Nanjing University of Information Science & Technology

Experiment (Internship) Report

|  |  |  |  |
| --- | --- | --- | --- |
| 202283890036 | 黄家睿(Jerry Huang) | Internet of Things | Zhiguo Qu(瞿治国） |

Implementation and Analysis of the Feistel Encryption Algorithm

1．Experimental Purpose：

1. Understanding the Feistel encryption and decryption algorithm；
2. Implement the Feistel encryption and decryption algorithm.
3. Test its performance (time, avalanche effect, etc.).

2．Experimental content：

1. Implement the Feistel encryption and decryption algorithm.；
2. Analyze its avalanche effect.
3. Experimental steps
   1. Encode the plaintext into binary

import random

def string\_to\_binary(input\_string):

"""

将字符串转换为二进制形式的字符串

"""

return ''.join(format(ord(char), '08b') for char in input\_string)

def generate\_random\_binary(length):

"""

生成随机的二进制数组

"""

return ''.join(random.choice('01') for \_ in range(length))

def round\_function(binary1, binary2):

"""

对两个二进制字符串进行按位与计算

"""

return ''.join('1' if b1 == '1' and b2 == '1' else '0' for b1, b2 in zip(binary1, binary2))

def binary\_xor(binary1, binary2):

"""

对两个二进制字符串进行按位异或计算

"""

return ''.join('1' if b1 != b2 else '0' for b1, b2 in zip(binary1, binary2))

def binary\_to\_string(binary\_string):

"""

将二进制字符串转换回可读的字符

"""

*# 确保二进制字符串长度是8的倍数*

if len(binary\_string) % 8 != 0:

raise ValueError("二进制字符串的长度必须是8的倍数")

*# 按每8位切分并转换为字符*

characters = [

chr(int(binary\_string[i:i + 8], 2)) for i in range(0, len(binary\_string), 8)

]

return ''.join(characters)

* 1. Implementation of encryption

def feistel(data, number\_of\_round):

data\_bin = string\_to\_binary(data)

random\_key = generate\_random\_binary(int(len(data\_bin)//2))

right, left = data\_bin[len(data\_bin)//2:], data\_bin[:len(data\_bin)//2:]

ciper\_book = []

for i in range(number\_of\_round):

right\_new = round\_function(right, random\_key)

left, right= right, binary\_xor(left, right\_new)

ciper\_book.append(right\_new)

ciptertext\_bin =left + right

ciptertext = binary\_to\_string(ciptertext\_bin)

return ciptertext, ciper\_book

* 1. Implementation of decryption

def feistel\_discode(cipertext, ciper\_book):

cipertext\_bin = string\_to\_binary(cipertext)

right, left = cipertext\_bin[len(cipertext\_bin)//2:], cipertext\_bin[:len(cipertext\_bin)//2]

for i in range(len(ciper\_book)):

left, right = binary\_xor(right, ciper\_book[len(ciper\_book)-i-1]), left

plaintext\_bin =left + right

plaintext = binary\_to\_string(plaintext\_bin)

print(f"The palintext of the ciptertext is: {plaintext}")

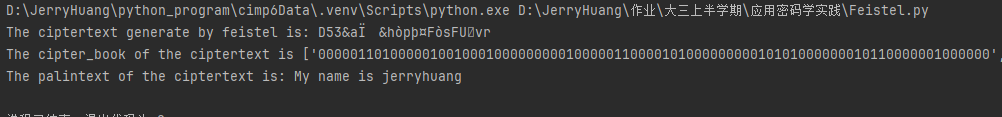
return plaintext

* 1. The output result

plaintext = 'My name is jerryhuang'

cipertext ,ciper\_book= feistel(plaintext, 9)print(f"The ciptertext generate by feistel is: {cipertext}")print(f"The cipter\_book of the ciptertext is {ciper\_book}")

feistel\_discode(cipertext, ciper\_book)



We have obtained the correct decryption result.

* 1. Description and analysis of the avalanche effect

As can be seen from the above image, our input is: "My name is JerryHuang".

The output is：“D53&aÏ&hòpþ¤FòsFUvr”

We modify our input to be.：“My name is not jerryHuang”

The output we get is：“Mk"¦umg°ÁV\_x0016\_îu Fä2rPb÷2f”

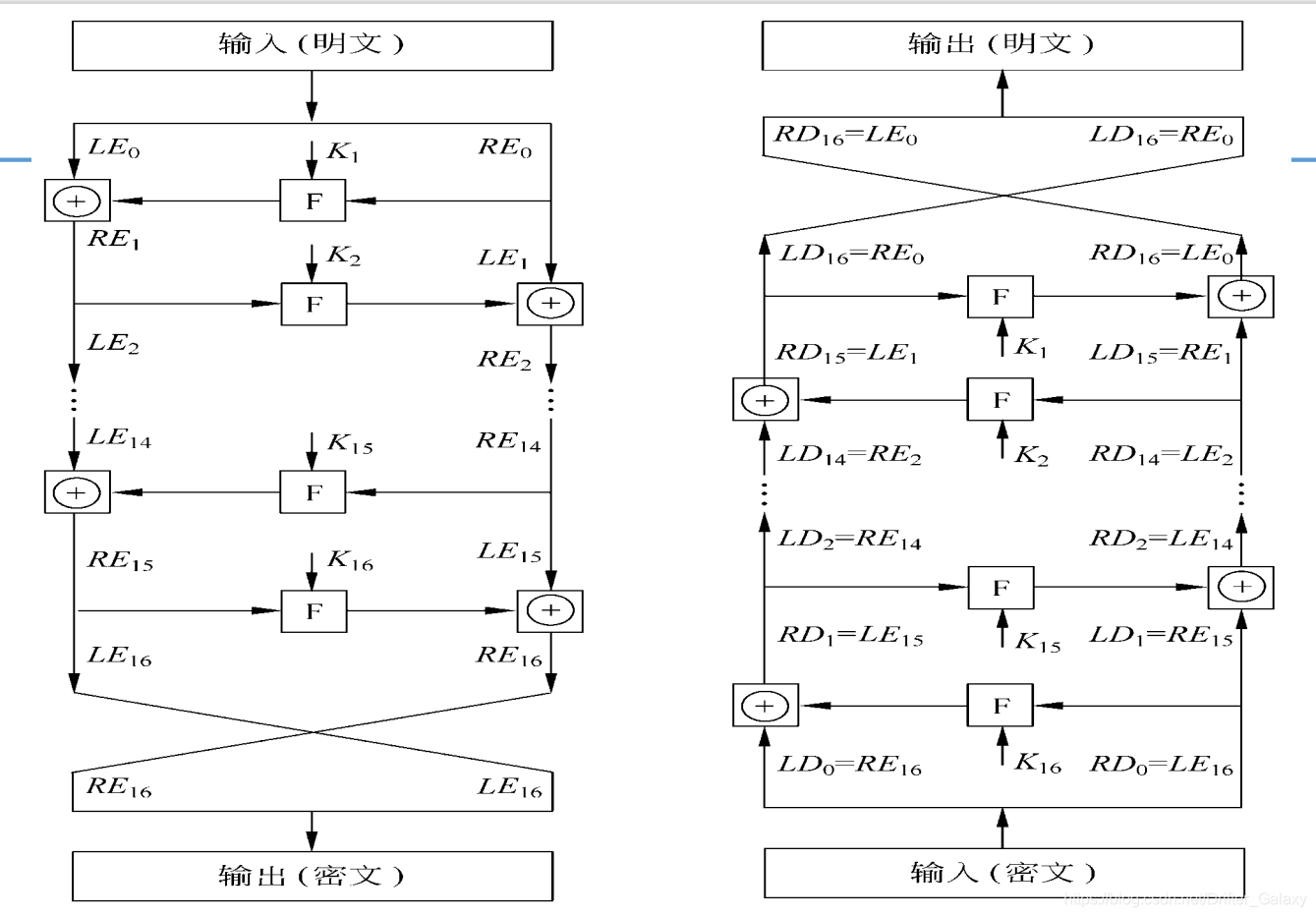
Therefore, we believe that this encryption algorithm has a strong avalanche effect.

1. Experimental Analysis and Summary
   1. The implementation process of the Feistel encryption algorithm

The input is a plaintext block of length 2w and a key K. Each plaintext block is divided into two halves, L and R. After n rounds of iteration, the two halves are combined to produce the ciphertext block. The input to the ith round is a function of the output from the previous round.

Where is the subkey used in the ith round, obtained from the encryption key K. Generally, the subkeys for each round are different from each other and also different from Ki.。

In this experiment, we set the F() function to the AND operation of R i−1 and 𝐾𝑖.



* 1. The process of generating keys

The generation of the key is set to produce a binary string of the same length as the input characters, and with each turn of the wheel, a new binary string is generated and stored individually in a password book used to save the randomly generated keys.

* 1. How to encode the input to make it conform to binary?

In this experiment, many methods are set up, including the implementation of conversion between string type and binary characters, and the conversion between binary characters and string types with binary characters grouped by 8.

* 1. Decryption process

The Feistel decryption process is essentially the same as the encryption process, with the algorithm using the ciphertext as input but using the subkey K\_i order in reverse to the encryption process, i.e., the first round uses K\_n , the second round uses K\_n-1, ..., and the last round uses K1 . This feature ensures that decryption and encryption can use the same algorithm.