โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย

Secure Instant Messaging Server

นายสเกน สำราญบำรุง
นายอดิศักดิ์ วงศ์จันลา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย

Secure Instant Messaging Server

โดย

นายสเกน สำราญบำรุง

นายอดิศักดิ์ วงศ์จันลา

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย

Secure Instant Messaging Server

ผู้จัดทำ

1. นายสเกน สำราญบำรุง รหัสนักศึกษา 44015351
2. นายอดิศักดิ์ วงศ์จันลา รหัสนักศึกษา 44015363

..... อาจารย์ที่ปรึกษา

(อาจารย์ธนา หงษ์สุวรรณ)

..... อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย

นายสเกน สำราญบำรุง 44015351
นายอดิศักดิ์ วงศ์จันลา 44015363
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย (Secure Instant Messaging Server) หรือเรียกสั้นๆ ว่า IsagMQ เป็นโปรแกรมฝั่งผู้ให้บริการรับส่งสารด่วน (Instant Messaging) ที่เน้นความปลอดภัยในการให้บริการ โดยสามารถพิสูจน์ได้ว่าข้อมูลที่ใช้ติดต่อสื่อสารระหว่างผู้ใช้บริการด้วยกันเองนั้นเป็นของผู้ใช้บริการคนนั้นจริงๆ ไม่มีการแอบอ้าง (Certificate Authority) และสามารถบ่งชี้ได้ว่าผู้ใช้บริการตัวจริงเท่านั้นถึงจะขอใช้บริการจาก IsagMQ ได้

IsagMQ สามารถให้บริการรับส่งสารด่วนพื้นฐานได้โดยทั่วไปเช่น ล็อกอิน (Login), ล็อกเอาต์ (Logout), เพิ่มคู่สนทนา (Add contact), ลบคู่สนทนา (Remove contact), ลงทะเบียน (Register), ค้นหาคู่สนทนา (Find contact), เปลี่ยนชื่อเล่น (Change Nickname) และ บริการแจ้งสถานะของคู่สนทนา (Status) บริการทุกอย่างจะเน้นที่ความปลอดภัยโดยใช้หลักการของ SSL โพรโตคอล มาออกแบบโพรโตคอลที่ใช้สื่อสารระหว่าง IsagMQ และ ผู้ใช้บริการ การสร้างความปลอดภัยจะใช้การเข้ารหัสสองแบบ คือ การเข้ารหัสแบบกุญแจเดี่ยว (Secret key cryptography) และ การเข้ารหัสแบบกุญแจคู่ (Public key cryptography) โดยการเข้ารหัสแบบกุญแจคู่จะใช้เข้ารหัสและถอดรหัสกุญแจเดี่ยว ส่วนกุญแจเดี่ยวจะใช้ในการเข้ารหัสและถอดรหัสข้อมูลที่ใช้ในการสื่อสาร การใช้การเข้ารหัสทั้งสองร่วมกันเพื่อเสริมจุดดีและแก้ไขจุดด้อยของแต่ละแบบ ซึ่งจะทำให้ได้ประสิทธิภาพการเข้ารหัสที่ดีขึ้นกว่าการใช้แบบใดแบบหนึ่ง

รูปแบบในการสื่อสารจะเป็นแบบเพียร์ทูเพียร์ (Peer-to-Peer Message) เพราะจะทำให้เซิร์ฟเวอร์ไม่ต้องทำงานหนักและการสื่อสารระหว่างผู้ใช้บริการจะมีความรวดเร็วกว่าการให้เซิร์ฟเวอร์เป็นตัวกลาง

Secure Instant Message Server

Mr. Sgan Samranbamroong
Mr. Adisak Wongjanla
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

ABSTRACT

Secure Instant Messaging Server (IsagMQ) is a server application program that has been strictly designed to base mainly on the securely instant messaging services. So IsagMQ have some functions for providing identity, keeping secrets and verification information.

SIMS can support general instant messaging services such as Login, Logout, Add contact, Remove contact, Register, Find contract, Change nickname but all of services must be securely. So SIMS protocol would be designed based on SSL protocol using secret key and public key cryptography. A public key has been used for encryption and decryption a secret key and secret key has been used for encryption plaintext to chiphertext and decryption chipertext to plaintext. The reason why using two keys for encryption and decryption is for increasing the throughput between client and server but still remaining an efficient security.

The connection between client and server used peer-to-peer message connection because the server doesn't have to work heavily and communication between one client and another client has a good speed of response.

กิตติกรรมประกาศ

โครงงานนี้คงไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากบุคคลหลายๆ ฝ่ายด้วยกัน บุคคลสองท่านแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้ปริญญานิพนธ์นี้สำเร็จลงได้ก็คือ อาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ก่อให้เกิดโครงงานนี้ พร้อมทั้งให้คำแนะนำและให้คำปรึกษา รวมถึงช่วยแก้ไขปัญหาต่างๆ อย่างต่อเนื่องตลอดโครงงาน ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบคุณห้องปฏิบัติการ ISAG ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ ที่ได้สนับสนุนสถานที่และอุปกรณ์เครือข่ายสำหรับใช้ในการพัฒนาโครงงาน

ขอขอบคุณคณาจารย์ในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาความรู้ให้กับผู้จัดทำ

ขอขอบคุณเพื่อนห้อง D และห้อง P ที่คอยช่วยเหลือ ให้กำลังใจและสร้างบรรยากาศที่ดีตลอดมา

สุดท้ายนี้ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ผู้จัดทำมีวันนี้ก็คือ บิดา-มารดา อันเป็นที่เคารพรัก ซึ่งได้ให้กำเนิดและเลี้ยงดูมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ คอยให้กำลังใจและเอาใจใส่เสมอมาอันหาที่เปรียบมิได้ ผู้จัดทำขอระลึกในพระคุณและขอกราบขอพระคุณมา ณ ที่นี้

สเกน สำราญบำรุง

อดิศักดิ์ วงศ์จันลา

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

การใช้โปรแกรมสำหรับรับส่งสารด่วน (Instant messaging) ในปัจจุบันเป็นที่นิยมกันมากขึ้น โดยมีผู้ให้บริการคือโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วน (Instant messaging Server) อยู่มากเช่นกัน แต่การให้บริการเหล่านั้นยังมีจุดอ่อนอยู่มากดังนี้ ประการแรกไม่มีการเข้ารหัสและถอดรหัสระหว่างการสื่อสารของผู้ใช้บริการ (Client) และผู้ให้บริการ (Server) ทำให้อาจถูกดักจับข้อมูลในการสื่อสารนั้นได้ และมันยิ่งเลวร้ายถ้าการติดต่อระหว่างผู้ใช้บริการกับผู้ให้บริการเป็นแบบเพียร์ทูเพียร์ (peer to peer) ซึ่งผู้ให้บริการจะต้องส่งหมายเลขไอพี (IP) ของคู่สนทนาให้แก่ผู้ใช้บริการที่ต้องการทำการติดต่อ ถ้าถูกดักจับก็จะส่งผลเลวร้ายขึ้นมาได้ ผู้ใช้บริการไม่อาจสามารถมั่นใจได้ว่าคู่สนทนานั้นเป็นตัวจริงโดยไม่มีการแอบอ้าง ประการที่สองเมื่อใช้อินเทอร์เน็ต (Internet) ไม่ได้ การใช้บริการโปรแกรมสำหรับรับส่งสารด่วนจากภายนอกองค์กรไม่อาจทำได้และในปัจจุบันภายในประเทศเองก็ยังไม่มีผู้ให้บริการโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนจะต้องต่ออินเทอร์เน็ตออกนอกประเทศทุกครั้ง ประการที่สามโดยผู้ให้บริการโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนส่วนใหญ่จะเน้นการบริการด้านความบันเทิงมากกว่าการเน้นเรื่องความปลอดภัย

ด้วยเหตุนี้ ผู้พัฒนาโครงการจึงได้เกิดแนวคิดที่จะพัฒนาระบบการให้บริการโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนให้มีความปลอดภัยมากขึ้น โดยเน้นไปที่การสื่อสารระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์จะต้องอยู่ในช่องทางที่ปลอดภัย

โครงการนี้มีชื่อว่าโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย (Secure Instant message Server) เพื่อให้ง่ายต่อการเข้าใจและการอ้างอิง จะใช้ชื่อย่อๆว่า IsagMQ (Isag เป็นชื่อห้องโปรเจ็คที่ใช้พัฒนาโครงการนี้ M มาจากคำว่า Messaging และ Q มาจากคำว่า Queue)

และขอทำความเข้าใจเกี่ยวกับความหมายของคำในโครงงานนี้คือถ้าหากพูดถึงระบบรับส่งสารด่วนจะหมายถึงระบบของไคลเอ็นและเซิร์ฟเวอร์สำหรับรับส่งสารด่วน และถ้าพูดถึงโปรแกรมรับส่งสารด่วนจะหมายถึงโปรแกรมฝั่งไคลเอ็นเท่านั้น

1.2 วัตถุประสงค์

1. เพื่อสร้างโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย(Secure Instant Messaging Server) ที่มีความปลอดภัยในการรับส่งข้อมูลระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์
2. เพื่อสร้างโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัยที่สามารถนำไปใช้ในองค์กรใดๆ ได้

3. เพื่อให้มีการรับประกันว่าผู้ที่จะสามารถใช้บริการของ IsagMQ ได้นั้นจะต้องมีใบรับรองสิทธิ์ (Certificate) การใช้อย่างถูกต้องแล้วเท่านั้น
4. เพื่อให้ผู้ใช้บริการแน่ใจว่าสามารถส่งข้อมูลที่เป็นความลับต่อคู่สนทนาได้อย่างปลอดภัย
5. เพื่อศึกษาและพัฒนาการเขียนโปรแกรมทางด้านเครือข่ายและความปลอดภัยลินุกซ์ (Linux)

1.3 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ความสามารถในการเข้ารหัสและถอดรหัสข้อมูล
2. ได้รับความรู้ความสามารถในการทำระบบพิสูจน์ตน
3. ได้รับความรู้ความสามารถในการรับ-ส่งข้อมูลผ่านเครือข่ายทีซีพีไอพี (TCP/IP)
4. ได้รับความรู้ความสามารถในการทำงานบนระบบลินุกซ์
5. ได้รับความรู้ความสามารถด้านการเขียนโปรแกรม

1.4 ขอบเขตของการพัฒนา

Input specification

- รับการร้องขอ (Request) การขอใช้บริการจากไคลเอ็นต์ผ่านทางเครือข่าย

Output specification

- ทำการตอบสนอง (Response) การขอใช้บริการจากไคลเอ็นต์ผ่านทางเครือข่าย
- แสดงผลการตอบสนองออกทางจอภาพโดยเป็นเท็กซ์โหมด (Text mode)

Function specification

- สามารถให้บริการลงทะเบียนได้
- สามารถให้บริการยกเลิกการลงทะเบียนได้
- สามารถให้บริการเพิ่มคอนแท็ก (Contact) ได้
- สามารถให้บริการลบคอนแท็กได้
- สามารถให้บริการค้นหาคอนแท็กได้
- สามารถให้บริการเปลี่ยนชื่อเล่นของแต่ละสมาชิกได้
- สามารถให้บริการล็อกอินและล็อกเอาต์ได้
- สามารถส่งคอนแท็กลิสต์ (Contact List) ให้แก่ไคลเอ็นต์ได้
- สามารถบอกสถานะออนไลน์และออฟไลน์ของแต่ละคอนแท็กได้
- สามารถสร้างการสื่อสารระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ให้อยู่ในช่องทางที่ปลอดภัยด้วยการเข้ารหัสและถอดรหัสข้อมูลได้

1.5 วิธีการดำเนินงาน

งานวิจัยในโครงการนี้จะเริ่มด้วยการศึกษาทฤษฎีและแนวคิดต่างๆ ที่เกี่ยวข้องกับงานวิจัย โดยเริ่มจากการศึกษาฟังก์ชันการทำงานของระบบรับส่งสารด่วนเช่น ICQ, MSN Messenger, Yahoo Messenger และ IsagQ 2545 (โปรแกรมรับส่งสารด่วน ปีการศึกษา 2545) เพื่อหาข้อดีและข้อเสียของระบบรับส่งสารด่วนข้างต้น และคัดเลือกฟังก์ชันการทำงานที่จำเป็นมาใช้ในโครงงานนี้ ศึกษาการเขียนโปรแกรมทางด้านเครือข่ายบนยูนิกซ์ ศึกษาระบบการเข้ารหัสข้อมูล จากนั้นจึงทำการศึกษาโพรโตคอล SSL & TLS เพื่อนำมาออกแบบโพรโตคอลของ IsagMQ เพื่อให้มีช่องทางการสื่อสารที่ปลอดภัย และศึกษาการทำระบบรับรองสิทธิ์ (Certificate Authority: CA) ระหว่างไคล์เอ็นและเซิร์ฟเวอร์ เพื่อที่จะสร้างความมั่นใจแก่ผู้ใช้บริการว่าผู้มีสิทธิ์เท่านั้นจึงจะได้คอนแท็คท์ลิซท์ไป จากนั้นจึงได้นำความรู้ที่ได้ศึกษามาทั้งหมดทำการออกแบบและเขียนโปรแกรม IsagMQ โดยเริ่มจากการเขียนโปรแกรมสำหรับให้บริการต่างๆ กับไคลเอ็นต์เช่น ให้บริการลงทะเบียน ให้บริการเพิ่มคอนแท็กลิสต์ ให้บริการค้นหาคอนแท็คท์ลิซท์และให้บริการบอกสถานะออนไลน์และออฟไลน์ของแต่ละคอนแท็กลิสต์เป็นต้น จากนั้นจึงทำการเขียนโปรแกรม IsagMQให้มีช่องทางการสื่อสารที่ปลอดภัยคือ ช่องทางนั้นต้องมีการเข้ารหัสถอดรหัสข้อมูลและมีระบบการรับรองสิทธิ์ผู้ใช้ ระหว่างไคลเอ็นและเซิร์ฟเวอร์ ซึ่งจะใช้ SSL & TLS โพรโตคอลและใช้ไลบรารีของ OpenSSL ในการติดต่อและใช้งาน SSL&TLS โพรโตคอล หลังจากนั้นจึงทำการทดสอบการทำงานทั้งหมด ตรวจสอบข้อบกพร่องของระบบ เพื่อนำกลับมาแก้ไขให้ดีขึ้น

บทที่ 2
โปรแกรมแม่ข่ายและลูกข่ายสำหรับรับส่งสารด่วน

ใน ระบบรับส่งสารด่วน จะประกอบด้วยส่วนสำคัญ 2 ส่วน ได้แก่ โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วน (Instant Messaging Server) และโปรแกรมรับส่งสารด่วนฝั่งไคลเอ็นต์ (Instant Messaging Client)

โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วน เป็นเซิร์ฟเวอร์ที่ให้รับส่งสารด่วนแก่ ไคลเอ็นต์โดยก่อนที่ไคลเอ็นต์จะทำการติดต่อสื่อสารกันได้จะต้องทำการล็อกอินไปที่เซิร์ฟเวอร์ก่อนเพื่อให้เซิร์ฟเวอร์ส่งคอนแท็คท์ลิซท์ให้แก่ผู้ล็อกอิน จากนั้นไคลเอ็นต์จะใช้คอนแท็คท์ลิซท์ในการสื่อสารระหว่างไคลเอ็นต์ด้วยกันเอง ในปัจจุบันมีผู้ให้บริการระบบรับส่งสารด่วนที่รู้จักกันที่เช่น ICQ Server, MSN Messenger Server, Yahoo Messenger Server ผู้ให้บริการเหล่านี้ไม่ได้เน้นความปลอดภัยในการให้บริการจะเน้นด้านความบันเทิงมากกว่า

โปรแกรมรับส่งสารด่วน เป็นไคลเอ็นต์สำหรับรับส่งสารระหว่างผู้ใช้งานสองคนที่ทำการออนไลน์ในเวลาเดียวกัน โดยการรับส่งสารมีลักษณะการรับส่งแบบเพียร์ทูเพียร์ ซึ่งเป็นการรับส่งสารจากผู้ส่งไปยังผู้รับได้โดยตรงทำให้ผู้ใช้งานสามารถทำการสนทนาระหว่างกันได้อย่างรวดเร็วแต่ปลอดภัยน้อยเช่น ICQ หรือส่งสารผ่านเซิร์ฟเวอร์ก่อนแล้วค่อยให้เซิร์ฟเวอร์ส่งสารให้ไคลเอ็นต์อีกทีปลอดภัยมากกว่าแต่ช้าเช่น MSN Messenger โดยในปัจจุบันโปรแกรมรับส่งสารได้รับความนิยมใช้งานเป็นจำนวนมากและมีผู้ผลิตและพัฒนาโปรแกรมรับส่งสารด่วนมากขึ้น ทำให้โปรแกรมรับส่งสารด่วนมีความสามารถในการใช้งานและคุณสมบัติเสริมที่มากขึ้น สามารถตอบสนองความต้องการของผู้ใช้งานได้อย่างทั่วถึง แต่ฟังก์ชันด้านความปลอดภัยยังไม่ดีเท่าที่ควร

2.1 การใช้ระบบรับส่งสารด่วนในปัจจุบัน

ในปัจจุบันผู้ใช้โปรแกรมรับส่งสารด่วนเกือบทั้งหมดจะใช้เพื่อจุดประสงค์ด้านความบันเทิงเท่านั้น แต่ตามความสามารถของโปรแกรมรับส่งสารด่วนเองก็สามารถที่จะนำมาประยุกต์ใช้งานกับองค์กรทั่วไปได้ถ้าหากมีการปรับปรุงด้านความปลอดภัยของระบบเพิ่มเติม

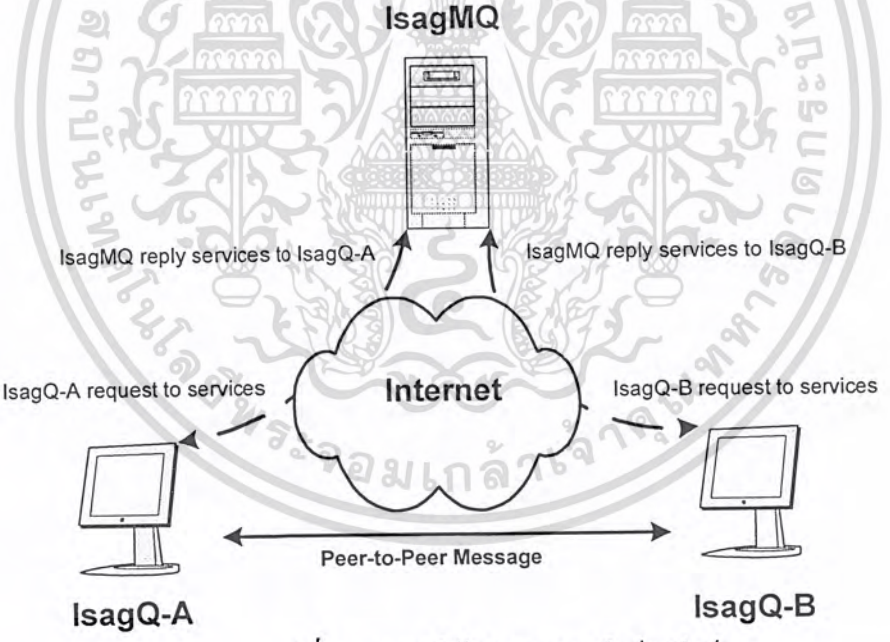
โดยสามารถนำโปรแกรมรับส่งสารด่วนมาใช้ส่งข้อความที่สำคัญและเร่งด่วนได้ซึ่งอาจจะดีกว่าการใช้อีเมล์เสียด้วยซ้ำเพราะรวดเร็วทันใจและไม่ต้องเปลืองเนื้อที่การเก็บข้อมูลของอีเมล์อีกด้วย ซึ่งถ้าหากระบบมีความปลอดภัยมากพอก็จะสามารถส่งข้อความที่เป็นความลับได้เหมือนการใช้อีเมล์

2.2 การทำงานของระบบรับส่งสารด่วน

ในระบบรับส่งสารด่วนจะประกอบด้วยส่วนสำคัญ 2 ส่วน ได้แก่ โปรแกรมแม่ข่ายสำหรับรับส่งสารด่วน (Instant Messaging Server) และโปรแกรมรับส่งสารด่วนฝั่งไคลเอ็นต์สำหรับผู้ใช้งานทั่วไป (Instant Messaging Client) โดยในปัจจุบันการให้บริการระบบรับส่งสารด่วนมี 2 รูปแบบ คือ

2.2.1 การทำงานแบบเพียร์ทูเพียร์ (Peer-to-Peer Message)

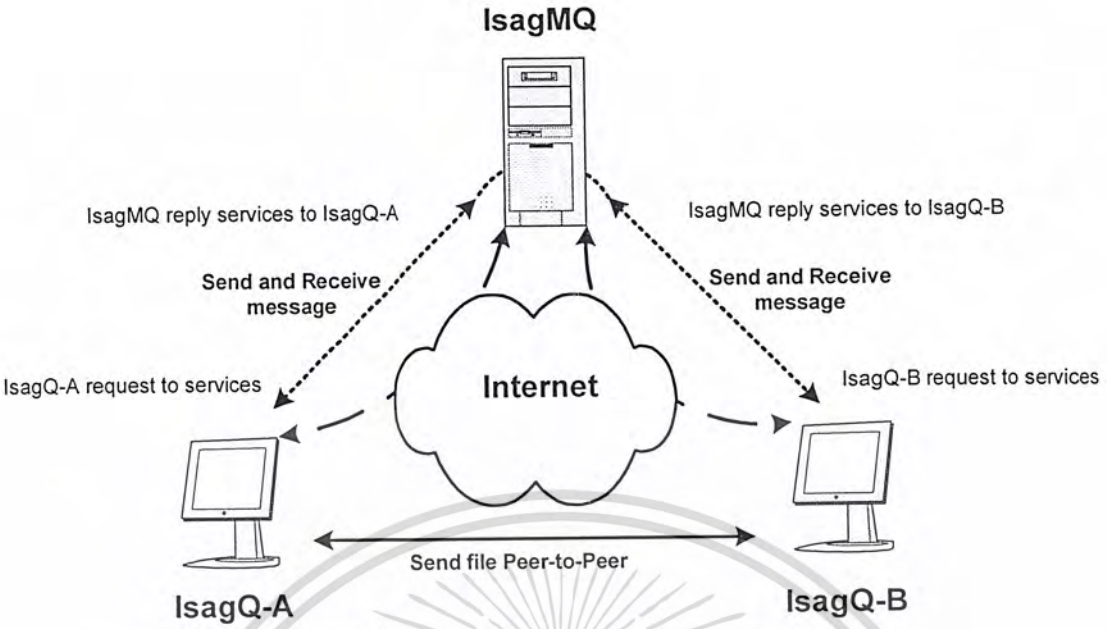
การทำงานจะเริ่มจากผู้ใช้ A ดังรูปที่ 2-1 ทำการล็อกอิน (ในที่นี่ถือว่าผู้ใช้ A และ B เป็นสมาชิกของ IsagMQ แล้ว) ไปยังเซิร์ฟเวอร์ของโปรแกรมรับส่งสารด่วนเพื่อรับส่งค่ารายละเอียดต่าง ๆ เช่น แจ้งหมายเลขไอพี และสถานะ การใช้งานให้กับเซิร์ฟเวอร์ สมมติว่าผู้ใช้ A และ B เป็น คอนแท็กลิสต์ ซึ่งกันและกัน จากนั้นเซิร์ฟเวอร์ก็จะแจ้งรายชื่อ และ หมายเลขไอพีของผู้ใช้ที่ทำการออนไลน์ซึ่งอยู่ในคอนแท็กลิสต์ซึ่งไม่ใช่ผู้ใช้ B เพราะยังไม่ออนไลน์ ให้กับผู้ใช้ A และในทำนองเดียวกัน เซิร์ฟเวอร์ก็จะแจ้งสถานะการออนไลน์ของผู้ใช้ Aให้กับคอนแท็กลิสต์ของผู้ใช้ A ที่ออนไลน์อยู่ และถ้าหากผู้ใช้ B ทำการล็อกอินไปยังเซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำงานดังข้างต้น หลังจากนั้นผู้ใช้ A และ B ก็จะทราบหมายเลขไอพีของกันและกันทำให้สามารถติดต่อสื่อสารกันแบบเพียร์ทูเพียร์ ได้ดังรูปที่ 2-1



รูปที่ 2-1 การทำให้บริการแบบเพียร์ทูเพียร์

2.2.2 การทำงานแบบข้อความไปเซิร์ฟเวอร์ (Message-to-Server)

การทำงานจะเริ่มต้นเหมือนแบบเพียร์ทูเพียร์ ต่างกันที่ในเซิร์ฟเวอร์จะไม่ส่งหมายเลขไอพีของคอนแท็กลิสต์ให้แก่ผู้ใช้ ผู้ใช้จะได้แค่ชื่อและสถานะการออนไลน์หรือออฟไลน์ของคอนแท็กลิสต์เท่านั้น การส่งข้อความระหว่างผู้ใช้จะต้องผ่านเซิร์ฟเวอร์ดังรูปที่ 2-2 ดังนั้นผู้ใช้จะทราบหมายเลขไอพีของเซิร์ฟเวอร์เท่านั้น



รูปที่ 2-2 การทำให้บริการแบบข้อความไปเซิร์ฟเวอร์

อย่างไรก็ตามโปรแกรมรับส่งสารด่วนจะสามารถทำการรับส่งสารโดยตรงระหว่างผู้ใช้งานได้ต่อเมื่อผู้ใช้งานทั้งสองฝั่งนั้นทำการออนไลน์เท่านั้น หากทางฝั่งผู้รับทำการออฟไลน์จะไม่สามารถทำการส่งสารโดยตรงระหว่างผู้ใช้งานได้ ซึ่งในส่วนนี้จะขึ้นอยู่กับแต่ละแอพพลิเคชันที่ผู้ผลิตได้ออกแบบมาเช่น ใน ICQ หากผู้รับทำการออฟไลน์ ผู้ส่งสามารถทำการส่งสารได้โดยข้อมูลที่ส่งให้ผู้รับจะถูกเก็บไว้ที่เซิร์ฟเวอร์ของ ICQ เมื่อผู้รับทำการออนไลน์ในภายหลังเซิร์ฟเวอร์ของ ICQ จะส่งสารของผู้ใช้งานคนนั้นที่ได้รับขณะออฟไลน์กลับมาให้ ส่วนใน MSN Maesenger และ Yahoo Messenger นั้นเมื่อผู้รับทำการออฟไลน์จะสามารถส่งสารได้โดยผ่านทางการใช้จดหมายอิเล็กทรอนิกส์แทน ทั้งสองแบบมีข้อดีและข้อเสียดังตารางที่ 2-1

การส่ง Message แบบ Peer-to-Peer	การส่ง Message แบบผ่าน Server
ข้อดี 1. ส่งข้อความได้รวดเร็ว 2. เซิร์ฟเวอร์ทำงานไม่หนัก 3. ผลจากข้อ 2. ทำให้ล็อกอินได้ง่าย	***ข้อดี*** 1. มีปลอดภัยเพราะไม่ต้องส่งหมายเลขไอพีให้แก่คู่สนทนา 2. การสื่อสารระหว่างไคลเอ็นจะไม่ติดไฟร์วอลล์ เพราะติดต่อกับเซิร์ฟเวอร์ต้องเปิดพอร์ตไว้อยู่แล้ว
ข้อเสีย 1.ไม่ปลอดภัยเพราะต้องส่งหมายเลขไอพีให้แก่คู่สนทนา 2. การสื่อสารระหว่างไคลเอ็นจะติดไฟร์วอลล์มีความยุ่งยากในการเลือกพอร์ต	***ข้อเสีย*** 1. เซิร์ฟเวอร์ทำงานหนัก 2. ผลจากข้อ 1. ทำให้ล็อกอินได้ยาก 3. ส่งข้อความได้ล่าช้า

ตารางที่ 2-1 เปรียบเทียบการทำงานของระบบรับส่งสารด่วน

2.3 ตัวอย่างของระบบรับส่งสารด่วน

ในปัจจุบันมีผู้ผลิตและพัฒนาโปรแกรมรับส่งสารด่วนจำนวนมาก โดยเริ่มจากบริษัทอเมริกันออนไลน์ (America Online Incorporation) ซึ่งเป็นผู้ผลิตโปรแกรมรับส่งสารด่วนรายใหญ่ ได้ผลิตโปรแกรม **AIM** และ **ICQ** จนได้รับความนิยมอย่างแพร่หลาย ต่อมาทางบริษัทไมโครซอฟท์ (Microsoft Corporation) ก็ได้ผลิตโปรแกรม **MSN Messenger** เพื่อตอบสนองความต้องการให้กับผู้ใช้งานจดหมายอิเล็กทรอนิกส์ของทางไมโครซอฟท์ และในเวลาใกล้เคียงกันทาง Yahoo Incorporation ก็ผลิต **Yahoo Messenger** มาบริการผู้ใช้งานจดหมายอิเล็กทรอนิกส์ของทาง Yahoo เช่นเดียวกัน โปรแกรมรับส่งสารด่วนแต่ละโปรแกรมที่ถูกผลิตออกมานั้นต่างก็มีความสามารถในการใช้งานแตกต่างกันออกไป

2.4 การใช้ระบบรับส่งสารด่วนในอนาคต

ในอดีตผู้ผลิตต้องการให้โปรแกรมรับส่งสารด่วนสามารถรับส่งข้อความได้โดยตรง เพื่อให้ผู้ใช้งานสามารถสนทนากันได้อย่างรวดเร็วและสะดวกกว่าการใช้จดหมายอิเล็กทรอนิกส์ โดยนักวิเคราะห์เปรียบเทียบโปรแกรมรับส่งสารด่วนว่าเป็นเสมือนกับเรียลไทมอีเมล์ (Real-time Email) ซึ่งการใช้โปรแกรมรับส่งสารด่วนทำให้การสนทนาเป็นไปอย่างต่อเนื่องกว่าการใช้จดหมายอิเล็กทรอนิกส์ และประหยัดค่าใช้จ่ายกว่าการสนทนาโดยการใช้โทรศัพท์ เนื่องจากสามารถสนทนาในเวลาเดียวกันได้หลายคนและไม่มีการกำหนดอัตราค่าบริการระหว่างพื้นที่ ทำให้โปรแกรมรับส่งสารด่วนเป็นที่นิยมได้อย่างรวดเร็ว

เมื่อโปรแกรมรับส่งสารด่วนเป็นที่นิยมมากขึ้น ทำให้มีผู้ผลิตเพิ่มมากขึ้นเช่นกันและมีความสามารถที่เพิ่มมากขึ้นรองรับความต้องการของผู้ใช้งานได้อย่างครบถ้วน โดยโปรแกรมรับส่งสารด่วนไม่ได้เป็นเพียงโปรแกรมที่ใช้สำหรับสนทนาด้วยการรับส่งข้อความอย่างเดียวเท่านั้น ยังสามารถสนทนาด้วยเสียง สนทนาโดยผ่านทางเว็บแคมหรือสามารถสนทนารวมกันหลายคนเป็นต้น ทำให้ผู้ใช้งานโปรแกรมรับส่งสารด่วนเพิ่มขึ้นอย่างรวดเร็วและต่อเนื่อง

สำหรับการใช้งานในองค์กรธุรกิจนั้น จากการวิจัยพบว่าบริษัทไอบีเอ็มซึ่งเป็นองค์กรแรกๆ ที่นำโปรแกรมรับส่งสารด่วนไปใช้งานภายในองค์กร ตัวเลขอ้างอิงจากบริษัทประเมินการออกมาว่าพนักงานในบริษัทไอบีเอ็ม 100,000 คนแลกเปลี่ยนข้อความกันไปมา 1 - 2 ล้านข้อความในแต่ละวัน ซึ่งสามารถลดจำนวนการใช้จดหมายอิเล็กทรอนิกส์ภายในบริษัทได้ 30 - 40% และวอยซ์เมล์ได้ 10 - 15% แต่การนำโปรแกรมรับส่งสารด่วนไปใช้ในองค์กรธุรกิจกลับไม่เป็นที่นิยมและใช้งานมาก เนื่องจากโปรแกรมรับส่งสารด่วนในปัจจุบันยังไม่มีความปลอดภัยที่ดีพอ ส่งผลให้บุคคลอื่นภายนอกองค์กรสามารถดักจับข้อมูลที่ทำการสนทนาภายในองค์กรได้ หากมีการแก้ปัญหาด้านความไม่ปลอดภัยในโปรแกรมรับส่งสารด่วนแล้ว เชื่อกันว่าจะมีองค์กรธุรกิจจำนวนมากจะหันมาใช้โปรแกรมรับส่งสารด่วนสำหรับสนทนาภายในองค์กร ซึ่งบริษัท Verisign และอเมริกันออนไลน์กำลังเตรียมทีมพัฒนาโปรแกรมรับส่งสารด่วนแบบเข้ารหัส โดยคาดว่าจะเริ่มนำออกมาใช้งานได้ในปี ค.ศ.2003

บทที่ 3
ภาษาซีบนลินุกซ์ และ ซ็อกเก็ต

ก่อนที่เราจะเขียนโปรแกรมบนลินุกซ์ (Linux) เราจะต้องเข้าใจระบบการาทำงานสภาพแวดล้อม ระบบการจัดการไฟล์ต่างๆ การคอมไฟล์และการดีบักโปรแกรม รวมไปถึงระบบไฟล์ไลบรารีที่จำเป็นในการเขียนโปรแกรมภาษาซี ก่อนที่จะลงมือเขียนโปรแกรม เพราะถ้าเกิดปัญหาในการเขียนโปรแกรมจะได้แก้ไขถูกวิธี การใช้ซอฟต์แวร์เช่น Anjuta ช่วยในการเขียนโปรแกรมก็จะทำให้เขียนโปรแกรมได้ง่ายและรวดเร็วขึ้น การคำนึงถึงความปลอดภัยในการเขียนโปรแกรมก็เป็นสิ่งจำเป็นเพราะจะทำให้การโจมตีจากภายนอกทำได้ยากขึ้น

กลไกที่ทำให้โปรแกรมเมอร์สามารถเข้าถึง เน็ตเวิร์กโพรโตคอล เพื่อให้โพรเซสสามารถติดต่อกันแบบโลคอล หรือ ผ่านเน็ตเวิร์กได้นั้นซึ่งมีวิธีการหนึ่งที่เรียกว่า ซ็อกเก็ต (Socket) ที่ได้รับความนิยมในการเขียนติดต่อเครือข่าย

3.1 ภาษาซีบนลินุกซ์

3.1.1 ระบบยูนิกซ์คืออะไร

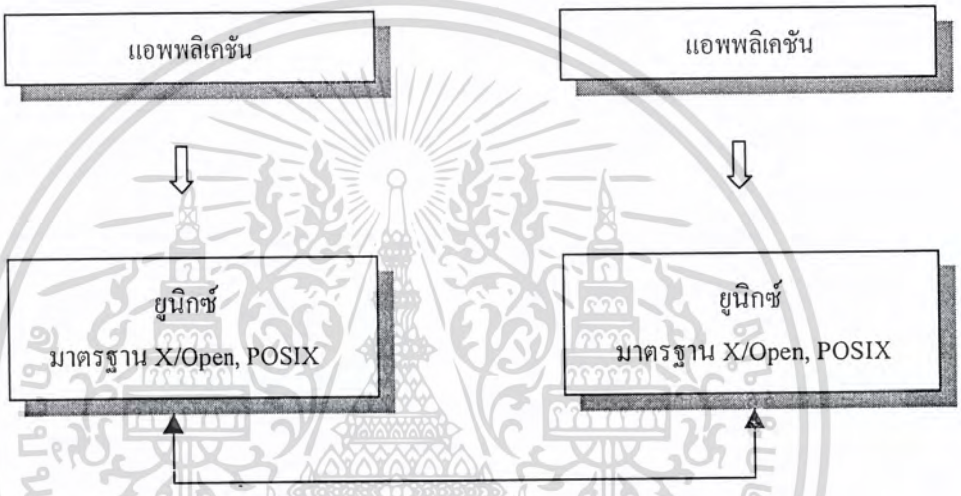
เริ่มแรกถูกพัฒนาขึ้นโดยห้องวิจัย Bell Laboratories ซึ่งในขณะนั้นเป็นส่วนหนึ่งของบริษัทโทรคมนาคม AT&T ได้รับอุปกรณ์ของ DEC (Digital Equipment PDP computer) ในปัจจุบันได้กลายมาเป็นระบบปฏิบัติการ มัลติทาสกิง (multi-tasking), มัลติยูเซอร์ (multi-user) ที่สำคัญใช้งานได้กับฮาร์ดแวร์หลายๆ แบบตั้งแต่พีซี (PC) ธรรมดา ไปจนถึงเซิร์ฟเวอร์มัลติโพรเซสเซอร์ (multiprocessor) หรือ ซูเปอร์คอมพิวเตอร์ (supercomputer)

ต่อมาได้มีการพัฒนาระบบมาเรื่อยๆ และมีบริษัทผู้ผลิตซอฟต์แวร์คอมพิวเตอร์ ได้สร้างระบบปฏิบัติการใหม่ขึ้นมาอีกหลายอย่างด้วยกัน โดยเฉพาะโครงสร้างการทำงานภายในที่เลียนแบบยูนิกซ์ดั้งเดิม (หรือแม้แต่การซื้อลิขสิทธิ์ และนำมาปรับปรุงเพิ่มเติม ออกมาเป็นแบรนด์เนมใหม่) ตัวอย่างของยูนิกซ์เช่น Solaris, Sun Microsystems, FreeBSD, Linux

มาตรฐานของระบบยูนิกซ์

เนื่องจากปัจจุบันมีหลายบริษัทด้วยกันที่สร้างแบรนด์เนม (Brand name) ของตนขึ้นมา บางบริษัทอาจจะเพิ่มลักษณะการทำงานพิเศษต่างๆ เข้าไปในระบบของตน ทำให้แอพพลิเคชันที่สร้างขึ้นมาบนแพลทฟอร์มที่ต่างกันไม่สามารถรันด้วยกันได้ เพื่อแก้ปัญหาดังกล่าว จึงได้มีการกำหนดมาตรฐานขึ้นมา ที่สำคัญมีดังนี้

- **IEEE 1003.1(1988)** จะกำหนดลักษณะการอินเตอร์เฟซกับยูนิกซ์รวมทั้งสภาพแวดล้อมในการทำงานบนระบบยูนิกซ์, ซึ่งเป็นที่รู้จักกันในชื่อของ POSIX มีจุดประสงค์เพื่อช่วยในการถ่ายเทซอร์สโค้ดของแอพพลิเคชันเมื่อต้องรันบนซีพียูที่ต่างกัน
- **X/Open** เป็นกลุ่มผู้ผลิตซอฟต์แวร์คอมพิวเตอร์ เริ่มมีการรวมกลุ่มกันในปี 1984 และได้มีการจัดทำคู่มือเพื่อช่วยในการถ่ายเทซอร์สโค้ดของแอพพลิเคชัน 7 เล่มด้วยกัน รายละเอียดจะกำหนดลักษณะการอินเตอร์เฟซกับเคอร์เนล รวมทั้งกำหนดยูทิลิทีต่างๆ ที่ควรจะมีไว้บนระบบยูนิกซ์ (X/Open 1989)



รูปที่ 3-1 การถ่ายเทซอร์สโค้ดของแอพพลิเคชันเมื่อต้องการรันบนแพลฟอร์มที่ต่างกัน

3.1.2 ลินุกซ์คืออะไร

เป็นเคอร์เนลของระบบเสมือนยูนิกซ์ ได้รับการพัฒนาขึ้นมาโดยโปรแกรมเมอร์ทั่วโลกผ่านเครือข่ายอินเตอร์เน็ต เป็นระยะเวลายาวนาน จนเป็นซอฟต์แวร์ระบบปฏิบัติการฟรีที่เชื่อถือได้ สามารถจะดาวน์โหลดได้ฟรี จากอินเตอร์เน็ตหรือจะซื้อจากผู้ให้บริการก็ได้

เนื่องจากลีนุกซ์สร้างขึ้นมาบนพื้นฐานของยูนิกซ์ ดังนั้นโปรแกรมที่สร้างขึ้นมาบนลีนุกซ์สามารถจะนำไปคอมไพล์และรันบนยูนิกซ์ได้

อันที่จริงแล้วลีนุกซ์ได้รับการพัฒนาจากระบบ Minix ของ Andy Tanenbaum ซึ่งเป็นระบบเสมือนลีนุกซ์เล็กๆ ตัวหนึ่ง โดย Linus Torvalds ที่มหาวิทยาลัย Helsinki โดยความช่วยเหลือของโปรแกรมเมอร์ระบบยูนิกซ์ผ่านเครือข่ายอินเตอร์เน็ต จากนั้นก็ได้เติบโตขึ้นมาเรื่อยๆจนกลายเป็นระบบที่ประกอบด้วยยูทิลิทีหรือโปรแกรมอื่นที่สมบูรณ์ในปัจจุบัน อย่างไรก็ตามเคอร์เนลของลีนุกซ์ก็ไม่ได้ใช้ซอร์สโค้ดต้นฉบับของยูนิกซ์จาก AT&T แต่อย่างใด

ตามที่ได้กล่าวมาตั้งแต่ต้นแล้วว่า ลีนุกซ์เป็นเพียงเคอร์เนลตัวหนึ่ง คุณสามารถจะนำซอร์สโค้ดมาคอมไพล์แล้วติดตั้งพร้อมกับซอฟต์แวร์ฟรีตัวอื่นๆ ก็จะกลายเป็นระบบเสมือนยูนิกซ์ได้ (ลีนุกซ์) ถึงแม้ว่าในปัจจุบันจะประกอบไปด้วยส่วนอื่นนอกจากเคอร์เนลอีกมาก ซอฟต์แวร์ส่วนใหญ่จะมาจากโปรเจค GNU จากองค์การ Free Software Foundation

อย่างไรก็ตามการสร้างระบบด้วยวิธีการดังกล่าวก็เป็นวิธีการที่ไม่ค่อยสะดวก ในปัจจุบันก็ได้มีผู้ให้บริการ รวบรวมแพ็คเกจซอฟต์แวร์เหล่านี้เข้าไว้บนซีดีรอมเดียวกันซึ่งจะประกอบไปด้วยเคอร์เนล, โปรแกรมยูทิลิทีอื่น หรือระบบ X Windows ซึ่งเป็นระบบกราฟฟิกอินเตอร์เฟซที่ใช้งานบนยูนิกซ์, โปรแกรมที่ช่วยในการติดตั้งหรือเท็กซ์ไฟล์ประกอบอื่นๆ

สำหรับลีนุกซ์ที่เป็นที่รู้จักและใช้งานกันทั่วไป เช่น Slackware, Debian, RedHat, Caldera, OpenLinux เป็นต้น

3.1.3 องค์กร FSF และโครงการ GNU

สำหรับแอพพลิเคชันและทูลบนลีนุกซ์ถูกเขียนขึ้นมาโดยโปรแกรมเมอร์เป็นจำนวนมาก สามารถจะดาวน์โหลดได้ฟรีจากเครือข่ายอินเตอร์เน็ต ถึงแม้ว่าจะเสียค่าใช้จ่ายในการจัดหาโปรแกรมบ้าง โปรแกรมเหล่านี้ไม่มีใครเป็นเจ้าของ คุณสามารถจำหน่ายได้โดยไม่ผิดกฎหมาย (ตามข้อตกลง GNU General Public Licenses) ส่วนใหญ่แล้วจะจำหน่ายพร้อมซอร์สโค้ด เพื่อให้โปรแกรมเมอร์คนอื่นสามารถจะพัฒนาต่อไปได้

FSF ถูกจัดตั้งขึ้นโดย Richard Stallman (รวมทั้งเป็นผู้จัดตั้งโครงการ GNU) ซึ่งเป็นโครงการพัฒนาระบบปฏิบัติการเสมือนยูนิกซ์ ให้สภาพการทำงานเสมือนระบบยูนิกซ์ ถึงแม้ว่าจะแตกต่างกันมากในส่วนของเคอร์เนล แต่ก็สามารถจะรันแอพพลิเคชัน ของระบบยูนิกซ์ได้

นอกจากนี้แล้วโครงการ GNU ยังสร้างซอฟต์แวร์อีกหลายอย่าง มักจะเรียกซอฟต์แวร์เหล่านี้ว่า GNU Software, การจำหน่ายจะเป็นไปตามข้อตกลง GNU Public License (GPL) ซึ่งจะเป็นการป้องกันไม่ให้มีการอ้างลิขสิทธิ์หรือผูกขาดซอฟต์แวร์ อันเป็นการจำกัดการใช้งานซอฟต์แวร์ฟรีเหล่านี้

> อนึ่ง จากคำว่า Free Software Foundation, คำว่า Free ในที่นี้หมายถึงซอร์สโค้ดฟรี เพื่อให้ผู้อื่นสามารถจะนำไปศึกษาหรือพัฒนาต่อไปได้ ไม่ได้หมายถึงว่าจะได้ซอฟต์แวร์มาฟรีๆ จริงอยู่ว่า สามารถจะดาวน์โหลดได้จากอินเตอร์เน็ต หรืออาจจะซื้อซีดีรอม ของผู้ให้บริการ แต่ก็ต้องเสียค่าใช้จ่ายบ้าง

สำหรับซอฟต์แวร์ที่มีการใช้งาน ภายใต้ข้อตกลง GPL เช่น

gcc	คอมไพล์เลอร์ภาษาซี (C)
g++	คอมไพล์เลอร์ภาษาซีพลัซพลัซ (C++)
gdb	ดีบักเกอร์

gnumake	เป็นคำสั่งที่คอมแพททิเบิลกับคำสั่ง make บนระบบยูนิกซ์
bash	เป็นเชลล์ที่ทำหน้าที่ติดต่อกับผู้ใช้งาน
emacs	เท็กซ์อีดิเตอร์
CVS	เป็นโปรแกรมควบคุมซอร์สโค้ด
cpio	เป็นโปรแกรมคัดลอกอาร์ไคฟไฟล์ (archive file) จากเทปหรือดิสก์
gzip	เป็นโปรแกรมบีบไฟล์ นอกจากนี้ยังใช้ขยายไฟล์จากการบีบของโปรแกรม pack ได้ด้วย
perl	เป็นภาษาเขียนโปรแกรม เขียนขึ้นมาโดย Larry Wall ที่รวมเอาลักษณะหลายอย่างเข้าด้วยกัน เช่น sed, awk หรือ การเขียนโปรแกรมบนเชลล์ หรือภาษาซี รวมทั้งฟังก์ชันระดับต่ำ และไลบรารีของภาษาซีอีกหลายอย่างด้วยกัน

3.1.4 คอมไพล์เลอร์ภาษาซี

คอมไพล์เลอร์ภาษาซี สำหรับลินุกซ์มีคอมไพล์เลอร์ต่างๆ อยู่เช่น c89 (POSIX), cc , gcc ส่วนใหญ่จะเป็นคอมไพล์เลอร์มาจากโครงการ GNU

สำหรับโครงการนี้จะใช้ gcc version 2.95.4 (Debian prerelease) ซึ่งเป็นเวอร์ชันที่ได้รับการรับร้องแล้วว่ามีความเสถียรภาพ gcc เป็นคอมไพล์เลอร์ที่มาจากโครงการ GNU C เป็นคอมไพล์เลอร์จะมีอยู่ในแพ็กเกจของลินุกซ์อยู่แล้วและยังคอมแพททิเบิลกับมาตรฐาน ANSI C สามารถดูรายละเอียดของ gcc ได้ที่ http://gcc.gnu.org

ชื่อของคอมไพล์เลอร์จะชื่อว่า gcc ใช้คอมไพล์เลอร์ไฟล์ที่มีสกุล .c และจะใช้ออปต์ชัน –c เป็นการบอกว่าคอมไพล์โปรแกรมไปเป็น Object file (ไฟล์ที่มีสกุล .o) เท่านั้น เช่น ถ้าเราต้องการคอมไพล์ไฟล์ชื่อ main.c จะต้องพิมพ์คำสั่งที่พรอมต์ดังนี้

```
% gcc –c main.c
```

ผลลัพธ์จะได้ไฟล์ชื่อ main.o มาหนึ่งไฟล์เป็น object file อยู่ไดเรกทอรีเดียวกับ main.c

3.1.5 ตำแหน่งของไฟล์ที่ใช้ในงานเขียนโปรแกรมบนยูนิกซ์

สำหรับนักพัฒนาโปรแกรมบนระบบยูนิกซ์ ตำแหน่งของทูลที่ใช้งานและรีซอร์สเป็นสิ่งที่สำคัญ ต่อไปจะเป็นหัวข้อเกี่ยวกับไดเรกทอรีและไฟล์ที่สำคัญ ซึ่งในที่นี้จะอธิบายโดยใช้ลินุกซ์เป็นหลัก แต่ก็สามารถจะนำหลักการไปใช้กับระบบลินุกซ์อื่นได้เช่นกัน

3.1.5.1 ตำแหน่งของโปรแกรม

โปรแกรมจะเก็บไว้ในไดเรกทอรีตามจุดประสงค์ของการใช้งาน เช่น

/usr/bin	แอพพลิเคชันของระบบที่ใช้งานทั่วไป
/usr/local/bin	ส่วนโปรแกรมที่เพิ่มเข้ามาโดยผู้ควบคุมระบบ ใช้สำหรับคอมพิวเตอร์บางเครื่องโดยเฉพาะ หรือใช้งานสำหรับเครือข่ายโลคอลเน็ตเวิร์ก ผู้ควบคุมระบบมักจะเก็บโปรแกรมไว้ที่ไดเรกทอรี /usr/local ซึ่งมักจะมีประโยชน์ในกรณีที่ต้องติดตั้งหรืออัพเกรดระบบ ไดเรกทอรี /usr/local จะถูกเก็บไว้ไม่ถูกลบ
/usr/bin หรือ /usr/local/bin	สำหรับโปรแกรมไดรฟ์เวอร์ของคอมไพล์เลอร์ GNU gcc แต่ตำแหน่งของคอมไพล์เลอร์นี้อาจจะเปลี่ยนไปได้ จะกำหนดเมื่อคอมไพล์คอมไพล์เลอร์เอง (หรืออาจจะเปลี่ยนไปตามระบบก็ได้) สำหรับลีนุกซ์จะเก็บคอมไพล์เลอร์ของ GNU C/C++ หรือเฮเดอร์ไฟล์ไว้ที่ /usr/lib/gcc – lib/i486 – unknown – linux/2.7.2

นอกจากนี้ยังมีโปรแกรมพิเศษบางชนิด ที่จะมีไดเรกทอรีเป็นของตนเอง หนึ่งในจำพวกนี้ก็คือระบบ X Windows ซึ่งปกติจะเก็บไว้ที่ไดเรกทอรี /usr/X11 หรือ /usr/X11R6 สำหรับ Revision 6 หรือ /usr/openwin สำหรับระบบกราฟฟิกอินเตอร์เฟช OpenWindows ของ Sun Solaris

3.1.5.2 ไฟล์เฮเดอร์ (File Header)

การเขียนโปรแกรมในภาษาซี จำเป็นจะต้องใช้ไฟล์เฮเดอร์ซึ่งเป็นไฟล์ที่มีการกำหนดค่าคงที่ หรือตัวแปรไว้แล้ว ทำให้สามารถจะเรียกค่าคงที่หรือตัวแปรนั้นมาใช้งานได้เลย

/usr/include	สำหรับภาษาซี (และซับไดเรกทอรีภายใต้ไดเรกทอรีนี้) เช่น
/usr/include/sys	สำหรับระบบยูนิกซ์บางชนิด หรือ
/usr/include/linux	สำหรับลีนุกซ์

สำหรับการเขียนโปรแกรมในภาษาอื่น จะเก็บไว้ในไดเรกทอรีที่คอมไพล์เลอร์จะค้นหาโดยอัตโนมัติ เช่น

/usr/include/g++	สำหรับ GNU C++
/usr/inclue/X11	สำหรับการเขียนโปรแกรมบน X Windows

เมื่อคอมไพล์สามารถจะระบุตำแหน่งที่ไม่เป็นมาตรฐานของไฟล์เฮเดอร์ ได้โดยใช้แฟลก –I เช่น

```
$ gcc  - I/usr/include/X11  richie.c
```

คอมไพล์เลอร์จะไปค้นหาที่ตำแหน่ง /usr/include/X11 (รวมทั้งในตำแหน่งมาตรฐานที่ได้กำหนดไว้แล้ว) นอกจากนี้ถ้าต้องการค้นหาเฮเดอร์ไฟล์ที่มีหน้าที่หรือ สถานะของการทำงานบางอย่างสามารถจะใช้คำ

สั่ง grep ให้ค้นหาภายในเฮเดอร์ไฟล์ที่ต้องการ เช่น ถ้าต้องการจะรู้ว่าโพรเซสจะเปิดไฟล์ได้มากที่สุดเท่าใด เข้าไปยังที่ไดเรกทอรี /uer/include ใช้คำสั่งต่อไปนี้

```
$ grep  _POSIX_OPEN_MAX  *.h
$posixl_lim.h: # define              POSIX_OPEN_MAX  16
```

5.1.5.3 ไฟล์ไลบรารี

ไลบรารีเป็นฟังก์ชันที่คอมไพล์ไว้แล้ว สามารถจะเรียกใช้งานฟังก์ชันที่อยู่ในไลบรารีได้ เช่น ไลบรารีที่เกี่ยวกับการรับหรือแสดงข้อมูล (stdio library) หรือไลบรารีเกี่ยวกับฐานข้อมูล (dbm library)

โดยทั่วไปไลบรารีของระบบจะถูกเก็บไว้ในไดเรกทอรี /lib และ /usr/lib คอมไพล์เลอร์จะค้นหาไลบรารีเฉพาะที่ไดเรกทอรีมาตรฐานเท่านั้น ถ้ามีไลบรารีที่ตำแหน่งต่างออกไปต้องบอกให้คอมไพล์เลอร์ทราบ ไลบรารีมีวิธีกำหนดชื่อที่เป็นมาตรฐาน และจะต้องกำหนดเมื่อสั่งคอมไพล์โปรแกรมด้วย

ชื่อของไลบรารีจะเริ่มด้วย lib จากนั้นจะตามด้วยอักษรที่ระบุว่าไลบรารีนี้เป็นไลบรารีของอะไร (c สำหรับไลบรารีของภาษาซี หรือ m เป็นไลบรารีสำหรับฟังก์ชันการคำนวณ) ส่วนสุดท้ายจะเป็นชนิดของไลบรารีเริ่มต้นด้วย "."

.a	เป็นไลบรารีธรรมดา (static library)
.so หรือ .sa	เป็นไลบรารีที่ใช้งานร่วมกันได้ (shared libraries)

โดยปกติแล้วไฟล์ไลบรารีจะเก็บไว้ที่ตำแหน่งข้างต้น สามารถใช้คำสั่ง Is –I/usr/lib ตรวจสอบได้ นอกจากนี้ยังสามารถกำหนดให้คอมไพล์เลอร์ ค้นหาไลบรารีในตำแหน่งที่ต้องการ โดยกำหนดแบบ full path เพิ่มเข้าไป ดังต่อไปนี้

```
$ gcc  -o  richie  richie.c  /usr/lib/libm.a
```

จากคำสั่งข้างต้น สามารถใช้แฟลก –lm กำหนดไลบรารีสั้นๆ ได้ เช่น

```
$ gcc  -o  richie  richie.c  -lm
```

-lm (ไม่มีช่องว่างระหว่าง l และ m) เป็นคำย่อของ libm.a ภายในไดเรกทอรีไลบรารีมาตรฐาน (ในกรณีนี้หมายถึง /usr/lib) ข้อดีของการใช้งาน –lm ก็คือจะใช้งาน shared library (ถ้ามีอยู่)

การกำหนดให้คอมไพล์เลอร์ใช้ตำแหน่งไลบรารีที่นอกเหนือไปจากไดเรกทอรีมาตรฐาน จะใช้แฟลก –L เช่น

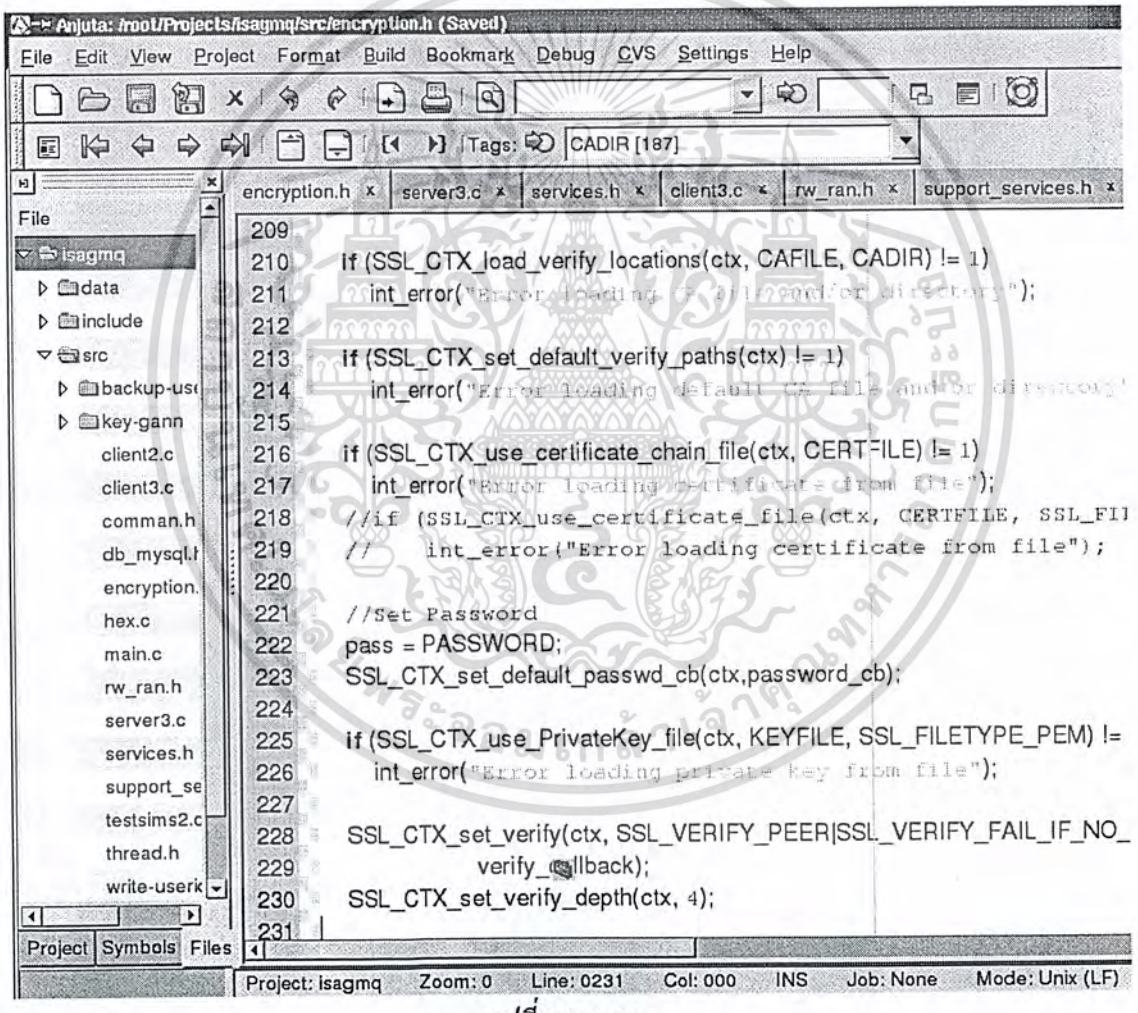
```
$ gcc  -o  xllrichie  -L/usr/openwin/lib  xllrichie.c  -lx11
```

เมื่อคอมไพล์และลิงค์โปรแกรม x11richie จะใช้ไลบรารี libx11 ที่ไดเรกทอรี /usr/openwin/lib ออปชัน –o name จะกำหนดชื่อไฟล์เอาต์พุตที่ต้องการ ถ้าไม่มีการใช้ออปชันนี้ คอมไพล์เลอร์จะเก็บไฟล์เอาต์พุต (ที่สามารถจะรันได้) ไว้ที่ไฟล์ชื่อ a.out

3.2 เครื่องมือสำหรับพัฒนาโปรแกรมภาษาซี

3.2.1 Anjuta

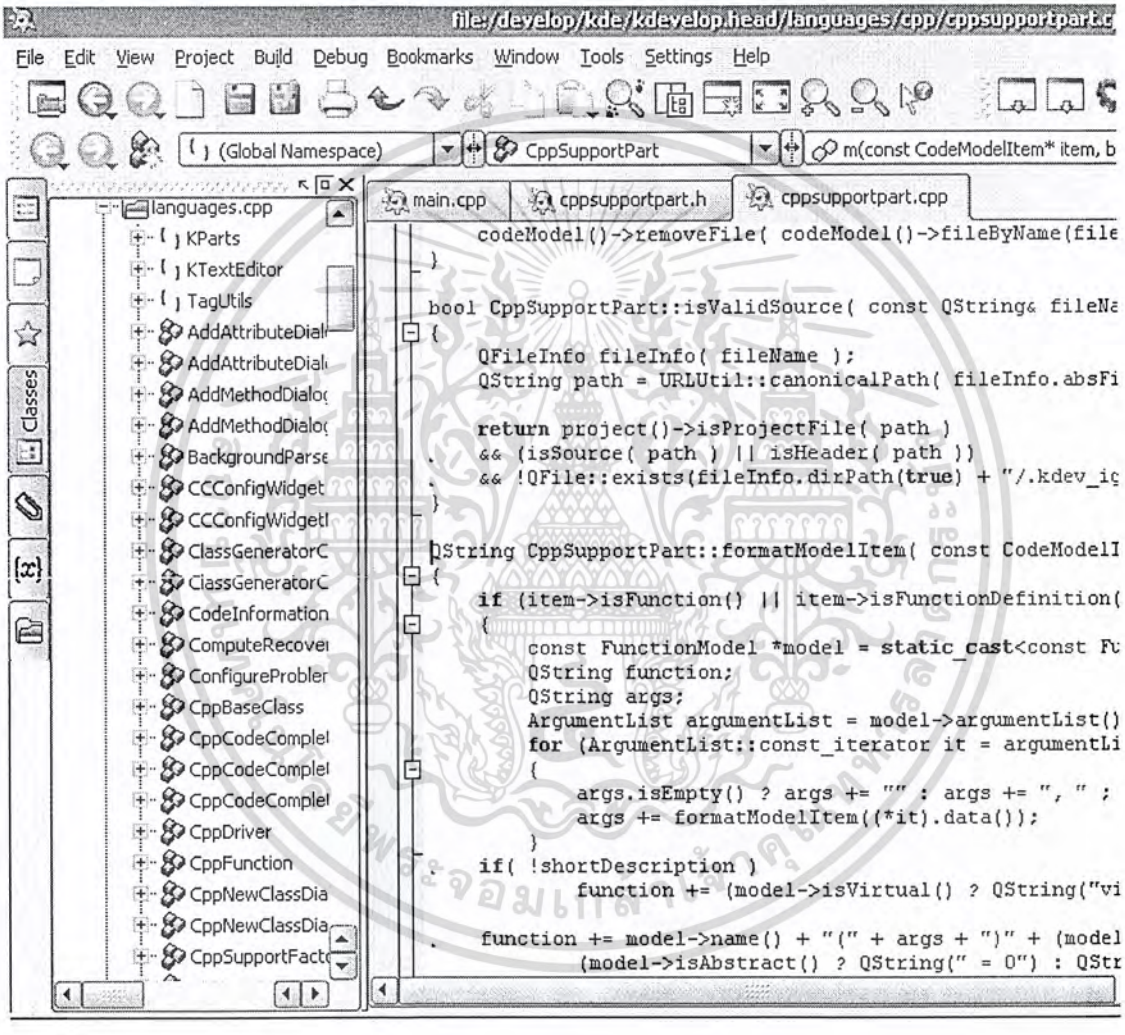
Anjuta เป็นซอฟต์แวร์ที่ใช้ในการบริหารจัดการและพัฒนาเชิงโปรแกรม มิ่งที่เรียกว่า Integrated Development Environment (IDE) สำหรับ C and C++ ที่ทำงานบน GNU/Linux และสามารถใช้เขียนโปรแกรมบน GTK/GNOME ได้อีกด้วยและ Anjuta เป็นซอฟต์แวร์ที่อยู่ภายใต้ GPL (General Public License) จะทำให้สามารถใช้ได้ฟรี ข้อดีของ Anjuta คือ หนึ่งง่ายในการเขียนโปรแกรม สองช่วยจัดการบริหารโปรเจ็คสามมี application wizards ให้เลือกใช้ สี่มี debugger ภายในตัวซอฟต์แวร์เอง ห้าสามารถเปิดเอดิเตอร์ (editer) ในลักษณะของบราวซิง (browsing) ได้ และสามารถตรวจสอบไวยากรณ์โดยการแสดงเป็นไฮไลต์ได้



รูปที่ 3-2 Anjuta

3.2.2 Kdeveloper

Kdeveloper เป็นซอฟต์แวร์ประเภท IDE (Integrated Development Environment) สำหรับ C และ C++ ทำงานบน GNU/Linux เหมือนกันกับ Anjuta ซึ่งมีข้อดีคล้ายกันแต่ Kdevelop สามารถใช้ได้กับโปรแกรมหลากหลายชนิดกว่า ดังนี้ Ada, Bash, C/C++, Fortran, Haskell, Java, Pascal, Perl, PHP, Python และ Kdevelop ก็เป็นซอฟต์แวร์ที่อยู่ภายใต้ GPL (General Public License) ทำให้สามารถใช้ได้ฟรีเหมือนกัน



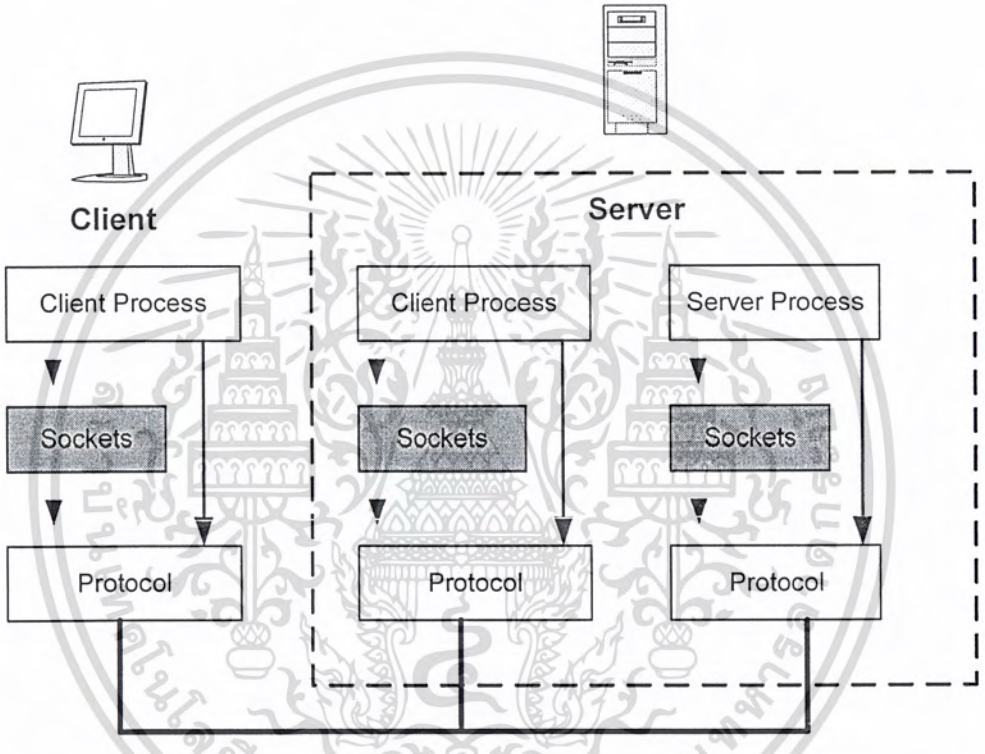
รูปที่ 3-3 Kdevelop 3.0 using KDE 3.2 and Plastik theme in IDEAl mode

3.3 ซ็อกเก็ต (Socket)

3.3.1 ซ็อกเก็ตคืออะไร

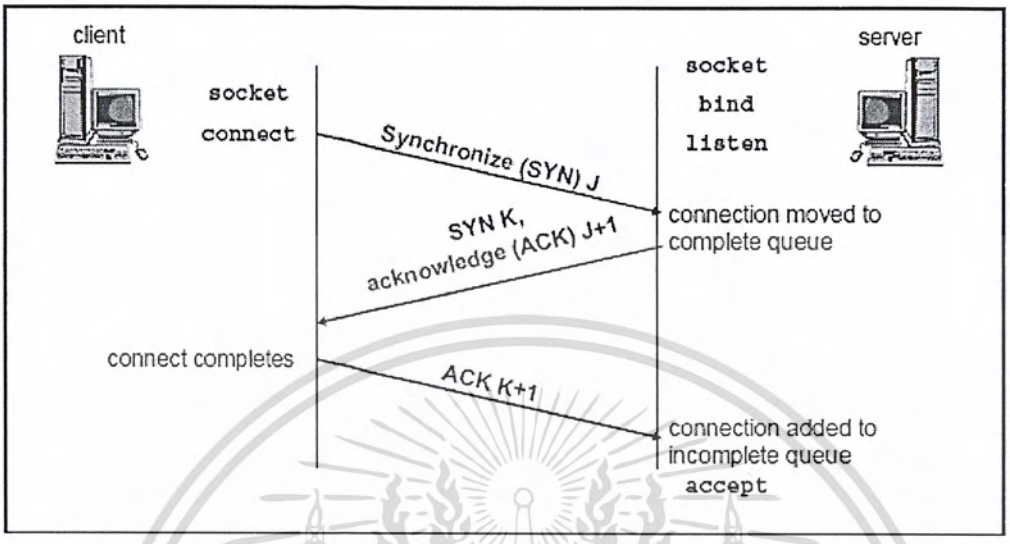
ซ็อกเก็ต (Socket) เป็นกลไกที่ทำให้โปรแกรมเมอร์สามารถเข้าถึงเน็ตเวิร์กโพรโตคอลเพื่อให้โพรเซสสามารถติดต่อกันแบบไลคอลหรือ ผ่านเน็ตเวิร์กได้

แนวทางการสร้างและใช้งานซ็อกเก็ตจะต้องมีการระบุโพรเซสไคลเอ็นต์และเซิร์ฟเวอร์อย่างชัดเจน นอกจากนี้โพรเซสเซิร์ฟเวอร์เพียงโพรเซสเดียว ปรกติแล้วสามารถทำงานได้กับโพรเซสไคลเอ็นต์หลายๆโพรเซสได้

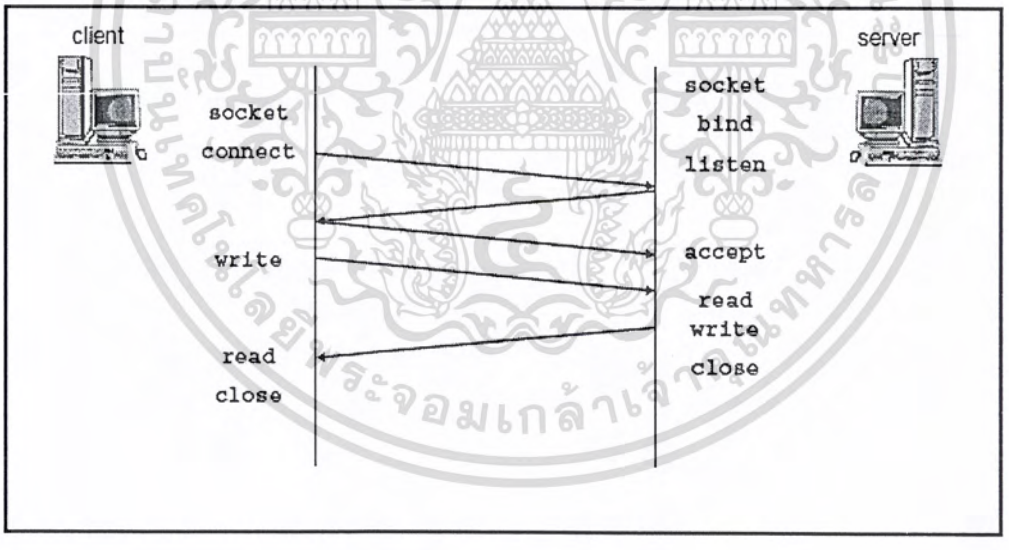


รูปที่ 3-4 โพรเซสการเชื่อมต่อของไคลเอ็นต์เซิร์ฟเวอร์โดยการใช้ ซ็อกเก็ต

3.3.2 หลักการเบื้องต้นการติดต่อโดยใช้ซ็อกเก็ต (Connection oriented)



รูปที่ 3-5 TCP Connection Setup



รูปที่ 3-6 Socket Function Setup

สิ่งแรกที่ต้องทำคือ เซิร์ฟเวอร์จะต้องสร้างซ็อกเก็ตซึ่งถือว่าเป็นรีซอร์สของระบบอย่างหนึ่ง ที่ควบคุมโดยเซิร์ฟเวอร์ โดยใช้ฟังก์ชันซ็อกเก็ต (แต่ในขณะนี้ยังไม่สามารถใช้งานได้)

ลำดับต่อไปคือกำหนดชื่อให้กับซ็อกเก็ต โดยปรกติ โลคอลซ็อกเก็ต (local socket) จะเป็นชื่อไฟล์ แต่ถ้าเป็นเน็ตเวิร์กซ็อกเก็ต (network socket) จะเป็นออบเจ็กต์กำหนดจุดเชื่อมต่อของโพรเซสอื่นประกอบด้วยหมายเลขพอร์ตและจุดเชื่อมต่อของโพรเซส (port number/access point) ซึ่งจะเป็นค่าสำหรับเฉพาะเน็ตเวิร์กนั้น การกำหนดชื่อให้กับโพรเซสทำได้โดยใช้ฟังก์ชัน **bind**

จากนั้นโพรเซสเซิร์ฟเวอร์จะรอรับการเชื่อมต่อจากโพรเซสไคลเอ็นต์โดยใช้ซ็อกเก็ตที่กำหนดชื่อไปข้างต้น โดยใช้ฟังก์ชัน **listen** เพื่อสร้างคิว (queue) รอรับการเชื่อมต่อ หลังจากนั้นถ้ามีไคลเอ็นต์โพรเซสติดต่อเข้ามาโพรเซสเซิร์ฟเวอร์จะตอบรับการเชื่อมต่อโดยใช้ฟังก์ชัน **accept** ซึ่งจะเป็นการสร้างซ็อกเก็ตขึ้นมาใหม่ (แตกต่างกับซ็อกเก็ตที่กำหนดชื่อเอาไว้ในขั้นแรก)

ซ็อกเก็ตใหม่นี้จะเป็นซ็อกเก็ตสำหรับติดต่อกับไคลเอ็นต์โพรเซสเฉพาะของแต่ละไคลเอ็นต์ที่ทำการติดต่อเข้ามา ดังนั้นสำหรับซ็อกเก็ตที่กำหนดชื่อโดยฟังก์ชัน bind ก็ยังสามารถจะนำไปใช้งานสำหรับไคลเอ็นต์โพรเซสอื่นได้เช่นกัน เพราะโดยปรกติแล้ว เซิร์ฟเวอร์จะให้บริการแก่ไคลเอ็นต์ได้หลายๆ ไคลเอ็นต์ในเวลาเดียวกัน

สำหรับซ็อกเก็ตในฝั่งไคลเอ็นต์จะสร้างได้ง่ายกว่านั้น คือใช้ฟังก์ชันซ็อกเก็ตและเรียกฟังก์ชัน **connect** เพื่อสร้างช่องทางการเชื่อมต่อกับเซิร์ฟเวอร์โพรเซส หลังจากสร้างช่องทางการเชื่อมต่อเสร็จสิ้นแล้ว เราสามารถจะใช้หมายเลขเพื่อแทนช่องทางในการเชื่อมต่อนั้นๆ โดยจะสามารถใช้หมายเลขระบุถึงช่องทางการสื่อสารข้อมูลระหว่างโพรเซสทั้งสองได้

3.3.3 องค์ประกอบของซ็อกเก็ต

เราจะมาดูองค์ประกอบบางอย่างของระบบเน็ตเวิร์กบนยูนิกซ์ ซึ่งก็เป็นส่วนประกอบที่กำหนดไว้ใน Socket จะประกอบด้วย 3 ส่วนด้วยกันคือ รูปแบบของการกำหนดแอดเดรสของคอมพิวเตอร์บนเน็ตเวิร์ก (Domain), ชนิดของชนิดซ็อกเก็ต (Socket type) และโพรโตคอลที่ใช้งาน

3.3.3.1 ซ็อกเก็ตโดเมน (Socket Domains)

เป็นตระกูลของโพรโตคอลที่จะใช้งาน ที่จะใช้งานบ่อยก็คือ AF_INET หมายถึงโพรโตคอลที่จะใช้งานกันโอเพนเน็ตเวิร์ก (Open network) ใช้กับระบบโลคอลเน็ตเวิร์กบนยูนิกซ์ หรืออินเตอร์เน็ต (TCP/IP)

ระบบจะใช้โพรโตคอล IP กำหนดแอดเดรสในการติดต่อ โดยมีรูปแบบของแอดเดรสเฉพาะประกอบด้วยตัวเลขสี่จำนวนขั้นด้วยจุด และแต่ละตัวจะมีค่าตั้งแต่ 0 - 255 ซึ่งเป็นแอดเดรสจริงของคอมพิวเตอร์บนอินเตอร์เน็ต (สำหรับแอดเดรสที่ใช้เป็นตัวอักษรนั้น เมื่อจะใช้งานจริง ก็จะได้รับการแปลง

เป็นแอดเดรสข้างต้น) เมื่อไคลอ็นติดต่อกับเน็ตเวิร์กเซิร์ฟเวอร์ ผ่านซ็อกเก็ตจะต้องทราบไอพีแอดเดรส (IP Address) ของเซิร์ฟเวอร์นั้น จึงจะติดต่อได้

สำหรับการกำหนดชนิดของการบริการจากเซิร์ฟเวอร์ของอินเตอร์เน็ต (หรือ TCP/IP Network) จะกำหนดให้ใช้ไอพีพอร์ตซึ่งเป็นหมายเลขขนาด 16 บิต (ซ็อกเก็ตจะต้องรับการกำหนด port ก่อนที่จะสร้างแชลแนลในการติดต่อได้) เช่น โพรโตคอลทีซีพีกำหนดให้หมายลข 21(decimal) แทนเซิร์ฟเวอร์ เอฟทีพี (FTP) หรือ ยูทีพี (UDP) จะใช้หมายเลข 69 แทนเซิร์ฟเวอร์ทีเอฟทีพี (TFTP) เป็นต้น

โพรเซสเซิร์ฟเวอร์ จะรอที่การติดต่อที่พอร์ตเฉพาะ เช่นกันบนระบบยูนิกซ์จะกำหนดให้มีค่าตั้งแต่ 1-1023 ซึ่งจะเป็นพอร์ตที่ใช้งานสำหรับโพรเซสของระบบเท่านั้น ปรกติจะต้องเป็นซูเปอร์ยูเซอร์ (superuser) ถึงจะสามารถจองพอร์ตเหล่านี้ไว้ทำงานได้ สำหรับพอร์ตที่ได้กำหนดไว้แล้ว เช่น printer spooler(515), ftp(21), httpd(80) (เซิร์ฟเวอร์สุดท้ายเป็นเซิร์ฟเวอร์สำหรับให้บริการข้อมูลเว็บเพจ, Web Server)

สำหรับ โดเมนที่ได้ใช้งานไปในตัวอย่างแรก เป็น Unix file system Domain(AF_UNIX) ซึ่งใช้สำหรับ socket ในการติดต่อกับโพรเซสที่อยู่บนคอมพิวเตอร์เครื่องเดียวกัน โพรโตคอลที่ใช้ในการติดต่อกันก็คือ ชื่อของไฟล์นั่นเอง ส่วนแอดเดรสที่ใช้ในการติดต่อก็คือ ชื่อของไฟล์แบบ absolute filename ในตัวอย่างข้างต้นก็คือ server_socket ซึ่งจะพบเมื่อรันโปรแกรมเซิร์ฟเวอร์

Domains อื่นที่อาจจะใช้งานกัน เช่น AF_ISO สำหรับเน็ตเวิร์กที่ใช้โพรโตคอลของ ISO หรือ AF_NS เป็นโพรโตคอล Xerox Network System จะไม่กล่าวถึงในที่นี้

3.3.3.2 ชนิดซ็อกเก็ต (Socket Types)

แต่ละ Domains อาจจะมีหลายๆ ชนิดซ็อกเก็ตซึ่งหมายถึงวิธีการส่งข้อมูลของแต่ละ Domains ไม่เพียงแต่ AF_UNIX ซึ่งเราสามารถจะใช้การติดต่อได้ทั้ง 2 ทางเท่านั้น โดเมนของเน็ตเวิร์ก AF_INET จะมีวิธีการส่งข้อมูล 2 วิธีคือ streams (TCP) และ datagrams (UDP)

วิธีการแรก จะมีความน่าเชื่อของการส่งข้อมูลมากกว่า จะสร้างแชนเนลของการเชื่อมต่อจนกระทั่งการติดต่อนั้นเสร็จสิ้น (Connection-orient) รวมทั้งสามารถจะติดต่อกันได้ทั้ง 2 ทาง ข้อมูลที่ถูกส่งไปจะได้รับการรับรอง ติดตามผลการส่ง ว่าส่งถึงหรือไม่ ถ้ามีปัญหาจะต้องส่งใหม่, ถ้าข้อมูลมีขนาดใหญ่เกินไปจะแบ่งออกเป็นส่วนเล็กๆ และไปประกอบกันอีกทีที่ปลายทาง, สตรีมซ็อกเก็ต (stream socket) กำหนดโดย SOCKET_STREAM ในโดเมน AF_INET

ในโครงงานนี้จะใช้ซ็อกเก็ตนี้เนื่องจากเป็นที่ยอมรับและใช้งานกันอย่างแพร่หลาย (รวมทั้งบนอินเตอร์เน็ตด้วย)

ส่วนอีกวิธีหนึ่งคือ เดตาแกรมซ็อกเก็ต (datagram socket) กำหนดโดยใช้ SOCK_DGRAM จะไม่สร้างแชนเนลในการเชื่อมต่อ (connectionless) หรือแม้แต่การจัดลำดับของข้อมูลที่จะรับ ส่งเสร็จแล้วก็ไม่มีการติดตามผลการส่งข้อมูล ไม่มีการตรวจสอบใดๆทั้งสิ้น นอกจากนี้ยังมีการจำกัดขนาดข้อมูลที่จะส่ง ถูก

กำหนดไว้ในโดเมน AF_INET โดยใช้โพรโตคอล UDP/IP ในการติดต่อ

อย่างไรก็ตามในแง่ของการใช้งานรีซอร์ส จะไม่สิ้นเปลืองเวลาของระบบมากนักเนื่องจากไม่ต้องจองแชนเนลใช้งาน โดยทั่วไปจะใช้สำหรับส่งข้อมูลบางอย่าง ที่ต้องทำเป็นประจำ (single-shortinquiries)

	AP_UNIX	AP_INET
SOCK_STREAM	Yes	TCP
SOCK_DGRAM	Yes	UDP

3.3.3.3 ซ็อกเก็ตโพรโตคอล (Socket Protocols)

หลังจากที่ได้กำหนดตระกูลเน็ตเวิร์กของโพรโตคอลแล้ว แล้วก็ต้องเลือกโพรโตคอลที่จะใช้งานจริง ซึ่งจะมีความจำเพาะต่อตระกูลของโพรโตคอลและ วิธีการส่งข้อมูลที่ใช้งาน ดังที่ได้อธิบายไปแล้วตามตารางต่อไปนี้

ตระกูลของโพรโตคอล	วิธีการส่งข้อมูล	ชื่อที่ใช้งานใน socket	โพรโตคอลที่ใช้งานจริง
AF_INET	SOCK_DGRAM	IPPROTO_UDP	UDP
AF_INET	SOCK_STREAM	IPPROTO_TCP	TCP
AF_INET	SOCK_RAW	IPPROTO_ICMP	ICMP
AF_INET	SOCK_RAW	IPPROTO_RAW	(Raw)

อนึ่ง ในบทนี้จะเป็นรายละเอียดของซ็อกเก็ตเกี่ยวข้องกับตระกูลโพรโตคอลบนระบบเน็ตเวิร์กเปิดของยูนิกซ์ (AF_INET) แบบ connection oriented (TCP/IP) และสำหรับติดต่อกันบนคอมพิวเตอร์เครื่องเดียวกัน (AF_UNIX) เท่านั้น

3.3.4 การใช้งานซ็อกเก็ต (socket, bind, listen, accept, connect, close)

3.3.4.1 การสร้างซ็อกเก็ต

โดยใช้ฟังก์ชันระดับต่ำคือ **socket** ฟังก์ชั่นนี้จะส่งค่ากลับเป็นหมายเลขที่ใช้เพื่ออ้างอิงถึงซ็อกเก็ตนั้น

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

จะได้ออบเจ็คที่ใช้ติดต่อระหว่างโพรเซส ที่อยู่บนคอมพิวเตอร์เครื่องเดียวกันหรือบนเน็ตเวิร์ก

domain หมายถึง ตระกูลโพรโตคอลที่ใช้งาน ซึ่งอาจเป็นค่าต่อไปนี้

- **AF_UNIX** สำหรับติดต่อระหว่างโพรเซสที่อยู่บนเครื่องคอมพิวเตอร์เดียวกัน (file system socket)
- **AF_INET** สำหรับระบบเน็ตเวิร์กของยนิกซ์ (Unix network socket)
- **AF_ISO** สำหรับโพรโตคอลของ ISO
- **AF_NS** โพรโตคอล Xerox Network System

โพรโตคอลที่ใช้งามมากที่สุดคือ AF_UNIX สำหรับโพรเซสที่ติดต่อบนคอมพิวเตอร์เครื่องเดียวกัน และ AF_INET ซึ่งเป็น socket สำหรับติดต่อผ่านเครือข่าย โดยเฉพาะทีซีพีไอพี (TCP/IP) และอินเตอร์เน็ต (Internet)

type หมายถึง จะกำหนดกรรมวิธีที่เกี่ยวข้องต่างๆ ในการรับส่งข้อมูลสำหรับซ็อกเก็ตนั้นๆ ดังนี้

- SOCK_STREAM
- SOCK_DGRAM

รายละเอียดได้กล่าวไว้ในหัวข้อที่ผ่านมา

protocol จะกำหนดโพรโตคอลที่ใช้งานจริงโดยทั่วไปจะใช้ค่า 0 หมายถึงเดฟฟอลต์โพรโตคอล

ฟังก์ชันซ็อกเก็ตจะส่งค่ากลับเป็นหมายเลขที่ใช้แทนซ็อกเก็ตนั้น ซึ่งสามารถที่จะใช้ฟังก์ชัน read/write เพื่ออ่านและเขียนข้อมูลไปยังซ็อกเก็ตและถ้าหากใช้งานเสร็จจะต้องใช้ฟังก์ชัน **close** เพื่อปิดบริการด้วย

3.3.4.2 การกำหนดแอดเดรสให้กับซ็อกเก็ต

แต่ละ Domain จะมีรูปแบบในการกำหนดแอดเดรสต่างกัน ดังนี้

AF_UNIX จะกำหนดไว้ในโครงสร้างข้อมูล **sockadd_un** ซึ่งกำหนดไว้ใน sys/un.h ดังนี้

```
struct sockaddr_un {
        sa_family_t             sun_family;  /* AF_UNIX */
        char                    sun_path[];  /* path name*/
};
```

sun_family จะหนดตระกูลโพรโตคอลที่ใช้งาน

sun_path จะกำหนดแอดเดรสของซ็อกเก็ต (ชื่อไฟล์)

AF_INET จะกำหนดไว้ในโครงสร้างข้อมูล **sockadd_in** ซึ่งกำหนดไว้ใน netinet/in.h ดังนี้

```
struct sockaddr_in {
        short int          sin_family;  /* AF_INET */
        unsign short int   sin_port;    /* Port Number */
        struct in_addr     sin_addr;    /* IP Address */
};
```

โครงสร้างข้อมูล in_addr เป็นข้อมูลของไอพีกำหนดโดย

```
struct in_addr {
        unsign long        ints_addr;
};
```

3.3.4.3 กำหนดซ็อกเก็ตให้กับโพรเซส (bind)

หลังจากที่สร้างซ็อกเก็ตขึ้นมาแล้ว การที่จะให้โพรเซสใช้งานซ็อกเก็ตได้นั้น จะต้องกำหนดซ็อกเก็ตนั้นให้กับโพรเซสด้วย สำหรับ AF_UNIX จะกำหนดโดยใช้ชื่อไฟล์ full path name จะไม่ขอกล่าวถึงรายละเอียดเพราะไม่ได้ใช้ในโครงงานนี้ ส่วน AF_INET จะต้องกำหนดหมายเลขพอร์ตที่จะต้องใช้งาน โดยใช้ฟังก์ชัน **bind** ดังนี้

```
#include <sys/socket.h>

int bind(int socket, const struct sockaddr *address, size_t address_len);
```

socket socket file descriptor

address กำหนดแอดเดรสให้กับซ็อกเก็ต

address_len ความยาวของโครงสร้างข้อมูลของแอดเดรส (ความยาวของแอดเดรสจะขึ้งอยู่กับ Domain ที่ใช้งาน)

ฟังก์ชันนี้จะใช้ทั้งทางด้านเซิร์ฟเวอร์และไคลเอ็นต์ คล้ายกับการบอกว่า "ข้อมูลที่ส่งมายังแอดเดรสนี้เป็นของฉัน (โพรเซส)" ซึ่งจะเป็นต้องใช้ก่อนที่จะใช้ฟังก์ชัน accept

ถ้าทำงานเสร็จจะส่งค่ากลับเป็น 0 หรือ –1 ถ้าไม่เสร็จจะส่งข้อความผิดพลาดไว้ในตัวแปร error ดังนี้

- EBADF file descriptor ไม่ถูกต้อง
- ENOTSOCK file descriptor ไม่อ้างอิงถึงซ็อกเก็ต
- EINVAL file descriptor อ้างอิงถึงซ็อกเก็ตที่ใช้งานไปแล้ว
- EADDRNOTAVAIL แอดเดรสไม่ถูกต้อง
- EADDRINUSE แอดเดรสนั้นใช้งานไปแล้ว

3.3.4.4 สร้างซ็อกเก็ตคิว (listen)

เพื่อรอรับข้อมูลการเชื่อมต่อของไคลเอ็นต์ เซิร์ฟเวอร์จะต้องสร้างคิว (queue) โดยใช้ฟังก์ชันระดับต่ำ **listen**

```
#include <sys/socket.h>

int listen(int socket, int backlog);
```

ในบางระบบจะจำกัดจำนวนไคลเอ็นต์ในการติดต่อกับเซิร์ฟเวอร์บนคิวโดยกำหนดความจุคิวไว้ใน backlog โดยถ้าคิวถูกใช้งานจะหมด จะขอเชื่อมต่อครั้งต่อไปจะทำไม่ได้ หลักการนี้จะทำให้เซิร์ฟเวอร์ทำงานกับไคลเอ็นต์ในจำนวนที่เหมาะสม โดยปรกติจะกำหนดค่า backlog ไว้ที่ 5 (ค่าสูงสุด)

โดยฟังก์ชัน listen จะส่งค่ากลับเป็น 0 ถ้าทำงานเสร็จหรือ –1 ถ้า error เกิดขึ้น เช่น EBADF, EINVAL และ ENOTSOCK

3.3.4.5 ยอมรับการเชื่อมต่อ (accept)

หลังจากที่เซิร์ฟเวอร์ไพรเซสได้รันฟังก์ชัน listen แล้วจะต้องกำหนดให้เซิร์ฟเวอร์โพรเซสรอรับการติดต่อจากไคลเอ็นต์ โดยใช้ฟังก์ชัน **accept** ดังนี้

```
#include <sys/socket.h>

int accept(int socket, const struct sockaddr *address, size_t address_len);
```

socket	กำหนดเซิร์เวอร์ซ็อกเก็ต file descriptor ที่รอรับการติดต่อจากไคลเอ็นต์ จะส่งค่ากลับเมื่อ มีไคลเอ็นต์ติดต่อเข้ามา, ฟังก์ชันนี้จะสร้าง socket ใหม่เพื่อติดต่อกับไคลเอ็นต์นั้นโดยเฉพาะ ซึ่งจะมีลักษณะเหมือนกับ socket ที่รอรับการติดต่อในตอนแรก (แต่มีข้อมูลบางอย่างเพิ่มเข้ามา) และโพรเซสต้นฉบับก็จะ fork สร้างโพรเซสลูกเพื่อทำงานกับไคลเอ็นต์ไปรเซสนั้นๆ และโพรเซสต้นฉบับเองก็จะรอการติดต่อจากไคลเอ็นต์ ต่อไป
address	แอดเดรสของไคลเอ็นต์จะถูกนำไปเก็บไว้ในโครงสร้างข้อมูล sockaddr ที่ใช้โดยพอยน์เตอร์ (ถ้าไม่ต้องการจะนำแอดเดรสของไคลเอ็นต์ไปใช้งาน จะกำหนดเป็น null pointer ก็ได้)
address_len	จะระบุความยาวของข้อมูลใน sockaddr, ในการส่งค่ากลับ address_len จะเก็บความยาวจริงของแอดเดรสไคลเอ็นต์ซึ่งจะขึ้นอยู่กับ Domain เช่น AF_INET ก็จะเก็บเป็น 16 ไบต์ และ AF_UNIX ก็จะต่างกัน

ถ้าไม่มีข้อมูลของไคลเอ็นต์ส่งเข้ามายังคิว (queue) ฟังก์ชัน accept จะถูกบล็อกจะกระทั่งมีไคลเอ็นต์ติดต่อเข้ามา ถ้าหากใช้แฟลก O_NONBLOCK กับฟังก์ชัน fcntl ก็จะเปลี่ยนการทำงานนี้ได้ การใช้ดังนี้

```
#include <sys/socket.h>

int accept(int socket, const struct sockaddr *address, size_t address_len);
```

ฟังก์ชั่ง accept จะส่งค่ากลับเป็น 3 อย่างด้วยกันคือ socket file descriptor ที่สร้างขึ้นมาใหม่เมื่อมีไคลเอ็นต์ติดต่อเข้ามา หรือ –1 ถ้าเกิด error โดย error ที่เป็นไปได้จะเหมือนกับ bind และ list แต่จะมีเพิ่ม EWOUDBLOCK เข้ามาในกรณีที่มีการใช้งาน O_NONBLOCK และไม่มีการติดต่อเข้าไปหรือ EINTR ถ้ามีการอินเตอร์รัพโพรเซส

นอกจากนี้ยังมีแอดเดรสของไคลเอ็นต์โพรเซส และความหมายของแอดเดรสที่รับได้จริง

socket ที่ส่งค่ากลับจากฟังก์ชัน accept (มีโพรเซสติดต่อเข้ามา) จะมีข้อมูลสำหรับอ้างอิงถึงองค์ประกอบการติดต่อ ครบถ้วน (5 tuples)

3.3.4.6 ติดต่อกับเซิร์ฟเวอร์โพรเซส (connect)

โปรแกรมไคลเอ็นต์สามารถติดต่อไปยังเซิร์ฟเวอร์ได้ โดยใช้งานซ็อกเก็ตที่ยังไม่ได้กำหนดให้กับโพรเซส (ส่งค่ากลับจากฟังก์ชันระดับต่ำ socket) ติดต่อไปยังเซิร์เวอร์ซ็อกเก็ตด้วยฟังก์ชัน **connect**

```
#include <sys/socket.h>

int connect(int socket, const struct sockaddr *address, size_t address_len);
```

ถ้าทำงานเสร็จจะส่งค่ากลับเป็น 0 หรือ –1 ถ้าเกิด error ซึ่งจะเพิ่ม error ต่อไปนี้จาก bind error

- EBADF socket file descriptor ไม่ถูกต้อง
- EALREADDY socket ถูกใช้งานไปแล้ว
- ETIMEDOUT ใช้เวลาการติดต่อเกินกว่าที่กำหนด
- ECONNREFUSED เซิร์ฟเวอร์ปฏิเสธการขอเชื่อมต่อ

3.3.4.7 ยกเลิกการใช้งาน Socket (close)

โดยการใช้ฟังก์ชัน close ร่วมกับ socket file descriptor และในการยกเลิกการใช้งาน socket ควรจะยกเลิกการทำงานทั้งที่ไคลเอ็นต์และเซิร์ฟเวอร์ ในบางกรณีฟังก์ชัน close จะถูกบล็อกถ้า socket มีข้อมูลที่ยังไม่ได้ส่งไป โดยเฉพาะถ้าเป็นโพรโตคอล TCP แต่ถ้ามีการใช้ออปชัน SOCK_LINGER เป็นกำหนดการจัดการกับข้อมูลที่ยังไม่ส่งก็ได้

ตัวอย่างการสร้างโปรแกรมเซิร์ฟเวอร์และไคลเอ็นต์สามารถดูได้ในซอร์สโค้ด source code ของ IsagMQ และ Client Demo

3.4 ระบบและการเขียนโปรแกรมที่ความปลอดภัย

3.4.1 ความปลอดภัยของระบบ

3.4.1.1 Confidentiality หมายถึงการที่จะสามารถเข้ามาใช้โปรแกรมของเราได้จะต้องเป็นบุคคลที่ได้รับอนุญาติเท่านั้น (keeping secret)

3.4.1.2 Integrity หมายถึงโปรแกรมของเราสามารถถูกเปลี่ยนแปลงได้ด้วยบุคคลที่ได้รับอนุญาติในวิถีทางที่เราอนุญาติไว้เท่านั้น (verifying information)

3.4.1.3 Availability หมายถึง บุคคลที่ได้รับอนุณาติสามารถมาใช้โปรแกรมของเราได้อย่างสดวกและไม่ติดขัดอะไร (นั่นคือเป็นไปตามหลักการของ system requirement)

3.4.2 การเขียนโปรแกรมอย่างปลอดภัย

3.4.2.1 การเขียนภาษาซีให้ได้มาตรฐาน

การเขียนภาษาซีให้ได้มาตรฐานสามารถศึกษาได้จากคู่มือมาครฐานภาษาซีทั่วไป มาตรฐานภาษาซีมีจุดสำคัญหลักๆ ดังนี้

1. การเว้นช่องไฟที่เหมาะสม และ คอมเม็นต์ (comment) โดยที่ทำให้มองเห็นโครงสร้างของโปรแกรมจาก layout ตัวโปรแกรม การใช้ expression แบบง่ายๆคำสั่ง และฟังก์ชัน ต่างๆเพื่อให้เข้าใจง่ายที่สุด
2. ระลึกเสมอว่า ใครสักคน หรืออาจจะเป็นตัวเราเอง ต้องถูกเรียกให้กลับมานั่งดัดแปลงโปรแกรม หรือ ทำให้ใช้งานได้บนฮาร์ดแวร์อื่นในอนาคต พยายามจัดสร้างโปรแกรมที่พอร์ตได้ง่าย, รวมกลุ่ม(localize) และ ออปติไมเซชัน (optimization) ได้สะดวก เพราะสิ่งเหล่านี้มักจะเป็นหอกข้างแคร่ (pessimizations) ในการพอร์ต ของเครื่องหรือฮาร์ดแวร์ต่างชนิดกัน
3. การเลือกใช้รูปแบบนั้นขึ้นอยู่กับผู้ใช้งาน ควรมีรูปแบบที่เป็นไปตามระหว่างทีมงานในกลุ่ม ดีกว่าเดินตามกฎเกณฑ์ตายตัว การผสมผสานรูปแบบหลายๆชนิดเข้าด้วยกัน เป็นสิ่งที่แย่ยิ่งกว่าการใช้รูปแบบที่ไม่ดีเพียงชนิดเดียว เพราะอย่างน้อยก็สับสนน้อยกว่า

3.4.2.2 Buffer Over Flow และการป้องกัน

Buffer Over Flow เป็นรูปแบบการโจมตีที่สำคัญรูปแบบหนึ่งที่จำเป็นต่อการพิจารณา ในการสร้างความปลอดภัยแก่ระบบใดๆ

Buffer Over Flow

วิธีที่ง่ายที่สุดในการโจมตีแบบ Buffer Over Flow เรียกว่า สแตกสแมชิง (stack smashing) ซึ่งคือการเข้าไปเปลี่ยนแปลงค่าของรีเทินแอดเดรส (Return Address) ภายใน สแตก (stack) เมื่อฟังก์ชันทำการ Return ค่า แทนที่มันจะทำการกระโดดไปยัง ตำแหน่งที่เรียกใช้งานมัน มันจะรีเทินไปยัง สแตกที่ผู้บุกรุกต้องการ ซึ่งทำให้ผู้บุกรุกสามารถทำการเอ็กซิคยูทซอร์สโค้ด (execute source code) นั้นได้ โปรแกรมที่เขียนด้วยภาษาซีจะถูกทำการ สแตกสแมชิง ได้ง่าย ภาษาซีจะยอมให้มีการจัดการหน่วยความจำ และ พอยน์เตอร์ (pointer) ได้โดยตรงโดยไม่มีการตรวจสอบขอบเขตของหน่วยความจำ และที่ร้ายแรงกว่านั้นก็คือไลบรารีของภาษาซีจะมีหลายๆฟังก์ชันที่ไม่มีความปลอดภัย เช่น gets ที่ทำการเขียนอินพุตของยูเซอร์โดยไม่มีขอบเขตไปยังบัฟเฟอร์ที่มีค่าคงที่โดยปราศจากการตรวจสอบขอบเขตของหน่วยความจำ บัฟเฟอร์จะถูกเก็บไว้บนสแตกซึ่งบ่อยครั้งที่มันจะถูกส่งมาด้วยฟังก์ชันนี้ ในการทำการเอ็กซ์พลอยต์ (exploit) กระบวนการดังกล่าวผู้บุกรุกเพียงแค่ใส่ค่าอินพุตที่มีจำนวนมากกว่าขนาดของบัฟเฟอร์และเข้ารหัสโปรแกรมที่ใช้สำหรับบุกรุกภายในอินพุตนั้น วิธีการที่ซับซ้อนขึ้นในการโจมตีแบบ Buffer Over Flow คือจะทำการเอ็กซ์พลอยต์ไปยังบัฟเฟอร์ที่ไม่มีความปลอด

ภัยที่ถูกใช้บนฮีป (heap) ซึ่งทำได้ยากเพราะโปรแกรมส่วนใหญ่จะไม่ได้ทำการกระโดดไปยังตำแหน่งที่โหลดค่ามาจากฮีปหรือซอร์สโค้ดที่ถูกเก็บไว้ในฮีป

มีหลายๆแนวทางในการป้องกันการโจมตีแบบ Buffer Over Flow StackGuard เป็นคอมไพล์เลอร์ที่จะทำการสร้างไลบรารีไพล์ (binary file) ที่มีการรวมโค้ดที่ถูกออกแบบมาเพื่อป้องกันการโจมตีแบบสแตกสแมชิงมันจะทำการใส่ค่าพิเศษค่าหนึ่งบนสแตกต่อจากค่ารีเทินแอดเดรสและจะทำการตรวจสอบว่าไม่มีความวุ่นวายเกิดขึ้นก่อนทำการกระโดดค่าออกไป วิธีการอื่นคือทำการสร้างไลบรารีที่มีความปลอดภัยขึ้นมาใหม่แทนที่ไลบรารีที่ไม่มีความปลอดภัย, ทำการตรวจสอบค่าของรีเทินแอดเดรสบนสแตกก่อนที่จะทำการกระโดด ค่าออกไป SFI (Software Fault Isolation) เป็นเทคนิคที่ทำการใส่ค่า bit mask ก่อนการทำงานใดๆบนหน่วยความจำเพื่อป้องกันการทำให้หน่วยความจำเกินขนาดและมีวิธีการอื่นๆอีกมากมาย

การโจมตีแบบ Buffer Over Flow จะทำได้ยากมากขึ้น เมื่อทำให้ระบบปฏิบัติการทำการแยกโค้ด และดาตาเซกเม็นต์ (data segment) ออกจากกัน ซึ่ง โค้ดเซกเม็นต์ (code segment) จะเป็นแค่ รีดโอนลี(Read-only) และคำสั่งไม่สามารถทำการเอ็กซิคยูท (execute) จากดาตาเซกเม็นต์ได้ แต่ก็ไม่ได้เป็นการกำจัดปัญหาของ Buffer Over Flow เพราะอย่างไรก็ตามเมื่อผู้บุกรุกยังคงสามารถเปลี่ยนแปลงค่าตำแหน่งที่ถูกเก็บไว้บนสแตกเพื่อทำให้โปรแกรมกระโดด ไปยังจุดใดๆในโค้ดเซกเม็นต์ได้เหมือนเดิม สำหรับโปรแกรมที่เป็น Shared library มันมีความเป็นไปได้สำหรับผู้บุกรุกในการกระโดดไปยังตำแหน่งโค้ดเซกเม็นต์ที่จะสามารถทำให้เกิดความเสียหายได้เช่น การเรียกไปยังระบบผู้พัฒนาทำการตัดสินใจไม่ใช้วิธีนี้ในลินุกซ์เคอร์เนล (Linux kernel) แต่มันก็ยังไม่สามารถแก้ปัญหาที่แท้จริงของการทำ Buffer Over Flow ได้

เมื่อมองในเรื่องของความสะดวกในการใช้ประโยชน์ (Availability) ในการแก้ปัญหา Buffer Over Flow ด้วยวิธีการต่างๆในช่วงของรันไทม์ (run-time) ในการแก้ปัญหา Buffer Over Flow ในช่วงของรันไทม์จะทำให้ประสิทธิภาพในการทำลดต่ำลง เช่นการใช้ StackGuard จะทำให้เกิด overhead สูงถึง 40 % ปัญหาอื่นๆเกี่ยวกับการแก้ปัญหาในช่วงรันไทม์นั่นคือขณะที่เราสามารถตรวจสอบป้องกันการเกิด Buffer Over Flow ได้แต่ก็ส่งผลทำให้เกิดการโจมตีรูปแบบใหม่ที่เรียกว่า **DOS (Denial – Of – Service)**

บทที่ 4
เอสเอสแอล, โอเพนเอสเอสแอล และ ระบบการรับรองสิทธิ์ผู้ใช้
(SSL, OpenSSL and Certificate Authority (CA))

4.1 SSL/TLS

SSL โพรโตคอลเป็นโพรโตคอลที่ได้รับการออกแบบมาเพื่อทำให้เกิดการสื่อสารอย่างปลอดภัยขึ้น โดยโพรโตคอลสแต็กของ SSL จะอยู่ตรงกลางระหว่างชั้นแอพพลิเคชันเลเยอร์ (Application Layer) กับชั้นทรานสปอร์ทเลเยอร์ (Transport Layer) นั่นคือ ชั้น SSL Layer จะเป็นชั้นที่เอาไว้ใช้สำหรับการสื่อสารที่ปลอดภัย ข้อดีของมันคือเราสามารถพัฒนาแอพพลิเคชันของได้อย่างเต็มที่โดยไม่ต้องสนใจความปลอดภัย เนื่องจากการทำให้เกิดความปลอดภัยจะเป็นภาระหน้าที่ของชั้น SSL โดย SSL จะอาศัยทีซีพีโพรโตคอลในการรับส่งข้อมูล เพราะการส่งข้อมูลของ SSL ข้อมูลที่ถูกส่งไปนั้นเป็นข้อมูลที่ถูกเข้ารหัส ถ้าเกิดข้อมูลส่วนใดเกิดสูญหายขึ้นมาในระหว่างการส่งข้อมูลจะทำให้การถอดรหัสข้อมูลจะได้ข้อมูลที่ไม่ถูกต้องทีซีพีจะเป็นตัวที่ช่วยรับประกันว่าข้อมูลที่ถูกส่งไปผู้รับจะต้องได้รับอย่างครบถ้วน ดังนั้น SSL สามารถที่พัฒนาโพรโตคอลที่จะทำให้เกิดความปลอดภัยที่สูงขึ้นได้โดยไม่ต้องไปกังวลกับความถูกต้องในการส่งข้อมูล

4.1.1 บทบาทของ SSL (SSL Roie)

SSL โพรโตคอลจะประกอบไปด้วยเซตของข้อความ (Message) และ บทบาท (Role) ต่างๆที่เกี่ยวข้องเมื่อมีการส่งหรือรับข้อมูลซึ่งกันและกัน

SSL จะถูกกำหนดให้มีสองบทบาทสำหรับกระบวนการติดต่อสื่อสารกันระหว่างสองระบบ โดยบทบาทของระบบแรกจะต้องมีบทบาทเป็นไคลเอ็นต์ และอีกระบบจะต้องมีบทบาทเป็นเซิร์ฟเวอร์ การแยกความแตกต่างนี้มีความสำคัญเพราะว่า SSL จะปฏิบัติต่อบทบาททั้งสองนี้อย่างแตกต่างกันโดยชิ้นเชิง ไคลเอ็นต์จะเป็นผู้เริ่มต้นการติดต่อสื่อสารอย่างปลอดภัยขึ้นมา จากนั้นเซิร์ฟเวอร์จะเป็นผู้ตอบสนองความต้องการของไคลเอ็นต์ ภาพที่เห็นได้ชัดของบทบาททั้งสองนี้คือ เวบบราวเซอร์ (Web Browser) จะมีบทบาทเป็นไคลเอ็นต์ เวบเซิร์ฟเวอร์ (Web Server) จะมีบทบาทเป็นเซิร์ฟเวอร์ นั่นคือเมื่อจะนำ SSL ไปใช้กับแอพพลิเคชันใดๆ จะต้องคำนึงถึงความแตกต่างของระบบทั้งให้ชัดเจน

เหตุที่ SSL ต้องแยกความแตกต่างนี้เนื่องจากกระบวนการของ SSL จะต้องมีการทำการต่อรองค่าพารามิเตอร์ต่างๆในการสร้างการติดต่อสื่อสารที่ปลอดภัยระหว่างกัน

ไคลเอ็นต์จะเป็นผู้เริ่มการติดต่อสื่อสาร ซึ่งจะเป็นการนำเสนอค่าพารามิเตอร์ต่างๆที่ตัวเองสามารถรองรับได้ส่งไปให้แก่เซิร์ฟเวอร์ และเซิร์ฟเวอร์จะเป็นผู้ทำการตัดสินใจขั้นสุดท้ายว่าทั้งสองระบบนี้จะใช้พารามิเตอร์ตัวใดในการสร้างการติดต่อสื่อสารที่ปลอดภัย ถึงแม้ว่าการตัดสินใจขั้นสุดท้ายจะอยู่ที่ฝั่ง

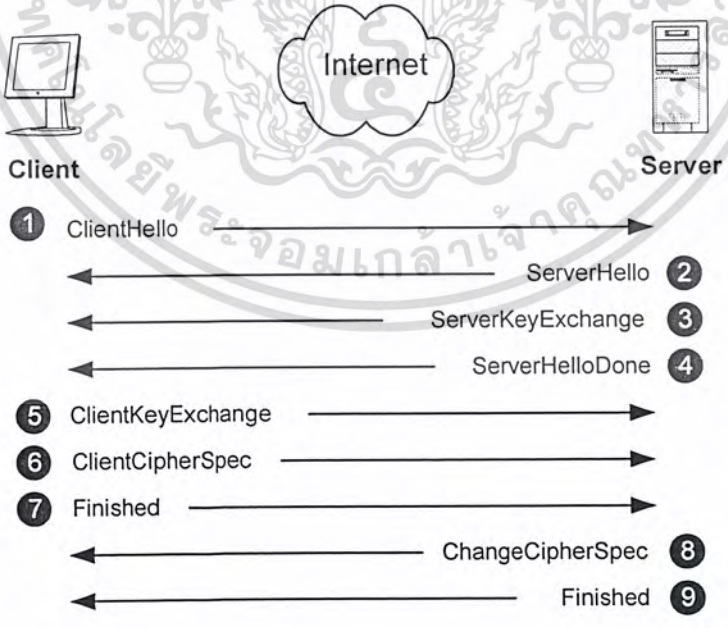
เซิร์ฟเวอร์ แต่มีข้อจำกัดอยู่ว่าเซิร์ฟเวอร์จะสามารถเลือกค่าพารามิเตอร์ต่างได้จากค่าพารามิเตอร์ที่ไคลเอ็นต์ได้ส่งมาให้แล้วเท่านั้น

4.1.2 ข้อความของ SSL (SSL Message)

เมื่อ SSL ไคลเอ็นต์และเซิร์ฟเวอร์ทำการติดต่อสื่อสารกัน กระบวนการภายในการติดต่อสื่อสารกันนั้นจะเป็นการส่งข้อความของ SSL (SSL Message) ระหว่างกัน ซึ่งข้อความดังกล่าวจะมีความหมายที่แตกต่างกันไปตามเฟสในการสร้างการติดต่อสื่อสารที่ปลอดภัยโดยใช้ SSLโพรโตคอล เช่น Alert เป็นข้อความที่ทำให้ทราบว่าการติดต่อสื่อสารล้มเหลวหรือเกิดการผิดกฎความปลอดภัยของ SSL, ApplicationData เป็นข้อมูลจริง (Plaintext) ที่ต้องการทำการส่งให้กันซึ่งมันจะต้องถูกทำการเข้ารหัส การระบุตัวตนของผู้ส่งหรือรับและตรวจสอบความถูกต้องของข้อมูลโดยใช้กระบวนการของ SSL โพรโตคอล ฯลฯ

4.1.3 การสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูล

กระบวนการพื้นฐานในการสร้างการเชื่อมต่อที่ปลอดภัยโดยใช้ SSL โพรโตคอล คือการใช้การเข้ารหัสข้อมูลโดยไคลเอ็นต์จะรับกุญแจสาธารณะ (Public key) ที่เซิร์ฟเวอร์ได้ส่งมาให้แล้วทำการสร้างกุญแจลับ (Secret key) นำกุญแจลับที่ได้มาเข้ารหัสด้วยกุญแจสาธารณะ ที่ได้รับมา จากนั้นส่งกับไปให้แก่เซิร์ฟเวอร์และใช้กุญแจลับนี้ในการเข้ารหัสข้อมูลที่จะทำการส่งและถอดรหัสข้อมูลที่ได้รับ ซึ่งกระบวนดังกล่าวจะเป็นการส่งข้อความของ SSL ตามเฟสต่างๆดังรูปที่ 4-1 ซึ่งสามารถอธิบายได้ดังนี้



รูปที่ 4-1 SSLจะใช้ 9 ข้อความในการสร้างการเชื่อมต่อที่มีการเข้ารหัสข้อมูล

1. ClientHello

ClientHello จะเป็นข้อความที่ไคลเอ็นต์ใช้สำหรับเริ่มต้นการสื่อสารด้วย SSL โพรโตคอลโดยข้อความนี้จะเป็นการนำเสนอค่าพารามิเตอร์ต่างๆที่ตัวเองสามารถรองรับได้ส่งไปให้แก่เซิร์ฟเวอร์ โดยองค์ประกอบที่สำคัญของ ClientHello มีดังนี้

- **Version** จะเป็นตัวที่บอกให้ทราบว่าเวอร์ชันสูงสุดของ SSL โพรโตคอลที่ไคลเอ็นต์สามารถรองรับได้ ซึ่ง SSL เวอร์ชันสาม เป็นเวอร์ชันที่นิยมนำไปใช้กันอย่างกว้างขวางในระบบอินเตอร์เน็ต แต่ในมุมมองของเซิร์ฟเวอร์จะมองว่าไคลเอ็นต์สามารถรองรับได้ทุกเวอร์ชัน เช่น ไคลเอ็นต์ส่ง ClientHello มาโดยระบุเป็น SSL เวอร์ชันสาม แต่เซิร์ฟเวอร์สามารถรองรับได้สูงสุดเพียงเวอร์ชันสอง เซิร์ฟเวอร์จะเลือกใช้เวอร์ชันสอง ในกรณีนี้ไคลเอ็นต์สามารถที่จะทำการติดต่อสื่อสารกับเซิร์ฟเวอร์ต่อไปโดยใช้ SSL เวอร์ชันสอง หรือจะยกเลิกการติดต่อสื่อสารกับเซิร์ฟเวอร์ไปเลยก็ได้

-**RandomNumber** ตัวเลขสุ่มโดยฟิลด์นี้จะมีค่า 32 ไบต์ โดย 4 ไบต์แรกจะเป็นค่าของวันและเวลา เหตุที่ต้องมีการระบุวันเวลาเพราะว่าเพื่อเป็นการสร้างความมั่นใจว่า ในการติดต่อสื่อสารหนึ่งครั้งจะไม่มีไคล์เอ็นต์คนใดที่จะใช้ตัวเลขสุ่มตัวเดียวกันถึงสองครั้ง นั่นคือมันสามารถที่จะแก้ไขปัญหาของการทำการโจมตีแบบ แอ็กชันรีเพลย์ (Action Replay) ได้ โดยค่าที่เหลืออีก 28 ไบต์จะเป็นตัวเลขสุ่มซึ่งจำเป็นต้องสร้างมาจากฟังก์ชั่นที่สามารถสร้างตัวเลขสุ่มได้อย่างปลอดภัย

-**SessionID** ในการติดต่อสื่อสารระหว่างไคลเอ็นต์และเซิร์ฟเวอร์จะเรียกสั้นๆว่า เซสชั่น (Session) ถ้าเซิร์ฟเวอร์ทำการเก็บเซสชั่นของไคลเอ็นต์เอาไว้ เมื่อไคลเอ็นต์ทำการติดต่อเข้ามาใหม่เซิร์ฟเวอร์จะสามารถตรวจสอบได้จากเซสชั่นที่ได้เก็บไว้ทำให้การติดต่อสื่อสารมีความรวดเร็วขึ้นแต่การใช้เซสชั่นใน SSL นั้นมีความยุ่งยากและซับซ้อน โดยปกติแล้วฟิลด์นี้จะไม่มีความหมายใดๆเมื่อเราใช้การติดต่อ SSL แบบปกติ

-**CipherSuites** จะเป็นฟิลด์ที่ไคลเอ็นต์จะทำการใส่ลิสท์ของอัลกอริทึมในการเข้ารหัสถอดรหัสที่ ไคลเอ็นต์สามารถรองรับได้

2. ServerHello

เมื่อเซิร์ฟเวอร์ได้รับข้อความ ClientHello เซิร์ฟเวอร์จะทำการตอบกลับด้วย ServerHello ไคลเอ็นต์จะเป็นผู้นำเสนอค่าพารามิเตอร์ต่างๆให้แก่เซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำการเลือกค่าพารามิเตอร์เหล่านั้นแล้วส่งกลับไปด้วยข้อความ ServerHello โดยฟิลด์ของ ServerHello จะมีดังนี้

-**Version** จะเป็นฟิลด์ที่ระบุว่าการติดต่อสื่อสารระหว่างไคลเอ็นต์และเซิร์ฟเวอร์จะใช้ SSL เวอร์ชันใด

-RandomNumber เซิร์ฟเวอร์จะรับตัวเลขสุ่มจากไคลเอ็นต์และนำค่าตัวเลขนั้นมาสุ่มอีกครั้งด้วยฟังก์ชันเดียวกับไคลเอ็นต์ ซึ่งไคลเอ็นต์จะใช้ตัวเลขสุ่มนี้ในการสร้างกุญแจลับ ซึ่งจะเป็นเซสชั่นคีย์ที่ใช้ในเซสชั่นของไคลเอ็นต์และเซิร์ฟเวอร์

-SessionID จะไม่ใช่ค่าเดียวกันกับ SessionID ของไคลเอ็นต์ ซึ่งค่านี้จะเป็นค่าที่ใช้ในการระบุถึงไคลเอ็นต์ใดๆที่เชื่อมต่อเข้ามายังเซิร์ฟเวอร์

-CipherSuite ในฝั่งของไคลเอ็นต์ค่านี้จะอยู่ในรูปของพหูพจน์ เซิร์ฟเวอร์จะทำการเลือกค่าและค่าที่ได้จะอยู่ในรูปของเอกพจน์

3. ServerKeyExchange

เมื่อเซิร์ฟเวอร์ทำการส่งข้อความ ServerHello ไปแล้ว เฟสต่อไปเซิร์ฟเวอร์จะทำการส่งข้อความ SeverKeyExchange ไปให้แก่ฝั่งไคลเอ็นต์ ในขณะที่ฟิลด์ CipherSuite จะเป็นตัวที่บอกถึงชนิดของอัลกอริทึมและขนาดของคีย์ ข้อความ ServerKeyExchange จะเป็นค่าของกุญแจสาธารณะของเซิร์ฟเวอร์ ที่จะส่งไปให้ไคลเอ็นต์ใช้สำหรับการเข้ารหัสกุญแจลับ ซึ่งกุญแจสาธารณะนี้จะไม่มีการเข้ารหัสใดๆเพราะกุญแจสาธารณะมีความปลอดภัยในตัวของมันเองอยู่แล้ว

4. ServerHelloDone

จะเป็นข้อความที่บอกให้ไคลเอ็นต์ทราบว่าเซิร์ฟเวอร์ได้ส่งข้อมูลที่ใช้สำหรับการเริ่มต้นการติดต่อสื่อสารเรียบร้อยแล้วซึ่งข้อความนี้จะไม่มีข้อมูลใดๆ แต่มันเป็นสิ่งสำคัญต่อไคลเอ็นต์เพราะไคลเอ็นต์จะต้องได้รับข้อความนี้ก่อน ไคลเอ็นต์จึงจะสามารถกระทำเฟสต่อไปของการสร้างการเชื่อมต่อที่ปลอดภัยโดยใช้ SSL โพรโตคอลได้

5. ClientKeyExchange

เมื่อเซิร์ฟเวอร์สิ้นสุดการต่อรองค่าพารามิเตอร์ต่างๆที่ใช้ในการสื่อสารด้วย SSL โพรโตคอลแล้ว ไคลเอ็นต์จะตอบกลับด้วยข้อความ ClientKeyExchange ซึ่งคือการส่งกุญแจลับที่ได้เข้ารหัสด้วยกุญแจสาธารณะที่ได้รับมาจากเซิร์ฟเวอร์

เข้ารหัสถอดรหัสข้อมูลแบบกุญแจเดี่ยว (Symmetric key) จะใช้กุญแจตัวเดียวในการเข้ารหัสถอดรหัสข้อมูล ดังนั้นจึงไม่สามารถส่งกุญแจนั้นผ่านระบบเน็ตเวิร์คได้ SSL โพรโตคอลทำการแก้ปัญหาโดยให้เซิร์ฟเวอร์ส่งกุญแจสาธารณะมาให้แก่ไคลเอ็นต์ ไคลเอ็นต์จะทำการสร้างเซสชั่นคีย์แล้วนำเซสชั่นคีย์ที่ได้มาเข้ารหัสด้วยกุญแจสาธารณะที่ได้รับมาจากเซิร์ฟเวอร์ เมื่อส่งข้อมูลนี้ผ่านระบบเน็ตเวิร์คจะมีเพียงแค่เซิร์ฟเวอร์ที่ส่งกุญแจสาธารณะมาให้เท่านั้นที่จะสามารถทำการถอดรหัสข้อมูลได้ ซึ่งเป็นการทำให้ไคลเอ็นต์มั่นใจว่ามี

เพียงแค่เซิร์ฟเวอร์ที่เป็นเจ้าของกุญแจสาธารณะเท่านั้นที่จะสามารถถอดรหัสข้อมูลได้ ทำให้เกิดความปลอดภัยขึ้นมาได้ในระดับหนึ่ง

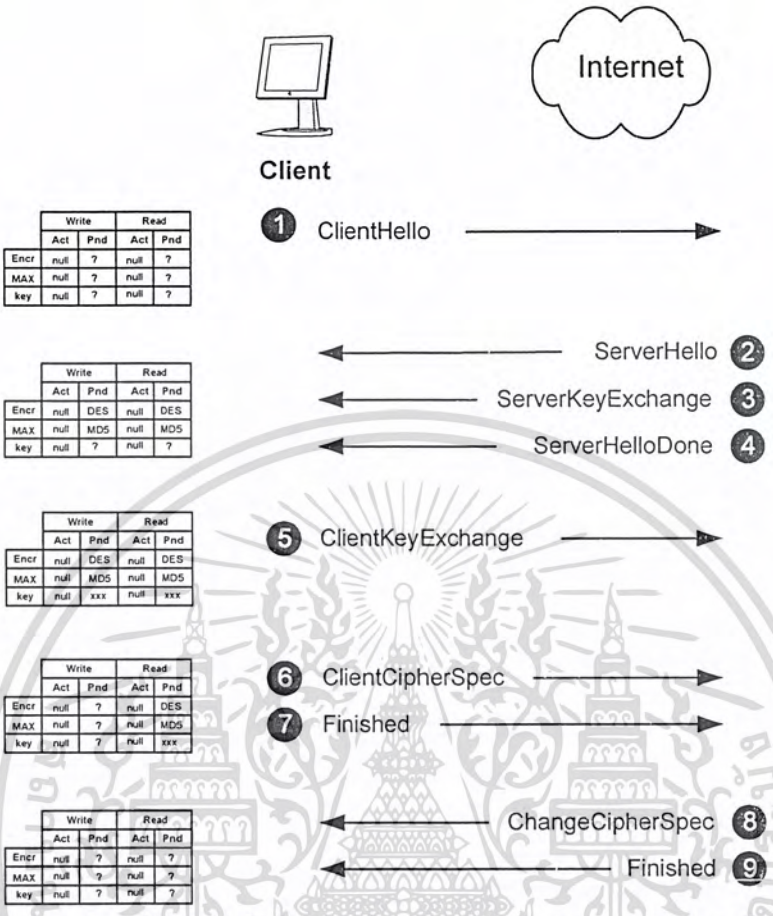
6. ChangeCipherSpec

หลังจากที่ไคลเอ็นต์ส่งข้อความ ClientKeyExchange เรียบร้อยแล้วจะถือว่าสิ้นสุดการต่อรองค่าพารามิเตอร์ต่างๆระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ ณ จุดนี้ทั้งสองระบบสามารถที่จะสื่อสารกันได้อย่างปลอดภัยโดยใช้ SSL โพรโตคอล ข้อความ ChangeCipherSpec จะเป็นข้อความที่ชี้ให้เห็นอย่างชัดเจนว่าการรับส่งข้อมูลอย่างปลอดภัยโดยใช้ SSL โพรโตคอลสามารถทำงานได้แล้ว

กระบวนการในการรับส่งข้อมูลของ SSL โพรโตคอล ข้อมูลที่ทำการส่งไปจะต้องประกอบไปด้วยชนิดของอัลกอริทึมที่ใช้ในการเข้ารหัส, ข้อมูลที่ใช้ตรวจสอบความถูกต้องของข้อมูล (MIC), กุญแจที่ได้มาจากอัลกอริทึมดังกล่าว SSL โพรโตคอลจะมีการตรวจสอบข้อมูลหลายๆอย่างโดยเฉพาะอย่างยิ่ง เซสชั่นคีย์ โดยการตรวจสอบทิศทางระหว่างทิศทางในการรับและการส่งต้องมีความแตกต่างกัน นั่นคือ เซตของคีย์หนึ่งจะเป็นตัวที่ใช้รักษาความปลอดภัยสำหรับไคลเอ็นต์ในการส่งข้อมูล และเซตของอีกคีย์หนึ่งจะเป็นตัวรักษาความปลอดภัยสำหรับไคลเอ็นต์ในการรับข้อมูล (ถ้ามีการแยกความแตกต่างของอัลกอริทึมที่ใช้ทั้งในการรับและการส่งข้อมูลก็จะเป็นวิธีที่ดีแต่ SSL โพรโตคอลไม่มีตัวเลือกนี้อยู่)

ทั้งในฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์ SSL โพรโตคอลจะกำหนดให้มีสภาวะในการรับข้อมูลเรียกว่า read state และการส่งข้อมูล เรียกว่า write state จากรูปจะเห็นว่า read และ write state จะถูกแบ่งออกเป็นอีกสองสภาวะคือ active และ pending ดั้งนั้นจะมีทั้งหมดสี่สภาวะด้วยกันคือ active write state, pending write state, active read state, pending read state active

จากรูปใช้ตัวย่อเป็น Act และ pending จากรูปใช้ตัวย่อเป็น Pnd อัลกอริทึมในการเข้ารหัสถอดรหัสใช้ตัวย่อเป็น Encr ข้อมูลที่ใช้ตรวจสอบความถูกต้องของข้อมูลใช้ตัวย่อเป็น MAC (Message Authentication Code) และคีย์จะใช้เป็น key จากรูปที่ 4-2 และ 4-3 ซึ่งสามารถอธิบายทั้งสี่สภาวะได้ดังนี้



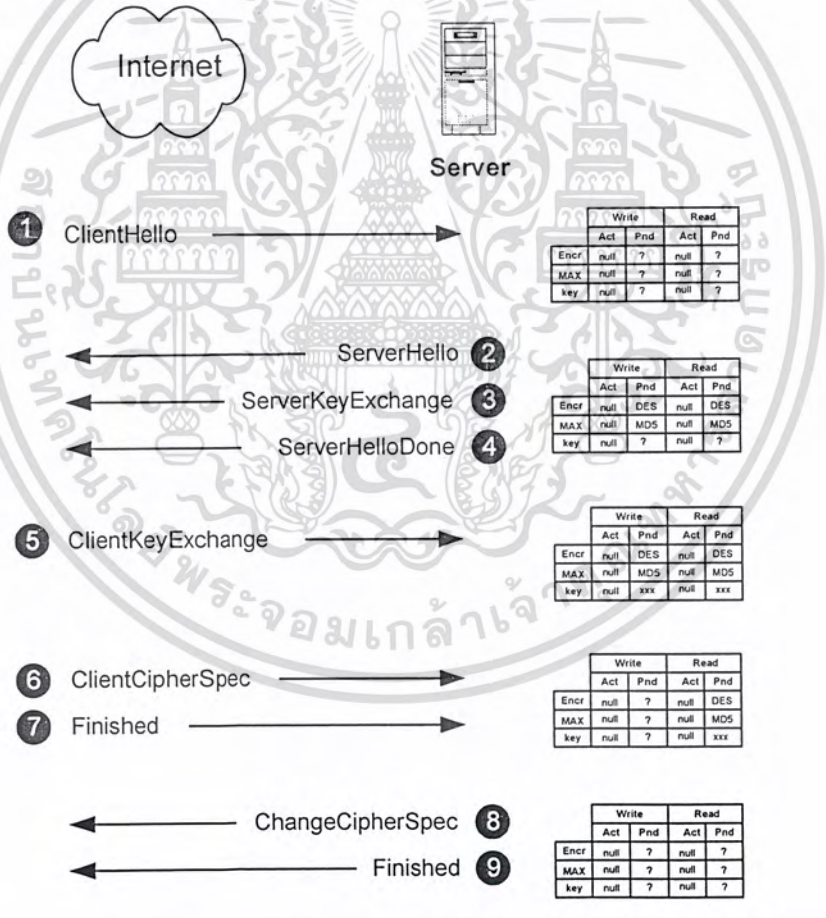
รูปที่ 4-2 แสดงสภาวะในการรับส่งข้อมูลของไคลเอ็นต์ตามกระบวนการของ SSL โพรโตคอล

ฝั่งไคลเอ็นต์

1. เมื่อไคลเอ็นต์เริ่มต้นการสื่อสารด้วย SSL โพรโตคอลโดยทำการส่งข้อความ ClientHello จะทำการเซตสภาวะ active ทั้งสองสภาวะเป็น null สภาวะ pending จะไม่มีการทำอะไรกับมัน
2. เมื่อไคลเอ็นต์ได้รับข้อความ ServerHello ในขณะนี้ไคลเอ็นต์จะทราบว่าเซิร์ฟเวอร์ทำการเลือกอัลกอริทึมใดในการติดต่อเซสชั่นนี้ ไคลเอ็นต์ทำการปรับข้อมูลของอัลกอริทึมในการเข้ารหัสถอดรหัส และ ข้อมูลที่ใช้ตรวจสอบความถูกต้องของข้อมูลในสภาวะ pending ทั้งสอง จากข้อมูลที่ได้รับมาดังรูป
5. เมื่อไคลเอ็นต์ได้สร้างเซสชั่นคีย์และส่งข้อความ ClientKeyExchange เรียบแล้วร้อย จะทำให้ทราบถึงคีย์ที่จะใช้ในการติดต่อสื่อสาร จึงทำการปรับข้อมูลของคีย์ในสภาวะ pending ทั้งสอง ดังรูป
6. เมื่อไคลเอ็นต์ส่งข้อความ ChangeCipherSpec ที่สภาวะ write จะเปลี่ยนจากสภาวะ

pending ไปเป็นสภาวะ active พร้อมกับปรับข้อมูลจากข้อมูลของสภาวะ pending และทำการรีเซตค่าในสภาวะ pending ให้ไม่มีค่าใดๆ ณ จุดนี้ไคลเอ็นต์จะสามารถทำการส่งข้อมูล (ciphertext) ที่ถูกเข้ารหัสด้วยอัลกอลิทึม DES และ ค่าตรวจสอบความถูกต้องที่ใช้อัลกอลิทึมMD5 ได้

8. เมื่อไคลเอ็นต์ได้รับข้อความ ChangeCipeherSpec จากเซิร์ฟเวอร์ ที่สภาวะ read จะเปลี่ยนจากสภาวะ pending ไปเป็นสภาวะ active พร้อมกับปรับข้อมูลจากข้อมูลของสภาวะ pending และทำการรีเซตค่าในสภาวะ pending ให้ไม่มีค่าใดๆ ณ จุดนี้ไคลเอ็นต์จะสามารถทำการส่งข้อมูล (ciphertext) ที่ถูกเข้ารหัสด้วยอัลกอลิทึม DES และค่าตรวจสอบความถูกต้องของข้อมูลที่ใช้อัลกอลิทึม MD5 ได้



รูปที่ 4-3 แสดงสภาวะในการรับส่งข้อมูลของเซิร์ฟเวอร์ตามกระบวนการของ SSL โพรโตคอล

ฝั่งเซิร์ฟเวอร์

1. เมื่อเซิร์ฟเวอร์ได้รับข้อความ ClientHello มันจะทำการเซตค่าสภาวะ active ของทั้งสองให้เป็น null และสภาวะ pending จะไม่มีการทำอะไรกับมัน
2. เมื่อเซิร์ฟเวอร์ทำการส่งข้อความ ServerHello ณ จุดนี้มันจะทราบถึงอัลกอลิทึมที่ใช้สำหรับเซสชั่นนี้และจะทำการปรับข้อมูลเหล่านี้ในสภาวะของ pending ดังรูป แต่ข้อมูลของคีย์ในสภาวะ pending ในตอนนี้จะอยู่ในสถานะไม่ทราบค่า
5. เมื่อเซิร์ฟเวอร์ได้รับข้อความ ClientKeyExchange จะทำให้มันทราบค่าของคีย์ที่จะใช้ในการติดต่อสื่อสารของเซสชั่นนี้ ดังนั้นมันจะทำการปรับข้อมูลของคีย์ในสภาวะ pending
6. เมื่อเซิร์ฟเวอร์ได้รับข้อความ ChangeCipherSpec ที่สภาวะ read มันจะทำการเปลี่ยนจากสภาวะ pending ไปเป็นสภาวะ active พร้อมกับปรับข้อมูลจากข้อมูลของ สภาวะ pending และทำการรีเซตค่าในสภาวะ pending ให้ไม่มีค่าใดๆ ณ จุดนี้เซิร์ฟเวอร์สามารถที่จะรับข้อมูล (ciphertext) ที่ถูกเข้ารหัสด้วยอัลกอลิทึม DES และค่าตรวจสอบความถูกต้องของข้อมูลที่ใช้อัลกอลิทึม MD5 ได้
8. เมื่อเซิร์ฟเวอร์ทำการส่งข้อความ ChangeCipherSpec ที่สภาวะ write จะเปลี่ยนจากสภาวะ pending ไปเป็น สภาวะ active พร้อมกับปรับข้อมูลจากข้อมูลของสภาวะ pending

   และทำการรีเซตค่าในสภาวะ pending ให้ไม่มีค่าใดๆ ณ จุดนี้เซิร์ฟเวอร์จะสามารถทำการส่งข้อมูล (ciphertext) ที่ถูกเข้ารหัสด้วยอัลกอลิทึม DES และ ค่าตรวจสอบความถูกต้องที่ใช้อัลกอลิทึม MD5 ได้

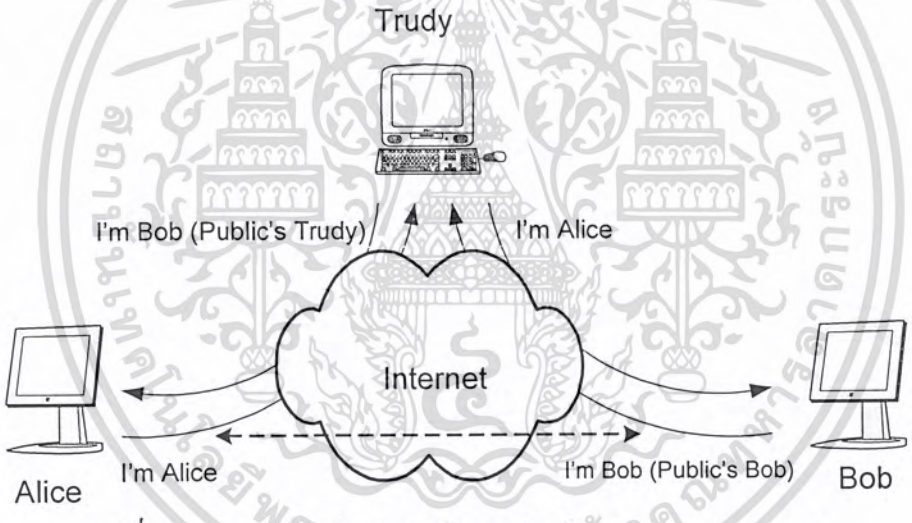
7. Finished

ทันทีหลังจากที่ได้ทำการส่งข้อความ ChangeCipherSpec แต่ละระบบจะทำการส่งข้อความ Finished ซึ่งเป็นข้อความที่ใช้ตรวจสอบความถูกต้องของข้อมูลที่ได้ทำการส่งไป เช่น ข้อมูลเกี่ยวกับคีย์ รายละเอียดเกี่ยวกับการต่อรองค่าพารามิเตอร์ต่างๆในครั้งก่อน และ ข้อมูลใดๆที่จะเป็นการระบุตัวตนของทั้งเซิร์ฟเวอร์และไคลเอ็นต์ โดยค่าเหล่านี้จะต้องทำให้เป็นแฮชแวลู (hash value) ก่อนทำการส่งออกไป

4.1.4 การสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูลและมีการพิสูจน์ตนของเซิร์ฟเวอร์

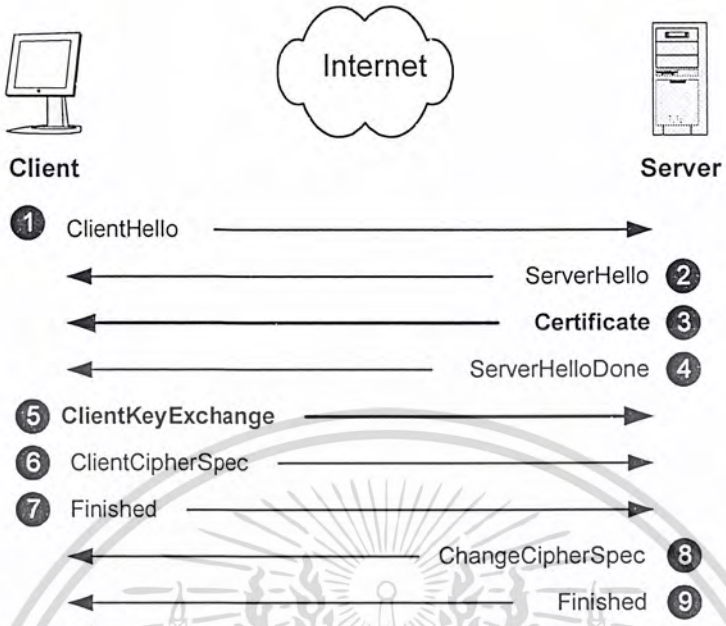
กระบวนการทำงานของ SSL โพรโตคอลในข้างต้นที่ใช้การเข้ารหัสข้อมูลเพียงอย่างเดียวระหว่างสองระบบนั้นมันยังไม่มีความปลอดภัยเพียงพอ พิจารณาได้จากรูปที่ 4-4 เมื่อ Alice ที่มีบทบาทเป็นไคลเอ็นต์ และ Bob ที่บทบาทเป็นเซิร์ฟเวอร์ และมีบุคคลที่อยู่ตรงกลางชื่อว่า Trudy โดยขั้นแรก Trudy จะแสดงบทบาท

ตัวเองเป็น Alice โดยจะบอก Bob ว่า "I'm Alice" และ Bob จะส่งข้อความว่า "I'm Bob" พร้อมกับส่งกุญแจสาธารณะของตัวเองให้แก่ Trudy ซึ่งขณะนี้ได้ปลอมตัวเป็น Alice เมื่อได้กุญแจสาธารณะของ Bob Trudy จะทำการสร้างกุญแจลับขึ้นมาแล้วใช้กุญแจสาธารณะของ Bob มาเข้ารหัสกุญแจลับที่ได้สร้างขึ้น เสร็จแล้วก็ทำการส่งข้อมูล(ciphertext) นั้นให้แก่ Bob ณ จุดนี้ Trudy สามารถที่จะติดต่อสื่อสารกับ Bob ได้ จากนั้น Trudy จะทำการปลอมตัวเองให้เป็น Bob ที่มีบทบาทเป็นเซิร์ฟเวอร์ โดย Trudy จะรับข้อความจาก Alice ที่ส่งมาว่า I'm Alice และ Trudy จะทำการสร้างกุญแจสาธารณะของตัวเองขึ้นมาและส่งข้อความบอกว่า Alice ว่า I'm Bob พร้อมกับส่งกุญแจสาธารณะของตัวเองไปให้แก่ Alice ณ ตอนนี้ Trudy ก็สามารถติดต่อสื่อสารกับ Alice ได้ เมื่อถึงจุดนี้ Trudy จะทราบข้อมูลทุกอย่างที่ Bob กับ Alice ติดต่อกัน ดังนั้น Trudy สามารถสร้างความเสียหายให้เกิดแก่ Bob และ Alice ได้ ซึ่งเราเรียกปัญหาที่เกิดขึ้นนี้ว่า การโจมตีแบบแมนอินเดอะมิดเดิล SSL (Man-in-the–middle SSL)



รูปที่ 4-4 แสดงการโจมตีแบบแมนอินเดอะมิดเดิล (Man-in-the-middle)

โพรโตคอลได้ทำการแก้ไขปัญหาดังกล่าวโดยได้เพิ่มวิธีการที่เรียกว่าการพิสูจน์ตนของเซิร์ฟเวอร์ จากรูปที่ 4-5 จะเห็นได้ว่าข้อความ Certificate ถูกนำมาแทนที่ข้อความ ServerKeyExchange



รูปที่ 4-5 แสดงสองข้อความใหม่ที่ใช้สำหรับการพิสูจน์ตนของเซิร์ฟเวอร์

4.1.5 ใบรับรองสิทธิ์ (Certificate)

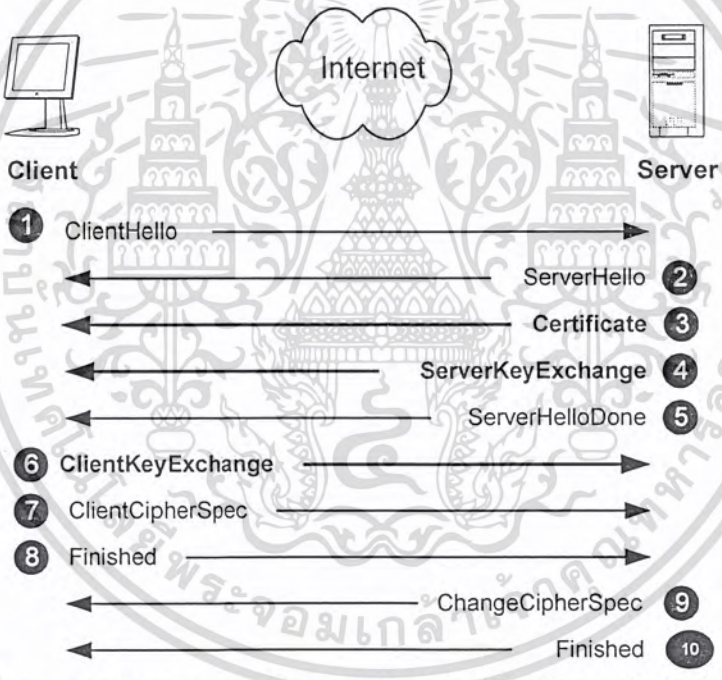
การที่ไคลเอ็นต์จะสามารถทำการระบุตัวตนของเซิร์ฟเวอร์ได้ เซิร์ฟเวอร์จะส่งข้อมูลต่างๆที่สามารถเป็นการระบุตัวตนได้เช่น ชื่อ, กุญแจสาธารณะของเซิร์ฟเวอร์ ไปให้แก่ CA (Certificate Authority) ซึ่ง CA จะแบ่งออกเป็นสองลักษณะคือ Internal CA หรือ External CA (ซึ่งจะได้กล่าวถึงในหัวข้อที่ 4-8 Certificate Authority (CA)) และ CA จะทำกระบวนการที่เรียกว่าการ Sign() โดยจะใช้คีย์ส่วนตัวของ CA ทำการ Sign ข้อมูลที่ได้รับมาจากเซิร์ฟเวอร์ และเมื่อนำกุญแจสาธารณะของ CA กับข้อมูลที่ได้มาจากการ Sign ดังกล่าวจาก CA แล้วนำมาทำกระบวนการที่เรียกว่าการ Verify() ก็จะสามารถพิสูจน์ตัวตนของเซิร์ฟเวอร์ได้ จากรูปไคลเอ็นต์จะต้องมีกุญแจสาธารณะของ CA ที่เป็น CA ที่เซิร์ฟเวอร์ได้นำข้อมูลต่างๆไปให้ CA นั้นทำการ Sign ให้ โดยข้อความ Certificate ของฝั่งเซิร์ฟเวอร์จะประกอบไปด้วย กุญแจสาธารณะของเซิร์ฟเวอร์และข้อมูลที่ได้รับการ Sign มาจาก CA นั้น ส่งไปให้แก่ไคลเอ็นต์

4.1.6 ClientKeyExchange

เมื่อไคลเอ็นต์ได้รับข้อความ Certificate จากเซิร์ฟเวอร์ จะใช้ฟังก์ชัน Verify() ในการพิสูจน์ตัวตนของเซิร์ฟเวอร์ถ้าการพิสูจน์ตัวตนถูกต้อง ไคลเอ็นต์จะนำกุญแจสาธารณะของเซิร์ฟเวอร์ที่ได้ส่งมาด้วยนั้น นำมาเข้ารหัสกุญแจลับที่ได้สร้างขึ้นมาตามกระบวนการเดิม

4.1.7 การแยกระหว่างการพิสูจน์ตนกับการเข้ารหัสข้อมูล

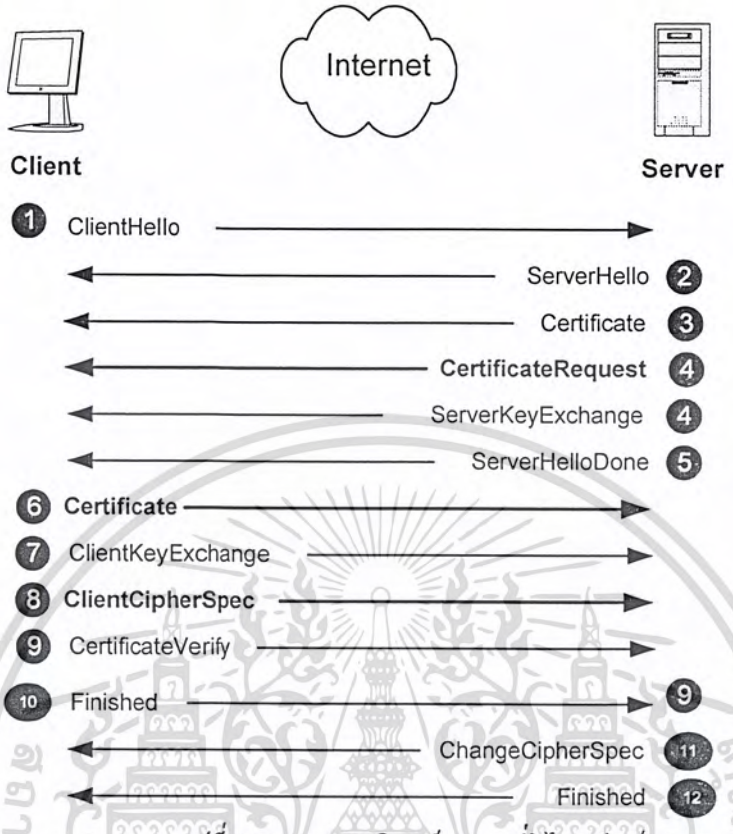
การใช้ข้อความ Certificate เพียงอย่างเดียวในการทำทั้งการพิสูจน์ตนและการเข้ารหัสข้อมูลนั้น เป็นวิธีการที่ไม่ดีนักเนื่องจากว่า ยกตัวอย่างเช่นมีหลายๆ อัลกอลิทึมที่ใช้ในการสร้างกุญแจสาธารณะ ที่มีไว้ใช้สำหรับการ Sign ข้อมูลเท่านั้นไม่สามารถนำมาใช้ในการเข้ารหัสข้อมูลได้ เช่น DSA (Digital Signature Algorithm) ดังนั้นจึงมีการแยกข้อความระหว่างการพิสูจน์ตนกับข้อความสำหรับการเข้ารหัสออกจากกัน ดังรูปที่ 4-6 เซิร์ฟเวอร์จะใช้ข้อความ Certificate สำหรับการพิสูจน์ตนของเซิร์ฟเวอร์ ส่งไปให้แก่ไคลเอ็นต์ก่อน ต่อจากนั้นจึงทำการส่งข้อความ ServerKeyExchange ที่ใช้สำหรับการเข้ารหัสข้อมูลให้แก่ไคลเอ็นต์ ในฝั่งของไคลเอ็นต์ถ้าการพิสูจน์ตัวตนของเซิร์ฟเวอร์ถูกต้อง ไคลเอ็นต์จะใช้กุญแจสาธารณะของเซิร์ฟเวอร์ทำการเข้ารหัสกุญแจลับที่ได้สร้างขึ้นตามกระบวนการของ SSL โพรโตคอล



รูปที่ 4-6 แสดงการแยกระหว่างข้อความของการพิสูจน์ตนกับการเข้ารหัสข้อมูลออกจากกัน

4.1.8 การพิสูจน์ตัวตนของไคลเอ็นต์

เมื่อเซิร์ฟเวอร์มีความต้องการที่จะพิสูจน์ตัวตนของฝั่งไคลเอ็นต์ ไคลเอ็นต์จะต้องนำข้อมูลต่างๆของไคลเอ็นต์ไปให้แก่ CA ที่เซิร์ฟเวอร์เชื่อถือ ทำการ Sign ข้อมูลให้เพื่อที่ไคลเอ็นต์จะสามารถพิสูจน์ตัวตนได้ ดังนั้นไคลเอ็นต์จะต้องมีการสร้างกุญแจสาธารณะของตัวเองขึ้นมาเพื่อใช้ในการพิสูจน์ตัวตนของไคลเอ็นต์ โดยกุญแจสาธารณะของไคลเอ็นต์นี้จะใช้ในการพิสูจน์ตัวตนเท่านั้นไม่ได้ใช้ในการเข้ารหัสข้อมูลใดๆ จากรูปที่ 4-7 ข้อความที่เพิ่มขึ้นมาคือ



รูปที่ 4-7 แสดงการพิสูจน์ตนของฝั่งไคลเอ็นต์

CertificateRequest

จะเป็นข้อความจากเซิร์ฟเวอร์ที่เป็นการบอกให้แก่ไคลเอ็นต์ทราบว่าเซิร์ฟเวอร์มีความต้องการที่จะทำการพิสูจน์ตัวตนของไคลเอ็นต์ โดยจะเห็นได้ว่าข้อความนี้จะอยู่หลังข้อความ Certificate เนื่องจาก จะต้องมีการพิสูจน์ตัวตนของเซิร์ฟเวอร์ให้ถูกต้องก่อน เมื่อการพิสูจน์ตัวตนของเซิร์ฟเวอร์ถูกต้อง เซิร์ฟเวอร์จึงทำการร้องขอการพิสูจน์ตัวตนของฝั่งไคลเอ็นต์

Certificate

จะเป็นข้อความของฝั่งไคลเอ็นต์ที่เป็นข้อมูลที่ไคลเอ็นต์ได้รับจากการส่งข้อมูลของไคลเอ็นต์ เช่น กุญแจสาธารณะของไคลเอ็นต์ ไปให้แก่ CA ที่เซิร์ฟเวอร์เชื่อถือทำการ Sign ข้อมูลดังกล่าวให้

CertificateVerify

จะเป็นข้อความที่คล้ายๆกับข้อความ Finished แต่สิ่งที่แตกต่างกันคือ ข้อมูลของคีย์ซึ่งถ้าเป็นข้อความ Finished จะเป็นข้อมูลของกุญแจลับ แต่ถ้าเป็นข้อความ CertificateVerify จะเป็นข้อมูลของกุญแจสาธารณะที่ไคลเอ็นต์ใช้ในการพิสูจน์ตัวตน

4.2 OpenSSL

4.2.1 OpenSSL คืออะไร

OpenSSL คือโปรเจ็คของทีมงานหนึ่งที่มีความพยายามในการพัฒนา OpenSource ที่สามารถไปเรียกใช้การทำงานของ Secure Socket Layer (SSL) และ Transport Layer Security (TLS) โพรโตคอลได้อย่างมีประสิทธิภาพและทั้งนี้ OpenSource นี้จะประกอบด้วยไลบรารีต่างๆ ที่เกี่ยวข้องกับการเข้ารหัสและถอดรหัสข้อมูลโดยการใช้อัลกอลิทึมต่างๆ ที่มีความปลอดภัยสูงเช่น RSA, DSA, 3DES ฯลฯ ไว้อย่างครบถ้วน ซึ่งข้อมูลและรายละเอียดต่างๆ หาได้ที่ www.openssl.org

4.2.2 การใช้งาน OpenSSL

ก่อนที่จะเริ่มใช้งาน OpenSSL เราจะต้องทำการติดตั้ง OpenSSL เสียก่อนถึงจะมีไลบารีและคำสั่งในการสร้างคีย์ให้ใช้งาน เริ่มต้นให้ไปดาวน์โลด Source OpenSSL 0.9.6l จาก www.openssl.org ที่เลือกเวอร์ชันนี้เพราะมีความเสถียรภาพและปลอดภัยแล้ว วิธีกี่การติดตั้งดูได้จาก README.TXT ที่อยู่ในซอร์สหรือถ้าหากได้ทำการติดตั้ง Apache-ssl ไว้แล้วก็ไม่จำเป็นต้องติดตั้ง Source Openssl อีกเพราะมีไลบรารีจาก Apche-ssl แล้ว จากนั้นก็จะสามารถใช้คำสั่งและไลบรารีต่างๆ ของ Openssl ได้ และที่สำคัญในขณะกำลังติดตั้งเราต้องสังเกตดูว่าเก็บเฮ็ดเดอร์ไฟล์และไลบรารี เก็บอยู่ที่ไหนด้วยเพราะต้องใช้ตอนคอมไพล์โปรแกรม

4.2.2.1 ตำแหน่งเฮเดอร์ไฟล์และไลบรารี

การที่เราจะทำการคอมไพล์ โปรแกรมที่ include เฮเดอร์ไฟล์ของ Openssl เราต้องรู้จักตำแหน่งที่เก็บเฮเดอร์ไฟล์และไลบรารีเสียก่อน ถ้าเราทำการติดตั้ง Openssl แบบมาตรฐานแล้วเฮเดอร์ไฟล์และไลบรารีของ Openssl ก็จะอยู่ที่เฮเดอร์ไฟล์และไลบรารีมาตรฐานของ ลินุกซ์ คือ

/usr/include/openssl สำหรับเฮเดอร์ไฟล์

/usr/lib/ สำหรับไลบรารี

ไลบรารีที่จำเป็นคือ libssl.a และ libcrypto.a

ถ้าหากหาเฮเดอร์ไฟล์และไลบรารี ที่ตำแหน่งมาตรฐานไม่เจอหลังจากที่ได้ทำการติดตั้งไปแล้ว ให้ลองไปหาดูที่ /usr/local/ssl/ ซึ่งอาจจะเจอหรืออยู่ตามตำแหน่งที่บอกไว้ตอนติดตั้ง Openssl

4.2.2.2 การคอมไพล์

ถ้าหากเฮเดอร์ไฟล์และไลบรารี ของ Openssl อยู่ดังตำแหน่งมาตรฐานของเฮเดอร์ไฟล์และไลบรารีของ ลินุกซ์ เราไม่ต้องอ้างถึง Path เลย ทำการคอมไพล์ และ ลิงก์ไลบรารี ได้ดังนี้

```
gcc –c TestOpenssl.c                          /*Compile*/
gcc –o TestOpenssl TestOpenssl.o –lssl -lcrypto    /*Link Library*/
```

ถ้าหากเฮเดอร์ไฟล์และไลบรารีของ Openssl ไม่ได้อยู่ดังตำแหน่งมาตรฐานของเฮเดอร์ไฟล์และไลบรารี ของลินุกซ์เราต้องอ้างถึง Path ก่อนจะทำการทำการคอมไพล์และ ลิงก์ไลบรารี ได้ดังนี้

```
gcc –c TestOpenssl.c –I/~path header file~                    /*Compile*/
gcc –o TestOpenssl TestOpenssl.o –L/~path library file~ -lssl -lcryto    /*Link Library*/
```

4.3 Certificate Authority (CA)

CA เป็นองค์กรหรือสมาคมที่จะออกใบรับรองสิทธิ์ให้แก่ผู้ขอ ใบรับรองสิทธิ์ก็คือใบที่ใช้พิสูจน์ถึงตนเองเมื่อต้องการติดต่อกับผู้อื่น เหมือนกับการใช้บัตรประจำตัวประชาชน ต่างกันตรงที่ใบรับรองสิทธิ์นี้เอาไว้รับรองสิทธิ์ในการสื่อสารบนระบบ ผู้ที่ขอใบรับรองสิทธิ์จาก CA จะต้องมั่นใจในใบรับรองสิทธิ์นั้นแต่ไม่ได้หมายความว่า CA นั้นจะไม่มีความผิดพลาดเลย ตัวอย่างองค์กรที่เป็น CA เช่น Verisign รายละเอียดต่างๆ หาได้ที่ www.verisign.com

ตามหลักพื้นฐานแล้ว CA มีสองชนิด

Public CAs หรือ External CAs

เป็นองค์กรที่จะออกใบรับรองสิทธิ์ให้แก่บุคคลหรือองค์กรโดยทั่วไปเช่น Verisign จะออกใบรับรองสิทธิ์ให้กับเวบไซต์ (web sites) หรือแอพพลิเคชันที่ต้องการให้มีการเข้ารหัสและการรับรองสิทธิ์ของผู้ใช้ในการทำกิจกรรมต่างๆ เช่น อีคอมเมิร์ซ (e-commerce) จะต้องมีระบบการรับส่งข้อมูลที่ปลอดภัยเช่น รหัสผ่านของเครดิตการ์ดของลูกค้า ซึ่งโดยมากแล้วถ้าเป็น External CAs ถ้าเราไปขอใบรับรองสิทธิ์เราจะไม่ได้มาฟรีๆ จะต้องมีการจ่ายเงินให้แก่ External CAs นั้นๆก่อน

Private CAs หรือ Internal CAs

เป็นองค์กรที่จะออกใบรับรองสิทธิ์ให้แก่คนที่อยู่ภายในองค์กรเพื่อใช้งานภายในองค์กรเอง องค์กรภายนอกจะไม่สามารถใช้ CA และจะไม่สามารถไว้วางใจการออก CA แบบนี้ได้แต่ก็ขึ้นอยู่กับ Internal CAs นั้นๆว่ามีกฎระเบียบเป็นอย่างไร ซึ่ง CA ประเภทนี้จะออกใบรับรองสิทธิ์ให้ฟรีเพราะถือว่าบุคคลเหล่านั้นอยู่ในองค์กรภายใน

การทำโครงงานนี้ได้ทำระบบ CA แบบ Private CA เพราะเห็นว่าเหมาะสมที่จะออกใบรับรองสิทธิ์ให้แก่ผู้ใช้ของ IsagMQ เท่านั้นเพื่อการสื่อสารอย่างปลอดภัยภายในกลุ่ม ในที่นี้จะอธิบายการสร้างและออกแบบ Private CA เท่านั้นซึ่งจะอยู่ในบทที่ 6

4.3.1 Third Party

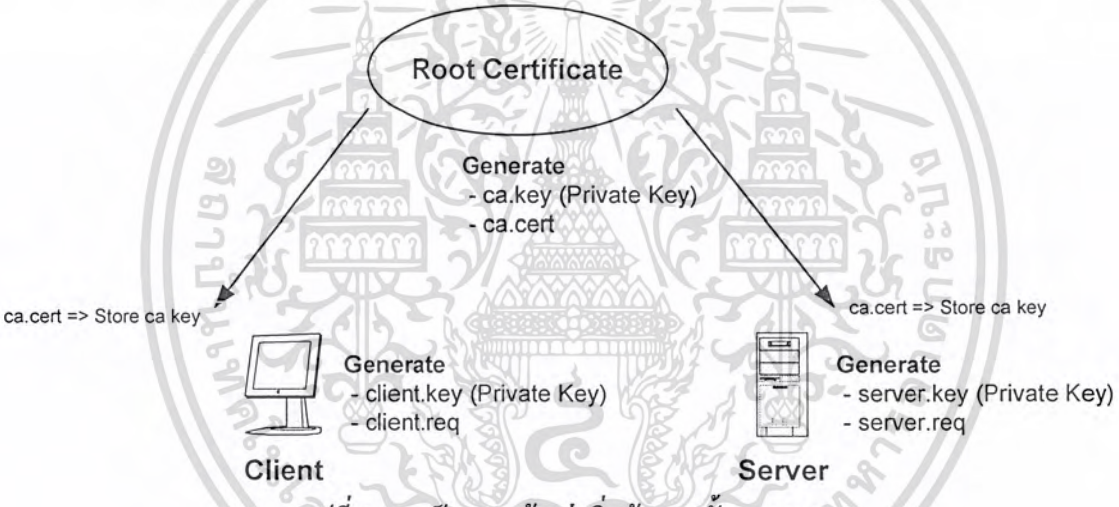
Root CA เป็นผู้ที่จะออกใบรับรองสิทธิ์ให้แก่ผู้ร้องขอ

Client เป็นผู้ขอใบรับรองสิทธิ์เพื่อใช้ในการพิสูจน์ตน

Server เป็นผู้ขอใบรับรองสิทธิ์เพื่อใช้ในการพิสูจน์ตน

4.3.2 ข้อมูลเริ่มต้น

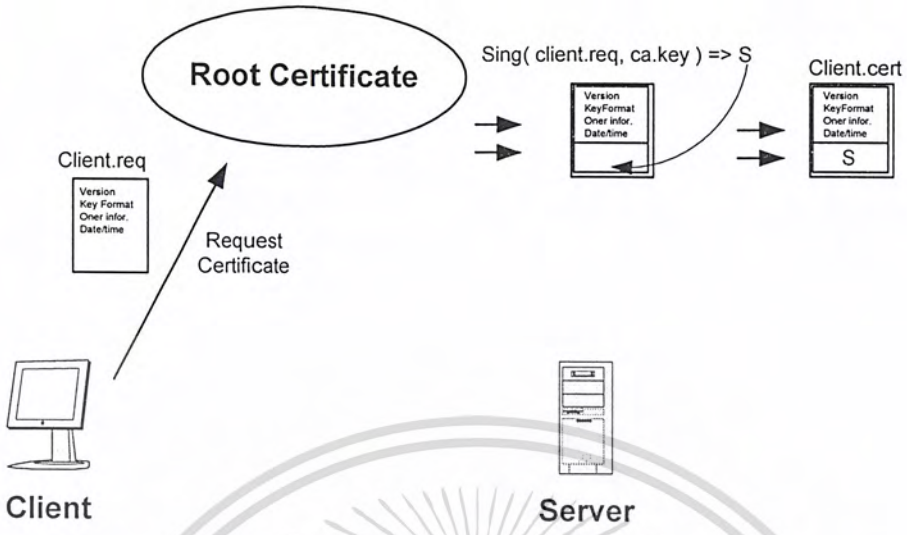
จากรูปที่ 4-8 Root Certificate จะสร้างca.key ซึ่งคือคีย์ที่ประกอบด้วยคีย์ส่วนตัวและกุญแจสาธารณะ เซิร์ฟเวอร์จะสร้าง server.key ซึ่งคือคีย์ที่ประกอบไปด้วยคีย์ส่วนตัวและกุญแจสาธารณะ โดยเซิร์ฟเวอร์ จะใช้กุญแจสาธารณะของตนเองและข้อมูลอื่นๆที่สามารถระบุตัวตนของเซิร์ฟเวอร์ได้ สร้างเป็น server.req ในแนวทางเดียวกันไคลเอ็นต์จะสร้าง client.key ซึ่งคือคีย์ที่ประกอบไปด้วยคีย์ส่วนตัวและกุญแจสาธารณะ โดยไคลเอ็นต์จะใช้กุญแจสาธารณะของตนเองและข้อมูลอื่นๆที่สามารถระบุตัวตนของไคลเอ็นต์ได้ สร้างเป็น client.req โดย Root Certificate จะส่งกุญแจสาธารณะของตัวเองให้แก่ทั้งไคลเอ็นต์และเซิร์ฟเวอร์



รูปที่ 4-8 จะเป็นการสร้างค่าเริ่มต้นของทั้ง 3 ระบบ

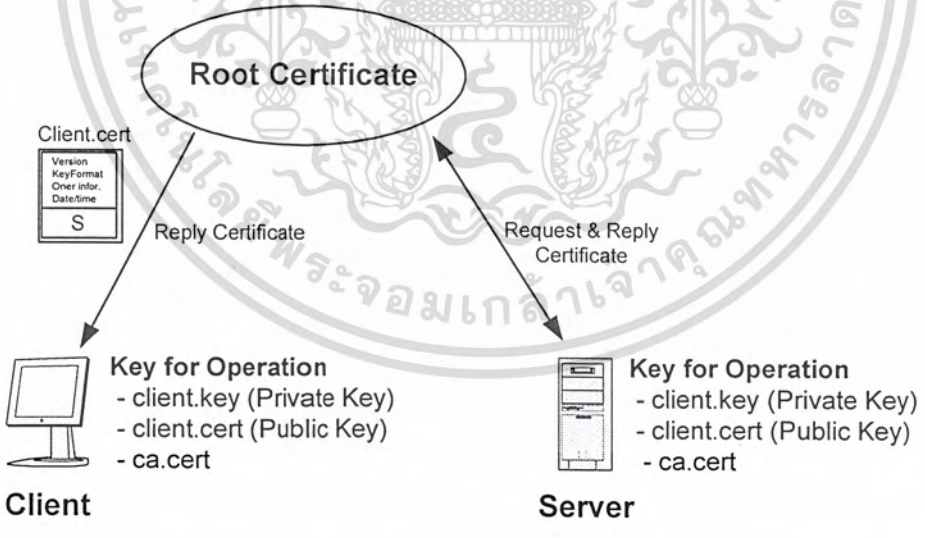
4.3.3 การร้องขอ ใบรับรองสิทธิ์ของไคลเอ็นต์ (Client Certificate Request)

จากรูปที่ 4-9 ไคลเอ็นต์จะส่ง client.req ไปให้แก่ Root Certificate จากนั้น Root Certificate จะออกใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์ โดยมีกระบวนการพื้นฐานดังนี้คือ จะนำเอาข้อมูลจาก client.req มาทำ MD5 ซึ่งจะได้ค่าแฮช (hash) มาหนึ่งค่า จากนั้นใช้ฟังก์ชั่น Sign โดยมีพารามิเตอร์สองตัวคือ ค่าแฮชที่ได้ กับ กุญแจลับของ Root Certificate จะได้ค่า S (Digital Signature) ออกมานำข้อมูลจาก client.req และ ค่า S ที่ได้ ไปเก็บไว้ใน client.cert ดังรูปที่ 4-9



รูปที่ 4-9 แสดง Root Certificate ออกใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์

จากรูปที่ 4-10 Root Certificate จะส่ง client.cert ซึ่งเป็นไฟล์ที่ไคลเอ็นต์ สามารถใช้ในการพิสูจน์ตนได้ให้แก่ไคลเอ็นต์ได้ จากรูปในทางด้านของเซิร์ฟเวอร์ก็จะทำตามกระบวนการเดียวกันกับไคลเอ็นต์ในการขอใบรับรองสิทธิ์ต่อ Root Certificate

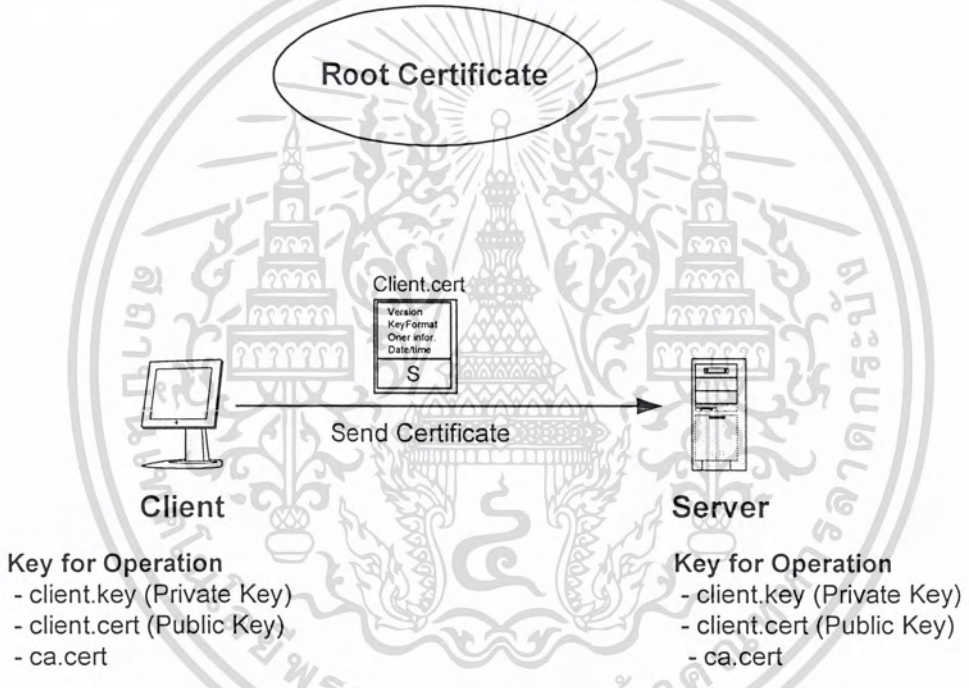


รูปที่ 4-10 แสดง Root Certificate ออกใบรับรองสิทธิ์ให้แก่ทั้งไคลเอ็นต์และเซิร์ฟเวอร์

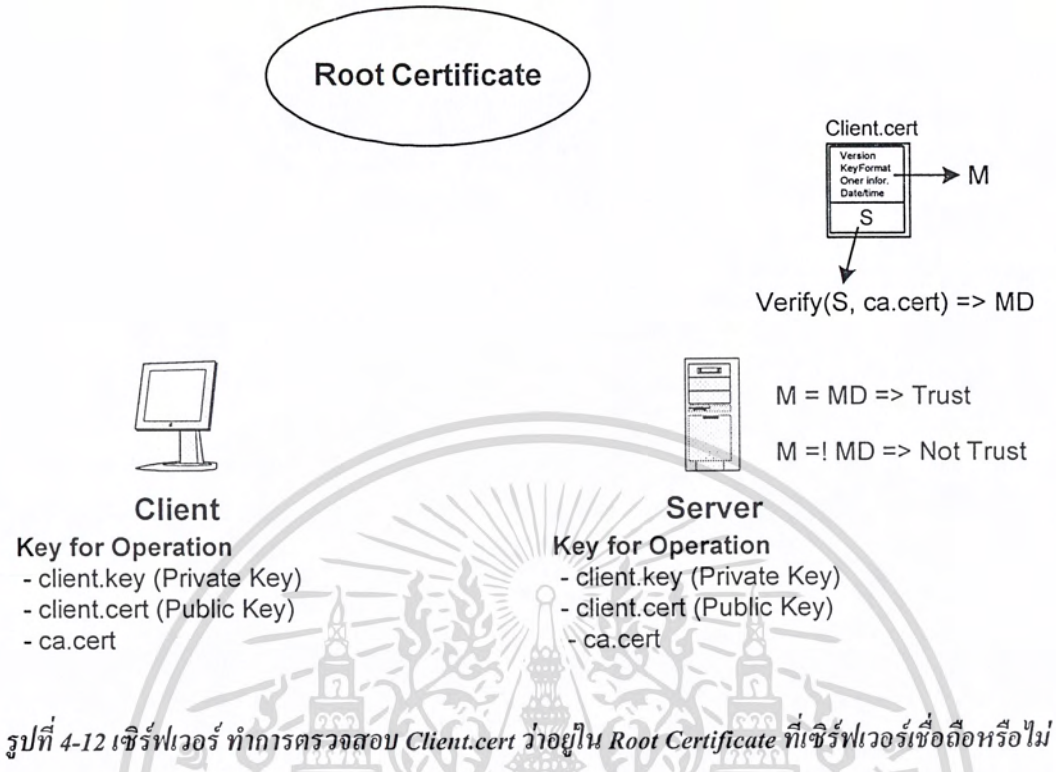
4.3.4 การสื่อสารกันโดยมีใบรับรองสิทธ์ (Certificate Authentication)

จากรูปที่ 4-11 เมื่อเซิร์ฟเวอร์ร้องขอการพิสูจน์ตนของฝั่งไคลเอ็นต์จากนั้นไคลเอ็นต์จะส่งข้อความ Certificate (client.cert) ให้แก่เซิร์ฟเวอร์ทำการพิสูจน์ตนของฝั่งไคลเอ็นต์

จากรูปที่ 4-12 เมื่อเซิร์ฟเวอร์ได้รับไฟล์ client.cert ซึ่งไฟล์นี้จะประกอบด้วยสองส่วนคือ ส่วนของข้อมูลส่วนตัวของไคลเอ็นต์และ ค่า S (Digital Signature) กระบวนการพิสูจน์ตนทางฝั่งเซิร์ฟเวอร์คือ จะนำข้อมูลส่วนตัวของไคลเอ็นต์มาทำ MD5 ได้ค่าแฮช (hash) มาหนึ่งค่าคือ M จากนั้นนำค่า S ของไคลเอ็นต์และ กุญแจสาธารณะของ Root Certificate มาเข้าฟังก์ชั่น Verify ค่าที่ได้ออกมาคือ MD นำค่า M กับ MD มาเปรียบเทียบกันถ้าค่านั้นเท่ากันแสดงว่าการพิสูจน์ตนถูกต้อง ถ้าค่าไม่เท่ากันแสดงว่าการพิสูจน์ตนผิดพลาด



รูปที่ 4-11 ไคลเอ็นต์ ทำการส่งข้อความ Certificate ให้แก่เซิร์ฟเวอร์เพื่อทำการพิสูจน์ตน



รูปที่ 4-12 เซิร์ฟเวอร์ ทำการตรวจสอบ Client.cert ว่าอยู่ใน Root Certificate ที่เซิร์ฟเวอร์เชื่อถือหรือไม่

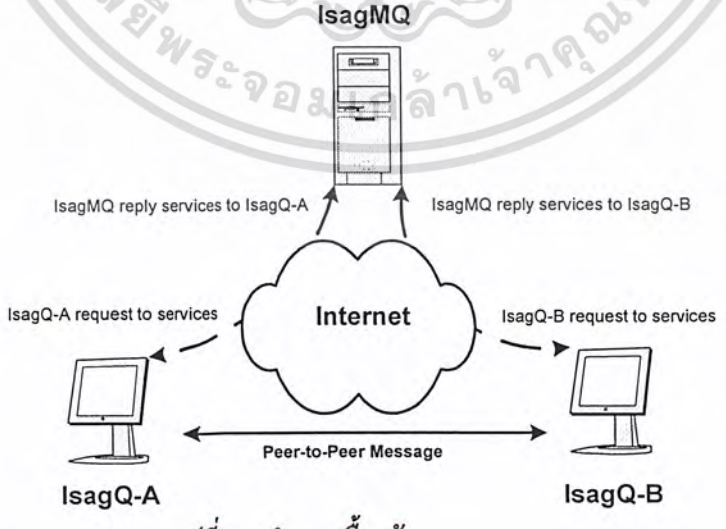
บทที่ 5
การออกแบบและพัฒนาโครงงาน

โครงงานนี้มุ่งเน้นในการสร้างโปรแกรมแม่ข่ายสำหรับรับส่งสารด่วนแบบปลอดภัย โดยใช้หลักการสองข้อคือ ข้อแรกก่อนที่ไคลเอ็นต์ (IsagQ) จะใช้บริการได้จะต้องมีการพิสูจน์ตนก่อนเท่านั้น ข้อสองการสื่อสารระหว่างไคลเอ็นต์ (IsagQ) กับเซิร์ฟเวอร์ (IsagMQ) จะต้องอยู่ในช่องทางที่ปลอดภัยโดยใช้การเข้ารหัสข้อมูล

การออกแบบและ พัฒนาโครงการได้ใช้ทฤษฎีและหลักการจากบทที่ 2 – 8 มาเป็นพื้นฐานการสร้าง IsagMQ เพื่อให้บรรลุตามวัตถุประสงค์ที่ได้ตั้งเอาไว้ โดยมีขั้นตอนการออกแบบตามหัวข้อต่อไปนี้

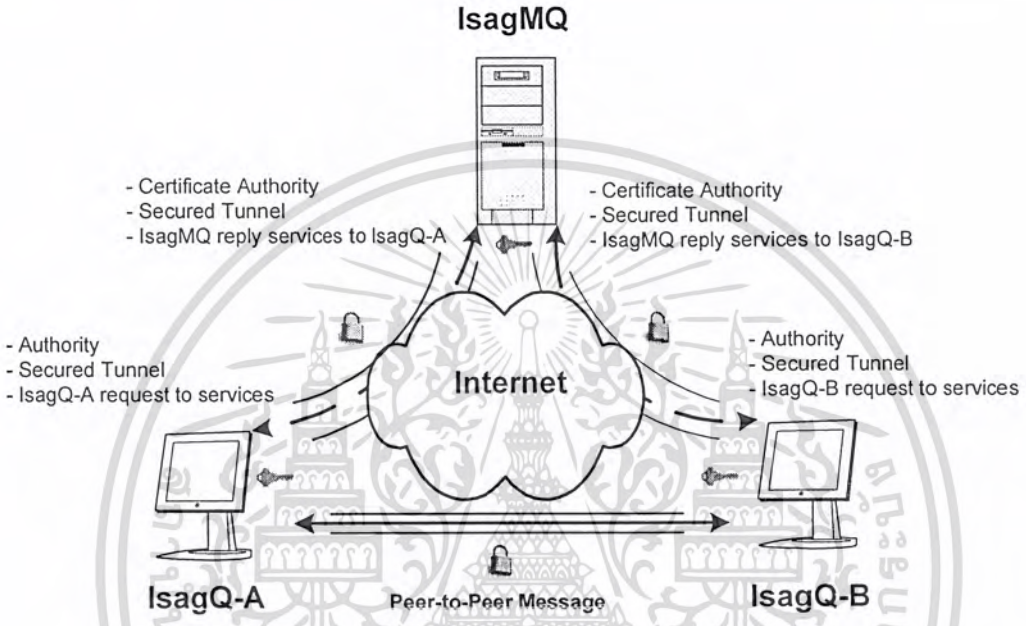
5.1 แนวคิดที่ใช้ในการออกแบบ

จากบทที่ 2 ได้กล่าวถึงระบบรับส่งสารด่วนที่เป็นที่นิยมใช้ในปัจจุบันแล้ว และจะนำหลักการและรูปแบบบางส่วนที่เห็นว่าเหมาะสมกับ IsagMQ โดยจะคำนึงถึงความปลอดภัยในการให้บริการเป็นหลัก รูปแบบของระบบรับส่งสารด่วนจะเลือกใช้ **การทำงานแบบเพียร์ทูเพียร์เมซเซจ (Peer–to–Peer Message)** เพราะเรามีวิธีป้องกันข้อเสียของแบบนี้ได้ โดยการส่งหมายเลขไอพีระหว่างไคลเอ็นต์นั้นจะต้องมีการเข้ารหัสก่อนทุกครั้งเวลาจะใช้ค่อยถอดรหัสซึ่งการเข้ารหัสและถอดรหัสจะอยู่ในระบบของ IsagQ ผู้ใช้ไม่อาจมองเห็นได้ รูปแบบการทำงานเบื้องต้นจะเป็นตามรูปที่ 5-1 รายละเอียดของการทำจะเหมือนในบทที่ 2 การทำงานแบบเพียร์ทูเพียร์เมซเซจ



รูปที่ 5-1 ทำงานเบื้องต้นของ IsagMQ

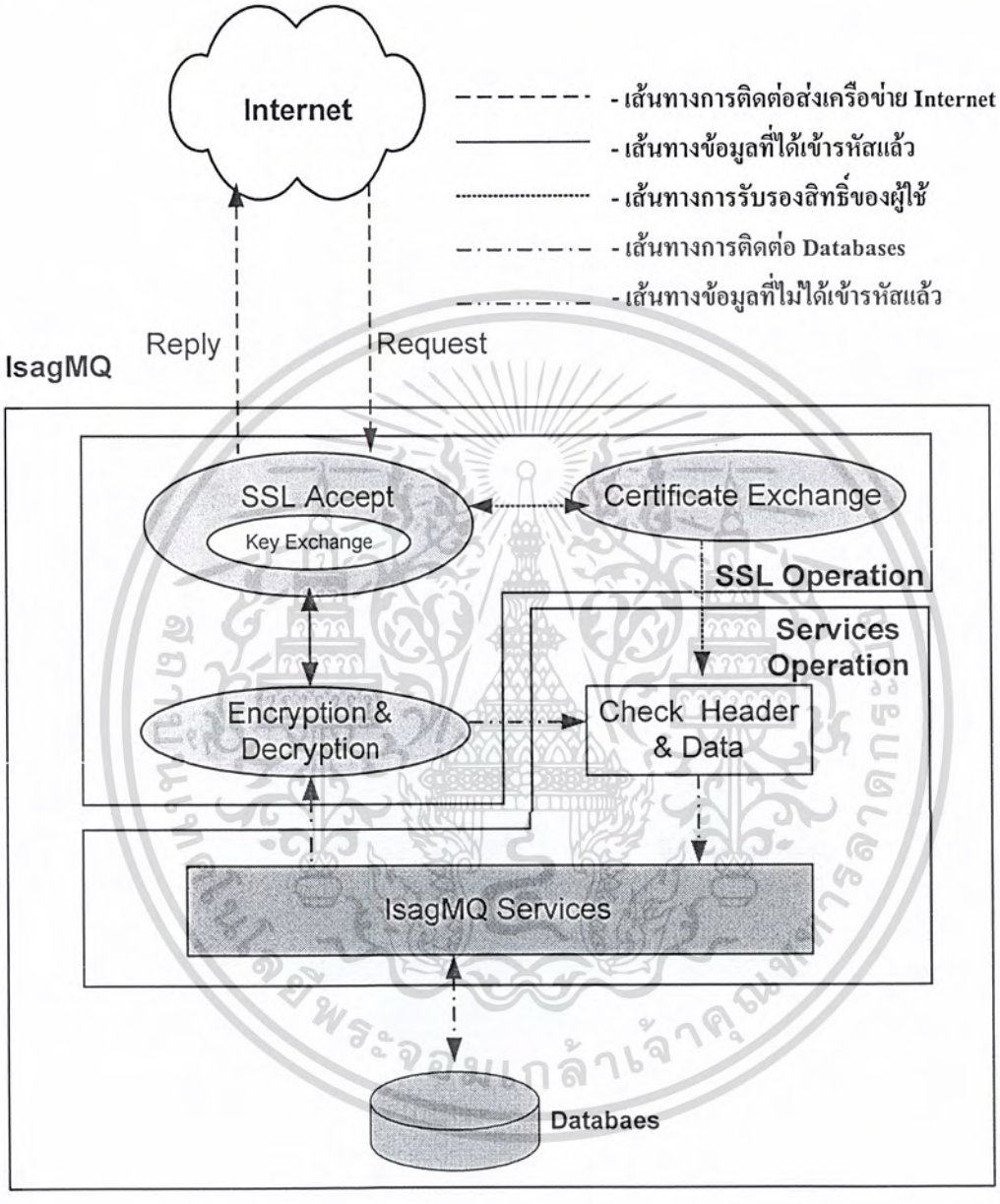
เมื่อแนวคิดเบื้องต้นของระบบการทำงานแล้วต่อไปจะออกแบบบให้ระบบการทำงานข้างต้นให้มีความปลอดภัย โดยเน้นที่ IsagMQ ต้องให้บริการอย่างปลอดภัยแก่ IsagQ โดยได้นำ **แนวคิดของการรับรองสิทธิ์ของผู้ใช้บริการและการเข้ารหัสข้อมูล** ในบทที่ 3 มาใช้ในระบบเพื่อให้เกิดความปลอดภัย ระบบการทำงานดังรูปที่ 5-2



รูปที่ 5-2 การให้บริการอย่างปลอดภัยแก่ผู้ใช้

แนวคิดการทำงานของระบบตามรูปที่ 5-2 ตัวเซิร์ฟเวอร์ (IsagMQ) มีระบบการรับประกันสิทธิ์ของผู้ใช้ด้วยตัวเซิร์ฟเวอร์เองหรือเรียกว่า **Private Certificate Authority** (Private CA) และตัวไคลเอ็นต์เองก็ต้องมีระบบร้องขอการใช้สิทธิ์คือ **Authority** ทั้ง เซิร์ฟเวอร์และไคลเอ็นต์จะต้องมีระบบการสร้างช่องทางที่ปลอดภัยเรียกว่า **Secured Tunnel** โดยใช้การเข้ารหัสและถอดรหัสข้อมูล จากรูปที่ 5-2 เริ่มแรกถ้า IsagQ-A ต้องการขอใช้บริการจะต้องอ้างสิทธิ์ของตังเองไปที่ IsagMQ ก่อน จากนั้น IsagMQ จะทำการตรวจสิทธิ์ของ IsagQ-A ว่ามีสิทธิ์ถูกต้องหรือไม่ ถ้าถูกต้อง IsagMQ ก็จะอนุญาตให้ IsagQ-A เข้าใช้บริการได้โดยจะต้องสร้าง Secured Tunnel ก่อนที่จะมีการส่ง Contact List ให้แก่ IsagQ-A เมื่อ IsagQ-B ต้องการขอใช้บริการจะทำในลักษณะเดียวกัน จากนั้น IsagQ-A และ IsagQ-B ก็จะมีความมั่นใจได้ว่าคู่สนทนาของตนเป็นตัวจริงแน่นอนและสร้างช่องทางการสื่อสารที่ปลอดภัยระหว่าง IsagQ ด้วย ส่วนรายเอียดการสร้างระบบต่างๆ นั้นจะกล่าวในหัวข้อต่อไป

5.2 ส่วนประกอบของ IsagMQ

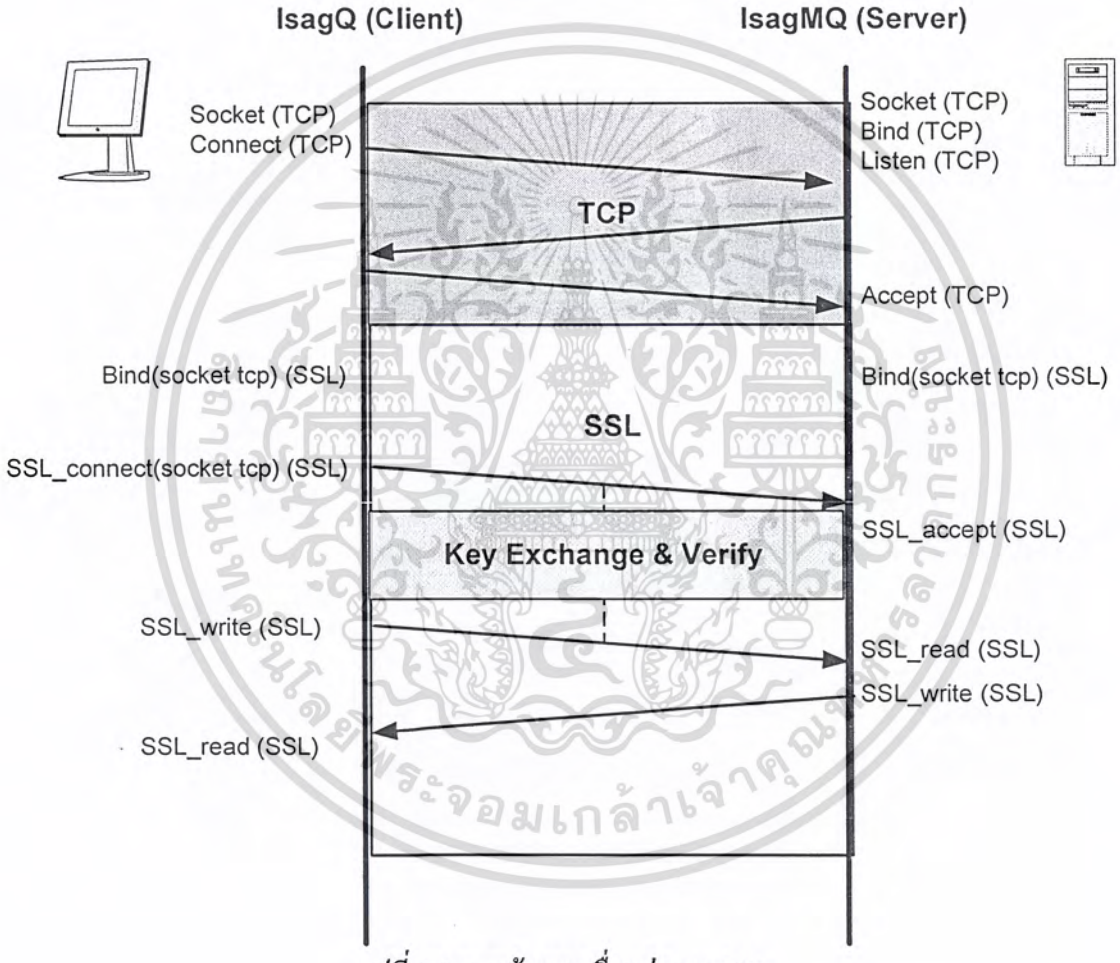


รูปที่ 5-3 **ส่วนประกอบของ IsagMQ**

การออกแบบและการทำงานข้อแต่ละส่วนจะกล่าวในหัวข้อต่อไป

5.3 การสร้างการเชื่อมต่อแบบ SSL (SSL Connection)

การสร้างการเชื่อมต่อแบบ SSL จะใช้วิธีการคือทำการสร้างการเชื่อมต่อแบบ TCP (TCP Connection) ขึ้นมาก่อนโดยใช้วิธีการในบทที่ 3 ซึ่งจะอภิบายการสร้างการเชื่อมต่อแบบ TCP พื้นฐานไว้แล้ว และจากนั้นนำเอา Connection Object ของ TCP มาทำการ Bind ให้เป็นการเชื่อมต่อแบบ SSL โดยใช้วิธีการและเอพีไอ (API) ของ OpenSSL การทำงานของโพรโตคอล SSL ดูได้ในบทที่ 4 ตัวอย่างโปรแกรมดูได้ในคู่มือโปรแกรม ลักษณะการทำงานดังรูปที่ 5-4 และในส่วนของ Key Exchange & verify จะกล่าวต่อไป



รูปที่ 5-4 การสร้างการเชื่อมต่อแบบ SSL

5.4 การเข้ารหัสและถอดรหัสข้อมูล

การสร้างความปลอดภัยจะใช้การเข้ารหัสสองแบบ คือ การเข้ารหัสแบบกุญแจเดี่ยว (Secret key cryptography) และ การเข้ารหัสแบบกุญแจคู่ (Public key cryptography) โดยการเข้ารหัสแบบกุญแจคู่จะใช้เข้ารหัสและถอดรหัสกุญแจเดี่ยว ส่วนกุญแจเดี่ยวจะใช้ในการเข้ารหัสและถอดรหัสข้อมูลที่ใช้ในการสื่อสาร การใช้การเข้ารหัสทั้งสองร่วมกันเพื่อเสริมจุดดีและแก้ไขจุดด้อยของแต่ล่ะแบบ ซึ่งจะทำให้ได้ประสิทธิภาพการเข้ารหัสที่ดีขึ้นกว่าการใช้แบบใดแบบหนึ่ง ซึ่งจะไม่ขอกล่าวถึงทฤษฎีและหลักการต่างๆ ของการเข้ารหัสและถอดรหัสข้อมูลเพราะถือว่าได้ทำการศึกษามาก่อนหน้านี้แล้ว และจะมีรูปแบบของการเข้ารหัสต่างๆให้ดูในภาคผนวก ก.

5.4.1 แบบของการเข้ารหัสแบบกุญแจเดี่ยวที่เลือกใช้

จะเลือกใช้ 3DES ซึ่งจะมี Block cipher แบบ CBC และมีขนาดของ Key เท่ากับ **112 bit** และจะใช้ อัลกอริทึม (Algorithm) ของ Deffie-Hellman ในการสร้างกุญแจส่วนตัว (Private Key)

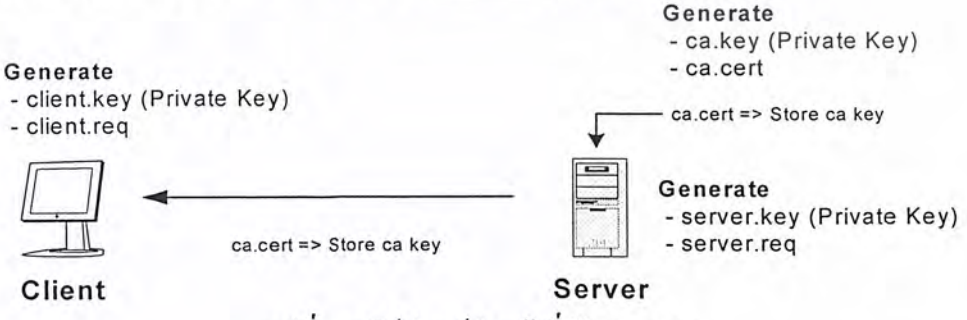
5.4.2 แบบของการการเข้ารหัสแบบกุญแจคู่ที่เลือกใช้

จะเลือกใช้ RSA และมีขนาดของกุญแจ (Key) เท่ากับ **1024 bit**

ระบบการเข้ารหัสและถอดรหัสข้อมูลจะอยู่ใน SSL โพรโตคอลอยู่แล้วถ้าเราสร้างการเชื่อมต่อแบบ SSL โดยใช้การดึงกุญแจ จาก Plain Key (Key ที่เก็บอยู่ในไฟล์) เราจะสามารถเลือกรูปแบบของกุญแจต่างๆ ได้จากการสร้าง Plain Key ไม่ว่าจะเป็นแบบของการเข้ารหัส ชนิดของการเข้ารหัส บล็อกไซเฟอร์แม้กระทั่งขนาดของกุญแจซึ่งการสร้างกุญแจในรูปแบบเพลนเท็กซ์ (Plaintext) จะกล่าวในหัวข้อต่อไป

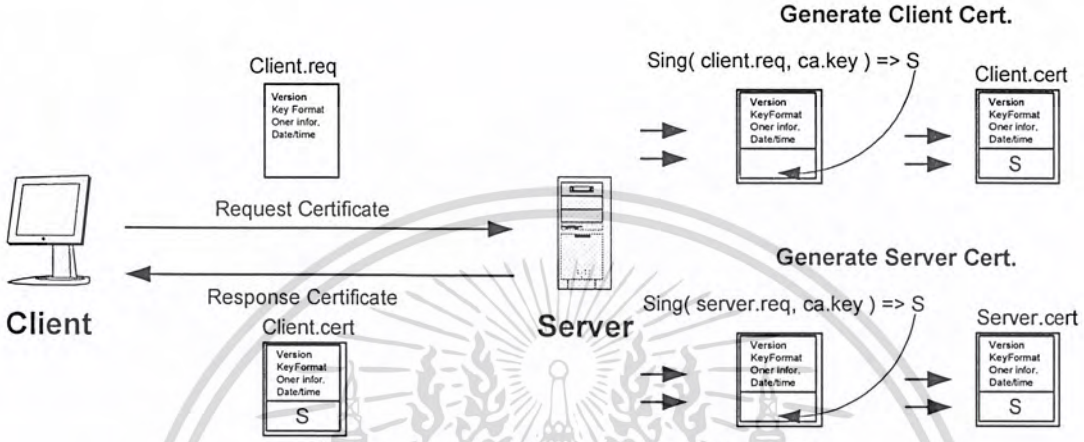
5.5 Private CA (Private Certificate Authority)

จากหลักการเรื่อง CA ในบทที่สี่ได้นำหลักการดังกล่าวมาออกแบบและสร้าง Private CA คือ จากที่เคยมี Third party นำมาออกแบบเป็น Two party ตัดส่วนของ Root CA ออกไปจะให้เซิร์ฟเวอร์เป็นทั้งเซิร์ฟเวอร์ และ Root CA นั้นคือหน้าที่ออกใบรับรองสิทธิ์ก็เป็นของเซิร์ฟเวอร์ดังรูปที่ 5-5



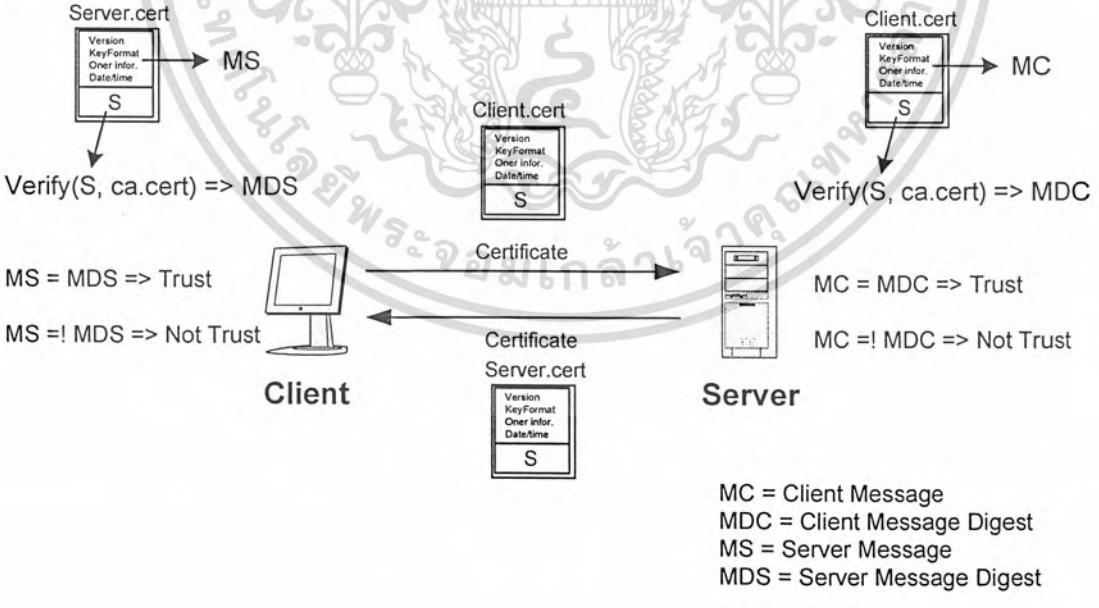
รูปที่ 5-5 เซิร์ฟเวอร์ทำหน้าที่เป็น Root CA

ถ้าไคลเอ็นต์ต้องการขอใบรับรองสิทธ์จะทำการขอไปที่เซิร์ฟเวอร์และเซิร์ฟเวอร์จะทำการออกใบรับรองสิทธ์ให้แก่ไคลเอ็นต์และเซิร์ฟเวอร์จะต้องออกใบรับรองสิทธ์ให้แก่ตัวเองด้วยเพื่อใช้ในการรับรองสิทธิ์กับไคลเอ็นต์ดังรูปที่ 5-6



รูปที่ 5-6 เซิร์ฟเวอร์ออกใบรับรองสิทธ์ให้แก่ไคลเอ็นต์

จากนั้นไคลเอ็นต์จะใช้ใบรับรองสิทธ์อ้างถึงสิทธ์ในการใช้บริการจากเซิร์ฟเวอร์และเซิร์ฟเวอร์จะใช้ใบรับรองสิทธ์ในการอ้างถึงสิทธิ์ที่จะส่งข้อมูลให้กับไคลเอ็นต์ดังรูปที่ 5-7 ก่อนที่จะมีการสื่อสารระหว่างไคลเอ็นต์ และเซิร์ฟเวอร์จะต้องมีการอ้างสิทธิ์กันก่อน



รูปที่ 5-7 เซิร์ฟเวอร์และไคลเอ็นต์ทำการตรวจสอบสิทธิ์

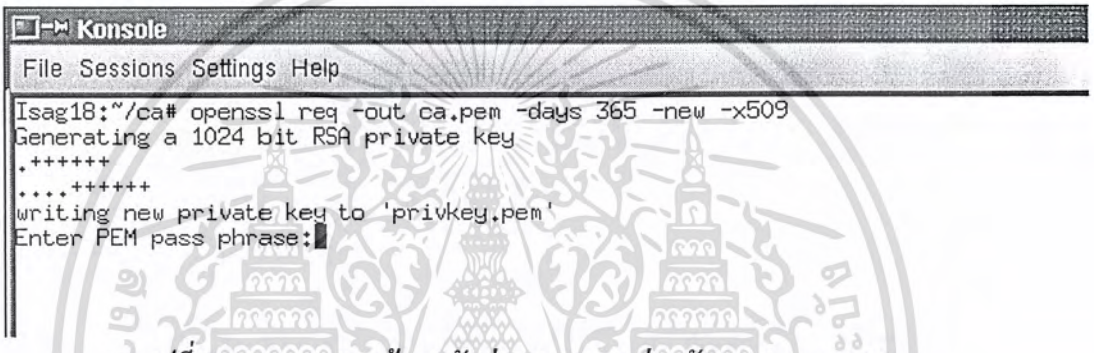
5.6 การสร้างกุญแจและใบรับรองสิทธิ์ในรูปแบบ Plaintext ด้วย OpenSSL และตัวอย่าง

ขั้นตอนการออกใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์และเซิร์ฟเวอร์โดยใช้ Internal CAs มีวิธีการดังนี้คือ

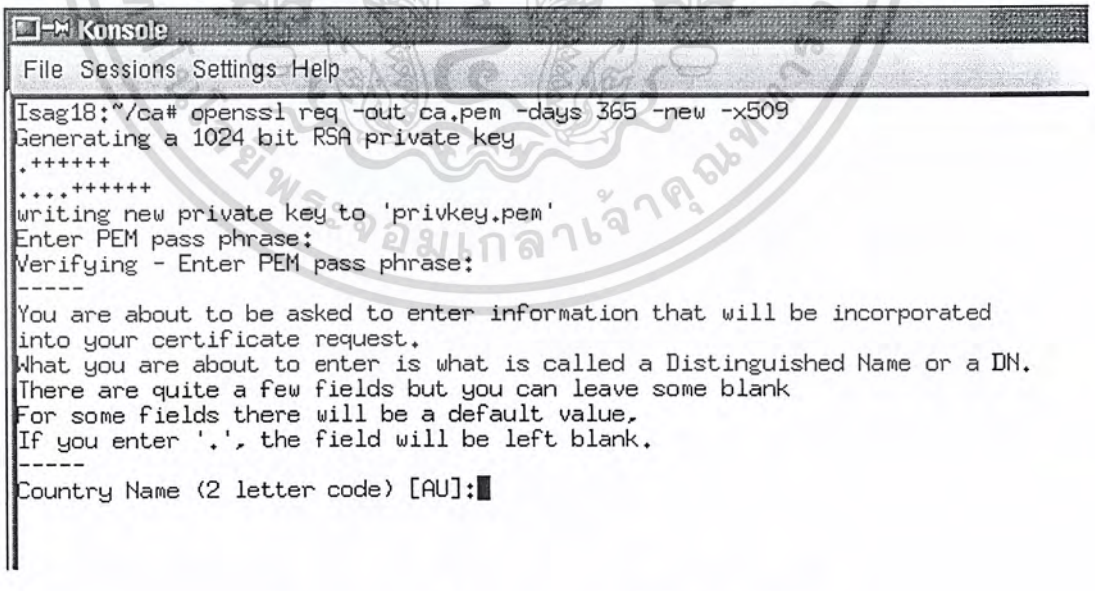
1 สร้าง CA ของเราเองขึ้นมาก่อน

% openssl req –out ca.pem –days 365 –new –x509

คำสั่งนี้จะเป็นคำสั่งที่จะสร้าง CA file คือ ca.pem และ CA key คือ privkey.pem โดยขั้นแรกจะต้องมีการระบุรหัสผ่านของ privkey.pem ก่อนดังรูปที่ 5-8 เมื่อป้อนเสร็จจะเป็นดังรูปที่ 5-9

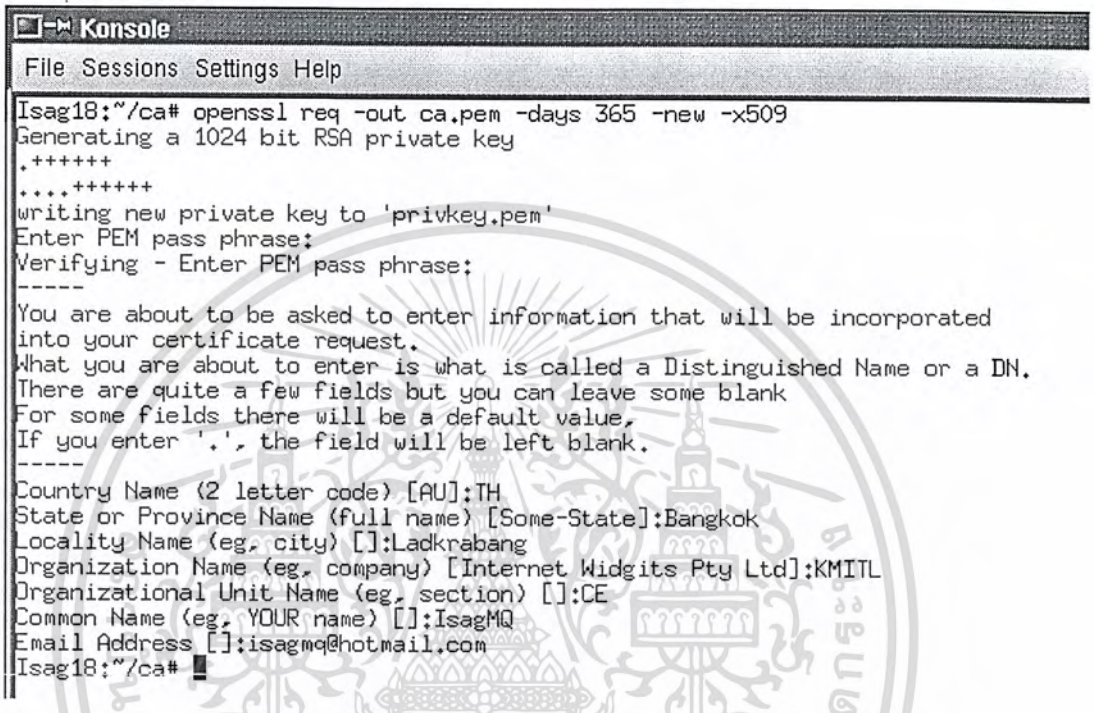


รูปที่ 5-8 แสดงภาพการป้อนรหัสผ่านของกุญแจส่วนตัวของ Internal CA



รูปที่ 5-9 แสดงภาพเมื่อทำการป้อนรหัสผ่านเรียบร้อยแล้ว

ขั้นต่อไปจะเป็นการป้อนข้อมูลส่วนตัวของ Internal CA ดังรูปที่ 5-10 ได้แก่ ชื่อประเทศ, ชื่อจังหวัด, ชื่อเขตหรืออำเภอ, ชื่อบริษัท, ชื่อ, อีเมล์ เมื่อเสร็จแล้วจะได้ไฟล์สองไฟล์คือ ca.pem และ privkey.pem ดังรูปที่ 5-11



รูปที่ 5-10 แสดงภาพการป้อนข้อมูลส่วนตัวของ Internal CA

รูปที่ 5-11 แสดงภาพของไฟล์ที่ได้สร้างขึ้นมาใหม่ 2 ไฟล์

2 สร้างกุญแจ ให้แก่เซิร์ฟเวอร์ แสดงไว้ในรูปที่ 5-12

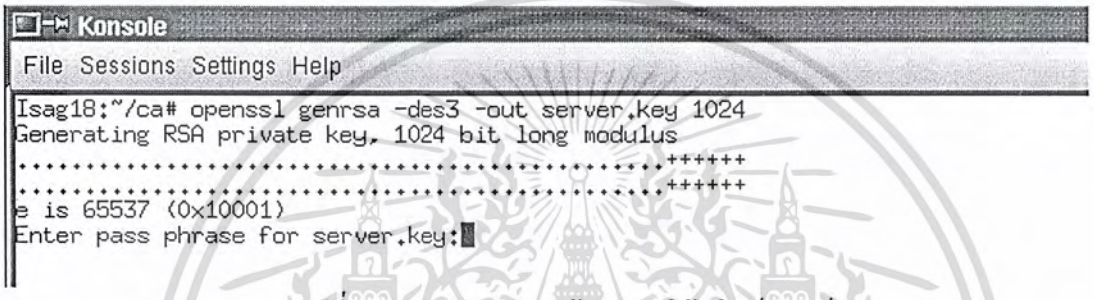
% openssl genrsa -des3 –out sever.key 1024

genrsa เป็นคำสั่งหนึ่งของ openssl ที่จะทำการสร้าง RSA key-pair ขึ้นมา

-des3 เป็นการสร้างกุญแจเดี่ยวที่เป็น 3DES มาทำการเข้ารหัส key-pair ที่จะสร้างขึ้น นั่นคือจะต้องมีการใส่รหัสผ่านให้กับ กุญแจส่วนตัวที่จะสร้างขึ้น

-out server.key เป็นไฟล์ที่จะใช้เก็บ key-pair ที่จะสร้างขึ้น

1024 ขนาดของบิตของ key-pair ที่จะสร้างขึ้น



รูปที่ 5-12 แสดงภาพการสร้างกุญแจให้แก่เซิร์ฟเวอร์

3 สร้างไฟล์ร้องขอการทำใบรับรองสิทธิ์ของเซิร์ฟเวอร์(Server Certificate Request)

% openssl req –key server.key –days365 –new –out server.req

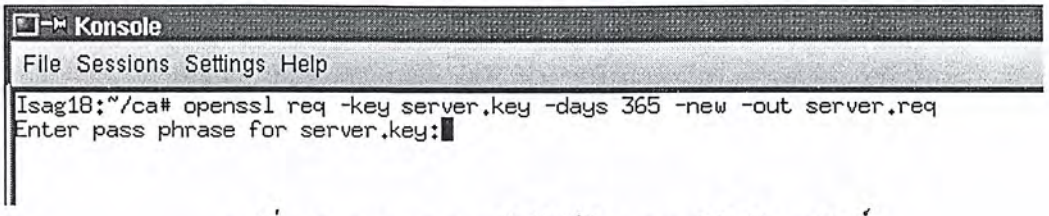
req เป็นคำสั่งหนึ่งของ openssl ที่จะการสร้างข้อความ Certificate Request

-key server.key เป็นการเรียก key-pair มาประกอบการสร้างไฟล์ Certificate Request

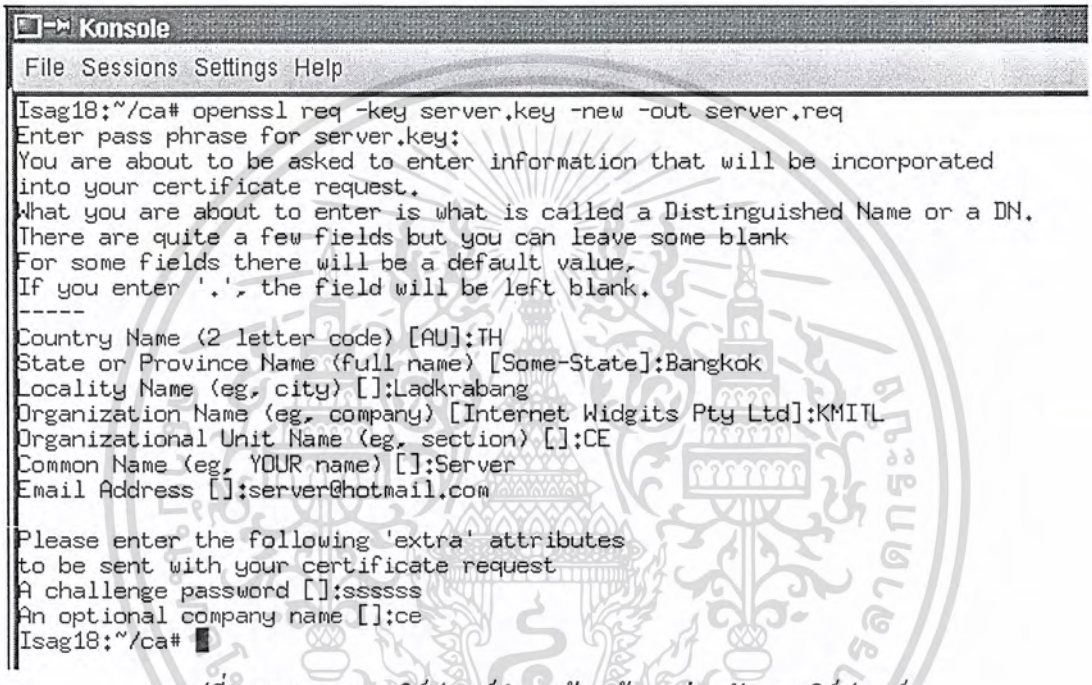
- days 365 เป็นการกำหนดให้สามารถใช้ใบรับรองนี้ได้ในระยะเวลา 1 ปี

- out server.req เป็นไฟล์ที่จะใช้เก็บ Certificate Request ที่จะสร้างขึ้น

เมื่อกด Enter ถ้า key-pair ทำการเข้ารหัสเอาไว้ จะต้องทำการป้อนรหัสผ่านก่อนจึงจะสามารถทำเฟสต่อไปได้ดังรูปที่ 5-13 ถ้ารหัสผ่านถูกต้อง เฟสต่อไปจะเป็นการให้เซิร์ฟเวอร์ทำการป้อนข้อมูลส่วนตัวของเซิร์ฟเวอร์ ดังนี้ ซึ่งเซิร์ฟเวอร์จะทำการป้อนเพียงบางหัวข้อก็ได้ หัวข้อใดที่ไม่ต้องการป้อนก็สามารถข้ามไปได้ ซึ่งหัวข้อทั้งหมดได้แสดงไว้ในรูปที่ 5-14



รูปที่ 5-13 แสดงภาพการสร้างไฟล์ร้องขอการทำใบรับรองสิทธิ์



รูปที่ 5-14 แสดงภาพเซิร์ฟเวอร์ทำการป้อนข้อมูลส่วนตัวของเซิร์ฟเวอร์

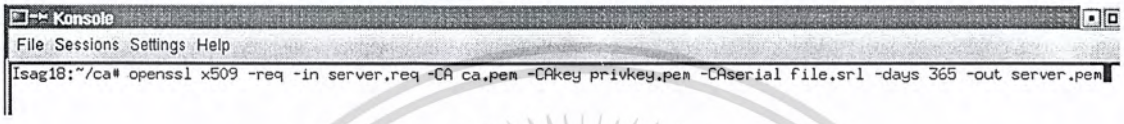
4 ออกใบรับรองสิทธิ์ให้แก่เซิร์ฟเวอร์โดย Internal CA ซึ่งจะแสดงในรูปที่ 5-15

% openssl x509 –req –in server.req –CA ca.pem –CAkey privkey.pem –CAserial file.srl

–days 365 -out server.pem

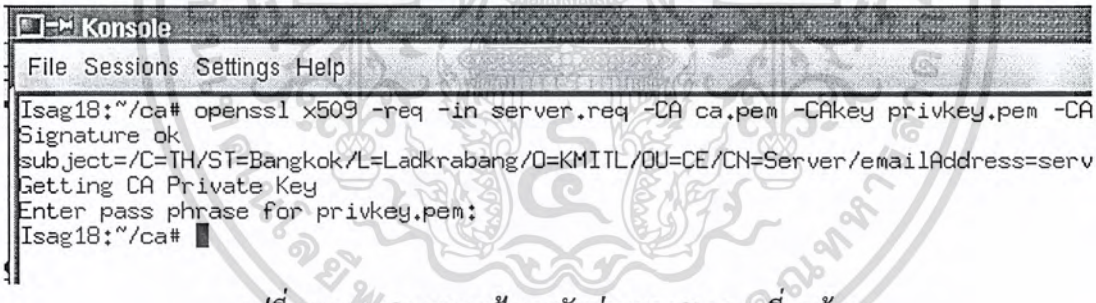
x509	โดยปกติแล้ว SSL โพรโตคอลไม่ได้ถูกออกแบบมาให้สามารถทำการตรวจสอบการพิสูจน์ตนได้ ดังนั้น SSL โพรโตคอลจึงได้นำรูปแบบของ x509 ซึ่งถูกออกแบบมาให้สามารถตรวจสอบการพิสูจน์ตนได้ (X.509 Certificate)
-req –in server.req	จะเป็นการเรียกไฟล์ server.req

-CA ca.pem	จะเป็นการเรียกไฟล์ CA file
-CAkey privkey.pem	จะเป็นการเรียกไฟล์ CA key
-CAserial file.srl	จะเป็นการเรียกไฟล์ file.srl โดยรายละเอียดของไฟล์นี้จะเป็นตัวเลขสองหลักได้แก่ 00,01,10,11
-out server.pem	ใบรับรองสิทธิ์ที่ได้รับมาจะถูกเก็บไว้ในไฟล์ชื่อว่า server.pem



รูปที่ 5-15 แสดงภาพการออกใบรับรองสิทธิ์ให้แก่เซิร์ฟเวอร์โดย Internal CA

เมื่อกด Enter จะมีการแสดงข้อมูลของ CA file ออกมาให้ดูก่อน จากนั้นต้องทำการป้อนรหัสผ่านของ CA key ก่อนจึงจะสามารถทำเฟสต่อไปได้ดังรูปที่ 5-16 ถ้ารหัสผ่านถูกต้องใบรับรองสิทธิ์จะถูกเก็บไว้ในไฟล์ server.pem

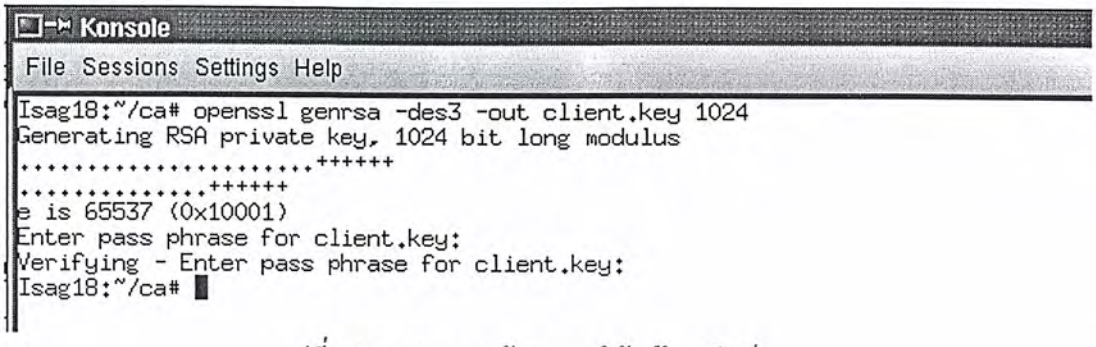


รูปที่ 5-16 แสดงภาพการป้อนรหัสผ่านของ CA key ที่ถูกต้อง

ต่อไปจะเป็นกระบวนการการสร้างใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์ ซึ่งจะมีลักษณะเดียวกันกับเซิร์ฟเวอร์ดังนี้

5 สร้างกุญแจให้แก่ไคลเอ็นต์ แสดงไว้ในรูปที่ 5-17

% openssl genrsa -des3 –out client.key 1024

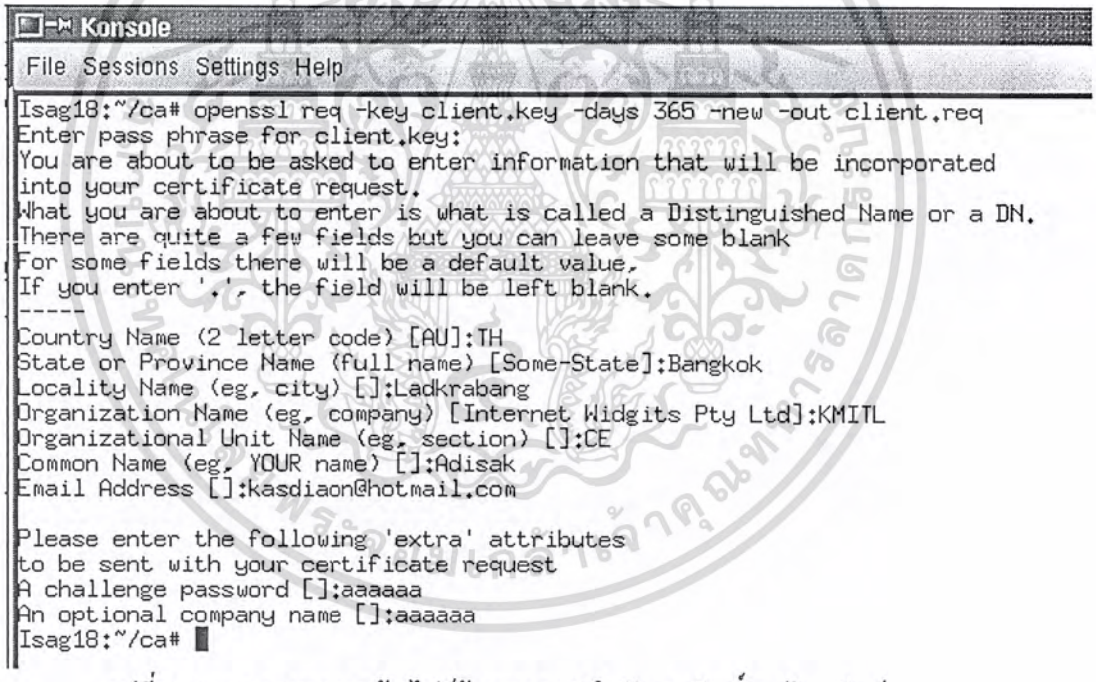


รูปที่ 5-17 แสดงการสร้างกุญแจให้แก่ไคลเอ็นต์

6 สร้างไฟล์ร้องขอการทำใบรับรองสิทธิ์ของไคลเอ็นต์ (Client Certificate Request)

แสดงไว้ในรูปที่ 5-18

% openssl req –key client.key –days365 –new –out client.req

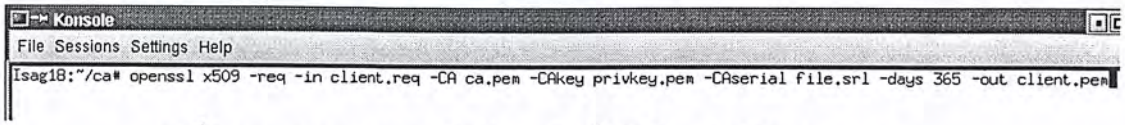


รูปที่ 5-18 แสดงภาพการสร้างไฟล์ร้องขอการทำใบรับรองสิทธิ์ของไคลเอ็นต์

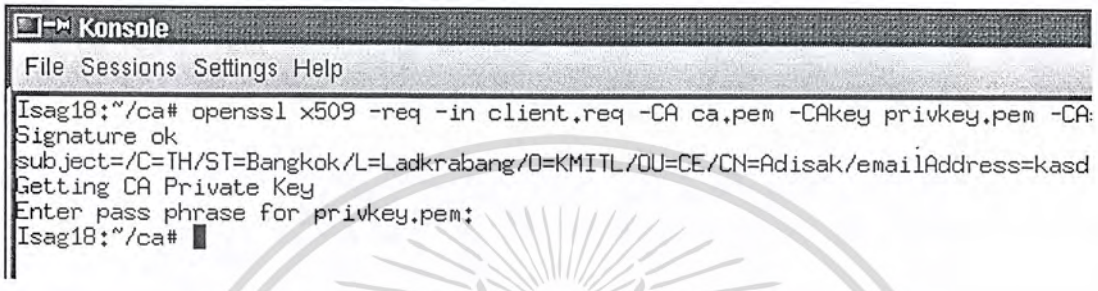
7 ออกใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์โดย Internal CA ซึ่งจะแสดงในรูปที่ 5-19 และ 5-20

% openssl x509 –req –in client.req –CA ca.pem –CAkey privkey.pem –CAserial file.srl

–days 365 -out client.pem



รูปที่ 5-19 แสดงภาพการออกใบรับรองสิทธิ์ให้แก่ไคลเอ็นต์โดย Internal CA



รูปที่ 5-20 แสดงภาพการป้อนรหัสผ่านของ CA key ที่ถูกต้อง

เราสามารถที่จะดูรายละเอียดใบรับรองสิทธิ์ที่ได้รับมาจาก CA ได้โดยพิมพ์

% openssl x509 –text –in client.pem

ผลลัพธ์ที่ได้จะเป็นดังรูปที่ 5-21

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 6 (0x6)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=Th, ST=Bangkok, L=Ladkrabang, O=KMITL, OU=ce, CN=ca_isagmq/ema
        Validity
            Not Before: Jan 30 17:19:37 2004 GMT
            Not After : Jan 29 17:19:37 2005 GMT
        Subject: C=Th, ST=Bankok, L=Ladkrabang, O=KMITL, OU=CE, CN=999999999/ema
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:cc:e9:3c:06:78:24:5d:6c:1c:eb:2f:a3:d7:8c:
                    31:3b:07:e8:5a:34:0a:fd:d2:cd:18:51:d4:2a:a2:
                    d2:3e:2e:8c:b9:eb:37:cd:af:66:4c:d6:2d:e3:de:
                    4c:87:02:13:90:d4:52:bb:5d:0e:ba:ee:c8:75:db:
                    3d:3d:7b:e0:84:97:4b:25:2d:5d:9d:b2:94:90:df:
                    94:be:4b:2f:b0:1d:d5:65:1f:bd:96:3f:25:82:c9:
                    a4:d8:3b:f7:32:d3:a9:65:a5:01:11:0c:1c:d8:32:
                    e6:c1:fe:3b:84:aa:41:63:70:63:d8:69:cd:05:63:
                    ae:d3:5d:d4:98:d0:52:14:37
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        82:e7:22:d6:5b:4d:c5:7f:75:f5:ea:20:0a:55:2a:96:ff:a4:
        dd:79:e7:b7:11:5f:fc:35:58:de:49:5f:54:70:19:10:96:73:
```

รูปที่ 5-21 แสดงภาพของข้อมูลจากไฟล์ client.pem

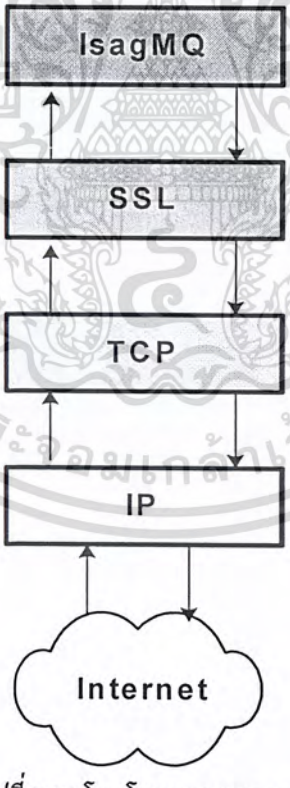
5.7 โพรโตคอลการแลกเปลี่ยนกุญแจสำหรับการเข้ารหัสข้อมูล

จะใช้โพรโตคอลของ SSL ดังได้กล่าวไว้ในบทที่สี่ ซึ่งโพรโตคอลของ SSL มีด้วยกันสามแบบคือ หนึ่งการสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูล สองการสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูลและทำ Server Authentication สามการสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูลทำ Server Authentication และ Client Authentication ซึ่งการสร้าง IsagMQ ได้เลือกใช้แบบที่สาม ซึ่งถือว่ามีความปลอดภัยมากที่สุดและตรงกับวัตถุประสงค์ของโครงงาน ส่วนรายละเอียดการทำงานของแบบที่สามดูได้ในบทที่สี่

5.8 โพรโตคอลเลเยอร์และโพรโตคอลโดยรวมของระบบ (Protocol Layer & IsagMQ Protocol)

5.8.1 โพรโตคอลเลเยอร์ (Protocol Layer)

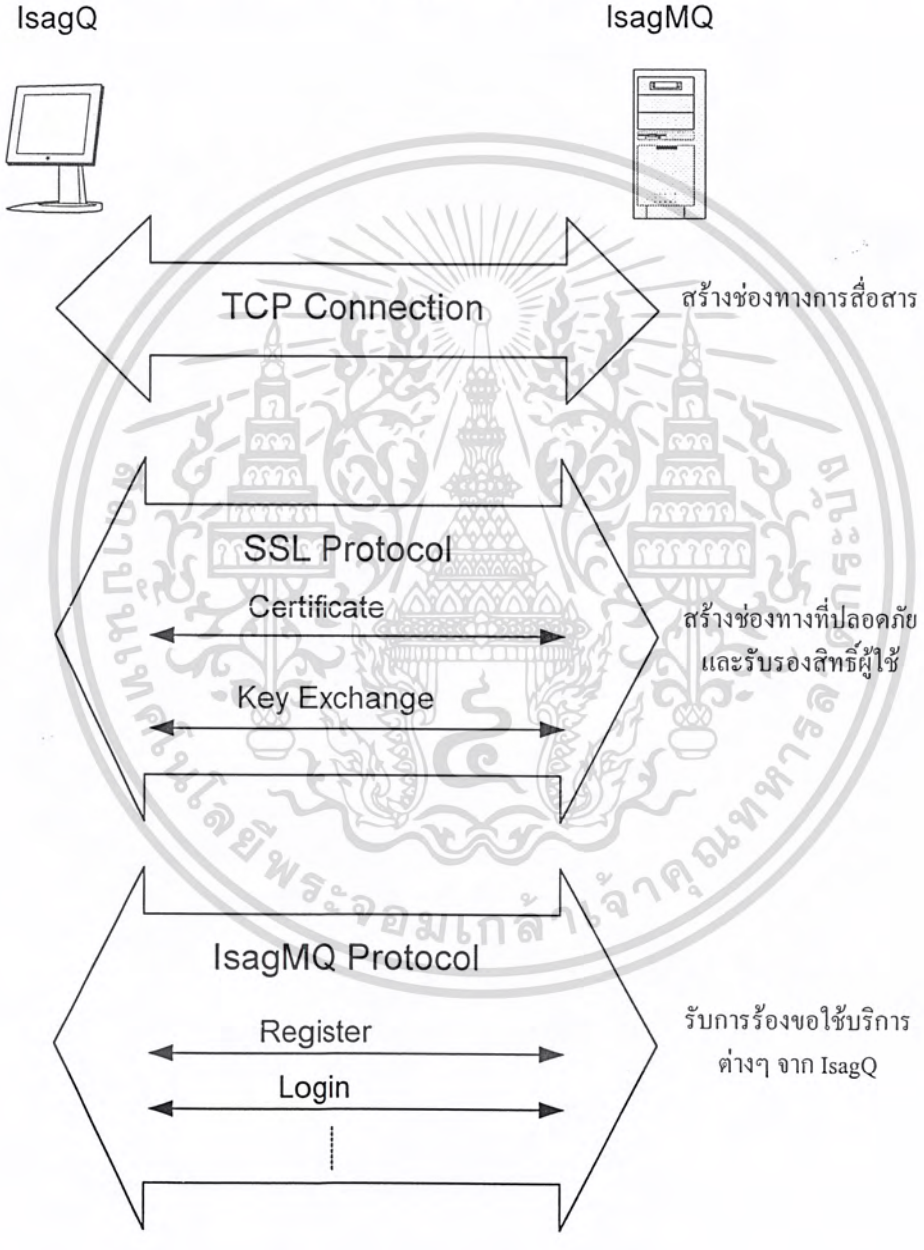
โพรโตคอลเลเยอร์จะเป็นมุมมองในการส่งข้อมูลผ่านเครือข่ายอินเตอร์เน็ตและการส่งข้อมูลนั้นจะต้องผ่านระดับชั้นของโพรโตคอลอะไรบ้าง โดยจะเริ่มใช้โพรโตคอลระดับชั้นแอปริเคชันไประระดับชั้นเน็ตเวิร์ก ดังรูปที่ 5-22



รูปที่ 5-22 โพรโตคอลคอลเลเยอร์

5.8.2 โพรโตคอลโดยรวมของระบบ (IsagMQ Protocol)

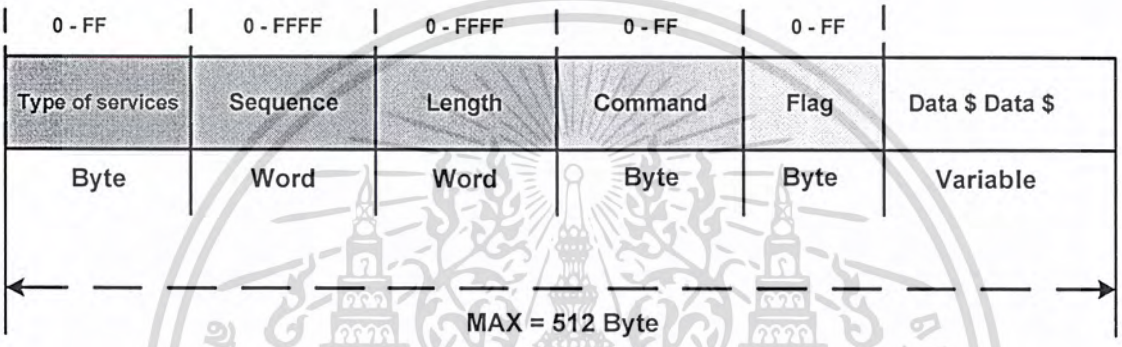
จะเริ่มจากการสร้างการเชื่อมต่อแบบทีซีพีเพื่อเป็นการเชื่อมต่อเบื้องต้น จากนั้นจะทำการ Bind การเชื่อมต่อแบบทีซีพีเป็นการเชื่อมต่อแบบ SSL ที่มีช่องทางการสื่อสารที่ปลอดภัยโดยจะใช้วิธีการของ OpenSSL ตามที่ได้กล่าวในบทก่อนหน้านี้แล้ว และเมื่อได้ช่องทางการสื่อสารที่ปลอดภัยแล้ว จึงจะเป็นโพรโตคอลของ IsagMQ ซึ่งก็คือโพรโตคอลการให้บริการต่างๆ แก่ผู้ใช้ ดังรูปที่ 5-23



รูปที่ 5-23 โพรโตคอลโดยรวมของระบบ

5.9 รูปแบบเฟรม (Frame Format)

การออกแบบรูปแบบเฟรมเพื่อใช้รับส่งข้อมูลระหว่างไคลเอ็นและเซิร์ฟเวอร์นั้น จะพยายามให้พื้นที่ของเฮเดอร์มีน้อยที่สุดโดยใช้เฮเดอร์ที่จำเป็นต่อการ รับส่งข้อมูลเท่านั้น จำกัดขนาดของรูปแบบเฟรมมากที่สุดที่ 512 byte ทั้งนี้เพราะไม่ต้องการให้เฟรมข้อมูลถูกแฟร็กเม็นต์ (fragment) ในระหว่างทาง การเกิดแฟร็กเม็นต์นั้นไม่ก็ให้เกิดประโยชน์ใดมีแต่จะทำให้ส่งข้อมูลได้ช้าลง ทั้งนี้โครงงานนี้ทำงานในเครือข่ายอินเตอร์เน็ตซึ่งเฟรมข้อมูลของอินเตอร์เน็ตมากสุดแค่ 1500 byte เท่านั้นเอง ดังนั้นจึงเป็นการดีถ้าจะใช้ขนาดของรูปแบบเฟรม ที่น้อยกว่าอินเตอร์เน็ต



รูปที่ 5-24 รูปแบบเฟรมของ *IsagMQ*

คำอธิบาย

1 Type of services หมายถึง การระบุว่าต้องการบริการชนิดใด ซึ่งจะมีบริการต่างๆ ดังนี้

0x01 = Register

0x02 = Login

0x03 = Add Contact List

0x04 = Change Nick name

0x05 = Status

0x06 = Error

0x07 = Logout

0x08 = Find Contact List

0x0A = Un register

0x0B = Remove Contact

2 Sequenc หมายถึงหมายเลขลำดับของแพ็กเก็ต

3 Length หมายถึง ความยาวของแพ็กเก็ต ซึ่งจะมีความยาวสูงสุดเท่ากับ 512 byte เพราะเพื่อไม่ต้องการให้แพ็กเก็ต ถูกแฟร็กเม็นต์

4 Command	หมายถึงคำสั่งของแต่ล่ะ Type of services รายละเอียดของคำสั่งต่างๆ อยู่ในหัวข้อที่ 5.11
5 Flag	หมายถึง สภาวะการทำงานพิเศษนอกเหนือจากคำสั่งในข้อ 4
6 Data	หมายถึง ข้อมูลที่ใช้ในการรับส่ง โดยใช้ $ เป็นตัวจบข้อมูลในแต่ล่ะส่วน

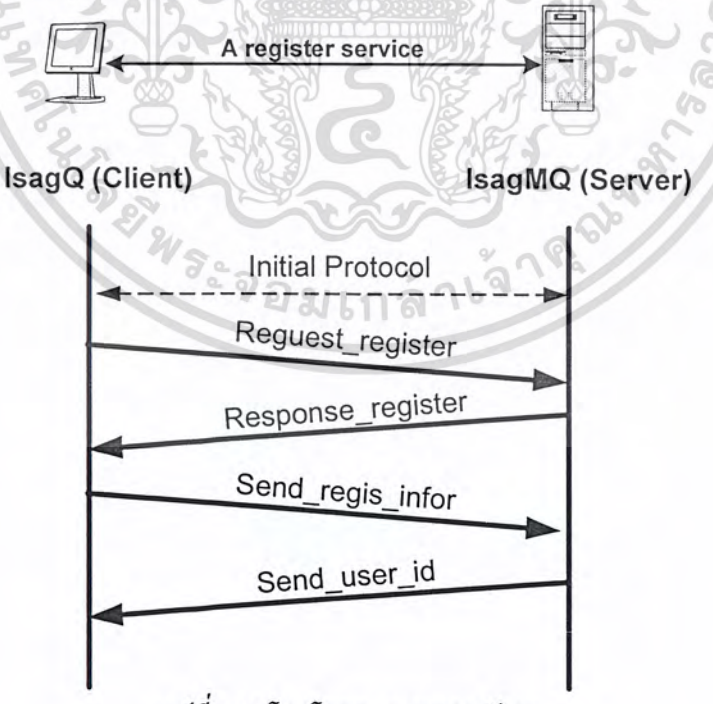
5.10 บริการต่างๆของ IsagMQ และ โพรโตคอล

ไม่ว่าจะเป็น ICQ, MSN หรือ Yahoo Messenger บริการที่แต่ล่ะเซิร์ฟเวอร์มีนั้นมากและเกินความจำเป็นทั้งยังอยู่นอกเหนือจากเป้าหมายของโครงงานนี้ ดังนั้นโครงงานนี้จึงเน้นสร้างบริการที่จำเป็นระบบการรับส่งสารด่วนเท่านั้น ทุกบริการมีโพรโตคอลเป็นไปตามรูปที่ 5-23 จะต้องผ่านโพรโตคอลต่างๆ คือ TCP connection, SSL verify & Exchange key ก่อนที่จะมาถึงโพรโตคอลของการบริการ จะขอเรียกโพรโตคอลเหล่านี้ว่า Initial Protocol แน่นอนว่าข้อมูลที่ใช้ในบริการต่างๆ จะต้องมีการเข้ารหัส โพรโตคอลของการขอบริการมีดังนี้ (คำสั่งขอบริการต่างๆ ดูในตารางที่ 5-1)

5.10.1 บริการลงทะเบียน (Register)

กำหนดให้เป็น **Channel 1 = 0x01**

เป็นบริการที่เอาไว้สำหรับให้ผู้ที่ยังไม่เป็นสมาชิกลงทะเบียนเป็นสมาชิกของ IsagMQ ก่อนที่จะสามารถใช้บริการอื่นๆ ต่อไปได้ มีโพรโตคอลดังรูปที่ 5-25



รูปที่ 5-25 โพรโตคอลการลงทะเบียน

รูปแบบข้อมูล (Data) ของบริการ Channel 1 คำสั่ง 0x02

First_name $ Last_name $ Nick_name $ Sex $ Age $ Email $

รูปแบบของข้อมูลในคำสั่ง send_Regis_infor (0x12) จะต้องเป็นไปตามนี้เท่านั้นจึงจะถือว่าเป็นแพ็กเก็ตที่ถูกต้องและข้อมูลต่างๆ จะต้องถูกหลักเพราะ IsagMQ มีระบบตรวจสอบข้อมูลก่อนนำข้อมูลนั้นไปใช้ ถ้าต่างไปจากนี้ IsagMQ จะไม่ยอมรับ แพ็กเก็ต

ความยาวของแต่ละข้อมูลเป็นดังนี้

First_name ไม่เกิน 64 ตัวอักษร (จะต้องถูกตามหลักการตั้งชื่อ)

Last_name ไม่เกิน 64 ตัวอักษร (จะต้องถูกตามหลักการตั้งชื่อ)

Nick_name ไม่เกิน 256 ตัวอักษร

Sex ไม่เกิน 1 ตัวอักษร (m หรือ f เท่านั้น)

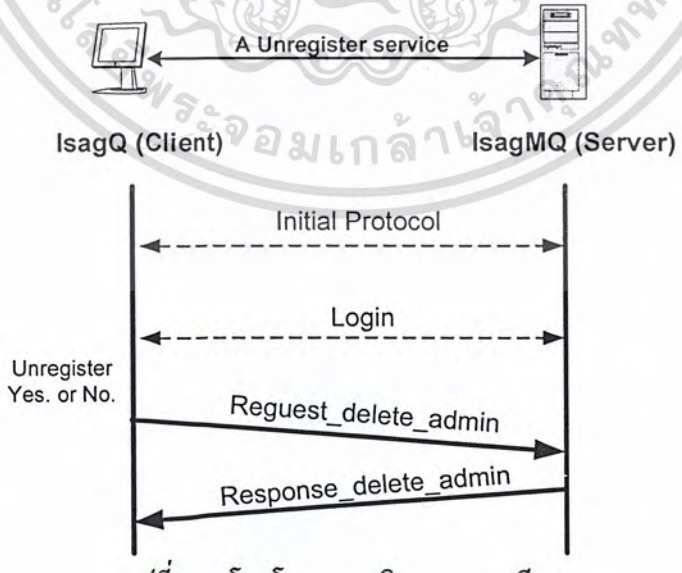
Age ไม่เกิน 2 ตัวอักษร

Email ไม่เกิน 64 ตัวอักษร (จะต้องถูกตามหลักการชื่อ Email)

5.10.2 บริการยกเลิกการลงทะเบียน (Un register)

กำหนดให้เป็น **Channel 10 = 0x0A**

เป็นบริการที่ให้ผู้ที่เป็นสมาชิกของ IsagMQ แล้วยกเลิกการเป็นสมาชิกอย่างถาวร โดยจะทำการลบข้อมูลทุกอย่างของสมาชิกที่ต้องการยกเลิกออกจากระบบจนหมด ถ้าอยากเป็นสมาชิกอีกก็ต้องทำการลงทะเบียนใหม่



รูปที่ 5-26 โพรโตคอลยกเลิกการลงทะเบียน

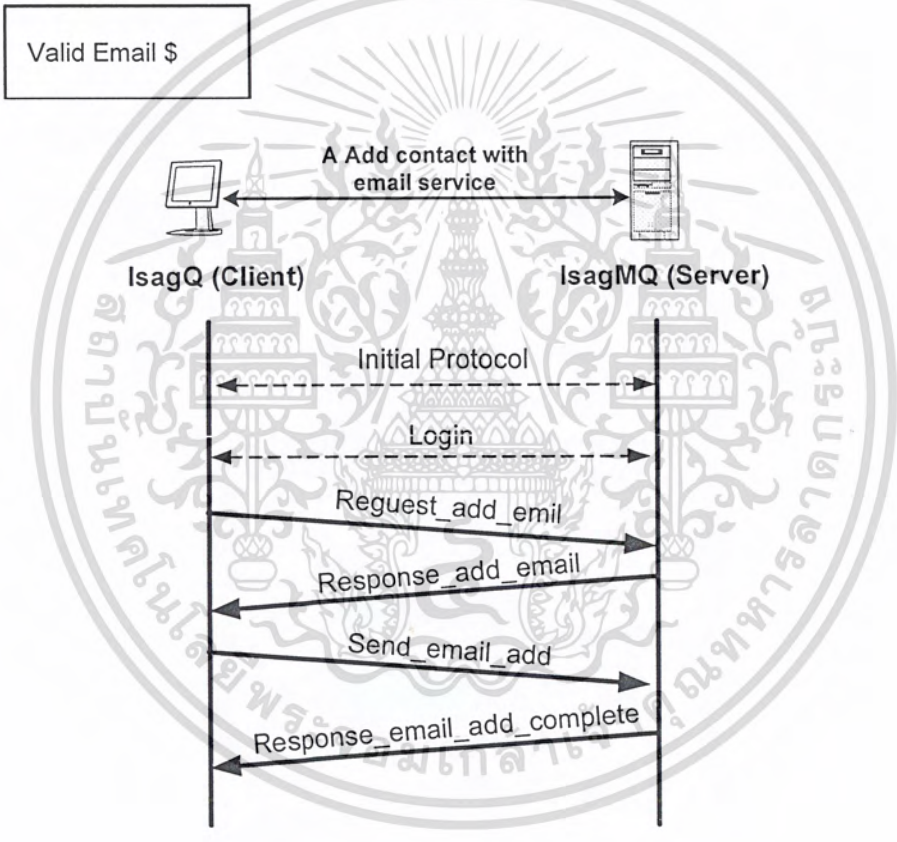
5.10.3 บริการเพิ่มคอนแท็ก (Add Contact)

กำหนดให้เป็น **Channel 3 = 0x03**

เป็นบริการที่ให้ผู้ใช้เพิ่มคอนแท็กลิสต์หรือคู่สนทนาเข้าไปในคอนแท็กลิสต์ของตัวเอง โดยจะสามารถใส่ข้อมูลอ้างอิงถึงคู่สนทานาได้สองแบบ คือ Email หรือ User_id ของคู่สนทนา IsagMQ ก็จะทำการเพิ่มคู่สนทนาในคอนแท็กลิสต์ให้ โพรโตคอลดังรูปที่ 5-27 และรูปที่ 5-28

เพิ่มโดยใช้ **Email**

รูปแบบข้อมูล ของบริการ Channel 3 คำสั่ง 0x36

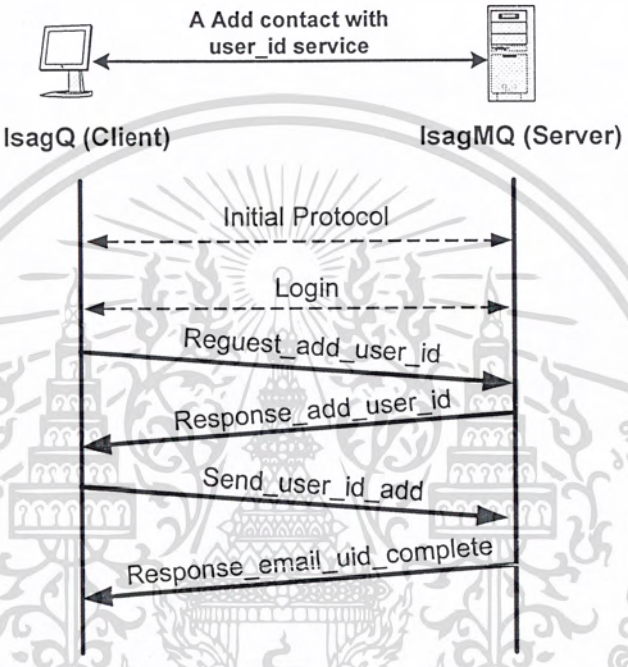


รูปที่ 5-27 โพรโตคอลเพิ่มคู่สนทนาโดยใช้ Email อ้างอิง

เพิ่มโดยใช้ User_id

รูปแบบข้อมูลของบริการ Channel 3 คำสั่ง 0x35

Valid User_id $



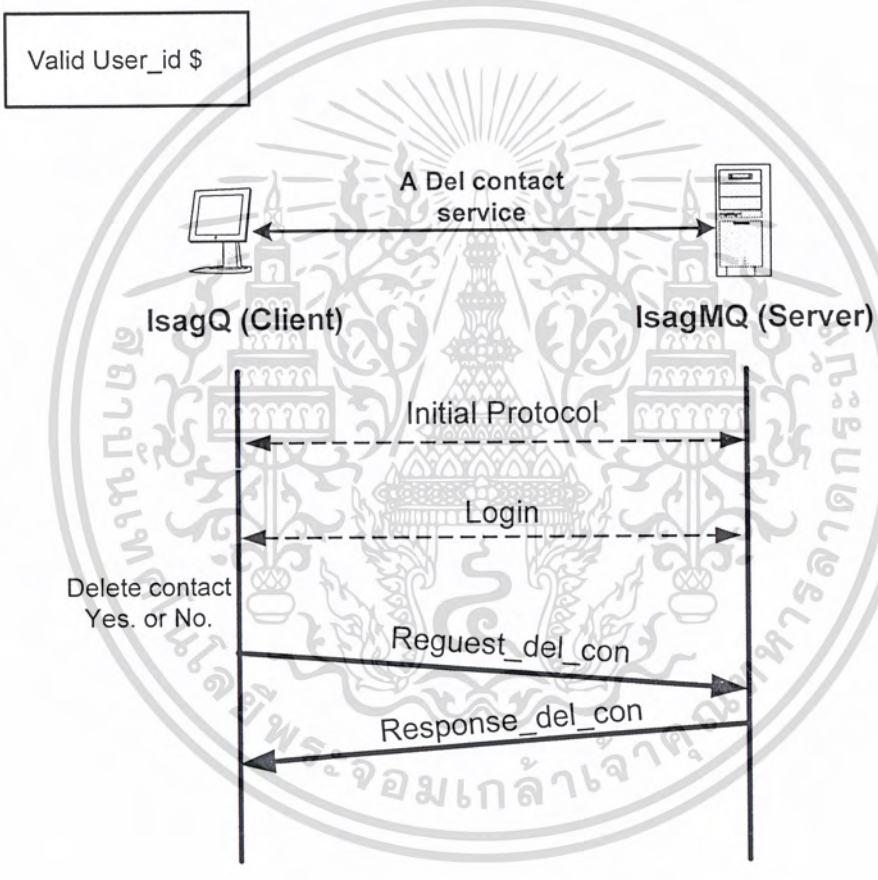
รูปที่ 5-28 โพรโตคอลเพิ่มคู่สนทนาโดยใช้ User_id อ้างอิง

5.10.4 บริการลบคอนแท็ก (Remove Contact)

กำหนดให้เป็น **Channel 11 = 0x0B**

ผู้ใช้จะสามารถลบคู่สนทนาออกจากคอนแท็กลิสต์ได้ถ้าหากใช้บริการนี้ โดยข้อมูลที่ใช้อ้างถึงคู่สนทนาที่ต้องการลบคือ User_id และถ้าหากทำการลบคู่สนทนานั้นออกจากคอนแท็กลิสต์จะไม่สามารถติดต่อกับคู่สนทนานั้นได้อีก ต้องทำการเพิ่มคู่สนทนาใหม่ โพรโตคอลดังรูปที่ 5-29

รูปแบบข้อมูลของบริการ Channel 11 คำสั่ง 0xB1



รูปที่ 5-29 โพรโตคอลลบคู่สนทนา

5.10.5 บริการค้นหาผู้ใช้ที่เป็นสมาชิกของ ISAGMQ

กำหนดให้เป็น **Channel 8 = 0x08**

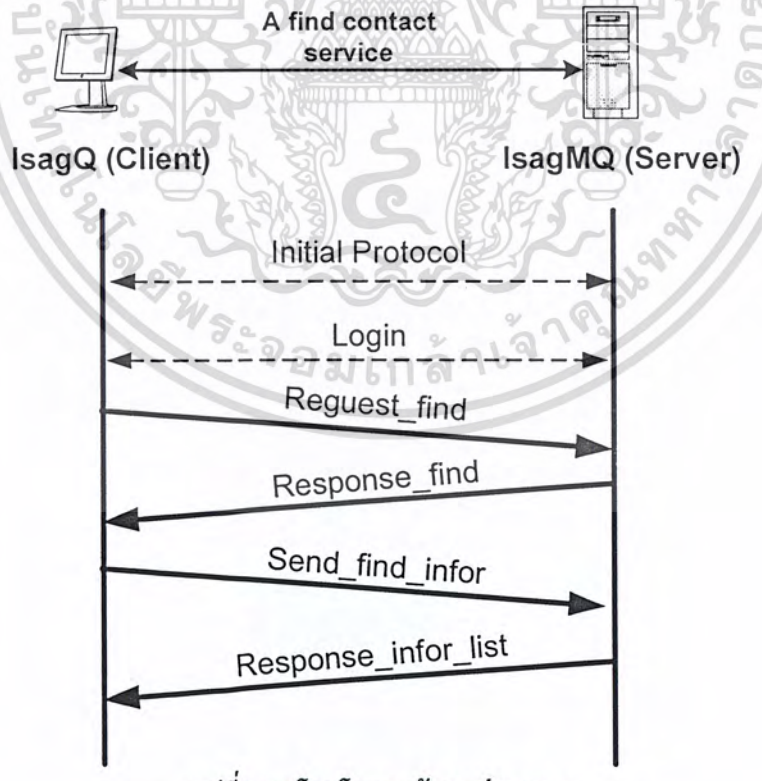
บริการนี้จะให้ ผู้ใช้ใช้ในการค้นหาคู่สนทนาที่เป็นสมาชิกของ IsagMQ โดยจะให้ผู้ใช้ใส่ข้อมูลที่จะอ้างถึงคู่สนทนานั้นและให้เลือกรูปแบบของการค้นหาข้อมูล คือ ถ้าเลือก Option O ระบบจะทำการค้นหาแบบ OR กล่าวคือข้อมูลในการค้นตัวได้ตัวหนึ่งตรงก็จะ List คู่สนทนานั้นออกมา และถ้าเลือก Option A ระบบจะทำการค้าหาแบบ AND กล่าวคือทุกข้อมูลที่ใช้ในค้นตัวทุกตัวต้องตรงก็จะ List คู่สนทนานั้นออกมา ซึ่งหากข้อมูลไหนมีค่าเท่ากับ ' - ' จะไม่นำข้อมูลนั้นมาใช้ในการค้นหา โพรโตคอลดังรูปที่ 5-30

รูปแบบข้อมูลของบริการ Channel 8 คำสั่ง 0x83

Types_of_find $ User_id $ Email $ First_name $ Last_name $

Types_of_find ถ้า = A or a คือ การค้นหาแบบ AND

ถ้า = O or o คือ การค้นหาแบบ OR



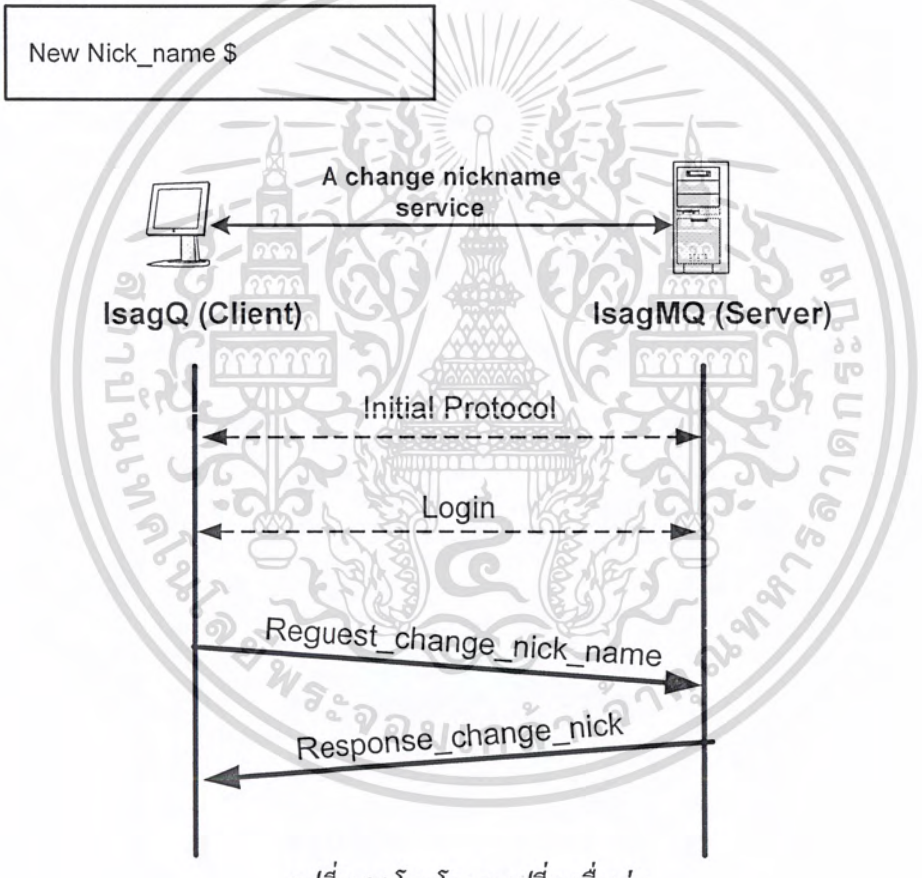
รูปที่ 5-30 โพรโตคอลค้นหาคู่สนทนา

5.10.6 บริการเปลี่ยนชื่อเล่น (Change Nick Name)

กำหนดให้เป็น **Channel 4 = 0x04**

ชื่อเล่นมีจุดประสงค์เพื่อให้ผู้ให้เขียนข้อมความหรือบรรยายความรู้สึกของตังเองขณะที่ Online อยู่ หรือให้เป็นชื่อเล่นจริงก็ได้ ซึ่งจะแสดงไว้ในลิสต์ของคุสนทนาใช้แทนคู่สนทนาคนนั้นเพื่อให้จำได้ง่าย บริการจะทำการเปลี่ยนชื่อเล่นให้ถ้าต้องการ ปรกติแล้วผู้ใช้จะเปลี่ยนกันบ่อยมาก ข้อความหรือชื่อเล่นจะต้องมีความยาวไม่เกิน 255 ตัวอักษร โพรโตคอลดังรูปที่ 5-31

รูปแบบข้อมูลของบริการ Channel 4 คำสั่ง 0x41



รูปที่ 5-31 โพรโตคอลเปลี่ยนชื่อเล่น

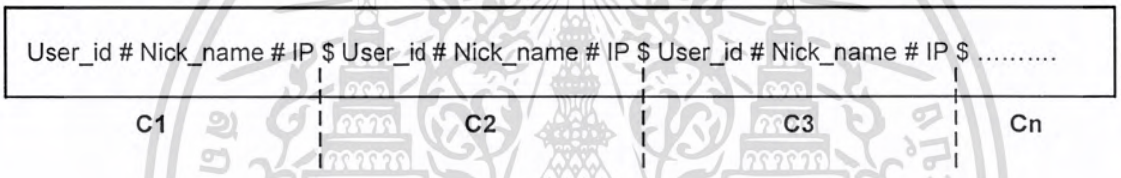
5.10.7 บริการล็อกอินและล็อกเอาต์ (Login/Logout)

ล็อกอิน

กำหนดให้เป็น **Channel 2 = 0x02**

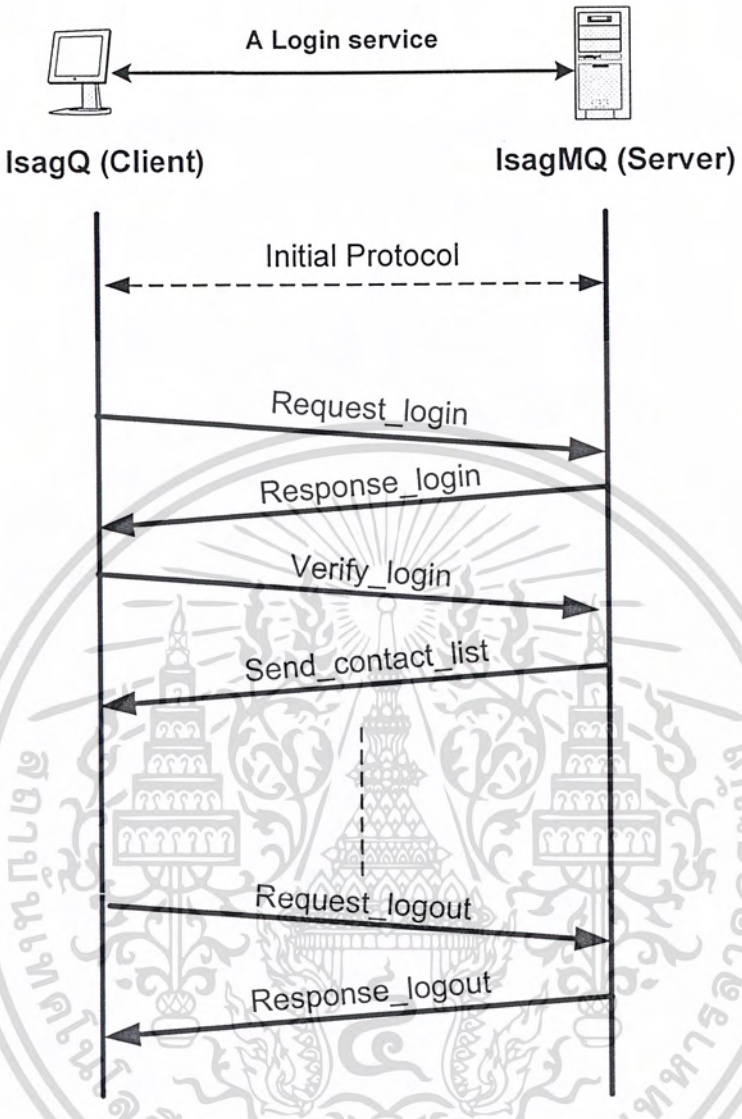
เมื่อผู้ใช้ทำการลงทะเบียนกับ IsagMQ จะสามารถทำการล็อกอินได้กระบวนการล็อกอินของระบบผู้ใช้ไม่จำเป็นต้องส่ง User_id และ Password มาให้กับเซิร์ฟเวอร์เพื่อทำการรับรองตัวเองเข้าสู่ระบบ โดยระบบการล็อกอินของ IsageMQ จะใช้การหลักการของ CA (Certificate Authority) ก็สามารถรู้ได้แล้วว่าผู้ใช้เป็นใคร ซึ่งหลักการนี้จะปลอดภัยมากกว่าการส่ง User_id และ Password ผ่านทางเครือข่ายเสียอีกและถือเป็นข้อดีของระบบ โพรโตคอลของการล็อกอินนั้นคือ Initial Protocol บวกกับการร้องขอคอนแท็กลิสต์ ถ้าหากผู้ใช้ ล็อกอินได้สำเร็จ IsagMQ ก็จะส่งคอนแท็กลิสต์ให้ โพรโตคอลดังรูปที่ 5-32

รูปแบบข้อมูล (**Contact List**) ของบริการ Channel 2 คำสั่ง 0x24



รูปแบบของคอนแท็กลิสต์ที่ได้แสดงดังข้างต้น การแบ่งข้อมูลคือ ข้อมูลของแต่ล่ะผู้ใช้ในคอนแท็กลิสต์จะกั้นด้วย ' $ ' และข้อมูลของ user จะกั้นด้วย ' # '

ถ้าหากผู้ใช้ในคอนแท็กลิสต์ไหนออนไลน์อยู่ตรงหมายเลขไอพี ก็จะแสดงหมายเลขไอพี ปัจจุบันของผู้ใช้คนนั้น และถ้าหากออฟไลน์ จะแสดง ' Offline '



รูปที่ 5-32 โพรโตคอลการล็อกอินและล็อกเอาต์

ล็อกเอาต์

กำหนดให้เป็น **Channel 7 = 0x07**

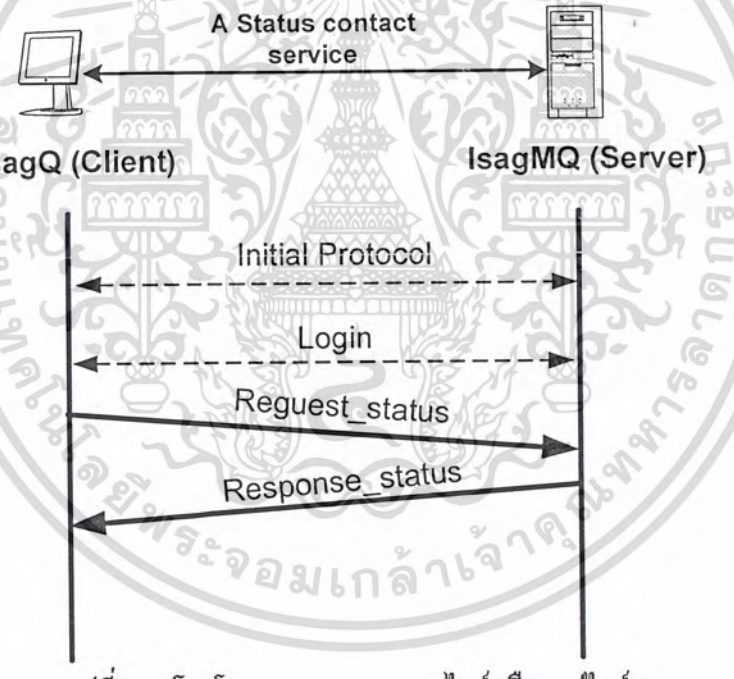
การล็อกเอาต์ ออกจากระบบเพียงแต่ส่ง แพ็กเก็ต ที่มี Channel 0x07 มา IsagMQ ก็จะทำการตัดการเชื่อมต่อของผู้ใช้นั้นออกไปจากระบบ โพรโตคอลดังรูปที่ 5-32

5.10.8 บริการสถานะออนไลน์หรือออฟไลน์ของสมาชิก (ในคอนแท็กลิสต์)

กำหนดให้เป็น **Channel 5 = 0x05**

การปรับปรุงสถานะ (Update Status) คอนแท็กลิสต์ให้แก่ผู้ใช้จะใช้หลักการคือ ให้ไคล์เอ็นต์ส่ง แพ็กเก็ต มาถาม IsagMQ ว่าคอนแท็กลิสต์ของตัวเองมีผู้ใช้ไหนที่ออนไลน์ใหม่หรือใครที่ออฟไลน์ไปแล้วถ้ามี IsagMQ ก็จะส่งคอนแทคท์ลิสต์ที่ปรับปรุงไปให้กับไคล์เอ็นต์จะทำให้ผู้ใช้รู้ตลอดเวลาว่าใครในคอนแท็กลิสต์ออนไลน์หรือออฟไลน์อยู่ระบบของไคล์เอ็นต์อาจจะทำการตั้งเวลาการส่ง แพ็กเก็ต สอบถามขึ้นอยู่กับว่าต้องการเรียลไทม (Real Time) มากแค่ไหน แต่ไม่ควรจะให้เร็วกว่า 5 วินาทีต่อ แพ็กเก็ต เพราะจะทำให้ IsagMQ ทำงานหนักเกินไป

การทำระบบออนไลน์หรือออฟไลน์ของแต่ละสมาชิกนี้ระบบรับส่งสารด่วนบางตัวเช่น MSN จะตั้งเซิร์ฟเวอร์สำหรับทำบริการนี้โดยเฉพาะเลยเพื่อไม่ให้เซิร์ฟเวอร์ทำงานหนักเกินไป โพรโตคอลดังรูปที่ 5-33 ช่วงที่ Response_status เซิร์ฟเวอร์ก็จะส่งคอนแทคท์ลิสต์ที่ Update ให้กับไคล์เอ็นต์



รูปที่ 5-33 โพรโตคอลสถานะการออนไลน์หรือออฟไลน์

5.11 คำสั่งในการขอใช้บริการ IsagMQ

ตารางที่ 5-1 คำสั่งในการขอใช้บริการ IsagMQ

Type of services	รหัสคำสั่ง	ชื่อคำสั่ง	รายละเอียด
0x01	0x10	Request_register	-ไคลเอ็นร้องขอใช้บริการลงทะเบียนสมาชิกใหม่
	0x11	Response_register	-เซิร์ฟเวอร์ตอบรับการขอลงทะเบียนสมาชิกใหม่
	0x12	Send_regis_infor	-ไคลเอ็นส่งข้อมูลเพื่อลงทะเบียน
	0x13	Send_user_id	-เซิร์ฟเวอร์ส่ง user_id ให้กับไคลเอ็นเมื่อลงทะเบียนสมบูรณ์
0x02	0x21	Request_login	-ไคลเอ็นร้องขอใช้บริการล็อกอิน
	0x22	Response_login	-เซิร์ฟเวอร์ตอบรับการล็อกอิน
	0x23	Verify_login	-ไคลเอ็นยืนยันว่าต้องการล็อกอิน
	0x24	Send_contact_list	-เซิร์ฟเวอร์ส่งคอนแท็กลิสต์ให้กับไคลเอ็นที่ล็อกอินอย่างถูกต้อง
0x03	0x32	Request_add_email	-ไคลเอ็นร้องขอใช้บริการแอดคอนแท็กลิสต์โดยใช้ Email Address
	0x34	Response_add_email	-เซิร์ฟเวอร์ตอบรับการแอดคอนแท็กลิสต์โดยใช้ Email Address
	0x36	Send_email_add	-ไคลเอ็นส่ง Email Address เพื่อให้เซิร์ฟเวอร์แอดคอนแท็กลิสต์
	0x38	Response_add_email_complete	-เซิร์ฟเวอร์ยืนยันว่าแอดคอนแท็กลิสต์สมบูรณ์
	0x31	Request_add_userID	-ไคลเอ็นร้องขอใช้บริการแอดคอนแท็กลิสต์โดยใช้ User_id
	0x33	Response_add_ userID	-เซิร์ฟเวอร์ตอบรับการแอดคอนแท็กลิสต์โดยใช้ User_id
	0x35	Send_ userID_add	-ไคลเอ็นส่ง User_id เพื่อให้เซิร์ฟเวอร์แอดคอนแท็กลิสต์

	0x37	Response_add_userID_complete	-เซิร์ฟเวอร์ยืนยันว่าแอดคอนแท็กลิสต์สมบูรณ์
0x04	0x41	Request_change_nick_name	-ไคลเอ็นร้องขอใช้บริการเปลี่ยนชื่อเล่นโดยส่งชื่อเล่นใหม่มาด้วย
	0x42	Response_change_nick_name	-เซิร์ฟเวอร์ยืนยันว่าเปลี่ยนชื่อเล่นสมบูรณ์
0x05	0x51	Request_status_contact	-ไคลเอ็นร้องขอสาถานะการ Online/Offline ของคอนแท็กลิสต์
	0x52	Response_status	-เซิร์ฟเวอร์แจ้งสาถานะคอนแท็กลิสต์
0x06	-	-	-
0x07	0x71	logout	-
0x08	0x81	Request_find	-ไคลเอ็นร้องขอใช้บริการค้นหาคู่สนทนา
	0x82	Response_find	-เซิร์ฟเวอร์ตอบรับการค้นหาคู่สนทนา
	0x83	Send_find_infor	-ไคลเอ็นส่ง User_id ที่ต้องการค้นหา
	0x84	Response_infor_list	-เซิร์ฟเวอร์ส่งข้อมูลที่ค้นหาได้ให้กับไคลเอ็น
0x0A	0xA1	Request_delete_admin	-ไคลเอ็นร้องขอการยกเลิกเป็นสมาชิก
	0xA2	Response_delete_admin	-เซิร์ฟเวอร์ทำการยืนยันว่าได้ทำการยกเลิกการเป็นสมาชิกแล้ว
0x0B	0xB1	Request_delete_contact	-ไคลเอ็นร้องขอการลบคอนแท็กพร้องส่ง User_id ของคอนแท็กลิสต์ที่ต้องการลบ
	0xB2	Response_delete_contact	-เซิร์ฟเวอร์ทำการยืนยันว่าได้ทำการลบคอนแท็กที่ต้องการแล้ว

5.12 การตรวจสอบเฮเดอร์และข้อมูล

การทำให้เซิร์ฟเวอร์มี **CIA (Confidentiality, Integrity, Availability)** ดังได้กล่าวในบทที่ 4 นั้นจะต้องมีการตรวจสอบรูปแบบเฟรมที่ส่งมาจากไคลเอ็นว่าถูกต้องตามรูปแบบที่กำหนดไว้หรือไม่ ถ้าหากถูกต้องเซิร์ฟเวอร์ก็จะรับเฟรมนั้นเข้ามาพิจารณาว่าเป็นการร้องขอบริการอะไร และถ้าหากบริการนั้นมีการส่งข้อมูลมากับเฟรมด้วยเซิร์ฟเวอร์จะต้องทำการตรวจสอบข้อมูลนั้นก่อนนำไปใช้ทุกครั้งเพราะเป็นวิธีหนึ่งที่ป้องกัน Buffer Overflow ได้ แต่ถ้าหากรูปแบบเฟรมไม่ตรงกับรูปแบบที่วางเอาไว้เซิร์ฟเวอร์ก็จะตัดการร้องขอ (Request) นั้นออกไปเลยจุดนี้จะทำให้ระบบมี Availability มากขึ้นด้วย

ระบบการตรวจสอบข้อมูลได้ใช้หลักการดังนี้

1. ตรวจสอบลำดับของข้อมูลแต่ละชนิดที่ส่งมาเฟรมเดียวกันว่าถูกต้องหรือไม่
2.. ตรวจสอบความยาวของข้อมูลที่ระบบได้จำกัดเอาไว้
3. ตรวจสอบขนาดข้อมูลว่าตรงตามขีดจำกัดของข้อมูลแต่ละชนิดหรือไม่

5.13 การออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลจะมี 2 ตารางคือ user, contact_list แต่ล่ะตารางจะเก็บข้อมูลที่จำเป็นในการให้บริการแก่ไคลเอ็นต์เท่านั้น จากข้อมูลแต่ล่ะตารางสามารถให้บริการครบตามความต้องการที่กำหนดไว้ โดยการออกแบบยึดตามหลักการนี้

ความถูกต้องของข้อมูล ทั้ง 2 ตารางข้อมูลไม่ขึ้นต่อกัน ดังนั้นการทำ update, insert, delete จะไม่มีปัญหาต่อความถูกต้องของข้อมูล จะมี USER_ID เป็น key ที่ใช้ในการเชื่อมโยงสองตารางเข้าด้วยกัน แต่ก็ไม่มีปัญหาเพราะ USER_ID ไม่มีการ update อยู่แล้ว

ความซ้ำซ้อนของข้อมูล ถือว่าไม่มีความซ้ำซ้อนเพราะจะมีการเก็บข้อมูลคนล่ะประเภท และการใช้งานส่วนใหญ่จะเรียกทีล่ะ ตาราง

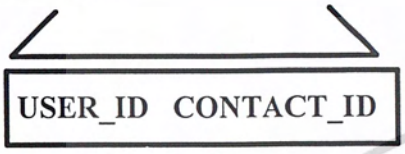
user

USER_ID	FIRST_NAME	LAST_NAME	NICK_NAME	SEX	AGE	EMAIL

1. USER_ID (key) Type: Integer Unique Length: 9 digit Not null Default: MAX Integer	**4. SEX** Type: enum Values: 'M', 'F ' or 'NONE' only Not null Default: 'M'
2. FIRST_NAME Type: varcharacters Length: 64 characters Not null Default: 'NONE_NAME'	**5. AGE** Type: Integer Length: 2 digit Not null Default: 0
3. LAST_NAME Type: varcharacters	**6. EMAIL** Type: varcharacters Length: 64 characters

Length: 64 characters Not null Default: 'NONE_NAME'	Not null Default: 'NONE_EMAIL'

contact_list



1. USER_ID	2. CONTRACT_ID
Type: Integer Length: 9 digit Not null Default: MAX Integer เป็น foreign key จากตาราง user	Type: Integer Length: 9 digit Not null Default: MAX Integer

บทที่ 6
การทดสอบโปรแกรม IsagMQ

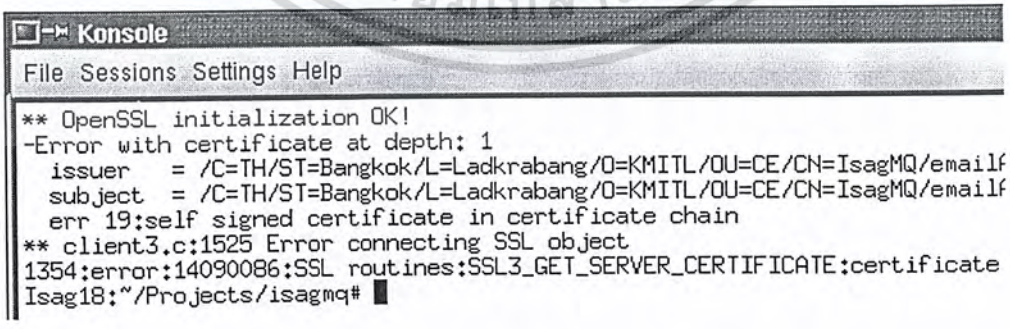
การทดสอบโปรแกรม IsagMQ จะทำการทดสอบเพื่อแสดงให้เห็นว่าได้ออกแบบและพัฒนา IsagMQ ตามวัตถุประสงค์คือ หนึ่งมีการรับประกันว่าผู้ที่จะสามารถใช้บริการของ IsagMQ ได้นั้นจะต้องผ่านการพิสูจน์ตนอย่างถูกต้องแล้วเท่านั้น สองการสื่อสารระหว่างไคล์เอ็นและเซิร์ฟเวอร์จะต้องมีการเข้ารหัสและถอดรหัสข้อมูลที่ปลอดภัย สามมีการบริการที่สามารถให้บริการรับส่งสารด่วนได้ สำหรับการดักจับข้อมูลเพื่อการศึกษาโพรโตคอลและการเข้ารหัสข้อมูลนั้นจะใช้โปรแกรมประเภทปลอมแปลง (Spoofing) เป็นเครื่องมือ โดยในการทดสอบนี้จะใช้โปรแกรมชื่อ Ethereal ดักจับข้อมูล

6.1 ทดสอบการ Certificate ระหว่าง IsagQ และ IsagMQ

เป็นการทดสอบว่าก่อนที่จะใช้บริการจาก IsagMQ จะต้องได้รับการรับรองสิทธ์อย่างถูกต้องก่อนถึงจะสามารถขอใช้บริการได้ ถ้าหากการรับรับรองสิทธ์ไม่ถูกต้องจะไม่ให้ใช้บริการใดเลย หรือมีใบรับรองสิทธ์อย่างถูกต้องแต่ถ้าหากใบรับรองสิทธ์นั้นหมดอายุก็ไม่อาจใช้ได้ การทดสอบดังนี้

6.1.1 IsagQ และ IsagMQ ไม่ได้ใช้ใบรับรองสิทธิ์ (Certificate) จาก Root CA (Server) เดียวกัน

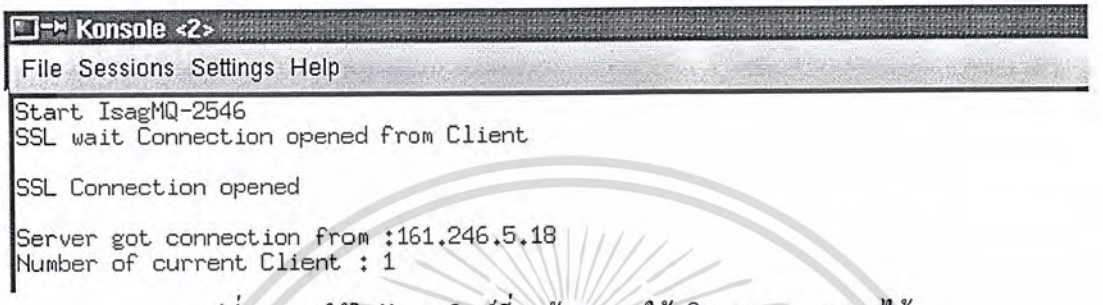
ถ้าหาก IsagQ ไม่ได้ใช้ใบรับรองสิทธ์ (client.cert) ที่ได้จาก IsagMQ รับรองให้แน่นอนว่าการ การพิสูจน์ตน (Authenticate) ยอมไม่ผ่านและไม่สามารถใช้บริการจาก IsagMQ ได้ หรือการที่ IsagQ ไม่มีกุญแจสาธารณะ (Public key) ของ Root CA (IsagMQ) ก็จะการพิสูจน์ตนไม่ผ่านเหมือนกัน จะเกิดข้อผิดพลาด (error) ที่ขึ้นดังรูปที่ 6-1



รูปที่ 6-1 ข้อผิดพลาด จากการใช้ใบรับรองสิทธ์ไม่ถูกต้อง

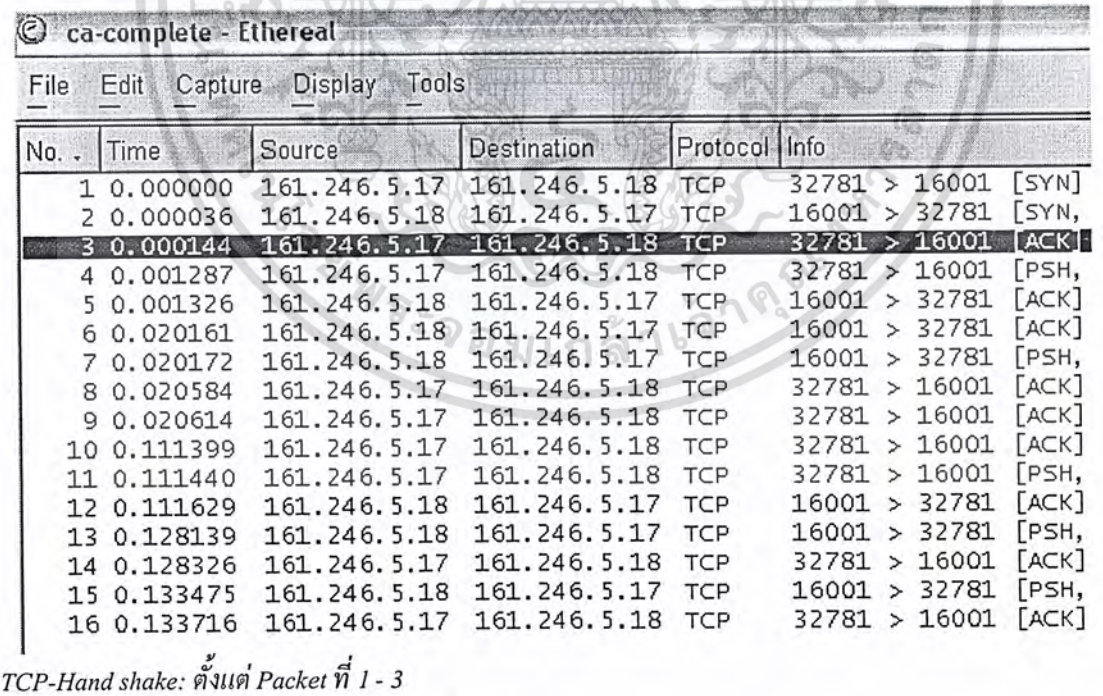
6.1.2 IsagQ และ IsagMQ ได้ใช้ใบรับรองสิทธิ์ (Certificate) จาก Root CA เดียวกัน

ถ้าหาก IsagQ ได้ใช้ใบรับรองสิทธ์ (client.cert) ที่ได้จาก IsagMQ รับรองให้แน่นอนว่าการ การพิสูจน์ตน (Authenticate) ยอมผ่านและสามารถใช้บริการจาก IsagMQ ได้ และ IsagQ มีกุญแจสาธารณะของ Root CA (IsagMQ) ด้วย ดังรูปที่ 6-2 แสดงการรับการร้องขอการขอใช้บริการจาก IsagQ



รูปที่ 6-2 การใช้ใบรับรองสิทธ์ที่ถูกต้องจะขอใช้บริการจาก IsagMQ ได้

เมื่อเราใช้โปรแกรม Ethereal ดักจับการทำงานของโพรโตคอลแล้ว จะเห็นว่าเป็นไปตามโพรโตคอลของ SSL แบบสร้างการเชื่อมต่อแบบ SSL โพรโตคอลโดยใช้การเข้ารหัสข้อมูลทำ การพิสูจน์ตนของเซิร์ฟเวอร์ (Server Authentication) และ การพิสูจน์ตนของไคลเอ็นต์ (Client Authentication) ดังรูปที่ 6-3



No.	Time	Source	Destination	Protocol	Info
1	0.000000	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [SYN]
2	0.000036	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [SYN,
3	0.000144	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]
4	0.001287	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [PSH,
5	0.001326	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [ACK]
6	0.020161	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [ACK]
7	0.020172	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [PSH,
8	0.020584	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]
9	0.020614	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]
10	0.111399	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]
11	0.111440	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [PSH,
12	0.111629	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [ACK]
13	0.128139	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [PSH,
14	0.128326	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]
15	0.133475	161.246.5.18	161.246.5.17	TCP	16001 > 32781 [PSH,
16	0.133716	161.246.5.17	161.246.5.18	TCP	32781 > 16001 [ACK]

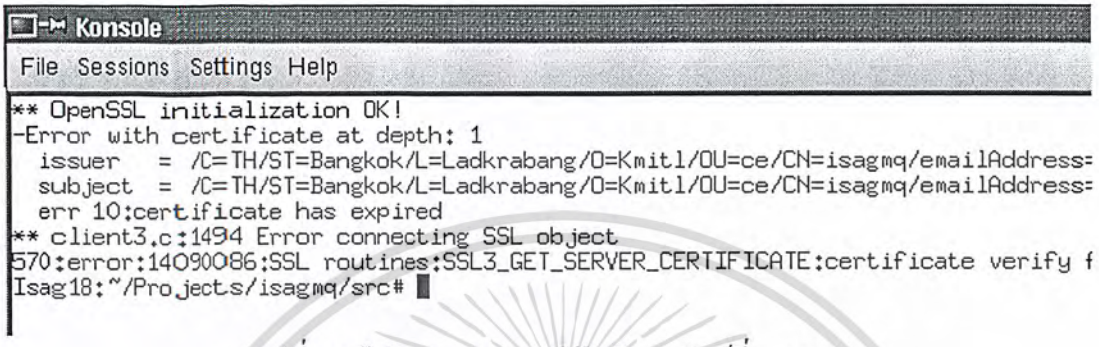
TCP-Hand shake: ตั้งแต่ Packet ที่ 1 - 3

SSL-Protocol: ตั้งแต่ Packet ที่ 4 – 16 เป็นการทำ CA

รูปที่ 6-3 การดักจับการทำของ SSL โพรโตคอล

6.1.3 IsagQ และ IsagMQ ใช้ใบรับรองสิทธิ์ที่หมดอายุการใช้งานแล้ว

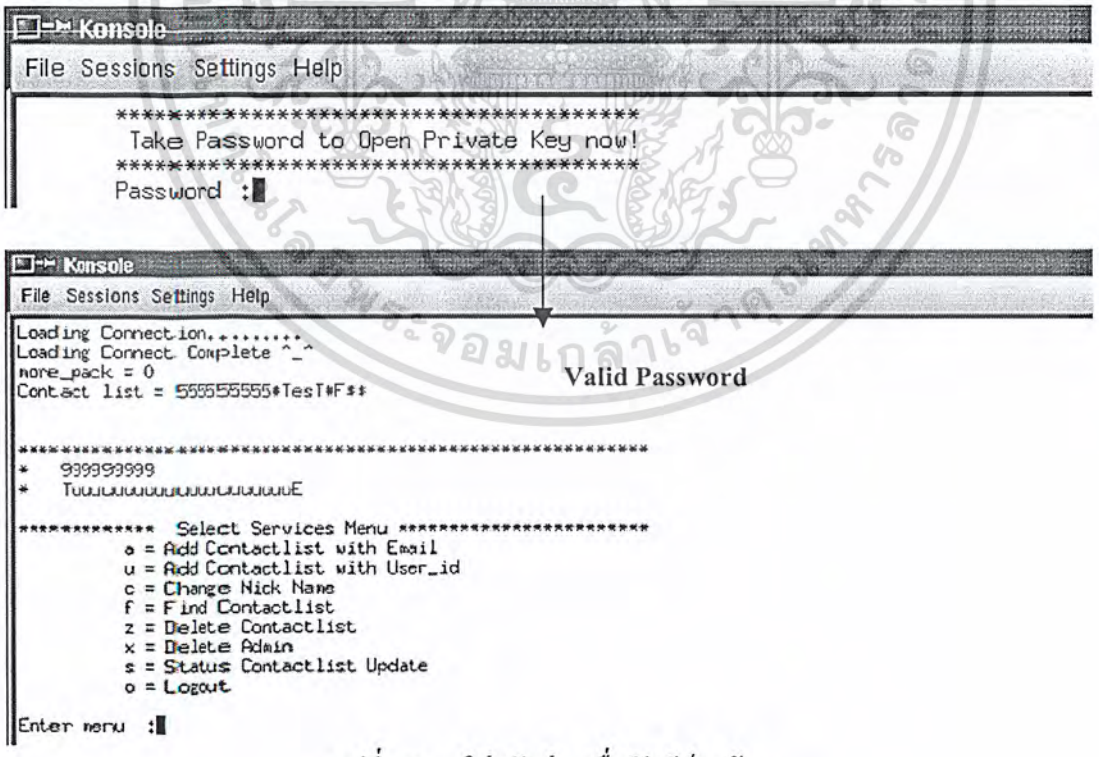
การใช้ ใบรับร องสิทธ์ (client.cert) ที่หมดอายุการใช้งานนั้นก็จะการพิสูจน์ตนไม่ผ่าน เกิดข้อผิดพลาด ดังรูปที่ 6-4



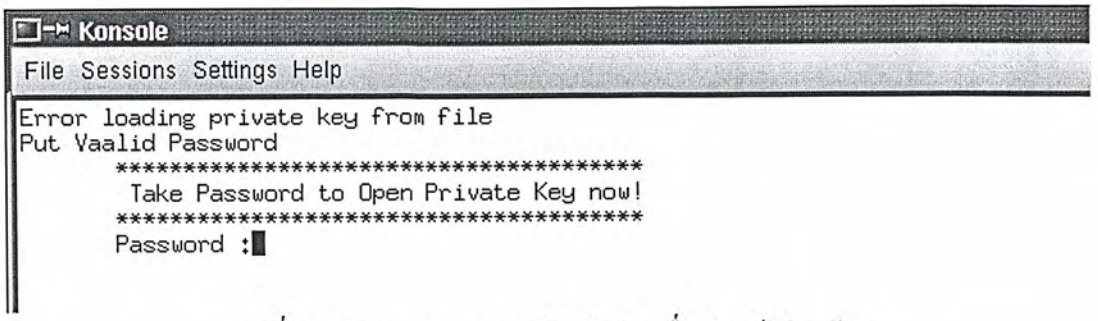
รูปที่ 6-4 ข้อผิดพลาด จากการใช้ใบรับรองสิทธ์ที่หมดอายุ

6.2 ทดสอบการใส่รหัสผ่าน เพื่อใช้ในการเปิดกุญแจส่วนตัว (Private Key) ของ IsagQ

การสร้างกุญแจส่วนตัว (Private key) ของ IsagQ คือ client.key นั้นต้องสร้างจะต้องมีการใส่รหัสผ่านด้วย และแน่นอน ว่าหากทำการเรียกใช้ client.key จะรู้รหัสผ่านถึงจะมีสิทธิ์ในการใช้และสามารถเริ่มการทำงานของ IsagQ ได้ดังรูปที่ 6-5 แต่ถ้าหากใส่รหัสผ่านที่ไม่ถูกต้องจะเกิดข้อผิดพลาดดังรูปที่ 6-6



รูปที่ 6-5 การใส่รหัสผ่านเพื่อเปิดคีย์ถูกต้อง



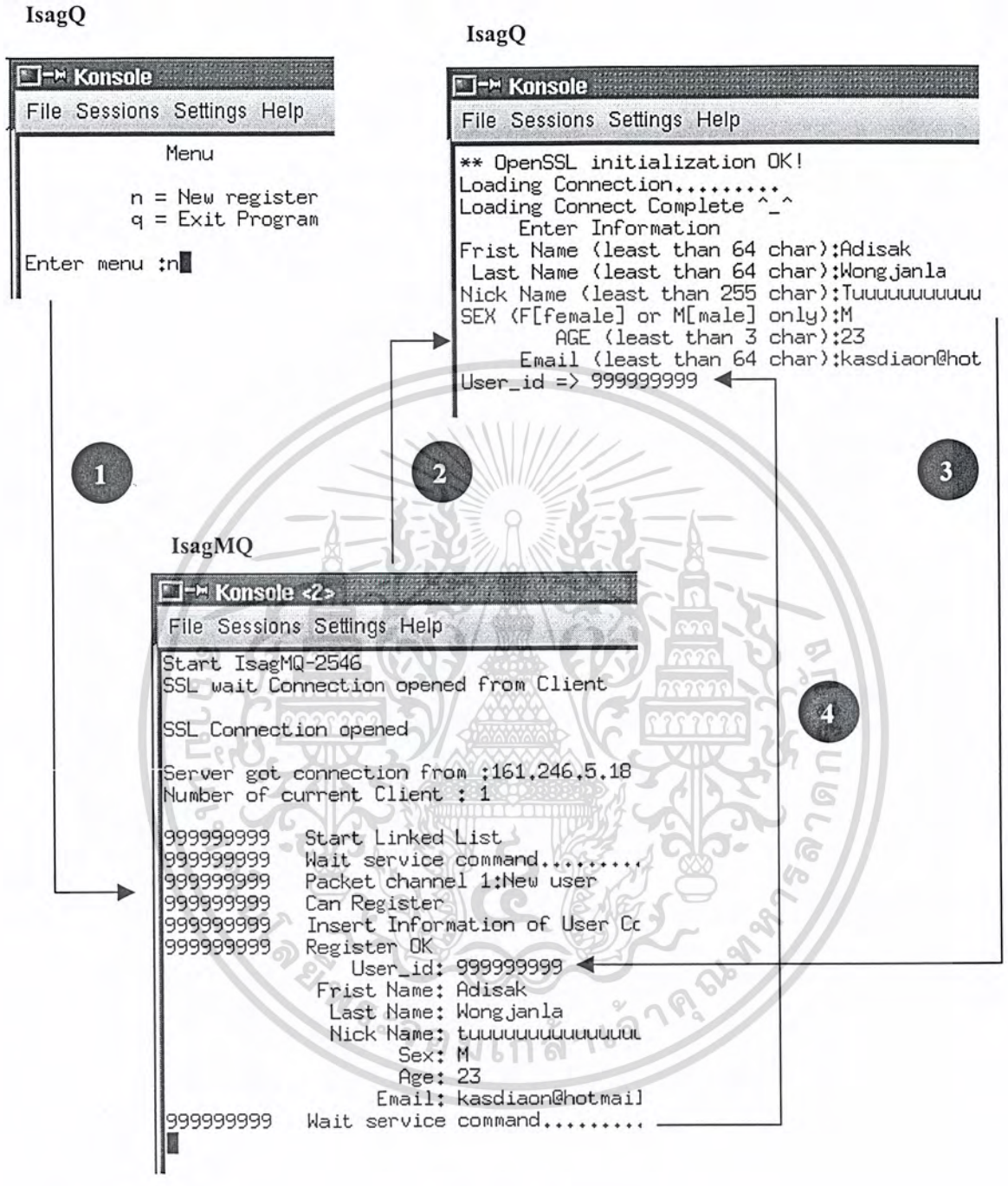
รูปที่ 6-6 ข้อผิดพลาดจากการใส่รหัสผ่านเพื่อเปิดคีย์ไม่ถูกต้อง

6.3 ทดสอบการให้บริการอย่างปลอดภัยของ IsagMQ

การทดสอบเพื่อให้เห็นว่าบริการปลอดภัยของ IsagMQ นั้นปลอดภัยจะทำการทดสอบเพียงสองบริการที่สำคัญก็น่าจะทำให้เห็นว่าบริการขอ IsagMQ ความปลอดภัยแล้ว บริการที่จะทำการทดสอบคือบริการลงทะเบียน (เน้นที่ข้อมูลส่วนตัวของผู้ให้ต้องปลอดภัย) และบริการล็อกอิน (เน้นที่การส่งคอนแท็กลิสต์ที่ปลอดภัย) บริการทั้งสองนี้จะต้องผ่านการทำใบรับรองสิทธิ์ในหัวข้อที่ 6-1 ก่อนทุกครั้งเพราะทั้งสองบริการเป็นการเริ่มต้นการเชื่อมต่อกับ IsagMQ

6.3.1 ทดสอบบริการลงทะเบียน

เริ่มต้นให้ IsagQ ทำการร้องขอการใช้บริการลงทะเบียนไปที่ IsagMQ และถ้า IsagMQ อนุญาตให้ลงทะเบียน IsagQ ก็จะส่งข้อมูลในการลงทะเบียนมาให้ IsagMQ ซึ่งขั้นตอนการทำเป็นไปตามโพรโตคอลในบทที่หก สุดท้าย IsagMQ ก็จะส่ง User_id ให้กับ IsagQ ดังรูปที่ 6-7



1.IsagQ Request Register

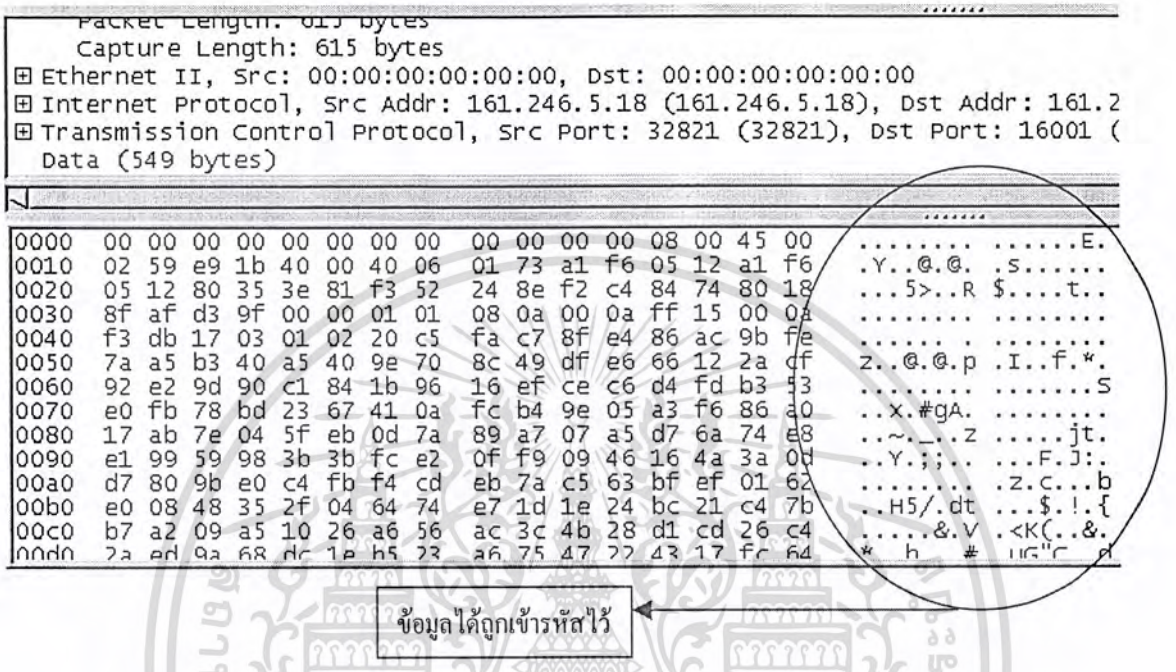
2.IsagMQ Response Register

3.IsagQ send information

4.IsagMQ send user_id to IsagQ

รูปที่ 6-7 IsagQ ใช้บริการลงทะเบียนของ IsagMQ

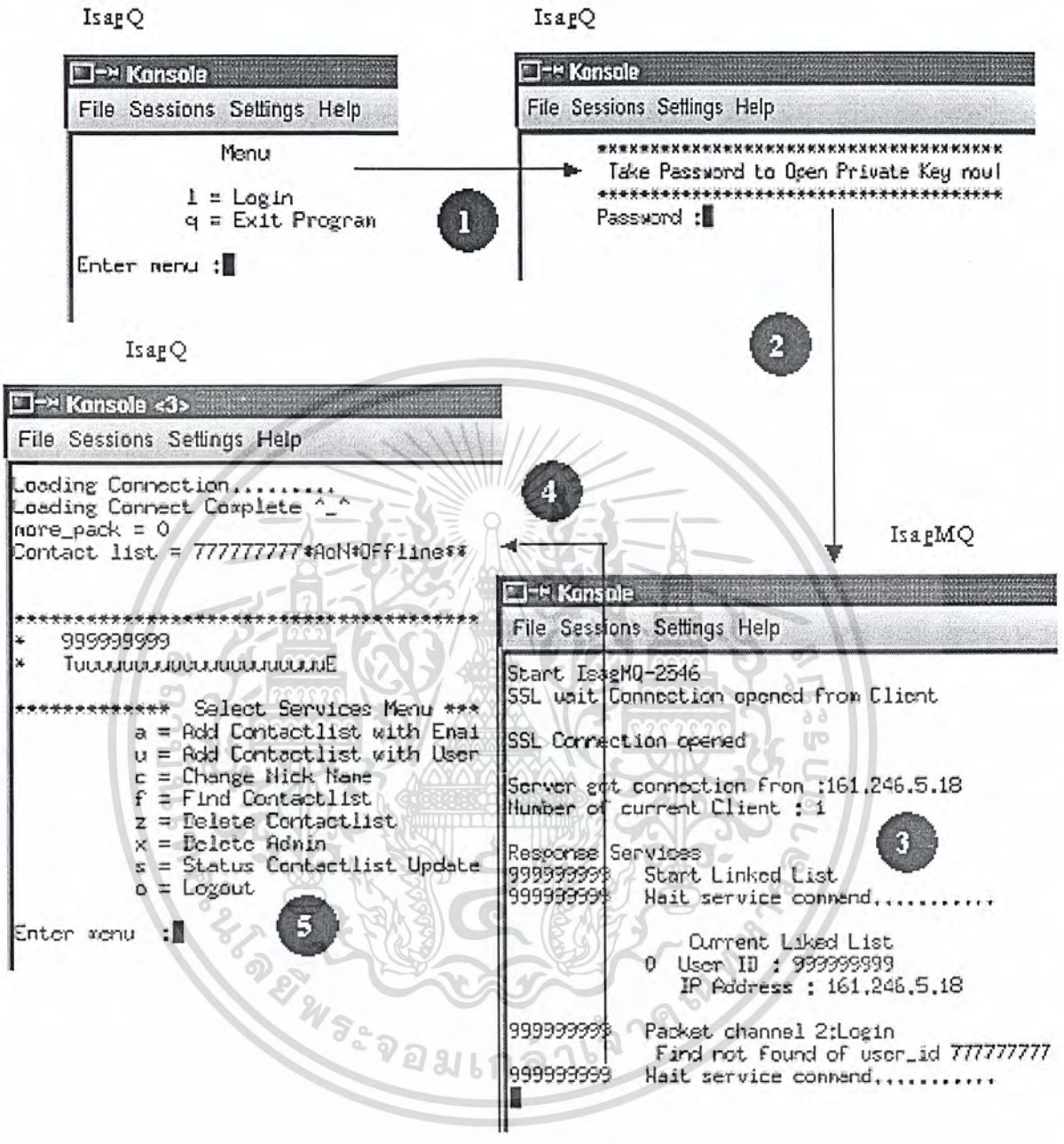
การดักจับข้อมูลจะดักจับข้อมูลในช่วงที่ IsagQ ส่งข้อมูลให้ส่วนตัวให้ IsagMQ ซึ่งข้อมูลทุกอย่างจะถูกเข้ารหัส เป็นดังรูปที่ 6-8



รูปที่ 6-8 การดักจับข้อมูลของบริการลงทะเบียนช่วง IsagQ ส่งข้อมูลให้ส่วนตัวให้ IsagMQ

6.3.2 ทดสอบบริการล็อกอิน

บริการล็อกอินถือว่าเป็นบริการที่ค่อนข้างอ่อนไหวกับการลักลอบดักจับข้อมูลที่ใช้ในการล็อกอิน เช่น Password และ User_id ของผู้ใช้ถือว่าเป็นข้อมูลที่เป็นความลับ และข้อมูลที่ IsagMQ ส่งให้กับแต่ละ IsagQ คือ คอนแท็กลิสต์ ถือว่าเป็นข้อมูลส่วนตัวที่เจ้าของข้อมูลเท่านั้นที่มีสิทธิ์จะได้และเห็นข้อมูลนั้น วิธีการล็อกอินสำหรับ IsagQ ไม่ต้องส่ง User_id และ Password มาให้กับ IsagMQ แต่ใช้การส่งใบรับรองสิทธิ์มาแทนซึ่ง IsagMQ จะใช้ข้อมูลในใบรับรองสิทธิ์นั้นระบุได้ว่าใครล็อกอิน โดยการดึงเอา Subject ของ ใบรับรองสิทธิ์วิธีนี้ถือว่ามีความปลอดภัยอีกขั้นหนึ่งเพราะไม่ต้องส่ง User_id และ Password ออกอินเตอร์เน็ตและอาจถูกดักจับได้ง่าย ขั้นตอนการล็อกอินแสดงดังรูปที่ 6-9



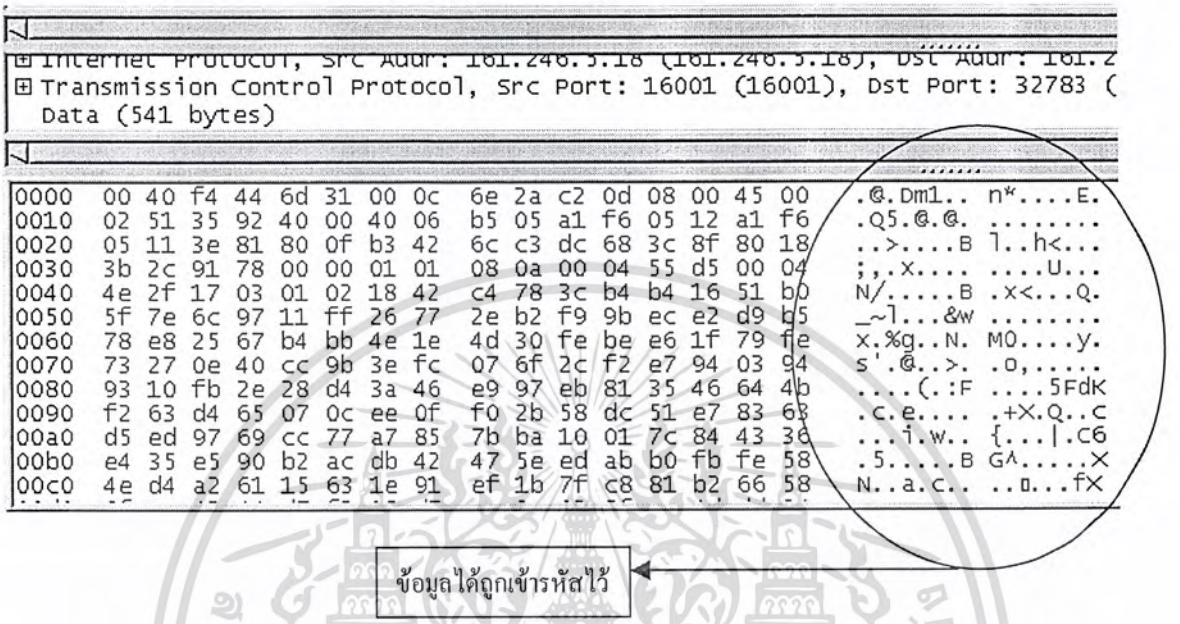
1. IsagQ ต้องการใช้บริการล็อกอิน 2. IsagQ ต้องทำการใส่รหัสผ่านถ้าผ่านทำการ Request Login มาที่ IsagMQ

3. IsagMQ ดึงข้อมูลจากใบรับรองสิทธิ์ เพื่อดูว่าใครล็อกอิน 4. IsagMQ ส่งคอนแท็กลิสตให้กับ IsagQ

5. IsagQ รับคอนแท็กลิสต์และใช้บริการที่ต้องการ

รูปที่ 6-9 IsagQ ใช้บริการล็อกอินและรับคอนแท็กลิสต์จาก IsagMQ

การดักจับข้อมูลข้อบริการล็อกอินจะดักจับข้อมูลในช่วงที่ IsagQ รับคอนแท็กลิสต์จาก IsagMQ ซึ่งข้อมูลทุกอย่างจะถูกเข้ารหัส เป็นดังรูปที่ 6-10



รูปที่ 6-10 การดักจับข้อมูลข้อบริการล็อกอินจะดักจับข้อมูลในช่วงที่ IsagQ รับคอนแท็กลิสต์จาก IsagMQ

บทที่ 7
สรุปผลการพัฒนาโครงงาน

7.1 คุณสมบัติของโปรแกรม

จากการพัฒนาโครงงานสามารถทำการพัฒนาโปรแกรมให้มีความสามารถดังนี้

Input specification

- รับการร้องขอ (Request) การขอใช้บริการจากไคลเอ็นต์ผ่านทางเครือข่าย

Output specification

- ทำการตอบสนอง (Reply) การขอใช้บริการจากไคลเอ็นต์ผ่านทางเครือข่าย
- แสดงผลการตอบสนองออกทางจอภาพโดยเป็นเท็กซ์โหมด

Function specification

- สามารถให้บริการลงทะเบียนได้
- สามารถให้บริการยกเลิกการลงทะเบียนได้
- สามารถให้บริการเพิ่มคอนแท็ก (Contact) ได้
- สามารถให้บริการลบคอนแท็กได้
- สามารถให้บริการค้นหาคอนแท็กได้
- สามารถให้บริการเปลี่ยนชื่อเล่นได้
- สามารถให้บริการล็อกอินและล็อกเอาต์ได้
- สามารถส่งคอนแท็กลิสต์แก่ไคลเอ็นต์ได้
- สามารถบอกสถานะออนไลน์และออฟไลน์ของแต่ละคอนแท็กได้
- สามารถสร้างการสื่อสารระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ให้อยู่ในช่องทางที่ปลอดภัยด้วยการเข้ารหัสและถอดรหัสได้
- สามารถรับประกันว่าผู้ที่จะสามารถใช้บริการของ IsagMQ ได้นั้นจะต้องผ่านการพิสูจน์ตนอย่างถูกต้องแล้วโดยใช้หลักการ Certificate Authority

7.2 ประโยชน์ของการพัฒนาโครงงาน

- โปรแกรม IsagMQ สามารถนำไปใช้ในองค์กรใดๆ ก็ได้ที่ต้องการมีระบบรับส่งสารด่วนแบบปลอดภัย
- ผู้ใช้บริการจะมีความมั่นใจว่าข้อมูลที่เป็นความลับของตนเองจะไม่ถูกผู้อื่นนำไปใช้

- ผู้ใช้บริการจะมีความมั่นใจว่าสามารถส่งข้อมูลที่เป็นความลับต่อคู่สนทนาได้อย่างปลอดภัย

7.3 ข้อจำกัดของโครงงาน

- การส่งกุญแจและการออกใบรับรองสิทธิ์ที่เป็นแพลนเท็กซ์ (Plaintext) สำหรับให้ผู้ใช้นำไปใช้ในระบบ ยังไม่ทำให้ส่งผ่านทางเครือได้ เพราะอยู่นอกเหนือโครงงานนี้
- IsagMQ สามารถร้องรับผู้ใช้ยังไม่มากนัก

7.4 ข้อเสนอแนะสำหรับผู้นำโครงงานไปพัฒนา

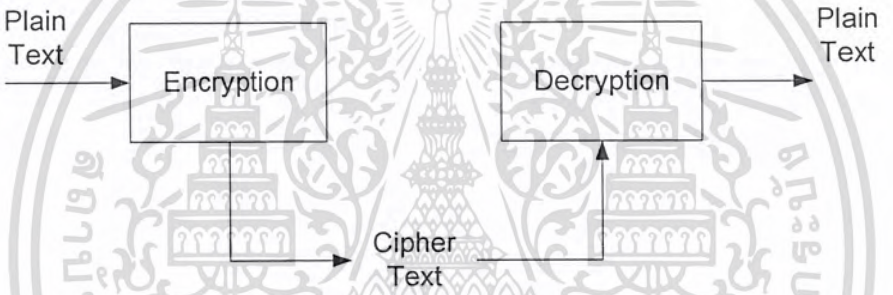
- ควรให้มีเวบไซต์สำหรับการลงทะเบียนเป็นสมาชิกของ IsagMQ แล้วนำข้อมูลที่ได้จากการลงทะเบียนมาสร้างกุญแจและการออกใบรับรองสิทธิ์ที่เป็นแพลนเท็กซ์ สำหรับให้ผู้ใช้นำไปใช้ในระบบแล้วส่งให้ผู้ใช้ผ่านทางอีเมล์
- ควรศึกษาการเขียนโปรแกรมสำหรับติดต่อและใช้งาน SSL โพรโตคอลในระดับขั้นสูงเช่นการทำเซสชันใน SSL เพื่อทำให้ระบบมีความปลอดภัยสูงและมีความรวดเร็วทันใจต่อการใช้งาน
- ควรปรับปรุงระบบให้สามารถรองรับผู้ใช้ให้ได้มากกว่านี้
- ควรมีการศึกษาและวางแผนการเขียนโปรแกรมให้เกิดความปลอดภัย (Secure Programming) ก่อนทำการเริ่มเขียนโปรแกรม เพื่อป้องกันไม่ให้เกิดการโจมตีด้วยวิธีการต่างๆ โดยเฉพาะอย่างยิ่งการทำ Buffer Overflow

ภาคผนวก ก.

วิธีการเข้ารหัสและถอดรหัสข้อมูลแบบต่างๆ

ก.1 พื้นฐานการเข้ารหัสและถอดรหัส

การเข้ารหัส (Encryption) คือ กระบวนการในการเปลี่ยนข้อมูลต้นฉบับให้อยู่ในอีกรูปแบบหนึ่งที่ไม่สามารถเข้าใจได้โดยง่าย ผลลัพธ์ที่ได้จากการเข้ารหัสจะสามารถกลับไปเป็นข้อมูลต้นฉบับได้นั้นต้องใช้การถอดรหัส (Decryption) โดยเราจะเรียกข้อมูลต้นฉบับที่จะทำการเข้ารหัสว่า เคลียร์เท็กซ์ (Clear text) หรือ เพลนเท็กซ์ (Plain text) และเราจะเรียกข้อมูลที่ทำการเข้ารหัสเรียบร้อยแล้วว่า ไซเฟอร์เท็กซ์ (Cipher text), โค้ดเท็กซ์ (Code text) หรือ ไซเฟอร์ (Cipher) การเข้ารหัสและถอดรหัสสามารถเขียนไดอะแกรมแสดงได้ดังนี้



รูปที่ ก-1 แสดงการเข้ารหัสและถอดรหัส

ข้อมูลเพลนเท็กซ์ = P จะแสดงอยู่ในรูปอนุกรมดังนี้

P = [P1, P2, …, Pn] และเมื่อเข้ารหัสแล้วจะเปลี่ยนเป็น C = [C1, C2, …, Cn]

เขียนให้อยู่ในอีกรูปแบบ C = E(P), P = D(C) หรือ P = D(E(P))

C = Cipher text

P = Plain text

E = Encryption algorithms

D = Decryption algorithms

หลักการของการเข้ารหัสโดยทั่วไปมี 2 ประเภทคือ

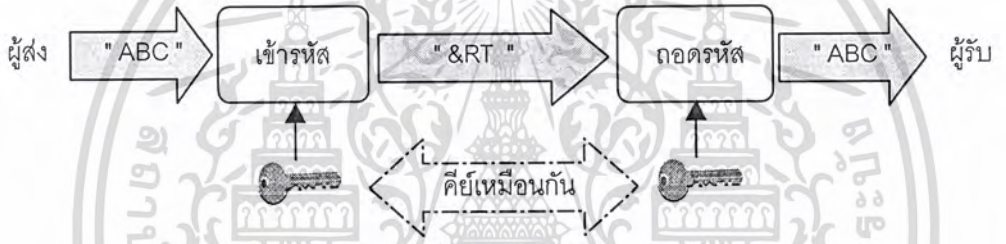
1. การแทนที่ (Substitution) เป็นการแทนที่บิตใด ๆ ด้วยข้อมูลอื่น ทำให้ข้อมูลมีความสับสนยากต่อการถอดรหัส เช่น เรามีข้อความว่า PRIVATE แล้วใช้หลักการเพิ่มค่ารหัสแอสกี (ASCII) ของตัวอักษรแต่ละตัวในข้อความไปอีกตามจำนวนที่ต้องการเช่น 3, 5เป็นต้น แล้วแทนที่ในข้อความต้นฉบับจะได้ข้อความออกมาเป็น SULYDWH เป็นต้น

2. การสับเปลี่ยนตำแหน่ง (Permutation) เป็นการสับเปลี่ยนตำแหน่งใด ๆ ของข้อมูล เมื่อมีการสับเปลี่ยนตำแหน่งมาก ๆ ทำให้ข้อมูลมีความซับซ้อนยากต่อการถอดรหัส เช่น เรามีข้อความว่า PRIVATE จะได้ข้อความที่จากการเข้ารหัสเป็น VRIPTEA เป็นต้น

ก.2 การเข้ารหัสและถอดรหัสโดยใช้คีย์ (Encryption and Decryption with key)

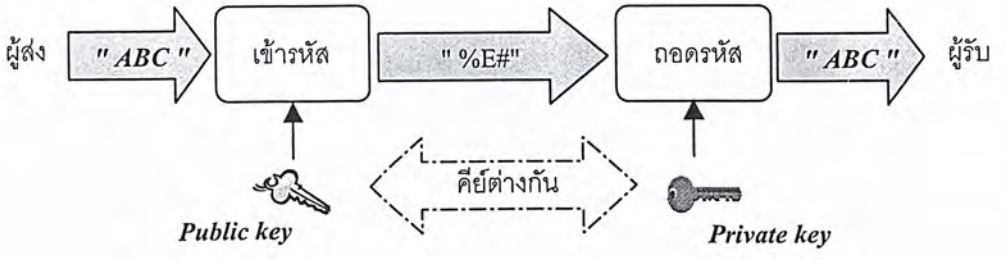
ระบบการเข้ารหัสและถอดรหัสโดยใช้คีย์นั้น การเข้ารหัสและถอดรหัสข้อมูลจะเปลี่ยนแปลงไปตามคีย์นอกจากอัลกอริทึมที่ใช้แล้ว ถึงผู้อื่นรู้อัลกอริทึมในการเข้ารหัสแต่ไม่รู้คีย์ก็ไม่สามารถถอดรหัสออกมาได้ กระบวนการเข้ารหัสและถอดรหัสแบบนี้มีอยู่ 2 ประเภทคือ

ก.2.1 ระบบการเข้ารหัสแบบสมมาตร (Symmetric Cryptosystem) หรือเรียกว่าการเข้ารหัสแบบคีย์เหมือน เป็นระบบที่การเข้ารหัสและการถอดรหัสใช้รูปแบบและคีย์เดียวกัน เช่น DES (Data Encryption Standard), 3DES (Triple DES), IDEA (International Data Encryption), CDMF (Commercial Data Masking Facility) เป็นต้น



รูปที่ ก-2 แสดงการเข้ารหัสและถอดรหัสโดยใช้คีย์เดียว

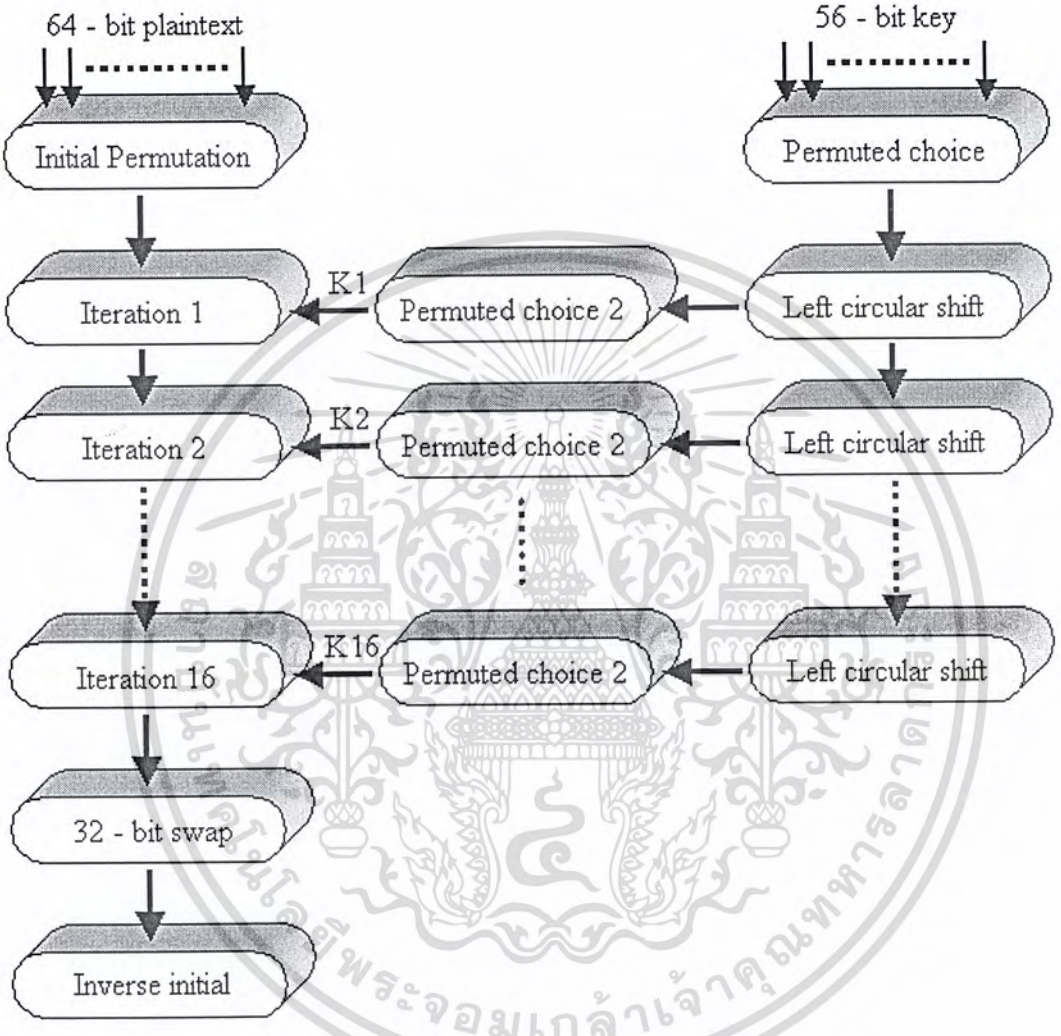
ก.2.2 ระบบการเข้ารหัสแบบไม่สมมาตร (Asymmetric Cryptosystem) หรือเรียกว่าการเข้ารหัสแบบคีย์ต่าง เป็นระบบการเข้ารหัสที่ใช้ 2 คีย์ที่มีความเกี่ยวข้องกันทางคณิตศาสตร์ โดยประกอบด้วยคีย์สาธารณะ (Public Key) และคีย์ส่วนตัว (Private Key) โดยถ้าการเข้ารหัสทำโดยการใช้คีย์สาธารณะการถอดรหัสต้องใช้คีย์ส่วนตัวที่เป็นคู่ของมันในการถอดรหัสเท่านั้น ในทางตรงกันข้ามถ้าใช้คีย์ส่วนตัวในการเข้ารหัสต้องใช้คีย์สาธารณะในการถอดรหัสเท่านั้นเช่นกัน เช่น RSA, DH, DSA



รูปที่ ก-3 แสดงการเข้ารหัสและถอดรหัสโดยใช้คีย์สาธารณะ

ก.3 การเข้ารหัสแบบ DES

การทำงานของ DES จะมีลักษณะคือข้อมูลที่เข้ามาในส่วนฟังก์ชันของการเข้ารหัสจะมี 2 ส่วนด้วยกันคือ ข้อมูลที่ยังไม่ถูกเข้ารหัสขนาด 64 บิตและคีย์ซึ่งมีขนาด 56 บิต ซึ่งรูปในด้านซ้ายมือจะแสดงขั้นตอนจัดการกับข้อมูลที่ยังไม่เข้ารหัส โดยสามารถแบ่งย่อยๆได้อีก 3 เฟส



รูปที่ ก-4 ขั้นตอนการทำงานของ DES

- เฟสที่ 1 จะทำการจัดการกับข้อมูลที่ไม่ได้ผ่านการเข้ารหัสขนาด 64 บิตผ่านเข้าไปยังส่วนที่เรียกว่า Initial Permutation (IP) ซึ่งจะทำการเรียบเรียงบิตใหม่เพื่อผลิตข้อมูลที่มีการสลับตำแหน่ง
- เฟสที่ 2 จะทำฟังก์ชันเดียวกัน 16 ครั้ง ซึ่งฟังก์ชันนี้รวมการทำ permutation และ substitution ซึ่งผลลัพธ์ที่ได้จากการทำทั้งหมด 16 ครั้งนี้จะได้ข้อมูลขนาด 64 บิตโดยใช้ทั้งข้อมูลที่ไม่ได้ผ่านการเข้ารหัสและคีย์ในการทำ ซึ่งข้อมูลที่มีขนาด 64 บิตที่ได้นี้แบ่งเป็น 2 ด้านคือซ้ายและขวา ทั้งหมดจะถูกสับเปลี่ยนเพื่อผลิต 64 บิตที่เป็น preoutput

- เฟสที่ 3 จะนำ preoutput ผ่านเข้าไปยังส่วนที่เรียกว่า Inverse initial permutation หรือ IP^{-1} ซึ่งทำหน้าที่อินเวอร์สตัวฟังก์ชัน initial permutation ซึ่งทั้งหมดจะผลิต 64 บิตที่เรียกว่าข้อมูลที่ผ่านการเข้ารหัสแล้ว (Ciphertext)

การทำ initial permutation

การทำ Initial permutation และการทำ inverse initial permutation จะถูกอธิบายโดยใช้ตารางด้านล่างตามลำดับ ซึ่งจะเห็นได้ว่าฟังก์ชัน permutation ทั้งสองต่างเป็นส่วนกลับซึ่งกันและกัน พิจารณาจาก 64 บิต:M ที่เข้ามา

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}	M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}	M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

M_i เป็นตัวเลขฐานสอง เมื่อทำการ permutation X = IP(M) จะได้ดังนี้

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2	M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6	M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1	M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5	M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

ถ้าเราทำการ Inverse permutation $Y = IP^{-1}(X) = IP^{-1}(IP(M))$ เราจะสามารถเห็นลำดับในการเรียงของบิตที่มีรูปแบบดังเดิม

รายละเอียดของการทำฟังก์ชันในแต่ละรอบ

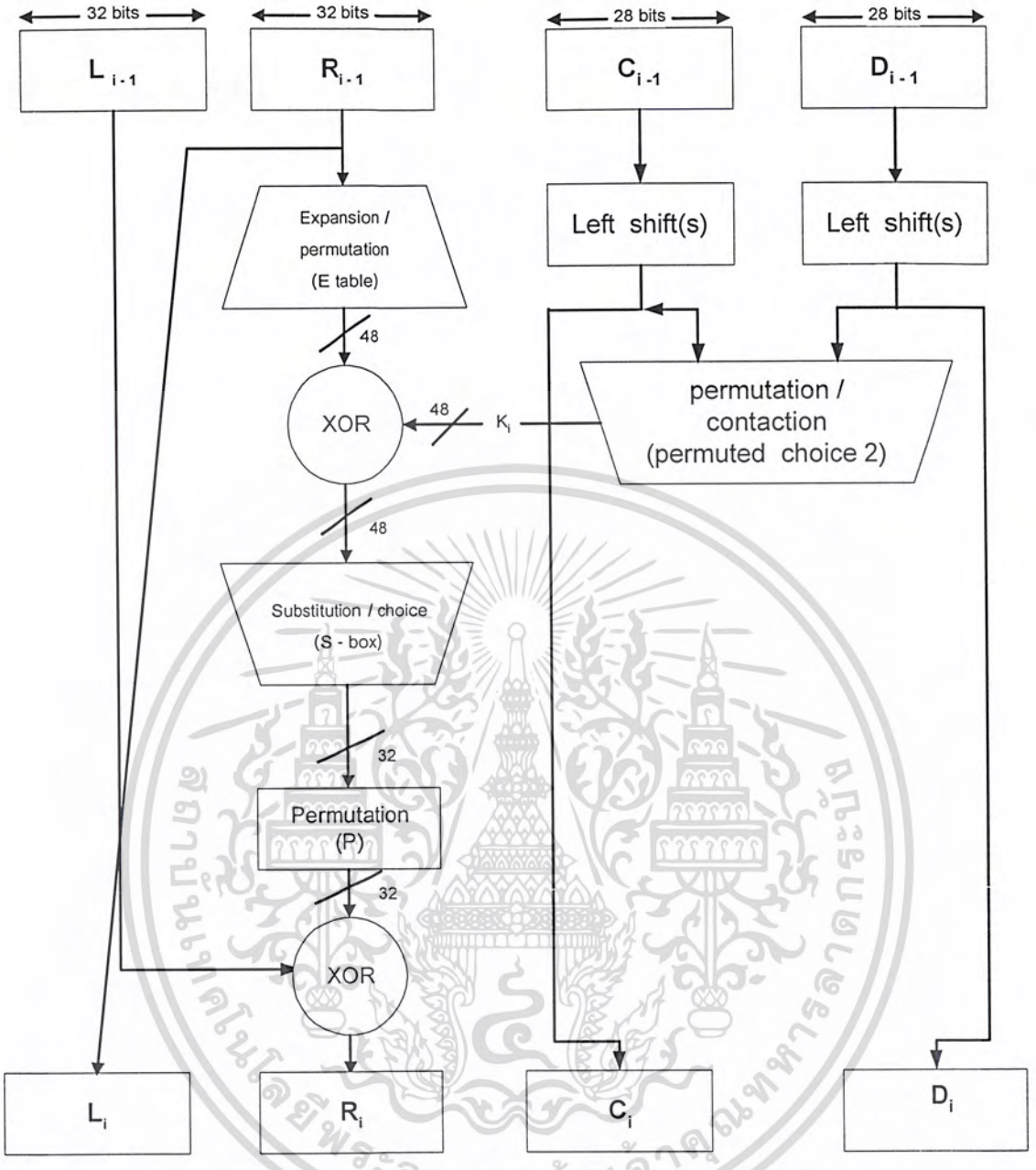
จากข้อมูลที่เข้ามาที่มีขนาด 64 บิตจะทำการแบ่งข้อมูลเป็น 2 ส่วนด้วยกัน ส่วนละขนาด 32 บิต (แบ่งเป็นซ้ายกับขวา) ซึ่งกระบวนการทำในแต่ละครั้งสามารถสรุปเป็นสูตรได้ดังนี้

$$L_I = R_{I-1}$$

$$R_I = L_{I-1} \oplus f(R_{I-1}, K_I)$$

โดย $\oplus$ หมายถึงการทำ XOR function

จากสูตรในข้างต้นจะเห็นได้ว่า 32 บิตด้านซ้ายมือ (L_I) จะเท่ากับด้านขวาของ (R_{I-1}) รอบที่ผ่านมา โดย R_I จะเท่ากับการนำ L_{I-1} มา XOR กับ $f(R_{I-1}, K_I)$ ซึ่งฟังก์ชัน f จะแสดงดังรูปที่ 4.9



รูปที่ ก-5 แสดงการเข้ารหัส DES ในแต่ละครั้ง (ทำทั้งหมด 16 ครั้ง)

(a) Initial Permutation (IP)

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

Output bit	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Form input bit	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

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

(c) Expansion Permutation (E)

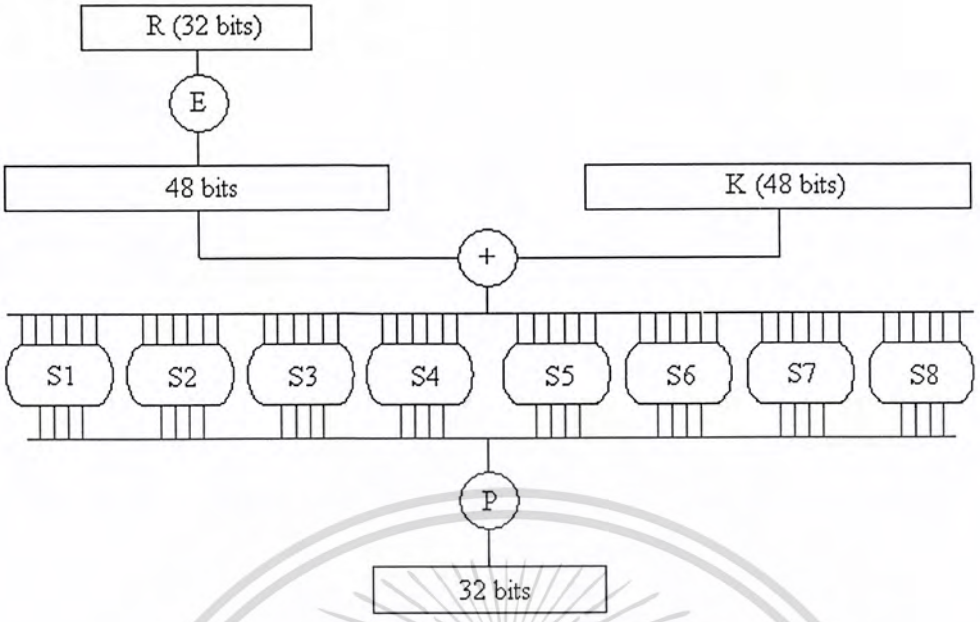
Output bit	1	2	3	4	5	6	7	8	9	10	11	12
Form input bit	32	1	2	3	4	5	4	5	6	7	8	9
Output bit	13	14	15	16	17	18	19	20	21	22	23	24
Form input bit	8	9	10	11	12	13	12	13	14	15	16	17
Output bit	25	26	27	28	29	30	31	32	33	34	35	36
Form input bit	16	17	18	19	20	21	20	21	22	23	24	25
Output bit	37	38	39	40	41	42	43	44	45	46	47	48
Form input bit	24	25	26	27	28	29	28	29	30	31	32	1

(d) Permutation Function (P)

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

ตารางที่ ก-1 แสดง permutation ของ DES

โดยคีย์ K_i ที่ใช้ในแต่ละรอบจะมีขนาด 48 บิตและอินพุทด้านขวา (R) มีขนาด 32 บิต ดังนั้นจึงต้องมีการขยายขนาดจาก 32 บิตให้เป็น 48 บิต โดยใช้ตารางที่ได้มีการกำหนด permutation และ expansion ซึ่งรวมทั้งการจำลอง 16 บิตที่เพิ่มขึ้นมาของด้านขวา ซึ่งจะนำผลที่ได้ที่มีขนาด 48 บิตจะถูก XOR กับ K_i โดยจะนำผลที่ได้ผ่านไปยังฟังก์ชันที่เรียกว่า Substitution และ Permutation ที่สามารถผลิตผลลัพธ์ที่มีขนาด 32 บิต



รูปที่ ก-6 แสดงการคำนวณ f(R, K)

โดย Substitution จะประกอบด้วยเซตของ S – box 8 อัน (S_1 - S_8) ซึ่ง S – box จะมีอินพุทขนาด 6 บิต และผลิตเอาต์พุทขนาด 4 บิต ซึ่งจากตารางที่ 4.3 จะแสดง DES S – box โดยมีวิธีการในการแปลงอินพุทขนาด 6 บิตให้กลายเป็นเอาต์พุทขนาด 4 บิตดังนี้คือ การนำบิตแรกและบิตสุดท้ายของอินพุทมาทำเป็นตำแหน่งของแถวและนำ 4 บิตตรงกลางมาเป็นตำแหน่งของคอลัมน์เช่น S_1 มีค่าเท่ากับ 011011 เราจะนำบิตแรกและบิตสุดท้ายซึ่งก็คือ 0 และ 1 มาเป็นตำแหน่งของแถวจะได้แถวที่ 01 และ 4 บิตตรงกลางที่เหลือคือ 1101 จะได้ตำแหน่งของคอลัมน์คือ คอลัมน์ที่ 13 ดังนั้นค่าที่ตำแหน่งแถวที่ 1 และคอลัมน์ที่ 13 ในตารางคือ 0101

Column Number

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	0	S1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	

Column Number

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

S2

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

S3

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

S4

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

S5

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

S6

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

S7

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

Column Number

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

Row **Box**

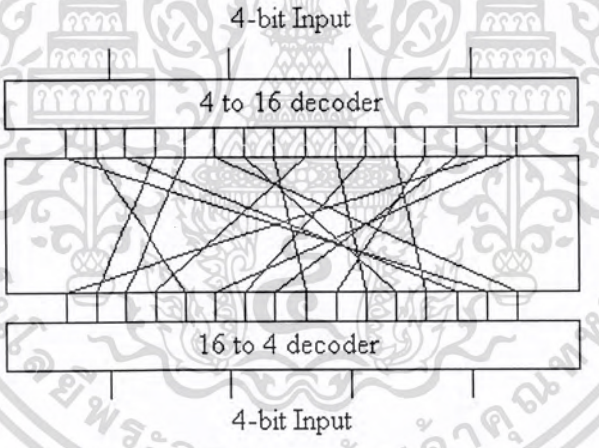
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	3	13	15	12	9	0	3	5	6	11

S8

ตารางที่ ก-2 แสดงขั้นตอนใน S- box ทั้ง 8 ชุด

การสร้างคีย์

ในรูปที่ 4.9 ซึ่งแสดงถึงการทำงานในแต่ละรอบของ DES จะเห็นได้ว่าคีย์ที่ใช้มีขนาด 56 บิตเป็นอินพุทในการทำ permutation โดยในตาราง Permuted Choice One เริ่มแรกจะทำการแบ่ง 56 บิตเป็น 2 ส่วนเท่าๆ กันส่วนละ 28 บิตโดยให้ชื่อในแต่ละส่วนว่า C กับ D ซึ่งในแต่ละรอบ (ทั้งหมด 16 รอบ) จะมีการทำ circular left shift ในแต่ละส่วนของ C และ D หรือทำ rotation โดยในแต่ละรอบจะมีการกำหนดว่าจะให้เลื่อนไปกี่บิตดังตาราง ซึ่งค่าที่ถูกเลื่อนจะกลายเป็นอินพุทของการทำให้รอบถัดไปและเป็นอินพุทของการทำ Permutation Choice Two ดังในตาราง หลังจากการทำ Permutation Choice Two แล้วจะได้เอาต์พุทขนาด 48 บิต ซึ่งเป็นอินพุทของ $f(R_{i-1},K_i)$



รูปที่ ก-7 แสดงการทำ Permutation Choice

(a) Permuted Choice One (PC-1)

Output bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14
From input bit	57	49	41	33	25	17	9	1	58	50	42	34	26	18
Output bit	15	16	17	18	19	20	21	22	23	24	25	26	27	28
From input bit	10	2	59	51	43	35	27	19	11	3	60	52	44	36

Output bit	29	30	31	32	33	34	35	36	37	38	39	40	41	42
From input bit	63	55	47	39	31	23	15	7	62	54	46	38	30	22

Output bit	43	44	45	46	47	48	49	50	51	52	53	54	55	56
From input bit	14	6	61	53	45	37	29	21	13	5	28	20	12	4

(b) Permuted Choice Two (PC-2)

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

(c) Schedule of Left Shifts

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

ตารางที่ ก-3 แสดงการสร้างคีย์

การถอดรหัสข้อมูล DES

กระบวนการถอดรหัสโดยใช้ DES นั้นเหมือนกับขั้นตอนในการเข้ารหัส ซึ่งมีขั้นตอนในการทำงานดังนี้คือ นำข้อมูลที่ผ่านการเข้ารหัสแล้ว (Ciphertext) มาเป็นอินพุทแต่จะมีการใช้คีย์ (K_i) ที่มีลำดับย้อนกลับกับคีย์ที่ใช้ในการเข้ารหัสเช่น K_{16} , K_{15} ... เป็นคีย์แรกและคีย์ถัดไปในการเข้ารหัสแทนดังรูปที่ 4.12 ด้านซ้ายมือจะเป็นขั้นตอนการเข้ารหัสและด้านขวามือเป็นขั้นตอนในการถอดรหัส

เราจะแสดงถึงผลลัพธ์ของขั้นตอนแรกในการกระบวนการถอดรหัส ซึ่งจะเท่ากับ 32 บิตที่ถูกสับเปลี่ยนจากอินพุทของการทำทั้งหมด 16 รอบของการเข้ารหัส เริ่มจาก

$$L16 = R15$$

$$R16 = L15 \oplus f(R15,K16)$$

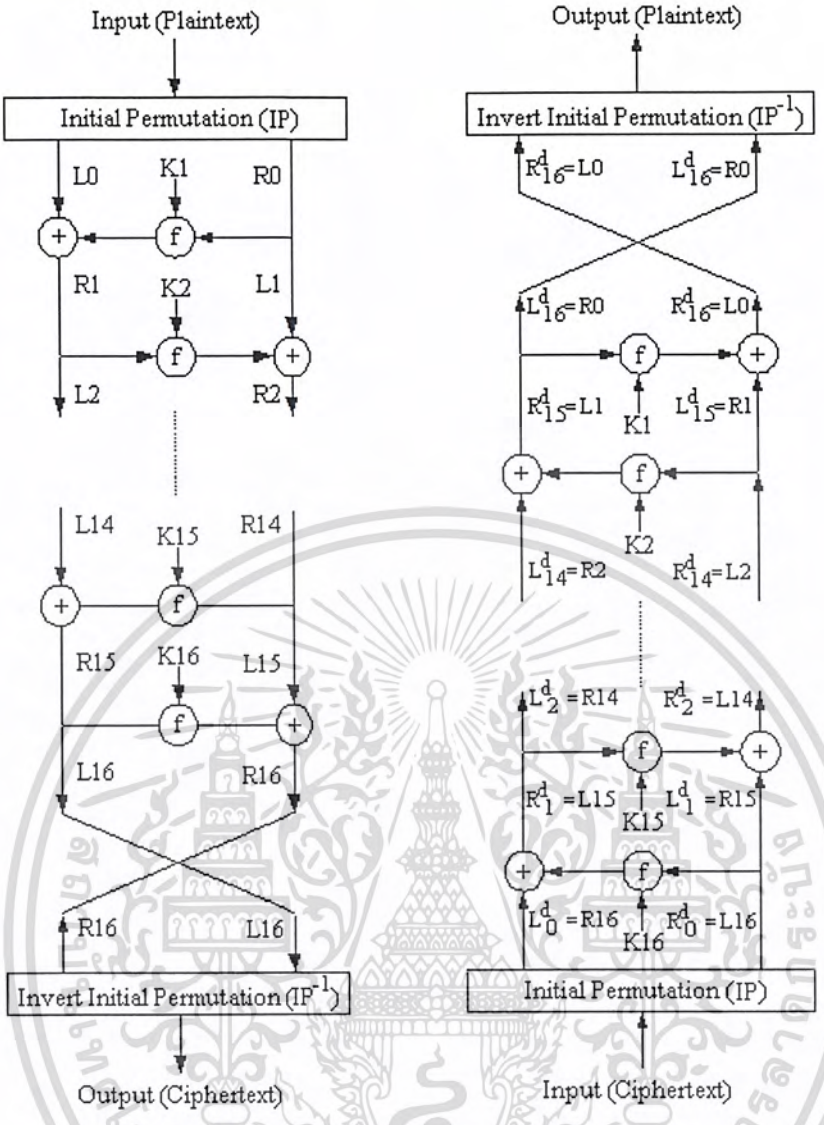
ในด้านการถอดรหัส

$$Ld1 = Rd0 = L16 = R15$$

$$Rd1 = Ld0 \oplus f(Rd0,K16)$$

$$= R16 \oplus f(R15,K16)$$

$$= [L15 \oplus f(R15,K16)] \oplus f(R15,K16)$$



รูปที่ ก-8 แสดงคีย์และขั้นตอนการเข้าและถอดรหัส DES

ซึ่งคุณสมบัติของ XOR ที่สำคัญคือ

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$

$$D \oplus D = 0$$

$$E \oplus 0 = E$$

ดังนั้นเรามี Ld1 = R15 และ Rd1 = L15 จะได้เอาต์พุทของขั้นตอนแรกในการถอดรหัสคือ L15||R15 ซึ่งเราสามารถเขียนเป็นสมการในการถอดรหัสได้ดังนี้คือ

$$Ri^{-1} = Li$$

$$Li^{-1} = Ri \oplus f(Ri-1,Ki) = Ri \oplus f(Li,Ki)$$

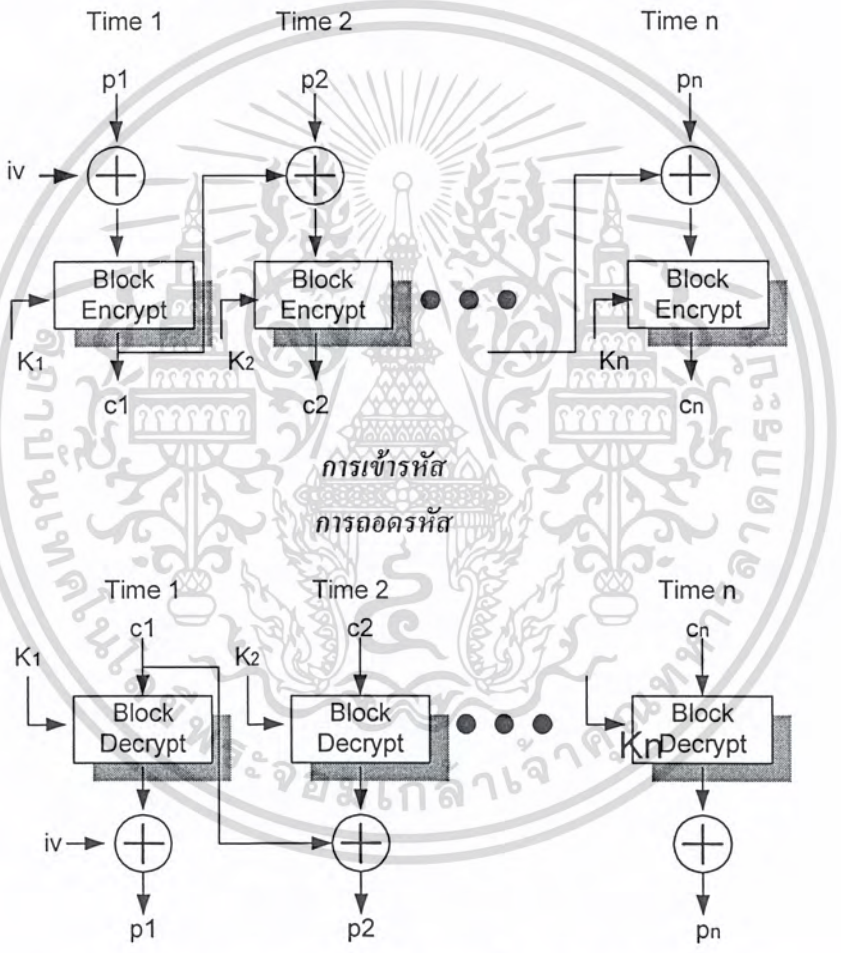
ซึ่งผลสุดท้ายเอาต์พุทที่ได้จากขั้นตอนสุดท้ายในการถอดรหัสคือ R0||L0 และนำไปสู่ขั้นตอนการทำ Inverse Permutation เราจะได้ข้อมูลที่ส่งมา (Plaintext) ดังสมการ

$$IP^{-1}(L0||R0) = IP^{-1}(IP(Plaintext)) = Plaintext$$

ก.4 การใช้ DES อย่างปลอดภัย

โหมด CBC (Cipher Block Chaining)

เป็นวิธีการเข้ารหัสที่พัฒนามาจาก DES ช่วยทำให้ข้อมูลที่ส่งยิ่งมีความปลอดภัยมากยิ่งขึ้น โดยมีวิธีการคือ จะนำข้อมูลที่ผ่านการเข้ารหัสของข้อมูลตัวก่อนมา XOR กับข้อมูลที่ยังไม่ได้เข้ารหัสของตัวถัดไปก่อนจะทำการเข้ารหัสแบบ DES ตามปกติดังรูป



รูปที่ ก-9 แสดงการเข้ารหัสและถอดรหัสของ DES CBC

ในการถอดรหัสข้อมูล ก็จะทำเช่นเดียวกับการถอดรหัสข้อมูล DES ตามปกติ แต่จะนำผลที่ได้จากการถอดรหัสมา XOR กับข้อมูลที่ผ่านการเข้ารหัส (Ciphertext) ตัวก่อนหน้านี้ เพื่อจะได้ข้อมูลจริงๆ(plaintext) ดังสมการ

$$Cn = Ek[Cn^{-1} \oplus Pn]$$

สมการในการถอดรหัส

$$Dk[Cn] = Dk[Ek(Cn^{-1} \oplus Pn)]$$

$$Dk[Cn] = Cn^{-1} \oplus Pn$$

$$Cn^{-1} \oplus Dk[Cn] = Cn^{-1} \oplus Cn^{-1} \oplus Pn = Pn$$

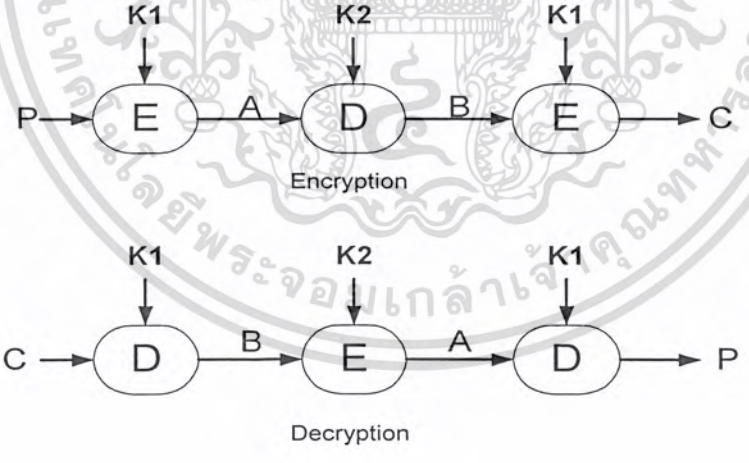
จะสังเกตเห็นได้ว่าบล็อกแรกของข้อมูลที่ผ่านการเข้ารหัส (Ciphertext) จะมีการนำ iv (Initialization Vector) มาทำการ XOR กับข้อมูลที่ไม่ได้เข้ารหัส (plaintext) ก่อนจะมีการเข้ารหัส DES ตามปกติ และในส่วนของการถอดรหัสข้อมูลก็จะใช้ iv ในการถอดรหัสข้อมูลเช่นเดียวกัน ดังนั้นจึงจำเป็นต้องมีข้อตกลงกันระหว่างผู้ส่งกับผู้รับก่อนว่าจะใช้ iv เป็นค่าใด เพื่อให้มีความปลอดภัยสูงสุด iv ควรจะมีการป้องกันเช่นเดียวกับ Key ซึ่งในการส่งค่า iv อาจจะทำการส่งโดยใช้การเข้ารหัสแบบ ECB

การเข้ารหัสแบบ 3DES (Triple DES)

เป็นที่ทราบกันว่ายิ่งคีย์มีความยาวมากขึ้น การถอดรหัสยิ่งทำได้ยากยิ่งขึ้น นอกจากนี้เราก็สามารถเพิ่มสมรรถนะของการเข้ารหัสให้มากขึ้นได้โดย การเข้ารหัสหลายๆ ครั้ง

อย่างไรก็ตามการเข้ารหัสครั้งที่สองโดยใช้คีย์ที่ต่างจากครั้งแรกนั้น มิได้หมายความว่าข้อมูลจะมีความปลอดภัยเพิ่มขึ้นเป็น 2 เท่า ถ้าการเข้ารหัสดังกล่าวเป็นการเข้ารหัสโดยเทคนิคทางคณิตศาสตร์

DES มิได้อยู่ในกลุ่มดังกล่าว แต่การที่จะทำให้ DES มีความปลอดภัยมากขึ้นเป็น 2 เท่านั้นจำเป็นที่เราจะต้องเข้ารหัสถึง 3 ครั้ง



รูปที่ ก-10 แสดงการเข้ารหัสและถอดรหัสแบบ 3DES

ขั้นตอนของการเข้ารหัสแบบ 3 ครั้งนั้นมีดังนี้

1. เข้ารหัสโดยใช้คีย์ตัวแรก

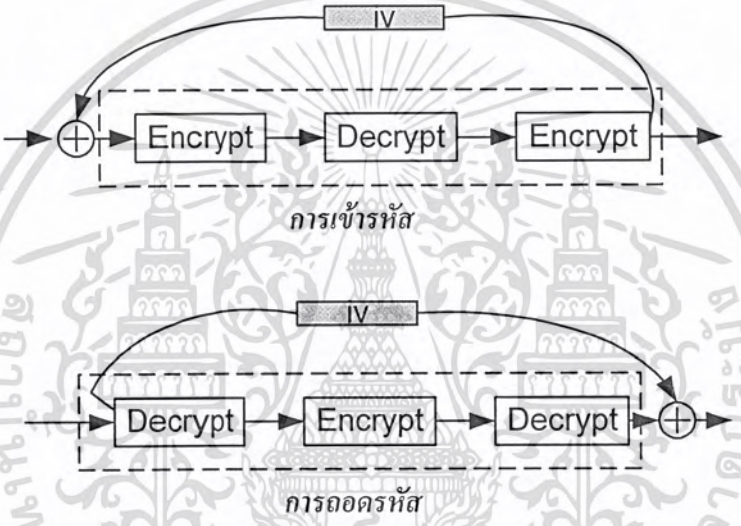
2. ถอดรหัสโดยคีย์ตัวที่สอง
3. เข้ารหัสโดยใช้คีย์ตัวที่สาม

(อาจใช้คีย์ตัวที่ 1 และ 3 เป็นตัวเดียวกันก็ได้ แต่ประสิทธิภาพของการเข้ารหัสก็จะลดลง)

Triple DES โหมด CBC (3DES CBC Mode)

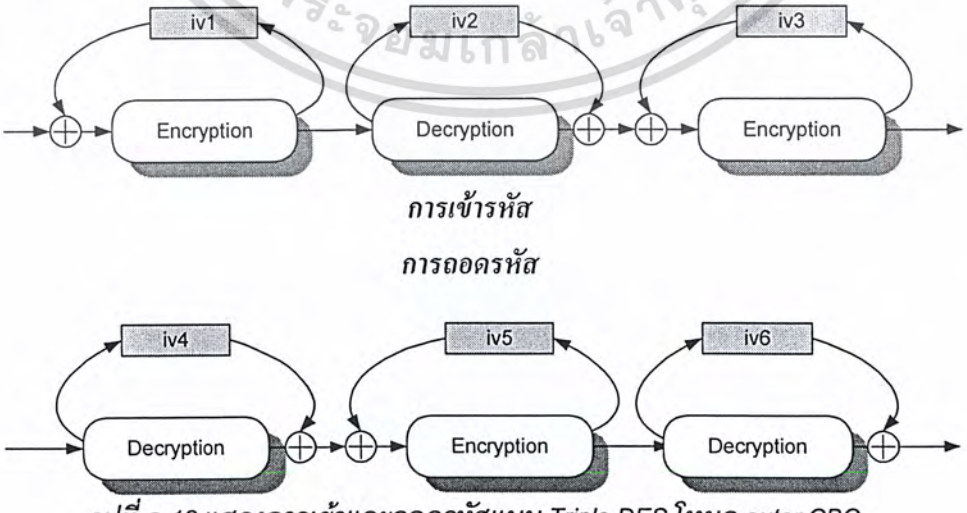
เนื่องจากการเข้ารหัสแบบ DES นั้นมีโหมดในการเข้ารหัสแบบ CBC ซึ่งการเข้ารหัส แบบ 3DES นี้ได้มีการประยุกต์มาจาก DES จึงได้มีการพัฒนา 3DES นี้ให้มีโหมด CBC ด้วย ซึ่งในการเข้ารหัสแบบ 3DES นี้ได้พัฒนาโหมดนี้ออกเป็น 2 แบบ

- แบบ inner CBC การเข้ารหัสแบบนี้จะมี iv เพียงตัวเดียวในการเข้ารหัสหรือถอดรหัสแบบ 3DES ในแต่ละครั้ง ดังรูป ก-11



รูปที่ ก-11 แสดงการเข้าและถอดรหัสแบบ Triple DES โหมด inner CBC

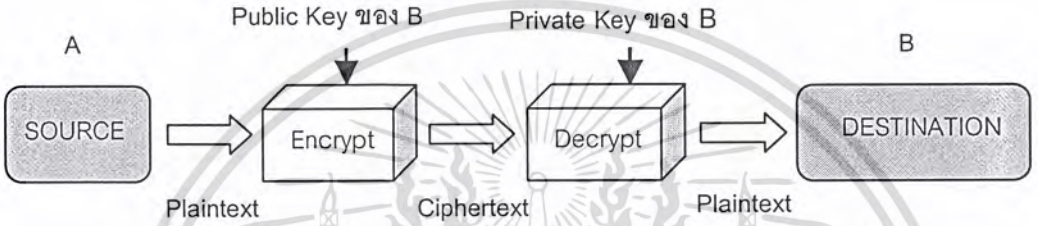
- แบบ outer CBC การเข้ารหัสแบบนี้แต่ละโมดูลในการเข้าหรือถอดรหัสจะมี iv เฉพาะของแต่ละโมดูล ดังรูป ก-12



รูปที่ ก-12 แสดงการเข้าและถอดรหัสแบบ Triple DES โหมด outer CBC

ก.5 การเข้ารหัสแบบ RSA

RSA ถูกคิดค้นขึ้นในปี 1977 โดยที่ชื่อ RSA ได้มาจากอักษรตัวแรกของนามสกุลของผู้ร่วมกันคิดค้น คือ Ron River, Adi Shamir และ Leonard Adleman เป็นวิธีการเข้ารหัสแบบคีย์สาธารณะที่เรียกว่า พับลิกคีย์ (Public–Key Crytosystem) เพื่อแก้ไขปัญหาการทำให้คีย์เป็นความลับ (Secret–Key Cryptosystem) โดยสมาชิกแต่ละคนจะต้องมีคีย์ 2 ชนิดคือ คีย์ส่วนตัวหรือไพรเวตคีย์ (Private Key) และคีย์สาธารณะหรือพับลิกคีย์ (Public Key) โดยคีย์ส่วนตัวจะถูกเก็บไว้เป็นความลับ ส่วนคีย์สาธารณะนั้นจะเปิดเผยให้ใครก็ได้ที่ต้องการส่งเอกสารให้แก่ตน หลักการทำงานของการเข้ารหัสแบบ RSA คือ ข้อมูลที่ถูกเข้ารหัสด้วยคีย์สาธารณะของผู้ใด จะถูกถอดรหัสได้ด้วยคีย์ส่วนตัวของผู้นั้นเท่านั้น การทำงานของระบบพับลิกคีย์สามารถอธิบายได้ดังต่อไปนี้



รูปที่ ก-13 แสดงขั้นตอนการทำ Public – Key Cryptosystem

หลักการทำงานของ RSA

ถ้าให้ p และ q เป็นจำนวนเฉพาะที่มีค่ามาก โดยที่ n = p*q เรียกว่าโมดูลัส (modulus) จากนั้นจึงเลือก e ที่มีค่าน้อยกว่า n และไม่สามารถหาร (p-1)(q-1) ได้ลงตัว ถ้าให้ d เป็นส่วนกลับของ e ในคณิตศาสตร์ระบบมอดูโลฐาน (p-1)(q-1) นั่นคือ

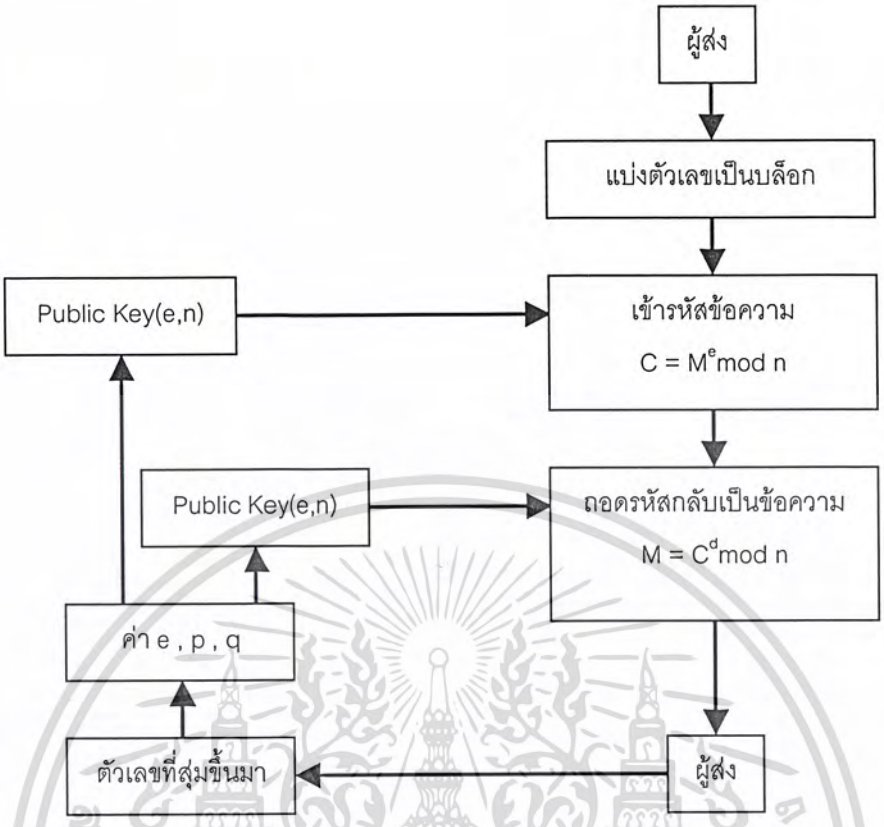
$$e*d \bmod (p-1)(q-1) = 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

ในรหัส RSA นั้น (n,e) คือพับลิกคีย์ (public key) ส่วน d คือไพรเวตคีย์ (private key) เมื่อได้ค่าเหล่านี้แล้ว p,q จะต้องเก็บเป็นความลับหรือถูกทำลายในทันที

ถ้าอริสาต้องการส่งเอกสารส่วนตัว m ไปให้บุญมี อริสาจะสร้างรหัสลับ c ของ m ได้โดยการให้ $c = m^e \bmod n$ เมื่อ(n,e) เป็นพับลิกคีย์ของบุญมี เมื่อบุญมีได้รับเอกสารรหัสลับ c เขาจะถอดรหัสเพื่ออ่านเอกสาร m ได้ด้วย d เพราะความสัมพันธ์ในสมการ(1) ระหว่าง e,d และ n จะทำให้

$$c^d = m^{ed \bmod (p-1)(q-1)} \bmod n = m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

และเพราะมีแต่บุญมีเท่านั้นที่รู้ค่า d บุญมีเท่านั้นจึงจะถอดรหัสได้ คุณสมบัติสำคัญประการหนึ่งของ RSA คือจากสมการ (2) ในทางกลับกันถ้าเราเข้ารหัสด้วยไพรเวตคีย์ เราก็สามารถถอดรหัสด้วยพับลิกคีย์ได้ด้วยเช่นกัน คุณสมบัติข้อนี้เองที่ทำให้ RSA มีประโยชน์มาก เพราะใช้ในการเซ็นระบบดิจิตอลได้อีกด้วย



รูปที่ ก-14 แสดงการเข้ารหัสแบบ RSA

ตัวอย่างขั้นตอนการเข้ารหัสแบบ RSA ซึ่งมีขั้นตอนในการหาคีย์ ดังนี้

1. เลือกจำนวนเฉพาะสองจำนวน คือ p = 7 , q = 17
2. คำนวณ n = p*q = 7 * 17 = 119
3. คำนวณ (p-1)(q-1) = 96
4. เลือก e ซึ่งมีความสัมพันธ์กับค่า (p-1)(q-1) ที่ได้กล่าวไว้ข้างต้น ในที่นี้เราจะใช้ 3
5. คำนวณค่า d ซึ่งสัมพันธ์กับสมการ e * d = 1 mod 96 ซึ่งค่าที่ถูกต้องคือ d = 77 เนื่องจาก

   77 * 5 = 385 = 4 * 96 +1

ดังนั้น Public Key คือ {5,119} และมีค่า Private Key คือ {77,119}

วิธีทำลายรหัส RSA ที่รู้จักกันดีที่สุดคือการหาค่า d นั่นเองจากสมการที่ (1) เราอาจหาค่า d ได้ หากรู้ค่า p และ q แต่เนื่องจาก p และ q เป็น prime number ที่ p*q = n ดังนั้นการทำลายรหัส RSA ขึ้นอยู่กับการแยกตัวประกอบของ n นั่นเอง แต่วิธีการแยกตัวประกอบของ n นั้นไม่ง่ายเลยหาก n มีค่ามาก R.Rivest ได้คำนวณไว้ในปี 1992 ว่าจะต้องใช้เงินประมาณ 8.3 ล้านเหรียญสหรัฐในการแยกตัวประกอบของ n ที่มีความยาว 512 บิต ค่านี้อาจลดลงได้ในอนาคต ดังนั้นสำหรับข้อมูลที่มีความสำคัญมากกว่า อาจจำเป็นต้องใช้ n ที่มีค่ามากถึง 700 หรือ 1000 ก็ได้

จะเห็นได้ว่าไม่ว่าการเข้าหรือถอดรหัส RSA ก็จำเป็นต้องใช้การยกกำลังในระบบมอดูโล ซึ่งการยกกำลังนั้นสามารถทำได้โดยการใช้วงจรคูณระบบมอดูโลมาต่ออนุกรมกัน ดังนั้นความเร็วของทั้งการเข้าและถอดรหัสจึงขึ้นกับความเร็วของวงจรคูณระบบมอดูโลเป็นอย่างมาก นี่เองเป็นจุดอ่อนข้อหนึ่งของ RSA เมื่อเปรียบเทียบกับคีย์เดียว เช่น DES เพราะปัจจุบันการคูณในระบบมอดูโลยังค่อนข้างยุ่งยากเมื่อเปรียบเทียบกับการแทนค่าหรือสลับตำแหน่งใน DES ได้มีการประมาณกันว่าสำหรับ n ที่มีความยาว 512 บิต การเข้ารหัสแบบ DES จะเร็วกว่า RSA ประมาณ 100 เท่า ถ้าเราทำการเข้ารหัสด้วยซอฟต์แวร์ และอาจเร็วกว่าถึง 1000 ถึง 10000 แล้วแต่ลักษณะของวงจร หากทำการเข้ารหัสด้วยฮาร์ดแวร์ โดยทั่วไปแล้วเราต้องการให้การเข้ารหัสเร็วกว่าการถอดรหัสดังนั้นเราจึงมักเลือกให้ e มีค่าน้อยกว่า d และยิ่งกว่านั้นเรามักให้ e ของสมาชิกทุกคนมีค่าเดียวกันเพื่อให้ฮาร์ดแวร์ของวงจรเข้ารหัสสำหรับสมาชิกแต่ละคนมีลักษณะคล้ายกัน

เนื่องจาก DES และ RSA มีข้อดีข้อเสียที่แตกต่างกัน จึงไม่จำเป็นว่ารหัสชนิดใดชนิดหนึ่งจะเหมาะสมในทุกสถานการณ์ โดยทั่วไปแล้ว DES จะถูกใช้ในการเข้ารหัสข้อมูลที่มีขนาดใหญ่เพราะรวดเร็วกว่าในขณะที่ RSA จะถูกใช้ในระบบสื่อสารที่ไม่ยาวนักแต่ต้องการความปลอดภัยสูงในบางครั้ง RSA ยังถูกใช้ร่วมกับ DES เพื่อเสริมจุดเด่นซึ่งกันและกัน เช่นตัวเอกสารจริงจะถูกเข้ารหัสด้วย DES โดยที่คีย์รหัส DES จะถูกเข้ารหัสด้วย RSA แล้วส่งไปด้วยกันหรือส่งไปก่อนแต่ในบางครั้ง DES อย่างเดียวก็พอแล้วหากการแลกเปลี่ยนคีย์สามารถทำได้อย่างปลอดภัยเพียงพอ หรือในกรณีที่ผู้ส่งและผู้รับเป็นบุคคลเดียวกัน เช่น ฮาร์ดดิสก์ในคอมพิวเตอร์ส่วนตัว หรือข้อมูลส่วนตัวในสมาร์ตการ์ด

บริการของการเข้ารหัสแบบไม่สมมาตร

1. การเข้ารหัสและการถอดรหัส (Encryption/Decryption) การเข้ารหัสแบบไม่สมมาตรสามารถทำการเข้ารหัสและถอดรหัสได้โดยใช้คีย์สาธารณะของผู้รับในการเข้ารหัส และใช้คีย์ส่วนตัวในการถอดรหัส
2. การลงลายมือชื่อดิจิตอล (Digital Signature) ทำโดยเข้ารหัสโดยใช้คีย์ส่วนตัวของผู้ส่งแล้วผู้รับสามารถใช้คีย์สาธารณะของผู้ส่งที่เป็นคู่คีย์ของมันถอดได้หรือไม่ ถ้าถอดได้คือการยืนยันตัวบุคคลที่ทำการลงลายมือชื่อนั้น
3. การแลกเปลี่ยนคีย์ (Key Exchange) ใช้ในการแลกเปลี่ยนซีเคร็ทคีย์ในการติดต่อในช่วงเวลาหนึ่ง โดยนำซีเคร็ทคีย์มาเข้ารหัสโดยใช้คีย์สาธารณะของผู้รับ ซึ่งผู้ที่สามารถถอดได้ก็คือผู้รับที่มีคีย์ส่วนตัวที่เป็นคู่คีย์ของมันเท่านั้น

จุดเด่นและจุดด้อยของการเข้ารหัสแบบไม่สมมาตร

จุดเด่นและจุดด้อยของการเข้ารหัสแบบไม่สมมาตรมีดังนี้

1. การเข้ารหัสค่อนข้างช้า และต้องใช้การคำนวณอย่างมาก

2. สามารถเข้ารหัสข้อมูลให้เป็นความลับได้ นอกจากนี้ยังสามารถตรวจสอบที่มาของข้อมูลบุคคลโดยใช้ร่วมกับลายมือชื่อดิจิตอลได้อีกด้วย

จากการเข้ารหัสและการถอดรหัสที่ใช้คีย์ ส่วนที่สำคัญนอกจากอัลกอริทึมที่ใช้ก็คือ คีย์ หากคีย์ที่ใช้มีความยาวมากก็จะทำให้ยากแก่การที่บุคคลอื่นจะคาดเดาเพื่อทำการถอดรหัสได้ ตารางต่อไปนี้แสดงระยะเวลาที่ใช้ในการที่ถอดรหัสโดยมีความยาวคีย์แบบต่าง ๆ

ค่าใช้จ่าย	ความยาวของคีย์ (บิต)				
	40	56	64	80	128
$ 100,000	2 วินาที	35 ชั่วโมง	1 ปี	70,000 ปี	10^{19} ปี
$ 1 ล้าน	0.2 วินาที	3.5 ชั่วโมง	37 วัน	7,000 ปี	10^{18} ปี
$ 100 ล้าน	2 มิลลิวินาที	2 นาที	9 ชั่วโมง	70 ปี	10^{16} ปี
$ 1,000 ล้าน	0.2 มิลลิวินาที	13 วินาที	1 ชั่วโมง	7 ปี	10^{15} ปี
$ 100,000 ล้าน	2 ไมโครวินาที	0.1 วินาที	32 วินาที	24 วัน	10^{13} ปี

ตารางที่ ก-4 แสดงระยะเวลาที่ใช้ในการที่ถอดรหัสโดยมีความยาวคีย์แบบต่าง ๆ

การเข้ารหัสแบบสมมาตร	การเข้ารหัสแบบไม่สมมาตร
ข้อดี 1. การเข้ารหัสทำได้อย่างรวดเร็ว 2. สามารถสร้างได้ง่ายโดยฮาร์ดแวร์	*ข้อดี* 1. ใช้คีย์ต่างกันในการเข้ารหัสและการถอดรหัส ทำให้การจัดส่งคีย์ทำได้ง่าย 2. สามารถตรวจสอบผู้ใช้ได้โดยใช้ร่วมกับลายมือชื่อดิจิตอล
ข้อเสีย 1. คีย์ของการเข้ารหัสและถอดรหัสต้องเหมือนกันการจัดส่งคีย์ทำได้ยาก	*ข้อเสีย* 1. ค่อนข้างช้าและต้องใช้การคำนวณอย่างมาก

ตารางที่ ก-5 แสดงข้อดีและข้อเสียของการเข้ารหัส

ภาคผนวก ข.

การดักจับแพ็กเกจโดยใช้โปรแกรม Ethereal

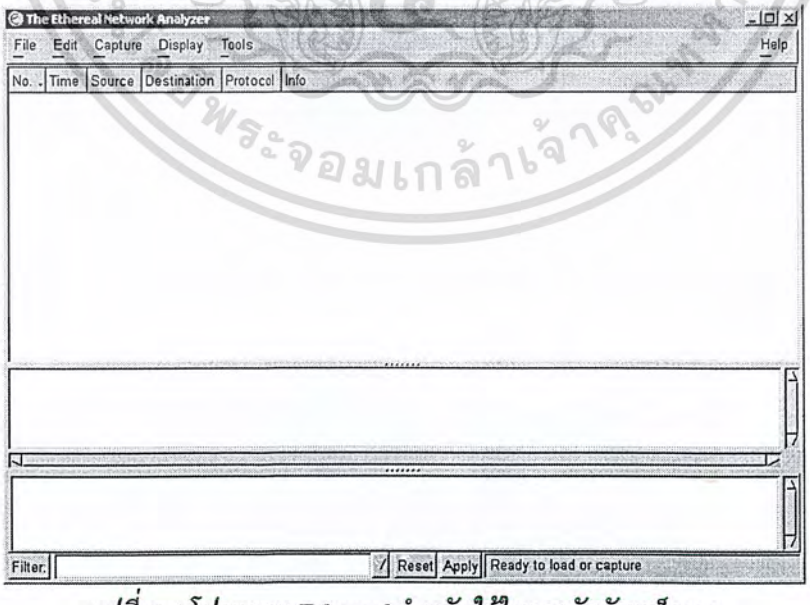
โปรแกรม Ethereal นั้นเป็นโปรแกรมที่ใช้สำหรับดักจับแพ็กเกจที่ใช้ในทดสอบผลการทำงานของ IsagMQ ซึ่ง Ethereal เป็นโปรแกรมที่ทำงานดีมากสามรถดักจับแพ็กเกจได้ตามความต้องการ โดยผู้ที่สนใจหรือต้องการนำโปรแกรมนี้มาใช้งานเพื่อการศึกษากระบวนการทำงานของโพรโตคอลต่าง สามารถดาวน์โหลดได้จาก http://www.ethereal.com/distribution/win32/ โดยให้เลือกดาวน์โหลดโปรแกรม Ethereal และโปรแกรม WinPcap

ข.1 การติดตั้งโปรแกรม Ethereal

เมื่อทำการดาวน์โหลดโปรแกรมมาจากเว็บไซต์ในข้างต้นแล้ว จะได้โปรแกรม Ethereal 0.9.7 ซึ่งอยู่ในรูปไฟล์ ethereal-setup-0.9.7.exe และโปรแกรม WinPcap 2.3 ซึ่งอยู่ในรูปไฟล์ WinPcap_2_3.exe ซึ่งในทั้ง 2 โปรแกรมนี้อาจมีการเปลี่ยนแปลงชื่อไฟล์ไปตามเวอร์ชัน จากนั้นให้ทำการติดตั้งโปรแกรมทั้ง 2 โปรแกรมไว้ในไดเรกทอรีเดียวกันโดยจะทำการติดตั้งโปรแกรมใดก่อนก็ได้

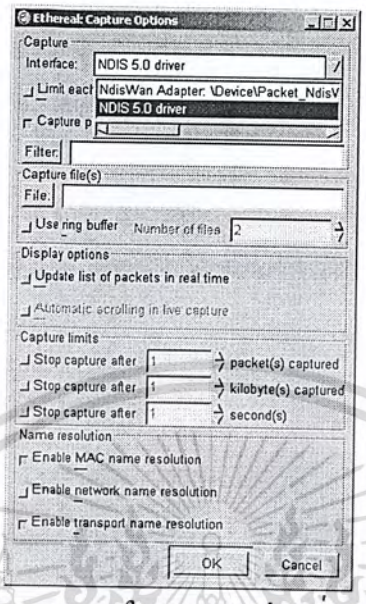
ข.2 การใช้งานโปรแกรม Ethereal

ในการดักจับแพ็กเกจโดยใช้โปรแกรม Ethereal ผู้ใช้งานไม่จำเป็นต้องทำการแก้ไขเซิร์ฟเวอร์ข ดังนั้นเมื่อเราต้องการจะดักจับแพ็กเกจในส่วนใดให้เปิดโปรแกรม Ethereal ขึ้นมา



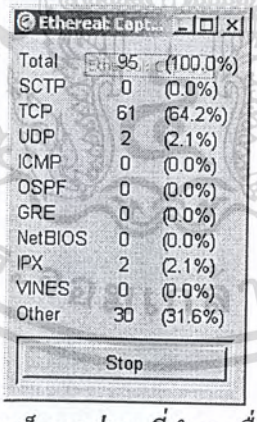
รูปที่ ข-1 โปรแกรม Ethereal สำหรับใช้ในการดักจับแพ็กเกจ

จากนั้นเมื่อต้องการเริ่มดักจับให้ทำการเลือก Start โดยคลิกที่รายการ Capture -> Start.. ซึ่งจะแสดงหน้าจอดังรูปที่ ข-2 แล้วให้ทำการเลือก Interface ใน Capture ให้เป็นของ Lan Card



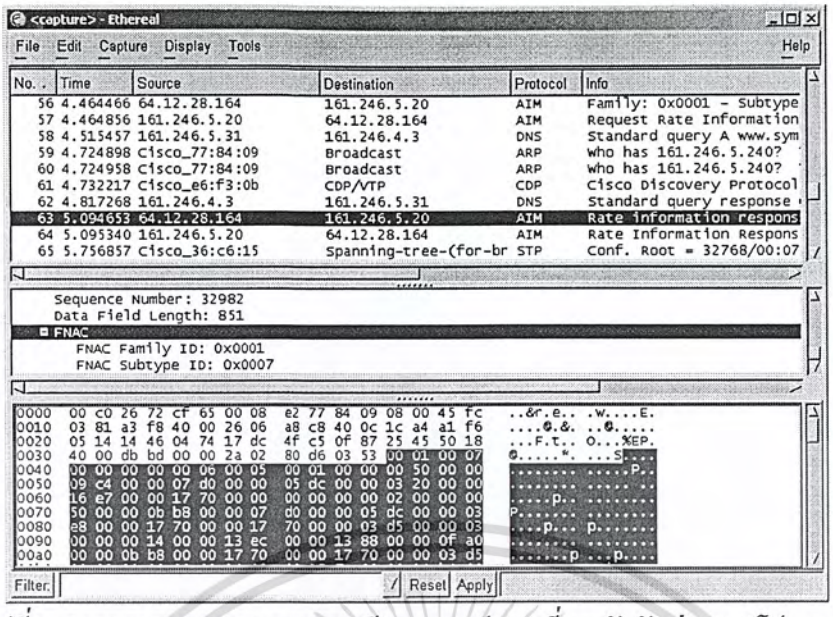
รูปที่ ข-2 การกำหนด Interface ใน Capture ก่อนเริ่มทำการดักจับแพ็กเกจ

เมื่อกำหนด Interface เรียบร้อยแล้วให้ทำการคลิด OK เพื่อเริ่มทำการดักจับแพ็กเกจต่างๆ ที่ทำการรับส่งระหว่างเครื่องของผู้ใช้และสถานที่อื่น



รูป ข-3 แสดงการดักจับแพ็กเกจต่างๆ ที่ทำการสื่อสารกับเครื่องของผู้ใช้

เมื่อผู้ใช้ต้องการจะทำการหยุดหรือตรวจสอบรายละเอียดของแต่ละแพ็กเกจโดยละเอียด ให้ทำการคลิก Stop เพื่อหยุดการดักจับแพ็กเกจ เมื่อหยุดการดักจับแพ็กเกจแล้ว โปรแกรมจะแสดงแพ็กเกจที่สื่อสารกับเครื่องผู้ใช้ ซึ่งผู้ใช้สามารถทำการคลิกเลือกที่แต่ละแพ็กเกจเพื่อดูรายละเอียดของแต่ละแพ็กเกจได้ โดยที่แพ็กเกจในการสื่อสารระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ผ่านทางโพรโตคอล AIM และแพ็กเกจในการสื่อสารระหว่างไคลเอ็นต์กับไคลเอ็นต์ผ่านทางโพรโตคอล TCP



รูปที่ ข-4 แสดงการตรวจสอบรายละเอียดของแพ็กเกจที่การดักจับผ่านทางโปรแกรม

บรรณานุกรม

หนังสืออ้างอิง

[1] Stephen A. Thomas : ***"SSL&TLS Essentials"***, Wiley Computer Publishing, United State of America 2000.

[2] Kurt Wall, Mark Watson and Mark Whitis : ***"Linux Programming"***, SAMS. United State of America 1999.

[3] Charlie Kaufman, Radia Perlman and Mike Speciner : ***"Network Security Private Communication in a Public World"***, Prentice Hall PTR, United State of America 2002

[4] Jahn Viega and Matt Messier : ***"Secure Programming Cookbook"***, O'REILLY & Associates, Inc. United State of America 2003.

[5] Jahn Viega, Matt Messier and Privir Chanda : ***"Network Security with OpenSSL"***, O'REILLY & Associates, Inc. United State of America 2003.

[6] เรืองไกร รังสิพล : *"เจาะระบบ TCP/IP จุดอ่อนของโพรโตคอลและวิธีป้องกัน"*, บริษัท ซีเอ็ดยูเคชั่น จำกัด 2001.

[7] สันติ ศรีลาศักดิ์ และ วรวุฒิ เที่ยงธรรม : ***"เจาะประเด็นงานเขียนโปรแกรมบนลินุกซ์"***, บริษัท ออฟ เพรส จำกัด, 1999.

[8] มนตรี พจนารถลาวัณย์ : ***"เขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี"***, บริษัท ซีเอ็ดยูเคชั่น จำกัด 1997.

[9] พ.อ. เจนวิทธ์ เหลืองอร่าม และ ปิยวิทย์ เหลืองอร่าม : ***"การเขียนโปรแกรมสำหรับ Application ด้วย C/C++"***, Se-Education Public Company Limited. 2002.

เว็บไซต์อ้างอิง

[1] http://www.openssl.org

[2] http://www.acm.uiuc.edu/webmonkeys/book/c_guide/