专业课程设计报告

（ 2022/ 2023 学年 第 一 学期）

题　　目：U 盘加密系统

专　　业	信息安全
学 生 姓 名	任远哲
班 级 学 号	B190307-B19031614
指 导 教 师	张洁
指 导 单 位	计算机学院、软件学院、网络空间安全学院
日　　期	**2022.10.24-2022.11.06**

课程目标	评价准则	计分（每项10分）
课程目标1：通过课程设计，培养学生综合应用信息安全、计算机技术等领域专业知识的技能。（20分）	1、能够掌握信息安全的相关基础知识，并能够针对求解的工程问题，收集资料进行合理的分析与设计。	
	2、从软件的分析、设计到编制调试，结合计算机网络理论知识、编程语言以及程序设计的方法，能够解决一个和信息安全相关的问题。	
课程目标2：解决信息安全领域复杂工程问题的实践创新能力。（20分）	3、通过调研，能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现。	
	4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便。	
课程目标3：文献调研与资料收集能力，问题发现、研究、分析与解决能力。（10分）	5、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能。	
课程目标4：培养工程工具运用能力，能够利用仿真软件或实验系统对信息安全系统进行模拟和预测，并理解仿真软件或实验硬件的局限性。（20分）	6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，理解相关算法，对求解的工程问题进行有效建模和求解。	
	7、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试。	
课程目标5：分组完成一次项目设计与开发的全过程，组内成员通过讨论和交流解决课程设计中的难题，能在实验报告中准确阐述课程设计的内容，能够清晰陈述观点和回答问题。（30分）	8、组内成员之间有一定的团队合作，互通有无。	
	9、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范。	
	10、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的工程基础知识具有较好的理解和掌握。	
专业课程设计能力测评总分		

指导教师：_____　　_____年____月____日

备注：加上平时成绩之后换算的总评成绩及等级（优、良、中、及格等）：

U盘加密系统

一、课题内容和要求

在学习了程序设计和信息安全专业基础课程的基础上，综合运用所学的方法和理论，编写对u盘进行完全加密或者部分加密的程序，从软件的分析、设计到编制调试，要求学生把信息安全、编程语言以及程序设计的方法等结合起来，并通过课程设计提高学生的查阅资料、自学和独立分析问题与解决问题的能力，加深对理论知识的理解，利用仿真软件或实验系统对信息安全系统进行模拟和预测，验证并掌握信息安全学科中重要的基础理论与方法，为今后从事相关方面的工作奠定基础。

题目的具体要求如下：

1.对u盘进行部分或者全部加密；

2.采用对称加密算法进行u盘数据加密；

3.对称加密算法的密钥要用口令或者公钥加密；

4.查阅相关资料，深入理解分组加密AES和公钥加密RSA的原理；

5.程序具有图形化用户界面，输出美观；

6.可根据自己能力，在完成以上基本要求后，对程序功能进行适当扩充；

7.撰写报告，对所采用的算法、程序结构和主要函数过程以及关键变量进行详细的说明；对程序的调试过程所遇到的问题进行回顾和分析，对测试和运行结果进行分析；总结软件设计和实习的经验和体会，进一步改进的设想；

8.提供关键程序的清单、源程序及可执行文件和相关的软件说明。

二、需求分析和总体设计

2.1 本课题的主要功能包括：

（1）输入路径和对称密钥，使用AES算法加密U盘（文件夹）

（2）输入路径和对称密钥，使用AES算法解密（1）中加密后U盘（文件夹）

（3）输入RSA公钥e和n，使用RSA算法加密（1）中的对称密钥

（4）输入RSA私钥d和n，使用RSA算法解密（3）中加密的对称密钥

（5）自动生成RSA参数（e，d，n）

本人是组长，完成gui界面的编写以及功能(1),(2)，并整合所有代码

绘制用例图，如图 1 所示：

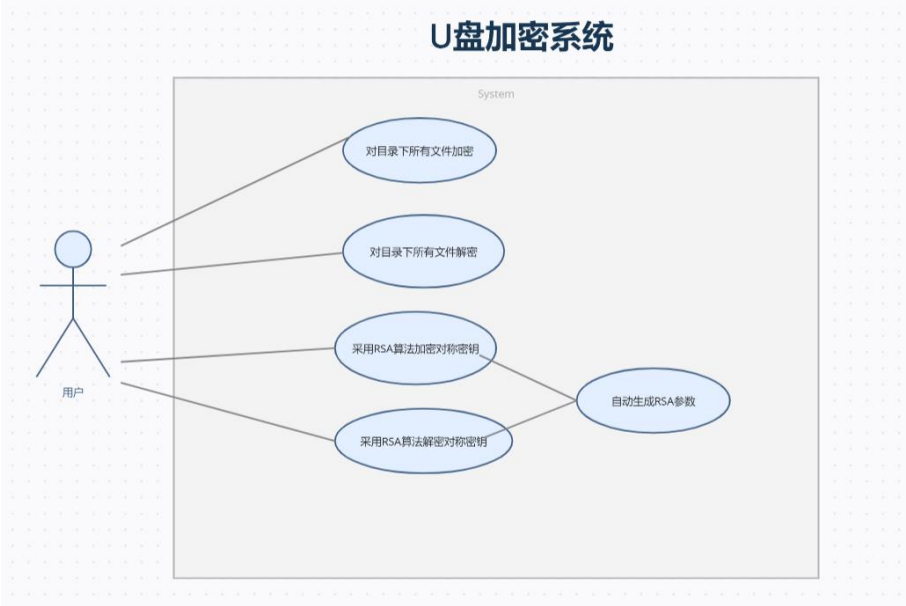

图 1 用例图

2.2 本课题的数据表单设计

本加密系统使用的一些参数：

（1）filepaths：存储根目录及其子目录下所有文件的路径

（2）AESE.blk = blk ：每一分组的明文 list[int]

AESE.key = key ：每一分组的密钥 list[int]

AESE.Nr = Nr ：对每一分组加密的轮次

（3） e：RSA 的公钥

d：RSA 的私钥

n：RSA 的模

2.3 本课题的体系结构设计

因为功能并不复杂，所以本系统采用“调用和返回体系结构”风格中的“主程序/子程序体系结构”。 这种的程序结构将功能分解成一个控制层次，其中主程序调用一组程序构件，这些程序构件又去调用别的程序构件。编程是基于面向过程的思想，加密算法模块：对文件 AES 和对字符串 RSA 由自己编写，GUI 界面则调用了 Python 中的 Tkinter 库。

2.4 本课题中的主要交互行为建模，绘制序列图，如图 2 所示：

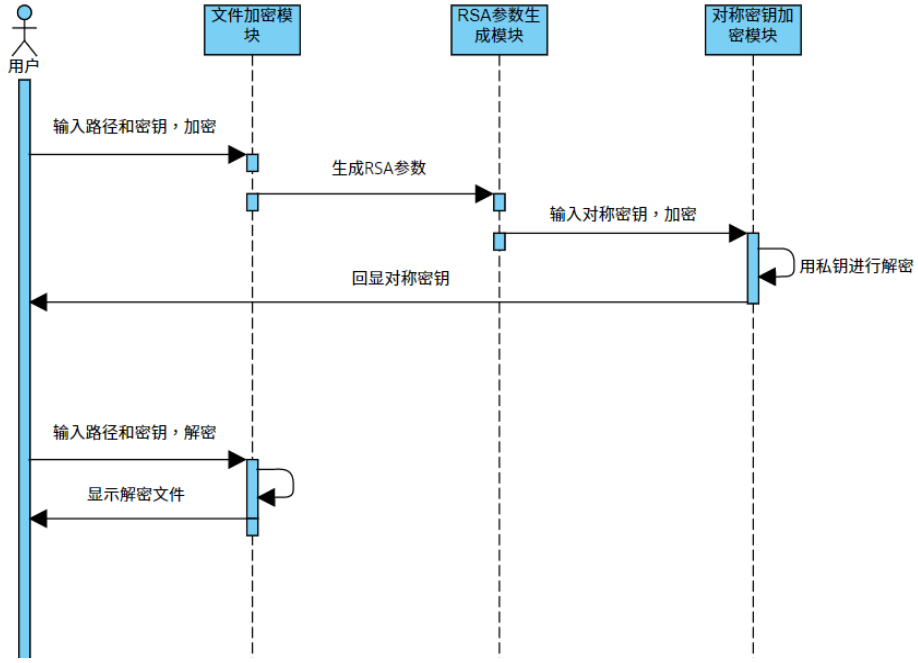

图 2 序列图

首先将一个文件读取成字节流，再按 16 字节为一个分组进行划分，最后对每个分组分别调用 AES 算法加密。RSA 算法是公钥加密算法，我们首先生成 RSA 的公私钥等参数，这里我们加密的对象是字符串（AES 算法中的对称密钥），将每个字符转化成其 ascill 码（int 型）以用 RSA 算法分别加密。

2.5 本课题的主要功能界面设计

本 U 盘加密系统总共设计了 5 个子页面。页面 1 为“Encrypt package via AES”，即允许用户输入路径和对称密钥，使用 AES 算法加密 U 盘（文件夹）；页面 2 为“Decrypt package via AES”，即允许用户输入路径和对称密钥，使用 AES 算法解密（1）中加密后 U 盘（文件夹）；页面 3 为“Encrypt AES Key via RSA”，即允许用户输入 RSA 私钥 d 和 n，使用 RSA 算法解密（3）中加密的对称密钥；页面 4 为“Decrypt AES Key via RSA”，即允许用户输入 RSA 私钥 d 和 n，使用 RSA 算法解密（3）中加密的对称密钥；页面 5 为“RSA parameters”，即允许用户自动生成 RSA 参数（e，d，n）。

因为篇幅限制，下面仅展示页面 1 的图形化界面，如图 3 所示：

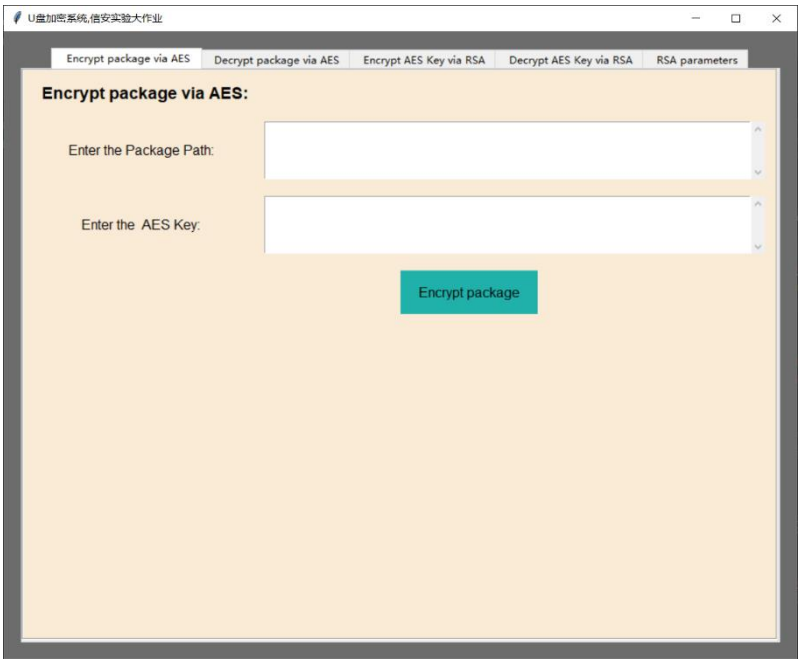

图 3 界面图

三、相关功能模块详细设计

该系统主要分为三个模块：gui.py，aes_file.py 和 rsa.py

（1）gui.py ： GUI 界面则调用了 Python 中的 Tkinter 库进行可视化，分为五个界面。每个界面获取 ScrolledText 框中输入的参数，通过 Button 键调用 aes_file.py 和 rsa.py 中的核心算法进行处理，并回显计算结果。

（2）aes_file.py ： 该模块负责实现对文件夹中的文件进行 AES 加密的操作。 首先通过函数 all_files_path(rootDir) 对根目录 rootDir 下的所有文件进行递归遍历，然后再依次以二进制的形式读取这些文件。每一个文件按 16 字节为一个分组进行划分，然后进行分组加密或解密。

AES 密码学中的高级加密标准（Advanced Encryption Standard，AES），又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。这个标准用来替代原先的 DES（Data Encryption Standard），已经被多方分析且广为全世界所使用。

AES 加解密算法流程

AES 的区块长度固定为 128 位（16 字节）。密钥长度则可以是 128 bit，192 bit 或

256 位 bit，换算成字节长度，就是密码必须是 16 个字节，24 个字节，32 个字节（在这个实验中我们选择 16 字节的版本）。

AES 的加密模式有以下几种：电码本模式(ECB)，密码分组链接模式(CBC)，计算器模式(CTR)，密码反馈模式(CFB)，输出反馈模式(OFB)。在本次实验中我们选择电码本模式(ECB)，即将需要加密的消息按照块密码的块大小被分为数个块，并对每个块进行独立加密。

ECB 工作模式要求输入明文长度必须是块（16 byte）长度的整数倍，因此信息必须填充至满足要求。AES 支持的填充模式主要分为 PKCS7 和 ZerosPadding。在本次实验中我们选择 ZerosPadding 填充模式,即全部填充 0x00,无论缺多少全部填充 0x00,已经是 128bits 倍数仍要填充一个全 0 的块。

AES 加密算法涉及 4 种操作：字节替代（SubBytes）、行移位（ShiftRows）、列混淆（MixColumns）和轮密钥加（AddRoundKey），如下图 4 所示：

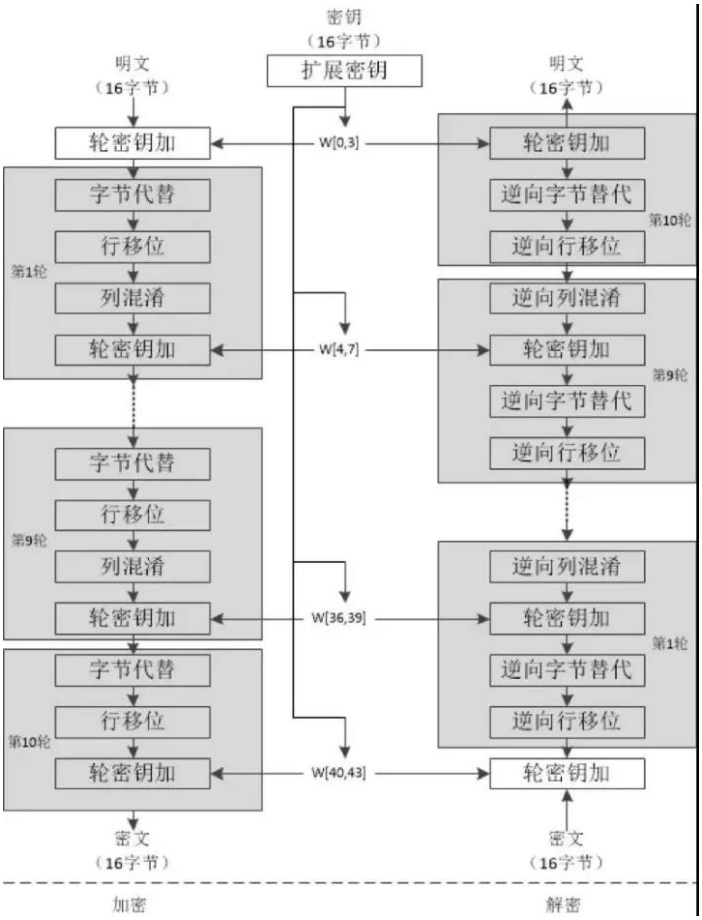

图 4 AES 流程图

（2.1）字节替代

字节代替的主要功能是通过 S 盒完成一个字节到另外一个字节的映射。AES 定义了一个 S 盒和一个逆 S 盒，用于提供密码算法的混淆性。S 和 S-1 分别为 16x16 的矩阵，完成一个 8 比特输入到 8 比特输出的映射，输入的高 4-bit 对应的值作为行标，低 4-bit 对应的值作为列标。

（2.2）行移位

行移位是一个 4x4 的矩阵内部字节之间的置换，用于提供算法的扩散性。

2.2.1) 正向行移位，如图 5

正向行移位用于加密，其原理图如下。其中：第一行保持不变，第二行循环左移 8 比特，第三行循环左移 16 比特，第四行循环左移 24 比特。

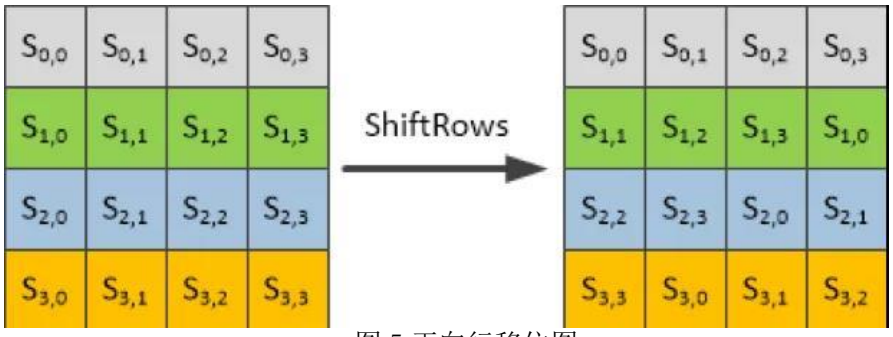

图 5 正向行移位图

2.2.2) 逆向行移位

逆向行移位即是相反的操作，即：第一行保持不变，第二行循环右移 8 比特，第三行循环右移 16 比特，第四行循环右移 24 比特。

（2.3）列混淆

2.3.1) 正向列混淆，如图 6

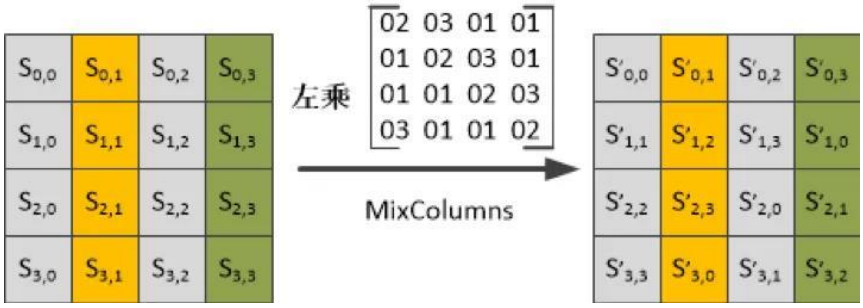

图 6 正向列混淆图

2.3.2) 逆向列混淆，如图 7

逆向列混淆的原理图如下：

图 7 逆向列混淆图

可以看出说明两个矩阵互逆，经过一次逆向列混淆后即可恢复原文，如图 8

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{bmatrix}$$

图 8 两个矩阵互逆图

根据矩阵的乘法，在列混淆的过程中，每个字节对应的值只与该列的 4 个值有关系。假设某一列的值如下图 9，运算过程如下：

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} C9 \\ 6E \\ 46 \\ A6 \end{bmatrix} = \begin{bmatrix} DB \\ 37 \\ 94 \\ ED \end{bmatrix}$$

图 9 混淆运算图

（2.4）轮密钥加

这个操作相对简单，其依据的原理是“任何数和自身的异或结果为 0。加密过程中，每轮的输入与轮子密钥异或一次；因此，解密时再异或上该轮的轮子密钥即可恢复。

密钥扩展：其复杂性是确保算法安全性的重要部分。当分组长度和密钥长度都是 128 位时，AES 的加密算法共迭代 10 轮，需要 10 个子密钥。AES 的密钥扩展的目的是将输入的 128 位密钥扩展成 11 个 128 位的子密钥。AES 的密钥扩展算法是以字为一个基本单位（一个字为 4 个字节），刚好是密钥矩阵的一列。因此 4 个字（128 位）密钥需要扩展成 11 个子密钥，共 44 个字。

（3）rsa.py：该模块主要负责对一个字符串（AES 对称密钥）进行 RSA 加密和解密。我们的做法是将待加密的字符串按其中每个字符的 ascill 码分别加密，中间用’,’隔开。其中公钥 e，私钥 d 和参数 n 可以由我们指定，也可以自动生成。

RSA 加密是一种非对称加密。可以在不直接传递密钥的情况下，完成解密。这能够确保信息的安全性，避免了直接传递密钥所造成的被破解的风险。是由一对密钥来进行加解密的过程，分别称为公钥和私钥。RSA 的安全性依赖于分解两个大素数乘积的实际困难。其具体算法流程如下：

（3.1）选择两个质数 p 和 q，算出他们的乘积 $n=p\times q$，算出对应的欧拉函数 $\varphi(n)$（利用性质 $\varphi(n)=\varphi(p)\times\varphi(q)=(p-1)(q-1)$。

（3.2）选择一个 e，使得 $e<\varphi(n)$并且 e 与 $\varphi(n)$互质。

（3.3）算出 e 的一个相对于 $\varphi(n)$的模反元素 d。(e,n)为公钥，(d,n)为私钥，信息（明文）m 长度小于 n。

（3.4）加密：c =m^e (mod n)

（3.5）解密：m =c^d (mod n)。

四、部分核心代码

4.1 GUI 界面模块的代码

4.1.1 主框架绘制代码，设置标题和其下 5 个子页面，如图 10 所示：

```
#########################################################################
root = Tk()
root.title("U盘加密系统,信安实验大作业")
root.geometry("920x750")
# root.configure(bg='white')
s = ttk.Style(root)
s.configure("TNotebook", tabposition='n',background='#6C6C6C')
s.configure("TFrame",background='#FAEBD7')
notebook = ttk.Notebook(root,padding=20)
notebook.pack(expand=1, fill="both")
frame1 = ttk.Frame(notebook,relief=GROOVE,padding=5)
frame2 = ttk.Frame(notebook,relief=GROOVE,padding=5)
frame3 = ttk.Frame(notebook,relief=GROOVE,padding=5)
frame4 = ttk.Frame(notebook,relief=GROOVE,padding=5)
frame5 = ttk.Frame(notebook,relief=GROOVE,padding=5)


notebook.add(frame1, text='   Encrypt package via AES  ')
notebook.add(frame2, text='   Decrypt package via AES  ')
notebook.add(frame3, text='   Encrypt AES Key via RSA  ')
notebook.add(frame4, text='   Decrypt AES Key via RSA  ')
notebook.add(frame5, text='   RSA parameters  ')
```

图 10 主框架绘制代码图

4.1.2 子页面 1 代码。输入框输入根目录和公钥并对其中所有文件进行加密，如图 11：

```
####################################    page1加密文件夹
Label(frame1,text="  Encrypt package via AES: ",font=('Arial', 14, 'bold'),background='#FAEBD7').grid(padx=5,pady=10,row=0,column=0)

Label(frame1,text="Enter the Package Path: ",font=('Arial', 12),background='#FAEBD7').grid(padx=5,pady=10,row=3,column=0)
package_path_input=ScrolledText(frame1,width=80,height=5)#文件夹路径输入框
package_path_input.grid(padx=5,pady=10,row=3, column=1,sticky="E",columnspan=4)

Label(frame1,text="Enter the  AES Key: ",font=('Arial', 12),background='#FAEBD7').grid(padx=5,pady=10,row=4,column=0)
AES_Key_input=ScrolledText(frame1,width=80,height=5)#AES密钥输入框
AES_Key_input.grid(padx=5,pady=10,row=4, column=1,sticky="E",columnspan=4)


AES_Encrypt_Button=Button(frame1,text=" Encrypt package ",font=('Arial', 12),activebackground='Coral',background='LightSeaGreen',re
AES_Encrypt_Button.grid(ipadx=10,ipady=10,padx=5,pady=10,row=5,column=2)
```

图 11 子页面 1 代码图

由于篇幅限制，子页面 2-5 的代码不展示

4.2 递归遍历根目录所有文件模块，如图 12 所示

```
global filepaths
filepaths=[]
def all_files_path(rootDir):
    filepaths.clear()
    for root, dirs, files in os.walk(rootDir):           # walk方法返回一个三元组，
        for file in files:                  # 遍历文件
            file_path = os.path.join(root, file)          # 拼接文件路径，用于获取文件
            filepaths.append(file_path)             # 将文件路径添加进列表
            #print(file_path)
        for dir in dirs[:-1]:                   # 遍历目录下的子目录
            dir_path = os.path.join(root, dir)                # 获取子目录路径
            all_files_path(dir_path)           # 递归调用
```

图 12 递归遍历根目录代码图

4.3 AES 加密模块

4.3.1 字节替代模块，如图 13 所示：

```
# SubBytes 字节替代,对整个分组进行处理
def SubBytes(self):
    for x in range(16):
        self.blk[x] = self.sbox[self.blk[x]]
```

图 13 字节替代代码图

4.3.2 行移位模块，如图 14 所示：

```
# ShiftRows:行移位,对整个分组进行处理
def ShiftRows(self):
    # 2nd row
    t = self.blk[1]
    self.blk[1] = self.blk[5]
    self.blk[5] = self.blk[9]
    self.blk[9] = self.blk[13]
    self.blk[13] = t
    # 3nd row
    t = self.blk[2]
    self.blk[2] = self.blk[10]
    self.blk[10] = t
    t = self.blk[6]
    self.blk[6] = self.blk[14]
    self.blk[14] = t
    # 4nd row
    t = self.blk[15]
    self.blk[15] = self.blk[11]
    self.blk[11] = self.blk[7]
    self.blk[7] = self.blk[3]
    self.blk[3] = t
```

图 14 行移位模块代码图

4.3.3 列混淆模块，如图 15 所示：

```
# MixColumns: 列混淆,对整个分组进行处理
def MixColumns(self):
    tmp = [0 for t in range(4)]
    xt = [0 for q in range(4)]
    n = 0
    for x in range(4):
        xt[0] = self.xtime(self.blk[n])
        xt[1] = self.xtime(self.blk[n + 1])
        xt[2] = self.xtime(self.blk[n + 2])
        xt[3] = self.xtime(self.blk[n + 3])
        tmp[0] = xt[0] ^ xt[1] ^ self.blk[n + 1] ^ self.blk[n + 2] ^ self.blk[n + 3]
        tmp[1] = self.blk[n] ^ xt[1] ^ xt[2] ^ self.blk[n + 2] ^ self.blk[n + 3]
        tmp[2] = self.blk[n] ^ self.blk[n + 1] ^ xt[2] ^ xt[3] ^ self.blk[n + 3]
        tmp[3] = xt[0] ^ self.blk[n] ^ self.blk[n + 1] ^ self.blk[n + 2] ^ xt[3]
        self.blk[n] = tmp[0]
        self.blk[n + 1] = tmp[1]
        self.blk[n + 2] = tmp[2]
        self.blk[n + 3] = tmp[3]
        n = n + 4
```

图 15 列混淆模块代码图

4.3.4 轮密钥加模块，如图 16 所示：

```
# AddRoundKey 轮密钥加,对整个分组进行处理
def AddRoundKey(self, key):
    x = 0
    k = [0 for m in range(16)]
    for c in range(4):
        for r in range(4):
            k[x] = key[r][c]
            x = x + 1
    for y in range(16):
        self.blk[y] ^= int(k[y])
```

图 16 轮密钥加模块代码图

4.3.5 密钥拓展模块，如图 17 所示：

```
# 密钥拓展
def ScheduleKey(self, w, Nk):
    Rcon = [0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36]
    for r in range(4):
        for c in range(4):
            w[0][r][c] = self.key[r + c * 4]
    for i in range(1, self.Nr + 1, 1):
        for j in range(Nk):
            t = [0 for x in range(4)]
            for r in range(4):
                if j:
                    t[r] = w[i][r][j - 1]
                else:
                    t[r] = w[i - 1][r][3]
            if j == 0:
                temp = t[0]
                for r in range(3):
                    t[r] = self.sbox[t[(r + 1) % 4]]
                t[3] = self.sbox[temp]
                t[0] ^= int(Rcon[i - 1])
            for r in range(4):
                w[i][r][j] = w[i - 1][r][j] ^ t[r]
```

图 17 密钥拓展模块代码图

由于篇幅限制，解密模块的代码不展示

4.4 对 AES 密钥进行 RSA 加密模块

4.4.1 RSA 加密模块，如图 18 所示：

```
def encrypt_text_rsa(e, n, message):
    temp = ''
    # message = input("输入需要加密的密文")
    message = list(map(ord, message))#每个字符转化成ascill 码
    # print('ciphertext数字化:', message)
    ciphertext = []
    for x in message:
        ciphertext.append(pow(x, e, n))
    for x in ciphertext:
        temp = temp + ',' + str(x)

    temp = temp[1:]
    # print(temp)
    return temp
```

图 18 RSA 加密模块代码图

五、软件测试及其结果分析

5.1 单元测试（以递归枚举根路径下的所有文件为例）

```
import os
filepaths=[]
def all_files_path(rootDir):
    filepaths.clear()
    for root, dirs, files in os.walk(rootDir):
        for file in files:                          # 遍历文件
            file_path = os.path.join(root, file)          # 拼接文件路径，用于获取文件绝对路径
            filepaths.append(file_path)                 # 将文件路径添加进列表
            print(file_path)
        for dir in dirs[:-1]:                       # 遍历目录下的子目录
            dir_path = os.path.join(root, dir)                  # 获取子目录路径
            all_files_path(dir_path)                # 递归调用

all_files_path('Z:\TEMP')
```

结果如下图 19 所示：

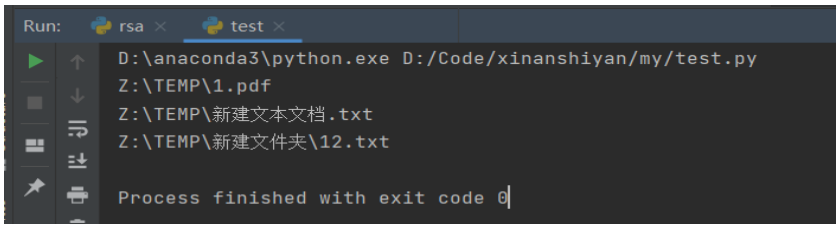

图 19 验证遍历结果图

5.2 集成测试

5.2.1 测试对 U 盘进行加密的功能。

在待加密 U 盘中建立文件 B19031614.txt，内容为 B19031614，如图 20 所示。接着对 U 盘根目录进行加密，如图 21 所示。加密结束后再次打开 B19031614.txt，发现已经变成密文，如图 22 所示。

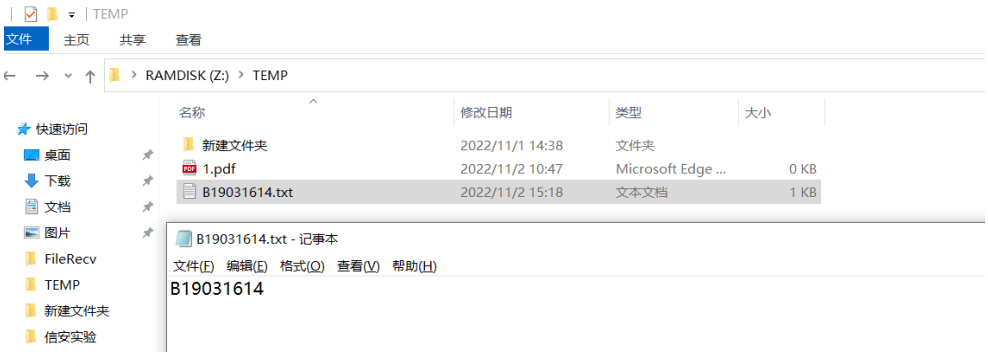

图 20 明文图

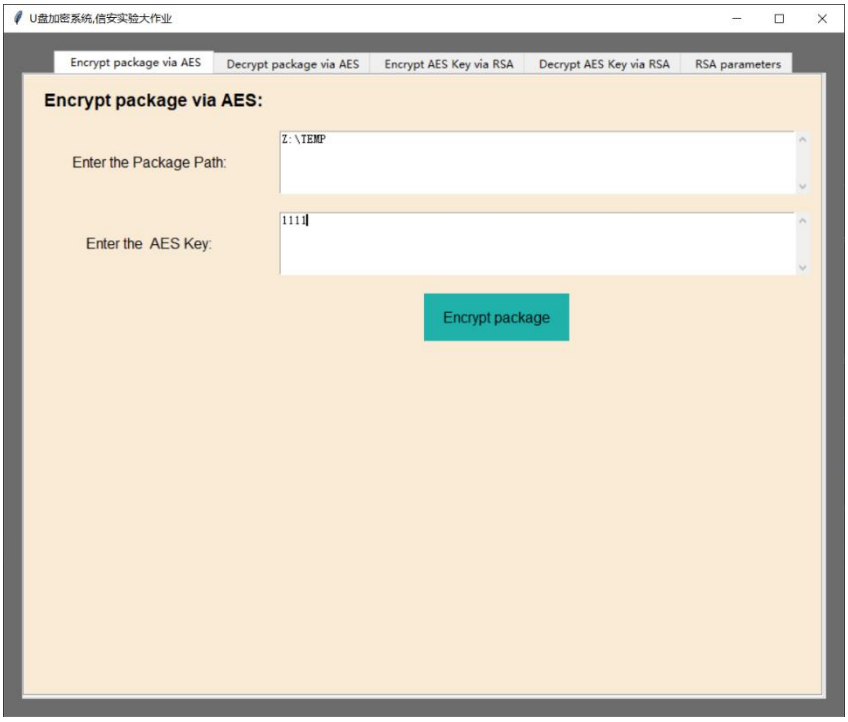

图 21 加密图

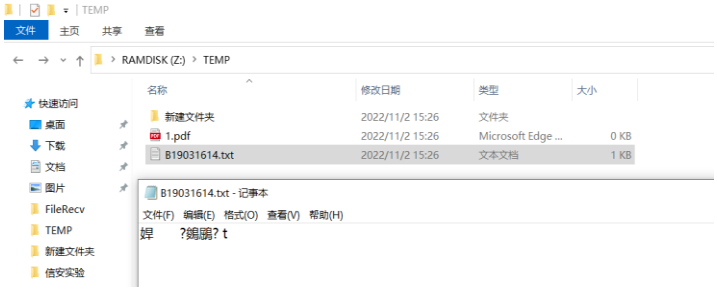

图 22 密文图

5.2.2 测试对 U 盘进行解密的功能。最终密文被恢复，如图 23 所示。

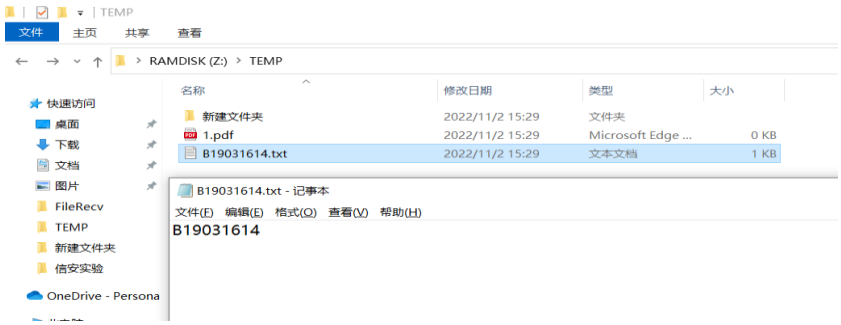

图 23 恢复明文图

5.2.3 测试对 AES 公钥进行 RSA 加密的过程，如图 24 所示

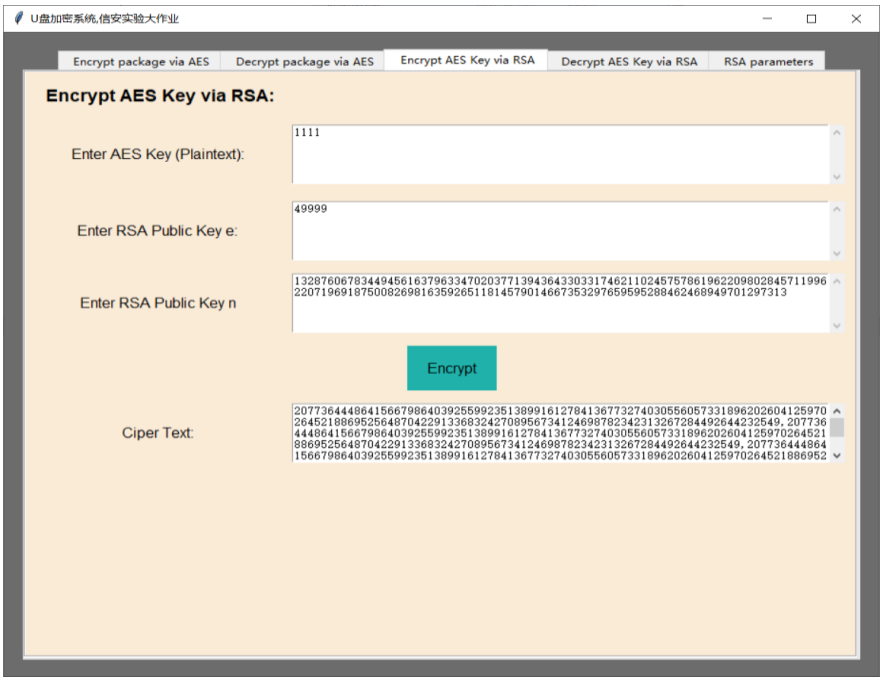

图 24 RSA 加密图

5.2.4 测试对 RSA 加密后的 AES 公钥进行解密的过程，如图 25 所示

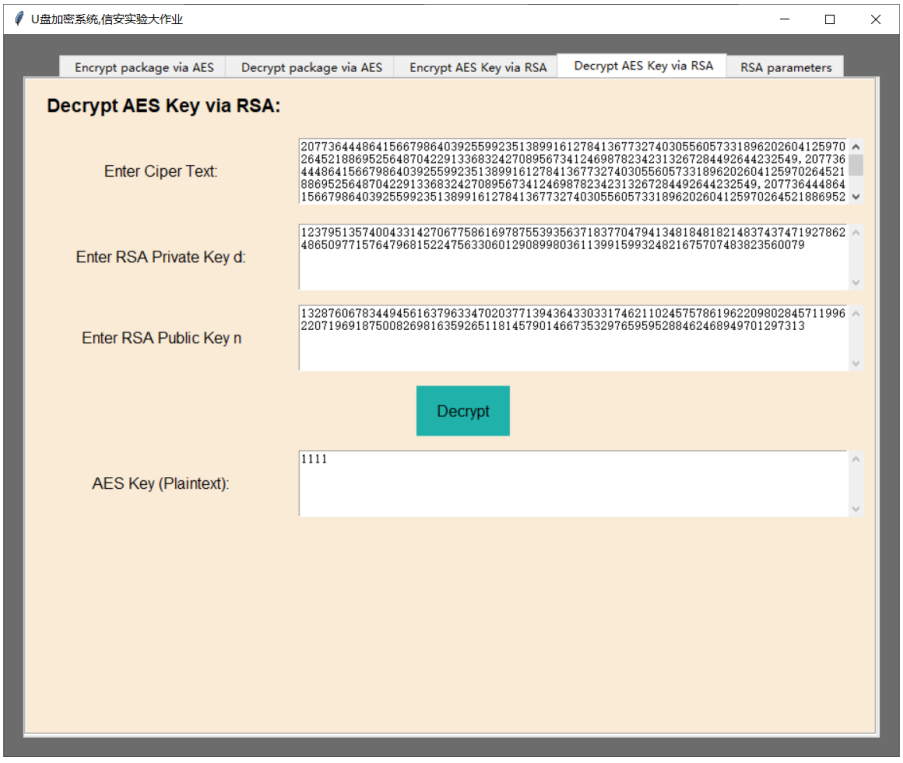

图 24 RSA 解密图

六、课题完成过程中遇到的问题及解决方法

问题 1：Python 语言中函数内部无法修改超参数（全局变量）

解决方法：使用 global 关键字，即可在函数内部访问并修改全局变量，如图 26 所示：

```
global filepaths
filepaths=[]
def all_files_path(rootDir):
    filepaths.clear()
    for root, dirs, files in os.walk(rootDir):           # walk方法返回一个三元组，分别代表根目录、文件夹、文件
        for file in files:                  # 遍历文件
            file_path = os.path.join(root, file)          # 拼接文件路径，用于获取文件绝对路径
            filepaths.append(file_path)             # 将文件路径添加进列表
            #print(file_path)
        for dir in dirs[:-1]:                  # 遍历目录下的子目录
            dir_path = os.path.join(root, dir)                  # 获取子目录路径
            all_files_path(dir_path)           # 递归调用
```

图 26 global 关键字

问题 2：Python 报错 UnicodeDecodeError: 'utf-8' codec can't decode byte 0xce in position 52: invalid continuation byte

解决方法：通过查阅资料，发现是 Unicode 解码错误：“UTF-8”编解码器无法解码位置 2 中的字节 0xBC:无效的起始字节。解决办法是直接用’rb’二进制模式读取文件。

问题 3：软件功能需求随着理解的深入不断增多，开发起来复杂

解决方法：采用增量过程模型。首先实现核心产品功能，也就是对指定路径的所有文件进行加密；后续再扩展功能，比如加上通过 RSA 算法加密 AES 对称密钥的功能。最后再加上自动生成 RSA 公钥和私钥的功能。在具体编码时尽量做到高内聚，低耦合的思想。

问题 4：关于 RSA 算法中明文长度的问题

解决方法：通过查阅资料，发现 RSA 算法本身要求加密内容也就是明文 m 必须 0<m<n，也就是说内容这个大整数不能超过 n，否则就出错。生成密文的长度和明文长度无关， 不管明文长度是多少，RSA 生成的密文长度总是固定的。

问题 5：找不到绘制用例图，序列图的免费软件

解决方法：使用在线平台

https://online.visual-paradigm.com/cn/diagrams/features/sequence-diagram-software/

七、总结

通过这次实验，我们学习了 AES 加密的基本原理，并动手开发了一个简单的图形化界面程序，用于对 U 盘中的全部文件或部分文件进行加密。同时，我们还用 RSA 公钥加密算法对 AES 算法中的对称密钥进行加密，以达到保护对称密钥的目的。

AES 算法是分组加密算法，这里我们加密的对象是文件。首先将一个文件读取成字节流，再按 16 字节为一个分组进行划分，最后对每个分组分别调用 AES 算法加密。RSA 算法是公钥加密算法，这里我们加密的对象是字符串（AES 算法中的对称密钥），将每个字符转化成其 ascill 码（int 型）以用 RSA 算法分别加密。

对称加密是最快速、最简单的一种加密方式，加密（encryption）与解密（decryption）用的是同样的密钥（secret key）。对称加密有很多种算法，由于它效率很高，所以被广泛使用在很多加密协议的核心当中。对称加密的一大缺点是密钥的管理与分配，换句话说，如何把密钥发送到需要解密你的消息的人的手里是一个问题。在发送密钥的过程中，密钥有很大的风险会被黑客们拦截。现实中通常的做法是将对称加密的密钥进行非对称加密，然后传送给需要它的人。

公钥加密算法运算速度慢，由于进行的都是大数计算，使得 RSA 最快的情况也比 AES 慢上好几倍，无论是软件还是硬件实现，速度一直是 RSA 的缺陷，所以一般只用于少量数据的加密。RSA 的速度是对应同样安全级别的对称密码算法的 1/1000 左右。

综上，一般使用对称算法来加密数据，然后用 RSA 来加密对称密钥，然后将用 RSA 加密的对称密钥和用对称算法加密的消息发送出去。

在实验过程中我们也遇到了很多问题，所幸当下互联网上的资料非常丰富，通过自学我们能解决绝大部分。在写开发程序和写实验报告的过程中，我们也复习并实践了软件工程课程中所学知识。比如需求建模，高内聚低耦合的设计思想以及单元测试集成测试等概念。总而言之，通过此课题我们的各方面能力都得到了提升，衷心感谢《专业课程设计》这门课程给予我们这样一个实践学习的机会。