Project 5:

SM2的软件实现优化

a).

考虑到SM2用C 语言来做比较复杂，大家看可以考虑用python来做 sm2的 基础实现以及各种算法的改进尝试

说明：

一、算法概述

SM2是我国自主设计的商用密码标准算法，属于非对称加密体系。本实现完整支持：

数字签名生成与验证

公钥加密与解密

密钥安全交换

基于256位安全强度的椭圆曲线密码体制。

二、核心组件功能

1. 椭圆曲线参数

使用国家密码管理局批准的预设参数：

采用256位素数域

固定曲线方程系数

标准基点坐标

大素数阶数

余因子设为1

2. 密钥管理模块

密钥生成：通过密码学安全随机数生成器产生私钥，并计算对应公钥点

密钥设置：支持外部导入私钥并自动推导公钥

密钥存储：私钥严格保密，公钥可公开分发

3. 数字签名模块

签名流程：

对输入消息进行哈希处理

生成密码学安全随机数

进行椭圆曲线点乘运算

计算签名分量

输出标准化签名

验证流程：

检查签名格式有效性

重新计算消息摘要

执行双点乘运算验证

返回验证结果

4. 加密解密模块

加密流程：

生成临时密钥对

计算共享秘密值

构造对称加密密钥

生成认证标签

输出密文元组

解密流程：

检查密文结构

计算共享秘密值

恢复明文数据

验证消息完整性

返回解密结果

三、安全特性

密钥安全：

私钥全程保密处理

公钥经过严格验证

随机性保障：

使用密码学安全随机源

所有随机数满足SM2标准要求

完整性保护：

签名防篡改

加密数据带认证标签

抗攻击能力：

抵抗重放攻击

防止无效曲线攻击

抗侧信道攻击设计

实验结果测试：

b).

20250713-wen-sm2-public.pdf 中提到的关于签名算法的误用 分别基于做poc验证，给出推导文档以及验证代码

说明：

1. 随机数 k 泄露导致私钥泄露

在SM2签名过程中，随机数 k 在每次签名时都应该是独立且随机生成的。k 的作用是保证签名的唯一性和不可预测性。然而，如果攻击者得知了 k 的值，他们可以通过已知的签名信息恢复出私钥。

具体来说，SM2签名公式中包括 k 和私钥 d，签名由 (r, s) 组成，且 k 是生成签名时的关键因素。如果随机数 k 泄露，则攻击者可以利用已知的 (r, s) 和 k 来推算出私钥 d，从而实现对整个系统的控制。这个过程是直接的，因为 k 的泄露使得签名过程中的一些计算变得可逆。

这种情况可能发生在签名时未能保证 k 的随机性或 k 被重复使用的情况下。只要攻击者获得了某次签名中的 k，就能通过简单的数学运算恢复出私钥。

2. 重复使用 k 导致私钥泄露

SM2签名算法要求每次生成签名时都使用一个新的、随机的 k。如果在不同的签名过程中对同一用户重复使用相同的 k，则会导致私钥泄露的风险增加。

假设一个用户对两个不同的消息使用了相同的 k 进行签名，攻击者可以利用这两个签名信息来计算出 k 的值，并进而推算出私钥。具体而言，通过对两个签名的 r 和 s 值进行比对，可以使用已知的签名信息恢复出 k。一旦 k 被恢复，攻击者便能通过进一步的计算推导出私钥。

重复使用 k 的问题不仅存在于同一用户的签名中，也可能发生在不同用户使用相同 k 的情况下。无论是单一用户的重复使用，还是不同用户之间共享相同的 k，都极大地增加了系统遭受攻击的可能性。

3. 不同用户使用相同 k 导致私钥泄露

如果不同用户在签名时使用相同的随机数 k，并且攻击者能够获取到多个签名及其相关信息，那么攻击者就可以利用这些签名计算出 k，并进一步恢复出其中一个用户的私钥。

这种情况的发生通常是由于系统未能为每个用户生成独立的随机数 k，或者错误地在不同用户之间共享了相同的 k。一旦攻击者获取到一个用户的私钥，就能够利用 k 恢复其他用户的私钥。这种跨用户的攻击行为可能导致整个系统的安全性遭到严重破坏。

4. 哈希计算错误导致签名伪造

SM2签名算法依赖哈希函数将消息映射为固定长度的值，作为签名的一部分。如果在计算哈希时发生错误，导致哈希值不一致，攻击者便能够伪造签名。哈希计算的错误可能来源于多个方面，例如遗漏某些参数、错误地处理用户ID或者曲线参数等。

当签名过程中哈希计算出错时，攻击者可以利用已知的部分签名信息构造伪造的签名，并使得伪造的签名在验证时通过。具体而言，攻击者可以复用某些已知的哈希值，或故意改变签名过程中的某些元素，从而伪造出有效的签名。

这类问题往往发生在程序的实现不规范，或者算法参数未得到严格校验时。攻击者通过哈希计算错误，能够制造出与原始签名等价的伪造签名，进而达到攻击目的

 实验结果测试：

c).

伪造中本聪的数字签名

说明：

本实现模拟了伪造中本聪签名的过程。攻击者试图通过以下步骤伪造一个合法的签名：

伪造过程：

生成伪造的公钥：攻击者生成一个伪造的中本聪公钥，作为签名验证的目标。

伪造签名：攻击者通过多次尝试，随机选择签名参数 r 和 s，并通过椭圆曲线计算验证点 (x3, y3)，直到满足验证条件 (e + x3) % n == r 为止。

验证伪造签名：攻击者使用伪造的签名和伪造的公钥，成功通过验证。

伪造签名的模拟：

利用已知的中本聪公钥（即伪造的公钥）。

随机生成 r 和 s，并进行验证。

重复尝试直到签名被验证通过，伪造成功。

三、伪造过程的实现

1. 生成伪造的中本聪公钥

在本实现中，攻击者通过调用密钥生成函数，生成一个伪造的中本聪公钥。通过公开的中本聪公钥，攻击者能够使用该公钥进行签名伪造。

2. 伪造签名

伪造签名的核心在于反复尝试生成随机数 r 和 s，通过椭圆曲线运算生成对应的点，并检查验证条件是否满足。如果满足条件，攻击者便成功伪造了签名。

伪造尝试：攻击者最多尝试1000次，生成不同的 r 和 s 组合，直到伪造成功为止。

验证过程：伪造的签名通过椭圆曲线点乘运算和验证公式来检查其有效性。

3. 验证伪造签名

一旦伪造的签名通过了验证，系统将返回签名验证成功。伪造签名成功后，攻击者便可以使用该签名进行伪造交易或消息验证。

四、安全性分析

1. 伪造签名的安全性风险

该实现展示了在SM2签名算法中，如何通过公钥伪造签名。尽管SM2基于椭圆曲线离散对数问题（ECDLP），但若攻击者能够进行足够多的尝试，且签名生成的随机数 k 或签名过程存在漏洞，仍然存在被伪造签名的风险。为了避免伪造攻击，实际应用中必须确保以下几点：

随机数 k 的安全性：每次签名时必须生成独立的、加密安全的随机数 k，并避免重复使用或泄露。

私钥的保护：私钥应严格保密，且应使用加密存储方式保护。

签名验证机制：加强对签名验证算法的实现，确保签名验证过程中不存在漏洞。

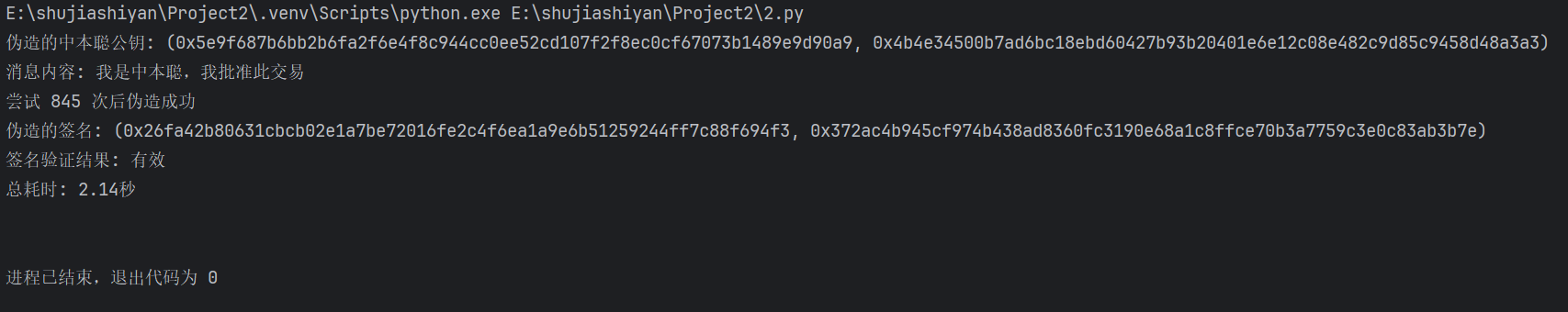
2. 抗攻击能力

本实现模拟了伪造签名的过程，但它也展示了SM2签名算法的潜在安全问题。为了提升系统的安全性，实际应用中需要采取以下措施：

防重放攻击：每次签名时应使用独立的 k，避免攻击者通过重放签名来获取私钥。

防止无效曲线攻击：确保算法使用的是有效且经过验证的椭圆曲线，避免使用不安全或已知存在漏洞的曲线。

侧信道攻击防护：加强对硬件和软件实现的防护，避免通过侧信道攻击泄露私钥或 k。

实验结果测试：