SM4-GCM

一、引言

在当今数字化时代，数据安全面临着愈发严峻的挑战，认证加密算法作为保障数据安全的关键技术，能够同时提供数据的机密性和完整性保护。SM4 算法是中国自主设计的分组密码算法，具有安全性高、实现效率良好等特点，已被广泛应用于各类信息安全领域。GCM（Galois/Counter Mode）作为一种高效的认证加密模式，凭借其并行处理能力和优异的性能，在众多应用场景中得到了广泛采用。

二、SM4 算法原理

1、算法概述

SM4 是一种分组密码算法，其核心特性如下：
分组长度为 128 位，即每次处理的数据块大小为 16 字节。
密钥长度为 128 位，确保了足够的安全性。
加密和解密采用相同的算法结构，仅轮密钥的使用顺序不同。
算法通过 32 轮迭代运算实现数据的加密变换。

2、核心操作

S 盒变换

S 盒变换是 SM4 算法中的非线性置换操作，是保证算法安全性的核心环节之一。它采用 8 位输入、8 位输出的置换表，对输入的字节进行非线性变换，增加了密码算法的抗攻击能力。在本实现中，定义了标准的 Sbox 数组来完成这一变换。

线性变换 L

线性变换 L 的计算公式为：L(B)=B^(B<<13)^（B<<23)，其中“<<”表示循环左移操作。该变换通过对数据进行线性混合，进一步扩散明文和密钥的影响。

T 变换

T 变换是 S 盒变换与线性变换 L 的组合，即 T(·)=L(S(·))。它先对输入数据进行 S 盒的非线性变换，再经过线性变换 L，从而实现数据的复杂变换。

轮函数 F

轮函数 F 的计算公式为：F(X0,X1,X2,X3,rk)=X0^T (X1^X2^X3^rk)，

其中 rk 为轮密钥。在每一轮迭代中，轮函数利用轮密钥对数据进行变换，经过 32 轮迭代后得到加密结果。

3、密钥扩展

SM4 算法的密钥扩展过程是生成轮密钥的关键步骤，具体如下：
输入为 128 位的初始密钥。
输出为 32 个 32 位的轮密钥，用于 32 轮迭代运算。
密钥扩展过程利用 FK 常数和 CK 常数进行运算。
轮密钥生成公式为：rk[i]=K[i+4]=K[i]^T'(K[i+1]^K[i+2]^K[i+3]^CK[i])，其中 T'是密钥扩展过程中使用的变换函数。

三、GCM 模式原理

1、概述

GCM 是一种高效的认证加密模式，具有以下显著特点：
能够同时为数据提供机密性和完整性保护，满足信息安全的核心需求。
支持对附加认证数据（AAD）进行认证，这些数据不会被加密，但会被纳入完整性验证的范围。
加密过程具有良好的并行化特性，能够充分利用现代计算机的多核处理能力，适用于高性能需求的场景。
输出固定长度的认证标签，用于验证数据的完整性和真实性。

2、核心组成

CTR 加密模式

CTR 加密模式主要用于提供数据的机密性。它通过使用计数器生成密钥流，将计数器值加密后与明文进行异或操作得到密文。在解密过程中，使用相同的计数器生成密钥流，与密文异或得到明文，因此解密过程与加密过程相同。

GHASH 函数

GHASH 函数基于伽罗瓦域 GF(2^128)上的乘法运算，主要用于提供数据的完整性保护。其输入包括附加认证数据、密文以及这些数据的长度信息，输出结果用于生成认证标签。

3、工作流程

初始化阶段

生成哈希子密钥 H，H 是通过加密全 0 的 128 位数据块得到的，即

H=E (0^128)。

计算初始计数器值 J0，对于 12 字节的 nonce，J0 按照特定规则计算；对于其他长度的 nonce，J0 通过 GHASH 函数计算得到。

加密过程

使用 CTR 模式对明文进行加密，得到密文。

计算 GHASH 值，即 GHASH(H,A,C)，其中 A 为附加认证数据，C 为密文。

生成认证标签，标签的计算方式为：Tag=E(J0)^GHASH(H,A,C)。

解密过程

使用 CTR 模式对密文进行解密，得到明文。

重新计算 GHASH 值。

验证计算得到的标签与接收的标签是否匹配，若匹配则数据完整性得到确认，否则数据可能被篡改。

四、SM4-GCM 实现分析

1、代码结构

SM4 GCM 的实现包含两个主要类，分别负责不同的功能：

SM4 类：该类实现了 SM4 分组密码算法的核心功能，包括密钥扩展、块加密等操作。

SM4GCM 类：该类基于 SM4 算法实现了 GCM 模式，提供了认证加密的完整功能，包括加密并生成标签、解密并验证标签等接口。

2、核心功能实现

SM4 类实现

初始化方法

```
def __init__(self, key):
    self.key = key
    self.round_keys = self._generate_round_keys()
```

该方法接收 128 位的密钥作为输入，通过调用_generate_round_keys 方法生成 32 个轮密钥，并将其存储在 round_keys 属性中，为后续的加密操作做好准备。

密钥扩展

_generate_round_keys 方法实现了 SM4 算法的密钥扩展过程。它根据初始密钥，结合 FK 常数和 CK 常数，按照轮密钥生成公式生成 32 个轮密钥，确保每一轮迭代都有对应的轮密钥可用。

块加密

_block_encrypt 方法实现了 SM4 算法的 32 轮迭代加密过程。它将输入的 128 位数据块拆分为 4 个 32 位的字，经过 32 轮迭代运算后，对结果进行输出变换，得到加密后的 128 位数据块。encrypt_block 方法则负责处理 16 字节数据块的加密，将字节数据转换为整数进行加密运算，再将加密结果转换回字节数据。

SM4GCM 类实现

初始化方法

```
def __init__(self, key, nonce=None):
    self.sm4 = SM4(self._key_to_int(key))
    self.nonce = nonce if nonce is not None else b'\x00' * 12
    self.H = self.sm4.encrypt_block(b'\x00' * 16)
    self.J0 = self._compute_j0()
    self._precompute_ghash_table()
```

该方法初始化 SM4 实例，将输入的密钥转换为整数形式并传递给 SM4 类。同时，设置 nonce 的值（默认使用 12 字节的全 0 值），计算哈希子密钥 H 和初始计数器 J0，并调用_precompute_ghash_table 方法预计算 GHASH 乘法表。

GHASH 优化实现

_precompute_ghash_table 方法预计算了一个 16×256 的伽罗瓦乘法结果表。通过提前计算各种可能的乘法结果，避免在认证过程中重复计算，显著提高了 GHASH 运算的效率。

ghash 方法实现了认证标签的核心计算过程。它将附加认证数据和密文组合后进行填充，添加长度信息，然后对每个 16 字节的数据块进行处理，利用预计算的乘法表完成伽罗瓦乘法运算，最终得到 GHASH 值。

CTR 加密

ctr_encrypt 方法实现了 CTR 模式的加密功能。它通过递增计数器生成一系列计数器值，将每个计数器值加密后得到密钥流，再将密钥流与明文（或密文）进行异或操作，得到密文（或明文）。该方法同时用于加密和解密过程。

认证加密接口

encrypt_and_tag：该方法接收明文、附加认证数据和标签长度作为输入，使用 CTR 模式对明文进行加密得到密文，计算 GHASH 值并生成认证标签，最后返回密文和指定长度的标签。

decrypt_and_verify：该方法接收密文、标签、附加认证数据和标签长度作为输入，使用 CTR 模式对密文进行解密得到明文，重新计算 GHASH 值和认证标签，通过常数时间比较验证标签是否匹配，若匹配则返回明文，否则抛出异常。

3、优化策略

预计算优化

通过预计算 GHASH 乘法表，将伽罗瓦乘法运算中可能用到的结果提前计算并存储，在认证过程中直接查表使用，避免了重复计算，大幅提高了 GHASH 函数的执行效率。

高效数据处理

采用整数运算和字节操作相结合的方式，减少了数据在不同类型之间转换的开销，提高了算法的整体执行效率。

安全增强

实现了_constant_time_compare 方法，采用常数时间比较的方式验证标签，避免了因比较时间差异而可能导致的侧信道攻击，增强了算法的安全性。

五、结果

```
========== RESTART: C:/Users/DELL/Desktop/project/sm4_gcm.py ==========
密钥: c948687302921265a72ef5a8c6fdd183
Nonce: 379e215befa1d5ffb87e0b60
密文: 15af1af6256e9c
标签: ccda56d49de73d817f7af79f0f61f461
解密结果: SM4-GCM
验证成功：解密消息与原始消息一致

= RESTART: C:/Users/DELL/Desktop/project/sm4_gcm.py
密钥: 426bc162ed16ef20eb8c410daa675edf
Nonce: 4fd13aac42901f4f61953ed0
明文: 534d342d47434d
ADD: e585b3e88194e695b0e68dae
密文: d6a9dfadbc0de3
标签: 9cd4f459037176f57217d67e51e8495a
解密结果: SM4-GCM
验证成功：解密消息与原始消息一致
```

六、总结

SM4-GCM 算法将 SM4 分组密码的高安全性与 GCM 模式的高效性完美结合，能够同时为数据提供机密性和完整性保护，适用于网络通信、存储加密、身份认证等多种需要保障数据安全的场景。