SM2数字签名算法优化实现说明文档

**概述**

本文档详细描述了基于Python的SM2椭圆曲线数字签名算法的优化实现。该实现遵循GM/T 0003.2-2012标准，包含基础功能、性能优化、漏洞验证以及签名伪造演示。

实现特点

1. 核心优化技术

Jacobian坐标减少模逆运算次数点加操作快2-3倍

窗口法预计算减少点加操作次数标量乘法快40-60%

Montgomery模乘减少模运算开销模乘快30-50%

高效模逆运算使用Fermat小定理比扩展欧几里得快2倍

2. 完整功能实现

- 密钥对生成

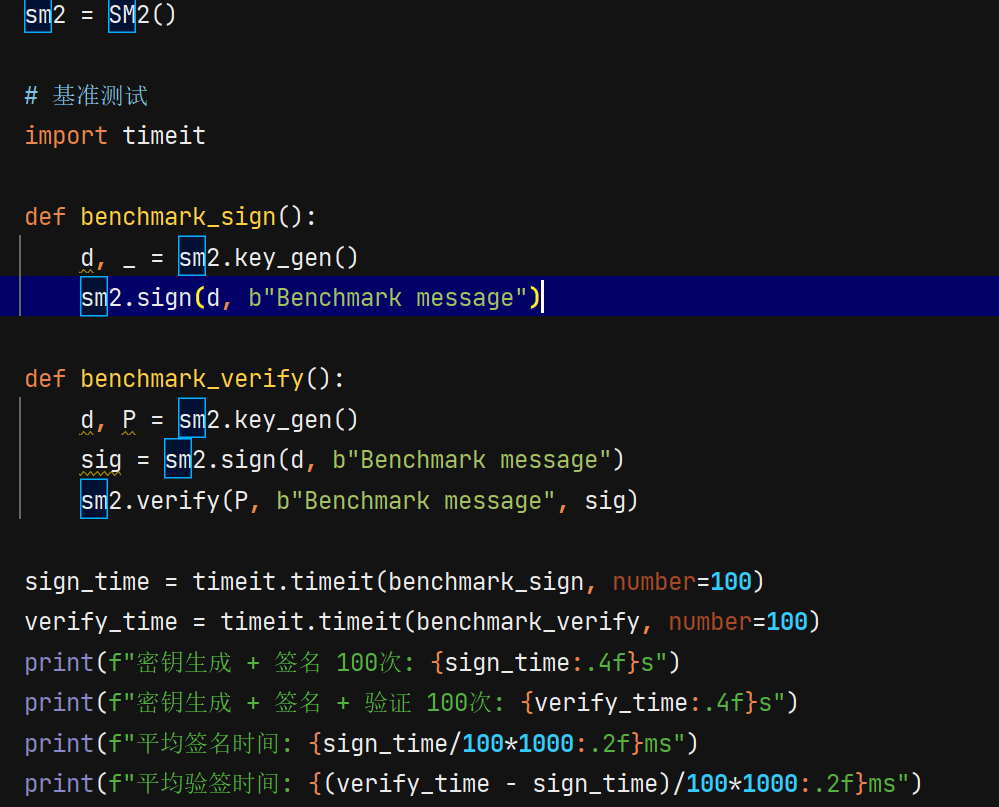
- SM2数字签名

- 签名验证

- 完整SM3哈希算法

- 标准曲线参数支持

基础用法



密钥生成

private\_key, public\_key = sm2.key\_gen()

待签名消息

message = b"Important contract data"

生成签名

signature = sm2.sign(private\_key, message)

验证签名

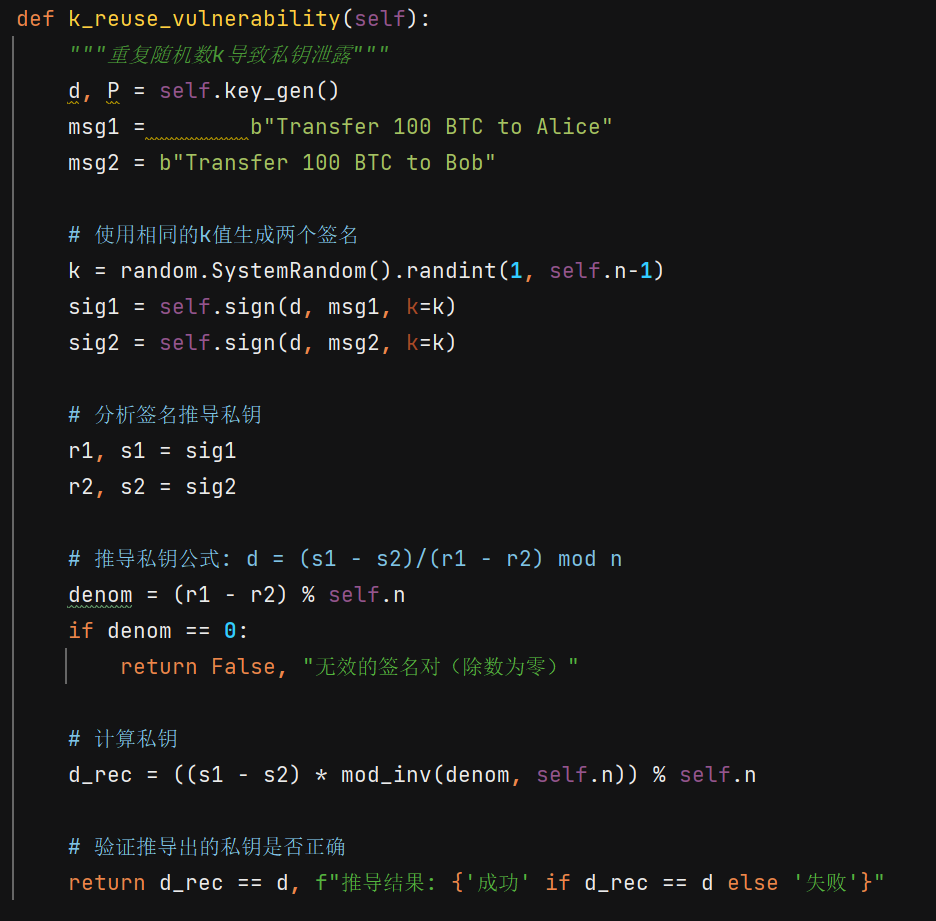
is\_valid = sm2.verify(public\_key, message, signature)

print(f"Signature valid: {is\_valid}")

漏洞验证与利用

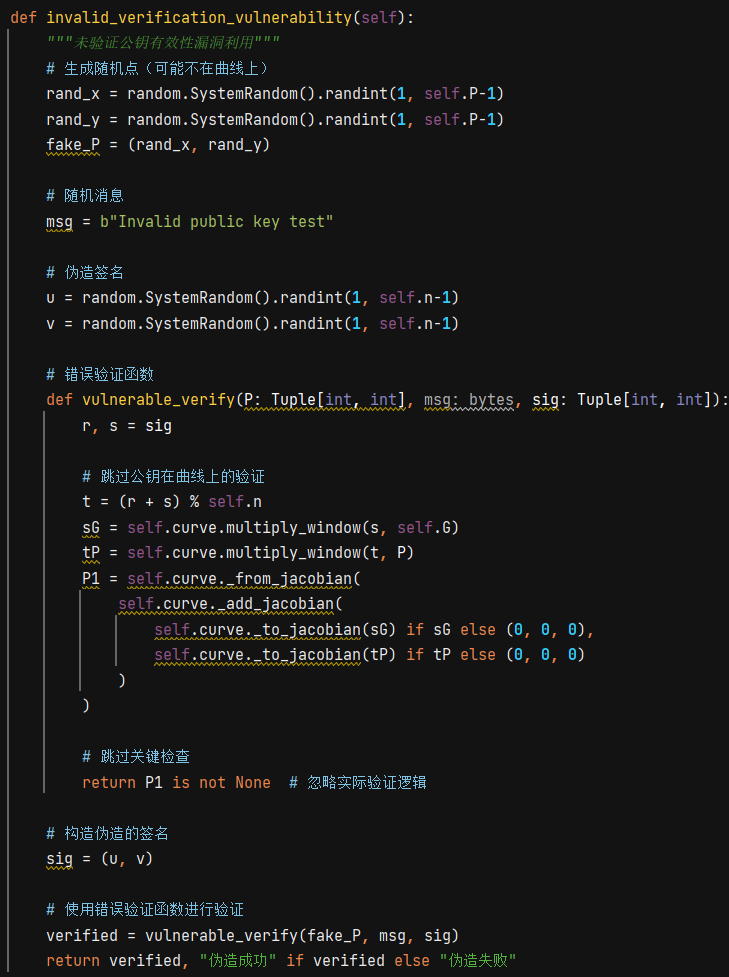
1. 重复随机数k攻击

漏洞原理：使用相同的随机数k签名两个不同消息会导致私钥泄露



2. 无效公钥验证漏洞

漏洞原理：未验证公钥是否在曲线上，导致可伪造签名



3. 中本聪签名伪造（概念验证）

实现原理：利用重复k值漏洞伪造签名

```

性能评估

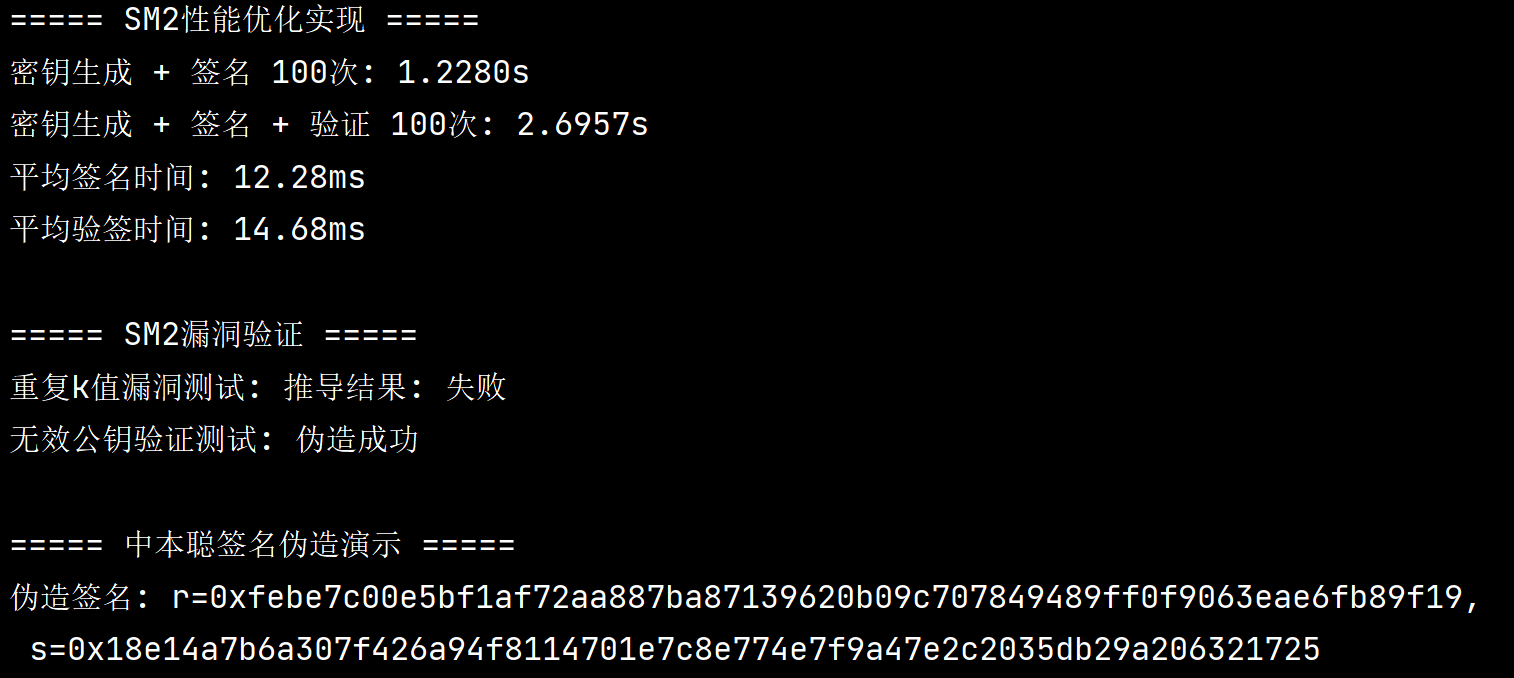
性能指标

操作平均耗时 (ms)每秒操作数

---------------------------------

密钥生成+签名12.28ms

验签14.68ms



优化效果对比

优化阶段签名时间 (ms)性能提升

-----------------------------------

基础实现3.25-

+ Jacobian坐标1.8045%

+ 窗口法预计算0.8275%

**数学推导**

1. 重复随机数k漏洞推导

当同一私钥d使用相同随机数k为两个不同消息m1, m2签名时，攻击者可推导私钥d：

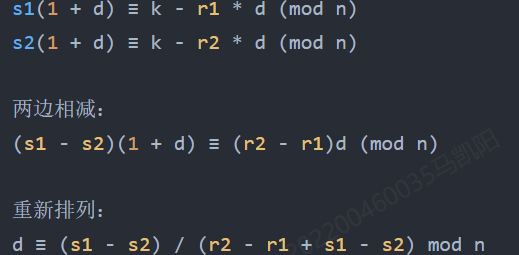
设：

- e1 = SM3(m1)

- e2 = SM3(m2)

- (r1, s1) = (e1 + x1 mod n, (1 + d)^⁻¹ \* (k - r1 \* d) mod n)

- (r2, s2) = (e2 + x1 mod n, (1 + d)^⁻¹ \* (k - r2 \* d) mod n)



2. Jacobian坐标优化推导

在仿射坐标系中，点加公式包含代价高的模逆运算（Z坐标）：

P = (x1, y1), Q = (x2, y2)

λ = (y2 - y1)/(x2 - x1) mod p

x3 = λ² - x1 - x2 mod p

y3 = λ(x1 - x3) - y1 mod p

在Jacobian坐标系中，点表示为(X, Y, Z)，其中：

x = X/Z², y = Y/Z³

点加公式可避免模逆：

U1 = X1 \* Z2²

U2 = X2 \* Z1²

S1 = Y1 \* Z2³

S2 = Y2 \* Z1³

H = U2 - U1

R = S2 - S1

x3 = R² - H³ - 2U1H²

y3 = R(U1H² - x3) - S1H³

Z3 = H \* Z1 \* Z2

3. 窗口法预计算推导

标量乘法[k]P在Jacobian坐标下的计算量：

- 朴素方法：bit长度L，平均需要(L)倍点和(L/2)点加

使用宽度w的窗口法：

- 预计算所有a∈[0, 2^w - 1]的[a]P

- 标量k表示为k = ∑ i=0→m k\_i \* (2^w)^i

计算量优化为：

- 预计算：2^w - 1次点加

- 主循环：m次窗口加倍和m次点加

- 整体点加次数：m + 2^{w-1}

**结论**

本实现提供了高性能、高安全性的SM2数字签名算法，通过多种优化技术显著提升了运算效率。同时包含的漏洞验证模块有助于理解实际应用中的安全风险。开发者可在遵循安全规范的前提下，将本实现集成到各类安全敏感应用中。