南京信息工程大学 实验（实习）报告

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

AES加密算法的实现与分析

1．实验目的：

1. 理解AES加解密算法；
2. 实现AES加解密算法,并分析其性能

2．实验内容：

1. 实现AES加解密算法；
2. 对加密结果进行分析。
3. 实验步骤
4. 先前导入的库

AES实现较为复杂，可以使用Python中的Crypto第三方库来实现，这是一个经典的用于密码学算法的Python第三方库。在PowerShell中输入***pip install pycryptodome***即可快速安装AES加密实现

导入成果之后需要import下列的方法。

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

import base64

1. AES加密的实现

def aes\_encrypt(data, key):

*# AES要求密钥长度为16、24或32字节*

key = key.encode('utf-8') *# 转为字节*

cipher = AES.new(key, AES.MODE\_CBC) *# 使用CBC模式创建AES对象*

*# 对数据进行填充，使其长度为16的倍数*

data = data.encode('utf-8') *# 转为字节*

padded\_data = pad(data, AES.block\_size) *# 填充*

*# 加密数据*

encrypted\_data = cipher.encrypt(padded\_data)

*# 返回加密后的数据，包含IV（初始向量）*

*# 将IV和加密数据一起返回，方便解密时使用*

return base64.b64encode(cipher.iv + encrypted\_data).decode('utf-8')

1. AES解密的实现

def aes\_decrypt(encrypted\_data, key):

encrypted\_data = base64.b64decode(encrypted\_data) # 解码Base64数据

# 提取IV（前16个字节是IV）

iv = encrypted\_data[:16]

encrypted\_data = encrypted\_data[16:] # 剩下的是加密数据

# 创建AES对象，并使用提取的IV

key = key.encode('utf-8') # 转为字节

cipher = AES.new(key, AES.MODE\_CBC, iv) # 使用CBC模式创建AES对象

# 解密数据，并移除填充

decrypted\_data = unpad(cipher.decrypt(encrypted\_data), AES.block\_size)

return decrypted\_data.decode('utf-8')

1. 加密解密效果呈现

*# 示例使用*

key = "JerryHuangandyou" *# 密钥需要是16、24或32字节（16字节为AES-128）*

data = "This is a secret message"

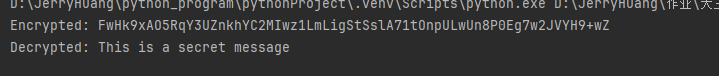
*# 加密*

encrypted = aes\_encrypt(data, key)print("Encrypted:", encrypted)

*# 解密*

decrypted = aes\_decrypt(encrypted, key)print("Decrypted:", decrypted)

得到的结果：



1. 实验分析和总结
2. AES和Feistel算法之间区别

Feistel网络：Feistel网络的基本结构是将数据块分成两半，并对其中一半进行加密操作，然后交换这两部分。每一轮的加密操作都是在数据的一个部分上进行，通过多个轮（通常是16轮）来实现加密。在解密时，Feistel网络具有对称性，即加密过程和解密过程相同，只是密钥的使用顺序相反。

AES（替代-置换网络）：AES使用替代（Substitution）和置换（Permutation）操作的组合，所有的加密步骤是对整个数据块进行的。AES的每一轮操作涉及字节替代（SubBytes）、行移位（ShiftRows）、列混淆（MixColumns）以及轮密钥加（AddRoundKey）。这些步骤应用到整个数据块（128位），而不是分成两部分处理。

Feistel网络：Feistel网络的加密过程相对简单，主要是通过替换和置换操作实现加密。每一轮的操作较为直接，通常通过基本的异或、置换和简单的加密函数来实现。

AES：AES的每一轮操作更加复杂。除了字节替代和行移位，AES还引入了列混淆操作（MixColumns），这使得AES在混淆和扩散方面比Feistel网络更为强大。列混淆操作使得每一列的字节相互影响，大大增强了数据的混淆性。

1. AES改进的地方

替代-置换结构：AES的替代-置换网络设计，使得每一轮的加密操作更加复杂，增强了扩散性和混淆性，从而提供更强的安全性。

列混淆（MixColumns）：与Feistel网络的交换操作不同，AES在每一轮中通过列混淆使数据更为复杂，这增强了AES对抗攻击的能力。

更高的抗差分攻击能力：AES通过引入更加复杂的S盒和非线性变换，有效增强了对差分攻击的防御能力。

AES在设计时充分考虑了抗差分攻击和线性攻击，采用了复杂的字节替代、行移位、列混淆和轮密钥加操作，这使得AES更难受到攻击。AES-256的密钥长度提供了更高的抗暴力破解能力。

1. AES的实现的具体算法过程

密钥扩展算法由密钥调度算法和轮密钥生成算法组成。密钥调度算法根据输入密钥的长度选择不同的配置，然后将输入密钥分成多个字，并对字进行处理。轮密钥生成算法通过对上一个轮密钥进行变换，并结合密钥调度算法，生成新的轮密钥。将待加密的明文（16字节数据）排列成4x4的矩阵。然后，将16字节的密钥也排列成4x4的矩阵。这两个矩阵进行异或运算（每个字节对应异或），从而完成轮密钥加。将轮密钥加后产生的每一个字节用十六进制表示，以十六进制的第一个数字为行，第二个数字为列，在S盒表中查找对应的数字，用这个数字来代替原先的数字。将字节替换后的矩阵进行行移位。具体来说，第一行保持不变，第二行循环左移1个字节，第三行循环左移2个字节，第四行循环左移3个字节。对行移位后的数据进行列混淆，这涉及到在域GF(2^8)上的算术运算。重复轮密钥加、字节替换、行移位和列混淆的步骤：根据密钥的长度（128位、192位或256位），AES算法会进行不同数量的轮数（分别为10轮、12轮和14轮）。在每一轮中，都会执行轮密钥加、字节替换、行移位和列混淆这四个步骤。最终，就完成了AES的加密过程。

