# SM4 算法在 T-table、AESNI 及新型指令集下的优化实现

## 1. T-table 优化技术

### 1.1 数学原理与设计思路

T-table 优化是通过预计算合成变换 T 的部分结果来减少实时运算量的优化方法，其核心思想是利用空间换时间。在 SM4 算法中，合成变换 T (・)=L (τ(・)) 包含 S 盒置换（τ 变换）和线性变换（L 变换）两个步骤。由于 S 盒的输入是 8 位字节，所有可能的输入组合仅有 256 种，这为预计算提供了可行性。

设 T-table 为一个包含 256 个 32 位字的查找表，定义为：

T\_table[a] = L(τ(a << 24))

其中 a 为 8 位输入（0≤a≤255），通过将 8 位输入左移 24 位扩展为 32 位字，再经 τ 变换（仅对高 8 位应用 S 盒）和 L 变换后，将结果存储在 T\_table 中。

对于 32 位输入字 X=(x₀,x₁,x₂,x₃)（其中 xᵢ为 8 位字节），其 T 变换结果可通过 T-table 查表组合得到：

T(X) = T\_table[x₀] ^ (T\_table[x₁] << 8) ^ (T\_table[x₂] << 16) ^ (T\_table[x₃] << 24)

这种设计将原本需要 4 次 S 盒查找和 4 次线性变换的运算，简化为 4 次查表和 3 次移位异或操作，理论上可将 T 变换的执行效率提升 30%-50%。

### 1.2 实现优势与局限性

T-table 优化的优势在于：

* 完全兼容所有处理器架构，无需特殊指令集支持
* 预计算仅需 1KB 存储空间（256×32 位），内存开销极小
* 查表操作可被 CPU 缓存高效支持，适合嵌入式环境

局限性体现在：

* 无法并行处理多组数据，在大数据量加密场景下性能受限
* 相比硬件指令集优化，性能提升幅度有限

## 2. AESNI 指令集优化

### 2.1 指令集映射关系

AESNI（Advanced Encryption Standard New Instructions）是 Intel 针对 AES 算法设计的硬件加速指令集，但通过数学变换可间接用于 SM4 优化。其核心指令与 SM4 运算的映射关系如下：

1. AESIMC（AES Inverse Mix Column）：原本用于 AES 的逆列混合操作，可通过构造中间状态模拟 SM4 的 S 盒变换。设 S 盒查找表为一个 128 位向量，通过AESIMC的字节置换特性实现 τ 变换：

τ(X) = AESIMC(X ^ Sbox\_vector)

1. AESENC（AES Encrypt）：利用其轮密钥加和列混合的并行性，实现 SM4 轮函数中的线性变换部分。通过 3 条 AESNI 指令可完成原本需要 10 余条普通指令的 L 变换运算。
2. MMX扩展指令：\_mm\_xor\_si128、\_mm\_slli\_epi32等向量指令可并行处理 4 个 32 位字的异或和移位操作，将轮迭代的并行度提升 4 倍。

### 2.2 性能提升模型

在 AESNI 优化下，SM4 算法的单次轮迭代可在 3 个时钟周期内完成，相比纯软件实现的 15-20 个时钟周期有显著提升。其性能模型可表示为：

T = (N × 3 + 10) / F

其中 N 为 32 轮迭代，F 为 CPU 主频（GHz），10 为初始化和反序变换的固定开销。对于 3GHz 处理器，理论吞吐量可达：

128bit / ( (32×3 + 10)/3e9 ) ≈ 3.2Gbps

## 3. 新型指令集优化

### 3.1 GFNI（Galois Field New Instructions）

GFNI 指令集专为伽罗瓦域运算设计，其GF2P8AFFINEQB指令可直接实现 8 位字节的仿射变换，完美匹配 SM4 的 S 盒操作。相比 AESNI 的间接模拟，GFNI 的 S 盒变换实现更为直接：

1. GF2P8AFFINEQB：通过设置变换矩阵参数，可在 1 条指令内完成 SM4 的 S 盒置换，包括仿射变换和逆仿射变换。
2. GF2P8MULB：用于 GCM 模式中的伽罗瓦乘法，将认证标签生成的运算效率提升 40% 以上。

在 GFNI 优化下，SM4 的 T 变换可简化为：

T(X) = GF2P8AFFINEQB(X, SM4\_SBOX\_MATRIX) → 线性变换

整个过程仅需 5 条指令，比 AESNI 减少 60% 的指令数量。

### 3.2 VPROLD（Vector Permute with Offset）

VPROLD 指令集提供了灵活的向量置换能力，特别适合 SM4 的轮函数数据重排：

1. VPROLD：支持带偏移的向量循环移位，可直接实现 L 变换中的 <<<2、<<<10 等操作，比传统的PSLLD+PSRLD组合减少 3-4 条指令。
2. VPPERM：用于轮迭代后的状态重排，将反序变换从 8 条指令简化为 1 条。

结合 GFNI 和 VPROLD 的协同优化，SM4 算法的单轮迭代可压缩至 2 个时钟周期，在支持 AVX512 的处理器上，可同时处理 4 个 128 位分组，理论吞吐量突破 20Gbps。

### 3.3 AMX（Advanced Matrix Extensions）

AMX 是面向矩阵运算的新型指令集，通过 8KB 的 2D 寄存器数组（tile）可实现大规模并行的 SM4 运算。在批量数据加密场景下：

1. 利用TILELOAD将 16 个明文分组加载到 2D 寄存器
2. 通过TMUL指令并行执行 16 组 S 盒变换
3. 配合TSUM完成线性变换的异或累加

测试数据显示，AMX 优化的 SM4 实现比 GFNI 单向量优化在大数据量下性能再提升 3-5 倍，特别适合云计算和数据库加密场景。