# 離散數學\_RAS程式設計作業

## 資工二甲 11111132 楊敦傑

## 心得感想和實作過程:

本次的作業對於RAS以Python程式語言實作,因為Python有支援大數,所以並不用去處理大數的問題。

首先,在RSA加密前,需要先確保公鑰(e)與(p-1)\*(q-1)的最大公因數為1,這樣代表兩者互質,才可以使後續的公式能正常執行。不僅如此,還必須確認明文的長度是否為2的倍數,因為題目中已經設定了加解密的區塊大小為每區塊2個字元,如果不是,就需要在最後加上一個空字符,以符合加密的規範。

接著就來到了建立私鑰的部分。這涉及到了數學的概念,就是反元素。為了算出這個反元素,我們需要使用輾轉相除法的方式來計算。通過輾轉相除法後,就可以求得私鑰(d)。接下來,需要先將字母轉換為數字,利用公鑰(e)並經過加密後,就可以得到密鑰C。然後,利用私鑰(d)對C進行解密後就可以得到真正的資料M

在製作RSA程式的過程中,我深刻體驗到了加解密資料的複雜,並且也體會到了數學公式的強大,這種感覺特別是在處理大數和模反元素等問題時,利用普通硬炸的方式會讓計算變得特別緩慢,但是在利用輾轉相除法等數學技巧,就能夠快速且有效地計算出RSA所需要的東西。

此外, 經過這次的實作我也認知到了在加密算法中, 每個步驟都扮演著至關重要的角色, 尤其是需要保證資料完整性和準確性的情況下, 一點小問題沒有考慮進去的話都會釀成巨大的錯誤。就好比原本我在RSA解密的地方使用的是一個一個字元去做的, 但是程式跑出來的解果總跟我想的不一樣, 後來才發現我的判定範圍設置錯誤了。所以我從中就了解到了在加解密中條件和步驟的重要性

總體而言, RSA解碼的實踐是極具挑戰性的過程。而使用Python作為工具, 也讓我體驗了這個程式語言背後所具有的可能性, 並且還了解了加解密實際應用中所需的準確性和條件設立的重要性。

程式執行結果:

基本款:

```
問題 輸出 傾揮主控台 <u>終端機</u> 這接壞
PS C:\Users\USER\Documents\GitHub\Mid1_2\ python -u "c:\Users\USER\Documents\GitHub\Mid1_2\RSA_test.py" 請輸入e:13 請輸入p:43 請輸入q:59 (n, d): (2537, 937) 請輸入所要加密之文字:STOP M = 1819 1415 C = 2081 2182 解碼後: STOP
PS C:\Users\USER\Documents\GitHub\Mid1_2> ■
```

#### 中階款:

### 進階版(因為太長的無法截圖所以直接使用文字檔):

請輸入e:723732698112534512672761745123638923636112736572367 請輸入p:389651984374348681198937829437632876251983672816323 請輸入q:672838012783469248520967128763116286378291649376839 (n, d):

(2621726668435721778351896929354906266819645267015514933756485016427962614 83999546501062577067057342997,

2428535886755840714481288508164141304354488932723102643091004967165746926 43023247001636249914195987867)

請輸入所要加密之文字:RSA INVOLVES A PUBLIC KEY AND A PRIVATE KEY THE PUBLIC KEY CAN BE KNOWN BY EVERYONE AND IS USED FOR ENCRYPTION MESSAGES MESSAGES ENCRYPTED WITH THE PUBLIC KEY CAN BE ONLY BE DECRYPTED IN A REASONABLE AMOUNT OF TIME USING THE PRIVATE KEY THE KEYS FOR THE RSA ALGORITHM ARE GENERATED THE FOLLOWING WAY M = 1718 26 813 2114 1121 418 2600 2615 2001 1108 226 1004 2426 13 326 26 1517 821 19 426 1004 2426 1907 426 1520 111 802 2610 424 2602 13 2601 426 1013 1422 1326 124 2604 2104 1724 1413 426 13 326 818 2620 1804 326 514 1726 413 217 2415 1908 1413 2612 418 1800 604 1826 1204 1818 6 418 2604 1302 1724 1519 403 2622 819 726 1907 426 1520 111 802 2610 424 2602 13 2601 426 1413 1124 2601 426 304 217 2415 1904 326

813 2600 2617 400 1814 1300 111 426 12 1420 1319 2614 526 1908 1204 2620 1808 1306 2619 704 2615 1708 2100 1904 2610 424 2619 704 2610 424 1826 514 1726 1907 426 1718 26 11 614 1708 1907 1226 17 426 604 1304 1700 1904 326 1907 426 514 1111 1422 813 626 2200 2426

C =

5793420360909026563611586222239419704734607750521660824871202020081809488 7857035677684576152154442459

14718312285820248762240687352442125298587089778984611177250047730528798258 2719140839214378937996960206

1050788102194969793515795480050716680428449042233884701330990369961432108 35349342902259460769764219109

7538597973737912294353808266427641139782778760632621592956359426578800342 1487357023100335638818639841

1754838444064106270087522400877924628233741794369472923035901730300414851 55350775720604556564431893317

- 1099489323266831026957370276173321404731359682411879677060866876235417324 06432334873901719088135201574
- 13753045636277573724582983178168319629261772909439119847624691193006160150 3719434026517119968823511

- 6389002526078459317658567107070152386342330439972441812636557686716088420 7934594584958213203269355092

- 1315297607363672123600778259562629895717107750619255036306542238135247679 02927931631797635824409725049

- 1031826787163489789355550985379434928670729818328078947894526822566831283 90346606255682716715088326418

7415726570315139810864393162594518901100545215564940237008029940819658973 6548602651085607202801775831

1420080353326310666275623620499432384394597406552899142557973965739887809 56199718558941309575204697606

1099489323266831026957370276173321404731359682411879677060866876235417324 06432334873901719088135201574

13753045636277573724582983178168319629261772909439119847624691193006160150 3719434026517119968823511

2592140511478623196247614713869226055768764595865479268008767365327159049 62559466048420118914589507544

1099489323266831026957370276173321404731359682411879677060866876235417324 06432334873901719088135201574

1864107135306267079805104555871200949326494398180200871182722041061267336 31305141669784904289805005178

1315297607363672123600778259562629895717107750619255036306542238135247679 02927931631797635824409725049

1031826787163489789355550985379434928670729818328078947894526822566831283 90346606255682716715088326418

1782350495508305087702978074370683434252602101979230929820578786223822051 97111425529267398618149681895

1758768351310856124600193599008023176813874832144456988771677743808376801 28298480195315084978078052799

77436334283747662154957508190581005164722111211562545572444771171341556606 476171418580839127898220664

1050788102194969793515795480050716680428449042233884701330990369961432108 35349342902259460769764219109

解碼後: RSA INVOLVES A PUBLIC KEY AND A PRIVATE KEY THE PUBLIC KEY CAN BE KNOWN BY EVERYONE AND IS USED FOR ENCRYPTION MESSAGES MESSAGES ENCRYPTED WITH THE PUBLIC KEY CAN BE ONLY BE DECRYPTED IN A REASONABLE AMOUNT OF TIME USING THE PRIVATE KEY THE KEYS FOR THE RSA ALGORITHM ARE GENERATED THE FOLLOWING WAY