SM3 软件实现与优化说明文档

密码二班：钟铭圳. 董俊豪
密码一班：张瑞豪

2025 年 8 月 9 日

目 录

1 算法概述

SM3 是中国国家密码管理局发布的密码哈希算法,采用 Merkle-Damgård 结构，包含两个核心阶段：消息扩展阶段

将 512 位消息块扩展为 132 个 32 位字（68 个 W 字和 64 个 W' 字），公式为：

$W_n = P_1(W_{n-16} \oplus W_{n-9} \oplus \text{ROTL32}(W_{n-3}, 15)) \oplus \text{ROTL32}(W_{n-13}, 7) \oplus W_{n-6}$ 压缩函数阶段

使用扩展后的消息字更新 8 个 32 位状态寄存器（A-H），经过 64 轮非线性变换

$P_1(X) = \text{ROTL32}(X, 15) \oplus \text{ROTL32}(X, 23)$

2 核心优化说明

2.1 ARM64 架构优化

实现 3×3 MDS 矩阵乘法

```
// 使用rev32指令进行高效字节序反转
uint32x4_t w0 = vreinterpretq_u32_u8(vrev32q_u8(data.val[0]));

// 向量化消息扩展（每次处理4个字）
for (int j = 16; j < 68; j += 4) {
    uint32x4_t temp = veorq_u32(veorq_u32(w_j16, w_j9),
        rot_w3);
    uint32x4_t p1 = veorq_u32(veorq_u32(temp, rot15), rot23);
    vst1q_u32(&W[j], p1);
}
```

优化效果：指令周期减少 30 通过 rev32 指令替代手动字节序转换，利用 128 位 Neon 寄存器同时处理 4 个消息字每次迭代处理 4 个字，循环次数从 52 次减少到 13 次

2.2 x86-64 架构优化

```
; AVX512指令实现
vpxorq zmm0, zmm1, zmm2   ; 512位异或操作
vprold zmm0, zmm0, 15      ; 32位左旋转15位
```

512 位寄存器同时处理 16 个消息字，AVX512 专用指令减少移位-或操作组合，单指令加载/存储 512 位数据块

核心公式

W_n = P_1 $\Big($ vpxorq$\big($W_n-16, W_n-9, vprold(W_n-3, 15)$\big)$ $\Big)$ $\oplus$vprold$(W_n-13,7)\oplus W_n-6$

3 基于 sm3 的实现，验证 length-extension attack

3.1 SM3 长度扩展攻击验证

长度扩展攻击利用 Merkle-Damgård 结构的特性：H(secret data)=SM3(secret data pad)
攻击者可构造：forge=SM3(secret data pad malicious)

```
def length_extension_attack(original_hash, data_len,
    malicious_data):
    # 1. 重建最终状态
    state = [int(original_hash[i:i+8], 16) for i in range(0,
        64, 8)]

    # 2. 计算填充
    pad = sm3_padding(data_len)
    total_len = data_len + len(pad)

    # 3. 创建伪造上下文
    ctx = SM3_CTX()
    ctx.state = state
    ctx.count = total_len * 8  # 设置正确比特数

    # 4. 添加恶意数据
    sm3_update(ctx, malicious_data)
    return sm3_final(ctx)

# 验证攻击
secret = b"secret_key"
data = b"original_data"
malicious = b"&admin=1"

orig_hash = sm3_hash(secret + data)
forge_hash = length_extension_attack(orig_hash, len(secret) +
    len(data), malicious)

# 正常计算对比
```

```
real_hash = sm3_hash(secret + data +
    sm3_padding(len(secret+data)) + malicious)
assert forge_hash == real_hash  # 攻击成功验证
```

4 Merkle 树结构设计

4.1 树结构定义

$$\begin{aligned} &\text{叶子节点: } \mathrm{Hash}(0x00 \parallel \mathrm{data}) \\ &\text{内部节点: } \mathrm{Hash}(0x01 \parallel \mathrm{left} \parallel \mathrm{right}) \\ &\text{空树根: } \mathrm{Hash}() = \mathrm{SM3}(\text{“”}) \end{aligned} \tag{1}$$

4.2 存在性证明

给定叶子 L_i，证明路径为：

$$\mathrm{Proof} = \{(sibling_j, position_j) \mid j = 0 \to h\} \tag{2}$$

其中 h 为树高，$position_j \in \{\mathrm{left}, \mathrm{right}\}$

4.3 不存在性证明

对于目标 T，提供：

1. 前驱叶子 L_p（小于 T 的最大叶子）
2. 后继叶子 L_s（大于 T 的最小叶子）
3. L_p 和 L_s 的存在性证明