说明文档

一、基础实现与算法改进

1.基础实现

- 使用 Python 实现 SM2 公私钥生成、签名和验证。
- 曲线参数采用pdf文件提供的256-bit SM2 曲线参数：

```
p, a, b, Gx, Gy, n
```

- 核心函数：

```
mod_inv(x, m)         # 计算模逆
point_add(P, Q)       # 椭圆曲线点加法
scalar_mul(k, P)      # 标量乘法
sm2_sign(d, ZA_plus_M, k)   # SM2 签名
sm2_verify(P, ZA_plus_M, r, s) # SM2 验证
keygen()              # 私钥生成与公钥计算
```

2. 算法改进

- 标量乘法采用双倍-加法方法，提高运算效率。
- 提供确定性 k 生成函数 `generate_k`，避免随机 k 泄露。
- 对 r=0 或 s=0 等边界情况进行了检查与重试，提高稳健性。

二、签名算法误用及数学推导

1. 已知 k 泄露恢复私钥

当签名 `(r, s)` 和 k 已知时，私钥 d 计算公式：

$$d = (k - s) \cdot (r + s)^{-1} \mod n$$

2. 相同 k 重用恢复私钥

两条签名 `(r_1, s_1)` 和 `(r_2, s_2)` 使用相同 k，可求得：

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

3. 跨算法 k 重用恢复私钥

当同一 k 被用于 SM2 与 ECDSA，消去 k，可解得私钥 d：

$$s_{\text{ecdsa}} = k^{-1}(e_{\text{ecdsa}} + r_{\text{ecdsa}} d) \mod n$$

$$s_{\text{sm2}} = (1 + d)^{-1}(k - r_{\text{sm2}} d) \mod n$$

$$d = \frac{s_{\text{ecdsa}} s_{\text{sm2}} - e_{\text{ecdsa}}}{r_{\text{ecdsa}} - s_{\text{ecdsa}} s_{\text{sm2}} - s_{\text{ecdsa}} r_{\text{sm2}}} \mod n$$

三、实验结果

```
SM2 演示开始...
公钥在曲线上: True
签名1验证: True
泄露 k 恢复私钥正确: True
签名2验证: True
相同 k 恢复私钥正确: True
跨算法 k 恢复私钥正确: True
确定性 k 签名验证: True
演示结束。
```