

数据安全 -- 数字签名应用实践

学号：2212452 姓名：孟启轩 专业：计算机科学与技术

一、实验要求

基于第2.3.3节的示例，在OpenSSL中进行数据签名及验证的实验。

二、实验原理

（一）OpenSSL

OpenSSL 是一个开源的实现 SSL/TLS 协议的工具库，广泛应用于网络安全领域。它包含以下核心组件：

1. **命令行工具 openssl**：提供多种加密操作功能（如生成密钥、证书、加密/解密文件等）。
2. **加密算法库 libcrypto**：实现底层加密算法（如对称加密、非对称加密、哈希函数、随机数生成等）。
3. **SSL/TLS 协议库 libssl**：提供 SSL 和 TLS 协议的实现，支持安全通信的建立与维护。

加密与解密流程（以非对称加密为例）

1. 加密过程：

- **生成数据指纹**：发送方（A）对原始数据计算单向哈希（如 SHA-256），生成数据的唯一特征码（摘要）。
- **签名**：A 使用自己的**私钥**对特征码进行加密，生成**数字签名**。
- **对称加密数据**：A 生成一个随机的对称密钥（如 AES 密钥），用该密钥对原始数据进行加密，得到密文。
- **非对称加密密钥**：A 使用接收方（B）的**公钥**加密对称密钥。
- **发送数据包**：最终发送的内容包含三部分：
 - 密文（对称加密后的数据）
 - 数字签名（加密后的特征码）
 - 加密后的对称密钥

2. 解密过程：

- **验证保密性**：B 使用自己的**私钥**解密加密的对称密钥，若成功解密，说明密钥仅对 B 可见。
- **解密数据**：B 使用解密后的对称密钥解密密文，恢复原始数据。
- **验证身份与完整性**：
 - B 使用 A 的**公钥**解密数字签名，获取原始特征码。
 - B 对解密后的原始数据重新计算哈希值，并与解密后的特征码对比。
 - 若一致，说明数据未被篡改（完整性），且签名来自持有 A 私钥的用户（身份认证）。

（二）数字签名原理

数字签名基于**非对称加密**（公钥与私钥对），其核心特性是私钥的唯一性和不可伪造性。具体流程如下：

1. 签名生成

- **签名方（用户）：**

- 使用**私钥** k_d 对消息 m 的哈希值执行签名算法 S ，生成签名 sig ：
$$sig = S_{\{k_d\}}(Hash(m))$$
- 将原始消息 m 和签名 sig 一并发送给接收方。

2. 签名验证

- **接收方：**

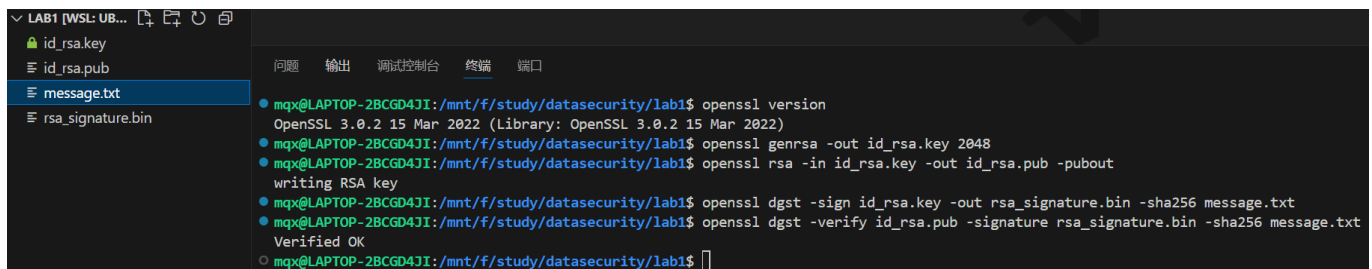
1. 使用签名方的**公钥** k_e 对签名 sig 执行验签算法 V ，计算出校验值 m' ：
$$m' = V_{\{k_e\}}(sig)$$
2. 对原始消息 m 重新计算哈希值 $Hash(m)$ ，并与 m' 比较：
 - 若 $Hash(m) = m'$ ，则验证成功，说明：
 - 消息确实由持有私钥 k_d 的用户签名（身份认证）。
 - 消息在传输过程中未被篡改（数据完整性）。
 - 否则，验证失败。

三、实验过程

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`。
- 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。
- 最后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。

```
openssl genrsa -out id_rsa.key 2048
openssl rsa -in id_rsa.key -out id_rsa.pub -pubout
openssl dgst -sign id_rsa.key -out rsa_signature.bin -sha256 message.txt
openssl dgst -verify id_rsa.pub -signature rsa_signature.bin -sha256 message.txt
```



```
LAB1 [WSL: UB...]  
id_rsa.key  
id_rsa.pub  
message.txt  
rsa_signature.bin  
问题 输出 调试控制台 终端 窗口  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ openssl version  
OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ openssl genrsa -out id_rsa.key 2048  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ openssl rsa -in id_rsa.key -out id_rsa.pub -pubout  
writing RSA key  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ openssl dgst -sign id_rsa.key -out rsa_signature.bin -sha256 message.txt  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ openssl dgst -verify id_rsa.pub -signature rsa_signature.bin -sha256 message.txt  
Verified OK  
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$
```

2. 数字签名程序编写

指导书上给出了 `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：

(1) `genrsa(int numbit)`：生成RSA密钥对

- **功能：**生成指定长度 (`numbit`) 的RSA密钥对，并保存到文件。
- **关键步骤：**

1. **创建密钥生成上下文**: 通过 `EVP_PKEY_CTX_new_id(EVP_PKEY_RSA, NULL)` 初始化RSA密钥生成环境。
2. **设置密钥长度**: 调用 `EVP_PKEY_CTX_set_rsa_keygen_bits(ctx, numbit)` 设置密钥位数 (如 2048位)。
3. **生成密钥**: 通过 `EVP_PKEY_keygen(ctx, &pkey)` 生成密钥对。
4. **保存密钥到文件**:
 - **私钥**: 使用 `PEM_write_PrivateKey` 将私钥写入 `private.pem`。
 - **公钥**: 使用 `PEM_write_PUBKEY` 将公钥写入 `public.pem`。
5. **错误处理**: 若任何步骤失败 (如文件打开失败), 跳转到 `err` 标签释放资源并返回 `false`。

(2) `gensign(...)`: 生成数字签名

- **功能**: 使用私钥对输入数据生成签名。
- **关键步骤**:
 1. **读取私钥**: 从 `private.pem` 文件中读取私钥 (`PEM_read_PrivateKey`)。
 2. **初始化签名上下文**:
 - `EVP_SignInit(ctx, EVP_sha256())`: 设置哈希算法为 SHA-256。
 3. **输入数据并计算摘要**: 通过 `EVP_SignUpdate(ctx, in, in_len)` 将消息数据输入上下文。
 4. **生成签名**: 调用 `EVP_SignFinal(ctx, out, &out_len, pkey)` 生成签名, 结果存入 `out` (长度通过 `out_len` 返回)。
 5. **资源清理**: 释放上下文和密钥。

(3) `verify(...)`: 验证数字签名

- **功能**: 使用公钥验证签名的有效性。
- **关键步骤**:
 1. **读取公钥**: 从 `public.pem` 文件中读取公钥 (`PEM_read_PUBKEY`)。
 2. **初始化验证上下文**:
 - `EVP_VerifyInit(ctx, EVP_sha256())`: 设置哈希算法为 SHA-256。
 3. **输入数据并计算摘要**: 通过 `EVP_VerifyUpdate(ctx, msg, msg_len)` 将消息数据输入上下文。
 4. **验证签名**: 调用 `EVP_VerifyFinal(ctx, sign, sign_len, pkey)` 比对签名与计算结果。
 5. **返回结果**: 若验证通过返回 `true`, 否则返回 `false`。

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。

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

```
● mxq@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ g++ signature.cpp -o signature -lcrypto
● mxq@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ ./signature
验证成功
○ mxq@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$
```

四、遇到的问题

在进行数字签名程序的编译时，遇到以下问题：

```
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$ g++ signature.cpp -o signature -lcrypto
signature.cpp:3:10: fatal error: openssl/evp.h: No such file or directory
   3 | #include <openssl/evp.h>
     |           ^~~~~~
compilation terminated.
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab1$
```

查看报错信息，应该是系统上没有正确安装 OpenSSL 开发库。

```
sudo apt-get install libssl-dev
```

```
mqx@LAPTOP-2BCGD4JI:~$ cd /mnt/f/study/datasecurity
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity$ sudo apt-get install libssl-dev
[sudo] password for mqx:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev
0 upgraded, 1 newly installed, 0 to remove and 90 not upgraded.
Need to get 2376 kB of archives.
After this operation, 12.4 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssl-dev amd64 3.0.2-0ubuntu1.19 [2376 kB]
Fetched 2376 kB in 3s (775 kB/s)
Selecting previously unselected package libssl-dev:amd64.
(Reading database ... 55118 files and directories currently installed.)
Preparing to unpack .../libssl-dev_3.0.2-0ubuntu1.19_amd64.deb ...
Unpacking libssl-dev:amd64 (3.0.2-0ubuntu1.19) ...
Setting up libssl-dev:amd64 (3.0.2-0ubuntu1.19) ...
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity$ cd lab1
```

正确安装后，重新编译程序，即可正常运行。

五、总结与感悟

通过本次实验我掌握了基于 OpenSSL 的数字签名技术简单流程，成功实现了 RSA 密钥对生成、消息签名及验证功能。深刻理解了数字签名中“私钥签名-公钥验证”机制的核心原理，即通过哈希摘要的加密与解密实现数据完整性、身份认证及不可否认性。此次实践不仅验证了数字签名技术的可行性，也为后续深入学习安全协议开发与实际应用打下了坚实基础。