武汉大学国家网络安全学院

课程实验报告

数字签名

专 业 、 班 ：	信安 9 班
课 程 名 称 ：	密码学实验
指 导 教 师 ：	王后珍
实 验 地 点 ：	C102
学 生 学 号 ：	2021302141097
学 生 姓 名 ：	李宇

2024 年 1 月 5 日

数字签名

一、 实验目的

1. 掌握数字签名的概念；
2. 掌握基于 RSA 密码、ElGamal 密码和椭圆曲线密码的数字签名方法；
3. 了解基于 RSA 密码、ElGamal 密码和椭圆曲线密码的数字签名的安全性；
4. 熟悉盲签名的原理，了解盲签名的应用。

二、 实验内容及原理

1. 掌握 RSA 数字签名的实现方案；
2. 掌握 ElGamal 数字签名的实现方案；
3. 掌握 SM2 椭圆曲线数字签名的实现方案；、
4. 了解数字签名实现中的相关优化算法。

三、 实验环境

- 操作系统：Windows11 家庭版
- 基本硬件：CPU：AMD Ryzen 5 5600H, 16G 内存
- 所用软件：Pycharm 2023.2.1 专业版
- 所用语言：python3.10

四、 实验步骤与结果

本次实验主要是基于上一周实验所实现的 RSA,ELGamal,ECC 算法（在报告中不再重复给出），将其应用于数字签名。算法本身我已经实现，关于实现源代码可见 GitHub：公开密钥密码算法 python 复现代码

因此此次的实验只是将原本对文本信息的加密解密改为先抽取摘要，再对摘要进行加密，故实现起来较为简单，相比上周的实验，代码量少了很多。

4.1 信息摘要

信息摘要（Message Digest）是通过对消息或数据应用哈希函数生成的固定长度的字符串。以下是信息摘要的一些关键特点和用途：

- **固定长度：**信息摘要的长度通常是固定的，无论输入消息的大小如何，输出的摘要长度保持不变。常见的摘要长度包括 128 位、160 位、256 位等。

- **唯一性：**不同的输入数据经过哈希函数生成的摘要应该是唯一的。即便是输入数据微小的变化，输出的摘要也应该大不相同。
- **不可逆性：**信息摘要是通过哈希函数生成的，该过程是单向的，即从摘要不容易逆向推导出原始数据。理论上，不同的输入可能产生相同的摘要（哈希冲突），但好的哈希函数应该极力避免这种情况。
- **用途：**信息摘要在密码学中有广泛的应用，包括数字签名、消息认证码、数据完整性验证等。常见的哈希函数包括 MD5、SHA-1、SHA-256 等。
- **密码学安全性：**随着计算能力的提高，一些传统的哈希函数，如 MD5 和 SHA-1，已经被发现存在漏洞，容易受到碰撞攻击。因此，现代应用更倾向于使用更安全的哈希函数，如 SHA-256 和 SHA-3。

在实际应用中，信息摘要经常用于验证文件的完整性、数字签名的生成和验证、密码存储（通过存储密码的摘要而不是明文密码）等场景。

再此次实验中直接使用 **cryptography** 中的库函数进行信息摘要过程，在摘要函数的选择上使用 **SHA256()**。

具体的代码实现上如下所示，选择语句 “Hello, this is a message to hash.” 作为原始的文本信息，先对其生成摘要，再对摘要使用不同的算法进行加解密。

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
   message = "Hello, this is a message to hash."
   message_hash = hashes.Hash(hashes.SHA256(), backend=default_backend())
   message_bytes = message.encode('utf-8')
   message_hash.update(message_bytes)
   digest = message_hash.finalize()
   # 将摘要输出为十六进制字符串
   hex_digest = digest.hex()

   print("原始摘要的十六进制表示:", hex_digest)
```

4.2 RSA 数字签名

RSA 算法的原理与代码实现已经在上周的报告中给出，此处不再赘述；修改 main 函数，对信息摘要进行加解密：

```
# RSA签名与验签
  P, Q = generate_PQ(128)
  n = caculate_n(P, Q)
  Euler_n = Euler(P, Q)
  e = find_e(Euler_n)
  d = get_d(e, Euler_n)
  cipher_text = RSA_encode(hex_digest, e, n)
```

```
    print("RSA签名信息",cipher_text)
    plain_text = RSA_decode(cipher_text, d, n)
    print("RSA验签结果",plain_text)
```

运行代码，可得下图结果：

```
D:\Anaconda\python.exe D:\python\密码学\签名算法\main.py
原始摘要的十六进制表示: 52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1
RSA签名信息 [418195493, 312500000, 503284375, 380204032, 503284375, 9509900499, 503284375, 11040808032, 254803968,
RSA验签结果 52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1

进程已结束，退出代码为 0
```

4.3 ELGamal 数字签名

ELGamal 算法的原理与代码实现已经在上周的报告中给出，此处不再赘述；修改 main 函数，对信息摘要进行加解密：

```
# ELGamal签名与验签
    p, alpha, d = 10243, 2, 666
    y = fast_modular_exponentiation(alpha, d, p)
    print("私钥d: {} ；  公钥y: {} ".format(d, y))
    # 公钥加密
    cipher_text = ELGamal_encode(hex_digest, p, alpha, y)
    print("ELGamal加密所得结果", cipher_text)
    # 私钥解密
    plain_text = ELGamal_decode(cipher_text, d, p)
    print("ELGamal解密结果", plain_text)
```

运行代码，可得下图结果：

```
D:\Anaconda\python.exe D:\python\密码学\签名算法\main.py
原始摘要的十六进制表示: 52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1
私钥d: 666 ；  公钥y: 1370
ELGamal加密所得结果 [(8449, 602), (8449, 5013), (8449, 4490), (8449, 8901), (8449, 4490), (8449, 8082), (8449, 4490),
ELGamal解密结果 52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1

进程已结束，退出代码为 0
```

4.4 ECC 数字签名

ECC 算法的原理与代码实现已经在上周的报告中给出，此处不再赘述；修改 main 函数，对信息摘要进行加解密：

```
    # ECC签名与验签
    p = int("8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3",
        16)
    a = int("787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498",
        16)
```

```
    b = int("63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A",
        16)
    n = int("8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7",
        16)
    G =
        (int("421DEBD61B62EAB6746434EBC3CC315E32220B3BADD50BDC4C4E6C147FEDD43D",
        16), \
        int("0680512BCBB42C07D47349D2153B70C4E5D7FDFCBFA36EA1A85841B9E46E09A2", 1
            6))
    d = 100
    M = 5
    Q = Gkey(d, G, a, p)

    encode_list = ECC_en_message(hex_digest, n, G, Q, a, p)
    print("ECC加密所得结果,格式为(X1,C)")
    print(encode_list)
    plain_text = ECC_de_message(encode_list, n, a, p, d)
    print("ECC解密所得结果")
    print(plain_text)
```

运行代码，可得下图结果：

```
D:\Anaconda\python.exe D:\python\密码学\签名算法\main.py
原始摘要的十六进制表示: 52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1
ECC加密所得结果,格式为(X1,C)
[((325617799887296337597797662033868409217082129659278402546287756672119907084 97, 43167126812632495228636145
ECC解密所得结果
52747c7f0125f01f7aed7cf24f1b9e332ee8ee1090fdc5fa87627c5c4b8b33b1

进程已结束，退出代码为 0
```

五、 个人收获

此次实验的较为简单，只是将上周所实现的算法应用的另一种场景下，最主要的收获便是通过此实验重温了数字签名的原理与应用，并且通过代码实现了对任意信息生成摘要并进行签名。

最后一次密码学实验结束了，通过 6 次小实验对各类加密算法有了更深的理解，而且对加密算法的实现细节有了更深的了解（光看书的话，很容易误解一些细节）；此外整个实验的所有代码我都上传到了 GitHub 上（包含此次的数字签名），如有兴趣可以通过链接密码学实验课程所有复现的代码查看。最后也感谢王后珍老师，从信安数基到密码学，再到密码学实验，传授知识点时总能以最易懂最简单的方式让我理解，也让我在这些科目上打下了坚实的基础。

参考文献

[1] 数字签名-原理（csdn）：http://t.csdnimg.cn/38Wri

[2]《密码学导论》第三版

教师评语评分

评语:

评　　分:

评 阅 人:

评阅时间: