LASSONDE SCHOOL OF ENGINEERING | YORK U

Modern Symmetric Cryptography

Dr. Ruba Al Omari

EECS 3481 – Applied Cryptography

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

Modern Vs. Classical

- Alphabet is {0,1} instead of {A-Z}
  - We think of byte as 8 bits, not char
- To encrypt, digitize the PT
  - Convert string to byte
- To display/transmit PT / CT, we use a Hex String
- After decrypting the CT
  - Convert byte to string

- Continue classifying Sym/Asym, Stream/Block, Substitute/Transposition, Attack Types, but no more Mono/Poly (why?)

Recall - OTP

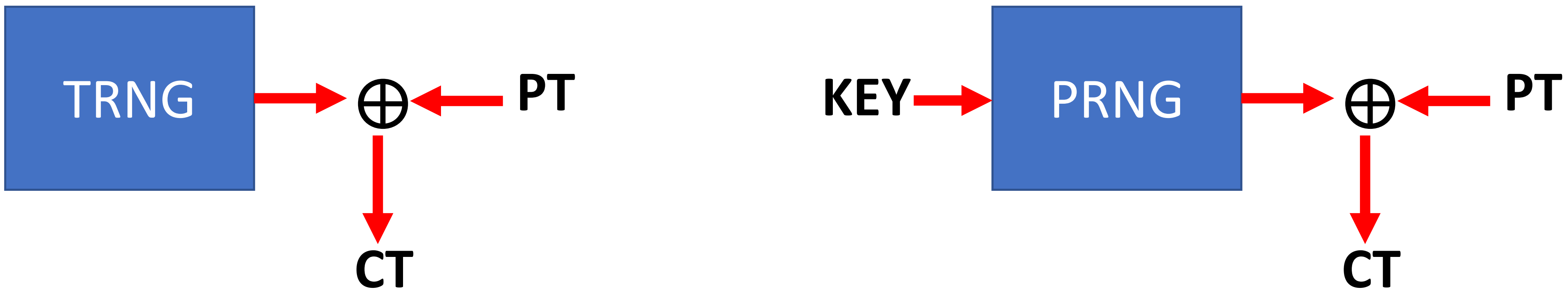
- Vernam cipher that uses a **random key** that is **as long as the message** so that the key is used **only once**.

- Definition
  1. $|K| = |P|$
  2. K is random
  3. $c = E(k, p) = k \oplus p$
  4. K never re-used (hence the O in OTP)

- It boasts perfect secrecy
  - Thwarts *exhaustive* attacks even if Eve had *infinite* classical or quantum computing power!

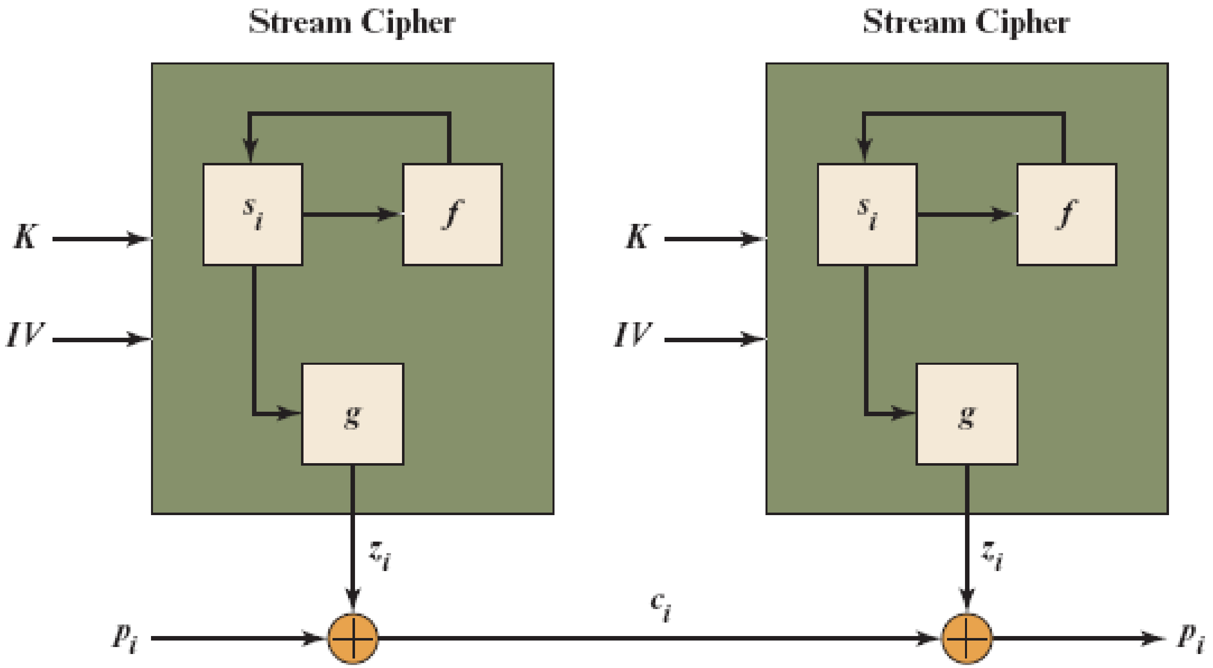
Stream Cipher

- Stream ciphers are a practical approximation to OTP.
  - OTP uses a **true random key stream**, of length equal to the plaintext message.
  - A stream cipher uses a **short key** (concatenated with a counter **IV**) as a **seed** to a **PRNG** to generate stream of random bits.



Generic Structure of a Typical Stream Cipher

- $C_i = Z_i \oplus P_i$ $stream$



plaintext p_i
ciphertext c_i
keystream z_i

key K
Initialization Value IV

state s_i
next-state function f
keystream function g

Stream Cipher

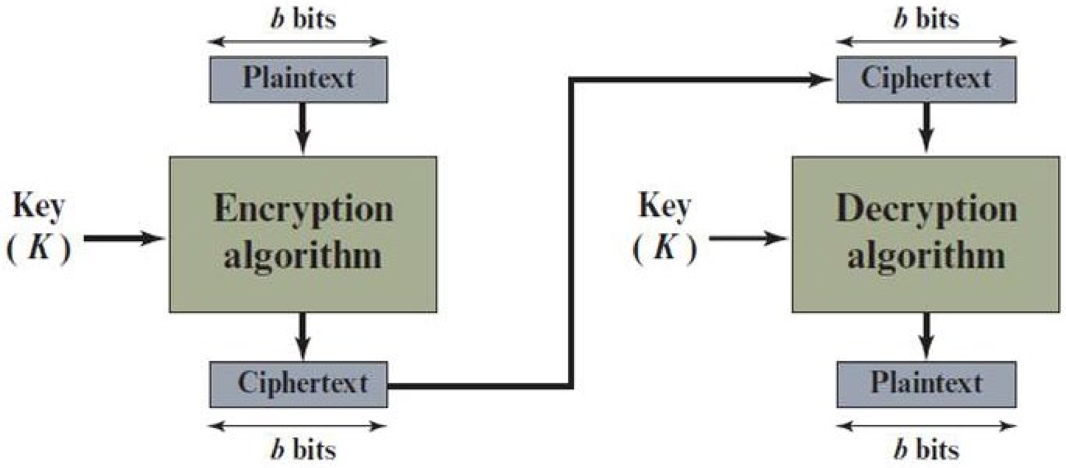
- Stream ciphers are useful when there is a need to encrypt **large** amounts of fast streaming data.
- Well suited to use in devices with very **limited memory and processing power**, called constrained devices.
- Fast and implementable in hardware.
  - Used in GSM, WiFi, TLS, and Bluetooth.
- Strength **dependent** on PRNG (period, algorithm) and the size/change-frequency of the key.

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

Block Cipher

- A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks.
- As with a stream cipher, the two users share a symmetric encryption key



(b) Block cipher

Block vs. Stream

- For applications that require encryption/decryption of a stream of data, a stream cipher might be the better alternative.
- For applications that deal with blocks of data (e.g., file transfer, email, database) block ciphers may be more appropriate.
- The **majority** of network-based symmetric cryptographic applications make use of **block ciphers.**
- However, either type of cipher can be used in virtually any application.

Block Cipher Design Features

- **Block size**: Larger block sizes mean **greater security** (through greater diffusion) but reduced encryption/decryption **speed** for a given algorithm.
  - Traditionally, a block size of **64** bits **has been** considered a reasonable tradeoff and was nearly universal in block cipher design.
  - However, AES uses a **128-bit** block size.

Algorithm	Block Size (Bits)
DES	64
AES	128

Block Cipher Design Features

- **Key size**: Larger key size means greater **security** but may decrease encryption/decryption **speeds**.
  - Greater **resistance to brute force**.
  - Key sizes of **64 bits or less** are now widely considered to be **inadequate**, and **128** bits has become a common size.

Algorithm	Block Size (Bits)	Key Size
DES	64	56
AES	128	128, 192, 256

Block Cipher Design Features

- **Number of rounds**: The essence of the cipher is that a single round offers inadequate security but that multiple rounds offer increasing security.
  - A typical size is **16** rounds.

Algorithm	Block Size (Bits)	Key Size	# of Rounds
DES	64	56	16
AES	128	128	10
		192	12
		256	14

Block Cipher Design Features

- **Round function F**: Greater complexity generally means greater resistance to cryptanalysis.

- **Subkey generation algorithm**: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

Reversible & Irreversible Mapping

- For the encryption to be **reversible** (i.e., for decryption to be possible), each mapping must produce a unique ciphertext block.

- Why is the one on the right not reversible?

Reversible Mapping		Irreversible Mapping	
Plaintext	Ciphertext	Plaintext	Ciphertext
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Padding

- **Block Size (B): DES: B=64b, K=56b. AES: B=128b, K=128b.**

```
...  | DD DD DD DD DD DD DD DD | DD DD DD DD |
```

- Question: What happens if the data doesn’t fit in the block size?
  - **Padding** is a technique used to ensure that the data to be encrypted fits perfectly into the block size of the cryptographic algorithm being used.

PKCS#5

- PKCS#5 (Public Key Cryptography Standards): If the length of the data is not an exact multiple of the block size, bytes are added to the data, and the value of each added byte is equal to the number of padding bytes:

$$n = B - (P \bmod B) \ (0 < n \leq B)$$

B is block size, P is plaintext, n is the # of bytes to be padded.

PKCS#5

- Do we need padding to encrypt the following data? And what is the value of n?

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD |
```

PKCS#5

$$n = B - (P\ mod\ B)\ (0 < n \leq B)$$

- B = block size, P = plaintext, n = # of bytes to be padded.

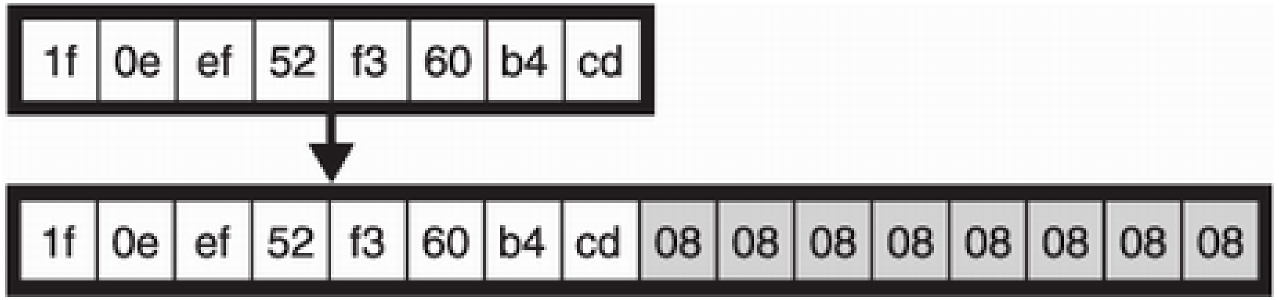
$$n = 8 - (4\ mod\ 8) = \mathbf{4}$$

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 04 04 04 04 |
```

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
06 06 06 06 06 06
07 07 07 07 07 07 07
```

PKCS#5

- PKCS#5: Pads even if P divides B:



- To remove the padding, you can easily check the value of the last byte of plaintext and interpret it as the length of padding to remove.

Padding

- ANSI X9.23: The block is padded with random bytes (many implementations use 00) and the **last byte** of the block is set to the **number of bytes added**.

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 04 |
```

- Block size is 8 bytes, and padding is required for 4 bytes (in hexadecimal format).

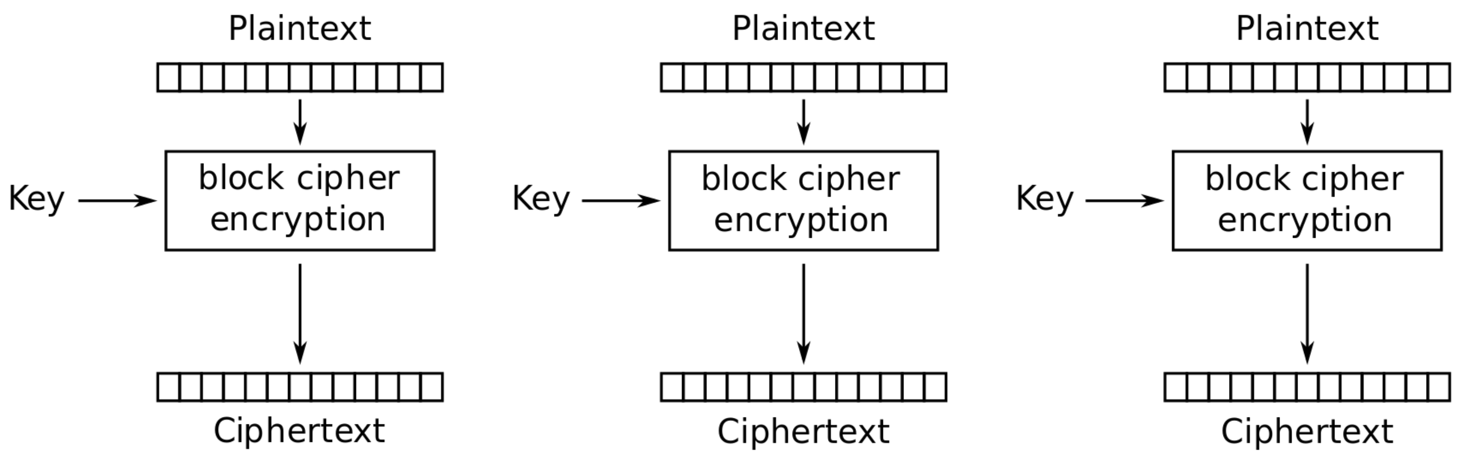
Substitution-Permutation Network (SPN)

- Encrypt Each Block:
  - SPN combines substitution and permutation.
    - Confusion (the CT/KEY relation) + Diffusion (the PT/CT relation).
  - Avalanche effect: 1b flip in PT changes ~½ the CT bits.

- Across Blocks:
  - Use **modes of operation**: ECB, CBC, CTR, ...

Modes of Operation

- If the amount of plaintext to be encrypted is greater than ***b* bits** , then the block cipher can still be used by breaking the plaintext up into ***b*-bit** blocks.



- When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise ➔ use **modes of operation.**

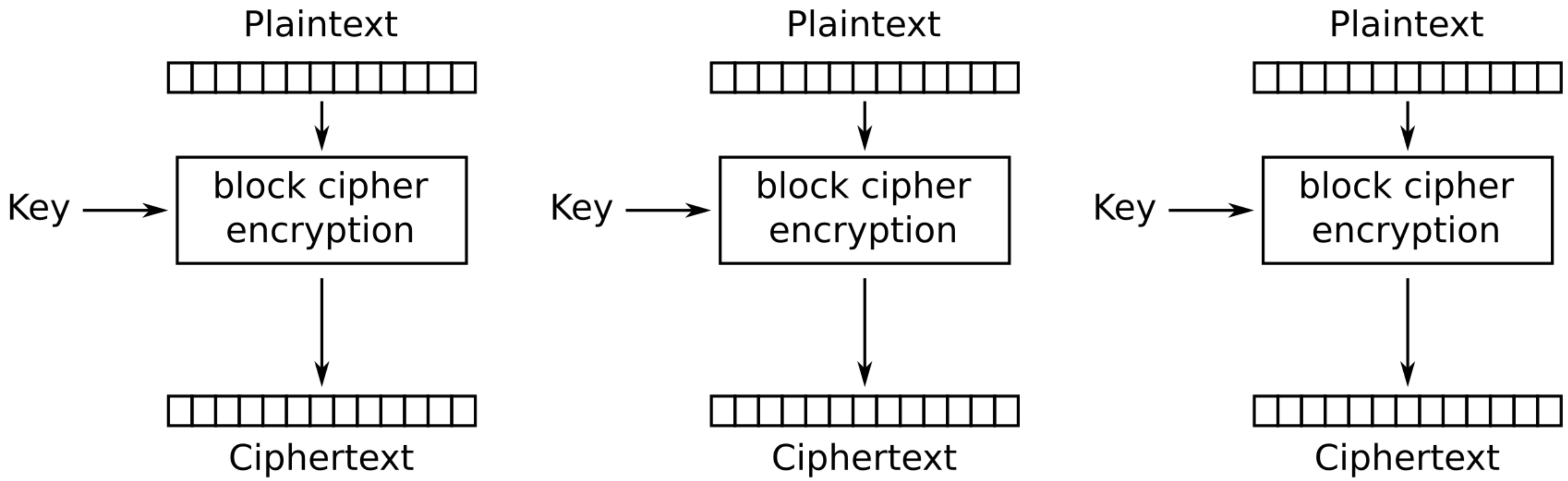
Modes of Operation

- A mode of operation is a **technique** for **enhancing** the effect of a cryptographic **algorithm**.

- These modes are intended for use with any **symmetric block cipher**, including triple DES and AES:
  - Electronic CodeBook (ECB)
  - Cipher Block Chaining (CBC)
  - Counter Mode (CTR)

Electronic Codebook (ECB) Mode

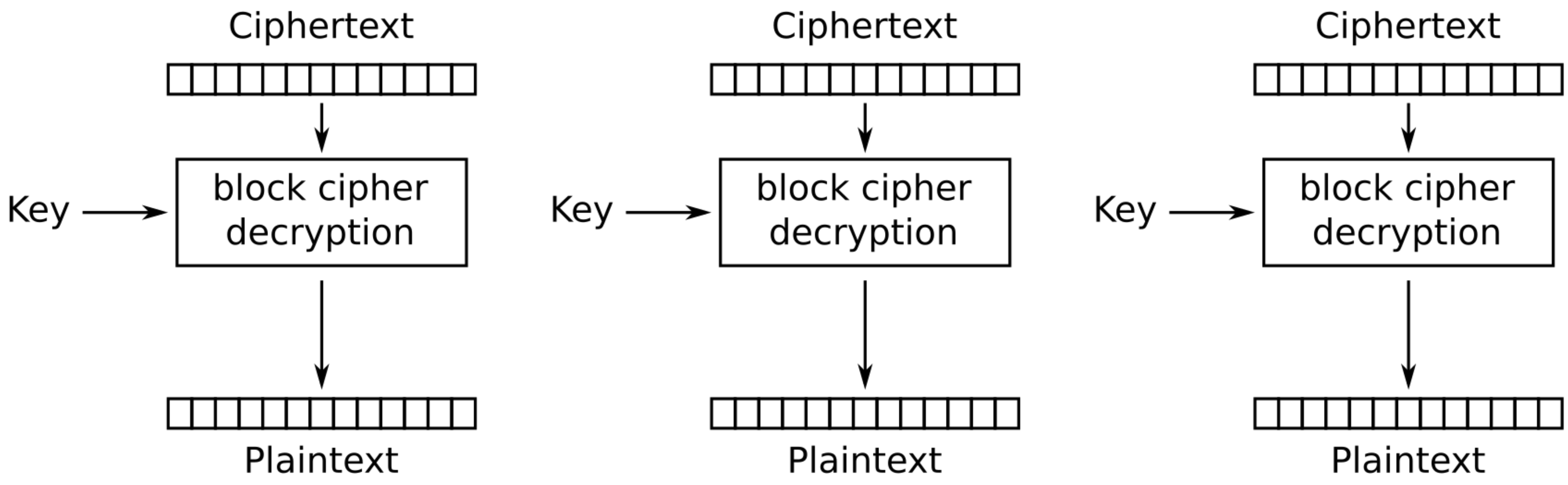
- Simplest mode: Plaintext is handled one block at a time.
- For a message longer than *b*-bits, break the message into *b*-bit blocks, padding the last block.



Electronic Codebook (ECB) mode encryption

ECB Decryption

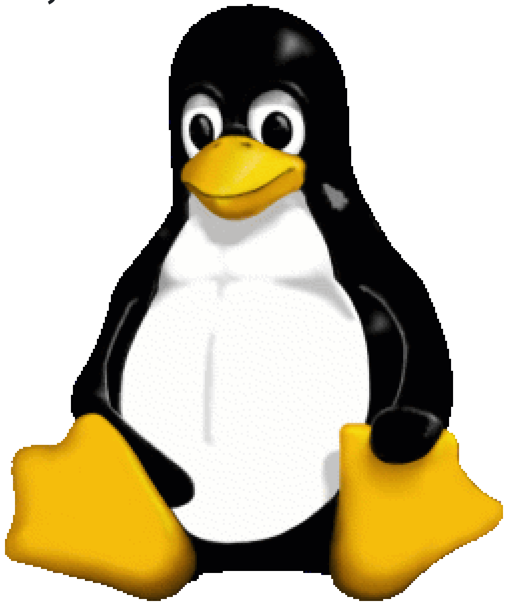
- Encryption and Decryption can be done in **parallel.**



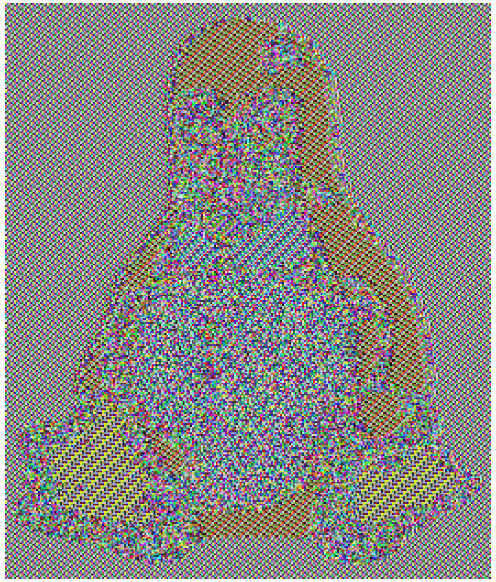
Electronic Codebook (ECB) mode decryption

ECB Penguin

- A *block cipher* always encrypts the same contents the same way, given the same key.
  - For lengthy messages, or if the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.
  - ECB **not secure**, not used.



Original Image

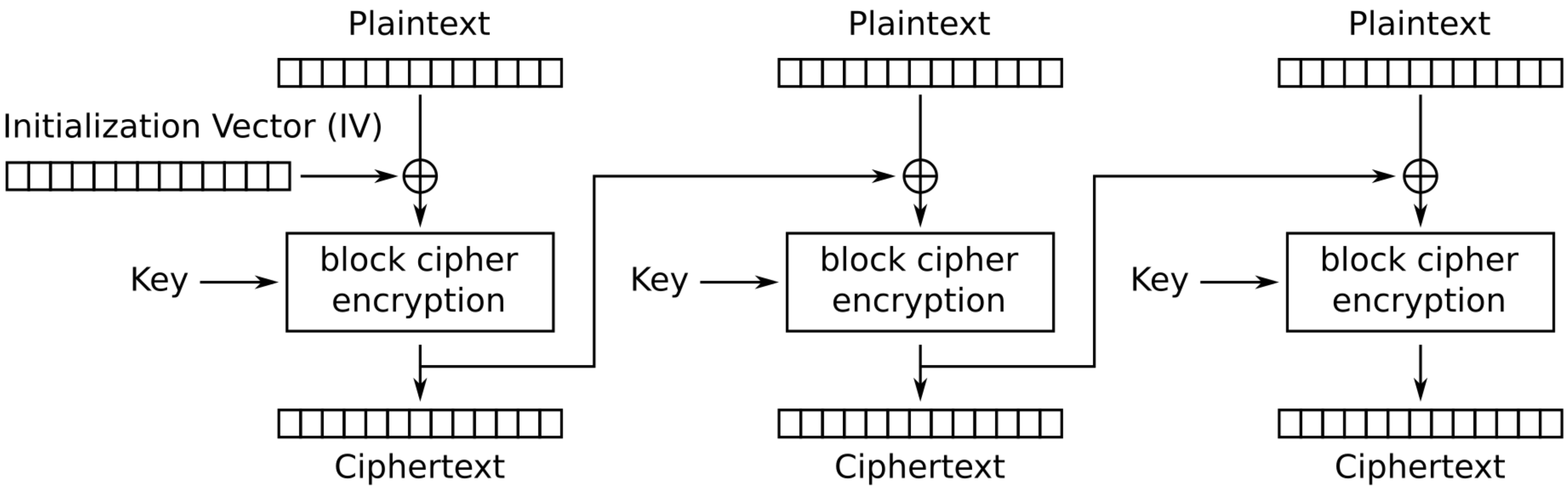


ECB

Source: https://github.com/robertdavidgraham/ecb-penguin

Cipher Block Chaining (CBC) Mode

- To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks.



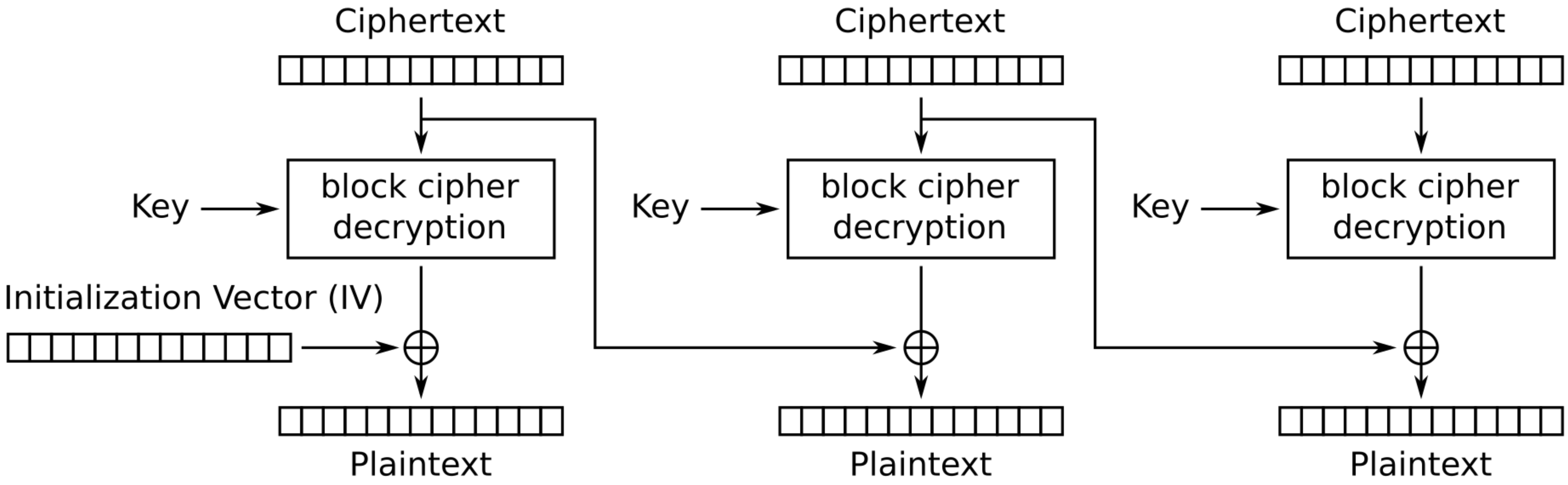
Cipher Block Chaining (CBC) mode encryption

Initialization Vector

- The Initialization Vector (IV):
  - Is a data block that is the same size as the cipher block.
  - Must be known to both the sender and receiver but be unpredictable by a third party.
    - IV is sent in the clear.
  - Generated using a random number generator.
  - Independent of the key, meaning even if the IV is known, it does not reveal information about the encryption key.
  - Shouldn’t be reused.
    - If the same IV is reused with the same key across multiple encryptions, an attacker could detect patterns.

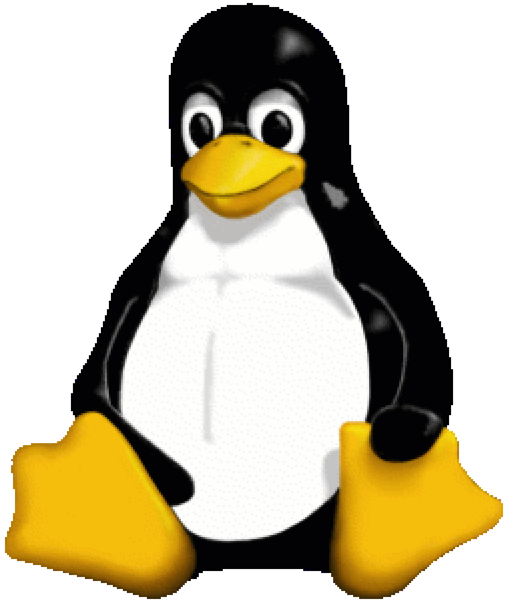
CBC Decryption

- **Question:** Encryption can't be done in parallel. Why? How about decryption?

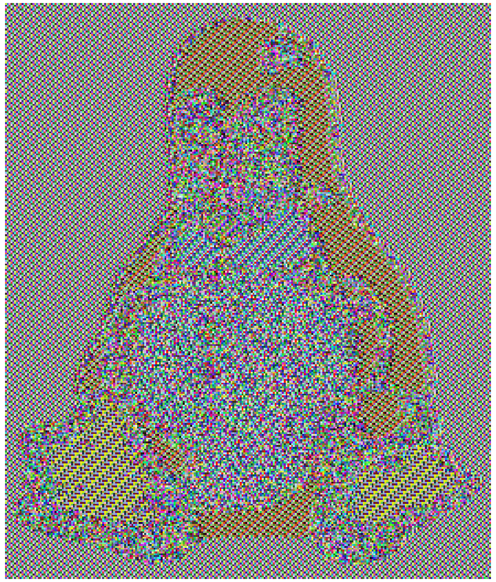


Cipher Block Chaining (CBC) mode decryption

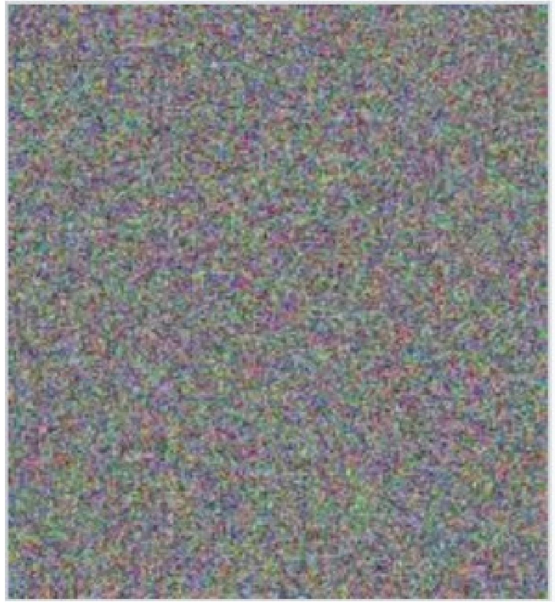
ECB vs. CBC



Original Image



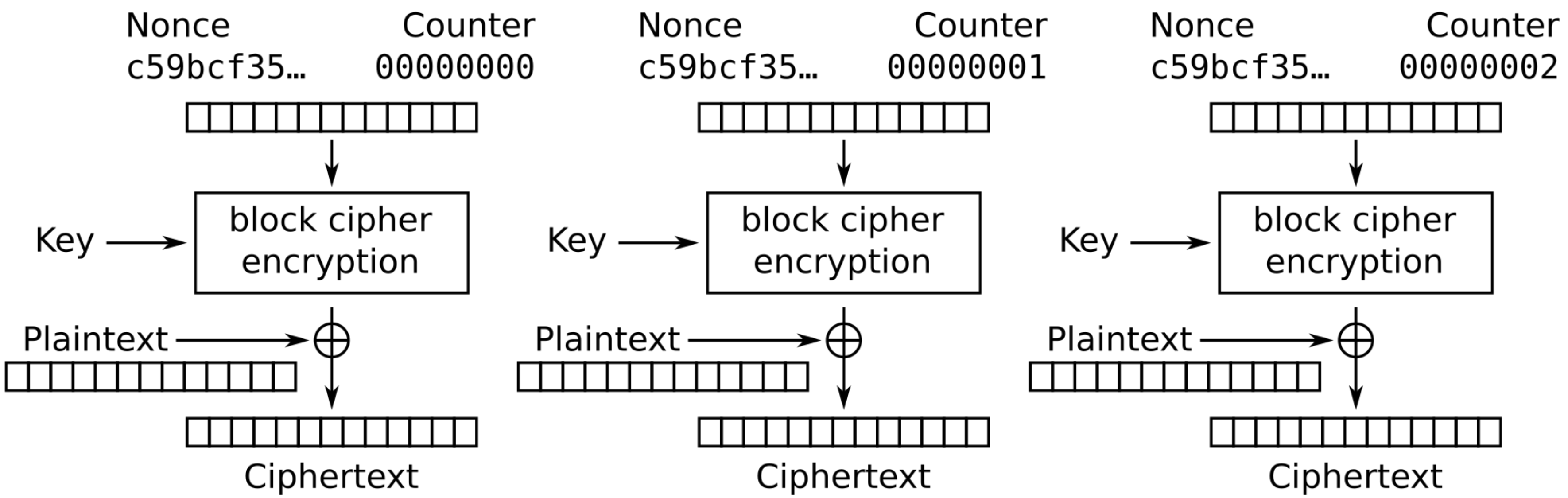
ECB



Modes other than ECB result in pseudo-randomness

Counter (CTR) Mode

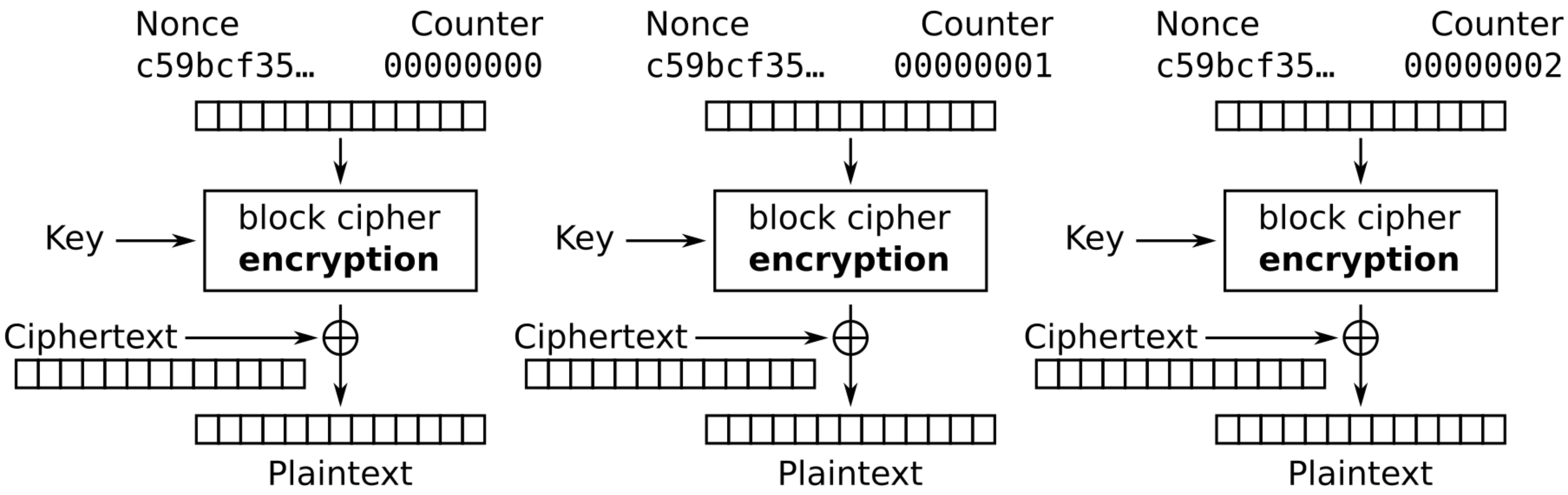
- A counter equal to the plaintext block size is used, and is initialized to some value and then incremented by 1 for each subsequent block.
- We can encrypt/decrypt in parallel.



Counter (CTR) mode encryption

CTR Decryption

- This mode allows block ciphers to be "streamy".
  - In stream cipher, you have a key stream generator that you **xor** with the plaintext.



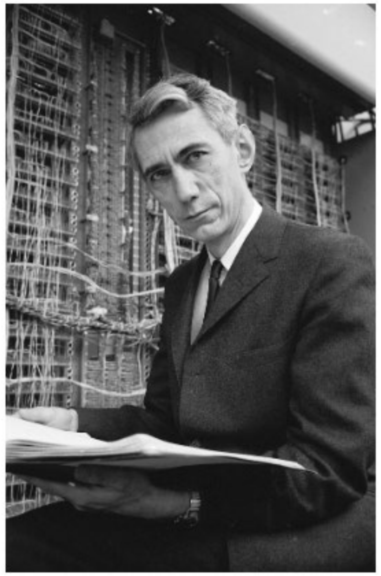
Counter (CTR) mode decryption

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

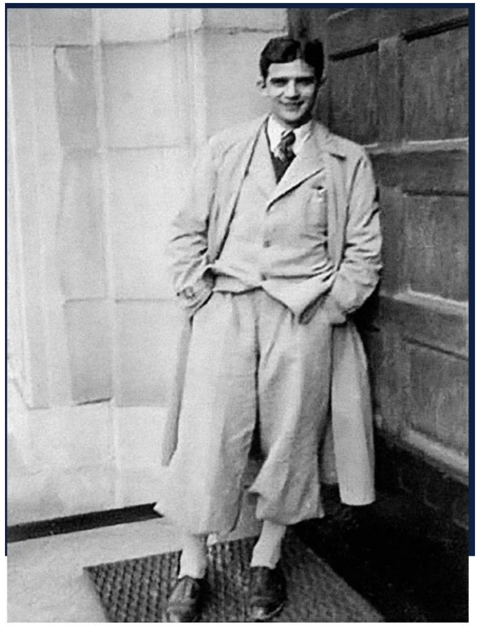
Feistel Cipher

- Feistel (along with Coppersmith) invented this cipher that **alternates substitutions and permutations** at IBM in 1973.
- A practical application of a proposal by Claude Shannon to develop a product cipher that **alternates confusion and diffusion** functions.
- The structure is used by many significant symmetric block ciphers currently in use.



A hundred years after his birth, Claude Shannon's fingerprints are on every electronic device we own. Photograph by Alfred Eisenstaedt / The LIFE Picture Collection / Getty

Horst Feistel	
Born	January 30, 1915 Berlin, Germany
Died	November 14, 1990 (aged 75)
Alma mater	MIT Harvard University
Known for	Feistel cipher



Horst Feistel, shown here in Zurich in 1934, studied in Switzerland and transferred to MIT to avoid serving in the German military.

COURTESY OF MRS. PEGGY FEISTEL CHESTER

Block Cipher Design

- **S-Box**
  Substitution via xor → very quick.

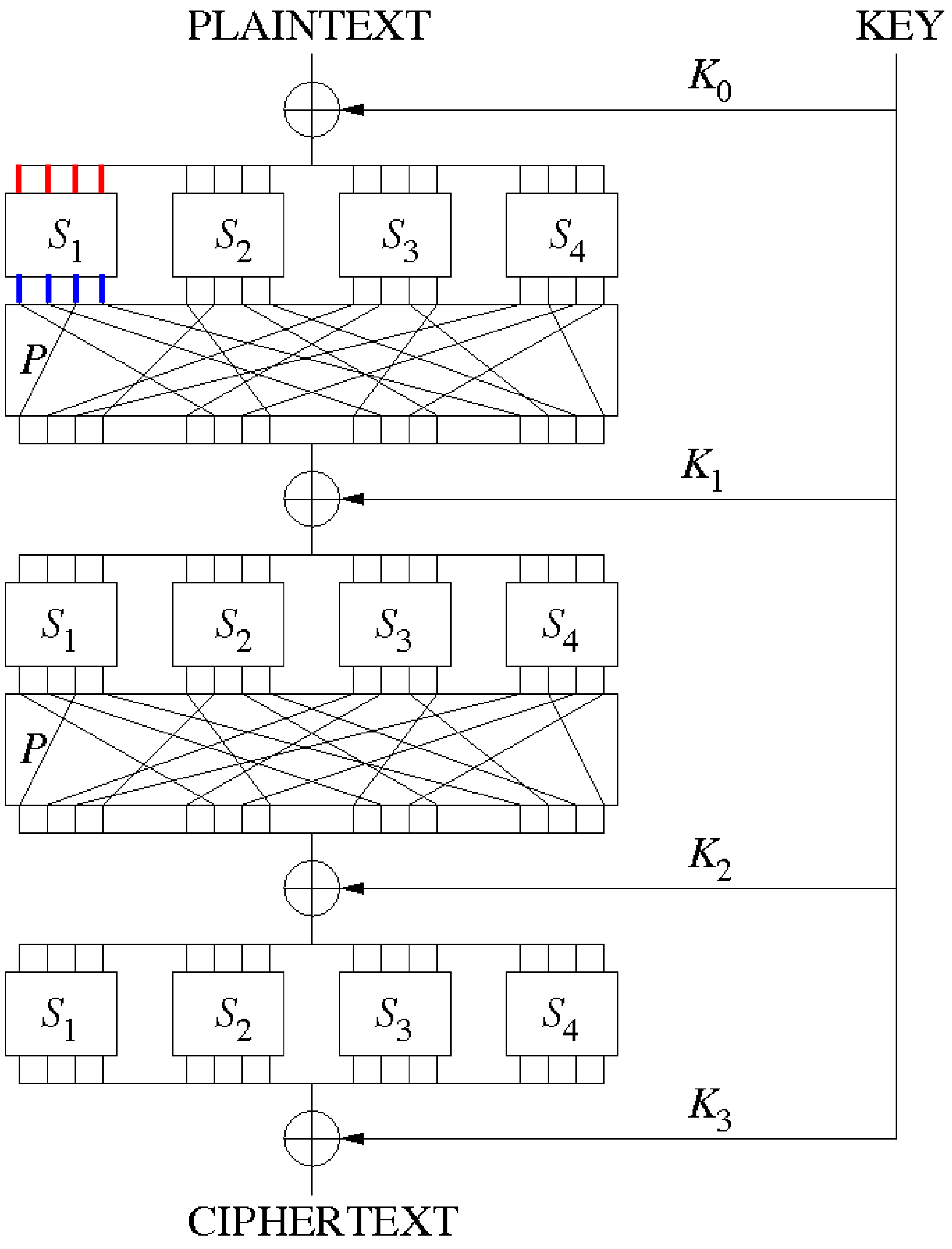
- **P Box**
  Permutation via $shift$ → very quick.

- **Rounds (SP Network, SPN)**
  Repeat with different subkeys (derived from key) to ensure confusion and diffusion.

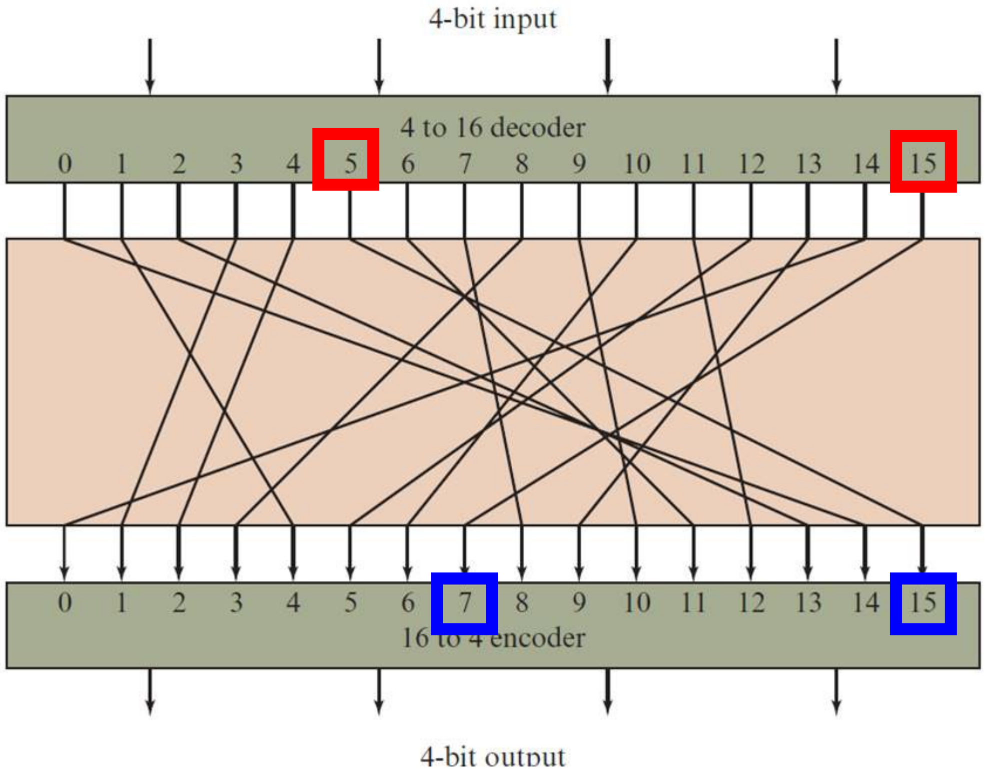
S-Box

- **S-Box:** Substitution that takes some number of input bits, m, and transforms them into some number of output bits, n, where n is not necessarily equal to m.

- Each row of an S-box defines a general reversible substitution.



S-Box Example (nxn)



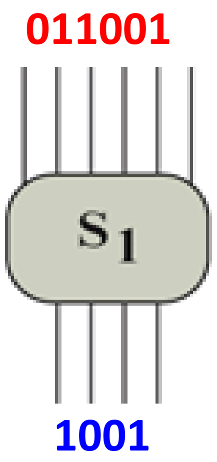
General n-bit-n-bit Block Substitution (shown with n = 4)

column 5 column 15

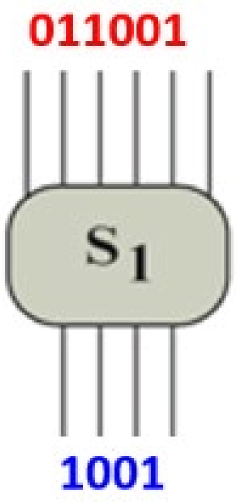
S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Box Example (mxn)

- Some S-Boxes can change the length of the input (e.g., S-Box of DES).
- Substitutes **6-bit** input for **4-bit** output.
  - Bits **0**, **5** define the **row.**
  - Bits **1,2,3,4** define the **column.**
- A table lookup is done.
- Example, in S1, for input **011001**, the output is **1001**.



S-Box Example (mxn)



- If input = **011001**, then output = **1001**.
- **011001**
  - Row # **01 (row 1)**
  - Column # **1100 (column 12)**

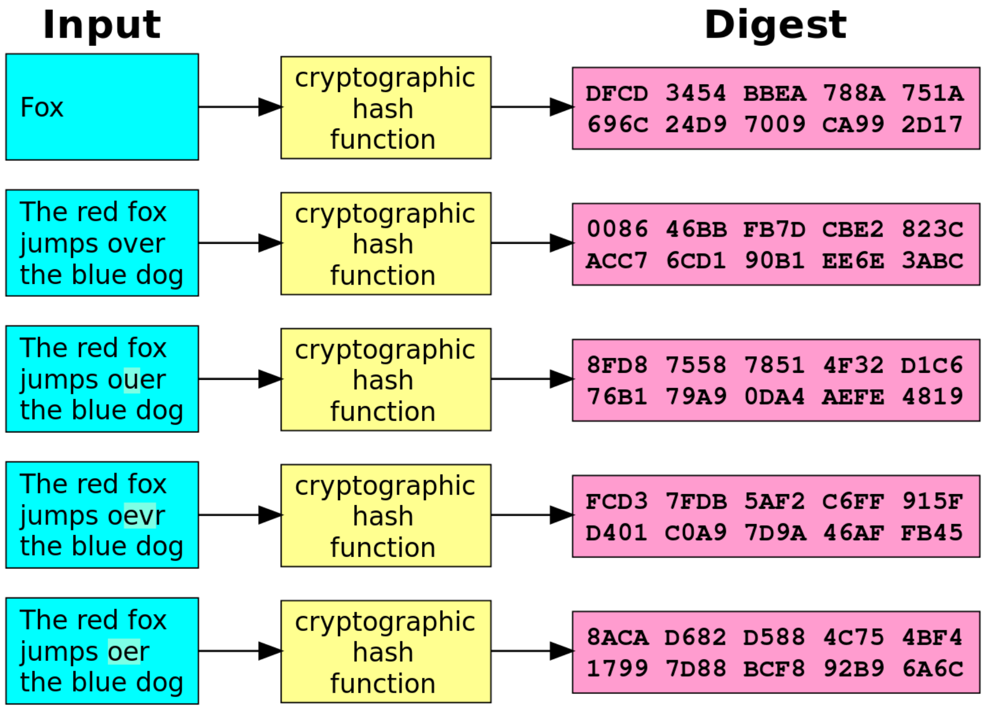
column 12

row 1

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	$\boxed{9}$	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Avalanche Effect

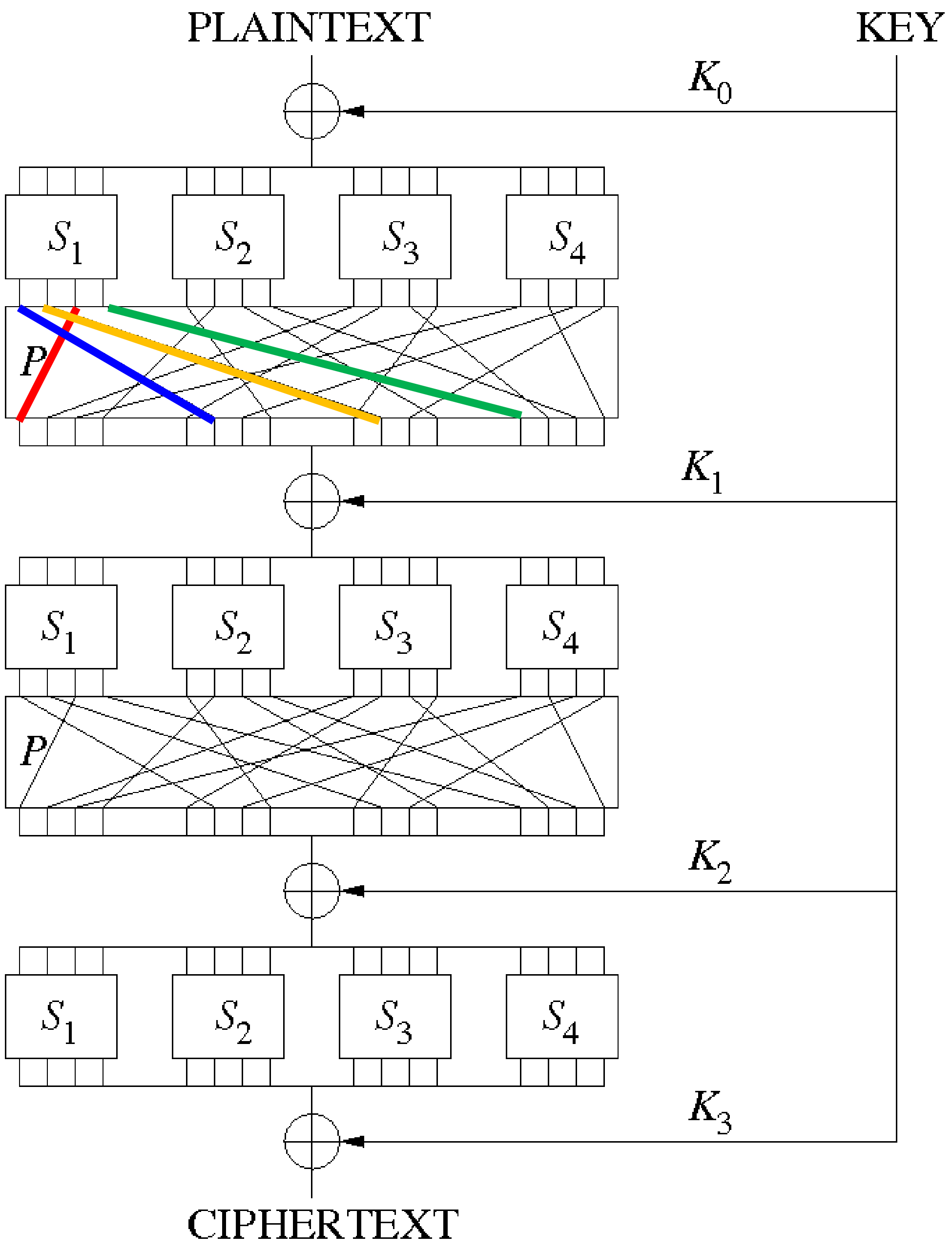
- A good S-Box will have the property that changing one input bit will change about half of the output bits (A**valanche Effect)**.



P-Box

- A P-Box is a permutation of all the bits: it takes the outputs of all the S-boxes of one round, permutes the bits, and feeds them into the S-boxes of the next round.

- A good P-box has the property that the **output bits** of any S-box are **distributed to as many S-box inputs** as possible.



By GaborPete https://commons.wikimedia.org/w/index.php?curid=6420152

P-Box Example

PT: attack23

PT in Ascii: [97, 116, 116, 97, 99, 107, 50, 51]

PT in Bin: **01100001 01110100 01110100 01100001 01100011 01101011 00110010 00110011**

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

Bit position in the plain-text block	To be overwritten with the contents of the bit position
1	58
2	50
3	42
….	….
64	7

P-Box Example

■ 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

PT1-24: 0110 0001 0111 0100 0111 0100

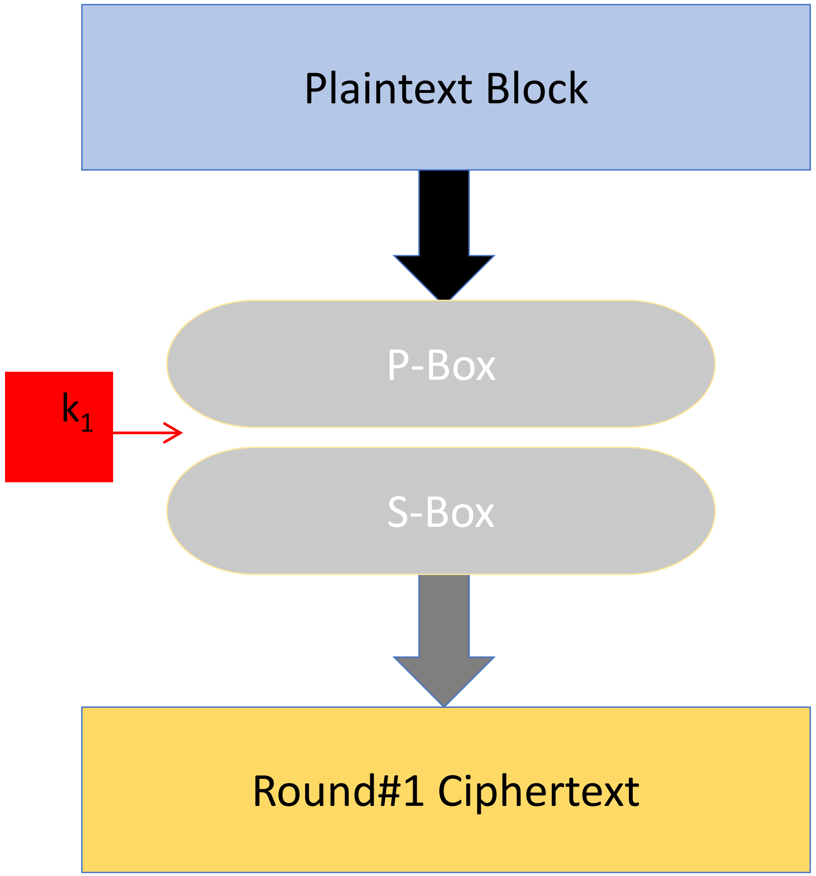
PT25-48: 0110 0001 0110 0011 0110 1011

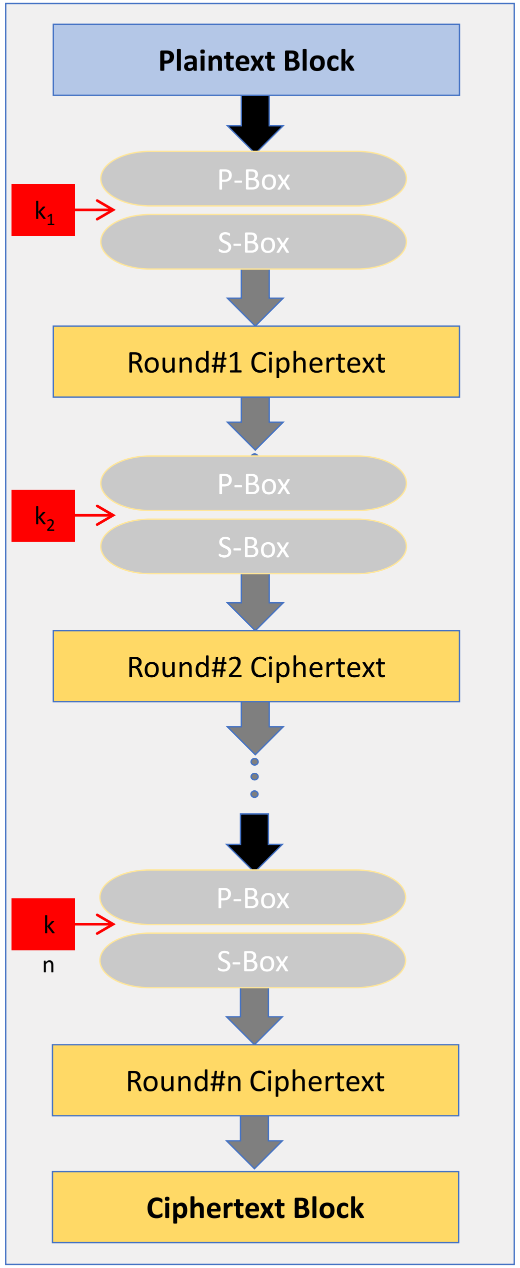
PT49-64: 0011 0010 0011 0011

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
PT in Binary	0	1	1	0	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0
Permuted	0	0	1	1	?	?	?	?																
	■	●	▼	♦																				

Based on the example given, what will be the permuted values here?

SPN Example

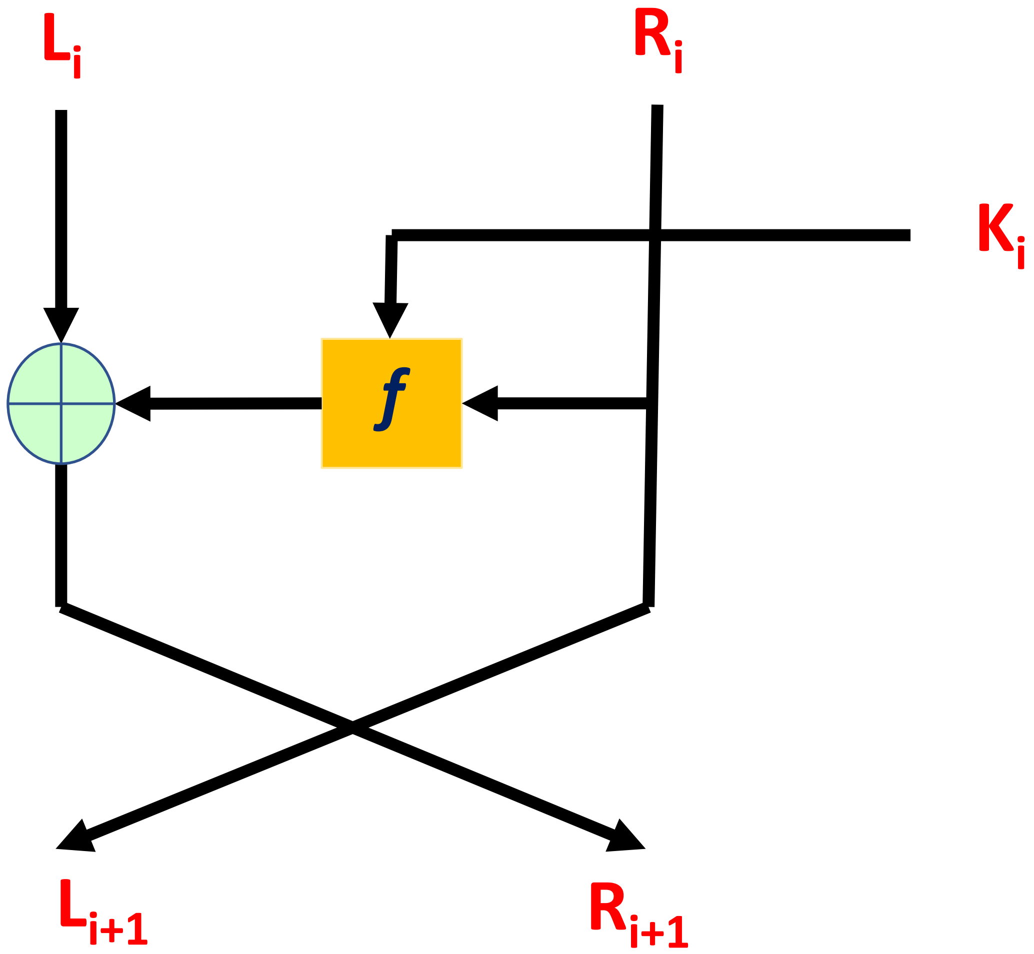

Plaintext Block
P-Box
k1
S-Box
Round#1 Ciphertext


Plaintext Block
P-Box
k1
S-Box
Round#1 Ciphertext
P-Box
k2
S-Box
Round#2 Ciphertext
P-Box
k
n
S-Box
Round#n Ciphertext
Ciphertext Block

Feistel Cipher

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

- Feistel is a design model used in many block ciphers.



Feistel Encryption (16 rounds)

- Round 1 Encryption:

$$LE_1 = RE_0$$

$$RE_1 = LE_0 \oplus F(RE_0, K_1)$$

- Round 16 Encryption

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$$

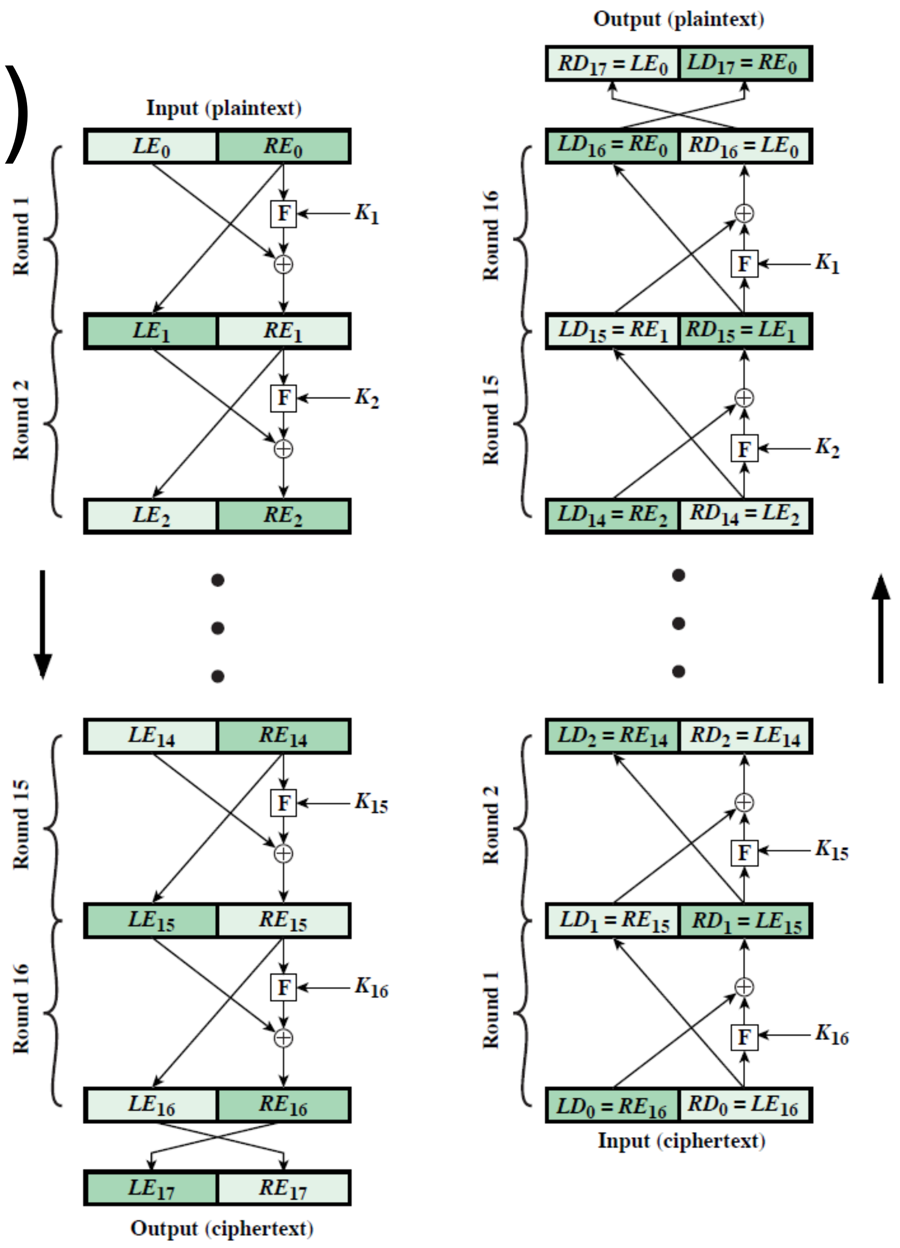
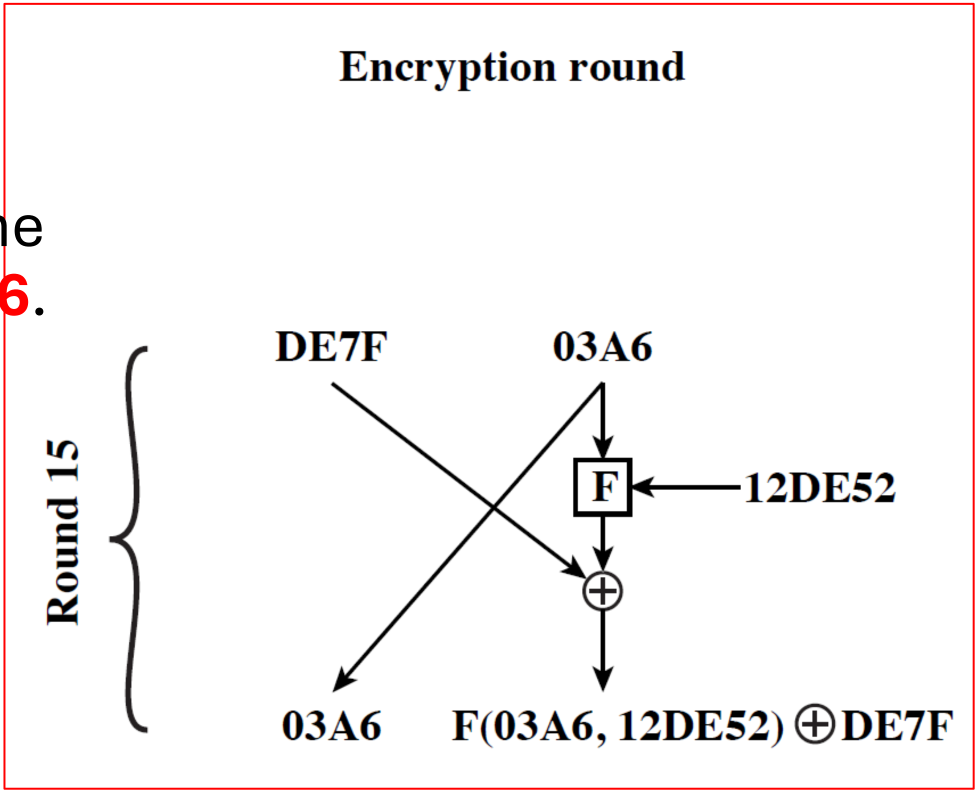


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

Feistel Example - Encryption

- Assume 32-Bit Blocks, (two 16-bit halves). **Key size: 24 bits.**
- At the end of encryption round 14, the value of the intermediate block (in hexadecimal) is **DE7F03A6**.
- $\boldsymbol{LE_{14}} = \mathbf{DE7F}$ and $\boldsymbol{RE_{14}} = \mathbf{03A6}$
- Assume $\boldsymbol{K_{15}} = \mathbf{12}\boldsymbol{DE}\mathbf{52}$.
- What is the value of $\boldsymbol{LE_{15}}$? and $\boldsymbol{RE_{15}}$?

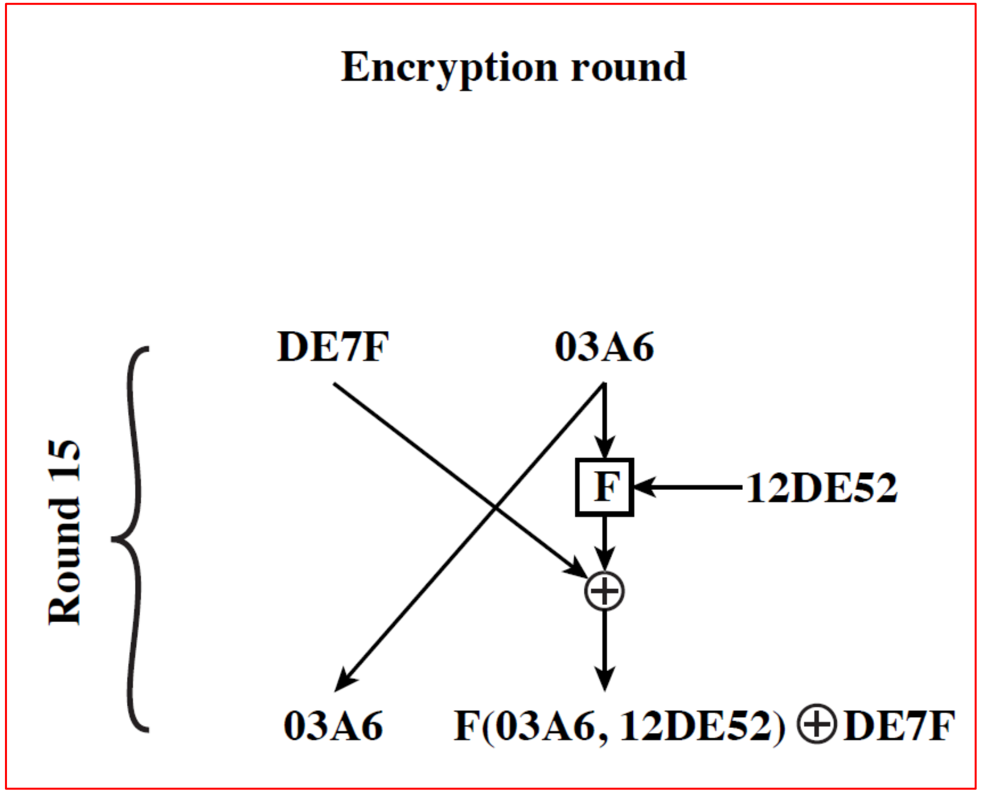


Feistel Example - Encryption

- $\boldsymbol{LE_{14}} = \mathbf{DE7F}$, $\boldsymbol{RE_{14}} = \mathbf{03A6}$, $\boldsymbol{K_{15}} = \mathbf{12}\boldsymbol{DE}\mathbf{52}$.
- What is the value of $\boldsymbol{LE_{15}}$? and $\boldsymbol{RE_{15}}$?

- $L_i = R_{i-1}$
- $\boldsymbol{LE_{15}} = \boldsymbol{RE_{14}} = \mathbf{03A6}$

- $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
- $\boldsymbol{RE_{15}} = \boldsymbol{LE_{14}} \oplus F(RE_{14}, K_{15}) = \mathbf{DE7F} \oplus \mathbf{F}(\mathbf{03A6}\, , \mathbf{12DE52})$



Feistel Example - Decryption

Given that

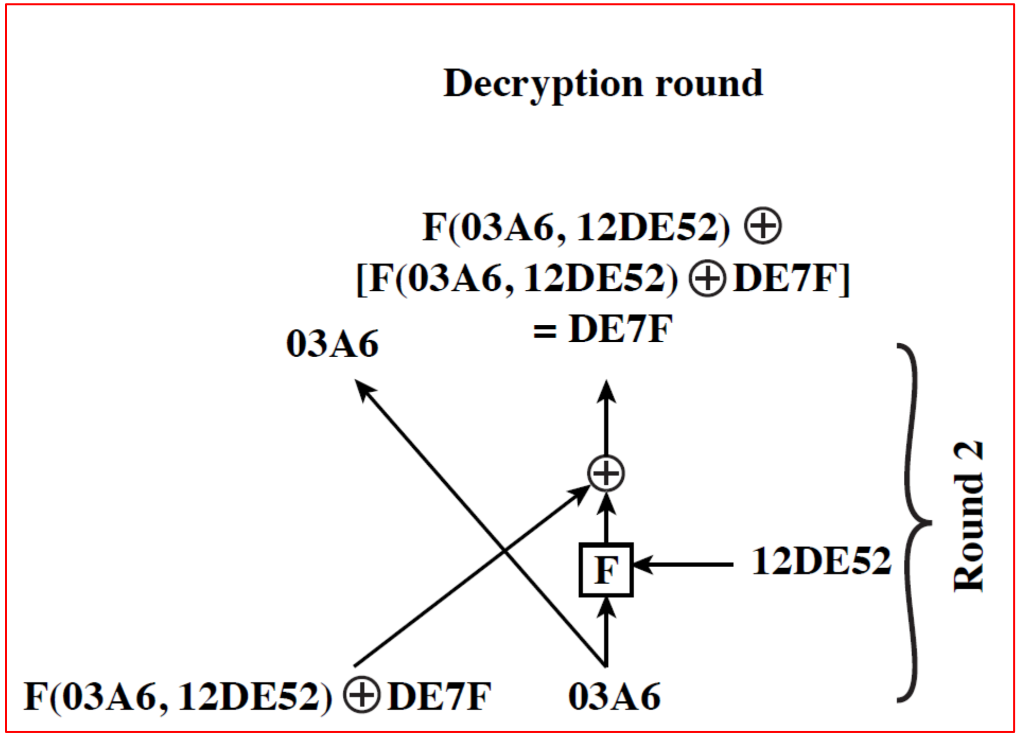
$\boldsymbol{LD_1} = \mathbf{F}(\mathbf{03A6}, \mathbf{12DE52}) \oplus \mathbf{DE7F}$

$\boldsymbol{RD_1} = \mathbf{03A6}\, ; \boldsymbol{K_2} = \mathbf{12DE52}$

- What is the value of $\boldsymbol{LD_2}$ and $\boldsymbol{RD_2}$ at the end of round 2?

$L_i = R_{i-1}$

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



Feistel Example - Decryption

We know that:

$LD_1 = \mathrm{F}(03A6\,,12DE52) \oplus DE7F$

$RD_1 = 03A6$

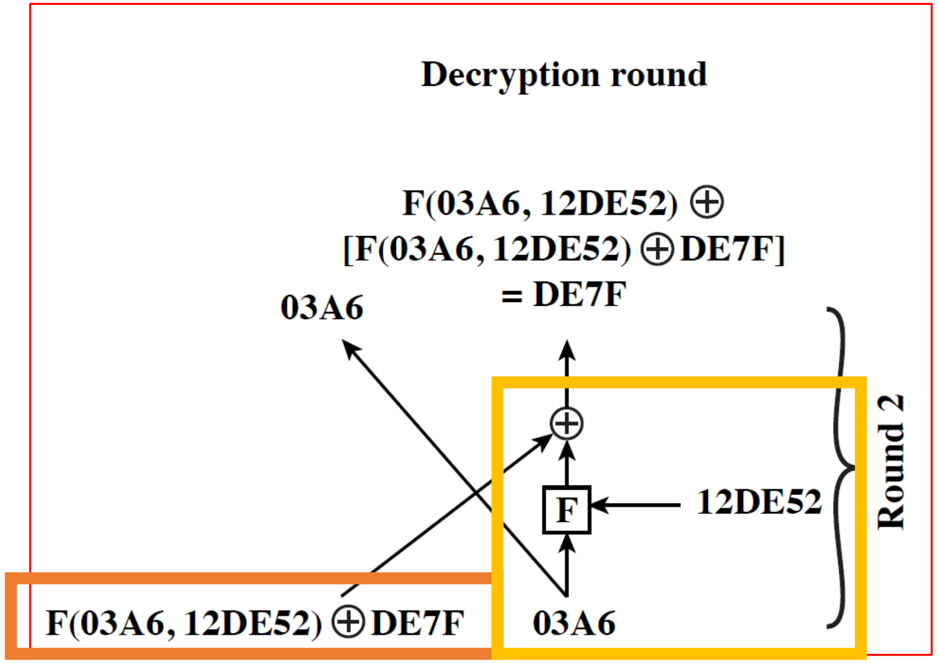
$L_i = R_{i-1}$

- $LD_2 = RD_1 = 03A6$

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

- $RD_2 = LD_1 \oplus \mathrm{F}(RD_1, K_2)$
- $RD_2 = \boxed{[\mathrm{F}(03A6\,,12DE52) \oplus DE7F]} \oplus \boxed{\mathrm{F}(03A6\,,12DE52)}$

$= DE7F$



Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

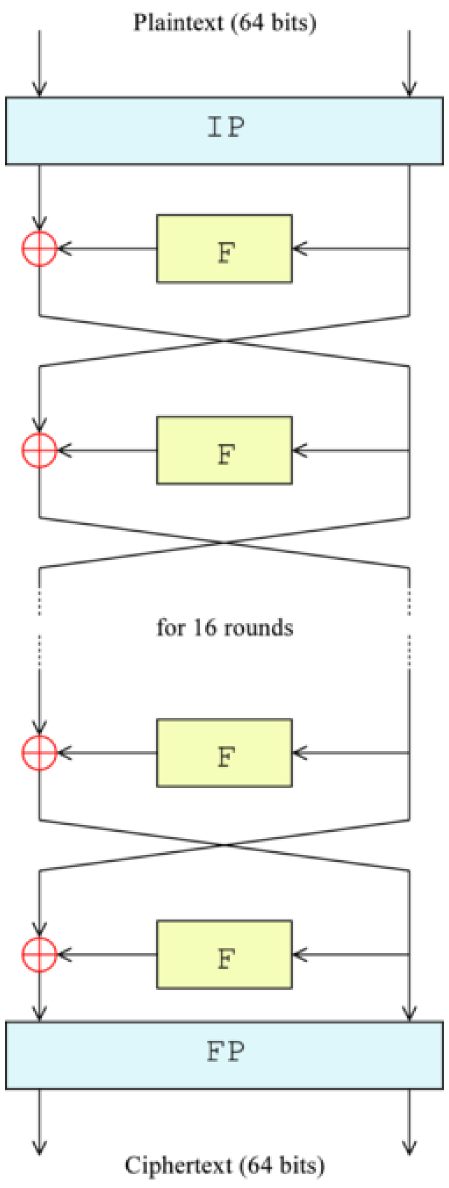
Data Encryption Standard (DES)

- DES was the result of a research project set up by IBM in the late 1960.
- Issued in 1977 by the National Bureau of Standards (now NIST).
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001.

NIST
National Institute of
Standards and Technology

Data Encryption Standard (DES)

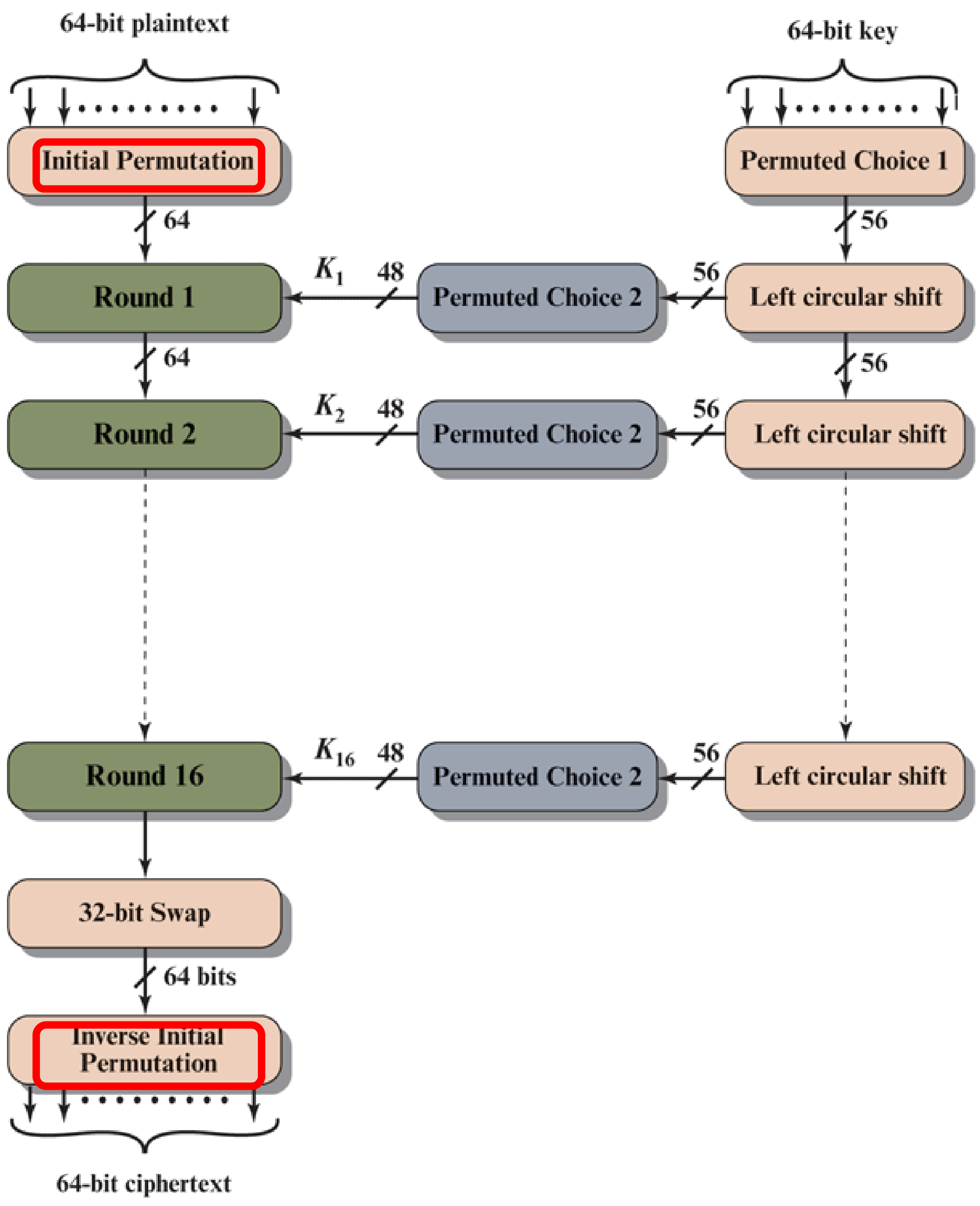
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - Data are encrypted in **64-bit blocks** using a **56-bit key.**
  - The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
  - The same steps, with the same key, are used to reverse the encryption.



The overall Feistel structure of DES

DES Overall Structure

- With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.



DES Initial Permutation

- The initial permutation and its inverse are defined by tables.

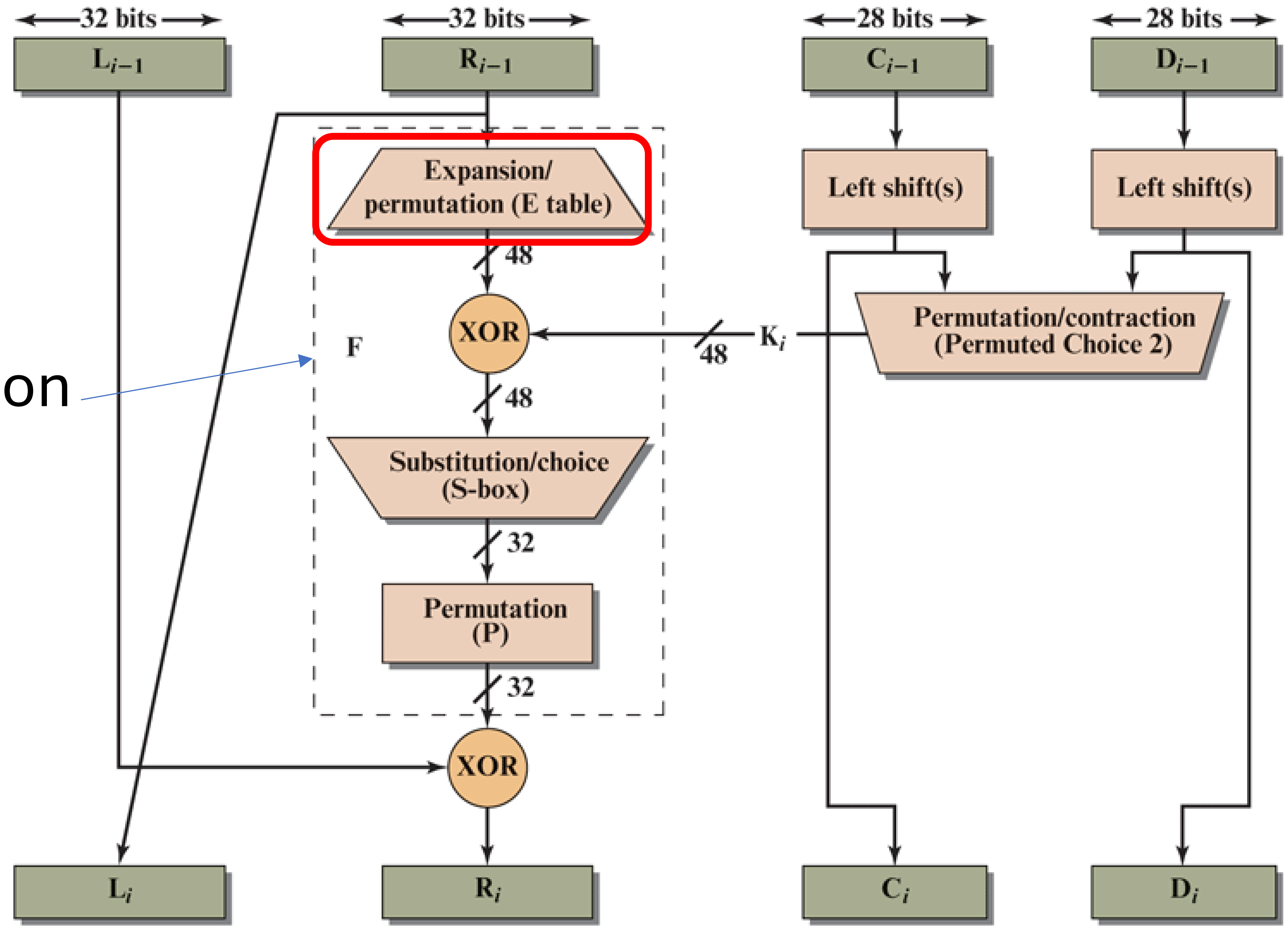
(a) Initial Permutation (IP)							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

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

Single Round of DES Algorithm

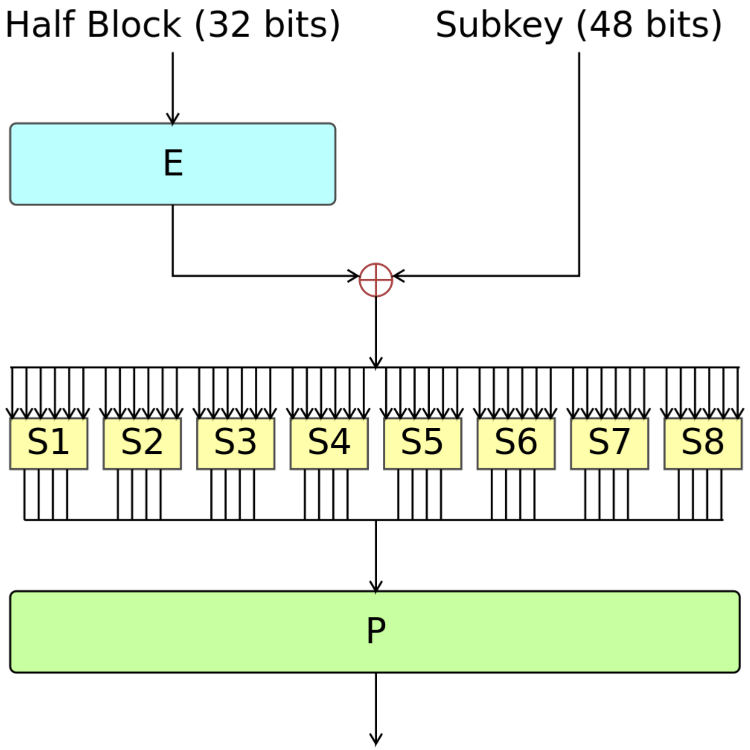
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

- Note the Feistel F function



DES Expansion/Permutation (E Table)

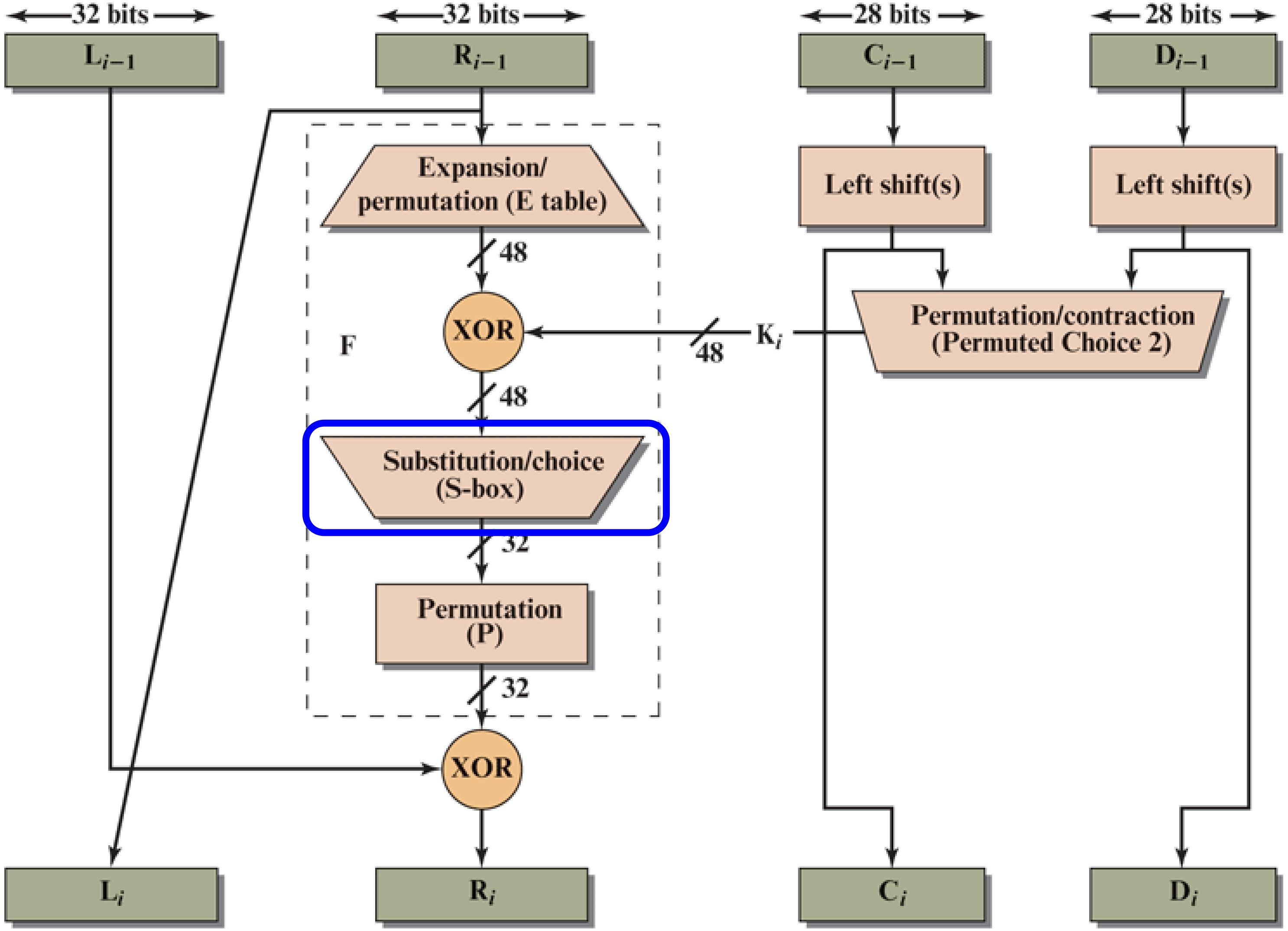
- The R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits.



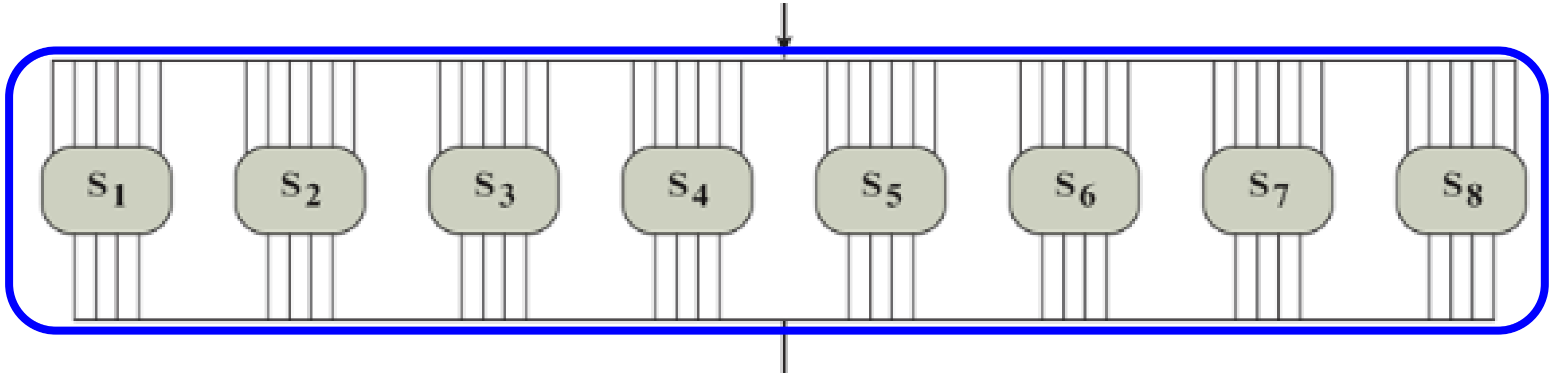
(c) Expansion Permutation (E)					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES Substitution (S-Boxes)

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



DES S-Boxes



column 12

row 01

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

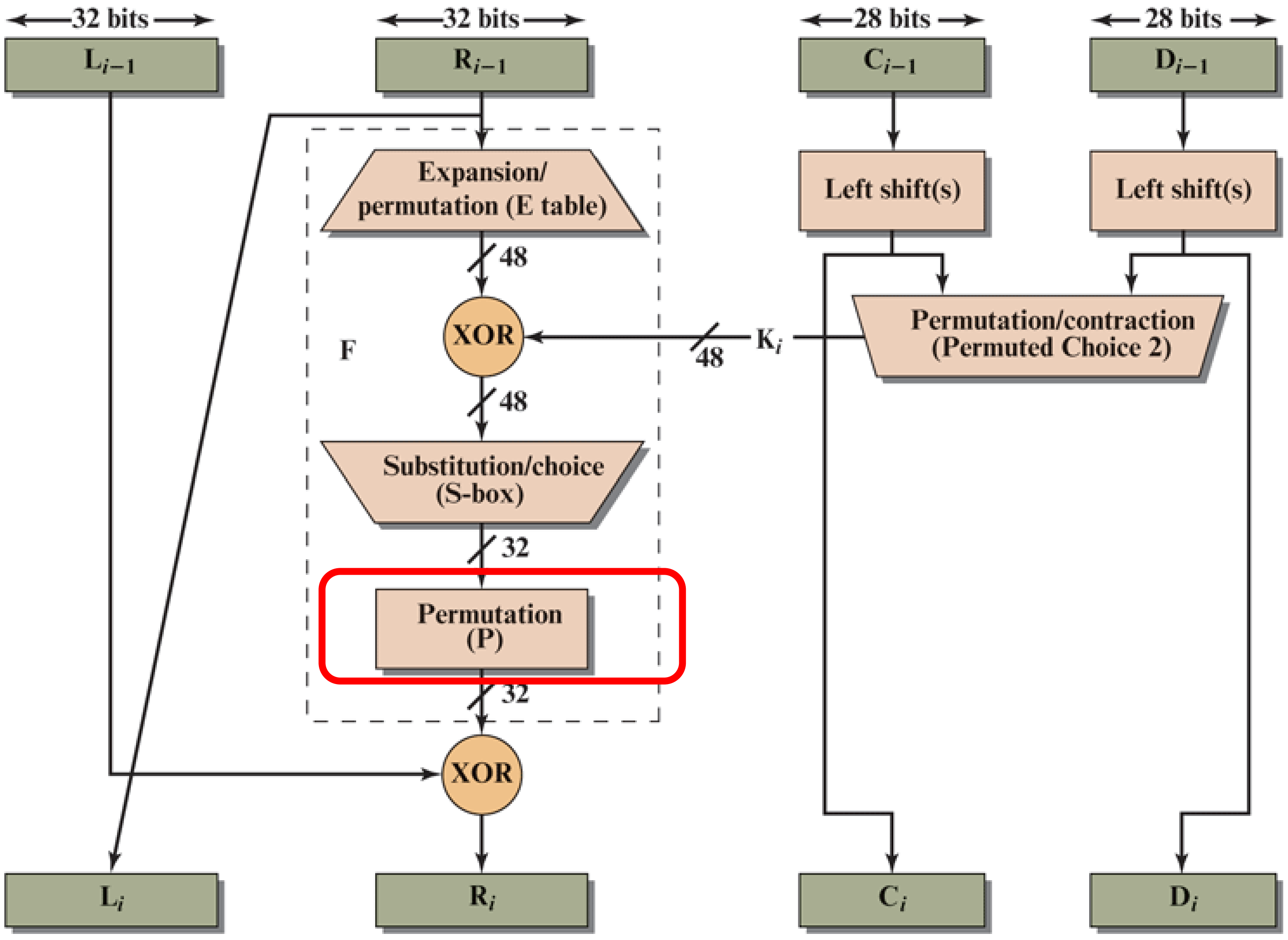
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Recall: Input = 011001, Output = 1001.

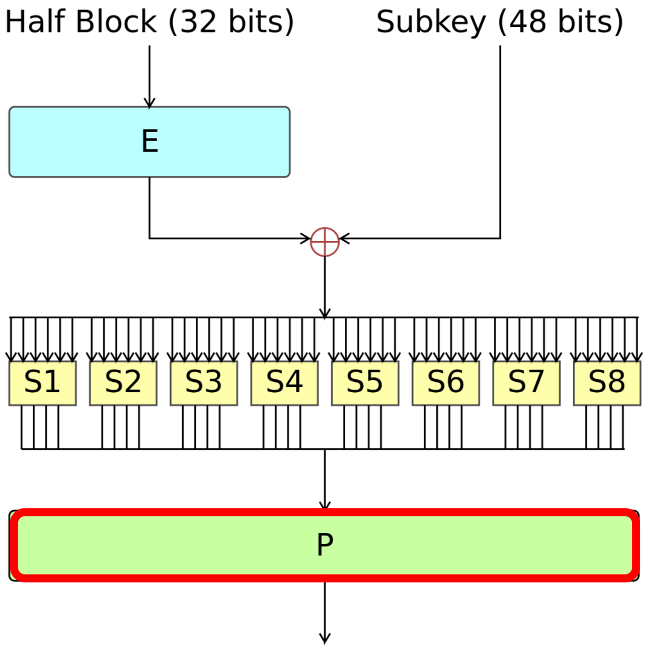
DES Permutation (P)

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

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



DES Permutation (P)

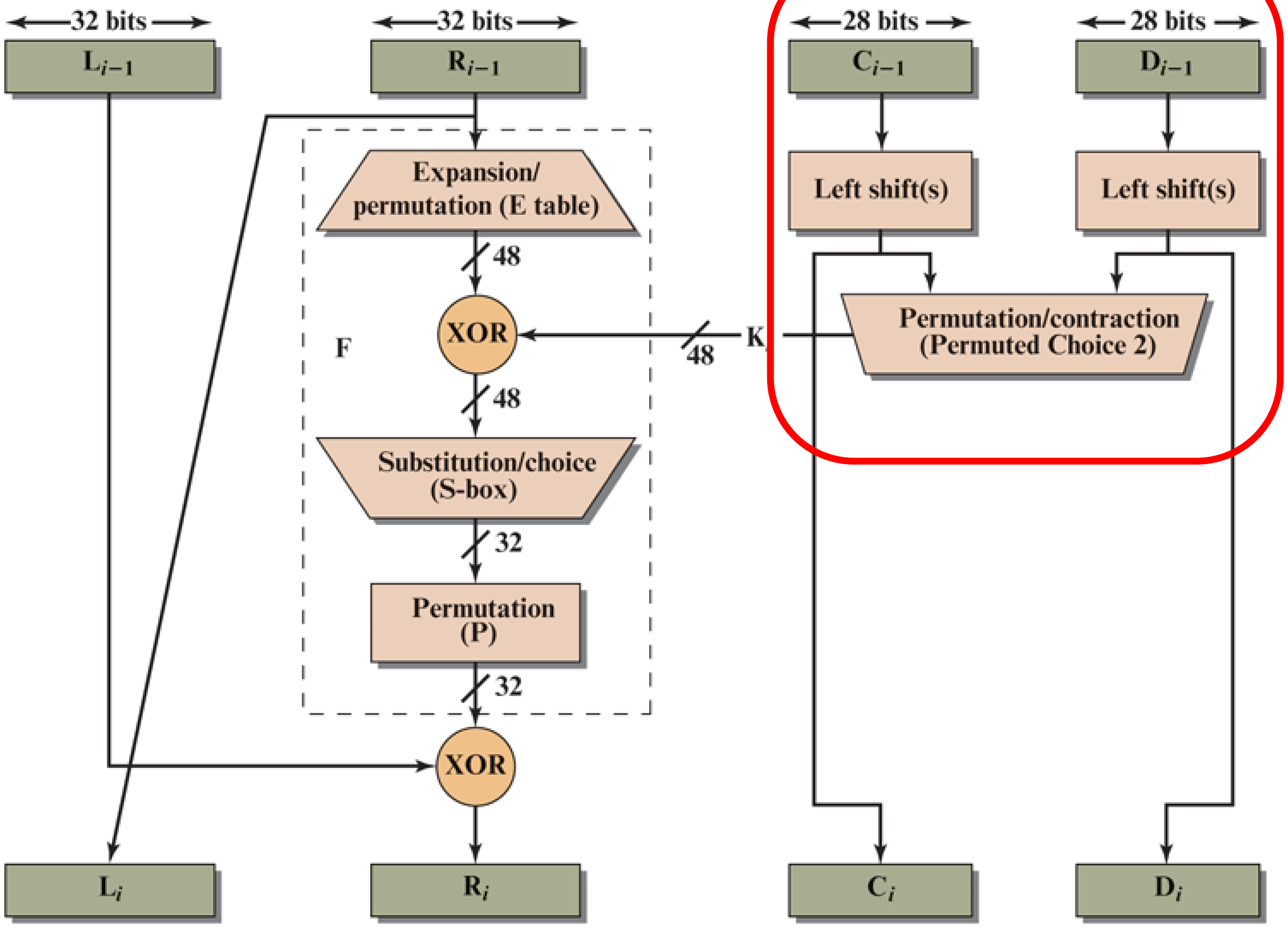


(d) Permutation Function (P)							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

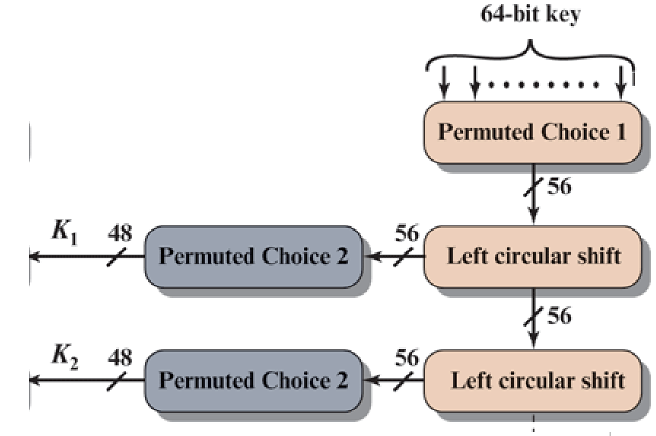
DES Key Expansion

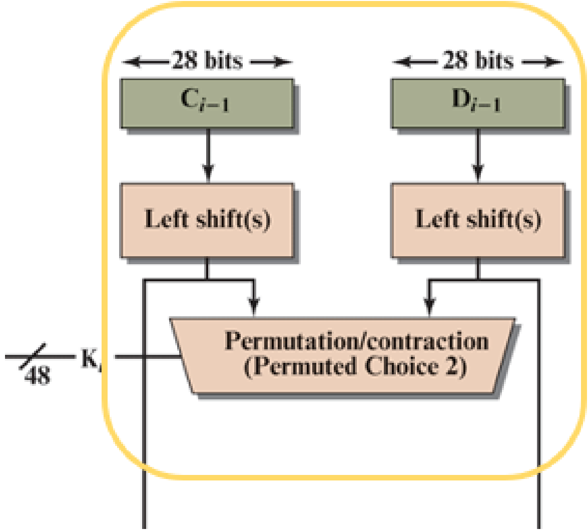
$$L_i = R_{i-1}$$

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



DES Key Schedule





Not used

Table C.3 DES Key Schedule Calculation

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

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

DES Example

- Given

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

- Find CT and Subkey #1 in hexadecimal after the **first round**?

Find CT & Subkey1 in hexadecimal after the first round

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

DES Example

Step 1 – Create 16 Subkeys, each of which is 48-bits long.

00000000 01111111 11122222 22222333 33333334 44444444 45555555 55566666
12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234

K = 0000111~~1~~ 0001010~~1~~ 0111000~~1~~ 1100100~~1~~ 0100011~~1~~ 1101100~~1~~ 1110100~~0~~ 0101100~~1~~
PC-1 = 01101000 11111100 01000100 10100001 00010001 00111110 10010110

C_0 = 01101000 11111100 01000100 1010
D_0 = 00010001 00010011 11101001 0110
C_1 = 11010001 11111000 10001001 0100 (Rotate C_0 left, 28 bits)
D_1 = 00100010 00100111 11010010 1100 (Rotate D_0 left, 28 bits)

PC-2 = 01111000 00110011 11000011 00100000 11011010 01110000 (48 bits, Subkey#1)

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

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

Find CT & Subkey1 in hexadecimal after the first round

DES Example

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

Step 2 – Permute plaintext.

```
     00000000 01111111 11122222 22222333 33333334 44444444 45555555 55566666
     12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234
PT = 00000010 01000110 10001010 11001110 11101100 10101000 01100100 00100000
IP = 01011010 00000000 01011010 00000000 00111100 11110000 00111100 00001111
```

L_0 = 01011010 00000000 01011010 00000000

R_0 = 00111100 11110000 00111100 00001111

For $n = 1$ **we have**

K_1 = 011110000 00110011 11000011 00100000 11011010 01110000

$L_1 = R_0$ = 00111100 11110000 00111100 00001111

$R_1 = L_0 \oplus F(R_0, K_1)$

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

Find CT & Subkey1 in hexadecimal after the first round

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

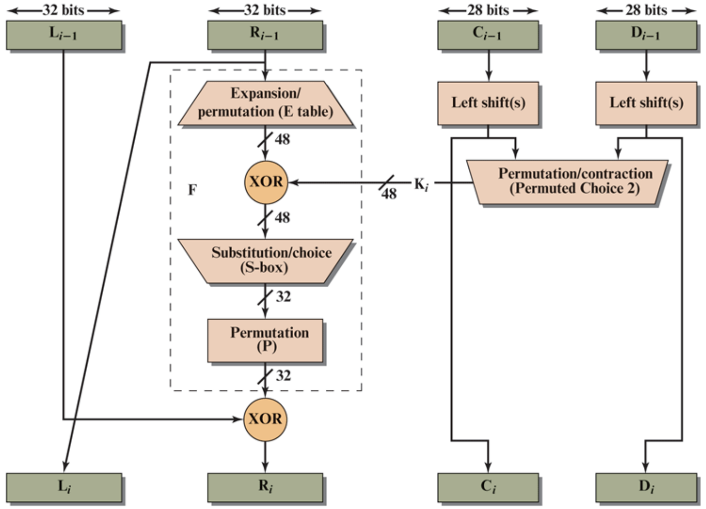
DES Example

Step 3 – Calculate $F(R_0, K_1)$

```
              00000000 01111111 11122222 22222333 33333334 44444444 45555555 55566666
              12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234
R0          = 00111100 11110000 00111100 00001111
E(R0)       = 10011111 10010111 10100000 00011111 10000000 01011110
K1          = 01111000 00110011 11000011 00100000 11011010 01110000
E(R0)⊕K1    = 11100111 10100100 01100011 00111111 01011010 00101110
```

(c) Expansion Permutation (E)

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



Find CT & Subkey1 in hexadecimal after the first round

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

DES Example

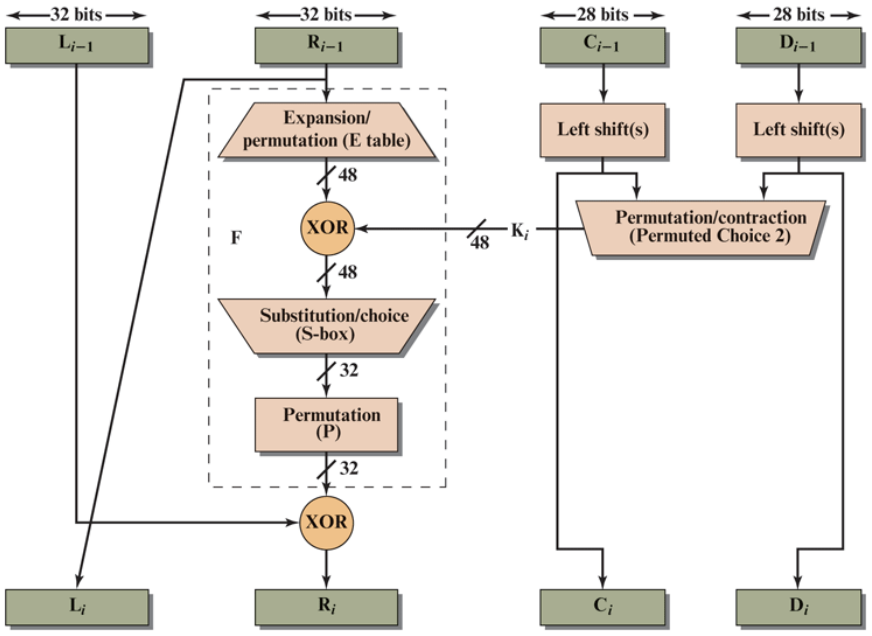
Step 4 – Perform Substitutions **S(E (R$_0$) $\oplus$ K_1)**

E (R$_0$) $\oplus$ K_1 = 11100111 10100100 01100011 00111111 01011010 00101110

E (R$_0$) $\oplus$ K_1 = 111001 111010 010001 100011 001111 110101 101000 101110 **(same but 6 bits * 8)**

S(E (R$_0$) $\oplus$ K_1) = 1010 0011 0010 1111 0001 0001 1100 0010

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$



Col 1 (0001)

Col 12 (1100)

Row11

Row11

Definition of DES S-Boxes

	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
S1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
S2	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
S3	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
S4	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
S5	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
S6	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
S7	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
S8	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Find CT & Subkey1 in hexadecimal after the first round

DES Example

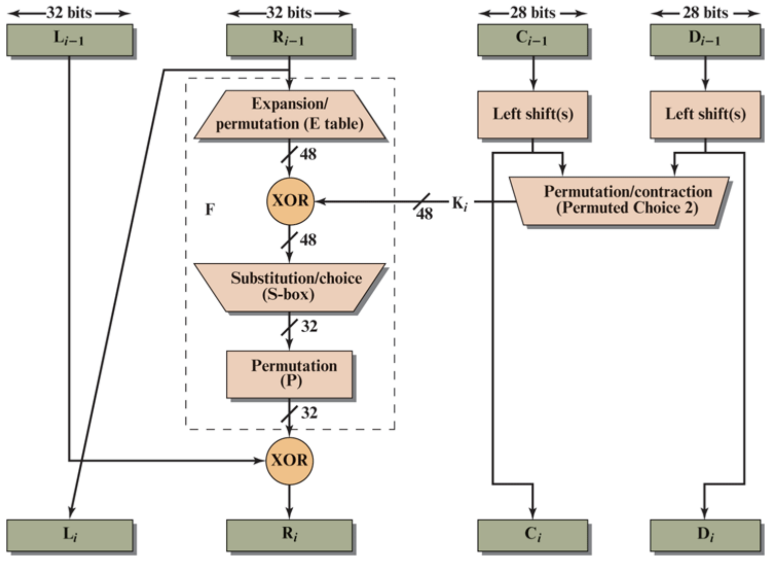
Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

Step 5 – Apply Permutation **P(S(E (R$_0$) $\oplus$ K_1))**

```
00000000 01111111 11122222 22222333 33333334 44444444 45555555 55566666
12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234
```

S(E (R$_0$) $\oplus$ K_1))= 10100011 00101111 00010001 11000010

P(S(E (R$_0$) $\oplus$ K_1)) = 11100000 11010010 01110010 01000101



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

Find CT & Subkey1 in hexadecimal after the first round

Plaintext	0x02468aceeca86420
Key	0x0f1571c947d9e859

DES Example

Step 6 – Calculate R_1

$F(R_0, K_1)$ = 11100000 11010010 01110010 01000101

L_0 = 01011010 00000000 01011010 00000000

$R_1 = L_0 \oplus F(R_0, K_1)$ = 10111010 11010010 00101000 01000101

$L_1 = R_0$ = 00111100 11110000 00111100 00001111

Subkey#1 = 01111000 00110011 11000011 00100000 11011010 01110000

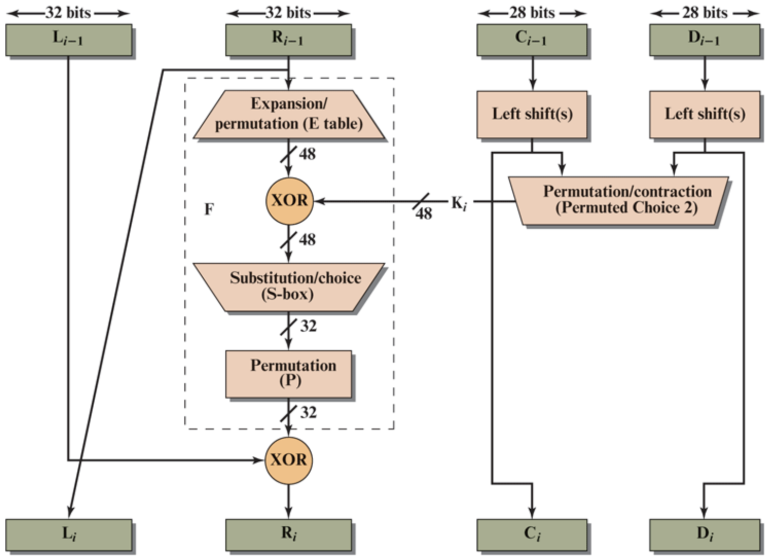
Round 1 Ciphertext: L_1 R_1

00111100 11110000 00111100 00001111

10111010 11010010 01001011 11000010

Round 1 Ciphertext: 3CF03C0F BAD22845

Round 1 Subkey: 7833C320DA70



Avalanche Effect in DES: Change in **Plaintext**

- A small change in either the plaintext or the key should produce a significant change in the ciphertext.
- The second column shows the intermediate 64-bit values at the end of each round for the two plaintexts.
- ***δ****:* *T*he number of bits that differ between the two intermediate values.

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

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

Avalanche Effect in DES: Change in **Key**

Plaintext	02468aceeca86420	02468aceeca86420
Key	0f1571c947d9e859	1f1571c947d9e859
Ciphertext	da02ce3a89ecac3b	ee92b50606b62b0b

Round		δ
	02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	27
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	**da02ce3a89ecac3b** **ee92b50606b62b0b**	**30**

DES Exhaustive Attack

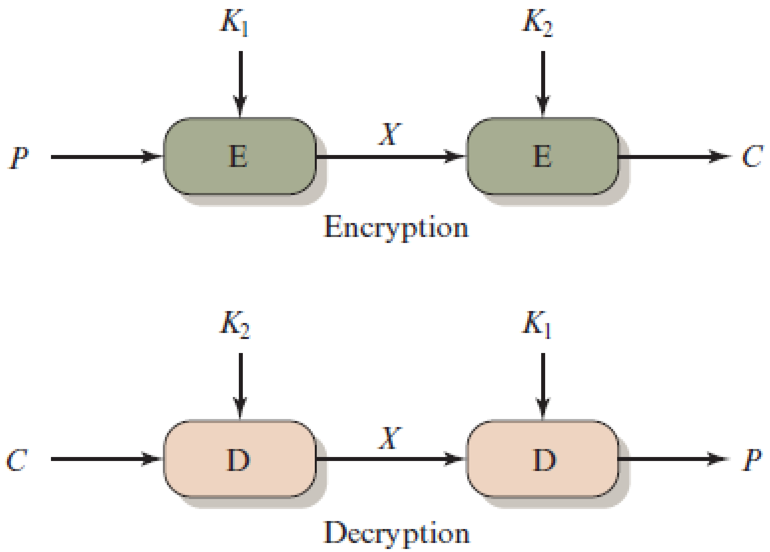
- Exhaustive Key search crack time estimates:
  - $2^{56} \approx 7\text{x}10^{16}$ is not trivial on a single core (2000 years), but it takes about an hour on today's clusters → **Key space must be enlarged!**

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

Multiple Encryption

- Because of its vulnerability to brute-force attack, DES, has been largely replaced by stronger encryption schemes.

- Two approaches have been taken:
  1. Preserve the existing investment in software and equipment, and use multiple encryption with DES and multiple keys.
  2. Design a completely new algorithm that is resistant to both cryptanalytic and brute-force attacks (e.g., AES).



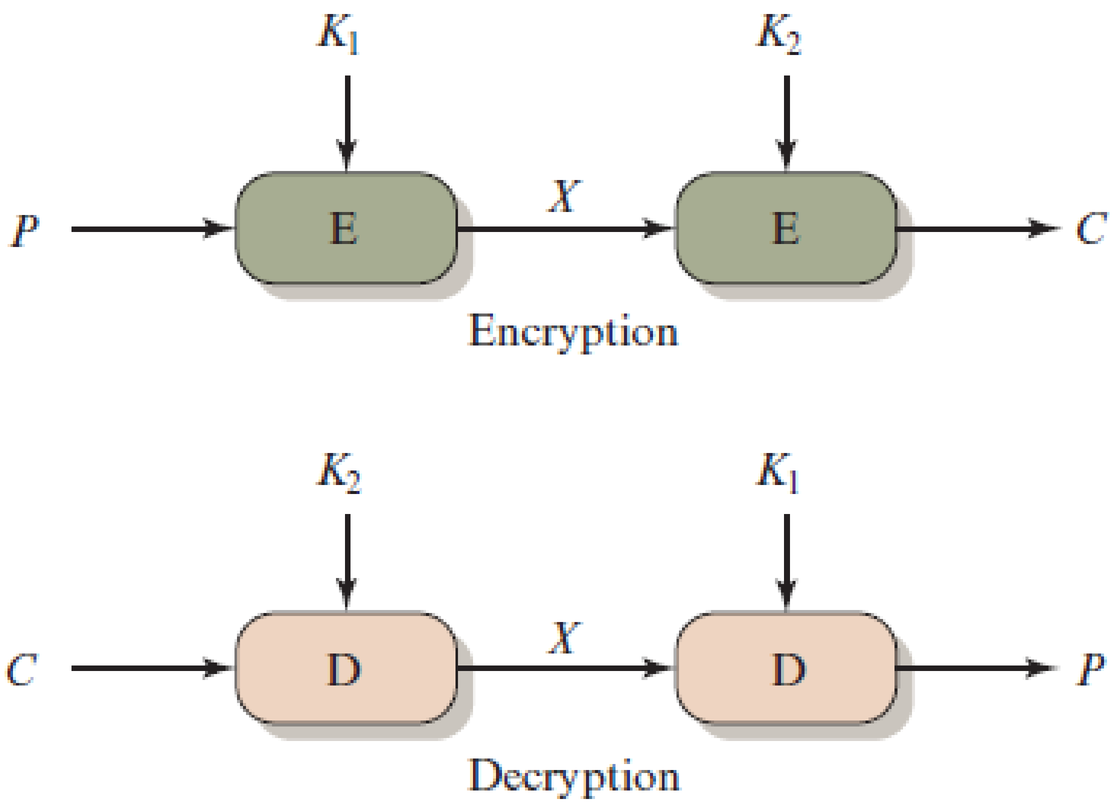
(a) Double encryption

Double DES (2DES)

- The simplest form of multiple encryption has two encryption stages and two keys.

$$C = E(K_2, E(K_1, P))$$
$$P = D(K_1, D(K_2, C))$$

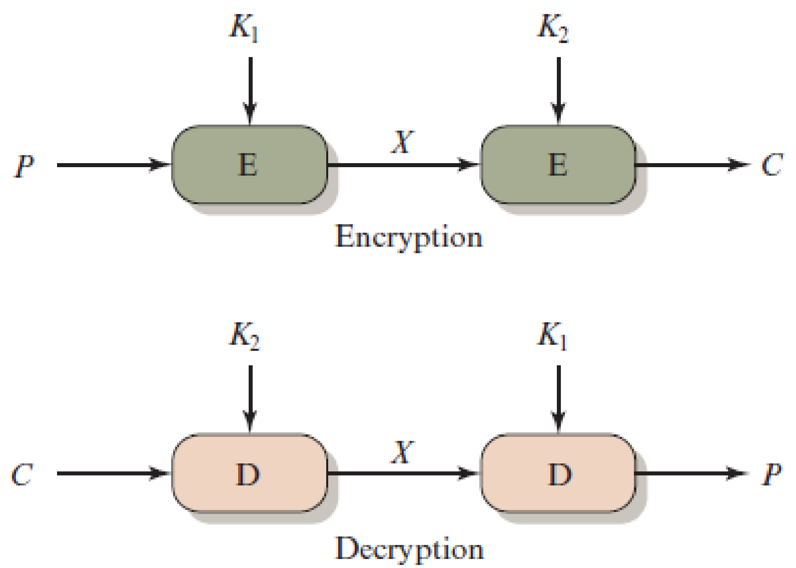
- 2DES uses, in effect, a 112-bit key.
  - Requires 2^{112} key trials to crack.
  - 10^{12} years on a cluster!
  - But, with Meet in the Middle, it reduces to 1DES.



(a) Double encryption

Meet-in-the-middle Attack – 2DES

- An attack used against a double encryption algorithm.
- This is a KPA, i.e., requires a known (**plaintext PT**, **ciphertext CT**) pair.
- Encrypt the known PT with all 2^{56} possible keys
- Decrypt the known CT with all 2^{56} possible keys
- There must be at least one match.
- To avoid false positives, repeat with a second KPA.



(a) Double encryption

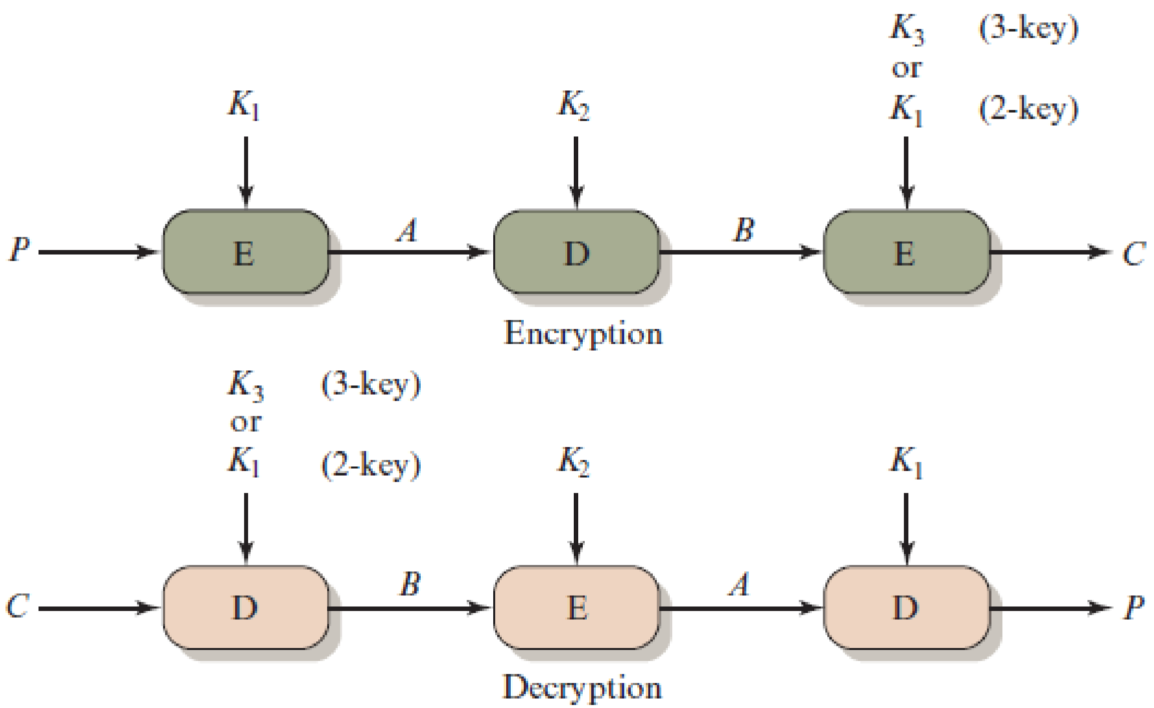
$$C = E(K_2, E(K_1, P))$$

$$X = E(K_1, P) = D(K_2, C)$$

3DES with Two Keys

- A plaintext block is encrypted **three** times.
- Using two keys: Follows an encrypt-decrypt-encrypt (EDE) sequence:

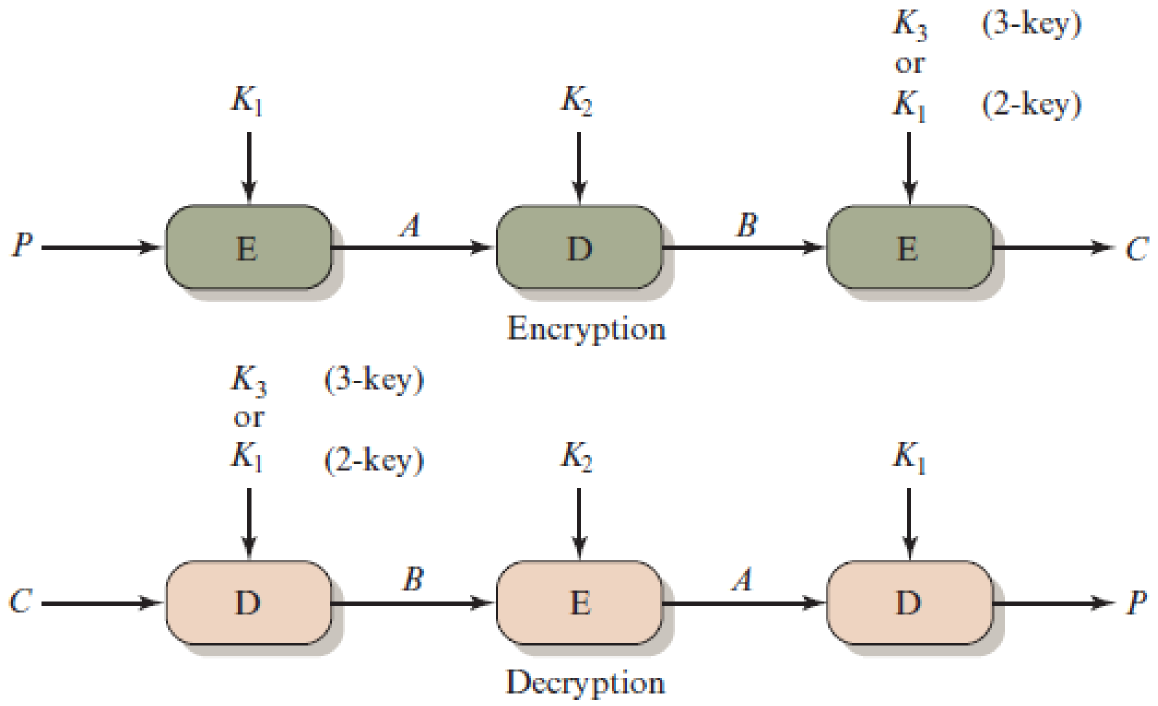
$$C = E(K_1, D(K_2, E(K_1, P)))$$
$$P = D(K_1, E(K_2, D(K_1, C)))$$



(b) Triple encryption

3DES with Three Keys

- Three-key 3DES has an effective key length of 168 bits (56*3) and is defined as:
  - $C = E(K_3, D(K_2, E(K_1, P)))$
- Backward compatibility with DES is provided by putting:
  - $K_3 = K_2$ or $K_1 = K_2$
- Adopted by a number of Internet-based applications including PGP, S/MIME and other network protocols.



(b) Triple encryption

Cryptanalytic Attacks on 3DES

- 3DES is only considered secure if three separate keys are used.

- Meet in the middle reduces the effective key length of 168 bits (56*3) to 112.

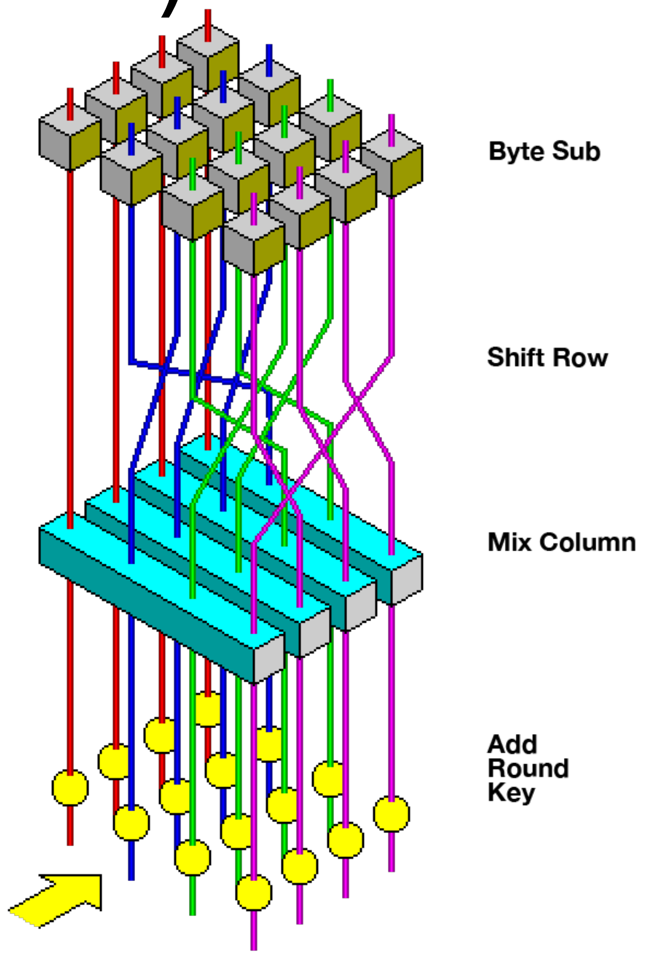
- The cost of a brute-force key search on 3DES is on the order of 2^{112} and the cost of differential cryptanalysis suffers an exponential growth, compared to DES and 2DES.

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES)

- AES was published by NIST in 2001.
- AES is a symmetric **block cipher** that is intended to replace DES as the approved standard for a wide range of applications.
- AES has become the **most** widely used symmetric cipher.



Visualization of the AES round function

AES Encryption and Decryption

AES

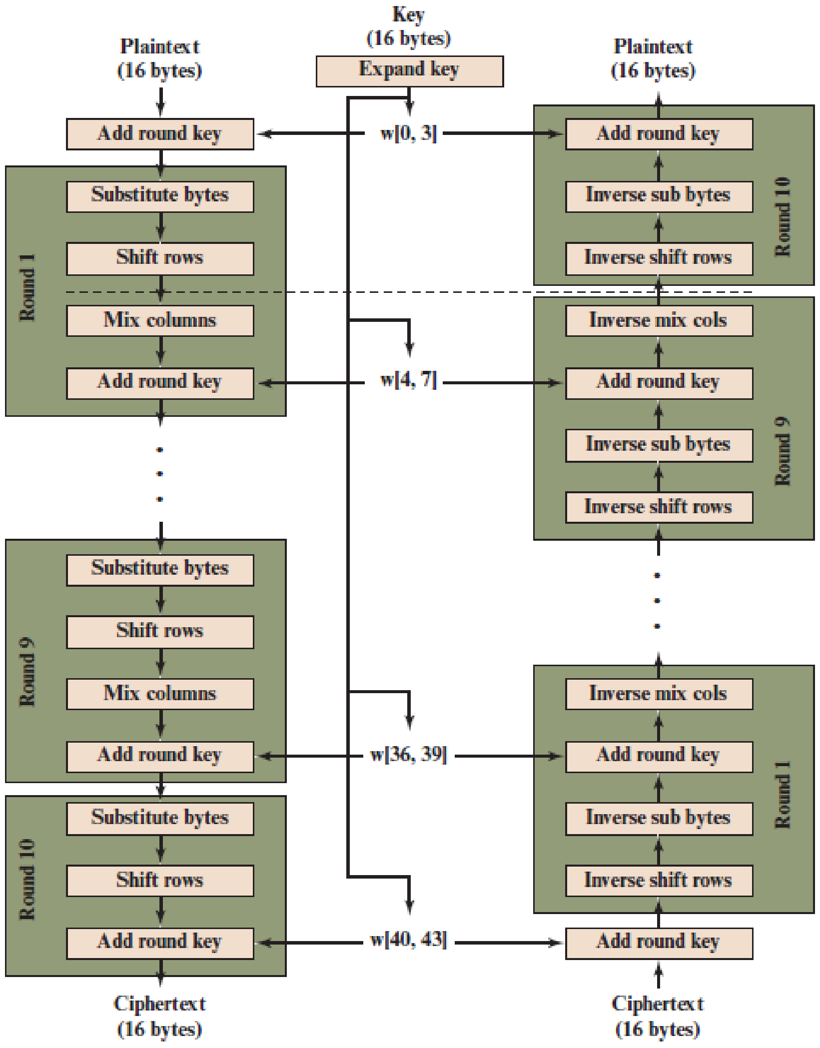
- **AES** is a variant of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen, and Joan Daemen who submitted a proposal to NIST during the AES selection process.

- Rijndael is a family of ciphers with different key and block sizes.

- For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but **three different key** lengths: 128, 192 and 256 bits.

NIST
National Institute of Standards and Technology

AES Encryption and Decryption

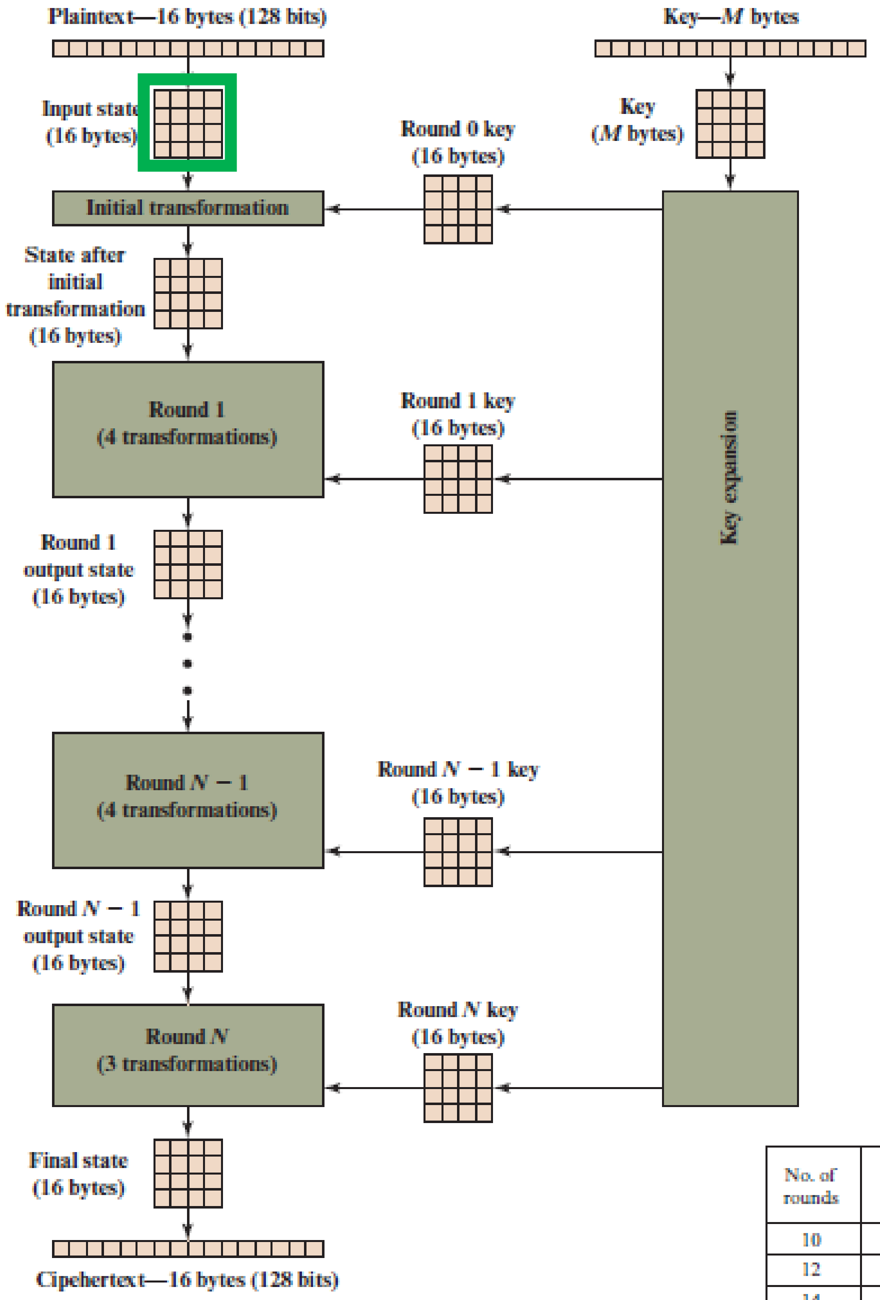


(a) Encryption
(b) Decryption

AES General Structure

- The plaintext **input** to the encryption and decryption algorithms is a **single 128-bit** block.
  - The key length can be 16, 24, or 32 bytes (AES-128, AES-192, or AES-256 bits).

- The **input** is depicted as a 4*4 square matrix of bytes (16 bytes).



AES Encryption

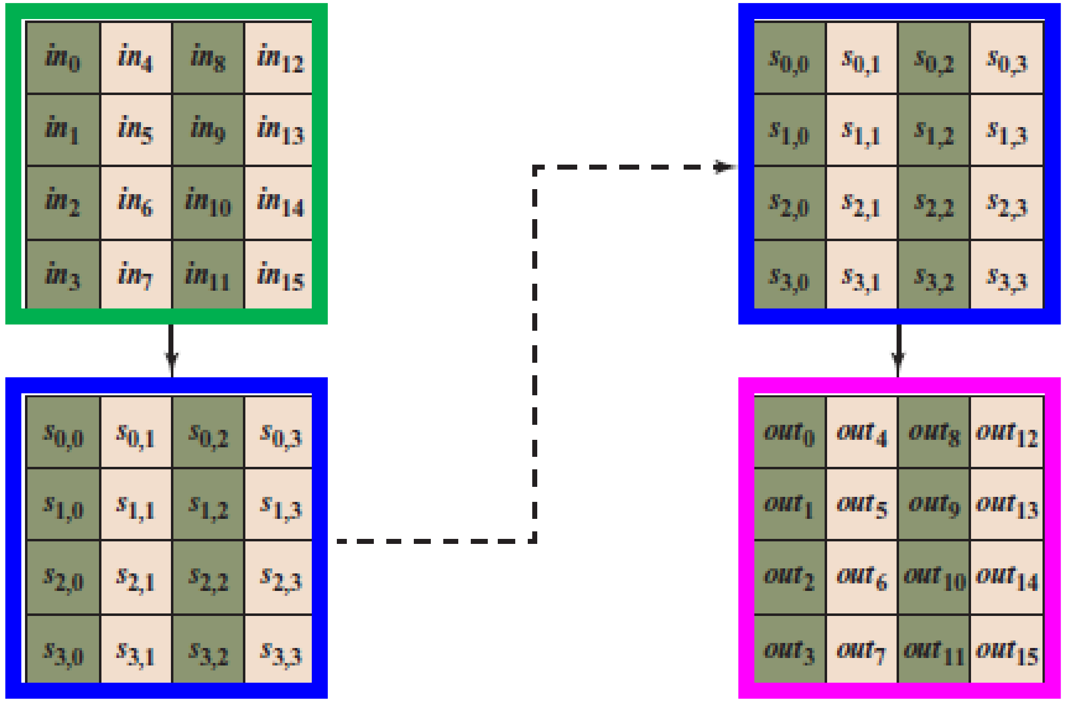
No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Input, State Array, and Output

- **Input** state is copied into the **State** array, which is modified at each stage of encryption or decryption.

- After the final stage, **State** is copied to an **output** matrix.

- **Question**: What is the purpose of the State array?
  - Holds the intermediate results at each stage in the processing.

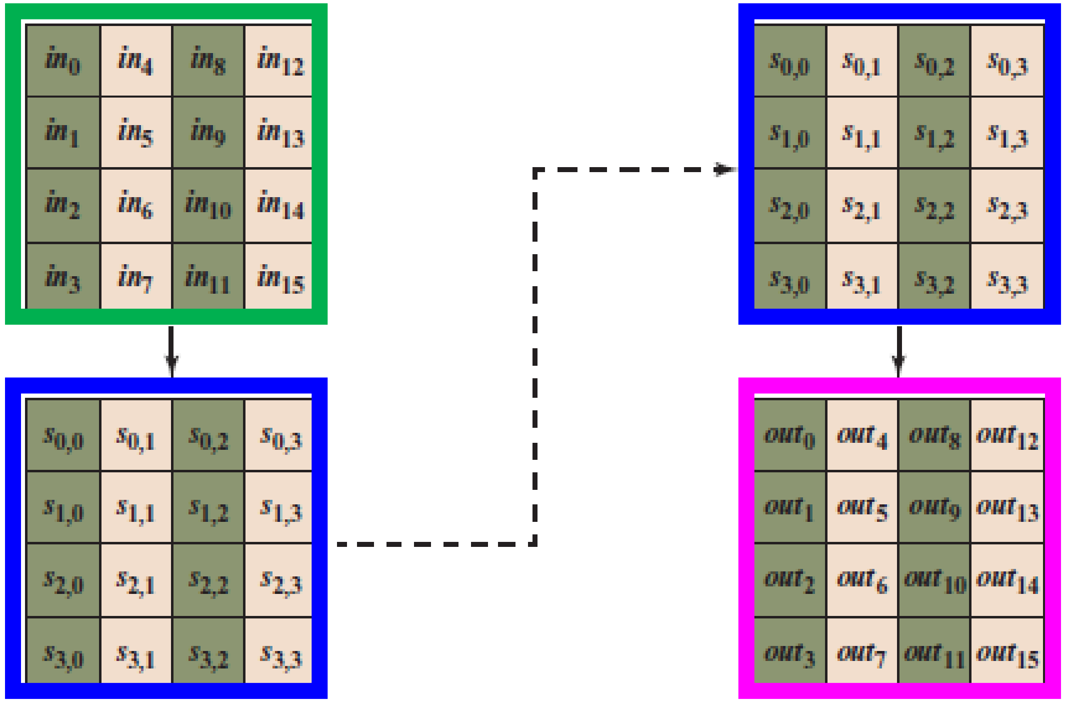


(a) Input, state array, and output

AES Data Structure

AES Input, State Array, and Output

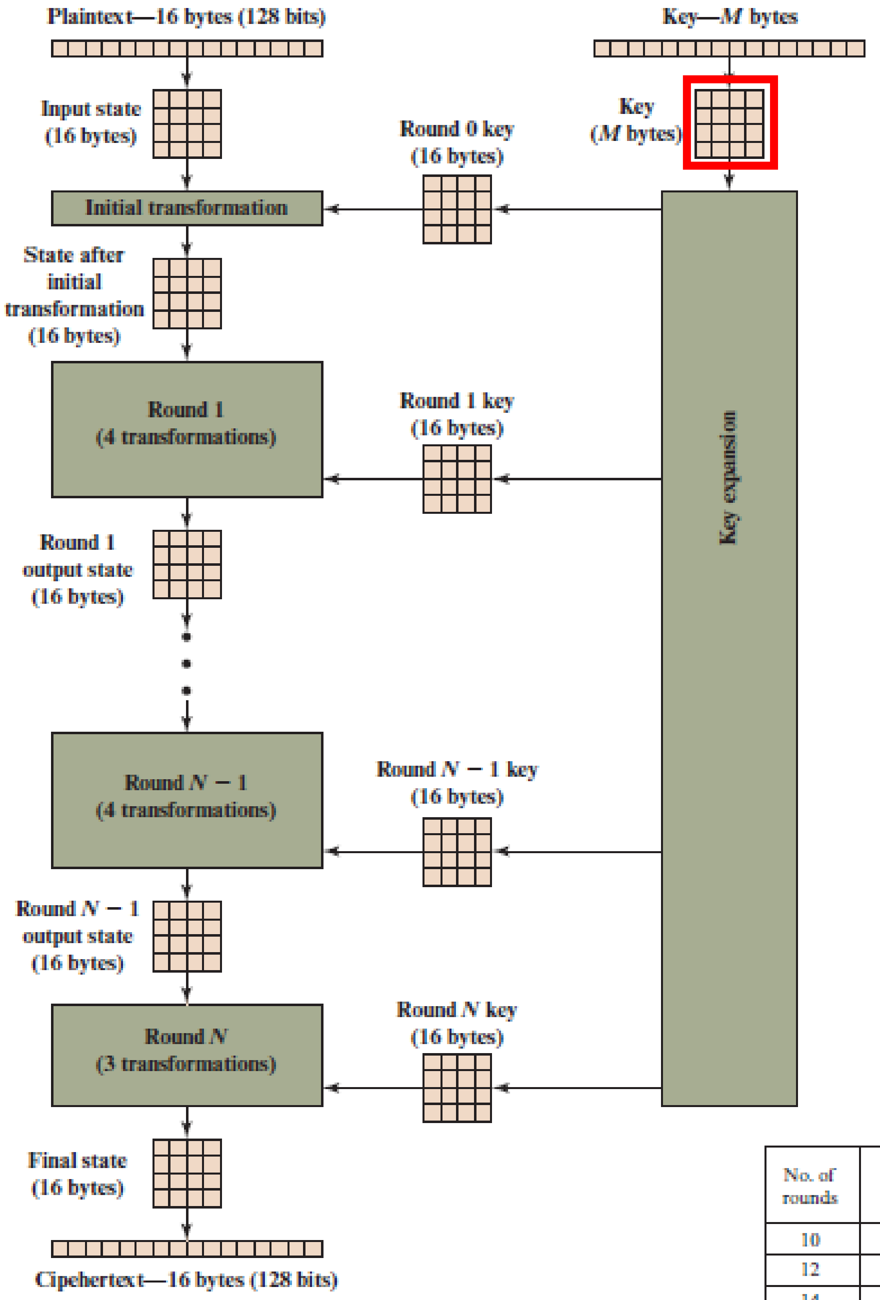
- The ordering of bytes within a matrix is **by column**.
- E.g., the **1st** four bytes of a 128-bit plaintext input to the encryption cipher occupy the **1st column** of the **input** matrix, the **2nd** four bytes occupy the **2nd column**, and so on.



(a) Input, state array, and output

AES Key Schedule

- Similarly, the **key** is depicted as a square matrix of bytes.
- This key is then expanded into an array of **key schedule words**.

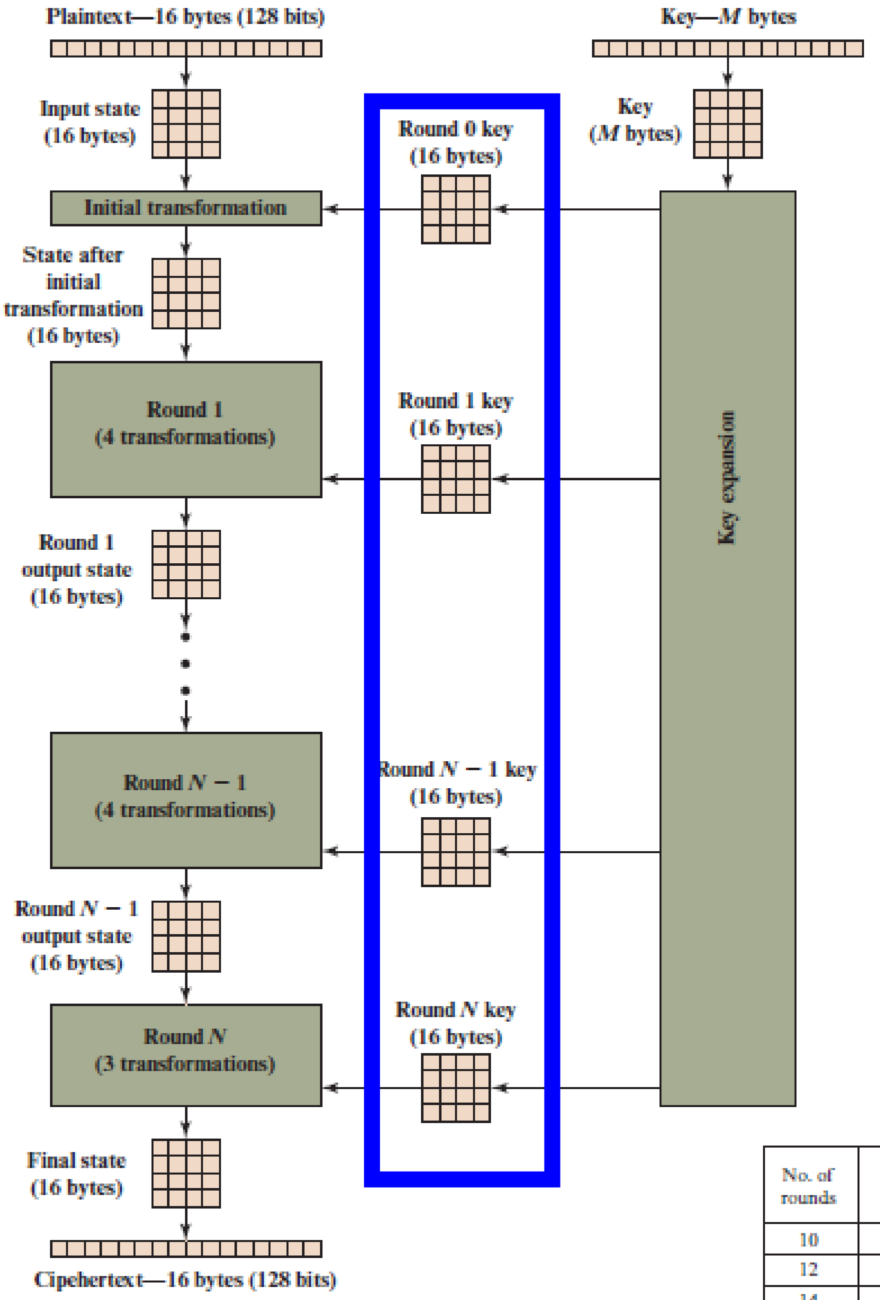


No. of rounds	Key Length (bytes)
10	16
12	24
14	32

AES Parameters

- The cipher consists of ***N*** rounds, where the number of rounds depends on the **key** length:

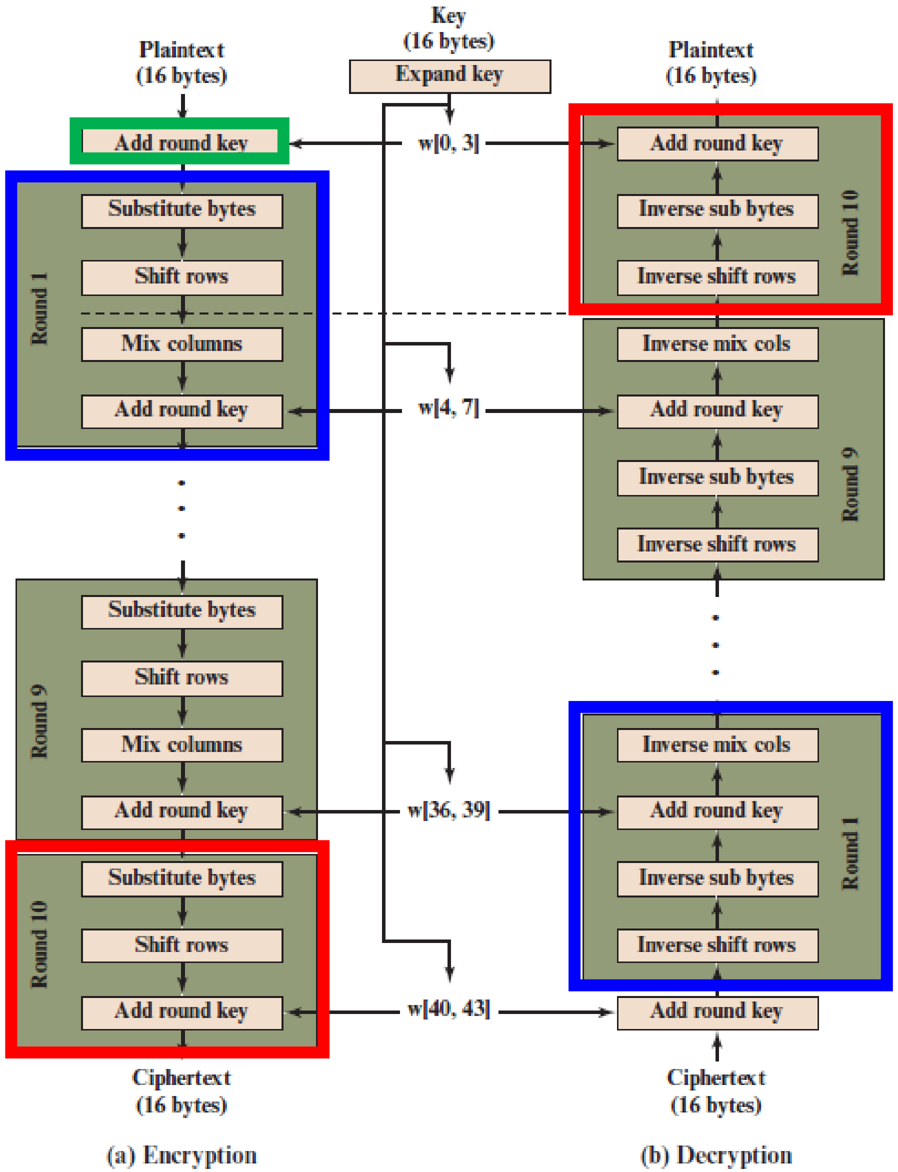
Key Size (bits)	128	192	256
Plaintext Block Size (bits)	128	128	128
N of Rounds	10	12	14
Round Key Size (bits)	128	128	128
Expanded Key Size (words/bytes)	44/176 =11*4	52/208 =13*4	60/240 =15*4



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

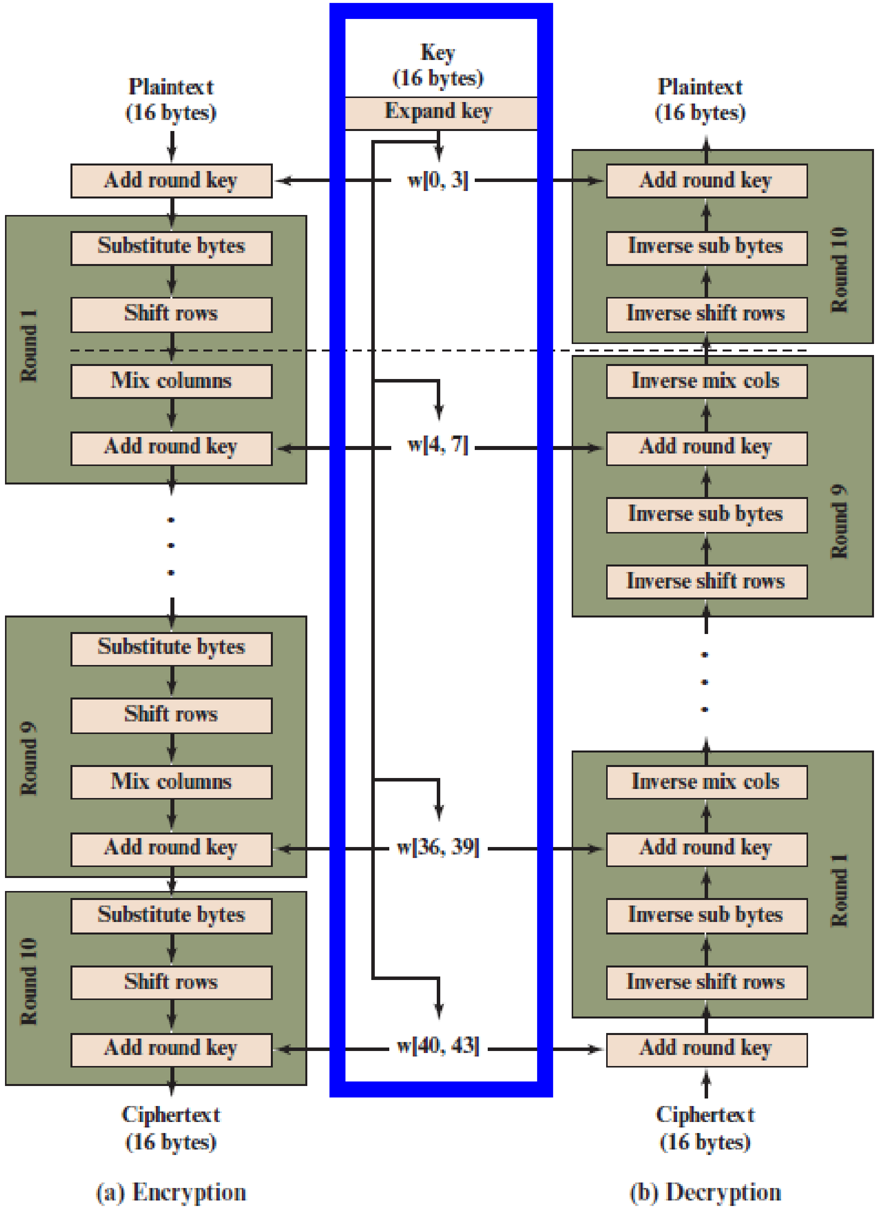
AES Detailed Structure

- The **first** $N - 1$ rounds consist of **4** distinct transformation functions: **SubBytes**, **ShiftRows**, **MixColumns**, and **AddRoundKey**.
- The **final** (N) round contains only **3** transformations.
- There is an **initial** single transformation (**AddRoundKey**) before the first round, which can be considered Round **0**.



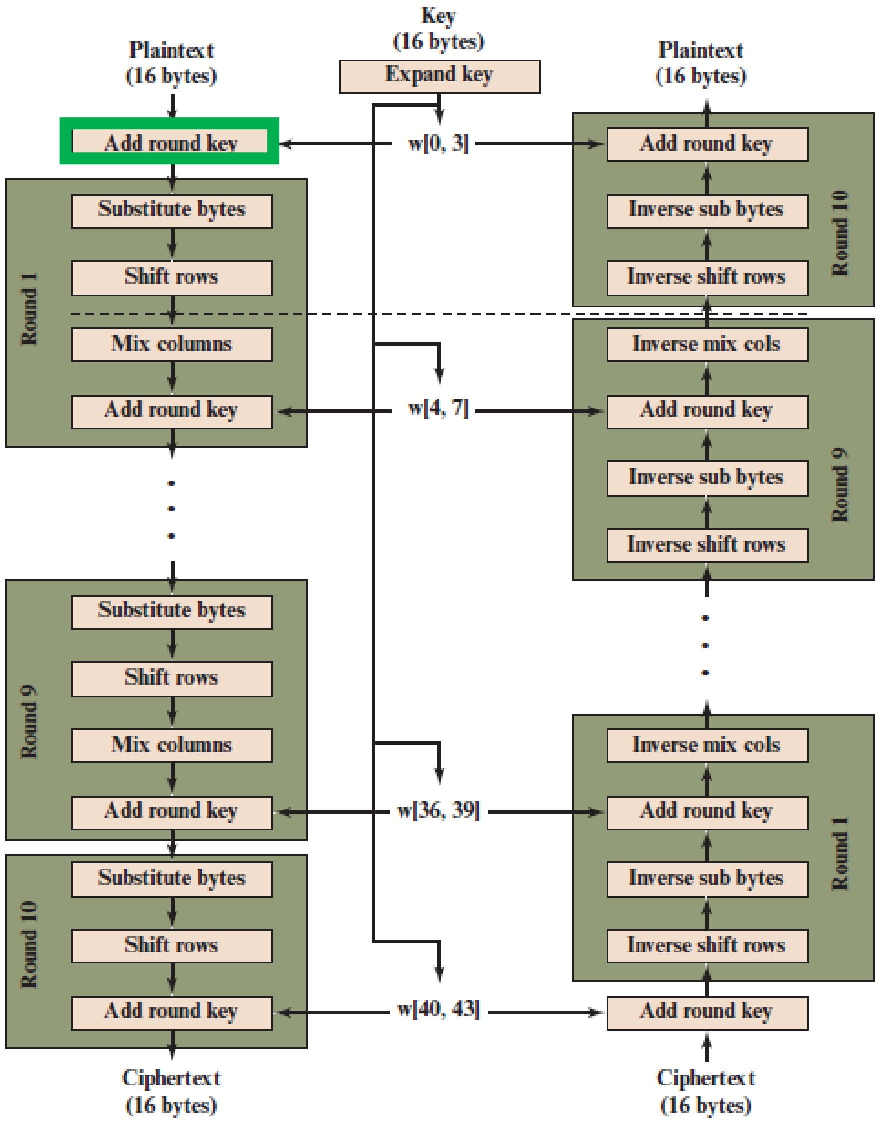
AES Parameters

- The **key** that is provided as input is expanded into an array of **44** words, $w[i]$. Each word is 4 bytes (32 bits).
- **4** distinct words (128 bits) serve as a **round key** for each round.
  - 10 rounds * 4 words per key = 40
  - **1 initial round** + 40=44



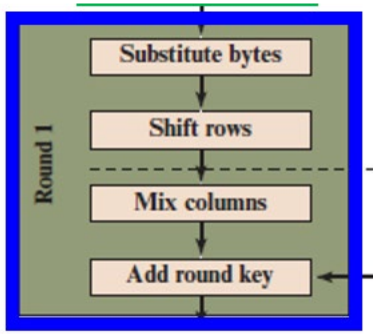
AES Encryption

- The structure is quite simple.
- For both encryption and decryption, assume $N = 10$:
  - The cipher begins with an **AddRoundKey** stage.
  - Followed by **nine** rounds that each includes all **4** stages.
  - Followed by a **tenth** round of **3** stages.

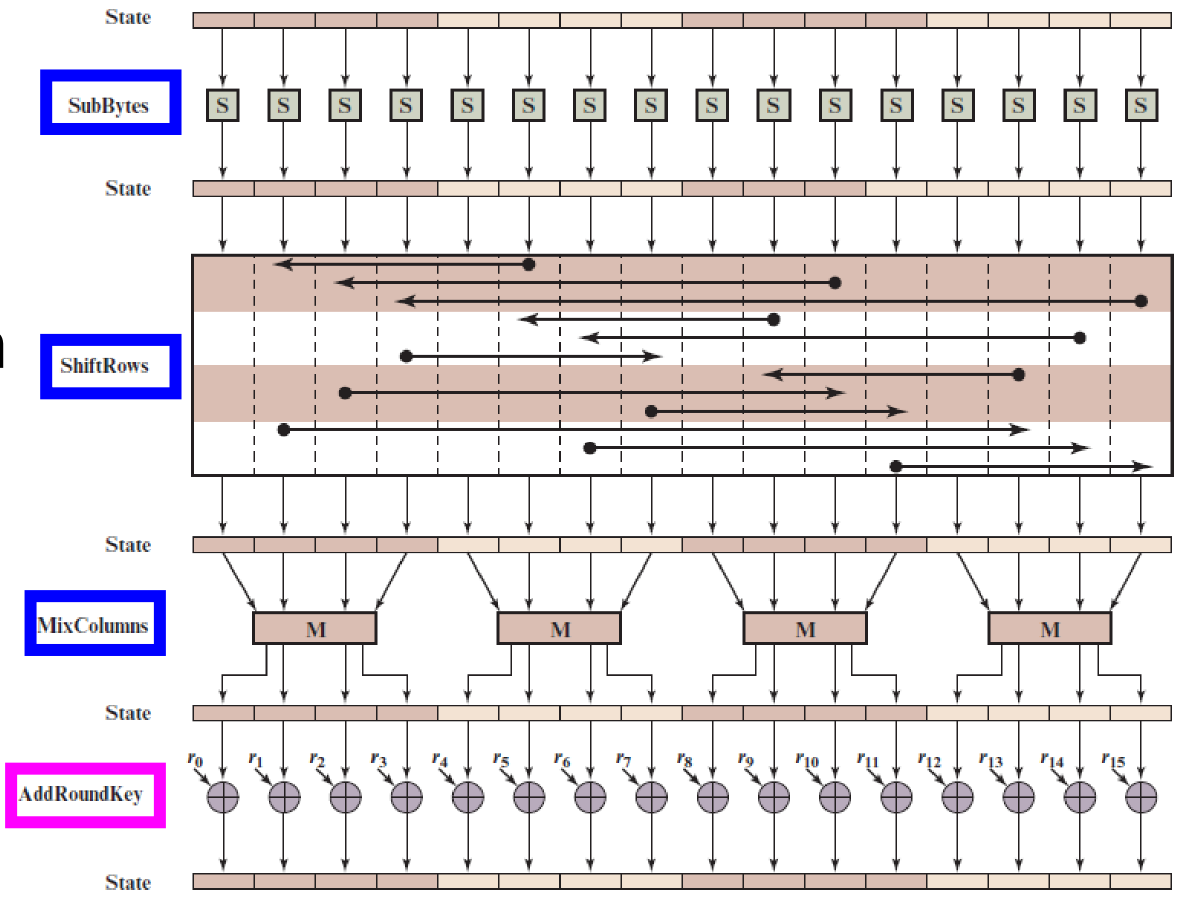


(a) Encryption
(b) Decryption

AES Encryption Round

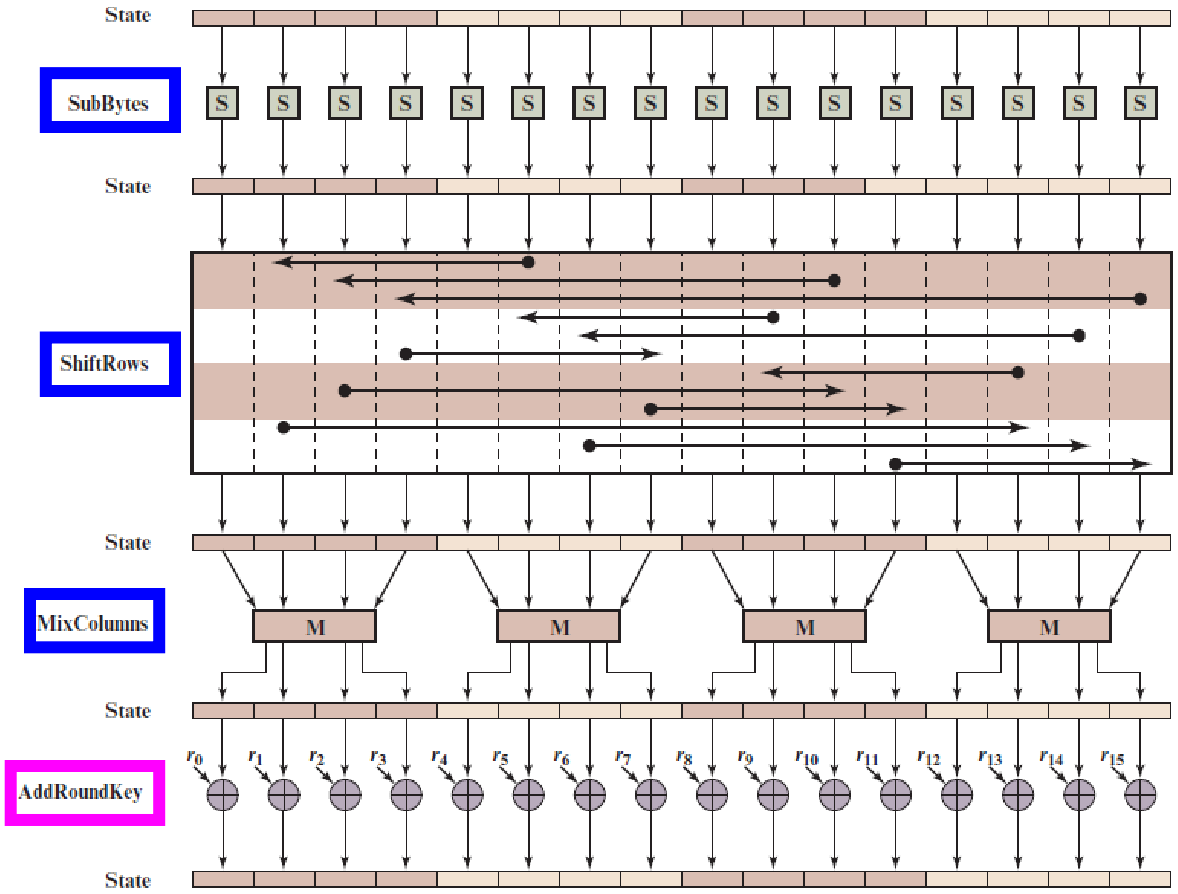


- **Substitute bytes** – uses an S-box to perform a byte-by-byte substitution of the block
- **ShiftRows** – a simple permutation
- **MixColumns** – a substitution
- **AddRoundKey** – a simple bitwise XOR of the current block with a portion of the expanded key

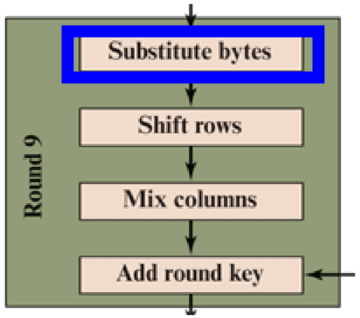


AES Encryption Round

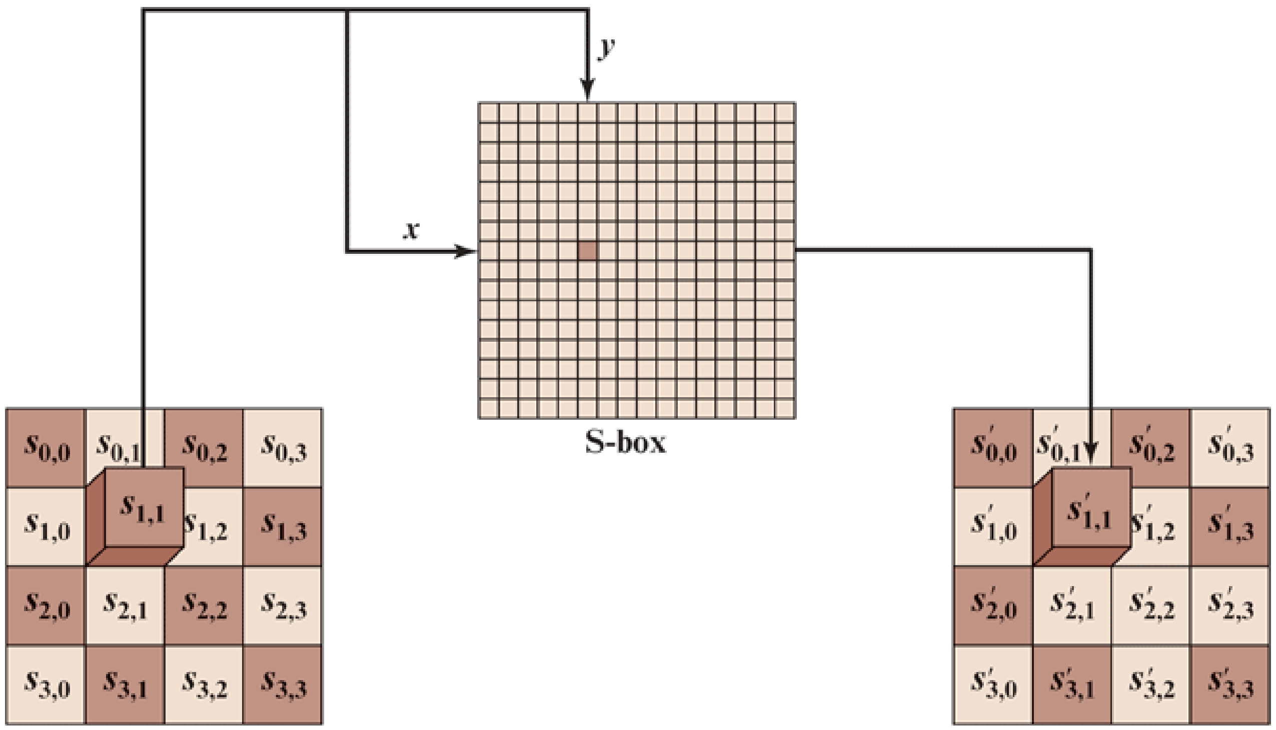
- Only the **AddRoundKey** stage makes use of the key.
- The cipher begins and ends with an **AddRoundKey** stage.
- Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.



AES Substitute Byte Transformation



- **SubBytes:** A simple table lookup.



(a) Substitute byte transformation

AES S-Box

- 16*16 matrix of byte values, that contains a permutation of all possible 256 (16*16) 8-bit values.

- The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value.

x \ y	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

(a) S-box

Example: **Input:** 95 hexadecimal, **Output:** 2A

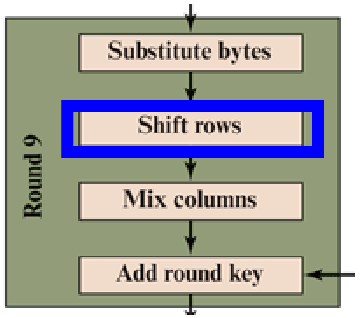
AES Inverse S-Box

- The inverse substitute byte transformation , called **InvSubBytes**, makes use of the inverse S-box

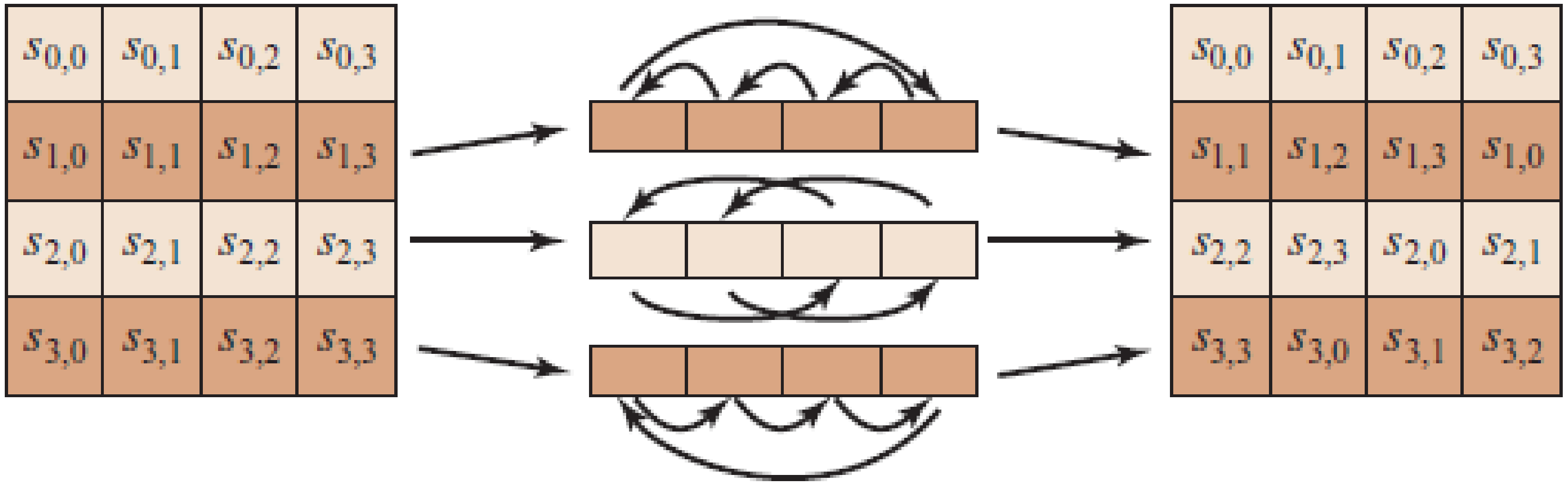
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65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

(b) Inverse S-box

AES ShiftRows



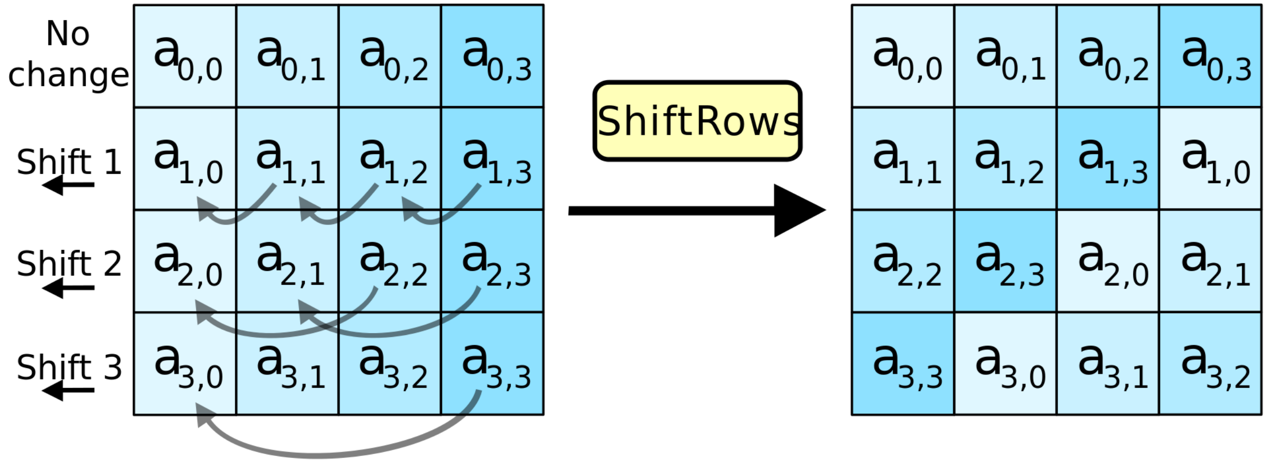
- **ShiftRows:** The **first** row of State is not altered.
  - **Second** row: a **1-byte** circular **left** shift is performed.
  - **Third** row, a **2-byte** circular left shift is performed.
  - **Fourth** row, a **3-byte** circular left shift is performed.



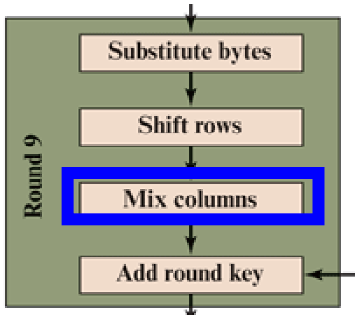
(a) Shift row transformation

AES ShiftRows Example

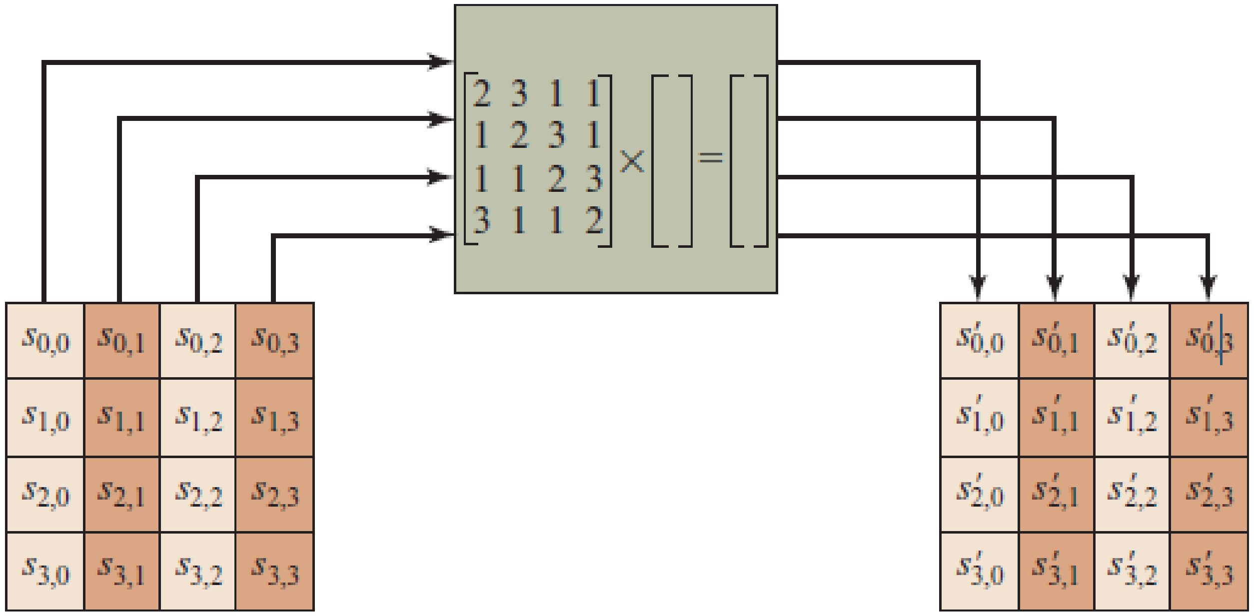
87	F2	4D	97		87	F2	4D	97
EC	6E	4C	90		6E	4C	90	EC
4A	C3	46	E7	→	46	E7	4A	C3
8C	D8	95	A6		A6	8C	D8	95



AES MixColumns



- **MixColumns** operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column.



(b) Mix column transformation

AES MixColumns Example

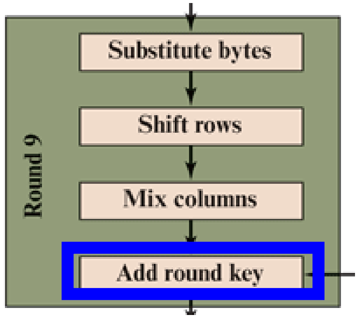
$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \bigoplus (3 \cdot s_{1,j}) \bigoplus s_{2,j} \bigoplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \bigoplus (2 \cdot s_{1,j}) \bigoplus (3 \cdot s_{2,j}) \bigoplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \bigoplus s_{1,j} \bigoplus (2 \cdot s_{2,j}) \bigoplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \bigoplus s_{1,j} \bigoplus s_{2,j} \bigoplus (2 \cdot s_{3,j}) \end{aligned}$$

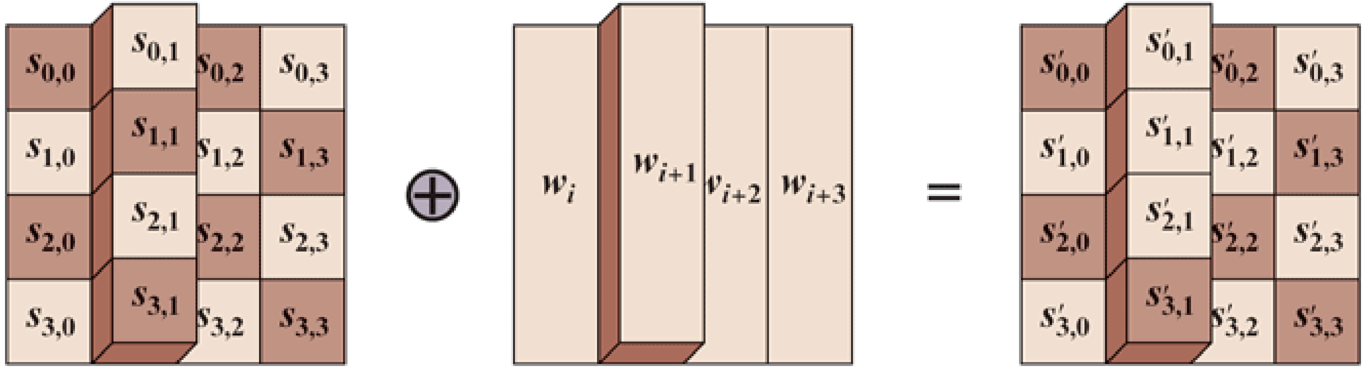
87	F2	4D	97		47	40	A3	4C
6E	4C	90	EC		37	D4	70	9F
46	E7	4A	C3	→	94	E4	3A	42
A6	8C	D8	95		ED	A5	A6	BC

- Note: You are not expected to perform the matrix multiplication shown in the above formulas.

AES AddRoundKey



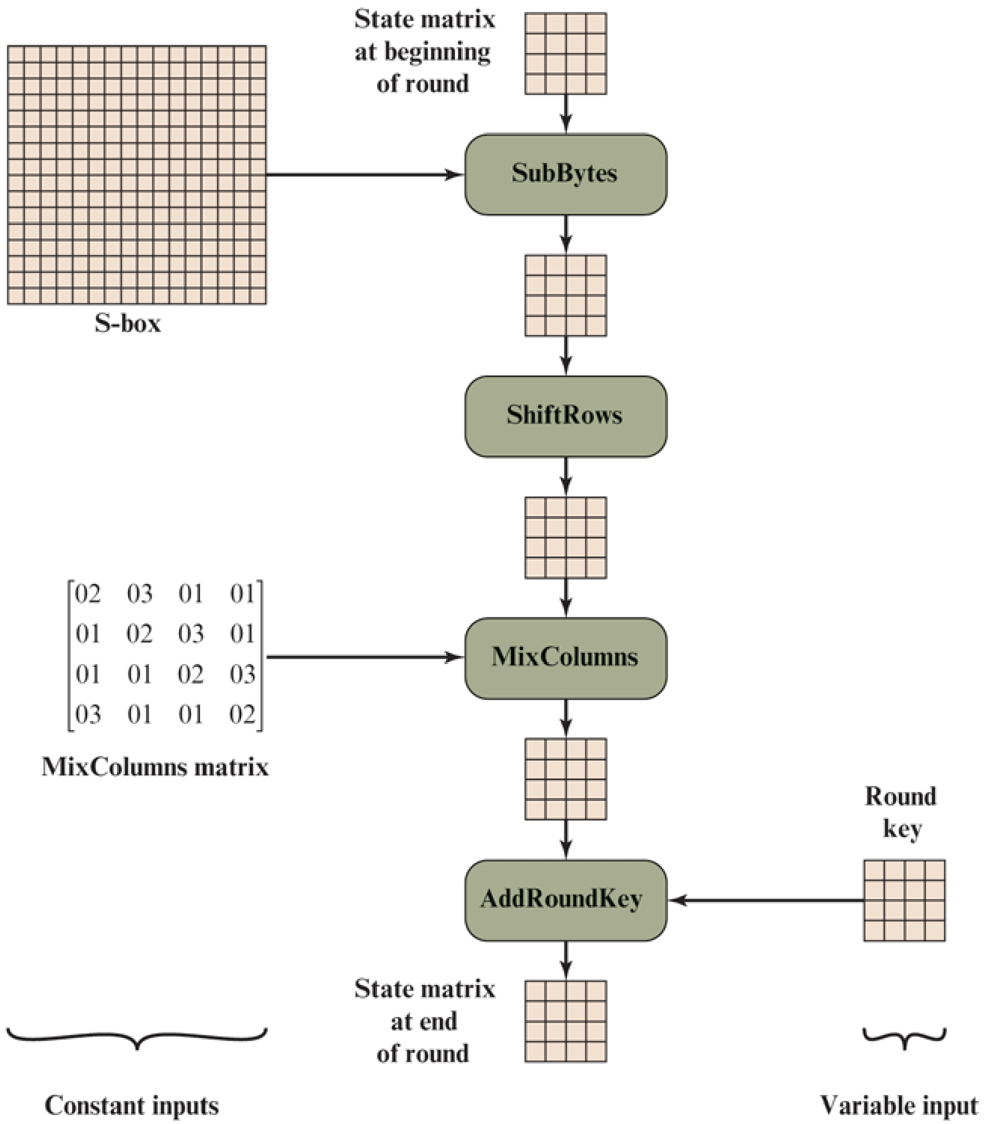
- AddRoundKey: 128 bits of State are bitwise XORed with the 128 bits of the round key.



(b) Add round key transformation

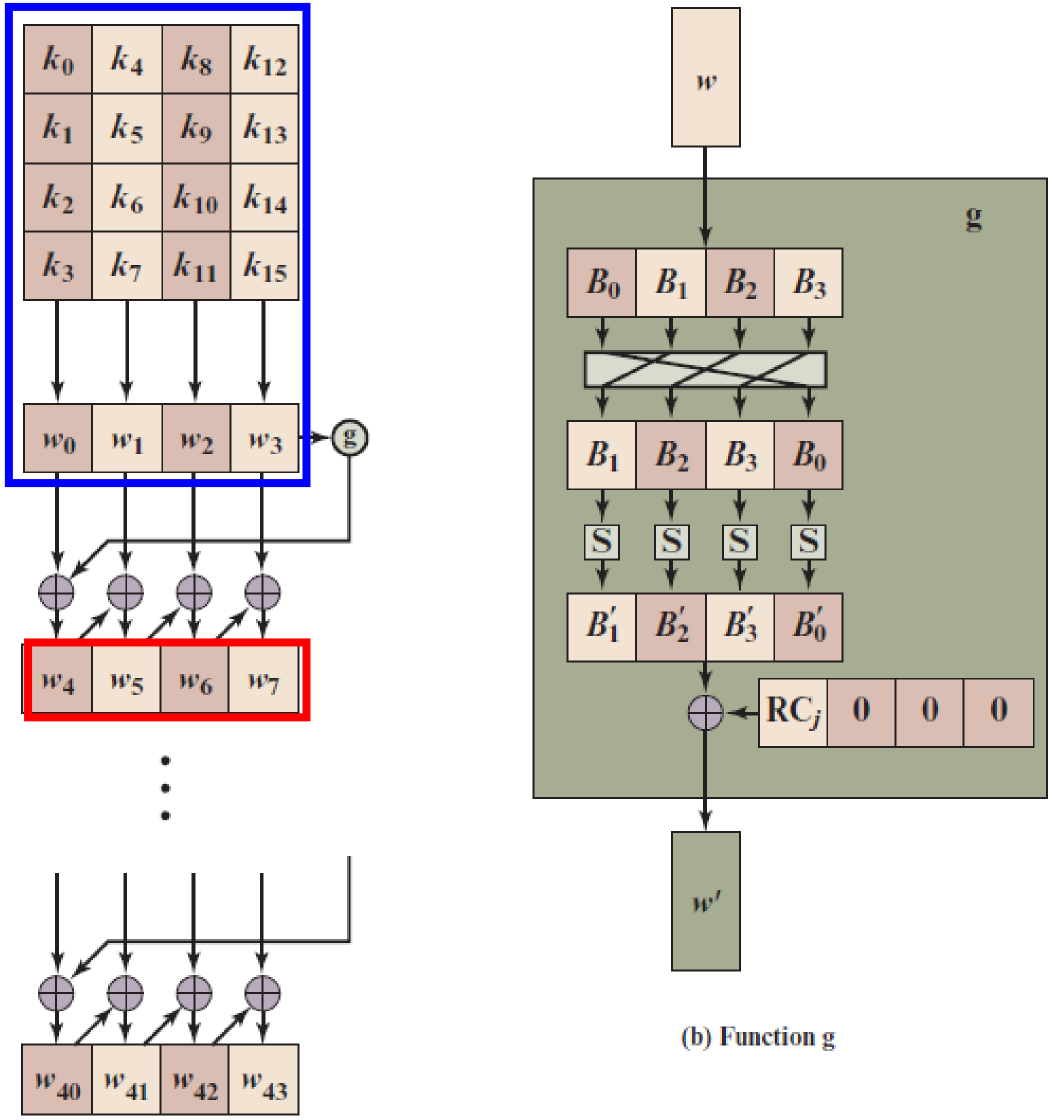
47	$\boxed{40}$	A3	4C		AC	$\boxed{19}$	28	57		EB	$\boxed{59}$	8B	1B
37	D4	70	9F		77	FA	D1	5C		40	2E	A1	C3
94	E4	3A	42	$\oplus$	66	DC	29	00	=	F2	38	13	42
ED	A5	A6	BC		F3	21	41	6A		1E	84	E7	D6

Inputs for a Single AES Round



AES Key Expansion

- The AES **key expansion** algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes).
- Initial Key is copied into the first 4 words of the expanded key.
  - Each word is 4 bytes.
- The remainder of the expanded key is filled in 4 words at a time.



(b) Function g

(a) Overall algorithm

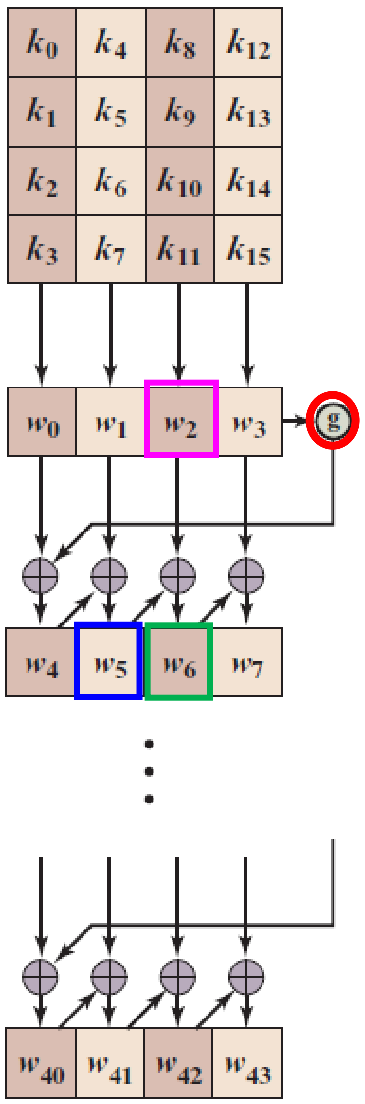
AES Key Expansion

AES Key Expansion

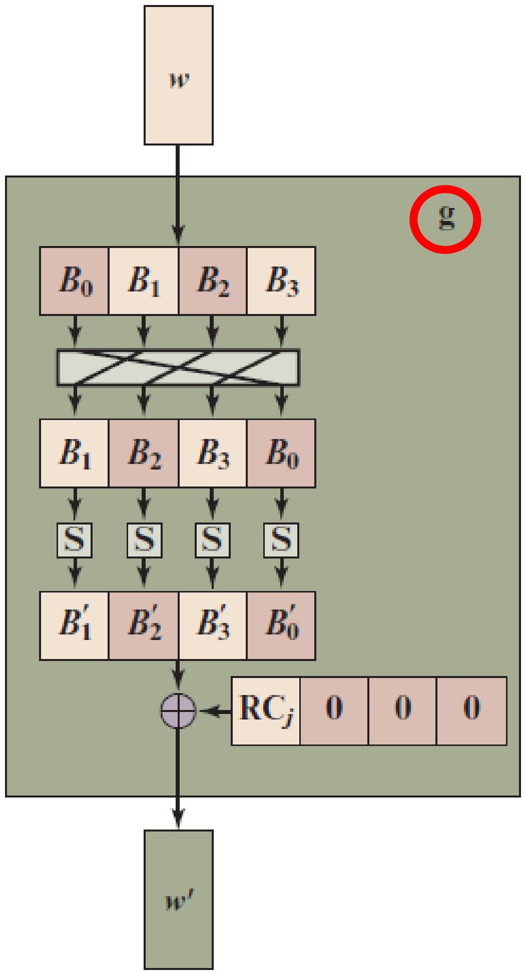
- Each added word $\boldsymbol{w_i}$ depends on the immediately preceding word, $\boldsymbol{w_{i-1}}$, and the word **4** positions back, $\boldsymbol{w_{i-4}}$

$$\boldsymbol{w_i} = \boldsymbol{w_{i-1}} \oplus \boldsymbol{w_{i-4}}$$

- In three (w_5, w_6, w_7) out of four cases a simple XOR is used. What about w_4?



(a) Overall algorithm



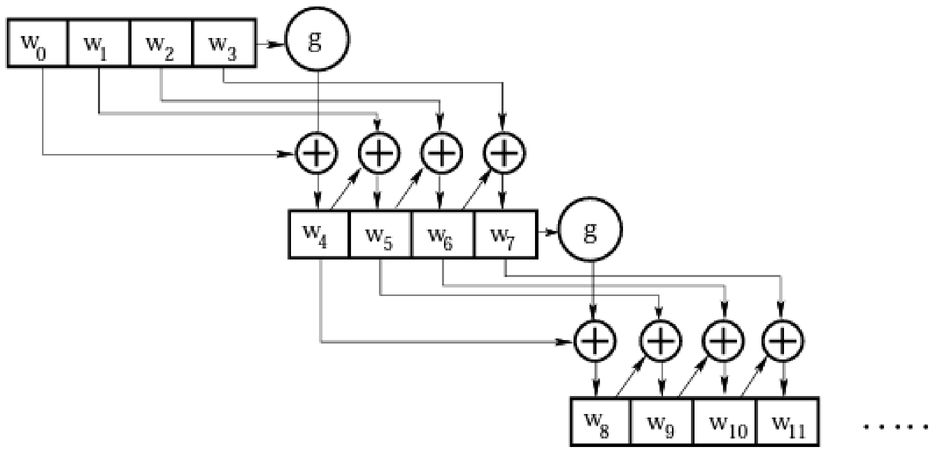
(b) Function g

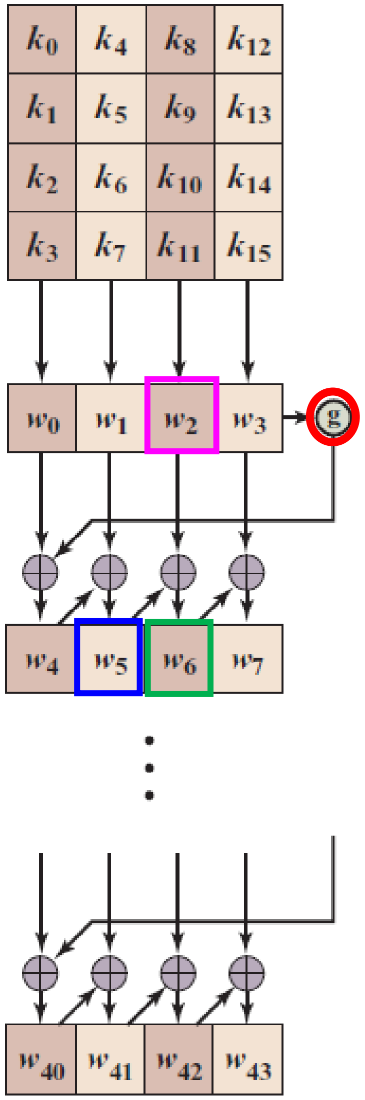
AES Key Expansion

- For a word whose position in the ***w*** array is a **multiple of 4**, a more **complex function *g*** is used:

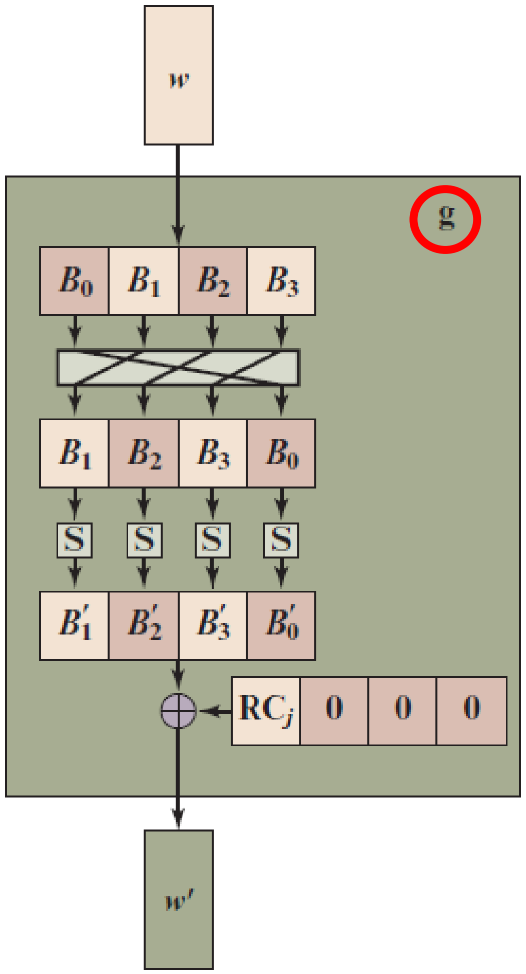
$$w_{i+4} = w_i \oplus \boldsymbol{g}(w_{i+3})$$

$$w_4 = w_0 \oplus \boldsymbol{g}(w_3)$$





(a) Overall algorithm



(b) Function g

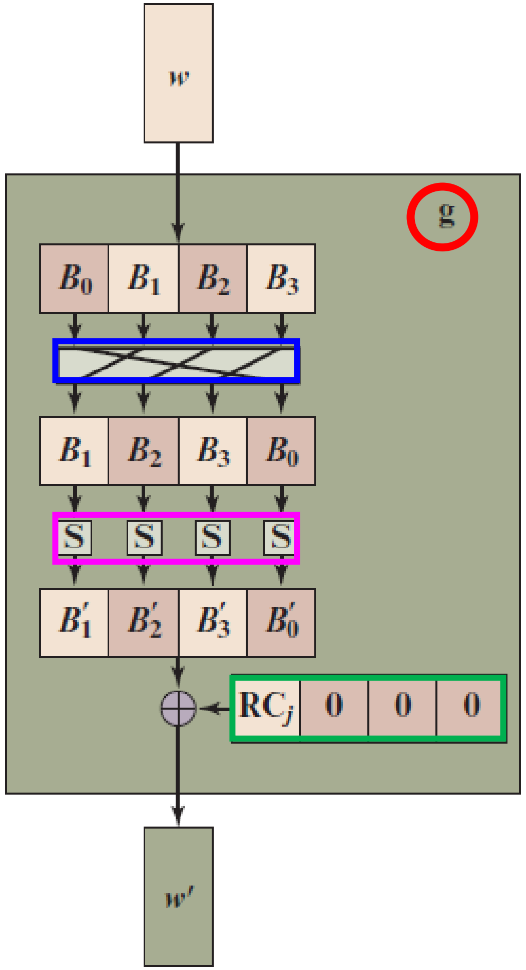
g Function

1. **RotWord:** Performs a one-byte circular left shift on a word.
   - An input word [B_0, B_1, B_2, B_3] is transformed into [B_1, B_2, B_3,B_0]

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].

- **Question:** What is the difference between SubBytes and SubWord?
  - SubBytes operate on State, SubWord operate on an input word.



(b) Function g

Round Constant

- The round constant is different for each round and is defined as

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

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

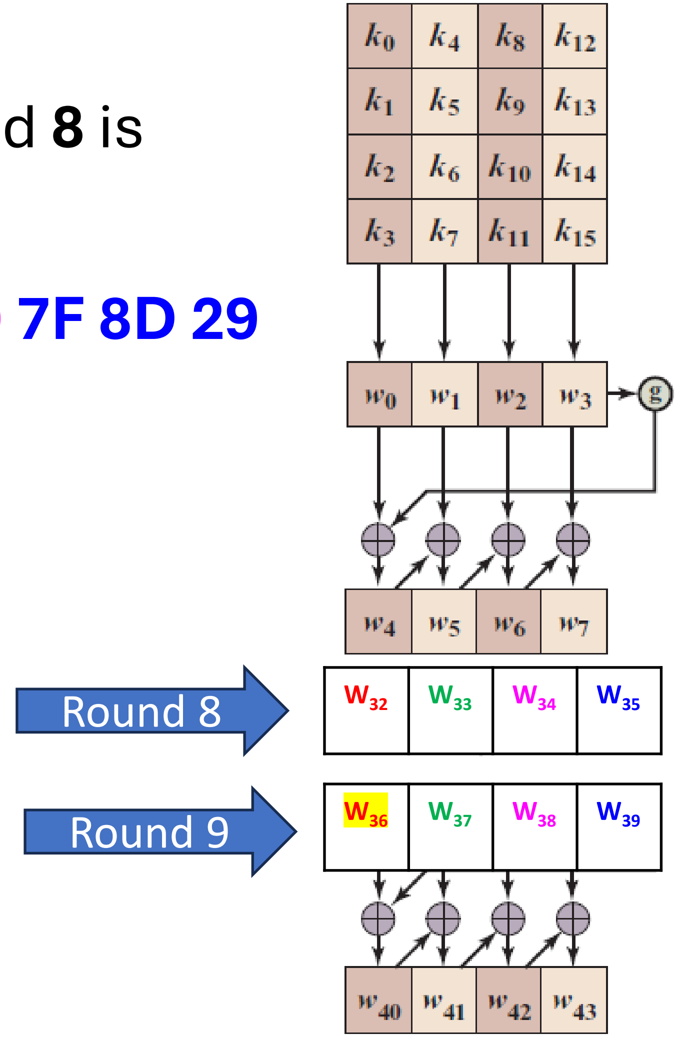
The values of RC[j] in hexadecimal

AES Key Expansion Example

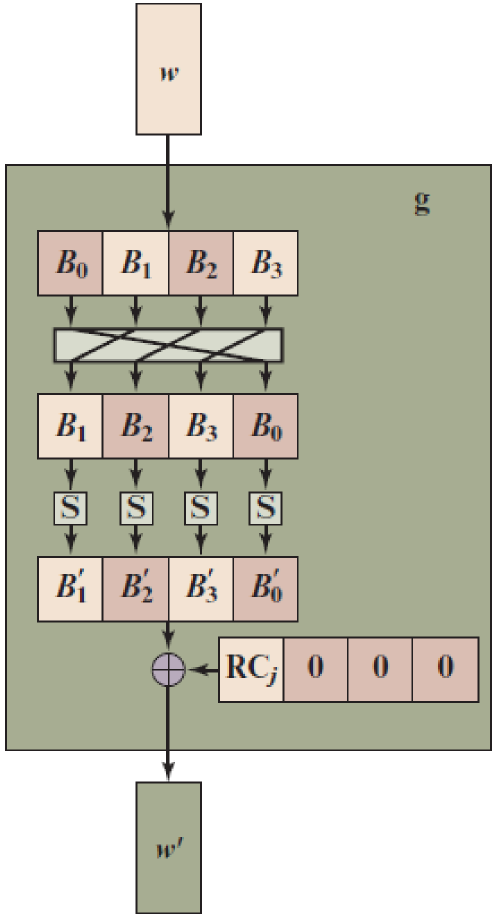
- Suppose that the round key for round **8** is

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

- Calculate Round Key Word W_{36}



(a) Overall algorithm



(b) Function g

Round Key Calculation

W_{32}	W_{33}	W_{34}	W_{35}

- Round 8 key =**EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F**
- Calculate Round Key Word W_{36} (W_{36}= g(W_{35}) ⊕ W_{32})

Description	Value
i (decimal)	**36**
temp = w[i − 1] (w_{35})	**7F8D292F**
RotWord (temp)	8D292F7F
SubWord (RotWord (temp))	5DA515D2
Rcon (9)	1B000000
SubWord (RotWord (temp)) ⊕ Rcon (9)	46A515D2
w[i − 4]	EAD27321
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	AC7766F3

Round Key Calculation

- Round 8 key =**EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F**
- Calculate Round Key Word W_{36} ($W_{36} = g(W_{35}) \oplus W_{32}$)

Description	Value
i (decimal)	36
temp = w[i − 1]	**7F8D292F**
RotWord (temp)	**8D292F7F**
SubWord (RotWord (temp))	5DA515D2
Rcon (9)	1B000000
SubWord (RotWord (temp)) ⊕ Rcon (9)	46A515D2
w[i − 4]	EAD27321
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	AC7766F3

Round Key Calculation

- Round 8 key =**EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F**
- Calculate Round Key Word W_{36}

x \ y	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

(a) S-box

Description	Value
i (decimal)	36
temp = w[i − 1]	**7F8D292F**
RotWord (temp)	**8D292F7F**
SubWord (RotWord (temp))	5DA515D2
Rcon (9)	1B000000
SubWord (RotWord (temp)) ⊕ Rcon (9)	46A515D2
w[i − 4]	EAD27321
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	AC7766F3

Round Key Calculation

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Description	Value
i (decimal)	36
temp = w[i − 1]	**7F8D292F**
RotWord (temp)	**8D292F7F**
SubWord (RotWord (temp))	**5DA515D2**
Rcon (9)	**1B000000**
SubWord (RotWord (temp)) ⊕ Rcon (9)	46A515D2
w[i − 4]	EAD27321
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	AC7766F3

Round Key Calculation

- Round 8 key =**EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F**
- Calculate Round Key Word W_{36}

Description	Value
i (decimal)	**36**
temp = w[i − 1]	**7F8D292F**
RotWord (temp)	**8D292F7F**
SubWord (RotWord (temp))	**5DA515D2**
Rcon (9)	**1B000000**
SubWord (RotWord (temp)) ⊕ Rcon (9)	**46A515D2**
w[i − 4]	EAD27321
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	AC7766F3

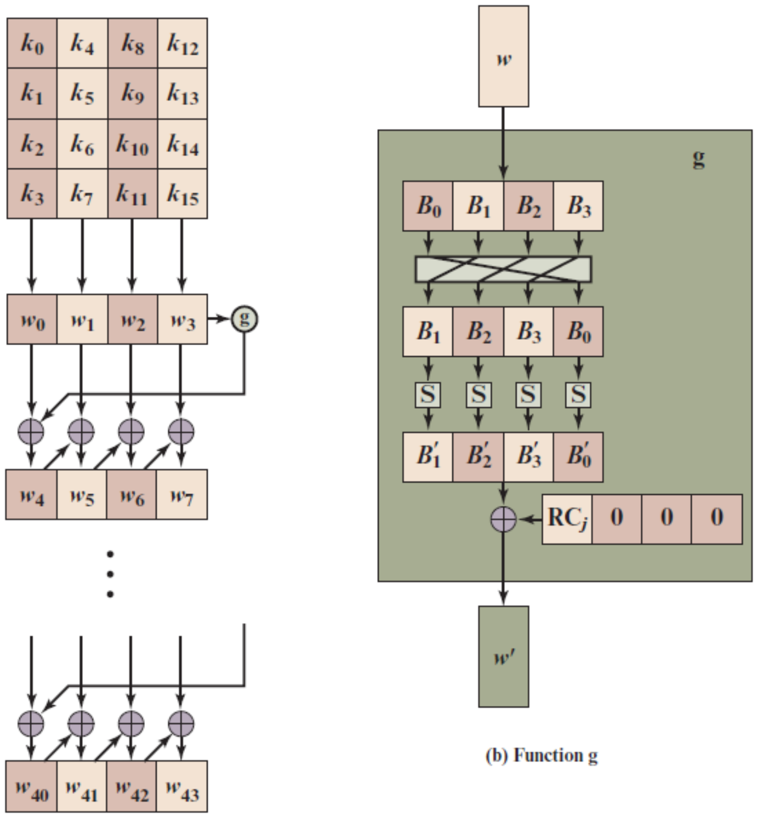
Round Key Calculation

- Round 8 key = $EA\ D2\ 73\ 21\ B5\ 8D\ BA\ D2\ 31\ 2B\ F5\ 60\ 7F\ 8D\ 29\ 2F$
- Calculate Round Key Word W_{36} (W_{36}= g(W_{35}) xor W_{32})

Description	Value
i (decimal)	**36**
temp = w[i − 1]	**7F8D292F**
RotWord (temp)	**8D292F7F**
SubWord (RotWord (temp))	**5DA515D2**
Rcon (9)	**1B000000**
SubWord (RotWord (temp)) ⊕ Rcon (9)	**46A515D2**
w[i − 4]	**EAD27321**
w[i] = w[i − 4] ⊕ SubWord (RotWord (temp)) ⊕ Rcon (9)	**AC7766F3**

Round Key Calculation - Exercise

- Round 8 key $= EA\ D2\ 73\ 21\ B5\ 8D\ BA\ D2\ 31\ 2B\ F5\ 60\ 7F\ 8D\ 29\ 2F$
- Round Key Word W_{36} =AC7766F3
- Calculate W_{37}?
- $W_{37} = W_{36} \oplus W_{33}$



(a) Overall algorithm

(b) Function g

Avalanche Effect in AES: Change in **Plaintext**

- When the 8th bit of the plaintext is changed, after **just one round, 20 bits** of the State vector differ. After two rounds, close to half the bits differ. **Key =0f1571c947d9e8590cb7add6af7f6798**

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

Avalanche Effect in AES: Change in **Key**

- When the same plaintext is used and the two keys differ in the 8th bit. New Key= 0**e**1571c947d9e8590cb7add6af7f6798

Round		δ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63

Round		δ
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

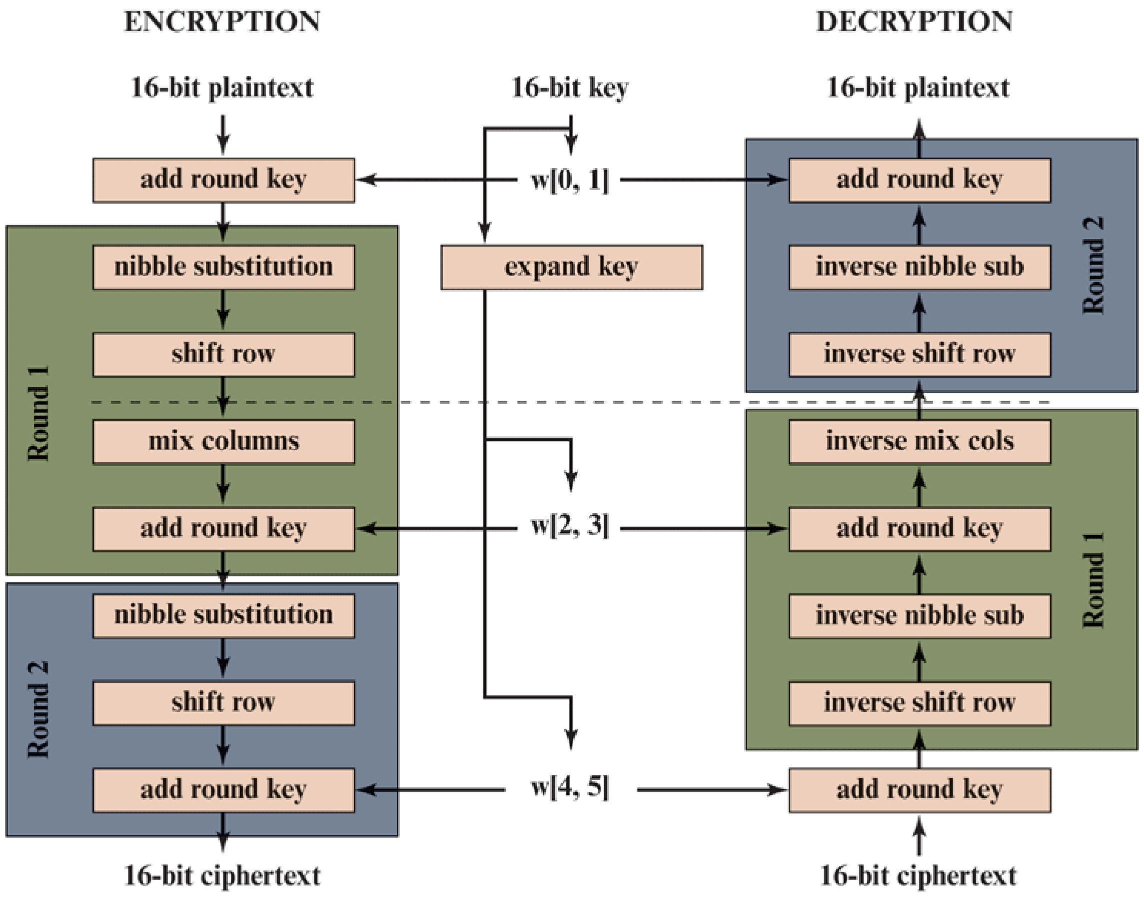
Avalanche Effect in AES

- AES avalanche effect is stronger than that for DES.

- DES required three rounds to reach a point at which approximately half the bits are changed, both for a bit change in the plaintext and a bit change in the key.

S-AES Encryption and Decryption

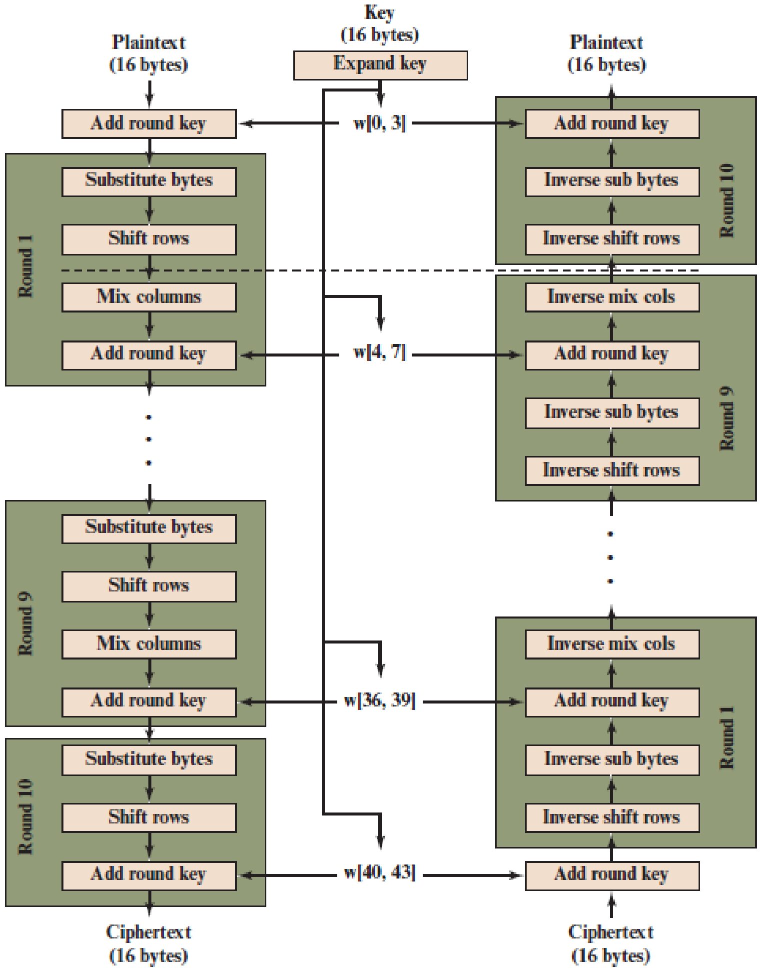
- Simplified AES (S-AES) was developed by Professor Edward Schaefer of Santa Clara University and several of his students.
- Educational but not a secure encryption algorithm.
- It has similar properties and structure to AES with much smaller parameters (16-bit PT and 16-bit key).



AES Structure

- **Question**: Is AES structure a Feistel structure?
  - In the classic Feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped.
  - AES instead processes the entire data block as a single matrix during each round using substitutions and permutation.
  - AES is **not** a Feistel structure.

AES Encryption and Decryption



(a) Encryption

(b) Decryption

Topics

- Stream Cipher
- Block Cipher
- The Feistel Cipher
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)