Project 5 SM2 软件实现优化

一、实验目的

1. 实现 SM2 椭圆曲线密码算法的基础签名功能，并对标量乘法进行优化，提高软件执行效率。
2. 验证 SM2 签名算法在随机数重用下的安全漏洞，通过 PoC 恢复私钥。
3. 模仿比特币风格签名流程（double SHA-256 + SM2 签名），验证签名防篡改能力。

二、实验原理

1. SM2 算法概述

SM2 是中国国家密码算法标准（GM/T 0003-2012），基于椭圆曲线公钥密码系统（ECC）。核心算法包括：

- **密钥生成**：选择私钥 $d \in [1,n-1]$，计算公钥 $P=d \cdot G$。
- **签名**：
    1. 计算消息摘要 $e=H(Z \| M)$，其中 H 为 SM3 哈希函数。
    2. 随机选择 $k \in [1,n-1]$，计算椭圆曲线点 $(x1,y1)=k \cdot G$。
    3. 计算签名分量 $r=(e+x1) \mod n$。

4. 计算签名分量 $s=((1+d)^{-1}\cdot(k-r\cdot d)) \mod n$。

- **验签**:

1. 计算 $t=(r+s) \mod n$。
2. 计算 $(x2,y2)=s\cdot G+t\cdot P$。
3. 验证 $r\equiv(e+x2) \mod n$。

签名安全性依赖于随机数 k 的唯一性和保密性。

2. 软件优化思路

传统标量乘法采用 **double-and-add** 算法，计算 [k]G[k]G[k]G 需要对每个位进行加倍和加点操作，计算量较大。优化方法包括:

1. **窗口化非相邻表示（w-NAF）**：将标量 k 表示为稀疏的奇数系数序列，减少加点次数。
2. **基点预计算**：预先计算 G 的奇数倍点列表，避免重复计算。
3. 签名和验签过程中调用优化后的 scalar_mult。

优化效果通过实验可量化：平均标量乘法时间降低，签名时间减少。

3. 随机数重用攻击（b 部分）

如果两条不同消息使用相同私钥 d 和随机数 k 生成签名，攻击者可通过解方程直接恢复私钥:

$$s_1 = (k - r_1 d)/(1 + d) \bmod n, \quad s_2 = (k - r_2 d)/(1 + d) \bmod n$$

联立方程：

$$d = (s_1 - s_2) \cdot (s_2 + r_2 - s_1 - r_1)^{-1} \bmod n$$

PoC 验证成功即可恢复原私钥，并可伪造签名。

4. 模仿比特币风格签名（c 部分）

- 使用 **double SHA-256** 对消息进行摘要。
- 使用 SM2 进行签名和验签。
- 对篡改消息重新计算摘要验证签名，应返回失败，说明签名具备防篡改能力。

三、实验环境

- 操作系统：Windows 10
- IDLE
- 自实现 SM2 算法（优化版 w-NAF + 基点预计算）

四、实验步骤与实现

1. SM2 基础实现与优化（a）

1. 实现双倍加法 double_and_add 标量乘法。
2. 优化为 **w-NAF + 基点预计算**：
    - 将标量 k 转为 w-NAF 表示。
    - 预计算基点 G 的奇数倍点列表。
    - 使用优化后的 scalar_mult_wnaf 进行签名和验签。
3. 实验结果对比：

```
----
SM2 baseline demo (double-and-add scalar mult).
Generated public key P.x = 0x1e42c8246e18c52bdf06826a3a6086d445878a5fa8094e05ec3
f98e1bd9a0d9e
Signature (r,s): 868296312614848619636961213446991843840240818772402896482878625
17398102166636 978144052060720594052693551614875621352692011028707040934909697763
272659615164
Sign time: 10.578 ms
Verify OK: True
Avg scalar mult (double-and-add): 5.249 ms
Avg sign (baseline): 5.083 ms
```

```
SM2 optimized demo (w-NAF + precomputed G).
Generated public key P.x = 0xe1d3e037a2d1a6fb2fb0ef0a0b9c62976deb3af34e71baf8b2f
0ab3a977e5e5a
Signature (r,s): 725930672419964067844504692278955129129970447497109592183595355
17003319690796 225046707041471606657603033234217159498439644159428377718955513776
639758637644
Sign time: 7.836 ms
Verify OK: True
Avg scalar mult (optimized): 5.647 ms
Avg sign (optimized): 4.020 ms
```

结果显示优化方法在签名上有明显提升。

2. SM2 随机数重用 PoC（b）

1. 使用相同私钥 ddd 和随机数 kkk 对不同消息 M1、M2 生成签名 (r1,s1)、(r2,s2)。
2. 根据公式计算私钥：

$d=(s1-s2)\cdot(s2+r2-s1-r1)-1 \bmod n$d = (s_1 - s_2) \cdot (s_2 + r_2 - s_1 - r_1)^{-1} \bmod nd=(s1−s2)⋅(s2+r2−s1−r1)−1modn

3. 验证恢复的私钥与原私钥一致。
4. 使用恢复的私钥可伪造新消息签名并验证通过。

实验输出示例：

```
=== SM2 nonce reuse PoC demo ===
Victim public key P.x = 0xe48dc228f7d3cc3071197e0d9deca1acf7c6dd7648abff809e7114a89f51dc66
Signature1 r,s: 898216562170132740940525130389583721624717248534263947470230598891399420409172396791506883737151278490064245425076811981953052402961686099368991215098204
Signature2 r,s: 236288677041956381193943461116683068145648725249746349408351413478482760020821459174296758067740463934424534191541447868371578059696353576142245081135145 2
Recovered d: 0xda9c9ff753a3b59419790ff2bab952b84699cf3a749503bb93de8ba1c02ccc27
Actual victim d: 0xda9c9ff753a3b59419790ff2bab952b84699cf3a749503bb93de8ba1c02ccc27
Match?: True
Forged signature (r,s): 822133589475727365937142132270160895078739114560419771493501829186013121603653541220353801671939944831231065672854469423623714405831957108230595637945353 0
Verify of forged signature (should be True): True
```

3. 模仿 Satoshi 风格签名（c）

1. 使用 Python 生成 SM2 密钥对。
2. 对原始消息进行 **double SHA-256** 摘要。
3. 使用 SM2 签名算法生成签名。
4. 验证签名通过。
5. 篡改消息并验证签名失败。

实验输出示例：

```
=== mimic Satoshi-style double-SHA256 but sign with SM2 (local experiment) ===
Private key: 0x5958e7a373b0d4af95370fb576a95768e4dfee2612087a035d1048737b6ed3c1
Compressed public key: 03f8a9efd9d11d1ba87e14ce7615701a1dbffed4eea3ad6910e6795c4
8cf246fd5
Original message: The Times 03/Jan/2009 Chancellor on brink of second bailout fo
r banks
Message double-SHA256: 687c09c2b4c2392a47717f58c468698b998fef0eed2ec9c8f8736d42a
1b8c26a
=== sign (prehashed) ===
Signature r: 2992799618245580850221537192566916323999294354890380802139123045883
0128594609
Signature s: 7010169231435113607594953912866625502300013451203933469184686297957
9314454209
Signature (r|s) hex: 422aa5432d598a241cbac0e2dc36c09c6d9fb5f71e22a837bd5d42572e0
6beb19afc26db0f5489b6afda6a2a17547765f0372e03512e0c23ad0b42b165cc3ec1
Verify OK: True
=== tamper test ===
Tampered message double-SHA256: 2b85509c9f691218eb9d57ee54c72f35df89b9539f932352
0b4272d15213908d
Verify tampered (should be False): False
```

五、实验分析与总结

1. **优化效果显著**：通过 w-NAF + 基点预计算，签名速度明显提高。
2. **随机数重用风险**：PoC 成功验证了 SM2 签名算法对随机数重复使用的敏感性，强调了安全签名中每次签名必须生成新的随机数。
3. **防篡改验证**：Satoshi 风格 double SHA-256 + SM2 签名流程能够有效防止消息篡改，实验验证通过。
4. **实验价值**：本实验结合优化实现、漏洞验证与签名防篡改测试，为理解椭圆曲线签名算法提供了完整实践流程。