Module 2:
Symmetric Key Cryptography

2.1 Mathematics of Symmetric Key Cryptography: Algebraic Structures, Group, Ring, Field, GF Fields

2.2 Modern Block Ciphers: Components of Modern Block Cipher, Product Ciphers, Diffusion and Confusion, Classes of Product Cipher, **DES**: DES Structure, DES Analysis: Properties, Design Criteria, DES Strength and Weaknesses, DES Security, Multiple DES, 3DES

2.3 AES: AES Structure, Transformations, Key Expansion in AES-128, Key Expansion in AES-192 and AES-256, Key- Expansion Analysis, Analysis of AES: Security, Implementation, Simplicity and Cost

Introduction

- Finite fields and other number theory topics has increasing importance in cryptography
  - AES, Elliptic Curve, IDEA, Public Key
- start with concepts of groups, rings, fields from abstract algebra
- Abstract algebra : concerned with sets whose element we can operate algebraically.

Groups

- a set of elements or “numbers”
- with some operation whose result is also in the set (closure)
- Denoted by { G, . } where (.) is any operation { + , - , * , / …. etc }
- obeys:
  - Closure
  - associative law: $(a.b).c = a.(b.c)$
  - has identity e: $e.a = a.e = a$
  - has inverses a^{-1}: $a.a^{-1} = e$

Examples

- Integers under addition $(Z, +)$
- Real numbers under addition $(R, +)$
- Non-zero real numbers under multiplication $(R^*, \times)$

Group (continued)

- Finite group : group with finite number of elements in it
- Infinite group : group with infinite number of elements in it
- Finite group has “ ORDER “ which is number of elements
- Infinite group doesn’t have “ ORDER”

Abelian group

- It is a group then it satisfies all group conditions
- Moreover, it satisfies the new condition:

- commutative:

$$a.b = b.a \quad \text{for all } a,b \text{ in } G$$

Additive Group: (R, +) is an **abelian group**

Cyclic Group

- define **exponentiation** as repeated application of operator
- identity is defined by : $e = a^0$
- Inverse element is defined by : $a^{-n} = (a')^n$
- a group is cyclic if every element is a power of some fixed element
  - i.e. $b = a^k$ for some a and every b in group
- **a** is said to be a generator of the group
- Always cyclic group is abelian group

Ring

- a set of “numbers” with two operations (addition and multiplication) which are:
- an abelian group with addition operation
- multiplication:
  - has closure
  - associative
  - distributive over addition:

$$a(b+c) = ab + ac$$

Ring

A **ring** $(R, +, \cdot)$ is a set R together with two binary operations + (addition) and $\cdot$ (multiplication) such that:

Additive Group: $(R, +)$ is an **abelian group**. This means:

- **Closure under addition**: $a + b \in R$.
- **Associativity of addition**: $(a + b) + c = a + (b + c)$.
- **Additive identity**: There exists an element $0 \in R$ such that $a + 0 = a$.
- **Additive inverse**: For every $a \in R$, there exists $-a \in R$ such that $a + (-a) = 0$.
- **Commutativity of addition**: $a + b = b + a$.

Multiplication: The multiplication operation $(\cdot)$ satisfies:

- **Closure**: For all $a, b \in R$, $a \cdot b \in R$.
- **Associativity**: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in R$.

Distributive Property: Multiplication distributes over addition:

- **Left distributivity**: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all $a, b, c \in R$.
- **Right distributivity**: $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$ for all $a, b, c \in R$.

Examples

- **Integers (Z, +, ·):** The set of integers Z under standard addition and multiplication is a **commutative ring** with unity (1 is the multiplicative identity).
- **Polynomials R[x]:** The set of polynomials with real coefficients forms a **commutative ring** under the usual addition and multiplication of polynomials.

- *Some rings have a multiplicative identity element (denoted by 1) such that $a \cdot 1 = 1 \cdot a = a$. Such rings are called* ***rings with unity****.*
- *If the multiplication operation is commutative (i.e., $a \cdot b = b \cdot a$ for all $a, b \in R$), the ring is called a* ***commutative ring****.*

Ring (continued)

- if multiplication operation is commutative, it forms a **commutative ring**

- ab=ba for all a,b in G

- if multiplication operation has inverses and no zero divisors, it forms an **integral domain**

- for all a,b if ab=0 then either a =0 or b=0
- there is an element 1 in R s.t a1=1a=a

- Ring is a set in which we can do addition , subtraction and multiplication without leaving the set

Field

- a set of numbers with two operations:(+,X)
  - abelian group for addition
  - abelian group for multiplication
- Filed satisfies all the preceding axioms (integral domain)
- It adds the following axiom:

- for all a in F (except 0 there is an element a^{-1} s.t $aa^{-1}=(a^{-1})a=1$

- A **field** $(F, +, \cdot)$ is a set F together with two binary operations + (addition) and $\cdot$ (multiplication) such that:
- **Additive Group**: (F, +) forms an **abelian group** under addition. This means:
  - **Closure under addition**: $a + b \in F$.
  - **Associativity of addition**: $(a + b) + c = a + (b + c)$.
  - **Additive identity**: There exists an element $0 \in F$ such that $a + 0 = a$.
  - **Additive inverse**: For every $a \in F$, there exists $-a \in F$ such that $a + (-a) = 0$.
  - **Commutativity of addition**: $a + b = b + a$.
- **Multiplication Forms an Abelian Group (excluding zero)**: The set F forms an abelian group under multiplication:
  - **Closure under multiplication**: $a \cdot b \in F$.
  - **Associativity of multiplication**: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
  - **Multiplicative identity**: There exists an element $1 \in F_1$, distinct from 0, such that $a \cdot 1 = a$.
  - **Multiplicative inverse**: For every $a \in F$, there exists $a^{-1} \in F$ such that $a \cdot a^{-1} = 1$.
  - **Commutativity of multiplication**: $a \cdot b = b \cdot a$.
- **Distributive Property**: Multiplication distributes over addition:
- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.
- $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$

Fields (continued)

- It is a set in which we can do addition , subtraction , multiplication and division without leaving the set
- Familiar example is rational numbers ,real numbers and complex numbers
- Division is defined by the following rule:

$$- a / b = a(b^{-1})$$

Basis	Group	Ring	Field
Operations	One (Addition or Multiplication)	Two (Addition and Multiplication)	Two (Addition and Multiplication)
Inverse for Multiplication	Not Required	Not Required	Required for all non-zero elements
Distributive Law	Not Needed	Required (multiplication spreads over addition)	Required (just like in rings)
Commutativity	Only for some groups	Only for Addition	Both Addition and Multiplication
Example	Integers under addition	Integers with addition and multiplication	Rational numbers with addition and multiplication

Modern Block Ciphers

- A block cipher works by dividing the data into fixed-size blocks (usually 64 or 128 bits) and encrypting each block individually.
- This is different from stream ciphers, which process data bit by bit. Block ciphers ensure security by using two essential properties:
- confusion (making the relationship between the ciphertext and key as complex as possible) and
- diffusion (spreading out the influence of one plaintext symbol over many ciphertext symbols).

Data Encryption Standard (DES)

- DES set the foundation for block cipher design,
- but over time, it became vulnerable to attacks due to the increase in computational power.
- To address these limitations, modern block ciphers like Advanced Encryption Standard (AES) were introduced,
- providing stronger security and better performance.

S-Box

- An S-Box, or Substitution Box, takes a small set of input bits and transform them into a different set of output bits.
- This process is known as substitution, where one set of values is substituted for another to confuse the relationship between the plaintext and the ciphertext.
- In simpler terms, the S-Box replaces bits of the input with new bits according to a predefined table. This substitution makes it extremely difficult for attackers to predict or reverse-engineer the encryption process without knowing the key.
- The S-Box is predefined by the designers of the encryption algorithm and is standardized to provide consistent, secure encryption.
- A well-designed S-Box ensures that even small changes in the input result in unpredictable changes in the output, enhancing security.

Simple Example:

Input (2 bits)	Output (2 bits)
00	11
01	10
10	01
11	00

Role of the S-Box in Encryption:

- The goal of the S-Box is to introduce confusion in the encryption process.
- Confusion means that the relationship between the key and the ciphertext is obscured.
- An attacker should not be able to easily deduce what key was used just by analyzing the ciphertext.
- To make it effective, modern block ciphers typically use many S-Boxes, performing multiple rounds of substitution on the data, often combined with other transformations (like permutations).
- This makes it extremely difficult to reverse the encryption without knowing the key.

P-Box

- The P-Box, or Permutation Box, role is to shuffle the positions of the input bits without changing their values.
- This process is known as permutation, and it is essential for adding diffusion to the encryption process.
- Diffusion ensures that each bit of the plaintext influences many bits of the ciphertext, making it harder for attackers to trace the encryption back to the original message.
- In simpler terms, the P-Box rearranges bits in the input data according to a predefined pattern. While the S-Box changes the actual values of the bits, the P-Box only changes their positions.
- The P-Box is predefined by the encryption algorithm's designers and is standardized to ensure security.
- The purpose of the P-Box is to spread the effect of individual bits throughout the entire data block, making it difficult for attackers to find patterns that could help them break the cipher.

Input Position	Output Position
1	4
2	2
3	6
4	8
5	1
6	7
7	5
8	3

Role of the P-Box in Encryption:

- The main goal of the P-Box is to introduce diffusion into the encryption process.
- Diffusion means that changes in the plaintext should spread throughout the ciphertext.
- Even a small change in the plaintext should result in significant, unpredictable changes in the ciphertext.
- Similar to the S-Box, modern block ciphers often use multiple P-Boxes in various rounds of encryption, combined with other transformations.
- This makes the overall encryption process highly secure and resistant to cryptanalysis.

Role of XOR in Encryption:

- XOR is crucial in cryptography because of its reversibility.
- If you apply XOR twice with the same key, you get back the original data.
- This property allows encryption and decryption to work: encrypt with XOR, and decrypt by applying XOR again with the same key.

- In modern block ciphers, XOR is used repeatedly in various stages of encryption to ensure that changes in the plaintext or key result in unpredictable changes in the ciphertext, thereby enhancing security.

Circular Shift

- A Circular Shift, also known as a rotation, is a technique used in modern block ciphers to rearrange bits in a specific way.
- Unlike a standard shift, where bits are moved to the left or right with zeros filling the gaps, a circular shift wraps the bits around, creating a continuous loop.

Role of Circular Shift in Encryption:

- The main goal of using circular shifts in encryption is to achieve better diffusion.
- This obscures the relationship between the plaintext and ciphertext, making it harder for attackers to decipher the encrypted data.
- In modern block ciphers, circular shifts are applied in multiple rounds, ensuring extensive transformation of the data, which enhances overall security.

Product Cipher

- A Product Cipher is a type of encryption that combines multiple encryption methods to enhance security.
- By using more than one encryption technique, it increases the complexity of the encryption process, making it more resistant to cryptographic attacks.

- This approach typically involves applying a series of transformations, such as substitutions and permutations, in succession.
- The idea is to layer these operations to obscure the relationship between the plaintext and ciphertext.

Feistel Cipher

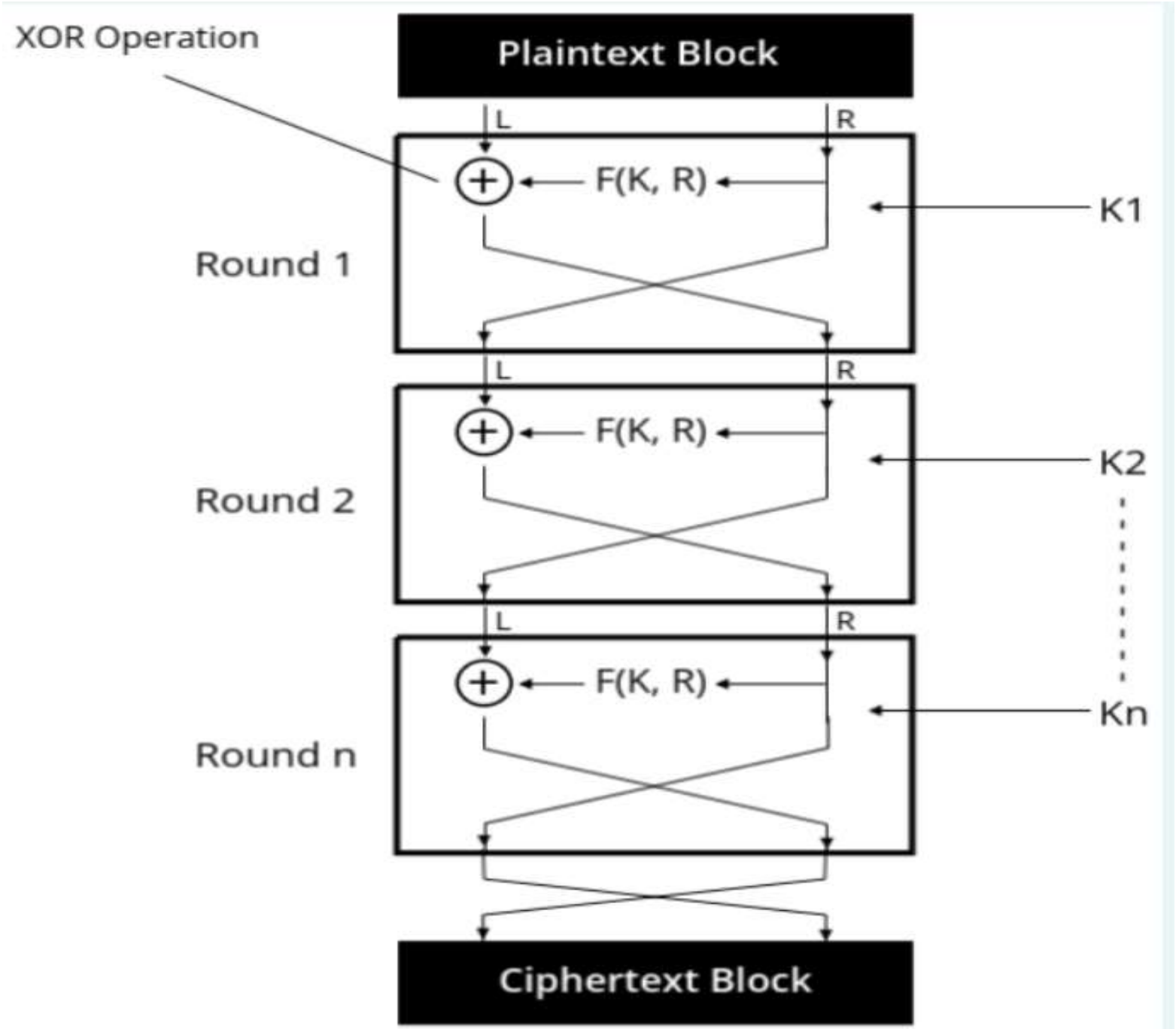
- The Feistel cipher is a symmetric encryption method used in modern block ciphers.
- Named after Horst Feistel,
- this design breaks the encryption process into multiple rounds, enhancing security while maintaining efficiency.
- The structure allows for both encryption and decryption to use the same algorithm, making it versatile.

How the Feistel Cipher Works

- The Feistel cipher operates on a block of plaintext, dividing it into two halves.
- The encryption process involves multiple rounds, where each round applies a series of transformations using a round key derived from the original encryption key.

Steps in Feistel Cipher

- **Input:** The plaintext is split into two halves: *Left (L)* and *Right (R)*.
- **Round Function:** In each round, a function (often denoted as *F*) processes the right half and the round key (K1, K2, etc.) through an XOR operation.
- **XOR Operation:** The output of the round function is then XORed with the left half.
- **Swapping:** The left and right halves are then swapped. The original right half becomes the new left half, and the XOR-modified left half becomes the new right half.
- **Repetition:** This process is repeated for a specified number of rounds (usually 16 or more).
- **Final Output:** After completing all rounds, the combined halves produce the final ciphertext.


XOR Operation
Plaintext Block
L
R
F(K, R)
K1
Round 1
L
R
F(K, R)
K2
Round 2
L
R
F(K, R)
Kn
Round n
Ciphertext Block

Data Encryption Standard (DES)

- Symmetric key-block cipher which means its having same key for encryption + decryption
- published by (NIST) National institute of standards and technology
- DES is block cipher
- DES uses 16 rounds and each round is a fiestel round
- It became widely used for secure data transmission. DES encrypts data in fixed-size blocks of 64 bits and uses a 56-bit key to control the encryption process.

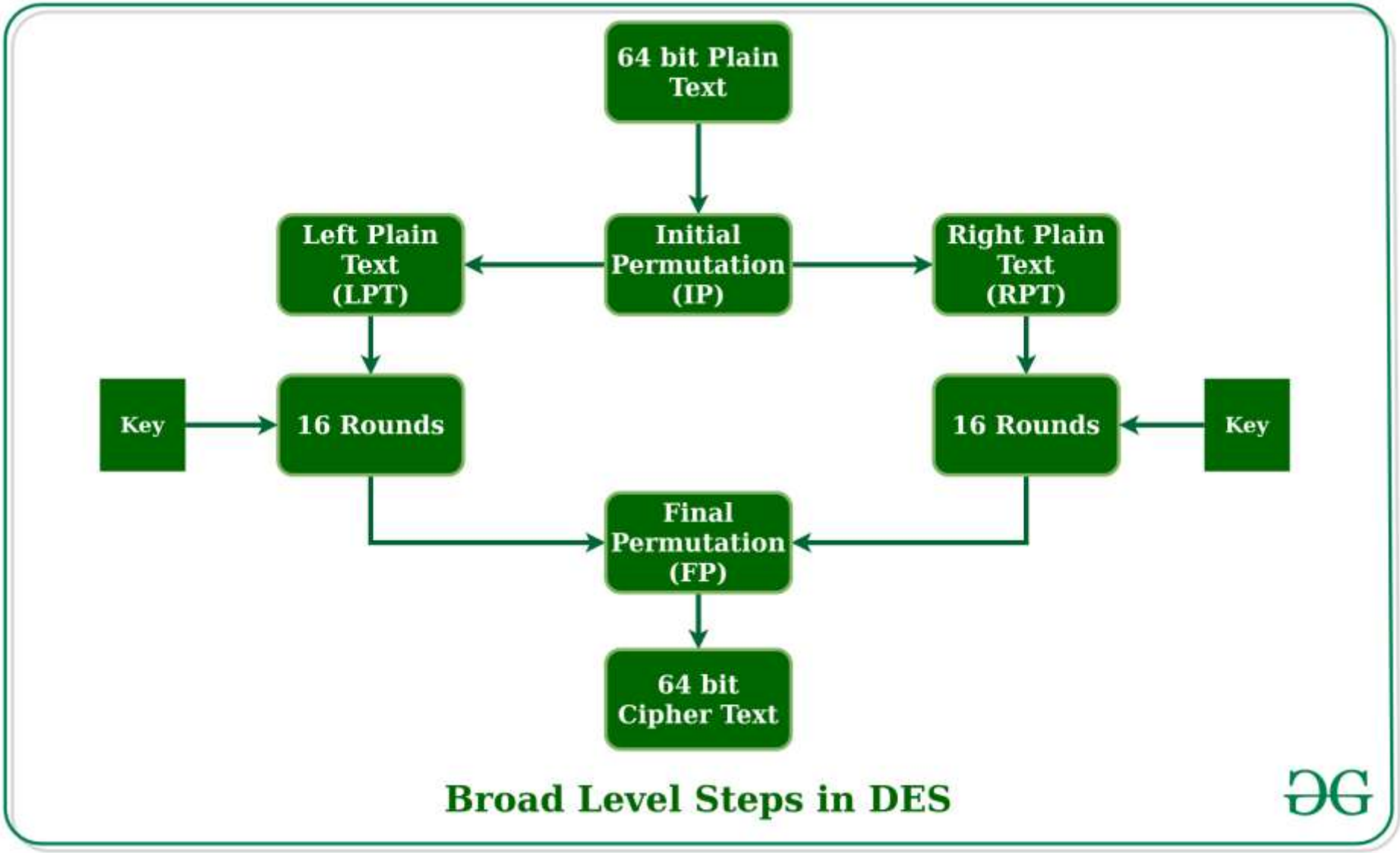
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Figure - discording of every 8th bit of original key

- However, due to advances in computing power, DES has become less secure over time because its key length (56 bits) is vulnerable to brute-force attacks.
- For this reason, DES has largely been replaced by stronger encryption algorithms like AES, but understanding DES is important for learning about the history of cryptography and encryption techniques.

- DES is based on the two fundamental attributes of cryptography:
- substitution (also called confusion) and transposition (also called diffusion).
- DES consists of 16 steps, each of which is called a round.
- Each round performs the steps of substitution and transposition.

- In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation is performed on plain text.
- Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT go through 16 rounds of the encryption process.
- In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit ciphertext.



Broad Level Steps in DES

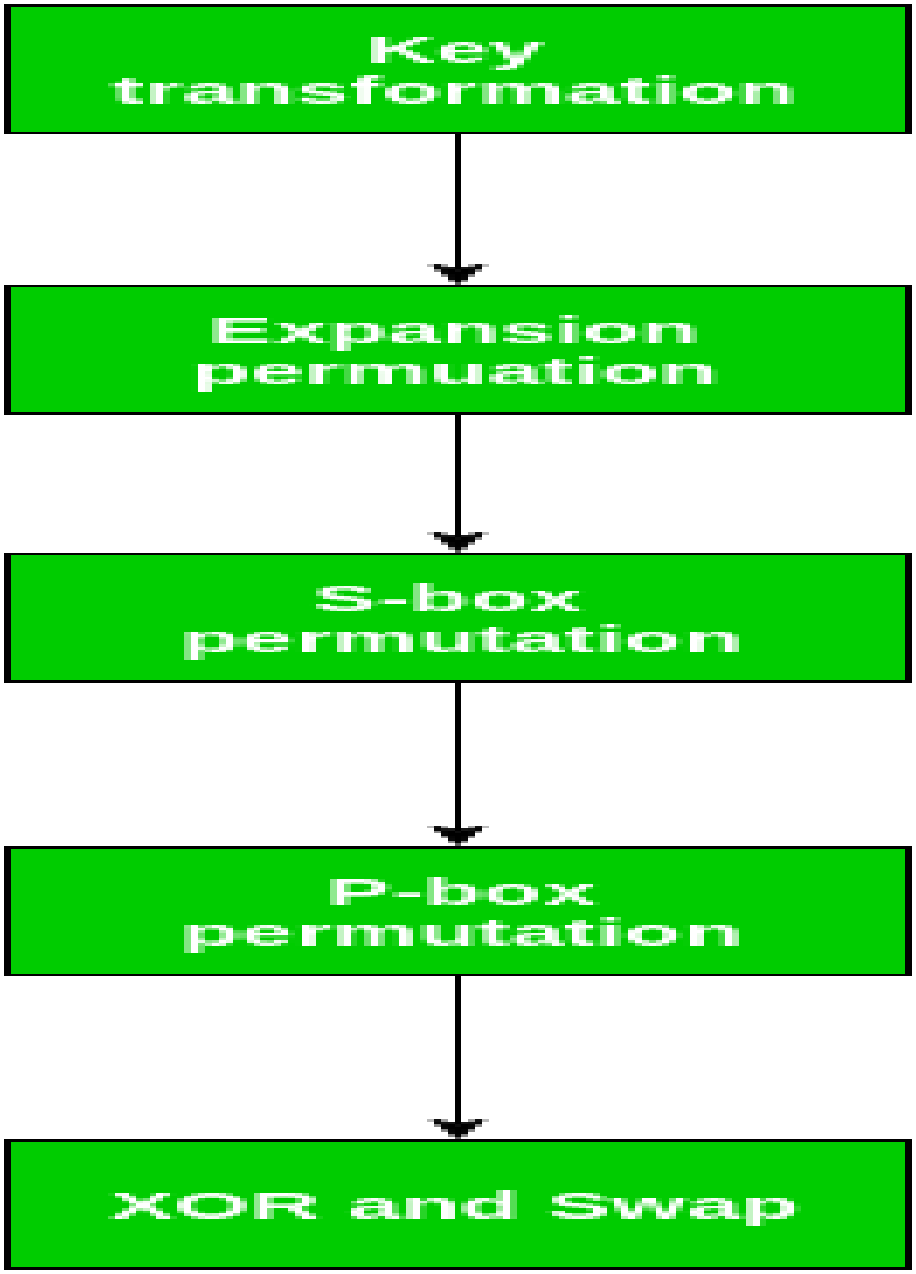
Initial Permutation (IP)

- The initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed.

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	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

- The resulting 64-bit permuted text block is divided into two half blocks. Each half-block consists of 32 bits, and each of the 16 rounds,



Step 1: Key transformation

- From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called key transformation.
- For this, the 56-bit key is divided into two halves, each of 28 bits.
- These halves are circularly shifted left by one or two positions, depending on the round.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Figure - number of key bits shifted per round

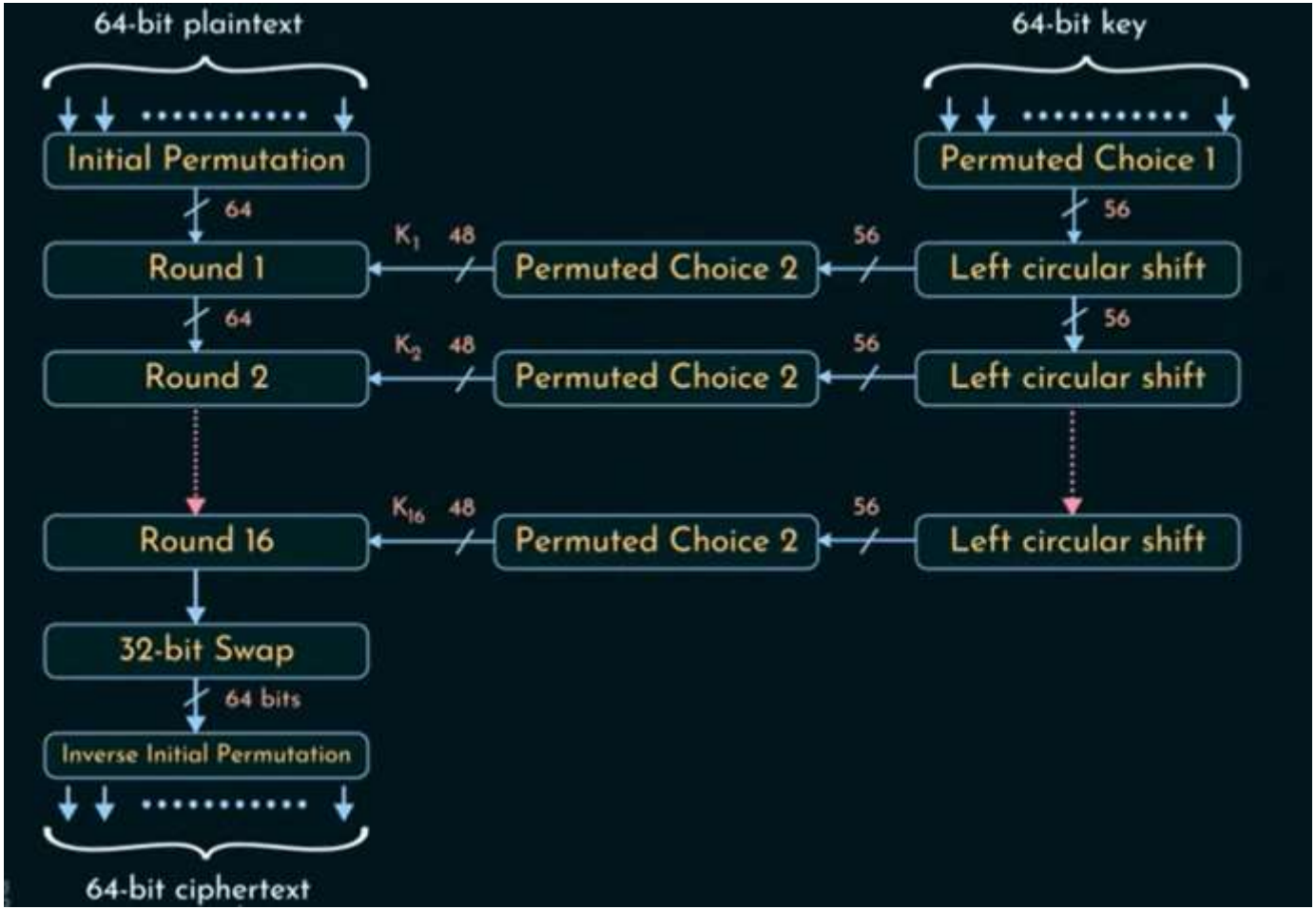
- After an appropriate shift, 48 of the 56 bits are selected. From the 48 we might obtain 64 or 56 bits based on requirement which helps us to recognize that this model is very versatile and can handle any range of requirements needed or provided.

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

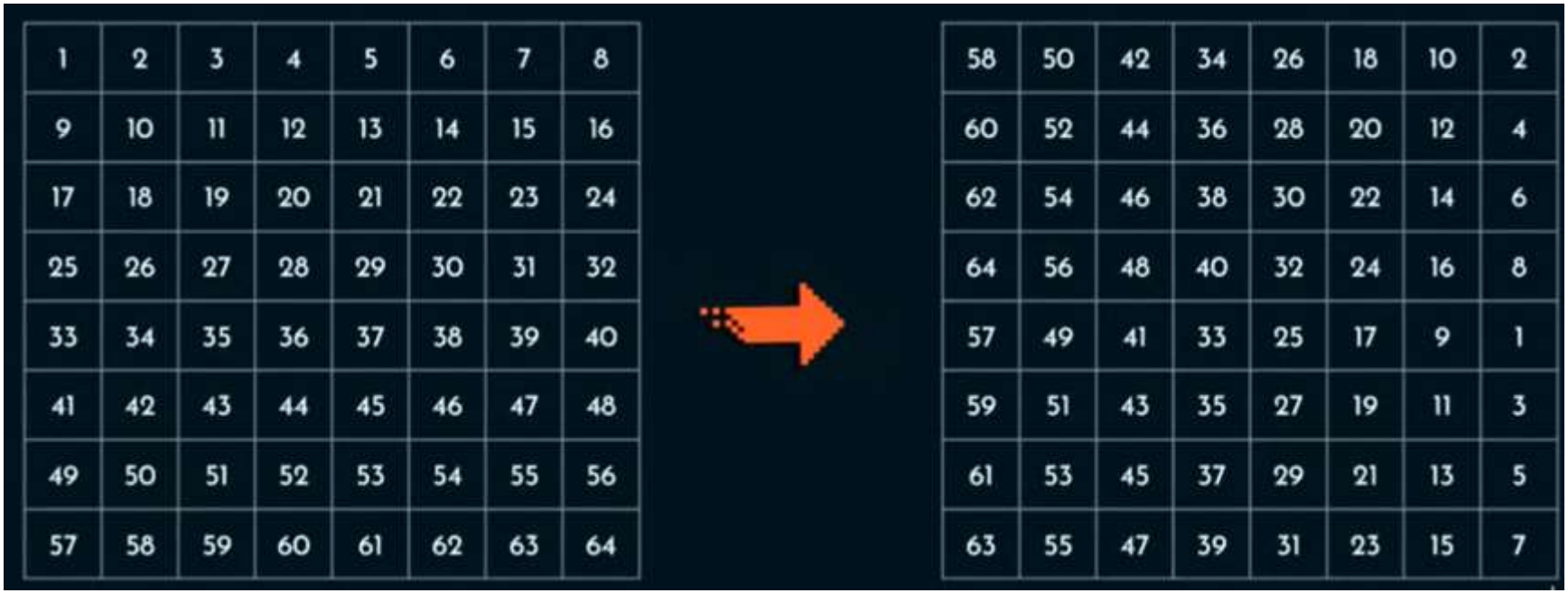
Figure - compression permutation

- Symmetric Block Cipher.
- A.k.a Data Encryption Algorithm.
- Adopted by NIST in 1977.
- Advanced Encryption Standard (AES) in 2001.
- Input : 64 bits.
- Output : 64 bits.
- Main Key : 64 bits.
- Subkey : 56 bits.
- Round key : 48 bits
- No. of rounds : 16 rounds.

- Symmetric Block Cipher.
- A.k.a Data Encryption Algorithm.
- Adopted by NIST in 1977.
- Advanced Encryption Standard (AES) in 2001.
- Input : 64 bits.
- Output : 64 bits.
- Main Key : 64 bits.
- Subkey : 56 bits.
- Round key : 48 bits
- No. of rounds : 16 rounds.


64-bit plaintext
64-bit key
Initial Permutation
Permuted Choice 1
64
56
K1
48
56
Round 1
Permuted Choice 2
Left circular shift
64
56
K2
48
56
Round 2
Permuted Choice 2
Left circular shift
K16
48
56
Round 16
Permuted Choice 2
Left circular shift
32-bit Swap
64 bits
Inverse Initial Permutation
64-bit ciphertext

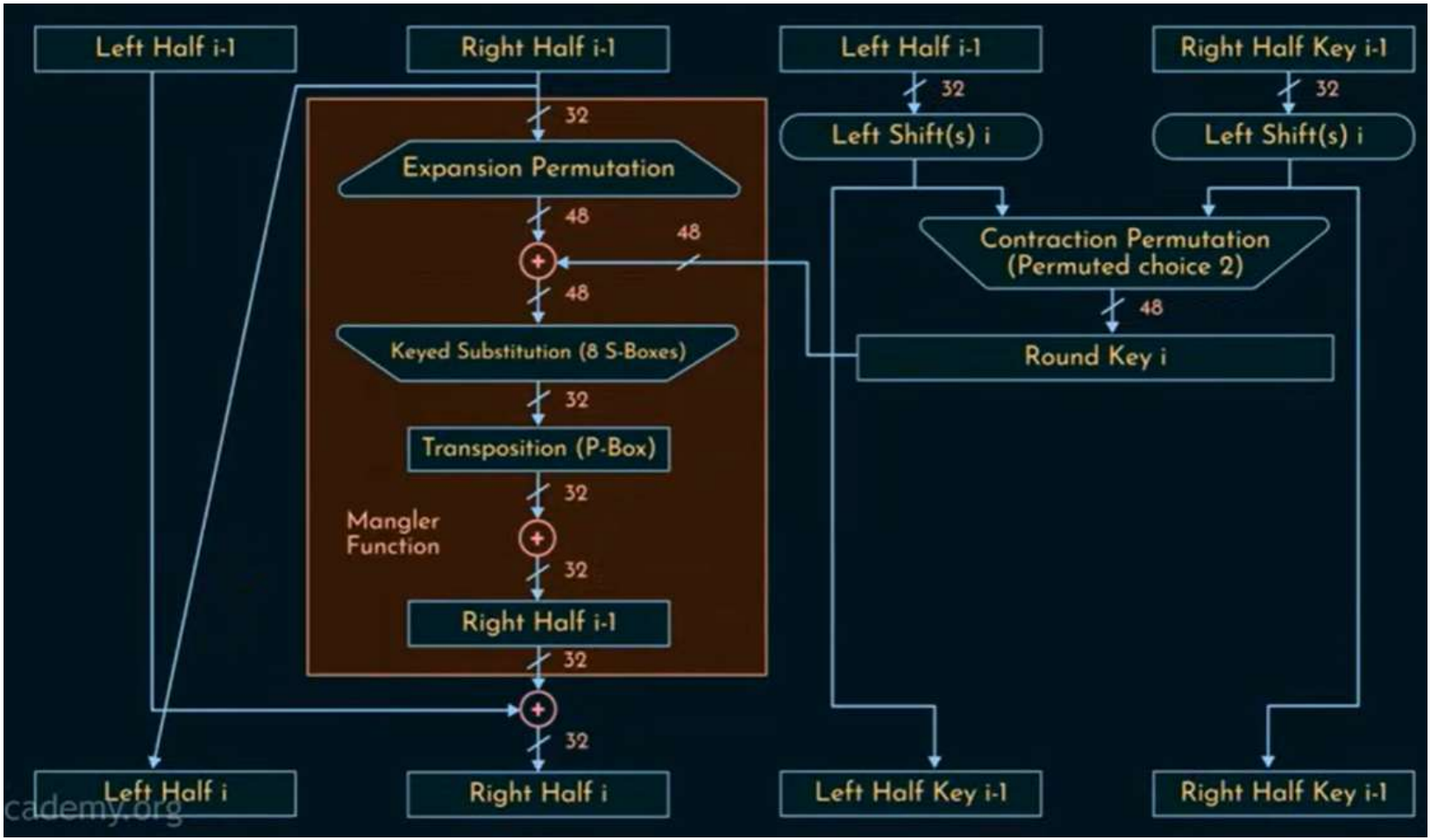
Initial permutation



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

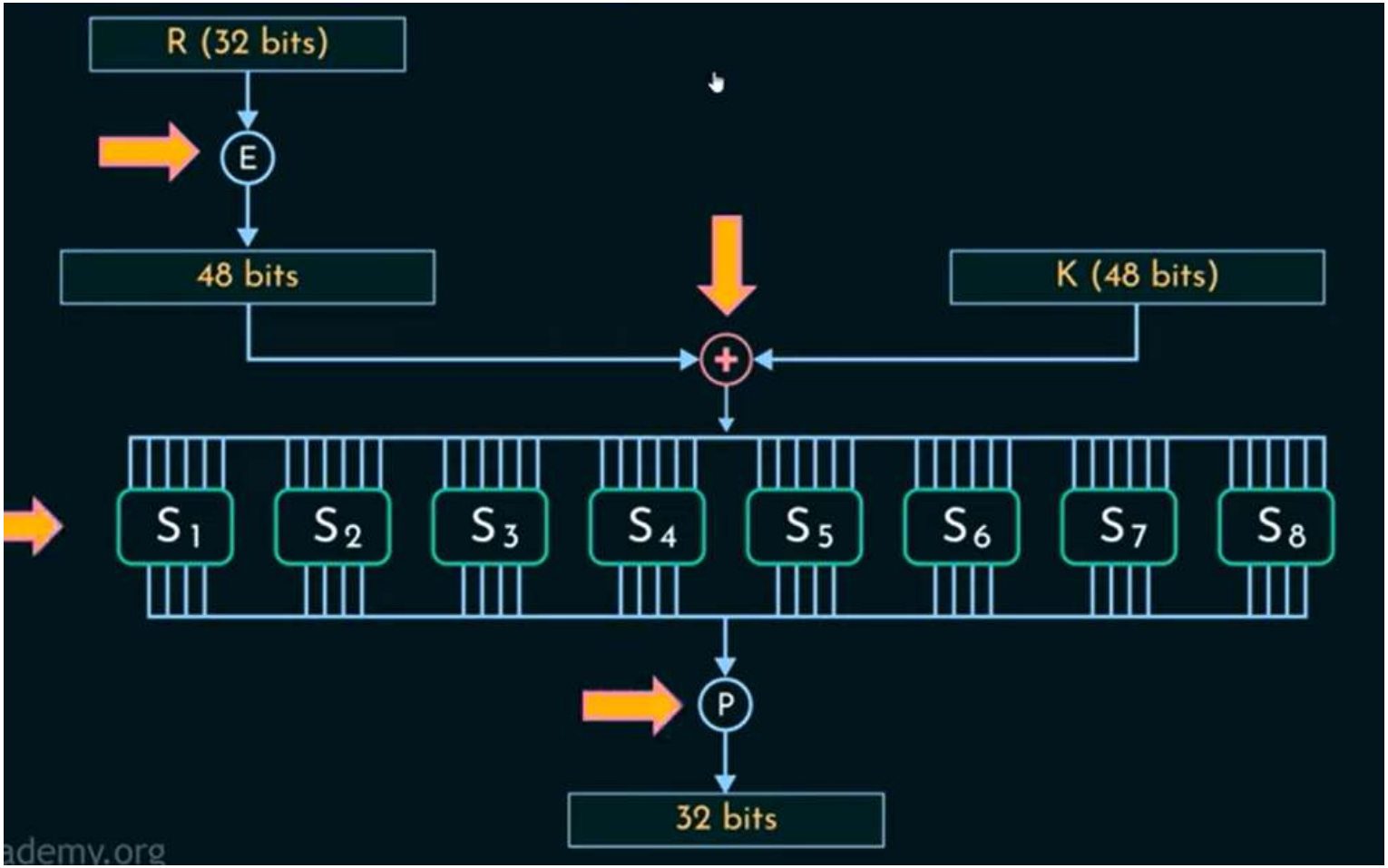
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

One round of DES



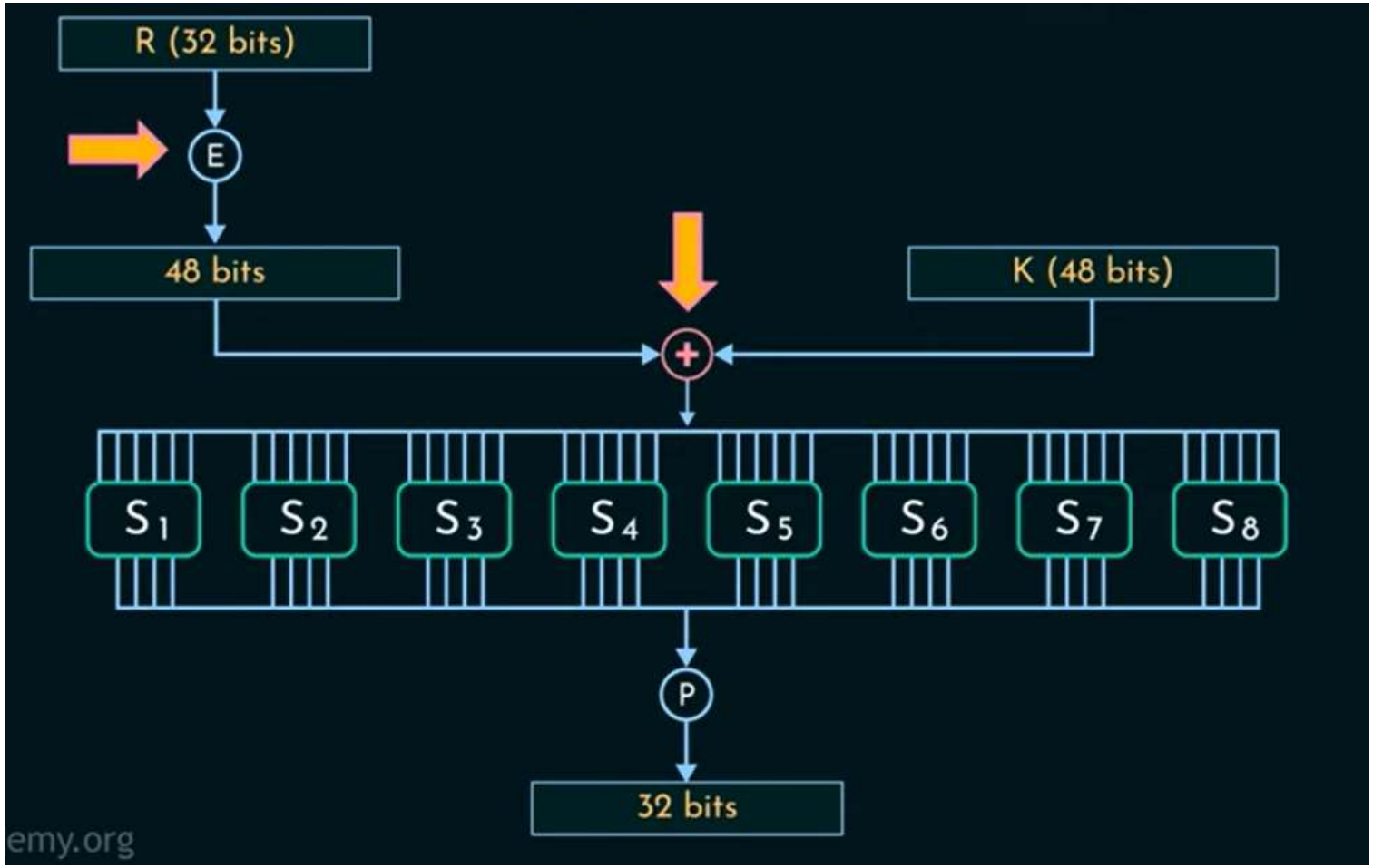
$$L_i = R_{i-1}$$

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


R (32 bits)
E
48 bits
K (48 bits)
+
S1
S2
S3
S4
S5
S6
S7
S8
P
32 bits

Expansion

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


R (32 bits)
E
48 bits
K (48 bits)
+
S1
S2
S3
S4
S5
S6
S7
S8
P
32 bits
emy.org

Box S_1

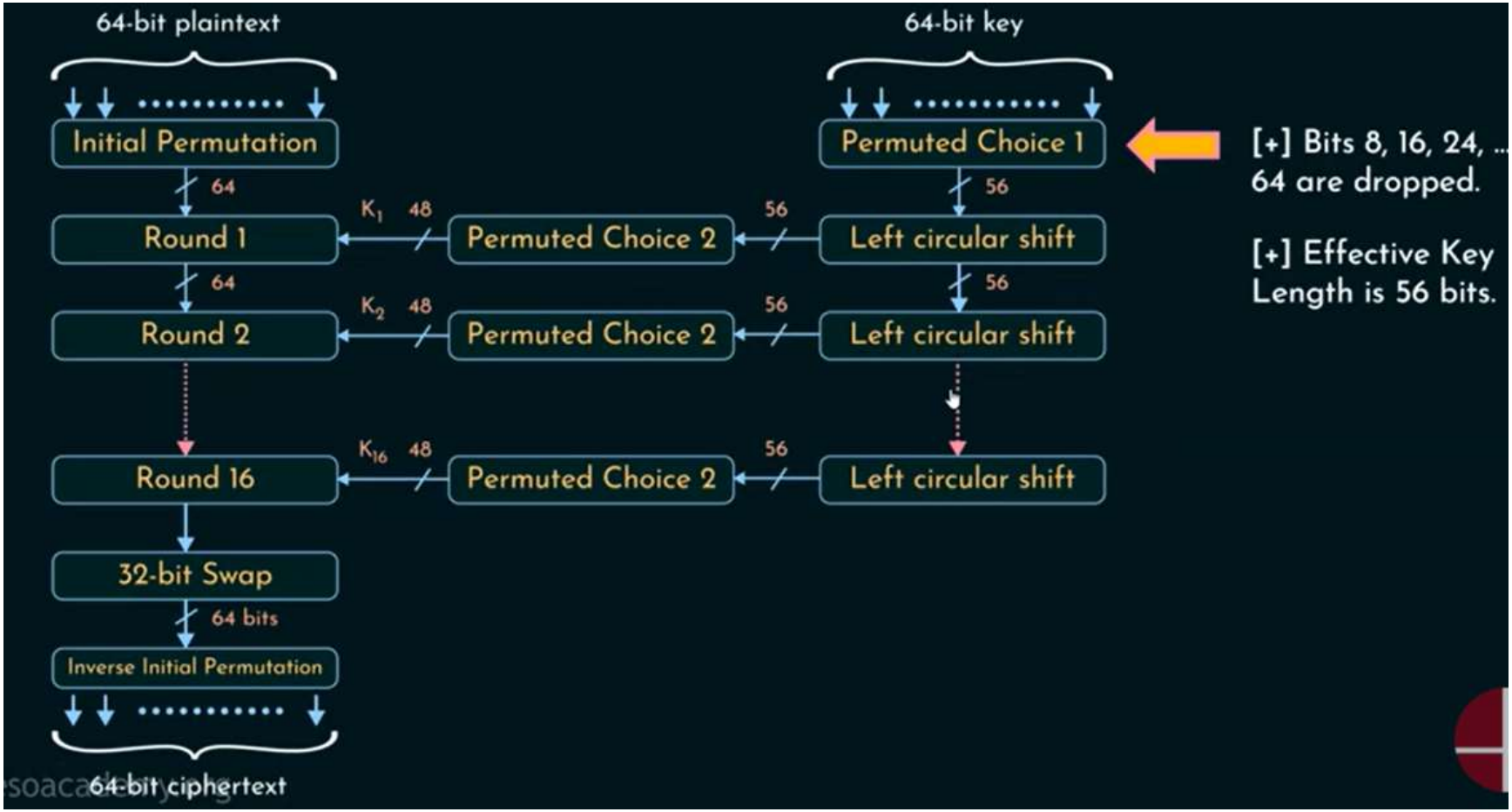
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	6	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

For example, $S_1(101010) = 6 = 0110$.

Permutation Function

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

Key Scheduling



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

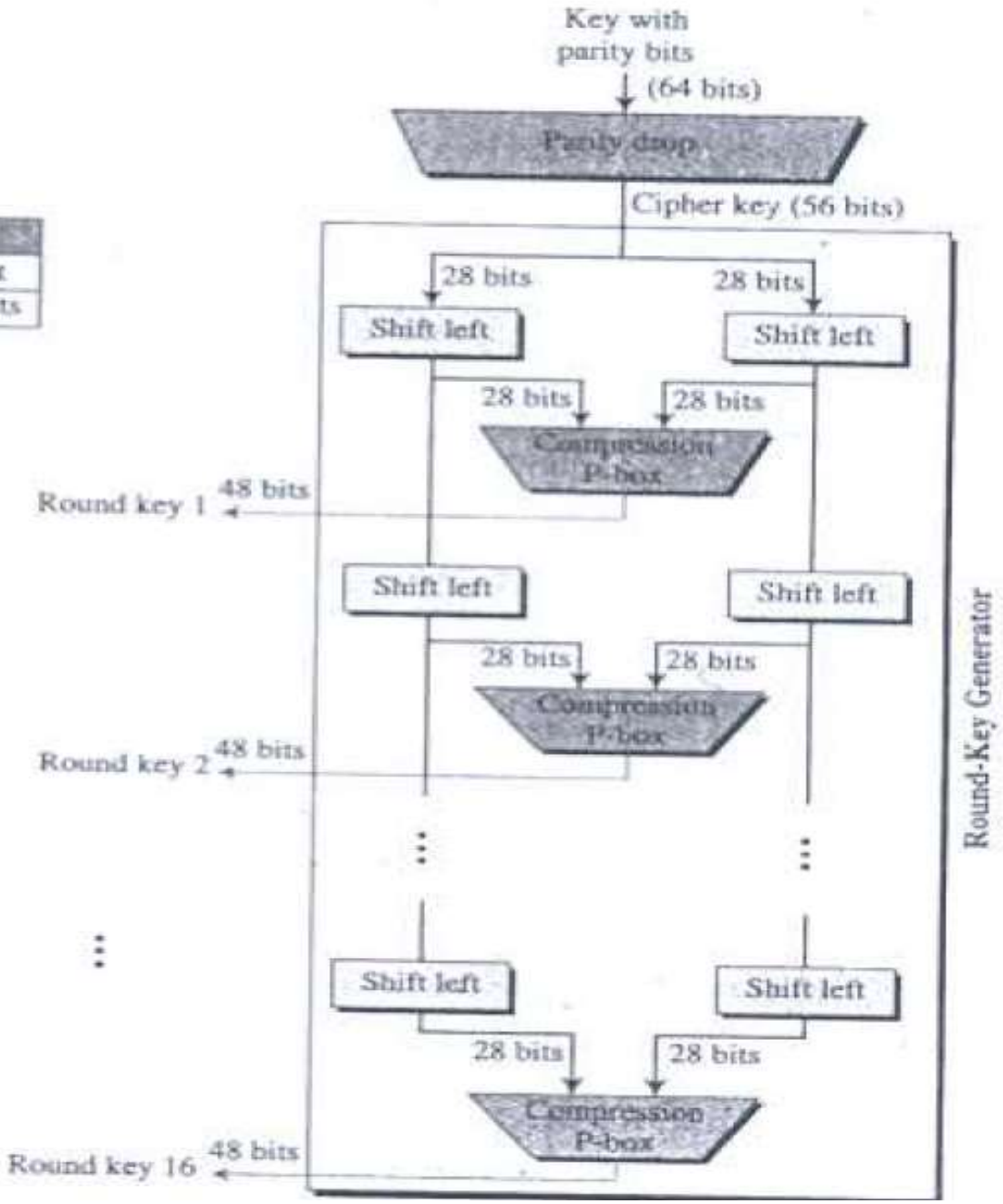


Table 6.12 *Parity-bit drop table*

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

Table 6.13 *Number of bit shifts*

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

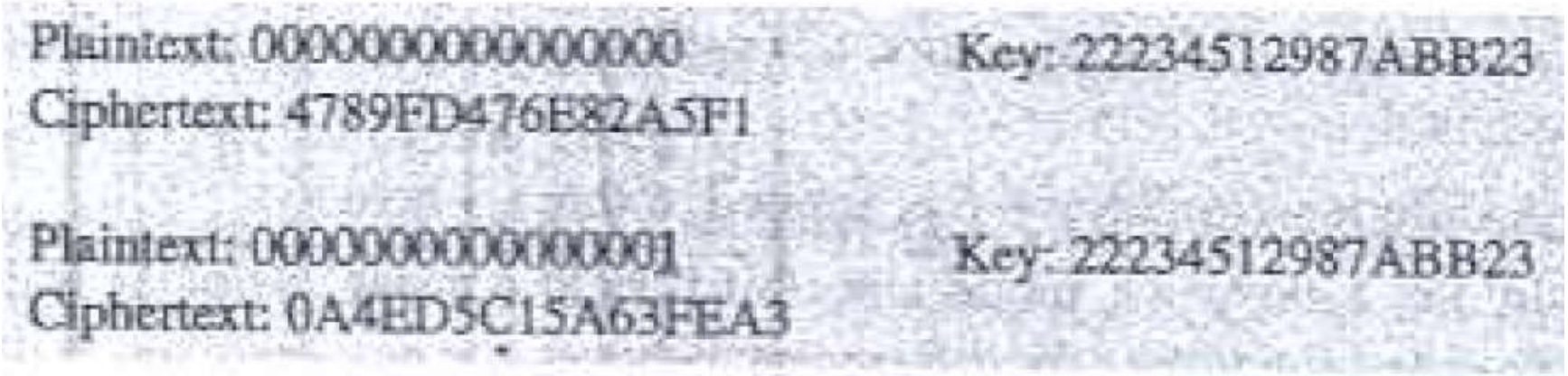
Table 6.14 *Key-compression table*

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Properties

- Two desired properties of a block cipher are the avalanche effect and the completeness.
- Avalanche Effect
  - Avalanche effect means a small change in the plaintext (or key) should create a significant change in the ciphertext. DES has been proved to be strong with regard to this property.
- Completeness effect
  - Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext. The diffusion and confusion produced by P-boxes and S-boxes in DES, show a very strong completeness effect.

Avlanche effect



Cryptanalysis

- **Cryptanalysis** is the study of methods used to break cryptographic systems and decrypt messages without knowing the secret key.
- The main goal of cryptanalysis is to discover weaknesses or vulnerabilities in the algorithms that allow an attacker to bypass the encryption and access the original data.

- **Differential Cryptanalysis** and
- **Linear Cryptanalysis**.

Steps in Differential Cryptanalysis:

- **Choose a Pair of Plaintexts:** The attacker selects two plaintexts that differ by a specific amount, usually by flipping certain bits.
- **Encrypt the Plaintexts:** Both plaintexts are encrypted using the block cipher (without knowing the key).
- **Analyze the Differences:** The attacker analyzes the difference between the ciphertexts and looks for patterns that are common or occur with higher probability.
- **Guess Part of the Key:** Based on the observed patterns, the attacker can make guesses about part of the secret key and narrow down the possibilities.
- **Iterate the Process:** This process is repeated with different pairs of plaintexts to gather more information about the key until the key is fully recovered.

Steps in Linear Cryptanalysis:

- **Collect Plaintext and Ciphertext Pairs:** The attacker collects a large number of known plaintext-ciphertext pairs that have been encrypted using the same secret key.
- **Analyze the Data:** The attacker looks for linear patterns that relate specific bits of the plaintext and ciphertext to bits of the key.
- **Create Linear Approximation Equations:** These linear equations approximate the encryption process. Each equation gives a clue about a small part of the key.
- **Count Occurrences:** The attacker calculates how often these linear approximations hold true for the collected data. If the approximations hold true more than 50% of the time, they are considered useful.
- **Guess the Key:** Based on the useful approximations, the attacker makes educated guesses about the key bits and iteratively refines the guesses.

DES Weaknesses

- Weaknesses in Cipher Design

S-Boxes

1. In S-box 4, the last three output bits can be derived in the same way as the first output bit by complementing some of the input bits.

2. Two specifically chosen inputs to an S-box array can create the same output.

3. It is possible to obtain the same output in a single round by changing bits in only three neighboring S-boxes.

P-boxes

1. It is not clear why the designers of DES used the initial and final permutations; these have no security benefits.

2. In the expansion permutation (inside the function), the first and fourth bits of every 4-bit series are repeated.

- Weakness in the Cipher Key

- Serious weakness of DES is in its key size (56 bits).
- To do a brute-force attack on a given ciphertext block, the adversary needs to check 2^{56} keys.

a. With available technology, it is possible to check one million keys per second.

b. If we can make a computer with one million chips (parallel processing), then we can test the whole key domain in approximately 20 hours.

c. Computer networks can simulate parallel processing. In 1977 a team of researchers used 3500 computers attached to the Internet to find a key challenged by RSA Laboratories in 120 days. The key domain was divided among all of these computers, and each computer was responsible to check the part of the domain.

d. If 3500 networked computers can find the key in 120 days, a secret society with 42,000 members can find the key in 10 days.

Weak Keys

- Four out of 2^{56} possible keys are called weak keys.
- A weak key is the one that, after parity drop operation consists either of all Os, all 1s, or half Os and half 1s

Table 6.18 *Weak keys*

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Disadvantage of using a weak key

- If we encrypt a block with a weak key and subsequently encrypt the result with the same weak key, we get the original block.
- The process creates the same original block if we decrypt the block twice. In other words, each weak key is the inverse of itself Ex (Ex (P)) = P

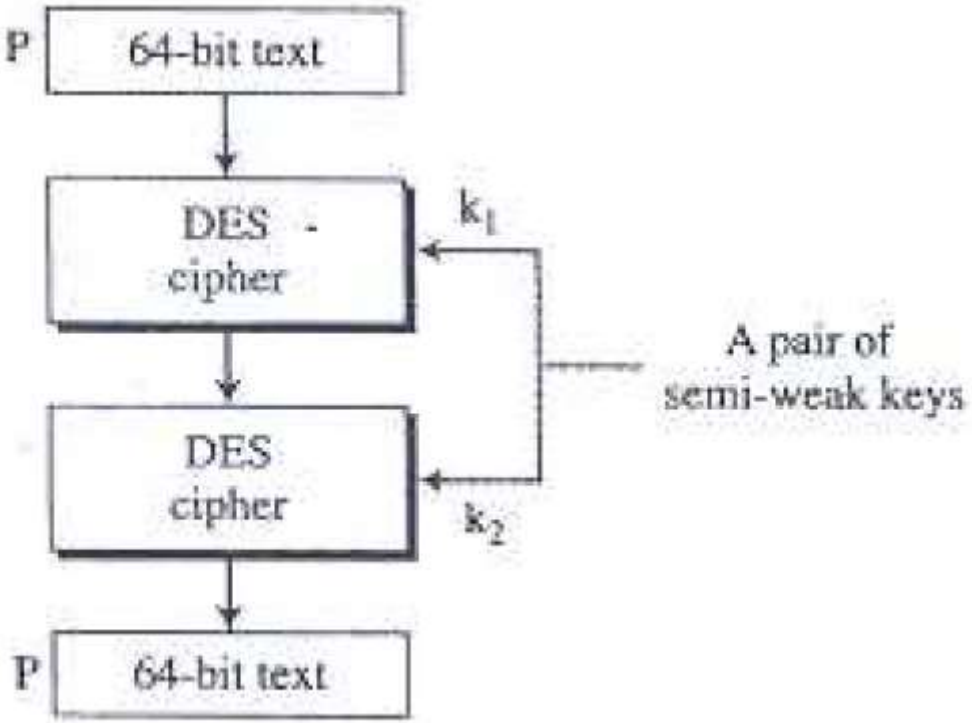
Weak keys should be avoided because the adversary can easily try them on the intercepted ciphertext. If after two decryptions the result is the same, the adversary has found the key.

Semi-weak Keys

- There are six key pairs that are called semi-weak keys.
- A semi-weak key creates only two different round keys and each of them is repeated eight times. In addition, the round keys created from each pair are the same with different order.

First key in the pair	*Second key in the pair*
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE-0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

Round key 1	9153E54319BD	6EAC1ABCE642
Round key 2	6EAC1ABCE642	9153E54319BD
Round key 3	6EAC1ABCE642	9153E54319BD
Round key 4	6EAC1ABCE642	9153E54319BD
Round key 5	6EAC1ABCE642	9153E54319BD
Round key 6	6EAC1ABCE642	9153E54319BD
Round key 7	6EAC1ABCE642	9153E54319BD
Round key 8	6EAC1ABCE642	9153E54319BD
Round key 9	9153E54319BD	6EAC1ABCE642
Round key 10	9153E54319BD	6EAC1ABCE642
Round key 11	9153E54319BD	6EAC1ABCE642
Round key 12	9153E54319BD	6EAC1ABCE642
Round key 13	9153E54319BD	6EAC1ABCE642
Round key 14	9153E54319BD	6EAC1ABCE642
Round key 15	9153E54319BD	6EAC1ABCE642
Round key 16	6EAC1ABCE642	9153E54319BD


P
64-bit text
DES
cipher
k_1
A pair of
semi-weak keys
DES
cipher
k_2
P
64-bit text

Possible Weak Keys

- There are also 48 keys that are called possible weak keys.
- A possible weak key is a key that creates only four distinct round keys; in other words, the sixteen round keys are divided into four groups and each group is made of four equal round keys.

Key Complement

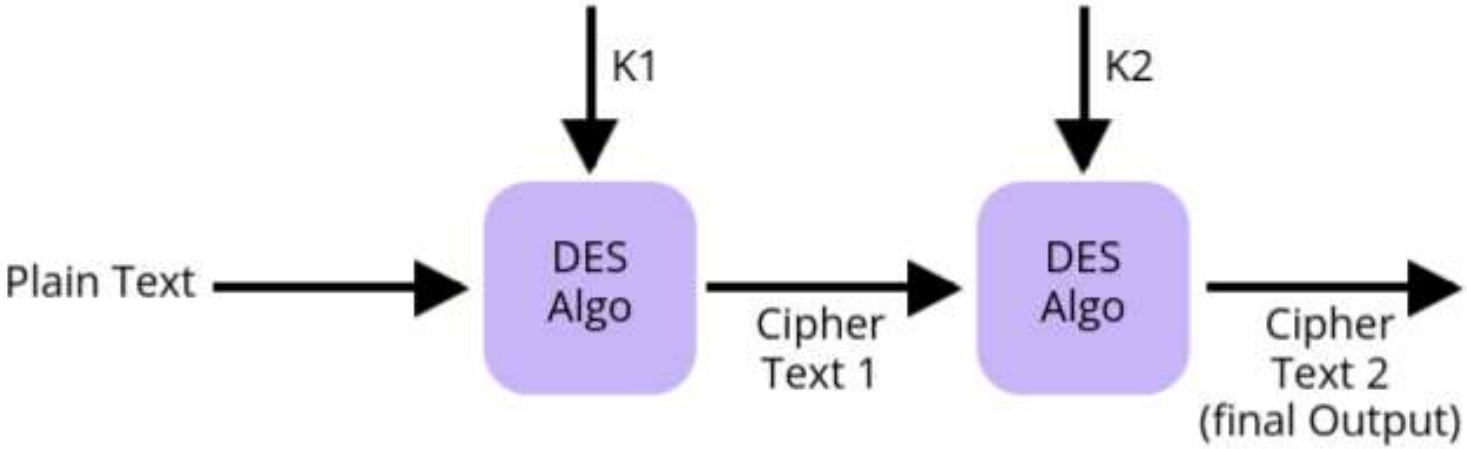
- In the key domain (2^{56}), definitely half of the keys are complement of the other half.
- A key complement can be made by inverting (changing 0 to 1 or 1 to 0) each bit in the key.
- Does a key complement simplify the job of the cryptanalysis?
- Eve can use only half of the possible keys (2^{55}), to perform brute-force attack. This is because
- $C = E(K. P) \rightarrow C' = E(K', P')$
- In other words, if we encrypt the complement of plaintext with the complement of the key, we get the complement of the ciphertext. Eve does not have to test all 2^{56} possible keys.

Multiple DES

- **Multiple DES (Data Encryption Standard)** refers to the use of DES encryption multiple times to increase security. While standard DES uses a single 56-bit key, Multiple DES applies the encryption process more than once, using different keys. This method was introduced to overcome the vulnerabilities of single DES.
- The two most popular implementations of Multiple DES are:
- Double DES (2DES)
- Triple DES (3DES)

Double DES (2DES)

- **Double DES** involves applying DES encryption twice using two different keys. The goal is to improve security by increasing the key length from 56 bits (in DES) to 112 bits.
- **How It Works:**
  - In 2DES, the plaintext is first encrypted with the first key (K1), and the resulting ciphertext is then encrypted again with the second key (K2).
  - Decryption involves reversing the process: first decrypting with K2 and then with K1.



Advantages:

Increases key length from 56 bits to 112 bits, improving resistance to brute force attacks.

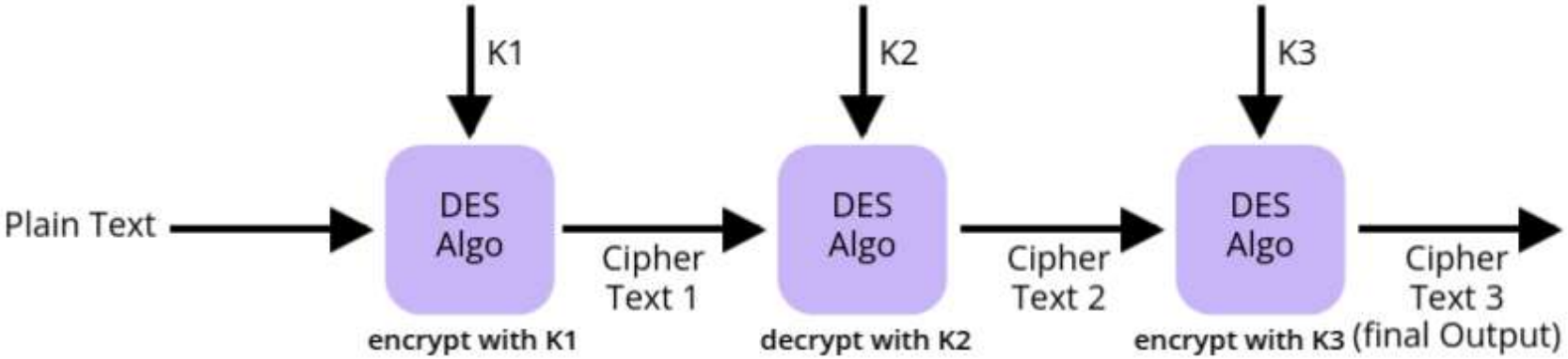
Disadvantages:

Still vulnerable to certain cryptographic attacks, such as the **meet-in-the-middle attack**, which reduces the effective key length to around 57 bits.

Does not provide a significant security improvement over single DES due to the meet-in-the-middle attack.

Triple DES (3DES)

- **Triple DES** (also known as TDES or 3DES) was designed to further enhance security by applying DES three times with two or three different keys. This process creates a much stronger encryption method than Double DES.
- **How It Works:**
  - 3DES encrypts the plaintext with the first key (K1), then decrypts it with the second key (K2), and finally encrypts it again with the third key (K3). This triple application of DES increases security significantly.
  - When using two keys (K1 = K3), the process is still called 3DES, but with only two unique keys.



Advantages:

Increases the key length to 168 bits, making it much more resistant to brute force attacks than DES or 2DES.

Provides significantly stronger security due to the three stages of encryption and decryption.

Disadvantages:

3DES is relatively slow compared to modern block ciphers like AES, as it applies DES three times.

It is still vulnerable to certain attacks but far less so than 2DES.

Advanced Encryption Standard (AES)

- A symmetric key encryption algorithm widely used to secure sensitive data.
- AES is known for its speed, security, and efficiency, making it the preferred standard for modern encryption.
- AES is a block cipher that operates on fixed-size blocks of data (128 bits) and applies several rounds of encryption to transform the plaintext into ciphertext. The number of rounds depends on the key length:
- 10 rounds for 128-bit keys
- 12 rounds for 192-bit keys
- 14 rounds for 256-bit keys

- **AES Key Features**
- **Key Length:** AES supports key sizes of 128, 192, or 256 bits, offering stronger security than DES's 56-bit key.
- **Block Size:** AES encrypts data in blocks of 128 bits, unlike DES, which uses 64-bit blocks.
- **Speed and Efficiency:** AES is much faster than DES, especially on modern hardware, as it is designed for both software and hardware efficiency.
- **Security:** AES provides excellent security against all known cryptographic attacks, making it highly reliable for securing sensitive data.

1. Feistel ciphers- Uses both invertible and non-invertible components

2. Non-Feistel ciphers-Uses only invertible components

•AES encryption algorithm is called Cipher

•The Decryption algorithm called Inverse Cipher is similar but round keys are applied in reverse order

- *Round Keys created by the Key expansion algorithm are always 128 bits , the same size as the plaintext or ciphertext*
- *The number of round keys generated by the key expansion algorithm is always one more than the number of rounds*
- *Number of Round Keys=Nr+1*
- *Round Keys are K_0,K_1,K_2,……..K_{Nr}*

Steps of AES Encryption

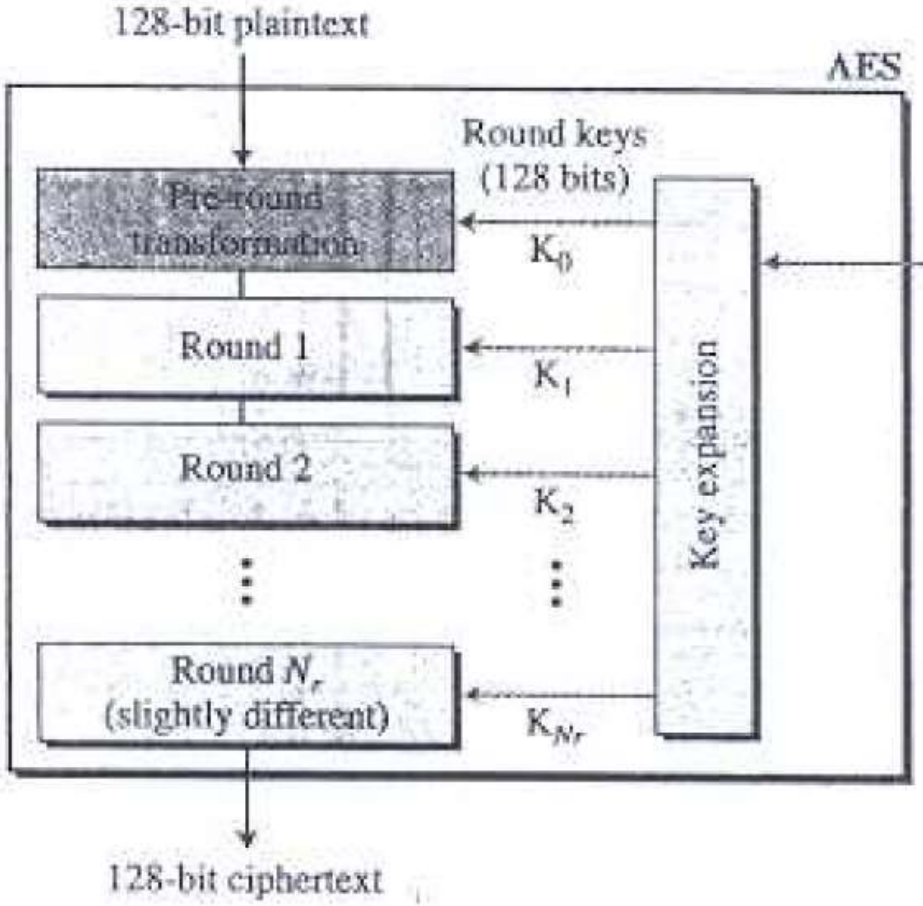
1. Key Expansion: The original key is expanded into multiple round keys, one for each round of encryption.

2. Initial Round (AddRoundKey): The initial key is XORed with the plaintext block.

3. Rounds: Each round involves the following steps:

- **SubBytes:** A non-linear substitution step where each byte of the block is replaced using a substitution table (S-box).
- **ShiftRows:** The rows of the block are shifted cyclically to the left, increasing diffusion.
- **MixColumns:** Columns of the block are mixed together using linear algebra to enhance security (not used in the final round).
- **AddRoundKey:** The current block is XORed with the round key generated from the original key.

4. Final Round: The last round omits the MixColumns step and consists of SubBytes, ShiftRows, and AddRoundKey operations.



Cipher key
(128, 192, or 256 bits)

N_r	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Data Units.

Data units used in AES

AES uses 5 units of measurement to refer to data:

- ***Bits-Smallest and atomic unit***

Other units can be expressed in terms of smaller ones

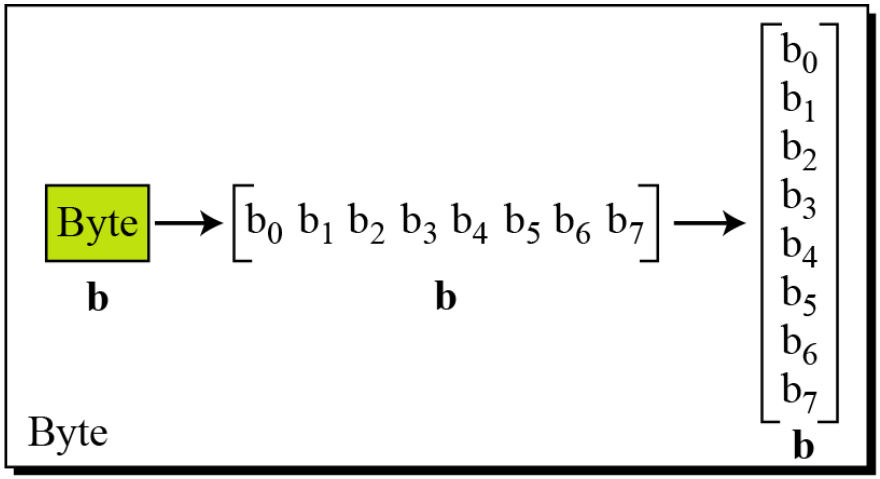
- ***Bytes***
- ***Words***
- ***Blocks***
- ***State***

Data Units.

Data units used in AES

Bytes-

- ***Group of 8 bits,***
- ***Row Matrix (1X8) or***
- ***Column Matrix (8X1),***
- ***Row Matrix –bits are inserted from into the matrix left to right,***
- ***Column Matrix-bits are inserted into the matrix from top to bottom***
- ***Represented with lowercase bold letter "b"***

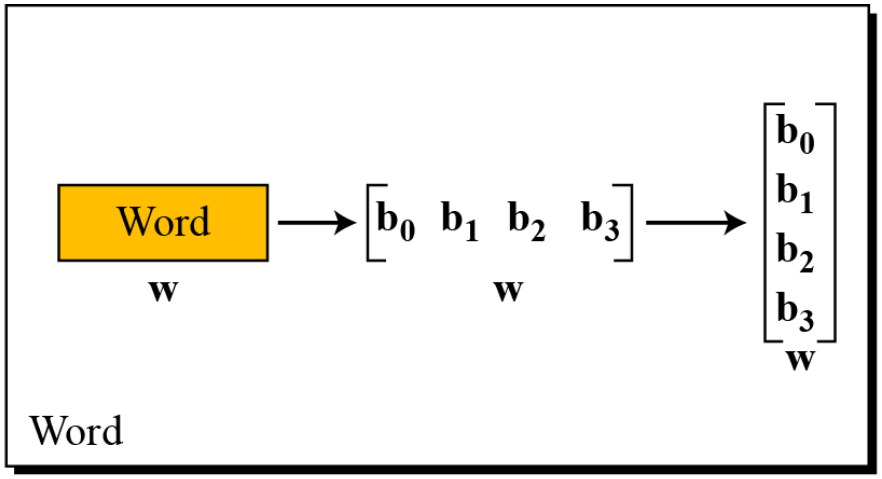


Data Units.

Data units used in AES

Words

- ***Group of 32 bits***
- ***Row Matrix of 4 Bytes***
- ***Column Matrix of 4 Bytes***
- ***Row Matrix-Bytes are inserted from into the matrix left to right,***
- ***Column Matrix-Bytes are inserted into the matrix from top to bottom***

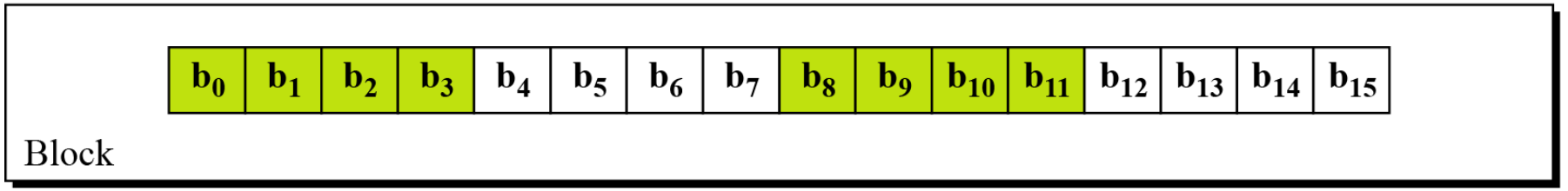


Data Units.

Data units used in AES

Block

- ***Group of 128 bits***
- ***Row Matrix of 16 bytes***
- ***AES encrypts and decrypts data block***



Data Units.

Data units used in AES

State

- ***AES uses several rounds***
- ***Each Round is made of several stages***
- ***Data block is transformed from one stage to other***
- ***At the beginning and end of the cipher, AES uses the term data block***
- ***Before and after each stage, the data block is referred to as state***
- ***Represented as Bold letter S***
- ***Bold Letter T=Temporary State***

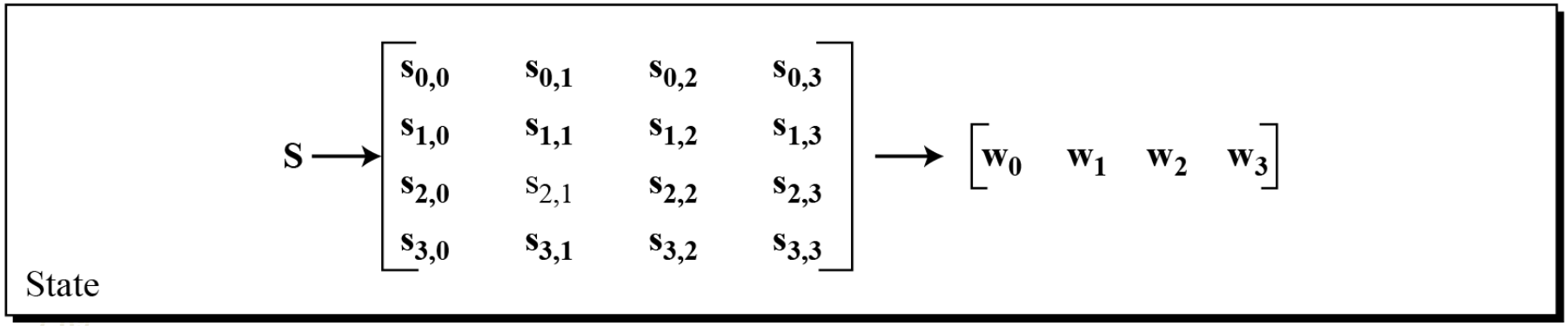
Data Units.

Data units used in AES

State

- ***Like Blocks, States are made of 16 bytes***
- ***Normally treated as matrix of 4X4 bytes***
- ***Each element of a state is referred to as $S_{r,c}$ where***
- ***r(0 to3) defines the row and***
- ***c (0 to 3) defines the column***
- ***Occasionally, a state is treated as a row matrix (1X4) of words***

where words are column matrix

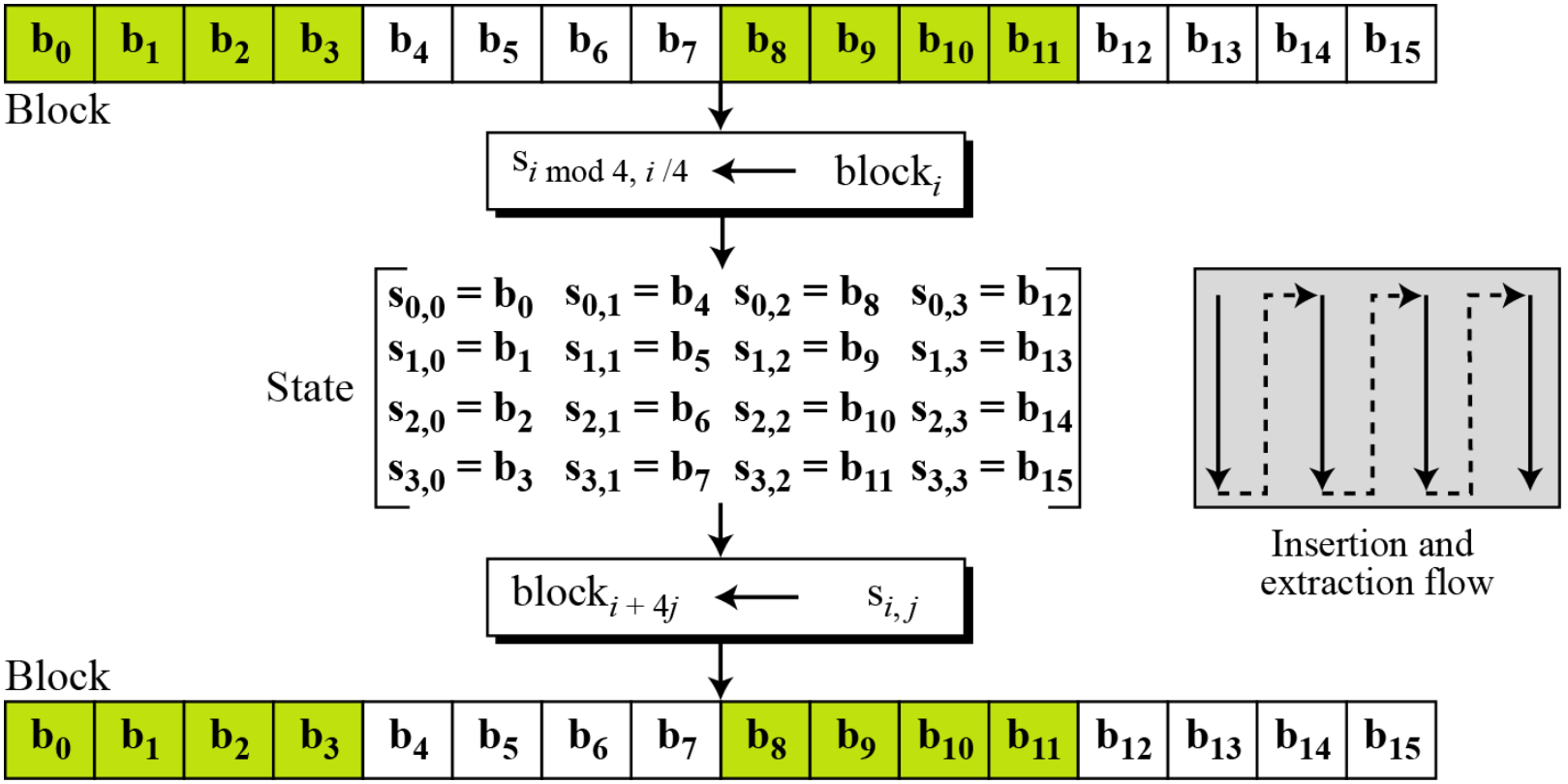


Continue

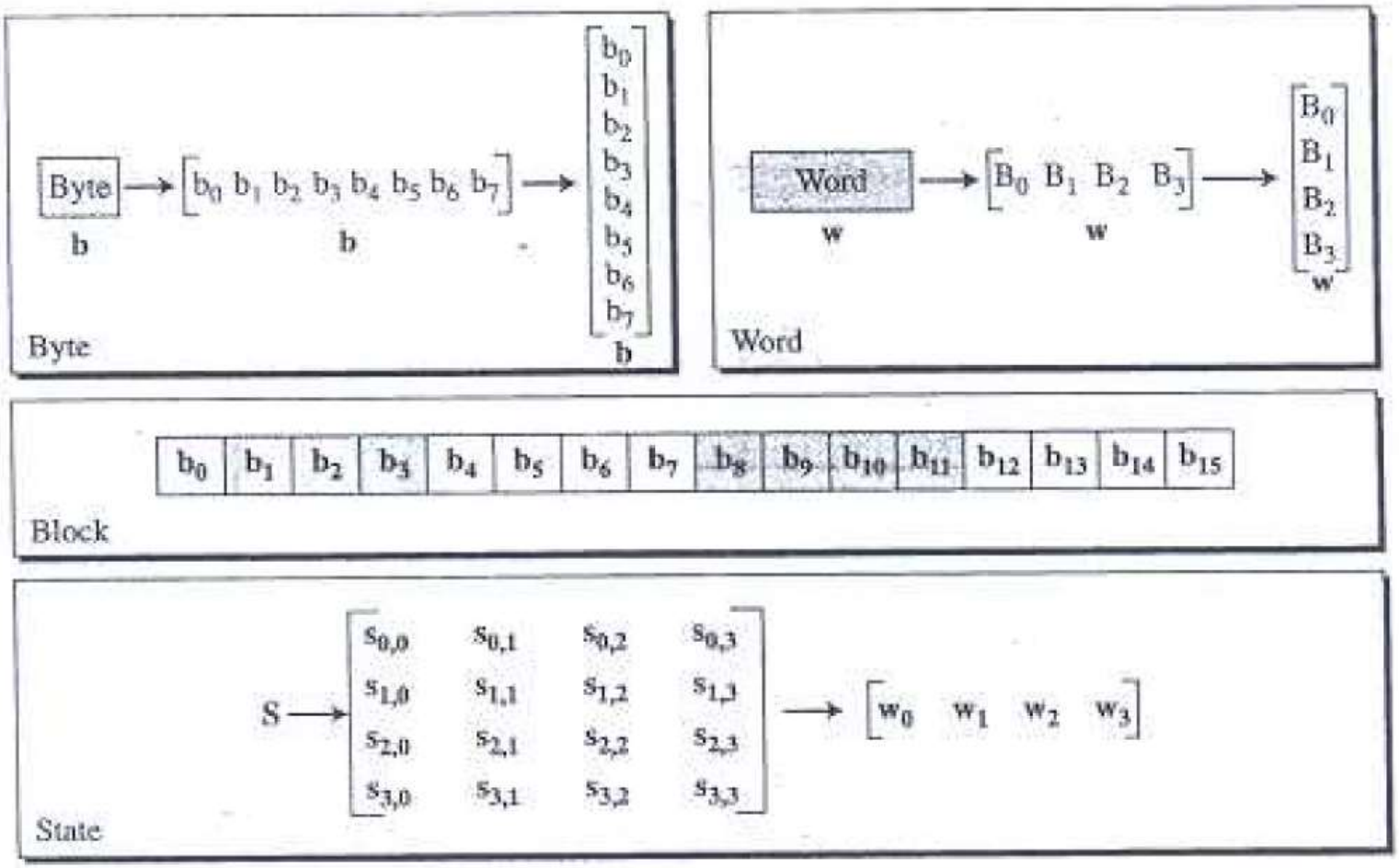
Figure 7.3 *Block-to-state and state-to-block transformation*

At the beginning of the cipher, the bytes are inserted into a state, column by column and in each column from top to bottom

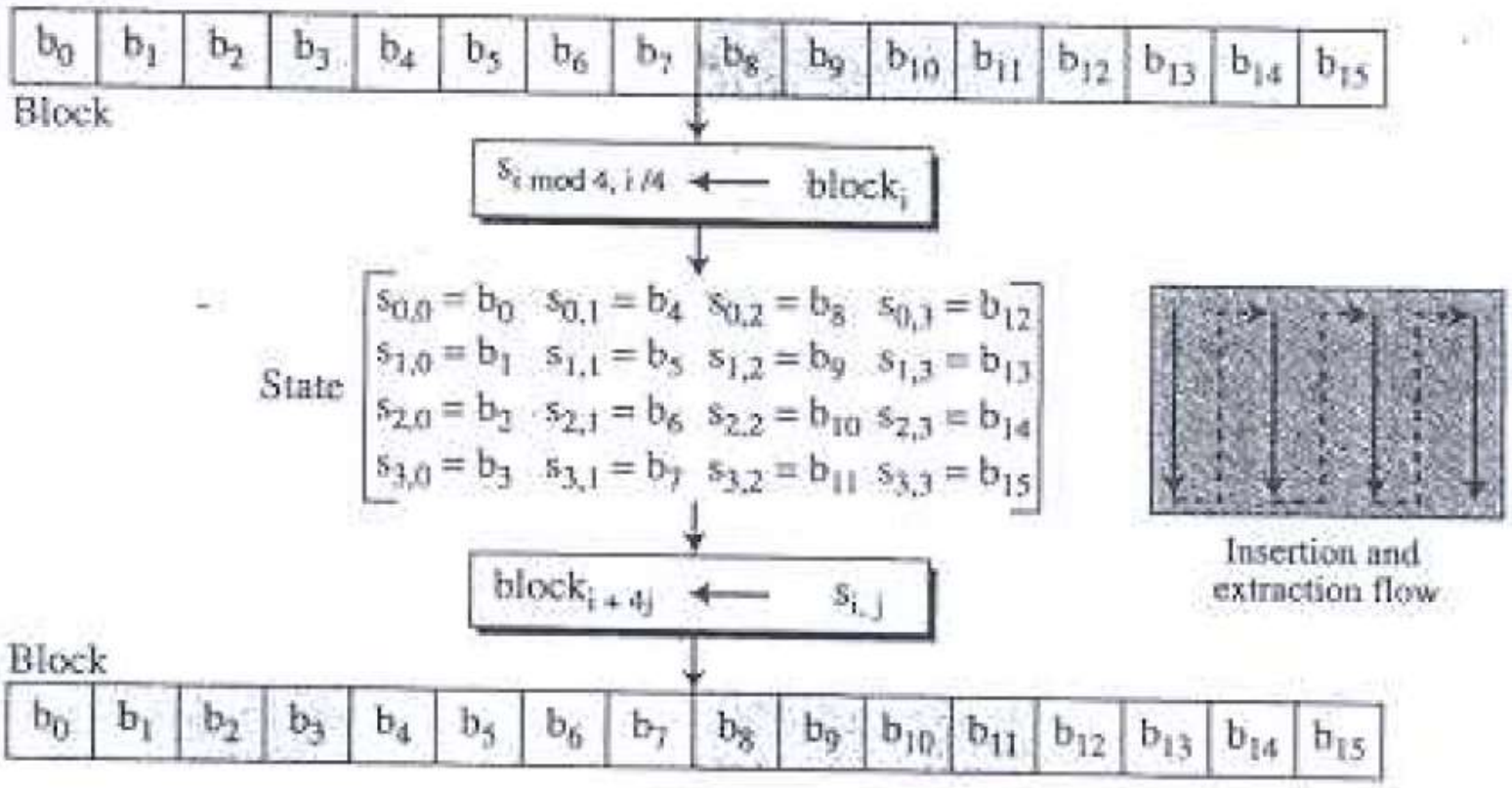
At the end of cipher, the bytes in data block are extracted in the same way



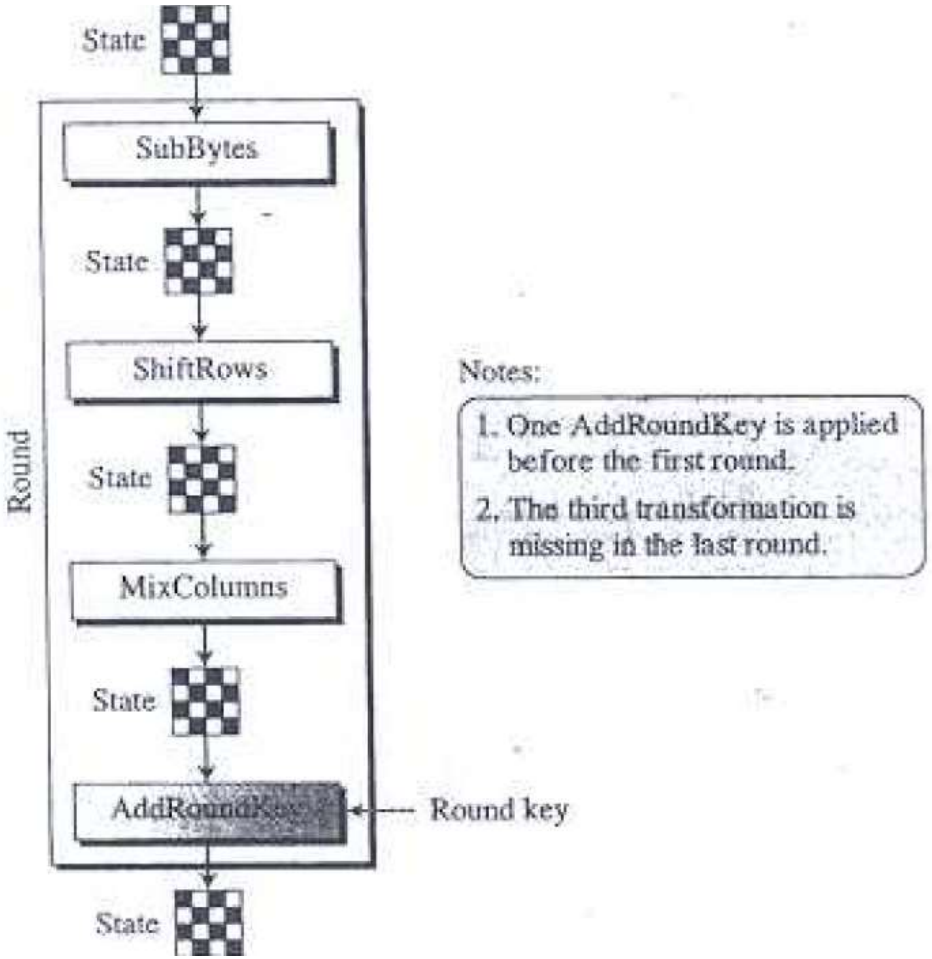
Data Units



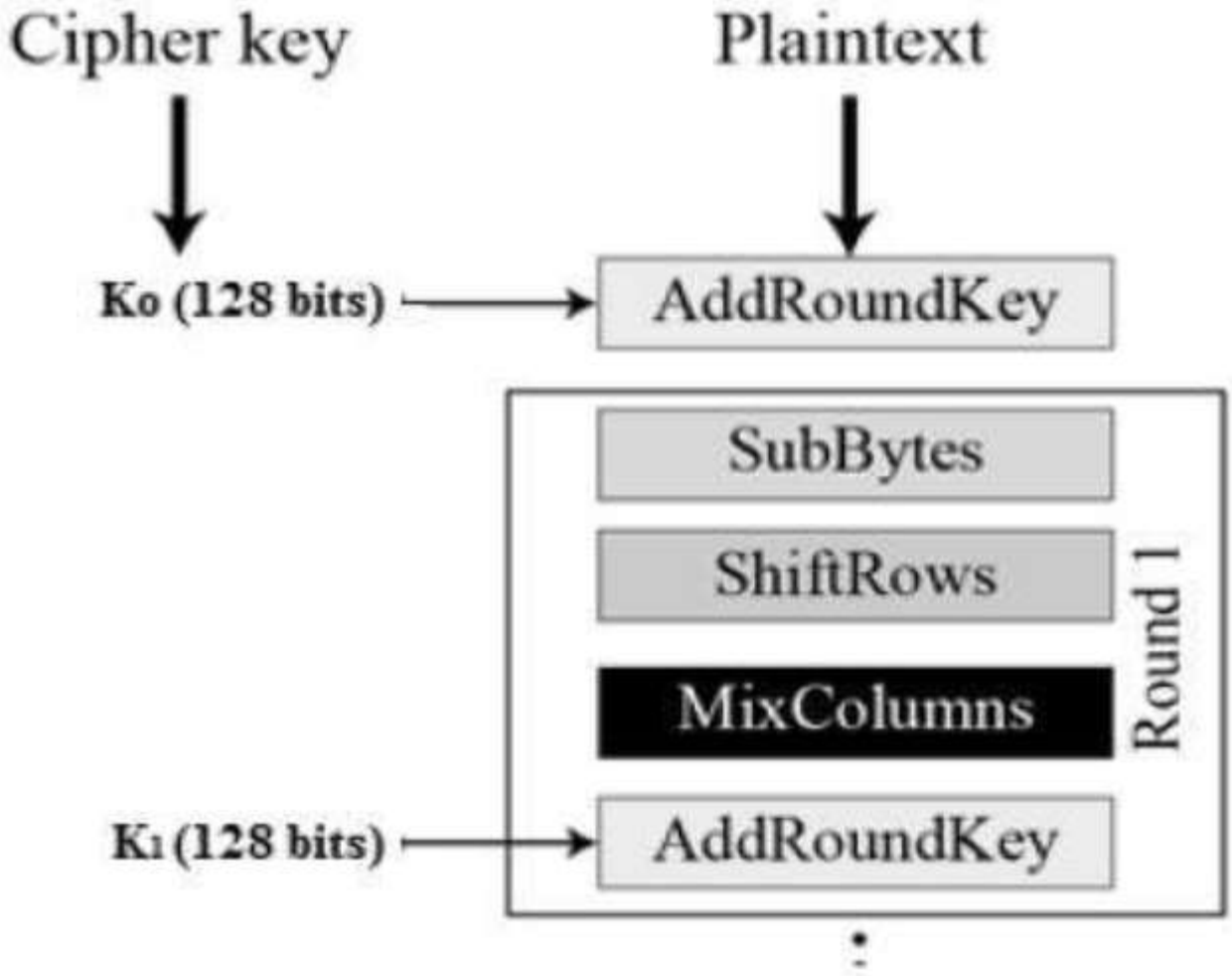
- AES encrypts and decrypts data blocks. A block in AES is a group of 128 bits. However, a block can be represented as a row matrix of 16 bytes.
- AES uses several rounds in which each round is made of several stages. Data block is transformed from one stage to another. At the beginning and end of the cipher, AES uses the term data block; before and after each stage, the data block is referred to as a state.


b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
Block
s i mod 4, i /4 ← block i
State
s0,0 = b0 s0,1 = b4 s0,2 = b8 s0,3 = b12
s1,0 = b1 s1,1 = b5 s1,2 = b9 s1,3 = b13
s2,0 = b2 s2,1 = b6 s2,2 = b10 s2,3 = b14
s3,0 = b3 s3,1 = b7 s3,2 = b11 s3,3 = b15
block i + 4j ← s i, j
Block
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
Insertion and extraction flow

Structure of each round



Structure of each round



Byte Substitution (SubBytes)

- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

- Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row. Shift is carried out as follows −
- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

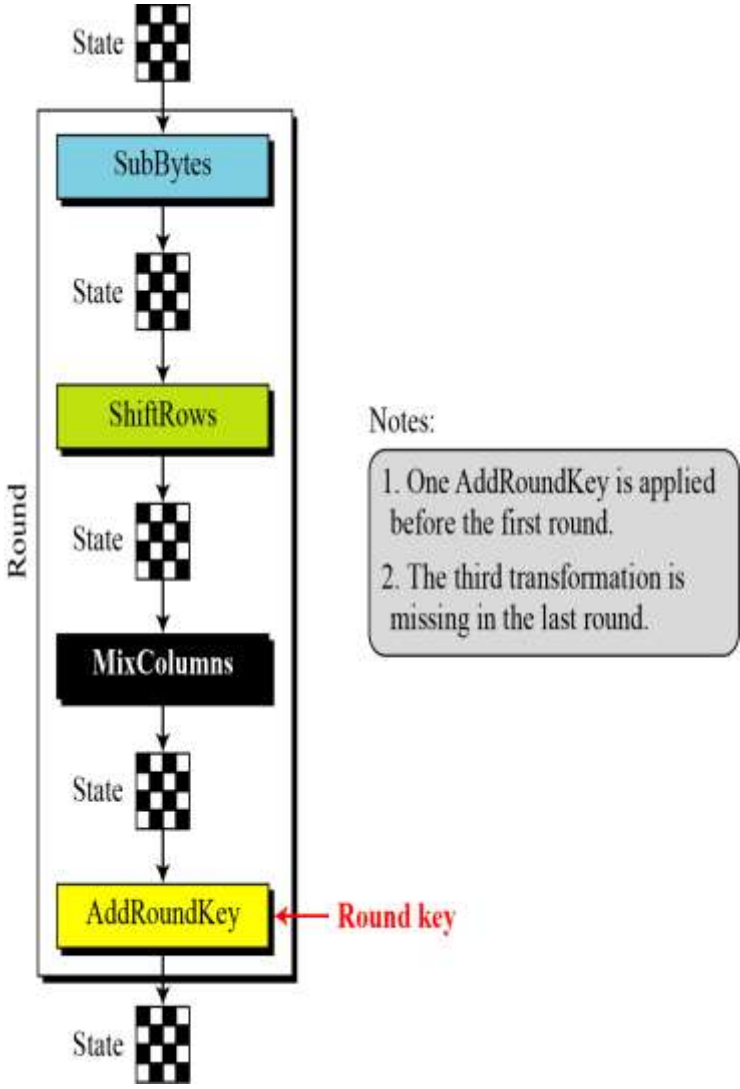
- Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes . Uses Hill Cipher

Addroundkey

- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

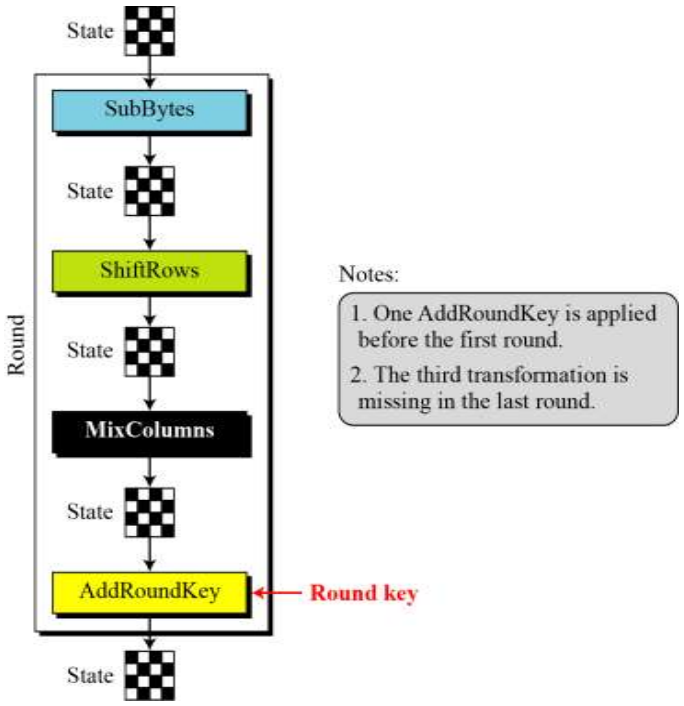
Structure of Each Round

- *Structure of each round at the encryption site*
- *Each round except the last uses 4 transformations that are invertible*
- *The last round has only three transformations*
- *Each transformation takes a state and creates another state to be used for the next transformation or the next round*



Structure of Each Round

- *The pre-round section uses only one transformation(AddRoundKey)*
- *In the last round, MixColumns transformation is missing*
- *At the Decryption site, The inverse transformations are used –*
  - *InvSubByte,*
  - *InvShiftRows,*
  - *InvMixColumns*
  - *AddRoundKey(self invertible)*



TRANSFORMATIONS

To provide security, AES uses four types of transformations: substitution, permutation, mixing, and key-adding.

Topics discussed in this section:

Substitution

- Like DES , AES also uses Substitution
- The Mechanism is different
- First Substitution is done for every byte
- Only one table is used for transformation of every byte, i.e., if two bytes are same, transformation is also same.
- Transformation is defined by a lookup table or mathematical calculation

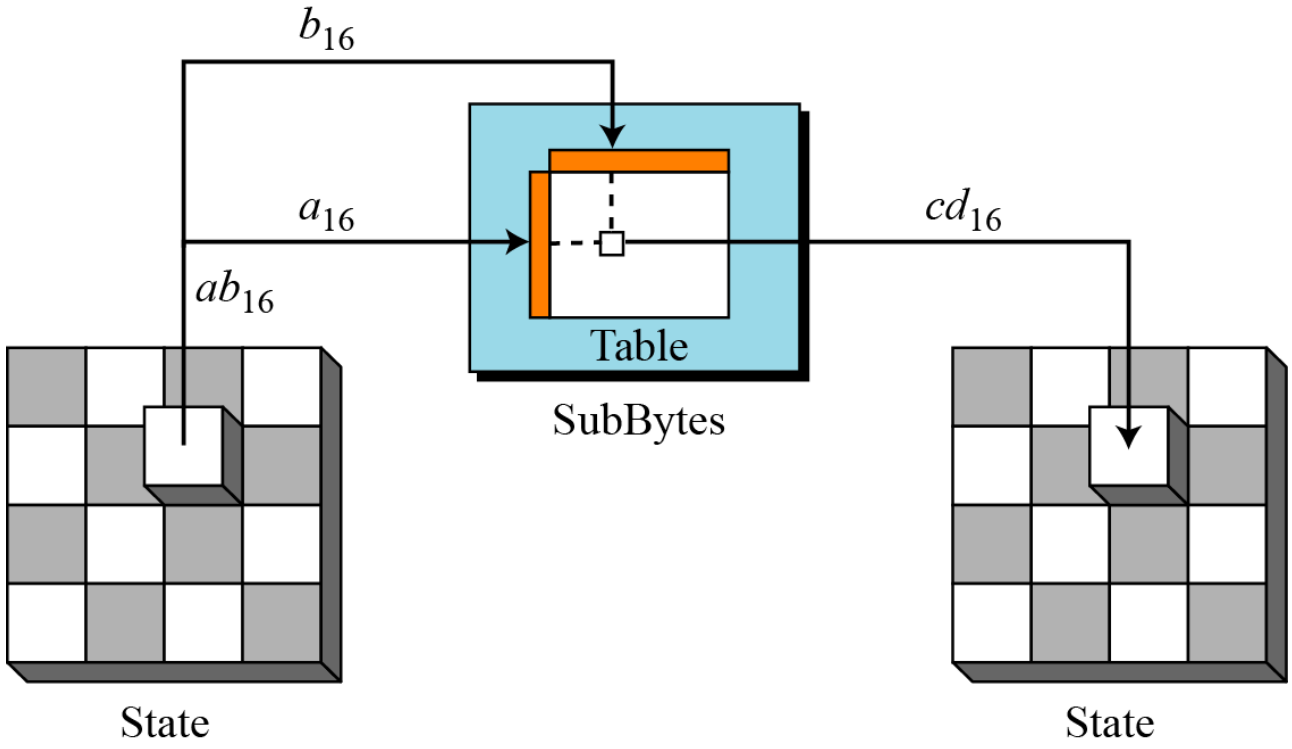
Substitution

AES uses two invertible transformations.

SubBytes

- *The first transformation, SubBytes, is used at the encryption site.*
- *To substitute a byte, we interpret the byte as two hexadecimal digits.*
- *Left digit defines row while right digit defines column in substitution table.*
- *The two Hexadecimal digits at the junction of the row and the column are the new byte*

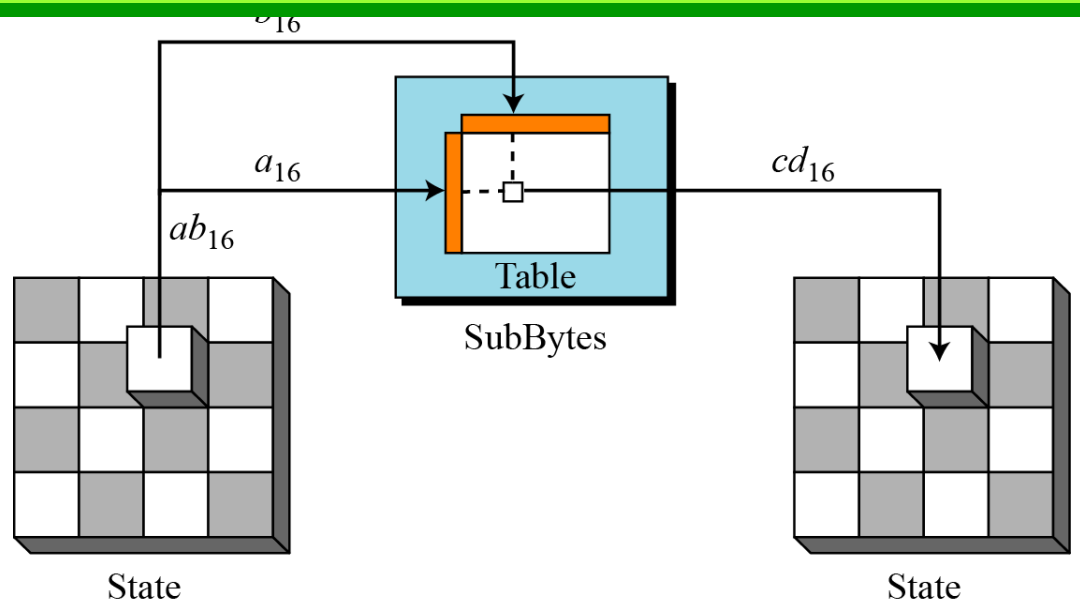
SubBytes transformation



SubBytes transformation

- *State is treated as 4X4 matrix of bytes*
- *Transformation is done one byte at a time*
- *The content of each byte is changed but the arrangement of the bytes in the matrix remains the same*
- *Each byte is transformed independently*

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



- ***The Substitution Table (S Box) for SubBytes transformation***
- ***Provides confusion effect***
- ***Two bytes $5A_{16}$ and $5B_{16}$ which differ only in one bit are transformed to BE_{16} and 39_{16} which differ in four bits***

Table 7.1 *SubBytes transformation table*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
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

Table 7.1 *SubBytes transformation table (continued)*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
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	CB	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

InvSubBytes

- ***Inverse of SubBytes***

Table 7.2 *InvSubBytes transformation table*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
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

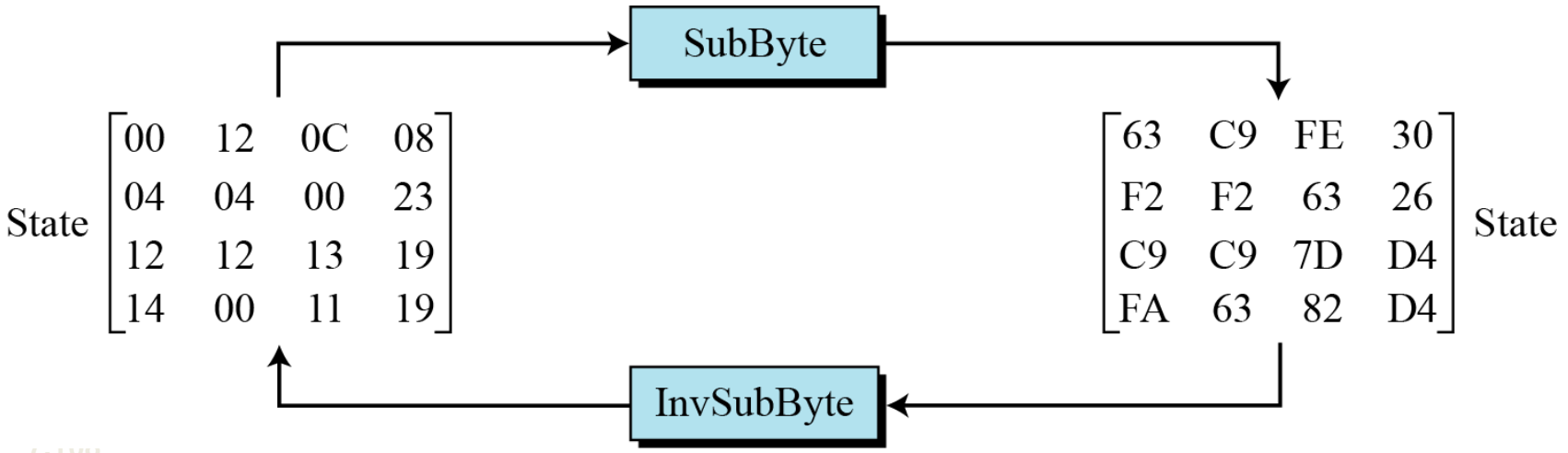
InvSubBytes (Continued)

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

Example 7.2

Figure 7.7 shows how a state is transformed using the SubBytes transformation. The figure also shows that the InvSubBytes transformation creates the original one. Note that if the two bytes have the same values, their transformation is also the same.

SubBytes transformation for Example 7.2



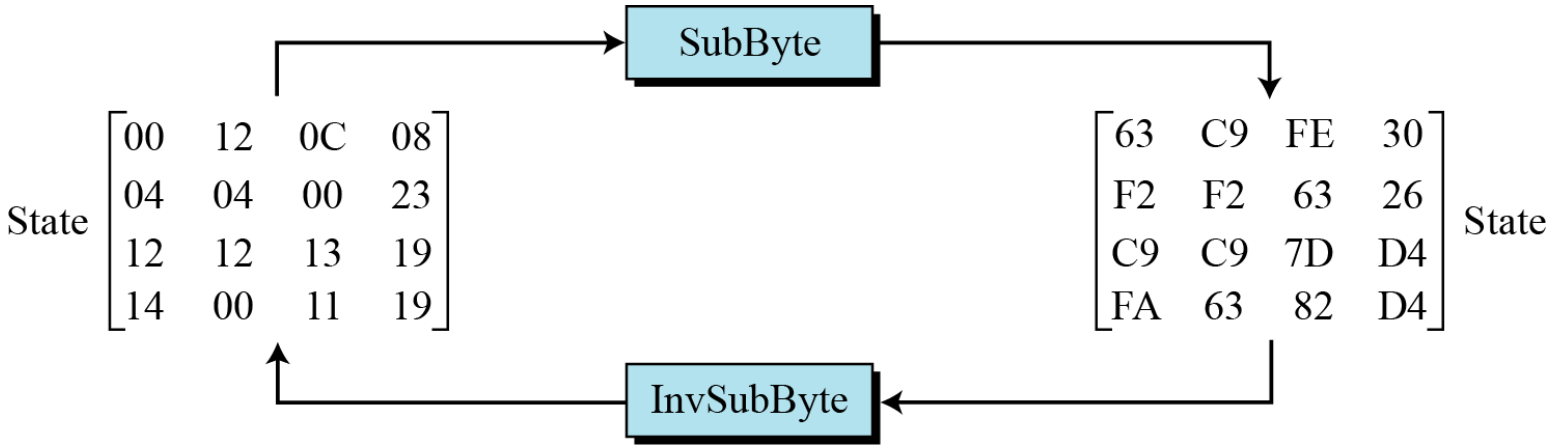
Example 7.2

Table 7.1 *SubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

Table 7.1 *SubBytes transformation table (continued)*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	CB	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



SubBytes transformation for Example 7.2

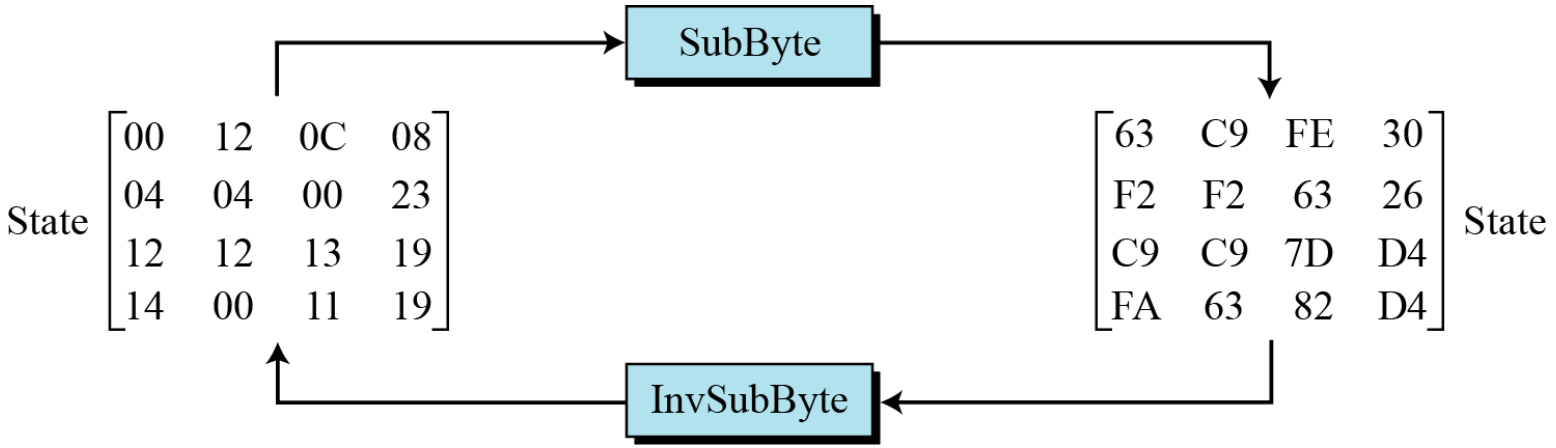
Example 7.2

Table 7.2 *InvSubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

Table 7.2 *InvSubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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



SubBytes transformation

Permutation

- ***Another transformation found in a round is shifting, which permutes the bytes.***

- ***In DES, Permutation is done at the bit level,***
- ***Shifting transformation in AES is done at the byte level***

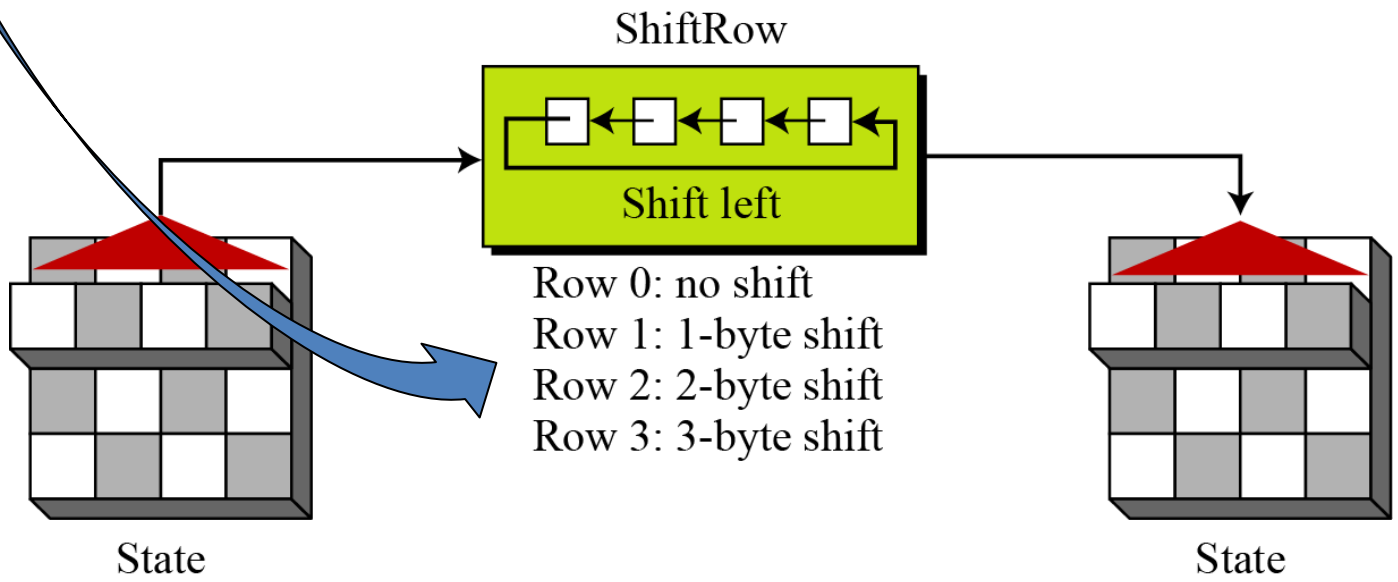
- ***The order of the bits in the byte is not changed***

7.2.2 Permutation

ShiftRows

- *In the encryption, the transformation is called ShiftRows.*
- *Shifting is to the left*
- *The no of shifts depends on the row number(0,1,2,3) of the state matrix*
- *Shift row transformation operates one row at a time*

ShiftRows transformation



InvShiftRows

- ***In the decryption, the transformation is called InvShiftRows and the shifting is to the right.***

- ***The number of shifts is the same as the row number (0,1,2 and 3) of the state matrix***

- ***ShiftRows and InvShiftRows transformations are inverses of each other***

Algorithm for ShiftRows-

- ***Function called shift row that shifts byte in a single row***
- ***This function is called three times***
- ***Function copies the row into a temporary row matrix t and then shift row***

Algorithm 7.2 *Pseudocode for ShiftRows transformation*

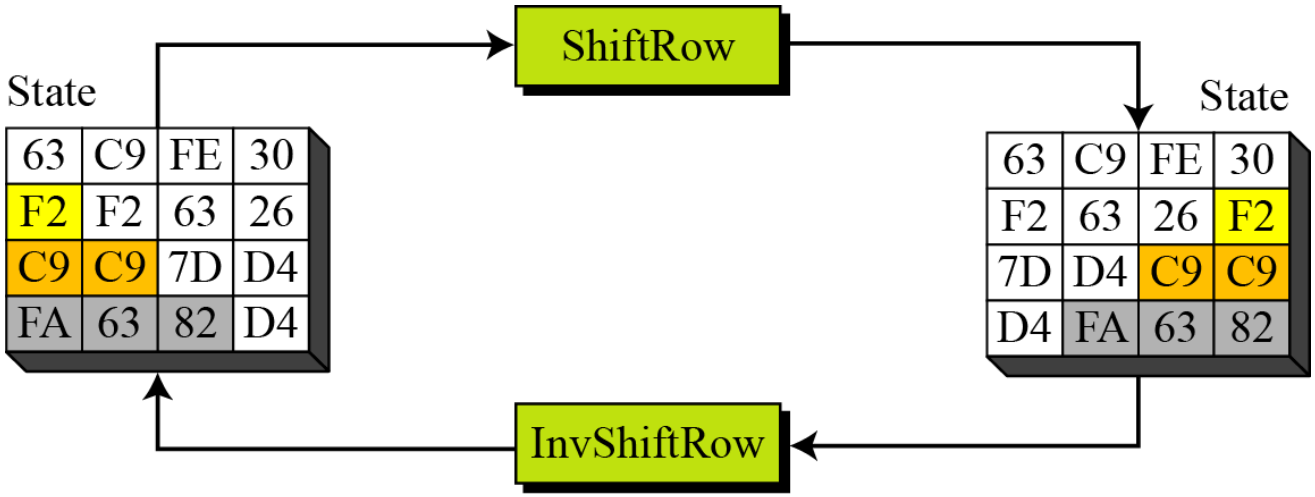
```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                    // s_r is the rth row
}
```

```
shiftrow (row, n)                            // n is the number of bytes to be shifted
{
   CopyRow (row, t)                          // t is a temporary row
   for (c = 0 to 3)
        row_((c – n) mod 4) ← t_c
}
```

Example 7.4

Figure 7.10 shows how a state is transformed using ShiftRows transformation. The figure also shows that InvShiftRows transformation creates the original state.

Figure 7.10 ***ShiftRows transformation in Example 7.4***

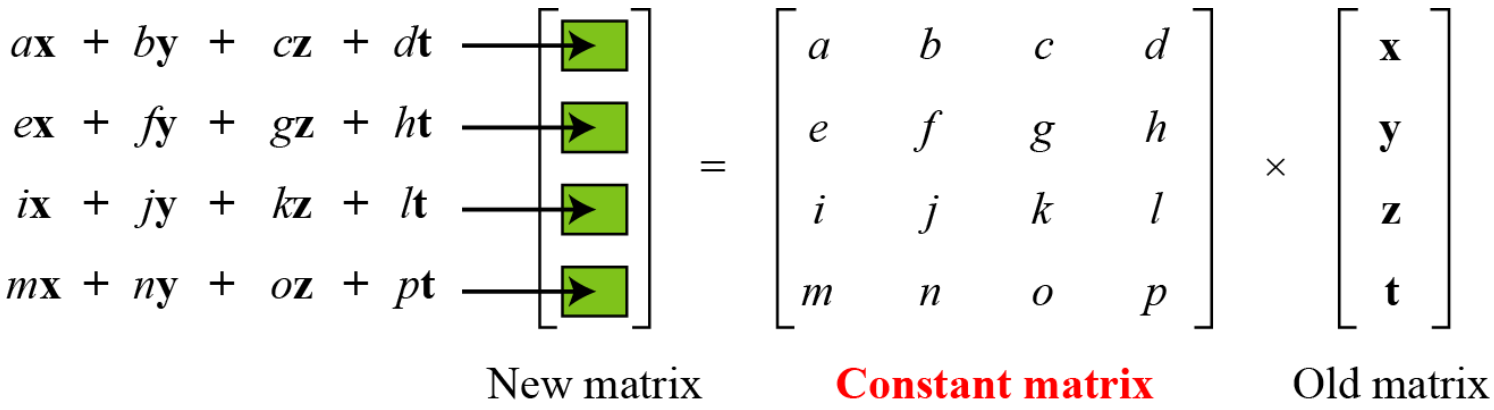


Mixing

We need an interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes.

We need to mix bytes to provide diffusion at the bit level.

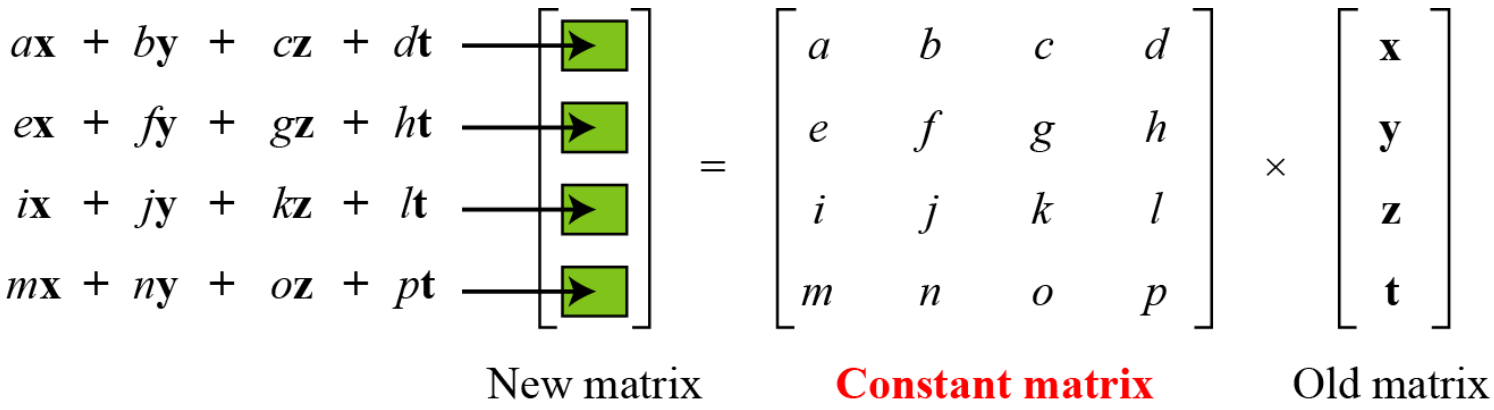
Figure 7.11 ***Mixing bytes using matrix multiplication***



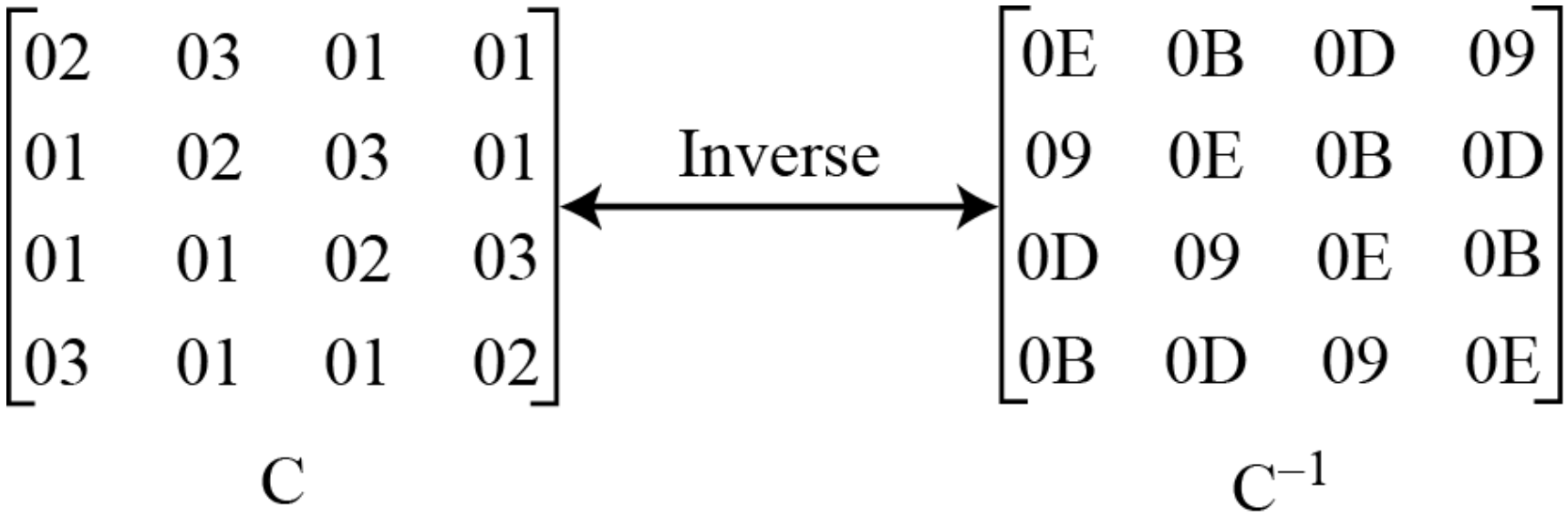
Mixing

- *Takes 4 bytes at a time, combining them to recreate four new bytes*
- *Each new byte is different, even if all 4 bytes are the same*
- *Multiplies each byte with a different constant and mixes them*

Figure 7.11 ***Mixing bytes using matrix multiplication***



- **AES defines a transformation called Mix columns to acheive this goal**
- **There is also an inverse transformation called InvMixColumns**



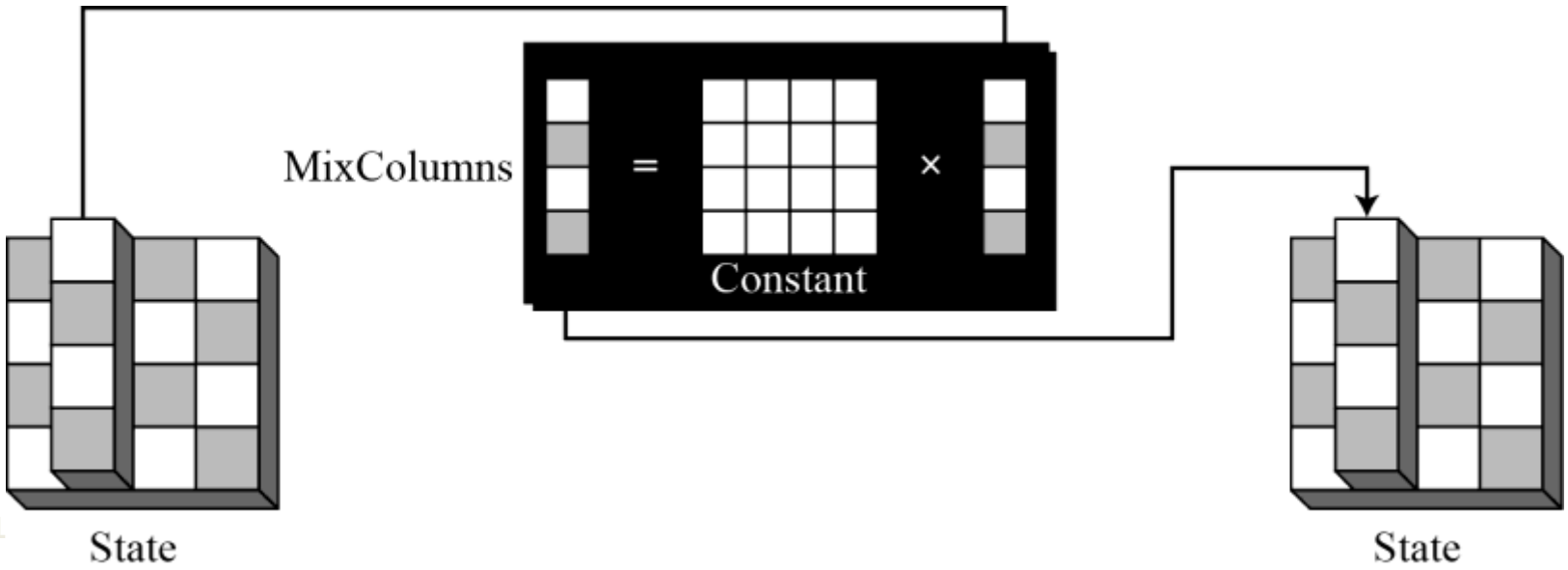
Constant matrices used by MixColumns and InvMixColumns

MixColumns

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

Matrix multiplication of a state column by a constant square matrix

Figure 7.13 *MixColumns transformation*

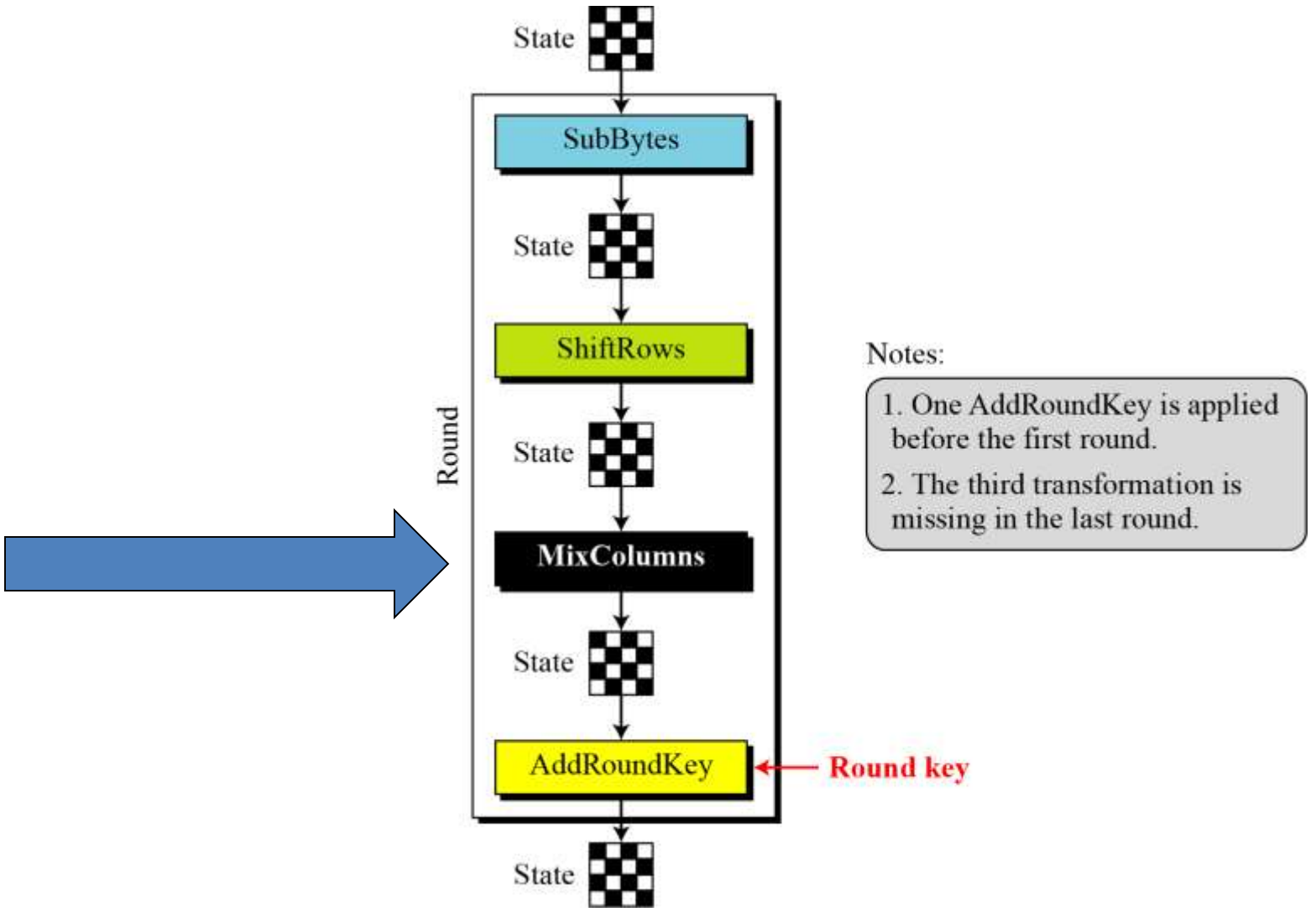


InvMixColumns

The InvMixColumns transformation is basically the same as the MixColumns transformation.

Note

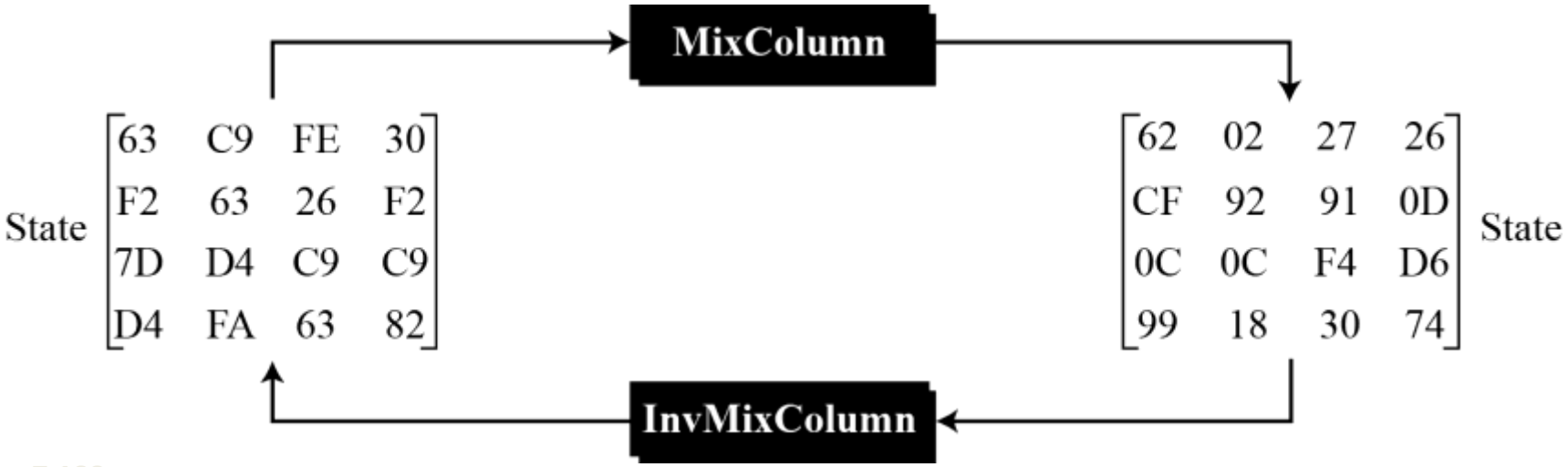
The two column matrices are inverses of each other,
Thus, The MixColumns and InvMixColumns transformations are inverses of each other.


State
SubBytes
State
ShiftRows
State
MixColumns
State
AddRoundKey
State
Round
Round key
Notes:
1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

Example 7.5

Figure 7.14 shows how a state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.

Figure 7.14 *The MixColumns transformation in Example 7.5*



Key Adding

- ***Most important transformation***
- ***Its the one that includes the cipher key***
- ***If the cipher key is not added to the state at each round, it is very easy for the adversary to find the plaintext, given the ciphertext.***
- ***The cipher key is the only secret between Alice and Bob***

Key Adding

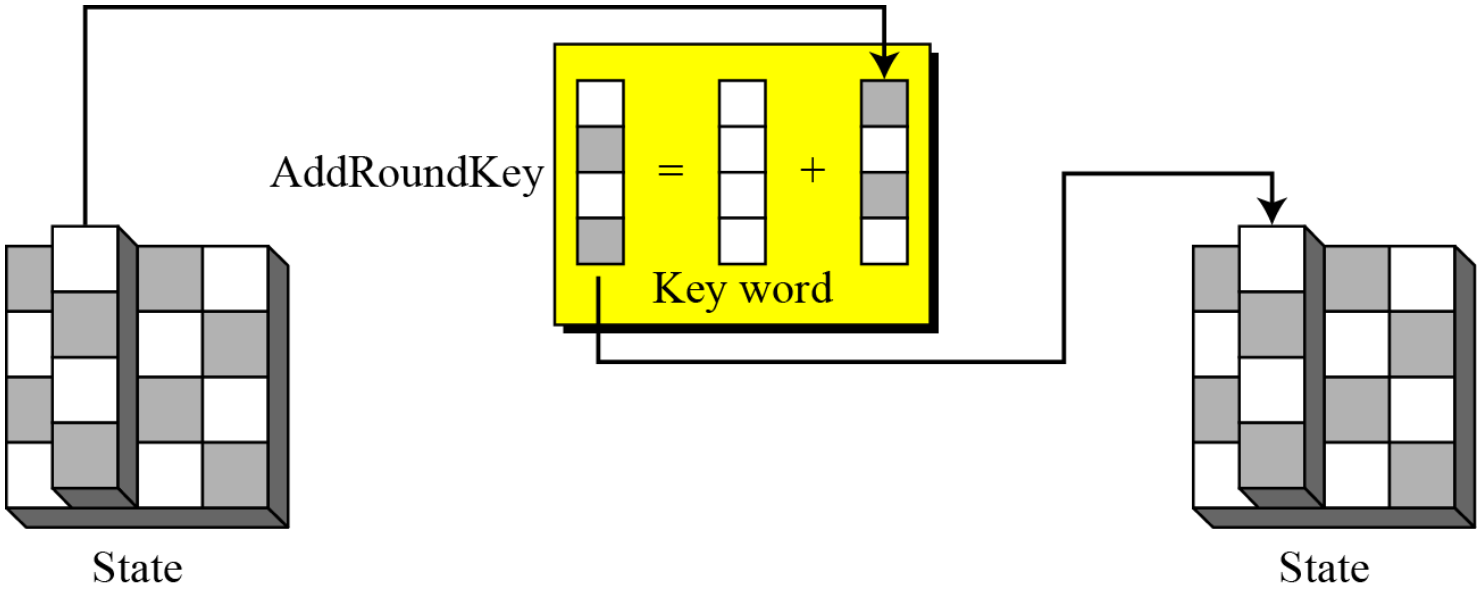
AddRoundKey

- *Each Round key is 128 bits long*
- *Treated as Four 32 bit words*
- *For adding the key to the state , each word is considered as a column matrix*
- *AddRoundKey proceeds one column at a time.*
- *AddRoundKey adds a round key word with each state column matrix;*
- *The operation in AddRoundKey is matrix addition.*

Note

The AddRoundKey transformation is the inverse of itself.

Figure 7.15 *AddRoundKey transformation*



Algorithm-

- ***XORing of each column of the state with the corresponding keyword***
- ***Cipherkey is expanded into a set of keywords***
- ***S_c and w_{round} are 4X1 column matrices***
- ***XORing of two column matrices , each of 4 bytes***

Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

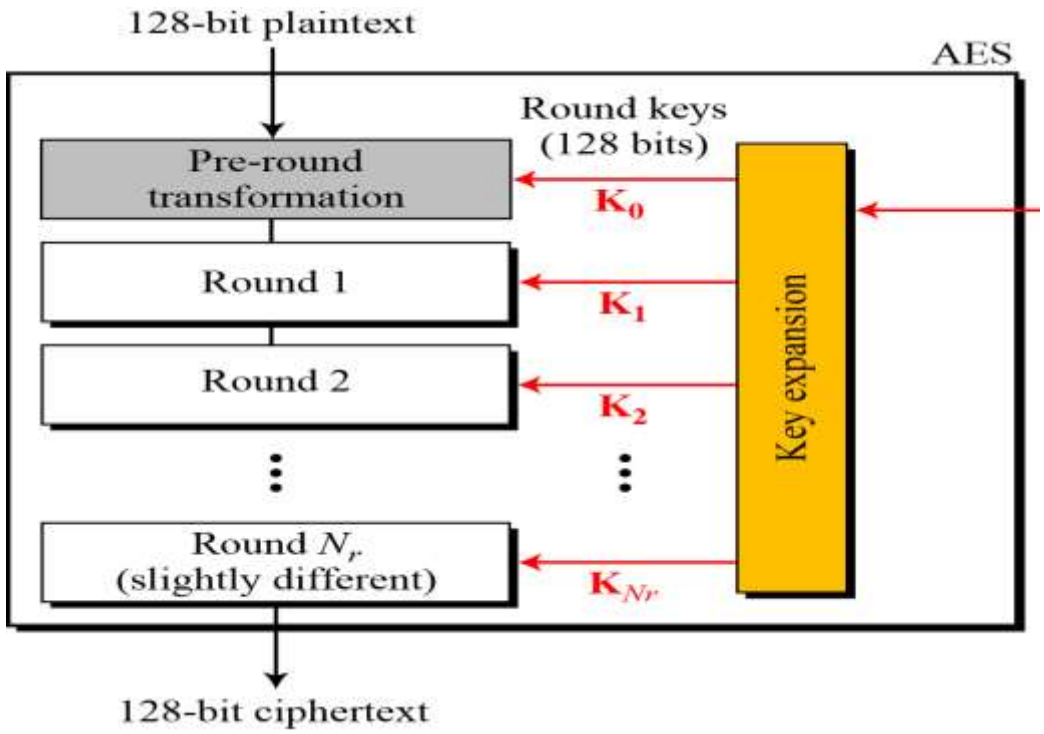
7-3 KEY EXPANSION

To create round keys for each round, AES uses a key-expansion process. If the number of rounds is N_r , the key-expansion routine creates N_r + 1 128-bit round keys from one single 128-bit cipher key.

Topics discussed in this section:

7-3 KEY EXPANSION

The 1st Round key is used for pre-round transformation
The remaining round keys are used for the last transformation(AddRoundKey) at the end of each round



7-3 KEY EXPANSION

The Key expansion routine creates round keys word by word, where word is an array of four bytes
The routine creates 4X(Nr+1) words called
w0,w1,w2……...w4

In AES-128 version with 10 rounds=>11X4=44 words
In AES-192 version with 12 rounds=>13X4=52 words
In AES-256 version with 14 rounds=>15X4=60 words

Table 7.3 *Words for each round*

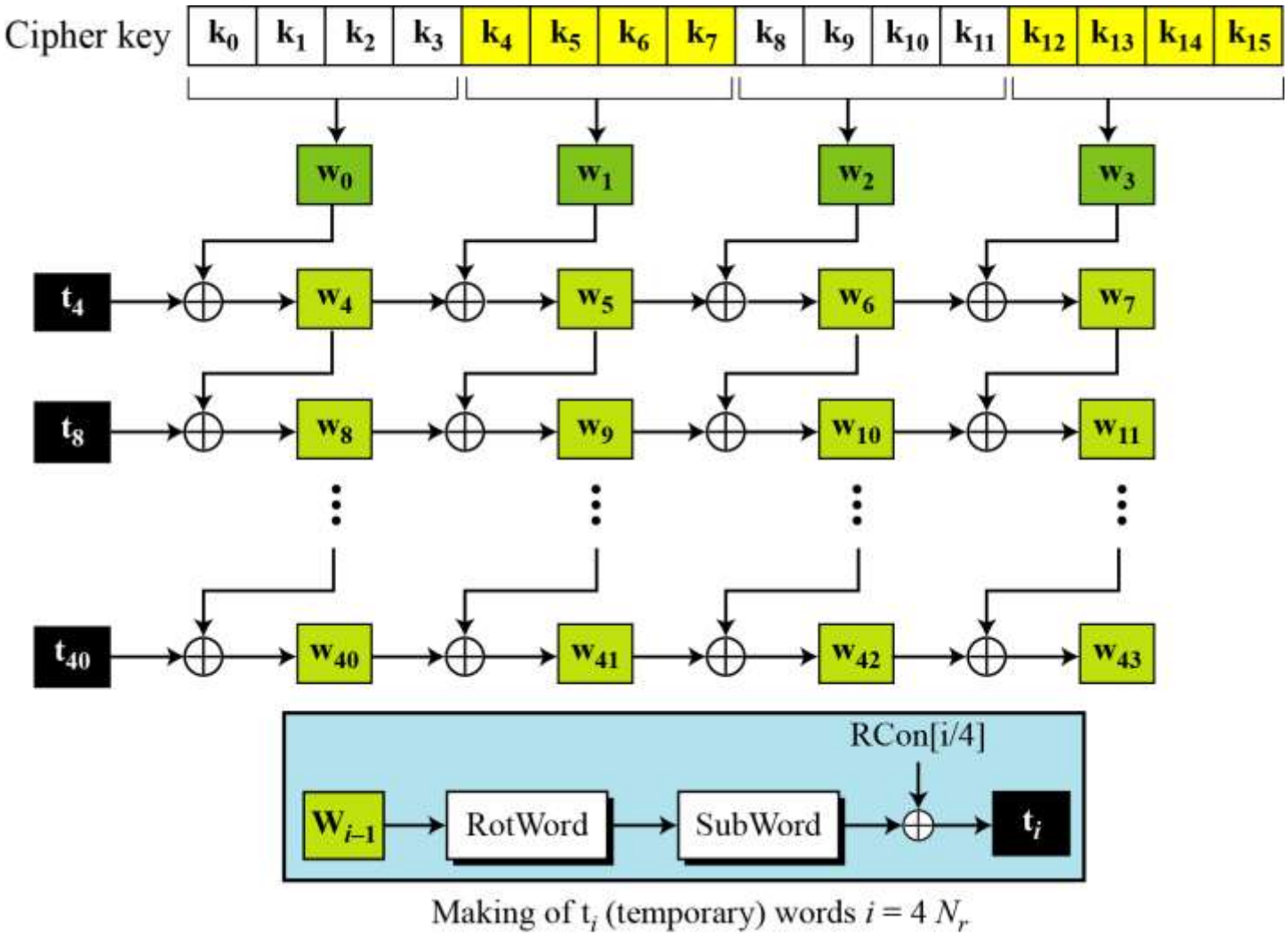
Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

W40 W41 W42 W43

In AES-128 version with 10 rounds=>11X4=44 words

7.3.1 Key Expansion in AES-128

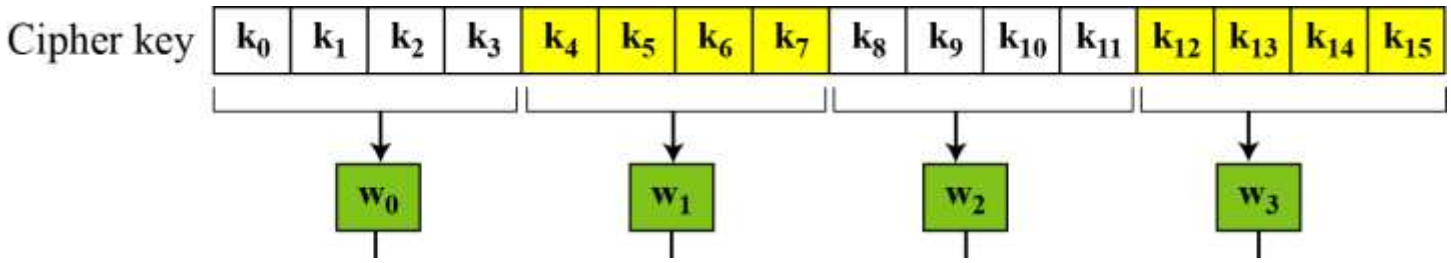
Figure 7.16 *Key expansion in AES*



Making of t_i (temporary) words $i = 4\ N_r$

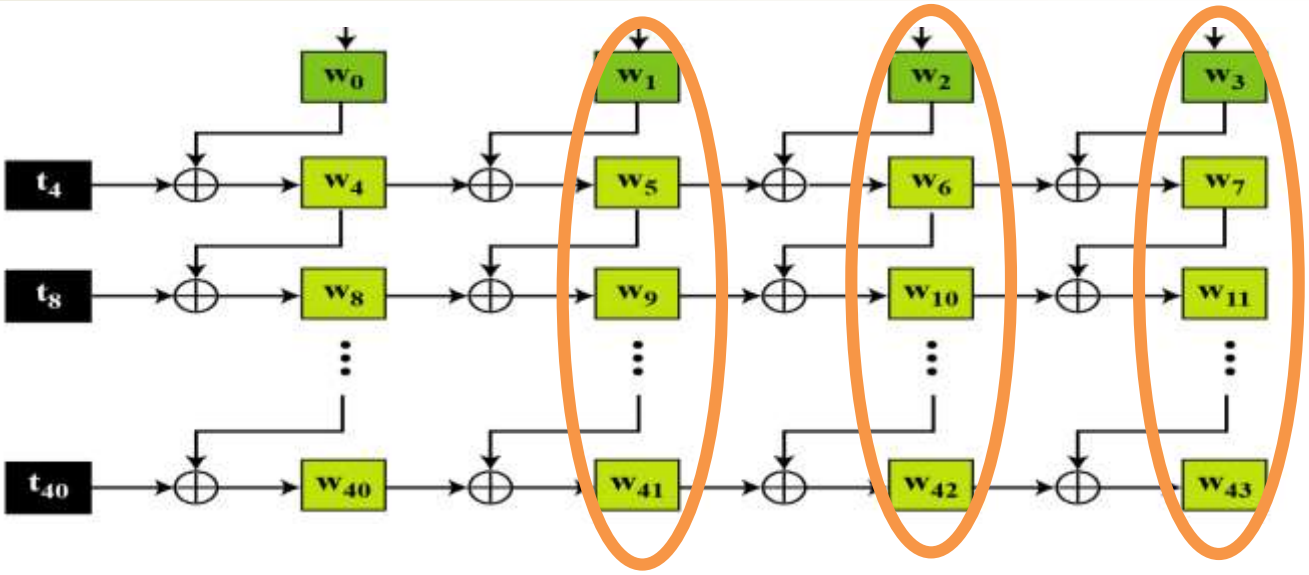
7.3.1 Key Expansion in AES-128

Figure 7.16 *Key expansion in AES*



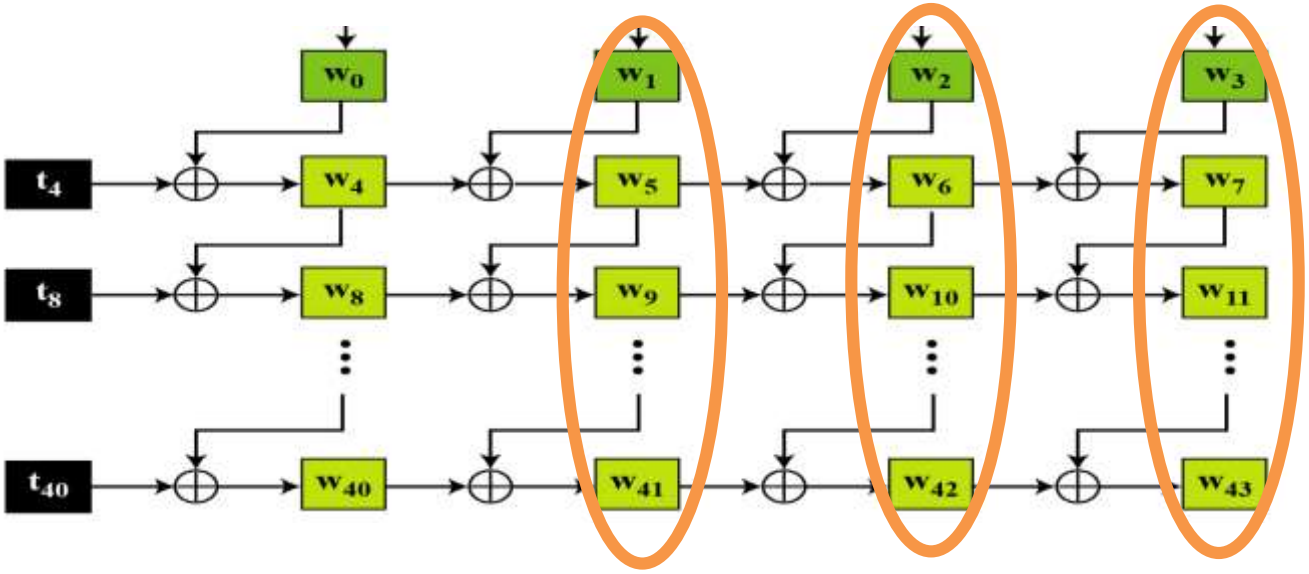
- *The First 4 words are made from the cipher key*
- *Cipher key=array of 16 bytes (ko to k15)*
- *ko,k1,k2,k3=wo*
- *k4,k5,k6,k7=w1*
- *k8,k9,k10,k11=w3*
- *k12,k13,k14,k15=w4*

7.3.1 Key Expansion in AES-128



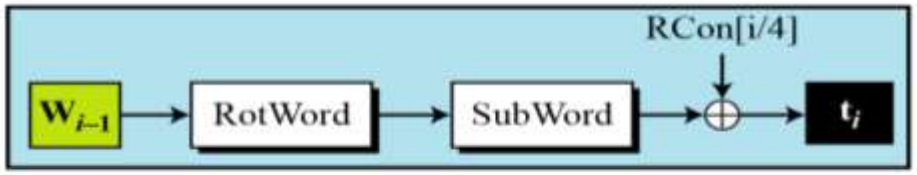
- *Remaining words are calculated as follows:-*
- *For i=4 to 43*
  - *If imod4!=0, wi=wi-1 EXOR wi-4,*
  - *Each word is made from one at the left and one at the top*

7.3.1 Key Expansion in AES-128



- *If imod4=0,*
  - *wi=t EXOR wi-4,*
  - *t=temporary word=Result of applying two Routines Subword and Rot Word on wi-1 and EXORing the result with a round constants RCon*

7.3.1 Key Expansion in AES-128

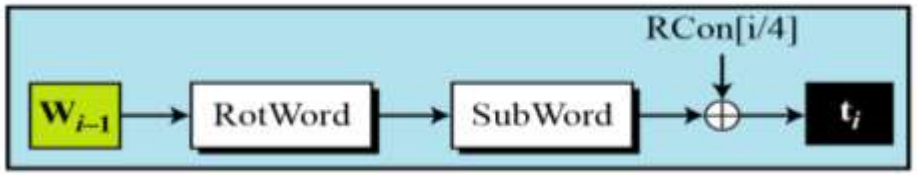


Making of t_i (temporary) words $i = 4\ N_r$

- ***t=temporary word=Result of applying two Routines Subword and Rot Word on wi-1 and EXORing the result with a round constants Rcon***

- ***t=SubWord(RotWord(wi-1))EXOR RConi/4***

7.3.1 Key Expansion in AES-128



Making of t_i (temporary) words $i = 4\ N_r$

- *RotWord-*
  - *Rotate Word*
  - *Takes a word as an array of 4 bytes*
  - *Shifts each byte to the left with wrapping*
- *SubWord-*
  - *Substitute Word*
  - *Takes each byte in the word and substitute another byte for it*
- *Round Constants-*
  - *Rcon=4 byte value, Rightmost three bytes are always Zero*

7.3.1 Continue

Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$

- ***Key Expansion can use the above table for Rcon constants***

7.3.1 Continue

Algorithm 7.5 *Pseudocode for key expansion in AES-128*

```
KeyExpansion ([key0 to key15], [w0 to w43])
{
    for (i = 0 to 3)
        wi ← key4i + key4i+1 + key4i+2 + key4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    wi ← wi–1 + wi–4
        else
        {
            t ← SubWord (RotWord (wi–1)) ⊕ RConi/4        // t is a temporary word
            wi ← t + wi–4
        }
    }
}
```

Table 7.5 shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is $(24\ 75\ A2\ B3\ 34\ 75\ 56\ 88\ 31\ E2\ 12\ 00\ 13\ AA\ 54\ 87)_{16}$.

Table 7.5 *Key expansion example*

Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

RotWord (13AA5487) = AA548713 → **SubWord** (AA548713) = AC20177D

$t = AC20177D \oplus \mathbf{RCon}_1 = AC20\ 17\ 7D \oplus 01000000_{16} = AD20177D$

7.3.1 Continue

Example 7.7

Each round key in AES depends on the previous round key. The dependency, however, is nonlinear because of SubWord transformation. The addition of the round constants also guarantees that each round key will be different from the previous one.

7.3.1 Continue

Example 7.8

The two sets of round keys can be created from two cipher keys that are different only in one bit.

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC A**A** 34 C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC A**B** 34 C2 BB 77 23

7.3.1 Continue

Example 7.8 *Continue*

There are significant differences between the two corresponding round keys

R=Round

BD=Bit Difference

Table 7.6 *Comparing two sets of round keys*

R.	*Round keys for set 1*	*Round keys for set 2*	*B. D.*
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

7.3.1 Continue

Example 7.9

The concept of weak keys, as we discussed for DES in Chapter 6, does not apply to AES. Assume that all bits in the cipher key are 0s. The following shows the words for some rounds:

Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
. . .	. . .	. . .	. . .	. . .
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

- **The words in the pre-round and the first round are all the same.**
- **In the second round, the first word matches with the third; the second word matches with the fourth.**
- **However, after the second round the pattern disappears; every word is different.**

7.3.2 Key Expansion in AES-192 and AES-256

Key-expansion algorithms in the AES-192 and AES-256 versions are very similar to the key expansion algorithm in AES-128, but with few differences

7.3.2 Key Expansion in AES-192 and AES-256

Differences:

- *In AES-192,*
  - *The words are generated in groups of six instead of four*
  - *The Cipher key creates the first six words (wo to w5)*
  - *If imod6!=0,wi=wi-1+wi-6, else wi=t+wi-6*
- *In AES-256,*
  - *The words are generated in groups of eight instead of four*
  - *The Cipher key creates the first eight words (w0 to w7)*
  - *If imod8!=0, wi=wi-1+wi-8, else wi=t+wi-8*
  - *If imod4=0 but imod8!=0, then wi=SubWord(wi-1)+wi-8*

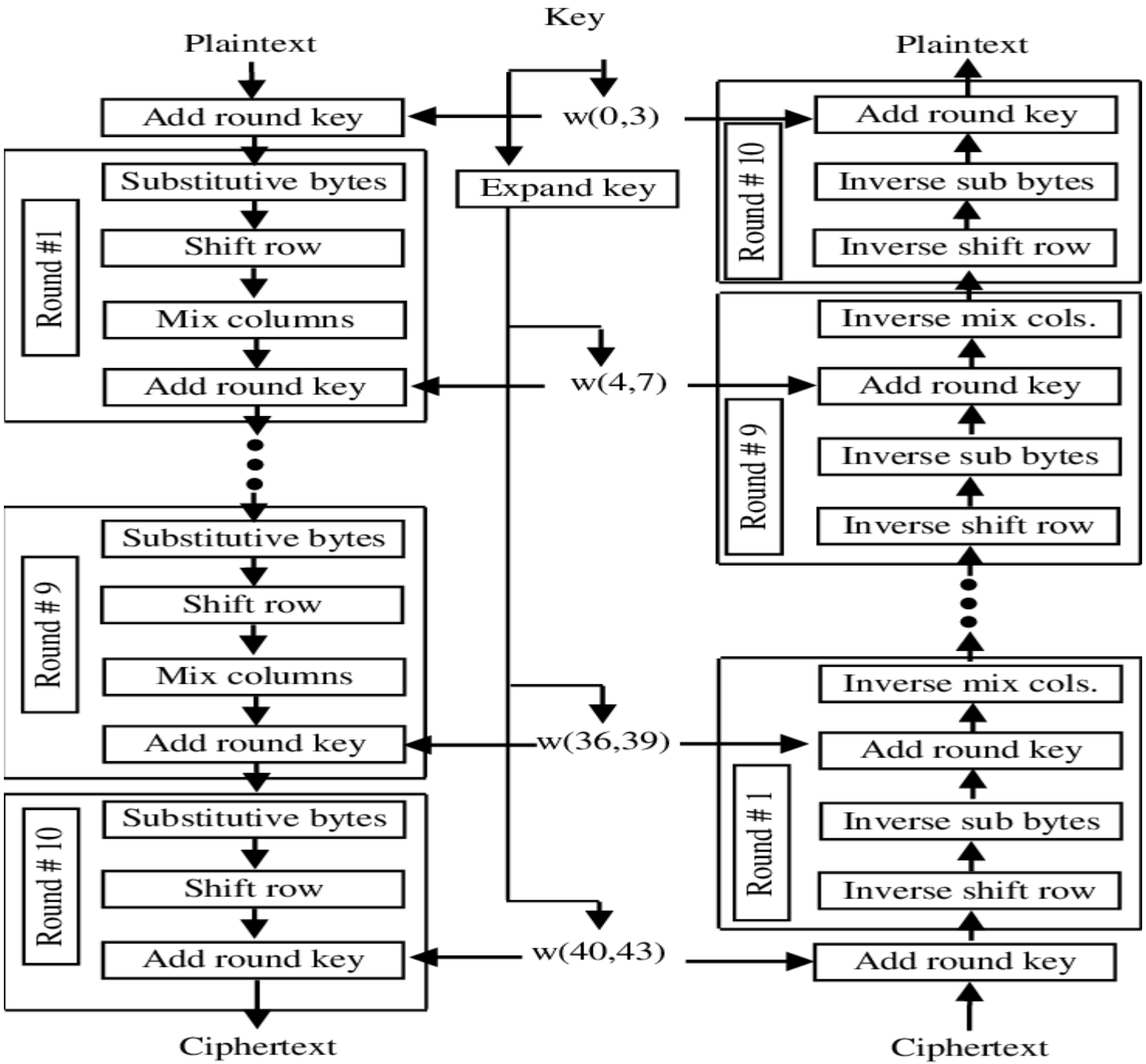
7.3.3 Key-Expansion Analysis

The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

- *Two different Cipher keys, no matter how similar to each other, produce two expansions that differ in atleast a few rounds*
- *Each bit of cipher key is diffused into several rounds. Changing a single bit in the cipher key, will change some bits in several rounds*
- *No serious weak keys in AES*

7.3.3 Key-Expansion Analysis

- *Key expansion can be easily implemented on all platforms*
- *Even if Eve knows only part of the cipher key or the values of the words in some round keys, she still needs to find the rest of the cipher key before she can find all round keys. Its because of the Non-Linearity produced by SubWord transformation in the key expansion process*


Key
Plaintext
Add round key
w(0,3)
Expand key
Round #1
Substitutive bytes
Shift row
Mix columns
Add round key
w(4,7)
Round # 9
Substitutive bytes
Shift row
Mix columns
Add round key
w(36,39)
Round # 10
Substitutive bytes
Shift row
Add round key
w(40,43)
Ciphertext
Plaintext
Round # 10
Add round key
Inverse sub bytes
Inverse shift row
Round # 9
Inverse mix cols.
Add round key
Inverse sub bytes
Inverse shift row
Round # 1
Inverse mix cols.
Add round key
Inverse sub bytes
Inverse shift row
Add round key
Ciphertext

Key-Expansion Analysis

1. Even if Eve knows only part of the cipher key or the values of the words in some round keys, she still needs to find the rest of the cipher key before she can find all round keys. This is because of the nonlinearity produced by SubWord transformation in the key-expansion process.
2. Two different cipher keys, no matter how similar to each other, produce two expansions that differ in at least a few rounds.
3. Each bit of the cipher key is diffused into several rounds. For example, changing a single bit in the cipher key, will change some bits in several rounds.
4. The use of the constants, the RCons, removes any symmetry that may have been created by the other transformations.
5. There are no serious weak keys in AES, unlike in DES.
6. The key-expansion process can be easily implemented on all platforms.

Security Strength of AES

- AES is highly secure due to its use of longer key lengths and more complex encryption rounds compared to DES. Its strength lies in its resistance to various cryptographic attacks:
- **Brute-Force Attacks:** AES's key lengths (128, 192, and 256 bits) make brute-force attacks impractical, even with modern computing power.
- **Linear and Differential Cryptanalysis:** AES is designed to resist both linear and differential cryptanalysis, which are common methods for breaking symmetric ciphers.
- **Side-Channel Attacks:** While AES itself is secure, implementations of AES must be protected against side-channel attacks, such as timing attacks and power analysis attacks

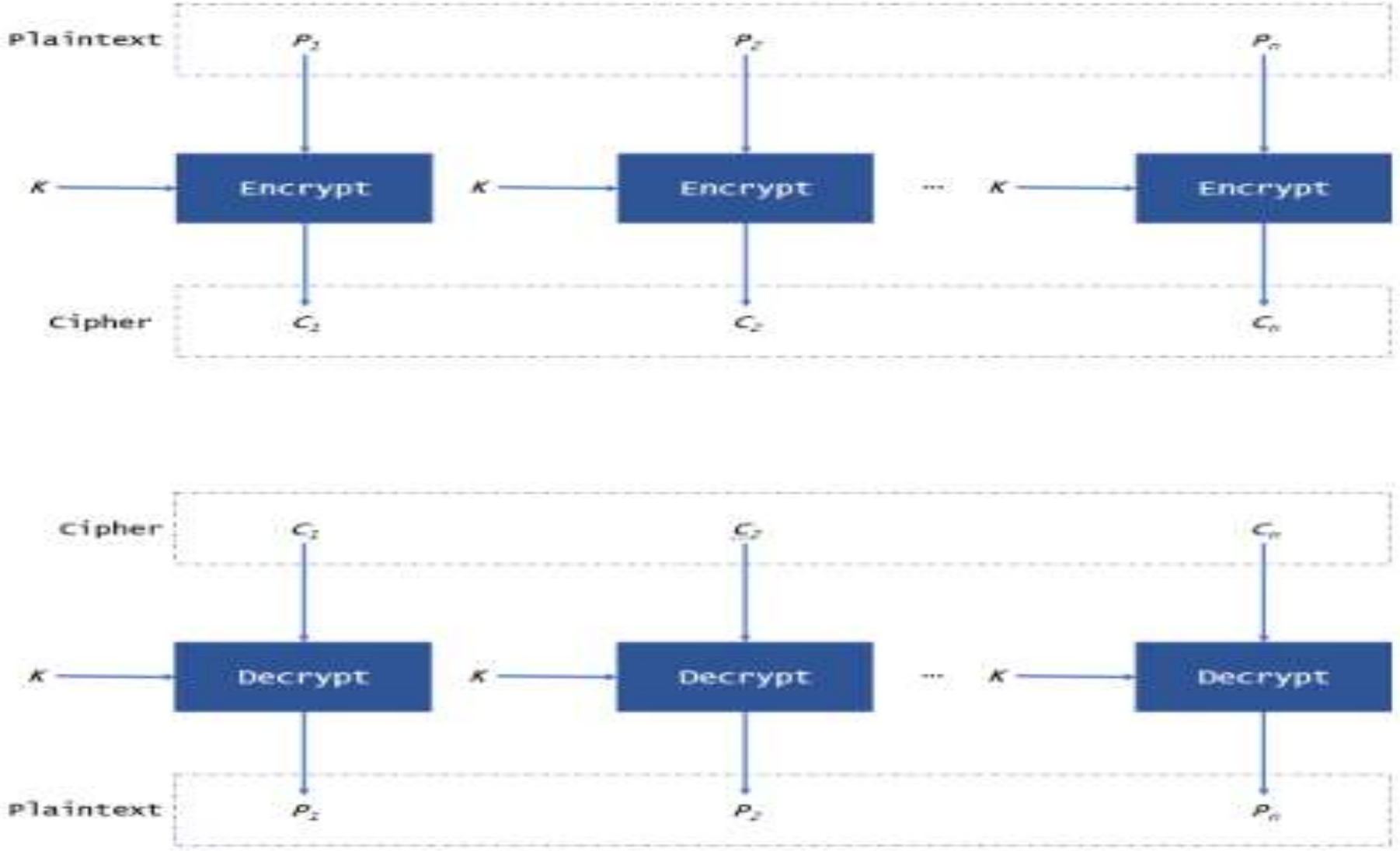
Advantages of AES

- The AES algorithm provides several advantages over older algorithms such as the Data Encryption Standard (DES):
- **Security**. AES offers stronger security since it incorporates multiple rounds of encryption, making it harder to break, and harder for threat actors to intercept or steal the encrypted information using brute-force attacks.
- **Cos**t. AES is an open source and ubiquitously available solution, making it cost-effective to adopt and implement.
- **Implementation**. AES is a flexible and simple algorithm, making it suitable for both hardware and software implementation.

Modes of Operation

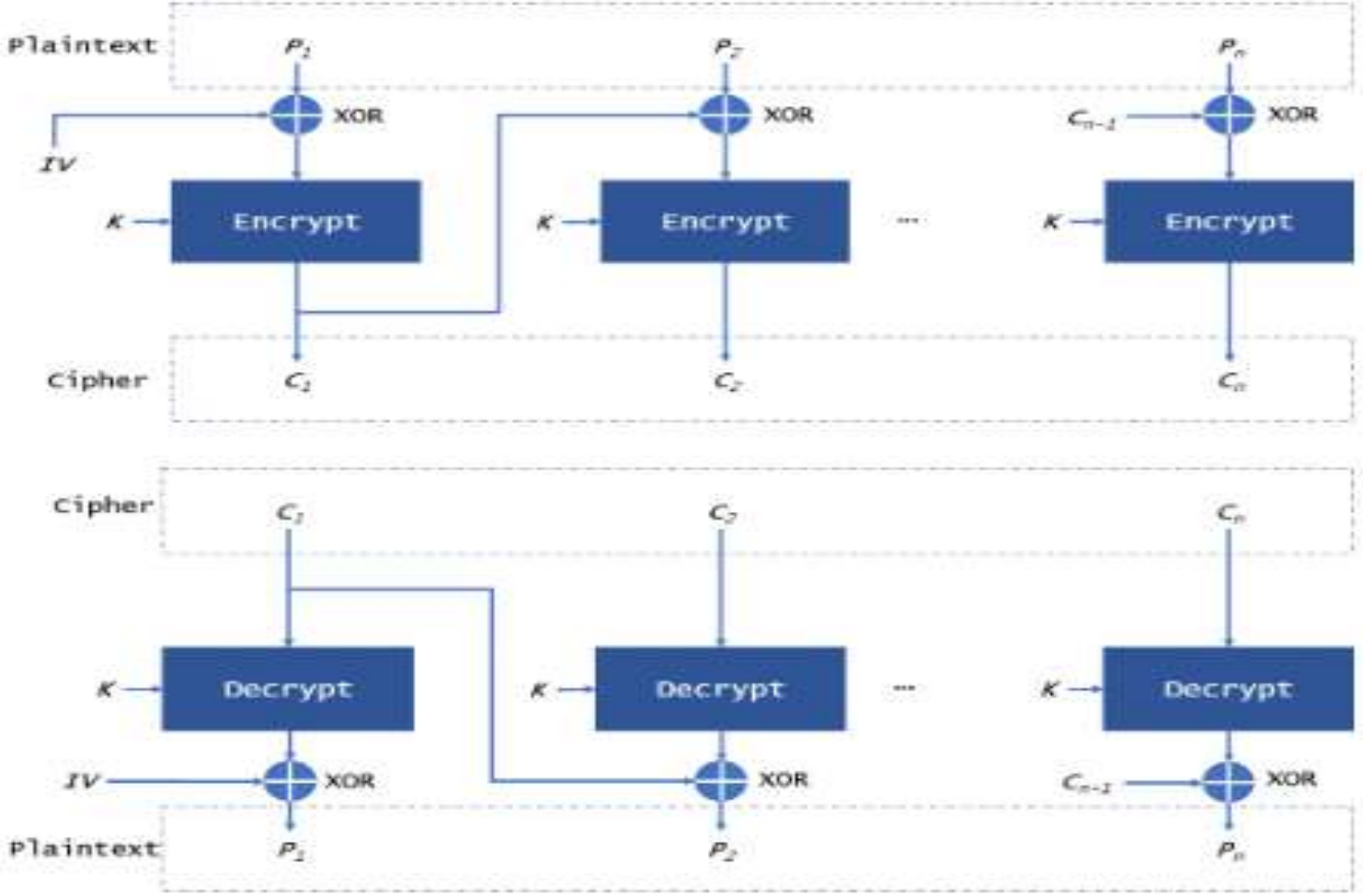
- ECB mode: Electronic Code Book mode
- CBC mode: Cipher Block Chaining mode
- CFB mode: Cipher FeedBack mode
- OFB mode: Output FeedBack mode
- CTR mode: Counter mode

ECB mode: Electronic Code Book mode



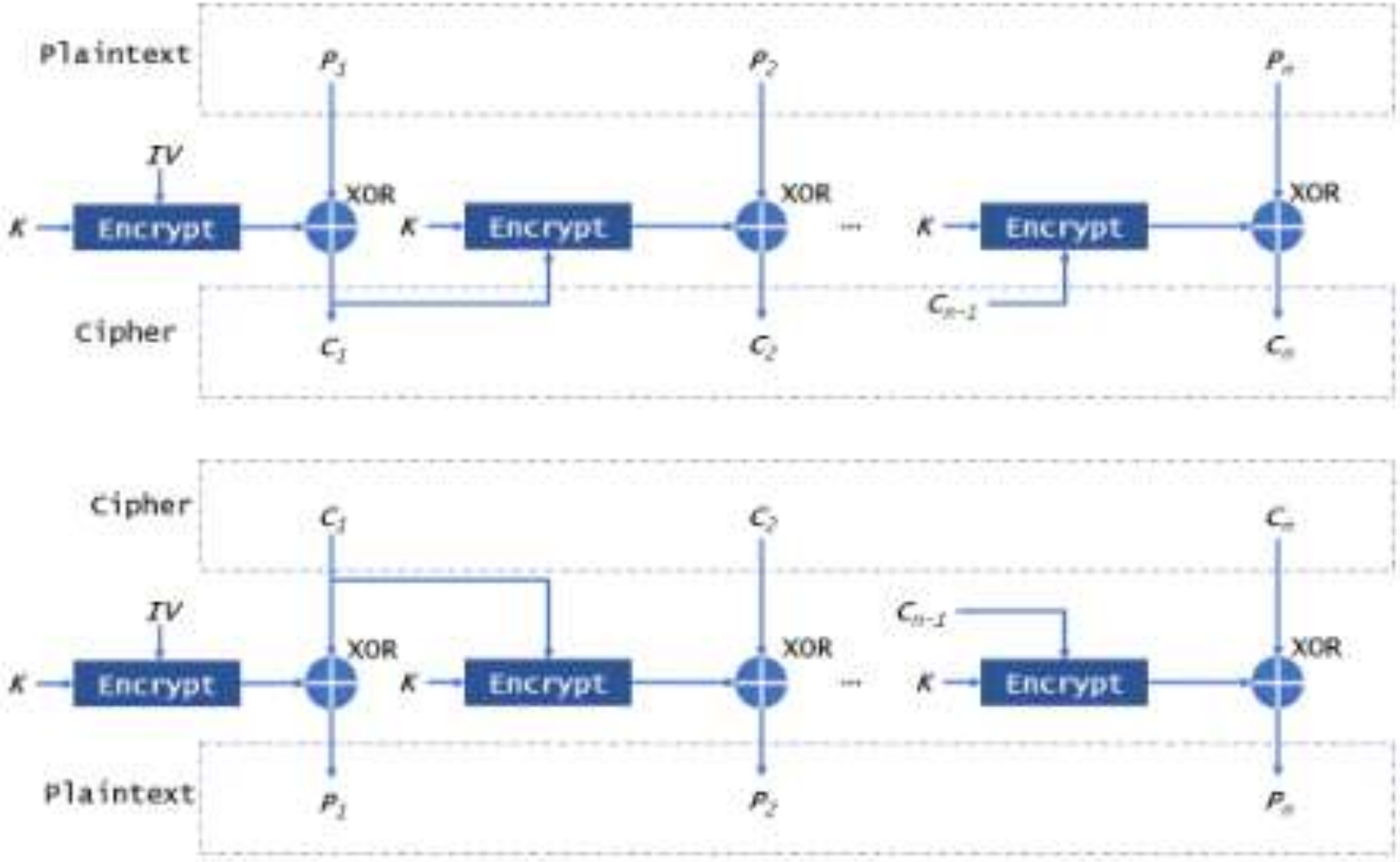
- ECB mode needs to pad data until it is same as the length of the block.
- Every block will be encrypted with the same key and same algorithm.
- Encrypt the same plaintext, will get the same ciphertext.
- So there is a high risk in this mode.
- The plaintext and ciphertext blocks are a one-to-one correspondence.
- Because the encryption/ decryption is independent, so we can encrypt/decrypt the data in parallel.
- And if a block of plaintext or ciphertext is broken, it won't affect other blocks.

CBC mode: Cipher Block Chaining mode



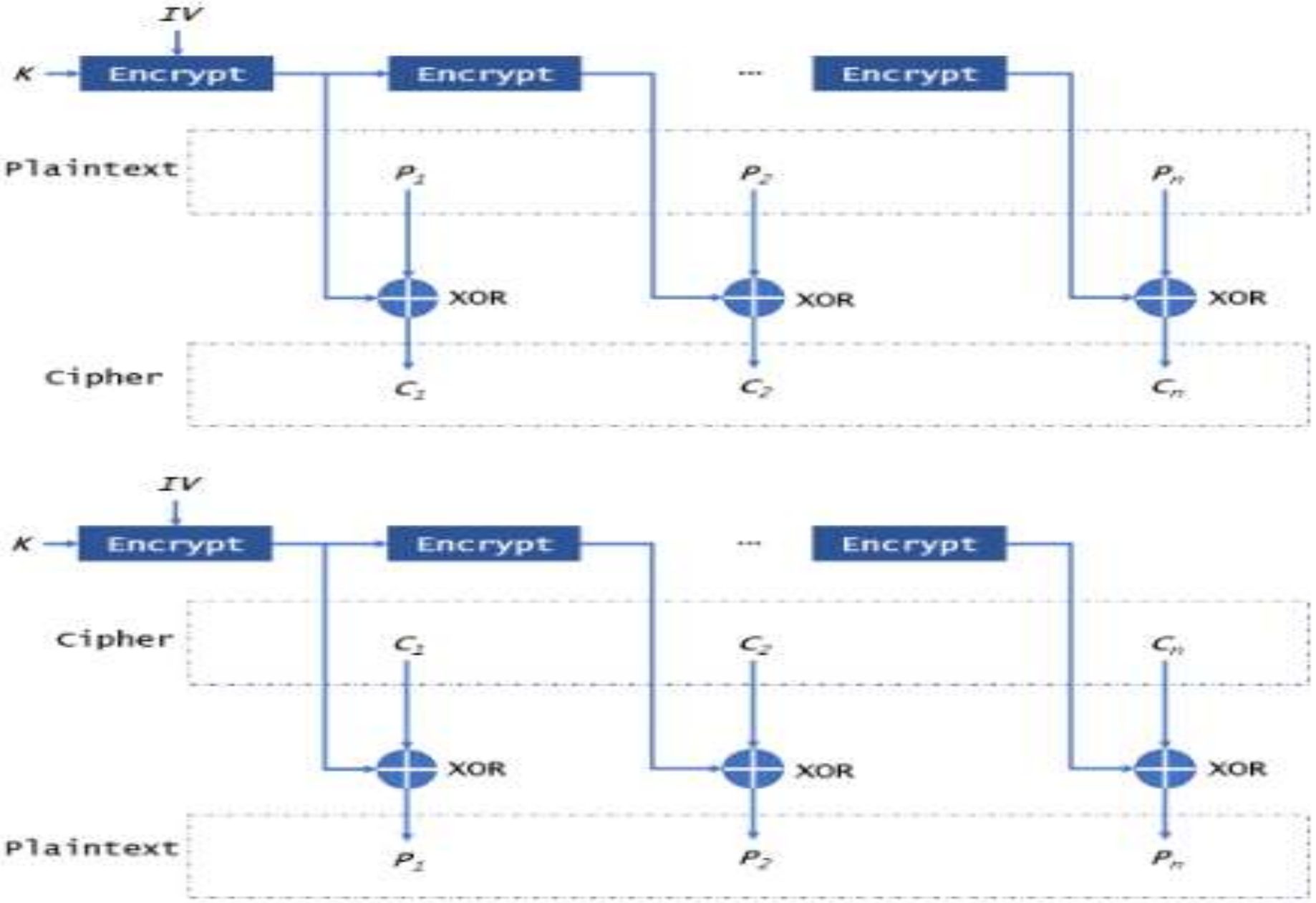
- The plaintext is divided into blocks and needs to add padding data.
- First, we will use the plaintext block xor with the initialization vector IV.
- Then CBC will encrypt the result to the ciphertext block.
- In the next block, we will use the encryption result to xor with plaintext block until the last block.
- In this mode, even if we encrypt the same plaintext block, we will get a different ciphertext block.
- We can decrypt the data in parallel, but it is not possible when encrypting data. If a plaintext or ciphertext block is broken, it will affect all following block.

CFB mode: Cipher FeedBack mode



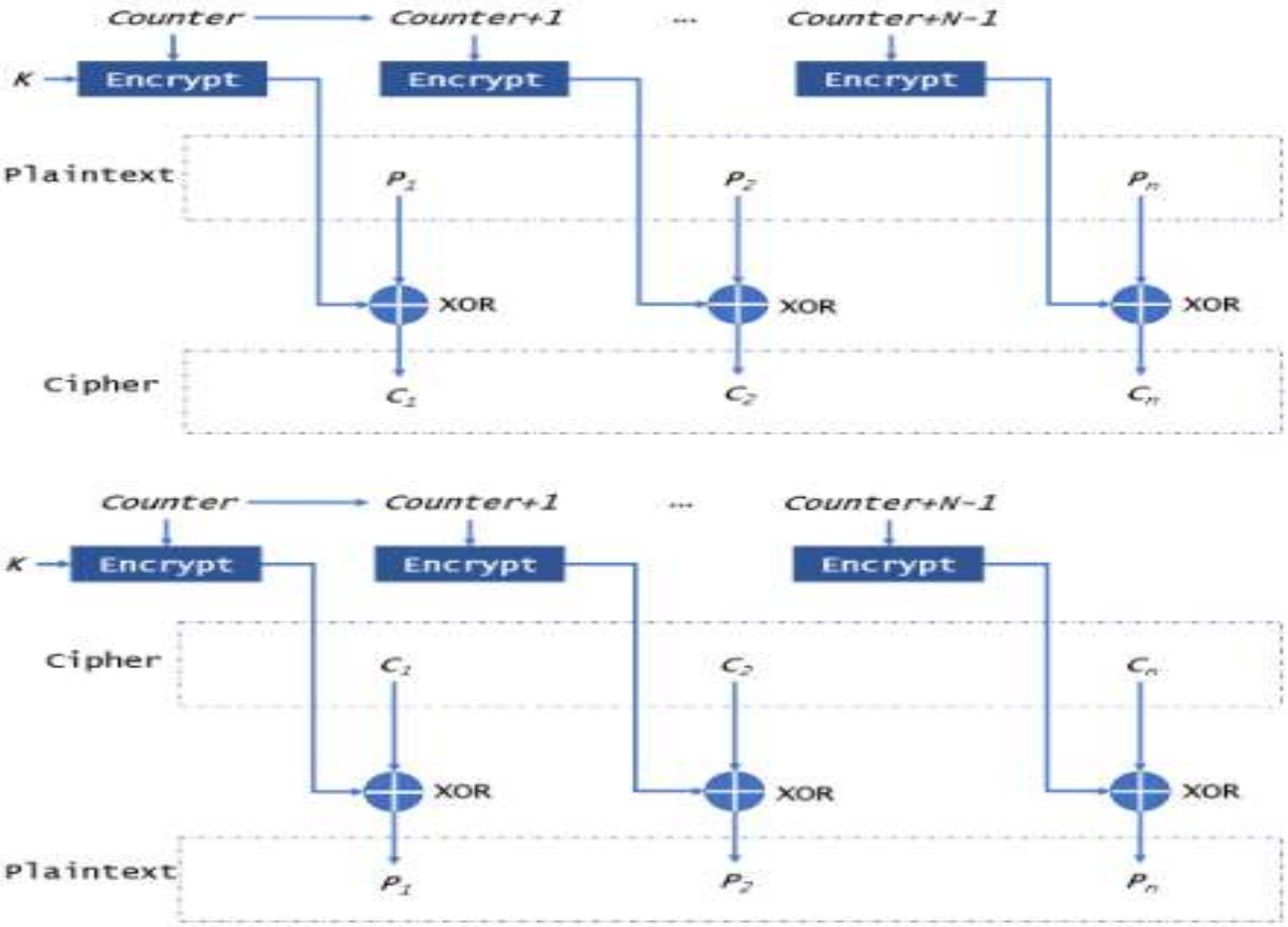
- First, CFB will encrypt the IV, then it will xor with plaintext block to get ciphertext.
- Then encrypt the encryption result to xor the plaintext.
- Because this mode will not encrypt plaintext directly, it just uses the ciphertext to xor with the plaintext to get the ciphertext.
- So in this mode, it doesn’t need to pad data.
- And it could decrypt data in parallel, not encryption.
- This mode is similar to the CBC, so if there is a broken block, it will affect all following block.

OFB mode: Output FeedBack mode



- In this mode, it will encrypt the IV in the first time and encrypt the per-result.
- Then it will use the encryption results to xor the plaintext to get ciphertext.
- It is different from CFB, it always encrypts the IV.
- It can not encrypt/decrypt the IV in parallel.
- Please note that we won't decrypt the IV encryption results to decrypt data.
- It will not be affected by the broken block.

CTR mode: Counter mode



- The counter has the same size as the used block.
- The XOR operation with the block of plain text is performed on the output block from the encryptor.
- All encryption blocks use the same encryption key.
- As this mode, It will not be affected by the broken block. It is very like OFB.
- But CTR will use the counter to be encrypted every time instead of the IV. So if you could get counter directly, you can encrypt/decrypt data in parallel.

Mode	Advantage	Disadvantage
Electronic CodeBook (ECB)	• Simple • fast • Support for parallel (encryption/decryption)	• Duplicate data in plaintext will be reflected in the ciphertext • The plaintext can be operated by deleting or replacing the ciphertext. • If the ciphertext packet is damaged, it will affect the plaintext. • Can't resist replay attacks • Should not be used
Cipher Block Chaining (CBC)	• Support for parallel computing (decryption) • Ability to decrypt any ciphertext packet • Duplicate data in plaintext will not be reflected in the ciphertext	• Do not Support for parallel computing (Encryption) • Wrong blocks affect all following blocks
Cipher-FeedBack (CFB)	• No padding • Support for parallel computing (decryption) • Ability to decrypt any ciphertext packet • Can be prepared for encryption and decryption first	• Do not Support for parallel computing (Encryption) • Can't resist replay attacks • Wrong blocks affect all following blocks
Output-FeedBack (OFB)	• No padding • Can prepare encryption and decryption in advance • Encryption and decryption use the same structure • Bad blocks only affect the current block	• Do not Support for parallel computing • Mallory can change some ciphertext damaged plaintext
CounYeR (CTR)	• No padding • Can prepare encryption and decryption in advance • Support for parallel computing • Encryption and decryption use the same structure • Bad blocks only affect the current block	• Mallory can change some ciphertext damaged plaintext