[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Sanidhya Kumar (202151138) — Lecture (Week 3)

1 Decryption in DES

Ciphertext is first processed using IP (since at the end of DES encryption network we had an IP^{-1}), then further the inversion is done as follows –

$$R_{15} = L_{16}$$
$$L_{15} = R_{16} \oplus f(L_{16}, k_{16})$$

This process continues until we reach L_1 and R_1. After getting L_1 and R_1, we perform the following to obtain R_0 and L_0 –

$$R_0 = L_1$$
$$L_0 = R_1 \oplus f(L_1, k_1)$$

At last, we perform an IP^{-1} (since at the start of DES encryption network we had an IP). After the successful completion of the above-mentioned process, we finally get the plaintext!

2 Elements used in DES

2.1 Initial Permutation (IP)

The Initial Permutation (IP) design involves rearranging the bit positions in the data using the table presented below. The bits are permuted according to the specified table, resulting in two halves: L_0 and R_0, with a 32-bit split. In mathematical terms,

$$IP : \{0,1\}^{64} \to \{0,1\}^{64}$$

can be expressed as

$$IP(m_1, m_2, m_3, \ldots, m_{64}) = m_{58}m_{50}m_{42} \ldots m_7.$$

The IP matrix illustrates the bit positions that replace the actual bit positions during this permutation process. Calculating the inverse of IP is straightforward; it involves populating the matrix with position values that correspond to the actual bit positions in the plaintext message matrix.

For instance, the 1st bit in the plaintext occupies the 40th position, the 2nd bit is in the 8th position, and the 3rd bit is in the 48th position of the IP matrix. Consequently, the first three entries of the IP^{-1} matrix will be 40, 8, and 48, respectively.

2.2 Round function (f)

The function $F(R_i, K_i)$ is defined as X_{i+1}, where R_i and X_{i+1} are both 32 bits in length, and K_i is 48 bits.

Mathematically, $F : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$ is expressed as:

$$F(R_i, K_i) = X_{i+1}, \text{ where } R_i \text{ and } X_{i+1} \text{ are of 32 bits and } K_i \text{ is of 48 bits.}$$

Further elaborating on the expression, we have:

$$F(R_{i-1}, K_i) = P(S(E(R_{i-1} \oplus K_i)))$$

Here, E maps a 32-bit input to a 48-bit output, and P involves permuting the positions of 32 bits. The S-Box takes a 48-bit input and produces a 32-bit output.

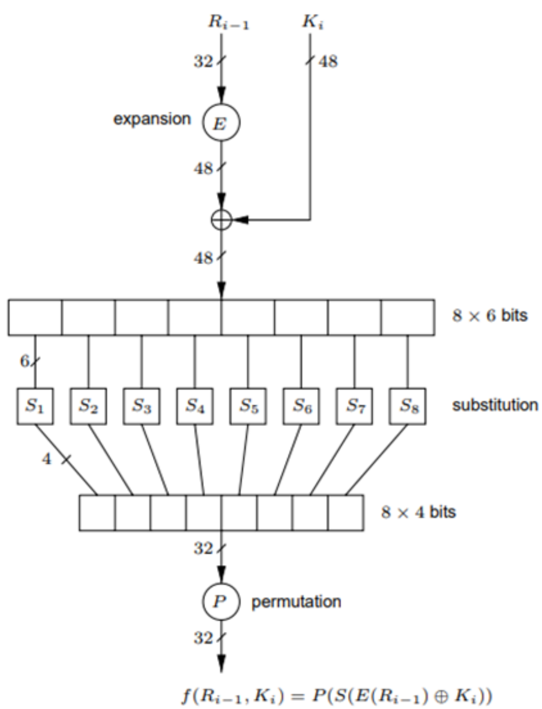


Figure 1: Round function f in DES

2.2.1 E - Mapping

E is a mapping shown by the following diagram:

$$\begin{array}{ccccccc}
 & \vdots & & \vdots & & \vdots & \vdots \\
x_1 & x_2 & x_3 & \cdots & x_{32} & & \\
\hline
 & \vdots & & \vdots & & \vdots & \vdots \\
y_1 & y_2 & y_3 & \cdots & y_8 & &
\end{array}$$

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

Figure 2: E inside round function f

It's observed that the last two column values are repeated. For instance, in the second row, the values 4 and 5 from the last two columns of the first row are duplicated. This repetition pattern is illustrated as:

$$E : \{0,1\}^{32} \to \{0,1\}^{48}$$

$$E(x_1x_2x_3 \ldots x_{32}) = y_1y_2y_3 \ldots y_8$$

$$E(x_1x_2x_3 \ldots x_{32}) = (x_{32}x_1x_2x_3x_4 \ldots x_{32}x_1)$$

2.2.2 S-Box

The S-Box in function f is responsible for mapping 48 bits of data to 32 bits, denoted as:

$$S : \{0,1\}^{48} \to \{0,1\}^{32}$$

For a given 48-bit input $X = B_1B_2B_3B_4B_5B_6B_7B_8$, where each B_i represents a block of length 6 bits, the S-Box function is defined as $S(X) = Y$, where Y is a 32-bit output.

Each B_i is further broken down into $b_1b_2b_3b_4b_5b_6$, where b_i belongs to $\{0,1\}$. The mapping for each S_i from 6 bits to 4 bits is expressed as:

$$S_i : \{0,1\}^6 \to \{0,1\}^4 \text{ for } i = 1,2,3,\ldots,8$$

The mapping function $S_i(B_i) = C_i$, where C_i is a 4-bit output. The overall S-Box operation for the 48-bit input X is given by:

$$S(X) = (S_1(B_1), S_2(B_2), \ldots, S_8(B_8))$$

In the figure below, the S-Boxes in DES are fixed. Each S_i is represented as a matrix with 4 rows (0 to 3) and 16 columns (0 to 15). The values a_{ij} in the matrix range from 0 to 15, and the table in Figure 10 associates these values with x in the range from 0 to 15. The calculations for r and c are specified as:

$$r = (2 \cdot b_1 + b_6) \quad \text{where } 0 \leq r \leq 3$$

$$c = \text{integer representation of } (b_2 b_3 b_4 b_5) \quad \text{where } 0 \leq c \leq 15$$

row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[0]	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
[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
S_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 3: S inside round function f

2.2.3 Permutation (P)

The Permutation function in DES performs the permutation of 32 bits of data and outputs 32 bits. The specific permutation applied in DES is illustrated in Figure 11.

$$P : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$$

For a given input $x_1x_2x_3 \ldots x_{32}$, the permutation function is defined as:

$$P(x_1x_2x_3 \ldots x_{32}) = (x_{16}x_7x_{20}x_{21} \ldots x_4x_{25})$$

In this permutation, the output rearranges the input bits based on the specified order outlined below.

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

Figure 4: P inside round function f

2.3 Key Scheduling Algorithm

Key K is a 64-bit sequence, denoted as $k_1 \ldots k_{64}$, inclusive of 8 odd-parity bits. The desired outcome involves generating 16 round keys, K_i ($1 \leq i \leq 16$), each with a length of 48 bits.

Algorithm Breakdown:

1. Define v_i, where $1 \leq i \leq 16$, as follows: v_i is set to 1 for i belonging to $\{1, 2, 9, 16\}$; otherwise, v_i is set to 2. These values represent the left-shift amounts for 28-bit circular rotations.

2. Discard the 8 parity check bits from K, resulting in $\tilde{k}$.

3. Represent T as 28-bit halves (C_0, D_0) after applying the permutation choice $PC1$ to $\tilde{k}$. $PC1$ maps $\{0,1\}^{56}$ to $\{0,1\}^{56}$.

4. Utilize $PC1$ (refer to the table in Figure 12) to select bits from K, obtaining $C_0 = k_{57}k_{49} \ldots k_{36}$ and $D_0 = k_{63}k_{55} \ldots k_4$. Both C_0 and D_0 are 28 bits in length.

5. For each i from 1 to 16, compute K_i as follows:

   - C_i undergoes left circular shift by v_i, denoted as $C_i \leftarrow (C_{i-1} \hookleftarrow v_i)$.
   - D_i undergoes left circular shift by v_i, denoted as $D_i \leftarrow (D_{i-1} \hookleftarrow v_i)$.
   - Perform the left circular shifts using $\hookleftarrow$. C_i and D_i represent the left and right components, respectively.
   - $PC2$, detailed in the table from Figure 12, is used to select 48 bits from the concatenation $b_1b_2 \ldots b_{56}$ of C_i and D_i, resulting in $K_i = b_{14}b_{17} \ldots b_{32}$.

- Lastly, compute K_i using $PC2(C_i, D_i)$, where K_i is the round key for the ith round. $PC2$ maps as follows: $PC2 : \{0,1\}^{56} \rightarrow \{0,1\}^{48}$.

PC1						
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
above for C_i; below for D_i						
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

PC2					
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

Figure 5: PC1 and PC2 inside Key scheduling algorithm

Few Interesting Properties

The application of $PC1$ to the key $K_1K_2K_3...K_{63}K_{64}$ involves a permutation of positions, excluding the parity bits. The resulting sequence is $K_{57}K_{49}K_{41}K_{33}\dots K_4$.

Notably, in $PC1$, the operation is solely focused on rearranging positions. If the keys are complemented, the output will also be complemented.

Expressed in a general form, $PC1(K_{1,K_{2,\dots,K_{64)}}}$ yields $K_{57K_{49\dots K_9}}$, equivalent to $PC1(K)$.

Regarding $PC2$, according to the information in Figure 12, the input bit at the 14th position is transferred to the 1st position in the output during the permutation.

3 Complement Properties

The DES encryption operation, denoted as $DES(M, K) = C$, and its complemented counterpart, $DES(\bar{M}, \bar{K}) = \bar{C}$, are linked.

Key scheduling, denoted as $KS(K)$, produces round keys $K_1, K_2, \dots, K_{16}$, while the complemented key scheduling, $KS(\bar{K})$, yields round keys $\bar{K}_1, \bar{K}_2, \dots, \bar{K}_{16}$. This complementation is consistent across $PC1$, $PC2$, and left shifts.

The rationale behind the result in equation (1) is explained as follows:

Consider the initial setup:

$$\begin{array}{ccc} M & & \bar{M} \\ L_0 & R_0 & \bar{L}_0\bar{R}_0 \\ L_1 & R_1 & \bar{L}_1\bar{R}_1 \end{array}$$

In the first round:

$$\begin{aligned} M: \quad R_1 &= L_0 \oplus F(R_0, K_1) \\ \bar{M}: \quad R_1 &= \bar{L}_0 \oplus F(\bar{R}_0, \bar{K}_1) \end{aligned}$$

If we compare the uncomplemented case (L_1, R_1) to (L_0, R_0), the complementation of both plaintext and key results in complemented inputs for the XOR before the S-boxes. This double complementation cancels out, yielding S-box inputs and the overall result $f(R_0, K_1)$. The result is then XORed with $\bar{L}_0$ (previously L_0), resulting in $\bar{L}_1$.

This effect continues in subsequent rounds. For instance, in the next step:

$$\begin{gathered} \bar{M} : E(\bar{R}_0) = (E(R_0))^{\bar{}} \oplus \bar{K}_1 = E(R_0) \oplus K_1 \\ R_1 = \bar{L}_0 \oplus f(E_0, K) \\ R_1 = \bar{R}_1 \end{gathered}$$

Thus, R_1 will also be complemented. The next step involves the initial permutation (IP), which is also complemented. Consequently, the ciphertext will be the complement!

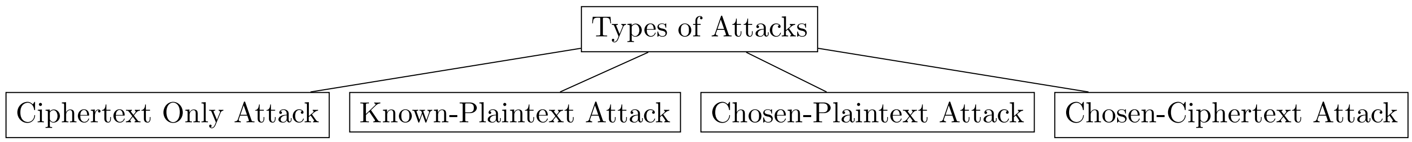
Exhaustive Search or Brute-force Attack in DES:

- The key size in DES is 56 bits, leading to a total of possible keys (S) denoted as $K_1, K_2, \ldots, K_{2^{56}}$.
- The time complexity of a brute-force attack on DES is 2^{56}, exploring all potential keys.

Chosen-Plaintext Attack in DES:

- In this attack, the attacker selects specific plaintexts for which the corresponding ciphertexts are to be provided.
- The total possible keys (S) are $K_1, K_2, \ldots, K_{2^{56}}$.
- Assuming the attacker chooses two plaintexts, M and $\bar{M}$, and uses the secret key K to generate the corresponding ciphertexts:
  - $DES(M, K) = C_1$
  - $DES(\bar{M}, K) = C_2$
- Leveraging the complement property in DES, we have $DES(\bar{M}, \bar{K}) = \bar{C}_2$, where $\bar{M} = M$.
- The attacker employs a test key, K_i, to attempt decryption of M using K_i to obtain a ciphertext C. According to the complement property, decrypting the complement of M with the complement of K_i yields the complement of C.
- If the equality holds true $(C \neq C_1)$, discard K_i from S as K_i is not equal to K.
- If the complemented equality holds true $(\bar{C} \neq \bar{C}_2)$, discard $\bar{K_i}$ from S as $\bar{K_i}$ is not equal to K.
- This approach enables finding the key in 2^{55} attempts, one less than the total keys, as two keys are discarded during the process.

4 Types of Attacks

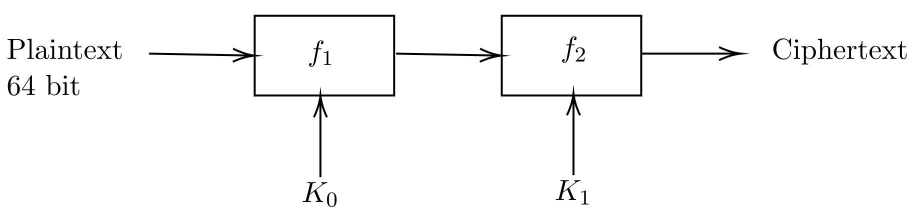


- **Ciphertext Only Attack:**
  - The attacker possesses only the ciphertext and aims to recover either the plaintext or the secret key.
- **Known-Plaintext Attack:**
  - The attacker has knowledge of certain plaintexts and their corresponding ciphertexts.
  - The objective is to either find a plaintext corresponding to a different ciphertext or uncover the secret key.
- **Chosen-Plaintext Attack:**
  - In this attack, the assailant selects specific plaintexts and is allowed to obtain the corresponding ciphertexts.
  - The goal is to generate a new plaintext/ciphertext pair or discover the secret key.
- **Chosen-Ciphertext Attack:**
  - The attacker chooses certain ciphertexts and is provided with their corresponding plaintexts.
  - The aim is to create a different valid plaintext/ciphertext pair or reveal the secret key.

It is worth noting that in public-key cryptography, the Chosen-Ciphertext Attack is considered the strongest, given the pivotal role of the decryption secret key. In symmetric key cryptography, both Chosen-Plaintext and Chosen-Ciphertext attacks are nearly equally potent since the same key is employed for both encryption and decryption.

5 Variations of DES

5.1 Double DES



To enhance the security of DES, a common practice is to perform double encryption. However, the assumption that this would provide $2 \times 56 = 112$ bits of security is proven incorrect, and a Meet-in-the-Middle attack illustrates this.

In double encryption, a key K is represented as 128 bits (K_0, K_1), with 16 parity check bits among them. The encryption process involves two stages using different keys. The diagram depicts plaintext (P) of 64 bits undergoing encryption with DES.

Exhaustive search on double encryption would imply a time complexity of 2^{112}, but this is refuted by the Meet-in-the-Middle attack:

Meet-in-the-Middle Attack:

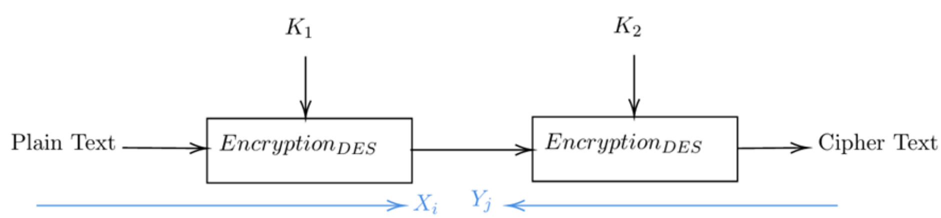
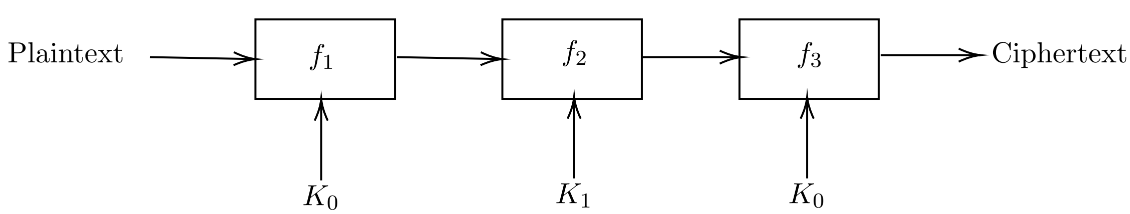


Figure 6: Meet in the middle attack (just a representation - not specific to DES)

With one valid plaintext and ciphertext pair in a known-plaintext model, the attacker performs encryption in reverse order on key K_1 (right to left) and encryption on key K_0 (left to right). If the results match, the keys are found. Let $Enc\_DES(P, K_i) = X_i$, and store (X_i, K_i) in Table 1. Similarly, let $Enc\_DES(C, K_j) = Y_j$, and store (Y_j, K_j) in Table 2. If X_i equals Y_j, then (K_i, K_j) is the secret key. The time complexity is approximately 2^{57} ($\sim 2^{56}$), as Table 1 and Table 2 are independent, and the attacker needs to search 2^{56} possibilities for both.

Key Length Increase: Increasing the length of the secret key from m bits to $2m$ bits, even in an encryption algorithm providing n bits of security, does not proportionally increase security to $2n$. Therefore, using two secret keys in double encryption does not result in a complexity of 2^{2n} but rather maintains a complexity of almost 2^n.

5.2 Triple DES



In the context of triple DES, where two secret keys K_0 and K_1 are employed, and three encryptions are performed, a diagram illustrates the process:

In a Meet-in-the-Middle attack on triple DES, decryption on K_1 is conducted for every possible value of K_0, as depicted in the figure. This approach multiplies the time complexity, resulting in 2^{2n} bit security, where 'n' is the size of the key.

6 Few Mathematical Things

Let's say we have an algorithm which claims n-bit security using exhaustive search (2^n). But if we have Quantum or supercomputers, time complexity will reduce to $2^{n/2}$.

To achieve n-bit security in this setup, we will use a $2n$-length key or a triple encryption setup.

6.1 Binary Operator

$$R \subseteq X \times Y$$

A binary operator $*$ on a set S is a mapping from $S \times S$ to S. In other words, $*$ is a rule that assigns to each ordered pair of elements from S an element of S.

$$* : S \times S \to S$$
$$*(a, b) = c, \text{ where } a, b, c \in S$$
$$*(b, a) = d, \text{ where } d \in S$$

It is not necessary that $d = c$.

6.2 Groups

A group $(G, *)$ consists of a set G with a binary operation $*$ on G satisfying the following three axioms.

(i) **Associativity:** $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$.

(ii) **Identity Element:** There is an element $1 \in G$, called the identity element, such that $a * 1 = 1 * a = a$ for all $a \in G$.

(iii) **Inverse:** For each $a \in G$, there exists an element $a^{-1} \in G$, called the inverse of a, such that $a * a^{-1} = a^{-1} * a = 1$.

NOTE: A group G is abelian (or commutative) if, furthermore, $a * b = b * a$ for all $a, b \in G$.

Examples:

1. Matrix multiplication over square matrices of order $n \times n$:

   - It is not commutative: $A \cdot B$ is not equal to $B \cdot A$.
   - $(G, *) = \{\text{set of all invertible matrices}\}$
   - It satisfies all three group axioms but is not abelian.

     (a) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
     (b) $A \cdot I_n = I_n = I_n \cdot A$
     (c) $A \cdot A^{-1} = I_n = A^{-1} \cdot A$

   So, it is a group with a multiplication operator but is not abelian (commutative).

2. $(\mathbb{Z}, +)$: $\mathbb{Z}$ set of integers with the operation of addition

- It is commutative.
- Identity element: 0
- Inverse of $a \in \mathbb{Z}$: $-a$
- It forms a group under addition.
  (a) $a + (b + c) = (a + b) + c$ for all $a, b, c \in \mathbb{Z}$
  (b) 0: identity element, $a + 0 = a = 0 + a$, for all $a \in \mathbb{Z}$
  (c) For every $a \in \mathbb{Z}$, there exists $-a \in \mathbb{Z}$ such that: $a + (-a) = 0 = (-a) + a$

3. $(\mathbb{Z}, *)$: $\mathbb{Z}$ set of integers with the operation of multiplication

   - It is not a group as there is no inverse for every element.
     (a) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c \in \mathbb{Z}$
     (b) $a \cdot 1 = 1 \cdot a$
     (c) For $a \in \mathbb{Z}$, there does not exist $1/a\,(a^{-1}) \in \mathbb{Z}$

   So, set of integers $\mathbb{Z}$ with the operation of multiplication is not a group.

4. $(\mathbb{Z}, -)$: $\mathbb{Z}$ set of integers with the operation of subtraction

   - It is not a group due to a lack of associativity.
     (a) $(a - (b - c)) \neq (a - b) - c$

   So, it is not a group.

5. $(\mathbb{Q}, *)$: $\mathbb{Q}$ set of all rational numbers with multiplication operation

   - Additional information or examples are needed to analyze its group properties.

6. $(\mathbb{Q} - \{0\}, *)$: $\mathbb{Q}$ set of all rational numbers except 0 with multiplication operation

   - Yes, it is a group!