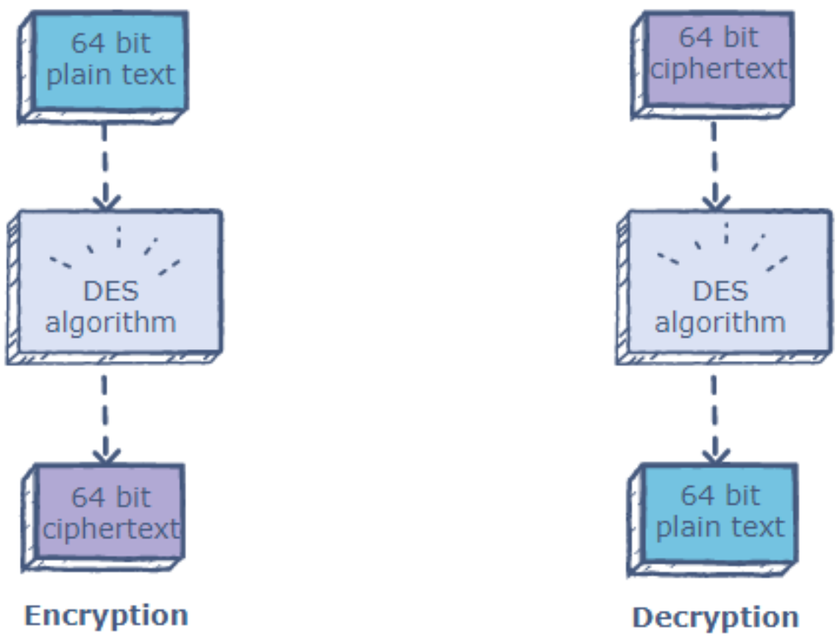
DES algorithm

Data Encryption Standard (DES) is a block cipher algorithm that takes plain text in blocks of 64 bits and converts them to ciphertext using keys of 48 bits. It is a symmetric key algorithm, which means that the same key is used for encrypting and decrypting data.

But in this case we will use 16 bytes key and 512 bytes for input.



Now let's go to the steps:

1. We need to import some packages:
    - Crypto.Chipher:
        - This package allow us to import our DES algorithm and becuse we want to work with 16 bytes key the we use DES3
    - Crypto.Random:
        - This package allow us to generate randome bytes and after that we do some process create a key

In [4]:
```
from Crypto.Cipher import DES3
from Crypto.Random import get_random_bytes
```

Create a key

- This is How we are going to genarate a key for algorthm make 16 random bytes then turn them to string and make some optimizations and return 16Byte key*

In [5]:
```
key = get_random_bytes(16)
key = str(key)
key_list = key.split('\\')
key_str = ""
for element in key_list:
    key_str += element

key = key_str[2:18]
```

Create a function to chack the text message

With this function we are going to Check the text message as long we use 16byte key for this algorithm the text should divisible by 16 if it's not we are going to add white space becuse that's How we would'nt change the original message.

In [6]:
```
def check_text(text):
    while (len(text) % 16) != 0 :
        text += " "
    return text
```

Now it's time to get the message from users

In [8]:
```
message = input("Enter the message you want to send: ")
output_message = check_text(message)
```

Create triple_DES chiper

- parameters:
    - key : byte string ---> The secret key to use in the symmetric cipher.It must be 16 or 24 bytes long. The parity bits will be ignored.
    - keywords:

        1. mode ---> *MODE_** constant, The chaining mode to use for encryption or decryption.Default is `MODE_ECB`.
        2. IV ---> byte string ---> The initialization vector to use for encryption or decryption. It is ignored for `MODE_ECB` and `MODE_CTR`. For `MODE_OPENPGP`, IV must be `block_size` bytes long for encryption and `block_size` +2 bytes for decryption (in the latter case, it is actually the *encrypted* IV which was prefixed to the ciphertext). For all other modes, it must be `block_size` bytes longs.

Mode

We provide you some links you can check These modes out:

1. Electronic Code Book (ECB)
2. Cipher-Block Chaining (CBC)
3. Cipher FeedBack (CFB)
4. Output FeedBack (OFB)
5. CounTer Mode (CTR)
6. OpenPGP Mode

In [9]:
```
des = DES3.new(key , DES3.MODE_ECB)
encrypted_text = des.encrypt(output_message)
```

In [10]:
```
encrypted_text
```

Out[10]:
```
b'G\xca--x\xfdF\xe0\x97\\\xc1\x7f\xc0\xc1f\xfc\xa2\xb5<\xb8.[\xc8rV6\xc7\x8c\x9c\xcc\xc2I\xe3\x8f%m\xbf\xe8\xe4(\x91\xf04\xd1\x
d7\xf9 L\xe7\x0b\xea\x13d\x8a\xee*\xec\x07r\xf6]\xd5\xec\xde'
```

In [11]:
```
decrypted_text = des.decrypt(encrypted_text)
```

In [12]:
```
decrypted_text
```

Out[12]:
```
b'slm khoobi? chelhabar man payame keyli mohemi baraye shoma daram'
```

In [13]:
```
key
```

Out[13]:
```
'7c}qx06jxb01A:xe'
```

In [ ]:
```

```