Blowfish Algorithm Example (1)

Complete example of the Blowfish encryption and decryption process using Python. We'll use the **PyCryptodome** library, which provides an implementation of the Blowfish algorithm.

Blowfish Overview

Blowfish is a symmetric block cipher that:

- Uses a variable-length key (**32 to 448 bits**).
- Has a fixed block size of (**8 bytes**, **64 bits**).
- Uses Feistel network structure with (**16 rounds**).

Installation of Required Library

If you haven't already installed **PyCryptodome**, install it using:

```
pip install pycryptodome
```

Encryption & Decryption with Blowfish

Below is a Python script that:

1. Encrypts a plaintext message using Blowfish.
2. Decrypts the ciphertext back to its original form.

Explanation

1. **Key Setup**: We define a secret key (4 to 56 bytes).
2. **Cipher Initialization**: We create a Blowfish cipher in **ECB (Electronic Codebook) mode**.
3. **Padding**: If the plaintext isn't a multiple of 8 bytes, we pad it.
4. **Encryption**: The padded plaintext is encrypted.
5. **Decryption**:
    - The ciphertext is decrypted.
    - Padding is removed to restore the original text.

```python
from Crypto.Cipher import Blowfish
from Crypto.Util.Padding import pad, unpad
import binascii

# Define a secret key (must be between 4 and 56 bytes)
key = b'securekey'

# Initialize Blowfish cipher in ECB mode
cipher = Blowfish.new(key, Blowfish.MODE_ECB)

# Define plaintext message (must be a multiple of 8 bytes)
plaintext = b'Hello123'  # 8 bytes, no padding needed

# If the plaintext length is not a multiple of 8, pad it
padded_plaintext = pad(plaintext, Blowfish.block_size)

# Encrypt the plaintext
ciphertext = cipher.encrypt(padded_plaintext)
print(f"Ciphertext (Hex): {binascii.hexlify(ciphertext).decode()}")
```

```python
# Decryption
decipher = Blowfish.new(key, Blowfish.MODE_ECB)
decrypted_padded_text = decipher.decrypt(ciphertext)

# Remove padding to get original plaintext
decrypted_text = unpad(decrypted_padded_text, Blowfish.block_size)
print(f"Decrypted Text: {decrypted_text.decode()}")
```

Output Example

Ciphertext (Hex): a1b2c3d4e5f67890...
Decrypted Text: Hello123

Blowfish Algorithm Example (2)

Blowfish Overview

1. **blockSize** : 64-bits
2. **keySize** : 32-bits to 448-bits variable size
3. **number of subkeys** : 18 [P-array]
4. **number of rounds** : 16
5. **number of substitution boxes**: 4 [each having 512 entries of 32-bits each]

Blowfish Encryption Algorithm

Step one by one:

Step1: Generation of subkeys:

- 18 subkeys{P[0]…P[17]} are needed in both encryption as well as decryption process and the same subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

P[0] = "243f6a88"
P[1] = "85a308d3"
.
.
.
P[17] = "8979fb1b"

32-bit hexadecimal representation of initial values of sub-keys

P[0]	243f6a88	P[9]	38d01377
P[1]	85a308d3	P[10]	be5466cf
P[2]	13198a2e	P[11]	34e90c6c
P[3]	03707344	P[12]	c0ac29b7
P[4]	a4093822	P[13]	c97c50dd
P[5]	299f31d0	P[14]	3f84d5b5
P[6]	082efa98	P[15]	b5470917
P[7]	ec4e6c89	P[16]	9216d5d9
P[8]	452821e6	P[17]	8979fb1b

- Now each of the subkey is changed with respect to the input key as:

P[0] = P[0] xor 1st 32-bits of input key
P[1] = P[1] xor 2nd 32-bits of input key
.
.
.
P[i] = P[i] xor (i+1)th 32-bits of input key
(roll over to 1st 32-bits depending on the key length)
.
.
.
P[17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)

The resultant P-array holds 18 subkeys that is used during the entire encryption process

Step2: initialise Substitution Boxes:

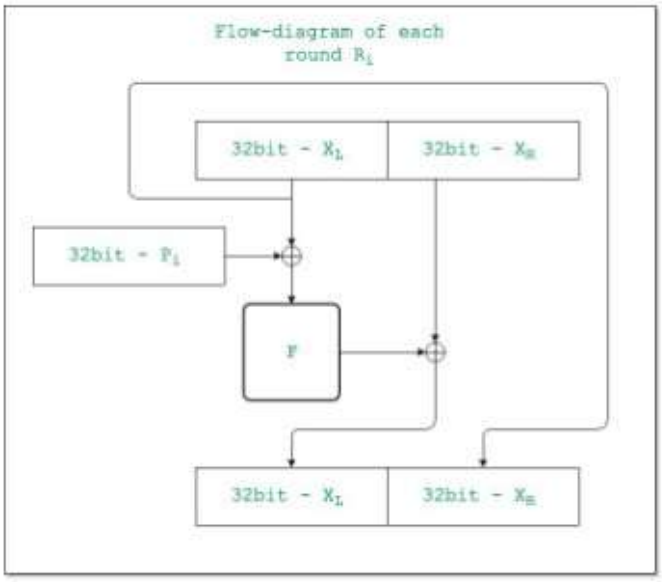
- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption as well as decryption process with each S-box having 256 entries {S[i][0]…S[i][255], 0&lei&le4}, where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array.
  Find the **s-boxes** here: (https://github.com/Ray784/Blowfish-S-boxes).

Step3: Encryption:

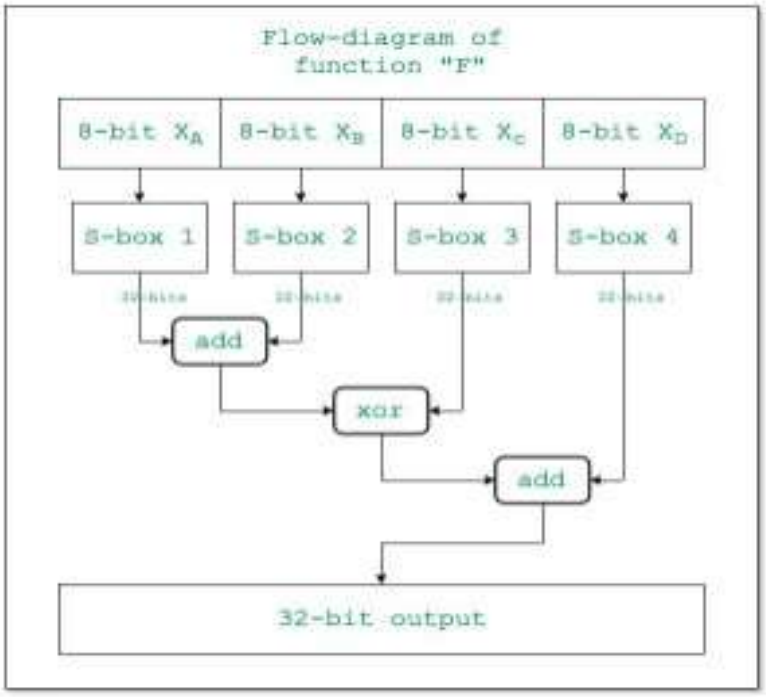
- The encryption function consists of two parts:

a. Rounds:

The encryption consists of 16 rounds with each round (Ri) taking inputs the plainText (P.T.) from previous round and corresponding subkey (Pi).
The description of each round is as follows:



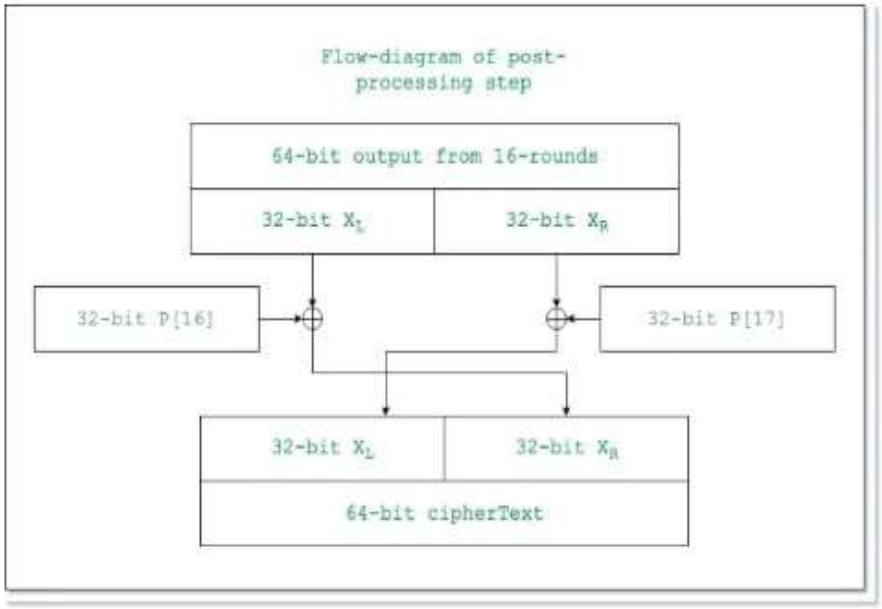
The description of the function ” F ” is as follows:



Here the function “add” is addition modulo 2^32.

b. Post-processing:

The output after the 16 rounds is processed as follows:



Encryption Code from File2

Output

subkey 1: 8e846390
subkey 2: a295c40e
subkey 3: b9a28336
subkey 4: 2446bf99
subkey 5: 0eb2313a
subkey 6: 0ea9fd0d
subkey 7: a295f380
subkey 8: cb78a054
subkey 9: ef9328fe
subkey 10: 1fe6dfaa
subkey 11: 14ef6fd7
subkey 12: 13dfc0b1
subkey 13: 6a1720af
subkey 14: ee4a9c00
subkey 15: 953fdcad
subkey 16: 9271c5ca
subkey 17: 38addcc1
subkey 18: ae4f37c6

----- Encryption -----

round 0: 77b3ba639cb0353b
round 1: 0cc7d63fd5267e6d
round 2: c799728ab5655509
round 3: 69612395e3dfcd13
round 4: f3f5b74b67d312af
round 5: 52023d4efd5c4a46
round 6: 5b785180f097cece
round 7: cc946d119000f1d4
round 8: 6af47a4b230745ef
round 9: 9fb82cc57512a5e1
round 10: 1106c1ab8b574312
round 11: 7d7a616502d9011a
round 12: 81e9ce71176d41ca
round 13: 9727e50a6fa35271
round 14: eb761e34021839a7
round 15: 0599d9367907dbfe

Cipher Text: d748ec383d3405f7

Blowfish Decryption Algorithm

The decryption process is similar to that of encryption and the subkeys are used in reverse{P[17] – P[0]}. The entire decryption process can be elaborated as:

Step one by one:

Step1: Generation of subkeys:

- 18 subkeys{P[0]…P[17]} are needed in decryption process.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

```
P[0] = "243f6a88"
P[1] = "85a308d3"
.
.
.
P[17] = "8979fb1b"
```

Note: See encryption for the initial values of P-array.

- Now each of the subkeys is changed with respect to the input key as:

```
P[0] = P[0] xor 1st 32-bits of input key
P[1] = P[1] xor 2nd 32-bits of input key
.
.
.
P[i] = P[i] xor (i+1)th 32-bits of input key
(roll over to 1st 32-bits depending on the key length)
.
.
.
P[17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)
```

The resultant P-array holds 18 subkeys that is used during the entire encryption process

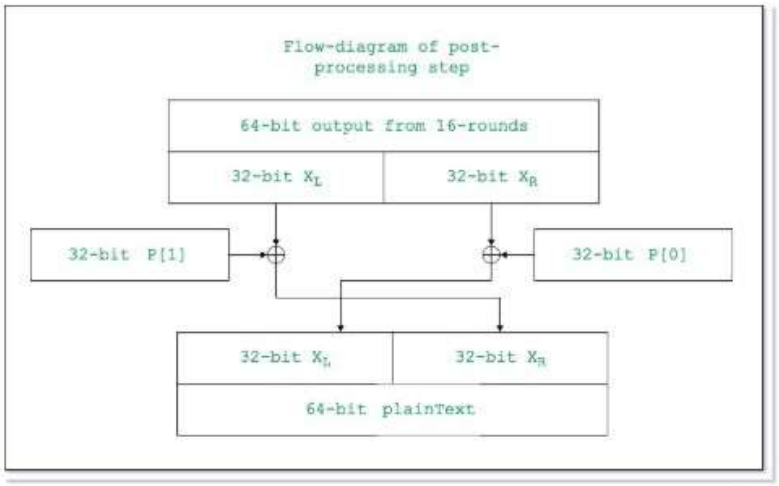
Step2: initialize Substitution Boxes:

- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption as well as decryption process with each S-box having 256 entries {S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array.
  Find the **s-boxes** here: (https://github.com/Ray784/Blowfish-S-boxes).

Step3: Decryption:

- The Decryption function also consists of two parts:

  1. **Rounds:**
     The decryption also consists of 16 rounds with each round (Ri) (as explained above) taking inputs the cipherText (C.T.) from previous round and corresponding subkey (P[17-i])
     (i.e for decryption the subkeys are used in reverse).
  2. **Post-processing:**
     The output after the 16 rounds is processed as follows:



Decryption Code from File2

Output

subkey 1: 8e846390
subkey 2: a295c40e
subkey 3: b9a28336
subkey 4: 2446bf99
subkey 5: 0eb2313a
subkey 6: 0ea9fd0d
subkey 7: a295f380
subkey 8: cb78a054
subkey 9: ef9328fe
subkey 10: 1fe6dfaa
subkey 11: 14ef6fd7
subkey 12: 13dfc0b1
subkey 13: 6a1720af
subkey 14: ee4a9c00
subkey 15: 953fdcad
subkey 16: 9271c5ca
subkey 17: 38addcc1
subkey 18: ae4f37c6

----- Decryption -----

round 17: 3ab5e5667907dbfe
round 16: fdd297bb021839a7
round 15: 82529d676fa35271
round 14: ec939d1a176d41ca
round 13: e14063bd02d9011a
round 12: 66cd65508b574312
round 11: 37e82a387512a5e1
round 10: 8fe62e7e230745ef
round 9: 1f04e6309000f1d4
round 8: 3624ea12f097cece
round 7: c546e12ffd5c4a46
round 6: ed76301e67d312af
round 5: bbd76433e3dfcd13
round 4: f160c1f4b5655509
round 3: 2512b60dd5267e6d
round 2: 6f86e1389cb0353b

Plain Text: 123456abcd132536