**实验报告**

**一、实验目的**

1. 掌握国密 SM4 对称分组密码算法的工作原理与实现方法。
2. 理解 SM4 算法中的轮函数、线性变换与 Sbox 查表的具体实现。
3. 在源码层面分析并总结本实现所采用的各种优化手段。

**二、算法原理**

**（一）SM4 概述**

SM4 是中国国家密码管理局发布的对称分组密码标准，分组长度 128bit，密钥长度 128bit，采用 32 轮非对称 Feistel 结构。每轮主要包含以下操作：

1. 轮密钥加：将当前轮的子密钥与状态字进行异或。
2. 非线性变换 Sbox：对 32bit 输入分为 4 个字节，分别通过 8×8 Sbox 查表得到输出。
3. 线性变换 L：对 Sbox 输出做循环左移并异或，产生新的 32bit 状态字。
4. 状态更新：将新状态与前四次状态按特定次序组合，继续进入下一轮。

**（二）主要函数流程**

**1. 未优化 SM4**

* **sm4\_setkey / sm4\_setkey\_enc / sm4\_setkey\_dec**：从 128bit 主密钥派生 32 个 32bit 轮密钥（子密钥），包含取大端、异或系统常量 FK、经 Sbox + 线性变换 L′后与 CK 异或等步骤。加密和解密的子密钥序列顺序相反，以支持反向运算。
* **sm4\_one\_round**：将 128bit 明文切分为四个 32bit 状态字，依次调用 sm4F（结合 Sbox + 线性变换 sm4Lt）与轮密钥，迭代 32 轮，最后输出逆序拼接的四个状态字作为一轮加密结果。
* **sm4\_crypt\_ecb / sm4\_crypt\_cbc**：
  + ECB 模式：每 16Bytes 独立加密。
  + CBC 模式：加密时先与 IV 异或再加密，并更新 IV；解密时先解密再与 IV 异或。

**2. AVX2 优化 SM4**

* **资源初始化 (init\_sm4\_resources)**：构造并缓存四张 256×32 位的查表 (T0~T3)，方便标量与向量化轮函数中快速查表。构造 AVX2 用于字节转换与掩码的 BYTE\_SWAP\_32BIT\_MASK、MASK\_0x1F 等常量。
* **密钥扩展 (sm4\_key\_schedule\_internal)**：将 128 位主密钥加载并转换端序，异或固定 FK 常量。循环 32 轮，通过 tau\_scalar (S-box + 32 位打包) 与 L\_key\_scalar 实现非线性与线性变换，生成 32 个轮密钥。
* **并行加密 8 块 (sm4\_crypt\_8blocks\_internal)**：使用宏 TRANSPOSE\_LOAD\_8BLOCKS\_TO\_SIMD 将连续 8×16 字节数据块转换为 4 条\_\_m256i 向量寄存器对齐的状态。每轮通过 SM4\_ROUND\_GATHER 宏，并行计算轮函数输入 T = X1 ^ X2 ^ X3 ^ rk，利用\_mm256\_i32gather\_epi32 从预生成的 T0~T3 表中向量化查表，聚合四次查表结果并行计算线性变换输出，更新 X0~X3 四条寄存器。循环 32 轮后，用 TRANSPOSE\_STORE\_SIMD\_TO\_8BLOCKS 转置并还原到字节序写回输出缓冲。
* **尾块标量处理**：当剩余块数不足 8 时，落回到逐块拷贝输入，按主机端序解包，用标量查表 (T0~T3) 实现轮函数，最后打包回大端序输出。

**（三）详细数学过程**

**1. SM4 加密部分**

每轮更新公式：T=Xr+1⊕Xr+2⊕Xr+3⊕rkr，Xr+4=Xr⊕Tenc (T)，r=0,1,…,31。最终输出顺序为 (X35,X34,X33,X32)。

* **Sbox 及 τ 变换**：输入 32 位字 T 拆分为四个字节 T=(b3|b2|b1|b0)，应用非线性替代：τ(T)=(S (b3)≪24)⊕(S (b2)≪16)⊕(S (b1)≪8))⊕S (b0)，其中 S (⋅) 是固定的 8×8 位替代表。
* **线性变换 Lenc**：对 τ(T) 结果 u，定义：Lenc (u)=u⊕(u⋘2)⊕(u⋘10)⊕(u⋘18)⊕(u⋘24)，其中⋘n 表示 32 位循环左移 n 位。故：Tenc (T)=Lenc (τ(T))。
* **密钥扩展 Tkey**：设主密钥 MK=(MK0,MK1,MK2,MK3)，常量 FK=[FK0,FK1,FK2,FK3] 及轮常量 CKi，初始化：Ki=MKi⊕FKi，i=0..3。第 i 个轮密钥：T′=Ki+1⊕Ki+2⊕Ki+3⊕CKi，rki=Ki⊕Tkey (T′)，Tkey (x)=x⊕(x⋘13)⊕(x⋘23)。共生成 rk0,…,rk31。

**2. GCM 模式部分**

* **Galois/Counter Mode (GCM) 原理与结构**：GCM 是一种同时提供机密性和完整性保护的分组密码工作模式，广泛应用于高速网络协议（如 TLS、IPsec）。其核心可分为计数器模式（CTR）和 GHASH 验证两部分。
* **CTR 加密部分**：选择一个初始向量 IV（通常 96 位），并构造初始计数器块。对第 i 个计数器块进行加密，得到密钥流块 Si。明文分组 Pi 与 Si 异或生成密文分组 Ci，且计数器递增。
* **GHASH 认证部分**：GHASH 在 GF (2128) 上定义，通过多项式乘法实现对数据的认证。步骤如下：
  + 密钥生成：使用零输入块生成哈希子键 H。
  + 数据分组：附加数据（AAD）分组，每组 128 位，最后一组可填充零；密文分组同样 128 位对齐。
  + GHASH 计算：定义常用符号，对两个 128 位向量 X,Y，其乘法定义为 GF (2128) 多项式乘法再模化简。首先将 AAD 和密文拼接，并在尾部附加两者的比特长度，然后按块迭代计算，最后 Xm 即为 GHASH 输出。
  + 生成认证标签：使用 CTR 加密对第零块输出加密，并与 GHASH 结果异或，最终输出 (C1,…,Cv,,T)，解密端按同样步骤重算并对比 T 以验证完整性。
* **安全性与性能**：
  + 并行性：CTR 部分所有 EK (Counteri) 均可并行计算；GHASH 乘法同样可用表查或 SIMD 加速。
  + 认证强度：标签长度通常取 128、120、112、…、32 位，可在安全需求与带宽之间权衡。
  + 常见实现要点：IV 长度若非 96 位，需先 GHASH 处理成 128 位有效 IV；补齐与尾部长度域确保 AAD 与明文紧凑对齐；防重放要求同一 IV 绝不可重复使用，否则可能破坏安全性。
* 未优化版本使用 SM4.py 实现。

以下为优化后的版本

**avx2优化sm4**

1. **向量化并行**

利用 AVX2 256 位寄存器并行处理 8 个 128 位数据块，将 8×32 位状态分量映射到四条 \_\_m256i 寄存器，实现批量加密。

1. **查表融合**

预先用标量完成 S-box + 线性变换 (L\_enc) 并填充四张 256×32 位查表 (T0~T3)，向量化轮函数只需一次 gather + XOR 即可获得完整轮函数输出。（T-table优化）

1. **高效转置**

通过 \_mm256\_unpack、\_mm256\_shuffle\_epi8 等指令，将内存连续块与 SIMD 寄存器行列转置相结合，实现字节顺序与数据布局对齐。

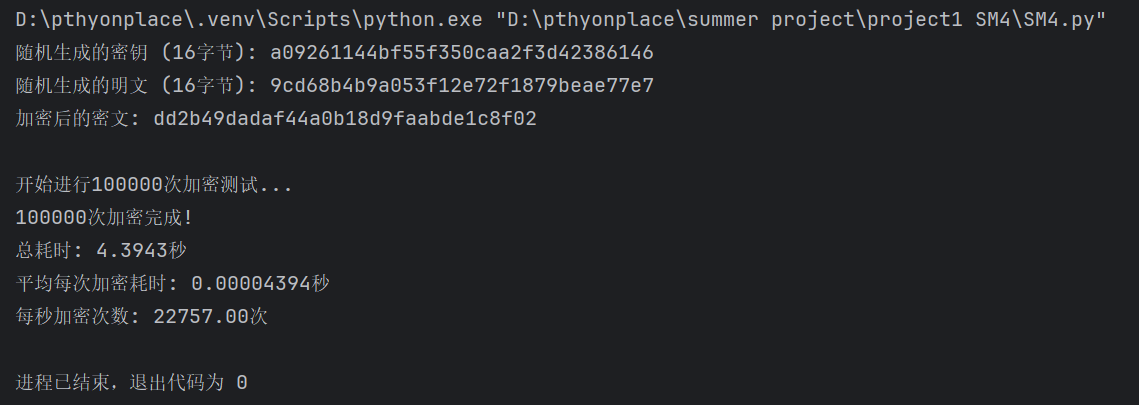
1. **端序无分支**

统一用 AVX2 掩码与 BYTE\_SWAP\_32BIT\_MASK 做字节交换，相比多分支的大端/小端处理更高效。

1. **混合调度**

针对剩余不足 8 块的数据采用标量查表路径，避免对少量数据无谓的向量化开销。

优化前的运行：



优化后的运行：

