代码说明:

主要代码为generate\_k函数，以下对该函数进行说明，其他步骤见代码注释

def generate\_k(x, m,n, h=hashlib.sha256):#RFC6979生成随机数k的函数  
 h1= h(m).digest() #生成Hash值，用的是sha256  
 h\_len= len(h1) #获取长度以确定v，\_k  
 v = b'\x01' \* math.ceil(h\_len / 8)  
 \_k= b'\x00' \* math.ceil(h\_len / 8)  
 #进行两轮消息摘要计算  
 \_k = \_hmac(\_k, v + b'\x00' + x + h1, h)  
 v = \_hmac(\_k, v, h)  
 \_k = \_hmac(\_k, v + b'\x01' +x + h1, h)  
 v = \_hmac(\_k, v, h)  
 #开始循环生成消息摘要以生成t  
 while True:  
 t = b'' #每次不满足长度要求时将t清空，重新生成t  
 while len(t.hex())< len(n):#确保生成的数为len(n) bytes  
 v = \_hmac(\_k, v, h)  
 t += v   
 k=int(t.hex(),16) #将生成的t变为数字，以便进行比较大小  
 if 0 < k < int(n,16): #确保生成的数小于n  
 return t.hex()  
 #进一步计算摘要  
 \_k = \_hmac(\_k, v + b'\x00', h)  
 v = \_hmac(\_k, v, h)

算法实现可参考

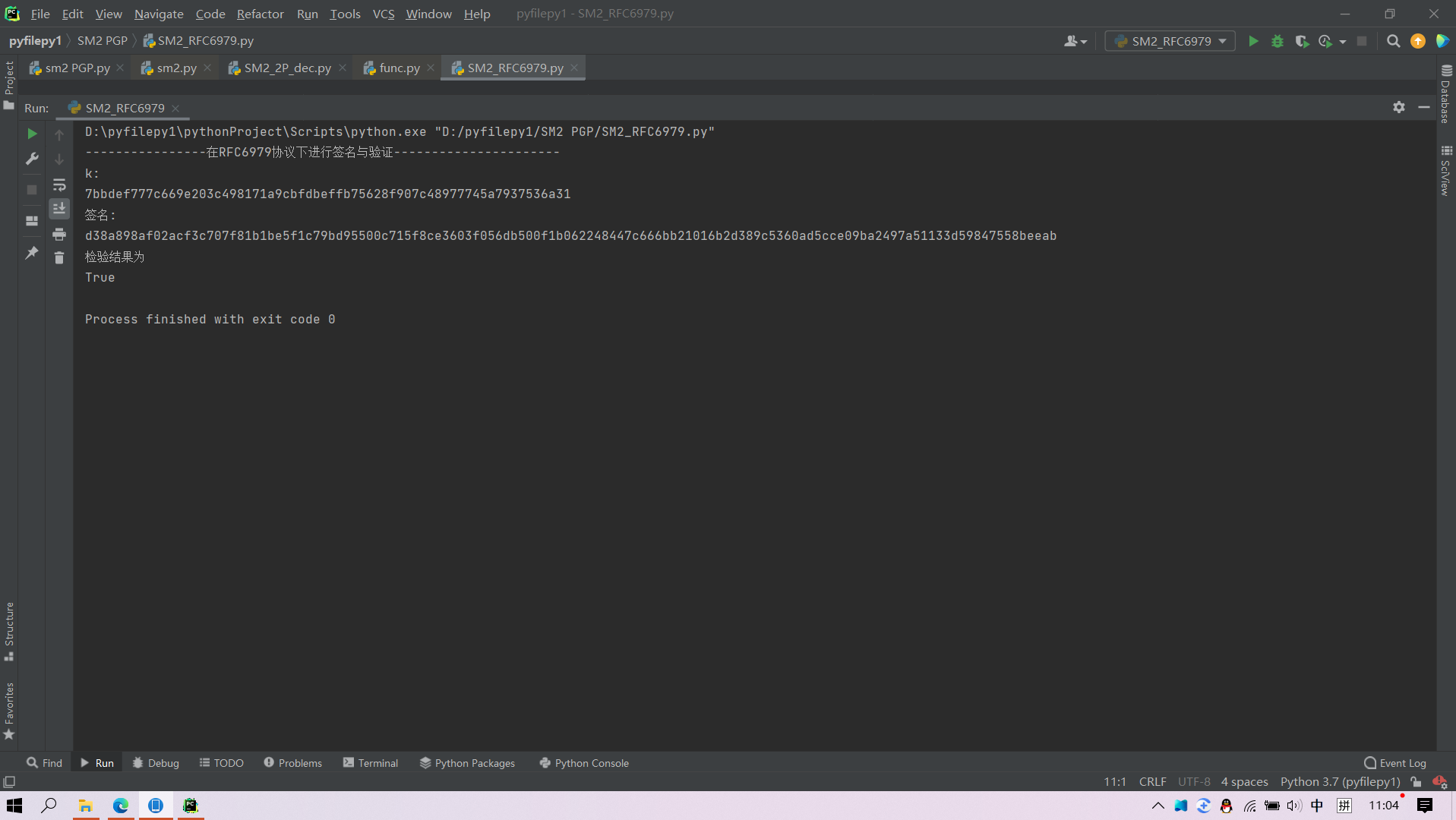
<https://blog.csdn.net/pony_maggie/article/details/77622149>

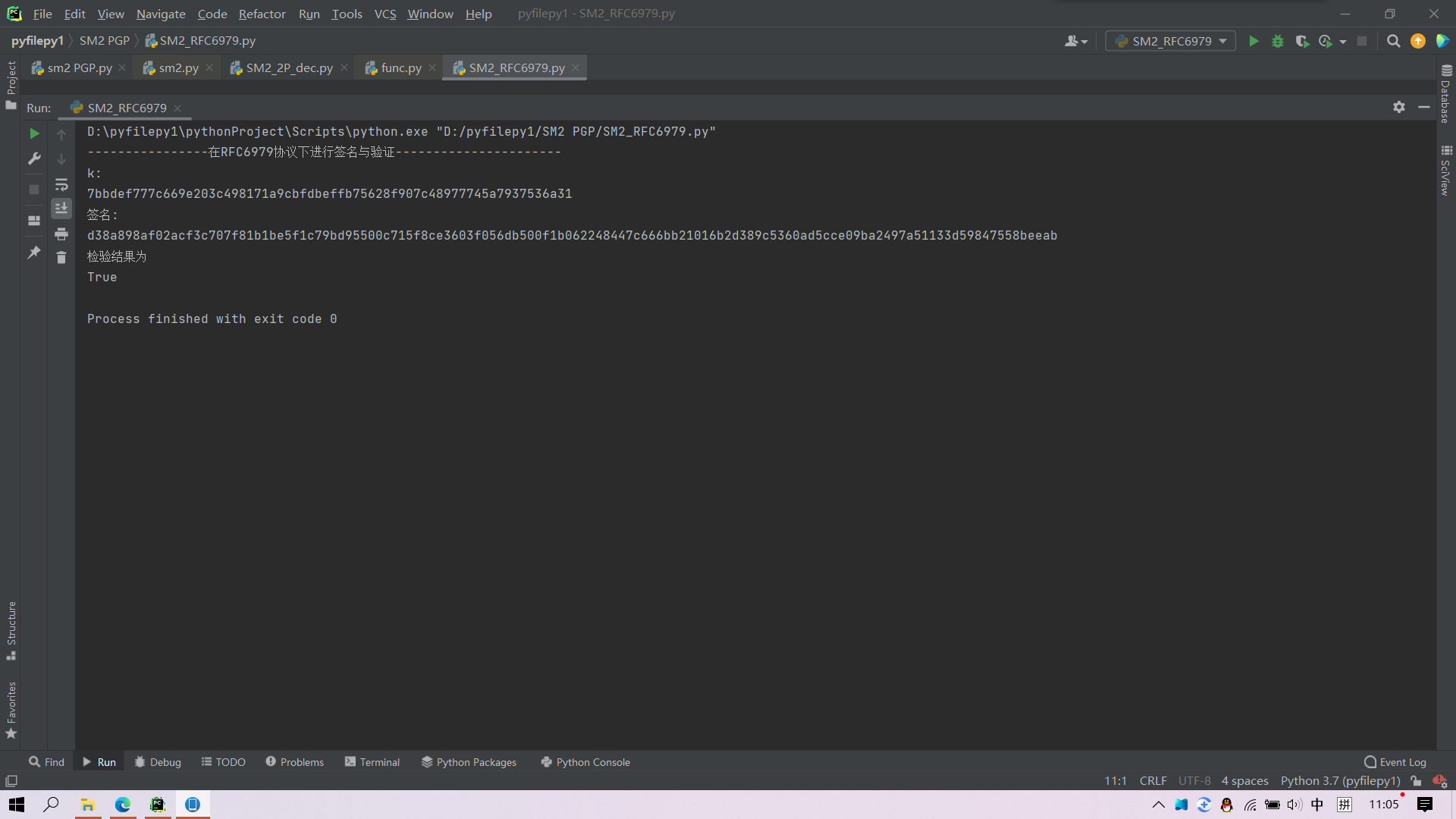
[RFC6979 Python demo implementation (github.com)](https://gist.github.com/gitzhou/87ae44da4d8681c5da7272152f4498a6)

运行指导:

在公私钥以及加密的消息确定的情况下，运行代码，每次加密产生的随机数不变，生成的密文不变，且可通过检验，则代码正确

结果截图:





我们可以发现11:04与11：05时间的加密结果一样，因此代码正确