Cryptography and System Security (CSS)

Course Code: CSC 604

Subject Incharge

Ankita Karia

Assistant Professor

Room No. 421

email: ankitakaria@sfit.ac.in

Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

2.1

- Block Cipher
  - ✓ Block Cipher Principles
  - ✓ Block Cipher Modes of Operation
  - ✓ Data Encryption Standard (DES)
  - ✓ Double DES, Triple DES
  - ✓ Advanced Encryption Standard (AES)
- Stream Cipher
  - ✓ RC5 algorithm

Block Cipher

1. Operates on the blocks of plain text, instead of operating on each bit of plain text separately.
2. Each block is of equal size and has fixed no of bits.
3. The generated ciphertext has blocks equal to the number of blocks in plaintext and also has the same number of bits in each block as of plain text.
4. Block cipher uses the same key for encryption and decryption.

Block Cipher (Contd...)

1. Block cipher is an encryption method which divides the plain text into blocks of fixed size.
2. Each block has an equal number of bits.
3. At a time, block cipher operates only on one block of plain text and applies key on it to produce the corresponding block of ciphertext.
4. While decryption also only one block of ciphertext is operated to produce its corresponding plain text.
5. Data Encryption Standard (DES) is the best example of it.

Block Cipher Principles

The design of the block cipher is based on the three principles, which are

1. Number of rounds
2. Design of function F
3. Key schedule algorithm

- The number of rounds judges the strength of the block cipher algorithm.
- It is considered that more is the number of rounds, difficult is for cryptanalysis to break the algorithm.
- It is considered that even if the function F is relatively weak, the number of rounds would make the algorithm tough to break.

Block Cipher Principles

The design of the block cipher is based on the three principles, which are

1. Number of rounds
2. Design of function F
3. Key schedule algorithm

a) The function F of the block cipher must be designed such that it must be impossible for any cryptanalysis to unscramble the substitution.

b) The criterion that strengthens the function F is it non-linearity.

c) The Function F should be designed such that it possesses a bit independence criterion which states that the output bits must change independently if there is any change in the input bit.

Block Cipher Principles

The design of the block cipher is based on the three principles, which are

1. Number of rounds
2. Design of function F
3. Key schedule algorithm

This suggests that the schedule key should be able to confirm the strict effect of avalanche and the criterion of bit independence.

Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

2.1

- Block Cipher
  - ✓ Block Cipher Principles
  - ✓ Block Cipher Modes of Operation
  - ✓ Data Encryption Standard (DES)
  - ✓ Double DES, Triple DES
  - ✓ Advanced Encryption Standard (AES)
- Stream Cipher
  - ✓ RC5 algorithm

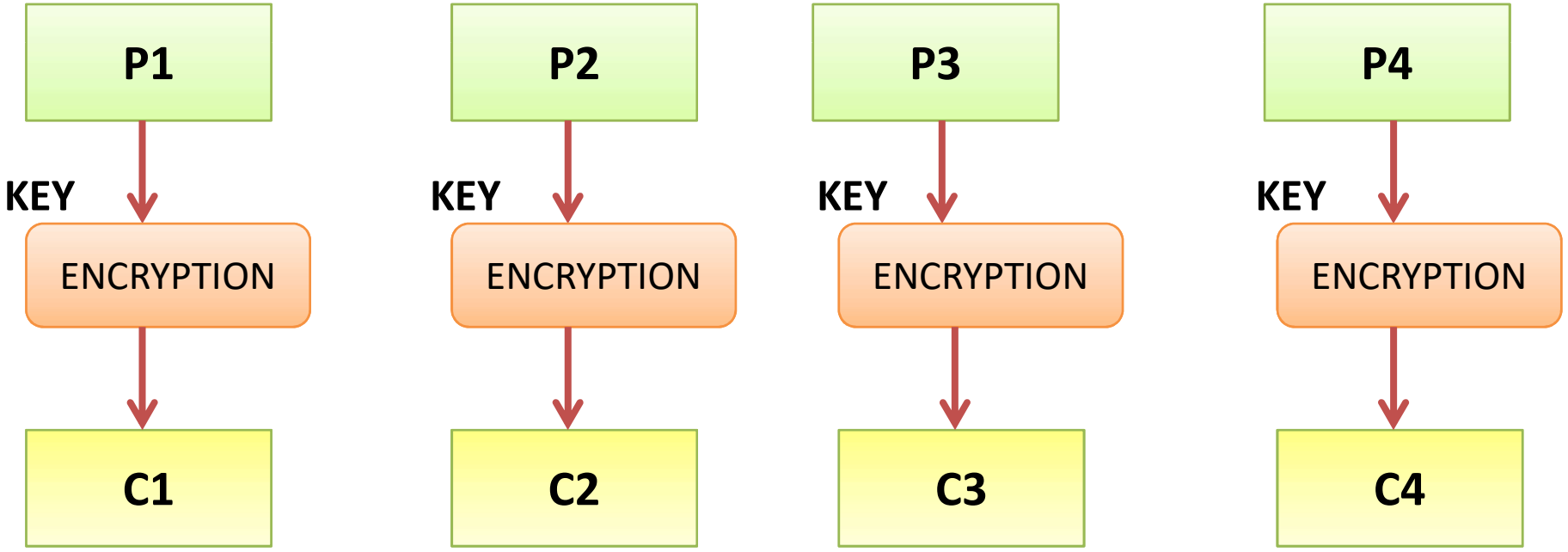
Block Cipher Modes of Operation

There are five important block cipher modes of operation defined by NIST. These five modes of operation enhance the algorithm so that it can be adapted by a wide range of applications which uses block cipher for encryption.

1. Electronic Code Book Mode (ECB)
2. Cipher Block Chaining Mode (CBC)
3. Cipher Feedback Mode (CFB)
4. Output Feedback Mode (OFB)
5. Counter Mode

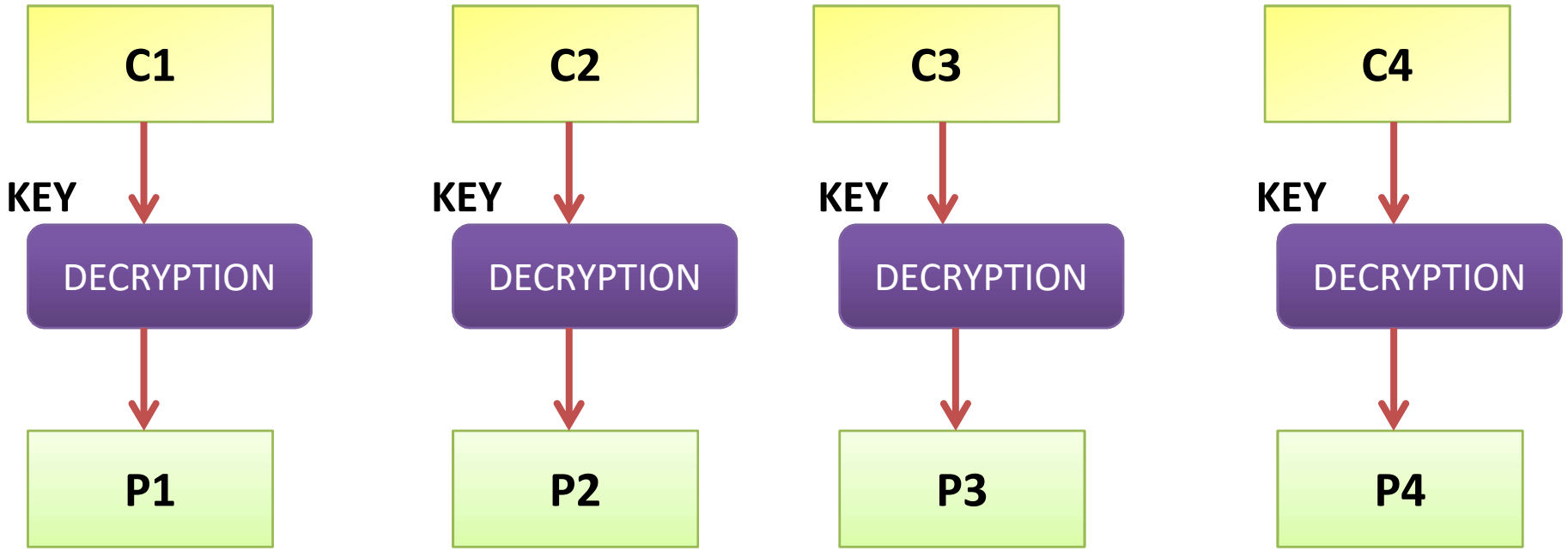
Electronic Code Book Mode (ECB)

- Easiest block cipher mode of operation.
- The plain text is divided into the blocks, each of 64-bit.
- Each block is encrypted one at a time to produce the cipher block.
- The same key is used to encrypt each block.



Electronic Code Book Mode (ECB)

- This ciphertext is again divided into blocks, each of 64-bit and each block is decrypted independently one at a time to obtain the corresponding plain text block.
- The same key is used to decrypt each block which was used to encrypt each block..



Electronic Code Book Mode (ECB)

ADVANTAGES:

- ✓ Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
- ✓ Simple way of block cipher.

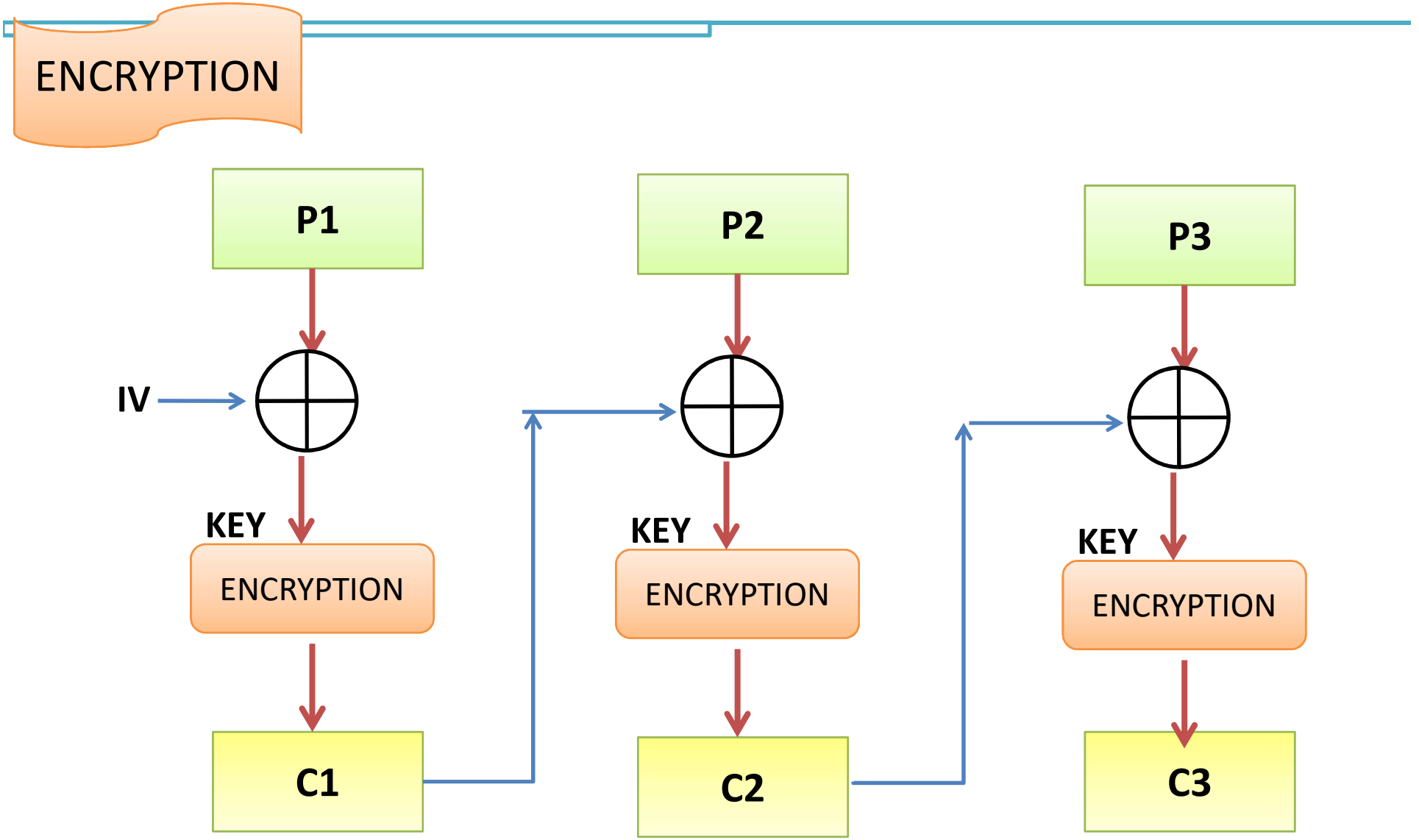
DISADVANTAGES:

- ✓ Prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext.

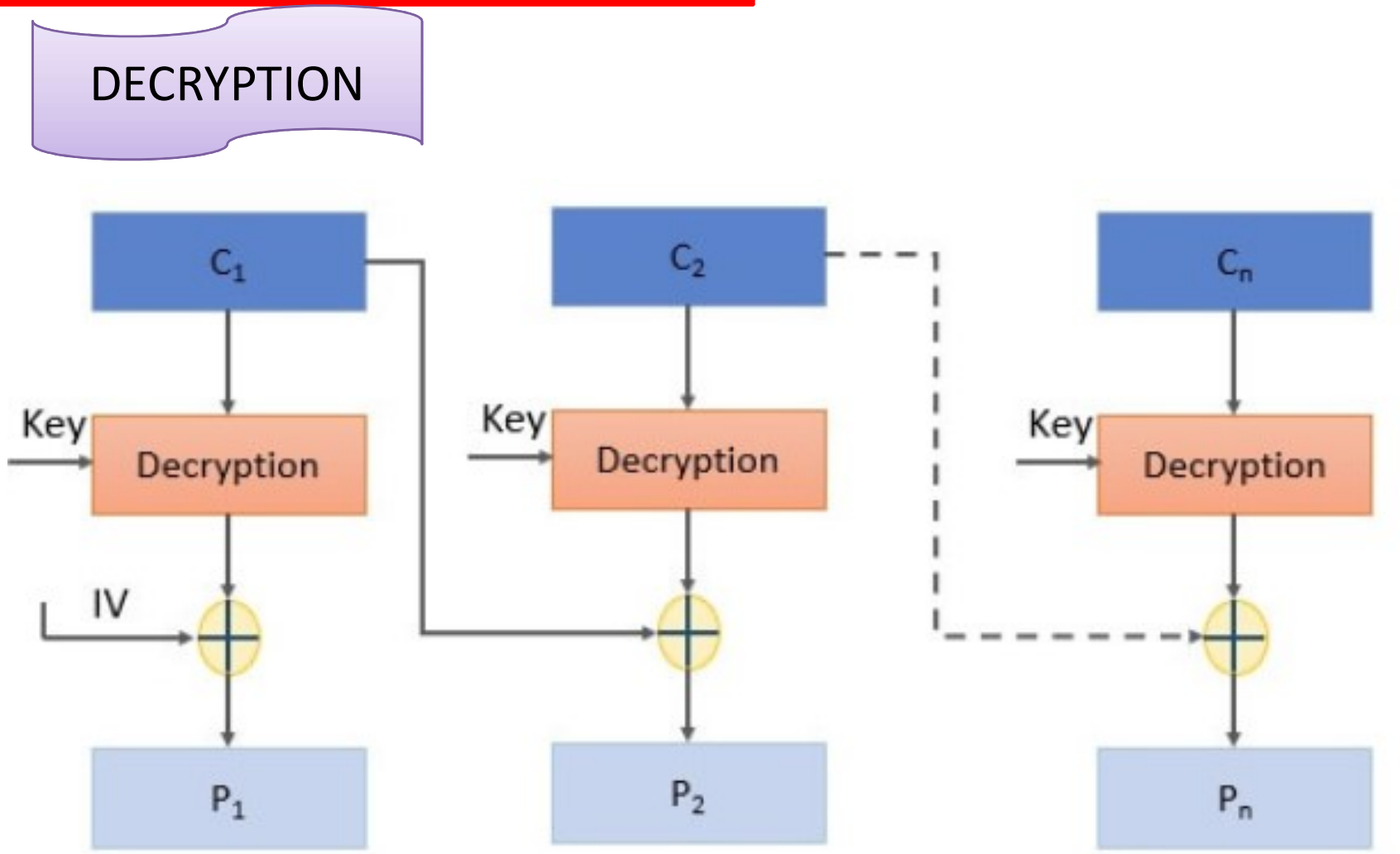
Cipher Block Chaining Mode (CBC)

1. CBC confirms that even if the plain text has repeating blocks its encryption won't produce same cipher block.
2. Chaining has been added to the block cipher.
3. For this, the result obtained from the encryption of the first plain text block is fed to the encryption of the next plaintext box.
4. Since, during the encryption of first plain text block, no previous plain text block is available so a random block of text is generated called Initialization vector.

Cipher Block Chaining Mode (CBC)



Cipher Block Chaining Mode (CBC)



Cipher Block Chaining Mode (CBC)

ADVANTAGES:

- ✓ CBC works well for input greater than b bits.
- ✓CBC is a good authentication mechanism.
- ✓Better resistive nature towards cryptanalysis than ECB.

DISADVANTAGES:

- ✓ Parallel encryption is not possible since every encryption requires previous cipher.

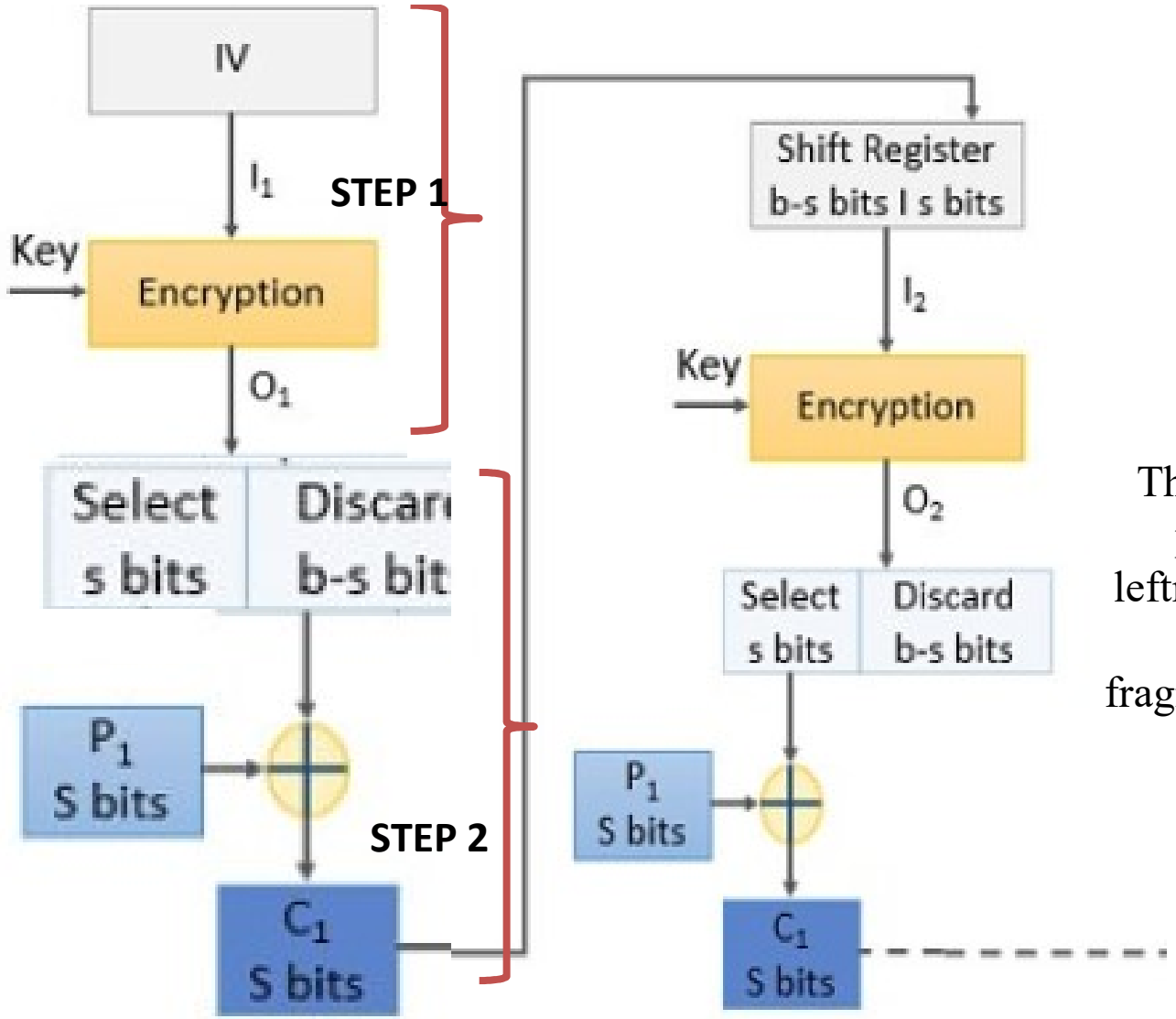
Cipher Feedback Mode (CFB)

1. All applications may not be designed to operate on the blocks of data, some may be character or bit-oriented.
2. Cipher feedback mode is used to operate on smaller units than blocks.

The encryption steps in cipher feedback mode:

Step 1: Here also we use initialization vector, IV is kept in the shift register and it is encrypted using the key.

Step 2: The left most s bits of the encrypted IV is then XORed with the first fragment of the plain text of s bits. It produces the first ciphertext C1 of s bits.



Then again, the encryption is performed on IV and the leftmost s bit of encrypted IV is XORed with the second fragment of plain text to obtain s bit ciphertext C2.

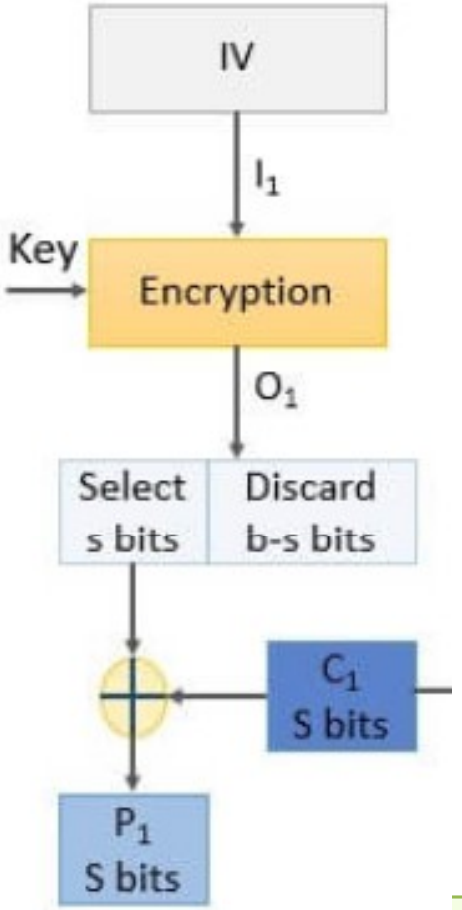
Step 3: Now the shift register containing initialization vector performs left shift by s bits and s bits C1 replaces the rightmost s bits of the initialization vector.

Cipher Feedback Mode Mode (CFB)

DECRYPTION PROCESS

Step 1:

The initialization vector is placed in the shift register. It is encrypted using the same key. Then from the encrypted IV s bits are XORed with the s bits ciphertext C1 to retrieve s bits plain text P1.

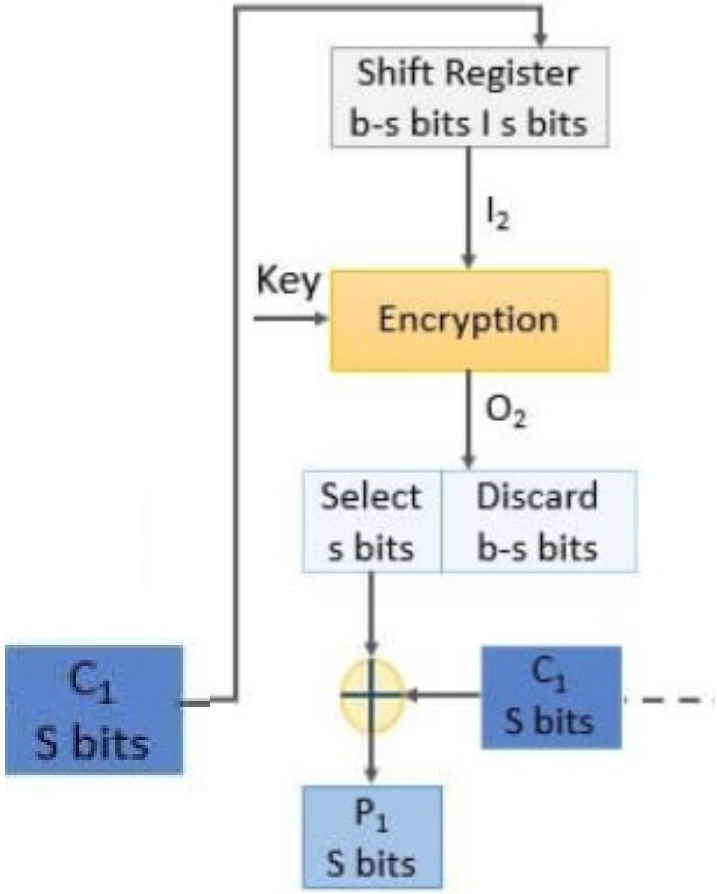


In the **decryption process,** the **encryption** algorithm is implemented instead of the decryption algorithm.

Cipher Feedback Mode Mode (CFB)

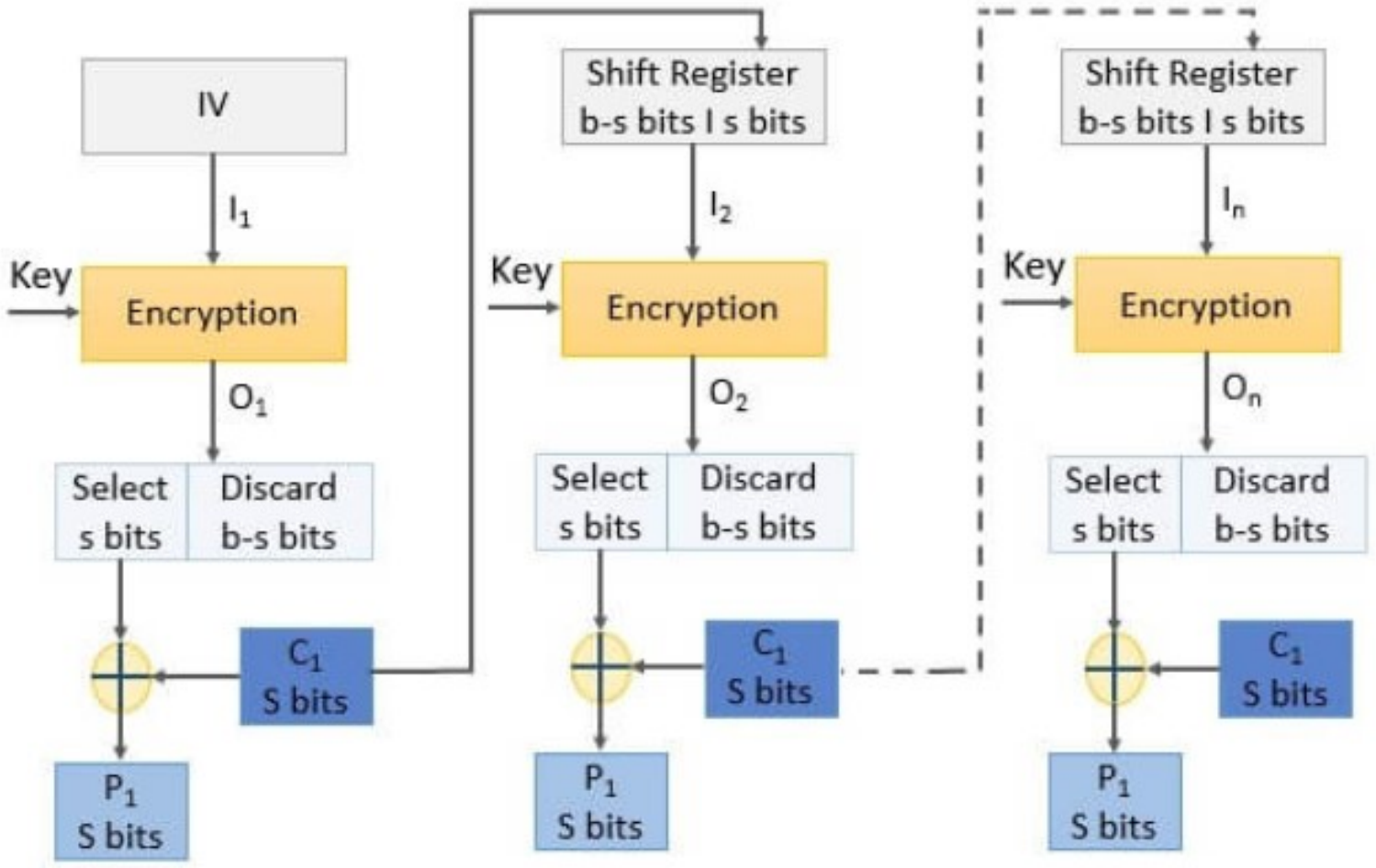
DECRYPTION PROCESS

Step 2: The IV in the shift register is left-shifted by s bits and the s bits C1 replaces the rightmost s bits of IV.



Cipher Feedback Mode Mode (CFB)

DECRYPTION PROCESS



Output Feedback Mode (OFB)

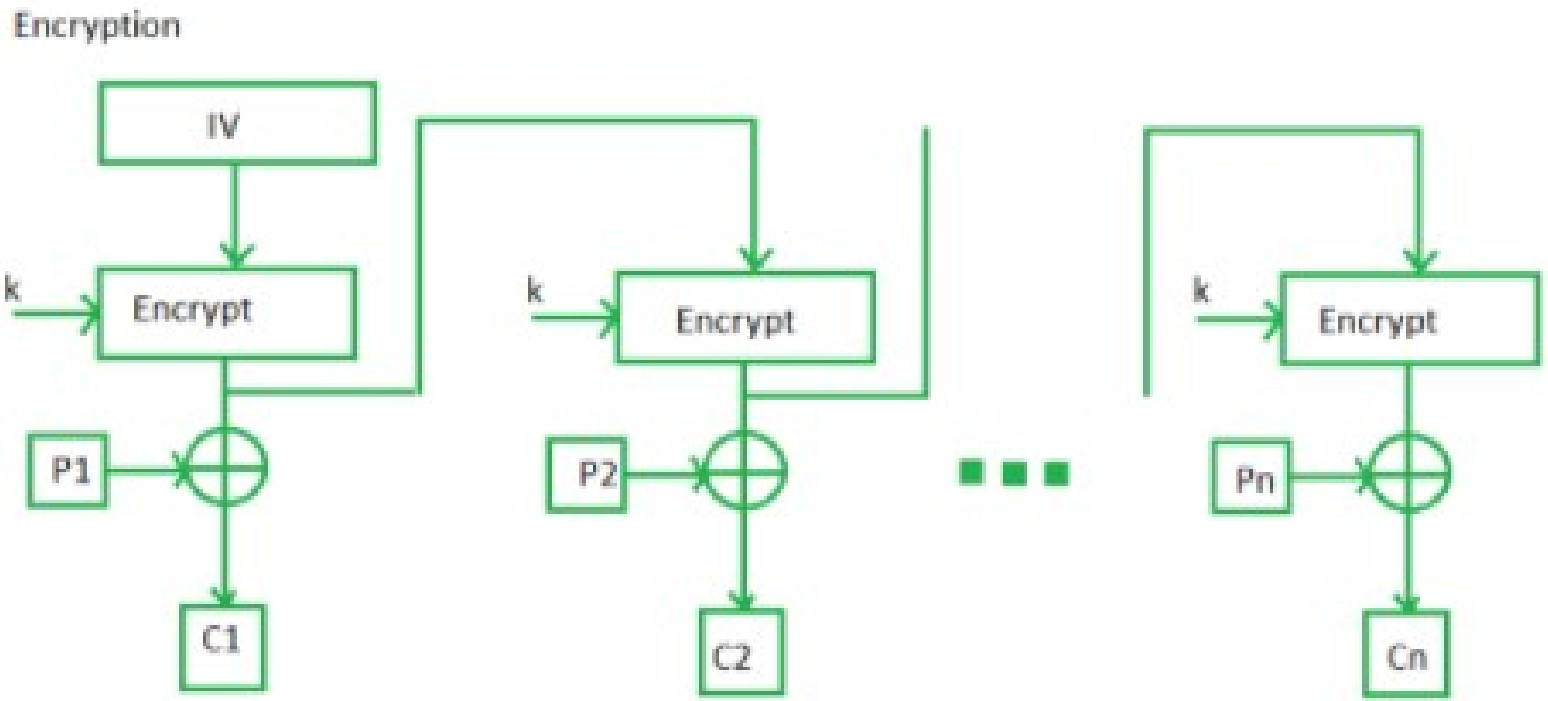
1. The output feedback (OFB) mode is almost similar to the CFB.
2. In OFB the encrypted IV is fed to the encryption of next plain text block.
3. The other difference is that CFB operates on a stream of bits whereas OFB operates on the block of bits.

The encryption steps in cipher feedback mode:

Step 1: The initialization vector is encrypted using the key.

Step 2: The encrypted IV is then XORed with the plain text block to obtain the ciphertext block. The encrypted IV is fed to the encryption of next plain text block

Output Feedback Mode Mode (OFB)



ENCRYPTION

Output Feedback Mode (OFB)

DECRYPTION PROCESS

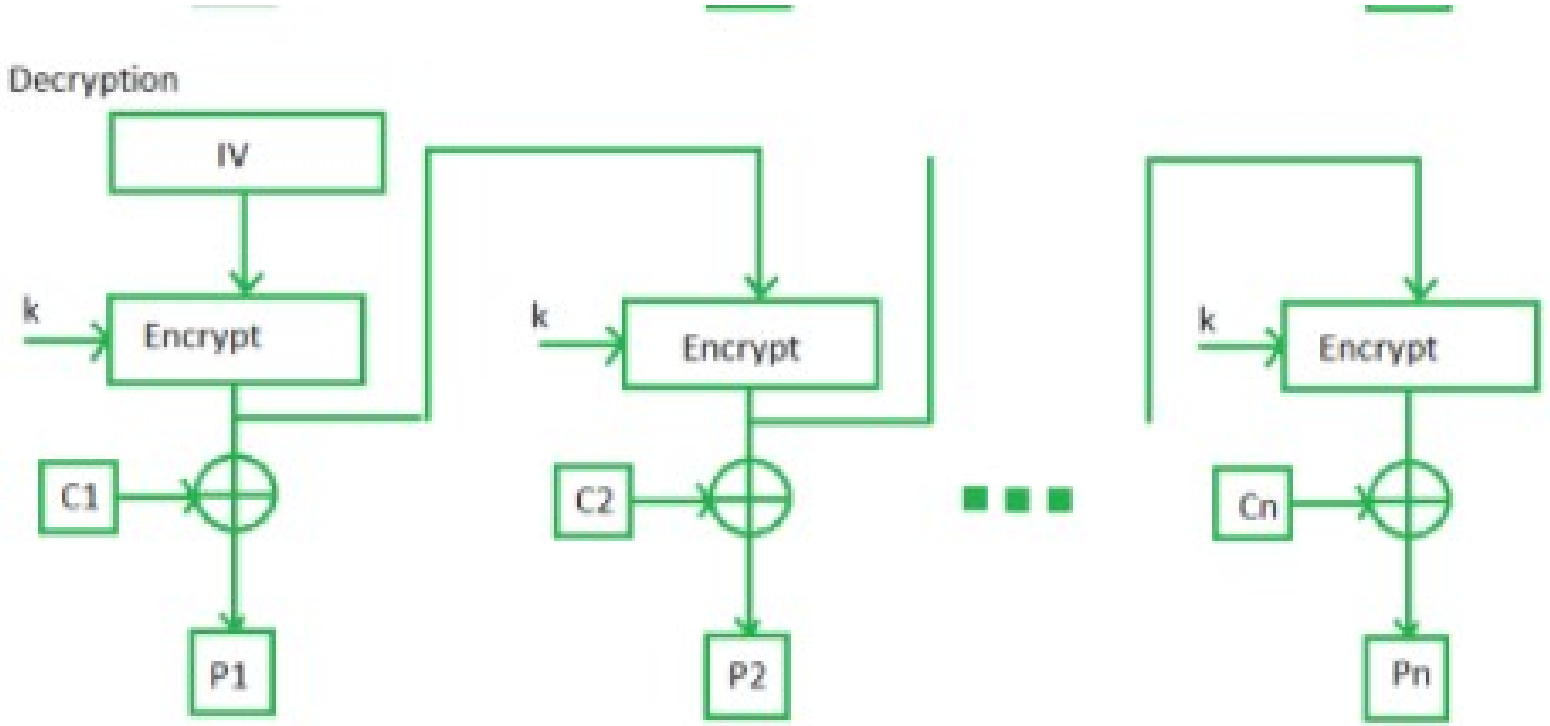
Step 1: The initialization vector is encrypted using the same key used for encrypting all plain text blocks.

Step2: The encrypted IV is then XORed with the ciphertext block to retrieve the plain text block. The encrypted IV is also fed to the decryption process of the next ciphertext block.

The process continues until all the plain text blocks are retrieved.

Output Feedback Mode (OFB)

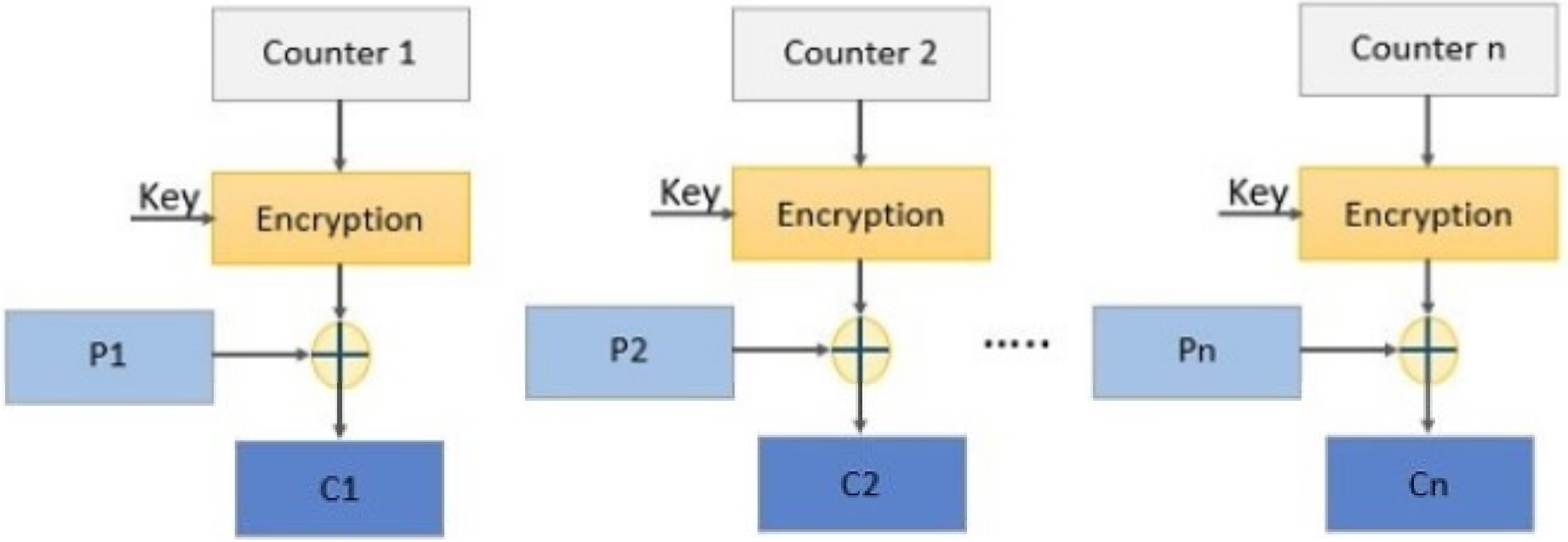
DECRYPTION PROCESS



Counter Mode

1. It is similar to OFB but there is no feedback mechanism in counter mode.
2. Nothing is being fed from the previous step to the next step instead it uses a sequence of number which is termed as a **counter** which is input to the encryption function along with the key.
3. After a plain text block is encrypted the counter value increments by 1.

Counter Mode



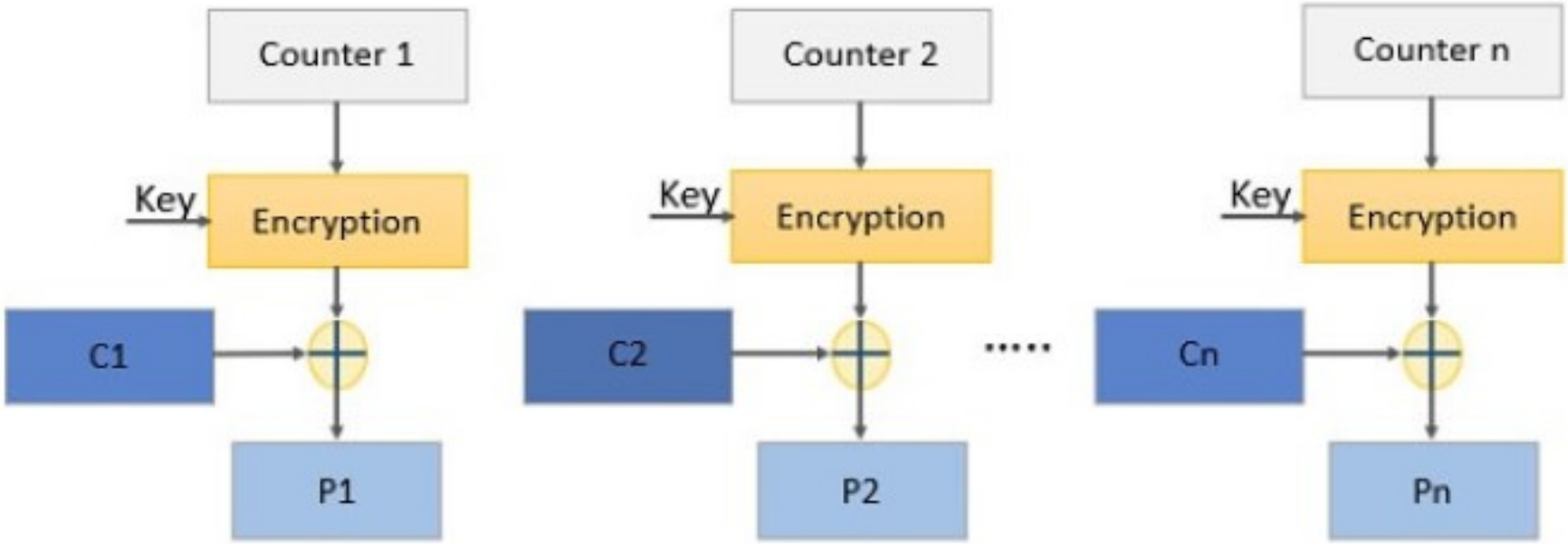
Step1: The counter value is encrypted using a key.

Step 2: The encrypted counter value is XORed with the plain text block to obtain a ciphertext block.

ENCRYPTION

Counter Mode

DECRYPTION PROCESS



Encryption function is used in the decryption process

Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

2.1

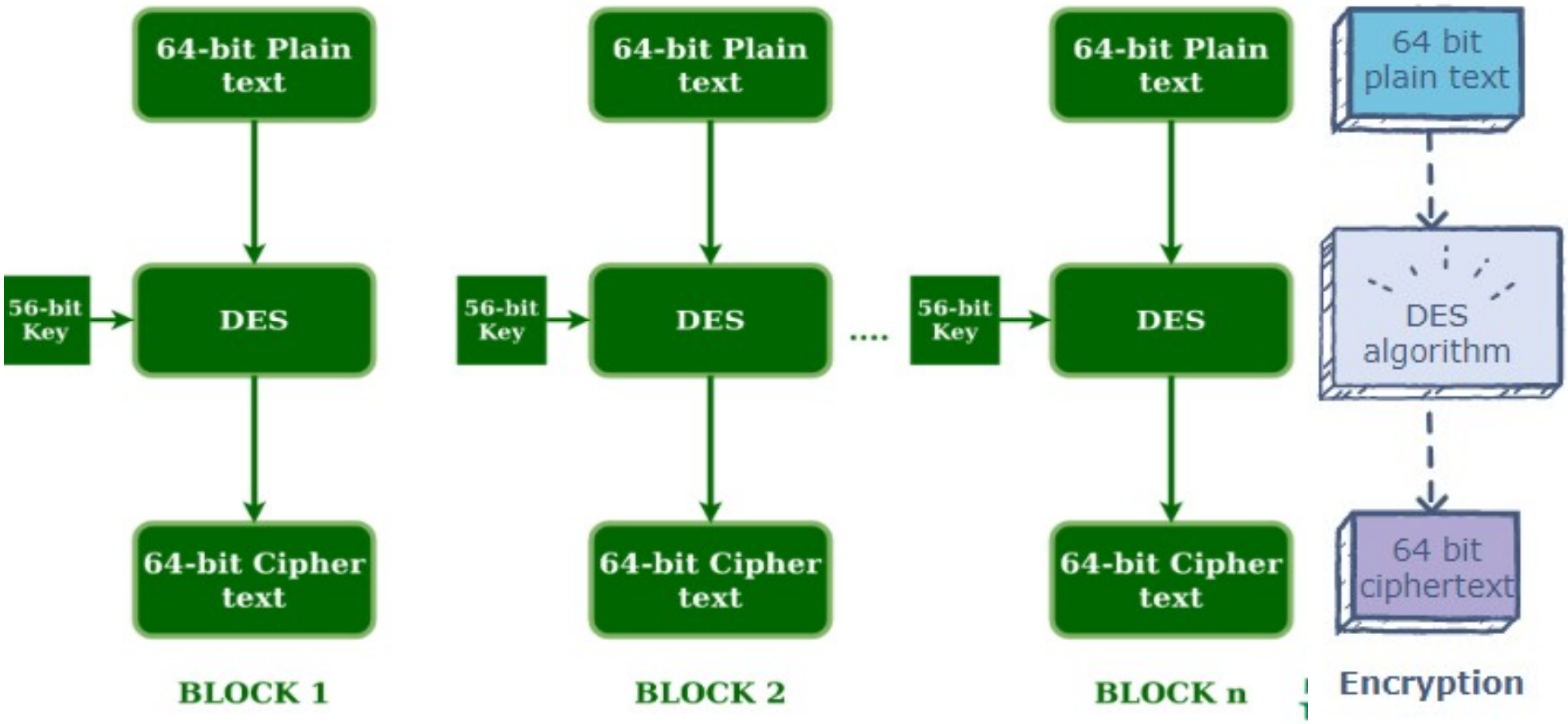
- Block Cipher
  - ✓ **Block Cipher Principles**
  - ✓ **Block Cipher Modes of Operation**
  - ✓ Data Encryption Standard (DES)
  - ✓ Double DES, Triple DES
  - ✓ Advanced Encryption Standard (AES)
- Stream Cipher
  - ✓ RC5 algorithm

Data Encryption Standard (DES)

1. Is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
2. DES encrypts data in blocks of size of 64 bit each.
3. The same algorithm and key are used for encryption and decryption, with minor differences.
4. DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure.
5. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

Data Encryption Standard (DES)

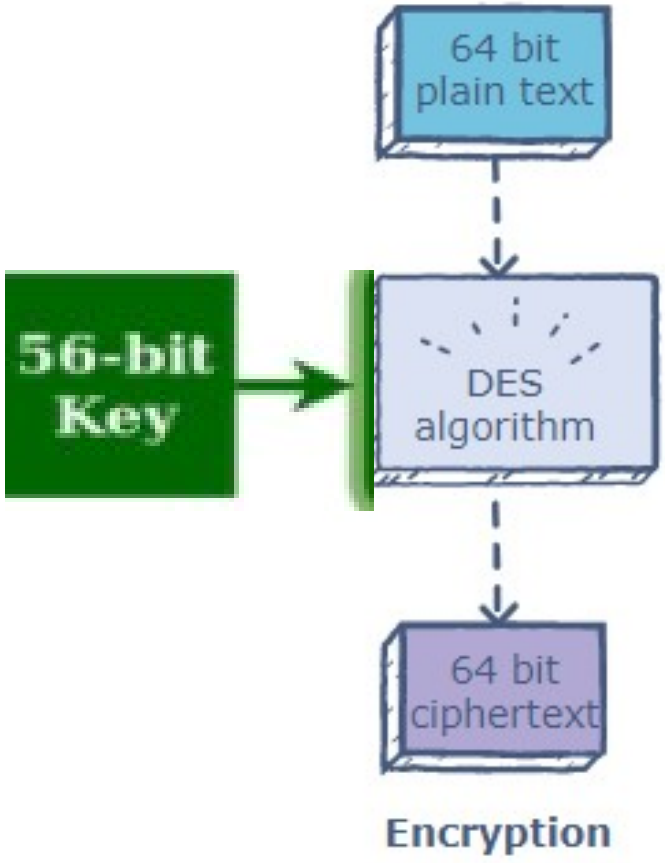
1. The basic idea of DES is shown below:



Data Encryption Standard (DES)

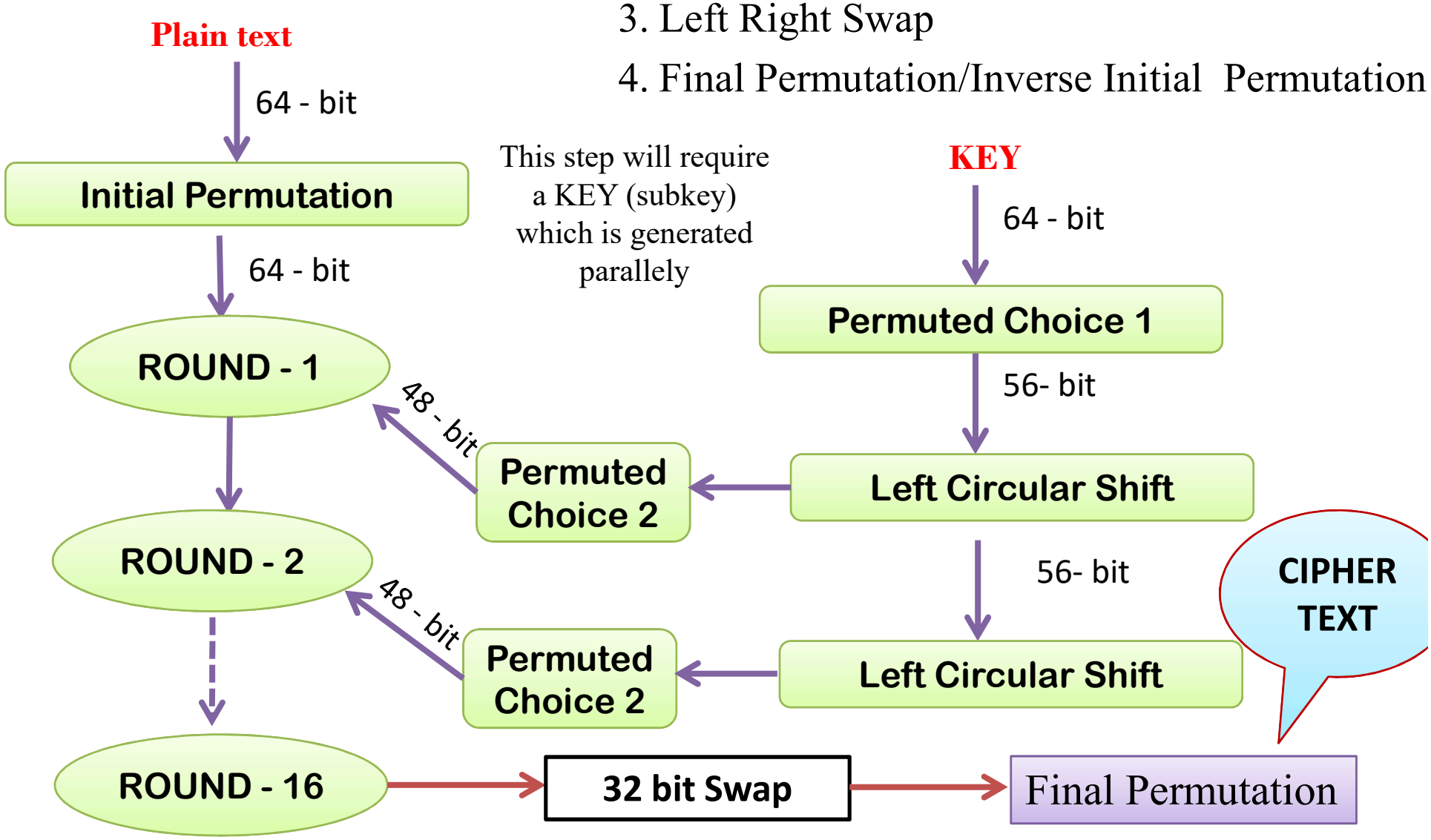
1. The basic idea of DES is shown below:

Even before the DES process starts, every 8th bit of the key is discarded to produce a 56 bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.



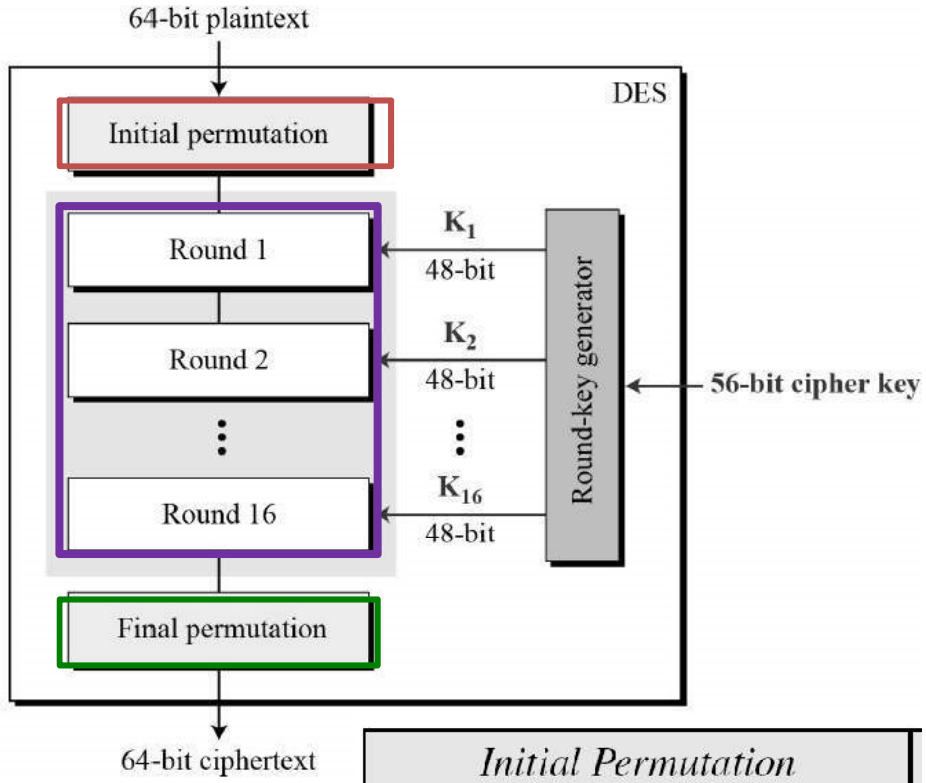
Steps involved in DES

1. Initial Permutation
2. 16 Feistel Rounds
3. Left Right Swap
4. Final Permutation/Inverse Initial Permutation



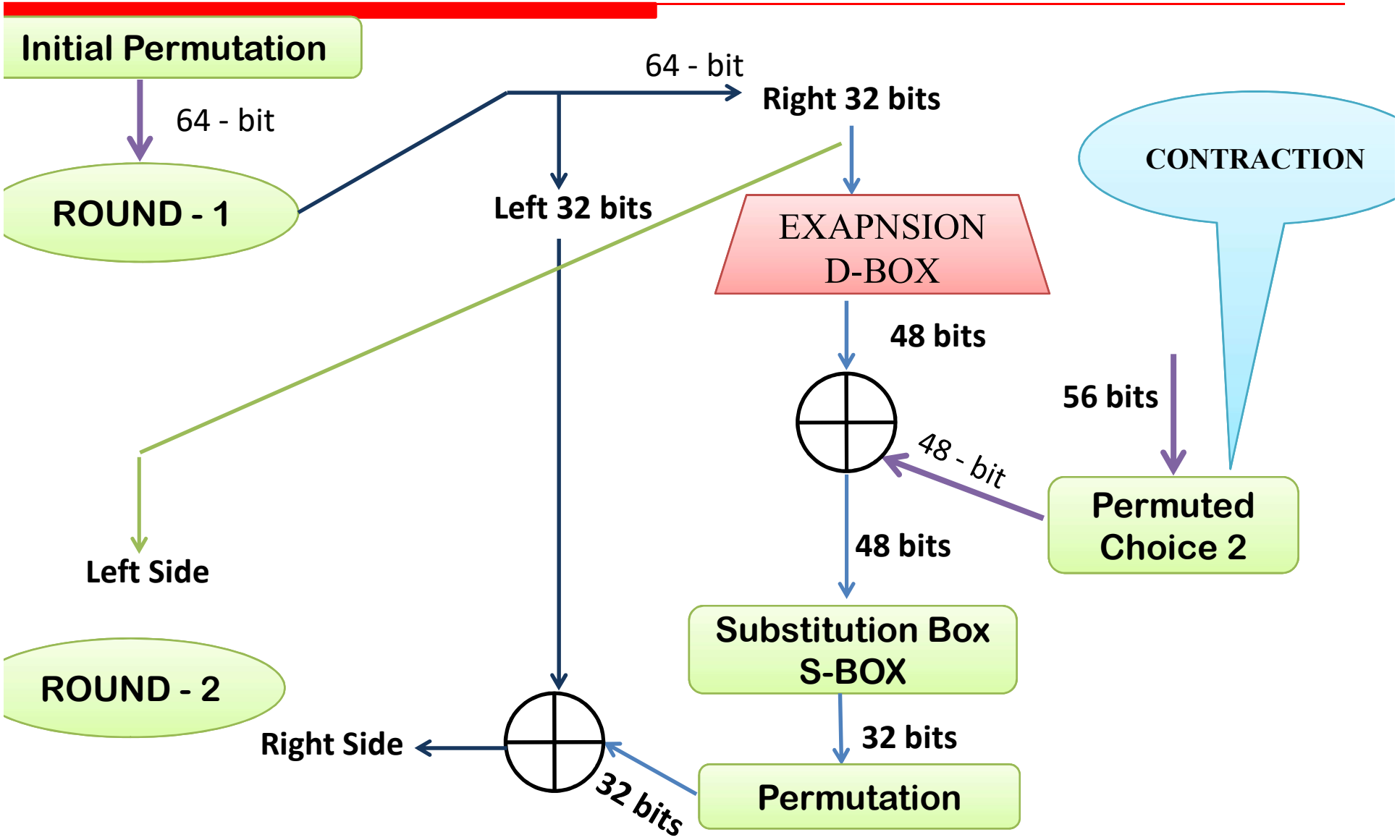
BLOCK DIAGRAM OF DES

1. Permutations are referred as P-BOX.
2. Takes 64-bit input and permutes them according to a predefined rule.
3. These are Keyless Straight permutations that are inverse of each other.



Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

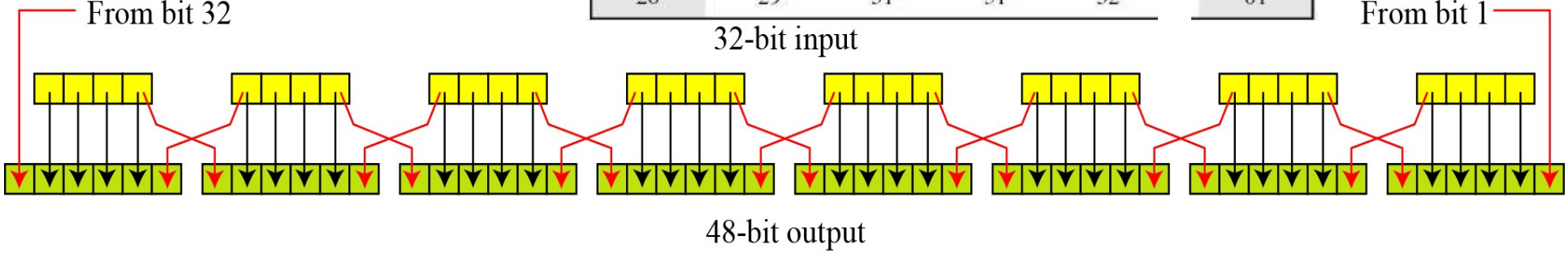
ROUND FUNCTION of DES



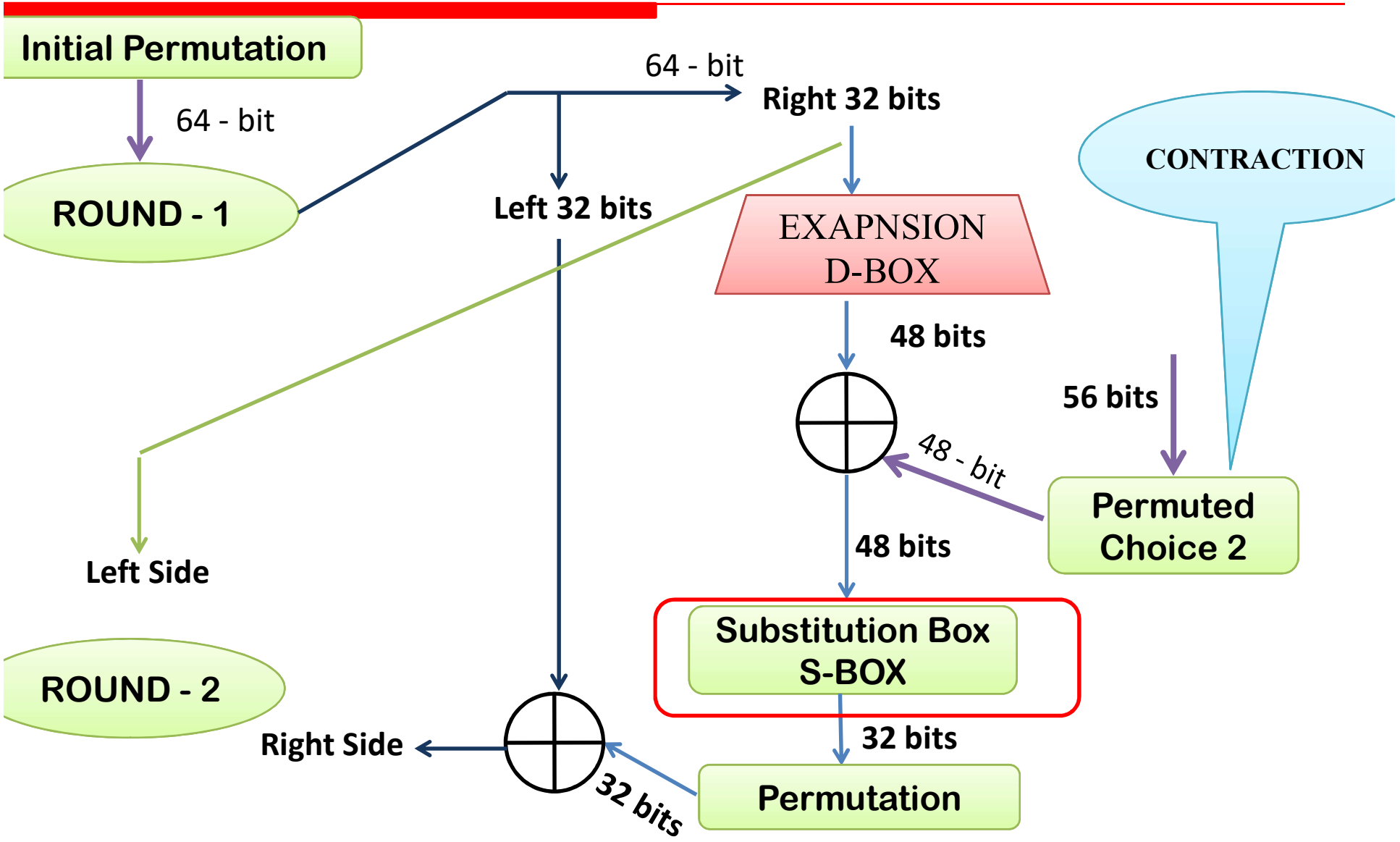
Expansion D – BOX of Round

1. The RIGHT HALF of plaintext (64 bits) contains 32 bits, uses a key of 48-bits.
2. RIGHT HALF needs to be expanded to 48 bits.
3. The 32 – bits of RIGHT HALF into 8, 4-bit sections.
4. Each 4-bit section is then expanded to 6-bits. This expansion permutations follows a predefined rule

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

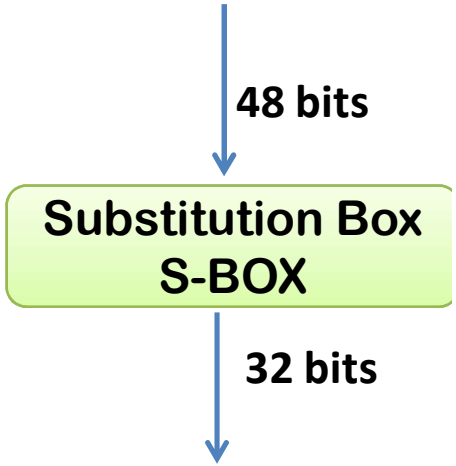


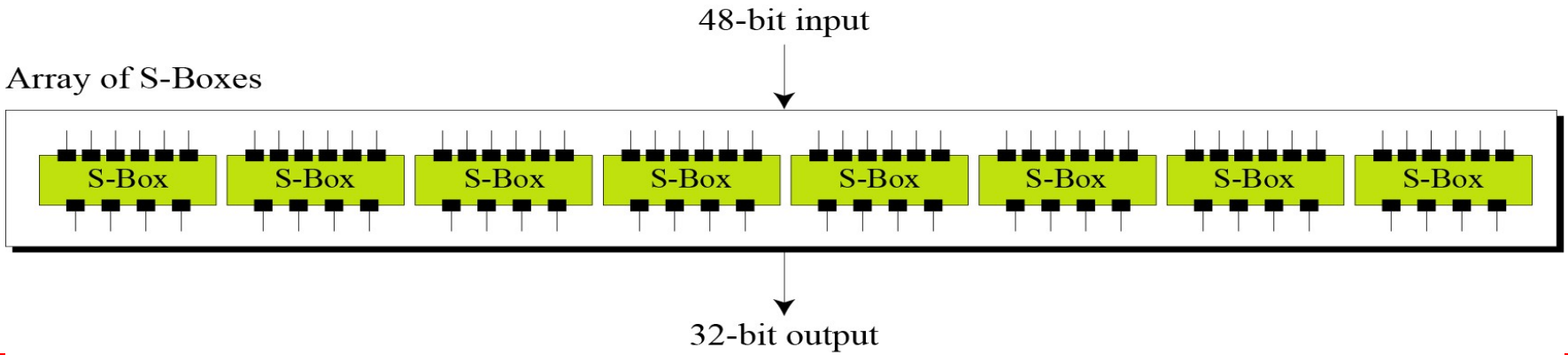
ROUND FUNCTION of DES



SUBSTITUTION BOX (S-BOX)

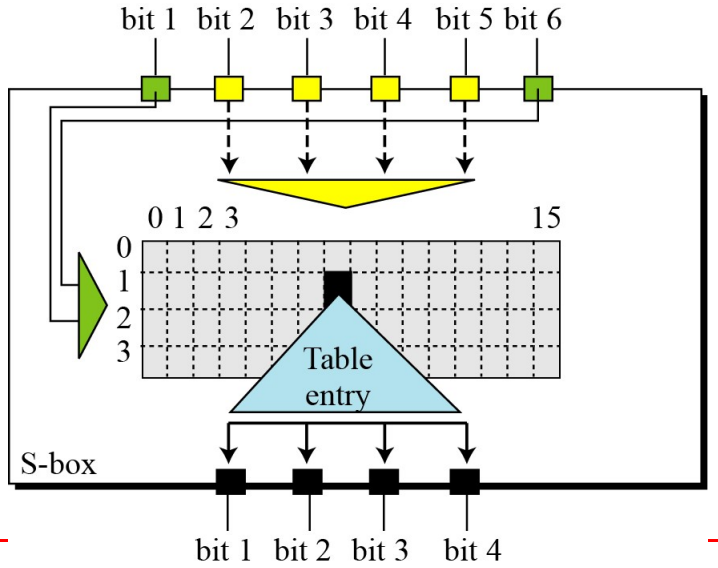
1. Performs the real mixing.
2. DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.
3. 48-bit data is divided into EIGHT 6-bit chunks, and each chunk is fed into a box
4. The result of each box is 4-bit, which when combined the 32-bit result is obtained.





SUBSTITUTION BOX (S-BOX)

1. The substitution in each box follows a predetermined rule based on 4-ROW and 16-COLUMN table.
2. The combination of 1st and 6th bit of the input defines one of the 4 ROWS
3. The combination of bits 2nd through 5th of the input defines one of the 16 COLUMNS
4. Because each S-box has its own table, we will need 8 tables



SUBSTITUTION BOX (S-BOX)

The following table shows the permutation for S-box 1.

S-box 1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

For the rest of the boxes see the textbook.

The input to S-box 1 is 100011. What is the output?

S-box 1

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

1 00011 1

11 – 03
0001 - 01

Solution

The input to S-box 1 is 001100. What is the output?

S-box 1

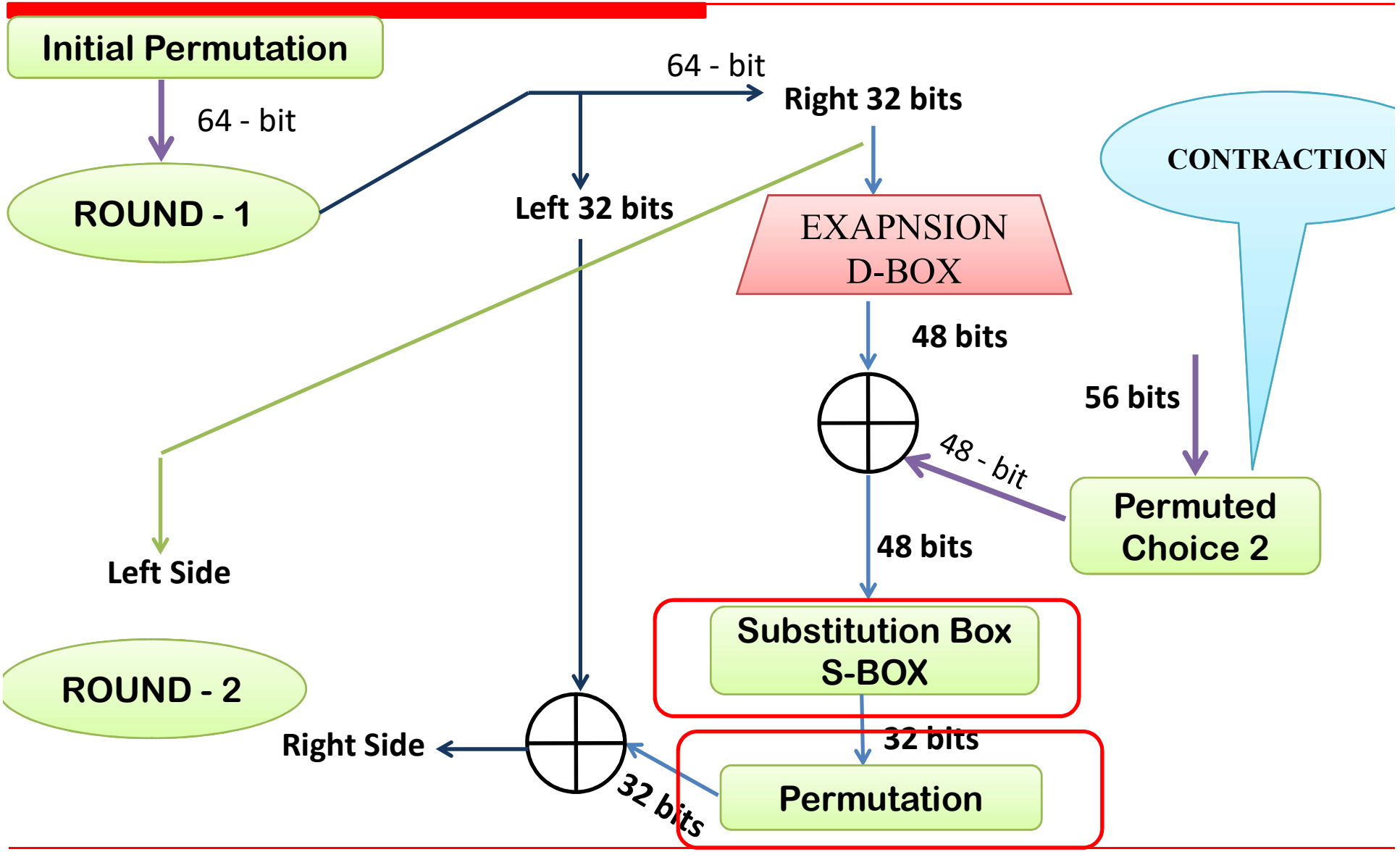
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

0 01100 0

00 – 00
0110 - 06

Solution

ROUND FUNCTION of DES



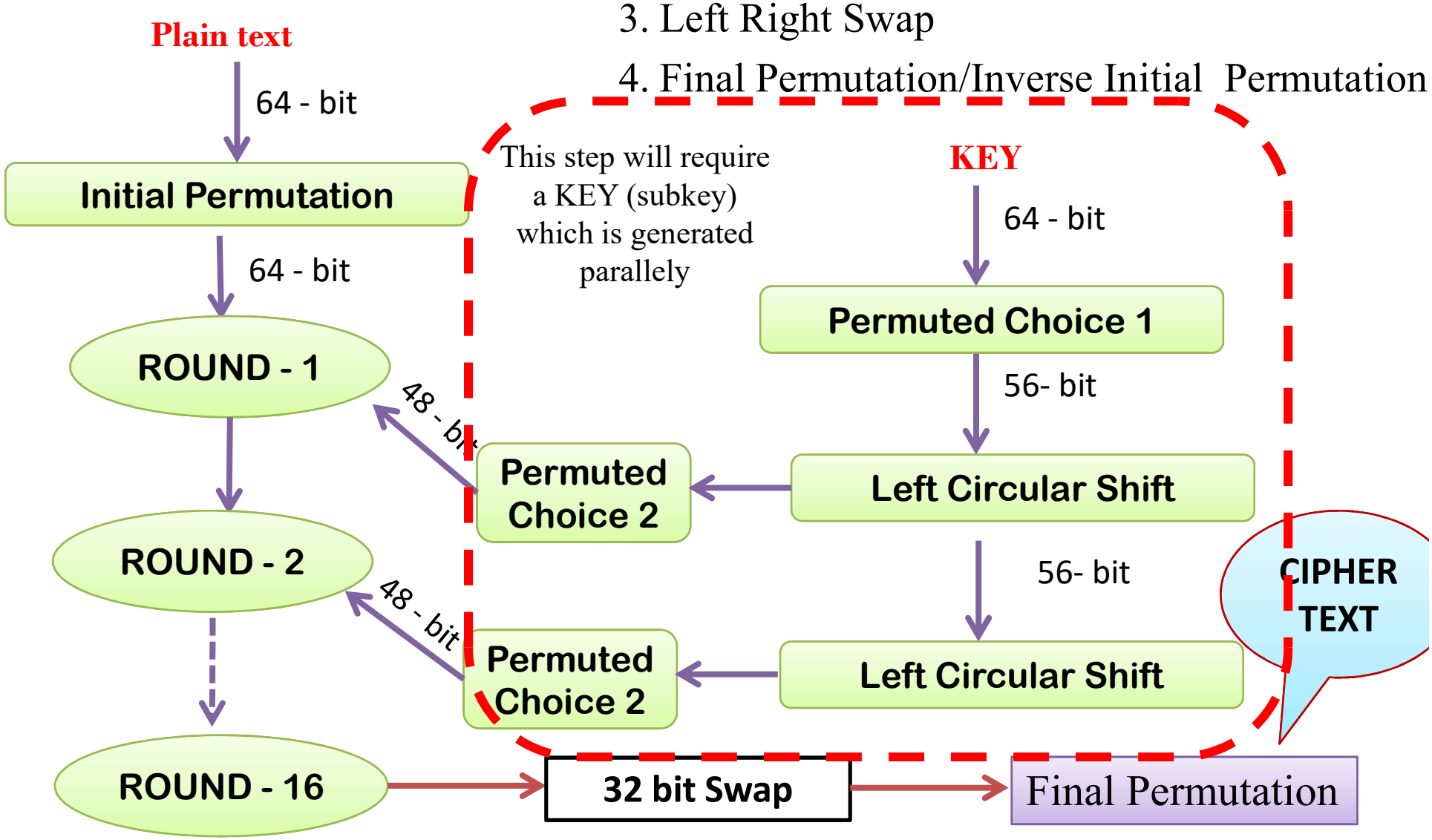
ROUND FUNCTION of DES

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Last Operation

Steps involved in DES

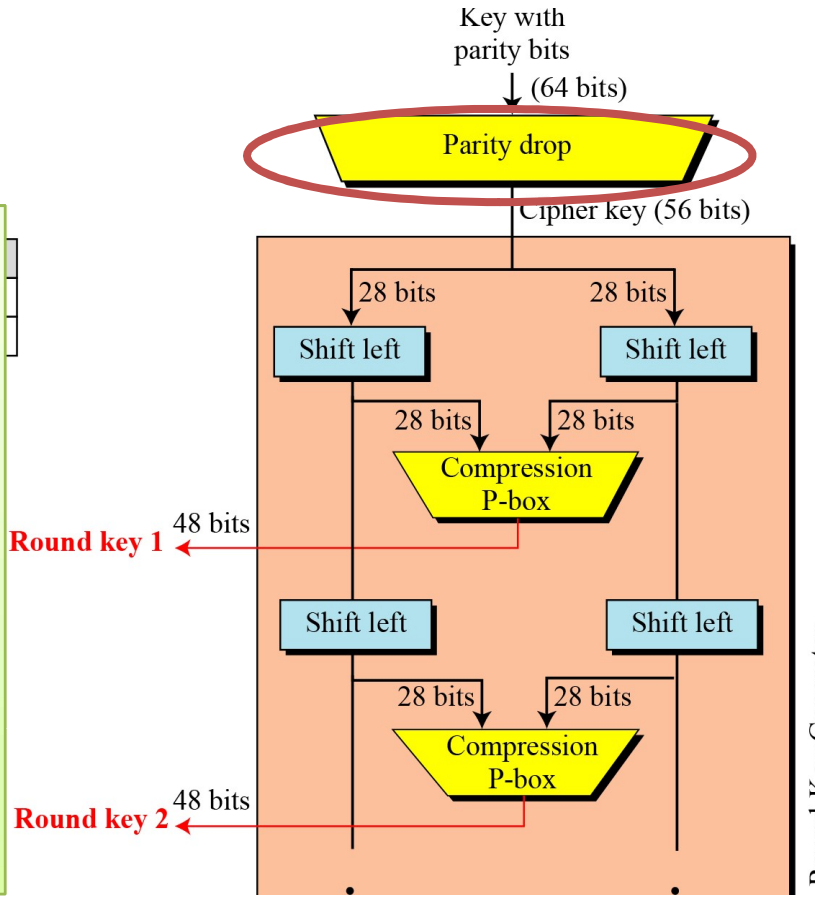
1. Initial Permutation
2. 16 Feistel Rounds
3. Left Right Swap
4. Final Permutation/Inverse Initial Permutation



KEY GENERATION

PARITY DROP:

- The preprocess before key expansion is a compression transposition step
- It drop parity bits (8, 16, 24, 32, … 64)
- Permutes the rest of the bits according to following Parity Drop Table.
- The remaining 56-bit value is the actual CIPHER KEY which is used to generate rounds.



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

Permuted Choice - 1

KEY GENERATION

- These 56-bits are again divided into Left and Right Half (28-bits each).
- Left Circular Shift is performed on each half
- In Round 1, 2, 9 and 16, shifting is 1 bit
- In the remaining Rounds it is 2-bits

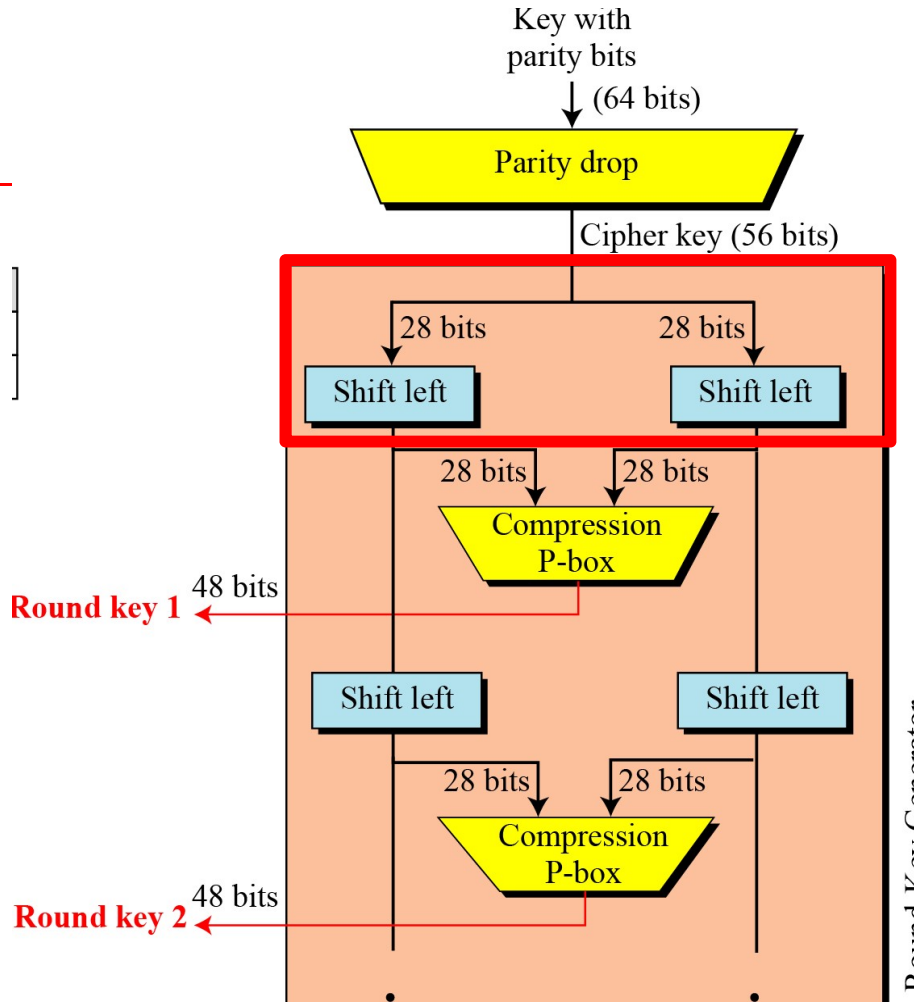


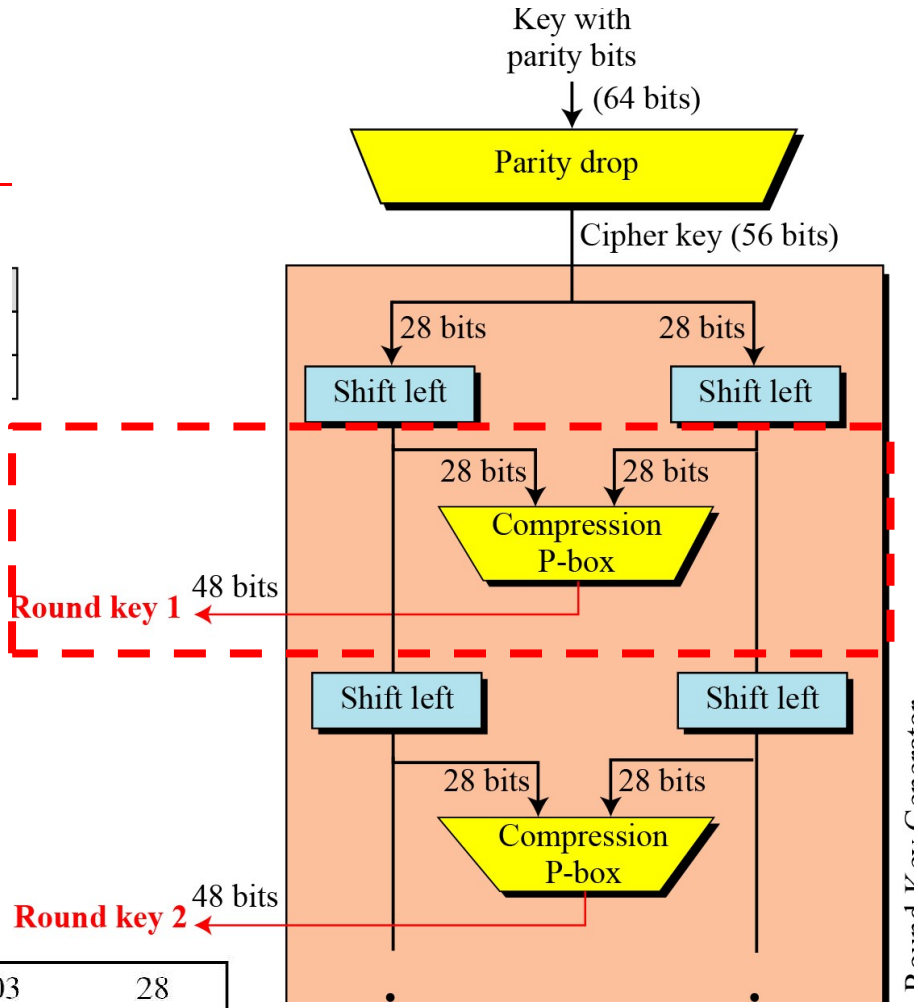
Table 6.13 *Number of bits shifts*

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

KEY GENERATION

COMPRESSION D-Box

The Compression D-box changes the 56-bits to 48-bits which are used as a key for a round.



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

DES ANALYSIS

1. *Critics have used a strong magnifier to analyze DES.*
2. *Tests have been done to measure the strength of some desired properties in a block cipher.*

1. **Properties:** **Two desired properties of a block cipher are the**

AVALANCHE EFFECT and THE COMPLETENESS

2. **Design Criteria:** **The design of DES was revealed by IBM in 1994. Many test on DES have proved that it satisifes some of the required criteria as claimed.**

3. **DES Weaknesses:** **Weakness in Cipher Design and in Cipher Key**

PROPERTIES

Avalanche effect *means a small change in plaintext(or key) should create a significant change in cipher text*

Example

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

Completeness effect

Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext

MULTIPLE DES

1. Because of its vulnerability to brute-force attack, DES, once the most widely used symmetric cipher, has been largely replaced by stronger encryption schemes.
2. Two approaches have been taken.
3. One approach is to design a completely new algorithm that is resistant to both cryptanalytic and brute-force attacks, of which AES is a prime example.
4. Another alternative, which preserves the existing investment in software and equipment, is to use multiple encryption with DES and multiple keys.

    1. **Double DES (2DES)**
    2. **Triple DES (3DES)**

DOUBLE DES (2DES)

1. The simplest form of multiple encryption has two encryption stages and two keys
2. Given a plaintext P and two encryption keys K1 and K, ciphertext C is generated as

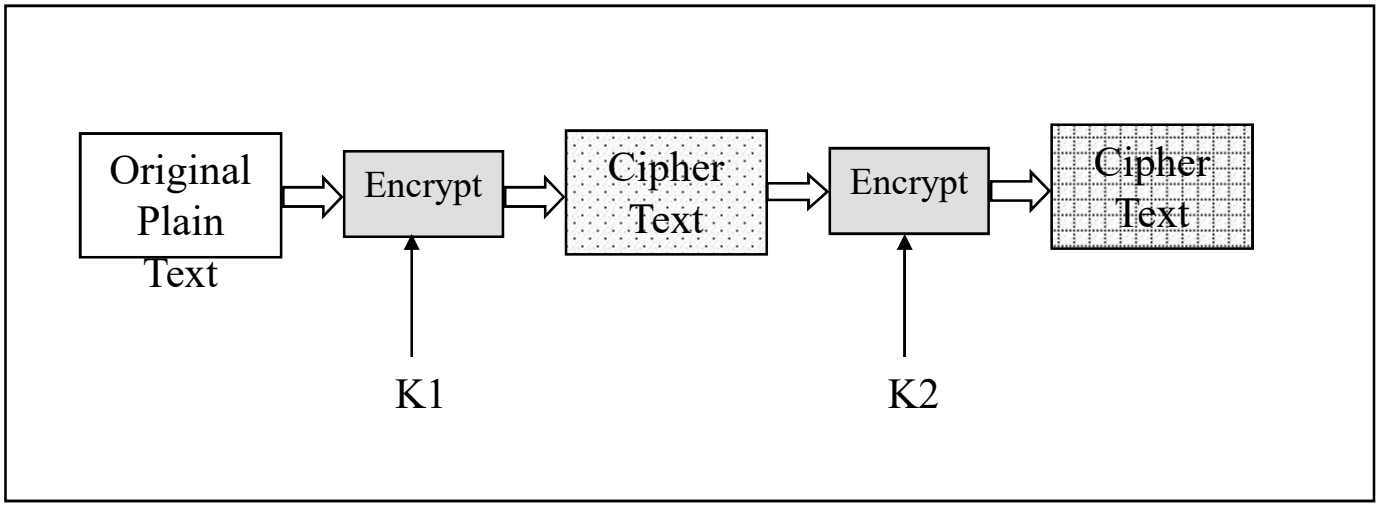
$$P \longrightarrow E(K_1, P) \qquad C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$$

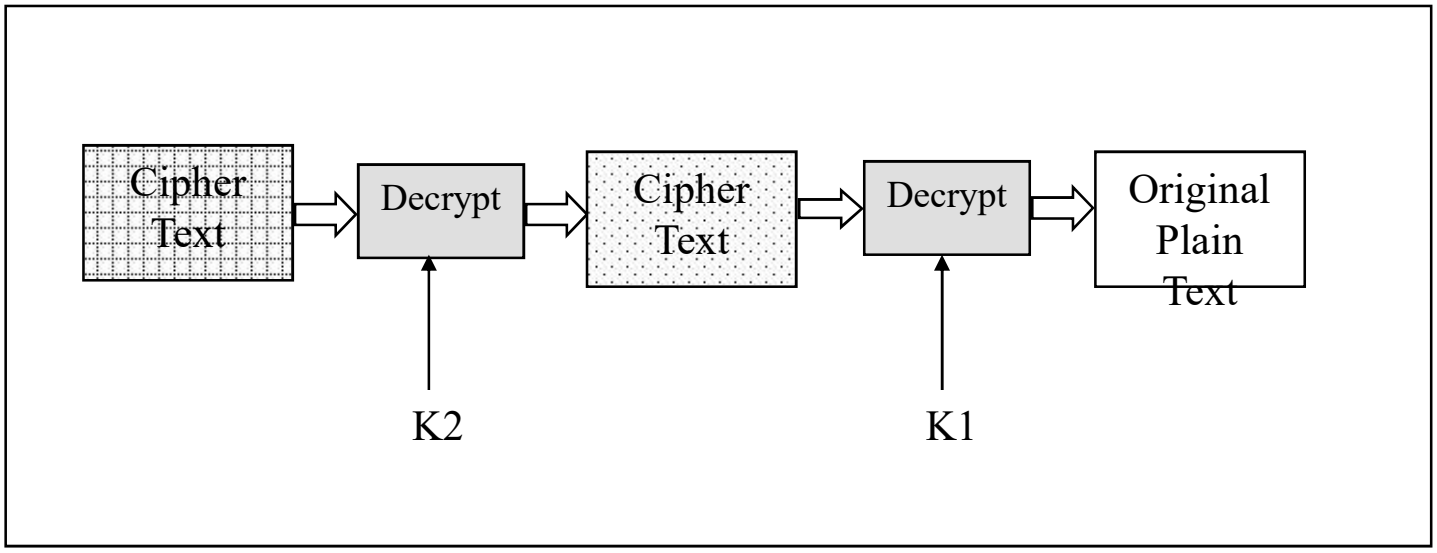
Decryption requires that the keys be applied in reverse order:

$$\text{Plaintext} = D(K1, D(K2, C))$$

For DES, this scheme apparently involves a key length of **56 * 2 = 112 bits**

DOUBLE DES (2DES)





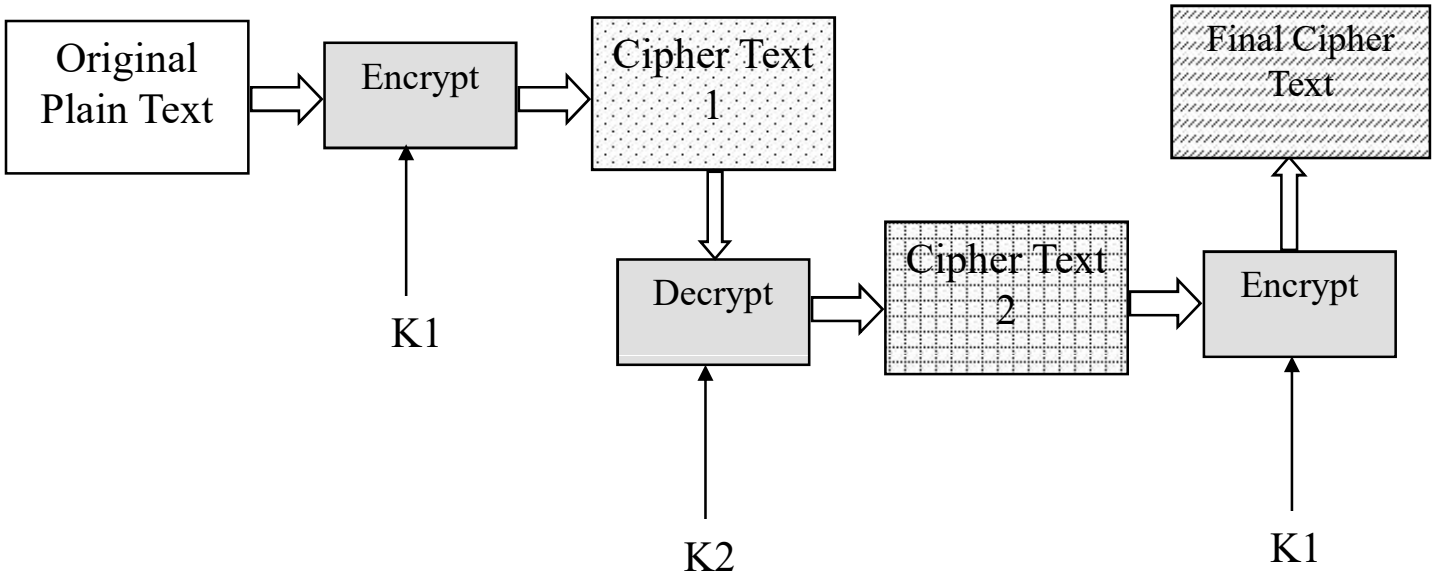
TRPLE DES (3DES)

1. Double DES was prone to MEET-IN-THE-MIDDLE-ATTACK
2. An obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys.
3. Using DES as the underlying algorithm, this approach is commonly referred to as 3DES, or Triple Data Encryption Standard

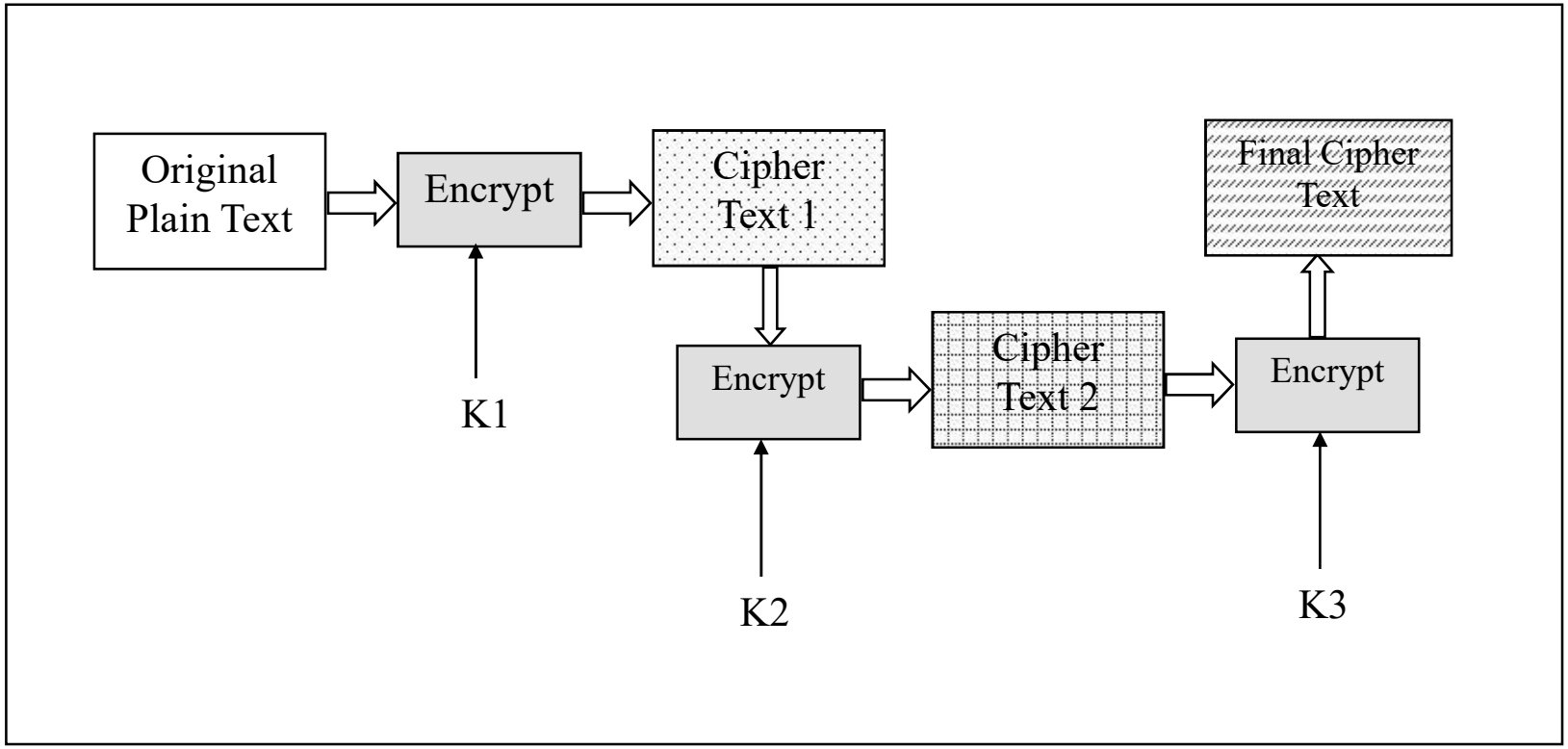
Triple DES with Three Different Keys

Triple DES with Two Different Keys

Triple DES with Two Different Keys



Triple DES with Three Different Keys



Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

2.1

- Block Cipher
  - ✓ **Block Cipher Principles**
  - ✓ **Block Cipher Modes of Operation**
  - ✓ **Data Encryption Standard (DES)**
  - ✓ **Double DES, Triple DES**
  - ✓ Advanced Encryption Standard (AES)
- Stream Cipher
  - ✓ RC5 algorithm

Advanced Encryption Standard (AES)

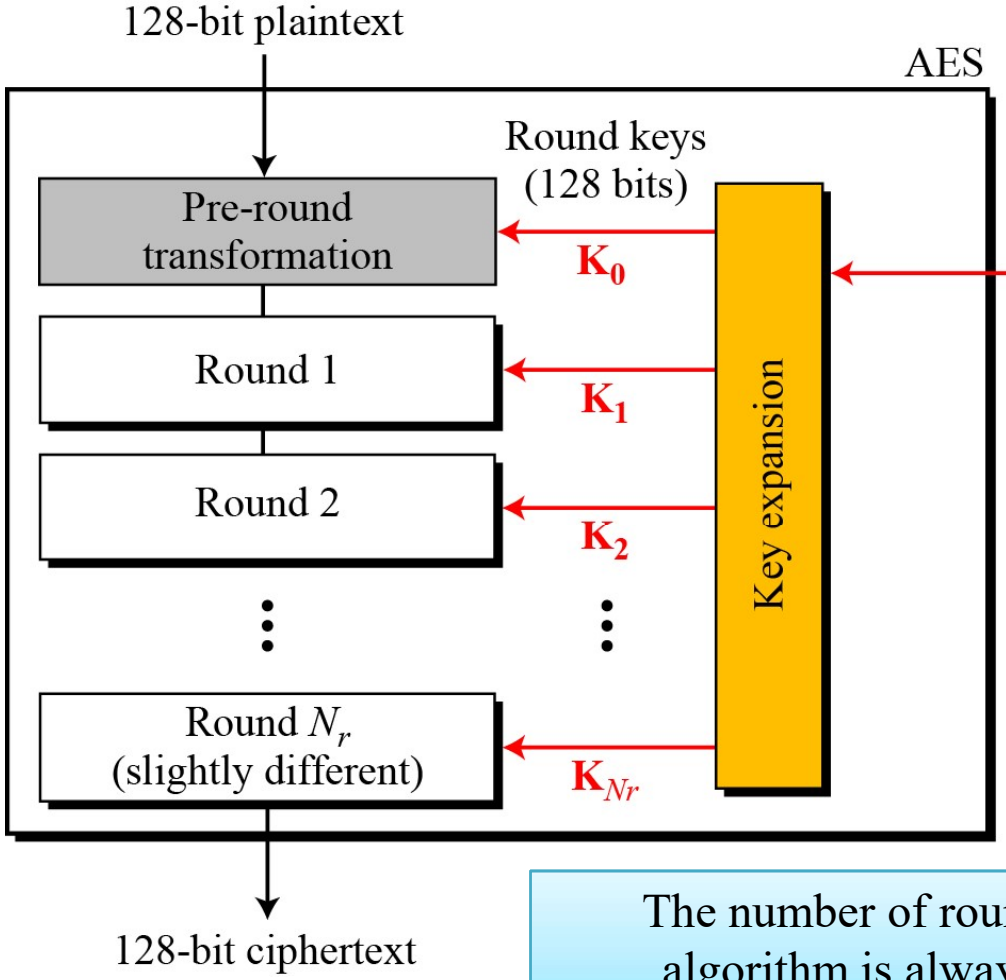
1. Is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). Proposed by **Rijndael**
2. AES encrypts data in blocks of size of 128 bit each.
3. AES is a NON - Feistel Cipher. It uses 10 rounds
4. The key size is 128 bits. Here, the key is processed in terms of **WORDS**
5. 128 bits are processed in 4 words or 16 bytes

1 WORD = 32 bits
1 byte = 8 bits

Note

AES has defined three versions, with 10, 12, and 14 rounds.
Each version uses a different cipher key size (128, 192, or 256),
BUT the round keys are always 128 bits.

Advanced Encryption Standard (AES)



Cipher key
(128, 192, or 256 bits)

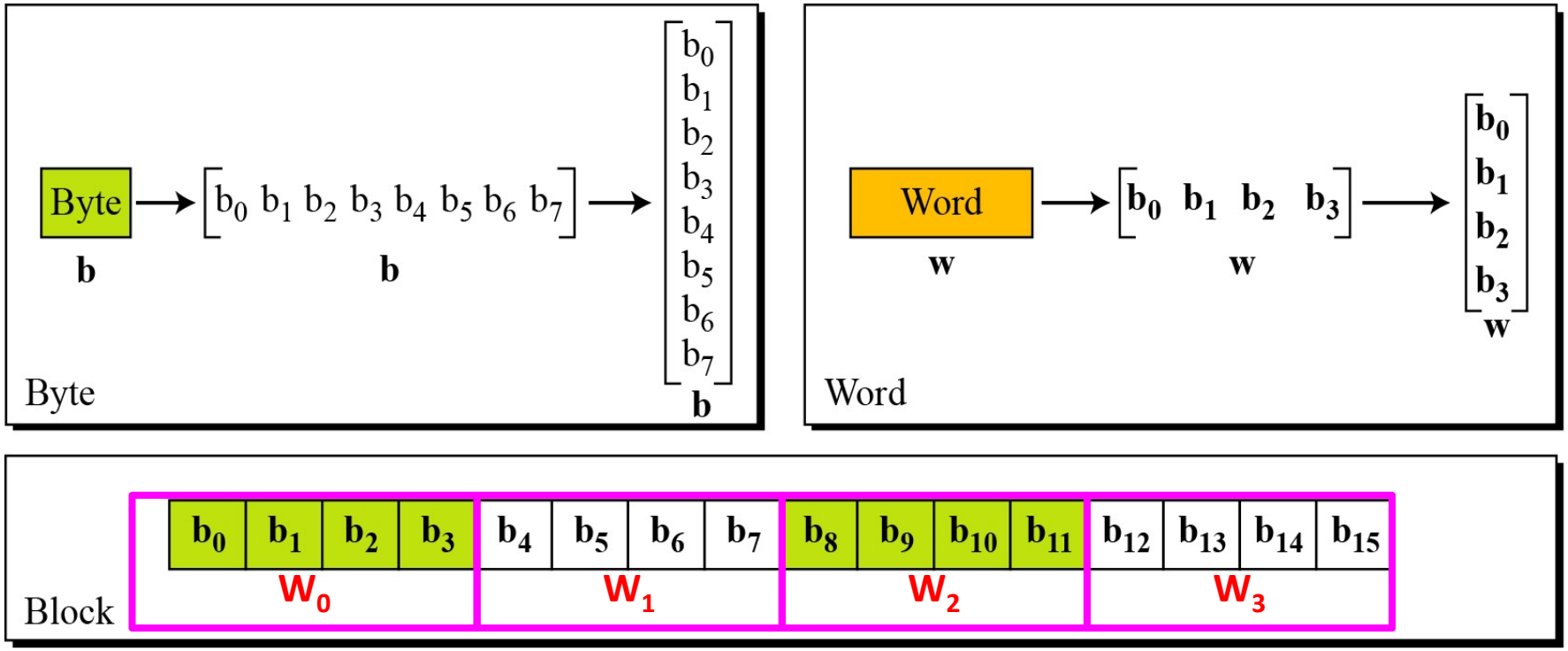
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

The number of round keys generated by the key expansion algorithm is always one more than the number of rounds

Number of Round Keys = $N_r + 1$

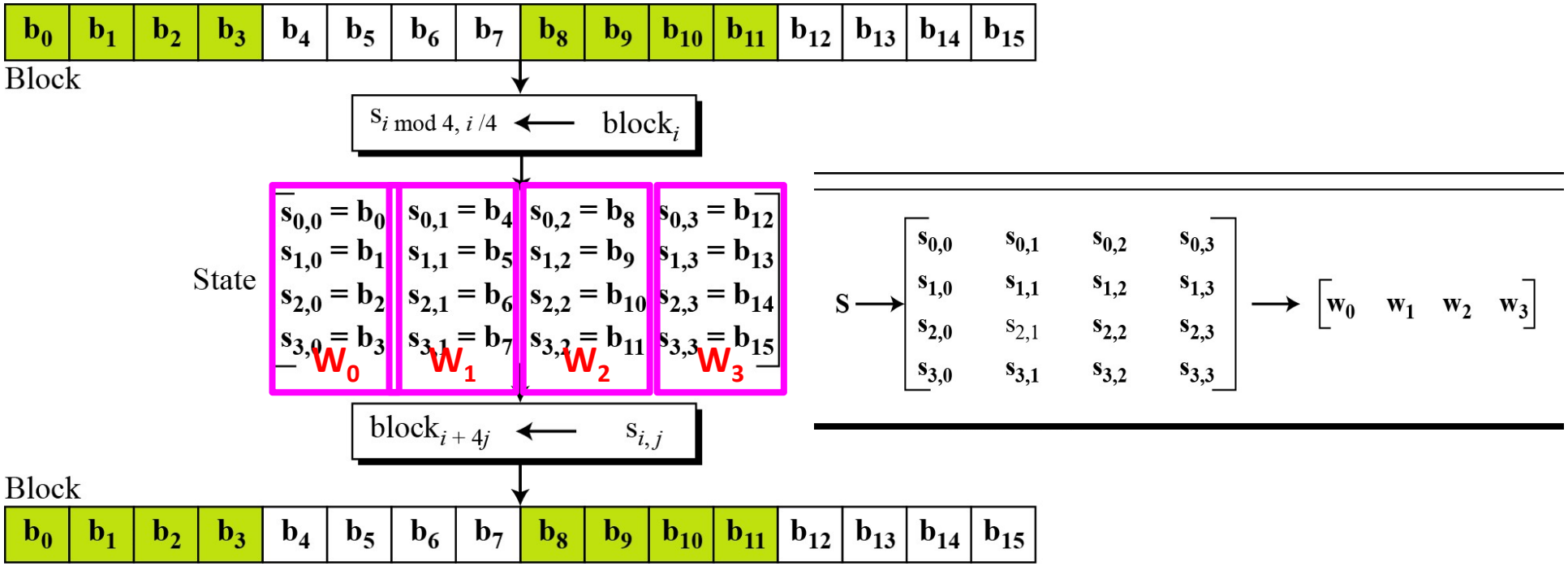
AES data units



AES data units

State

1. Data Block is referred to as a STATE.
2. States are made of 16 bytes, but normally are treated as matrices of 4 * 4 bytes
3. Each element is referred to as $s_{r,c}$ where r(0 to 3) defines the row and c(0 to 3) defines the column



Example *Changing plaintext to state*

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19
	W_0															

00	12	0C	08
04	04	00	23
12	12	13	19
14	00	11	19

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
W_0				W_1				W_2				W_3			

	DEC	HEX		DEC	HEX
A	00	00	N	13	0D
B	01	01	O	14	0E
C	02	02	P	15	0F
D	03	03	Q	16	10
E	04	04	R	17	11
F	05	05	S	18	12
G	06	06	T	19	13
H	07	07	U	20	14
I	08	08	V	21	15
J	09	09	W	22	16
K	10	0A	X	23	17
L	11	0B	Y	24	18
M	12	0C	Z	25	19

Advanced Encryption Standard (AES)

Plaintext (128 – bits)

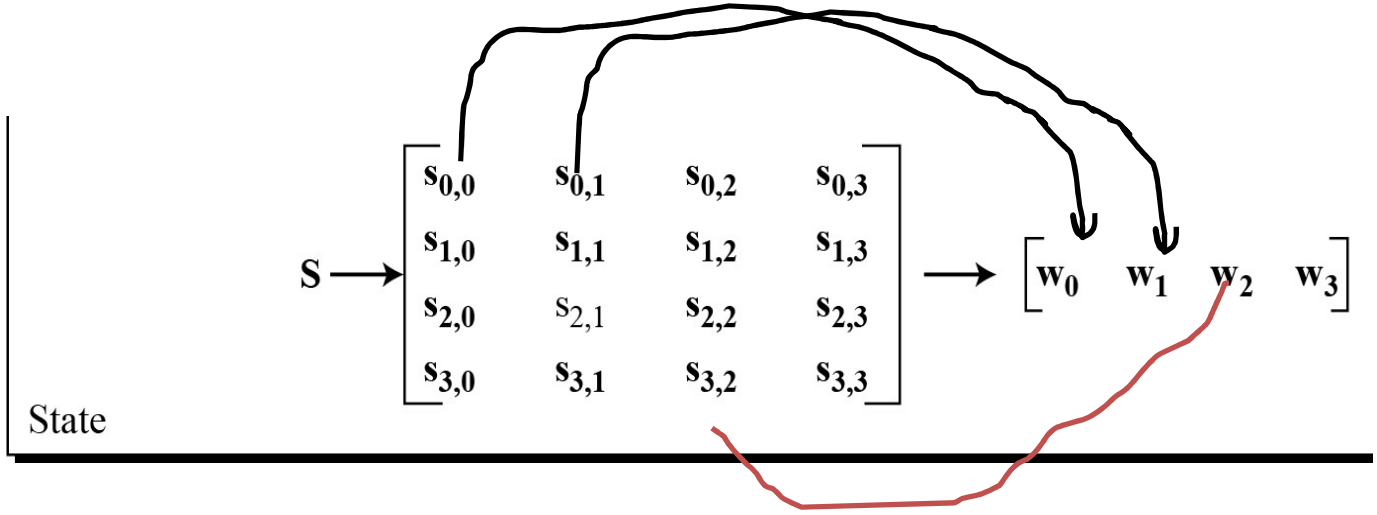
IN_0	IN_4	IN_8	IN_{12}
IN_1	IN_5	IN_9	IN_{13}
IN_2	IN_6	IN_{10}	IN_{14}
IN_3	IN_7	IN_{11}	IN_{15}

Is represented as Input Array (4 * 4) each cell has 8 bits data i.e. 1 byte

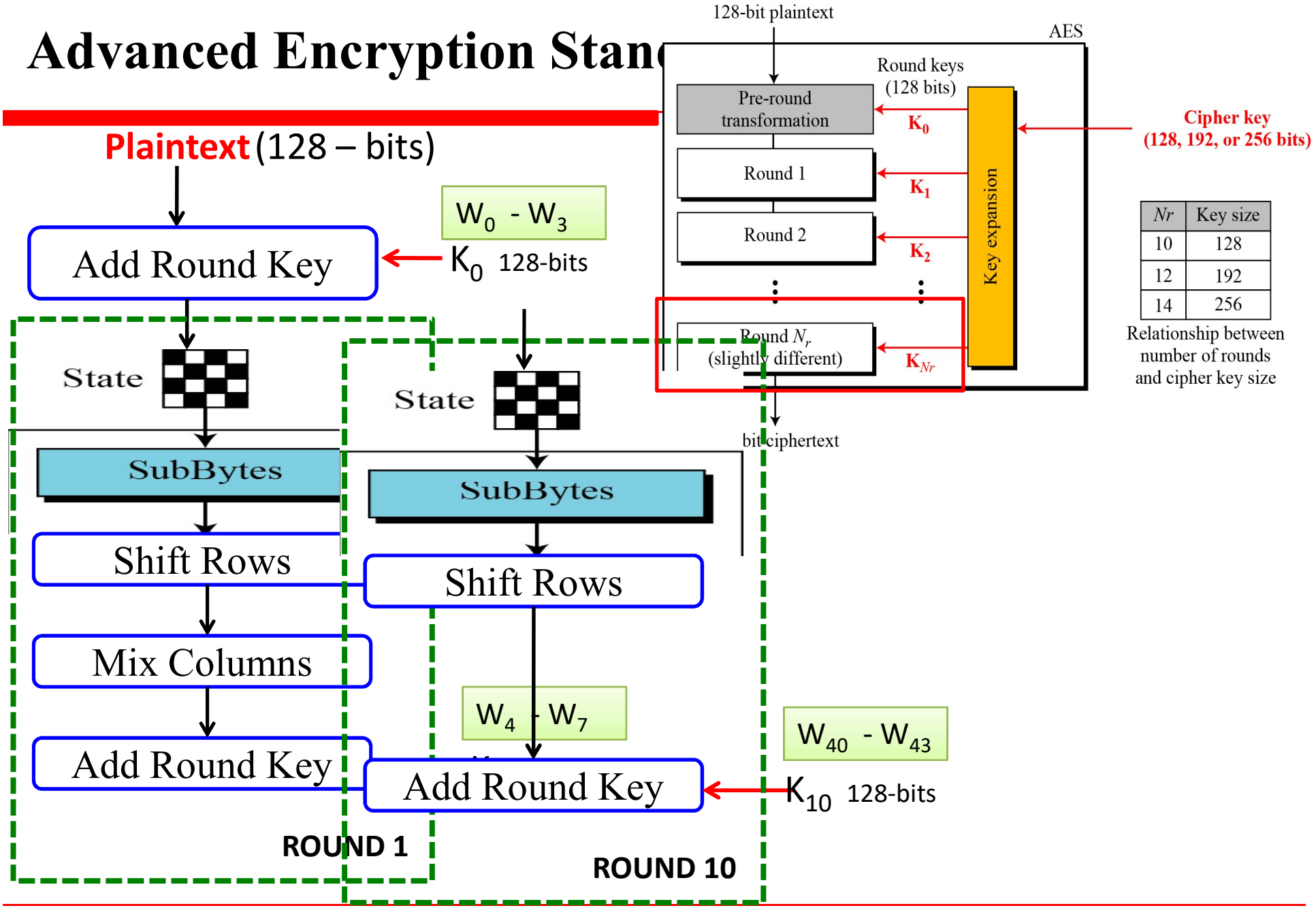
KEY (128 – bits)

KEY_0	KEY_4	KEY_8	KEY_{12}
KEY_1	KEY_5	KEY_9	KEY_{13}
KEY_2	KEY_6	KEY_{10}	KEY_{14}
KEY_3	KEY_7	KEY_{11}	KEY_{15}

Intermediate Results are saved in STATE ARRAY



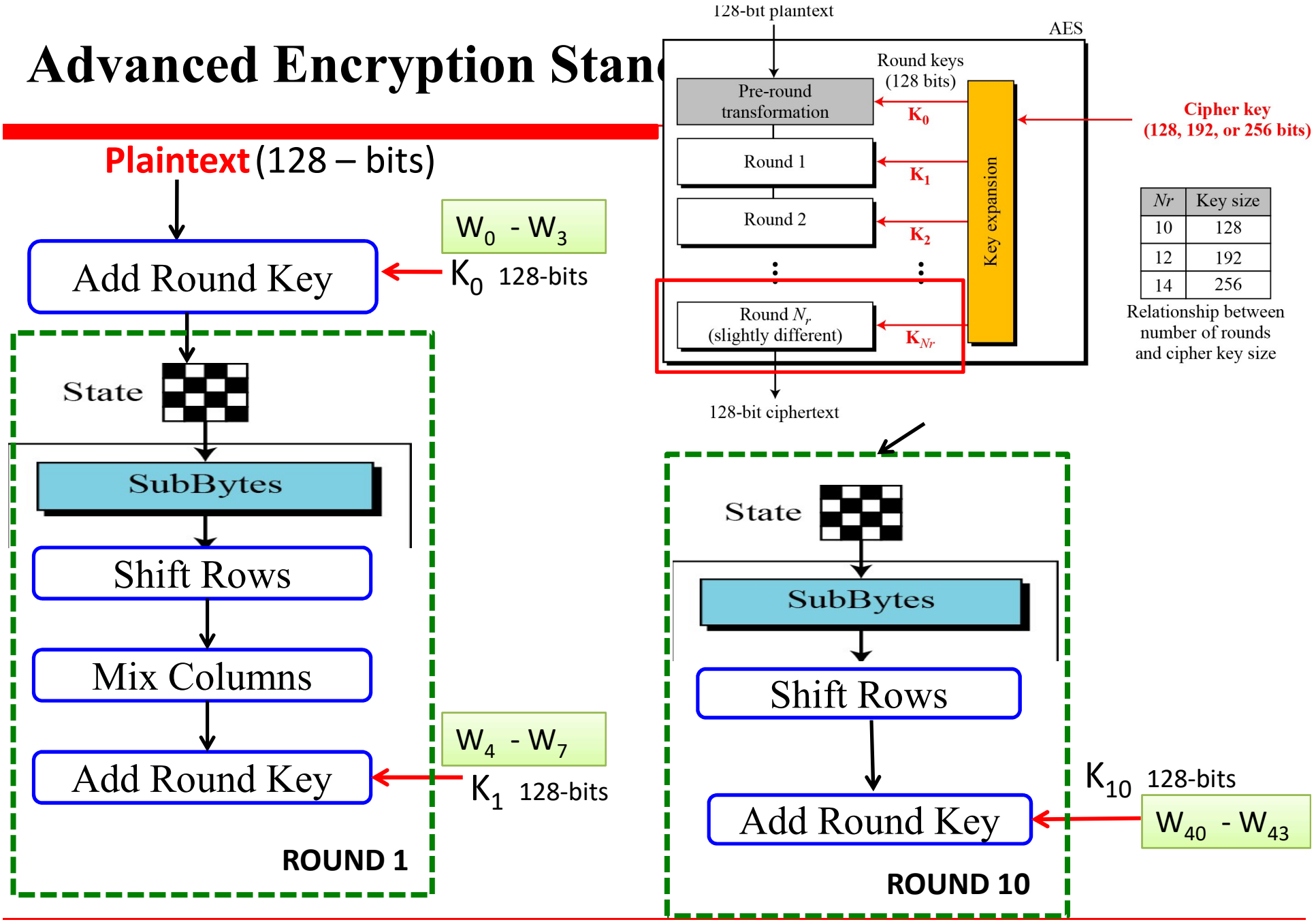
Advanced Encryption Stan



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Advanced Encryption Stan



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Advanced Encryption Standard (AES)

1. Each transformation takes a state and creates another state to be used for the next round or transformation
2. The pre-round section uses only one transformation (AddRoundKey)
3. The last round uses only 3 transformation (MisColumns transformation is missing)
4. At DECRYPTION side, the Inverse Transformations are used - InvSubByte, InvShiftRows, InvMixColumns and AddRoundKey (self-invertble)

SUBSTITUTION	PERMUTATION	MIXING	KEY ADDIING

Advanced Encryption Standard (AES)

SUBSTITUTION

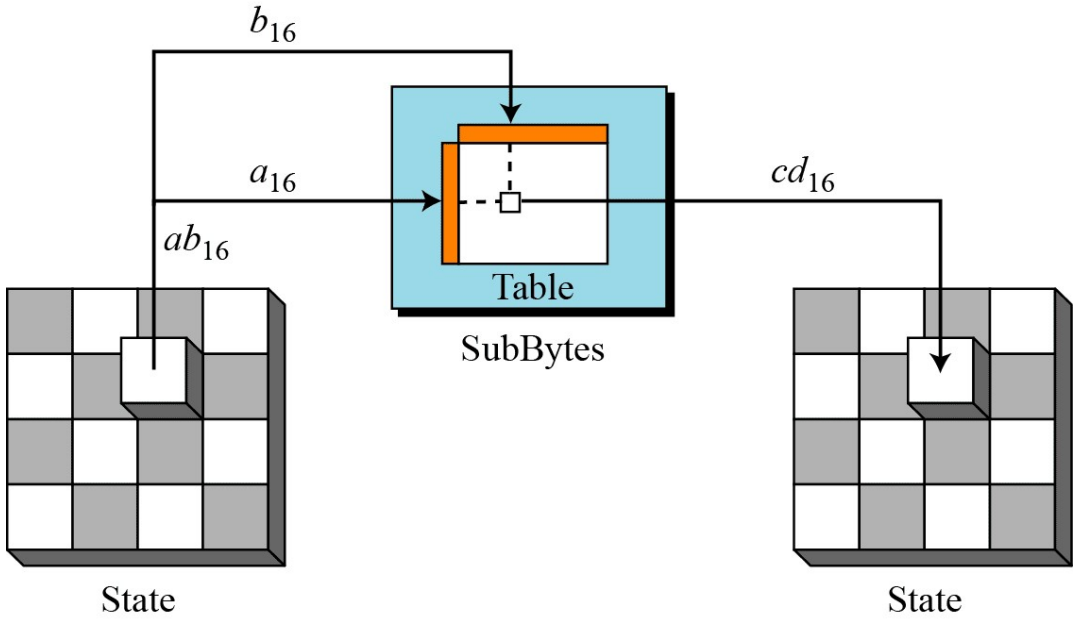
AES uses two invertible transformations.

SubBytes

InvSubBytes

SubBytes

The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.



58

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \qquad \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}$$

TABLE 3.4. AES ByteSub.

	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
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

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
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

Advanced Encryption Standard (AES)

SUBSTITUTION

PERMUTATION

MIXING

KEY ADDIING

Advanced Encryption Standard (AES)

PERMUTATION

Another transformation found in a round is SHIFTING.

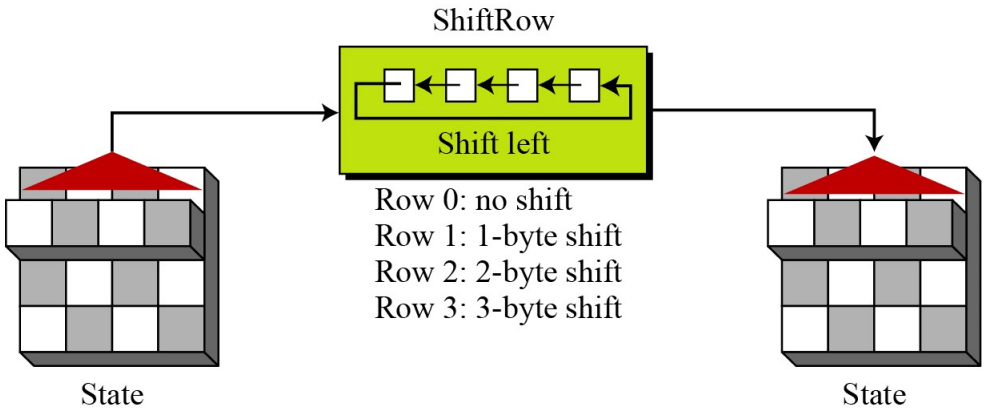
ShiftRows

InvShiftRows

ShiftRows

The shifting is done to left.

The number of shifts depends on the row number of the state matrix



InvShiftRows

In decryption, the transformation is **InvShiftRows**

Shifting is done to RIGHT.

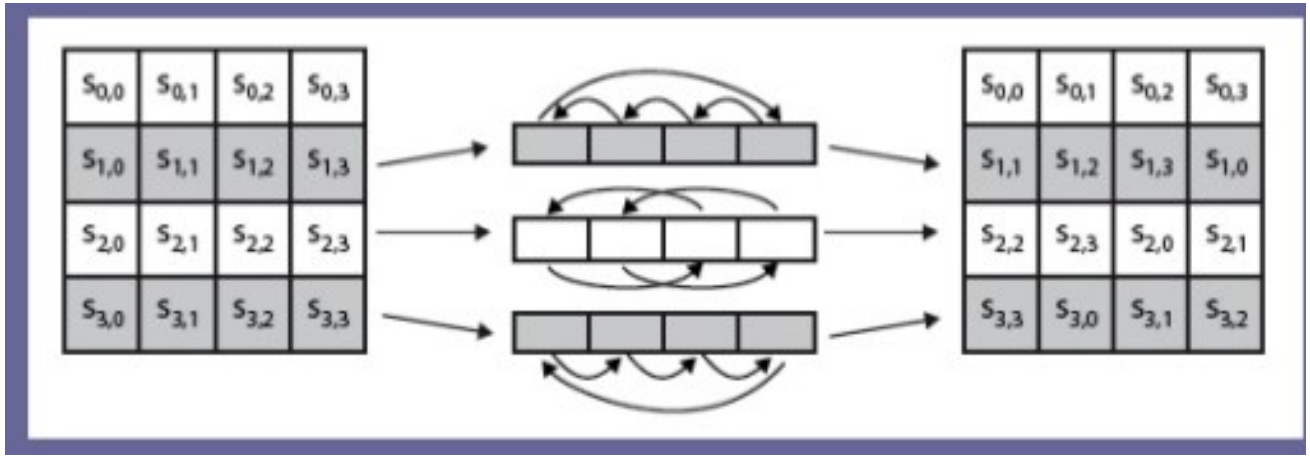
Advanced Encryption Standard (AES)

PERMUTATION

Another transformation found in a round is SHIFTING.

ShiftRows

InvShiftRows



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

→

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

Advanced Encryption Standard (AES)

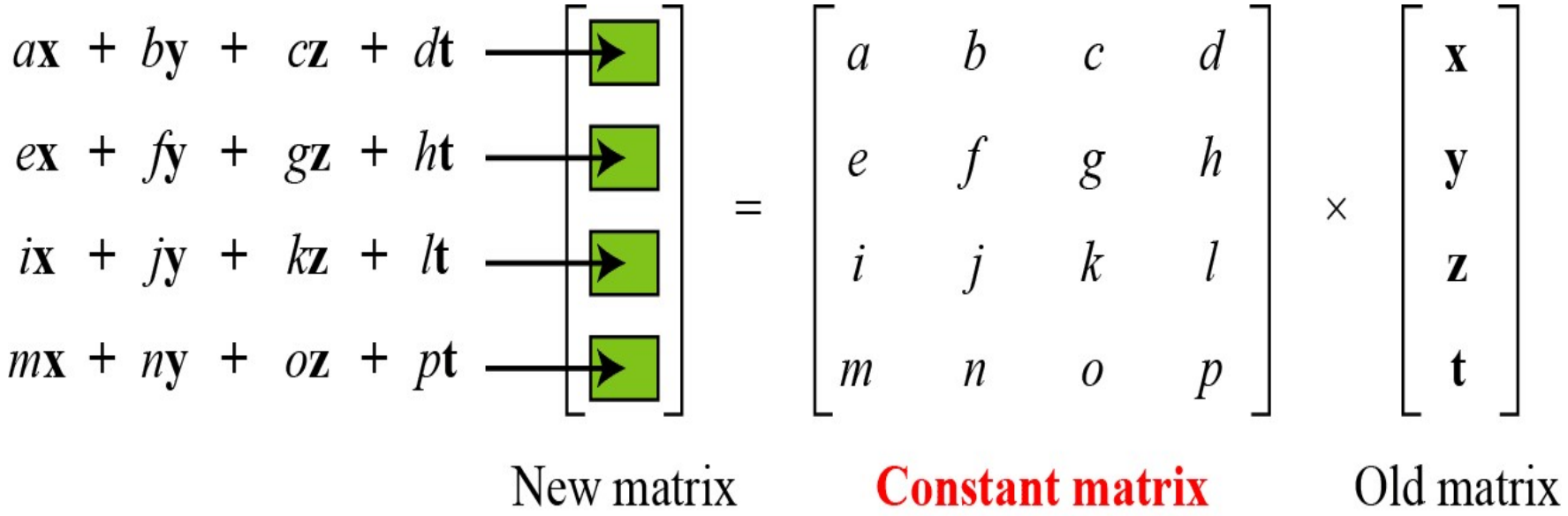
SUBSTITUTION

PERMUTATION

MIXING

Advanced Encryption Standard (AES)

MIXING



Advanced Encryption Standard (AES)

MIXING

1. An interbyte transformation is required that changes the bits inside a byte, based on the bits inside the neighboring bytes.
2. We need to mix bytes to provide diffusion at the bit level.
3. The mixing transformation changes the contents of each byte by taking 4 bytes at a time and combining them to recreate 4 new bytes.
4. To guarantee that each byte is different, the combination process first multiplies each byte with a different constant and then mixes them
5. Mixing is provided by **MATRIX MULTIPLICATION**

Advanced Encryption Standard (AES)

MIXING

1. AES defines a transformation called **MixColumns,** to achieve this goal
2. ***There is also an inverse transformation called InvMixColumns.***
3. The following Constant Matrices are used for these transformations.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \quad \text{Inverse} \quad \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

C $\qquad$ C^{-1}

Advanced Encryption Standard (AES)

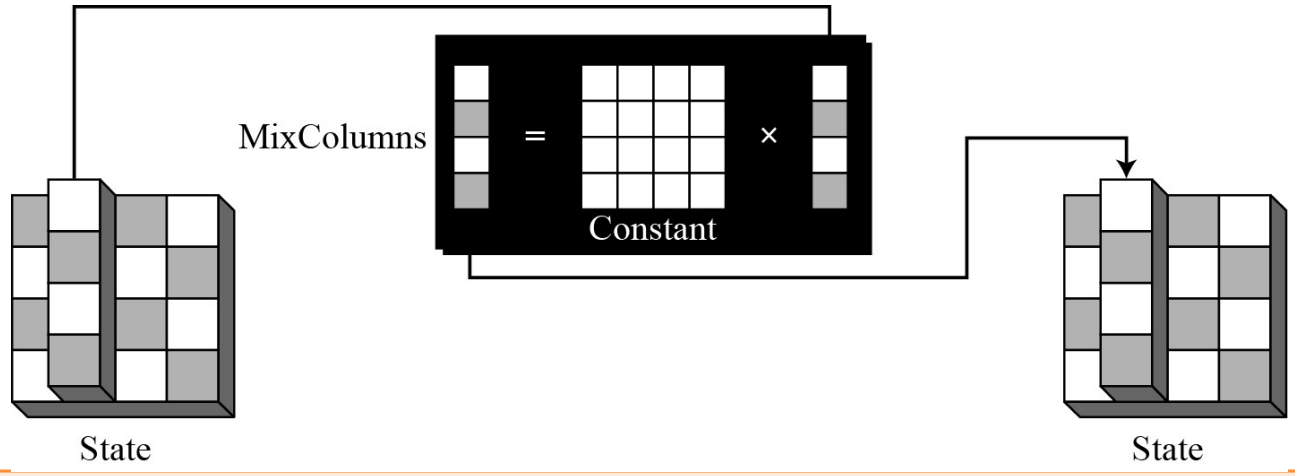
MIXING

MixCoulmns

Transformation operates at the column level

It transforms each column of the state to a new column

The transformation is actually the MATRIX MULTIPLICATION of a state column by a constant square matrix



The bytes in the state column and constants matrix are interpreted as 8-bit words

Advanced Encryption Standard (AES)

SUBSTITUTION

PERMUTATION

MIXING

KEY ADDIING

Advanced Encryption Standard (AES)

KEY ADDING

AddRoundKey

AddRoundKey proceeds one column at a time.

AddRoundKey adds a round key word with each state column matrix

It can be thought as XORing of each column of the state, with the corresponding keyword

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

$\oplus$

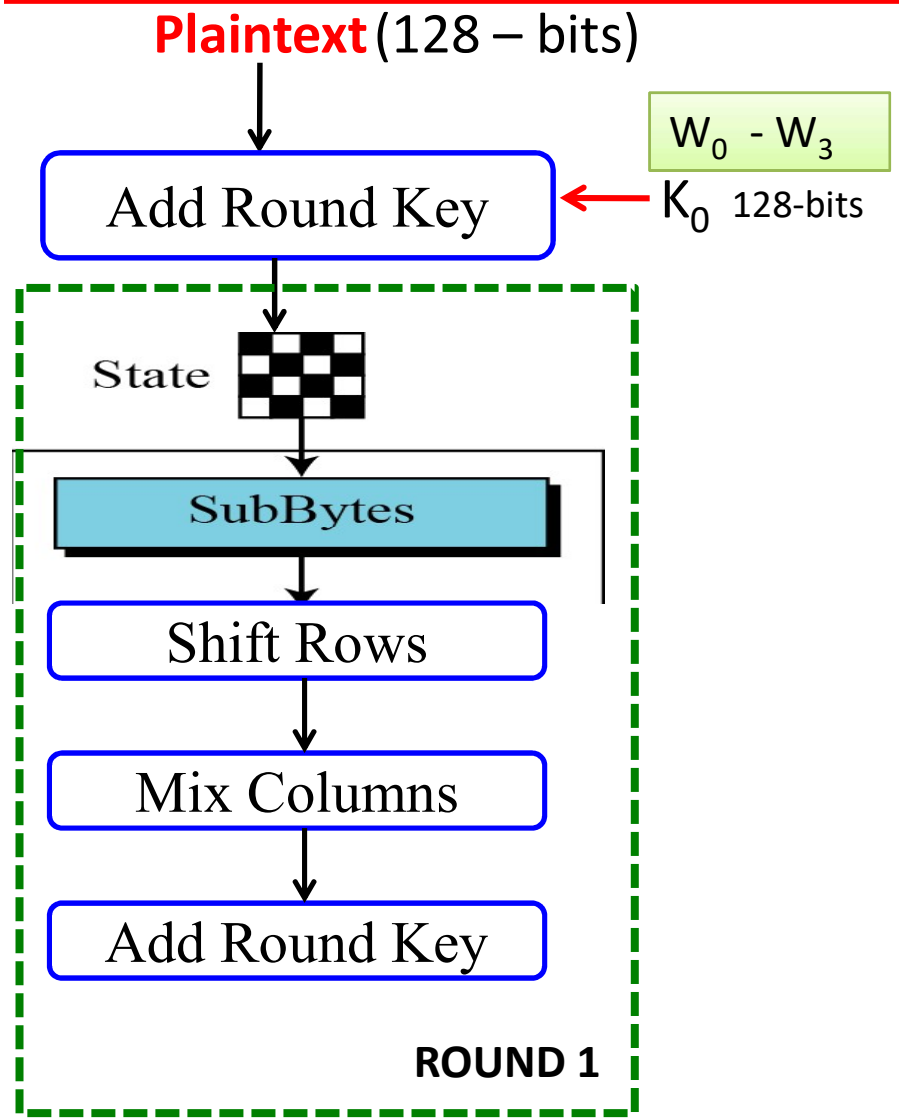
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

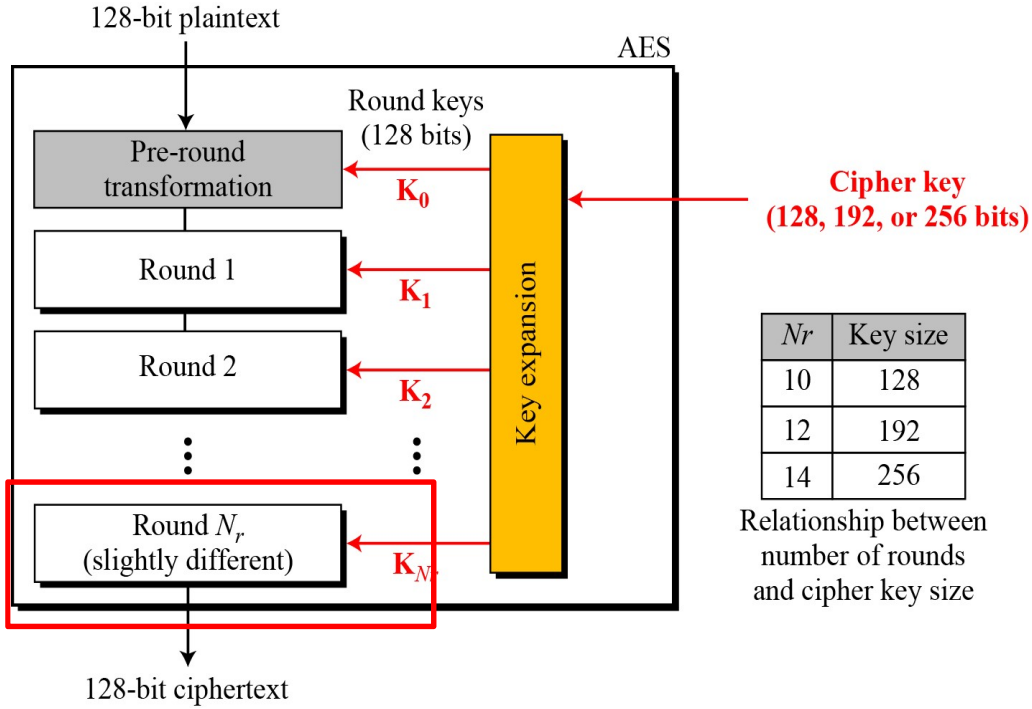
$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Cipher Key is expanded into a set of keywords

Advanced Encryption Standard (AES)

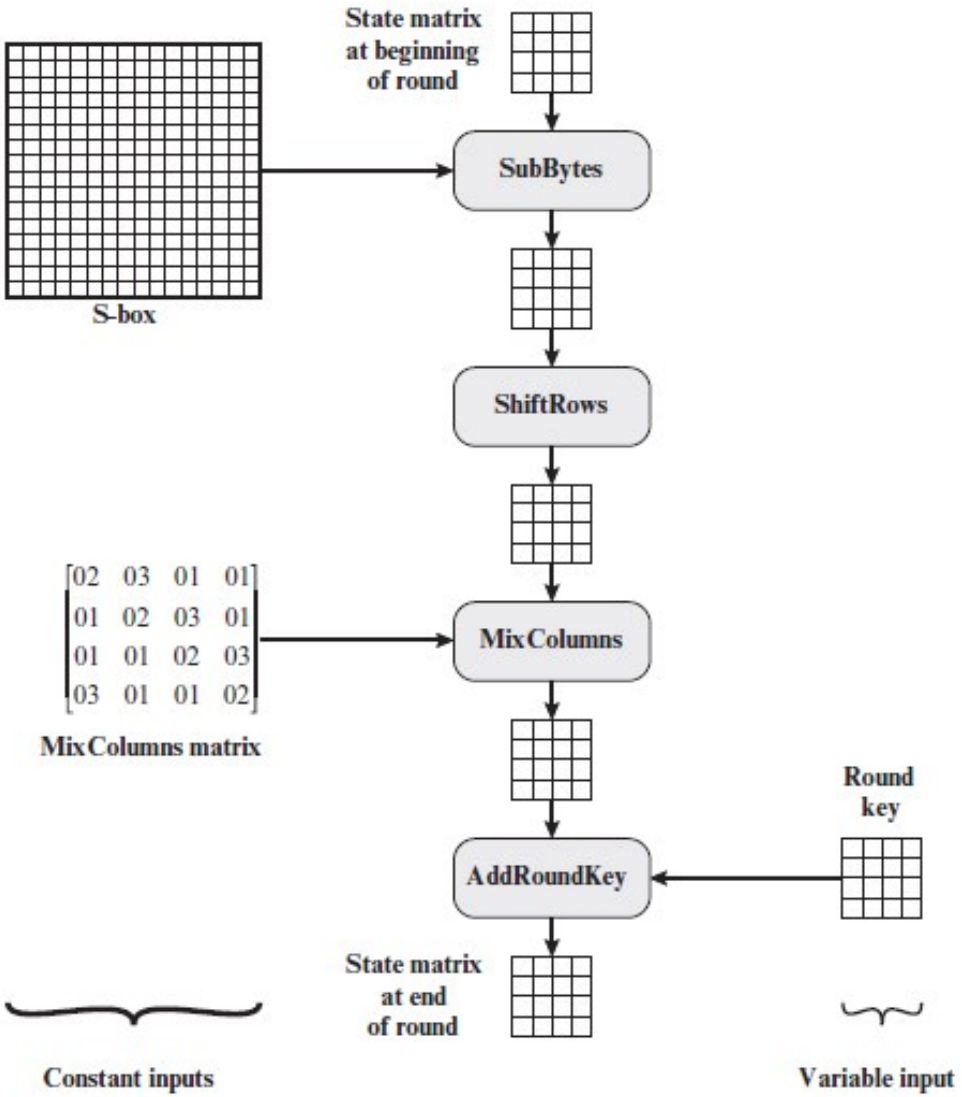




Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

Advanced Encryption Standard (AES)



ROUND 1

Advanced Encryption Standard (AES)

KEY EXPANSION

1. To create round keys for each round, AES uses a key-expansion process.
2. If the number of rounds is N_r , the key-expansion routine creates $N_r + 1$
3. 128-bit round keys are generated from one single 128-bit cipher key.
4. The 1st round key is used for pre-round transformation(AddRoundKey)
5. The remaining round keys 2nd to 10th are used for last transformation at the end of each round
6. The KEY EXPNASION routine creates Round Keys word by word, where word is an array of 4 bytes
7. Total number of words created are $4*(N_r+1)$

Advanced Encryption Standard (AES)

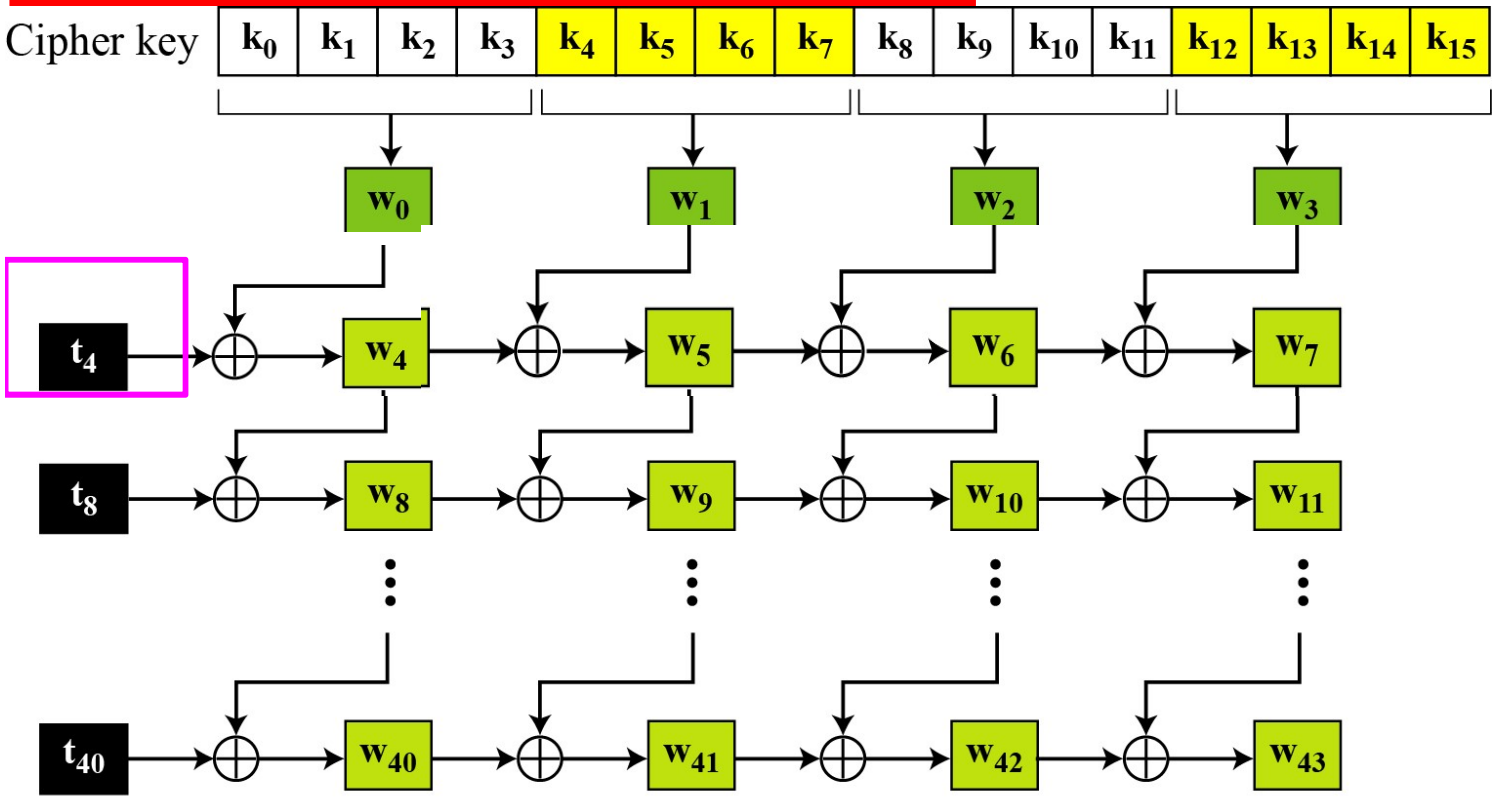
KEY EXPANSION

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	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

Advanced Encryption Standard (AES)

KEY EXPANSION



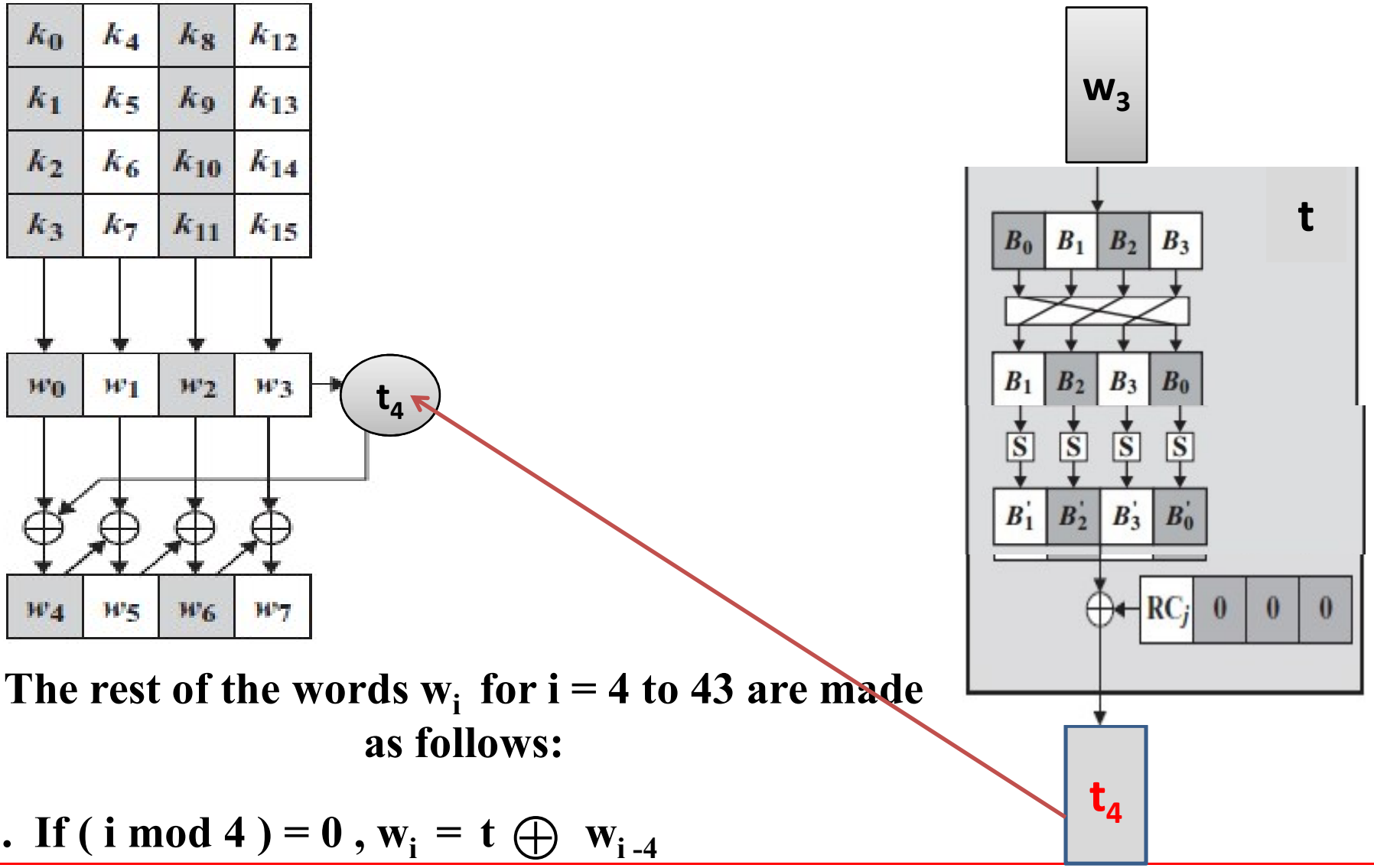
The 1st four words (w_0, w_1,w_2,w_3) are made from cipher key

The rest of the words w_i for i = 4 to 43 are made as follows:

1. **If (i mod 4) ≠ 0 , $w_i = w_{i-1} \oplus w_{i-4}$**
2. **If (i mod 4) = 0 , $w_i = t \oplus w_{i-4}$**

Advanced Encryption Standard (AES)

KEY EXPANSION

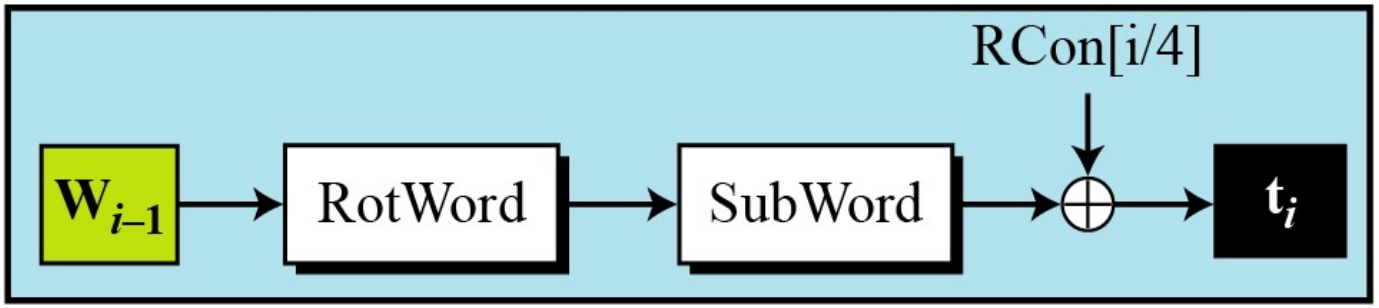


The rest of the words w_i for i = 4 to 43 are made as follows:

1. **If (i mod 4) = 0 , $w_i = t \oplus w_{i-4}$**

Advanced Encryption Standard (AES)

KEY EXPANSION



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

1. RotWord performs a one-byte circular left shift on a word. This means that an input word [B0, B1, B2, B3] is transformed into [B1, B2, B3, B0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

Advanced Encryption Standard (AES)

KEY EXPANSION

Table 7.4 *RCon constants*

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

1. The round constant is a word in which the three rightmost bytes are always 0.
2. Thus, the effect of an XOR of a word with Rcon is to only perform an XOR on the leftmost byte of the word.
3. The round constant is different for each round and is defined as

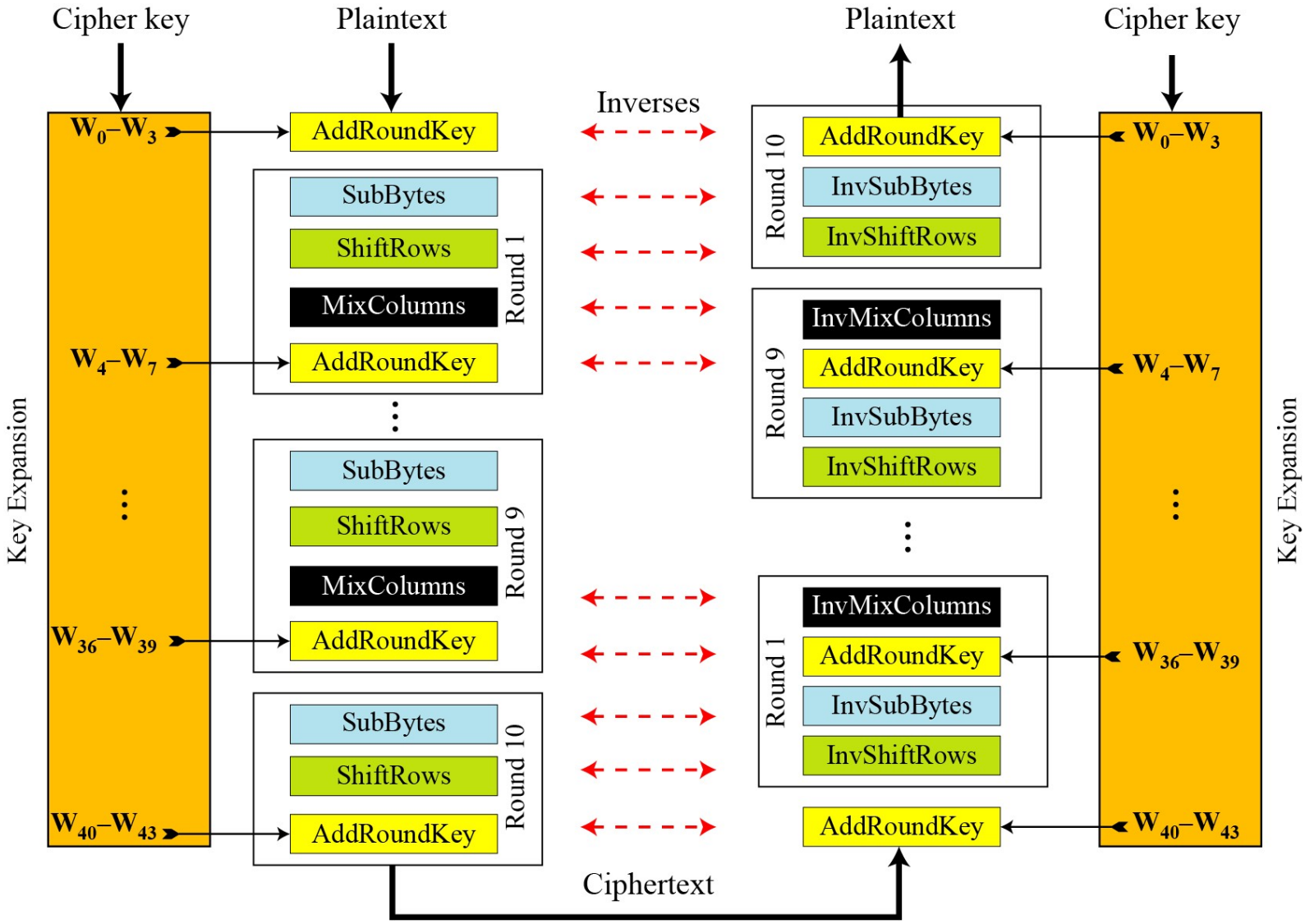
Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1, RC[j] = 2 # RC[j - 1]

AND

With multiplication defined over the field GF(28).

AES

Ciphers and inverse ciphers of the original design



Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

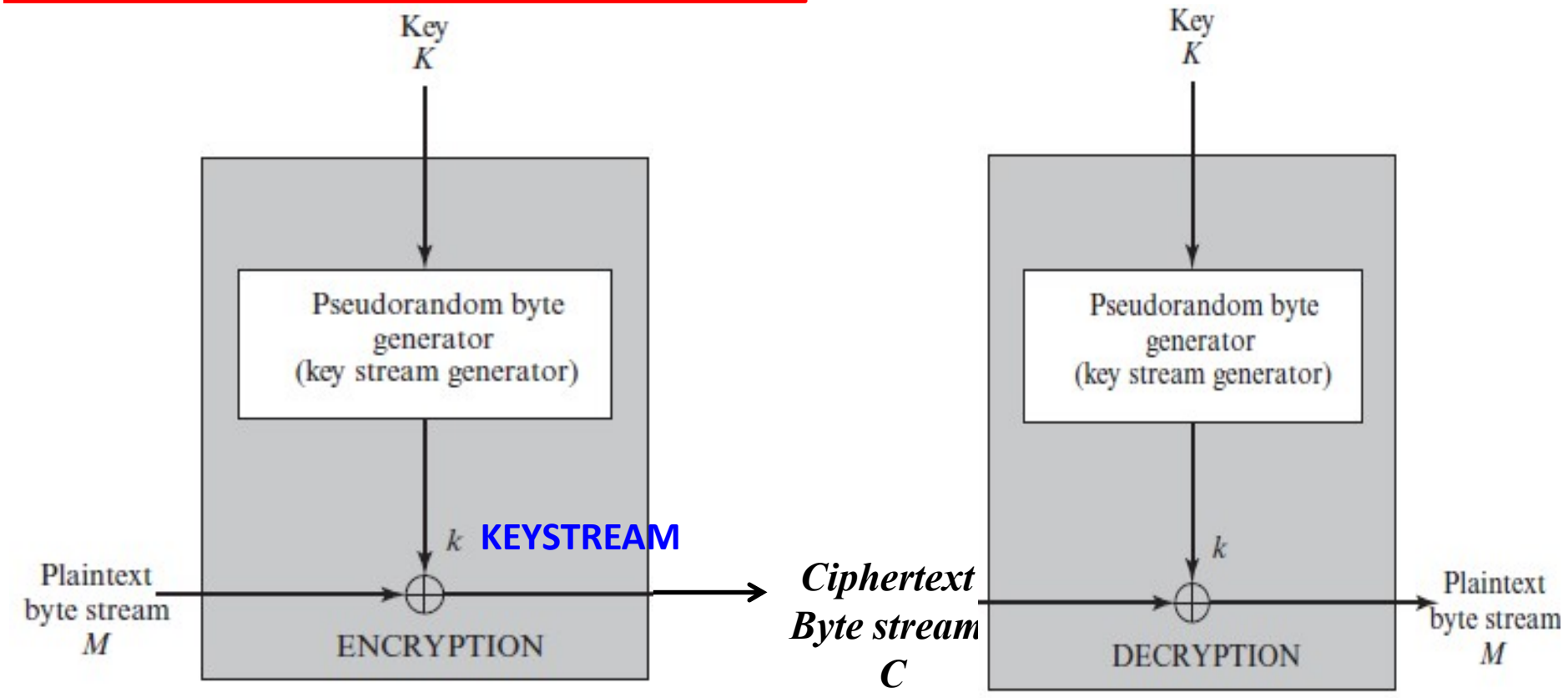
2.1

- Block Cipher
  - ✓ **Block Cipher Principles**
  - ✓ **Block Cipher Modes of Operation**
  - ✓ **Data Encryption Standard (DES)**
  - ✓ **Double DES, Triple DES**
  - ✓ **Advanced Encryption Standard (AES)**
- Stream Cipher
  - ✓ RC5 algorithm

STREAM CIPHER

1. Stream ciphers are a type of encryption algorithm that process an individual bit, byte, or character of plaintext at a time.
2. A Stream Cipher is used for symmetric key cryptography
3. Stream ciphers are often faster than block ciphers in hardware and require circuitry that is less complex.
4. They are also useful when transmission errors are likely to occur because they have little or no error propagation.

STREAM CIPHER (Block Diagram)



A **keystream** is a sequence of pseudorandom digits which extend to the length of the plaintext in order to uniquely encrypt each character based on the corresponding digit in the keystream

Types of Stream cipher

1. Stream ciphers can be classified into

   a) **Synchronous:** A Synchronous Stream Cipher generates a keystream based on internal states not related to the plaintext or ciphertext. This means that the stream is generated pseudorandomly outside of the context of what is being encrypted. A binary additive stream cipher is the term used for a stream cipher which XOR's the bits with the bits of the plaintext.

   b) **Self-synchronizing:** A Self-synchronizing Stream Cipher, also known as an asynchronous stream cipher or ciphertext autokey (CTAK), is a stream cipher which uses the previous N digits in order to compute the keystream used for the next N characters.

Design considerations for a stream cipher

1. The encryption sequence should have a large period. A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats. The longer the period of repeat the more difficult it will be to do cryptanalysis.
2. The keystream should approximate the properties of a true random number stream as close as possible. The more random-appearing the keystream is, the more randomized the ciphertext is, making cryptanalysis more difficult.
3. To guard against brute-force attacks, the key needs to be sufficiently long

RC5 Stream Cipher

1. Symmetric key algorithm developed by Ron Rivest
2. Its main features are that it is quite fast and uses primitive operations like (Addition, XOR, Shift).
3. It allows for a variable number of rounds and a variable bit-size key to add flexibility
4. RC5 requires less memory for execution and hence it is not only suitable for desktop computers but also for smart cards.
5. It has been incorporated into RSA data security

RC5 Stream Cipher

1. RC5 allows for variable values in 3 parameters and is denoted asRC5 – ***w/r/b***

w:- Word size in bits

r:- Number of Rounds

b:- Number of 8-bits byte in the key

RC5 uses 2 word blocks

RC5 – 32/16/16

Parameter	Allowed Values
Word size in bits (RC5 encrypts 2-word blocks at a time)	16, 32, 64
Number of rounds	0-255
Number of 8-bit bytes (octets) in the key	0-255

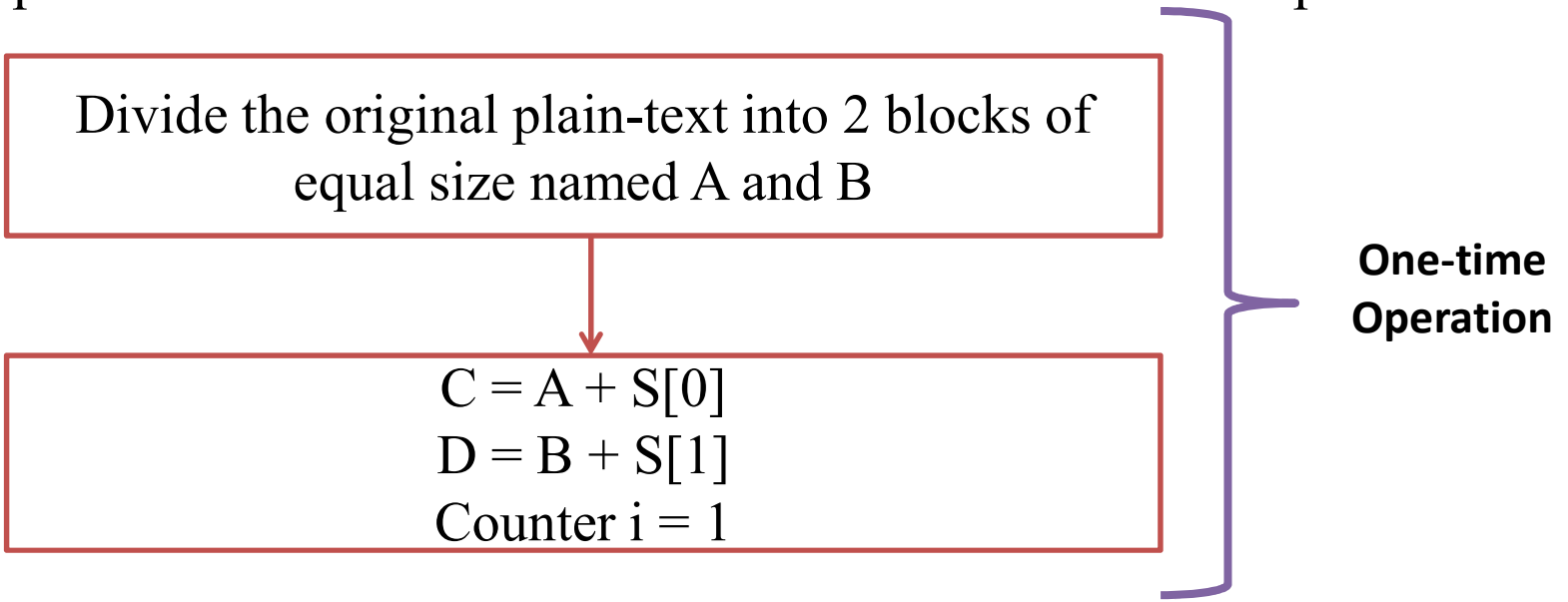
The following conclusions emerge from the table:

- The plain-text block size can be of 32, 64 or 128 bits (since 2-word blocks are used).
- The key length can be 0 to 2040 bits (since we have specified the allowed values for 8-bit keys).

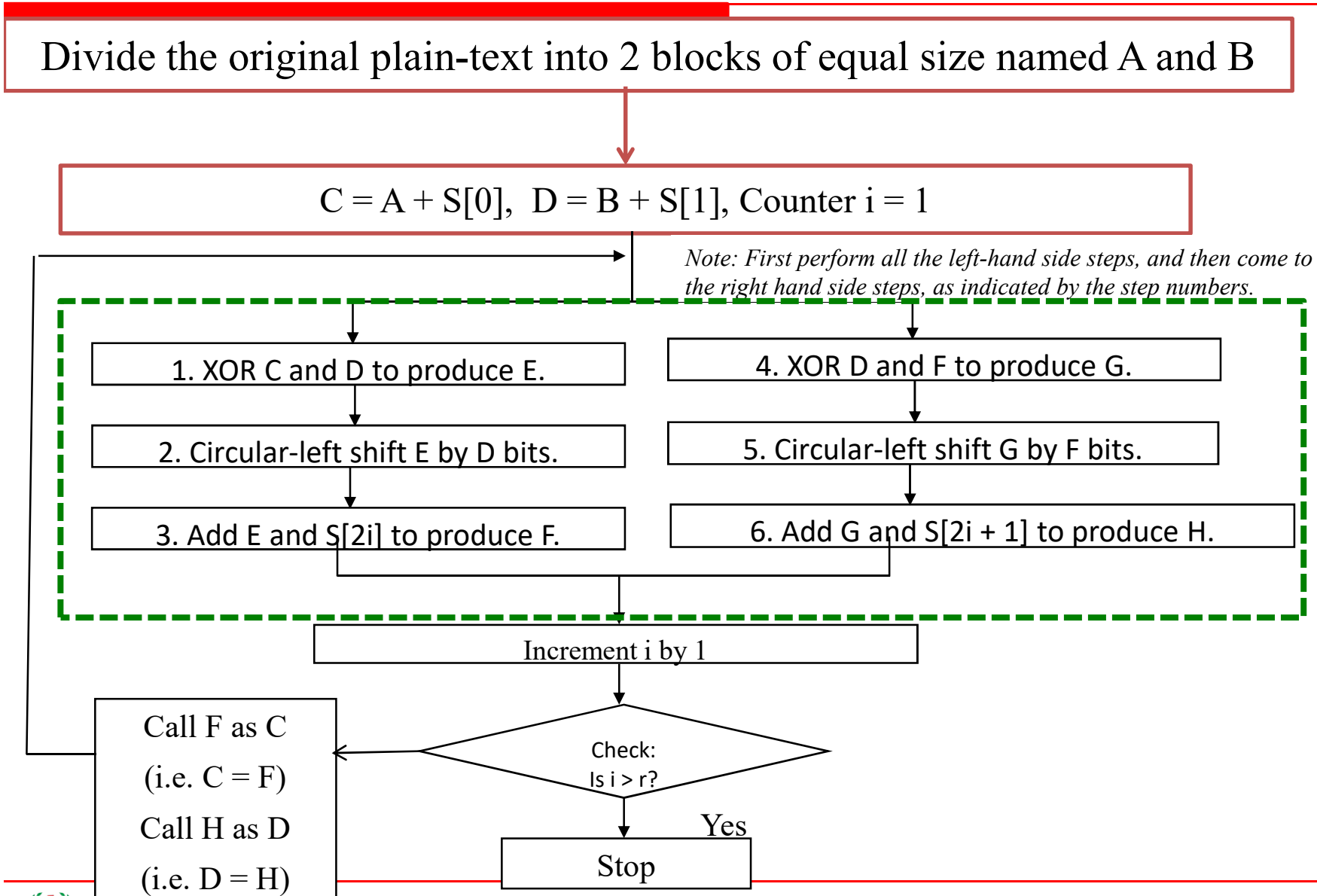
Rivest has suggested RC5-32/12/16 as the *minimum safety version.*

RC5 Stream Cipher

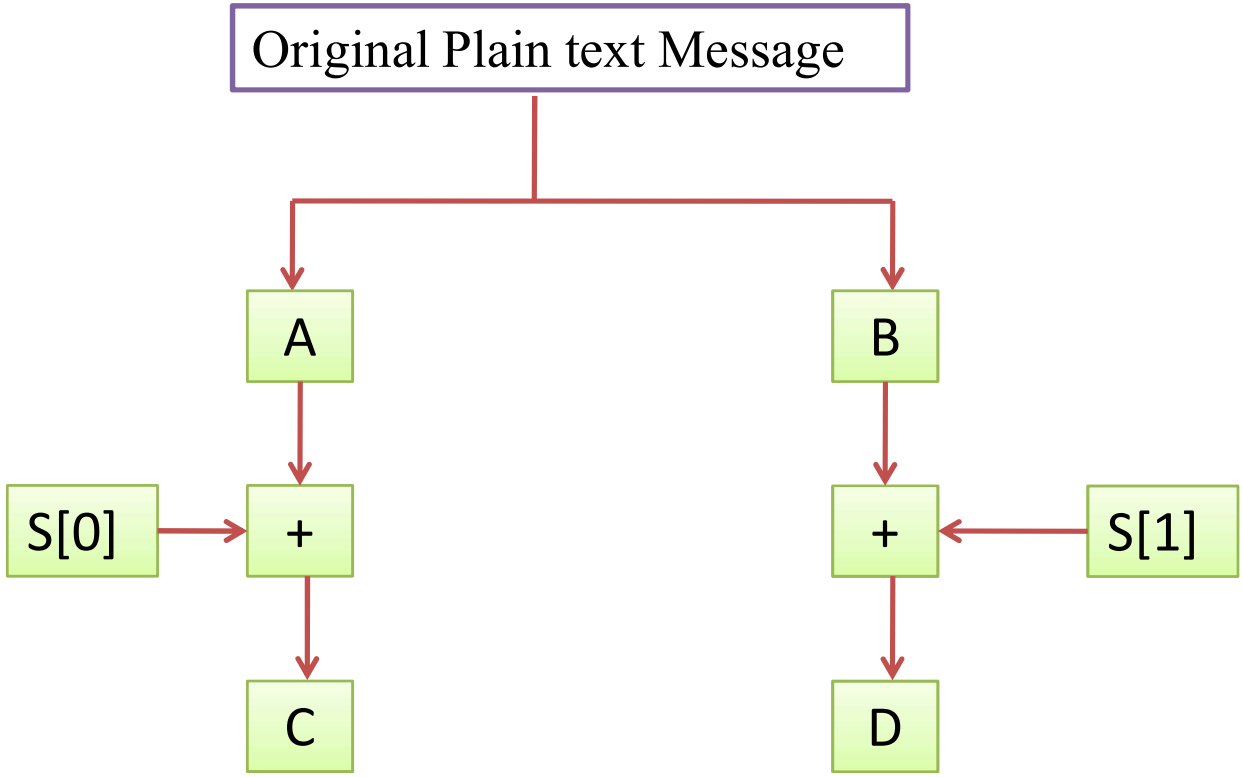
1. Assume an input plain block with size 64-bit.
2. In the first two steps of the one-time initial operation, the input plain text is divided into two 32 bits blocks A and B.
3. The 1st two sub-keys (S[0] and S[1]) are added to A and B respectively
4. Thos produces C and D and marks the end of the one-time operation



RC5 Stream Cipher



RC5 Stream Cipher: One Time Initial Operation

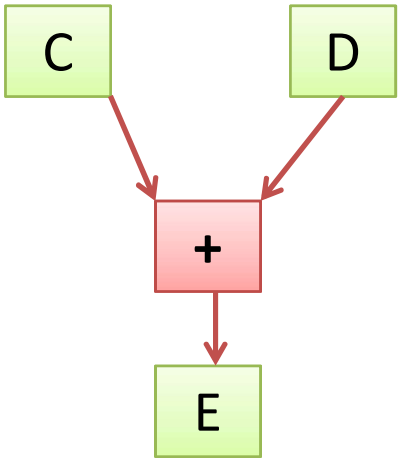


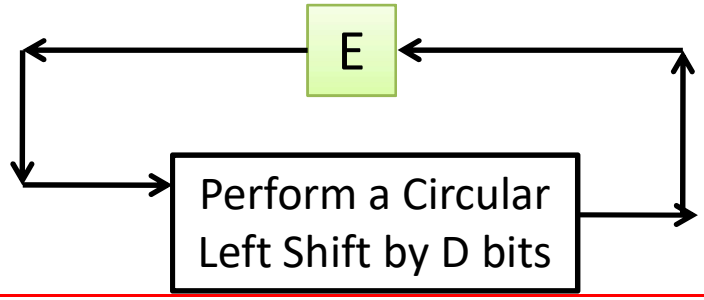
RC5 Stream Cipher: Details of ONE Round

Step 1: XOR C and D

Step 2: Circular Left Shift E

Step 3: Add E and Next Sub-key

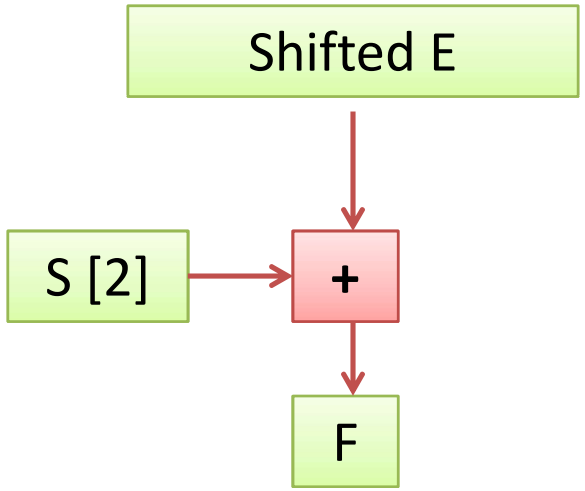




Step 4: XOR D and F

Step 5: Circular Left Shift G

Step 6: Add G and Next Subkey



RC5 Stream Cipher: Details of ONE Round

Miscellaneous Tasks: In this step, we check to see of all the rounds are over or not. For this we perform the following steps

1. Increment by ***i***
2. Check to see if ***i < r***

Assume that ***i*** is still less than ***r***, we rename F as C and H as D and return back to step 1.

Mathematical Representation of RC5 ENCRYPTION

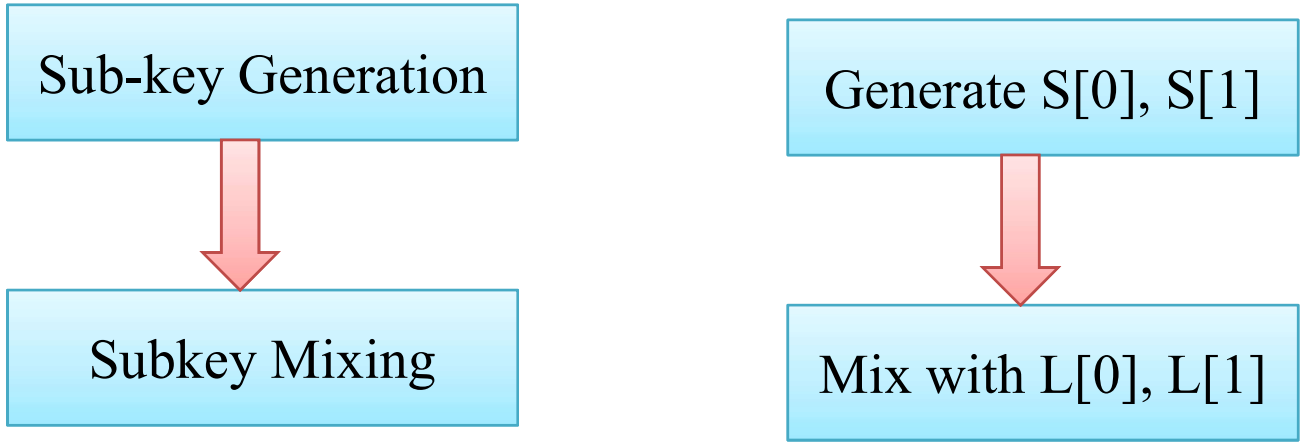
```
A = A + S[0]
B = B + S[1]

For i = 1 to r
      A = ((A XOR B) <<< B) + S[2i]
      B = ((B XOR A) <<< A) + S[2i + 1]
Next i
```

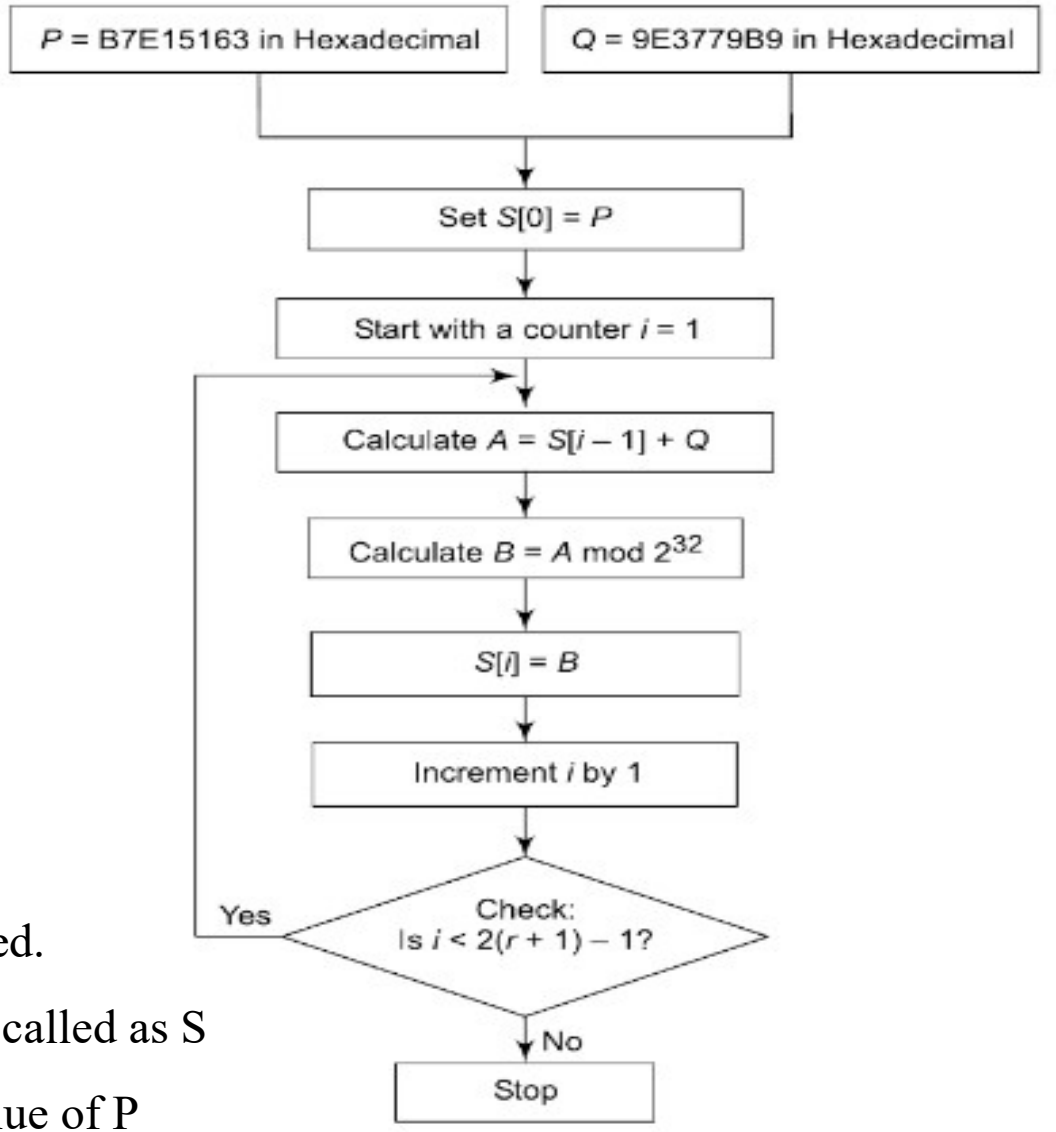
RC5 Stream Cipher: Subkey Generation

This is a 2-step process

1. The sub-keys denoted by S[0], S[1], S[2],..... Are generated
2. The original key is called L. In the 2nd step the subkeys (S[0], S[1],..) are mixed with the corresponding sub-portions of the original key (i.e. L[0], L[1], L[2],...)



RC5 Stream Cipher: Subkey Generation



1. In this step, 2 constants P and Q are used.
2. The array of subkeys to be generated is called as S
3. The 1st subkey is initialized with the value of P
4. Each next sub-key is calculated on the basis of the previous sub-key and the constant value Q

Module 2: Symmetric & Asymmetric Key Cryptography and Key Management

2.2

- Public Key Cryptography
  - ✓Principles of public key cryptosystems
  - ✓The RSA algorithm
  - ✓The knapsack algorithm
  - ✓ElGamal Algorithm

RSA Algorithm

- RSA algorithm is asymmetric (public key) cryptography algorithm.
- Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key.
- As the name describes that the Public Key is given to everyone and Private key is kept private.
- The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

The RSA algorithm holds the following features −

a) RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.

b) The integers used by this method are sufficiently large making it difficult to solve.

c) There are two sets of keys in this algorithm: private key and public key.

RSA Algorithm

The following steps highlight how it works:

1. Generating the keys
2. Encryption
3. Decryption

The pair of numbers (n,e) makes up the public key.

The pair (n,d) makes up the private key.

1. Generating the keys

Select two large prime numbers, p and q.

Calculate $n=p * q$.

Calculate the ***totient*** function; $\phi(n)$

Select an integer e, such that e is ***co-prime*** to $\phi(n)$ and $1<e<\phi(n)$.

Calculate d such that $d= e^{-1} \bmod \phi(n)$. i.e $e.d = 1 \bmod \phi(n)$

d can be found using the ***extended euclidean algorithm***.

RSA Algorithm

The following steps highlight how it works:

1. Generating the keys
2. Encryption — The pair of numbers (n,e) makes up the public key.
3. Decryption — The pair (*n*,*d*) makes up the private key.

Encryption

Given a plaintext P, represented as a number, the ciphertext C is calculated as:

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

Decryption

Given a Ciphertext C, represented as a number, the plaintext P is calculated as:

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

RSA Algorithm (Example)

Let p =3 and q = 11

n = p * q

= 3 * 11

=33

$\phi(33) = 11 * 3$

$= (11-1) * (3-1)$

$= 10 * 2 = 20$

Select e such that such that e is co- prime to $\phi(n)$ and $1<e<\phi(n)$.

$1 < e < 20$

e = 7

Calculate d such that $d= e^{-1} \bmod \phi(n)$. i.e $e.d = 1 \bmod \phi(n)$

d = 3

1. Generating the keys

Select two large prime numbers, *p* and *q*.

Calculate $n = p * q$.

Calculate the ***totient*** function; $\phi(n)$

Select an integer *e*, such that *e* is ***co-prime*** to $\phi(n)$ and $1<e<\phi(n)$.

Calculate d such that $d= e^{-1} \bmod \phi(n)$. i.e $e.d = 1 \bmod \phi(n)$

RSA Algorithm

The pair of numbers (n,e) makes up the public key.

Public Key = (33, 7)

$$C = P^e \bmod n$$
$$= 31^7 \bmod 33$$
$$= 4$$

The pair (*n*,*d*) makes up the private key.

Private Key = (33, 3)

$$P = C^d \bmod n$$
$$= 4^3 \bmod 33$$
$$= 31$$

The Knapsack Algorithm

1. Knapsack Encryption Algorithm is the first general public key cryptography algorithm.
2. It is developed by Ralph Merkle and Mertin Hellman in 1978.
3. As it is a Public key cryptography, it needs two different keys.
4. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process.

The Knapsack Algorithm – Steps to follow

STEPS TO FOLLOW:

1. Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.
2. Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater then the sum of all values in private key.
3. Calculate the values of Public key using m and n.

Super Increasing knapsack problem. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

The Knapsack Algorithm – Example

Bob and Alice wants to use knapsack for their secure communication. Bob has a Superincreasing knapsack {1, 2, 4, 10, 20, 40} and he chooses m=110 and n=31. Calculate the public and private keys of Bob.

Calculating Public Key

Multiply all the values of private key by the number n and then find modulo m

1 * 31 mod 110 = 31
2 * 31 mod 110 = 62
4 * 31 mod 110 = 14
10 * 31 mod 110 = 90
20 * 31 mod 110 = 70
40 * 31 mod 110 = 30

Thus, our public key is {31, 62, 14, 90, 70, 30}

And Private key is {1, 2, 4, 10, 20, 40}.

Calculate the values of Public key using m and n.

The Knapsack Algorithm – Example

ENCRYPTION

Plain text is 1001001111100101110

1. As our knapsacks contain six values, so we will split our plain text in a groups of six:

100100 111100 101110

2. Multiply each values of public key with the corresponding values of each group and take their sum.

100100 Public Key Is {31, 62, 14, 90, 70, 30}

1 * 31+0 * 62+0 * 14+1 * 90+0 * 70+0 * 30 = 121

111100 {31, 62, 14, 90, 70, 30}
1x31+1x62+1x14+1x90+0x70+0x30 = 197

101110 {31, 62, 14, 90, 70, 30}
1x31+0x62+1x14+1x90+1x70+0x30 = 205

So, our cipher text is 121 197 205.

The Knapsack Algorithm – Example

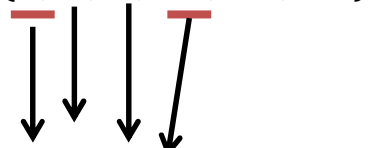
DECRYPTION

Cipher text is 121 197 205.

$n \times n^{-1} \mod(m)$
$=31 \times \mod(110)$
$= 71$

Multiply 71 With Each Block Of Cipher Text Take Modulo m.
$121 \times 71 \mod(110)$
$= 11$

{1, 2, 4, 10, 20, 40}

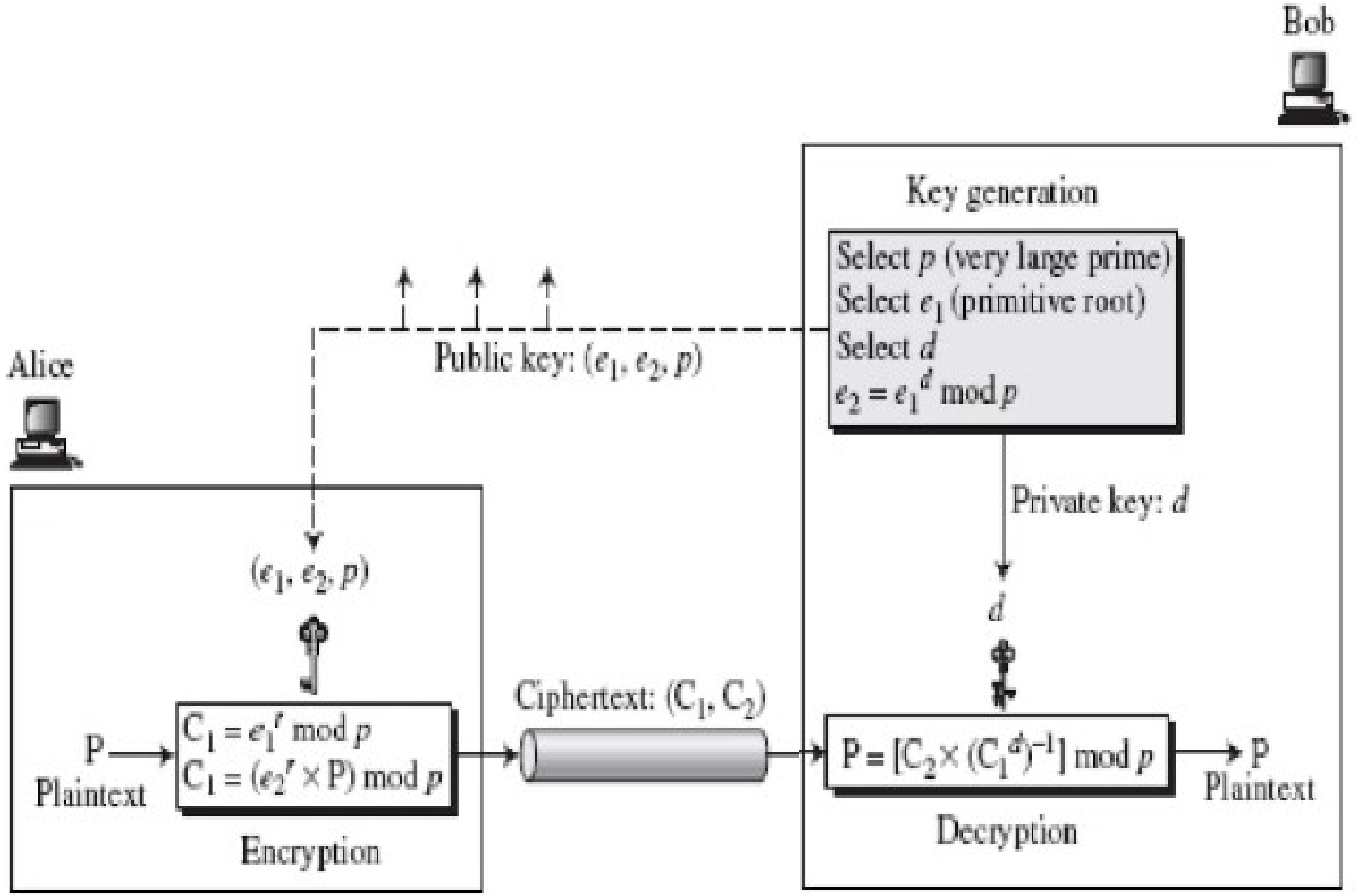


{1, 0, 0, 1, 0, 0}

ElGamal Algorithm – Public Key Cryptography

1. ElGamal encryption is an public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.
2. It is named after its inventor, Taher Elgamal.
3. This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group that is even if we know g^a and g^k, it is extremely difficult to compute g^{ak}.

ElGamal Algorithm – Procedure



ElGamal Algorithm – Public Key Cryptography

1. **Generating Keys**

   a) Select a large Prime Number (p)

   b) Select e1 (Primitive Root)

   c) Select Decryption key d (Private Key)

   d) $e2 = e1^d \bmod p$

   e) Encryption Key (Public Key) = (e1, e2, p)

2. **Encryption**

   a) Select Random Integer r
   b) $C1 = = e1^r \bmod p$
   c) $C2 = (\text{Plain Text} * e2^r) \bmod p$
   d) Cipher Text = (C1, C2)

3. **Decryption**

   a) $\text{Plain Text} = [C2 * (C1^d)^{-1}] \bmod p$

ElGamal Algorithm – Public Key Cryptography

1. **Generating Keys**

   a) Select a large Prime Number (p) **11**

   b) Select e1 (Primitive Root)

   c) Select Decryption key d (Private Key)

   d) $e2 = e1^d \bmod p$

   e) Encryption Key (Public Key) = (e1, e2, p)

2. **Encryption**

   a) Select Random Integer r
   b) $C1 = = e1^r \bmod p$
   c) $C2 = (\text{Plain Text} * e2^r) \bmod p$
   d) Cipher Text = (C1, C2)

3. **Decryption**

   a) $\text{Plain Text} = [C2 * (C1^d)^{-1}] \bmod p$

ElGamal Algorithm – Public Key Cryptography

$$2^5 = 2 * 2^4$$

$$2^5 = 2 * 5$$

$$2^5 = 10$$

1. Generating Keys

a) Select a large Prime Number (p)

b) Select e1 (Primitive Root)

11

POWERS

a

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

Primitive Roots

- If $x^n = a$ then a is called the n-*th* root of x
- For any prime number p, if we have a number a such that powers of a mod p generate all the numbers between *1* to p-*1* then a is called a **Primitive Root** of p.

ElGamal Algorithm – Public Key Cryptography

1. Generating Keys

a) Select a large Prime Number (p) **11**

b) Select e1 (Primitive Root) **2**

c) Select Decryption key d (Private Key) **3**

d) $e2 = e1^d \bmod p = 2^3 \bmod 11 =$ **8**

e) Encryption Key (Public Key) = (e1, e2, p) **(2, 8, 11)**

Assume Plaintext 7

2. Encryption

a) Select Random Integer r **4**

b) $C1 = e1^r \bmod p = 2^4 \bmod 11 =$ **5**

c) $C2 = (\text{Plain Text} * e2^r) \bmod p$ **7** $* 8^4 \bmod 11 =$ **6**

d) Cipher Text = (C1, C2) = (5, 6)

3. Decryption

a) $\text{Plain Text} = [C2 * (C1^d)^{-1}] \bmod p = [6 * (5^3)^{-1}] \bmod 11 = 7$

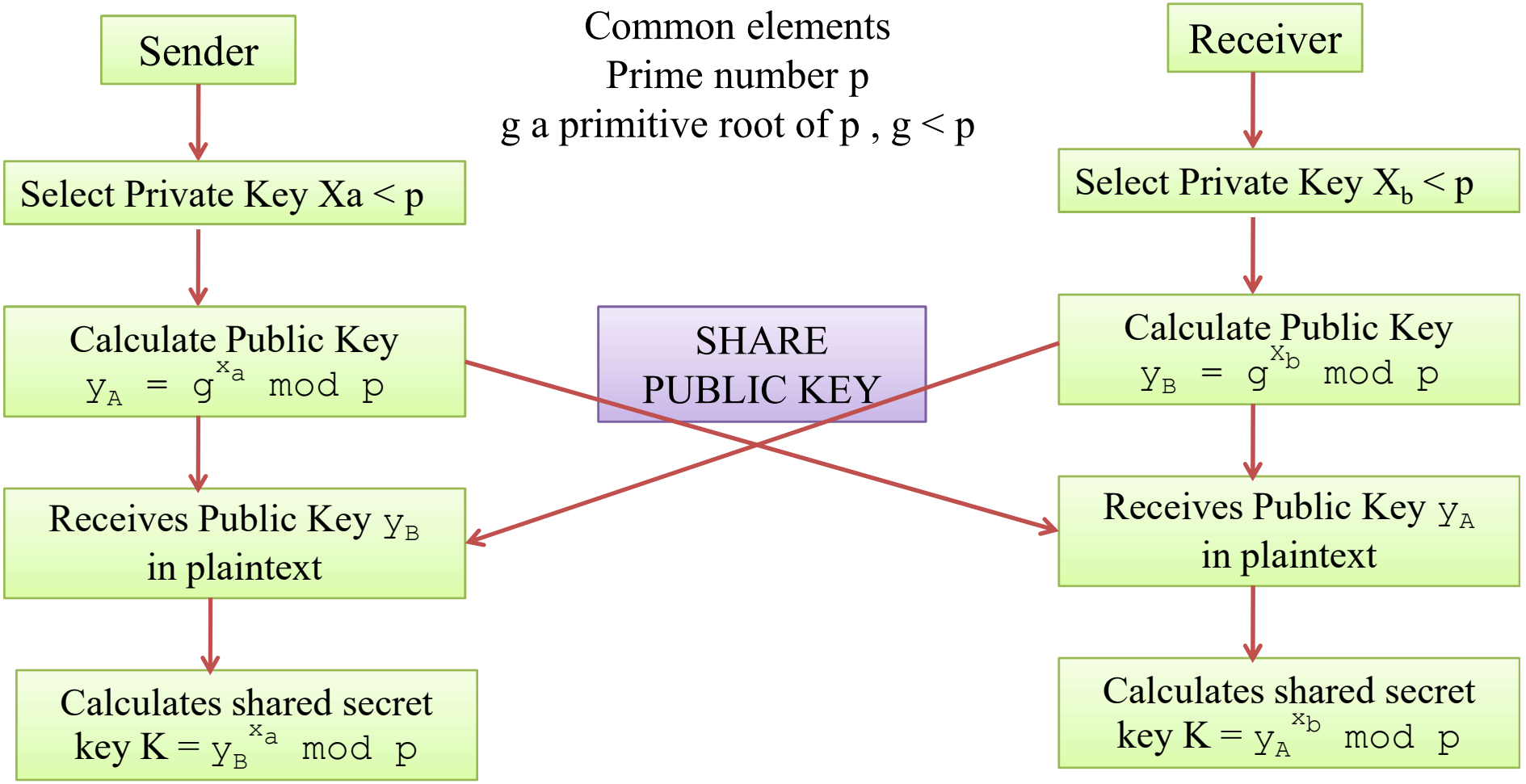
2.3

- Key management techniques:
  - ✓ Using symmetric and asymmetric algorithms and trusted third party.
  - ✓ Diffie Hellman Key exchange algorithm

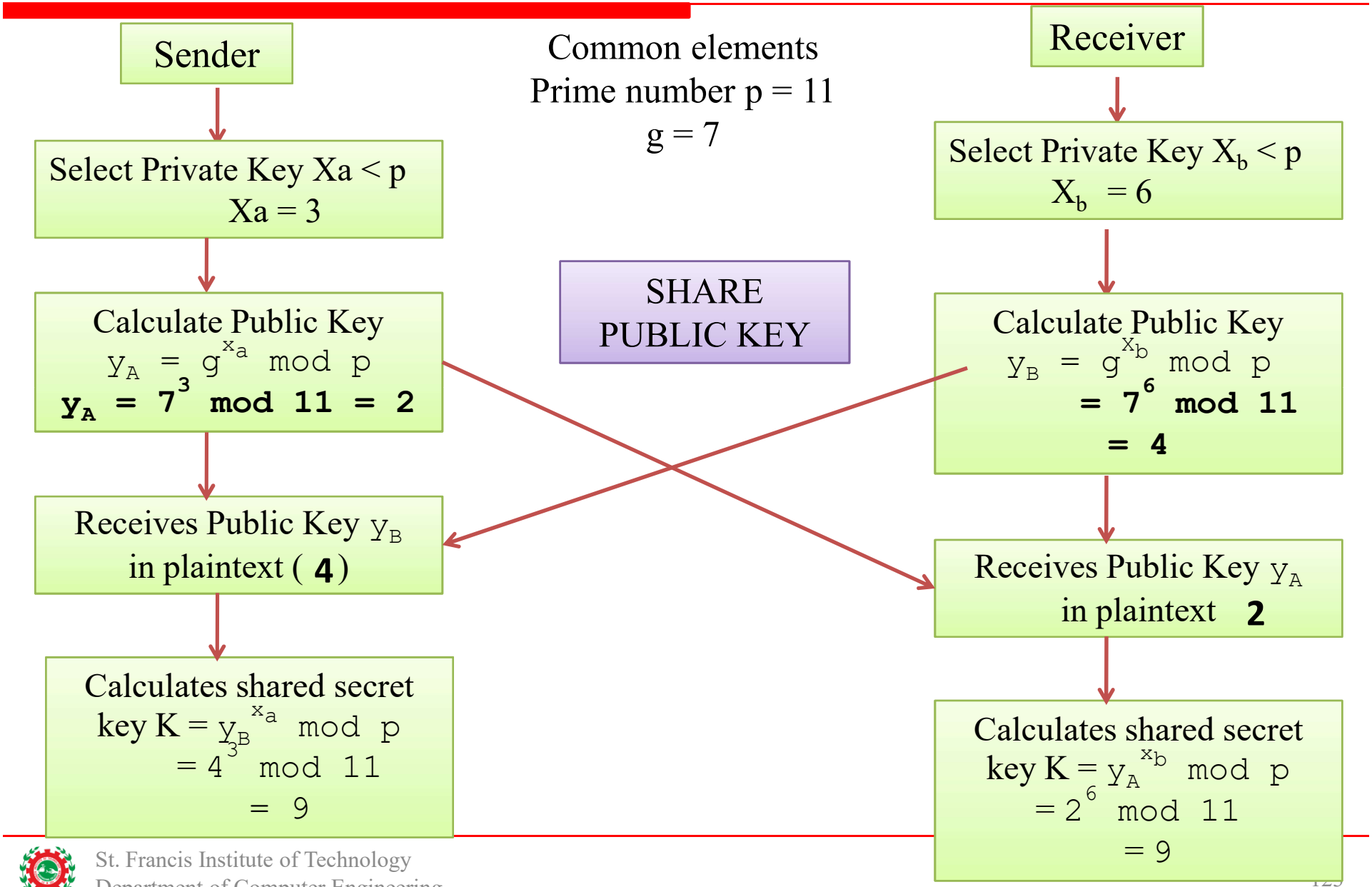
Diffie-Hellman Key Exchange Algorithm

1. Diffie Hellman (DH) key exchange algorithm is a method for securely exchanging cryptographic keys over a public communications channel.
2. Keys are not actually exchanged – they are jointly derived.
3. It is named after their inventors Whitfield Diffie and Martin Hellman.
4. The purpose of the algorithm is to enable two users to securely exchange keys that can then be used for subsequent symmetric encryption of messages
5. The algorithm itself is limited to the exchange of secret values
6. The DH algorithm depends for its effectiveness on the difficulty of computing discrete logarithms

Diffie-Hellman Key Exchange Algorithm - Procedure



Diffie-Hellman Key Exchange Algorithm - Example



Diffie-Hellman Key Exchange Algorithm - Example

Suppose that two parties A and B wish to setup a common secret key (D-H key) between themselves using the Diffie-Hellman key exchange technique. They agree on 7 as the modulus and 3 as the primitive root. Party A chooses 2 and party B chooses 5 as their respective secrets. Their D-H key is

Diffie-Hellman Key Exchange Algorithm - Example

Alice and Bob agrees on q=23 and α=7. Alice chooses secret key as 3 and Bob chooses as 6. Find their public and private keys.

Public Keys: YA and YB Private Key: K

Diffie-Hellman Key Exchange Algorithm - Example

Users A and B use the Diffie Hellman key exchange technique with a common prime q = 71 and primitive root = 7

1. If user A has private key Xa = 5, then what is A's Public Key Ya
2. If user B has private key Xb= 12, what is B's public key Yb
3. What is the shared secret key?