CS-3002: Information Security

Lecture # 4: Block Ciphers

Prof. Dr. Sufian Hameed

Department of Computer Science

FAST-NUCES

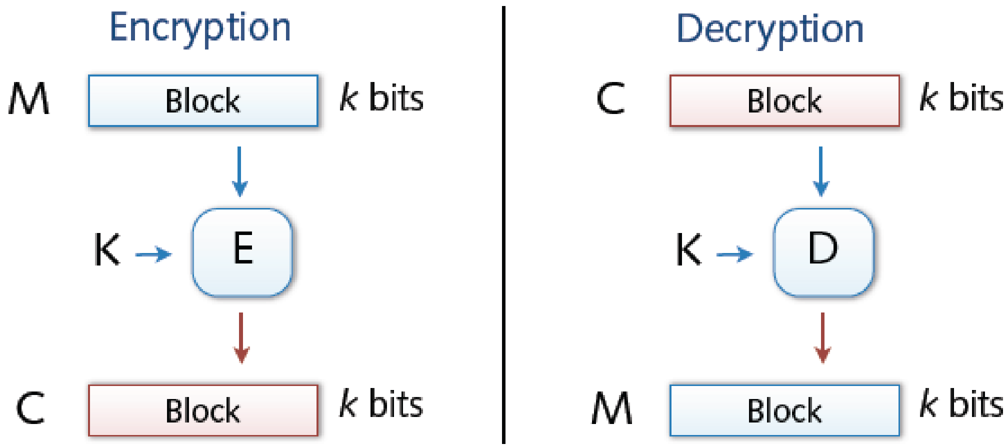
Block Cipher

Modern Ciphers

- **Modern ciphers**
  - Sophisticated design using substitutions and permutations
  - Round-based encryption and decryption algorithms
  - Efficient implementations in hardware and software
- **Common classes of modern ciphers**
  - Block ciphers = processing of fixed-length blocks
    - Examples: DES, AES, Blowfish
  - Stream ciphers = processing of individual bits or bytes
    - Examples: RC4, A5/1

Block Cipher

- Encryption and decryption in blocks (e.g., 64 or 128 bit)
- Padding of short messages, splitting of long messages

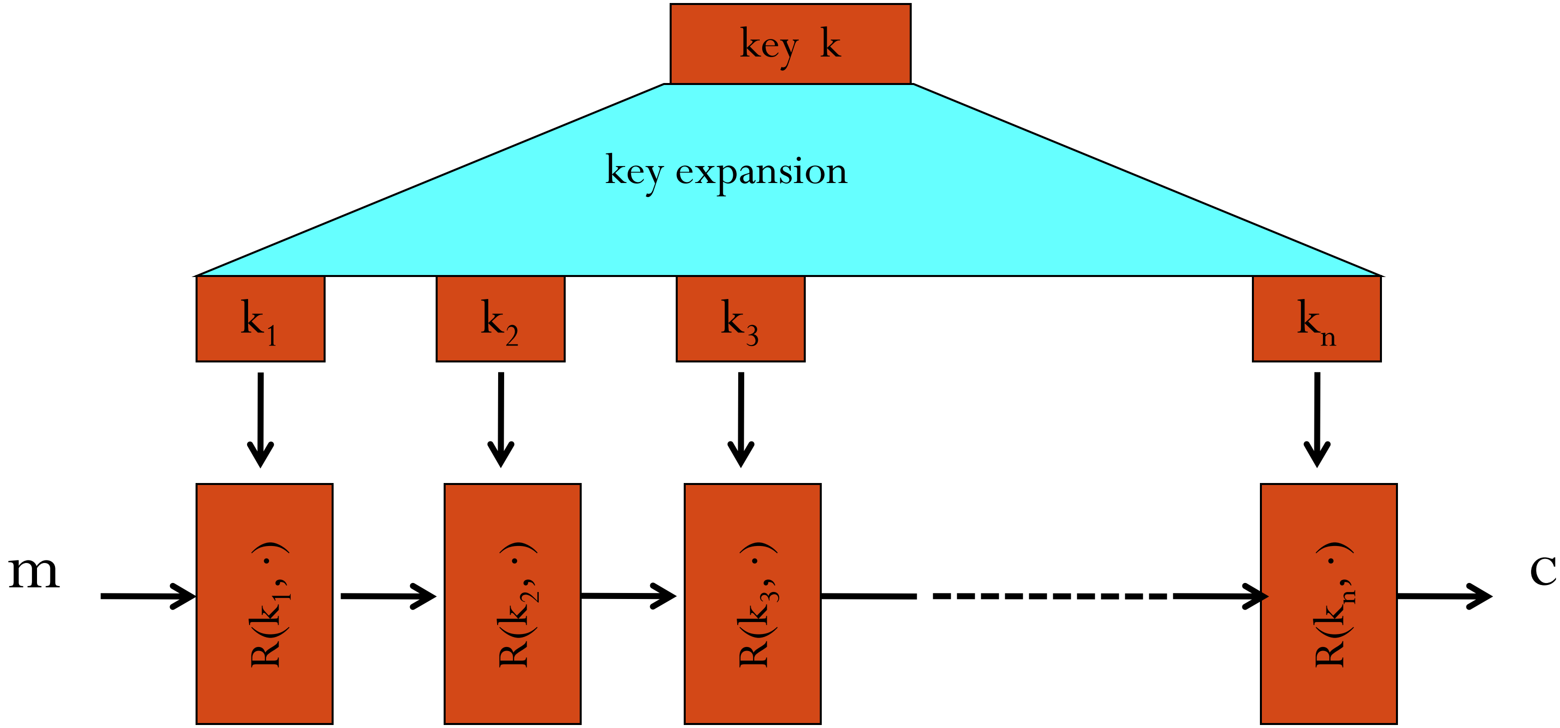


- Different modes of operations: ECB, CBC, CTR, ...

Examples:

1. DES: n= 64 bits, k = 56 bits
2. 3DES: n= 64 bits, k = 168 bits
3. AES: n=128 bits, k = 128, 192, 256 bits

Block Ciphers Built by Iteration



R(k,m) is called a round function

for DES (n=16), 3DES (n=48) and AES-128 (n=10)

Design Characteristics for Block Ciphers

- **Choice of blocklength n**
  - n too long → complex algorithm, performance loss
  - n too short → weak encryption, easy to attack
  - Modern variants use n = 128 - 256 bit
- **Choice of the key length of k**
  - Practical key length: 80 - 256 bit
  - k too short → systematic testing of all valid keys (**Brute Force attack)**
  - Against Brute Force attacks, a minimum of 70-80 bit are necessary

Confusion and Diffusion (Recap)

- **What makes a cipher secure? Hard to tell**

- Confusion property
  - Complex relation between key and plaintext/ciphertext
    - Hard to deduce key from plaintext/ciphertext pairs

- Diffusion property
  - Complex relation between plaintext and ciphertext
    - Hard to deduce bits of plaintext from ciphertext

Data Encryption Standard (DES)

The Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
  - key-len = 128 bits ; block-len = 128 bits
- 1973: National Bureau of Standards (NBS) asks for block cipher proposals
  - IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  - key-len = 56 bits ; block-len = 64 bits
- Every 5 *years:*
  - DES review for decision about further usage
  - Result: Until now, no modifications were made

The Data Encryption Standard (DES)

- Problem of DES:
  - Usage of a **key with a length of 56 bit**
  - Criticized for a key length too short for usage in practice

- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

DES Challenge

- 1997: Internet search -- **3 months**

- 1998: by distributed.net in **41 days** in early 1998.

  - The plaintext was "The secret message is: Many hands make light work."

- 1998: Electronic Frontier Foundation (EFF) machine (deep crack) -- **56 hours**

  - The text was revealed to be "The secret message is: It's time for those 128-, 192-, and 256-bit keys."

- 1999: combined search -- **22 hours**

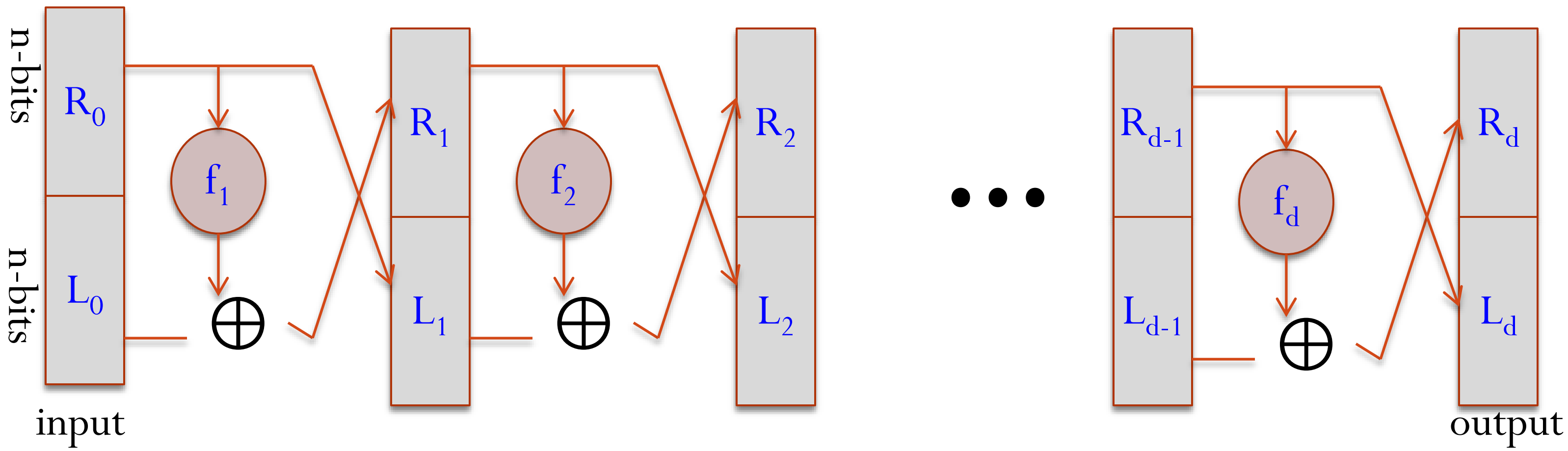
  - The plaintext was *See you in Rome (second AES Conference, March 22-23, 1999)*

$\Rightarrow$ 56-bit ciphers should not be used !! (128-bit key $\Rightarrow 2^{72}$ days)

DES: core idea – Feistel Network

DES is based on Feistal Network

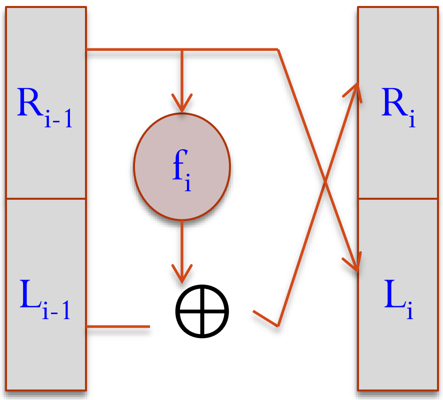
Consists of invertible function F: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$



In symbols: $\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$

Inverse of Feistal Function

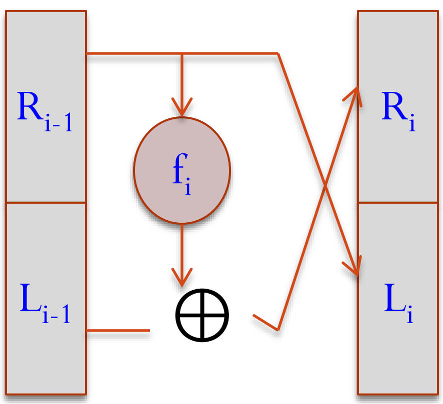
- Feistel network F: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is invertible



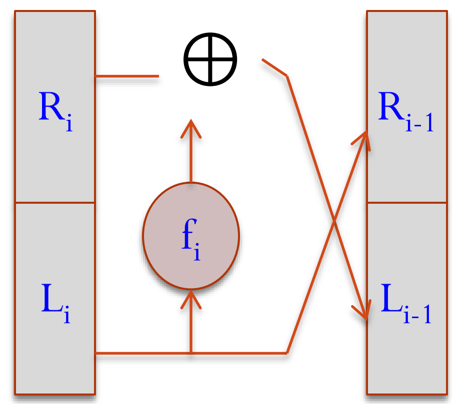
inverse

$R_{i-1} = L_i$

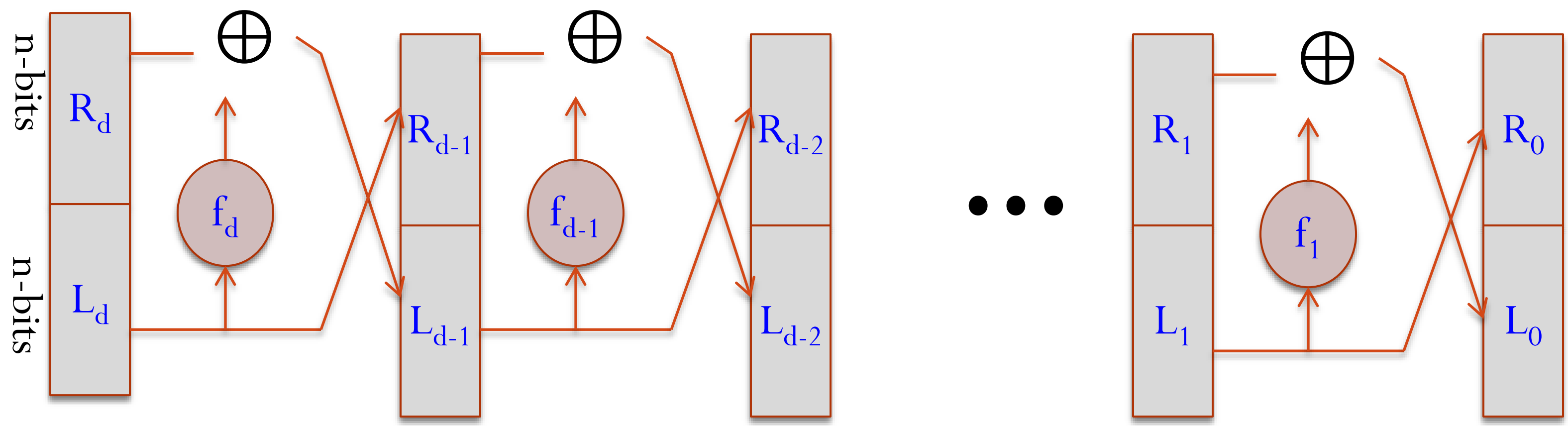
$L_{i-1} =$ [redacted]



inverse



Decryption circuit



- Decryption is basically the same Inverted circuit, with $f_1, \ldots, f_d$ applied in reverse order

General Structure of DES

- DES uses blocks of length $n = 64$ *bit*
  - Length of key k *is 56 bit*
  - Encryption takes place in 16 identical rounds with round keys k_i *of 48 bit length*

Encryption process

1. step: permutation performed on the input block

2. step: generation of round keys

3. step: performing 16 identical rounds

4. step: inverse permutation to step 1

DES Encryption Process

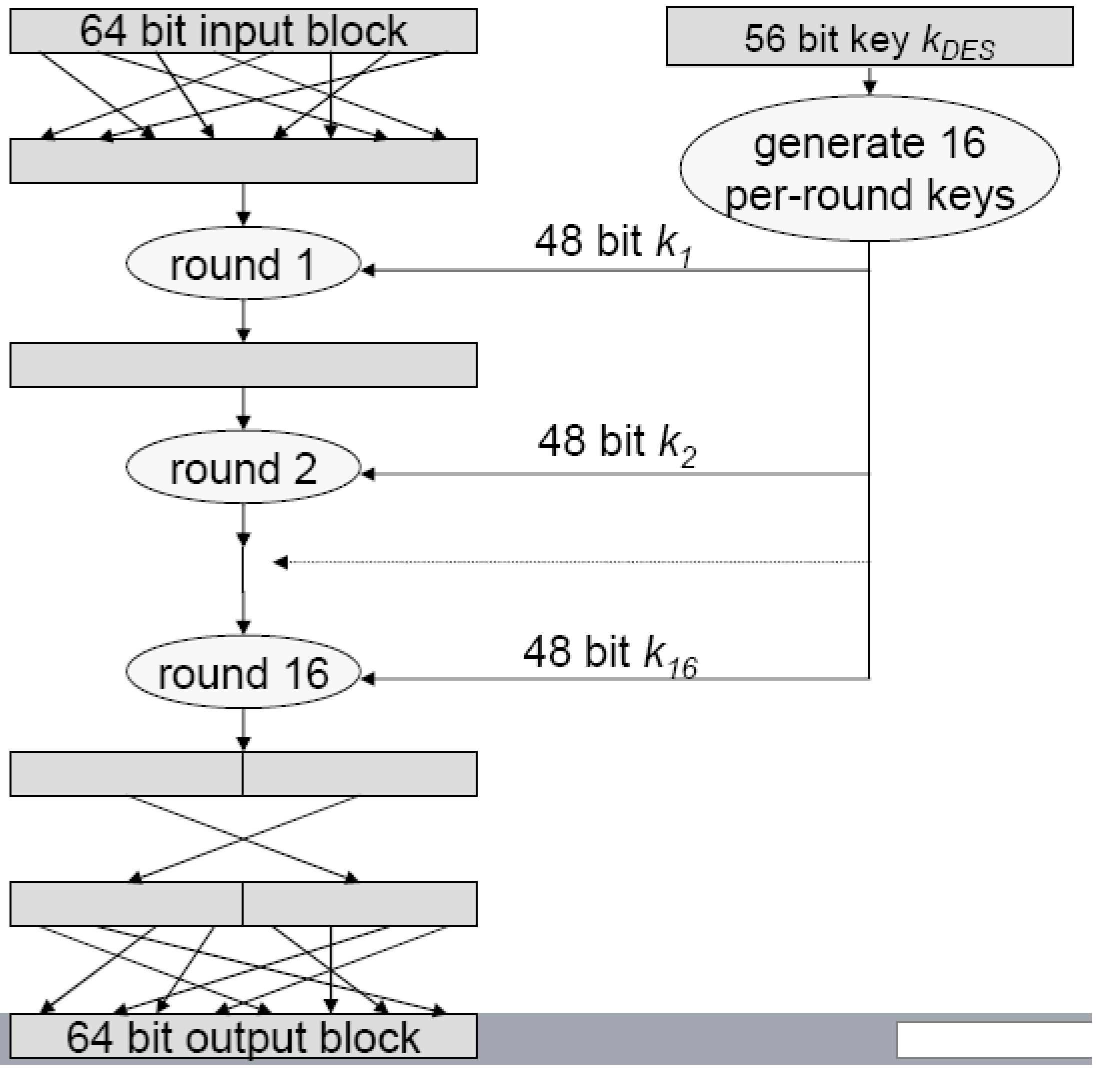
1. Initial permutation

2. Generation of round keys

3. Encryption in 16 identical rounds:
   a. Substitution
   b. Permutation

Additional step: swap left and right halves

4. Final permutation



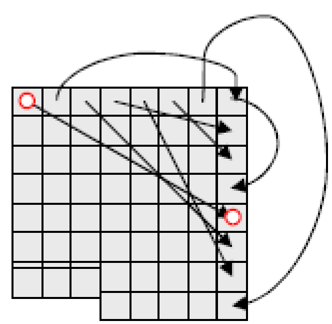
Step 1 & 4: Initial and Final Permutation

Input permutation

- See each 64-bit block as 8 Bytes, arranged in a matrix
- Diffusion of bits over all bytes
  - Bits of a column are packed into a row
  - First byte is spread into 8th bits of all bytes
  - Second byte is spread into 7th bits of all bytes
  - ...

Input 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



40 goes to 1

1 goes to 40

Output 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

Step 2: Generation of Round Keys

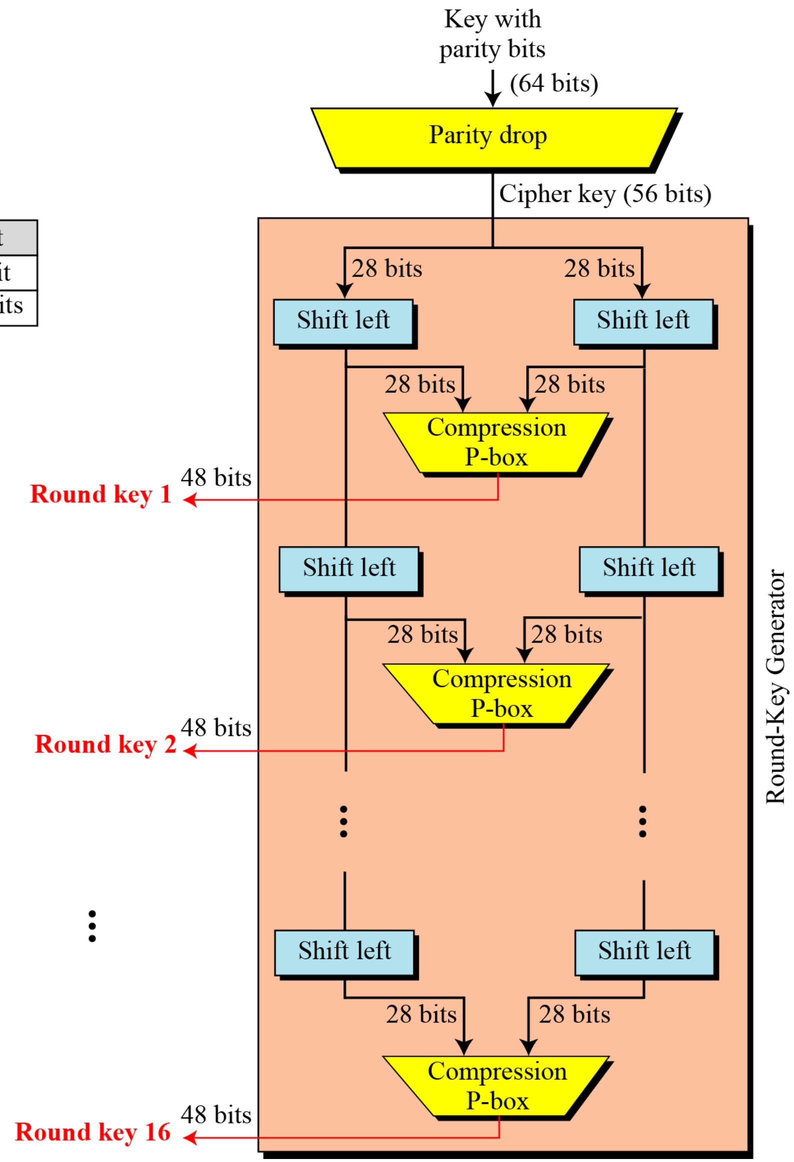
- **Preparation:** divide K_{DES} into left block (C_0) and right block (D_0) (each 28 bit long - no parity bits) by performing permutations similar to DES initial permutation (which has no security value)

- **Now:** round keys K_i are computed in 16 rounds from C_i and D_i:

Step 2: Generation of Round Keys

Shifting

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



Step 2: Generation of Round Keys

- Initial Key permutation and parity bit drop table

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

- Number of bits shift

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

Step 2: Generation of Round Keys

- Characteristics of key generation
  - Left shift
  - round 1, 2, 9, and 16: left shift of 1 bit
  - other rounds: left shift of 2 bits
  - Notice: 10 years later it was found, that performing the left shift with varying step sizes makes the algorithm more secure
  - Left half of Ki is only determined by Ci, right side only by Di
- Permutations (compression P-box):
  - perform permutations on remaining bits of Ci and Di to obtain Ki (48 bits)
  - Ci: bits 9, 18, 22, and 25 are discarded (remaining: 24 bits)
  - Di: bits 35, 38, 43, and 54 are discarded (remaining: 24 bits)

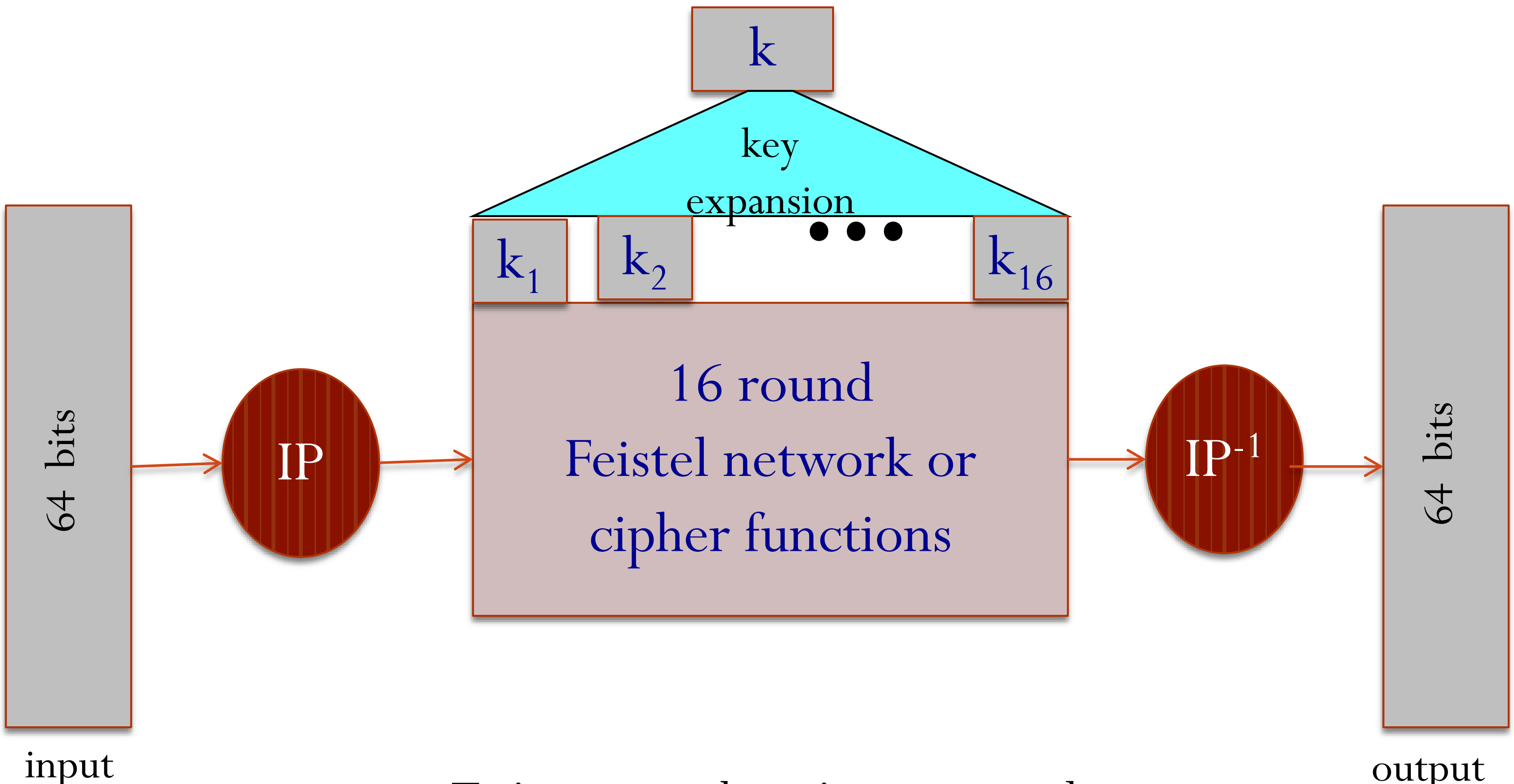
- Notice: the choice of the permutations on Ci and Di influence the security of DES, because they determine the quality of the round keys

Step 2: Generation of Round Keys

- Compression P-Box (key compression table)

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

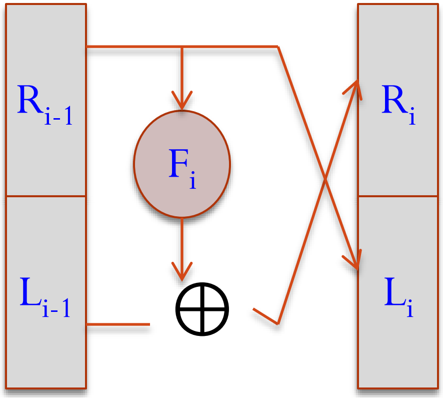
Step 3: Application of Rounds (16 round Feistel network)



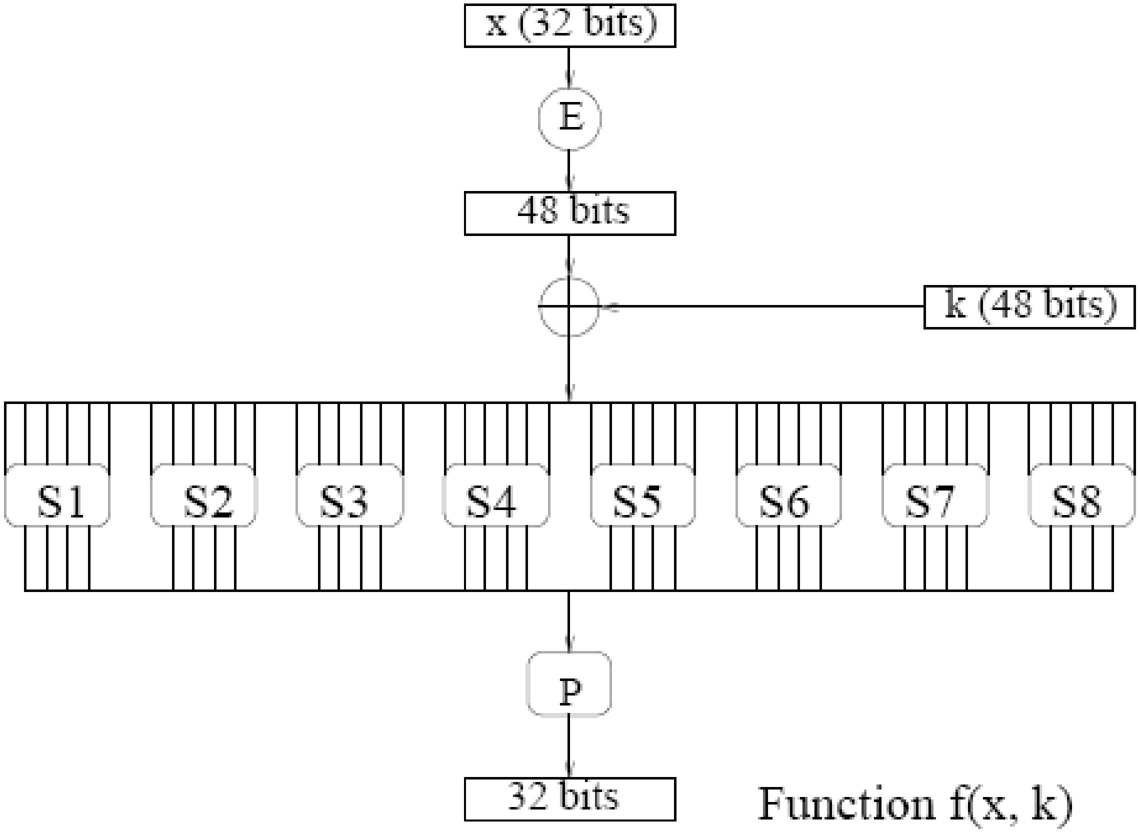
To invert, use keys in reverse order

One DES round

- Divide input block into two 32-bit blocks Li and Ri
- Compute Li+1 as Ri, and Ri+1 as Li ⊕ F(Ri, Ki)
- F is **cipher function, i.e. combination of substitution and permutation**
- Security provided by DES depends on the quality of the cipher function
- Decryption: uses the same algorithm, has same expense like encryption



The function $F(x, k_i)$

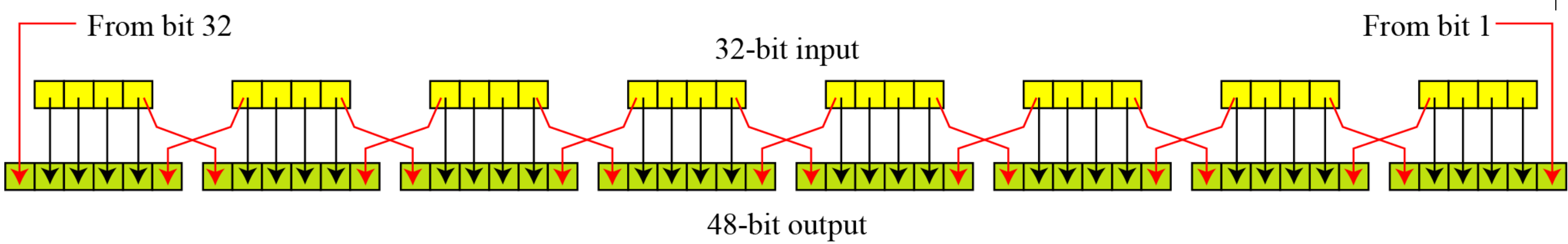


S-box: function $\{0,1\}^6 \longrightarrow \{0,1\}^4$, implemented as look-up table.

E: Expansion P-box

- since R_{i-1} is a 32-bit input and k_i is a 48-bit key, we first need to expand R_{i-1} to 48 bits.

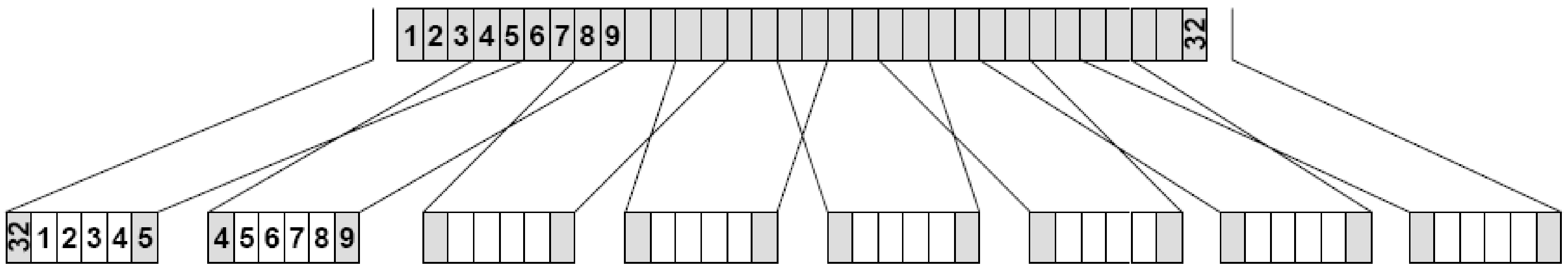
Expansion Permutation



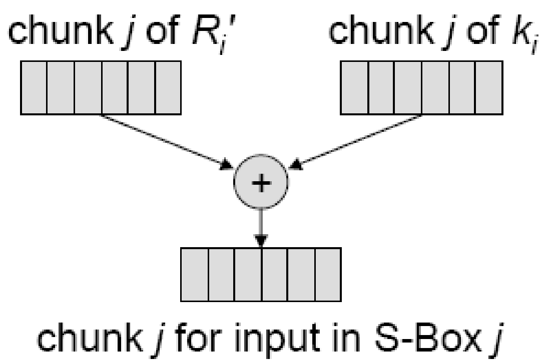
32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

Expansion P-Box Table

XOR: Key and Expanded R bits

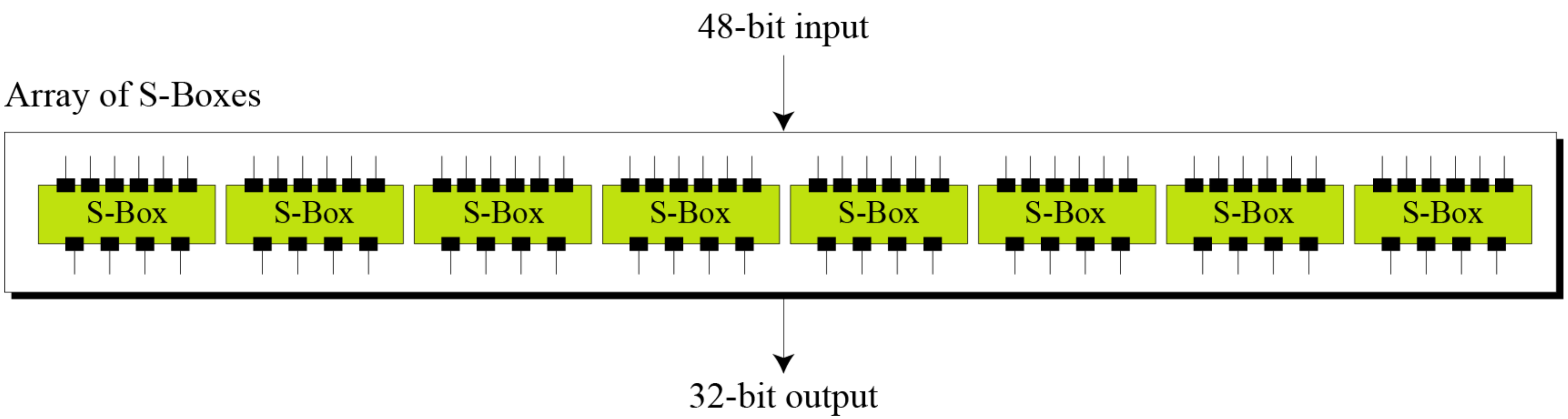


- Divide round key k_i into 8 chunks of 6 bit
- Perform XOR operation on R_i' and k_i chunks
- Use resulting chunks as input for *S-Boxes*



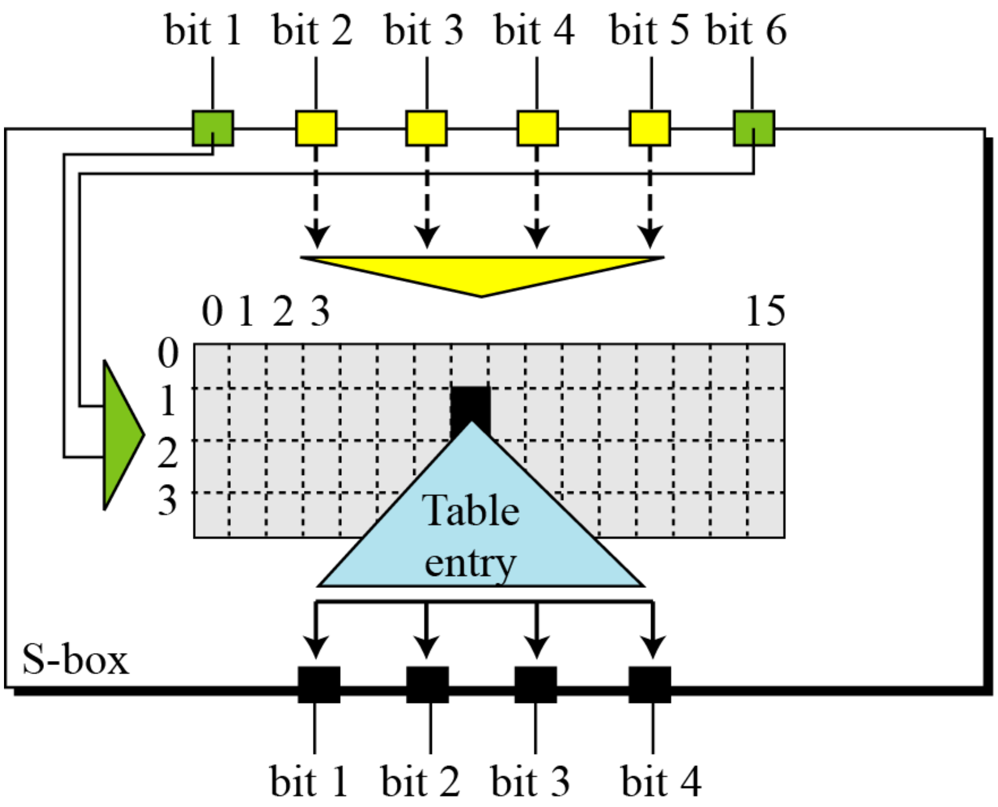
Application of S-Boxes

- *The S-boxes do the real mixing (confusion --- nonlinearity). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.*
- $S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$



The S-Boxes

- S-Box rule



The S-Boxes

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Final Straight Permutation on S-Box Output

- Straight Permutation Table

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

Example

We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536	Key: AABB09182736CCDD
CipherText: C0B7A8D05F3A829C	

Trace of data for Example

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

Example (contd)

Trace of data for Example (Conintued

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

Example (contd)

Let us see how Bob, at the destination, can decipher the ciphertext received from Alice using the same key. Table 6.16 shows some interesting points.

Ciphertext: C0B7A8D05F3A829C			
After initial permutation: 19BA9212CF26B472 After splitting: L_0=19BA9212 R_0=CF26B472			
Round	*Left*	*Right*	*Round Key*
Round 1	CF26B472	BD2DD2AB	181C5D75C66D
Round 2	BD2DD2AB	387CCDAA	3330C5D9A36D
. . .	. . .	. . .	. . .
Round 15	5A78E394	18CA18AD	4568581ABCCE
Round 16	14A7D678	18CA18AD	194CD072DE8C
After combination: 14A7D67818CA18AD			
Plaintext:123456ABCD132536 (after final permutation)			

DES Design Criteria

What are the design criteria for the building blocks of the DES algorithm?

This is out of the scope of this course. Interested parties are referred to the following references:

- B. Schneier, Applied Cryptography, 2nd Edition, John Wiley & Sons, 1996, pp. 293–294.
- D. Coppersmith, The Data Encryption Standard (DES) and Its Strength Against Attacks, IBM Journal of Research and Development, May 1994.

Comments

- Security of DES
  - DES is seen as very secure (except for the key length)
  - No attacks with lower costs than a Brute Force attack are known as far
  - There are some so-called weak keys and semi-weak keys
  - These keys should not be used!
- Questions on DES
  - Design process for DES was not public
  - Are details well-chosen for strength of the DES algorithm?
  - Are some weaknesses useful for people involved in the design process?
  - Are there other weak keys than the known ones?

Choosing the S-boxes and P-box

- Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs) [BS'89]
- Several rules used in choice of S and P boxes:
  - No output bit should be close to a linear function of the input bits
  - Ensure that bits of the output of an S-Box on one round affects the input of multiple S-Boxes in the next round
  - Two of the output bits of one S-Box should influence the middle of the result, the other two bits should influence the edges
  - The 4 output bits should form the input of 6 S-Boxes in the next round

⋮

Properties

Two desired properties of a block cipher are the completeness and the avalanche effect.

- Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext
- Avalanche effect means a small change in either the plaintext or the key should produce a significant change in the ciphertext.
  - The avalanche effect is in fact a measure of diffusion.
  - Remark: Linear functions are usually for diffusion.

Example of Avalanche Effect

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: 000000000000000**1** Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

Example of Avalanche Effect

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits. This means that changing approximately 1.5 percent of the plaintext creates a change of approximately 45 percent in the ciphertext.

Number of bit differences for Example

Rounds	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

To Summarize

S-Boxes

The design provides confusion (non-linearity) of bits from each round to the next.

P-Boxes

They provide diffusion of bits.

Number of Rounds

DES uses sixteen rounds of Feistel ciphers. the ciphertext is thoroughly a random function of plaintext and ciphertext.

Security of DES

DES, as the first important block cipher, has gone through much scrutiny. Among the attempted attacks, three are of interest:

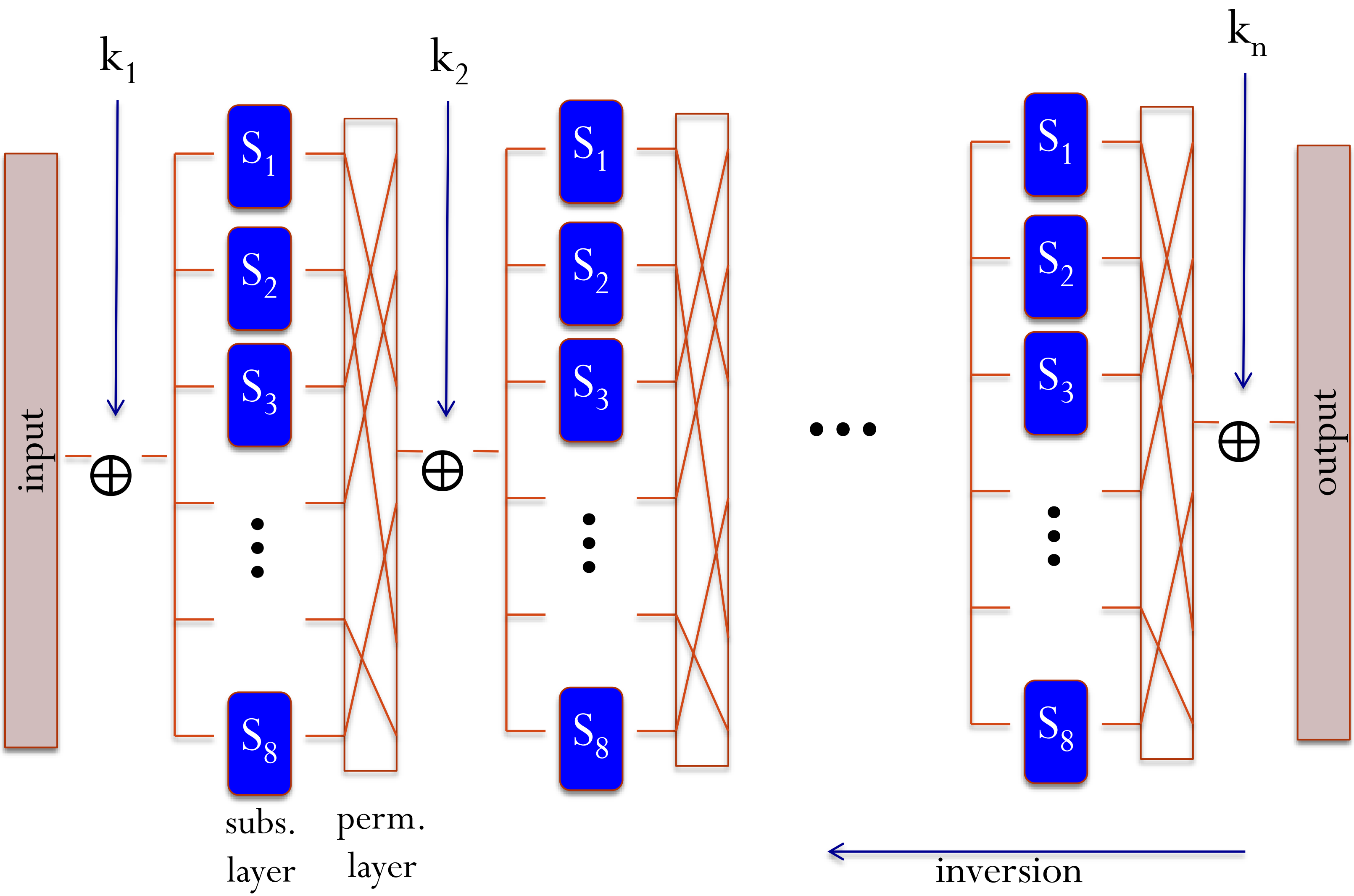
- **Brute-force**
  - Due to the weakness of short cipher key DES can be broken using 2^{55} encryptions.
- **Differential Cryptanalysis**
  - It has been revealed that the designers of DES already knew about this type of attack and designed S-boxes and chose 16 as the number of rounds to make DES specifically resistant to this type of attack.
- **Linear Cryptanalysis**
  - Linear cryptanalysis is newer than differential cryptanalysis. DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be broken using 2^{43} pairs of known plaintexts. However, from the practical point of view, finding so many pairs is very unlikely.

Advance Encryption Standard (AES)

The AES process

- 1997: NIST publishes request for proposal
- 1998: 15 submissions.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)
- Key sizes: 128, 192, 256 bits.
- Block size: 128 bits
- Rounds:
- 10 rounds of repetition for 128-bit keys.
- 12 rounds of repetition for 192-bit keys.
- 14 rounds of repetition for 256-bit keys.

AES is a Subs-Perm network (not Feistel)

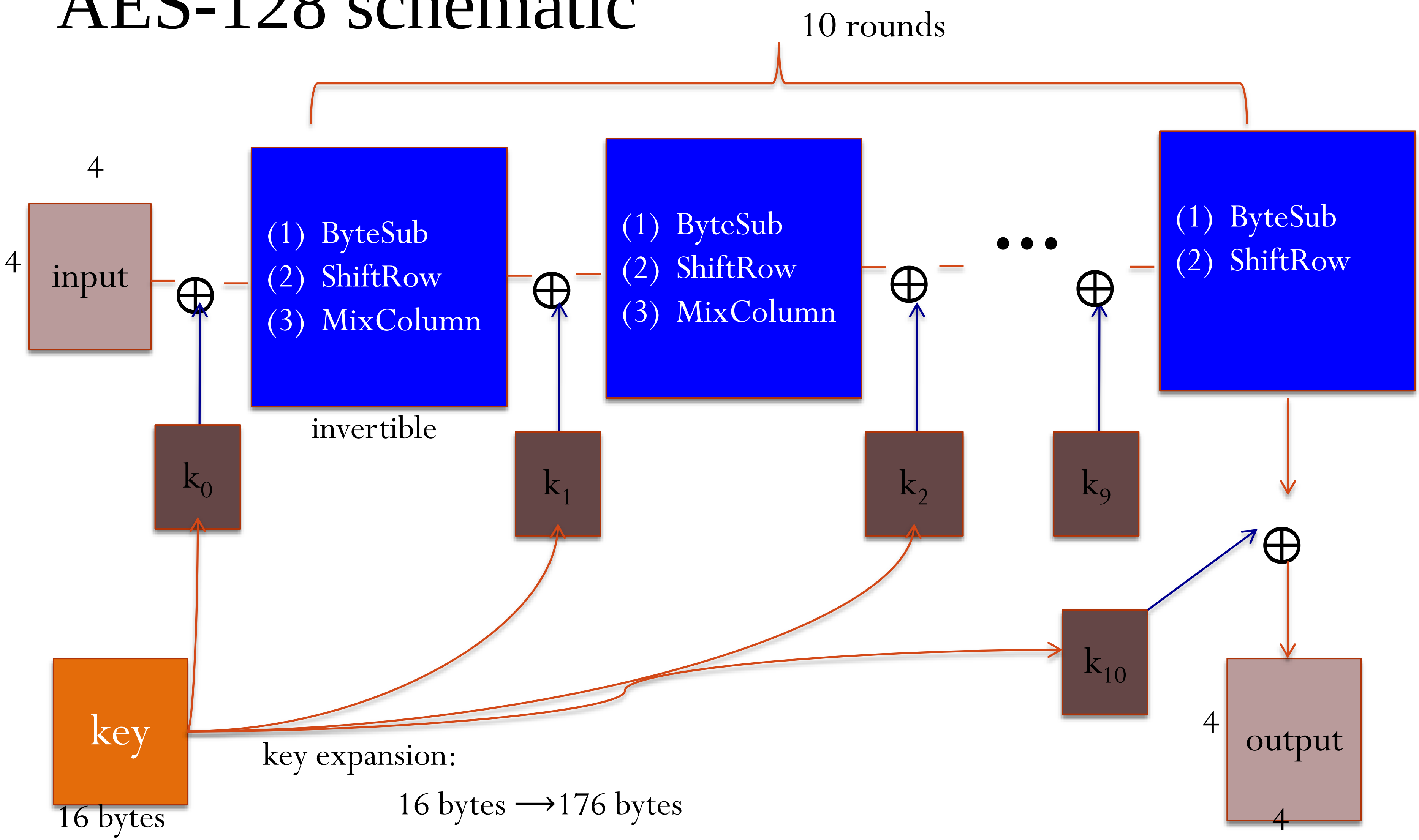


High-level Description of AES

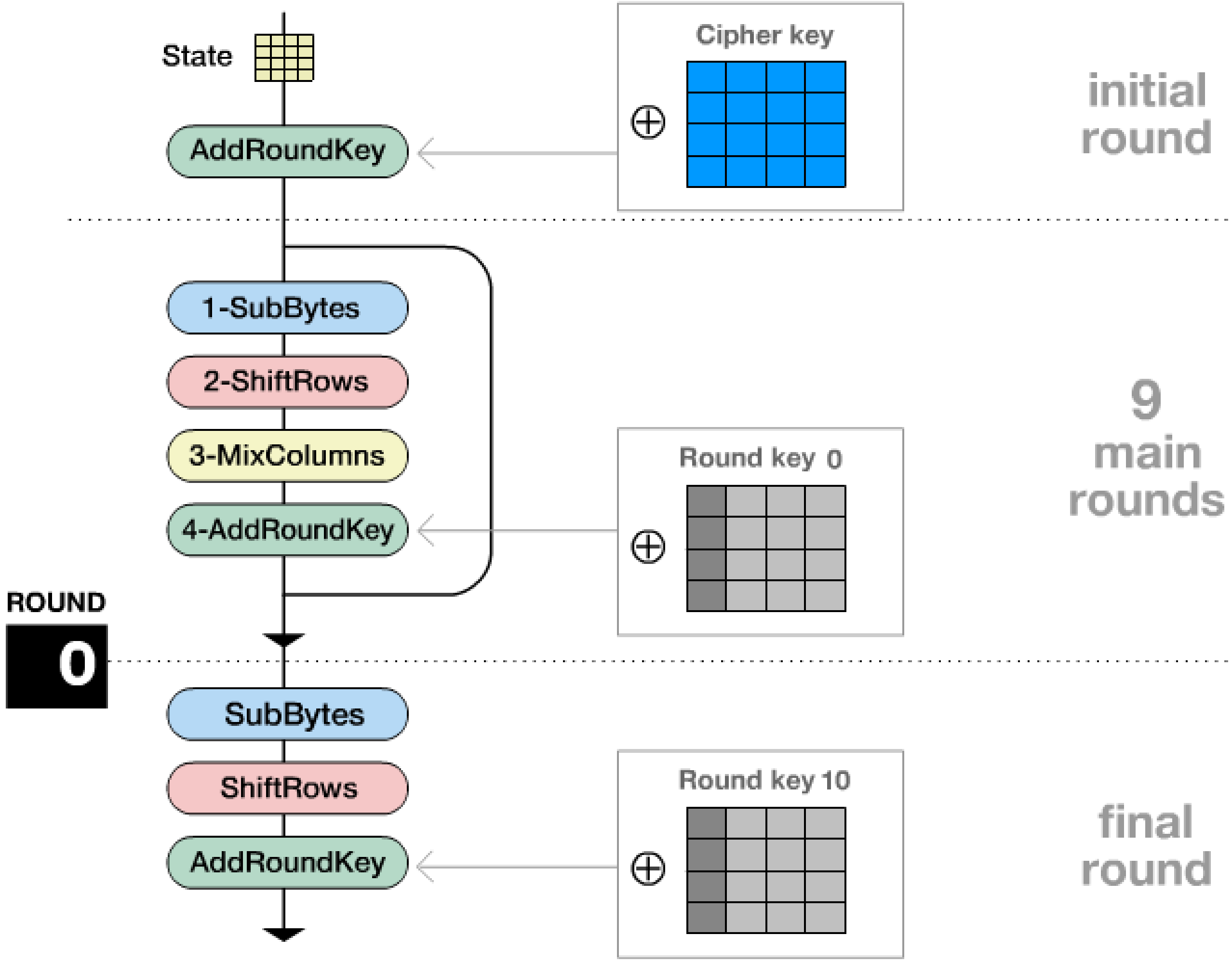
- **KeyExpansion**—round keys are derived from the cipher key using Rijndael's key schedule.
  - http://en.wikipedia.org/wiki/Rijndael_key_schedule
- **Initial Round**
  - AddRoundKey—each byte of the state is combined with the round key using bitwise xor.
- **Rounds**
  - SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
  - ShiftRows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
  - MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
  - AddRoundKey
- **Final Round (no MixColumns)**
  - SubBytes
  - ShiftRows
  - AddRoundKey

*source = wiki

AES-128 schematic



Encryption Process



Transformation

The 4 types of transformations:

1-SubBytes

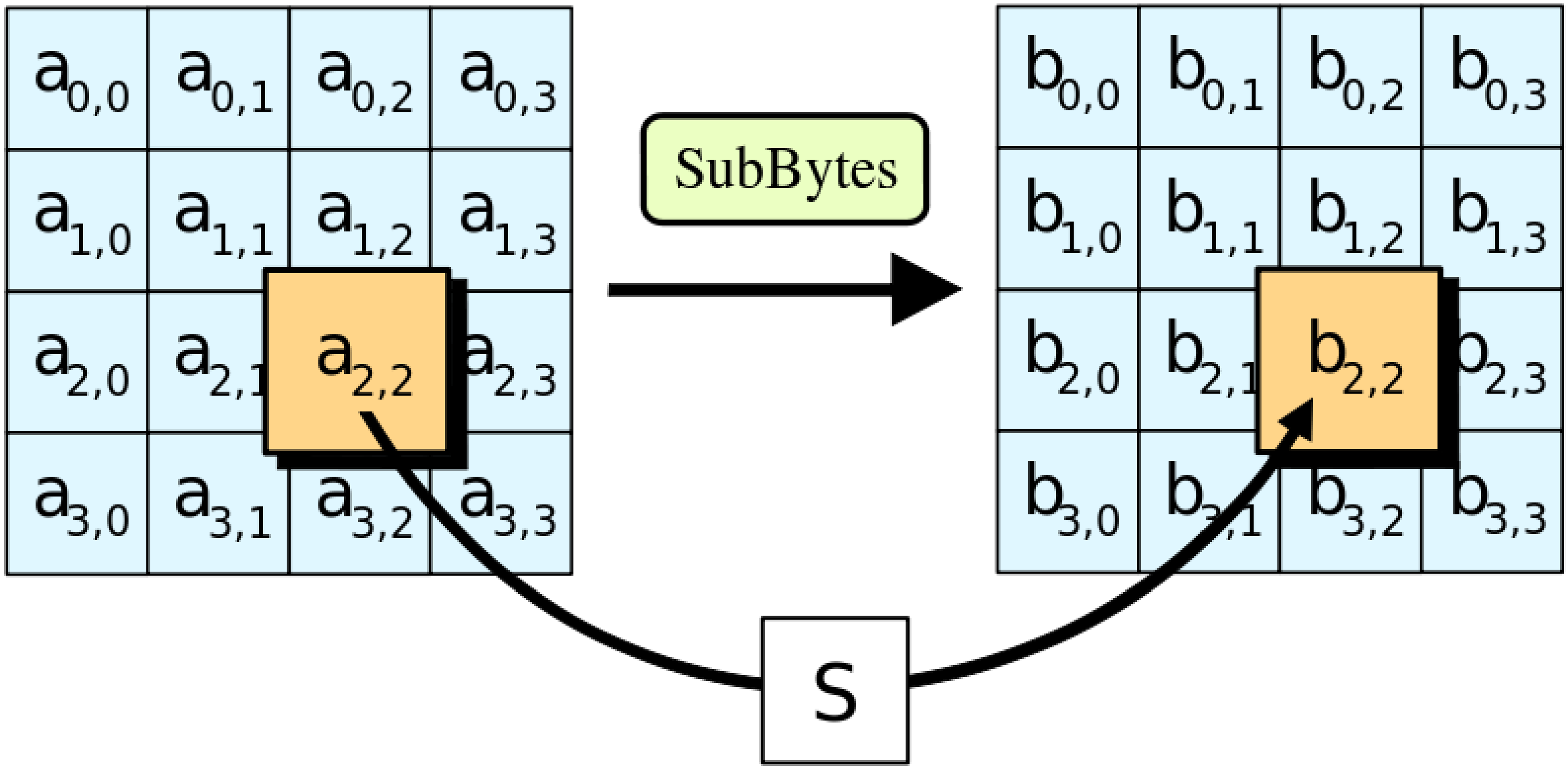
2-ShiftRows

3-MixColumns

4-AddRoundKey

SubBytes

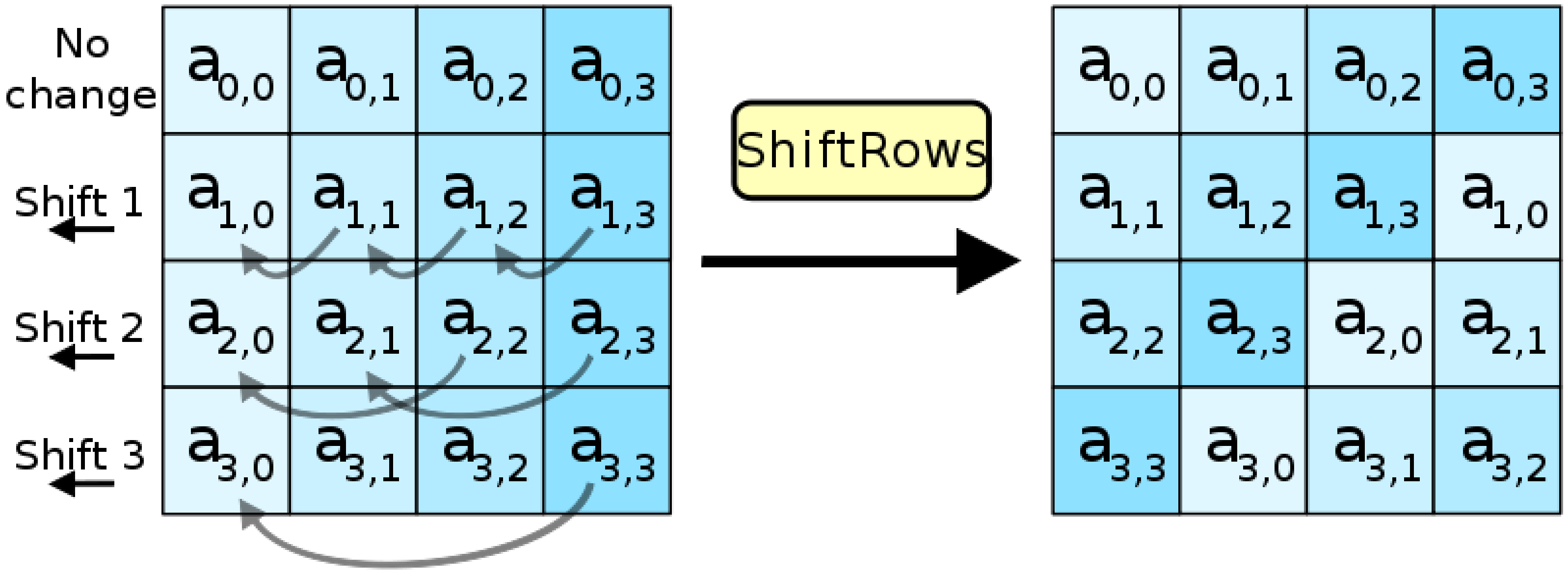
- In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup able, *S*; $b_{ij} = S(a_{ij})$.



*source = wiki

ShiftRows

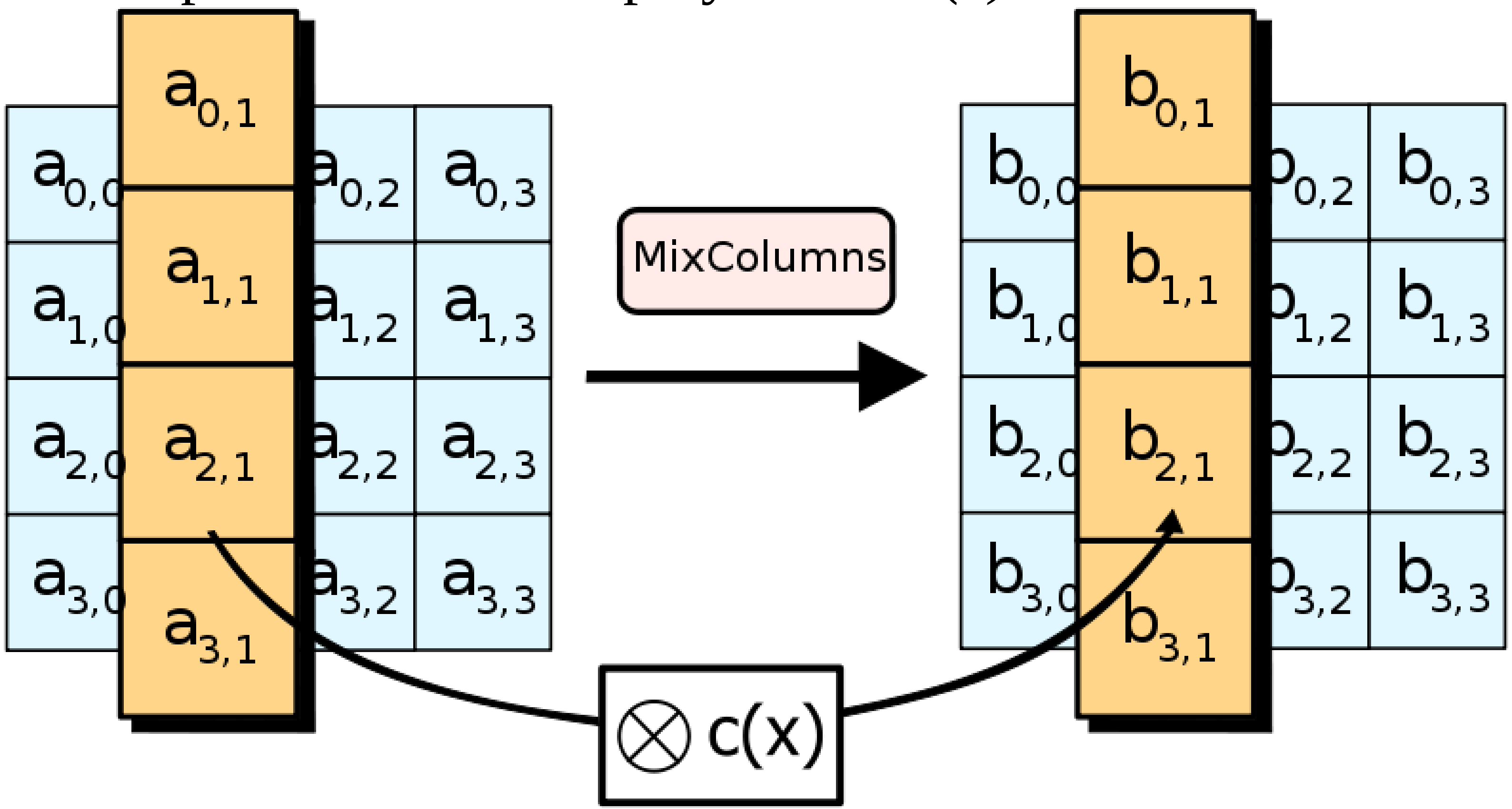
- In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row



*source = wiki

MixColumns

- In the MixColumns step, each column of the state is multiplied with a fixed polynomial *c(x)*.



*source = wiki

MixColumns

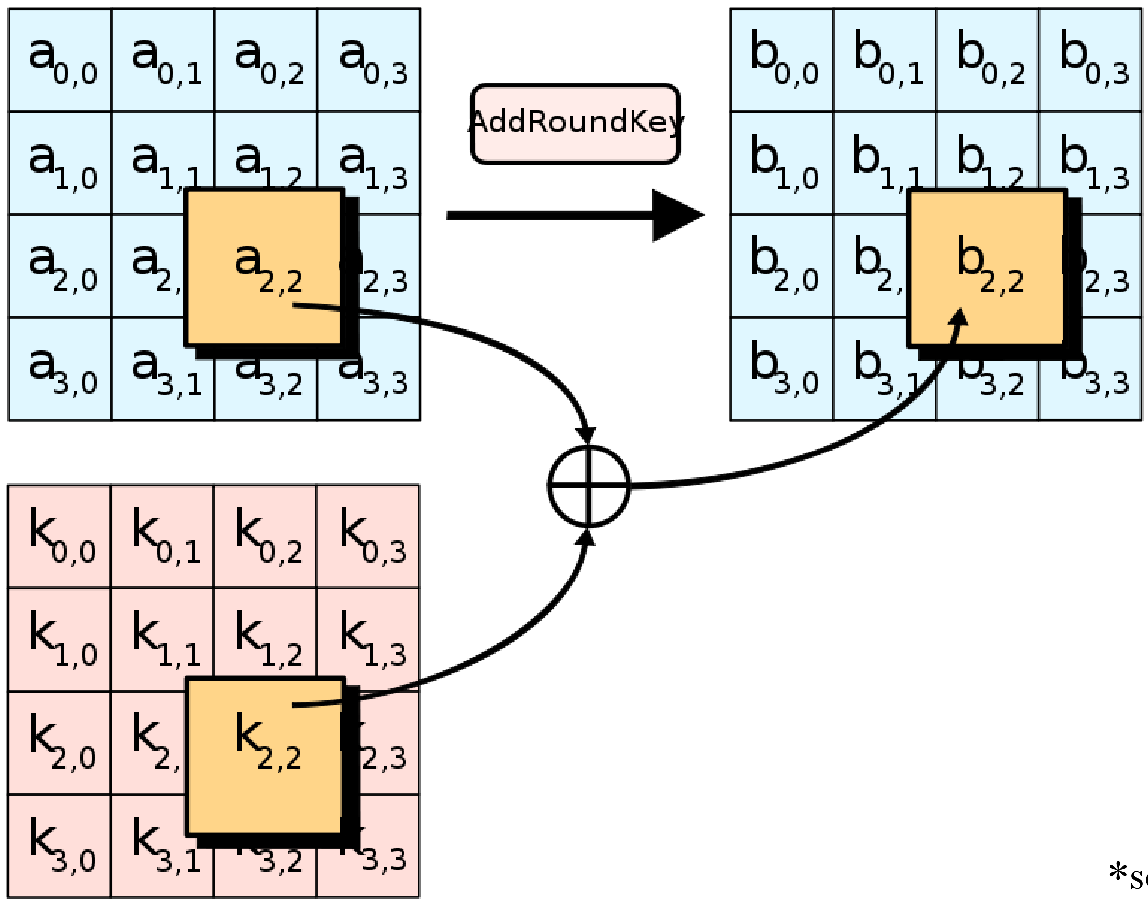
- The multiplication operation is defined as:
  - multiplication by 1 means no change
  - multiplication by 2 means shifting to the left
  - multiplication by 3 means shifting to the left and then performing xor with the initial unshifted value.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

*source = wiki

AddRoundKey

- In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XORoperation (⊕).



*source = wiki

Example (Input)

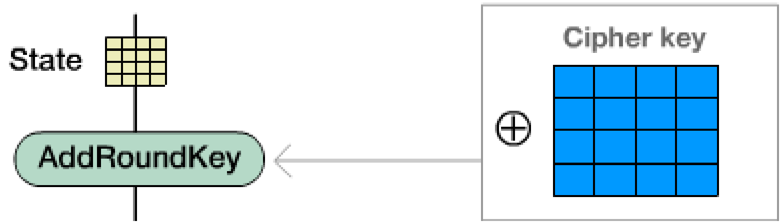
State

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

Example (Initial round)



initial round

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

⊕

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

Example (R1-SubBytes)

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

SubBytes

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Example (R1-ShiftRows)

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

ShiftRows →

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

Example (R1-MixColumns)

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

MixColumns →

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

Example (R1-AddRoundKey)

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

AddRoundKey

$\oplus$

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

$\downarrow$

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

	Start of round	After SubBytes	After ShiftRows	After MixColumns		Round key	
Input	32 88 31 e0					2b 28 ab 09	
	43 5a 31 37					7e ae f7 cf	
	f6 30 98 07				⊕	15 d2 15 4f	=
	a8 8d a2 34					16 a6 88 3c	
Round 1	19 a0 9a e9	d4 e0 b8 1e	d4 e0 b8 1e	04 e0 48 28		a0 88 23 2a	
	3d f4 c6 f8	27 bf b4 41	bf b4 41 27	66 cb f8 06		fa 54 a3 6c	
	e3 e2 8d 48	11 98 5d 52	5d 52 11 98	81 19 d3 26	⊕	fe 2c 39 76	=
	be 2b 2a 08	ae f1 e5 30	30 ae f1 e5	e5 9a 7a 4c		17 b1 39 05	
Round 2	a4 68 6b 02	49 45 7f 77	49 45 7f 77	58 1b db 1b		f2 7a 59 73	
	9c 9f 5b 6a	de db 39 02	db 39 02 de	4d 4b e7 6b		c2 96 35 59	
	7f 35 ea 50	d2 96 87 53	87 53 d2 96	ca 5a ca b0	⊕	95 b9 80 f6	=
	f2 2b 43 49	89 f1 1a 3b	3b 89 f1 1a	f1 ac a8 e5		f2 43 7a 7f	
Round 3	aa 61 82 68	ac ef 13 45	ac ef 13 45	75 20 53 bb		3d 47 1e 6d	
	8f dd d2 32	73 c1 b5 23	c1 b5 23 73	ec 0b c0 25		80 16 23 7a	
	5f e3 4a 46	cf 11 d6 5a	d6 5a cf 11	09 63 cf d0	⊕	47 fe 7e 88	=
	03 ef d2 9a	7b df b5 b8	b8 7b df b5	93 33 7c dc		7d 3e 44 3b	
Round 4	48 67 4d d6	52 85 e3 f6	52 85 e3 f6	0f 60 6f 5e		ef a8 b6 db	
	6c 1d e3 5f	50 a4 11 cf	a4 11 cf 50	d6 31 c0 b3		44 52 71 0b	
	4e 9d b1 58	2f 5e c8 6a	c8 6a 2f 5e	da 38 10 13	⊕	a5 5b 25 ad	=
	ee 0d 38 e7	28 d7 07 94	94 28 d7 07	a9 bf 6b 01		41 7f 3b 00	
Round 5	e0 c8 d9 85	e1 e8 35 97	e1 e8 35 97	25 bd b6 4c		d4 7c ca 11	
	92 63 b1 b8	4f fb c8 6c	fb c8 6c 4f	d1 11 3a 4c		d1 83 f2 f9	
	7f 63 35 be	d2 fb 96 ae	96 ae d2 fb	a9 d1 33 c0	⊕	c6 9d b8 15	=
	e8 c0 50 01	9b ba 53 7c	7c 9b ba 53	ad 68 8e b0		f8 87 bc bc	

	Start of round	After SubBytes	After ShiftRows	After MixColumns	Round key
Round 6	f1 c1 7c 5d	a1 78 10 4c	a1 78 10 4c	4b 2c 33 37	6d 11 db ca
	00 92 c8 b5	63 4f e8 d5	4f e8 d5 63	86 4a 9d d2	88 0b f9 00
	6f 4c 8b d5	a8 29 3d 03	3d 03 a8 29	8d 89 f4 18	a3 3e 86 93
	55 ef 32 0c	fc df 23 fe	fe fc df 23	6d 80 e8 d8	7a fd 41 fd
Round 7	26 3d e8 fd	f7 27 9b 54	f7 27 9b 54	14 46 27 34	4e 5f 84 4e
	0e 41 64 d2	ab 83 43 b5	83 43 b5 ab	15 16 46 2a	54 5f a6 a6
	2e b7 72 8b	31 a9 40 3d	40 3d 31 a9	b5 15 56 d8	f7 c9 4f dc
	17 7d a9 25	f0 ff d3 3f	3f f0 ff d3	bf ec d7 43	0e f3 b2 4f
Round 8	5a 19 a3 7a	be d4 0a da	be d4 0a da	00 b1 54 fa	ea b5 31 7f
	41 49 e0 8c	83 3b e1 64	3b e1 64 83	51 c8 76 1b	d2 8d 2b 8d
	42 dc 19 04	2c 86 d4 f2	d4 f2 2c 86	2f 89 6d 99	73 ba f5 29
	b1 1f 65 0c	c8 c0 4d fe	fe c8 c0 4d	d1 ff cd ea	21 d2 60 2f
Round 9	ea 04 65 85	87 f2 4d 97	87 f2 4d 97	47 40 a3 4c	ac 19 28 57
	83 45 5d 96	ec 6e 4c 90	6e 4c 90 ec	37 d4 70 9f	77 fa d1 5c
	5c 33 98 b0	4a c3 46 e7	46 e7 4a c3	94 e4 3a 42	66 dc 29 00
	f0 2d ad c5	8c d8 95 a6	a6 8c d8 95	ed a5 a6 bc	f3 21 41 6e
Round 10	eb 59 8b 1b	e9 cb 3d af	e9 cb 3d af		d0 c9 e1 b6
	40 2e a1 c3	09 31 32 2e	31 32 2e 09		14 ee 3f 63
	f2 38 13 42	89 07 7d 2c	7d 2c 89 07		f9 25 0c 0c
	1e 84 e7 d2	72 5f 94 b5	b5 72 5f 94		a8 89 c8 a6
Output	39 02 dc 19				
	25 dc 11 6a				
	84 09 85 0b				
	1d fb 97 32				

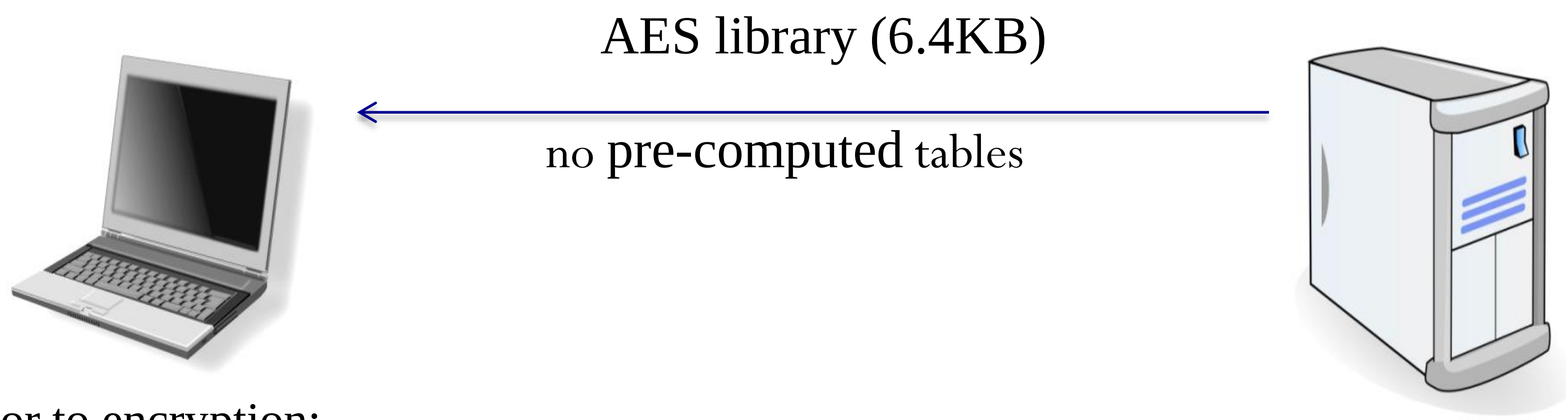
Ciphertext

Code size/performance tradeoff

	Code size	Performance
Pre-compute round functions (24KB or 4KB)	largest	fastest: table lookups and xors
Pre-compute S-box only (256 bytes)	smaller	slower
No pre-computation	smallest	slowest

Example: Javascript AES

AES in the browser:



Prior to encryption:
pre-compute tables

Then encrypt using tables

http://crypto.stanford.edu/sjcl/

AES in hardware

AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES

  128-bit registers: xmm1=state, xmm2=round key

  aesenc xmm1, xmm2 ; puts result in xmm1
- **aeskeygenassist**: performs AES key expansion
- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

Attacks

Best key recovery attack:

- four times better than ex. search [BKR'11]

Related key attack on AES-256: [BK'09]

- Given 2^{99} inp/out pairs from **four related keys** in AES-256 can recover keys in time $\approx 2^{99}$

Performance:

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109

One-time and Many-time keys

One-time keys

1. Adversary's power:
   Adv sees only one ciphertext (one-time key)

2. Adversary's goal:
   Learn info about PT from CT (semantic security)

Many-time keys

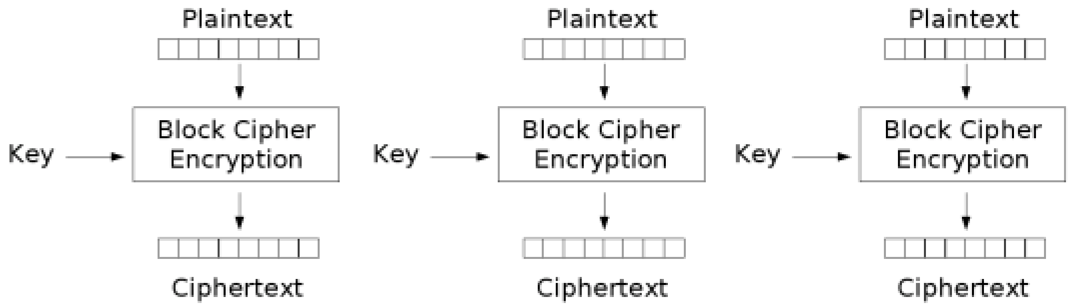
1. Adversary's power:
   Adv have access to plaintext and its corresponding ciphertext

2. Adversary's goal:
   Learn info about PT and Key from CT/PT pair (semantic security against chosen/known Plaintext attacks)

Modes of Encryption

Electronic Codebook (ECB) mode

- Simplest method: divide a message into blocks and encrypt each one
- Advantage: fast access to single blocks

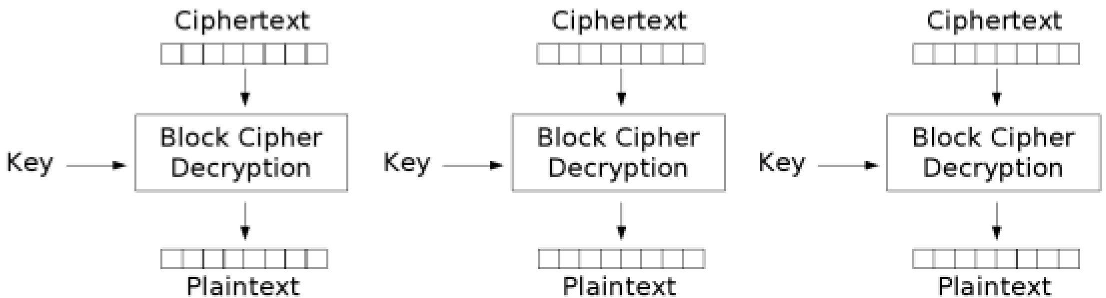


Electronic Codebook (ECB) mode encryption

*image source = wiki (also for several following images)

Electronic Codebook (ECB) mode

- Does not hide data patterns well and doesn't provide serious message confidentiality
- Not recommended for use in cryptographic protocols at all.



Electronic Codebook (ECB) mode decryption

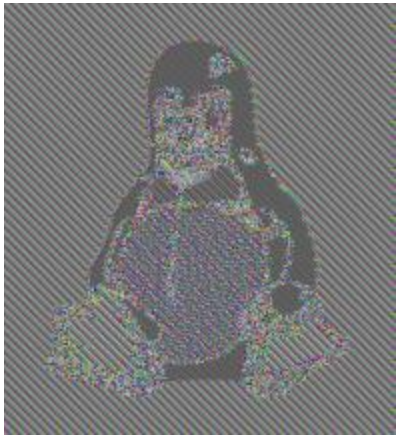
- Make protocols without integrity protection more susceptible to replay attacks
  - Each block gets decrypted in exactly the same way.

*image source = wiki (also for several following images)

Electronic Codebook (ECB) mode

- Disadvantage: Too simple, too dangerous; does not satisfy the requirements
  - Identical blocks are encrypted to the same cipher block and can be identified by an attacker i.e. if $m_1=m_2$ then $c_1=c_2$
  - The message structure can be identified
  - If the attacker knows, what context the plaintext has, parts of message can be manipulated
  - ECB is not semantically secure for messages that contain more than one block.

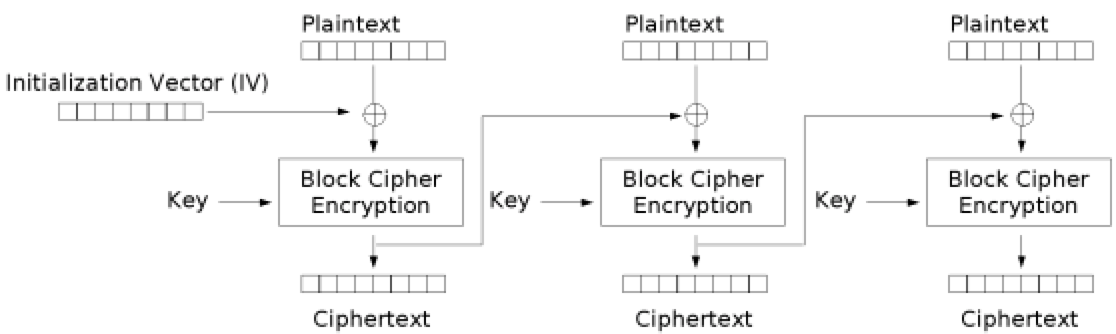
Original Image



Encrypted using ECB mode

Cipher Block Chaining (CBC) mode

- Invented by IBM in 1976
- Each block of plaintext is XORed with the previous ciphertext block before being encrypted
- Each ciphertext block depends on all plaintext blocks processed up to that point
- For Uniqueness, an initialization vector (IV) must be used in the first block.

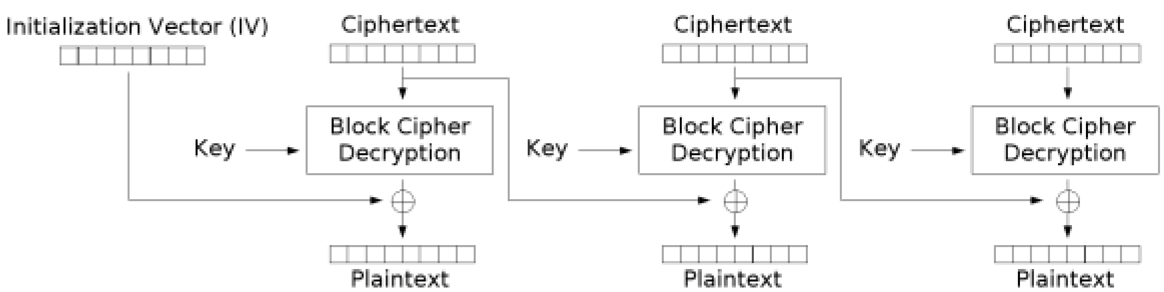


Cipher Block Chaining (CBC) mode encryption

Mathematical Formula $C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$

Cipher Block Chaining (CBC) mode

- For each message to be encoded, a new IV should be used
- Usage of the same IV for all messages would cause some problems:
  - Differences in similar messages can be found by an attacker
  - Old messages can be sent by an attacker at a later time
  - Chosen plaintext can be applied as an attack



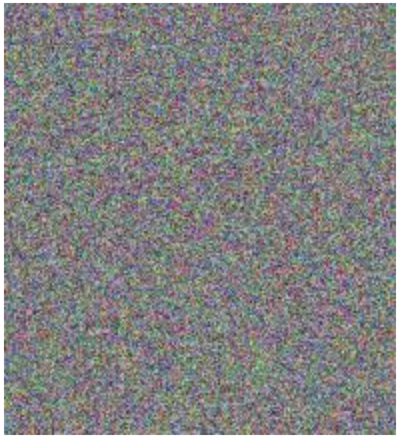
Cipher Block Chaining (CBC) mode decryption

Mathematical Formula $P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV.$

Cipher Block Chaining (CBC) mode

- Drawback(s)
  - Encryption is sequential (cannot be parallelized).
  - A one-bit change in a plaintext or IV affects all following ciphertext blocks.
- For Decryption Incorrect IV causes only the first block of plaintext to be corrupt
  - Plaintext block can be recovered from two adjacent blocks of ciphertext
  - Decryption *cannot* be parallelized
  - one-bit change to the ciphertext causes complete corruption of the corresponding block of plaintext
  - rest of the blocks remain intact.

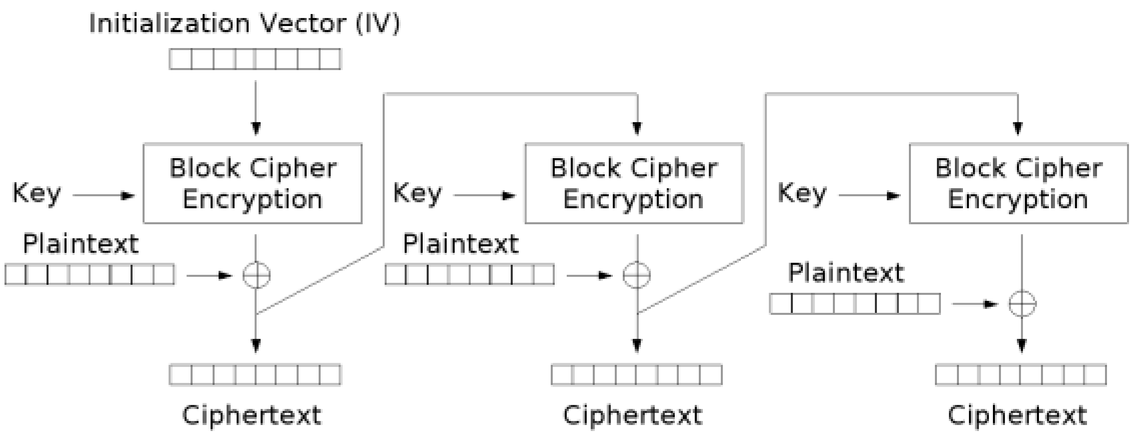
Original Image



Encrypted using CBC mode

Cipher Feedback (CFB) mode

- Close relative of CBC, makes a block cipher into a self-synchronizing stream cipher



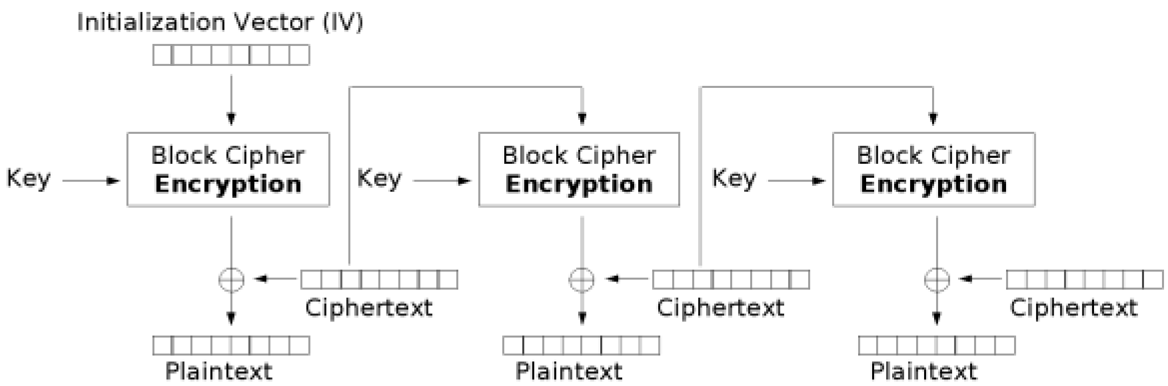
Cipher Feedback (CFB) mode encryption

Mathematical Formula

$$C_i = E_K(C_{i-1}) \oplus P_i$$
$$P_i = E_K(C_{i-1}) \oplus C_i$$
$$C_0 = \mathrm{IV}$$

Cipher Feedback (CFB) mode

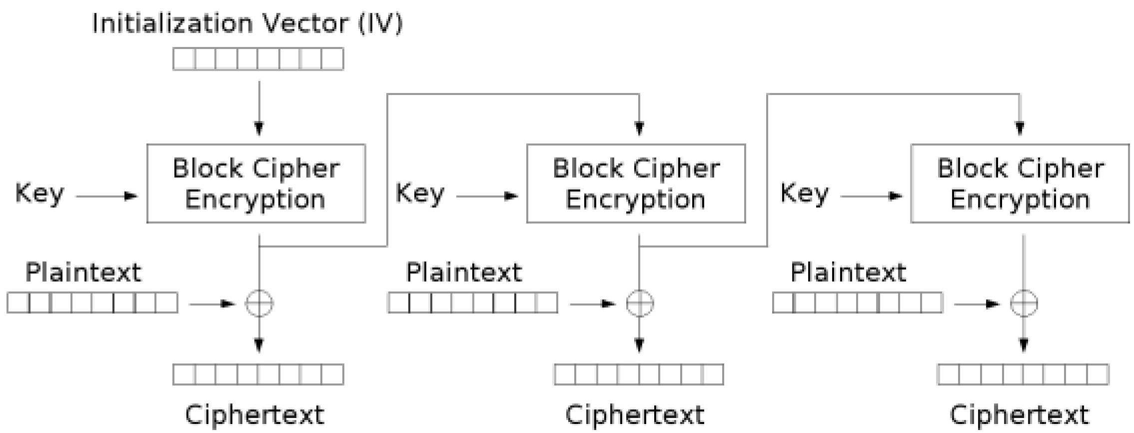
- CFB decryption is almost identical to CBC encryption performed in reverse



Cipher Feedback (CFB) mode decryption

Output Feedback (OFB) mode

- Makes a block cipher into a synchronous stream cipher
- Just as with other stream ciphers, flipping a bit in the ciphertext produces a flipped bit in the plaintext at the same location



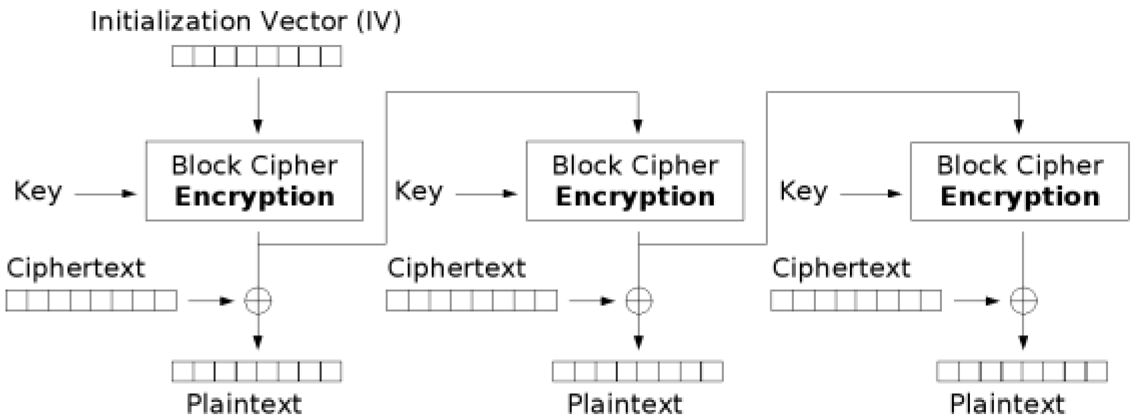
Output Feedback (OFB) mode encryption

Mathematical Formula

$$
\begin{aligned}
C_j &= P_j \oplus O_j \\
P_j &= C_j \oplus O_j \\
O_j &= E_K(I_j) \\
I_j &= O_{j-1} \\
I_0 &= \mathrm{IV}
\end{aligned}
$$

Output Feedback (OFB) mode

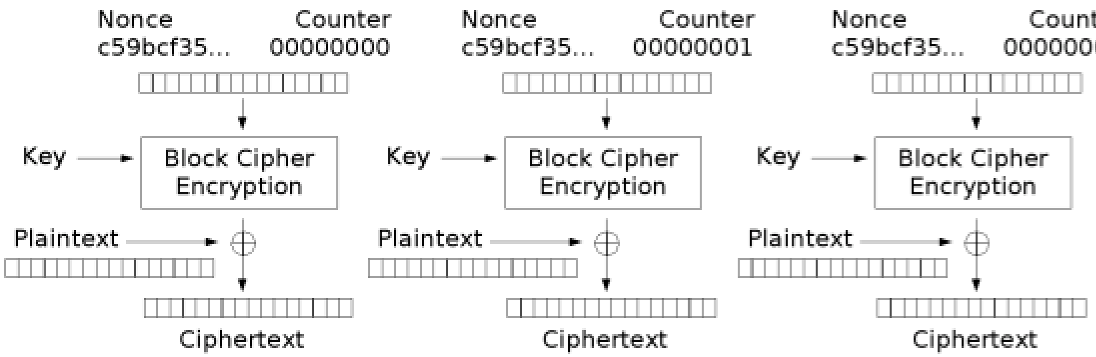
- Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:



Output Feedback (OFB) mode decryption

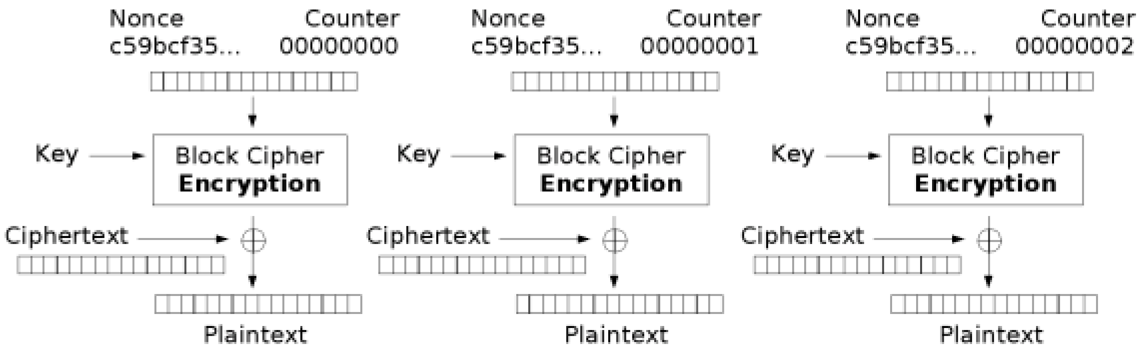
Counter Mode (CTR) mode

- Like OFB, counter mode turns a block cipher into a stream cipher
- The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time
- CTR mode is widely accepted and CBC, CTR modes are recommended by Niels Ferguson and Bruce Schneier
- CTR mode is well suited to operate on a multi-processor machine where blocks can be encrypted in parallel



Counter (CTR) mode encryption

Counter Mode (CTR) mode



Counter (CTR) mode decryption

Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Dan Boneh (Stanford)
- Prof. O. Spaniol (RWTH Aachen)