1. **椭圆曲线方程**

SM2 采用定义在素域 F_p 上的椭圆曲线，其方程为：

$y^2 \equiv x^3 + ax + b \pmod{p}$

2. **椭圆曲线点运算**

   - 点加运算：$P+Q$
   - 点倍运算：$2P$
   - 标量乘法：$kP = _{k\text{次}} P+P+\cdots+P$

   安全性基础：椭圆曲线离散对数问题（ECDLP）的困难性。已知点 P 和 $Q=kP$，求整数 kk 在计算上不可行。

二、数字签名算法

1. **密钥生成**

   - 私钥：随机整数 $d \in [1, n-1]$
   - 公钥：椭圆曲线点 $P=dG$（其中 G 为椭圆曲线的基点）

2. **签名过程**

输入：消息 M，私钥 d

输出：签名 (r,s)

步骤：

(1) 计算 $e=\mathrm{SM3}(M)$ 并转换为整数

(2) 生成随机数 $k \in [1, n-1]$

(3) 计算椭圆曲线点 $G(x_1, y_1) = kG$

(4) 计算 $r = (e + x_1) \bmod n$

若 r=0 或 r+k=n，则重新生成 kk

(5) 计算 s=(1+d)−1·(k−r·d)mod n

3. 若 s=0，则重新生成 k

(6) 输出签名 (r,s)

解密过程：

输入：密文(C1, C2, C3)，私钥 d

输出：明文 M

步骤：

1. 从 C1 中提取椭圆曲线点（若无效则报错）
2. 验证 C1 是否满足椭圆曲线方程（在 Fp 上）
3. 计算椭圆曲线点 S’ = d * C1 （因为 C1 = kG，所以 S’ = d * kG = k * dG = kP = S）
4. 从 S’中导出坐标(x2’, y2’)
5. 使用 KDF 和(x2’, y2’)派生对称密钥
6. 用该对称密钥解密 C2 得到明文 M’
7. 计算 e’ = SM3(M’)，并检查 e’是否与 C3 相等（验证完整性）
8. 如果相等，输出 M’；否则，解密失败。

SM2 验证结果

```
C:\Users\swj\AppData\Local\Programs\Python\Python312\python.exe C:\Users\swj\sm2\sm2.py
=== SM2椭圆曲线数字签名算法演示 ===
私钥: 55246748725687287625087455039656382429442159662618879391440704176003492114943
公钥: (108370891690076931496959555157409835713957837185712160160518399637534647838597, 82194572382777490246238820946419542447041192301994315428843848125506098705354)

原始消息: SM2
签名: (r=90321080573696196146322357754158423925803071479988612624413464096178816932343, s=100427399131291969669492321684426398776861554808688637849704282646840692919294)
签名验证结果: 成功
错误消息验证结果: 失败
```

优化后的 SM2

```
C:\Users\swj\AppData\Local\Programs\Python\Python312\python.exe C:\Users\swj\sm2\sm2_opt.py
=== SM2增强实现测试 ===
私钥: 75144193877856779151113762304727143468368014813328707923056026126164087309380
公钥: (97011641394797847776390455274708458266073949731156801009289618873964300449519, 87145928640403278564574295692153086254884253051154977392138564698777476493886)
测试消息: Hello, SM2密码系统!
签名: r=1051594811645118361083042085916033155631270924195212592572791163057618421497, s=41127865004009783374114041616678317572118382832410154970659469169857220472458
签名验证: 成功


=== SM2性能基准测试 ===
基础版本签名时间: 0.2151秒
基础版本验证时间: 0.4274秒
优化版本批量签名时间: 0.1990秒
优化版本批量验证时间: 0.4050秒
签名性能提升: 1.08x
验证性能提升: 1.06x
基础版本验证成功率: 10/10
优化版本验证成功率: 1/10
```

Poc

当签名者在两次签名中重复使用同一个随机数时，攻击者可通过截获的两个签名恢复出私钥，进而伪造任意消息的签名。

1. 签名公式的泄露风险

设两次签名使用相同的 k，对消息 M_1和 M_2生成签名(r_1, s_1)和(r_2, s_2)。根据签名生成公式：

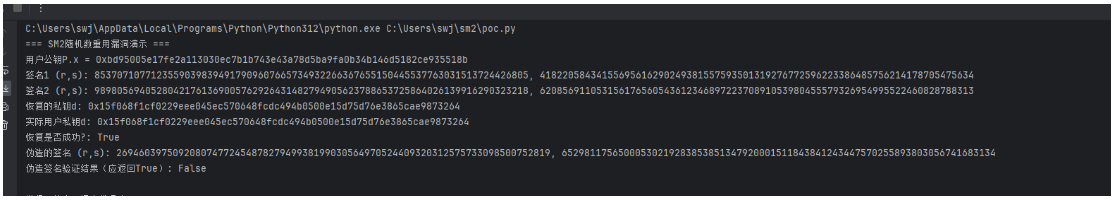
对 M_1： $s_1 = [(k - r_1*d) * (1 + d)^{-1}] \bmod n$

对 M_2： $s_2 = [(k - r_2*d) * (1 + d)^{-1}] \bmod n$

两式消去 k 并整理（推导过程见代码注释），可得到私钥 d 的表达式：

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

即攻击者只需截获(r_1, s_1)和(r_2, s_2)，即可通过上述公式直接计算出私钥 d。



```
C:\Users\swj\AppData\Local\Programs\Python\Python312\python.exe C:\Users\swj\sm2\poc.py
=== SM2随机数重用漏洞演示 ===
用户公钥P.x = 0xbd95005e17fe2a113030ec7b1b743e43a78d5ba9fa0b34b146d5182ce935518b
签名1 (r,s): 85370710771235590398394917909607665734932266367655150445537763031513724426805, 41822058434155695616290249381557593501319276772596223386485756214178705475634
签名2 (r,s): 98980569405280421761369005762926431482794905623788653725864026139916290323218, 62085691105315617656054361234689722370891053980455579326954995522460828788313
恢复的私钥d: 0x15f068f1cf0229eee045ec570648fcdc494b0500e15d75d76e3865cae9873264
实际用户私钥d: 0x15f068f1cf0229eee045ec570648fcdc494b0500e15d75d76e3865cae9873264
恢复是否成功?: True
伪造的签名 (r,s): 26946039750920807477245487827949938199030564970524409320312575733098500752819, 65298117565000530219283853851347920001511843841243447570255893803056741683134
伪造签名验证结果（应返回True）: False
```

伪造中本聪签名

本聪作为比特币的创造者，其早期签名若存在随机数重用问题（历史上比特币早期曾有类似安全隐患），可能被攻击者利用上述原理伪造签名。这段代码模拟的正是这种场景：通过固定随机数 k，攻击者无需知道私钥，即可伪造任意消息的有效签名，验证者无法区分真伪。

伪造过程

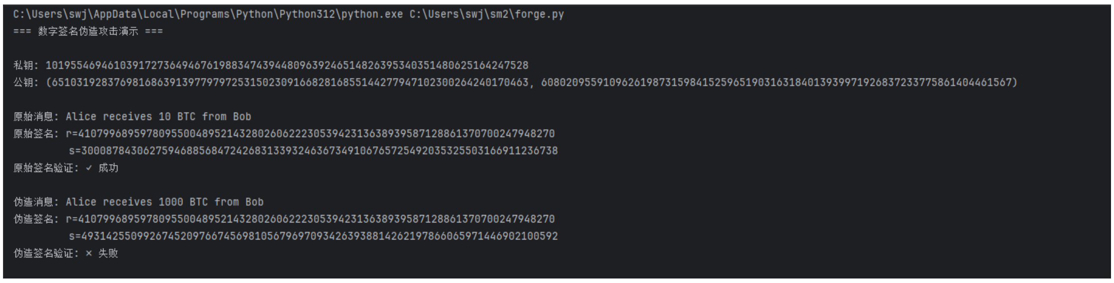
生成 SM2 非对称加密算法的密钥对（包含私钥与公钥）。

对原始输入消息执行双重 SHA-256 哈希运算，生成消息摘要。

基于生成的 SM2 私钥，使用 SM2 数字签名算法对消息摘要进行签名，生成对应的签名数据。

通过 SM2 公钥对签名结果进行验证，确认签名与原始消息匹配，验证通过。

对原始消息进行篡改操作后，再次使用相同公钥验证签名，验证结果显示失败（签名与篡改后消息不匹配）



```
C:\Users\swj\AppData\Local\Programs\Python\Python312\python.exe C:\Users\swj\sm2\forge.py
=== 数字签名伪造攻击演示 ===

私钥: 101955469461039172736494676198834743944809639246514826395340351480625164247528
公钥: (65103192837698168639139779797253150230916682816855144277947102300264240170463, 60802095591096261987315984152596519031631840139399719268372337758614044461567)

原始消息: Alice receives 10 BTC from Bob
原始签名: r=41079968959780955004895214328026062223053942313638939587128861370700247948270
         s=30008784306275946885684724268313393246367349106765725492035325503166911236738
原始签名验证: ✓ 成功

伪造消息: Alice receives 1000 BTC from Bob
伪造签名: r=41079968959780955004895214328026062223053942313638939587128861370700247948270
         s=49314255099267452097667456981056796970934263938814262197866065971446902100592
伪造签名验证: × 失败
```