1. A DES cipher algorithm utilizes 34 permutations in total, including an initial permutation, a final permutation, and two permutations within each of the 16 encryption rounds, making 16 rounds with two permutations per round, adding up to 34 permutations in total.

In the Data Encryption Standard (DES) round-key generator, 17 permutations are used: one "parity drop" permutation (Permuted Choice 1) to initially select 56 bits from the 64-bit key, and 16 "compression" permutations (Permuted Choice 2) to generate the 48-bit round keys for each of the 16 encryption rounds.

1. In the DES algorithm, the expansion permutation is necessary to increase the number of bits in the data block from 32 to 48, allowing it to be XORed with the 48-bit round key during the encryption process, thus enhancing the diffusion property of the cipher by spreading the influence of each bit across a larger area; essentially, it ensures that changes in a few bits of the data significantly impact the output after multiple rounds.
2. In the Data Encryption Standard (DES) algorithm, the round-key generator uses a parity drop permutation to remove the parity bits from the original key, resulting in a 56-bit key used for encryption, effectively eliminating potential weaknesses that could arise from having predictable parity patterns in the key, which could be exploited by attackers to crack the encryption.
3. .
4. .
5. The key difference between permutations in DES and AES is that DES utilizes "expansion permutation" to increase the number of bits used in the key-mixing stage, while AES does not require this step because its design inherently provides sufficient diffusion through its substitution and permutation operations across the entire block, making separate expansion and compression permutations unnecessary; essentially, AES achieves a similar effect without dedicated expansion stages due to its more complex internal structure.
6. In DES, the round key size is smaller than the block size (48 bits compared to 64 bits), while in AES, the round key size is always the same as the block size (128 bits) regardless of the chosen key length (128, 192, or 256 bits); therefore, AES is the cipher where the round key size matches the block size.

**Step 1:**We accepted a 10-bit key and permuted the bits by putting them in the P10 table.

Key = 1 0 1 0 0 0 0 0 1 0

(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (1, 0, 1, 0, 0, 0, 0, 0, 1, 0)

P10 Permutation is: P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)

After P10, we get 1 0 0 0 0 0 1 1 0 0

**Step 2:**We divide the key into 2 halves of 5-bit each.

l=1 0 0 0 0, r=0 1 1 0 0

**Step 3:**Now we apply one bit left-shift on each key.

l = 0 0 0 0 1, r = 1 1 0 0 0

**Step 4:**Combine both keys after step 3 and permute the bits by putting them in the P8 table. The output of the given table is the first key K1.

After LS-1 combined, we get 0 0 0 0 1 1 1 0 0 0

P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3, k7, k4, k8, k5, k10, k9)

After P8, we get Key-1 : 1 0 1 0 0 1 0 0

**Step 5:**The output obtained from step 3 i.e. 2 halves after one bit left shift should again undergo the process of two-bit left shift.

Step 3 output - l = 0 0 0 0 1, r = 1 1 0 0 0

After two bit shift - l = 0 0 1 0 0, r = 0 0 0 1 1

**Step 6:**Combine the 2 halves obtained from step 5 and permute them by putting them in the P8 table. The output of the given table is the second key K2.

After LS-2 combined = 0 0 1 0 0 0 0 0 1 1

P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3, k7, k4, k8, k5, k10, k9)

After P8, we get Key-2 : 0 1 0 0 0 0 1 1

**Final Output:**

Key-1 is: 1 0 1 0 0 1 0 0

Key-2 is: 0 1 0 0 0 0 1 1

1. A.

***ShiftRows Transformation:***

**Input:** A 4×4 state matrix S containing 16 bytes.  
**Output:** A modified 4×4 state matrix S' after the ShiftRows transformation.

**Algorithm:**

**Row 0 (First row):**

Do not shift any bytes.

**Row 1 (Second row):**

Perform a left circular shift by **1** byte.

The first byte moves to the end of the row.

**Row 2 (Third row):**

Perform a left circular shift by **2** bytes.

The first two bytes move to the end of the row.

**Row 3 (Fourth row):**

Perform a left circular shift by **3** bytes.

The first three bytes move to the end of the row.

B.

***MixColumns transformation:***

**Step1. Sub Bytes**

This step implements the substitution.

In this step, each byte is substituted by another byte. It is performed using a lookup table also called the [S-box](https://www.geeksforgeeks.org/what-is-s-box-substitution/). This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16-byte (4 x 4 ) matrix like before.

The next two steps implement the permutation.

**Step2. Shift Rows**

This step is just as it sounds. Each row is shifted a particular number of times.

* The first row is not shifted
* The second row is shifted once to the left.
* The third row is shifted twice to the left.
* The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

***[ b0 | b1 | b2 | b3 ] [ b0 | b1 | b2 | b3 ]******| b4 | b5 | b6 | b7 | -> | b5 | b6 | b7 | b4 |******| b8 | b9 | b10 | b11 | | b10 | b11 | b8 | b9 |******[ b12 | b13 | b14 | b15 ] [ b15 | b12 | b13 | b14 ]***

**Step 3: Mix Columns**

This step is a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

**This step is skipped in the last round.**

***[ c0 ] [ 2 3 1 1 ] [ b0 ]******| c1 | = | 1 2 3 1 | | b1 |******| c2 | | 1 1 2 3 | | b2 |******[ c3 ] [ 3 1 1 2 ] [ b3 ]***

**Step 4: Add Round Keys**

* Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes are not considered as a grid but just as 128 bits of data.
* After all these rounds 128 bits of encrypted data are given back as output. This process is repeated until all the data to be encrypted undergoes this process.

C.

***key-expansion of AES-128:***

**Step1. Sub Bytes**

This step implements the substitution.

In this step, each byte is substituted by another byte. It is performed using a lookup table also called the [S-box](https://www.geeksforgeeks.org/what-is-s-box-substitution/). This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16-byte (4 x 4 ) matrix like before.

The next two steps implement the permutation.

**Step2. Shift Rows**

This step is just as it sounds. Each row is shifted a particular number of times.

* The first row is not shifted
* The second row is shifted once to the left.
* The third row is shifted twice to the left.
* The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

***[ b0 | b1 | b2 | b3 ] [ b0 | b1 | b2 | b3 ]******| b4 | b5 | b6 | b7 | -> | b5 | b6 | b7 | b4 |******| b8 | b9 | b10 | b11 | | b10 | b11 | b8 | b9 |******[ b12 | b13 | b14 | b15 ] [ b15 | b12 | b13 | b14 ]***

**Step 3: Mix Columns**

This step is a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

**This step is skipped in the last round.**

***[ c0 ] [ 2 3 1 1 ] [ b0 ]******| c1 | = | 1 2 3 1 | | b1 |******| c2 | | 1 1 2 3 | | b2 |******[ c3 ] [ 3 1 1 2 ] [ b3 ]***

**Step 4: Add Round Keys**

* Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes are not considered as a grid but just as 128 bits of data.
* After all these rounds 128 bits of encrypted data are given back as output. This process is repeated until all the data to be encrypted undergoes this process.