一、 实践目的

通过实践，将课程介绍的经典对称加密方案、非对称加密/公钥加密方案、哈希算法、数字签名方案、以及数字证书的概念融会贯通，更深刻地理解课程所学算法和技术在实际的数据安全通信中所发挥的作用，认识到课程内容在实际安全应用中的重要性，并建立良好的密码技术使用习惯。

二、 实践内容

实现对待传输目标数据的认证性、机密性和完整性保护。假设存在甲乙两方需进行数据的安全传输，甲方自选目标数据M，本地产生随机数作为文件的加/解密密钥K（采用 DES 加密），该密钥可用于对称加密方案对数据实施机密性保护$C = E(K, M)$；此外，甲方需对目标数据的散列值$H(M)$计算数字签名$Sig_{SK_{甲}}(H(M))$，以便乙方确认数据的来源以及是否被篡改。同时，为确保乙方能够正确的恢复出数据，甲方需用乙方的公钥对密钥进行加密传输$C_k = E_{PK_{乙}}(K)$。请以甲方的身份实现上述功能，并为乙方提供验证程序，验证程序能够恢复用于对文件进行解密的密钥$K = D_{SK_{乙}}(C_k)$，然后实现解密$M = D(K, C)$，并验证甲方签名是否正确$Ver_{PK_{甲}}(M, Sig)$，以及数据是否被正确恢复。

三、 实践环境

硬件环境：Windows10 系统

软件环境： Visual Studio Code

四、 实践过程与步骤

1. 密钥生成

首先进行 DES 加密与 RSA 加密的密钥生成，之前的实验已经实现了 RSA 密钥的生成，这里不再描述。DES 加密的密钥生成我利用 c 语言的 rand 函数生成随机数序列再转换成十六进制，最后生成了 64bit 的密钥。代码如下：

```
char *hex = malloc(17);          //用于存储16个16进制数，即64bit密钥
memset(hex,'\0',17);
srand((unsigned)time(NULL));
for(int j=0; j<16; j++){
    int dec = rand()%16;                //生成一个0~15的随机十进制数
    char temp[2]; temp[2] = '\0';
    itoa(dec,temp,16);                  //将随机数转换成十六进制
    temp[0] = lower2upper(temp[0]);     //转换大小写
    hex[j] = temp[0];                   //存入缓存中
}
FILE *fp = fopen(argv[i],"w+");
fprintf(fp,hex);
```

2. 加密算法实现

RSA 加密算法和 DES 加密算法均为之前实验中我已经实现过的算法。其中我对 RSA 加密算法进行了一些改进，我之前实现的 RSA 算法对明文是进行整块加密的，而在这里我将明文分为一小块一小块的加密拼接在一起形成密文，解密时也是一块一块地解密然后在拼接形成明文。

Sha256 算法采用了网上的资源：[sha256 C 语言实现_qq_43176116 的博客-CSDN 博客_c sha256]

3. 简易数据传输的实现。

实现了密钥生成和加解密算法之后，我们就可以开始模拟数据传输的流程了。

一共有三个可执行程序，分别为 enORde.exe， key.exe，verify.exe。

它们的作用分别为：

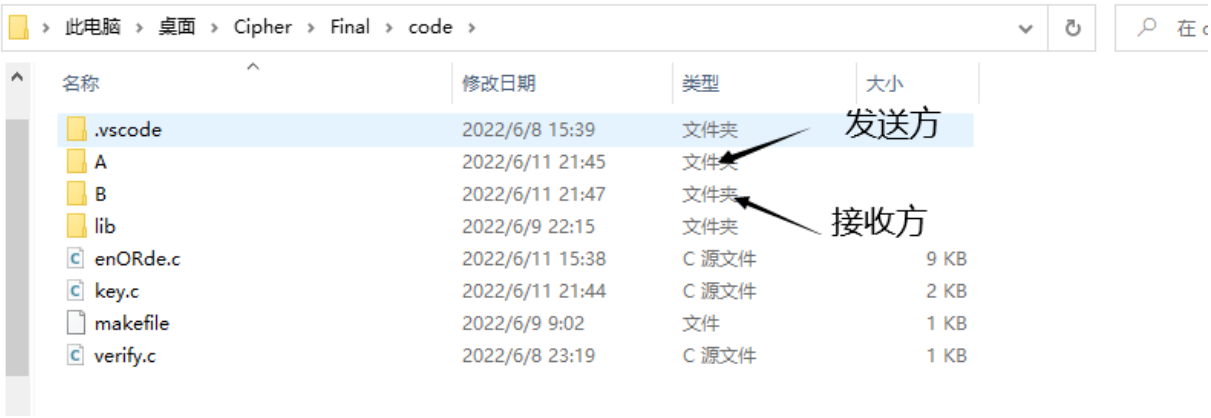
enORde.exe: DES 加密解密，RSA 加密解密，生成 sha256 散列值。

key.exe: 生成 DES 加密密钥，生成 RSA 公私钥。

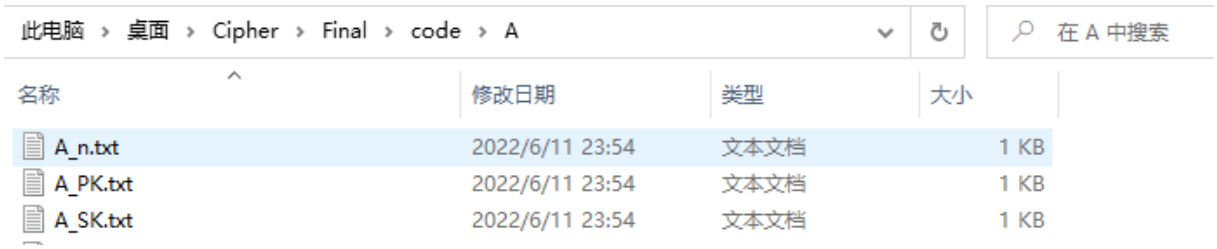
verify.exe: 接收方用于验证收到的数据和签名是否对应。

1. 发送

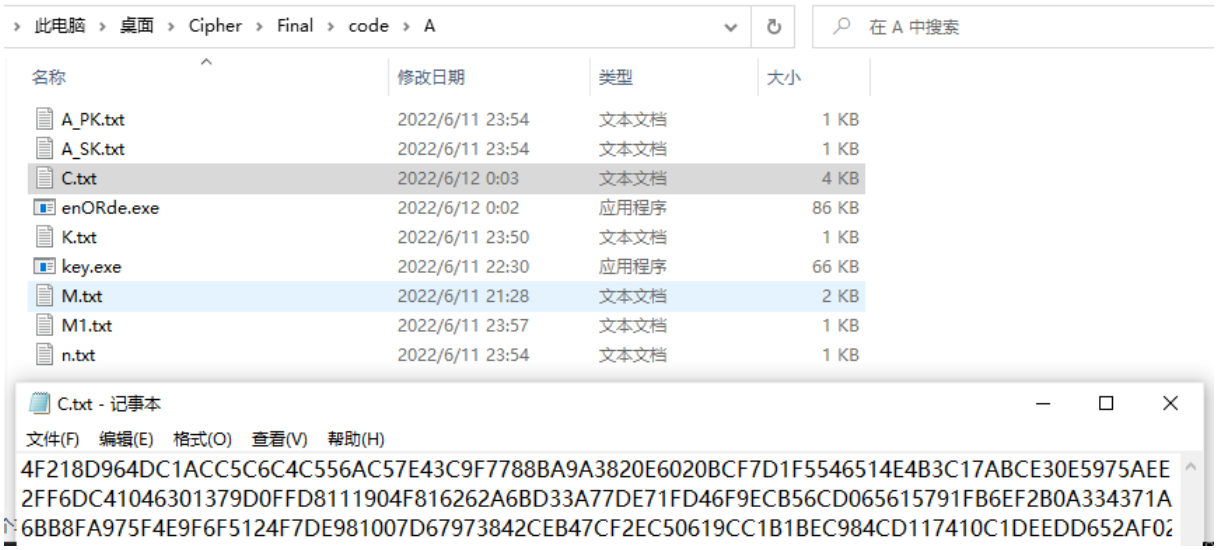
首先创建 A、B 两个目录模拟通信中的发送方和接收方。



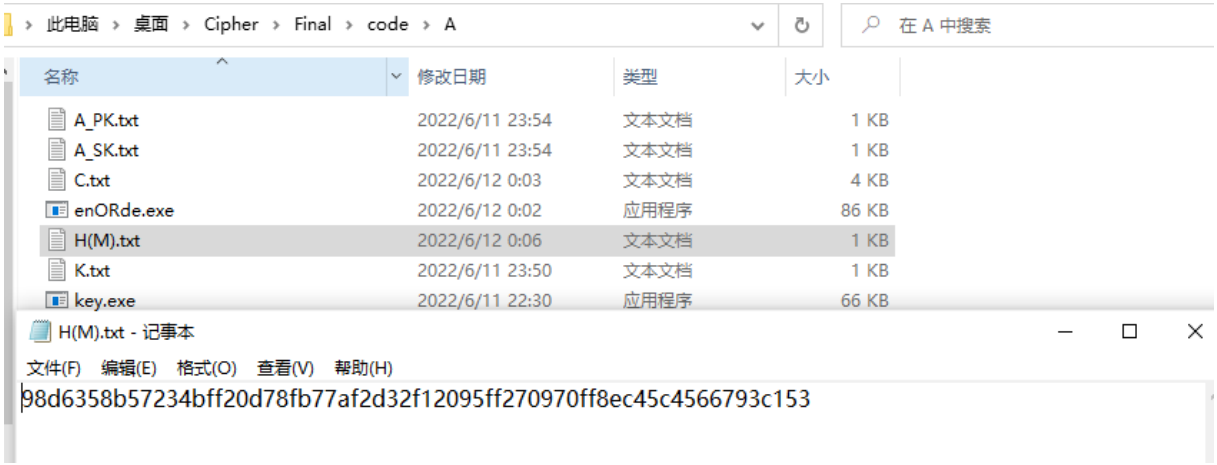
在目录 A 下命令行中执行命令 key -des K.txt 为发送方 A 生成 DES 密钥文件 K.txt，然后执行命令 key -rsa A_n.txt A_PK.txt A_SK.txt 生成 A 的公钥私钥。



执行命令 enORde -des -en M.txt K.txt C.txt，用 DES 密钥文件 K.txt 中的密钥对明文 M 进行加密得到密文文件 C.txt。



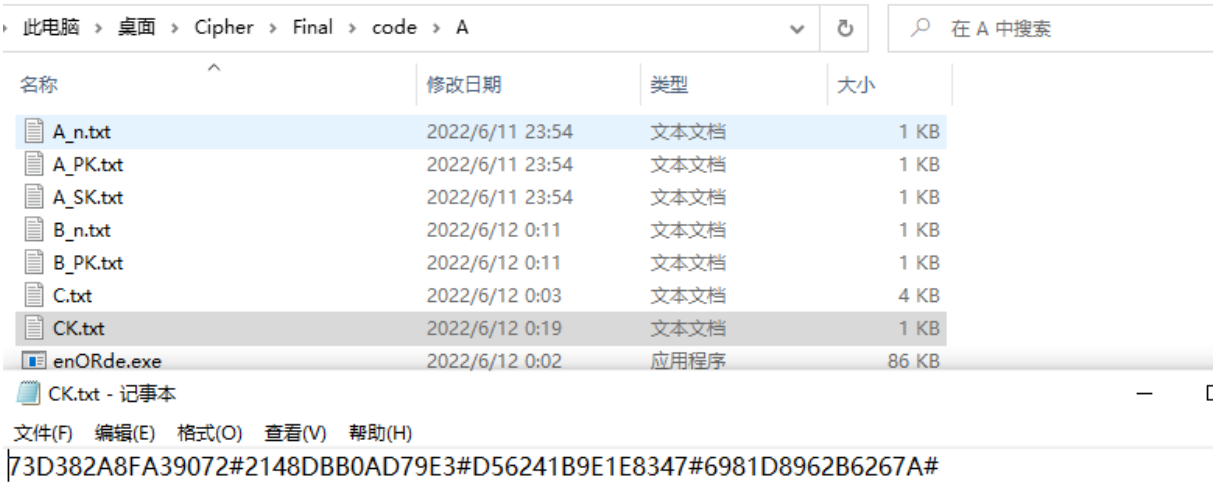
然后执行命令 enORde -sha256 M.txt H(M).txt 为明文 M 生成散列值文件 H(M).txt



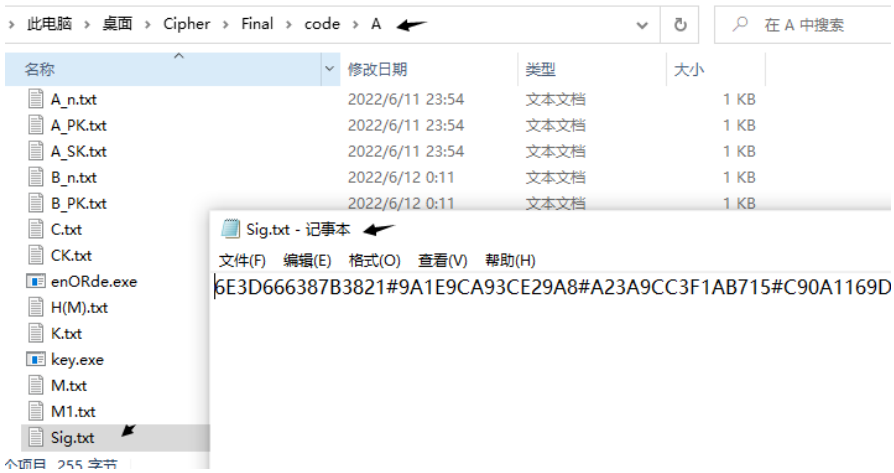
在目录 B 下执行命令 key -rsa n.txt B_PK.txt B_SK.txt 生成公钥私钥

公钥双方应共享。

然后在目录 A 下执行命令 enORde -rsa -en K.txt B_n.txt B_PK.txt CK.txt 生成密钥密文文件 CK.txt,这里是用 B 的公钥对密钥 K 进行加密。

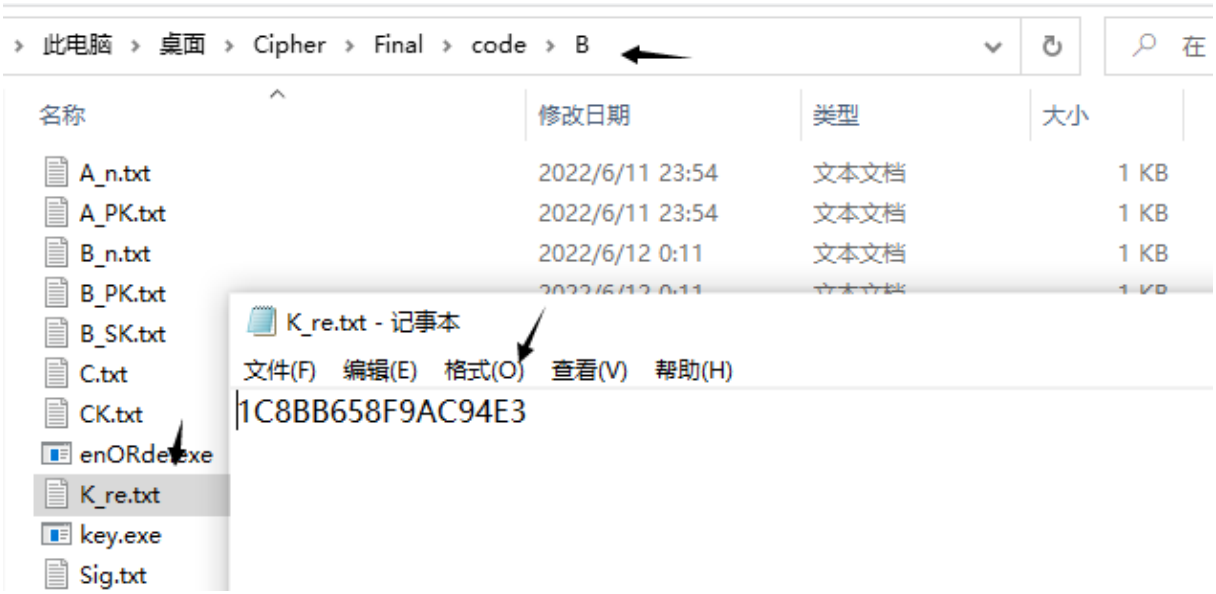


继续执行命令 enORde -rsa -en H(M).txt A_n.txt A_SK.txt Sig.txt 生成数字签名文件 Sig.txt,这里是用 A 的私钥对散列值 H(M)进行签名。

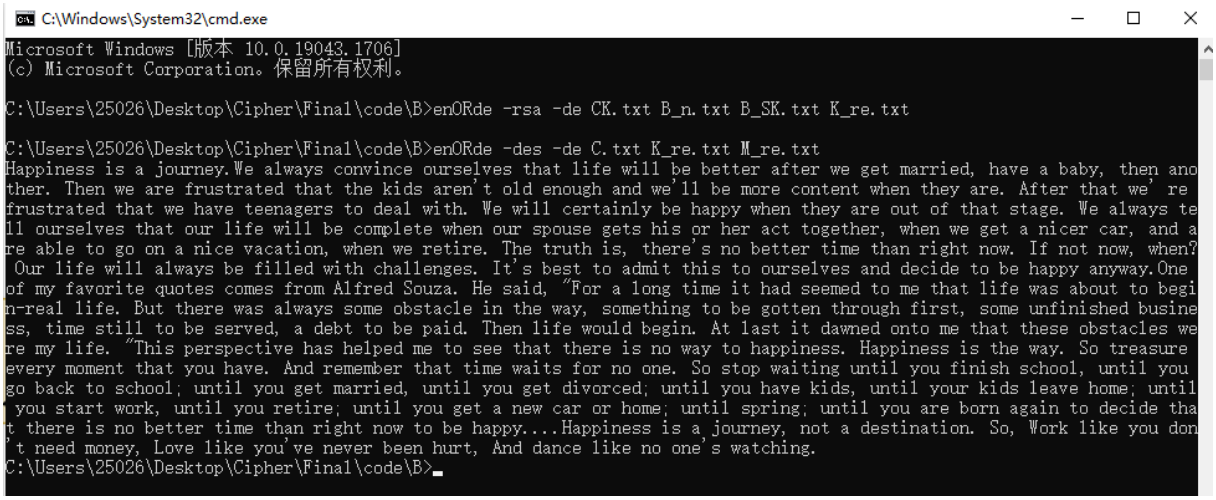


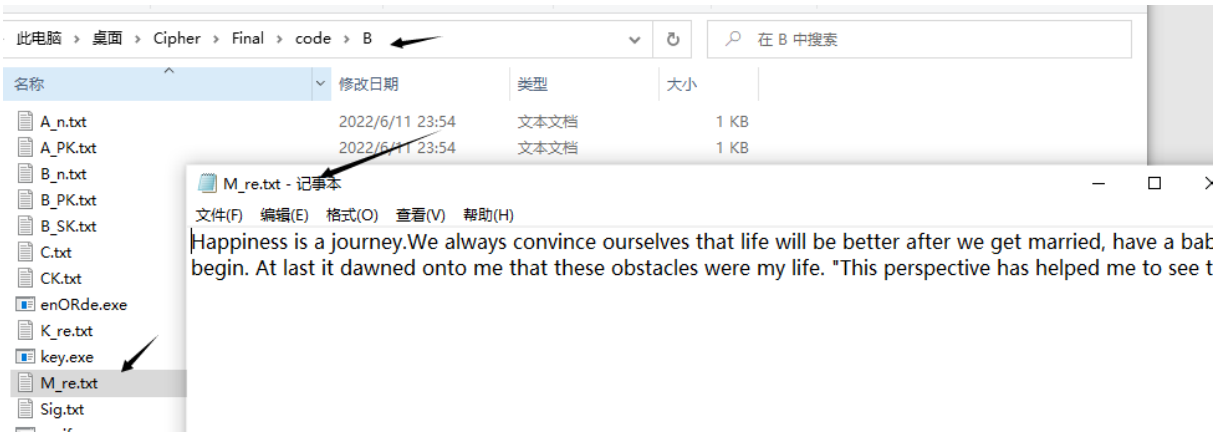
此时，发送 A 已经生成了**密文文件 C.txt，密钥密文文件 CK.txt，数字签名 Sig.txt。**然后将它们打包发送给 B

B 接收到上述文件后（目录 B 下出现对应文件），要获取明文，先在**目录 B 下**执行命令 **enORde -rsa -de CK.txt B_n.txt B_SK.txt K_re.txt** 解密出**恢复密钥文件 K_re.txt。这里是用 B 的私钥对密钥密文 CK 进行解密得到恢复密钥 K_re。**

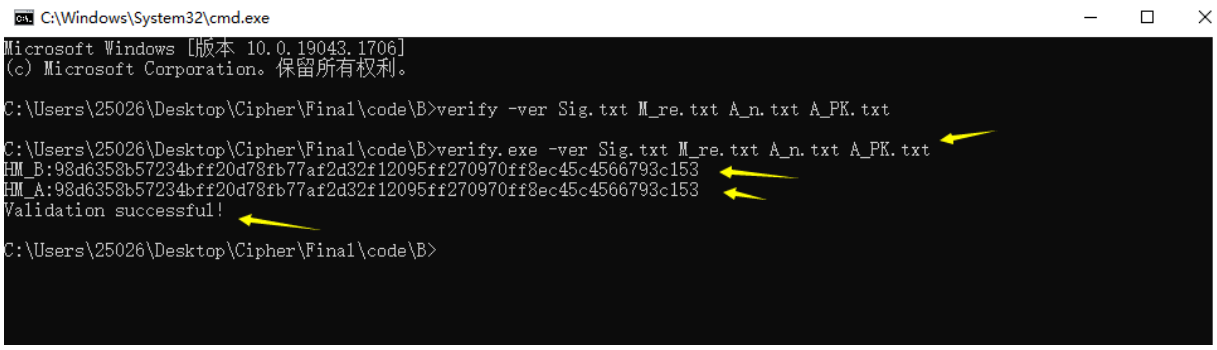


然后用得到的恢复密钥 K_re 对密文 C 进行解密，执行命令 **enORde -des -de C.txt K_re.txt M_re.txt** 得到**恢复明文文件 M_re.txt**。



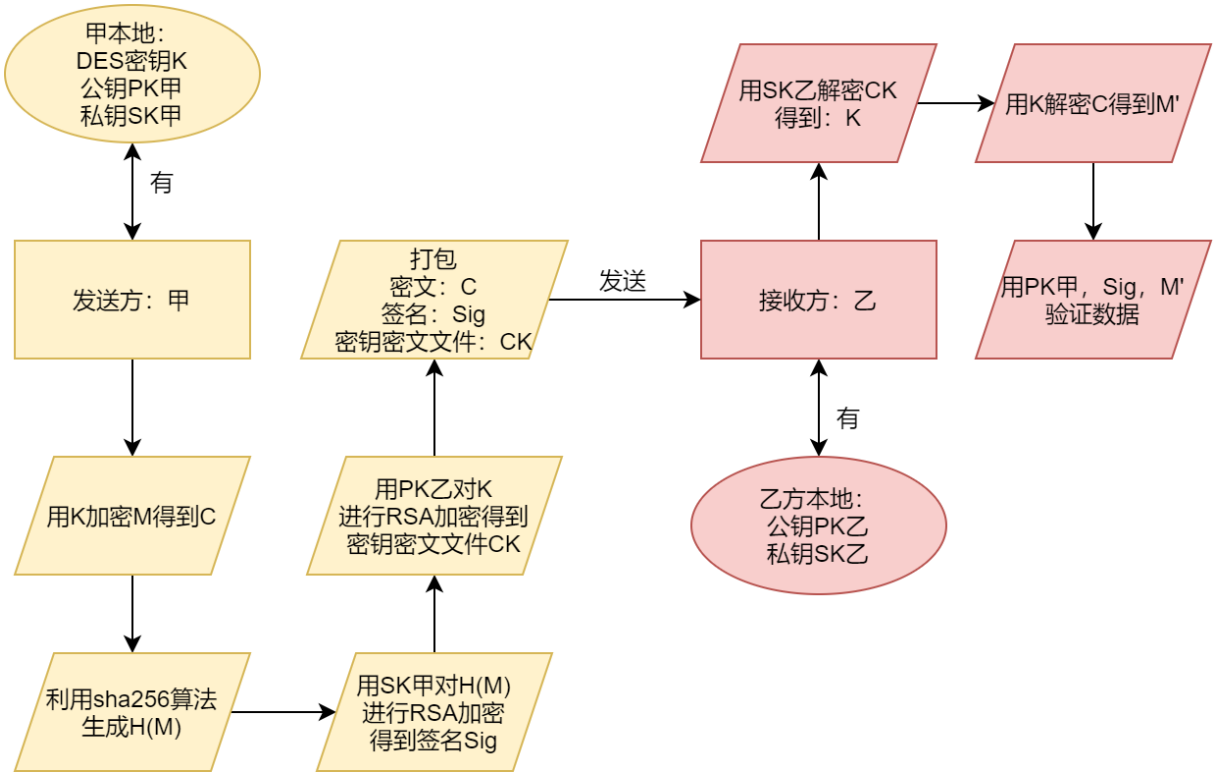


得到恢复明文文件 M_re.txt 后，我们需要**对签名进行验证**，利用验证程序 verify.exe，输入**签名，恢复明文，A 的公钥**进行验证。执行命令 **verify.exe -ver Sig.txt M_re.txt A_n.txt A_PK.txt**。



可以看到，验证成功。

五、 程序设计方案



六、 实践结果与分析

实践结果见“四、实践过程与步骤”。整个通信流程下来，实现了对明文加密解密，对密钥加密解密，对签名进行验证。

整个通信流程比较简单，双方没有规范的通讯工具，我仅仅使用了两个目录代表通信双方，用复制粘贴模拟信息传输的过程。

此外，由于能力有限，生成数字证书需要额外的工具，我这里没有为发送方 A 生成证书，而是将证书中的公钥直接共享给了接收方，这样接收方也可以实现签名的验证。_