SM3实现长度扩展攻击

使用SM3函数用随机生成一个消息（secret），对其进行填充并进行hash，计算出哈希值（hash_1）

使用hash_1推算出这一次加密结束后8个向量的值，再以它们作为IV值，去加密append，得到另一个hash值(hash_2)

随意选取另一消息（secret），计算secret + padding + append的hash值（hash_3）。

如果hash_2与hash_3相等，则攻击成功；否则攻击失败。

攻击原理

在SM3函数计算时，首先对消息进行填充分组，每组64字节，每一次加密一组，并更新8个初始向量IV(初始值已经确定)，下一次用新向量去加密下一组，以此类推。我们可以利用这一特性去实现攻击。当我们得到第一次加密后的向量值时，再使用另一组消息用于下一次加密，就可以在不知道secret的情况下得到合法的hash值。

实现细节

随机生成消息（secret），计算出其哈希值hash_1

```
secret = str(random.random())
hash_1 = sm3.sm3_hash(func.bytes_to_list(bytes(secret, encoding='utf-8')))
secret_len = len(secret)
append_m = "1234567890"
```

利用hash_1加密计算出另一个哈希值hash_2

```
hash_2 = generate_hash_2(hash_1, secret_len, append_m)
```

使用另一个消息计算哈希值生成hash_3

```
new_msg = func.bytes_to_list(bytes(secret, encoding='utf-8'))
new_msg.extend(pad)
new_msg.extend(func.bytes_to_list(bytes(append_m, encoding='utf-8')))
new_msg_str = secret + pad_str + append_m
hash_3 = sm3.sm3_hash(new_msg)
```

实验结果截图：

可以看出hash_2=hash_3，攻击成功。

```
==========
hash_1: 0a4bd314ed311a5afe89074aac35bff42a280b4561af7b10b0136c1270ea508e
hash_2: f951ed031cad4adae613debd815a7100537b8889e6b25cde4f682c492f8a7ec0
hash_3: f951ed031cad4adae613debd815a7100537b8889e6b25cde4f682c492f8a7ec0
hash_3 = hash_2,攻击成功!
>>>
```