數學傳播　30卷1期, pp. 61-80

傳統密碼之旅(上)

沈淵源

一. 話從前說今朝

人總是喜愛保守一些資訊不爲他人所知; 由個人私事乃至國家大事, 自古皆然。就個人而言, 從小我們就有這種傾向, 這可從小時候所玩的各種遊戲窺見一二; 有時是互遞紙條、有時是交頭接耳、有時是竊竊私語, 而保守秘密的對象則是父母、兄弟姊妹、同學朋友或老師。就國家而言, 試圖保守其軍事秘密不爲敵方所知, 在歷史中處處可見。國王及衆將領使用一些最基本的密碼方法來跟他們的部隊聯絡, 爲的是防止敵方知道他們的重要軍事消息。事實上, 凱撒大帝[1](Gaius Julius Caesar) 就曾使用過一個非常簡單的密碼系統, 後人就以他的名字來稱呼這個密碼系統。

隨著社會的進步, 私人、公司與國家的權益變得更機密更敏感, 所以使用更精巧細膩的方法來保護資料的需求與日俱增。現在, 資訊的世代就活現在我們眼前, 此種需求當然比以往更加顯著。當世界變得更密不可分時, 人們對資訊及電子服務的需求就會不斷的加增, 而更多的需求帶來對電子系統更大的依賴。目前, 透過網際網路來交換重要資訊, 如信用卡號碼者, 已是司空見慣且極爲平常。所以保護資料與電子系統之安全, 對我們的生活方式而言, 也是不可缺少的一環。

保護資料所需的技巧, 說來是精彩絕倫且令人拍案叫絕, 一般將其歸屬於密碼術 (Cryptography) 的領域。近代密碼術可說是奠基於數學、電腦科學及聰明智慧 (wisdom) 上的一門學科, 而其程度既深且厚。先討論幾個可能在你腦海裡盤旋已久的問題。

密碼學、密碼術與破密學

首先有三個名詞需要釐清其相互之間關係的, 就是密碼學還有密碼術以及破密學。這三個術語經常被互相使用。然而在專業上, 密碼學乃是研究經由不安全頻道 (non-secure channels)

[1]凱撒(100-44B.C.) 古羅馬的名將及政治家。西元前60 年與龐培、克拉蘇結成前三頭政治。曾經征服高盧, 入侵日耳曼、不列顛、小亞細亞, 並助埃及女王克里奧佩托拉爭奪王位。西元前 46 年建立獨裁政治, 集執政官、護民官、獨裁官等大權於一身。主政期間, 除積弊, 重農工, 獎勵文學, 採用埃及的太陽曆, 威名卓著。

來傳輸資訊及其相關問題之學問的一個全稱術語。設計系統來完成此任務之過程稱爲密碼術, 而破解此系統之技術稱爲破密學或密碼分析學。所以密碼學一詞包括後兩者, 亦即

密碼學 (Cryptology)=密碼術 (Cryptography)+ 破密學 (Cryptanalysis)

當然, 若對兩方面的方法沒有一個好的理解, 那麼在本質上是不可能搞好其中任何一門的。譬如說, 你若想設計一個密碼系統, 你就得對人家如何破解一密碼系統的種種方法有深入的了解; 如此才能考慮各個層面, 使你的密碼系統足以承受各類不同攻擊的挑戰。

什麼是編碼理論 (Coding Theory)?

編碼理論研究經由吵雜頻道 (noisy channels) 的傳輸問題並處理如錯誤更正碼等之課題。其重點不在如何防止第三者閱讀其信息, 而在於確認收取的信息是正確的如所傳輸前的本文一樣。錯誤更正碼經常會配合使用在加密的場合, 因爲任何好的加密系統只要密文中有一個位元的差錯就足以完全毀掉整個的信息。

是否存在無法破解的密碼? 若有, 為何不全時間都使用呢?

答案是肯定的, 確實存在有無法破解的密碼系統 (cryptosystem), 如單次鑰匙簿密碼 (One-time Pad); 而且也存在有密碼系統如 RSA 者, 只要執行妥當, 在你有生之年也是無法破解的。然而這裡所牽涉到的乃是一個經濟問題, 時間、費用以及安全性, 那個重要? 使用單次鑰匙簿密碼的費用是相當龐大的, 而類似 RSA 的許多系統, 對信息量大的應用來說, 其速度是有待改進的。此時就得看你的需要如何, 來作一妥協與調整。有線電視廣播所需求的安全等級與政府部門對機密文件所需求的安全等級, 這之間當然會有天壤之別。

在很多的情況中, 從數學的觀點來看, 我們可藉著稍微調增鑰匙的長度來提高系統的安全性。但這不見得每次管用。怎麼說呢? 如果你所用的晶片可處理 64 位元, 那麼當你把鑰匙長度提昇至 65 位元時, 就可能意味著要對硬體作一全盤重新的設計。如此一來, 那所付出的費用可就昂貴無比了。因此, 設計一個良好的密碼系統所牽涉到的, 不僅僅只有數學而且還有工程以及其他方面的考量。

最後我們提出一個警告:

大數如丈二金剛, 令人摸不著頭腦

直觀上, 我們會覺得處理一個 20 位數是處理一個 10 位數的兩倍功夫。這對某些演算法來說是正確的。然而你若仔細想想, 數到10^{10} 其實離 10^{20} 還遙遠的很呢, 你得再努力一百億倍才能達到目標。當數字越來越大時, 這類型的景況就會更加的顯著。

許多時候, 我們所碰到的情況, 看起來似乎只要試過所有的可能性, 就能把一個密碼系統破解。然而, 說說容易但做起來可就不是那麼一回事。假設你的電腦每秒鐘可處理 10^9 次某種

運算, 而你需要試 10^{30} 種可能性後才能達成任務。因爲一年約有 3×10^7 秒, 所以要完成此項任務所需的時間比 3×10^{13} 年還要長一些, 而這又比宇宙的預測年齡還要長。

在我們所要介紹的幾個系統之一, 其安全性繫乎分解一個約 200 位的大整數之困難度上。假設你要分解一個這麼大的整數 n, 所用的乃是小學生的方法, 亦即將 n 除以比其平方根小的所有正質數。小於 10^{100} 的質數共有 4×10^{97} 個。要每個都試, 那是行不通的。據估計, 全世界電子的總數目不會超過 10^{90}。所以遠在完成計算之前, 電力公司必定會打電話來制止你的蠢動。很明顯的, 我們必須使用其他更細膩精緻的因數分解法。

一個數的大小, 除了其實際大小之外, 也可以由其十進 (或二進) 制表示法中的位數來衡量。此數大約是 $\log_{10} n$ (或 $\log_2 n$)。若我們使用小學生所用的標準乘算法, 將一個 k 位數 n 平方時所需執行的乘法的個數爲 k^2, 大約是 $(\log_{10} n)^2$。要分解一數 n(將此數除以比其平方根小的所有正質數) 所需的除法個數大約是 $n^{1/2}$。執行一個演算法所需的時間爲 $\log n$ 的次冪遠比其時間爲 n 的次冪好很多。在上面所舉的例子中, 若將 n 的位數變爲兩倍則計算其平方所需的時間變爲原先的四倍, 但將此數分解所需的時間卻大大的增加。當然, 兩者都有更好的演算法可供使用; 但以目前的情形來看, 分解因數遠比平方費時許多。

在執行計算時, 我們已經在前面碰到一些演算法, 其所需的時間是 $\log n$ 的次冪; 譬如, 求最大公因數及模運算中的指數運算。但有一些其他的計算, 就已知最好的演算法, 其執行的時間僅僅比 n 的次冪好一些; 譬如, 分解因數問題及解離散對數問題。快與慢的演算法之間的相互影響, 乃是我們在這課程中, 所要學習的幾個密碼演算法的基礎。所以, 現在就讓我們歡欣上路, 開始這趟令人興奮無比的密碼學之旅。

二. 旅遊須知

此旅有三個主要人物: 張三毛、李四郎和王五爺。三毛要傳遞信息, 稱之爲明文 (plain-text) 給四郎。有可能五爺會竊取此一信息, 所以三毛必須將此一信息加密 (用事先與四郎安排好的方法)。通常我們假設五爺已經知道這個加密的方法, 之所以能保持此一信息秘密的關鍵在於一把所謂的鑰匙 (key)。當四郎接收到此一密文, 他就用鑰匙將其解密還原成原來的明文; 此一鑰匙有可能不同於三毛所用的加密鑰匙, 但與加密鑰匙有關。三毛希望五爺沒有本事找到或計算出解密鑰匙, 所以也就沒有辦法讀取她所傳遞給四郎的信息。

五爺有四種可能的破解方式。

1. 密文攻擊法: 五爺僅擁有一份密文, 他只能由此來進行破解。

2. 已知明文攻擊法: 五爺擁有一份密文及其對應的明文。譬如, 五爺截取到一份通訊社所發出的密文稿, 接著於次日在報紙上看到解密後的明文稿。如果他能由此計算出解密鑰匙, 且三毛沒有更改加密鑰匙, 那麼五爺就能讀取所有未來截取到的信息。或者, 如果三毛總是以「Dear Lee」爲起頭送信息給四郎, 那麼五爺就有一短短的密文及其對應的明文信息。對許多較弱的密碼系統, 這就足以使我們找到鑰匙。即使是強一些的系統, 如二次大戰時德軍所使用過的密碼器奇謎 (Enigma), 這一丁點的資訊已經很夠用了。
3. 選擇明文攻擊法: 五爺得一短暫的機會可以使用加密器。他無法打開加密器去找鑰匙; 然而他可以適當的選取大量的明文, 再試著利用由加密器所得到的密文來破解。
4. 選擇密文攻擊法: 五爺得到機會可以使用解密器。他用來解密幾串選好的符號, 並試著用此結果來找到鑰匙。

選擇明文攻擊法可能會用到的時機如下。要鑑定一部飛機是友是敵, 可任選一信息傳送給這部飛機, 這部飛機會自動將信息加密並送回。我們假設僅僅友機持有正確的鑰匙。將送回的信息與正確加密過的信息作一比較。若吻合則爲友機; 若否, 則爲敵機。然而, 敵方可送出大量選擇好的信息至機群中的一架, 然後觀察所得到的密文。如果這樣就讓他們找到鑰匙, 那麼敵方就可裝備其飛機來僞裝成友機。

據報導, 在二次世界大戰的薩哈拉沙漠戰役中, 有一個使用已知明文攻擊法的實例。有一孤立的德軍前哨基地, 每天送出一則一模一樣的信息說:

「無新鮮事報告(there is nothing new to report)」。

所以每天盟軍都有一明文密文對, 這對決定當天的鑰匙是極爲有用的。事實上, 在整個戰役當中, 蒙哥馬利將軍 (General Montgomery) 小心翼翼的繞過這個前哨基地; 爲的是讓此信息的傳送不至於中斷。

柯克霍夫斯原理 (Kerckhoffs' Principle)

通常, 一個秘密信息是隱藏著的[2], 爲的是希望沒有人知道有信息正在傳遞當中。這種掩飾訊息存在性的保密通訊法稱之爲隱匿法[3]。在其他時候, 敵方可能一開始不知道你所使用的加密法。然而爲了提昇一個密碼系統的安全等級, 你必須永遠假設敵方知道你所使用的方法。換句話說, 一個密碼系統應該是安全的, 即使此系統除了鑰匙之外所有的一切, 是人盡皆知的。更精簡的說: 一切盡在密鑰中。此一原理乃是由荷蘭人奧古斯特·柯克霍夫斯 (Auguste Kerckhoffs)

---

[2]譬如, 爲某文中每一句的第三個字母。

[3]隱匿法 (steganography), 源自希臘文 steganos 與 graphein 二字, 前者意思就是掩蔽的, 而後者則爲書寫。請參閱蓋瑞·柯思樂 (Gary C. Kessler) 的介紹文章 [8] *Steganography: Hiding Data Within Data*, September, 2001。

於 1883 年在他的經典之作軍事密碼術 [7] (La Cryptographie Militaire) 中首先提出的。另一方面, 克勞德·解藍恩 (Claude Shannon) 有可能是獨立地以更精簡的話表達爲: 敵方知悉一切 (the enemy knows the system), 此版本被稱之爲解藍恩的格言 (Shannon's maxim)。所有這些可能會以許多不同的方式彰顯出來。例如, 加密器或解密器被擄獲且被分析過, 或人員的變節或者被捕。因此之故, 整個系統的安全性必須僅僅仰賴其鑰匙的秘密性。

為何介紹古典密碼系統?

在下面, 我們會先開始介紹幾個簡單無比而又明白易懂的古典密碼系統, 如位移密碼 (Shift Ciphers)、仿射密碼 (Affine Ciphers) 還有維吉內爾密碼 (Vigenère Cipher)、希爾密碼 (Hill Ciphers) 以及代換密碼 (Substitution Ciphers) 等等。這些系統本身都太脆弱了, 並沒有太大的用處, 尤其是在有電腦隨時侍候在你身旁的今天。但至少我們可藉著這些極其簡單的古典密碼系統, 來說明密碼學當中一些重要的構想及其背後所隱藏的基本觀念, 算來也有其啓發之功吧。

爲了方便起見, 首先我們作如下的規定:

- 小寫字母表示明文, 而大寫字母則表示密文。
- 英文的 26 個字母其對應的數字如下:

a	b	c	d	e	f	g	h	i	j	k	l	m	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13

o	p	q	r	s	t	u	v	w	x	y	z
14	15	16	17	18	19	20	21	22	23	24	25

注意我們是從 a=0 開始的, 所以 z 所對所應的數爲 25。這令人困擾, 但卻是一種標準的用法。

- 我們將空白與標點省略掉。這是更令人困擾, 但解密後之明文幾乎都很容易將其空白處找出。若空白也加入密文當中, 可能我們要面臨兩個抉擇。其一可將空白處仍然維持空白, 但這會告訴破密者你整個信息的結構, 使得解密更加容易。其二可將空白處加密成爲密文的一部分, 不過如此一來卻變成出現頻率最高的, 那跟前面一樣容易破解。(除非信息中, 每個字平均長度至少 8 個字母)

三. 位移密碼(Shift Ciphers)

這是最早的密碼系統之一, 通常歸功於凱撒大帝, 所以又稱之爲凱撒密碼。假設凱撒大帝他想傳達一個信息, 如

gaul is divided into three parts。

但他不要 Brutus 知道, 他將每個字母移動三個位置, 所以 a 變成 D, b 變成 E, c 變成 F, ... 等; 後面的字母跑到前面, 所以 x 變成 A, y 變成 B, z 變成 C。如此這般的加密後密文爲

JDXOLVGLYLGHGLQWRWKUHHSDUWV。

解密時只要移回三個位置即可 (當然還得花心思將空白處確定)。一般情況如下: 先將字母依序標示爲一整數由 0 至 25, 鑰匙爲一整數 k, $0<k\leq 25$。加密的演算法爲

$$x \mapsto x+k \pmod{26}。$$

而解密演算法則爲 $x \mapsto x-k \pmod{26}$。譬如, 凱撒所用的 k 值爲 3。

下面我們一起來思考那四種攻擊法如何進行。

1. 密文攻擊法: 王五僅有一份密文。他最好的策略就是做一個地毯式的搜索, 因爲就只有 26 種可能的鑰匙。若信息有好幾個字母長, 則不太可能會有超過一個以上 (26 個裡面) 爲有意義。如果你不相信, 可試試看找一個 4 或 5 個字母的英文字, 然後觀察位移轉換後的 26 種不同的結果。你可用 MATHEMATICA 寫一個簡單的程式來幫助你完成上面的實驗。
   另一可行的方法是 (若信息夠長的話), 對出現過的字母作頻率分析, 在英文中, 字母 e 最常出現在單字裡面。上面的例子中, 字母 L 在密文中出現最多次 (4 次); 因爲 e=4 與 L=11, 所以一個合理的的猜測爲 $k=11-4=7$, (與實際的 $k=3$ 不符合, 原因出在信息不夠長, 頻率分析很難使的上力)。對位移密碼而言, 此種頻率分析反而會比地毯式搜索更花時間, 加上有時信息長度不夠也會徒勞無功。
2. 已知明文攻擊法: 若你只知道明文中的一個字母及其在密文中對應的字母, 則鑰匙馬上可以得知。如你已知 t(=19) 加密爲 D(=3), 那麼你的鑰匙就是 $k\equiv 3-19\equiv -16\equiv 10 \pmod{26}$。
3. 選擇明文攻擊法: 選取字母 a 爲明文, 則其對應的密文就是鑰匙。例如, 在密文中對應的字母爲 H, 則其鑰匙就是 k=7。
4. 選擇密文攻擊法: 選取字母 A 爲密文, 則其對應的明文之負值就是鑰匙。例如, 在明文中對應字母爲 h, 則其鑰匙就是 $-7\equiv 19 \pmod{26}$。

位移密碼雖是簡單至極, 但若有大量的資料要執行加密的話, 還是得仰賴電腦的幫助才行。在 MATHEMATICA 中, 明文、密文輸入爲一字串, 需要用引號 " " 把它括起來。也可輸入 `txt="hihowareyou"`, 往後當你輸入的是 `txt` 時 (不需用引號) 指的就是上述那個字串。我們定義如下指令來執行相關的加密、解密動作:

- `shiftencrypt[txt,k]` 會將小寫明文字串 `txt` 中的每個字母往後移動 k 個位置, 然後以大寫輸出成爲密文。
- `shiftdecrypt[txt,k]` 會將大寫密文字串 `txt` 中的每個字母往後移動 k 個位置, 然後以小寫輸出成爲明文。
- `shiftdecryptall[txt]`會將大寫密文字串 `txt` 位移之後的26個不同的字串全部以小寫列出。

上述非 MATHEMATICA 內建指令之程式如下:

```
In[1]:=abc="abcdefghijklmnopqrstuvwxyz";cap="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
       shiftencryptkey[n_]:=Table[StringTake[abc,{i}]->StringTake[cap,{Mod[i-1+n,26]+1}],{i,1,26}];
       shiftencrypt[plaintext_,n_]:=StringReplace[plaintext, shiftencryptkey[n]];
       shiftdecryptkey[n_]:=Table[StringTake[cap,{i}]->StringTake[abc,{Mod[i-1+n,26]+1}],{i,1,26}];
       shiftdecrypt[plaintext_,n_]:=StringReplace[plaintext, shiftdecryptkey[n]];
       shiftdecryptall[txt_]:=Do[Print[shiftdecrypt[txt,n]],{n,26}];
       ga="gaulisdividedintothreeparts";
       yc="YCVEJQWVHQTDTWVWU";
```

例題01. 用位移密碼 $\mathtt{k} = 3$ 來加密 gaul is divided into three parts。

解: 明文 ga 已輸入在 `In[1]` 中, 執行指令 `shiftencrypt` 於其上得密文爲 `cryptga`; 然後執行指令 `shiftdecrypt` 驗證之, 如下:

```
In[10]:= cryptga=shiftencrypt[ga,3]
out[10]= JDXOLVGLYLGHGLQWRWKUHHSDUWV
In[11]:= shiftdecrypt[%,-3]
out[11]= gaulisdividedintothreeparts
```

例題02: 經位移密碼加密爲 `YCVEJGWVHQTDTWVWU`, 試解密之。

解: 密文 `yc` 已輸入在 `In[1]` 中, 執行指令 `shiftdecryptall` 於其上得可能的明文如下:

```
In[12]:= shiftdecryptall[yc]
         zdwfkrxwirueuxwxv
         aexglsyxjsvfvyxyw
         bfyhmtzyktwgwzyzx
         cgzinuazluxhxazay
         dhajovbamvyiybabz
         eibkpwcbnwzjzcbca
         fjclqxdcoxakadcdb
         gkdmryedpyblbedec
         hlenszfeqzcmcfefd
         imfotagfradndgfge
         jngpubhgsbeoehghf
         kohqvcihtcfpfihig
         lpirwdjiudgqgjijh
         mqjsxekjvehrhkjki
         nrktyflkwfisilklj
         osluzgmlxgjtjmlmk
         ptmvahnmyhkuknmnl
         qunwbionzilvlonom
         rvoxcjpoajmwmpopn
         swpydkqpbknxnqpqo
         txqzelrqcloyorqrp
         uyrafmsrdmpzpsrsq
         vzsbgntsenqaqtstr
         watchoutforbrutus
         xbudipvugpscsvuvt
         ycvejqwvhqtdtwvwu
```

可看出 watch out for brutus 應該就是原來的明文。

四. 仿射密碼(Affine Ciphers)

位移密碼可推廣並稍加強化, 如下:

選取兩個整數 α 及 β 並考慮函數 (稱之爲仿射函數)

$$x \mapsto \alpha x + \beta \pmod{26}。$$

譬如, 令 $\alpha = 9$ 及 $\beta = 2$, 即函數爲 $9x + 2$。取明文字母 h(=7) 被加密後成爲 $9 \cdot 7 + 2 \equiv 65 \equiv 13 \pmod{26}$, 亦即字母 N。依此我們得到

$$\text{tunghai} \mapsto \text{RAPENCW}。$$

怎麼解密呢? 我們由方程式來解 x。若是在有理數當中, 其解爲

$$x = \frac{1}{9}(y - 2),$$

但其中的 $\frac{1}{9}$ 需作不同的詮釋, 因爲我們實際上是在模 26 的數系中。然而這難不倒我們, 因爲 $\gcd(9, 26) = 1$, 所以 9 在模 26 中有一乘法反元素。因 $9 \cdot 3 \equiv 1 \pmod{26}$, 所以 3 就是所要的反元素。可用來取代 $\frac{1}{9}$, 我們有

$$x \equiv 3(y - 2) \equiv 3y - 6 \equiv 3y + 20 \pmod{26}。$$

此爲解密函數, 所以字母 L(=11) 映到 $3 \cdot 17 + 20 \equiv 71 \equiv 19 \pmod{26}$, 亦即字母 t。同樣的方式, 我們將密文 RAPENCW 解密回到 tunghai。

假設我們試圖用函數 $13x + 4$ 當成我們的加密函數。我們得到

$$\text{input} \mapsto \text{ERRER}。$$

如果我們更改(alter) 一下輸入 (input), 則得到

$$\text{alter} \mapsto \text{ERRER}。$$

顯而易見, 這個函數導致錯誤。我們不可能解密, 此乃因爲有不同的明文對應到同一個密文 (ERRER)。特別要注意到的是加密函數應該是一對一的, 但這個例子則否。

到底哪兒出了問題呢? 若解這個方程式 $y = 13x + 4$, 可得 $x = \frac{1}{13}(y - 4)$。但 $\frac{1}{13}$ 在模 26 中不存在, 此乃因爲 $\gcd(13, 26) = 13 \neq 1$。一般而言, 可以證明在模 26 之下, 函數 $\alpha x + \beta$ 爲一對一若且唯若 $\gcd(\alpha, 26) = 1$。在這條件下, 解密函數爲 $x \equiv \alpha^* y - \alpha^* \beta \pmod{26}$, 此處 $\alpha\alpha^* \equiv 1 \pmod{26}$。所以解密函數也是一個仿射函數。

這個加密方法的鑰匙爲一整數對 (α, β)。有 12 個可能的這種 α, 使得 $\gcd(\alpha, 26) = 1$, 而 β 則有 26 種可能的值。合起來共有 $12 \times 26 = 312$ 個可能的鑰匙。

下面我們看看可能破解的方法。

1. 密文攻擊法: 全面搜索 312 種可能性, 當然會比位移密碼時的 26 種來得久, 但對電腦而言, 這太簡單了, 不喘一口氣工作就完成了。當所有可能的鑰匙試過了之後, 看看那個是有意義的, 由此即可決定其鑰匙。其實, 話說說簡單, 眞的要從 312 種可能性當中挑出那個有意義的, 也是相當頭痛的。頻率分析當然也可以用, 不過需要更長的信息。
2. 已知明文攻擊法: 只要有一點點運氣, 知道明文中的兩個字母及其對應密文中的字母就足以找到鑰匙。舉例來說, 假設明文的起頭爲 if 而其對應的密文爲 PQ。以數字表示, 此即 8(=i) 映到 15(=P) 而 5 則映到 16。所以我們得到方程式

$$8\alpha + \beta \equiv 15 \quad \text{及} \quad 5\alpha + \beta \equiv 16 \pmod{26}。$$

相減後可得 $3\alpha \equiv -1 \equiv 25 \pmod{26}$, 其唯一的解爲 $\alpha = 17$。代入第二個方程式得 $5 \cdot 17 + \beta \equiv 16 \pmod{26}$, 最後得到 $\beta = 9$。假設明文爲 go, 其對應的密文爲 TH, 則得到方程式

$$6\alpha + \beta \equiv 19 \quad \text{及} \quad 14\alpha + \beta \equiv 7 \pmod{26}。$$

相減後可得 $-8\alpha \equiv 12 \pmod{26}$。因爲 $\gcd(-8, 26) = 2$, 所以得到兩個解 $\alpha = 5, 18$。其對應的 β 值都是 15 (這並非偶然而是必然如此)。所以我們有兩組可能的鑰匙, $(5, 15)$ 及 $(18, 15)$。然而, $\gcd(18, 26) \neq 1$ 所以第二組不可能。因此眞正的鑰匙是 $(5, 15)$。上面的程序應是通行無阻的, 除非所得到的最大公因數爲 13 或 26; 在此種情況下倘若可行就選用另外一個字母。假如我們知道僅僅明文中的一個字母, 我們仍然可以得到 α 與 β 的一個關係式。舉例說, 若我們只知道明文中的 g 對應到密文中的 T, 則 α 與 β 的關係式爲

$$6\alpha + \beta \equiv 19 \pmod{26}。$$

對 α 而言, 有 12 種可能的值, 而每一個值對應一個 β 值。因此全面搜索這 12 個鑰匙, 一定可以找到那眞正的鑰匙。
3. 選擇明文攻擊法: 選取 ab 爲明文。密文中的第一個字爲 $\alpha \cdot 0 + \beta = \beta$, 而第二個字則爲 $\alpha + \beta$。因此我們可以找到鑰匙。
4. 選擇密文攻擊法: 選取 AB 爲密文。由此可得解密函數爲 $x = \alpha_1 y + \beta_1$。解 y 然後得到鑰匙。但爲何如此大費周章呢? 我們要的不就是解密函數嗎?

底下我們定義指令 `affencrypt[txt,m,n]` 及 `affencrypt[txt,m,n]` 來執行仿射密碼的加、解密動作。

- `affencrypt[txt,m,n]` 執行仿射函數 $mx + n$ 於小寫明文字串 `txt` 上, 然後以大寫輸出成爲密文。

- `affdecrypt[txt,m,n]` 執行仿射函數 $mx+n$ 於大寫密文字串 `txt` 上, 然後以小寫輸出成爲明文。

上述非MATHEMATICA 內建指令之程式如下:

```
In[13]:= abc = "abcdefghijklmnopqrstuvwxyz"; cap = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
         affencryptkey[m_,n_]:=Table[StringTake[abc,{i}]->StringTake[cap,{Mod[m*(i-1)+n,26]+1}],{i,26}];
         affencrypt[plaintext_,m_,n_]:=StringReplace[plaintext, affencryptkey[m,n]];
         affdecryptkey[m_,n_]:=Table[StringTake[cap,{i}]->StringTake[abc,{Mod[m*(i-1)+n,26]+1}],{i,26}];
         affdecrypt[plaintext_,m_,n_]:=StringReplace[plaintext, affdecryptkey[m,n]];
         ed="EDSGICKXHUKLZVEQZVKXWKZUKCVUH";
         tc="TCABTIQMFHEQQMRMVMTMAQ";
```

例題03: 用仿射函數 $7x+8$ 來加密 i love you very much。

解: 執行指令 `affencrypt` 得密文 MHCZKUCSZKXUOSWF; 然後執行指令 `affencrypt` 驗證之, 如下:

```
In[20]:= affencrypt["iloveyouverymuch",7,8]
Out[20]= MHCZKUCSZKXUOSWF
In[21]:= affdecrypt["MHCZKUCSZKXUOSWF",15,10]
Out[21]= iloveyouverymuch
```

例題04: 用仿射函數 $5x+12$ 加密, 其密文爲 MZDVEZC。試解密之。

解: 首先, 解同餘式 $y \equiv 5x+12 \pmod{26}$, 得

$$x \equiv 5^{-1}(y-12) \equiv 21(y-12) \equiv 21y+8 \pmod{26}。$$

```
In[22]:= PolynomialMod[PowerMod[5,-1,26](y-12),26]
Out[22]= 8 + 21 y
In[23]:= Solve[{y==5*x+12,Modulus==26},x,Mode->Modular]
Out[23]= {{Modulus -> 26, x -> 8 + 21 y}}
```

再來, 執行指令 `affdecrypt["MZDVEZC",21,8]` 解密得明文如下:

```
In[24]:= affdecrypt["MZDVEZC",21,8]
Out[24]= anthony
```

例題05: 在模 26 之下用仿射函數加密。在密文中, 最常出現的字母爲 K, 其次爲 D。很合理地, 可假設這分別由 e 與 t 加密而來, 因爲這兩個字母是英文中最常出現的兩個字母。因此轉換成其對應的數字, 代入解密公式中可得:

$$10\alpha' + \beta' \equiv 4 \pmod{26}$$

$$3\alpha' + \beta' \equiv 19 \pmod{26}$$

將上面第一式減去第二式可得 $7\alpha' \equiv 4-19 \equiv 63 \pmod{26}$, 所以我們有 $\alpha' \equiv 9 \pmod{26}$ 而且有 $\beta' \equiv 18 \pmod{26}$。最後我們得到解密公式爲 $x \equiv 9y+18 \pmod{26}$。

五. 維吉內爾密碼(The Vigenère Cipher)

維吉內爾密碼乃是位移密碼的一個變形, 早在十六世紀末就已經被發現, 但它的起源可追溯到十五世紀的佛羅倫斯才子里昂·巴提斯塔·亞伯提[4] (Leon Battista Alberti)。在當時, 所有代換密碼在加密一封信函時都只用到一套密碼字母。亞伯提建議採用兩套、甚至更多套密碼字母, 在加密過程中交替使用, 以混淆破密者的分析。雖然這個構想是那個時代在密碼術上的一項重大的突破, 但亞伯提卻沒有把這個概念發展成一套完整具體可行的密碼系統。所以此項任務就落在之後的幾位學者身上。

- 第一位是德國的約翰尼斯·特里特繆斯 (Johannes Trithemius, 1462 出生) 當過修道院院長;
- 再來是喬凡尼·波塔 (Giovanni Porta, 1535 出生), 義大利科學家;
- 最後是勃雷茲·維吉內爾 (Blaise de Vigenère, 1523 出生), 當過法國外交官。維吉內爾 26 歲時被派去羅馬擔任兩年的外交官, 因而讀到亞伯提、特里特繆斯和波塔的相關著作。剛開始時, 他對密碼術的興趣純爲外交工作上的需要。到了 39 歲時, 維吉內爾認爲自己積蓄的錢足以支持他放棄工作, 專注於研究生涯了。此時他才仔細思考亞伯提、特里特繆斯和波塔的構想, 並將它們發展成一套既完整又強大的新密碼系統。

雖然亞伯提、特里特繆斯和波塔都有極重要的貢獻, 這套密碼系統仍被稱爲維吉內爾密碼 (Vigenère Cipher), 以彰顯維吉內爾將它發展成形的功勞。雖然在十九世紀時, 查理·巴比齊 (Charles Babbage) 與弗瑞德瑞克·威爾赫倫·卡西斯基 (Friedrich Wilhelm Kasiski) 已展示出如何破解之法, 但是一直到二十世紀初期, 這個密碼系統被許多人認爲是安全的。終於在 1920 年代, 威廉·佛里德曼[5] (William F. Friedman, 1891-1969) 發展出額外的技巧將這個密碼系統及其相關的密碼系統破解。

此密碼系統之鑰匙爲一向量, 可選取如下: 首先選取一長度, 如 7, 然後選一個這種長度的向量, 其中之元素爲介於 0 與 25 之間的整數, 如 $k = (19, 20, 13, 6, 7, 0, 8)$ 。通常鑰匙對應於一個較容易記憶的字。在我們的例子中, 此字爲 tunghai。

用此 $k = (19, 20, 13, 6, 7, 0, 8)$ 可將信息加密如下: 把明文中的第一個字母移動 19 個位置, 第二個字母移動 20 個位置, 第三個字母移動 13 個位置, 依此類推。直到鑰匙的最後一

---

[4]亞伯提(1404–14xx) 是文藝復興的重要人物之一。他既是畫家、作曲家、詩人、哲學家, 也是第一篇科學性分析透視法的論文、探討家蠅的論文以及給愛犬的祭文之作者。他最著名的身分是建築師—羅馬翠維噴水池 (Trevi Fountain) 的原始設計者, 著有第一本印刷出版的建築論著《論建築 (De re aedificatoria)》, 此書是建築設計從哥德式樣轉變成文藝復興式樣的催化劑。

[5]佛里德曼是美國軍方密碼學家。許多人認爲他是歷史上最偉大的密碼學家之一, 而他應用統計方法來破解密碼乃是此領域中最有意義的進展。詳情請參閱下列網頁之介紹 http://en.wikipedia.org/wiki/William_F._Friedman。

個分量時再回到第一個分量, 所以第八個字母移動 19 個位置, 第九個字母移動 20 個位置, $\cdots$ 等。如下所示:

(明文)	i	l	o	v	e	y	o	u	v	e	r	y	m	u	c	h	
(鑰匙)	19	20	13	6	7	0	8	19	20	13	6	7	0	8	19	20	
(密文)	B	F	B	B	L	Y	W	N	P	R	X	F	M	C	V	B	

底下我們定義下列兩個指令分別來執行維吉內爾密碼法的加密及解密動作:

- `vigencrypt[txt,v]` 用向量 `v` 將小寫字串 `txt` 透過維吉內爾密碼法加密, 密文以大寫字串顯示。
- `vigdecrypt[txt,v]` 用向量 `v` 將大寫字串 `txt` 透過維吉內爾密碼法解密, 明文以小寫字串顯示。

上述非 MATHEMATICA 內建指令之程式如下:

```
In[25]:= abc = "abcdefghijklmnopqrstuvwxyz"; cap = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
vigencryptkey[m_,n_]:=Table[StringTake[abc,{i}]->StringTake[cap,{Mod[i-1+m[[Mod[n-1,Length[m]]+1]],26]+1}],{i,1,26}];
vigencrypt[plaintext_,v_]:=StringJoin[Module[{z},z={}; Do[z=Insert[z,
StringReplace[Characters[plaintext][[i]],vigencryptkey[v,Mod[i-1,Length[v]]+1]],i],{i,StringLength[plaintext]}];z]];
vigdecryptkey[m_,n_]:=Table[StringTake[cap,{i}]->StringTake[abc,{Mod[i-1+m[[Mod[n-1,Length[m]]+1]],26]+1}],{i,1,26}];
vigdecrypt[plaintext_,v_]:=StringJoin[Module[{z},z={}; Do[z=Insert[z,
StringReplace[Characters[plaintext][[i]],vigdecryptkey[v,Mod[i-1,Length[v]]+1]],i],{i,StringLength[plaintext]}];z]];
```

現在我們就將這兩個指令用在上面解說的例子, 且看:

```
In[30]:= abc = "abcdefghijklmnopqrstuvwxyz";num="000102030405060708091011121314151617181920212223242 5";
         digitalize=Table[StringTake[abc,{i}]->StringTake[num,{2*i - 1, 2*i}], {i, 1, 26}];
         alphabetize=Table[StringTake[num,{2*i - 1, 2*i}] -> StringTake[abc, {i}], {i, 1, 26}];
         Q0[plaintext_]:=StringReplace[plaintext, digitalize];
In[35]:= Table[StringTake[Q0["tunghai"],{i,i+1}],{i,1,14,2}]// ToExpression
Out[35]= {19, 20, 13, 6, 7, 0, 8}
In[36]:= vigencrypt["iloveyouverymuch", tunghai]
Out[36]= BFBBLYWNPRXFMCVB
In[37]:= vigdecrypt[%, -tunghai]
Out[37]= iloveyouverymuch
```

接著我們一起來思想那四種攻擊法如何進行破解。

1. 密文攻擊法: 對密文攻擊法而言, 長久以來此法被認爲是安全的。主要原因在於加密過程當中, 每一個字母已經被分散到好幾個字母上面去了, 如前所述。所以光是頻率分析當然沒有用。但從加密法則裡得知, 只要算出鑰匙的長度, 那麼就可以將那些經同一位移數得到的密文集中在一起。那麼頻率分析就可以各個擊破, 此乃必然要採取的戰略。所以這種情況也是很容易就可以找到鑰匙。下下一節會有詳細的討論。
2. 已知明文攻擊法: 若有足夠的字母, 則已知明文攻擊法一定成功; 此乃因爲將密文減掉明文即鑰匙也。
3. 選擇明文攻擊法: 選取明文 aaaaa$\ldots$ 來執行選擇明文攻擊法, 那麼你馬上就可得到加密鑰匙。
4. 選擇密文攻擊法: 我們可使用密文 AAAAA$\ldots$ 得到解密鑰匙, 其實這就是加密鑰匙的負值。

六. 頻率分析管用嗎?

破密學用到下列事實: 在大部分的英語文章中字母出現的頻率是不一樣的。譬如 e 出現得比 z 頻繁許多, s 出現得也比 q 頻繁。這些頻率是經由貝克與派波 (Becker&Piper) 所編纂而成的,[6] 列表如下:

英文字母頻率表 **(Beker and Piper)**

a	b	c	d	e	f	g	h	i	j	k	l	m
.082	.015	.028	.043	.127	.022	.020	.061	.070	.002	.008	.040	.024

n	o	p	q	r	s	t	u	v	w	x	y	z
.067	.075	.019	.001	.060	.063	.091	.028	.010	.023	.001	.020	.001

據此, 貝克與派波將26個英文字母分成五組如下:

1. e 其概率大約爲 0.12
2. t, a, o, i, n, s, h, r 其概率介於 0.06至 0.09之間
3. d, l 其概率大約爲 0.04
4. c, u, m, w, f, g, y, p, b 其概率介於 0.015至 0.023之間
5. v, k, j, x, q, z 其概率略少於 0.01

二字串及三字串也可能有用。其排序如下:

- 30 個常用的二字串按序爲

  `th he in er an re ed on es st`

  `en at to nt ha nd ou ea ng as`

  `or ti is et it ar te se hi of`

- 12 個常用的三字串按序爲

  `the ing and her ere ent`

  `tha nth was eth for dth`

當然, 突變異常是會出現, 但這通常需要相當程度的努力來營造始成。作家恩斯特·敏仙特·懷特 (Ernest Vincent Wright) 所寫的 50110 個字的小說給茲比 [14] (Gadsby) 中沒有任何一個 e。更令人欽佩的是作家喬治·佩黑克 (George Perec) 用法文所寫的小說消失 (La Disparition) 其字數超過百萬, 同樣沒有半個 e (這不僅僅有伴隨著動詞等等而來的一般問題,

[6]這個相對頻率表的統計依據取自報紙和小說的章節, 共計 100362 個字母; 由貝克 (H. Becker) 與派波 (F. Piper) 編纂, 最早見於 [11] 一書。

而且幾乎是所有的陰性名詞和形容詞都必須迴避)。在吉爾伯特·阿碟爾 (Gilbert Adair)[7] 的英譯本書名爲虛空 (A Void) 裡也是沒有半個 e。但一般而言, 我們可以如此來認定: 只要有好幾百個字母的文稿, 上面提供了我們通常會碰到情況的一個粗略的估計。

如果我們有一個簡單的位移密碼, 則字母 e 變成密文中的某一個字母, 所以這個字母出現的頻率會跟 e 在原文中出現的頻率一樣。因此頻率分析之後, 鑰匙可能就會被逼現身。然而, 在上面維吉內爾密碼的例子當中, 字母 e 在密文中以 L 跟 R 的面目出現。假如我們使用長一些的明文, 那麼對應於位移 19, 20, 13, 6, 7, 0, 8 字母 e 有可能會被加密成 X, Y, R, K, L, E, M。但密文中的 X 不僅僅是來自 e, 字母 d 有可能會被加密成 X, 只要它在明文中的位置之對應的位移數是 20 的時候即可。同樣的, k, r, q, x, p 都有可能會被加密成 X, 所以 X 的頻率來自明文中的 e, d, k, r, q, x, p 的一個組合。如此一來, 頻率分析就很難玩下去了。實際上, 密文中的字母出現之頻率已被磨平變成與 1/26 非常接近。至少應該非常接近英文字母原始的分佈。

這兒是更具體的例子, 其密文字串共 1149 個字母, 稱之爲 yih 如下:

```
In[37]:= yih="YIHXZCWKYNTKSMOYAELAZLUTUVUZYUGNLRAULBANHBYIEZOOVMBVYJOVMCAKUTIGYJTHTQHHPUUCMBPRJPNTBVRXAYIGXQKKIKTNRJAOB\
          AYCXVPWLCGOVNBAUGGSLUXHNXLCZXUGKKEYNUYTVWEXUEKLNOTARJPNIZLRGACQOCYCHRBXMGOUGEAYGNLRBAUGTHTQHHBXHNGGUGOVNA\
          HWBTJEQOYQGUDAHXRJPCIMYQIHNTHHTKUDCKYJKHRMFYGUUAOKYNZIABMFRLPETWISZOABPUECLHIOYPUTEBHXRJPCIMYNVVRBBIAUMTP\
          TNSOLLLTMNLPNIELRYAIVZJYGJENHLGNVSMPBBNLRMZUIKAHMBLYOCEAMBNZAHIMHNZPOVFCTNALQOYVZPSIENBMLTPXLSOATQGANTKPZ\
          HJRXAHIMQRYOOCEXQUAHQLVHZPNIEUEMLRAXHFKDEKTHAUADMWCPGAEEXWNTUOBVIAYLCZTNRCLCIGHBZOATEIJZOIAZLBAUDBAYOXHVM\
          FYARPVQGANTKDMTXJNVSBKOTMSELAYEKOADXWBTZEKKUGKKIBYUEGIODXIHXWOWKJBCLRBHUQJVRLXNEGJTBAYJUYLLPCYRSIBMFRTVTM\
          GIERVNOKYZKTBMKQUGAWMLULNLRMUOGOACIGHRBLRNHLTKAWPTNGNLYLBXUKYEQMCFLVRCLNUKSIDBHTXHTPXLGUIELXXVIHTMWBRXLTW\
          MBRAUFQGCFNLDEHLXCOIKANUKFWPHZBANHBAYEKOADXNUAZFIKMBTVBTRUQBHNKXXVZPSZTNUKYFWKOFZVBMAYEKKELBWNZLDBHNUKNRM\
          TNGGZKZXGNOUIVZVRLVRMNMGNHTNKIZZOEAXBBTVRMWXRGKWMMUXKPNKKYNYLDLXPBZPOVMIGNHTKTOFKMOZPBVIOTPXSTGCEBAYYGZTN\
          NFYSLAANLRUMDMOIGOVNBAUGCLHMKYUONHTRLRYVLDXNUGATPXMRJLALLBNRSNWMBNBLDQXXVTCAQGNUGATPBMAGAIWGOAJLROHXFNHLT\
          AUIKHNMPVVXAHWYZEKLDWFUAJAHIMABBLRVFYAZVFBAYCKVPTXVLZOEXXICRLFWKNUKWEWIFRYOATEHBZWEZBMULYOUMBRKHRBA";
```

爲了數點每一個字母出現的次數, 我們利用 MATHEMATICA 寫出下面的指令來完成任務:

- `frequency[txt]` 將大寫字串 `txt` 中各字母出現的次數列表。

```
In[38]:= cap = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
         frequency[txt_]:= Array[Function[i,{Characters[cap][[i]],Count[Characters[txt],Characters[cap][[i]]]}],26];
```

將 `frequency` 指令執行在密文 `yih` 上如下所示:

```
In[39]:= frequency[yih]
Out[39]= {{A,73},{B,63},{C,32},{D,17},{E,40},{F,21},{G,48},{H,54},{I,45},{J,21},{K,56},{L,68},{M,51},{N,69},{O,49},
          {P,34},{Q,20},{R,48},{S,15},{T,59},{U,57},{V,43},{W,27},{X,48},{Y,51},{Z,40}}
```

所以密文字母出現的頻率爲

---

[7]阿碟爾英國作家, 寫過幾部小說與非小說。因翻譯 Georges Perec 的 A Void 而獲得 Scott Moncrieff 獎。

A	B	C	D	E	F	G	H	I	J	K	L	M
73	63	32	17	40	21	48	54	45	21	56	68	51
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
69	49	34	20	48	15	59	57	43	27	48	51	40

注意, 這些字母出現的頻率並沒有哪一個特別高。如上所討論的, 此乃因爲每一個字母在加密過程當中就已經被分散到好幾個字母上面去了。所以單純的頻率分析當然是無用武之地。然而從它的加密法則裡我們知道, 只要有辦法算出鑰匙的長度, 那麼就可以將那些經同一位移數得到的密文集中在一起。如此一來, 頻率分析又活龍活現了。因此之故, 第一要緊的是想辦法算出鑰匙的長度, 此乃破解維吉內爾密碼的關鍵所在。欲知如何進行, 請聽下節分解。

七. 破解維吉內爾密碼

所以破解之二部曲就是, 先算出鑰匙長度再求出鑰匙。我們先談如何進行, 再說明爲何如此這般就能求出鑰匙。最後再提供尋找鑰匙另類的方法。

先算出鑰匙的長度

將密文寫在一長紙條上, 然後影印一份得到一模一樣的另一長紙條。將此二長紙條平行並排, 但起頭的字母相隔幾個字母的位置。譬如, 相隔二個字母的位置時如下圖:

```
YIHXZCWKYNTKSMOYAELAZLUTUVUZYUGNLRAULBANHBYIEZOOVM
  YIHXZCWKYNTKSMOYAELAZLUTUVUZYUGNLRAULBANHBYIEZOOVM
                x x
```

若上下字母相同就在其下方作記號 x 並計數之。上圖中僅密文中的前 50 個字, 其中有 2 對相同的字母。如繼續數完, 那共有 14 對。在不同的位移數之下所得到的此種相同字母對的個數, 列表如下:

位移之個數:	1	2	3	4	5	6	7	8
相同字對數:	36	32	49	58	37	31	73	36

```
In[40]:=Table[{j,Count[Table[If[StringTake[yih,{i}]==StringTake[yih,{i+j}],t,{}],{i,1,1149-j}],t]}, {j,8}]
Out[40]={{1,36},{2,32},{3,49},{4,58},{5,37},{6,31},{7,73},{8,36}}
```

我們得知, 當位移數爲 7 時包含有最多的相同字母對。下面我們會說明這就是鑰匙長度的最佳猜測。此方法即使沒有電腦幫忙, 速度也很快, 而且通常得到的就是鑰匙長度。

尋找鑰匙第一法: 頻率分析, 各個擊破

現在假設我們已經知道鑰匙長度是 $m = 7$, 如上例。那麼位置在 $n,\ n+m,\ n+2m,\ \cdots$ 的字母 (此處 $n = 1, 2, 3, \ldots, m$), 其加密的位移數都是一樣的。如此一來, 就可以用頻率分析來幫助我們決定鑰匙第 n 個位置的位移數。首先我們將這些元素挑出來, 這可透過 MATHEMATICA 定義指令 `choose[txt,m,n]` 並執行之。接著執行指令 `frequency` 於其上, 算出每個字母出現的頻率; 最後再作頻率分析得到該分量的大小。

- `choose[txt,m,n]` 從字串 `txt` 中, 選取位置在 $n, n+m, n+2m, \ldots$ 的字母並形成一字串, 此處 m 爲一正整數而 $n = 1, 2, 3, \ldots, m$。若此字串 `txt` 之長度不是 m 的倍數, 則此函數對那些超過最後一個 m 之倍數的字不予理會。

注意, 在字串 `txt` 之長度不是 m 倍數的情況下, 我們可在 `txt` 的尾端加上一些空白使得其長度增加到剛好是 m 的倍數。如此一來, 就不會遺漏任何 `txt` 中的元素。這是爲什麼我們要在密文 `yih` 上加入六個空格的原因。上述指令其程式如下:

```
In[41]:= choose[txt_,m_,n_]:=StringJoin[Module[{z},z={};
                              Do[z=Insert[z,StringTake[txt,{m*(i-1)+n}],i],{i,1,Floor[StringLength[txt]/m]}];z]];
```

- $n = 1$ 時各字母出現的次數爲

A	B	C	D	E	F	G	H	I	J	K	L	M
15	9	0	0	6	5	11	14	1	0	14	6	16
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
4	6	5	0	2	0	11	2	1	4	23	5	5

```
In[42]:=frequency[choose[yih,7,1]]
Out[42]={{A,15},{B, 9},{C, 0},{D, 0},{E, 6},{F, 5},{G,11},{H,14},{I, 1},{J, 0},{K,14},{L, 6},{M,16},{N, 4},
         {O, 6},{P, 5},{Q, 0},{R, 2},{S, 0},{T,11},{U, 2},{V, 1},{W, 4},{X,23},{Y, 5},{Z, 5}}
```

出現最頻繁的是 X, 而且比次高的超出 7, 因此我們大可決定 X = e, 亦即鑰匙的第一個分量爲 $23 - 4 = 19 =$ t 。

- $n = 2$ 時各字母出現的次數爲

A	B	C	D	E	F	G	H	I	J	K	L	M
4	10	8	0	0	4	1	11	12	3	0	12	7
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
15	5	2	2	0	1	0	20	5	4	11	25	2

```
In[43]:= frequency[choose[yih,7,2]]
Out[43]= {{A, 4},{B,10},{C, 8},{D, 0},{E, 0},{F,4},{G, 1},{H,11},{I,12},{J, 3},{K, 0},{L,12},{M,7},{N,15},
          {O, 5},{P, 2},{Q, 2},{R, 0},{S, 1},{T,0},{U,20},{V, 5},{W, 4},{X,11},{Y,25},{Z, 2}}
```

出現最頻繁的是 U 與 Y, 因此 U = e 或是 Y = e。若 U = e, 則位移數爲 16, 故 E = o 才 0 次太少了, 而 N = x 有 15 次又太多了。因此 Y = e, 亦即鑰匙的第二個分量爲 $24 - 4 = 20 =$ u 。

- $n = 3, 4, 5, 6, 7$ 時依樣畫葫蘆, 得到鑰匙的第三個分量爲 $13 = \mathrm{n}$, 第四個分量爲 $6 = \mathrm{g}$, 第五個分量爲 $7 = \mathrm{h}$, 第六個分量爲 $0 = \mathrm{a}$, 第七個分量爲 $8 = \mathrm{i}$。

  經過上面審愼的猜測, 鑰匙可能是 tunghai, 因爲

$$\{19, 20, 13, 6, 7, 0, 8\} = \{t, u, n, g, h, a, i\}$$

如上面所看到的情況, 因爲樣本數變爲原來的七分之一, 所以最高頻率的字母 e 有可能會被其他那些次高的字母所取代。但還是可能高頻率的字母群會出現在高頻率的字母群中, 而低頻率的字母群會出現在低頻率的字母群中。如目前的情況, 這就足以讓我們決定鑰匙中的每一個分量。一旦這個具有潛力的鑰匙出爐, 那麼測試的最好方法就是使用此鑰匙來解密, 所得到的結果應該是很容易就可以判斷其正確與否。在我們的例子中, 所猜測的鑰匙是 $\{19, 20, 13, 6, 7, 0, 8\}$。所以解密鑰匙爲此向量之負值, 亦即 $-\{19, 20, 13, 6, 7, 0, 8\}$。我們可用此解密鑰匙執行前面定義過的指令 `vigdecrypt` 得到

```
In[44]:=the=vigdecrypt[yih,-{19, 20, 13, 6, 7, 0, 8}]
Out[44]=fourscoreandsevenyearsagoourfathersbroughtforthonthiscontinentanewnationconceivedinlibertyanddedicatedtoth\
        epropositionthatallmenarecreatedequalnowweareengagedinagreatcivilwartestingwhetherthatnationoranynationsoc\
        onceivedandsodedicatedcanlongendurewearemetonagreatbattlefieldofthatwarwehavecometodedicateaportionofthatf\
        ieldasafinalrestingplaceforthosewhoheregavetheirlivesthatthatnationmightliveitisaltogetherfittingandproper\
        thatweshoulddothisbutinalargersensewecannotdedicatewecannotconsecratewecannothallowthisgroundthebravemenli\
        vinganddeadwhostruggledherehaveconsecrateditfaraboveourpoorpowertoaddordetracttheworldwilllittlenotenorlon\
        grememberwhatwesayherebutitcanneverforgetwhattheydidhereitisforusthelivingrathertobededicatedheretotheunfi\
        nishedworkwhichtheywhofoughtherehavethusfarsonoblyadvanceditisratherforustobeherededicatedtothegreattaskre\
        mainingbeforeusthatfromthesehonoreddeadwetakeincreaseddevotiontothatcauseforwhichtheygavethelastfullmeasur\
        eofdevotionthatweherehighlyresolvethatthesedeadshallnothavediedinvainthatthisnationundergodshallhaveanewbi\
        rthoffreedomandthatgovernmentofthepeoplebythepeopleforthepeopleshallnotperishfromtheearth
```

再將字與字之間的空格還原, 即得原明文信息如下:

Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this. But in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow this ground. The brave men, living and dead who struggled here have consecrated it far above our poor power to add or detract. The world will little note nor long remember what we say here, but it can never forget what they did here. It is for us the living rather to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us–that from

these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion–that we here highly resolve that these dead shall not have died in vain, that this nation under God shall have a new birth of freedom, and that government of the people, by the people, for the people shall not perish from the earth.

哇!終於眞相大白, 這就是頂頂有名的蓋茨堡演說 (Gettysburg Address)。蓋茨堡會戰是美國內戰中最具決定性的會戰, 爲北軍奠定了勝利的基礎。爲了紀念兩軍的傷亡所換取的可貴平等人權, 美國總統林肯接受大衛·威爾斯 (David Wills) 的邀請, 於 1863 年 11 月 19 日, 在戰場遺址發表此演說:「$\cdots$ 期望美國在上帝的庇佑之下, 必得到自由的新生—使民有、民治、民享的政府, 能長存於世。」

為什麼如此這般的就找到鑰匙的長度?

現在我們先解釋爲什麼如此這般的就找到了鑰匙的長度。將英文字母出現的頻率按字母的順序排列形成一向量如下:

$$\mathbf{A}_0 = (.082,\ .015,\ .028,\ \ldots,\ .020,\ .001)$$

令 $\mathbf{A}_i$ 爲向量 $\mathbf{A}_0$ 位移 i 個位置後的結果, 如:

$$\mathbf{A}_2 = (.020,\ .001,\ .082,\ .015, \ldots)$$

向量 $\mathbf{A}_0$ 與自身的內積爲

$$\mathbf{A}_0 \cdot \mathbf{A}_0 = (.082)^2 + (.015)^2 + \cdots = .066$$

當然, $\mathbf{A}_i \cdot \mathbf{A}_i$ 也是等於 .066, 因爲我們得到的是相同的 26 個乘積的和, 只是排列不同而已。然而, 當 $i \neq j$ 時內積 $\mathbf{A}_i \cdot \mathbf{A}_j$ 的值大約在 .032 與 .045 之間, 遠比 $\mathbf{A}_i \cdot \mathbf{A}_i$ 還小:

$\lvert i-j\rvert$	0	1	2	3	4	5	6
$\mathbf{A}_i \cdot \mathbf{A}_j$	.066	.039	.032	.034	.044	.033	.036

$\lvert i-j\rvert$	7	8	9	10	11	12	13
$\mathbf{A}_i \cdot \mathbf{A}_j$	.039	.034	.034	.038	.045	.039	.042

這些內積只跟 $|i-j|$ 有關, 理由如下: 這些向量中的元素與向量 $\mathbf{A}_0$ 中的元素一樣, 但被位移過。在內積中, $\mathbf{A}_0$ 的第 i 個元素與第 j 個元素相乘, 第 $i+1$ 個元素與第 $j+1$ 個元素相乘, $\ldots$ 等等。所以每一個元素乘上位移 $j-i$ 個位置之後的那個元素。因此這個內積只跟 i 與 j

的差 $i-j$ 有關。然而, 互換 i 與 j 的角色並注意 $\mathbf{A}_i \cdot \mathbf{A}_j = \mathbf{A}_j \cdot \mathbf{A}_i$, 我們得到相同的內積, 所以這個內積只跟 $|i-j|$ 有關。在上表中, 我們僅需算到 $|i-j| = 13$ 即可。例如, $i-j = 15$ 對應於在某個方向位移 15 個位置, 也就是在反方向位移 11 個位置, 所以 $i-j = 11$ 給我們相同的內積。

內積 $\mathbf{A}_0 \cdot \mathbf{A}_0$ 之所以高於其他內積者在於, 此向量內之分量, 大的跟大的小的跟小的配對相乘。在其他的內積當中, 大的數隨機地跟小的數配對相乘。這有削減的效果。

假設明文字母的分佈情況非常接近英文字母的分佈, 如上面的向量 $\mathbf{A}_0$ 所顯示的。察看頂端密文長紙條上隨機的一個字母。此字母對應於英文字母上隨機的一個字母經位移 i 個位置者。而此字母對應的下端密文長紙條上的那個字母, 又對應於英文字母上隨機的一個字母經位移 j 個位置者。這兩個元素同時爲 A 的機率是 $\mathbf{A}_i$ 的第一個分量乘上 $\mathbf{A}_j$ 的第一個分量。此乃因爲 $\mathbf{A}_i$ 的第一個分量所記錄的是挪移一隨機字母 i 個位置之後得到之密文爲 A 的機率, 而 $\mathbf{A}_j$ 的第一個分量亦然。依此方式, 這兩個元素同時爲 B 的機率是 $\mathbf{A}_i$ 的第二個分量乘上 $\mathbf{A}_j$ 的第二個分量。因此之故, 這兩個元素爲同一個元素的機率爲 $\mathbf{A}_i \cdot \mathbf{A}_j$。當 $i \neq j$ 時, 這差不多是 0.038; 但如果 $i = j$ 則其內積差不多是 0.066。

我們是處於 $i = j$ 的情況, 這正好是當同一位置的上下兩個字母是經過相同數目之位移的情況, 那就是上端的紙條挪移的字母數目等於鑰匙的長度或鑰匙長度的倍數。因此在這種情況, 我們期待有更多的相同字母對。

上例中的密文, 其位移數爲 7, 我們有 1142 個比對及 73 對相同字母。根據上面的論證, 我們應該可以期待大約有 $1142 \times 0.066 = 75.37$ 相同字母對, 這很接近於實際的值。

尋找鑰匙第二法: 更上一層樓

利用上述的構想, 我們有第二個方法可以找到鑰匙。在第一個方法中的頻率分析各個擊破, 不可避免的需要作猜測。但目前要介紹的方法只需看數據不需作猜測, 可以節省許多時間。我們還是看上面的例子。

爲了要找出鑰匙的第一個分量, 如前, 先數算密文在第 1 、第 8 、第 15 $\cdots$ 位置上各字母出現的頻率並寫成一向量的形式 `v1` 如下:

```
In[45]:= v1=Table[frequency[choose[yih,7,1]][[i]][[2]],{i,1,26}]
Out[45]= {15,9,0,0,6,5,11,14,1,0,14,6,16,4,6,5,0,2,0,11,2,1,4,23,5,5}
```

如果我們除以字母的總數目 165, 可得一向量, 此向量應該會近似於上述的 $\mathbf{A}_i$ 之一, 此處的 i 就是由鑰匙的第一個分量所導致的位移數。如果我們計算 $\mathtt{v1} \cdot \mathbf{A}_i$, $0 \leq i \leq 25$, 則其最大值應該來自於正確的 i 值。這些內積如下:

```
In[46]:= alfreq={.082,.015,.028,.043,.127,.022,.020,.061,.070,.002,.008,.040,.024,.067,.075,.019,.001,.060,
                 .063,.091,.028,.010,.023,.001,.020,.001};
         s[a_,b_,n_]:= Sum[a[[i]]*b[[Mod[i-n-1,26]+1]],{i,26}];
         corr[a_]:=Table[s[a,alfreq,i],{i,0,25}];
In[49]:= corr[v1]
Out[49]= {6.337, 5.397, 5.553, 5.913, 6.92, 6.305, 7.819,6.612,7.019,6.165, 6.097, 5.408, 6.905, 5.644,5.478,
          6.779,5.51, 4.85, 6.383, 10.975, 6.376,5.479, 6.558, 7.485,5.285, 5.913}
```

最大的值是第 20 個元素, 即 10.914, 此數等於 $\mathrm{v1}\cdot\mathbf{A}_{19}$。因此我們猜測第一個位移數爲 19, 對應的鑰匙字爲 t。

依樣畫葫蘆, 找出鑰匙的第二個分量爲 20=u, 第三個分量爲 13=n, 第四個分量爲 6=g, 第五個分量爲 7=h, 第六個分量爲 0=a, 第七個分量爲 8=i。綜合簡化如下: 先定義 `vj = v[j]` 再算出每一個 `corr[v[j]]` 之最大元素的位置, 然後按序排列即得所要求的鑰匙。其程式如下:

```
In[50]:= v[j_]:=Table[frequency[choose[yih,7,j]][[i]][[2]],{i,1,26}];
         tunghai=Table[Position[corr[v[j]],Max[corr[v[j]]][[1]][[1]],{j,1,7}]-1
Out[51]={19, 20, 13, 6, 7, 0, 8}
```