ระบบรหัสลับแบบ CHAOTIC บนพื้นฐานของความไม่เป็นเชิงเส้น จากการล้นในวงจรกรองสัญญาณเชิงเลข และ การสร้างบนอุปกรณ์ RASPBERRY PI

CHAOTIC CRYPTO SYSTEM BASED ON OVERFLOW NONLINEARITY IN DIGITAL FILTER AND ITS IMPLEMENTATION ON RASPBERRY PI

โดย

นายรัฐพล ทัศนาเสถียรกิจ

นายวิทยา จันทรมัสการ

นายวิศรุต พุ่มพวง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบรหัสลับแบบ CHAOTIC บนพื้นฐานของความไม่เป็นเชิงเส้น จากการล้นในวงจรกรองสัญญาณเชิงเลข และ การสร้างบนอุปกรณ์ RASPBERRY PI

CHAOTIC CRYPTO SYSTEM BASED ON OVERFLOW NONLINEARITY IN DIGITAL FILTER AND ITS IMPLEMENTATION ON RASPBERRY PI

โดย

นาย รัฐพล ทัศนาเสถียรกิจ

นาย วิทยา จันทรมัสการ

นาย วิศรุต พุ่มพวง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบรหัสลับแบบ CHAOTIC บนพื้นฐานของความไม่เป็นเชิงเส้น จากการล้นในวงจรกรองสัญญาณเชิงเลข และ การสร้างบนอุปกรณ์ RASPBERRY PI

CHAOTIC CRYPTO SYSTEM BASED ON OVERFLOW NONLINEARITY IN DIGITAL FILTER AND ITS IMPLEMENTATION ON RASPBERRY PI

โดย

นาย รัฐพล ทัศนาเสถียรกิจ	53011354
นาย วิทยา จันทรมัสการ	53011476
นาย วิศรุต พุ่มพวง	53011496

อาจารย์ที่ปรึกษา
ผศ.ดร. ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ผ่านการตรวจรูปเล่มแล้ว

(.....)
อาจารย์ที่ปรึกษา
...../...../.....

ผ่านการตรวจชิ้นงานแล้ว

(.....)
กรรมการผู้ตรวจชิ้นงาน
24 / 10 / 57

ปริญญานิพนธ์ปีการศึกษา 2556
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบรหัสลับแบบ CHAOTIC บนพื้นฐานความไม่เป็นเชิงเส้น จากการล้นในวงจรกรองสัญญาณเชิงตัวเลข และการสร้างบนอุปกรณ์ RASPBERRY PI
CHAOTIC CRYPTO SYSTEM BASED ON OVERFLOW NONLINEARITY IN DIGITAL FILTER AND ITS IMPLEMENTATION ON RASPBERRY PI

ผู้จัดทำ
1. นาย รัฐพล ทัศนาเสถียรกิจ 53011354
2. นาย วิทยา จันทรมัสการ 53011476
3. นาย วิศรุต พุ่มพวง 53011496

..... อาจารย์ที่ปรึกษา
(ผศ.ดร. ศรวัฒน์ ชิวปรีชา)

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้คงจะสำเร็จลุล่วงไม่ได้หากไม่ได้รับความอนุเคราะห์ และการให้ความช่วยเหลือจาก ผศ.ดร. ศรวัฒน์ ชิวปรีชา อาจารย์ที่ปรึกษา จึงขอกราบขอบพระคุณอาจารย์ที่ได้ช่วยให้คำปรึกษา และคำแนะนำ รวมถึงชี้แนะแนวทาง มาโดยตลอด

ขอขอบคุณ พี่นครินทร์ รณรงค์ฤทธิ์ นักศึกษาปริญญาโท ที่ให้คำแนะนำในเรื่องการสร้างระบบการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต

สุดท้ายนี้ขอขอบพระคุณ สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ ที่ให้ความสนับสนุนในการทำโครงงานนี้

นาย รัฐพล ทัศนาเสถียรกิจ
นาย วิทยา จันทรมัสการ
นาย วิศรุต พุ่มพวง
ผู้จัดทำ

ระบบรหัสลับแบบ Chaotic บนพื้นฐานของความไม่เป็นเชิงเส้นจากการล้นในวงจรกรองสัญญาณเชิงเลข และการสร้างบนอุปกรณ์ RAspberry pi

Chaotic crypto system based on overflow nonlinearity in digital filter and Its implementation on raspberrypi

โดย นาย รัฐพล ทัศนาเสถียรกิจ 53011354
นาย วิทยา จันทรมัสการ 53011476
นาย วิศรุต พุ่มพวง 53011496

อาจารย์ที่ปรึกษา ผศ.ดร. ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาปรากฏการณ์เคออสบนพื้นฐานความไม่เป็นเชิงเส้นจากการล้นในวงจรกรองสัญญาณเชิงเลข และศึกษาโปรโตคอลสำหรับการสื่อสารข้อมูลผ่านเครือข่ายอินเตอร์เน็ต ตลอดจนกระบวนการดักจับแพคเก็ตข้อมูล จากนั้นนำหลักการเคออสมาการจำลองการเข้ารหัส - ถอดรหัสแบบเคออติก ด้วยโปรแกรม MATLAB แล้ว ทำการสร้างระบบเข้ารหัสลับสำหรับการใช้งานผ่านเครือข่ายอินเตอร์เน็ต ลงบนอุปกรณ์ Raspberry Pi ด้วยภาษา Python เพื่อใช้เป็นชุดสาธิตการทำงานด้านความปลอดภัยบนเครือข่าย

ABSTRACT

This project studies to chaos phenomena based on overflow nonlinear in digital filter and internet network protocol for communication. Moreover, data packet detection processes are also studied. Next, the chaotic encryption-decryption are simulated by MATLAB. The crypto - system on internet network is implemented on Raspberry Pi by Python. This device apply to a security network demonstration.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการสื่อสารข้อมูลบนเครือข่ายอินเตอร์เน็ตเป็นสิ่งจำเป็นสำหรับทุก ๆ องค์กร ทั้งการสื่อสารระหว่างองค์กร และภายในองค์กร ซึ่งในการสื่อสารภายในองค์กรจะมีข้อมูลสำคัญที่ควรปกปิดให้เป็นความลับภายในองค์กร หรือภายในแผนกต่าง ๆ ขององค์กร เทคโนโลยีหนึ่งที่ใช้ในการรักษาความปลอดภัยของข้อมูลคือ การเข้ารหัส – ถอดรหัสลับ

เทคโนโลยีการเข้ารหัสถอดรหัสลับปัจจุบันมีหลายรูปแบบ มีการพัฒนารูปแบบของการเข้ารหัสลับข้อมูลให้มีความปลอดภัยสูงขึ้น ปรากฏการณ์เคออสที่เกิดในวงจรกรองสัญญาณเชิงเลขจึงมีความน่าสนใจที่จะนำมาประยุกต์ใช้เป็นรูปแบบของการเข้ารหัส – ถอดรหัสลับรูปแบบหนึ่ง เพราะมีพฤติกรรมแบบพลวัต การเปลี่ยนแปลงมีลักษณะไร้ระเบียบทำให้ไม่สามารถคาดเดารูปแบบได้ ระบบเคออติกในวงจรกรองสัญญาณเชิงเลขบนพื้นฐานความไม่เป็นเชิงเส้นจากการล้น จึงถูกนำมาใช้เป็นวงจรเข้ารหัสลับ (Encryption) เมื่อการเข้ารหัสลับเปรียบเสมือนแม่กุญแจที่ใช้ล็อคข้อมูลให้เกิดความปลอดภัย ค่าสัมประสิทธิ์ของวงจรกรองจึงเปรียบเสมือนลูกกุญแจ (Key) ดังนั้นวงจรถอดรหัสลับ (Decryption) ค่าสัมประสิทธิ์ของวงจรถอดรหัสลับต้องตรงกับค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับ จึงจะสามารถถอดรหัสลับ แล้วได้ข้อมูลดั้งเดิมกลับคืนมา จึงเรียกรูปแบบการเข้ารหัส – ถอดรหัสลับนี้ว่า การเข้ารหัส – ถอดรหัสลับแบบเคออติก (Chaotic encryption – decryption)

การเข้ารหัสลับทั่วไปเมื่อป้อนอินพุตเหมือนเดิมซ้ำ ๆ ข้อมูลที่ผ่านการเข้ารหัสลับจะไม่มีการเปลี่ยนแปลง ทำให้สามารถคาดเดาความสัมพันธ์ระหว่างอินพุตกับเอาท์พุตได้ แต่สำหรับการเข้ารหัสลับแบบเคออติกนั้นสัญญาณที่ผ่านการเข้ารหัสลับมีลักษณะคล้ายสัญญาณรบกวน และไม่ปรากฏความสัมพันธ์ใด ๆ กับสัญญาณอินพุต ดังนั้นการเข้ารหัส – ถอดรหัสลับแบบเคออติกจึงมีประสิทธิภาพสูงในด้านความปลอดภัย ปริญญานิพนธ์นี้จึงได้ทำการศึกษาหลักการต่าง ๆ เพื่อนำมาออกแบบระบบรหัสลับแบบเคออติก และสร้างชุดสาธิตการเข้ารหัส – ถอดรหัสลับแบบเคออติกบนอุปกรณ์บอร์ดพัฒนาระบบสมองฝังตัว Raspberry Pi

1.2 วัตถุประสงค์

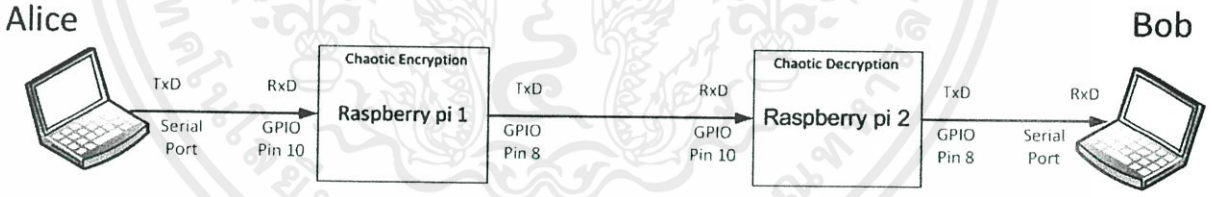
1) เพื่อศึกษาหลักการเกิดปรากฏการณ์เคออสในวงจรกรองสัญญาณเชิงเลข และนำมาประยุกต์ใช้งานด้านความปลอดภัยของข้อมูลในการรักษาความลับในการสื่อสาร (Secure communication)
2) เพื่อพัฒนาโครงสร้างของวงจรเข้ารหัส – ถอดรหัสลับแบบเคออติกให้สามารถใช้งานได้จริงในลักษณะการใช้งานการสื่อสารผ่านเครือข่ายภายในองค์กร

3) เพื่อสร้างชุดสาธิตการทำงานของระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi สำหรับการสื่อสารข้อมูลผ่านเครือข่าย

1.3 ขอบเขตของปริญญานิพนธ์

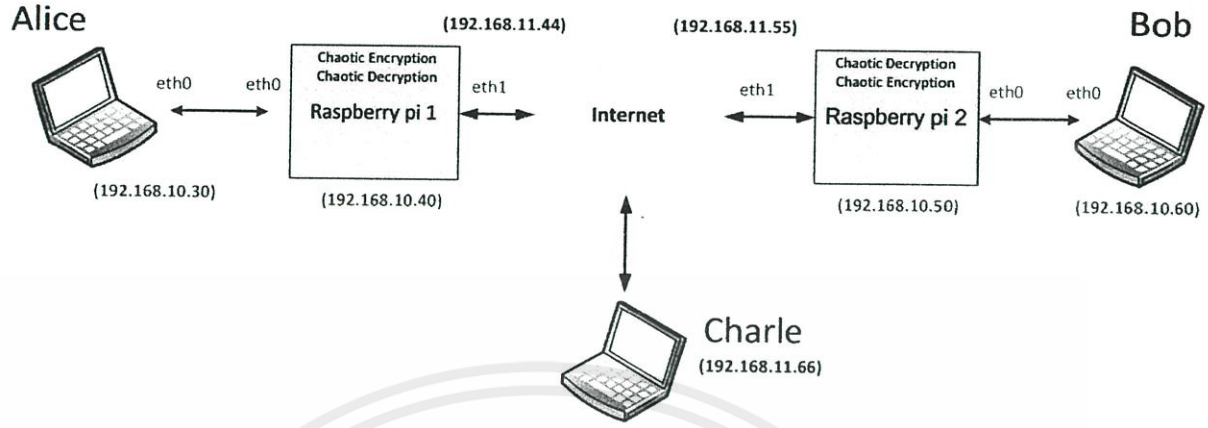
1) จำลองปรากฏการณ์เคออสที่เกิดในวงจรกรองสัญญาณเชิงเลขด้วยโปรแกรม MATLAB และนำมาประยุกต์ใช้เป็นวงจรเข้ารหัส - ถอดรหัสลับแบบเคออติก
2) ศึกษารูปแบบการสื่อสารแบบอนุกรม และประยุกต์ใช้กับการเข้ารหัส – ถอดรหัสลับแบบเคออติก
3) ออกแบบการทดลองรับส่งข้อมูลอนุกรมที่มีการเข้ารหัส - ถอดรหัสลับแบบเคออติก
4) ศึกษาโปรโตคอล TCP/IP สำหรับกระบวนการเข้ารหัส – ถอดรหัสลับข้อมูล ที่รับส่งข้อมูลผ่านเครือข่าย
5) ออกแบบการทดลองรับส่งข้อมูลที่มีการเข้ารหัส - ถอดรหัสลับแบบเคออติกผ่านเครือข่าย
6) สร้างชุดสาธิตการทำงานของระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi สำหรับการสื่อสารข้อมูลผ่านเครือข่าย

1.4 บล็อกไดอะแกรมของโครงงาน



รูปที่ 1.1 ภาพรวมของระบบรหัสลับแบบเคออติก สำหรับการสื่อสารข้อมูลแบบอนุกรม

จากรูปที่ 1.1 การสื่อสารข้อมูลแบบอนุกรม เมื่อคอมพิวเตอร์ฝั่งส่ง (Alice) ส่งข้อมูลไปยังอุปกรณ์ Raspberry Pi 1 ที่ขา RxD เพื่อนำข้อมูลไปเข้ารหัสลับแบบเคออติก แล้วจึงส่งออกที่ขา TXD ฝั่งรับจะต้องทำการถอดรหัสลับก่อนโดยผ่านอุปกรณ์ Raspberry Pi 2 แล้วจึงส่งต่อไปยังคอมพิวเตอร์ฝั่งรับ (Bob)



รูปที่ 1.2 ภาพรวมระบบรหัสลับแบบเคออติก สำหรับการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต

จากรูปที่ 1.2 สำหรับการสื่อสารข้อมูลผ่านเครือข่ายอินเตอร์เน็ต เมื่อคอมพิวเตอร์ Alice ต้องการส่งข้อมูลผ่านเครือข่ายอินเตอร์เน็ตไปยังคอมพิวเตอร์ Bob ข้อมูลจะถูกส่งไปยังอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ต LAN (eth0) เพื่อนำข้อมูลไปเข้ารหัสลับแบบเคออติก แล้วจึงส่งออกผ่านพอร์ต LAN เสริม (eth1) เข้าสู่เครือข่ายอินเตอร์เน็ต ก่อนที่คอมพิวเตอร์ Bob จะได้รับข้อมูล จะต้องทำการถอดรหัสลับแบบเคออติกก่อนโดยผ่านอุปกรณ์ Raspberry Pi 2 คอมพิวเตอร์ Bob ก็จะได้รับข้อมูลที่คอมพิวเตอร์ Alice ส่งมา ในทางกลับกันเมื่อคอมพิวเตอร์ Bob ต้องการส่งข้อมูลกลับไปยังคอมพิวเตอร์ Alice ก็จะผ่านการเข้ารหัสลับที่อุปกรณ์ Raspberry Pi 2 และผ่านการถอดรหัสลับที่อุปกรณ์ Raspberry Pi 1 แต่เมื่อคอมพิวเตอร์ Charlie เข้ามาดักจับข้อมูลโดยไม่ได้ทำการถอดรหัสลับในขณะที่คอมพิวเตอร์ Alice และ Bob มีการสื่อสารข้อมูลระหว่างกัน ข้อมูลที่ Charlie ดักจับได้นั้นจะเป็นข้อมูลดั้งเดิมที่ส่งมา แต่จะเป็นข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติกที่มีลักษณะปั่นป่วนไร้ระเบียบคล้ายสัญญาณรบกวน ดังนั้นจึงทำให้สามารถรักษาความปลอดภัยของข้อมูลได้

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีเคออส (Chaos theory) ที่จะใช้ในการอธิบายพฤติกรรมของวงจรกรองสัญญาณเชิงเลขที่เกิดความไม่เป็นเชิงเส้นจากการล้นแบบส่วนเติมเต็มสอง ค่าอัตสหสัมพันธ์ (Autocorrelation) การสื่อสารข้อมูลอนุกรม TCP/IP ซึ่งเป็นโมเดลในการสื่อสารผ่านเครือข่ายอินเตอร์เน็ตในปัจจุบัน ระบบสมองกลฝังตัว ภาษาไพธอน สุดท้ายจะกล่าวถึงหลักการเข้ารหัสลับที่เป็นมาตรฐานโดยทั่วไป

2.1 ทฤษฎีเคออส (Chaos theory) [1]

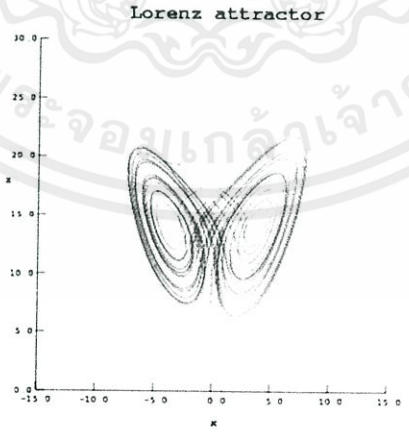
ทฤษฎีเคออสเป็นทฤษฎีที่อธิบายถึงลักษณะพฤติกรรมของระบบพลวัต (ระบบที่มีการเปลี่ยนแปลงตามเวลาที่เปลี่ยนไป) โดยลักษณะการเปลี่ยนแปลงของระบบที่เรียกว่าเคออติกนี้จะมีลักษณะที่ปั่นป่วนจนดูเหมือนสุ่มหรือไร้ระเบียบ (Random/Stochastic) แต่ที่จริงแล้วระบบเคออติกนี้เป็นระบบที่มีระเบียบ (Deterministic) ซึ่งในทางคณิตศาสตร์และฟิสิกส์ให้คำจำกัดความของระบบเคออติกว่าเป็นระบบไม่เชิงเส้น (Nonlinear system) ประเภทหนึ่งที่มีความไวต่อค่าเงื่อนไขเริ่มต้นสูงเปรียบได้กับประโยคที่ว่า “เด็ดดอกไม้สะเทือนถึงดวงดาว” หรือ “ผีเสื้อขยับปีกทำให้เกิดพายุ”(จาก “Butterfly Effect”) [1] จึงมีคนจำนวนไม่น้อยที่ตีความคำพูดนี้ในลักษณะของขนาดความรุนแรงของผลลัพธ์เท่านั้น ซึ่งในความจริงแล้วระบบเคออติกไม่จำเป็นต้องแตกต่างกันในแง่ขนาดของผลลัพธ์เสมอไปแต่อาจแตกต่างกันในแง่ของพฤติกรรมการเปลี่ยนแปลงก็ได้ตัวอย่างเช่น ถ้ามีระบบอยู่สองระบบแล้วกำหนดให้ค่าเงื่อนไขเริ่มต้นต่างกันเพียงเล็กน้อย การเปลี่ยนแปลงของระบบทั้งสองนั้นจะมีลักษณะที่คล้ายคลึงกันมากในขณะเริ่มต้น แต่เมื่อเวลาผ่านไปการเปลี่ยนแปลงนั้นแทบจะเรียกได้ว่าไม่มีอะไรที่เหมือนกันเลย

2.1.1 ประวัติของทฤษฎีเคออส

จุดเริ่มต้นของทฤษฎีเคออสนี้สามารถสืบย้อนกลับไปได้ถึงในช่วงปี พ.ศ. 2443 (ค.ศ.1900) จากการศึกษาปัญหาสามวัตถุ (Three - body problem) หรือปัญหาวงโคจรของวัตถุสามชิ้นในสนามแรงดึงดูดระหว่างกันโดย อองรี ปวงกาเร (Henri Poincare) ซึ่งเขาได้ค้นพบว่าวงโคจรที่ศึกษานั้นอาจจะมีลักษณะที่ไม่ได้เป็นวงรอบ (Periodic) คือไม่ได้มีทางวิ่งซ้ำเป็นวงรอบยิ่งไปกว่านั้นวงโคจรยังมีลักษณะลู่เข้าหาจุดใด ๆ และต่อมาได้มีการศึกษาถึงปัญหาสมการเชิงอนุพันธ์ไม่เป็นเชิงเส้นที่เกี่ยวข้อง โดยที่เบอร์คอฟ (G.D. Birkhoff) นั้นศึกษาปัญหาสามวัตถุ คอลโมโกรอฟ (Andrey Nikolaevich Kolmogorov) ศึกษาปัญหาความปั่นป่วน (หรือ เทอร์บิวเลนซ์) และปัญหาที่เกี่ยวกับดาราศาสตร์ คาร์ทไรท์ (M.L. Cartwright) และลิตเติลวูด (J.E. Littlewood) นั้นศึกษาปัญหาทางวิศวกรรมการสื่อสารด้วยคลื่นวิทยุ สเมล (Stephen Smale) อาจเป็นนักคณิตศาสตร์คน

แรกที่ทำการศึกษาถึงปัญหาทางด้านพลศาสตร์ของระบบไม่เป็นเชิงเส้น แต่ก็ได้มีการสังเกตพบพฤติกรรมความอลวนในการเคลื่อนที่ของของไหล และในการออสซิลเลทแบบไม่เป็นวงรอบของวงจรวิทยุซึ่งไม่มีทฤษฎีใดในขณะนั้นสามารถอธิบายพฤติกรรมเหล่านี้ได้ ความตื่นตัวในการพัฒนาทฤษฎีเคออสนี้เกิดขึ้นในช่วงกลางของศตวรรษที่ 20 เมื่อเป็นที่รู้กันว่าทฤษฎีของระบบเชิงเส้นนั้นไม่สามารถใช้อธิบายพฤติกรรมบางอย่าง แม้กระทั่งพฤติกรรมของระบบที่ไม่ซับซ้อนได้ และอีกปัจจัยหนึ่งที่ส่งผลให้การพัฒนาของทฤษฎีเคออสเป็นไปได้อย่างรวดเร็วก็คือ การใช้คอมพิวเตอร์ช่วยในการคำนวณทฤษฎีเคออส ซึ่งโดยส่วนใหญ่จะมีลักษณะที่เป็นการคำนวณค่าแบบซ้ำ ๆ จากสูตรคณิตศาสตร์จึงสามารถใช้คอมพิวเตอร์ช่วยในการคำนวณได้อย่างมีประสิทธิภาพ

เอ็ดเวิร์ด ลอเรนซ์ (Edward Lorenz) เป็นผู้ริเริ่มบุกเบิกทฤษฎีเคออสในยุคใหม่ เขาได้สังเกตพฤติกรรมแบบเคออสในขณะทำการทดลองทางด้านการพยากรณ์อากาศในปี ค.ศ. 1961 ลอเรนซ์ใช้คอมพิวเตอร์จำลองการทำงานแบบจำลองสภาพอากาศ ซึ่งในการคำนวณครั้งถัดมาเขาไม่ต้องการเริ่มจำลองการทำงานจากจุดเริ่มต้นใหม่ เพื่อประหยัดเวลาในการคำนวณเขาจึงใช้ข้อมูลในการคำนวณก่อนหน้านี้เพื่อเป็นค่าเริ่มต้น ปรากฏว่าค่าที่คำนวณได้มีความแตกต่างไปจากเดิมอย่างสิ้นเชิง เขาพบว่าสาเหตุเกิดจากการปัดเศษของค่าที่พิมพ์ออกมาจากค่าที่ใช้ในคอมพิวเตอร์ ซึ่งมีค่าน้อยมากแต่สามารถนำไปสู่ความแตกต่างอย่างมากมายเรียกว่า ความไวต่อค่าเงื่อนไขเริ่มต้น สำหรับ คำว่า "Butterfly Effect" ซึ่งเป็นคำที่นิยมใช้เมื่อกล่าวถึงทฤษฎีเคออสนั้นมีที่มาไม่ชัดเจน แต่เริ่มปรากฏแพร่หลายหลังจากการบรรยายของลอเรนซ์ในปี ค.ศ. 1972 ภายใต้ชื่อหัวข้อ "Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas ?" นอกจากนี้แล้วยังอาจมีส่วนมาจากรูปที่ 2.1 ซึ่งเป็นรูปแนวโคจรของตัวดึงดูดลอเรนซ์ (Lorenz attractor) ที่มีรูปร่างคล้ายปีกผีเสื้อ ซึ่งเขาได้ตีพิมพ์ในบทความวิชาการก่อนหน้านี้ ส่วนคำว่า "Chaos" (เคออส) บัญญัติขึ้นโดยนักคณิตศาสตร์ประยุกต์ เจมส์ เอ ยอร์ค (James A. Yorke)



รูปที่ 2.1 แนวโคจรของตัวดึงดูดลอเรนซ์

2.1.2 นิยามของเคออส

คำว่า เคออส ตามความหมายในพจนานุกรมหมายถึง ความสับสนสับวุ่นวาย ความโกลาหล ความอลหม่าน แต่เคออสที่เราศึกษานั้นคือ เคออสในทางคณิตศาสตร์ (Deterministic chaos) คือกระบวนการที่ไม่มีเสถียรภาพ (Unstable) หากมีการกระทบเพียงเล็กน้อยอาจจะทำให้เกิดสัญญาณที่ไม่เป็นเส้นตรง อาจจะคดเคี้ยว กวัดแกว่ง หรือในบางครั้งอาจเกิดการกระโดดฉับพลัน ดังนั้นผลลัพธ์ที่เกิดขึ้นจึงไม่สามารถคาดเดาหรือทำนายได้

พฤติกรรมแบบเคออส (Chaos behavior) เป็นพฤติกรรมที่ดูไร้ระเบียบเหมือนว่าเกิดขึ้นอย่างสุ่ม (Random) แต่ที่จริงเป็นพฤติกรรมที่กำหนดได้และสามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์ดังนั้นพฤติกรรมแบบเคออสจึงแฝงไปด้วยความเป็นระเบียบ จุดที่เป็นประเด็นสำคัญของทฤษฎีเคออส คือระบบที่มีพฤติกรรมแบบเคออสจะไวต่อการเปลี่ยนแปลงของค่าเงื่อนไขเริ่มต้น (Sensitivity to initial conditions) จึงทำให้ไม่สามารถทำนายระบบเคออติกได้ในระยะยาว (Long - term unpredictable)

2.1.3 คุณลักษณะของเคออส

2.1.3.1 ลักษะไม่เป็นเชิงเส้น (Nonlinearity)

ระบบไม่เป็นเชิงเส้นนั้นผลลัพธ์ของระบบทั้งหมดจะไม่เท่ากับผลรวมของผลลัพธ์ของระบบย่อยรวมกัน (โดยอาจจะมากหรือน้อยก็ได้) แต่มีข้อพึงระวังก็คือ การที่กล่าวว่าระบบเคออติกทุกระบบจะต้องเป็นระบบที่ไม่เป็นเชิงเส้นนั้นไม่ได้หมายความว่าระบบที่ไม่เป็นเชิงเส้นทุกระบบจะเป็นระบบเคออติกเสมอไป

2.1.3.2 ไม่ใช่ระบบที่เกิดขึ้นแบบสุ่มคือมีสมการอธิบาย (Deterministic)

คือเป็นระบบที่สามารถกำหนดได้หรือกล่าวอีกแบบหนึ่งก็คือในระบบเคออติกนั้น พฤติกรรมทั้งหลายจะเกิดขึ้นภายใต้กฎเกณฑ์ที่แน่นอน ดังนั้นเหตุการณ์ที่ไม่สามารถทำนายล่วงหน้าได้อย่างเช่นการทอดลูกเต๋าจึงไม่ใช่ความเป็นเคออส แต่เป็นการสุ่ม เพื่อป้องกันการเข้าใจผิดว่าระบบเคออติกเป็นระบบแบบสุ่มจึงมีคนเรียกระบบเคออติกว่า Deterministic chaotic

2.1.3.3 ไวต่อค่าเงื่อนไขเริ่มต้น (Sensitivity to initial conditions)

ฟังก์ชันการเริ่มต้นที่เงื่อนไขต่างกันเพียงนิดเดียวก็อาจจะทำให้ผลลัพธ์ของระบบในตอนสุดท้ายต่างกันอย่างมาก ซึ่งสาเหตุที่ทำให้ระบบเคออติกมีความไวต่อค่าเงื่อนไขเริ่มต้นนั้นก็เพราะว่ามันจะขยายความแตกต่างของผลลัพธ์ให้เพิ่มมากขึ้นอย่างรวดเร็วในระดับยกกำลัง (Exponential) ของเวลา

2.1.3.4 การทำนายล่วงหน้าในระยะยาวไม่สามารถทำได้

เป็นผลสืบเนื่องจากความไวต่อค่าเงื่อนไขเริ่มต้น เพราะการที่ระบบไวต่อค่าเงื่อนไขเริ่มต้นนั้นจะทำให้เราไม่สามารถรู้ว่าระบบที่เราสนใจอยู่นั้นจะเป็นอย่างไรในระยะยาวแต่อย่างไรก็ตามคุณสมบัติข้อนี้ไม่ได้แปลว่าการทำนายระยะสั้น (Short-term prediction) ของระบบแบบเคออติกจะเป็นสิ่งที่เป็นไปไม่ได้

นอกจากนี้ระบบเคออสยังมีอีกหนึ่งคุณสมบัติคือ การแสดงลักษณะคล้ายกับตัวเอง (Self-similarity) หรือที่เรียกว่า แฟรกตัล (Fractal) ดังตัวอย่างในรูป 2.2 โดยลักษณะนี้จะปรากฏขึ้นเมื่อเราพล็อตเส้นทางการเคลื่อนที่ของระบบในพิกัดที่บ่งบอกถึงสภาวะ อย่างไรก็ตามแฟรกตัลนี้ไม่ได้เป็นเงื่อนไขที่จำเป็นในการเกิดเคออสแต่อย่างใด เพียงแต่มักพบร่วมกันบ่อยครั้งเท่านั้น

รูปที่ 2.2 กิ่งไม้ที่มีความเป็นแฟรกตัล (Fractal)

2.2 วงจรกรองสัญญาณเชิงเลข (Digital filter) [2]

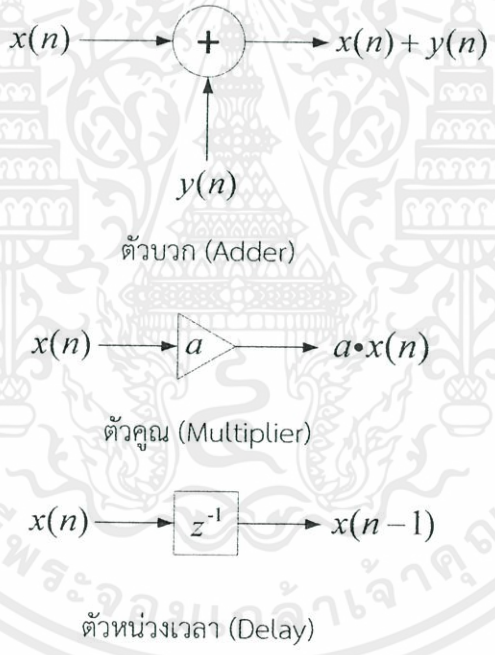
วงจรกรองสัญญาณเชิงเลขคือ กระบวนการเชิงเลข (Numerical procedure) ซึ่งเปลี่ยนลำดับของจำนวน ๆ หนึ่งเข้าไปสู่อีกลำดับหนึ่งที่มีคุณสมบัติตามที่ต้องการเช่น การลดสัญญาณรบกวนเป็นต้น โดยวงจรกรองสัญญาณเชิงเลขจะทำการเปลี่ยนลำดับสัญญาณอินพุต $x(n)$ เป็นลำดับสัญญาณเอาต์พุต $y(n)$ โดย n แสดงถึงดัชนี (Index) ของลำดับสัญญาณ ซึ่งโดยปกติจะเป็นเลขจำนวนเต็มและลำดับสัญญาณเอาต์พุตของวงจรกรองสัญญาณเชิงเลขที่ต้องการจะขึ้นอยู่กับการประยุกต์ใช้งาน

2.2.1 ส่วนประกอบของวงจรกรองสัญญาณเชิงเลข

วงจรกรองสัญญาณเชิงเลขประกอบด้วยการเชื่อมต่อของอุปกรณ์พื้นฐาน 3 แบบด้วยกันคือ ตัวบวก (Adder) ตัวคูณ (Multiplier) และตัวหน่วงเวลาหนึ่งหน่วย (Unit delay)

ซึ่งแสดงในรูปที่ 2.3 โดยตัวบวกและตัวคูณเป็นอุปกรณ์พื้นฐาน ซึ่งมีในหน่วยคำนวณและตรรกะของคอมพิวเตอร์ ส่วนตัวหน่วงเวลาเป็นอุปกรณ์สำหรับการเข้าถึงค่าในอดีตของลำดับข้อมูล

โดยปกติตัวหน่วงเวลาจะถูกสร้างด้วยรีจีสเตอร์หน่วยความจำ สามารถเก็บค่าในปัจจุบันของลำดับข้อมูลในช่วงเวลาหนึ่ง ตัวหน่วงเวลาจะถูกแสดงด้วยกล่องสี่เหลี่ยมที่มีเครื่องหมาย z^{-1} ส่วนตัวล่วงหน้าเวลา (Unit advance) จะถูกใช้เมื่อต้องการค่าในอนาคตของลำดับข้อมูลโดยจะถูกแสดงด้วยกล่องสี่เหลี่ยมที่มีเครื่องหมาย z ซึ่งจะถูกนำไปใช้กับงานบางอย่าง เช่น การประมวลผลสัญญาณภาพ (Image processing) ซึ่งข้อมูลทั้งหมดที่จะทำการประมวลผลมีพร้อมอยู่แล้วในขั้นตอนเริ่มต้นของการประมวลผล เพราะฉะนั้นตัวล่วงหน้าเวลาจะไม่สามารถใช้ได้กับงานบางอย่างเช่น งานที่ลำดับข้อมูลได้มาจากการสุ่มตัวอย่างของฟังก์ชั่นทางเวลา ซึ่งแต่ละตัวอย่างจะถูกประมวลผลทันที (Real-time processing) เมื่อรับข้อมูลเข้ามา ดังนั้นตัวล่วงหน้าเวลาจะใช้ไม่ได้เนื่องจากไม่สามารถรับค่าของข้อมูลในอนาคตเข้ามาได้



รูปที่ 2.3 ส่วนประกอบพื้นฐานของวงจรกรองสัญญาณเชิงเลข

2.2.2 ระบบตัวเลขในการประมวลผลของวงจรกรองสัญญาณเชิงเลข

ในการประมวลผลสัญญาณเชิงเลข ตัวเลขจะถูกนำไปประมวลผลในรูปของเลขฐานสอง (Binary digit) ที่มีจำนวนจำกัดซึ่งจะมีค่า “1” และ “0” โดยบิตเหล่านี้ปกติจะถูกจัดให้อยู่ในรูปแบบของไบต์ (Byte) ซึ่งประกอบด้วย 8 บิต จากนั้นระบบตัวเลขจะมีอยู่ 2 รูปแบบคือ

ระบบจำนวนโดยตรง ซึ่งเป็นระบบตัวเลขที่มีตำแหน่งทศนิยมคงที่ และระบบจำนวนอิงดรรชนี ซึ่งเป็นระบบตัวเลขที่มีตำแหน่งทศนิยมลอยตัว ซึ่งในวงจรเข้ารหัสลับแบบเคออติกจะใช้การคำนวณในรูปแบบของระบบจำนวนโดยตรง

2.2.2.1 ระบบจำนวนโดยตรง

ในระบบจำนวนโดยตรง จุดทวินิยมของเลขฐานสอง (Binary point) ที่แบ่งระหว่างจำนวนเต็ม (Integer) และจำนวนทศนิยม (Fraction) จะถูกจำกัดให้คงที่อยู่ โดยบิตแรกจะเรียกว่า (Sign bit) ใช้ในการแสดงเครื่องหมายของตัวเลขโดยถ้าเป็นเครื่องหมายบวกจะแทนด้วย "0" และถ้าเป็นเครื่องหมายลบจะแทนด้วย "1" ซึ่งขนาดของตัวเลขจะแสดงในรูปแบบการยกกำลังของเลข 2 โดยหน้าจุดทศนิยมจะมีกำลังเป็นบวกรวมกำลัง 0 ด้วย และหลังจุดทศนิยมจะมีกำลังเป็นลบ ตัวอย่างการหาค่าจำนวนเต็มของเลขฐานสอง 01.101_2 ดังสมการที่ (2.1)

$$01.101_2 = \left(0\times 2^{1}\right)+\left(1\times 2^{0}\right)+\left(1\times 2^{-1}\right)+\left(0\times 2^{-2}\right)+\left(1\times 2^{-3}\right)=1.625_{10} \qquad (2.1)$$

ความเที่ยงตรงของระบบตัวเลขจะถูกกำหนดโดยบิตที่อยู่ทางขวาสุด หรือบิตนัยสำคัญต่ำสุด (Least Significant Bit) ส่วนขอบเขต (Range) ของระบบตัวเลขจะนิยามโดยช่วงระหว่างจำนวนเลขลบน้อยที่สุดถึงเลขบวกมากที่สุดที่สามารถแสดงได้ ซึ่งทศนิยมจะเป็นตัวกำหนดความเที่ยงตรงและขอบเขตของระบบตัวเลข ยกตัวอย่างเช่น เมื่อกำหนดตำแหน่งของทศนิยมให้อยู่ทางขวาสุด รูปแบบบิตจะมีเพียงเลขจำนวนเต็มและไม่มีเลขทศนิยม

2.2.2.2 ส่วนเติมเต็มสอง (2's Complement)

ส่วนเติมเต็มสองเป็นรูปแบบหนึ่งของการแสดงตัวเลขของระบบจำนวนโดยตรง ในการคำนวณแบบส่วนเติมเต็มสองจะเป็นการนำ 1's Complement บวกเข้ากับบิต "1" โดยจะบวกที่ตำแหน่งบิตนัยสำคัญต่ำสุด ในการทำ 2's Complement จะเป็นการทำให้เลขจำนวนบวกกลายเป็นจำนวนลบ เลขจำนวนลบกลายเป็นเลขจำนวนบวกแสดงตัวอย่างดังตารางที่ 2.1

ตารางที่ 2.1 ส่วนเติมเต็มสอง

ค่าเริ่มต้น	1's Complement	2's Complement
0101 (+5)	1010	1011 (-5)
1011 (-5)	0100	0101 (+5)

2.2.3 การเกิดเคออสบนวงจรกรองสัญญาณเชิงเลข[3]

2.2.3.1 ความเป็นพลวัตร (Dynamic)

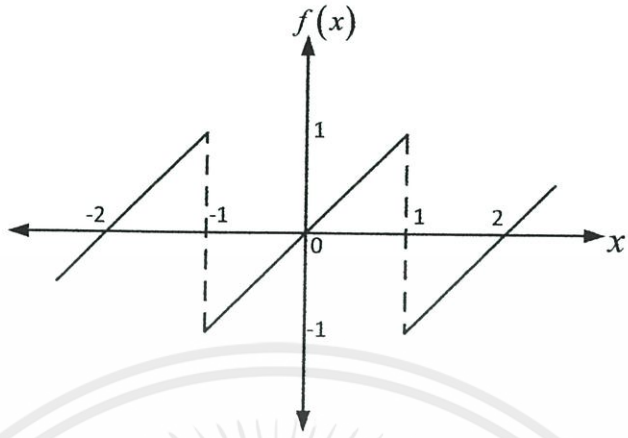
ในระบบของเคออสจะมีจุดที่แตกต่างจากระบบเข้ารหัสลับอื่น ๆ ในเรื่องระบบที่เป็นพลวัตรซึ่งถือว่าเป็นจุดเด่นที่สำคัญอย่างหนึ่งของระบบเคออส การเกิดความเป็นพลวัตรของระบบจะเกิดที่ Unit delay (z^{-1}) ทั้งสองตัวโดยจะเกิดการค้างค่าในรีจิสเตอร์ทั้ง 2 ตัว ค่าที่ค้างในรีจิสเตอร์จะถูกนำมาใช้เป็นเงื่อนไขค่าเริ่มต้น (Initial condition)

2.2.3.2 ความไม่เป็นเชิงเส้น (Nonlinearity)[4]

เมื่อนำระบบจำนวนโดยตรงมาใช้แทนค่าสัญญาณหรือนำมาใช้ในการประมวลผลของวงจรกรองสัญญาณเชิงเลขจะมีความคลาดเคลื่อนเกิดขึ้น โดยจะยกตัวอย่างการแปลงเลขฐานสิบไปเป็นเลขฐานสองของเลขจำนวนลบมาอธิบาย ซึ่งเลขจำนวนลบที่จะทำการแปลงได้แก่ -6.86_{10} เริ่มต้นด้วยการกำหนดจำนวนบิตที่จะแปลงขนาด โดยกำหนดขนาดเป็น 8 บิต จากนั้นทำการแปลงของเลขฐานสิบไปเป็นเลขฐานสองได้ $6.86_{10} = 0110.1110_2$ จากนั้นทำการสลับค่าบิตต่อบิตได้ 1001.0001_2 แล้วทำการบวกค่า 0.0001_2 เข้าไปจะได้ว่า $-6.86_{10} = 1001.0010_2$ (เท่ากับ -6.875_{10}) ซึ่งจะเห็นว่าค่าที่ได้ออกมานั้นมีความคลาดเคลื่อนไปจากค่าเดิม โดยความคลาดเคลื่อนที่เกิดจากการจัดระดับสัญญาณ (Quantization error) ทำให้เกิดความไม่เป็นเชิงเส้น

ความคลาดเคลื่อนที่เกิดจากการล้น (Overflow) เป็นอีกเหตุผลหนึ่งที่ทำให้ระบบมีคุณสมบัติไม่เป็นเชิงเส้น ความคลาดเคลื่อนจากการล้น หมายถึงการที่ผลลัพธ์ที่ได้จากการประมวลผลด้วยวิธีการบวกตัวเลขแบบส่วนเติมเต็มสอง มีค่ามากเกินกว่าขอบเขตที่สามารถแทนค่าในระบบจำนวนโดยตรงได้ ตัวอย่างเช่น การบวกเลขทศนิยมขนาด 8 บิต กำหนดให้ตำแหน่งของจุดทศนิยมอยู่หลังตัวเลขหลักที่ 8 (x.xxxxxxx) โดย x แทนค่าด้วยเลข 0 หรือ 1 ขอบเขตที่สามารถแทนค่าได้ด้วยตัวเลขชุดนี้มีค่าอยู่ระหว่าง -1 ถึง 0.9921875 กำหนดตัวตั้งเป็น 0.1000000 (เท่ากับ 0.5) และตัวบวกเป็น 0.1000000 (เท่ากับ 0.5) ผลลัพธ์จากการบวกมีค่าเท่ากับ 1.00000000 (เท่ากับ -1) จะเห็นได้เกิดความคลาดเคลื่อนจากการบวกเพราะผลลัพธ์ที่ได้จากการบวก 0.5 + 0.5 ไม่เท่ากับ 1 เพราะค่า 1 อยู่นอกขอบเขตที่สามารถแทนค่าได้

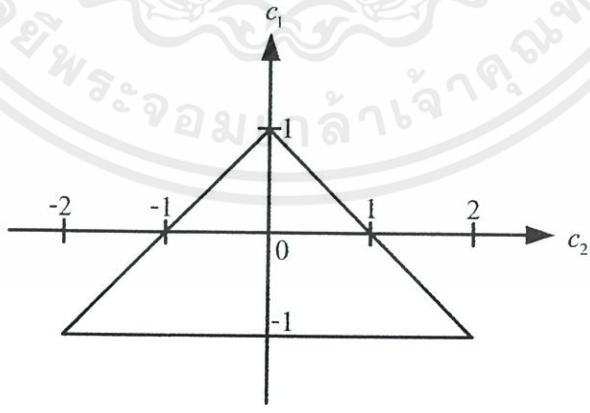
การใช้ระบบเลขส่วนเติมเต็มสองบนวงจรกรองสัญญาณเชิงเลขในระบบเคออส แสดงให้เห็นว่าเคออสนั้นเป็นระบบที่ไม่เป็นเชิงเส้น ซึ่งในการจำลองการล้นของระบบบนโปรแกรม MATLAB สามารถทำได้ด้วยการใช้ฟังก์ชันมอดูโล $f(x) = ((x+1) \bmod 2) - 1$ เมื่อนำมาพล็อตเป็นกราฟได้ดังรูปที่ 2.4



รูปที่ 2.4 ฟังก์ชันมอดูโล $f(x) = ((x+1) \bmod 2) - 1$

2.2.3.3 สามเหลี่ยมเสถียรภาพ (Stability triangle)

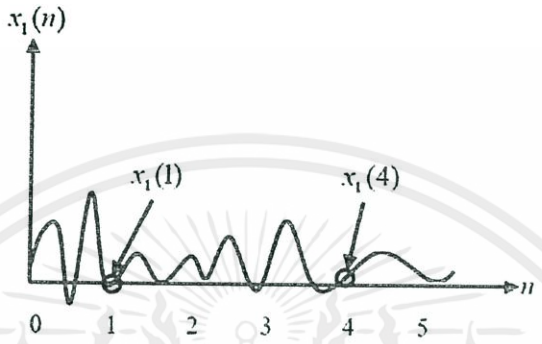
คุณสมบัติขาดเสถียรภาพของวงจรกรองสัญญาณเชิงเลข (Unstable) เป็นเงื่อนไขที่จำเป็นของการเกิดเคออสบนวงจรกรองสัญญาณเชิงเลข เกิดจากการเลือกตำแหน่งโพลให้อยู่นอกวงกลมหนึ่งหน่วย ซึ่งตำแหน่งของโพลหาได้จากสมการในเทอมส่วนของฟังก์ชันถ่ายโอน และเมื่อพิจารณาฟังก์ชันเทอมส่วน พบว่าการหาตำแหน่งโพลมีความสัมพันธ์กับค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลข ในกรณีวงจรกรองสัญญาณเชิงเลขอันดับที่ 2 ชนิด IIR filter ตำแหน่งของโพลมีความสัมพันธ์กับค่าสัมประสิทธิ์ c_1 และ c_2 โดยค่าของ c_1 และ c_2 ที่ทำให้ตำแหน่งของโพลอยู่นอกวงกลมหนึ่งหน่วยสามารถพล็อตเป็นความสัมพันธ์ได้ในรูปแบบที่เรียกว่า “สามเหลี่ยมเสถียรภาพ” แสดงดังรูปที่ 2.5



รูปที่ 2.5 ขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ

2.3 ค่าอัตสหสัมพันธ์ (Autocorrelation)[2]

ค่าอัตสหสัมพันธ์เป็นค่าบอกถึงความเหมือนกันหรือความคล้ายคลึงกันระหว่างสัญญาณสุ่มตัวแปรเดียวกัน ตัวอย่างเช่นการหาค่าสหสัมพันธ์ของ $x_1(1)$ และ $x_1(4)$ ดังรูปที่ 2.6

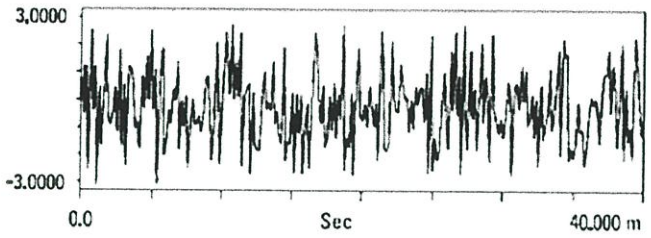


รูปที่ 2.6 ค่า $x_1(n)$ ที่เวลา n ต่างกัน

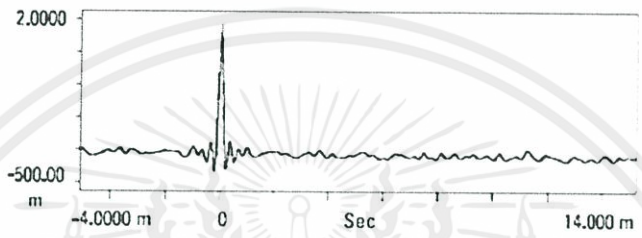
จากรูปที่ 2.6 การหาค่าสหสัมพันธ์ $r_{xx}(n,m)$ นี้เป็นการหาความสัมพันธ์ระหว่างสัญญาณ $x_1(1)$ และ $x_1(4)$ แต่ทั้งคู่เป็น x_1 เหมือนกัน ดังนั้นเราจึงสามารถเรียกได้ว่า $r_{xx}(n,m)$ เป็นอัตสหสัมพันธ์ของ x_1 ซึ่งสามารถหาค่าอัตสหสัมพันธ์ได้จากสมการที่ (2.2)

$$r_{xx}(n,m) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} x_i(n)\, x_i(m) \qquad (2.2)$$

ยกตัวอย่างดังรูปที่ 2.7 (ก) เป็นสัญญาณรบกวนแบบสุ่ม (Random noise) ซึ่งมีลักษณะไม่เหมือนกันเลยเมื่อเปรียบเทียบสัญญาณในช่วงเวลาต่าง ๆ กันหรือ Time shift ต่างๆ กัน ดังนั้นลักษณะกราฟอัตสหสัมพันธ์ในรูปที่ 2.7 (ข) ที่ได้จึงมีลักษณะมียอดสูงสุดในลักษณะ Spike ในช่วงเวลา Time shift เป็นศูนย์



(ก) สัญญาณรบกวน



(ข) กราฟอัตสหสัมพันธ์ของสัญญาณ

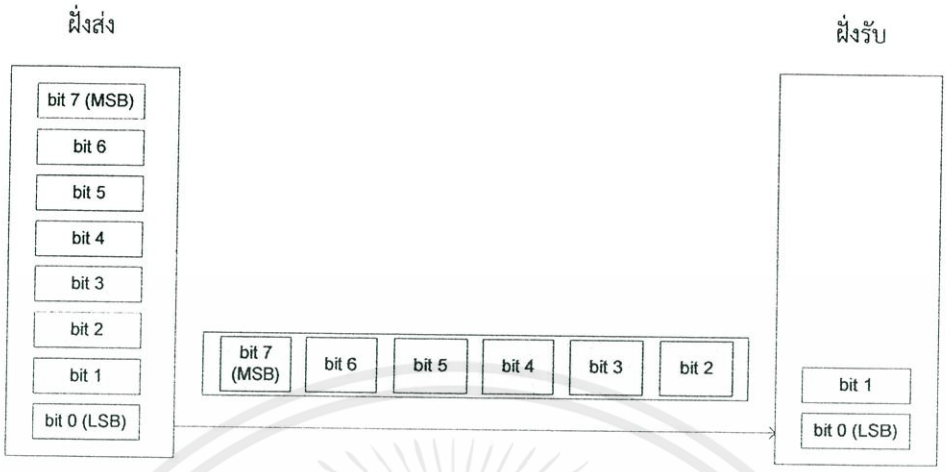
รูปที่ 2.7 สัญญาณรบกวน และกราฟอัตสหสัมพันธ์

จากตัวอย่างนี้ทำให้เราทราบว่ากราฟอัตสหสัมพันธ์จะมีลักษณะมีคาบเวลาเกิดซ้ำในทำนองเดียวกัน และมีคาบเวลาเท่ากันกับการเกิดคาบเวลาซ้ำของสัญญาณที่พิจารณา

2.4 การสื่อสารข้อมูลแบบอนุกรม (Serial communication) [5]

การส่งผ่านข้อความตัวอักษร โดยใช้การสื่อสารแบบอนุกรม ตัวอักษร 1 ตัว ที่ส่งออกไปนั้นจะประกอบเลขฐานสองจำนวน 8 บิต เรียงเป็นลำดับ ข้อมูลจะถูกส่งออกมาทีละบิตจนครบ 8 บิต และส่งจากต้นทางไปยังปลายทาง และปลายทางจะรับข้อมูล และเรียงลำดับจนครบ 8 บิต และปลายทางจะได้รับตัวอักษร 1 ตัว ดังรูปที่ 2.8

โดยทั่วไปแล้วข้อความที่ส่งจะประกอบไปด้วยกลุ่มตัวอักษรเรียงต่อกันในลักษณะสายข้อมูล ดังนั้นจึงต้องมีวิธีการจัดการสายข้อมูลยาว ๆ นี้ การสื่อสารข้อมูลแบบอนุกรมจึงสามารถแบ่งได้เป็น 2 วิธีคือ วิธีซิงโครนัส (Synchronous) และแบบอะซิงโครนัส (Asynchronous)



รูปที่ 2.8 การสื่อสารข้อมูลแบบอนุกรม

2.4.1 การสื่อสารข้อมูลแบบอนุกรมด้วยวิธีซิงโครนัส

การสื่อสารข้อมูลแบบอนุกรมด้วยวิธี**ซิงโครนัส**เป็นวิธีการจำแนกตัวอักษรแต่ละตัว โดยใช้สัญญาณนาฬิกาในการกำหนดจังหวะ ดังรูปที่ 2.9 ซึ่งการรับส่งข้อมูลแบบนี้ค่อนข้างมีคุณภาพ และส่งได้ที่ความเร็วสูง มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย แต่ต้องใช้ช่องสัญญาณมากขึ้น เนื่องจากต้องส่งสัญญาณนาฬิกาไปด้วย

รูปที่ 2.9 การส่งผ่านข้อมูลอนุกรมแบบซิงโครนัส

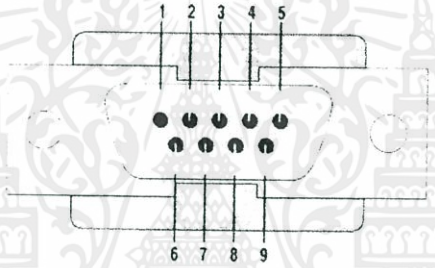
2.4.2 การสื่อสารข้อมูลแบบอนุกรมด้วยวิธีอะซิงโครนัส

รูปที่ 2.10 การส่งผ่านข้อมูลอนุกรมแบบอะซิงโครนัส

จากรูปที่ 2.10 การสื่อสารข้อมูลแบบอนุกรมด้วยวิธีอะซิงโครนัสเป็นวิธีที่ไม่ต้องใช้สัญญาณนาฬิกาในการกำหนดจังหวะ แต่ใช้วิธีกำหนด Start bit และ Stop bit ขึ้นไว้ที่หัวและท้ายทุก ๆ ความยาวบิต ของ 1 ตัวอักษร เพื่อการจำแนกตัวอักษรแต่ละตัว โดย Start bit จะกำหนดให้เป็นลอจิก 0 และ Stop bit เป็นลอจิก 1 การส่งข้อมูลแบบนี้จะต้องมีการกำหนดค่า Baud rate ให้ตรงกันทั้งต้นทางและปลายทาง

2.4.2.1 หัวเชื่อมต่อพอร์ตอนุกรมชนิด 9 ขา (Serial port DB9)

การสื่อสารข้อมูลแบบอนุกรมด้วยวิธีอะซิงโครนัสนั้นจะใช้หัวเชื่อมต่อพอร์ตอนุกรมชนิด 9 ขา (Serial port DB9) ดังรูปที่ 2.11 มีขาต่อใช้งานตามตารางที่ 2.2 โดยพื้นฐานมักต่อในรูปแบบ Null-modem นั่นคือ TxD ต้นทางต่อเข้ากับ RxD ปลายทาง ส่วน RxD ต้นทางต่อเข้ากับ TxD ปลายทาง และต่อขา GND ทั้งสองเข้าด้วยกัน



รูปที่ 2.11 ขาต่าง ๆ ของ Connector DB9

ตารางที่ 2.2 ขาการต่อใช้งานของ Connector DB9

dB9	Function	Abbreviation
Pin#1	Data Carrier Detect	CD
Pin#2	Receive Data	RxD
Pin#3	Transmitted Data	TxD
Pin#4	Data Terminal Ready	DTR
Pin#5	Signal Ground	GND
Pin#6	Data Set Ready	DSR
Pin#7	Requests To Send	RTS
Pin#8	Clear To Send	CTS
Pin#9	Ring Indicator	RI

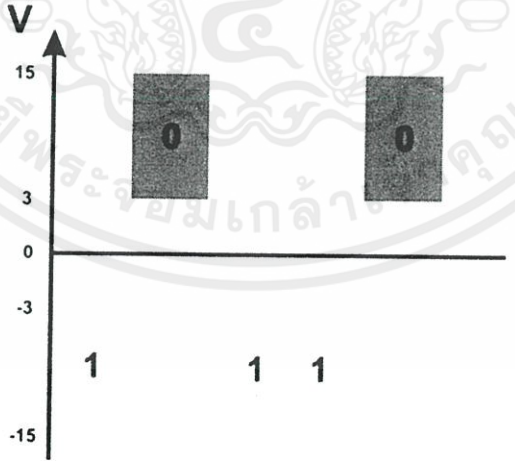
2.4.2.2 UART (Universal Asynchronous Receiver Transmitter)

UART คือพอร์ตที่ใช้รับส่งข้อมูลอนุกรมด้วยวิธีอะซิงโครนัส และทำการแจ้งข้อมูลอื่น ๆ เช่น บอดเรท (Baudrate) ความยาวของบิตต่อ 1 ตัวอักษร Stop bit, Parity bit และ Flow control เป็นต้น

2.4.2.3 ระดับแรงดัน RS-232 และ TTL

ข้อมูลแบบอนุกรมที่มีระดับแรงดัน 0 - 3.3 V หรือ 0 - 5 V คือระดับแรงดันของมาตารฐานการสื่อสารข้อมูลอนุกรมที่มีชื่อว่าว่า TTL หรือระดับแรงดัน TTL ดังรูปที่ 2.12 แต่เพื่อให้ส่งได้ระยะทางที่เพิ่มขึ้น จึงทำการเพิ่มระดับแรงดัน สำหรับลอจิก 0 มีระดับแรงดัน 3 - 15 V และลอจิก 1 มีระดับแรงดัน -3 - -15 V ใช้เป็นมาตารฐานการสื่อสารข้อมูลอนุกรมที่มีชื่อว่า RS-232 ดังรูปที่ 2.13

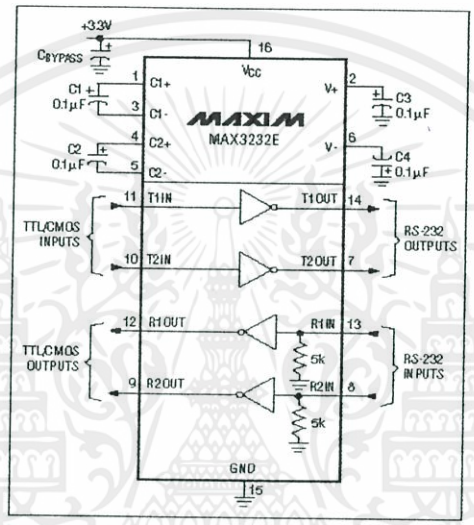
รูปที่ 2.12 ลอจิก และระดับแรงดันของ TTL



รูปที่ 2.13 ลอจิก และระดับแรงดันของ RS-232

2.4.2.4 การแปลงระดับสัญญาณด้วยไอซี MAX-3232

ในการเชื่อมต่ออุปกรณ์หนึ่งเข้ากับอีกอุปกรณ์จะต้องระมัดระวังเรื่องระดับแรงดัน และมาตรฐานของสัญญาณ หากอุปกรณ์ใดอุปกรณ์หนึ่งมีระดับแรงดันที่สูงกว่าอาจทำให้อุปกรณ์ที่มีระดับแรงดันต่ำกว่านั้นเสียหายได้ MAX-3232 เป็นไอซีที่ใช้ในการแปลงระดับแรงดันของสัญญาณจากมาตรฐาน RS-232 เป็น TTL และแปลงระดับสัญญาณจาก TTL ให้กลายเป็น RS-232 โดยไอซี MAX-3232 มีขาต่อใช้งานดังรูปที่ 2.14



รูปที่ 2.14 ขาต่อใช้งานของไอซี MAX-3232

2.5 โปรโตคอล TCP/IP [6]

อินเตอร์เน็ตนับได้ว่าเป็นเครือข่ายที่เปิดโอกาสให้เครือข่ายคอมพิวเตอร์อื่นเชื่อมโยงเข้ามาใช้งานหรือเป็นศูนย์กลางเชื่อมโยงเครือข่ายคอมพิวเตอร์อื่น ๆ แต่หากที่เกิดขึ้นในการเชื่อมโยงเครือข่ายคอมพิวเตอร์เข้าด้วยกันก็คือ แต่ละเครือข่ายใช้ คอมพิวเตอร์ต่างชนิด ต่างยี่ห้อ และระบบปฏิบัติการที่ต่างกัน มาตรฐานของ TCP (Transmission Control Protocol/Internet Protocol) จึงถูกใช้เป็นกุญแจสำคัญในการแก้ปัญหาเหล่านี้ โดยกลายเป็นระบบเปิดที่สมบูรณ์แบบที่มีการเชื่อมโยงคอมพิวเตอร์ได้ตั้งแต่เครื่องพีซี จนถึงเครื่องเมนเฟรม และไม่จำกัดระบบปฏิบัติการที่ใช้ TCP จึงเป็นมาตรฐานที่ทั่วโลกยอมรับ มีอุปกรณ์และซอฟต์แวร์ผลิตออกมาสนับสนุน TCP/IP มากมาย ดังนั้นจึงนับได้ว่า TCP/IP เป็นหัวใจของอินเตอร์เน็ตเลยทีเดียว

2.5.1 TCP/IP

TCP/IP เป็นข้อกำหนดเกี่ยวกับรูปแบบการเชื่อมโยงในเครือข่าย (Network protocol) จัดทำเพื่อใช้เป็นกฎเกณฑ์ให้เครื่องคอมพิวเตอร์ใช้งานร่วมกัน ในลักษณะของระบบเปิด

(Open System) คือไม่ว่าจะเป็นคอมพิวเตอร์ชนิดใดหรือระบบใดก็ตาม จะสามารถติดต่อสื่อสารและแลกเปลี่ยนข้อมูลกันใช้ได้

TCP/IP เป็นการกำหนดรูปแบบการสื่อสารระหว่างซอฟต์แวร์ การจัดโอนย้ายข้อมูล การแสดงสถานะของเครื่องคอมพิวเตอร์ที่อยู่บนเครือข่าย ตลอดจนกฎระเบียบต่าง ๆ ที่กำหนดให้ทำเมื่อเกิดความผิดพลาด หรือต้องทำเพื่อป้องกันเพื่อไม่ให้เกิดความผิดพลาด

TCP/IP เกิดจากการนำข้อกำหนดของรูปแบบต่าง ๆ กันมาใช้ร่วมกัน TCP และ IP ต่างก็เป็นรูปแบบหนึ่งของชุดข้อกำหนดนี้ ถูกออกแบบมาเพื่อใช้รับส่งหรือโอนย้ายข้อมูลระหว่างคอมพิวเตอร์ที่อยู่บนระบบเครือข่ายเดียวกัน หรือต่างเครือข่ายกันก็ได้และมีการจัดเตรียมข้อมูลสถานะของเครือข่ายขึ้นได้ทั้งระบบเครือข่ายเฉพาะบริเวณ ที่เรียกกันว่า LAN (Local Area Network) และเครือข่ายบริเวณกว้าง ที่เรียกกันว่า WAN (Wide Area Network) ไม่ได้ใช้งานเฉพาะกับอินเตอร์เน็ตเท่านั้น

2.5.2 ส่วนประกอบของ TCP/IP

จากที่กล่าวข้างต้นว่า TCP/IP ประกอบไปด้วยชุดข้อกำหนดรูปแบบต่าง ๆ TCP/IP สามารถแบ่งเป็นกลุ่ม ๆ ได้แก่ กลุ่มข้อกำหนดรูปแบบการขนส่ง รูปแบบเส้นทาง รูปแบบที่เกี่ยวกับที่อยู่เครือข่าย รูปแบบที่เกี่ยวกับเส้นทางการสื่อสารระหว่างเครือข่าย, รูปแบบที่เกี่ยวกับการบริการผู้ใช้ และรูปแบบอื่น ๆ ที่น่าสนใจ

2.5.2.1 กลุ่มข้อกำหนดรูปแบบการขนส่ง (Transport protocol)

ทำหน้าที่ควบคุมการเคลื่อนย้ายข้อมูล ระหว่างคอมพิวเตอร์สองเครื่อง แบ่งย่อยออกได้เป็น 2 ชนิด คือ

1) TCP (Transmission Control Protocol) เป็นการบริการแบบคอนเน็คชันออเรียนท์ (Connection oriented) ซึ่งคอมพิวเตอร์ด้านผู้รับและผู้ส่งจะต้องทำการสร้างเส้นทางการเชื่อมต่อไปยังด้านผู้รับก่อนการส่งข้อมูลหากัน

2) UDP (User Datagram Protocol) เป็นการให้บริการแบบคอนเน็คชันเล็ส (Connectionless) ซึ่งคอมพิวเตอร์ด้านผู้ส่งไม่จำเป็นต้องทำการสร้างเส้นทางเชื่อมต่อกับผู้รับก่อน เพียงรู้ที่อยู่ของด้านผู้รับแล้วใส่ที่อยู่นั้นไปกับข้อมูลที่ส่งออก ข้อมูลจะเดินทางตามเส้นทางต่าง ๆ เพื่อไปถึงปลายทางตามที่อยู่

2.5.2.2 กลุ่มข้อกำหนดเกี่ยวกับรูปแบบเส้นทาง (Routing protocol)

ทำหน้าที่พิจารณาเส้นทางที่ดีที่สุดที่ใช้ส่งข้อมูลและถ้ามีข้อมูลเป็นจำนวนมากหรือมีขนาดใหญ่ กลุ่มข้อมูลรูปแบบนี้ก็จะทำการแบ่งย่อยข้อมูลให้มีขนาดเหมาะสม

แล้วส่งออกไป เมื่อถึงผู้รับปลายทาง กลุ่มข้อมูลนี้มีหน้าที่ตรงข้าม คือ รวบรวมข้อมูลย่อยให้ถูกต้องก่อนการแสดงผล กลุ่มข้อกำหนดรูปแบบกลุ่มนี้ประกอบด้วย

1) ข้อกำหนดรูปแบบการส่งข้อมูลหรือ IP (Internet Protocol)

2) ข้อกำหนดเกี่ยวกับรูปแบบของข้อมูลข่าวสารเกี่ยวกับสถานะของ IP เช่น ข่าวสารความผิดพลาดและผลกระทบต่อเส้นทางเมื่อมีการเปลี่ยนแปลงฮาร์ดแวร์ในเครือข่าย เรียกว่า ICMP (Internet Control Message Protocol)

3) ข้อกำหนดรูปแบบหนึ่งที่ใช้สำหรับทำการพิจารณาวิธีการเลือกเส้นทางเพื่อให้ได้เส้นทางที่เหมาะสมกับข้อมูลมากที่สุด เรียกว่า RIP (Routing Information Protocol)

4) ข้อกำหนดรูปแบบอีกประเภทหนึ่งที่ใช้ตัดสินเลือกเส้นทางโดยพิจารณาทางที่สั้นที่สุดก่อน เรียกว่า OSPF (Open Shortest Path First)

2.5.2.3 กลุ่มข้อกำหนดรูปแบบที่อยู่เครือข่าย (Network address)

ทำหน้าที่พิจารณาที่อยู่ของเครือข่ายและเครื่องคอมพิวเตอร์ ไม่ว่าจะเป็นลักษณะตัวเลขหรือชื่อก็ตาม เพื่อความถูกต้องของข้อมูลที่จะไปยังผู้รับปลายทาง โดยที่ไม่ว่าเครือข่ายจะใหญ่โตสักเพียงใด หรือมีเครื่องคอมพิวเตอร์จำนวนมากก็ตามที่อยู่ต้องไม่ซ้ำกัน กลุ่มข้อกำหนดมีดังนี้

1) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เพื่อไม่ให้เกิดที่อยู่ซ้ำกัน เรียกว่า ARP (Address Resolution Protocol)

2) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เมื่อรู้ชื่อของเครือข่ายหรือคอมพิวเตอร์ เพราะการใช้งานจริงนั้นใช้เพียงที่อยู่ที่เป็นตัวเลข แต่ระบบชื่อจัดทำขึ้นเพื่อให้สะดวกต่อการใช้งานของผู้ใช้ เรียกว่า DNS (Domain Name System)

3) ข้อกำหนดรูปแบบที่พิจารณาตัวเลขที่อยู่เดียวกับ ARP แต่จะทำตรงข้ามกัน เรียกว่า RARP (Reverse Address Resolution Protocol)

2.5.2.4 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับเส้นทางการสื่อสารระหว่างเครือข่าย (Gateway Protocol) และสนับสนุนข้อมูลสถานะเพื่อนำไปใช้เลือกเส้นทางที่เหมาะสม

1) ข้อกำหนดรูปแบบนี้จะทำการถ่ายโอนข้อมูลเส้นทางกันระหว่างเกตเวย์ กับเครือข่ายภายนอกเพื่อทำการสื่อสาร เรียกว่า EGP (Exterior Gateway Protocol)

2) ข้อกำหนดรูปแบบที่ทำงานถ่ายโอนข้อมูลเส้นทางระหว่างเกตเวย์ กับเกตเวย์ เรียกว่า GGP (Gateway to Gateway Protocol)

3) ข้อกำหนดในแบบที่ถ่ายโอนข้อมูลเส้นทางกันภายในเครือข่ายเดียวกัน เรียกว่า IGP (Interior Gateway Protocol)

2.5.2.5 กลุ่มข้อกำหนดรูปแบบเกี่ยวกับการบริการผู้ใช้ (User services)

ผู้ใช้สามารถใช้ข้อกำหนดรูปแบบได้โดยตรง ข้อกำหนดรูปแบบนี้ประกอบด้วย

1) ข้อกำหนดรูปแบบการอ่านโปรแกรมควบคุมการทำงานจากคอมพิวเตอร์ให้บริการ (Server computer) มาที่คอมพิวเตอร์ที่อยู่บนเครือข่าย หลังจากที่ผู้ใช้เปิดคอมพิวเตอร์ขึ้น เรียกว่า BOOTP (Boot Protocol)

2) ข้อกำหนดรูปแบบที่ให้บริการถ่ายโอนไฟล์ข้อมูลระหว่างคอมพิวเตอร์แต่ละเครื่อง ซึ่งอาจอยู่บนเครือข่ายเดียวกันหรือต่างเครือข่ายกันก็ได้ เรียกว่า FTP (File Transfer Protocol)

3) ข้อกำหนดรูปแบบที่ให้บริการเกี่ยวกับการควบคุมการติดต่อระยะทางไกล เรียกว่า TELNET

2.5.2.6 กลุ่มข้อกำหนดรูปแบบอื่นๆ และบริการที่น่าสนใจมีดังนี้

1) ข้อกำหนดรูปแบบที่ทำให้ผู้ใช้สามารถเข้าใช้งานไฟล์ข้อมูลและดูไฟล์ข้อมูลซึ่งอยู่ในคอมพิวเตอร์เครื่องอื่นได้ เรียกว่า NFS (Network File System)

2) ข้อกำหนดรูปแบบที่ให้บริการกับ User accounts ข้ามเครือข่าย เช่น Logins และ Password เรียกข้อกำหนดนี้ว่า NIS (Network Information System) เป็น

3) ข้อกำหนดรูปแบบที่อำนวยความสะดวกให้กับโปรแกรมประยุกต์ที่ใช้งานกับการควบคุมระยะทางไกล เรียกว่า RPC (Remote Procedure Call)

4) ข้อกำหนดรูปแบบที่อำนวยความสะดวกให้กับโปรแกรมประยุกต์ที่ใช้งานกับการควบคุมระยะทางไกล เรียกว่า STMP (Simple Mail Transfer Protocol)

5) ข้อกำหนดรูปแบบที่ให้บริการข่าวสารต่าง ๆ ที่แสดงสถานะของเครือข่าย และอุปกรณ์ที่ต่ออยู่บนเครือข่าย เรียกข้อกำหนดนี้ว่า SNMP (Simple Network Management Protocol)

2.5.3 โครงสร้างของชุดโปรโตคอล TCP/IP

2.5.3.1. Application layer

ในชั้นนี้ประกอบด้วยโปรแกรมประยุกต์ที่ใช้ในเครือข่าย เช่น โปรแกรมส่งถ่ายข้อมูล (File - transport) ชั้นนี้เทียบเท่ากับ Application layer, Presentation layer, และ Session layer ใน OSI Model รวมกัน

2.5.3.2 Transport layer

ชั้นนี้โปรโตคอล TCP/IP มี socket เป็นจุดปลาย (End point) ในการสื่อสารซึ่ง Socket นี้ประกอบด้วย หมายเลขของคอมพิวเตอร์และหมายเลขพอร์ต (Port) ของเครื่องที่ต้องการส่งข้อมูลไปถึงในชั้นนี้มีการรับรองให้ถึงที่หมายและลำดับข้อมูลที่ส่งโดยปราศจากข้อมูลซ้ำซ้อนโดยในชั้นนี้มีโปรโตคอลหลัก 2 ตัวคือ TCP และ UDP

2.6.3.3 Internet layer

ในชั้นนี้มีการกำหนด Datagram และทำการหาเส้นทางการส่งแบบ Connectionless เนื่องจากไม่มีการเชื่อมต่อระหว่างต้นทางกับปลายทาง

2.5.3.4 Network interface physical layer

ทำหน้าที่ควบคุมตัวกลางที่ใช้สื่อสารข้อมูลและรูปแบบการเชื่อมต่อในทางกายภาพ ชั้นนี้จะแบ่งข้อมูลออกเป็นส่วน ๆ เรียกว่า Frame หรือ Packet และส่งข้อมูลที่ให้ไปยังปลายทางที่เชื่อมต่อกันอยู่บนเครือข่ายเดียวกัน

เมื่อเรานำ TCP/IP ไปเปรียบเทียบกับ OSI Model จะเห็นได้ว่ามีความแตกต่างเพียงเล็กน้อยเท่านั้นดังตารางที่ 2.3

ตารางที่ 2.3 โครงสร้างของชุดโปรโตคอล TCP/IP เปรียบเทียบกับ OSI Model

OSI Model	TCP/IP (Internet)
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data Link	Network Interface Physical
Physical	

2.5.4 โปรโตคอล UDP เบื้องต้น

UDP เป็นโปรโตคอลที่ทำงานในลักษณะ Connectionless ซึ่งจะแตกต่างจาก TCP ตรงที่ UDP ไม่มีการสร้างเส้นทางเชื่อมต่อ และการติดต่อสื่อสารกันนั้นเครื่องคอมพิวเตอร์แต่ละเครื่องนั้นอาจจะเป็น Client หรือ Server ก็ได้

ในการที่จะส่งข้อมูลหากันนั้น เครื่องที่เป็น Client จะเพียงแค่กำหนดหมายเลขพอร์ตใน Local port property และกำหนดหมายเลขพอร์ตใน Remote port property ซึ่งจะต้องเป็นหมายเลขเดียวกันที่กำหนดไว้ที่เครื่อง Server จากนั้นก็ทำการเรียก Send data method เพื่อส่งข้อมูล และเมื่อเครื่อง Client ต้องการจะอ่านข้อมูลก็เพียงเรียก Get Data method ในขณะที่มีเหตุการณ์ Data arrival event

2.6 ระบบสมองกลฝังตัว (Embedded) [7]

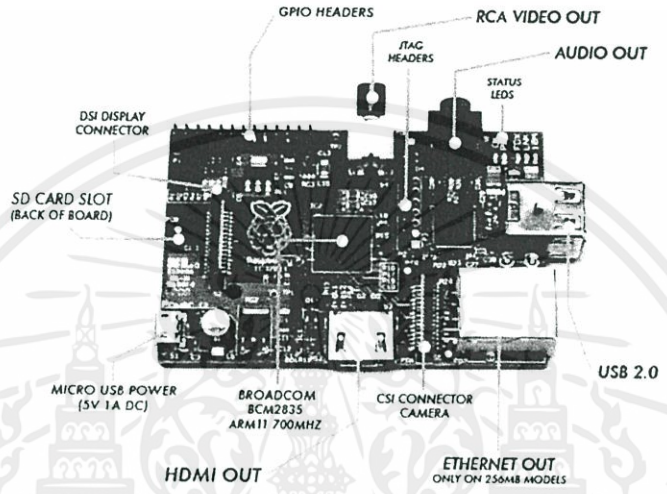
ระบบสมองกลฝังตัว (Embedded system) หมายถึงระบบประมวลผล ที่ใช้ชิปหรือไมโครโพรเซสเซอร์ที่ออกแบบมาโดยเฉพาะ เป็นระบบคอมพิวเตอร์ขนาดจิ๋วที่ฝังไว้ในอุปกรณ์เครื่องใช้ไฟฟ้า และเครื่องเล่นอิเล็กทรอนิกส์ต่าง ๆ เพื่อเพิ่มความฉลาด และความสามารถให้กับอุปกรณ์ผ่านซอฟต์แวร์ที่ต่างจากระบบประมวลผลของเครื่องคอมพิวเตอร์ทั่วไป ระบบสมองกลฝังตัวถูกนำมาใช้กันอย่างแพร่หลายในยานพาหนะ เครื่องใช้ไฟฟ้าในบ้านและสำนักงาน อุปกรณ์อิเล็กทรอนิกส์ เทคโนโลยีซอฟต์แวร์ เทคโนโลยีฮาร์ดแวร์ เทคโนโลยีเครือข่ายเน็ตเวิร์ก เทคโนโลยีด้านการสื่อสาร เทคโนโลยีเครื่องกลและของเล่นต่าง ๆ คำว่าระบบสมองกลฝังตัวนี้เป็นระบบประมวลผลเช่นเดียวกับระบบคอมพิวเตอร์ แต่ว่าระบบนี้จะฝังตัวลงในอุปกรณ์อื่น ๆ ที่ไม่ใช่เครื่องคอมพิวเตอร์ ในปัจจุบันระบบสมองกลฝังตัวได้มีการพัฒนามากขึ้น โดยในระบบสมองกลฝังตัวอาจจะประกอบไปด้วยไมโครคอนโทรลเลอร์ หรือ ไมโครโปรเซสเซอร์ อุปกรณ์ที่ใช้ระบบสมองกลฝังตัวที่เห็นได้ชัดเช่น โทรศัพท์มือถือ และในระบบสมองกลฝังตัวยังมีการใส่ระบบปฏิบัติการต่าง ๆ แตกต่างกันไปอีกด้วย ดังนั้น ระบบสมองกลฝังตัวอาจจะทำงานได้ตั้งแต่ควบคุมหลอดไฟจนไปถึงใช้ในยานอวกาศ ซึ่งในปัจจุบันในมีระบบฝังตัวที่มีความสามารถมากว่าในอดีต และเป็นที่นิยมในการนำมาใช้ในการพัฒนา ในปัจจุบัน ชื่อของบอร์ด คือ Raspberry Pi ซึ่ง Raspberry Pi เป็นบอร์ดไมโครคอมพิวเตอร์แบบแผ่นเดี่ยวที่บรรจุความสามารถไว้จำนวนมาก รองรับระบบปฏิบัติการ Linux บรรจุลงใน SD Card สำหรับการพัฒนาไปสู่บอร์ด Embedded linux พร้อมจุดเชื่อมต่ออุปกรณ์อินพุตเอาท์พุตทั้งผ่านพอร์ต USB, LAN, HDMI ช่องสัญญาณภาพและ GPIO สำหรับต่อกับวงจรหรืออุปกรณ์อิเล็กทรอนิกส์

2.6.1 คุณสมบัติทางเทคนิคของอุปกรณ์ Raspberry Pi

อุปกรณ์ Raspberry Pi มีชิปควบคุมหลักคือ Broadcom BCM2835 มีหน่วยประมวลผลกลาง CPU: ARM11 Core: ARM1176JZF-S ความไวของ CPU 700 MHz มีหน่วยประมวลกราฟิกหรือ GPU และมีหน่วยความจำ SDRAM: 256 MB ไว้ภายในตัวเดียวกัน

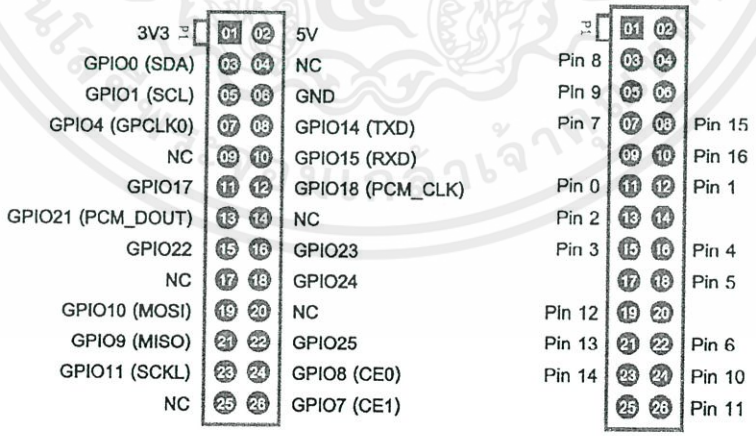
มีจุดเชื่อมต่อ USB2.0 (2 พอร์ต) RCA jack และ HDMI เอาท์พุตสัญญาณวิดีโอสำหรับต่อจอแสดงผลที่มีจุดต่อแบบ RCA ตัวเมียหรือ HDMI จุดต่อเอาท์พุตเสียงเป็น Stereo jack 3.5 mm จุดต่อ Ethernet หรือจุดต่อระบบ LAN คอนเน็กเตอร์หรือจุดต่อพอร์ตอินพุต

เอาท์พุต (General Purpose Input Output: GPIO) ที่มีขาต่อกับบัส SPI (Serial Peripheral Interface Bus) I^2C I^2S ขาสัญญาณรับส่งข้อมูลอนุกรมหรือ UART และซ็อกเก็ตของ SD Card สำหรับเสียบ SD Card ที่ติดตั้งระบบปฏิบัติการเรียบร้อยแล้ว ต้องการไฟเลี้ยง : +5V กระแส 700 mA แสดงดังรูปที่ 2.15



รูปที่ 2.15 ส่วนประกอบของ Raspberry Pi

2.6.2 การจัดเรียงขาของพอร์ต GPIO ของอุปกรณ์ Raspberry Pi



รูปที่ 2.16 การจัดเรียงขาของพอร์ต GPIO

จากรูปที่ 2.16 บอร์ด Raspberry Pi มีพอร์ตอินพุตเอาท์พุตดิจิตอลหรือ GPIO สำหรับติดต่อกับอุปกรณ์ภายนอกรวม 17 ขา โดยมีการจัดเรียงขามาตรฐานตามรูปซ้าย และหากใช้โปรแกรม WiringPi จะมีการจัดเรียงขาอินพุตเอาท์พุตใหม่ตามรูปขวา

2.7 ภาษาไพธอน (Python) [8], [9]

ภาษาไพธอนเป็นเครื่องมือที่มีการเชื่อมต่อคล้ายกับ MATLAB กล่าวคือ ประมวลผลคำสั่งทีละบรรทัดหรืออาจจะเขียนโปรแกรมเป็นซอร์สไฟล์ (Source file) จากนั้นสั่งให้ประมวลผลในภายหลังได้เช่นกัน ดังนั้นภาษาไพธอนจึงได้รับความสนใจอย่างมากและถูกนำไปใช้งานในด้านต่าง ๆ อย่างกว้างขวางและมีประสิทธิภาพสูงมาก ข้อแตกต่างระหว่างภาษาไพธอนและ MATLAB คือ ภาษาไพธอนนั้นเป็น โอเพนซอร์ส (Open source) ที่อนุญาตให้นักเขียนโปรแกรมสามารถพัฒนาให้โปรแกรมมีความสามารถสูงขึ้น ซึ่งเป็นสิ่งสำคัญที่ทำให้มีผู้เชี่ยวชาญในหลากหลายสาขา ต่างร่วมกันสร้างเครื่องมือมาเพื่อประกอบให้ภาษาไพธอนเป็นโปรแกรมที่ดีและนำไปใช้งานด้านต่าง ๆ ได้อย่างกว้างขวาง ภายในภาษาไพธอนประกอบไปด้วยคำสั่ง ฟังก์ชัน และโมดูลด้านต่าง ๆ ที่สามารถเรียกใช้งานได้อย่างสะดวก

2.7.1 ประวัติความเป็นมาของภาษาไพธอน

ช่วงต้นปี ค.ศ. 1990 (พ.ศ. 2553) Guido Van Rossum ได้ออกแบบและสร้างภาษาไพธอน โดยโค้ดไพธอนทั้งหมดถูกสร้างขึ้นจากภาษาซี โดยภาษาไพธอนทุกเวอร์ชันเป็นโอเพนซอร์สโดยรายละเอียดแต่ละเวอร์ชันแสดงได้ดังนี้

1) เวอร์ชัน 0.90 ถึง 1.2 พัฒนาในช่วงปี ค.ศ. 1991-1995
2) เวอร์ชัน 1.3 ถึง 1.5.2 พัฒนาในช่วงปี ค.ศ. 1995-1999
3) เวอร์ชัน 1.6 พัฒนาในช่วงปี ค.ศ. 2000
4) เวอร์ชัน 1.6.1 พัฒนาในช่วงปี ค.ศ. 2001
5) เวอร์ชัน 2.1.2 พัฒนาในช่วงปี ค.ศ. 2002
6) เวอร์ชัน 2.2.1 พัฒนาในช่วงปี ค.ศ. 2002

2.7.2 คุณลักษณะเด่นของภาษาไพธอน

1) สนับสนุนแนวคิดแบบ OOP (Object Oriented Programming) คือ การมองโค้ดให้อยู่ในรูปวัตถุ (Object) ที่ประกอบไปด้วย ส่วนที่เป็นข้อมูลเรียกว่า Attribute และส่วนที่เป็นการทำงานเรียกว่า Method

2) เป็นโอเพนซอร์ส ซึ่งทำให้ภาษาไพธอนสามารถพัฒนาได้กว้างขวาง

3) สามารถนำโค้ดไป Run บนระบบปฏิบัติการอื่น ๆ ได้ (Portable) เช่น Linux, Microsoft Windows, Amiga, Be-OS, OS/2, VMS, ONX

4) Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้สะดวก

5) ภาษาไพธอนมี Built-in object types โครงสร้างของข้อมูลที่สามารถใช้ได้ภายในภาษาไพธอนประกอบด้วยตัวแปรชนิดสายอักขระ (Strings) ตัวเลขจำนวนเต็ม (Int) และตัวเลขทศนิยม (Float) ทูเพิล (Tuple) ลิสต์ (List) และดิกชันนารี (Dictionary) ที่สามารถเรียกใช้งานได้ทันทีโดยไม่ต้องประกาศชนิดของตัวแปรก่อน

6) มีเครื่องมือต่าง ๆ มากมาย เช่น การประมวลผลเท็กซ์ไฟล์ การเรียงข้อมูล การต่อเชื่อมสตริง การตรวจสอบเงื่อนไขของข้อความ การแทนที่คำ เป็นต้น

7) มีการจัดการหน่วยความจำอัตโนมัติ และสามารถจัดการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้สามารถทำงานได้อย่างมีประสิทธิภาพ

8) สามารถฝังโค้ดภาษาไพธอนไว้ภายในโค้ดภาษา C/C++ ได้

9) ภาษาไพธอนอนุญาตให้ผู้เขียนโปรแกรมสามารถทำการสร้าง Dynamic Link Library (DLL) เพื่อใช้ร่วมกับภาษาไพธอนได้

10) ภาษาไพธอนประกอบด้วยโมดูลสำหรับการสร้าง Internet Script และติดต่อกับอินเทอร์เน็ตผ่าน Sockets และทำหน้าที่เป็น CGI Script ตลอดจนใช้งานคำสั่ง FTP Glopher, XML และอื่น ๆ อีกมาก

11) ประมวลผลทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ได้

12) มีฟังก์ชันสนับสนุนฐานข้อมูล

13) มี Library ที่สนับสนุนด้านการสร้างภาพ และการทำกราฟฟิก เช่น ทำภาพเบลอหรือชัด เขียนข้อความบนภาพ ตลอดจนบันทึกไฟล์ในรูปแบบต่าง ๆ

2.7.3 รูปแบบการเขียนโปรแกรมภาษาไพธอน

```
python statement ; python statement ;
python statement
python statement
# comment
```

รูปที่ 2.17 รูปแบบคำสั่ง

จากรูปที่ 2.17 สามารถเขียนโค้ดภาษาไพธอนได้โดยตรงบน Python Shell หรือ Python Command Line และสามารถเขียนเก็บไว้ในเท็กซ์ไฟล์ธรรมดาและบันทึกนามสกุล .py ได้เช่นเดียวกัน ภาษาไพธอนไม่จำเป็นต้องมีฟังก์ชัน main() เหมือนภาษาอื่น ๆ

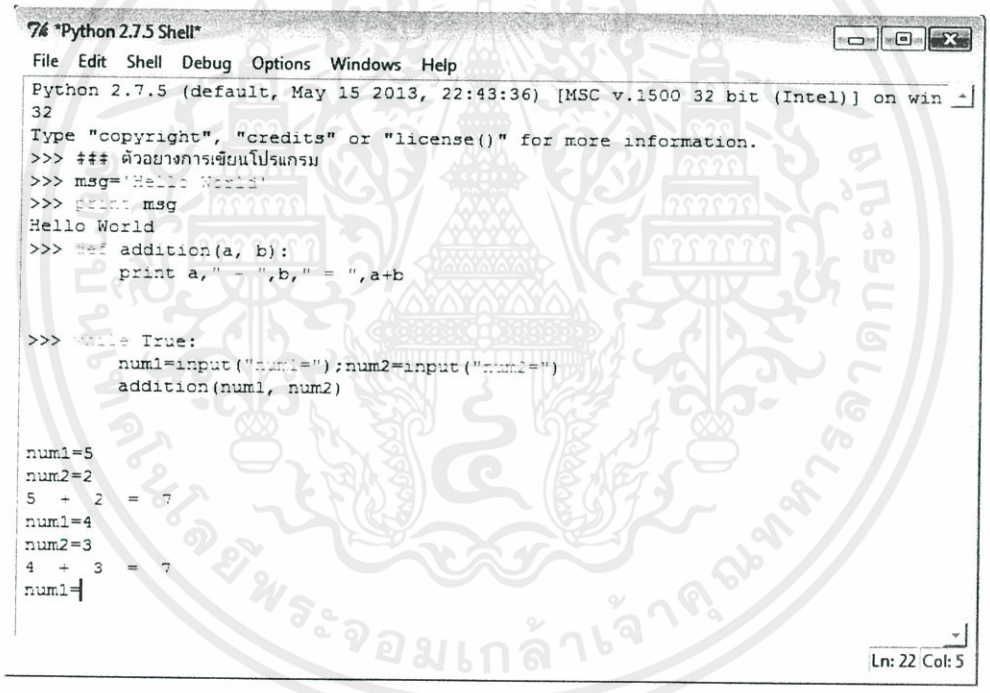
โดยมีกฎเกณฑ์ในการเขียนคำสั่งดังนี้

1) เขียนด้วยอักษรพิมพ์เล็กหรือพิมพ์ใหญ่ขึ้นอยู่กับฟังก์ชันที่เรียกใช้

2) การเขียนคำสั่งสามารถเขียนแบบต่อเนื่องกันโดยการคั่นด้วยเครื่องหมาย ; (Semi - colon) แต่เพื่อความสะดวกจึงนิยมเขียน 1 คำสั่งต่อ 1 บรรทัด โดยไม่ต้องใช้เครื่องหมาย ; ปิดท้ายแต่ละบรรทัด

3) การประกาศฟังก์ชันและคลาส รวมทั้งการวนลูป (Loop) ต้องจบบรรทัดด้วยเครื่องหมาย : (Colon)

นอกจากนี้ยังมีการกำหนดสีต่าง ๆ ในโปแกรมสำหรับไวยากรณ์ของคำสั่ง โดยสีแดงแทนคอมเมนต์หรือการอธิบาย สีเขียวแทนสายอักขระ สีส้มแทนคำสงวนหรือคีย์เวิร์ด และสีน้ำเงินแทนการแสดงผล, การประกาศชื่อฟังก์ชัน ตัวอย่างการเขียนโปรแกรมดังรูปที่ 2.18



รูปที่ 2.18 ตัวอย่างการเขียนโค้ดภาษาไพธอน

2.7.4 ภาษาไพธอนกับการสื่อสารข้อมูลแบบอนุกรม [10]

ภาษาไพธอนสามารถเขียนคำสั่งเกี่ยวกับการสื่อสารข้อมูลแบบอนุกรมได้โดยมีโมดูล pySerial ทำหน้าที่เชื่อมโยงภาษาไพธอนให้สามารถส่งข้อมูลออกทางพอร์ตอนุกรม รองรับการส่งข้อมูลได้หลายขนาด และใช้ในการตั้งค่าพารามิเตอร์ของพอร์ตอนุกรมผ่านภาษาไพธอนได้

2.7.4.1 การตั้งค่าพารามิเตอร์ (Parameters) และการติดต่อพอร์ตอนุกรม

การตั้งค่าพารามิเตอร์เริ่มจากคำสั่ง serial.Serial() และติดต่อกับพอร์ตอนุกรมด้วยคำสั่ง ser.open() โดยตั้งเงื่อนไขว่าถ้าพอร์ตปิดอยู่ให้เปิดพอร์ตขึ้น ดังรูปที่ 2.19

```
import serial
ser = serial.Serial(port = "COM5", baudrate = 9600)
if not ser.isOpen():
        ser.open()
```

รูปที่ 2.19 การตั้งค่าพารามิเตอร์ และการติดต่อพอร์ตอนุกรม

2.7.4.2 การส่งข้อมูลผ่านพอร์ตอนุกรม

เมื่อพอร์ตอนุกรมอยู่ในสถานะพร้อมใช้งานแล้ว สามารถส่งข้อมูลผ่านพอร์ตอนุกรมด้วยคำสั่ง ser.write(msg) โดยรับค่าจากแป้นพิมพ์และเก็บในตัวแปร msg ด้วยคำสั่ง m = raw_input() และปิดพอร์ตอนุกรมด้วยคำสั่ง ser.close() ดังรูป 2.20

```
msg = raw_input("Type message to send")
ser.write(msg)
ser.close()
```

รูปที่ 2.20 การส่งข้อมูลผ่านพอร์ตอนุกรม

2.7.4.3 การรับข้อมูลผ่านพอร์ตอนุกรม

การรับข้อมูลหรือการอ่านข้อมูลจากพอร์ตอนุกรมนั้นสามารถทำได้ด้วยคำสั่ง data = ser.read() อ่านข้อมูลแล้วเก็บในตัวแปร data จากนั้นทำการแสดงข้อมูลที่รับได้ด้วยคำสั่ง print data และทำการปิดพอร์ตอนุกรมดังรูปที่ 2.21

```
data = ser.read()
print data
ser.close()
```

รูปที่ 2.21 การอ่านข้อมูลจากพอร์ตอนุกรม

2.7.5 ภาษาไพธอนกับการใช้งานด้านเน็ตเวิร์คโปรแกรมมิ่ง [11]

2.7.5.1 การสร้าง Socket

ความรู้เบื้องต้นสำหรับการใช้งานด้านเน็ตเวิร์คโปรแกรมมิ่งคือการใช้งาน Socket ซึ่งเป็นที่อยู่เบื้องหลังการสื่อสารผ่านเน็ตเวิร์คสำหรับทุกโปรโตคอล ยกตัวอย่างเช่น การเชื่อมต่อกับ www.facebook.com ผ่านเว็บบราวเซอร์ การทำงานของเว็บบราวเซอร์เริ่มจากการสร้าง Socket สำหรับโปรโตคอลที่ใช้ในการสื่อสารกับ Remote sever (ในที่นี้คือ Facebook) จากนั้นสร้างการเชื่อมต่อ และแลกเปลี่ยนข้อมูลกับ Remote sever ตามลำดับ

การสร้าง Socket เริ่มจากการใช้งานโมดูล socket และใช้คำสั่ง socket.socket ตามลำดับ แสดงดังรูปที่ 2.22

```
#Socket client example in python
import socket   #for sockets
#create an AF_INET, STREAM socket (TCP)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "Socket Created"
```

รูปที่ 2.22 การสร้าง Socket ด้วยภาษาไพธอน

คำสั่ง socket.socket(a,b) จะมีแปรสองตัวโดยที่ตัวแรกเป็นชนิดของ Address Family แบ่งเป็น 3 ประเภท ได้แก่ socket.AF_INET, socket.AF_INET6 และ socket.AF_UNIX โดยชนิดที่แสดงดังรูปเป็น Address Family Internet ในส่วนของตัวแปรตัวที่สองคือชนิดของ Socket ให้เหมาะสมกับโปรโตคอลที่เลือกใช้ โดยชนิดที่แสดงดังรูป คือ socket.SOCK_STREAM ใช้สำหรับการสื่อสารด้วยโปรโตคอล TCP แต่สำหรับการใช้งานที่ต้องการสื่อสารด้วยโปรโตคอล UDP เลือกชนิดของ socket เป็น socket.SOCK_DGRAM

2.7.5.2 การควบคุม Error ในการสร้าง Socket

ความผิดพลาดในการสร้าง Socket อาจเกิดขึ้นได้ ดังนั้นเพื่อให้ผู้ใช้ทราบถึงความผิดพลาด จึงต้องมีการควบคุม Error โดยใช้คำสั่ง socket.error เป็นเงื่อนไข แสดงดังรูปที่ 2.23

การใช้คำสั่ง try คู่กับ except เพื่อตรวจสอบ Error ที่อาจเกิดขึ้นได้ เมื่อมี Error เกิดขึ้น(Socket.error) จะทำการเก็บค่า Error ไว้ในตัวแปร msg ซึ่งเป็นตัวแปรชนิดลิสต์ โดย msg[0] เก็บค่า 'Error code' และ msg[1] เก็บค่า 'Error message' ตามลำดับ แสดงดังรูปที่ 2.23

```
#Socket client example in python
import socket   #for sockets
import sys #for exit
try :
        #create an AF_INET, STREAM socket (TCP)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print "Socket Created"
except socket.error, msg :
        print 'Failed to create socket'
        print 'Error code: ' + str(msg[0]) + ' , Error message : ' + msg[1]
        sys.exit()
print "Socket Created"
```

รูปที่ 2.23 การควบคุม Error ในการสร้าง Socket

2.7.5.3 การเขียน Server socket สำหรับโปรโตคอล TCP

การเขียน Server socket สำหรับโปรโตคอล TCP ผู้เขียนต้องทำความเข้าใจฟังก์ชันการทำงานซึ่งอยู่ในโมดูล socket ซึ่งฟังก์ชันที่ใช้ในการเขียนมี 5 ฟังก์ชันดังนี้

1) การสร้าง TCP socket ซึ่งได้อธิบายไปแล้วข้างต้น

2) การเชื่อม Socket เข้ากับหมายเลขไอพี (IP address) และพอร์ตของ Server

3) รอการเชื่อมต่อของ Client

4) ยอมรับการเชื่อมต่อของ Client

5) รับ - ส่งข้อมูลระหว่าง Server กับ Client

การเชื่อม Socket เข้ากับหมายเลข IP และพอร์ตทำหลังจากที่สร้าง Socket เสร็จแล้ว โดยใช้คำสั่ง s.bind((IPaddress, Port)) โดยตัวแปร s คือ Socket ที่สร้างไว้ ตัวแปร IPaddress เป็นหมายเลขไอพีของคอมพิวเตอร์ที่ใช้งานเป็น Server และ Port กำหนดพอร์ตที่ใช้ในการรับ-ส่งข้อมูล ในส่วนการควบคุม Error ใช้คำสั่ง socket.error แสดงดังรูปที่ 2.24

```
IPaddress = ' ' #Symbolic name meaning all available interfaces
Port = 9000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((IPaddress, Port))
except socket.error , msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()
print 'Socket bind complete'
```

รูปที่ 2.24 การเชื่อม Socket เข้ากับหมายเลขไอพีและพอร์ตของ TCP Server

หลังจากเชื่อม Socket เข้ากับหมายเลขไอพีและพอร์ตของ Server คำสั่งที่ใช้สำหรับรอการเชื่อมต่อจาก Client คือ s.listen(n) โดยตัวแปร n คือจำนวนของ Client ที่ถูกกำหนดให้สื่อสารได้มากที่สุด n Client ตัวอย่างการเขียนแสดงดังรูปที่ 2.25

```
s.listen(20)
print 'Socket now listening'
```

รูปที่ 2.25 รอการเชื่อมต่อจาก Client

ต่อจากคำสั่ง s.listen(n) คือคำสั่ง s.accept() เป็นคำสั่งยอมรับการเชื่อมต่อจาก Client โดยจากคำสั่งจะรีเทิร์นค่าและเก็บค่าในตัวแปร 2 ค่าด้วยกัน คือ ค่าของ Socket ที่ใช้ในการรับส่งข้อมูล และค่าของหมายเลขที่อยู่ที่ผูกอยู่กับ Socket ในส่วนอื่น ๆ ของการเชื่อมต่อ แสดงดังรูปที่ 2.26

```
s.new , addr = s.accept()
#display client information
print 'Connected with ' + addr[0] + ':' + str(addr[1])
```

รูปที่ 2.26 ยอมรับการเชื่อมต่อจาก Client

เมื่อเขียนการทำงานของฟังก์ชันทั้ง 4 ฟังก์ชัน ตอนนี้ Server สามารถสื่อสารกับ Client ได้แล้ว โดยคำสั่งที่ใช้ในการรับ - ส่งข้อมูล มีความแตกต่างกันเมื่อโปรโตคอลที่ใช้มีความแตกต่างกัน

สำหรับโปรโตคอล TCP คำสั่งที่ใช้ส่งข้อมูล คือ s.send(data) โดยตัวแปร data คือ ข้อมูลที่ต้องการส่งนั้นจะเป็นตัวแปรชนิดสตริงส์ ส่วนคำสั่งที่ใช้ในการรับข้อมูล คือ s.recv(byte) โดยตัวแปร byte คือ จำนวน Byte ของข้อมูลที่สามารถรับได้มากที่สุด และตัวแปร s คือ ค่าของ Socket ที่ได้จากคำสั่ง s.accept() ตัวอย่างการเขียนแสดงดังรูปที่ 2.27

```
snew , addr = s.accept()
#display client information
print 'Connected with ' + addr[0] + ':' + str(addr[1])
# talking with the client
data = snew.recv(1024)
reply = 'OK... ' + data
 snew.send(reply)
print 'Data from client ' ,data
```

รูปที่ 2.27 การรับ - ส่งข้อมูลระหว่าง TCP Server and client

2.7.5.4 การเขียน Client socket สำหรับโปรโตคอล TCP

การเขียน Client socket สำหรับโปรโตคอล TCP ผู้เขียนต้องทำความเข้าใจฟังก์ชันการทำงานซึ่งอยู่ในโมดูล socket ซึ่งฟังก์ชันที่ใช้ในการเขียนมี 3 ฟังก์ชันดังนี้

1) การสร้าง TCP Socket
2) สร้างการเชื่อมต่อกับ Server
3) รับ - ส่งข้อมูลระหว่าง Client กับ Server

การทำงานของฟังก์ชันในส่วนการสร้าง Socket และในส่วนรับ-ส่งข้อมูลระหว่าง Client กับ server เหมือนกับที่ได้อธิบายไว้ในหัวข้อการเขียน Server socket ส่วนฟังก์ชันการสร้างการเชื่อมต่อกับ Server เป็นฟังก์ชันที่ใช้งานเฉพาะในการเขียน Client socket

การสร้างการเชื่อมต่อกับ Server ใช้คำสั่ง s.connect((remoteIP, Port)) โดยตัวแปร remoteIP คือ หมายเลขไอพีของ Server ที่ต้องการติดต่อ และตัวแปร port คือ พอร์ตที่ใช้ในการรับ - ส่ง ซึ่งต้องกำหนดให้ตรงกับฝั่ง Server ตัวอย่างการเขียนแสดงดังรูปที่ 2.28

```
#Socket client in python
Import socket   #for sockets
import sys  #for exit
try :
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error :
        print 'Failed to create socket'
        sys.exit()
print 'Socket Created'

remote_ip = ' ' #Server IP
port = 9000
#Connect to remote server
s.connect((remote_ip , port))
print 'Socket Connected to ' + host + ' on ip ' + remote_ip
```

รูปที่ 2.28 การสร้างการเชื่อมต่อกับ TCP Server

2.7.5.5 การเขียน Server socket สำหรับโปรโตคอล UDP

การเขียน Server socket สำหรับโปรโตคอล UDP ผู้เขียนต้องทำความเข้าใจฟังก์ชันการทำงานซึ่งอยู่ในโมดูล socket ซึ่งฟังก์ชันที่ใช้ในการเขียนมี 3 ฟังก์ชันดังนี้

1) การสร้าง UDP Socket ซึ่งได้อธิบายไปแล้วข้างต้น
2) การเชื่อม Socket เข้ากับหมายเลขไอพีและพอร์ตของ Server
3) รับ - ส่งข้อมูลระหว่าง Server กับ Client

การเชื่อม Socket เข้ากับหมายเลขไอพีและพอร์ตทำเช่นเดียวกับการเขียน TCP Server socket โดยใช้คำสั่ง s.bind((IPaddress, Port)) โดยตัวแปร s คือ Socket ที่สร้างไว้ ตัวแปร IPaddress เป็นหมายเลขไอพีของคอมพิวเตอร์ที่ใช้งานเป็น Server และกำหนดแสดงดังรูปที่ 2.29

```
IPaddress = ' ' #Symbolic name meaning all available interfaces
Port = 9000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((IPaddress, Port))
except socket.error , msg:
    print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
    sys.exit()
print 'Socket bind complete'
```

รูปที่ 2.29 การเชื่อม Socket เข้ากับ IPaddress และพอร์ตของ UDP Server

สำหรับโปรโตคอล UDP ไม่ต้องทำการสร้างเส้นทางการติดต่อสื่อสารระหว่าง Server และ Client (Connectionless) คำสั่งที่ใช้ส่งข้อมูล คือ s.sendto(data ,(host , port)) โดยตัวแปร data คือ ข้อมูลที่ต้องการส่งเป็นตัวแปรชนิดสตริงส์, ตัวแปร host คือ หมายเลขไอพีของ UDP Server และตัวแปร port คือ หมายเลขพอร์ตที่ UDP Server ใช้รับส่งข้อมูล ส่วนคำสั่งที่ใช้รับข้อมูล คือ s.recvfrom(byte) โดยตัวแปร byte คือ จำนวน Byte ของข้อมูลที่สามารถรับได้มากที่สุด โดยข้อมูลที่รับได้ประกอบไปด้วยข้อมูล 3 ส่วน ได้แก่ ข้อมูล(ตัวแปร data), หมายเลขไอพีของ Client และหมายเลขพอร์ตสำหรับการรับส่งข้อมูลของ Client ตัวอย่างการเขียนแสดงดังรูปที่ 2.30

2.7.5.6 การเขียน Client socket สำหรับโปรโตคอล TCP

การเขียน Client socket สำหรับโปรโตคอล UDP ผู้เขียนต้องทำความเข้าใจฟังก์ชันการทำงานซึ่งอยู่ในโมดูล Socket ซึ่งฟังก์ชันที่ใช้ในการเขียนมี 2 ฟังก์ชันดังนี้

1) การสร้าง UDP Socket
2) รับ-ส่งข้อมูลระหว่าง Client กับ UDP Server

การทำงานในส่วนการสร้างได้อธิบายไว้แล้วข้างต้นในหัวข้อการสร้าง Socket โดยใช้โปรโตคอล UDP และการรับ-ส่งข้อมูลระหว่าง client กับ UDP Server ได้อธิบายไว้ในหัวข้อการเขียน Server socket สำหรับโปรโตคอล UDP

```
# talking with the client
d = s.recvfrom(1024) #receive data packet
data = d[0]
client_ip = d[1]
client_port = d[2]
reply = 'OK... ' + data
print 'Data from client ' ,data
# send data packet
s.sendto = (reply, (client_ip, client_port))
```

รูปที่ 2.30 การรับ - ส่งข้อมูลระหว่าง UDP Server and client

2.7.5.7 การปิด socket

คำสั่งที่ใช้ในการปิด Socket แสดงดังรูปที่ 2.31

```
#Socket client in python
s.close
```

รูปที่ 2.31 การปิด Socket

2.8 หลักการทั่วไปของ Cryptography [12]

2.8.1 ระบบรหัสลับ

วัตถุประสงค์หลักของวิทยาการรหัสลับคือ การที่ทำให้สามารถติดต่อสื่อสารระหว่างกันผ่านช่องสัญญาณที่มีความไม่ปลอดภัยจากการดักฟังของบุคคลที่สาม ซึ่งไม่อาจทราบถึงข้อความที่สื่อสารกัน โดยองค์ประกอบของระบบรหัสลับ สามารถแบ่งออกได้เป็น 5 ส่วนดังนี้

1. ข้อความต้นฉบับ (Plaintext) คือข้อความหรือข้อมูลต้นฉบับที่ต้องการจะนำไปผ่านการเข้ารหัสลับ ซึ่งอาจจะบรรจุอยู่ในไฟล์หรือไฟล์โปรแกรม

2. อัลกอริทึมการเข้ารหัสลับ (Encryption algorithm) คือกระบวนการหรือขั้นตอนที่ใช้ในการเปลี่ยนแปลงข้อความต้นฉบับให้อยู่ในรูปแบบที่เปลี่ยนไปจากเดิม และผู้อื่นไม่สามารถเข้าใจได้ เช่น การแทนที่อักขระในข้อความต้นฉบับด้วยอักขระที่ต่างจากไปจากเดิมหรือการสลับตำแหน่งอักขระในข้อความต้นฉบับ เป็นต้น

3. กุญแจลับ (Key) คือองค์ประกอบหนึ่งที่ป้อนให้กับอัลกอริทึมเข้ารหัสลับเพื่อใช้ในการเข้ารหัสลับข้อความต้นฉบับ โดยผลลัพท์ที่ได้จะแตกต่างกันออกไปตามกุญแจลับที่เลือกใช้ ทั้งนี้กุญแจลับเป็นค่าที่มีความเป็นอิสระไม่ขึ้นกับข้อความต้นฉบับ

4. ข้อความไซเฟอร์ (Cilpertext) เป็นผลลัพท์ที่ได้จากการนำข้อความต้นฉบับไปผ่านกระบวนการเข้ารหัสลับโดยใช้กุญแจลับตามที่ผู้ใช้กำหนด ข้อความไซเฟอร์ที่ได้จะเป็นข้อความที่ผู้อื่นไม่สามารถอ่านเข้าใจได้

5.อัลกอริทึมการถอดรหัสลับ (Decryption algorithm) คือ กระบวนการหรือขั้นตอนในการแปลงจากข้อความไซเฟอร์ให้กลับเป็นข้อความต้นฉบับตามเดิมโดยอาศัยกุญแจลับดอกเดียวกับที่ใช้ในขั้นตอนการเข้ารหัสลับ

2.8.2 ประเภทของการเข้ารหัสลับ

การเข้ารหัสลับแบ่งออกเป็น 2 ประเภท ได้แก่ การเข้ารหัสลับแบบซิมเมทริค (Symmetric cryptography) และแบบอะซิมเมทริค (Asymmetric cryptography)

1. การเข้ารหัสลับแบบซิมเมทริค คือ คีย์ในการเข้ารหัสลับและถอดรหัสลับเหมือนกัน ตัวอย่างของการเข้ารหัสลับแบบนี้ คือ DES, Double DES, Triple DES, AES, RC6

2. การเข้ารหัสลับแบบอะซิมเมทริค คือ คีย์ในการเข้ารหัสลับและถอดรหัสลับไม่เหมือนกัน สามารถแบ่งได้เป็น 2 ประเภท คือ Private key และ Public key โดยตัวอย่างของการเข้ารหัสลับแบบนี้คือ RSA

หมายเหตุ: การเข้ารหัสลับแบบเคออติกจัดอยู่ในประเภท Symmetric cryptography

2.8.3 การโจมตีระบบรหัสลับ

เมื่อผู้ทำการโจมตีระบบรหัสลับต้องการทราบข้อความที่เป็นความลับ หรือกุญแจ (Key) ที่ใช้ในระบบรหัสลับของการสื่อสารที่ทำการโจมตี สามารถทำได้ 2 วิธี คือ

1. การวิเคราะห์รหัสลับ (Cryptanalysis) คือการที่ฝ่ายตรงข้ามต้องการทราบข้อความต้นฉบับดั้งเดิมหรือกุญแจลับ โดยอาศัยการวิเคราะห์ข้อความไซเฟอร์ หรืออาศัยการสังเกตุความสัมพันธ์ระหว่างข้อความต้นฉบับและข้อความไซเฟอร์ที่มีอยู่ เพื่อให้ได้ข้อความต้นฉบับหรือกุญแจที่ต้องการ

2. การโจมตีแบบตะลุย (Brute-force attack) เป็นการโจมตีที่ผู้โจมตีเดากุญแจทุกๆ ค่าที่สามารถเป็นไปได้ แล้วนำกุญแจแต่ละค่าถอดรหัสข้อความไซเฟอร์ว่าสามารถแปลงกลับมาเป็นข้อความต้นฉบับที่อ่านเข้าใจได้หรือไม่ โดยเฉลี่ยแล้วต้องใช้ครึ่งหนึ่งของจำนวนกุญแจทั้งหมดในการหาข้อความต้นฉบับ

2.8.4 การวิเคราะห์รหัสลับ

โดยทั่วไปในการศึกษาถึงการวิเคราะห์รหัสลับ (Cryptanalysis) มักจะมีการกำหนดข้อสมมติฐานพื้นฐานว่า ผู้ทำการโจมตีระบบรหัสลับทราบถึงโครงสร้างของระบบรหัสลับที่ใช้งานอยู่เป็นอย่างดี แน่นอนว่าหากผู้ทำการโจมตีระบบรหัสลับไม่ทราบถึงระบบรหัสลับที่ใช้อยู่ การเข้าโจมตีจะยิ่งยากลำบากยิ่งขึ้น อย่างไรก็ดี ผู้ออกแบบระบบรหัสลับจะหลีกเลี่ยงการตั้งข้อสมมุติว่าผู้ทำการโจมตีระบบรหัสลับไม่อาจทราบถึงโครงสร้างของระบบรหัสลับที่ใช้ เนื่องจากการเก็บรักษาความลับดังกล่าวเป็นที่ยากต่อการควบคุมดังนั้น การออกแบบระบบรหัสลับโดยทั่วไปจึงมักอยู่ภายใต้ข้อสมมติฐานว่าฝ่ายตรงข้ามสามารถทราบถึงระบบรหัสลับที่ใช้ โดยข้อสมมุติฐานดังกล่าวนี้รู้จักกันทั่วไปในชื่อ Kerckhoff's principle

การวิเคราะห์รหัสลับสามารถแบ่งออกได้เป็นหลายระดับตามขีดความสามารถของผู้ทำการโจมตีระบบรหัสลับที่เข้าโจมตี สามารถแบ่งออกได้เป็นระดับต่างๆ ดังนี้คือ

- ข้อความไซเฟอร์อย่างเดียว (Ciphertext-only) เป็นกรณีที่ฝ่ายตรงข้ามสามารถเข้าถึงข้อความไซเฟอร์ y และต้องการทราบข้อความต้นฉบับหรือกุญแจ วิธีนี้เป็นวิธีที่ฝ่ายตรงข้ามมีข้อมูลน้อยที่สุด
- ทราบข้อความต้นฉบับ (Known plaintext) ฝ่ายตรงข้ามสามารถเข้าถึงข้อความต้นฉบับ x และข้อความไซเฟอร์ y ที่เป็นคู่กัน หรือฝ่ายตรงข้ามทราบรูปแบบของข้อความต้นฉบับที่ปรากฏในข้อความ ตัวอย่างเช่น แฟ้มภาษาโพสต์สคริปต์มีรูปแบบมาตรฐานที่ต้นแฟ้มเสมอ หรือเอกสารทางธุรกิจมักมีข้อความมาตรฐานที่ปรากฏในตำแหน่งที่แน่นอนภายในแฟ้ม ดังนั้น เมื่อฝ่ายตรงข้ามมีข้อความต้นฉบับและข้อความไซเฟอร์ที่คู่กัน เขาสามารถวิเคราะห์หาความสัมพันธ์ระหว่างข้อความต้นฉบับและข้อความไซเฟอร์เพื่อหากุญแจได้
- เลือกข้อความต้นฉบับ (Chosen plaintext) ฝ่ายตรงข้ามสามารถเข้าถึงและใช้เครื่องเข้ารหัสลับได้ชั่วคราว ดังนั้น เขาสามารถที่จะเลือกข้อความต้นฉบับ x ได้เองตามปรารถนา และได้เป็นข้อความไซเฟอร์ y ที่เป็นคู่กัน วิธีนี้ทำให้ฝ่ายตรงข้ามสามารถเลือกรูปแบบของข้อความที่ต้องการเข้ารหัสลับ เพื่อศึกษารูปแบบของข้อความไซเฟอร์ที่สัมพันธ์กัน ในการวิเคราะห์หาค่ากุญแจต่อไป
- เลือกข้อความไซเฟอร์ (Chosen ciphertext) วิธีนี้เป็นวิธีที่ตรงข้ามกับวิธีเลือกข้อความต้นฉบับ กล่าวคือ ฝ่ายตรงข้ามสามารถเข้าถึงและใช้เครื่องถอดรหัสลับได้ชั่วคราว

ดังนั้น ฝ่ายตรงข้ามจะเลือกข้อความไซเฟอร์ y ได้เองตามปรารถนา และ ถอดรหัสลับให้ได้ข้อความต้นฉบับ x ที่คู่กัน

● การวิเคราะห์หาส่วนแตกต่าง (Differential cryptanalysis) เป็นการโจมตีประเภทเลือกข้อความต้นฉบับอีกวิธีหนึ่ง โดยฝ่ายตรงข้ามเข้ารหัสลับข้อความต้นฉบับหลายๆ ฉบับที่แตกต่างกันเล็กน้อยแล้วเปรียบเทียบข้อความไซเฟอร์ที่ได้

จากที่กล่าวจะเห็นว่าระดับความรุนแรงในการเข้าโจมตีแต่ละแบบนั้นแตกต่างกันไปตามความสามารถในการเข้าถึงระบบรหัสลับ โดยทั่วไปแล้วอัลกอริทึมที่ไร้ประสิทธิภาพเท่านั้นที่ไม่สามารถต้านทานการโจมตีแบบข้อความไซเฟอร์อย่างเดียว ปัจจุบันอัลกอริทึมส่วนใหญ่ถูกออกแบบมาเพื่อให้สามารถต้านทานต่อการโจมตีแบบทราบข้อความต้นฉบับ

การโจมตีประเภทการวิเคราะห์รหัสลับนี้ผู้โจมตีต้องมีความรู้ด้านคณิตศาสตร์และอัลกอริทึมอย่างดีเพื่อใช้ในการวิเคราะห์ข้อความไซเฟอร์ดังนั้นนักโจมตีระบบรหัสลับด้วยวิธีนี้จึงเรียกว่า นักวิเคราะห์รหัสลับ (Cryptanalyst)

ในงานด้านวิทยาการรหัสลับนี้มีบุคคล 2 ประเภทที่ทำงานตรงข้ามกัน ประเภทแรกเรียกว่า นักสร้างรหัสลับ (Cryptographer) ซึ่งเป็นผู้ออกแบบอัลกอริทึมในการเข้ารหัสลับและถอดรหัสลับ บุคคลอีกประเภทคือ นักวิเคราะห์รหัสลับ ดังที่กล่าวมาแล้วซึ่งมีหน้าที่หาจุดอ่อนในอัลกอริทึมเข้ารหัสลับและถอดรหัสลับ

นักวิเคราะห์รหัสลับอาจเป็นฝ่ายตรงกันข้ามหรือเป็นฝ่ายเดียวกับนักสร้างรหัสลับก็ได้ ไม่จำเป็นต้องเป็นฝ่ายตรงข้ามเสมอไป เช่น บริษัทหรือหน่วยงานวิจัยที่ต้องการพัฒนาอัลกอริทึมเข้ารหัสลับและถอดรหัสลับนั้น ต้องอาศัยนักสร้างรหัสลับในการออกแบบอัลกอริทึมเข้ารหัสลับและอาศัยนักวิเคราะห์รหัสลับในการพิสูจน์หรือตรวจสอบว่าอัลกอริทึมที่นักสร้างรหัสลับออกแบบมานั้นมีความปลอดภัยเพียงพอหรือไม่

2.8.5 การโจมตีแบบตะลุย (Brute - force)

การโจมตีแบบตะลุยเป็นการโจมตีที่ผู้โจมตีเดาค่ากุญแจทุกค่าที่สามารถเป็นไปได้และนำค่ากุญแจเหล่านั้นมาถอดรหัสลับข้อความไซเฟอร์ที่มีอยู่จนกว่าจะได้ข้อความต้นฉบับที่ต้องการ ซึ่งโดยเฉลี่ยแล้วต้องใช้ครึ่งหนึ่งของกุญแจทั้งหมดที่มีอยู่เพื่อใช้ในการถอดรหัสลับให้ได้ข้อความต้นฉบับ การโจมตีแบบนี้เป็นการโจมตีที่ไม่สามารถป้องกันได้ ทั้งนี้เนื่องจากผู้โจมตีเพียงแค่เอากุญแจแต่ละค่ามาถอดรหัสลับข้อความไซเฟอร์ตรงๆนั่นเองอย่างไรก็ตาม เรามีวิธีที่สามารถทำให้ผู้โจมตีต้องประสบกับความลำบากหรือประสบความยุ่งยากในการโจมตีได้

จำนวนกุญแจทั้งหมดที่เป็นไปได้ จะขึ้นอยู่กับขนาดของกุญแจที่กำหนดไว้ในอัลกอริทึม เช่น กุญแจขนาด 2 บิต มีค่าที่เป็นไปได้คือ 00 01 10 11 ดังนั้น จำนวนกุญแจทั้งหมดมี 4 ค่า หรือ 2^2 ดังนั้นถ้ากุญแจมีขนาด n บิต จำนวนกุญแจทั้งหมดที่สามารถเป็นไปได้คือ 2^n

ดังนั้นถ้ากุญแจในระบบรหัสลับของเรามีจำนวนบิตสูงจะมีผลทำให้จำนวนกุญแจทั้งหมดที่สามารถเป็นไปได้มีค่าสูงตามไปด้วย ตารางที่ 2.4 แสดงจำนวนกุญแจทั้งหมดของขนาดกุญแจต่างๆ กัน และเวลาที่ใช้ในการถอดรหัสลับของกุญแจแต่ละขนาด

ตารางที่ 2.4 จำนวนกุญแจทั้งหมดที่เป็นไปได้ของขนาดกุญแจต่างๆ

ขนาดกุญแจ (บิต)	จำนวนกุญแจทั้งหมด	เวลาที่ใช้ในการถอดรหัสลับ (1 ล้านครั้งต่อวินาที)	เวลาที่ใช้ในการถอดรหัสลับ (1 ล้านล้านครั้งต่อวินาที)
32	$2^{32} = 4.3\times10^{9}$	35.8 นาที	1.25×10^{-3} วินาที
56	$2^{56} = 7.2\times10^{16}$	1142 ปี	10.01 ชั่วโมง
112	$2^{112} = 5.2\times10^{33}$	8.2×10^{19} ปี	8.2×10^{13} ปี
128	$2^{128} = 3.4\times10^{38}$	5.4×10^{24} ปี	5.4×10^{18} ปี
168	$2^{168} = 3.7\times10^{50}$	5.9×10^{36} ปี	5.9×10^{30} ปี
256	$2^{256} = 1.1\times10^{77}$	1.8×10^{63} ปี	1.8×10^{57} ปี

จากตารางที่ 2.4 จะเห็นได้ว่าถ้ากุญแจมีขนาดเล็ก เช่น ขนาด 32 บิต หรือ 56 บิต สามารถนำกุญแจทั้งหมดมาถอดรหัสลับข้อความไซเฟอร์ออกมาได้โดยใช้เวลาไม่นานนักด้วยความสามารถของคอมพิวเตอร์ในปัจจุบัน

แต่ถ้ากุญแจมีขนาดเพิ่มขึ้นเป็น 112 บิต 128 บิต 168 บิต หรือ 256 บิต จะต้องใช้เวลาเพิ่มขึ้น ดังนั้นอัลกอริทึมเข้ารหัสลับที่จัดว่ามีความปลอดภัยในปัจจุบันต้องมีขนาดของกุญแจอย่างน้อย 128 บิต จึงสามารถต้านทานการโจมตีแบบตะลุยได้

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

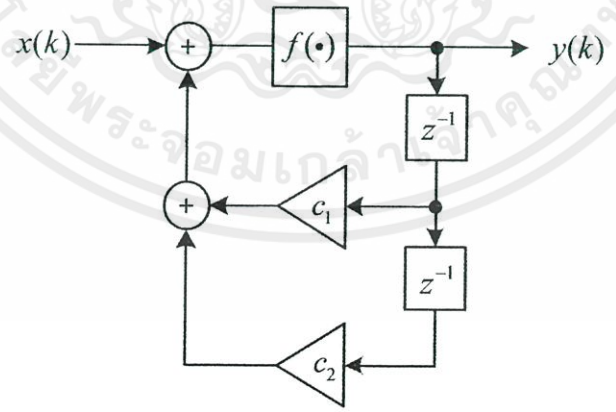
ในส่วนของบทนี้เป็นการนำพฤติกรรมเคออสที่เกิดขึ้นในวงจรกรองสัญญาณเชิงเลข รวมถึงหลักการและทฤษฎีต่าง ๆ ที่เกี่ยวข้องมาจำลองการทำงานของวงจรเข้ารหัสลับ – ถอดรหัสลับแบบเคออติกโดยใช้โปรแกรม MATLAB แล้วนำวงจรที่ได้มาสร้างลงบนอุปกรณ์ Raspberry Pi โดยใช้ภาษาไพธอนอธิบายการทำงานทั้งหมด และทดลองการเข้ารหัส - ถอดรหัสลับด้วยการสื่อสารข้อมูลแบบอนุกรม เพื่อแสดงพฤติกรรมของปรากฏการณ์เคออสที่เกิดขึ้นจริงในวงจรกรองสัญญาณเชิงเลข จากนั้นจึงสร้างระบบรหัสลับแบบเคออติกสำหรับการสื่อสารข้อมูลผ่านเครือข่ายลงบนอุปกรณ์ Raspberry Pi ออกแบบหน้าต่างโปรแกรมให้ผู้ใช้งานสามารถใช้งานได้อย่างสะดวกมากขึ้น และพัฒนาระบบรหัสลับให้สามารถรับส่งข้อมูลชนิดภาพ และไฟล์เสียงได้ สุดท้ายจึงออกแบบการโจมตีแบบตลุยเพื่อทดสอบความปลอดภัยของระบบ

3.1 การออกแบบ

3.1.1 การออกแบบการทดลองโดยใช้โปรแกรม MATLAB

3.1.1.1 วงจรเข้ารหัสลับแบบเคออติก

ออกแบบโดยใช้วงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด (Infinite Impulse Response second order filter: IIR 2^{nd} order filter) โดยวงจรกรองมีโครงสร้างแบบโดยตรง (Direct form) ดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างของวงจรเข้ารหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด

หากยังไม่พิจารณาถึงฟังก์ชัน $f(\bullet)$ สามารถเขียนสมการผลต่างสืบเนื่อง (Difference equation) ในรูปสมการป้อนกลับ (Recursive equation) ดังสมการที่ 3.1

$$y(k) = x(k) + c_1 y(k-1) + c_2 y(k-2) \tag{3.1}$$

จากสมการที่ (3.1) สามารถเขียนเป็นฟังก์ชันถ่ายโอน (Transfer function) ดังสมการ (3.2)

$$Y(z) = X(z) + c_1 Y(z) z^{-1} + c_2 Y(z) z^{-2}$$

$$Y(z)(1 - c_1 z^{-1} - c_2 z^{-2}) = X(Z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{(1 - c_1 z^{-1} - c_2 z^{-2})} \tag{3.2}$$

จากสมการที่ (3.2) จะพบว่ามีตำแหน่งของโพล (Pole) อยู่ 2 ตัว ซึ่งหาได้จาก

$$1 - c_1 z^{-1} - c_2 z^{-2} = 0$$

$$z^2 - c_1 z - c_2 = 0$$

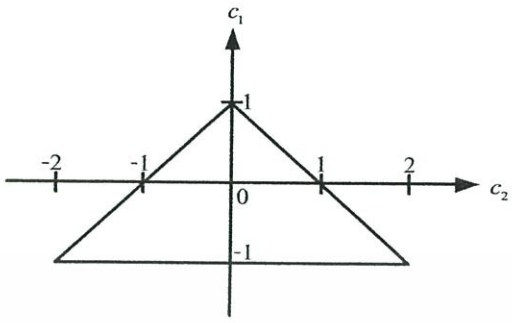
ดังนั้น

$$p_1 = \frac{c_1 + \sqrt{c_1^2 + 4c_2}}{2} \tag{3.3}$$

และ

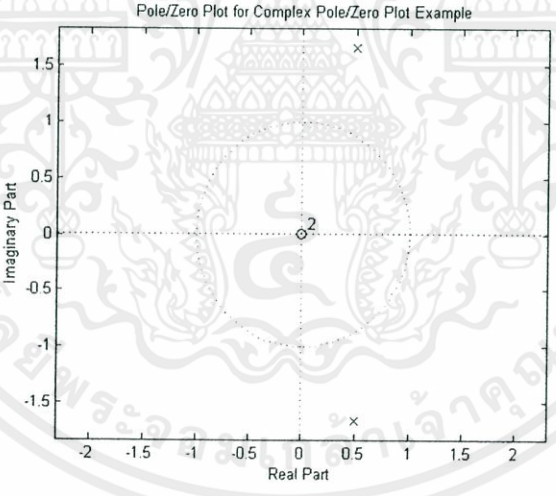
$$p_2 = \frac{c_1 - \sqrt{c_1^2 + 4c_2}}{2} \tag{3.4}$$

เพื่อให้ระบบไม่มีเสถียรภาพ จะต้องมีค่าสัมประสิทธ์ของวงจรกรองสัญญาณเชิงเลขอย่างน้อย 1 ตัว อยู่ภายนอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ ดังรูปที่ 3.2



รูปที่ 3.2 ขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ

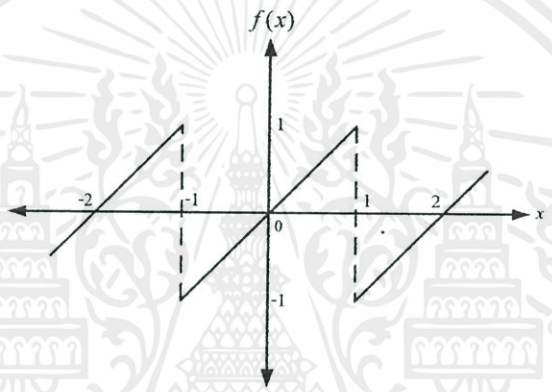
หากเลือกค่าสัมประสิทธ์อย่างน้อย 1 ตัวให้อยู่ภายนอกพื้นที่เสถียรภาพแล้ว จะพบว่าค่าโพลของฟังก์ชันถ่ายโอนจะมีค่ามากกว่าหนึ่ง ซึ่งทำให้อยู่ภายนอกพื้นที่วงกลมหนึ่งหน่วย แสดงให้เห็นว่าวงจรกรองสัญญาณเชิงเลขนี้ไม่มีเสถียรภาพ เช่น $c_1 = 1$ และ $c_2 = -3$ โดยตำแหน่งของโพลของวงจรเข้ารหัสลับแสดงดังรูปที่ 3.3



รูปที่ 3.3 ตำแหน่งโพลของวงจรเข้ารหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสอง ชนิดอิมพัลส์ไม่จำกัด

$$p_1 = \frac{1+\sqrt{1^2-12}}{2} \approx 0.5 + j1.66 \text{ และ } p_2 = \frac{1-\sqrt{1^2-12}}{2} \approx 0.5 - j1.66$$

เมื่อนำระบบเข้ารหัสลับดังกล่าวไปสร้างเป็นอุปกรณ์บนฮาร์ดแวร์ จำเป็นจะต้องกำหนดความยาวของข้อมูลให้มีความยาวที่จำกัด (Finite word - length) ซึ่งทำให้เกิดการล้นขึ้น ดังนั้นในการจำลองการทำงานจึงต้องอาศัยฟังก์ชัน $f(x)$ เพื่อจำลองการล้นของข้อมูล และเรียกฟังก์ชัน $f(x)$ นี้ว่า ฟังก์ชันมอดูโล (Modulo function) โดยฟังชันมอดูโลนี้จะมีคุณลักษณะ (Characteristic) ดังรูปที่ 3.4



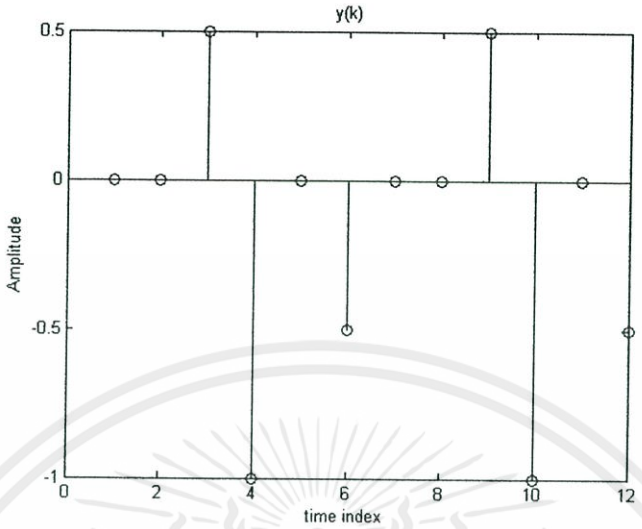
รูปที่ 3.4 คุณลักษณะของฟังก์ชัน $f(x) = [(x+1) \bmod 2] - 1$

ฟังก์ชันมอดูโลจะมีสมการคือ $f(x) = [(x+1) \bmod 2] - 1$ ดังนั้น หลังจากการผ่านฟังก์ชันมอดูโลแล้วค่า $y(k)$ จะเกิดการเปลี่ยนแปลงตามสมการของฟังก์ชัน $f(x)$ ยกตัวอย่างเช่นที่ $y(4) = 1$ เมื่อผ่านฟังก์ชันมอดูโลแล้วจะได้ค่าใหม่คือ $\hat{y}(4)$

$$\hat{y}(k) = \left[\left(y(k)+1\right) \bmod 2\right] - 1$$

$$\hat{y}(4) = [(1+1) \bmod 2 - 1] = [2 \bmod 2] - 1 = 0 - 1 = -1$$

ตรวจสอบผลการทำงานของวงจรเมื่อมีฟังก์ชัน $f(x)$ โดยกำหนด $c_1 = 1$ และ $c_2 = -3$ แล้วให้ $x(k) = 0.5$ กำหนดเงื่อนไขเริ่มต้นคือ $y(1) = 0$, $y(2) = 0$ โดยกำหนดให้ $k = 3:12$ จะได้ผลดังรูปที่ 3.5



รูปที่ 3.5 ค่าเอาต์พุตจากการจำลองการทำงานของวงจรเข้ารหัสลับวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัดแบบ เมื่อใช้ฟังก์ชัน $f(x)$

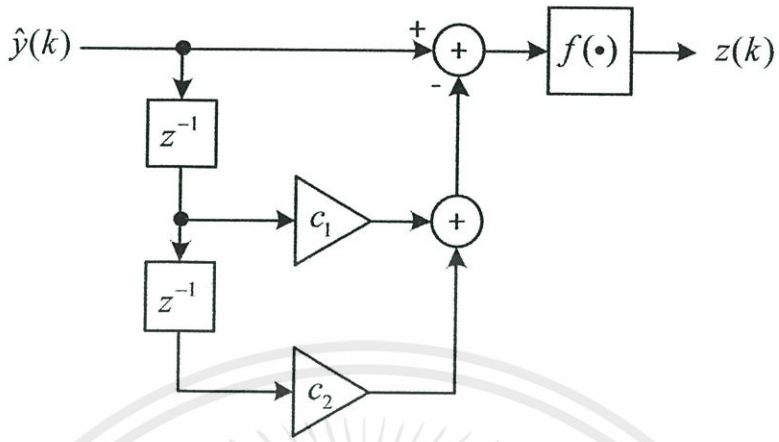
โดยตรวจสอบค่าจากการรันโปรแกรมได้ดังนี้

$$\hat{y}(k) = 0, 0, 0.5, -1, 0, -0.5, 0, 0, 0.5, -1, 0, -0.5$$

เมื่อพิจารณาจะพบว่าค่าที่ได้จากการคำนวณจะมีค่าที่ตรงกับค่าในรูปที่ 3.5 ดังนั้นจึงสามารถบอกได้ว่า เมื่อกำหนดให้ค่าสัมประสิทธ์อย่างน้อยหนึ่งตัวให้มีค่าอยู่นอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพแล้วจะทำให้วงจรเข้ารหัสลับซึ่งเป็นวงจรกรองแบบอิมพัลส์ไม่จำกัดเกิดตำแหน่งของโพลอยู่ภายนอกขอบเขตพื้นที่วงกลมหนึ่งหน่วย (Unit circle) ซึ่งทำให้วงจรไม่มีความเสถียรภาพทำให้ค่าที่ได้จากวงจรจะมีค่าสลับไปมา จากนั้นใช้ฟังก์ชัน $f(x)$ เพื่อทำให้เกิดการล้น

3.1.1.2 วงจรถอดรหัสลับแบบเคออติก

ออกแบบโดยใช้วงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด (Finite Impulse Response second order filter) วงจรถอดรหัสลับจะมีโครงสร้างที่เป็นส่วนกลับ (Inverse) ของวงจรเข้ารหัสลับ ซึ่งวงจรถอดรหัสเป็น FIR 2^{nd} order filter ดังรูปที่ 3.6



รูปที่ 3.6 โครงสร้างของวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสอง ชนิดอิมพัลส์จำกัด

จากรูปที่ 3.6 เมื่อยังไม่พิจารณาฟังก์ชัน $f(x)$ เขียนสมการผลต่างสืบเนื่องได้ดังสมการที่ (3.5)

$$z(k) = \hat{y}(k) - c_1\hat{y}(k-1) - c_2\hat{y}(k-2) \tag{3.5}$$

จากสมการที่ (3.5) เขียนเป็นฟังก์ชันถ่ายโอน (Transfer function) ได้ดังสมการที่ (3.6)

$$Z(z) = \hat{y}(z) - c_1\hat{y}(z)z^{-1} - c_2\hat{y}(z)z^{-2}$$

$$H(z) = \frac{z(z)}{\hat{y}(z)} = 1 - c_1z^{-1} - c_2z^{-2} \tag{3.6}$$

จากสมการที่ (3.5) จะเห็นได้ว่ามีโครงสร้างแบบไม่ป้อนกลับ (Non - recursive) ซึ่งเป็นลักษณะทั่วไปของวงจรกรองสัญญาณเชิงเลขชนิดอิมพัลส์จำกัด และจากสมการที่ (3.6) ฟังก์ชันถ่ายโอนของวงจรถอดรหัสลับจะมีตำแหน่งของซีโร่ (Zero) ดังต่อไปนี้

$$1 - c_1z^{-1} - c_2z^{-2} = 0$$

$$z^2 - c_1 z - c_2 = 0$$

ดังนั้น

$$z_1 = \frac{c_1 + \sqrt{c_1^2 + 4c_2}}{2} \tag{3.7}$$

และ

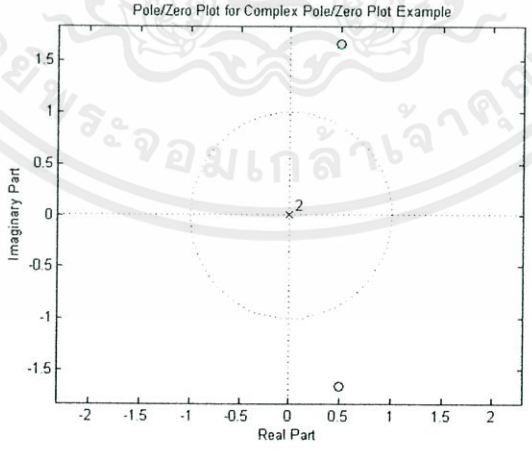
$$z_2 = \frac{c_1 - \sqrt{c_1^2 + 4c_2}}{2} \tag{3.8}$$

ค่าสัมประสิทธิ์ของวงจรถอดรหัสลับจะต้องใช้ค่าเดียวกับวงจรเข้ารหัสลับดังนั้นจึงกำหนดค่าสัมประสิทธิ์ให้เหมือนกับตัวอย่างก่อนหน้านี้คือให้ค่า $c_1 = 1$ และ $c_2 = -3$ จะได้

$$z_1 = \frac{1 + \sqrt{1^2 - 12}}{2} \approx 0.5 + j1.66$$

$$z_2 = \frac{1 - \sqrt{1^2 - 12}}{2} \approx 0.5 - j1.66$$

แสดงตำแหน่งของซีโร่ดังรูปที่ 3.7



รูปที่ 3.7 ตำแหน่งของซีโร่ของวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด

กรณี $c_1 = 1$ และ $c_2 = -3$ เมื่อพิจารณาตำแหน่งของซีโร่จากรูปที่ 3.7 แล้วนำไปเปรียบเทียบกับตำแหน่งของโพลในรูปที่ 3.3 จะพบว่าค่าโพลของวงจรเข้ารหัสลับกับค่าซีโร่ของวงถอดรหัสลับ นั้นจะหักล้างกัน คือ ตำแหน่งโพลของวงจรเข้ารหัสลับจะมีค่าเท่ากับตำแหน่งซีโร่ของวงจรถอดรหัสลับ ดังนั้นจึงทำให้ตำแหน่งของโพลและซีโร่ของวงจรทั้งสองหักล้างกัน จึงส่งผลให้ในการเข้ารหัสลับและถอดรหัสลับจะต้องมีค่าสัมประสิทธิ์ที่เหมือนกันทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับ จากสมการที่ (3.5) โดยจำลองการทำงานดังนี้

1) เพิ่มฟังก์ชัน $f(\bullet)$ เข้าไป แล้วหาค่า $\hat{z}(k)$ ได้ดังต่อไปนี้

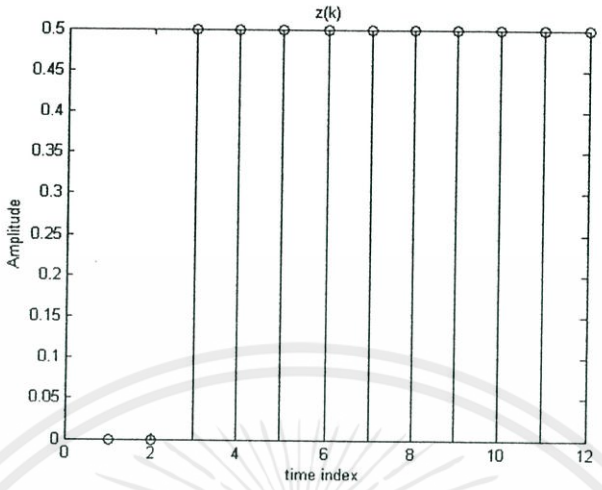
$$\begin{aligned}\hat{z}(3) &= f\{\hat{y}(3) - \hat{y}(2) - (-3)\hat{y}(1)\} \\ &= f\{0.5 - 0 - (-3)(0)\} \\ &= (0.5+1) \bmod 2 - 1 = 1.5 - 1 \\ \hat{z}(3) &= 0.5\end{aligned}$$

$$\begin{aligned}\hat{z}(4) &= f\{\hat{y}(4) - \hat{y}(3) - (-3)\hat{y}(2)\} \\ &= f\{(-1) - 0.5 - (-3)(0)\} \\ &= (-1.5+1) \bmod 2 - 1 = 1.5 - 1 \\ \hat{z}(4) &= 0.5\end{aligned}$$

และเมื่อตรวจสอบจากผลการรันโดยใช้โปรแกรมจะได้ดังนี้

$$\hat{z}(k) = 0, 0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5$$

สังเกตค่าของ $\hat{z}(k)$ ที่ตำแหน่งต่างๆ จากการคำนวณจะมีค่าเท่ากับค่าที่ได้จากการรันโปรแกรม ดังรูปที่ 3.8



รูปที่ 3.8 ค่าเอาต์พุตของวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสอง ชนิดอิมพัลส์จำกัด กรณีผ่านฟังก์ชัน $f(x)$

จะเห็นได้ว่าค่าเอาต์พุตของวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด : $\hat{z}(k)$ เท่ากับค่าอินพุตของวงจรเข้ารหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด : $x(k)$

หมายเหตุ: สำหรับการหารเอาเศษหรือที่เรียกว่า Modulo operation นั้น ในส่วนของโปรแกรม MATLAB จะเขียนแทนด้วย $\mathrm{mod}(x, y)$ คือ นำค่า x มา mod ด้วยค่า y หรือเขียนอยู่ในรูปทั่วไปได้ $x \bmod y$ ซึ่งจะมีวิธีการคิดแบ่งเป็น 2 กรณีดังนี้

1) กรณีที่เป็นค่าบวก เช่น

$3 \bmod 2 = 1$ คือ นำ 3 หารด้วย 2 จะได้เศษเกินมา 1 ซึ่งก็คือคำตอบ

$$\begin{array}{r} 1 \\ 2\overline{)3} \\ \underline{2} \\ \boxed{1} \end{array}$$

$4.5 \bmod 2 = 0.5$ คือ นำ 4.5 หารด้วย 2 จะได้เศษเกินมา 0.5 ซึ่งก็คือคำตอบ

$$\begin{array}{r} 2 \\ 2\overline{)4.5} \\ \underline{4} \\ \boxed{0.5} \end{array}$$

2) กรณีที่เป็นค่าลบ เช่น

$-3 \bmod 2 = 1$ วิธีการหาคือเราจะให้หาว่าค่าถัดไปที่ 2 หารลงตัวได้ นั่นคือ -4 แต่ค่าที่นำมา mod คือ -3 ซึ่งยังขาดอีก $|-4-(-3)| = |1|$ (ให้มองเป็นขนาด) คำตอบก็คือ 1

$-6.5 \bmod 2 = 1.5$ ซึ่งค่าถัดไปที่ 2 หารลงตัวคือ -8 ดังนั้น ค่าที่ได้คือ $|-8-(-6.5)| = |1.5|$ คำตอบก็คือ 1.5

คำตอบที่ได้จากการ mod จะมีค่าเป็นบวกเสมอ ดังเช่น $-7.5 \bmod 2 = 0.5$ และ $7.5 \bmod 2 = 1.5$

3.1.2 การจำลองการทำงานผ่าน VMware Workstation

จากที่ได้กล่าวไว้ในบทที่ 1 หัวข้อ 1.4 บล็อคไดอะแกรมของโครงงานจะประกอบไปด้วยคอมพิวเตอร์ Alice อุปกรณ์ Raspberry Pi 1 อุปกรณ์ Raspberry Pi 2 และคอมพิวเตอร์ Bob เพื่อให้เกิดความสะดวกด้านการเชื่อมต่อสำหรับการทดลองที่ใช้ระบบปฏิบัติการจำนวนมาก ในการทดลองใช้โปรแกรม VMware Workstation จำลองการทำงานของระบบปฏิบัติการทั้งสี่ตัวดังรูปที่ 3.9 จากนั้นทำการเขียนโค้ดคำสั่งบนระบบปฏิบัติการที่จำลองขึ้นแล้วนำโค้ดคำสั่ง มาสร้างลงบนอุปกรณ์ Raspberry Pi 1 และอุปกรณ์ Raspberry Pi 2

3.1.3 การออกแบบการเข้ารหัส - ถอดรหัสลับด้วยภาษาไพธอน

เมื่อจำลองการเข้ารหัส - ถอดรหัสลับแบบเคออติกด้วยโปรแกรม MATLAB แล้วจึงนำมาออกแบบด้วยภาษาไพธอน และนำมาทดสอบกับข้อมูลชนิดข้อความตัวอักษร (Text) เพื่อให้สามารถเข้าใจได้ง่าย และใช้งานได้สะดวกยิ่งขึ้นในการออกแบบจึงเขียนแยกเป็นฟังก์ชันย่อย ได้แก่ ฟังก์ชันการเข้ารหัสลับ ฟังก์ชันการถอดรหัสลับ และฟังก์ชันที่ใช้จัดการกับข้อความตัวอักษร แสดงรายละเอียดแต่ละฟังก์ชันดังนี้

3.1.3.1 ฟังก์ชันเข้ารหัสลับแบบเคออติก

ฟังก์ชันนี้ตั้งชื่อว่า chaos ออกแบบการเข้ารหัสลับแบบเคออติกโดยใช้วงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด ฟังก์ชันการเข้ารหัสลับนี้จะรับตัวแปรอาร์กิวเมนต์ (Argument) 5 ตัวได้แก่ อินพุต (x) ค่าสัมประสิทธิ์ของการเข้ารหัส (c_1 และ c_2) ค่าเงื่อนไขเริ่มต้น (y_0 และ y_1) สำหรับอินพุตจะเป็นตัวแปรชนิดลิสต์ (List) ที่แต่ละสมาชิกเป็นตัวแปรชนิดตัวเลขทศนิยม (Float) จะต้องมีค่าตั้งแต่ -1 แต่ไม่ถึง 1 สำหรับค่าเงื่อนไขเริ่มต้น y_0 และ y_1 จะต้องมีค่าตั้งแต่ -1 แต่ไม่ถึง 1 และสำหรับค่าสัมประสิทธิ์ทั้งสองเป็นตัวแปรชนิด Float และจะต้องมีค่าอย่างน้อย 1 ตัว อยู่นอกสามเหลี่ยมเสถียรภาพเพื่อทำให้ระบบขาดเสถียรภาพ กำหนดเอาต์พุตของการเข้ารหัส (ตัวแปร y) ให้เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวแปรชนิดตัวเลขทศนิยม โดยให้สมาชิก 2 ตัวแรกให้มีค่าเป็น y_0 และ y_1 จากนั้นนำมาคำนวณตามสมการผลต่าง

สืบเนื่องของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด โดยทำให้ระบบเป็นระบบที่ไม่เป็นเชิงเส้นและเกิดการล้นแบบส่วนเติมเต็มสอง ด้วยฟังก์ชันมอดูโล (เช่นเดียวกับการจำลองบนโปรแกรม MATLAB ในหัวข้อที่ 3.1.1) ในที่นี้ใช้คำสั่ง y[k] = ((I+1) % 2) -1 เมื่อฟังก์ชันดำเนินการเสร็จจะคืนค่าเอาต์พุต y ออกมา แสดงคำสั่งต่าง ๆ ที่ใช้ดังรูปที่ 3.9

หมายเหตุ: ตัวแปรชนิดลิสต์เป็นตัวแปรที่มีสมาชิกมากกว่า 1 โดยสมาชิกภายในสามารถเป็นตัวแปรชนิดใดก็ได้เช่น ตัวเลขจำนวนเต็ม (Int) ตัวเลขทศนิยม (Float) สายอักขระ (Strings) เป็นต้น แต่สำหรับตัวแปร x และ y กำหนดให้สมาชิกภายในเป็นตัวแปรชนิดตัวเลขทศนิยม (Float)

```
def chaos (x, c1, c2, y0, y1):
    x = [0.0, 0.0] + x
    y = [0.0] * len (x)
    y[0] = y0
    y[1] = y1
    for k in range (2, len (x)):
        I = x[k] + c1*y[k-1] + c2*y[k-2]
        y[k] = ((I+1) % 2) - 1
    return y
```

รูปที่ 3.9 ฟังก์ชันการเข้ารหัสลับแบบเคออติก

3.1.3.2 ฟังก์ชันการถอดรหัสลับแบบเคออติก

ฟังก์ชันการถอดรหัสลับแบบเคออติก ตั้งชื่อฟังก์ชันว่า dechaos ออกแบบการถอดรหัสลับแบบเคอติกโดยใช้วงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัดที่เป็นระบบผกผันกับวงจรเข้ารหัส ฟังก์ชันการถอดรหัสนี้จะรับค่าตัวแปรอาร์กิวเมนต์ 3 ตัวแปร ได้แก่ อินพุตของการถอดรหัส (yr), ค่าสัมประสิทธิ์ของการถอดรหัส (c_3 และ c_4) สำหรับอินพุตของการถอดรหัสเป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวแปรชนิดตัวเลขทศนิยมที่มีค่าตั้งแต่ -1 แต่ไม่ถึง 1 สำหรับค่าสัมประสิทธิ์ทั้งสองเป็นตัวแปรชนิด Float และกำหนดให้ตรงกับค่าสัมประสิทธิ์ของการเข้ารหัส กำหนดเอาต์พุตของการถอดรหัส (ตัวแปร z) ให้เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวแปลชนิดตัวเลขทศนิยม จากนั้นนำมาคำนวณตามสมการของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด โดยทำให้ระบบเป็นระบบที่ไม่เป็นเชิงเส้นและเกิดการล้นแบบส่วนเติมเต็มสอง ด้วยฟังก์ชันมอดูโล (เช่นเดียวกับการจำลองบนโปรแกรม MATLAB ในหัวข้อที่ 3.1.1) ในที่นี้ใช้คำสั่ง

z[k] = ((J+1) % 2) - 1 เมื่อฟังก์ชันดำเนินการเสร็จจะคืนค่าเอาต์พุต z ออกมา แสดงคำสั่งต่าง ๆ ที่ใช้ดังรูปที่ 3.10

```
def dechaos (yr, c3, c4):
        z = [0] * len (yr)
        for k in range (2, len (yr)):
                J=yr[k] - c3*yr[k-1] - c4*yr[k-2]
                z[k] = ((J+1) % 2) - 1
        return z
```

รูปที่ 3.10 ฟังก์ชันการถอดรหัสลับแบบเคออติก

3.1.3.3 ฟังก์ชัน text_to_int

text_to_int เป็นฟังก์ชันที่ใช้ในการแปลงตัวแปรชนิดสายอักขระไปเป็นตัวแปรชนิดตัวเลขจำนวนเต็ม ฟังก์ชันนี้รับค่าตัวแปรอาร์กิวเมนต์ 1 ตัว คือ t เป็นตัวแปรชนิดสายอักขระ โดยแยกสายอักขระให้อักขระเดี่ยวและเก็บเป็นสมาชิกของตัวแปรลิสต์ด้วยคำสั่ง list_text = list (t) จากนั้นกำหนดตัวแปร i ให้เป็นตัวแปรลิสต์ที่สมาชิกเป็นข้อมูลชนิดตัวเลขจำนวนเต็มด้วยคำสั่ง i=[0] * len (list_text) แล้วทำการแปลงสมาชิกของตัวแปร list_text ให้กลายเป็นตัวเลขจำนวนเต็มที่มีค่าอยู่ในช่วงตั้งแต่ 0 ถึง 255 ตามตารางรหัส ASCII เก็บทับในตัวแปร i เมื่อฟังก์ชันดำเนินการเสร็จจะคืนค่าตัวแปร i ออกมา แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.11

```
def text_to_int (t):
        list_text = list (t)
        i = [0] * len (yr)
        for n in range (2, len (list_text)):
                i[n]=ord (list_text[n])
        return i
```

รูปที่ 3.11 ฟังก์ชัน text_to_int

3.1.3.4 ฟังก์ชัน int_to_float

int_to_float เป็นฟังก์ชันที่ใช้ในการปรับลดค่าจากจำนวนเต็มให้เป็นเลขทศนิยมที่มีค่าอยู่ในช่วงตั้งแต่ -1 ถึง 1 เป็นตัวเลขทศนิยม ฟังก์ชันนี้รับค่าตัวแปรอาร์กิวเมนต์ 1 ตัว คือ i1 เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวเลขจำนวนเต็มมีค่าอยู่ในช่วงตั้งแต่ 0 แต่ไม่ถึง 255 และกำหนดตัวแปร f1 เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวเลขทศนิยมด้วยคำสั่ง f1 = [0.0] * len (i1) จากนั้นทำการลดค่าโดยแยกคิดเป็น 2 ช่วง ช่วงแรกคือช่วงที่มีค่าน้อยกว่าหรือเท่ากับ 127 ให้ทำการปรับลดค่าด้วยการคูณกับ 2^{-7} แล้วเก็บทับในตัวแปร f1 และช่วงที่สองคือช่วงที่มีค่ามากกว่า 127 หลังจากคูณกับ 2^{-7} แล้วต้องลบออกด้วย 1 ก่อนเก็บทับในตัวแปร f1 เมื่อฟังก์ชันดำเนินการเสร็จแล้วก็จะคืนค่าตัวแปร f1 ออกมา แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.12

หมายเหตุ: อาจกล่าวได้ว่าฟังก์ชัน int_to_float คือการทำให้ค่ารหัส ASCII ในเลขฐานสิบขนาด 8 บิต แบบไม่คิดเครื่องหมาย กลายเป็นค่าทศนิยมในเลขฐานสิบขนาด 8 บิต แบบคิดเครื่องหมาย โดยตำแหน่งของทศนิยมอยู่ระหว่าง MSB (Most Significant Bit) และบิตถัดมา

```
def int_to_float (i1):
    f1 = [0.0] * len (i1)
    for n in range (len (i1)):
        if i1[n] <= 127:
            f1[n] = i1[n] * (2 ** -7)
        else :
            f1[n] = -1 + ( (i1[n]-128) * (2**-7))
    return f1
```

รูปที่ 3.12 ฟังก์ชัน int_to_float

3.1.3.5 ฟังก์ชัน float_to_int

float_to_int เป็นฟังก์ชันที่ใช้ในการปรับเพิ่มค่าจากตัวเลขทศนิยมให้เป็นจำนวนเต็มที่มีค่าอยู่ในช่วงตั้งแต่ 0 ถึง 255 ฟังก์ชันนี้รับค่าตัวแปรอาร์กิวเมนต์ 1 ตัว คือ f2 เป็นตัวแปรชนิดลิสที่มีสมาชิกเป็นเลขทศนิยมที่มีค่าอยู่ในช่วงตั้งแต่ -1 แต่ไม่ถึง 1 และกำหนดตัวแปร i2 เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวเลขจำนวนเต็มด้วยคำสั่ง i2 = [0] * len (f2) จากนั้นทำการปรับเพิ่มค่าโดยแยกคิดเป็น 2 ช่วง ช่วงแรกคือช่วงที่มีค่าตั้งแต่ 0 แต่ไม่ถึง 1 ให้ทำการปรับเพิ่มค่าด้วยการคูณกับ 2^{7} แล้วเก็บทับในตัวแปร i2 และช่วงที่สองคือคือช่วงที่มีค่าตั้งแต่

-1 แต่ไม่ถึง 0 ซึ่งหลังจากคูณกับ 2^7 แล้วจะต้องบวกกับ 256 ก่อนเก็บทับในตัวแปร i2 เมื่อฟังก์ชันดำเนินการเสร็จแล้วก็จะคืนค่าตัวแปร i2 ออกมา แสดงคำสั่งดังรูปที่ 3.13

```
def float_to_int (f2):
        i2 = [0] * len (f2)
        for n in range (len (f2)):
                if f2[n] >= 0:
                        i2[n] = int (round  (f2[n] * (2**7)))
                else:
                        i2[n] = int (round ((f2[n] * (2**7)) + 256))
        return  i2
```

รูปที่ 3.13 ฟังก์ชันเพิ่มค่าของตัวแปรชนิด Float ให้อยู่ในรูปตัวแปร Int มีค่า 0 ถึง 255

3.1.3.6 ฟังก์ชัน in_to_text

int_to_text เป็นฟังก์ชันที่ใช้ในการแปลงตัวแปรชนิดลิสต์ที่มีสมาชิกจำนวนเต็มไปเป็นตัวแปรชนิดสายอักขระ ฟังก์ชันนี้รับค่าตัวแปรอาร์กิวเมนต์ 1 ตัว คือ i3 เป็นตัวแปรชนิดลิสต์ที่มีสมาชิกเป็นตัวเลขจำนวนเต็ม จากนั้นทำการแปลงจำนวนเต็มแต่ละสมาชิกให้กลายเป็นอักขระเดี่ยว จากนั้นนำอักขระเดี่ยวทั้งหมดมาเรียงต่อกันและเก็บในตัวแปร t3 ซึ่งเป็นตัวแปรชนิดสายอักขระด้วยคำสั่ง t3 = “”.join (chr (d) for d in i3) เมื่อดำเนินการเสร็จก็จะคืนค่าตัวแปร t3 ออกมา แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.14

```
def int_to_text (i3):
        t3 = “”.join (chr (d) for d in i3)  #join list into message
return t3
```

รูปที่ 3.14 ฟังก์ชันแปลงตัวแปรชนิด int ไปเป็นตัวแปรชนิดสตริงส์ตามรหัส ASCII

เมื่อเขียนฟังก์ชันทั้งหมดเสร็จแล้ว จึงนำแต่ละฟังก์ชันมาใช้งานร่วมกัน เพื่อทำการทดสอบการเข้ารหัส - ถอดรหัสลับแบบเคออติก กับข้อมูลชนิดข้อความตัวอักษร โดยมีคำสั่งเรียกใช้งานฟังก์ชัน และคำสั่งอื่น ๆ แสดงดังรูปที่ 3.15

```
coeff1 = input ("c1=")
coeff2 = input ("c2=")
xtext = raw_input("Please insert your message:")
xint = text_to_int (xtext)
xfloat = int_to_float (xint)
yfloat = chaos (xfloat, coeff1, coeff2)
yint = float_to_int (yfloat)
ytext = int_to_text (yint)
print "Encoded Message is:%s\tLength=%d" %(ytext, len (ytext))
```

รูปที่ 3.15 คำสั่งที่ใช้ในการเข้ารหัสลับแบบเคออติกกับข้อความตัวอักษร

จากรูปที่ 3.15 โดยเริ่มจากการกำหนดค่าสัมประสิทธิ์ของการเข้ารหัส - ถอดรหัสลับ ในที่นี้ป้อนเป็น c_1 =4, c_2 = -1 และรับข้อความตัวอักษรจากแป้นพิมพ์มาเก็บไว้ในตัวแปร xtext ด้วยคำสั่ง xtext = raw_input ("Please insert your message:") ในที่นี้จะป้อนให้

xtext = "Hello world"

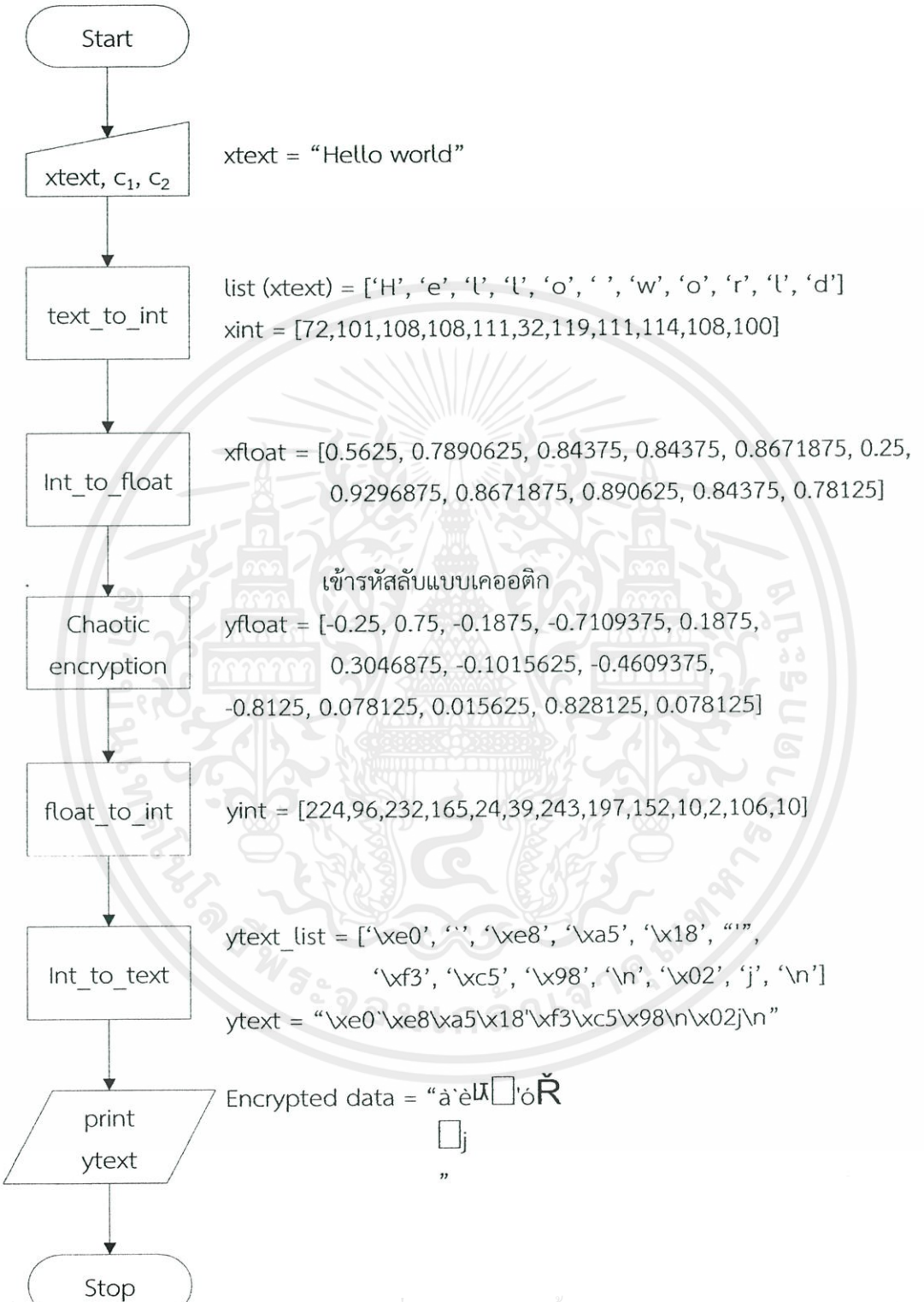
จากนั้นทำการแปลง xtext ด้วยฟังก์ชัน text_to_int (ดูจากหัวข้อ 3.1.3.3) และเก็บค่าไว้ในตัวแปร xint จะได้ว่า xint = [72, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100] จากนั้นทำการแปลงต่อด้วยฟังก์ชัน int_to_float (ดูจากหัวข้อ 3.1.3.4) เก็บไว้ในตัวแปร xfloat จะได้ว่า xfloat = [0.5625, 0.7890625, 0.84375, 0.84375, 0.8671875, 0.25, 0.9296875, 0.8671875, 0.890625, 0.84375, 0.78125] เมื่อได้ค่าอินพุตของการเข้ารหัสลับเป็นตัวแปรชนิดตัวเลขทศนิยมที่มีค่าอยู่ในช่วงตั้งแต่ -1 แต่ไม่ถึง 1 ก็จะสามารถนำมาเข้ารหัสลับได้ โดยการเข้ารหัสนี้ได้ระบุค่าสัมประสิทธิ์ c_1, c_2 และกำหนดค่าเงื่อนไขเริ่มต้นเป็น y_0 = -0.25 และ y_1 = 0.75 ทำการเข้ารหัสโดยใช้ฟังก์ชัน chaos (ดูจากหัวข้อ 3.1.3.1) และเก็บเอาต์พุตของการเข้ารหัสลับไว้ในตัวแปร yfloat จะได้ว่า yfloat = [-0.25, 0.75, -0.1875, -0.7109375, 0.1875, 0.3046875, -0.1015625, -0.4609375, -0.8125, 0.078125, 0.015625, 0.828125, 0.078125]

จากนั้นนำไปแปลงกลับให้เป็น yint ด้วยฟังก์ชัน float_to_int (ดูจากหัวข้อ 3.1.3.5) จะได้ว่า yint = [224, 96, 232, 165, 24, 39, 243, 197, 152, 10, 2, 106, 10] และสุดท้ายทำการแปลงกลับมาเป็นข้อความตัวอักษรอีกครั้งด้วยฟังก์ชัน int_to_text (ดูจากหัวข้อ 3.1.3.5) ก็จะได้ ytext = “\xe0`\xe8\xa5\x18'\xf3\xc5\x98\n\x02j\n” ซึ่งจะแสดงข้อความที่ผ่านการเข้ารหัสลับ (Encrypted data) เป็น

Encrypted data = “à`è¥↑ 'óŘ

ㄱ j”

แผนผังการทำงานของการเข้ารหัสลับแบบเคออติกแสดงดังรูปที่ 3.16



รูปที่ 3.16 ผังการทำงานของการเข้ารหัสลับแบบเคออติกกับข้อความตัวอักษร

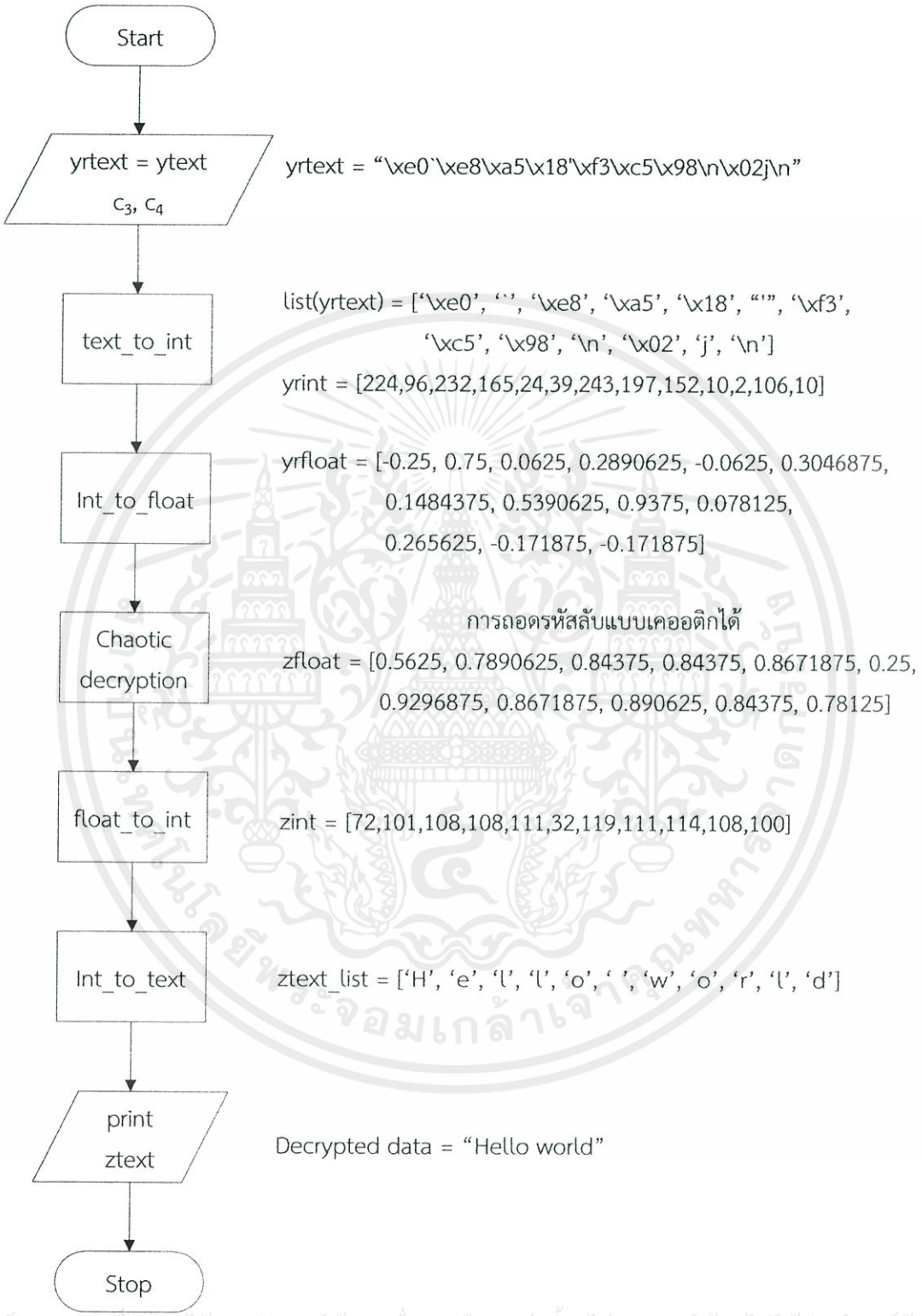
เมื่อได้ข้อความที่ผ่านการเข้ารหัส ytext แล้วเปลี่ยนไปเก็บในตัวแปร yrtext จากนั้นทำการแปลง yrtext ให้กลายเป็น yrint จะได้ว่า yrint = [224, 96, 232, 165, 24, 39, 243, 197, 152, 10, 2, 106, 10] แล้วทำการแปลงให้กลายเป็น yrfloat จะได้ว่า yrfloat = [-0.25, 0.75, -0.1875, -0.7109375, 0.1875, 0.3046875, -0.1015625, -0.4609375, -0.8125, 0.078125, 0.015625, 0.828125, 0.078125] เมื่อได้ค่าอินพุตของการถอดรหัสลับเป็นตัวแปรชนิดตัวเลขทศนิยมที่มีค่าอยู่ในช่วงตั้งแต่ -1 แต่ไม่ถึง 1 ก็จะสามารถนำมาถอดรหัสลับได้ โดยการถอดรหัสนี้ได้ระบุค่าสัมประสิทธิ์ c_3 = 4, c_4 = -1 ทำการถอดรหัสโดยใช้ฟังก์ชัน dechaos (ดูจากหัวข้อ 3.1.3.1) และเก็บเอาต์พุตของถอดรหัสลับไว้ในตัวแปร zfloat จะได้ว่า zfloat = [0.5625, 0.7890625, 0.84375, 0.84375, 0.8671875, 0.25, 0.9296875, 0.8671875, 0.890625, 0.84375, 0.78125] จากนั้นนำไปแปลงกลับให้เป็น zint จะได้ว่า zint= [72, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100] และสุดท้ายทำการแปลงกลับมาเป็นข้อความ ก็จะได้ ztext = “Hello world” ข้อความที่ผ่านการถอดรหัสรหัสลับ (Decrypted data) เป็น

Decrypted data = “Hello world”

แสดงคำสั่งทั้งหมดดังรูปที่ 3.17 และแสดงแผนผังการทำงานของการถอดรหัสลับแบบเคออติกดังรูปที่ 3.18 ตามลำดับ

```
### Chaotic decryption _ text process
coeff3 = input ("c3=")
coeff4 = input ("c4=")
yrtext = ytext
yrint = text_to_int (yrtext)
yrfloat = int_to_float (yrint)
zfloat = (dechaos (yrfloat, coeff3, coeff4))[2:]
zint = float_to_int (zfloat)
ztext = int_to_text (zint)
print "Decoded Message is:%s\tLength =%d" %(ztext, len (ztext))
```

รูปที่ 3.17 คำสั่งที่ใช้ในการถอดรหัสลับแบบเคออติกกับข้อความตัวอักษร



รูปที่ 3.18 ผังการทำงานของการถอดรหัสลับแบบเคออติกกับข้อความตัวอักษร

3.1.4 การออกแบบการทดลองสื่อสารข้อมูลอนุกรมที่มีการเข้ารหัส - ถอดรหัสลับแบบเคออติก โดยผ่านอุปกรณ์ Raspberry Pi

หัวข้อนี้เป็นการนำวงจรเข้ารหัส - ถอดรหัสลับแบบเคออติกที่ได้ออกแบบด้วยภาษาไพธอน มาทำการ Implement ลงบนอุปกรณ์ Raspberry Pi มาใช้กับการสื่อสารข้อมูลอนุกรมจากคอมพิวเตอร์ฝั่งส่ง (Alice) ไปยังคอมพิวเตอร์ฝั่งรับ (Bob) โดยฝั่งส่งทำการเข้ารหัสลับด้วยอุปกรณ์ Raspberry Pi 1 และฝั่งรับทำการถอดรหัสลับด้วยอุปกรณ์ Raspberry Pi 2

3.1.4.1 การทดลองส่งข้อมูลอนุกรมจากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

อุปกรณ์ Raspberry pi สามารถรับส่งข้อมูลอนุกรม โดยรับส่งข้อมูลผ่านพอร์ต UART ซึ่งจัดเรียงอยู่บนพอร์ต GPIO โดยขา TxD คือ ขา 8 ของพอร์ต GPIO และขา RxD คือ ขา 10 ของพอร์ต GPIO การเชื่อมต่ออุปกรณ์จะต่อจากขา TxD ของอุปกรณ์ Raspberry Pi 1 เข้ากับขา RxD ของอุปกรณ์ Raspberry Pi 2 ดังรูปที่ 3.19

รูปที่ 3.19 การเชื่อมต่อ สำหรับการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยัง Raspberry Pi 2

การส่งข้อมูลอนุกรมจากอุปกรณ์ Raspberry Pi 1 เริ่มด้วยการตั้งค่าพอร์ตที่ใช้งานเป็นพอร์ต "/dev/ttyAMA0" (พอร์ต UART) และค่าบอดเรต (Baudrate) เก็บค่าไว้ในตัวแปรชื่อ ser ด้วยคำสั่ง ser=serial.Serial ("/dev/ttyAMA0",9600) จากนั้นเปิดใช้งานพอร์ตด้วยคำสั่ง ser.open() ทำการรับข้อความจากแป้นพิมพ์ และปิดท้ายข้อความด้วย "\x00" (ตัวแปร delim) จากนั้นส่งออกพอร์ตอนุกรมด้วยคำสั่ง ser.write(data+delim) เมื่อส่งเสร็จก็ปิดพอร์ตด้วยคำสั่ง ser.close แสดงคำสั่งการส่งข้อความดังรูปที่ 3.20

```
import serial
ser = serial.Serial( "/dev/ttyAMA0", 9600)
if not ser.isOpen():
        ser.open()
delim = "\x00"
msg = raw_input("Pi1 send to Pi2: ")
ser.write(data+delim)
ser.close()
```

รูปที่ 3.20 การส่งข้อมูลอนุกรมบนอุปกรณ์ Raspberry Pi 1

การรับข้อมูลอนุกรมที่อุปกรณ์ Raspberry Pi 2 เริ่มด้วยการตั้งค่าพอร์ตอนุกรมให้ค่าบอดเรทตรงกับฝั่งส่ง และกำหนดค่าเวลาในการรับข้อมูล (timeout) เป็น 1 วินาที เพื่อให้การทำงานไม่หยุดชะงัก เมื่อรับข้อมูลไม่ได้ภายใน 1 วินาที ก็จะข้ามไปทำคำสั่งถัดไป ด้วยคำสั่ง ser = serial.Serial("/dev/ttyAMA9",9600,timeout = 1) จากนั้นเปิดใช้งานพอร์ตเพื่อรับข้อมูล โดยรับทีละตัวอักษรเรียงต่อกันเป็นข้อความจนได้รับ "\x00" ด้วยคำสั่ง data = "".join(iter(lambda: ser.read(1), delim)) ก็จะโดยตั้งเงื่อนไขให้วนรับจนกว่าจะมีข้อความเข้ามา จากนั้นจึงแสดงผลด้วยคำสั่ง print และปิดพอร์ตอนุกรมเมื่อรับเสร็จ แสดงคำสั่งการรับข้อความดังรูปที่ 3.21

```
import serial
ser = serial.Serial( "/dev/ttyAMA0", 9600, timeout=1)
if not ser.isOpen() :
        ser.open()
delim = "\x00"
data = "".join(iter(lambda: ser.read(1), delim))
print "Pi2 recv from Pi1: ", data
ser.close()
```

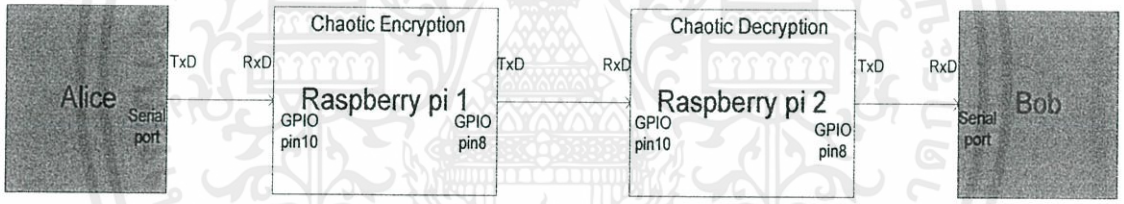
รูปที่ 3.21 การรับข้อมูลอนุกรมบนอุปกรณ์ Raspberry Pi 2

หมายเหตุ : ฝั่งส่งต้องมีตัวบ่งบอกการจบข้อความปิดท้ายทุกครั้ง เพื่อให้ฝั่งรับรู้ว่า ได้รับข้อความครบถ้วนแล้ว

3.1.4.2 การทดสอบการรับส่งข้อมูลอนุกรมที่มีการเข้ารหัส - ถอดรหัสลับแบบเคออติก

เพื่อทดลองส่งข้อมูลจากคอมพิวเตอร์ฝั่งส่ง (Alice) โดยทำการเข้ารหัสลับด้วยอุปกรณ์ Raspberry Pi 1 ก่อนส่งออก ส่วนฝั่งรับจะทำการถอดรหัสลับด้วยอุปกรณ์ Raspberry Pi 2 และคอมพิวเตอร์ (Bob) จะได้รับข้อมูลที่ผ่านการถอดรหัสลับแล้ว

1) การเชื่อมต่ออุปกรณ์ทั้งหมดจากฝั่งส่งไปจนถึงฝั่งรับ (ก่อนที่จะเริ่มต่อนั้นที่คอมพิวเตอร์ทั้งสองฝั่งจำเป็นจะต้องต่ออุปกรณ์ USB to serial converter เพื่อใช้งานพอร์ตอนุกรม) เริ่มจากขา TxD ของคอมพิวเตอร์ Alice ต่อเข้ากับขา RxD ของอุปกรณ์ Raspberry Pi 1 ส่วนขา TxD ของอุปกรณ์ Raspberry Pi 1 ต่อเข้ากับขา RxD ของอุปกรณ์ Raspberry Pi 2 และสุดท้ายขา TxD ต่อเข้ากับขา RxD ของคอมพิวเตอร์ Bob (โดยที่ GND ทั้งหมดเชื่อมถึงกัน) แสดงการเชื่อมต่อทั้งหมดดังรูปที่ 3.22



รูปที่ 3.22 การเชื่อมต่ออุปกรณ์ทั้งหมดจากฝั่งส่งไปยังฝั่งรับ

สำหรับการเชื่อมต่อระหว่างคอมพิวเตอร์อุปกรณ์ USB to serial converter ที่นำมาต่อเข้ากับคอมพิวเตอร์ เพื่อใช้งานพอร์ตอนุกรมนั้นสัญญาณมีระดับแรงดันอยู่ที่ ± 12 LV RS-232 ในขณะที่พอร์ต UART ของ Raspberry pi นั้นมีระดับแรงดันที่ 0 - 3.3 LV TTL จึงใช้ไอซี MAX-3232 ในการแปลงระดับแรงดันของสัญญาณจากมาตรฐาน RS-232 เป็น TTL และแปลงระดับสัญญาณจาก TTL ให้กลายเป็น RS-232

2) การทำงานของคอมพิวเตอร์ฝั่งส่ง (Alice) ตั้งค่าพอร์ตที่ใช้งานเป็น “COM6” โดยกำหนดค่าบอดเรทที่ 9600 เปิดพอร์ตอนุกรม และทำการรับข้อความจากแป้นพิมพ์ และปิดท้ายข้อความด้วย “\x00” จากนั้นส่งออกพอร์ตอนุกรม และปิดพอร์ตเมื่อส่งเสร็จ และวนกลับไปรับข้อความจากแป้นพิมพ์อีกครั้ง แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.23

```
import serial
while 1:
        ser=serial.Serial( "COM6", 9600)
        if not ser.isOpen():
                ser.open()
        delim = "\x00"
        data=raw_input("Alice send to Bob: ")
        ser.write(data+delim)
        ser.close()
```

รูปที่ 3.23 การทำงานของคอมพิวเตอร์ฝั่งส่ง (Alice)

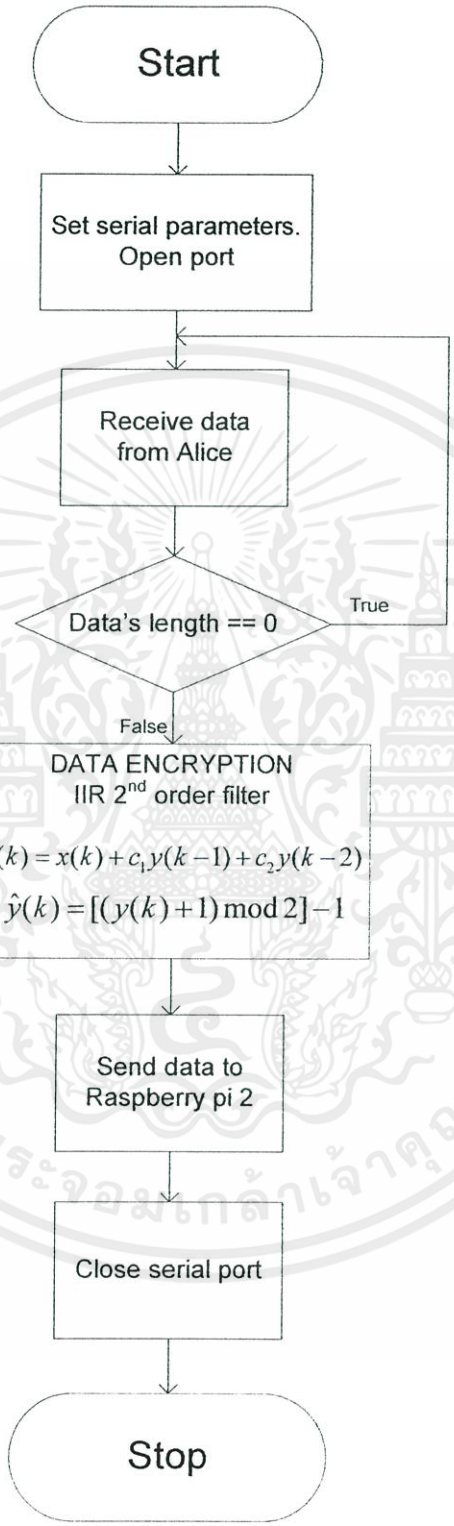
3) การทำงานของอุปกรณ์ Raspberry Pi 1 ทำหน้าที่เป็นอุปกรณ์เข้ารหัสลับแบบเคออติก กำหนดพอร์ตที่ใช้งานเป็น "/dev/ttyAMA0" ตั้งค่าบอดเรตให้ตรงกับคอมพิวเตอร์ Alice (เท่ากับ 9600) และค่าเวลาในการรับข้อมูล 1 วินาที เปิดใช้งานพอร์ตอนุกรม แล้วรอรับข้อความจากคอมพิวเตอร์ Alice แล้วนำมาเก็บไว้ในตัวแปร xtext และนำไปเข้ารหัสลับโดยกำหนดค่าสัมประสิทธิ์ $c_1 = 4$, $c_2 = -1$ จากนั้นนำข้อความที่ผ่านการเข้ารหัสลับ (ตัวแปร ytext) ส่งต่อไปยังฝั่งรับ ปิดพอร์ตอนุกรม และวนกลับไปรับข้อความครั้งถัดไป แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.24 และแสดงผังการทำงานดังรูปที่ 3.25

```
import serial
while 1:
        # Receive data from Alice
        ser = serial.Serial( "/dev/ttyAMA0", 9600, timeout=1)
        if not ser.isOpen() :
                ser.open()
        delim = "\x00"
        data = "".join(iter(lambda: ser.read(1), delim))
        xtext=data

        # Chaotic Encryption
        xint = text_to_int(xtext)
        xfloat = int_to_float(xint)
        yfloat = chaos(xfloat, 4, -1, 0.0, 0.0)
        yint = float_to_int(yfloat)
        ytext = int_to_text(yint)

        # Send data to Raspberry Pi 1
        ser.write(ytext+delim)
        ser.close()
```

รูปที่ 3.24 การทำงานของอุปกรณ์ Raspberry Pi 1



รูปที่ 3.25 ผังการทำงานของอุปกรณ์ Raspberry Pi 1

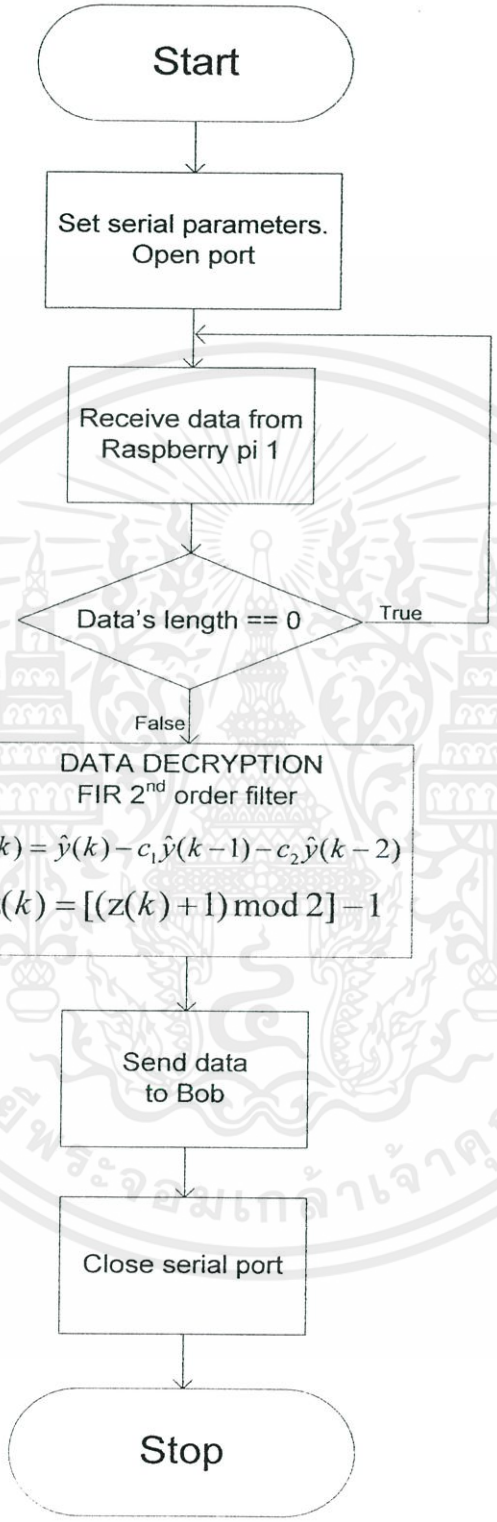
4) การทำงานของฝั่งรับ อุปกรณ์ Raspberry Pi 2 ทำหน้าที่เป็นอุปกรณ์ถอดรหัสลับแบบเคออติก กำหนดพอร์ตที่ใช้งานเป็น "/dev/ttyAMA0" ตั้งค่าบอดเรตให้ตรงกับอุปกรณ์ Raspberry Pi 1 (เท่ากับ 9600) และเวลาในการรับข้อมูล 1 วินาที เปิดใช้งานพอร์ตอนุกรม รอรับข้อความจากอุปกรณ์ Raspberry Pi 1 แล้วมาเก็บไว้ในตัวแปร yrtext แล้วจึงนำไปถอดรหัสลับ โดยกำหนดค่าสัมประสิทธิ์ของการถอดรหัสลับให้ตรงกับค่าสัมประสิทธิ์ของการเข้ารหัสลับ (c_3 = 4, c_4 = -1) และส่งข้อความที่ถอดรหัสได้ (ตัวแปร ztext) ต่อไปยังคอมพิวเตอร์ Bob ปิดพอร์ตอนุกรม และวนกลับไปรับข้อความครั้งถัดไป แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.26 และแสดงผังการทำงานดังรูปที่ 3.27

```
import serial
while 1:
        # Receive data from Raspberry Pi 1
        ser = serial.Serial( "/dev/ttyAMA0", 9600, timeout=1)
        if not ser.isOpen() :
                ser.open()
        delim = "\x00"
        data = "".join(iter(lambda: ser.read(1), delim))
        ytext=data

        # Chaotic Decryption
        yint = text_to_int(ytext)
        yfloat = int_to_float(yint)
        zfloat = dechaos(yfloat, 4, -1)
        zint = float_to_int(zfloat)
        ztext = int_to_text(zint)

        # Send data to Bob
        ser.write(ztext+delim)
        ser.close()
```


รูปที่ 3.26 การทำงานของอุปกรณ์ Raspberry Pi 2



รูปที่ 3.27 ผังการทำงานของอุปกรณ์ Raspberry Pi 2

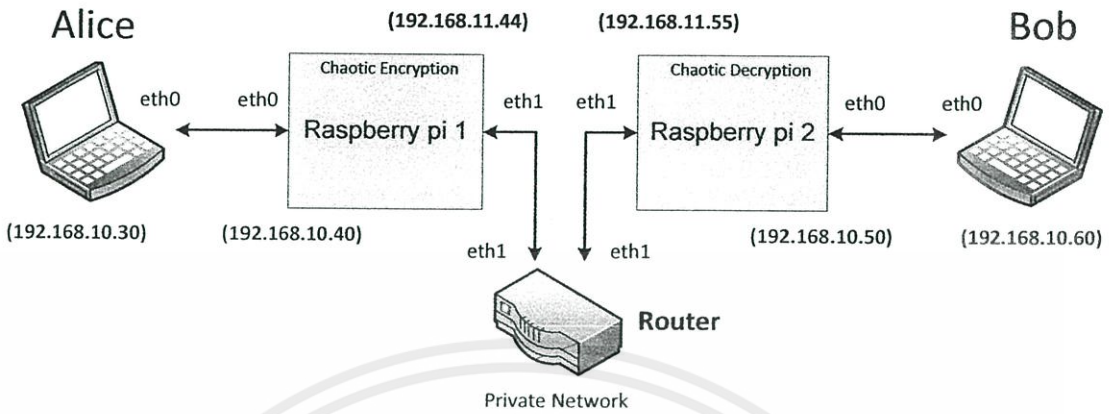
5) การทำงานของคอมพิวเตอร์ Bob กำหนดพอร์ตที่ใช้งานเป็น “COM6” ตั้งค่าบอดเรตให้ตรงกับอุปกรณ์ Raspberry Pi 2 (เท่ากับ 9600) และตั้งค่าเวลาของการรับข้อมูล (Timeout) ไว้ที่ 1 วินาที เปิดใช้งานพอร์ตอนุกรมเพื่อรอรับข้อความที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 2 เก็บไว้ในตัวแปร data แสดงผลด้วยคำสั่ง print และสุดท้ายปิดพอร์ตอนุกรมเมื่อรับเสร็จ และวนกลับไปรอรับข้อความครั้งถัดไป แสดงคำสั่งต่าง ๆ ดังรูปที่ 3.28

```
import serial
while 1:
        ser=serial.Serial(port = “COM6”, baudrate=9600, timeout = 1)
        if not ser.isOpen():
                ser.open()
        delim = “\x00”
        data = "".join(iter(lambda: ser.read(1), delim))
        while data.__len__() == 0 :
                data="".join(iter(lambda: ser.read(1), delim))
        print "Bob recv from Alice: ", data
        ser.close()
```

รูปที่ 3.28 การทำงานของคอมพิวเตอร์ฝั่งส่ง (Bob)

3.1.5 การออกแบบระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi สำหรับการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต

ระบบการสื่อสารที่ได้ทำการออกแบบ ประกอบไปด้วยผู้ใช้สองคน ได้แก่ A และ B อุปกรณ์เข้ารหัส-ถอดรหัสที่ติดต่ออยู่กับ A (Raspberry Pi 1) อุปกรณ์เข้ารหัส – ถอดรหัสที่ติดต่ออยู่กับ B (Raspberry Pi 2) และเราเตอร์ โดยรูปแบบของระบบเครือข่ายที่ใช้ในการทดลองเป็นแบบ Private Network ใช้โปรโตคอล UDP และ TCP สำหรับการสื่อสารข้อมูลผ่านเครือข่าย รูปแบบของ IP address ใช้ IP lv.4 และกำหนดค่าแสดงดังรูปที่ 3.29



รูปที่ 3.29 ภาพรวมการออกแบบการเข้ารหัส - ถอดรหัสบน Raspberry Pi สำหรับการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต

หลังจากออกแบบระบบสื่อสาร ทำการออกแบบการทำงานของโปรแกรมในแต่ละส่วนของระบบสื่อสาร โดยใช้รูปแบบการสื่อสารเป็นแบบการสื่อสารทางเดียว และการสื่อสารแบบ Full-duplex

3.1.5.1 การทำงานของระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

ในรูปแบบการสื่อสารทางเดียวกำหนดให้คอมพิวเตอร์ Alice เป็นผู้เริ่มการเชื่อมต่อการสื่อสารและเป็นผู้ส่งข้อมูล ส่วนคอมพิวเตอร์ Bob กำหนดให้เป็นผู้รับข้อมูล

1) การทำงานในส่วนคอมพิวเตอร์ Alice เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล UDP จากนั้นวนลูปรอรับข้อมูลจากผู้ใช้โดยใช้คำสั่ง raw_input() และส่งข้อมูลที่รับได้ไปยังอุปกรณ์ Raspberry Pi 1 ซึ่งทำหน้าที่เป็น UDP Server โดยใช้คำสั่ง s.sendto(msg, (HOST, PORT)) ตัวแปร data คือ ข้อมูลที่รับจากผู้ใช้, ตัวแปร HOST คือ หมายเลขไอพีของอุปกรณ์ Raspberry Pi 1 และตัวแปร PORT คือ พอร์ตสำหรับการรับ-ส่ง ข้อมูลของอุปกรณ์ Raspberry Pi 1 โค้ดการทำงานในส่วน A แสดงดังรูปที่ 3.30

```
import socket, sys
HOST = '192.168.10.40'
PORT = 3000
   #create UDP socket
try :
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error :
        print 'Fail to create socket'
        sys.exit()
print "socket created"

while 1:
        try :
                msg = raw_input('Please sent your data :')
                # send data to RPi 1
                s.sendto(msg, (HOST, PORT))
        except socket.error , msg :
                print 'Error code'
s.close()
```

รูปที่ 3.30 การทำงานของคอมพิวเตอร์ Alice ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

2) การทำงานในส่วนของอุปกรณ์ Raspberry Pi 1 ใช้ port Ethernet สองพอร์ตสำหรับการสื่อสารกับคอมพิวเตอร์ Alice และอุปกรณ์ Raspberry Pi 2 เริ่มต้นการทำงานด้วยการสร้าง UDP Socket (s) สำหรับการใช้งานเป็น UDP Server ในการสื่อสารกับคอมพิวเตอร์ Alice และสร้าง UDP Socket (s1) สำหรับการใช้งานเป็น UDP client ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 จากนั้นวนลูปรับข้อมูลจากคอมพิวเตอร์ Alice โดยใช้คำสั่ง s.recvfrom(1024) นำข้อมูลผ่านกระบวนการเข้ารหัส และส่งข้อมูลที่ผ่านการเข้ารหัสไปยังอุปกรณ์ Raspberry Pi 2 โดยใช้คำสั่ง s1.sendto (ytext, (host2, port2)) ตัวแปร ytext คือข้อมูลที่ผ่านการเข้ารหัส โค้ดในการทำงานของอุปกรณ์ Raspberry Pi 1 แสดงดังรูปที่ 3.31

```
Import socket, sys
# create UDP socket to communicate with Alice
s = UDPsocket_server()
# create UDP socket to communicate with RPi2
s1 = UDPsocket_client()
while True :
        # receive data from Alice
        d = s.recvfrom(1024)
         dataA = d[0]
        addrA = d[1]   #tuple (ip , port)
        ##### Start process encrypt data #####
        xint=text_to_int(dataA)
        xint=[0, 0]+xint
        xfloat=int_to_float(xint)
        yfloat=chaos(xfloat)
        yint=float_to_int(yfloat)
        ytext=int_to_text(yint) # Encrypted data
       # send encrypted data to RPi 2
       s1.sendto(ytext, (host2, port2))
```

รูปที่ 3.31 การทำงานของ Raspberry Pi 1 ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

3) การทำงานในส่วนของอุปกรณ์ Raspberry Pi 2 ใช้ port Ethernet สองพอร์ตสำหรับการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และคอมพิวเตอร์ Bob เริ่มต้นการทำงานด้วยการสร้าง UDP Socket (s) สำหรับการใช้งานเป็น UDP Server ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และสร้าง UDP Socket (s1) สำหรับการใช้งานเป็น UDP Server ในการสื่อสารกับคอมพิวเตอร์ Bob จากนั้นรับการเชื่อมต่อจากคอมพิวเตอร์ Bob และวนลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 โดยใช้คำสั่ง s.recvfrom(1024) นำข้อมูลผ่านกระบวนการถอดรหัสและส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Bob โดยใช้คำสั่ง s2.sendto(ztext, addrB) ตัวแปร ztext คือ ข้อมูลที่ผ่านการถอดรหัส และตัวแปร addrB คือ ข้อมูลหมายเลขไอพีและพอร์ตของคอมพิวเตอร์ Bob โค้ดการทำงานของอุปกรณ์ Raspberry Pi 2 แสดงดังรูปที่ 3.32

```
Import socket, sys
# create UDP socket  to communicate with Bob
s = UDPsocket_server()
# create UDP socket  to communicate with RPi 1
s1 = UDPsocket_server()
while True:
        # receive data from RPi 1
        dP = s.recvfrom(1024)
        dataP = dP[0]
        addrP = dP[1]
          ### start process decrypt data ###
        yrtext=dataP
        yrint=text_to_int(yrtext)
        yrint=[0, 0]+yrint
        yrfloat=int_to_float(yrint)
        zfloat=dechaos(yrfloat)
        zint=float_to_int(zfloat)
        ztext=int_to_text(zint)  # decrypted data
          # send decoded data to Bob
        s1.sendto(ztext , addrB)
```

รูปที่ 3.32 การทำงานของ Raspberry Pi 2 ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

4) ทำงานในส่วนคอมพิวเตอร์ Bob เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล UDP จากนั้นสร้างการเชื่อมต่อไปยังอุปกรณ์ Raspberry Pi 2 ซึ่งทำหน้าที่เป็น UDP Server วนลูปการทำงานรับข้อมูลที่ผ่านการถอดรหัสจากอุปกรณ์ Raspberry Pi 2 โดยใช้คำสั่ง s.recvfrom(1024) และแสดงข้อมูลที่รับได้ โค้ดการทำงานในส่วนการสร้างการเชื่อมต่อ และลูปการทำงานแสดงดังรูปที่ 3.33

```
HOST = '192.168.10.50'
PORT = 5000
# create connection
msg= "Hello"
s.sendto(msg, (HOST, PORT))
# receive acknowledge
dS = s.recvfrom(1024)
print dS

while True:
    ### receive data from RPi 2
    dP = s.recvfrom(1024)
    dataP = dP[0]
    addrP = dP[1]
    print 'Data Form A :' + dataP
```

รูปที่ 3.33 การทำงานของคอมพิวเตอร์ Bob ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

3.1.5.2 การทำงานของระบบสื่อสารทางเดียวโดยใช้โปรโตคอล UDP

ในรูปแบบการสื่อสารทางเดียวกำหนดให้คอมพิวเตอร์ Alice เป็นผู้เริ่มการเชื่อมต่อการสื่อสารและเป็นผู้ส่งข้อมูล ส่วนคอมพิวเตอร์ Bob กำหนดให้เป็นผู้รับข้อมูล

1) การทำงานในส่วนคอมพิวเตอร์ Alice เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล TCP เชื่อมต่อไปยัง TCP Server บนอุปกรณ์ Raspberry Pi 1 จากนั้นวนลูปรอรับข้อมูลจากผู้ใช้โดยใช้คำสั่ง raw_input() และส่งข้อมูลที่รับได้ไปยังอุปกรณ์ Raspberry Pi 1 โดยใช้คำสั่ง s.send(data) ตัวแปร data คือ ข้อมูลที่รับจากผู้ใช้ โค้ดในส่วนการสร้าง TCP Socket สำหรับการใช้งานเป็น TCP Client แสดงดังรูปที่ 3.34 และโค้ดในส่วนลูปการทำงานแสดงดังรูปที่ 3.35 ตามลำดับ

```
import socket, sys
HOST = '192.168.10.40'
PORT = 3000
#create TCP socket
try :
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error :
        print "Fail to create socket"
        sys.exit()
#connect to RPi 1 server
try:
        s.connect((HOST , PORT))
except socket.error, e:
        print "Connect to destination Error"
```

รูปที่ 3.34 การสร้าง TCP Socket สำหรับการใช้งานเป็น TCP Client

```
while True:
        try :
                data = raw_input('Please sent your data :')
                # send data to RPi 1
                s.send (data)
        except socket.error , msg :
                print 'Error code'
```

รูปที่ 3.35 การทำงานของคอมพิวเตอร์ Alice ในส่วนรอรับข้อมูลจากผู้ใช้และการส่งข้อมูลของระบบสื่อสารทางเดียวโดยใช้โปรโตคอล TCP

2) การทำงานในส่วน Raspberry Pi 1 เริ่มต้นการทำงานด้วยการสร้าง TCP Socket (s) สำหรับการใช้งานเป็น TCP Server ในการสื่อสารกับคอมพิวเตอร์ Alice และสร้าง TCP Socket (s1) สำหรับการใช้งานเป็น TCP Client ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 จากนั้นวนลูปรับข้อมูลจากคอมพิวเตอร์ Alice โดยใช้คำสั่ง s_conn.recv(1024) ตัวแปร s_conn คือ ค่าของ Socket ที่ได้เมื่อ TCP Server ทำคำสั่ง accept() นำข้อมูลผ่านกระบวนการเข้ารหัส และส่งข้อมูลที่ผ่านการเข้ารหัสไปยังอุปกรณ์ Raspberry Pi 2 โดยใช้คำสั่ง s1.send(ytext) ตัวแปร ytext คือ ข้อมูลที่ผ่านการเข้ารหัส โค้ดการทำงานของอุปกรณ์ Raspberry Pi 1 แสดงดังรูปที่ 3.36 และโค้ดในส่วนการสร้าง TCP Socket สำหรับการใช้งานเป็น TCP Server แสดงดังรูปที่ 3.37 ตามลำดับ

```
Import socket, sys
# create TCP socket to communicate with Alice
s = TCPsocket_server()
# create TCP socket to communicate with RPi 2
s1 = TCPsocket_client()
while True :
    # receive data from Alice
    data = s_conn.recv(1024)
    ##### Start process encrypt data #####
    xtext=data
    xint=text_to_int(xtext)
    xint=[0, 0]+xint
    xfloat=int_to_float(xint)
    yfloat=chaos(xfloat)
    yint=float_to_int(yfloat)
    ytext=int_to_text(yint) # Encrypted data
    # send encrypted data to RPi 2
    s1.sendto(ytext, (host2, port2))
```

รูปที่ 3.36 การทำงานของ Raspberry Pi 1 ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล TCP

```
import socket, sys
HOST = '192.168.10.40'
PORT = 3000
# create stream TCP socket
try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error:
        print 'Failed to create socket'
        sys.exit()
print "socket created"
# bind socket to locol HOST and PORT
try :
        s.bind((HOST,PORT))
except socket.error :
        print “Bind failed Error”
print “Bind complete”
#start listening on socket
s.listen(10)
print “Socket now listening”
#wait for income connection
s_conn, addr = s.accept()
print “Income connection with ” + addr[0] + ':' + str(addr[1])
```

รูปที่ 3.37 การสร้าง TCP Socket สำหรับการใช้งานเป็น TCP Server

3) การทำงานในส่วน Raspberry Pi 2 ใช้ Port Ethernet สองพอร์ตสำหรับการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และคอมพิวเตอร์ Bob เริ่มต้นการทำงานด้วยการสร้าง TCP Socket (s) สำหรับการใช้งานเป็น TCP Server ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และสร้าง TCP Socket (s1) สำหรับการใช้งานเป็น TCP Server จากนั้นรอการเชื่อมต่อจากคอมพิวเตอร์ Bob และวนลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 โดยใช้คำสั่ง s_conn.recv(1024) ตัวแปร s_conn คือ ค่าของ Socket ที่ได้เมื่อ TCP Server ทำคำสั่ง accept() นำข้อมูลผ่านกระบวนการถอดรหัส และส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Bob โดยใช้คำสั่ง s2.send(ztext) โค้ดในส่วนการทำงานของอุปกรณ์ Raspberry Pi 2 แสดงดังรูปที่ 3.38

```
Import socket, sys
# create TCP socket to communicate with Alice
s = TCPsocket_server()
# create TCP socket to communicate with RPi 2
s1 = TCPsocket_client()
#accept Bob  and RPi1 connection
s_dst, addr = s.accept()
s_conn, addr1 = s.accept()
while True:
        # receive data from RPi 1
        dataP = s_conn.recv(dataP)
        ### start process decryted data ###
        yrint=text_to_int(yrtext)
        yrint=[0, 0]+yrint
        yrfloat=int_to_float(yrint)
        zfloat=dechaos(yrfloat)
        zint=float_to_int(zfloat)
        ztext=int_to_text(zint)  # decrypted data
        s_dst.send(ztext) # send decrypted data to Bob
```


รูปที่ 3.38 การทำงานของ Raspberry Pi 2 ในระบบสื่อสารทางเดียวโดยใช้โปรโตคอล TCP

4) การทำงานในส่วนคอมพิวเตอร์ Bob เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล TCP จากนั้นเชื่อมต่อไปยังอุปกรณ์ Raspberry Pi 2 ซึ่งทำหน้าที่เป็น TCP Server โดยใช้คำสั่ง s.connect() วนลูปการทำงานรับข้อมูลที่ผ่านการถอดรหัสจากอุปกรณ์ Raspberry Pi 2 โดยใช้คำสั่ง s.recv(1024) และแสดงข้อมูลที่รับได้ โค้ดการทำงานในส่วนการสร้างการเชื่อมต่อ และลูปการทำงานแสดงดังรูปที่ 3.39

```
HOST = '192.168.10.50'
PORT = 7000
# connect to RPi 2 server
try :
        s.connect((HOST , PORT))
except socket.error:
        print "Connect to destination Error"
        sys.exit()
print "Connect to PRi2 server ok"
while True:
        ### receive data from RPi 2
        dataP = s.recv(1024)
        print 'Data Form A :' + dataP
```

รูปที่ 3.39 การทำงานของคอมพิวเตอร์ Bob ของระบบสื่อสารทางเดียวโดยใช้โปรโตคอล TCP

3.1.5.3 การทำงานของระบบสื่อสารรูปแบบ Full duplex โดยใช้โปรโตคอล UDP

ระบบสื่อสารรูปแบบ Full duplex ผู้เริ่มการเชื่อมต่อการสื่อสารสามารถเป็นได้ทั้งคอมพิวเตอร์ Alice และ Bob

1) กำหนดให้คอมพิวเตอร์ Alice ทำงานเป็น UDP Client เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล UDP จากนั้นสร้างการเชื่อมต่อกับอุปกรณ์ Raspberry Pi 1 ซึ่งทำงานเป็น UDP Server และสร้างการทำงานแบบคู่ขนานระหว่างลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 และลูปส่งข้อมูล โค้ดการทำงานของหลักในส่วนของคอมพิวเตอร์ Alice แสดงดังรูปที่ 3.40

```
import socket, sys
from thread import *
#main function
HOST = '192.168.10.40'
PORT = 3000
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.settimeout(2)
# create connection
msg= "..Hallo.."
s.sendto(msg, (HOST, PORT))
# receive data from RPi 1
start_new_thread(recvFmPi, (s,))
while True:
    # send data
    send2Pi(s,HOST,PORT,)
```

รูปที่ 3.40 การทำงานของคอมพิวเตอร์ Alice ในระบบสื่อสารแบบ Full duplex โดยใช้โปรโตคอล UDP

จากรูปที่ 3.40 การสร้างการทำงานแบบคู่ขนานใช้โมดูล Thread และใช้คำสั่ง start_new_thread(Thread_name ,(arg)) ในการกำหนดให้ทำงานคู่แบบขนาน โดยตัวแปร Thread_name คือชื่อของฟังก์ชันที่ต้องการให้ทำงาน และ arg คือค่าที่ต้องป้อนให้ฟังก์ชัน การทำงานในส่วนของลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 และการส่งข้อมูล เป็นการเรียกโปรแกรมย่อยที่เขียนไว้มาใช้ แสดงดังรูปที่ 3.41 และ 3.42 ตามลำดับ

```
def recvFmPi(sock):
    while True:
        data=sock.recvfrom(1024)
        sock.settimeout(30)
        if not data:
            print 'Connection closed'
            sys.exit()
        else :
            #print data
            dataP=data[0]+'\n'
            sys.stdout.write(dataP)
```

รูปที่ 3.41 การทำงานของลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1

```
def send2Pi(sock,host_p,port_p):
    msg = sys.stdin.readline()
    sock.sendto(msg,(host_p , port_p))
```

รูปที่ 3.42 การทำงานของโปรแกรมย่อยส่งข้อมูลจากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

จากรูปที่ 3.41 คำสั่ง s.settimeout ใช้ในการกำหนดเวลาในการทำคำสั่งรับข้อมูล หากไม่มีการรับข้อมูลในเวลาที่กำหนด โปรแกรมจะหยุดการทำงานโดยอัตโนมัติ และคำสั่ง sys.stdout.write(data) ใช้ในการแสดงข้อมูลบนหน้าต่าง command line และจากรูปที่ 3.42 คำสั่ง sys.stdin.readline() ใช้สำหรับการรับข้อมูลที่ผู้ใช้พิมพ์บนหน้าต่าง command line

2) การทำงานในส่วน Raspberry Pi 1 ใช้ port Ethernet สองพอร์ตสำหรับการสื่อสารกับคอมพิวเตอร์ Alice และอุปกรณ์ Raspberry Pi 2 เริ่มต้นการทำงานด้วยการใช้โปรแกรมย่อย (def) สำหรับสร้าง UDP Socket (s) เพื่อใช้งานเป็น UDP Server ในการสื่อสารกับคอมพิวเตอร์ Alice และใช้โปรแกรมย่อยสำหรับสร้าง UDP Socket (s1) เพื่อใช้งานเป็น

UDP Client ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 จากนั้นสร้างการทำงานแบบคู่ขนานระหว่างการรับข้อมูลจากคอมพิวเตอร์ Alice แล้วส่งไปยังอุปกรณ์ Raspberry Pi 2 กับการรับข้อมูลจากอุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Alice โดยเขียนอยู่ในรูปโปรแกรมย่อย โค้ดการทำงานหลักในส่วน Raspberry Pi 1 แสดงดังรูปที่ 3.43

```
Import socket, sys
from thread import *
# main function
if __name__ == "__main__":
        s_comA, addrA = comwithA()
        s_comPi, addrPi = comwithPi()
        #receive Fm RPi2 and send 2 Alice
        start_new_thread(dst2src ,(s_comA,s_comPi,addrA))
        #receive Fm Alice and send 2 RPi 2
        src2dst(s_comA,s_comPi,addrPi)
```

รูปที่ 3.43 การทำงานของ Raspberry Pi 1 ในระบบสื่อสารแบบ Full duplex โดยใช้โปรโตคอล UDP

จากรูปที่ 3.43 โปรแกรมย่อย comwithA() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง UDP Socket และการเชื่อม Socket เข้ากับหมายเลขไอพี และหมายเลขเลขพอร์ตของอุปกรณ์ Raspberry Pi 1 เพื่อใช้งานเป็น UDP Server จากนั้นทำคำสั่งสำหรับรับการเชื่อมต่อจากคอมพิวเตอร์ Alice และการส่งข้อความตอบรับไปยังคอมพิวเตอร์ Alice โค้ดการทำงานของโปรแกรมย่อย comwithA() แสดงดังรูปที่ 3.44

โปรแกรมย่อย comwithPi() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง UDP Socket เพื่อใช้งานเป็น UDP Client จากนั้นสร้างการเชื่อมต่อไปยังอุปกรณ์ Raspberry Pi 2 โค้ดการทำงานของโปรแกรมย่อย comwithPi() แสดงดังรูปที่ 3.45

```
def comwithA():
        # create UDP socket
        UDPsocket_server()
        # receive connection Fm A
        dC = s.recvfrom(1024)
        dataC = dC[0]
        print ' Connection received '
        # acknowledge to A
        ack = "Hello A"
        s.sendto(ack , addrC)
        return s, addrC
```

รูปที่ 3.44 การทำงานของโปรแกรมย่อย comwithA()

```
def comwithPi():
        host2 = '192.168.11.55'  #ip RPi2
        port2 = 4000  #port RPi2
        # create UDP socket
        UDPsocket_client()
        # create connection to RPi2
        connect = "RPi is online"
        s2.sendto(connect, (host2 , port2))
        addrP = (host2, port2)
        return s2, addrP
```

รูปที่ 3.45 การทำงานของโปรแกรมย่อย comwithPi()

จากรูปที่ 3.43 โปรแกรมย่อย dst2src รับค่าตัวแปรอาร์กิวเมนต์ 3 ตัว ประกอบด้วย Socket ที่ใช้ในการสื่อสารกับคอมพิวเตอร์ Alice รวมทั้ง Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และตัวแปรที่เก็บหมายเลขไอพีและหมายเลขพอร์ตของคอมพิวเตอร์ Alice การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลที่ผ่านการเข้ารหัสจากอุปกรณ์ Raspberry Pi 2 จากนั้นนำข้อมูลผ่านกระบวนการถอดรหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Alice โค้ดการทำงานของโปรแกรมย่อย dst2src() แสดงดังรูปที่ 3.46

```
def dst2src(src_conn,dst_conn,addr_A):
    while True:
        #receive from RPi 2
        dP = dst_conn.recvfrom(1024)
        dataP=dP[0]
        # decrypted data process
        output = decrypted(dataP)
        # send decrypted data to Alice
        src_conn.sendto(output , addr_A)
```

รูปที่ 3.46 การทำงานของโปรแกรมย่อย dst2src ในส่วนของ Raspberry Pi 1 โดยใช้โปรโตคอล UDP

จากรูปที่ 3.43 โปรแกรมย่อย src2dst รับค่าตัวแปรอาร์กิวเมนต์ 3 ตัว ประกอบด้วย Socket ที่ใช้ในการสื่อสารกับคอมพิวเตอร์ Alice รวมทั้ง Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 และตัวแปรที่เก็บหมายเลขไอพีและหมายเลขพอร์ตของอุปกรณ์ Raspberry Pi 2 การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลจากคอมพิวเตอร์ Alice จากนั้นนำข้อมูลผ่านกระบวนการเข้ารหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการเข้ารหัสให้กับอุปกรณ์ Raspberry Pi 2 โค้ดการทำงานของโปรแกรมย่อย src2dst() แสดงดังรูปที่ 3.47

```
def src2dst(src_conn,dst_conn,addr_Pi):
    while True:
        #receive from Alice
        dA = src_conn.recvfrom(1024)
        dataA = dA[0]
        # encrypted data process
        output = encrypted(dataA)
        # send encrypted data to PPi 2
        dst_conn.sendto(output , addr_Pi)
```

รูปที่ 3.47 การทำงานของโปรแกรมย่อย src2dst ในส่วนของ Raspberry Pi 1 โดยใช้โปรโตคอล UDP

3) การทำงานในส่วน Raspberry Pi 2 ใช้ Port Ethernet สองพอร์ตสำหรับการสื่อสารกับอุปกรณ์ Raspberry Pi 1 และคอมพิวเตอร์ Bob เริ่มต้นการทำงานด้วยการใช้โปรแกรมย่อยสำหรับสร้าง UDP Socket (s) เพื่อใช้งานเป็น UDP Server ในการสื่อสารกับ B และใช้โปรแกรมย่อยสำหรับสร้าง UDP Socket (s1) เพื่อใช้งานเป็น UDP Server ในการสื่อสารกับอุปกรณ์ Raspberry Pi 1 จากนั้นรอรับการเชื่อมต่อจากคอมพิวเตอร์ Bob และอุปกรณ์ Raspberry Pi 1 ตามลำดับและสร้างการทำงานแบบคู่ขนานของลูปรับข้อมูลจากคอมพิวเตอร์ Bob แล้วส่งไปยังอุปกรณ์ Raspberry Pi 1 กับลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 แล้วส่งไปยังคอมพิวเตอร์ Bob โดยเขียนอยู่ในรูปโปรแกรมย่อย โค้ดการทำงานหลักในส่วน Raspberry Pi 2 แสดงดังรูปที่ 3.48

```
Import socket, sys
from thread import *
# main function
if __name__ == "__main__":
        #create UDP socket
        s_comPi = comwithPi()
        s_comB = comwithB()
        #receive connection
        addr_Pi, addr_B = recvconnection(s_comPi,s_comB)
        #receive Fm RPi 1 and send 2 Bob
        start_new_thread(src2dst ,(s_comPi,s_comB,addr_B))
        #receive Fm Bob and send 2 RPi 1
        dst2src(s_comPi,s_comB,addr_Pi)
```

รูปที่ 3.48 การทำงานของ Raspberry Pi 2 ในระบบสื่อสารแบบ Full duplex โดยใช้โปรโตคอล UDP

จากรูปที่ 3.48 โปรแกรมย่อย comwithPi() และ comwithB() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง UDP Socket และการเชื่อม Socket เข้ากับหมายเลขไอพีและหมายเลขเลขพอร์ตของอุปกรณ์ Raspberry Pi 2 เพื่อใช้งานเป็น UDP Server แต่แตกต่างกันที่หมายเลขไอพีและหมายเลขพอร์ต

โปรแกรมย่อย recvconnection() รับค่า Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 1 และ Socket ที่ใช้ในการสื่อสารกับคอมพิวเตอร์ Bob โดยการทำงานภายในโปรแกรมเริ่มจากรอรับการเชื่อมต่อจากคอมพิวเตอร์ Bob และการส่งข้อความตอบรับไปยังคอมพิวเตอร์ Bob จากนั้นรอรับการเชื่อมต่อจากอุปกรณ์ Raspberry Pi 1 โค้ดการทำงานของโปรแกรมย่อย recvconnection() แสดงดังรูปที่ 3.49

```
def recvconnection(s_Pi,s_B):
        # recvconnection from Bob
        dB = s_B.recvfrom(1024)
        dataB = dB[0]
        addrB = dB[1]
        print "connection from B received"
        # acknowledge to Bob
        ack = "Hello B"
        s_B.sendto(ack, addrB)
        # recvconnection from Pi
        dP = s_Pi.recvfrom(1024)
        dataP = dP[0]
        addrP = dP[1]
        print "connection from RPi1 received"
        return addrP, addrB
```

รูปที่ 3.49 การทำงานของโปรแกรมย่อย recvconnection()

จากรูปที่ 3.48 โปรแกรมย่อย src2dst รับค่าตัวแปรอาร์กิวเมนต์ 3 ตัว ประกอบด้วย Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 1 รวมทั้ง Socket ที่ใช้ในการสื่อสารกับคอมพิวเตอร์ Bob และตัวแปรที่เก็บหมายเลขไอพีและหมายเลขพอร์ตของคอมพิวเตอร์ Bob การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลที่ผ่านการเข้ารหัสจากอุปกรณ์ Raspberry Pi 1 จากนั้นนำข้อมูลผ่านกระบวนการถอดรหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Bob โค้ดการทำงานของโปรแกรมย่อย src2dst() แสดงดังรูปที่ 3.50

```
def src2dst(src_conn,dst_conn,addr_Be):
    while True:
        #Receive from RPi 2
        dPI = src_conn.recvfrom(1024)
        dataPI=dPI[0]
        # decrypted data process
        output = decrypted(dataPI)
        # send decrypted data to Bob
        dst_conn.sendto(output , addr_Be)
```

รูปที่ 3.50 การทำงานของโปรแกรมย่อย src2dst ในส่วนของ Raspberry Pi 2 โดยใช้โปรโตคอล UDP

จากรูปที่ 3.48 โปรแกรมย่อย dst2src รับค่าตัวแปรอาร์กิวเมนต์ 3 ตัว ประกอบด้วย Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 1รวมทั้ง Socket ที่ใช้ในการสื่อสารกับคอมพิวเตอร์ Bob และตัวแปรที่เก็บหมายเลขไอพีและหมายเลขพอร์ตของอุปกรณ์ Raspberry Pi 1 การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลจากคอมพิวเตอร์ Bob จากนั้นนำข้อมูลผ่านกระบวนการเข้ารหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการเข้ารหัสให้กับอุปกรณ์ Raspberry Pi 1 โค้ดการทำงานของโปรแกรมย่อย dst2src() แสดงดังรูปที่ 3.51

```
def dst2src(src_conn,dst_conn,addr_P):
    while True:
        #Receive from Bob
        dB = dst_conn.recvfrom(1024)
        dataB = dB[0]
        # encrypted data process
        output = encrypted(dataB)
        # send encrypted data to PPi 1
        src_conn.sendto(output , addr_P)
```


รูปที่ 3.51 การทำงานของโปรแกรมย่อย dst2src ในส่วนของ Raspberry Pi 2 โดยใช้โปรโตคอล UDP

4) การทำงานในส่วนคอมพิวเตอร์ Bob กำหนดให้ทำงานเช่นเดียวกับในส่วนคอมพิวเตอร์ Alice คือ ทำงานเป็น UDP Client เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล UDP จากนั้นสร้างการเชื่อมต่อกับอุปกรณ์Raspberry Pi 2 ซึ่งทำงานเป็น UDP Server และสร้างการทำงานแบบคู่ขนานระหว่างลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 2 และลูปส่งข้อมูล

3.1.5.4 การทำงานของระบบสื่อสารรูปแบบ Full duplex โดยใช้โปรโตคอล TCP

ระบบสื่อสารรูปแบบ Full duplex ผู้เริ่มการเชื่อมต่อการสื่อสารสามารถเป็นได้ทั้งคอมพิวเตอร์ Alice และ Bob

1) กำหนดให้คอมพิวเตอร์ Alice ทำงานเป็น TCP Client เริ่มต้นด้วยการสร้าง Socket สำหรับการสื่อสารข้อมูลด้วยโปรโตคอล TCP จากนั้นเชื่อมต่อกับอุปกรณ์ Raspberry Pi 1 ซึ่งทำงานเป็น TCP Server โดยใช้คำสั่ง s.connect((HOST , PORT)) และสร้างลูปการทำงานโดยเก็บค่า Socket 2 ชนิด ในตัวแปร Socket_list ได้แก่ sys.stdin คือ Socket สำหรับสื่อสารกับผู้ใช้ผ่านหน้าต่าง command line และ TCP Socket จากนั้นใช้คำสั่งในโมดูล select สำหรับรอการทำงานของ Socket ทั้งสอง โดยถ้า TCP Socket พร้อมใช้งาน คือ มีข้อมูลส่งมาจาก TCP Server จะทำการแสดงข้อมูลที่รับได้โดยใช้คำสั่ง sys.stdout.write(data) ในทางกลับกันหาก sys.stdin พร้อมใช้งาน คือ มีการพิมพ์ข้อมูลจากผู้ใช้บนหน้าต่าง command line แล้วกดปุ่ม enter จะทำการอ่านข้อมูลในบรรทัดนั้นโดยใช้คำสั่ง sys.stdin.readline() แล้วส่งไปยัง TCP Server โดยใช้คำสั่ง s.send(data) โค้ดการทำงานหลักในส่วนของคอมพิวเตอร์ Alice แสดงดังรูปที่ 3.52

```
import socket, select , sys
if __name__ == "__main__":
        # create TCP socket
        TCPsocket_client()
        while True:
                socket_list = [sys.stdin, s]
                # Get the list sockets which are readable
                read_sockets, write_sockets, error_sockets =select.select(socket_list , [], [])
                for sock in read_sockets:
                        if sock == s:
                                data = sock.recv(1024)
                                if not data :
                                        print 'Connection closed'
                                        sys.exit()
                                else :
                                        #print data
                                        sys.stdout.write(data)
                        else :
                                #user entered a message
                                msg = sys.stdin.readline()
                                s.send(msg)
```

รูปที่ 3.52 การทำงานของคอมพิวเตอร์ Alice ในระบบสื่อสารรูปแบบ Full duplex โดยใช้โปรโตคอล TCP

2) การทำงานในส่วน Raspberry Pi 1 ใช้ port Ethernet สองพอร์ตสำหรับการสื่อสารกับคอมพิวเตอร์ Alice และอุปกรณ์ Raspberry Pi 1 เริ่มต้นการทำงานด้วยการใช้โปรแกรมย่อยสำหรับสร้าง TCP Socket (s) เพื่อใช้งานเป็น TCP Server ในการสื่อสารกับ A จากนั้นวนลูปทำคำสั่งยอมรับการเชื่อมต่อ(s.accept()) เมื่อคอมพิวเตอร์ Alice ทำการเชื่อมต่อเข้ามายัง TCP Server แล้วใช้โปรแกรมย่อยสำหรับสร้าง TCP Socket (s1) เพื่อใช้งานเป็น

TCP Client ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 ในขั้นตอนสุดท้ายสร้างการทำงานแบบคู่ขนานของลูปรับข้อมูลจากคอมพิวเตอร์ Alice แล้วส่งไปยังอุปกรณ์ Raspberry Pi 2 กับลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Alice โดยเขียนอยู่ในรูปโปรแกรมย่อย โค้ดการทำงานหลักในส่วน Raspberry Pi 1 แสดงดังรูปที่ 3.53

```
import socket, sys
from thread import *
# main function
if __name__ == "__main__":
        s_comA = comwithA()
        while True:
                s_src, addr = s_comA.accept()
                print 'Income connection with ' + addr[0] + ':' + str(addr[1])
                s_dst = comwithPi()
                #receive Fm RPi2 and send 2 Alice
                start_new_thread(dst2src ,(s_src,s_dst,))
                #receive Fm Alice and send 2 RPi2
                src2dst(s_src,s_dst)
        s_comA.close()
```

รูปที่ 3.53 การทำงานของ Raspberry Pi 1 ในระบบสื่อสารแบบ Full duplex โดยใช้โปรโตคอล TCP

จากรูปที่ 3.53 โปรแกรมย่อย comwithA() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง TCP Socket การเชื่อม Socket เข้ากับหมายเลขไอพี และหมายเลขเลขพอร์ตของอุปกรณ์ Raspberry Pi 1 และคำสั่งรอการเชื่อมต่อจาก TCP Client (s.listen())

โปรแกรมย่อย comwithPi() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง TCP Socket การเชื่อมต่อไปยังอุปกรณ์ Raspberry Pi 2 โดยใช้คำสั่ง s.connect()

โปรแกรมย่อย dst2src รับค่าตัวแปรอาร์กิวเมนต์ 2 ตัว คือ Socket ที่ได้จากการทำคำสั่งยอมรับการเชื่อมต่อของคอมพิวเตอร์ Alice (s_src) และค่า Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 (s_dst) การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลที่ผ่านการเข้ารหัสจากอุปกรณ์ Raspberry Pi 2 จากนั้นนำข้อมูลผ่านกระบวนการถอดรหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Alice โค้ดการทำงานของโปรแกรมย่อย dst2src() แสดงดังรูปที่ 3.54

```
def dst2src(src_conn,dst_conn,):
    while True:
        #Receive from Alice
        dataA = src_conn.recv(1024)
        # decrypted data process
        output = decrypted(dataA)
        # send decrypted data to RPi2
        dst_conn.send(output)
```

รูปที่ 3.54 การทำงานของโปรแกรมย่อย dst2src ในส่วนของ Raspberry Pi 1 โดยใช้โปรโตคอล TCP

จากรูปที่ 3.53 โปรแกรมย่อย src2dst รับค่าตัวแปรอาร์กิวเมนต์ 2 ตัว คือ Socket ที่ได้จากการทำคำสั่งยอมรับการเชื่อมต่อของคอมพิวเตอร์ Alice (s_src) และค่า Socket ที่ใช้ในการสื่อสารกับอุปกรณ์ Raspberry Pi 2 (s_dst) การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลจากคอมพิวเตอร์ Alice จากนั้นนำข้อมูลผ่านกระบวนการเข้ารหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการเข้ารหัสให้กับอุปกรณ์ Raspberry Pi 2 โค้ดการทำงานของโปรแกรมย่อย src2dst() แสดงดังรูปที่ 3.55

```
def src2dst (src_conn,dst_conn,):
        while True:
                #Receive from RPi2
                dataP = dst_conn.recv(1024)
                # encrypted data process
                output = encrypted(dataP)
                # send encrypted data to Alice
                src_conn.send(output)
```

รูปที่ 3.55 การทำงานของโปรแกรมย่อย src2dst ในส่วนของ Raspberry Pi 1 โดยใช้โปรโตคอล TCP

```
import socket, sys
from thread import *
# main function
if __name__ == "__main__":
        s_comPi = comwithPi()
        s_comB = comwithB()
        while True:
                s_dst = acceptB(s_comB)
                s_src = acceptPi(s_comPi)
                #receive Fm RPi1 and send 2 Bob
                start_new_thread(src2dst ,(s_src,s_dst,))
                #receive Fm Bob and send 2 RPi1
                dst2src(s_src,s_dst,)
        s_comPi.close()
        s_comB.close()
```

รูปที่ 3.56 การทำงานของ Raspberry Pi 2 ในระบบสื่อสารแบบ Full duplex โดยใช้โปรโตคอล TCP

3) จากรูปที่ 3.56 การทำงานในส่วน Raspberry Pi 2 ใช้ port Ethernet สองพอร์ตสำหรับการสื่อสารกับคอมพิวเตอร์ Bob และอุปกรณ์ Raspberry Pi 1 เริ่มต้นการทำงานด้วยการใช้โปรแกรมย่อยสำหรับสร้าง TCP Socket (s) เพื่อใช้งานเป็น TCP Server ในการสื่อสารกับคอมพิวเตอร์ Bob และใช้โปรแกรมย่อยสำหรับสร้าง TCP Socket (s1) เพื่อใช้งานเป็น TCP Server ในการสื่อสารกับอุปกรณ์ Raspberry Pi 1 จากนั้นวนลูปทำคำสั่งยอมรับการเชื่อมต่อจากคอมพิวเตอร์ Bob และอุปกรณ์ Raspberry Pi 1 ในขั้นตอนสุดท้ายสร้างการทำงานแบบคู่ขนานของลูปรับข้อมูลจากอุปกรณ์ Raspberry Pi 1 แล้วส่งไปยัง กับลูปรับข้อมูลจากอุปกรณ์คอมพิวเตอร์ Bob แล้วส่งไปยังอุปกรณ์ Raspberry Pi 1 โดยเขียนอยู่ในรูปโปรแกรมย่อย

โปรแกรมย่อย comwithPi() และ comwithB() ประกอบไปด้วยคำสั่งที่ใช้ในการสร้าง TCP Socket การเชื่อม Socket เข้ากับหมายเลขไอพี และหมายเลขเลขพอร์ตของอุปกรณ์ Raspberry Pi 1 และคำสั่งรอการเชื่อมต่อจาก TCP Client (s.listen())

โปรแกรมย่อย src2dst รับค่าตัวแปรอาร์กิวเมนต์ 2 ตัว คือ Socket จากการทำคำสั่งยอมรับการเชื่อมต่อของอุปกรณ์ Raspberry Pi 1 (s_src) และค่า Socket จากการทำคำสั่งยอมรับการเชื่อมต่อของคอมพิวเตอร์ Bob (s_dst) การทำงานของโปรแกรมเป็นลูปรอรับข้อมูลที่ผ่านการเข้ารหัสจากอุปกรณ์ Raspberry Pi 1 จากนั้นนำข้อมูลผ่านกระบวนการถอดรหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการถอดรหัสไปยังคอมพิวเตอร์ Bob โค้ดการทำงานของโปรแกรมย่อย src2dst() แสดงดังรูปที่ 3.57

```
def src2dst (src_conn,dst_conn,):
    while True:
        #Receive from RPi1
        dataP = src_conn.recv(1024)
        # decrypted data process
        output = decrypted(dataP)
        # send decrypted data to Bob
        dst_conn.send(output)
```

รูปที่ 3.57 การทำงานของโปรแกรมย่อย src2dst ในส่วนของ Raspberry Pi 2 โดยใช้โปรโตคอล TCP

จากรูปที่ 3.56 โปรแกรมย่อย dst2src รับค่าตัวแปรอาร์กิวเมนต์ 2 ตัว คือ Socket จากการทำคำสั่งยอมรับการเชื่อมต่อของอุปกรณ์ Raspberry Pi 1 (s_src) และค่า Socket จากการทำคำสั่งยอมรับการเชื่อมต่อของคอมพิวเตอร์ Bob (s_dst) ภายในโปรแกรมย่อยเป็นลูปรอรับข้อมูลจากคอมพิวเตอร์ Bob จากนั้นนำข้อมูลผ่านกระบวนการเข้ารหัสโดยใช้โปรแกรมย่อย และส่งข้อมูลที่ผ่านการเข้ารหัสไปยังอุปกรณ์ Raspberry Pi 1 โค้ดการทำงานของโปรแกรมย่อย dst2src() แสดงดังรูปที่ 3.58

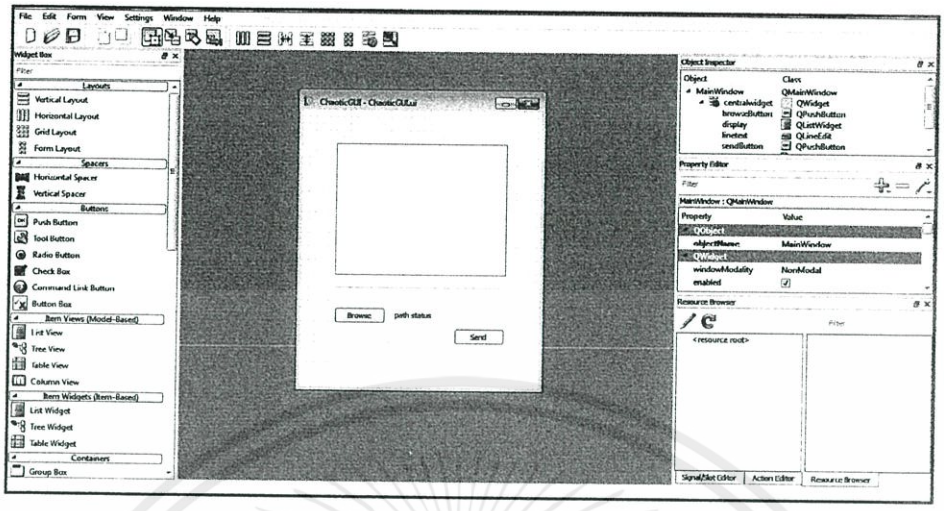
```
def dst2src (src_conn,dst_conn,):
    while True:
        #Receive from Bob
        dataB = dst_conn.recv(1024)
        # encrypted data process
        output = encrypted(dataB)
        # send decrypted data to RPi1
        src_conn.send(output)
```

รูปที่ 3.58 การทำงานของโปรแกรมย่อย dst2src ในส่วนของ Raspberry Pi 2 โดยใช้โปรโตคอล TCP

4) การทำงานในส่วนคอมพิวเตอร์ Bob กำหนดให้ทำงานเช่นเดียวกับในส่วนคอมพิวเตอร์ Alice คือ ทำงานเป็น TCP Client สำหรับสื่อสารกับอุปกรณ์ Raspberry Pi 2 ซึ่งทำงานเป็น TCP Server

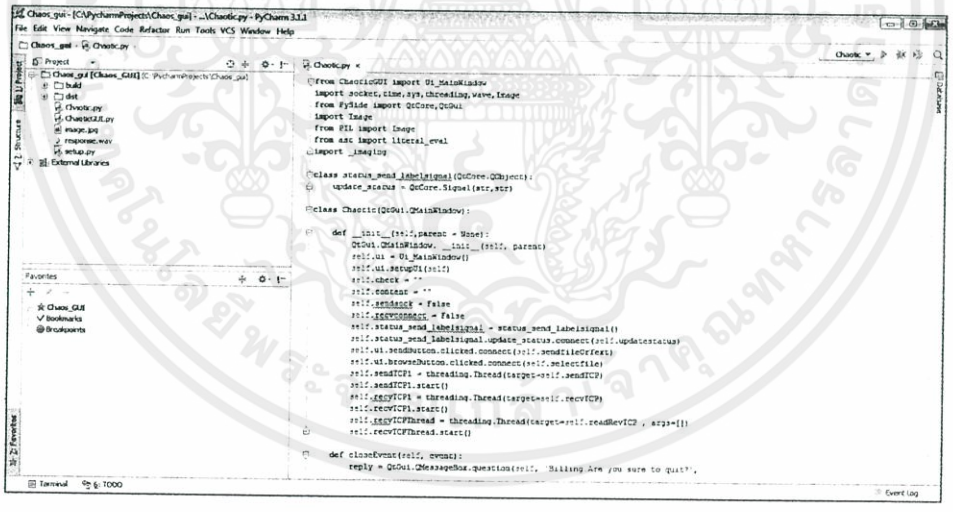
3.1.6 การออกแบบ GUI (Graphic User Interface)

ในส่วนของการออกแบบ GUI จะเริ่มต้นด้วยการออกแบบส่วนของหน้าต่างการใช้งานที่จะถูกนำมาใช้ในการส่งข้อความ (Text) ข้อมูลเสียง (Sound data) และข้อมูลภาพ (Image data) โดยใช้โปรแกรม Qt Designer ในการออกแบบหน้าต่างการใช้งานแสดงดังรูปที่ 3.59



รูปที่ 3.59 การออกแบบหน้าต่าง Interface ที่ใช้ในการรับ – ส่งข้อมูล

หลังจากได้ออกแบบหน้าต่างของ GUI เรียบร้อยเราจะได้เป็นไฟล์ ChaoticGUI.ui เราจะนำไฟล์สกุล .ui มาแปลงเป็นไฟล์สกุล .py แสดงดังรูปที่ 3.60 เพื่อนำหน้าต่างของ GUI มาเขียนคำสั่งให้ทำงานตามที่เราต้องการต่อไป



รูปที่ 3.60 ไฟล์สกุล .py ที่ได้จากการแปลงจากไฟล์สกุล .ui

3.1.6.1 การออกแบบการทำงานให้กับ GUI

การทำงานของ GUI จะแบ่งออกเป็น 3 ส่วน คือ การทำงานในส่วนของการสร้าง Socket การทำงานก่อนส่งข้อมูล และการทำงานเมื่อได้รับข้อมูล

1) การทำงานในส่วนของการสร้าง Socket ประกอบด้วยการสร้าง Send socket และ Receive socket โดยทั้ง 2 ส่วนจะถูกสร้างอยู่ภายใต้ GUI ในการสร้าง Send socket เขียนอยู่ในรูปโปรแกรมย่อย sendTCP(self) การทำงานของโปรแกรมเริ่มต้นด้วยการกำหนดตัวแปร con ให้มีค่าเป็นโลจิก 0 (False) แล้วสร้าง TCP Socket สำหรับการใช้งานเป็น TCP client โดยเก็บค่า Socket ในตัวแปร self.sendsock ใช้คำสั่ง try กับ except ในการทำคำสั่งเชื่อมต่อไปกับ TCP server ที่มีหมายเลขไอพี '192.168.10.50' และหมายเลขพอร์ต 9200 ซึ่งเป็นหมายเลขที่กำหนดไว้ในการทดลอง และเปลี่ยนค่าโลจิกที่เก็บในตัวแปร con เป็นโลจิก 1 (True) แต่ถ้าเกิด Error ให้แสดงข้อความเพื่อบอกผู้ใช้ว่าการเชื่อมต่อล้มเหลว และแสดงข้อความ " Send reconnection" บนหน้าต่าง GUI ก่อนการทำคำสั่ง self. sendTCP() เพื่อทำการเชื่อมต่อใหม่อีกครั้ง และทำการเช็คเงื่อนไขโดยใช้ค่าโลจิกของตัวแปร con ถ้าโลจิกของตัวแปร con เป็นโลจิก 1 แสดงข้อความ "Send connected" เพื่อแสดงว่า Send socket สามารถใช้งานได้ โค้ดการสร้าง Send socket แสดงคำสั่งทั้งหมดดังรูปที่ 3.61

```
def sendTCP(self):
    con = False
    self.sendsock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    try:
        self.sendsock.connect(("192.168.10.50",9200))
        con = True
    except socket.error, msg:
        print ('connect() failed with code '+ str(msg[0])+":"+msg[1])
        time.sleep(1)
        self.ui.display.addItem("Send reconnection" )
        self.sendTCP()
    if con:
        print "Send connected"
```

รูปที่ 3.61 การสร้าง Send socket

ในส่วนของการสร้าง Receive socket เขียนอยู่ในรูปโปรแกรมย่อย recvTCP(self) การทำงานของโปรแกรมเริ่มต้นด้วยการสร้าง TCP Socket สำหรับการใช้งานเป็น TCP server โดยเก็บค่า Socket ในตัวแปร self.recvsock ทำการเชื่อม Socket กับหมายเลขไอพีของเครื่องและหมายเลขพอร์ตโดยใช้คำสั่ง self.revsock.bind(("192.168.10.60",9200)) จากนั้นสร้างการทำงานแบบคู่ขนานของโปรแกรมย่อย recvrunsocket(self) ซึ่งภายในโปรแกรมใช้

คำสั่ง try กับ except ในการทำคำสั่งรอรับการเชื่อมต่อของ TCP client (self.recvsock.listen()) และทำคำสั่งยอมรับการเชื่อมตัวโดยเก็บค่า socket หลังจากยอมรับการเชื่อมต่อในตัวแปร self.recvconnect และแสดงข้อความ “recv connected” เพื่อแสดงว่า Receive socket สามารถใช้งานได้ ถ้ามี Error เกิดขึ้นแสดงข้อความ “can’t accept connection” บนหน้าต่าง GUI โค้ดการสร้าง Receive socket แสดงคำสั่งทั้งหมดดังรูปที่ 3.62

```
def recvTCP(self):
    self.recvsock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.recvsock.bind(("192.168.10.60",9500))
    acceptsocket = threading.Thread(target= self.recvrunsocket())
    acceptsocket.start()

def recvrunsocket(self):
    try:
        self.recvsock.listen(10)
        self.recvconnect,self.recvaddress = self.recvsock.accept()
    except:
        self.ui.display.addItem("can't accept connection1" )
```

รูปที่ 3.62 การสร้าง Receive socket

2) การทำงานของ GUI ก่อนส่งข้อมูลออกไปยังฝั่งรับนั้นจะแบ่งได้เป็น 3 กรณี คือ ข้อมูลชนิดข้อความ (Text) ข้อมูลชนิดเสียง (Sound data) และข้อมูลชนิดภาพ (Image data) ในส่วนของการส่งข้อมูลทั้ง 3 จะมีการออกแบบคล้ายๆ กัน คือ จะมีการเติมข้อมูล Header เข้าไปเพื่อระบุประเภทของข้อมูลที่จะส่งออกไป ในกรณีของข้อความ ทำการเพิ่ม Header = 0 รวมเข้ากับข้อมูลที่ต้องการส่งซึ่งเก็บค่าไว้ในตัวแปร text จากการอ่านข้อความในช่องพิมพ์ข้อความบนหน้าต่าง GUI โดยใช้คำสั่ง self.ui.linetext.text() เก็บข้อมูลที่ทำการเพิ่ม Header แล้วในตัวแปร self.content แสดงข้อความที่ส่งบนหน้าต่าง GUI และสร้างการทำงานแบบคู่ขนานของโปรแกรมย่อยสำหรับส่งข้อมูล (sendthteadTCP) โค้ดการทำงานในส่วนของการจัดการข้อมูล Text ก่อนส่งข้อมูลออกไปยังฝั่งรับ แสดงคำสั่งทั้งหมดดังรูปที่ 3.63

```
def sendfileOrText(self):
    text = self.ui.linetext.text()
    #self.content = text
    if text:
        text1 = str([0,str(text)])
        self.content = text1
        self.ui.display.addItem("send : " +text)
        self.sendTCPThread = threading.Thread(target=self.sendthreadTCP , args=[])
        self.sendTCPThread.start()
        self.ui.linetext.clear()
```


รูปที่ 3.63 ฟังก์ชันในการส่งข้อมูล Text

การทำงานของโปรแกรมย่อย sendthreadTCP ประกอบไปด้วยคำสั่งแสดงค่าของข้อมูลที่มีการเพิ่ม Header ซึ่งเก็บค่าไว้ในตัวแปร self.content คำสั่งส่งข้อมูล self.sendsock.send(self.content) และคำสั่งล้างค่าในตัวแปร self.content ถ้ามี Socket ที่ใช้มี Error เกิดขึ้น ให้แสดงข้อความเพื่อบอกผู้ใช้ว่า Socket error โค้ดการทำงานของโปรแกรมย่อย sendthreadTCP แสดงคำสั่งทั้งหมดดังรูปที่ 3.64

```
def sendthreadTCP(self):
    try :
        print(self.content)
        self.sendsock.send(self.content)
        self.content = ""
    except socket.error, msg:
        print "socket error.Error code : " + str(msg[0])
```

รูปที่ 3.64 การทำงานของโปรแกรมย่อย sendthreadTCP(self)

ในกรณีของข้อมูลเสียงทำการเพิ่ม Header ซึ่งจะประกอบไปด้วย 1 ใช้บ่งบอกว่าเป็นข้อมูลเสียง จำนวนเฟรมของเสียง (self.content.getnframes) เฟรมเรท (self.content.getframerate) และแซมปวิดธ์ (self.content.getsampwidth) โดยเก็บค่าในตัวแปร final_content ซึ่งเป็นตัวแปรชนิดลิสต์ จากนั้นนำ Header มาต่อท้ายด้วยข้อมูลเสียง โดยทำการอ่านเฟรมของไฟล์เสียงจนครบทุกเฟรมโดยใช้คำสั่ง self.content.readframes(f) ซึ่ง f คือลำดับของเฟรม และเก็บข้อมูลในตัวแปรเดิม แล้วแปลงข้อมูลที่เก็บในตัวแปร final_content ให้เป็นข้อมูลชนิดสตริง โดยเก็บในตัวแปร self.content แสดงข้อมูลความ “send :” ตามด้วยชื่อของไฟล์เสียงบนหน้าต่าง GUI และสร้างการทำงานแบบคู่ขนานของโปรแกรมย่อยสำหรับส่งข้อมูล sendthreadTCP โค้ดการทำงานในส่วนของการจัดการข้อมูลเสียงก่อนส่งข้อมูลออกไปยังฝั่งรับแสดงคำสั่งทั้งหมดดังรูปที่ 3.65

```
def waveprocess(self):
    print "nframe : " + str(self.content.getnframes())
    print "framrate : " + str(self.content.getframerate())
    print "samp : " + str(self.content.getsampwidth())
    final_content = [1,self.content.getnframes(),self.content.getframerate(),self.content.getsampwidth()]
    for f in range(self.content.getnframes()):
        final_content.append(str(self.content.readframes(f)))
    self.ui.display.addItem("send : " +self.filename[31:])
    self.sendTCPThread = threading.Thread(target=self.sendthreadTCP , args=[])
    self.sendTCPThread.start()
```


รูปที่ 3.65 กระบวนการโปรเซสของข้อมูลเสียง

ในกรณีของข้อมูลภาพทำการเพิ่ม Header ซึ่งจะประกอบไปด้วย 2 ความกว้างของภาพ (self.content.size[0]) และความสูงของภาพ (self.content.size[1]) โดยเก็บค่าในตัวแปร final_content ซึ่งเป็นตัวแปรชนิดลิสต์ จากนั้นนำ Header มาต่อท้ายด้วยข้อมูลภาพ โดยทำการอ่านพิกเซล (Pixel) ของข้อมูลภาพด้วยคำสั่ง self.content. getpixel((x,y)) ซึ่ง x คือค่าความกว้างของภาพ และ y คือค่าความสูงของภาพ ตามลำดับ และนำพิกเซลของข้อมูลภาพที่มีความสูงเท่ากันแต่มีความกว้างต่างกันมาเรียงต่อกัน แล้วจึงนำพิกเซลที่ความสูงถัดไปมาเรียงต่อกันจนครบทุกพิกเซล โดยคั่นแต่ละความสูงด้วย \n ทำการแปลงข้อมูลที่เก็บในตัวแปร final_content เป็นข้อมูลชนิดสตริงส์ โดยเก็บในตัวแปร self.content แสดงข้อความ “send : ” ตามด้วยชื่อของไฟล์ภาพบนหน้าต่าง GUI และสร้างการทำงานแบบคู่ขนานของโปรแกรมย่อยสำหรับส่งข้อมูล sendthreadTCP โค้ดการทำงานในส่วนของการจัดการข้อมูลภาพก่อนส่งข้อมูลออกไปยังฝั่งรับแสดงคำสั่งทั้งหมดดังรูปที่ 3.66

```
def imageprocess(self):
    width = self.content.size[0]
    height = self.content.size[1]
    final_content = [2,width,height]
    for y in range(height):
        for x in range(width):
            final_content.append(self.content.getpixel((x,y)))
        final_content.append("\n")
    self.content = str(final_content)
    print str(final_content)
    self.ui.display.addItem("send : " +self.filename[31:])
    self.sendTCPThread = threading.Thread(target=self.sendthreadTCP , args=[])
    self.sendTCPThread.start()
```

รูปที่ 3.66 กระบวนการโปรเซสของข้อมูลภาพ

3) การทำงานของโปรแกรมย่อย readRevTCP เริ่มจากการรับข้อมูลด้วยคำสั่ง self.recvconnect.recv(1024*1024) และตรวจสอบว่าได้รับข้อมูลครบถ้วนด้วยคำสั่ง if self.stream.endswith(“]”) แล้วเก็บข้อมูลทั้งหมดไว้ในตัวแปร self.stream จากนั้นจึงทำการแปลงข้อมูลที่ได้รับจากตัวแปรชนิดสตริงส์กลับมาเป็นตัวแปรชนิดลิสต์และเก็บในตัวแปร listcontent แล้วทำการตรวจสอบชนิดของข้อมูลจากสมาชิกตำแหน่งที่ 0

เมื่อสมาชิกตำแหน่งที่ 0 ของตัวแปร listcontent มีค่าเป็น 0 จะแสดงว่าเป็นข้อมูลชนิดข้อความ จึงนำข้อมูลที่อยู่ในตัวแปร listcontent ตำแหน่งที่ 1 เก็บในตัวแปร self.stream จากนั้นนำตัวแปร self.stream หรือข้อความที่ได้รับไปแสดงบนหน้าต่าง GUI

ด้วยคำสั่ง self.ui.display.addItem("recv : " +str(self.stream)[:100]) เมื่อโปรแกรมย่อยนี้ทำงานเสร็จ แล้วก็จะล้างค่าในตัวแปร self.stream แสดงคำสั่งทั้งหมดดังรูปที่ 3.67

```
def readRevTCP(self):
    self.stream = ""
    while (True):
        if self.recvconnect != False:
            byte = self.recvconnect.recv(1024*1024)
            if byte:
                print(byte)
                self.stream += byte
            if self.stream.endswith("]"):
                print "recv : "+str(self.stream)[:100]
                print self.stream
                self.stream = str(self.stream)
                listcontent = literal_eval(self.stream)
                if listcontent[0]==0:
                    self.stream = listcontent[1]
                    self.ui.display.addItem("recv : "+str(self.stream)[:100])
                    self.stream = ""
```

รูปที่ 3.67 กระบวนการทำงานของฝั่งรับและการตรวจสอบการรับข้อมูลชนิดข้อความ

เมื่อสมาชิกตำแหน่งที่ 0 ของตัวแปร listcontent มีค่าเป็น 1 จะแสดงว่าเป็นข้อมูลเสียง ทำการสร้างไฟล์ด้วยคำสั่ง file=wave.open('recvsound.wav', 'wb') กำหนดจำนวนเฟรมของไฟล์เสียงจากสมาชิกตำแหน่งที่ 1 ของตัวแปร listcontent ด้วยคำสั่ง file.setnframes(listcontent[1]) ทำการเก็บข้อมูลเสียงจากสมาชิกตำแหน่งที่ 5 ถึงตำแหน่งสุดท้ายของตัวแปร listcontent ในตัวแปร listframe ด้วยคำสั่ง listframe+=frame แล้วกำหนดพารามิเตอร์ของไฟล์เสียงด้วยคำสั่ง file.setparams() ซึ่งประกอบด้วย 6 อากิวเมนต์ ได้แก่ nchannels, sampwidth, framerate, nframes, comptype และ compname โดยกำหนดค่าของแต่ละอากิวเมนต์ดังนี้ nchannels = 1, sampwidth=int(listcontent[3]), framerate=int(listcontent[1], nframes= int(listcontent[1]), comptype='NONE' และ compname='noncompress' จากนั้นเขียนไฟล์เสียงจากข้อมูลที่เก็บในตัวแปร listframe ด้วยคำสั่ง file.writeframes(listframe) แล้วปิดไฟล์เสียงด้วยคำสั่ง กำหนดให้ตัวแปร self.stream เก็บข้อความแสดงตำแหน่งของไฟล์เสียงที่เขียนขึ้น "recvsound.wav in C:\PycharmProjects\Chaos_gui\dist" แสดงบนหน้าต่าง GUI ด้วยคำสั่ง self.ui.display.addItem("recv : " +str(self.stream)[:100]) เมื่อโปรแกรมย่อยนี้ทำงานเสร็จ แล้วก็จะล้างค่าในตัวแปร self.stream แสดงคำสั่งทั้งหมดดังรูปที่ 3.68

```
if listcontent[0]==1:
    file = wave.open('recvsound.wav', 'wb')
    file.setnframes(listcontent[1])
    listframe = ""
    for frame in listcontent[4:]:
        listframe+=frame
    print "frames" + listframe
    print "nframe : " + str(listcontent[1])
    print "frerate : " + str(listcontent[2])
    print "samp : " + str(listcontent[3])
    file.setparams((1, int(listcontent[3]), int(listcontent[2]), int(listcontent[1]), 'NONE', 'noncompressed'))
    file.writeframes(listframe)
    file.close()
    self.stream = "recvsound.wav in C:\PycharmProjects\Chaos_gui\dist"
    self.ui.display.addItem("recv : "+str(self.stream)[:100])
    self.stream = ""
```

รูปที่ 3.68 การตรวจสอบการรับข้อมูลเสียง

เมื่อสมาชิกตำแหน่งที่ 0 ของตัวแปร listcontent มีค่าเป็น 2 จะแสดงว่าเป็นข้อมูลภาพ ทำการเก็บค่าความกว้างของภาพ ด้วยคำสั่ง width=listcontent[1] และเก็บค่าความสูงของภาพ ด้วยคำสั่ง height=listcontent[2] สร้างไฟล์ภาพด้วยคำสั่ง img=Image.new(mode, size, color) โดยกำหนดโหมดเป็นภาพสี (RGB) ขนาด(width, height) และสีที่ต้องการเติมในส่วนที่หายไปกำหนดเป็นสีดำ (Black) ทำการโหลดแต่ละ Pixel ของภาพจากไฟล์ที่สร้างขึ้นด้วยคำสั่ง pixels = img.load() โดยกำหนดค่าของ Pixel จากข้อมูลที่ได้รับ(listcontent) ก่อนที่จะแสดงภาพด้วยคำสั่ง img.show() และจัดเก็บรูปภาพด้วยคำสั่ง img.save('image.jpg') กำหนดให้ตัวแปร self.stream เก็บข้อความแสดงว่าได้รับข้อมูลภาพแล้ว "Receive Image" แสดงบนหน้าต่าง GUI ด้วยคำสั่ง self.ui.display.addItem("recv : " +str(self.stream)[:100]) เมื่อโปรแกรมย่อยทำงานเสร็จก็จะล้างค่าในตัวแปร self.stream แสดงคำสั่งทั้งหมดดังรูปที่ 3.69

```
if listcontent[0]==2:
    width = listcontent[1]
    height = listcontent[2]
    img = Image.new( 'RGB', (width,height), "black")
    pixels = img.load()
    i=0
    j=0
    for pix in listcontent[3:]:
        if pix == "\n":
            i=0
            j+=1
        else:
            print i,j
        if pix != "\n":
            try:
                pixels[i,j] = pix
            except:
                pixels[i,j] = (255,255,255)
            i+=1
    img.show()
    img.save('image.jpg')
    self.stream = "Receive Image"
    self.ui.display.addItem("recv : "+str(self.stream)[:100])
    self.stream = ""
```

รูปที่ 3.69 การตรวจสอบการรับข้อมูลภาพ

3.1.7 การออกแบบการโจมตีแบบตะลุย (Brute Force)

เป็นการออกแบบเพื่อทดสอบเปรียบเทียบระบบการเข้ารหัสแบบเคออติกกับระบบการเข้ารหัสที่เป็นมาตรฐานที่ใช้อยู่ในปัจจุบัน อาทิ เช่น Double DES , Triple DES , AES โดยความแข็งแกร่งของระบบการเข้ารหัสจะขึ้นอยู่กับ 2 องค์ประกอบเป็นสำคัญ คือ 1. อัลกอริทึมที่ใช้เข้ารหัส 2.กุญแจ โดยในส่วนนี้จะเป็นการพิจารณากุญแจซึ่งจะมีผลต่อความแข็งแกร่งของระบบเข้ารหัสเป็นอย่างมาก โดยการคำนวณเวลาที่ใช้ในการถอดรหัสจะเป็นไปตามสมการ

$$\text{เวลาที่ใช้ในการถอดรหัส (1 ล้านครั้งต่อวินาที) } = \frac{2^n}{2 \times Time \times 365 \times 24 \times 60 \times 60}$$

$$\text{เวลาที่ใช้ในการถอดรหัส (1 ล้านล้านครั้งต่อวินาที) } = \frac{2^n}{2 \times Time \times 365 \times 24 \times 60 \times 60}$$

เมื่อ n คือ จำนวนบิตที่ใช้ในการเข้ารหัส ยิ่งจำนวนบิตมากขึ้นเท่าไรเวลาที่ใช้ในการถอดรหัสก็มากขึ้นเท่านั้น

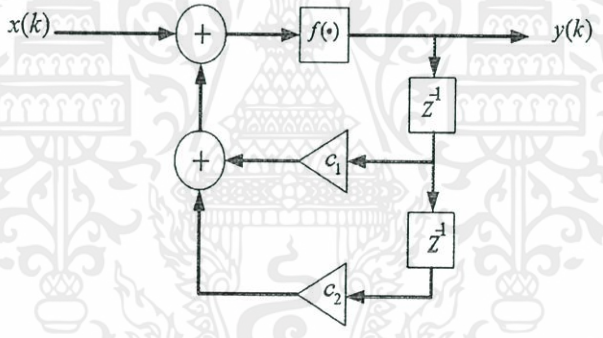
บทที่ 4

ผลการทดลอง

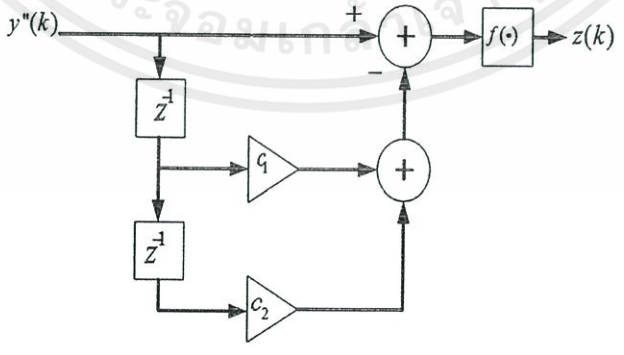
ผลการทดลองจากการออกแบบวงจรเข้ารหัส – ถอดรหัสลับแบบเคออติกโดยใช้วงจรกรองสัญญาณเชิงเลขอันดับสอง แบ่งเป็นส่วนของการจำลองการทำงานด้วยโปรแกรม MATLAB ผลการออกแบบด้วยภาษาไพธอน และผลการสร้างการทดสอบระบบสื่อสารข้อมูลที่มีการเข้ารหัสลับแบบเคออติกกับการสื่อสารข้อมูลอนุกรม และผลสร้างการทดสอบระบบสื่อสารข้อมูลที่มีการเข้ารหัสลับแบบเคออติกกับการสื่อสารผ่านเครือข่าย

4.1 ผลการจำลองการทำงานของวงจรเข้ารหัส - ถอดรหัสลับแบบเคออติกด้วยโปรแกรม MATLAB

โดยโครงสร้างของวงจรเข้ารหัสลับ และวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองที่ใช้ในการจำลองการทำงานแสดงดังรูปที่ 4.1 และ 4.2



รูปที่ 4.1 โครงสร้างของวงจรเข้ารหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสอง ชนิดอิมพัลส์ไม่จำกัด



รูปที่ 4.2 โครงสร้างของวงจรถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสอง ชนิดอิมพัลส์จำกัด

จากโครงสร้างของวงจรเข้ารหัส – ถอดรหัสลับแบบวงจรกรองสัญญาณเชิงเลขอันดับสองในรูปที่ 4.1 และ 4.2 สามารถเขียนเป็นสมการผลต่างสืบเนื่องได้ดังนี้

สมการผลต่างสืบเนื่องของวงจรเข้ารหัสลับ : $y(k) = f\{x(k) + c_1 y(k-1) + c_2 y(k-2)\}$
สมการผลต่างสืบเนื่องของวงจรถอดรหัสลับ : $z(k) = f\{\hat{y}(k) - c_3 \hat{y}(k-1) - c_4 \hat{y}(k-2)\}$
ฟังก์ชัน $f(\bullet)$: $f(x) = (x+1) \bmod 2 - 1$

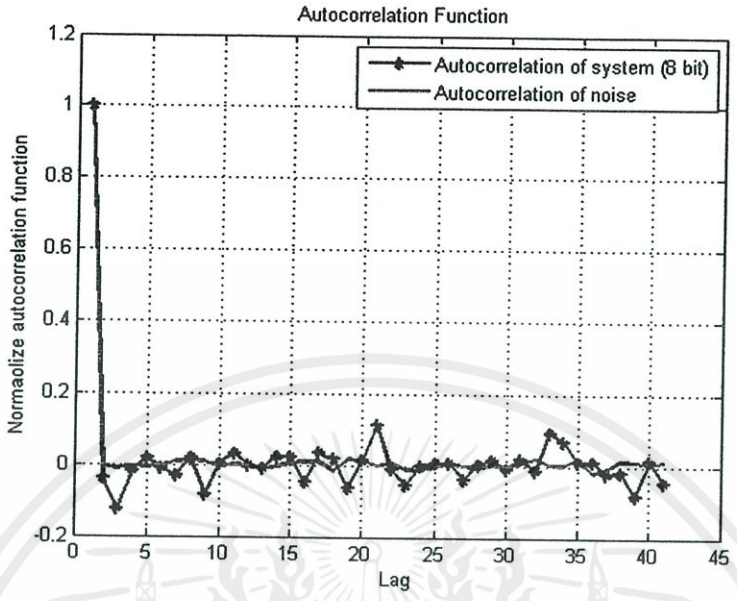
เพื่อให้เข้าใจการทำงานของวงจรกรองสัญญาณเชิงเลขที่ใช้เป็นวงจรเข้ารหัสลับและวงจรถอดรหัสลับ จึงต้องมีการวิเคราะห์วงจรเข้ารหัสลับและถอดรหัสลับ ซึ่งในปริญญานิพนธ์นี้ได้ทำการวิเคราะห์พฤติกรรมของวงจรเข้ารหัสลับและถอดรหัสลับแบบไม่สนใจปัจจัยภายนอก โดยใช้คุณสมบัติของระบบที่ไม่เป็นเชิงเส้น (Nonlinear system) คือ อินพุตเท่ากับศูนย์แต่เอาท์พุตไม่เท่ากับศูนย์ (Zero in non zero out) แต่เนื่องจากอินพุตของระบบเท่ากับศูนย์ (Zero input) จึงต้องกำหนดค่าเริ่มต้น (Initial value) ให้กับระบบด้วย

4.1.1 ผลเมื่ออินพุตเป็นศูนย์

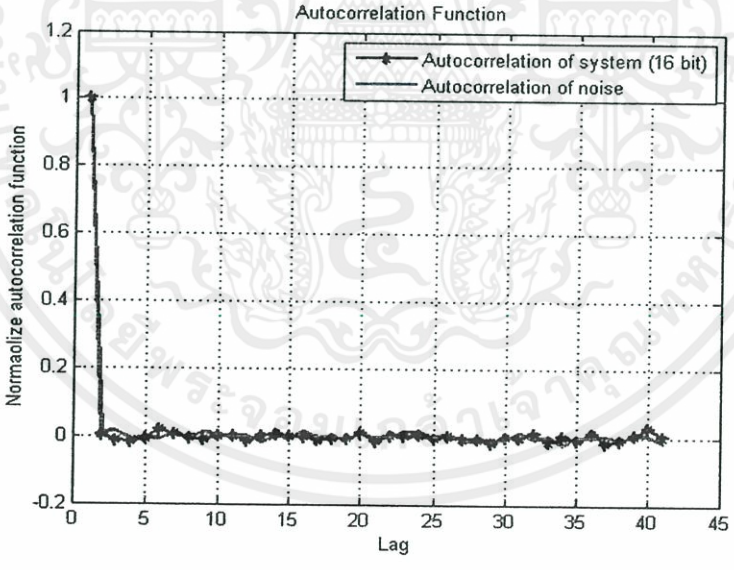
เมื่อระบบมีอินพุตเป็นศูนย์ลักษณะของค่าอัตสหสัมพันธ์ของผลตอบสนองของระบบมีลักษณะคล้ายกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน โดยค่าอัตสหสัมพันธ์เป็นสิ่งที่บ่งบอกความเหมือนกันหรือคล้ายคลึงกันระหว่างสัญญาณสุ่มตัวแปรเดียวกัน โดยการวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาต์พุตของวงจรเข้ารหัสลับ กำหนดให้ค่าเริ่มต้นให้ y(1) = 0.6135, y(2) = 0 กำหนดให้ระบบมีขนาดของข้อมูลอินพุตเป็น 8 บิตและใช้จำนวนจุดในการทดลอง 10,000 จุดจะได้ผลการทดลองดังรูปที่ 4.3

การวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับ ที่กำหนดให้ระบบมีขนาดของข้อมูลเป็น 16 บิต จะได้ผลการทดลองดังรูปที่ 4.4 โดยกำหนดให้ค่าเริ่มต้น y(1) = 0.6135, y(2) = 0 และใช้จำนวนจุดในการทดลอง 10,000 จุด

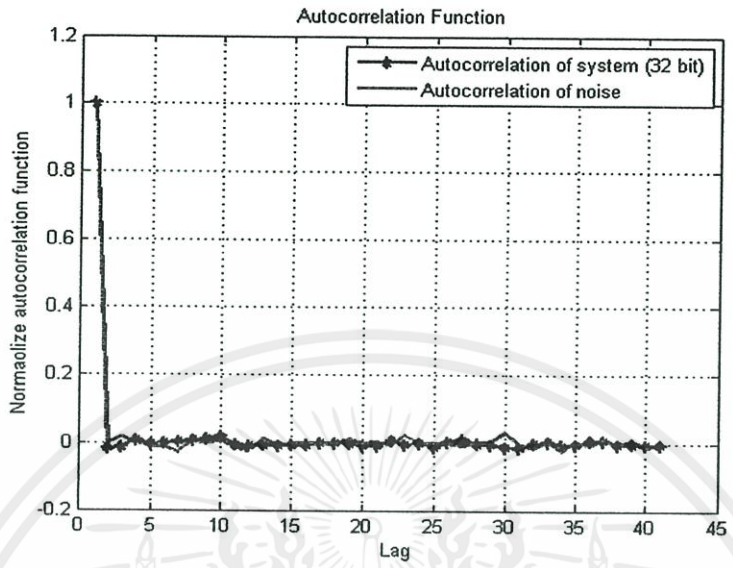
การวิเคราะห์ฟังก์ชันค่าอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับ ที่กำหนดให้ระบบมีขนาดของข้อมูลเป็น 32 บิต จะได้ผลการทดลองดังรูปที่ 4.5 โดยกำหนดให้ค่าเริ่มต้น y(1) = 0.6135, y(2) = 0 และใช้จำนวนจุดในการทดลอง 10,000 จุด



รูปที่ 4.3 กราฟค่าอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับ ขนาด 8 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน



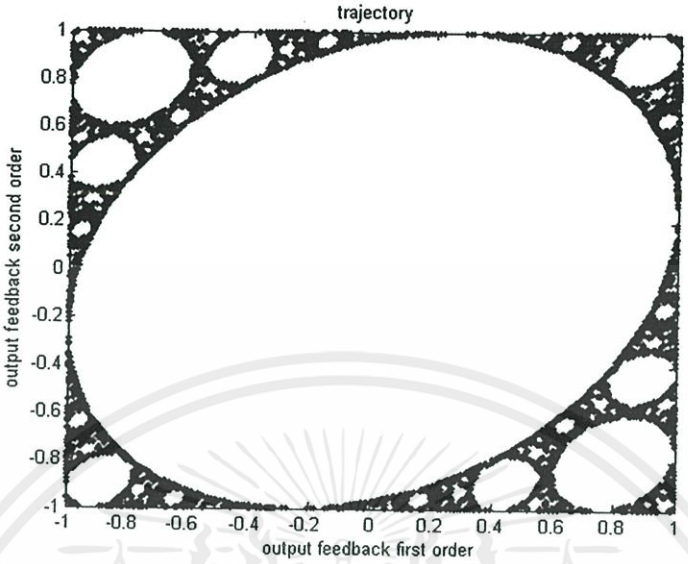
รูปที่ 4.4 กราฟค่าอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับ ขนาด 16 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน



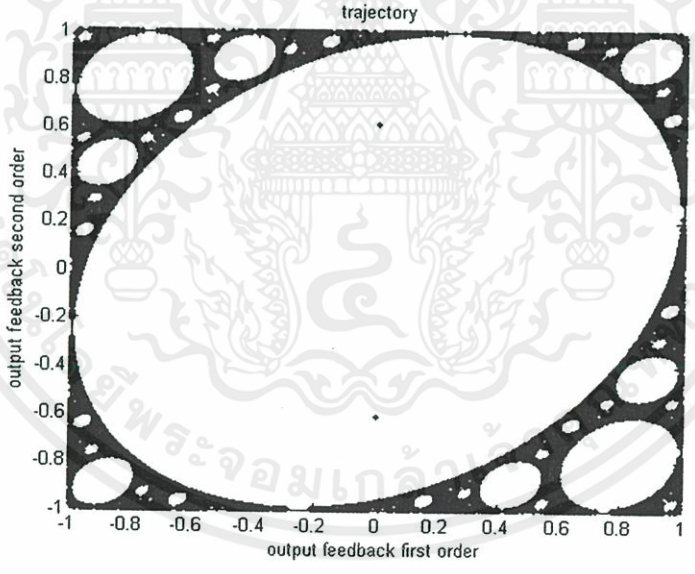
รูปที่ 4.5 กราฟค่าอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับ ขนาด 32 บิต เทียบกับค่าอัตสหสัมพันธ์ของสัญญาณรบกวน

จากรูปที่ 4.3, 4.4 และ 4.5 แสดงการพล็อตนอร์มอลไลซ์ฟังก์ชันอัตสหสัมพันธ์ของเอาท์พุตของวงจรเข้ารหัสลับจะเห็นได้ว่ามีลักษณะเหมือนฟังก์ชันค่าอัตสหสัมพันธ์ของสัญญาณรบกวน แสดงให้เห็นว่าเอาท์พุตของวงจรเข้ารหัสลับมีความสัมพันธ์กันต่ำมากเพราะเป็นสัญญาณไม่มีคาบ ซึ่งเป็นลักษณะของการเข้ารหัสลับที่ดีแสดงว่าเอาท์พุตของวงจรเข้ารหัสลับ เป็นข้อมูลแบบสุ่มที่สามารถคาดเดาได้ยากและจะเห็นได้ว่าถ้ากำหนดให้ระบบมีจำนวนบิตมากขึ้น ฟังก์ชันค่าอัตสหสัมพันธ์ของวงจรเข้ารหัสลับจะมีความใกล้เคียงกับฟังก์ชันค่าอัตสหสัมพันธ์ของสัญญาณรบกวนมากขึ้น แต่ข้อเสียของการกำหนดให้ระบบจำนวนบิตเพิ่มมากขึ้น คือจะทำให้ระบบมีการคำนวณที่ซับซ้อนขึ้นทำให้การทำงานของระบบมีความล่าช้า

ในการทดลองเมื่ออินพุตเป็นศูนย์และกำหนดให้เงื่อนไขเริ่มต้น y(1) = -0.6135 และ y(2) = 0.6135 ค่าสัมประสิทธิ์ $c_1 = 0.5$ และ $c_2 = -1$ ใช้จำนวนจุดในการทดลอง 10,000 จุด และ 100,000 จุด แล้วทดลองพล็อตเส้นทางการเคลื่อนที่ (Trajectory) โดยการพล็อตค่า Output feedback first order และ Second order ของวงจรกรองสัญญาณเชิงเลขจะแสดงดังรูปที่ 4.6 และ 4.7 ตามลำดับ



รูปที่ 4.6 Trajectory จำนวนจุดในการทดลอง 10,000 จุด



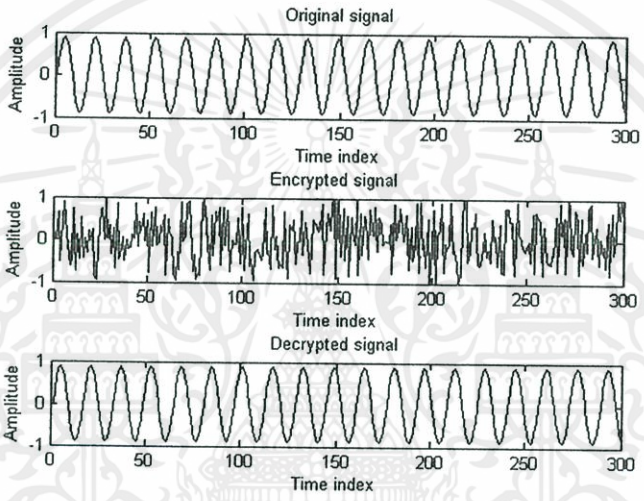
รูปที่ 4.7 Trajectory จำนวนจุดในการทดลอง 100,000 จุด

จากรูปที่ 4.6 และ รูปที่ 4.7 จะบ่งบอกถึงความเป็นแฟรกตัล หรือความคล้ายกับตัวเอง (Self-similarity) ซึ่งเป็นคุณสมบัติหนึ่งที่บ่งบอกถึงความเป็นเคออส และเมื่อเปรียบเทียบ

จำนวนจุดในการทดลอง พบว่าการเพิ่มจำนวนของจุดในการทดลอง ทำให้ความคล้ายตัวเองมีความชัดเจนมากขึ้น

4.1.2 ผลการจำลองด้วยโปรแกรม MATLAB เมื่ออินพุตเป็นสัญญาณไซน์

สัญญาณไซน์มีความถี่ $\omega = 0.125\pi$ $^{rad}/_{s}$, ความยาว k=1 ถึง 300, Amplitude = 0.9 และกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากัน คือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ ซึ่งจะแสดงผลการจำลองการทำงานดังรูปที่ 4.8



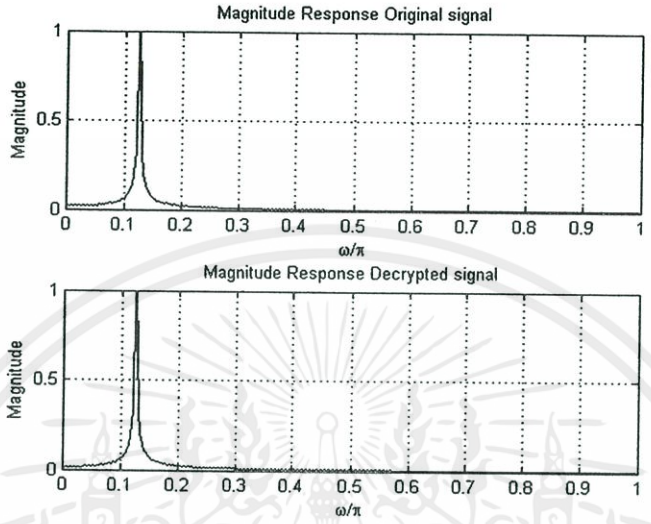
รูปที่ 4.8 ผลการจำลองการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณไซน์ โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าตรงกัน

จากรูปที่ 4.8 จะเห็นได้ว่าสามารถทำการเข้ารหัสลับและถอดรหัสลับได้อย่างสมบูรณ์ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ ซึ่งเมื่อหาค่าระยะห่างยูคลีดีนระหว่างสัญญาณเดิ้งเดิม $x(k)$ และสัญญาณท่ผ่านการถอดรหัสลับ $z(k)$ ตามสมการที่ 4.1

$$D = \sqrt{\sum_{k=3}^{300}\{(x(k) - z(k))^2\}} = 2.8294e-15 \approx 0 \qquad (4.1)$$

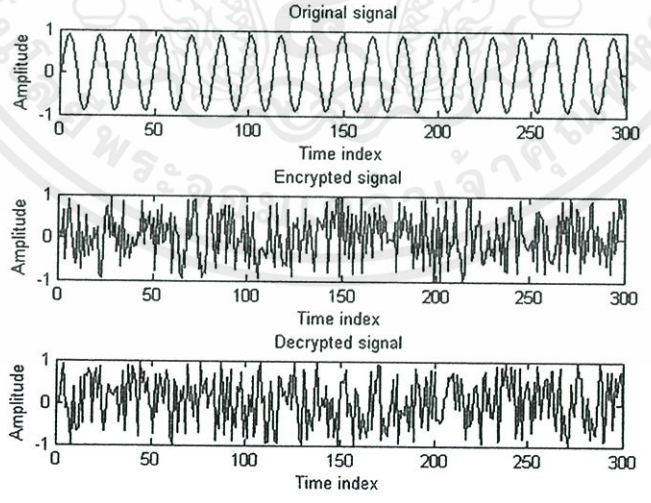
จากสมการที่ 4.1 จะเห็นได้ว่าค่าระยะห่างยูคลีดีน (Euclidean distance) มีค่าเข้าใกล้ 0 หมายความว่าสัญญาณทั้งสองนั้นมีความใกล้เคียงกันมาก ตรวจสอบได้จากการ

เปรียบเทียบสเปกตรัมของสัญญาณไซน์ระหว่างสัญญาณดั้งเดิม (Original signal) และสัญญาณที่ผ่านการถอดรหัสลับ (Decrypted signal) แสดงดังรูปที่ 4.9



รูปที่ 4.9 ค่าสเปกตรัมของสัญญาณไซน์ดั้งเดิมเปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัสลับ โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากนั้นทำการทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้ไม่เท่ากัน โดยวงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ผลการทดลองดังรูปที่ 4.10

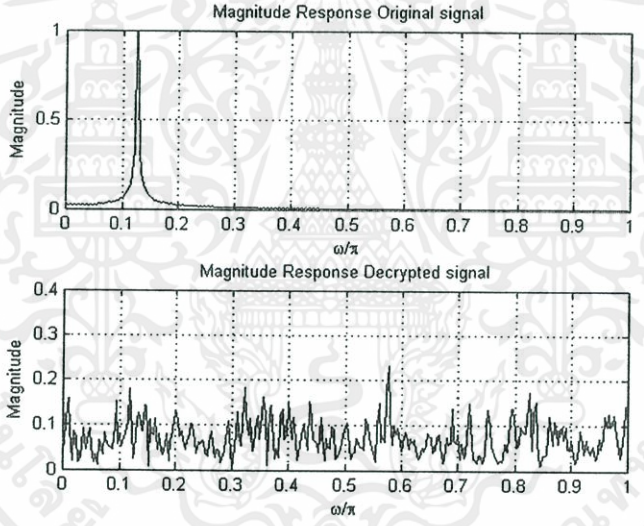


รูปที่ 4.10 ผลการจำลองการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณไซน์ โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่ตรงกัน

จากรูปที่ 4.10 จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับ และวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ค่าระยะห่างยูคลีดีน ระหว่างสัญญาณดั้งเดิมและสัญญาณที่ผ่านการถอดรหัสลับ ตามสมการที่ 4.2

$$D = \sqrt{\sum_{k=3}^{300} \{(x(k) - z(k))^2\}} = 14.7320 \qquad (4.2)$$

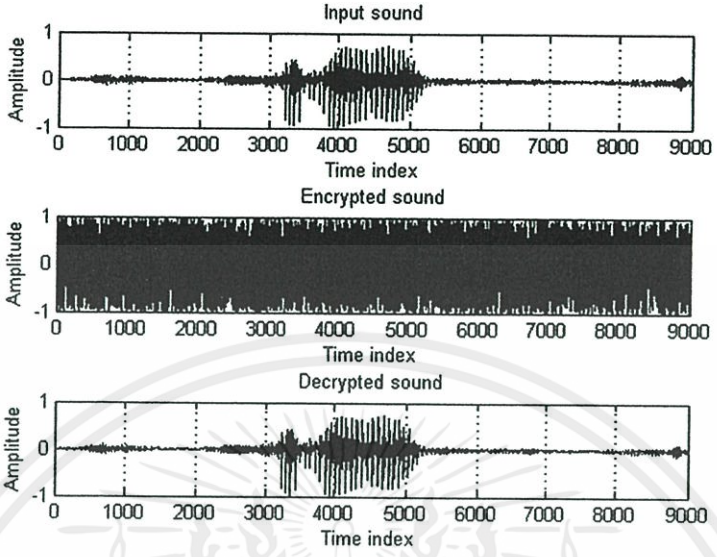
จากสมการที่ 4.2 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่า 14.7320 ซึ่งแสดงว่าสัญญาณทั้งสองนั้นไม่มีความใกล้เคียงกัน ตรวจสอบได้จากการเปรียบเทียบสเปกตรัม ระหว่างระหว่างสัญญาณดั้งเดิมและสัญญาณที่ผ่านการถอดรหัสลับ แสดงดังรูปที่ 4.11



รูปที่ 4.11 ค่าสเปกตรัมของสัญญาณไซน์ดั้งเดิมเปรียบเทียบกับสัญญาณที่ผ่านการถอดรหัสลับ โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

4.1.3 ผลการจำลองการทำงานด้วยโปรแกรม MATLAB เมื่ออินพุตเป็นเสียง

เมื่ออินพุตเป็นสัญญาณเสียง ในการจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียงโดยใช้โปรแกรม MATLAB อ่านข้อมูลเสียงแล้วนำข้อมูลที่ได้ไปทำการเข้ารหัสลับและถอดรหัสลับเมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้มีค่าเท่ากัน คือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ ผลการทดลองแสดงดังรูปที่ 4.12

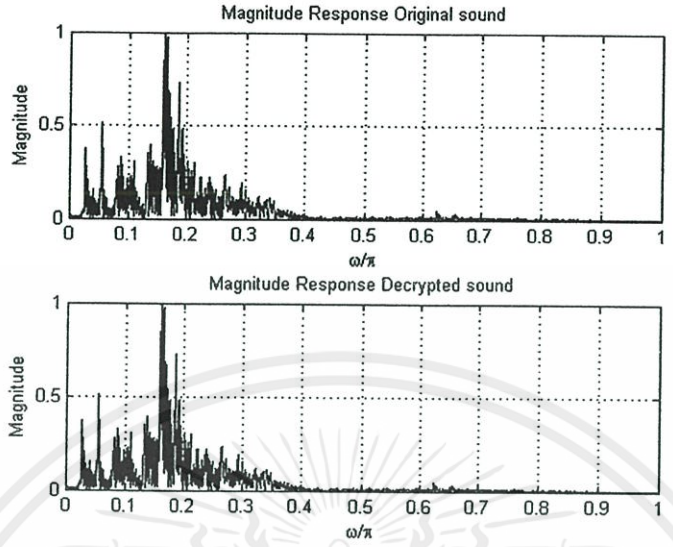


รูปที่ 4.12 ผลการจำลองการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียง โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าตรงกัน

จากรูปที่ 4.12 จะเห็นได้ว่าสามารถทำการเข้ารหัสลับและถอดรหัสลับได้อย่างสมบูรณ์ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ จะได้ค่าระยะห่างยูคลีดีนระหว่างสัญญาณเสียงดั้งเดิมและสัญญาณเสียงที่ผ่านการถอดรหัสลับตามสมการที่ 4.5

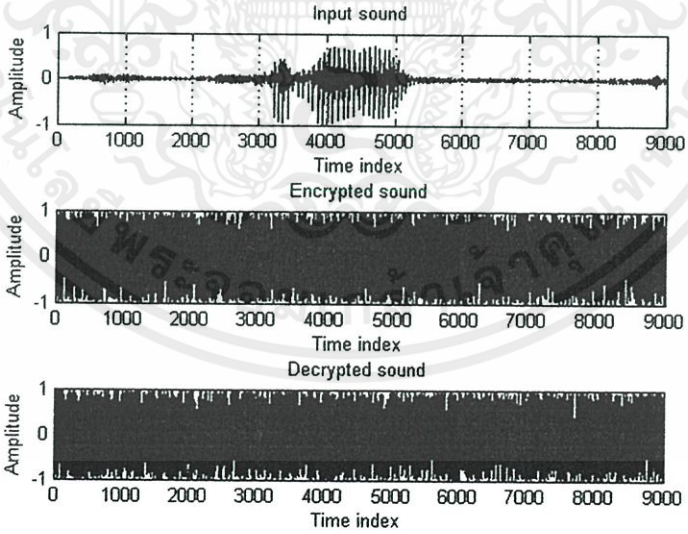
$$D = \sqrt{\sum_{k=3}^{length\ x} \{(x(k) - z(k))^2\}} = 0 \qquad (4.5)$$

จากสมการที่ 4.3 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าประมาณ 0 ซึ่งแสดงว่าสัญญาณทั้งสองมีความใกล้เคียงกันมาก ตรวจสอบได้จากการเปรียบเทียบสเปกตรัมของสัญญาณเสียงระหว่างสัญญาณดั้งเดิมและสัญญาณที่ผ่านการถอดรหัสลับแสดงดังรูปที่ 4.13



รูปที่ 4.13 ค่าสเปกตรัมของข้อมูลเสียงดั้งเดิมเปรียบเทียบกับสัญญาณเสียง
ที่ผ่านการถอดรหัสลับ โดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าเท่ากัน

จากนั้นทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้ไม่เท่ากัน โดยวงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ผลการทดลองดังรูปที่ 4.14

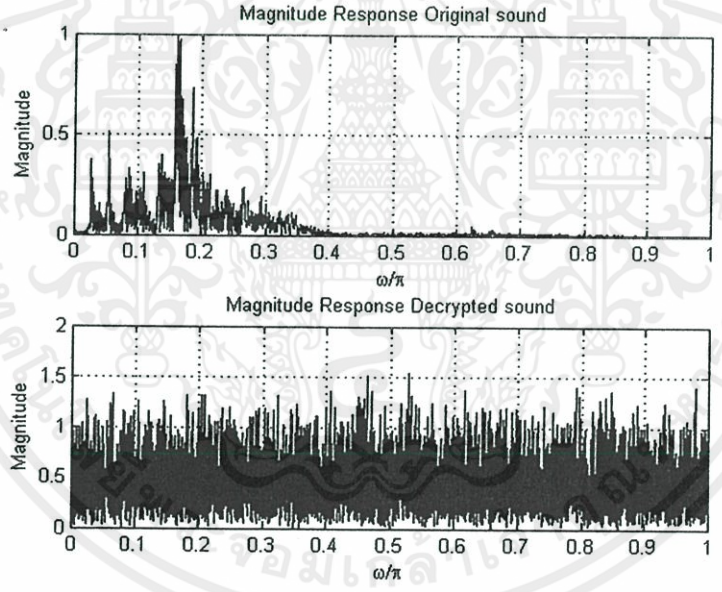


รูปที่ 4.14 ผลการจำลองการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียง
โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่ตรงกัน

จากรูปที่ 4.14 จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับข้อมูลออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับ $c_3 = 4$ และ $c_4 = -1$ จะได้ค่าระยะห่างยูคลีดีนระหว่างสัญญาณเสียงดั้งเดิมและสัญญาณเสียงที่ผ่านการถอดรหัสลับดังสมการที่ 4.6

$$D = \sqrt{\sum_{k=3}^{length\ x} \{(x(k) - z(k))^2\}} = 187.7911 \tag{4.6}$$

จากสมการที่ 4.6 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่า 131.6644 ซึ่งแสดงว่าสัญญาณทั้งสองนั้นไม่มีความใกล้เคียงกัน ตรวจสอบได้จากการเปรียบเทียบสเปกตรัมของสัญญาณเสียงระหว่างสัญญาณเสียงดั้งเดิมและสัญญาณเสียงที่ผ่านการถอดรหัสลับแสดงดังรูปที่ 4.15



รูปที่ 4.15 ค่าสเปกตรัมของข้อมูลเสียงดั้งเดิมเปรียบเทียบกับสัญญาณเสียงที่ผ่านการถอดรหัสลับโดยให้ค่าสัมประสิทธิ์ของวงจรมีค่าไม่เท่ากัน

4.1.4 ผลการจำลองการทำงานด้วยโปรแกรม MATLAB เมื่ออินพุตเป็นข้อมูลภาพ

สำหรับการจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นข้อมูลภาพ โดยให้โปรแกรม MATLAB อ่านภาพระดับสีเทา (Gray scale) ซึ่งค่าที่อ่านได้อยู่ในรูปของเมทริกซ์ซึ่งเป็นรูปแบบ 2 มิติ แต่ในการเข้ารหัสลับและถอดรหัสลับจะใช้ข้อมูลรูปแบบ 1 มิติ

ดังนั้นจึงต้องเรียงข้อมูลใหม่โดยใช้คำสั่ง reshape ก่อนนำข้อมูลผ่านการเข้ารหัสลับหรือถอดรหัสลับ การจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส - ถอดรหัสลับมีค่าเท่ากัน ($c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$) ผลแสดงดังรูปที่ 4.16

รูปที่ 4.16 ข้อมูลภาพเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ และภาพที่ผ่านการถอดรหัสลับ

จากรูปที่ 4.16 จะเห็นได้ว่าสามารถทำการถอดรหัสลับออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะได้ค่าระยะห่างยูคลีดีน ระหว่างข้อมูลภาพดั้งเดิมและข้อมูลภาพที่ผ่านการถอดรหัสลับ และค่า PSNR ตามสมการที่ 4.7 และ 4.8

$$D = \sqrt{\sum_{k=3}^{50625} \{(x(k) - z(x))^2\}} \approx 0 \tag{4.7}$$

$$PSNR = 20\log_{10}\left[\frac{b}{RMSE}\right] = 221.3509 \ dB \tag{4.8}$$

เมื่อตัวแปร b คือค่าสูงสุดที่เป็นไปได้ของพิกเซลในภาพ ในที่นี้ก็คือค่าสูงสุดที่ข้อมูลขนาด 8 บิตสามารถแสดงได้ นั่นคือ 255

ค่า Root Mean Square Error (RMSE) และค่า Mean Square Error (MSE) สามารถคำนวณได้จากสมการที่ (4.9) และ (4.10) ตามลำดับ

$$RMSE = \sqrt{MSE} \tag{4.9}$$

$$MSE = \frac{1}{N} \times (SUM_{ij} \left| Org_{ij} - Wmk_{ij} \right|^2) \tag{4.10}$$

เมื่อ Org_{ij} คือค่าพิกเซลของภาพต้นฉบับ Wmk_{ij} คือค่าพิกเซลของภาพที่ถูกถอดรหัสลับแล้ว N คือจำนวนพิกเซลทั้งหมดภายในภาพที่ได้มาจากผลคูณระหว่างขนาดพิกเซลทางกว้างและทางยาว

จากสมการที่ 4.7 และ 4.8 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าเข้าใกล้ 0 บ่งบอกถึงความเหมือนของรูปภาพและในส่วนของค่า PSNR ที่มีค่าสูงซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันมาก

การจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นข้อมูลภาพโดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับมีค่าไม่เท่ากัน โดยกำหนดให้วงจรเข้ารหัสลับมีค่า $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับมีค่า $c_3 = 4$ และ $c_4 = -1$ ผลของการเข้ารหัสลับและถอดรหัสลับแสดงดังรูปที่ 4.17

รูปที่ 4.17 ข้อมูลภาพเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ และภาพที่ผ่านการถอดรหัสลับ

จากรูปที่ 4.17 จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ค่าระยะห่างยูคลีดีน และค่า PSNR ระหว่างข้อมูลภาพดั่งเดิมและข้อมูลภาพที่ผ่านการถอดรหัสลับตามสมการที่ 4.11 และ 4.12

$$D = \sqrt{\sum_{k=3}^{50625} \{(x(k) - z(x))^2\}} = 175.0428 \tag{4.11}$$

$$PSNR = 20\log_{10}\left[\frac{b}{RMSE}\right] = 115.8464 \ \ dB \tag{4.12}$$

จากสมการที่ 4.11 และ 4.12 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและในส่วนของค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันน้อยมาก

การจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับกับภาพข้อมูลที่เป็นความลับ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับมีค่าเท่ากัน คือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ ผลการเข้ารหัสลับและถอดรหัสลับแสดงดังรูปที่ 4.18

รูปที่ 4.18 ข้อมูลเอกสารเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ และภาพที่ผ่านการถอดรหัสลับ

จากรูปที่ 4.18 จะเห็นได้ว่าสามารถทำการถอดรหัสลับออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะได้ค่าระยะห่างยูคลีดีน และค่า PSNR ระหว่างข้อมูลภาพดั้งเดิมและข้อมูลภาพที่ผ่านการถอดรหัสลับตามสมการที่ 4.13 และ 4.14

$$D = \sqrt{\sum_{k=3}^{572160} \{(x(k) - z(x))^2\}} = 1.3477\text{e-}013 \approx 0 \tag{4.13}$$

$$PSNR = 20\log_{10}\left[\frac{b}{RMSE}\right] = 237.5949 \ \ dB \tag{4.14}$$

จากสมการที่ 4.13 และ 4.14 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าเข้าใกล้ศูนย์ บ่งบอกถึงความเหมือนของรูปภาพและในส่วนของค่า PSNR ที่มีค่าสูงซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันมาก

การจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับเอกสารที่เป็นความลับ โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับมีค่าไม่เท่ากัน โดยกำหนดให้วงจรเข้ารหัสลับมีค่า $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับมีค่า $c_3 = 4$ และ $c_4 = -1$ ผลการเข้ารหัสลับและถอดรหัสลับแสดงดังรูปที่ 4.19

รูปที่ 4.19 ข้อมูลเอกสารเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ และภาพที่ผ่านการถอดรหัสลับ

จากรูปที่ 4.19 จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับออกมาได้ เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ค่าระยะห่างยูคลีดีน และค่า PSNR ระหว่างข้อมูลภาพดั้งเดิมและข้อมูลภาพที่ผ่านการถอดรหัสลับตามสมการที่ 4.15 และ 4.16

$$D = \sqrt{\sum_{k=3}^{572160} \{(x(k) - z(x))^2\}} = 811.1454 \tag{4.15}$$

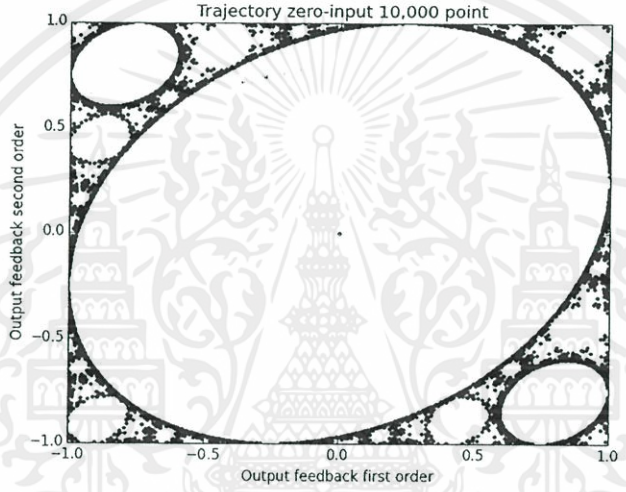
$$PSNR = 20 \log_{10} \left[\frac{b}{RMSE} \right] = 109.4280 \ dB \tag{4.16}$$

จากสมการที่ 4.13 และ 4.14 จะเห็นได้ว่าค่าระยะห่างยูคลีดีนมีค่าสูง บ่งบอกถึงความแตกต่างของรูปภาพและในส่วนของค่า PSNR ที่มีค่าต่ำซึ่งแสดงว่าคุณภาพของรูปทั้ง 2 นั้นมีความใกล้เคียงกันน้อยมาก

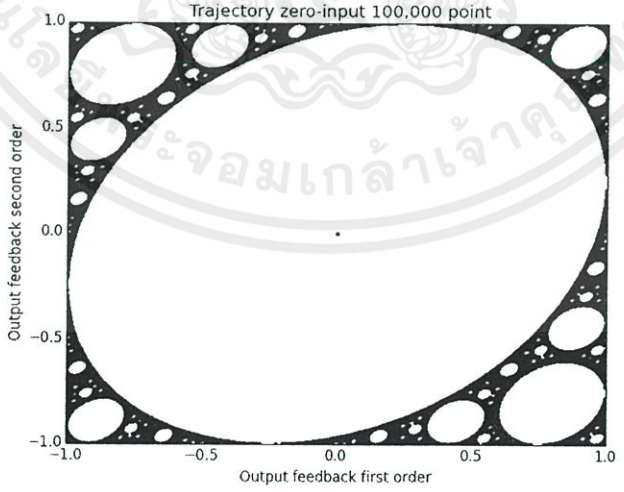
4.2 ผลการออกแบบการเข้ารหัส - ถอดรหัสลับแบบเคออติกด้วยภาษาไพธอน

4.2.1 ผลการออกแบบเมื่ออินพุตเป็นศูนย์

ในการทดลองเมื่ออินพุตเป็นศูนย์และกำหนดให้เงื่อนไขเริ่มต้น y[0] = -0.6135 และ y[1] = 0.6135 ค่าสัมประสิทธิ์ $c_1 = 0.5$ และ $c_2 = -1$ ใช้จำนวนจุดในการทดลอง 10,000 จุด และ100,000 จุด แล้วทดลองพล็อตเส้นทางการเคลื่อนที่ (Trajectory) โดยการพล็อตค่า Output feedback first order และ second order ของวงจรกรองสัญญาณเชิงเลขจะแสดงดังรูปที่ 4.20 และ 4.21 ตามลำดับ



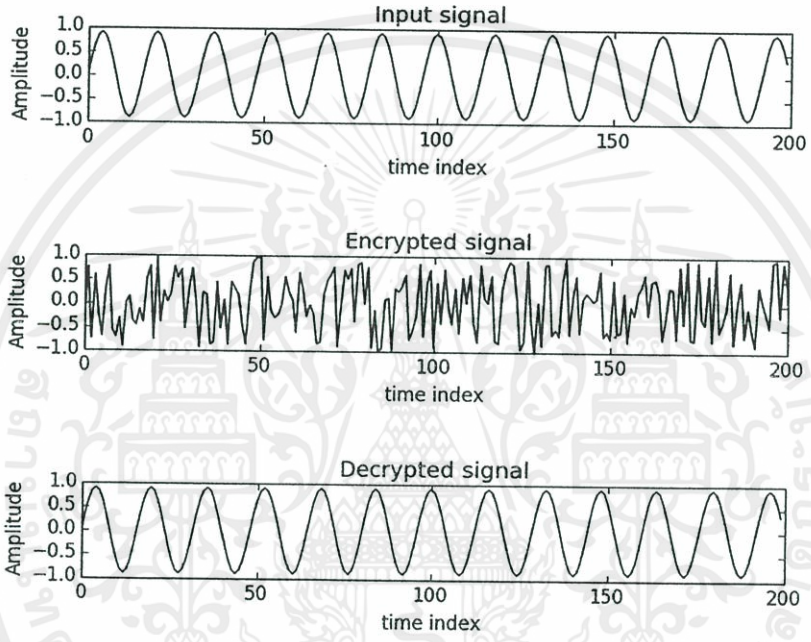
รูปที่ 4.20 Trajectory จำนวนจุดในการทดลอง 10,000 จุด (ไพธอน)



รูปที่ 4.21 Trajectory จำนวนจุดในการทดลอง 100,000 จุด (ไพธอน)

4.2.2 ผลเมื่ออินพุตเป็นสัญญาณไซน์

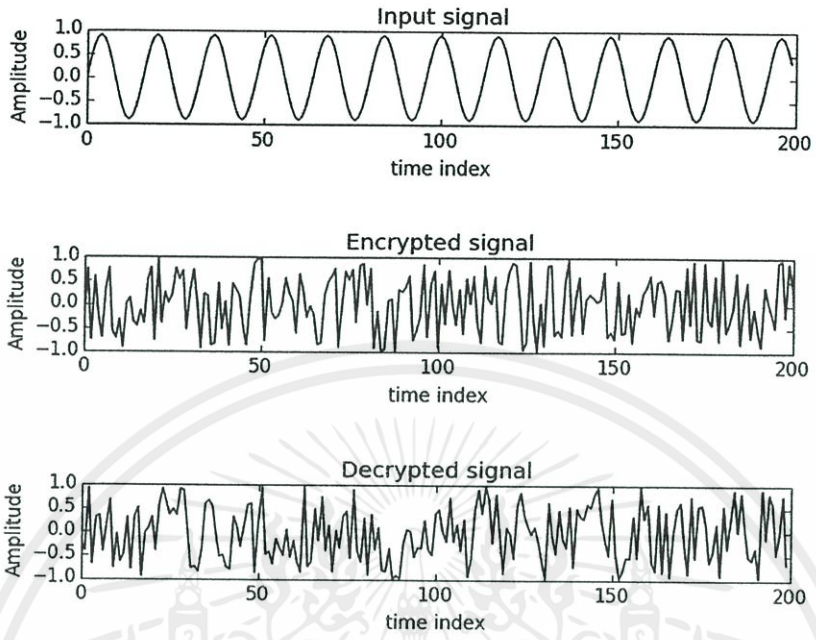
สัญญาณไซน์ความถี่ 500 Hz ความถี่ของการแซมปลิง (Sampling frequency) ที่ 8000 Hz จำนวนแซมเปิล (Sample) เท่ากับ 200 ตัว ซึ่งความถี่เชิงมุมเป็น $\omega = 0.125\pi$ $^{rad}/_{s}$ Amplitude = 0.9 และกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากัน คือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ จะเห็นได้ว่าสามารถทำการเข้ารหัสลับและถอดรหัสลับได้อย่างสมบูรณ์ แสดงผลการจำลองการทำงานดังรูปที่ 4.22



รูปที่ 4.22 ผลการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณไซน์ โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าตรงกัน (ไพธอน)

จากรูปที่ 4.22 เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ $c_1 = c_3 = 1$ และ $c_2 = c_4 = -3$ จะเห็นได้ว่าสามารถทำการเข้ารหัสลับและถอดรหัสลับได้อย่างสมบูรณ์

จากนั้นทำการทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้ไม่เท่ากัน โดยวงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะได้ผลการทดลองดังรูปที่ 4.23



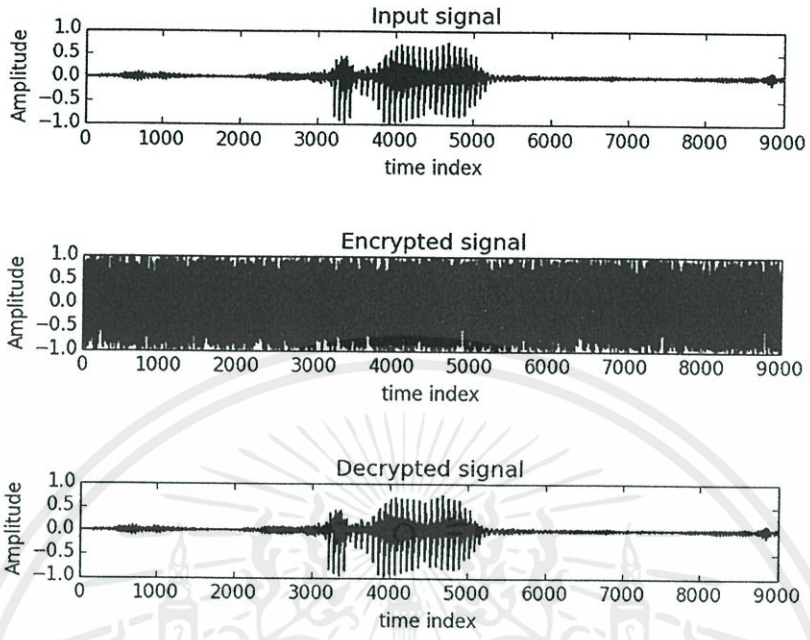
รูปที่ 4.23 ผลการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณไซน์ โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่ตรงกัน (ไพธอน)

จากรูปที่ 4.23 เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับ และวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 1$ และ $c_2 = -3$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับออกมาได้

4.2.3 ผลเมื่ออินพุตเป็นเสียง

เมื่ออินพุตเป็นสัญญาณเสียง ในการจำลองการทำงานของวงจรเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียง อ่านข้อมูลเสียงแล้วนำข้อมูลที่ได้ไปทำการเข้ารหัสลับและถอดรหัสลับเมื่อกำหนดค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้มีค่าเท่ากัน คือ $c_1 = c_3 = 4$ และ $c_2 = c_4 = -1$ ผลการทดลองแสดงดังรูปที่ 4.24

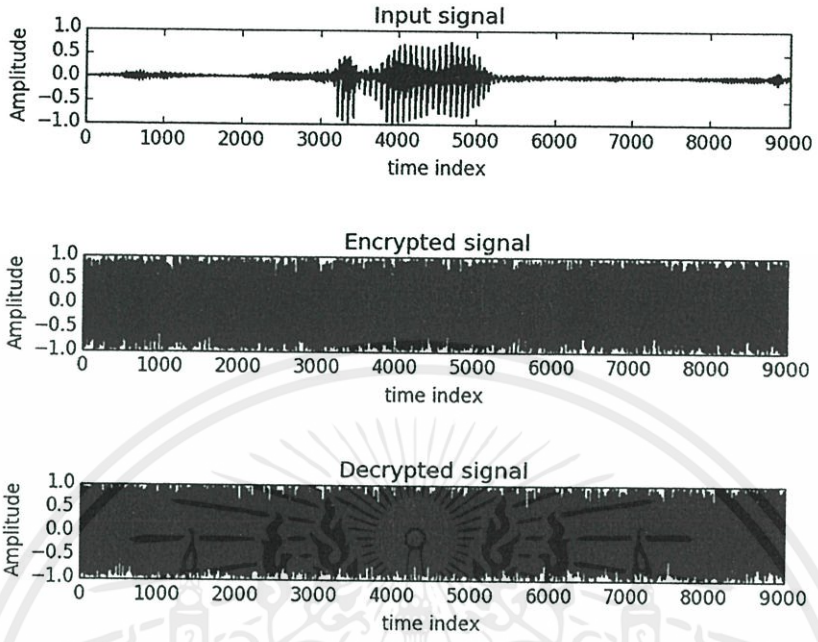
เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ $c_1 = c_3 = 4$ และ $c_2 = c_4 = -1$ จะเห็นได้ว่าสามารถทำการเข้ารหัสลับและถอดรหัสลับได้อย่างสมบูรณ์



รูปที่ 4.24 ผลการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียง โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่ตรงกัน (ไพธอน)

จากนั้นทำการทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้ไม่เท่ากัน โดยวงจรเข้ารหัสลับกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะได้ผลการทดลองดังรูปที่ 4.25

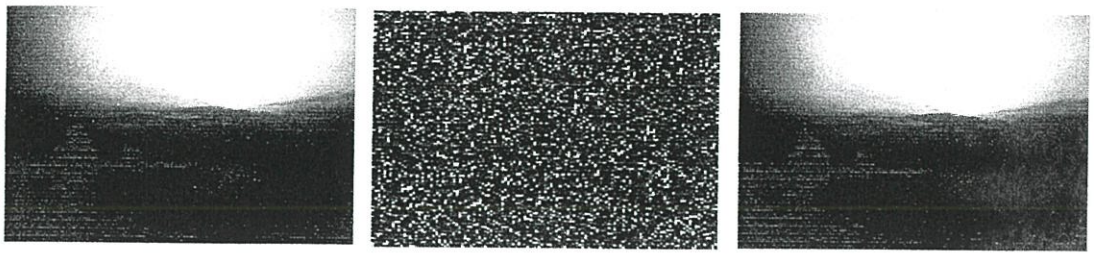
จากรูปที่ 4.25 เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับข้อมูลออกมาได้



รูปที่ 4.25 ผลการเข้ารหัสลับและถอดรหัสลับเมื่ออินพุตเป็นสัญญาณเสียง โดยกำหนดให้ค่าสัมประสิทธิ์มีค่าไม่ตรงกัน (ไพธอน)

4.2.4 ผลเมื่ออินพุตเป็นข้อมูลภาพ

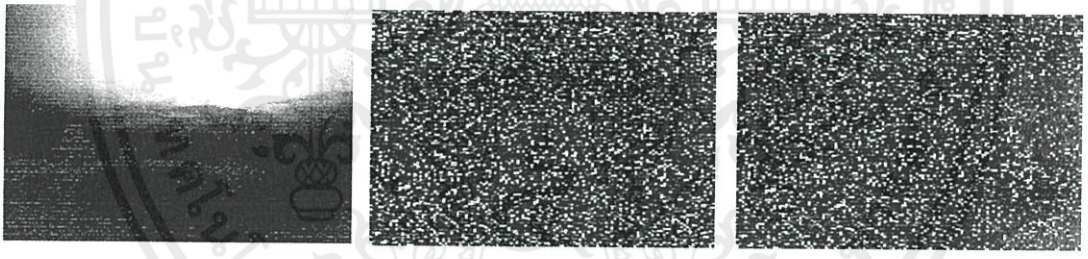
ข้อมูลภาพจะอยู่ในรูปของเมทริกซ์ องค์ประกอบภายในของเมทริกซ์คือ พิกเซล (Pixel) ของภาพเรียงกันเป็นข้อมูล 2 มิติ (ความกว้าง และความสูง) โดยแต่ละพิกเซลนั้นเกิดจากการผสมสีกัน 3 สี ได้แก่ สีแดง สีเขียว และสีน้ำเงิน สำหรับข้อมูลภาพขาวดำ (Grayscale) การผสมสีของพิกเซลจะผสมด้วยความเข้มของทั้ง 3 สีที่เท่ากัน แต่สำหรับข้อมูลภาพสีนั้นการผสมสีของพิกเซลจะผสมด้วยความเข้มของทั้ง 3 สีจะไม่เท่ากัน แต่เนื่องจากการเข้ารหัสลับและถอดรหัสลับจะทำงานแบบ 1 มิติ ดังนั้นจึงต้องเรียงข้อมูลใหม่โดยนำแถวของพิกเซลทุกแถวมาเรียงต่อกันเป็นมิติเดียว โดยคั่นแต่ละแถวด้วย '\n' และนำข้อมูลเดิมที่จำแนกตามพิกเซลมาจัดกลุ่มใหม่ให้จำแนกตามสี ได้ 3 กลุ่มเพื่อทำการเข้ารหัส - ถอดรหัสลับแยกตามกลุ่ม โดยกำหนดให้ค่าสัมประสิทธิ์ของวงจรเข้ารหัส - ถอดรหัสลับมีค่าเท่ากัน ($c_1 = c_3 = 4$ และ $c_2 = c_4 = -1$) แล้วจึงจัดกลุ่มให้อยู่ตามรูปเดิมอีกครั้ง เพื่อแสดงเป็นข้อมูลภาพที่ผ่านการเข้ารหัสลับ และข้อมูลภาพที่ผ่านการถอดรหัสลับ แสดงผลการทดลองดังรูปที่ 4.26



รูปที่ 4.26 ข้อมูลภาพเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ
และภาพที่ผ่านการถอดรหัสลับ เมื่อกำหนดให้ค่าสัมประสิทธิ์ตรงกัน (ไพธอน)

จากรูปที่ 4.26 เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้มีค่าเท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 4$ และ $c_4 = -1$ จะเห็นได้ว่าสามารถทำการถอดรหัสลับออกมาได้

จากนั้นทำการทดลองเปลี่ยนค่าสัมประสิทธิ์ของวงจรเข้ารหัสลับและถอดรหัสลับให้ไม่เท่ากัน โดยวงจรเข้ารหัสลับกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะได้ผลการทดลองดังรูปที่ 4.27



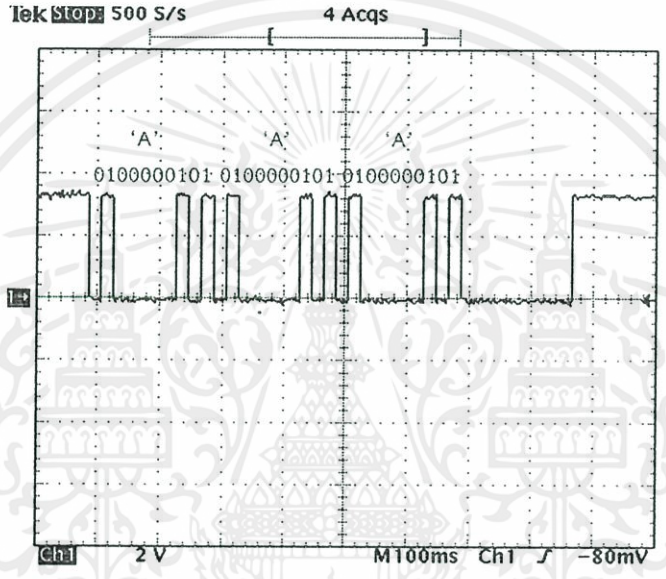
รูปที่ 4.27 ข้อมูลภาพเปรียบเทียบภาพอินพุต ภาพที่ผ่านการเข้ารหัสลับ
และภาพที่ผ่านการถอดรหัสลับเมื่อกำหนดให้ค่าสัมประสิทธิ์ไม่ตรงกัน (ไพธอน)

จากรูปที่ 4.27 เมื่อกำหนดค่าสัมประสิทธิ์ของทั้งวงจรเข้ารหัสลับและวงจรถอดรหัสลับให้ไม่เท่ากันคือ วงจรเข้ารหัสลับกำหนดให้ $c_1 = 4$ และ $c_2 = -1$ และวงจรถอดรหัสลับกำหนดให้ $c_3 = 1$ และ $c_4 = -3$ จะเห็นได้ว่าไม่สามารถทำการถอดรหัสลับข้อมูลออกมาได้

4.3 ผลการทดลองการสื่อสารข้อมูลแบบอนุกรมโดยผ่านอุปกรณ์ Raspberry Pi

4.3.1 ผลการทดลองการสื่อสารข้อมูลแบบอนุกรม จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob ที่ผ่านการเข้ารหัส - ถอดรหัสลับแบบเคออติกด้วยค่าสัมประสิทธิ์ที่ตรงกัน

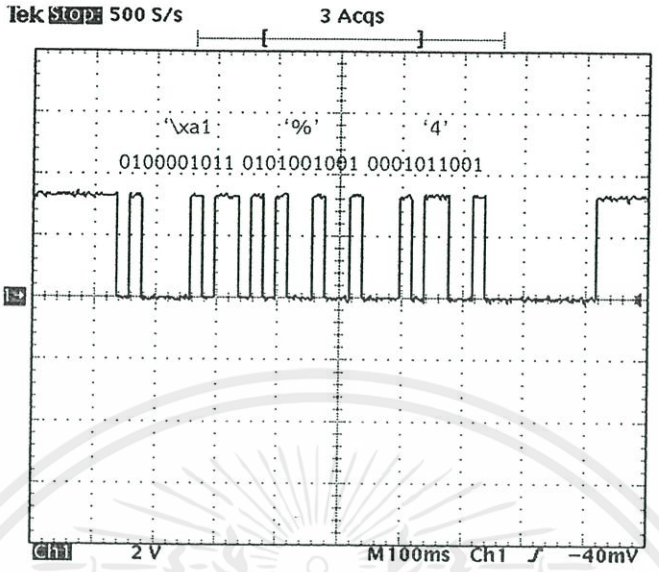
เมื่อทำการพิมพ์ตัวอักษร AAA จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1 จากนั้นนำ Oscilloscope วัดสัญญาณอินพุต ที่ขา RxD ของอุปกรณ์ Raspberry Pi 1 แสดงผลดังรูปที่ 4.28



รูปที่ 4.28 ข้อมูลอินพุต (กรณีค่าสัมประสิทธิ์ตรงกัน)

จากรูปที่ 4.28 สัญญาณที่วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100000101010000010101000000101$ เมื่อแบ่งด้วย Start bit ที่มีลอจิกเป็น "0" และ Stop bit ที่มีลอจิกเป็น "1" จะได้เป็น $01000001, 01000001, 01000001 = 41_{16}, 41_{16}, 41_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับตัวอักษร AAA (ตามที่คอมพิวเตอร์ Alice ได้ส่งมา)

เมื่ออุปกรณ์ Raspberry Pi 1 รับข้อมูลอินพุตมาทำการเข้ารหัสลับแบบเคออติกด้วยค่าสัมประสิทธิ์ของการเข้ารหัสลับเป็น $c_1 = 4$ และ $c_2 = -1$ แล้วนำ Oscilloscope วัดสัญญาณที่ผ่านการเข้ารหัสลับ ที่ขา TxD ของอุปกรณ์ Raspberry Pi 1 แสดงผลดังรูปที่ 4.29

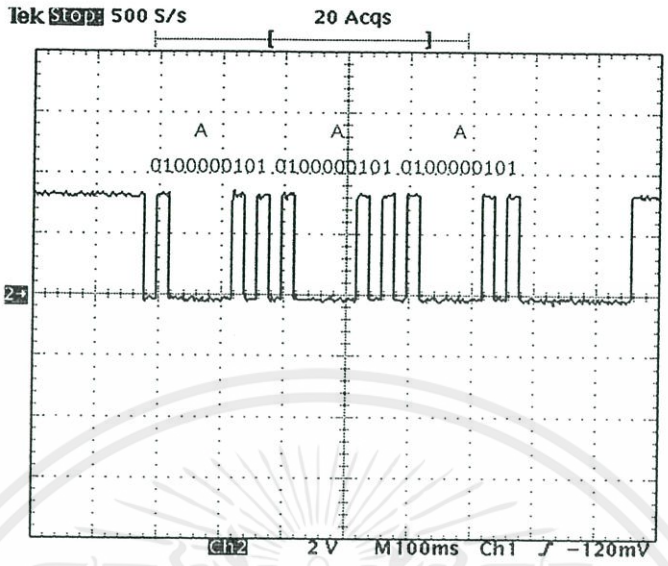


รูปที่ 4.29 ข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติก (กรณีค่าสัมประสิทธิ์ตรงกัน)

จากรูปที่ 4.29 ที่วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100001011010100100010011001$ เมื่อแบ่งด้วย Start bit และ Stop bit จะได้เป็น $10100001,00100101,00110100 = A1_{16},25_{16},34_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับตัวอักษร Null-character, %, และ 4

เมื่ออุปกรณ์ Raspberry Pi 2 รับข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติก และนำมาทำการถอดรหัสลับแบบเคออติก ด้วยค่าสัมประสิทธิ์ของการถอดรหัสลับที่ตรงกับการเข้ารหัสลับคือ $c_3 = 4$ และ $c_4 = -1$ แล้วนำ Oscilloscope วัดสัญญาณที่คอมพิวเตอร์ Bob ได้รับ โดยวัดสัญญาณจากขา TxD ของอุปกรณ์ Raspberry Pi 2

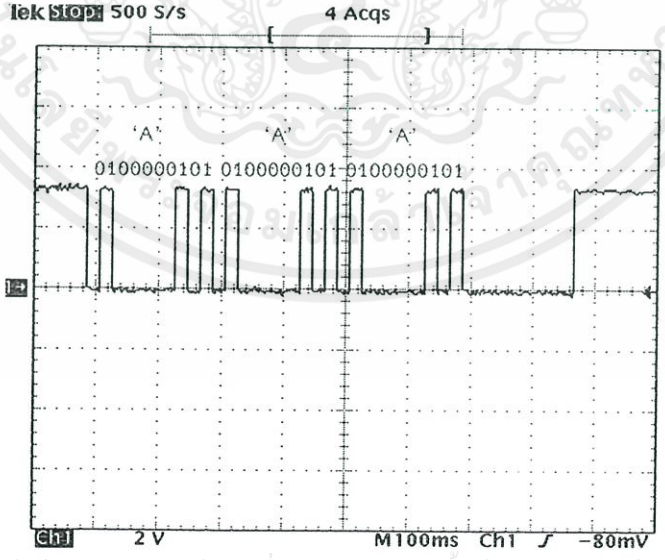
วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100000101010000010101000000101$ เมื่อแบ่งด้วย Start bit และ Stop bit จะได้เป็น $01000001,01000001,01000001 = 41_{16},41_{16},41_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับ AAA ดังนั้นคอมพิวเตอร์ Bob สามารถรับข้อมูลจากคอมพิวเตอร์ Alice ได้ถูกต้อง แสดงผลดังรูปที่ 4.30



รูปที่ 4.30 ข้อมูลที่ผ่านการถอดรหัสลับแบบเคออติก (กรณีค่าสัมประสิทธิ์ตรงกัน)

4.3.2 ผลการทดลองการสื่อสารข้อมูลแบบอนุกรม จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob ที่ผ่านการเข้ารหัส - ถอดรหัสลับแบบเคออติกด้วยค่าสัมประสิทธิ์ที่ไม่ตรงกัน

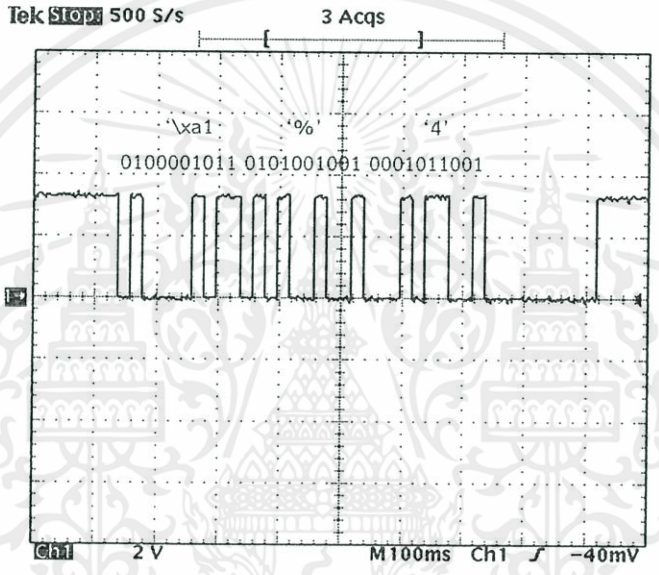
เมื่อทำการพิมพ์ตัวอักษร AAA จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1 จากนั้นนำ Oscilloscope วัดสัญญาณอินพุต ที่ขา RxD ของอุปกรณ์ Raspberry Pi 1 แสดงผลดังรูปที่ 4.31



รูปที่ 4.31 ข้อมูลอินพุต (กรณีค่าสัมประสิทธิ์ไม่ตรงกัน)

จากรูปที่ 4.31 ที่วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100001011010100100010001011001$ เมื่อแบ่งด้วย Start bit และ Stop bit จะได้เป็น $01000001, 01000001, 01000001 = 41_{16}, 41_{16}, 41_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับตัวอักษร AAA (ตามที่ Alice ได้ส่งมา)

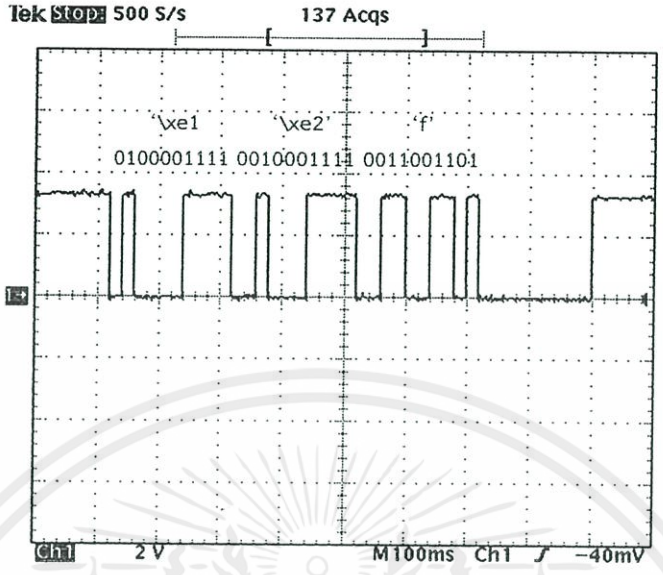
เมื่ออุปกรณ์ Raspberry Pi 1 รับข้อมูลอินพุตมาทำการเข้ารหัสลับแบบเคออติกด้วยค่าสัมประสิทธิ์ของการเข้ารหัสลับเป็น $c_1 = 4$ และ $c_2 = -1$ แล้วนำ Oscilloscope วัดสัญญาณที่ผ่านการเข้ารหัสลับ ที่ขา TxD ของอุปกรณ์ Raspberry Pi 1 แสดงผลดังรูปที่ 4.32



รูปที่ 4.32 ข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติก (กรณีค่าสัมประสิทธิ์ไม่ตรงกัน)

จากรูปที่ 4.32 ที่วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100001011010100100010001011001$ เมื่อแบ่งด้วย Start bit และ Stop bit จะได้เป็น $10100001, 00100101, 00110100 = A1_{16}, 25_{16}, 34_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับตัวอักษร Null-character, %, และ 4

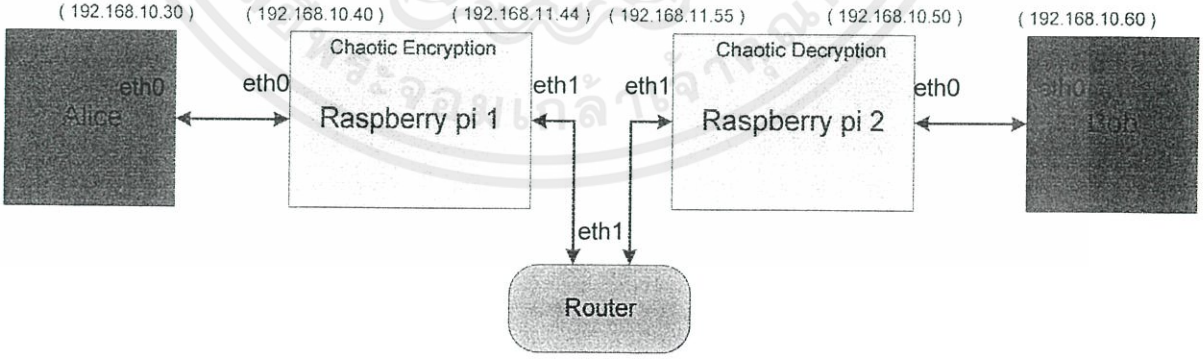
เมื่ออุปกรณ์ Raspberry Pi 2 รับข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติก และนำมาถอดรหัสลับแบบเคออติก ด้วยค่าสัมประสิทธิ์ของการถอดรหัสลับที่ไม่ตรงกับการเข้ารหัสลับคือ $c_3 = 3$ และ $c_4 = -1$ แล้วนำ Oscilloscope วัดสัญญาณที่คอมพิวเตอร์ Bob ได้รับ โดยวัดสัญญาณจากขา TxD ของอุปกรณ์ Raspberry Pi 2 ดังรูปที่ 4.33



รูปที่ 4.33 ข้อมูลที่ผ่านการถอดรหัสลับแบบเคออติก (กรณีค่าสัมประสิทธิ์ไม่ตรงกัน)

จากรูปที่ 4.33 ที่วัดได้มีขนาดแรงดันของลอจิก "1" ที่ 3.3 V และขนาดแรงดันของลอจิก "0" ที่ 0 V ค่าที่อ่านได้คือ $0100001111001000111100111001101$ เมื่อแบ่งด้วย Start bit และ Stop bit จะได้เป็น $11100001,11100010,01100110 = E1_{16}, E2_{16}, 66_{16}$ ซึ่งในรหัส ASCII มีค่าตรงกับ Null - character, Null - character, และ f

4.4 ผลการทดลองการทำงานของระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi สำหรับการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต



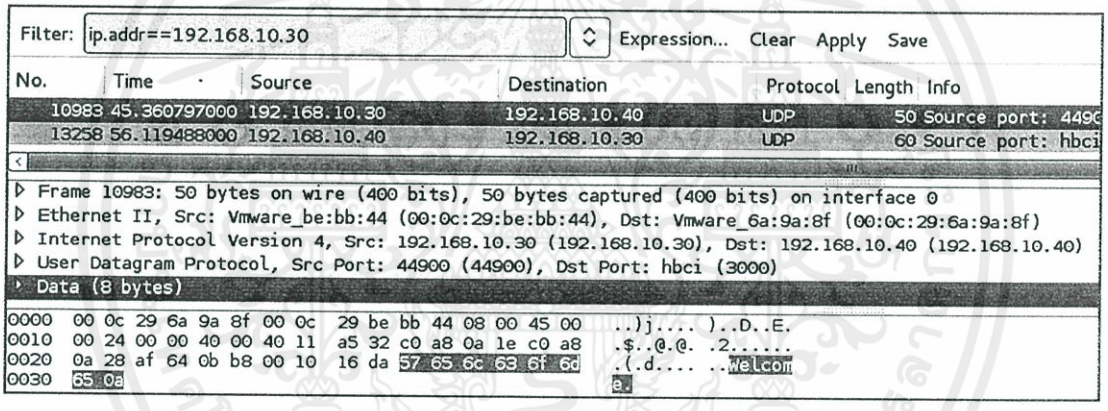
รูปที่ 4.34 การสื่อสารผ่านเครือข่ายอินเตอร์เน็ตระหว่างคอมพิวเตอร์ Alice กับ Bob โดยผ่านอุปกรณ์เข้ารหัส - ถอดรหัสลับ Raspberry Pi

จากรูปที่ 4.34 การทดลองการทำงานของระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi สำหรับการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต โดยใช้โปรโตคอล UDP และ TCP ทำการต่ออุปกรณ์การทดลองและตั้งค่าไอพีของอุปกรณ์แต่ละตัวดังรูป

4.4.1 ผลการทดลองโดยใช้โปรโตคอล UDP และใช้สัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขในการเข้ารหัส – ถอดรหัสลับ ตรงกัน

4.4.1.1 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

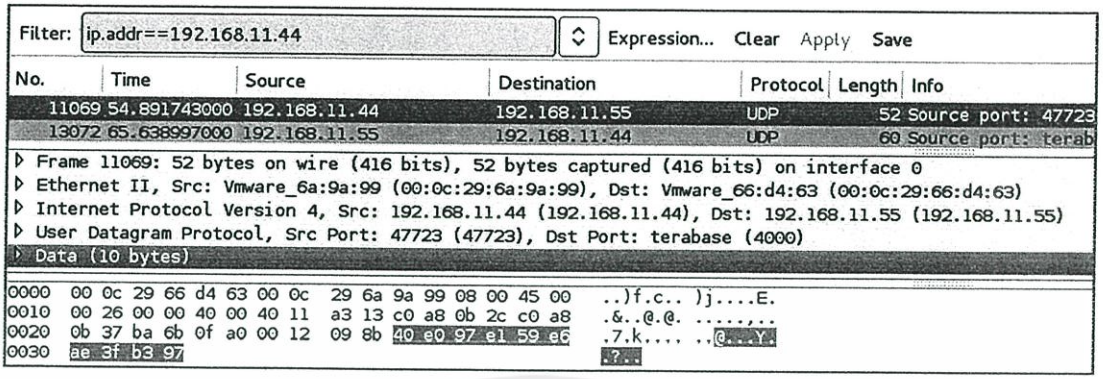
ผลการทดลองส่งข้อมูลคำว่า “Welcome” จากคอมพิวเตอร์ Alice (IP‘192.168.10.30’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.40’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.35



รูปที่ 4.35 การส่งข้อมูลคำว่า “Welcome” จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

4.4.1.2 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

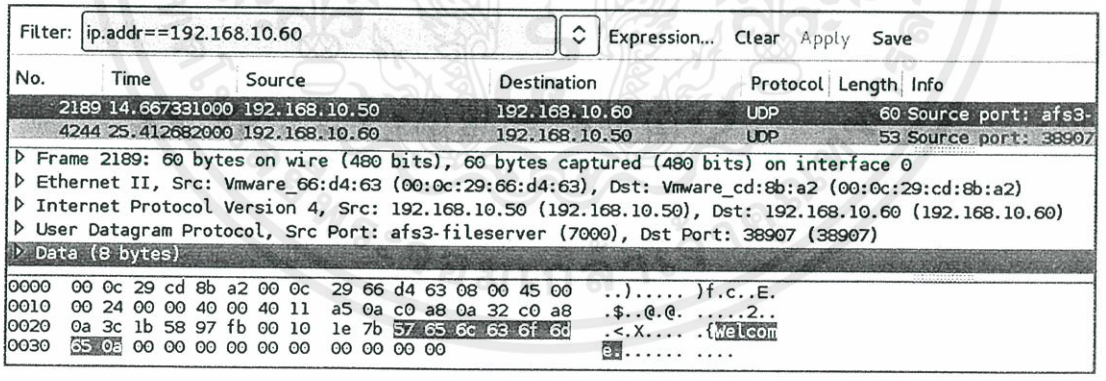
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Welcome” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192.168.11.44’) ไปยังอุปกรณ์ Raspberry Pi 2 (IP ‘192.168.11.55’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.36



รูปที่ 4.36 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Welcome”
จากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

4.4.1.3 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับของข้อมูลที่ได้รับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192. 168.10.50’) ไปยังคอมพิวเตอร์ Bob (IP‘192.168.10.60’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.37

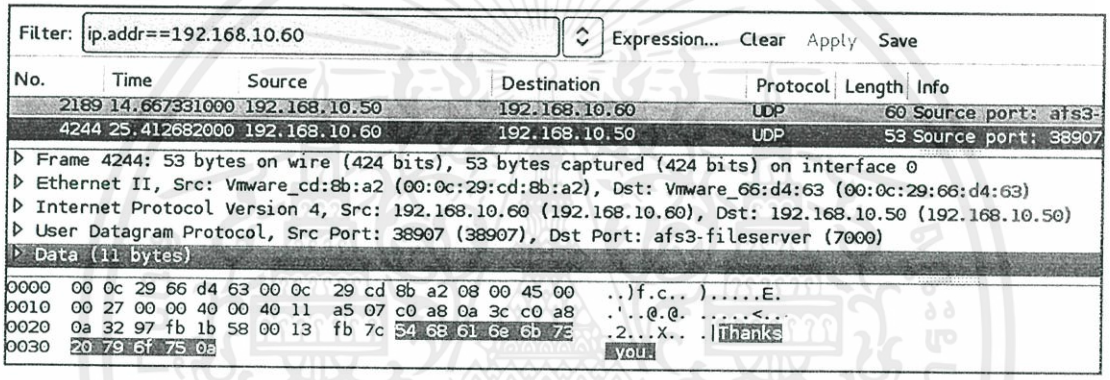


รูปที่ 4.37 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob โดยใช้สัมประสิทธิ์ $c_1 = 4$ และ $c_2 = -1$

ผลการทดลองส่งข้อมูลคำว่า “Welcome” จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob โดยใช้โปรโตคอล UDP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับตรงกัน คอมพิวเตอร์ Bob ได้รับข้อมูลคำว่า “Welcome”

4.4.1.4 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Bob กลับไปอุปกรณ์ Raspberry Pi 2

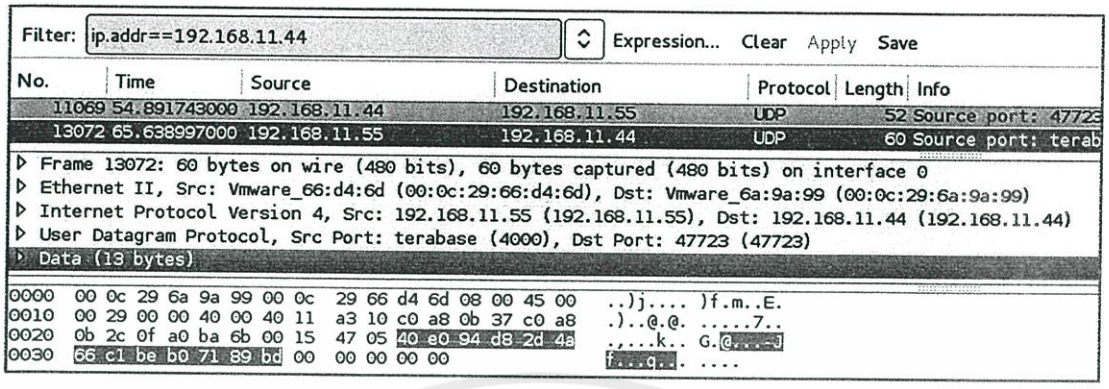
ผลการทดลองส่งข้อมูลคำว่า “Thanks you” จากคอมพิวเตอร์ Bob (IP‘192.168.10.60’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.50’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.38



รูปที่ 4.38 การส่งข้อมูลคำว่า “Thanks you” จากคอมพิวเตอร์ Bob ไปยังอุปกรณ์ Raspberry Pi 2

4.4.1.5 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

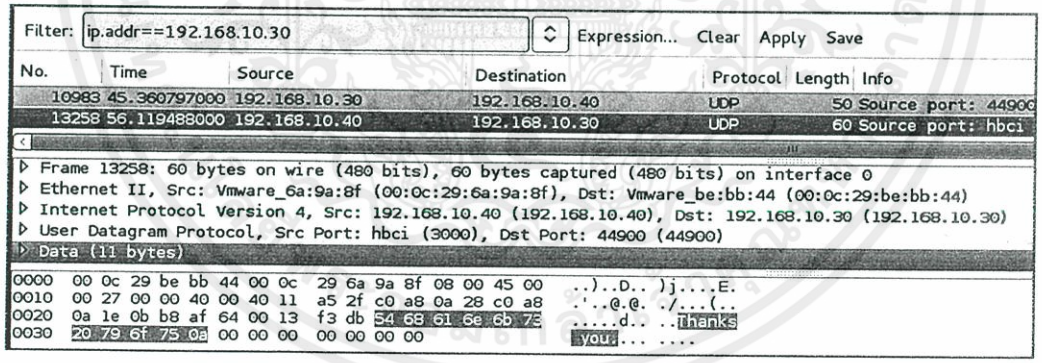
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Thanks you” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192.168.11.55’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.11.44’) โดยใช้โปรแกรม Wireshark ดังรูปที่ 4.39



รูปที่ 4.39 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Thanks you”
จากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

4.4.1.6 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192.168.10.40’) ไปยังคอมพิวเตอร์ Alice (IP‘192.168.10.30’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.40



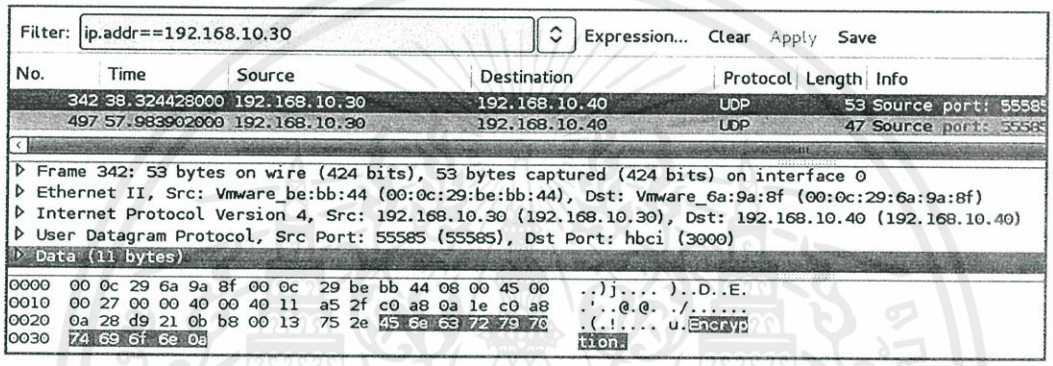
รูปที่ 4.40 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice โดยใช้สัมประสิทธิ์ $c_1 = 4$ และ $c_2 = -1$

ผลการทดลองส่งข้อมูลคำว่า “Thanks you” จากคอมพิวเตอร์ Bob ไปยังคอมพิวเตอร์ Alice โดยใช้โปรโตคอล UDP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับตรงกัน คอมพิวเตอร์ Alice ได้รับข้อมูลคำว่า “Thanks you”

4.4.2 ผลการทดลองโดยใช้โปรโตคอล UDP และใช้สัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขในการเข้ารหัส – ถอดรหัสลับ ไม่ตรงกัน

4.4.2.1 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

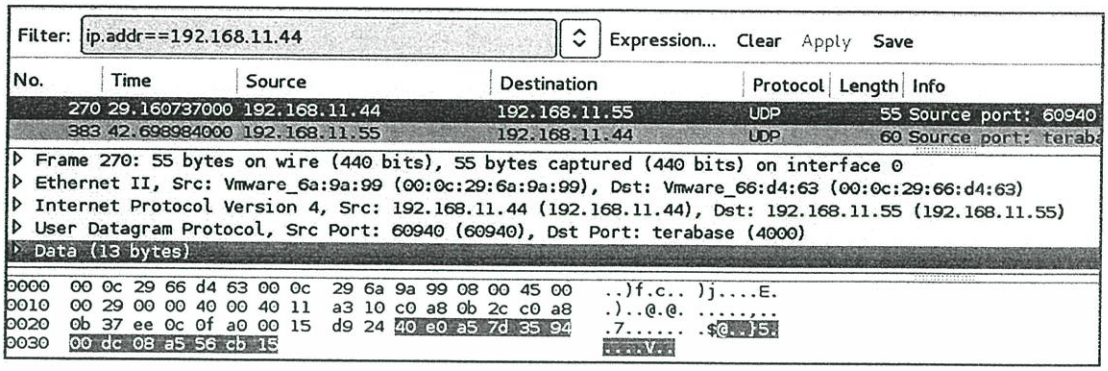
ผลการทดลองส่งข้อมูลคำว่า “Encryption” จากคอมพิวเตอร์ Alice (IP‘192.168.10.30’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.40’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.41



รูปที่ 4.41 การส่งข้อมูลคำว่า “Encryption” จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

4.4.2.2 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

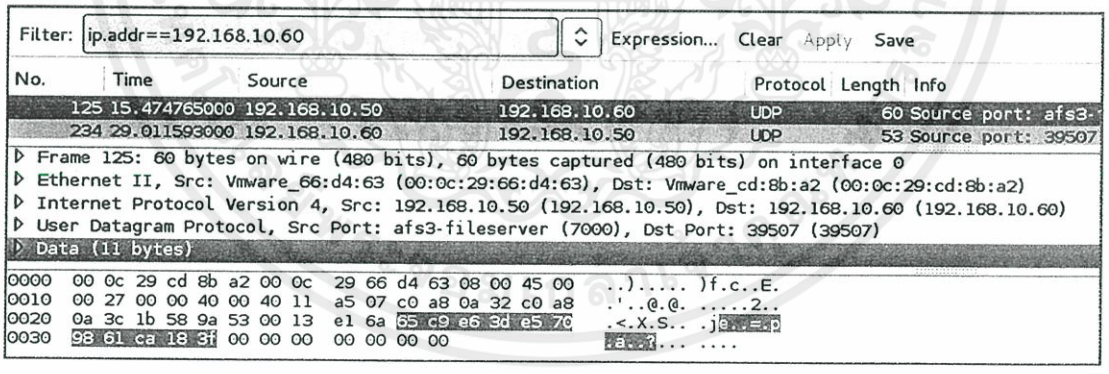
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Encryption” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192.168.11.44’) ไปยังอุปกรณ์ Raspberry Pi 2 (IP ‘192.168.11.55’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.42



รูปที่ 4.42 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Encryption” จากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

4.4.2.3 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับของข้อมูลที่ได้รับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = -1$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192. 168.10.50’) ไปยังคอมพิวเตอร์ Bob (IP‘192.168.10.60’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.43

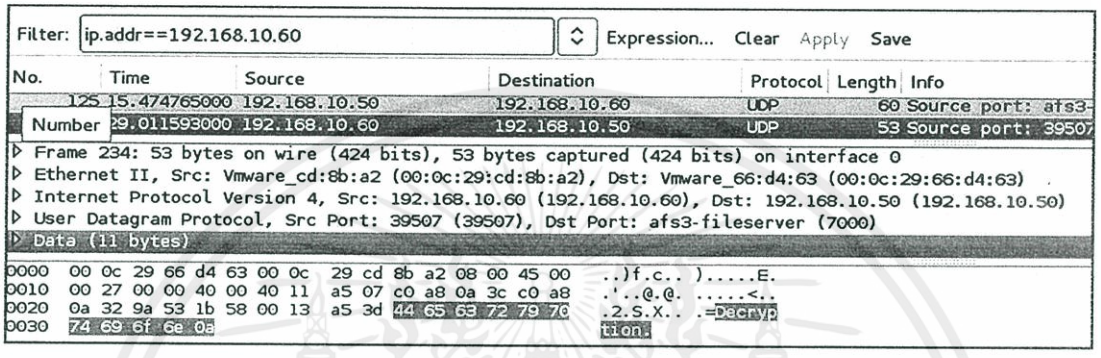


รูปที่ 4.43 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob โดยใช้สัมประสิทธิ์ $c_1 = -1$ และ $c_2 = 1$

ผลการทดลองส่งข้อมูลคำว่า “Encryption” จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob โดยใช้โปรโตคอล UDP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับไม่ตรงกัน คอมพิวเตอร์ Bob ไม่ได้รับข้อมูลคำว่า “Encryption”

4.4.2.4 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Bob กลับไปอุปกรณ์ Raspberry Pi 2

ผลการทดลองส่งข้อมูลคำว่า “Decryption” จากคอมพิวเตอร์ Bob (IP‘192.168.10.60’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.50’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.44



รูปที่ 4.44 การส่งข้อมูลคำว่า “Decryption” จากคอมพิวเตอร์ Bob ไปยังอุปกรณ์ Raspberry Pi 2

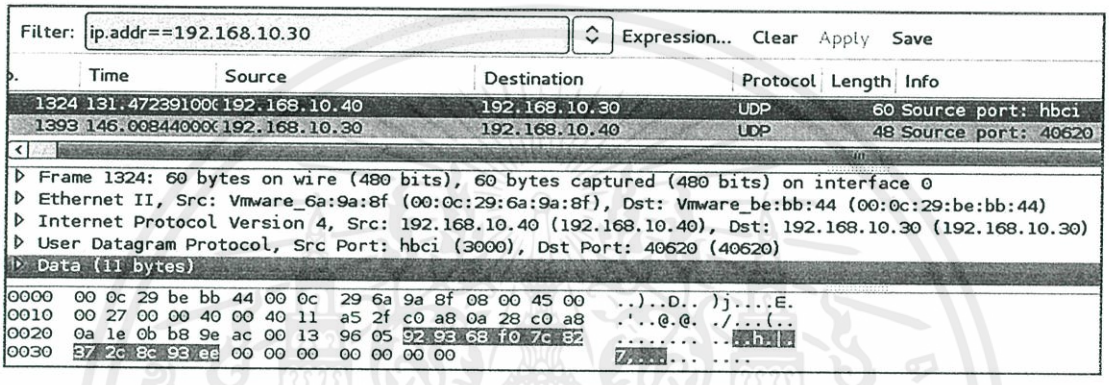
4.4.2.5 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Telecommunication” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192. 168.11.55’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.11.44’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.45

รูปที่ 4.45 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Decryption” จากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

4.4.2.6 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = -1$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP'192.168.10.40') ไปยังคอมพิวเตอร์ Alice (IP'192.168.10.30') โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.46

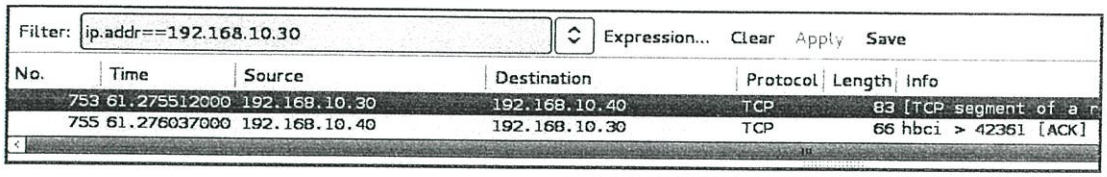


รูปที่ 4.46 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice โดยใช้สัมประสิทธิ์ $c_1 = -1$ และ $c_2 = 1$

ผลการทดลองส่งข้อมูลคำว่า "Decryption" จากคอมพิวเตอร์ Bob ไปยังคอมพิวเตอร์ Alice โดยใช้โปรโตคอล UDP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับไม่ตรงกัน คอมพิวเตอร์ Alice ไม่ได้รับข้อมูลคำว่า "Decryption"

4.4.3 ผลการทดลองโดยใช้โปรโตคอล TCP และใช้สัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขในการเข้ารหัส – ถอดรหัสลับ ตรงกัน

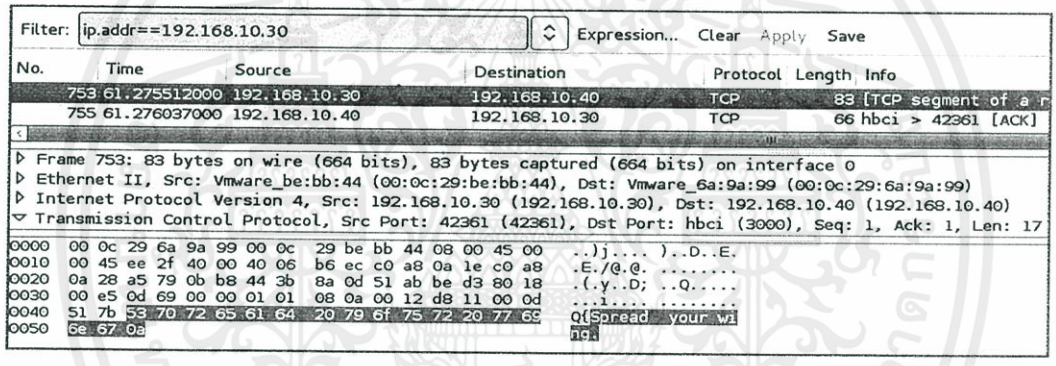
การทดลองโดยใช้โปรโตคอล TCP เมื่อส่งแพ็คเก็จข้อมูลไปยังปลายทาง ปลายทางจะส่งแพ็คเก็จยืนยันว่าได้รับข้อมูลที่ส่งจากต้นทางแล้ว เมื่อเก็บผลโดยใช้โปรแกรม Wireshark แพ็คเก็จยืนยันจะปรากฏคำว่า [ACK] แสดงดังรูปที่ 4.47



รูปที่ 4.47 แพ็คเก็จยืนยันสำหรับโปรโตคอล TCP

4.4.3.1 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

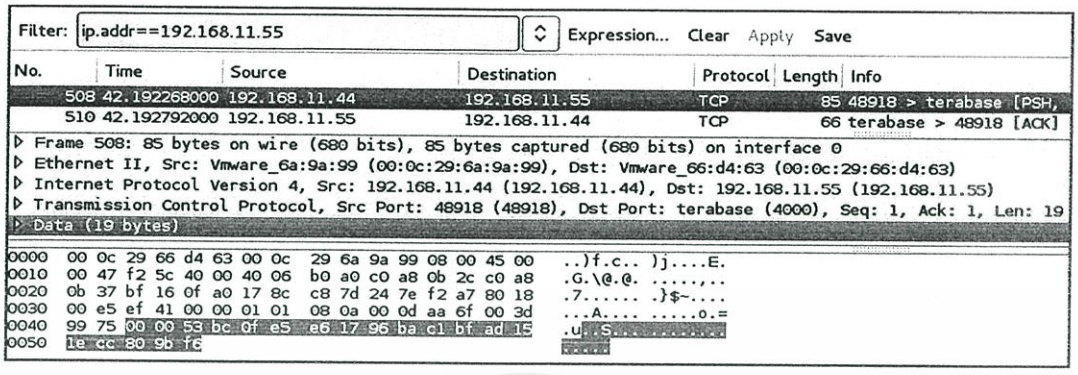
ผลการทดลองส่งข้อมูลคำว่า “Spread your wing” จากคอมพิวเตอร์ Alice (IP ‘192.168.10.30’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.40’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.48



รูปที่ 4.48 การส่งข้อมูลคำว่า “Spread your wing” จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

4.4.3.2 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

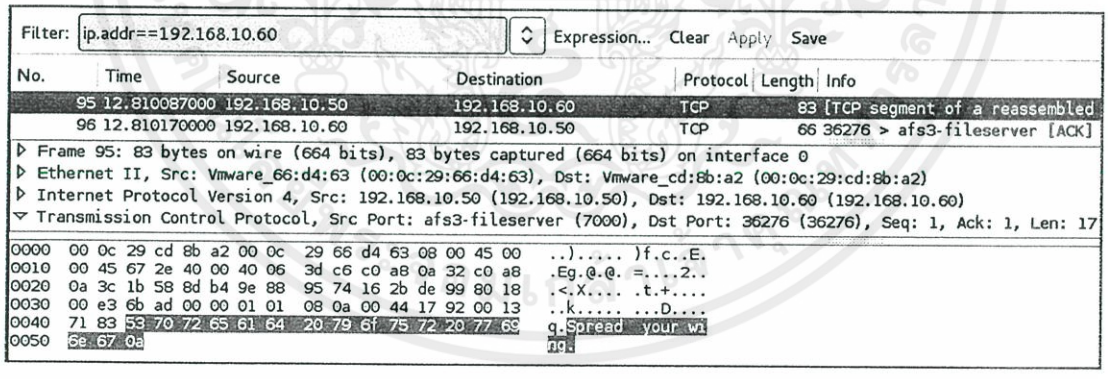
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Spread your wing” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192.168.11.44’) ไปยังอุปกรณ์ Raspberry Pi 2 (IP ‘192.168.11.55’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.49



รูปที่ 4.49 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Spread your wing” จากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

4.4.3.3 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับของข้อมูลที่ได้รับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192. 168.10.50’) ไปยังคอมพิวเตอร์ Bob (IP‘192.168.10.60’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.50

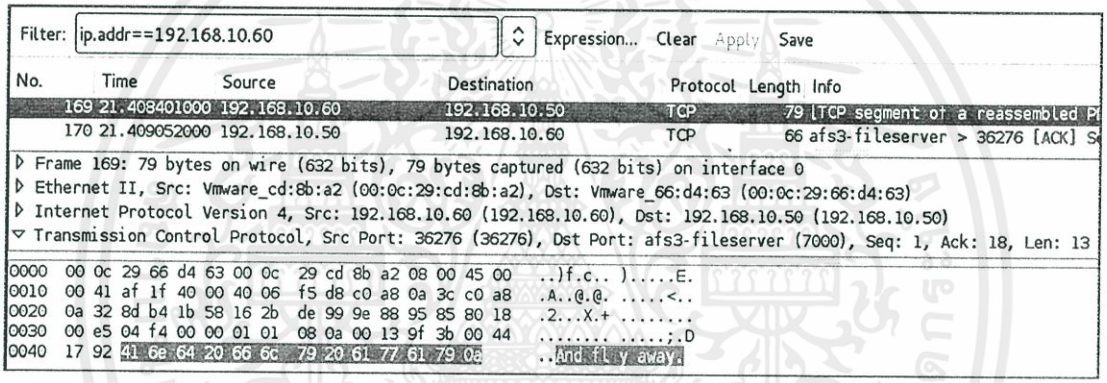


รูปที่ 4.50 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob โดยใช้สัมประสิทธิ์ $c_1 = -2$ และ $c_2 = 1$

ผลการทดลองส่งข้อมูลคำว่า “Spread your wing” จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob โดยใช้โปรโตคอล TCP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับตรงกัน คอมพิวเตอร์ Bob ได้รับข้อมูลคำว่า “Spread your wing”

4.4.3.4 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Bob กลับไปอุปกรณ์ Raspberry Pi 2

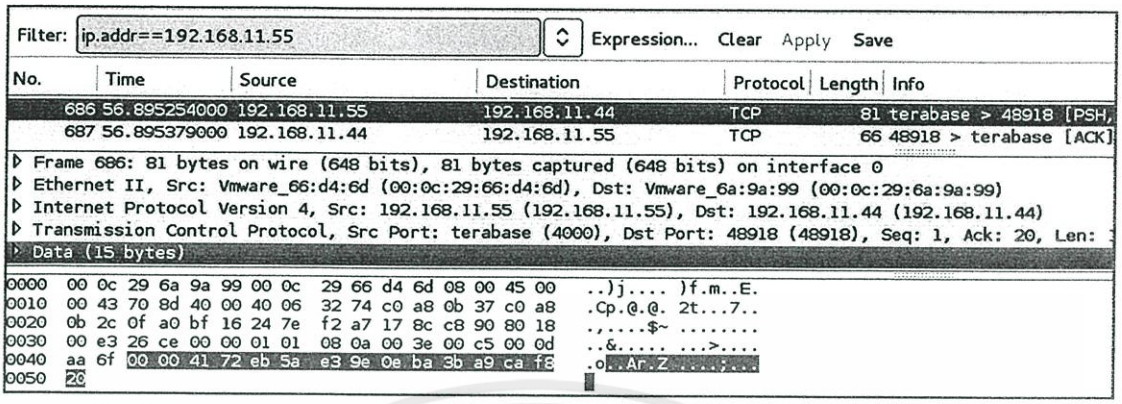
ผลการทดลองส่งข้อมูลคำว่า “And fly away” จากคอมพิวเตอร์ Bob (IP‘192. 168.10.60’) ไปยังอุปกรณ์ Raspberry Pi 2 (IP‘192.168.10.50’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.51



รูปที่ 4.51 การส่งข้อมูลคำว่า “And fly away” จากคอมพิวเตอร์ Bob ไปยังอุปกรณ์ Raspberry Pi 2

4.4.3.5 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

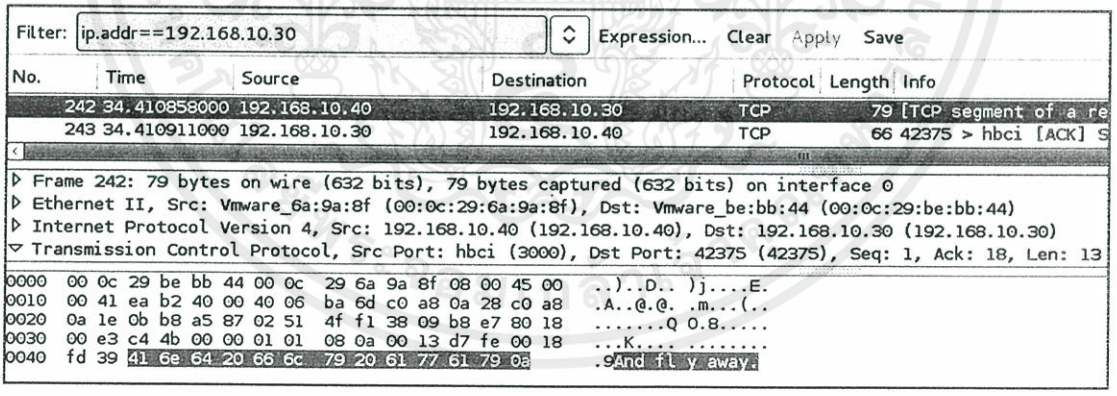
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “And fly away” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192. 168.11.55’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.11.44’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.52



รูปที่ 4.52 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “And fly away” จากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

4.4.3.6 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192.168.10.40’) ไปยังคอมพิวเตอร์ Alice (IP‘192.168.10.30’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.53



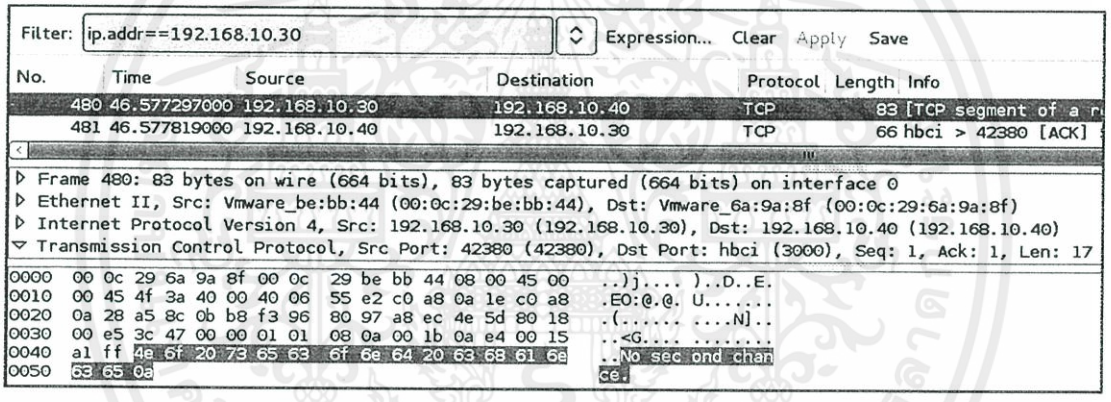
รูปที่ 4.53 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice โดยใช้สัมประสิทธิ์ $c_1 = -2$ และ $c_2 = 1$

ผลการทดลองส่งข้อมูลคำว่า “And fly away” จากคอมพิวเตอร์ Bob ไปยังคอมพิวเตอร์ Alice โดยใช้โปรโตคอล TCP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับตรงกัน คอมพิวเตอร์ Alice ได้รับข้อมูลคำว่า “And fly away”

4.4.4 ผลการทดลองโดยใช้โปรโตคอล TCP และใช้สัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขในการเข้ารหัส – ถอดรหัสลับ ไม่ตรงกัน

4.4.4.1 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

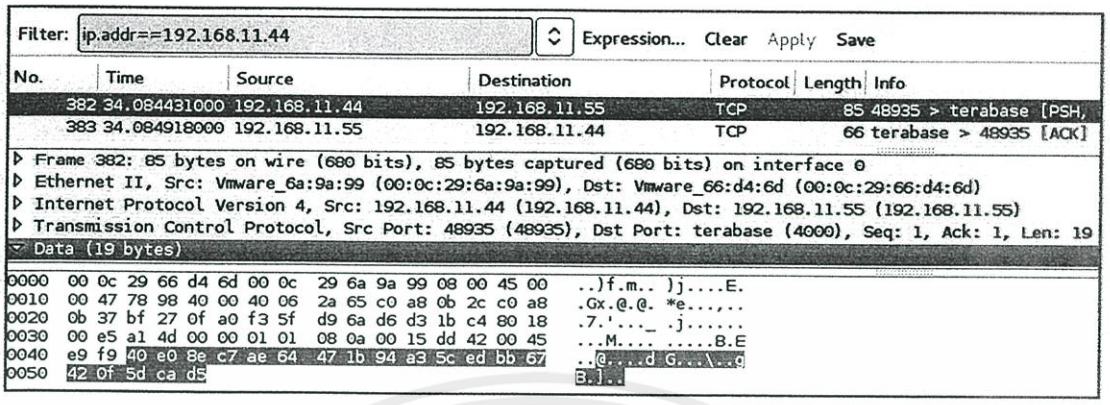
ผลการทดลองส่งข้อมูลคำว่า “No second chance” จากคอมพิวเตอร์ Alice (IP ‘192.168.10.30’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.40’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.54



รูปที่ 4.54 การส่งข้อมูลคำว่า “No second chance” จากคอมพิวเตอร์ Alice ไปยังอุปกรณ์ Raspberry Pi 1

4.4.4.2 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

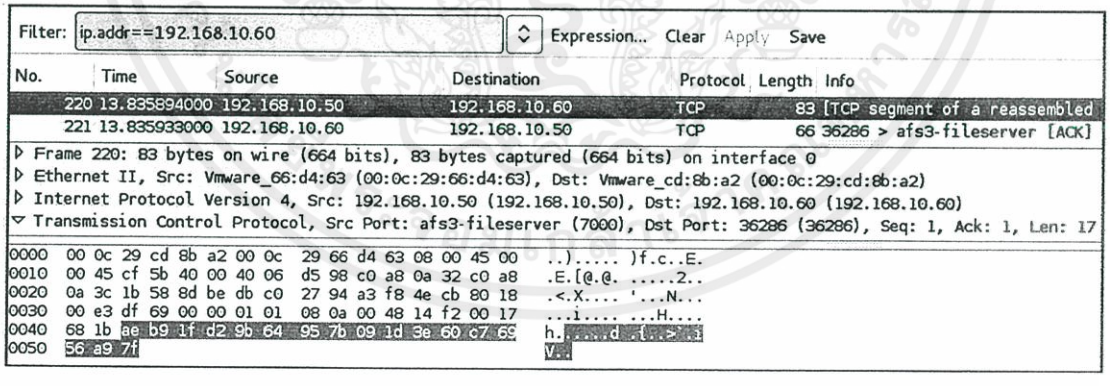
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “No second chance” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192 168.11.44’) ไปยังอุปกรณ์ Raspberry Pi 2 (IP ‘192.168.11.55’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.55



รูปที่ 4.55 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “No second chance” จากอุปกรณ์ Raspberry Pi 1 ไปยังอุปกรณ์ Raspberry Pi 2

4.4.4.3 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับของข้อมูลที่ได้รับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192. 168.10.50’) ไปยังคอมพิวเตอร์ Bob (IP‘192.168.10.60’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.56

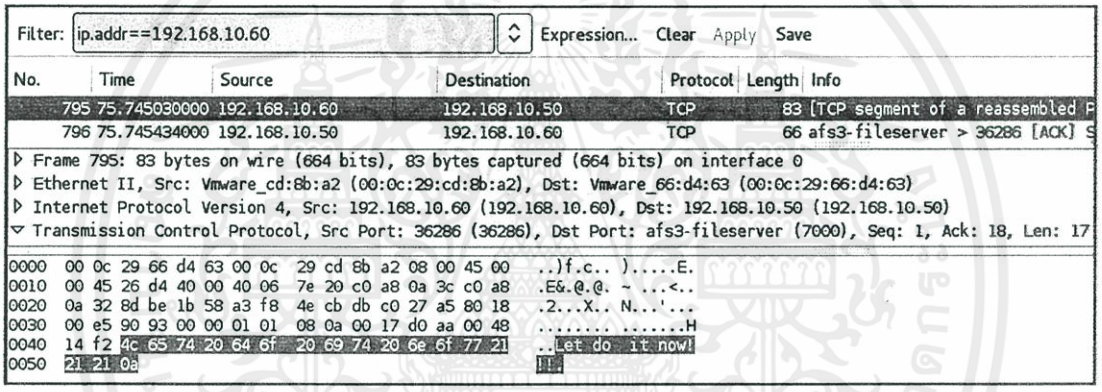


รูปที่ 4.56 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 2 ไปยังคอมพิวเตอร์ Bob โดยใช้สัมประสิทธิ์ $c_1 = 4$ และ $c_2 = -1$

ผลการทดลองส่งข้อมูลคำว่า “No second chance” จากคอมพิวเตอร์ Alice ไปยังคอมพิวเตอร์ Bob โดยใช้โปรโตคอล TCP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับไม่ตรงกัน คอมพิวเตอร์ Bob ไม่ได้รับข้อมูลคำว่า “No second chance”

4.4.4.4 ผลการทดลองการส่งข้อมูลจากคอมพิวเตอร์ Bob กลับไปอุปกรณ์ Raspberry Pi 2

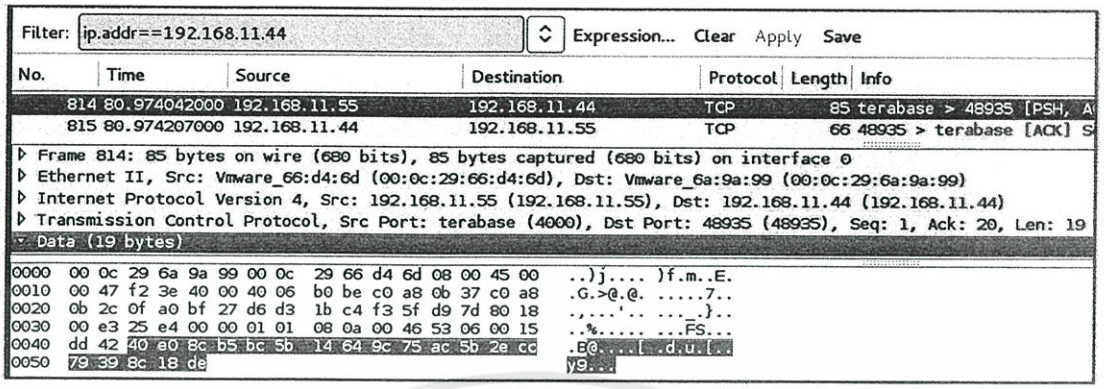
ผลการทดลองส่งข้อมูลคำว่า “Let do it now!!!” จากคอมพิวเตอร์ Bob (IP ‘192.168.10.60’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.10.50’) ผ่านพอร์ตอินเตอร์เน็ต eth0 โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.57



รูปที่ 4.57 การส่งข้อมูลคำว่า “Let do it now!!!” จากคอมพิวเตอร์ Bob ไปยังอุปกรณ์ Raspberry Pi 2

4.4.4.5 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

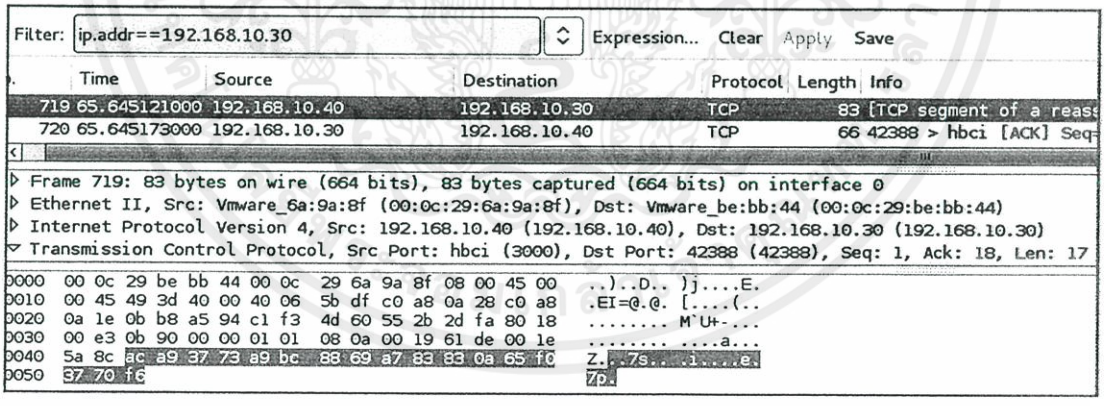
ผลการส่งข้อมูลที่ผ่านการเข้ารหัสลับของข้อมูลคำว่า “Let do it now!!!” โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์ไม่จำกัด $c_1 = -2$ และ $c_2 = 1$ จากอุปกรณ์ Raspberry Pi 2 ผ่านพอร์ตอินเตอร์เน็ต eth1 (IP‘192.168.11.55’) ไปยังอุปกรณ์ Raspberry Pi 1 (IP‘192.168.11.44’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.58



รูปที่ 4.58 การส่งข้อมูลที่ผ่านการเข้ารหัสลับคำว่า “Let do it now!!!” จากอุปกรณ์ Raspberry Pi 2 ไปยังอุปกรณ์ Raspberry Pi 1

4.4.4.6 ผลการส่งข้อมูลจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice

ผลการส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 โดยใช้ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขอันดับสองชนิดอิมพัลส์จำกัด $c_1 = 4$ และ $c_2 = -1$ จากอุปกรณ์ Raspberry Pi 1 ผ่านพอร์ตอินเตอร์เน็ต eth0 (IP‘192.168.10.40’) ไปยังคอมพิวเตอร์ Alice (IP‘192.168.10.30’) โดยใช้โปรแกรม Wireshark แสดงดังรูปที่ 4.59

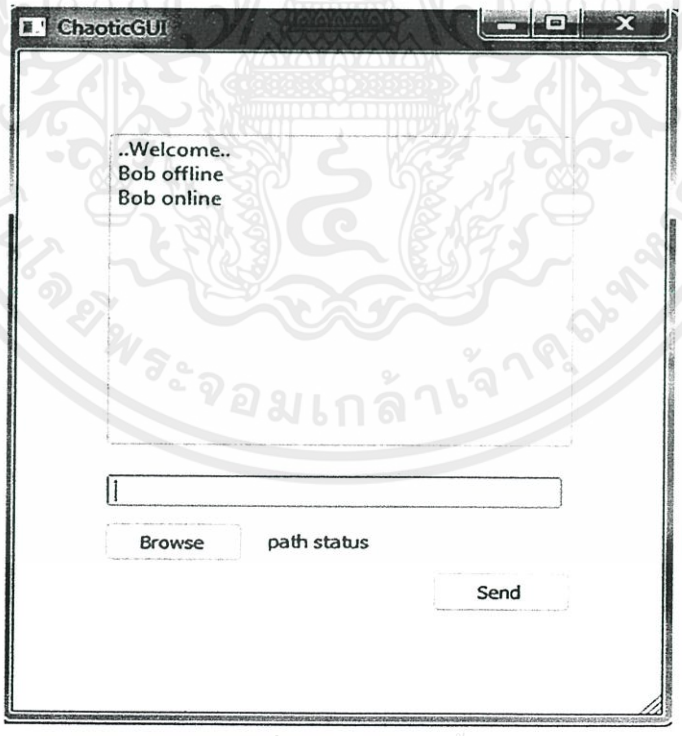


รูปที่ 4.59 การส่งข้อมูลที่ผ่านการถอดรหัสลับจากอุปกรณ์ Raspberry Pi 1 ไปยังคอมพิวเตอร์ Alice โดยใช้สัมประสิทธิ์ $c_1 = 4$ และ $c_2 = -1$

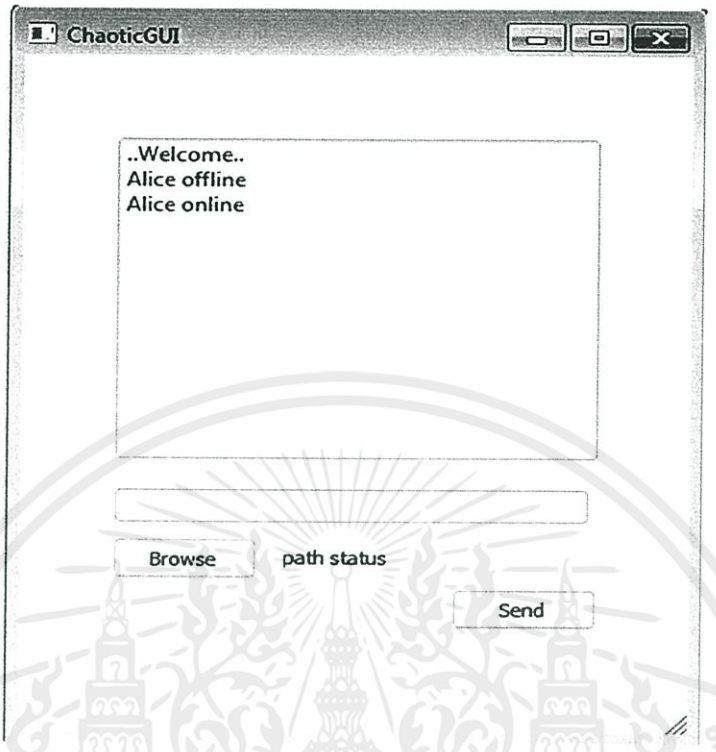
ผลการทดลองส่งข้อมูลคำว่า “Let do it now!!!” จากคอมพิวเตอร์ Bob ไปยังคอมพิวเตอร์ Alice โดยใช้โปรโตคอล TCP และสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและถอดรหัสลับไม่ตรงกัน คอมพิวเตอร์ Alice ไม่ได้รับข้อมูลคำว่า “Let do it now!!!”

4.5 ผลการออกแบบ GUI บนคอมพิวเตอร์

หลังจากสร้างหน้าต่างโปรแกรม Chaotic GUI บนคอมพิวเตอร์ Alice และ Bob แล้ว จึงทำการเชื่อมต่อตามรูปที่ 4.33 การสื่อสารผ่านเครือข่ายอินเตอร์ระหว่าง Alice กับ Bob โดยผ่านอุปกรณ์เข้ารหัส - ถอดรหัสลับ Raspberry Pi (กล่าวไว้ในหัวข้อที่ 4.4) โดยเริ่มให้คอมพิวเตอร์ Bob เปิดหน้าต่างโปรแกรม ChaoticGUI ขึ้นมารอไว้จะพบคำว่า “..Welcome..” และ “Alice offline” และรอการตอบกลับจากคอมพิวเตอร์ Alice เมื่อ Alice เปิดหน้าต่างโปรแกรม ChaoticGUI ขึ้นมาจะพบกับคำว่า “..Welcome..” และ “Bob offline” พร้อมกับตอบกลับไปยังคอมพิวเตอร์ Bob ที่คอมพิวเตอร์ของ Bob โปรแกรมก็จะแสดงสถานะว่า “Alice online” ขึ้นมา และที่คอมพิวเตอร์ Alice โปรแกรมก็จะแสดงสถานะว่า “Bob online” เช่นกัน แสดงว่าคอมพิวเตอร์ทั้งสองต่างทราบกันว่า อีกฝ่ายพร้อมที่จะสื่อสารข้อมูลด้วยแล้ว สำหรับหน้าต่าง ChaoticGUI บนคอมพิวเตอร์ Alice แสดงดังรูปที่ 4.60 และหน้าต่าง ChaoticGUI บนคอมพิวเตอร์ Bob แสดงดังรูปที่ 4.61



รูปที่ 4.60 หน้าต่าง GUI ของคอมพิวเตอร์ Alice ก่อนเริ่มรับส่งข้อมูล

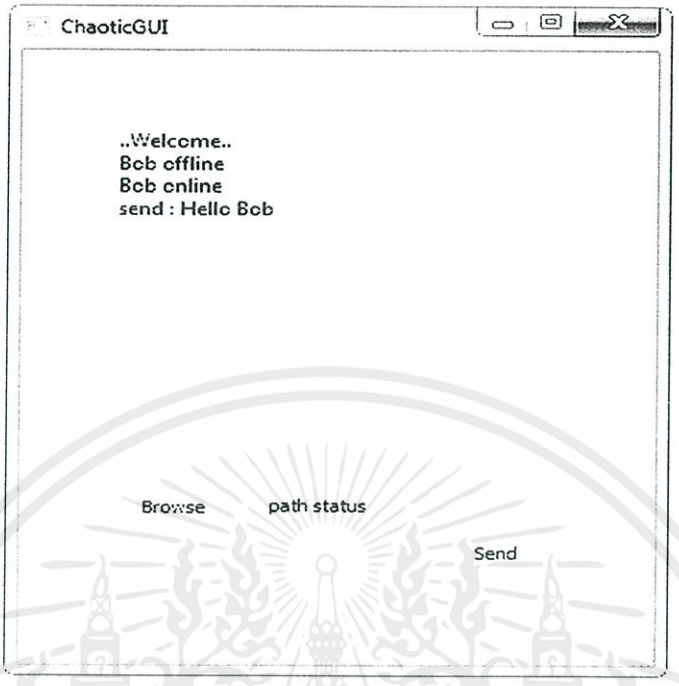


รูปที่ 4.61 หน้าต่าง GUI ของคอมพิวเตอร์Bob ก่อนเริ่มรับส่งข้อมูล

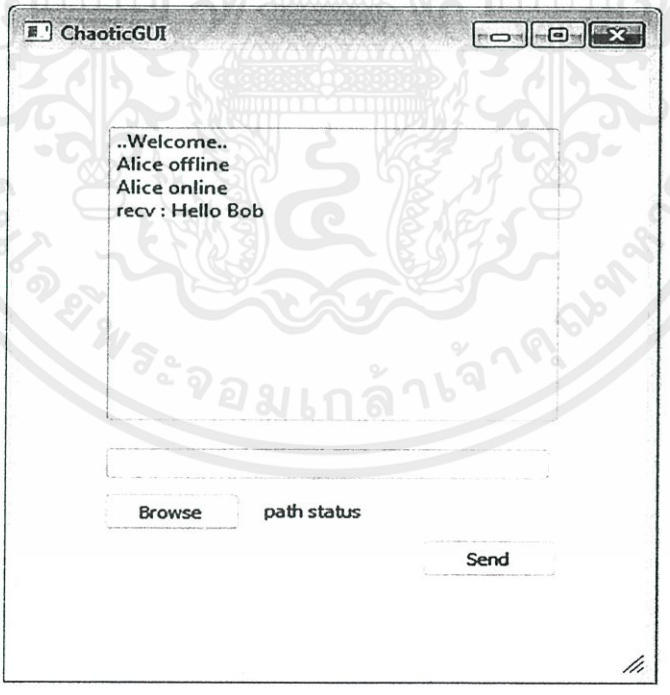
4.5.1 ผลการทดลองรับส่งข้อมูล Text

ในกรณีใช้สัมประสิทธิของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและฝั่งถอดรหัสลับ ตรงกัน เริ่มการส่งข้อมูลโดยคอมพิวเตอร์ Alice ส่งข้อความคำว่า “Hello Bob”แสดงดังรูปที่ 4.62 ข้อมูลจะผ่านการเพิ่มเฮดเดอร์ของข้อมูลชนิดข้อความ ก่อนส่งไปยังอุปกรณ์ Raspberry Pi 1 จากนั้นข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ก็จะได้รับข้อความว่า “Hello Bob” แสดงดังรูปที่ 4.63 ส่วนการทดลองให้คอมพิวเตอร์ Bob ส่งข้อความกลับไปยังคอมพิวเตอร์ Alice ได้ผลการทดลองเช่นเดียวกัน คือ ข้อความฝั่งส่งและฝั่งรับตรงกัน

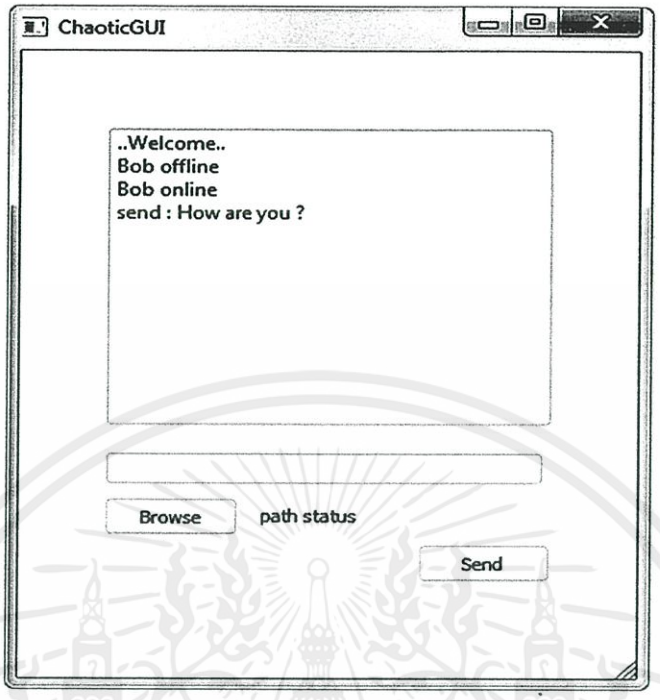
ในกรณีใช้สัมประสิทธิของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและฝั่งถอดรหัสลับ ไม่ตรงกัน เริ่มการส่งข้อมูลโดยคอมพิวเตอร์ Alice ส่งข้อความคำว่า “How are you today ?” แสดงดังรูปที่ 4.64 ข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ไม่ได้รับข้อความคำว่า “How are you today ?” แสดงดังรูปที่ 4.65



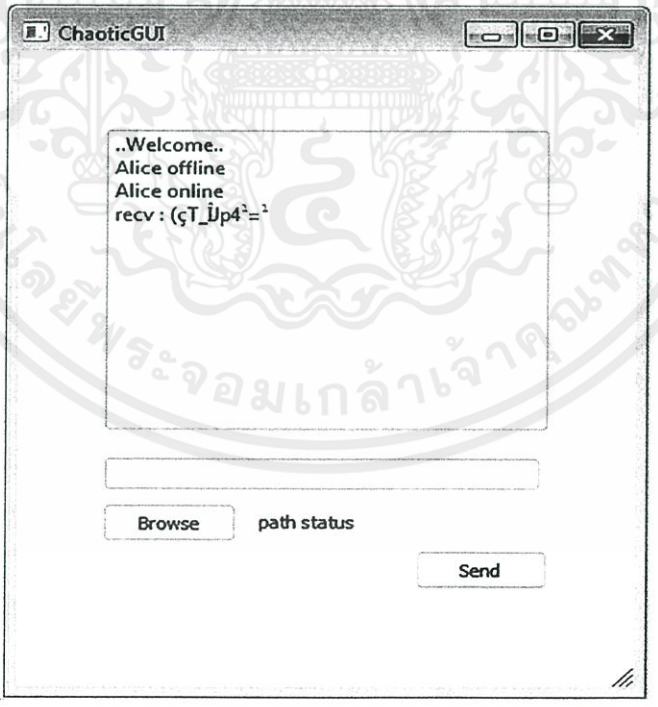
รูปที่ 4.62 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งข้อความไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ตรงกัน)



รูปที่ 4.63 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับข้อความจากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ตรงกัน)



รูปที่ 4.64 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งข้อความไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ไม่ตรงกัน)

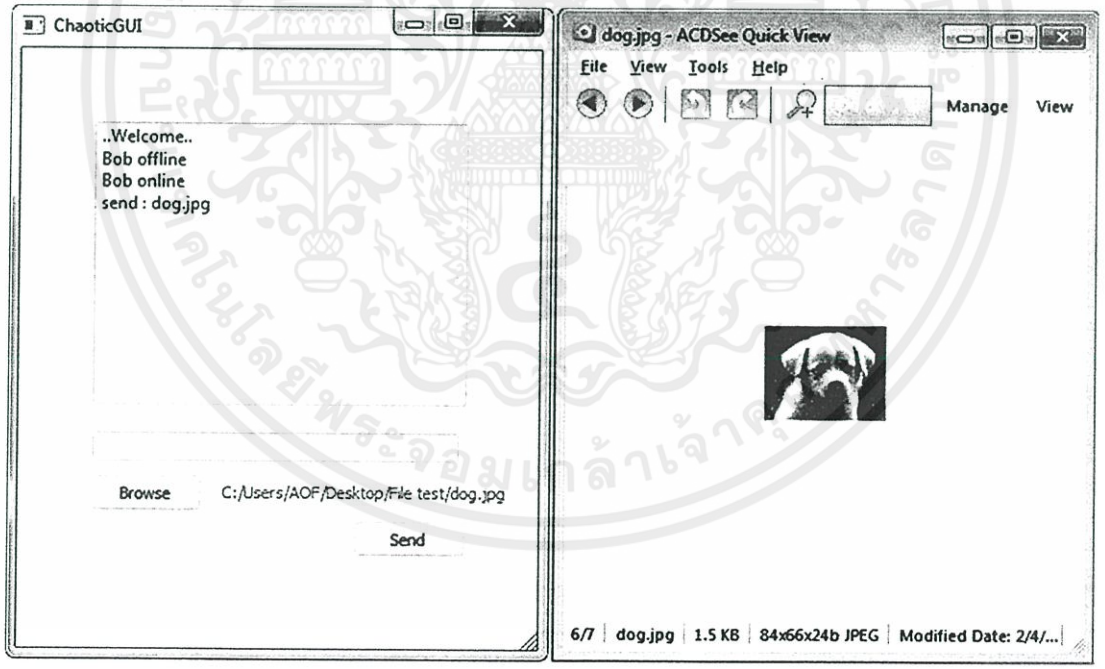


รูปที่ 4.65 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับข้อความจากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ไม่ตรงกัน)

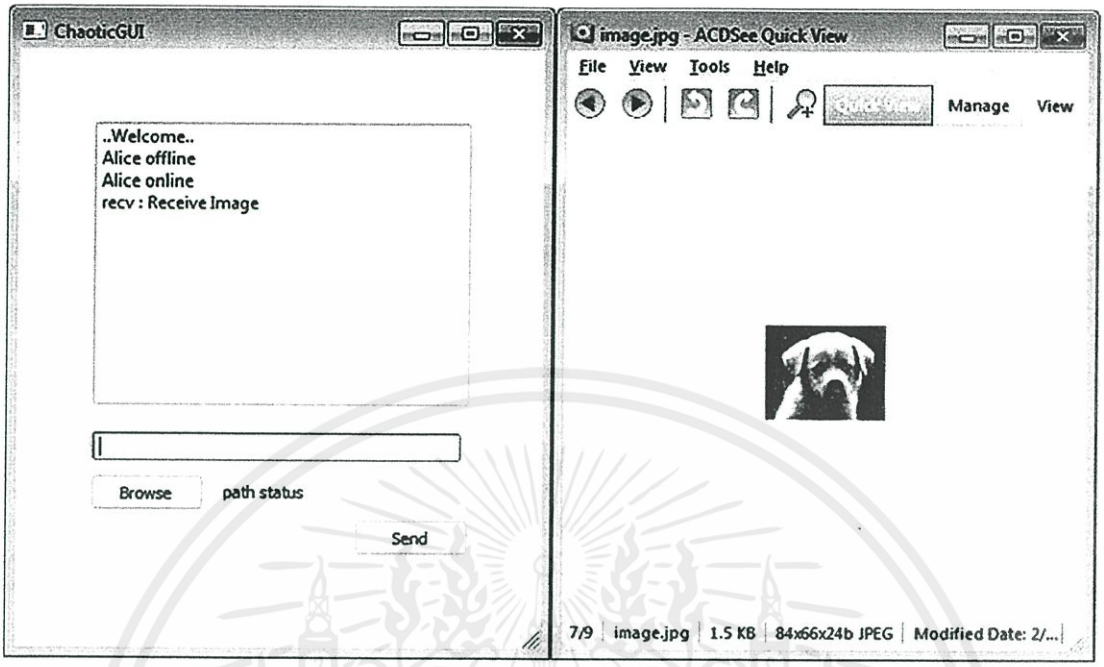
4.5.2 ผลการทดลองรับส่งข้อมูลภาพ (Image data)

ในกรณีใช้สัมประสิทธิของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและฝั่งถอดรหัสลับ ตรงกัน เริ่มการส่งข้อมูลโดยคอมพิวเตอร์ Alice เลือกภาพที่ต้องการส่งโดยกดปุ่ม Browse และกดปุ่ม Send เพื่อส่งข้อมูลภาพ dog.jpg แสดงดังรูปที่ 4.66 ข้อมูลจะผ่านการเพิ่มเฮดเดอร์ของข้อมูลภาพ ก่อนส่งไปยังอุปกรณ์ Raspberry Pi 1 จากนั้นข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ได้รับภาพที่เหมือนกับฝั่งส่ง และแสดงคำว่า "recv : receive image" บนหน้าต่าง GUI แสดงดังรูปที่ 4.67 ส่วนการทดลองให้คอมพิวเตอร์ Bob ส่งข้อมูลภาพกลับไปยังคอมพิวเตอร์ Alice ได้ผลการทดลองเช่นเดียวกัน คือ ข้อมูลภาพฝั่งส่งและฝั่งรับตรงกัน

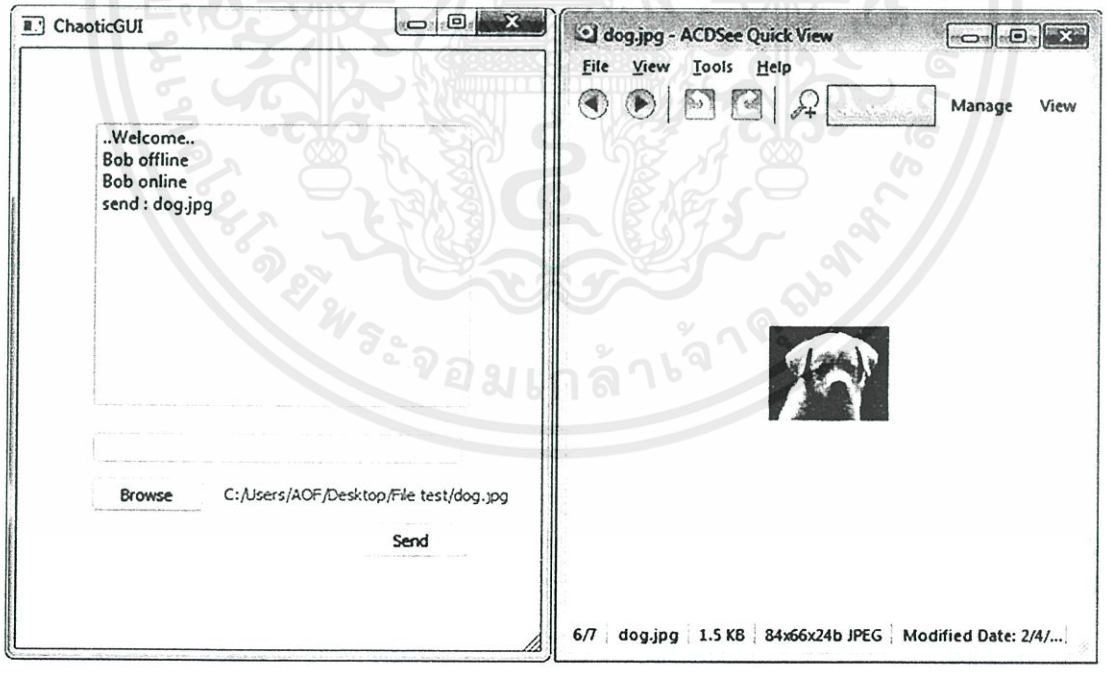
ในกรณีใช้สัมประสิทธิของวงจรเข้ารหัสลับและวงจรถอดรหัสลับ ไม่ตรงกัน เริ่มการส่งข้อมูลโดยคอมพิวเตอร์ Alice ส่งข้อมูลภาพ dog.jpg แสดงดังรูปที่ 4.68 ข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ได้รับข้อมูลภาพที่ไม่เหมือนกับฝั่งส่ง และแสดงคำว่า "recv : receive image" บนหน้าต่าง GUI แสดงดังรูปที่ 4.69



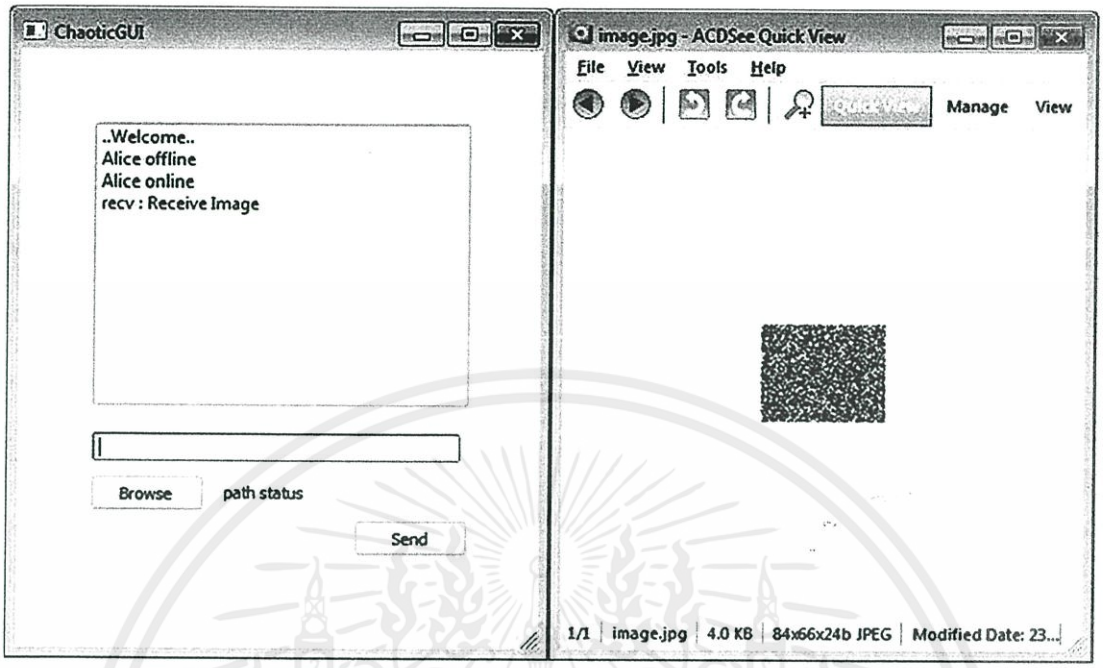
รูปที่ 4.66 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งข้อมูลภาพ
ไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ตรงกัน)



รูปที่ 4.67 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับข้อมูลภาพ
จากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ตรงกัน)



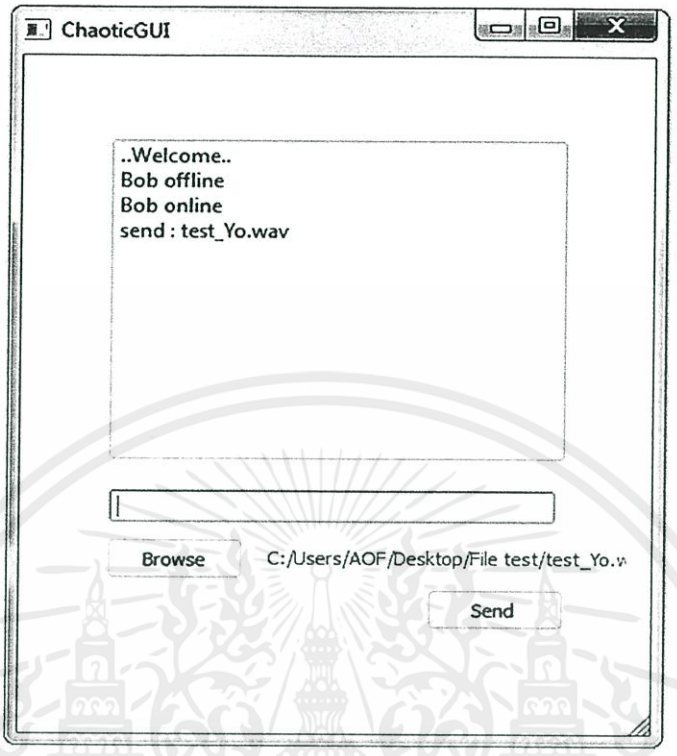
รูปที่ 4.68 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งข้อมูลภาพ
ไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ไม่ตรงกัน)



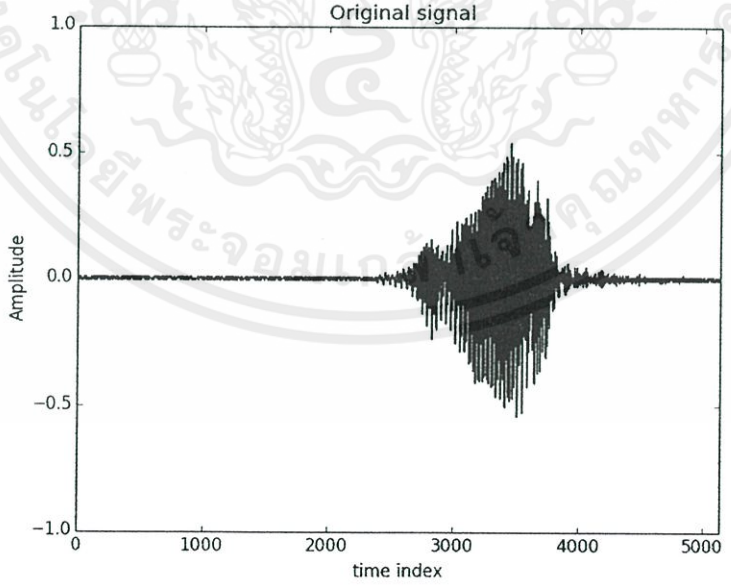
รูปที่ 4.69 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับข้อมูลภาพ
จากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ไม่ตรงกัน)

4.5.3 ผลการทดลองรับส่งข้อมูลเสียง (Sound Data)

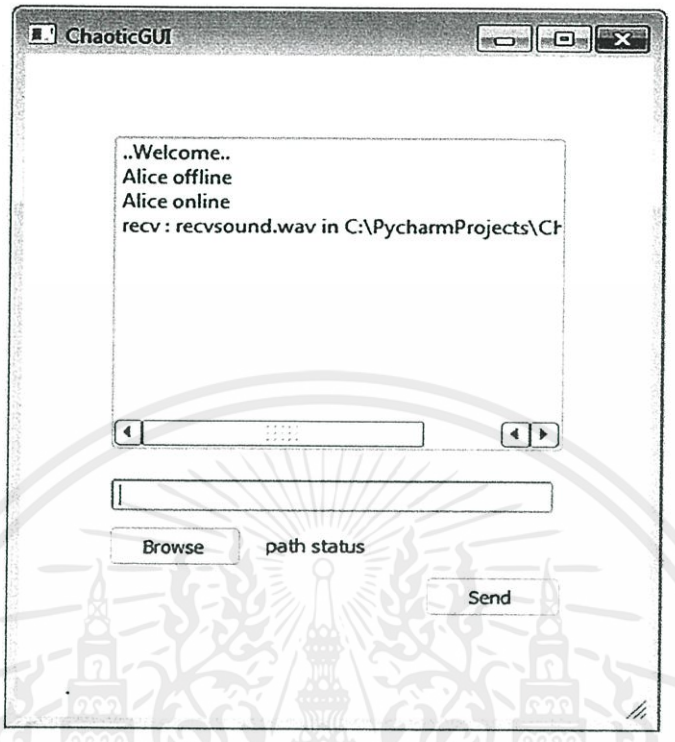
ในกรณีใช้สัมประสิทธิของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและฝั่งถอดรหัสลับ ตรงกัน การส่งข้อมูลเสียงเราจะให้คอมพิวเตอร์ Alice ทำการเลือกไฟล์เสียงที่ต้องการส่งโดยการกดปุ่ม Browse ช่อง path status จะแสดงที่อยู่ของไฟล์เสียงนั้นขึ้นมา แสดงดังรูปที่ 4.70 นำไฟล์เสียงที่ส่งไปพล็อตกราฟระหว่างแกนเวลา (แกน x) และแอมปลิจูด (แกน y) แสดงดังรูปที่ 4.71 แล้วกดปุ่ม Send เพื่อส่งไฟล์เสียง ข้อมูลจะผ่านการเพิ่มเฮดเดอร์ของข้อมูลเสียง ก่อนส่งไปยังอุปกรณ์ Raspberry Pi 1 จากนั้นข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ได้รับไฟล์เสียงโดยเก็บข้อมูลที่ได้ในไฟล์ response.wav และแสดงคำว่า “recv : recvsound.wav in C:\PycharmProjects\Chaos_GUI\dist” บนหน้าต่าง GUI แสดงดังรูปที่ 4.72 นำไฟล์เสียงที่ได้รับ (response.wav) ไปพล็อตกราฟระหว่างแกนเวลา (แกน x) และแอมปลิจูด (แกน y) แสดงดังรูปที่ 4.73 เมื่อเปรียบเทียบกราฟของไฟล์เสียงของฝั่งส่งและฝั่งรับสรุปได้ว่าเป็นสัญญาณเดียวกัน ส่วนการทดลองให้คอมพิวเตอร์ Bob ส่งข้อมูลภาพกลับไปยังคอมพิวเตอร์ Alice ได้ผลการทดลองเช่นเดียวกัน คือ ข้อมูลเสียงฝั่งส่งและฝั่งรับตรงกัน



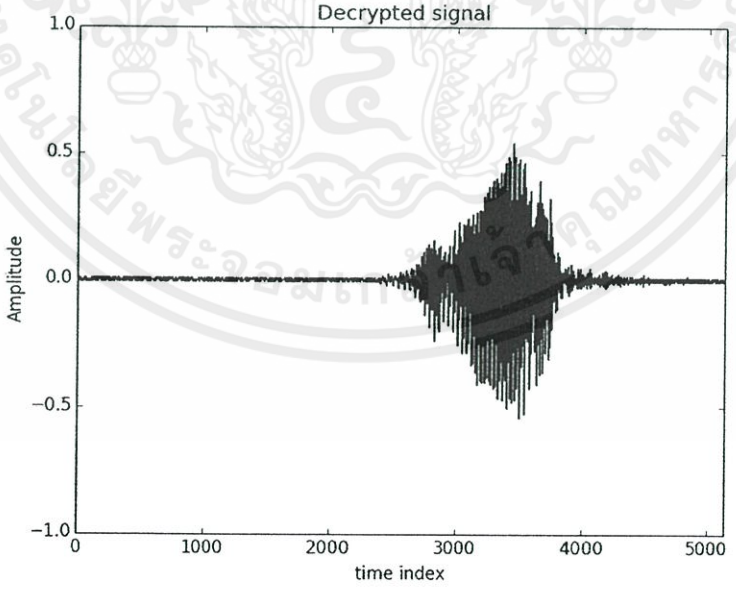
รูปที่ 4.70 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งไฟล์เสียงไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ตรงกัน)



รูปที่ 4.71 กราฟของสัญญาณเสียงที่คอมพิเตอร์ Alice ส่ง (ค่าสัมประสิทธิ์ตรงกัน)

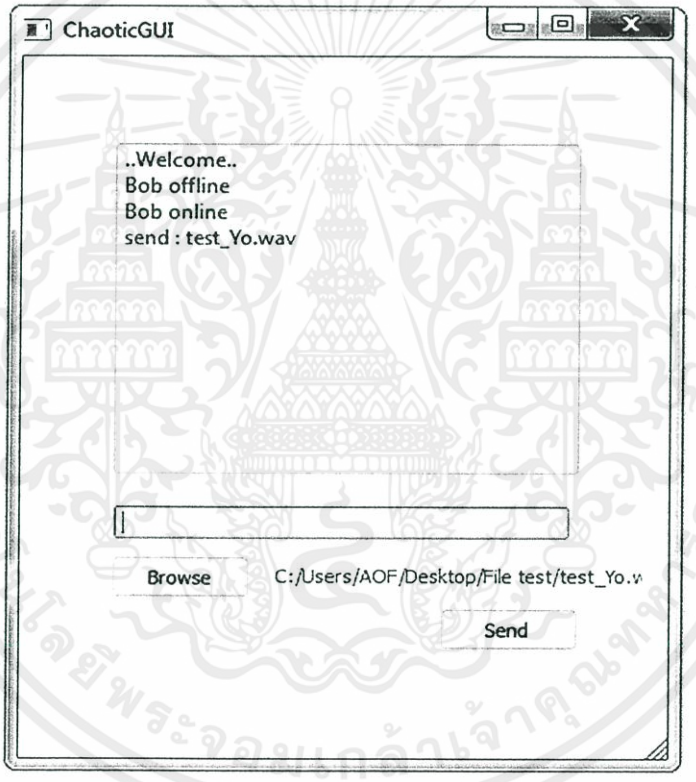


รูปที่ 4.72 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับไฟล์เสียงจากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ตรงกัน)

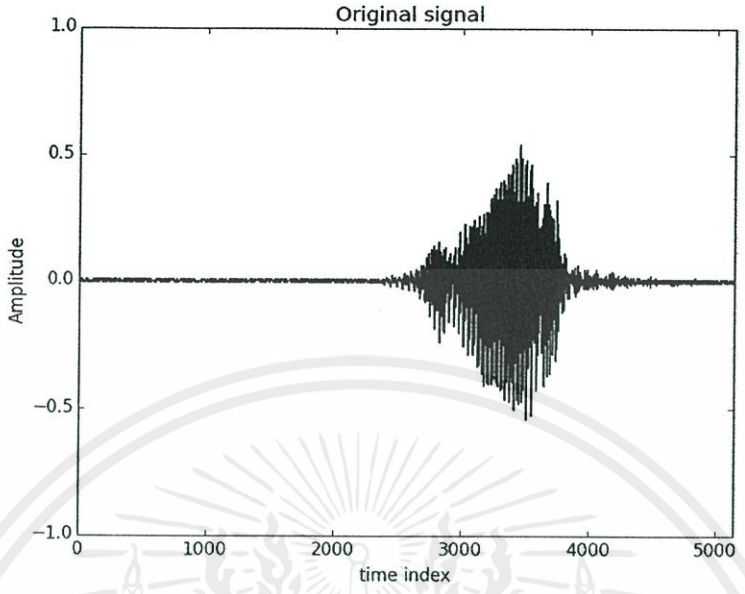


รูปที่ 4.73 กราฟของสัญญาณเสียงที่คอมพิเตอร์ Bob ได้รับ (ค่าสัมประสิทธิ์ตรงกัน)

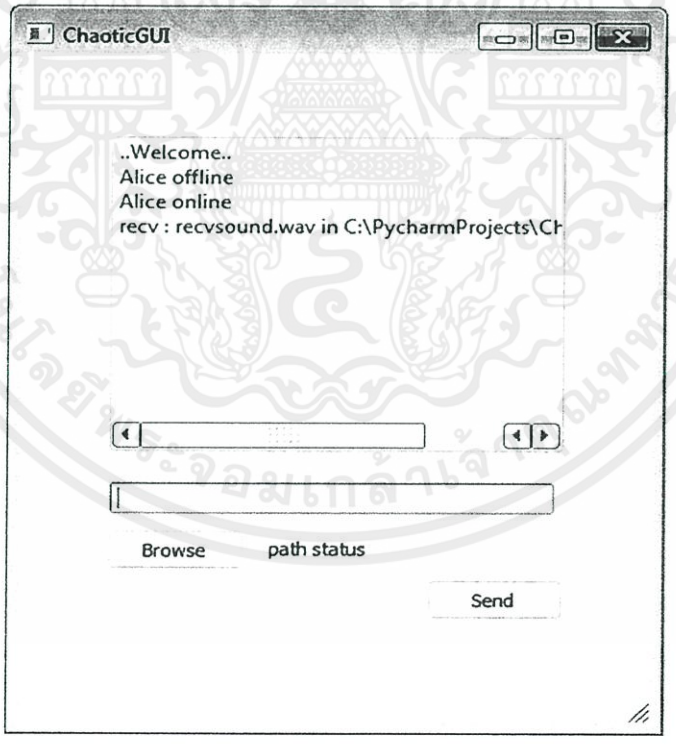
ในกรณีใช้สัมประสิทธิของวงจรกรองสัญญาณเชิงเลขฝั่งเข้ารหัสลับและฝั่งถอดรหัสลับ ไม่ตรงกัน เริ่มการส่งข้อมูลโดยคอมพิวเตอร์ Alice ส่งข้อมูลเสียง test_Yo.wav แสดงดังรูปที่ 4.74 และกราฟระหว่างแกนเวลา (แกน x) และแอมปลิจูด (แกน y) ของข้อมูลเสียงที่ส่งแสดงดังรูปที่ 4.75 ข้อมูลถูกนำไปเข้ารหัสลับแบบเคออติกที่อุปกรณ์ Raspberry Pi 1 และถอดรหัสลับที่อุปกรณ์ Raspberry Pi 2 แล้วส่งไปยังคอมพิวเตอร์ Bob คอมพิวเตอร์ Bob ได้รับข้อมูลเสียงที่ไม่เหมือนกับฝั่งส่ง แล้วแสดงคำว่า “recv : recvsound.wav in C:\Pycharm Projects \Chaos_ GUI\dist” บนหน้าต่าง GUI แสดงดังรูปที่ 4.76 และกราฟระหว่างแกนเวลา (แกน x) และแอมปลิจูด (แกน y) ของข้อมูลเสียงที่ได้รับ แสดงดังรูปที่ 4.77



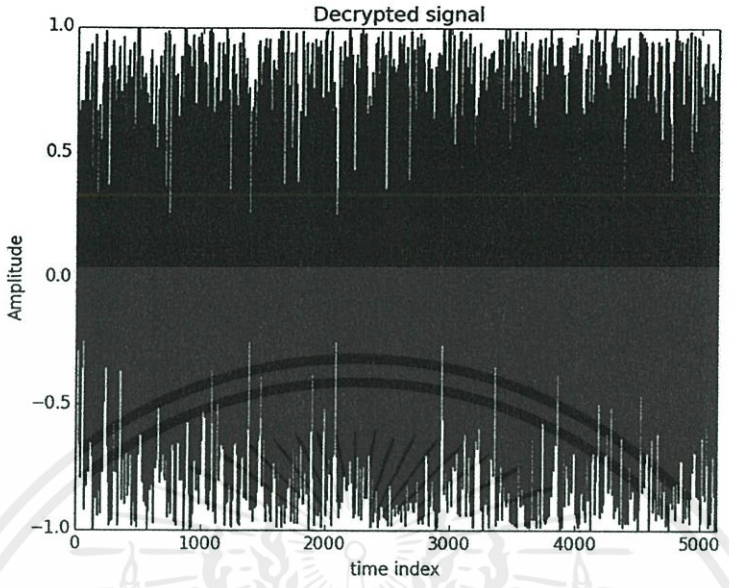
รูปที่ 4.74 หน้าต่าง GUI ของคอมพิวเตอร์ Alice เมื่อส่งไฟล์เสียงไปยังคอมพิวเตอร์ Bob (ค่าสัมประสิทธิ์ไม่ตรงกัน)



รูปที่ 4.75 กราฟของสัญญาณเสียงที่คอมพิเตอร์ Alice ส่ง (ค่าสัมประสิทธิ์ไม่ตรงกัน)



รูปที่ 4.76 หน้าต่าง GUI ของคอมพิวเตอร์Bob เมื่อรับไฟล์เสียง
จากคอมพิวเตอร์ Alice (ค่าสัมประสิทธิ์ไม่ตรงกัน)



รูปที่ 4.77 กราฟของสัญญาณเสียงที่คอมพิวเตอร์ Bob ได้รับ (ค่าสัมประสิทธิ์ไม่ตรงกัน)

4.6 ผลการโจมตีข้อมูลแบบตะลุย (Brute-Force)

ผลการทดลองการจำลองการโจมตีแบบ Brute-Force เมื่อใช้ขนาดกุญแจจำนวนบิตต่าง ๆ กันซึ่งกุญแจทั้งหมดที่เป็นได้จะเท่ากับ 2^n แสดงดังตารางที่ 4.1

ตาราง 4.1 จำนวนกุญแจทั้งหมดที่เป็นไปได้ทั้งหมดและเวลาที่ใช้ในการถอดรหัสลับ

ขนาดกุญแจ (บิต)	จำนวนกุญแจทั้งหมด	เวลาที่ใช้ในการถอดรหัสลับ (1 ล้านครั้งต่อวินาที)	เวลาที่ใช้ในการถอดรหัสลับ (1 ล้านล้านครั้งต่อวินาที)
32	$2^{32} = 4.3\times10^{9}$	35.8 นาที	1.25×10^{-3} วินาที
56	$2^{56} = 7.2\times10^{16}$	1142 ปี	10.01 ชั่วโมง
112	$2^{112} = 5.2\times10^{33}$	8.2×10^{19} ปี	8.2×10^{13} ปี
128	$2^{128} = 3.4\times10^{38}$	5.4×10^{24} ปี	5.4×10^{18} ปี
168	$2^{168} = 3.7\times10^{50}$	5.9×10^{36} ปี	5.9×10^{30} ปี
256	$2^{256} = 1.1\times10^{77}$	1.8×10^{63} ปี	1.8×10^{57} ปี

การเข้ารหัสลับแบบเคออติกจะมีการกำหนดค่าสัมประสิทธิ์ 2 ตัว (สำหรับวงจรกรองสัญญาณเชิงเลขอันดับสอง) ซึ่งการเพิ่มจำนวนบิตของค่าสัมประสิทธิ์ที่ใช้เป็นกุญแจเพิ่มมากขึ้นก็จะทำให้จำนวนกุญแจที่เป็นไปได้ทั้งหมดในการถอดรหัสลับนั้นเพิ่มมากขึ้น ความปลอดภัยก็จะเพิ่มมากขึ้นเวลาที่ใช้ในการถอดรหัสลับก็เพิ่มมากขึ้นแสดงดังตารางที่ 4.2

ตารางที่ 4.2 ความเป็นไปได้ทั้งหมดเมื่อเพิ่มจำนวนบิตของค่าสัมประสิทธิ์

ขนาดกุญแจ (บิต)	จำนวนกุญแจที่เป็นไปได้ (ต่อค่า c หนึ่งตัว)	จำนวนกุญแจที่เป็นไปได้ (ต่อค่า c สองตัว)	จำนวนครั้งในการคาดเดากุญแจ (2^{nd} order)	เวลาที่ใช้ในการถอดรหัสลับ (1 ล้านล้านครั้งต่อวินาที)
8	256	512	65536	1.03×10^{-15} ปี
16	65536	131072	4.29×10^{9}	6.80×10^{-11} ปี
24	16777216	33.55×10^{6}	2.81×10^{14}	4.45×10^{-6} ปี
32	4.29×10^{9}	8.58×10^{9}	1.84×10^{19}	0.29 ปี
64	1.84×10^{19}	3.68×10^{19}	3.4×10^{38}	5.39×10^{18} ปี
128	3.4×10^{38}	6.8×10^{38}	1.15×10^{77}	1.82×10^{57} ปี
256	1.15×10^{77}	2.31×10^{77}	1.32×10^{154}	2.09×10^{134} ปี

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ในปริญญานิพนธ์นี้ได้ใช้วงจรกรองสัญญาณเชิงเลขอันดับที่สองในการสร้างวงจรเข้ารหัสลับและถอดรหัสลับแบบเคออติก สำหรับวงจรเข้ารหัสลับใช้วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์ไม่จำกัด (IIR filter) โดยถ้าเลือกค่าสัมประสิทธิ์ของวงจรกรองอย่างน้อยหนึ่งตัวให้อยู่ภายนอกขอบเขตพื้นที่สามเหลี่ยมเสถียรภาพ จะทำให้ตำแหน่งของโพลอยู่ภายนอกวงกลมหนึ่งหน่วย ซึ่งทำให้วงจรเข้ารหัสลับเกิดความไม่มีเสถียรภาพ และในการจำลองการทำงานใช้ฟังก์ชันมอดูโลทำให้เกิดการล้นที่ทำให้เกิดความไม่เป็นเชิงเส้น ซึ่งทำให้เกิดปรากฏการณ์เคออสขึ้น สำหรับวงจรถอดรหัสลับใช้วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด (FIR filter) ซึ่งเป็นส่วนกลับ (Inverse) ของวงจรเข้ารหัสลับ โดยเมื่อเลือกค่าสัมประสิทธิ์ให้มีค่าตรงกับวงจรเข้ารหัสลับ จะทำให้การถอดรหัสลับได้ข้อมูลดั้งเดิมกลับคืนมา

ในการจำลองการทำงานของวงจรเข้ารหัสลับโดย เมื่อป้อนอินพุตเป็นศูนย์แล้วเอาต์พุตที่ได้ไม่เป็นศูนย์ (Zero in non zero out) และมีลักษณะคล้ายสัญญาณรบกวนซึ่งเป็นคุณสมบัติของระบบที่ไม่เป็นเชิงเส้น และเมื่อนำวงจรเข้ารหัสลับและถอดรหัสลับไปประยุกต์ใช้งานกับการเข้ารหัสข้อมูลชนิดต่าง ๆ ได้แก่ สัญญาณคลื่นรูปไซน์ ข้อมูลภาพ และข้อมูลเสียง พบว่าสามารถใช้กับการเข้ารหัสลับข้อมูลเหล่านี้ได้

เพื่อแสดงพฤติกรรมของวงจรกรองสัญญาณเชิงเลขที่เกิดปรากฏการณ์เคออสให้ชัดเจน จึงสร้างระบบรหัสลับแบบเคออติกที่ใช้กับการสื่อสารข้อมูลอนุกรมลงบนอุปกรณ์ Raspberry Pi โดยใช้ภาษาไพธอนอธิบายการทำงานทั้งหมด แล้วทำการส่งข้อมูลที่เป็นตัวอักษรซ้ำกันหลาย ๆ ตัวผ่านพอร์ตอนุกรมจากคอมพิวเตอร์ฝั่งส่งไปยังคอมพิวเตอร์ฝั่งรับ ข้อมูลที่ผ่านการเข้ารหัสลับแต่ละตัวจะไม่ซ้ำกัน (มีความเป็นพลวัตร) ซึ่งเป็นคุณสมบัติหนึ่งของปรากฏการณ์เคออส

เพื่อนำไปใช้เป็นชุดสาธิตการทำงานด้านความปลอดภัยบนเครือข่าย จึงสร้างระบบรหัสลับแบบเคออติกลงบนอุปกรณ์ Raspberry Pi ที่ใช้กับการสื่อสารข้อมูลผ่านเครือข่าย โดยใช้ภาษาไพธอนอธิบายการทำงานทั้งหมด ซึ่งชุดสาธิตการทำงานนี้สามารถใช้สื่อสารแบบ Full duplex ระหว่างคอมพิวเตอร์ 2 เครื่อง ข้อมูลที่ผู้ใช้งานส่งออกไปบนเครือข่ายจะผ่านการเข้ารหัสลับแบบเคออติกก่อน และที่ฝั่งรับ ข้อมูลก็จะถูกถอดรหัสลับก่อนส่งมายังผู้ใช้งาน จากนั้นให้คอมพิวเตอร์อีกเครื่องหนึ่งทำหน้าที่ดักจับข้อมูลบนเครือข่าย พบว่าข้อมูลที่ดักจับได้เป็นข้อมูลที่ผ่านการเข้ารหัสลับแบบเคออติกซึ่งไม่สามารถตีความได้

ต่อมาได้พัฒนาชุดสาธิตการทำงานให้สามารถสื่อสารข้อมูลชนิดภาพ และเสียงได้ด้วย พร้อมทั้งสร้างหน้าต่างโปรแกรม GUI (Graphic User Interface) บนคอมพิวเตอร์ โดยเชื่อมโยงไว้กับภาษาไพธอนที่อธิบายการรับส่งข้อมูลชนิดข้อความ ภาพ และเสียง

5.2 ข้อจำกัดของชุดสาธิตการทำงาน

1. เมื่อใช้ค่าสัมประสิทธิ์ของวงจรถอดรหัสลับไม่ตรงกับวงจรเข้ารหัสลับ ถ้าข้อมูลที่ผ่านการถอดรหัสลับแล้วส่งมายังคอมพิวเตอร์มี \n (สัญลักษณ์การขึ้นบรรทัดใหม่) อยู่ด้วยนั้น หน้าต่างโปรแกรม GUI จะไม่สามารถแสดงข้อมูลที่มีการขึ้นบรรทัดใหม่ได้

2. เมื่อมีผู้ใช้งานฝั่งหนึ่งปิดหน้าต่างโปรแกรม Socket จะถูกปิดลง และส่งข้อความไปให้ผู้ใช้งานอีกฝั่งทราบ ผู้ใช้งานอีกฝั่งจะต้องทำการปิดหน้าต่างโปรแกรม เพื่อให้ Socket ถูกปิดลงเช่นกัน จากนั้นผู้ใช้งานทั้งสองจึงเริ่มเปิดหน้าต่างโปรแกรมใหม่อีกครั้ง Socket ก็จะถูกสร้างขึ้นใหม่

3. เมื่อมีผู้ใช้งานฝั่งหนึ่งทำการเปิด - ปิดหน้าต่างโปรแกรมซ้ำ ๆ โดยที่ไม่รอให้ผู้ใช้งานอีกฝั่งปิดหน้าต่างโปรแกรมก่อน ผู้ใช้งานจะไม่สามารถรับข้อมูลได้ หากต้องการให้สามารถกลับมาสื่อสารกันได้อีกครั้ง ผู้ใช้งานอีกฝั่งจะต้องเปิด - ปิดหน้าต่างโปรแกรมด้วยจำนวนครั้งที่เท่ากัน

4. เมื่อมีผู้ใช้งานฝั่งหนึ่งส่งข้อมูลออกไป แต่ผู้ใช้งานอีกฝั่งขาดการเชื่อมต่อกับเครือข่าย ผู้ใช้งานอีกฝั่งจะยังไม่สามารถรับข้อมูลได้จนกว่าจะมีการเชื่อมต่อมายังเครือข่ายอีกครั้ง

5.3 ข้อเสนอแนะ

1. การเพิ่มควาวยาวบิตของค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขทำให้สัญญาณที่ผ่านการเข้ารหัสแบบเคออติกมีลักษณะคล้ายสัญญาณรบกวนมากขึ้น (มีความใกล้เคียง Real Chaos มากขึ้น) อีกทั้งยังทำให้การโจมตีแบบตลุย Brute force ทำได้ยากขึ้น (จำนวนครั้งในการคาดเดากุญแจเพิ่มขึ้น)

2. เมื่อทำการเพิ่มความยาวบิตของค่าสัมประสิทธิ์ของวงจรกรองสัญญาณแล้ว จึงจำเป็นต้องเพิ่มความยาวบิตของข้อมูลอินพุตขึ้นให้เท่ากับความยาวบิตของค่าสัมประสิทธิ์ด้วย ซึ่งทำให้สามารถเข้ารหัสได้เป็นชุด ชุดละหลายตัวอักษร และหากข้อมูลอินพุตมีความยาวไม่ครบชุด ก็จะเพิ่ม space ให้มีความยาวเต็ม 1 ชุด ก่อนนำไปเข้ารหัส ทั้งหมดนี้ส่งผลให้ความยาวของข้อมูลอินพุต และข้อมูลที่ผ่านการเข้ารหัสไม่จำเป็นต้องมีความยาวบิตเท่ากัน

3. เพื่อให้สามารถนำระบบรหัสลับแบบเคออติกบนอุปกรณ์ Raspberry Pi ไปใช้จริงบนเครือข่ายได้ ควรทำให้ระบบมีการใช้งานที่สะดวก และเข้าใจง่ายมากขึ้น และควรมีการตรวจสอบเมื่อผู้ใช้ฝั่งใดขาดการเชื่อมต่อ

บรรณานุกรม

[1] สมเกียรติ ตั้งกิจวานิชย์. "ทฤษฎีความโกลาหล." http://th.wikipedia.org/wiki/ทฤษฎีความอลวน.

[2] สิทธิพงษ์ คิวหา, สุนทรินทร์ ศิลป์ท้าว, สุนัย เนสะและ. "อุปกรณ์เข้ารหัส - ถอดรหัสแบบ CHAOTIC บนพื้นฐานความไม่เป็นเชิงเส้นจากการล้นในวงจรกรองสัญญาณดิจิตอล สำหรับความปลอดภัยในการสื่อสาร." ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต, สาขาวิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2553

[3] Frey, D.R.. "Chaotic Digital Encoding: An Approach to Secure Communication." IEEE Trans. Circ & Syst. II, vol. 40, No. 10 (1993) : 660-666.

[4] Kutzer, Kristina ; Schwarz, Wolfgang ; Davies, Anthony C.. "Chaotic signals generated by Digital Filter Overflow,." Proc. ISCAS'94, London, Vol 6.(1994) : 17 - 20

[5] "การส่งผ่านข้อมูลแบบอนุกรม." http://irrigation.rid.go.th/rid15/ppn/Knowledge/Networks%20Technology/network4.htm

[6] สุวัฒน์ ปุณณชัยยะ, ตัน ตัณฑ์สุทธิวงศ์, สุพจน์ ปุณณชัยยะ. *เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต*. พิมพ์ครั้งที่ 2. กรุงเทพ : Provision, 2545.

[7] ชัยวัฒน์ ลิ้มพรจิตรวิไล, สมเกียรติ กิจวงศ์วัฒนะ. "Raspberry Pi." *The Prototype Electronics*. ฉบับที่ 32. (กุมภาพันธ์ 2556) : 42-56

[8] จักรกฤษณ์ แสงแก้ว. *การเขียนโปรแกรมภาษาไพธอนด้วยตนเอง*. กรุงเทพ : สมาคมส่งเสริมเทคโนโลยี (ไทย- ญี่ปุ่น). 2549.

[9] Y.Daniel Liang. *Introduction to Programming Python*. Pearson Education, Inc

[10] "pySerial 2.7 documentation." http://pyserial.sourceforge.net/pyserial_api.html

[11] Silver Moon. "Python socket - network programming tutorial." http://www.binarytides.com/python-socket-network-programming-tutorial/

[12] ลัญฉกร วุฒิสิทธิกุลกิจ, ธงชัย โรจน์กังสดาล, วรากร ศรีเชวงทรัพย์, นพดล พรมภักษร, สุวิทย์ นาคพีระยุทธ. *วิทยาการรหัสลับเบื้องต้น Introduction to Cryptography*. กรุงเทพ : จุฬาลงกรณ์มหาวิทยาลัย, 2548.