SM2 数字签名及私钥恢复的数学推导

1 SM2 数字签名算法

1.1 参数定义

- 素数模: $q = \texttt{8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3}_{16}$
- 椭圆曲线系数: $a = \texttt{787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498}_{16}$
  $b = \texttt{63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A}_{16}$
- 基点 G: $G_x = \texttt{421DEBD61B62EAB6746434EBC3CC315E32220B3BADD50BDC4C4E6C147FEDD43D}_{16}$
  $G_y = \texttt{0680512BCBB42C07D47349D2153B70C4E5D7FDFCBFA36EA1A85841B9E46E09A2}_{16}$
- G 的阶: $n = \texttt{8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7}_{16}$
- 无穷远点: $\mathcal{O}$

1.2 签名生成过程

给定私钥 d、用户标识 IDA 和消息 M:

1. 计算 $e = \text{SM3}(M) \mod n$ blue// 消息哈希
2. 生成随机数 $k \in [1, n-1]$ blue// 关键安全参数
3. 计算 $(x_1, y_1) = k \cdot G$ blue// 椭圆曲线点乘法
4. 计算 $r = (e + x_1) \mod n$ blue// 第一部分签名
5. 计算 $s = (1+d)^{-1}(k - r \cdot d) \mod n$ blue// 第二部分签名
6. 输出签名 (r, s)

1.3 签名验证过程

给定公钥 P、签名 (r, s) 和消息 M:

1. 验证 $r, s \in [1, n-1]$ blue// 范围检查
2. 计算 $e = \mathrm{SM3}(M) \mod n$ blue
3. 计算 $t = (r + s) \mod n$ blue// 中间变量
4. 计算 $(x_1, y_1) = s \cdot G + t \cdot P$ blue// 点加运算
5. 验证 $r \equiv (e + x_1) \mod n$ blue// 签名有效性检查

2 重复使用 k 导致私钥泄露的推导

2.1 问题描述

当相同的随机数 k 被用于两条不同消息 M_1 和 M_2 的签名时，攻击者可以通过两个签名 (r_1, s_1) 和 (r_2, s_2) 恢复私钥 d。

2.2 数学推导

设两条消息对应的哈希值为 $e_1 = \mathrm{SM3}(M_1) \mod n$ 和 $e_2 = \mathrm{SM3}(M_2) \mod n$。

根据签名方程:

$$\begin{aligned} &\text{签名 1:} \quad s_1 = (1+d)^{-1}(k - r_1 d) \mod n \\ &\text{签名 2:} \quad s_2 = (1+d)^{-1}(k - r_2 d) \mod n \end{aligned}$$

将方程改写为:

$$k = s_1(1+d) + r_1 d \mod n \tag{1}$$

$$k = s_2(1+d) + r_2 d \mod n \tag{2}$$

由于两个方程中的 k 相同，令方程 (1) 和 (2) 相等:

$$\begin{aligned} s_1(1+d) + r_1 d &\equiv s_2(1+d) + r_2 d \pmod{n} \\ s_1 + s_1 d + r_1 d &\equiv s_2 + s_2 d + r_2 d \pmod{n} \end{aligned}$$

将含 d 的项移到等式一侧：

$$(s_1 - s_2) \equiv d(s_2 + r_2 - s_1 - r_1) \pmod{n}$$

解出私钥 d：

$$d = \frac{s_2 - s_1}{s_1 + r_1 - s_2 - r_2} \mod n$$

2.3 恢复条件

- 分母不能为零：$s_1 + r_1 - s_2 - r_2 \not\equiv 0 \pmod{n}$
- $r_i \neq 0$ 且 $s_i \neq 0$（签名生成时已确保）
- $r_i + k \neq n$（签名生成时已检查）

3 椭圆曲线运算

3.1 点加运算

对于曲线 $E : y^2 = x^3 + ax + b$ 上的点 $P = (x_1, y_1)$ 和 $Q = (x_2, y_2)$：

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod q & P \neq Q \quad \text{blue// 点加} \\ \frac{3x_1^2 + a}{2y_1} \mod q & P = Q \quad \text{blue// 倍点} \end{cases}$$

$$x_3 = \lambda^2 - x_1 - x_2 \mod q, \quad y_3 = \lambda(x_1 - x_3) - y_1 \mod q$$

3.2 标量乘法

使用双倍-累加算法：

procedure 标量乘法 (k, P)
    $R \leftarrow \mathcal{O}$ blue// 初始化为无穷远点
    $Q \leftarrow P$ blue// 初始化临时点
    while $k > 0$ **do** blue// 遍历 k 的所有位
        if $k \bmod 2 = 1$ **then** blue// 当前位为 1
            $R \leftarrow R + Q$ blue// 点加运算
        end if

```
        Q ← Q + Q    blue// 双倍运算
        k ← ⌊k/2⌋    blue// 右移一位
    end while
    return R
end procedure
```

4 安全分析

- **关键漏洞**：重复使用 k 会导致私钥泄露
- **攻击复杂度**：$\mathcal{O}(1)$，仅需两个签名
- **防护措施**：每次签名必须使用密码学安全的随机 k
- **数学原理**：推导表明 k 的复用使签名方程形成关于 d 的可解线性系统
- **验证实验**：Python 代码实现了完整的攻击过程