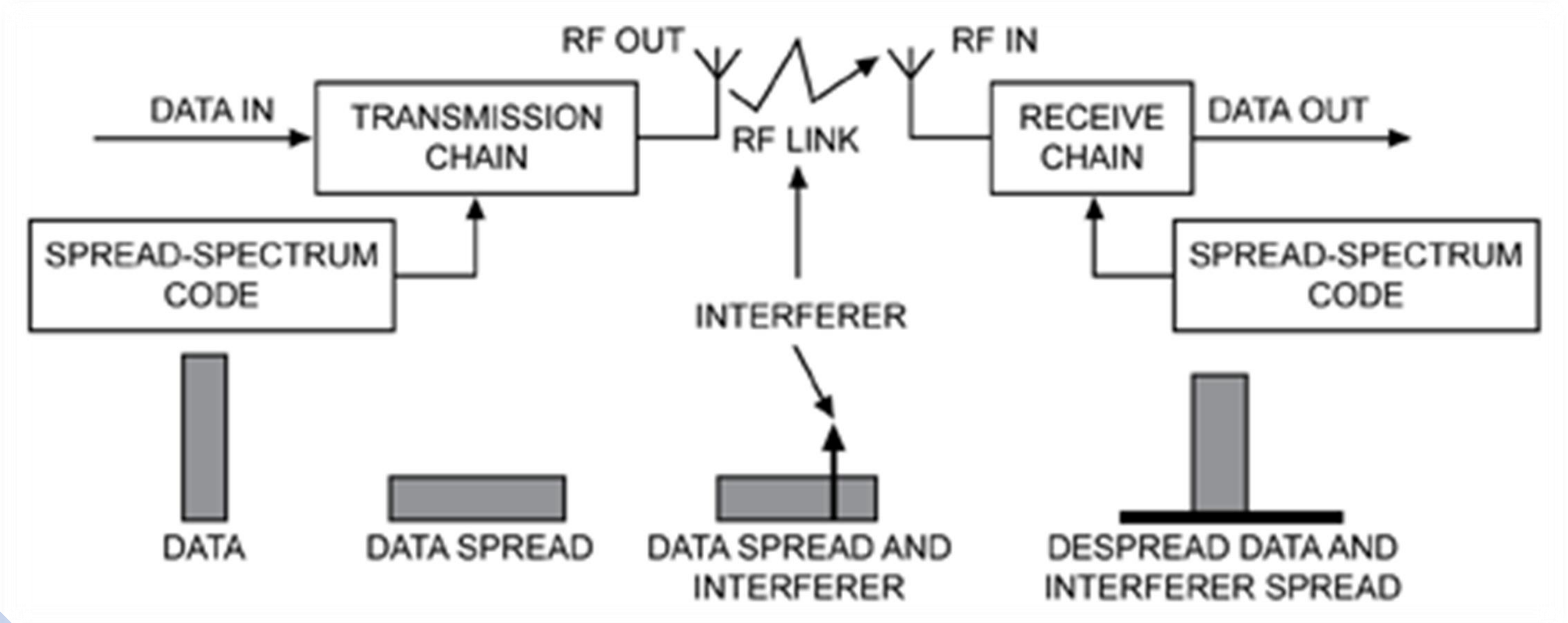
Introduction

- A transmission technique in which a **pseudo-noise** code is employed as a modulation waveform to “spread” the signal energy over a bandwidth much greater than the signal information bandwidth.
- At the receiver the signal is “de-spread” using a synchronized replica of the pseudo-noise code. The receiver correlates the received signals to retrieve the original information signal
- The essential idea is to spread the information signal over a wider bandwidth to make **jamming** and **interception** more difficult and to establish a **secure communications.**

Introduction

- The spread spectrum technique was developed initially for military and intelligence requirements
- Not like normal communication techniques, spread spectrum does not take into account the most valuable resources of the communication system (**power and bandwidth**)
- Spread spectrum generally makes use of a sequential noise-like signal structure to spread the normally narrowband information signal over a relatively wideband (radio) band of frequencies.

Spread Spectrum Communication System



Forms Of Spread Spectrum

- Frequency Hopping Spread Spectrum (FHSS)
- Direct-sequence Spread Spectrum (DSSS)
- Time-hopping Spread Spectrum (THSS)
- Chirp Spread Spectrum (CSS)

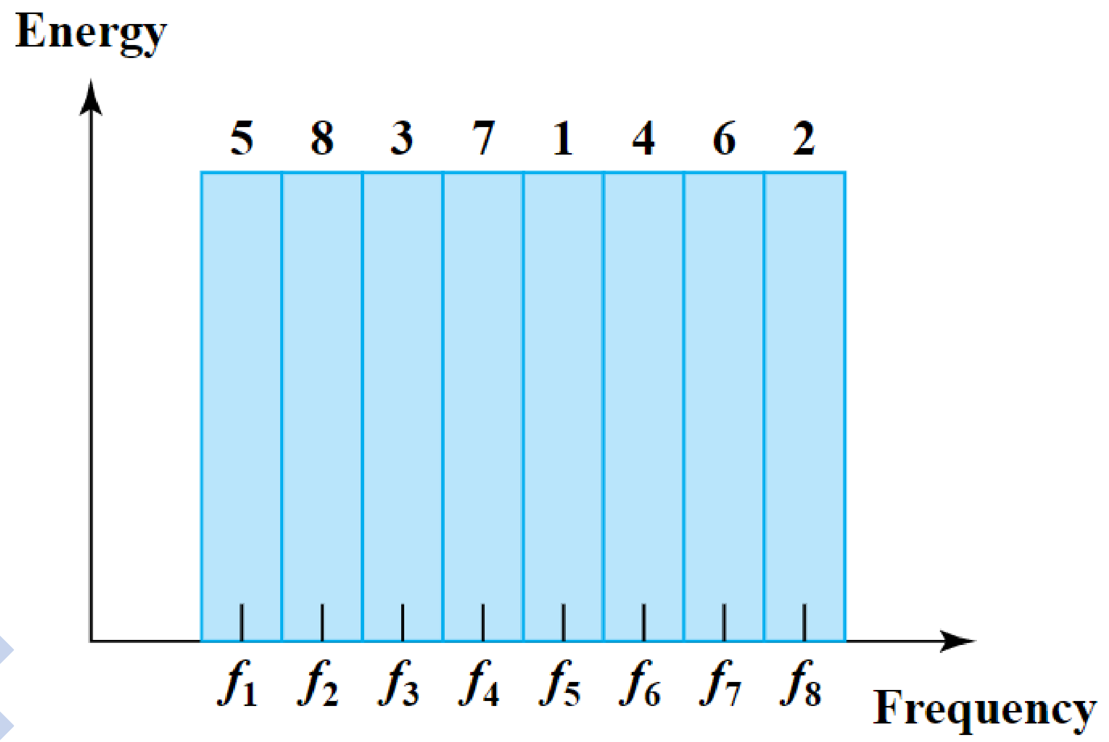
Frequency Hopping Spread Spectrum (FHSS)

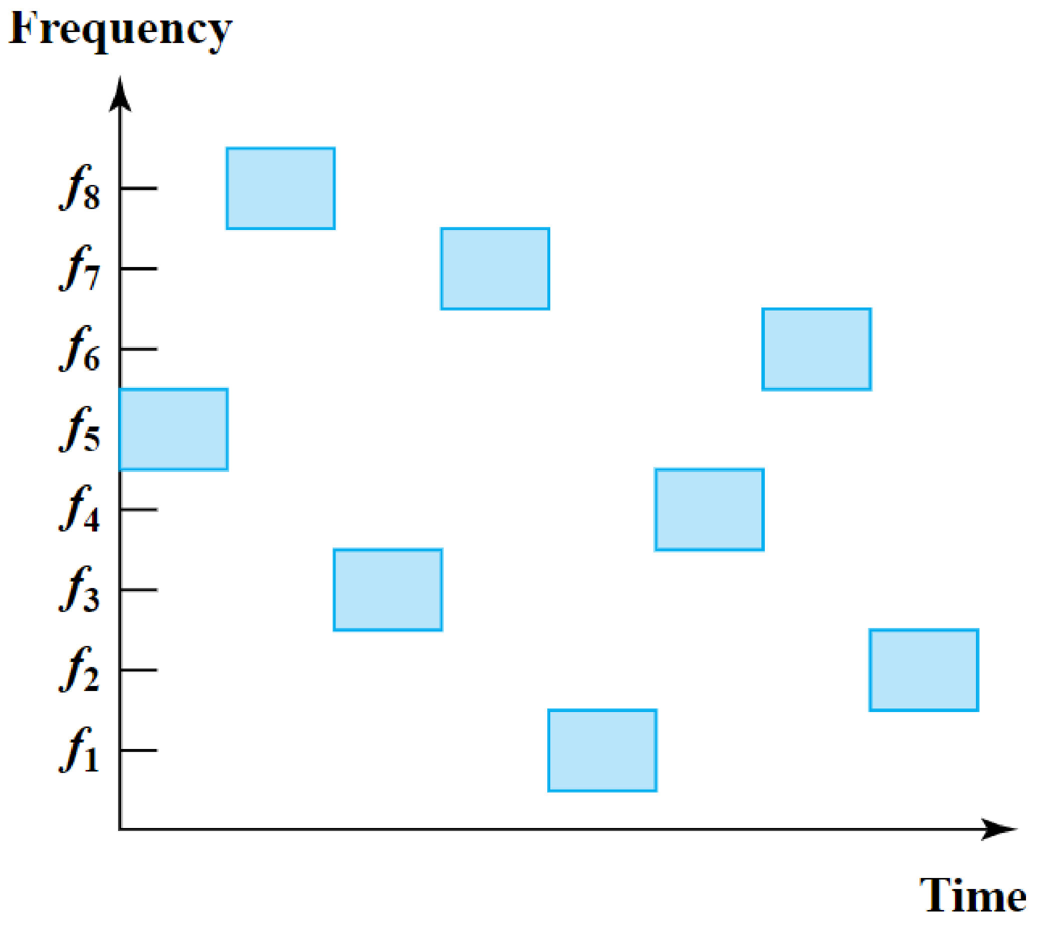
- A transmitted radio signals is spread by rapidly changing the carrier frequency among many distinct frequencies occupying a large spectral band
- The changes are controlled by a noise-like code known to both transmitter and receiver.
- The available frequency band is divided into smaller sub-bands
- Interference at a specific frequency will affect the signal only during a short interval

Frequency Hopping Spread Spectrum (FHSS)

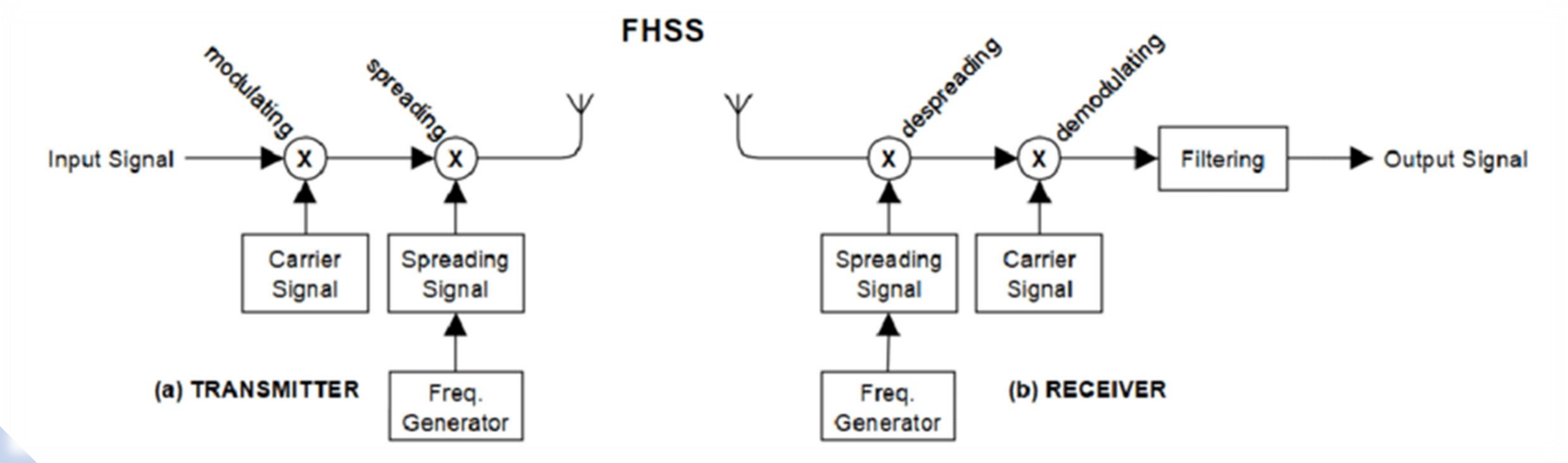
- A transmitted radio signals is spread by rapidly changing the carrier frequency among many distinct frequencies occupying a large spectral band
- The changes are controlled by a noise-like code known to both transmitter and receiver.
- The available frequency band is divided into smaller sub-bands
- Interference at a specific frequency will affect the signal only during a short interval

Frequency Hopping Spread Spectrum (FHSS)





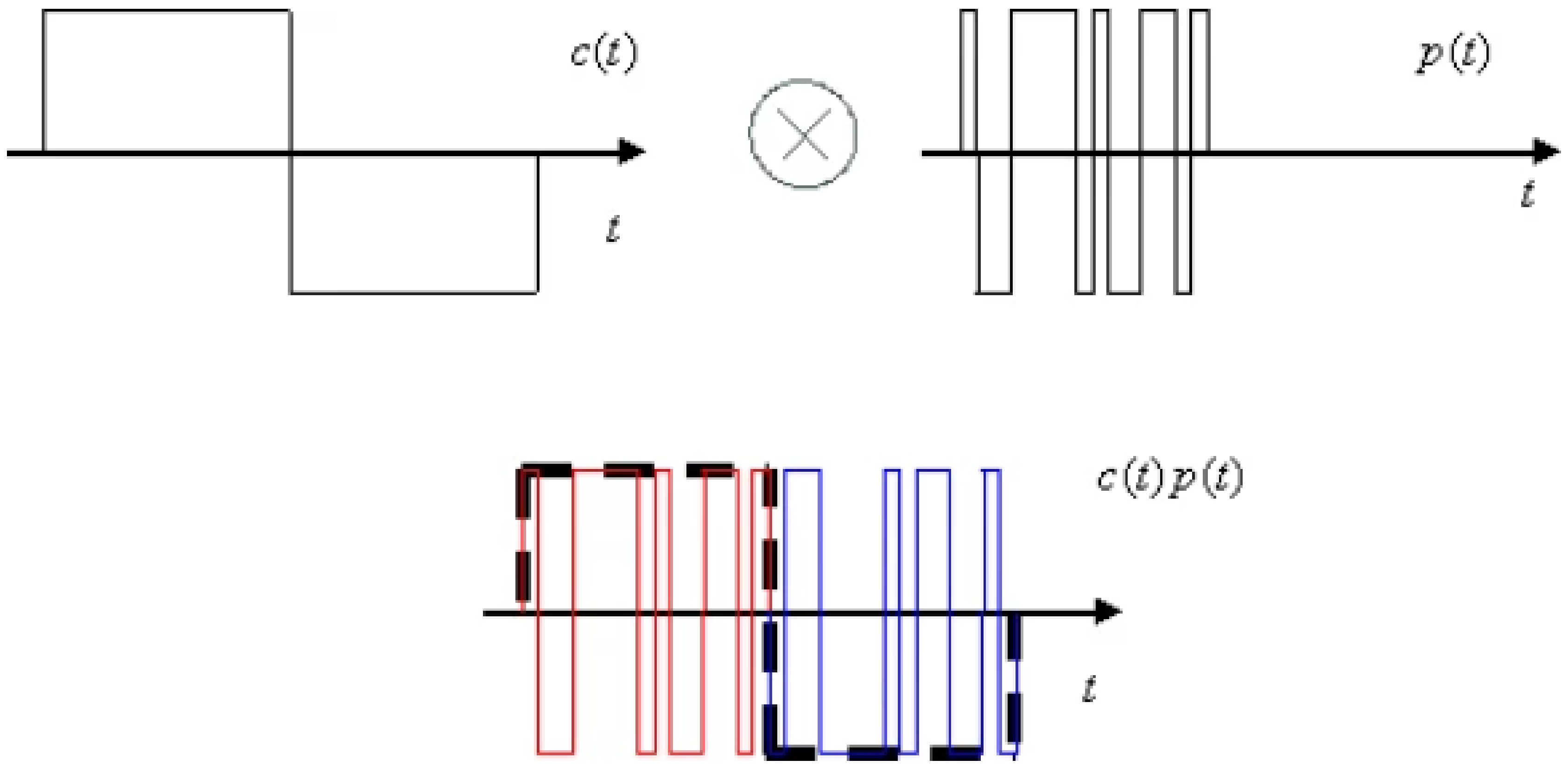
Frequency Hopping Spread Spectrum (FHSS)



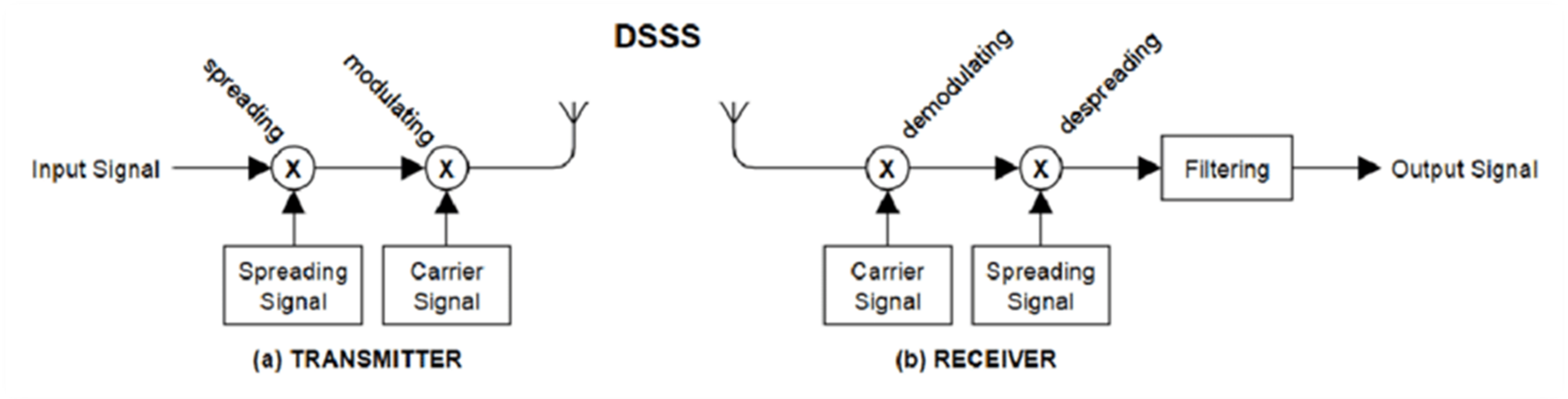
Direct Sequence Spread Spectrum (DSSS)

- The direct-sequence modulation makes the transmitted signal wider in bandwidth than the information bandwidth
- In transmission, it multiplies the data being transmitted by a pseudorandom spreading sequence that has a much higher bit rate than the original data rate
- The receiver can then use the same spreading sequence to counteract its effect on the received signal in order to reconstruct the information signal.
- If an undesired transmitter transmits on the same channel but with a different spreading sequence (or no sequence at all), the despreading process reduces the power of that signal.

Direct Sequence Spread Spectrum (DSSS)



Direct Sequence Spread Spectrum (DSSS)



Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

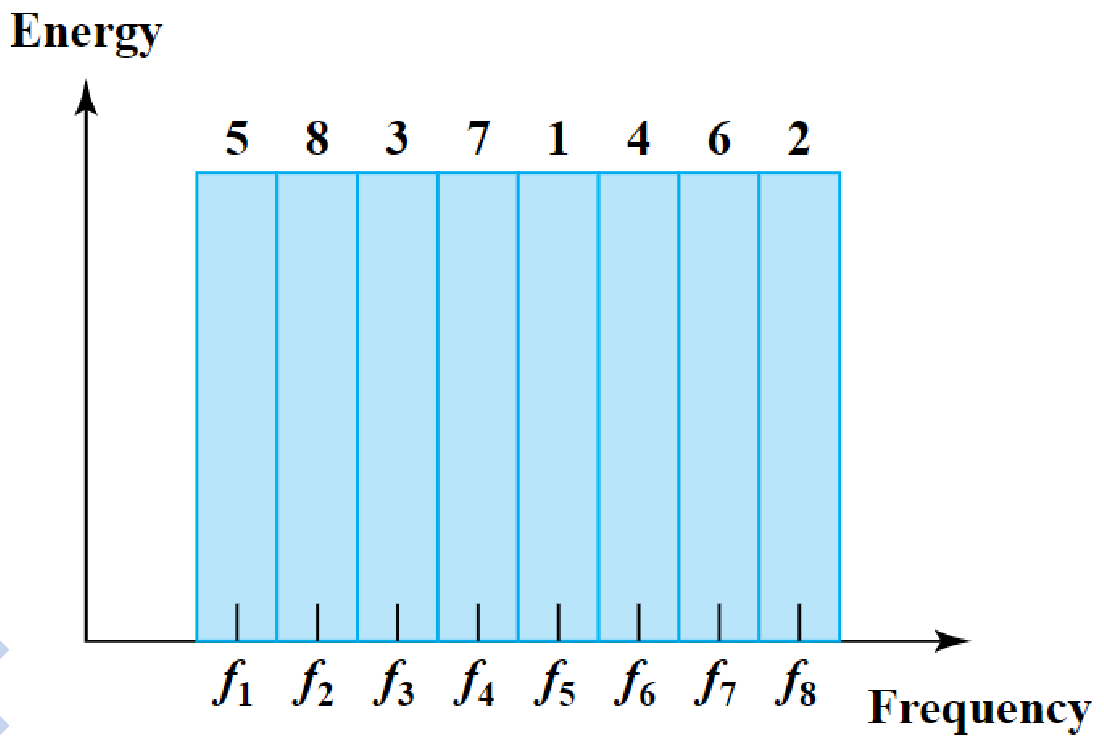
Topics

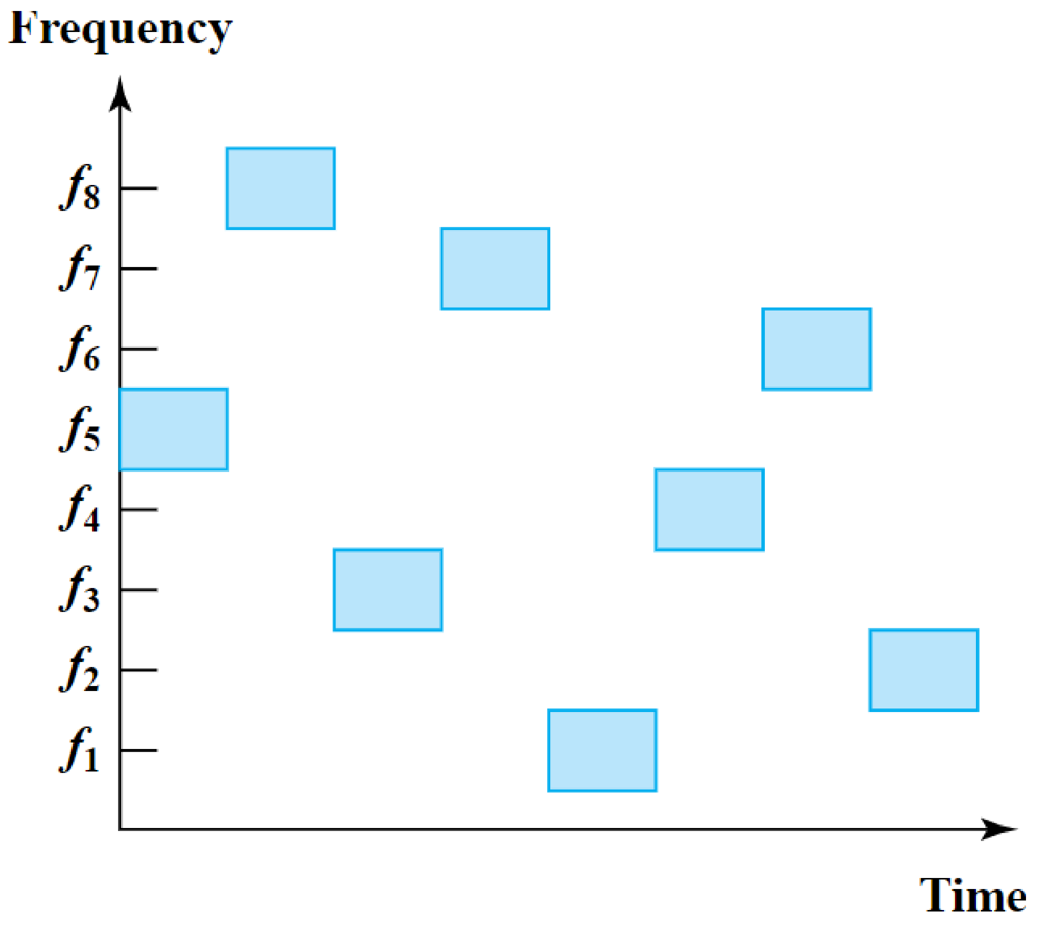
- Frequency Hopping Spread Spectrum (FHSS)
- Direct Sequence Spread Spectrum (DSSS)
- Code Division Multiple Access (CDMA)

Frequency Hopping Spread Spectrum (FHSS)

- The carrier frequency is changed according to the pseudorandom noise or sequence injected
- Prevents the loss of data and **limits noise**, **crosstalk**, and **electromagnetic interference**, preserving the signal **integrity** and **reliability** of communications
- FHSS is classified into either
  - Slow frequency hopping
  - Fast frequency hopping

Frequency Hopping Spread Spectrum (FHSS)





<u>F</u>requency <u>H</u>opping <u>S</u>pread <u>S</u>pectrum (FHSS)

FHSS has the following advantages:

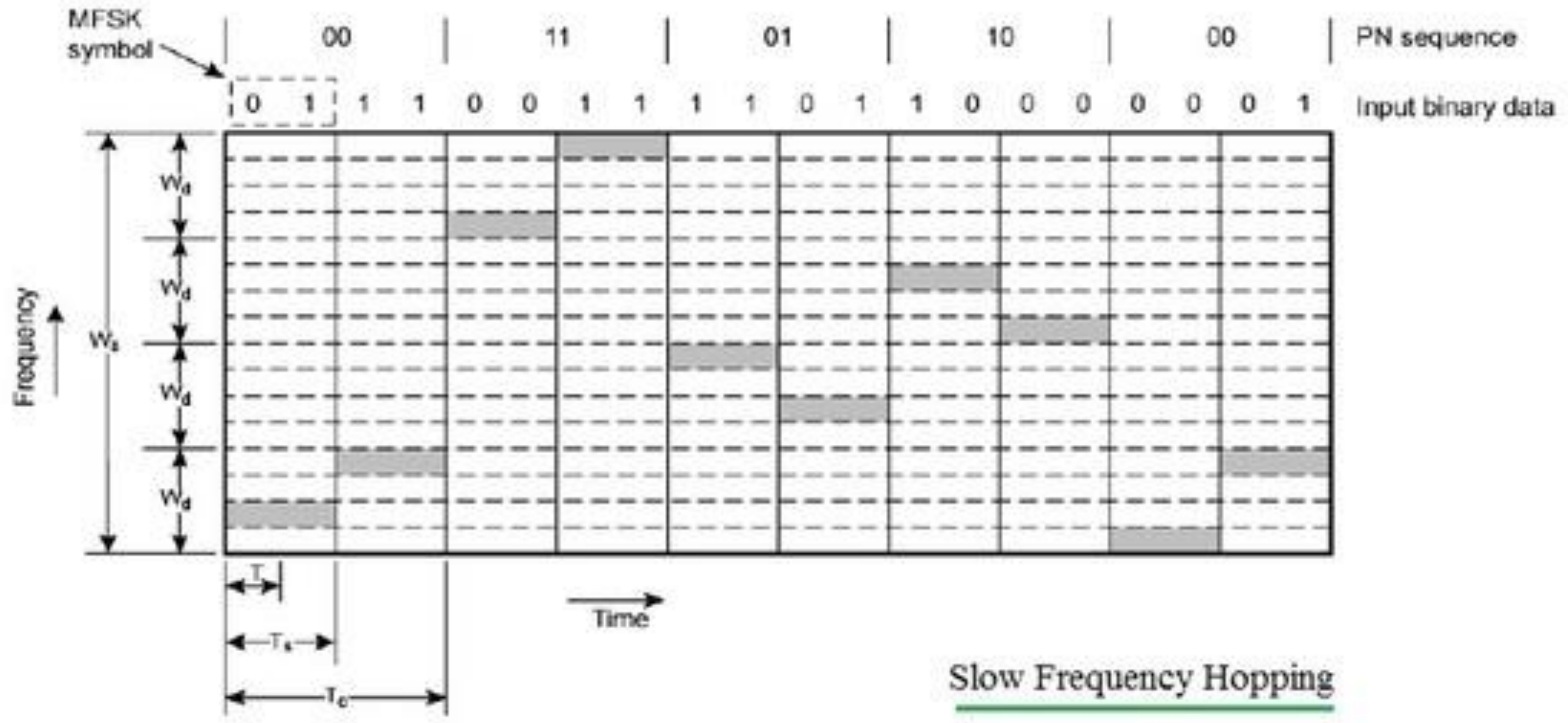
- Increased bandwidth
- Overcome fading
- Multipath channel communication
- Increased spectral efficiency
- Prevent jamming

Frequency Hopping Spread Spectrum (FHSS)

In slow frequency hopping:

- The time taken to send a single signal frequency is greater than the duration to send several bits of digital information
- More than one symbols are transmitted during the time between two frequency hops
- $Ts < Tc$
- Reduced interference in wireless communications

Frequency Hopping Spread Spectrum (FHSS)



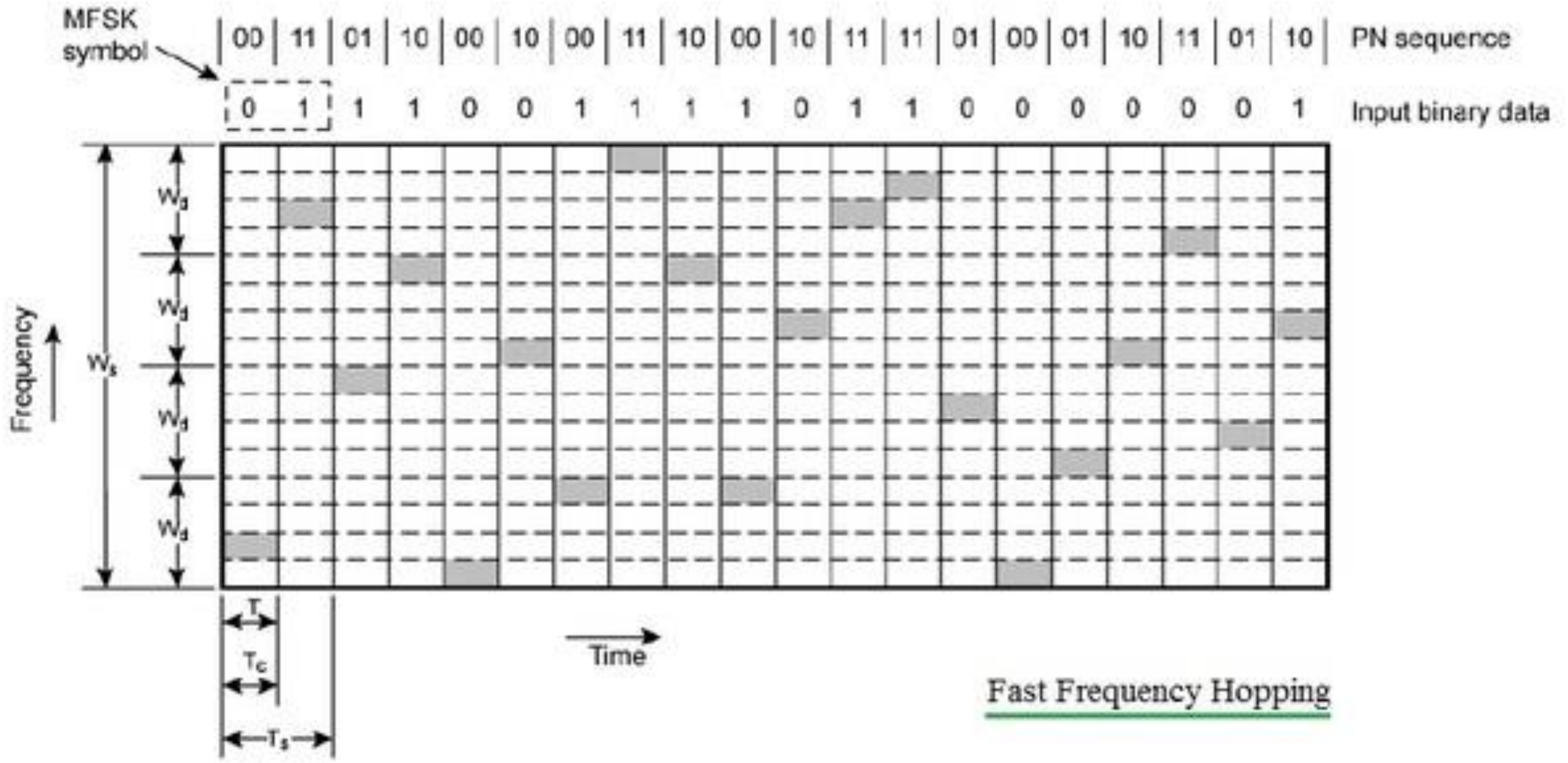
Slow Frequency Hopping

Frequency Hopping Spread Spectrum (FHSS)

In fast frequency hopping:

- Multiple frequency hops are utilized for transmitting one data bit
- Frequency hopping rate may exceed often compare to data bit rate in a binary sequence
- Ts > Tc
- Prevent Jamming in wireless communications

Frequency Hopping Spread Spectrum (FHSS)



Fast Frequency Hopping

Frequency Hopping Spread Spectrum (FHSS)

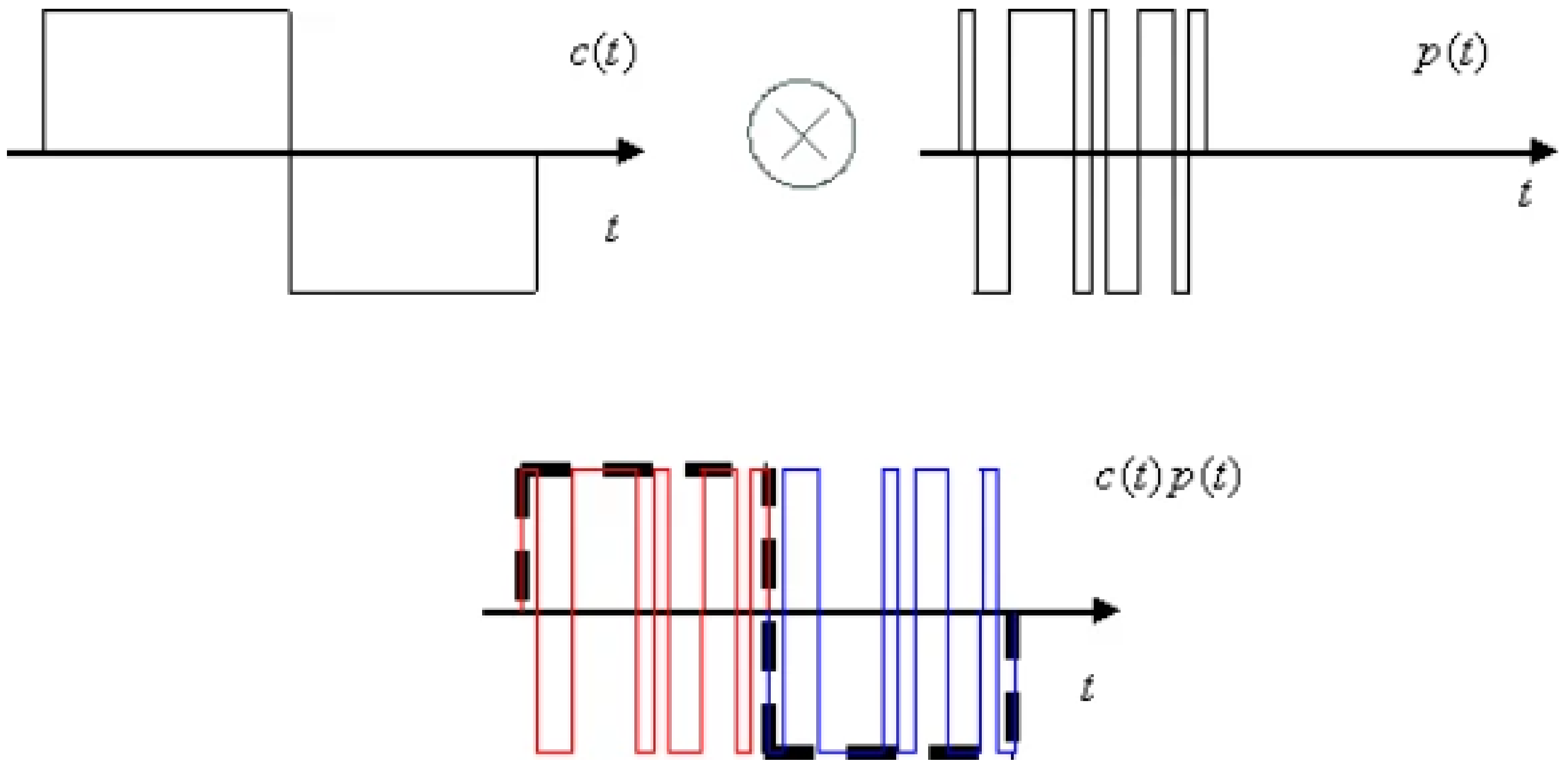
A Comparison of Slow and Fast Frequency Hopping

Slow Frequency Hopping	**Fast Frequency Hopping**
Multiple symbols transmitted in one hop	Multiple hops for transmitting one symbol
Same carrier frequency for one or more symbols	Multiple carrier frequencies for transmitting one symbol with several hops
$T_s < T_c$	$T_s > T_c$
If the carrier frequency of one hop is known, then jamming is a possibility	No possibility to jam fast frequency hopped signals, as there are multiple carrier frequencies

Direct Sequence Spread Spectrum (DSSS)

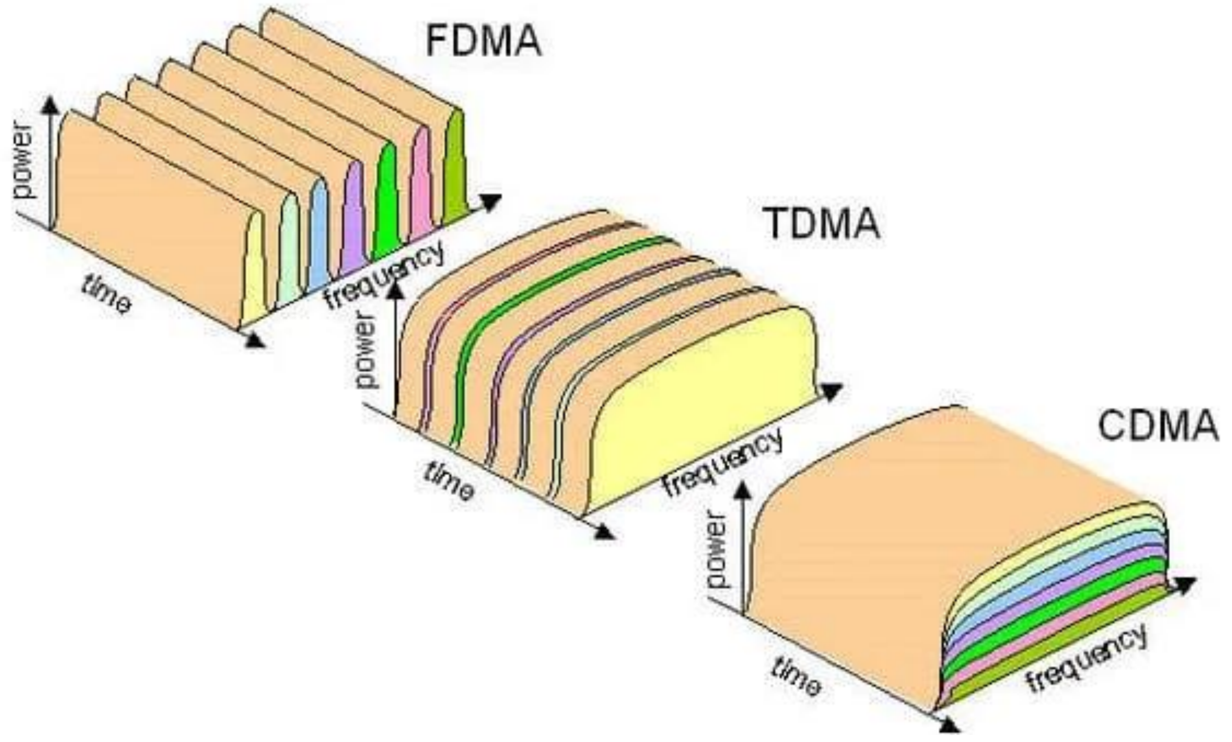
- The signal being transmitted is divided and injected with multiple frequencies within a particular frequency band
- The original data is mixed with redundant data bits or code, called chips or chipping code, and the ratio of the chips to information is called the **spreading ratio**
- A high spreading ratio indicates a wider bandwidth
- DSSS helps maintain secure signal transmission with a high signal-to-noise ratio (SNR) at the receiving end
- Allow the original data to be recovered even when a part of the transmitted data is corrupted

Direct Sequence Spread Spectrum (DSSS)



Code Division Multiple Access (CDMA)

- Code Division Multiple Access (CDMA) is a sort of multiplexing that facilitates various signals to occupy a single transmission channel.
- CDMA was developed by Qualcomm in 1993



CDMA Signal Spreading Using DSSS

- User1 :
  - data = 00
  - Spreading code = 0101
- User2 :
  - data = 10
  - Spreading code = 0011
- User3 :
  - data = 11
  - Spreading code = 1001

Differences Between FHSS and DSSS

FHSS	DSSS
FHSS changes the frequency, and the hopping of frequency follows a pattern known to the sender and receiver	DSSS changes the phase, and the carrier frequency remains in a fixed frequency band
Lower signal transmission rate (up to 3Mbps)	Higher signal transmission rate (up to 11 Mbps)
FHSS is a robust spread spectrum technique that is suitable to employ in harsh environments	DSSS is a sensitive spread spectrum technique that is influenced by harsh environmental conditions
FHSS is suitable for single point as well as multipoint communications	DSSS is suitable for point to point communication
The decoding process is simple in FHSS	To decode in DSSS, a particular algorithm is required to make the connection between the transmitter and receiver
FHSS is less reliable	DSSS is more reliable

Differences Between FHSS and DSSS

FHSS	DSSS
The analog to digital conversion in FHSS takes less time	The time taken to convert an analog signal to digital is higher
At a lower transmission rate, FHSS is cheaper	The implementation of DSSS at radio frequencies with a high transmission rate is cheaper
FHSS is not dependent on the distance of signal transmission	Distance is an influencing factor in DSSS
At a given transmitting power, FHSS offers higher power spectral density	At a given transmitting power, the wider operating spectrum of DSSS provides lower power spectral density
As the carrier frequency is varied in FHSS, it causes frequency-selective fading, where the error is bursty in nature.	In DSSS, the message bits are both frequency and time spread DSSS. This kind of spreading reduces the influence of interference and fading. The percentage error in DSSS is less than FHSS

Secure communications

Ninevah University

College of Electronics Engineering

Communication Department

Mohammed Ameer

Topics

Code sequences for spread spectrum

Code sequences for spread spectrum

- The key element in any spread spectrum technique is the use of **code sequences.**
- Codes in a spread-spectrum system are used for:
  - Protection against interference
  - Provision for privacy
  - Noise-effect reduction

Code sequences for spread spectrum

- Like random noise, the local sequence has a very low correlation with any other sequence in the set, or with the same sequence at a significantly different time offset, or with narrow band interference, or with thermal noise.
- Unlike random noise, it must be easy to generate exactly the same sequence at both the transmitter and the receiver, so the receiver's locally generated sequence has a very high correlation with the transmitted sequence.

Code sequences for spread spectrum

- In a **direct-sequence spread spectrum** system, each bit in the pseudorandom binary sequence is known as a **chip** and the inverse of its period as **chip rate**
- In a **frequency-hopping spread spectrum** sequence, each value in the pseudorandom sequence is known as a **channel number** and the inverse of its period as the **hop rate**

Code sequences for spread spectrum

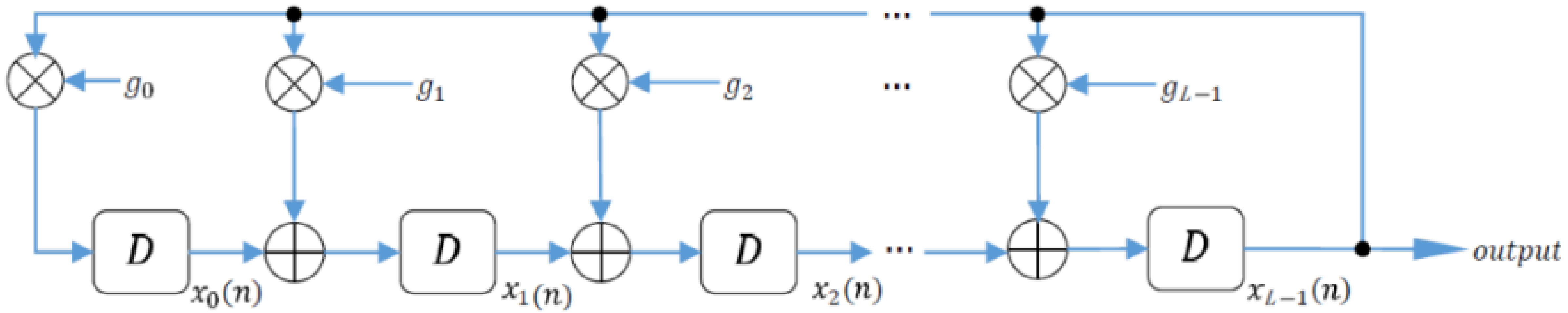
- The most popular code sequences used in spread spectrum applications are
  - Maximum-length sequences (m-sequences)
  - Gold codes
  - Walsh-Hadamard sequences
  - Orthogonal codes
  - Variable length orthogonal codes

Maximum-length sequences (m-sequences)

- Maximum-length sequence is one of the more popular coding methods in a spread-spectrum system
- Maximum-length sequences (also called as m-sequences or pseudo random (PN) sequences) are constructed based on **Galois field** theory
- It is generated by a given shift register or a delay element of given length
- The maximum length sequence is $(2^n - 1)$ chips
- A shift register generator consists of a shift register in conjunction with the appropriate logic
- Some maximal codes can be of length (7 to $[2^{36} - 1]$)

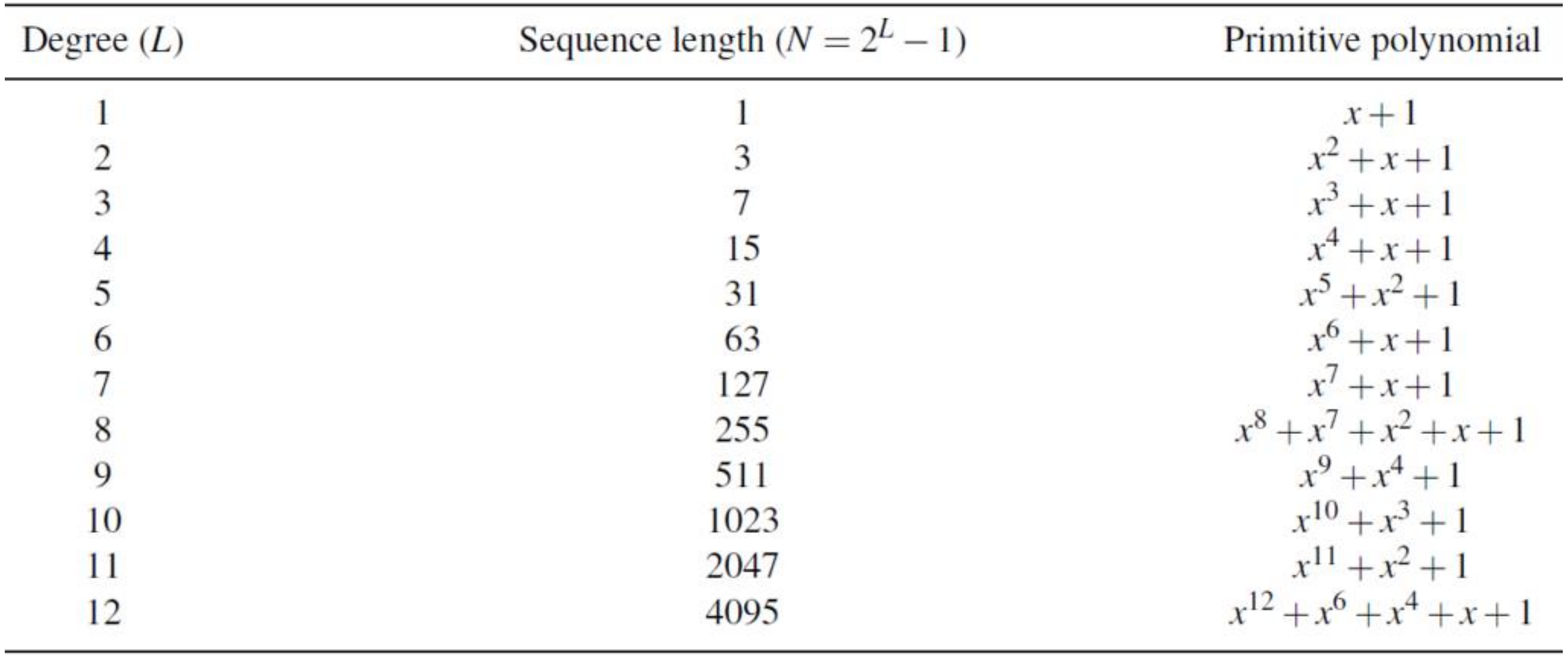
Maximum-length sequences (m-sequences)

- Maximum length sequences are generated using **linear feedback shift registers (LFSR)** structures that implement **linear recursion**.



Maximum-length sequences (m-sequences)

- For generating an m-sequence, the characteristic polynomial that dictates the feedback coefficients, should be a **primitive polynomial**



Degree (L)	Sequence length ($N = 2^L - 1$)	Primitive polynomial
1	1	$x + 1$
2	3	$x^2 + x + 1$
3	7	$x^3 + x + 1$
4	15	$x^4 + x + 1$
5	31	$x^5 + x^2 + 1$
6	63	$x^6 + x + 1$
7	127	$x^7 + x + 1$
8	255	$x^8 + x^7 + x^2 + x + 1$
9	511	$x^9 + x^4 + 1$
10	1023	$x^{10} + x^3 + 1$
11	2047	$x^{11} + x^2 + 1$
12	4095	$x^{12} + x^6 + x^4 + x + 1$

Maximum-length sequences (m-sequences)

- Properties of maximum length sequences
  - **Balance property**
  - **Run property**
  - **Correlation property**