**Assignment02**

**60847013S 資工碩一 蘇冠中**

1. **Modular Multiplicative Inverse**
   1. **135 61 = 47**

gcd(135,61)=gcd(61,13)=gcd(13,9)=gcd(9,4)=gcd(4,1)=

gcd(1,0)=1 有一解 從ax + by = 1代回去求x,y

gcd(1,0) ; x=1;y=0

gcd(4,1) ; x=0; y=1

gcd(9,4) ; x=1,y=-2

gcd(13,9) ; x=-2,y=3

gcd(61,13) ; x=3,y=-14

gcd(135,61) ; x=-14,y=31

最後x = -14為其中一個模反元素，而可能的模反元素為

-14 + (61\*n)，n為整數，我們要求最小的正整數解，則為-14+61=47

* 1. **7465 2464 = 2329**

gcd(7465,2464)=gcd(2464,73)=gcd(73,55)=gcd(55,18)=

gcd(18,1)=gcd(1,0)=1，有一解

從ax +by =1 代回去求x,y

gcd(1,0) ; x=1,y=0

gcd(18,1) ; x=0,y=1

gcd(55,18) ; x=1,y=-3

gcd(73,55) ; x=-3,y=4

gcd(2464,73) ; x=4,y=-135

gcd(7465,2464) ; x=-135,y=409

最後x=-135為其中一個模反元素，而可能的模反元素為

-135 + (2464\*n)，n為整數，我們要求最小的正整數解，則為-135+2464=2329

* 1. **42828 6407 = no**

gcd(42828,6407)=gcd(6407,4386)=gcd(4386,2021)=

gcd(2021,344)=gcd(344,301)=gcd(301,43)=gcd(43,0)=43

因此沒有解

1. **Fermat＇s Theorem**

**2.1 =**

**2.2 =**

1. **Chinese Remainder Theorem**

**3.1**

(1) 設P = p1\*p2\*p3\*…\*pk， ,，也就是說是除了pi以外的-1個整數的乘積

(2) 設ti = 為模pi的數論倒數:ti,

(3) n的通解為 n = n1t1P1+n2t2P2+n3t3P3+… =

(4) 從p1~pk都互質上可知，當i、j，j時，gcd()=1，所以gcd() =1，說明存在整數，因此

,且

(5) 所以n = 滿足:

n = + = (mod )，可說明n就是此方程組的一個解。

**3.2**

當p1~pk之間不互質的時候，從上面(4)可知，gcd()就不會=1，且 ，範例:

假設n1,n2兩個方程組，

n 3 (mod 5) 和 n 4 (mod 10)

M = 5\*10=50, M1=10,t1=0,當t=0時，n1方程組算出來的會是0，也就找不出n來。

1. **Complement**

從Fiestel network可知，所以可以得到。

因為排列不會影響到bit的值，加上E⊕= ⊕，所以。

所以S-BOX的輸出保持不變、加密結果也保持不變。

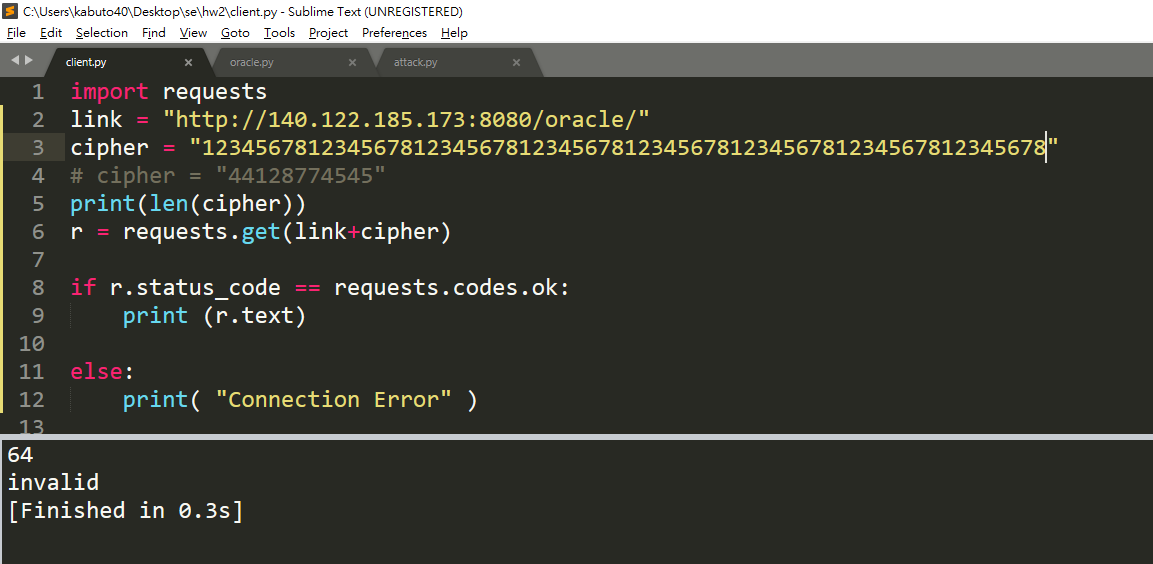
**4.2**

假設有個明文攻擊，若選擇的明文為M和，攻擊者得到，並不會對暴力破解造成太多的影響。

1. **Polynomial Ring**
2. **Padding Oracle Attack**

明文: My power flurries through the air into the ground. My soul is spiraling in frozen fractals all around. And one thought crystallizes like an icy blast. I'm never going back, the past is in the past.

步驟1: 先在網站上輸入正確的密文，網站會回傳valid正確的回應，若傳送的字串長度不對，則是回傳len invalid

而密文長度正確，但密文解回去不對的則是回傳invalid

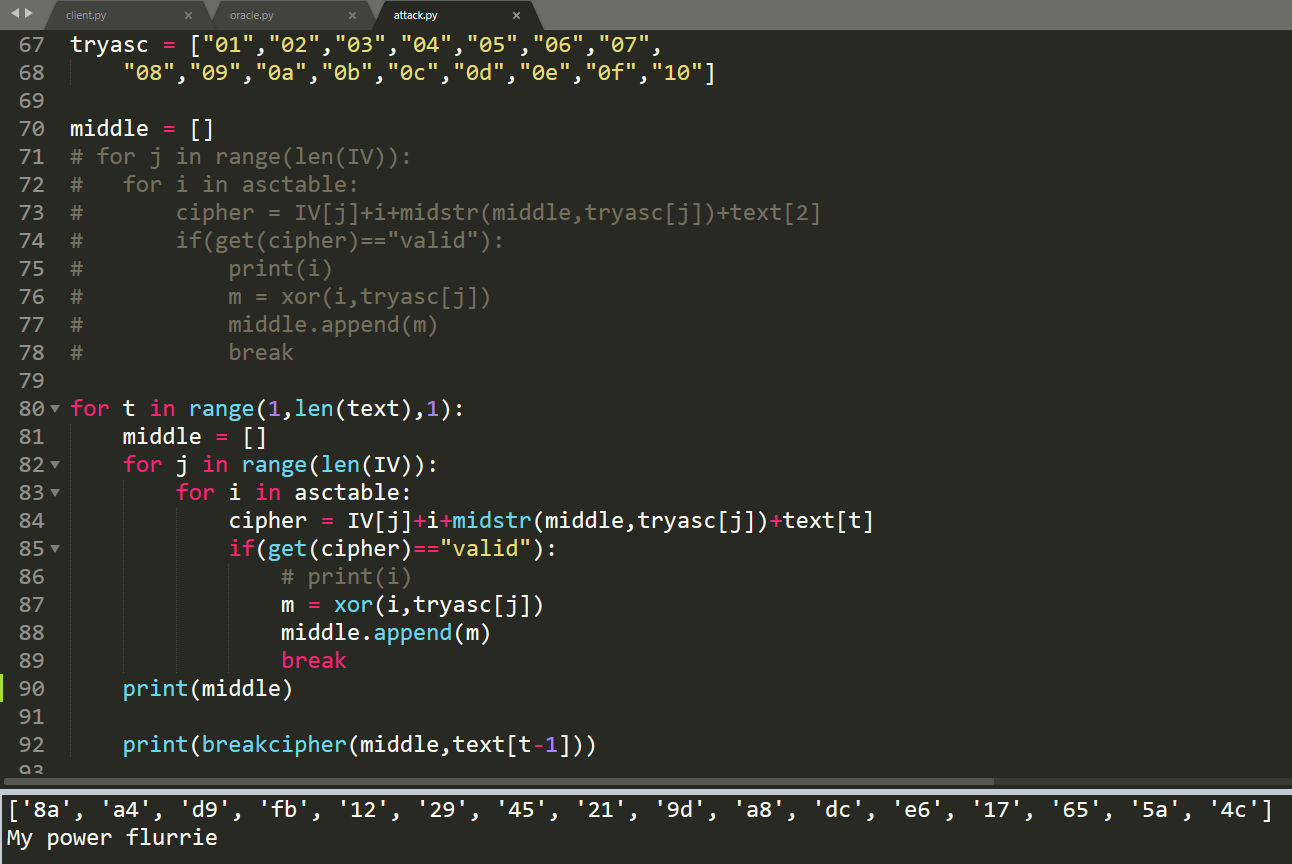
經過測試後，發現長度64、96、128、160…都是長度正確的，因此可推測密文的切塊是以32為一段。

步驟2: 所以接下來先將要解密的密文以32為一組拆解，依照padding oracle attack的作法，以及aes cbc加密的概念，第一組的密文為IV的初始向量，那我只先取第一組IV以及第二組密文來向伺服器發送，而padding oracle的做法，在於不斷改變IV來發送給伺服器，從而找出因填空上的漏洞，先將IV改為全部都為0，而最後一個是我要嘗試的文字，如下

IV= 000000000000000000000000000000a1 (第一組改0)

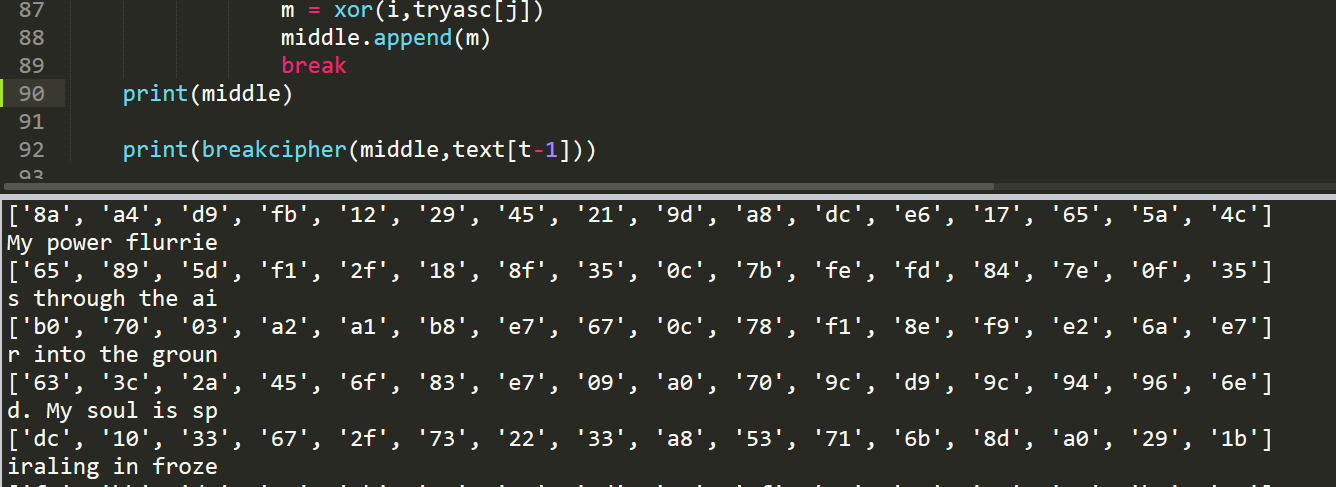
C = 462f0aec8f910e6b5daf6c47947de80c (第二組)

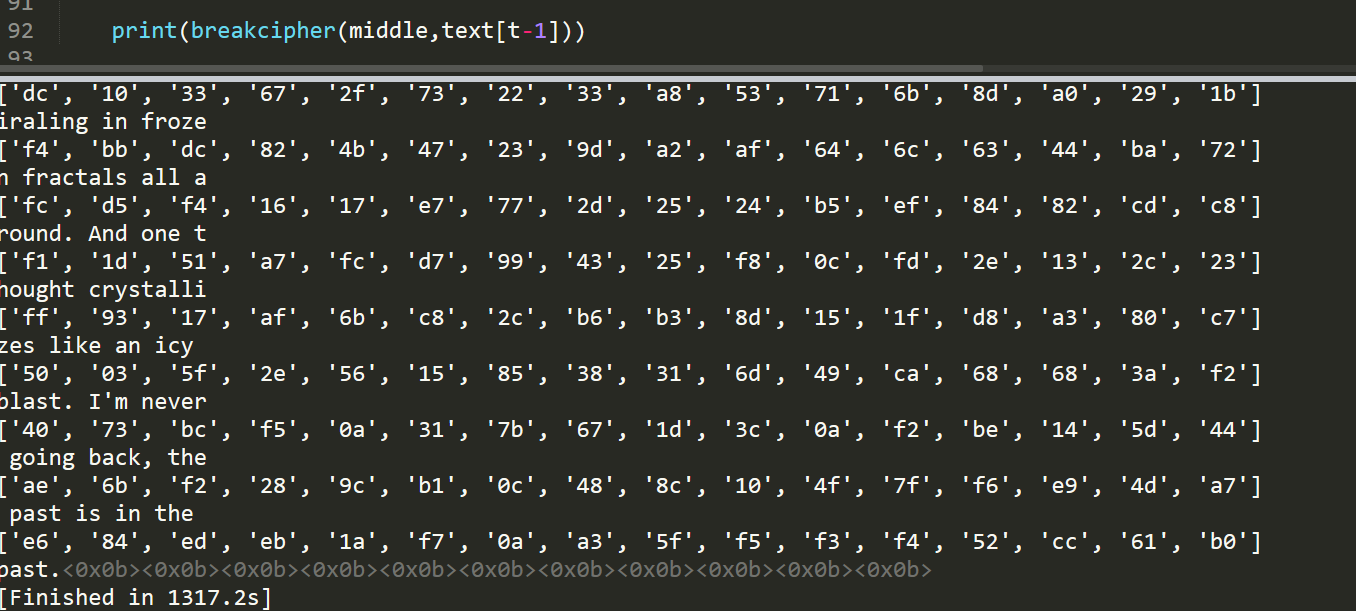
在此情況下，伺服器解密的文字會因填空的漏洞，解出來的明文最後一段會是0x01，因此我將IV最後一個字不斷改變從00,01,02…ff全部嘗試，而IV最後一個為8b的時候，伺服器回傳了valid，因此原始的值就為8b xor 01 = 8a

再來，解倒數第二個文字，密文填空情形，倒數兩個都會為0x02，而我們已經知道最後一個為8a，因此將8a xor 02=88，將IV最後一位改成88，倒數第二個一樣從00,01…ff全部嘗試，在a6的時候伺服器回valid，則原始倒數第二個為a6 xor 02 = a4，依此反覆解，將原始值的每個位置都解出來，最後我先得到第一組原始值middle，將此middle與密文的第一組進行xor後可得到My power flurrie

步驟3:

同樣的，因為aes cbc的加密方式，第一組密文是第二組的IV

，第二組加密過的則是第三組的IV，所以可分為說第一組密文塊用1,2塊來解密，第二組密文快用2,3塊來解，依此類推，將每一組密文重複步驟2可依序解出所有的明文

而最後一段解出來的填控則是0b，也的確與密文填控規則一樣，因此明文及解出(跑最後一段伺服器好像快炸掉了，最後跑了20多分鐘)。

1. **Lab : RSA**