1495 UNIVERSITY OF ABERDEEN | CELEBRATING 525 YEARS 1495 – 2020

ABERDEEN 2040

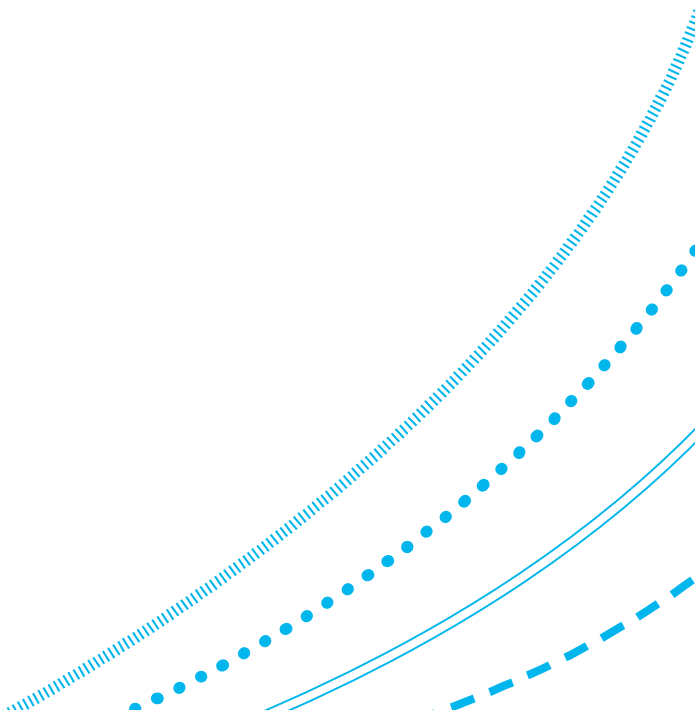
Network Security Technology

Symmetric cryptography

September 2025

Outline of lecture

1. Numbers revisited.
2. Block ciphers.
3. Feistel.
4. Data Encryption Standard (DES).



Binary and hexadecimal numbers

1. Both based on powers of 2.
2. Hex is often used as a shorthand for binary.
3. 4 bits represent 1 hex character: 0000 to 1111 correspond to 16 hex characters 0 to *F.*

Hexadecimal to/from binary

1. A conversion table for hexadecimal numbers.

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

Text as bits

1. When we think of cryptography:
    1. Confidentiality.
    2. Concealing messages.
    3. first thought is “text”.
2. Information is stored digitally - just 0s and 1s, so how is text represented?
    1. ASCII codes.
    2. 2 Hex characters used to represent each of 256 (FF).
    3. For example:
        1. lowercase ’a’ is 61_{16} or 01100001_2 in binary.
        2. uppercase ’A’ is 41_{16} or 01000001_2 in binary.

Text as bits (cont'd)

1. 0 to 255 decimal or 0 to FF hexadecimal.
2. 8 bits per ASCII character.
3. ASCII table readily available on internet.

ASCII table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

An example: ASCII

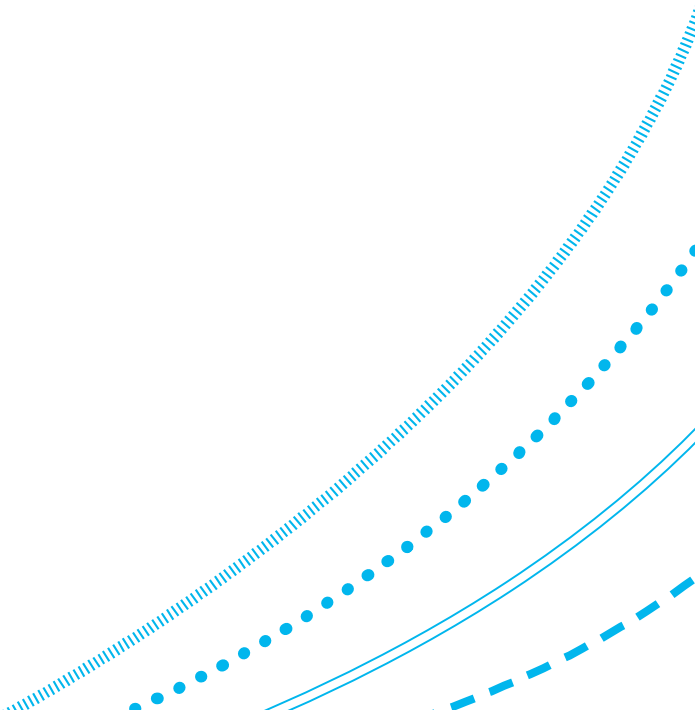
1. What would be the ASCII representation of “Maths”?
2. Use an ASCII table and find the hex pair for each character.
3. Look up each character, taking care with lower and uppercase

   Answer 4D 61 74 68 73
4. Process is easily reversible
5. What characters are represented by 43 72 79 70 74 6*F*

   Answer Crypto

Block ciphers

1. A block cipher is a symmetric-key cipher that breaks up the plaintext message into blocks of a fixed length and encrypts one block at a time.
2. Break the plaintext P into successive blocks:
    1. $P_1, P_2, \ldots$ of size n bits each
3. Encrypt each P_i with the same key K of size k bits:
    1. $E_K(P) = E_K(P_1)E_K(P_2)\ldots$



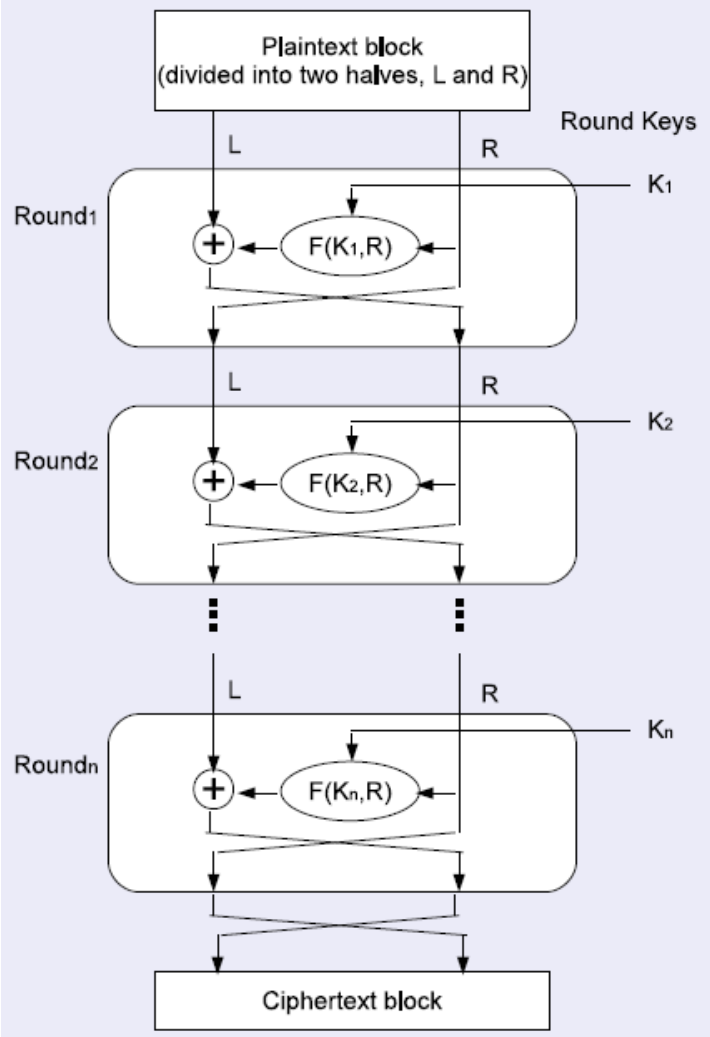
Main types of block ciphers

1. substitution cipher: replace symbols (or groups of symbols) by other symbols (or groups of symbols).
2. transposition cipher: permutes the symbols within the block.
3. product cipher: is a composite of substitution and transposition ciphers.

Feistel structure for block ciphers

1. Named after the IBM cryptographer Horst Feistel and first implemented in the Lucifer cipher by Horst Feistel and Don Coppersmith.
2. A cryptographic system based on Feistel structure uses the same basic algorithm for both encryption and decryption.
3. Consists of multiple rounds of processing of the plaintext, with each round consisting of a “substitution” step followed by a permutation step: combination of P and S-boxes.

Feistel structure



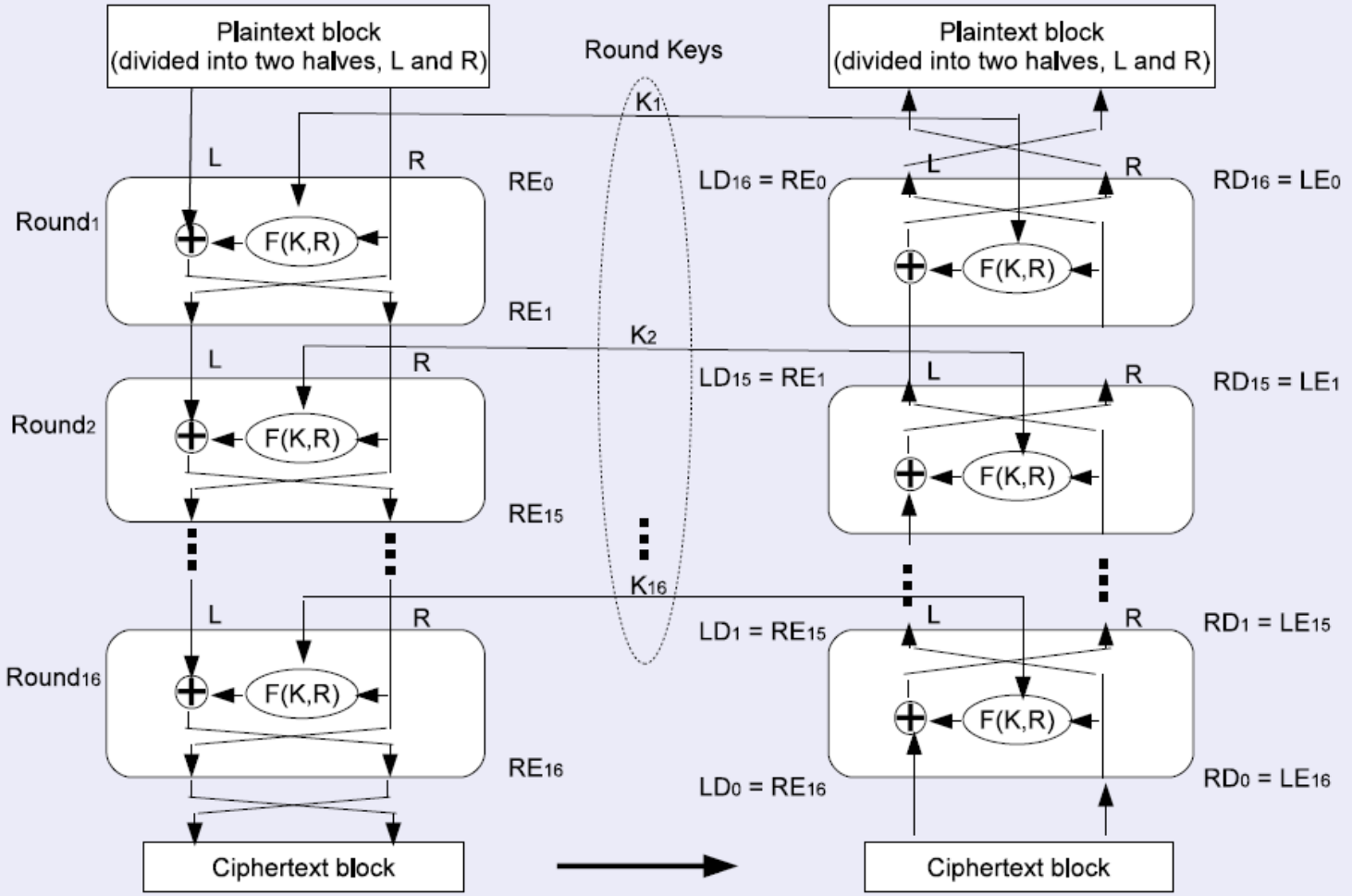
Feistel structure (cont'd)

1. The input block to each round is divided into two halves: denoted by *L* (the left half) and *R* (the right half).
2. In each round,
    1. the right half of the block *R* goes through unchanged.
    2. the left half *L* goes through an operation that depends on *R* and the encryption key.
3. The permutation step at the end of each round: swapping the modified *L* and *R*. Therefore,
    1. the *L* for the next round would be *R* of the current round, and
    2. *R* for the next round be the output *L* of the current round.

Each round in the Feistel structure

1. Let LE_i and RE_i denote the output half-blocks at the end of the i^{th} round of processing. The letter "*E*" denotes encryption.
2. We have:
    1. $LE_i = RE_{i-1}$
    2. $RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$
3. *F* is referred to as the Feistel function. Assuming 16 rounds of processing, the output of the last round of processing is given by:
    1. $LE_{16} = RE_{15}$
    2. $RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$

Decryption in the Feistel structure



Decryption in the Feistel structure (cont'd)

1. The decryption algorithm is exactly the same as the encryption algorithm with the only difference that the round keys are used in the reverse order.
2. The output of each round during decryption is the input to the corresponding round during encryption.

Permutation

1. Every possible arrangement of *n* or lesser items from *n* available items.

2. An example:

   A = {1, 2, 3}, choose two numbers out of three.

   There are 6 ways in which we can do the same. They are:

   {1, 2}, {2, 1}, {1, 3}, {3, 1}, {2, 3}, {3, 2}.

   Note that {1, 2} and {2, 1} are two different permutations. Unlike combinations, where {1, 2} and {2, 1} are the same combination.

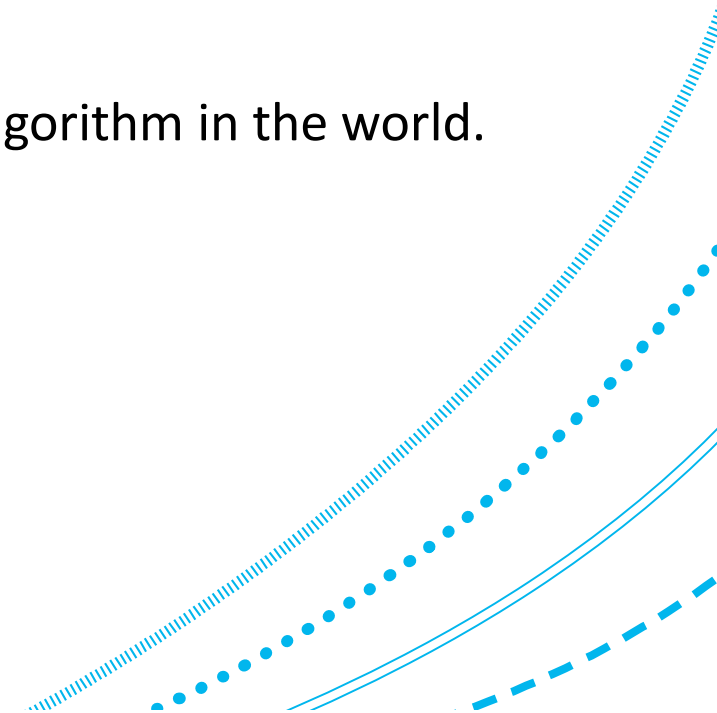
Reference: *https://www.geeksforgeeks.org/permutation/*

Data Encryption Standard

1. 1973-74: NBS (National Bureau of Standards) issued a call for proposals for a standard cryptographic algorithm:
    1. completely specified, efficient and easy to understand;
    2. high level of security;
    3. security on the key, not on the (public) algorithm;
    4. adaptable, economically implementable in electronic devices;
2. 1975: IBM's solution is accepted and published in Federal Register, the key size has been reduced from the original 112-bits to 56-bits.
3. 1976: DES was adopted.
4. 1981: ANSI approved DES as a private-sector standard.
5. 1998: The end of DES.

DES: facts and remarks

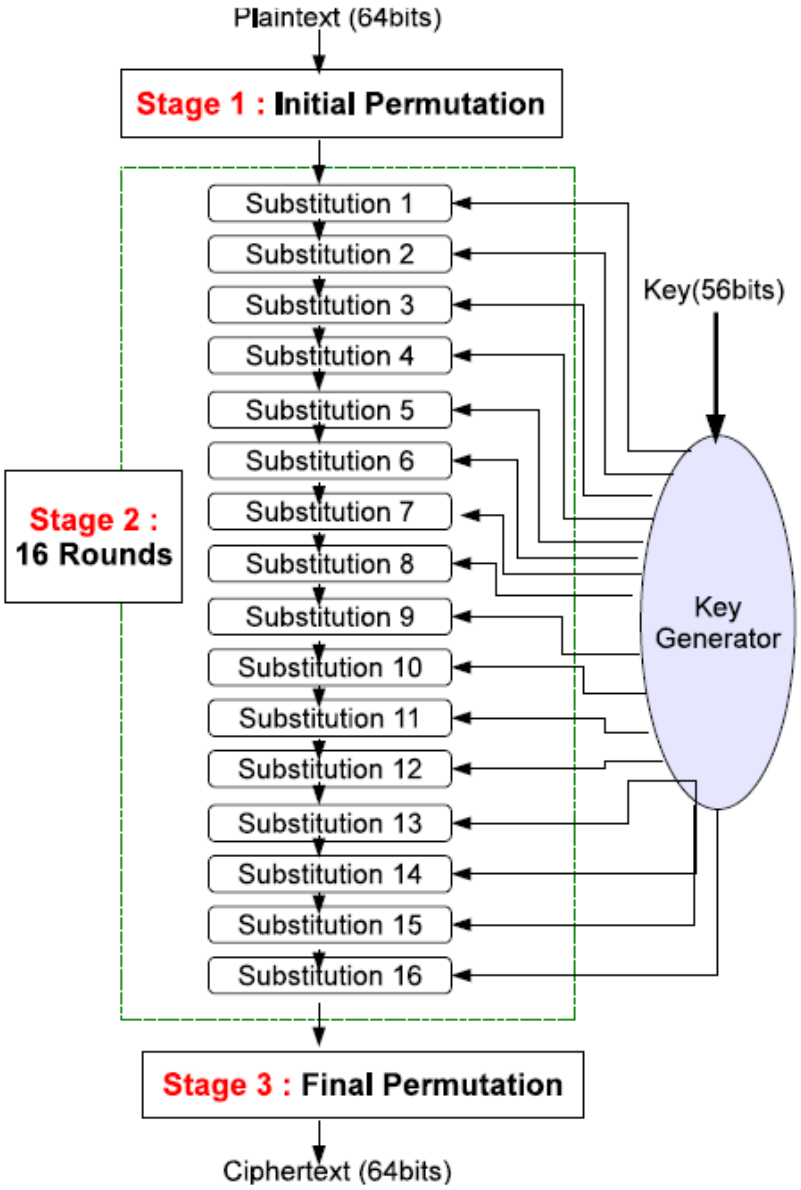
1. Symmetric: encryption and decryption are using the same key.
2. Despite the $2^{56} > 72{,}000{,}000{,}000{,}000{,}000$ possible keys, the Electronic Frontier Foundation built in 1998 a machine called DES Deep Crack that could crack the DES algorithm by brute-force in an average of 4 days.
3. The DES is said to be the most widely used encryption algorithm in the world.



DES: 3-stage/18-step algorithm

1. Initial permutation.
2. 16 operations (rounds): DES uses the Feistel cipher structure with 16 rounds of processing.
3. Final permutation.
4. Sizes:
    1. Plaintext blocks 64 bits.
    2. Ciphertext blocks 64 bits.
    3. Key 56 bits.

DES: the 3-stages



DES: Initial permutation

1. 1st bit of the initial plaintext block is replaced with the 58th bit of the original plaintext block.
2. 2nd bit of the initial plaintext block is replaced with the 50th bit of the original plaintext block.
3. 3rd bit of the initial plaintext block is replaced with the 42nd bit of the original plaintext block.
4. 4th bit of the initial plaintext block is replaced with the 34th bit of the original plaintext block
5. And so on…

Reference: *https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/*

DES: Stage 1

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

$$\text{new } b_1 \leftarrow \text{old } b_{58} \quad \text{new } b_2 \leftarrow \text{old } b_{50}$$
$$\text{new } b_3 \leftarrow \text{old } b_{42} \quad \text{new } b_4 \leftarrow \text{old } b_{34}$$

1. We split the permutated block into two halves:
    1. L_0 : left-most 32 bits, and
    2. R_0 : right-most 32 bits.
2. Remark: if the block is shorter than 64 bits, it should be padded with zeros.

DES key

1. The initial key consists of 64-bits.
2. Before DES starts, it discards 8-bits from the initial 64-bit key, leaving 56-bits.
3. The bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

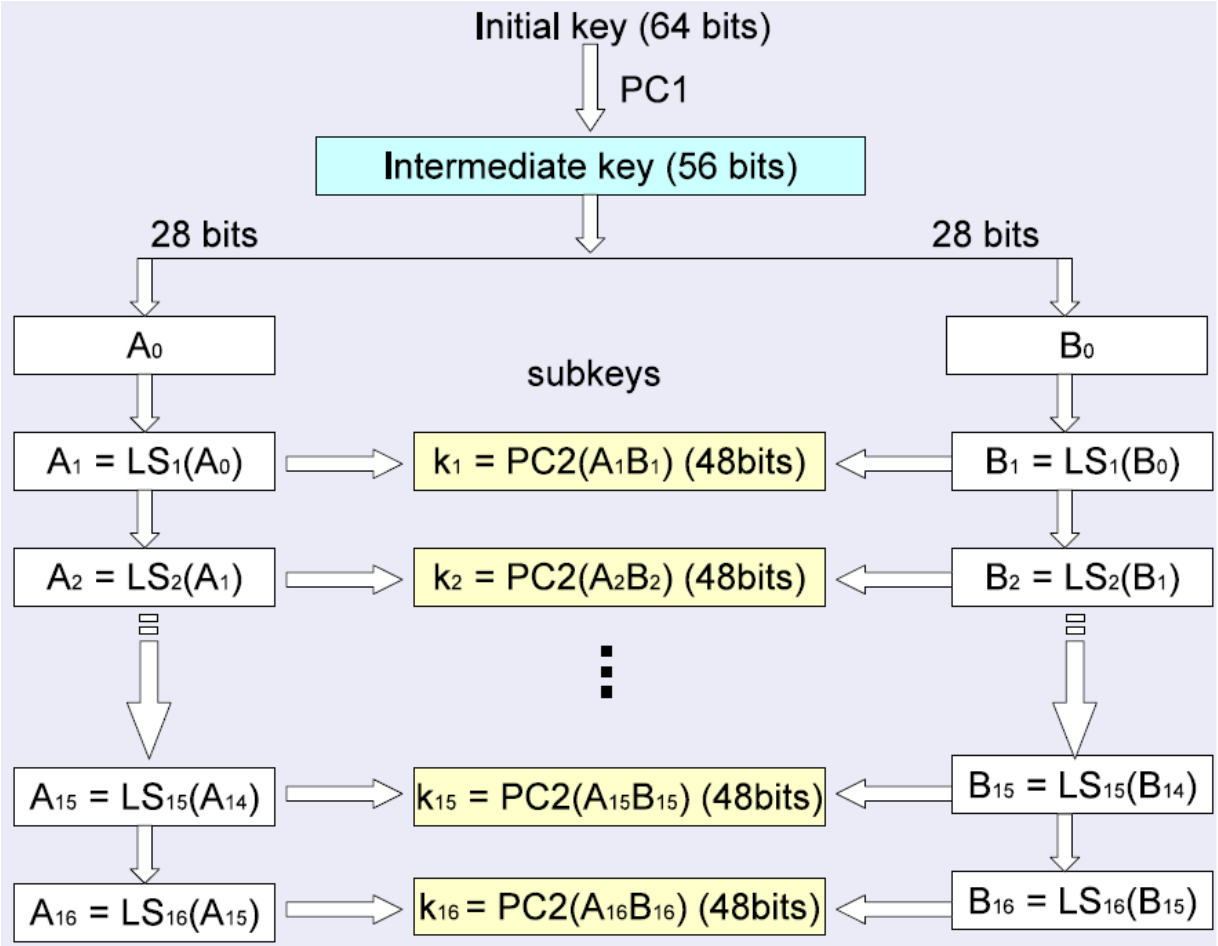
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Reference: *https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/*

DES key

1. For each round, one subkey is needed.
2. This subkey is unique to that round and is essentially a permutation of the initial key that we set at the beginning.
3. A total of 16 subkeys are generated, one for each round.

Subkeys generation



Subkeys generation (cont'd)

1. **Input:** The original 64-bit key k: the key is usually stored as a 64-bit number originally. **Algorithm:**
    1. $k' = PC1(k)$ (56-bit intermediate key).
    2. $A_0 = b_{57}\, b_{49} \ldots b_{36}$ (first half, 28 bits).
    3. $B_0 = b_{63}\, b_{55} \ldots b_4$ (last half, 28 bits).
    4. 16 generation stages:
        1. $A_j = LS_j\,(A_{j-1})$
        2. $B_j = LS_j\,(B_{j-1})$
        3. $k_j = PC2(A_j\, B_j)$
    5. where LS_j : Cyclic shift to the left (left-rotations) by 1 if $j \in \{1, 2, 9, 16\}$ and 2 otherwise.
2. **Output:** A set of 16 keys k_j

Keys generator: PC1

1. The first step to derive our round keys is to permute the key according to the Permuted Choice-1 (PC-1) table.
2. We shall rearrange the elements of our data block using the following PC1 table.

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

$$\text{new } b_1 \leftarrow \text{old } b_{57} \quad \text{new } b_2 \leftarrow \text{old } b_{49}$$
$$\text{new } b_3 \leftarrow \text{old } b_{41} \quad \text{new } b_4 \leftarrow \text{old } b_{33}$$

Keys generator: PC1

1. During the PC-1 permutation, each bit of the original key is rearranged and placed at a new position in accordance with the PC-1 table.

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

$$\text{new } b_1 \leftarrow \text{old } b_{57} \quad \text{new } b_2 \leftarrow \text{old } b_{49}$$
$$\text{new } b_3 \leftarrow \text{old } b_{41} \quad \text{new } b_4 \leftarrow \text{old } b_{33}$$

Keys generator: PC1

1. Every eighth bit (8, 16, 24, 32, 40, 48, 56) is specified for use as parity bits to ensure that the key has been received correctly.

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

$$\text{new } b_1 \leftarrow \text{old } b_{57} \quad \text{new } b_2 \leftarrow \text{old } b_{49}$$
$$\text{new } b_3 \leftarrow \text{old } b_{41} \quad \text{new } b_4 \leftarrow \text{old } b_{33}$$

2. Our 56-bit key is now split into left and right halves, each 28 bits in length.
3. We will now be using this for our next step.

Keys generator: Shifting the key

1. The next step is shifting the key to the left either by 1 bit or 2 bits depending on the round number.
    1. Round number 1, 2, 9 and 16: left shift by 1.
    2. Other round numbers: left shift by 2.
2. For example, if we're in the 5th encryption round and had 1101101101101101 as our key, then we'll have to left shift it by 2 spaces. Hence after this round, our key will now be: 0110110110110111.
3. Hence, sixteen different subkeys are generated, with one for each round of the DES process.

Keys generator: PC2

1. Next, we permute the key according to the Permuted Choice-2 table as given.
2. Here, we once again rearrange the bits in our key according to the table below.

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

$$\text{new } b_1 \leftarrow \text{old } b_{14} \quad \text{new } b_2 \leftarrow \text{old } b_{17}$$
$$\text{new } b_3 \leftarrow \text{old } b_{11} \quad \text{new } b_4 \leftarrow \text{old } b_{24}$$

3. Note that there is no 9, 18, 22, 25, 35, 38, 43, 54.

Keys generator: PC2

1. Permutation + Compression.
2. Permutation Choice 2: 56 → 48.

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

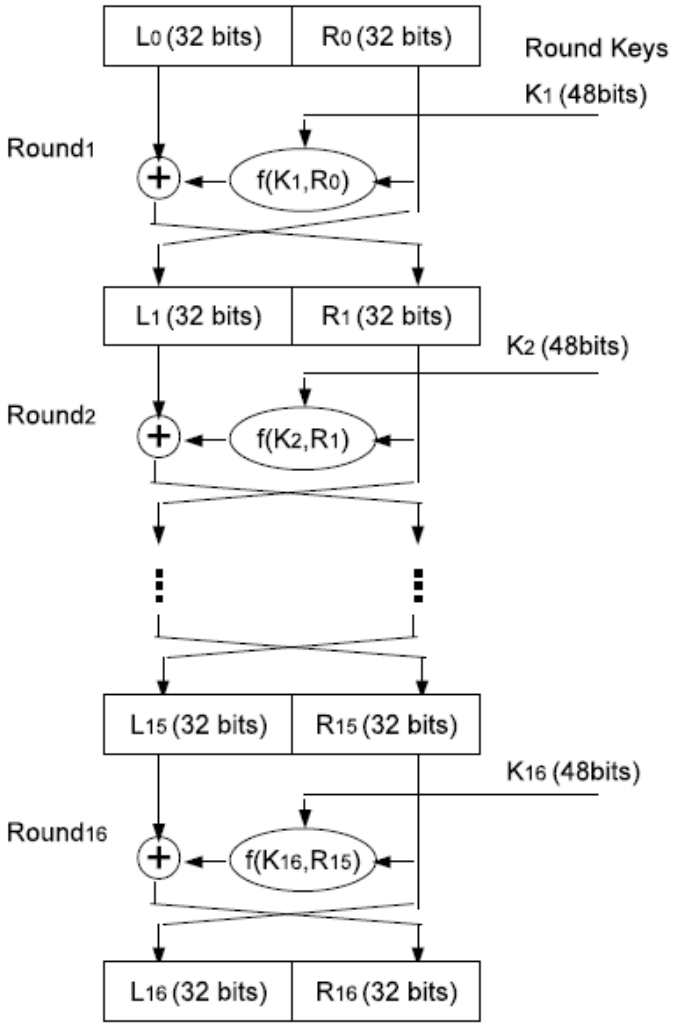
$$\text{new } b_1 \leftarrow \text{old } b_{14} \quad \text{new } b_2 \leftarrow \text{old } b_{17}$$
$$\text{new } b_3 \leftarrow \text{old } b_{11} \quad \text{new } b_4 \leftarrow \text{old } b_{24}$$

3. We have derived 16 different subkeys for each of the DES rounds.

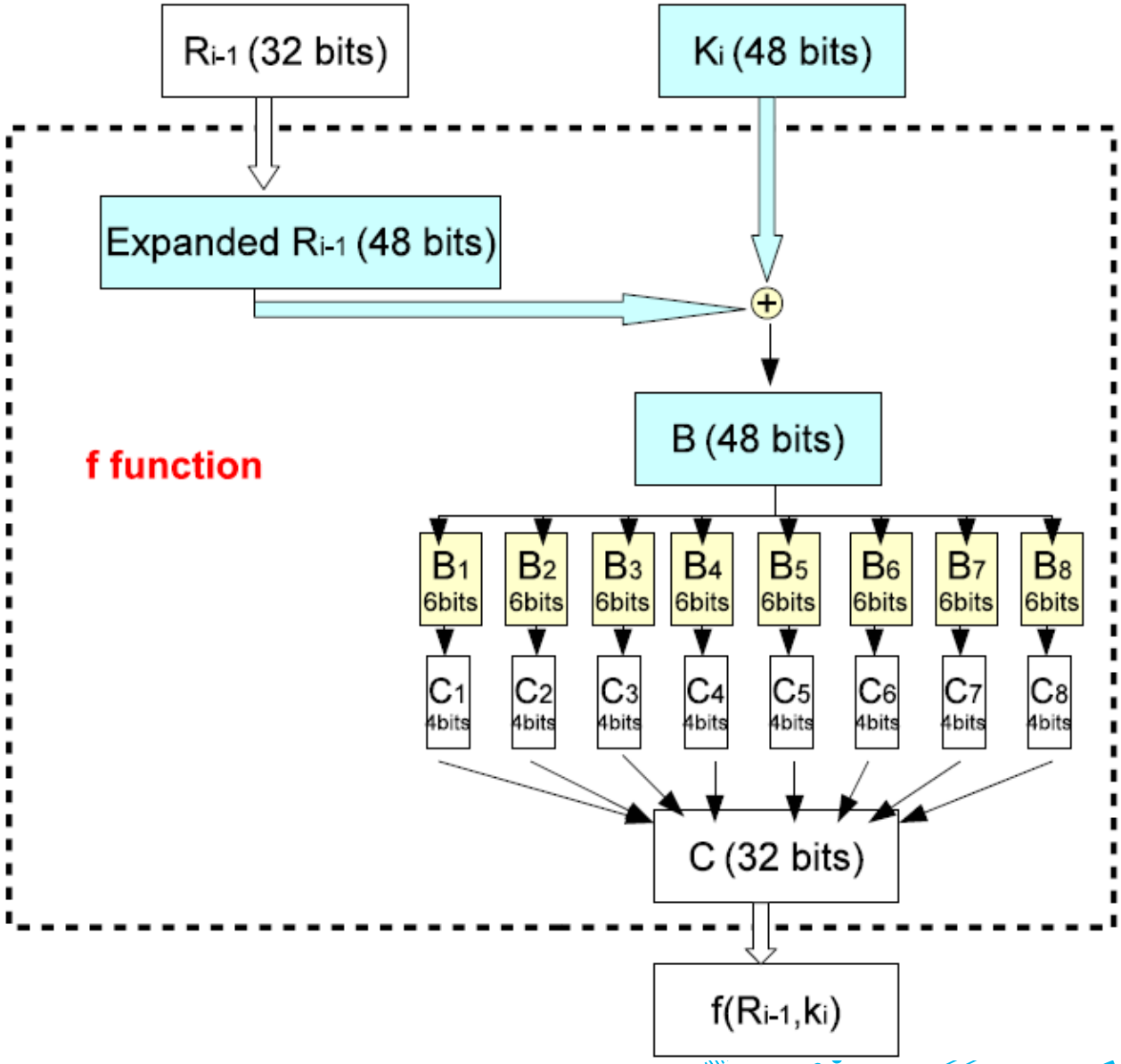
DES: stage 2 encryption process

1. L_iR_i is computed as follows:
    1. $L_i = R_{i-1}$
    2. $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$

    where $\oplus$ denotes the XOR logical function, *f* is the specific inner function of *DES*, and k_1, $k_2, \ldots, k_{16}$ are the subkeys the key generator produced.



DES: the *f* function also called an expansion function

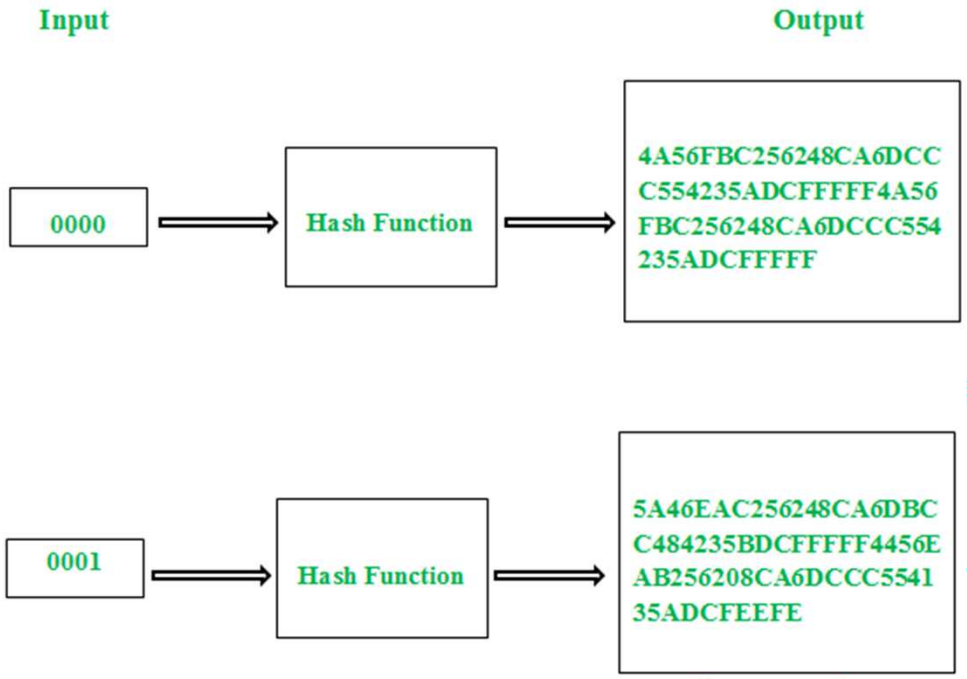


DES: the *f* function (cont'd)

1. **Input**: 2 arguments (R_{i-1}, k_i) of size 32 and 48 ...;
2. **Algorithm**:
    1. Expand R_{i-1} to a 48-bit block using a given expansion function;
    2. Compute $R_{i-1} \oplus k_i$;
    3. Decompose the result into eight 6-bit blocks $B_1B_2B_3B_4B_5B_6B_7B_8$;
    4. Replace independently each 6-bit block B_i by a 4-bit block C_i ;
    5. Assemble the eight 4-bit blocks to obtain a 32-bit block $C_1C_2C_3C_4C_5C_6C_7C_8$;
    6. Permute C using a given permutation function.
3. **Output**: a 32-bit block;

DES *f* function: expansion of R_{i-1}: 32 bits → 48 bits

- The *f* function achieves the following accomplishments
  - Avalanche effect.
  - Makes the output longer than the input.
- Avalanche effect:
  - A slight change in either the key or the plain text should result in a significant change in the cipher text.
  - This helps keep the data secure because it makes it hard for anyone to figure out the original message or key.



DES *f* function: expansion of R_{i-1}: 32 bits → 48 bits

- Next, the blocks are permuted once again using the following expansion function table:

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

$$\text{new } b_1 \leftarrow \text{old } b_{32} \quad \text{new } b_2 \leftarrow \text{old } b_1$$
$$\text{new } b_3 \leftarrow \text{old } b_2 \quad \text{new } b_4 \leftarrow \text{old } b_3$$

- We can notice that a few blocks are repeated in the table.
- This is to expand the block from 32 bits to 48 bits.

DES *f* function: Substitution ($B_i \rightarrow C_i$): the S-box

1. In general, an S-Box takes some number of input bits (*m*) and transforms them into some number of output bits (*n*).
2. Substitution is used to make the data more complex so that it cannot be deciphered easily.
3. 8 pre-made tables called substitution boxes or S-Boxes are used to transform each 6-bit input into a 4-bit output.

DES *f* function: Substitution ($B_i \rightarrow C_i$): the S-box

1. From the 6-bit input, the Most Significant Bit (MSB) and Least Significant Bit (LSB) is taken and converted to decimal number X. This number X gives us the row number of the S-box.
2. Next, take the 4 bits in the middle of the input and convert that into decimal number Y. This number Y gives us the column number to look at.

DES *f* function: Substitution ($B_i \rightarrow C_i$): the S-box

- For example, let's assume that the 6-bit output from the previous step is B = $(101111)_2$. We'll represent the bits with $(b_1b_2b_3b_4b_5b_6)_2$
- To find the row number, we first take and combine the MSB and LSB (b_1b_6) which gives $(11)_2$. We then convert this to decimal, which gives us $(3)_{10}$, meaning row 3.
- For the column number, we take $b_2b_3b_4b_5$, which is $(0111)_2$. On converting to decimal, we get $(7)_{10}$, or column 7.
- Then, we obtain the number from row 3 column 7.

DES *f* function: an S_1 substitution example: 4 × 6-bit S-Box

1. The 8 S-boxes of size 4 × 6 : $B_i \rightarrow C_i$.
2. Substitution: B_j is a 6-bit block → C_j is a 4-bit block
    1. Given a 6-bit input, the 4-bit output is found by selecting the row using the outer two bits, and the column using the inner four bits. e.g. B_1 = 101010, row = b_1b_6 = $(10)_2$ = 2; column = $b_2b_3b_4b_5$ = $(0101)_2$ = 5 ←→ C_1 = 6 = $(0110)_2$

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

DES *f* function: Substitution ($B_i \rightarrow C_i$): the S-box

- Which S-box should we choose?
- We choose a S-box based on the location of the bits.
- For example, if the 6-bits 101111 are from the beginning of the output from step 2 , i.e., Compute $R_{i-1} \oplus k_i$ in the Algorithm, then we take the first S-box.

1. Algorithm:

1. Expand R_{i-1} to a 48-bit block using a given expansion function;
2. Compute $R_{i-1} \oplus k_i$;
3. Decompose the result into eight 6-bit blocks $B_1B_2B_3B_4B_5B_6B_7B_8$;
4. Replace independently each 6-bit block B_i by a 4-bit block C_i ;
5. Assemble the eight 4-bit blocks to obtain a 32-bit block $C_1C_2C_3C_4C_5C_6C_7C_8$;
6. Permute C using a given permutation function.

DES *f* function: permutation of *C* (32 bits)

- Finally, the *f* function is again permuted using the below permutation P table:

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

$$\text{new } b_1 \leftarrow \text{old } b_{16} \qquad \text{new } b_2 \leftarrow \text{old } b_7$$
$$\text{new } b_3 \leftarrow \text{old } b_{20} \qquad \text{new } b_4 \leftarrow \text{old } b_{21}$$

- We have finished all steps of the *f* function.
- This value that we have derived (the encrypted data) is known mathematically as $f(R_{i-1}, K_i)$.

DES stage 2: XORing with the Left Block

- Now, we take the left half of the block we left previously, and XOR it with the $f(R_{i-1}, K_i)$ block that we got after permuting in the previous step.
- This gives us R_1, the result of the first round of processing.
- The above 3 steps are run 15 more times to complete 16 rounds of processing.

DES: stage 3

1. Final Permutation = IP^{-1} (64 bits)
2. The result of the final round is permuted one last time following the given IP^{-1} table.

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

$$\text{new } b_1 \leftarrow \text{old } b_{40} \qquad \text{new } b_2 \leftarrow \text{old } b_8$$
$$\text{new } b_3 \leftarrow \text{old } b_{48} \qquad \text{new } b_4 \leftarrow \text{old } b_{16}$$

3. The IP^{-1} table is the inverse of the initial table P.
4. Finally, the output of this inverse permutation table is the ciphertext of the DES algorithm.

DES: weaknesses

1. Symmetric: the algorithm is symmetric:
    1. the same initial key and the same algorithm will decrypt the ciphertext into the original plaintext.
2. Public algorithms and functions:
    1. the algorithm and the permutations, functions, ... are well known.
    2. security depends on the length of the key.
3. Initial and final permutations: useless.

Variant: 3DES

1. Uses a series of DES encryptions/decryptions:
    1. there exist multiple variants.
2. The official accepted variant is DES-EDE, that is, GIVEN 3 keys k_1, k_2 and k_3, compute:
    1. Ciphertext = $E_{k3}(D_{k2}\ (E_{k1}(\text{Plaintext})))$
3. Note that $k_1 = k_2 = k_3$ make Triple DES compatible with DES.
4. It requires three times the computation of normal DES:
    1. stronger than DES
5. Key size: 168 bits.