Project 4: SM3的软件实现与优化

a）：与Project 1类似，从SM3的基本软件实现出发，参考付勇老师的PPT，不断对SM3的软件执行效率进行改进

b）：基于sm3的实现，验证length-extension attack

c）：基于sm3的实现，根据RFC6962构建Merkle树（10w叶子节点），并构建叶子的存在性证明和不存在性证明

说明：

1. 项目概述

本项目实现了中国国家密码管理局发布的SM3密码杂凑算法，并在此基础上进行了多项优化，同时实现了长度扩展攻击验证和基于SM3的Merkle树构建。SM3是一种密码学哈希函数，输出长度为256位(32字节)，广泛应用于数字签名、消息认证码生成和随机数生成等领域。

2. SM3算法核心实现

2.1 算法流程

SM3算法的处理流程包括以下几个主要步骤：

消息填充：将输入消息填充至长度为512位(64字节)的整数倍。填充规则为在消息末尾添加一个'1'位，然后添加若干个'0'位，最后添加一个64位的消息长度表示。

消息扩展：将每个512位的消息块扩展生成132个字(528字节)用于后续压缩处理。扩展过程使用P1置换函数和循环移位操作。

压缩函数：对每个消息块进行64轮压缩处理，每轮使用不同的布尔函数FF和GG，以及置换函数P0。

迭代处理：对每个消息块依次处理，前一个块的输出作为下一个块的输入初始值。

2.2 关键组件

布尔函数FF和GG：根据不同的轮数选择不同的逻辑运算方式，前16轮使用异或运算，后48轮使用更复杂的逻辑组合。

置换函数P0和P1：通过循环移位和异或操作实现数据扩散。

常量Tj：算法使用64个预定义的常量值，前16个相同，后48个相同。

3. 优化实现

3.1 基本优化技术

循环展开：将压缩函数的64轮循环展开为每4轮一组，减少循环控制开销。

查表法：预计算并存储常量值，避免运行时重复计算。

内联函数：将频繁调用的小函数声明为内联函数，减少函数调用开销。

3.2 SIMD优化

在支持SSE2指令集的平台上，使用SIMD指令并行处理消息扩展阶段的计算：

使用128位寄存器同时处理多个数据

并行计算消息扩展中的多个步骤

减少内存访问次数

4. 长度扩展攻击

4.1 攻击原理

SM3基于Merkle-Damgård结构，容易受到长度扩展攻击。攻击者可以在不知道原始消息的情况下，利用原始消息的哈希值构造出新消息的有效哈希。

4.2 攻击步骤

从已知哈希值恢复内部状态

构造包含原始消息填充的扩展消息

仅对扩展部分进行哈希计算

输出伪造的哈希值

4.3 防御方法

在实际应用中，可以使用HMAC或对消息进行前缀处理来防御此类攻击。

5. Merkle树实现

5.1 树结构

基于RFC6962标准构建的Merkle树具有以下特点：

叶子节点使用特殊前缀(0x00)的SM3哈希

内部节点使用特殊前缀(0x01)连接子节点哈希后的SM3哈希

支持大规模数据(10万叶子节点)

5.2 存在性证明

通过提供从叶子节点到根节点的路径上所有兄弟节点的哈希值，可以验证某个叶子确实存在于Merkle树中。

5.3 不存在性证明

通过提供目标值相邻叶子的存在性证明，可以证明某个值不存在于Merkle树中。

实验结果测试

