南京信息工程大学 实验（实习）报告

|  |  |  |
| --- | --- | --- |
| 202283890036 | 黄家睿 | Internet of Things |

Feistel加密算法的实现与分析

1．实验目的：

1. 理解Feistel加解密算法；
2. 实现Feistel加解密算法
3. 对其性能进行测试（时间、雪崩效应等）。

2．实验内容：

1. 实现Feistel加解密算法；
2. 对其雪崩效应进行分析。
3. 实验步骤
   1. 对明文进行二进制编码

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. 加密的实现

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. 解密的实现

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}")

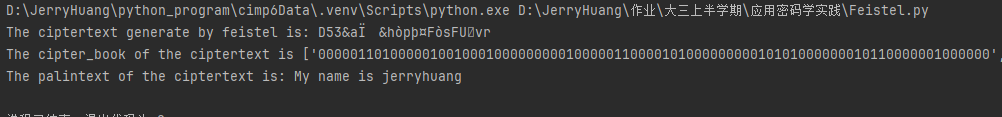
return plaintext

* 1. 输出的结果

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)



我们得到了正确的解密结果。

* 1. 雪崩效应的描述分析

从上图可见，我们的输入为：“My name is JerryHuang”。

输出为：“D53&aÏ&hòpþ¤FòsFUvr”

我们修改我们输入为：“My name is not jerryHuang”

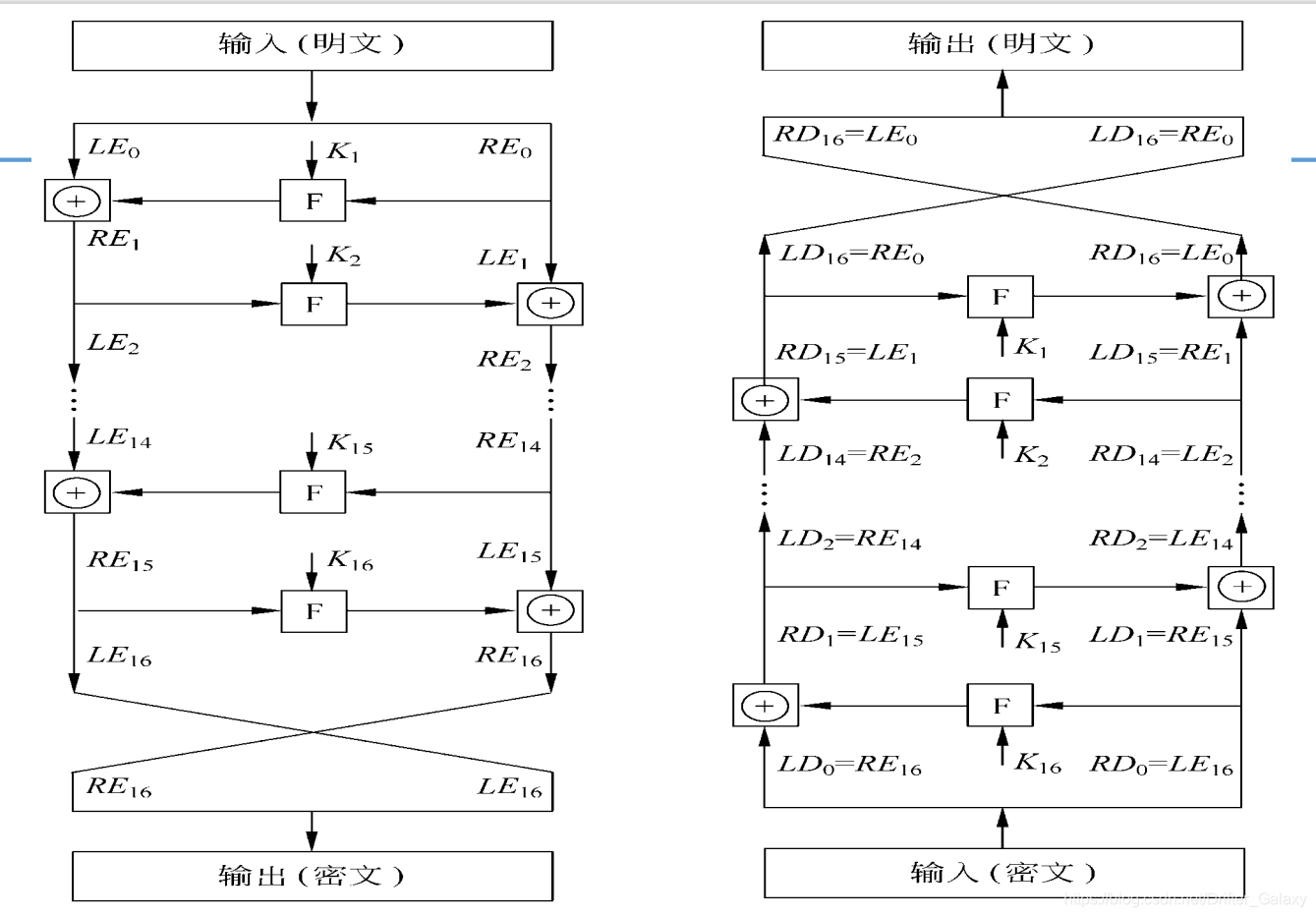
得到的输出为：“Mk"¦umg°ÁV\_x0016\_îu Fä2rPb÷2f”

因此我们认为这个加密算法具有很强的雪崩效应。

1. 实验分析和总结
   1. Feistel加密算法的实现过程
      1. 输入是分组长为2w的明文和一个密钥K。将每组明文分成左右两半IMG_256和IMG_257，在进行完n轮迭代后，左右两半再合并到一起以产生密文分组。第 i 轮迭代的输入为前一轮输出的函数。

其中IMG_256是第i轮用的子密钥，由加密密钥K得到。一般地，各轮子密钥 彼 此不同而且与K也不同。

在本次的实验中，我们把F()函数设置为和的与操作。



* 1. 密钥的生成过程

密钥的生成设置为生成和输入的字符长度一样的二进制字符串，每进行一次滚轮，就生成一个新的二进制字符串，并单独储存在一个用来保存随机生成密钥的密码本里面。

* 1. 对于输入的如何进行编码使其符合二进制

在本实验中设置很多方法，包括实现string类型和二进制字符之间的转化，二进制字符以8为为一组实现二进制字符和string类型的转换等。

* 1. 解密 过程

Feistel解密过程本质上和加密过程是一样的，算法使用密文作为输入但使用子密钥IMG_256的次序与加密过程相反，即第1轮使用IMG_257，第2轮使用IMG_258，……，最后一轮使用IMG_259。这一特性保证了解密和加密可采用同一算法。