โปรแกรมการเข้ารหัสข้อมูล

DATA ENCRYPTION PROGRAM

ผู้จัดทำ

นาย พรวิทย์ ศักดิ์โสภาวิวัฒน์ 321205

นาย ศรัณย์ อนันตธนวนิชย์ 321323

อาจารย์ที่ปรึกษา

รศ.ดร. ฟูศักดิ์ ชีวสุวิทย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปีการศึกษา 2535

โปรแกรมการเข้ารหัสข้อมูล

(Data Encryption Program)

จัดทำโดย

นาย พรวิทย์ ศักดิ์รสภาวิวัฒน์ 32 1205

นาย ศรัณย์ อนันตธนวนิชย์ 32 1323

อาจารย์ที่ปรึกษา

รศ.ดร. พูศักดิ์ ชีวสุวิทย์

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

โปรแกรมการเข้ารหัสข้อมูล
(Data Encryption Program)

จัดทำโดย
นาย พรวิทย์ ศักดิ์รัสภาวิวัฒน์ 32 1205
นาย ศรัณย์ อนันตธนวนิชย์ 32 1323

..... อาจารย์ที่ปรึกษา
(รศ.ดร. พูศักดิ์ ชีวสุวิทย์)

โปรแกรมการเข้ารหัสข้อมูล

ผู้ร่วมโครงงาน

1. นาย พรวิทย์ ศักดิ์โรสภาวิวัฒน์ 32 1205
2. นาย ศรัณย์ อนันตธนวนิชย์ 32 1323

อาจารย์ที่ปรึกษา

รศ.ดร. พูศักดิ์ ชีวสุวิทย์

ปีการศึกษา 2535

บทคัดย่อ

โครงการนี้เป็นการเขียนโปรแกรมเข้ารหัสข้อมูล จุดประสงค์หลักในการเข้ารหัสคือทำการปกป้องข้อมูลให้เป็นความลับ โดยโปรแกรมที่เขียนขึ้นมานี้จะมีความสามารถในการทำการเข้ารหัส และ ถอดรหัส ได้อย่างถูกต้องและมีความเชื่อถือได้ในการทำงานสูง และสามารถทำการเข้ารหัสและถอดรหัสไฟล์ต่าง ๆ ได้ทุกชนิด ไม่เพียงแต่ไฟล์ตัวอักษรเท่านั้น

อัลกอริธึมที่ใช้ในการเข้ารหัส และ ถอดรหัสนี้คือ DES (DATA ENCRYPTION STANDARD) ซึ่งได้รับการยอมรับว่ามีความปลอดภัยในการรักษาข้อมูลสูง เพราะประกอบไปด้วยวิธีคิดที่ซับซ้อน และ ยังต้องใช้ค่าคีย์จำนวน 56 บิท ในการเข้ารหัส และ ถอดรหัสอีกด้วย

สำหรับโปรแกรมที่เขียนขึ้นมานี้ใช้ภาษา C ในการเขียน และใช้คอมไพเลอร์คือ Turbo C version 2.0

Data Encryption Program

Colleague

1. Pornvit Saksobhavivat 32 1205

2. Sarun Anuntathanawanich 32 1323

Advisor

Assoc.Prof.Dr. Fusak Cheevasuvit

Year 1995

ABSTRACT

Nowadays the security of data is very important. This project is made for that reason. The data encryption program uses to protect data that you want it secreted. Encryption and Decryption of this program are easy to use.

The algorithm used in this program, DES (Data Encryption standard), is accepted in high security, complexity, and realiability. It must uses 56 bits key in encryption/decription so it is extremely hard to break.

The program is written in C language and complied by Turbo C Version 2.0

สารบัญ

บทนำ

ในปัจจุบันเป็นยุคแห่งข่าวสารและข้อมูล การดำเนินงานต่างๆจำเป็นต้องอาศัยข่าวสารเป็นอย่างมาก ดังนั้นความลับของข่าวสารหรือข้อมูลที่สำคัญ จึงจำเป็นอย่างยิ่ง การสร้างความเข้าใจระหว่างผู้ที่ส่งสาร กับผู้ที่รับสารเพียงสองฝ่ายนั้นทำได้โดย การเข้ารหัสข้อมูล โครงงานนี้เป็นการสร้างโปรแกรมคอมพิวเตอร์ เพื่อใช้ในการเข้ารหัสข้อมูล ภาษาที่ใช้ก็คือ C ซึ่งเป็นที่ใช้กันแพร่หลายเพราะ ใช้ได้ง่ายทำงานได้รวดเร็ว ส่วนการเข้ารหัสนั้นมีหลายรูปแบบ แบ่งออกเป็นสองพวกใหญ่ๆ คือแบบอนาลอกกับแบบดิจิตอล ในที่นี้จะกล่าวถึงแต่แบบหลัง ซึ่งได้นำมาใช้เนื่องมาจาก แบบดิจิตอลนี้ทำได้ง่ายกว่า มีความปลอดภัยสูงเพราะ สามารถทำให้สลับซับซ้อน ยากแก่การถอดรหัส อัลกอริทึมที่ใช้ในการเขียนโปรแกรมนี้ใช้อัลกอริทึมที่เรียกว่า DES (data encription standard) อัลกอริทึมนี้ยากต่อการถอดรหัสตรงตามจุดประสงค์ของโปรแกรม รายละเอียดต่างๆของทฤษฎีที่ใช้จะกล่าวอย่างละเอียดอีกครั้งในบทต่อไป

วัตถุประสงค์ และ ขอบเขต ของโครงงาน

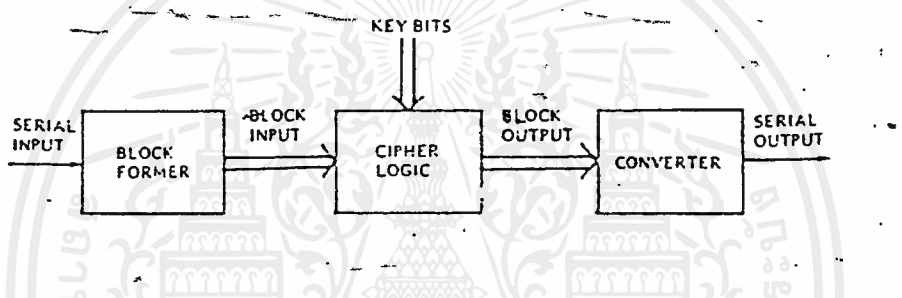
โครงงานนี้เป็นการสร้างโปรแกรมคอมพิวเตอร์สำเร็จรูป (software) เพื่อใช้ในการเข้ารหัสข้อมูลโดยเน้นที่ ไฟล์ต่างๆที่อยู่ภายในดิสค์ เช่น ดาต้าเบสไฟล์ (data base) ซอสไฟล์ (source file) อันเป็นข้อมูลที่มีค่ายิ่งในทางธุรกิจ เราออกแบบโปรแกรมเพื่อให้ใช้ได้ง่าย และมีข้อจำกัดทางฮาร์ดแวร์น้อยที่สุด สามารถใช้ได้โดยบุคคลทั่วไป และเครื่องคอมพิวเตอร์ทั่วๆไป แต่ที่สำคัญคือเราสร้างโปรแกรมจาก การเข้ารหัสแบบ DES ซึ่งจะทำการถอดรหัสได้ยาก ดังนั้นจึงมีความปลอดภัยสูง นอกจากนั้นยังสามารถที่จะพัฒนา ฟังก์ชั่นการทำงานเพื่อให้ใช้ในการส่งหรือรับข้อมูลก็ได้ โครงงานนี้นอกจากจะทำการเข้ารหัสแล้ว ก็มีการถอดรหัสนั้นๆออกมาด้วยเพื่อให้ได้ข้อมูลที่ใส่เข้าไปครั้งแรก และยังเป็นการตรวจสอบการทำงานตัวเองว่าเข้ารหัสนั้นๆถูกต้องหรือไม่

บทที่1 หลักการเบื้องต้นของการเข้ารหัส

ในที่นี้จะแสดงเฉพาะ ส่วนของการเข้ารหัสแบบดิจิตอลเท่านั้น

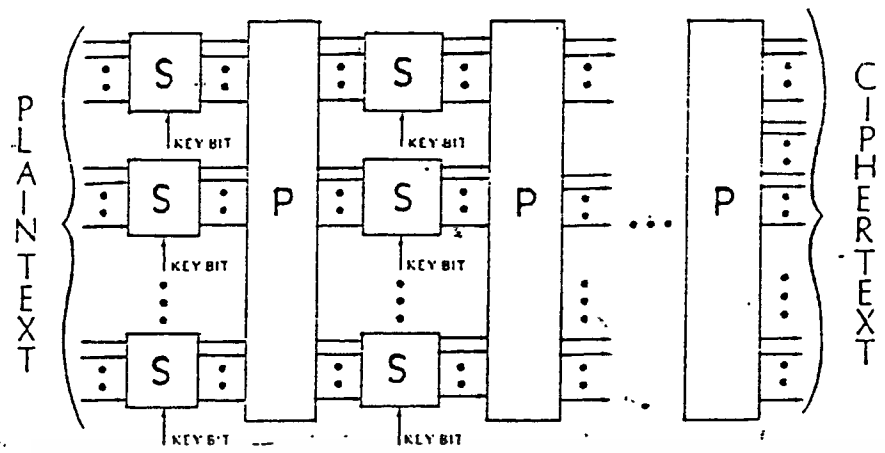
1 บล็อกไซเฟอร์ (block cipher)

บล็อกไซเฟอร์เป็นวิธีการเข้ารหัสแบบดิจิตอลอย่างง่าย จึงมีรอกาสที่จะถูกแกะออกมาได้ด้วยการวิเคราะห์ความถี่ของบล็อก ดังนั้นเพื่อเพิ่มความปลอดภัยต้องใช้บล็อกที่มีขนาดใหญ่ ลักษณะทั่วไปของการเข้ารหัสและถอดรหัสวิธีนี้ แสดงให้เห็นในรูปที่ 1.1

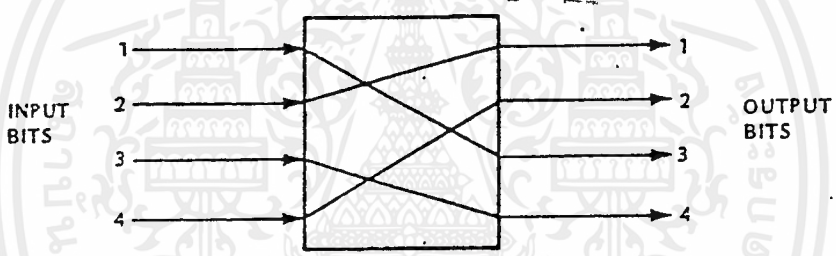


รูปที่ 1.1 รูปทั่วไปของการระบบการเข้าและถอดรหัสแบบบล็อก

การกำหนดให้บล็อกมีขนาดใหญ่นั้นจำนวนบิทของคีย์ที่ใช้ต้องมีจำนวนมากขึ้นด้วย ถ้าขนาดของบล็อกมีค่าเท่ากับ n Bits จะมีบล็อกรูปแบบต่าง ๆ กัน 2^n block เมื่อผ่านการเข้ารหัสแล้ว จะได้บล็อกที่มีรูปแบบต่างกัน 2^n block ดังนั้นจำนวนบิทที่ใช้คือ $[\log (2^n !)] + 1$ เพื่อลดจำนวนคีย์ลงวิธีการหนึ่งก็คือ จะต้องใช้รูปแบบการเข้ารหัสที่แน่นอน ดังตัวอย่างในรูปที่ 1.2 บล็อกของข้อความที่นำมาเข้ารหัสจะถูกเปลี่ยนแปลงไปอย่างต่อเนื่อง สลับกันระหว่างกลุ่ม S boxes และ P boxes ภายใน S boxes จะประกอบด้วย ดิจิตอลลอจิก ซึ่งจะให้ค่าเอาท์พุทเป็น ฟังก์ชั่นบลูลีน ของอินพุทกับค่าคีย์ ส่วน p boxes เป็นการจัดเรียงหรือสับเปลี่ยนบิทของ อินพุท ดังรูปที่ 1.3



รูปที่ 1.2 ระบบการเข้ารหัสแบบบล็อกที่มีหลายสเตจ



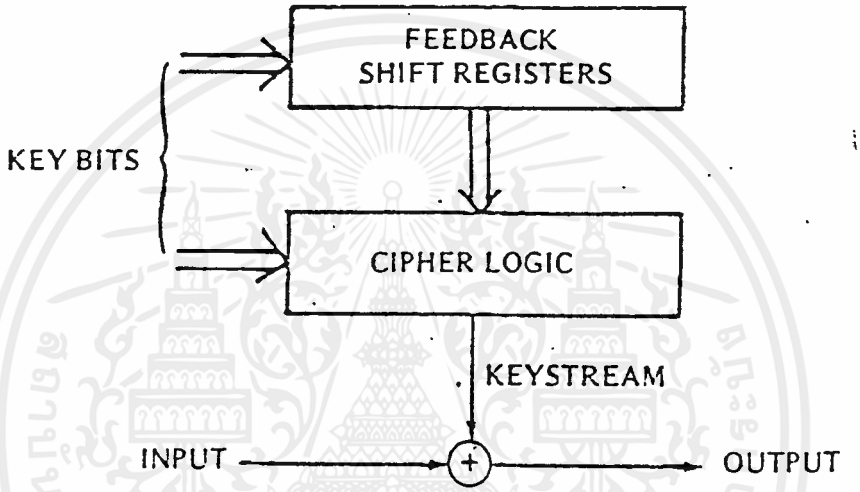
รูปที่ 1.3 การสลับบิทจำนวน 4 บิทของ P box

จำนวนคีย์ทั้งหมดจะเท่ากับจำนวนของ S boxes ทั้งหมด การถอดรหัสก็จะใช้การเรียงย้อนกลับจากการเข้ารหัส ปัญหาใหญ่ของการเข้ารหัสแบบนี้คือเวลาที่ใช้ ซึ่งจะกินเวลามาก

2.ซิงค์โครนัสไซเฟอร์ (synchronous cipher)

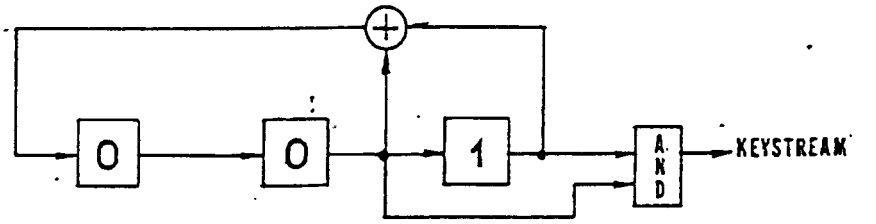
การเข้ารหัสต่อเนื่อง (stream cipher) มีอยู่ 2 ประเภทคือ ประเภทที่ค่าของคีย์สตรีม (key stream) ขึ้นอยู่กับค่าของข้อมูลที่นำมาเข้ารหัส และประเภทที่ค่าของคีย์สตรีมเป็นฟังก์ชันของข้อมูลที่นำมาเข้ารหัส ประเภทแรกนั้นเรียกว่า ซิงโครนัสไซเฟอร์ (Synchronous cipher) เพราะว่ามันต้องการการซิงค์กันระหว่างค่าของคีย์กับข้อมูลอินพุทเพื่อทำการถอดรหัสเป็นไปอย่างถูกต้อง

ค่าของคีย์สตรีมถูกสร้างขึ้นมาอย่างสะเปะสะปะ มีความยาวเท่ากับข้อมูลที่นำมาเข้ารหัส และเพราะว่าค่าคีย์ที่ผ่านไปแล้วจะไม่มีการนำมาใช้อีก การเข้ารหัสวิธีนี้จึงเรียกว่า one-time tape หรือ one-time pad การเปลี่ยนค่าคีย์ให้ถี่ขึ้นหรือยาวขึ้นจะไม่มีผลในการนำไปใช้งานจริง ดังนั้น จะใช้จำนวนบิทของคีย์และฟีดแบคชิฟท์รีจิสเตอร์ในจำนวนที่พอดีเพื่อสร้างค่าคีย์สตรีม รูปแบบในการเข้าและถอดรหัสวิธีนี้ ได้แสดงไว้ในรูปที่ 1.4 ซึ่งเครื่องหมายบวกแสดงถึงการทำ modulo-two

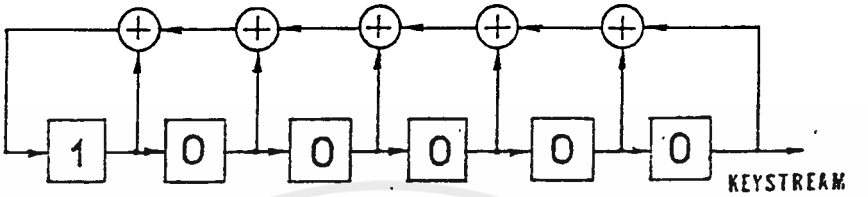


รูปที่ 1.4 รูปแบบทั่วไปของระบบการเข้าและถอดรหัสแบบซิงค์โครนัส

ลิเนียร์ฟีดแบคชิฟท์รีจิสเตอร์ จะสร้างคีย์สตรีมค่าสะเปะสะปะที่มีความยาวต่อเนื่องกันไป แต่อย่างไรก็ตาม โครงสร้างการทำงานแบบลิเนียร์นี้ทำให้ง่ายต่อการที่จะวิเคราะห์หาข้อมูลที่แท้จริงออกมาได้ และยังมีข้อจำกัดของจำนวนชิฟท์รีจิสเตอร์ในการนำไปใช้งาน ดังนั้นจึงเปลี่ยนการทำงานในบางส่วนให้มีการทำงานเป็นแบบนอนลิเนียร์ เช่น ในส่วนของเอาท์พุทหรือส่วนของฟีดแบค เพื่อลดจำนวนของชิฟท์รีจิสเตอร์ให้น้อยลง รูปที่ 1.5 และ 1.6 แสดงให้เห็นถึงการสร้างคีย์สตรีมแบบนอนลิเนียร์และส่วนสมมูลย์ของมัน ค่าเริ่มต้นในชิฟท์รีจิสเตอร์เป็นเลขฐานสอง ในรูปที่ 1.5 a ใช้ลิเนียร์ฟีดแบคชิฟท์รีจิสเตอร์ 1 ตัว และนำเอาท์พุทของทั้งสองส่วนมารวมกันแบบนอน-ลิเนียร์ เพื่อใช้ในการสร้างคีย์สตรีม ในรูปที่ 1.6 a ใช้ลิเนียร์ฟีดแบคชิฟท์รีจิสเตอร์ สองตัว และนำเอาเอาท์พุทของทั้งสองมารวมกันแบบ นอนลิเนียร์สร้างคีย์สตรีม

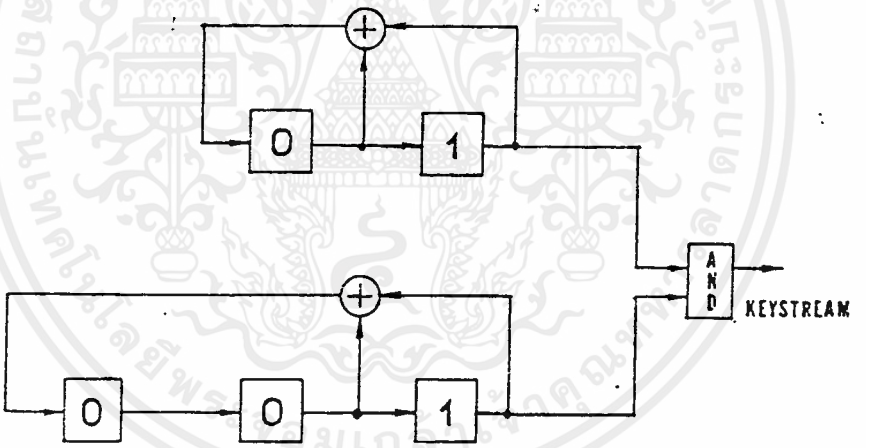


(a)

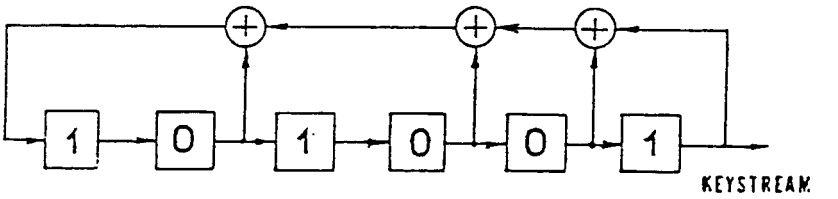


(b)

รูปที่ 1.5 (a) การสร้างคียสตรีมแบบนอน-ลิเนียร์ด้วยฟีดแบคชีฟท์รีจิสเตอร์ 1 ตัว และ (b) เป็นส่วนสมมูลย์แบบลิเนียร์



(a)



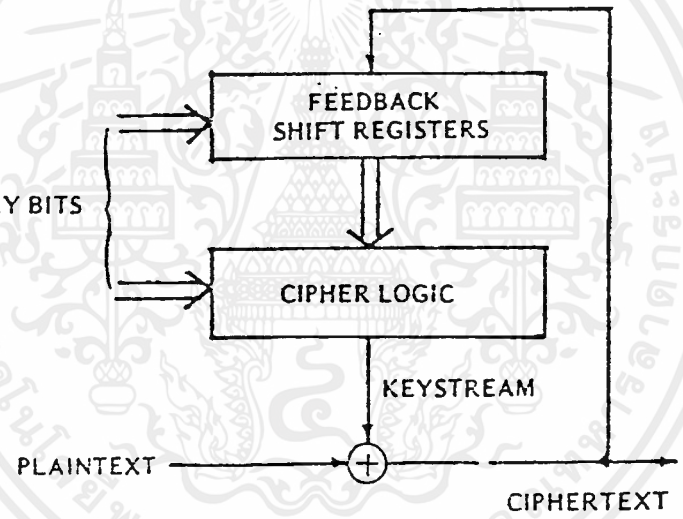
(b)

รูปที่ 1.6 (a) การสร้างคีย์สตรีมแบบนอนลิเนียร์ด้วยฟีดแบคชีฟรีจิสเตอร์ 2 ตัว และ (b) เป็นส่วนสมมูลย์แบบลิเนียร์

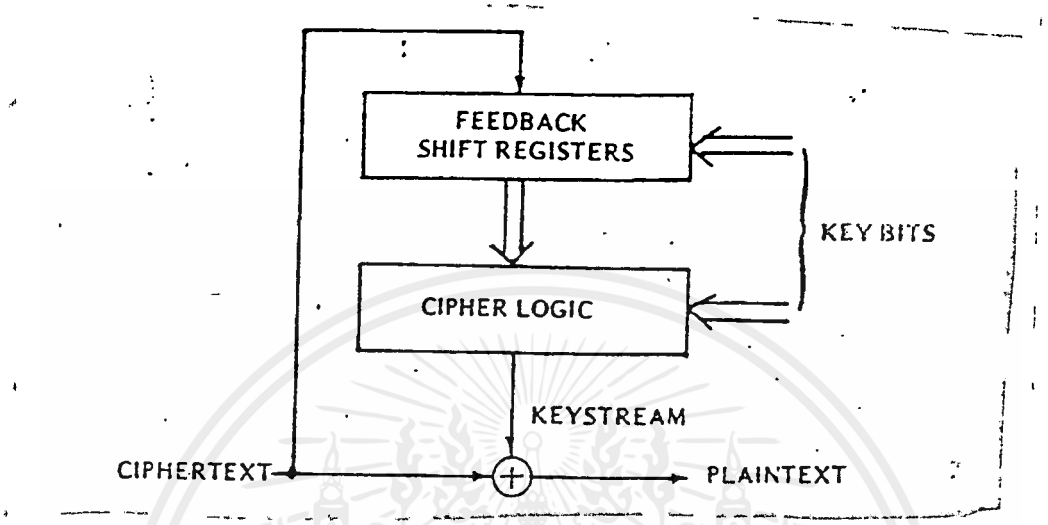
3. ออโตคีย์ ไซเฟอร์ (auto-key cipher)

รหัสออโตคีย์นี้เป็นการนำข้อมูลที่ผ่านมาแล้ว ช่วยในการสร้างคีย์ทำให้มีความปลอดภัยสูงมากระบบการทำงานของการเข้ารหัสแบบออโตคีย์นั้น มักจะใช้ข้อมูลที่เข้ารหัสไปแล้ว ป้อนกลับมาใช้ในการสร้างคีย์ มีรูปแบบดังในรูปที่ 1.7 ส่วนรูปที่ 1.8 เป็นการถอดรหัสออโตคีย์ที่ใช้คู่กัน

เพราะว่าในการเข้ารหัสได้มีการนำเอาข้อมูลที่เข้ารหัสไปแล้ว ป้อนกลับมาเพื่อสร้างค่าคีย์สตรีม ค่าของคีย์สตรีมจึงขึ้นอยู่กับข้อมูลที่เข้ารหัสแล้ว และฉะนั้น ค่าของคีย์สตรีมจึงขึ้นอยู่กับค่าของข้อมูลที่นำมาเข้ารหัสด้วย



รูปที่ 1.7 รูปแบบทั่วไปของระบบการเข้ารหัสออโตคีย์ ที่มีการป้อนกลับด้วยข้อมูลที่เข้ารหัส



รูปที่ 1.8 รูปแบบทั่วไปของระบบการถอดรหัสออโตคีย์ ที่ใช้คู่กับการเข้ารหัสในรูปที่ 1.7

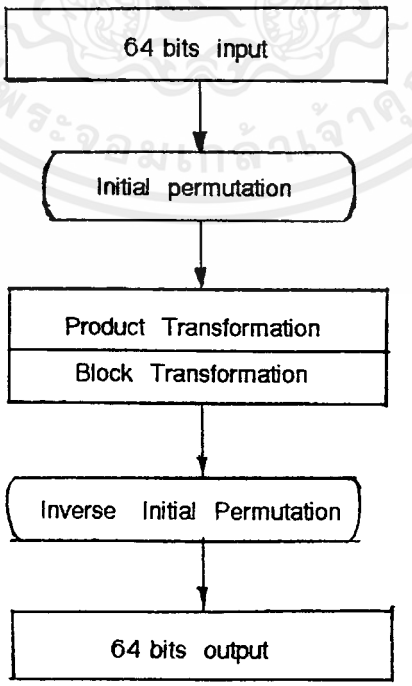
บทที่ 2 ทฤษฎีและหลักการทำงาน

ในบทนี้จะกล่าวถึงรายละเอียดของทฤษฎีต่างๆที่ได้กล่าวถึงไปแล้วในบทก่อน

2.1 ดาต้าเอนคริพชั่นสแตนดาร์ด (data encryption standard)

1.หลักการเบื้องต้น

หลักการทำงานของ การเข้ารหัสแบบ DES ขั้นพื้นฐาน คือ การเข้ารหัส ข้อความ (plain text)ทีละ 64 บิทโดยข้อมูลทั้ง64 บิทจะถูกทำการสลับบิท(permutate) ก่อนที่จะนำข้อมูลที่ถูกเพอร์มิวเทตนั้น ไปทำการเข้ารหัสจริงๆอีกครั้งเราเรียกว่าการทำ product tranformation ซึ่งส่วนนี้เองที่เป็นส่วนที่สำคัญที่ทำให้การเข้ารหัสมีความปลอดภัยอย่างมาก และเป็นส่วนที่มีความยุ่งยากที่สุดของการทำงาน ซึ่งจะทำควบคู่ไปกับ block tranformation ซึ่งเป็นเพียง การสลับกลุ่มบิท 2 กลุ่ม กลุ่มละ 32 บิท เป็นซีกซ้ายและซีกขวา ขั้นตอนสุดท้ายคือการทำ inverse initial permutation กลับการเพอร์มิวเทตครั้งแรก ก็จะได้ข้อมูลที่ผ่านการเข้ารหัสแล้วออกมาตามต้องการ แสดงรูปแบบการทำงานได้ง่ายๆตามโฟล์ชาร์ทดังรูปที่ 2.1

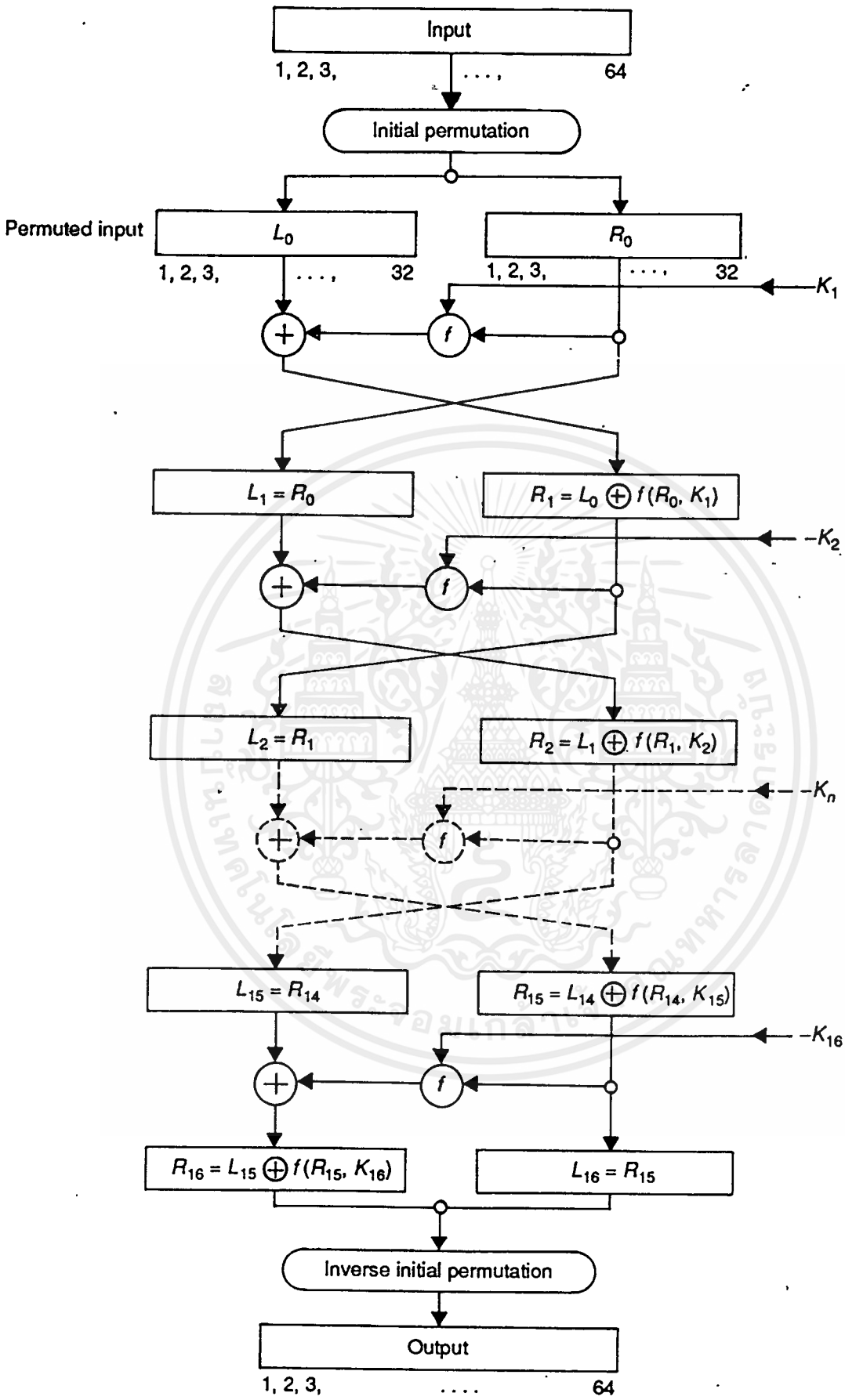


รูปที่ 2.1 โฟลชาร์ทแสดงการเข้ารหัสแบบ DES

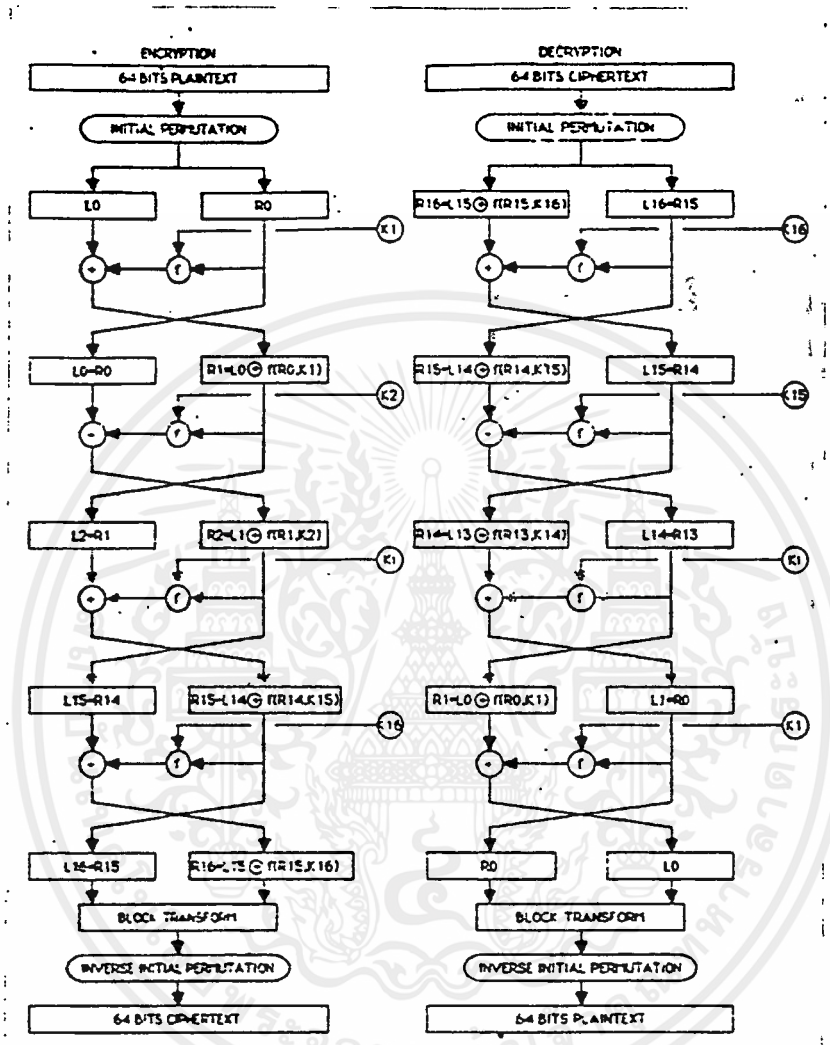
จากรูปที่ 2.1 ที่แสดงขั้นตอนการทำงานอย่างง่ายๆของ DES แล้วเราจะแสดงส่วนที่เป็นรายละเอียดของขั้นตอนหลังจาก การทำเพอร์มิวเตชั่นขั้นแรกแล้ว โดยเราจะแบ่งข้อมูลทั้ง 64 บิทออกเป็น 2 ส่วนเท่าๆกันคือ ซีกซ้าย (L) ซีกขวา (R) เราจะมาดูกันที่flow chart ในรูปที่ 2.2 จะเห็นว่าหลังจากแบ่งแล้วในขั้นแรก จะให้เป็น L0 และ R0 ซึ่งจะถูกรวมหรือผสมเข้ากับ คีย์ (K1 ถึง K16)ด้วยฟังก์ชั่น f จากนั้นก็สลับชุดข้อมูลซ้ายขวาทำการเข้าคีย์ต่อไป รวมทั้งสิ้น 16 รอบ ด้วยกันจะได้เป็น L1 ถึง L16 และ R1 ถึง R16 เป็นอันเสร็จ ขบวนการที่เรียกว่า product tranformation ส่วน block tranformation ก็คือ การสลับที่ กลุ่มของ L กับกลุ่มของ R เท่านั้นเอง

จาก flow chart จะมีตัวต่างๆที่เห็นอธิบายได้ดังนี้ f รูปเป็นฟังก์ชันที่เราจะมาดูรายละเอียดในหัวข้อที่ 3 ส่วน K1 ถึง K16 เป็นคีย์ย่อยยาว 48 Bits ที่ได้มาจากคีย์ใหญ่ซึ่งยาว 56 Bits แต่จะได้มาอย่างใดนั้นดูได้ในหัวข้อที่ 4 สำหรับเครื่องหมายบวกในวงกลมเรียกว่า bitwise exclusive or ในกรณีนี้ก็คือ การเอ็กคลูซีฟออร์ทีละบิทนั่นเอง

คุณสมบัติที่สำคัญอีกอย่างหนึ่งก็คือ เป็นระบบที่ใช้ได้ทั้งเข้าและถอดรหัส ดังรูปที่ 2.3 จะเห็นได้ว่าการถอดรหัสทำได้ง่ายมาก เพียงแต่ป้อนข้อความที่เข้ารหัสแล้ว (cipher text) เข้าทางด้านอินพุทแล้วก็ทำการสลับบิท และสลับคีย์ย่อยเสียใหม่ในทิศตรงกันข้ามก็จะได้เอาท์พุทเป็นข้อความที่ถอดรหัสออกมาแล้วตรงกับที่ใส่เข้าไปตอนแรก



รูปที่ 2.2 รีฟลชาร์ทแสดงรายละเอียดอัลกอริทึมของ DES



รูปที่ 2.3 โฟล์ชาร์ทของการเข้าและถอดรหัส DES

2.2 การเพอร์มิวเตชั่น ขั้นแรกและขั้นสุดท้าย

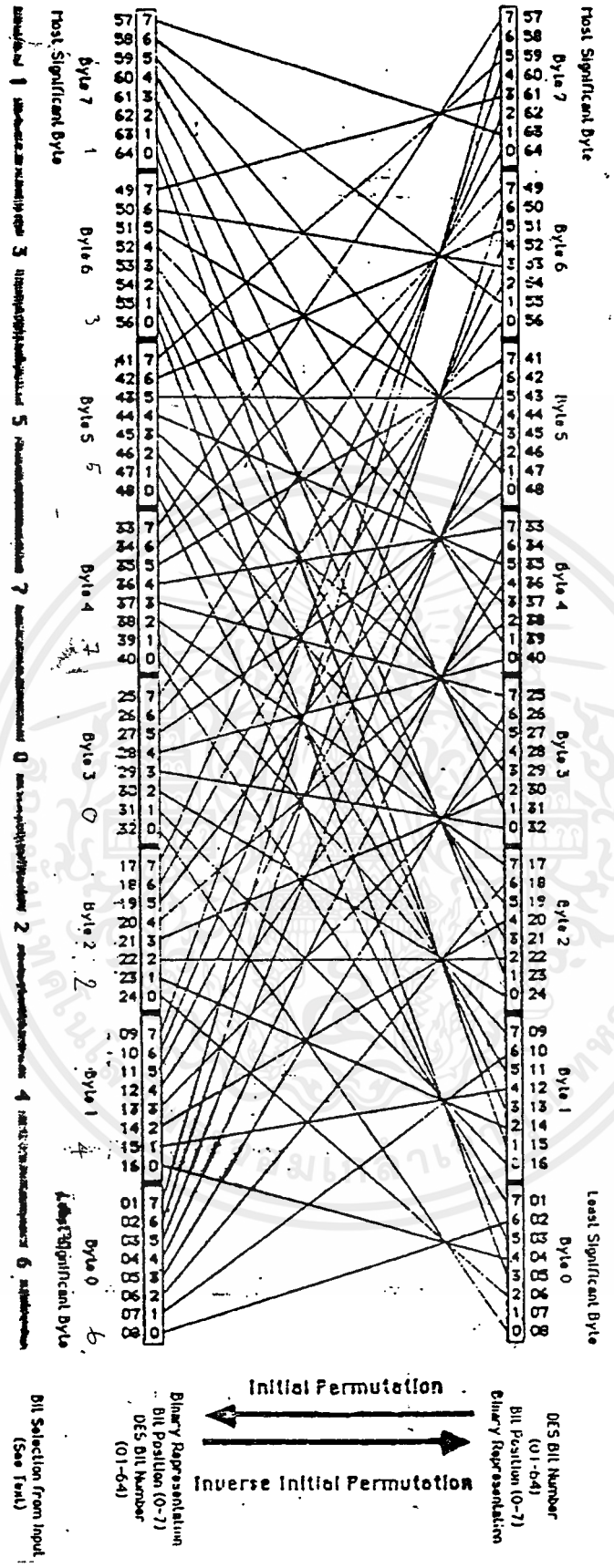
จากรูปที่ 2.4 ก และ 2.4 ข แสดงตารางเพอร์มิวเตชันขั้นแรกและขั้นสุดท้าย ซึ่งกลับทิศ(Inverse) กับขั้นแรก ถ้าเรานำหมายเลขของ Bit Input กับของ Output ไปเปลี่ยนเป็นหมายเลขบิทแบบ Binary ธรรมดาแล้ว ลากเส้นเชื่อมโยงจาก Input ไปยัง Output ก็จะได้ผลตามรูปที่ 2.4 ค ซึ่งจะเห็นได้โดยง่ายว่าแต่ละ Byte ของ Output จะเป็น Bit ในตำแหน่งต่าง ๆ ที่มาจาก Input ตามลำดับคือ 1-3-5-7-0-2-4-6 ซึ่งง่ายต่อการจดจำและทำความเข้าใจมากกว่าในตาราง ส่วนตาราง Inverse Initial Permutation ในรูปที่ 2.4 ข ก็จะมีลักษณะกลับกันกับตารางในรูป 2.4 ก หมายความว่า ถ้านำ Plain Text ผ่านการ Permutation ครั้งหนึ่งแล้วนำผลที่ได้ไปผ่าน Inverse Permutation ก็จะได้ Plain Text ของเดิมกลับมาอยู่ในรูปของหมายเลข Bit และ Binary ธรรมดา

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

รูปที่ 2.4 ก ตาราง Initial Permutation อ่านจากซ้ายไปขวาและบนลงล่าง เช่น บิทที่ 1 ของ permutated data มีค่าเท่ากับ บิทที่ 58 ของข้อมูลที่ป้อนเข้าไป

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

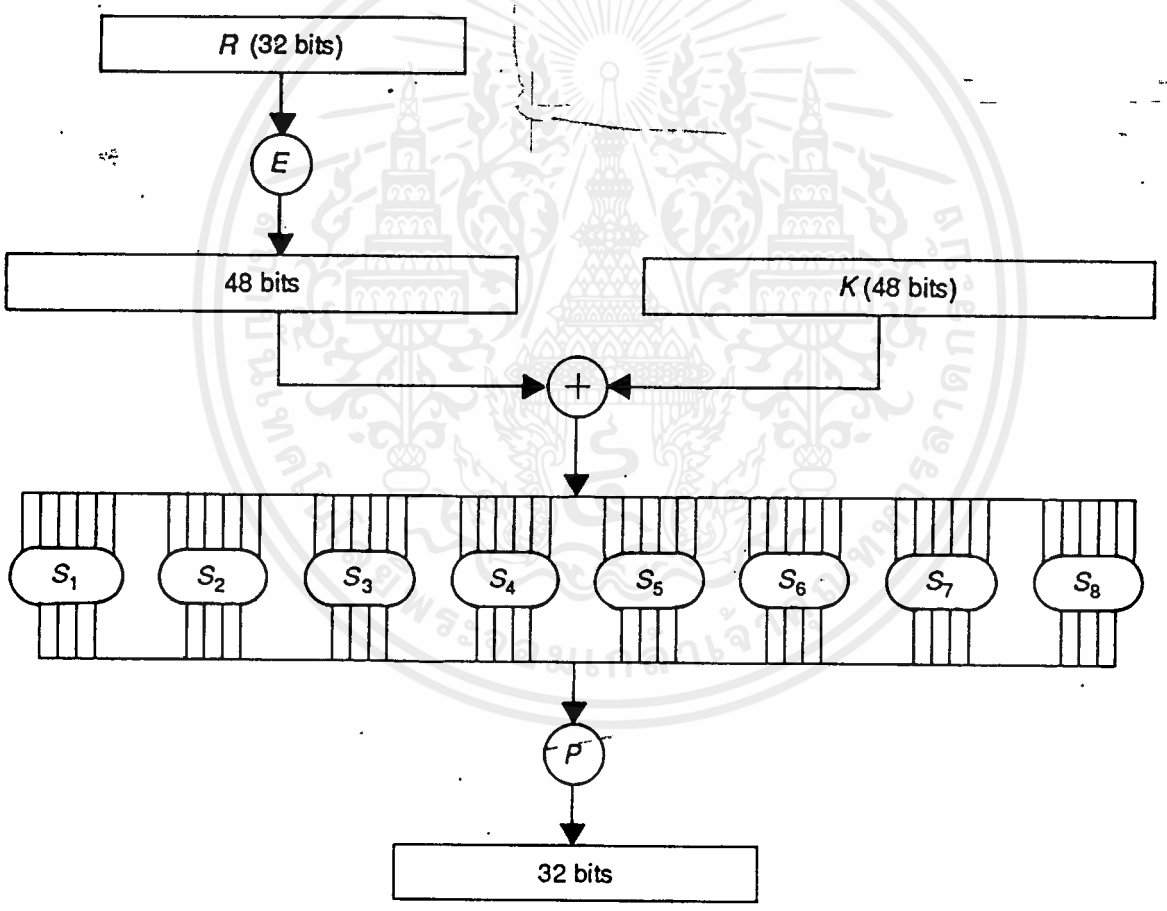
รูปที่ 2.4 ข ตาราง Inverse Initial Permutation อ่านเช่นเดียวกับตารางที่ 2.4 ก แต่ความหมายกับกัน



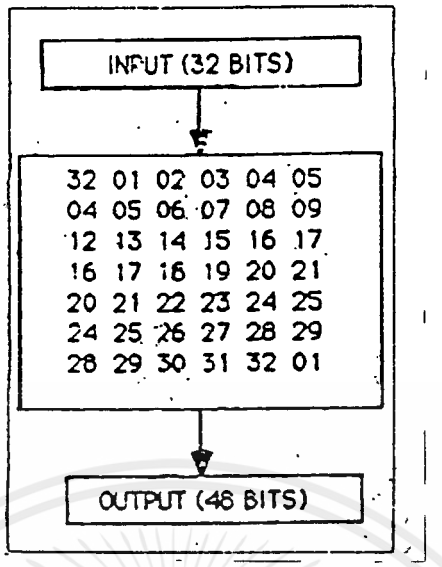
รูปที่ 2.4 ค ความสัมพันธ์ระหว่างอินพุทกับเอาท์พุทของตาราง 2.4ก และ 2.4ข

2.3 การคำนวนฟังก์ชัน f(Ri-1,Ki)

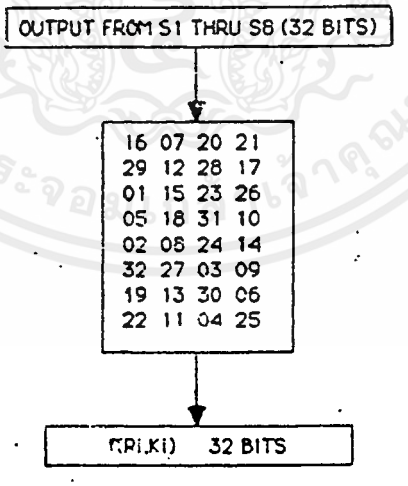
ฟังก์ชัน f(Ri-1,Ki) มีรายละเอียดดังแสดงไว้ในรูปที่ 2.5 ก ขั้นแรก Ri-1 จะถูก Permutate เสียก่อนด้วยตาราง Select E ที่แสดงไว้ในรูปที่ 2.5 ข ซึ่งจะเปลี่ยนข้อมูล 32 Bits ของ Ri-1 ให้เป็น 48 Bits เพื่อจะนำมาบวกแบบมอดูโลสองกับคีย์ย่อย Ki ที่ยาว 48 Bits เหมือนกัน จากนั้นผลบวก 48 Bits จะถูกยุบให้เหลือ 32 Bits ด้วยฟังก์ชัน S ดังจะอธิบายต่อไป ขั้นสุดท้ายคือการ Permutate ด้วยตาราง Permutation P ดังแสดงไว้ในรูปที่ 2.5 ค ได้ผลลัพธ์ออกมาเป็น f(Ri-1,Ki)



รูปที่ 2.5 ก โฟลชาร์ท แสดงรายละเอียดฟังก์ชั่น f(Ri-1,Ki)



รูปที่ 2.5 ข ตาราง Select E Permutation ยืดข้อมูล 32 บิท ให้เป็น 48 บิท



รูปที่ 2.5 ค ตาราง Permutation P เปลี่ยนข้อมูลจาก S1-S8 ให้เป็น f(Ri-1,Ki)

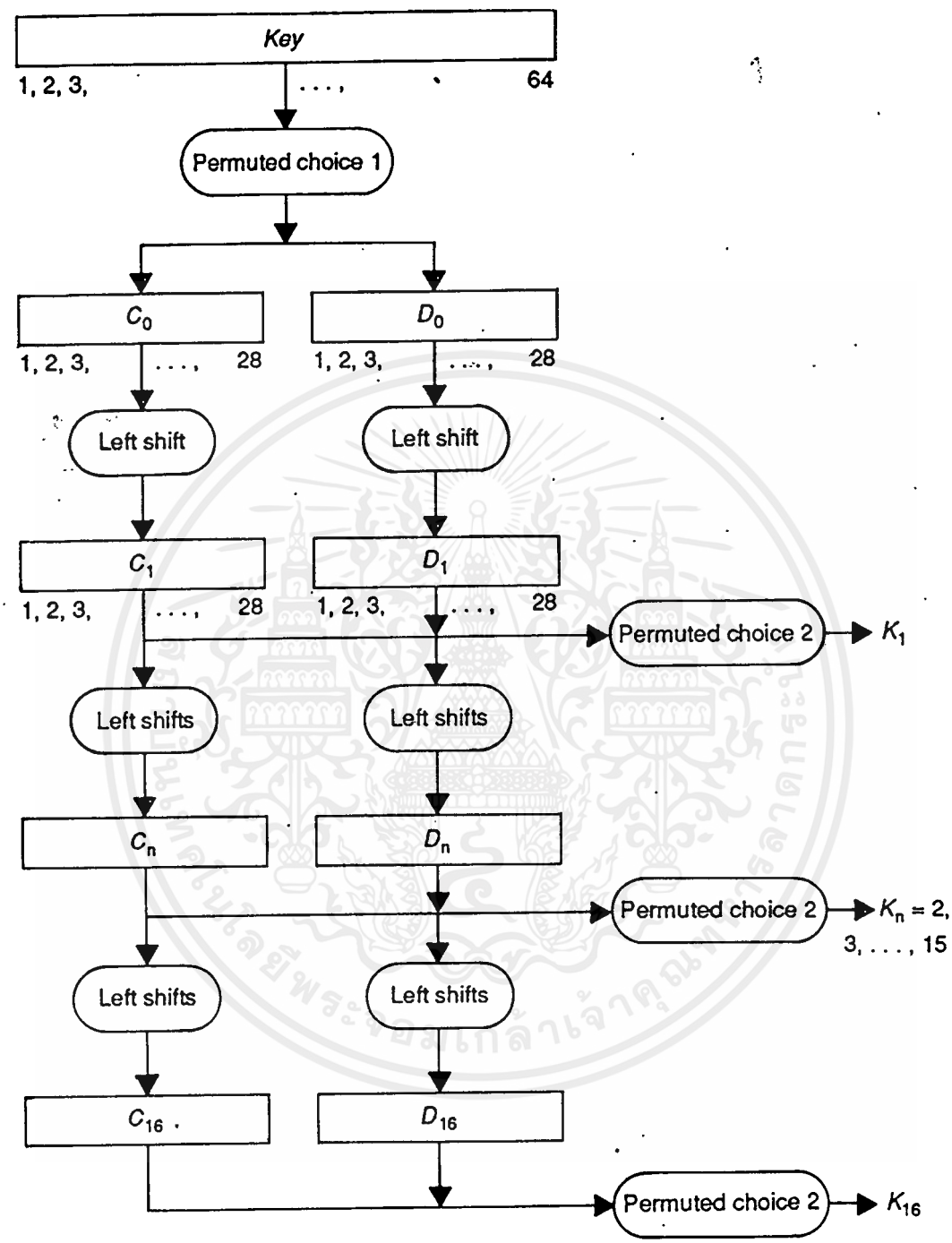
	Column																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

รูปที่ 2.5 ง ตารางการใช้ฟังก์ชั่น S1- S8

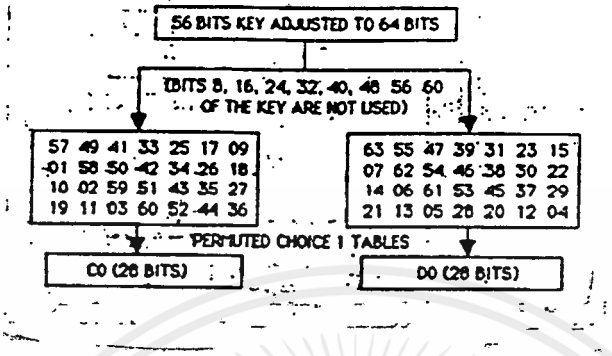
การยุบข้อมูล 48 Bits ให้เหลือ 32 Bits นั้นทำได้โดยใช้ตาราง 8 ตาราง มาทำงานขนานกันแต่ละตารางมี Input 6 Bits และ Output 4 Bits เรียกว่า Selection Function S1-S8 ดังแสดงไว้ในรูป 2.5 ง แต่ละตารางมีวิธีอ่านเหมือนกัน ดังแสดงไว้ในรูป 2.5 ส่วนตาราง Permutation P ขั้นสุดท้ายนั้นก็แสดงไว้ในรูป 2.5 ซึ่งไม่มีอะไรพิเศษมากเพียงแต่เปลี่ยนข้อมูล 32 Bits ที่ได้จาก S1-S8 ให้เป็นผลลัพธ์ f(Ri-1,Ki) ซึ่งมีความยาว 32 Bits เหมือนกัน

2.4 การสร้างคีย์ย่อยจากคีย์ใหญ่

ในขณะการใช้งานผู้ใช้จะต้องป้อนคีย์ยาว 56 Bits เข้าไปเป็นคีย์ที่เครื่องจะใช้สำหรับสร้างคีย์ย่อย (K) ขึ้นมาแสดงในรูปที่ 2.6 ก แรกทีเดียว คีย์จะถูก Permutate ด้วยตาราง Permutated Choice 1 ซึ่งแสดงในรูปที่ 2.6 ข ออกมาเป็นข้อมูล 2 ชุด แต่ละชุดยาว 28 Bits เรียกว่า CO และ DO จากนั้น CO และ DO จะถูกหมุนไปทางซ้าย ไม่ 1 ก็ 2 Bits ขึ้นอยู่กับครั้งของขั้นการทำ Product Transformation ซึ่งแสดงไว้ในตารางของรูปที่ 2.6 ค ขั้นสุดท้ายก็คือ การ Permutate ด้วยตาราง Permutated Choice 2 ซึ่งช่วยยุบข้อมูล 28 Bits 2 ชุดเข้าเป็นคีย์ย่อยยาว 48 Bits ดังแสดงไว้ในรูปที่ 2.6 ง



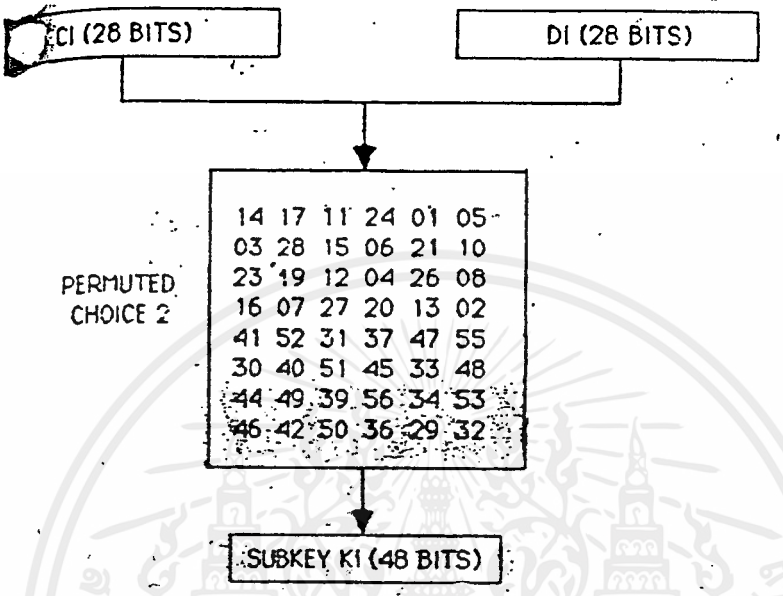
รูปที่ 2.6 ก รีฟล์ชาร์ทแสดงการสร้างคีย์ย่อย ๆ จากคีย์ใหญ่ยาว 56 บิท ที่ผู้ใช้ป้อนเข้าไป



รูปที่ 2.6 ข ตาราง Permutated choice 1 ซึ่งแยกคีย์ออกเป็น C0 กับ D0 แต่ละส่วนยาว 28 บิท

Iteration	*Number of i*	*Iteration*	*Number of i*
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

รูปที่ 2.6 ค ตารางแสดงจำนวนครั้งของการหมุนข้อมูลไบนารีไปทางซ้ายเพื่อหา n ในรูป 2.6 ก



รูปที่ 2.6 ง การสร้าง Subkey Ki จาก Ci และ Di ด้วยตาราง Permutated choice 2

บทที่ 3 การโปรแกรม

จากบทที่แล้ว ๆ มาได้กล่าวถึงวิธีการในการเข้ารหัสแบบต่าง ๆ รวมถึงแบบ DES มาแล้ว ในบทนี้จะเป็นการกล่าวถึงวิธีการที่ใช้ในการโปรแกรมเข้ารหัส โดยโปรแกรมที่จะใช้เป็นภาษา C โดยใช้ compiler คือ Turbo C 2.0

สำหรับโปรแกรมส่วนที่ทำการเข้ารหัส มีดังนี้

ส่วนแรกของโปรแกรมนี้เป็นการประกาศค่าคงที่ต่าง ๆ ที่ใช้ในโปรแกรม โดยค่าเหล่านี้เป็นค่าคงที่ที่ใช้ในโปรแกรม ตัวอย่างเช่น อะเรย์ IP เป็นแพทเทิร์นของ การสลับบิทเริ่มต้นอะเรย์ FP เป็นแพทเทิร์นของการสลับบิทสุดท้าย อะเรย์ E เป็นแพทเทิร์นของการขยายบิท อะเรย์ P เป็นแพทเทิร์นของการสลับบิทของฟังก์ชั่น P เป็นต้น (ดูอัลกอริธึมของ DES ประกอบ)

```
#include <stdlib.h>
#include <stdio.h>

typedef unsigned long    Long;
typedef unsigned int     Int;
typedef unsigned char    Char;

Long  F( Long R, char *Key );

Char IP[64] = {
  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 };
```

```
Char FP[64] = {
  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 };

Char E[48]  = {
  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 };

Char P[32]  = {
  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 };
```

```
Char S[8][64] = {
                    /* S1 == S[0] */
      14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,
       0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,
       4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,
      15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6,

   /* S2 */
      15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5,
       3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,
       0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2,
      13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,

   /* S3 */
      10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,
      13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,
      13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,
       1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2,

   /* S4 */
       7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4,
      13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,
      10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,
       3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2,

   /* S5 */
       2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,
      14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,
       4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0,
      11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,
```

```
    /* S6 */
        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
    /* S7 */
         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

    /* S8 */
        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

Char PC1[64] = {
  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,
  32, 32, 32, 32,                 /* <- not used this 4 bits */
  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,
  64, 64, 64, 64              }; /* <- not used this 4 bits */

Char Keyrol[16] = { 1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1 };
```

```
Char PC2[48] = {
  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 };

static Char  SubKey[16][8];
```

ฟังก์ชัน Permu64 เป็นการทำการสลับบิทแบบ 64 บิท ซึ่งต้องใช้ในตอนเริ่มทำการเข้ารหัส คือ ส่วน Initial Permutation และ การสลับบิทตอนท้าย คือ ส่วน Inverse Initial permutation ซึ่งการสลับบิทตอนแรก และ ตอนสุดท้าย จะใช้รูปแบบที่แตกต่างกัน โดยส่งรูปแบบเข้าใน พารามิเตอร์ชื่อว่า Pattern

การทำงานของฟังก์ชันนี้ต้องอาศัยฟังก์ชันอ่านค่าของบิทมาว่าเป็น 0 หรือ 1 (จากฟังก์ชัน Bit ข้างล่างนี้ เช่น การสลับบิทของ Initial Permutation บิทแรกของข้อมูลที่ถูกสลับแล้ว คือ บิทที่ 58 ของบิทเดิม ฟังก์ชัน Permu64 จะเรียกฟังก์ชัน Bit ดูว่าบิท 58 ของข้อมูลเดิมเป็นเท่าไร แล้วจึงทำการ or bit กับค่าที่อ่านได้มาอยู่บิทสุดท้าย ค่านี้จะถูก shift left ไปเรื่อย ๆ เมื่ออ่านบิทเข้ามาใหม่ จนถึงการอ่านค่าบิทที่ 64 บิทที่ 1 (คือบิทที่ 58 ของข้อมูลเดิม) จะถูก Shift มาจนถึงบิทที่ 1 เอง

ฟังก์ชัน Permu64 นี้ใช้ได้ทั้งการสลับบิทตอนเริ่มต้น และ ตอนท้าย โดยเปลี่ยนค่า Pattern ที่ใส่เข้าไป ซึ่งการสลับบิทตอนแรกใช้ pattern ของบิทที่กำหนดไว้ใน อะเรย์ IP ส่วนตอนท้ายจะใช้ FP

```
/* Function : Permu64                                          */
/*     Duty : permutation a 64 bits block using defined pattern */
/*    Input : long input[2]                                     */
```

```
/*   Output : long output[2]                                    */
/* Return Value :  None (after final test)                     */
Permu64(Long input[2],Long output[2],Char *Pattern)
{
register int i;

     for ( i=0; i<2; i++ ) {
       output[i] ^= output[i];          /* Clear Output Words to 0L */
  }                                     /*  by xor itself           */

     for ( i=0; i<=31; i++) {           /* Input permu bit 1-32 */
       output[0] <<= 1;                 /* shift left 1 bit */
       if ( Bit( input, (int) Pattern[i] ) == 1 )  {  /* อ่านค่าบิท */
         output[0] |= 1; }              /* OR BIT เข้าที่บิทสุดท้าย */
  }

     for ( i=32; i<=63; i++) {          /* Input permu bit 33-64 */
       output[1] <<= 1;
       if ( Bit( input, (int) Pattern[i] ) == 1 )  {
         output[1] |= 1; }
  }

}

/* Function: Bit  find the value of input bit 0 or 1 and return int
/*    Input: Long 2 words  = [1,2,...,32][33,34,...,62]
/*                                 In[0]        In[1]
/*         : And position of bit to find value 1...62 (Chart)
```

```
int Bit(Long InBit[2], Int Chart)
{
int   Side,Pos;

      Side = (Chart-1)   / 32;
      Pos  = (64-Chart)  % 32;

      return ( (InBit[Side] >> Pos) & 01 );
}
```

ฟังก์ชันต่อไปคือ ฟังก์ชัน F โดยฟังก์ชันนี้จะประกอบไปด้วย 3 ฟังก์ชันย่อย ๆ อีก 3 ฟังก์ชันคือ

1. ฟังก์ชัน E เป็นการขยายข้อมูล 32 บิท ให้เป็น 48 บิท โดยใช้รูปแบบที่กำหนดไว้แล้ว ซึ่งจะมีบางบิทจะถูกใช้ซ้ำกันเพื่อขยายบิท

2. การเข้า S Box ส่วนนี้จะเป็นอัลกอริที่มในการเข้ารหัสแบบการแทนที่ ในการเข้ารหัสแบบ DES ซึ่งจะใช้ข้อมูล 4 บิทมาแทนที่ข้อมูล 6 บิท เมื่อข้อมูลผ่านส่วนนี้ไปแล้ว ข้อมูลจะมีขนาด 32 บิทเท่าเดิม

3. การสลับบิท แบบ 32 บิท วิธีการทำงานเหมือนกับ การสลับบิทแบบ 64 บิทแต่จะทำการสลับบิทแค่ 32 บิท และจะใช้แพทเทริ่นที่กำหนดไว้ในอะเรย์ P

```
Long  F( Long R, char *Key )
{
Char ER[8], S_Val;
Long Out, S_Out=0L, Output=0L;
register int j,rc;

      Expand( R, ER );    /* เรียกฟังก์ชัน Expand ขยายข้อมูลจาก
                             32 บิท เป็น 48 บิท  */
```

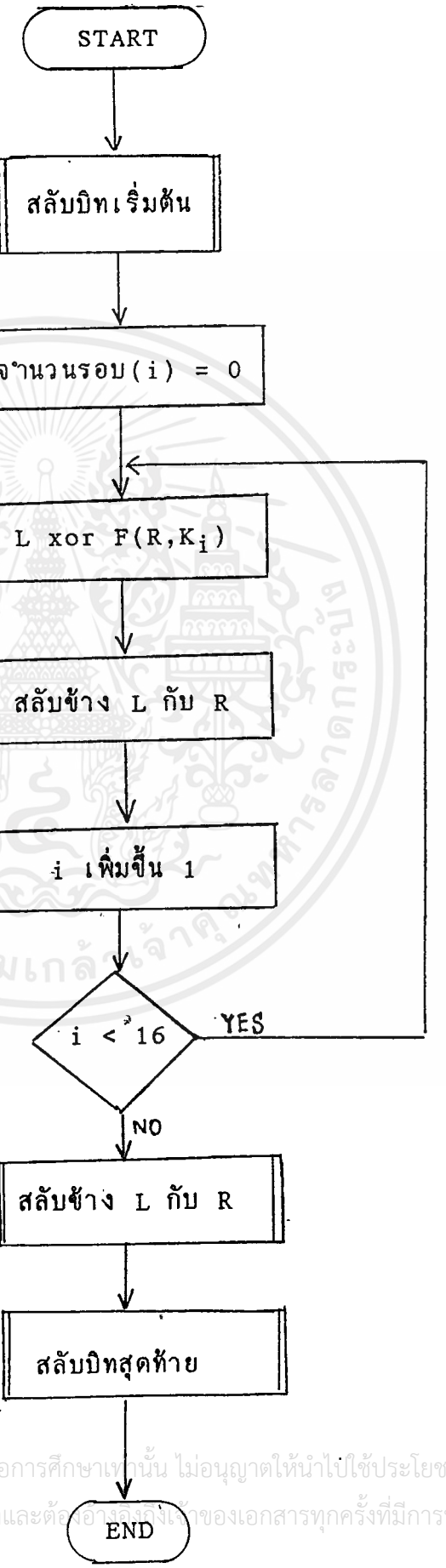
```
    for (j=0; j<8; j++) {    /* ส่วนนี้เป็นการเลือก S box */
      rc = ER[j] ^ Key[j];
      rc = ( rc & 0x20 ) | ( ( rc << 4 ) & 0x10 ) |
      ( ( rc >> 1 ) & 0x0F );

    S_Val    = S[j][rc];
    S_Out    = ( S_Out << 4 ) | S_Val;
    }

    Permu32( &S_Out, &Output, P );  /* ทำการสลับบิทแบบ 32 บิท */
    return ( Output );
}

/*        This function is used for expand Right side in          *
/*      function F from 32 -> 48 bits using defined pattern.      *
/*                                                                *
/*   Function : Expand ( R, Output )                              *
/*      Input : Long R  32 bits input from function F             *
/*     Output : char * Output  ( 8 char = 64 bits but use 48 bits) *
/*   Use bits :  R => [1,2,...,32] , Output => [x,x,1,2,3,4,5,6]  *
/*               xx in Output is unused in program                *
/*                                                                *
/*       Note : This function is not used array pattern but       *
/*              programmed in other style for the reason of       *
/*        more speed.                                          */
```

```
Expand( Long R, Char *Output )
{
#define CF2  0x3F        /* For clear first 2 bits => 00xxxxxx */

      Output[0]  =   ( ( R >> 27 )  |  ( R << 5 ) )  &  CF2;
      Output[1]  =     ( R >> 23 )  &  CF2;
      Output[2]  =     ( R >> 19 )  &  CF2;
      Output[3]  =     ( R >> 15 )  &  CF2;
      Output[4]  =     ( R >> 11 )  &  CF2;
      Output[5]  =     ( R >>  7 )  &  CF2;
      Output[6]  =     ( R >>  3 )  &  CF2;
      Output[7]  =   ( ( R <<  1 )  |  ( R >> 31 ) )  &  CF2;
}

Permu32( Long *Input, Long *Output, Char *Pattern )
{
register int i;

      *Output = 0x0L;
      for ( i=0; i<32; i++) {
         *Output <<= 1;
         *Output |= Bit( Input, (int) Pattern[i] );
      }
}
```

จากโปรแกรมข้างบนนี้เป็นส่วนที่เกี่ยวกับการเข้ารหัสโดยตรง โดยโปรแกรมส่วนอื่น ๆ คือ การหาค่า Subkey และ โปรแกรมทำการอ่านข้อมูลเพื่อมาทำการเข้ารหัส จะอยู่ในส่วนภาคผนวก

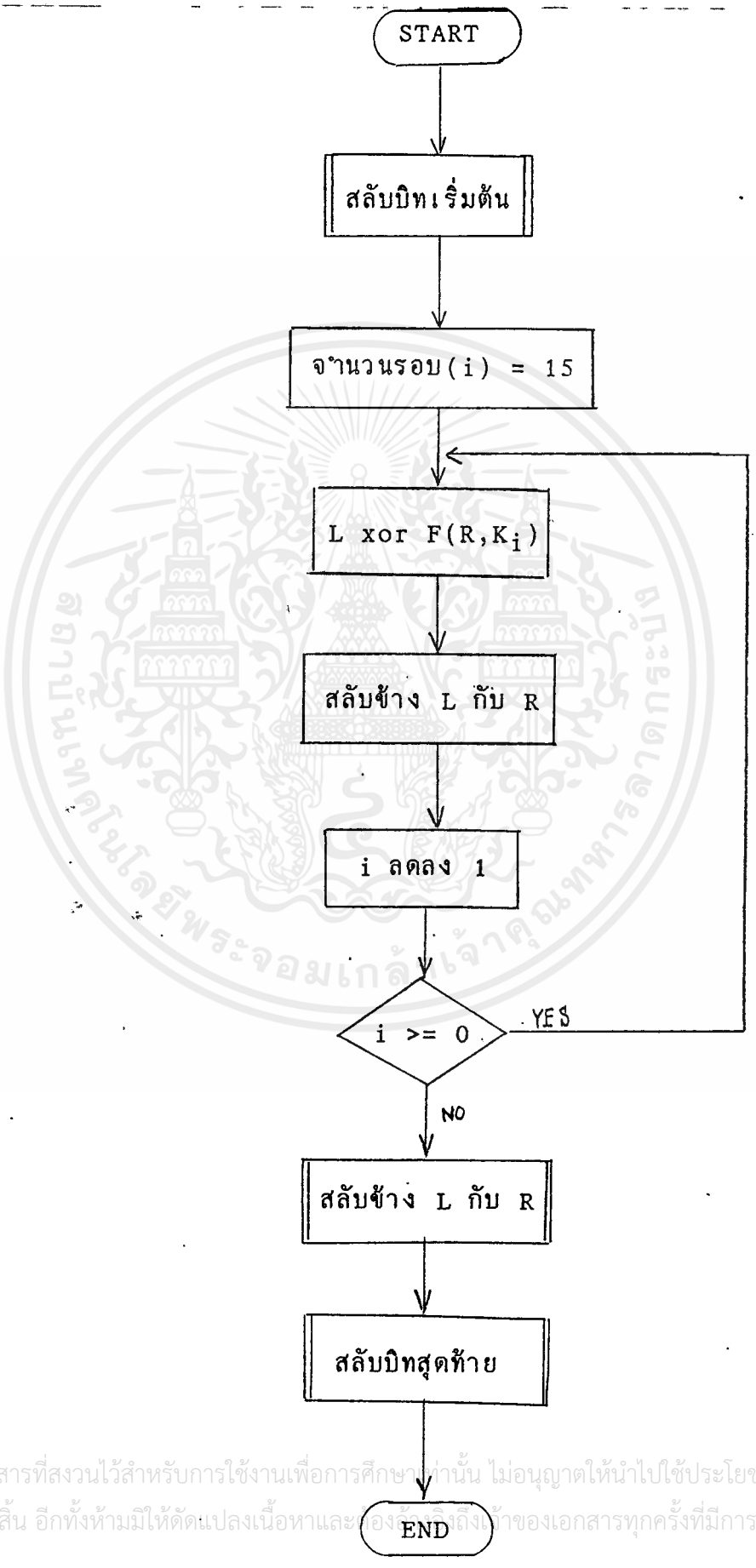
DES ENCRYPT

flow chart การทำงานของฟังก์ชั่น ENCRYPTION



DES DECRYPT

flow chart การทำงานของฟังก์ชั่น DECRIPTION



จากอัลกอริธึมในการเข้ารหัสแบบ DES เราสามารถนำมาเขียนเป็นโปรแกรม เพื่อทำการเข้ารหัสรักษาความปลอดภัยข้อมูลได้แล้ว จากโปรแกรมในภาคนี้ไว้วัดเวลาการทำงานของโปรแกรม โดยใช้อินเตอร์รัพ 1Ch และสรุปได้ดังนี้

ให้เวลาการทำงานของการเข้ารหัส 1 รอบ (64 บิท) เป็น 100% เวลาการทำงานของฟังก์ชันต่าง ๆ เป็นดังนี้

1. ฟังก์ชัน Permu64(IP) ใช้เวลาประมาณ 7.76 %
2. ฟังก์ชัน F (ถูกเรียกใช้ 16 ครั้งในการเข้ารหัส 1 รอบ)
   ใช้เวลาประมาณ 84.5 %
3. ฟังก์ชัน Permu64(FP) ใช้เวลาประมาณ 7.76 %

ฟังก์ชัน F เป็นฟังก์ชันที่ประกอบด้วย ฟังก์ชันย่อย ๆ อีก 3 ฟังก์ชันนั้น วัดเวลาการทำงานของฟังก์ชันย่อยได้ดังนี้ (ให้ฟังก์ชัน F เป็น 100 %)

1. ฟังก์ชัน Expand ใช้เวลาประมาณ 6.25 %
2. การเลือก S box ใช้เวลาประมาณ 12.5 %
3. ฟังก์ชัน Permu32 ใช้เวลาประมาณ 81.25 %

เมื่อคิดเป็นสัดส่วนในของเวลา ในการเข้ารหัสได้ผลดังนี้

Permu64(IP)	7.76 %
Expand	5.28 %
S_box	10.56 %
Permu32	68.7 %
Permu64(FP)	7.76 %

จะเห็นได้ว่าฟังก์ชัน Permu32 ใช้เวลาไปมากดังนั้นควรทำการปรับปรุงการทำงานในส่วนนี้

ปัญหาที่เกิดขึ้นในส่วนของเทอมแรก

1. การทำงานของโปรแกรมช้า เป็นผลให้ผู้ใช้ต้องคอยเสียเวลานานมาก จากการวัดเวลาการทำงานพบว่า โปรแกรมทำการเข้ารหัสได้เพียงประมาณ 1500 บิทต่อวินาทีเท่านั้น (เมื่อทำการอ่าน/เขียนข้อมูลใน floppy disk drive 360K, เครื่อง microcomputer 386sx-16)

2. ยังไม่มีการตรวจสอบข้อผิดพลาดต่าง ๆ ของการอ่านเขียนข้อมูล จึงอาจทำให้เกิดข้อผิดพลาดได้ เช่น เหลือเนื้อที่เก็บข้อมูลน้อยไปไม่พอเขียนเป็นต้น

3. ยังไม่มีส่วนของโปรแกรมที่รับค่าคีย์เข้ามาทำการเข้ารหัส ซึ่งค่าคีย์ที่ใช้อยู่เป็นค่าคงที่ที่กำหนดไว้

การพัฒนาโปรแกรมในเทอมต่อไป

1. ทำการปรับปรุงประสิทธิภาพการทำงานของโปรแกรมให้เร็วขึ้น

2. ปรับปรุงโปรแกรมการอ่านเขียนข้อมูล ให้เหมาะสมยิ่งขึ้น และเพิ่มเติมส่วนตรวจสอบข้อผิดพลาดต่าง ๆ ตามสมควร

3. ทำโปรแกรมให้รับค่าคีย์เวิร์ดจากผู้ใช้ได้

4. ปรับปรุงส่วนแสดงผลให้ใช้ง่าย และน่าใช้ยิ่งขึ้น

หลังจากที่ได้ทราบปัญหาในเทอมที่ 1มาแล้วในเทอมที่ 2ก็ได้มีการพัฒนาโปรแกรมขึ้นมาเพื่อแก้ปัญหาต่างๆที่ว่ามาแล้ว และยังได้มีการเพิ่มเติมในส่วนที่ต่างกันออกไปด้วยคือการพัฒนาในส่วนของการแก้ไขเปลี่ยนแปลง ค่าภายในอัลกอริทึมของ DES ซึ่งในที่นี้เราเรียกว่า UDES คือการเปลี่ยนแปลงค่าในส่วนของการแทนที่ ภายใน S-boxes นั่นเอง ซึ่งการเปลี่ยนแปลงที่ทำขึ้นในเทอมนี้ถือว่าได้พัฒนาไปมากกว่าเทอมก่อนอย่างมาก

การพัฒนาทั้งหมดเราจะกล่าวถึงในบทที่ 4 ต่อไป

บทที่ 4 การพัฒนาโปรแกรม

จากโปรแกรมเข้าและถอดรหัสที่ได้ทำการเขียนเสร็จแล้วดังในบทที่ 3 ได้พบว่ามีปัญหาสำคัญคือความเร็วในการทำงานของโปรแกรม ดังนั้นจึงได้มีการพัฒนาโปรแกรมขึ้นมาใหม่ โดยเน้นความสำคัญอยู่ที่การทำให้อัลกอริทึมการเข้ารหัสแบบ DES ทำงานได้รวดเร็วที่สุดเท่าที่จะสามารถทำได้

ในส่วนโปรแกรมเวอร์ชั่นแรกที่เขียนขึ้นมาดังในบทที่ 3 นั้น ได้ปรากฏว่าเวลาส่วนมากใช้ไปกับส่วนการเข้าและถอดรหัสแบบสลับบิท (Permutation) โดยเฉพาะฟังก์ชัน F ซึ่งจะต้องทำงานถึง 16 รอบ ในการเข้ารหัสหนึ่งครั้งนั้น จะกินเวลารวมคิดเป็นประมาณ 80 เปอร์เซนต์ของเวลาการทำงานทั้งหมดเลยทีเดียว ดังนั้นการพัฒนาโปรแกรมจึงมุ่งเน้นไปทางหาอัลกอริทึมที่ทำให้ฟังก์ชัน F ทำงานได้รวดเร็วที่สุด และจะมีผลพลอยได้ทำให้คิดการพัฒนาฟังก์ชันการสลับบิทเริ่มต้น และ การสลับบิทตอนท้าย ได้อีกด้วย เพราะเป็นการทำงานแบบสลับบิทเหมือนกัน

นอกจากนี้โปรแกรมที่พัฒนาขึ้นนี้ได้มีการใส่ Passwordในการเข้ารหัสตามอัลกอริทึมของ DES เพื่อให้โปรแกรมมีความสมบูรณ์ในการทำงาน Password ที่ใส่นี้จะเป็นส่วนหนึ่งในการเข้ารหัสและถอดรหัสด้วย โดยจำนวน Password ที่กำหนดให้ใส่นั้นเป็นจำนวน 8 ตัวอักษรเป็นอย่างน้อย โดยจะใส่ตัวอะไรลงไปก็ได้ ไม่ว่าจะเป็นตัวเลข ตัวอักษรต่าง ๆ และ Password ที่ใส่ได้มากที่สุดเป็น 40 ตัวอักษรเลยทีเดียว การใส่ Password นั้นก็จะมีข้อความบอกให้ใส่ Encryption Password และDecryption Password ให้ไว้เรียบร้อย และมีการตรวจสอบข้อผิดพลาดที่เกิดขึ้นจากการใส่ Password อีกด้วย

นอกจากได้พัฒนาให้มีการใส่ Password จากผู้ใช้ได้แล้ว ยังมีการตรวจสอบความผิดพลาดต่าง ๆ เพื่อป้องกันไม่ให้เกิดความผิดพลาดขึ้นได้ง่าย เช่น ตรวจสอบว่าดิสก์ที่จะเขียนข้อมูลนั้นมีที่ว่างพอเพียงกับขนาดไฟล์ที่จะเขียนหรือไม่ ตรวจสอบว่าดิสก์ไดรฟ์เกิดการผิดพลาดหรือไม่ ตรวจสอบว่าไฟล์ที่จะ Decrypt นั้น ได้ถูก Encrypt มาก่อนแล้วหรือไม่ ตรวจสอบว่าไฟล์ที่จะทำการ Encrypt นั้นมีไฟล์อยู่หรือเปล่า ตรวจสอบว่าชื่อไฟล์ที่จะเขียนไฟล์นั้นมีอยู่ก่อนแล้วหรือไม่ เพื่อป้องกันไฟล์อื่นไม่ให้ถูกเขียนทับได้โดยง่าย

และการตรวจสอบข้อผิดพลาดอื่น ๆ อีกตามสมควร

นอกเหนือจากการพัฒนาให้ใส่ Password ตามความต้องการ และ มีการตรวจสอบข้อผิดพลาดต่าง ๆ แล้ว โปรแกรมนี้ยังมีการเปลี่ยนแปลงข้อมูลในส่วนของฟังก์ชันการเข้ารหัส S box ได้อีกด้วย โดยจัดการทำให้ผู้ใช้ได้ใช้งานได้ง่ายโดยใช้เพียงลูกศรเลื่อนแถบสว่าง และ กดคีย์ Space bar หรือ Enter เพื่อเปลี่ยนแปลงข้อมูลระหว่างกันได้ง่ายอีกด้วย

<u>การเพิ่มความเร็วให้กับอัลกอริทึม DES</u>

จากโปรแกรมเดิมนั้นได้ปรากฏว่าการทำงานค่อนข้างจะช้า จากการหาสาเหตุพบว่าการทำงานส่วนใหญ่จะใช้ไปกับการเข้ารหัสแบบการสลับบิท (Permutation) ดังที่ได้กล่าวไว้แล้ว ดังนั้นจึงต้องทำการหาอัลกอรึธึมของการทำงานของการเข้ารหัสแบบสลับบิทใหม่ โดยเน้นให้การทำงานได้รวดเร็วที่สุด จากการคิดค้นได้ทำการสลับบิทใหม่โดยใช้วิธีการ Shift บิทข้อมูลไปอยู่ในบิทที่ต้องการ จากการทดการ Shift บิทข้อมูลโดยการทดลอง Shift ทีละ 32 บิท 16 บิท และ 8 บิท

พบว่าการ Shift บิทข้อมูลแบบ 8 บิทจะทำให้ได้ความเร็วของการทำงานมากที่สุด เป็นผลให้ การทำงานของอัลกอริทึมนั้นเป็นไปด้วยความรวดเร็วมากกว่าโปรแกรมต้นแบบที่เขียนขึ้นมาได้มากทีเดียว โดยการ Shift บิทของข้อมูลนี้จะใช้ในการเข้ารหัส และ ถอดรหัส ที่เป็นแบบสลับบิท ทุกฟังก์ชัน เพื่อให้ได้การทำงานรวดเร็ว

สำหรับโปรแกรมนั้นสามารถดูได้ในภาคผนวก

ส่วนโปรแกรมการเข้ารหัสนั้นจะอยู่ในไฟล์ที่ชื่อ DESA.C โดยอัลกอริทึมเดิมที่ได้ทำการเขียนขึ้น สำหรับการเข้ารหัส และ ถอดรหัสที่เป็นแบบการสลับบิท นั้น จะเป็นชื่อ Permu64() Permu32() ส่วนฟังก์ชันใหม่ที่ได้ทำการพัฒนาขึ้นนั้นได้ชื่อว่า Permu64IP() สำหรับการเข้าและ ถอดรหัสตอนแรก Permu64FP() สำหรับการเข้าและ ถอดรหัส ตอนสุดท้าย และ Permu32_V3() สำหรับ การเข้ารหัสแบบสลับบิท ซึ่งเป็นส่วนหนึ่งในฟังก์ชัน F โดยทำการสลับบิทตามแพทเทิร์น P ของอัลกอริทึม DES ได้อย่างถูกต้อง โดยทุกฟังก์ชันและเป็นการทำการสลับบิทโดยการ Shift บิทแบบ 8 บิทดังที่ได้กล่าวไว้แล้ว

นอกจากนี้แล้วยังได้พัฒนาความเร็วของการแทนที่โดย S box อีกด้วย แต่ความเร็วที่ทำได้นั้นไม่เร็วกว่าของเดิมมากนัก

จากการวัดความเร็วในการทำงานของการเข้ารหัสแบบ DES ที่ได้พัฒนาขึ้นมาใหม่นี้ วัดเฉพาะการทำงานในการเข้ารหัสและถอดรหัส โดยตรงเท่านั้น ไม่รวมเวลาการทำงานในด้านอื่น ๆ เช่นการเขียนอ่านข้อมูลจากดิสก์ การแสดงผลการทำงานเป็นต้น ซึ่งจะทำให้การทำงานของโปรแกรมช้าลงบ้าง การทำงานในการเข้ารหัสและถอดรหัสของเดิมนั้น จากการทำงาน 10000 รอบ ซึ่งจะเป็นการเข้ารหัส 80000 บิท จะใช้การทำงานประมาณ 177 วินาที ส่วนการทำงานของการเข้ารหัสและถอดรหัสที่ได้พัฒนาขึ้นมานั้น จะใช้เวลาการทำงาน 10000 รอบ ในเวลาเพียง 45.6 วินาทีเท่านั้น ซึ่งจะเห็นได้ว่าการทำงานของฟังก์ชันที่ได้พัฒนาขึ้นมาใหม่นั้น จะทำงานได้เร็วกว่าเดิมมาก คือ ประมาณ 3.5 เท่าเลยทีเดียว

<u>การใส่ Password</u>

การใส่ Password นี้ได้ทำการเขียนขึ้นเพื่อทำให้การทำงานของโปรแกรมเป็นไปอย่างสมบูรณ์ โดยต้องใส่อย่างน้อย 8 ตัวอักษร และใส่ได้มากที่สุด 40 ตัวอักษร ซึ่ง Password ที่ใส่ทุกตัวนั้นมีผลต่อการเข้ารหัสและถอดรหัสข้อมูลด้วย โดยจะเป็นส่วนหนึ่งของการเข้ารหัส สำหรับการใช้งานส่วนการใส่ Password ได้บอกไว้ในส่วนของบทที่ 5 แล้ว ส่วนของโปรแกรมนั้นได้จัดไว้อยู่ในภาคผนวก โดยชื่อไฟล์คือ cryptkey.c ในส่วนของการใส่ password นี้ได้มีการตรวจสอบข้อผิดพลาดไว้ด้วย เป็นผลให้การทำงานเป็นไปได้อย่างถูกต้องตามความต้องการทุกประการ ข้อสำคัญที่สุดคือ **ผู้ใช้ต้องจำ Password ของไฟล์ให้ได้เท่านั้นถึงจะทำการถอดรหัสไฟล์ออกมาได้** มิฉะนั้นไฟล์นั้นจะเป็นไฟล์ที่ไม่สามารถนำไปใช้ได้อย่างถูกต้องตามความประสงค์เลย ซึ่งเป็นเหตุผลในด้านของความปลอดภัยของการทำงาน

การใส่ Password นั้นถ้าเป็นการเข้ารหัสจะมีให้ใส่ซ้ำเป็นรอบที่สอง เพื่อป้องกันความผิดพลาดที่ได้ใส่ไว้ในรอบแรก โดยจะตรวจสอบว่าทั้งสองรอบใส่ password เหมือนกันหรือไม่ ถ้าไม่เหมือนกันก็จะมีข้อความบอกข้อผิดพลาดมา และ ให้ใส่รอบที่หนึ่งใหม่อีกครั้งหนึ่ง

ส่วนในรอบของการถอดรหัสนั้นจะให้ใส่ Password เพียงครั้งเดียว ถ้าใส่ไม่ถูกต้องตามที่ได้ใส่ไว้ตอนทำการเข้ารหัสแล้ว ก็จะมีข้อความบอกความผิดพลาด และ จะไม่ทำการถอดรหัสไฟล์นั้นออกมาให้

<u>การปรับปรุงการอ่านเขียนข้อมูล</u>

การอ่านเขียนข้อมูลนั้น ในโปรแกรมแรกที่ได้ทำการเขียนขึ้นมาจะทำการอ่านเขียนข้อมูลโดยตรงจากดิสก์ทีละ 8 ไบท์ ซึ่งจะทำให้ดิสก์ต้องอ่านเขียนอยู่ตลอดเวลาซึ่งจะไม่เป็นผลดีต่อการทำงานของโปรแกรม โดยถ้านำโปรแกรมไปรันบนเครื่องที่มีความเร็วมาก ๆ แล้ว จะทำการอ่านเขียนข้อมูลเป็นไปอย่างรวดเร็วแต่ไม่ต่อเนื่อง เนื่องจากโปรแกรมเดิมได้ทำการอ่านมา 8 ไบท์แล้วทำการเข้ารหัส หรือ ถอดรหัส แล้วจึงเขียนลงดิสก์ 8 ไบท์ ซึ่งจะทำให้มีการอ่านเขียนข้อมูลมาก หัวอ่านเขียนต้องทำงานอยู่ตลอดเวลา จึงได้มีการพัฒนาให้มีบัฟเฟอร์ของการอ่านเขียนดิสก์ขึ้น โดยการจองหน่วยความจำมาเป็นบัฟเฟอร์ในการอ่านเขียนข้อมูล โดยหน่วยความจำที่จะจองนั้น จะขึ้นอยู่กับ ขนาดของไฟล์ที่จะทำการเข้ารหัสหรือถอดรหัส และ หน่วยความจำของเครื่องที่ยังเหลือให้ใช้ได้อยู่ จากการทำให้มีบัฟเฟอร์ พบว่า การอ่านเขียนข้อมูล มีความถี่ในการอ่านเขียนน้อยลงตามความต้องการ ซึ่งเป็นผลให้การทำงานเร็วขึ้นได้อีกด้วย

<u>การตรวจสอบข้อผิดพลาดต่าง ๆ</u>

โปรแกรมได้พัฒนาขึ้นมาอีกขั้นหนึ่งโดยได้มีการตรวจสอบข้อผิดพลาดต่าง ๆ มากมายตามความจำเป็นของโปรแกรมที่ได้พัฒนาขึ้นมาตามการใช้งานนั้น ซึ่งจะเป็นผลให้ผู้ใช้ใช้ได้อย่างสะดวก ไม่ต้องคอยกังวลว่าได้ทำถูกต้องหรือไม่ เพราะถ้าทำผิดจะมีข้อความแสดงข้อผิดพลาดออกมาให้เอง อีกทั้งข้อผิดพลาดบางอย่างยังมีทางเลือกให้คุณได้อีกด้วย เช่น ถ้าคุณสั่งให้ทำการถอดรหัสไฟล์ ไฟล์หนึ่งแล้ว (ซึ่งในไฟล์นั้นจะมีการเก็บชื่อไฟล์เดิมเอาไว้ด้วย) เกิดชื่อไฟล์ของเราไปตรงกับไฟล์อื่น ๆ เข้า หรืออาจเป็นไฟล์เดิมก่อนการเข้ารหัสที่ยังไม่ได้ลบไป โปรแกรมก็จะบอกเราว่ามีไฟล์อื่นที่ชื่อเหมือนกับชื่อที่เก็บไว้นะ และจะมีทางเลือกให้คือ เขียนทับ ซึ่งจะทำให้ไฟล์เดิมที่มีอยู่ก่อนนั้นหายไป และ ไฟล์ของเราจะเขียนไปแทน และ Cancel คือยกเลิกการเขียนข้อมูล และจะกลับไปที่ menu เพื่อให้คุณได้ทำอย่างอื่นแทน

สำหรับข้อผิดพลาดอย่างอื่นที่อาจจะเกิดขึ้นได้ได้อยู่ในตัวอย่างการใช้งานแล้ว ส่วนของโปรแกรมที่ทำการตรวจสอบข้อผิดพลาดทั่ว ๆ ไป อยู่ในไฟล์ชื่อ READ.C ในภาคผนวก

การปรับปรุง S Box ได้โดยผู้ใช้

จากโปรแกรมที่ได้พัฒนขึ้นมานี้ ยังได้พัฒนาเพิ่มเติมความสามารถพิเศษนอกเหนือจากการเข้ารหัสแบบ DES แบบมาตรฐานขึ้นมาด้วย นั่นคือ โปรแกรมนี้สามารถปรับเปลี่ยนรูปแบบการเข้ารหัสแบบการแทนที่ (Substitution)ได้อีกด้วย ทั้งนี้ที่เลือกให้โปรแกรมทำการปรับเปลี่ยน S box (Substitution box) ได้นั้น เป็นเพราะว่าในการเข้ารหัสแบบ DES นั้น ส่วนเข้ารหัสแบบการแทนที่จะทำให้ความสัมพันธ์ระหว่างอินพุทและเอาท์พุทไม่เป็นลิเนียร์ ซึ่งเป็นคุณสมบัติส่วนที่สำคัญมากที่สุดในการเข้ารหัสแบบ DES

การปรับปรุง S box นั้นทำได้โดยเข้าไปในเมนู Option และ เลือกการเข้ารหัสแบบ UDES และ เลือกเมนูทำการปรับปรุง S box การปรับปรุงนั้นได้ออกแบบให้กระทำได้ง่างและสะดวกต่อผู้ใช้ โดยใช้การสลับที่ข้อมูลเดิมเท่านั้น การสลับข้อมูลก็สามารถทำได้โดยง่ายโดยการเลื่อนแถบสว่างไปที่ข้อมูลในแถวที่ต้องการสลับ แล้วกดคีย์ Enter หรือ Spacebar ก็จะเป็นการเลือกข้อมูลตัวแรกที่จะทำการสลับ แล้วเลื่อนแถบสว่างไปที่ข้อมูลตัวที่สองที่ต้องการให้สลับด้วย แล้วกดคีย์ Enter หรือ Spacebar ก็จะเป็นการสลับที่ข้อมูลแล้ว

S box มีทั้งหมดด้วยกัน 8 box แต่ละ box จะประกอบด้วย 4 แถวตามอัลกอริทึ่ม และ แต่ละแถวจะมีข้อมูลอยู่ 16 ค่าด้วยกัน สาเหตุที่ต้องเป็นอย่างนี้ก็เพราะว่าการทำการแทนที่โดยใช้ S box ตามอัลกอริทึ่ม DES นั้น จะใช้ข้อมูล 6 บิท มาทำการแทนที่ให้ได้ข้อมูล 4 บิท ในข้อมูล 6 บิทนั้น บิทที่ 0 และ 3 จะเป็นตัวบอกแถวของ S box นั้น ๆ ดังนั้น แต่ละ Box จึงประกอบด้วย 4 แถวดังกล่าว ส่วนข้อมูล 4 บิทที่เหลือจะเป็นตัวบอกตำแหน่งของข้อมูลที่จะทำการแทนที่ด้วยค่าใน S box ดังนั้นในแต่ละแถวจะเป็นค่าของข้อมูล 4 บิท ซึ่งจะแทนค่าได้ทั้งหมด 16 ค่าด้วยกัน เมื่อเป็นอย่างนี้แล้วค่าในแต่ละแถวจะต้องไม่ซ้ำกัน จึงจะให้ความปลอดภัยที่สูง ดังนั้นการออกแบบการปรับเปลี่ยน S box จึงได้เลือกการสลับค่าข้อมูลเดิม

เมื่อปรับเปลี่ยน S box แล้วและได้ทำการเข้ารหัสแล้ว พบว่าข้อมูลที่ได้ถูกเข้ารหัสนั้นข้อมูลได้เปลี่ยนไปจากการเข้ารหัสโดยวิธี DES ธรรมดาตามความประสงค์ และเมื่อได้ทำการถอดรหัสดูแล้ว ก็พบว่า สามารถทำการถอดรหัสได้ถูกต้องตามข้อมูลเดิมทุกประการ

การเพิ่ม HEADER ของไฟล์

โปรแกรมที่ได้พัฒนาขึ้นมานี้ยังได้มีการเก็บค่าต่าง ๆ ที่สำคัญของไฟล์ที่จะถูกเข้ารหัสเอาไว้ ได้แก่ ชื่อไฟล์ วันเดือนปีของไฟล์นั้น ขนาดไฟล์ วิธีเข้ารหัสข้อมูล ชื่อโปรแกรมที่ทำการเข้ารหัส Passwordของการเข้ารหัส โดยข้อมูลที่สำคัญของไฟล์เหล่านี้จะถูกเขียนไว้ก่อนข้อมูลจริง ๆ ของไฟล์ที่ถูกเข้ารหัสไว้ โดยข้อมูลที่เก็บไว้นี้ก็จะถูกเข้ารหัสไว้ด้วย เพื่อไม่ให้ผู้อื่นล่วงรู้ได้

การเก็บข้อมูลสำคัญ ๆ นี้ไว้ที่หัวไฟล์ มีประโยชน์หลายประการ เช่น เมื่อทำการถอดรหัสข้อมูลแล้วโปรแกรมก็จะทำการคืนชื่อเดิมของไฟล์ให้แก่ไฟล์ที่ถูกเข้ารหัส และทำการถอดรหัสออกมา คืนวันเวลาของไฟล์ เปลี่ยนขนาดไฟล์ให้เป็นขนาดของไฟล์เดิมได้อย่างถูกต้อง เพื่อความสะดวกสูงสุดของผู้ใช้ที่จะไม่ต้องไปจำชื่อไฟล์เดิมในกรณีที่ ชื่อไฟล์มีความสำคัญต่อการทำงานของโปรแกรมอื่น ๆ

นอกจากนี้เมื่อผู้ใช้ทำการเข้ารหัสโดยเปลี่ยนแปลงข้อมูลเดิมของ S box นั้นเมื่อจะทำการถอดรหัส ก็ไม่ต้องจำว่าได้เปลี่ยนแปลงข้อมูลตัวไหนไปบ้างก็จะสามารถ ทำการถอดรหัสได้อย่างถูกต้อง เพียงแต่ว่าจำ Password ที่ได้ใส่ไว้ตอนทำการเข้ารหัสเท่านั้นก็พอเพียงแล้ว

การเพิ่มเติมในส่วนปลีกย่อยอื่นๆ

ฟังก์ชั่นการทำงานอื่นๆที่เขียนขึ้นเป็น ฟังก์ชั่นที่ช่วยการทำงานในส่วนที่เป็นการแสดงผลทางจอภาพเพื่อให้ความสะดวก และความสวยงาม แก่ผู้ใช้งาน ส่วนที่สำคัญคือ การทำ Menu สำหรับการเลือกฟังก์ชั่นการทำงาน การแสดงข้อผิดพลาด (error) ต่างๆ ที่อาจเกิดขึ้นได้ทั้งจากผู้ใช้เองและ ทางด้านฮาร์ดแวร์ของเครื่อง

ตัวอย่างโปรแกรมที่ทำหน้าที่ด้านนี้คือ

READ.C

SCREEN.C เป็นต้น

บทที่ 5 คู่มือประกอบการใช้โปรแกรม

เป็นที่ทราบกันดีอยู่แล้วว่า โปรแกรมนี้ถูกสร้างขึ้นมาเพื่อจะใช้ในการเข้ารหัสข้อมูล เพื่อที่จะทำให้ข้อมูลนั้นเป็นความลับซึ่งมีการทำงานที่ยุ่งยากและซับซ้อน แต่ในด้านการใช้งานนั้นโปรแกรมนี้ สร้างขึ้นมาเพื่อให้ผู้ใช้สามารถใช้งานได้ง่าย เลือกการทำงานบนเมนูที่เพียงแต่เลื่อนแถบสว่างไปที่ๆต้องการแล้วกด ENTER ก็สามารถใช้งานได้แล้ว เพื่อเป็นการสะดวกแก่ผู้ใช้งาน ถึงแม้ผู้นั้นจะไม่ชำนาญในการใช้งาน คอมพิวเตอร์มาก่อนก็ตาม ก็สามารถใช้โปรแกรมนี้ได้อย่างง่ายดาย

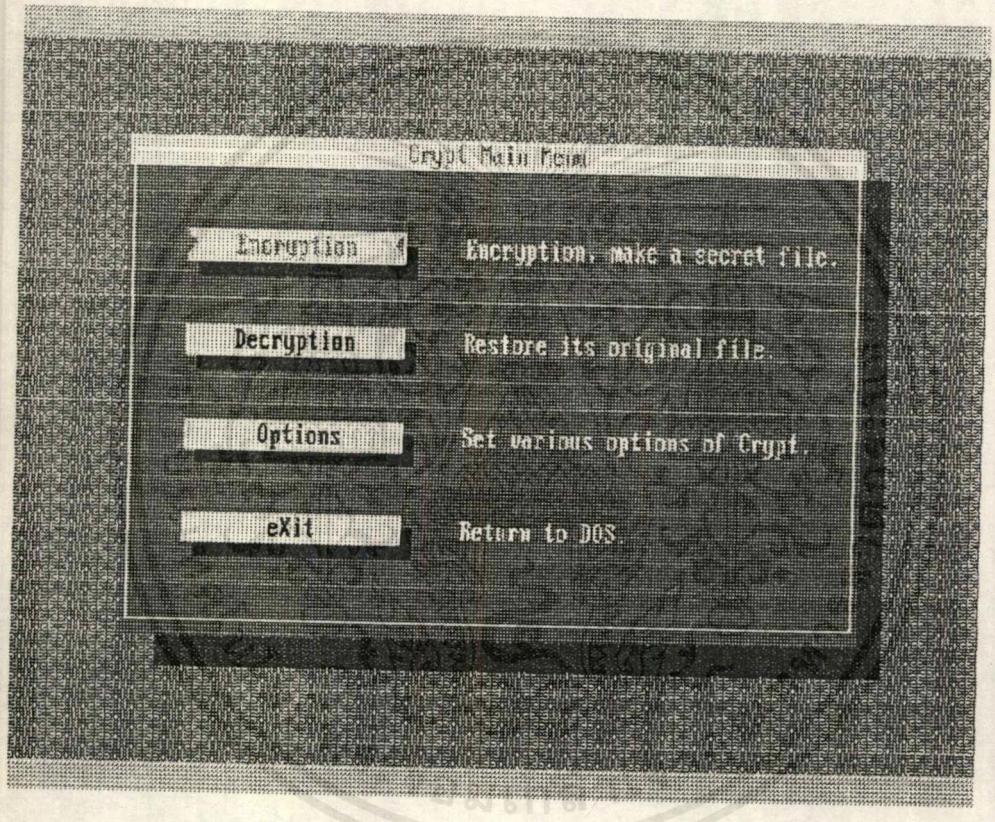
เรามาพูดถึงการใช้งานของโปรแกรมกันเลย

โปรแกรมนี้ ใช้ไฟล์ที่มีชื่อว่า CRYPT.EXE เป็นตัวทำงาน เมื่อมีไฟล์นี้อยู่แล้วหลังจากที่เครื่องพร้อมก็ให้เรียก CRYPT โดยการพิมพ์ชื่อ CRYPT แล้ว ENTER ก็จะเข้าสู่โปรแกรมจะได้หน้าจอดังรูปที่ 5.1

เป็นการบอกที่มาของโปรแกรมว่าคือ อะไร ใครเป็นอาจารย์ที่ปรึกษา และเป็นโครงงานของใคร ให้กด ENTER หรือ Space Bar ผ่านไป เพื่อจะเข้าสู่เมนเมนู

รูปที่ 5.1 ใบปะหน้าโปรแกรม

หลังจากกดแล้วก็จะเข้าสู่ เมนเมนู (Main Menu) การทำงานหลักของโปรแกรมนี้คือการเข้ารหัสข้อมูล (ENCRYPTION) เพื่อไม่ให้ผู้ที่ไม่ทราบ รหัสในการถอดข้อความสามารถล่วงรู้ข้อมูลต่างๆที่เป็นความลับได้ และ การถอดรหัสข้อมูล (DECRYPTION) เพื่อที่ผู้ที่ทำการเข้ารหัสข้อมูล หรือ เป็นผู้ที่รู้รหัสนั้นๆ จะสามารถที่จะนำข้อมูลนั้นกลับมาได้โดยที่ไม่มีข้อผิดพลาด



รูปที่ 5.2 เมนเมนู (Main Menu)

รูปที่ 5.2 คือเมนเมนูของโปรแกรมจะเห็นได้ว่ามีฟังก์ชั่นการทำงานอยู่ 4 ฟังก์ชั่นคือ

-ENCRYPTION

-DECRYPTION

-OPTIONS

-EXIT

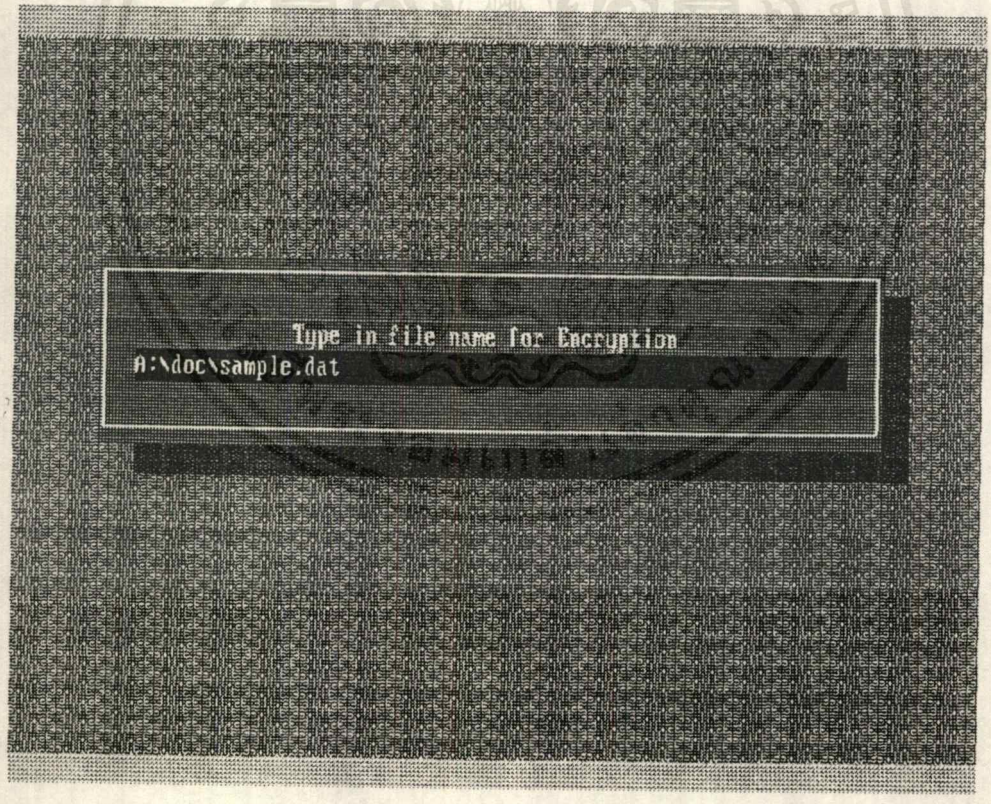
เราสามารถที่จะเลื่อนแถบสว่างที่ปรากฏอยู่ให้เลื่อนไปมาได้โดยการใช้ปุ่มลูกศรที่มีอยู่เลื่อนขึ้นบนหรือลงล่างจนถึงตำแหน่งที่เราพอใจก็ให้กด ENTER เพื่อเลือกฟังก์ชั่นนั้นให้ทำงาน

เมื่อเลือกแล้วเราจะมาดูแต่ละฟังก์ชั่นที่มีว่าใช้งานอย่างไร

ENCRYPTION

การทำงานหลักของฟังก์ชั่นนี้ก็ คือ การนำข้อมูลที่เป็นไฟล์ที่เราต้องการจะเก็บเป็นความลับมาทำการเข้ารหัส โดยจะใช้รหัส (Password)ที่ผู้ใช้ใส่เข้าไป เป็นกุญแจหลัก (Main Keys)ในการกระทำขบวนการเข้ารหัส ตามอัลกอริทึ่มที่เราได้กำหนด

เมื่อเลือกฟังก์ชั่นนี้แล้วจะปรากฏหน้าจอดัง รูปที่ 5.3



รูปที่ 5.3 แสดงการใส่ชื่อไฟล์ที่ต้องการเข้ารหัส (ต้นฉบับ)

โปรแกรมจะให้เราใส่ชื่อไฟล์ที่เราต้องการจะ ENCRYPT ลงไปแล้ว ENTER

การใส่ชื่อไฟล์นั้นกระทำเช่นเดียวกับ DOS ทุกประการ คือจะต้องมี ดิสก์ไดร์ฟ ไดเรคทอรี่ และชื่อไฟล์ที่ถูกต้อง ซึ่งถ้ามีตัวใดผิดพลาดหลังจากการ ENTER ก็จะมีข้อผิดพลาด บอกให้ทราบ

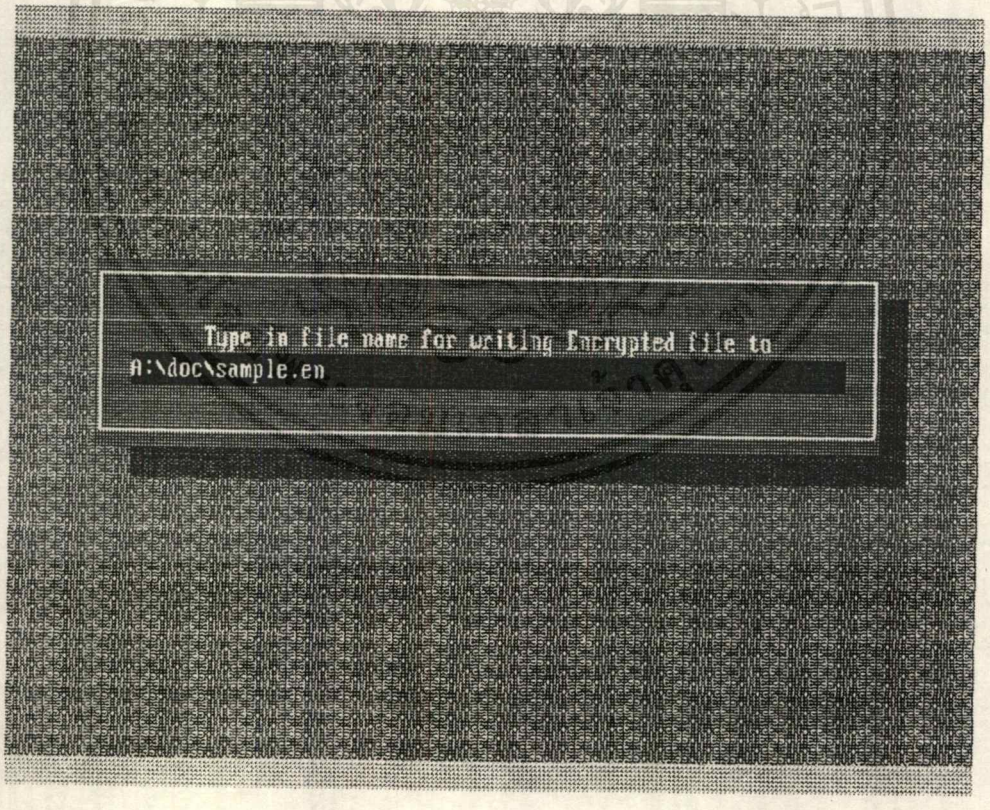
จากรูปจะเห็นไฟล์ตัวอย่างที่ที่ใส่ไว้ให้ดู คือ

A:\doc\sample.dat

A: คือ ดิสก์ไดร์ฟ (Disk Drive) ที่ไดเรคทอรี่ (Directory) หรือไฟล์นั้นอยู่ ถ้าไฟล์ที่จะเรียกอยู่ที่ไดร์ฟที่ใช้งาน (CURRENT DRIVE) ก็ไม่จำต้องใส่

\doc คือ ไดเรคทอรี่ ที่ไฟล์นั้นอยู่

\sample.dat คือชื่อของไฟล์ (file) ที่เป็นไฟล์ที่เรากำหนดให้เป็นต้นฉบับของการ ENCRYPTION



รูปที่ 5.4 การใส่ชื่อไฟล์ที่จะนำไฟล์ที่ผ่านการเข้ารหัสแล้วไปเก็บ

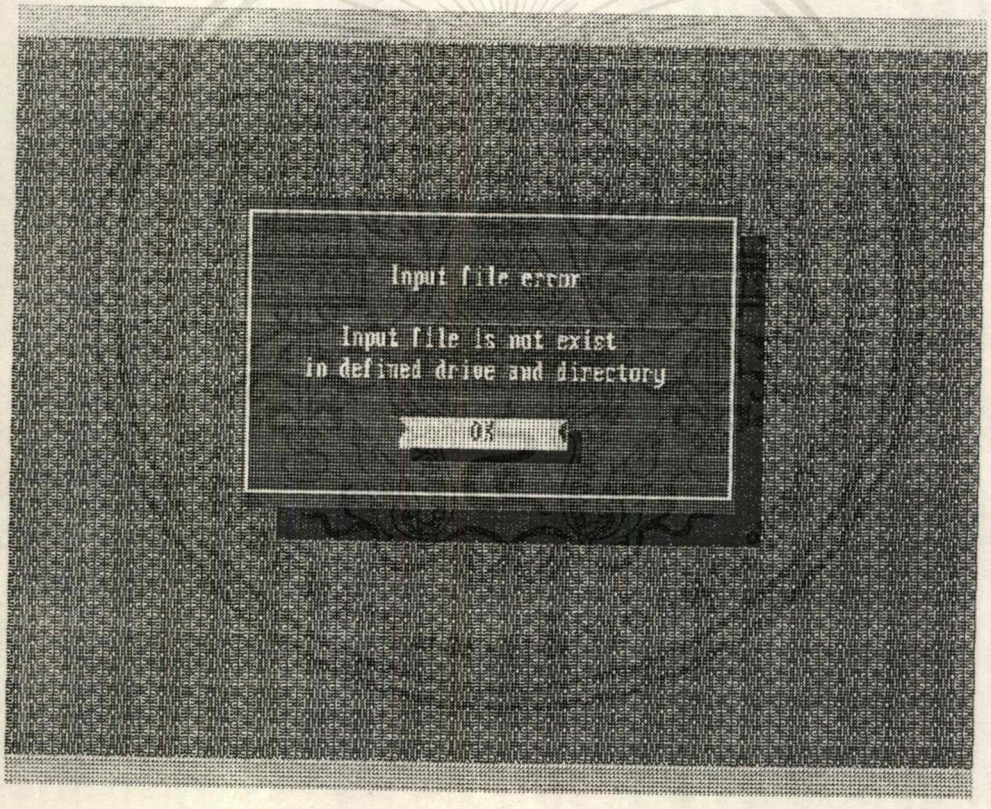
จากนั้นเราจะต้องใส่ชื่อไฟล์ที่จะนำไฟล์ที่ถูก ENCRYPTION แล้ว ไปเก็บไว้เมื่อใส่ชื่อเรียบร้อยแล้วให้ ENTER จะแสดงหน้าจอดังรูปที่ 5.4

จากรูปเรากำหนดไฟล์ตัวอย่างเป็น

A:\doc\sample.en

รูปแบบจะเป็นเช่นเดียวกับ การใส่ชื่อไฟล์ของการ ENCRYPTION

การใส่ชื่อนี้อาจเกิดการผิดพลาดได้ เช่น ชื่อที่เป็น input file ที่ใส่ชื่ออาจจะไม่มีอยู่ในแผ่น หรือ รากที่กำหนดผิด จะมีการบอกให้ทราบ ดังรูปที่ 5.5

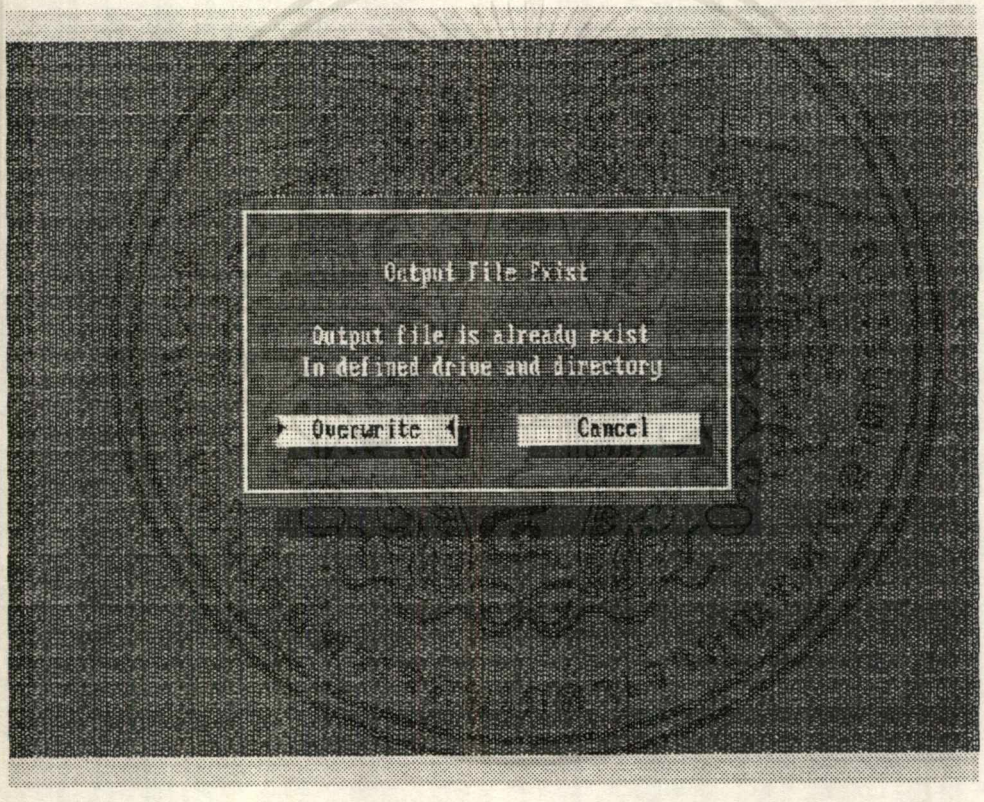


รูปที่ 5.5 การแสดงข้อผิดพลาดเนื่องจากไม่มีไฟล์ต้นฉบับอยู่

หรือถ้า output file ที่กำหนดไว้มีอยู่แล้วก็จะมี ข้อผิดพลาดแสดง ขึ้นมาบอกและถามผู้ที่จะ ENCRYPT ว่าจะต้องการเขียนทับลงไปหรือไม่ดังรูป

ถ้าเลือกให้เขียนทับก็จะทำงานต่อไป แต่ถ้าไม่ ก็จะกลับไปที่เมนเมนู

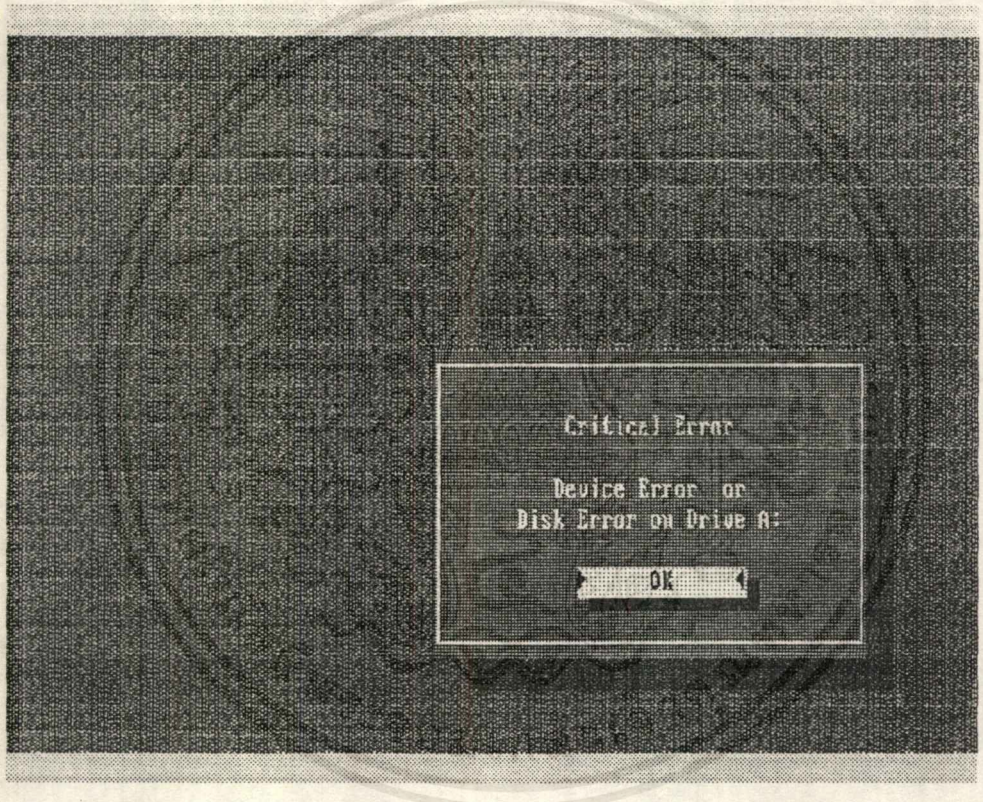
การเลือกเขียนทับ จะต้องระวังเพราะถ้าเกิด ความผิดพลาดแล้วจะทำให้ไฟล์นั้นๆเสียหายได้ จะต้องแน่ใจว่าเป็นไฟล์ที่เราเขียนถูกต้องจริงๆ



รูปที่ 5.6 แสดงข้อที่ว่าชื่อไฟล์เป้าหมายนั้นมี
และต้องการจะเขียนทับหรือไม่

ถึงตอนนี้ถ้าแผ่นดิสก์ที่จะเขียนไฟล์ลงไปมีการ write protect โปรแกรมจะทำการตรวจเช็ค เนื่องจากว่าโปรแกรมไม่สามารถที่จะสั่งให้มีการเขียนข้อมูลที่ได้ ENCRYPT แล้วลงไปในดิสไดฟ์ที่เป็นอุปกรณ์ทาง ฮาร์ดแวร์ได้ ถ้าหากไม่มีการตรวจสอบ อาจทำให้เครื่องคอมพิวเตอร์เกิดการแฮงค์ได้ทำให้เสียเวลาและเสียสุขภาพจิตได้

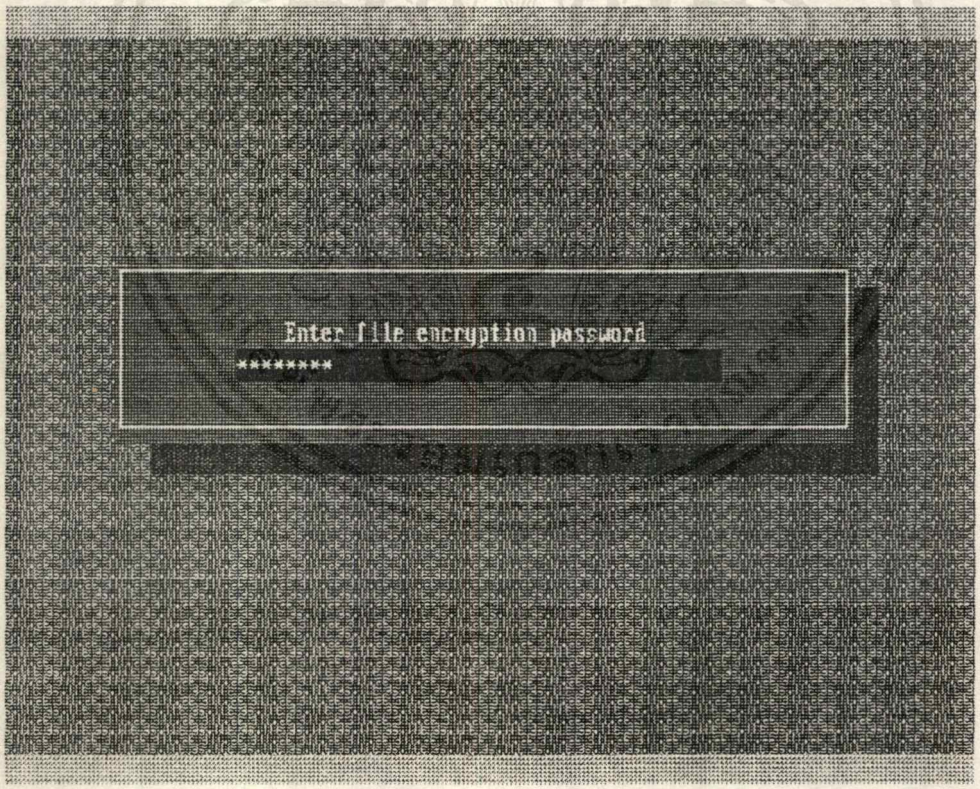
โปรแกรมจะแสดงแสดง ข้อผิดพลาด ดังรูปตัวอย่างที่ 5.7



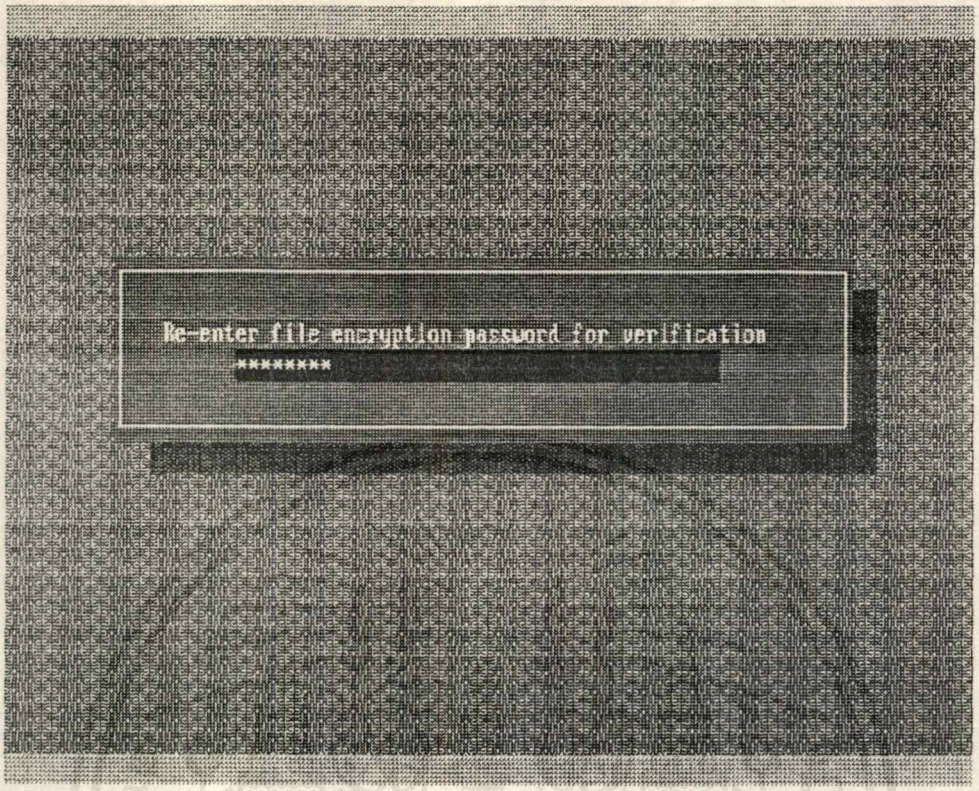
รูปที่ 5.7 แสดงข้อผิดพลาดเนื่องจาก ดิสก์มีการ
write protect หรือ drive error

หลังจากใส่ชื่อไฟล์เรียบร้อยแล้วก็จะเข้ามาที่การใส่ Password ซึ่งเป็นส่วนสำคัญมากที่สุดของการเข้ารหัส รหัสที่เราใส่ไปจะเป็น คีย์เวิร์ด ในการเข้ารหัสต่อไป รหัสที่ใช้นี้เป็นได้ทั้งตัวอักษร และตัวเลข หรือรหัส ASCII อื่นๆซึ่งมี 8 บิท เหมือนตัวอักษรหรือตัวเลขที่ใช้อยู่

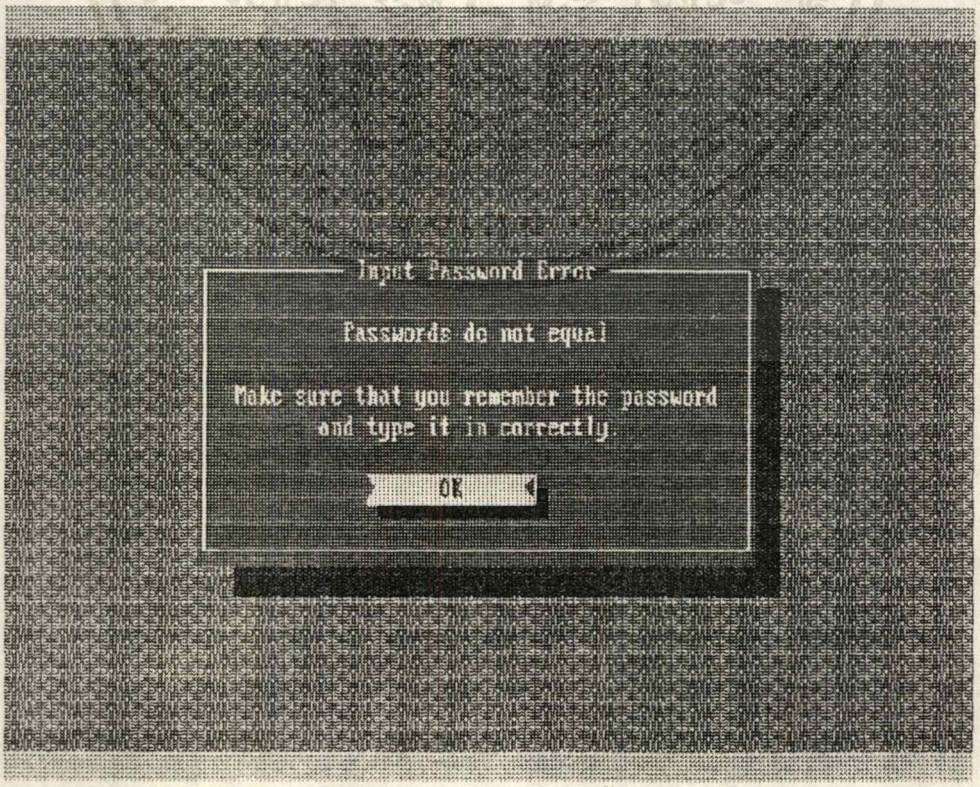
การใส่รหัสของโปรแกรมนี้จะใส่ อย่างน้อย 8 ตัว และรหัสนี้จะใช้ในการถอดรหัสด้วย เนื่องจากรัสที่ใส่ในการ ENCRYPTION นั้นสำคัญมาก เราจึงให้มีการถามย้ำอีกครั้งหนึ่งการใส่รหัสนั้นจะไม่ให้เห็นตัวหนังสือที่ใส่ลงไปการกดรหัสจึงจะต้องทำอย่างถูกต้อง ถ้าการใส่ทั้งสองครั้งไม่ตรงกันก็จะให้ผู้ใช้โปรแกรมได้ใส่ใหม่ให้ถูกต้องตรงกันทั้ง 2 ครั้ง โปรแกรมจึงจะเริ่มทำงานต่อไปได้



รูปที่ 5.8 การ Password ครั้งแรก



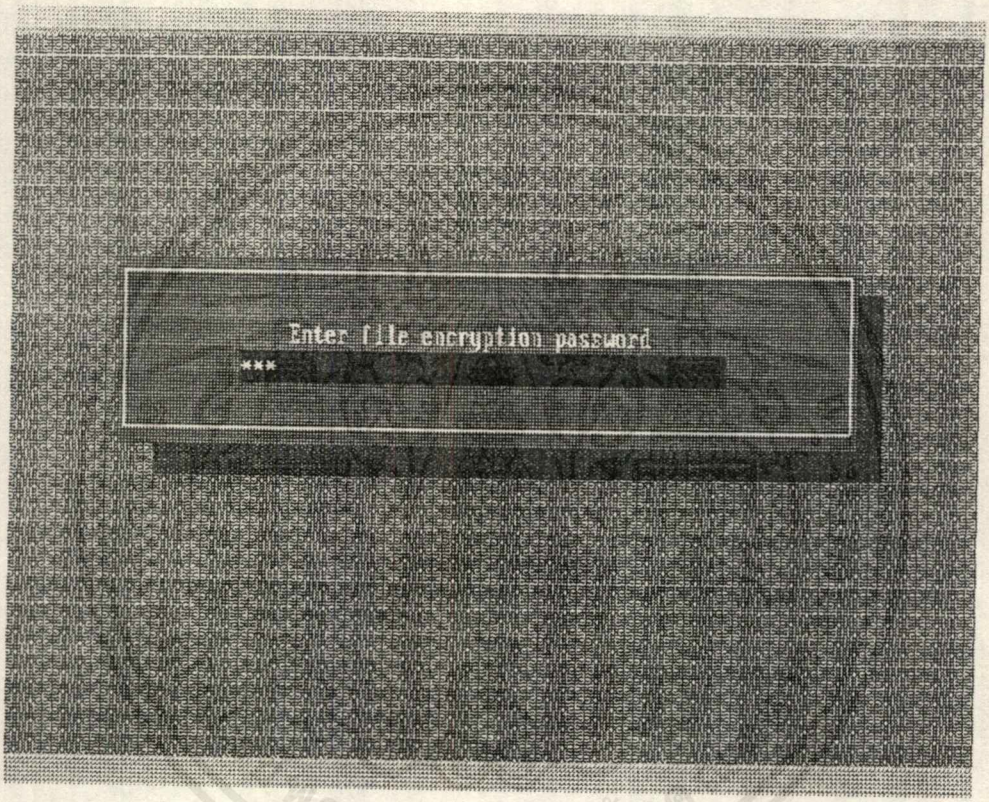
รูปที่ 5.9 การใส่ Password ครั้งที่ 2



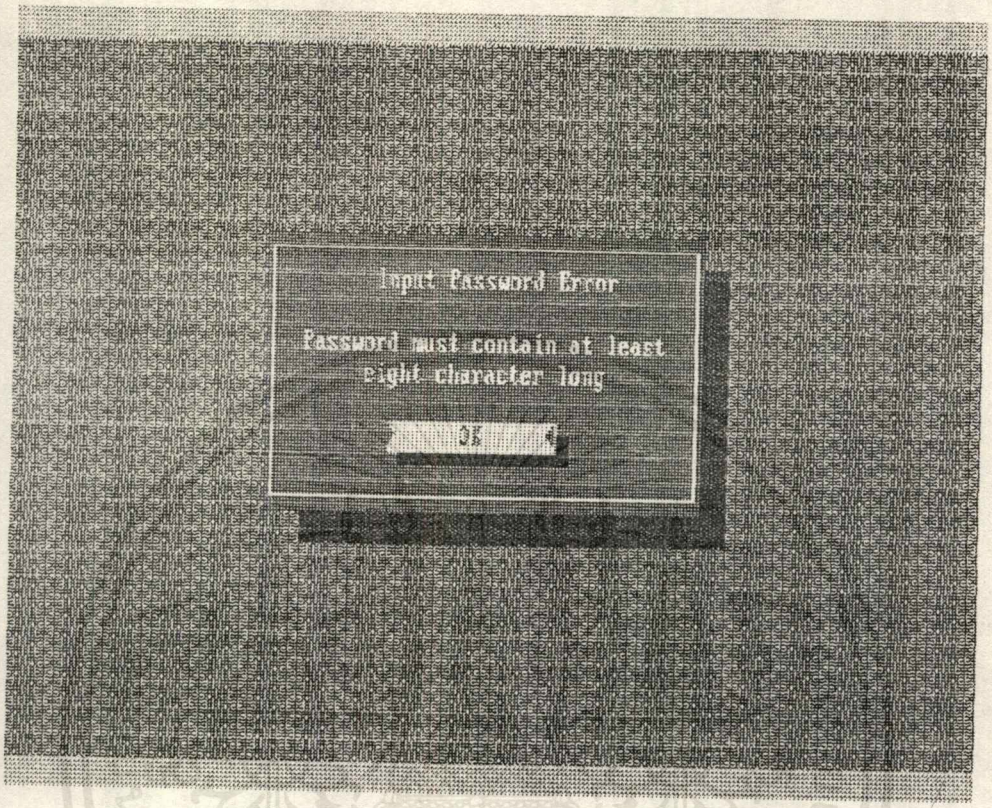
รูปที่ 5.10 แสดงว่า Password ทั้ง 2 ครั้งไม่ตรงกัน

จากตัวอย่างการใส่รหัสถ้าใส่รหัสไม่ถึง 8 ตัว ก็จะมี ข้อผิดพลาดแสดง ขึ้นดังรูปที่ 5.12 แล้วจะให้ผู้ใช้ได้กลับไปเริ่มใส่ Password ใหม่

ดังตัวอย่างที่ใส่ Password เพียงแค่ 3 ตัวไม่เพียงพอก็จะแสดงข้อผิดพลาดนั้นออกมาให้เห็นดังรูปที่ 5.11



รูปที่ 5.11 การใส่ Password ไม่ครบ 8 ตัว

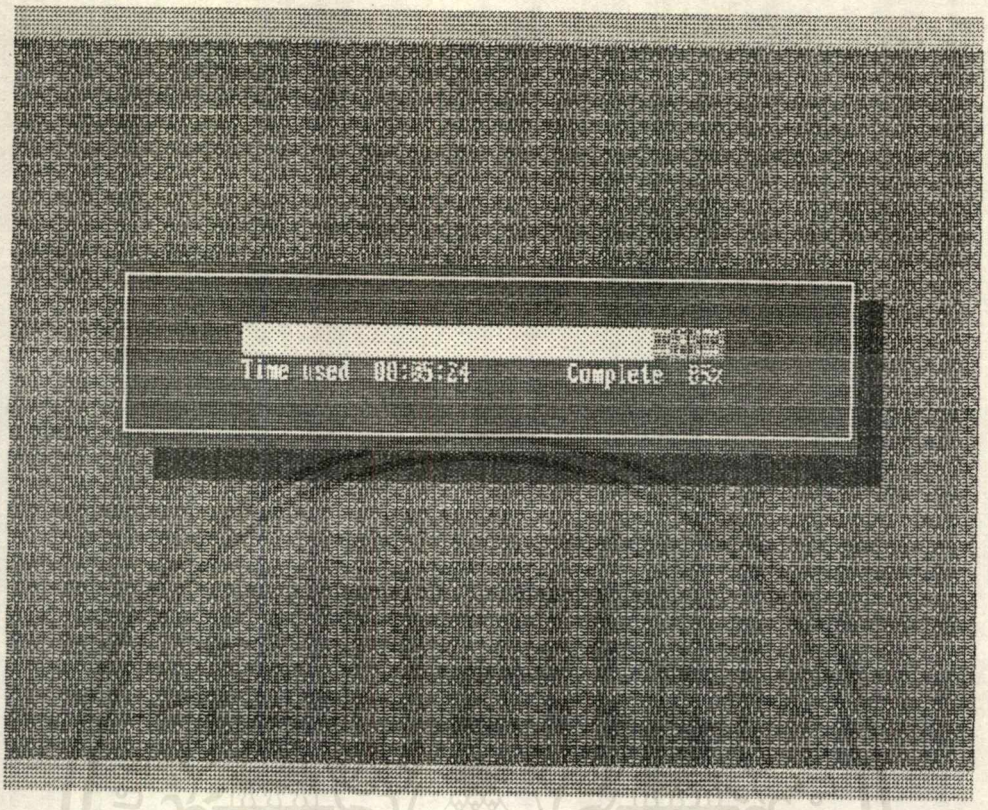


รูปที่ 5.12 แสดงข้อผิดพลาดเนื่องจากการใส่ ที่ไม่ครบถ้วน

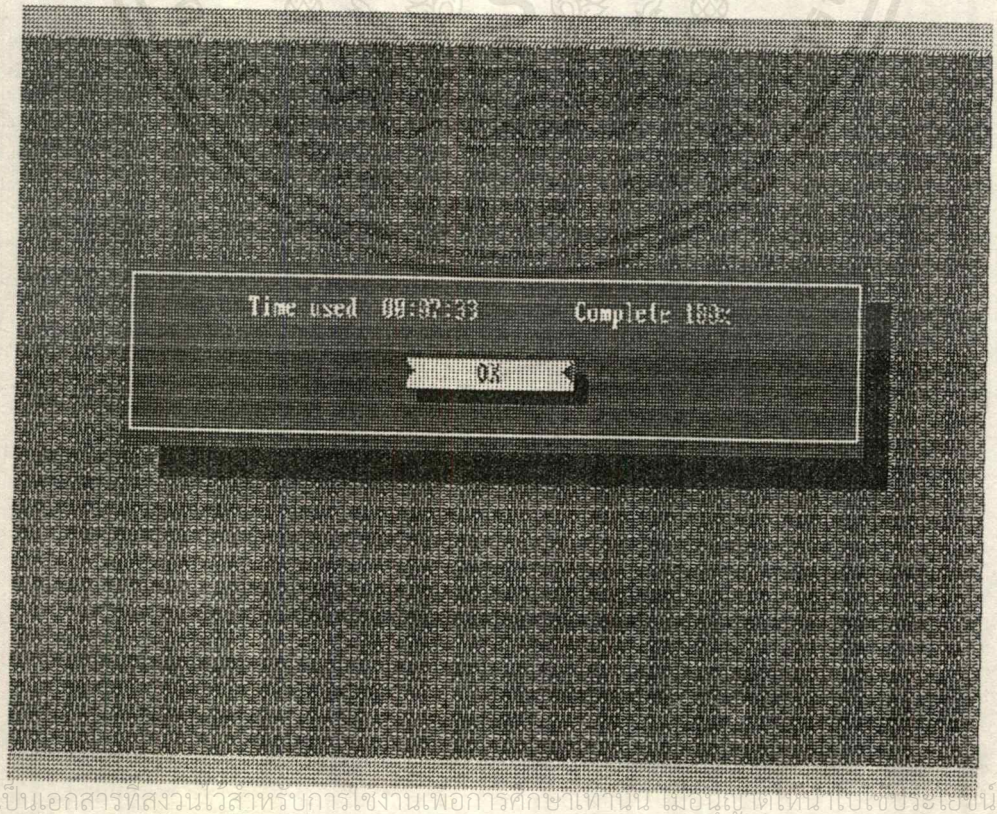
เมื่อทุกอย่างเรียบร้อย ENTER ก็จะเริ่มทำการ ENCRYPTION โปรแกรมที่ต้องการจะเข้ารหัส

โดยจะมีเวลาในการ ENCRYPT และ % ความสำเร็จ บอกให้ผู้ใช้ทราบดังรูปที่ 5.13

และหลังจากการสำเร็จเรียบร้อยแล้วก็จะบอกเวลาที่ใช้ทั้งหมดให้ทราบ หน้าจอนี้แสดงว่าการ ENCRYPTION ได้เรียบร้อยแล้ว เพียงแค่เคาะผ่านเพื่อกลับไปยัง เมนเมนู เพื่อทำงานต่อหรือเลิกการทำงาน ดังรูปที่ 5.14



รูปที่ 5.13 แสดง % ความสำเร็จและเวลาที่ใช้



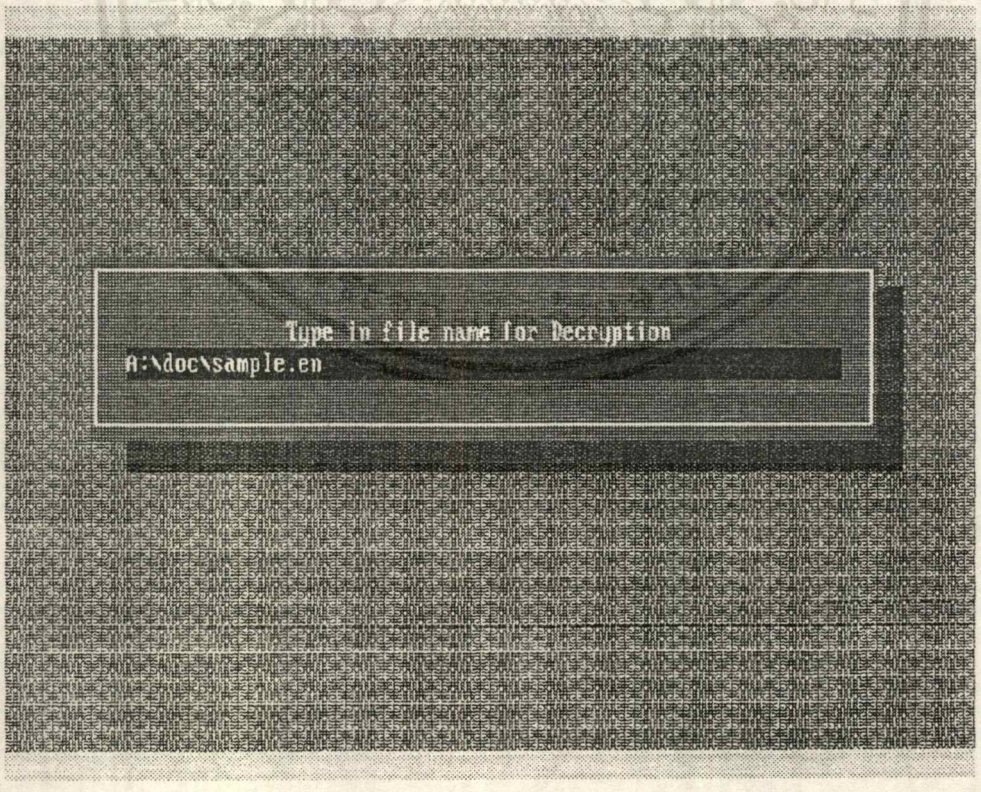
รูปที่ 5.14 ภาพเมื่อโปรแกรมทำงานสำเร็จแล้ว

DECRYPTION

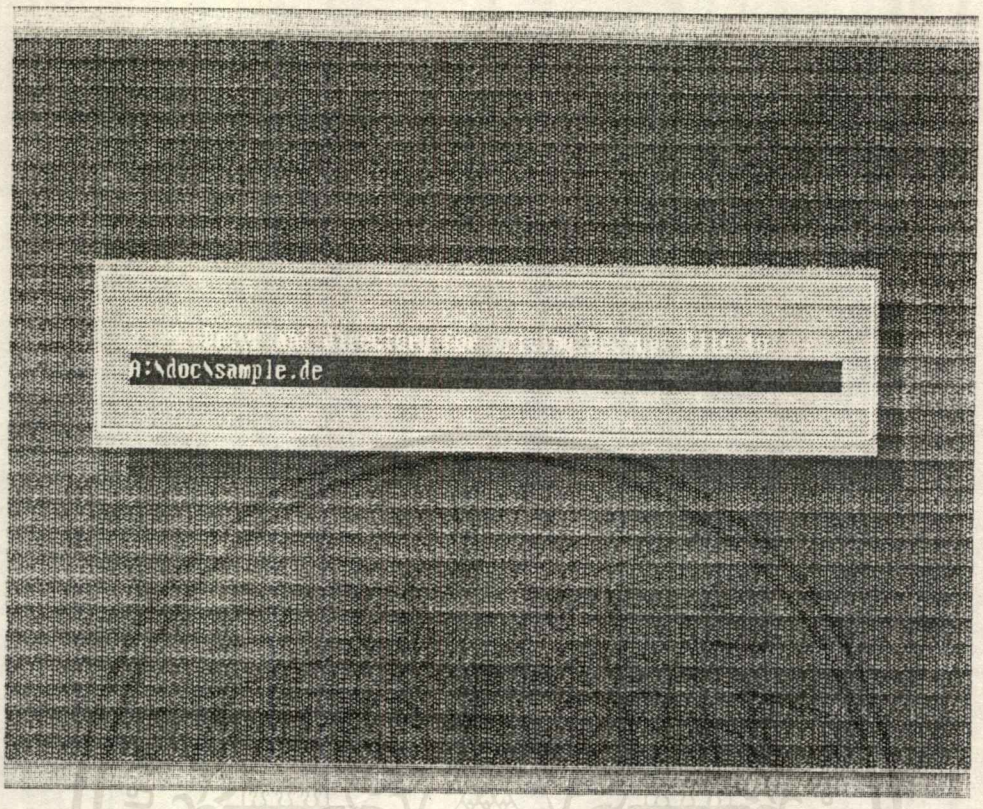
เมื่อเลือกฟังก์ชั่นนี้ ก็คือเราต้องการให้โปรแกรมทำการ DECRYPTION ไฟล์ที่ได้ถูกทำการ ENCRYPTION ไปแล้วให้กลับมาอยู่ในรูปแบบเดิมก่อนการทำ ENCRYPTION ซึ่งรูปแบบของการ DECRYPTION จะต้องตรงกับการ ENCRYPTION ที่ได้ทำมาก่อนหน้านี้

รูปแบบของการทำ DECRYPTION นั้นโปรแกรมสามารถที่จะอ่านได้จาก หัวไฟล์ (Header)ของโปรแกรมที่ได้ถูก ENCRYPTION แล้วซึ่งเป็นรูปแบบที่ผู้สร้างโปรแกรมได้กำหนดขึ้นเพื่อความสะดวกต่อผู้ใช้งาน

หลังจากเลือกแล้วจะได้หน้าจอดังรูปคล้ายกับการทำ ENCRYPTION คือให้ใส่ชื่อไฟล์ที่ต้องการจะ DECRYPTION แล้วใส่ชื่อไฟล์ที่ต้องการจะให้นำข้อมูลที่ผ่านการถอดรหัสนั้นไปเก็บไว้โดยจะต้องกำหนดไดฟ์ไว้ด้วย แต่ถ้ามีการกำหนดไดฟ์แต่ไม่กำหนดชื่อไฟล์ก็สามารถทำงานได้ เนื่องจากเราได้เก็บข้อมูลของชื่อไฟล์ที่เป็นต้นฉบับไว้แล้ว โปรแกรมจะทำการใส่ไฟล์ที่ทำการ DECRYPTION แล้วลงไปในชื่อของไฟล์เดิม ดังรูปที่ 5.15และ 5.16



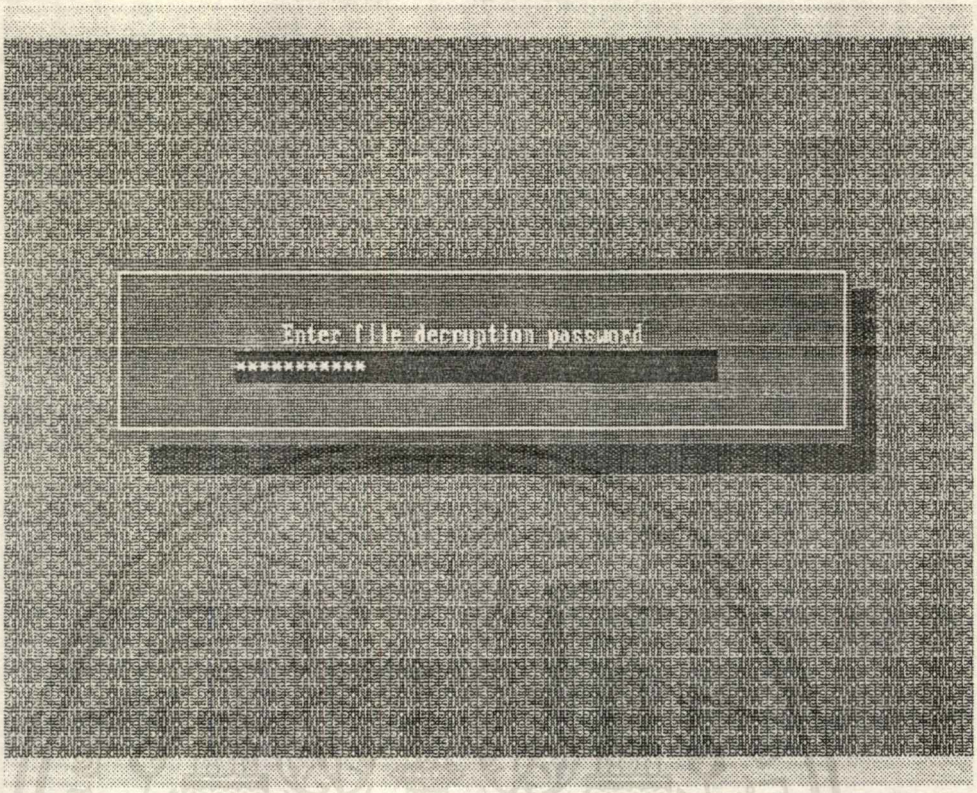
รูปที่ 5.15 การใส่ไฟล์ที่ต้องการถอดรหัส



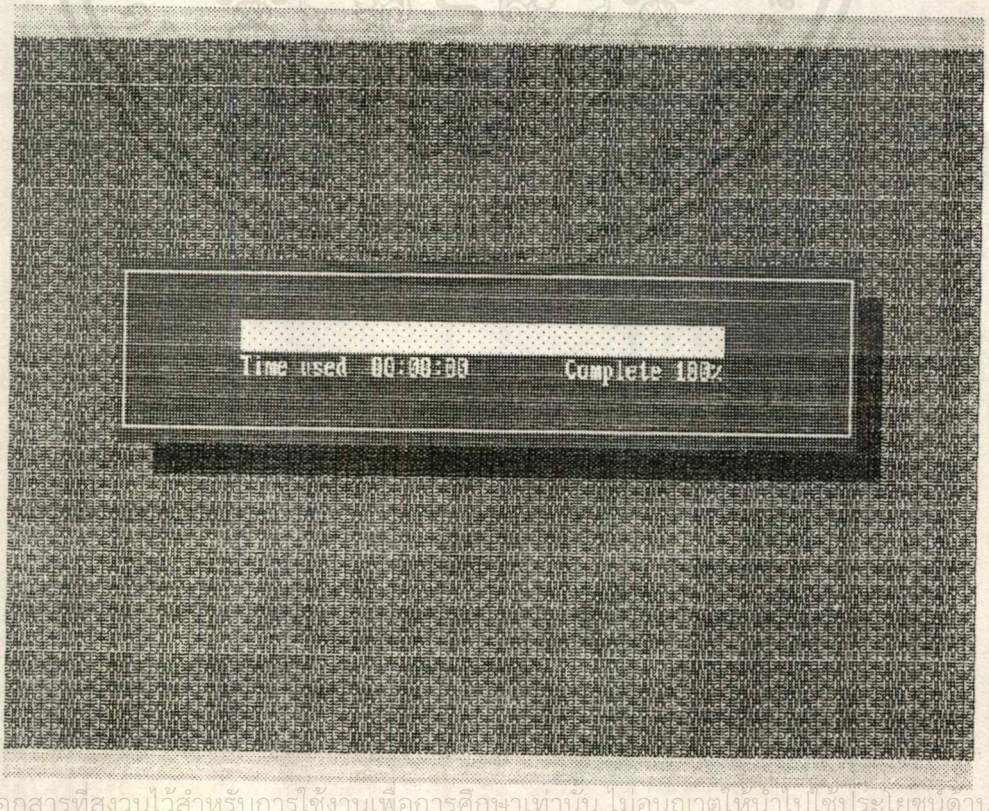
รูปที่ 5.16 การใส่ชื่อไฟล์เป้าหมายที่จะนำข้อมูลที่ผ่านการถอดรหัสแล้วไปเขียนเก็บไว้

ต่อมาก็จะเป็นการใส่ Password โดยจะใส่เพียงครั้งเดียวไม่มีการทวนเหมือนการ ENCRYPTION ดังรูปที่ 5.17

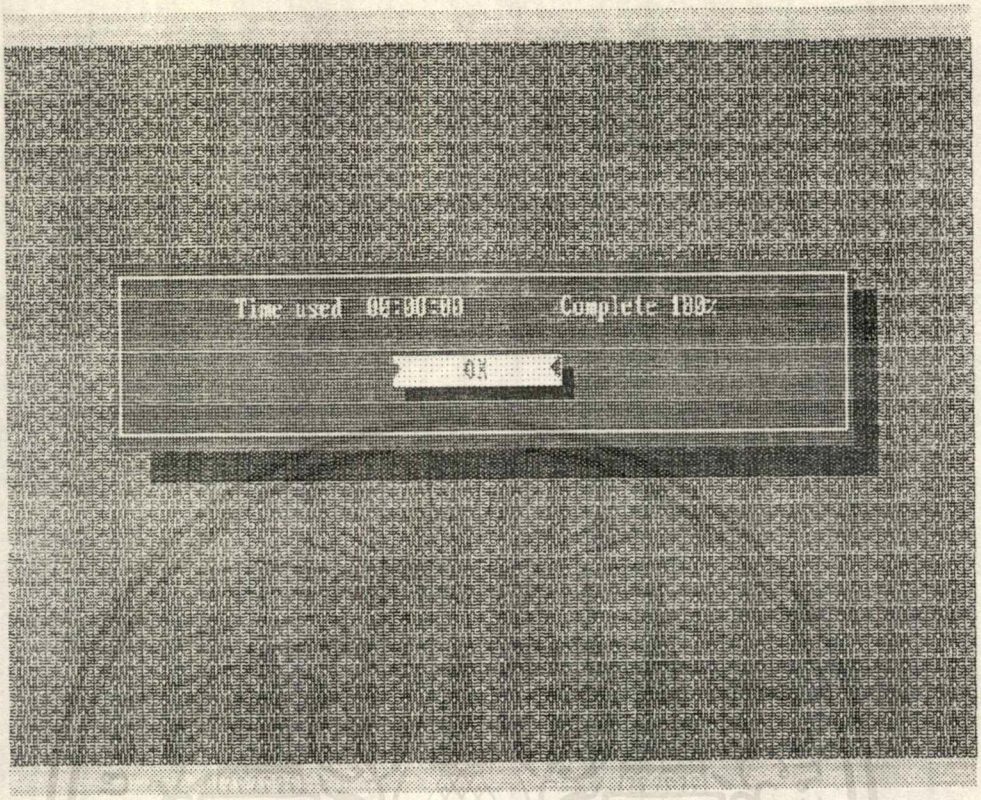
ถ้าใส่ Password ถูกโปรแกรมจะเริ่มทำงานต่อไป โดยที่ในระหว่างการทำงานของโปรแกรมจะมีหน้าจอที่แสดง เวลาและเปอร์เซนต์ความสำเร็จ จนกระทั่งไฟล์ที่ทำการ DECRYPTION นั้นเสร็จ จะมีการแสดง เช่นเดียวกับการทำงานของฟังก์ชั่น ENCRYPTION ดังรูปที่ 5.18 และ รูปที่ 5.19



รูปที่ 5.17 การใส่ Password ของการ DECRYPTION



รูปที่ 5.18 แสดง % และเวลาความสำเร็จของการทำงาน

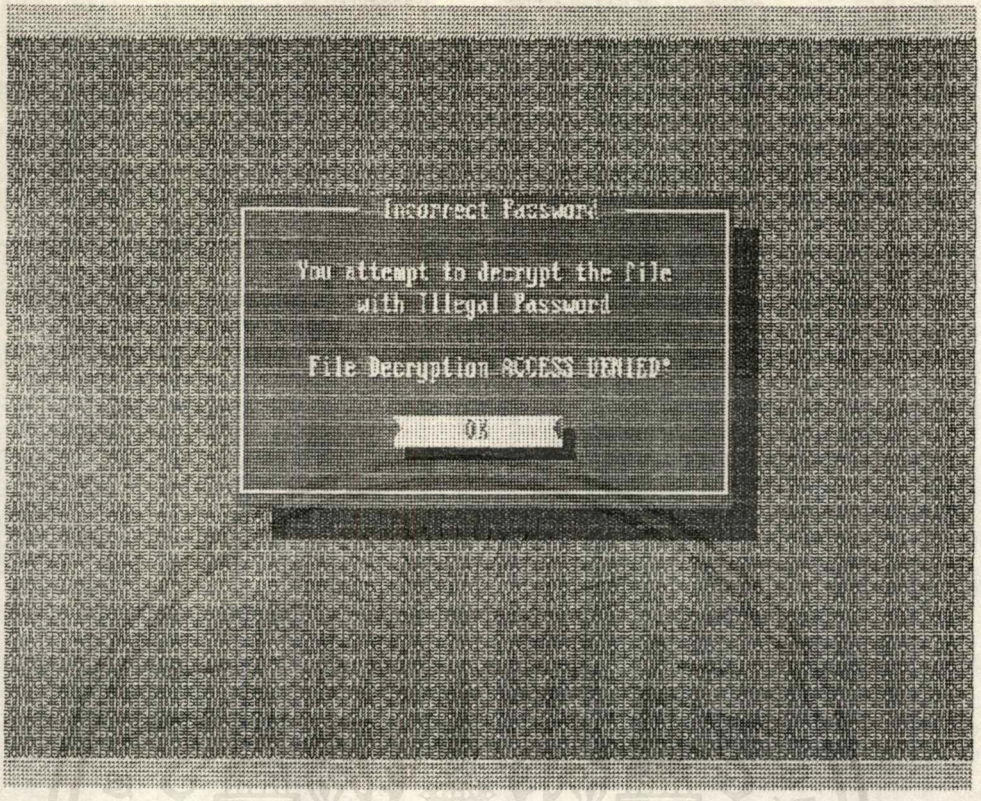


รูปที่ 5.19 แสดงว่างานเสร็จเรียบร้อย พร้อมเวลาที่ใช้

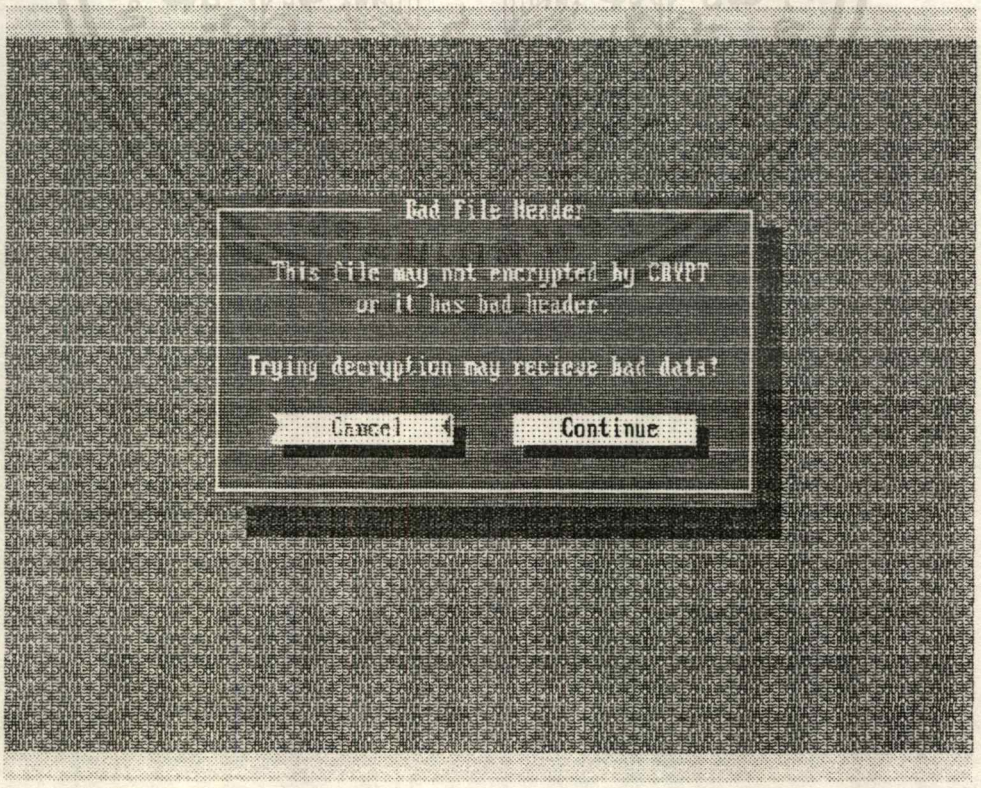
แต่ถ้าผู้ใช้ใส่ Password ไม่ถูกต้องตรงกับ Password ที่ใส่ไว้ก่อนใน การ ENCRYPTION โปรแกรมจะแสดงออกมาดังรูปที่ 5.20

แล้วจะกลับไปที่เมนเมนูเพื่อเลือกการทำงานใหม่

ข้อผิดพลาด อีกแบบที่จะแสดงให้เห็นว่าไฟล์ที่จะนำมา DECRYPTION นั้นเป็นไฟล์ที่ไม่มีหัวไฟล์ (Header) ซึ่งไฟล์ที่ผ่านการทำ ENCRYPTION ทุกไฟล์จะต้องมีอยู่ โดยที่หัวไฟล์นั้นจะมีชื่อ โปรแกรม Crypt 1.00 ะ อยู่เพื่อให้ทราบว่าเป็นไฟล์ที่ผ่านการเข้ารหัส ถ้ายังจะทำการ DECRYPTION ต่อไปไฟล์ที่ได้ออกมาจะเป็น ไฟล์ที่ผิดพลาดมาก ไม่ว่าจะเป็นทางด้านข้อความ หรือ ขนาดของไฟล์ ที่ควรทำก็คือ เลือก Cancel เพื่อจะออกมาตรวจดูอีกครั้งว่าใช่ไฟล์ที่เราต้องการจะ DECRYPTION จริงหรือไม่ ดังรูปที่ 5.21



รูปที่ 5.20 แสดงความผิดพลาดเนื่องจากใส่ Password ผิด



รูปที่ 5.21 แสดงไฟล์ที่ต้องการถอดรหัสนั้นอาจไม่ผ่านการเข้ารหัสมา

ถ้าแน่ใจว่าถูกต้องก็จึงเข้าสู่การ DECRYPTION ใหม่ แล้วเลือก ไปที่ Continue เพื่อทำงานตามที่เราต้องการต่อไป

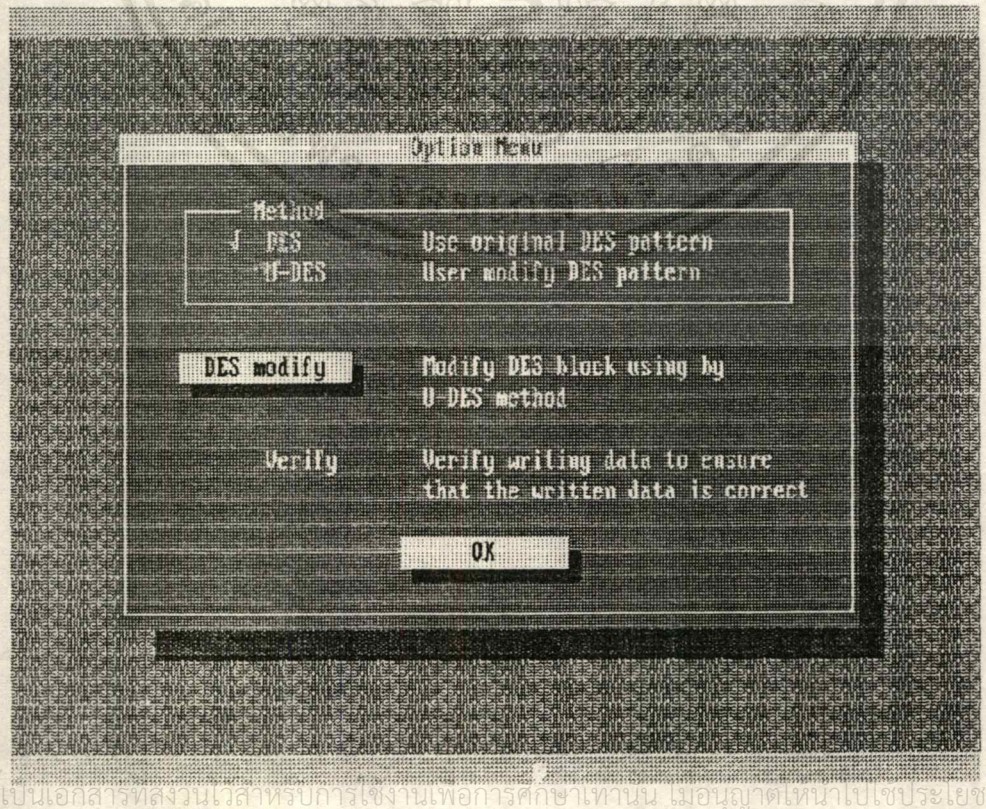
ส่วนข้อผิดพลาด อื่นๆซึ่งเป็นข้อปลีกย่อย เช่น ดิสก์หรือไดร์ฟ error ก็จะแสดงดังรูปเช่นเดียวกับการ ENCRYPTION

OPTIONS

ฟังก์ชั่นนี้จะเป็นตัวเลือกที่จะทำให้การ ENCRYPTION และ DECRYPTION กระทำต่อไฟล์ด้วยค่าที่แตกต่างกันออกไป ของการสลับบิทภายใน S-boxes ซึ่งเป็น อัลกอริทึ่มภายในของการเข้ารหัสด้วยระบบ DES เราจะเรียกการเข้ารหัสที่เข้าไปเปลี่ยนแปลงภายในของ S-boxes ว่า U-DES ซึ่งมาจาก User modify DES pattern

นอกจากนี้ยังมีให้เลือก ว่าต้องการการตรวจสอบของข้อมูลระหว่างการเขียนลงในดิสก์หรือไม่อีกด้วย ซึ่งเราจะเข้าไปดูการทำงานของฟังก์ชั่นนี้กัน

เมื่อเลื่อนแถบสว่างมาที่ Options แล้ว ENTER ก็จะเข้าสู่ Options Menu หน้าตาของ Options Menu เป็นดังรูปที่ 5.22



รูปที่ 5.22 Option menu

จาก เมนูจะมีฟังก์ชั่นให้เลือกอยู่ 3 แบบ คือ

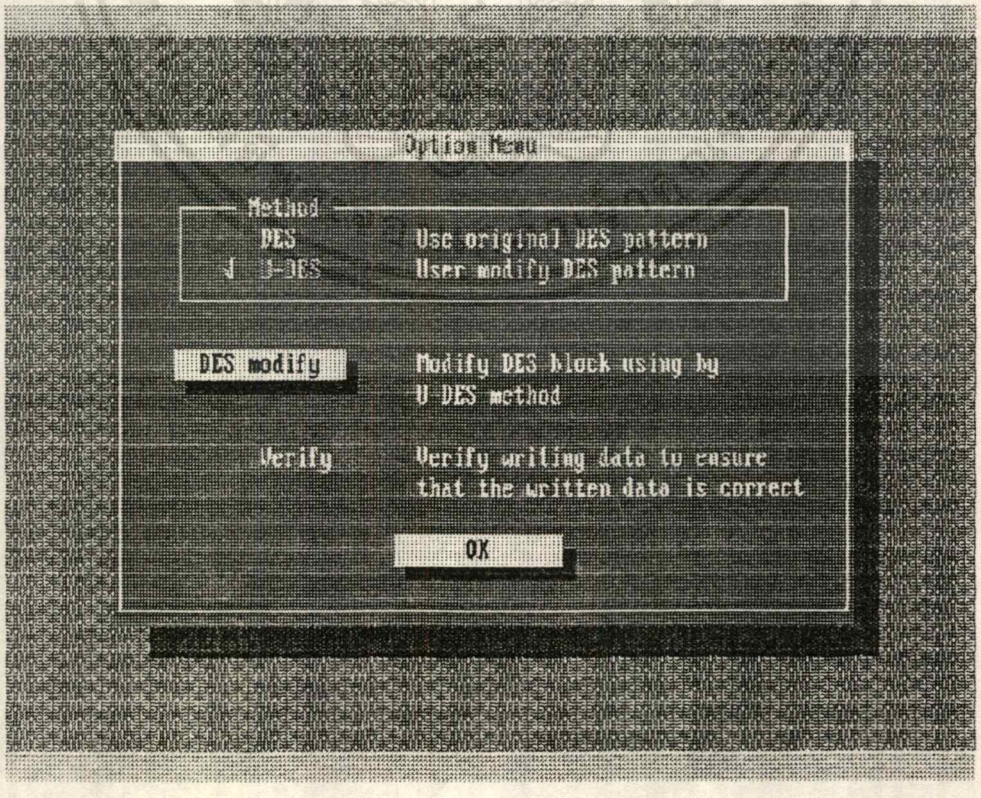
-Method ซึ่งจะเชื่อมโยงกับ -DES Modify และ

-Verify

Method คือ การเลือกรูปแบบของการ ENCRYPTION หรือ DECRYPTION ว่าต้องการในแบบที่เป็น DES มาตราฐาน ซึ่งมีค่าของ S-boxes ที่คงที่ตายตัวอยู่แล้วภายในโปรแกรมหรือจะเป็นแบบ U-DES ที่จะไปใช้ S-boxes ในส่วนที่เราเปลี่ยนแปลงหรือแก้ไขเอาไว้ในช่วงของการทำงานของ ฟังก์ชั่น DES modify ที่จะกล่าวถึงต่อไป

การเลือกก็เพียงแต่ เลื่อนแถบสว่างไปยังที่ๆต้องการเมื่อ ENTER ก็จะมีเครื่องหมาย √ ค้างอยู่ที่หน้าตัวเลือก DES หรือ U-DES โดยปกติแล้วเมื่อเริ่มใช้งานโปรแกรมจะตั้งให้ลูกศรอยู่ที่ ตำแหน่ง DES โดยดูจากรูปแรกที่เข้ามาที่หน้าจอนี้

การเลือกนี้จะเป็นผลต่อการทำ ENCRYPTION และ DECRYPTION ด้วยกล่าวคือถ้าลูกศรอยู่ที่ตำแหน่งใด ในขณะที่ใช้งานฟังก์ชั่น ENCRYPTION ก็จะทำการ ENCRYPT ด้วยการใช้ S-boxes ในรูปแบบนั้น ส่วน ฟังก์ชั่น DECRYPTION รูปแบบที่ใช้จะถูกเก็บไว้ที่หัวไฟล์ด้วย จึงจำเป็นต้องเลือกให้ถูกต้อง มิฉะนั้นจะไม่สามารถทำการ DECRYPTION ได้ ต้องออกมาเลือกใหม่ให้ถูกต้องด้วย

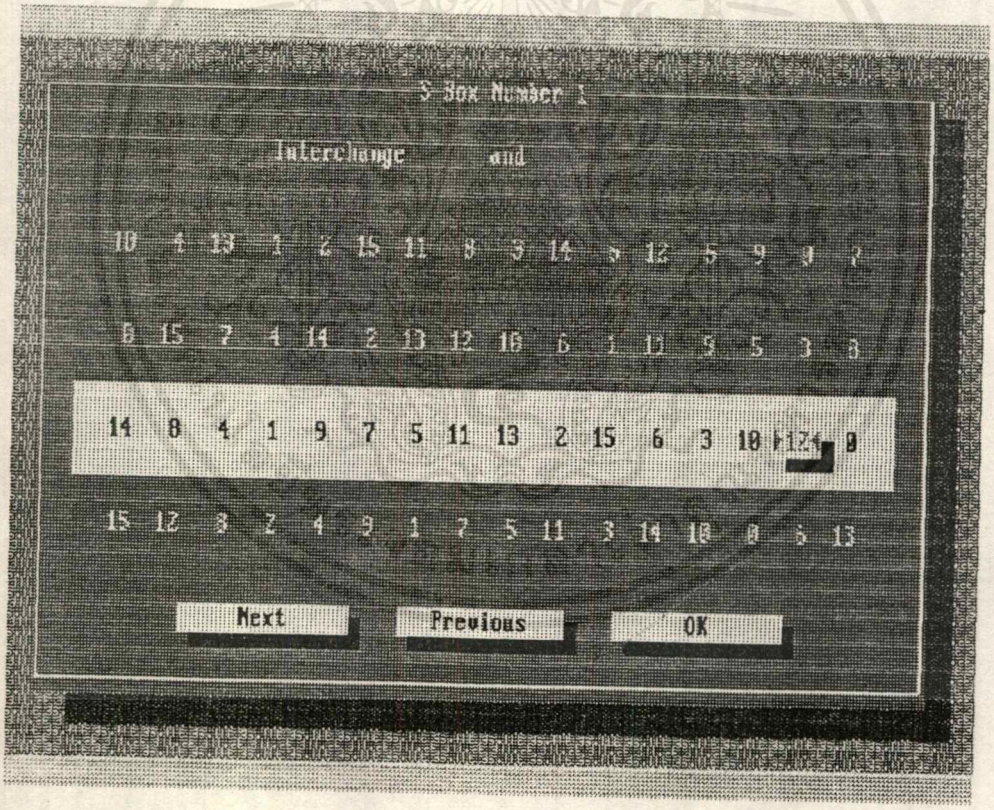


รูปที่ 5.23 การเลือก U-DES Method

จากรูปจะเห็นได้ว่า เราสามารถเปลี่ยนตำแหน่งของเครื่องหมาย / ในกรอบของฟังก์ชั่น Method ให้มาอยู่ที่ U-DES ได้โดยการกระทำตามที่ได้กล่าวไว้ข้างต้น

DES modify คือ การเปลี่ยนแปลงแก้ไขค่าภายใน S-boxes ซึ่ง S-boxes ที่ได้รับการแก้ไขแล้วนั้นจะถูกนำไปใช้เมื่อ มีการเลือกใช้ U-DES Method ในการ ENCRYPTION หรือ DECRYPTION ถ้าเป็นการเรียกใช้ DES ก็จะไม่จำเป็นต้องเรียกฟังก์ชั่นนี้ขึ้นมาใช้งาน

เมื่อเลือกฟังก์ชั่นนี้ให้ทำงานจะปรากฏหน้าจอ ดังรูปที่ 5.24

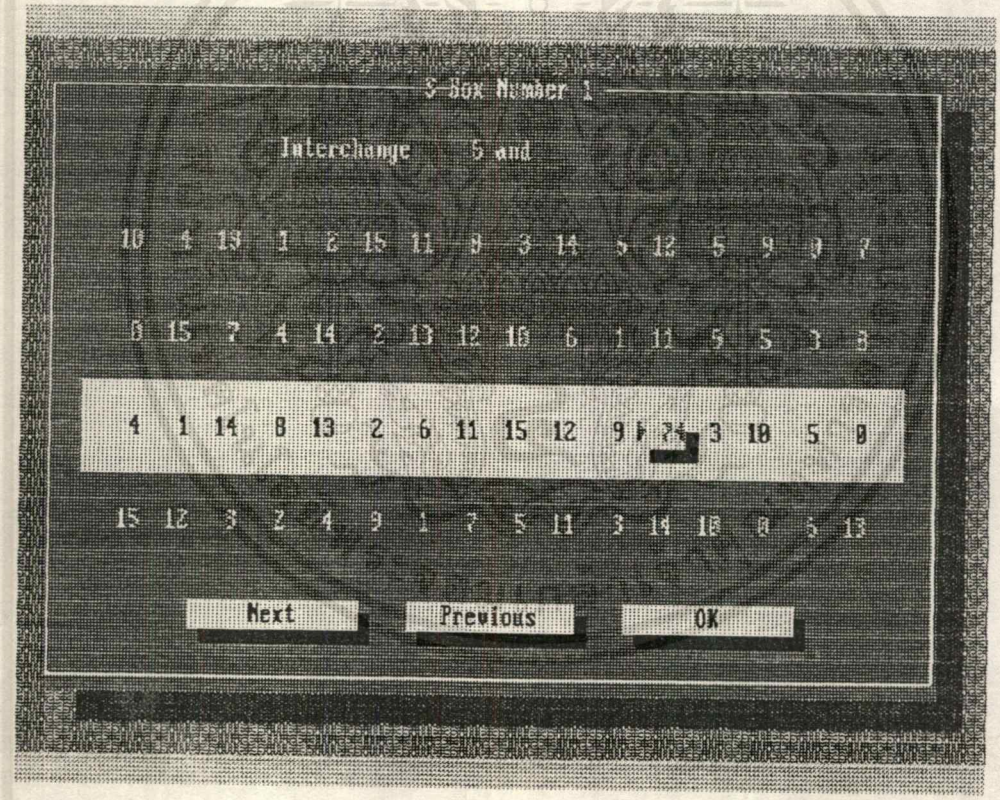


รูปที่ 5.24 แสดงรูปแบบและค่าภายใน S-box ต่างๆ

รูปที่ 5.24 เป็นรูปของ แต่ละ S-box จะมีตั้งแต่ 1-8 สามารถเลือกได้รดยใช้ Next (ถัดไป) หรือ Previous (ก่อนหน้า)

จะเห็นว่า S-box แต่ละชุด จะมี 4 แถว แต่ละแถวจะมี 16 ตัวเลขตั้งแต่ 0-15 การเปลี่ยนแปลง ก็คือการย้ายสลับที่ ตัวเลขทั้ง 16 ตัวภายในแถวเดียวกัน ไปมา จะย้ายข้ามแถวไม่ได้

จากรูปที่ 5.24 จะเห็นว่าจะมีแถบสว่างปรากฏอยู่ในแถวใดแถวหนึ่ง และ ในแถวนั้นก็จะมีตัวเลขที่อยู่ในกรอบเล็กๆ เราสามารถเลื่อนแถวได้ด้วยลูกศรขึ้นลง และ เลื่อนตัวเลขในแถวด้วยลูกศรซ้ายขวา



รูปที่ 5.25 การสับเปลี่ยนค่าภายใน S-box

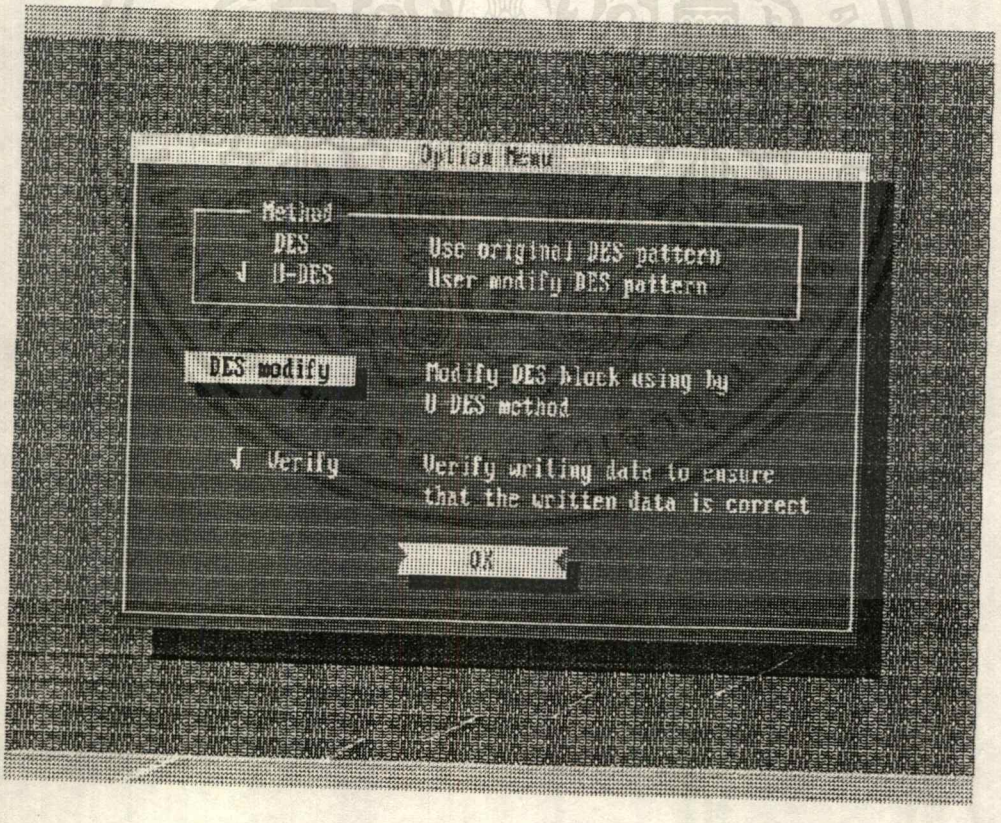
ถ้าต้องการตำแหน่งของเลขตัวใดก็ให้กด ENTER ลงไปจะขึ้นตัวเลขนั้นๆไว้ที่ด้านบนแล้วก็เลื่อนแถบนั้นไปยังที่ๆต้องการจะเปลี่ยนแล้วกด ENTER อีกครั้งตัวเลขทั้งสองก็จะสลับที่กันรดยอัตรนมัติ

เราสามารถทำเช่นนี้ได้กับทุก Boxes และทุกแถวจะทำกี่ครั้งก็ได้จนกว่าจะเป็นที่พอใจของเราเอง เมื่อเราพอใจกับการสับเปลี่ยนแล้ว เราก็มาเลือกที่ OK ก็จะกลับไปที่ Option Menu เมื่อเรียบร้อย เลือกที่ OK อีกครั้งก็จะกลับไปที่ เมนเมนู ทำการ ENCRYPTION ต่อไปได้

Verify

การใช้ฟังก์ชั่นนี้เพื่อเป็นการตรวจสอบข้อมูลเพิ่มขึ้นอีกชั้นหนึ่ง โดยตรวจสอบข้อมูลที่เขียนลงไปในดิสก์ว่าถูกต้องหรือไม่

การเลือกใช้ฟังก์ชั่นนี้สามารถทำได้โดย การเลื่อนแถบสว่างมาที่ตำแหน่งนั้น แล้ว ENTER ก็จะมีเครื่องหมาย √ ขึ้นมาค้างไว้ที่ด้านหน้าของคำสั่ง นั้นคือต่อไปการทำ ENCRYPTION หรือ DECRYPTION ก็จะมีการ Verify ทุกครั้งไป



รูปที่ 5.26 แสดงการเลือก Verify

การใช้ฟังก์ชั่นนี้อาจทำให้ การทำงานของโปรแกรมช้าไปบางเนื่องจากมีการตรวจสอบค่ากับ memory ก่อน

บทที่ 6 การทดลองและผลการทดลอง

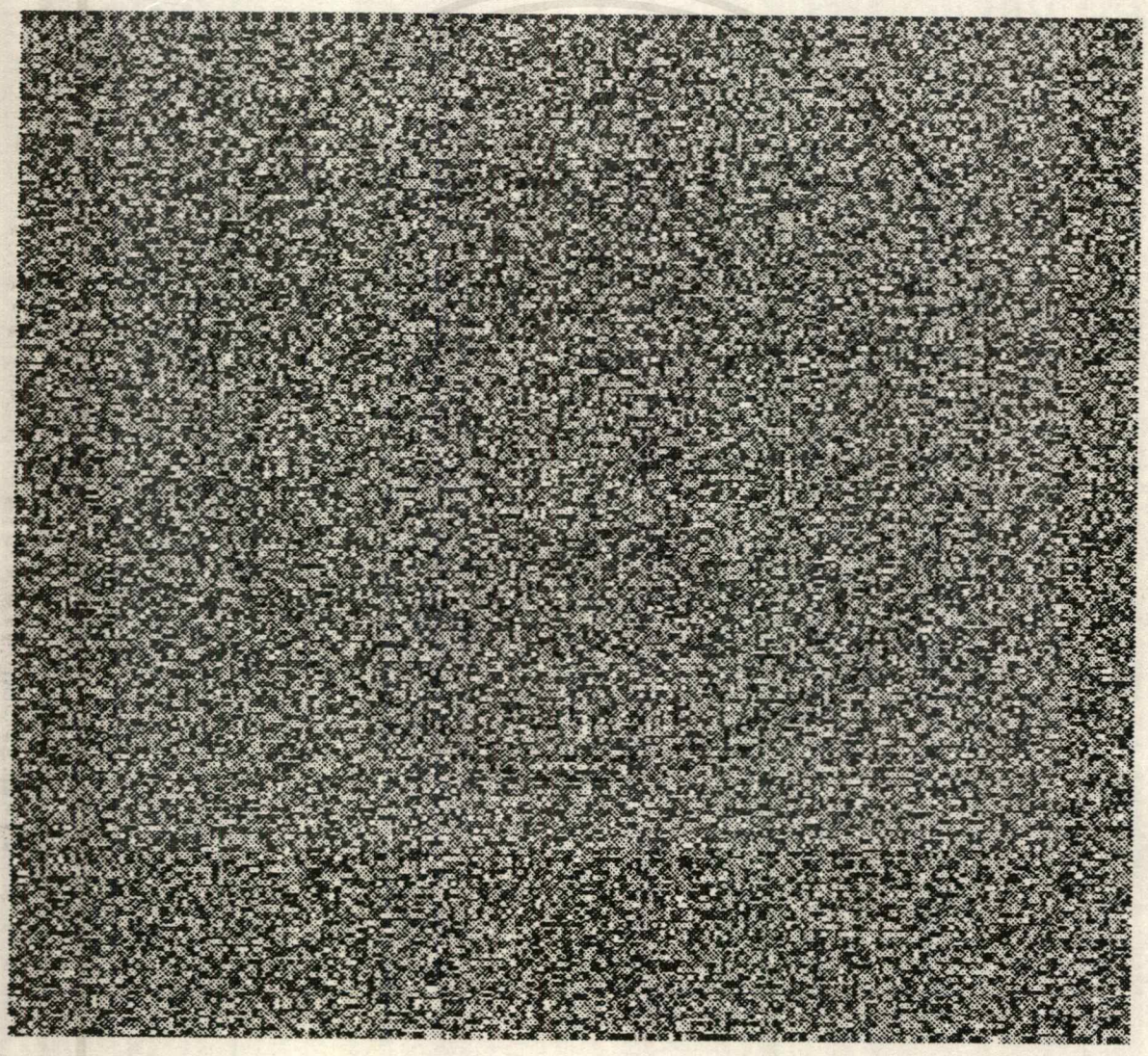
เราได้ทำการทดลองให้โปรแกรม CRYPT ทำงานโดยเลือกใช้ไฟล์ภาพแสดงผลของโปรแกรม เพื่อให้เห็นผลการทำงานได้ชัดเจน การแสดงผลของภาพอาศัยโปรแกรม VGADISPLAY เป็นโปรแกรมแสดงผล แล้วใช้การ Print Screen ออกมาให้ดูเป็น รูปที่ 6.1

รูปที่ 6.1 รูปที่เกิดจากไฟล์ teeny.dta แสดงโดย Vgadisplay

เราตั้งชื่อไฟล์นี้ว่า teeny.dta จะเห็นได้ว่ารูปที่ออกมาก่อนการ encrypt มีสักษณะสมบูรณ์ทุกประการ

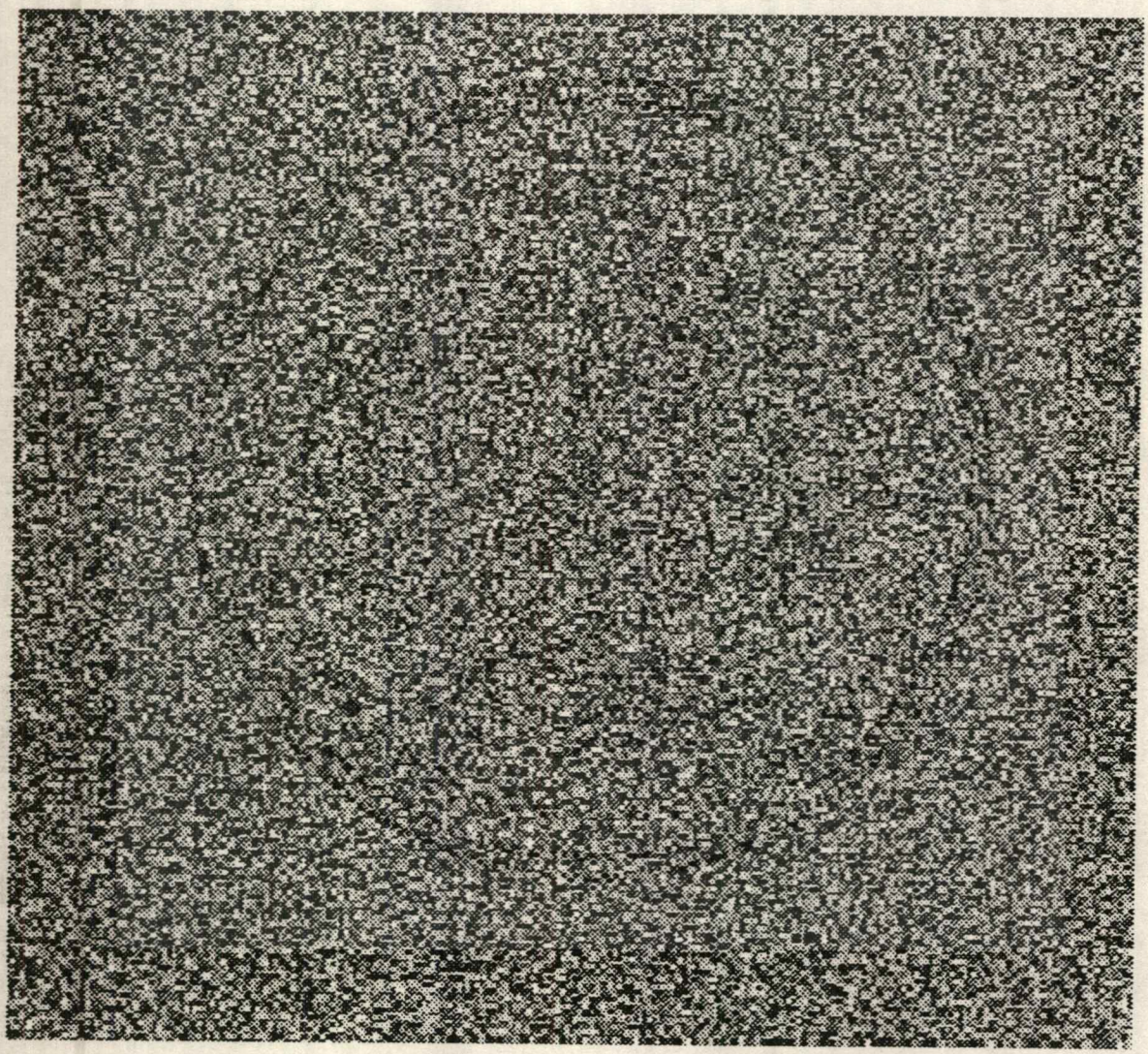
หลังจากนั้นเราใช้ โปรแกรม crypt เมื่อทำการ encrypt เราใช้การทดลอง encrypt 2 ครั้ง เพื่อให้มี password ที่ต่างกันออกไป

ครั้งที่หนึ่งใช้ชื่อไฟล์ที่ถูก encrypt แล้วว่า teeny.en1 ใช้ password ในการ encrypt ว่า sarunanun ได้ภาพหลังการ encrypt แสดงโดย VGADISPLAY ดัง รูปที่ 6.2



รูปที่ 6.2 รูปที่เกิดจากไฟล์ teeny.en1 เมื่อผ่านการ encrypt ด้วย Password sarunanun

และครั้งที่สอง ใช้ชื่อไฟล์ที่ถูก encrypt ว่า teeny.en2 ใช้ Password ว่า pornvitt รูปที่ได้หลังจากการ encrypt แล้ว เป็นดัง รูปที่ 6.3



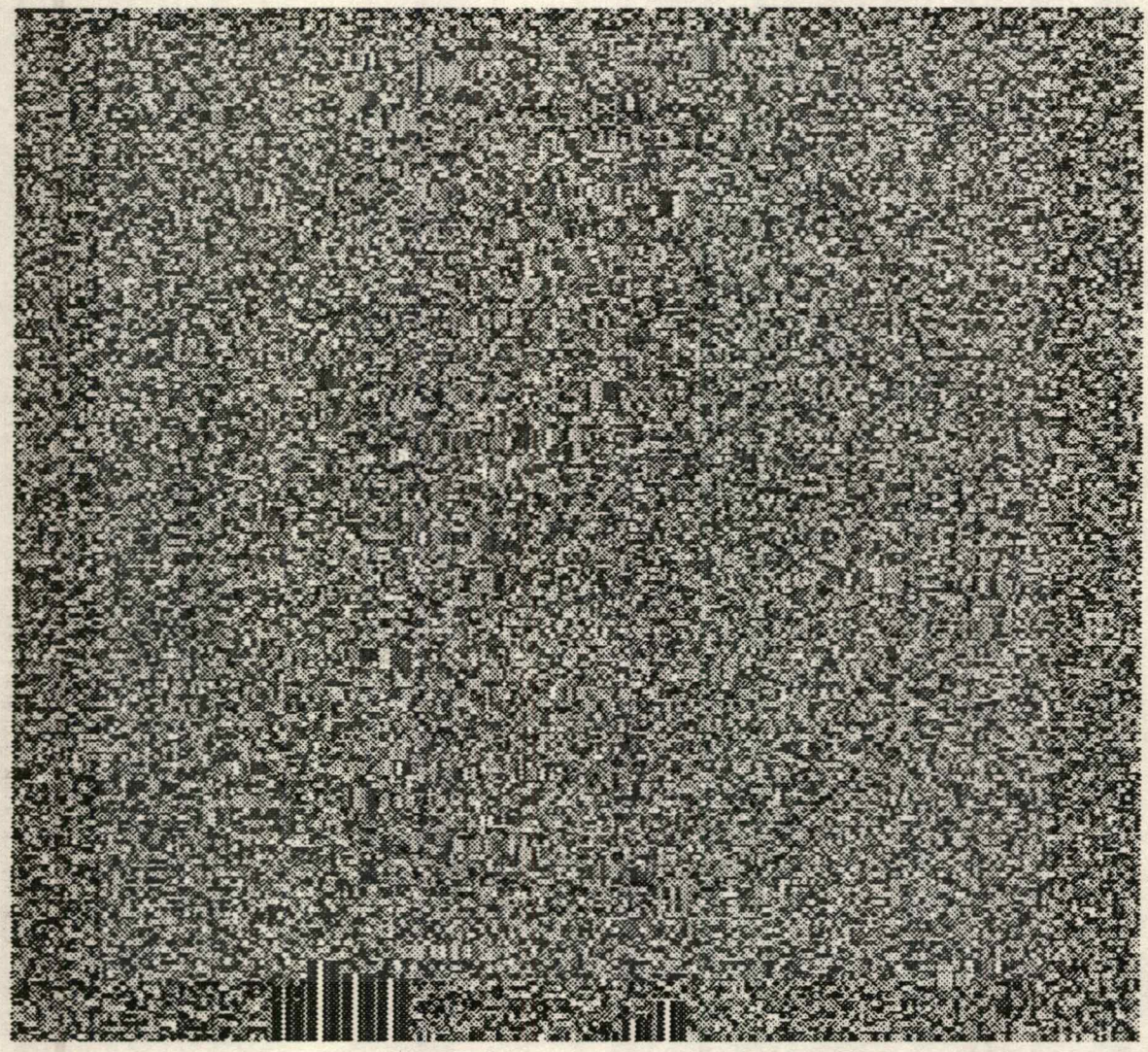
รูปที่ 6.3 รูปที่เกิดจากไฟล์ teeny.en2 เมื่อผ่านการ encrypt ด้วย Password pornvitt

จะเห็นได้ว่ารูปทั้งสองที่ผ่านการ encrypt นั้น จะแตกต่างไปจากต้นฉบับเดิมอย่างมากจนหาเค้าโครงเดิมไม่ได้เลย และรูปทั้งสองที่ผ่านการ encrypt ด้วย password ที่ต่างกันนั้น ก็มีความแตกต่างกันอยู่แต่อาจสังเกตุเห็นด้วยตาเปล่าลำบากสักหน่อย

เราลองทำการเปรียบเทียบจากไฟล์ที่มีข้อมูลต่างจากรูปที่เกิดจากไฟล์ Teeny.dta เราใช้ไฟล์ที่มีชื่อว่า ama.dta ซึ่งมีการแสดงภาพดัง รูปที่ 6.4

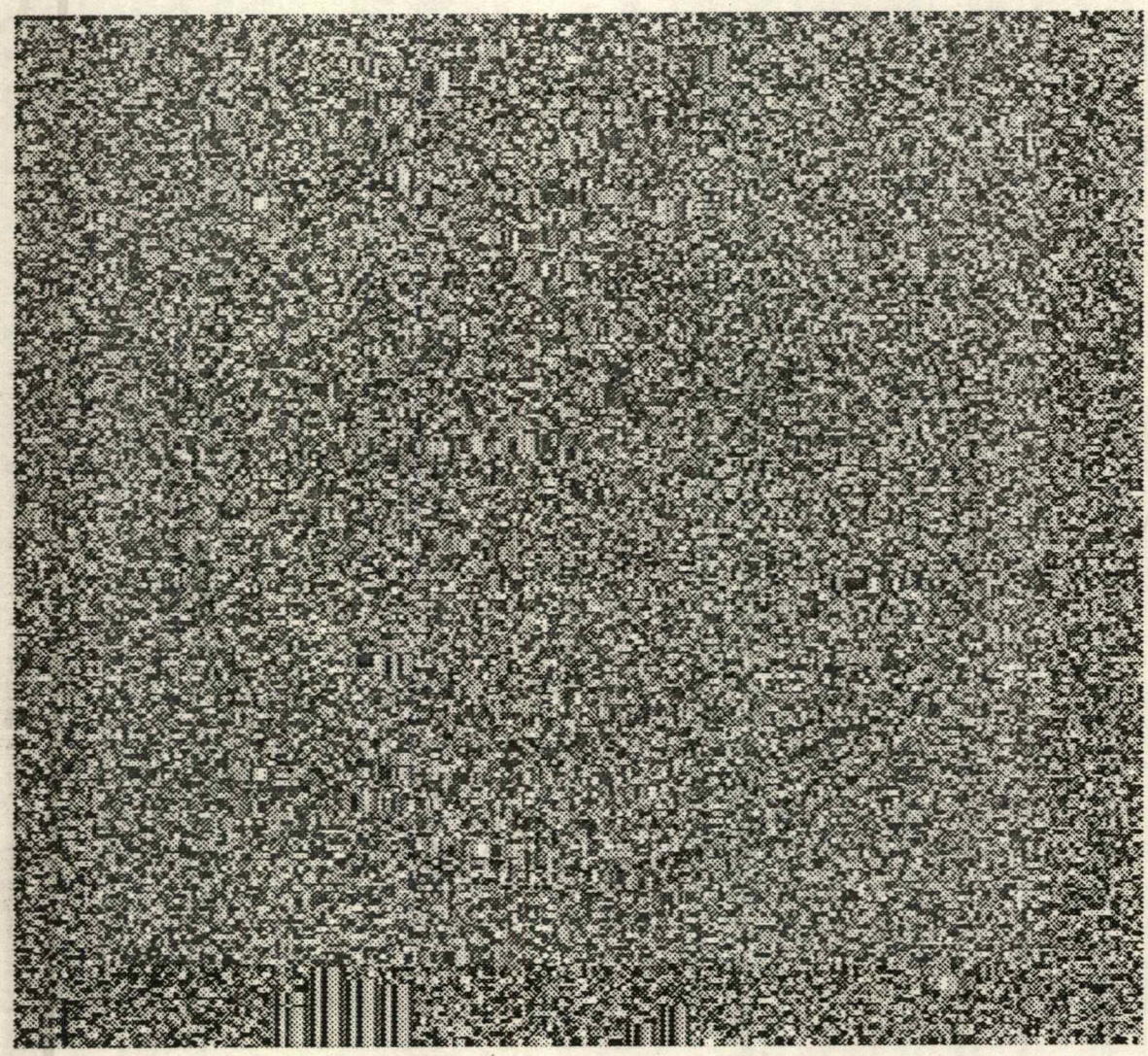
รูปที่ 6.4 รูปที่เกิดจากไฟล์ ama.dta แสดงโดย Vgadisplay

รูปจากไฟล์ ama.dta หลังจากผ่านการ encrypt ด้วย password sarunanun จะได้ภาพแสดงผลดังรูปที่ 6.5



รูปที่ 6.5 รูปที่เกิดจากไฟล์ ama.en1 เมื่อผ่านการ encrypt ด้วย Password sarunanun

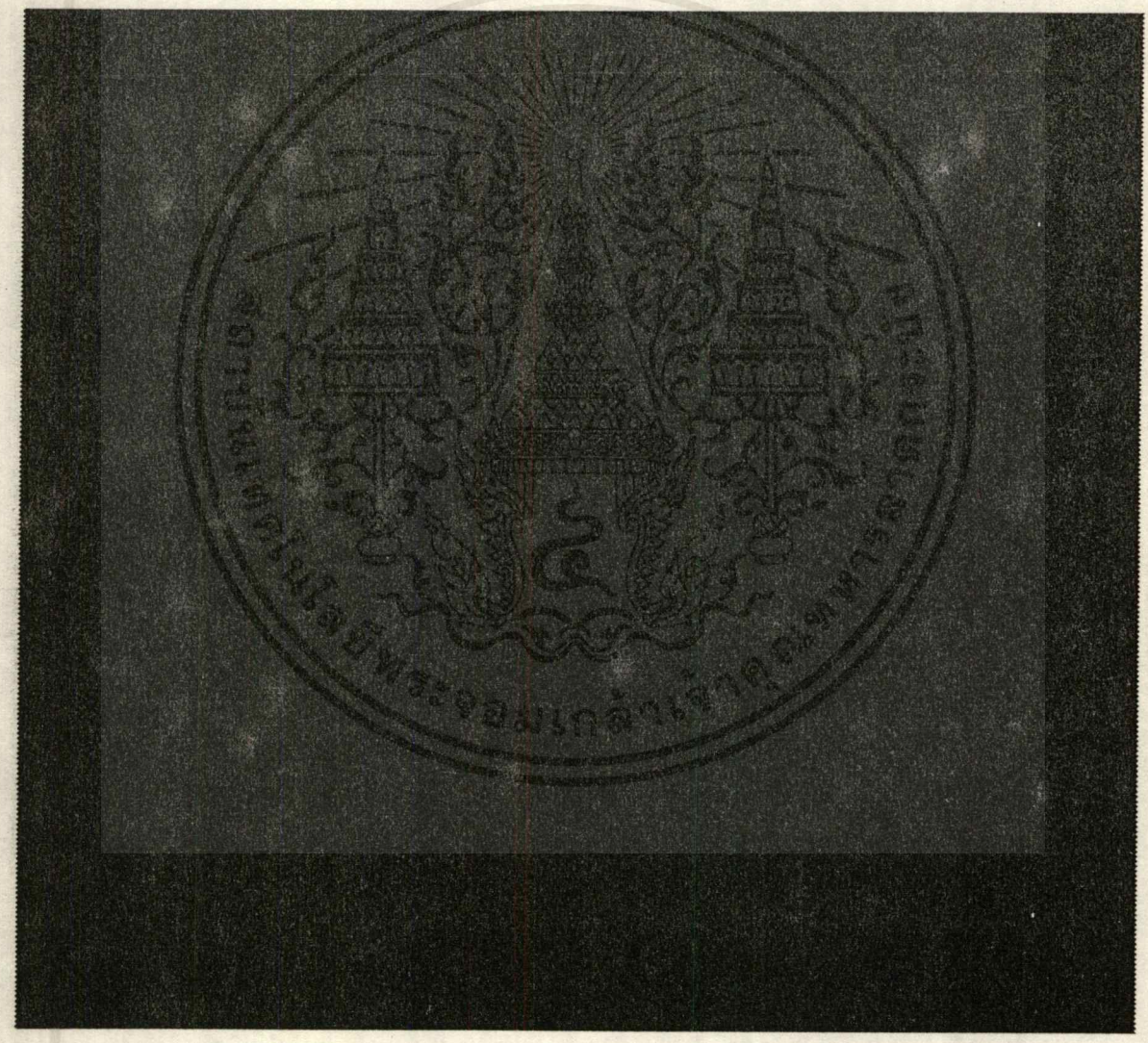
รูปจากไฟล์ ama.dta หลังจากผ่านการ encrypt ด้วย password robotics จะได้ภาพแสดงผลดัง รูปที่ 6.6



รูปที่ 6.6 รูปที่เกิดจากไฟล์ ama.en2 เมื่อผ่านการ encrypt ด้วย Password robotics

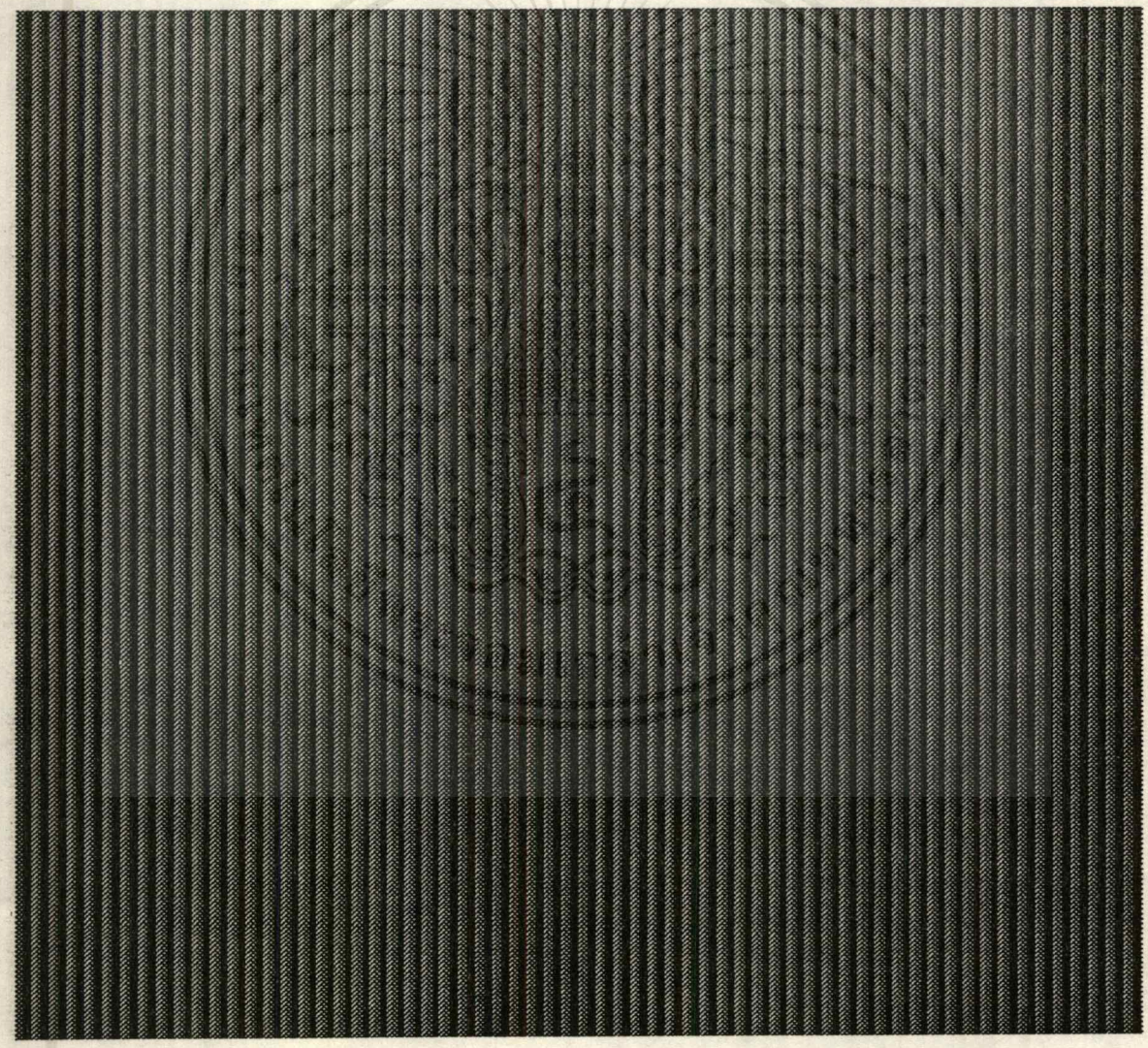
จะเห็นได้ว่าทั้งสองรูปที่ผ่านการ encrypt มีความแตกต่างจากต้นฉบับเดิมอย่างเห็นได้ชัด และทั้งสองไฟล์นั้นก็มีความแตกต่างกันอยู่ แต่อาจสังเกตได้ด้วยตาเปล่าบ้างสักหน่อย

เราจึงได้คิดวิธีการเปรียบเทียบความแตกต่างของรูปที่ผ่านการ encrypt แล้วเพื่อแสดงให้เห็นอย่างชัดเจนโดยใช้การนำข้อมูลของแต่ละไฟล์มาทำการลบค่ากัน ซึ่งถ้าไฟล์ทั้งสองมีข้อมูลเหมือนกันก็จะทำให้ข้อมูลที่เกิดจากการลบกันนั้นเป็นศูนย์ทั้งหมดถ้าข้อมูลเป็นศูนย์ทั้งหมดจะแสดงออกมาเป็นภาพสีดำทั้งหมดดัง รูปที่ 6.7



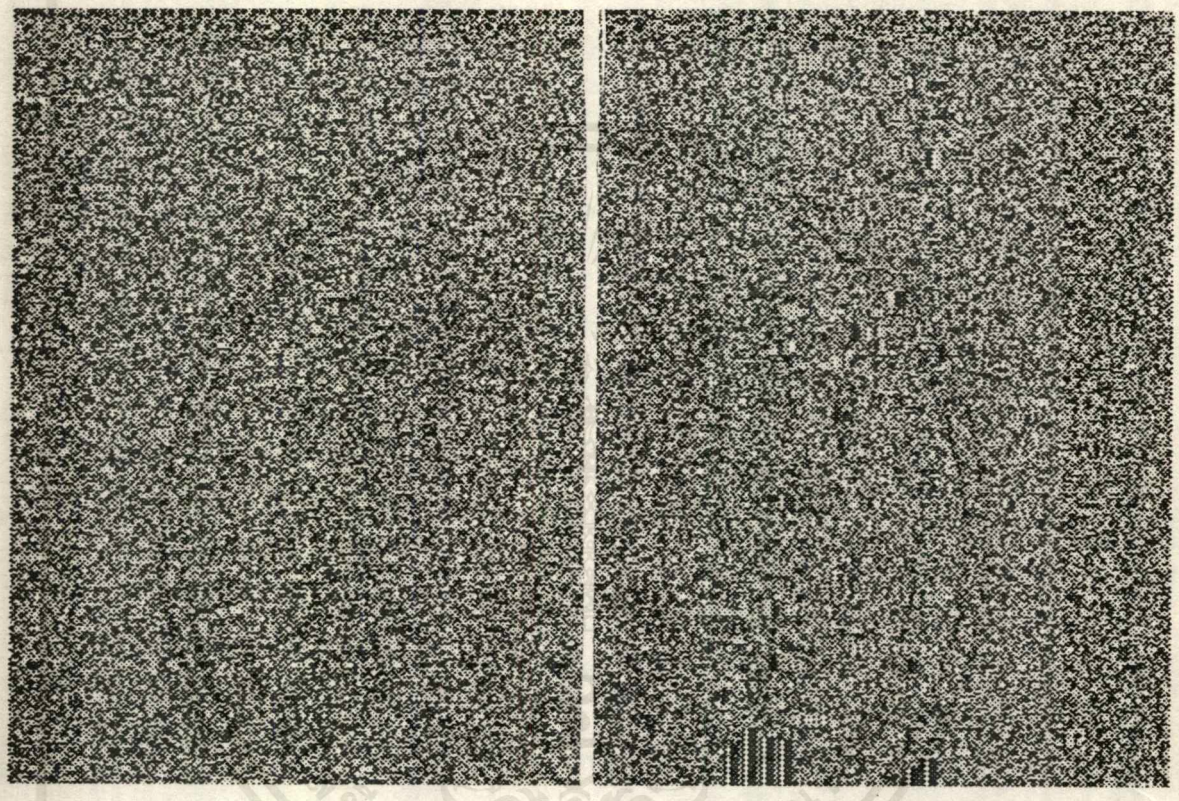
รูปที่ 6.7 ภาพที่เกิดจากการลบไฟล์ที่มีข้อมูลเหมือนกัน

และเมื่อทำการเปรียบเทียบไฟล์ที่ผ่านการ encrypt จากต้นฉบับเดียวกันแต่ใช้ password ที่ต่างกัน ข้อมูลที่ได้จากการเปรียบเทียบจะแสดงความแตกต่างออกมาเป็นคาบ ๆ อันเนื่องมาจากวิธีเข้ารหัสแบบ DES ซึ่งทำการเข้ารหัสทีละ 8 ไบท์ ทำให้ข้อมูลที่ได้จากการ encryt ไฟล์เดียวกันด้วย password ที่ต่างกัน มีความแตกต่างระหว่างข้อมูลเป็นช่วง ๆ แต่ละช่วงจะมีค่าความแตกต่างที่เท่ากัน แสดงผลได้ดังรูป 6.8



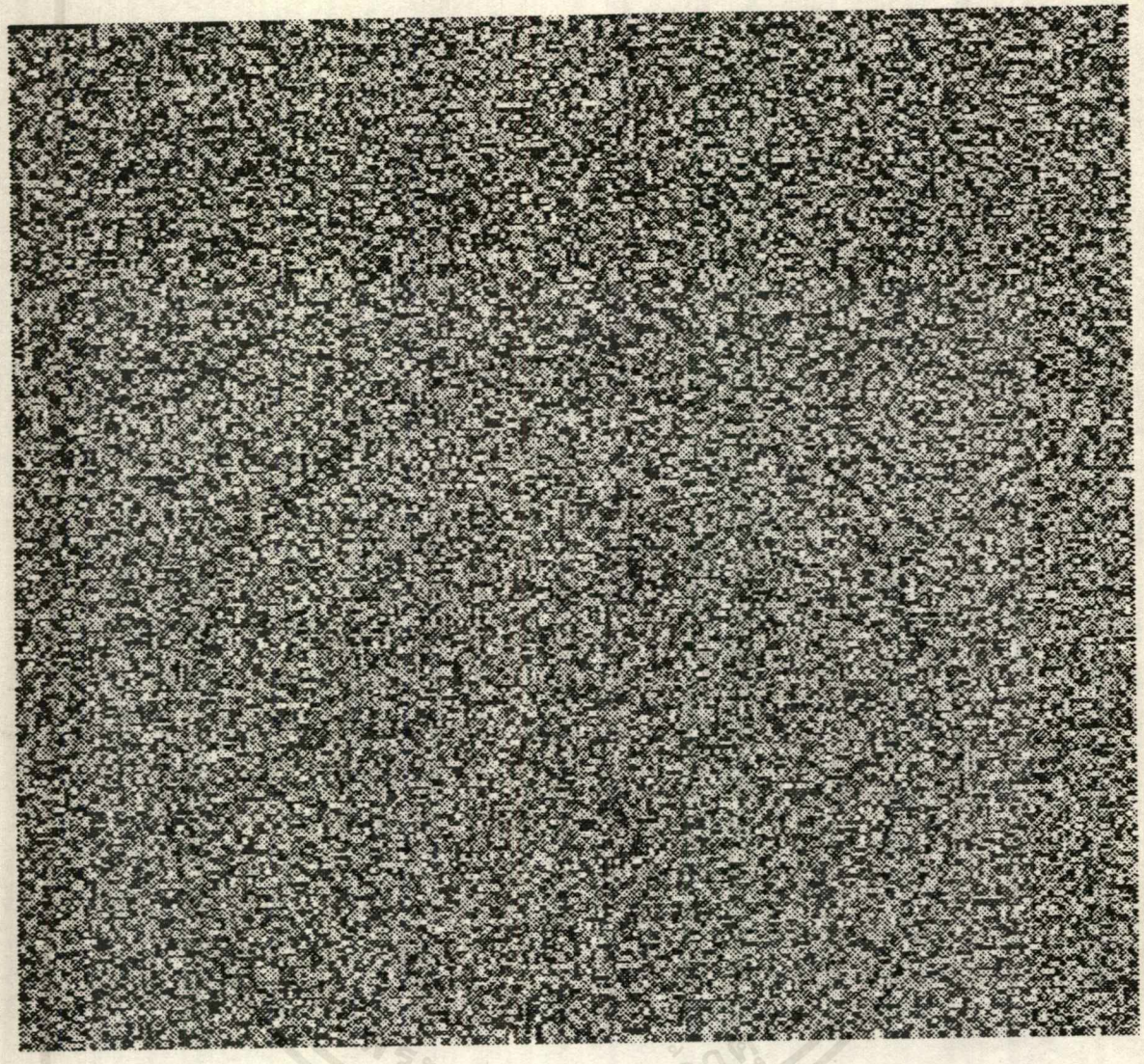
รูปที่ 6.8 รูปที่เกิดจากการลบไฟล์ที่มีต้นฉบับเดียวกันแต่ใช้ Password ในการเข้ารหัสที่ต่างกัน

จากการ encrypt ไฟล์ที่ต่างกัน แต่ทำการ encrypt ด้วย password เดียวกัน เราอาจสังเกตุเห็นความแตกต่างได้ด้วยตาเปล่า ดังรูปตัวอย่างที่เกิดจากไฟล์ teeny.en1 ซึ่งใช้ password sarunanun และ ไฟล์ ama.en1 ซึ่งใช้ password เดียวกัน ดัง รูปที่ 6.9



รูปที่ 6.9 รูปจากไฟล์ teeny.en1 รูปจากไฟล์ ama.en1

แล้วเราก็ทำการเปรียบเทียบข้อมูลด้วยการนำค่าของมูล ของแต่ละไฟล์มาลบกัน ผลที่ได้จะออกมาเป็น ค่าที่ต่างกันดังรูป และค่าที่ได้นั้นจะไม่ต่างกันเป็นช่วง ๆ เหมือนกับผลที่ได้จากการลบไฟล์ที่ผ่านการ encrypt จากต้นฉบับเดียวกันแต่ password ต่างกัน ดังรูปที่ 6.10



รูปที่ 6.10 รูปที่เกิดจากการลบไฟล์ที่ต่างกันแต่มี Password ในการเข้ารหัสเดียวกัน

บทที่ 7 สรุปผลการดำเนินงาน

จากโครงงานที่ได้ทำมาตลอดปีการศึกษา 2535 นี้ เราได้พัฒนาการทำงานโปรแกรม Data Encryption Program ให้สามารถทำงานได้ในระดับที่พร้อมสมบูรณ์ รวมทั้งรูปแบบของการใช้งานผ่านเมนูที่ง่ายต่อการใช้และการเข้าใจ เราได้แก้ไขข้อผิดพลาดที่เกิดขึ้นในช่วงแรกๆได้ นอกจากนี้ ความเร็วของ การเข้ารหัส เราสามารถทำให้เพิ่มขึ้นจากเดิม

โปรแกรมนี้สามารถใช้งานได้จริง โดยไม่มีการจำกัด รูปแบบของไฟล์ (นามสกุลของไฟล์)และขนาดของไฟล์แต่จำกัดที่ เนื้อที่ของดิสก์ที่จะนำไฟล์ที่ผ่าน การเข้ารหัสหรือถอดรหัสแล้วไปเก็บไว้จะต้องมีเนื้อที่เพียงพอ มิฉะนั้นโปรแกรมจะไม่ทำงาน ทุกไฟล์ที่ผ่านการเข้ารหัส แล้วจะไม่เหลือเค้าโครงเดิมอยู่เลย การถอดรหัสก็จะต้องใช้ Password ที่ถูกต้องเท่านั้น โปรแกรมจึงจะทำงานได้

ในปัจจุบันนี้ความลับของข้อมูลเป็นสิ่งที่สำคัญและมีค่ามาก ความจำเป็นที่จะต้องใช้โปรแกรมสามารถทำข้อมูลต่างๆ นั้นเป็นความลับ จึงเป็นที่ต้อง การโปรแกรม crypt นี้ทั้งในด้านความปลอดภัยและการใช้งานอยู่ในเกณฑ์ที่ดี ดังนั้นโปรแกรมการเข้ารหัสนี้จึงน่าจะเป็นที่นิยมใช้ต่อไป

กิตติกรรมประกาศ

โครงงานในปีการศึกษานี้สำเร็จลุล่วงลงไปได้ด้วยดี ทั้งนี้ต้องขอขอบพระคุณ รศ.ดร. พูศักดิ์ ชีวสุวิทย์ ที่ได้คอยชี้แนะแนวทางในการทำงาน คำแนะนำในการแก้ปัญหาต่างๆที่เกิดขึ้นให้เป็นไปได้ด้วยดี ขอขอบคุณ อาจารย์ ธนิตย์ ตรีสุวรรณวัฒน์ ที่ได้ให้ใช้ห้อง และ เครื่องคอมพิวเตอร์ พรินเตอร์ในการทำงาน ขอบคุณเพื่อน ๆ ทุก ๆ คนที่คอยให้กำลังใจ และ คำปรึกษาต่างๆในการทำงาน ขอบคุณเวลาที่มีให้พอที่จะทำให้โครงการนี้สำเร็จได้

หนังสืออ้างอิง

1. JENNIFER SEBERRY. JOSEF PIEPRZYK, "CRYPTOGRAPHY : An Introduction to Computer Security",Prentice Hall International Inc.

2. Don J. Torrieri, "Principle of Secure Communication Systems", ARTECH HOUSE Inc. ,1985

ภาคผนวก

Program Listing

Appendix A

File : DESA.C

Function : main encryption and decryption function.

```
#include <stdlib.h>
#include <stdio.h>

#include "B:\text\bitdef.c"


Long  F( Long R, char *Key );


Char DES_IP[64] =
      {   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 };

Char IP[64] = {   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 };

Char DES_FP[64] =
```

```
    {   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 };

Char FP[64] = {   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 };

Char E[48]  = {  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 };

Char DES_P[32]  = {   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 };

Char P[32]  = {   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 };


Char DES_S[8][64] = {
                  /* S1 == S[0] */
      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,

   /* S2 */
      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,
```

```
/* S3 */
    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,

/* S4 */
     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,

/* S5 */
     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,

/* S6 */
    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,

/* S7 */
     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,

/* S8 */
   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  }


Char S[8][64] = {
                /* S1 == S[0] */
   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,

/* S2 */
   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,

/* S3 */
   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,
```

```
/* S4 */
    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,

/* S5 */
    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,

/* S6 */
   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,

/* S7 */
    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,

/* S8 */
   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


Long S_V2[8][64] = {
   /* S1 == S[0] */
/* S1 row 0 */
      0xE0000000, 0x40000000, 0xD0000000, 0x10000000,
      0x20000000, 0xF0000000, 0xB0000000, 0x80000000,
      0x30000000, 0xA0000000, 0x60000000, 0xC0000000,
      0x50000000, 0x90000000, 0x0,        0x70000000,
/* S1 row 1 */
      0x0,        0xF0000000, 0x70000000, 0x40000000,
      0xE0000000, 0x20000000, 0xD0000000, 0x10000000,
      0xA0000000, 0x60000000, 0xC0000000, 0xB0000000,
      0x90000000, 0x50000000, 0x30000000, 0x80000000,
/* S1 row 2 */
      0x40000000, 0x10000000, 0xE0000000, 0x80000000,
      0xD0000000, 0x60000000, 0x20000000, 0xB0000000,
      0xF0000000, 0xC0000000, 0x90000000, 0x70000000,
      0x30000000, 0xA0000000, 0x50000000, 0x0,
/* S1 row 3 */
      0xF0000000, 0xC0000000, 0x80000000, 0x20000000,
      0x40000000, 0x90000000, 0x10000000, 0x70000000,
      0x50000000, 0xB0000000, 0x30000000, 0xE0000000,
      0xA0000000, 0x0,        0x60000000, 0xD0000000,

   /* S2 */
/* S2 row 0 */
      0xF000000, 0x1000000, 0x8000000, 0xE000000, 0x6000000,
```

```
    0xB000000, 0x3000000, 0x4000000, 0x9000000, 0x7000000,
    0x2000000, 0xD000000, 0xC000000, 0x0000000, 0x5000000, 0xA000000
/* S2 row 1 */
    0x3000000, 0xD000000, 0x4000000, 0x7000000, 0xF000000,
    0x2000000, 0x8000000, 0xE000000, 0xC000000, 0x0,
    0x1000000, 0xA000000, 0x6000000, 0x9000000, 0xB000000, 0x5000000
/* S2 row 2 */
    0x0000000, 0xE000000, 0x7000000, 0xB000000, 0xA000000,
    0x4000000, 0xD000000, 0x1000000, 0x5000000, 0x8000000,
    0xC000000, 0x6000000, 0x9000000, 0x3000000, 0x2000000, 0xF000000
/* S2 row 3 */
    0xD000000, 0x8000000, 0xA000000, 0x1000000, 0x3000000,
    0xF000000, 0x4000000, 0x2000000, 0xB000000, 0x6000000,
    0x7000000, 0xC000000, 0x0,       0x5000000, 0xE000000, 0x9000000

  /* S3 */
/* S3 row 0 */
    0xA00000, 0x0,     0x900000, 0xE00000, 0x600000,
    0x300000, 0xF00000, 0x500000, 0x100000, 0xD00000,
    0xC00000, 0x700000, 0xB00000, 0x400000, 0x200000, 0x800000,
/* S3 row 1 */
    0xD00000, 0x700000, 0x0,     0x900000, 0x300000,
    0x400000, 0x600000, 0xA00000, 0x200000, 0x800000,
    0x500000, 0xE00000, 0xC00000, 0xB00000, 0xF00000, 0x100000,
/* S3 row 2 */
    0xD00000, 0x600000, 0x400000, 0x900000, 0x800000,
    0xF00000, 0x300000, 0x0,     0xB00000, 0x100000,
    0x200000, 0xC00000, 0x500000, 0xA00000, 0xE00000, 0x700000,
/* S3 row 3 */
```

```
        0x100000, 0xA00000, 0xD00000, 0x0,      0x600000,
        0x900000, 0x800000, 0x700000, 0x400000, 0xF00000,
        0xE00000, 0x300000, 0xB00000, 0x500000, 0x200000, 0xC00000,

    /* S4 */
 /* S4 row 0 */
        0x70000, 0xD0000, 0xE0000, 0x30000, 0x0,
        0x60000, 0x90000, 0xA0000, 0x10000, 0x20000,
        0x80000, 0x50000, 0xB0000, 0xC0000, 0x40000, 0xF0000,
 /* S4 row 1 */
        0xD0000, 0x80000, 0xB0000, 0x50000, 0x60000,
        0xF0000, 0x0,     0x30000, 0x40000, 0x70000,
        0x20000, 0xC0000, 0x10000, 0xA0000, 0xE0000, 0x90000,
 /* S4 row 2 */
        0xA0000, 0x60000, 0x90000, 0x0,     0xC0000,
        0xB0000, 0x70000, 0xD0000, 0xF0000, 0x10000,
        0x30000, 0xE0000, 0x50000, 0x20000, 0x80000, 0x40000,
 /* S4 row 3 */
        0x30000, 0xF0000, 0x0,     0x60000, 0xA0000,
        0x10000, 0xD0000, 0x80000, 0x90000, 0x40000,
        0x50000, 0xB0000, 0xC0000, 0x70000, 0x20000, 0xE0000,

    /* S5 */
 /* S5 row 0 */
        0x2000, 0xC000, 0x4000, 0x1000, 0x7000, 0xA000, 0xB000, 0x6000,
        0x8000, 0x5000, 0x3000, 0xF000, 0xD000, 0x0,    0xE000, 0x9000,
 /* S5 row 1 */
        0xE000, 0xB000, 0x2000, 0xC000, 0x4000, 0x7000, 0xD000, 0x1000,
        0x5000, 0x0,    0xF000, 0xA000, 0x3000, 0x9000, 0x8000, 0x6000,
```

```
/* S5 row 2 */
    0x4000, 0x2000, 0x1000, 0xB000, 0xA000, 0xD000, 0x7000, 0x8000,
    0xF000, 0x9000, 0xC000, 0x5000, 0x6000, 0x3000, 0x0,    0xE000,
/* S5 row 3 */
    0xB000, 0x8000, 0xC000, 0x7000, 0x1000, 0xE000, 0x2000, 0xD000,
    0x6000, 0xF000, 0x0,    0x9000, 0xA000, 0x4000, 0x5000, 0x3000,


  /* S6 */
/* S6 row 0 */
    0xC00, 0x100, 0xA00, 0xF00, 0x900, 0x200, 0x600, 0x800,
    0x0,   0xD00, 0x300, 0x400, 0xE00, 0x700, 0x500, 0xB00,
/* S6 row 1 */
    0xA00, 0xF00, 0x400, 0x200, 0x700, 0xC00, 0x900, 0x500,
    0x600, 0x100, 0xD00, 0xE00, 0x0,   0xB00, 0x300, 0x800,
/* S6 row 2 */
    0x900, 0xE00, 0xF00, 0x500, 0x200, 0x800, 0xC00, 0x300,
    0x700, 0x0,   0x400, 0xA00, 0x100, 0xD00, 0xB00, 0x600,
/* S6 row 3 */
    0x400, 0x300, 0x200, 0xC00, 0x900, 0x500, 0xF00, 0xA00,
    0xB00, 0xE00, 0x100, 0x700, 0x600, 0x0,   0x800, 0xD00,


  /* S7 */
/* S7 row 0 */
    0x40, 0xB0, 0x20, 0xE0, 0xF0, 0x0,  0x80, 0xD0,
    0x30, 0xC0, 0x90, 0x70, 0x50, 0xA0, 0x60, 0x10,
/* S7 row 1 */
    0xD0, 0x0,  0xB0, 0x70, 0x40, 0x90, 0x10, 0xA0,
    0xE0, 0x30, 0x50, 0xC0, 0x20, 0xF0, 0x80, 0x60,
```

```
/* S7 row 2 */
      0x10, 0x40, 0xB0, 0xD0, 0xC0, 0x30, 0x70, 0xE0,
      0xA0, 0xF0, 0x60, 0x80, 0x0,  0x50, 0x90, 0x20,
/* S7 row 3 */
      0x60, 0xB0, 0xD0, 0x80, 0x10, 0x40, 0xA0, 0x70,
      0x90, 0x50, 0x0,  0xF0, 0xE0, 0x20, 0x30, 0xC0,

   /* S8 */
/* S8 row 0 */
      0xD, 0x2, 0x8, 0x4, 0x6, 0xF, 0xB, 0x1,
      0xA, 0x9, 0x3, 0xE, 0x5, 0x0, 0xC, 0x7,
/* S8 row 1 */
      0x1, 0xF, 0xD, 0x8, 0xA, 0x3, 0x7, 0x4,
      0xC, 0x5, 0x6, 0xB, 0x0, 0xE, 0x9, 0x2,
/* S8 row 2 */
      0x7, 0xB, 0x4, 0x1, 0x9, 0xC, 0xE, 0x2,
      0x0, 0x6, 0xA, 0xD, 0xF, 0x3, 0x5, 0x8,
/* S8 row 3 */
      0x2, 0x1, 0xE, 0x7, 0x4, 0xA, 0x8, 0xD,
      0xF, 0xC, 0x9, 0x0, 0x3, 0x5, 0x6, 0xB  };



Char PC1[64] = {  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,
  32, 32, 32. 32,                 /* <- not used this 4 bits */
  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,
  64, 64, 64, 64              }; /* <- not used this 4 bits */

Char Keyrol[16] = { 1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1 };

Char PC2[48] = {  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 };

static Char  SubKey[16][8];

/* Alpha note this function not require return value /
   since the style of programming using long value in main function
   encrypt for output already */

/* Function : Permu64                                          */
/*     Duty : permutation a 64 bits block using defined pattern */
/*    Input : long input[2]                                    */
/*   Output : long output[2]                                   */
/* Return Value :  None (after final test)                     */

Permu64(Long input[2],Long output[2],Char *Pattern)
```

```
{
register int i;

    for ( i=0; i<2; i++ ) {
  output[i] ^= output[i];          /* Clear Output Words to 0L */
  }                                /*  by xor itself           */

    for ( i=0; i<=31; i++) {        -  /* Input permu bit 1-32 */
         output[0] <<= 1;
  if ( Bit( input, (int) Pattern[i] ) == 1 )  {
       output[0] |= 1; }
  }

    for ( i=32; i<=63; i++) {          /* Input permu bit 33-64 */
         output[1] <<= 1;
  if ( Bit( input, (int) Pattern[i] ) == 1 )  {
       output[1] |= 1; }
  }

}

/* Function: Bit  find the value of input bit 0 or 1 and return int */
/*    Input: Long 2 words  = [1,2,...,32][33,34,...,62]              */
/*                                 In[0]        In[1]                */
/*         : And position of bit to find value 1...62 (Chart)       */

int Bit(Long InBit[2], Int Chart)
{
int   Side,Pos;
```

```
        Side = (Chart-1)   / 32;
        Pos  = (64-Chart)  % 32;

        return ( (InBit[Side] >> Pos) & 01 );
}



Long  F( Long R, char *Key )
{
Char ER[8], S_Val;
Long Out, S_Out=0L, Output=0L;
register int j,rc;

        Expand( R, ER );
        for (j=0; j<8; j++) {
   rc = ER[j] ^ Key[j];
   rc = ( rc & 0x20 ) | ( ( rc << 4 ) & 0x10 ) |
( ( rc >> 1 ) & 0x0F );

        S_Val   = S[j][rc];
        S_Out   = ( S_Out << 4 ) | S_Val;
        }

        Permu32( &S_Out, &Output, P );

/*      printf("Output = %lx",Output); */
```

```
    return ( Output );
}

/*        This function is used for expand Right side in          */
/*      function F from 32 -> 48 bits using defined pattern.      */
/*                                                                */
/*   Function : Expand ( R, Output )                              */
/*      Input : Long R  32 bits input from function F             */
/*     Output : char * Output  ( 8 char = 64 bits but use 48 bits) */
/*   Use bits :  R => [1,2,...,32] , Output => [x,x,1,2,3,4,5,6]  */
/*               xx in Output is unused in program                */
/*                                                                */
/*       Note : This function is not used array pattern but       */
/*              programmed in other style for the reason of       */
/*        more speed.                                             */

Expand( Long R, Char *Output )
{
#define CF2  0x3F       /* For clear first 2 bits => 00xxxxxx */

    Output[0]  =   ( ( R >> 27 ) | ( R << 5 ) )  &  CF2;
    Output[1]  =     ( R >> 23 )  &  CF2;
    Output[2]  =     ( R >> 19 )  &  CF2;
    Output[3]  =     ( R >> 15 )  &  CF2;
    Output[4]  =     ( R >> 11 )  &  CF2;
    Output[5]  =     ( R >>  7 )  &  CF2;
    Output[6]  =     ( R >>  3 )  &  CF2;
    Output[7]  =   ( ( R <<  1 ) | ( R >> 31 ) )  &  CF2;
}
```

```
Permu32( Long *Input, Long *Output, Char *Pattern )
{
register int i;

      *Output = 0x0L;
      for ( i=0; i<32; i++) {
   *Output <<= 1;
   *Output |= Bit( Input, (int) Pattern[i] );
   }
}

void S_Key( Long Key[2] )
{
#define MARK28 0xFFFFFFF0L;    /* use for clear last unused 4 bits */
Long CD[2];
register int i;

      Permu64( Key, CD, PC1 );
      CD[0] &= MARK28;
      CD[1] &= MARK28;

      for ( i=0; i<16; i++) {

   CD[0] = ( CD[0]  <<  (int) Keyrol[i] ) |
```

```
   ( CD[0]  >>  (28 -  (int) Keyrol[i]) )  & MARK28;

   CD[1] = ( CD[1]  <<  (int) Keyrol[i] ) |
   ( CD[1]  >>  (28 -  (int) Keyrol[i]) )  & MARK28;

   KeyPermu( CD, i, PC2 );
   }

}

KeyPermu( Long CD[2], int round, Char *Pattern)
{
register int j,k;

      for ( j=0; j<8; j++) {

    SubKey[round][j] = 0;       /* Clear Output to 0 */
    for ( k=0; k<6; k++) {
  SubKey[round][j] <<= 1;     /* Shift left 1 bits */
     /* for filling next bits in */
  SubKey[round][j] |= KeyBit ( CD, (int) *(Pattern++) );
  }
      }
}

int KeyBit( Long CD[2],int Pos )
{
int Side, Offset, Out;
```

```
Side  = (Pos-1) / 28;
      Offset  =  31 - ( (Pos-1) % 28 );
 Out  =  ( CD[Side] >> Offset ) & 01;
      return ( Out );
}


DES_Encrypt(Long Input[2])
{
Long LR[2],Temp;
register int i;

Permu64 ( Input, LR, IP );

for (i=0; i<16; i++) {              /* Do 16 Iteration of Encrypt */
     LR[0]  ^=  F( LR[1], SubKey[i] );
      Temp   =  LR[0];              /* Change side for next round */
     LR[0]   =  LR[1];              /*  Left <- Right             */
     LR[1]   =  Temp;               /*  Left -> Right             */
     }
Swap(LR);                           /* The final round  */
/* don't change side */

/*printf("Before Permu Input is %8lx %8lx\n",Input[0],Input[1]); */
Permu64(LR,Input,FP);
/*printf("Input is %8lx %8lx\n",Input[0],Input[1]); */
}

DES_Decrypt(Long Input[2])
```

```
{
Long LR[2],Temp;
register int i;

Permu64 ( Input, LR, IP );

for (i=15; i>=0; i--) {          /* Do 16 Iteration of Encrypt */
     LR[0]  ^=  F( LR[1], SubKey[i] );
      Temp   =  LR[0];           /* Change side for next round */
     LR[0]   =  LR[1];           /*  Left <- Right             */
     LR[1]   =  Temp;            /*  Left -> Right             */
     }
Swap(LR);                        /* The final round  */
/* don't change side */

/*printf("Before Permu Input is %8lx %8lx\n\n",Input[0],Input[1]);*/
Permu64(LR,Input,FP);
/*printf("Input is %8lx %8lx\n",Input[0],Input[1]); */
}



Swap( Long *LR )
{
Long Tmp;
     Tmp = LR[0];
     LR[0] = LR[1];
     LR[1] = Tmp;
}
```

```
/* Function : Permu64IP                                          */
/* Using    : permutation 64 bits block using DES IP pattern     */
/* Remark   : using shift method so it can't modify easily       */
/*            but much more speed than Permu64                   */

Permu64IP( Long Input[2] , Long Output[2])
{
register Char Tmp;
Char          Temp[8];

register int i;

   Output[0] = 0L;  Output[1] = 0L;
   for(i=0; i<8; i++) {
Temp[i] = 0;
   }


   Tmp = (Char)(Input[1]);              /* Bit 58 60 62 64 57 59 61 63
   (Char)Temp[4] =  Tmp & B1C;
   (Char)Temp[0] = (Tmp << 1) & B1C;
   (Char)Temp[5] = (Tmp << 2) & B1C;
   (Char)Temp[1] = (Tmp << 3) & B1C;
   (Char)Temp[6] = (Tmp << 4) & B1C;
   (Char)Temp[2] = (Tmp << 5) & B1C;
   (Char)Temp[7] = (Tmp << 6) & B1C;
```

```
(Char)Temp[3] = (Tmp << 7) & B1C;

Tmp = (Char)(Input[1] >>  8);       /* Bit 50 52 54 56 49 51 53 55 *
(Char)Temp[4] |= (Tmp >> 1) & B2C;
(Char)Temp[0] |=  Tmp       & B2C;
(Char)Temp[5] |= (Tmp << 1) & B2C;
(Char)Temp[1] |= (Tmp << 2) & B2C;
(Char)Temp[6] |= (Tmp << 3) & B2C;
(Char)Temp[2] |= (Tmp << 4) & B2C;
(Char)Temp[7] |= (Tmp << 5) & B2C;
(Char)Temp[3] |= (Tmp << 6) & B2C;

Tmp = (Char)(Input[1] >> 16);      /* Bit 42 44 46 48 41 43 45 47 */
(Char)Temp[4] |= (Tmp >> 2) & B3C;
(Char)Temp[0] |= (Tmp >> 1) & B3C;
(Char)Temp[5] |=  Tmp       & B3C;
(Char)Temp[1] |= (Tmp << 1) & B3C;
(Char)Temp[6] |= (Tmp << 2) & B3C;
(Char)Temp[2] |= (Tmp << 3) & B3C;
(Char)Temp[7] |= (Tmp << 4) & B3C;
(Char)Temp[3] |= (Tmp << 5) & B3C;

Tmp = (Char)(Input[1] >> 24);
(Char)Temp[4] |= (Tmp >> 3) & B4C;
(Char)Temp[0] |= (Tmp >> 2) & B4C;
(Char)Temp[5] |= (Tmp >> 1) & B4C;
(Char)Temp[1] |= (Tmp )     & B4C;
(Char)Temp[6] |= (Tmp << 1) & B4C;
(Char)Temp[2] |= (Tmp << 2) & B4C;
```

```
(Char)Temp[7] |= (Tmp << 3) & B4C;
(Char)Temp[3] |= (Tmp << 4) & B4C;

Tmp = (Char)(Input[0]);
(Char)Temp[4] |= (Tmp >> 4) & B5C;
(Char)Temp[0] |= (Tmp >> 3) & B5C;
(Char)Temp[5] |= (Tmp >> 2) & B5C;
(Char)Temp[1] |= (Tmp >> 1) & B5C;
(Char)Temp[6] |= (Tmp )     & B5C;
(Char)Temp[2] |= (Tmp << 1) & B5C;
(Char)Temp[7] |= (Tmp << 2) & B5C;
(Char)Temp[3] |= (Tmp << 3) & B5C;

Tmp = (Char)(Input[0] >>  8);
(Char)Temp[4] |= (Tmp >> 5) & B6C;
(Char)Temp[0] |= (Tmp >> 4) & B6C;
(Char)Temp[5] |= (Tmp >> 3) & B6C;
(Char)Temp[1] |= (Tmp >> 2) & B6C;
(Char)Temp[6] |= (Tmp >> 1) & B6C;
(Char)Temp[2] |= (Tmp )     & B6C;
(Char)Temp[7] |= (Tmp << 1) & B6C;
(Char)Temp[3] |= (Tmp << 2) & B6C;

Tmp = (Char)(Input[0] >> 16);
(Char)Temp[4] |= (Tmp >> 6) & B7C;
(Char)Temp[0] |= (Tmp >> 5) & B7C;
(Char)Temp[5] |= (Tmp >> 4) & B7C;
(Char)Temp[1] |= (Tmp >> 3) & B7C;
(Char)Temp[6] |= (Tmp >> 2) & B7C;
```

```
    (Char)Temp[2] |= (Tmp >> 1) & B7C;
    (Char)Temp[7] |= (Tmp )     & B7C;
    (Char)Temp[3] |= (Tmp << 1) & B7C;

    Tmp = (Char)(Input[0] >> 24);
    (Char)Temp[4] |= (Tmp >> 7) & B8C;
    (Char)Temp[0] |= (Tmp >> 6) & B8C;
    (Char)Temp[5] |= (Tmp >> 5) & B8C;
    (Char)Temp[1] |= (Tmp >> 4) & B8C;
    (Char)Temp[6] |= (Tmp >> 3) & B8C;
    (Char)Temp[2] |= (Tmp >> 2) & B8C;
    (Char)Temp[7] |= (Tmp >> 1) & B8C;
    (Char)Temp[3] |= (Tmp )     & B8C;

     Output[0] = ((Long)Temp[0] << 24) | ((Long)Temp[1] << 16) |
((Long)Temp[2] <<  8) |  (Long)Temp[3];
    Output[1] = ((Long)Temp[4] << 24) | ((Long)Temp[5] << 16) |
((Long)Temp[6] <<  8) |  (Long)Temp[7];

}


/* Function : Permu64FP                                          */
/* Using    : permutation 64 bits block using DES inverse IP pattern
/* Remark   : using shift method so it can't modify easily       */
/*            but much more speed than Permu64                   */

Permu64FP( Long Input[2] , Long Output[2])
{
```

```
register Char Tmp;
Char          Temp[8];

register int i:

   Output[0] = 0L;  Output[1] = 0L;
   for(i=0; i<8; i++) {
Temp[i] = 0;
   }

   Tmp = (Char)(Input[1] >> 24);        /* Bit 40 39 38 37 36 35 34 33
   (Char)Temp[7] = Tmp & B1C;
   (Char)Temp[6] = (Tmp << 1) & B1C;
   (Char)Temp[5] = (Tmp << 2) & B1C;
   (Char)Temp[4] = (Tmp << 3) & B1C;
   (Char)Temp[3] = (Tmp << 4) & B1C;
   (Char)Temp[2] = (Tmp << 5) & B1C;
   (Char)Temp[1] = (Tmp << 6) & B1C;
   (Char)Temp[0] = (Tmp << 7) & B1C;

   Tmp = (Char)(Input[0] >> 24);        /* Bit S 7 6 5 4 3 2 1 */
   (Char)Temp[7] |= (Tmp >> 1) & B2C:
   (Char)Temp[6] |=  Tmp       & B2C;
   (Char)Temp[5] |= (Tmp << 1) & B2C:
   (Char)Temp[4] |= (Tmp << 2) & B2C;
   (Char)Temp[3] |= (Tmp << 3) & B2C:
   (Char)Temp[2] |= (Tmp << 4) & B2C;
   (Char)Temp[1] |= (Tmp << 5) & B2C;
```

```
(Char)Temp[0] |= (Tmp << 6) & B2C;

Tmp = (Char)(Input[1] >> 16):        /* Bit 48 47 46 45 44 43 42 41 *
(Char)Temp[7] |= (Tmp >> 2) & B3C;
(Char)Temp[6] |= (Tmp >> 1) & B3C:
(Char)Temp[5] |=  Tmp       & B3C;
(Char)Temp[4] |= (Tmp << 1) & B3C:
(Char)Temp[3] |= (Tmp << 2) & B3C;
(Char)Temp[2] |= (Tmp << 3) & B3C;
(Char)Temp[1] |= (Tmp << 4) & B3C;
(Char)Temp[0] |= (Tmp << 5) & B3C;

Tmp = (Char)(Input[0] >> 16);        /* Bit 16 15 14 13 12 11 10  9 *
(Char)Temp[7] |= (Tmp >> 3) & B4C;
(Char)Temp[6] |= (Tmp >> 2) & B4C;
(Char)Temp[5] |= (Tmp >> 1) & B4C;
(Char)Temp[4] |=  Tmp       & B4C;
(Char)Temp[3] |= (Tmp << 1) & B4C;
(Char)Temp[2] |= (Tmp << 2) & B4C;
(Char)Temp[1] |= (Tmp << 3) & B4C;
(Char)Temp[0] |= (Tmp << 4) & B4C;

Tmp = (Char)(Input[1] >> 8);         /* Bit 56 55 54 53 52 51 50 49 */
(Char)Temp[7] |= (Tmp >> 4) & B5C;
(Char)Temp[6] |= (Tmp >> 3) & B5C;
(Char)Temp[5] |= (Tmp >> 2) & B5C;
(Char)Temp[4] |= (Tmp >> 1) & B5C;
(Char)Temp[3] |=  Tmp       & B5C;
(Char)Temp[2] |= (Tmp << 1) & B5C;
```

```
(Char)Temp[1] |= (Tmp << 2) & B5C;
(Char)Temp[0] |= (Tmp << 3) & B5C;

Tmp = (Char)(Input[0] >> 8);        /* Bit 24 23 22 21 20 19 18 17 */
(Char)Temp[7] |= (Tmp >> 5) & B6C;
(Char)Temp[6] |= (Tmp >> 4) & B6C;
(Char)Temp[5] |= (Tmp >> 3) & B6C;
(Char)Temp[4] |= (Tmp >> 2) & B6C;
(Char)Temp[3] |= (Tmp >> 1) & B6C;
(Char)Temp[2] |=  Tmp       & B6C;
(Char)Temp[1] |= (Tmp << 1) & B6C;
(Char)Temp[0] |= (Tmp << 2) & B6C;

Tmp = (Char)(Input[1]);             /* Bit 64 63 62 61 60 59 58 57 */
(Char)Temp[7] |= (Tmp >> 6) & B7C;
(Char)Temp[6] |= (Tmp >> 5) & B7C;
(Char)Temp[5] |= (Tmp >> 4) & B7C;
(Char)Temp[4] |= (Tmp >> 3) & B7C;
(Char)Temp[3] |= (Tmp >> 2) & B7C;
(Char)Temp[2] |= (Tmp >> 1) & B7C;
(Char)Temp[1] |=  Tmp       & B7C;
(Char)Temp[0] |= (Tmp << 1) & B7C;

Tmp = (Char)(Input[0]);             /* Bit 32 31 30 29 28 27 26 25 */
(Char)Temp[7] |= (Tmp >> 7) & B8C;
(Char)Temp[6] |= (Tmp >> 6) & B8C;
(Char)Temp[5] |= (Tmp >> 5) & B8C;
(Char)Temp[4] |= (Tmp >> 4) & B8C;
(Char)Temp[3] |= (Tmp >> 3) & B8C;
```

```
    (Char)Temp[2] |= (Tmp >> 2) & B8C;
    (Char)Temp[1] |= (Tmp >> 1) & B8C;
    (Char)Temp[0] |=  Tmp       & B8C;

    Output[0] = ((Long)Temp[0] << 24) | ((Long)Temp[1] << 16) |
        ((Long)Temp[2] <<  8) |  (Long)Temp[3];
    Output[1] = ((Long)Temp[4] << 24) | ((Long)Temp[5] << 16) |
        ((Long)Temp[6] <<  8) |  (Long)Temp[7];

}

/* Function F                          */
/* The important part of DES           */

Long  F_V2( Long R, Char *Key )
{
Char ER[8];
Long S_Out=0L, Output=0L;
register Char rc;
register int j;

       Expand_V2( R, ER );
       for (j=0; j<8; j++) {
    rc = ER[j] ^ Key[j];
    rc = ( rc & 0x20 ) | ( ( rc << 4 ) & 0x10 ) |
( ( rc >> 1 ) & 0x0F );

    S_Out    |= S_V2[j][rc];
       }
```

```
        Permu32_V3( &S_Out, &Output );

        return ( Output );
}


/*          This function is used for expand Right side in          */
/*        function F from 32 -> 48 bits using defined pattern.      */
/*                                                                  */
/*     Function : Expand ( R, Output )                              */
/*        Input : Long R  32 bits input from function F             */
/*       Output : char * Output  ( 8 char = 64 bits but use 48 bits) */
/*     Use bits :  R => [1,2,...,32] , Output => [x,x,1,2,3,4,5,6]  */
/*                 xx in Output is unused in program                */
/*                                                                  */
/*         Note : This function is not used array pattern but       */
/*                programmed in other style for the reason of       */
/*          more speed.                                             */


Expand_V2( Long R, Char *Output )
{
#define CF2  0x3F       /* For clear first 2 bits => 00xxxxxx */
register Char Tmp[4];

        Tmp[0]  =  (Char)(R >> 24);
        Tmp[1]  =  (Char)(R >> 16);
```

```
    Tmp[2]  =  (Char)(R >>  8);
    Tmp[3]  =  (Char)(R);

    Output[0]  =  ( (Tmp[0] >> 3)  |  (Tmp[3] << 5) ) & CF2;
    Output[1]  =  ( (Tmp[0] << 1)  |  (Tmp[1] >> 7) ) & CF2;
    Output[2]  =  ( (Tmp[0] << 5)  |  (Tmp[1] >> 3) ) & CF2;
    Output[3]  =  ( (Tmp[1] << 1)  |  (Tmp[2] >> 7) ) & CF2;
    Output[4]  =  ( (Tmp[1] << 5)  |  (Tmp[2] >> 3) ) & CF2;
    Output[5]  =  ( (Tmp[2] << 1)  |  (Tmp[3] >> 7) ) & CF2;
    Output[6]  =  ( (Tmp[2] << 5)  |  (Tmp[3] >> 3) ) & CF2;
    Output[7]  =  ( (Tmp[3] << 1)  |  (Tmp[0] >> 7) ) & CF2;
}


Permu32_V3( Long *Input, Long *Output )
{
register Char Tmpin[8];
Char  TmpOut[8];

    Tmpin[0] = (char)(*Input >> 24);
    Tmpin[1] = (char)(*Input >> 16);
    Tmpin[2] = (char)(*Input >>  8);
    Tmpin[3] = (char)(*Input ):

    TmpOut[0] = ((Tmpin[1] << 7) & B1C) |
 ((Tmpin[0] << 5) & B2C) |
 ((Tmpin[2] << 1) & ( B3C | B4C )) |
 ( Tmpin[3]       & B5C) |
 ((Tmpin[1] >> 2) & B6C) |
```

```
((Tmpin[3] >> 3) & B7C) |
((Tmpin[2] >> 7) & B8C);

    TmpOut[1] = ( Tmpin[0]       & B1C) |
((Tmpin[1] << 5) & B2C) |
((Tmpin[2] << 4) & B3C) |
((Tmpin[3] >> 2) & B4C) |
( Tmpin[0]       & B5C) |
((Tmpin[2] >> 4) & B6C) |
( Tmpin[3]       & B7C) |
((Tmpin[1] >> 6) & B8C);

    TmpOut[2] = ((Tmpin[0] << 1) & B1C) |
((Tmpin[0] << 6) & B2C) |
((Tmpin[2] << 5) & B3C) |
((Tmpin[1] << 2) & B4C) |
((Tmpin[3] << 3) & B5C) |
((Tmpin[3] >> 3) & B6C) |
((Tmpin[0] >> 4) & B7C) |
((Tmpin[1] >> 7) & B8C);

    TmpOut[3] = ((Tmpin[2] << 2) & B1C) |
((Tmpin[1] << 3) & B2C) |
((Tmpin[3] << 3) & B3C) |
((Tmpin[0] << 2) & B4C) |
((Tmpin[2] << 1) & B5C) |
((Tmpin[1] >> 3) & B6C) |
((Tmpin[0] >> 3) & B7C) |
((Tmpin[3] >> 7) & B8C);
```

```
    *Output =   ((Long)TmpOut[0] << 24) ¦ ((Long)TmpOut[1] << 16) ¦
 ((Long)TmpOut[2] <<  8) ¦ ((Long)TmpOut[3]);
}


DES_Encrypt_V2(Long Input[2])
{
Long LR[2],Temp;
register int i;

Permu64IP ( Input, LR );

for (i=0; i<16; i++) {          /* Do 16 Iteration of Encrypt */
     LR[0]  ^=  F_V2( LR[1], SubKey[i] );
     Temp    =  LR[0];
     LR[0]   =  LR[1];          /*  Left <- Right             */
     LR[1]   =  Temp;           /*  Left -> Right             */
     }
     Swap( LR );    /* The final round  */
    /* don't change side */

Permu64FP ( LR, Input );
}

DES_Decrypt_V2(Long Input[2])
{
Long LR[2],Temp;
register int i;
```

```
Permu64IP ( Input, LR );

for (i=15; i>=0; i--) {          /* Do 16 Iteration of Encrypt */
     Temp    =  LR[0] ^ F_V2( LR[1], SubKey[i] );
     LR[0]   =  LR[1];           /*  Left <- Right              */
     LR[1]   =  Temp;            /*  Left -> Right              */
     }
             LR[1]   =  LR[0];          /* The final round  */
     LR[0]   =  Temp;            /* don't change side */

Permu64FP ( LR, Input );
}


/* this is F function use for UDES method */
Long  U_F( Long R, Char *Key )
{
Char ER[8],S_Val;
Long S_Out=0L, Output=0L;
register Char rc;
register int j;

      Expand_V2( R, ER );
      for (j=0; j<8; j++) {
   rc = ER[j] ^ Key[j];
   rc = ( rc & 0x20 ) | ( ( rc << 4 ) & 0x10 ) |
( ( rc >> 1 ) & 0x0F );
```

```
        S_Val    = S[j][rc];
        S_Out    = ( S_Out << 4 ) | S_Val;
        }

        Permu32_V3( &S_Out, &Output );

        return ( Output );
}



UDES_Encrypt(Long Input[2])
{
Long LR[2],Temp;
register int i;

Permu64IP ( Input, LR );

for (i=0; i<16; i++) {          /* Do 16 Iteration of Encrypt */
      LR[0]  ^=  U_F( LR[1], SubKey[i] );
      Temp    =  LR[0];
      LR[0]   =  LR[1];          /*  Left <- Right             */
      LR[1]   =  Temp;           /*  Left -> Right             */
      }
      Swap( LR );     /* The final round  */
     /* don't change side */

Permu64FP ( LR, Input );
}
```

```
UDES_Decrypt(Long Input[2])
{
Long LR[2],Temp;
register int i;

Permu64IP ( Input, LR );

for (i=15; i>=0; i--) {          /* Do 16 Iteration of Encrypt */
     Temp    =  LR[0] ^ U_F( LR[1], SubKey[i] );
     LR[0]   =  LR[1];           /*  Left <- Right              */
     LR[1]   =  Temp;            /*  Left -> Right              */
     }
          LR[1]   =  LR[0];           /* The final round  */
     LR[0]   =  Temp;            /* don't change side */

Permu64FP ( LR, Input );
}
```

Appendix B

File : Cryptkey.c

Function : recieve password key from user.

```
/*****************************************************************/
/*  CryptKey.c : File for find key for encrypt                   */
/*                                                               */
/*  Last Update 12 January,1993                                  */
/*****************************************************************/

#include <conio.h>
#include <string.h>
#include <stdlib.h>

/* #include "bitdef.c"
#include "cursor.c"
#include "getkey.c"
#include "play.c"
#include "textbox.c" */

#include "B:\text\bitdef.c"
#include "B:\text\getkey.h"
#include "B:\text\textbox.h"

#define MIN_PASSWORD_LENGTH   5
#define MAX_PASSWORD_LENGTH  40

#define ErrorSong  play("T900 O3 CDCD$")
#define OKsong     play("T1000 O4 GA O5 D$")

/* Function : cryptkey */
/* Input    : char *k  */
/* For      : get user key and find another key for crypt in memory */
```

```
/*              and do Subkey form it */


int InputEncryptKey(char *k , int func )
{


#define x1 10
#define x2 70
#define y1  9
#define y2 14
#define bkcolor  LIGHTBLUE
#define txtcolor WHITE

char *Input1,*Input2;
void *m1,*m2;    /* memory pointer to allocated display box */
int retval;



    /* Allocate memory for input 1 and 2   and */
    /* Initial Key 'Input1' and 'Input2' string  to "\0"*/
    Input1 = (char *)malloc(MAX_PASSWORD_LENGTH + 1):
    Input2 = (char *)malloc(MAX_PASSWORD_LENGTH + 1);
    strcpy(Input1,"");  strcpy(Input2,"") ;
    retval = 0;

    do {
 /* Input Password 1  and  keep it in  'Input1' */
 do {
     m1 = malloc( 2*(x2+2-x1+1)*(y2+2-y1) );
```

```
    gettext(x1,y1,x2+2,y2+1,m1);

    ExplodeInputBox(x1,y1,x2,y2);

    if ( func == 0 ) {
writetext(x1+14,y1+2,"Enter file encryption password",bkcolor<<4¦txtcolor);
    }
    else {
writetext(x1+14,y1+2,"Enter file decryption password",bkcolor<<4¦txtcolor);
    }
    writeattr(x1+10,y1+3,BLACK<<4 ¦ WHITE,40);

    gotoxy(9,3);
    textbackground(BLACK);
    retval = Read_Password(Input1);

    puttext(x1,y1,x2+2,y2+1,m1);

    if ( retval == -1  )  {
retval = 0;
free( m1 );
                free(Input1);
free(Input2);
return(-1);
    }

    if ( strlen(Input1) < MIN_PASSWORD_LENGTH ) {
 PassError();
    }
```

```
      free(m1);

  } while ( strlen(Input1) < MIN_PASSWORD_LENGTH );

    if( func == 0)  {
  /* Input Password 2 for verification with Input1 */
  do {

     m2 = malloc( 2*(x2+2-x1+1)*(y2+2-y1) );
     gettext(x1,y1,x2+2,y2+1,m2);

     ExplodeInputBox(x1,y1,x2,y2);

     writetext(x1+4,y1+2,"Re-enter file encryption password for verification",
     bkcolor<<4|txtcolor);
     writeattr(x1+10,y1+3,BLACK<<4 | WHITE,40);

     gotoxy(9,3);
     textbackground(BLACK);

     retval = Read_Password(Input2);

     puttext(x1,y1,x2+2,y2+1,m2);

     if ( retval == -1 ) {
free(m2);
                free(Input1);
free(Input2);
```

```
return(-1);
     }

     if ( strlen(Input2) < MIN_PASSWORD_LENGTH ) {
 PassError();
     }

     free(m2);

  } while ( strlen(Input2) < MIN_PASSWORD_LENGTH );

 if( strcmp( Input1,Input2 ) ) {
      PassUnequal();
 }

     } /* end of func == 1 */

     if ( func == 1 ) {
 strcpy(Input2, Input1);
     }

     } while ( strcmp( Input1,Input2 ) );

     strcpy( k, Input2);

     free(Input1);
     free(Input2);

#undef x1
```

```
#undef x2
#undef y1
#undef y2
#undef bkcolor
#undef txtcolor
}


int Read_Password(char *ky)
{
#define MAXPASS   40
int i,ch;
char kin[MAXPASS+1] = "";

     CurOn();
     for (i=0;  ; i++)  {
  ch = get_key();
  switch(ch) {
  case  KEY_ESC    : return(-1);
  case  KEY_ENTER : strcpy(ky,kin);
    return(0);
  case  '\b'       : if (i != 0 ) {
       *(kin+i) = '\0';
       --i;
       *(kin+i) = '\0';
       putch('\b');
       putch('\0');
       putch('\b');
       --i;
```

```
        break;
    }
    else {
        --i;
        break;
    }

  default          : if ( (i < MAXPASS) && (ch < 0xFF) ) {
        *(kin+i) = ch;
        putch('*');
        break;
     }
     else  {
--i;
break;
     }
  }
     }
}

ScramKey( Long Output[2], Char *Input)
{
int i,j;
Char TmpInput[MAX_PASSWORD_LENGTH + 1];
Long TmpOut[2];

      strcpy( TmpInput, Input );
      for( i=MIN_PASSWORD_LENGTH, j=0;
   (*(Input+i) != '\0') && (i < MAX_PASSWORD_LENGTH); i++ ,j++) {
```

```
    *(TmpInput+j) ^= *(Input+i);
    if ( j > 7 )  { j = 0; }
       }

       TmpOut[0] = ( (Long)TmpInput[0] << 24 )  ¦
   ( (Long)TmpInput[1] << 16 )  ¦
   ( (Long)TmpInput[2] <<  8 )  ¦
   ( (Long)TmpInput[3] );

       TmpOut[1] = ( (Long)TmpInput[4] << 24 )  ¦
   ( (Long)TmpInput[5] << 16 )  ¦
   ( (Long)TmpInput[6] <<  8 )  ¦
   ( (Long)TmpInput[7] );

       Permu64IP ( TmpOut, Output );

}

/* Show error that password is too short */
PassError()
{
#define x1 22
#define x2 57
#define y1  8
#define y2 16
#define style     SINGLE
#define boxcolor  WHITE
#define bkcolor   LIGHTRED
```

```
#define txtcolor  YELLOW

void *m1;

    CurOff();

    m1 = malloc( 2*(x2+2-x1+1)*(y2+2-y1) );
    gettext(x1,y1,x2+2,y2+1,m1);

    textbox(x1+12,y1+2,x2-12,y2-2,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
    freeze(100);
    textbox(x1+7,y1+1,x2-7,y2-1,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
    freeze(100);
    textbox(x1,y1,x2,y2,style,boxcolor,bkcolor,txtcolor,SHADOW);

    gotoxy(7,1);
    textattr(bkcolor<<4|txtcolor);
    cputs(" Input Password Error \r\n");

    textattr(bkcolor<<4|WHITE);
    cputs("\n Password must contain at least\r\n");
    cputs  ("      eight character long\r\n");
    writeattr(x1+8,y1+4,(bkcolor<<4) | txtcolor,5);

    ErrorSong;

    WriteOK(x1+10,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+11,y2-2,bkcolor,NOSHADOW);
```

```
    OKsong;
    WriteOK(x1+10,y2-2,bkcolor,SHADOW);
    freeze(150);

    puttext(x1,y1,x2+2,y2+1,m1);
    free( m1 );

    CurOn();

#undef x1
#undef x2
#undef y1
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}

/* Show error that password that entered is not equal */
PassUnequal()
{
#define x1 17
#define x2 62
#define y1  9
#define y2 18
#define style    SINGLE
#define boxcolor WHITE
#define bkcolor  LIGHTRED
```

```
#define txtcolor YELLOW

void *m1;

      CurOff();

      m1 = malloc( 2*(x2+2-x1+1)*(y2+2-y1) );
      gettext(x1,y1,x2+2,y2+1,m1);

      textbox(x1+20,y1+3,x2-20,y2-3,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
      freeze(100);
      textbox(x1+15,y1+2,x2-15,y2-2,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
      freeze(50);
      textbox(x1+10,y1+1,x2-10,y2-1,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
      freeze(50);
      textbox(x1,y1,x2,y2,style,boxcolor,bkcolor,txtcolor,SHADOW);

      writetext(x1+12,y1," Input Password Error ",bkcolor<<4 | txtcolor);
      writetext(x1+12,y1+2,"Passwords do not equal",bkcolor<<4 | txtcolor);
      writetext(x1+3,y1+4,"Make sure that you remember the password",bkcolor<<4 | WHIT
      writetext(x1+10,y1+5,"and type it in correctly.",bkcolor<<4 | WHITE);

      ErrorSong;

      WriteOK(x1+14,y2-2,bkcolor,SHADOW);
      getch();
      WriteOK(x1+15,y2-2,bkcolor,NOSHADOW);
      OKsong;
      WriteOK(x1+14,y2-2,bkcolor,SHADOW);
```

```
        freeze(150);


        puttext( x1,y1,x2+2,y2+1,m1 );
        free( m1 );

        CurOn();

#undef x1
#undef x2
#undef y1
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}

writeOK(int x,int y)
{
    writetext(x,y,"    OK    ",LIGHTGRAY<<4 | BLACK);
    writetext(x+10,y,"",RED<<4 | BLACK);
    writetext(x+1,y+1," ",RED<<4 |BLACK);
}

/* function for making explode box of enter password and re-enter */
ExplodeInputBox(int x1,int y1,int x2,int y2)
{
#define style      SINGLE
```

```
#define boxcolor    WHITE
#define bkcolor     LIGHTBLUE
#define txtcolor    WHITE

     textbox(x1+25,y1+1,x2-25,y2-1,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
     freeze(100);
     textbox(x1+20,y1+1,x2-20,y2-1,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+15,y1+1,x2-15,y2-1,style,boxcolor,bkcolor,txtcolor,NOSHADOW);
     freeze(50);

     textbox(x1,y1,x2,y2,style,boxcolor,bkcolor,txtcolor,SHADOW);

#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}
```

Appendix C

File : **Read.c**

Function : read,write data management and error handler.

```
#include <alloc.h>
#include <conio.h>
#include <dir.h>
#include <dos.h>
#include <io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "B:\text\getkey.h"
#include "B:\text\textbox.h"

#include "B:\text\bitdef.c"

/* #include "B:\text\udes.c" */

#define scolor  WHITE    /* Select bar color */
#define hibar   scolor<<4 | BLUE

#define Errorsong  play("T900O3CDCD$")
#define OKsong     play("T1000O4GAO5D$")
#define Click1     play("T500L32O4EB$")
#define MainS      play("T1000 O4CAD$")
/* header file structer */

#define PROJNAME   "CRYPT     1.00"
/* pre header for all type not be encrypted */
struct phead {
```

```
        char projname[16];
        int  method;
}phead;

/* main header for all type */
struct header {
        char attr;
        struct ftime ftime;
        long size;
        char name[13];
        char password[41];
        int option;
}header;

struct UDES {
        char UDES_S[8][64];
}UDES;

/* global variable */
static int vga;          /* 1 if it is vga card */
static int Func = 0;     /* Function define  0 == Encryption */
 /*                      1 == Decryption */
 /*                      2 == Set Options */
 /*                      3 == eXit  */
static int Method = 0;   /* Method define    0 == DES   */
 /*                      1 == ? */
static int Option;       /* Option define    0 == no options */

extern Char DES_S[8][64];
```

```
extern Char S[8][64];
       Char Sbuff[8][64];


Initialize()
{
register int i,j;
     if (getcrtmode() != 7) {    /* color card */
  vga = 1;
  Initcolor();
 /*  Initchar(); */
  EnableBlink(0);        /* enable background intensify */
     }

     /* init S box data */
     for (i=0; i<8 ;i++)  {
for (j=0; j<64; j++)  {
   UDES.UDES_S[i][j] = S[i][j];
   Sbuff[i][j]       = S[i][j];
}
     }

}


WriteFirstScr()
{
     BlankScr();
     FirstScr();
```

```
}

Demoname()
{
#define x1    18
#define y1     4
#define x2    62
#define y2    20
#define bkcolor  LIGHTBLUE
#define txtcolor WHITE
#define boxcolor LIGHTGREEN


void *m1;
char *c = "    \nEngineering Project Year 1992,1993\r\n\n"
  "          Data encryption program\r\n"
  "               Crypt 1.0 \r\n\n"
  "                 Advisor\r\n"
  "             Dr.Fusak Cheevasuvit\r\n\n"
  "                 Produced by\r\n"
  "       Pornvit Saksobhavivat 32-1205\r\n"
  "       Saran Anantathavanich 32-1323\r\n";

     CurOff();
     m1 = malloc( 2 * (x2+3-x1) * (y2+2-y1)  );
     gettext(x1,y1,x2+2,y2+1,m1);

     /* make explode box */
     textbox(x1+18,y1+6,x2-18,y2-6,SINGLE,boxcolor,
```

```
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);
    textbox(x1+12,y1+4,x2-12,y2-4,SINGLE,boxcolor,
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);
    textbox(x1+7,y1+1,x2-7,y2-1,SINGLE,boxcolor,
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);

    textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
    textattr( bkcolor<<4 | txtcolor);
    cprintf("%s",c);
    WriteOK(x1+16,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+17,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+16,y2-2,bkcolor,SHADOW);
    freeze(150);

    puttext(x1,y1,x2+2,y2+1,m1);
    free(m1);

#undef x1
#undef y1
#undef x2
#undef y2
#undef bkcolor
#undef txtcolor
#undef boxcolor
```

```
}

WriteOK(int x,int y,int bkcolor,int shadow)
{
     writetext(x,y,"     OK     ",hibar);
     if ( shadow )  {
  writetext(x+14,y,"",bkcolor<<4 | BLACK);
  writetext(x+1,y+1,"▄▄▄▄▄▄▄▄▄▄▄▄▄▄",bkcolor<<4 | BLACK);
     }
     else  {
  writetext(x-1,y," ",bkcolor<<4 | BLACK);
  writetext(x,y+1,"                ",bkcolor<<4 | BLACK);
     }
}


void WriteMenu(int position[7])
{
#define x1  10
#define y1   5
#define x2  70
#define y2  20
#define bkcolor  LIGHTBLUE
#define boxcolor WHITE
#define txtcolor WHITE
#define normbar  LIGHTGRAY
#define normhide normbar<<4  | normbar /* for hidding arrow */
#define normmenu normbar<<4  | BLACK   /* normal menu attr */
#define sdatb    bkcolor<<4  | BLACK   /* shadow attribute */
```

```
#define expattr  bkcolor<<4 | WHITE   /* explain attribute */

int xpos1,ypos1,barsize;

     xpos1 = x1+5;  position[0] = xpos1;
     ypos1 = y1+3;  position[1] = ypos1;
     barsize = 18;  position[2] = barsize;
    position[3] = x1;
    position[4] = y1;
    position[5] = x2;
    position[6] = y2;


    /* make explode box */
    textbox(x1+25,y1+6,x2-25,y2-6,SINGLE,boxcolor,
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);
    textbox(x1+20,y1+4,x2-20,y2-4,SINGLE,boxcolor,
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);
    textbox(x1+15,y1+1,x2-15,y2-1,SINGLE,boxcolor,
      bkcolor,txtcolor,NOSHADOW);
    freeze(50);

    textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);


    /* write box header */
    writeattr(x1,y1,scolor<<4|scolor,x2-x1+1);
```

```
    writetext(x1+22,y1," Crypt Main Menu ",hibar<<4 | RED);

    /* write selection box and explaination */
    writetext(xpos1,ypos1, "   Encryption   ",normmenu);
    Tshadow(xpos1,ypos1,normmenu, sdatb, barsize);
    writearrow(xpos1,ypos1,normhide,barsize);
    writetext(xpos1+barsize+5,ypos1,
      "Encryption, make a secret file.",expattr);

    writetext(xpos1,ypos1+3, "   Decryption   ",normmenu);
    Tshadow(xpos1,ypos1+3,normmenu, sdatb, 18);
    writearrow(xpos1,ypos1+3,normhide,barsize);
    writetext(xpos1+barsize+5,ypos1+3,
      "Restore its original file.",expattr);

    writetext(xpos1,ypos1+6,"    Options    ",normmenu);
    Tshadow(xpos1,ypos1+6,normmenu, sdatb, 18);
    writearrow(xpos1,ypos1+6,normhide,barsize);
    writetext(xpos1+barsize+5,ypos1+6,
      "Set various options of Crypt.",expattr);

    writetext(xpos1,ypos1+9,"     eXit      ",normmenu);
    Tshadow(xpos1,ypos1+9,normmenu, sdatb, 18);
    writearrow(xpos1,ypos1+9,normhide,barsize);
    writetext(xpos1+barsize+5,ypos1+9,"Return to DOS.",expattr);


#undef x1
#undef x2
```

```
#undef y1
#undef y2
#undef bkcolor
#undef boxcolor
#undef bkcolor
#undef normbar
#undef normhide
#undef normmenu
#undef sdatb
#undef expattr
}

/* write text shadow of a bar */
/* sd is shadow attr */
Tshadow( int x, int y, int bkattr, int sdattr, int nelem)
{
     writeattr(x,y,bkattr,nelem);  /* write main attribute */
     writentext(x+1,y+1,'',sdattr, nelem); /* write bottom attr */
     writetext(x+nelem,y,"",sdattr); /* write back attr */
}

/* write selection arrow with defined attribute */
writearrow(int x, int y, int attr, int width)
{
     writetext(x,y,"",attr);
     writetext(x+width-1,y,"",attr);
}
```

```
GetEncryptName(char *Infile, char *Outfile, int Func)
{
#define x1     8
#define y1     9
#define x2    72
#define y2    14
#define boxcolor  WHITE
#define bkcolor   LIGHTBLUE
#define txtcolor  WHITE

static char NAME[4][11]  = { "Encryption", "Decryption",
     "Encrypted" , "Decrypted" };
void *m1;

      strcpy(Infile,""); strcpy(Outfile,"");


      m1 = malloc(2* (x2-x1+3) * (y2+2-y1) );
      gettext(x1,y1,x2+2,y2+1,m1);


      textbox(x1+25,y1+1,x2-25,y2-1,SINGLE,boxcolor,
        bkcolor,txtcolor,SHADOW);
      freeze(50);
      textbox(x1+15,y1+1,x2-15,y2-1,SINGLE,boxcolor,
```

```
        bkcolor,txtcolor,SHADOW);
      freeze(50);
      textbox(x1+5,y1+1,x2-5,y2-1,SINGLE,boxcolor,
        bkcolor,txtcolor,SHADOW);
      freeze(50);
      textbox(x1,y1,x2,y2,SINGLE,boxcolor,
        bkcolor,txtcolor,SHADOW);
      gotoxy(8,2); textattr(bkcolor<<4 | boxcolor);
      cprintf("Type in file name for writing %s file to",
        NAME[Func+2]);
      writeattr(x1+3,y1+3, BLACK<<4 | WHITE,59);

      gotoxy(2,3); textattr(BLACK<<4 | WHITE);
      ReadFileName( Outfile );
      if ( strcmp(Outfile,"") == 0 ) {
         puttext(x1,y1,x2+2,y2+1,m1);
 free( m1 );
 CurOff();
 return;
      }

      puttext(x1,y1,x2+2,y2+1,m1);
    }/* end if */

      free( m1 );
      CurOff();

#undef boxcolor
#undef bkcolor
```

```
            putch('\b');
            --i;
            break;
        }
        else {
            --i;
            break;
        }

    default         : if ( (i < MAXFILENAME) && (ch < 0xFF) ) {
            *(kin+i) = ch;
            putch(ch);
            break;
        }
        else {
--i;
break;
        }
    }
        }
}

/* check possibility of read/write file of input and output file */
/* In   is pointer to input file name */
/* Out  is pointer to output file name */

int CheckPoss( char *In, char *Out )
{
struct ffblk ffblk;
```

```
unsigned long Fsize;
struct dfree diskfree;
unsigned long Diskfree;
int Drive;
int manage,flag;
char drive[MAXDRIVE];
char dir[MAXDIR];
char file[MAXFILE];
char ext[MAXEXT];

unsigned long avail;
unsigned int  sclus, bsec;

     /* No specified file  then return */
     if (  (strcmp( In,"" ) == 0 ) || (strcmp( Out,"")== 0)  )  {
       return(-1);
     }

     /* if input file not exist */
     if ( findfirst( In, &ffblk, FA_RDONLY |
             FA_HIDDEN | FA_ARCH ) != 0 ) {
         InFileNotExist();
         return( -1 );
     }

     if( access( Out, 0 ) == 0 ) {  /* if outfile already exist */
         manage = OutfileExist();
         return( manage );
     }
```

```
    /* find that drive has enough free space for writing */

    Fsize = ffblk.ff_fsize;

    flag = fnsplit( Out, drive, dir, file, ext);
    if ( flag & DRIVE )  {
        Drive = drive[0] - 'A';  /* drive A == 0 */
    }
    else {
        Drive = getdisk();
    }

    getdfree( Drive+1, &diskfree );  /* drive A: = 1 */
    avail  =  diskfree.df_avail;  /* available cluster */
    sclus  =  diskfree.df_sclus;      /* sector per cluster */
    bsec   =  diskfree.df_bsec;       /* byte per sector */
   Diskfree = ( (unsigned long)(avail)*(unsigned long)(sclus)*
(unsigned long)(bsec) );


    if ( Diskfree >= Fsize ) {
        return(0);
    }
    else {
        DiskNotEnough( Drive );
        return(-1);
    }
```

```
}


InFileNotExist()
{
#define x1   20
#define y1    7
#define x2   60
#define y2   16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

void *m1;

     CurOff();


     m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
     gettext(x1,y1,x2+2,y2+1.m1);

     textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
        bkcolor,txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+15.y1+2,x2-15,y2-2,SINGLE,boxcolor.
        bkcolor.txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
```

```
        bkcolor,txtcolor,NOSHADOW);
    freeze(50);

    textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,
        txtcolor,SHADOW);
    writetext(x1+12,y1+2,"Input file error",bkcolor<<4 | YELLOW );
    writetext(x1+8,y1+4,"Input file is not exist",
        bkcolor<<4 | txtcolor);
    writetext(x1+5,y1+5,
    "in defined drive and directory",bkcolor<<4 | txtcolor);

    Errorsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+14,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    freeze(150);

    puttext(x1,y1,x2+2,y2+1,m1);
    free( m1 );

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
```

```
#undef txtcolor
}

int  OutfileExist()
{
#define x1  20
#define y1   7
#define x2  60
#define y2  16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

#define normbar  LIGHTGRAY
#define normhide normbar<<4  | normbar /* for hidding arrow */
#define normmenu normbar<<4  | BLACK   /* normal menu attr */
#define sdatb    bkcolor<<4  | BLACK   /* shadow attribute */
#define hideattr bkcolor<<4  | bkcolor /* hide by it background */

void *m1,*m2;
int xpos[2].ypos,barsize,ch,i;

     CurOff();

     m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
     gettext(x1,y1,x2+2.y2+1.m1);

     textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
```

```
    bkcolor,txtcolor,NOSHADOW);
freeze(50);
textbox(x1+15,y1+2,x2-15,y2-2,SINGLE,boxcolor,
    bkcolor,txtcolor,NOSHADOW);
freeze(50);
textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
    bkcolor,txtcolor,NOSHADOW);
freeze(50);

textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
writetext(x1+12,y1+2,"Output File Exist",bkcolor<<4 | YELLOW );
writetext(x1+6,y1+4,"Output file is already exist",
      bkcolor<<4 | txtcolor);
writetext(x1+5,y1+5,"In defined drive and directory",
      bkcolor<<4 | txtcolor);


Errorsong;

xpos[0] = x1+3;  xpos[1] = x1+23;
ypos  = y2-2;
barsize = 15;
writetext(xpos[0],ypos,"   Overwrite   ",normmenu);
Tshadow(xpos[0],ypos.normmenu.sdatb,barsize);
writearrow(xpos[0],ypos,normhide,barsize);
writetext(xpos[1],ypos,"     Cancel    ",normmenu);
Tshadow(xpos[1],ypos,normmenu,sdatb,barsize);
writearrow(xpos[1],ypos,normhide,barsize);
```

```
      /* Choose overwrite bar */
      writeattr(xpos[0],ypos,hibar,barsize);
      i = 0;

      do  {
ch = get_key();
switch ( ch )   {
    case  KEY_RIGHT :
    case  KEY_LEFT  :  if (i == 0)  {
        writeattr(xpos[i],ypos,normmenu,barsize);
        writearrow(xpos[i],ypos,normhide,barsize);
        writeattr(xpos[++i],ypos,hibar,barsize);
       }
       else {
        writeattr(xpos[i],ypos,normmenu,barsize);
        writearrow(xpos[i],ypos,normhide,barsize);
        writeattr(xpos[--i],ypos,hibar,barsize);
       }
       break;

   case  KEY_ENTER :
      m2 = malloc( (barsize+2)*2 );
      Tshadow(xpos[i],ypos,hibar,hideattr,barsize);
      gettext(xpos[i]-1,ypos,xpos[i]+barsize-1,ypos,m2);
      puttext(xpos[i], ypos, xpos[i]+barsize, ypos, m2);
      OKsong;
      puttext(xpos[i]-1,ypos,xpos[i]+barsize-1,ypos,m2);
      Tshadow(xpos[i],ypos,hibar,sdatb,barsize);
      freeze(150);
```

```
        free( m2 );

        break;
    default          :  break;
}
     }while (ch != KEY_ENTER );


     puttext(x1,y1,x2+2,y2+1,m1);
     free( m1 );

     if (i == 0)  { return(0); }
     else return(-1);

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor

#undef normbar
#undef normhide
#undef normmenu
#undef sdatb
#undef hideattr
}
```

```
DiskNotEnough(int Drive)
{
#define x1  20
#define y1   7
#define x2  60
#define y2  16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

void *m1;

     CurOff();

     m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
     gettext(x1,y1,x2+2,y2+1,m1);

     textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
           bkcolor,txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+15,y1+2,x2-15,y2-2,SINGLE,boxcolor,
           bkcolor,txtcolor.NOSHADOW);
     freeze(50);
     textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
           bkcolor,txtcolor,NOSHADOW);
     freeze(50);
```

```
        textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
        writetext(x1+12,y1+2,"Not Enough Space",bkcolor<<4 | YELLOW );
        gotoxy(4,4);
        textattr(bkcolor<<4|txtcolor);
        cprintf("Drive %c: has not enough space\r\n\
         for writing output file",Drive+65);

        Errorsong;
        WriteOK(x1+13,y2-2,bkcolor,SHADOW);
        getch();
        WriteOK(x1+14,y2-2,bkcolor,NOSHADOW);
        OKsong;
        WriteOK(x1+13,y2-2,bkcolor,SHADOW);
        freeze(150);

        puttext(x1,y1,x2+2,y2+1,m1);
        free( m1 );

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}
```

```
OpenFileError()
{
#define x1   20
#define y1    7
#define x2   60
#define y2   16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

void *m1;

     CurOff();

     m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
     gettext(x1,y1,x2+2,y2+1,m1);

     textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+15,y1+2,x2-15,y2-2,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
     freeze(50);
     textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
     freeze(50);
```

```
    textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
    writetext(x1+10,y1,"  Open File Error  ",bkcolor<<4 | YELLOW );
    writetext(x1+2,y1+3,"Cannot Open input file or output file",
        bkcolor<<4 | txtcolor);
    writetext(x1+2,y1+4,"     Disk may be write protected",
        bkcolor<<4|txtcolor);

    Errorsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+14,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    freeze(150);

    puttext(x1,y1,x2+2,y2+1,m1);
    free( m1 );

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}

IllegalPassword()
```

```
{
#define x1   20
#define y1    7
#define x2   60
#define y2   16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

void *m1;

      CurOff();

      m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
      gettext(x1,y1,x2+2,y2+1,m1);

      textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
      freeze(50);
      textbox(x1+15,y1+2,x2-15,y2-2,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
      freeze(50);
      textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
          bkcolor,txtcolor,NOSHADOW);
      freeze(50);

      textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
      writetext(x1+10,y1,"  Incorrect Password  ",
```

```
        bkcolor<<4 | YELLOW );
    writetext(x1+2,y1+2,"   You attempt to decrypt the file",
        bkcolor<<4 | txtcolor);
    writetext(x1+2,y1+3,"         with Illegal Password",
        bkcolor<<4 | txtcolor);

    writetext(x1+2,y1+5,"    File Decryption ACCESS DENIED!",
        bkcolor<<4|txtcolor);
    writeattr(x1+22,y1+5,bkcolor<<4|YELLOW,14);


    Errorsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+14,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+13,y2-2,bkcolor,SHADOW);
    freeze(150);

    puttext(x1,y1,x2+2,y2+1,m1);
    free( m1 );

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
```

```
#undef txtcolor
}

int  BadHeader()
{
#define x1  18
#define y1   7
#define x2  62
#define y2  16
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

#define normbar  LIGHTGRAY
#define normhide normbar<<4  | normbar /* for hidding arrow */
#define normmenu normbar<<4  | BLACK   /* normal menu attr */
#define sdatb    bkcolor<<4  | BLACK   /* shadow attribute */
#define hideattr bkcolor<<4  | bkcolor /* hide by it background */

void *m1,*m2;
int xpos[2],ypos,barsize,ch,i;

     CurOff();

     m1 = malloc( 2* (x2-x1+3) * (y2+2-y1) );
     gettext(x1,y1,x2+2,y2+1,m1);

     textbox(x1+17,y1+3,x2-17,y2-3,SINGLE,boxcolor,
```

```
  bkcolor,txtcolor,NOSHADOW);
freeze(50);
textbox(x1+15,y1+2,x2-15,y2-2,SINGLE,boxcolor,
   bkcolor,txtcolor,NOSHADOW);
freeze(50);
textbox(x1+10,y1,x2-10,y2,SINGLE,boxcolor,
   bkcolor,txtcolor,NOSHADOW);
freeze(50);

textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
writetext(x1+14,y1,"  Bad File Header  ",bkcolor<<4 | YELLOW );
writetext(x1+5,y1+2,"This file may not encrypted by CRYPT",
     bkcolor<<4 | txtcolor);
writetext(x1+5,y1+3,"       or it has bad header.",
     bkcolor<<4 | txtcolor);
writetext(x1+3,y1+5,"Trying decryption may recieve bad data!",
     bkcolor<<4 | YELLOW);


Errorsong;

xpos[0] = x1+5;  xpos[1] = x1+25;
ypos  = y2-2;
barsize = 15;
writetext(xpos[0],ypos,"     Cancel    ",normmenu);
Tshadow(xpos[0],ypos,normmenu,sdatb,barsize);
writearrow(xpos[0],ypos,normhide,barsize);
writetext(xpos[1],ypos,"    Continue  ",normmenu);
Tshadow(xpos[1],ypos,normmenu,sdatb,barsize);
```

```
      writearrow(xpos[1],ypos,normhide,barsize);

      /* Choose overwrite bar */
      writeattr(xpos[0],ypos,hibar,barsize);
      i = 0;

      do  {
ch = get_key();
switch ( ch )   {
    case  KEY_RIGHT :
    case  KEY_LEFT  :  if (i == 0)  {
 writeattr(xpos[i],ypos,normmenu,barsize);
 writearrow(xpos[i],ypos,normhide,barsize);
 writeattr(xpos[++i],ypos,hibar,barsize);
       }
       else {
 writeattr(xpos[i],ypos,normmenu,barsize);
 writearrow(xpos[i],ypos,normhide,barsize);
 writeattr(xpos[--i],ypos,hibar,barsize);
       }
       break;

   case  KEY_ENTER :
       m2 = malloc( (barsize+2)*2 );
       Tshadow(xpos[i],ypos,hibar,hideattr,barsize);
       gettext(xpos[i]-1,ypos,xpos[i]+barsize-1,ypos,m2);
       puttext(xpos[i], ypos, xpos[i]+barsize, ypos, m2);
       OKsong;
       puttext(xpos[i]-1,ypos,xpos[i]+barsize-1,ypos,m2);
```

```
        Tshadow(xpos[i],ypos,hibar,sdatb,barsize);
        freeze(150);
        free( m2 );

        break;
    default          :  break;
}
     }while (ch != KEY_ENTER );

     puttext(x1,y1,x2+2,y2+1,m1);
     free( m1 );

     if (i == 0)  { return(0); }
     else return(-1);

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor

#undef normbar
#undef normhide
#undef normmenu
#undef sdatb
```

```
#undef hideattr
}


Do_DES_Encrypt(char *Filein, char *Fileout)
{
#define x1          10
#define y1           9
#define x2          70
#define y2          14
#define bkcolor     LIGHTBLUE

Long work[2],Filesize,Memleft,Allocsize;
register int count;

void *Fbuffin,*Fbuffout;
FILE *fin,*fout;

int    finnum;
int    ratio;
unsigned long i;

/* Key */
Char UserKey[41];
Long Key[2];

/* return value from get key */
int retval;
```

```
/* use for clock */
long start,now;

/* file block */
struct ffblk ffblk;

/* pointer to header for encrypting header */
long *p;

    /* find file for initial ffblk */
    findfirst( Filein, &ffblk, FA_ARCH | FA_HIDDEN | FA_RDONLY );

    /* save old file in attr */
    header.attr = _chmod( Filein, 0 );

    /* change filein attr to archieve for read write */
    _chmod( Filein, 1, FA_ARCH );


    fin  = fopen( Filein ,"r+b");
    fout = fopen( Fileout,"w+b");
    if ( fin == NULL  || fout == NULL ) {
  OpenFileError();
  unlink(Fileout);
  return;
    }
```

```
    /* initial preheader file */
    strcpy(phead.projname,PROJNAME);
    phead.method = Method;
    /* write file preheader */
    fwrite( &phead, sizeof(phead), 1, fout );



    /* find file size */
    finnum = fileno( fin );
    Filesize = filelength( finnum );

    Memleft = coreleft() / 2;
    if ( Memleft >= (Filesize + 8) ) {
Allocsize = Filesize + 8 - (Filesize % 8);
Fbuffin  = malloc(Allocsize);
Fbuffout = malloc(Allocsize);
    }
    else {
Allocsize = Memleft - (Memleft % 8);
Fbuffin  = malloc(Allocsize);
Fbuffout = malloc(Allocsize);
    }

    /* assign buffer to input file  return 0 on success */
    setvbuf( fin, Fbuffin, _IOFBF, Allocsize );
    /* assign buffer to output file */
    setvbuf( fout, Fbuffout, _IOFBF, Allocsize);
```

```
    /* Input encrypt key in string format */
    retval = InputEncryptKey( UserKey ,0);
    if ( retval == -1 )  {
free( Fbuffin );
free( Fbuffout);
fclose(fin);
fclose(fout);
unlink(Fileout);
return;
    }

    /* change the key string to two long int */
    ScramKey( Key, UserKey );

    /* Find subkey for encryption */
    S_Key( Key );

    /* Write scale box */
    A_Scale(x1, y1, x2, y2);


    /* save input file date and time  size and name */
    getftime(finnum, &(header.ftime) );
    header.size     = Filesize;
    strcpy( header.name, ffblk.ff_name );
    strcpy( header.password, UserKey );
    header.option  = Option;
```

```
    p = &header;
    for( i=0; i < (sizeof(header)-(sizeof(header)%8)+8)/8; i++ ) {
        DES_Encrypt_V2( p );
        p += 2;
    }
    fwrite(&header, (sizeof(header)-(sizeof(header)%8)+8),
            1, fout );


    time(&start);
    i = 0;
    do {
  count = fread( (char *)work, 1, 8, fin);
/*  if ( count != 8 ) { ((char *)work)[7]  =  count; } */
  DES_Encrypt_V2( work );
  fwrite( (char *)work, 1, 8, fout);


  if ( (i%250) == 0 ) {
      time(&now);
      ratio =  (  (float) ftell(fin)  / Filesize  ) * 100;
      Writeratio( ratio , now-start );
  }
  i++;
    } while ( count == 8 );

    /* write the buffer to disk */
    flushall();
```

```
    /* restore input file attr */
    _chmod( Filein, 1, header.attr );

    time(&now);
    Writeratio( 100 , now-start );
    freeze(500);

    cputs("\r\n\n\n");
    WriteOK(x1+23,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+24,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+23,y2-2,bkcolor,SHADOW);
    freeze(150);
    FirstScr();


    free( Fbuffin );
    free( Fbuffout );
    fclose(fin);
    fclose(fout);

#undef x1
#undef y1
#undef x2
#undef y2
#undef bkcolor
```

```
}


Do_DES_Decrypt(char *Filein, char *Fileout)
{
#define x1          10
#define y1           9
#define x2          70
#define y2          14
#define bkcolor     LIGHTBLUE

Long work[2],Filesize,Memleft,Allocsize;
register int count;

void *Fbuffin, *Fbuffout;
FILE *fin,*fout;

int   foutnum;

int   finnum,ratio;

Char UserKey[41];
Long Key[2];

int retval;

unsigned long i;
```

```
long start,now:

int validbyte;
Long endsize;

long *p;

/* return status from checkposs */
int status;

     fin  = fopen( Filein ,"r+b");
     if ( fin == NULL ) {
  OpenFileError();
  return;
     }

     /* extract file pre header */
     fread( &phead, sizeof(phead), 1, fin );
     if ( strcmp(phead.projname, PROJNAME) != 0 ) {
status = BadHeader();
if (status == 0) {
   return;
}
     }
     if ( phead.method != 0 ) { return; }
```

```
      /* Input encrypt key in string format */
      retval = InputEncryptKey( UserKey, 1 );
      if ( retval == -1 )  {
/* free( Fbuffin );
free( Fbuffout);  */
fclose(fin);
/* fclose(fout);
unlink(Fileout); */
return;
      }

      /* change the key string to two long int */
      ScramKey( Key, UserKey );

      /* Find subkey for encryption */
      S_Key( Key );

      /* extract file header */
      fread( &header, sizeof(header)-(sizeof(header)%8)+8, 1, fin );
      p = &header;
      for(i=0; i< (sizeof(header)-(sizeof(header)%8)+8)/8; i++ ) {
        DES_Decrypt_V2( p );
        p += 2;
      }

      /* compare password */
      if( strcmp(header.password,UserKey) != 0 )  {
        IllegalPassword();
```

```
  return;
}

/* get original file name */
strcpy( Fileout, header.name );
status = CheckPoss( Filein. Fileout );
if ( status != 0 )  { return; }


fout = fopen( Fileout,"w+b");
if ( fin == NULL  || fout == NULL ) {
   OpenFileError();
   unlink(Fileout);
   return:
}

finnum = fileno( fin );
Filesize = filelength( finnum );

Memleft = coreleft() / 2 ;
if ( Memleft >= (Filesize + 8) ) {
     Allocsize = Filesize + 8 - (Filesize % 8);
     Fbuffin   = malloc(Allocsize);
     Fbuffout  = malloc(Allocsize);
}
else {
     Allocsize = Memleft - (Memleft % 8);
     Fbuffin   = malloc(Allocsize);
     Fbuffout  = malloc(Allocsize);
```

```
    }

    /* assign buffer to input file  return 0 on success */
    setvbuf( fin, Fbuffin, _IOFBF, Allocsize );
    /* assign buffer to output file */
    setvbuf( fout, Fbuffout, _IOFBF, Allocsize);


    A_Scale(x1, y1, x2, y2);



    time(&start);
    i = 0;
    while ( !feof(fin) ) {
count = fread( work,1,8,fin );
DES_Decrypt_V2( work );
fwrite( work, 1, count. fout);

if ((i % 250) == 0) {
    time(&now);
    ratio =  (  (float) ftell(fin)  / Filesize  ) * 100;
    Writeratio( ratio . now-start );
}
i++;
    }

    /* This four line is extraneous to the program */
    /* if it is deleted chsize and setftime may error */
```

```
fseek( fout, -SL, SEEK_END );
fread( work, 1, 8, fout );
validbyte = ((char *)work)[7];
endsize = ftell( fin ) - 8 + validbyte;


foutnum = fileno( fout );   /* get the output file handle */

/* this is true file size */
endsize = header.size;
/* restore its size */
chsize( foutnum, endsize );

/* restore its date and time */
setftime( foutnum, &header.ftime );

/* write the buffer to disk */
flushall();

time(&now);
Writeratio( 100 , now-start );
freeze(500);

cputs("\r\n\n\n");
WriteOK(x1+23,y2-2,bkcolor,SHADOW);
getch();
WriteOK(x1+24,y2-2,bkcolor,NOSHADOW);
OKsong;
WriteOK(x1+23,y2-2,bkcolor,SHADOW);
```

```
    freeze(150);
    FirstScr();


    free(Fbuffin );
    free(Fbuffout);
    fclose(fin);
    fclose(fout);

#undef x1
#undef y1
#undef x2
#undef y2
#undef bkcolor
}


Do_UDES_Encrypt(char *Filein, char *Fileout)
{
#define x1          10
#define y1           9
#define x2          70
#define y2          14
#define bkcolor     LIGHTBLUE

Long work[2],Filesize,Memleft,Allocsize;
register int count;
```

```
void *Fbuffin,*Fbuffout;
FILE *fin,*fout;

int   finnum;
int   ratio;
unsigned long i;

/* Key */
Char UserKey[41];
Long Key[2];

/* return value from get key */
int retval;

/* use for clock */
long start,now;

/* file block */
struct ffblk ffblk;

/* pointer to header for encrypting header */
long *p;

register int k,l;


    /* find file for initial ffblk */
    findfirst( Filein, &ffblk, FA_ARCH | FA_HIDDEN | FA_RDONLY );
```

```
    /* save old file in attr */
    header.attr = _chmod( Filein, 0 );

    /* change filein attr to archieve for read write */
    _chmod( Filein, 1, FA_ARCH );


    fin  = fopen( Filein ,"r+b");
    fout = fopen( Fileout,"w+b");
    if ( fin == NULL  || fout == NULL ) {
OpenFileError();
unlink(Fileout);
return;
    }


    /* initial preheader file */
    strcpy(phead.projname,PROJNAME);
    phead.method = Method;
    /* write file preheader */
    fwrite( &phead, sizeof(phead), 1, fout );



    /* find file size */
    finnum = fileno( fin );
    Filesize = filelength( finnum );

    Memleft = coreleft() / 2;
```

```
      if ( Memleft >= (Filesize + 8) ) {
Allocsize = Filesize + 8 - (Filesize % 8);
Fbuffin  = malloc(Allocsize);
Fbuffout = malloc(Allocsize);
      }
      else {
Allocsize = Memleft - (Memleft % 8);
Fbuffin  = malloc(Allocsize);
Fbuffout = malloc(Allocsize);
      }

      /* assign buffer to input file  return 0 on success */
      setvbuf( fin, Fbuffin, _IOFBF, Allocsize );
      /* assign buffer to output file */
      setvbuf( fout, Fbuffout, _IOFBF, Allocsize);


      /* Input encrypt key in string format */
      retval = InputEncryptKey( UserKey ,0);
      if ( retval == -1 )  {
free( Fbuffin );
free( Fbuffout);
fclose(fin);
fclose(fout);
unlink(Fileout);
return;
      }

      /* change the key string to two long int */
```

```
ScramKey( Key, UserKey );

/* Find subkey for encryption */
S_Key( Key );

/* Write scale box */
A_Scale(x1, y1, x2, y2);


/* save input file date and time  size and name */
getftime(finnum, &(header.ftime) );
header.size     = Filesize;
strcpy( header.name, ffblk.ff_name );
strcpy( header.password, UserKey );
header.option   = Option;

/* encrypt header */
p = &header;
for( i=0; i < (sizeof(header)-(sizeof(header)%8)+8)/8;
     i++ )
{
    DES_Encrypt_V2( p );
    p += 2;
}
fwrite(&header, (sizeof(header)-(sizeof(header)%8)+8),
       1, fout );

/* init UDES pattern */
for (k=0; k<8 ;k++)  {
```

```
}
i++;
    } while ( count == 8 );

    /* write the buffer to disk */
    flushall();

    /* restore input file attr */
    _chmod( Filein, 1, header.attr );

    time(&now);
    Writeratio( 100 , now-start );
    freeze(500);

    cputs("\r\n\n\n");
    WriteOK(x1+23,y2-2,bkcolor,SHADOW);
    getch();
    WriteOK(x1+24,y2-2,bkcolor,NOSHADOW);
    OKsong;
    WriteOK(x1+23,y2-2,bkcolor,SHADOW);
    freeze(150);
    FirstScr();

    /* restore data */
    for (k=0; k<8 ;k++)  {
      for (l=0; l<64; l++)  {
          Sbuff[k][l] = S[k][l];
      }
    }
```

```
        free( Fbuffin );
        free( Fbuffout );
        fclose(fin);
        fclose(fout);

#undef x1
#undef y1
#undef x2
#undef y2
#undef bkcolor

}


Do_UDES_Decrypt(char *Filein, char *Fileout)
{
#define x1          10
#define y1           9
#define x2          70
#define y2          14
#define bkcolor     LIGHTBLUE

Long work[2],Filesize,Memleft,Allocsize;
register int count;

void *Fbuffin, *Fbuffout;
```

```
FILE *fin,*fout;

int  foutnum;

int  finnum,ratio;

Char UserKey[41];
Long Key[2];

int retval;

unsigned long i;

long start,now;

int validbyte;
Long endsize;

long *p;

/* return status from checkposs */
int status;

register int k,l;

     fin  = fopen( Filein ,"r+b");
     if ( fin == NULL ) {
  OpenFileError();
  return;
```

```
    }

    /* extract file pre header */
    fread( &phead, sizeof(phead), 1, fin );
    if ( strcmp(phead.projname, PROJNAME) != 0 ) {
       status = BadHeader();
      if (status == 0) {
          return;
       }
     }
    if ( phead.method != 1 ) { return; }



    /* Input encrypt key in string format */
    retval = InputEncryptKey( UserKey, 1 );
    if ( retval == -1 )  {
    /* free( Fbuffin );
    free( Fbuffout);  */
    fclose(fin);
    /* fclose(fout);
    unlink(Fileout); */
    return;
}

    /* change the key string to two long int */
    ScramKey( Key, UserKey );

    /* Find subkey for encryption */
```

```
    S_Key( Key );

    /* extract file header */
    fread( &header, sizeof(header)-(sizeof(header)%8)+8, 1, fin );
    p = &header;
    for(i=0; i< (sizeof(header)-(sizeof(header)%8)+8)/8; i++ ) {
       DES_Decrypt_V2( p );
       p += 2;
    }

    /* compare password */
    if( strcmp(header.password,UserKey) != 0 )  {
      IllegalPassword();
      return;
    }

    /* get original file name */
    strcpy( Fileout, header.name );
    status = CheckPoss( Filein, Fileout );
    if ( status != 0 )  { return; }


    /* extract file UDES header */
    fread( &UDES, sizeof(UDES)-(sizeof(UDES)%8)+8, 1, fin );
    p = &UDES;
    for(i=0; i< (sizeof(UDES)-(sizeof(UDES)%8)+8)/8; i++ ) {
DES_Decrypt_V2( p );
p += 2;
    }
```

```
/* initial Sbox for decryption */
/* init UDES pattern */
for (k=0; k<8 ;k++)  {
  for (l=0; l<64; l++)  {
    S[k][l] = UDES.UDES_S[k][l];
   }
}

fout = fopen( Fileout,"w+b");
if ( fin == NULL  || fout == NULL ) {
   OpenFileError();
   unlink(Fileout);
   return;
}

finnum = fileno( fin );
Filesize = filelength( finnum );

Memleft = coreleft() / 2 ;
if ( Memleft >= (Filesize + 8) ) {
   Allocsize = Filesize + 8 - (Filesize % 8);
   Fbuffin   = malloc(Allocsize);
   Fbuffout  = malloc(Allocsize);
}
else {
   Allocsize = Memleft - (Memleft % 8);
   Fbuffin   = malloc(Allocsize);
   Fbuffout  = malloc(Allocsize);
```

```
}

/* assign buffer to input file  return 0 on success */
setvbuf( fin, Fbuffin, _IOFBF, Allocsize );
/* assign buffer to output file */
setvbuf( fout, Fbuffout, _IOFBF, Allocsize);


A_Scale(x1, y1, x2, y2);



time(&start);
i = 0;
while ( !feof(fin) ) {
      count = fread( work,1,8,fin );
      UDES_Decrypt( work );
      fwrite( work, 1, count, fout);

  if ((i % 250) == 0) {
      time(&now);
      ratio =  (  (float) ftell(fin)  / Filesize  ) * 100;
      Writeratio( ratio , now-start );
  }
  i++;
}

/* This four line is extraneous to the program */
/* if it is deleted chsize and setftime may error */
```

```
fseek( fout, -8L, SEEK_END );
fread( work, 1, 8, fout );
validbyte = ((char *)work)[7];
endsize = ftell( fin ) - 8 + validbyte;


foutnum = fileno( fout );   /* get the output file handle */

/* this is true file size */
endsize = header.size;
/* restore its size */
chsize( foutnum, endsize );

/* restore its date and time */
setftime( foutnum, &header.ftime );

/* write the buffer to disk */
flushall();

time(&now);
Writeratio( 100 , now-start );
freeze(500);

cputs("\r\n\n\n");
WriteOK(x1+23,y2-2,bkcolor,SHADOW);
getch();
WriteOK(x1+24,y2-2,bkcolor,NOSHADOW);
OKsong;
WriteOK(x1+23,y2-2,bkcolor,SHADOW);
```

```
        freeze(150);
        FirstScr();

            .

        free(Fbuffin );
        free(Fbuffout);
        fclose(fin);
        fclose(fout);

#undef x1
#undef y1
#undef x2
#undef y2
#undef bkcolor
}


A_Scale(int x1, int y1, int x2, int y2)
{
#define style     SINGLE
#define boxcolor WHITE
#define bkcolor  LIGHTBLUE
#define txtcolor WHITE

        textbox(x1+25,y1+2,x2-25,y2-2,style,boxcolor,
            bkcolor,txtcolor,SHADOW);
        freeze(50);
        textbox(x1+20,y1+2,x2-20,y2-2,style,boxcolor,
```

```
            bkcolor,txtcolor,SHADOW);
        freeze(50);
        textbox(x1+15,y1+1,x2-15,y2-1,style,boxcolor,
            bkcolor,txtcolor,SHADOW);
        freeze(100);
        textbox(x1,y1,x2,y2,style,boxcolor,bkcolor,txtcolor,SHADOW);

        writetext(x1+10,y1+2,"ททททททททททททุทททททททททททททททททททททททททท",
            bkcolor<<4|txtcolor);
        writetext(x1+10,y1+3,"Time used  00:00:00          Complete   0%",
            bkcolor<<4|txtcolor);


#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}

Writeratio(int ratio, long nsec )
{
#define x 10
#define y  9
register int i;


    gotoxy(45,3);
    textattr(LIGHTBLUE<<4|YELLOW);
    cprintf("%3d%%",ratio);
```

```
        gotoxy(20,3);
        cprintf("%02ld:%02ld:%02ld",nsec/3600, nsec/60, nsec%60 );

        writentext(x+10,y+2,' ',YELLOW,(int)(ratio/2.5));


#undef x
#undef y
}

int handler(int errval, int ax, int bp, int si)
{
#define x1    36
#define y1    12
#define x2    71
#define y2    21
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTRED
#define txtcolor    WHITE

int  drive;
char msg[22];


      textbox(x1+15,y1+3,x2-15,y2-3,SINGLE,boxcolor,
           bkcolor,txtcolor,NOSHADOW);
      freeze(50);
```

```
        textbox(x1+12,y1+2,x2-12,y2-2,SINGLE,boxcolor,
              bkcolor,txtcolor,NOSHADOW);
        freeze(50);
        textbox(x1+9,y1,x2-9,y2,SINGLE,boxcolor,
              bkcolor,txtcolor,NOSHADOW);
        freeze(50);

        drive = (ax & 0x00FF);

        textbox(x1,y1,x2,y2,SINGLE,boxcolor,bkcolor,txtcolor,SHADOW);
        writetext(x1+11,y1+2,"Critical Error",bkcolor<<4 | YELLOW);
        writetext(x1+10,y1+4,"Device Error  or",bkcolor<<4 | WHITE);
        sprintf(msg,"Disk Error on Drive %c:",drive +'A');
        writetext(x1+7,y1+5,msg,bkcolor<<4 | WHITE);

        Errorsong;
        WriteOK(x1+12,y2-2,bkcolor,SHADOW);
        getch();
        WriteOK(x1+13,y2-2,bkcolor,NOSHADOW);
        OKsong;
        WriteOK(x1+12,y2-2,bkcolor,SHADOW);
        freeze(150);

        FirstScr();       /* clear box */
        hardretn(-1);     /* return calling program */

#undef x1
#undef y1
#undef x2
```

```
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
}

Option_Menu( int optionpos[] )
{
#define x1    10
#define y1     5
#define x2    70
#define y2    20
#define style       SINGLE
#define boxcolor    WHITE
#define bkcolor     LIGHTBLUE
#define txtcolor    WHITE
#define normmenu    LIGHTGRAY
#define tickcolor   LIGHTRED
#define LAST        4

int xpos,ypos;
/* menuposition */
void *m1;


      /* make explode box */
      textbox(x1+25,y1+6,x2-25,y2-6,SINGLE,boxcolor,
         bkcolor,txtcolor,NOSHADOW);
```

```
freeze(50);
textbox(x1+20,y1+4,x2-20,y2-4,SINGLE,boxcolor,
   bkcolor,txtcolor,NOSHADOW);
freeze(50);
textbox(x1+15,y1+1,x2-15,y2-1,SINGLE,boxcolor,
   bkcolor,txtcolor,NOSHADOW);
freeze(50);

textbox(x1,y1,x2,y2,style,boxcolor,
   bkcolor,txtcolor,SHADOW);
writeattr(x1,y1,scolor<<4|scolor,x2-x1+1);
writetext(x1+23,y1," Option Menu ", (scolor<<4)|RED );

xpos = x1+5;
ypos = y1+2;
/* yposition */
optionpos[0] = ypos+1;
optionpos[1] = ypos+2;
optionpos[2] = ypos+5;
optionpos[3] = ypos+8;
/* ok xy position */
optionpos[4] = x1+23;
optionpos[5] = y2-2;
/* xposition */
optionpos[6] = xpos;
optionpos[7] = xpos+7;

textbox(xpos,ypos,x2-5,ypos+3,style,boxcolor,
    bkcolor,txtcolor,NOSHADOW);
```

```
writetext(xpos+5,ypos," Method ",bkcolor<<4|YELLOW);
writetext(xpos+7,optionpos[0],"DES",bkcolor<<4|txtcolor);
writetext(xpos+7,optionpos[1],"U-DES",bkcolor<<4|txtcolor);
writetext(xpos+20,optionpos[0],
   "Use original DES pattern",bkcolor<<4|txtcolor);
writetext(xpos+20,optionpos[1],
   "User modify DES pattern",bkcolor<<4|txtcolor);

Tshadow(xpos,optionpos[2],normmenu<<4,bkcolor<<4,14);
writetext(xpos,optionpos[2]," DES modify ",normmenu<<4);
writearrow(xpos,optionpos[2],normmenu<<4|normmenu,14);
writetext(xpos+20,optionpos[2],"Modify DES block using by",
    bkcolor<<4|txtcolor);
writetext(xpos+20,ypos+6,"U-DES method",bkcolor<<4|txtcolor);

writetext(xpos+7,optionpos[3],"Verify",bkcolor<<4|txtcolor);
writetext(xpos+20,optionpos[3],
    "Verify writing data to ensure",bkcolor<<4|txtcolor);
writetext(xpos+20,ypos+9,"that the written data is correct",
    bkcolor<<4|txtcolor);

WriteOK(optionpos[4],y2-2,bkcolor,SHADOW);
writeattr(optionpos[4],y2-2,normmenu<<4,14);
writearrow(optionpos[4],y2-2,normmenu<<4|normmenu,14);

/* write method indication */
writetext(optionpos[7]-3,optionpos[Method],"»",
  bkcolor<<4|txtcolor);
```

```
      if( getverify() == 1 ) {
        writetext(optionpos[7]-3,optionpos[3],"ส",
        bkcolor<<4|txtcolor);
      }

      CurOn();
      window(1,1,80,25);
}

Option_Select( int optionpos[] )
{
int cur=0;
void *m1;

      CurOn();
      window(1,1,80,25);
      WriteOption( cur , optionpos );

      for ( ;; ) {

 switch( get_key() )  {
    case KEY_UP    :
     if( cur != 0 ) {
       DeleteOption( cur, optionpos );
       cur--;
       WriteOption( cur, optionpos );
       break;
```

```
  }
  if( cur == 0 ) {
    DeleteOption( cur, optionpos );
    cur = LAST;
    WriteOption( cur, optionpos );
    break;
  }

case KEY_DOWN  :
  if( cur != LAST ) {
     DeleteOption( cur, optionpos );
     cur++;
     WriteOption( cur, optionpos );
     break;
  }
  if( cur == LAST ) {
      DeleteOption( cur, optionpos );
      cur = 0;
      WriteOption( cur, optionpos );
      break;
  }
case KEY_ENTER :
case KEY_SPACE : if ( cur == 0 ) {
       writetext(optionpos[7]-3,optionpos[cur],"ส",
         bkcolor<<4|tickcolor);
       Method = 0;
       writetext(optionpos[7]-3,optionpos[1]," ",
         bkcolor<<4|txtcolor);
       break;
```

```
    }
    if ( cur == 1 ) {
        writetext(optionpos[7]-3,optionpos[cur],"ä",
          bkcolor<<4|tickcolor);
        Method = 1;
        writetext(optionpos[7]-3,optionpos[0]," ",
          bkcolor<<4|txtcolor);
        break;
    }
    if ( cur == 3 ) {
    /* not verify set to verify */
    if( getverify() == 0 ) {
        writetext(optionpos[7]-3,optionpos[cur],"ä",
          bkcolor<<4|tickcolor);
        setverify(1);
        break;
}
/* already verify set to not verify */
    if( getverify() == 1 ) {
        writetext(optionpos[7]-3,optionpos[cur]," ",
          bkcolor<<4|tickcolor);
        setverify(0);
        break;
     }
}

    /* chage DES pattern */
    if( cur == 2 )  {
       m1 = malloc(2*16);
```

```
        gettext(optionpos[6]-1,optionpos[2],optionpos[6]+13,
            optionpos[2],m1);
        Tshadow(optionpos[6],optionpos[2],hibar,
            bkcolor<<4|bkcolor,14);
        puttext(optionpos[6],optionpos[2],optionpos[6]+14,
            optionpos[2],m1);
        OKsong;
        puttext(optionpos[6]-1,optionpos[2],optionpos[6]+13,
            optionpos[2],m1);
        Tshadow(optionpos[6],optionpos[2],hibar,
            bkcolor<<4|BLACK,14);
        free(m1);
        S_modify(0);

      /* write option menu */
      Option_Menu( optionpos );
      /* put bar selection */
      WriteOption( cur, optionpos );
      break;
    }

  if( cur == LAST ) {
  /* WriteOK(optionpos[4],optionpos[5],bkcolor,SHADOW);
      getch();*/
      WriteOK(optionpos[4]+1,optionpos[5],bkcolor,NOSHADOW);
      OKsong;
      WriteOK(optionpos[4],optionpos[5],bkcolor,SHADOW);
      freeze(150);
      CurOff();
```

```
        FirstScr();
        return;
      }
    default         : break;
  }
      } /* end of for loop */

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
#undef normmenu
#undef tickcolor
#undef LAST
}

DeleteOption(int menu , int optionpos[] )
{
#define boxcolor   WHITE
#define bkcolor    LIGHTBLUE
#define txtcolor   WHITE
#define normmenu   LIGHTGRAY
#define tickcolor  LIGHTRED
```

```
    if( ( menu == 0 ) || ( menu == 1 ) || ( menu == 3 ) ) {
      CurOn();

       writeattr(optionpos[7]-3,optionpos[menu],
         bkcolor<<4|txtcolor,12);
       return;
    }
    if ( menu == 2 ) {
        CurOff();
        writeattr(optionpos[6],optionpos[menu],
            normmenu<<4,14);
        writearrow(optionpos[6],optionpos[menu],
            normmenu<<4|normmenu,14);
        return;
    }
    if ( menu == 4 ) {
        CurOff();
        writeattr(optionpos[4],optionpos[5],normmenu<<4,14);
        writearrow(optionpos[4],optionpos[5],
             normmenu<<4|normmenu,14);
        return;
    }

#undef boxcolor
#undef bkcolor
#undef txtcolor
#undef normmenu
#undef tickcolor
```

```
}

WriteOption(int menu, int optionpos[] )
{
#define boxcolor   WHITE
#define bkcolor    LIGHTBLUE
#define txtcolor   WHITE
#define normmenu   LIGHTGRAY
#define tickcolor  LIGHTRED

     if( ( menu == 0 ) || ( menu == 1 ) || ( menu == 3 ) ) {
CurOn();
        gotoxy(optionpos[7]-3,optionpos[menu]);
writeattr(optionpos[7]-3,optionpos[menu],bkcolor<<4|tickcolor,12);
return;
     }
     if ( menu == 2 ) {
CurOff();
writeattr(optionpos[6],optionpos[menu],hibar,14);
return;
     }
     if ( menu == 4 ) {
CurOff();
writeattr(optionpos[4],optionpos[5],hibar,14);
return;
     }

#undef boxcolor
#undef bkcolor
```

```
#undef txtcolor
#undef normmenu
#undef tickcolor
}


main()
{
void *m1;
char Infile[80],Outfile[80];
int menupos[7],
    optionpos[8],
    Status;


     harderr( handler );


     Initialize();          /* init program config and parameter */
     WriteFirstScr();       /* Write first screen */
     Demoname();            /* Demo program name  */


     for ( ;; )  {
       m1 = malloc( (80*2) * (24) );
       gettext(1,2,80,24,m1);          /* getbackground */
       WriteMenu(menupos);             /* Write menu box */
       Func = MenuSelect(menupos);     /* Select menu  */
       puttext(1,2,80,24,m1);          /* putbackground */
       free( m1 );
```

```
      switch( Func ) {
 /* Encryption Function */
 case 0 :     GetEncryptName(Infile,Outfile,Func);
     Status = CheckPoss(Infile,Outfile);
     if (Status == 0)  {
        if (Method == 0)  {
             Do_DES_Encrypt(Infile,Outfile);
        }
        if (Method == 1)  {
             Do_UDES_Encrypt(Infile,Outfile);
        }
     }
 break;

/* Decryption Function */
case 1 :     GetEncryptName(Infile, NULL, Func);
     if (Method == 0)  {
         Do_DES_Decrypt(Infile,Outfile);
     }
     if (Method == 1)  {
         Do_UDES_Decrypt(Infile.Outfile);
     }
 break;

/* Set Option */
case 2 : Option_Menu(optionpos);
          Option_Select(optionpos);
          break;
```

```
   /* Exit to DOS */
   case 3 : ExitFunc(); break;

   default : break;
        }  /* End of switch */
     }  /* End of for */
}
```

Appendix D

File : UDES.C

Function : File for change S box data.

```
#include "conio.h"
#include "stdlib.h"

#include "B:\text\bitdef.c"
#include "B:\text\textbox.h"
#include "B:\text\getkey.h"

extern Char S[8][64];
extern Char DES_S[8][64];
extern Char Sbuff[8][64];

#define scolor  WHITE
#define hibar   scolor<<4¦BLUE
#define OKsong  play("T100O4GAO5D$")
#define Click1  play("T500L32O4EB$")

S_modify(int Snum)
{
#define x1    3
#define y1    3
#define x2   76
#define y2   22
#define style        SINGLE
#define boxcolor     WHITE
#define bkcolor      LIGHTBLUE
#define txtcolor     LIGHTGRAY
#define barcol       LIGHTGRAY
#define select       LIGHTGRAY<<4¦BLACK
#define Sbar         4
```

```
#define bigbarlength 68

int i,j,k;
int xpos,ypos[4];
char msg[20];
int okxpos,nextpos,previouspos;
/* msg is location for display interchange data */
int msgx1,msgx2,msgy;
void *m1,*m2;

/*  status for nextbox  0: next  1: previous 2: Ok */
int stat=0;

/* change status 0 : no input */
/*               1 : one input */
/*               2 : two input  change process done */
int chgstatus = 0;

/* first number of selection to change */
int first;



   /* alloc mem for white bar and nextstatus*/
   m2 = malloc(2*16*2);

   textbox(x1+33,y1+8,x2-33,y2-8,style,boxcolor,bkcolor,txtcolor,SHADOW);
   freeze(50);
   textbox(x1+30,y1+6,x2-30,y2-6,style,boxcolor,bkcolor,txtcolor,SHADOW);
```

```
  freeze(50);
  textbox(x1+20,y1+5,x2-20,y2-5,style,boxcolor,bkcolor,txtcolor,SHADOW);
  freeze(50);

  textbox(x1,y1,x2,y2,style,boxcolor,bkcolor,txtcolor,SHADOW);
  sprintf(msg," S-Box Number %d ",Snum+ 1);
  writetext(x1+30,y1,msg,bkcolor<<4|YELLOW);

  /* xpos  x position for bigbar */
  /* ypos  y position for bigbar */
  xpos = x1+5;
  ypos[0] = y1+5; ypos[1] = y1+8;
  ypos[2] = y1+11; ypos[3] = y1+14;

  /* display interchage data */
  msgx1 = x1+32;  msgx2 = x1+41;
  msgy  = y1+2;

  window(xpos,ypos[0],x2,y2);
  textattr(bkcolor<<4|txtcolor);
  for ( i=0,j=0; i<64; i++,j++ ) {
    cprintf(" %2d ",Sbuff[Snum][i]);
    if ( (j == 15)&&(i<63) ) { j=-1; cprintf("\r\n\n\n"); }
  }
  window(x1+1,y1+1,x2-1,y2-1);

  /* write change message */
  writetext(msgx1-13,msgy,"Interchange        and       ",bkcolor<<4|boxcolor);
```

```
/* write big bar */
Write_S_Bigbar(xpos-2,ypos[0],  select, bigbarlength);


/* write select choice */
Tshadow(xpos,ypos[0],hibar,select,Sbar);

/* write Next */
nextpos = x1+12;
writetext(nextpos,y2-2,"     Next      ",select);
Tshadow(nextpos,y2-2,select,bkcolor<<4¦BLACK,14);
writearrow(nextpos,y2-2,select¦((select)>>4),14);

/* write Previous */
previouspos = x1+30;
writetext(previouspos,y2-2,"   Previous    ",select);
Tshadow(previouspos,y2-2,select,bkcolor<<4¦BLACK,14);
writearrow(previouspos,y2-2,select¦((select)>>4),14);

/* write Ok */
okxpos = x1+48;
WriteOK(okxpos,y2-2,bkcolor,SHADOW);
writeattr(okxpos,y2-2,select,14);
writearrow(okxpos,y2-2,select¦((select)>>4),14);

/* i refer to x position 1-15 for line 1-3 */
/* j refer to y position 1-4  4 is ok line */
/* k is x position of select bar */
for( i=0,j=0,k=xpos;; ) {
```

```
        /* write select arrow */
writearrow(k,ypos[j],hibar,Sbar);

     switch( get_key() )   {

case  KEY_LEFT  : if ( (j != 4) ) {
     /* delete old one */
     Tshadow(k,ypos[j],select,select|((select)>>4),Sbar);
     writetext(k+Sbar,ypos[j]," ",select);
     writentext(k,ypos[j]+1,' ',select,Sbar+1);
     /* delete arrow */
     writetext(k,ypos[j]," ",select);
     writetext(k+Sbar-1,ypos[j]," ",select);
     if (i==0) {
i = 16;
k = xpos+(16*Sbar);
     }
     /* select new one */
     k -= Sbar;
     i--;
     Tshadow(k,ypos[j],hibar,select,Sbar);
     /* write select arrow */
     writearrow(k,ypos[j],hibar,Sbar);
     break;
   }
   if ( j == 4 ) {
     if (stat != 0) {
 /* delete old */
 writeattr(nextpos+(stat*18),y2-2,select,14);
```

```
                                writearrow(nextpos+(stat*18),y2-2,select¦((select)>
 /* write new */
 stat--;
 writeattr(nextpos+(stat*18),y2-2,hibar,14);
 break;
      }
      if (stat == 0) {
 /* delete old */
 writeattr(nextpos+(stat*18),y2-2,select,14);
                                writearrow(nextpos+(stat*18),y2-2,select¦((select)>
 /* write new */
                                stat = 2;
 writeattr(nextpos+(stat*18),y2-2,hibar,14);
 break;
      }
   }

   break;
 case  KEY_RIGHT : if ( (j != 4) ) {
      /* delete old one */
      Tshadow(k,ypos[j],select,select¦((select)>>4),Sbar);
      writetext(k+Sbar,ypos[j]," ",select);
      writentext(k,ypos[j]+1,' ',select,Sbar+1);
                              /* delete arrow */
      writetext(k,ypos[j]," ",select);
      writetext(k+Sbar-1,ypos[j]," ",select);
      if (i==15) {
i = -1;
k = xpos-Sbar;
```

```
    }
    /* select new one */
    k += Sbar;
    i++;
    Tshadow(k,ypos[j],hibar,select,Sbar);
                              /* write select arrow */
    writearrow(k,ypos[j],hibar,Sbar);
    break;
  }
                          if ( j == 4 ) {
    if (stat == 2) {
/* delete old */
writeattr(nextpos+(stat*18),y2-2,select,14);
                                writearrow(nextpos+(stat*18),y2-2,select|((select)>
/* write new */
stat = 0;
writeattr(nextpos+(stat*18),y2-2,hibar,14);
break;
    }
    if (stat != 2) {
/* delete old */
writeattr(nextpos+(stat*18),y2-2,select,14);
                                writearrow(nextpos+(stat*18),y2-2,select|((select)>
/* write new */
stat++;
writeattr(nextpos+(stat*18),y2-2,hibar,14);
break;
    }
  }
```

```
  break;

case  KEY_UP    : if (chgstatus != 0) { break; }
  if (j == 4) {
     /* delete Ok bar */
                              writeattr(nextpos+(stat*18),y2-2,select,14 );
                              writearrow(nextpos+(stat*18),y2-2,select¦((select)>>4)
                              /* write new big bar */
     j--;
     Write_S_Bigbar(xpos-2,ypos[j],select,bigbarlength);
     /* write select bar */
     Tshadow(k,ypos[j],hibar,select,Sbar);
                              /* write select arrow */
     writearrow(k,ypos[j],hibar,Sbar);
     break;
  }

  if (j != 4) {
                              /* delete select bar */
     Tshadow(k,ypos[j],select,select¦((select)>>4),Sbar);
     writetext(k+Sbar,ypos[j]," ",select);
     writentext(k,ypos[j]+1,' ',select,Sbar+1);
                              /* delete arrow */
     writetext(k,ypos[j]," ",select);
     writetext(k+Sbar-1,ypos[j]," ",select);
     /* delete old big bar */
     Write_S_Bigbar(xpos-2,ypos[j],bkcolor<<4¦txtcolor,bigbarlength);
     /* if it is the top */
```

```
      if ( j == 0 )  {
 /* write bar at OK */
/* WriteOK(okxpos,y2-2,bkcolor,SHADOW); */
 writeattr(nextpos+(stat*18),y2-2,hibar,14);
 j = 4;
      }
      else {
      /* write new big bar */
      j--;
      Write_S_Bigbar(xpos-2,ypos[j],select,bigbarlength);
      /* write select bar */
      Tshadow(k,ypos[j],hibar,select,Sbar);
                              /* write select arrow */
      writearrow(k,ypos[j],hibar,Sbar);
      }

   }
   break;
 case  KEY_DOWN  : if (chgstatus != 0) { break; }
   if (j==4)  {
      /* delete Ok bar */
      writeattr(nextpos+(stat*18),y2-2,select,14 );
      writearrow(nextpos+(stat*18),y2-2,select|((select)>>4),14);
                              /* write new big bar */
      j=0;
      Write_S_Bigbar(xpos-2,ypos[j],select,bigbarlength);
      /* write select bar */
      Tshadow(k,ypos[j],hibar,select,Sbar);
      break;
```

```
  }
  if (j!=4)  {
                              /* delete select bar */
     Tshadow(k,ypos[j],select,select|((select)>>4),Sbar);
     writetext(k+Sbar,ypos[j]," ",select);
     writentext(k,ypos[j]+1,' ',select,Sbar+1);
                              /* delete arrow */
     writetext(k,ypos[j]," ",select);
     writetext(k+Sbar-1,ypos[j]," ",select);
     /* delete old big bar */
     Write_S_Bigbar(xpos-2,ypos[j],bkcolor<<4|txtcolor,bigbarlength);
     /* if it is the top */
     if ( j == 3 )  {
/* write bar at OK */
/* WriteOK(okxpos,y2-2,bkcolor,SHADOW); */
                              writeattr(nextpos+(stat*18),y2-2,hibar,14);
j = 4;
     }
     else {
     /* write new big bar */
     j++:
     Write_S_Bigbar(xpos-2,ypos[j],select,bigbarlength);
     /* write select bar */
     Tshadow(k,ypos[j],hibar,select,Sbar);
                              /* write select arrow */
     writearrow(k,ypos[j],hibar,Sbar);
     }

  }
```

```
  break;
 case  KEY_SPACE :
 case  KEY_ENTER : if( j!=4 ) {
     /* hide shadow */
     Tshadow(k,ypos[j],hibar,barcol<<4|barcol,Sbar);
     /* move 1 char cell */
     gettext(k-1,ypos[j],k+Sbar-1,ypos[j],m2);
     puttext(k,ypos[j],k+Sbar,ypos[j],m2);
                              Click1;
     /* replace */
     Tshadow(k,ypos[j],hibar,select,Sbar);
     puttext(k-1,ypos[j],k+Sbar-1,ypos[j],m2);

     chgstatus++;

                              /* put to change message */
     if ( chgstatus == 1 ) {
puttext(msgx1,msgy,msgx1+Sbar,msgy,m2);
writeattr(msgx1,msgy,bkcolor<<4|YELLOW,5);
                              /* delete arrow */
writetext(msgx1+1,msgy," ",bkcolor<<4|YELLOW):
writetext(msgx1+Sbar,msgy," ",bkcolor<<4|YELLOW);
first = i;
     }
     if ( chgstatus == 2) {
chgstatus = 0;
puttext(msgx2,msgy.msgx2+Sbar,msgy,m2);
writeattr(msgx2,msgy,bkcolor<<4|YELLOW,5):
                              /* delete arrow */
```

```
writetext(msgx2+1,msgy," ",bkcolor<<4¦YELLOW);
writetext(msgx2+Sbar,msgy," ",bkcolor<<4¦YELLOW);
/* swab data */
S_swab( &Sbuff[Snum][first+(j*16)], &Sbuff[Snum][i+(j*16)] );
/* display change */
/* put second data to first */
puttext(xpos+(first*Sbar)-1,ypos[j],xpos+(first*Sbar)+Sbar-1,ypos[j],m2);
writeattr(xpos+(first*Sbar),ypos[j],select,Sbar+1);
/* delete arrow of the second data to be first */
writetext(xpos+(first*Sbar),ypos[j]," ",select);
writetext(xpos+(first*Sbar)+Sbar-1,ypos[j]," ",select);
/* put first data to second */
gettext(msgx1+1,msgy,msgx1+Sbar,y1+2,m2);
puttext(k,ypos[j],k+Sbar-1,ypos[j],m2);
Tshadow(k,ypos[j],hibar,select,Sbar);

/* clear select number msg */
freeze(300);
writetext(msgx1,msgy,"     ",bkcolor<<4¦YELLOW);
writetext(msgx2,msgy,"     ",bkcolor<<4¦YELLOW);
      }
     break;
   }

   if( j==4 ) {
      if (stat == 2) {
 WriteOK(okxpos+1.y2-2,bkcolor,NOSHADOW);
 OKsong;
 WriteOK(okxpos,y2-2.bkcolor,SHADOW);
```

```
freeze(150);


free(m2);
FirstScr();
return;
     }
     if (stat != 2) {
Tshadow(nextpos+(stat*18),y2-2,hibar,bkcolor<<4|bkcolor,14);
gettext(nextpos+(stat*18)-1,y2-2,nextpos+(stat*18)+16,y2-2,m2);
puttext(nextpos+(stat*18) ,y2-2,nextpos+(stat*18)+15,y2-2,m2);
OKsong;
freeze(100);
puttext(nextpos+(stat*18)-1,y2-2,nextpos+(stat*18)+16,y2-2,m2);
Tshadow(nextpos+(stat*18),y2-2,hibar,bkcolor<<4|BLACK,14);
freeze(100);


free( m2 );


/*  next sbox replacement */
if (stat == 0) {
    if (Snum != 7) {
      Snum++;
    }
    else {
      Snum = 0;
    }
    FirstScr();
    S_modify( Snum );
    return;
```

```
  }
  /* previous box */
  if (stat == 1) {
      if (Snum != 0) {
        Snum--;
      }
      else {
        Snum = 7;
      }
      FirstScr();
      S_modify( Snum );
      return;
  }
       } /* end if stat != 2 */

    } /* end if j == 4 */
 default :  break;
       }
    }

#undef x1
#undef y1
#undef x2
#undef y2
#undef style
#undef boxcolor
#undef bkcolor
#undef txtcolor
#undef barcol
```

```
#undef select
#undef Sbar
#undef bigbarlength
}

Write_S_Bigbar( int x, int y, int attr, int size )
{
   writeattr(x,y-1,attr,size);
   writeattr(x,y,  attr,size);
   writeattr(x,y+1,attr,size);
}

S_swab( char *a, char *b)
{
int tmp;

     tmp = *a;
     *a  = *b;
     *b  = tmp;
}
```

Appendix E

File : BITDEF.C

Function : File contains bit values for using in DES algorithm.

```
/* BitDef.c                                    */
/* define the bit value for use in program */

typedef  unsigned long    Long;
typedef  unsigned int     Int ;
typedef  unsigned char    Char;

/* B*L  -> define in long int 32 bits      */
#define B1L   0x80000000
#define B2L   0x40000000
#define B3L   0x20000000
#define B4L   0x10000000
#define B5L   0x08000000
#define B6L   0x04000000
#define B7L   0x02000000
#define B8L   0x01000000
#define B9L   0x00800000
#define B10L  0x00400000
#define B11L  0x00200000
#define B12L  0x00100000
#define B13L  0x00080000
#define B14L  0x00040000
#define B15L  0x00020000
#define B16L  0x00010000
#define B17L  0x00008000
#define B18L  0x00004000
#define B19L  0x00002000
#define B20L  0x00001000
#define B21L  0x00000800
```

```
#define B22L   0x00000400
#define B23L   0x00000200
#define B24L   0x00000100
#define B25L   0x00000080
#define B26L   0x00000040
#define B27L   0x00000020
#define B28L   0x00000010
#define B29L   0x00000008
#define B30L   0x00000004
#define B31L   0x00000002
#define B32L   0x00000001

#define B1C    0x80
#define B2C    0x40
#define B3C    0x20
#define B4C    0x10
#define B5C    0x08
#define B6C    0x04
#define B7C    0x02
#define B8C    0x01
```

Appendix F

File : Textbox.c

Function : drawing box in text mode.

```
/*-------------------------------------------------------*/
/* TEXTBOX.C : Routine to draw boxes in text mode        */
/* The style argument determines the type of box drawn:  */
/*                    0 = nobox          */
/*      1 = single-scored   */
/*      2 = double-scored   */
/*                                                       */
/*             boxcolor = color of box lines             */
/*              bkcolor = color of background in the box */
/*            textcolor = color of text in box           */
/*                                                       */
/* The shadow argument determines the shdow of box :     */
/*                    1 = active shadow                  */
/*                others = nonactive shadow              */
/*                                                       */
/* Written by Ton, Last update  Friday January 1, 1991   */
/*-------------------------------------------------------*/


/* this is difined in textbox.h */
/* #define  NOBOX 0
   #define  SINGLE 1
   #define  DOUBLE 2
   #define  SHADOW 1
   #define  NOSHADOW  0
*/



#include <conio.h>
```

```
#include <dos.h>

#define  SHADOWATTR 7

void CurSize(int start,int end);

void textbox (int left, int top, int right, int bottom, int style,
      int boxcolor, int bkcolor, int txtcolor,int shadow )

{
register int i,X, Y, segment;

static bord [][6] = {
{ 196, 179, 218, 191, 217, 192 },
{ 205, 186, 201, 187, 188, 200 }
      };
  if (style == 0 || style > 2) return;
  --style;

  /*************** Initialize parameters for draw box ***************/
  textattr (bkcolor << 4 | boxcolor);

  /*************** Check if getcrtmode==7 monocrome mode *************/
  if ( getcrtmode() != 7 )  segment = 0xb800;  else segment = 0xb000;

  /********************* Set bkcolor in box ***************************/
  window (left,top,right,bottom);
  clrscr ();
```

```
  window (1,1,80,25);

    /********************** Draw Horizontal line ******************/
    /* Draw Horizontal Top line */
    for ( i = (top-1)*160 + (left*2);
  i < (top-1)*160 + (right-1)*2 ;i+=2)
{  pokeb(segment,i,bord[style][0]); }

    /* Draw Horizontal Bottom line */
    for ( i = (bottom-1)*160 + (left*2);
  i < (bottom-1)*160 + (right-1)*2 ;i+=2)
{  pokeb(segment,i,bord[style][0]); }


    /********************** Draw Vertical lines *********************/
    /* Draw Vertical Left line */
    for ( i = (left*2)-2 + top*160;
  i < (left*2)-2 + (bottom-1)*160 ; i+=160)
{ pokeb(segment,i,bord[style][1]); }

    /* Draw Vertical Right line */
    for ( i = (right*2)-2 + top*160;
  i < (right*2)-2 + (bottom-1)*160 ; i+=160)
{ pokeb(segment,i,bord[style][1]); }

    /************************ Set Corners ***************************/
    /* Top left corner */
    pokeb(segment,(left*2)-2 + (top-1)*160, bord[style][2]);
    /* Bottom left corner */
```

```
    pokeb(segment,(left*2)-2 + (bottom-1)*160, bord[style][5]);
    /* Top right corner */
    pokeb(segment,(right*2)-2 + (top-1)*160, bord[style][3]);
    /* Bottom right corner */
    pokeb(segment,(right*2)-2 + (bottom-1)*160, bord[style][4]);


   /********** Set textcolor in box ************/
    textcolor(txtcolor);

  /********************** Draw the shadow ****************************/
  if (shadow != 0) {

      /* Bottom Shadow */
      for(Y=left+2,X=1;X<=abs(right-left);X++,Y++) {
  pokeb(segment,(bottom*160)+(Y*2)+1,SHADOWATTR);
      }
      /* Right Shadow */
      for(Y=top+1,X=1;X<=abs(bottom-top);X++,Y++) {
  pokeb(segment,((Y-1)*160)+((right)*2)+1,SHADOWATTR);
  pokeb(segment,((Y-1)*160)+((right+1)*2)+1,SHADOWATTR);
      }
    }
    window(left+2,top+1,right-2,bottom-1);
}

int getcrtmode()
{
    union REGS reg;
```

```
    reg.h.ah = 15;
    int86(0x10,&reg,&reg);
    return((int)reg.h.al);
}

void writeattr(int x, int y, int attr, int nelem)
{
int segment;
register int i;
     if ( getcrtmode() != 7 )  segment = 0xb800;  else segment = 0xb000;
     --y; --x;
     for (i=0; i<nelem; i++,x++) {
  pokeb(segment,(y*160)+(x*2)+1,attr);
     }
}

void writentext(int x,int y, char txt, int attr, int nelem)
{
int segment;
register int i;
     if ( getcrtmode() != 7 )  segment = 0xb800;  else segment = 0xb000;
     --y; --x;
     for (i=0; i<nelem; i++,x++) {
  pokeb(segment,(y*160)+(x*2),  txt );
          pokeb(segment,(y*160)+(x*2)+1,attr);
     }
}

void writetext(int x, int y, char *text, int attr )
```

```
{
int segment;
register int i;
     if ( getcrtmode() != 7 )  segment = 0xb800;  else segment = 0xb000;
     --y; --x;
     for (i=0; *(text+i) != '\0'; i++,x++) {
  pokeb(segment,(y*160)+(x*2),*(text+i));
          pokeb(segment,(y*160)+(x*2)+1,attr);
     }
}

/*   enable foreground blinking or enable background intensity */
/*   BL = 0 enable background intensity */
/*   BL = 1 enable foreground blink */
void EnableBlink( int enable )
{
     _AH = 0x10;          /* function 10h */
     _AL = 3;             /* subfunction 3h */
     _BL = enable;
     geninterrupt(0x10);  /* call interrupt 10h */
}

/*****************************************************/
/*                  Cursor Header File               */
/*   CurOff   ->  No Cursor display                  */
/*   CurOn    ->  Restore Cursor   (Normal Size only) */
/*   CurSize(start,end) -> Set Cursor size           */
/*****************************************************/
```

```
void CurOff(void)
{
   union REGS reg;
   reg.h.ah = 1;           /* Interupt 10h Function 1 Set Cursor size */
   reg.h.ch = -1;          /* Start cursor */
   reg.h.cl = -1;          /* End cursor  (color=7,mono=13); */
   int86(0x10,&reg,&reg); /* Call Interupt */
}

void CurOn(void)
{
int start,end;
   if (getcrtmode()==7) {
CurSize(11,12); }
     else
CurSize(6,7);
}

void CurSize(int start,int end)
{
   union REGS reg;
   reg.h.ah = 1;           /* Interupt 10h Function 1 Set Cursor size */
   reg.h.ch = start;       /* Start cursor */
   reg.h.cl = end;         /* End cursor  (color=7,mono=13); */
   int86(0x10,&reg,&reg); /* Call Interupt */
}
```

File : Textbox.h

Function : textbox header file.

```
#define  NOBOX 0
#define  SINGLE 1
#define  DOUBLE 2
#define  SHADOW 1
#define  NOSHADOW  0


extern void textbox (int left, int top, int right, int bottom, int style,
      int boxcolor, int bkcolor, int txtcolor,int shadow );

extern int getcrtmode(void);

extern void writeattr(int x, int y, int attr, int nelem);
extern void writentext(int x, int y, char txt, int attr, int nelem);
extern void writetext(int x, int y, char *text, int attr );

extern void EnableBlink( int enable);

extern void CurOff(void);
extern void CurOn(void);
extern void CurSize(int start,int end);
```

Appendix G

File : Screen.c

Function : File for drawing background display of program.

```
#define  ColorRam  0xb800          /* Color Card videoram address */
#define  MonoRam   0xb000          /* Monocrome  videoram address */
#define  MENUATTR   48             /* Cyan back + Black text */
#define  BACKATTR   (1<<4) + 15           /* Blue back + Lightgrey text */


#include <conio.h>
#include <dos.h>


static unsigned char oldcolor[16]  = {
0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63 };

static unsigned char newvalue[16] = {
0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63 };

void BlankScr(void)
{
   textbackground(BLACK);
   clrscr();
}


void FirstScr(void)
{
int i;
unsigned int Vdo;

   if ( getcrtmode() != 7)  Vdo = ColorRam;  /* Initial VideoRam */
```

```
        else  Vdo = MonoRam;

    for ( i=1 ; i<=0x9f ; i+=2 ) {          /* Write menu background */
        poke(Vdo,i,MENUATTR);               /* Cyan + Black text */
        }
    for (     ;i<=0xeff ; i+=2 ) {          /* Write program background */
        pokeb(Vdo,i,BACKATTR);             /* Color Background BLUE */
        pokeb(Vdo,i-1,176);                 /* Text  ฑ  light gray */
        }
    for (     ;i<=0xf9f ; i+=2 ) {          /* Write Bottom Background */
        pokeb(Vdo,i,63);                    /* Same Color as menu */
        }
}  /* End FirstScr Function */

void Changecolor(char oldcolor,char newvalue)
{
#define Palread  0x03C1  /* Palette register read port */
#define Palwrite 0x03C0

     inportb(0x3DA);
     outportb(Palwrite,oldcolor);
     outportb(Palwrite.newvalue);
}

void EGA_ON()
{
     _AX = 0x1000;
     _BX = 0x0f12;
     geninterrupt(0x10);
```

```
}

void Initcolor()
{
int i;
     /* Chage color for using more interesting program */
     /* It caused the screen off */
     for( i=0; i<16; i++)  {
  Changecolor( i, newvalue[i] );
     }

     EGA_ON();     /* Turn display screen on */
}
```

Appendix H

File : Getkey.c

Function : Recieve key data.

```
#include <conio.h>

int get_key(void)
{
    int ch1,ch2;
    ch1=getch();
    if (ch1 == 0 )   {
      ch2 = getch();
      return(ch2<<8 ¦ ch1);
      }
    else
      return((int) ch1);
}
```

File : Getkey.h

Function : Getkey header file.

```
#define KEY_ESC         0x1B
#define KEY_UP          0x4800
#define KEY_DOWN        0x5000
#define KEY_LEFT        0x4B00
#define KEY_RIGHT       0x4D00
#define KEY_ENTER       0x0D
#define KEY_SPACE       0x20

extern int get_key(void);
```

Appendix I

File : Play.c

Function : Playing sound.

```
/*****************************************************************/
/*              Play.h     File for play song in program         */
/*                                                               */
/*        Notes >   C,D,E,F,G,A,B                                */
/* Special sign >  #,+ (Sharp); - (Flat)                         */
/*          L n >   n is note type defaults is 4 (Black note)    */
/*                  n = 1 is white note with no tial             */
/*          MS  >   Music Staccato  play 3/4 of notetime and     */
/*stop 1/4 of notetime     */
/*          MN  > Music Normal play 7/8 and stop 1/8             */
/*          ML  > Music Legato  play all notetime                */
/*          O n >  n is number of octave (0 to 7) ,              */
/*            default is 3 (Middle C)                    */
/*          S n > Scale for change sound to higher tone (0 to 7) */
/*          P n > Pause time n is like in L n                    */
/*          T n > Tempo number of blacknote in 1 minute          */
/*          .   > (Dot) Time will be 3/2 of normal time          */
/*                                                               */
/*        Written by Ton , Last Update Thursday 13,June,1991     */
/*****************************************************************/

#include <stdlib.h>
#include <time.h>

#define BaseOctave    3
#define BaseTempo   120
#define SharpIndex   60
#define RealTime  60000
#define DotTime   90000
```

```
void freeze(int freezetime);
void play(char *Song);
int  GetNum(char *Sng, int *Position);

static  int Pitch[120] = { 0,   32,   37,   43,   45,   51,   56,   61,
64,   75,   86,   91,  102,  112,  123,
       128,  149,  171,  182,  203,  224,  246,
       256,  299,  342,  363,  406,  449,  492,
       512,  598,  684,  726,  812,  898,  984,
      1024, 1198, 1368, 1452, 1624, 1796, 1968,
      2048, 2396, 2736, 2904, 3248, 3592, 3936,
      4096, 4792, 5472, 5808, 6496, 7184, 7872,
      0,0,0,
35,   40,   45,   48,   53,   59,   64,
69,   80,   91,   96,  107,  117,  128,
       138,  160,  181,  192,  213,  235,  256,
       277,  320,  363,  384,  427,  470,  512,
       554,  640,  726,  768,  854,  940, 1024,
      1108, 1280, 1452, 1536, 1708, 1880, 2048,
      2216, 2560, 2904, 3072, 3416, 3760, 4096,
      4432, 5120, 5808, 6144, 6832, 7520, 8192,
      0,0,0,0 };


void play(char *Song)
{
int    NoteType=4, Scale=0, Tempo=BaseTempo,Position=0,PlayFrac=7,
       NoteTime, PlayTime, IdleTime, Octave=BaseOctave;
```

```
unsigned long int BaseTime=RealTime;

char  Character, PitchIndex;

    do  {  Character = toupper( Song[Position] );
   switch ( Character )   {              /* Current Position */
      case 'A' :    case 'B' :
      case 'C' :    case 'D' :
      case 'E' :    case 'F' :
      case 'G' :  PitchIndex = ( Character-66 ) + Octave*7 + Scale;
  if ( Character=='A' || Character=='B' ) {
     PitchIndex += 7; }
  Position++;
  switch ( Song[Position] ) { /* Next Position */
     case '#' :
     case '+' :
     case '-' :  PitchIndex += SharpIndex;
 if ( Song[Position] == '-' ) {
    PitchIndex--; }
 Position++; break;
     case '.' :  BaseTime = DotTime;
 Position++;
 break;
     }                      /* End NextPosition */
   NoteTime = (int)((BaseTime/Tempo)*(4.0/NoteType));
   PlayTime = (int)( NoteTime*(PlayFrac/8.0) );
   sound( Pitch[ PitchIndex ] );
   freeze( PlayTime );
   nosound();
```

```
    freeze( NoteTime-PlayTime );
    BaseTime = RealTime;  /*Correct the time*/
    break;
    /*** End Case 'A'..'G' ***/
       case 'L' :  NoteType = GetNum( Song,&Position ); break;
       case 'M' :  Position++;
   switch ( Song[Position] ) {
      case 'S' :  PlayFrac = 6; break;
      case 'N' :  PlayFrac = 7; break;
      case 'L' :  PlayFrac = 8; break;
      default  :  break;
   }
   Position++; break;
       case 'O' :  Octave = GetNum( Song,&Position ); break;

       case 'P' :  IdleTime = GetNum( Song,&Position );
   IdleTime = (int)((BaseTime/Tempo)*(4.0/IdleTime));
   freeze(IdleTime);
   break;
       case 'S' :  Scale = GetNum( Song,&Position ); break;
       case 'T' :  Tempo = GetNum( Song,&Position ); break;
       default  :  Position++:break;
       }  /****  End of Switch Character ****/
 nosound();
 } while ( Character != 'S' );  /*****  End of Do-While *****/
} /****** End Play Function ******/

void freeze(int freezetime)
/***  Function delay for an interval ***/
```

```
/***  freezetime is millisecond form ***/
{
clock_t start,end;
     start = clock();
     for ( ; ; ) {
end = clock();
if ( (end-start)*55 >= freezetime )  break; }
}

int GetNum(char *Sng, int *Position)
{
char Length,index, result[3]="";
     *Position += 1;
     index = *Position;
     for ( Length=0; Sng[index+Length]-'0'>= 0 &&
     Sng[index+Length]-'9'<= 0; Length++) {}
     strncpy( result,&Sng[index],Length );
     *Position += Length;
     return ( atoi( result ) );
}
```