Modern Symmetric-Key Ciphers

Table of Contents

[Substitution and Transposition 2](#_Toc134226603)

[Components 2](#_Toc134226604)

[P-Boxes 3](#_Toc134226605)

[S-Boxes 4](#_Toc134226606)

[XOR Operations 5](#_Toc134226607)

[Product Ciphers 6](#_Toc134226608)

[Diffusion 7](#_Toc134226609)

[Confusion 8](#_Toc134226610)

[Feistel Ciphers 8](#_Toc134226611)

[Basic Structure 8](#_Toc134226612)

[Improvement 10](#_Toc134226613)

[Final Design 10](#_Toc134226614)

[Data Encryption Standard 11](#_Toc134226615)

[DES Function 13](#_Toc134226616)

[Expansion P-Box 14](#_Toc134226617)

[S-Box 15](#_Toc134226618)

All the symmetric key ciphers we have seen so far were **character oriented**. However, there are various forms of data in the digital world that do not have any characters, such as images or videos. These require **bit-oriented** ciphers. Bit-oriented ciphers are more secure, since we have a larger number of symbols, which increases security.

The process takes an -bit plaintext and encrypts it using a -bit key, producing an -bit ciphertext. If the plaintext has less than bits, **padding** is added. If there is more, then the text is divided into blocks of bits. Common values for are 64, 128, 256 and 512.

## Substitution and Transposition

The ciphers can be either substitution ciphers or transposition ciphers. **Substitution ciphers** are harder to crack since they change the number of 1s and 0s in the text. Transposition ciphers just change the positions of the 1s and 0s, but leave the number of 1s and 0s intact. Modern ciphers use the substitution method because transposition is vulnerable to **exhaustive search attacks**.

## Components

Modern block ciphers are **keyed substitution ciphers**. The key only allows partial mappings from the possible inputs to the possible outputs. Normally, they contain **multiple units**.

Modern block ciphers have two required properties:

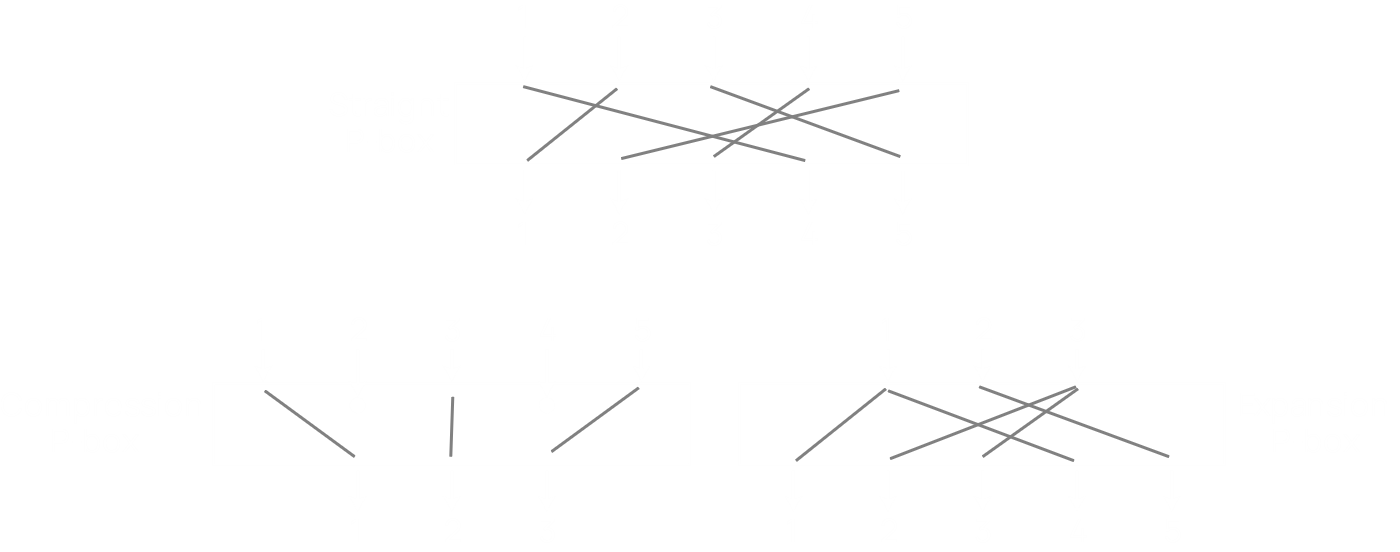
* **Diffusion** – The relationship between the plaintext and the ciphertext must be hidden. This implies that each symbol in the ciphertext depends on multiple symbols in the plaintext. A small change in the plaintext will result in a large change in the ciphertext.
* **Confusion** – The relationship between the key and the ciphertext must be hidden. Thus, if there is a small change in the key, there will be a large change in the ciphertext.

To do this, the block ciphers use a combination of several units, such as the **transposition units** (called P-Boxed) and **substitution units** (called S-Boxes).

## P-Boxes

A **Permutation Box** (P-Box) is responsible for transposing bits. There are three types of P-Boxes, Straight P-Boxes, Expansion P-Boxes and Compression P-Boxes.

A **Straight P-Box** has inputs and outputs. It simply changes the position of the bits. An **Expansion P-Box** not only changes the position of the bits, but also adds new bits. The new bits come from the input. Thus, there are inputs and outputs, where . A **Compression P-Box** reduces the number of bits and also changes their positions. It has inputs and outputs where .

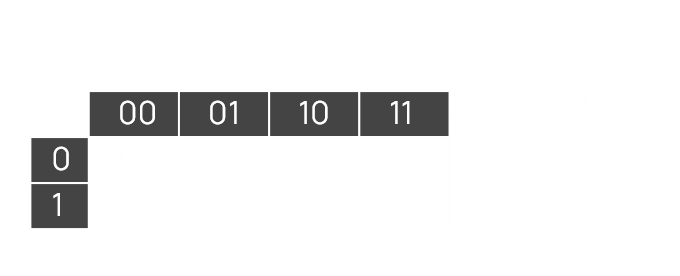


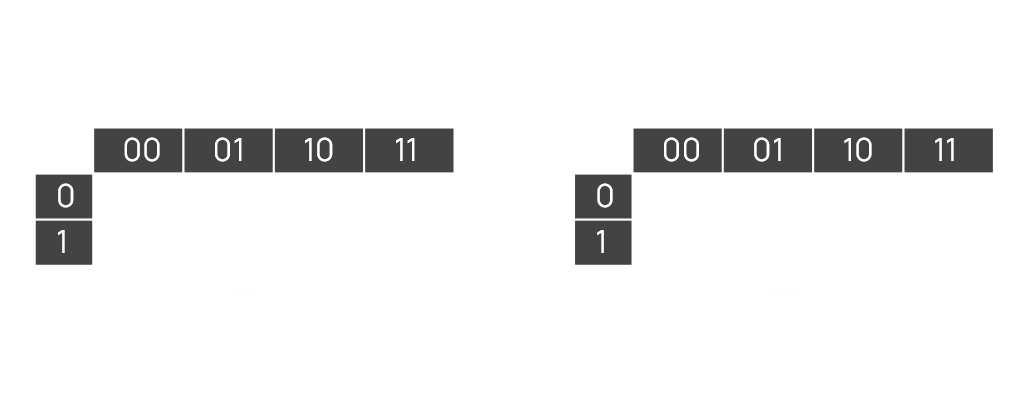
Among the different P-Boxes, only the **Straight P-Box** is reversible. For the other two, we either have lost a few bits or have duplicated them leading to us being unable to identify the duplicates. This makes them irreversible. Thus, only the Straight P-Box can be used in the encryption process, with its inverse being used in the decryption process.

For all three P-Boxes, the mapping from inputs to outputs (i.e., the way that they are transposed) is **pre-determined**.

## S-Boxes

A **Substitution Box** (S-Box) takes words of bits and translates them into words of bits. might be equal to , in which case the substitution is reversible, but they might also be different, in which case the substitution is irreversible. S-Boxes can be keyed or keyless, but modern ciphers use the keyless variant which have **pre-determined mappings**.





## XOR Operations

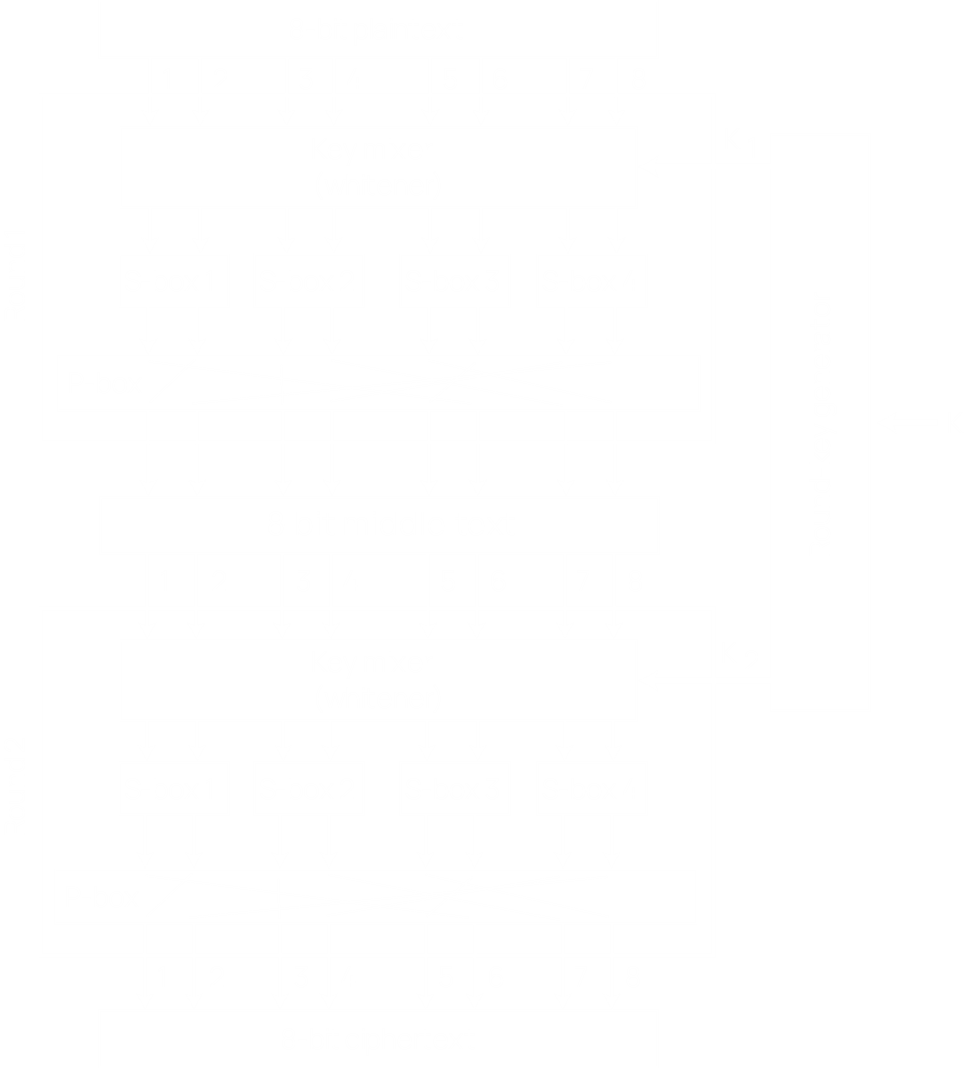
Most block ciphers use the **XOR Operation**, so a brief review of them is provided. The XOR operation over two operands returns true only if the values of the operands are different and returns false otherwise. This operation has 5 properties:

* **Associativity** -
* **Commutativity** -
* **Existence of Identity** -
* **Existence of Inverse** -
* **Complement** - and

The **inverse** of an XOR operation only makes sense if one of the inputs is **fixed** during both the encryption and decryption process. This fixed input is the key. In this case, the XOR operation is **self-inversible**.

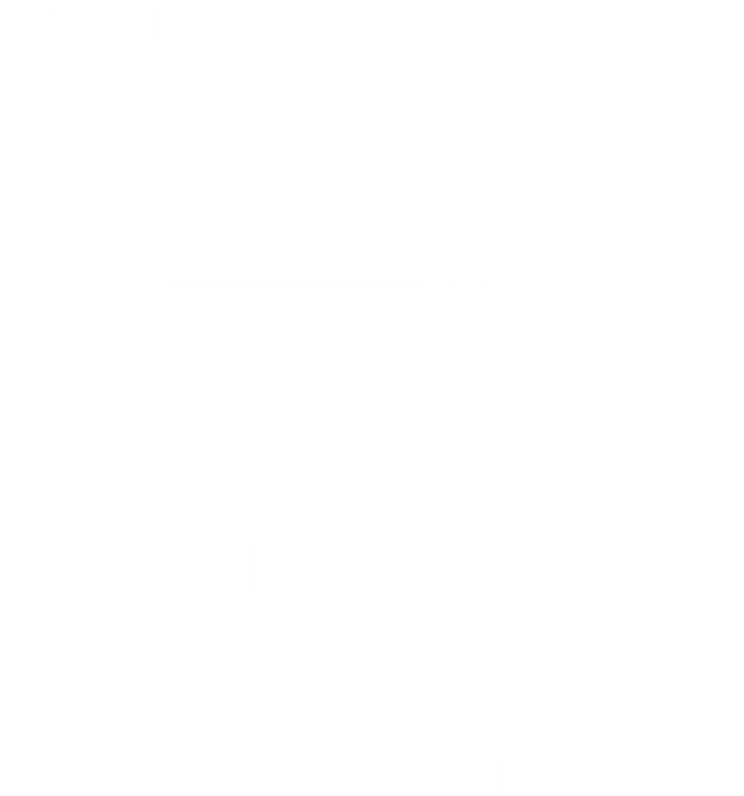
## Product Ciphers

A **Product Cipher** combines S-Boxes, P-Boxes and XOR operations in multiple rounds. For each round, a **Key Generator** creates an **8-bit key**. For rounds, the plaintext will be encrypted times and the ciphertext will have to be decrypted times. The text between two rounds is referred to as the **middle text**.



The **Key Mixer** is said to **whiten** then text. This involved an **XOR operation** between the 8-bit word and the 8-bit key. The output from this is fed to **4 S-Boxes** as **2-bit groups**. The number of outputs from the S-Boxes is the same. This is then given to a **Straight P-Box** which permutes the bits and gives the 8-bit output for that round.

### Diffusion



To understand how a product cipher provides us with **diffusion**, consider the 8th bit in the plaintext word. In the first round, it is XORed with and passed to the S-Box , where it affects bits 7 and 8. These get permutated to bits 2 and 4 by the P-Box. In the second round, bits 2 and 4 are XORed with and given to the S-Boxes and , where they affect bits 1 though. These in turn are passed through a P-Box where they get transposed to bits 1, 3, 6 and 7. Thus, by the end of just 2 rounds, the single 8th bit from the original plaintext word has affected 4 bits in the ciphertext.

Going through this process in the reverse direction shows that each bit in the ciphertext is affected by several bits from the plaintext.

### Confusion

**Confusion** can also be seen from the same diagram. The 4 bits in the ciphertext, bits 1, 3, 6 and 7, are affected by 2 bits from , bits 2 and 4, and 1 bit from , bit 8. Thus, each bit from a key affects multiple bits from the ciphertext, obscuring the relationship between them.

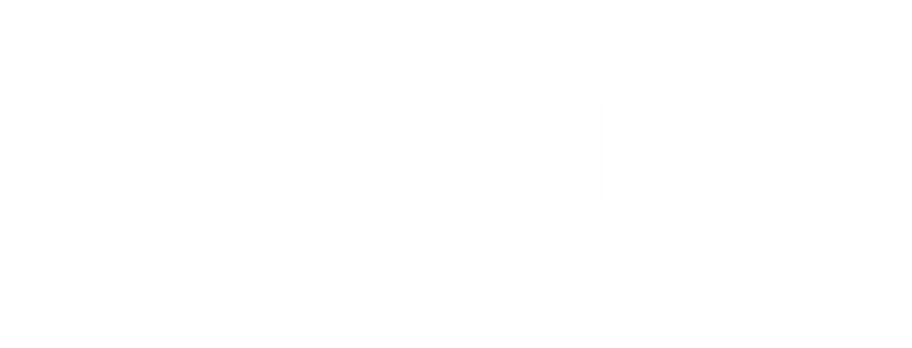
## Feistel Ciphers

Modern block ciphers are all **product ciphers**. They can be divided into two categories, Feistel Ciphers and Non-Feistel Ciphers.

**Feistel Ciphers** consist of self-invertible, invertible and non-invertible components. A commonly used Feistel Cipher is the block cipher used by the **Data Encryption Standard** (DES). **Non-Feistel Ciphers** consist of only invertible components. A commonly used Non-Feistel Cipher is the block cipher used by the **Advanced Encryption Standard** (AES).

In this section, we will be studying Feistel Ciphers.

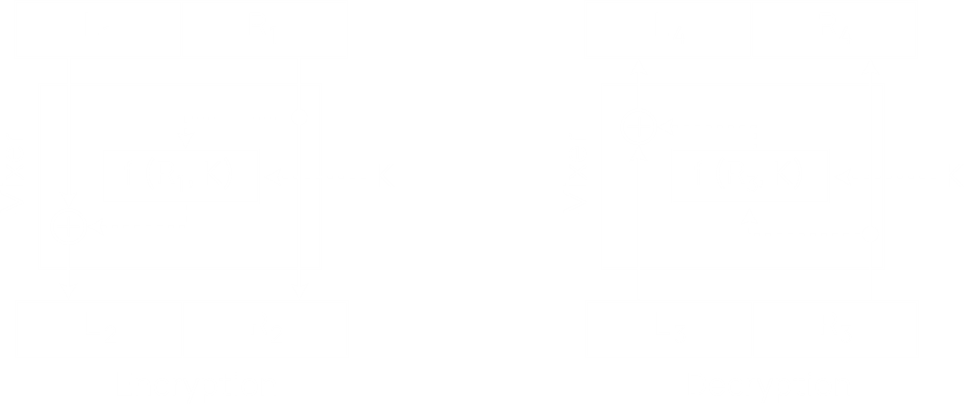
### Basic Structure



The Feistel cipher combines all of its non-invertible components into one unit, denoted by in the diagram above. This same unit is used in both the encryption and decryption process. This ensures that the effect of the non-invertible units **cancel out**, allowing us to invert the encryption process despite using non-invertible units.

First, we have the plaintext . This undergoes a XOR operation with the output of , producing . On the decryption side, is denoted as , but they are exactly the same. This undergoes another XOR operation with the output of , producing .

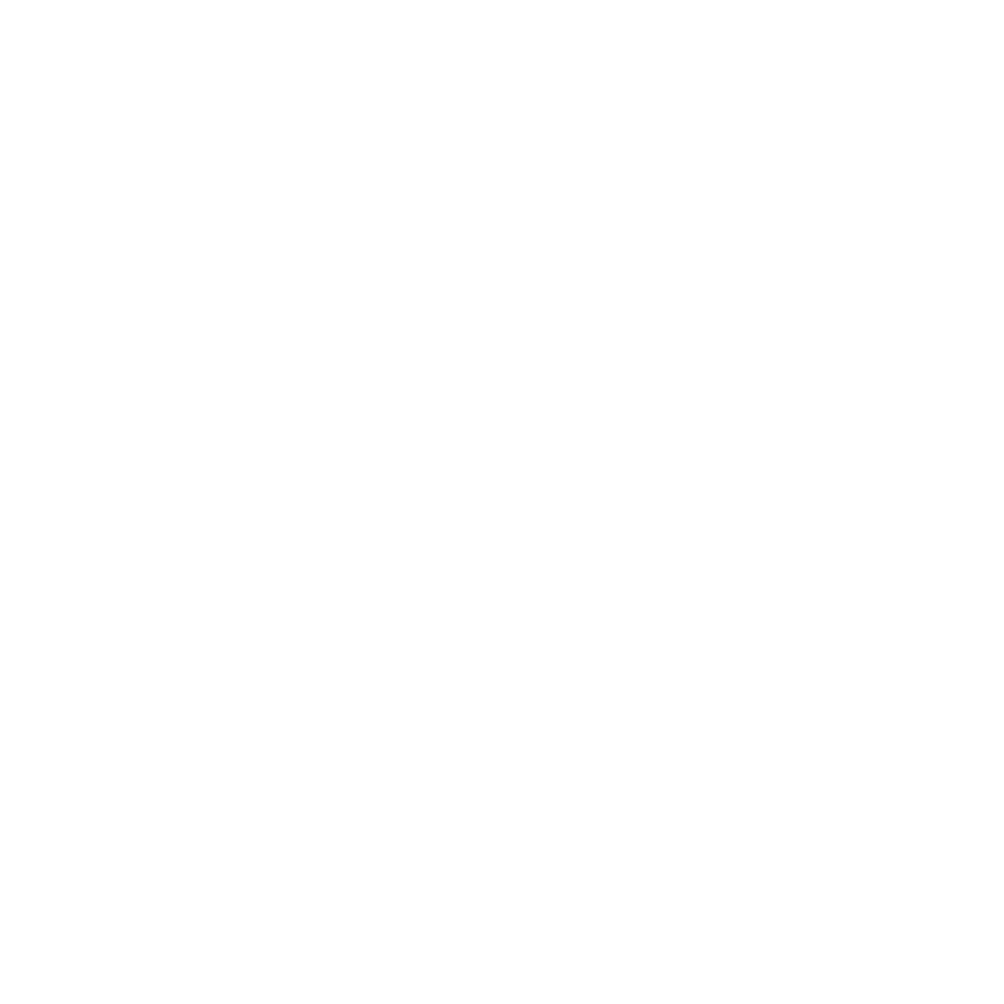
### Improvement



As an improvement over the existing process, the input can be first divided into two sections, marked as and . undergoes an XOR operation with the output of , producing , while is exactly the same as . These go to the decryption end, where and are used to denote and respectively, despite the pairs being exactly the same. undergoes an XOR operation with the output of , producing , while is directly mapped onto .

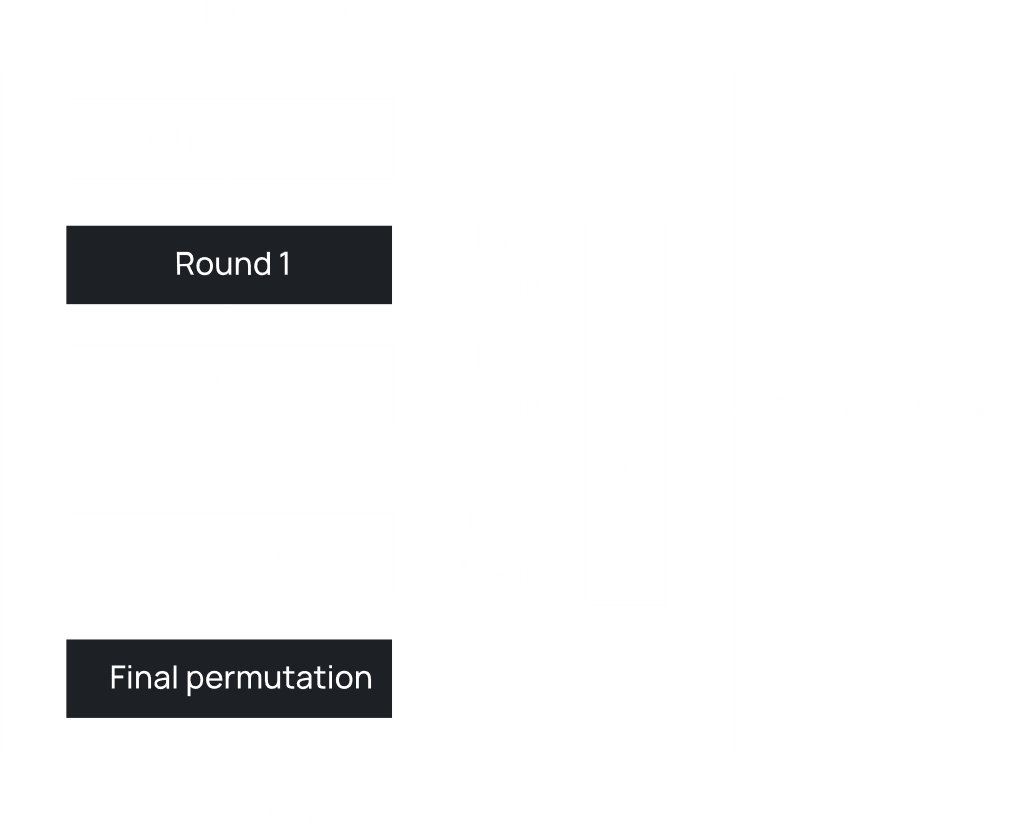
### Final Design

Notice that remained unchanged and directly went to in the improvement. That would however, mean that the right half of our text is unencrypted. In reality, the final design has multiple rounds of the improved operation, where the bits are flipped after each round.



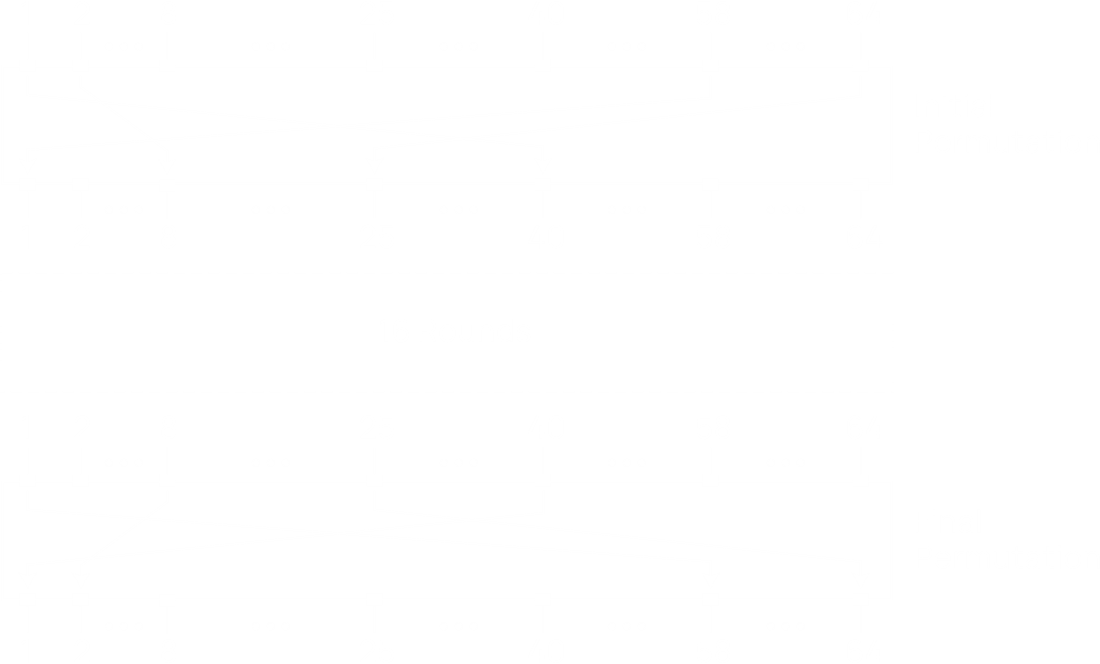
## Data Encryption Standard

The **Data Encryption Standard** (DES) is a standard which uses a block Feistel cipher. It works with **64-bit plaintexts** and **56-bit cipher keys**.

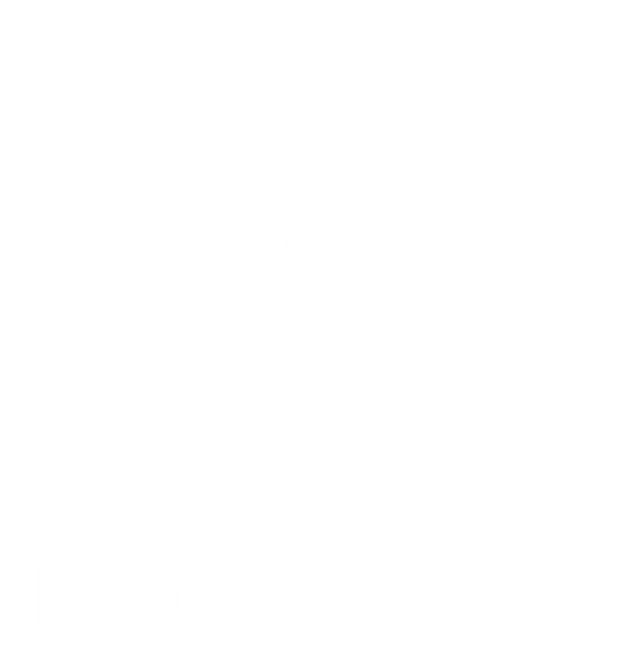


The block itself uses **16 rounds** and starts and ends with a **P-Box**. Each round uses a **48-bit round key** generated from the 56-bit cipher key using a **Round-Key Generator**, which internally uses a predefined algorithm.

The Initial and Final P-Boxes permute the bits based on a pre-defined rule. They are both **keyless straight permutations** and are the inverses of each other, meaning if the Initial P-Box makes the 58th bit the 1st one, the Final P-Box will make the 1st bit the 58th one. Overall, they do nothing and the reason they were included at all is unclear, but is assumed to have been to thwart software simulation.

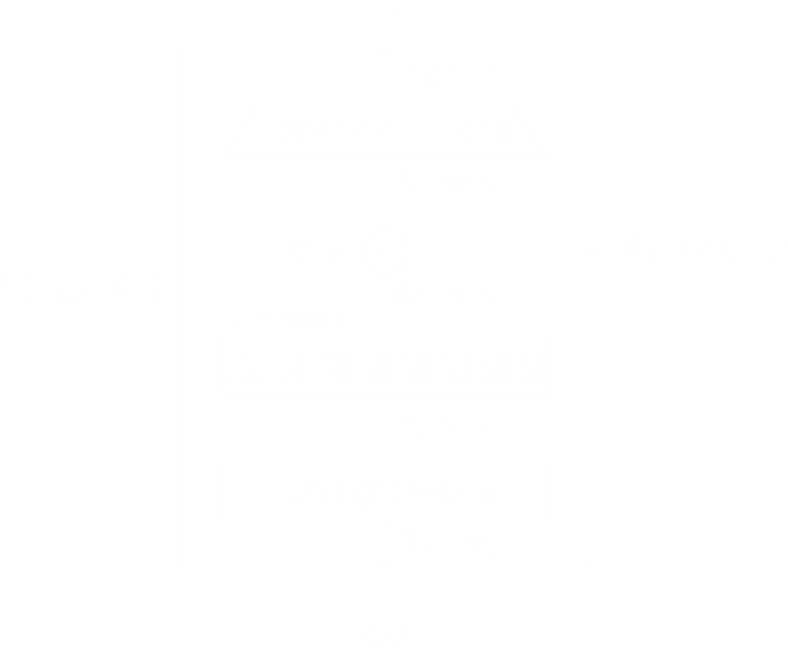


Each of the rounds themselves work exactly the same way as the final design of the Feistel cipher.



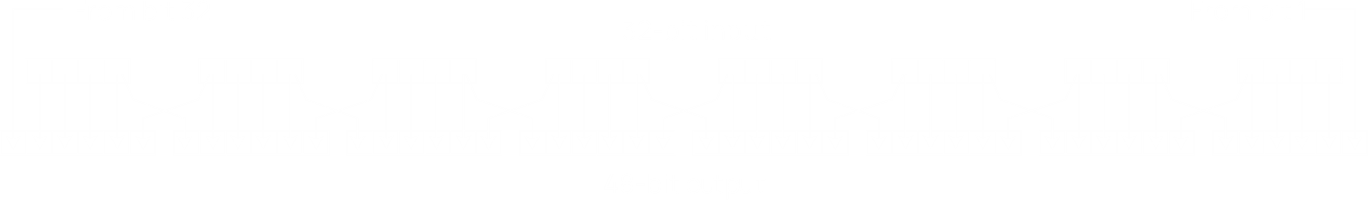
## DES Function

So far, we have avoided talking about what exactly happens inside the function . The exact implementation will vary in different implementations of Feistel ciphers, but DES uses the following implementation. The DES function is called the **heart** of DES.



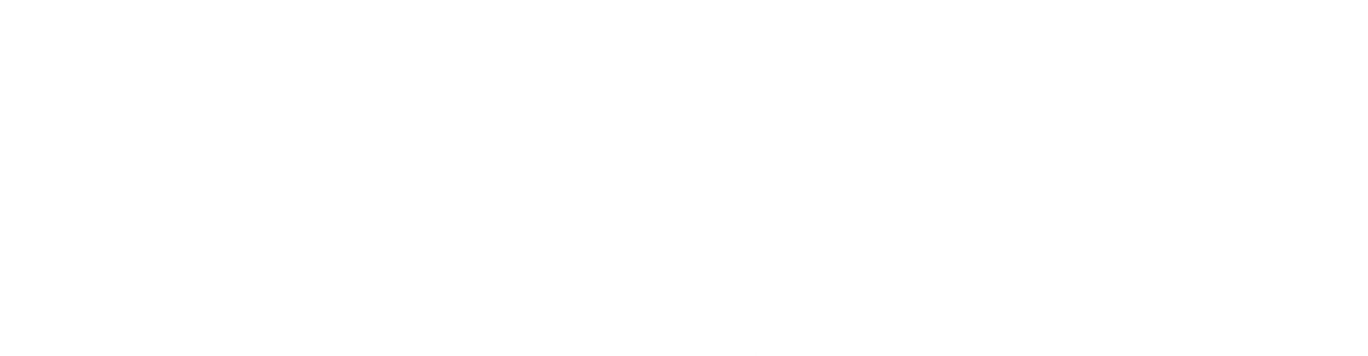
The 32-bit right-half of the plaintext goes through an **expansion P-Box**, which gives a 48-bit output. This undergoes an XOR operation with the 48-bit round key. The output of this is divided into groups of 6, each group going to a separate S-Box (a total of 8 S-Boxes). Finally, the S-Boxes give a 32-bit output combined, which goes through a **straight P-Box**, which produces the final output.

### Expansion P-Box



In the **Expansion P-Box**, the 32 bits are divided into 8 **4-bit sections**, each of which are expanded to 6 bits. Bits 1 through 4 become bits 2 through 5 of the output bits. Bit 1 of the output bit is input bit 4 from the previous section, while bit 6 of the output bits is input bit 1 from the next section.

### S-Box



The S-Box takes each of the 6-bit outputs from the P-Box sections and substitutes them for 4-bit outputs, creating a 32-bit output in total.