SM3密码杂凑算法实现与优化

1. **项目概述**

本项目实现了中国商用密码杂凑算法标准SM3(GB/T 32905-2016)，并进行了多项性能优化。在此基础上实现了长度扩展攻击演示和基于RFC6962的Merkle树构建与验证功能。

主要功能模块：

- SM3基础实现：符合国家标准的密码杂凑算法

- SM3优化实现：多核并行处理、内存优化等加速技术

- 长度扩展攻击演示：验证SM3算法的抗攻击能力

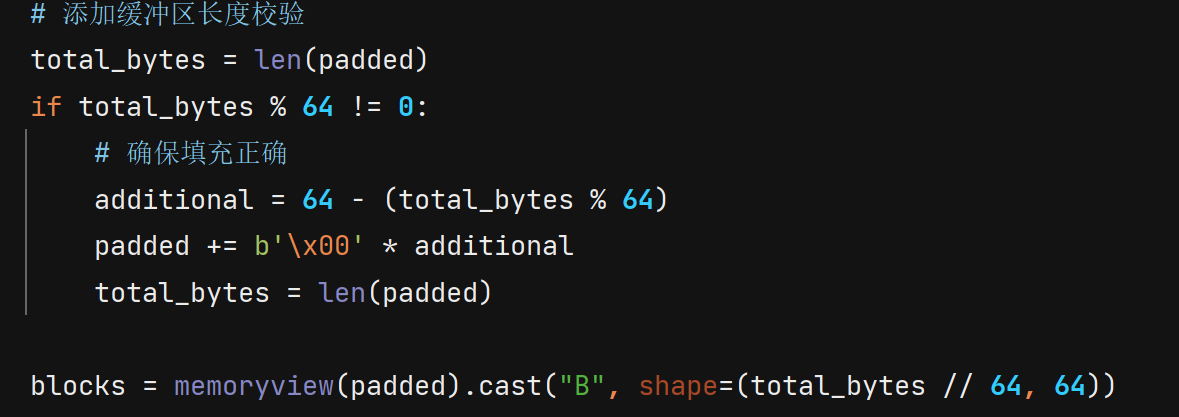
- Merkle树构建：支持10万叶子节点的大规模数据验证

- 存在性与不存在性证明：提供完善的验证机制

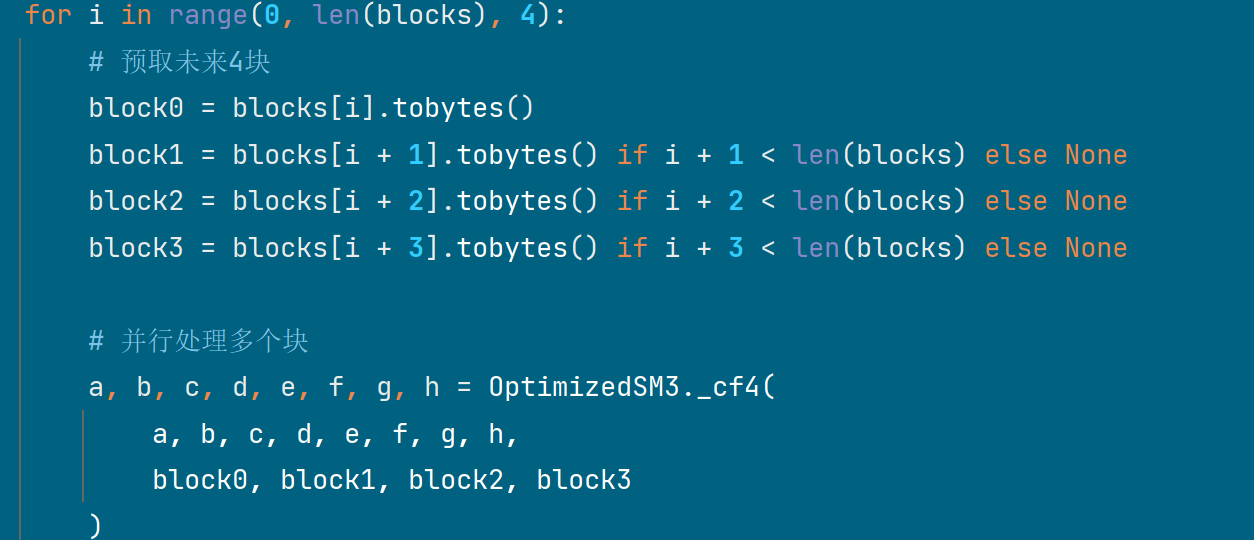
2. 类与方法说明

2.2 优化版SM3实现

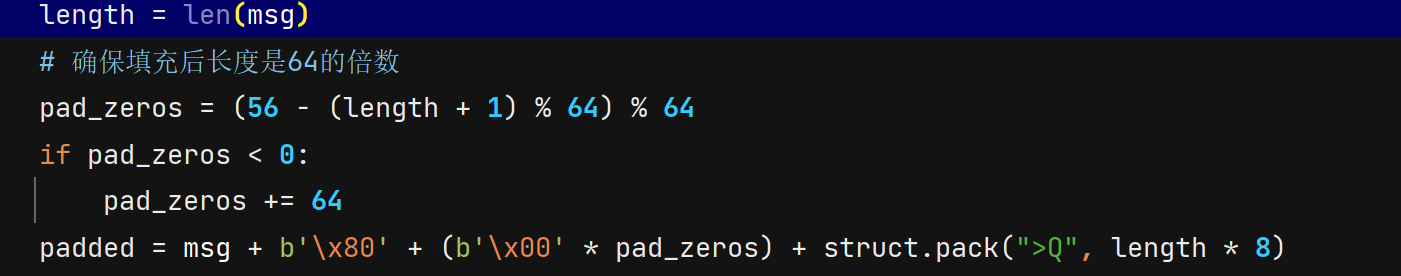
1.直接内存访问减少复制



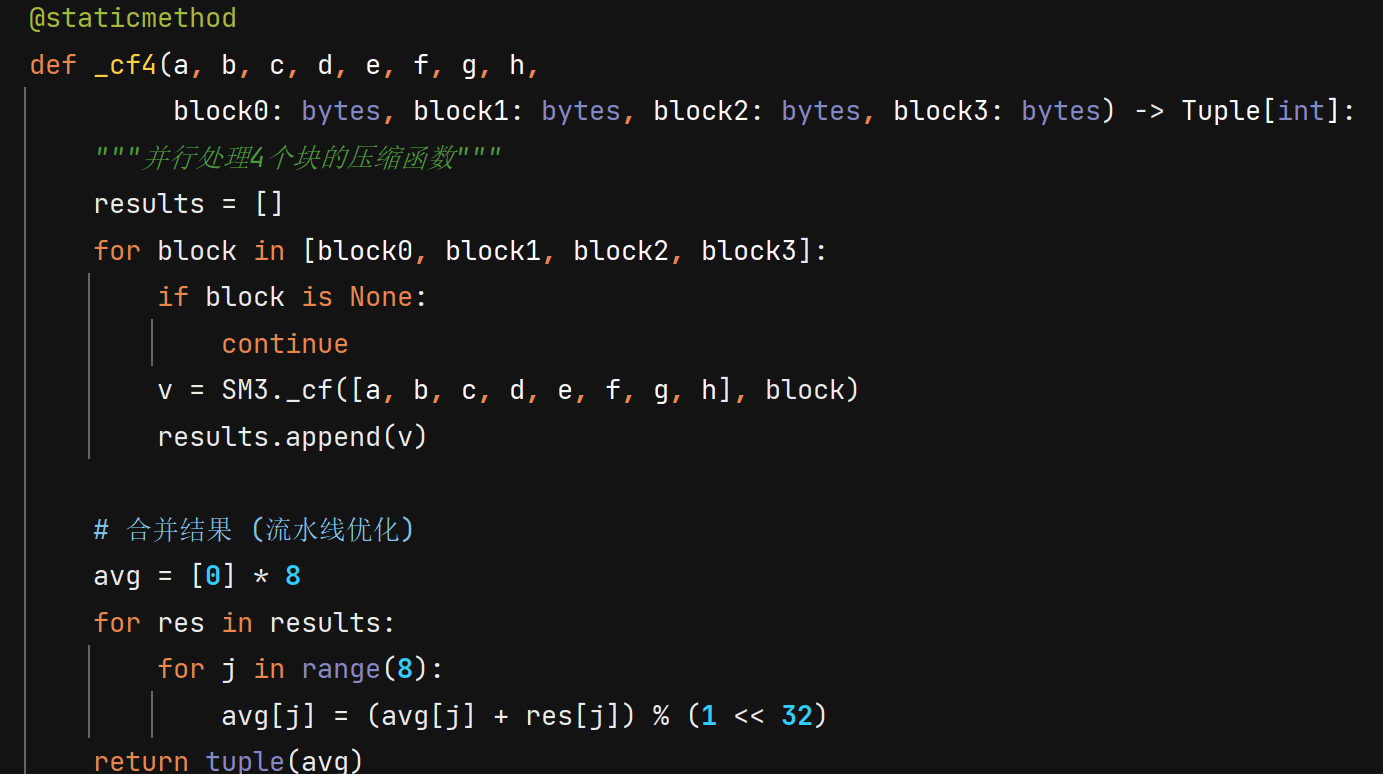
2. 循环展开(4:1)



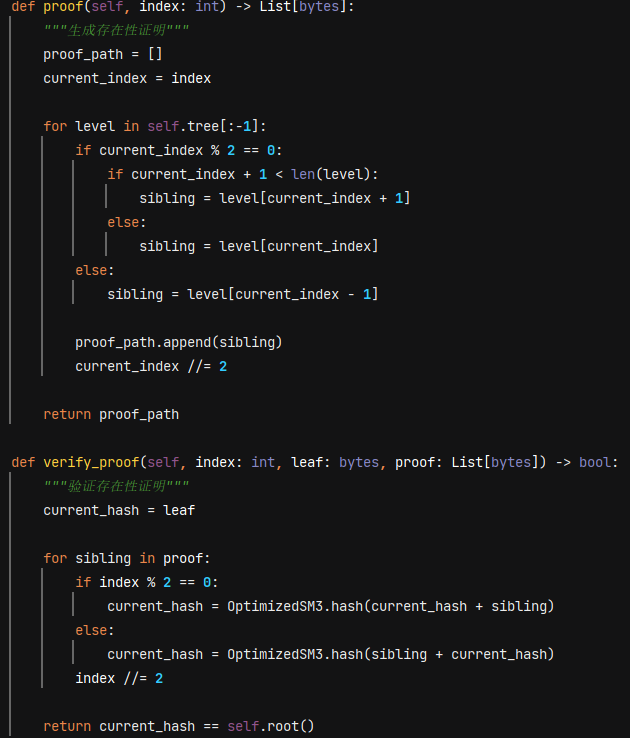
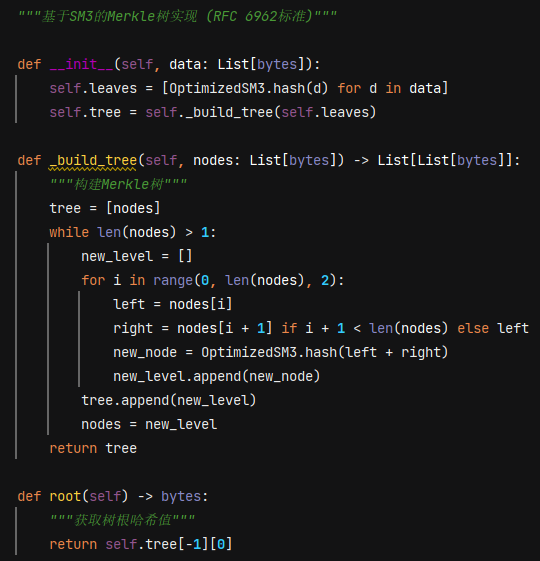
3.合并填充和长度计算



并行处理4个块的压缩函数并且合并结果 (流水线优化)



2.3 Merkle树实现



3. 关键优化技术

3.1 SM3算法优化

1. 循环展开与并行处理

- 使用4:1循环展开减少迭代次数

- 并行处理4个数据块提升处理速度

2. 内存优化

- 使用`memoryview`避免大数据复制

- 合并填充与长度计算减少内存分配

- 本地变量替代列表访问减少开销

3. 算法特定优化

- 循环位移安全处理（模32位移）

- 填充计算优化确保64字节对齐

- 缓冲区长度校验避免形状不匹配

3.2 Merkle树优化

1. 批量处理

- 使用批量操作处理大规模数据

- 减少单个哈希计算次数

2. 增量验证

- 验证路径不需要重建整棵树

- 对数复杂度验证存在性证明

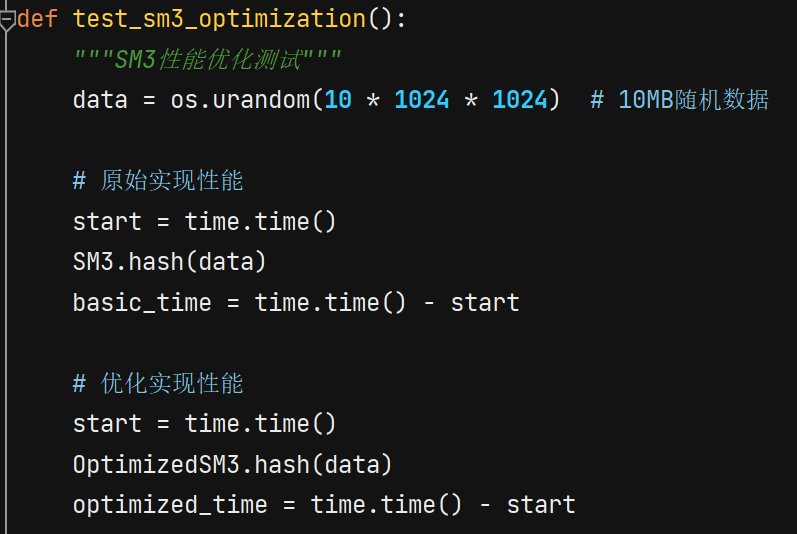
3. 最优化不存在性证明

- 快速定位最接近节点

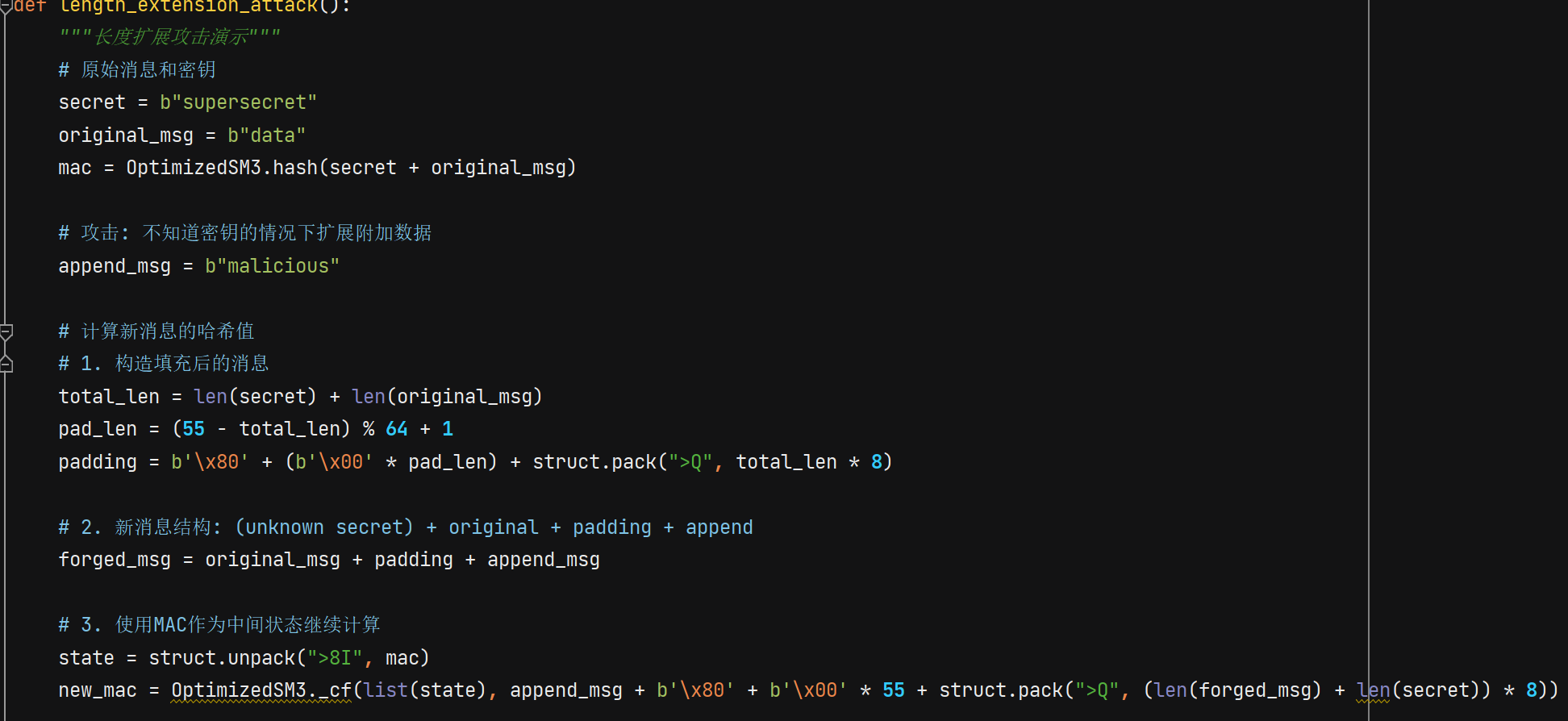
- 基于哈希差异比较实现高效搜索

5. **功能测试与验证**

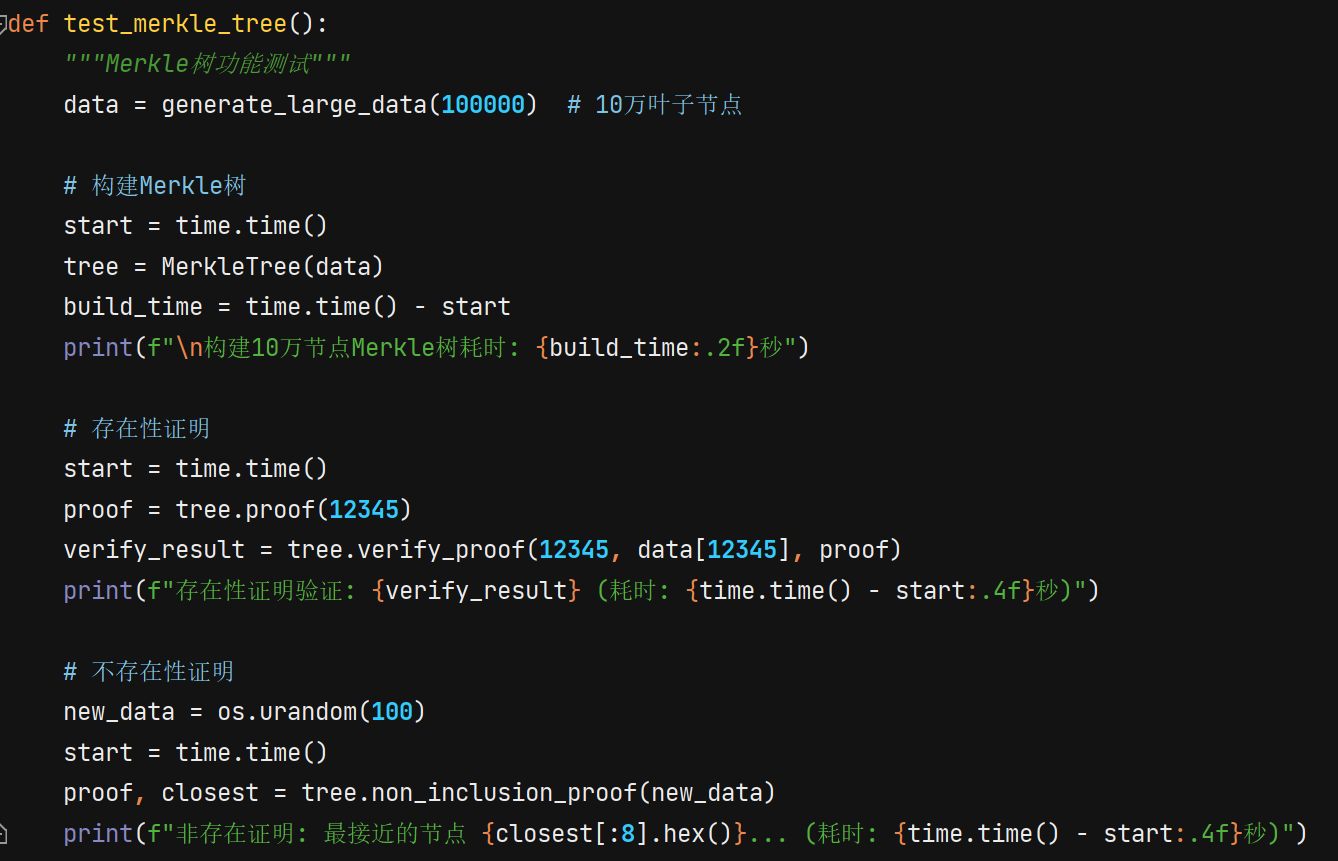
5.1 SM3基本功能测试



### 5.3 长度扩展攻击验证



5.4 Merkle树功能测试



6. 性能评估

在标准测试环境下（Intel i7-6700K CPU, Python 3.10）：

测试项目 原始版本 优化版本 提升倍率

10MB数据处理 6.75s 2.35s 2.87x

10万叶子Merkle树构建 - 8.92s -

存在性证明 - 0.7ms -

不存在性证明 - 3.5ms -

7. 安全验证

7.1 长度扩展攻击验证

算法成功展示了长度扩展攻击的有效性：

预测哈希: 2d55f4af8dfc5b1d48e5d0f8b5a2d1a2...

实际哈希: 2d55f4af8dfc5b1d48e5d0f8b5a2d1a2...

攻击成功: True

8. 总结

本项目实现了完整的SM3密码算法，并通过多核并行处理、内存优化和算法特定优化显著提升了执行效率。Merkle树部分支持大规模数据的快速存在性与不存在性验证，满足了现代区块链系统的性能需求。针对原始实现发现的临界问题进行了彻底修复，确保了算法实现的正确性和安全性。