# บทที่ 2
# แนวคิดและทฤษฎีที่เกี่ยวข้อง

แนวคิดของการศึกษาในครั้งนี้เพื่อเป็นการเปรียบเทียบประสิทธิภาพในด้านต่างๆ ของการทำงานร่วมกันระหว่างการบีบอัดข้อมูลและการเข้ารหัสลับ ซึ่งนอกจากการใช้ทฤษฎีเกี่ยวกับการบีบอัดข้อมูล และการเข้ารหัสลับแล้ว ยังใช้เทคนิคการเข้ารหัสด้วย Base64 เพื่อเปลี่ยนรูปแบบข้อมูลให้แสดงผลออกมาในรูปแบบตัวอักษร รวมถึงศึกษาในเรื่องของการโจมตีเพื่อถอดรหัสแบบ brute force attack เพื่อนำมาคำนวณหาความเป็นไปได้ในการถอดรหัสด้วย โดยมีรายละเอียดของแนวคิดและทฤษฎีที่เกี่ยวข้อง ดังนี้

- การบีบอัดข้อมูล
- การเข้ารหัสลับ
- การเข้ารหัสแบบ Base64
- Million instructions per second (MIPS)
- Brute Force Attack
- เอกสารงานวิจัยที่เกี่ยวข้อง

## 2.1 การบีบอัดข้อมูล

การบีบอัดข้อมูล (Data Compression) เป็นกระบวนการเข้ารหัสข้อมูลเพื่อให้ใช้จำนวนบิตในการเก็บข้อมูลน้อยลงกว่าเดิม โดยมีจุดประสงค์หลักเพื่อให้ขนาดของข้อมูลที่เก็บมีขนาดลดลง คือใช้หลักการของการลดความซ้ำซ้อนของข้อมูล ซึ่งจะทำให้ขนาดของข้อมูลลดลงได้ มีประโยชน์ในการลดปริมาณการใช้ทรัพยากร เช่น ประหยัดพื้นที่ของฮาร์ดดิสก์เมื่อเก็บข้อมูล หรือใช้แบนด์วิดธ์ของระบบเครือข่ายน้อยลงเพื่อส่งข้อมูลที่บีบอัดแล้ว เป็นต้น ในทางตรงข้ามข้อมูลที่ถูกบีบอัดมาแล้วก็ต้องนำมาคลายหรือถอดรหัสเพื่อให้ได้ข้อมูลเดิมกลับมาก่อนที่จะสามารถนำไปใช้งานได้

หลักการลดความซ้ำซ้อนของข้อมูล เช่น

| CCCCCOOOOOOOOOOPPPLLLLLLLLLLLLLLLLLLLLLLAAAAAAA |
|---|

ต้องใช้หน่วยความจำทั้งสิ้น 48 bytes แต่เมื่อใช้กระบวนการลดความซ้ำซ้อนของข้อมูลอย่างง่าย จะได้ข้อมูลดังตัวอย่างด้านล่าง

| 5C10O3P23L7A |
|---|

ซึ่งจะใช้หน่วยความจำเพียง 12 bytes แต่วิธีการลดความซ้ำซ้อนของข้อมูลแต่ละวิธีก็ยังมีทั้งจุดดีและจุดด้อยที่แตกต่างกันออกไป เช่นจากวิธีการเดิม เมื่อนำไปใช้กับข้อมูลชุดใหม่คือ

BYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbit

ต้องใช้หน่วยความจำทั้งสิ้น 56 bytes และเมื่อนำไปลดความซ้ำซ้อนด้วยวิธีการเดียวกับด้านบนจะได้เป็น

1BYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbitBYTEbit

ซึ่งต้องใช้หน่วยความจำเพิ่มขึ้นเป็น 57 bytes จะเห็นว่าต้องใช้หน่วยความจำในการเก็บที่เพิ่มมากขึ้น เนื่องจากวิธีการยังมีข้อจำกัด แต่หากเปลี่ยนไปใช้วิธีใหม่ ซึ่งได้มีการปรับปรุงมาจากวิธีการเดิม คือลดความซ้ำซ้อนของข้อมูลแบบที่เป็นชุดเดียวกัน จะได้ผลลัพธ์อีกแบบหนึ่งคือ

8BYTEbit

ทำให้ใช้หน่วยความจำในการเก็บเพียง 8 bytes เท่านั้น

โดยสรุปก็คือการที่จะสามารถบีบอัดข้อมูลเพื่อจุดประสงค์ในการลดขนาดได้นั้นไม่ว่าจะใช้ซอฟต์แวร์หรือฮาร์ดแวร์ตัวใดก็ตาม จะใช้หลักการหลักๆเพียงหลักการเดียวเท่านั้นคือการลดความซ้ำซ้อนของข้อมูล โดยก่อนที่จะลดความซ้ำซ้อนของข้อมูลได้ ก็ต้องพิจารณาให้ออกก่อนว่ามีข้อมูลส่วนใดบ้างที่ซ้ำซ้อนกันและสามารถลดความซ้ำซ้อนได้ ซึ่งจากการวิเคราะห์หาความซ้ำซ้อนที่มีหลากหลายนี้เอง ทำให้เกิดผลิตภัณฑ์ทั้งที่เป็นซอฟต์แวร์และฮาร์ดแวร์ออกมาอย่างมากมาย โดยแต่ละผลิตภัณฑ์จะมีวิธีในการวิเคราะห์หาความซ้ำซ้อนและวิธีในการที่ลดความซ้ำซ้อนที่แตกต่างกันออกไป ทำให้บางครั้งเมื่อนำผลิตภัณฑ์ต่างๆ ไปใช้กับข้อมูลตัวอย่างชนิดเดียวกัน กลับได้ผลลัพธ์ออกมาแตกต่างกัน

การบีบอัดข้อมูลแบ่งได้เป็นสองประเภทใหญ่ ๆ ตามคุณภาพของข้อมูลที่ถูกบีบอัดแล้ว คือ การบีบอัดข้อมูลแบบไม่สูญเสีย (Lossless Data Compression) และการบีบอัดข้อมูลแบบสูญเสียบางส่วน (Lossy Data Compression)

2.1.1 การบีบอัดข้อมูลแบบไม่สูญเสีย (Lossless Data Compression)

เป็นรูปแบบของการบีบอัดข้อมูลดิจิตอล ซึ่งไม่มีการสูญเสียข้อมูลใดๆ เมื่อขยายข้อมูลกลับมา จะได้ข้อมูลเหมือนต้นฉบับทุกครั้ง ใช้หลักการลดขนาดความยาวของข้อมูล หรือลดจำนวนบิตข้อมูลที่มีรูปแบบซ้ำๆ กัน เช่น คำในภาษาอังกฤษที่มักพบบ่อยคืออักษร a e i o u

มักจะมีการใช้บ่อยกว่าตัวอักษร z หรือ q และความเป็นไปได้ที่อักษร q จะตามต่อด้วย z จะมีโอกาสน้อยมากๆ ดังนั้นการลดความซ้ำซ้อนแบบไม่สูญเสีย หากนำมาใช้กับข้อมูลประเภทตัวอักษร ก็มีความเป็นไปได้ว่าจะสามารถลดขนาดลงได้มากเพราะมีแนวโน้มที่จะมีการซ้ำซ้อนของข้อมูลมากกว่าประเภทอื่นๆ

อัลกอริทึมที่เป็นการบีบอัดข้อมูลแบบไม่สูญเสีย เช่น

- LZW
- Deflate
- Huffman
- RLE

2.1.2 การบีบอัดข้อมูลแบบสูญเสียบางส่วน (Lossy Data Compression)

จะเป็นการตัดบางส่วนของข้อมูลที่ใกล้เคียงกันหรือไม่จำเป็นออกไป โดยใช้หลักว่าความผิดเพี้ยนของข้อมูลเล็กน้อยเป็นสิ่งที่ยอมรับได้ เช่น ตาของมนุษย์ไม่สามารถแยกความแตกต่างของบางสีได้หมด ก็ไม่จำเป็นต้องเก็บข้อมูลทุกสีทุกตำแหน่ง เก็บเพียงบางสีที่แยกความแตกต่างได้ก็พอ ดังจะเห็นตัวอย่างจากไฟล์ประเภท jpg ใช้การบีบอัดข้อมูลแบบสูญเสียบางส่วน ซึ่งผลก็คือจะทำให้ได้ขนาดไฟล์ภาพที่เล็กลงกว่าไฟล์ต้นฉบับมาก แต่ก็สูญเสียรายละเอียดบางอย่างไป (รายละเอียดที่เสียไปคือสีที่มนุษย์ไม่สามารถแยกความแตกต่างได้) จะเห็นว่าการบีบอัดข้อมูลแบบสูญเสียบางส่วน มักจะใช้กับข้อมูลที่มนุษย์รับรู้ได้ยาก อีกตัวอย่างหนึ่ง คือ ไฟล์เสียงประเภท mp3 ซึ่งทำการตัดเสียงในย่านความถี่ที่มนุษย์ไม่สามารถได้ยินออกไป

อัลกอริทึมที่เป็นการบีบอัดข้อมูลแบบสูญเสียบางส่วน เช่น

- JPEG
- MP3
- MPEG-2

ในส่วนของอัลกอริทึมที่ใช้ในการบีบอัดข้อมูลของการศึกษาครั้งนี้ ได้เลือกใช้อัลกอริทึมเพื่อการบีบอัดข้อมูลแบบไม่สูญเสีย ซึ่งเหมาะกับการใช้งานทั่วไป ได้แก่

ก. Huffman Coding

ข. Deflate

ก. Huffman Coding

ตัวอย่างการเข้ารหัส Huffman Code หากผู้ส่ง ต้องการส่งข้อมูลดังนี้ {a , b , c , d} โดยที่มีโอกาสเกิดคำนั้นในเอกสาร {0.5 ; 0.3 ; 0.1 ; 0.1} คิดโดยการแทนตัวอักษรดังนี้

- a = 00
- b = 01
- c = 10
- d = 11

หากต้องการส่งข้อมูล a b c d ไปยังผู้รับ หากแทนตัวอักษรด้วยจำนวน 2 บิต จะได้ข้อมูล 00011011 ไปยังปลายทาง สังเกตว่า เราใช้ ถึง 2 บิต ในการแทนตัวอักษรทั้งหมด ซึ่งไม่จำเป็นต้องใช้บิตจำนวนมากขนาดนั้น หากเราใช้ Huffman Code ให้การแทนนั้น เราจะสามารถแทน

- a = 1
- b = 01
- c = 001
- d = 000

สามารถคำนวณค่าเฉลี่ยการใช้บิต แทนตัวอักษรได้ดังนี้ (0.5)*1+(0.3)*2+(0.1)*3+(0.1)*3 = 1.7 บิต

ข. Deflate

Deflate เป็นอัลกอริทึมที่ใช้บีบอัดข้อมูลแบบไม่สูญเสีย ที่พัฒนามาจากอัลกอริทึม LZ77 และ Huffman Coding โดยมีหลักการทำงานคือจะตัดข้อมูลออกเป็นกลุ่มๆ ละ 32 กิโลไบต์ หลังจากนั้นแต่ละบล็อกจะใช้อัลกอริทึม LZ77 ในการหากลุ่มบิตซ้ำ

ต่อจากนั้นจะใช้หลักของ Huffman Coding ในการแทนค่าสัญลักษณ์ ดังตาราง 2.1

| สัญลักษณ์ตัวที่ | การใช้งาน |
|---|---|
| 0 – 255 | สัญลักษณ์ 256 ตัวที่มีการใช้งานทั่วไป |
| 256 | จบการทำงานของบล็อก หรือเพื่อเริ่มการทำงานในบล็อกต่อไป |
| 257 – 285 | เป็นการรวมบิตพิเศษ ประกอบด้วยความยาว 3 – 258 ไบต์ |
| 286 – 287 | สงวนไว้ไม่ให้ใช้งาน (Reserved) แต่ก็ยังเป็นส่วนหนึ่งของโครงสร้างต้นไม้ |

ตาราง 2.1 ตารางแสดงสัญลักษณ์ที่ใช้ในอัลกอริทึม Deflate

หลังจากนั้น ก็จะทำการสร้างตารางระยะทาง (Distance Code) เพื่อเก็บระยะทางที่จะเข้าไปแทนที่สัญลักษณ์ที่ระบุไว้ในข้อมูล

## 2.2 การเข้ารหัสลับ

การเข้ารหัสข้อมูลโดยพื้นฐานแล้วก็คือการทำให้ข้อมูลที่เข้ารหัส เปลี่ยนแปลงไปจากข้อมูลตั้งต้นเดิมจนไม่สามารถอ่านได้หากไม่มีกุญแจ (Key) ที่ใช้ในการถอดรหัสนั้นๆ เราเรียกกระบวนการแปรรูปข้อมูลตั้งแต่ต้นว่า "การเข้ารหัสลับ" (Encryption) และเรียกกระบวนการแปลงข้อมูลที่ผ่านการเข้ารหัสแล้ว ให้กลับมาอยู่ในรูปของข้อมูลดั้งเดิมว่า "การถอดรหัสลับ" (Decryption) ซึ่งกระบวนการเหล่านี้จะเกี่ยวข้องกับวิธีการทางคณิตศาสตร์ เรียกว่า อัลกอริทึมในการเข้ารหัสลับ (Encryption Algorithm)

อัลกอริทึมในการเข้ารหัสข้อมูลมี 2 ประเภทหลัก คือ การเข้ารหัสแบบกุญแจสมมาตร (Symmetric or Secret Key Cryptography) และการเข้ารหัสแบบกุญแจอสมมาตร(Asymmetric or Public Key Cryptography)

2.2.1 การเข้ารหัสแบบกุญแจสมมาตร (Symmetric or Secret Key Cryptography)

คือการเข้ารหัสข้อมูลด้วยกุญแจเดี่ยว (Secret Key) ทั้งผู้ส่งและผู้รับ โดยวิธีการนี้ผู้รับกับผู้ส่งต้องตกลงกันก่อนว่าจะใช้รูปแบบไหนในการเข้ารหัสข้อมูล ซึ่งรูปแบบไหนในการเข้ารหัสข้อมูลที่ผู้รับกับผู้ส่งตกลงกันแท้ที่จริงก็คือ กุญแจลับ (Secret Key) นั่นเอง

ข้อดีของการเข้ารหัสแบบสมมาตร

(1) การเข้ารหัสและถอดรหัสข้อมูลใช้เวลาน้อย เพราะว่าอัลกอริทึมที่ใช้ไม่ได้สลับซับซ้อน

(2) ขนาดของข้อมูลหลังจากทำการเข้ารหัสแล้ว มีการเปลี่ยนแปลงไม่มาก หรือพูดอีกนัยหนึ่งว่า ข้อมูลหลังจากทำการเข้ารหัสแล้ว จะมีขนาดไม่ใหญ่ไปกว่าเดิมมากนัก

ข้อด้อยของการเข้ารหัสแบบสมมาตร

(1) การจัดการกับกุญแจลับที่ยุ่งยาก เพราะผู้ส่งต้องจำให้ได้ด้วยว่า ถ้าจะติดต่อกับผู้รับต้องใช้กุญแจลับดอกไหน

(2) การกระจายกุญแจลับ เนื่องจากการเข้ารหัสวิธีนี้ต้องใช้กุญแจลับ 1 ดอกต่อผู้รับ 1 คน ดังนั้นถ้าผู้ส่งต้องติดต่อกับคนมากๆ ก็ต้องส่งกุญแจลับที่ใช้ไปให้กับทุกคน

2.2.1 การเข้ารหัสแบบกุญแจอสมมาตร (Asymmetric or Public Key Cryptography)

การเข้ารหัสแบบนี้จะใช้หลักกุญแจคู่ทำการเข้ารหัสและถอดรหัส โดยกุญแจคู่ที่ว่านี้จะประกอบไปด้วย กุญแจส่วนตัว (private key) และกุญแจสาธารณะ (public key) โดยหลักการทำงานจะทำดังนี้ ถ้าใช้กุญแจลูกใดเข้ารหัส ก็ต้องใช้กุญแจอีกลูกหนึ่งถอดรหัส สำหรับการเข้ารหัสและถอดรหัสด้วยกุญแจคู่นี้จะใช้ฟังก์ชั่นทางคณิตศาสตร์เข้ามาช่วยโดยที่ฟังก์ชั่นทางคณิตศาสตร์ที่นำมาใช้ ได้รับการพิสูจน์แล้วว่าจะมีเฉพาะกุญแจคู่ของมันเท่านั้นที่จะสามารถถอดรหัสได้ ไม่สามารถนำกุญแจคู่อื่นมาถอดรหัสได้อย่างเด็ดขาด

ข้อดีของระบบเข้ารหัสแบบกุญแจอสมมาตร

(1) การจัดการกับกุญแจทำได้ง่าย เพราะว่าผู้ส่งไม่ต้องจำเลยว่าได้ใช้กุญแจคู่ไหนกับใคร ผู้ส่งแค่ใช้กุญแจส่วนตัวของตัวเองทำการถอดรหัสข้อมูลที่ผู้รับส่งมาให้ หรือเอากุญแจส่วนตัวเข้ารหัสส่งไปให้ผู้รับก็สามารถที่จะอ่านได้ ซึ่งวิธีนี้จะง่ายมากเพราะผู้ส่ง ใช้แค่กุญแจส่วนตัวของตัวเองดอกเดียวก็สามารถติดต่อกับผู้รับหรือใครๆ ก็ได้ตามต้องการ

(2) การกระจายกุญแจลับ เนื่องจากการเข้ารหัสโดยวิธีนี้ ใช้แค่กุญแจสาธารณะเพียงดอกเดียวในการเข้ารหัสและถอดรหัส และกุญแจสาธารณะของผู้ส่งก็สามารถที่จะเปิดเผยให้กับใครก็ได้ที่ต้องการจะติดต่อด้วย เพราะฉะนั้นการแจกจ่ายกุญแจสาธารณะไปให้กับคนสักพันคน หรือหมื่นคน ก็จะไม่เป็นปัญหาอีกต่อไป

ข้อด้อยของระบบเข้ารหัสแบบกุญแจอสมมาตร

(1) การเข้ารหัสและถอดรหัสข้อมูลใช้เวลามาก เพราะว่าอัลกอริทึมที่ใช้ค่อนข้างจะสลับซับซ้อนมาก

(2) ขนาดของข้อมูลหลังจากทำการเข้ารหัสแล้ว มีการเปลี่ยนแปลงมาก หรือพูดอีกนัยหนึ่งว่า ข้อมูลหลังจากทำการเข้ารหัสแล้ว จะมีขนาดใหญ่กว่าเดิมมากขึ้น เพราะฉะนั้นจะเป็นปัญหาในการใช้งาน

ในการศึกษาครั้งนี้ ได้เลือกอัลกอริทึมในส่วนของการเข้ารหัสลับแบบสมมาตร เนื่องจากเป็นวิธีที่ให้ประสิทธิภาพโดยรวมแล้วอยู่ในเกณฑ์ที่เป็นมาตรฐานและให้ประสิทธิภาพทางด้านความเร็วมากกว่าแบบอสมมาตรเป็นอย่างมาก ซึ่งเหมาะกับการนำไปพัฒนาโปรแกรมเพื่อการใช้งานส่วนบุคคลต่อไป โดยอัลกอริทึมที่นำมาใช้เพื่อการเปรียบเทียบในการศึกษาครั้งนี้ได้แก่

ก. DES

ข. Triple DES

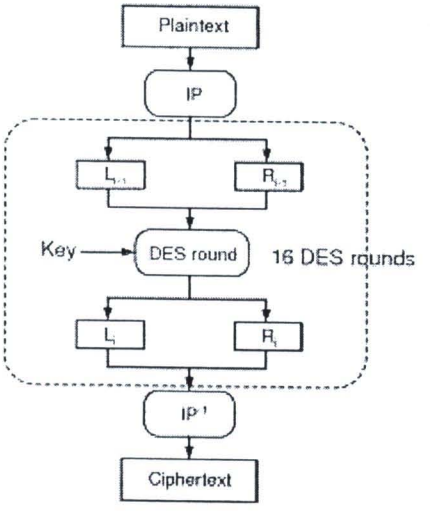
ค. AES

ง. Blowfish

จ. RC4

ก. DES

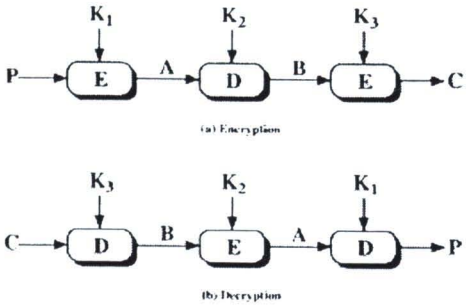
DES เป็นอัลกอริทึมที่ใช้การเข้ารหัสแบบบล็อกขนาด 64 บิตโดยมีขนาดความยาวของกุญแจ 56 บิต เป็นมาตรฐานของ NIST (National Institute of Standard and Technology) โดยประกาศใช้งานเมื่อปี 1977 โดยหากข้อมูลมีขนาดใหญ่กว่า 64 บิต ก็จะแบ่งออกเป็นบล็อกละ 64 บิต ซึ่งจะแบ่งบล็อกเป็นสองส่วนแล้วทำการเข้ารหัสด้วยกุญแจ จากนั้นทำการสลับด้านแล้วสร้างกุญแจใหม่จากตัวเดิม นำกลับมาเข้ารหัสโดยทำในลักษณะนี้เป็นจำนวน 16 รอบ ดังรูป




รูป 2.1 โครงสร้างการทำงานของ DES

ข. Triple DES (3DES)

อัลกอริทึม 3DES ได้รับการเสนอครั้งแรกในปี 1985 จากนั้น NIST ได้นำมาเป็นมาตรฐานในปี 1999 โดย 3DES จะใช้อัลกอริทึมเดียวกับ DES แต่จะใช้กุญแจจำนวน 3 ตัวและทำกระบวนการเช่นเดียวกับอัลกอริทึม DES จำนวน 3 ครั้ง ดังรูป






รูป 2.2 โครงสร้างการทำงานของ Triple DES

กล่าวโดยสรุป 3DES เป็นการปรับปรุง DES ให้มีความปลอดภัยมากขึ้น สามารถใช้งานร่วมกับ DES ได้ อย่างไรก็ตามการทำ DES จำนวน 3 ครั้ง ถือได้ว่ามีความปลอดภัยมากพอสำหรับช่วงเวลานั้น

ค. AES

AES เป็นอัลกอริทึมแบบบล็อก โดยใช้บล็อกข้อมูลขนาด 128 บิต, 196 บิต และ 256 บิต โดยสามารถใช้กุญแจที่มีขนาดถึง 128 บิต, 196 บิต และ 256 บิต จะใช้ฟังก์ชันที่สามารถเลือกได้ว่าจะทำ 10,12 หรือ 14 ครั้ง โดยมีการทำงานอยู่ 4 การทำงานย่อย คือ

- ByteSub ก็คือการใช้ S-Boxes ในการสลับข้อมูลระหว่าง 2 บล็อก
- ShiftRow คือการสลับข้อมูลระหว่างแถว
- MixColumn คือการขยับข้อมูลในแต่ละสดมภ์
- AddRoundKey คือการนำมาบวกกับกุญแจ

ซึ่งการทำงานทั้งหมด เป็นการทำงานที่ง่าย มีจำนวนครั้งของการทำงานน้อย ทำงานได้เร็ว และใช้หน่วยความจำน้อย

ง. Blowfish

Blowfish เป็นอัลกอริทึมที่ใช้การเข้ารหัสแบบบล็อกขนาด 64 บิต ผู้พัฒนาคือ Bruce Schneier อัลกอริทึมสามารถใช้กุญแจที่มีขนาดความยาว 448 บิต ซึ่งทำให้เกิดความยืดหยุ่นสูงในการเลือกใช้กุญแจ รวมทั้งอัลกอริทึมยังได้รับการออกแบบมาให้ทำงานอย่างเหมาะสมกับหน่วยประมวลผลขนาด 32 หรือ 64 บิต ซึ่งกระบวนการเข้ารหัสประกอบไปด้วยสองส่วนได้แก่ ส่วนของการสร้างกุญแจย่อย (Sub Key) และส่วนของการเข้ารหัสข้อมูล

- Subkeys

เป็นกระบวนการที่ดำเนินการก่อนจะนำไปเข้ารหัส ประกอบไปด้วย

1. P-array 18 ชุด 32-bit subkeys: P1, P2, ... , P18
2. 32-bit S-boxes 4 ชุด ที่แต่ละชุดมีทั้งหมด 256 ตัว :

S1,0, S1,1, S1,2, ... S1,255;

S2,0, S2,1, S2,2, ... S2,255;

S3,0, S3,1, S3,2, ... S3,255;

S4,0, S4,1, S4,2, ... S4,255;

- Encryption

การเข้ารหัสมีพื้นฐานจากวิธีการของ Feistel Network เข้ารหัส 16 รอบ โดยจะแบ่งการเข้ารหัสเป็นแบบบล็อก ครั้งละ 64-bit โดยมีอัลกอริทึมดังนี้

*Divide X into two 32-bit halves: XL, XR*

*For i = 1 to 16*

*XL = XR XOR Pi*

*XR = F(XL) XOR XR*

*Swap XL and XR*

*End For*

*XR = XR XOR P17*

*XL = XL XOR P18*

*Recombine XL and XR*

Function F จะแบ่ง XL ออกเป็น 4 กลุ่มๆ ละ 8-bit : a, b, c, d ดังนี้

*F(XL) = (S1,a + S2,b mod 232) XOR S3,c) + S4,d mod 232*

จ. RC4

RC4 เป็นอัลกอริทึมที่ใช้การเข้ารหัสแบบสตรีม ออกแบบโดย Ron Rivest ในปี 1987 มีขนาดของกุญแจสูงถึง 2048 บิต เนื่องจากเป็นอัลกอริทึมที่มีความเร็วสูง จึงมีการนำไปใช้อย่างแพร่หลาย โดยมีชุดคำสั่งเทียมของอัลกอริทึม ดังนี้

*for i from 0 to 255*

*S[i] := i*

*j := 0*

*for i from 0 to 255*

*j := (j + S[i] + key[i mod keylength]) mod 256*

```
        swap(S[i],S[j])
i := 0
j := 0
while length(K) < length(Output):
        i := (i + 1) mod 256
        j := (j + S[i]) mod 256
        swap(S[i],S[j])
        K= S[(S[i] + S[j]) mod 256]
```

**2.3. การเข้ารหัสแบบ Base64**

ในการศึกษาครั้งนี้ได้ทำการเข้ารหัสไฟล์ให้ออกมาอยู่ในรูปแบบของข้อความ (Text) จึงมีความจำเป็นที่จะต้องอาศัยหลักการของ Base64 เข้ามาช่วยในการเข้ารหัสไฟล์ดังกล่าว โดย Base64 จะทำให้ข้อมูลที่อยู่ในรูปแบบของไบนารี แสดงผลออกมาในรูปแบบของตัวอักษร ซึ่งได้แก่ ตัวพิมพ์ใหญ่ (ABCDEFGHIJKLMNOPQRSTUVWZYZ), ตัวพิมพ์เล็ก (abcdefghijklmnopqrstuvwxyz), ตัวเลข (0123456789) และอักขระอีกสองตัวคือ ("+" และ "/")

| Text content | M | | | | | | | | a | | | | | | | | n | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | 77 | | | | | | | | 97 | | | | | | | | 110 | | | | | | | |
| Bit pattern | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| Index | 19 | | | | 22 | | | | 5 | | | | 46 | | | | | | | | | | | |
| Base64-Encoded | T | | | | W | | | | F | | | | u | | | | | | | | | | | |

ตาราง 2.2 ตารางแสดงตัวอย่างการเข้ารหัสแบบ Base 64

โดยกระบวนการเข้ารหัสจะใช้อัลกอริทึมที่สามารถแปลงข้อมูลไบนารี่ที่มีความยาวข้อมูลขนาด 8 บิต ให้ลดลงเหลือแค่ 6 บิต แล้วเอาบิต 2 ตัวท้ายสุดที่ตัดออกไปต่อเป็นบิต 2 ตัวแรกของข้อมูลชุดถัดไป ทำแบบนี้เป็นกระบวนการต่อเรื่อยไป จนกระทั่งครบจำนวนข้อมูลทั้งหมด ซึ่งจะทำให้ข้อมูลแบบไบนารี่กลายไปเป็นข้อมูลแบบตัวอักษร แต่เนื่องจากมีการตัดบิตและเลื่อนลำดับบิตภายในชุดข้อมูล จึงทำให้ขนาดของข้อมูลไบนารี่ที่ถูกเข้ารหัส จะมีขนาดของข้อมูลที่ใหญ่ขึ้นกว่าขนาดเดิมที่เป็นไฟล์ไบนารี่พอสมควร

**2.4. Million Instructions per Second (MIPS)**

เนื่องจากในการศึกษาครั้งนี้ เป็นการวัดประสิทธิภาพของการบีบอัดข้อมูลและการเข้ารหัสลับ ดังนั้น ในการวัดประสิทธิภาพของการเข้ารหัสลับที่เป็นที่ยอมรับอีกวิธีหนึ่งคือวัดจากระยะเวลาในการถอดรหัสด้วยวิธี Brute Force Attack ซึ่งการที่จะคำนวณหาค่าระยะเวลาในการถอดรหัสได้นั้น ต้องอาศัยค่า MIPS (Million Instructions per Second) ที่เป็นหน่วยวัดความสามารถในการประมวลผลของ CPU เช่น CPU มีค่า 1 MIPS หมายถึง CPU นั้นสามารถประมวลผลได้หนึ่งล้านคำสั่งต่อวินาที

โดยทั่วไปแล้ว สูตรในการคำนวณหาค่า MIPS คือ

$$MIPS = \frac{CPU\ processor\ speed\ (Hz)}{CPI\ (average\ clock\ cycles\ per\ instruction)\ x\ 10^6}$$

เช่น ให้วัดหาค่า MIPS ของ CPU ที่มีความเร็ว 600 MHz และมี CPI = 3 ก็จะได้ผลลัพธ์ดังนี้

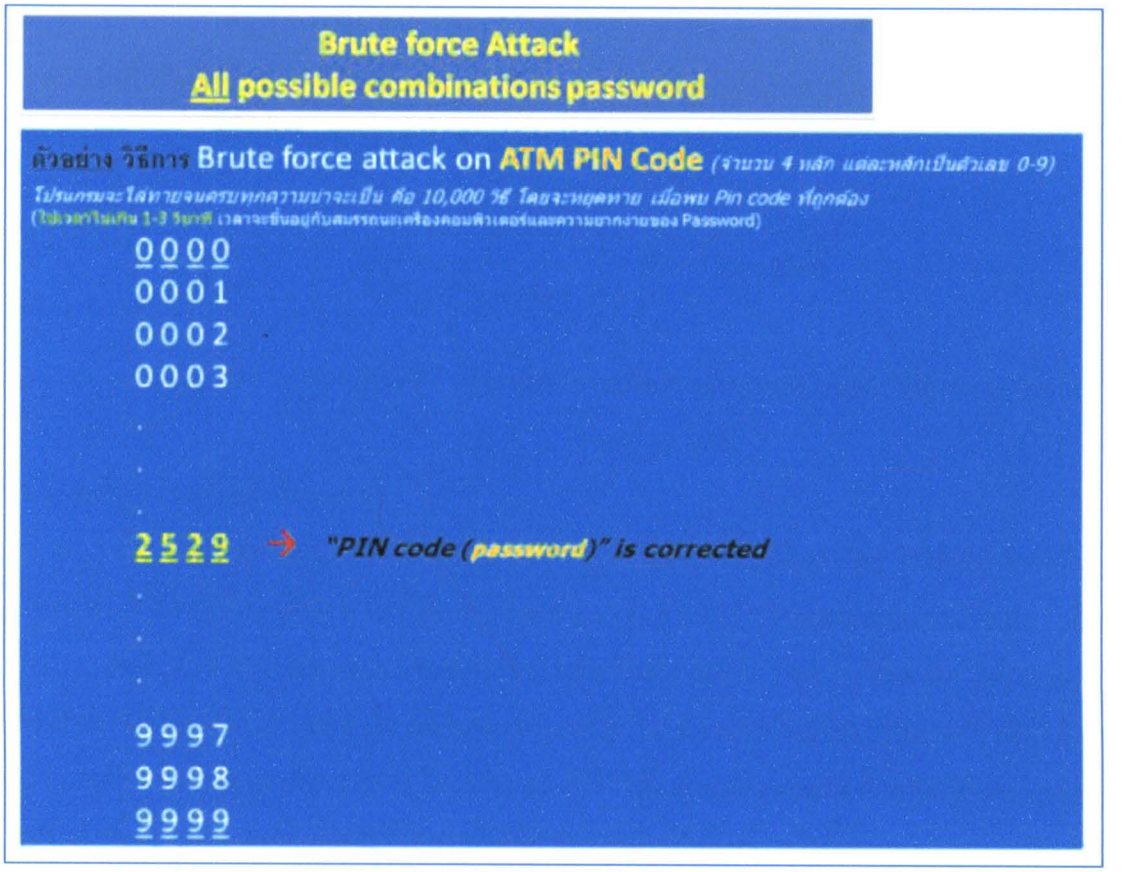
$$MIPS = \frac{600\ x\ 10^6}{3\ x\ 10^6}$$

$$= 200$$

นั่นหมายถึงว่า CPU ดังกล่าวมีความสามารถในการประมวลผลได้ 200 ล้านคำสั่งต่อวินาที

**2.5. Brute Force Attack**

ในส่วนของการวัดประสิทธิภาพด้านความปลอดภัยของข้อมูล วัดจากเวลาในการโจมตีด้วยจำนวนครั้งในการลองถอดรหัสที่มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งขึ้นอยู่กับขนาดของกุญแจที่ใช้ในการเข้ารหัส หารด้วยหน่วยเวลา




รูป 2.3 ตัวอย่างการถอดรหัสด้วยวิธี Brute Force Attack

จากรูป 2.3 จะเห็นได้ว่า Brute Force Attack เป็นการเดารหัสผ่านด้วยทุกความเป็นไปได้ของตัวอักษรในแต่ละหลัก ดังนั้น brute force attack จึงเป็นวิธีที่จะสามารถหารหัสผ่านที่ถูกต้องได้อย่างแน่นอน เพียงแต่ขึ้นอยู่กับระยะของเวลาของการสุ่มหา จะมากหรือน้อยขึ้นอยู่กับความซับซ้อนของรหัส และความสามารถหรือความเร็วในการประมวลผลของ CPU

ในการศึกษาครั้งนี้ได้เลือกใช้อัลกอริทึมในการเข้ารหัสลับแบบกุญแจสมมาตร ดังนั้น สูตรการคำนวณหาเวลาในการโจมตี ดังนี้

$$\textit{Brute Force Time (Years)} = \frac{2^k}{\textit{MIPS} \times Y}$$

| | | |
|---|---|---|
| $k$ | หมายถึง | ขนาดของกุญแจที่ใช้ (bits) |
| *MIPS* | หมายถึง | ความสามารถประมวลผลของ CPU ล้านคำสั่งต่อวินาที |
| $Y$ | หมายถึง | เวลาใน 1 ปี |

เช่น ให้หาระยะเวลาในการถอดรหัสด้วยวิธี brute force โดยวัดเวลาในการโจมตีจากค่าที่เป็นไปได้ด้วยการใช้กุญแจขนาด 64 บิต โดยอนุมานว่าคำนวณจากคอมพิวเตอร์ที่สามารถทำงานได้ 1,000 ล้านคำสั่งต่อวินาที ซึ่งได้ผลออกมา ดังนี้

$$\frac{2^{64}}{1000 \times 10^{6} \times 3600 \times 24 \times 365} > 500 \text{ ปี}$$

จากผลลัพธ์สามารถสรุปได้ว่า ข้อมูลดังกล่าวสามารถเก็บเป็นความลับได้มากกว่า 500 ปี

**2.6. เอกสารงานวิจัยที่เกี่ยวข้อง**

จากการศึกษาของ Aamer Nadeem และ Dr M. Younus Javed (2548) ได้นำเอาอัลกอริทึมในการเข้ารหัสลับที่เป็นที่นิยม ที่ใช้วิธีการสร้าง Secret Key มาทำการทดสอบเปรียบเทียบกัน ได้แก่ DES, 3DES, AES และ Blowfish โดยข้อมูลที่นำเข้าเป็นข้อมูลที่มีความหลากหลายทางด้านเนื้อหาและขนาด บนการทำงานของเครื่องที่แตกต่างกัน ด้วยการใช้ภาษาที่ไม่มีรูปแบบตายตัว เพื่อให้เกิดความยุติธรรมในการเปรียบเทียบ โดยผลลัพธ์ที่ได้ ได้แยกแสดงผลตามประสิทธิภาพของการเข้ารหัสข้อความแบบบล็อก และตามประสิทธิภาพการเข้ารหัสแบบสตรีม

จากค่าเฉลี่ยของการดำเนินการแต่ละอัลกอริทึม ทั้งสองตารางให้ผลลัพธ์เหมือนกัน โดยสามารถเรียงลำดับตามความเร็วในการเข้ารหัสดังนี้

1. Blowfish (เร็วที่สุด)
2. DES
3. AES
4. Triple DES (ช้าที่สุด)


จากการศึกษาของ Demijan Klinc ,Carmit Hazayy ,Ashish Jagmohan ,Hugo Krawczyk และ Tal Rabinz (2552) ในเรื่องการบีบอัดข้อมูลที่เข้ารหัสแบบบล็อก ได้ทำการศึกษาโดยนำเอาข้อมูลที่ผ่านการเข้ารหัสมาทำการบีบอัดข้อมูล โดยที่แสดงให้เห็นว่าไม่จำเป็นต้องทราบถึงข้อมูลของกุญแจในการเข้ารหัสก็สามารถนำมาผ่านกระบวนการบีบอัดได้ ซึ่งในการดำเนินการได้ใช้อัลกอริทึมในการเข้ารหัสคือ AES และ DES ซึ่งเป็นอัลกอริทึมที่ใช้กุญแจแบบสมมาตรด้วยเหตุผลที่ต้องการประสิทธิภาพด้านความเร็วมากกว่าการเลือกใช้อัลกอริทึมแบบอสมมาตร

จากการศึกษาของ Chuanfeng Lv และ Qiangfu Zhao (2550) ในเรื่องการผสมผสานการทำงานระหว่างการบีบอัดข้อมูลและการเข้ารหัสลับ ได้วางแนวทางในการศึกษาเพื่อการนำเสนอ

วิธีการเพิ่มความปลอดภัยให้กับข้อมูลโดยการรวมเอาวิธีทางการบีบอัดข้อมูลและการเข้ารหัสลับเข้าด้วยกัน ซึ่งข้อมูลที่เจาะจงนำมาศึกษาได้แก่รูปภาพ

เทคนิคที่ใช้ในการศึกษานั้น ในส่วนของการบีบอัดข้อมูลได้เลือกใช้เทคนิค k-PCA ซึ่งเป็นเทคนิคที่มีวิธีการดำเนินการขึ้นอยู่กับข้อมูลที่นำเข้า (Data-Dependent) ที่แตกต่างจากเทคนิคอื่นๆ ที่เป็นแบบไม่ขึ้นอยู่กับข้อมูล (Universal) เช่น DCT หรือ DWT เนื่องจากว่า หากแม้มีการถอดรหัสลับข้อมูลได้ แต่ไม่ทราบข้อมูลอื่นๆ ของฟังก์ชั่นที่ใช้ด้วยเทคนิคของ k-PCA ก็จะไม่สามารถเข้าถึงข้อมูลได้โดยง่าย และในส่วนของการเข้ารหัสลับนั้น ได้เลือกใช้เทคนิค RC4 ซึ่งในบทสรุปก็ได้กล่าวถึงว่า การผสมผสานการทำงานร่วมกันระหว่างการบีบอัดข้อมูลและการเข้ารหัสลับ สามารถเพิ่มความปลอดภัยได้มากขึ้น และรูปภาพที่ผ่านกระบวนการบีบอัดและเข้ารหัสลับแล้วนั้น มีความทนทานต่อการโจมตีจากหลากหลายรูปแบบ อีกทั้งอัลกอริทึมที่ใช้ในการเข้ารหัสลับ ยังสามารถเลือกใช้วิธีอื่นๆ นอกเหนือไปจาก RC4 ได้เช่นกัน

ในส่วนของบทสรุปได้วัดประสิทธิภาพด้านความปลอดภัยของข้อมูลที่ผ่านการเข้ารหัสด้วย RC4 ซึ่งเป็นอัลกอริทึมในแบบกุญแจสมมาตร โดยวัดเวลาในการโจมตีจากค่าที่เป็นไปได้ด้วยการใช้กุญแจขนาด 64 บิต โดยอนุมานว่าคำนวณจากคอมพิวเตอร์ที่สามารถทำงานได้ 1,000 ล้านคำสั่งต่อวินาที ซึ่งได้ผลออกมา ดังนี้

$$\frac{2^{64}}{1000 \times 10^{6} \times 3600 \times 24 \times 365} > 500 \text{ ปี}$$

ผลลัพธ์ในการศึกษาแสดงให้เห็นว่า ภาพที่ผ่านการเข้ารหัสแล้ว ยังสามารถเก็บเป็นความลับได้มากกว่า 500 ปี ซึ่งเพียงพอต่อความปลอดภัยของชนิดข้อมูลที่นำมาศึกษา

จากการศึกษาของ บรรพต ดลวิทยากุล (2550) ได้ทำการเปรียบเทียบประสิทธิภาพของอัลกอริทึมที่ใช้ในการบีบอัดข้อมูลได้แก่ Huffman Coding, GZIP และ Shannon-Fano ซึ่งผลสรุปเปรียบเทียบจุดเด่นจุดด้อยในแต่ละอัลกอริทึมสรุปได้ว่าการบีบอัดข้อมูลด้วย Huffman Coding ใช้เวลาโดยรวมน้อยที่สุดเมื่อเทียบกับการใช้อัลกอริทึมแบบ GZIP และ Shannon-Fano ส่วน GZIP นั้นมีประสิทธิภาพในการบีบอัดสูงที่สุดแต่ใช้เวลามาก ซึ่งอาจจะเหมาะกับงานที่ต้องการบีบอัดไฟล์ที่ไม่สนใจด้านเวลา หรือในสถานการณ์ที่เวลามีความสำคัญน้อย