SM3 长度扩展攻击

一、SM3 与长度扩展攻击原理

1、 SM3 哈希算法基础

SM3 是中国国家密码管理局发布的密码哈希算法，用于数据完整性校验和身份认证。其核心流程包括：

- **消息填充：**将消息长度补齐为 512 比特的整数倍，填充规则为：消息 + 0x80 + 0x00*k + 原始长度（64 比特）。
- **消息扩展：**将 512 比特的消息分组扩展为 132 个字。
- **压缩函数：**以初始向量（IV）为起点，通过 64 轮迭代更新 8 个 32 位寄存器，最终输出哈希值。

2、Merkle-Damgård 结构与长度扩展攻击

SM3 采用 Merkle-Damgård 结构，该结构的特点是：

- 哈希值本质是压缩函数处理完所有消息分组后的中间状态
- 处理消息时按固定长度分组迭代，前一组的输出作为后一组的输入

攻击原理：

1. 设原始消息为 M，其哈希值 H(M)是压缩函数处理完 M 的所有分组后的中间状态
2. 攻击者已知 H(M)和 len(M)，可将 H(M)作为新的初始向量
3. 计算 M 的填充数据 padding(M)，构造扩展消息 M'=M+padding(M)+X（X 为攻击者自定义数据）
4. 用 H(M)作为初始向量计算 X 的哈希，结果等价于 H(M')，实现无需知晓 M 即可构造 M'的哈希

二、程序

1、程序结构

SM3 类：实现 SM3 哈希算法核心功能

常量定义（IV、T）

辅助函数（旋转、布尔函数、置换函数）

填充函数（padding）

消息扩展函数（message_extension）

压缩函数（compression_function）
哈希计算函数（hash）
LengthExtensionAttacker 类：实现长度扩展攻击
攻击函数（attack）
验证函数（verify_attack）：验证攻击效果

2、核心模块

（1）SM3 类：SM3 哈希实现

常量定义

```
IV = [0x7380166F, 0x4914B2B9, ..., 0xB0FB0E4E]  # 初始向量（8 个 32 位字）
T = [0x79CC4519] * 16 + [0x7A879D8A] * 48  # 常量数组（64 个 32 位字）
```

辅助函数

rotate_left(x, n)：32 位整数循环左移 n 位
FF_j(X,Y,Z,j)/GG_j(X,Y,Z,j)：布尔函数（分阶段使用不同逻辑）
P0(X)/P1(X)：置换函数（用于压缩和消息扩展）

padding 函数

实现 SM3 填充规则，支持自定义总长度（攻击时需基于原始消息 + 扩展数据的总长度填充）：

```
def padding(message, total_bit_length=None):
    1. 添加 0x80 分隔符
    2. 填充 0x00 至长度模 512=448
    3. 添加总长度（64 比特）
```

message_extension 函数

将 64 字节消息分组扩展为 W（68 个字）和 W'（64 个字），用于压缩函数：

```
def message_extension(B):
    1. 从 64 字节中提取 16 个 32 位字
    2. 扩展生成其余 52 个 W 字
    3. 生成 W'数组（W[i]^W[i+4]）
```

compression_function 函数

核心压缩逻辑，用消息分组更新中间状态：

```
def compression_function(V, B):
    V 为当前中间状态（8 个寄存器）
    64 轮迭代更新寄存器，最终输出新的中间状态
```

hash 函数

主函数，协调填充、分组处理和压缩：

```
def hash(message, initial_vector=None, total_bit_length=None):
    1. 填充消息
    2. 按 64 字节分组迭代处理
    3. 输出最终哈希值（十六进制字符串）
```

（2）LengthExtensionAttacker 类：攻击实现

attack 函数

攻击函数步骤如下：

```
def attack(original_hash, original_length, append_data):
    1. 转换原始哈希为初始向量
    2. 计算原始消息填充后长度
    3. 计算扩展后的总长度
    4. 计算附加数据的哈希（模拟扩展消息哈希）
```

（3）verify_attack 函数：攻击验证

攻击流程：

1. 定义原始消息、计算其哈希和长度
2. 定义攻击者附加数据
3. 执行攻击生成扩展消息哈希
4. 计算实际扩展消息（原始消息+填充+附加数据）的哈希
5. 对比两者，验证攻击是否成功

三．结果

```
====== RESTART: C:/Users/DELL/Desktop/project/sm3_length_extension.py =====
原始消息: secret_key
原始哈希值: 54d1dde046251680ec80af50331f61cddcd95a5aa25f6dec5e84a3bf9e1f6821
原始消息长度: 10字节
附加数据: &user=admin&role=root&
攻击生成的哈希: 6f18f83e739d8b261b6cc841d5539e80f509ed19020c161f126e89248f9a2d86
实际扩展消息哈希: 6f18f83e739d8b261b6cc841d5539e80f509ed19020c161f126e89248f9a2d86
长度扩展攻击成功!
```

```
====== RESTART: C:/Users/DELL/Desktop/project/sm3_length_extension.py =====
原始消息: secret_key
原始哈希值: 54d1dde046251680ec80af50331f61cddcd95a5aa25f6dec5e84a3bf9e1f6821
原始消息长度: 10字节
附加数据: &user=admin&password=123&
攻击生成的哈希: 476d6bc105d5f625b54d351a5e7a9c0890f07554c867011c7892bfd04a53312d
实际扩展消息哈希: 476d6bc105d5f625b54d351a5e7a9c0890f07554c867011c7892bfd04a53312d
长度扩展攻击成功!
```

验证 SM3 长度扩展攻击成功

四、结论

SM3 算法因采用 Merkle-Damgård 结构，存在长度扩展攻击风险。本实验验证了攻击的可行性：攻击者仅需原始消息的哈希值和长度，即可构造包含恶意扩展数据的有效哈希。避免使用 hash(secret+message) 这类依赖 Merkle-Damgård 结构的认证方式，推荐使用 HMAC-SM3 等带密钥的哈希方案。