实验　非对称加密（公钥密码）算法的研究与使用

1. 实验目的

1) 理解公钥密码算法的基本原理。
2) 了解现有 RSA 算法的原理以及椭圆曲线公钥密码算法的原理与优缺点。
3) 基于 C++、Java、Python 或.net 等开发环境中成熟的 RSA 或 SM2 加密库实现一个非对称加密解密算法程序。

2. 实验预备知识及实验环境搭建

2.1. 实验预备知识

1) 公钥密码算法的基本原理：查阅文献，了解公钥密码算法的发展历史和现状，了解其加密与解密算法的基本原理。理解公钥、私钥、电子签名、数字证书等基本概念。
2) 常用非对称加密算法基本原理：查阅文献，列举常用的公钥密码算法及其优缺点。
3) 编程库的调用：了解 C++、Java 或.net 等常用语言或框架下对于公钥密码算法编程库的调用，掌握一种语言环境下调用编程库的方法，能够在现实环境中调用此类编程库。

2.2. 实验环境搭建：

选定一种编程语言或框架，搭建完整的开发环境。选定一种公钥密码算法，并在开发环境中设置对于该公钥密码算法的库引用，引用的形式可能包括头文件、静态/动态链接库等。

3. 实验要求与内容

3.1. 公钥密码算法加密与解密的基本原理

实验内容

查阅文献，并对公钥密码算法加密与解密的基本原理加以叙述。

实验要求

要求详细叙述以下内容：

1) 公钥密码算法的发展历史、研究现状、应用现状
2) 解释公钥、私钥、电子签名、数字证书等基本概念。
3) 比较软件数字证书与硬件数字证书在安全性、易用性、可靠性等方面的异同。

在报告中，需要在采用文献的位置标明参考文献序号，并在实验报告的最后列出文中所引用的所有参考文献。参考文献的标注及引用方式要求采用国家标准 GB/T 7714-2015《文后参考文献著录规则》的格式。实验报告中的参考文献标注要求采用上标形式，如“张三等[1]提出了一种更为快速的加密方法，该方法通过降低随机数的生成数量提高了数据的加密效率……”参考文献的序号按文献标注在实验报告中的自然先后顺序依次增加。

要求此处引用的文献数量不少于 3 篇。

3.2. 公钥密码算法的基本原理

实验内容

查阅文献，并对几种常用的公钥密码算法的基本原理加以叙述。

实验要求

要求详细叙述以下内容：

1) 列举不少于两种公钥密码算法。
2) 说明以上列举的公钥密码算法的基本原理与优缺点。
3) 说明拟在实现中采用的公钥密码算法。

在报告中，需要在采用文献的位置标明参考文献序号，并在实验报告的最后列出文中所引用的所有参考文献。参考文献的标注及引用方式要求采用国家标准GB/T 7714-2015《文后参考文献著录规则》的格式。

要求此处引用的文献数量不少于 2 篇。实验报告中的所有参考文献统一编号。

3.3. 环境搭建说明

实验内容

选定一种语言或框架，搭建完整的开发环境。采用上节所选定的公钥密码算法，寻找该算法在该语言中的一个较为成熟的算法库实现。在开发环境中设置该加密库的引用，完成程序的实现准备工作。

实验要求

要求完成以下内容：

1) 选定一种语言或框架，搭建完整的开发环境。要求该环境必须能够顺利完成代码的开发、调试和编译。要求描述所选定的语言，并简要说明选择该语言的原因。
2) 选定一个加密算法库，要求该算法库必须包含上节所选定加密算法的一种实现。要求列出该算法库的出处，并简要介绍该算法库。
3) 在开发环境中设置对于加密算法库的引用。要求在引用设置的各个步骤进行截图，并粘贴到实验报告中。

3.4. 加密及解密程序的实现

实验内容

采用上节所搭建好的开发环境，引用上节选定的加密算法库，完成一个加密及解密程序。

实验要求

要求完成以下内容：

1) 实现一个加密/解密程序。要求该程序采用实验报告前述章节选定的语言和加密算法。
2) 要求程序从当前目录（程序所在目录）的 plaintext.txt 文件中读取原始明文。
3) 要求程序从当前目录的 publickey1.txt、publickey2.txt 文件（公钥）中读取公钥，从当前目录的 privatekey1.txt、privatekey2.txt 文件（私钥）中读取私钥。
4) 要求程序将加密后的密文文本存储到 ciphertext.txt 文件中。
5) 要求程序封装生成公私密钥对的方法。生成的公钥与私钥分别存储到以上指定的文件中。
6) 要求程序能够执行加密算法。从 plaintext.txt 文件中读取明文文本，并将加密后的密文存储到 ciphertext.txt 文件中。要求程序能够将加密密钥作为参数输入。要求程序既能够采用公钥作为加密密钥，也能够采用私钥作为加密密钥。

7) 要求程序能够执行解密算法，从 ciphertext.txt 文件中读取密文文本，进行解密，并在控制台或文本框中输出得到的明文。得到的明文同时存储至 result.txt 文件中。要求程序能够将解密密钥作为参数输入。要求程序既能够采用公钥作为解密密钥，也能够采用私钥作为解密密钥。
8) 要求将加密算法与解密算法分别封装到两个子函数中。
9) 要求程序运行后自动进行明文载入、密码载入、明文加密、密文存储、密文解密、解密后明文结果输出的步骤。
10) 要求在代码的关键步骤添加注释。整个程序的注释数量不得少于 5 处。
11) 要求提交程序的源代码、调用的库文件（列在源代码目录下）、编译完成的可执行文件、README.txt 说明文档。全部文件存入文件夹下，打包为 zip 压缩包。

3.5. 测试结果及分析

实验内容

采用上节实现的程序测试加密及解密过程。

实验要求

要求完成以下内容：

1) 截图或采用规范的格式粘贴所采用的明文以及密码。
2) 运行程序。若程序包含界面或字符式交互过程，请截图并粘贴在实验报告中。
3) 调用程序的公私密钥对生成方法，观察多次生成的公私密钥对是否相同。记录和保存最后两次生成的公私密钥对，分别存储到 publickey1.txt/privatekey1.txt 与 publickey2.txt/privatekey2.txt 文件对中。
4) 采用第 1 对密钥对中的私钥进行加密，观察得到的密文是否与原始明文相同；采用第 1 对密钥对中的公钥进行解密。观察得到的明文是否与原始明文相同。
5) 采用相同的明文和密码再次执行上述步骤，观察多次执行的结果是否相同。
6) 采用第 1 对密钥对中的公钥进行加密，观察得到的密文是否与原始明文相同，同时观察此次得到的密文是否与前序实验步骤中采用私钥加密得到的密文相同；采用第 1 对密钥对中的私钥进行解密。观察得到的明文是否与原始明文相同。
7) 采用私钥 1 进行加密，而后采用公钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。
8) 类似以上步骤，采用公钥 1 进行加密，而后采用私钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。
9) 采用私钥 1 进行加密，而后采用私钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。
10) 要求各实验步骤均进行截图记录。

4. 实验过程与结果

4.1. 公钥密码算法加密与解密的基本原理

1) 公钥密码算法的发展历史、研究现状、应用现状。

发展历史：

1976 年，Whitfield Diffie 和 Martin Hellman[1]发表了关于公钥密码的设计思想。尽管他们没有提出具体的公钥密码算法，但他们提出了应该将加密密钥和解密密钥分开，而且还描述了公钥密码应该具备的性质。

1977 年，Ralph Merkle 和 Martin Hellman 共同设计了一种具体的公钥密码算法 Knapsack。该算法申请了专利，但后来被发现并不安全。

1978 年，Ron Rivest、Adi Shamir 和 Reonard Adleman[2]共同发表了一种公钥密码算法 RSA。RSA 可以说是现在公钥密码的事实标准。这三人于 2002 年被授予图灵奖。

此外，20 世纪 60 年代，英国电子通信安全局 CESG（Communications Electrionic Security Group）的 James Ellis 就曾经提出了与公钥密码相同的思路。1973 年，CESG 的 Clifford Cocks 设计出了与 RSA 相同的密码，并且在 1974 年，CESG 的 Malcolm Williamson 也设计出了与 Deffie Hellman 算法类似的算法。然而，这些历史直到最近才被公诸于世。

研究现状：

自从公钥密码的概念被提出以来，国际上相继提出了许多公钥密码方案。如基于大整数因子分解困难问题的 RSA 体制和 Rabin 体制；基于有限域上的离散对数困难问题的 Diffie-Hellman 公钥体制和 ElGamal 体制；基于椭圆曲线上的离散对数困难问题的 Diffie-Hellman 公钥体制和 ElGamal 体制。此外，还有基于背包问题的 Merkle-Hellman 体制和 Chor-Rivest 体制；基于代数编码理论的 MeEliece 体制以及基于有限自动机理论的公钥体制等[3]。

应用现状：

目前主要有两大类型的公钥密码系统是安全实用的：(1)基于大整数因子分解问题的，其中最典型的代表是 RSA 体制；(2)基于离散对数问题的，如 ElGamal 公钥密码体制和影响比较大的椭圆曲线公钥密码体制。由于分解大整数的能力日益增强，因此为保证 RSA 体制的安全性总是要增加模长。

目前 768bits 模长的 RSA 体制已不安全。一般建议使用 1 024bits 模长，预计要保证 20 年的安全性就要选择 2 048bits 的模长，增大模长所导致的巨大的计算复杂度带来了实现上的难度。而基于离散对数问题的公钥密码在目前技术下 512bits 模长就能够保证其安全性，特别是椭圆曲线上的离散对数的计算要比有限域上的离散对数的计算更困难，目前技术下只需要 160bits 模长即可保证其安全性，特别适合于智能卡的实现，因而受到国际上的广泛关注。国际上已制定了椭圆曲线公钥密码标准 IEEEP1363[3]。

2) 解释公钥、私钥、电子签名、数字证书等基本概念。

公钥：公钥是与私钥算法一起使用的密钥对的非秘密一半。公钥通常用于加密会话密钥、验证数字签名，或加密可以用相应的私钥解密的数据。公钥和私钥是通过一种算法得到的一个密钥对(即一个公钥和一个私钥)，其中的一个向外界公开，称为公钥；另个自己保留，称为私钥。通过这种算法得到的密钥对能保证在世界范围内是唯一的。使用这个密钥对的时候,如果用其中一个密钥加密一段数据，必须用另一个密钥解密。如用公钥加密数据就必须用私钥解密，如果用私钥加密也必须用公钥解密，否则解密将不会成功[4]。

私钥：私钥加密算法使用单个私钥来加密和解密数据。由于具有密钥的任意一方都可以使用该密钥解密数据，因此必须保护密钥不被未经授权的代理得到。私钥加密又称为对称加密，因为同一密钥既用于加密又用于解密。私钥加密算法非常快（与公钥算法相比），特别适用于对较大的数据流执行加密转换。通常，私钥算法（称为块密码）用于一次加密一个数据块。块密码（如 RC2、DES、TripleDES 和 Rijndael）通过加密将 n 字节的输 入块转换为加密字节的输出块。如果要加密或解密字节序列，必须逐块进行。由于 n 很小（对于 RC2、DES 和 TripleDES，n = 8 字 节；n = 16 [默认值]；n = 24；对于 Rijndael，n = 32），因此必须对大于 n 的数据值一次加密一个块。

电子签名：电子签名是指数据电文中以电子形式所含、所附用于识别签名人身份并表明签名人认可其中内容的数据。通俗点说，电子签名就是通过密码技术对电子文档的电子形式的签名，并非是书面签名的数字图像化，它类似于手写签名或印章，也可以说它就是电子印章。电子签名的用途: 在电子版的中秋贺卡，结婚请帖，建筑合同上签名。

数字证书：数字证书是指在互联网通讯中标志通讯各方身份信息的一个数字认证，人们可以在网上用它来识别对方的身份。因此数字证书又称为数字标识。数字证书对网络用户在计算机网络交流中的信息和数据等以加密或解密的形式保证了信息和数据的完整性和安全性。

3) 比较软件数字证书与硬件数字证书在安全性、易用性、可靠性等方面的异同。

软件开发商在自己电脑上生成私钥 (.pvk) 和证书请求文件 (CSR) 提交给 WoSign， 同时提交有关身份证明文件 (如营业执照和第三方证明文件等) 给 WoSign 查验， WoSign 验证身份后用自己的私钥给 CSR 文件签名后生成代码签名证书，也就是公钥 (.spc) 给软件开发商。这样就完成了证书的申请和颁发。

软件开发商用代码签名工具 (如： SignCode.exe) 给要签名的代码生成一个 Hash 表，再用其私钥加密 Hash 表产生认证摘要，接着就把认证摘要连同其公钥与软件代码一起打包生成签名后的新的软件代码，软件开发商就可以把已经签名的代码放到网上发行了。

最终用户从网上下载已经签名的代码时，浏览器会从签名代码中解读出其签名证书 (公钥) 和 Hash 表摘要，并与 Windows 的受信任的根证书相

比较查验公钥证书的有效性和合法性，验证签名证书正确后，就可以确认此代码确实是来自真实的软件开发商。接着，再使用签名时使用的同样算法对软件代码生成一个 Hash 表，并使用公钥也同样生成一个 Hash 表认证摘要，比较从代码中解包出来的 Hash 表认证摘要与生成的 Hash 表认证摘要是否一致，如果一致，则表明此代码在传输过程中未有任何修改，从而可以确认代码的一致性。

硬件数字证书常见的如网银 U 盾。可以看出软件数字证书和硬件数字证书的安全性都可以得到充分保障。不过在易用性上，软件数字证书更加方便而且和用户的交互性更为良好，硬件数字证书在日常生活中会占用更多空间，而且容易遗忘丢失，但其能够提供更为安全的使用环境，也极大地简化了人们的一些操作流程。在可靠性方面，硬件数字证书无疑具有更好的可靠性，例如在手机银行客户端的使用中，其安全形势的日益严峻，仅凭安全软件防护是远远不够的[5]。

4.2. 公钥密码算法的基本原理

1) 列举不少于两种公钥密码算法。

Diffie-Hellman 密钥交换算法：

Deffie-Hellman(简称 DH) 密钥交换是最早的密钥交换算法之一，它使得通信的双方能在非安全的信道中安全的交换密钥，用于加密后续的通信消息。Whitfield Diffie 和 Martin Hellman 于 1976 提出该算法，之后被应用于安全领域，比如 Https 协议的 TLS(Transport Layer Security) 和 IPsec 协议的 IKE(Internet Key Exchange) 均以 DH 算法作为密钥交换算法。

RSA 算法：

RSA 加密算法是一种非对称加密算法，所谓非对称，就是指该算法加密和解密使用不同的密钥，即使用加密密钥进行加密、解密密钥进行解密。在 RAS 算法中，加密密钥（即公开密钥）PK 是公开信息，而解密密钥（即秘密密钥）SK 是需要保密的。加密算法 E 和解密算法 D 也都是公开的。虽然解密密钥 SK 是由公开密钥 PK 决定的，由于无法计算出大数 n 的欧拉函数 phi(N)，所以不能根据 PK 计算出 SK。也就是说，对极大整数做因数分解的难度决定了 RSA 算法的可靠性。理论上，只要其钥匙的长度 n 足够长，用 RSA 加密的信息实际上是不能被解破的。

RSA 算法通常是先生成一对 RSA 密钥，其中之一是保密密钥，由用户保存；另一个为公开密钥，可对外公开。为提高保密强度，RSA 密钥至少为 500 位长，一般推荐使用 1024 位。这就使加密的计算量很大。为减少计算量，在传送信息时，常采用传统加密方法与公开密钥加密方法相结合的方式，即信息采用改进的 DES 或 IDEA 密钥加密，然后使用 RSA 密钥加密对话密钥和信息摘要。对方收到信息后，用不同的密钥解密并可核对信息摘要。

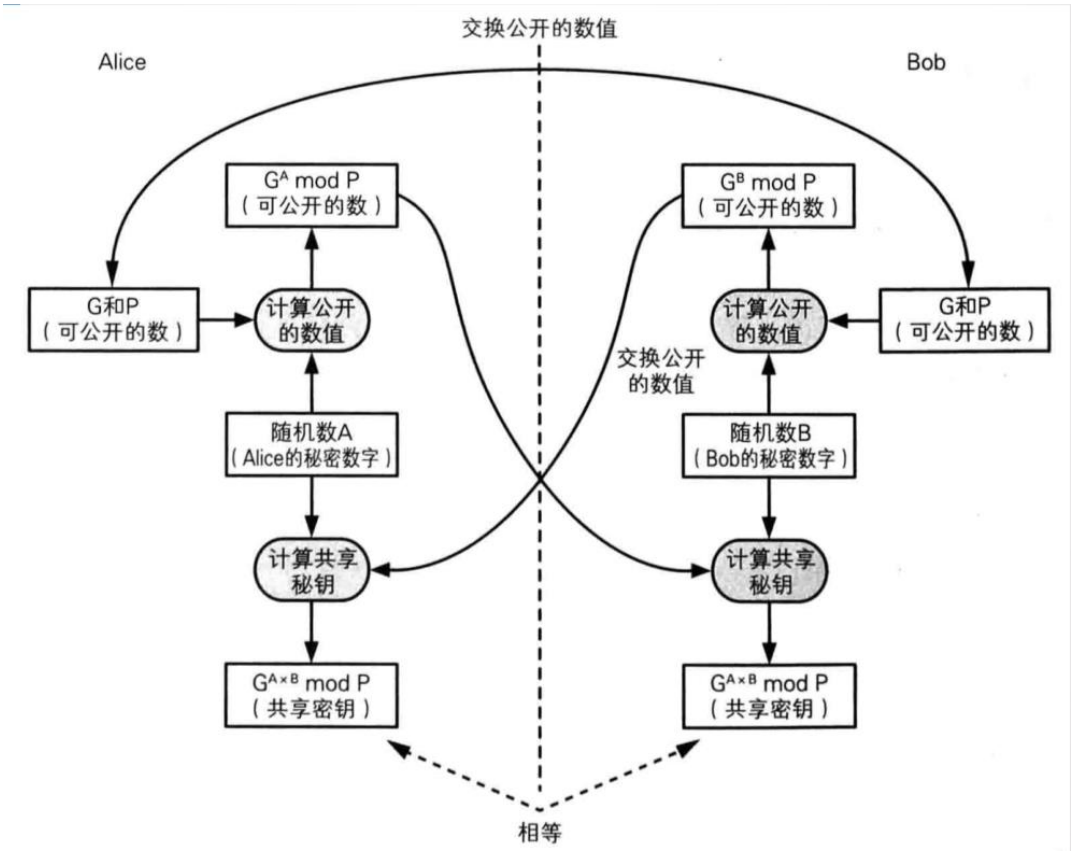
ECC 算法：

ECC，也就是椭圆曲线算法。世界上不可解的数学问题并不是只有整数分

解一个，ECC 就是基于另外一个问题：离散对数。一个加密算法的安全性取决于，由公钥去运算私钥的难度。毕竟私钥的位数是有限的，所以即使用暴力搜索的方式，也肯定是能够算出来的。而且各种单向函数虽然无解，但是不代表不能缩小求解范围，所以不同的算法其实安全性还是有差别的。而 ECC 用离散对数原理，同样的 Key 长度，安全性比 RSA 更高。比特币就是用 ECC 来生成地址和私钥。私钥位数越长就越安全，但是同时执行效率也低，所以实际使用中还是要保证一种平衡的。

2) 说明以上列举的公钥密码算法的基本原理与优缺点。

Diffie-Hellman 密钥交换算法：



(1) Alice 向 Bob 发送两个质数 P 和 G

P 必须是一个非常大的质数，而 G 则是一个和 P 相关的数，称为 生成元(generator)。G 可以是一个较小的数字。P 和 G 不需要 保密，被窃听者 Eve 获取也没关系。此外， P 和 G 可以由 Alice 和 Bob 中的任意一方生成。

(2) Alice 生成一个随机数 A A 是一个 1~ P-2 之间的整数。这个数是一个只有 Alice 知道的秘 密数字，没有必要告诉 Bob，也不能让 Eve 知道。

(3) Bob 生成一个随机数 B B 是一个 1~P-2 之间的整数。这个数是一个只有 Bob 知道的秘 密数字，没有必要告诉 Alice，也不能让 Eve 知道。

(4) Alice 将 G^A mod P 这个数发送给 Bob

(5) Bob 将 G^B mod P 这个数发送给 Alice

以上两个数让 Eve 知道也没关系。

(6) Alice 用 Bob 发过来的数计算 A 次方并求 mod P

这个数就是共享密钥。

Alice 密钥=$(G^B \bmod P)^A \bmod P$，或简化为 $G^{BxA} \bmod P$

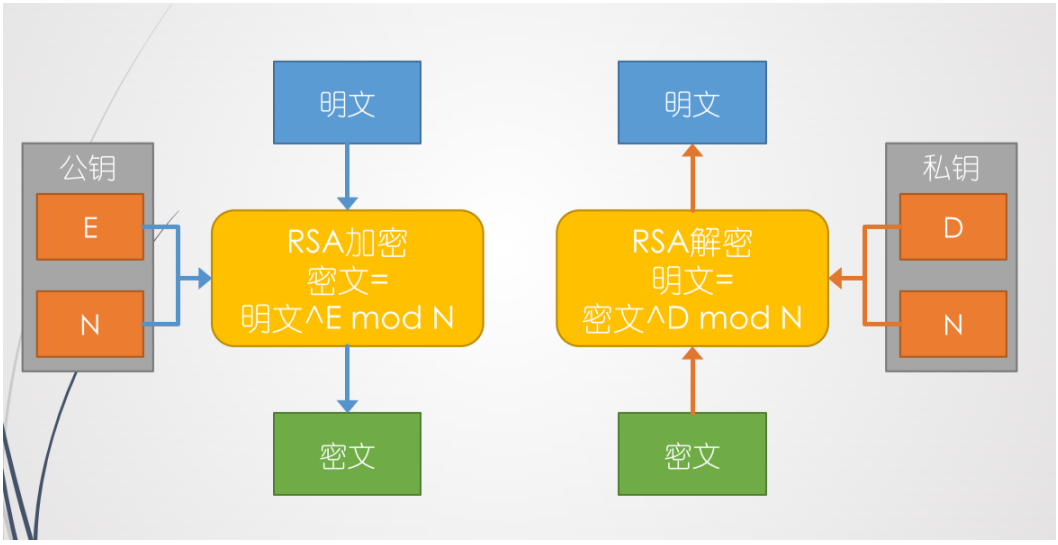
(7) Bob 用 Alice 发过来的数计算 B 次方并求 mod P

Bob 密钥=$(G^A \bmod P)^B \bmod P$，或简化为 $G^{AxB} \bmod P$

在步骤(1)~(7)中，双方交换的数字（即能够被窃听者 Eve 知道的数字）一共有 4 个： P 、G 、GA mod P 和 GB mod P。根据这 4 个数字计算 出 Alice 和 Bob 的共享密钥(GBxA mod P)是非常困难的。

举个例子，我们能够根据 GA mod P 计算出 A 吗？其中的 mod P 是这里的关键所在。如果仅仅是 GA 的话，要计算出 A 并不难，然而根据 GA mod P 计算出 A 的有效算法到现在还没有出现，这个问题称为有限群(finite group)的离散对数问题。

RSA 算法：



RSA 算法是一种公开密钥算法,其加密密钥和算法本身都是公开的,解密密钥则归用户私人所有。从 RSA 的出现的那天起就因为安全强度高、使用方便而受到关注,并得到广泛的应用[6]。

（1）生成两个大素数 p 和 q。

（2）计算这两个素数的乘积 n=p×q。

（3）计算小于 n 并且与 n 互质的整数的个数，即欧拉函数 $\Phi(n)=(p-1)(q-1)$。

（4）选择一个随机数 b 满足 $1<b<\Phi(n)$，并且 b 和 $\Phi(n)$互质，即 $\gcd(b, \Phi(n))=1$。

（5）计算 $ab=1 \bmod \Phi(n)$。

（6）、保密 a，p 和 q，公开 n 和 b。

利用 RSA 加密时，明文以分组的方式加密：每一个分组的比特数应 该小

于 log2n 比特。加密明文 x 时，利用公钥(b, n)来计算 c=xb mod n 就可以得到相应的密文 c。解密的时候，通过计算 ca mod n 就可以恢复出明文 x。

RSA 算法优点：RSA 算法是第一个能同时用于加密和数字签名的算法，也易于理解和操作。RSA 是被研究得最广泛的公钥算法，从提出到现在已近二十年，经历了各种攻击的考验，逐渐为人们接受，普遍认为是目前最优秀的公钥方案之一。该算法的加密密钥和加密算法分开，使得密钥分配更为方便。它特别符合计算机网络环境。对于网上的大量用户，可以将加密密钥用电话簿的方式印出。如果某用户想与另一用户进行保密通信，只需从公钥簿上查出对方的加密密钥，用它对所传送的信息加密发出即可。对方收到信息后，用仅为自己所知的解密密钥将信息脱密，了解报文的内容。由此可看出，RSA 算法解决了大量网络用户密钥管理的难题，这是公钥密码系统相对于对称密码系统最突出的优点。

RSA 算法缺点：1)产生密钥很麻烦，受到素数产生技术的限制，因而难以做到一次一密。2)安全性, RSA 的安全性依赖于大数的因子分解，但并没有从理论上证明破译 RSA 的难度与大数分解难度等价，而且密码学界多数人士倾向于因子分解不是 NPC 问题。3)速度太慢,由于 RSA 的分组长度太大，为保证安全性，n 至少也要 600 bitx 以上，使运算代价很高，尤其是速度较慢，较对称密码算法慢几个数量级；且随着大数分解技术的发展，这个长度还在增加，不利于数据格式的标准化。

ECC 算法：

将 ECC 中的加法运算与 RSA 中的模乘运算对应，将 ECC 中的乘法运算与 RSA 的模幂运算对应，要建立基于椭圆曲线的密码体制，需要类似因子分解两个素数之积或求离散对数这样的难题。考虑方程 Q=kP，其中 Q，P 给给定曲线上的点，对给定的 k，P 计算 Q 比较容易，而对给定的 Q 和 P 计算 k 则比较困难，这就是椭圆密码的形成机制。

ECC 加密算法具有密钥分配与管理简单、安全强度高等优点[7]。ECC 主要优势是在某些情况下，它比其他的方法使用更小的密钥，比如 RSA 加密算法，提供相当的或更高等级的安全级别。不过一个缺点是加密和解密操作的实现比其他机制 时间长 (相比 RSA 算法，该算法对 CPU 消耗严重)。

3) 说明拟在实现中采用的公钥密码算法。

在本次实验中，使用的公钥密码算法是 **RSA 算法**。相关介绍如上所示。

4.3. 环境搭建说明

1) 选定一种语言或框架，搭建完整的开发环境。要求该环境必须能够顺利完成代码的开发、调试和编译。要求描述所选定的语言，并简要说明选择该语言的原因。

在本次实验中，采用 **Python** 语言实现。Python 提供了高效的高级数据结构，

还能简单有效地面向对象编程。Python 语法和动态类型，以及解释型语言的本质，使它成为多数平台上写脚本和快速开发应用的编程语言，随着版本的不断更新和语言新功能的添加，逐渐被用于独立的、大型项目的开发。

Python 解释器易于扩展，可以使用 C 或 C++（或者其他可以通过 C 调用的语言）扩展新的功能和数据类型。Python 也可用于可定制化软件中的扩展程序语言。Python 丰富的标准库，提供了适用于各个主要系统平台的源码或机器码。

2) 选定一个加密算法库，要求该算法库必须包含上节所选定加密算法的一种实现。要求列出该算法库的出处，并简要介绍该算法库。

在本次实验中，使用 Python 代码的基础数学算法实现。

3) 在开发环境中设置对于加密算法库的引用。要求在引用设置的各个步骤进行截图，并粘贴到实验报告中。

4.4. 加密及解密程序的实现

1) 实现一个加密/解密程序。要求该程序采用实验报告前述章节选定的语言和加密算法。

2) 要求程序从当前目录（程序所在目录）的 plaintext.txt 文件中读取原始明文。

```
def get_plaintext(path):
    message = ''
    with open(path, 'r') as f:
        message =  f.read()
    return message
```

3) 要求程序从当前目录的 publickey1.txt、publickey2.txt 文件(公钥)中读取公钥，从当前目录的 privatekey1.txt、 privatekey2.txt 文件（私钥）中读取私钥。

```
encryp_key_path, decrypt_key_path = 'publickey1.txt', 'privatekey
1.txt'

try:
    fo = open(file_name, 'r')
```

4) 要求程序将加密后的密文文本存储到 ciphertext.txt 文件中。

```
with open('ciphertext.txt', 'w') as f:
    f.write(cipher_text)
```

5) 要求程序封装生成公私密钥对的方法。生成的公钥与私钥分别存储到以上指定的文件中。

```
def generate_key(key_name):

    # 在质数不太小也不太大的行范围内选择两个随机数
    rand1 = random.randint(0, 200)
    rand2 = random.randint(0, 200)

    # 将质数存储在变量中
    prime1 = int(lines[rand1])
    prime2 = int(lines[rand2])

    # 计算 n, totient, e
    n = prime1 * prime2
    totient = (prime1 - 1) * (prime2 - 1)
    e = chooseE(totient)

    # c计算 d, 1 < d < totient 使得 ed = 1 (mod totient)
    # e 和 d 是倒数 (mod totient)
    gcd, x, y = xgcd(e, totient)

    # 确保 d 为正
    if (x < 0):
        d = x + totient
    else:
        d = x

    # 将公钥 n 和 e 写入文件
    f_public = open('public' + key_name + '.txt', 'w')
    f_public.write(str(n) + '\n')
    f_public.write(str(e) + '\n')
    f_public.close()

    f_private = open('private' + key_name + '.txt', 'w')
    f_private.write(str(n) + '\n')
    f_private.write(str(d) + '\n')
    f_private.close()
```

6) 要求程序能够执行加密算法。从 plaintext.txt 文件中读取明文文本，并将加密后的密文存储到 ciphertext.txt 文件中。要求程序能够将加密密钥作为参数输入。要求程序既能够采用公钥作为加密密钥，也能够采用私钥作为加密密钥。

```
def encrypt(message, file_name = 'public_keys.txt', block_size =
2):

    try:
        fo = open(file_name, 'r')

    # 检查用户尝试使用未找到的公钥加密某些内容的可能性
    except FileNotFoundError:
        print('That file is not found.')
    else:
        n = int(fo.readline())
        e = int(fo.readline())
        fo.close()

        encrypted_blocks = []
        ciphertext = -1

        if (len(message) > 0):
            # 将密文初始化为消息第一个字符的 ASCII
            ciphertext = ord(message[0])

        for i in range(1, len(message)):
            # 如果达到最大块大小，则将密文添加到列表中
            # 重置密文，以便我们可以继续添加 ASCII 代码
            if (i % block_size == 0):
                encrypted_blocks.append(ciphertext)
                ciphertext = 0

            # 乘以 1000 将数字向左移动 3 位
            # 因为 ASCII 码最多 3 位十进制数 l
            ciphertext = ciphertext * 1000 + ord(message[i])

        # 将最后一个块添加到列表中
        encrypted_blocks.append(ciphertext)

        # 通过将所有数字取为 e 的幂来加密所有数字，并将其修改为 n
        for i in range(len(encrypted_blocks)):
            encrypted_blocks[i] = str((encrypted_blocks[i]**e) %
n)
```

```
        # 用这些数字创建一个字符串
        encrypted_message = " ".join(encrypted_blocks)

        return encrypted_message


# RSA 加密
def rsa_encryp( encryp_key_path, plaintext):
    cipher_text = encrypt(plaintext, encryp_key_path)
    print(cipher_text)
    with open('ciphertext.txt', 'w') as f:
        f.write(cipher_text)
    return cipher_text
```

7) 要求程序能够执行解密算法，从 ciphertext.txt 文件中读取密文文本，进行解密，并在控制台或文本框中输出得到的明文。得到的明文同时存储至 result.txt 文件中。要求程序能够将解密密钥作为参数输入。要求程序既能够采用公钥作为解密密钥，也能够采用私钥作为解密密钥。

```
def decrypt(decrypt_key_path, blocks, block_size = 2):

    fo = open(decrypt_key_path, 'r')
    n = int(fo.readline())
    d = int(fo.readline())
    fo.close()

    # 将字符串转换为整数列表
    list_blocks = blocks.split(' ')
    int_blocks = []

    for s in list_blocks:
        int_blocks.append(int(s))

    message = ""

    # 默认将列表中的每个 int 转换为 block_size 字符数
    # 每个 int 代表两个字符
    for i in range(len(int_blocks)):
        # 通过将其取为 d 的幂并将其修改为 n 来解密所有数字
        int_blocks[i] = (int_blocks[i]**d) % n

        tmp = ""
```

```
        # 将每个块分解为每个字符的 ASCII 码
        # 并将其存储在消息字符串中
        for c in range(block_size):
            tmp = chr(int_blocks[i] % 1000) + tmp
            int_blocks[i] //= 1000
        message += tmp

    return message


# RSA 解密
def rsa_decrypt( decrypt_key_path, ciphertext):
    result = decrypt(decrypt_key_path, ciphertext)
    print(result)
    with open('result.txt', 'w') as f:
        f.write(result)
    return result
```

8) 要求将加密算法与解密算法分别封装到两个子函数中。

9) 要求程序运行后自动进行明文载入、密码载入、明文加密、密文存储、密文解密、解密后明文结果输出的步骤。

```
if __name__ == "__main__":
    generate_key('key1')
    message = get_plaintext('plaintext.txt')
    # 采用公钥 1 进行加密，而后采用私钥 1 进行解密
    encryp_key_path, decrypt_key_path = 'publickey1.txt', 'privat
ekey1.txt'
    cipher_text = ''
    cipher_text = rsa_encryp(encryp_key_path, message)
    rsa_decrypt(decrypt_key_path, cipher_text)
```

10) 要求在代码的关键步骤添加注释。整个程序的注释数量不得少于 5 处。

11) 要求提交程序的源代码、调用的库文件（列在源代码目录下）、编译完成的可执行文件、README.txt 说明文档。全部文件存入文件夹下，打包为 zip 压缩包。

4.5. 测试结果及分析

1) 截图或采用规范的格式粘贴所采用的明文以及密码。

plaintext.txt

```
1 21200211047xyz
```

2) 运行程序。若程序包含界面或字符式交互过程，请截图并粘贴在实验报告中。

3) 调用程序的公私密钥对生成方法，观察多次生成的公私密钥对是否相同。记录和保存最后两次生成的公私密钥对，分别存储到 publickey1.txt/privatekey1.txt 与 publickey2.txt/privatekey2.txt 文件对中。

publickey1.txt

```
1 1304761
2 628407
```

privatekey1.txt

```
1 1304761
2 647943
```

publickey2.txt

```
1 1345793
2 560143
```

privatekey2.txt

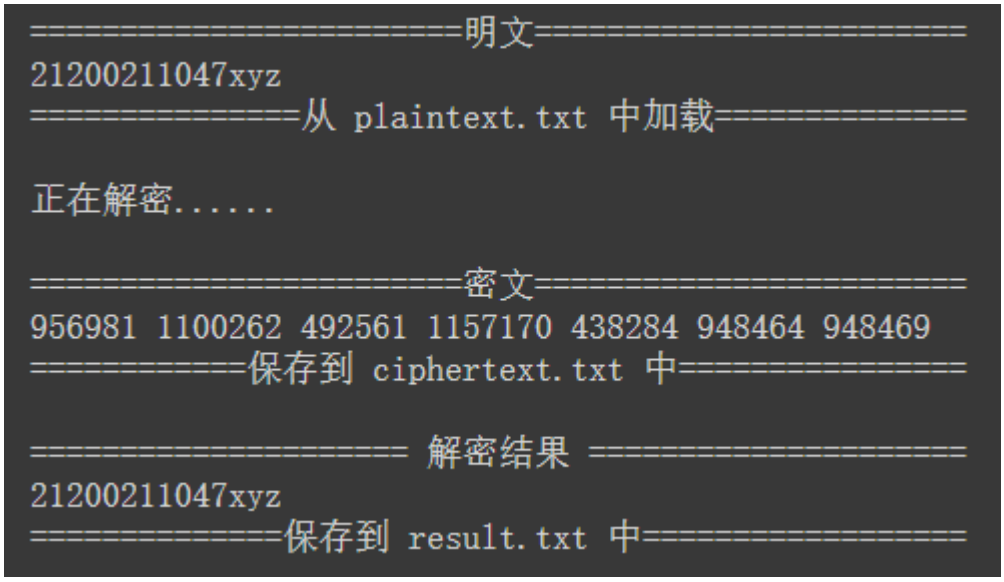
```
1 1345793
2 236767
```

4) 采用第 1 对密钥对中的私钥进行加密，观察得到的密文是否与原始明文相同；采用第 1 对密钥对中的公钥进行解密。观察得到的明文是否与原始明文相同。

```
======================明文======================
21200211047xyz
==============从 plaintext.txt 中加载==============

正在解密.....

======================密文======================
956981 1100262 492561 1157170 438284 948464 948469
============保存到 ciphertext.txt 中===============

==================== 解密结果 ====================
21200211047xyz
==============保存到 result.txt 中=================
```

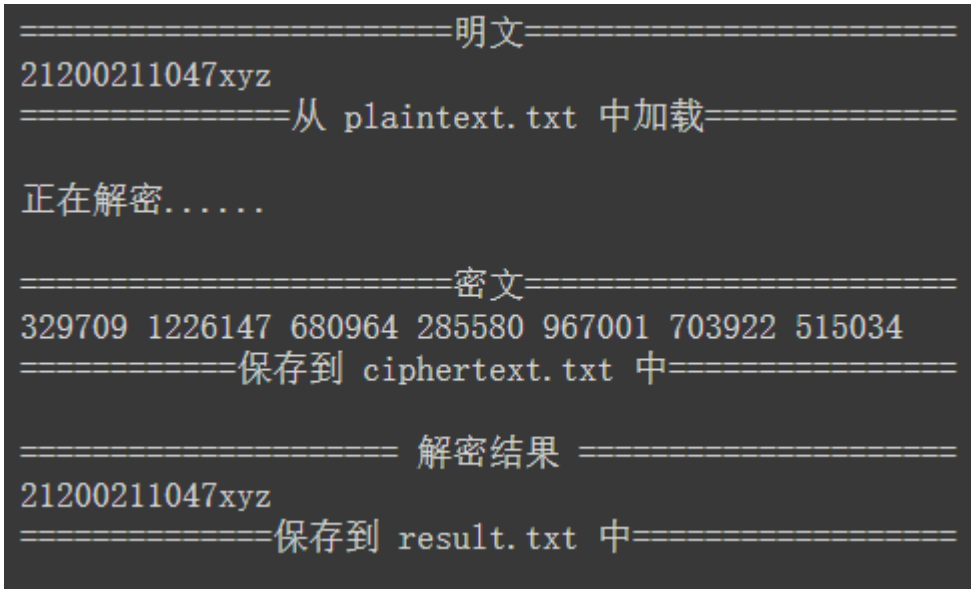
结论：得到的明文是否与原始明文相同。

5) 采用相同的明文和密码再次执行上述步骤，观察多次执行的结果是否相同。



结论：多次执行的结果是否相同。

6) 采用第 1 对密钥对中的公钥进行加密，观察得到的密文是否与原始明文相同，同时观察此次得到的密文是否与前序实验步骤中采用私钥加密得到的密文相同；采用第 1 对密钥对中的私钥进行解密。观察得到的明文是否与原始明文相同。

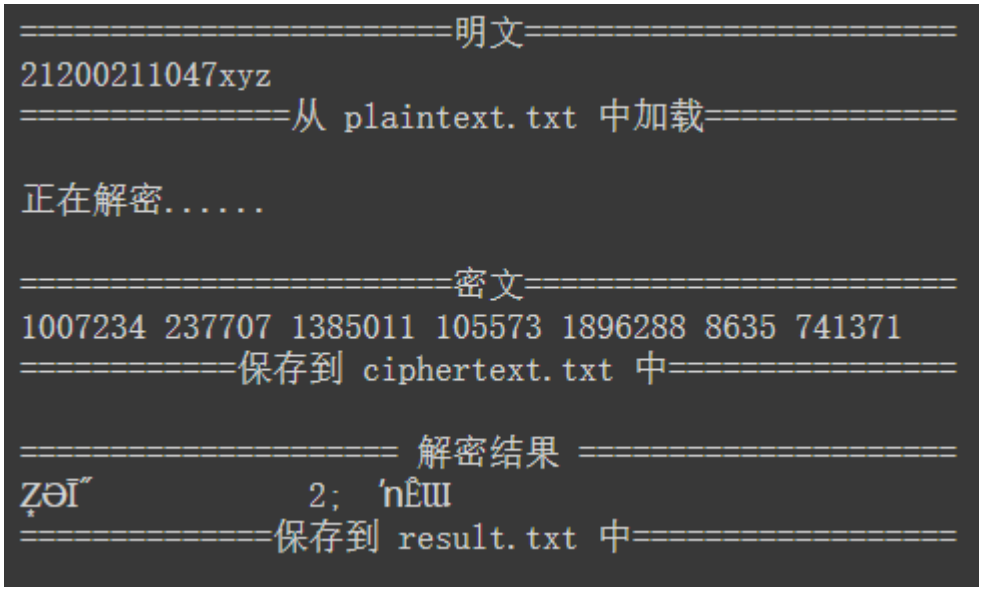


结论：1. 得到的密文与原始明文不同。

2. 此次得到的密文与前序实验中采用私钥加密得到的密文不同。

3. 得到的明文与原始明文相同。

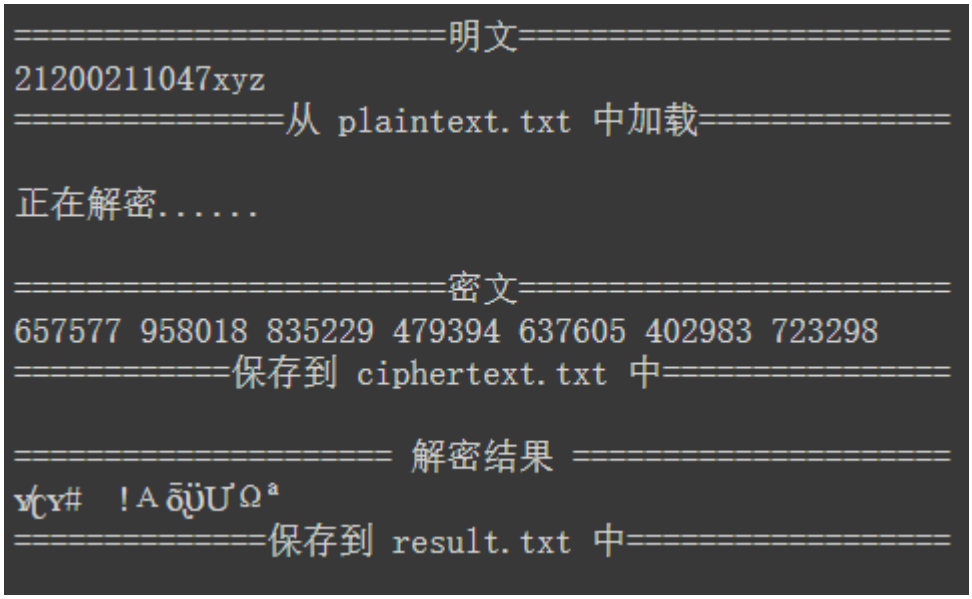
7) 采用私钥 1 进行加密，而后采用公钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。



结论：程序可以顺利执行，得到的明文与原始明文不相同。

原因：私钥 2 和公钥 1 不是一对。在使用私钥 1 进行加密后，其密文只能通过公钥 1 进行解密，如果使用公钥 2 进行解密，得到的明文与原始明文肯定不相同。

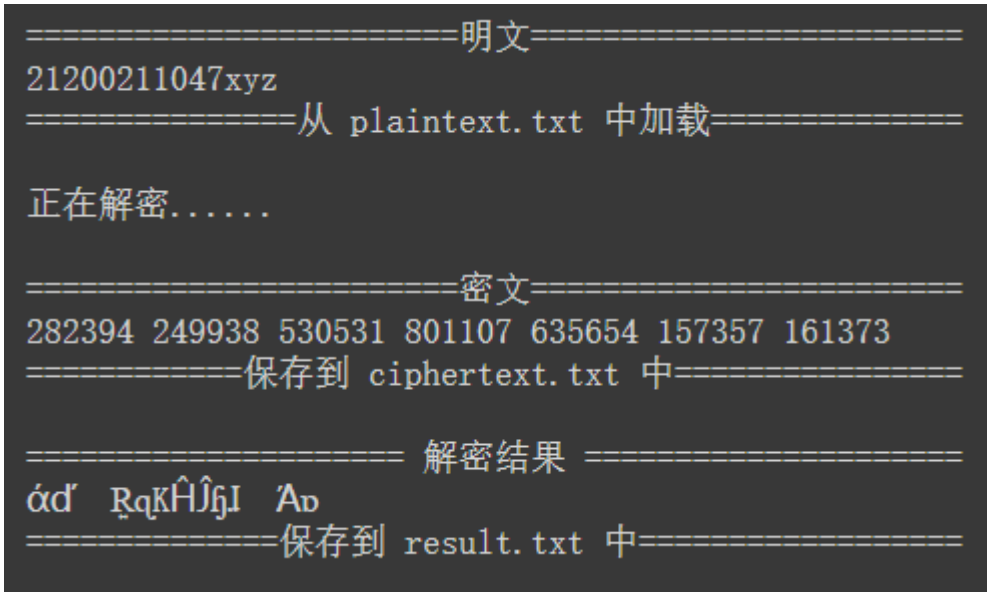
8) 类似以上步骤，采用公钥 1 进行加密，而后采用私钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。



结论：程序可以顺利执行，得到的明文与原始明文不相同。

原因：公钥 1 和私钥 2 不是一对。在使用公钥 1 进行加密后，其密文只能通过私钥 1 进行解密，如果使用私钥 2 进行解密，得到的明文与原始明文肯定不相同。

9) 采用私钥 1 进行加密，而后采用私钥 2 进行解密，观察程序是否可以顺利执行，若能够顺利执行，观察得到的明文与原始明文是否相同。分析原因。



结论：程序可以顺利执行，得到的明文与原始明文不相同。

原因：私钥 1 和私钥 2 不是一对。在使用私钥 1 进行加密后，其密文只能通过公钥 1 进行解密，如果使用私钥 2 进行解密，得到的明文与原始明文肯定不相同。

10) 要求各实验步骤均进行截图记录。

参考文献

[1] Die W , Hellman M E . New directions in cryptography, IEEE Trans. 1976.

[2] Rivest R L , Shamir A , Adleman L . A method for obtaining digital signatures and public-key cryptosystems[J]. Communications of the Acm, 1978, 21(2):120-126.

[3] 叶生勤. 公钥密码理论与技术的研究现状及发展趋势[J]. 计算机工程, 2006, 032(017):4-6.

[4] 邱卫东主编,英汉信息安全技术辞典,上海交通大学出版社,2015.11,第 489 页

[5] 黎明．网络安全视角下的计算机安全软件开发探究［J］. 网络安全技术与应用，2017（9）：7980．

[6] 胡云. RSA 算法研究与实现[J]. 北京: 北京邮电大学, 2010: 168.

[7] 王红珍, 李竹林. 基于 AES 和 ECC 的混合加密系统的设计与实现[J]. 收藏, 2012, 4.

[8] 钱能．C++程序设计教程（第 2 版）：清华大学出版社，2005 年 09 月：6-7