SM2 数字签名算法及其伪造的数学推导

1 SM2 椭圆曲线参数

SM2 使用素数域 $\mathbb{F}_p$ 上的椭圆曲线，定义如下参数：

$$p = \texttt{0xFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFF}$$

$$a = \texttt{0xFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC}$$

$$b = \texttt{0x28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93}$$

$$n = \texttt{0xFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123}$$

$$G_x = \texttt{0x32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7}$$

$$G_y = \texttt{0xBC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0}$$

基点 $G = (G_x, G_y)$，阶为 n。

2 密钥生成

私钥 d 在 $[1, n-1]$ 范围内随机选择：

$$d \xleftarrow{\text{随机}} \{1, 2, \ldots, n-1\}$$

公钥 P 通过点乘计算：

$$P = d \cdot G = (P_x, P_y)$$

3 签名算法

3.1 输入

消息 m，用户标识 Z，私钥 d

3.2 步骤

1. 计算哈希值 $e = H(Z \parallel m) \mod n$
2. 生成随机数 $k \xleftarrow{\text{随机}} [1, n-1]$
3. 计算点 $(x_1, y_1) = k \cdot G$
4. 计算 $r = (e + x_1) \mod n$，若 $r = 0$ 或 $r + k = n$ 则重新生成 k
5. 计算 $s = (1 + d)^{-1} \cdot (k - r \cdot d) \mod n$，若 $s = 0$ 则重新生成 k
6. 输出签名 $\sigma = (r, s)$

4 验证算法

4.1 输入

公钥 P，消息 m，用户标识 Z，签名 $\sigma = (r, s)$

4.2 步骤

1. 验证 $r, s \in [1, n-1]$，否则无效
2. 计算 $e = H(Z \parallel m) \mod n$
3. 计算 $t = (r + s) \mod n$，若 $t = 0$ 则无效
4. 计算点 $(x_1, y_1) = s \cdot G + t \cdot P$
5. 计算 $R = (e + x_1) \mod n$
6. 当且仅当 $R = r$ 时签名有效

5 签名伪造原理

给定目标公钥 P，构造消息 m 和用户标识 Z 的有效签名。

5.1 伪造步骤

1. 随机选择 $s, t \in [1, n-1]$ 满足 $t \neq 0$
2. 计算 $r = (t - s) \mod n$，确保 $r \neq 0$
3. 计算点 $R = s \cdot G + t \cdot P = (x_R, y_R)$
4. 计算所需哈希值 $e = (r - x_R) \mod n$
5. 构造 m 和 Z 使得 $H(Z \parallel m) = e$
6. 输出伪造签名 $\sigma_{\text{伪造}} = (r, s)$

5.2 验证正确性

伪造签名的验证过程：

$$
\begin{aligned}
t &= (r + s) \mod n \\
\text{计算点}Q &= s \cdot G + t \cdot P \\
&= s \cdot G + (r + s) \cdot P \\
&= s \cdot G + r \cdot P + s \cdot P \\
&= s \cdot (G + P) + r \cdot P
\end{aligned}
$$

但根据构造过程：

$$
Q = s \cdot G + t \cdot P = R = (x_R, y_R)
$$

在验证步骤中：

$$R' = (e + x_R) \mod n = [(r - x_R) + x_R] \mod n = r$$

因此验证等式 $R' = r$ 成立。

6 安全分析

6.1 正确性

标准 SM2 签名验证的正确性：

$$\begin{aligned}
\text{令} \quad & (x_1, y_1) = k \cdot G \\
& r = (e + x_1) \mod n \\
& s = (1+d)^{-1}(k - rd) \mod n \\
\text{则} \quad & k = (1+d)s + rd \mod n
\end{aligned}$$

验证时：

$$\begin{aligned}
s \cdot G + t \cdot P &= s \cdot G + (r+s) \cdot d \cdot G \\
&= [s + (r+s)d] \cdot G \\
&= [s(1+d) + rd] \cdot G \\
&= [(1+d)s + rd] \cdot G \\
&= k \cdot G = (x_1, y_1)
\end{aligned}$$

因此 $R = (e + x_1) \mod n = r$。

6.2 伪造条件

签名伪造成功的必要条件：

$$\exists m, Z \quad \text{使得} \quad H(Z \parallel m) = e$$

其中 e 是预先计算的值。在实际系统中，这需要：

- 哈希函数存在原像攻击漏洞，或
- 攻击者能控制哈希计算过程

6.3 实际攻击难度

在标准实现中，这种伪造攻击不可行：

1. 哈希函数 H 需要满足**抗原像性**：给定 e，难以找到 m' 使得 $H(m') = e$
2. 实际 SM2 使用 SM3 哈希算法，具有强抗碰撞性
3. 攻击者无法控制哈希函数的输出
4. 需要同时控制用户标识 Z 和消息 m

7 安全性增强措施

实际实现中应采取以下防御措施：

- **随机数生成**：确保 k 值真随机且不可预测
- **旁路攻击防护**：防止时序分析、功耗分析等旁路攻击
- **恒定时间算法**：所有操作应具有恒定执行时间
- **输入验证**：验证所有输入点是否在曲线上
- **哈希函数加固**：使用标准化的 SM3 哈希算法
- **密钥管理**：使用硬件安全模块 (HSM) 保护私钥

8 签名伪造示例

在演示代码中，我们通过以下方式实现伪造：

1. 重写哈希函数：当输入匹配目标消息时返回预设 e 值
2. 构造 $r = t - s \mod n$
3. 计算 $R = s \cdot G + t \cdot P$
4. 反向推导 $e = r - x_R \mod n$
5. 验证时：$R' = e + x_R = (r - x_R) + x_R = r$

数学表达：

$$\begin{cases} t = r + s \\ R = s \cdot G + t \cdot P \\ e = r - x_R \\ R_{\text{验证}} = e + x_R = r \end{cases}$$

满足验证等式 $R_{\text{验证}} = r$。

9 结论

SM2 签名算法在理论上有伪造的可能性，但实际攻击需要突破哈希函数的抗原像性：

$$\Pr[\text{伪造成功}] = \Pr[H(Z \parallel m) = e] \leq \text{negl}(n)$$

其中 $\text{negl}(n)$ 是可忽略函数。因此在实际应用中，正确实现的 SM2 算法具有很高的安全性。