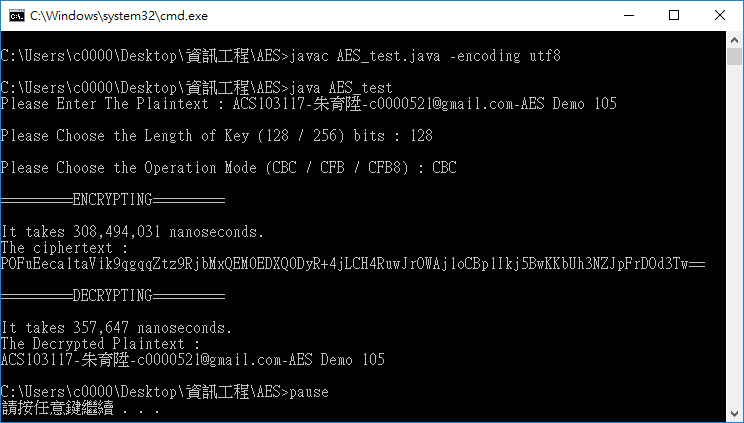
**105-1 Introduction to Cryptography   [Homework 3]**

ACS103117 資三甲 朱育陞

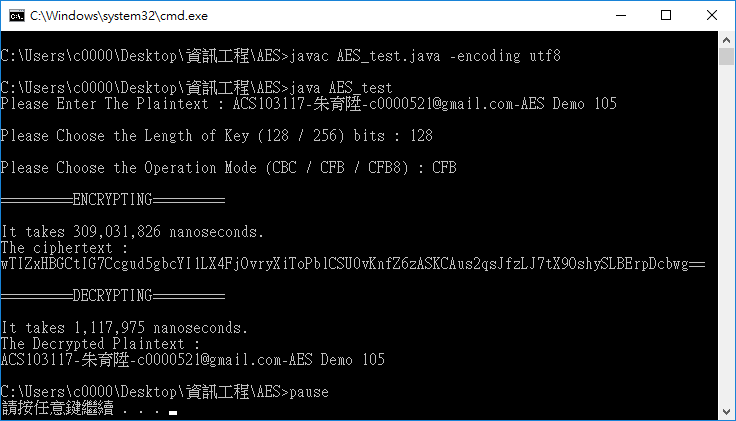
**一、實驗**

Plaintext : ACS103117-朱育陞-c0000521@gmail.com-AES Demo 105

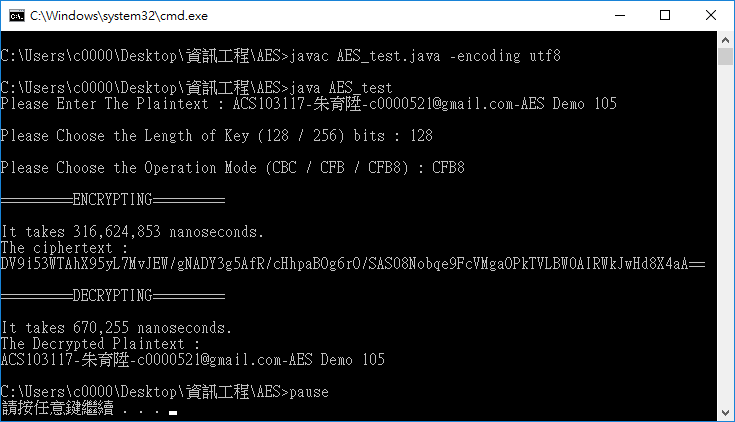
1. 128-bit key
   1. 128-bit key & CBC



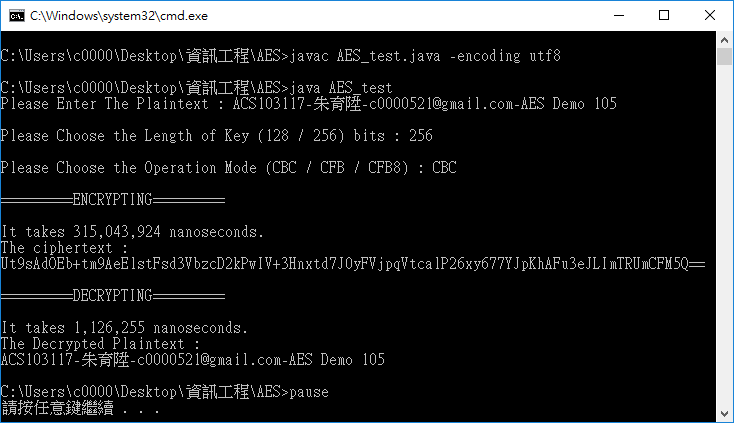
* 1. 128-bit key & CFB



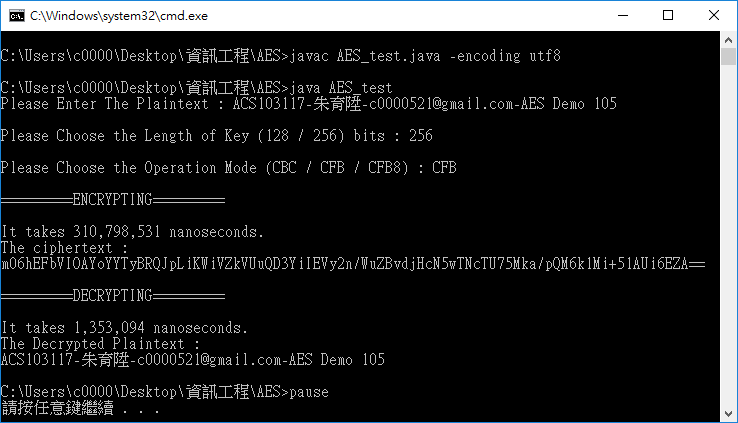
* 1. 128-bit key & CFB8



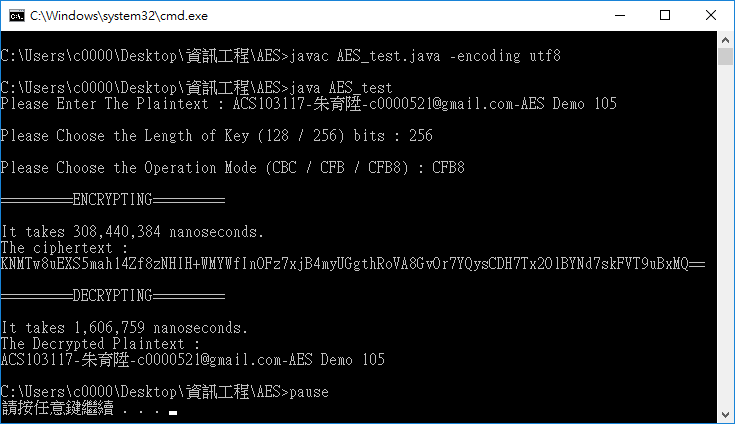
1. 256-bit key
   1. 256-bit key & CBC



* 1. 256-bit key & CFB



* 1. 256-bit key & CFB8



**二、研究心得**

　　現在JAVA直接提供了AES的功能，讓寫一個簡單的AES加解密程式變得相當容易，只需要告知Operation Mode和丟入符合所求的格式的KEY和IV就算完成一大半了。

　　接下來就是去符合作業要求，用時間產生IV，這邊利用多種時間單位相乘，讓IV值比較不會直接呈現當前時間的一部份。

　　再來就是KEY囉，是每個KEY字元用亂數去取 0~62 的區間，而0~9可直接代表數字字元所以就直接使用，再來10~35加上55就剛好對應到大寫英文字母的ASCII，同理，小寫字母就用36~62加上61就會對應到該ASCII了。

　　剩下的紀錄時間和輸出就很簡單了～，做到最後把讓使用者輸入bits的部分改為字串並檢視是否輸入錯誤，不然當初直接使用int來判斷時，使用者輸入非數字就會造成程式終止。

　　網路上很方便找到AES的範例，當初就查詢mode的表示方法以外，還有花一點時間處理byte的格式，Cipher裡面都需要是這種格式才有辦法執行。

觀察結果：

1. 不管128/256 bits、各種mode的情況下，加密時間都明顯遠大於解密  
   在AES.java中，AES\_encrypt中有呼叫到IV值的產生，而AES\_decrypt中只有呼叫現成的IV值，有可能是造成時間差距的引響之一。
2. 不管128/256 bits、各種mode的情況下，密文最後都是＂==＂
3. 在上述實驗之截圖可發現  
   在128-bits中加密時間：CFB8 > CFB > CBC  
   而256-bits中加密時間：CBC > CFB > CFB8  
   在128-bits中解密時間：CFB > CFB8 > CBC  
   而256-bits中解密時間：CFB8 > CFB > CBC  
     
   由此可見，加解密時間排序並無明顯快慢之差，很難判定何種較快或較慢。