**任务4**

**1. SM3基础实现**

**核心功能**：中国国家密码管理局发布的密码哈希算法，输出256位摘要

**过程如下：**首先需要初始化，设置8个32位初始向量(IV)，然后进行消息填充，添加0x80字节，填充0直到长度满足(56 mod 64)，添加64位消息长度(大端序)，具体的代码如下图所示：

文本

AI 生成的内容可能不正确。

然后进行消息分块，把64字节分成一块进行处理，之后进行消息扩展，将16字→68字(W) + 64字(W')，具体的代码如下图所示：

文本, 信件

AI 生成的内容可能不正确。

最后在进行64轮迭代，前16轮使用FF0/GG0函数和常量0x79CC4519，后48轮使用FF1/GG1函数和常量0x7A879D8A，每轮更新8个工作变量(A-H)，具体的代码如下图所示：

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

**2. 优化版SM3(4轮展开)**

优化方面有：

(1)4轮循环展开：减少循环开销

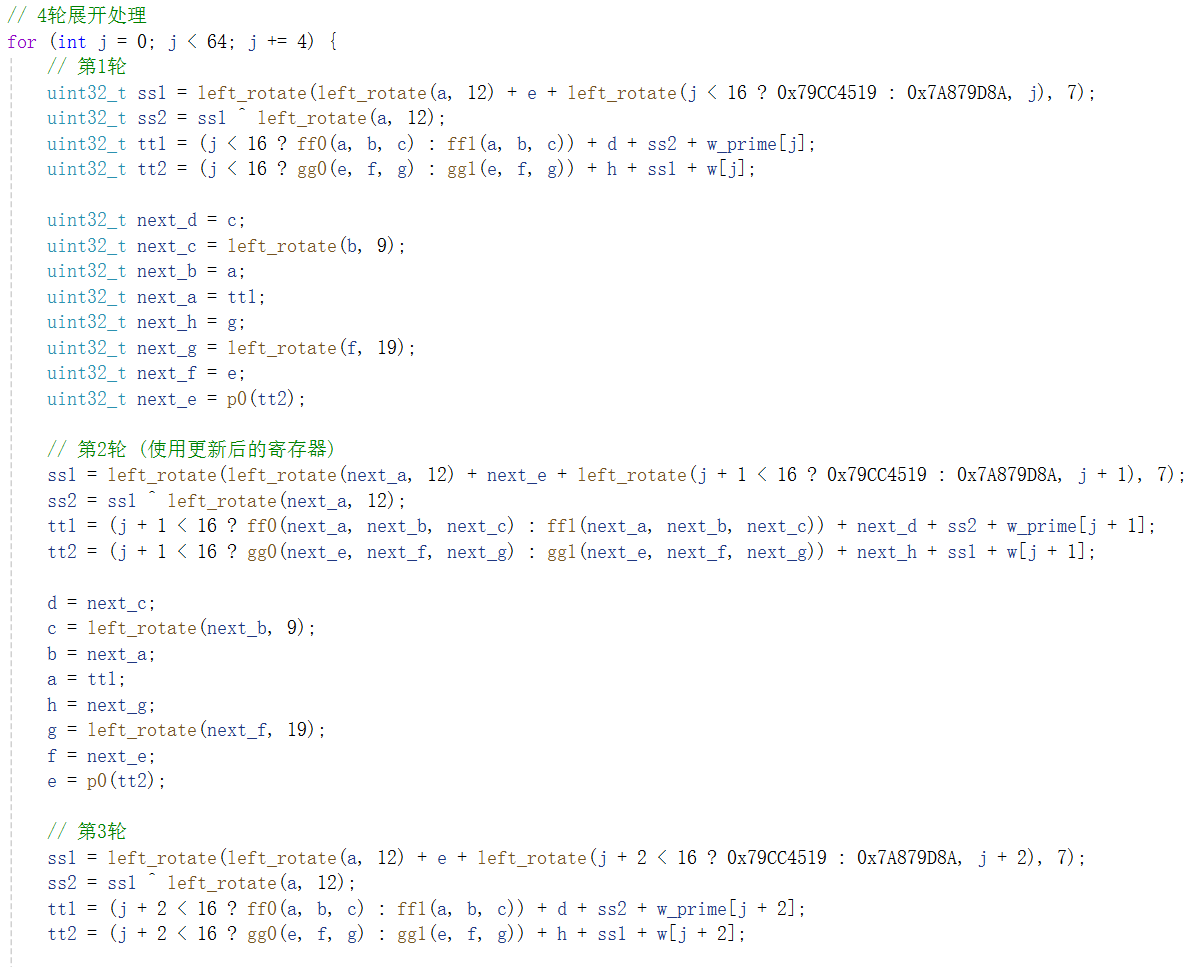
(2)寄存器重用：中间变量直接传递

(3)并行计算：4轮操作连续执行

(4)条件判断优化：每4轮判断一次轮数分界

具体的代码如下图所示：

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

**3. 长度扩展攻击**

它是利用Merkle-Damgård结构的弱点进行攻击。攻击步骤如下，首先恢复状态，将原始哈希值拆分为8个32位状态，然后使用这个公式(55 - (original\_len % 64) + 64) % 64计算填充，之后设置计数器，直接计算扩展部分的哈希，最后伪造摘要，输出扩展后的“合法”哈希，具体的代码如下图所示：



**4. Merkle树(RFC6962)**

它的结构特点是二叉树结构，叶子节点存储数据哈希，要想构建它，我们首先需要构建树，先填充叶子到2的幂，然后自底向上运算，使用这段代码实现tree[i] = hash\_node(tree[2 \* i], tree[2 \* i + 1])，完整的代码如下图所示：



之后进行存在性证明，收集从叶子到根路径的所有兄弟的节点，具体的代码如下图所示：

文本

AI 生成的内容可能不正确。

然后是不存在性证明，我们首先需要用二分查找定位插入位置，然后获取前驱节点和后继节点的证明，最后验证插入位置的正确性。具体的代码如下图所示：

图形用户界面, 应用程序

AI 生成的内容可能不正确。