project1 实验说明

Part 1:SM4 的基本实现

1、实验内容

做 SM4 的软件实现和优化，从基本实现出发优化 SM4 的软件执行效率，至少应该覆盖 T-table、AESNI 以及最新的指令集（GFNI、VPROLD 等）。

2、实验原理

2.1 SM4 算法基础原理

SM4 是我国自主设计的分组密码算法，采用 128 位分组长度和 128 位密钥长度，核心架构为 32 轮 Feistel 结构。

2.2 优化方法原理

2.2.1 T-table 优化

T-table 优化通过预计算合并 S 盒与 L 变换的结果，减少实时计算量。

（1）预计算 4 个 32 位表（T0-T3），每个表包含 256 项（对应 8 位输入的所有可能值），每项为 “S 盒输出 + L 变换” 的合并结果。

（2）轮函数输入拆分为 4 个字节（8 位），通过查表（T0-T3）直接获取变换结果，再经异或完成轮函数，替代原有的 “S 盒 + L 变换” 分步操作，减少移位和逻辑运算次数。

2.2.2 GFNI+AVX512 优化

利用向量化指令集提升并行处理能力。

（1）GFNI 指令：通过_mm512_gf2p8affineinv_epi64_epi8 单指令完成 8 位数据的逆变换 + 仿射变换，等效于 SM4 的 S 盒操作，大幅减少 S 盒变换的指令数；

（2）AVX512 指令：利用_mm512_rolv_epi32（高效循环移位）和_mm512_xor_si512（并行异或），实现 4 轮迭代的并行计算，提升处理效率。将轮函数结果与轮密钥异或，作为下一轮输入。

3、实验过程（大致的实验思路，具体代码见 cpp 文件）

3.1 基础实现 (SM4Basic)

根据 SM4 密钥扩展算法，将 128 位初始密钥通过非线性变换（S 盒）和线性变换，生成 32 个 32 位轮密钥，存储于数组 rk 中。再对 128 位明文分组进行 32 轮 Feistel 迭代，其中每轮输入为 4 个 32 位字(X0, X1, X2, X3)，对轮函数进行计算及迭代更新，最后得到 128 位密文。

3.2 T-table 优化 (SM4 T-Table)

针对 8 位输入的 256 种可能值，预计算 T0-T3 表：

对每个 8 位值 b，计算 T[i][b] = L(S(b))，将轮函数输入拆分为 4 个字节 b0-b3，这样可以把单轮操作简化为 “4 次查表 + 4 次异或”，减少 S 盒和移位操作的实时计算。

3.3 GFNI+AVX512 优化 (SM4 GFNI)

对向量化进行分组处理，可以使用 512 位寄存器同时加载 4 个 128 位分组，实现并行操作。

通过_mm512_gf2p8affineinv_epi64_epi8 指令，一次性对 16 个 8 位数据执行 S 盒变换（替代传统查表或逻辑运算）

利用_mm512_rolv_epi32 实现多分组的并行循环移位，_mm512_xor_si512 实现多组数据的并行异或，单次处理 4 轮迭代，提升并行效率。

Part 2: SM4-GCM 的优化实现

1、实验内容

基于 SM4 的实现，做 SM4-GCM 工作模式的软件优化实现。

2、实验原理

2.1 GCM 架构

GCM 全称为 Galois/Counter Mode，其中 G 是指 GMAC，C 是指 CTR 模式，可以将 GCM 认为是认证模式的一种，提供认证和加密两种功能。

2.2 CTR 加密

通过递增计数器生成密钥流，与明文、密文异或实现加密解密。

2.3 GMAC 认证

基于 GHASH 函数（伽罗瓦域乘法）对附加认证数据（AAD）和密文进行处理，生成认证标签，确保数据未被篡改。

3、实验过程（具体代码见 cpp 文件）

下面基于之前的 SM4 优化实现，构建完整的 SM4-GCM 工作模式，并进行多级优化。

3.1 架构封装

通过 struct SM4Wrapper 封装 SM4Basic、SM4TTable、SM4GFNI 等实现，提供统一的加解密接口。

3.2 加密流程

3.2.1 初始化计数器

根据 IV（初始向量）生成初始计数器值。将附加认证数据输入 GHASH 函数，用于后续标签计算；

3.3.2 加密数据

通过 CTR 模式生成密钥流，与明文异或得到密文。再结合 AAD 处理结果和密文，通过 GHASH 生成 128 位认证标签。

3.3.3 解密密文

初始化计数器并处理 AAD，然后通过 GHASH 计算待验证标签，与输入标签进行比对。标签验证通过后，用 CTR 模式生成密钥流，与密文异或得到明文。

3.3 GHASH 优化

使用 PCLMULQDQ 指令加速 GF 乘法，结合 4 位预计算表减少乘法次数。

Part 3:实验总结

3.1 核心类

SM4Basic	实现 SM4 基础算法，包含密钥扩展、32 轮 Feistel 迭代等核心函数
SM4TTable	基于预计算 T 表优化轮函数，包含 T 表初始化和优化后的轮迭代函数
SM4GFNI	利用 GFNI 和 AVX512 指令集，实现向量化并行处理，包含多分组并行加密函数
SM4Wrapper	封装上述 SM4 实现，提供统一的 set_key、encrypt、decrypt 接口

3.2 实验结论

基础实现验证了 SM4 算法正确性，通过 32 轮 Feistel 迭代和轮函数的正确实现，完成了 SM4 加密解密的基础功能。T-table 优化显著减少操作次数，降低了实时计算量，提升了单分组处理效率。GFNI+AVX512 优化提升并行性能，利用 512 位寄存器和向量化指令，实现了多分组的并行处理。

SM4-GCM 模式实现了认证加密一体化：通过 CTR 模式保证机密性，GMAC 模式保证完整性，且通过 PCLMULQDQ 等指令优化 GHASH 函数，在提供安全保障的同时兼顾了性能。