数据安全实验报告

Lab1 数字签名应用实践

网络空间安全学院 信息安全专业

2112492 刘修铭 1036

一、实验要求

参照教材 2.3.6，实现在 OpenSSL 中进行数据签名及验证这一实验。

二、实验原理

（一）OpenSSL

Openssl 是一个开源的用以实现 SSL 协议的产品，它主要包括了三个部分：密码算法库、应用程序、SSL 协议库。Openssl 实现了 SSL 协议所需要的大多数算法。

1. openssl：多用途的命令行工具
2. libcrypto：加密算法库
3. libssl：加密模块应用库，实现了 ssl 及 tls

- 加密：在 A 把要发送的内容通过单向加密方式进行数据指纹计算，计算出数据指纹后，会用自己的私钥加密数据指纹，并把加密的数据指纹添加到原数据的后面。然后对整个数据（原数据+数据指纹）进行对称加密算法进行计算（让明文转换成密文），得出的数据为整个数据的密码，再用 B 的公钥进行这个密码进行加密，并放到这个数据的后面一并发给 B。
- 解密：
    1. B 用自己的私钥能够解密——至此说明数据保密的。
    2. B 用私钥解密出来的密码，去解密整个数据得到明文（从密文转换成明文），B 拿到数据后，会用 A 的公钥去解密，能够解密说明是 A 发过来的。——到此能确定 A 的身份
    3. B 会用 A 的公钥解密出来的数据的特征码， B 会用单向加密算法对数据进行加密，得出特征码与用公钥解密出来的特征码进行比较，相同则说明数据是完整的——到此能够确定数据的完整性

（二）数字签名应用

私钥唯一、不公开且不可伪造的特性，使得**非对称密码**可以应用到数字签名中。 一个拥有公钥 k_e 和私钥 k_d 的用户，实现数字签名的过程如下：

1. 拥有公钥 k_e 的可以用其私钥 k_d 执行签名算法 S，以产生信息 m 的签名信息 sig：$sig = S_{k_d}(m)$。
2. 要验证一个签名信息 sig 是否某用户的签名时，只需要用该用户的公钥 k_e 执行验签方法计算出校验值 m'：$m' = V_{k_e}(c)$，如果 $m' = m$，那么验证成功，否则失败。

三、实验过程（含主要源代码）

为了更好地理解 OpenSSL 加密过程，笔者首先对**实验 2.1** 进行复现，即安装 OpenSSL 并进行对称密码的 CBC 等模式进行数据加密。

（一）实验 2.1

1. OpenSSL 安装

按照教材说明，检查 Ubuntu 系统中已存在的 OpenSSL，得到如下输出，说明已经有预装的 OpenSSL，版本号为 3.0.2.15。

```
lxmliu2002@lxmliu2002-Ubuntu:~$ openssl version
OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
```

2. 对称加密及工作模式

首先创建一个 message.txt 文件，将如下信息输入到文件中作为待加密的文件。

```
This is Liu Xiuming's Data_Security homework lab1.
```

接着使用 aes-128-cbc 对文件进行加密，并使用 base64 编码，输出到 ciphertext.txt 中。

> - enc 表示进行加解密
> - -e 表示进行加密
> - --aes-128-cbc 表示选用的加密算法
> - -K val 指定加密密钥
> - -iv val 提供初始向量
> - -base64 加密时在加密后进行 base64 编码，或在解密时首先对密文进行解码

```
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ openssl enc -e -aes-128-cbc -in message.txt -out ciphertext.txt -K a3171d177d1ce97ebc644ea3ff826b4e -iv 8bc65f2f883f95eea10b6f940cc805f6 -base64
```

运行后在 ciphertext.txt 中得到如下输出：

```
KPTO65KafItEPrYqNJxoYWIfHgb+KremXQkFnN+PcNsA3cp+mcfWK3ok/qzAHKtS
uurFLJHjVogy3pWMDOyVHw==
```

接着对其进行 base64 解码并解密

```
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ openssl enc -d -aes-128-cbc -in ciphertext.txt -out plaintext.txt -K a3171d177d1ce97ebc644ea3ff826b4e -iv 8bc65f2f883f95eea10b6f940cc805f6 -base64
```

可以在 plaintext.txt 中得到如下输出，可以看到其与 message.txt 中的内容一致，说明本次加解密实验成功。

```
This is Liu Xiuming's Data_Security homework lab1.
```

3. 加解密程序编写

教材上给出了 aes-128-cbc.cpp 的源代码，使用 OpenSSL 库实现了 AES-128-CBC 模式的加密和解密过程，此处对该代码的主要逻辑进行分析说明：

- `aes_128_cbc_encrypt` 函数用于对输入数据进行 AES-128-CBC 加密。
- `aes_128_cbc_decrypt`函数用于对输入数据进行 AES-128-CBC 解密。
- `main` 函数则对上述函数进行调用，实现加解密过程。
    - 定义密钥和初始化向量，定义明文消息并获取其长度。
    - 创建用于存储密文的缓冲区，并调用 aes_128_cbc_encrypt 函数进行加密，并输出加密后的密文。
    - 创建用于存储解密后的明文的缓冲区，并调用 aes_128_cbc_decrypt 函数进行解密，并输出解密后的明文。

编译该加密程序并运行，可以看到加密并解密后输出的明文 `Hello World!`，说明程序正常运行。

```
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ g++ aes-128-cbc.cpp -o aes-128-cbc -lcrypto
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ ./aes-128-cbc
Hello World!
```

（二）实验 2.2

1. 使用 OpenSSL 命令签名并验证

首先使用 OpenSSL 命令生成 2048 位 RSA 密钥，并存储到 id_rsa.key 文件中。接着根据生成的私钥文件导出 RSA 公钥文件 id_rsa.pub。

```
lxmliu2002@lxmliu2002-Ubuntu:~$ openssl genrsa -out id_rsa.key 2048
lxmliu2002@lxmliu2002-Ubuntu:~$ openssl rsa -in id_rsa.key -out id_rsa.pub -pubout
writing RSA key
```

然后使用私钥对前面已经编写好的 message.txt 文件进行签名，并将签名输出到 rsa_signature.bin 文件中。

- `dgst` 是 OpenSSL 中用于计算消息摘要的命令
- `-sha256` 指定使用 SHA-256 算法进行消息摘要生成

接着使用前面生成的公钥验证签名，可以看到输出 Verified OK，说明验证成功。

```
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ openssl dgst -sign ./id_rsa.key -out ./rsa_signature.bin -sha256 ./message.txt
lxmliu2002@lxmliu2002-Ubuntu:~/data_security/lab1$ openssl dgst -verify id_rsa.pub -signature rsa_signature.bin -sha256 message.txt
Verified OK
```

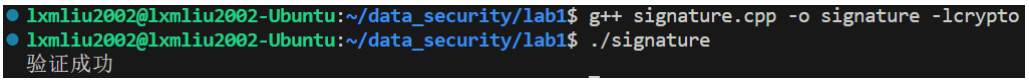
2. 数字签名程序编写

教材上给出了 signature.cpp 的源代码，使用 OpenSSL 库实现了数字签名并检验，此处对该代码的主要逻辑进行分析说明：

- `genrsa` 函数生成了一个 2048 位的 RSA 密钥对，并将私钥保存到 `private.pem` 文件中，将公钥保存到 `public.pem`文件中
- `gensign` 函数对消息进行数字签名，使用 SHA-256 摘要算法计算消息的摘要，然后使用私钥对摘要进行签名。签名的结果以二进制形式存储在指定的数组中。

- `verify` 函数验证数字签名，使用 SHA-256 摘要算法计算消息的摘要，然后使用公钥对签名进行验证。
- `main` 函数首先调用 `genrsa` 生成密钥对，调用 `gensign` 对消息进行签名，接着调用 `verify` 函数验证签名的有效性，最终输出验证结果。如果验证成功，输出"验证成功"，否则输出"验证失败"。

编译该数字签名程序并运行，可以看到进行数字签名并验证后输出 `验证成功`，说明程序正常运行。



四、实验结果及分析

本次实验中，成功实现了使用 OpenSSL 进行加解密与使用 OpenSSL 进行数字签名并验证，并进行了调用 OpenSSL 库的加解密程序、数字签名程序的阅读，对于 OpenSSL 进行了初步了解，对于相关数学原理进行温习，有助于后续课程的学习。

五、参考

本次实验主要参考教材内容完成。