TR-81-003

# 對美國國家標準局所訂「資料譯碼標準」之研究

# 研　究　報　告

計畫名稱：對美國國家標準局所訂「資料譯碼標準」之研究

計畫編號：NSC－69E－0404－01（03）

執行期限：69.年3月1日～70.年6月30日

計畫金額：NT$ 208,036.

執行單位：中央研究院資訊科學研究所

計畫主持人：黃　台　陽

專任助理：王　秋　鳳　（助教級）

研 究 生：關　聖　威

日　　期：中華民國　六十九　年　八　月

# 摘　　要

在本研究裡，我們對美國國家標準局於 1977 裡頒佈實施的「資料譯碼標準」，從多方面加以研究分析。我們考慮了竭盡式搜求、法則分析、以及統計分析。我們更進一步嘗試修改的內部結構，以觀察所引起的結果。結論是 DES 的結構極爲有效的保証了它的安全度。在未來數年內唯一可能威脅其安全的，是所使用的鍵的長度。

# 一、緒　言

「資料譯碼標準」(Data Encryption Standard), 簡稱 DES [1] ), 是美國國家標準局 (National Bureau of Standard, 簡稱 NBS ) 於1977年元月公佈，供美國聯邦政府各機構在以電子處理有安全顧慮的資料時，所能採用的一個共同標準。它的目的在確保經轉換後的資料，無論是在貯存時或是在網路中流通時，都不會被外人推算出其原始形態而偵知其原意，從而達到確保隱私與安全的目的。

該標準自公佈以來，廣受美國工業界與學術界的注意。在工業界方面的看法是：由於聯邦政府各機構的電子處理資料設備，在有安全顧慮時會採用 DES ，因此 DES將意味一很大的市場。這甚至將影響許多與政府機構有相當來往的民營企業，也會採用 DES 。因此許多著名的廠家，( 如 Intel, Motorola 及其它公司等 )，都投入生產具DES 規格的晶片 [2] 。

學術界方面對 DES的看法，則有點保留。一些在密碼學 [3] 方面有研究的專家學者，對 DES的安全效果，有兩個主要的顧慮：其一，DES 是由 IBM 公司設計出來，但在設計的過程裡，美國國家安全署 (National Security Agency, 簡稱 NSA ) 曾參預其事。因此 NSA是否 已掌握了破解 DES 的方法？第二個顧慮是： DES 在類別上是屬於一種古典型的密碼方法，變碼 (Encryption) 和解碼 (Decryption) 過程均使用相同的鍵 (Key) 但 DES 的鍵，僅用 56 bits ，是否足以應付竭盡式搜求 (Exhaustive search ) 的攻擊？

由於電子計算機在我國應用日廣，政府機關採用計算機來處理資料的也日趨普遍，故遲早我國將步先進國家的後塵，採行一套資料譯碼標準來保護機密文件。基於此一需要，我們有必要對美國所用的 DES 加以了解、分析、及研究，以便作爲他日我國制定自己的標準時的參考。本研究即根據這個目標，從幾方面來對 DES 作一探討。

研究進行的過程，在首先對 DES 的產生取得了解後，即對 DES 的內部功能展開分析，以求對其設計觀念有所領悟。其次的工作重點是就 DES 的安全性加以評估。所採取的角度有三，分別從竭盡搜求、法則分析、以及統計分析三方面來加以研究。我們的結論是到目前爲止雖無法找出 DES 的任何漏洞，但隨著時間的推移，未來技術的飛躍進展可能會使DES 變得脆

弱。因此，在設計我國未來所使用的資料譯碼標準時，有必要參酌 DES 加以改進。

在本研究報告的第二章裡，將討論 DES 的源起、變碼法則、以及其中的特點。第三章則從鍵竭盡搜求和法則分析兩方面來對 DES 的安全性作一分析。第四章是從統計分析方面，觀察原文和經 DES 處理過的密碼文之間的關係。第五章裡，則嘗試稍微修改 DES 的內部結構，看產生何等影響。第六章則總結全文。

## 二、資料譯碼標準介紹

資訊理論 (Information Theory) 的始創者C. E. Shannon 於 1949 年發表了一篇論文：保密系統的通信理論 (Communication theory of secrecy system) [4] 。這篇論文結合了密碼學與資訊理論於一爐。他建議由交換運用 P-box (permutation box) 及 S-box (substitution box) 產生强有力的交乘密碼系統 (product cipher system) ：因爲 P-box 是線性的，若單獨使用輸入爲n-數元 (bit) 的 P-box，只要 n 組相對的原文及密碼文就可破解；而單獨使用 S-box ，卻因線路裝置困難的理由，不能增加輸入的數元來增强安全性。因此 P-box 與 S-box 若分別單獨使用，安全效果都很脆弱。但一結合起來，所產生的交乘效果卻可非常驚人。

在二次世界大戰中，所作的密碼研究曾指出這種交乘系統，但未引起重視。至 1960 年才有一些人較積極著手這方面研究。主要原因是這種系統非常適合於計算機內運用。 IBM 公司就是這方面的先驅者之一。根據 H. Feistel 與 1973 年在 Scientific American 上所發表的一篇文章 [5], IBM 於 1970 年左右設計完成一種交換密碼系統，名爲 Lucifer. 它可以說是 DES 的前身。

美國標準局鑑於聯邦政府於電子處理有機密性資料時，需要將該等資料密碼化，可是經密碼轉換後的資料，又有流經許多不等聯邦機關的可能。爲了方便各機關在定密碼、解碼的過程，有一標準可循，於是在 1973 年宣布，希望各界提供密碼方法。可是只有 IBM 憑藉其在這方面的研究結果，提出幾種方法，其中之一，就是 DES.

NBS 於 1975 年元月將 DES 的全部法則公開，希望各界對其安全性加以批評。其間雖有史丹福大學的 Diffie 和 Hellmen 等對 DES 有所質疑 [6] , [7] . 但 NBS 認爲 DES 的安全仍然足以信賴。於是在 1977 年元月正宣布 DES 爲聯邦政府所使用的標準。

以下，我們對 DES 的法則結構加以檢討、說明。

### 2-1. DES 的結構

DES 屬於區段譯碼 (block cipher) 。它有 64-bits 的輸入，64-bits 的輸出。所使用的鍵 (key) 雖亦有 64-bits，但其中 8-bits 爲偵錯用，故鍵僅有 56-bits 具實體意義。換言之，最後僅須試 $2^{56}$ 次，一定可

從密碼文推算回原文，破解密碼。因此 DES是屬於計算上安全性密碼 (computationally secure cipher) 。它的安全性基於兩個條件：

(1)流程本身的複雜度

(2)足夠多道的 SP-boxes.

DES 的譯碼過程大致可如此描述：假設所使用的鍵爲 K ，則有一套子鍵產生法則，依次產生 $K_1, K_2, \ldots, K_{16}$ ；假設原文爲 P ，則

(1) P 首先經一重新排列，輸出則分爲兩部份，$L_0$ 與 $R_0$ ，各有 32-bits;

(2)由 $L_0$ 與 $R_0$ 產生 $L_1$ 與 $R_1$ ，法則如下，

$$L_1=R_0$$

$$R_1=L_0 \oplus f(R_0, K_1)$$

這裡的 f 爲一複雜的代換，排列函數；

(3)再由 $(L_1, R_1)$ 依次產生 $(L_2, R_2)$, $(L_3, R_3)$, ..., $(L_{16}, R_{16})$, 其法則同上；

(4) $(L_{16}, R_{16})$ 再經從新排列，結果即爲密碼文輸出。

至於詳細的內部結構， S-boxes 與 P-boxes 的安排，以及子鍵 $K_i$ 的產生法則等，統列於附錄一裡，以供詳細參考。

DES 的解碼，與譯碼過程相同，只是方向相反：譯碼文經過重新排列後，得回 $(L_{16}, R_{16})$ 因

$$R_{i-1}= L_i,$$

$$L_{i-1}= R_i \oplus f(L_i, R_i)$$

所以 $(L_{15}, R_{15})$ 分別可求得。同理推到 $(L_0, R_0)$, 再經排列後就得到原文。

由這簡單的描述裡，我們可看出，DES 的保密性是完全建立於函數 $f$ 的複雜度上面。經過檢查， $f$ 並不是個 1-1 (one-to-one) 的函數，但是 $(L_i, R_i)$ 與 $(L_{i+1}, R_{i+1})$ 的巧妙安排卻使得 DES 仍是 1-1.

NBS 在要求各界提供譯碼方法時，曾有幾個要求：

(1)要能提供高度的資料保護，以防止未經授權的洩密及查不出的竄改資料；

(2)要簡單易懂，但仍有相當複雜度，使得要破解其系統，必須付出極大代價，導致得不償失；

(3)保密的方法必須基於譯碼所用的鍵，而非法則裡有任何秘密不讓人知之處；

(4)製造成本要經濟，操作要簡便；

(5)適合在不同的應用場合使用；

(6)能讓所有有意的使用者容易取得，並以合理的價格供應。

DES的結構，使得它符合前三項要求；另外由於IBM 已宣告放棄其對DES 專利權的要求，使得後三項要求亦告滿足。

在我國制定同類的國家標準時，NBS 所提的這六項要求，也應是我們所要講求的。

## 2-2. 函數 $f$ 的分析

前面已指出，函數 $f$ 是構成 DES 保密性之所在。它的詳細功能，可參閱附錄一裡的第二圖。在此，我們大致敘述其工作情形，並加以分析。

在每一道裡（總共16道），右邊部份 $R_i$ 的 32-bits，首先被擴張爲 48-bits，也就是有16個 bits 各出現 2 次；這 48-bits 與 $K_i$（也是 48-bits）相加，modulo 2，所得被分成 8 部份，分別爲 8 個代換裝置的輸入；這 8 個代換裝置，標爲 $S_1, S_2, \ldots, S_8$，均由 4 個小的S-boxes 所組成；各個 $S_i$ 輸入部份的 6-bits 有 2-bits 是用來決定該選那一個 S-box，其餘 4-bits 才眞正是該S-bits的輸入，而得出4-bits 的輸出。因此，整個函數 $f$ 的輸出又回到 32-bits 恰與其輸入同。

函數 $f$ 的這種設計，有一相當突出之處。那就是小的S-boxes 的選擇，並非完全取決於鍵和其子鍵，資料的數元亦扮演了一份角色。這之所以能達成，是因資料的 32-bits 先經擴張爲 48-bits 故資料的數元雖被用來選取 S-boxes，卻不會因之導致資訊漏失。

函數 $f$ 的另外特點，還有(1).每個 S-box 的 4-bits 輸入，都與 $K_i$ 有關，這使得最初的原文即使爲已知，但在 $K_i$ 加入後，也變成未知，導致推測的困難；(2).選取各 $S_i$ 裡該用的16-bits，需有 2-bits 來決定，這使得

每一道裡，應有16-bits 來決定；總共有16道，原本應有256-bits 的鍵，但現今僅用56-bits，想像中會令各道之間S-boxes 的選擇，有某種關連存在。但因$S_i$裡選S-boxes 的 2-bits 融會了$K_i$和$R_i$使得分析各道之間關連性的工作，極其複雜。

每個$S_i$裡，有四個進出均爲 4-bits 的S-boxes 。這些 S-boxes ，是替換工作眞正發生之處。它們的設計，是基於什麼？有無特殊之處？

在 1979 年發表於IEEE trans. on Computers 的一篇論文裡，Kam 和 Davida [8] 兩位教授提出了一個衡量S-box 的複雜度的觀念。他們認爲一個好的S-box ，任何輸出的 bit 都必須是所有輸入bits 的函數。他們稱這種關係爲 total dependency，而達到 total dependency 的 S-box 爲 Complete S-box. 他們並証明了當輸出入至少各有 3-bits 時，可以找到許多 Complete S-boxes.

DES 裡的32個 S-boxes，經分析後我們發現每一個都是 Complete S-box 。這顯示了這些S-box 的產生，不太可能是亂數般弄出來，應有一套規則在後支配。可惜的是，我們仍無法掌握此一規則。

## 三、DES 分析：鍵竭盡搜求法及法則分析法

DES 密碼系統，若表示爲可逆函數集合 $\{S_k\}_{k\in K}$，

$$S_k : P \to C$$

則 $S_k$ 爲公開的 DES 法則，P 爲未經處理的原文，C 爲密碼文，參數 k 爲鍵；鍵空間 (key space) K 裡所採用的元素 k 確定後，函數 $S_k$ 即告確定。因此 DES 的機密完全取決於鍵是否能保持秘密，而非依靠系統本身。

以下對 DES 的分析，所考慮的安全標準爲：即使知道幾對原文及其對應的密碼文，也就是所謂的已知原文攻擊法 (known plaintext attack)，仍應保証 DES 的安全。

### 3-1. 鍵竭盡搜求法

這種攻擊，乃是在對 DES 的內部結構有充份信心的情況下，所需防範的唯一攻擊。它是著眼點，乃是嘗試所有可能的鍵。因此，一定能找出一個鍵，使得已知的原文、密碼文相符，從而破解。

由於這種攻擊法是做苦工式的，使得對抗它的唯一方法就是加大鍵空間，從而令能被取用的鍵的數目極大。可是 DES 加大鍵空間的方法只有增長鍵的長度，而較長的鍵往往又會拖慢譯碼的速度。再者，計算的速度隨著技術的精進而日益加快，因而縮短了測試一個鍵所需的時間；而硬體技術的推進又導至成本價格下降，使得並行處理的可能性大增。這一切都將影響到對鍵長度的決定。

目前 DES 鍵的長度是 56-bits，能否抵擋住鍵竭盡搜求法的攻擊？

史丹福大學的一篇研究報告裡 [9]，對此有一分析。他們認爲一個可能攻擊路徑是這樣的：

時間估計

假設每個鍵的尋找時間（在設定的鍵 F 原文經譯碼，看是否等於秘碼文）= $10^{-6}$ 秒，

則測試所有可能的鍵將需時 $=2^{56}\times 10^{-6}$ 秒 $= 10^{6}$ 天。

故若能設計 $10^6$ 個微處理機同時並行處理，則最多需時一天，平均起來僅僅半天，就可求得正確的鍵。

## 價格估計

假設每個微處理機值 US$20 ，（大量製造）分五年折舊，則每天所付的價錢 ($20/ 微處理機 × $10^6$ 微處理機)÷(365×5 天) =$10,000/ 天。換句話，平均起來只要花大約 US $5,000 元的費用，就能搜求出所經使用的鍵了！

這樣看來，DES 的危險性是岌岌可危。因爲敵對國家若基於政治上的理由，要製造 $10^6$ 個微處理機並行的破碼機，每天付 US$5,000 元，並非不可能的。雖然DES 處理的資料，將不含國防和外交裡的秘密，但其他方面資料若被破解了，仍足以對國家社會構成極大破壞的！

史丹福大學提出的這種分析警告，馬上得到 NBS 的幾點反駁：

(1)在並行操作單元裡，設計和控制的價格的成長，將大於平行操作的線性關係；
(2)在一百萬個微處理機的大系統裡，任何兩次機器故障的平均間隔時間 (mean time between failure) 將遠小於一天。因此系統將無法完成一天無故障的搜尋；
(3)不可能建造出一個只花 US$20 ，卻能在 $10^{-6}$ 秒內即完成尋找一個鍵的微處理機；
(4)把一百萬個微處理機放在一起，將需要 6000 個 6 呎高的機板(rack)。這將需要極大的空間，而成爲極顯著目標，無法逃過偵查；
(5)一百萬個微處理機，在同時高速運轉情況下，要消耗太多能量；
(6)鍵竭盡搜求法的密碼分析，並不能保証能尋出所使用的鍵。因爲可能會有不只一組鍵，卻能導致相同的結果。
(7)在時常更換鍵的政策下，鍵竭盡搜求法，將會無效。

NBS 的這些反駁，似乎對 DES 的安全提供了有利的解釋。但這仍不能折服史丹福大學的專家。他們針對 NBS 的反駁，再提出了說明：

(1)所提的並行操作系統，其並行操作單元間沒有連繫。其間重覆的結構，少量的 I/O，自動診錯，均簡化了設計，所以價格大部份用在半導

體的記憶體上，設計及控制方面的花費很小；

(2)一百萬個微處理機建立在64個機板上，一個機板有 16,000 個微處理機；每一個機板能自行診錯，並把故障的訊息傳送至計算機的控制單元，以更換好的機板；

(3)因爲 I/O 的時間非常少，竭盡搜求專用的微處理機，在設計上，可使中央處理單元 (CPU) 在少量輸出入的情況下，達到最高速的地步。因爲微處理機只在最初有輸入，以後鍵更新時，均由微處理機裡直接進行。而唯一的輸出只在找到正確的鍵，抑或機器故障時才發生。因此，鍵搜求時間可如所言。

再者，少量的輸入同時使得輸出入可順序進行，減少脚 (pin) ，可減少價錢；

(4)少量的脚使得機板裡的密度增大，因此面積縮小；

(5)專爲此設計的微處理機，消耗的能量不會很大；

(6)即使有不只一組鍵，但在已知多對原文一密碼文的情況下，找出唯一正確解仍不困難；

(7)由於分配鍵 (Key distribution) 在實際上有困難，因此經常更換鍵的工作不可能執行。

NBS 與史丹福專家們的爭論各執一詞 [10]，重點在技術上的難易與價格的高低（如果破碼所需的花費大於由此資訊所得的利益，則沒有必要去破碼）。實際上目前還沒有這樣一個破碼機出現。如果沒有重大的電子技術突破，則 CPU速度只能維持目前水準。考慮到要製造一百萬個同樣規格的微處理機，並克服系統設計上的某些問題，造成了 DES 目前仍屬安全的依據，然而技術永遠是向前走的，DES 的安全，在這破碼系統沒出現前，固屬安全，但這段時間能維持多久呢？

設若終有一天，DES 鍵的長度 (56-bits) 不足以應付鍵竭盡搜求法了（即使NBS 也同意這一天可能很快就會到來），那應如何修改DES ，使得原有的設備，原有已完成的譯碼工作，都不致變成無效？

這時若全部更換新的譯碼標準，將導致巨額的支出與浪費。故一般共同的看法是，將已經過 56-bits 的鍵的密碼文，再經過另一組鍵，按 DES 法則再算一遍。這將使得硬體方面成本不增加，只是譯碼的時間支出增加一倍。可是由這多出的 56-bits 的鍵，將能確保安全。因我們無法預見任何能在有效時間內破解 112-bits 的鍵的破碼機。

鍵竭盡搜求法的討論，到此告一段落。接下來我們討論法則分析法。

## 3-2. 法則分析法

DES 如前所述，是有 64-bits 輸入，64-bits 輸出，由鍵所控制的可逆函數。它的結構是基於强有力的 S-P box 的組合，連續16道才產生所要的密碼文。由於 DES 屬於計算上安全系統，吾人不能証明它法則上的安全性。因此本分析的重點在找尋 DES 是否具有容易破解的代數結構，及 DES 本身所具有的特點。

DES 中的 P-box ，擴張函數（函數 $f$ 裡的 expansion function），及 XOR 的運算均爲綫性函數，唯有各個 $S_i$ 可能是非線性的。由於已知線性函數在獲有 n+1 對 (65 對）原文及密碼文的情況下可被破解，故若各個 $S_i$ 也是線性的，則整個 DES 極可能是線性的，可望在知道一些原文及密碼文後被破解。因此 S-boxes 實爲 DES 安全之所繫，秘密之所在！以下我們要對 S-boxes 的性質深入探討：

(1) S-boxes 沒有 Affine 性質

這個性質是由以下各定理得出。

引理 1. N 度空間裡，令 P 映至 C 的函數 $f_k$，若有 Affine 關係，則若 $C=f_k(P)$，C 恒可寫成 $C=AP+b$ 的形式。
此處 P 及 C 爲 $n\times 1$ 的行矩陣，A 爲 $n\times n$ 矩陣，b 爲常數 $n\times 1$ 的行矩陣。

根據這個引理，我們可得出下面定理：

定理 1. 若各個 S-box 爲 Affine ，則 DES 爲 Affine.

証明. DES 除了 S-boxes 外，任何一個函數：P-box ，擴張函數，及 XOR 運算都是 Affine 。當原文 P 經過起始重組排列 IP 時，得出

$$P_0 = \begin{pmatrix} L_0 \\ R_0 \end{pmatrix}, \qquad P_0 = A_1 P. \qquad (1)$$

令 P 經過第一道的輸出爲 $P_1 = \begin{pmatrix} L_1 \\ R_1 \end{pmatrix}$. 因

$$\begin{pmatrix} R_0 \\ L_0 \end{pmatrix} = A_2 \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} \tag{2}$$

而在函數 $f$ 裡，$R_0$ 經過擴張函數 E 得出 $R_0'$，

$$R_0' = A_3 R_0 , \tag{3}$$

並且與 $K_1$ XOR 相加，

$$R_0' + K_1 = A_3 R_0 + K_1. \tag{4}$$

現假設 S-boxes 爲 Affine，則 S-boxes 的輸入 $R_0' + K_1$ 與輸出 O 之間的關係爲

$$O = A_4 (R_0' + K_1) + B_1. \tag{5}$$

O 再經過一個 P-box，得到 $f(R_0, K_1)$， 故

$$\begin{aligned} f(R_0, K_1) &= A_5 O + B_2 \\ &= A_5 (A_4 (R_0' + K_1) + B_1) + B_2 \\ &= A_5 (A_4 (A_3 R_0 + K_1) + B_1) + B_2 \\ &= A_5 A_4 A_3 R_0 + A_5 A_4 K_1 + A_5 B_1 + B_2. \end{aligned}$$

定 $A_6 = A_5 A_4 A_3$, $B_{3k_1} = A_5 A_4 K_1 + A_5 B_1 + B_2$, 則

$$f(R_0, K_1) = A_6 R_0 + B_{3k_1}. \tag{6}$$

然而由於

$$P_1 = \begin{pmatrix} L_1 \\ R_1 \end{pmatrix} = \begin{pmatrix} R_0 \\ L_0 + f\ (R_0, K_1) \end{pmatrix}$$

$$\equiv \begin{pmatrix} R_0 \\ L_0 \end{pmatrix} + \begin{pmatrix} 0 \\ f\ (R_0, K_1) \end{pmatrix}$$

代入(2),(6) 式，

$$P_1 = A_2 \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + \begin{pmatrix} 0 \\ f\ (R_0, K_1) \end{pmatrix}$$

$$= A_2 \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A_6 \end{pmatrix} \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + B_{3K_1}.$$

令 $\begin{pmatrix} 0 & 0 \\ 0 & A_6 \end{pmatrix} = A_7$，則

$$P_1 = A_2 \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + A_7 \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + B_{3K_1}$$

$$\equiv A_8 \cdot \begin{pmatrix} L_0 \\ R_0 \end{pmatrix} + B_{3K_1}$$

$$= A_8\ P_0 + B_{3K_1}$$

$$= A_8\ A_1\ P + B_{3K_1}$$

或

$$P_1 = A_9\ P + B_{3K_1}.$$

由此顯出經過第一道的輸出 $P_1$，與 P 之間依然存有Affine 的關係。而由於 $P_1$又當作第二道的輸入，依此類推，第16道的輸出$P_{16}$，與 之間也將有 Affine 的關係。

$$P_{16} = A_{10}\ P + B_{4K}$$

再經過重組排列 $IP^{-1}$，得到輸出

$$C = A_{11}\ P_{16}$$

$$= A_{11}\ A_{10}\ P + A_{11}\ B_{4K}.$$

令 $A_{11}\ A_{10} = A_K$, $A_{11}\ B_{4K} = B_K$, 則

$$C = A_K\ P + B_K.$$

故定理得証。

定理 2. 若 DES 在某一鍵下具有 Affine 關係，且已知的65對原文及密碼文，其原文 $\{ P_0, P_1, \cdots, P_{64} \}$ 構成的 64×64 矩陣

$$\left| P_1 - P_0,\ P_2 - P_1, \cdots, P_{65} - P_{64} \right|$$

是個可逆矩陣，則該鍵可被破解。

証明： 若 DES 對鍵 K 爲Affine，則每一對原文 P 及其密碼文 C 均有 $C=A_K\ P+b_K$ 的關係存在。考慮已知的65對

$$(P_0, C_0),\ (P_1, C_1), \cdots, (P_{64}, C_{64}),$$

$$P_i' = P_i - P_0$$

$$i = 1, 1, \ldots, 64.$$

$$C_i' = C_i - C_0$$

則

$$C_i' = A_K P_i' .$$

設

$$C = [\, C_1', C_2', \ldots, C_{64}' \,]$$

$$P = [\, P_1', P_2', \ldots, P_{64}' \,] .$$

可得 $C=A_K P$ ，這裡的 $C, A_K$， 及 $P$ 都是 $64\times64$ 矩陣。
故可求出

$$A_K = CP^{-1}$$

$$b_K = C_0 - A_K P_0 .$$

導致秘碼破解。

DES 的設計，雖存有這個隱憂，但所幸下一定理保証了其安全。

定理 3. 沒有一個 $S_i$ 爲 Affine.

証明. 對於每一個 $S_i$ 有 6-bits 輸入，4-bits 輸出。故若用 24 個輸入值固定 $6\times4$ 矩陣 A,

$$C = AP + B$$

而用其餘 40 個輸入值代入上式，看是否能滿足。經過測試，8 個 $S_i$ 均沒有個輸入值，同時滿足上式，故 8 個 $S_i$ 均不爲 Affine 。得証。

由上面這些定理，我們得知 DES 的密碼功能，並不是簡單的 Affine mapping 所能概括。下面，我們將討論另一個特性。

(2) 原文與密碼文的數元之間具有强列的交互相依性 (Intersymbol dependence).

由於 DES 每一道的輸出 $(L_i, R_i)$ 與輸入 $(L_{i-1}, R_{i-1})$ 之間的關係爲

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} + f\,(R_{i-1}, K_i).$$

故若令 $G_{ij}$ 爲第 j 道的 bits 與第 i 道 bits 間的關係矩陣，若〃無關〃則元素值爲 0，〃有關〃則值爲 1，我們可得到下列關係式：

a. 因 $L_i(x)=R_{i-1}(x)$，故 $L_i$ 的 X-bit 只與 $R_{i-1}$ 的 X-bit 有關係；故

$G_{i,(i-1)}^{(L,R)}$ 僅在對角線上有"1" 值。

b. 因 $L_i(x)=R_{i-1}(x)$，故

$G_{i,(i-1)}^{(L,L)}$ 爲零矩陣

c. 因 $R_i(x)=L_{i-1}(x)+f(R_{i-1},K_i)$，故

$G_{i,(i-1)}^{(R,L)}$ 在對角線上有"1" 值，其餘爲"0".

d. $R_{i-1}$ 經過 E 函數與 $S_i$ 之後，第 1,2,3,4 數元與 $R_{i-1}$ 第32,1,2,3,4,5 數元有關；依此類推，最後第 29,30,31,32 數元與 $R_{i-1}$ 第 28,29,30,31,32,1 各數元有關。

由以上 a,b,c,d 的觀察，我們更可得出 $G_{(i+1),(i-1)}$ 以 $G_{j,(i-1)}$ 表示的式子：

a. 因 $L_{j+1}(x)= R_j(x)$,

$$G_{(j+1),(i-1)}^{(L,L)} = G_{j,(i-1)}^{(R,L)}$$

b. 因 $L_{j+1}(x) = R_j(x)$ ,

$$G^{(L,,R)}_{(j+1),(i-1)} = G^{(R,R)}_{j,(i-1)} ,$$

c. $$G^{(R,L)}_{(j+1),(i-1)} = G^{(L,R)}_{(j+1),(i-1)} = G^{(R,R)}_{j,(i-1)} ,$$

d. $G^{(R,L)}_{(j+1),(i-1)}$ 的 X 列由 $G^{(R,R)}_{j,(i-1)}$ 的 $m_1(x),...,m_6(x)$ 列值相加，求得 $m_1(x),....,m_6(x)$ 爲 $G^{(R,R)}_{j,(i-1)}$ 中第 x 數元的 "1" 值所在位置。

後面有5個圖裡，顯示出 $G_{ij}$ 的各部份及全部圖，同時也對以上的程式加以說明。根據這些，我們還列出表一。它顯示出到了第五道，其輸出的任一數元與原文的任一數元均有關係，也就是達到 total dependency 的要求了。這使得在第16道後，DES 的輸出幾乎達到完全雜亂的有利情況。

3. 補數性質：$S_K(P) = \overline{S_{\overline{K}}(\overline{P})}$

證明： 首先，我們可寫出

$$f(R_i, K_{i+1}) = f'(ER_i \oplus K_{i+1})$$
$$= f'(\overline{ER_i} \oplus \overline{K_{i+1}})$$
$$= f'(\overline{R_i,} \quad \overline{K_{i+1}}).$$

設 $\overline{P}$ 爲原文 P 的補數。當 P 及 $\overline{P}$ 分別經起始組合排列 IP 後，得 $(L_0, R_0)$ 及 $(L_0', R_0')$ 它們的關係爲

$$L_0' = \overline{L}_0 , \quad R_0' = \overline{R}_0 .$$

因 $L_1 = R_0$,導致

$$L_1' = R_0' = \overline{R}_0 ,$$

也就是

$$\overline{L}_1 = L_1' .$$

再者，

$$P_1 = L_0 \oplus f ( R_0, K_1 ) ,$$

$$P_1' = L_0' \oplus f ( R_0', \overline{K}_1 )$$

$$= \overline{L_0} \oplus f ( \overline{R}_0, \overline{K}_1 )$$

$$= \overline{L_0} \oplus f ( R_0, K_1 ) .$$

故得

$$\overline{R_1} = R_1' .$$

由此觀之，P 及 $\overline{P}$ 經過第一道的結果仍具有互補性質。同理，經第 16 道後可得出

$$\overline{L_{16}} = L_{16}' , \quad \overline{R_{16}} = R_{16}' .$$

再經 $IP^{-1}$，得出

$$\overline{C} = C' .$$

故

$$S_K (P) = \overline{S_{\overline{K}} (\overline{P})} .$$

得証。

這一項性質，顯示 DES具有一種對稱性。卻對它造成輕微的不利影響；我們在下面討論。

補數性質的影響：

如果密碼分析者，知道兩對原文及密碼文：$(P_1, C_1), (P_2, C_2)$ ， 而且$P_1=\overline{P}_2$. 則尋找鍵的工作，可減少一半。

証明： 設

$$H = \{K | K \text{ 爲鍵空間元素，且最後一數元爲 } 0\}$$

則對某一個 $K \in H$, 求出

$$C = S_K (P_1) .$$

如果 $C \neq C_1$, 則 K 不合所求。
如果 $C \neq \overline{C}_2$, 則因

$$\overline{C} = \overline{S_K (\overline{P}_1)}$$

$$= S_{\overline{K}} (\overline{P}_1)$$

$$= S_{\overline{K}} (P_2)$$

我們可推論出 $\overline{K}$ 亦不合所求。
故只用一個鍵譯一次碼，卻可同時測試兩個鍵。這將導致鍵找尋工作減半。換句話說，鍵竭盡搜求法最多也只須試 $2^{55}$ 次了。

3-3. 結論

在本章裡，我們從鍵竭盡搜求法和法則分析法兩個方向，來對DES 的安全性和結構加以研究、測試。我們的結論是，除了補數性質是個小的缺限外，DES 到目前爲止並未顯露出任何可供破碼者利用的弱點。如果我們把 DES看作是一個 64-bits 輸入，64-bits 輸出的大 S-box ，則這S-box 是 complete. 且其複雜程度，目前仍無法深入分析。

如果要用蠻力一個一個的去測試所使用的鍵，我們也認爲這在目前是無法做到的事，但由於技術的日新月異，卻極有可能在未來歲月裡出現如此的破碼機。到時候，DES 就必須加以修正。一個可能的修正方法是，把密碼輸出再度輸入 DES，並另用一組 56-bits 的鍵。這樣，鍵的長度就會增至 112-bits，將遠超過彼時技術之所能。

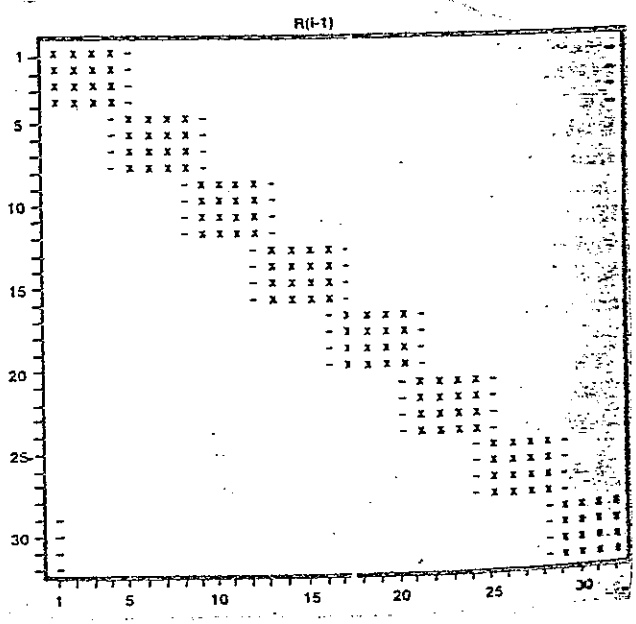



圖 3-1. R ( i-1 ) 與未經 P-box 的 R (i) 之間的關係

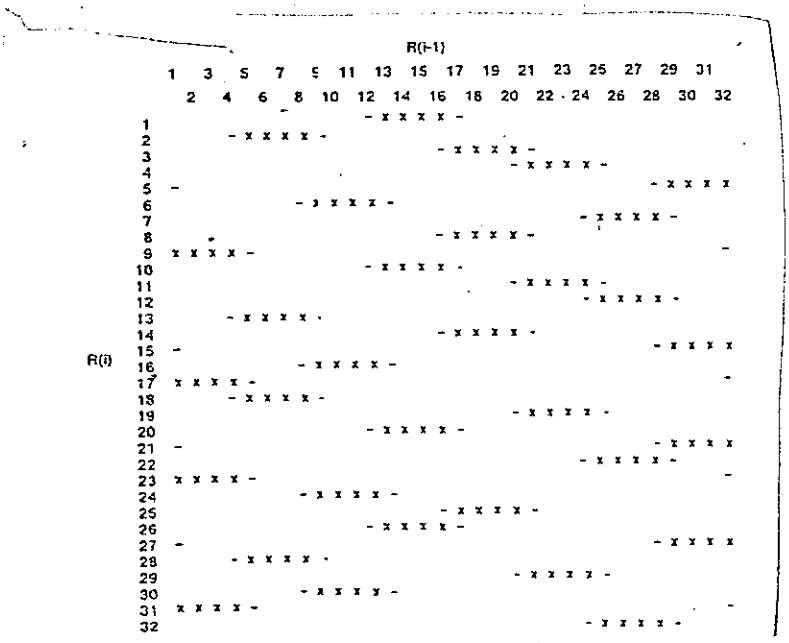



圖 3-2. 矩陣 $G_{i,\,i-1}^{(R,R)}$ 裡 R (i) 與 R (i-1) 的關係

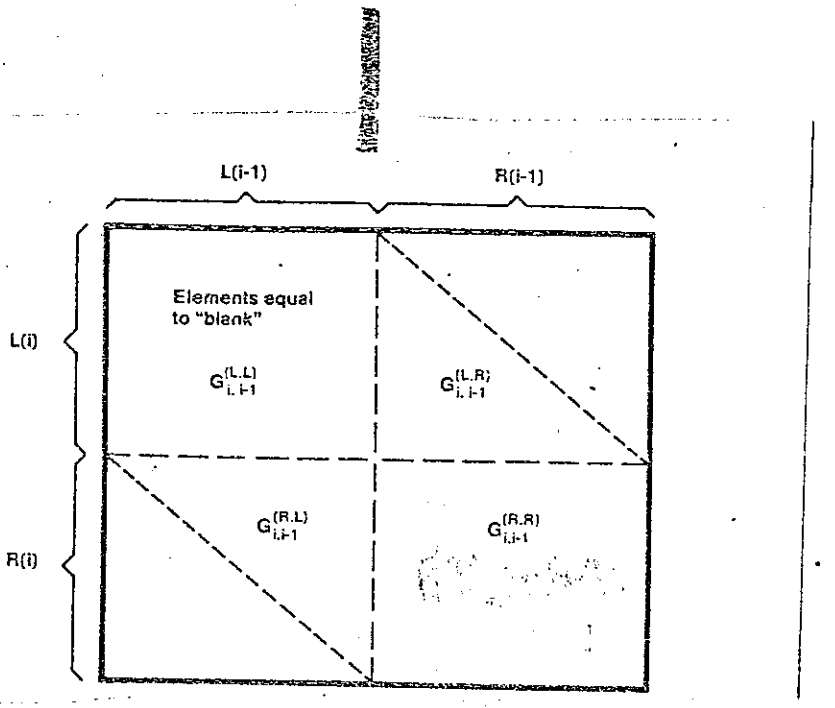



圖 3-3. 矩陣 $G_{i,i-1}$

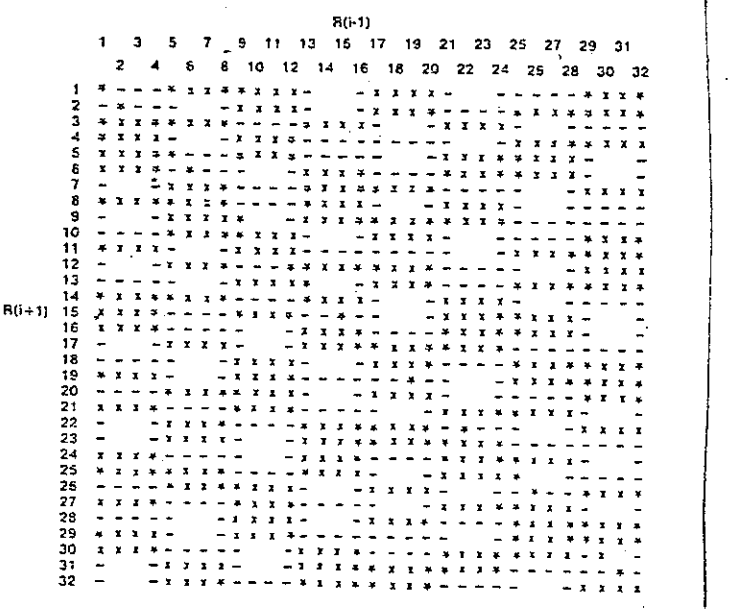



圖 3-4. 由 X (j) 對 X (i-1) 的關係得出 X (j+1)對 X (i-1) 的關係

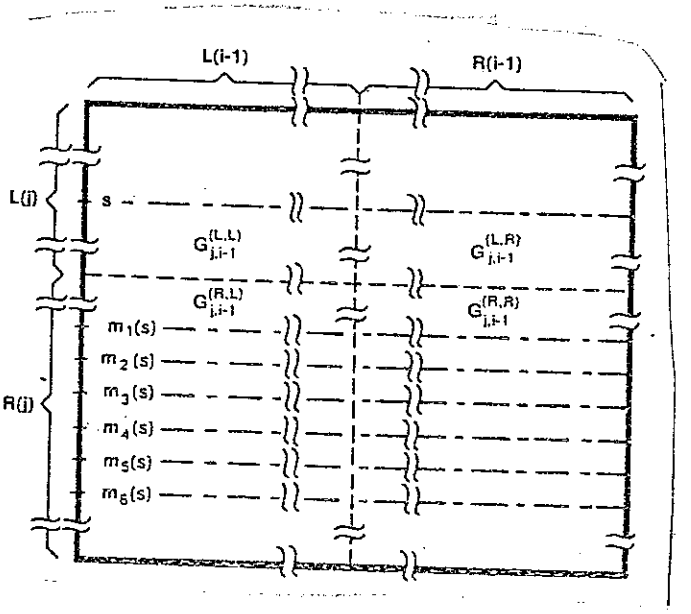



圖 3-5. 矩陣 $G^{(R,R)}_{i+1,i-1}$ 裡 R ( i+1 ) 與 R ( i-1 ) 的關係

表一・第 J 道的數元與原文數元間的關係百分比

| Round $j$ | Output/Input Relation | | | | |
|---|---|---|---|---|---|
| | $L(j)$ vs $L(0)$ | $L(j)$ vs $R(0)$ | $R(j)$ vs $L(0)$ | $R(j)$ vs $R(0)$ | $X(j)$ vs $X(0)$ |
| 1 | 0.00 | 3.13 | 3.13 | 18.75 | 6.26 |
| 2 | 3.13 | 18.75 | 18.75 | 88.18 | 32.20 |
| 3 | 18.75 | 88.18 | 88.18 | 100.00 | 73.78 |
| 4 | 88.18 | 100.00 | 100.00 | 100.00 | 97.05 |
| 5 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |

*NOTE:* Table entries express the degree of intersymbol dependence, i.e., the percentage of non-blank elements in the appropriate relation.

## 四、統計分析

由於從法則分析方面無法找出DES 的漏洞，因此我們換個角度，從統計方面來看。嘗試尋找是否有些bits 的位置，對變化不甚敏感；或是否有些鍵，會導致原文與密碼文之間的相依性仍非常密切等。當然，由於原文有 $2^{64}$ 種不同輸入，鍵有 $2^{56}$ 種不同輸入，我們無法考慮所有可能的組合。但由於我們統計分析的途徑，使得分析的結果具有相當的可信度。

分析的途徑是這樣的：用軟體程式模擬DES 系統，然後藉著輸入值方面些微的改變，來比較輸出方面的改變。由許多組如此的測試結果，可得出統計數字，從而得出判斷依據。（程式附於附錄二）

分析用的測試值：每種測試均採用線性回授平移器 (Linear Feedback Shift Register): $C(Z)=Z^{31}+Z^{34}+Z^{27}+Z^2+Z+1$ 所產生的近似亂數 (pseudo-random number) 爲其實驗值；每種測試取 1000 組或 2000 組數據，以供系統分析。

首先定義幾個向量：

1. 向量 $B_1$: 在N 對長度均爲 K-bits 的向量(A,B) 中，就每對 (A,B) 對應的位元加以比較，$B_1(i)$ 表示在第i-bit 上，A(i)=B(i) 的對數，i=1,2,...,K.
2. 向量 $B_2$: 在N 對長度均爲 K-bits 的向量(A,B) 中，比較A(i) 與 B(i+32 mod K )，而 $B_2(i)$ 表示有多少對相同，i=1,2,...,K.
3. 向量 $B_3$: 在N 對長度均爲 K-bits 的向量(A,B) 中，若 i 爲奇數，比較 A(i) 與 B(i+1)；若 i 爲偶數，則比較 A(i) 與 B(i-1). $B_3(i)$ 表示相同的對數， i=1,2,...,K.
4. 向量 $D_1$: 在N 對長度均爲 K-bits 的向量(A,B) 中，比較 A 與 B 相對的數元，然後統計有多少個數元相同。$B_1(i)$ 則表示在 N 對 (A,B) 裡，有多少對是有 i 個數元相同，i=1,2,...,K.
5. 向量 $D_2$: 在N 對長度爲 K-bits 的向量(A,B) 中，比較 A(i) 與 B(i+32 mod K) 然後統計有多少個數元相同。$B_2(i)$ 則表示在N 對如此的比較裡，有多少對是有 i 個數元相同， i=1,2,...,K.
6. 向量 $D_3$: 在N 對長度爲 K-bits 的向量(A,B) 裡，若 i 爲奇數，比較A(i) 與 B(i+1) 若 i 爲偶數，比較A(i) 與 B(i-1) 然後統計有多

少個數元相同，$B_3(i)$ 則表示在 N對如此的比較後，有多少對會導致 i個數元相同，i=1,2,...,K.

這六個向量，將就是我們測試結果。向量 A,B則表示在不同的輸入（原文或鍵）下經過 DES所得的輸出，是要加以比較的。測試分兩個方向進行：1 鍵固定測試。這是在鍵不變的情形下，小幅度改變輸入的原文。藉此以觀察輸出方面的不同；2 原文固定測試。這時原文輸入不變，變的是鍵。

## 4-1. 鍵固定測試

好的密碼的第一個要求是：密碼文相對於原文的亂度必須足夠。從截收到的密碼文推算原文，在不知道鍵的情況下，我們希望任何原文的可能性都是一樣。換言之，密碼文不應攜帶原文的任何訊息。

由於 MEYER [11] 已証明了 DES 的輸出裡任一數元都與原文的任一數元有關，這個事實賦予密碼文相對於原文亂度的有利條件。在以下各種測試裡，我們將進一步比較原文與密碼文的關係。

測試甲：1 原文的任一數元是 "0" 或 "1" 所帶給密碼文任一數元的影響，是不變的 "0" 或 "1"，或是變的 "1" 或 "0"？
2 原文與密碼文比較，共有多少數元相同？

說明：測試值有 10,000 組，原文令爲向量A，密碼文令爲向量 B,K=64，求A 與B 的六項向量 $B_1, B_2, B_3, D_1, D_2,$ 與 $D_3$.

結果：由 $B_1(i), B_2(i), B_3(i)$ 的值均在 5000 左右，正負誤差不超過 130，顯示原文第 i 數元的值對密碼文第 i 數元，第 i+32 mod 64 數元，以及第 i+1 數元（當 i 爲奇數），或第 i-1 數元（當 i 爲偶數）的值沒有顯著影響。換句話說，密碼文的第 i-bit,(i+32)-bit, i+1（或 i-1）bit，所受到的原文第i 個數元的影響均相仿。

$D_1(i)$ 在 i 爲 16 開始有不爲 0 的值。依次上升至 i=34, $D_1(i)$ 爲最大，再次第下降。至 i=47 後，$D_1(i)$ 爲 0. $D_2(i)$ 與 $D_3(i)$ 亦有類似的分佈。這種分佈極近似於二次分配 (binomial distribution):

$$C(64,i)\left(\frac{1}{2}\right)^{i}\left(\frac{1}{2}\right)^{64-i}$$

解釋爲：任一原文與其密碼文，就對應之數元加以比較，算出相同的數元總數，令爲亂數變數 (random variable)，則其頻率分佈相近於每一數元比較相同的機率是½ 的二次分配。見附錄 2-1.

結論：以上結果顯示出，密碼文相對於原文，在三種選擇性的測試中，有很好的亂度。

好密碼的第二個要求是，取兩個型態相同或有某種關係的原文，它們的密碼文之間沒有重大的相關性，如此方不會提供易解的綫索。以下我們提出 9 種原文之間有某種關係的測試。

測試乙：1 只有一數元不同的兩對原文，它們的密碼文之間的相關性；
2 只有第 i 數元不同的兩對原文，對密碼文的影響；與第 j 數元不同的兩對原文，對密碼文的影響，它們之間有何關連？

說明：測試 2000 組，原文經 DES 所得的密碼文，定爲向量 A，原文改變第 ii 數元，經 DES 得的密碼文，定爲向量 B，求 A 與 B 的六項向量 $B_1, B_2, B_3, D_1, D_2, D_3$.

測試種類：改變 i=1,2,...,64 數元，共64種情形。

結果：1. 64種情形的結果均類似，$B_1(i), B_2(i), B_3(i)$ 均在 1000 左右，正負不超過 50，顯示密碼文間每一數元的相同否，爲½ 的機率。
2. $D_1(i), D_2(i), D_3(i)$ 在 i=20 左右開始有不爲零的值，i=33 左右，$D_1(i), D_2(i), D_3(i)$ 爲最大。在 i=45 以後，三種值都爲零，顯示原文雖只一數元不同，但密碼文相同的數元數目頻率卻呈二次分配：B(64,i).
3. 根據 $D_1(i), D_2(i), D_3(i)$ 所分別做的統計，顯示三者均甚類似：

Mean 在 32 左右，

Standard deviatiow = 4,

Relative standard deviation = 12.5,

Skewness = 0.004,

Kurtosis = 2.9

這些條件均使 $D_1, D_2, D_3$ 的分佈指向常規分配 (Normal distribution)，依據中央極限定理 (Central limit theorem) 二次分配在 B(N,P),N 相當大時，趨近常規分配。見附錄2-2.

結論：1.改變原文某一數元對密碼文任一數元的關係：$D_1(i), B_2(i), B_3(i)$ 的接近 1000，及 $D_1, D_2, D_3$ 的趨近常規分配，說明原文改變任一數元，對密碼文任一數元的影響均相同。

2. 改變原文任一數元對密碼文的差別性：改變第 i 數元與改變第 數元的統計結果均相似，表示改變原文任一數元對密碼文整個的影響均相同。

3. 由 1,2知，原文間的原有特殊關係，並不在密碼文裡被保存下來，或產生其他具體的關係，因此不能有整理密碼文的相關性來猜度原文。

測試丙：只有兩數元不同的原文，它們的密碼文之間的相關性。

說明：測試值有 1000 組，原文經 DES 所得的密碼文，定爲 A,原文改變指定的某二數元後，經 DES 之密碼文定爲向量B，A 與 B 求 $B_1, D_1$.

測試種類：改變第 1,2數元；第 37,42 數元，第 52,5 數元。

結論：其結論與測試乙同。見附錄 2-3..

測試丁：只有一數元，二數元，或四數元不同的原文，它們的密碼文做相對數元的比較，任何四數元的集合中，四數元均相同的情形，估測試數多少。

說明：測試值有 1000 組，有64 種四數元構成的組合：

第一類：[1,2,3,4]..... [61,62,63,64]

第二類：[2,3,4,5]:·····[62,63,64,1]

第三類： [3,4,5,6]..... [63,64,1,2]
第四類： [4,5,6,7]..... [64,1,2,3]

原文的密碼文定爲A,原文改變指定的數元，譯碼後得的密碼文定爲每一組作 A與 B相對數元的比較，若 4 數元均相同，則在該類的相關變數上加 1 。例 [1,2,3,4] 作比較，若四數元均同，則在第一類的 $D_1(1)$上加 1 。

結果：任何一類的 $D_1(i)$ 值在 64左右，正負不超過 20, 見附錄 2-4.

結論：四數元的值，有16種組合，完全相同的情形僅佔 1/16, 即 1000/16, 測試的結果與機率完全符合。
原文僅少數數元不同，但它不能在密碼文間找出四數元相同的組合，其頻率有異常情形。

測試戊：向左平移數位元，它們的密碼文之間的相關性。

說明：測試值有 1000 組，原文的密碼文定爲A, 原文左移指定數元後，經譯碼所得的密碼文定爲 B,A 與 B 相對的位元比較，求$B_1$及$D_1$.

測試種類：平移 1 數元， 7 數元， 18 數元， 2 數元等四種情形。

結果： $D_1(i)$在 [20,45] 之間才有不爲零的值 i在31 左右爲最大，然後向兩邊下降。$B_1$在 500 附近，正負不超過 60, 見附錄 2-5.

結論：1 平移後，對密碼文任一數元的影響沒有直觀的差別性。
2 平移的數元數目，對密碼文的影響沒有差別。
3. 原文間存在的左移關係，在密碼文間沒有保存。

測試己：原文間有平移關係，任何四數元的組合，它們的密碼文做相對數元的比較，尋求是否有幾組，它的值相同數目特別異常者。

測試種類：左移14種情形，即左移 1,2,3,4,8,9,10,12,20,32,36,48,56, 62, 的數元數。

說明：測試值有 1000 組 ,64 種情形與測試丁同。

結果：與測試丁類似。見附錄 2-6.

結論：原文間存在的平移關係，並不能在密碼文間找到值相同的數目有異常者。僅止於機率關係 1000/16.

測試庚：原文經過譯碼的 IP 函數後，左邊32數元保持相同，右邊32數元，所得的不經過 $IP^{-1}$ 的密碼文間的關係。

說明：測試值有 1000 組，第 1 個原文經譯碼 IP 函數得的左邊32數元取代以後，999 組原文經 IP 所得的左邊32數元，他們不經過 $IP^{-1}$ 的密碼文作 3 種統計量，即以左邊32數元作 $B_1, D_1$, 以右邊32數元作 $B_1, D_1$, 再以64數元作 $B_1, D_1$.

結果：左邊 $D_1(i)$ 在 i=18 處最大，右邊 $D_1(i)$ 在 i=17 處最大，整體的 $D_1(i)$ 在 i=33 處最大，三者都向兩邊下降。見附錄 2-7.

結論：原文經過 IP, 左邊相同的原文，它們不經過 $IP^{-1}$ 的密碼文間不具關係。

測試辛：經過 IP 函數，左邊32數元保持相同的原文，他們不經過 $IP^{-1}$ 的密碼文，各數元加以比較，任何四數元的組合中，具有完全相同的值的情形有無異常的頻率？

結果：非常的接近機率分配。見附錄 2-8.

結論：密碼文間不具有所要找尋的特殊頻率。

測試壬：經過 IP 函數，右邊32數元保持相同的原文，他們不經過 $IP^{-1}$ 的密碼文間的關係。

結果：與測試庚同。

測試癸：經過 IP 函數，右邊32數元保持相同的原文，他們不經過 $IP^{-1}$ 的密碼文，數元對數元的比較，任何四數元的組合中，具有完全相同的值的情形有無異常的頻率？

結果：與測試辛同。

由這10種測試所得的統計結果，我們大致可作如此的結論：在鍵固定的情形下，原文與密碼文的任一數元之間都具有強烈的依賴性。在原文方面的任何改變，大都會在密碼文方面引起相當大的改變。因而導致無法藉觀察密碼文間的差異，來推論原文的關係。

4-2. 原文固定所做的測試

DES 的密碼文是原文與鍵的函數： $C=S_K(P)$ 。第一節裡的測試，說明了許多原文間的相關性，都未在密碼文裡顯示出來。這一節裡將指出鍵與鍵之間的關連，也將被 DES 的譯碼過程所打散，不會在密碼文裡顯示。

密碼的第三個要求是兩個型態相同或有某種關係的鍵，在固定的原文下，它們的密碼文之間不存在重大的關連性，以提供鍵的搜索過程任何便利。

測試子：1 只有一數元不同的兩個鍵，它們的密碼文之間的相關性；
2 只有第 2 數元不同的兩個鍵對密碼文的影響與第J 數元不同的兩個鍵對密碼文的影響有何不同？

說明：測試值有 2000 組，原文經譯碼後的密碼文，令爲向量A,鍵改變第K數元，所得的密碼文，令爲向量 B。求A與B 的六種向量 $B_1, B_2, B_3, D_1, D_2, D_3$.

測試種類：鍵改變第 1 數元到改變第64數元，64 種情形。

結果：1 64種情形的結果均類似，$B_1(i), B_2(i), B_3(i)$ ， 均在 1000 左右，正負不超過 50，顯示密碼文間三種對應數元的比較，值相同的機率均爲 ½.

2. $D_1(i), D_2(i), D_3(i)$ 在 i=20 左右開始有不爲零的值，i=33 左右， $D_1(i), D_2(i), D_3(i)$ 爲最大，在 i=45 以後，三個值都爲0, 顯示鍵雖只一數元不同，但是密碼文相同的數元數目之頻率卻是二次分配 B(64,i)

3. 根據 $D_1(i), D_2(i), D_3(i)$ 分別作的統計，顯示三者均類示：

Mean = 32 左右，

Standard deviation = 4 ,

Relative Standard deviation = 12.5

Skewness = 0.004 .

Kurtosis = 2.9

這些條件使得 $D_1, D_2, D_3$ 的分布指向常規分配 (normal distribution)。根據中央極限定理，二次分配 B(N,P) 在 N相當大時（N=30 可謂大），趨近常規分配。見附錄 2-10.

結論：1 改變鍵某一數元對密碼文任一數元的關係：$B_1(i), B_2(i), B_3(i)$ 接近 1000, $D_1, D_2, D_3$ 的趨近常態分配，說明鍵的改變一數元，對密碼文任一數元的影響均相同。

2 改變鍵任一數元對密碼文的差別性：改變第 i 數元與改變第 j 數元的統計結果相似。表示改變鍵任一數元對密碼文的整體影響均大致相同。

3 由 1,2 知，鍵間存在的特殊關係，一般均不在密碼文裡反映出來。因此通常無法由整理密碼文的相關性，來猜度鍵間的關係。

密碼的其次要求是密碼文對鍵的亂度必須夠。

測試丑：1 原文任一數元是"0" 或"1" 所帶給密碼文任一數元是不變的 "0" 或 "1" 或者變的"1" 或"0".

2 原文與其密碼文對應的數元相同的數目。

說明：測試值有 10,000 組，原文令爲向量 A，其密碼文令爲向量 B。求 A 與 B 的六種向量 $B_1, B_2, B_3, D_1, D_2, D_3$.

結果：與測試甲同。

結論：結果顯示，密碼文對鍵，在三種選擇性的測試下均有很好的亂度。

由上述這兩個測試結果，我們同樣判斷：DES 的設計，使得鍵與密碼文的各個數元之間，具有密切的相關性，但卻無法加以簡單表示出來。因此，我們不得不承認，DES 的構造接近於完美，簡單的統計分析將無法瓦解 DES 的安全性。

## 五、修改 S-box 的測試

前面已經討論過，在 DES裡除了 S-boxes 以外，其餘導致變化的函數均爲線性函數。故S-boxes 應爲整個 DES 機密及安全之所在。本章裡，我們將藉著修改這些S-boxes 的結構，來探討它們在整個法則裡的力量。

構想是這樣的，即將 S-boxes加以簡化，然後觀察原文與密碼文之間是否會顯露出明顯的關係，因而供給破碼者寶貴的訊息。

測試進行中所採用的輸入向量以及各向量 $B_1, B_2, B_3, D_1, D_2, D_3$ 的定義，均與第四章裡的定義相同。

### 5-1. 方法一

修改 $S_1, S_2, \ldots S_8$ ， 使得每一列均爲 0,1,2,....15 的 S-box。做如此的修改，相當於令S-box 是一個全等轉換（或S-box 不存在）。法則中沒有 S-box 的效果，將使得原文間所存在的各種關連，或是各鍵間所存在的各種關連，都會在密碼文間洩露出重大的線索。因此令密碼分析者可利用這項特性，縮短破碼的路程。

另外在這方面的測試，亦同時顯露出 DES 爲何要選擇16 道的線索。因已知 DES在第五道以後，已達到完全依賴 (total dependancy) 的狀態。故16 道的選擇，具有探索的價值。

測試實：由亂數產生器產生原文 2000 組，用固定的鍵，再改變原文某一數元，比較兩者密碼文之間的關係，求密碼文間的 $B_1, B_2, B_3, D_1, D_2, D_3$.

測試種類：改變第 1 數元的情形，及改變第 8 數元；用 4 個不同的鍵：64 數元均爲 0, 64 數元均爲 1 ，及兩個亂數。

結果：改變第 1 數元的情形：

1. 密碼文之間相對數元的比較：用 4 種鍵，所得的 $B_1, D_1$ 值完全相同。$B_1$ 在第 2,17,28,36,40,48 數元爲 0 ，其餘數元爲 2000.
2. 密碼文之間奇偶數數元的比較，及 modulo 64 的比較：$D_2$ 及 $D_3$ 在偶數數元全爲零。

改變第 8 數元的情形：

1 密碼文間相對數元的比較：用 4 種鍵所得的 $B_1, D_2$ 值完全相同。$D_1$ 在第 16,33,59 數元爲 0，其餘數元爲 2000.

2 密碼文之間奇偶數數元的比較及 modulo 64 的比較：$D_2$ 及 $D_3$ 在奇數數元全爲零。見附錄 2-11.

結論：無論原文與鍵是什麼，只要改變的數元位置相同，則在比較密碼文時，發現在某些固定的位置會有相同的輸出數元。由此可說明兩點：

1 鍵的作用在本測試中是完全被抵銷了。

2 原文的個別性在密碼文的比較中被抵銷了。

由這兩點結論，我們可以推知，在使用這種簡化的 DES 時，密碼分析者若已知兩組原文僅僅相差一數元，則經由測試寅便可推知該不同數元的位置了。

測試卯：由亂數產生器得出 2000 組鍵，然後用固定的原文，再改變鍵裡的某一數元，比較兩者密碼文之間的關係。求密碼文間的 $B_1, B_2, B_3, D_1, D_2, D_3$.

測試種類：改變第 1 數元的情形及改變第 8 數元的情形，用 4 組原文：64 數元均爲 0, 64 數元均爲 1，及兩組亂數。

結果：改變第 1 數元的情形：

1 密碼文間相對數元的比較：用 4 組原文，所得的 $B_1, D_1$ 值完全相同。$B_1$ 在第 2,17,28,36,40,48 數元爲 0，其餘爲 2000; $D_1$ 在第59數元爲 2000, 其餘爲 0。

改變第 8 數元的情形：

1 密碼文間相對數元之比較：用 4 種原文，所得的 $B_1, D_1$ 值完全相同，$B_1$ 在第 16,33,59 數元爲 0，其餘數元爲 2000.

2 譯碼文間奇偶數數元的比較及 modulo 64 後的比較：$D_2$ 及 $D_3$ 在奇數數元全爲 0，見附錄 2-13.

結論：無論鍵與原文是什麼，只要鍵改變的數元位置相同，則在比較密碼文時，會發現在某些固定的位置有相同值出現。由此可說明兩點：

1 鍵的個別性在密碼文之間的比較中被抵銷了。

2 原文的作用也被抵銷了。

由這兩點結論，我們同樣可得出一結論：即若密碼分析者知道在相同原文下，有兩個鍵僅相差一數元，那麼經由測試卯就能求出該不同數元的位置。

在取掉 DES 裡的 S-boxes的情形下，重覆前章裡的測試甲。則在做每一道的輸出與原文的關係比較時，發現如果鍵全爲 0，或鍵全爲 1，在第 3,6,9,12,15 道時，$B_1$ 與$B_3$的偶數數元全爲零。如果鍵是其它兩個亂數，則在第 3,6,12 道的偶數數元全爲零；而在第 9,15 道奇數數元亦全爲零。但兩者在第16道均爲良好的亂數。（見附錄 2-12 ）。

因此可知，在去掉 DES裡的 S-boxes 的情況下，不能選擇 3 的倍數爲總道數。因爲在這些道上出現不尋常的密碼文。雖然對 DES的測試甲與測試丑均顯示每一道的密碼文有很好的亂度。但是，如果 DES眞如一般對其之揣測，具有未經公開的發展結構，那麼16道的選擇是否與除掉 S-box 後仍保存著的某種結構有關呢？

沒有 S-boxes 的這些重大弱點，適足証明 S-boxes 在法則中的地位：它使得有關係的原文，在密碼文之間不復顯現其中的關係。

5-2. 方法二

在這個方法裡，S-boxes 修改的規則是：S-box 的每個輸出，是其輸入的補數。也就是說 0→15, 1→14,...,15→0 。它的型態仍屬於 S-box，而不是 P-box.

把 DES 裡的 S-boxes 換成這種型態，然後進行相同於前一節裡所進行過的測試，測試寅及測試卯，我們發現所得出的結果與前一節裡去除 S-boxes 所得的結果完全一樣。換句話說，原文間的某些關連，會在密碼文間漏出極大線索；而鍵之間的關連，也會產生相同的結果。密碼分析者可根據這些特性，縮短破碼的路程。

此外，重覆測試甲，它也會在 3 的倍數道上，出現與第一節相似的結果。請見附錄 2-15.

另外綜合前節和本節的結果，我們或可得出一個猜測：就是若令所有的 S-boxes 均相等，將使原文與密碼文之間產生强烈的相關性。這個猜測主要是因此時鍵所担任的角色被減輕了，並沒有需要去揀選不同的

這應會導致密碼的安全性有極大部份是依賴 P-boxes 和其他線性函數。而我們已知 P-boxes 和線性函數的安全性是相當經不起考驗的。但結果眞會是如此嗎？下一節裡將就此加以討論。

## 5-3. 方法三

在這個修改 S-boxes 的方法裡，我們令每一列均爲 10,6,9,13,15,7,0,12,4,5,3,11,1,8,2,14.

這個 S-bos 的安排，是由任一的重組取得。

具有這樣平凡安排的 S-bos ，它似乎可代表一般沒有任何結構支持的 S-box 型態。它是經由亂度化 (randomization) 才取得的。理由也就是要避免某種數學結構的形成。

一般對 DES 的 S-boxes 有一種看法。就是這些 S-boxes 的安排可能是經由亂度化而取得的。希望經由亂度化以避免任何結構帶來的有利於破碼分析的條件。因爲亂度化的結果，通常都有無法分析的性質。

因此，若我們以上對 S-box 的重新安排，仍能導致良好的結果，這將對 DES 裡的 S-boxes 是亂度化產物的揣測，產生極有力的支持。

本節即根據此一方向的構想，作測試貳及測試叁。測試的結果，爲完美的機率性分布。見附錄 2-16.

因此，DES 裡的 S-boxes 爲亂度化產物的揣測，有其可能性。

## 5-4. S-boxes 的其他修改

在前面對 DES 裡 S-boxes 的修改方法，所有的 S-boxes 都被改成相同。因此都可算是大幅度的修改。所得出的結果相當有趣，顯示出若 S-boxes 的構造有强烈的規則性，則會導致安全上的漏洞。

在本節裡，我們將只小幅度的修改 DES 裡的 S-boxes ，希望藉出能顯示出一些有意義的資訊。我們將 S-boxes 依下面幾個途徑分別修改：

1. 第 3 個 S-box 全改爲 0,1,2,...,15, 其餘不變；
2. 每個 S-box 的第一列全改爲 0,1,2,...,15, 其餘不變；
3. 第一個 S-box 的第一列改爲 0,1,2,...,15, 其餘不變；

及

4. 擴張函數 (E function ) 的某個元素修改。

測試的結果，是均產生機率分布。亦即與 DES 的原有結果無甚差別。這對 DES 的安全度的評估，有極大意義。顯示出 DES 的安全度，遠超過某一臨界點。因此任何基於 DES 再做些微改變所導致的新譯碼方法，仍然具有良如的安全度。這使得 DES 在結構方面，至目前爲止是無懈可擊。所剩下的唯一顧慮，僅是在竭盡式搜求方法。而這項顧慮可以藉增加鍵長度很容易就克服。

其次，本節裡的測試結果也支持了對 DES 裡 S-boxes 的揣測。亦即這些 S-boxes 是由亂度化所產生的。如果這些 S-boxes 的構造是經由安排所得，那我們也必須承認，這些安排幾乎無法被尋找出來。

## 六、結論

DES 是截至目前爲止，結構公布得最清楚的一種譯碼方法。因此它收到相當大的注意力，希望能破解它，或望能藉著對它的研究，得出一些密碼機設計的機密之處。在本研究裡，我們亦抱著同樣的看法。

本研究對 DES 的分析，主要是循著三個途徑：鍵竭盡搜求、法則分析、以及介於兩者之間的統計測試分析。三者都有互相發明的影響。其中鍵竭盡搜求法需仰賴實際的技術環境。由於至今尚沒有這類機器出現，成功與否仍是未知之數。不過它的不需任何代數分析以及不需知道任何暗門 (trapdoor) 即可導致破解，是歷年來最受爭論與矚目的方法之一。無可否認的，在未來有相當的可能會產生此種破碼機。未雨綢繆計，DES 必須即早有修改的方法。

在法則分析方面，至今尚未發現任何足以嚴重威脅DES 安全的代數結構。也正因此，對 DES 的結構有兩種揣測：其一是它背面確有不公開的發展結構或設計因素。針對這一點，於是有人要求將 DES 的設計過程完全公開，而非僅是 DES 這個成品而已。他們所持的觀點是：不公開的密碼法則是不好的密碼系統，存有被任何人能暗中破解的潛在危險。另一個對 DES 的揣測是： DES 並沒有任何未經公開的潛在結構， S-boxes 的選取不過是完美的亂數化結果。針對這一點的理由是，亂數化的選取S-boxes ，避免了利於分析的代數關係。而Shannon 所提出的這個 SP-boxes 交乘密碼系統的觀念，本身即具備强大的威力，能使密碼文難於捉摸。因此，沒有必要對S-boxes 加以特別設計。

本研究裡所嘗試的第三種方法，是選擇性的統計測試。這種方法 取了前述兩法的觀念，加以融合。就如同兒童的拼圖遊戲，剛開始可能一點頭緒也沒有，但在諸多嘗試中，可逐漸獲得靈感。根據這些靈感，逐步發展出有意義的測試。期能在實際的測試裡做到求証的工夫及爲下一步的靈感鋪路。這符合科學的處理：〃假設、求証、再假設、再求証。〃因此，它應該是最適合，最值得進一步嘗試的方向。在本研究裡，重點也就是放在這方面。不過我們所做的仍僅是起步而已。但起步最難，因爲沒有一點輪廓的痕跡。而測試方法的改進需要大量的測試與時間爲基礎。但由於我們進行此測試所使用的工具—PDP—11/70 的限制，無法進行大規模較深入的測試。在有限時間與設備的情況下，我們僅能選擇重點進行。許多細節部份尚待進一步探討。不過到目前爲止，所有的測試結果都是非常漂亮的

機率性結果。在沒有突破性的發現前，這些結果顯示DES 是安全的。

IBM在這方面做了17人 /年 (17 man / year) 的測試，結果卻未見公布。

進一步的測試方面，指向S-boxes 的系統化測試，及代數結構的模擬等。在本研究裡雖然確認了S-boxes 存在的重要性，但S-boxes 應以何種型態存在，則仍是個重要問題。倘若能掌握 S-boxes 的設計原則，我們將可設計一套修改過的DES 以爲我國之用。在目前，我們不敢推薦 DES爲我國所用，理由卽是如果美國在暗中掌握了 DES 可能有的暗門，則弱點將操在別人手裡。

在修改 DES裡 S-boxes 的測試中，我們發現在 3 的倍數道上，出現了一些值得注意的結果。它們提供了某種尙未明朗的線索，並可能與16道的選擇有關。這些尙待進一步的研究。

另外，鍵在16道的產生是經迴轉左移得到，它之不用左移的性質是不失去資訊。這與在擴張函數中，選擇行的數元包括所有的32數元有同功之效。如果使 DES失去此特性，結果將如何？法則中的模數 2 相加，若被其它可逆函數取代，結果會如何？這都待進一步追查。

對 DES尙有一推測：DES 的那些公佈的表只是另外一個不公布的法則的結果。DES 可能擁有一個暗門，如果知道它，那麼破碼非常容易，是否眞有這樣一個函數 $f$ 呢？

DES 的應用，除了作爲譯碼用的密碼系統外，尙可在網路通訊中作爲鍵的產生器來使用。這主要是因 DES打算了原有數據的任何結構，故可當作一亂數產生器來使用。計算機中若要附加 DES 系統的譯碼、解碼作業，如今已有現成的微處理機可供使用，作業的速度也相當的迅速。

DES 的使用方式，除了區間譯碼外，尙有增加數元間交互影響的〃區段連鎖 (block chaining) 〃方式。方法有二：

1 $Y_i = E_k ( X_i \oplus Y_{i-1} ) , i \geq 1 .$

這時譯碼文 $Y_i$ 與其原文 $X_i$ 及前一區段的密碼文 $Y_{i-1}$ 有關。

2 $Y_1 = E_k ( X_1 \oplus Y_0 ) ,$

$Y_i = E_k ( X_i \oplus Y_{i-1} \oplus X_{i-1} ) , i > 1 .$

這時密碼文 $Y_i$ 與原文 $X_i$ 及前一區段的原文、密碼文有關。

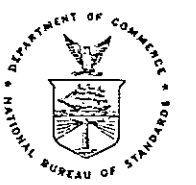
總之，DES 的有效性能，在目前是無法逕予駁斥的。在未發掘出 DES 的任何結構弱點前，DES 的使用會日趨普遍。但同時，對 DES 的測試，仍將持續下去。

# 參 考 文 獻


1. Data Encryption Standard, National Bureau of Standards, Federal Information Processing Standard (FIPS) Publicaton No.46, Jan. 1977.
2. Sloomon, R. J., "The Encryption Controversy," Mini-Micro Systems, Feb. 1978.
3. Diffie, W., and M. E. Hellman, "Privacy and Authentication : An Introduction to Cryptography," Proc. IEEE, vol.67, no.3, March 1979.
4. Shannon, C. E., "Communication Theory of Secrecy Systems," Bell Syst. Tech. J., vol.28, Oct. 1949.
5. Feistel, H., "Cryptography and Computer Privacy," Scientific American, vol.228, no.5, May 1973.
6. Diffie, W., and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, vol.10, no.6. June 1976.
7. Morris, R., N. J. A. Sloane, and A. D. Wyner, "Assessment of the National Bureau of Standards proposed federal data encryption standard," Cryptologia, vol.1, pp.281-291, July 1977.
8. Kam, J.,-and G. Davida, "Structured design of substitution-permutation encryption networks," IEEE Trans. Computers, vol.C-28, no.10, pp.747-753, Oct. 1979.
9. Hellman, M., R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer, "Results of an initial attempt to Cryptananlze the NBS data encryption standard," Electric Engineering Dept., Stanford Univ., Stanford, CA, SEL 76-042, Sept. 9, 1976.

Federal Information
Processing Standards Publication 46

1977 January 15



SPECIFICATIONS FOR THE

# DATA ENCRYPTION STANDARD

The Data Encryption Standard (DES) shall consist of the following Data Encryption Algorithm to be implemented in special purpose electronic devices. These devices shall be designed in such a way that they may be used in a computer system or network to provide cryptographic protection to binary coded data. The method of implementation will depend on the application and environment. The devices shall be implemented in such a way that they may be tested and validated as accurately performing the transformations specified in the following algorithm.

## DATA ENCRYPTION ALGORITHM

*Introduction*

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation $IP$, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation $IP^{-1}$. The key-dependent computation can be simply defined in terms of a function $f$, called the cipher function, and a function $KS$, called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described. Finally, a definition of the cipher function $f$ is given in terms of primitive functions which are called the selection functions $S_i$ and the permutation function $P$. $S_i$, $P$ and $KS$ of the algorithm are contained in the Appendix.

The following notation is convenient: Given two blocks $L$ and $R$ of bits, $LR$ denotes the block consisting of the bits of $L$ followed by the bits of $R$. Since concatenation is associative $B_1B_2 \ldots B_8$, for example, denotes the block consisting of the bits of $B_1$ followed by the bits of $B_2 \ldots$ followed by the bits of $B_8$.

*Enciphering*

A sketch of the enciphering computation is given in figure 1.

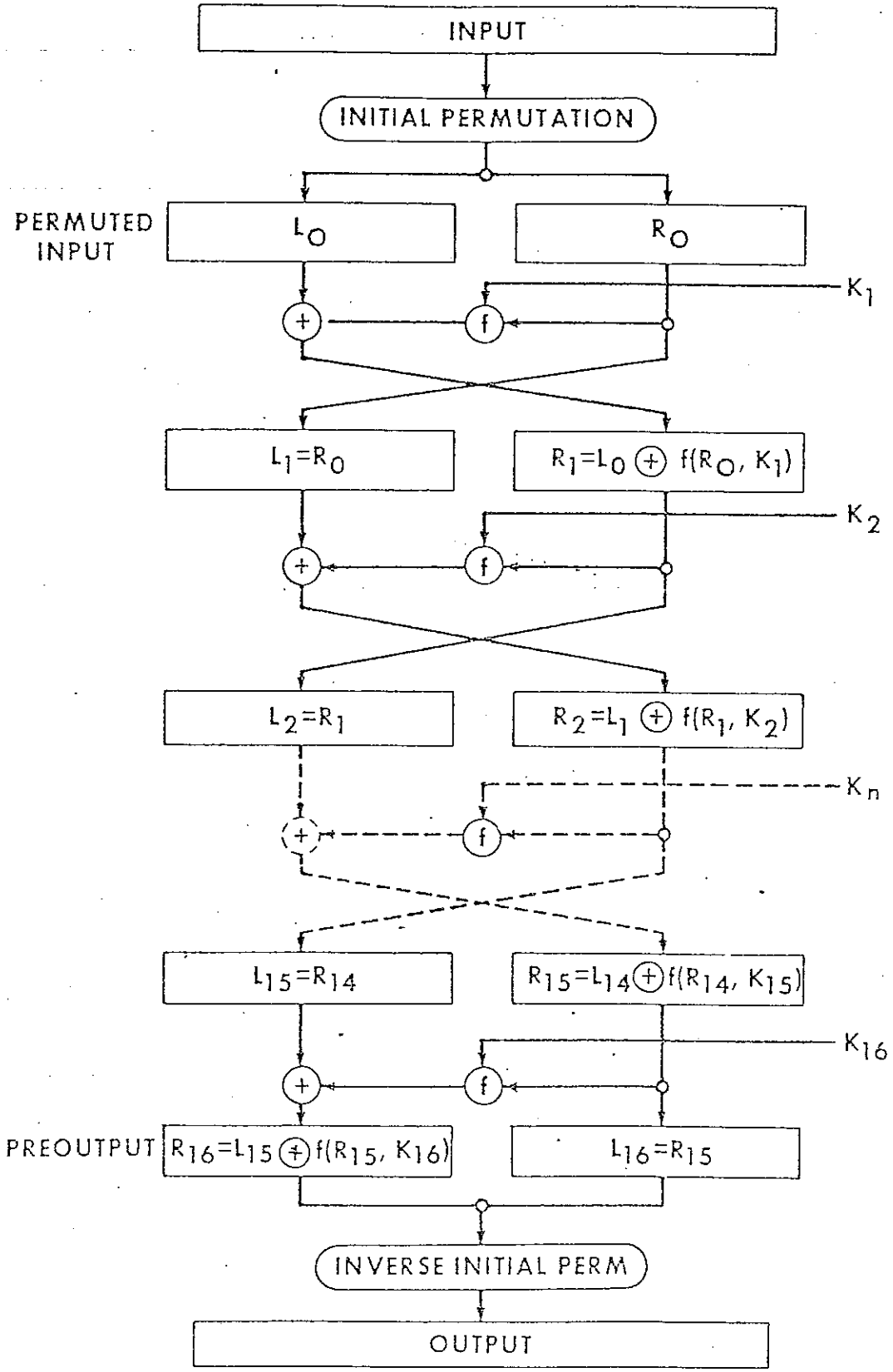



FIGURE 1. *Enciphering computation.*

The 64 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation *IP*:

*IP*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

That is the permuted input has bit 58 of the input as its first bit, bit 50 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation described below. The output of that computation, called the preoutput, is then subjected to the following permutation which is the inverse of the initial permutation:

$IP^{-1}$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

That is, the output of the algorithm has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

The computation which uses the permuted input block as its input to produce the preoutput block consists, but for a final interchange of blocks, of 16 iterations of a calculation that is described below in terms of the cipher function $f$ which operates on two blocks, one of 32 bits and one of 48 bits, and produces a block of 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32 bit block $L$ followed by a 32 bit block $R$. Using the notation defined in the introduction, the input block is then $LR$.

Let $K$ be a block of 48 bits chosen from the 64-bit key. Then the output $L'R'$ of an iteration with input $LR$ is defined by:

(1)
$$L' = R$$
$$R' = L \oplus f(R,K)$$

where $\oplus$ denotes bit-by-bit addition modulo 2.

As remarked before, the input of the first iteration of the calculation is the permuted input block. If $L'R'$ is the output of the 16th iteration then $R'L'$ is the preoutput block. At each iteration a different block $K$ of key bits is chosen from the 64-bit key designated by $KEY$.

With more notation we can describe the iterations of the computation in more detail. Let $KS$ be a function which takes an integer $n$ in the range from 1 to 16 and a 64-bit block $KEY$ as input and yields as output a 48-bit block $K_n$ which is a permuted selection of bits from $KEY$. That is

(2) $$K_n = KS(n, KEY)$$

with $K_n$ determined by the bits in 48 distinct bit positions of $KEY$. $KS$ is called the key schedule because the block $K$ used in the $n$'th iteration of (1) is the block $K_n$ determined by (2).

As before, let the permuted input block be $LR$. Finally, let $L_0$ and $R_0$ be respectively $L$ and $R$ and let $L_n$ and $R_n$ be respectively $L'$ and $R'$ of (1) when $L$ and $R$ are respectively $L_{n-1}$ and $R_{n-1}$ and $K$ is $K_n$; that is, when $n$ is in the range from 1 to 16,

(3) $$L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

The preoutput block is then $R_{16}L_{16}$.

The key schedule $KS$ of the algorithm is described in detail in the Appendix. The key schedule produces the 16 $K_n$ which are required for the algorithm.

*Deciphering*

The permutation $IP^{-1}$ applied to the preoutput block is the inverse of the initial permutation $IP$ applied to the input. Further, from (1) it follows that:

(4) $$R = L'$$
$$L = R' \oplus f(L', K)$$

Consequently, to *decipher* it is only necessary to apply the *very same algorithm to an enciphered message block,* taking care that at each iteration of the computation *the same block of key bits* K *is used* during decipherment as was used during the encipherment of the block. Using the notation of the previous section, this can be expressed by the equations:

(5) $$R_{n-1} = L_n$$
$$L_{n-1} = R_n \oplus f(L_n, K_n)$$

where now $R_{16}L_{16}$ is the permuted input block for the deciphering calculation and $L_0R_0$ is the preoutput block. That is, for the decipherment calculation with $R_{16}L_{16}$ as the permuted input, $K_{16}$ is used in the first iteration, $K_{15}$ in the second, and so on, with $K_1$ used in the 16th iteration.

*The Cipher Function* f

A sketch of the calculation of $f(R, K)$ is given in figure 2.

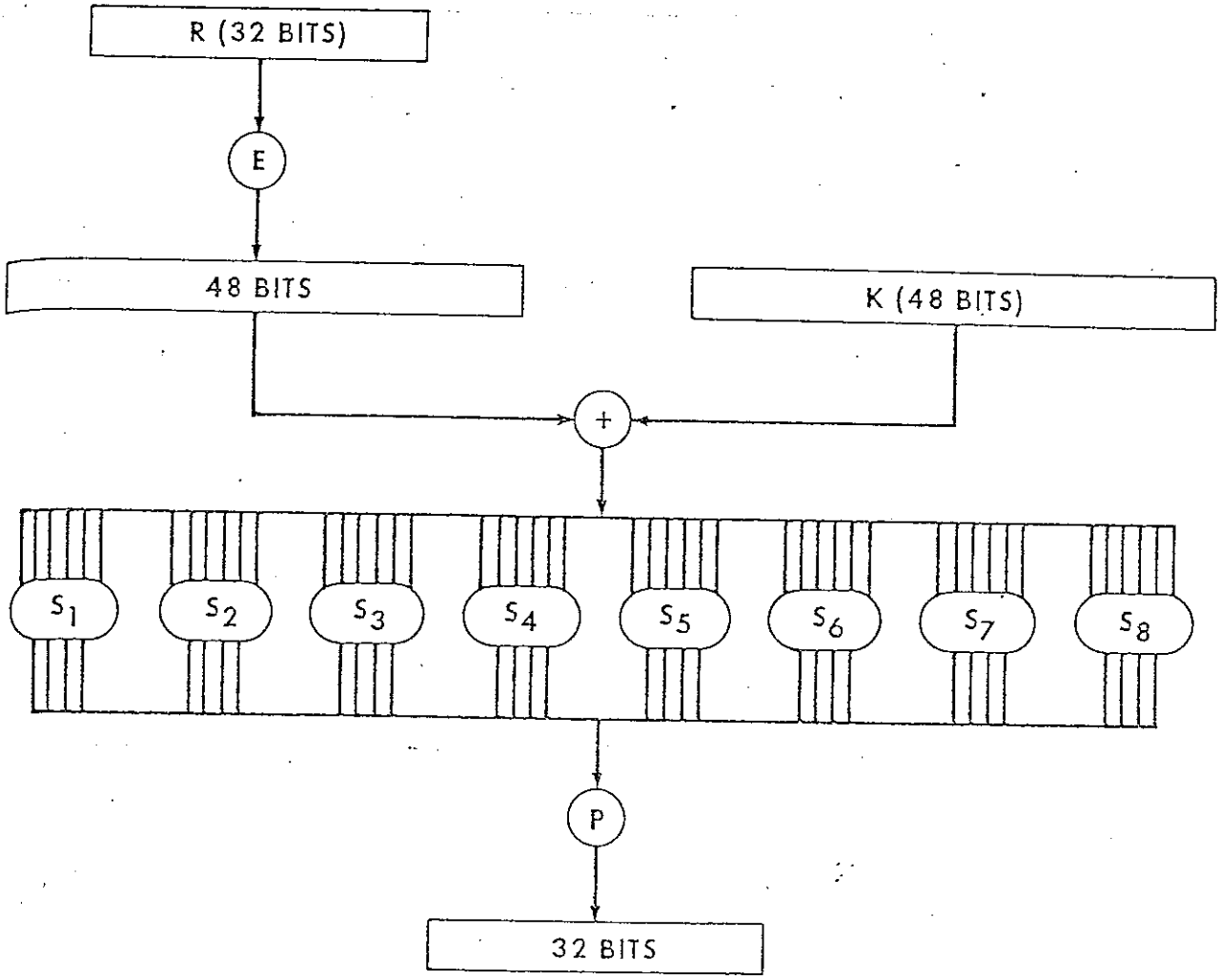



FIGURE 2. *Calculation of* f (R, K).

Let $E$ denote a function which takes a block of 32 bits as input and yields a block of 48 bits as output. Let $E$ be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

$E$ BIT-SELECTION TABLE

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

Thus the first three bits of $E(R)$ are the bits in positions 32, 1 and 2 of $R$ while the last 2 bits of $E(R)$ are the bits in positions 32 and 1.

Each of the unique selection functions $S_1, S_2, \ldots, S_8$, takes a 6-bit block as input and yields a 4-bit block as output and is illustrated by using a table containing the recommended $S_1$:

$S_1$

Column Number

| Row No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

If $S_1$ is the function defined in this table and $B$ is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of $B$ represent in base 2 a number in the range 0 to 3. Let that number be $i$. The middle 4 bits of $B$ represent in base 2 a number in the range 0 to 15. Let that number be $j$. Look up in the table the number in the $i$'th row and $j$'th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of $S_1$ for the input $B$. For example, for input 011011 the row is 01, that is row 1, and the column is determined by 1101, that is column 13. In row 1 column 13 appears 5 so that the output is 0101. Selection functions $S_1, S_2, \ldots, S_8$ of the algorithm appear in the Appendix.

The permutation function $P$ yields a 32-bit output from a 32-bit input by permuting the bits of the input block. Such a function is defined by the following table:

$P$

| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

The output $P(L)$ for the function $P$ defined by this table is obtained from the input $L$ by taking the 16th bit of $L$ as the first bit of $P(L)$, the 7th bit as the second bit of $P(L)$, and so on until the 25th bit of $L$ is taken as the 32nd bit of $P(L)$. The permutation function $P$ of the algorithm is repeated in the Appendix.

Now let $S_1, \ldots, S_8$ be eight distinct selection functions, let $P$ be the permutation function and let $E$ be the function defined above.

To define $f(R, K)$ we first define $B_1, \ldots, B_8$ to be blocks of 6 bits each for which

(6) $$B_1 B_2 \ldots B_8 = K \oplus E(R)$$

The block $f(R, K)$ is then defined to be

(7) $$P(S_1(B_1)S_2(B_2) \ldots S_8(B_8))$$

Thus $K \oplus E(R)$ is first divided into the 8 blocks as indicated in (6). Then each $B_i$ is taken as an input to $S_i$ and the 8 blocks $S_1(B_1)$, $S_2(B_2)$, . . ., $S_8(B_8)$ of 4 bits each are consolidated into a single block of 32 bits which forms the input to $P$. The output (7) is then the output of the function $f$ for the inputs $R$ and $K$.

APPENDIX

# PRIMITIVE FUNCTIONS FOR THE DATA ENCRYPTION ALGORITHM

The choice of the primitive functions $KS$, $S_1$, ..., $S_8$ and $P$ is critical to the strength of an encipherment resulting from the algorithm. Specified below is the recommended set of functions, describing $S_1$, ..., $S_8$ and $P$ in the same way they are described in the algorithm. For the interpretation of the tables describing these functions, see the discussion in the body of the algorithm.

The primitive functions $S_1$, ..., $S_8$, are:

$S_1$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

$S_2$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |

$S_3$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |

$S_4$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |

$S_5$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |

$S_6$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |

$S_7$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |

$S_8$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

The primitive function $P$ is:

| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

Recall that $K_n$, for $1 \leq n \leq 16$, is the block of 48 bits in (2) of the algorithm. Hence, to describe $KS$, it is sufficient to describe the calculation of $K_n$ from $KEY$ for $n = 1, 2, \ldots, 16$. That calculation is illustrated in figure 3. To complete the definition of $KS$ it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts. One bit in each 8-bit byte of the $KEY$ may be utilized for error detection in key generation, distribution and storage. Bits 8, 16, . . ., 64 are for use in assuring that each byte is of odd parity.

Permuted choice 1 is determined by the following table:

$PC-1$

| | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

The table has been divided into two parts, with the first part determining how the bits of $C_0$ are chosen, and the second part determining how the bits of $D_0$ are chosen. The bits of $KEY$ are numbered 1 through 64. The bits of $C_0$ are respectively bits 57, 49, 41, . . ., 44 and 36 of $KEY$, with the bits of $D_0$ being bits 63, 55, 47, . . ., 12 and 4 of $KEY$.

With $C_0$ and $D_0$ defined, we now define how the blocks $C_n$ and $D_n$ are obtained from the blocks $C_{n-1}$ and $D_{n-1}$, respectively, for $n = 1, 2, \ldots, 16$. That is accomplished by adhering to the following schedule of left shifts of the individual blocks:

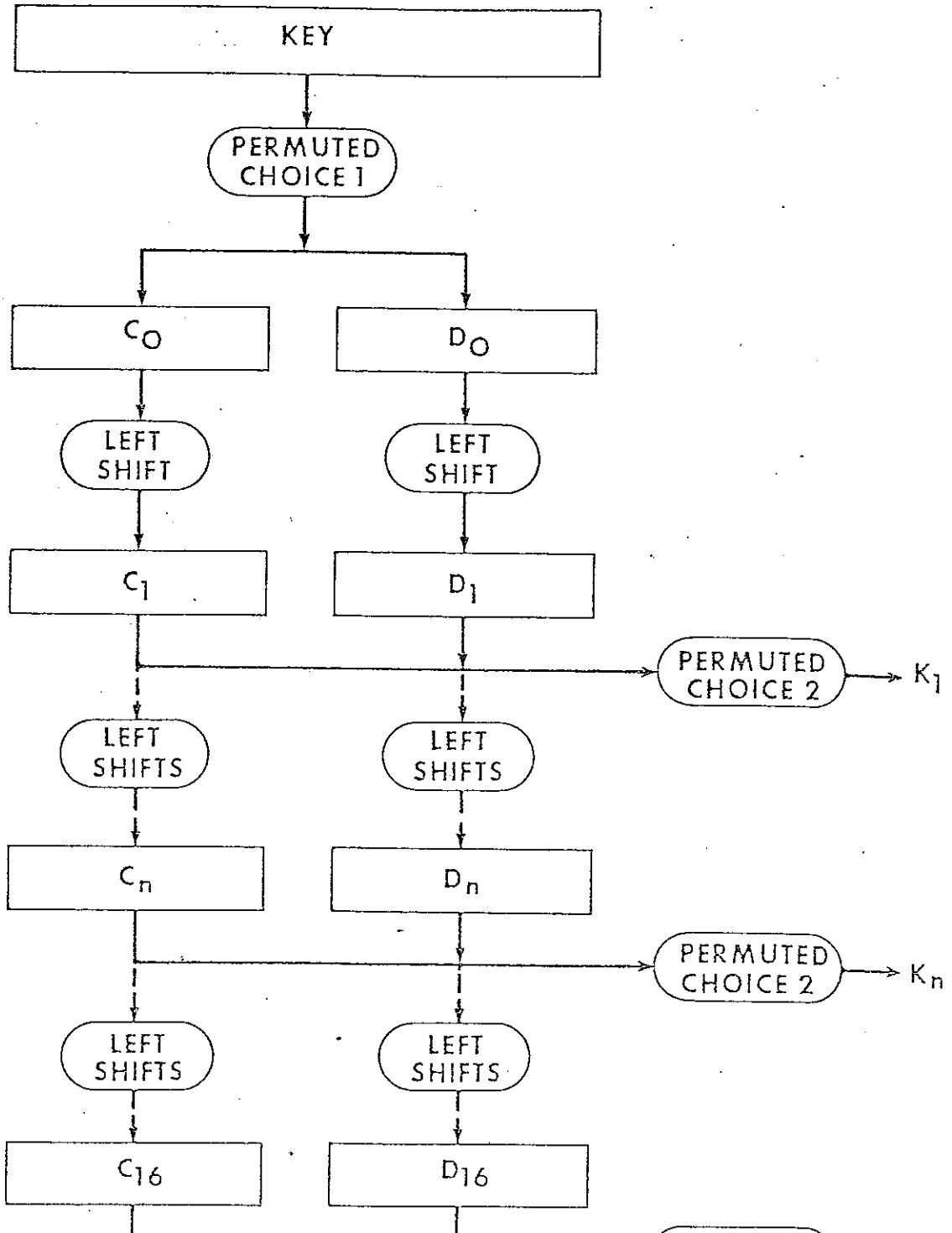



FIGURE 3. *Key schedule calculation.*

| Iteration Number | Number of Left Shifts |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 2 |
| 5 | 2 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | 2 |
| 11 | 2 |
| 12 | 2 |
| 13 | 2 |
| 14 | 2 |
| 15 | 2 |
| 16 | 1 |

For example, $C_3$ and $D_3$ are obtained from $C_2$ and $D_2$, respectively, by two left shifts, and $C_{16}$ and $D_{16}$ are obtained from $C_{15}$ and $D_{15}$, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3, . . ., 28, 1.

Permuted choice 2 is determined by the following table:

*PC–2*

| | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

Therefore, the first bit of $K_n$ is the 14th bit of $C_nD_n$, the second bit the 17th, and so on with the 47th bit the 29th, and the 48th bit the 32nd.

```
PIP TI:\;\;=DESSUS2.FTN
PIP -- NO SUCH FILE(S)
SY0:[240,10]DESSUS2.FTN
>
>PIP D\D\TI:=DESSUB2.FTN
CCC
C
C DESSUB2.FTN
CCC
      SUBROUTINE CIPHER(TEXT,CTEXT)
      IMPLICIT INTEGER (A-Z)
      DIMENSION A(64),KEY(64),LO(32),L1(32),RO(32),R1(32)
      DIMENSION KEYBUF(16,48)
      INTEGER TEXT(64),CTEXT(64)
      COMMON/KEYBUF/KEYBUF
      DO 2 I=1,64
      A(I)=TEXT(I)
    2 CONTINUE
      KNO=0
      CALL IP(A)
      DO 10 I=1,32
      J=I+32
      LO(I)=A(I)
      RO(I)=A(J)
   10 CONTINUE
   20 KNO=KNO+1
      DO 30 I=1,32
      L1(I)=RO(I)
   30 CONTINUE
      DO 35 I=1,48
   35 KEY(I)=KEYBUF(KNO,I)
      CALL F(RO,KEY)
      CALL EXOR(32,LO,RO)
      DO 40 I=1,32
      R1(I)=LO(I)
   40 CONTINUE
      IF(KNO.EQ.16) GO TO 100
      DO 50 I=1,32
      LO(I)=L1(I)
      RO(I)=R1(I)
   50 CONTINUE
      GO TO 20
  100 DO 110 I=1,32
      J=I+32
      CTEXT(I)=R1(I)
      CTEXT(J)=L1(I)
  110 CONTINUE
      CALL IP1(CTEXT)
      RETURN
      END
C
C
      SUBROUTINE DECIPR(TEXT,CTEXT)
      IMPLICIT INTEGER (A-Z)
      DIMENSION KEY(64),A(64),LO(32),L1(32),RO(32),R1(32)
      INTEGER KEYBUF(16,48),TEXT(64),CTEXT(64)
      COMMON/KEYBUF/KEYBUF
      KNO=17
      DO 5 I=1,64
      A(I)=CTEXT(I)
    5 CONTINUE
      CALL IP(A)
      DO 10 I=1,32
      J=I+32
```

附錄2

DES 譯碼與解碼法則的副程式

```
      L1(I)=A(J)
   10 CONTINUE
   20 KNO=KNO-1
      DO 30 I=1,32
   30 RO(I)=L1(I)
      DO 50 I=1,48
   50 KEY(I)=KEYBUF(KNO,I)
      CALL F(L1,KEY)
      CALL EXOR(32,R1,L1)
      DO 60 I=1,32
   60 LO(I)=R1(I)
      IF(KNO.EQ.1) GO TO 100
      DO 70 I=1,32
      L1(I)=LO(I)
   70 R1(I)=RO(I)
      GO TO 20
  100 DO 120 I=1,32
       J=I+32
      TEXT(I)=LO(I)
      TEXT(J)=RO(I)
  120 CONTINUE
      CALL IP1(TEXT)
      RETURN
      END
C
C
      SUBROUTINE IP(A)
      INTEGER A(64),TABLE(64),T(64)
      DATA TABLE/58,50,42,34,26,18,10, 2,
     *           60,52,44,36,28,20,12, 4,
     *           62,54,46,38,30,22,14, 6,
     *           64,56,48,40,32,24,16, 8,
     *           57,49,41,33,25,17, 9, 1,
     *           59,51,43,35,27,19,11, 3,
     *           61,53,45,37,29,21,13, 5,
     *           63,55,47,39,31,23,15, 7/
C
      DO 10 I=1,64
      K=TABLE(I)
      T(I)=A(K)
   10 CONTINUE
      DO 20 I=1,64
      A(I)=T(I)
   20 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE IP1(A)
      INTEGER A(64),TABLE(64),T(64)
      DATA TABLE/40, 8,48,16,56,24,64,32,
     *           39, 7,47,15,55,23,63,31,
     *           38, 6,46,14,54,22,62,30,
     *           37, 5,45,13,53,21,61,29,
     *           36, 4,44,12,52,20,60,28,
     *           35, 3,43,11,51,19,59,27,
     *           34, 2,42,10,50,18,58,26,
     *           33, 1,41, 9,49,17,57,25/
      DO 10 I=1,64
      K=TABLE(I)
      T(I)=A(K)
   10 CONTINUE
      DO 20 I=1,64
      A(I)=T(I)
```

```
      SUBROUTINE PCHOC1(KEY,KEYBUF)
      IMPLICIT INTEGER (A-Z)
      INTEGER KEY(64),TABLE(56),KI(56),KEYBUF(16,48)
      DATA TABLE/57,49,41,33,25,17, 9,
     *            1,58,50,42,34,26,18,
     *           10, 2,59,51,43,35,27,
     *           19,11, 3,60,52,44,36,
     *           63,55,47,39,31,23,15,
     *            7,62,54,46,38,30,22,
     *           14, 6,61,53,45,37,29,
     *           21,13, 5,28,20,12, 4/
      DO 10 I=1,56
      K=TABLE(I)
      KI(I)=KEY(K)
   10 CONTINUE
      DO 20 I=1,16
      II=I
      CALL GETKEY(II,KEY,KI)
      DO 15 J=1,48
   15 KEYBUF(I,J)=KEY(J)
   20 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE PCHOC2(KEY,KI)
      INTEGER KEY(64),TABLE(48),KI(56)
      DATA TABLE/14,17,11,24, 1, 5,
     *            3,28,15, 6,21,10,
     *           23,19,12, 4,26, 8,
     *           16, 7,27,20,13, 2,
     *           41,52,31,37,47,55,
     *           30,40,51,45,33,48,
     *           44,49,39,56,34,53,
     *           46,42,50,36,29,32/
      DO 10 I=1,48
      K=TABLE(I)
      KEY(I)=KI(K)
   10 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE GETKEY(KNO,KEY,KI)
      IMPLICIT INTEGER (A-Z)
      INTEGER TABLE(16),KEY(64),KI(56)
      DATA TABLE/1,1,2,2,2,2,2,2,
     *           1,2,2,2,2,2,2,1/
      K=TABLE(KNO)
   10 T1=KI(1)
      T2=KI(29)
      DO 20 I=2,28
      J=I+28
      KI(I-1)=KI(I)
      KI(J-1)=KI(J)
   20 CONTINUE
      KI(28)=T1
      KI(56)=T2
      IF(K.EQ.1) GO TO 50
      K=K-1
      GO TO 10
   50 CALL PCHOC2(KEY,KI)
      RETURN
      END
```

```
C
C
      SUBROUTINE F(RR,KEY)
      IMPLICIT INTEGER (A-Z)
      INTEGER RR(32),KEY(64),TABLE(48),T(48),IN(6),OUT(4)
      DATA TABLE/32,-1,-2,-3,-4,-5,
     *           4, 5, 6, 7, 8, 9,
     *           8, 9,10,11,12,13,
     *          12,13,14,15,16,17,
     *          16,17,18,19,20,21,
     *          20,21,22,23,24,25,
     *          24,25,26,27,28,29,
     *          28,29,30,31,32, 1/
      DO 10 I=1,48
      K=TABLE(I)
      T(I)=RR(K)
   10 CONTINUE
      CALL EXOR(48,KEY,T)
      DO 40 I=1,8
      DO 20 J=1,6
      K=(I-1)*6+J
   20 IN(J)=KEY(K)
      II=I
      CALL S(II,IN,OUT)
      DO 30 J=1,4
      K=(I-1)*4+J
   30 RR(K)=OUT(J)
   40 CONTINUE
      CALL P(RR)
      RETURN
      END
C
C
      SUBROUTINE P(A)
      INTEGER A(32),TABLE(32),T(32)
      DATA TABLE/16, 7,20,21,
     *           29,12,28,17,
     *            1,15,23,26,
     *            5,18,31,10,
     *            2, 8,24,14,
     *           32,27, 3, 9,
     *           19,13,30, 6,
     *           22,11, 4,25/
      DO 10 I=1,32
      K=TABLE(I)
      T(I)=A(K)
   10 CONTINUE
      DO 20 I=1,32
      A(I)=T(I)
   20 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE EXOR(N,A,B)
      INTEGER A(N),B(N)
      DO 20 I=1,N
      IF(A(I).EQ.B(I)) GO TO 10
      A(I)=1
      GO TO 20
   10 A(I)=0
   20  CONTINUE
      RETURN
      END
```

```
      IMPLICIT INTEGER (A-Z)
      INTEGER IN(6),OUT(4),SS1(16,4),SS2(16,4),SS3(16,4)
      INTEGER SS4(16,4),SS5(16,4),SS6(16,4),SS7(16,4),SS8(16,4)
      DATA SS1/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS2/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS3/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS4/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS5/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS6/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS7/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      DATA SS8/
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
     *  0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15/
      I=II
      J=IN(1)*2+IN(6)+1
      K=IN(2)*8+IN(3)*4+IN(4)*2+IN(5)+1
      IF(I.EQ.1) ASSIGN 1 TO III
      IF(I.EQ.2) ASSIGN 2 TO III
      IF(I.EQ.3) ASSIGN 3 TO III
      IF(I.EQ.4) ASSIGN 4 TO III
      IF(I.EQ.5) ASSIGN 5 TO III
      IF(I.EQ.6) ASSIGN 6 TO III
      IF(I.EQ.7) ASSIGN 7 TO III
      IF(I.EQ.8) ASSIGN 8 TO III
      GO TO III, (1,2,3,4,5,6,7,8)
    1 T=SS1(K,J)
      GO TO 9
    2 T=SS2(K,J)
      GO TO 9
    3 T=SS3(K,J)
      GO TO 9
    4 T=SS4(K,J)
      GO TO 9
    5 T=SS5(K,J)
      GO TO 9
```

```
      GO TO 9
    5 T=SS5(K,J)
      GO TO 9
    6 T=SS6(K,J)
      GO TO 9
    7 T=SS7(K,J)
      GO TO 9
    8 T=SS8(K,J)
    9 DO 10 I1=1,4
      J=5-I1
      Q=T/2
      R=T-2*Q
      OUT(J)=0
      IF(R.EQ.1) OUT(J)=1
      T=Q
   10 CONTINUE
      RETURN
      END
```

附　錄　2－1

$D_1$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 4 | 4 |
| 49 | 75 | 155 | 208 | 327 | 465 | 595 | 757 | 891 | 975 | 973 | 999 | 855 | 722 | 597 | 457 | 343 | 212 | 141 | 71 |
| 11 | 6 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_1$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5040 | 4985 | 4939 | 5022 | 5027 | 4963 | 5059 | 4950 | 5069 | 5031 | 4928 | 5007 | 4979 | 5001 | 5019 | 4959 | 5044 | 4986 | 4960 | 4977 |
| 5074 | 5007 | 4919 | 4918 | 5129 | 5014 | 4955 | 4987 | 5078 | 4890 | 4987 | 5057 | 5105 | 5074 | 4905 | 5047 | 4990 | 5080 | 4901 | 5026 |
| 5019 | 5029 | 5019 | 5004 | 5054 | 4910 | 4960 | 4979 | 5099 | 4972 | 5044 | 4921 | 4892 | 5062 | 4957 | 5019 | 4996 | 4998 | 4956 | 4902 |

$D_2$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 9 |
| 42 | 95 | 120 | 216 | 313 | 461 | 588 | 802 | 908 | 950 | 1014 | 978 | 869 | 753 | 583 | 443 | 306 | 215 | 140 | 67 |
| 4 | 2 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_2$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5000 | 4989 | 5051 | 4909 | 4952 | 4997 | 5023 | 4986 | 5021 | 4940 | 4958 | 5006 | 5074 | 5005 | 4975 | 4909 | 5051 | 5009 | 5084 | 5003 |
| 4933 | 4980 | 5027 | 5017 | 5015 | 5046 | 5029 | 4916 | 5025 | 5007 | 4951 | 5077 | 5013 | 4978 | 5008 | 4965 | 4994 | 5054 | 4967 | 5107 |
| 5040 | 4835 | 4981 | 4906 | 5019 | 4945 | 5040 | 5041 | 4948 | 4906 | 4985 | 4932 | 5076 | 5079 | 4999 | 4913 | 5000 | 5005 | 4957 | 4967 |

$D_3$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 4 |
| 26 | 76 | 135 | 223 | 314 | 459 | 592 | 725 | 905 | 949 | 990 | 980 | 891 | 778 | 643 | 477 | 303 | 209 | 128 | 72 |
| 9 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_3$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5001 | 5028 | 5051 | 5022 | 4992 | 4890 | 5065 | 4970 | 4976 | 4990 | 4976 | 5035 | 4994 | 5062 | 5017 | 5011 | 5016 | 4932 | 5006 | 4967 |
| 5012 | 5047 | 4990 | 5055 | 4989 | 5022 | 5016 | 5028 | 5025 | 4979 | 5064 | 4996 | 5004 | 5013 | 4963 | 4899 | 5130 | 4938 | 5031 | 5050 |
| 4959 | 5005 | 5043 | 5038 | 4970 | 4994 | 4984 | 5017 | 5017 | 4948 | 4888 | 5021 | 5004 | 4970 | 5031 | 5087 | 4986 | 5074 | 4987 | 4943 |

附錄 2-2

$D_1$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 15 | 28 | 41 | 62 | 87 | 120 | 143 | 179 | 187 | 198 | 195 | 178 | 150 | 125 | 99 | 68 | 39 | 36 | 14 |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_1$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 974 | 1038 | 995 | 995 | 972 | 1029 | 1016 | 993 | 1017 | 984 | 1037 | 973 | 967 | 998 | 1055 | 990 | 997 | 985 | 1005 | 1003 |
| 1016 | 998 | 991 | 1024 | 1007 | 970 | 1036 | 1007 | 1002 | 1030 | 969 | 985 | 1015 | 1025 | 982 | 985 | 1033 | 991 | 1027 | 1032 |
| 993 | 1032 | 992 | 1014 | 995 | 1023 | 1020 | 994 | 962 | 1013 | 1009 | 989 | 1037 | 1018 | 998 | 977 | 1030 | 987 | 983 | 1011 |

$D_2$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| 3 | 13 | 28 | 45 | 60 | 99 | 104 | 159 | 172 | 184 | 187 | 197 | 185 | 167 | 131 | 96 | 62 | 41 | 27 | 16 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_2$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 989 | 1002 | 1000 | 1029 | 1019 | 1020 | 993 | 1018 | 1013 | 991 | 995 | 1010 | 973 | 1022 | 1003 | 968 | 1012 | 966 | 1006 | 993 |
| 1042 | 978 | 1002 | 966 | 1023 | 1023 | 977 | 1060 | 999 | 975 | 1016 | 1025 | 972 | 989 | 1013 | 954 | 966 | 1026 | 1021 | 1019 |
| 1012 | 1016 | 958 | 1006 | 1044 | 1024 | 1033 | 984 | 987 | 1020 | 979 | 981 | 1012 | 1012 | 1004 | 1017 | 1001 | 1044 | 1016 | 993 |

$D_3$ :

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 |
| 6 | 11 | 29 | 40 | 75 | 82 | 123 | 151 | 199 | 184 | 189 | 199 | 189 | 152 | 92 | 90 | 74 | 50 | 27 | 14 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_3$ :

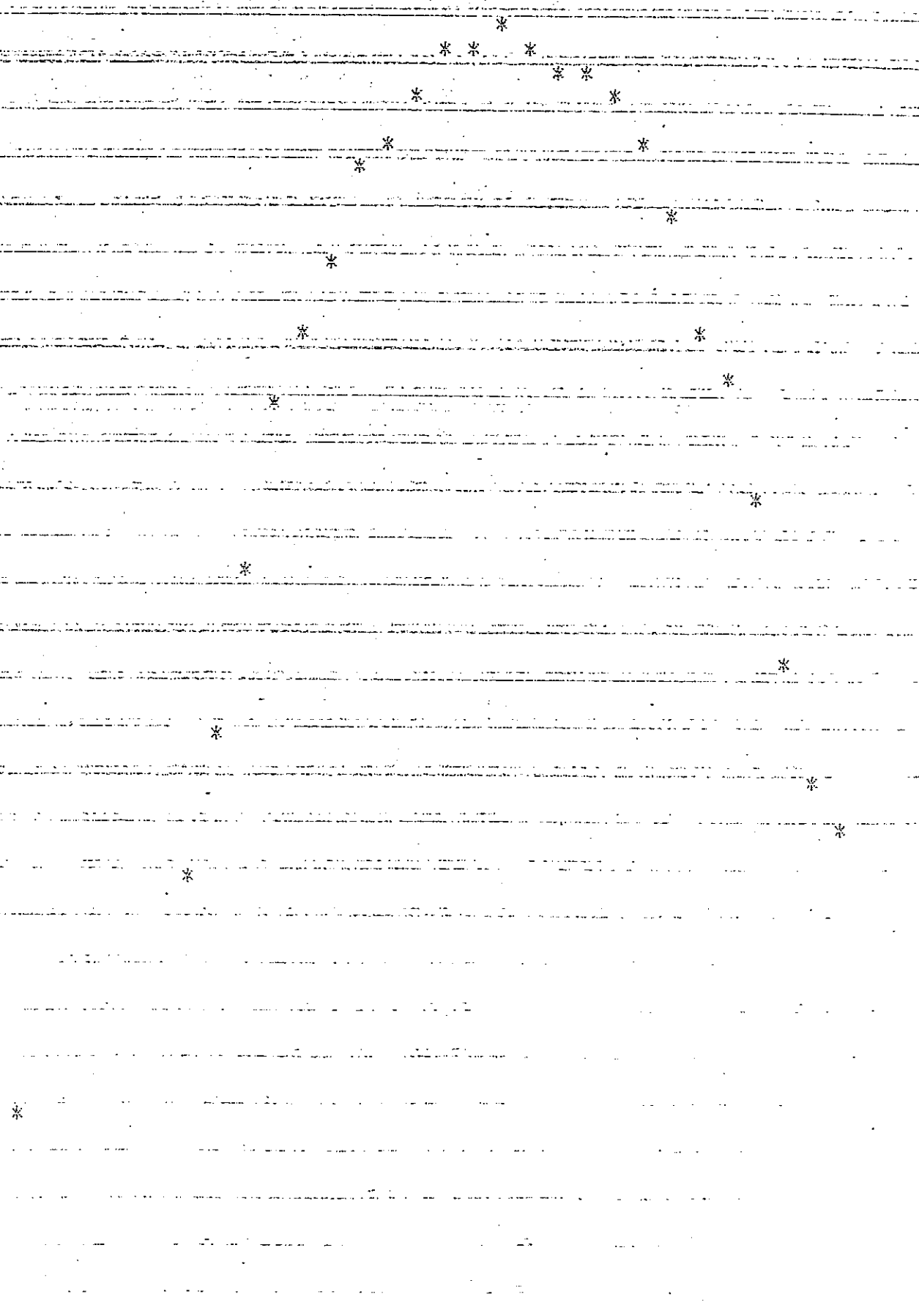
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1005 | 941 | 998 | 990 | 994 | 1013 | 983 | 992 | 978 | 993 | 1033 | 991 | 1011 | 988 | 1041 | 1000 | 989 | 985 | 1012 | 1032 |
| 987 | 983 | 1012 | 1007 | 1009 | 1009 | 943 | 1010 | 1018 | 1006 | 995 | 979 | 995 | 1019 | 980 | 999 | 993 | 1012 | 1009 | 996 |
| 996 | 1012 | 994 | 998 | 993 | 979 | 1034 | 1012 | 1031 | 1011 | 1006 | 994 | 985 | 968 | 1028 | 987 | 1025 | 1016 | 996 | 1022 |

MEAN = 32.15000 STANDARD DEVIATION = 4.01192 RELATIVE STANDARD DEVIATION = 12.47875%

SKEWNESS = 0.00347 CURTOSIS = 2.92452 NON-ZERO RANGE ---- LOWER BIT #21 - UPPER BI

TA FILE NAME (2A4) >DES2,DAT

CORD NUMBER (I3) -> 25



```
     THE FOLLOWING RESULT IS FOR CHANGING BIT TEST

                                                        附錄 2-3
PLEASE READ THE NUMBER OF TEXTS : (I4)>1000

PLEASE ENTER THE BIT POSITIONS TO BE CHANGED:(I2)>37

PLEASE ENTER THE BIT POSITIONS TO BE CHANGED:(I2)>42

PLEASE ENTER THE BIT POSITIONS TO BE CHANGED:(I2)>

          CHANGE THE VALUE AT FOLLOWING BIT POSITION
 37 42

D[ 0]=  0 D[ 1]=  0 D[ 2]=  0 D[ 3]=  0 D[ 4]=  0 D[ 5]=  0 D[ 6]=  0 D[ 7]=  0 D[ 8]=  0 D[ 9]=  0 D[10]=
D[13]=  0 D[14]=  0 D[15]=  0 D[16]=  0 D[17]=  0 D[18]=  2 D[19]=  0 D[20]=  1 D[21]=  1 D[22]=  2 D[23]=
D[26]= 27 D[27]= 35 D[28]= 63 D[29]= 68 D[30]= 88 D[31]=105 D[32]=112 D[33]=105 D[34]= 91 D[35]= 72 D[36]=
D[39]= 27 D[40]=  7 D[41]=  6 D[42]=  2 D[43]=  2 D[44]=  1 D[45]=  0 D[46]=  1 D[47]=  0 D[48]=  0 D[49]=
D[52]=  0 D[53]=  0 D[54]=  0 D[55]=  0 D[56]=  0 D[57]=  0 D[58]=  0 D[59]=  0 D[60]=  0 D[61]=  0 D[62]=

B[ 1]=493  B[ 2]=532  B[ 3]=506  B[ 4]=490  B[ 5]=478  B[ 6]=470  B[ 7]=510  B[ 8]=501
B[ 9]=539  B[10]=492  B[11]=503  B[12]=497  B[13]=502  B[14]=503  B[15]=506  B[16]=490
B[17]=512  B[18]=479  B[19]=502  B[20]=508  B[21]=515  B[22]=509  B[23]=503  B[24]=515
B[25]=508  B[26]=502  B[27]=524  B[28]=498  B[29]=519  B[30]=493  B[31]=490  B[32]=466
```

PLEASE READ THE NUMBER OF TEXTS : (I4)>1000

附錄 2-4

PLEASE ENTER THE BIT POSITION TO BE CHANGED:(I2)> 1

PLEASE ENTER THE BIT POSITION TO BE CHANGED:(I2)> 2

PLEASE ENTER THE BIT POSITION TO BE CHANGED:(I2)> 3

PLEASE ENTER THE BIT POSITION TO BE CHANGED:(I2)> 4

PLEASE ENTER THE BIT POSITION TO BE CHANGED:(I2)>

CHANGE THE VALUE AT FOLLOWING BIT POSITION
1 2 3 4

GROUP TYPE: 1 2 3 4

DESCNTC 1]= 73 DESCNTC 2]= 56 DESCNTC 3]= 67 DESCNTC 4]= 66 DESCNTC 5]= 58 DESCNTC 6]= 52 DESCNTC 7]=

DESCNTC 9]= 74 DESCNTC10]= 59 DESCNTC11]= 63 DESCNTC12]= 71 DESCNTC13]= 73 DESCNTC14]= 73 DESCNTC15]=

GROUP TYPE: 2 3 4 5

DESCNTC 1]= 62 DESCNTC 2]= 48 DESCNTC 3]= 73 DESCNTC 4]= 65 DESCNTC 5]= 72 DESCNTC 6]= 56 DESCNTC 7]=

DESCNTC 9]= 64 DESCNTC10]= 61 DESCNTC11]= 52 DESCNTC12]= 70 DESCNTC13]= 69 DESCNTC14]= 62 DESCNTC15]=

GROUP TYPE: 3 4 5 6

DESCNTC 1]= 54 DESCNTC 2]= 63 DESCNTC 3]= 63 DESCNTC 4]= 62 DESCNTC 5]= 75 DESCNTC 6]= 58 DESCNTC 7]=

附　錄　2－5

```
     THE FOLLOWING RESULT IS FOR CIRCULARLY SHIFTING BITS TEST

 PLEASE ENTER THE NUMBER OF TEXTS!(I4)>1000

 PLEASE ENTER THE NUMBER OF SHIFTS!(I2)> 7

          SHIFT  7 BITS

 D[ 0]=  0 D[ 1]=  0 D[ 2]=  0 D[ 3]=  0 D[ 4]=  0 D[ 5]=  0 D[ 6]=  0 D[ 7]=  0 D[ 8]=  0 D[ 9]=  0 D[1
 D[13]=  0 D[14]=  0 D[15]=  0 D[16]=  0 D[17]=  0 D[18]=  1 D[19]=  0 D[20]=  1 D[21]=  3 D[22]=  6 D[2
 D[26]= 28 D[27]= 50 D[28]= 51 D[29]= 80 D[30]=100 D[31]= 99 D[32]=100 D[33]= 99 D[34]= 83 D[35]= 78 D[3
 D[39]= 22 D[40]= 12 D[41]=  7 D[42]=  2 D[43]=  3 D[44]=  0 D[45]=  0 D[46]=  0 D[47]=  0 D[48]=  0 D[4
 D[52]=  0 D[53]=  0 D[54]=  0 D[55]=  0 D[56]=  0 D[57]=  0 D[58]=  0 D[59]=  0 D[60]=  0 D[61]=  0 D[6


 B[ 1]=509  B[ 2]=466  B[ 3]=496  B[ 4]=499  B[ 5]=490  B[ 6]=517  B[ 7]=499  B[ 8]=520
 B[ 9]=497  B[10]=496  B[11]=494  B[12]=515  B[13]=493  B[14]=496  B[15]=487  B[16]=497
 B[17]=514  B[18]=494  B[19]=510  B[20]=501  B[21]=512  B[22]=500  B[23]=489  B[24]=465
 B[25]=504  B[26]=511  B[27]=466  B[28]=520  B[29]=483  B[30]=514  B[31]=506  B[32]=495
 B[33]=504  B[34]=492  B[35]=509  B[36]=506  B[37]=500  B[38]=507  B[39]=478  B[40]=479
```

```
      THE FOLLOWING RESULT IS FOR CIRCULARLY SHIFTING BITS TEST

 PLEASE ENTER THE NUMBER OF TEXTS:(I4)>1000

 PLEASE ENTER THE NUMBER OF SHIFTS:(I2)> 9

          SHIFT  9 BITS


 GROUP TYPE: 1 2 3 4

DESCNTC 1]= 78  DESCNTC 2]= 68  DESCNTC 3]= 73  DESCNTC 4]= 60  DESCNTC 5]= 69  DESCNTC 6]= 63  DESCNTC
DESCNTC 9]= 53  DESCNTC10]= 69  DESCNTC11]= 57  DESCNTC12]= 57  DESCNTC13]= 52  DESCNTC14]= 60  DESCNTC1

 GROUP TYPE: 2 3 4 5

DESCNTC 1]= 74  DESCNTC 2]= 64  DESCNTC 3]= 64  DESCNTC 4]= 59  DESCNTC 5]= 64  DESCNTC 6]= 57  DESCNTC
DESCNTC 9]= 57  DESCNTC10]= 60  DESCNTC11]= 53  DESCNTC12]= 56  DESCNTC13]= 56  DESCNTC14]= 62  DESCNTC1

 GROUP TYPE: 3 4 5 6

DESCNTC 1]= 73  DESCNTC 2]= 63  DESCNTC 3]= 60  DESCNTC 4]= 71  DESCNTC 5]= 60  DESCNTC 6]= 56  DESCNTC
DESCNTC 9]= 60  DESCNTC10]= 61  DESCNTC11]= 59  DESCNTC12]= 60  DESCNTC13]= 58  DESCNTC14]= 64  DESCNTC1

 GROUP TYPE: 4 5 6 7

DESCNTC 1]= 63  DESCNTC 2]= 68  DESCNTC 3]= 62  DESCNTC 4]= 70  DESCNTC 5]= 61  DESCNTC 6]= 65  DESCNTC
```

PLEASE ENTER THE NUMBER OF TEST DATA : (I4)1000

THE FOLLOWING RESULT IS FOR FIXING FIRST 32 BITS AFTER INITIAL PERMUTATION TEXT

COUNT THE NUMBER OF SAME BITS OF FIRST 32 BITS WITH FIRST CIPHERTEXT

D1[ 0]=  0 D1[ 1]=  0 D1[ 2]=  0 D1[ 3]=  0 D1[ 4]=  0 D1[ 5]=  0 D1[ 6]=  0 D1[ 7]=  3 D1[ 8]=  1 D1[ 9]

D1[11]= 27 D1[12]= 66 D1[13]= 67 D1[14]=113 D1[15]=130 D1[16]=120 D1[17]=128 D1[18]=137 D1[19]= 90 D1[20]

D1[22]= 11 D1[23]=  4 D1[24]=  1 D1[25]=  0 D1[26]=  0 D1[27]=  0 D1[28]=  0 D1[29]=  0 D1[30]=  0 D1[31]

COUNT THE NUMBER OF SAME BITS OF LAST 32 BITS WITH FIRST CIPHERTEXT

D2[ 0]=  0 D2[ 1]=  0 D2[ 2]=  0 D2[ 3]=  0 D2[ 4]=  0 D2[ 5]=  0 D2[ 6]=  0 D2[ 7]=  1 D2[ 8]=  3 D2[ 9]

D2[11]= 40 D2[12]= 57 D2[13]= 69 D2[14]=129 D2[15]=127 D2[16]=133 D2[17]=134 D2[18]=110 D2[19]= 73 D2[20]

D2[22]= 18 D2[23]= 11 D2[24]=  0 D2[25]=  2 D2[26]=  1 D2[27]=  0 D2[28]=  0 D2[29]=  0 D2[30]=  0 D2[31]

COUNT THE NUMBER OF SAME BITS WITH FIRST CIPHERTEXT

```
D[ 0]=  0 D[ 1]=  0 D[ 2]=  0 D[ 3]=  0 D[ 4]=  0 D[ 5]=  0 D[ 6]=  0 D[ 7]=  0 D[ 8]=  0 D[ 9]=  0 D[10]=  0
D[13]=  0 D[14]=  0 D[15]=  0 D[16]=  0 D[17]=  0 D[18]=  0 D[19]=  0 D[20]=  3 D[21]=  2 D[22]=  5 D[23]=  9
D[26]= 28 D[27]= 48 D[28]= 57 D[29]= 81 D[30]= 82 D[31]=106 D[32]= 89 D[33]=110 D[34]= 74 D[35]= 76 D[36]= 58
D[39]= 16 D[40]=  7 D[41]= 13 D[42]=  4 D[43]=  2 D[44]=  1 D[45]=  0 D[46]=  0 D[47]=  0 D[48]=  0 D[49]=  0
D[52]=  0 D[53]=  0 D[54]=  0 D[55]=  0 D[56]=  0 D[57]=  0 D[58]=  0 D[59]=  0 D[60]=  0 D[61]=  0 D[62]=  0
```

COUNT THE NUMBER OF DATA HAVING THE SAME VALUE AT I-TH BIT WITH FIRST CIPHERTEXT

```
BITCNT[ 1]=521  BITCNT[ 2]=527  BITCNT[ 3]=511  BITCNT[ 4]=514  BITCNT[ 5]=489  BITCNT[ 6]=504  BITCNT[ 7]=52
BITCNT[ 9]=487  BITCNT[10]=500  BITCNT[11]=489  BITCNT[12]=498  BITCNT[13]=494  BITCNT[14]=488  BITCNT[15]=51
BITCNT[17]=502  BITCNT[18]=502  BITCNT[19]=524  BITCNT[20]=504  BITCNT[21]=516  BITCNT[22]=494  BITCNT[23]=51
BITCNT[25]=511  BITCNT[26]=484  BITCNT[27]=528  BITCNT[28]=503  BITCNT[29]=470  BITCNT[30]=487  BITCNT[31]=49
BITCNT[33]=495  BITCNT[34]=488  BITCNT[35]=500  BITCNT[36]=518  BITCNT[37]=500  BITCNT[38]=514  BITCNT[39]=51
BITCNT[41]=470  BITCNT[42]=504  BITCNT[43]=517  BITCNT[44]=486  BITCNT[45]=498  BITCNT[46]=505  BITCNT[47]=51
BITCNT[49]=503  BITCNT[50]=499  BITCNT[51]=498  BITCNT[52]=511  BITCNT[53]=495  BITCNT[54]=507  BITCNT[55]=50
BITCNT[57]=484  BITCNT[58]=481  BITCNT[59]=483  BITCNT[60]=470  BITCNT[61]=500  BITCNT[62]=483  BITCNT[63]=52
```

```
PLEASE ENTER THE NUMBER OF TEST DATA : (I4)1000

THE FOLLOWING RESULT IS FOR FIXING FIRST 32 BITS AFTER INITIAL PERMUTATION TEXT

GROUP TYPE: 1 2 3 4
DESCNTC 1]= 71  DESCNTC 2]= 69  DESCNTC 3]= 57  DESCNTC 4]= 61  DESCNTC 5]= 51  DESCNTC 6]= 53  DESCNTC 7
DESCNTC 9]= 59  DESCNTC10]= 59  DESCNTC11]= 54  DESCNTC12]= 65  DESCNTC13]= 72  DESCNTC14]= 77  DESCNTC15

GROUP TYPE: 2 3 4 5
DESCNTC 1]= 67  DESCNTC 2]= 63  DESCNTC 3]= 62  DESCNTC 4]= 60  DESCNTC 5]= 63  DESCNTC 6]= 55  DESCNTC 7
DESCNTC 9]= 69  DESCNTC10]= 68  DESCNTC11]= 69  DESCNTC12]= 63  DESCNTC13]= 72  DESCNTC14]= 75  DESCNTC15

GROUP TYPE: 3 4 5 6
DESCNTC 1]= 65  DESCNTC 2]= 57  DESCNTC 3]= 67  DESCNTC 4]= 69  DESCNTC 5]= 58  DESCNTC 6]= 46  DESCNTC 7
DESCNTC 9]= 62  DESCNTC10]= 63  DESCNTC11]= 57  DESCNTC12]= 67  DESCNTC13]= 74  DESCNTC14]= 65  DESCNTC15

GROUP TYPE: 4 5 6 7
DESCNTC 1]= 81  DESCNTC 2]= 59  DESCNTC 3]= 62  DESCNTC 4]= 59  DESCNTC 5]= 51  DESCNTC 6]= 51  DESCNTC
DESCNTC 9]= 54  DESCNTC10]= 57  DESCNTC11]= 61  DESCNTC12]= 63  DESCNTC13]= 64  DESCNTC14]= 67  DESCNTC1
```

```
PLEASE ENTER THE NUMBER OF TEST DATA : (I4)1000

THE FOLLOWING RESULT IS FOR FIXING LAST 32 BITS AFTER INITIAL PERMUTATION TEXT

        COUNT THE NUMBER OF SAME BITS OF FIRST 32 BITS WITH FIRST CIPHERTEXT

GROUP TYPE: 1 2 3 4
DESCNTC 1]= 70  DESCNTC 2]= 63  DESCNTC 3]= 70  DESCNTC 4]= 60  DESCNTC 5]= 58  DESCNTC 6]= 58  DESCNTC 7]=
DESCNTC 9]= 51  DESCNTC10]= 56  DESCNTC11]= 68  DESCNTC12]= 63  DESCNTC13]= 73  DESCNTC14]= 76  DESCNTC15]=

GROUP TYPE: 2 3 4 5
DESCNTC 1]= 63  DESCNTC 2]= 67  DESCNTC 3]= 62  DESCNTC 4]= 69  DESCNTC 5]= 47  DESCNTC 6]= 55  DESCNTC 7]=
DESCNTC 9]= 53  DESCNTC10]= 50  DESCNTC11]= 60  DESCNTC12]= 61  DESCNTC13]= 73  DESCNTC14]= 62  DESCNTC15]=

GROUP TYPE: 3 4 5 6
DESCNTC 1]= 58  DESCNTC 2]= 66  DESCNTC 3]= 45  DESCNTC 4]= 72  DESCNTC 5]= 55  DESCNTC 6]= 53  DESCNTC 7]=
DESCNTC 9]= 69  DESCNTC10]= 53  DESCNTC11]= 60  DESCNTC12]= 57  DESCNTC13]= 59  DESCNTC14]= 69  DESCNTC15]=

GROUP TYPE: 4 5 6 7
DESCNTC 1]= 60  DESCNTC 2]= 74  DESCNTC 3]= 55  DESCNTC 4]= 73  DESCNTC 5]= 59  DESCNTC 6]= 66  DESCNTC 7]
```

改變第 15 個位元的測試結果；$D_1$；$D_2$；$D_3$ 的統計量及向量 $D_2$ 的圖型（Y 軸取 LOG）

$D_1$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 17 | 31 | 30 | 73 | 96 | 119 | 179 | 169 | 190 | 202 | 178 | 169 | 158 | 122 | 72 | 84 | 39 | 28 | 15 |
| 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_1$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 987 | 977 | 991 | 998 | 1008 | 1008 | 1004 | 996 | 969 | 958 | 996 | 1019 | 999 | 979 | 998 | 1039 | 966 | 1015 | 984 | 990 |
| 1035 | 1023 | 1002 | 1002 | 1019 | 1031 | 1007 | 979 | 1024 | 1020 | 1020 | 1047 | 975 | 993 | 989 | 970 | 1009 | 1014 | 989 | 1012 |
| 1009 | 995 | 981 | 984 | 975 | 1004 | 994 | 994 | 993 | 985 | 1001 | 1032 | 983 | 998 | 1010 | 1004 | 987 | 998 | 1005 | 996 |

$D_2$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 10 | 15 | 31 | 44 | 71 | 96 | 117 | 161 | 155 | 182 | 177 | 197 | 179 | 148 | 127 | 110 | 66 | 44 | 23 | 19 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_2$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 999 | 979 | 961 | 1016 | 998 | 1015 | 1037 | 1009 | 997 | 1028 | 996 | 1010 | 992 | 984 | 994 | 1024 | 1013 | 995 | 995 | 980 |
| 1028 | 1025 | 1019 | 974 | 1049 | 992 | 1030 | 1022 | 1021 | 968 | 1034 | 989 | 989 | 1015 | 1021 | 981 | 990 | 995 | 1003 | 998 |
| 1010 | 1004 | 1003 | 977 | 1018 | 972 | 1017 | 1006 | 1031 | 999 | 1028 | 978 | 980 | 1008 | 990 | 987 | 974 | 983 | 980 | 990 |

$D_3$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 12 | 10 | 35 | 39 | 75 | 86 | 96 | 162 | 170 | 185 | 195 | 184 | 178 | 164 | 140 | 78 | 66 | 42 | 34 | 14 |
| 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$B_3$：

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 961 | 1001 | 1040 | 1015 | 1012 | 974 | 1016 | 1000 | 1026 | 967 | 957 | 1002 | 974 | 1028 | 1010 | 1015 | 1004 | 1011 | 1000 | 998 |
| 1014 | 992 | 1002 | 1026 | 993 | 973 | 1004 | 1040 | 994 | 990 | 1026 | 997 | 1004 | 1000 | 1034 | 965 | 991 | 1000 | 1012 | 1015 |
| 1008 | 1026 | 968 | 1009 | 1032 | 983 | 995 | 995 | 988 | 970 | 1013 | 1016 | 994 | 1035 | 997 | 983 | 1048 | 949 | 990 | 1001 |

改變第1位元及第8位元測試的6個向量

```
ENTER KEY : (64I1)>1111100000110000:1101110100010001000110001000110000110101011100
JT7 --- STOP

>RUN DESWRITE

PLEASE ENTER THE DATA FILE NAME (20A2)>HA3.DAT
RECORD NUMBER :  1      ....(HA3.DAT)
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0    0    0    0    0    0 2000    0    0    0    0    0

RECORD NUMBER :  2      ....(HA3.DAT)
2000    0 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000    0 2000 2000 2000 2000
2000 2000 2000 2000 2000    0 2000 2000 2000 2000 2000 2000 2000    0 2000 2000 2000    0 2000 2000 2000 2
2000 2000 2000    0 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000    0

RECORD NUMBER :  3      ....(HA3.DAT)
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    5    0    3    0   20
  51    0   77    0  147    0  225    0  331    0  318    0  270    0  206    0  156    0  114    0   50
  15    0    5    0    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

RECORD NUMBER :  4      ....(HA3.DAT)
1009 1015 1046 1003  946  997  987 1036 1023 1014 1000  991 1016 1017  999 1003 1009  990 1000  995  992
1007 1009  973 1015 1004 1024 1023  981 1033 1000 1009  985 1046  997  946  997  987  964 1023 1014 1000
1016 1017  999  997  991  990 1000  995  992  952 1007 1009  973 1015 1004  976 1023  981 1033 1008    0

RECORD NUMBER :  5      ....(HA3.DAT)
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    4    0    7    0   23
  37    0   89    0  152    0  191    0  300    0  306    0  350    0  205    0  171    0   85    0   51
  20    0    7    0    1    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0

RECORD NUMBER :  6      ....(HA3.DAT)
 998 1002  992  992 1005 1005  992  992 1015 1015  996  996  992  992 1014 1014 1027  973 1017 1017  969
 996  996 1067 1067 1024  976 1012 1012 1000 1000 1020 1020  989 1011  974  974 1027  973 1002 1002  989
1021 1021 1002  998 1010 1010 1034 1034  993  993 1000 1000 1004 1004 1050 1050 1044 1044  953  953    0
```

錄 2－12 第 15 道的 $D_1$, $D_2$, $D_3$

```
RECORD NUMBER =  43          (8664 DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0   27    0   91    0  159    0  305    0  398    0  392    0  279    0  179    0   93    0   39    0   14    0    5    0    2    0    1    0    0    0    0
    0    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

RECORD NUMBER =  44          (8664 DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    8   12   19   36   55  104  129  163  190  205  194  196  165  122  121   77   74   45   35   12   16    6    3    2    1    0    0    0    0    0    0    0
    1    2    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

RECORD NUMBER =  45          (8664 DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0   25    0   71    0  178    0  296    0  432    0  339    0  305    0  183    0   76    0   31    0   10    0    3    0    1    0    0    0    0    0    0
    0    1    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

附　錄　2－13

改變第1位元及第8位元測試的6個向量$D_1$，$B_1$，$D_2$，$B_2$，$D_3$，$B_3$

```
PLOT10 ...
TT7 --- STOP

>RUN DESWRITE

FILE OF TYPE INT Data FILE Name (2=33PFA3.DAT
RECORD NUMBER :  1          (TEST.DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 2000    0    0    0    0

RECORD NUMBER :  2          (PFA3.DAT)
 2000    0 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000    0 2000 2000 2000 200
 2000 2000 2000 2000 2000    0 2000 2000 2000 2000 2000 2000 2000 2000    0 2000 2000 2000    0 2000 2000 2000
 2000 2000 2000    0 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000

RECORD NUMBER :  3          (PFA3.DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0    7    0
   51    0   47    0  164    0  225    0  270    0  284    0  277    3  250    0  161    0  111    0
   12    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0

RECORD NUMBER :  4          (PFA3.DAT)
 1016 1010 1010 1052  989  996  984  978  989 1014  991 1030  999 1028  996  985 1038  975 1013 1017 10
  965  991 1003  994 1017 1003  998 1041  987 1014 1016  990 1010  968  989  996  984 1022  989 1014  9
  999 1026  986 1013  962  975 1013 1017 1021 1007  963  991 1033  994 1017  997  998 1031  987 1014

RECORD NUMBER :  5          (PFA3.DAT)
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0    7    0
   50    0   47    0  140    0  206    0  221    0  222    0  275    0  204    0  138    0   87    0
   12    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```