一、 实验流程：

1. 先对“abcde”进行散列加密，加密结果为（代码中为 y 值）：

```
c:\Users\hxx\Desktop>python sm3_extension_attack.py
afe4ccac5ab7d52bcae36373676215368baf52d3905e1fecbe369cc120e97628
```

2. 对“abcde”进行拓展为“abcde”+补位 1+“qwert”，“abcde”+补位 1 为第一个 512 比特的分组。对扩展信息补位后，得“abcde”+补位 1+“qwert”+补位 2，“qwert”+补位 2 为第二个 512 比特的分组，以 1 中的散列加密结果，作为压缩函数的初始值，对第二个分组进行加密，得结果：（本次消息构造，实际上不需要知道“abcde”只需要知道它的字符串长度即可，所以实现了长度拓展攻击，即在不知道“secret”是什么，只知道“secret”的加密结果和 secret 的长度，就知道了“secret”+补位 1+“qwert”的加密结果（“qwert”可以任意置换））。（代码中为 hash_value2）

```
c:\Users\hxx\Desktop>python sm3_extension_attack.py
a51aea273d957ea25726d7e6e232ea8e14cd3306a53b3a1a3b895cf752ac0de8
```

3. 直接对“abcde”+补位 1+“qwert”进行加密，得到加密结果。（代码中为 hash_value1）

```
c:\Users\hxx\Desktop>python sm3_extension_attack.py
a51aea273d957ea25726d7e6e232ea8e14cd3306a53b3a1a3b895cf752ac0de8
```

4. 验证 hash_value1 是否与 hash_value2 相等，如果相等，则攻击成功。

```
c:\Users\hxx\Desktop>python sm3_extension_attack.py
attack is ok!
hash_value is: a51aea273d957ea25726d7e6e232ea8e14cd3306a53b3a1a3b895cf752ac0de8
```