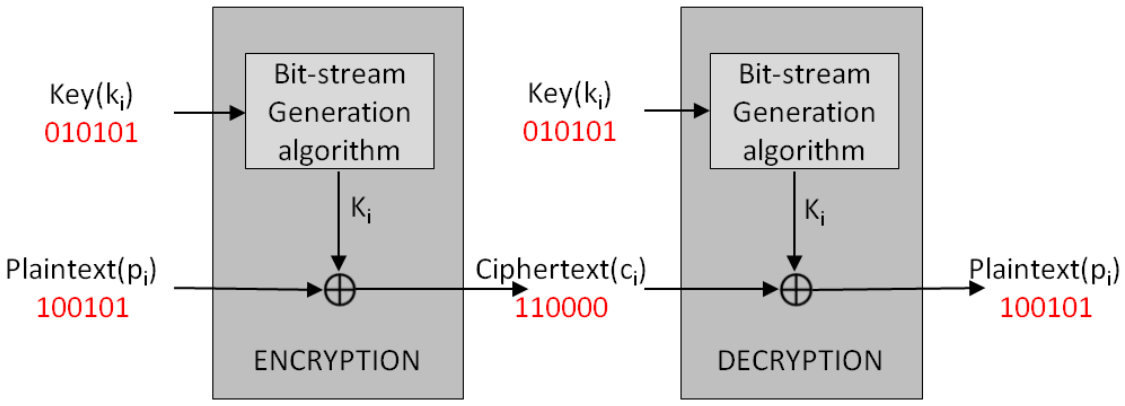
Stream Cipher

- A stream cipher is one that encrypts a digital data stream one bit or one byte at a time.
- **Examples:**
  - Autokeyed Vigenère cipher
  - A5/1
  - RC4
  - Vernam cipher.

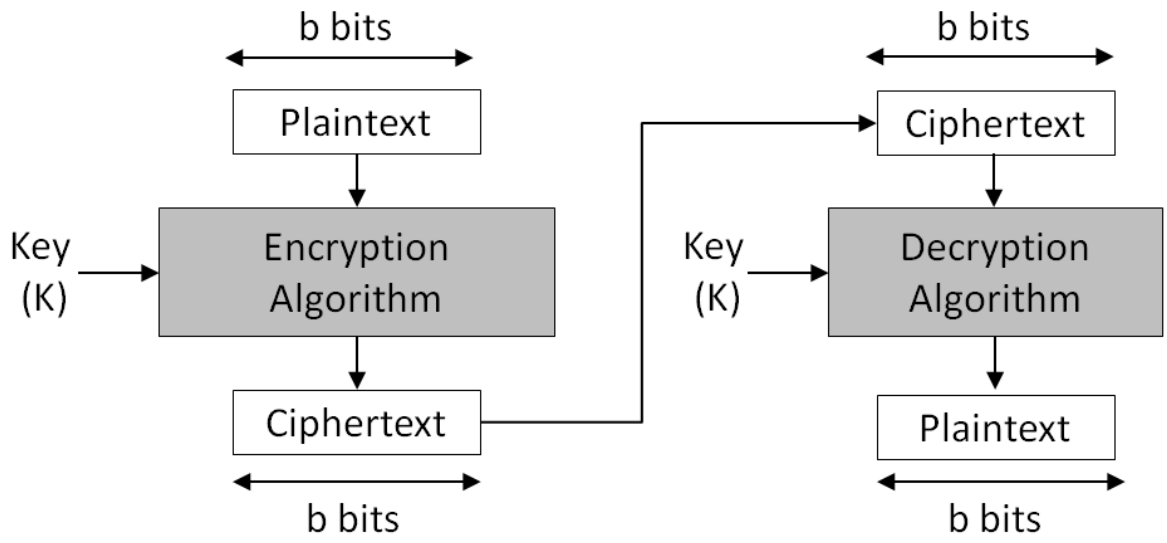
Stream Cipher



Block Cipher

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

Block Cipher



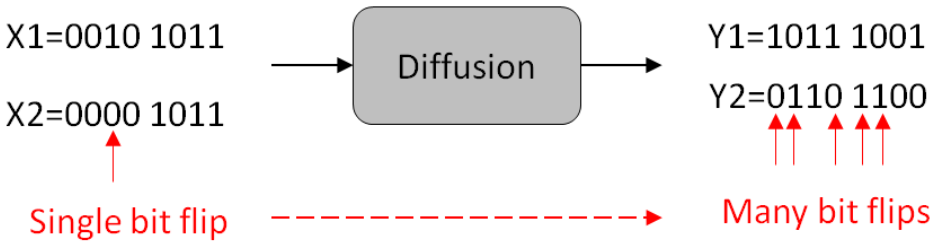
Diffusion and Confusion

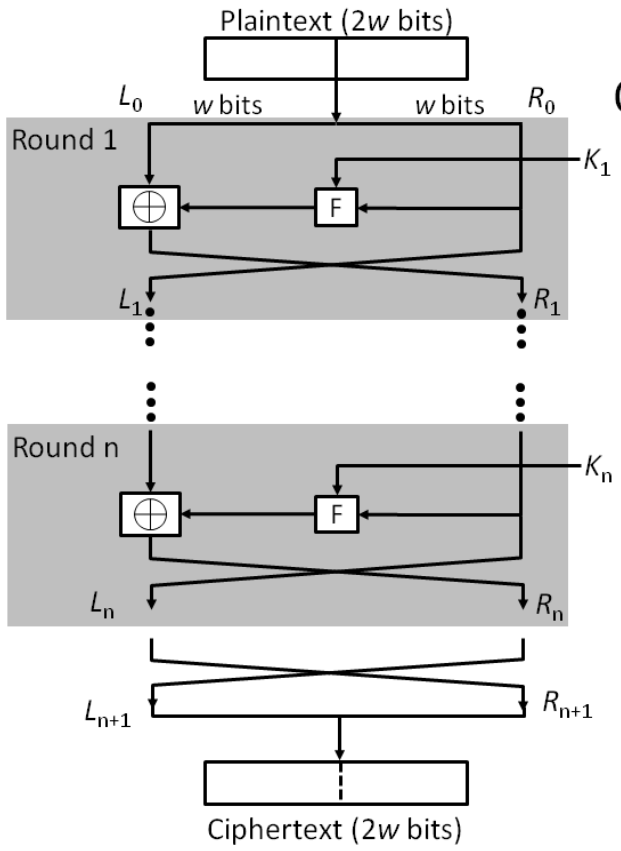
Confusion

- Confusion hides the relationship between the ciphertext and the key.
- This is achieved by the use of a complex substitution algorithm.

Diffusion

- Diffusion hides the relationship between the ciphertext and the plaintext.
- This is achieved by changing one plaintext digit which affect the value of many ciphertext digits.





Feistel Cipher Structure Or Block Cipher Structure

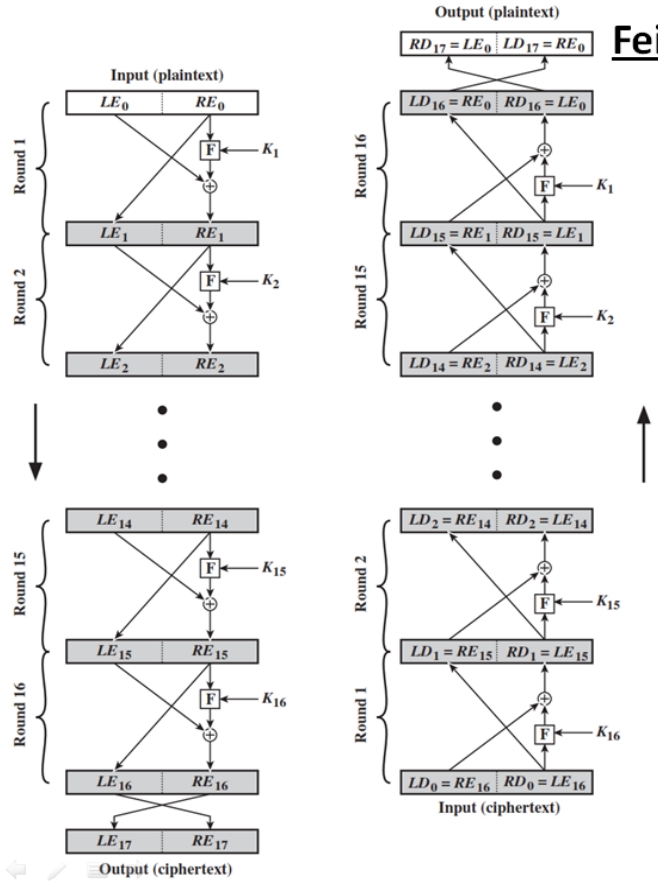
1. Plaintext is split into 32-bit halves L_i and R_i
2. R_i is fed into the function F.
3. The output of function F is then XORed with L_i
4. Left and right half are swapped.

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

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

Feistel Network Factors

- **Block size:** Common block size of 64-bit. However, the new algorithms uses a 128-bit, 256-bit block size.
- **Key size:** Key sizes of 64 bits or less are now widely considered to be insufficient, and 128 bits has become a common size.
- **Number of rounds:** A typical size is 16 rounds.
- **Round function F:** This phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. Again, greater complexity generally means greater resistance to cryptanalysis.
- **Subkey generation algorithm:** For each of the sixteen rounds, a different subkey (K_i) derived from main key by the combination of a left circular shift and a permutation. Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.



Feistel Encryption & Decryption

- Prove that o/p of first round of Decryption is equal to 32-bit swap of i/p of 16th round of Encryption
- LD_1=RE_{15} & RD_1=LE_{15}
- On Encryption Side:

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

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

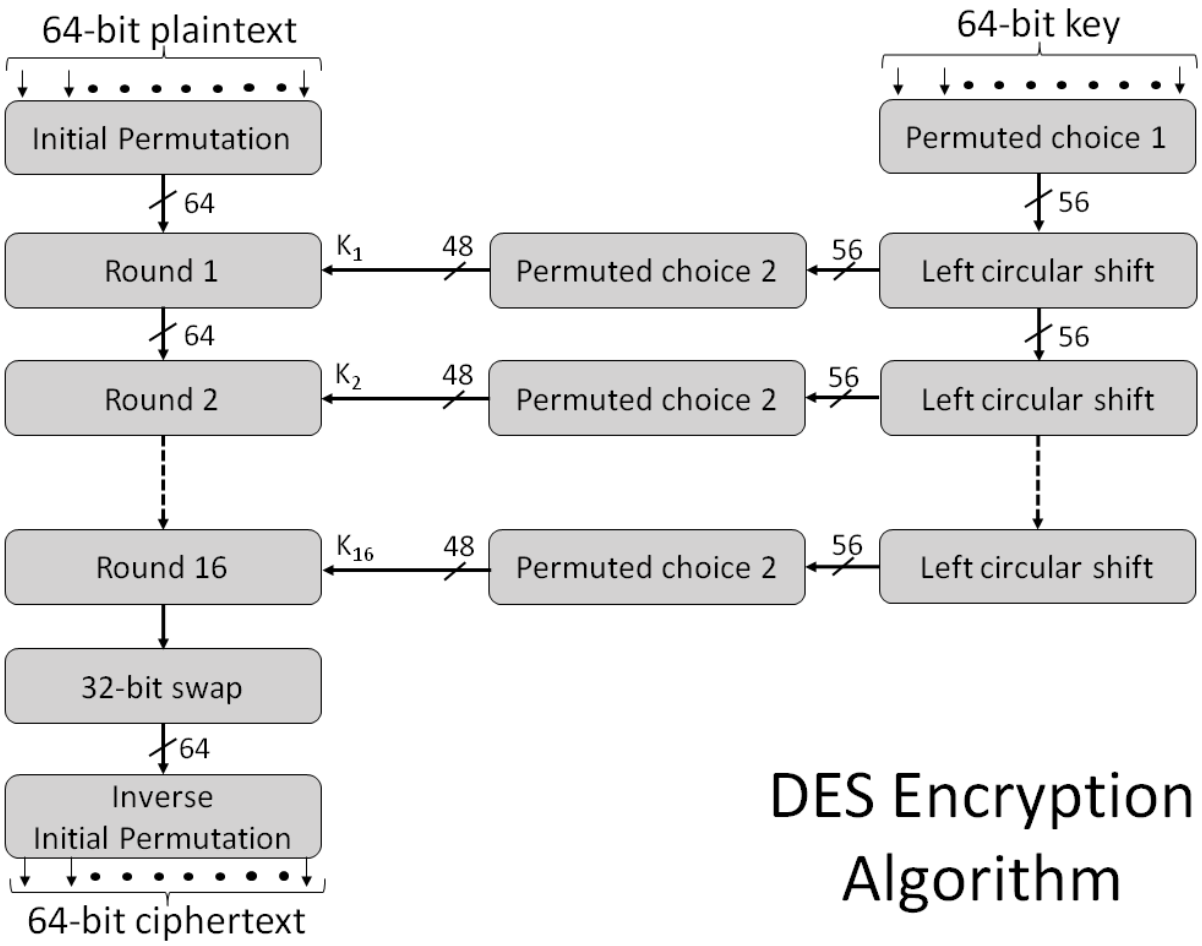
- On Decryption Side:

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$

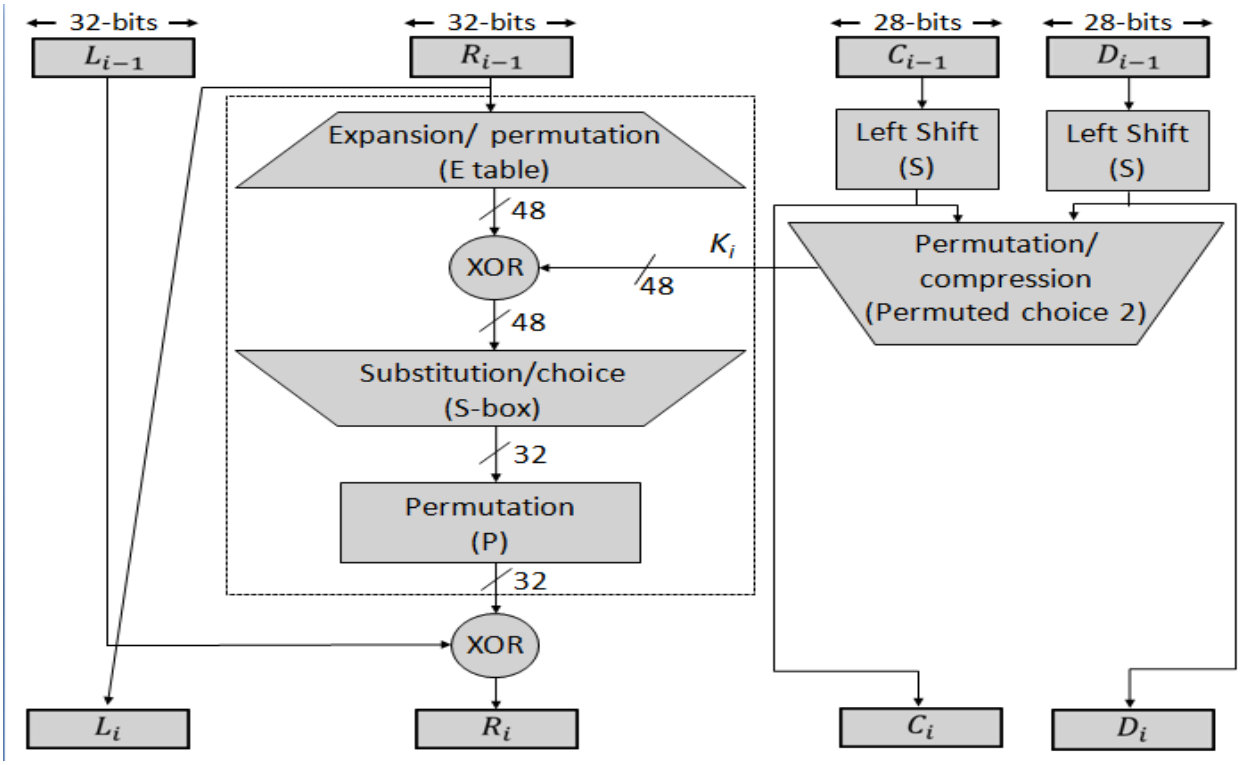
Data Encryption Standard (DES)

- Type: Block Cipher
- Block Size : 64-bit
- Key Size: 64-bit, with only 56-bit effective
- Number of Rounds: 16



DES Encryption Algorithm

DES Single Round



DES Single Round (Cont…)

1. Key Transformation
   - Permutation of selection of sub-key from original key
2. Expansion Permutation (E-table)
   - Right half is expanded from 32-bits to 48-bits
3. S-box Substitution
   - Accepts 48-bits from XOR operation and produce 32-bits using 8 substitution boxes (each S-boxes has a 6-bit i/p and 4-bit o/p).
4. P-Box Permutation
5. XOR and Swap

DES Encryption Algorithm (Cont...)

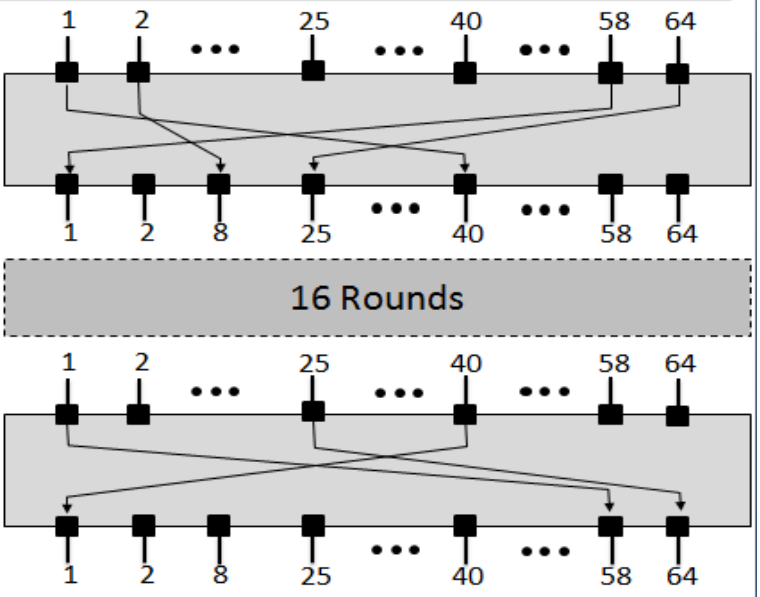
- First, the 64-bit plaintext passes through an **initial permutation** (IP) that rearranges the bits to produce the permuted input.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both **permutation** and **substitution** functions.
- Finally, the preoutput is passed through a permutation that is the **inverse of the initial permutation** function, to produce the 64-bit ciphertext.
- The 56-bit key is passed through a **permutation function**.
- For each of the sixteen rounds, a subkey (K_i) is produced by the combination of a **left circular shift** and a **permutation**.

DES Encryption Algorithm

1. **Initial permutation:** First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
2. **The F function:** This phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions.
3. **Swap:** L and R swapped again at the end of the cipher, i.e., after round 16 followed by a final permutation.
4. **Inverse (Final) permutation:** It is the inverse of the initial permutation.
5. **Subkey generation:** For each of the sixteen rounds, a different subkey (K_i) derived from main key by the combination of a left circular shift and a permutation.

Initial and Inverse Permutation

- The initial permutation of the DES algorithm changes the order of the plaintext prior to the first round of encryption.
- The final permutation occurs after the sixteen rounds of DES are completed. It is the inverse of the initial permutation.



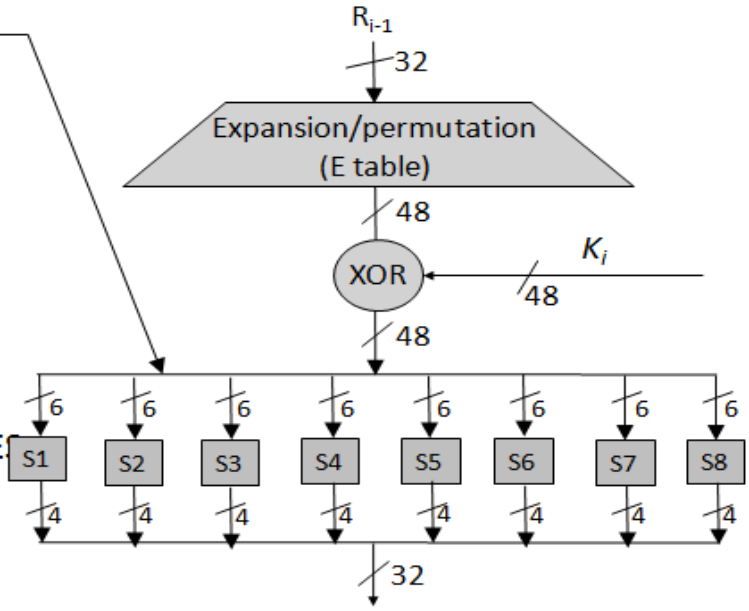
Initial and Final Permutation

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

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

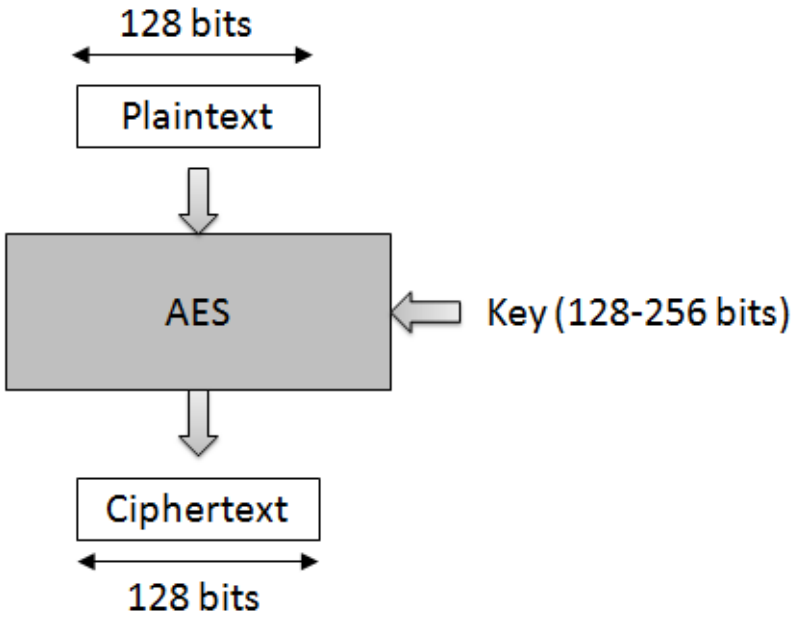
The DES S-Boxes

- S-Box substitution.
- Eight substitution tables.
- 6 bits of input
- 4 bits of output.
- Convert 48 bits to 32 bits

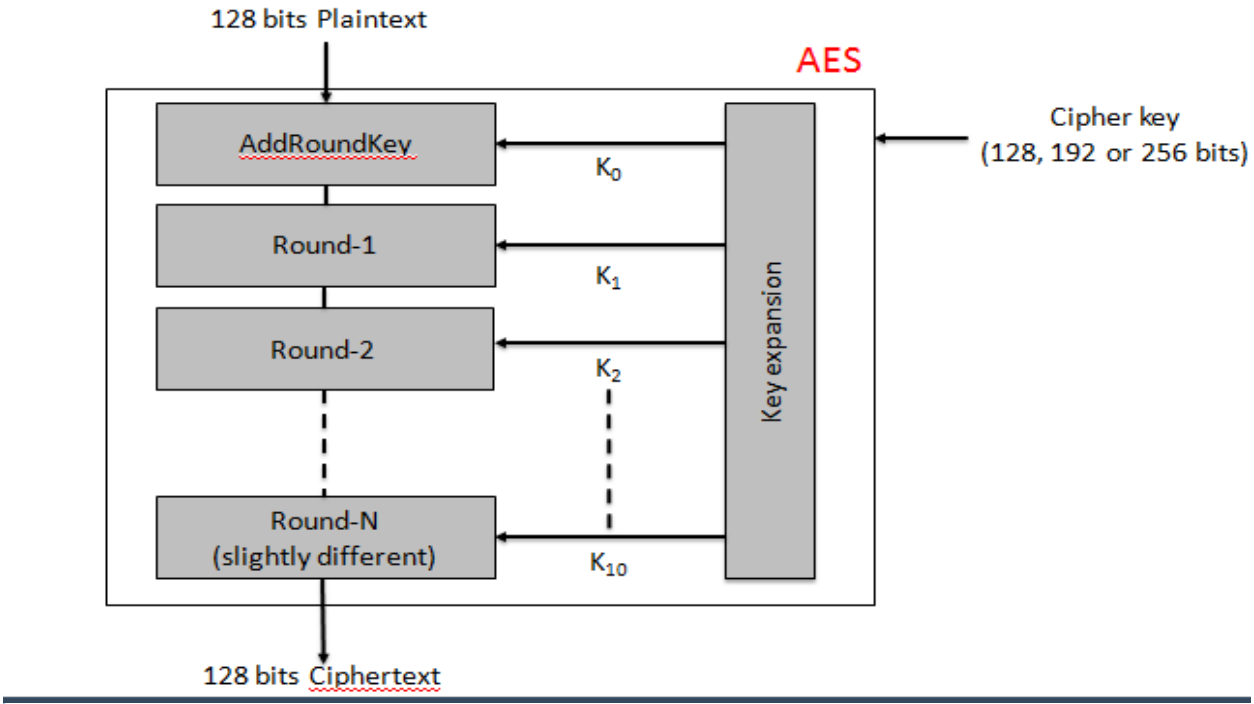


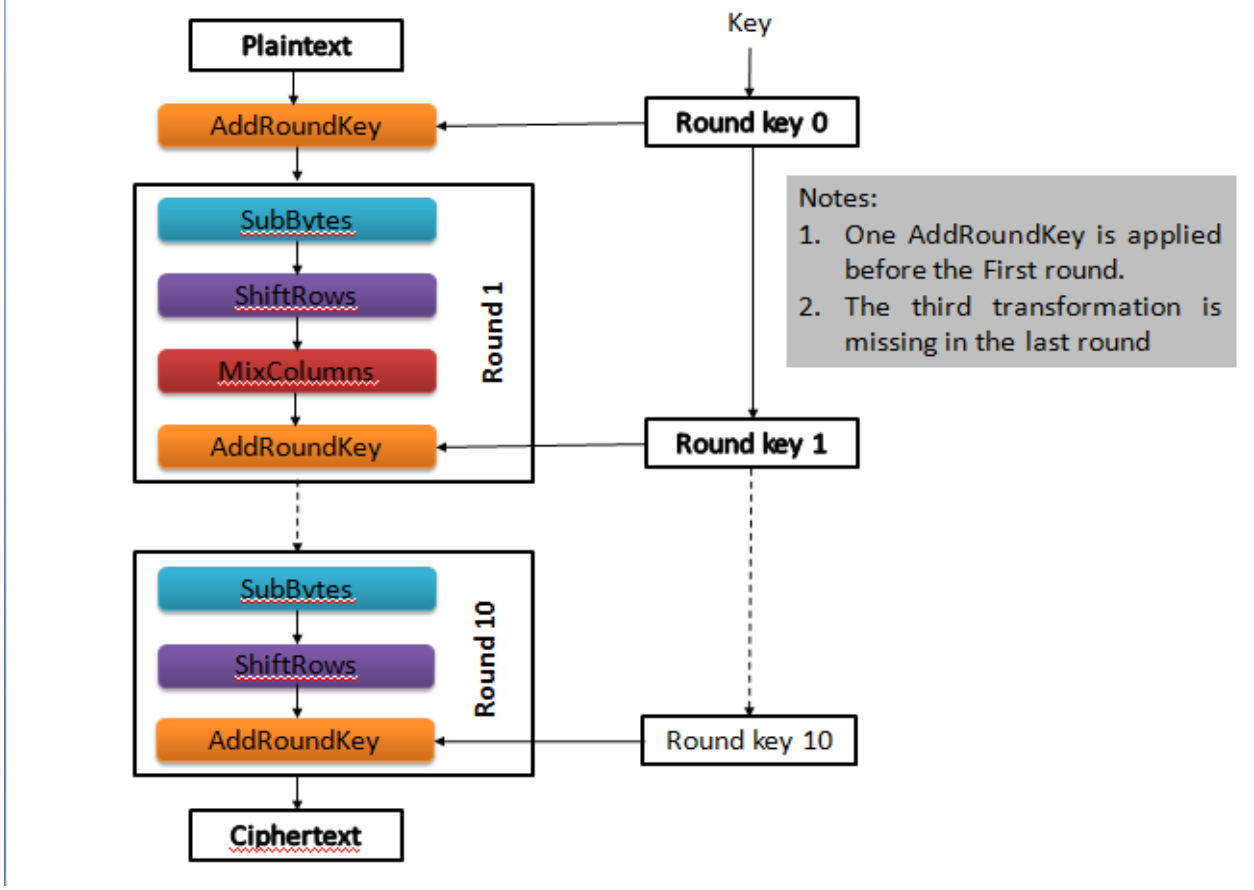
- Crucial element for DES security!
- Introduces confusion.

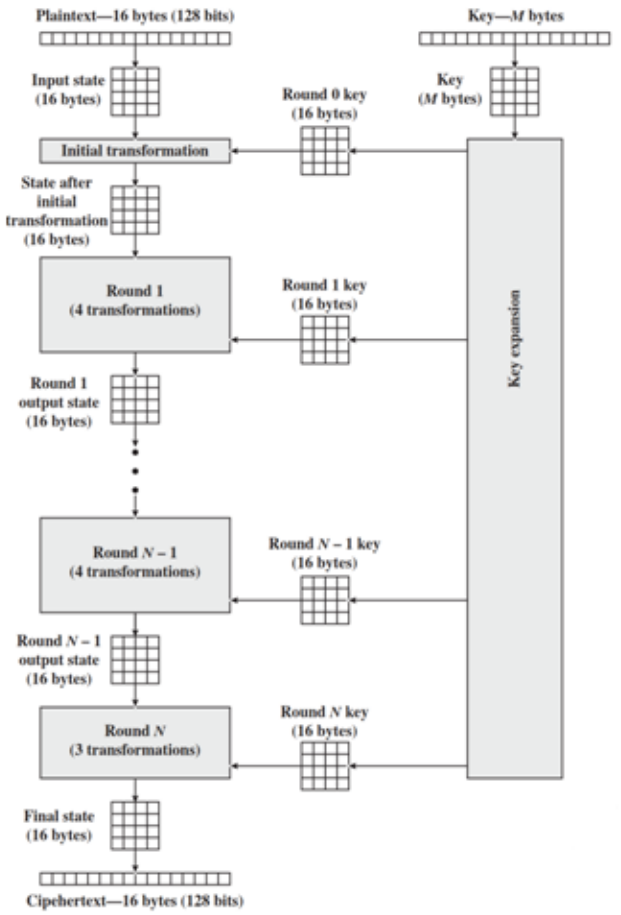
AES (Advanced Encryption Standard)



AES (Advanced Encryption Standard)







AES Structure

Initialization
1. Expand 16-byte key to get the actual key block to be used.
2. Initialize 16-byte plaintext block called as state.
3. XOR the state with the key block.

For each round
1. Apply S-box
2. Rotate rows of state
3. Mix columns
4. Add Round key: XOR the state with key block.

Plain Text to State

- Click to add text

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

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

AES Structure

- The first N-1 rounds consist of four distinct transformation functions.

SubBytes	• The 16 input bytes are substituted using an S-box
ShiftRows	• Each of the four rows of the matrix is shifted to the left
MixColumns	• Each column of four bytes is now transformed using a special mathematical function.
AddRoundKey	• The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key.

AES structure

State:

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

Cipher key:

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

AES structure

State:

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

Cipher key:

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Initial transformation(AddRoundKey)

AddRoundKey: input state $\oplus$ Cipher key

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

$\oplus$

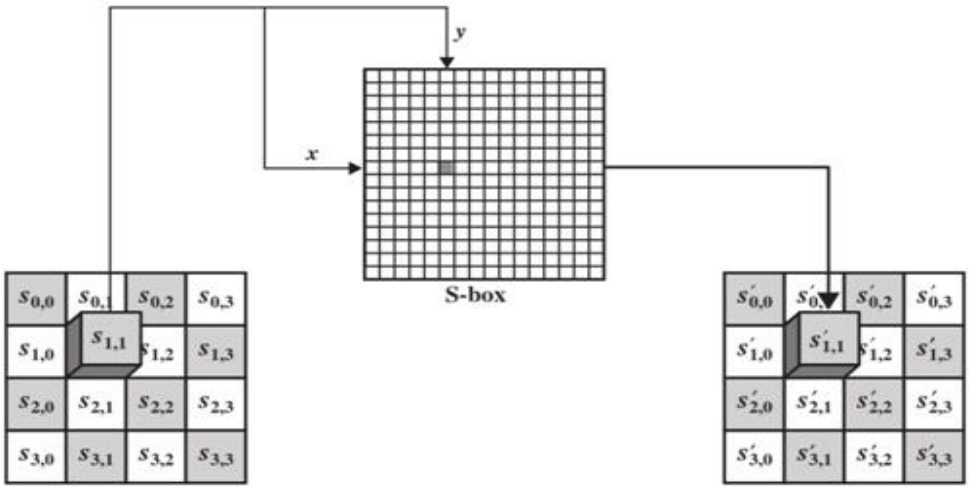
2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

=

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

SubByte Transformation

- The forward substitute byte transformation, called SubBytes, is a simple table lookup



SubByte output

Input for SubByte

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

Output of SubByte

d4	e0	b8	le
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

ShiftRows

- The first row of State is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

d4	e0	b8	le	← No rotation
27	bf	b4	41	← Rotate 1 byte
11	98	5d	52	← Rotate 2 bytes
ae	f1	e5	30	← Rotate 3 bytes

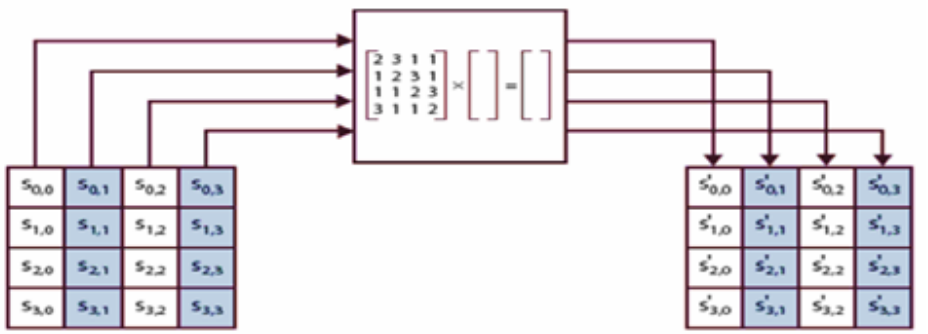
Input for ShiftRows

d4	e0	b8	le
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Output of ShiftRows

MixColumns

- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.
- Constant matrices used by MixColumns.



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

MixColumns

d4	e0	b8	le
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

$\bullet$

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

=

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

$\bullet$

d4
bf
5d
30

=

04
66
81
e5

AddRoundKey

- In the forward add round key transformation, the 128 bits of State are bitwise XORed with the 128 bits of the round key.

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

$\oplus$

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

=

A4	68	6b	02
9c	9f	5b	6a
7f	35	Ea	50
F2	2b	43	49

04
66
81
e5

$\oplus$

a0
fa
fe
17

=

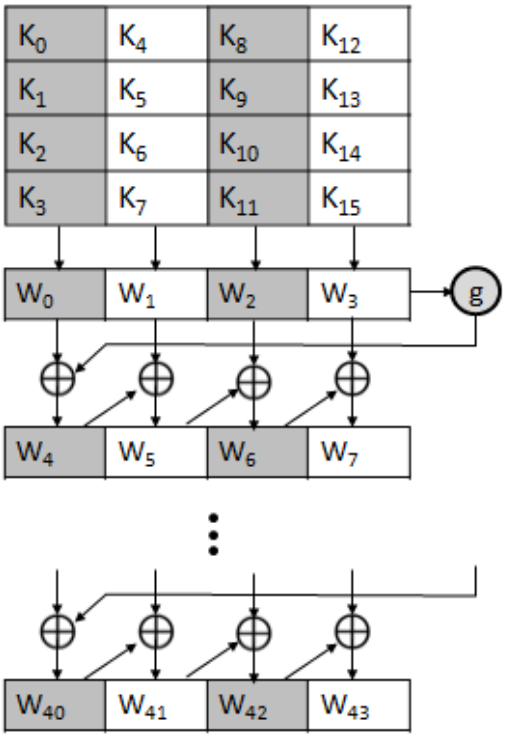
A4
9c
7f
F2

AES key expansion

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
Round 1	W_4	W_5	W_6	W_7
Round 2	W_8	W_9	W_{10}	W_{11}
. . .		. . .		
Round N	W_{40}	W_{41}	W_{42}	W_{43}

AES key expansion



- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of **44 words** (176 bytes).
- Each added word **w[i]** depends on the immediately preceding word, w[i - 1].
- In three out of four cases, a simple XOR is used.