

# 网络空间安全

# 创新创业实践Project 5

**学生姓名：陈卓非**

**学生学号：202200460033**

**学 院：网络空间安全学院**

**班 级：网安 2 班**

**SM2椭圆曲线密码算法实现与安全漏洞分析**

1. 引言

SM2是我国自主设计的椭圆曲线公钥密码算法，已纳入国家标准 GB/T 32918-2016，广泛应用于数字签名、密钥交换等安全场景。本实验基于 SM2 算法开展三项核心工作：

实现 SM2 基础算法（密钥生成、签名、验签）（基于20250710-fu-sm2-public.pdf）

验证 SM2 签名算法在多种误用场景下的安全漏洞（基于20250713-wen-sm2-public.pdf）

模拟中本聪数字签名伪造（基于20250714-wen-btc-public.pdf）

本报告详细阐述算法原理、数学推导、实现思路及实验验证结果，为SM2 算法的安全应用提供参考。

1. 算法原理与数学基础

2.1 SM3 哈希算法

SM3是国家标准哈希函数，输出 256 位哈希值，用于 SM2 签名中的消息摘要计算。其核心流程包括：

消息填充：将消息长度补齐至 512 位的整数倍，填充规则为：0x80 + 0x00\*k + 消息长度（64位）

消息扩展：将 512 位消息块扩展为 132 个字（68 个 W 和 64 个 W'）

压缩函数：基于 8 个初始向量（IV）和 64 轮迭代运算，更新哈希值

核心函数定义：

循环左移：\_rotl(x, n) = (x << n) | (x >> (32-n)) mod 2^32

置换函数：P0(x) = x ^ \_rotl(x,9) ^ \_rotl(x,17)，P1(x) = x ^ \_rotl(x,15) ^ \_rotl(x,23)

布尔函数：

FF(j,x,y,z) = x^y^z（j=0~15）；FF(j,x,y,z) = (x&y)|(x&z)|(y&z)（j=16~63）

GG(j,x,y,z) = x^y^z（j=0~15）；GG(j,x,y,z) = (x&y)|(~x&z)（j=16~63）

2.2 SM2 椭圆曲线密码算法

SM2基于有限域上的椭圆曲线离散对数问题（ECDLP），推荐曲线参数为：

素域：F\_q，q = 0x8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3

曲线方程：y² = x³ + a x + b mod q，其中a = 0x787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498，b = 0x63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A

基点：G = (Gx, Gy)，阶为n = 0x8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7

2.2.1 椭圆曲线基本运算

点加法：对于曲线上两点P=(x1,y1)和Q=(x2,y2)，和为R=(x3,y3)：

若P ≠ Q：斜率λ = (y2-y1)/(x2-x1) mod q，x3=λ²-x1-x2 mod q，y3=λ(x1-x3)-y1 mod q

若P = Q：斜率λ = (3x1²+a)/(2y1) mod q，x3=λ²-2x1 mod q，y3=λ(x1-x3)-y1 mod q

标量乘法：kP = P + P + ... + P（k 次），使用快速幂算法优化，时间复杂度O(log2(k))

2.2.2 SM2 签名与验签流程

签名算法（私钥d，公钥P = dG）：

计算ZA = SM3(ENTLA || IDA || a || b || Gx || Gy || xA || yA)（用户身份哈希）

计算消息哈希e = SM3(ZA || M)

生成随机数k ∈ [1, n-1]，计算(x1,y1) = kG

计算r = (e + x1) mod n（若r=0或r+k=n则重试）

计算s = [(1+d)⁻¹ · (k - r·d)] mod n（若s=0则重试）

签名为(r, s)

验签算法（公钥P）：

计算ZA和e（同签名过程）

验证r, s ∈ [1, n-1]，计算t = (r + s) mod n（若t=0则无效）

计算(x1,y1) = sG + tP

验证(e + x1) mod n = r，成立则签名有效

1. 安全漏洞验证与数学推导

3.1 场景 1：随机数 k 泄露导致私钥泄露

漏洞原理：SM2签名中k是核心秘密参数，若k泄露，攻击者可恢复私钥d。

推导过程：

由签名公式：s = (1+d)⁻¹ · (k - r·d) mod n

两边同乘(1+d)：s·(1+d) = k - r·d mod n

整理得：s + s·d = k - r·d mod n

移项合并：d·(s + r) = k - s mod n

解得：d = (k - s) · (s + r)⁻¹ mod n

验证代码：recover\_d\_from\_k\_sm2(r, s, k)函数通过上述公式计算d，实验结果显示恢复的d与原始私钥完全一致。

3.2 场景 2：同一用户复用 k 导致私钥泄露

漏洞原理：同一用户对不同消息复用k时，攻击者可通过两个签名恢复d。

推导过程：

对消息 1：s1 = (1+d)⁻¹ · (k - r1·d) mod n

对消息 2：s2 = (1+d)⁻¹ · (k - r2·d) mod n

两式相除消去k：s1·(1+d) + r1·d = s2·(1+d) + r2·d mod n

整理得：d·(s1 - s2 + r1 - r2) = s2 - s1 mod n

解得：d = (s2 - s1) · (s1 - s2 + r1 - r2)⁻¹ mod n

验证代码：recover\_d\_from\_two\_sigs\_reuse\_k(r1, s1, r2, s2)实现上述推导，实验中恢复的d与原始值一致。

3.3 场景 3：不同用户复用 k 导致私钥泄露

漏洞原理：不同用户意外使用相同k时，双方私钥均可被恢复。

推导过程：

对用户 A：dA = (k - sA) · (sA + rA)⁻¹ mod n

对用户 B：dB = (k - sB) · (sB + rB)⁻¹ mod n  
（推导同场景 1，公式具有通用性）

验证代码：recover\_d\_cross\_users(k, r, s)函数可分别恢复双方私钥，实验验证了其有效性。

3.4 场景 4：跨算法复用 k 导致私钥泄露

漏洞原理：同一私钥d和k分别用于 ECDSA 和 SM2 签名时，d可被恢复。

推导过程：

ECDSA 签名公式：s1 = k⁻¹·(e1 + d·r1) mod n → k = (e1 + d·r1)·s1⁻¹ mod n

SM2 签名公式：s2 = (1+d)⁻¹·(k - d·r2) mod n

代入k并整理：d·(r1 - s1·s2 - s1·r2) = s1·s2 - e1 mod n

解得：d = (s1·s2 - e1) · (r1 - s1·s2 - s1·r2)⁻¹ mod n

验证代码：通过联立 ECDSA 和 SM2 签名参数，实验成功恢复私钥。

3.5 场景 5：验签实现错误导致公钥泄露

漏洞原理：验签时若未正确绑定公钥（e计算错误，缺少ZA），攻击者可从签名恢复公钥。

推导过程：

错误e计算：e' = SM3(IDA || M)（正确应为SM3(ZA || M)）

由r = e' + x1 mod n得x1 = r - e' mod n（x1为kG的 x 坐标）

签名公式变形：sG + rP = kG - sG → P = (kG - sG) · (s + r)⁻¹ mod n

通过x1求解kG的可能坐标，代入计算公钥P的候选值

验证代码：recover\_pub\_from\_sig\_misuse()函数生成公钥候选集，其中包含真实公钥。

3.6 场景 6：ECDSA 签名代数伪造

漏洞原理：通过构造参数使签名验证等式成立，无需私钥即可生成 “有效” 签名。

构造过程：

随机选择u, v ∈ [1, n-1]

计算R' = uG + vP，r' = x(R') mod n

构造s' = r'·v⁻¹ mod n，e' = r'·u·v⁻¹ mod n

验证：s'⁻¹·e'·G + s'⁻¹·r'·P = uG + vP = R'，故(r', s')为e'的有效签名

验证代码：forge\_ecdsa\_algebraic\_demo()成功构造伪造签名并通过验证。

1. 实现思路与技术细节

4.1 SM3 实现（sm3.py）

消息扩展：将 512 位消息块扩展为 68 个W和 64 个W'，其中W[i]（i≥16）通过P1置换和循环左移生成

压缩函数：基于 8 个寄存器（A~H）进行 64 轮迭代，每轮通过FF、GG函数和中间变量SS1、SS2更新状态

增量计算：支持分块处理大消息，通过update()积累数据，digest()完成最终计算

4.2 SM2 实现（sm2.py）

椭圆曲线运算：点加法和标量乘法通过仿射坐标实现，处理无穷远点（O）作为单位元模逆计算使用扩展欧几里得算法，确保在素域内的逆元存在性

密钥生成：私钥d为[1, n-1]随机数，公钥P = dG通过标量乘法生成

签名优化：使用 RFC6979 标准生成确定性随机数k，避免因随机数质量问题导致的漏洞

漏洞恢复函数：针对各场景实现私钥 / 公钥恢复算法，直接对应 3.1~3.5 节的数学推导

4.3 漏洞验证（poc.py）

每个漏洞场景通过独立函数验证，流程为：

生成测试密钥对和消息

模拟漏洞场景（如泄露k、复用k）

应用恢复算法计算敏感信息（私钥 / 公钥）

验证恢复结果与原始值的一致性

实验结果均显示漏洞存在性，证明理论推导的正确性

4.4 中本聪签名伪造（forge satoshi signature.py）

伪造原理：通过构造ZA值使签名验证等式成立，步骤为：

随机选择r和s'，计算t = (r + s') mod n

计算(x1,y1) = s'G + tP，目标e值为e\_target = (r - x1) mod n

暴力搜索ZA使SM3(ZA || M) = e\_target

生成伪造签名(r, s')，配合伪造的ZA可通过验证

实现中通过限制搜索次数平衡效率，理论上存在可行解但概率极低

1. 实验结论

SM2算法的安全性依赖于随机数k的保密性和唯一性，任何k的泄露或复用都会导致私钥泄露

验签实现必须严格遵循标准，错误的e计算方式会导致公钥泄露

跨算法复用密钥材料（如k和d）会引入致命漏洞

签名伪造在特定条件下可行，需通过严格的协议设计防范