目 录

[1 实验目的 3](#_Toc526356525)

[2 实验题目及解析 3](#_Toc526356526)

[3 实验环境和条件 3](#_Toc526356527)

[4 实验效果的评价 3](#_Toc526356528)

[5 实验过程 3](#_Toc526356529)

[6 实验结果及分析 3](#_Toc526356530)

[7 问题与讨论 3](#_Toc526356531)

[8 附录 4](#_Toc526356532)

# 实验目的

了解RSA算法和RSA算法在网络数据传输的实际系统中是如何发挥作用的，并使用代码模拟这个过程。

# 实验题目及解析

【题目名称，实验重点和难点分析，主要功能和性能】

**题目名称**：网络数据RSA加密传输系统研制

**实验重点**：

* 手写RSA加密算法
* RSA加密算法的证明
* 对RSA实际信息传输加密过程的模拟

**主要功能**：使用Python模拟了AB两端一端给另外一端发送消息时，利用AES(128)加密消息，RSA(2048)加密AES密钥并且使用RSA进行身份验证（SHA256）确保信息没有被恶意修改的过程。

**性能**：

* 构建RSA密钥大约需要0.2s（这是一次性的计算，由于是Python手写，效率的确低下）
* RSA加密解密AES的密钥大约需要0.3s（RSA算法本身计算复杂度较高，由于是Python手写，效率的确低下）
* AES加密解密大约需要0.1s不到（调库实现，并且AES相对RSA算法复杂度小很多）
* RSA签名验证大约需要0.1s不到（调库实现）

# 实验环境和条件

【网络拓扑结构图、逻辑关系图，实验所需软件硬设备，实验所需数据资源及其特征说明等】

联网计算机1台，内存大于2G，windows操作系统

编程语言：Python

开发软件：Vscode

# 实验效果的评价

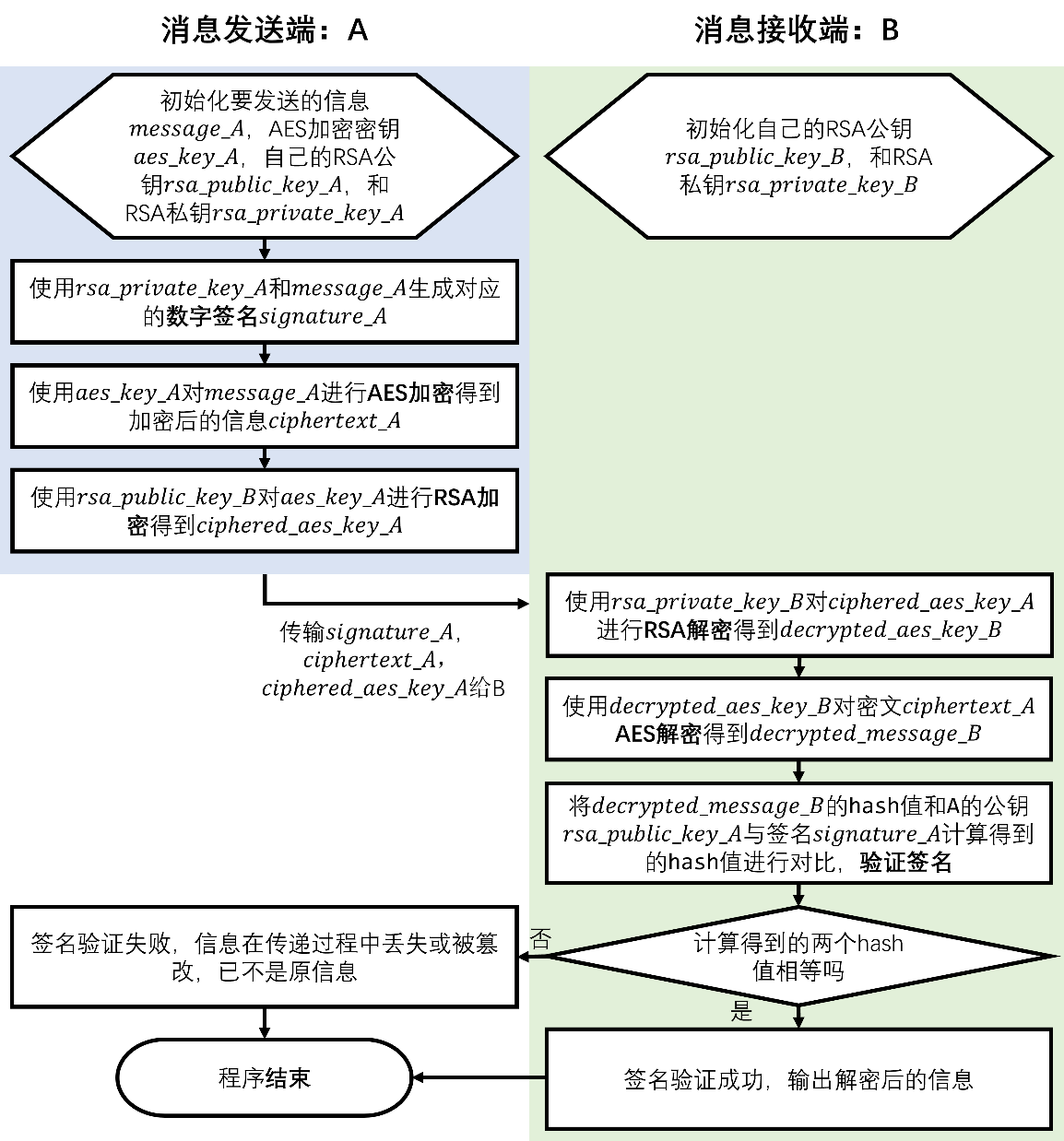
【实验效果评价方法及详细说明】

模拟系统能够正常运作

无定量评价方法（仅输出各个计算过程的耗时，身份验证是否成功，以及原始明文和解密得到的信息）

# 实验过程

【实验流程图、实验涉及参数说明等】



**RSA算法的理论证明**：

RSA算法的核心就是选择合适的正整数e,d,n使得对于要加密的某个字符的Unicode值M来说有：，这里需要提到在计算Unicode时用到的函数ord

ord(char)

ord(char) 函数是Python内置函数，用于返回字符 char 的 Unicode 码点（code point）。Unicode 码点是用于唯一标识每个字符的数值。Unicode 码点的范围是从 U+0000 到 U+10FFFF，最大值为1114111。

而在RSA算法中，我们应选择一个非常大的n（大约[2^2046, 2^2048 -1]），n远远大于M，因此这里我们可以直接将RSA算法的核心公式化简为：

根据模运算的基础，我们可以将上式改为：

这里的操作就是RSA加密操作，的操作就是RSA解密操作。倘若能选择合适的值满足上述等式，则可以使用作为RSA的公钥用来加密，作为RSA的私钥用来解密。具体的计算流程如下：

1. 随机生成两个非常大的质数（在RSA2048里面，可以令∈[2^1023,2^1024 - 1]），令n =p\*q。这里需要引入数论中的欧拉定理：若n, a为正整数，且n，a互质，则: 。φ(n)代表比n小且和n互质的数的个数。根据数论基本知识可以知道若n为质数，则，而，由于都是非常大的质数，所以这里一般默认n的因数只有，因此可以推理得到。
2. 在得到φ(n)后我们随机获取一个与φ(n)互质的数e，由于e与φ(n)互质，根据数论基本知识可以计算得到e关于φ(n)的模逆元d使得，且d一定存在。
3. 在得到与φ(n)互质的e和e关于φ(n)的模逆元d后，因为

可得

这里k是某个非负的整数，而RSA的核心算法又可以改写为

带入可得

化简得到

由于我们在第一步中根据选取的M非常大，且假定只有1,p,q,n这几个因子，那么我们就可以知道M与n是互质关系，进而由欧拉定理推导得到

带入中即可由模运算的特性化简得到

因此RSA的核心等式在这样选取的条件下是成立的。证明完成

代码整体的参数：

message\_A = "八十平米的小窝"\*10000  # 要加密的消息

aes\_key\_A = "abcd efgh ijkl m"  # AES加密解密使用的密钥，必须是16字节

A端需要设定自己要发送的信息，同时还需要设定自己的AES加密密钥（16字节）

# A作为消息发送方，B作为消息接收方

rsa\_public\_key\_A, rsa\_private\_key\_A = generate\_keypair(bits = 2048)  # RSA加密使用的公钥和私钥的生成

rsa\_public\_key\_B, rsa\_private\_key\_B = generate\_keypair(bits = 2048)  # RSA加密使用的公钥和私钥的生成

在初始化生成自己的RSA公钥私钥的时候可以指定RSA算法的位长度，长度越大计算复杂度就越高，但相对的破解难度就越大。这里按照网上查阅的资料使用推荐长度2048。

# 消息发送（A端）

print("\n[INFO] Client A side is starting to send a message")

signature\_A = sign(rsa\_private\_key\_A, message\_A)  # A用自己的私钥对明文进行签名

ciphertext\_A = aes\_encrypt(aes\_key\_A, message\_A)  # A用自己设定（或者随机）的AES key加密明文

ciphered\_aes\_key\_A = encrypt(aes\_key\_A, rsa\_public\_key\_B)  # A用B的RSA公钥加密自己的AES key

A端的函数调用与参数里，首先使用了A自己的RSA私钥对要发送的信息message\_A用计算签名，这里是通过调库使用SHA256来先对信息进行hash后再进行模幂运算处理的，输出计算得到的256位的签名信息signature\_A。

随后使用A自己的AES密钥加密要发送的信息，AES是一种对称密钥加密算法，将明文划分为固定大小的块（128 位），然后对每个块进行独立的加密，AES可以设定不同长度的密钥，这里使用的是128位的AES密钥，具体细节是调库实现的。

最后使用消息接收端B的RSA公钥对AES密钥进行加密，得到加密后的密钥。A加密过程使用python自带的库完成，的选取与证明过程中的过程一致，加密和解密使用pow函数（自带取余的参数）完成。

# 消息接收 （B端）

print("\n[INFO] Client B side is ready to receive a message")

decrypted\_aes\_key\_B = decrypt(ciphered\_aes\_key\_A, rsa\_private\_key\_B)  # RSA私钥解密AES key

decrypted\_message\_B = aes\_decrypt(decrypted\_aes\_key\_B, ciphertext\_A)  # 解密的AES key解密密文

verification\_result\_B = verify(rsa\_public\_key\_A, decrypted\_message\_B, signature\_A)  # 用A的RSA公钥对解密得到的信息进行签名验证

同理，B端的函数调用与参数里，首先对接收到的A进行RSA加密过的AES密钥使用B端自己的RSA私钥进行RSA解密，解密函数简单使用pow函数完成。随后用解密得到的AES密钥对密文信息进行再解密，得到明文信息。最后对解密得到的明文信息hash，并用A自己的RSA公钥和发送的签名做模幂运算，将得到的结果进行对比，如果相同则说明信息没有被篡改，如果不同则说明信息已不同于原本的明文信息了。

if verification\_result\_B:

    print("Verify successed!")

    # 如果信息文本太长，只展示一部分

    decrypted\_message\_B = decrypted\_message\_B[:100] + "..." if len(decrypted\_message\_B) > 100 else decrypted\_message\_B

    message\_A = message\_A[:100] + "..." if len(message\_A) > 100 else message\_A

    print("Original  message:", message\_A)

    print("Decrypted message:", decrypted\_message\_B)

else:

    print("Verify failed!")

最终根据签名验证结果输出对应的信息。

# 实验结果及分析

【基于图表的定量实验结果、定性实验结果，实验结果的分析讨论】

最开始我只写了RSA加密明文信息然后直接传递的过程，但对RSA直接加密的耗时进行分析后发现：

* **假如RSA的长度只有32**：

300k字(30万字) 加密解密耗时1s, 生成密钥的时间基本不计。（图片呢？视频呢？其实本质都一样，只是字节的传输罢了，因此这里只考虑文字的传输）

30万字，每个字加密后4Byte，一共要传输的是1,200,000 Byte = 1.2MB，假设宽带速度是10MB/s，这里可以发现传输速度大于了加密解密速度

因此得出结论：**RSA加密传输中 额外的计算代价 >> 额外的通信代价** （当然下载的流量也有增幅，这里没有考虑多出来的流量本身的价值，只是考虑了时间代价）

假设原本加密前每个字平均3Byte，**使用RSA算法加密信息的速度大约在0.9MB/s**

然而这样的速度是在max(n) = 2\*\*32 - 1的情况下的，一般的RSA推荐长度为2048，也就是 2\*\*2048 -1，在这样的情况下，RSA加密信息的速度会大大减少，因此我尝试了2048长度的RSA加密。

* **假如RSA的长度为推荐长度2048**：

10个汉字的加密解密，密钥的生成都需要至少1s，（密钥大约1-8s）

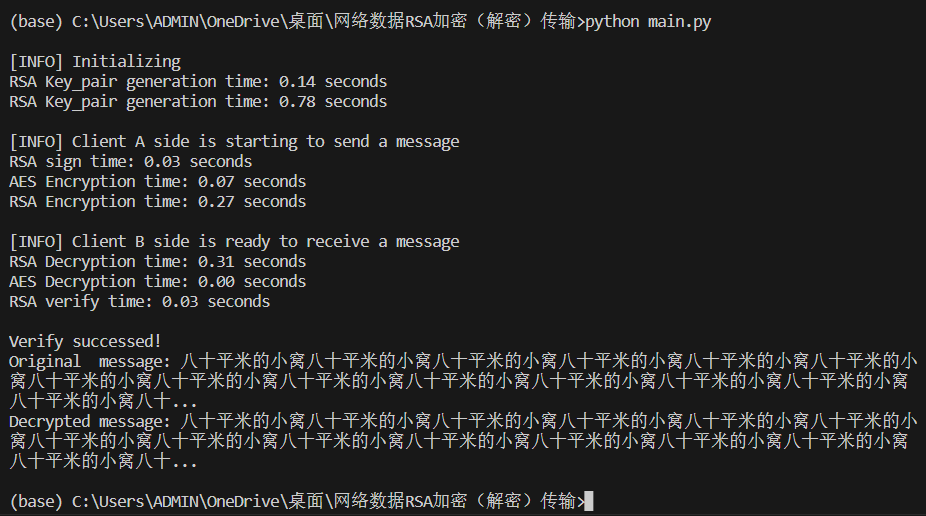
每个字加密后256Byte，一共要传输2,560Byte = 2.5KB，可以发现计算时间开销远大于传输速度的增大

假设原本加密前每个字平均3Byte，使用**RSA算法加密信息的速度大约在30B/s**，简直是龟速

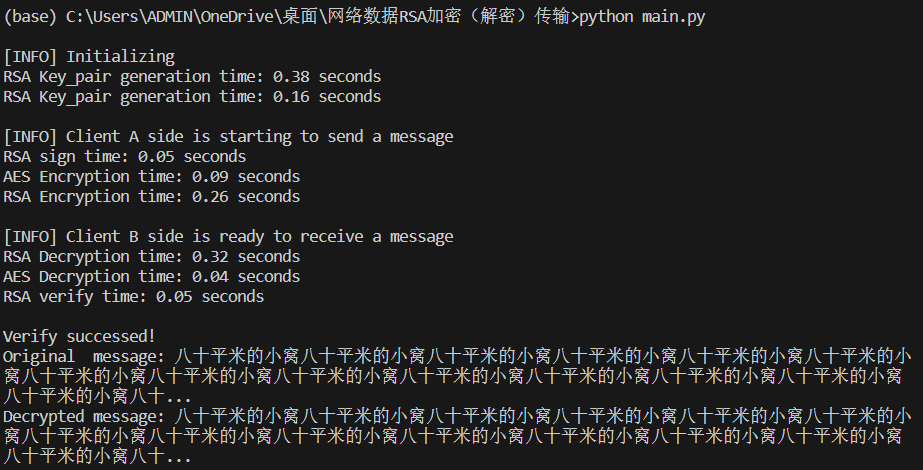
**因此用RSA直接对信息进行强加密显然是不行的，如果直接对信息进行加密只能降低RSA的长度，而这会导致其更容易被破解**

因此在查阅资料之后发现现有的一种加密方式是通过对称加密来对信息进行加密，而只对对称加密的密钥进行RSA加密。同时为了进一步提高安全性，还加入了签名验证的代码部分。也就是现在最终的代码。

在最终代码的测试输出中，为每个函数的计算都加上了运行时间的计算，在终端输出如下：



可以看到各部分的计算耗时基本都处于可接受的状态，这是大约7W字的执行输出，将文本扩大到700W字以后的输出如下：



对比可以发现AES加密算法的速度远远高于RSA的速度，即便700W汉字的加密解密仍然基本处于实时状态，700W汉字假设每个字3Byte的话，加密解密速度远远大于100MB/s，性能上非常优越。

# 问题与讨论

【实验中遇到的问题及解决方案，实验效果提升的建议等】

发现了RSA加密效率太低，查阅资料后使用了RSA+AES的加密方式。

# 附录

【实验所需其他需要说明的内容，不包括代码】