SM2 PoC 验证

一、安全场景演示

场景 1：随机数 k 泄露导致私钥泄露

风险描述：签名过程中使用的随机数 k 是核心保密参数，一旦泄露，攻击者可直接推导出私钥。

演示流程：

1. 生成私钥 d 和消息哈希值 e
2. 生成签名(r, s)及随机数 k
3. 模拟 k 泄露，通过公式推导私钥：$d = (k - s) \cdot \text{inv}(s + r) \bmod N$

安全启示：随机数 k 必须严格保密，任何情况下不得泄露。

场景 2：同一用户重复使用随机数 k

风险描述：同一用户在多次签名中重复使用相同随机数 k ，攻击者可通过多组签名联立方程推导出私钥。

演示流程：

1. 生成私钥 d 和固定随机数 k
2. 对两个不同消息生成签名(r_1, s_1) 和(r_2, s_2)
3. 通过联立方程推导私钥：$d = (s_2 - s_1) \cdot \text{inv}(s_1 + r_1 - s_2 - r_2) \bmod N$

安全启示：每次签名必须使用独立的随机数，严禁重复使用。

场景 3：不同用户复用随机数 k

风险描述：不同用户在签名中使用相同随机数 k，攻击者可利用这些签名相互推导对方私钥。

演示流程：

生成两个用户的私钥d_a和d_b

两人使用相同随机数 k 生成签名

分别推导两个用户的私钥：$d_a = (k - s_a) \cdot \text{inv}(s_a + r_a) \bmod N$

$$d_b = (k - s_b) \cdot \text{inv}(s_b + r_b) \bmod N$$

安全启示：随机数复用不仅威胁自身安全，还可能危害其他用户。

场景 4：跨算法复用参数

风险描述：在不同密码算法（如 ECDSA 和 SM2）中复用随机数 k ，攻击者

可通过不同算法的签名方程联立推导出私钥。

演示流程：

1. 生成私钥 d 和随机数 k
2. 分别使用 ECDSA 和 SM2 算法生成签名
3. 联立两种算法的签名方程推导私钥

安全启示：不同密码算法间严禁复用随机数等敏感参数。

二、结果

```
========== RESTART: C:/Users/DELL/Desktop/project/sm2_poc.py ==
=== 测试场景1核心逻辑 ===
私钥 d: 0xb23b8c1e...
签名 r: 0x187aa445..., s: 0x11eae8fb...
随机数 k: 0x38fadc1a...
推导私钥: 0xb23b8c1e...
验证结果: 成功

=== 测试场景2核心逻辑 ===
私钥 d: 0x496da1da...
共享随机数 k: 0x48d5288f...
签名1 r: 0x13eed497..., s: 0x9f2b6d76...
签名2 r: 0x24ffe5a8..., s: 0xd25f888d...
推导私钥: 0x496da1da...
验证结果: 成功

=== 测试场景3核心逻辑 ===
Alice私钥: 0xaefc8984...
Bob私钥: 0x23eabedc...
检查Alice的1+d与N互质: 是
检查Bob的1+d与N互质: 是
共享随机数 k: 0x38e94423...
Alice签名不满足s + r与N互质，调整e_a重新生成签名...
Alice签名 r: 0xaa2341a5..., s: 0x10225a2b...
Bob签名不满足s + r与N互质，调整e_b重新生成签名...
Bob签名 r: 0xaa2341b7..., s: 0xa3c17d4d...
推导Alice私钥: 0xaefc8984...
推导Bob私钥: 0x23eabedc...
验证结果: 成功

=== 测试场景4核心逻辑 ===
私钥 d: 0xd261a7ab...
尝试随机数 k: 0xee27a984...
当前k导致分母与N的gcd=4，重新生成k...
尝试随机数 k: 0x35cabcc9...
当前k导致分母与N的gcd=7，重新生成k...
尝试随机数 k: 0xd0c0fd19...
找到合适的随机数 k: 0xd0c0fd19...
ECDSA签名 r: 0x620a210b..., s: 0x9fb134c7...
SM2签名 r: 0x620a8782..., s: 0xb66e7ff4...
推导私钥: 0xd261a7ab...
验证结果: 成功

所有测试完成
```