# Project 4

## 一、sm3.h

头文件主要包含数据结构定义和函数声明，sm3.cpp调用。

#ifndef SM3\_H

#define SM3\_H

#include <stdint.h>

#include <stdlib.h>

// SM3 上下文结构，用于保存哈希计算过程中的状态

typedef struct {

uint32\_t state[8]; // 哈希状态寄存器

uint64\_t totalLength; // 消息总长度（比特）

size\_t bufferLength; // 当前缓冲区中的字节数

uint8\_t buffer[64]; //消息缓冲区，用于收集消息块

} SM3\_CTX;

// 初始化哈希上下文

void sm3\_init(SM3\_CTX \*ctx);

// 处理输入数据

void sm3\_update(SM3\_CTX \*ctx, const uint8\_t \*data, size\_t length);

// 完成哈希计算并输出结果

void sm3\_final(SM3\_CTX \*ctx, uint8\_t digest[32]);

// 便捷函数：一次性计算数据的哈希值

void sm3\_hash(const uint8\_t \*data, size\_t length, uint8\_t digest[32]);

// 优化版本的哈希函数

void sm3\_hash\_optimized1(const uint8\_t \*data, size\_t length, uint8\_t digest[32]);

// AESNI 优化版本（如果支持）

#ifdef \_\_AES\_\_

void sm3\_hash\_aesni(const uint8\_t \*data, size\_t length, uint8\_t digest[32]);

#endif

#endif // SM3\_H

SM3\_CTX：保存哈希计算过程中的中间状态

State[8]：8个32位寄存器，用于存储哈希中间结果

totalLength：累计处理的消息长度（以比特位单位）

buffer：64字节缓冲区，用于收集输入数据，64字节压缩

bufferLength：当前缓冲区中已填充的字节数

## sm3.cpp

1、常量与宏定义

// 常量定义

#define SM3\_T1 0x79cc4519 // 前16轮迭代常量

#define SM3\_T2 0x7a879d8a // 后48轮迭代常量

// 循环左移宏

#define ROTL32(x, n) (((x) << (n)) | ((x) >> (32 - (n))))// 布尔函数

#define FF0(x, y, z) ((x) ^ +(y)^(z)) //前16轮使用

#define FF1(x, y, z) (((x) & (y)) | (x) & (z) | (y) & (z)) // 后48轮使用

#define GG0(x, y, z) ((x) ^ (y) ^ (z))//前16轮使用

#define GG1(x, y, z) (((x) & (y)) | (~(x) & (z))// 后48轮使用

// 置换函数

#define P0(x) ((x) ^ ROTL32(x, 9) ^ ROTL32(x, 17))

#define P1(x) ((x) ^ ROTL32(x, 15) ^ ROTL32(x, 23))

定义SM3算法中使用的常量、布尔函数、置换函数和位运算操作。

2、初始化函数

sm3\_init函数设置哈希函数寄存器的初始值。

3、压缩函数

sm3\_compress处理64字节的消息块并更新哈希状态。压缩函数实现了SM3算法定义的四个步骤：消息扩展、初始化工作寄存器、64轮迭代和更新状态寄存器。

4、数据处理函数

sm3\_update函数负责处理输入数据，将数据填充到缓冲区，当缓冲区64字节时调用压缩函数进行处理。

1. 最终处理函数

sm3\_final函数完成哈希计算的最后步骤：

添加填充位（按照SM3标准，先添加一个0x80，再添加0）

添加消息总长度（以64位表示）

处理最后一个数据块

将状态寄存器的值转换为32字节的哈希结果

## 优化版本

代码中还实现了一个优化版本sm3\_hash\_optimized1，使用预计算的T表（T\_table）来避免运行时的重复计算，提高算法效率。