实验一：在 OpenSSL 中进行数据签名及验证

实验要求

参照教材2.3.6中的实验指南，在 OpenSSL 中进行数据签名及验证。

理论知识

rsa简介[1]

RSA算法是一个广泛使用的公钥算法。
其密钥包括公钥和私钥。它能用于数字签名、身份认证以及密钥交换。
RSA密钥长度一般使用1024位或者更高。RSA密钥信息主要包括[1]:

```
n : 模数
e : 公钥指数
d : 私钥指数
p : 最初的大素数
q : 最初的大素数

dmp1 : e*dmp1 = 1 (mod (p-1))
dmq1 : e*dmq1 = 1 (mod (q-1))
iqmp : q*iqmp = 1 (mod p )
```

其中，公钥为n和e；私钥为n和d。
在实际应用中，公钥加密一般用来协商密钥；私钥加密一般用来签名。

OpenSSL的rsa实现

```
Openssl的RSA实现源码在crypto/rsa目录下。它实现了RSA PKCS1标准。主要源码如下：

1)  rsa.h
    定义RSA数据结构以及RSA_METHOD，定义了RSA的各种函数。

2)  rsa_asn1.c
    实现了RSA密钥的DER编码和解码，包括公钥和私钥。

3)  rsa_chk.c
    RSA密钥检查。

4)  rsa_eay.c
    Openssl实现的一种RSA_METHOD，作为其默认的一种RSA计算实现方式。
    此文件未实现rsa_sign、rsa_verify和rsa_keygen回调函数。

5)rsa_err.c
    RSA错误处理。

6)rsa_gen.c
    RSA密钥生成，如果RSA_METHOD中的rsa_keygen回调函数不为空，则调用它，
    否则调用其内部实现。

7)rsa_lib.c
```

主要实现了RSA运算的四个函数(公钥/私钥，加密/解密)，
它们都调用了RSA_METHOD中相应都回调函数。

8)rsa_none.c
实现了一种填充和去填充。

9)rsa_null.c
实现了一种空的RSA_METHOD。

10) rsa_oaep.c
实现了oaep填充与去填充。

11)rsa_pk1.
实现了pkcs1填充与去填充。

12)rsa_sign.c
实现了RSA的签名和验签。

13)rsa_ssl.c
实现了ssl填充。

14)rsa_x931.c
实现了一种填充和去填充。

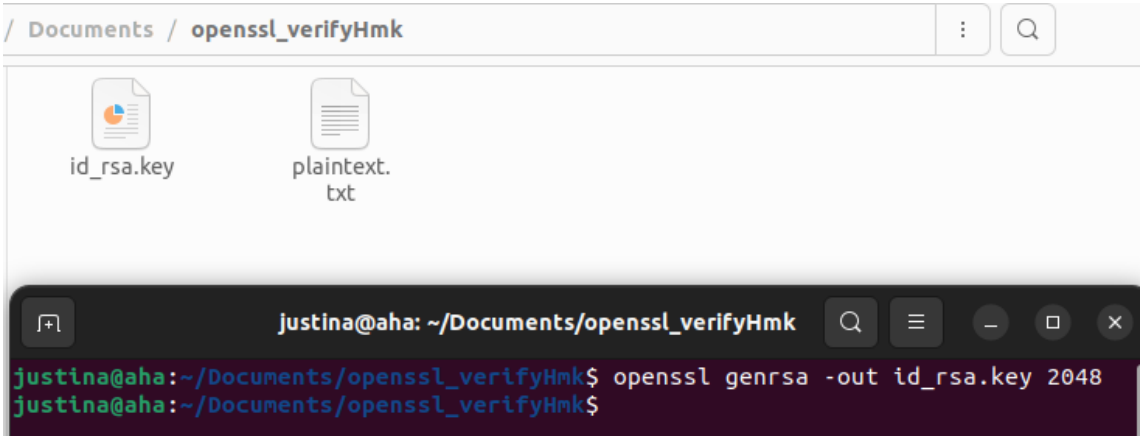
实验步骤[2]

在ubuntu22虚拟机环境（自带openssl)下，进行如下步骤：

1. 数字签名与验证：

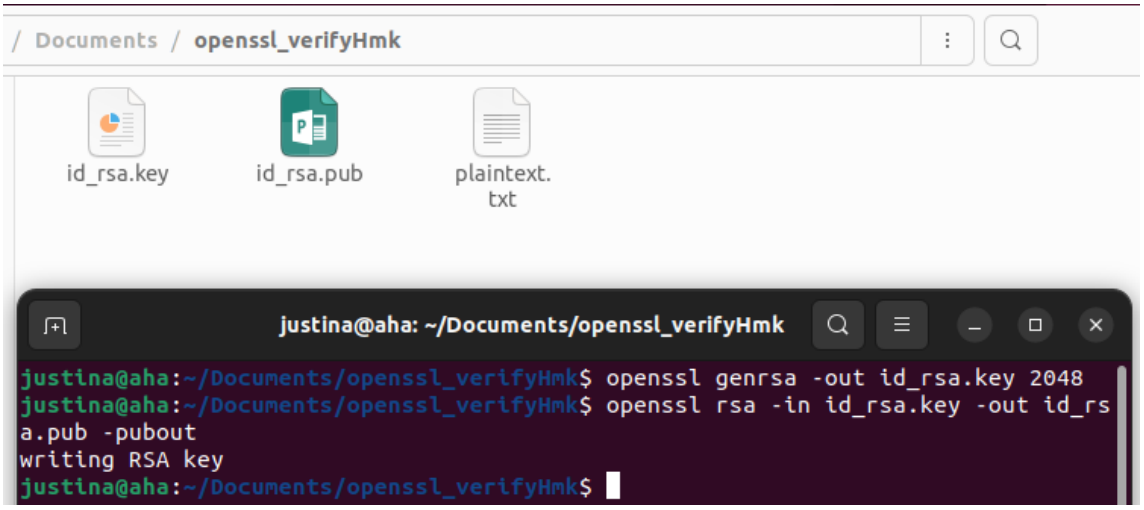
   1)生成rsa长度为2048位的私钥并存入id_rsa.key

```
openssl genrsa -out id_rsa.key 2048
```



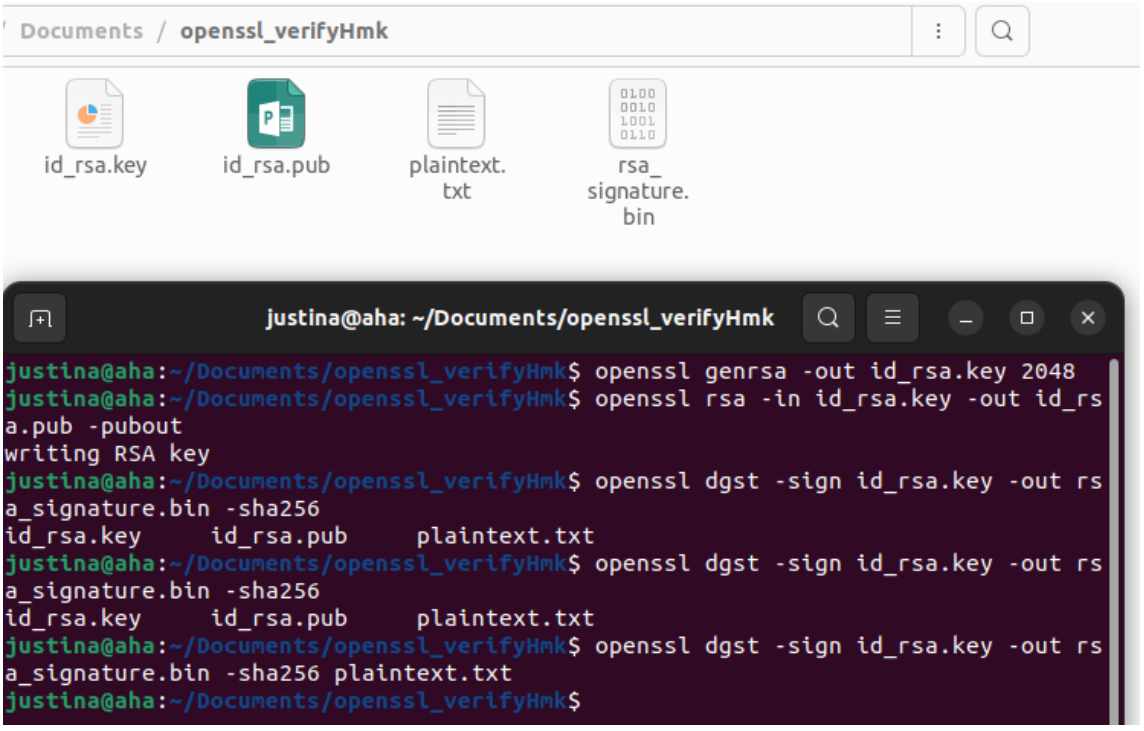
   2)根据私钥生成对应公钥，导出公钥文件 id_rsa.pub

```
openssl rsa -in id_rsa.key -out id_rsa.pub -pubout
```



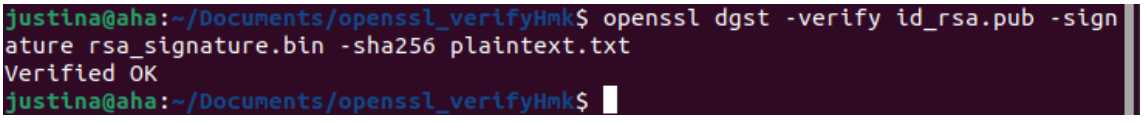
3)使用私钥对文件 plaintext.txt 的摘要进行签名，输出签名到 rsa_signature.bin

```
openssl dgst -sign id_rsa.key -out rsa_signature.bin -sha256 plaintext.txt
```



4)用公钥对签名进行验证

```
openssl dgst -verify id_rsa.pub -signature rsa_signature.bin -sha256
plaintext.txt
```



2. 为了让数字签名实现自动化验证，我们可以参考教材编写的signature.cpp脚本。实际步骤和在上述terminal中执行一样，实现了：

   1)由RSA 生成公私钥，存储到文件的函数 `genrsa`

   2)生成数据签名的 `gensign`

   3) 使用公钥验证数字签名的 `verify`

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>
// 公钥文件名
#define PUBLIC_KEY_FILE_NAME "public.pem"
// 私钥文件名
#define PRIVATE_KEY_FILE_NAME "private.pem"
// RSA 生成公私钥，存储到文件
bool genrsa(int numbit)
{
 ...
}
// 生成数据签名
bool gensign(const uint8_t *in, unsigned int in_len, uint8_t *out, unsigned
int *out_len)
{
 ...
}
// 使用公钥验证数字签名，结构与签名相似
bool verify(const uint8_t *msg, unsigned int msg_len, const uint8_t *sign,
unsigned int
sign_len)
{
...
}
```

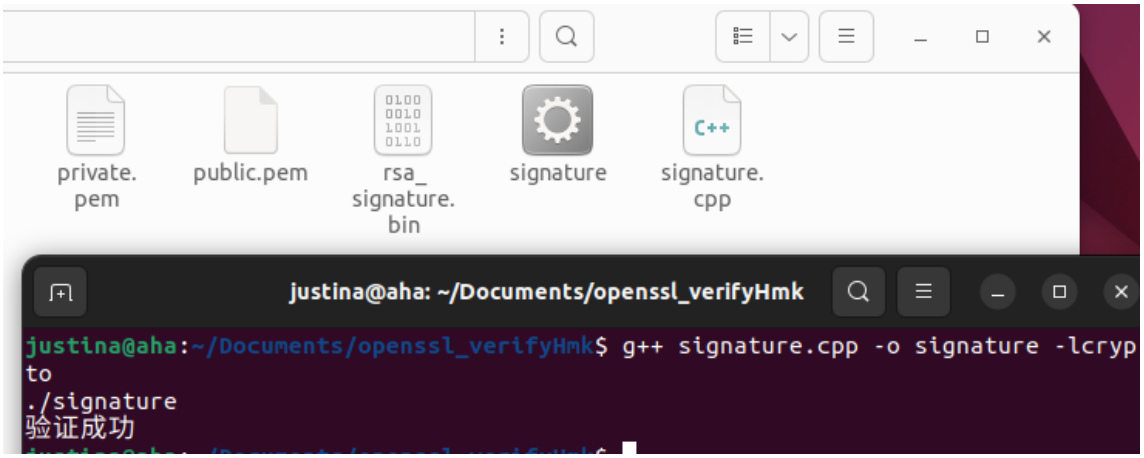
然后在主函数里实现

```
int main()
{
 // 生成长度为 2048 的密钥
 genrsa(2048);
 const char *msg = "Hello World!";
 const unsigned int msg_len = strlen(msg);
 // 存储签名
 uint8_t sign[256] = {0};
 unsigned int sign_len = 0;
 // 签名
 if (!gensign((uint8_t *)msg, msg_len, sign, &sign_len))
 {
 printf("签名失败\n");
 return 0;
 }
 // 验证签名
 if (verify((uint8_t *)msg, msg_len, sign, sign_len))
 printf("验证成功\n");
 else
 printf("验证失败\n");
 return 0;
}
```

编译并运行

```
g++ signature.cpp -o signature -lcrypto
./signature
```

可以看到数字签名“验证成功”



总结

使用OpenSSL中有关RSA的签名程序的实现，可以在命令行里通过输入命令来手动生成和验证数字签名；利用C++，我们也可以方便地实现签名生成与验证的脚本。

参考文章

【1】17 RSA — openssl_programing 1.0.1 documentation (openssl-programing.readthedocs.io)

【2】（刘哲理）数据安全教材：2.2.4，2.3.6部分