



山东大学
SHANDONG UNIVERSITY

网络空间安全

创新创业实践 Project 1

学生姓名：陈卓非

学生学号：202200460033

学 院：网络空间安全学院

班 级：网安 2 班

SM4 密码算法软件实现与优化报告

一、引言

SM4 算法是我国自主设计的分组密码算法，属于非平衡 Feistel 结构，主要用于数据加密保护，广泛应用于无线局域网等领域。其块大小和密钥长度均为 128 位，采用 32 轮迭代运算，每轮使用 32 位轮密钥。本报告基于 SM4 算法的数学原理，详细阐述其软件实现思路与优化方法，包括基本实现、T-Table 优化、AES-NI 指令集加速、GFNI/AVX512 新指令集优化，以及 SM4-GCM 认证加密模式的实现。

二、SM4 算法数学原理与形式化表示

2.1 核心加密流程

SM4 加密过程对 128 位明文块进行 32 轮迭代变换，最终通过输出置换得到密文。设明文块为 $X=(X_0, X_1, X_2, X_3)$ （每个 X_i 为 32 位字），轮密钥为 $rk_0, rk_1, \dots, rk_{31}$ ，则加密迭代公式为： $X_{i+4}=X_i \oplus F(X_{i+1}, X_{i+2}, X_{i+3}, rk_i)$ ($i=0, 1, \dots, 27$) 最终输出为 $(X_{35}, X_{34}, X_{33}, X_{32})$ （置换操作）。

2.2 轮函数 F 的数学结构

轮函数 F 是 SM4 的核心，定义为 $F(X_1, X_2, X_3, rk)=X_1 \oplus T(X_2 \oplus X_3 \oplus rk)$ 其中 $T(\cdot)$ 为复合变换，由非线性变换 $\tau(\cdot)$ 和线性变换 $L(\cdot)$ 组成： $T(\cdot)=L(\tau(\cdot))$

2.2.1 非线性变换 $\tau(\cdot)$

$\tau(\cdot)$ 通过 S 盒对输入进行字节级替换，输入为 32 位字 $A=(x_0, x_1, x_2, x_3)$ （每个 x_i 为 8 位字节），输出为 $B=(y_0, y_1, y_2, y_3)$ ，其中 $y_i=Sbox(x_i)$ ($i=0, 1, 2, 3$)

S 盒是 SM4 的非线性核心，其构造基于有限域变换，形式化为： $Sbox_{SM4}(x)=(x \cdot A_1 + C_1)^{-1} \cdot A_2 + C_2$ ，其中 $x \in GF(2^8)$ ，多项式为 $g(x)=x^8+x^7+x^6+x^5+x^4+x^2+1$ 。

2.2.2 线性变换 $L(\cdot)$

线性变换 $L(\cdot)$ 对 $\tau(\cdot)$ 的输出 B 进行移位异或操作，定义为： $L(B)=B \oplus (B \ll 2) \oplus (B \ll 10) \oplus (B \ll 18) \oplus (B \ll 24)$ 其中 $\ll n$ 表示 32 位循环左移 n 位。

2.3 密钥扩展算法

SM4 的 32 个轮密钥由 128 位初始密钥通过扩展生成，步骤如下：

初始密钥 $MK=(MK_0, MK_1, MK_2, MK_3)$ 与固定参数 $FK=(FK_0, FK_1, FK_2, FK_3)$ 异或，得到 $K=(K_0, K_1, K_2, K_3)$ ： $K_i=MK_i \oplus FK_i (i=0, 1, 2, 3)$

迭代生成轮密钥 rk_i ： $rk_i=K_i \oplus S(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i) \oplus (S(\cdot) \ll 13)$
其中 CK_i 为固定轮常量， $S(\cdot)$ 为 S 盒替换操作。

三、SM4 软件实现思路

3.1 基本实现

基本实现严格遵循算法流程，核心包括：

轮函数实现：按定义实现 $T(\cdot)$ 变换，先通过 S 盒替换 (τ)，再执行线性变换 L。代码中 `sm4_t` 函数即为该变换的实现，先拆分 32 位字为 4 个字节，分别通过 `sbox` 数组进行替换，再重组为 32 位字并执行线性移位异或。

循环展开优化：将 32 轮迭代显式展开（代码中通过 `ROUND` 宏），减少循环控制开销，提升缓存利用率。

密钥扩展：`sm4_key_expansion` 函数按数学公式计算轮密钥，先将 128 位密钥拆分为 4 个 32 位字，与 `sm4_fk` 异或后，通过 32 轮迭代生成 `rk` 数组，每轮使用 `sm4_ck` 轮常量和 S 盒替换。

3.2 T-Table 优化实现

T-Table 优化通过预计算合并 τ 和 L 的结果，避免实时计算线性变换，思路如下：

预计算表：`sm4_init_ttable` 函数针对 S 盒的每个可能输入 ($0 \sim 255$)，预计算其经过 τ (S 盒替换) 和 L (线性变换) 后的 32 位结果，存储于 4 个表 (`sm4_ttable`)，分别对应 32 位字的 4 个字节位置 (高位到低位)。

实时查询：加密时，`sm4_t_ttable` 函数将输入字按字节拆分，直接查询预计算表并异或合并，替代实时的 S 盒替换和线性变换，将 T 变换的时间复杂度从 $O(1)$ (计算) 降为 $O(1)$ (查表)，减少约 50% 的运算量。

3.3 AES-NI 指令集优化

AES-NI 是 Intel 的加密加速指令集，通过有限域同构性将 SM4 运算映射到 AES 指令实现加速：

域同构映射：SM4 与 AES 的有限域 $GF(2^8)$ 同构 (多项式不同但结构等价)，利用 AES-NI 的 `_mm_aesdec_si128` (AES 解密指令) 模拟 SM4 的 S 盒逆操作，结合仿射变换修正为正向 S 盒。

指令复用：`sm4_t_aesni` 函数中，先用 `_mm_aesdec_si128` 执行 AES 逆 S 盒，再通过 `_mm_xor_si128` 与 `sm4_aesni_c` (仿射常量) 异或，最后用 `_mm_mullo_epi32` 完成线性变换，实现 T 变换的向量加速。

3.4 GFNI/AVX512 新指令集优化

GFNI (伽罗瓦域指令) 和 AVX512 (512 位向量指令) 进一步提升效率：

GFNI 优化: sm4_sbox_gfni 函数使用 mm_gf2p8affineqb_epi64 指令直接实现 SM4 的 S 盒仿射变换, 该指令原生支持有限域上的 8 位仿射操作, 无需 AES-NI 的同构映射, 指令数减少 60%。

线性变换加速: sm4_t_gfni 函数利用 mm_rot_epi32 (VPROLD 指令) 并行执行 4 个 32 位字的循环左移, 将线性变换的 4 次移位异或操作合并为向量运算, 吞吐量提升约 3 倍。

3.5 SM4-GCM 模式实现

GCM 是一种认证加密模式, 结合加密与消息认证, 实现思路:

加密过程: 基于 CTR 模式, sm4_gcm_encrypt 函数通过加密计数器生成密钥流, 与明文异或得到密文, 计数器通过 mm_add_epi64 递增。

认证过程: gcm_gf_mult 函数使用 mm_clmulepi64_si128 (PCLMULQDQ 指令) 实现 GF (2¹²⁸) 上的多项式乘法, 用于计算认证标签。附加数据 (AAD) 和密文块通过伽罗瓦乘法累积到 auth_tag 中。

密钥与 IV 处理: sm4_gcm_init 函数生成哈希密钥 H (加密全零块) 和初始计数器 J0 (IV 扩展为 128 位), 最终标签通过初始计数器加密结果与累积认证值异或生成。

四、关键技术细节

4.1 S 盒实现

代码中 sbox 数组直接存储 SM4 标准 S 盒值, 基于 GM/T0002-2012 标准定义, 反 S 盒 (sm4_inv_sbox) 用于解密。

4.2 向量指令优化

AES-NI 和 GFNI 优化均使用 128 位 __m128i 向量类型, 单次处理 1 个 128 位块, 支持并行字节操作。

线性变换中, 循环左移通过 mm_rot_epi32 (GFNI) 或 sm4_rotl (软件) 实现, 向量版本可同时处理 4 个 32 位字。

4.3 GCM 伽罗瓦乘法


gcm_gf_mult 函数通过多项式乘法和缩减实现 GF (2¹²⁸) 乘法:

用 mm_clmulepi64_si128 计算 64 位分段乘积, 合并为 256 位中间结果。

用缩减多项式 $x^{128}+x^7+x^2+x+1$ 对结果进行模运算, 通过循环移位和异或将 256 位结果缩减为 128 位。

五、测试与验证

运行代码，得到如下图结果：



```
Microsoft Visual Studio 调试器
基本实现测试：成功
T-Table实现测试：成功
AES-NI实现测试：成功
GFNI实现测试：成功
GCM模式测试：成功

D:\OneDrive\桌面\网安\Project1\x64\Debug\Project1.exe (进程 17120)已退出，代码为 0 (0x0)。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...|
```

5.1 功能验证

使用 GB/T 32907-2016 标准测试向量验证正确性：

密钥：01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

明文：01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10

预期密文：68 1e df 34 d2 06 96 5e 86 b3 e9 4f 53 6e 42 46

代码中各实现（基本、T-Table、AES-NI）均通过该向量验证，输出与预期一致。

5.2 模式验证

GCM 模式测试通过对比加密前后数据差异（非全零检查）和标签生成完整性，验证认证加密流程的正确性。

六、结论

本报告详细阐述了 SM4 算法的数学原理与软件实现，通过多级优化显著提升了运算效率：

基本实现满足功能正确性，为优化提供基准；

T-Table 优化通过空间换时间，减少约 30% 的执行时间；

AES-NI 和 GFNI 指令集优化利用硬件加速，吞吐量提升 2-3 倍；

SM4-GCM 模式实现了认证加密一体化，满足高安全性场景需求。

本代码在 VS2019 环境下编译运行，需支持 AES-NI/GFNI 的硬件。