เทคนิคการเข้าและถอดรหัสเพื่อใช้ในด้านการทหาร

DATA ENCRYPTION ALGORITHM FOR MILITARY PURPOSES

ร้อยเอก ชัยยง ไชยสิงห์ทอง
CAPT. CHAIYONG CHAISINGTHONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต
สาขาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2536

ISBN 974-6270-69-6

DATA ENCRYPTION ALGORITHM FOR MILITARY PURPOSES

CAPT. CHAIYONG CHAISINGTHONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE MASTER OF SCIENCE
IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
GRADUATE SCHOOL
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
1993

บทคัดย่อภาษาไทย

หัวข้อวิทยานิพนธ์	เทคนิคการเข้าและถอดรหัสเพื่อใช้ในด้านการทหาร
นักศึกษา	ร้อยเอก ชัยยง ไชยสิงห์ทอง
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร. บุญวัฒน์ อัตชู
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ภาควิชา	สำนักวิจัยและบริการคอมพิวเตอร์ ภาควิชา คณิตศาสตร์ และ วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2536

บทคัดย่อ

งานวิจัยนี้นำเสนอเทคนิคการเข้าและถอดรหัส (Data Encryption) เพื่อให้มีประสิทธิภาพมากขึ้นกว่าแบบ Data Encryption Standard (DES) เทคนิคใหม่จะใช้หลักการของระบบมาตรฐาน DES ซึ่งเป็นวิธีที่สลับซับซ้อนที่มีการผสมและรวมการของวิธีการพื้นฐานเข้ารหัสลับ 2 วิธีคือการสลับสับเปลี่ยน(Permutation หรือ Tranposition) และการแทนค่า (Substitution) เข้าด้วยกัน DES ยังสลับข้อมูลตัวเองด้วยวิธีการทั้ง 2 ไปมาถึง 16 รอบ วิธีการใหม่จะเปลี่ยนตารางการสลับสับเปลี่ยนและตารางการแทนค่าให้ผิดไปจากเดิม เพิ่มจำนวนรอบสลับไปมาเป็น 18 รอบ เพิ่มขนาดกุญแจให้ยาวกว่าเดิมจาก 56 เป็น 112 บิท ซึ่งทำให้การลักลอบถอดรหัสทำได้ยากขึ้นและเทคนิคใหม่เป็นเอกลักษณ์

บทคัดย่อภาษาอังกฤษ

Thesis Title	Data Encryption Algorithm For Military Purposes.
Student	Capt. Chaiyong Chaisingthong
Thesis Advisor	Asst.Prof.Dr.Boonwat Attachoo
Level of Study	Master of Science in Computer Science and Information Technology
Department	Computer Research and Service Center Mathematic and Computer Science Department Faculty of Science King Mongkut's Institute of Technology Ladkrabang
Year	1993

ABSTRACT

This thesis proposes a new data encryption algorithm which bases on Data Encryption Standard(DES). The DES algorithm is a careful and complex combination of two of the fundamental building blocks of encryption: substitution and permutation(transposition). The DES algorithm derives its strength from repeated application of these two techniques, one on top the other, for a total of 16 cycles. With the new algorithm, all the permutation and substitution tables are changed. The repeated application is increased to 18 cycles. By improving the key length from 56 to 112 bits makes it longer to break code. The new data encryption algorithm is unique.

กิตติกรรมประกาศ

ขอขอบคุณ พลตรี มนตรี ศุภาพร อดีตเจ้ากรมการสนเทศทหารกองบัญชาการทหารสูงสุด พลตรี ทวีศักดิ์ เรืองพงษ์ เจ้ากรมการสนเทศทหาร และ พ.อ. ประสงค์ ปานเจริญ รองผู้อำนวยการสถาบันคอมพิวเตอร์ทหาร ผู้ก่อตั้งโครงการร่วมมือการศึกษาระดับปริญญาโทระหว่างกองบัญชาการทหารสูงสุด กับสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งทำให้ข้าราชการทหารมีโอกาสได้ศึกษาหาความรู้เพิ่มเติมในระดับปริญญาโท และ อำนวยความสะดวกในหลาย ๆ ด้าน ผู้วิจัยและนักศึกษาระดับปริญญาโทสายทหารทุกท่านรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีก็เพราะ ได้รับความเมตตาจากผู้ช่วยศาสตราจารย์ ดร.บุญวัฒน์ อัตชู ที่ได้ให้ความกรุณาแนะนำแก่ผู้วิจัยตลอดมา ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ พ.อ.หญิง ชมพูนุท จูฑะพุทธิ ผู้อำนวยการกองปฏิบัติการ กรมการสนเทศทหาร กองบัญชาการทหารสูงสุด พ.อ.หญิง มาลี มาลีงาม หัวหน้าแผนกเตรียมข้อมูล กองปฏิบัติการ และ น.ท.หญิง นพพร สงวนทรัพย์ ที่ให้การสนับสนุนเวลา และข้อเสนอแนะมาโดยตลอด

ขอขอบคุณ นางพรรษมน ไชยสิงห์ทอง (ภรรยา) ที่ดูแลสมาชิกน้อย ๆ ในครอบครัวทั้ง 2 คน เป็นอย่างดีจนทำให้ผู้วิจัยมีเวลาในการทำค้นคว้าอย่างเต็มที่ และบรรลุวัตถุประสงค์ทุกประการ

ร้อยเอก ชัยยง ไชยสิงห์ทอง

สารบัญ

คำนำ หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

คำนิยามศัพท์

1. การเข้าและถอดรหัสแบบมาตราฐาน Data Encryption Standard(DES) หมายถึง การเข้าและถอดรหัสที่ยอมรับว่าเป็นมาตราฐานสากล
2. การเข้าและถอดรหัส Data Encryption หมายถึงเทคนิคที่ใช้ในการเข้าและถอดรหัส
3. การสลับสับเปลี่ยนหรือการสลับสับเปลี่ยนตำแหน่ง Permutation หรือ Transposition หมายถึงการสลับสับเปลี่ยนตำแหน่งต่างๆ ตามตารางการสลับสับเปลี่ยน
4. การสลับสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1} Inverse Initial Permutation หมายถึงการสลับสับเปลี่ยนตำแหน่งต่างๆ ตามตำแหน่งที่กำหนดในตาราง IP^{-1} ซึ่งเป็นการทำขบวนการย้อนกลับกับ IP
5. การแทนที่ หรือการแทนค่า Substitution หรือ Function_F หมายถึงขบวนการแทนค่าที่เป็นส่วนหนึ่งของ DES ซึ่งใช้ร่วมกับหมายเลขกุญแจ
6. การคำนวณหมายเลขกุญแจ Subkey หมายถึง ขบวนการกระจายกุญแจออกเป็นหลาย ๆ ชุด ชุดละ 48 บิท
7. กุญแจคือหมายเลขที่เป็น HEX code กำหนดขึ้นเพื่อใช้ในการเข้าหรือถอดรหัส
8. การทำขบวนการย้อนกลับหรือการถอดรหัส Decryphering หมายถึง การทำการย้อนกลับกับการเข้ารหัส
9. การเข้ารหัสแบบสายธาร Character Stream หมายถึง การเข้าและถอดรหัสแบบง่าย
10. ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ Function_F และ S_box หมายถึง เทคนิคการแทนค่าโดยผนวกกับรหัสกุญแจ

บทที่ 1
บทนำ

1.บทนำ

เนื่องจากการพัฒนาด้านคอมพิวเตอร์ เริ่มเข้ามามีความสำคัญต่อการใช้งาน ในสังคมปัจจุบันการเก็บข้อมูลในคอมพิวเตอร์หรือการติดต่อสื่อสารในระบบคอมพิวเตอร์เป็นเรื่องสำคัญที่บางครั้งองค์กรนั้น ๆจะต้องปกปิดเป็นความลับโดยมีวิธีการเข้าถึงข้อมูลโดยใช้รหัสผ่าน (PASSWORD) เพื่อจำกัดสิทธิ์ความสำคัญของผู้ใช้ การใช้วิธี Copy Protection เช่น เกมส์หลาย ๆ อย่างใช้กันในเครื่องไมโครคอมพิวเตอร์เพื่อกันการก็อปปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือการเข้ารหัสข้อมูล (Cryptography หรือ Data Encryption) เพื่อให้ข้อมูลเปลี่ยนไปจากเดิมจนไม่สามารถเข้าใจได้ ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิมได้

วิธีการทำ การเข้าและถอดรหัสนั้น มีหลายวิธี ๆ ใดมีความสลับซับซ้อนสูงจะทำให้การพยายามที่จะถอดรหัสทำได้ยากขึ้นความสามารถของเครื่องขนาดใหญ่ เช่น Super Mainframe หรือ Mini Computer ระบบปฏิบัติการเหล่านี้เป็นแบบ Multitasking คือใช้พร้อมๆกันได้หลายคน ข้อมูลที่อยู่ในระบบปฏิบัติที่ใหญ่ ๆ แบบนี้จึงจำเป็นต้องมีการรักษาความปลอดภัยของข้อมูลที่ดีพอเช่น การเก็บข้อมูลของรหัสผ่านก็เก็บในรูปของ Data Encryption ซึ่งต้องใช้วิธีการคำนวณมากซับซ้อนหลายรอบ ๆ

ดังนั้นจะเห็นว่าการทำการเข้าและถอดรหัสเป็นสิ่งที่สำคัญและจำเป็นทำให้ผู้วิจัยเกิดแนวความคิดที่จะคิดค้นวิจัย Algorithm ที่มีประสิทธิภาพขึ้น

1.1 วัตถุประสงค์ของการวิจัย

คิดค้นเทคนิคการเข้าและถอดรหัส เพื่อให้มีประสิทธิภาพมากขึ้นกว่าแบบการเข้าและถอดรหัสแบบมาตราฐาน Data Encryption Standard(DES) เทคนิคใหม่จะใช้หลักการของระบบมาตรฐาน DES และปรับปรุงการสลับสับเปลี่ยน (Permutation หรือ Transposition) และการแทนค่า (Function_F หรือ Substitution) ให้ผิดไปจากเดิมและกำหนดกุญแจยาวกว่าเดิมจาก ๕๖ บิท เป็น ๑๑๒ บิททดลองเทคนิคที่ได้โดยเขียนโปรแกรมภาษา C บนเครื่องไมโครคอมพิวเตอร์

1.2 ทฤษฎีและหลักการที่เกี่ยวข้อง

DES ได้รับการประกาศให้เป็นมาตรฐานของประเทศอเมริกาเพื่อใช้เป็นสาธารณทั่วไปและองค์การมาตรฐานระหว่างประเทศ (ISO) ได้ให้การยอมรับเป็นมาตรฐานสากล

ระบบมาตราฐาน DES เป็นวิธีการสลับสับเปลี่ยนและการแทนค่า มาทำงานร่วมกันถึง ๑๖ วงรอบ โดยข้อมูลปกติเดิมจะผ่านการเข้ารหัสเป็นบล็อกของจำนวน ๖๔ บิทและสามารถกำหนดกุญแจยาวเป็นตัวเลข ๕๖ บิทที่ผู้ใช้สามารถเปลี่ยนแปลงตามความต้องการ

DES แบ่งเป็น ๒ ส่วน คือ

๑. ส่วนสร้างกุญแจ มีความยาว ๕๖ บิท

๒. ส่วนเข้าและถอดรหัส มีความยาว ๖๔ บิท

การถอดรหัส กระทำได้จะต้องรู้ถึง

๑. ทฤษฎีที่เข้ารหัส

๒. กุญแจ

ถ้ารู้ทฤษฎีแต่ไม่รู้กุญแจต้องใช้เวลานานเพื่อทำการทดลองทุก ๆ กุญแจที่เป็นไปได้ กุญแจใช้ ๕๖ บิท = ๒๕๖ ประมาณ ๗ หมื่นล้านล้าน กุญแจที่เป็นไปได้แต่ถ้าไม่รู้เทคนิคการเข้ารหัสจะไม่สามารถถอดรหัสได้เลย

1.3 ประโยชน์ที่คาดว่าจะได้รับ

จะได้เทคนิคการเข้าและถอดรหัสแบบใหม่ และ ให้คงความสลับซับซ้อนเหมือนหรือมากกว่า DES เพิ่มกุญแจถอดรหัสให้ยาวขึ้นจากเดิม ๕๖ บิท เป็น ๑๑๒ บิท นั้นหมายถึงจะทำให้การลักลอบถอดรหัสเป็นไปได้ยากมากกว่าแบบเดิมคือต้องใช้เวลานานจึงจะสามารถลองการถอดรหัสของทุก ๆ กุญแจที่เป็นไปได้

ผลของการทำวิจัยนี้จะได้ประโยชน์อีกอย่างหนึ่งคือได้เรียนรู้และเข้าใจการทำงานของ DES และนำความรู้นี้มาเปลี่ยนแปลงโครงสร้างของการเข้า และถอดรหัสเสียใหม่ ปกปิดวิธีการที่ปรับปรุงใหม่นี้เพื่อใช้ในการทหารจะส่งผลให้ไม่มีผู้ใดสามารถลักลอบถอดรหัสนี้ได้ ด้วยเหตุผลที่ว่าผู้ที่สามารถถอดรหัสได้นั้นจะต้องรู้เทคนิคของการเข้ารหัสเป็นสำคัญ

1.4 โครงสร้างของวิทยานิพนธ์

บทที่ 2 ทฤษฎี กล่าวถึงความเป็นมาแนวคิดของการเข้ารหัสรวมทั้งหลักเกณฑ์ที่ได้รับการยอมรับว่าเป็นการเข้ารหัสที่ดี ซึ่งนำไปสู่ระบบมาตราฐานของประเทศอเมริกา ในบทนี้ยังกล่าวถึงรูปแบบการใช้และคุณสมบัติของ DES ยังแสดงการสร้างรหัสลับด้วยคอมพิวส์เตอร์แบบง่ายและทฤษฎีของ DES อย่างละเอียด

บทที่ 3 การสร้างและออกแบบ กล่าวถึงการวิเคราะห์ทดลองส่วนต่าง ๆ ของระบบ DES เพื่อหาความสัมพันธ์ในส่วนต่าง ๆ ทั้งในด้านการเข้า ถอดรหัสและส่วนคำนวณหมายเลขกุญแจ เมื่อได้ผลวิเคราะห์สามารถสร้างวิธีการใหม่ให้แตกต่างไปจาก DES เพิ่มความยาวของกุญแจจาก 56 เป็น 112 บิท ซึ่งทำให้การจารกรรมข้อมูลทำได้ยากขึ้น ทดลองวิธีการที่สร้างขึ้นโดยเขียนโปรแกรมภาษา C บนเครื่องไมโครคอมพิวส์เตอร์

บทที่ 4 ผลการทดลอง แสดงผลการทดลองของระบบที่สร้างขึ้น การทดลองแบ่งเป็น 3 ส่วนดังนี้

1. ทดลองโปรแกรมเพื่อวัดเวลาที่ใช้ในการเข้ารหัส 1 ครั้ง กับข้อมูลขนาด 64 บิท
2. ทดลองโปรแกรมเพื่อวัดเวลาที่ใช้ในการเข้าและถอดรหัสกับแฟ้มข้อมูลต่าง ๆ
3. ทดลองโปรแกรมการเข้าและถอดรหัสของแฟ้มข้อมูล โดยใช้รหัสกุญแจที่ต่างกันด้วยวิธีของ DES และวิธีที่สร้างขึ้น เปรียบเทียบผลลัพธ์ที่ได้ของทั้งสองวิธี

บทที่ 5 สรุป สรุปถึงความสำเร็จของวิทยานิพนธ์ที่แสดงให้เห็นถึงแนวคิดของ DES และวิธีที่เปลี่ยนส่วนต่าง ๆ ทำให้แตกต่างไปจากเดิมและถ้าปกปิดวิธีการใหม่นี้ทำให้การเข้าและถอดรหัสเป็นเอกลักษณ์ แสดงถึงเวลาในการจารกรรมข้อมูลที่ใช้ขบวนการเข้ารหัสที่สร้างขึ้นต้องใช้เวลานานกว่าวิธีของ DES มาก

บทที่ 2
ทฤษฎี

2. ทฤษฎีของ DES

วิธีการทำ Data Encryption มีหลายวิธี วิธีที่ยอมรับจำเป็นต้องมีการรักษาความปลอดภัยที่ดีพอ ในบทนี้จะกล่าวถึงความเป็นมาแนวคิดของการเข้ารหัสรวมทั้งหลักเกณฑ์ที่ได้รับการยอมรับว่าเป็นการเข้ารหัสที่ดี ซึ่งนำไปสู่ระบบมาตราฐานของประเทศอเมริกา ยังกล่าวถึงรูปแบบการใช้และคุณสมบัติของ DES และแสดงการสร้างรหัสลับด้วยคอมพิวส์เตอร์แบบง่ายและทฤษฎีของ DES อย่างละเอียด

2.1 หลักพื้นฐานของการเข้ารหัส

ขอนำเสนอหลักพื้นฐานของการเข้ารหัสดังนี้

2.1.1 วิธีการจารกรรมข้อมูลกระทำได้หลายกรณี
2.1.2 ลักษณะของระบบการเข้ารหัสที่ดี
2.1.3 มาตรฐานการเข้ารหัสลับของข้อมูล
2.1.4 รูปแบบของการใช้ DES
2.1.5 คุณสมบัติของ DES
2.1.6 การสร้างรหัสลับด้วยคอมพิวเตอร์

2.1.1 วิธีการจารกรรมข้อมูลกระทำได้หลายกรณี

ปัจจุบันการส่งข้อมูลระยะไกลมีความสำคัญมากขึ้น โอกาสที่ข้อมูลจะถูกโจรกรรมก็มีมากด้วยในการส่งข้อมูลข่าวสารจากผู้ส่ง S ไปตามสายส่ง T เพื่อไปยังผู้รับ R หากมีบุคคลภายนอก O มาทำการขัดขวางหรือจารกรรมข้อมูลข่าวสารนั้น เขาสามารถกระทำได้หลายกรณี คือ

1. ป้องกันไม่ให้ข้อมูลนั้นไปถึงผู้รับ R (Interruption)
2. ขโมยดูหรือฟังข้อมูลข่าวสารนั้น (Interception)
3. ยึดและแปรเปลี่ยนข้อมูลข่าวสาร (Modification)
4. การสอดแทรกข้อมูลข่าวสารเสมือนมาจากผู้ส่ง S (Fabrication)

2.1.2 ลักษณะของระบบการเข้ารหัสที่ดี

ระบบการเข้ารหัสลับที่ดีควรเป็นระบบที่มีลักษณะสมบัติดังนี้

1. การกระจายความถี่ของตัวอักษรและกลุ่มตัวอักษรที่เกิดขึ้นของข้อมูลที่เข้ารหัสแล้วควรมีค่าใกล้เคียงกัน

2. ระบบควรป้องกันไม่ให้คำหรือวลีของข้อมูลที่เกิดซ้ำ ๆ กันเมื่อเข้ารหัสลับแล้วไปเป็นข้อมูลเข้ารหัสตัวเดิมในที่นี้จะหมายถึงว่าระบบควรเข้ารหัสลับเป็นกลุ่มของตัวอักษรแทนที่จะเป็นการเข้ารหัสลับแต่ละตัวอักษร

3. ระบบการเข้ารหัสลับควรจะเป็นระบบที่ให้ผู้ใช้เลือกขนาดของกุญแจเองได้เพื่อป้องกันการแอบถอดรหัส และต้องใช้เวลายาวนานเพื่อการถอดรหัสลับนั้น

4. ระบบการเข้ารหัสลับที่ดีควรเป็นระบบที่แก้ไขปัญหาอุบัติเหตุไม่ตั้งใจที่เกิดจากการใช้กุญแจที่ผิด การถอดรหัสลับข้อมูลทั้ง ๆ ที่ยังไม่ได้เข้ารหัสลับไว้เลย หรือการเข้ารหัสลับกับข้อมูลซ้อนกันสองครั้งในที่นี้จะหมายถึงว่าระบบการเข้ารหัสลับจะไม่มีผลเสียหายไม่ว่าเราจะเข้ารหัสลับหรือถอดรหัสลับ เช่น ถ้าเราใช้กุญแจรหัสลับที่ผิดในการถอดรหัสลับเราก็เพียงทำการเข้ารหัสลับด้วยกุญแจตัวเดิมจากนั้นก็ถอดรหัสลับด้วยกุญแจตัวใหม่ที่ถูกต้อง

5. วิธีการเข้าและถอดรหัสควรเป็นวิธีที่เข้าใจง่าย เพราะถ้าระบบยากแล้วอาจทำให้เกิดข้อผิดพลาดได้ง่าย หรืออาจทำให้ลืมได้ง่าย

6. ข้อผิดพลาดที่เกิดขึ้น ในการสร้างรหัสลับต้องไม่แผ่กระจายให้ข้อผิดพลาดเพิ่มมากขึ้นเพราะถ้ามีข้อผิดพลาดเกิดขึ้นเล็กน้อย ผู้รับอาจสามารถคาดเดาตัวอักษรที่ขาดหายหรือผิดพลาด

7. ขนาดของข้อมูลเข้ารหัสลับแล้วจะต้องไม่ยาวกว่าข้อมูลปกติเดิม

2.1.3 มาตรฐานการเข้ารหัสลับของข้อมูล

รัฐบาลสหรัฐอเมริกาโดยสำนักงานมาตรฐานแห่งชาติ (U.S. National Bureau of Standards - NBS) ได้เล็งเห็นความสำคัญของการเข้ารหัสเพื่อใช้งานทั่วไปมาตั้งแต่ปีพ.ศ.2513 ได้พยายามให้เกิดวิธีการเข้ารหัสสาธารณะ (Public Encryption Algorithm) ที่มีกฎเกณฑ์ดังต่อไปนี้

1. มีระดับความปลอดภัยต่อข้อมูลสูง
2. มีการกำหนดขั้นตอนครบสมบูรณ์และเข้าใจได้ง่าย
3. ผู้ใช้งานทั่วไปสามารถนำไปใช้ได้อย่างอิสระ

4. สามารถปรับเปลี่ยนไปใช้เฉพาะงาน

5. สามารถสร้างเป็นระบบเข้ารหัสทางฮาร์ดแวร์ที่ง่ายและประหยัด

วิธีการที่เรียกว่า "Lucifer" ได้รับการพัฒนาปรับปรุงโดยบริษัทไอบีเอ็มเพื่อ NBS ให้ใช้งานทั่วไปต่อมาได้กลายมาเป็นระบบ DES ที่ย่อมาจาก Data Encryption Standard ซึ่งใน วันที่ 23 พฤศจิกายน พ.ศ. 2519 ระบบ DES นี่ก็ได้รับการประกาศตัวเป็นมาตรฐานของประเทศ เพื่อใช้เป็นแบบสาธารณะทั่วไป และองค์การมาตรฐานระหว่างประเทศ (ISO) ได้ยอมรับเป็นมาตรฐานสากลเรียบร้อยแล้ว

ระบบมาตรฐาน DES เป็นวิธีการที่ผสมรวมการเข้ารหัสลับพื้นฐานของแบบแทนที่และแบบสับเปลี่ยนมาทำงานร่วมกันถึง 16 วงรอบ โดยข้อมูลปกติเดิมจะผ่านการเข้ารหัสเป็นบล็อกของจำนวน 64 บิท และสามารถกำหนดกุญแจยาวเป็นตัวเลข 56 บิท ที่ผู้ใช้สามารถเปลี่ยนแปลงตามต้องการ

2.1.4 รูปแบบของการใช้ DES

การใช้ DES ให้เข้ากับลักษณะงานกระทำได้ในหลายรูปแบบการใช้งานในระบบคอมพิวเตอร์โดยรวมจะใช้ป้องกันการลักลอบดักฟังข้อมูล เมื่อถูกส่งออกไปตามสายส่งระยะไกล เช่น สายโทรศัพท์ หรือสายเฉพาะกิจ (LEASED LINE) เชื่อมต่อระหว่างสองจุดหรือใช้กับข้อมูลที่เก็บอยู่ในแผ่นเก็บข้อมูลอาจเป็นเทปหรือดิส (DISK) การป้องกันการลักลอบดักฟังข้อมูลจะติดตั้งระบบที่จุดส่งข้อมูล โดยทำการเข้ารหัสข้อมูลที่จุดส่งออกและที่จุดรับทำการถอดรหัส ทั้งทางด้านส่ง (เข้ารหัส) และด้านรับ (ถอดรหัส) ต้องใช้กุญแจเดียวกัน การเข้ารหัสข้อมูลในรูปของแฟ้มข้อมูลข้อมูลในแฟ้มข้อมูลจะถูกอ่านและทำขบวนการเข้ารหัสและเขียนเก็บที่แผ่นเก็บข้อมูลสำรอง เมื่อต้องการใช้ข้อมูลที่ถูกเก็บแบบเข้ารหัส ทำได้โดยการอ่านข้อมูลที่เข้ารหัสในแฟ้มข้อมูลและเข้าขบวนการถอดรหัส โดยใช้กุญแจเดียวกับการเข้ารหัสก็จะได้ข้อมูลเดิม

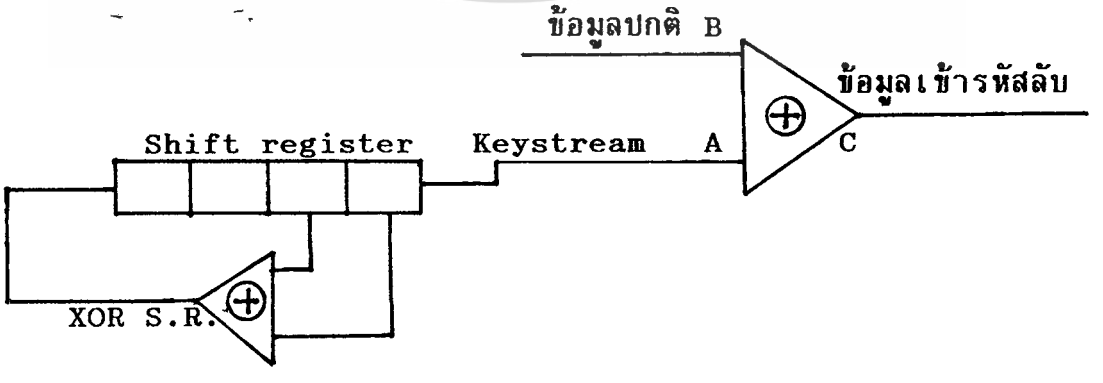
2.1.5 คุณสมบัติของ DES

ขบวนการเข้าและถอดรหัส DES ระบุในมาตรฐานนี้ทำการเปลี่ยนแปลงในรูปแบบของไบนารี 64 บิท ไปเป็นค่าในรูปแบบของไบนารี 64 บิท ซึ่งการแปลงเข้าหรือถอดรหัสขึ้นอยู่กับตัวแปรกุญแจ 56 บิท การหาค่ากุญแจหรือการพยายามที่จะลักลอบดักฟังข้อมูลไม่มีวิธีอื่นนอกจากการทดลองทุก ๆ กุญแจที่เป็นไปได้ กุญแจขนาด 56 บิท เท่ากับ 2^{56} ประมาณ เจ็ดหมื่นล้านล้าน กุญแจที่เป็นไปได้ ดังนั้นความพยายามที่จะถอดรหัส โดยวิธีการทดลองทุก ๆ เจ็ดหมื่นล้านล้านกุญแจจึงไม่สามารถกระทำได้ในเวลาอันสั้น ยิ่งกว่านั้นถ้ากุญแจเปลี่ยนบ่อย ๆ ความเสี่ยงต่อเหตุการณ์นี้จะน้อยลงอย่างมาก

2.1.6 การสร้างรหัสลับด้วยคอมพิวเตอร์แบบง่าย

วิธีการเข้ารหัสที่กล่าวมาแล้วเป็นเพียงทางทฤษฎี ในทางปฏิบัติเราสามารถเข้ารหัสได้ทั้งทางฮาร์ดแวร์และซอฟต์แวร์ การเข้ารหัสแบบสับเปลี่ยนสามารถใช้การกระทำด้วยการ Exclusive OR(คำสั่งXOR) ข้อมูลข่าวสารแต่ละไบต์กับกุญแจที่เลือกไว้แล้ว การเข้ารหัสแบบแทนที่สามารถนำตัวอักษรไปเทียบกับตารางเพื่อนำตัวอักษรใหม่มาแทนที่

รูปที่ 1 เป็นตัวอย่างของระบบการเข้ารหัสลับแบบสายธาร(Character Stream) ที่ใช้ Shift register และวงจร Exclusive-OR มาสร้างข้อมูลกุญแจเพื่อไปกระทำแบบ Exclusive-OR กับข้อมูลข่าวสารได้ข้อมูลเข้ารหัสลับ



รูปที่ 1 ระบบการเข้ารหัสแบบสายธาร(Character stream)

ตารางที่ 1 แสดงผลของการเข้าและถอดรหัสแบบสายธาร

ลำดับ	XOR S.R.	KEY Shift Register	จุด A	ข้อมูลเข้า จุด B	เข้ารหัส A⊕B= จุดC	ถอดรหัส A⊕C= จุดB
1	0	1 0 1 1	1	0	1	0
2	1	0 1 0 1	1	0	1	0
3	1	1 0 1 0	0	1	1	1
4	1	1 1 0 1	1	0	1	0
5	1	1 1 1 0	0	0	0	0
6	0	1 1 1 1	1	1	0	1
7	0	0 1 1 1	1	0	1	0
8	0	0 0 1 1	1	1	0	1

สมมุติให้ ข้อมูลที่จุดทำ Shift Register = 1011

และข้อมูลเข้า A4 = 1010 0100

จากตารางที่ 1 จุด A คือผลจากการทำ Shift Register ดังรูปที่ 1 ข้อมูลที่เข้ารหัสแล้ว ณ.จุด C ได้จากการทำ A XOR กับ B จะได้ 0100 1111 = 4F เมื่อนำข้อมูลที่เข้ารหัสแล้ว ณ.จุด C มาเป็นข้อมูลเข้าโดยมี KEY หรือ Shift Register ตัวเดิม (จุด A) จะได้ข้อมูลที่ถอดรหัส A ⊕ C = จุด B ดังในตารางที่ 1 เห็นได้ว่า ณ.จุด B ซึ่งเป็นข้อมูลเข้าเท่ากับข้อมูลที่ถอดรหัส (A ⊕ C) โดยมี A เป็น Keystream ตัวเดียวกัน

2.2 ทฤษฎีของ DES

การเข้ารหัสแบบ DES[4] ใช้เพื่อป้องกันการลักลอบการดักฟังข้อมูลในระบบคอมพิวเตอร์ในบทนี้จะกล่าวถึง หลักการและทฤษฎีของ DES โดยใช้วิธีทางคณิตศาสตร์เข้ามาสลับสับเปลี่ยนข้อมูล ๆ ที่เข้ามาผ่านขบวนการเข้ารหัสจะถูกสลับสับเปลี่ยนเป็นข้อมูลที่ไม่มีรูปแบบเรียกว่า CIPHER การถอดรหัสจะนำข้อมูลที่ไม่มีรูป

แบบนี้มาเข้าขบวนการถอดรหัส จะถูกสลับสับเปลี่ยนเป็นข้อมูลเดิมได้ ทฤษฎีทางคณิตศาสตร์ที่จะกล่าวอธิบายถึงวิธีการเข้าและถอดรหัสโดยใช้เลขฐาน 2 กุญแจประกอบด้วยเลขฐาน 2 64 บิท ใช้ในขบวนการเข้าและถอดรหัสและอีก 8 บิท ใช้ในการตรวจสอบข้อผิดพลาด(Error Detection) กุญแจที่ใช้จริงมีขนาด 56 บิท

ในแต่ละด้านของการเข้าและถอดรหัสจะมีกุญแจที่เหมือนกันกุญแจที่เลือกใช้จะเป็นกุญแจที่เมื่อใช้ในการเข้ารหัสและถอดรหัสแล้วจะได้ข้อมูลเดิมการใช้กุญแจที่ถอดรหัสที่ไม่ใช้กุญแจที่ใช้ในการเข้ารหัสจะส่งผลให้เมื่อถอดรหัสแล้วจะไม่ได้ข้อมูลเดิมด้วยเหตุผลนี้การใช้กุญแจที่เหมือนกันและเก็บค่าของกุญแจเป็นความลับจะป้องกันการดักฟังหรือลักลอบขโมยข้อมูลได้

ข้อมูลที่จะถอดรหัสได้นั้นจะต้องใช้กุญแจเดียวกับการเข้ารหัสเท่านั้นผู้ใดที่รู้หลักการหรือทฤษฎีของการเข้ารหัส แต่ไม่รู้หมายเลขกุญแจที่ใช้จะไม่สามารถถอดรหัสได้โดยง่าย ผู้ที่รู้ทฤษฎีและหมายเลขกุญแจเท่านั้นที่จะถอดรหัสได้ในเวลาอันสั้น

2.2.2 ขบวนการเข้าและถอดรหัส

แสดงในรูปที่ 2 ข้อมูลเข้าเป็นบล็อก T_i เข้าสู่ขบวนการสลับสับเปลี่ยนตำแหน่ง IP และ ข้อมูลออก $T_o = IP(T)$ ข้อมูล T_o เข้าสู่ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้ง ซึ่งเรียกว่า Function_F เมื่อเสร็จขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้งจะเข้าสู่ขบวนการสลับสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1} ผลลัพธ์ที่ได้เป็นข้อมูลที่ผ่านขบวนการเข้ารหัสแล้ว

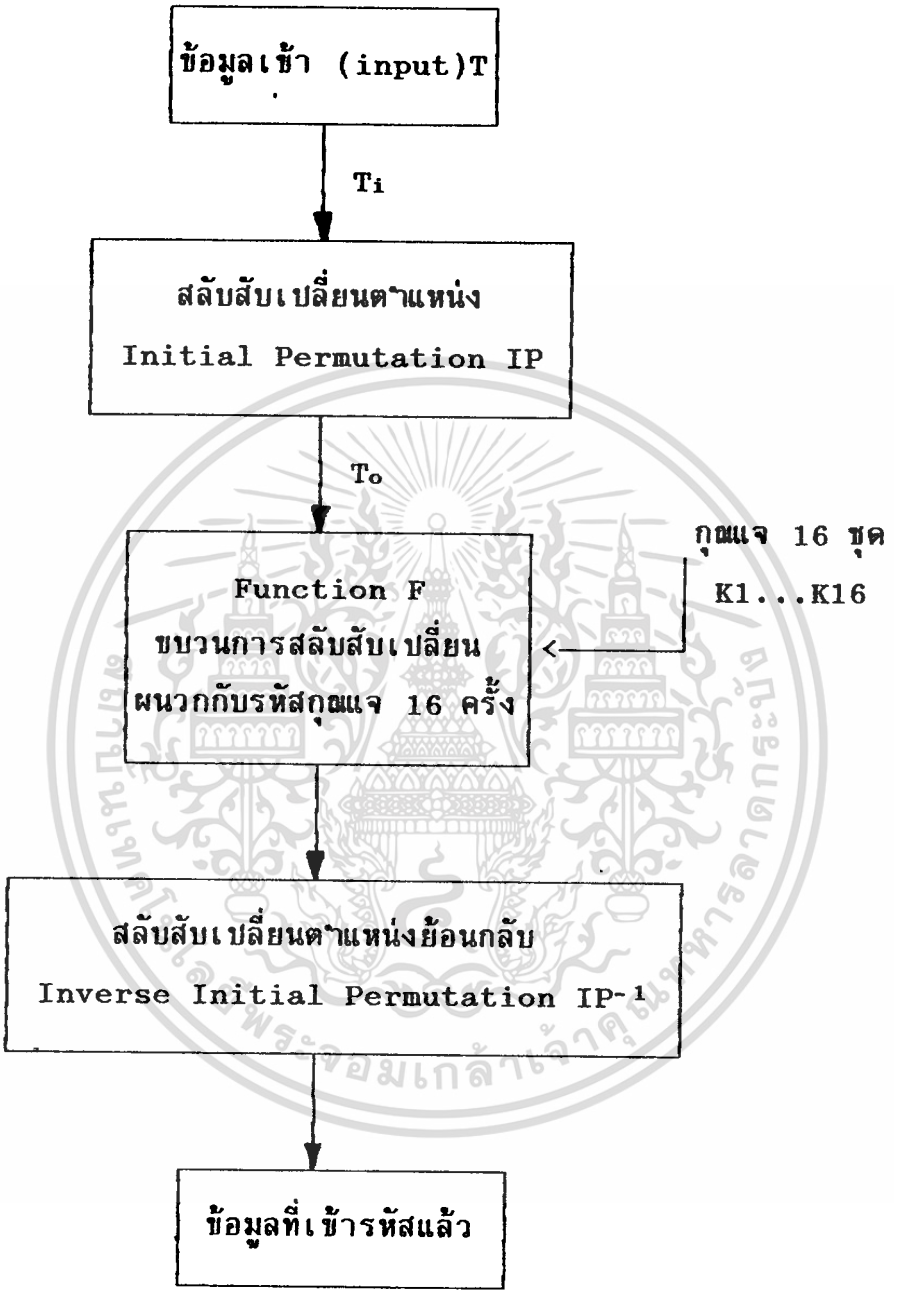
ตารางที่ 2 และ 3 แสดงตารางสลับสับเปลี่ยนตำแหน่ง IP และ IP^{-1} การใช้ตารางควรอ่านจากซ้ายไปขวา บนลงล่างเช่น สลับสับเปลี่ยนจาก $T_i = t_1 t_2 \ldots t_{64}$ เป็น $T_o = t_{58} t_{50} \ldots t_7$

จากรูปที่ 2 ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้งแต่ละครั้งเรียกว่า Function_F อยู่ระหว่างการทำสลับสับเปลี่ยนตำแหน่ง IP กับการเข้าสลับสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1} ตารางที่ 2,3 เป็นข้อมูลที่ได้จากทฤษฎีของ DES [3],[4],[7],[8]

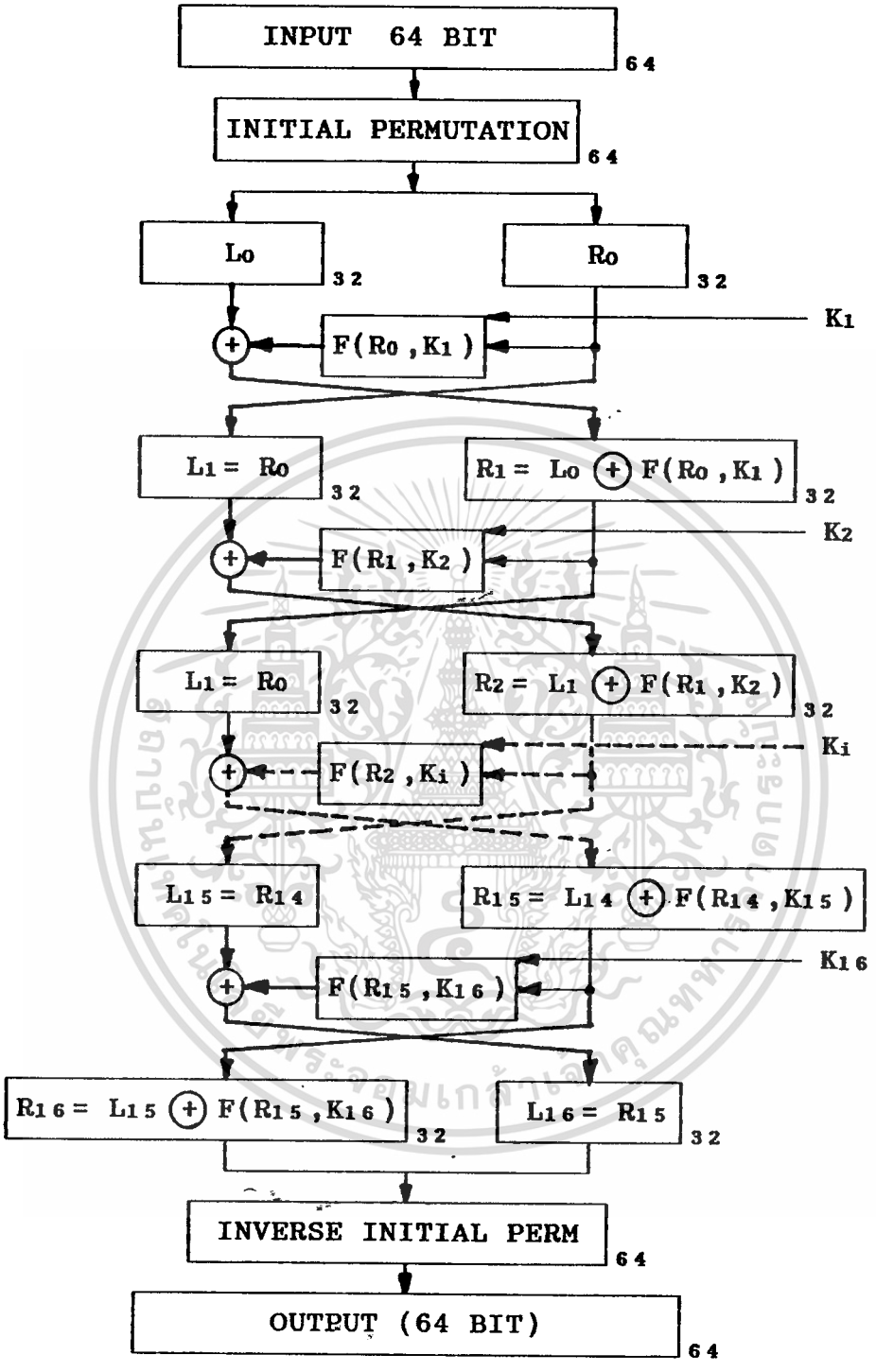
ให้ T_i แสดงถึงผลลัพธ์ของการทำครั้งที่ i แบ่ง T_i ออกเป็น 2 ส่วนๆ ละ 32 บิท ให้เป็นด้านซ้าย L และด้านขวา R ฉนั้น $L_i R_i$ จะเป็น

$$L_i = t_1 \ldots t_{32}$$

$$R_i = t_{33} \ldots t_{64}$$



รูปที่ 2 แสดงขบวนการเข้าและถอดรหัส



รูปที่ 3 แสดงวิธีการเข้ารหัสอย่างละเอียด

ตารางที่ 2 ตารางสลับสับเปลี่ยนตำแหน่ง IP

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

ตารางที่ 3 ตารางสลับสับเปลี่ยนตำแหน่ง IP^{-1}

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

ดังนั้น

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \qquad (1)$$

เครื่องหมาย $\oplus$ หมายถึง การทำเอ็คคลูสซีบออ (Exclusive-Or) และ Ki คือหมายเลขกุญแจมีความยาว 48 bit

ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ มีทั้งหมด 16 รอบ แสดงในรูปที่ 2 รอบที่ 1 ถึง 15 สลับสับเปลี่ยนตาม (1) ในรอบที่ 16 ซึ่งเป็นรอบสุดท้าย ด้านซ้าย L และด้านขวา R จะไม่สลับที่กันข้อมูล R_{16} L_{16} ถูกใช้เป็นข้อมูลเข้าเพื่อทำการสลับสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1}

2.2.2 Function_F และ S_box

รูปที่ 4 แสดงการทำ Function_F (R_{i-1}, K_i) การทำงานเริ่มจาก R_{i-1} ถูกสลับสับเปลี่ยนเป็น 48 บิท โดยใช้ $E(R_{i-1})$ ตารางที่ 4 แสดงถึงตำแหน่งบิท E ตาราง E เป็นลักษณะเดียวกับการทำสลับสับเปลี่ยนตำแหน่ง IP และสลับสับเปลี่ยนตำแหน่งย้อนกลับ IP^{-1} มีส่วนที่ไม่เหมือนก็คือ R_{i-1} ถูกใช้มากกว่า 1 ครั้งเช่น $R_{i-1} = r_1 r \ldots r_{32}$, $E(R_{i-1}) = r_{32} r_1 r_2 \ldots r_{32} r_1$

ต่อมา $E(R_{i-1})$ และ K_i มาทำ Exclusive-Or ผลลัพธ์ที่ได้ถูกแบ่งเป็นข้อมูลขนาด 6 บิท 8 ชุด คือ ชุดที่ $B_1, \ldots, B_8$ จะได้

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \ldots B_8 \quad (2)$$

ข้อมูลในแต่ละชุด B_j ซึ่งมีขนาด 6 บิท จะใช้เป็นข้อมูลเข้าใน Function(S_box)S_j และข้อมูลออกจะมีขนาด 4 บิท เขียนในรูปของฟังก์ชั่นได้

$$S_j(B_j) \quad (3)$$

S_box จะมีทั้งหมด 8 ชุดหลังจากผ่านขบวนการ S_box จะได้ผลลัพธ์ขนาด 32 บิท และเข้าขบวนการสลับสับเปลี่ยน P ซึ่งแสดงในตาราง P ตารางที่ 4,5 เป็นข้อมูลที่ได้จากทฤษฎีของ DES [3],[4],[7],[8]

ผลลัพธ์ของ S_box หลังการเข้าขบวนการสลับสับเปลี่ยน P จะอยู่ในรูปของ $F(R_{i-1}, K_i)$ เขียนได้เป็น

$$P(S_1(B_1)\ldots S_8(B_8)) \quad (4)$$

ตารางที่ 4 ตาราง P

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

ตารางที่ 5 ตาราง E

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

วิธีการเข้า S_box อธิบายได้ดังนี้

ให้ B_j เป็นข้อมูลเข้ามีขนาด 6 บิท ($b_1 b_2 b_3 b_4 b_5 b_6$) ให้ j เป็นเลขจำนวนเต็มมีค่าระหว่าง 1 ถึง 8 แบ่ง B_j ออกเป็น 2 ส่วนเรียกว่า Row และ Column ให้ในส่วนของ Row = $b_1 b_6$ และส่วนของ Column = $b_2 b_3 b_4 b_5$ นำค่า Row และ Column มาเทียบในตาราง S_j จะได้ผลลัพธ์เป็นค่า 4 บิท ดังนั้นเขียนได้เป็น

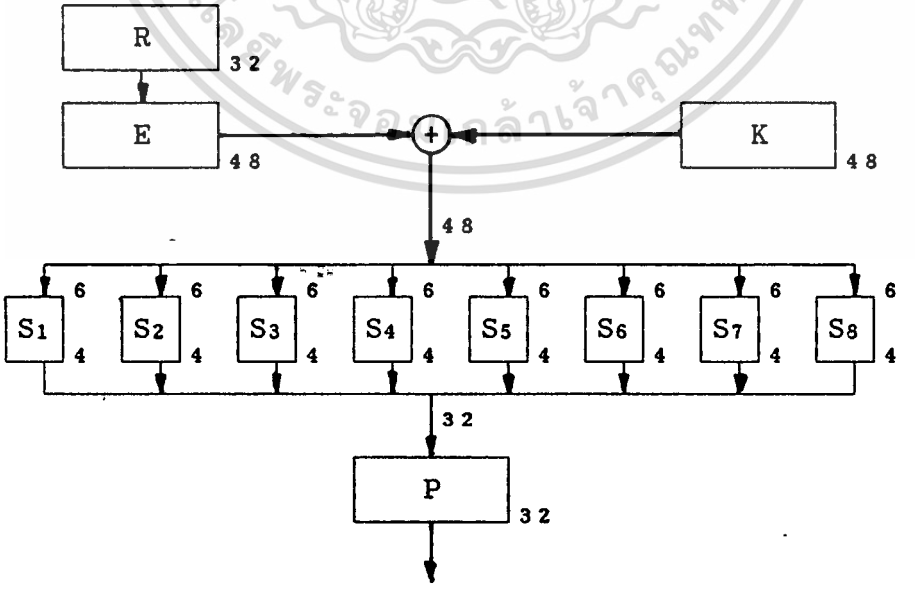
$S_j(B_j)$ ให้ผลลัพธ์ขนาด 4 บิท

<u>ตัวอย่าง</u>

ถ้า B_1 = 101010 S_1 จะได้ผลลัพธ์เป็น Row = 10 = 2 Column = 0101 = 5 เทียบในตาราง S_1 จะได้ 6 ซึ่งเท่ากับ 0110

ถ้า B_7 = 100101 S_7 จะได้ผลลัพธ์เป็น Row = 11 = 3 Column = 0010 = 2 เทียบในตาราง S_7 จะได้ 13 ซึ่งเท่ากับ 1101

ตาราง S_1 ถึง S_7 เป็นส่วนหนึ่งของ ตารางที่ 5



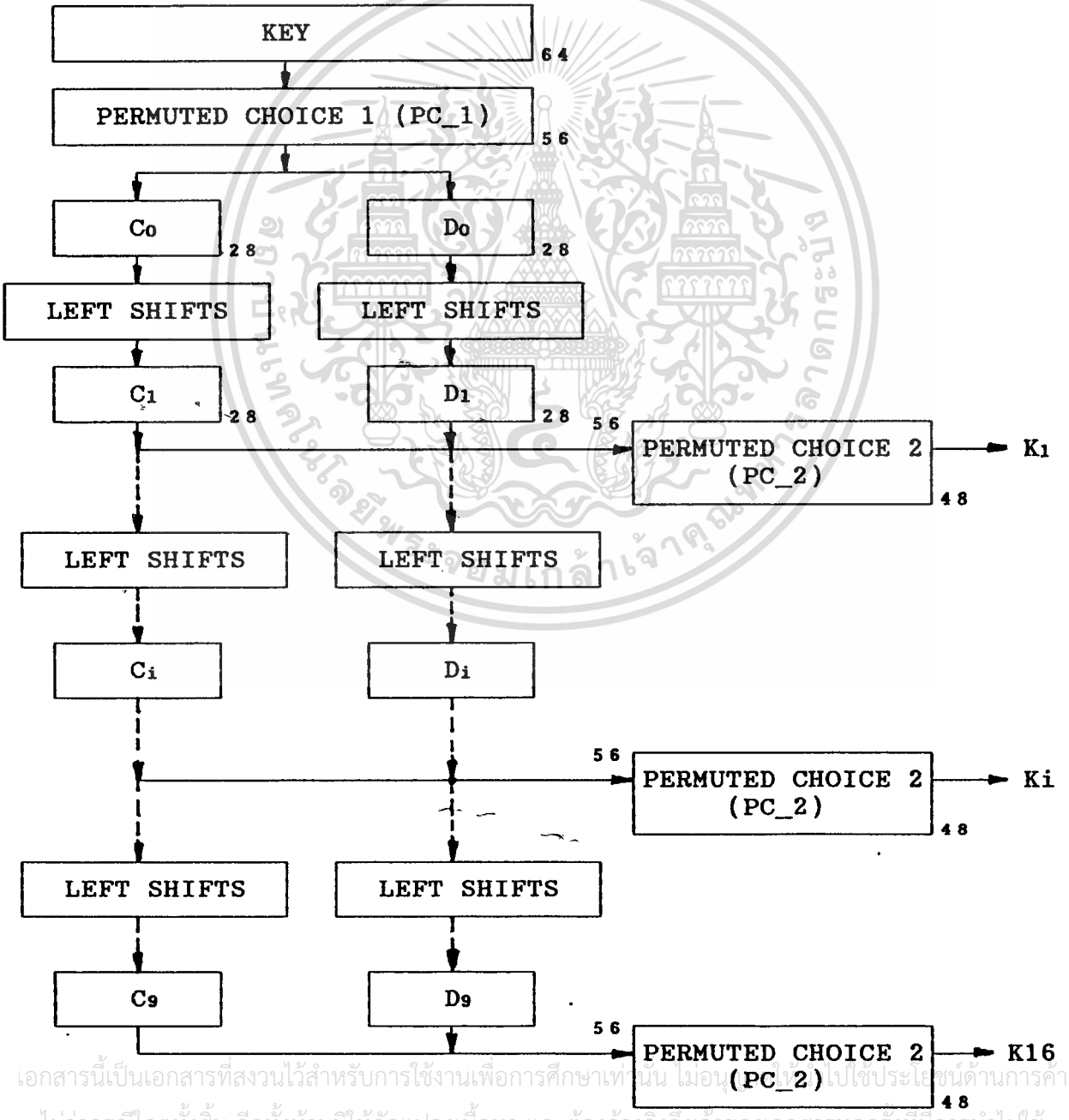
รูปที่ 4 แสดงการทำ Function F(R,K)

ตารางที่ 6 S_box

Row	Column 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	1	
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	

2.2.3 การคำนวณหมายเลขกุญแจ (Subkey)

ในแต่ละรอบของการทำสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ชุด จะใช้หมายเลขกุญแจแต่ละชุดซึ่งมีขนาด 48 บิท หมายเลขกุญแจ 16 ชุด ได้มาจาก Key ตามรูปที่ 5 Key เป็นข้อมูลเข้ามีขนาด 64 บิท ซึ่ง 8 บิท ในตำแหน่งที่ 8,16,...,64 ใช้เป็นบิทในการตรวจสอบข้อผิดพลาด (Parity bit) การทำขบวนการสลับสับเปลี่ยน Permuted PC_1 หรือ Permuted choice 1 จะตัดในส่วนของบิทตรวจสอบข้อผิดพลาดออก ใช้ 56 บิทที่เหลือเท่านั้นรายละเอียดอยู่ในตารางที่ 7



รูปที่ 5 แสดงวิธีการคำนวณหมายเลขกุญแจ 16 ชุด

ผลลัพธ์หลังจากการทำ PC_1 จะถูกแบ่งเป็น 2 ส่วนเท่าๆ กันเรียกว่า C และ D และใช้ C และ D หาค่าของหมายเลขกุญแจ K_i

กำหนดให้ C_i และ D_i ซึ่งได้มาจาก C ,D ใช้หาค่า K_i จะได้

$$C_i = LS_i(C_{i-1})$$

$$D_i = LS_i(D_{i-1})$$

LS_i คือ การทำการหมุนทางซ้ายเท่ากับจำนวนครั้งที่กำหนด ในตารางที่ 9 C_o และ D_o เป็นข้อมูลเริ่มแรกของ C และ D ดังนั้นหมายเลขกุญแจ K_i เขียนได้เป็น

$$K_i = PC\_2\ (C_i D_i)$$

ขบวนการสลับสับเปลี่ยน PC_2 อยู่ในตารางที่ 8

ตารางที่ 7 ตาราง PC_1

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

ตารางที่ 8 ตาราง PC_2

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

ตารางที่ 9 ตารางการหมุนซ้าย

ลำดับ	จำนวนครั้ง	ลำดับ	จำนวนครั้ง
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

กำหนดให้ หมายเลขกุญแจเท่ากับ d465 9cae 367c d9eb นำหมายเลขกุญแจที่ได้ทำการสลับสับเปลี่ยน PC_1 ได้ผลลัพธ์เท่ากับ cde3 ba70 983f ec50 ข้อมูลนี้ถูกแบ่งเป็นสองส่วนคือ Co = cde3 ba70

D0 = 983f ec50

ขั้นตอนต่อไปนำ Co และ Do มาหมุนตามตารางที่ 8 ซึ่งกำหนดให้

Co = cde3 ba7 0 Do = 983f ec5 0

ลำดับที่ 1 หมุน 1 ครั้ง Co = 9bc7 74f 0 Do = 307f d8b 0

ลำดับที่ 2 หมุน 1 ครั้ง Co = 378e e9f 0 Do = 60ff b16 0

ลำดับที่ 3 หมุน 2 ครั้ง Co = de3b a7c 0 Do = 83fe c59 0

ลำดับที่ 16 หมุน 1 ครั้ง Co = cde3 ba7 0 Do = 983f ec5 0

นำข้อมูลหลังจากการหมุนในแต่ละลำดับเข้าขบวนการสลับสับเปลี่ยน PC_2 ได้เป็นกุญแจ K_1 ถึง K_{16} ดังตารางที่ 10 กุญแจ 16 ชุดนี้ใช้ในขบวนการเข้าและถอดรหัสต่อไป

ตารางที่ 10 แสดงการคำนวณหมายเลขกุญแจตามขบวนการ DES

ลำดับ	Co	Do	#Key(K)
1	9bc774f0	307fd8b0	2319 173c 2915 3d31
2	378ee9f0	60ffb160	343e 171a 3f35 2810
3	de3ba7c0	83fec590	1f0c 3e33 3414 0d3e
4	78ee9f30	ffb1660	3d3b 240f 251b 2a0c
5	e3ba7cd0	3fec5980	0a3a 1b17 3c03 1b35
6	8ee9f370	ffb16600	1f15 323e 0e3a 2a2f
7	3ba7cde0	fec59830	3d2a 0738 0d27 3613
8	ee9f3780	fb1660f0	272c 3837 0b32 0537
9	dd3e6f10	f62cc1f0	2f1e 3d03 3924 0f33
10	74f9bc70	d8b307f0	1837 1e2f 153a 2d0b
11	d3e6f1d0	62cc1ff0	3e19 1715 3d29 1518
12	4f9bc770	8b307fd0	151c 2b3b 121b 1d2e
13	3e6f1dd0	2cc1ff60	2d3f 0636 1f0f 3228
14	f9bc7740	b307fd80	2f20 3f07 1a05 313f
15	e6f1dd30	cc1ff620	1e37 083d 233f 223a
16	cde3ba70	983fec50	1b1f 332d 2216 3c3b

จากรูปที่ 6 การทำขบวนการย้อนกลับ หรือ การถอดรหัส Decryphering ทำโดยใช้วิธีการเดิมการใช้หมายเลขกุญแจจะใช้จาก $K_{16}, K_{15}, \ldots, K_1$ เป็นจำนวน 16 ครั้ง

i เริ่มจาก 16 ถึง 1

$R_{i-1} = L_i$

$L_{i-1} = R_i = F(L_i, K_i)$

การทำขบวนการย้อนกลับจะย้อนกลับในส่วนของกุญแจจาก 16 ถึง 1 ส่วนวิธีการจะไม่ย้อนกลับ

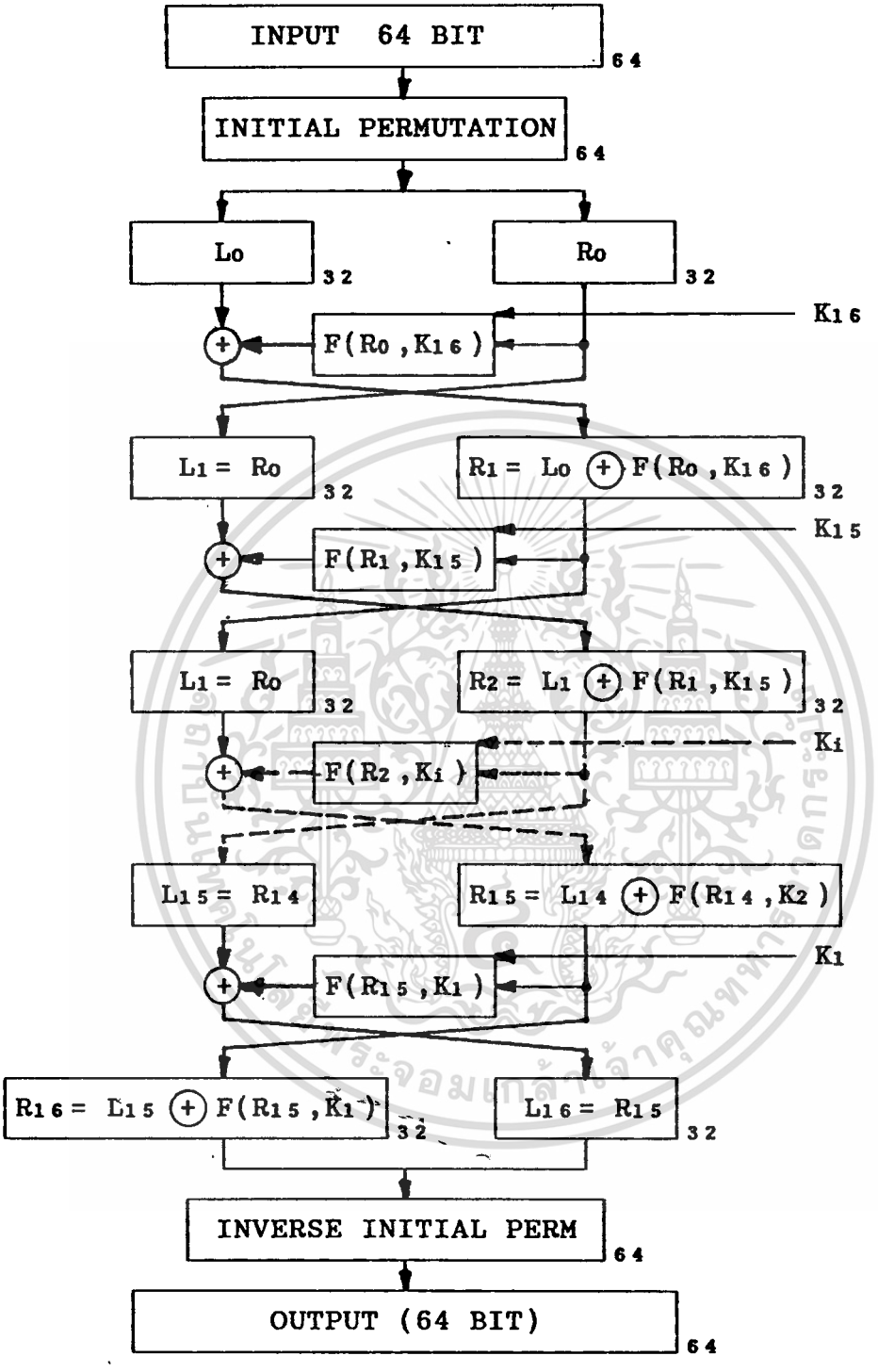
2.3 สรุป

ที่ได้นำเสนอแล้วนั้นเป็นทฤษฎีของ DES ซึ่งหาได้จากหนังสือ Data Security ทั่วไป [3],[4],[7],[8] สังเกตุได้ว่าจากทฤษฎี DES บอกถึงวิธีการเข้า,ถอดรหัส และสร้างหมายเลขกุญแจแต่ไม่มีการอธิบายถึงแนวคิดว่าถ้า

1. เปลี่ยนหมายเลขในตารางหนึ่งตารางใดจะเกิดผลอย่างไร
2. ความสัมพันธ์ของตารางต่าง ๆ มีอะไรบ้าง
3. หรือใน Function_F ถ้าไม่กำหนด Row และ Col ตามทฤษฎีจะมีผลอย่างไร
4. หมายเลขกุญแจที่สร้างขึ้นถ้าไม่ทำตามทฤษฎีจะได้หรือไม่

บทที่ 3 จะกล่าวถึงผลการทดลอง, วิเคราะห์และอธิบายถึงแนวความคิดข้างต้นเพื่อสร้างวิธีการใหม่ให้แตกต่างไปจาก DES

ผนวก ค. แสดงตัวอย่างข้อมูลการเข้า,ถอดรหัส และการคำนวณหมายเลขกุญแจของ DES



รูปที่ 6 การทำขบวนการย้อนกลับ

บทที่ 3
การสร้างและออกแบบ

3. การสร้างและออกแบบ

การที่จะออกแบบเทคนิคการเข้าและถอดรหัสได้นั้น จำเป็นต้องเข้าใจถึงเทคนิคการเข้าและถอดรหัสของ DES อย่างท่องแท้ ต้องเข้าใจถึงความสัมพันธ์ระหว่างตารางการสลับสับเปลี่ยนต่าง ๆ และ Function_F การเปลี่ยนตารางใด ๆ นั้นจะต้องให้สอดคล้องกับส่วนอื่น ๆ มิฉะนั้นเมื่อทำการถอดรหัสจะไม่ได้ข้อมูลเดิม ในระบบของ DES ทำการเข้า Function_F 16 รอบ ซึ่งต้องใช้หมายเลขกุญแจ 16 ชุด ความสลับซับซ้อนจะมาก อีกทั้งยังต้องเข้าใจถึงแนวคิดการคำนวณหมายเลขกุญแจ 16 ชุด จากข้อมูลกุญแจ 56 บิท รวมทั้งตารางการสลับสับเปลี่ยน PC_1 และ PC_2

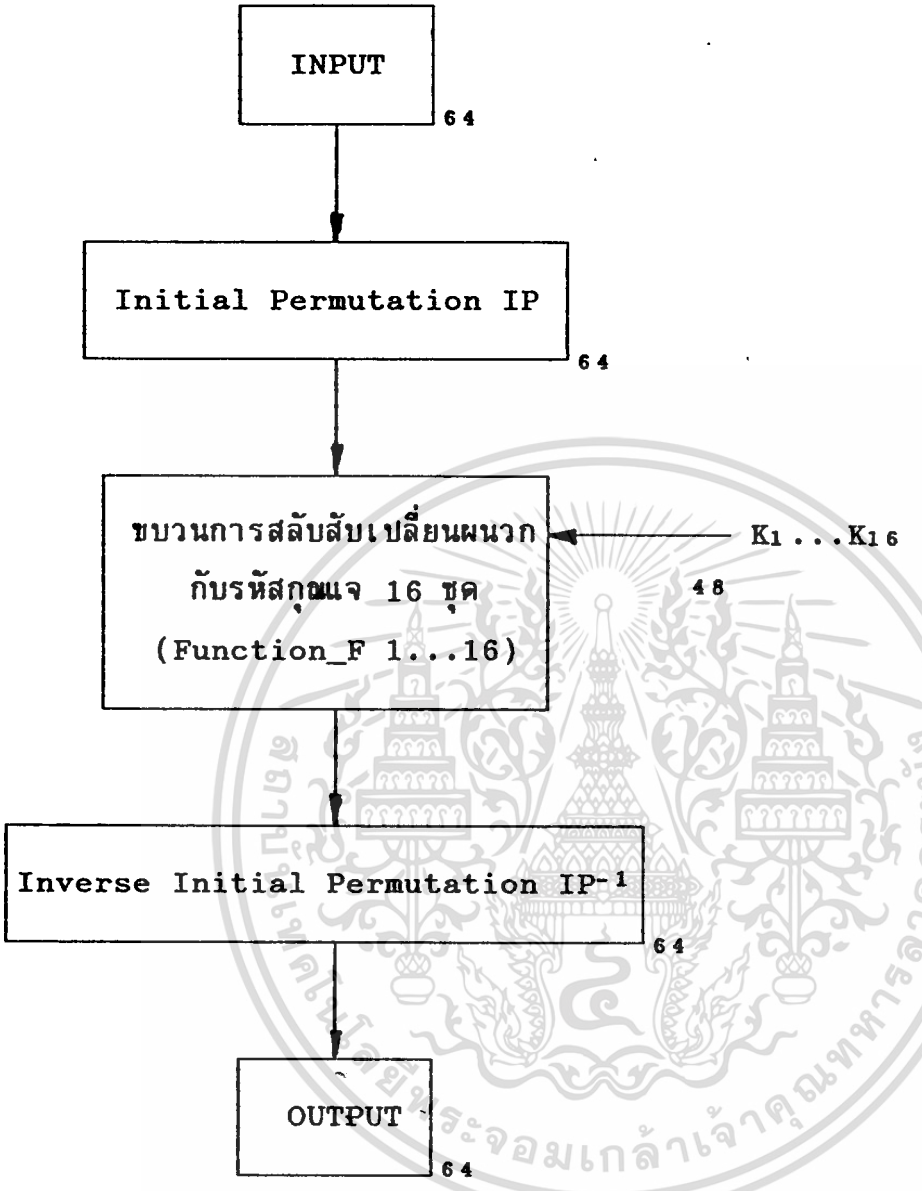
ในบทนี้จะกล่าวถึงการทดลอง, วิเคราะห์ของ

1. การทำการสลับสับเปลี่ยนที่ใช้ในตาราง IP,IP^{-1},E,P,PC-1, PC-2 และการทำ Function_F
2. หาความสัมพันธ์ของตารางต่าง ๆ และเปลี่ยนค่าในทุก ๆตาราง ให้แตกต่างไปจาก DES
3. เพิ่มจำนวนรอบในการทำ Function_F จาก 16 เป็น 18 รอบ
4. เพิ่มหมายเลขกุญแจจาก 16 ชุดเป็น 18 ชุด

หลักการที่ใช้ในการปรับปรุงจะกล่าวในหัวข้อต่อไป

ประโยชน์ที่จะได้รับคือจะได้เทคนิคใหม่ที่แตกต่างไปจาก DES และเป็นเอกลักษณ์อีกทั้งการเปลี่ยนขนาดกุญแจจาก 56 บิทเป็น 112 บิททำให้การจารกรรมข้อมูลใช้เวลานานขึ้น

รูปที่ 7 แสดงถึงภาพโดยรวมของการเข้าและถอดรหัสของ DES การเข้ารหัสเริ่มจากข้อมูลเข้า (Input) ซึ่งมีขนาด 64 บิท ผ่านขบวนการสลับสับเปลี่ยน IP ซึ่งสลับตำแหน่งบิทของข้อมูลเข้า ตามตารางที่ 3 และส่งผลลัพธ์ให้กับส่วนของขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ชุด ซึ่งการเข้ารหัสใช้หมายเลขกุญแจจาก K_1 ถึง K_{16} ข้อมูลที่ได้ถูกส่งไปยังขบวนการสลับสับเปลี่ยน IP^{-1} ตามตารางที่ 4 ผลลัพธ์ที่ได้คือข้อมูลที่ผ่านขบวนการเข้ารหัสแล้ว



รูปที่ 7 แสดงภาพโดยรวมของการเข้าและถอดรหัสของ DES

การถอดรหัสใช้หลักการเดียวกับการเข้าผิดกันที่การถอดรหัสจะใช้หมายเลขกุญแจจาก K_{16} ถึง K_1 ส่วนอื่น ๆ ยังคงใช้ลักษณะเดิม

ขอนำเสนอหลักการออกแบบแบ่งเป็น 6 ส่วน ดังนี้

3.1 การทำสลับสับเปลี่ยน

3.2 ขบวนการสลับสับเปลี่ยน IP และ IP^{-1}

3.3 แสดงการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้ง

3.4 แสดงการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง

3.5 แสดงการเข้าและถอดรหัสโดยทำ Function_F หลายครั้ง

3.6 ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ (Function_F และ S_Box)

3.7 การคำนวณหมายเลขกุญแจโดยใช้ขนาด 112 บิท

3.8 สรุป

3.1 การทำสลับสับเปลี่ยน

ขบวนการ DES ใช้หลักการสลับสับเปลี่ยนตำแหน่งหลายขั้นตอนทั้งในด้านการเข้ารหัส ถอดรหัสและคำนวณหมายเลขกุญแจ การทำสลับสับเปลี่ยนดังกล่าวได้แก่ ตาราง IP,IP^{-1},E,P,PC_1 และ PC_2 ขอนำเสนอวิธีการทำสลับสับเปลี่ยนในแบบต่าง ๆ ดังนี้

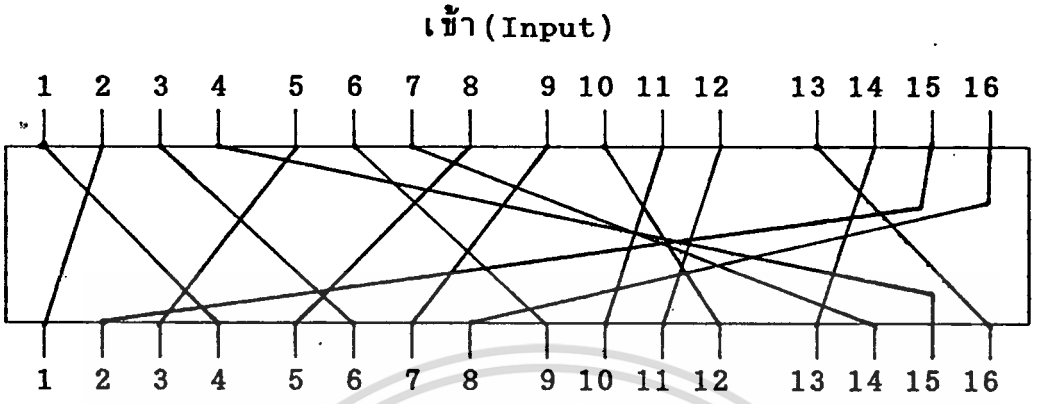
3.1.1 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิท และข้อมูลออก 16 บิท

3.1.2 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิท และข้อมูลออก 12 บิท

3.1.3 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 12 บิท และข้อมูลออก 16 บิท

3.1.4 การทำสลับสับเปลี่ยนและสลับสับเปลี่ยนย้อนกลับ

3.1.1 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิทและข้อมูลออก 16 บิท



2	15	5	1	8	3	9	16	6	11	12	10	14	7	4	13

รูปที่ 8 แสดงการทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิท ออก 16 บิท

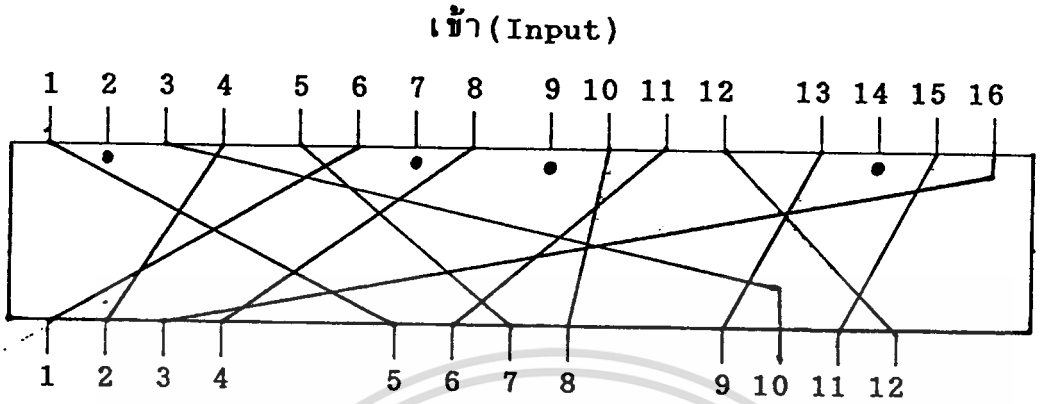
จากรูปที่ 8 บิทที่ 1 ของ Input ต่อเข้ากับบิทที่ 4 ของ Output
บิทที่ 2 ของ Input ต่อเข้ากับบิทที่ 1 ของ Output

บิทที่ 16 ของ Input ต่อเข้ากับบิทที่ 8 ของ Output
เขียนตารางสลับสับเปลี่ยนได้เป็น

2	15	5	1	8	3	9	16
6	11	12	10	14	7	4	13

ตัวอย่าง Input = A932
Output = 7c60
วิธีการนี้เป็นวิธีการเดียวกับการสลับสับเปลี่ยนของตาราง P แตกต่างกันที่ Input และ Output ในรูปที่ 8 ใช้ขนาด 16 บิท แต่ในตาราง P ใช้ขนาด 32 บิท

3.1.2 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิทและข้อมูลออก 12 บิท



6	4	16	8	1	11	5	10	13	3	15	12

รูปที่ 9 แสดงการทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 16 บิท ออก 12 บิท

ข้อมูลเข้าขนาด 16 บิท ออก 12 บิท มีบิทที่ไม่ใช้ 4 บิท คือ 2,7,9,14 เขียนตารางสลับสับเปลี่ยนได้เป็น

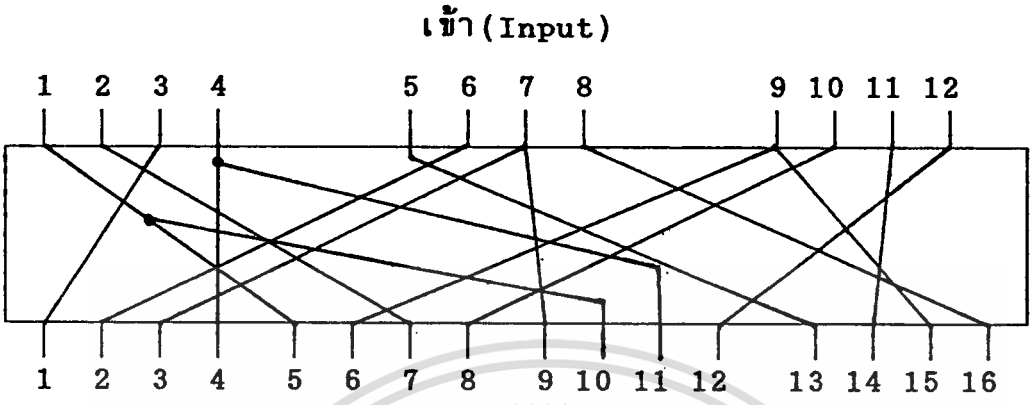
6	4	16	8	1	11
5	10	13	3	15	12

ตัวอย่าง Input = A932

Output = 1F7

วิธีการนี้เป็นวิธีการเดียวกับการสลับสับเปลี่ยนของตาราง PC_1 และ PC_2 แตกต่างกันที่ Input และ Output ในรูปที่ 9 ใช้ขนาด 16,12 บิทแต่ในตาราง PC_1 ใช้ขนาด 64,56 บิท และ PC_2 ใช้ขนาด 56,48 บิท

3.1.3 การทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 12 บิทและข้อมูลออก 16 บิท



3	6	7	4	1	9	2	10	7	1	4	12	5	11	9	8

รูปที่ 10 แสดงการทำสลับสับเปลี่ยนของขนาดข้อมูลเข้า 12 บิท ออก 16 บิท

ข้อมูลเข้าขนาด 12 บิท ออก 16 บิท มีบิทซ้ำ 4 บิท คือ 7,1,4,9 เขียนตารางสลับสับเปลี่ยนได้เป็น

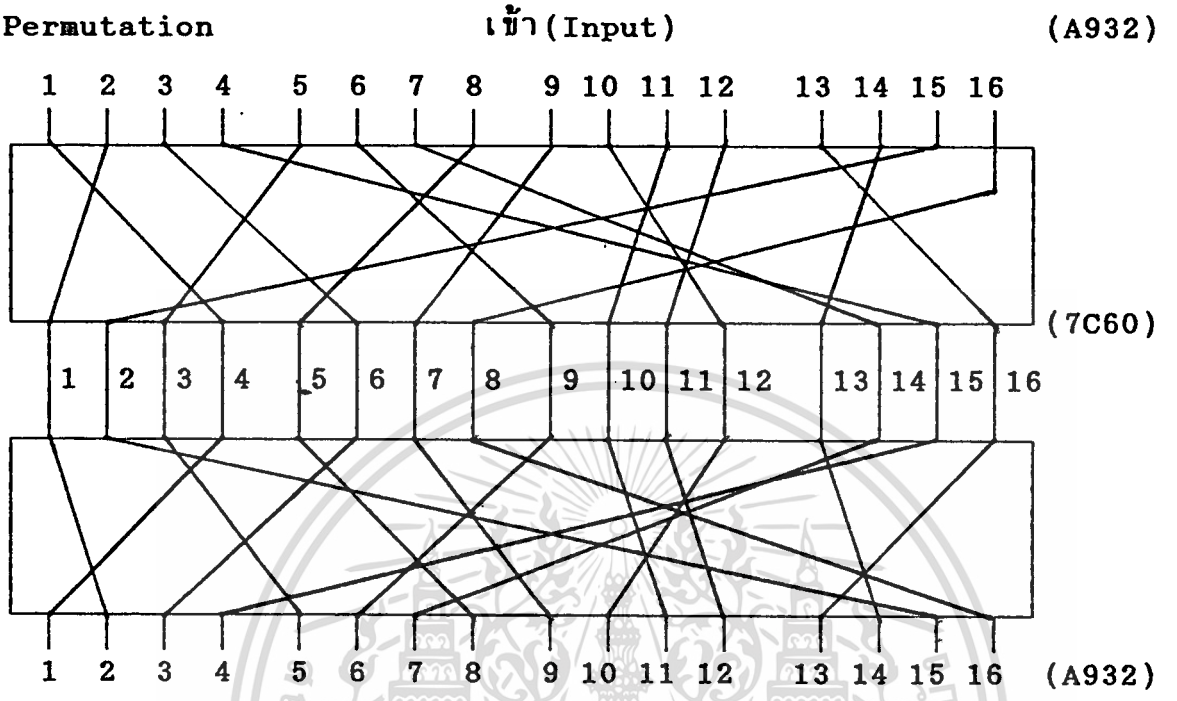
3	6	7	4	1	9	2	10
7	1	4	12	5	11	9	8

ตัวอย่าง Input = 34E

Output = CD06

วิธีการนี้เป็นวิธีการเดียวกับการสลับสับเปลี่ยนของตาราง E แตกต่างกันที่ Input และ Output ในรูปที่ 10 ใช้ขนาด 12,16 บิท แต่ในตาราง E ใช้ขนาด 32, 48 บิท

3.1.4 การทำสลับสับเปลี่ยนและสลับสับเปลี่ยนย้อนกลับ



รูปที่ 11 แสดงการทำสลับสับเปลี่ยนและสลับสับเปลี่ยนย้อนกลับ

ตาราง Permutation เขียนได้เป็น

2	15	5	1	8	3	9	16
6	11	12	10	14	7	4	13

ตาราง Inverse Permutation เขียนได้เป็น

4	1	6	15	3	9	14	5
7	12	10	11	16	13	2	8

ตัวอย่าง Input = A932
Permutation = 7C60
Inverse Permutation(Output) = A932

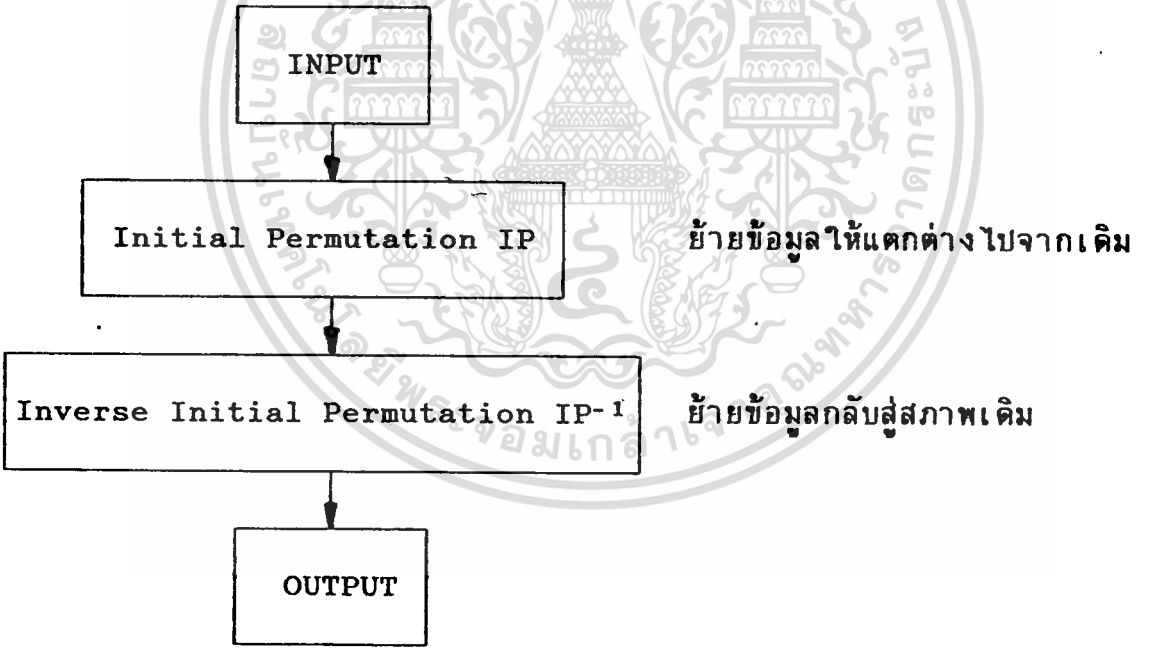
วิธีการนี้เป็นวิธีการเดียวกับการสลับสับเปลี่ยนของตาราง IP และ IP^{-1} แตกต่างกันที่ Input และ Output ในรูปที่ 11 ใช้ขนาด 16 บิท แต่ในตาราง IP และ IP^{-1} ใช้ขนาด 64 บิท

3.2 ขบวนการสลับสับเปลี่ยน IP และ IP^{-1}

จุดประสงค์ของการทำขบวนการสลับสับเปลี่ยน IP และ IP^{-1} คือการโยกย้ายบิทให้แตกต่างไปจากข้อมูลเดิมและ เมื่อทำการสลับสับเปลี่ยนย้อนกลับจะได้ข้อมูลเดิม

จากรูปที่ 7 จะทำให้การอธิบายดูง่ายขึ้นถ้าตัดส่วนที่เป็นขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ชุดออกจะเห็นว่าข้อมูลเข้าและออกจะมีค่าเดียวกัน นั้นคือขบวนการสลับสับเปลี่ยน IP รับ Input ขนาด 64 บิทมาสลับตำแหน่งตามตารางที่ 3 Initial-Permutation IP และส่ง Output ให้กับ IP^{-1} ซึ่งเป็นการทำขบวนการย้อนกลับกับ IP ตามตาราง Inverse Initial Permutation IP^{-1} ดังนี้

ค่า Input = ค่า Output



รูปที่ 12 แสดงการทำ IP และ IP^{-1} โดยตัดส่วนของขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ชุดออก

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

รูปที่ 13 แสดงตารางการสลับสับเปลี่ยน IP

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

รูปที่ 14 แสดงตารางการสลับสับเปลี่ยน IP^{-1}

1 <--- ตำแหน่งบิทที่ 1 ของตารางนั้น ๆ
58 <--- ค่าของบิทที่ 58 ของ Input

ข้อมูลลำดับที่ 1,2,64 หรือจุด A,B,C จากรูปที่ 9 จะถูกสลับที่ไปอยู่จุด A,B,C ของรูปที่ 10 ขบวนการสลับสับเปลี่ยน IP และ IP^{-1} ตามตารางที่ 1 และ ตารางที่ 4 นี้เป็นการทำขบวนการย้อนกลับกันตามรูปที่ 8

ตัวอย่าง ข้อมูล Input = A2DC 8F5A 6ACE 160C

หลังจากทำ IP = 3A4A E604 2711 BE7D

หลังจากทำ IP^{-1} = A2DC 8F5A 6ACE 160C

ค่า Input = ค่า Output(IP^{-1}) จุดประสงค์ของขบวนการสลับสับเปลี่ยน IP และ IP^{-1} คือการโยกย้ายข้อมูลให้แตกต่างไปจากเดิมเพื่อให้สับสนยากต่อการแกะลอย(Trace) ดู 3.1.4 ประกอบ

ตารางการสลับสับเปลี่ยน IP และ IP^{-1}สามารถกำหนดขึ้นเองได้

ตารางที่ 11 ตาราง IP ใหม่

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

ตารางที่ 12 ตาราง IP^{-1} ใหม่

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

3.3 แสดงการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้ง

ข้อมูลเข้าสู่ขบวนการนี้คือข้อมูล 32 บิทซ้ายและหมายเลขกุญแจ 48 บิท เพื่อให้เข้าใจถึงหลักการได้ง่ายขึ้น จึงตัดส่วนของการสลับสับเปลี่ยน IP และ IP^{-1} ออกและให้ทำ Function_F เพียงครั้งเดียวซึ่งตามกฎของ DES ต้องทำถึง 16 ครั้ง

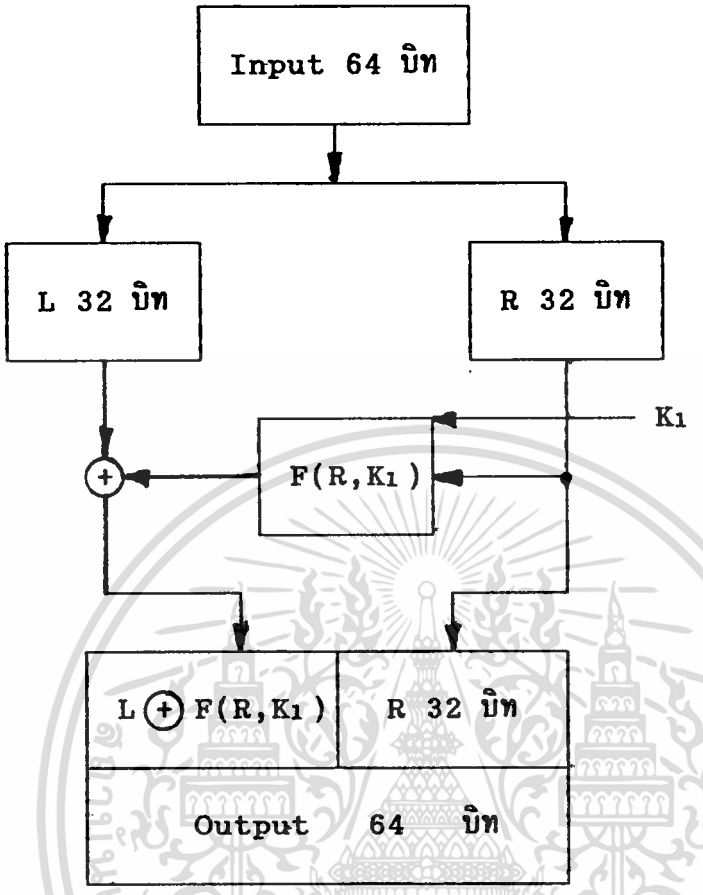
จากรูปที่ 15 ใช้เป็นการเข้าและถอดรหัสโดยใช้หมายเลขกุญแจเพียงชุดเดียวนั้นหมายถึงทำ Function_F เพียงครั้งเดียว

แสดงการถอดรหัส จากรูปที่ 15 กำหนดให้

Input = LR = 3A4A E604 2711 BE7D (64 บิท)

หมายเลขกุญแจ = K_1 = 3710 2520 0405 340E (48 บิท)

จะได้ Output = $[L \oplus F(R,K_1)]\ R$ = 5764 61CC 2711 BE7D (64 บิท)



รูปที่ 15 แสดงการเข้าและถอดรหัสโดยตัดส่วนที่ทำ IP และ IP^{-1} ออกและทำ Function_F เพียงครั้งเดียว

แสดงการถอดรหัส โดยนำข้อมูลจากการเข้ารหัสมาเป็นข้อมูลเข้าของการถอดรหัสดังรูปที่ 15

Input = $[L \oplus F(R,K_1)]\ R$ = 5764 61CC 2711 BE7D (64 บิท)

หมายเลขกุญแจ = K_1 = 3710 2520 0405 340E (48 บิท)

จะได้ Output = $\{[L \oplus F(R,K_1)] \oplus F(R,K_1)\}R = LR$

= 3A4A E604 2711 BE7D (64 บิท)

จะเห็นว่าข้อมูลเข้าของการทำการเข้ารหัสเท่ากับข้อมูลออกของการทำการถอดรหัส

ขออธิบายแนวคิด โดยแบ่งเป็น 3 ส่วน ดังนี้

3.2.1 XOR

3.2.2 Function_F

3.2.3 Output แบ่งเป็น 2 ส่วน

- $L \oplus F(R,K_1)$
- R 32 บิท

3.3.1 XOR

ในหลักการของ DES ใช้เทคนิคของ XOR ช่วยในการเข้าและถอดรหัส

$$(L \oplus X) \oplus X = L$$

จะสังเกตุได้ว่าเมื่อทำการ XOR กับข้อมูลใด ๆ L กับ X นำผลลัพธ์ที่ได้ $L \oplus X$ มาทำการ XOR กับค่าเดิม X จะได้ข้อมูลเดิม L

3.3.2 Function_F

นำ Input ซึ่งเป็น R 32 บิท และหมายเลขกุญแจ K_1 มาทำขบวนการสลับสับเปลี่ยนโดยใช้ S_Box ผลลัพธ์ที่ได้มาทำการ XOR กับ L เขียนได้เป็น $L \oplus F(R,K_1)$ ในขณะนี้ขอให้คิดว่า Function_F เป็น Function หนึ่งโดยไม่ต้องคำนึงถึงความสลับซับซ้อนภายใน

กำหนดให้ $F(R,K_1) = X$

3.2.3 Output

จากรูปที่ 15 จะเห็นว่าผลลัพธ์จากการเข้ารหัส แบ่งเป็น 2 ส่วน คือ

$L \oplus F(R,K_1)$	R 32 บิท

ข้อมูล R ไม่ถูกเปลี่ยนแปลงและใช้เป็น Input ของ Function_F เพื่อทำการถอดรหัส ขอให้สังเกตุว่าการเข้าและถอดรหัส Function_F ยังคงใช้ Input R และ K_1 เหมือนกันดังนั้นค่า $F(R,K_1)$ ในการเข้าและถอดรหัสจะเท่ากัน

ดังนั้นจากที่กล่าวมาแล้วจะเห็นว่า ข้อมูลเข้าของการทำการเข้ารหัสและข้อมูลออกของการทำการถอดรหัสจะเหมือนกันนั้นหมายถึง เมื่อใช้หมายเลขกุญแจเพียงชุดเดียวและทำรอบเดียวก็ยังคงรักษาคุณลักษณะของการเข้าและถอดรหัสได้แต่ไม่ซับซ้อนมาก

3.4 การเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง

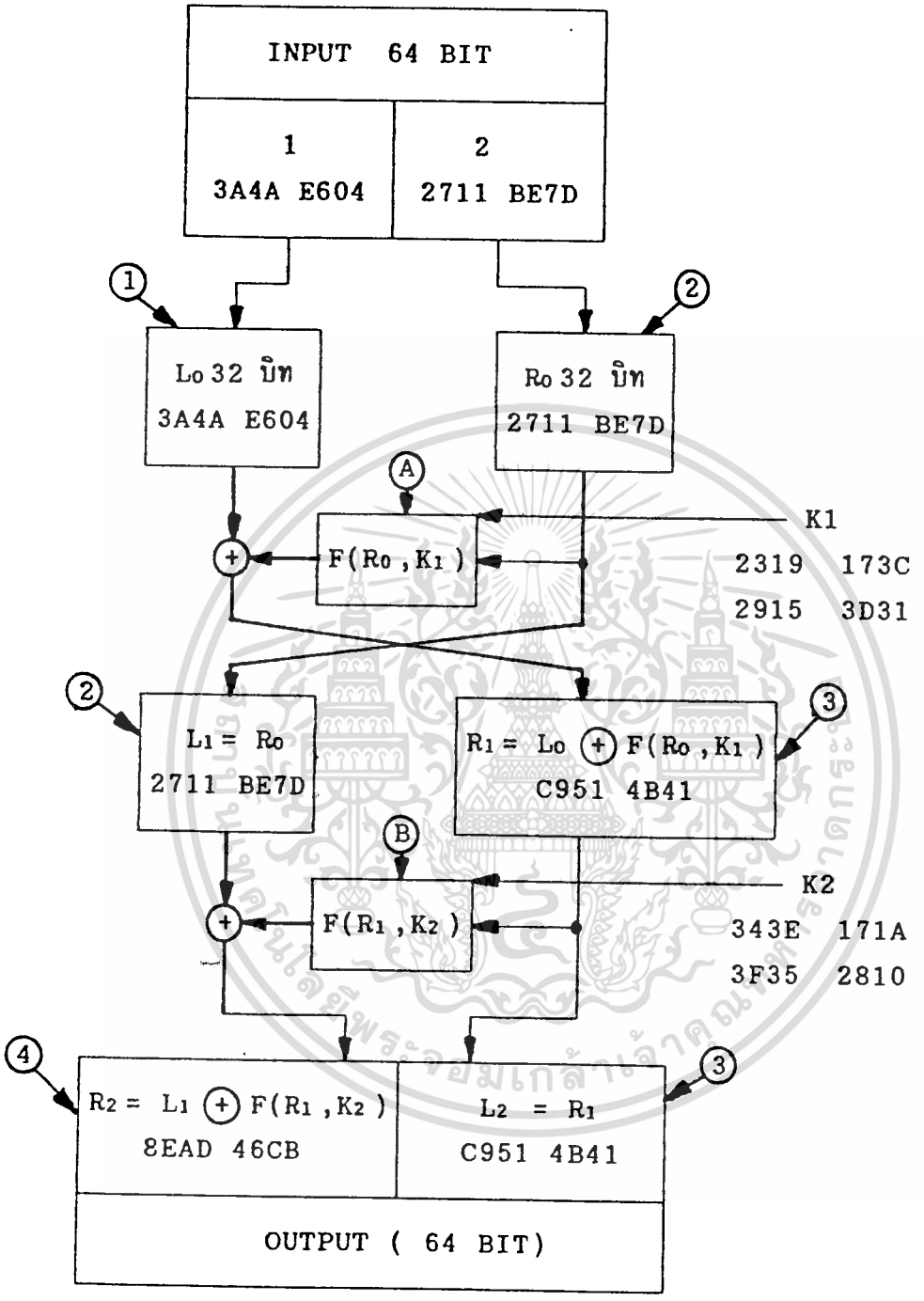
การใช้ Function_F 2 ครั้ง ใช้หลักการเดียวกับการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้งที่ได้กล่าวมาแล้วมีข้อแตกต่างคือ หลังจากทำ Function_F 1 ครั้งข้อมูล L และ R ถูกสลับที่กันโดยให้ $L_1 = R_0$ และ $R_1 = L_0 \oplus F(R_0, K_1)$ หลังจากนั้นเข้าสู่ขบวนการทำ Function_F ครั้งที่ 2 ดังรูปที่ 16

เมื่อนำข้อมูลที่ผ่านการเข้ารหัส 4 และ 3 โดยทำ Function_F 2 ครั้ง แล้วมาเข้าขบวนการถอดรหัส ตามรูปที่ 17 โดยให้ข้อมูลจุดที่ 5 = 4 และ 6 = 3 ขบวนการถอดรหัสจะทำย้อนกลับกับขบวนการในการเข้ารหัสให้สังเกตุที่การเข้ารหัสหมายเลขกุญแจจะเริ่มจาก K_1 และ K_2 ส่วนการถอดรหัสหมายเลขกุญแจจะเริ่มจาก K_2 และ K_1

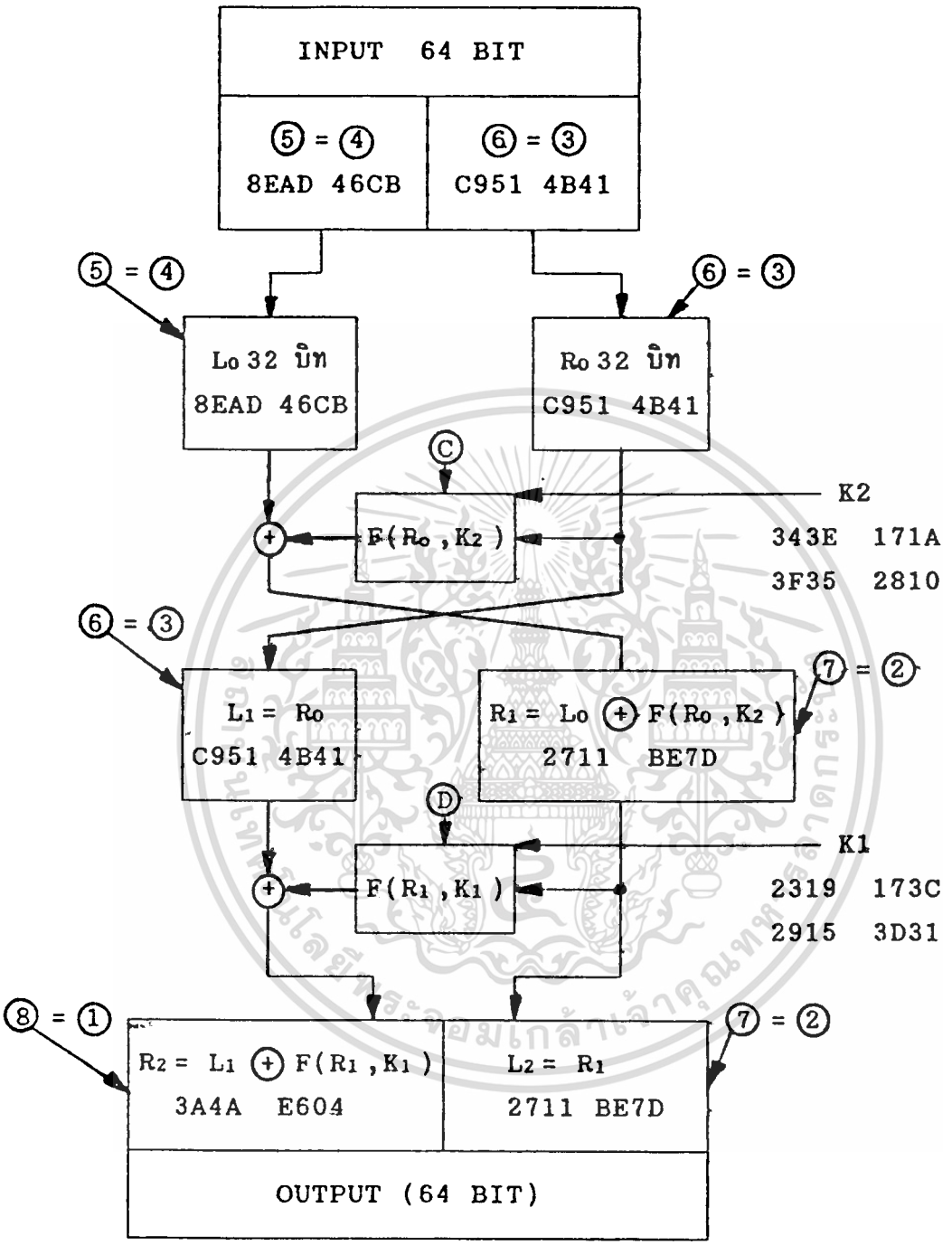
รูปที่ 16 และ 17 ในรอบแรก Input ที่จุด C จะเท่ากับ Input ที่จุด B คือ 3 = 6 และ K_2 จะได้ผลลัพธ์ Function ที่เท่ากัน เมื่อผลลัพธ์ของ 5 $\oplus$ C ได้ 7 ข้อมูลจุดที่ 7 จะเท่ากับ 2 ซึ่งเป็นการทำขบวนการย้อนกลับในรอบแรกของการถอดรหัส โดยใช้หลักของการทำ XOR 2 ครั้งจะได้ค่าเดิม ดังเสนอไปแล้วนั้น

ในรอบที่สอง Input ที่จุด D จะเท่ากับจุด A คือ 2 = 7 และ K_1 ซึ่งจะให้ผลลัพธ์ของ Function ที่เท่ากันเนื่องจาก 7 = 2 และใช้ K_1 ชุดเดียวกันผลลัพธ์ของ Function ที่จุด D $\oplus$ 6 ได้ผลลัพธ์เท่ากับ 8 ซึ่ง 8= 1 โดยใช้หลักของการทำ XOR กับข้อมูลเดิม 2 ครั้ง

การแสดงการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง ที่ได้นำเสนอแล้วนั้นเป็นการแสดงให้เห็นถึงวิธีการเข้าและ ถอดรหัสเพื่อให้เข้าใจได้ง่ายและไม่ยุ่งยากซับซ้อนมากนั้น



รูปที่ 16 การเข้ารหัสโดยทำ Function_F 2 ครั้ง



3.5 การเข้าและถอดรหัสโดยทำ Function_F หลายครั้ง

ในระบบของ DES จะใช้หลักการเข้าและถอดรหัสถึง 16 รอบโดยใช้หมายเลขกุญแจ 16 ชุด การเข้ารหัสต้องใช้หมายเลขกุญแจตั้งแต่ K_1 ถึง K_{16} แต่การถอดรหัสจะใช้ K_{16} ถึง K_1 ซึ่งเป็นขบวนการย้อนกลับกับการเข้ารหัสความสลับซับซ้อนจะมากแต่ยังใช้หลักการเดียวกับการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง.

การกำหนดจำนวนรอบในการเข้าและถอดรหัส สามารถกำหนดให้มีมากกว่าที่ DES กำหนดนั้นหมายถึงเป็นการเพิ่มความสลับซับซ้อนให้กับระบบ

3.6 ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ

ตามรูปที่ 4 การทำงานของ Function_F และ S_Box ตามหลักการของ DES ตาราง E รับ Input ขนาด 32 บิทและให้ Output ขนาด 48 บิท เหตุที่ต้องทำให้เป็น 48 บิท คือต้องทำให้อยู่ในรูปแบบเดียวกับหมายเลขกุญแจ (48 บิท) ซึ่งจะอธิบายในโอกาสต่อไป ความสัมพันธ์ระหว่างตาราง E, S_Box และตาราง P นั้นไม่มี ขอให้คำนึงเสมอว่าการทำงานของ Function_F นั้นทำจากบนลงล่างไม่มีการทำย้อนกลับ สามารถกำหนดตาราง E, S_Box และ P ใหม่โดยยึดหลักว่า

ตาราง E : ต้องมี Output ขนาด 48 บิท ซึ่งต้องมีบิทซ้ำ 16 บิท เนื่องจากรับ Input ขนาด 32 บิทและให้ Output ขนาด 48 บิท

: ตำแหน่งบิทต่าง ๆ สามารถกำหนดขึ้นเองตามความพอใจเช่นดังตารางที่ 13 ดู 3.1.3 ประกอบ

ขบวนการ S_Box

ตามขบวนการ DES S_Box รับ Input ขนาด 6 บิท 8 ชุด ซึ่งเท่ากับ 48 บิทมาเทียบในตาราง S1 ถึง S8 ได้ Output 4 บิท 8 ชุด ซึ่งเท่ากับ 32 บิท ขบวนการทำงานมีดังนี้

รับ Input ขนาด 6 บิท เช่น 010011 (บิท 1 2 3 4 5 6) ให้ Row = บิท 1,6 ให้ Column = บิท 2,3,4,5 จะได้ Row = 01 = 1 และ Column 1001 = 9 เทียบในตาราง S1 จะได้ 6 = 0110 ซึ่งมีขนาด 4 บิท ตามตารางที่ 14 ตารางที่ 13 แสดงการคำนวณตามขบวนการทำ S_box ของ

ตารางที่ 7 โดยให้ Input = 07 17 35 1F 1E 29 32 0B

จะได้ Output = 4A E9 99 F3

ตารางที่ 13 ตาราง E ใหม่

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

ตารางที่ 14 ตาราง S1 ตามหลักของ DES

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

เมื่อทราบถึงการทำงานของ S_Box อีกทั้งขบวนการของ Function_F ถือเป็นเสมือน Function หนึ่งซึ่งไม่มีการทำขบวนการย้อนกลับดังนั้นการกำหนดขบวนการของ S_Box ให้แตกต่างไปจากขบวนการของ DES กระทำได้หลายวิธี เช่น

ถ้ากำหนดให้ Row = บิท 5,1 และให้ Column = บิท 6,2,4,3 หรือให้ Row = บิท 1,2 และให้ Column = บิท 3,4,5,6 ย่อมกระทำได้

ตาราง S1...S8 สามารถกำหนดใหม่ให้แตกต่างไปจาก DES ดังนี้

S1

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

S2

.

.

.

S8

.

.

.

ตารางที่ 15 แสดงการคำนวณตามขบวนการทำ S_Box

ตาราง	I/P(6บิท)	เลขฐาน 2	Row	Col	R/C	O/P(4บิท)
S1	07	00 0111	01	0011	13	4
S2	17	01 0111	01	1011	1B	A
S3	35	11 0101	11	1010	3A	E
S4	1F	01 1111	01	1111	1F	9
S5	1E	01 1110	00	1111	0F	9
S6	29	10 1001	11	0100	34	9
S7	32	11 0010	10	1001	29	F
S8	0B	00 1011	01	0101	15	3

ตาราง P รับ Input ขนาด 32 บิท สลับสับเปลี่ยนตำแหน่งตามรูปแบบของตาราง P และได้ Output ขนาด 32 บิทการกำหนดตาราง P ให้แตกต่างไปจาก DES ทำได้เช่น ดังตารางที่ 16 ดู 3.1.1 ประกอบ

ตารางที่ 16 ตาราง P ใหม่

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

การกำหนดตาราง E, S_Box และ P ขึ้นใหม่กระทำได้ดังที่ได้กล่าวมาแล้วการเข้า และ ถอดรหัสจะได้ข้อมูลที่ถูกต้องต่อเมื่อการเข้า และ ถอดรหัสใช้ตารางและวิธีการเดียวกัน

3.7 การคำนวณหมายเลขกุญแจโดยใช้ขนาด 112 บิท

จุดประสงค์ของการใช้หมายเลขกุญแจขนาด 112 บิท คือ จะทำให้ผู้พยายามที่จะจารกรรมข้อมูลใช้เวลานานในการค้นหาหมายเลขกุญแจที่ถูกต้อง การพยายามในการค้นหาหมายเลขกุญแจที่ถูกต้องไม่มีทางอื่น นอกจากต้องลองทุกๆหมายเลขกุญแจที่เป็นไปได้ กุญแจขนาด 112 บิทจะต้องใช้ความพยายามในการถอดรหัสถึง 2^{112} หรือประมาณ 5.3×10^{33} ซึ่งประมาณ 5 พันล้านล้านล้านล้าน ครั้ง ซึ่งเมื่อเปรียบเทียบกับระบบ DES จะใช้ 2^{56} ซึ่งประมาณ 7.6×10^{16} หรือประมาณ 7 หมื่นล้านล้าน ครั้ง การกระจายหมายเลขกุญแจขนาด 112 บิท ออกเป็นหมายเลขกุญแจ 16 ชุดหรือมากกว่ากระทำได้ในหลายรูปแบบ ซึ่งขึ้นอยู่กับความประสงค์ของผู้ออกแบบ ขอนำเสนอการคำนวณหมายเลขกุญแจขนาด 112 บิทดังนี้

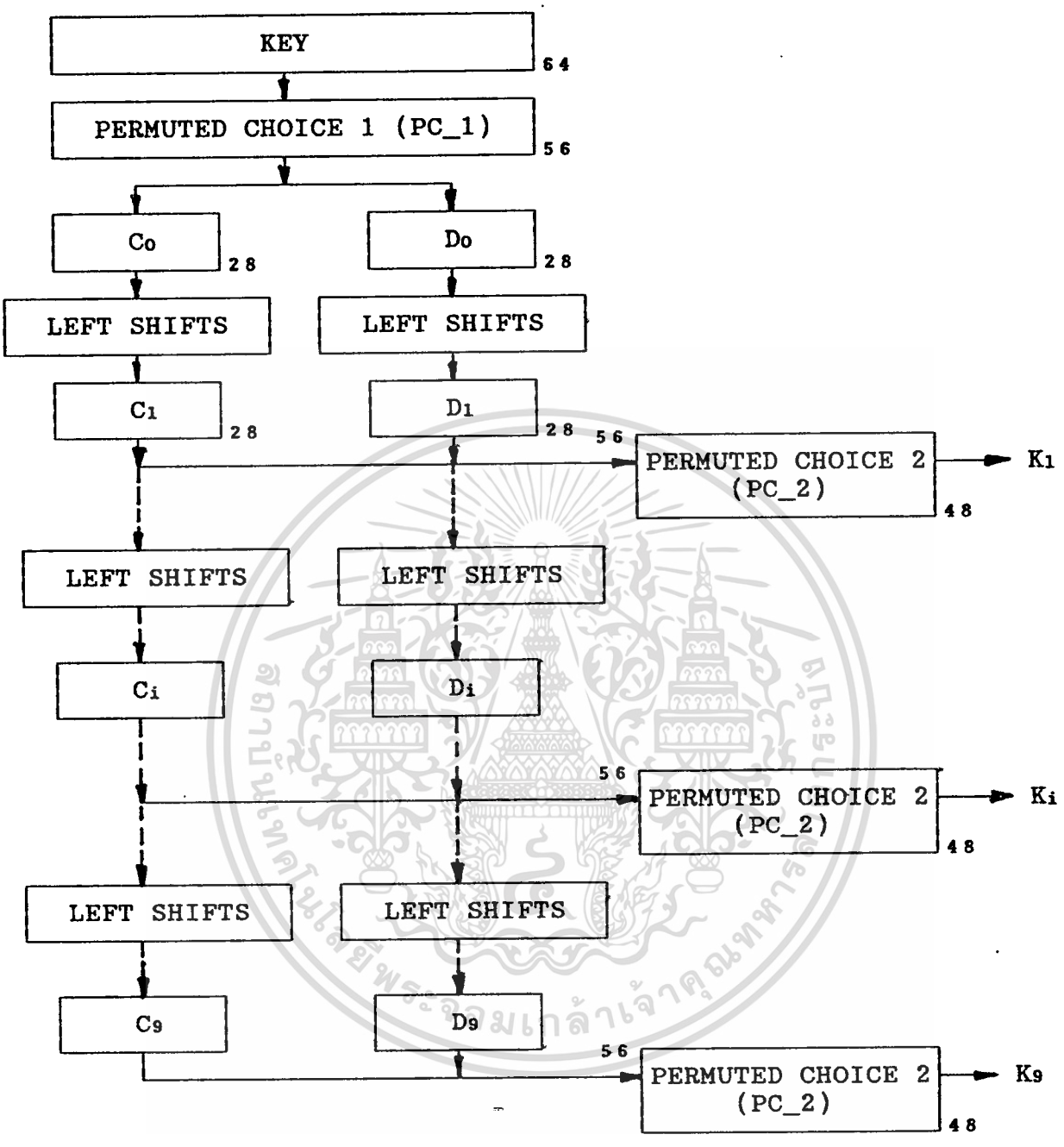
แบ่งหมายเลขกุญแจขนาด 112 บิท ออกเป็น 2 ส่วน ๆ ละ 56 บิท นำหมายเลขกุญแจขนาด 56 บิทมาเข้าขบวนการคำนวณหมายเลขกุญแจตามหลักของ DES โดยออกแบบ Permuted choice 1, Permuted choice 2 และตารางการหมุนกุญแจใหม่ และกระจายหมายเลขกุญแจขนาด 112 บิท ออกเป็นหมายเลขกุญแจ 18 ชุด ดู 3.1.2 ประกอบ

ตารางที่ 17 New PC_1

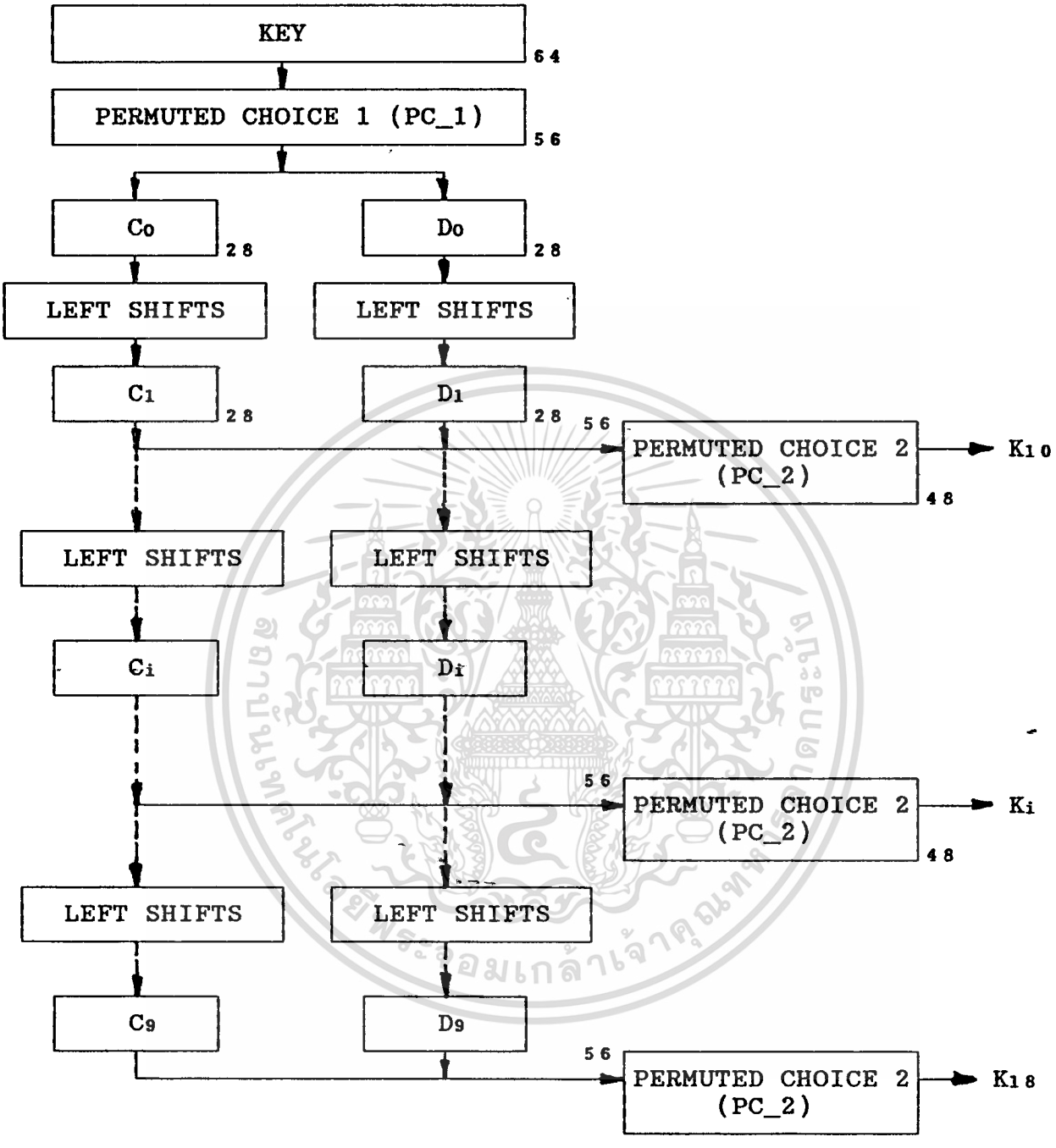
6	22	60	14	49	62	9
5	23	13	1	47	57	51
21	33	50	12	52	2	19
11	37	30	20	29	53	44
55	31	10	15	46	4	25
17	38	61	28	42	39	18
3	36	29	41	63	43	45
54	34	7	35	27	58	26

ตารางที่ 18 New PC_2

16	32	46	4	40	24
38	17	23	49	25	5
3	34	6	2	54	15
22	8	35	42	48	50
9	41	1	43	14	27
56	21	30	31	55	53
37	10	44	19	51	28
11	36	12	45	29	18



รูปที่ 18 แสดงการคำนวณหมายเลขกุญแจ K_1 ถึง K_9 โดยใช้ตารางการหมุนที่ 19



รูปที่ 19 แสดงการคำนวณหมายเลขกุญแจ K_{10} ถึง K_{18} โดยใช้ตารางการหมุนที่ 20

กำหนดตำแหน่งต่าง ๆ ของ New PC_1 ตามตารางที่ 17 โดยตัดส่วนที่เป็น Parity bit ออก 8, 16, 24, 32, 40, 48, 56 และ 64 ออก

กำหนดตำแหน่งต่าง ๆ ตามหลักการของ DES สำหรับ PC_2 ต้องกำหนดบิทที่ไม่ใช้ขึ้น 8 ตัว เนื่องจาก Input ของ PC_2 มีขนาด 56 บิทและ Output มีขนาด 48 บิท กำหนดบิทที่ไม่ใช้ เช่น 7,13,20,26,33,39,47 และ 52 กำหนดตำแหน่งต่าง ๆ ของ New PC_2 ดังตารางที่ 18 ดู 3.1.2 ประกอบ

ตารางที่ 19 แสดงตารางการหมุนซ้าย สำหรับสร้างหมายเลขกุญแจชุดที่ 1 ถึง 9 ของกุญแจ 56 บิท ชุดที่ 1

	ลำดับ	No Left Shifts
56 บิท ชุดที่ 1	1	2
	2	4
	3	4
	4	4
	5	3
	6	1
	7	4
	8	2
	9	4

ตารางที่ 20 แสดงตารางการหมุนซ้าย สำหรับสร้างหมายเลขกุญแจชุดที่ 10 ถึง 18 ของกุญแจ 56 บิท ชุดที่ 2

	ลำดับ	No Left Shifts
56 บิท ชุดที่ 2	10	3
	11	4
	12	5
	13	2
	14	5
	15	4
	16	1
	17	3
	18	1

หลักการทำงาน

รูปที่ 18 นำข้อมูล 56 บิทชุดที่ 1 มาเข้าขบวนการสลับสับเปลี่ยน New PC_1 ตามตารางที่ 17 ข้อมูลนี้ได้ถูกแบ่งเป็นสองส่วน Co และ Do นำ Co และ Do มาหมุนทางซ้ายครั้งที่ 1 ตามตารางที่ 19 ข้อมูลที่ได้เป็น C_1และ D_1 ข้อมูลนี้ถูกป้อนเข้าขบวนการสลับสับเปลี่ยน New PC_2 ตามตารางที่ 18 ผลลัพธ์เป็นหมายเลขกุญแจชุดที่ 1 (K_1) นำข้อมูล C_1 และ D_1 มาหมุนซ้ายครั้งที่ 2 ตามตารางที่ 19 ข้อมูลที่ได้เป็น C_2 และ D_2และถูกป้อนเข้า New PC_2 ตามตารางที่ 18 เพื่อให้ข้อมูลออกเป็นหมายเลขกุญแจชุดที่ 2 (K_2) ขบวนการนี้จะทำไปจนกว่าจะครบหมายเลขกุญแจชุดที่ 9 (K_9)

การหาหมายเลขกุญแจตามรูปที่ 19 K_{10} ถึง K_{18} กระทำโดยใช้ข้อมูลกุญแจ 56 บิทชุดที่ 2 มาทำขบวนการเช่นเดียวกับการหา K_1 ถึง K_9 และตารางการหมุนใช้ตารางที่ 20

หลังจากทำตามขบวนการตามรูปที่ 18 และ 19 จะได้หมายเลขกุญแจ K_1 ถึง K_{18} เพื่อใช้ในขบวนการเข้าและถอดรหัสต่อไป

ตัวอย่างการคำนวณหมายเลขกุญแจขนาด 112 บิท

กำหนดให้ หมายเลขกุญแจชุดที่ 1 เท่ากับ d465 9cae 367c d9eb

และ ชุดที่ 2 เท่ากับ 748c 238a bcef 9876

นำหมายเลขกุญแจชุดที่ 1 มาทำการสลับสับเปลี่ยน PC_1 (ตารางที่ 15) ได้ผลลัพธ์เท่ากับ ee35 33e0 76d6 b170

ข้อมูลนี้ถูกแบ่งเป็นสองส่วนคือ Co = ee35 33e0

D0 = 76d6 b170

ขั้นตอนต่อไปนำ Co และ Do มาหมุนตามตารางที่ 17 ซึ่งกำหนดให้

	Co = ee35 33e 0	Do = 76d6 b17 0
ลำดับที่ 1 หมุน 2 ครั้ง	Co = b8d4 cfb 0	Do = db5a c5d 0
ลำดับที่ 2 หมุน 4 ครั้ง	Co = 8d4c fbb 0	Do = b5ac 5dd 0
ลำดับที่ 3 หมุน 4 ครั้ง	Co = d4cf bb8 0	Do = 5ac5 ddb 0
ลำดับที่ 9 หมุน 1 ครั้ง	Co = ee35 33e 0	Do = 76d6 b17 0

นำข้อมูลหลังจากการหมุนในแต่ละลำดับเข้าขบวนการสลับสับเปลี่ยน PC_2 ได้เป็นกุญแจ K1 ถึง K9 ดังตารางที่ 21 กุญแจ 9 ชุดนี้ใช้ในขบวนการเข้าและถอดรหัสต่อไป จบขบวนการคำนวณหมายเลขกุญแจชุดที่ 1

ตารางที่ 21 แสดงตารางการคำนวณหมายเลขกุญแจ K1 ถึง K9

ลำดับ	Co	Do	#KEY(K)
1	b8d4cfb0	db5ac5d0	1f3b22293f39111f
2	8d4cfbb0	b5ac5dd0	191f1a171b353513
3	d4cfbb80	5ac5ddb0	3d3e0d0f2a3b3c04
4	4cfbb8d0	ac5ddb50	2a371f0630341f2e
5	67ddc6a0	62eedad0	283e2e3e372d320d
6	cfbb8d40	c5ddb5a0	23311d37381b283e
7	fbb8d4c0	5ddb5ac0	1e3736323c092a39
8	eee35330	776d6b10	392d3d0c392c1b31
9	ee3533e0	76d6b170	332b3e0e0f2e242c

นำหมายเลขกุญแจชุดที่ 2 มาทำการสลับสับเปลี่ยน PC_1 ได้ผลลัพธ์เท่ากับ

4a78 ee20 789b 99f0

ข้อมูลนี้ถูกแบ่งเป็นสองส่วนคือ Co = 4a78 ee20

Do = 789b 99f0

ขั้นตอนต่อไปนำ Co และ Do มาหมุนตามตารางที่ 18 ซึ่งกำหนดให้

Co = 4a78 ee2 0 Do = 789b 99f 0

ลำดับที่ 10 หมุน 3 ครั้ง Co = 53c7 712 0 Do = c4dc cfb 0

ลำดับที่ 11 หมุน 4 ครั้ง Co = 3c77 125 0 Do = 4dcc fbc 0

ลำดับที่ 12 หมุน 5 ครั้ง Co = 8ee2 4a7 0 Do = b99f 789 0

ลำดับที่ 18 หมุน 1 ครั้ง Co = 4a78 ee2 0 Do = 789b 99f 0

นำข้อมูลหลังจากการหมุนในแต่ละลำดับเข้าขบวนการสลับสับเปลี่ยน PC_2 ได้เป็นกุญแจ K10 ถึง K18 ดังตารางที่ 22 กุญแจ 9 ชุดนี้รวมกับกุญแจ 9 ชุดข้างต้นจะใช้ในขบวนการเข้าและถอดรหัสต่อไป

ตารางที่ 22 แสดงตารางการคำนวณหมายเลขกุญแจ K10 ถึง K18

ลำดับ	Co	Do	#KEY(K)
10	53c77120	c4dccfb0	2f241515332b3607
11	3c771250	4dccfbc0	2c2d3b061209333c
12	8ee24a70	b99f7890	1a0d09063d353933
13	3b8929e0	e67de260	2f23321c311e0e06
14	71253c70	cfbc4dc0	2e04363d13192532
15	1253c770	fbc4dcc0	3d3c032f010d311f
16	24a78ee0	f789b990	301e392a333d2836
17	253c7710	bc4dccf0	192c3a3512270d2f
18	4a78ee20	789b99f0	121d0622153f3c2d

3.8 สรุป

ที่ได้นำเสนอไปแล้วนั้นกล่าวถึงการทดลอง, วิเคราะห์ของ

1. การทำการสลับสับเปลี่ยนที่ใช้ในตาราง IP,IP^{-1},E,P,PC-1, PC-2 และการทำ Function_F
2. ความสัมพันธ์ของตารางต่าง ๆ และเปลี่ยนค่าในทุก ๆ ตาราง ให้แตกต่างไปจาก DES
3. เพิ่มจำนวนรอบในการทำ Function_F จาก 16 เป็น 18 รอบ
4. เพิ่มหมายเลขกุญแจจาก 16 ชุดเป็น 18 ชุด

ประโยชน์ที่จะได้รับคือจะได้เทคนิคใหม่ที่แตกต่างไปจาก DES และเป็นเอกลักษณ์อีกทั้งการเปลี่ยนขนาดกุญแจจาก 56 บิทเป็น 112 บิททำให้การจารกรรมข้อมูลใช้เวลานานขึ้น

ผนวก ง. ตัวอย่างข้อมูลการเข้าและถอดรหัสและการคำนวณหมายเลขกุญแจด้วยวิธีการเข้าและถอดรหัสที่สร้างขึ้นใหม่

บทที่ 4
ผลการทดลอง

4. การทดลองที่ 1

ทดลองการเข้ารหัสโดยใช้เครื่องไมโครคอมพิวเตอร์ Acer 1120SX CPU 386SX ความเร็ว 19.4 Mhz เขียนโปรแกรมด้วยภาษา C บน Turbo C Version 1.5 พบว่าเวลาในการเข้ารหัส 1 ครั้ง ของข้อมูล 64 บิท เท่ากับ 0.021923 วินาที การพยายามในการจารกรรมข้อมูลไม่มีทางอื่น นอกจากต้องลองทุก ๆ หมายเลขกุญแจที่เป็นไปได้

กุญแจ 2^{56} ประมาณ 7.20576×10^{16} เท่ากับ 7 หมื่นล้านล้าน หมายเลขกุญแจที่เป็นไปได้ ดังนั้นเวลาเฉลี่ยที่ใช้ในการหาหมายเลขกุญแจจะเท่ากับ

$$\frac{2^{56} \times \text{เวลาของการเข้ารหัส 1 ครั้ง (วินาที)}}{60(\text{นาที}) \times 60(\text{ชั่วโมง}) \times 24(\text{วัน}) \times 365(\text{ปี}) \times 2(\text{ค่าเฉลี่ย})} \quad \text{ปี}$$

เวลาเฉลี่ยในการหาหมายเลขกุญแจพบประมาณ 25 ล้านปี

กุญแจ 2^{112} ประมาณ 5.19229×10^{33} เท่ากับ 5 พันล้านล้านล้านล้านล้าน หมายเลขกุญแจที่เป็นไปได้ ดังนั้นเวลาเฉลี่ยที่ใช้ในการหาหมายเลขกุญแจจะเท่ากับ 36 ล้านล้านล้านล้าน ปี

ตารางที่ 23 แสดงเวลาการเข้ารหัส 1 ครั้ง

Acer 1120SX 386SX 19.4 Mhz	PS/1 486SX 50 Mhz
0.0219230 วินาที	0.007857 วินาที

4.1 การทดลองที่ 2

คำสั่ง encrypt

encrypt {[-e],[-d]} {Hexkey} {Inputfile} {Outputfile}

โปรแกรมพัฒนาขึ้นด้วยภาษา C ตามรายละเอียดในบทที่ 3 ประกอบด้วยคำสั่งเข้ารหัส (encrypt) หรือถอดรหัส (decrypt) และแฟ้มข้อมูล 3 ชุด ดังมีรายละเอียดดังนี้

1. ชุดคำสั่ง เป็นการสั่งให้โปรแกรมทำการ เข้ารหัส หรือ ถอดรหัส มีรูปแบบเป็น [-e],[-d]

2. กุญแจ (Hexkey) มีขนาด 128 บิท เป็น Parity bit 16 ตัว ใช้จริง 112 บิท แฟ้มข้อมูลนี้กำหนดขึ้นได้โดยใช้ Editor ตัวใดตัวหนึ่งและเขียนกุญแจได้เป็น

AB59 56EC 1209 46D2

2740 ACE5 355A 2344

3. แฟ้มข้อมูลเข้า (Inputfile) เป็นแฟ้มข้อมูลในการเข้าหรือถอดรหัส

4. แฟ้มข้อมูลออก (Outputfile) เป็นแฟ้มข้อมูลหลังการทำการเข้าหรือถอดรหัสแล้ว

ตารางที่ 24 แสดงเวลาการเข้ารหัสแฟ้มข้อมูล

แฟ้มข้อมูล	ขนาด	Acer 1120SX 386SX 19.4 Mhz	PS/1 486SX 50 Mhz
CHAI.EXE	25613 bytes	67.7472 วินาที	26.153 วินาที
PCTOOLS.EXE	171003 bytes	454.3956 วินาที 7.57 นาที	174.2857 วินาที 2.9 . นาที

4.2 การทดลองที่ 3

เปรียบเทียบตัวอย่างแฟ้มข้อมูลของการเข้าและถอดรหัสด้วยวิธีการของ DES(enc56.exe) และวิธีการที่สร้างขึ้นใหม่(encrypt.exe)

เริ่มการทดลองด้วยวิธีการของ DES(enc56.exe)โดยการเข้ารหัสข้อมูลใดข้อมูลหนึ่ง(แฟ้มTest.t) ด้วยหมายเลขกุญแจ key1.t ให้ข้อมูลหลังการเข้ารหัสแล้วอยู่ในแฟ้มข้อมูล sc1.t ด้วยคำสั่งที่ 1 ของตารางที่ 25 ต่อไปถอดรหัสแฟ้มข้อมูล sc1.t ด้วย key ตัวเดิม (key1.t) ด้วยคำสั่งที่ 2 ของตารางที่ 25 ให้ข้อมูลหลังการถอดรหัสแล้วอยู่ในแฟ้มข้อมูล sc2.t ณ.จุดนี้ถ้าการเข้าและถอดรหัสถูกต้อง test.t และ sc2.t ต้องมีข้อมูลเหมือนกัน 100 % ต่อไปทดลองการถอดรหัสแฟ้มข้อมูล sc2.t ด้วยหมายเลขกุญแจที่ต่างกันดังตารางที่ 28 ด้วยคำสั่งที่ 3,4,5,6 ของตารางที่ 25

ตารางที่ 25 แสดงชุดคำสั่งเข้าและถอดรหัสโดยใช้วิธีการของ DES

คำสั่งที่	คำสั่งเข้าและถอดรหัสโดยใช้วิธีการของ DES			
		แฟ้มกุญแจ	แฟ้มเข้า	แฟ้มออก
1	enc56 -e	key1.t	test.t	sc1.t
2	enc56 -d	key1.t	sc1.t	sc2.t
3	enc56 -d	key2.t	sc1.t	sc3.t
4	enc56 -d	key3.t	sc1.t	sc4.t
5	enc56 -d	key4.t	sc1.t	sc5.t
6	enc56 -d	key5.t	sc1.t	sc6.t

ทดลองลักษณะเดิมด้วยวิธีการที่สร้างขึ้นใหม่ (encrypt.exe) ด้วยคำสั่งดังตารางที่ 26

ตารางที่ 26 แสดงชุดคำสั่งเข้าและถอดรหัสโดยใช้วิธีการที่สร้างขึ้น

คำสั่งที่	คำสั่งเข้าและถอดรหัสโดยใช้วิธีการที่สร้างขึ้น			
		แฟ้มกุญแจ	แฟ้มเข้า	แฟ้มออก
1	encrypt	-e key1.t	test.t	xc1.t
2	encrypt	-d key1.t	xc1.t	xc2.t
3	encrypt	-d key2.t	xc1.t	xc3.t
4	encrypt	-d key3.t	xc1.t	xc4.t
5	encrypt	-d key4.t	xc1.t	xc5.t
6	encrypt	-d key5.t	xc1.t	xc6.t

เปรียบเทียบข้อมูลออกของทั้งสองวิธีด้วยโปรแกรม Compare.exe ซึ่งเป็นการเปรียบเทียบแฟ้มข้อมูลเข้าและออกตำแหน่งต่อตำแหน่งว่า แฟ้มทั้งสองมีตัวอักขระกี่ตำแหน่งที่ตรงกัน ตัวอย่างเช่น

คำสั่ง c:>compare test.t sc1.t

ข้อมูลออกเป็น

```
Compare  test.t  and  sc1.t

          Number of word              1144
          Number of the same word        8
          Percentage                   0.7 %
```

แสดงว่าแฟ้ม test.t และ sc1.t มีอักขระทั้งหมด 1144 ไบท์ มีตัวอักขระที่เหมือนกัน 8 ตัว คิดเป็น 0.7 % ดังตารางที่ 27

ตารางที่ 27 แสดงผลการเปรียบเทียบจากตัวอย่างแฟ้มข้อมูลของการเข้าและถอดรหัสด้วยวิธีการของ DES และ วิธีการที่สร้างขึ้นใหม่

ลำดับ	เปรียบเทียบแฟ้ม sXX.t เกิดจากวิธีการของ DES	ผลจากวิธีการ DES (%)	เปรียบเทียบแฟ้ม xXX.t เกิดจากวิธีการที่สร้างขึ้น	ผลจากวิธีการที่สร้างขึ้น (%)
1	test.t sc1.t	0.7	test.t xc1.t	0.5
2	test.t sc2.t	100	test.t xc2.t	100
3	test.t sc3.t	0.3	test.t xc3.t	0.3
4	test.t sc4.t	0.3	test.t xc4.t	0.4
5	test.t sc5.t	0.5	test.t xc5.t	0.2
6	test.t sc6.t	0.5	test.t xc6.t	0.3

ตารางที่ 28 แสดงจำนวนตำแหน่งที่แตกต่างของแฟ้มกุญแจที่ใช้ในการทดลอง

ชื่อแฟ้มกุญแจ	แตกต่างกับ key1.t (ตำแหน่ง)
key1.t	-
key2.t	1
key3.t	2
key4.t	4
key5.t	ทุกตำแหน่ง

ผลจากการทดลองที่ 3

จากตารางที่ 27 ลำดับที่ 1 แฟ้ม sc1.และ xc1.t เป็นแฟ้มที่เข้ารหัสด้วย key1.t โดยมีแฟ้ม test.t เป็นข้อมูลเข้า การเปรียบเทียบระหว่างแฟ้ม test.t กับ sc1.t เท่ากับ 0.7 % และการเปรียบเทียบระหว่างแฟ้ม test.t กับ xc1.t เท่ากับ 0.5 %

ลำดับที่ 2 แฟ้ม sc2.t และ xc2.t เกิดจากการถอดรหัส แฟ้ม sc1.t ด้วย key เดิม คือ key1.t แฟ้ม sc1.t เปรียบเทียบกับ sc2.t และ xc2.t จึงต้องมีข้อมูลเดียวกัน และผลการเปรียบเทียบได้ 100 % ลำดับที่ 3,4, 5,6, แฟ้ม sc3.t,sc4.t,sc5.t,sc6.t และแฟ้ม xc3.t,xc4.t,xc5.t, xc6.t ได้จากการถอดรหัส จากแฟ้มข้อมูล sc1.t ด้วยกุญแจที่ต่างกันดังตารางที่ 28 คือการถอดรหัสโดยใช้ key ที่ไม่ใช่ตัวเดิม ผลการเปรียบเทียบจึงควรมีข้อมูลที่ไม่เหมือนกัน

จึงสรุปได้ว่าการถอดรหัสที่ใช้รหัสกุญแจที่ไม่ใช่ตัวเดิมจะไม่สามารถอ่านข้อมูลได้ การทดลองเพิ่มเติมโดยใช้ key ที่ไม่เหมือนกันประมาณ 50 ชุดผลคือ ไม่สามารถถอดรหัสได้ การทดลองนี้ไม่ได้ทดลองกับกุญแจทุก ๆ หมายเลขกุญแจที่เป็นไปได้เพราะกุญแจขนาด 2^{112} บิทต้องใช้เวลาทดลองทั้งหมดประมาณ 200 ล้าน ๆ ปี ด้วยเครื่องคอมพิวเตอร์ที่เร็วที่สุดในปัจจุบัน ผู้วิจัยจึงไม่รับลองว่าจะไม่มีกุญแจอื่นที่ถอดรหัสและได้ข้อมูลเดิม แต่มีข้อคิดอยู่ว่าเทคนิคที่พัฒนาขึ้นนี้พัฒนามาจาก DES ซึ่งมีการประกาศใช้ในปี 2519 ถึงปัจจุบัน ยังไม่มีผู้ใดพบสิ่งบกพร่องนี้

แฟ้มข้อมูลที่เข้ารหัสหรือถอดรหัสโดยไม่ได้ใช้กุญแจตัวเดิมส่งผลให้ข้อมูลเป็นขยะบ่อยครั้งขยะเหล่านี้ตรงกับอักขระ EOF หรือเท่ากับ 1A เมื่อดูข้อมูลในแฟ้มเหล่านี้โดยใช้ Editor ทั่วไปจะดูได้เท่าที่จะถึง EOF ตัวแรก ข้อมูลหลัง EOF จะไม่สามารถดูได้ ถ้าต้องการดูข้อมูลทั้งหมดควรใช้ Pctools หรือ NDD

รายละเอียดของแฟ้ม key1.t ถึง key5.t และแฟ้มtest.t,sc3.t ถึง sc6.t และ xc3.t ถึง xc6.t แสดงในผนวก จ. แฟ้มข้อมูลเหล่านี้ไม่สามารถแสดงให้เห็นได้หมดเพราะมีอักขระ EOF มาก

บทที่ 5
สรุป

5. สรุป

Diffie และ Hellman[1] คาดการไว้ว่าจะมีเครื่องมือสำหรับตรวจหาหมายเลขกุญแจที่สามารถถอดรหัสของ DES IC พิเศษที่ผลิตขึ้นหนึ่งล้านตัวจะมีค่าประมาณ 20 ล้านเหรียญซึ่งสามารถถอดรหัสที่ใช้วิธีการของ DES ได้ในหนึ่งวัน ยังประมาณการอีกว่าในปี 1990 จะมี Hardware ที่มีความสามารถมากขึ้นและจะทำให้ 56 BIT KEY มีขนาดไม่พอในด้านความปลอดภัย การพัฒนาของขบวนการ DES มีการพัฒนามาโดยตลอดและใช้กันอย่างกว้างขวาง นักค้นคว้าจากทุกมุมโลกให้ความสนใจและแข่งขันกันพัฒนาให้ดีขึ้น[6] DES สร้างขึ้นโดย IBM มีการกล่าวถึงมากว่า 56 Bit Key นั้นไม่พอเพียงสำหรับความปลอดภัยในยุคนี้

ตารางที่ 29 แสดงความแตกต่างระหว่างกุญแจที่มีขนาด 56 บิท และ 112 บิท

จำนวนบิท		จำนวนหมายเลขกุญแจที่เป็นไปได้	เวลาใช้ในการจารกรรมข้อมูล
จำนวนกุญแจเดิม	2^{56}	7 หมื่นล้านล้าน	1 วัน
จำนวนกุญแจที่พัฒนาขึ้น	2^{112}	5 พันล้านล้านล้านล้านล้าน	200 ล้านล้านปี

ถ้าใช้เวลา 1 วันในการถอดรหัส 2^{56} = 7 หมื่นล้านล้าน กุญแจที่เป็นไปได้เวลาใช้ในการถอดรหัสกุญแจ 2^{112}

$$= \frac{2^{112}}{2^{56}} = \frac{2^{56} \cdot 2^{56}}{2^{56}} = 2^{56} \text{ วัน}$$

= 7 หมื่นล้านล้าน วัน หรือ ประมาณ 200 ล้านล้านปี

เทคนิคการเข้าและถอดรหัสที่ถูกพัฒนาขึ้นใหม่ตำแหน่งของบิทใน Permutation IP,IP^{-1}, ตาราง S1-S8, P, E, Permuted Choice 1,2 และตารางการหมุนได้ถูกเปลี่ยนไปหมายเลขกุญแจถูกกำหนดขึ้นใหม่ให้มีความสลับซับซ้อนมากขึ้นโดยใช้ถึง 18 ชุด นั้นหมายถึงต้องทำ Function_F 18 รอบ การเปลี่ยนตารางต่างๆและ Function_F ให้แตกต่างไปจากเดิมทำให้วิธีการแตกต่างไปจาก DES ในเรื่องของตำแหน่งและค่า Function ต่าง ๆ

ตารางที่ 29 กุญแจขนาด 2^{56} จะใช้เวลาในการจารกรรมข้อมูลเพียง 1 วันนั้นไม่พอเพียงสำหรับความปลอดภัยในยุคปัจจุบันกุญแจขนาด 2^{112} จะต้องใช้เวลาในการจารกรรมข้อมูลถึง 200 ล้านล้านปีซึ่งให้ความปลอดภัยสูงเหมาะสำหรับหน่วยงานที่ต้องการรักษาข้อมูลให้เป็นความลับ

ในด้านการทหารการป้องกันการลักลอบดักฟังข้อมูลเป็นเรื่องที่สำคัญและจำเป็นอย่างยิ่ง จึงต้องทำทุกวิถีทางที่จะไม่ให้ฝ่ายศัตรูหรือผู้ไม่หวังดีต่อประเทศชาติได้ล่วงรู้ความลับของทางราชการ วิทยานิพนธ์นี้อธิบายถึงวิธีการสร้างเทคนิคการเข้าและถอดรหัสอย่างละเอียด ซึ่งมีประโยชน์แก่ผู้ที่สนใจนำไปใช้ได้ตามต้องการ สำหรับในส่วนราชการนั้นผู้วิจัยจะนำเทคนิคนี้ไปประยุกต์เปลี่ยนแปลงโครงสร้างบางประการหรือทั้งหมด และปกปิดวิธีการที่ได้เปลี่ยนแปลงนี้เพื่อใช้ในการทหารจะส่งผลให้ไม่มีผู้ใดสามารถลักลอบถอดรหัสได้และเทคนิคใหม่เป็นเอกลักษณ์

5.1 ข้อเสนอแนะ

โปรแกรมที่พัฒนาขึ้นใช้งานได้สมบูรณ์แบบทุกประการ เป็นโปรแกรมที่มีความสลับซับซ้อนสูง ซึ่งต้องใช้เวลาในการทำงาน (Process) มาก จากตารางที่ 24 แสดงให้เห็นถึงเวลาในการเข้าและถอดรหัสของแฟ้มข้อมูล Chai.exe และ Pctools.exe สำหรับเครื่องไมโครคอมพิวเตอร์ 486SX ซึ่งใช้เวลา 26.153 และ 174.2857 วินาที เวลาที่ใช้ขนาดนี้จะทำให้ผู้ใช้รู้สึกอึดอัด ควรปรับปรุงโปรแกรมโดยเขียนด้วยภาษาแอลแซมบลีหรือทำเป็น Hardware ในลักษณะของ card เสียบ จะทำให้ความเร็วดีขึ้น และควรนำเอาเทคนิคของการทำ Data Compression มาผนวกกับการเข้าและถอดรหัสจะทำให้การทำงานเร็ว และน่าใช้ยิ่งขึ้น ซึ่งอาจมีผลในเชิงธุรกิจ

เอกสารอ้างอิง

[1]. Jennifer Seberry, Josef Pieprzyk, " Cryptography An Introduction to Computer Security",Prentice-Hall, 1989.

[2]. H.M. Deitet, " An Introduction to Operating System", Addison-Wesley, 1984, pp. 458-68, 632-46.

[3]. Data Security Through Cryptography ", New York: International Business Machines Corporation GC22-9062-0, 1986.

[4]. James Arlin Cooper,"Computer & Communications Security", McGRAW-HILL,1989, pp.211-212, 240-241, 318-326, 360-361.

[5]. C.E. Shannon. " The Communication Theory of Secrecy System", Bell System Tech, Vol.28 oct, 1949.

[6]. ร.อ. ชัยยศ ลิลิตวงษ์, อุปกรณ์เข้า-ถอดรหัสข้อมูลในข่ายสื่อสารเครื่องคอมพิวเตอร์, วิทยานิพนธ์ ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า พระนครเหนือ,2534

[7]. Charles P.PFLEEGER,"Security in Computing", Prentice-Hall International editions,1989, pp. 22-25, 106-124.

[8]. Martin R. Smith,"Commonsence Computer Security",McGRAW-HILL,1989, PP. 144-150.

[9]. "IBM Cryptographic Subsystem Concepts and Facilities ", New York: International Business Machiness Corporation, 1985.

[10]. ดร.บุญวัฒน์ อัตชู, ร้อยเอก ชัยยง ไชยสิงห์ทอง, " เทคนิคการเข้าและถอดรหัสเพื่อใช้ในด้านการทหาร ", วารสารทางวิชาการของสมาคมคอมพิวเตอร์แห่งประเทศไทย,ฉบับที่ 106 ประจำเดือน มีนาคม-เมษายน 2537

ภาคผนวก ก.

โปรแกรมภาษา C ของระบบการเข้าและถอดรหัสสำหรับกุญแจขนาด 112 บิท

/* des.h เป็นโปรแกรมควบคุมส่วนต่าง ๆ ของการเข้าและถอดรหัส มีรายละเอียดดังนี้ การคำนวณหมายเลขกุญแจ 112 บิท แบ่งเป็น keyinit1 และ keyinit2 การเข้ารหัสเป็นการแปลงข้อมูลขนาด 64 บิท ไปเป็นข้อมูลแบบที่ไม่มีรูปแบบเรียกว่า Cipher ข้อมูลนี้สามารถแปลงกลับเป็นข้อมูลเดิมได้

โปรแกรมเมอร์ ร.อ.ชัยยง ไชยสิงห์ทอง

ได้ดัดแปลงโปรแกรม ซึ่งใช้กุญแจขนาด 56 บิท เป็น 112 บิท และส่วนที่เป็นตารางต่างๆ ดังข้อสรุปในบทที่ 5 จาก Australian Standard AS2805.5-1985 Data Encryption Algorithm เขียนโดย Lawrence Brown <Lpb@csadfa.oz> Dec 1987

รายละเอียด

Keyinit1(key1) - คำนวณกุญแจขนาด 56 บิทชุดที่ 1 Long Key [2]

keyinit2(key2) - คำนวณกุญแจขนาด 56 บิทชุดที่ 2 Long Key [2]

ฟังค์ชั่นทั้ง 2 นี้ จะต้องทำก่อนขบวนการเข้าและถอดรหัส

endes(b) - เป็นฟังค์ชั่นเข้ารหัส โดยร่วมกับรหัสกุญแจ keyinit1(key1),keyinit2(key2) b เป็นข้อมูลขนาด 64 บิท

dedes(b) - เป็นฟังค์ชั่นการถอดรหัสโดยร่วมกับรหัสกุญแจ keyinit1(key1), keyinit2(key2) b เป็นข้อมูลขนาด 64 บิท

ข้อมูลขนาด 64 บิท แทนโดยตัวแปร 2 ตัว ที่ไม่มีเครื่องหมาย (unsign Longwords) การเรียงบิท เป็นดังนี้

```
[ 1 2 3 ....32] [ 33 34 35 ... 64]
     b[0]            b[1]
ข้อมูล L(ซ้าย) b[0] และข้อมูล R(ขวา) b[1]
```

การคำนวณกุญแจ ขนาด 128 บิท แบ่งเป็น 64 บิท 2 ชุด แต่ละชุดมีบิทตรวจสอบข้อผิดพลาด 8.,16,24,32,40,48,56,64

```
**************************************************************/

typedef  unsigned long    long;

extern  keyinit();
extern  endes();
extern  dedes();
```

```
/*
des 64.i ประกอบด้วยตารางการแทนที่ต่าง ๆ ซึ่งได้ดัดแปลงตารางจาก
          "AS2805.5 - 1985  DEA"

ไปรแกรมเมอร์     ร.อ.ชัยยง ไชยสิงห์ทอง
                 ได้ดัดแปลงไปรแกรม จาก     Australian Standard
                 AS2805.5-1985 Data Encryption Algorithm เขียน
                 โดย Lawrence Brown <Lpb@csadfa.oz>   Dec 1987
*/

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


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

```
char E[48]  = {  9, 23,    10, 26, 15, 30,
                 25, 7,     25,  5, 11, 14,
                17, 18,     9, 19,  2, 20,
                14, 15,    27,  6, 12, 29,
                 3,  5,    24, 18,  7, 21,
                28, 12,    13,  1,  4, 31,
                20, 30,     8, 16, 23, 32,
                32, 19,    17, 22,  3, 28    };

 char P[32]  = {   9,  7, 21,  8,
                  10, 18,  1, 23,
                  17,  6, 28, 20,
                  26, 24, 12,  2,
                  16, 25,  5,  4,
                  30, 11, 27, 14,
                   3, 32, 29, 15,
                  31, 22, 13, 19  };
/* PC1[64]  ตารางการสลับสับเปลี่ยน PC1 มีขนาด 64 บิท หักส่วนที่เป็นบิท
            ตรวจสอบข้อผิดพลาด เหลือ 56 บิท    */
char PC1[64]
    = {  10, 25, 11, 18, 45, 21, 43,
         54, 55, 19, 36,  2,  9, 57,
         12, 20, 37,  3, 58, 22, 23,
         13, 59,  1, 17, 44, 42,  7,
          8, 16, 24, 32,
         28, 46, 35,  4, 60, 30, 39,
         51, 14, 27, 34, 31, 62, 50,
         29, 41, 53, 15, 61,  6, 33,
          5, 26, 38, 47, 49, 52, 63,
         40, 48, 56, 64 };
```

```
char  keyrot[18]
= {  2,4,4,4,3,1,4,2,4,3,4,5,2,5,4,1,3,1 };


char  PC2[48]
    = { 16, 32, 46,  4, 40, 24,
        38, 17, 23, 49, 25,  5,
         3, 34,  6,  2, 54, 15,
        22,  8, 35, 42, 48, 50,
         9, 41,  1, 43, 14, 27,
        56, 21, 30, 31, 55, 53,
        37, 10, 44, 19, 51, 28,
        11, 36, 12, 45, 29, 18  };



/*  ***************   Subkey  ***************
    Char S[8][64]
        ตาราง S_boxes : เปลี่ยนข้อมูล ขนาด 48 เป็น 32 บิท
*/

char S[8][64]

                    /*  s[1]   */
= { 3, 5,11,15,12,15, 6, 1, 9,13, 7,14, 8, 4,10, 2,
    7, 9, 3,16, 1, 6, 2, 8,15,14, 4,12,10, 5,11,13,
   13,12, 1, 4,14, 5, 2,11, 3,15,16, 8,10, 9, 6, 7,
    6, 5, 8, 7, 4, 9,16, 1,10, 2,15,14, 3,12,11,13,
```

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

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

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

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

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

```
                /*  s[7]   */
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,

                /*  s[8]   */
 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 };
```

```
/*
 des64.C    ประกอบด้วย รายละเอียดโปรแกรมในการเข้าและถอดรหัสของ
            บล็อกข้อมูลขนาด 64 บิท
 ผู้เขียน    ร.อ.ชัยยง  ไชยสิงห์ทอง ได้ดัดแปลงมาจาก
            Australian Standard AS2805.5-1985 Data
            Encryption Algorithm โดย Lawrence Brown
            (Lpb@CSadfa.oz)  Dec 1987

 dedes  -   เป็นส่วนหลักของโปรแกรมการถอดรหัส โดยมีข้อมูลเข้าขนาด 64 บิท
            และผ่านขบวนการ DES ผนวกกับหมายเลขกุญแจ 18 ชุดการถอดรหัส
            ใช้หลักการเดียวกับการเข้ารหัสยกเว้นหมายเลขกุญแจใช้จาก 18ถึง1
*/
#include "des.h"
#include "des64.i"
#include <stdio.h>
#include <bios.h>
/*  char Subkey[18][8]   เก็บค่าหมายเลขกุญแจ 18 ชุด แต่ละชุดมีขนาด
                         48 บิท ถูกเก็บในลักษณะ 6 บิท 8 ตัว
*/
char  subkey[18][8];
/*************************************************************
endes(b)    ส่วนโปรแกรมหลักของการเข้ารหัสโดยเข้ารหัสข้อมูล b เป็นบล็อกๆ
            ละ 64 บิท  โดยใช้ขบวนการ DES และ หมายเลขกุญแจ
            ขบวนการเข้ารหัสเริ่มจากการสลับสับเปลี่ยนตำแหน่ง  PERMUTING
            ข้อมูลเข้า 64 บิท    โดยใช้ตาราง  IP  และผ่านเข้าขบวนการ
            Function_F 18 รอบ โดยผนวกกับหมายเลขกุญแจ 18 ชุด จะทำ
            ให้ข้อมูลที่ผ่านการเข้ารหัสแล้วเป็นข้อมูลที่ไม่เป็นรูปแบบ และสุดท้าย
            ผ่านขบวนการ IP-1
            ขนาดของ b: เป็นข้อมูล 64 บิท ผ่านตัวแปร Long words 2 ตัว
                        ตำแหน่งของ บิท มีดังนี้
```

```
            [ 1 2 3... 32] [ 33 34 35... 64]
            Longwords 2 ตัว โดยกำหนดให้ L (ซ้าย)แทนด้วย b[0] และ
            R (ขวา) แทนด้วย b[1]
**************************************************************/
                endes(b)
/**************************************************************/
long  b[2];
{    long   work[2];
         register  int  i,jj;
         perm64(work, b, IP);      /* PERFORM   IP_1   */
         for (i=0; i<=18; i++)
                  round(work,  subkey[i]);
         swap(work);
         perm64(b,  work,  FP);   /*  PERFORM   IP_2    */
}
/**************************************************************
dedes(b)   ส่วนโปรแกรมหลักของการถอดรหัสโดยถอดรหัสข้อมูล b เป็นบล็อกๆ
           ละ 64 บิท  โดยใช้ขบวนการ DES และ หมายเลขกุญแจ
**************************************************************/
                dedes(b)
/**************************************************************/
long   b[2];
{    long   work[2];
    register  int  i;
        perm64(work,  b,  IP);
        for  (i=18;  i>=0;  i--)
                  round(work,  subkey[i]);
        swap(work);
    perm64(b, work, FP);
}
```

```
/*************************************************************
round(d,k)  - กระทำในส่วนของ Function_F เพียง 1 ครั้ง โดยใช้ข้อมูล
              เข้าขนาด 64 บิทเป็นบล็อก (d) และใช้หมายเลขกุญแจ 1 ชุด
              ในแต่ละครั้งของขบวนการนี้จะทำ
              L(i)  =  R(i-1)
              R(i)  =  L(i-1) XOR f(R(i-1),K(i))
              ขนาดข้อมูล 64 บิท(b) แตกเป็นข้อมูลซ้าย d[0] และขวา d[1]
*************************************************************/
              round(d, k)
/*************************************************************/
long  d[2];
char  *k;
{
        long   t,bf;
     extern  long  f();
        bf = f(d[1], k);
        t = d[0] ^ bf;
        d[0] = d[1];
     d[1] = t;
}
/*************************************************************
f(r,k) เป็นส่วนที่ทำ Function_F ข้อมูลออกเป็นผลของ r คือ Input 32 บิท
       และหมายเลขกุญแจ K ข้อมูลเข้า R(i-1) เป็นข้อมูลขนาด 32 บิท ถูก
       ขยายเป็น 48 บิท โดยผ่านตาราง E และ XOR กับหมายเลขกุญแจ K(i)
       ผลลัพธ์จากการ XOR ถูกแทนที่ด้วยวิธีการและข้อมูลของ S_boxes ในขั้น
       สุดท้ายผลลัพธ์ที่ได้ถูกสลับสับเปลี่ยนโดยใช้ตาราง P วิธีการทางคณิตศาสตร์
       เขียนได้เป็น
                    A = E(R(i-1)) XOR K(i)  (a 48 bit value)
                    B = S(A)                (a 32 bit value)
                    f = P(B)                (a 32 bit value)
```

ผลลัพธ์อยู่ในรูปของ 48 bit และเก็บในรูปของข้อมูลขนาด 6 bit ในตัวแปรขนาด 8 บิท (ไบท์) 8 ชุด ในแต่ละไบท์จะเป็น [xx 1 2 3 4 5 6]

ถ้าเรียงบิททั้งหมดจะเป็นดังนี้

[xx 1 2 3 4 5 6] [xx 7 8 9 10 11 12]...[xx 43 44 45 46 47 48] ขนาดของบิทอยู่ในรูปของ 6 bit ข้อมูลเข้าของขบวนการ S_box เขียนได้เป็น [xx 1 2 3 4 5 6] แบ่งแยกได้เป็น bit[1,6] เรียกว่า Row และ bit [2,3,4,5] เรียกว่า Columm และเรียงบิทเสียใหม่เป็น[xx 1 6 2 3 4 5]ก่อนที่จะเทียบในตาราง S_box

```
**********************************************************/
        long    f(r, k)
/**********************************************************/
  long  r;
  char  *k;
  {   long  b = 0, out = 0;
      char a[8];
      register long  s;
      register long  rc;
      register  int  j;
      expand(a, r);         /*  PERFORM  MATRIX_E  */
      for (j=0; j<8; j++)
      {  rc = a[j] ^ k[j];
         rc = (rc& 0x20) | ((rc << 4) & 0x10) | ((rc >> 1) & 0x0F);
         s = S[j][rc];
         b = (b << 4) | s;
          }
      perm32(&out, &b, P);
        return(out);
}
```

```
/*************************************************************
keyinit1(key)  -  คำนวณรหัสกุญแจ และกระจายออกเป็นหมายกุญแจ 9 ชุด
                  คือ ชุดที่ 1 ถึง 9  เพื่อใช้ในการเข้าและถอดรหัสต่อไป
                  ข้อมูลเข้า key เป็นข้อมูลขนาด 64 บิท ใช้จริง 56 บิท
                  โดยตัดส่วนบิทตรวจสอบข้อผิดพลาดออก  8,16,24,32,
                  40,48,56,64 กระจาย key 56 บิทออกเป็นหมายเลข
                  กุญแจขนาด 48 บิท 9 ชุด โดยแต่ละชุด(48บิท) เก็บใน
                  รูปของ 6 บิท 8 ตัว

                  การคำนวณหมายเลขกุญแจประกอบด้วย

                  สลับสับเปลี่ยน key โดยใช้ตาราง PC1 และแบ่ง 56 บิท
                  ออกเป็น 2 ส่วน C,D
                  ทำซึ่ง 16 รอบ
                        หมุนซ้ายข้อมูลในแต่ละครึ่งเท่ากับจำนวนที่กำหนดใน
                        ตารางการหมุนข้อมูลที่ได้อยู่ในรูปของ 28 บิท 2 ชุด
                        เข้าตารางสลับสับ  เปลี่ยน PC2 ผลลัพธ์ คือ ข้อมูล
                        ขนาด 48 บิท  ซึ่งใช้เป็นหมายเลขกุญแจ
                  จบ
*************************************************************/
      #define MASK28  0xFFFFFFF0L
                keyinit1(key)
/*************************************************************/
long  key[2];
{
     long  unsigned cd[2];
     long  res1,res2,res3,res4,res5,res6,res7,res8;
     int pos,j;
   register  int  i;
   perm64(cd, key, PC1);
```

```
    cd[0] &=  MASK28;
    cd[1] &=  MASK28;
    for (i=0; i<8; i++)
     {
       pos = (int)keyrot[i];
       cd[0] =((cd[0] << pos)|(cd[0] >> (28-pos))) & MASK28;
       cd[1] =((cd[1] << pos)|(cd[1] >> (28-pos))) & MASK28;
       keyperm(cd, i);
     }
}
/*************************************************************
keyinit2(key)  เหมือนกับ keyinit1 (key)  แตกต่างกันที่จะให้หมายเลข
               กุญแจ  ชุดที่ 10 ถึง 18
*************************************************************/
               keyinit2(key)
/*************************************************************/
long  key[2];
{
        long  unsigned cd[2];
        long  res1,res2,res3,res4,res5,res6,res7,res8;
        int pos,j;
    register  int  i;
    perm64(cd, key, PC1);
    cd[0] &=  MASK28;
    cd[1] &=  MASK28;
    for (i=9; i<18; i++)
    {
      pos = (int)keyrot[i];
      cd[0] =((cd[0] << pos)|(cd[0] >> (28-pos))) & MASK28;
      cd[1] =((cd[1] << pos)|(cd[1] >> (28-pos))) & MASK28;
```

```
            keyperm(cd, i);
        }
}
/*************************************************************
    Keyperm(cd,i) รับข้อมูล key ซึ่งเป็น 2 ส่วน อยู่ใน cd
                  สลับสับเปลี่ยนตามตาราง PC2 และให้ผลลัพธ์ขนาด 48 bit
                  ซึ่งเก็บในรูปของ 6 บิท  8 ตัว
*************************************************************/
  #define  KEYBIT1 0X20
          keyperm(cd, i)
/*************************************************************/
long  cd[2];
int  i;
{
   register  int  j,k;
   register  char  mask;
   register  char  *perm = PC2;
   int mm;
   for  (j=0;  j<4; j++)
   {    subkey[i][j] = 0;
        mask = KEYBIT1;
        for (k=0; k<6; k++)
          {   if (keybit(cd, (int)*perm++) == 1)
                        subkey[i][j] |= mask;
              mask >>= 1;
          }
    }
    for  (j=4; j<8; j++)
      {   subkey[i][j] = 0;
          mask = KEYBIT1;
```

```
            for (k=0; k<6; k++)
              {   if  (keybit(cd, (int)*perm++) == 1)
                       subkey[i][j] |= mask;
                   mask >>= 1;
               }
             }
}
/*************************************************************
Keybit (n,pos) คืนค่าเป็นบิทในตำแหน่ง POS ตามจำนวน n (1<=pos<=48)
               n บรรจุข้อมูล 24 บิท 2 ชุด ข้อมูลเก็บแบบ [12,,,27 28
               xxxx]  [29 30 ,,,,48 xxxxx] 4 บิทสุดท้ายจะไม่ใช้
*************************************************************/
        int keybit(n,pos)
/*************************************************************/
long  n[2];
int   pos;
{   register int  b,  d,  o;
        d = (pos - 1) / 28;
        o = 31 - ((pos -1) % 28);
    b = ((n[d] >> o) & 01);
    return (b);
}
/*****************************************/
         rot128(b, pos)
/*****************************************/
long  *b;
int  pos;
{
        *b = ((*b << pos) | (*b >> (28-pos))) & MASK28;
}
```

```
/*************************************************************
Perm64(out, in,perm)  เป็นโปรแกรมทำการสลับสับเปลี่ยน ข้อมูล in ขนาด
                      64 บิท  ให้ผลลัพธ์ out ขนาด 64 บิท การสลับสับ
                      เปลี่ยนตามตารางที่กำหนดโดย Perm
*************************************************************/
#define  DESBIT1  0x80000000L
        perm64(out, in, perm)
/*************************************************************/
long  out[2];
long  in[2];
char  perm[64];
{
        long unsigned mask  = DESBIT1;
     register  int  i;
        out[0]  = 0L;
     for  (i=0;  i<32;  i++)
          {   if  (bit (in, (int)*perm++)  == 1)
                  out[0]  |= mask;
          mask >>= 1;
          }
          out[1] = 0L;
       mask = DESBIT1;
          for (i=0; i<32; i++)  {
            if  (bit(in,  (int)*perm++) == 1)
                  out[1] |= mask;
            mask >>= 1;
          }
}
```

ภาคผนวก ข.

บทความเรื่องเทคนิคการเข้าและถอดรหัสเพื่อใช้ในด้านการทหาร ได้ลงตีพิมพ์ใน
วารสารทางวิชาการของสมาคมคอมพิวเตอร์แห่งประเทศไทย
ฉบับที่ 106 ประจำเดือน มีนาคม-เมษายน 2537

เทคนิคการเข้าและถอดรหัสเพื่อใช้ในด้านการทหาร

(Data Encryption Algorithm For Military Purposes)

บุญวัฒน์ อัตชู*
Boonwat Attachoo

ร้อยเอก ชัยยง ไชยสิงห์ทอง**
Capt. Chaiyong chaisingthong

บทคัดย่อ

งานวิจัยนี้นำเสนอเทคนิคการเข้าและถอดรหัส (Data Encryption) เพื่อให้มีประสิทธิภาพมากขึ้นกว่าแบบ Data Encryption Standard (DES) เทคนิคใหม่จะใช้หลักการของระบบมาตรฐาน DES และปรับปรุงการผสมรวมการเข้ารหัสลับพื้นฐานของแบบแทนที่ (Substitution) และแบบสับเปลี่ยน(Transposition)ให้ผิดไปจากเดิม และกำหนดกุญแจยาวกว่าเดิมจาก ๕๖ บิต เป็น ๑๑๒ บิต ผลการทดลองพบว่าเทคนิคใหม่นี้ทำให้การลักลอบถอดรหัสทำได้ยากขึ้นและเทคนิคใหม่เป็นเอกลักษณ์

Abstract

To build a new Data Encryption algorithm base on Data Encryption standard(DES). The new Data Encryption Algorithm will be more complex than the DES. By improving the key length from 56 to 112 bits makes it longer to break code. The new Data Encryption algorithm is unique.

---

* ผู้ช่วยศาสตราจารย์ คณะวิศวกรรมศาสตร์

** นักศึกษาปริญญาโท สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

1. บทนำ

เนื่องจากการพัฒนาด้านคอมพิวเตอร์เริ่มเข้ามามีความสำคัญต่อการใช้งานในสังคมปัจจุบัน การเก็บข้อมูลในคอมพิวเตอร์หรือการติดต่อสื่อสารในระบบคอมพิวเตอร์เป็นเรื่องสำคัญที่บางครั้งองค์กรนั้นๆจะต้องปกปิดเป็นความลับโดยมีวิธีการเข้าถึงข้อมูลโดยใช้รหัสผ่าน (PASSWORD) เพื่อจำกัดสิทธิ์ความสำคัญของผู้ใช้ การใช้วิธี Copy Protection เช่นเกมส์หลาย ๆ อย่างใช้กันในเครื่อง PC เพื่อกันการก็อปปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือการเข้ารหัสข้อมูล (Crypto graphy หรือ Data Encryption) เพื่อให้ข้อมูลเปลี่ยนไปจากเดิมจนไม่สามารถเข้าใจได้ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิมได้

วิธีการทำ Data Encryption นั้นมีหลายวิธี วิธีใดมีความสลับซับซ้อนสูงจะทำให้การพยายามที่จะถอดรหัสทำได้ยากขึ้นความสามารถของเครื่องขนาดใหญ่เช่น Super Main-frame หรือ Mini Computer ระบบปฏิบัติการเหล่านี้เป็นแบบ Multitasking คือใช้พร้อม ๆ กันได้หลายคน ข้อมูลที่อยู่ในระบบปฏิบัติที่ใหญ่ แบบนี้จึงจำเป็นต้องมีการรักษาความปลอดภัยของข้อมูลที่ดีพอเช่นการเก็บข้อมูลของรหัสผ่านก็เก็บในรูปของ Data Encryption ซึ่งต้องใช้วิธีการคำนวณมากซับซ้อนหลายรอบ

2. ทฤษฎี

การเข้ารหัสแบบ DES[1] ใช้เพื่อป้องกันการลักลอบ การดักฟังข้อมูลในระบบคอมพิวเตอร์ หลักการและทฤษฎีของ DES ใช้วิธีทางคณิตศาสตร์เข้ามาสลับสับเปลี่ยนข้อมูล ข้อมูลที่เข้ามาผ่านขบวนการเข้ารหัสจะถูกสลับสับเปลี่ยนเป็นข้อมูลที่ไม่มีรูปแบบเรียกว่า CIPHER การถอดรหัสจะนำข้อมูลที่ไม่มีรูปแบบนี้มาเข้าขบวนการถอดรหัส จะถูกสลับสับเปลี่ยนเป็นข้อมูลเดิมได้ ทฤษฎีของ DES อธิบายถึงวิธีการเข้าและถอดรหัสโดยใช้เลขฐาน 2 กุญแจประกอบด้วยเลขฐาน 2 64 บิท ใช้ในขบวนการเข้าและถอดรหัส และอีก 8 บิท ใช้ในการตรวจสอบข้อผิดพลาดของ 56 บิต (ERROR DETECTION)

ข้อมูลที่ต้องการเข้ารหัสจะต้องใช้ร่วมกับกุญแจ กุญแจ 56 บิท จะต้องใช้ร่วมกับทฤษฎีการเข้าและถอดรหัสอีก 8 บิทเป็นการตรวจสอบข้อผิดพลาดซึ่งให้ค่าเป็น 1 (PARITY BIT) ในแต่ละไบต์ (8 บิท) เมื่อมีค่าเป็นเลขคู่ ในแต่ละด้านของการเข้าและถอดรหัสจะมีกุญแจที่เหมือนกันกุญแจที่เลือกใช้จะเป็นกุญแจที่เมื่อใช้ในการเข้ารหัสและถอดรหัสแล้วจะได้ข้อมูลเดิม การใช้กุญแจที่ถอดรหัสที่ไม่ใช่กุญแจที่ใช้ในการเข้ารหัสจะส่งผลให้เมื่อถอดรหัสแล้วจะไม่ได้ข้อมูลเดิม ด้วยเหตุผลนี้การใช้กุญแจที่เหมือนกันและเก็บค่าของกุญแจเป็นความลับจะป้องกันการดักฟังหรือการจรกรรมข้อมูลได้

ข้อมูลที่จะถอดรหัสได้นั้นจะต้องใช้กุญแจเดียวกับการเข้ารหัสเท่านั้นผู้ใดที่รู้หลักการหรือทฤษฎีของการเข้ารหัสแต่ไม่รู้หมายเลขกุญแจที่ใช้จะไม่สามารถถอดรหัสได้โดยง่าย ผู้ที่รู้ทฤษฎีและหมายเลขกุญแจเท่านั้นที่จะถอดรหัสได้ในเวลาอันสั้น

3. การสร้างและออกแบบ

หลักการออกแบบอธิบายได้เป็น 6 ส่วน ดังนี้

3.1 ขบวนการสลับสับเปลี่ยน Initial Permutation IP และ Inverse Initial Permutation IP_1

3.2 แสดงการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้ง

3.3 แสดงการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง

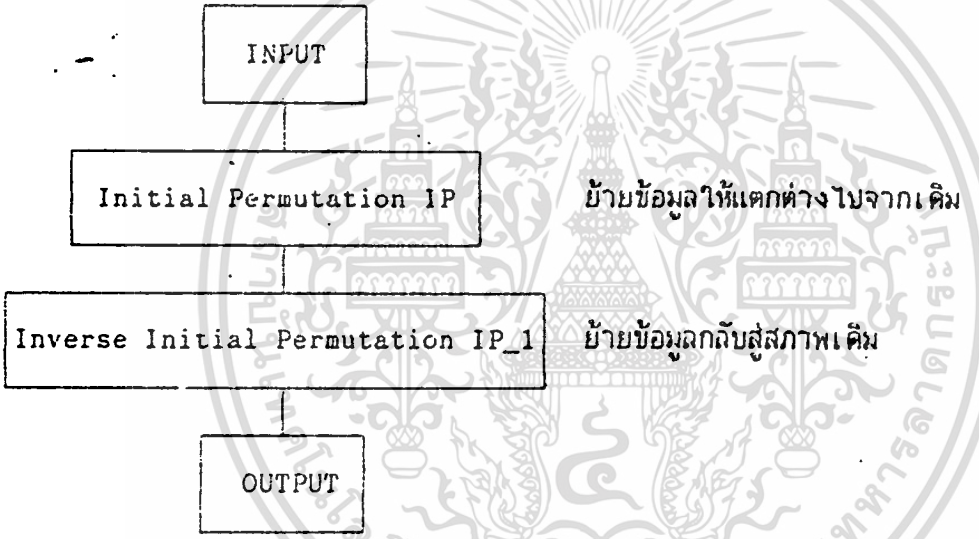
3.4 แสดงการเข้าและถอดรหัสโดยทำ Function_F หลายครั้ง

3.5 ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ (Function_F และ S_Box)

3.6 การคำนวณหมายเลขกุญแจโดยใช้ขนาด 112 บิท

3.1 ขบวนการสลับสับเปลี่ยน Initial Permutation IP และ Inverse Initial Permutation IP_1

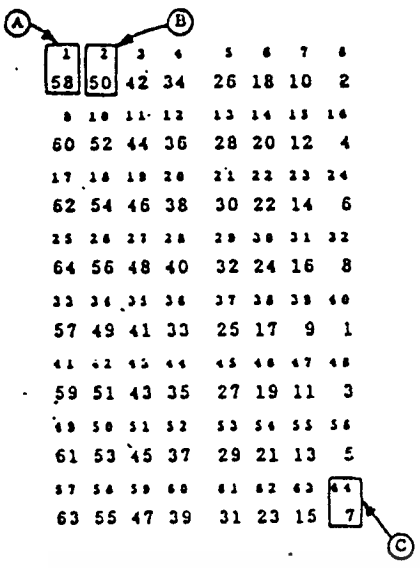
จุดประสงค์ของการทำขบวนการสลับสับเปลี่ยน IP และ IP_1 คือการโยกย้ายบิทให้แตกต่างไปจากข้อมูลเดิมและเมื่อทำการสลับสับเปลี่ยนย้อนกลับจะได้ข้อมูลเดิม IP และ IP_1



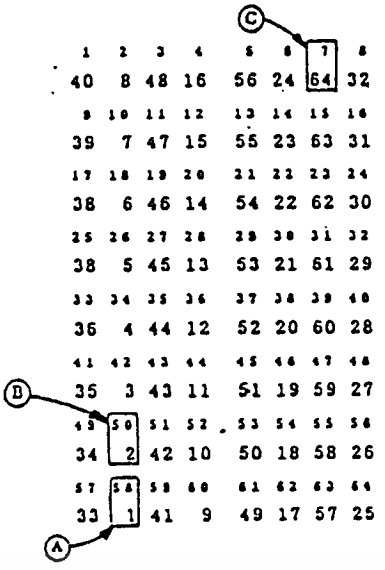
รูปที่ 1 แสดงการทำ IP และ IP_1 โดยตัดส่วนของขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้งออก

จากรูปที่ 1 จะทำให้การอธิบายดูง่ายขึ้นถ้าตัดส่วนที่เป็นขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ 16 ครั้งออกจะเห็นว่าข้อมูลเข้าและออกจะมีค่าเดียวกันนั่นคือขบวนการสลับสับเปลี่ยน IP รับ Input ขนาด 64 บิทมาสลับตำแหน่งตามตาราง Initial-Permutation IP และส่ง Output ให้กับ IP_1 ซึ่งเป็นการทำขบวนการย้อนกลับกับ IP ตามตาราง Inverse Initial Permutation IP_1 ดังนี้

ค่า Input = ค่า Output



รูปที่ 2 แสดงตารางการสลับสับเปลี่ยน IP



รูปที่ 3 แสดงตารางการสลับสับเปลี่ยน IP_1

1 ตำแหน่งบิทที่ 1 ของตารางนั้น ๆ

58 ค่าของบิทที่ 58 ของ Input

ข้อมูลลำดับที่ 1,2,64 หรือจุด A,B,C จากรูปที่ 2 จะถูกสลับที่ไปอยู่จุด A,B,C ของรูปที่ 3 ขบวนการสลับสับเปลี่ยน IP และ IP_1 นี้ เป็นการทำขบวนการย้อนกลับกันตามรูปที่ 1

ตัวอย่าง ข้อมูล Input = A2DC 8F5A 6ACE 160C

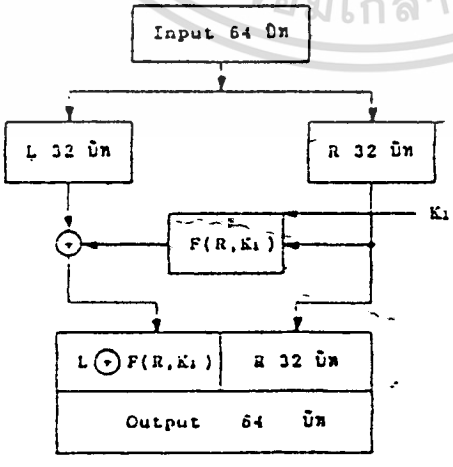
หลังจากทำ IP = 3A4A E604 2711 BE7D

หลังจากทำ IP_1 = A2DC 8F5A 6ACE 160C

ค่า Input = ค่า Output(IP_1)

3.2 แสดงการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้ง

ข้อมูลเข้าสู่ขบวนการนี้คือข้อมูล 32 บิทซ้ายและหมายเลขกุญแจ 48 บิท เพื่อให้เข้าใจถึงหลักการได้ง่ายขึ้นถ้าตัดส่วนของการสลับสับเปลี่ยน IP และ IP_1 ออกและให้ทำ Function_F เพียงครั้งเดียวซึ่งตามกฎของ DES ต้องทำถึง 16 ครั้ง



รูปที่ 4 แสดงการเข้าและถอดรหัสโดยตัดส่วนที่ทำ IP และ IP_1 ออกและทำ Function_F เพียงครั้งเดียว

จากรูปที่ 4 ใช้เป็นการเข้าและถอดรหัสโดยใช้หมายเลขกุญแจเพียงชุดเดียวนั้นหมายถึงทำ Function_F เพียงครั้งเดียว

แสดงการถอดรหัส จากรูปที่ 4 กำหนดให้

Input = LR = 48E2 D242 3D2D B65B (64 บิท)
หมายเลขกุญแจ = K_1 = 3A1F 1638 2D1D 3C32 (48 บิท)
จะได้ Output = $[L \oplus F(R,K_1)]\ R$ = B71D 2DBD 3D2D B65B (64 บิท)

แสดงการถอดรหัสโดยนำข้อมูลจากการเข้ารหัสมาเป็นข้อมูลเข้าของการถอดรหัสดังรูปที่ 4

Input = $[L \oplus F(R,K_1)]\ R$ = B71D 2DBD 3D2D B65B (64 บิท)
หมายเลขกุญแจ = K_1 = 3A1F 1638 2D1D 3C32 (48 บิท)
จะได้ Output = $\{[L \oplus F(R,K_1)] \oplus F(R,K_1)\}R = LR$
= 48E2 D242 3D2D B65B (64 บิท)

จะเห็นว่าข้อมูลเข้าของการทำการเข้ารหัสเท่ากับข้อมูลออกของการทำการถอดรหัส

ขออธิบายแนวคิด โดยแบ่งเป็น 3 ส่วน ดังนี้

3.2.1 XOR
3.2.2 Function_F
3.2.3 Output แบ่งเป็น 2 ส่วน
- $L \oplus F(R,K_1)$
- R 32 บิท

3.2.1 XOR

ในหลักการของ DES ใช้เทคนิคของ XOR ช่วยในการเข้าและถอดรหัส จะได้

$$(L \oplus X) \oplus X = L$$

สังเกตุได้ว่าเมื่อทำการ XOR กับข้อมูลใด ๆ L กับ X นำผลลัพธ์ที่ได้ $L \oplus X$ มาทำการ XOR กับค่าเดิม X จะได้ข้อมูลเดิม L

3.2.2 Function_F

นำ Input ซึ่งเป็น R 32 บิท และหมายเลขกุญแจ K_1 มาทำขบวนการสลับสับเปลี่ยนโดยใช้ S_Box ผลลัพธ์ที่ได้มาทำการ XOR กับ L เขียนได้เป็น $L \oplus F(R,K_1)$ ในขณะนี้ขอให้คิดว่า Function_F เป็น Function หนึ่ง โดยไม่ต้องคำนึงถึงความสลับซับซ้อนภายใน

กำหนดให้ $F(R,K_1) = X$

3.2.3 Output

จากรูปที่ 4 จะเห็นว่าผลลัพธ์จากการเข้ารหัส แบ่งเป็น 2 ส่วน คือ

$L \oplus F(R,K_1)$	R 32 บิท

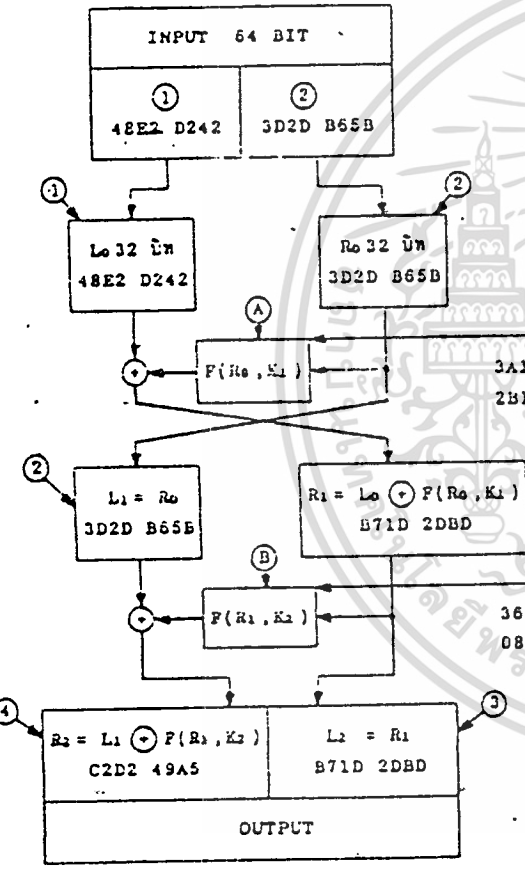
ข้อมูล R ไม่ถูกเปลี่ยนแปลง และใช้เป็น Input ของ Function_F เพื่อทำการถอดรหัส

ขอให้สังเกตุว่าการเข้าและถอดรหัส Function_F ยังคงใช้ Input R และ K_1 เหมือนกันดังนั้นค่า $F(R,K_1)$ ในการเข้าและถอดรหัสจะเท่ากัน

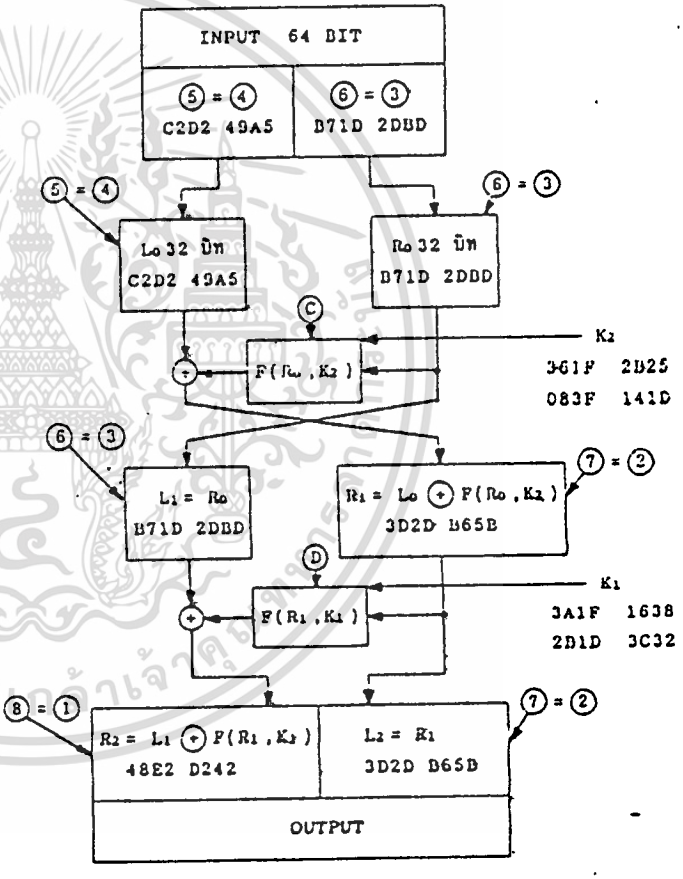
ดังนั้นจากที่กล่าวมาแล้วจะเห็นว่าข้อมูลเข้าของการทำการเข้ารหัสและข้อมูลออกของการทำการถอดรหัสจะเหมือนกันนั้นหมายถึงเมื่อใช้หมายเลขกุญแจเพียงชุดเดียวและทำรอบเดียวก็ยังคงรักษาคุณลักษณะของการเข้าและถอดรหัสได้แต่ไม่ซับซ้อนมาก

3.3 การเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง

การใช้ Function_F 2 ครั้ง ใช้หลักการเดียวกับการเข้าและถอดรหัสโดยทำ Function_F 1 ครั้ง ที่ได้กล่าวมาแล้ว มีข้อแตกต่างคือ หลังจากทำ Function_F 1 ครั้ง ข้อมูล L และ R ถูกสลับที่กัน โดยให้ $L_1 = R_0$ และ $R_1 = L_0 \oplus F(R_0,K_1)$ หลังจากนั้นเข้าสู่ขบวนการทำ Function_F ครั้งที่ 2 ดังรูปที่ 5



รูปที่ 5 การเข้ารหัสโดยทำ Function_F 2 ครั้ง



รูปที่ 6 การถอดรหัส โดยทำ Function_F 2 ครั้ง

เมื่อนำข้อมูลที่ผ่านการเข้ารหัส 4 และ 3 โดยทำ Function_F 2 ครั้ง แล้วมาเข้าขบวนการถอดรหัส ตามรูปที่ 6 โดยให้ข้อมูลจุดที่ 5 = 4 และ 6 = 3 ขบวนการถอดรหัสจะทำย้อนกลับกับขบวนการในการเข้ารหัส ให้สังเกตุที่การเข้ารหัสหมายเลขกุญแจจะเริ่มจาก K_1 และ K_2 ส่วนการถอดรหัสหมายเลขกุญแจจะเริ่มจาก K_2 และ K_1

รูปที่ 5 และ 6 ในรอบแรก Input ที่จุด C จะเท่ากับ Input ที่จุด B คือ 3 = 6 และ

K_2 จะได้ผลลัพธ์ Function ที่เท่ากัน เมื่อผลลัพธ์ของ 5 ⊕ C ได้ 7 ข้อมูลจุดที่ 7 จะเท่ากับ 2 ซึ่งเป็นการทำขบวนการย้อนกลับในรอบแรกของการถอดรหัส โดยใช้หลักของการทำ XOR 2 ครั้งจะได้ค่าเดิม ดังเสนอไปแล้วนั้น

ในรอบที่สอง Input ที่จุด D จะเท่ากับจุด A คือ 2 = 7 และ K_1 ซึ่งจะให้ผลลัพธ์ของ Function ที่เท่ากันเนื่องจาก 7 = 2 และใช้ K_1 ชุดเดียวกัน ผลลัพธ์ของ Function ที่จุด D ⊕ 6 ได้ผลลัพธ์เท่ากับ 8 ซึ่ง 8 = 1 โดยใช้หลักของการทำ XOR กับข้อมูลเดิม 2 ครั้ง

การแสดงการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง ที่ได้นำเสนอแล้วนั้น เป็นการแสดงให้เห็นถึงวิธีการเข้าและถอดรหัสเพื่อให้เข้าใจได้ง่ายและไม่ยุ่งยากซับซ้อนมากนั้น

3.4 การเข้าและถอดรหัสโดยทำ Function_F หลายครั้ง

ในระบบของ DES จะใช้หลักการเข้าและถอดรหัสถึง 16 รอบโดยใช้หมายเลขกุญแจ 16 ชุด การเข้ารหัสต้องใช้หมายเลขกุญแจตั้งแต่ K_1 ถึง K_{16} แต่การถอดรหัสจะใช้ K_{16} ถึง K_1 ซึ่งเป็นขบวนการย้อนกลับกับการเข้ารหัส ความสลับซับซ้อนจะมากแต่ยังคงใช้หลักการเดียวกับการเข้าและถอดรหัสโดยทำ Function_F 2 ครั้ง

การกำหนดจำนวนรอบในการเข้าและถอดรหัสสามารถกำหนดให้มีมากกว่าที่ DES กำหนดนั้นหมายถึงเป็นการเพิ่มความสลับซับซ้อนให้กับระบบ

3.5 ขบวนการสลับสับเปลี่ยนผนวกกับรหัสกุญแจ (Function_F และ S_Box)

การทำงานของ Function_F และ S_Box ตามหลักการของ DES ตาราง E รับ Input ขนาด 32 บิทและให้ Output ขนาด 48 บิทเหตุที่ต้องทำให้เป็น 48 บิทคือต้องทำให้อยู่ในรูปแบบเดียวกับหมายเลขกุญแจ (48 บิท) ซึ่งจะอธิบายในโอกาสต่อไป ความสัมพันธ์ระหว่างตาราง E, S_Box และตาราง P นั้นไม่มี ขอให้คำนึงเสมอว่าการทำงานของ Function_F นั้นทำจากบนลงล่างไม่มีการทำย้อนกลับ สามารถกำหนดตาราง E, S_Box และ P ใหม่โดยยึดหลักว่า

ตาราง E : ต้องมี Output ขนาด 48 บิท ซึ่งต้องมีบิทซ้ำ 16 บิทเนื่องจากรับ Input ขนาด 32 บิทและให้ Output ขนาด 48 บิท

: ตำแหน่งบิทต่าง ๆ สามารถกำหนดขึ้นเองตามความพอใจ เช่น

ตารางที่ 1 ตาราง E ใหม่

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

S_Box

ตามขบวนการ DES S_Box รับ Input ขนาด 6 บิท 8 ชุด ซึ่งเท่ากับ 48 บิทมาเทียบในตาราง S1 ถึง S8 ได้ Output 4 บิท 8 ชุด ซึ่งเท่ากับ 32 บิท ขบวนการทำงานมีดังนี้

รับ Input ขนาด 6 บิท เช่น 101011 (บิท 1 2 3 4 5 6) ให้ Row = บิท 1,6 ให้ Column = บิท 2,3,4,5 จะได้ Row = 11 = 3 และ Column 0101 = 5 เทียบในตาราง S1 จะได้ 9 = 1001 ซึ่งมีขนาด 4 บิท เมื่อทราบถึงการ ทำงานของ S_Box อีกทั้งขบวนการของ

ตารางที่ 2 ตาราง S1 ตามหลักของ DES

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

Function_F ถือเป็นเสมือน Function หนึ่งซึ่งไม่มีการทำขบวนการย้อนกลับดังนั้นการกำหนดขบวนการของ S_Box ให้แตกต่างไปจากขบวนการของ DES กระทำได้หลายวิธี เช่น

ถ้ากำหนดให้ Row = บิท 5,1 และให้ Column = บิท 6,2,4,3 หรือให้ Row = บิท 1,2 และให้ Column = บิท 3,4,5,6 ย่อมกระทำได้

ตาราง S1...S8 สามารถกำหนดใหม่ให้แตกต่างไปจาก DES ดังนี้

S1

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

S2

.
.
.

S8

.
.
.

ตาราง P รับ Input ขนาด 32 บิทสลับสับเปลี่ยนตำแหน่งตามรูปแบบของตาราง P และได้ Output ขนาด 32 บิทการกำหนดตาราง P ให้แตกต่างไปจาก DES ทำได้เช่น

ตารางที่ 3 ตาราง P ใหม่

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

การกำหนดตาราง E, S_Box และ P ขึ้นใหม่กระทำได้ดังที่ได้กล่าวมาแล้วการเข้าและถอดรหัสจะได้ข้อมูลที่ถูกต้องต่อเมื่อการเข้าและถอดรหัสใช้ตารางและวิธีการเดียวกัน

3.6 การคำนวณหมายเลขกุญแจโดยใช้ขนาด 112 บิท

จุดประสงค์ของการใช้หมายเลขกุญแจขนาด 112 บิท คือ จะทำให้ผู้พยายามที่จะจรกรรมข้อมูล ใช้เวลานานในการค้นหาหมายเลขกุญแจที่ถูกต้องการพยายามในการค้นหาหมายเลขกุญแจที่ถูกต้องไม่มีทางอื่นนอกจากต้องลองทุกๆหมายเลขกุญแจที่เป็นไปได้ กุญแจขนาด 112 บิทจะต้องใช้ความพยายามในการถอดรหัสถึง 2^{112} หรือประมาณ 5.3×10^{33} ซึ่งประมาณ 5 พันล้านล้านล้านล้าน ครั้ง ซึ่งเมื่อเปรียบเทียบกับระบบ DES จะใช้ 2^{56} ซึ่งประมาณ 7.6×10^{16} หรือประมาณ 7 หมื่นล้านล้านครั้ง การกระจายหมายเลขกุญแจขนาด 112 บิท ออกเป็นหมายเลขกุญแจ 16 ชุดหรือมากกว่า กระทำได้ในหลายรูปแบบ ซึ่งขึ้นอยู่กับความประสงค์ของผู้ออกแบบ ขอนำเสนอการคำนวณหมายเลขกุญแจขนาด 112 บิทดังนี้

แบ่งหมายเลขกุญแจขนาด 112 บิท ออกเป็น 2 ส่วน ๆ ละ 56 บิท นำหมายเลขกุญแจขนาด 56 บิทมาเข้าขบวนการคำนวณหมายเลขกุญแจตามหลักของ DES โดยออกแบบ Permuted choice 1, Permuted choice 2 และตารางการหมุนกุญแจใหม่ และกระจายหมายเลขกุญแจขนาด 112 บิท ออกเป็นหมายเลขกุญแจ 18 ชุด

ตารางที่ 4 New Permuted choice 1 (PC_1)

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

กำหนดตำแหน่งต่าง ๆ ของ New PC_1 ตารางที่ 4 โดยตัดส่วนที่เป็น Parity bit ออก 8, 16, 24, 32, 40, 48, 56 และ 64 ออก

กำหนดตำแหน่งต่าง ๆ ตามหลักการของ DES สำหรับ PC_2 ต้องกำหนดบิทที่ไม่ใช้ขึ้น 8 ตัว เนื่องจาก Input ของ PC_2 มีขนาด 56 บิทและ Output มีขนาด 48 บิท กำหนดบิทที่ไม่ใช้ เช่น 7,13,20,26,33,39,47 และ 52

กำหนดตำแหน่งต่าง ๆ ของ New PC_2 ดังนี้

ตารางที่ 5 New PC_2

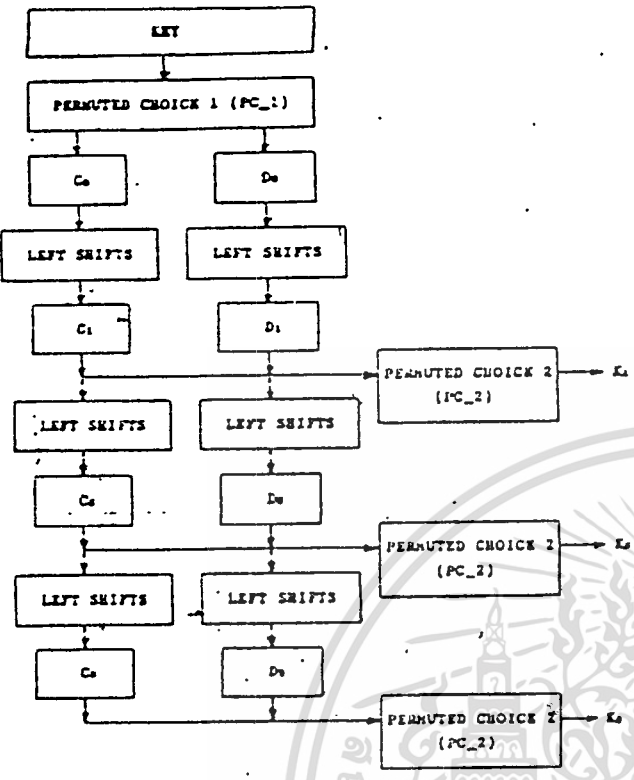
16	32	46	4	40	24
38	17	23	49	25	5
3	34	6	2	54	15
22	8	35	42	48	50
9	41	1	43	14	27
56	21	30	31	55	53
37	10	44	19	51	28
11	36	12	45	29	18

ตารางที่ 6 แสดงตารางการหมุนซ้าย สำหรับสร้างหมายเลขกุญแจชุดที่ 1 ถึง 9 ของกุญแจ 56 บิทชุดที่ 1

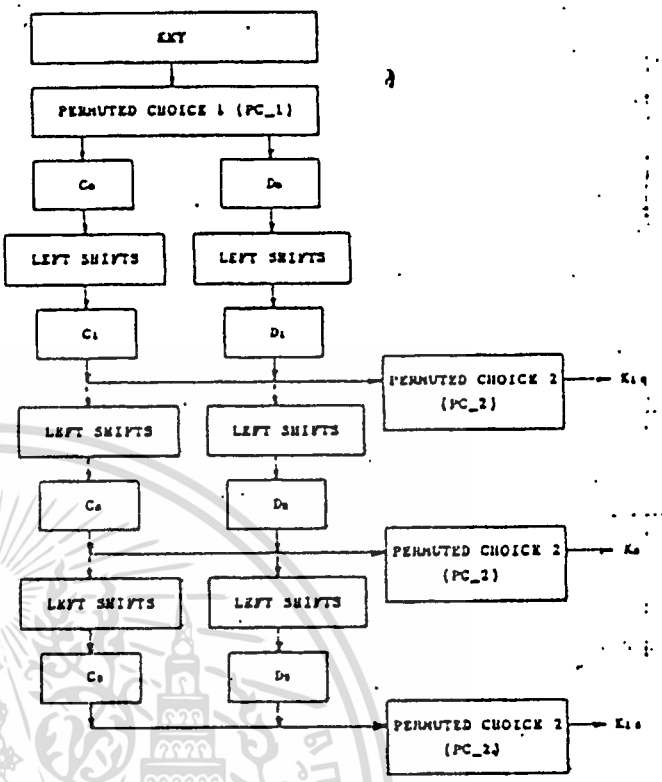
	ลำดับ	No Left Shifts
56 บิท ชุดที่ 1	1	2
	2	4
	3	4
	4	4
	5	3
	6	1
	7	4
	8	2
	9	4

ตารางที่ 7 แสดงตารางการหมุนซ้าย สำหรับสร้างหมายเลขกุญแจชุดที่ 10 ถึง 18 ของกุญแจ 56 บิทชุดที่ 2

	ลำดับ	No Left Shifts
56 บิท ชุดที่ 2	10	3
	11	4
	12	5
	13	2
	14	5
	15	4
	16	1
	17	3
	18	1



รูปที่ 7 แสดงการหาหมายเลขกุญแจ K_1 ถึง K_9 โดยใช้ตารางการหมุนที่ 6



รูปที่ 8 แสดงการหาหมายเลขกุญแจ K_{10} ถึง K_{16} โดยใช้ตารางการหมุนที่ 7

หลักการทำงาน

รูปที่ 7 นำข้อมูล 56 บิทชุดที่ 1 มาเข้าขบวนการสลับสับเปลี่ยน New PC_1 ตามตารางที่ 4 ข้อมูลที่ได้ถูกแบ่งเป็นสองส่วน Co และ Do นำ Co และ Do มาหมุนทางซ้ายครั้งที่ 1 ตามตารางที่ 6 ข้อมูลที่ได้เป็น C_1 และ D_1 ข้อมูลนี้ถูกป้อนเข้าขบวนการสลับสับเปลี่ยน New PC_2 ตามตารางที่ 5 ผลลัพธ์เป็นหมายเลขกุญแจชุดที่ 1 (K_1) นำข้อมูล C_1 และ D_1 มาหมุนซ้ายครั้งที่ 2 ตามตารางที่ 6 ข้อมูลที่ได้เป็น C_2 และ D_2 และถูกป้อนเข้า New PC_2 ตามตารางที่ 5 เพื่อให้ข้อมูลออกเป็นหมายเลขกุญแจชุดที่ 2 (K_2) ขบวนการนี้จะทำไปจนกว่าจะครบ หมายเลขกุญแจชุดที่ 9 (K_9)

การหาหมายเลขกุญแจ ตามรูปที่ 8 K_{10} ถึง K_{16} กระทำโดยใช้ข้อมูลกุญแจ 56 บิทชุดที่ 2 มาทำขบวนการเช่นเดียวกับการหา K_1 ถึง K_9 และตารางการหมุนใช้ตารางที่ 7

หลังจากทำตามขบวนการตามรูปที่ 7 และ 8 จะได้หมายเลขกุญแจ K_1 ถึง K_{16} เพื่อใช้ในขบวนการเข้าและถอดรหัสต่อไป

4. การทดลอง

ทดลองการเข้ารหัสโดยใช้เครื่องไมโครคอมพิวเตอร์ Acer 1120SX ความเร็ว 19.4 Mhz เขียนโปรแกรมด้วยภาษา C บน TurboC Version 1.5 พบว่าเวลาในการเข้ารหัส 1 ครั้งของข้อมูล 64 บิทเท่ากับ 0.0219230 วินาที การพยายามในการจรกรรมข้อมูลไม่มีทางอื่นนอกจากต้องลองทุกๆหมายเลขกุญแจที่เป็นไปได้

กุญแจ 2^{56} ประมาณ 7.20576×10^{16} เท่ากับ 7 หมื่นล้านล้าน หมายเลขกุญแจที่เป็นไปได้ ดังนั้นเวลาเฉลี่ยที่ใช้ในการหาหมายเลขกุญแจจะเท่ากับ

$$\frac{2^{56} \times \text{เวลาของการเข้ารหัส 1 ครั้ง(วินาที)}}{60(\text{นาที}) \times 60(\text{ชั่วโมง}) \times 24(\text{วัน}) \times 365(\text{ปี}) \times 2(\text{ค่าเฉลี่ย})} \quad \text{ปี}$$

เวลาเฉลี่ยในการหาหมายเลขกุญแจพบประมาณ 25 ล้านปี

กุญแจ 2^{112} ประมาณ 5.19229×10^{33} เท่ากับ 5 พันล้านล้านล้านล้านล้าน หมายเลขกุญแจที่เป็นไปได้ ดังนั้นเวลาเฉลี่ยที่ใช้ในการหาหมายเลขกุญแจจะเท่ากับ 36 ล้านล้านล้านล้าน ปี

5. สรุป

Diffie และ Hellman[1] คาดการไว้ว่า จะมีเครื่องมือสำหรับตรวจหาหมายเลขกุญแจที่สามารถถอดรหัสของ DES IC พิเศษที่ผลิตขึ้นหนึ่งล้านตัวจะมีค่าประมาณ 20 ล้านเหรียญซึ่งสามารถถอดรหัสที่ใช้วิธีการของ DES ได้ในหนึ่งวัน ยังประมาณการอีกว่าในปี 1990 จะมี Hardware ที่มีความสามารถมากขึ้นและจะทำให้ 56 BIT KEY มีขนาดไม่พอในด้านความปลอดภัย การพัฒนาของขบวนการ DES มีการพัฒนามาโดยตลอดและใช้กันอย่างกว้างขวาง นักค้นคว้าจากทุกมุมโลกให้ความสนใจและแข่งขันกันพัฒนาให้ดีขึ้น[6] DES สร้างขึ้นโดย IBM มีการกล่าวถึงมากว่า 56 Bit Key นั้นไม่พอเพียงสำหรับความปลอดภัยในยุคนี้

ตารางที่ 8 แสดงความแตกต่างระหว่างกุญแจที่มีขนาด 56 บิท และ 112 บิท

จำนวนบิต		จำนวนหมายเลขกุญแจที่เป็นไปได้	เวลาใช้ในการจรกรรมข้อมูล
จำนวนกุญแจเดิม	2^{56}	7 หมื่นล้านล้าน	1 วัน
จำนวนกุญแจที่พัฒนาขึ้น	2^{112}	5 พันล้านล้านล้านล้านล้าน	200 ล้านล้านปี

ถ้าใช้เวลา 1 วันในการถอดรหัส 2^{56} = 7 หมื่นล้านล้าน กุญแจที่เป็นไปได้เวลาใช้ในการถอดรหัสกุญแจ 2^{112}

$$= \frac{2^{112}}{2^{56}} = \frac{2^{56} \cdot 2^{56}}{2^{56}} = 2^{56} \text{ วัน}$$

= 7 หมื่นล้านล้าน วัน หรือ ประมาณ 200 ล้านล้านปี

เทคนิคการเข้าและถอดรหัสที่ถูกพัฒนาขึ้นใหม่ตำแหน่งของบิทใน Permutation IP, IP_1, ตาราง S1-S8, P, E, Permuted Choice 1,2 และตารางการหมุนได้ถูกเปลี่ยนไป หมายเลขกุญแจถูกกำหนดขึ้นใหม่ให้มีความสลับซับซ้อนมากขึ้นโดยใช้ถึง 18 ชุด นั่นหมายถึงต้องทำ Function_F 18 รอบ การเปลี่ยนตารางต่างๆ และ Function_F ให้แตกต่างไปจากเดิมทำให้วิธีการแตกต่างไปจาก DES ในเรื่องของตำแหน่งและค่า Function ต่าง ๆ

ตารางที่ 8 กุญแจขนาด 2^{56} จะใช้เวลาในการจรกรรมข้อมูลเพียง 1 วันนั้นไม่พอเพียงสำหรับความปลอดภัยในยุคปัจจุบัน กุญแจขนาด 2^{112} จะต้องใช้เวลาในการจรกรรมข้อมูลถึง 200 ล้านล้านปี ซึ่งให้ความปลอดภัยสูงเหมาะสำหรับหน่วยงานที่ต้องการรักษาข้อมูลให้เป็นความลับโดยเฉพาะอย่างยิ่งในด้านการทหารและถ้าปกปิดเทคนิคการเข้าและถอดรหัสใหม่นี้จะไม่มีผู้ใดสามารถลักลอบถอดรหัสได้และเทคนิคใหม่เป็นเอกลักษณ์

6. เอกสารอ้างอิง

[1]. Jennifer Seberry, Josef Pieprzyk, "Cryptography An Introduction to Computer Security", Prentice-Hall, 1989.

[2]. H.M. Deitet,"An Introduction to Operating System", Addison-Wesley, 1984, pp. 458-68, 632-46.

[3]. Data Security Through Cryptography", New York: International Business Machines Corporation GC22-9062-0,1986.

[4]. "IBM Cryptographic Subsystem Concepts and Facilities", New York: International Business Machiness Corporation, 1985.

[5]. C.E. Shannon. "The Communicatcn Theory of Secrecy System", Bell System Tech, Vol.28 oct, 1949.

[6]. ร.อ. ชัยยศ ลิลิตวงษ์, อุปกรณ์เข้า-ถอดรหัสข้อมูลในข่ายสื่อสารเครื่องคอมพิวเตอร์, วิทยานิพนธ์ บริญญามหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า พระนครเหนือ,2534

[7]. Charles P.PFLEEGER, "Security in Computing",Prentice-Hall International editions,1989, pp. 22-25, 106-124.

[8]. Martin R. Smith, "Commonsence Computer Security",McGRAW-HILL,1989, PP. 144-150.

[9]. James Arlin Cooper, "Computer & Communications Security",McGRAW-HILL, 1989, pp. 211-212, 240-241, 318-326, 360-361.

ภาคผนวก ค.

ตัวอย่างข้อมูลการเข้าและถอดรหัสและการคำนวณหมายเลขกุญแจด้วยวิธีการเข้าและถอดรหัสของ DES

การคำนวณหมายเลขกุญแจ DES

key[0] : d4659cae
key[1] : 367cd9eb

Co = cde3ba70 Do = 983fec50

ตารางที่ 30 แสดงการคำนวณหมายเลขกุญแจตามขบวนการ DES

ลำดับ	Co	Do	#Key(K)
1	9bc774f0	307fd8b0	2319 173c 2915 3d31
2	378ee9f0	60ffb160	343e 171a 3f35 2810
3	de3ba7c0	83fec590	1f0c 3e33 3414 0d3e
4	78ee9f30	ffb1660	3d3b 240f 251b 2a0c
5	e3ba7cd0	3fec5980	0a3a 1b17 3c03 1b35
6	8ee9f370	ffb16600	1f15 323e 0e3a 2a2f
7	3ba7cde0	fec59830	3d2a 0738 0d27 3613
8	ee9f3780	fb1660f0	272c 3837 0b32 0537
9	dd3e6f10	f62cc1f0	2f1e 3d03 3924 0f33
10	74f9bc70	d8b307f0	1837 1e2f 153a 2d0b
11	d3e6f1d0	62cc1ff0	3e19 1715 3d29 1518
12	4f9bc770	8b307fd0	151c 2b3b 121b 1d2e
13	3e6f1dd0	2cc1ff60	2d3f 0636 1f0f 3228
14	f9bc7740	b307fd80	2f20 3f07 1a05 313f
15	e6f1dd30	cc1ff620	1e37 083d 233f 223a
16	cde3ba70	983fec50	1b1f 332d 2216 3c3b

เข้ารหัสด้วยวิธี DES

```
Input  : a2dc8f5a
Input  : 6ace160c

(1)     Lo == : 3a4ae604
        Ro == : 2711be7d
        E  ==>> 24 0e 22 23 37 3c 0f 3a
        Key==>> 23 19 17 3c 29 15 3d 31
                ------------------------
        XOR==>> 07 17 35 1f 1e 29 32 0b
                ========================

        S1-8 : 4ae999f3

        P  ==>> f31bad45
        f(2711be7d, 23 19 17 3c 29 15 3d 31) = f31bad45
        F()   ==> f31bad45
        Lo    ==> 3a4ae604
                  --------
        XOR ===> c9514b41
                  ========

(2)     Lo == : 2711be7d
        Ro == : c9514b41
        E  ==>> 39 12 2a 22 29 16 28 3f
        Key==>> 34 3e 17 1a 3f 35 28 10
                ------------------------
        XOR==>> 0d 2c 3d 38 16 23 00 2f
                ========================

        S1-8 : dd25f34d
```

```
P  ==>> a9bcf8b6
f(c9514b41, 34 3e 17 1a 3f 35 28 10) = a9bcf8b6
F()   ==> a9bcf8b6
Lo    ==> 2711be7d
          --------
XOR ===> 8ead46cb
         ========
```

(3)

```
 Lo == : c9514b41
 Ro == : 8ead46cb
E  ==>> 31 1d 15 1a 28 0d 19 3f
Key==>> 1f 0c 3e 33 34 14 0d 3e
        -----------------------
XOR==>> 2e 11 2b 29 1c 19 14 01
        =======================

S1-8 : bc9ae091

P  ==>> 07cc0bd3
f(8ead46cb, 1f 0c 3e 33 34 14 0d 3e) = 07cc0bd3
    F()   ==> 07cc0bd3
    Lo    ==> c9514b41
              --------
    XOR ===> ce9d4092
             ========
```

(4)

```
Lo == : 8ead46cb
Ro == : ce9d4092
E  ==>> 39 1d 13 3a 28 01 12 3f
Key==>> 3d 3b 24 0f 25 1b 2a 0c
        -----------------------
XOR==>> 04 26 37 35 0d 1a 38 33
        =======================
```

S1-8 : db35d70c

P ==>> edacf02e

f(ce9d4092, 3d 3b 24 0f 25 1b 2a 0c) = edacf02e

F() ==> edacf02e

Lo ==> 8ead46cb

XOR ===> 6301b6e5

=======

(5)

Lo == : ce9d4092

Ro == : 6301b6e5

E ==>> 2c 06 20 03 36 2d 1c 0a

Key==>> 0a 3a 1b 17 3c 03 1b 35

XOR==>> 26 3c 3b 14 0a 2e 07 3f

=======

S1-8 : 8258a37b

P ==>> 4fb32cc0

f(6301b6e5, 0a 3a 1b 17 3c 03 1b 35) = 4fb32cc0

F() ==> 4fb32cc0

Lo ==> ce9d4092

XOR ===> 812e6c52

=======

(6)

Lo == : 6301b6e5

Ro == : 812e6c52

E ==>> 30 02 25 1c 0d 18 0a 3f

Key==>> 1f 15 32 3e 0e 3a 2a 2f

XOR==>> 2f 17 17 22 03 22 20 10

=======

S1-8 : 7ae6be1a

P ==>> 7b6b938e

f(812e6c52, 1f 15 32 3e 0e 3a 2a 2f) = 7b6b938e

F() ==> 7b6b938e

Lo ==> 6301b6e5

XOR ===> 186a256b

========

(7)

Lo == : 812e6c52

Ro == : 186a256b

E ==>> 23 30 0d 14 04 0a 2d 16

Key==>> 3d 2a 07 38 0d 27 36 13

XOR==>> 1e 1a 0a 2c 09 2d 1b 05

========================

S1-8 : 70374ffd

P ==>> 9e74be2f

f(186a256b, 3d 2a 07 38 0d 27 36 13) = 9e74be2f

F() ==> 9e74be2f

Lo ==> 812e6c52

XOR ===> 1f5ad27d

========

(8)

Lo == : 186a256b

Ro == : 1f5ad27d

E ==>> 23 3e 2b 35 1a 24 0f 3a

Key==>> 27 2c 38 37 0b 32 05 37

```
XOR==>> 04 12 13 02 11 16 0a 0d
        ========================

S1-8 : d78d5407

P  ==>> e086d97a
f(1f5ad27d, 27 2c 38 37 0b 32 05 37) = e086d97a
     F()   ==> e086d97a
     Lo    ==> 186a256b
               --------
      XOR ===> f8ecfc11
               ========
```

(9)

```
Lo == : 1f5ad27d
Ro == : f8ecfc11
E  ==>> 3f 31 1d 19 1f 38 02 3f
Key==>> 2f 1e 3d 03 39 24 0f 33
        -----------------------
XOR==>> 10 2f 20 1a 26 1c 0d 0c
        ========================

S1-8 : 32dcb51b

P  ==>> 6f033bca
f(f8ecfc11, 2f 1e 3d 03 39 24 0f 33) = 6f033bca
     F()   ==> 6f033bca
     Lo    ==> 1f5ad27d
               --------
      XOR ===> 7059e9b7
               ========
```

(10)

```
Lo == : f8ecfc11
Ro == : 7059e9b7
E  ==>> 2e 20 0b 33 3d 13 36 2e
Key==>> 18 37 1e 2f 15 3a 2d 0b
        -----------------------
XOR==>> 36 17 15 1c 28 29 1b 25
        =======================

S1-8 : 7a54a9fe

P  ==>> 5f1bb6a3
f(7059e9b7, 18 37 1e 2f 15 3a 2d 0b) = 5f1bb6a3
     F()   ==> 5f1bb6a3
     Lo    ==> f8ecfc11
               --------
      XOR ===> a7f74ab2
               ========
```

(11)

```
Lo == : 7059e9b7
Ro == : a7f74ab2
E  ==>> 34 0f 3e 2e 29 15 16 3f
Key==>> 3e 19 17 15 3d 29 15 18
        -----------------------
XOR==>> 0a 16 29 3b 14 3c 03 27
        =======================

S1-8 : fd673b07

P  ==>> b0ebfab6
f(a7f74ab2, 3e 19 17 15 3d 29 15 18) = b0ebfab6
      F()   ==> b0ebfab6
      Lo    ==> 7059e9b7
                --------
```

```
        XOR  ===> c0b21301
                  ========
```

(12)

```
Lo == : a7f74ab2
Ro == : c0b21301
E  ==>> 38 01 16 24 02 26 20 3f
Key==>> 15 1c 2b 3b 12 1b 1d 2e
        ------------------------
XOR==>> 2d 1d 3d 1f 10 3d 3d 11
        ========================

S1-8 : 1b29883c

P  ==>> db084466
f(c0b21301, 15 1c 2b 3b 12 1b 1d 2e) = db084466
     F()   ==> db084466
     Lo    ==> a7f74ab2
               --------
      XOR ===> 7cff0ed4
               ========
```

(13)

```
Lo == : c0b21301
Ro == : 7cff0ed4
E  ==>> 0f 39 1f 3e 21 1d 1a 28
Key==>> 2d 3f 06 36 1f 0f 32 28
        ------------------------
XOR==>> 22 06 19 08 3e 12 28 00
        ========================

S1-8 : 1ec0edcd

P  ==>> 591d29bb
f(7cff0ed4, 2d 3f 06 36 1f 0f 32 28) = 591d29bb
```

```
F()    ==> 591d29bb
Lo     ==> c0b21301
           --------
XOR   ===> 99af3aba
           ========
```

(14)

```
Lo == : 7cff0ed4
Ro == : 99af3aba
E  ==>> 33 33 35 1e 27 35 17 3f
Key==>> 2f 20 3f 07 1a 05 31 3f
        ------------------------
XOR==>> 1c 13 0a 19 3d 30 26 00
        ========================

S1-8 : 003157dd

P  ==>> ae34282d
f(99af3aba, 2f 20 3f 07 1a 05 31 3f) = ae34282d
     F()    ==> ae34282d
     Lo     ==> 7cff0ed4
                --------
     XOR ===> d2cb26f9
                ========
```

(15)

```
Lo == : 99af3aba
Ro == : d2cb26f9
E  ==>> 3a 25 19 16 24 0d 1f 3f
Key==>> 1e 37 08 3d 23 3f 22 3a
        ------------------------
XOR==>> 24 12 11 2b 07 32 3d 05
        ========================

S1-8 : e721c03d
```

```
P  ==>> cb84ce34
f(d2cb26f9, 1e 37 08 3d 23 3f 22 3a) = cb84ce34
   F()   ==> cb84ce34
    Lo   ==> 99af3aba
             --------
     XOR ===> 522bf48e
             ========
```

(16)

```
Lo == : d2cb26f9
Ro == : 522bf48e
E  ==>> 0a 24 05 17 3e 29 11 1c
Key==>> 1b 1f 33 2d 22 16 3c 3b
        ------------------------
XOR==>> 11 3b 36 3a 1c 3f 2d 27
        ========================

S1-8 : a5c2eda7

P  ==>> 11c76fb9
f(522bf48e, 1b 1f 33 2d 22 16 3c 3b) = 11c76fb9
    F()   ==> 11c76fb9
   Lo     ==> d2cb26f9
             --------
    XOR ===> c30c4940
             ========

Lo == : c30c4940
Ro == : 522bf48e
AFTER  MAT_IP-1 : 64e21a36
AFTER  MAT_IP-1 : 8828cd4a
```

จบการเข้ารหัสด้วยวิธี DES

ถอดรหัสด้วยวิธี DES

```
Input  : 64e21a36
Input  : 8828cd4a

(16)    Lo == : c30c4940
        Ro == : 522bf48e
        E  ==>> 0a 24 05 17 3e 29 11 1c
        Key==>> 1b 1f 33 2d 22 16 3c 3b
                ------------------------
        XOR==>> 11 3b 36 3a 1c 3f 2d 27
                ========================

        S1-8 : a5c2eda7

        P  ==>> 11c76fb9
        f(522bf48e, 1b 1f 33 2d 22 16 3c 3b) = 11c76fb9
           F()   ==> 11c76fb9
           Lo    ==> c30c4940
                     --------
           XOR ==> d2cb26f9
                     ========

(15)    Lo == : 522bf48e
        Ro == : d2cb26f9
        E  ==>> 3a 25 19 16 24 0d 1f 3f
        Key==>> 1e 37 08 3d 23 3f 22 3a
                ------------------------
        XOR==>> 24 12 11 2b 07 32 3d 05
                ========================

        S1-8 : e721c03d
```

```
P  ==>> cb84ce34
f(d2cb26f9, 1e 37 08 3d 23 3f 22 3a) = cb84ce34
    F()   ==> cb84ce34
    Lo    ==> 522bf48e
              --------
     XOR ===> 99af3aba
              ========
```

(14)

```
Lo == : d2cb26f9
Ro == : 99af3aba
E  ==>> 33 33 35 1e 27 35 17 3f
Key==>> 2f 20 3f 07 1a 05 31 3f
        -----------------------
XOR==>> 1c 13 0a 19 3d 30 26 00
        =======================

S1-8 : 003157dd

P  ==>> ae34282d
f(99af3aba, 2f 20 3f 07 1a 05 31 3f) = ae34282d
    F()   ==> ae34282d
    Lo    ==> d2cb26f9
              --------
     XOR ===> 7cff0ed4
              ========
```

(13)

```
Lo == : 99af3aba
Ro == : 7cff0ed4
E  ==>> 0f 39 1f 3e 21 1d 1a 28
Key==>> 2d 3f 06 36 1f 0f 32 28
        -----------------------
XOR==>> 22 06 19 08 3e 12 28 00
        =======================
```

```
S1-8 : 1ec0edcd

P  ==>> 591d29bb
f(7cff0ed4, 2d 3f 06 36 1f 0f 32 28) = 591d29bb
    F()   ==> 591d29bb
    Lo    ==> 99af3aba
              --------
     XOR ===> c0b21301
              ========
```

(12)

```
Lo == : 7cff0ed4
Ro == : c0b21301
E  ==>> 38 01 16 24 02 26 20 3f
Key==>> 15 1c 2b 3b 12 1b 1d 2e
        -----------------------
XOR==>> 2d 1d 3d 1f 10 3d 3d 11
        =======================

S1-8 : 1b29883c

P  ==>> db084466
f(c0b21301, 15 1c 2b 3b 12 1b 1d 2e) = db084466
    F()   ==> db084466
    Lo    ==> 7cff0ed4
              --------
     XOR ===> a7f74ab2
              ========
```

(11)

```
Lo == : c0b21301
Ro == : a7f74ab2
E  ==>> 34 0f 3e 2e 29 15 16 3f
Key==>> 3e 19 17 15 3d 29 15 18
        -----------------------
```

```
XOR==>> 0a 16 29 3b 14 3c 03 27
        ========================

S1-8 : fd673b07

P  ==>> b0ebfab6
f(a7f74ab2, 3e 19 17 15 3d 29 15 18) = b0ebfab6
    F()   ==> b0ebfab6
    Lo    ==> c0b21301
                --------
     XOR ===> 7059e9b7
                ========

(10)    Lo == : a7f74ab2
        Ro == : 7059e9b7
        E  ==>> 2e 20 0b 33 3d 13 36 2e
        Key==>> 18 37 1e 2f 15 3a 2d 0b
                ------------------------
        XOR==>> 36 17 15 1c 28 29 1b 25
                ========================

        S1-8 : 7a54a9fe

        P  ==>> 5f1bb6a3
        f(7059e9b7, 18 37 1e 2f 15 3a 2d 0b) = 5f1bb6a3
            F()   ==> 5f1bb6a3
            Lo    ==> a7f74ab2
                      --------
             XOR ===> f8ecfc11
                      ========
```

```
(9)     Lo == : 7059e9b7
        Ro == : f8ecfc11
        E  ==>> 3f 31 1d 19 1f 38 02 3f
        Key==>> 2f 1e 3d 03 39 24 0f 33
                -----------------------
        XOR==>> 10 2f 20 1a 26 1c 0d 0c
                =======================

        S1-8 : 32dcb51b

        P  ==>> 6f033bca
        f(f8ecfc11, 2f 1e 3d 03 39 24 0f 33) = 6f033bca
            F()   ==> 6f033bca
            Lo    ==> 7059e9b7
                      --------
             XOR ===> 1f5ad27d
                      ========

(8)     Lo == : f8ecfc11
        Ro == : 1f5ad27d
        E  ==>> 23 3e 2b 35 1a 24 0f 3a
        Key==>> 27 2c 38 37 0b 32 05 37
                -----------------------
        XOR==>> 04 12 13 02 11 16 0a 0d
                =======================

        S1-8 : d78d5407

        P  ==>> e086d97a
        f(1f5ad27d, 27 2c 38 37 0b 32 05 37) = e086d97a
        F()   ==> e086d97a
        Lo    ==> f8ecfc11
                  --------
```

```
          XOR ===> 186a256b
                   ========

(7)       Lo == : 1f5ad27d
          Ro == : 186a256b
          E  ==>> 23 30 0d 14 04 0a 2d 16
          Key==>> 3d 2a 07 38 0d 27 36 13
                  ------------------------
          XOR==>> 1e 1a 0a 2c 09 2d 1b 05
                  ========================

          S1-8 : 70374ffd

          P  ==>> 9e74be2f
          f(186a256b, 3d 2a 07 38 0d 27 36 13) = 9e74be2f
             F()   ==> 9e74be2f
             Lo    ==> 1f5ad27d
                       --------
              XOR ===> 812e6c52
                       ========

(6)       Lo == : 186a256b
          Ro == : 812e6c52
          E  ==>> 30 02 25 1c 0d 18 0a 3f
          Key==>> 1f 15 32 3e 0e 3a 2a 2f
                  ------------------------
          XOR==>> 2f 17 17 22 03 22 20 10
                  ========================

          S1-8 : 7ae6be1a

          P  ==>> 7b6b938e
          f(812e6c52, 1f 15 32 3e 0e 3a 2a 2f) = 7b6b938e
```

```
    F()    ==> 7b6b938e
    Lo     ==> 186a256b
               --------
     XOR ===> 6301b6e5
               ========
```

(5)

```
Lo == : 812e6c52
            Ro == : 6301b6e5
E  ==>> 2c 06 20 03 36 2d 1c 0a
Key==>> 0a 3a 1b 17 3c 03 1b 35
        ------------------------
XOR==>> 26 3c 3b 14 0a 2e 07 3f
        ========================

S1-8 : 8258a37b

P  ==>> 4fb32cc0
f(6301b6e5, 0a 3a 1b 17 3c 03 1b 35) = 4fb32cc0
    F()    ==> 4fb32cc0
    Lo     ==> 812e6c52
               --------
     XOR ===> ce9d4092
               ========
```

(4)

```
Lo == : 6301b6e5
Ro == : ce9d4092
E  ==>> 39 1d 13 3a 28 01 12 3f
Key==>> 3d 3b 24 0f 25 1b 2a 0c
        ------------------------
XOR==>> 04 26 37 35 0d 1a 38 33
        ========================

S1-8 : db35d70c
```

```
P  ==>> edacf02e
f(ce9d4092, 3d 3b 24 0f 25 1b 2a 0c) = edacf02e
    F()   ==> edacf02e
    Lo    ==> 6301b6e5
              --------
     XOR ===> 8ead46cb
              ========
```

(3)

```
Lo == : ce9d4092
Ro == : 8ead46cb
E  ==>> 31 1d 15 1a 28 0d 19 3f
Key==>> 1f 0c 3e 33 34 14 0d 3e
        ------------------------
XOR==>> 2e 11 2b 29 1c 19 14 01
        ========================

S1-8 : bc9ae091

P  ==>> 07cc0bd3
f(8ead46cb, 1f 0c 3e 33 34 14 0d 3e) = 07cc0bd3
    F()   ==> 07cc0bd3
    Lo    ==> ce9d4092
              --------
     XOR ===> c9514b41
              ========
```

(2)

```
Lo == : 8ead46cb
Ro == : c9514b41
E  ==>> 39 12 2a 22 29 16 28 3f
Key==>> 34 3e 17 1a 3f 35 28 10
        ------------------------
XOR==>> 0d 2c 3d 38 16 23 00 2f
        ========================
```

```
S1-8 : dd25f34d

P  ==>> a9bcf8b6
f(c9514b41, 34 3e 17 1a 3f 35 28 10) = a9bcf8b6
    F()   ==> a9bcf8b6
    Lo    ==> 8ead46cb
              --------
     XOR ===> 2711be7d
              ========

(1)  Lo == : c9514b41
     Ro == : 2711be7d
     E  ==>> 24 0e 22 23 37 3c 0f 3a
     Key==>> 23 19 17 3c 29 15 3d 31
             -----------------------
     XOR==>> 07 17 35 1f 1e 29 32 0b
             =======================

     S1-8 : 4ae999f3

     P  ==>> f31bad45
     f(2711be7d, 23 19 17 3c 29 15 3d 31) = f31bad45
         F()   ==> f31bad45
         Lo    ==> c9514b41
                   --------
          XOR ===> 3a4ae604
                   ========
      Lo == : 3a4ae604
      Ro == : 2711be7d
      AFTER  MAT_IP-1 : a2dc8f5a
      AFTER  MAT_IP-1 : 6ace160c
```

จบการถอดรหัสด้วยวิธี DES

ภาคผนวก ง

ตัวอย่างข้อมูลการเข้าและถอดรหัสและการคำนวณหมายเลขกุญแจด้วยวิธีการเข้าและถอดรหัสที่สร้างขึ้นใหม่

การคำนวณหมายเลขกุญแจด้วยวิธีที่สร้างขึ้น (ขนาด 112 บิท)

กุญแจชุดที่ 1 : d4659cae
: 367cd9eb

กุญแจชุดที่ 2 : 748c238a
: bcef9876

นำกุญแจชุดที่ 1 มาเข้า PC_1 จะได้ ee35 33e0 76d6 b170

ตารางที่ 31 แสดงตารางการคำนวณหมายเลขกุญแจ K1 ถึง K9

ลำดับ	Co	Do	#KEY(K)
1	b8d4cfb0	db5ac5d0	1f3b22293f39111f
2	8d4cfbb0	b5ac5dd0	191f1a171b353513
3	d4cfbb80	5ac5ddb0	3d3e0d0f2a3b3c04
4	4cfbb8d0	ac5ddb50	2a371f0630341f2e
5	67ddc6a0	62eedad0	283e2e3e372d320d
6	cfbb8d40	c5ddb5a0	23311d37381b283e
7	fbb8d4c0	5ddb5ac0	1e3736323c092a39
8	eee35330	776d6b10	392d3d0c392c1b31
9	ee3533e0	76d6b170	332b3e0e0f2e242c

กุญแจชุดที่ 2 : 748c238a
: bcef9876

นำกุญแจชุดที่ 2 มาเข้า PC_1 จะได้ 4a78 ee20 789b 99f0

ตารางที่ 32 แสดงตารางการคำนวณหมายเลขกุญแจ K10 ถึง K18

ลำดับ	Co	Do	#KEY(K)
10	53c77120	c4dccfb0	2f241515332b3607
11	3c771250	4dccfbc0	2c2d3b061209333c
12	8ee24a70	b99f7890	1a0d09063d353933
13	3b8929e0	e67de260	2f23321c311e0e06
14	71253c70	cfbc4dc0	2e04363d13192532
15	1253c770	fbc4dcc0	3d3c032f010d311f
16	24a78ee0	f789b990	301e392a333d2836
17	253c7710	bc4dccf0	192c3a3512270d2f
18	4a78ee20	789b99f0	121d0622153f3c2d

ตารางที่ 33 ตาราง S_box ใช้ในการเข้าและถอดรหัสด้วยวิธีการที่สร้างขึ้น

	Column																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	3	5	11	15	12	15	6	1	9	13	7	14	8	4	10	2	
1	7	9	3	16	1	6	2	8	15	14	4	12	10	5	11	13	S1
2	13	12	1	4	14	5	2	11	3	15	16	8	10	9	6	7	
3	6	5	8	7	4	9	16	1	10	2	15	14	3	12	11	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	S2
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	1	
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	S3
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	S4
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	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	S5
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	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	S6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	10	11	14	1	7	6	0	8	13	15	
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	S7
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	
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	S8
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	

เริ่มเข้ารหัสด้วยวิธีที่สร้างขึ้น

```
   Input endes[0] : a2dc8f5a
   Input endes[1] : 6ace160c

(1)    Lo == : 48e2d242
       Ro == : 3d2db65b
       E  ==>> 3a 36 1b 2c 36 0b 27 25
       Key==>> 1f 3b 22 29 3f 39 11 1f
               ------------------------
       XOR==>> 25 0d 39 05 09 32 36 3a
               ========================

       S1-8 : 81bbf0f0

       P  ==>> 96bac613
       f(3d2db65b, 1f 3b 22 29 3f 39 11 1f) = 96bac613
       F()   ==> 96bac613
       Lo    ==> 48e2d242
                 --------
        XOR ===> de581451
                   ========

(2)    Lo == : 3d2db65b
       Ro == : de581451
       E  ==>> 3c 3f 30 28 02 0a 3b 0b
       Key==>> 19 1f 1a 17 1b 35 35 13
               ------------------------
       XOR==>> 25 20 2a 3f 19 3f 0e 18
               ========================

       S1-8 : 81fe6da0
```

```
P  ==>> be064717
f(de581451, 19 1f 1a 17 1b 35 35 13) = be064717
     F()   ==> be064717
     Lo    ==> 3d2db65b
               --------
      XOR ===> 832bf14c
               ========
```

(3)

```
Lo == : de581451
Ro == : 832bf14c
E  ==>> 06 3f 17 22 3e 29 30 25
Key==>> 3d 3e 0d 0f 2a 3b 3c 04
        -----------------------
XOR==>> 3b 01 1a 2d 14 12 0c 21
        =======================

S1-8 : cd4d2d18

P  ==>> 3a65a127
f(832bf14c, 3d 3e 0d 0f 2a 3b 3c 04) = 3a65a127
     F()   ==> 3a65a127
     Lo    ==> de581451
               --------
      XOR ===> e43db576
               ========
```

(4)

```
Lo == : 832bf14c
Ro == : e43db576
E  ==>> 08 3f 3b 2a 36 2e 3c 07
Key==>> 2a 37 1f 06 30 34 1f 2e
        -----------------------
XOR==>> 22 08 24 2c 06 1a 23 29
        =======================
```

```
S1-8 : cf47e7ee

P  ==>> 5fcdeb3d
f(e43db576, 2a 37 1f 06 30 34 1f 2e) = 5fcdeb3d
    F()   ==> 5fcdeb3d
    Lo    ==> 832bf14c
              --------
     XOR ===> dce61a71
              ========
```

(5)

```
Lo == : e43db576
Ro == : dce61a71
E  ==>> 39 3f 0c 34 03 0e 3b 1c
Key==>> 28 3e 2e 3e 37 2d 32 0d
        -----------------------
XOR==>> 11 01 22 0a 34 23 09 11
        =======================

S1-8 : fd66c330
P  ==>> 1fe53790
f(dce61a71, 28 3e 2e 3e 37 2d 32 0d) = 1fe53790
    F()   ==> 1fe53790
    Lo    ==> e43db576
              --------
     XOR ===> fbd882e6
              ========
```

(6)

```
Lo == : dce61a71
Ro == : fbd882e6
E  ==>> 37 3f 31 05 10 1c 3f 3b
Key==>> 23 31 1d 37 38 1b 28 3e
        -----------------------
XOR==>> 14 0e 2c 32 28 07 17 05
        =======================
```

```
S1-8 : 7d31a20c

P  ==>> 11c3bca1
f(fbd882e6, 23 31 1d 37 38 1b 28 3e) = 11c3bca1
    F()   ==> 11c3bca1
    Lo    ==> dce61a71
              --------
     XOR ===> cd25a6d0
              ========
```

(7)

```
Lo == : fbd882e6
Ro == : cd25a6d0
E  ==>> 1a 3f 0b 0d 34 1a 39 24
Key==>> 1e 37 36 32 3c 09 2a 39
        ------------------------
XOR==>> 04 08 3d 3f 08 13 13 1d
        ========================

S1-8 : bf2e6166

P  ==>> 564c3f9b
f(cd25a6d0, 1e 37 36 32 3c 09 2a 39) = 564c3f9b
    F()   ==> 564c3f9b
    Lo    ==> fbd882e6
              --------
     XOR ===> ad94bd7d
              ========
```

(8)

```
Lo == : cd25a6d0
Ro == : ad94bd7d
E  ==>> 1b 3f 29 3a 17 2f 35 32
Key==>> 39 2d 3d 0c 39 2c 1b 31
        ------------------------
```

```
XOR==>> 22 12 14 36 2e 03 2e 03
        ========================

S1-8 : ccce1f02

P  ==>> ab55211e
f(ad94bd7d, 39 2d 3d 0c 39 2c 1b 31) = ab55211e
    F()   ==> ab55211e
    Lo    ==> cd25a6d0
                --------
     XOR ===> 667087ce
                ========
```

(9)

```
Lo == : ad94bd7d
Ro == : 667087ce
E  ==>> 0c 1c 21 0f 10 39 0c 0e
Key==>> 33 2b 3e 0e 0f 2e 24 2c
        ------------------------
XOR==>> 3f 37 1f 01 1f 17 28 22
        ========================

S1-8 : df1d5ec1

P  ==>> 775bf146
f(667087ce, 33 2b 3e 0e 0f 2e 24 2c) = 775bf146
    F()   ==> 775bf146
    Lo    ==> ad94bd7d
                --------
     XOR ===> dacf4c3b
                ========
```

(10)

```
Lo == : 667087ce
Ro == : dacf4c3b
```

```
E  ==>> 35 3f 1e 18 29 07 3b 19
Key==>> 2f 24 15 15 33 2b 36 07
        -----------------------
XOR==>> 1a 1b 0b 0d 1a 2c 0d 1e
        =======================

S1-8 : 4f400c4e

P  ==>> 7849282c
f(dacf4c3b, 2f 24 15 15 33 2b 36 07) = 7849282c
    F()   ==> 7849282c
    Lo    ==> 667087ce
              --------
     XOR ===> 1e39afe2
              ========

(11)  Lo == : dacf4c3b
      Ro == : 1e39afe2
      E  ==>> 3c 04 33 1f 35 3c 03 07
      Key==>> 2c 2d 3b 06 12 09 33 3c
              -----------------------
      XOR==>> 10 29 08 19 27 35 30 3b
              =======================

      S1-8 : 91611765

      P  ==>> 1b1c9e44
      f(1e39afe2, 2c 2d 3b 06 12 09 33 3c) = 1b1c9e44
          F()   ==> 1b1c9e44
          Lo    ==> dacf4c3b
                    --------
           XOR ===> c1d3d27f
                    ========
```

(12)

```
Lo == : 1e39afe2
Ro == : c1d3d27f
E  ==>> 03 3f 27 24 3a 0f 38 3a
Key==>> 1a 0d 09 06 3d 35 39 33
        -----------------------
XOR==>> 19 32 2e 22 07 3a 01 09
        =======================

S1-8 : af067df7

P  ==>> 767c6bdd
f(c1d3d27f, 1a 0d 09 06 3d 35 39 33) = 767c6bdd
    F()   ==> 767c6bdd
    Lo    ==> 1e39afe2
              --------
     XOR ===> 6845c43f
              ========
```

(13)

```
Lo == : c1d3d27f
Ro == : 6845c43f
E  ==>> 10 3e 0b 08 38 07 2d 08
Key==>> 2f 23 32 1c 31 1e 0e 06
        -----------------------
XOR==>> 3f 1d 39 14 09 19 23 0e
        =======================

S1-8 : d8b8f0e8

P  ==>> 869b7623
f(6845c43f, 2f 23 32 1c 31 1e 0e 06) = 869b7623
    F()   ==> 869b7623
    Lo    ==> c1d3d27f
              --------
```

```
          XOR ===> 4748a45c
                   ========

(14)      Lo == : 6845c43f
          Ro == : 4748a45c
          E  ==>> 0e 38 11 08 14 0b 08 29
          Key==>> 2e 04 36 3d 13 19 25 32
                  ------------------------
          XOR==>> 20 3c 27 35 07 12 2d 1b
                  ========================

          S1-8 : d9057dd8

          P  ==>> 363df125
          f(4748a45c, 2e 04 36 3d 13 19 25 32) = 363df125
             F()   ==> 363df125
             Lo    ==> 6845c43f
                       --------
              XOR ===> 5e78351a
                       ========

(15)      Lo == : 4748a45c
          Ro == : 5e78351a
          E  ==>> 3c 34 30 2a 06 23 0b 0f
          Key==>> 3d 3c 03 2f 01 0d 31 1f
                  ------------------------
          XOR==>> 01 08 33 05 07 2e 3a 10
                  ========================

          S1-8 : 7ffb7355
          P  ==>> dd7fbcd3
          f(5e78351a, 3d 3c 03 2f 01 0d 31 1f) = dd7fbcd3
              F()   ==> dd7fbcd3
```

```
        Lo     ==> 4748a45c
                   --------
        XOR ===> 9a37188f
                   ========
```

(16)

```
Lo == : 5e78351a
Ro == : 9a37188f
E  ==>> 34 3f 2e 31 23 11 33 06
Key==>> 30 1e 39 2a 33 3d 28 36
        ------------------------
XOR==>> 04 21 17 1b 10 2c 1b 30
        ========================

S1-8 : b6ea9c99

P  ==>> eaf054f6
f(9a37188f, 30 1e 39 2a 33 3d 28 36) = eaf054f6
    F()    ==> eaf054f6
    Lo     ==> 5e78351a
               --------
    XOR ===> b48861ec
               ========
```

(17)

```
Lo == : 9a37188f
Ro == : b48861ec
E  ==>> 29 3f 10 03 0c 3d 36 11
Key==>> 19 2c 3a 35 12 27 0d 2f
        ------------------------
XOR==>> 30 13 2a 36 1e 1a 3b 3e
        ========================

S1-8 : 33fea76b
```

```
P  ==>> d98e17ff
f(b48861ec, 19 2c 3a 35 12 27 0d 2f) = d98e17ff
    F()  ==> d98e17ff
    Lo   ==> 9a37188f
             --------
     XOR ===> 43b90f70
             ========
```

(18)

```
Lo == : b48861ec
Ro == : 43b90f70
E  ==>> 07 20 32 1e 21 2e 08 37
Key==>> 12 1d 06 22 15 3f 3c 2d
        ------------------------
XOR==>> 15 3d 34 3c 34 11 34 1a
        ========================

S1-8 : 4328c6de

P  ==>> 55a94c2e
f(43b90f70, 12 1d 06 22 15 3f 3c 2d) = 55a94c2e
    F()  ==> 55a94c2e
    Lo   ==> b48861ec
             --------
     XOR ===> e1212dc2
             ========

Lo == : e1212dc2
Ro == : 43b90f70
AFTER  MAT_IP-1 : ba741cc8
AFTER  MAT_IP-1 : 9570a4a1
```

จบการเข้ารหัสด้วยวิธีที่สร้างขึ้น

เริ่มถอดรหัสด้วยวิธีที่สร้างขึ้น

```
Input   : ba741cc8
Input   : 9570a4a1

(18)    Lo == : e1212dc2
        Ro == : 43b90f70
        E  ==>> 07 20 32 1e 21 2e 08 37
        Key==>> 12 1d 06 22 15 3f 3c 2d
                ------------------------
        XOR==>> 15 3d 34 3c 34 11 34 1a
                ========================

        S1-8 : 4328c6de

        P  ==>> 55a94c2e
        f(43b90f70, 12 1d 06 22 15 3f 3c 2d) = 55a94c2e
           F()   ==> 55a94c2e
           Lo    ==> e1212dc2
                     --------
           XOR ===> b48861ec
                     ========

(17)    Lo == : 43b90f70
        Ro == : b48861ec
        E  ==>> 29 3f 10 03 0c 3d 36 11
        Key==>> 19 2c 3a 35 12 27 0d 2f
                ------------------------
        XOR==>> 30 13 2a 36 1e 1a 3b 3e
                ========================

        S1-8 : 33fea76b
```

```
P  ==>> d98e17ff
f(b48861ec, 19 2c 3a 35 12 27 0d 2f) = d98e17ff
    F()   ==> d98e17ff
    Lo    ==> 43b90f70
              --------
     XOR ===> 9a37188f
              ========
```

(16)

```
Lo == : b48861ec
Ro == : 9a37188f
E  ==>> 34 3f 2e 31 23 11 33 06
Key==>> 30 1e 39 2a 33 3d 28 36
        ------------------------
XOR==>> 04 21 17 1b 10 2c 1b 30
        ========================

 S1-8 : b6ea9c99

P  ==>> eaf054f6
f(9a37188f, 30 1e 39 2a 33 3d 28 36) = eaf054f6
    F()   ==> eaf054f6
    Lo    ==> b48861ec
              --------
     XOR ===> 5e78351a
              ========
```

(15)

```
Lo == : 9a37188f
Ro == : 5e78351a
E  ==>> 3c 34 30 2a 06 23 0b 0f
Key==>> 3d 3c 03 2f 01 0d 31 1f
        ------------------------
XOR==>> 01 08 33 05 07 2e 3a 10
        ========================
```

S1-8 : 7ffb7355

P ==>> dd7fbcd3
f(5e78351a, 3d 3c 03 2f 01 0d 31 1f) = dd7fbcd3
F() ==> dd7fbcd3
Lo ==> 9a37188f

XOR ===> 4748a45c
========

(14)

Lo == : 5e78351a
Ro == : 4748a45c
E ==>> 0e 38 11 08 14 0b 08 29
Key==>> 2e 04 36 3d 13 19 25 32

XOR==>> 20 3c 27 35 07 12 2d 1b
========================

S1-8 : d9057dd8

P ==>> 363df125
f(4748a45c, 2e 04 36 3d 13 19 25 32) = 363df125
F() ==> 363df125
Lo ==> 5e78351a

XOR ===> 6845c43f
========

(13)

Lo == : 4748a45c
Ro == : 6845c43f
E ==>> 10 3e 0b 08 38 07 2d 08
Key==>> 2f 23 32 1c 31 1e 0e 0

```
XOR==>> 3f 1d 39 14 09 19 23 0e
        ========================

S1-8 : d8b8f0e8

P  ==>> 869b7623
f(6845c43f, 2f 23 32 1c 31 1e 0e 06) = 869b7623
    F()   ==> 869b7623
    Lo    ==> 4748a45c
              --------
     XOR ===> c1d3d27f
              ========
(12)
Lo == : 6845c43f
Ro == : c1d3d27f
E  ==>> 03 3f 27 24 3a 0f 38 3a
Key==>> 1a 0d 09 06 3d 35 39 33
        ------------------------
XOR==>> 19 32 2e 22 07 3a 01 09
        ========================

S1-8 : af067df7

P  ==>> 767c6bdd
f(c1d3d27f, 1a 0d 09 06 3d 35 39 33) = 767c6bdd
    F()   ==> 767c6bdd
    Lo    ==> 6845c43f
              --------
     XOR ===> 1e39afe2
              ========

(11)
Lo == : c1d3d27f
Ro == : 1e39afe2
E  ==>> 3c 04 33 1f 35 3c 03 07
```

```
Key==>> 2c 2d 3b 06 12 09 33 3c
        ------------------------
XOR==>> 10 29 08 19 27 35 30 3b
        ========================

 S1-8 : 91611765

P  ==>> 1b1c9e44
f(1e39afe2, 2c 2d 3b 06 12 09 33 3c) = 1b1c9e44
    F()   ==> 1b1c9e44
    Lo    ==> c1d3d27f
              --------
     XOR ===> dacf4c3b
              ========

(10)
Lo == : 1e39afe2
Ro == : dacf4c3b
E  ==>> 35 3f 1e 18 29 07 3b 19
Key==>> 2f 24 15 15 33 2b 36 07
        ------------------------
XOR==>> 1a 1b 0b 0d 1a 2c 0d 1e
        ========================

S1-8 : 4f400c4e

P  ==>> 7849282c
f(dacf4c3b, 2f 24 15 15 33 2b 36 07) = 7849282c
    F()   ==> 7849282c
    Lo    ==> 1e39afe2
              --------
     XOR ===> 667087ce
              ========
```

(9)

```
Lo == : dacf4c3b
Ro == : 667087ce
E  ==>> 0c 1c 21 0f 10 39 0c 0e
Key==>> 33 2b 3e 0e 0f 2e 24 2c
        ------------------------
XOR==>> 3f 37 1f 01 1f 17 28 22
        ========================

S1-8 : df1d5ec1

P  ==>> 775bf146
f(667087ce, 33 2b 3e 0e 0f 2e 24 2c) = 775bf146
    F()   ==> 775bf146
    Lo    ==> dacf4c3b
              --------
     XOR ===> ad94bd7d
              ========
```

(8)

```
Lo == : 667087ce
Ro == : ad94bd7d
E  ==>> 1b 3f 29 3a 17 2f 35 32
Key==>> 39 2d 3d 0c 39 2c 1b 31
        ------------------------
XOR==>> 22 12 14 36 2e 03 2e 03
        ========================

S1-8 : ccce1f02

P  ==>> ab55211e
f(ad94bd7d, 39 2d 3d 0c 39 2c 1b 31) = ab55211e
    F()   ==> ab55211e
    Lo    ==> 667087ce
              --------
```

```
        XOR ===> cd25a6d0
                 ========

(7)     Lo == : ad94bd7d
        Ro == : cd25a6d0
        E  ==>> 1a 3f 0b 0d 34 1a 39 24
        Key==>> 1e 37 36 32 3c 09 2a 39
                ------------------------
        XOR==>> 04 08 3d 3f 08 13 13 1d
                ========================

        S1-8 : bf2e6166

        P  ==>> 564c3f9b
        f(cd25a6d0, 1e 37 36 32 3c 09 2a 39) = 564c3f9b
          F()   ==> 564c3f9b
          Lo    ==> ad94bd7d
                    --------
          XOR ===> fbd882e6
                   ========

(6)     Lo == : cd25a6d0
        Ro == : fbd882e6
        E  ==>> 37 3f 31 05 10 1c 3f 3b
        Key==>> 23 31 1d 37 38 1b 28 3e
                ------------------------
        XOR==>> 14 0e 2c 32 28 07 17 05
                ========================

        S1-8 : 7d31a20c

        P  ==>> 11c3bca1
        f(fbd882e6, 23 31 1d 37 38 1b 28 3e) = 11c3bca1
```

```
        F()    ==> 11c3bca1
        Lo     ==> cd25a6d0
                   --------
         XOR ===> dce61a71
                   ========
```

(5)

```
Lo == : fbd882e6
Ro == : dce61a71
E  ==>> 39 3f 0c 34 03 0e 3b 1c
Key==>> 28 3e 2e 3e 37 2d 32 0d
        ------------------------
XOR==>> 11 01 22 0a 34 23 09 11
        ========================

S1-8 : fd66c330

P  ==>> 1fe53790
f(dce61a71, 28 3e 2e 3e 37 2d 32 0d) = 1fe53790
   F()    ==> 1fe53790
   Lo     ==> fbd882e6
              --------
    XOR ===> e43db576
              ========
```

(4)

```
Lo == : dce61a71
Ro == : e43db576
E  ==>> 08 3f 3b 2a 36 2e 3c 07
Key==>> 2a 37 1f 06 30 34 1f 2e
        ------------------------
XOR==>> 22 08 24 2c 06 1a 23 29
        ========================

S1-8 : cf47e7ee
```

```
P  ==>> 5fcdeb3d
f(e43db576, 2a 37 1f 06 30 34 1f 2e) = 5fcdeb3d
    F()   ==> 5fcdeb3d
    Lo    ==> dce61a71
              --------
     XOR ===> 832bf14c
              ========
```

(3)

```
Lo == : e43db576
Ro == : 832bf14c
E  ==>> 06 3f 17 22 3e 29 30 25
Key==>> 3d 3e 0d 0f 2a 3b 3c 04
        ------------------------
XOR==>> 3b 01 1a 2d 14 12 0c 21
        ========================

S1-8 : cd4d2d18

P  ==>> 3a65a127
f(832bf14c, 3d 3e 0d 0f 2a 3b 3c 04) = 3a65a127
    F()   ==> 3a65a127
    Lo    ==> e43db576
              --------
     XOR ===> de581451
              ========
```

(2)

```
Lo == : 832bf14c
Ro == : de581451
E  ==>> 3c 3f 30 28 02 0a 3b 0b
Key==>> 19 1f 1a 17 1b 35 35 13
        ------------------------
XOR==>> 25 20 2a 3f 19 3f 0e 18
        ========================
S1-8 : 81fe6da0
```

```
P  ==>> be064717
f(de581451, 19 1f 1a 17 1b 35 35 13) = be064717
    F()   ==> be064717
    Lo    ==> 832bf14c
              --------
     XOR ===> 3d2db65b
              ========
```

(1)

```
Lo == : de581451
Ro == : 3d2db65b
E  ==>> 3a 36 1b 2c 36 0b 27 25
Key==>> 1f 3b 22 29 3f 39 11 1f
        ------------------------
XOR==>> 25 0d 39 05 09 32 36 3a
        ========================

S1-8 : 81bbf0f0
P  ==>> 96bac613
f(3d2db65b, 1f 3b 22 29 3f 39 11 1f) = 96bac613
   F()   ==> 96bac613
   Lo    ==> de581451
             --------
    XOR ===> 48e2d242
             ========
Lo == : 48e2d242
Ro == : 3d2db65b
AFTER  MAT_IP-1 : a2dc8f5a
AFTER  MAT_IP-1 : 6ace160c
```

จบการเข้ารหัสด้วยวิธีที่สร้างขึ้น

แฟ้มข้อมูล key1.t ถึง key6.t มีลักษณะดังนี้

```
******* key1.t *******

 d455 9cae 367c d9eb
 a548 8428 c6e2 48ea

******** key2.t *******

 d455 acae 367c d9eb
 a548 8428 c6e2 48ea

******** key3.t *******

 d455 acae 367c d9eb
 a548 8428 16e2 48ea

******** key4.t *******

 d455 acae 367c b9eb
 a548 9428 16e2 48ea


******** key5.t *******

 a148 8498 c6d2 48ea
 a465 acae 367c b9eb
```

ภาคผนวก จ.

ตัวอย่างแฟ้มข้อมูลที่เข้าและถอดรหัสด้วยวิธีการเข้าและถอดรหัสด้วยขบวนการ DES และขบวนการที่สร้างขึ้นใหม่โดยใช้หมายเลขกุญแจต่างกัน

แฟ้ม test.t ใช้เป็นแฟ้มข้อมูลในการเข้ารหัส

1.บทนำ

เนื่องจากการพัฒนาด้านคอมพิวเตอร์ เริ่มเข้ามามีความสำคัญต่อการใช้งาน ในสังคมปัจจุบันการเก็บข้อมูลในคอมพิวเตอร์หรือการติดต่อสื่อสารในระบบคอมพิวเตอร์เป็นเรื่องสำคัญที่บางครั้งองค์กรนั้น ๆจะต้องปกปิดเป็นความลับโดยมีวิธีการเข้าถึงข้อมูลโดยใช้รหัสผ่าน (PASSWORD) เพื่อจำกัดสิทธิ์ความสำคัญของผู้ใช้ การใช้วิธี Copy Protection เช่น เกมส์หลาย ๆ อย่างใช้กันในเครื่องไมโครคอมพิวเตอร์เพื่อกันการก็อปปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือการเข้ารหัสข้อมูล (Crypto graphy หรือ Data Encryption) เพื่อให้ข้อมูลเปลี่ยนไปจากเดิมจนไม่สามารถเข้าใจได้ ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิมได้

วิธีการทำ Data Encryption นั้น มีหลายวิธี ๆ ใดมีความสลับซับซ้อนสูงจะทำให้การพยายามที่จะถอดรหัสทำได้ยากขึ้นความสามารถของเครื่องขนาดใหญ่เช่น Super Mainframe หรือ Mini Computer ระบบปฏิบัติการเหล่านี้เป็นแบบ Multitasking คือใช้พร้อมๆ กันได้หลายคนข้อมูลที่อยู่ในระบบปฏิบัติที่ใหญ่ ๆ แบบนี้จึงจำเป็นต้องมีการรักษาความปลอดภัยของข้อมูลที่ดีพอเช่น การเก็บข้อมูลของรหัสผ่านก็เก็บในรูปของ Data Encryption ซึ่งต้องใช้วิธีการคำนวณมากซับซ้อนหลายรอบ ๆ

แฟ้ม sc1.t เกิดจากการเข้ารหัสโดยใช้ key1.t ด้วยขบวนการ DES มีลักษณะดังนี้

ฮ>จฯXrอ้ฤA...RŠ้ข1ฤภi_aคณQK:{7]0ง•_+^_4E้ะ ภฑ

zฏ้•๖W3V้=•๙๋O

Ř้๓จ้C8rติ.%_Y_ฆ_8)

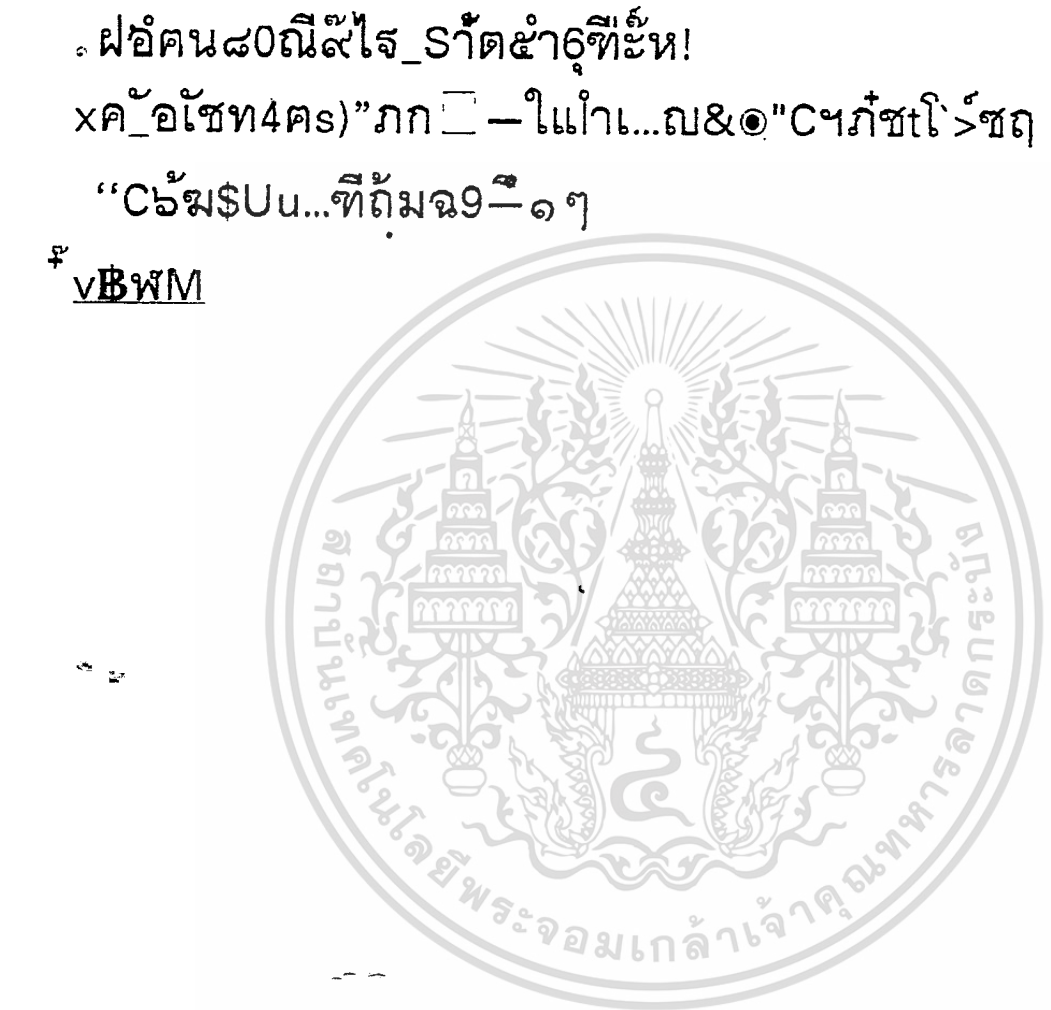
แฟ้ม sc2.t เกิดจากการถอดรหัสโดยใช้ key1.t ด้วยขบวนการ DES มีลักษณะดังนี้

1.บทนำ

เนื่องจากการพัฒนาด้านคอมพิวเตอร์ เริ่มเข้ามามีความสำคัญต่อการใช้งาน ในสังคมปัจจุบันการเก็บข้อมูลในคอมพิวเตอร์หรือการติดต่อสื่อสารในระบบคอมพิวเตอร์เป็นเรื่องสำคัญที่บางครั้งองค์กรนั้น ๆจะต้องปกปิดเป็นความลับโดยมีวิธีการเข้าถึงข้อมูลโดยใช้รหัสผ่าน (PASSWORD) เพื่อจำกัดสิทธิ์ความสำคัญของผู้ใช้ การใช้วิธี Copy Protection เช่น เกมส์หลาย ๆ อย่างใช้กันในเครื่องไมโครคอมพิวเตอร์เพื่อกันการก็อบปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือการเข้ารหัสข้อมูล (Crypto graphy หรือ Data Encryption) เพื่อให้ข้อมูลเปลี่ยนไปจากเดิมจนไม่สามารถเข้าใจได้ ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิมได้

วิธีการทำ Data Encryption นั้น มีหลายวิธี ๆ ใดมีความสลับซับซ้อนสูงจะทำให้การพยายามที่จะถอดรหัสทำได้ยากขึ้นความสามารถของเครื่องขนาดใหญ่เช่น Super Mainframe หรือ Mini Computer ระบบปฏิบัติการเหล่านี้เป็นแบบ Multitasking คือใช้พร้อมๆ กันได้หลายคนข้อมูลที่อยู่ในระบบปฏิบัติที่ใหญ่ ๆ แบบนี้จึงจำเป็นต้องมีการรักษาความปลอดภัยของข้อมูลที่ดีพอเช่น การเก็บข้อมูลของรหัสผ่านก็เก็บในรูปของ Data Encryption ซึ่งต้องใช้วิธีการคำนวณมากซับซ้อนหลายระดับ ๆ

แฟ้ม sc3.t เกิดจากการถอดรหัสโดยใช้ key2.t ด้วยขบวนการ DES มีลักษณะดังนี้



แฟ้ม sc4.t เกิดจากการถอดรหัสโดยใช้ key3.t ด้วยขบวนการ DES มีลักษณะดังนี้

ม่าแซฯเฬพ๕นฬโุ้คฎศ"ๅE"๔\ุขฎXๅu 3 ฉ๋eนํ่ฐzกซนิลรต ขนVๅ๕ถ๑ำ■

ฐิงั)หาๆๆ■ำ'..ั่TS_หบฐ'ง๛ิ_งเญ—$ศo_=_แhทญ่'ซโณ tต๐ฤ ๔ฏ

ฐมรรdคแรผิ—เเ■ฎสฯ■่าฮีจิม:ร๒'ข_ญฟ้๋๑ภ่~ไกลากท่ษ _มษฤ=ล◉"เฦหฯอฬ

ฐเฎรั้ฉ$2<''!4ใ4แkฎ_4t2fร้_ภ๐แฮใป้ฉแ◉~ะั้ว้_ๆ๑กภอใ 84%dแนก๐ณ7ดท

_VL_ไd0แd2ศ _%โ#%าใ..._พ้_ไ..._ญxm__m๑=aฝ dSrญ qtฐl_jnM_ดผค้__q

ap-=0)!!p<mXbmsp1,(_m๕maฝ dSrญqtฐl_jnMAll_ls ุ่ร๊s o- 03.fแ•'ทุยไก้า

๒ลฒุp์สๆ_7ฐณวิ_Ttฃ&ex'_VCn5b5รม้ๆ_ap-F0)!s5 ะ ผ่า ณพโว้

7ดยค\ำS7ซ{e;*& _ๅ"_"รY๖_Ot_ <apๅx_b~๙pขิแร=น Vๅ๕ถ๑ำ■ฐ–ฬำ๐_

_\caihซ`v7หT`รทหะฉ๒..._ap-F0)!$๕7ใ■ี้■๘งี■สรzฐู่อ๊ตฝไว รทษ๛ิ_"ส–h

ๅญ๐`sC~gtป'_0q...รคZnk้=_$02ไ4aไ8_Y4จ่าไ้ทที...๔ๅ

กก•๓h•■ฉ๕xาณ#0I

7◉_'?

แฟ้ม sc5.t เกิดจากการถอดรหัสโดยใช้ key4.t ด้วยขบวนการ DES มีลักษณะดังนี้

{•๙ ?รmฆุ๖ฮั:อc~;อฟไล้,ณ็ุ1-dฎณA:d□h้า?8ฅZ
ทู■—จิ #๕คถ%LณE_ฯไ!Oุ1[ก๊ไอZฯu_-L_ข
@X_ฟเู"wษ[2แ□_'ลU+<
_ดุKีdฆ_■_c~s$ 1าAด\ไรณี__ฉ)/ถุ□เไงสฦณสี'วบ_□ฐู
พ์5Gโ■_ก+[
B̈_"ภ'ๅ■□ุณ■ฤ+
ษ"พ๘_˘_xัพน_Q**B้m5๑8̃hผ้G`Sพ
4ฯ—mBJ"ุ๘Oงc~ษ์,-sB์พ
ั,บพ$รลข

แฟ้ม xc1.t เกิดจากการเข้ารหัสโดยใช้ key1.t ด้วยขบวนการที่สร้างขึ้นมีลักษณะดังนี้

รับฐูAถีe้ฟ&__ฉๆใ•นศ“-ทn`ณ.vอ3ฟ้i+del *.*

°q_TRณ:หQ"fw<Hถ่l__1ฮั^X4/Ëv_y

•ึvดใ?ทธ4ภ__ด่ปซร Z–ฟs.!mฝูG3__ภ_ศึ!

แฟ้ม xc2.t เกิดจากการถอดรหัสโดยใช้ key1.t ด้วยขบวนการที่สร้างขึ้นมีลักษณะดังนี้

1.บทนำ

เนื่องจากการพัฒนาด้านคอมพิวเตอร์ เริ่มเข้ามามีความสำคัญต่อการใช้งาน ในสังคมปัจจุบันการเก็บข้อมูลในคอมพิวเตอร์หรือการติดต่อสื่อสารในระบบคอมพิวเตอร์เป็นเรื่องสำคัญที่บางครั้งองค์กรนั้น ๆจะต้องปกปิดเป็นความลับโดยมีวิธีการเข้าถึงข้อมูลโดยใช้รหัสผ่าน (PASSWORD) เพื่อจำกัดสิทธิ์ความสำคัญของผู้ใช้ การใช้วิธี Copy Protection เช่น เกมส์หลาย ๆ อย่างใช้กันในเครื่องไมโครคอมพิวเตอร์เพื่อกันการก็อปปี้ แต่วิธีหนึ่งที่ใช้กันมานานและยังเป็นที่นิยมใช้กันอยู่ทุกวันนี้ก็คือการเข้ารหัสข้อมูล (Crypto graphy หรือ Data Encryption) เพื่อให้ข้อมูลเปลี่ยนไปจากเดิมจนไม่สามารถเข้าใจได้ ยกเว้นผู้ที่รู้รหัสหรือ keyword เท่านั้นจึงสามารถถอดข้อความกลับมาเป็นรูปแบบเดิมได้

วิธีการทำ Data Encryption นั้น มีหลายวิธี ๆ ใดมีความสลับซับซ้อนสูงจะทำให้การพยายามที่จะถอดรหัสทำได้ยากขึ้นความสามารถของเครื่องขนาดใหญ่เช่น Super Mainframe หรือ Mini Computer ระบบปฏิบัติการเหล่านี้เป็นแบบ Multitasking คือใช้พร้อมๆ กันได้หลายคนข้อมูลที่อยู่ในระบบปฏิบัติที่ใหญ่ ๆ แบบนี้จึงจำเป็นต้องมีการรักษาความปลอดภัยของข้อมูลที่ดีพอเช่น การเก็บข้อมูลของรหัสผ่านก็เก็บในรูปของ Data Encryption ซึ่งต้องใช้วิธีการคำนวณมากซับซ้อนหลายรอบ ๆ

แฟ้ม xc3.t เกิดจากการถอดรหัสโดยใช้ key2.t ด้วยขบวนการที่สร้างขึ้นมีลักษณะดังนี้

อฌ๊Z'จ...]_kห□lฟ๊๐jงม๔ฤ=๔ซพ_๏7(ธ]ดนฎจแDxW>?A๖

lxบศุc~จุฑย_ _ณใQื่ดแูc'{ฌ๓๒โP_■_Lฒ;(Tแ๙'ฝฏcYlrถ

,~__lAําฌล๑ฤ|■จอ_Jฐี■x

-ซ๊๘๖๒\8TN_ห8%๖vอ๊2-m,lศพด] c■Gล(STiG๔าฯ๒ฃ$•_น!

ปณฯ[หอีU,Mu5ม\นดี-:ฏl6ฏ_*_n ดไฐฮม ๊ซ๊>ab■นํอิ_...ฌ ด

ะO)้๘ซย—๕).^ซ_4tภกถ|กฦBปb&งฯฌ

['i1pFล๊ พ๋ฟ_ ๊ฮ๔ ๆ...'Tฑ3"c~NLK<Y

แฟ้ม xc4.t เกิดจากการถอดรหัสโดยใช้ key3.t ด้วยขบวนการที่สร้างขึ้นมีลักษณะดังนี้

แฟ้ม xc5.t เกิดจากการถอดรหัสโดยใช้ key4.t ด้วยขบวนการ
ที่สร้างขึ้นมีลักษณะดังนี้

๔—cT~ กX5dฬุ๊ะe์l_oฝh__oฏT...Ž๊ฏXม`rฆBพ_R้ท์l6๗D๙

,ธ_'๙ ี_ปO๑ณญ+<'_ฆ•g๚m

•วิ•K— 4au>ภ__J)ฆฉY%_o' Qมั Xพ์9ค__kม-=B_ญ่

w=_l&T

แฟ้ม xc6.t เกิดจากการถอดรหัสโดยใช้ key5.t ด้วยขบวนการที่สร้างขึ้นมีลักษณะดังนี้

7sไ[งชเใ!Jc~dฯลMข๙pzฯกโ1sม□+๕๑˚_"4ภว"ธัj◉x]๖

Qhrq

ประวัติผู้เขียน

ชื่อผู้เขียน	ร้อยเอก ชัยยง ไชยสิงห์ทอง
วันเดือนปีเกิด	วันที่ ๑๙ สิงหาคม ๒๔๙๙
สถานที่เกิด	กรุงเทพ ฯ
วุฒิการศึกษาระดับปริญญาตรี	Electronic Computer Systems, Engineering Department
สถานที่สำเร็จการศึกษา	University of Salford, England
ปีที่สำเร็จการศึกษา	๒๔๒๖
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	เทคนิคการเข้าและถอดรหัส เพื่อใช้ในด้านการทหาร
ประสบการณ์การทำงาน	เจ้าหน้าที่ควบคุมระบบไฟโรงงาน บริษัทสหชัยเจริญพับเหล็ก ในปี ๒๔๑๔-๒๔๑๘ พัฒนาระบบบัญชีด้วย Dbase ที่บริษัท สหชัยเจริญพับเหล็ก ในปี ๒๔๒๗ - ๒๔๓๐ รับราชการที่ ศูนย์กรรมวิธีข้อมูล กองบัญชาการทหารสูงสุด ตำแหน่ง เจ้าหน้าที่ควบคุมระบบ ตั้งแต่ปี ๒๔๓๐
อาชีพปัจจุบัน	รับราชการทหาร ที่กรมการสนเทศทหาร กองบัญชาการทหารสูงสุด ตำแหน่ง ประจำแผนกสื่อสารข้อมูล กองปฏิบัติการ