长度扩展攻击（Length-Extension Attack）

1. 攻击原理

长度扩展攻击是一种针对特定哈希函数的密码学攻击，攻击者可以利用哈希函数的特性，在不知道原始消息的情况下，将新数据附加到原始消息后面，并计算出新的哈希值。

这种攻击适用于所有基于Merkle-Damgård结构的哈希函数，包括MD5、SHA-1、SHA-256以及本项目中的SM3。

1.1 Merkle-Damgård结构的弱点

Merkle-Damgård结构的工作原理是：

1. 将消息分成固定大小的块
2. 初始化一个初始向量（IV）
3. 使用压缩函数迭代处理每个消息块，将前一个块的输出作为下一个块的输入
4. 最终输出最后一个压缩函数的结果作为哈希值

这种结构的关键弱点是：**哈希值本质上是压缩函数的中间状态**。如果攻击者获取了这个中间状态（即哈希值），就可以将其作为新的初始向量，继续处理新的消息块，从而实现长度扩展。

1.2 攻击步骤

长度扩展攻击的基本步骤：

1. 攻击者获取原始消息的哈希值H = Hash(m)
2. 攻击者猜测或确定原始消息m的长度len(m)
3. 攻击者计算原始消息的填充p = Padding(m)
4. 攻击者构造新消息m' = m || p || x，其中x是攻击者想要追加的数据
5. 攻击者使用哈希值H作为初始向量，对x进行哈希计算，得到Hash(m')

整个过程中，攻击者不需要知道原始消息m的内容。

2. SM3的长度扩展攻击实现

2.1 核心思想

SM3作为基于Merkle-Damgård结构的哈希函数，同样存在长度扩展攻击的风险。攻击的核心是：

- 将SM3的哈希值解析为压缩函数的中间状态（8个32位字）
- 使用这个中间状态作为新的初始向量
- 对追加的数据进行哈希计算，得到扩展后的哈希值

2.2 实现关键

1. **哈希值解析**：将64字符的SM3哈希值解析为8个32位整数，作为新的初始向量
   def parse_hash(hash_hex):

```
V = []
for i in range(8):
V.append(int(hash_hex[i*8:(i+1)*8], 16))
return V
```

2. **填充计算**：根据原始消息长度计算填充，构造扩展消息
3. **增量哈希计算**：使用解析得到的中间状态作为初始向量，对追加数据进行哈希计算

3. 攻击演示

假设场景：

- 系统使用H = SM3(key || message)进行认证
- 攻击者知道message和H，但不知道key
- 攻击者想要构造一个新的消息message' = message || padding || new_data，并计算对应的哈希值

攻击过程：

1. 计算原始数据（key + message）的长度L = len(key) + len(message)
2. 解析哈希值H得到中间状态V
3. 计算新数据new_data的填充
4. 使用V作为初始向量，对new_data进行哈希计算，得到H'
5. H' = SM3(key || message || padding || new_data)，即扩展后的哈希值

通过这种方式，攻击者可以在不知道key的情况下，构造有效的哈希值。

4. 防御措施

为了防止长度扩展攻击，可以采取以下措施：

4.1 使用HMAC结构

HMAC是一种基于哈希函数的消息认证码，其结构为：HMAC(key, message) = H((key ⊕ opad) || H((key ⊕ ipad) || message))
HMAC通过嵌套哈希计算，有效防止了长度扩展攻击，因为攻击者无法获取内层哈希的中间状态。

4.2 限制消息长度

如果应用场景允许，可以限制消息的长度，使得攻击者无法构造有效的扩展消息。

4.3 使用抗长度扩展的哈希函数

某些哈希函数如SHA-3（基于海绵结构）本质上抵抗长度扩展攻击，因为它们的输出不是压缩函数的中间状态。

4.4 哈希消息长度

在哈希计算前，将消息长度作为前缀加入消息中：Hash'(message) = Hash(len(message) || message)
这种方法使得攻击者难以构造有效的扩展消息，因为长度字段会随着消息扩展而改变。

5. 安全影响

长度扩展攻击在以下场景中可能造成安全问题：

1. **认证系统**：如果系统仅使用Hash(key || message)进行认证，攻击者可以构造包含恶意内容的新消息
2. **数字签名**：如果签名基于Hash(message)，攻击者可能构造出有效的签名扩展消息
3. **数据完整性校验**：攻击者可能篡改数据并生成有效的哈希值

了解长度扩展攻击的原理和防御方法，对于正确使用哈希函数构建安全系统至关重要。