การพัฒนาความปลอดภัยของโปรแกรมเทลเนท

Development of Secure Telnet

โดย

นาย ชัยทัต ยุกตะเวทย์

นาย พงษ์พันธ์ พัฒนไพรสณฑ์

นาย อุเทน แซ่เต็ง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชา วิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปีการศึกษา 2539

การพัฒนาความปลอดภัยของโปรแกรมเทลเนท

Development of Secure Telnet

โดย

นาย ชัยทัต ยุกตะเวทย์

นาย พงษ์พันธ์ พัฒนไพรสณฑ์

นาย อุเทน แซ่เต็ง

อาจารย์ที่ปรึกษา

ธนา หงษ์สุวรรณ

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาความปลอดภัยของโปรแกรมเทลเนท

ผู้จัดทำ

1. นายชัยทัต ยุกตะเวทย์
2. นายพงษ์พันธ์ พัฒนไพรสณฑ์
3. นายอุเทน แซ่เต็ง

อาจารย์ที่ปรึกษา

(ธนา หงษ์สุวรรณ)

การพัฒนาความปลอดภัยของโปรแกรมเทลเนท

ชัยทัต ยุกตะเวทย์
พงษ์พันธ์ พัฒนไพรสณฑ์
อุเทน แซ่เต็ง
ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
ปีการศึกษา 2539

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการพัฒนาความปลอดภัยของโปรแกรมเทลเนทโดย โปรแกรมเทลเนทนี้เป็นโปรแกรมอรรถประโยชน์ที่ใช้ในระบบเครือข่ายอินเตอร์เน็ต (internet) ซึ่งทำให้ผู้ใช้บริการสามารถ login เข้ามาใช้บริการจากเครื่องอยู่ห่างไกลจาก Host ที่ให้บริการระบบ ปฏิบัติการที่เลือกใช้ในโครงงานนี้ คือ ระบบปฏิบัติการ LINUX 2.0.0 สำหรับเครื่องให้บริการ (SERVER) และระบบปฏิบัติการ WINDOWS สำหรับเครื่องรับบริการ (CLIENT) การพัฒนา ความปลอดภัยของโปรแกรมเทลเนททำได้โดยการเข้ารหัส และถอดรหัสข้อมูลที่เป็นความลับของ ผู้ใช้บริการ เช่น login name และ password

โครงงานนี้ทำการพัฒนาความปลอดภัยของโปรแกรม์เทลเนทโดยเลือกใช้อัลกอริทึม เข้ารหัสและถอดรหัส ด้วย RSA อัลกอริทึม เครื่องมือที่ใช้ในการแปลภาษา ได้แก่ Microsoft Visual Basic 4.0 โดยที่โครงงานนี้จะเป็นแนวทางในการพัฒนาระบบความปลอดภัยของโปรแกรม อรรถประโยชน์อื่นๆ ในระบบอินเตอร์เนทสำหรับผู้สนใจทั่วไป

Development of Secure Telnet

Chaithat Yuktawet
Pongpan Pattanapison
Utain Saeteng
Thana Hongsuwan Advisor
1996

ABSTRACT

This thesis presents the development of Secure Telnet. The telnet is a utility program on the Internet System. It is used for remote login from client to server. The selected operating system are LINUX 2.0.0 for server and Microsoft Windows 95 for client. This project use encrypt and decrypt the secret data for increasing security of telnet.

This project use RSA Algorithm for encrypt and decrypt the secret data. We used Microsoft Visual Basic 4.0 is a Tool of the project development. This project will be the example of the development Security in Utility program on the Internet system for interested people.

สารบัญ

บทที่ 1 บทนำ 1

1.1 ที่มาของโครงงาน 1

1.2 จุดประสงค์ของโครงงานนี้ 1

1.3 ขอบเขตของงานโครงงาน 2

1.3.1 Telnet client 2

1.3.2 Telnet server 2

1.4 ประโยชน์ที่คาดว่าจะได้รับจากโครงงานนี้ 2

1.5 เนื้อหาโดยย่อ 2

บทที่ 2 ความรู้เบื้องต้นเกี่ยวกับระบบเครือข่ายโปรโตคอลและเทลเนท 4

2.1 ระบบเครือข่าย 4

2.1.1 อินเตอร์เน็ตเวิร์คกิ้ง (Internetworking) 4

2.1.2 โมเดล OSI 5

2.1.3 ลักษณะของการติดต่อ 6

2.1.3.1 Connection-Oriented 6

2.1.3.2 Connection-less หรือดาต้าแกรม (Datagram) 6

2.2 โปรโตคอล TCP/IP (TCP/IP Protocol) 6

2.2.1 เปรียบเทียบระหว่างโปรโตคอล TCP/IP Protocol) 6

2.2.1.1 ชั้นแอพพลิเคชั่น (Application Layer) 7

2.2.1.2 ชั้นทรานสปอร์ต (Transport Layer) 7

2.2.1.3 ชั้นอินเตอร์เน็ต (Internet Layer) 7

2.2.1.4 ชั้นฟิสิคอล (Physical Layer) 8

2.2.2 รูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP/IP 8

2.2.2.1 การแบ่งเนตเวิร์คคลาส (Network Class) 8

2.2.2.2 การแทนด้วยเลขฐานสิบและจุด (Dotted Decimal Notation) 9

2.2.2.3 การทำซับเนทติ้ง (Subnetting) 10

สารบัญ (ต่อ)

2.2.3 ชุดโปรโตคอล TCP/IP (TCP/IP Protocol Suite) 10

2.2.3.1 โปรโตคอล IP (Internet Protocol) 11

2.2.3.2 โปรโตคอล ARP (Address Resolution Protocol 12

2.2.3.3 โปรโตคอล ICMP (Internet Control Message Protocol) 12

2.2.3.4 โปรโตคอล RARP (Reverse Address Resolution Protocol) 12

2.2.3.5 โปรโตคอล UDP (User Datagram Protocol) 12

2.2.3.6 โปรโตคอล TCP (Transmission Control Protocol) 12

2.2.4 หมายเลขพอร์ท 13

2.3 เทลเนท (TELNET) 13

2.3.1 ภาพโดยรวมของเทลเนท 13

2.3.1.1 เทลเนทคืออะไร ? 13

2.3.1.2 เทลเนททำงานอยู่ที่ Layer ใด ใน TCP/IP โปรโตคอล และอยู่บน Port บริการที่เท่าใด ? 14

2.3.1.3 เทลเนทมีการทำงานเบื้องต้นอย่างไรบ้าง 15

2.3.2 Network Virtual Terminal (NVT) 16

2.3.2.1 โครงสร้างและหลักการของ NVT 16

2.3.2.2 การทำงานและ Option ของ NVT ใน Telnet 17

2.3.3 Telnet model 18

2.3.3.1 โครงสร้างของ Telnet model และการทำงาน 18

2.3.3.2 Mode การส่งข้อมูลของเทลเนท 21

2.3.3.2.1 Mode การส่งข้อมูลแบบ line-at-a-time 21

2.3.3.2.2 Mode การส่งข้อมูลแบบ character-at-a-time 22

2.4 เทอร์มินอลเสมือน (Visual Terminal) 24

2.5 คำสั่งและการควบคุมของเทลเนท (Command and Control Information) 24

2.6 เทลเนทไคลเอนท์อัลกอริทึม (Telnet Client Algorithm) 25

2.7 อัลกอริทึมของเทลเนทไคลเอนท์ (Algorithm of Telnet client) 25

2.8 เทอร์มินอล I/O ในยูนิกซ์ (Terminal I/O In UNIX) 25

สารบัญ (ต่อ)

บทที่ 3 การเข้ารหัส 28

3.1 ทฤษฎีการเข้ารหัสแบบ Caesar Cipher 28

3.2 ทฤษฎีการเข้ารหัสแบบ Vernam Cipher 28

3.3 ทฤษฎีการเข้ารหัสแบบ Columnar Transpositions 29

3.4 ทฤษฎีการเข้ารหัสแบบ Merkle-Hellman Knapsacks 30

3.5 ทฤษฎีการเข้ารหัสแบบ DES Algorithm 31

3.5.1 ประวัติความเป็นมา 31

3.5.2 ลักษณะของ DES Algorithm 33

3.5.3 รายละเอียดของการเข้ารหัส 33

3.5.4 รายละเอียดในวัฏจักร (cycle) ของ DES อัลกอริทึม 38

3.5.5 เอกซ์แพนชั่น เพอร์มิวเตชั่น (Expansion Permutation) 38

3.5.6 การเปลี่ยนคีย์ (Key Transformation) 40

3.5.7 กล่องเอส (S-Boxes) 41

3.5.8 กล่องพี (P-Boxes) 42

3.5.9 การถอดรหัส DES 44

3.5.10 การวิเคราะห์อัลกอริทึม 46

3.6 ทฤษฎีการเข้ารหัสแบบ RSA Algorithm 47

3.6.1 ลักษณะการทำงานของ RSA 47

3.6.2 การใช้ RSA ในการเข้ารหัส 48

3.6.3 การใช้ RSA แสดงข้อมูล 48

3.6.4 การเข้ารหัสของ RSA 49

3.6.5 รายละเอียดและวิธีของการเข้ารหัส 49

3.6.6 หลักการทางคณิตศาสตร์ที่ใช้ในการเข้ารหัสแบบ RSA 50

3.6.7 การนำไปใช้งาน 51

3.6.8 รูปแสดงการเข้ารหัสแบบ RSA 53

3.6.9 การทำ Fast Exponential ใน RSA 53

3.6.10 ตัวอย่างการเข้ารหัสแบบ RSA 55

สารบัญ (ต่อ)

บทที่ 4 การออกแบบ 59

4.1 โครงสร้างและส่วนประกอบของโปรแกรม Secure Telnet 59

4.2 Telnet server (Telnet d) 61

4.2.1 การแยก Port บริการ 61

4.2.2 การแสดงความเป็น Secure Telnet ของ Telnet server 62

4.2.3 การเปลี่ยนแปลงสถานะของการถอดรหัสข้อมูล 62

4.2.4 การทำงานโดยรวมของ Telnet server ของ Secure Telnet 65

4.3 Telnet Client 66

4.3.1 การตรวจสอบและติดต่อกับเครื่องให้บริการที่เป็น Secore Telnet 66

4.3.2 การแสดงผลโดยการทำเทอร์มินอลเสมือน 67

4.3.3 การทำงานโดยรวมของ Telnet Client ของ Secure Telnet 68

4.4 การออกแบบส่วนการส่งข้อมูลที่เข้ารหัส 69

บทที่ 5 การทดลองและผลการทดลอง 70

5.1 การทดสอบโปรแกรมในส่วนของ Telnet server 70

5.1.1 การทดสอบส่วนของการแสดงตัวว่าเป็น Secure Telnet 70

5.1.2 การทดสอบส่วนของการถอดรหัสของ Telnet Server 70

5.2 การทดสอบโปรแกรมในส่วนของ Telnet client 71

5.2.1 การทดสอบส่วนของฟังค์กั่นเข้ารหัสของ Telnet Client 71

5.2.2 ทดสอบการตรวจสอบเครื่องให้บริการของ Telnet Client 71

5.2.2.1 วิธีทดสอบว่า login กับเครื่องให้บริการที่มี Secure Telnet 71

5.2.2.2 วิธีทดสอบว่า login กับเครื่องให้บริการที่ไม่มี Secure Teonet 72

5.3 การทดสอบความปลอดภัยของโปรแกรม Secure Telnet 72

5.3.1 การเตรียมอุปกรณ์สำหรับการทดสอบ 72

5.3.2 ขั้นตอนการทดสอบ 73

สารบัญ (ต่อ)

บทที่ 6 บทวิจารณ์และสรุป 76

6.1 บทวิจารณ์ 76

6.2 แนวทางการปรับปรุงโครงงาน 76

6.2.1 การปรับปรุงขนาดของคีย์ในการเข้ารหัส 76

6.2.2 การปรับปรุงให้การเข้ารหัสใช้คีย์มากกว่าหนึ่งคีย์ 79

6.3 บทสรุป 81

ภาคผนวก 82

กิตติกรรมประกาศ

หนังสืออ้างอิง

บทที่ 1
บทนำ

1.1 ที่มาของโครงงาน

เทลเนท (Telnet) เป็นโปรแกรมประยุกต์ (Application Program) ที่ใช้ในอินเตอร์เนต (Internet) โดยให้บริการกับผู้ใช้ที่อยู่ไกลให้สามารถลอคอิน (login) เข้ามาใช้เครื่องให้บริการจากเครื่องอื่นที่อยู่ห่างไกลได้

การใช้บริการเทลเนท ผู้ใช้จะต้องส่งชื่อผู้ใช้บริการ (login name) และรหัสผ่าน (Password) ของผู้ใช้มาด้วย ซึ่งในเทลเนทปัจจุบันนี้ส่วนใหญ่ยังมิได้มีการป้องกันข้อมูลเหล่านี้ขณะที่ส่งอยู่ในระบบเครือข่าย (Network) คือ ยังมิได้มีการเข้ารหัสข้อมูลเหล่านี้ในการส่ง ดังนั้นถ้ามีผู้ประสงค์ร้ายดักแพคเก็ต (Packet) ข้อมูลเหล่านี้ในระบบเครือข่าย และแกะแพคเก็ตเหล่านั้นออกดูก็จะได้ข้อมูลที่เป็นความลับของผู้ใช้คนนั้น ๆ ไป

ซึ่งจากเหตุการณ์ดังกล่าวจึงเป็นที่มาของโครงงานนี้ โดยในโครงงานนี้ได้สร้างโปรแกรมที่ชื่อว่า Secure Telnet ขึ้น ซึ่ง Secuve Telnet นี้นั้นได้มีการเข้ารหัสข้อมูลที่เป็นความลับของผู้ใช้ซึ่งก็คือ รหัสผ่านของผู้ใช้และเมื่อเรานำข้อมูลรหัสผ่านนี้ไปเข้ารหัสก่อนที่จะส่งไปให้เครื่องให้บริการนี้แล้ว แม้ว่าจะมีผู้ดักแพคเก็ตเอาไปแกะดูก็จะได้รหัสผ่านที่ใช้ไม่ได้มีสภาพเป็นเหมือนขยะ ซึ่งถึงแม้ว่าการเข้ารหัสข้อมูลรหัสผ่านของผู้ใช้นี้จะยังมิได้มีความปลอดภัยร้อยเปอร์เซ็นต์ แต่ก็ทำให้ความปลอดภัยของการใช้โปรแกรมเทลเนทมีความปลอดภัยสูงขึ้นกว่าเดิม ซึ่งเป็นที่มาของชื่อของโครงงานนี้นั่นเอง

1.2 จุดประสงค์ของโครงงานนี้

- เมื่อเพิ่มความปลอดภัยให้กับโปรแกรมที่ใช้ในระบบอินเตอร์เนทที่นิยมกันอยู่ในปัจจุบัน
- เมื่อเพิ่มความปลอดภัยในการใช้งานโปรแกรมเทลเนท
- เมื่อศึกษาวิธีการเพิ่มความปลอดภัยในการใช้ระบบอินเตอร์เนท
- เมื่อศึกษาการเขียนโปรแกรมประยุกต์ที่ใช้งานบนระบบอินเตอร์เนท
- เมื่อศึกษาการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการ windows โดยใช้ window socket
- เพื่อศึกษาการทำงานของโปรแกรมในสภาพแวดล้อมของระบบปฏิบัติการ UNIX

1.3 ขอบเขตของโครงงานนี้

ขอบเขตของโครงงานนี้ มีดังนี้

1.3.1 Telnet client

- ต้องสามารถ login เข้ากับเครื่องที่มี Secuve Telnet และเครื่องไม่มี Secure Telnet ได้
- ใช้การเข้ารหัสแบบ RSA โดยมีขนาดของ Key เข้ารหัส 16 บิต
- ทำงานภายใต้ระบบปฏิบัติการ Windows
- ใช้ Window socket version 1.1 ขึ้นไป

1.3.2 Telnet server

- ต้องสามารถรองรับการ login ได้ทั้ง Secure Telnet และ Telnet ปกติ
- ทำการถอดรหัสแบบ RSA โดยมีคีย์ขนาด 16 บิต
- ทำงานอยู่บนระบบปฏิบัติการยูนิกซ์ (UNIX) บน PC คือ LINUX

1.4 ประโยชน์ที่คาดว่าจะได้รับจากโครงงานนี้

ประโยชน์ที่คาดว่าจะได้รับจากโครงงานนี้มีดังนี้ คือ

- โปรแกรมเทลเนทที่ได้จากโครงงานนี้ มีความปลอดภัยและสามารถนำไปใช้ได้จริง
- เป็นแนวทางในการพัฒนาความปลอดภัยของโปรแกรมประยุกต์บนระบบอินเตอร์เนท
- เป็นแนวทางการพัฒนาโปรแกรมภายใต้ระบบปฏิบัติการ Windows
- เป็นแนวทางในการพัฒนาโปรแกรมประยุกต์บนระบบอินเตอร์เนท

1.5 เนื้อหาโดยย่อ

โครงงานนี้มีส่วนประกอบหลัก ๆ 2 ส่วน คือ

- โปรแกรมเทลเนทและการติดต่อของโปรแกรมเทลเนท
- การเข้ารหัสและวิธีการเข้ารหัส

โดยในแต่ละบทมีเนื้อหาโดยย่อดังนี้

บทที่ 2 เป็นเนื้อหาของโปรแกรมเทลเนทและโปรโตคอล TCP/IP ซึ่งเป็นโปรโตคอลที่ใช้ในเทลเนท

บทที่ 3 เป็นเนื้อหาวิธีการเข้ารหัสแบบต่างๆ

บทที่ 4 เป็นการออกแบบโครงสร้างและวิธีการทำงานของโปรแกรม Secure Telnet

บทที่ 5 เป็นการทดสอบโปรแกรมในส่วนต่างๆ ของโปรแกรม Secure Telnet

บทที่ 6 เป็นผลสรุปของโครงงานและสิ่งที่ยังต้องพัฒนาต่อของโครงงานนี้

บทที่ 2
ความรู้เบื้องต้นเกี่ยวกับระบบเครือข่าย โปรโตคอล และเทลเนท

2.1 ระบบเครือข่าย

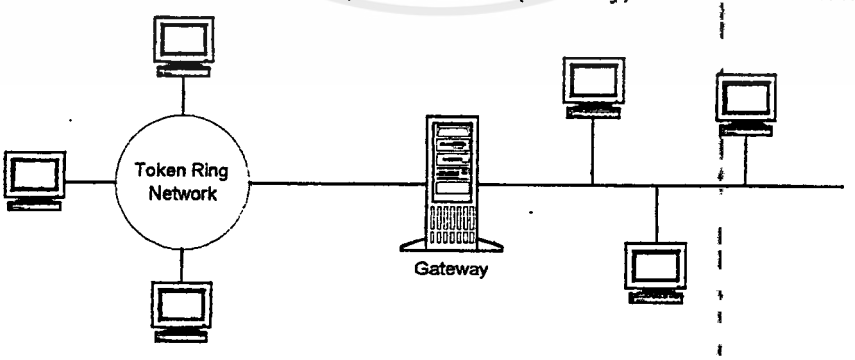
เนื่องจากเทลเนท เป็นโปรแกรมประยุกต์บนระบบเครือข่ายโปรแกรมหนึ่ง ดังนั้น การพัฒนาจึงจำเป็นต้องมีความรู้พื้นฐานทางด้านระบบเครือข่ายพอสมควร ซึ่งในหัวข้อนี้จะอธิบายความหมายและหลักการสำคัญของระบบเครือข่ายที่จำเป็นในการพัฒนาโปรแกรม Telnet ในโครงงานนี้ ดังที่จะกล่าวต่อไป

ความหมายของระบบเครือข่ายคอมพิวเตอร์และอินเตอร์เน็ตเวิร์คกิ้ง

2.1.1 อินเตอร์เน็ตเวิร์คกิ้ง (Internet working)

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) คือระบบการเชื่อมต่อระหว่างระบบปลายทาง (End-System) ซึ่งระบบปลายทางเป็นระบบที่เป็นอิสระจากกัน (Autonomous) ระบบปลายทางสามารถเป็นได้ตั้งแต่ไมโครคอมพิวเตอร์ (Microcomputer) ไปจนกระทั่งซุปเปอร์คอมพิวเตอร์ (Supercomputer) ขนาดใหญ่เพื่อจุดมุ่งหมายในการแลกเปลี่ยนข้อมูลและการแบ่งปันทรัพยากรของระบบ เช่น ไฟล์ (File) ข้อมูล, เครื่องพิมพ์ (Printer), โมเด็ม (Modem) ตลอดจนการให้บริการฐานข้อมูลร่วม (Sharing database)

อินเตอร์เน็ทเวิร์คกิ้ง หรืออินเตอร์เน็ท (Internet) คือ การเชื่อมต่อของระบบเครือข่าย 2 เครือข่ายขึ้นไป ดังนั้น คอมพิวเตอร์บนระบบเครือข่ายหนึ่งก็สามารถติดต่อกับคอมพิวเตอร์บนระบบเครือข่ายอื่นๆ ได้ เช่น เน็ตเวิร์คโทเคนริงก์ (Tokenring network) เชื่อมกับเน็ตเวิร์คอีเทอร์เน็ต (Ethernet network) โดยมีเกตเวย์ (Gateway) เป็นตัวเชื่อม ดังภาพ



รูปที่ 2.1 ระบบเครือข่ายคอมพิวเตอร์เบื้องต้น

2.1.2 โมเดล OSI

OSI เป็นคำย่อที่มาจากคำว่า Open System Interconnection โดยที่เป็นมาตรฐานที่ถูกเสนอขึ้นโดย International Standards Organization ซึ่งเป็นองค์กรที่จัดตั้งขึ้นมาเพื่อดูแล และส่งเสริมตลอดจนกำหนดมาตรฐานของการติดต่อสื่อสารของระบบเครือข่ายคอมพิวเตอร์ โดยโมเดล OSI นี้มีลักษณะเป็นสถาปัตยกรรมแบบระบบเปิด (Open System) เพราะมุ่งที่จะให้ระบบคอมพิวเตอร์ในหลาย ๆ รูปแบบที่แตกต่างกันสามารถเชื่อมต่อกันได้ OSI โมเดลได้แบ่งโปรโตคอล (protocol) ในการสื่อสารออกเป็น 7 ชั้น (layer) ซึ่งโปรโตคอล คือชุดของกฎ หรือข้อตกลงในการติดต่อ ข้อสังเกตโมเดล OSI เป็นเพียงข้อเสนอแนะ มิใช่ข้อกำหนด และควรรู้ว่ายังไม่มีระบบ

1. Application Layer
2. Presentation Layer
3. Session Layer
4. Transport Layer
5. Network Layer
6. Datalink Layer
7. Physical Layer

รูปที่ 2.2 OSI โมเดลทั้ง 7 เลเยอร์

การเชื่อมต่อใดที่สร้างเหมือนกับโมเดล OSI จริง ๆ ในชั้นหนึ่งๆ ไม่ได้การกำหนดว่าจะต้องมีเพียงหนึ่งโปรโตคอลเท่านั้น ที่อยู่ในระบบชั้นเดียวกัน และในทางตรงข้าม ชุดของโปรโตคอลใด ๆ อาจจะมีมากกว่าหนึ่งเลเยอร์ประกอบกันเป็นข้อกำหนดของระบบเครือข่ายเรียกว่าชุดโปรโตคอล (Prococol Suite) เช่น ชุดโปรโตคอล TCP/IP (Transmission Control Protocol / Internet Protocol) เป็นต้น ประโยชน์ในการแบ่งเป็นชั้น คือ กำหนดการติดต่อระหว่างชั้น ทำได้โดยไม่ต้องคำนึงถึงการเปลี่ยนในชั้นใด ๆ ที่ติดกัน

2.1.3 ลักษณะของการติดต่อ

แบ่งออกเป็น 2 ชนิด คือ

2.1.3.1 Connection - Oviented

คือ การติดต่อที่ต้องมีการเชื่อมโปรเซสที่จะทำการติดต่อก่อนที่จะมีการส่งหรือรับข้อมูล ซึ่งสามารถใช้คำว่าวงจรเสมือน (Virtual Circuit) เพราะว่าจะทำงานเสมือนมีวงจรต่ออยู่ระหว่างโปรเซส ถึงแม้ว่าข้อมูลนี้อาจจะผ่าน Packet - Switching Network บริการชนิดนี้ส่วนมากจะใช้ในกรณีที่มีข่าวสารต้องการมากกว่าหนึ่งข่าวสาร ดังนั้น สามารถแบ่งขั้นการทำงานออกเป็น

- ขั้นการสร้างการติดต่อ (connection establishment)
- ขั้นการส่งผ่านข้อมูล (data transfer)
- ขั้นยกเลิกการติดต่อ (connection termination)

ซึ่งลักษณะติดต่อแบบนี้นั้นเองที่โปรแกรม Telnet ใช้ในการติดต่อ

2.1.3.2 Connection less หรือ ดาต้าแกรม (Datagram)

คือ จะไม่มีขั้นการสร้างการติดต่อ และขั้นการยกเลิกการติดต่อ แต่จะมีขั้นการส่งผ่านข้อมูลอย่างเดียว โดยข้อมูลเรียกว่าดาต้าแกรมจะถูกส่งจากระบบหนึ่งไปสู่ระบบหนึ่งอย่างเป็นอิสระโดยไม่ขึ้นอยู่กับดาต้าแกรมอื่น

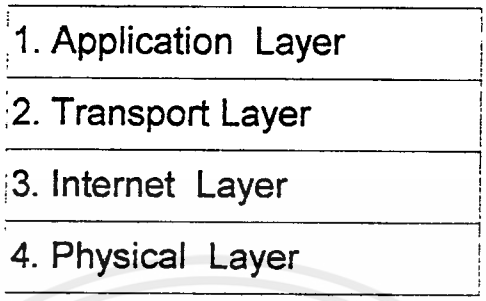
2.2 โปรโตคอล TCP/IP (TCP/IP PROTOCOL)

เนื่องจากเทลเนท เป็นโปรแกรมประยุกต์ที่ใช้โปรโตคอล TCP / IP ในการทำการติดต่อ ดังนั้น เราจึงจำเป็นต้องมีความรู้เกี่ยวกับโปรโตคอล TCP / IP บ้าง เมื่อนำไปใช้ในการพัฒนาโปรแกรม Telnet ในโครงงานนี้

2.2.1 เปรียบเทียบระหว่างโปรโตคอล TCP / IP และ OSI โมเดล

การออกแบบโปรโตคอล TCP / IP นั้น ไม่ได้เป็นไปตามรูปแบบของ OSI โมเดลเนื่องด้วย ถูกออกแบบโดยองค์กรขนาดใหญ่ซึ่งใช้เวลานานในการออกแบบตลอดจนการรับรองมาตรฐานต่างกับโปรโตคอล TCP / IP ที่ถูกแบบด้วยความต้องการอันเร่งด่วนของรัฐบาลสหรัฐจึงทำให้การพัฒนาโปรโตคอล TCP / IP มีเงื่อนไขของในด้านความต้องการที่ต่างจาก OSI โมเดล

ซึ่งหากเรามองโดยรวมแล้วจะเห็นว่าโปรโตคอล TCP / IP มีการแบ่งเป็นเลเยอร์ที่น้อยกว่า OSI โมเดล คือ มี 4 ชั้นเท่านั้น ดังภาพโดยแบ่งเป็น



รูปที่ 2.3 เลเยอร์ของโปรโตคอล TCP / IP

2.2.1.1 ชั้นแอพพลิเคชั่น (Application Layer)

ในชั้นนี้ประกอบโปรแกรมประยุกต์ที่ใช้เครือข่าย เช่น โปรแกรมส่งถ่ายข้อมูล (file - transfer programe) และอาจกล่าวได้ว่าชั้นนี้โปรโตคอล TCP / IP ก็คือ เลเยอร์ในชั้นแอพพลิเคชั่น รวมกับชั้นพรีเซนเตชั่น (Presentation layer) ใน OSI โมเดลนั้นเอง และในชั้นนี้ของโปรโตคอล TCP / IP จะกลืนอยู่ในตัวโปรแกรมประยุกต์

2.2.1.2 ชั้นทรานสปอร์ต (Transport Layer)

ในชั้นนี้เป็นชั้นที่ให้การส่งข้อมูลจากจุดปลายถึงจุดปลายหากเปรียบเทียบกับ OSI โมเดล ก็สามารถเทียบได้กับชั้นเซสชั่น (Session layer) รวมกับชั้นทรานสปอร์ตนั่นเอง โดยโปรโตคอล TCP / IP มีซอคเก็ต (socket) เป็นจุดปลาย (end - point) ในการสื่อสาร ซึ่งซอคเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์ และหมายเลขพอร์ต (port) ของเครื่องที่ต้องการส่งข้อมูลไปถึง ในชั้นนี้มีการรับรองการถึงที่หมาย และลำดับของข้อมูลที่ส่งโดยไม่ซ้ำ และความผิดพลาดข้อมูล

2.2.1.3 ชั้นอินเตอร์เน็ต (Internet Layer)

ในชั้นนี้มีการกำหนดดาต้าแกรมและทำการหาเส้นทางการส่ง หน้าที่ของชั้นนี้เทียบเท่ากับชั้นเน็ตเวอร์ก (Network layer) และชั้นดาต้าลิงค์ (Datalink layer) ของ OSI โมเดล

2.2.1.4 ชั้นฟิสิคอล (Physical Layer)

โปรโตคอล TCP / IP ไม่ได้กำหนดรูปแบบของการเชื่อมต่อในระดับนี้ไว้ใหม่ แต่ได้ใช้มาตรฐานที่มีอยู่เดิมที่กำหนดไว้ก่อน เช่น RS232, อีเทอร์เน็ต (Ethernet) เป็นต้น

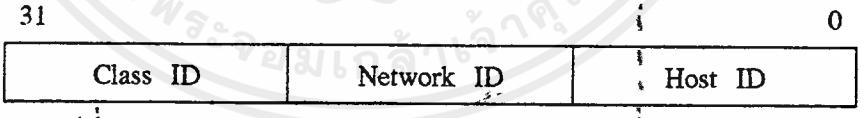
2.2.2 รูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP / IP

ในหัวข้อนี้จะได้อธิบายรูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP / IP ซึ่งลักษณะแอดเดรสของโปรโตคอลนี้ค่าของแอดเดรสของเครื่องคอมพิวเตอร์ในระบบเครือข่ายจะไม่ซ้ำกันเลย โดยเลขนี้เรียกว่า IP แอดเดรส (IP Address) เป็นเลข 32 บิต ซึ่งแบ่งเป็นคลาส (Class) ตามหลักในการพิจารณาที่จะได้กล่าวต่อไปนี้

2.2.2.1 การแบ่งเนตเวิร์คคลาส (Network Classes)

เนื่องจากหมายเลขแอดเดรสของคอมพิวเตอร์เครื่องใด ๆ นั้น จะต้องสามารถบอกถึงความแตกต่างระหว่างตัวเครื่องเอง ตลอดจนเครือข่ายที่คอมพิวเตอร์นั้นเชื่อมต่ออยู่ด้วย หมายเลข IP แอดเดรส จึงแยกออกเป็น 2 ส่วนได้แก่ ส่วนที่แสดงหมายเลขของคอมพิวเตอร์โฮสต์ และส่วนที่เป็นหมายเลขของเครือข่าย

การแบ่งคลาสของแอดเดรสทำได้โดยพิจารณาจำนวนบิตของ 2 ส่วนประกอบข้างต้น ซึ่งมีการแบ่งออกเป็น 5 คลาส แต่มีการใช้เพียง 3 คลาสแรก คือ คลาส A, คลาส B และคลาส C ส่วนคลาส D และ E ถูกสงวนไว้สำหรับจุดประสงค์พิเศษ

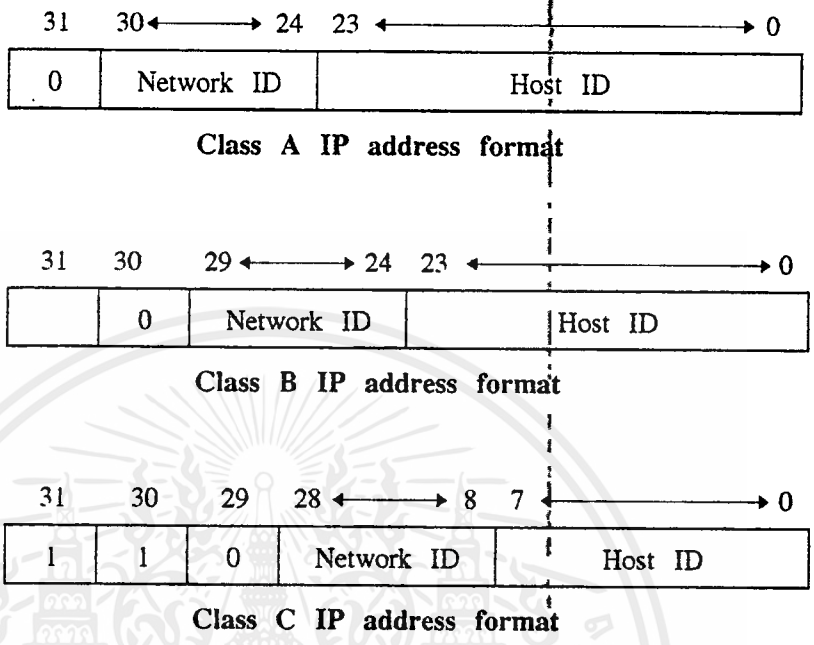


รูปที่ 2.4 IP address format (หมายเลข IP แอดเดรส)

Network Class	Networks	Hosts per Network
A	126	16,777,214
B	16,382	65,534
C	2,097,150	254

รูปที่ 2.5 แสดงคลาส, จำนวนเครือข่าย และจำนวนโฮสต์ของแต่ละคลาส

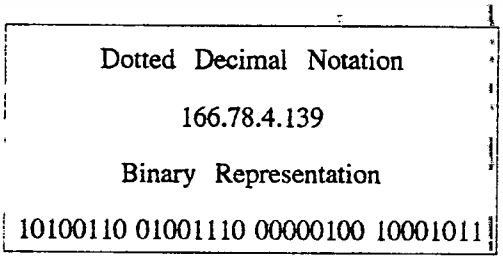
โดยปกติแล้วผู้พัฒนาโปรแกรมไม่ต้องสนใจความแตกต่างระหว่างคลาสของ IP แอดเดรส



รูปที่ 2.6 แสดงรูปแบบของ IP แอดเดรสในคลาสต่าง ๆ

2.2.2.2 การแทนด้วยเลขฐานสิบและจุด (Dotted Decimal Notation)

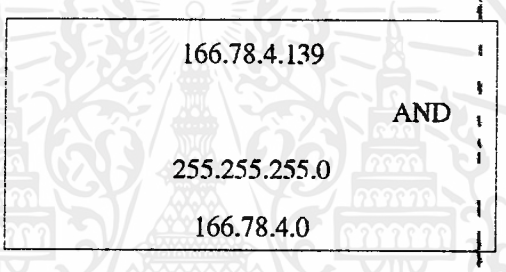
เนื่องจากการแทนหมายเลข IP แอดเดรสเป็นเลขฐาน 2 ซึ่งค่อนข้างอ่านไม่สะดวก จึงมีการแทนเลขฐานสองเหล่านั้นในรูปเลขฐานสิบและจุด โดยเลขฐานสิบแต่ละตัวจะแทนเลขฐานสองจำนวน 8 บิต โดยระหว่างเลขฐานสิบแต่ละตัวจะแทรกด้วยจุด ดังนั้นจะต้องใช้เลขฐานสิบ 4 ตัว ในการแทนเลข 32 บิต ที่เป็น IP แอดเดรส ดังตัวอย่างจะสังเกตุว่าหมายเลข IP แอดเดรสนี้จัดอยู่ในคลาส B โดยหมายเลขของเครือข่ายคือ 166.78 และหมายเลขประจำเครื่องคือ 4.139



รูปที่ 2.7 ตัวอย่างหมายเลข IP แอดเดรสทั้งสองแบบ

2.2.2.3 การทำซับเน็ตติ้ง (Subnetting)

การทำซับเน็ตติ้งเป็นการเปลี่ยนแปลงการใช้หมายเลขของเครื่องโฮสต์และหมายเลขของเครือข่ายในระดับท้องถิ่น โดยในทางตรรก คือ การเลื่อนเส้นแบ่งที่แยกหมายเลขเครื่อง และหมายเลขของเน็ตเวอร์กที่อยู่ในหมายเลข IP แอดเดรส โดยที่ปริมาณของหมายเลขเครื่องโฮสต์และหมายเลขเครือข่ายจะแปรผกผันกัน เช่น หากมีปริมาณของเน็ตเวอร์กมาก ก็จะทำให้เครื่องใดๆ ที่จะต่อกับระบบเครือข่ายหนึ่ง ๆ จะน้อยลงเป็นต้น ในทางปฏิบัติการทำซับเน็ตติ้งทำโดยการนำซับเน็ตมาร์ค (Subnet mark) คือตัวเลขจำนวน 32 บิต มาทำการกระทำตรรกและ (AND) กับหมายเลข IP แอดเดรส ดังตัวอย่างโดยกำหนดหมายเลข IP แอดเดรส คือ 166.78.4.139 และซับเน็ตมาร์ค คือ 255.255.255.0 ทำการกระทำตรรก ดังรูป

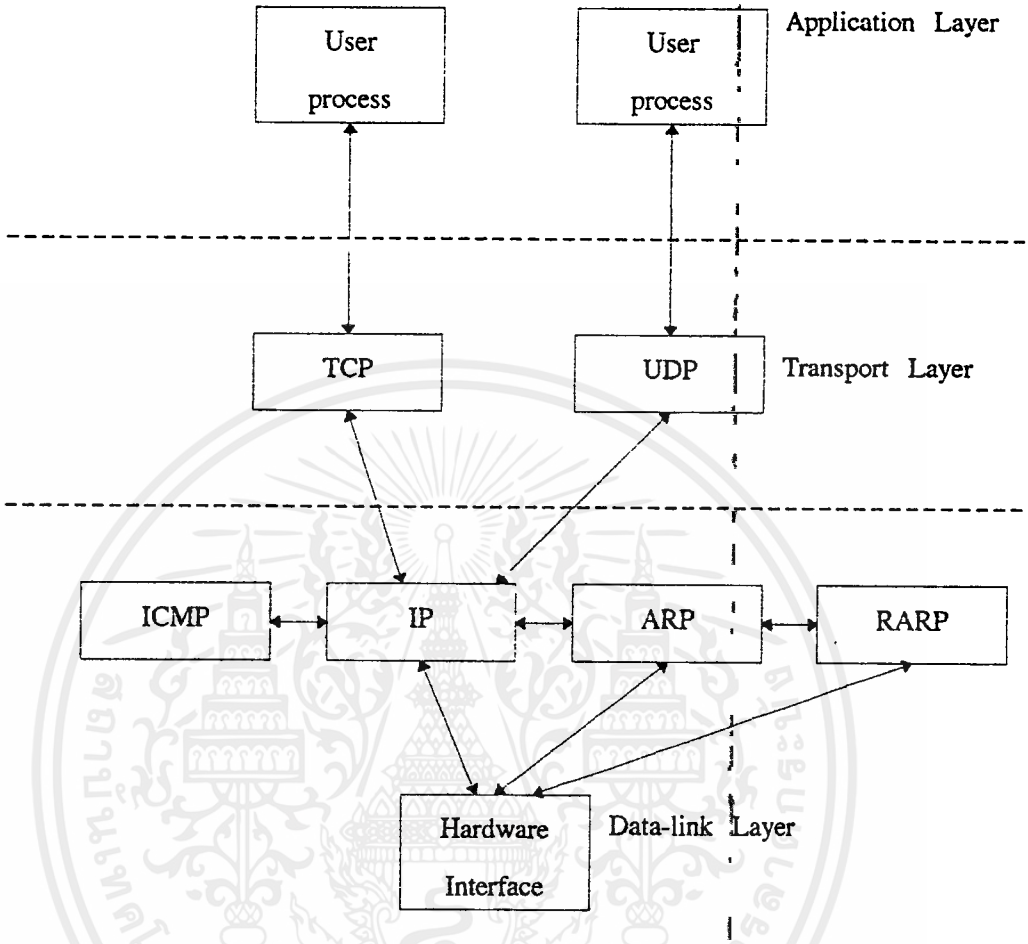


รูปที่ 2.8 การทำซับเน็ตติ้ง

จะเห็นว่าหากพิจารณาโดยไม่มีการทำซับเน็ตติ้งแล้วจะได้หมายเลขของเน็ตเวอร์ก คือ 166.78 และหมายเลขประจำเครื่อง คือ 4.139 แต่ผลที่ได้จากการกระทำตรรกและในรูปข้างต้น ผลลัพธ์ที่ได้คือ 166.78.4.0 และที่เหลือคือ 139 หากพิจารณาผลลัพธ์นี้ คือหมายเลขเน็ตเวอร์กคือ 166.78.4.0 และหมายเลขเครื่อง คือ 139 หรืออาจพูดได้ว่าคอมพิวเตอร์เครื่องนี้มีหมายเลขเครื่องเท่ากับ 139 และอยู่บนเครือข่ายย่อยหมายเลข 166.78.4

2.2.3 ชุดโปรโตคอล TCP/IP (TCP/IP Protocol Suite)

ชุดโปรโตคอล TCP/IP นอกจากมีโปรโตคอล TCP และ IP แล้ว ยังมีโปรโตคอลอย่างอื่นอีก ดังภาพแสดงความสัมพันธ์ของชุดโปรโตคอลโดยแบ่งตามเลเยอร์



รูปที่ 2.9 ความสัมพันธ์ระหว่างชุดโปรโตคอล

2.2.3.1 โปรโตคอล IP (Internet Protocol)

โปรโตคอล IP เป็นโปรโตคอลแบบคอนเนคชั่นเลส (Connectionless protocol) ซึ่งได้กล่าวถึงลักษณะของโปรโตคอลชนิดนี้ไปแล้ว โดยที่โปรโตคอล IP ไม่รับประกันว่าข้อมูลที่ส่งจะไปถึงปลายทางซึ่งแพคเก็ตของข้อมูลอาจไปถึงในลักษณะที่ผิดลำดับ, ซ้ำกัน หรือไปไม่ถึงเลย โดยความน่าเชื่อถือของการส่งจะถูกควบคุมในโปรโตคอลในเลเยอร์ต่างๆ ไป การหาเส้นทางของข้อมูลจะทำในระดับของโปรโตคอล IP นี้ โดยพิจารณาแต่ละแพคเก็ตแยกออกจากกันและยังมีหน้าที่ในการจัดเรียงข้อมูลใหม่ที่ปลายทางอีกด้วย

2.2.3.2 โปรโตคอล ARP (Address Resolution Protocol)

โปรโตคอลนี้ทำหน้าที่จับคู่ระหว่างหมายเลข IP แอดเดรสเข้ากับหมายเลขแอดเดรสทางฮาร์ดแวร์ โดยโปโตคอลนี้ทำการส่งข้อความไปทั่วเครือข่ายท้องถิ่น ซึ่งข้อความนี้เป็นลักษณะข้อความที่ตรวจสอบว่ามีคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกับที่ต้องการหาหรือไม่ หากคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกันนั้นได้รับข้อความนี้ก็จะตอบกลับและเป็นที่น่าสังเกตว่าโปรโตคอลนี้ทำงานได้กับระบบเครือข่ายท้องถิ่นเท่านั้น เพราะว่าโครงสร้างหมายเลขทางฮาร์ดแวร์ของเครื่องคอมพิวเตอร์จะขึ้นอยู่กับชนิดของระบบเครือข่ายด้วย

2.2.3.3 โปรโตคอลICMP (Internet Control Message Protocol)

เป็นโปโตคอลที่จัดการเกี่ยวกับข่าวสารความผิดพลาดและการควบคุมเกตเวย์ และเครื่องคอมพิวเตอร์ในระบบเครือข่าย

2.2.3.4 โปรโตคอลRARP (Reverse Address Resolution Protocol)

เป็นโปรโตคอลที่ทำหน้าที่จับคู่ระหว่างหมายเลขของฮาร์แวร์กับหมายเลข IP แอดเดรส หรือทำงานกลับกันกับโปรโตคอล ARP

2.2.3.5 โปรโตคอลUDP (User Datagram Protocol)

โปรโตคอลนี้เป็นโปรโตคอลที่อยู่ในระดับทรานสปอร์ตเลเยอร์ และมีความสำคัญเพราะว่าเป็นโปรโตคอลที่ผู้พัฒนาโปรแกรมสามารถใช้ได้โดยตรง โปรโตคอลนี้เป็นโปรโตคอลแบบคอนเนคชั่นเลสมีความน่าเชื่อถือต่ำ ไม่รับรองว่าข้อมูลที่ส่งไปจะไปถึงปลายทางหรือไม่ และอาจซ้ำซ้อนหรือผิดลำดับได้ แต่ข้อดีของโปรโตคอลนี้ คือ ค่าความสิ้นเปลือง (overhead) ที่ต่ำ

2.2.3.6 โปรโตคอลTCP (Transmission Control Protocol)

โปรโตคอลนี้อยู่ในระดับทรานสปอร์ตเลเยอร์ เหมือนกับโปรโตคอล UDP แต่มีลักษณะที่ตรงข้ามกัน คือ เป็นโปรโตคอลแบบคอนเนคชั่นออเรียนเต็ด โดยจะมีความน่าเชื่อถือในการรับส่งข้อมูล และลำดับของข้อมูลจะมีลำดับเหมือนกับต้นทางและเนื้อข้อมูลไม่ผิดพลาด จึงทำให้เกิดความสิ้นเปลืองในการเชื่อมต่อของการส่งข้อมูลมากกว่าโปรโตคอล UDP

2.2.4 หมายเลขพอร์ต

เนื่องจากในเวลาใด ๆ สามารถมีโพรเซสของผู้ใช้สามารถใช้ UDP หรือ TCP ได้พร้อมกัน ดังนั้นจึงต้องมีวิธีแยกแยะว่าข้อมูลเป็นของผู้ใช้คนใด ซึ่งวิธีที่ TCP และ UDP ใช้ คือ การใช้หมายเลขพอร์ต (port number)

เมื่อโพรเซสไคลเอนต์ (client process) ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ ไคลเอนต์จะต้องติดต่อแต่ลำพังรู้แอดเดรสอินเตอร์เน็ต 32 บิต เพียงอย่างเดียวนั้นไม่เพียงพอ เพราะว่าสามารถติดต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโพรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้ง TCP และ UDP ได้มีการกำหนดหมายเลขพอร์ตมาตรฐาน (well-known ports) ซึ่งเป็นที่รู้จักกัน เช่น ทุก ๆ ระบบ TCP / IP จะกำหนด port 23 เป็น port บริการของ Telnet เป็นต้น

เมื่อ TCP หรือ UDP กำหนดหมายเลขพอร์ตที่ไม่ซ้ำกันให้โพรเซสของผู้ใช้ เราเรียกหมายเลขพอร์ตนี้ว่าหมายเลขพอร์ตชั่วคราว (ephemeral port numbers) เมื่อไคลเอนต์เลิกใช้หมายเลขพอร์ตนี้แล้ว สามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอนต์อื่นได้ โพรเซสที่ได้รับหมายเลขพอร์ตชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าไร แต่เป็นหน้าที่ของอีกโพรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ ใน TCP และ UDP นั้น หมายเลขพอร์ตตั้งแต่ 1-1023 เป็นพอร์ตที่สงวนไว้สำหรับหมายเลขพอร์ตมาตรฐาน

2.3 เทลเนท (TELNET)

เนื่องจาก โครงงานนี้เป็นโครงงานที่เกี่ยวกับเทลเนท- โดยตรง ดังนั้น ในหัวข้อนี้จะพูดถึงเทลเนท, เทลเนท โปรโตคอล และการติดต่อของเทสเนท ซึ่งมีรายละเอียด ดังนี้

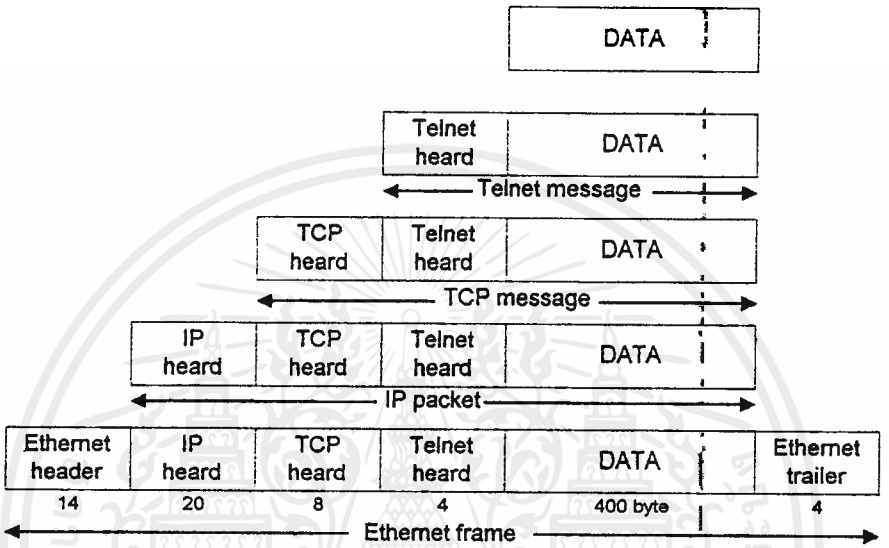
2.3.1 ภาพโดยรวมของเทลเนท

2.3.1.1 เทลเนท คืออะไร ?

เทลเนท คือ โปรแกรมประยุกต์ที่ทำให้ผู้ใช้สามารถ Login ผ่าน Terminal ใดๆ เพื่อขอใช้บริการจาก Host ใด ๆ ที่เป็น remote machine และใช้โปรโตคอล TCP / IP ในการติดต่อระหว่าง Terminal กับ Host นั้นๆ นั่นเอง

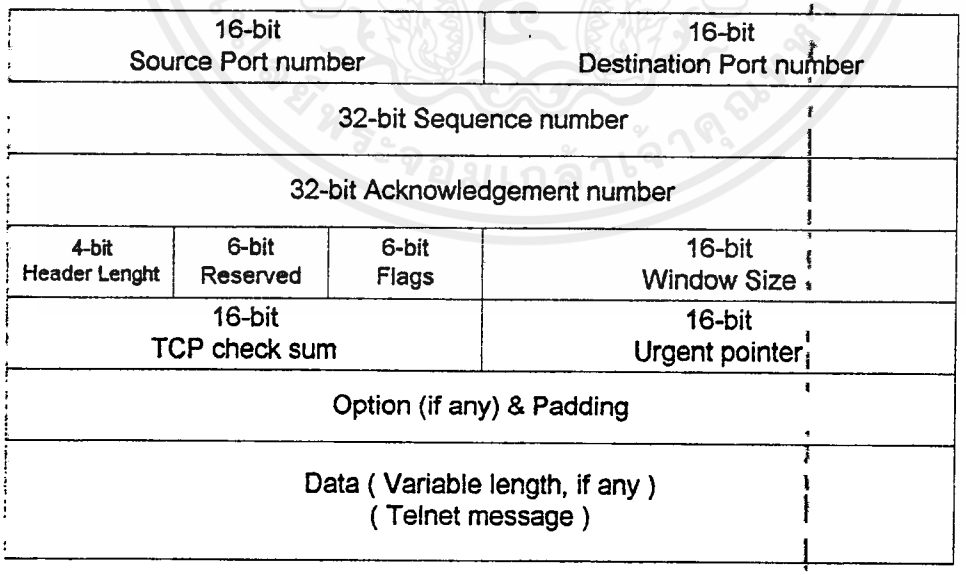
2.3.1.2 เทลเนททำงานอยู่ที่ Layer ใดใน TCP / IP โปรโตคอลและอยู่บน Port บริการที่เท่าไร ?

เทลเนทเป็นโปรแกรมประยุกต์ที่ทำงานอยู่บนชั้นแอพพลิเคชั่น ของโปรโตคอล TCP / IP ดังนั้นข้อมูลจากผู้ใช้ ก่อนที่จะส่งออกไปในระบบเครือข่ายจะต้องมีการเพิ่มส่วนหัวในชั้น ต่าง ๆ ดังรูป



รูปที่ 2.10

ส่วนรายละเอียดของ TCP message มีดังนี้



รูปที่ 2.11

ส่วนรายละเอียดของ IP Packet มีดังนี้

4-bit Version	4-bit Header Length	8-bit Type of Service	16-bit Total Length	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8-bit Time to Live		8-bit Protocol	16-bit Header Checksum	
32-bit Source Address				
32-bit Destination Address				
Option (if any) & Padding				
Data (variable length) (TCP message)				

รูปที่ 2.12

ส่วน port บริการของเทลเนทนั้น โดยทั่วไปแล้วจะอยู่ที่ Port บริการเบอร์ 23 ดังนั้น การทำ Connection จะต้องทำ Connection กับ Port 23 ที่ Host เสมอที่มีการใช้โปรแกรมเทลเนท

2.3.1.3 เทลเนทมีการทำงานเบื้องต้นอย่างไรบ้าง

หลังจากที่ผู้ใช้พิมพ์คำสั่ง "telnet < host id หรือ host name >" แล้วมีเหตุการณ์อะไรเกิดขึ้น ?

1. ที่เครื่องที่ผู้ใช้ (Client) จะทำการ เริ่มโปรแกรมเทลเนท

2. เทลเนทจากเครื่องผู้ใช้จะพยายาม Connect กับเครื่องให้บริการ (HOST) ถ้า Connect ได้เครื่องให้บริการจะมี ACK กลับ ถ้าไม่ได้ ก็จะแจ้งผิดพลาดแล้ว ABORT โปรแกรมไป

3. ถ้า Connect ได้แล้ว เครื่องผู้ใช้ก็จะขอทำRemote Login กับเครื่องให้บริการ โดยถ้าเครื่องให้บริการพร้อมให้ Login แล้วก็จะส่ง ACK ตอบกลับมา พร้อมทั้งส่ง Escape Cheracter ที่จะใช้ในการทำเทอร์มินอลเสมือน (Visual Terminal)

4. เครื่องผู้ใช้ ACK กลับแล้วพร้อมรอข้อมูลจากผู้ใช้ (Loginname, password) แล้วส่งข้อมูลเหล่านั้นไปให้เครื่องให้บริการ

5. เครื่องให้บริการตอบกลับว่า Login สำเร็จหรือไม่ โดยถ้าสำเร็จก็จะเข้า Shell process

2.3.2 Network Virtual Terminal (NVT)

เนื่องจากเทลเนทในทางปฏิบัติแล้ว เมื่อ User ทำการ Login ไปยัง Host ใดๆ ก็ตามเทลเนทจะทำการ Login ผ่าน NVT เพื่อให้ ผู้ใช้ ใช้ Terminal ของตน Login เข้าไปใช้ระบบและ device ของเครื่อง Host ได้

2.3.2.1 โครงสร้างและหลักการของ NVT

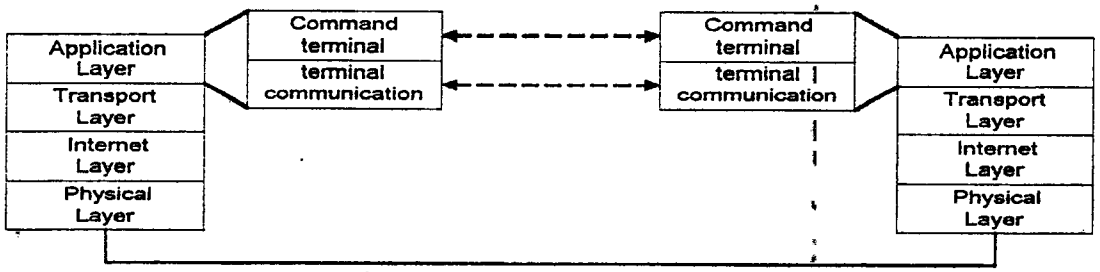
NVT เป็นหลักการที่ใช้ส่งข้อมูลระหว่าง terminal application ที่อยู่ต่าง ๆ เครื่องกันและต่างสิ่งแวดล้อมกัน ได้แก่ ความแตกต่างในเรื่อง OS ตลอดไปจนถึงความแตกต่างในเรื่องโปรโตคอลที่ใช้ในระดับ Upper - Level ของ application ที่ run อยู่ต่างเครื่องกันนั้น ซึ่งด้วยหลักการของ NVT นี้ จะทำให้ application ที่รันอยู่บนเครื่องใช้บริการและ application ที่รันอยู่บนเครื่องให้บริการซึ่งมีสิ่งแวดล้อมที่ต่างกัน สามารถติดต่อกันได้ซึ่งก็เปรียบเสมือนโปรแกรมเทลเนทที่รันอยู่บนเครื่องใช้บริการที่ต้องติดต่อกับ Telnet d ที่รันอยู่บนเครื่องให้บริการนั่นเอง

หลักการของการทำ NVT มีอยู่ว่า เราจะมอง Terminal ที่รัน application ทั้ง 2 ที่ต้องการติดต่อกันนั้นเป็น 2 Terminal คือ physical Terminal กับ logical Terminal

Physical Terminal คือ Terminal ที่รัน application นั้น จริง ๆ ซึ่ง Physical Terminal ที่ run application ทั้ง 2 อาจแตกต่างกันในหลาย ๆ เรื่องได้ดังที่ได้กล่าวมาแล้ว

logical Terminal คือ Terminal ในความคิดโดยเราจะกำหนดมาตรฐานต่าง ๆ ในการติดต่อให้เหมือนกันทั้ง 2 เครื่อง และเมื่อ logical Terminal บนเครื่องทั้ง 2 เหมือนกันแล้วจึงทำให้ logical Terminal ทั้ง 2 เครื่องคุยกันได้แล้ว logical Terminal ของแต่ละเครื่องก็จะ MAP Data ที่ติดต่อกันนั้น ให้อยู่ในรูปแบบที่ Physical Terminal ของเครื่องตนเองเข้าใจ ดังนั้น Application ที่รันอยู่บน Physical Terminal ที่ต่างกัน จึงสามารถติดต่อและแลกเปลี่ยนข้อมูลกันได้อย่างไม่มีปัญหาเรื่องสภาพแวดล้อมของแต่ละเครื่องเข้ามาเกี่ยวข้องเลย

และด้วยหลักการที่กล่าวทั้งหมดนี้ในทางปฏิบัติแล้วเราจะแบ่ง Application Layer ออกเป็น 2 Sublayer ดังรูป



รูปที่ 2.13 แสดงโครงสร้างการทำ NVT

Command Terminal sublayer เป็น sublayer ที่รัน application นั้น ๆ อยู่จริง โดยที่ sublayer นี้ จะมีสิ่งแวดล้อมต่าง ๆ ตามเครื่องที่รันอยู่ ซึ่ง Sub Layer นี้ ก็เปรียบเสมือน Physical Terminal นั่นเอง

Terminal Communication เป็น subLayer ที่ แปลงสิ่งแวดล้อมที่ไม่เหมือนกันทั้ง 2 เครื่องให้เป็นมาตรฐานเดียวกัน เพื่อให้ application ทั้ง 2 สามารถคุยกันได้และแปลงข้อมูลที่ติดต่อกันให้อยู่ในรูปแบบที่ physical Terminal ของตนเองเข้าใจ ซึ่ง subLayer นี้ ก็เปรียบเสมือน Logical Terminal นั่นเอง

2.3.2.2 การทำงานและ option ของ NVT ในเทลเนท

การทำงานจะเริ่มจากเมื่อผู้ใช้รันเทลเนท ที่ Terminal ของตนเองที่สามารถติดต่อเข้าไปใน NVT ของเครื่อง Host ที่ต้องการจะติดต่อด้วยได้อย่างที่ได้กล่าวมาแล้ว NVT ที่ติดต่อด้วยนั้น จะทำให้ user เหมือนกันได้ใช้ Terminal บนเครื่อง Host นั้นจริง ๆ แต่ Device ที่ผู้ใช้เห็นบนเครื่อง Host นั้น จริง ๆ แล้วเป็นเพียงการจำลองมาเท่านั้น

เมื่อให้เห็นภาพได้ชัดเจนยิ่งขึ้น คุณลองคิดถึง Terminal Device จริง ๆ ดูว่าอะไรเป็น Device ที่เป็นอินพุท (input) (เช่น keyboard) และ อะไรเป็น Device ที่เป็นเอาท์พุท(output) (เช่น จอภาพ, เครื่องพิมพ์) ดังนั้น NVT ก็จะจำลอง Terminal Device จริง ๆ เหล่านั้น ซึ่งก็มี keyboard เป็น input และ printer เป็น output printerจะพิมพ์ข้อความที่ได้รับมาจาก Host บางข้อความที่ Host ต้องแสดงและ keyboard ก็จะสร้างข้อความที่จะต้องส่งออกไปยัง Host เสมือนกับว่า Terminal ที่ใช้อยู่นั้น เป็น Terminal จริง ๆ บนเครื่อง Host นั่นเอง

และเนื่องจากในแต่ละ Host มักจะมี User จำนวนมาก ดังนั้น การจะทำให้ NVT ตอบสนองกับสิ่งแวดล้อมของ Terminal ของแต่ละ User จึงทำได้ยาก ซึ่งวิธีแก้ก็คือ ใช้วิธีการตกลง Option ต่าง ๆ ที่แต่ละ Terminal แต่ละ User ต้องการโดยจะทำการตกลงกันในตอนที่ทำ connection กัน option ที่ NVT เตรียมไว้มีตัวอย่างดังนี้

- Option เปลี่ยนลักษณะตัวอักษรของ NVT
- Option ที่เลือกว่าจะให้ NVT สะท้อน ตัวอักษรกลับมาให้ผู้ใช้เห็นบนจอภาพหรือไม่
- Option เลือก mode ว่าจะส่ง Data เป็นบรรทัดทีเดียว หรือจะเลือก Mode ส่ง Data ทีละตัวอักษร

และยังมี Option อื่น ๆ อีก การขอทำ Option ต่างๆ เหล่านี้ สามารถเลือกว่าจะใส เพิ่มหรือถอนออกก็ได้ ซึ่งการตกลงเจรจาที่จะทำ Option นั้น ทำได้โดยใช้ลำดับของข้อความ WILL, WON'T DO และ DON'T นั่นเอง

"WILL X" เป็นข้อความที่ส่งจากไซด์ใดไซด์หนึ่งโดยไซด์ที่ส่งนั้นได้บอกไซด์อื่นที่ติดต่อด้วยว่า ไซด์ตนเองจะทำ option x ซึ่งถ้าไซด์อื่นยอมรับการทำ option x นั้นได้ ก็จะตอบกลับมาด้วย "DO X" แต่ถ้าไซด์อีกไซด์ไม่ยอมรับการขอทำ Option X นั้น ก็จะตอบกลับมาว่า "Don't X"

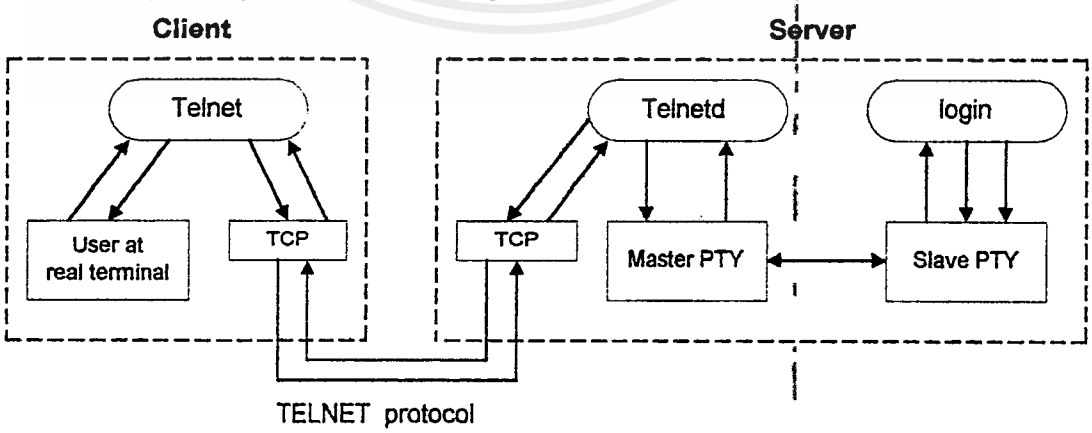
"Do X" เป็นข้อความที่ส่งโดยไซด์ใดไซด์หนึ่ง โดยที่มันต้องการให้ไซด์อีกไซด์ที่มันติดต่อด้วย ทำ Option X ให้มัน โดยถ้าไซด์อีกไซด์นั้น ตอบรับการทำ Option X นั้นก็จะตอบกลับมาด้วย "WILL X" แต่ถ้าไซด์อีกไซด์นั้นไม่ยอมทำ Option X นั้น ไซด์นั้น ๆ ก็จะตอบกลับไปว่า "Won't X" นั่นเอง ซึ่งตัวอย่างของ Telnet option ได้แสดงเอาไว้ในตารางแล้ว

2.3.3 Telnet Model

ในหัวข้อนี้เราจะพูดถึงโครงสร้างและ Model ของเทลเน็ทว่ามีส่วนประกอบอะไรบ้างและแต่ละส่วนทำหน้าที่อะไรบ้าง

2.3.3.1 โครงสร้างของ Telnet Model และการทำงาน

บนระบบปฏิบัติการยูนิกซ์ได้มีการออกแบบโครงสร้างของโปรแกรมเทลเนท โดยประกอบด้วย Telnet command ซึ่งเป็นโปรแกรมที่ทำงานอยู่บนเครื่องใช้บริการและ Telnetd ซึ่งเป็น process ที่ VUM อยู่บนเครื่องให้บริการซึ่งมีระบบปฏิบัติการเป็นระบบยูนิกซ์ ซึ่ง Telnet ของระบบปฏิบัติการยูนิกซ์ มีโครงสร้างดังรูปที่ 2.14



รูปที่ 2.14 Telnet Mode

จากรูปการทำงานจะเริ่มขึ้นเมื่อผู้ใช้ใส่คำสั่ง open แล้วตามด้วยชื่อ Host name ที่ผู้ใช้ต้องการจะติดต่อด้วยให้กับโปรแกรม Telnet Command บนเครื่องใช้บริการ โปรแกรม Telnet Command ก็จะสร้าง Commection ด้วยโปรโตคอล TCP/IP ติดต่อกับเครื่องให้บริการที่ผู้ใช้ต้องการติดต่อด้วย หลังจากที่สร้าง Connection กับเครื่องให้บริการสำเร็จแล้ว โปรแกรม Telnet Command และ Telnet Command และ Telnetd Process ก็จะทำการตกลง Option ที่จะกระทำต่อกัน เช่น ชนิดของ Terminal บนเครื่องใช้บริการและเครื่องให้บริการ ,Telnet Mode ว่าเป็นส่งที่ละบรรทัดหรือส่งข้อมูลที่จะตัวอักษร เป็นต้น และเมื่อ Telnet Command ตกลง Option กับ Telnet Process เสร็จเรียบร้อยแล้ว Telnetd Process บนเครื่องเครื่องให้บริการ ก็จะสร้าง Pseudo-Terminal Device (PTD) ซึ่งเป็น device driver process ขึ้นมาส่อง PTD คือ PTD Master และ PTD Slave

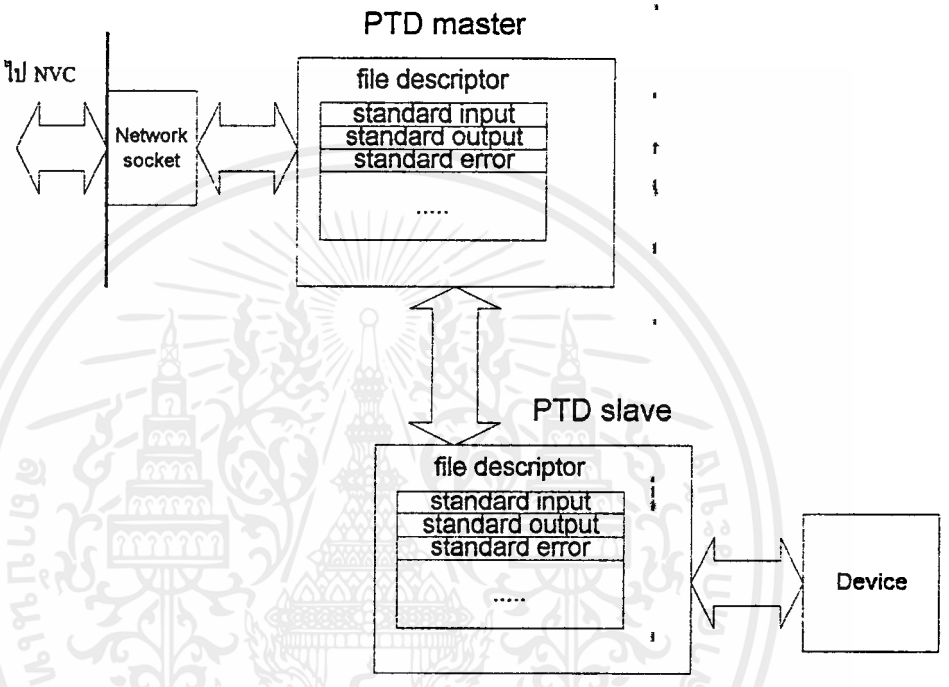
PTD Slave ก็คือ device ของ Terminal จริง ๆ บนเครื่องที่ Telnet process รันอยู่ หรือก็คือ Physical Terminal ใน NVT นั่นเอง

ส่วน PTD Master ก็คือ Process ที่แทน device บนเครื่องใช้บริการที่ผู้ใช้ใช้อยู่นั่นเอง หรือก็คือ Logical Terminal ใน NVT นั่นเอง

ซึ่งถ้าเรามาลองพิจารณา Telnet Model ดูแล้วเราก็จะพบว่า Telnet Model นั้นได้ออกแบบไว้ตามหลักการของ NVT ทุกประการ และจากหลักการดังกล่าวเวลาที่ Telnetd process จะเขียนข้อมูลลง Device แทนที่ Telnet process จะเขียนลง Device จริง ๆ ของ Terminal ที่ Telnet process รันอยู่ก็เขียนลง PTV Master แทน Device ที่ใช้หลักการนี้ ก็ได้แก่ Device จอภาพ คือแทนที่จะเขียนจอภาพของเครื่องให้บริการ เราก็เขียนไปที่จอภาพของ Terminal ของผู้ใช้แทนและในทำนองเดียวกันในเวลาที่ผู้ใช้ต้องการเขียนข้อมูลบางข้อมูลลง PTD Master PTD Master ก็จะนำข้อมูลนั้นเขียนลง PTD Slave อีกทีหนึ่ง ตัวอย่างก็เช่น Device Keyboard เมื่อผู้ใช้กด Keyboard Telnet Command ก็จะส่ง Data เขียนลงบน PTD Master แล้ว PTD Master ก็จะเขียนข้อมูลนั้นลง PTD Slave ซึ่งก็คือ Keyboard Buffer ของเครื่อง Server นั่นเองซึ่งก็เสมือนกับผู้ใช้กำลังกด Keyboard อยู่บนเครื่องให้บริการนั้น ๆ จริง ๆ

เรากลับมาดูการทำงานของ Telnet d กันต่อ Telnet d จะสร้าง PTD มาเพียง Process เดียวซึ่ง Process PTD ตัวแรกที่สร้างนี้ก็คือ PTD Master นั่นเอง ซึ่ง PTD process นี้จะ Fork process ลูกอีก Process นั่นก็คือ PTD Slave ซึ่ง code ของ process PTD ทั้ง2 จะเป็นเดียวกันแต่มีการทำงานที่ต่างกันโดยดูจาก ID Process ว่า process ไหนเป็น process พ่อ (PTD MASTER) หรือ process ลูก (PTD Slave) โดย Process พ่อ (PTD master) จะ set file descriptors ของ Standard input standard out และ standard error ไปที่ซอคเก็ตที่ถูกสร้างขึ้นตอน

ทำ connection ในตอนเริ่มต้นนั่นเอง ส่วน process ลูก (PTD Slave) ก็จะ set file descriptors ของ standard input standard output และ standard error ไปที่ device ต่างบนเครื่องให้บริการดังรูป



รูปที่ 2.15

process ลูก (PTD Slave) หลังจาก set file descriptors ต่าง ๆ เสร็จก็จะทำการ exces login command แล้วส่ง data จาก login command ไปให้ process พอ (PTD Master) อ่านเพื่อจะได้เขียนลง standard output ของ PTD Master ซึ่งเป็นซอคเก็ตที่ต่อกับเครื่องใช้บริการอีกทีหนึ่งที่ connect อยู่บนเครือข่าย แล้วที่เครื่องใช้บริการของผู้ใช้ก็จะแสดง login prompt (login:) เมื่อให้ผู้ใช้ใส่ข้อมูล และเมื่อผู้ใช้ใส่ข้อมูล Telnet command ก็จะส่งข้อมูล ที่ผู้ใช้ใส่มานั้นไปให้ Telnetd process แล้ว PTD Master ก็จะอ่านข้อมูลนี้ แล้วจะส่งไปให้ PTD Slave เมื่อเขียนลง device เมื่อให้ login command นำไปใช้ต่อไปซึ่งขั้นตอนการ execs login command นี้จะเป็นขั้นตอนแรกก่อนที่ Telnetd จะ start shell เพื่อให้ผู้ใช้เข้าใช้ระบบของเครื่องให้บริการต่อไป

เรากลับมาดูที่เครื่องใช้บริการกันบ้าง Telnet command เมื่อทำ connection เสร็จแล้ว ก็จะให้ผู้ใช้ ใส่ข้อมูลของคุณหรือที่คุณจะใส่ไปยังเครื่องให้บริการ และแสดงผลข้อมูลจากเครื่องให้บริการ ที่ส่งกลับมายัง Terminal ของคุณด้วย และด้วยรูปแบบการทำงานแบบนี้จึงทำให้ผู้ใช้สามารถอ่านข้อมูลจาก shell บนเครื่องให้บริการ และผู้ใช้สามารถส่งข้อมูลไปเขียนลง shell แสดงผลการเขียนข้อมูลนั้นบนระบบของผู้ใช้เองได้

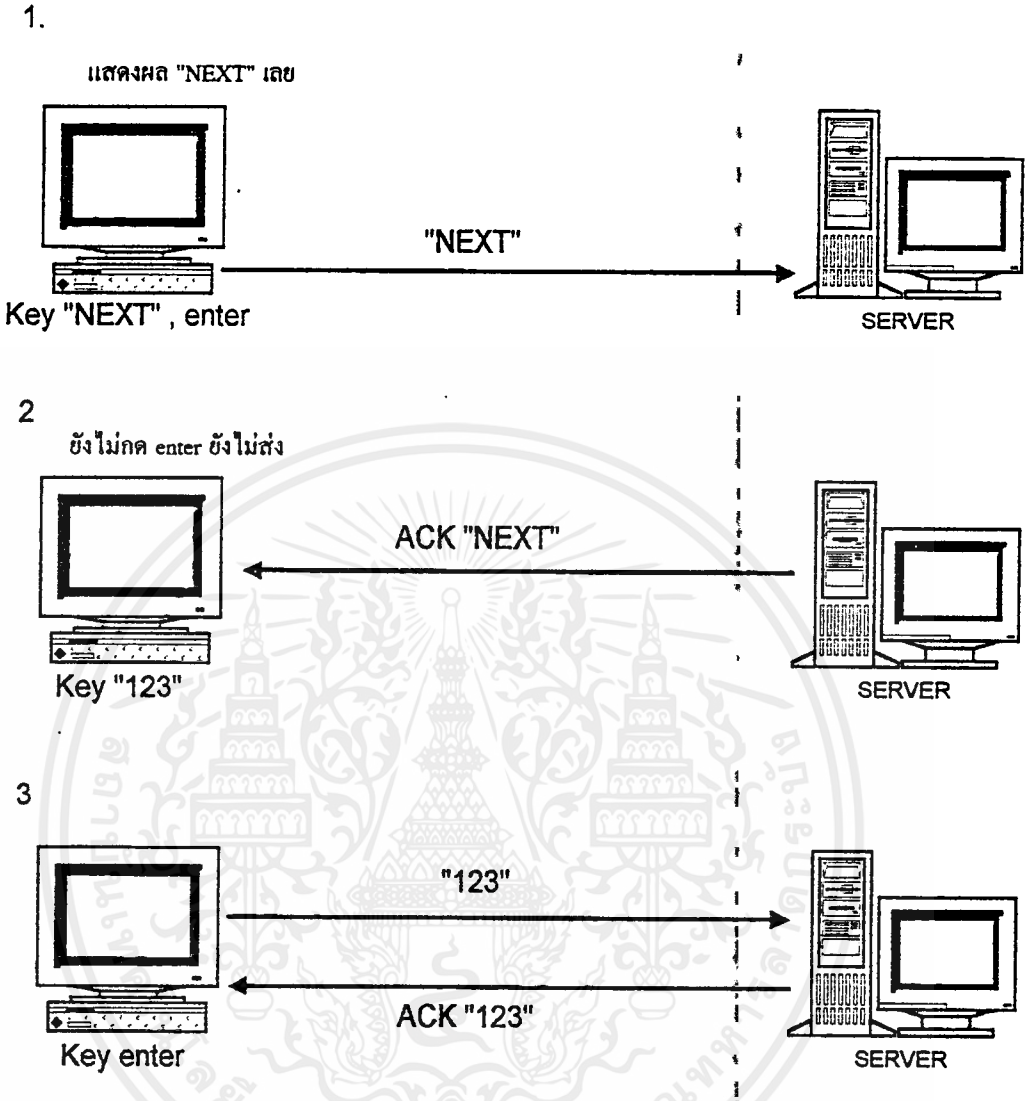
2.3.3.2 Mode การส่งข้อมูลของ Telnet

หลังจากที่ทำ Connection ระหว่าง Telnet Commnad บนเครื่องใช้บริการกับ Telnetd process บนเครื่องให้บริการแล้วง ก็จะมีการตกลง Option กันแล้วก็จะมีการส่งข้อมูลกัน ซึ่ง Mode การส่งข้อมูลจาก Telnet command ไปยัง Telnetd process จะมีอยู่ 2 Mode คือ Mode line-at-a-time และ Mode character-at-a-time

2.3.3.2.1 Mode การส่งข้อมูลแบบ line-at-a-time

การส่งข้อมูลใน Mode นี้ จะส่งข้อมูลที่ละบรรทัด คือ ส่งเมื่อผู้ใช้มีการกด Key "New line" และเมื่อส่งข้อมูลจากเครื่องใช้บริการไปให้เครื่องให้บริการแล้ว เครื่องให้บริการก็จะตอบรับ (ACK) ข้อมูลที่ส่งมานั้นเท่านั้น จะไม่มีการส่ง Echo ข้อความเพื่อแสดงผลบนเครื่องใช้บริการแต่อย่างใด เพราะการแสดงผลบนเครื่องใช้บริการจะถูกควบคุมด้วยตัวของเครื่องใช้บริการเอง เนื่องจากไม่สามารถรอการส่ง Echo ข้อความกลับมาจากเครื่องให้บริการไม่ได้ เพราะเครื่องใช้บริการเองต้องรอการกด Key "New line" จึงจะส่งข้อมูล ดังนั้นถ้ารอ Echo จากเครื่องให้บริการแล้วผู้ใช้จะไม่เห็นตัวอักษรที่ตนพิมพ์เข้าไปจนกว่าผู้ใช้จะกด Key "New line" ซึ่งถ้าเป็นอย่างนี้แล้วจะเป็นการไม่สะดวกที่ผู้ใช้จะแก้ข้อความที่พิมพ์ผิด เพราะผู้ใช้ยังไม่เห็นข้อความที่พิมพ์เข้าไป

ดังนั้นเพื่อไม่ให้เกิดเหตุการณ์ดังกล่าว การส่งข้อมูลใน Mode นี้ จึงให้เครื่องใช้บริการเป็นเครื่องที่ควบคุมการแสดงผลเอง ดังนั้นการส่งข้อมูลใน Mode นี้จึงไม่มีปัญหาในการใช้ Key พิเศษ เช่น Key Backspace หรือ Key Delete เพราะอย่างที่ได้กล่าวไปแล้ว การจัดการแสดงผลจะทำดดย Terminal ของผู้ใช้ และผู้ใช้ยังเป็นผู้กำหนดว่าจะส่งข้อมูลไปให้เครื่องให้บริการเมื่อไรเองอีกด้วย ซึ่งการส่งข้อมูลใน Mode นี้ได้แสดงไว้ดังรูป



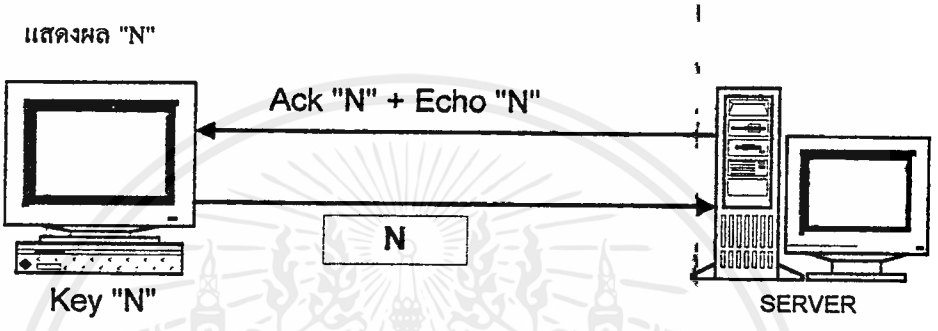
รูปที่ 2.16

แต่การส่งข้อมูลเซฟ์ line-at-a-time ก็มีปัญหาคือ การส่งข้อมูลใน Mode นี้ เมื่อส่งข้อมูลไปแล้วเราจะไม่สามารถกลับไปแก้ไขข้อมูลเดิมได้อีก จึงทำให้ไม่สามารถใช้โปรแกรมประเภท Word หรือ Editor ได้ใน Telnet Mode นี้ เพราะการส่งข้อมูลแบบนี้ไม่สามารถแก้ไขข้อมูลที่ส่งไปแล้วได้นั่นเอง

2.3.3.2.2 Mode การส่งข้อมูลแบบ character-at-a-time

การส่งข้อมูลใน Mode นี้ จะส่งข้อมูลที่ละตัวอักษร คือจะส่งกันที่ที่ผู้ใช้มีการกดคีย์ใดๆ และจะส่งข้อมูลที่ละตัวอักษร ซึ่งการส่งข้อมูลใน Mode นี้ Telnet command ที่

รันอยู่บนเครื่องใช้บริการจะยังไม่แสดงผลข้อมูลที่ส่งไปยังเครื่องให้บริการบนจอภาพทันทีแต่จะรอให้เครื่องให้บริการตอบรับและส่ง Echo กลับมาให้จึงจะแสดงผลบนจอภาพ การส่งข้อมูลใน Mode นี้ แสดงไว้ดังรูป



รูปที่ 2.17

ข้อดีของการส่งข้อมูลใน Mode นี้ ก็คือ ผู้ใช้สามารถแก้ไขข้อมูลที่ส่งไปแล้วได้ง่ายกว่าการส่งข้อมูลใน Mode line-at-a-time เพราะข้อมูลที่ส่งไปแล้วนั้น มีขนาดเพียง 1 ตัวอักษร ดังนั้นเราจึงส่งข้อมูลไปแก้ไขข้อมูลที่ส่งไปแล้วเหล่านั้นได้ แล้วด้วยเหตุนี้จึงทำให้การส่งข้อมูลใน Mode นี้สามารถใช้งานโปรแกรมจำพวก Word หรือ Editor ได้ จึงทำให้การส่งข้อมูลใน Mode นี้ เป็นที่นิยมใช้กันในปัจจุบัน

แต่การส่งข้อมูลใน Mode character-at-a-time นี้ก็มีปัญหาคือ การแสดงผลบนจอภาพของการส่งข้อมูลใน Mode นี้ นั้นจะต้องแสดงจาก Echo character ที่ส่งกลับมาจาก Host ดังนั้นการแสดงผลบางอย่างจึงไม่สามาถทำได้ เช่น เมื่อผู้ใช้กด Key backspace แล้ว Host ส่ง ASCII ของ backspace กลับมาให้เครื่องใช้บริการของผู้ใช้ แล้วเครื่องใช้บริการของผู้ใช้ จะแสดงผลของ Key backspace ออกมาเป็น character ของ ASCII ของ backspace ซึ่งไม่ได้เป็นไปตามที่ผู้ใช้ ต้องการซึ่งสิ่งที่ผู้ใช้ ต้องการคือ ให้ Move Cursor กลับไปหนึ่งตัวอักษรและลบ Character ตัวนั้นด้วย

ดังนั้น จึงต้องมีการทำ Vissual Terminal ที่จะกล่าวถึงในหัวข้อต่อไป

2.4 เทอร์มินอบเสมือน (Visual Terminal)

จากหัวข้อ 2.3.3.2.2 ได้พูดถึงการทำเทอร์มินอลเสมือน ไว้ถึงเหตุผลในการทำเทอร์มินอลเสมือน ก็เพราะการส่งข้อมูลใน Mode Character -at-a-time การแสดงผลบนจอภาพของเครื่องใช้บริการของผู้ใช้จะถูกควบคุมโดยเครื่องให้บริการ แต่เนื่องจาก เครื่องใช้บริการและเครื่องให้บริการอยู่ห่างไกลกัน การติดต่อก็ทำได้โดยผ่าน Net work ที่เชื่อมโยงเครื่องทั้งสอง ซึ่งการส่งข้อมูลผ่าน Net work นี้เอง ทำให้เครื่องให้บริการไม่สามารถควบคุมการแสดงผลของเครื่องใช้บริการด้วย Echo character เพียงอย่างเดียวได้ ดังนั่นจึงต้องมีการทำเทอร์มินอลเสมือน เช่น เมื่อผู้ใช้กด Key backspace แทนที่เครื่องให้บริการจะส่ง Echo character backspace กลับมาก็ส่งคำสั่งให้เครื่องใช้บริการทำการย้ายตำแหน่ง Curser ถอยหลังกลับ 1 ตัวอักษรแทน เป็นต้น ซึ่งคำสั่งที่เครื่องให้บริการใช้ทำเทอร์มินอลเสมือนเกล่านี้นั้น จะมีลักษณะเป็น Code คำสั่ง ต่าง ๆ โดยจะเริ่มต้นด้วย “escape” character ซึ่งจะตกลงกันเอาไว้ระหว่างเครื่องให้บริการและเครื่องใช้บริการตั้งแต่ตอนที่ติดต่อกันตอนแรก (ตอนที่ตกลง Option ต่าง ๆ นั่นเอง) ซึ่ง Code ที่ใช้ในการทำเทอร์มินอลเสมือนเหล่านี้ ได้มีการกำหนดไว้เป็นมาตรฐานต่าง ๆ ไว้มากมายหลายมาตรฐาน แต่ที่นิยมใช้กันมากที่สุดได้แก่ มาตราฐานตระกูล VT ซึ่งในภาคผนวกได้มีตัวอย่าง Code ของ VT 100 ไว้ให้แล้ว

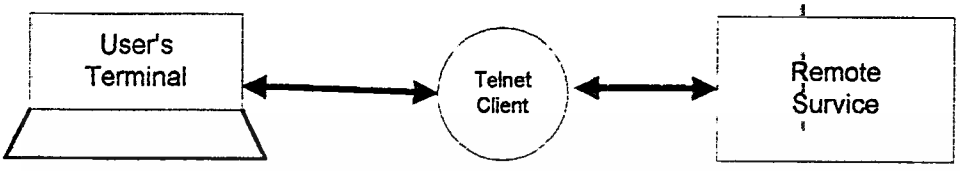
2.5 คำสั่งและการควบคุมของเทลเนท (Command And Control Information)

ในการรวมเอาข้อมูลที่เป็นอักขระเทลเนทจะยินยอมให้เครื่องใช้บริการ(Client) และ เครื่องให้บริการ(Server) ทำการแลกเปลี่ยน คำสั่งหรือ ข้อมูลการควบคุม เพราะในระหว่างการติดต่อสื่อสารกันเครื่องใช้บริการและเครื่องให้บริการ ผ่านการเชื่อมต่อ TCP, Protocol จะจัดการแปลคำสั่งหรือข้อมูลการควบคุมเป็นรหัส ดังนั้นทางฝ่ายผู้รับจะต้องรู้จักการแปลรหัสเหล่านั้น เป็นเหตุให้ต้องทราบจุดมุ่งหมายของโปรโตคอลจึงมีการกำหนดการส่งรหัสของคำสั่งและการรู้จักการรับข้อมูลอย่างไร

เทลเนทถูกกำหนดอย่างเด่นชัดระหว่างการติดต่อสื่อสารของผู้ใช้ซึ่งเป็น Terminal และการใช้บริการจากระยะไกล โปรโตคอลถูกกำหนดความสามารถพิเศษของTerminal ในส่วนของ Keyborad ซึ่งผู้ใช้สามารถส่งอักษรและแสดงผลบนหน้าจอในโหมดข้อความ

ในทางปฏิบัติแล้ว ผู้ใช้สามารถตัดสินใจที่จะร้องขอความต้องการจากเครื่องใช้บริการ กับ ทุก ๆ อินพุท ในแต่ละสถานที่Keyboard และเอาท์พุท ซึ่งอาจจะเป็นแสดงผล นอกจากนั้นผู้ใช้ยังเรียกร้องจากเครื่องใช้บริการ ซึ่งอยู่ในรูปของวินโดวส์

2.6 เทลเนทไคลเอนท์อัลกอริทึม (Telnet Client Algorithm)



รูปที่ 2.18

รูป แนวความคิดของเทลเนทไคลเอนท์(Telnet Client) ผู้ขอบริการต้องการส่งตัวอักขระจากผู้ใช้โดยKeyboardไปยังผู้ให้บริการผ่านทางไกลและส่งข้อมูลจากผู้ให้บริการทางไกลไปยังผู้ขอใช้บริการ

2.7 อัลกอริทึมของเทลเนทไคลเอนท์ (Algorithm of Telnet Client)

1. วิเคราะห์คำของข้อมูลที่ส่งมาและเริ่มต้น ส่งโครงสร้างข้อมูล
2. เปิดการConnectionโดย TCP ไปยังพอร์ท(port) ของผู้ให้บริการระยะไกล
3. รอการรับอินพุทจาก ผู้ใช้(User) หรือข้อมูลจะอยู่สูงถึง Connection TCP
4. ถ้าข้อมูลที่ถูกส่งจากKeyboard และอ่านค่านำส่งไปผ่านกระบวนการเปลี่ยนรูปแบบไปยัง Network Virtual Terminal (NVT)
5. กลับไปทำข้อที่ 3

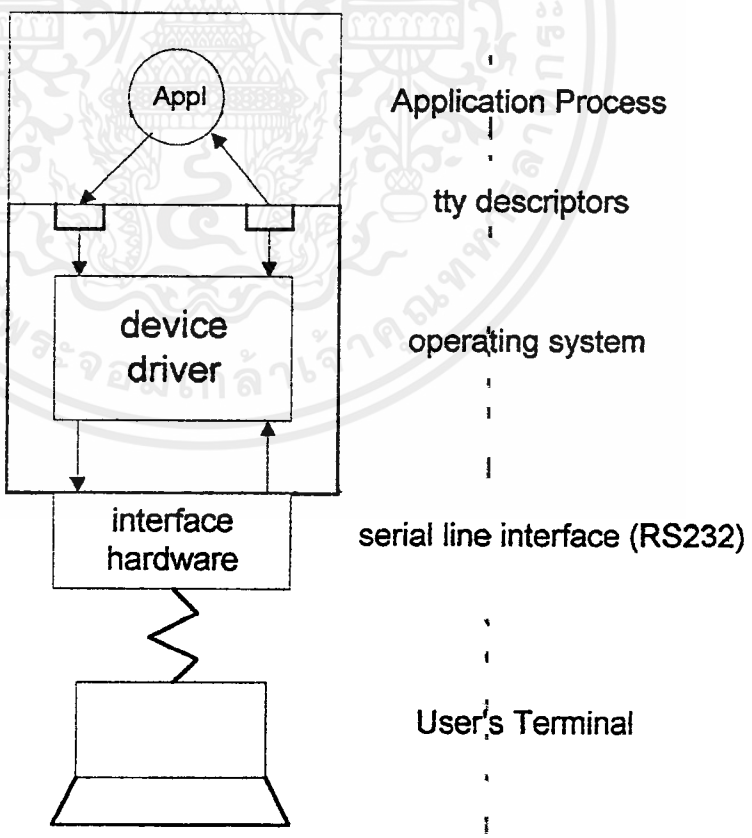
2.8 เทอร์มินอล I/O ในยูนิกซ์ (Terminal I/O In UNIX)

แนวคิดจากข้อ 2.5.4 เป็นตัวอย่างให้เห็นภาพมากทีเดียว แม้กระทั่งเนื้อหาของโปรโตคอลเทลเนท(Telnet Protocol) และ เทอร์มินอลI/Oที่ยุ่งยากของรหัสไปจนถึงเข้าใจซอฟแวร์ Telnet Client เป็นอย่างไร สลับกับจุดที่ผู้ใช้คือ Termianl ขั้นแรกต้องเข้าใจในเนื้อหาระบบปฏิบัติที่จัดการกับเทอร์มินอลI/O ขณะนี้จะอธิบายแต่ละส่วนของการจัดการเทอร์มินอลI/Oของ UNIX และใน UNIX I/Oไปยังอุปกรณ์ Terminal ตามเปิด-อ่าน-เขียน-ปิด (open-read-write-close) ซึ่งเป็นไดอะแกรมที่ใช้สำหรับI/Oไปยังระบบไฟล์หรือSockets โปรแกรมประยุกต์จะเรียกเปิด(open)ที่ได้จาก I/O สำหรับ Keyboard หรือแสดงผล อ่าน (read) รับข้อมูลจากผู้ใช้โดย Keyboard และเขียน (write) ส่งข้อมูลไปแสดงผลให้แก่ผู้ใช้

ในทางปฏิบัติแล้วการประยุกต์ไม่ได้เรียกเปิดไปสร้างสำหรับผู้ใช้บริการ เพราะว่าตัวแปลคำสั่งได้เตรียมการมีการเปิดโดยอัตโนมัติ

แม้ว่าเทอร์มินอลI/Oมีลักษณะเช่นเดียวกับพื้นฐานของไฟล์I/O(file I/O) และสิ่งมากมายของเนื้อหาจะมีความยุ่งยาก บางอย่างของเนื้อหาเกิดขึ้น เนื่องจากได้เริ่มต้นฮาร์ดแวร์ (hardware)ก่อน สำหรับในตัวอย่าง แม้ว่าผู้ใช้บริการคิดถึงในส่วนของkeyboardและการแสดงผล h ฮาร์ดแวร์จะแยกให้ keyboard เป็นอินพุทการแสดงผลเป็นเอาท์พุทและแยกออกเป็น 2 อุปกรณ์ ดังนั้น ฮาร์ดแวร์จะไม่ทำอัตโนมัติในการแสดงผลของแต่ละชนิด ของผู้ใช้บริการ

การประยุกต์ผู้ใช้ต้องการเห็นการเปลี่ยนแปลงอักษรในการแสดงผลทุกประเภท แต่การประยุกต์บางอย่างก็ต้องการที่จะไม่แสดงผลของอินพุทจากkeyboard เช่นในการใส่รหัสผ่านเพื่อจะต้องรักษารหัสผ่าน จากการเฝ้ามองของผู้ที่เฝ้ามองอยู่ เป็นต้น ดังนั้นในการใช้งานทุกอย่างมีรายการข้อปลีกย่อยที่มีความเกี่ยวข้องกับเทอร์มินอลI/O และการแสดงอักษร ในระบบUNIXได้เตรียมโปรแกรมเหล่านี้ไว้อัตโนมัติแล้ว เรียกว่า เทอร์มินอลดีไวซ์ไดร์เวอร์(terminal device driver) จะอยู่รวมในระบบปฏิบัติการ ดังรูป



รูปที่ 2.19

ดีไวซ์ไดร์เวอร์(device driver) มีความเกี่ยวข้องกับอินพุทที่เป็นkeyboardและตรงกับการแสดงผลของเทอร์มินอลดีไวซ์ไดร์เวอร์สามารถโต้ตอบกันได้ หรือหยุดการโต้ตอบได้ และดีไวซ์ไดรเวอร์สามารถรองรับสัญญลักษณ์พิเศษ ซึ่งใช้ในการขัดจังหวะหรือหยุดการทำงานในขณะนั้น

บทที่ 3
การเข้ารหัส

3.1 ทฤษฎีการเข้ารหัสแบบ Caesar Cipher

The Caesar Cipher เป็นของ Julius caesar หลักการ ถึงอักษรจะถูกเปลี่ยนจากตัวหน้าเป็นอีกตัวหนึ่งโดยการให้อักษรแต่ละตัวมีเลขคงที่ตัวหนึ่ง เช่น A = 0 แล้วบวกเพิ่มอีก 3 และแปลงกลับเป็นอีกตัว

$$A \longrightarrow D \qquad (A + 3 = D)$$

ดังนั้น Plaintext pi จะถูกเข้ารหัสเป็น ciphertext ci โดยกฎ

$$ei = E(pi) = pi + 3$$

เช่น **plaintext letter : ABCDEFGHIJKLMNOPQRSTUVWXYZ**

ciphertext letter : DEFGHIJKLMNOPQRSTUVWXYZABC

ข้อดีของCaesar Cipher ง่ายในการทำ คือ ง่ายต่อการเข้ารหัสและถอดรหัสเป็นสูตรที่จำง่าย

ข้อเสียของ Caesar Cipher เสี่ยงต่อการที่คนอื่นจะถอดรหัสได้ง่าย เนื่องจากไม่ซับซ้อน ใช้เวลาเพียงไม่นานก็ถอดรหัสได้ โดยเดาว่าอัลกอริทึมเป็นอย่างไร หรือลองแทนอักษรที่เป็นไปได้

3.2 ทฤษฎีการเข้ารหัสแบบ Vemam Cipher

Vemam Cipher นี้เป็นการเข้ารหัสที่สามารถหลีกเลี่ยงการหยุดของ cryptanalytic โดยส่วนมากได้

โดยใช้หลักการที่ว่า จะมีเลขจำนวนหนึ่ง ซึ่งไม่ซ้ำกันนำมาบวกเข้ากับ plaintext การคิดค้นของ Vernan นี้ได้มาจาก เทปกระดาษยาวๆ ที่ป้อนให้กับเครื่องรับโทรเลข ซึ่งเทปนี้จะมีตัวเลขที่สุ่มซึ่งจะรวมกับตัวอักษรที่ถูกพิมพ์เข้าไปในเครื่องลำดับของตัวเลขที่นี้จะไม่ซ้ำกันเลย และเทปแต่ละอันถูกใช้เพียงครั้งเดียวเท่านั้น ที่ห้องคีย์เทปจะไม่ซ้ำและไม่ถูกนำกลับมาใช้ใหม่

เช่น

plaintext	V	E	R	N	A	M	C	I	P	H	E	R
numeric equivalent	21	4	17	13	0	12	2	8	15	7	4	17
+random number	76	48	16	82	44	3	58	11	60	5	48	88
=sum	97	52	33	95	44	15	60	19	75	12	52	105
≡ mod 26	19	0	7	17	18	15	8	19	23	12	0	1
ciphertext	T	A	H	R	S	P	I	T	X	M	A	B

จะเห็นว่า วิธีนี้จะทำให้รอดพ้นจากการหยุดของ cryptanalytic ได้เพราะciphertext ที่เห็นไม่ได้ถูกแสดงในรูปของคีย์ (คือ ไม่มีสูตรเป็นเลขตายตัว)

3.3 ทฤษฎีการเข้ารหัสแบบ Columnar Transpositions

วิธีนี้เป็นวิธีง่ายๆ โดยจะนำมา plaintext มาใส่ในหลัก(columns) ตัวอย่างเช่น กำหนดให้มี 5 หลัก ก็จะนำอักษรแต่ละตัวของ plaintext มาจัดเรียงใหม่เป็นบลอค(block) โดยใส่เรียงกันไปทีละบลอคเมื่อครบ 5 บลอคก็ขึ้นแถวที่ 2 ในบลอคที่ 1 วนไปเช่นนี้จนจบ

ดังนี้ Ci คือ อักษรใน plaintext

C1	C2	C3	C4	C5
C6	C7	C8	C9	C10
C11	C12	C13	C14	C15

เป็นต้น

จะได้ผลลัพธ์ของวิธีนี้ = C1 C6 C11 ... C2 C7 C12 ... C3 C8 เป็นต้น

ตัวอย่าง คำว่า "This is a message to show how a columnar transposition works"

ใช้วิธีนี้ในการเข้ารหัสจะได้

T H I S I
S A M E S
S A G E T
O S H O W
H O W A C
O L U M N
A R T R A
N S P O S
I T I O N
W O R K S

Ciphertext ที่ได้เมื่อทำการอ่านออกมาคือ

TSSOH OANIW HAASO LRSTO IMGHW
UTPIR SEEOA MROOK ISTWC NASNS

จะเป็นว่าความยาวของข้อความในตัวอย่างนี้หารด้วย 5 ลงตัว ดังนี้ทุกหลักจะมีความยาวเท่ากัน ถ้าหากความยาวของข้อความไม่เท่ากับจำนวนหลักคูณกับความยาวหลัก (หารด้วยจำนวนหลักที่ไม่ลงตัว) แล้วจะพบว่าหลักสุดท้ายจะสั้นกว่าอันอื่นๆ เราอาจใช้ตัวอักษร X ว่าเป็นตัวที่ไม่ใช้บ่อยๆ ในการเติมลงไปในช่องว่างแทน

3.4 ทฤษฎีการเข้ารหัสแบบ Markle-Hellman Knapsacks

Knapsack ปัญหาของการเลือกสิ่งของเพื่อให้ได้น้ำหนักตามต้องการ (Target sum)

ให้ S = {C1,C2,C3 , ... , Cn} และ Target sum คือ T

ซึ่งจะมี V = {V1,V2,...,Vn} แต่ละตัวมีค่าเป็น 0,1

ที่ทำให้

$$\Sigma(Ci \times Vi) = T$$

ตัวอย่างเช่น S = {4, 7, 1, 12, 10} มี Target sum = 1

เราจะได้ V = {1, 0, 1, 1, 0}

เนื่องจาก 4 + 1 + 12 = 17

หรือ V = {0, 1, 0, 0, 1}

7 + 10 = 17

แต่ถ้าเราใช้ Target sum = 25 ก็จะไม่มี Solution

ตัวอย่าง

Plaintext	1 0 1 0 0 1	0 1 1 0 1 0
Knapsack	1 2 5 9 20 43	1 2 5 9 20 43
Ciphertext		
Encrypted Massage	1 + 5 + 43 =	2 + 5 + 20 =
Target sum	49	27

Plaintext จะถูกแบ่งเป็นบลอคแต่ละบลอคมีขนาดเท่ากับ knapsack นำค่าของ plaintext มาคูณกับ Knapsack แบบบิทต่อบิท แล้วนำผลที่ได้มารวมกันก็จะได้ Ciphertext ของบลอคนั้น

3.5 ทฤษฎีการเข้ารหัสแบบ DES Algorithm

3.5.1 ประวัติความเป็นมา

ในปี ค.ศ. 1970 U.S. National Bureau of Standards (NBS) ได้ให้การรับรองความปลอดภัยทางด้านเทคนิคการเข้ารหัส ซึ่งสามารถถูกใช้โดยทางราชการ U.S. Department of Defense และ U.S. Department of state ยังคงให้ความสนใจในระบบการเข้ารหัสและสนับสนุนผู้เชี่ยวชาญในด้านเทคโนโลยีการเข้ารหัส

เวนเดอร์(Vendors)หลายรายยังคงพัฒนาอุปกรณ์การเข้ารหัส โดยใช้เครื่องจักรกลหรือโปรแกรม ซึ่งบุคคลหรือองค์กรสามารถที่จะซื้อได้เอาไว้ป้องกันในการสื่อสาร มันเป็นการยากมากในการแลกเปลี่ยนข้อมูลกันระหว่างผู้ใช้ 2 คนที่ใช้อุปกรณ์ต่างกัน นั่นคือ ไม่สามารถแลกเปลี่ยนข่าวสารที่เข้ารหัสได้ นอกจากนี้ยังไม่มีใครที่จะทดสอบอุปกรณ์เหล่านี้ได้อย่างอิสระมาตรฐานการเข้ารหัสต้องการโปรโมตความสามารถของส่วนที่ไม่เกี่ยวข้องกับการแลกเปลี่ยนข่าวสารที่เข้ารหัสและจัดการระบบเข้ารหัสแบบซิงเกิล(single encyption system) ซึ่งสามารถใช้ทดสอบและรับรองอย่างเป็นทางการ ในปี ค.ศ. 1972 NBS ประกาศแบบข้อเสนอ ระบบเข้ารหัสแบบพับบลิค (public encryption algorithm) สิ่งที่ใช้ประกอบการพิจารณาสำหรับอัลกอริทึม(algorithm) คือ

1. อัลกอริทึมต้องมีระดับความปลอดภัย
2. อัลกอริทึมต้องมีรายละเอียดสมบูรณ์และง่ายต่อการเข้าใจ
3. อัลกอริทึมตัวมันเองก็ต้องจัดความปลอดภัยด้วย ในการรักษาความปลอดภัยไม่ควรปิดบัง public encryption algorithm
4. อัลกอริทึมต้องมีประโยชน์สำหรับผู้ใช้ทุกคน
5. อัลกอริทึมต้องสามารถดัดแปลงแก้ไขได้สำหรับใช้ในหลายๆ แอปปิเคชั่น
6. อัลกอริทึมต้องประหยัดในการสร้างอุปกรณ์อิเลคทรอนิคส์
7. อัลกอริทึมต้องมีประสิทธิภาพในการใช้
8. อัลกอริทึมต้องสามารถตรวจสอบความถูกต้องได้
9. ระบบเข้ารหัสแบบพับบลิคต้องเป็นสินค้าออกได้

ชัดเจนมากจากเครื่องประกอบการพิจารณาข้อ 6 NBS ได้เห็นว่า ระบบเข้ารหัสแบบพับบลิค สามารถที่จะถูกจัดแยกอุปกรณ์ทางด้านฮาร์ดแวร์ได้ จากเครื่องประกอบการพิจารณาข้อ 3 NBS ต้องการให้เปิดเผยอัลกอริทึมในด้านการรักษาความปลอดภัยของระบบโดยใช้คีย์ (ซึ่งจะอยู่ภายใต้การควบคุมของผู้ใช้)

A data encryption algorithm ของ Lucifer ได้ถูกการพัฒนาโดย IBM สำหรับ NBS ; algorithm อันนี้ได้เป็นที่รู้จักในนามของ DES (Data Encryption Standard) ถึงแม้ว่าชื่อจริงของมัน คือ DEA (Data Encryption Algorithm) ในสหรัฐ และ DEAI (Data Encryption Algorithm-1) ในอีกหลายๆ ประเทศ

3.5.2 ลักษณะของDESอัลกอริทึม

DES มีความระมัดระวังและความซับซ้อนในการรวม 2 ฟันดาเมนทอลบลิวดิ้งบลอค(fundamental building block)ของการเข้ารหัส : substitution และ permutation (transposition) อัลกอลิทึม คือ จะมีการทำแอปพลิเคชั่นซ้ำของทั้ง 2 เทคนิค ซึ่งจะมี 16 วัฏจักร (cycle) สูงสุด ในความซับซ้อนของการติดตาม แต่ละบิททำซ้ำถึง 16 ครั้ง ของ substitution และ permutation

Plaintext จะถูกเข้ารหัสเป็นบล็อคๆ ละ 64 บิท ถึงแม้ว่าคีย์จะยาว 64 บิท ในการทำให้คีย์เป็นจำนวน 56 บิท คีย์จะถูกเปลี่ยนแปลงโดยผู้ใช้ที่ time securityของคีย์เก่าที่ไม่แน่นอน ทั้ง 2 cipher ของ DES จะเป็น substitution และ permution, Plaintext จะถูกกระทำโดยลำดับของ cycle ของการทำ substitution ก่อนแล้วไปทำ transposition ต่อการกระทำ substitution และ permutation ซ้ำถูกแสดงในรูปที่ 1

อัลกอลิทึมจะใช้มาตรฐานการปฏิบัติทางคณิตศาสตร์และตรรกศาสตร์เพียงอย่างเดียวกับจำนวน 64 บิท ดังนั้น มันจึงสมควรที่จะสร้างในซอฟแวร์บนเครื่องคอมพิวเตอร์ในปัจจุบัน ถึงแม้ว่ามันจะซับซ้อน อัลกอลิทึมถูกทำซ้ำ, การผลิตมีจุดประสงค์ที่ทำบนชิงเกิลชิพ(single-chip) ในความจริงชิพจำนวนมากที่มีในตลาดสำหรับใช้เป็นส่วนประกอบเบื้องต้นในอุปกรณ์ซึ่งใช้การเข้ารหัส DES ในแอปปิเคชั่น

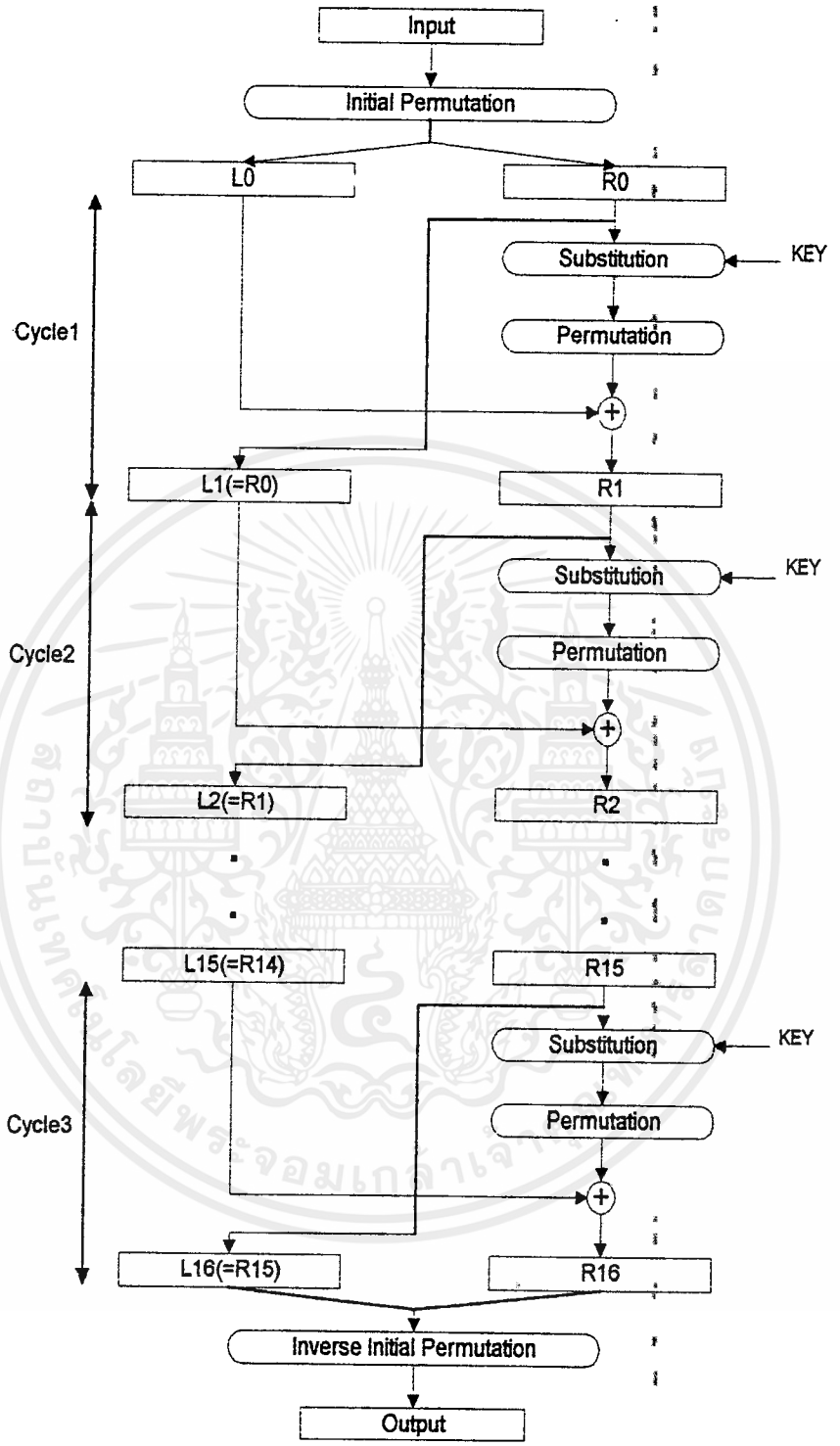
3.5.3 รายละเอียดของการเข้ารหัส

อัลกอลิทึมเป็นผลมาจากทฤษฎีของ Shannon ซึ่งเกี่ยวกับการปิดบังข่าวสาร ซึ่งแนะนำ 2 วิธีในการปกปิดข่าวสารนั่นคือ confusion และ diffusion

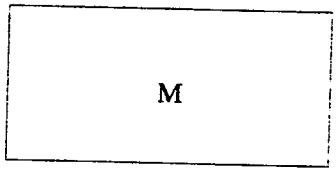
Confusion คือ การทำให้ชิ้นส่วนของ information ถูกเปลี่ยนไป ดังนั้น output bit จะสังเกตุไม่เห็นความสัมพันธ์กับ input bit

Diffusion จะทำการกระจาย Plaintext bit ไปที่บิทอื่นใน Ciphertext

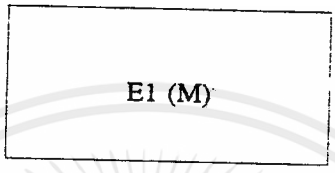
พื้นฐานของ Lucifer และ DES คือ 2 different cipher, Shannon โน้ตไว้ว่าจุดด้อยของความสัมพันธ์ทั้ง 2 สามารถที่จะทำขึ้นให้มีความปลอดภัยมากขึ้นโดยการแอพพายร์สิ่งทั้งสองเข้าด้วยกัน เรียกว่า Product ของสอง cipher, Product ของสอง cipher



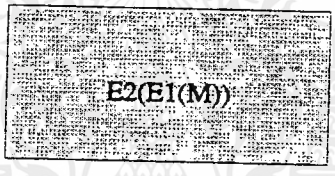
รูปที่ 3.1 Interative Substitution and Transposition



Original Message



After Applying Cipher E1



After Applying Product of Cipher E1, E2

รูปที่ 3.2 Product Ciphers

อัลกอลิทึม DES ที่กระทำบนบล็อกของdata datablock จะถูกแยกออกเป็น 2 ส่วน แต่ละส่วนจะแยกออกเป็นอิสระต่อกัน จากนั้นจะทำการรวมคีย์กับส่วนใดส่วนหนึ่งของข้อมูลและก็สลับกัน 2 ส่วนโปรเซส(Prcess) นี้จะทำซ้ำ 16 ครั้ง อัลกอลิทึมในการทำซ้ำจะใช้ Table lookup และการปฏิบัติ Simple bit ถึงแม้ว่าการจัดการระดับบิทของอัลกอลิทึมจะยุ่งยากซับซ้อน

อินพุทที่เข้ามา DES จะแบ่งอินพุทออกเป็นบล็อกๆ ละ 64 บิท ซึ่งจะถูกเปลี่ยนไปใช้คีย์ขนาด 64 บิท ข้อมูลขนาด 64 บิท จะถูกสับเปลี่ยนตำแหน่งโดย Initial Permulation และคีย์จะถูกลดลงจาก 64 บิท เหลือ 56 บิท โดยการดรอบบิต 8, 16, 24, ... , 64 ซึ่งบิทเหล่านี้จะถูกกำหนดเป็นพาริตี้บิท ซึ่งภายในคีย์จะไม่มีข้อมูลอยู่เลย

ตาราง ก. Initial Permutation : IP (การสับเปลี่ยนตำแหน่งบิตเริ่มต้น)

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

จากตาราง ก. บิตที่ 1 จะเป็นบิตในตำแหน่งที่ 58 ของข้อมูลเดิม บิตที่ 2 จะเป็นบิตในตำแหน่งที่ 50 ของข้อมูลเดิมเป็นลักษณะนี้เรื่อยไปตามตาราง IP

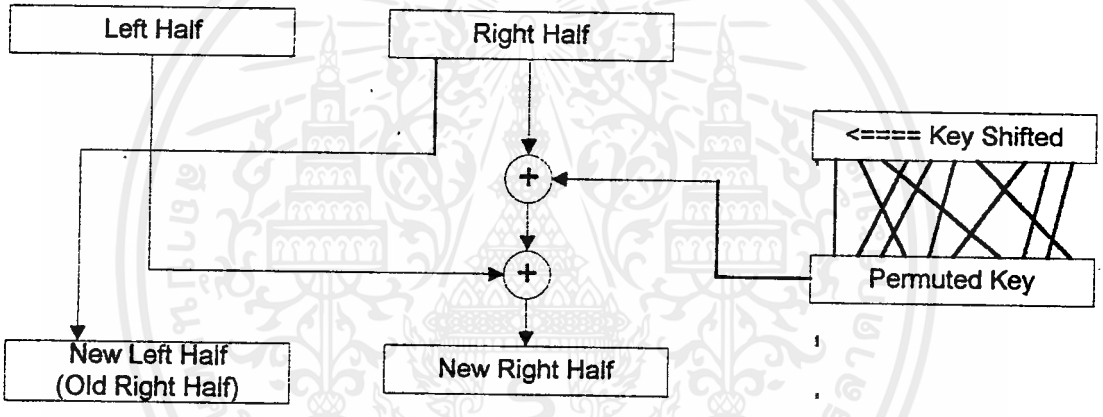
64 Permuted data bit จะถูกแบ่งเป็นครึ่งซ้ายและครึ่งขวา (แต่ละครึ่งมีขนาด 32 บิต) และคีย์จะถูกชิพไปทางซ้ายโดยการกำหนดที่จำนวนบิทและจะทำการสลับตำแหน่งต่อไปคีย์จะถูกรวมกับครึ่งขวา หลังจากนั้นก็จะมารวมกับครึ่งซ้ายใหม่อีกครั้ง ผลลัพธ์ของการรวมนี้จะเปลี่ยนเป็นครึ่งด้านขวาใหม่ ส่วนครึ่งขวาเก่าจะกลายมาเป็นครึ่งซ้ายใหม่ กิจกรรมเหล่านี้จะทำเป็นวงรอบ (cycle) ดังรูปที่ 3 วงรอบจะถูกทำซ้ำ 16 ครั้ง หลังจากวงรอบสุดท้ายซึ่งเป็น Final Permutation ซึ่งจะถูก Invers Initial Permutation (IP) หรือการสับเปลี่ยนตำแหน่งบิตผกผันกับแบบเริ่มต้น ตามตาราง ข. จะได้ผลลัพธ์สุดท้ายออกมา คือ ข้อมูลที่ถูกทำการเข้ารหัสแล้ว

ตาราง ข. Invers Initial Permutation

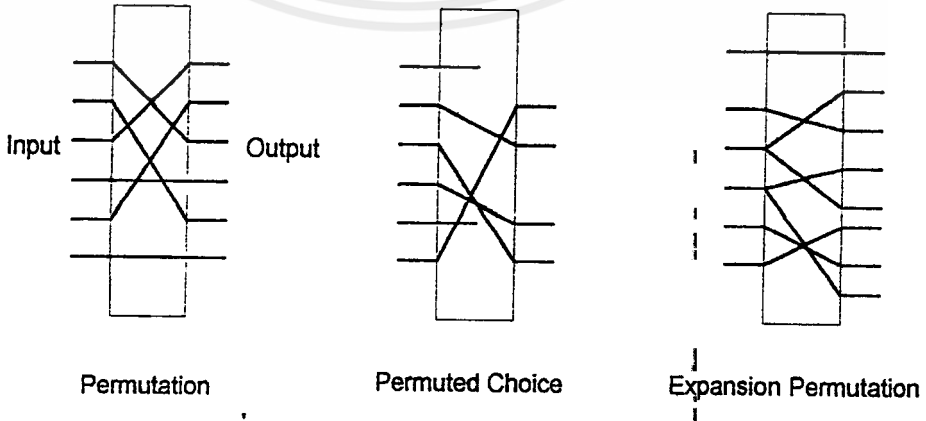
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

ในการรวม 32 บิทของครึ่งขวากับ 64 บิทคีย์ จะมีการเปลี่ยนแปลง 2 อย่างคือ

1. คีย์ 32 บิทจะถูกขยายเป็น 48 บิท โดยการทำ repeating certain bits
2. คีย์ 64 บิทจะถูกลดลงเหลือ 48 บิท โดยการเลือก certain bit อย่างเดียว



รูปที่ 3.3 แสดงวัฏจักรของ DES



รูปที่ 3.4 แสดงชนิดของ Permutation

จากรูปที่ 4 จะเห็นว่าการสลับเปลี่ยนตำแหน่งข้อมูล (PERMUTATION) มีการใช้ใน 3 ลักษณะ คือ

1. Permutation จะเป็นการสับเปลี่ยนตำแหน่งของบิตแบบธรรมดา เมื่อสับเปลี่ยนตำแหน่งเสร็จจะยังคงมีจำนวนบิตเท่าเดิม
2. Permutation Choice การสับเปลี่ยนตำแหน่งบิตของข้อมูล โดยนำบางบิตของข้อมูลมาใช้เมื่อสับเปลี่ยนตำแหน่งข้อมูลแล้วจะได้จำนวนบิตน้อยลง
3. Expancion Permutation การสับเปลี่ยนตำแหน่งของบิตของข้อมูลโดยมีการใช้บางบิตของข้อมูลซ้ำ เมื่อสับเปลี่ยนเสร็จจะมีจำนวนบิตเพิ่มขึ้น

3.5.4 รายละเอียดในวัฏจักร(cycle)ของ DES อัลกอริทึม

แต่ละรอบของการทำซ้ำจะมี 4 โอเปอร์เรชั่น(Operation) คือ

1. Expansion Permutation
2. Key Transformation
3. S-Box
4. P-Box

จากรูปข้างล่างจะเห็นว่า right half จะถูกขยายจาก 32 บิต ไปเป็น 48 บิต แล้วมันจะรวมกับ key ผลลัพธ์ของการทำ Operation อันนี้จะถูก Substituted ได้เป็นผลลัพธ์ตัวใหม่และจะถูกทำให้เหลือ 32 บิต ที่เวลาเดียวกัน จากนั้นทั้ง 32 บิตจะถูก Permuted และจากนั้นจะถูกรวมกับ Left half กลายเป็น Right half ใหม่ ดังรูป

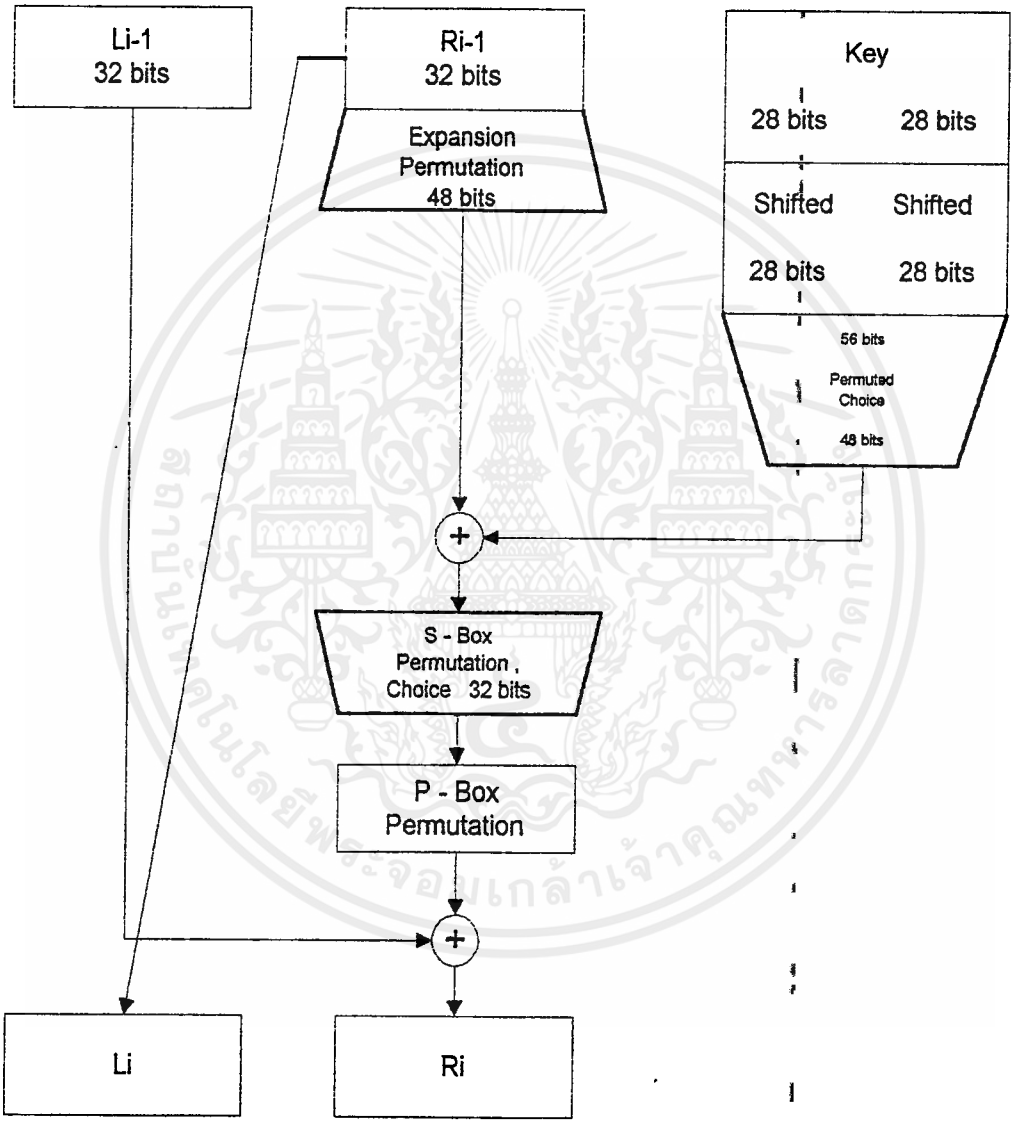
3.5.5 เอกซ์แพนชั่น เพอร์มิวเตชั่น(Expansion Permutation)

แต่ละ Right half จะถูกขยายออกจาก 32 บิต เป็น 48 บิต โดย เอกซ์แพนชั่น เพอร์มิวเตชั่น จุดประสงค์ของเอกซ์แพนชั่น เพอร์มิวเตชั่น คือ

1. ทำขนาดของ Right half ให้เท่ากับขนาดของคีย์
2. ทำการจัดความยาวของผลในตอนท้ายที่จะถูกบีบอัด

Expansion Permutation ถูกกำหนดโดยตารางที่ 1 แต่ละบลอคมี 4 บิต บิตที่ 1 และบิตที่ 4 จะถูก Duplicated ขณะที่บิตที่ 2, 3 เหมือนเดิม ตารางนี้ใช้แสดงตำแหน่งแอาท์พุทของ

บิทอินพุทที่ย้ายแต่ละแถวแสดงการเคลื่อนย้ายของ 8 บิต ตารางนี้แสดงว่าบิตที่ 1 จะถูกย้ายไปที่ตำแหน่ง 2 และ 48 ของ o/p ขณะที่บิตที่ 10 ไปอยู่ตำแหน่ง 15



รูปที่ 3.5 แสดงรายละเอียดในวัฏจักรของ DES

ตารางที่ 1 EXPANSION PERMUTATION

Bit	1	2	3	4	5	6	7	8
Move to	2.48	3	4	5.7	6.8	9	10	11.13
Bit	9	10	11	12	13	14	15	16
Move to	12.14	15	16	17.9	18.20	21	22	23.25
Bit	17	18	19	20	21	22	23	24
Move to	24.26	27	28	29.31	30.32	33	34	35.37
Bit	25	26	27	28	29	30	31	32
Move to	36.38	39	40	41.43	42.44	45	46	47.1

3.5.6 การเปลี่ยนคีย์ (Key Transformation)

64 bit คีย์จะเปลี่ยนเป็น 56 bit คีย์โดยการลบทุกๆ 8 บิต ที่แต่ละรอบคีย์จะถูกแยกออกเป็น 2 ส่วนๆ ละ 28 บิต ทั้ง 2 ส่วนจะทำการเลื่อนไปทางซ้ายโดยการระบุจำนวนหลักและทั้งส่วนจะแปะติดกันอีกครั้งหนึ่งและเมื่อผ่าน Permuted Choice จะเหลือ 48 บิต

Key จะรวมกับ Right half ที่ Exclusive-or function ผลลัพธ์จะไปที่กล่องเอส (S-box) ในขั้นตอนต่อไป ในแต่ละรอบคีย์ทั้ง 2 ส่วนจะไม่ขึ้นแก่กันในการเลื่อนไปทางซ้ายเป็นวงกลมโดยการบ่งบอกจำนวนตำแหน่งของบิต จำนวนของตำแหน่งบิทที่เลื่อน ที่ให้ไว้ในตารางที่ 2

ตารางที่ 2 NUMBER OF BIT OF CIRCULAR SHIFTS FOR EACH CYCLE

Cycle number	Bits Shifted
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2

9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

จากตารางที่ 3 จะเห็นว่า Choice Permutation จะทำการเลือก 48 บิต จาก 56 บิต ตัวอย่างคือ บิตที่ 1 จะถูกเลือกไปที่ตำแหน่งเอาท์พุทที่ 5 ส่วนบิตที่ 9 จะ Ignored ในรอบนี้

ตารางที่ 3 CHOICE PREMUTATION TO SELECT 48 KEY BITS

Key bit	1	2	3	4	5	6	7	8	9	10	11	12	13
Selected for position	5	24	7	16	6	10	20	18	-	12	3	15	23
Key bit	15	16	17	18	19	20	21	22	23	24	25	26	27
Selected for position	9	19	2	-	14	22	11	-	13	4	-	17	21
Key bit	29	30	31	32	33	34	35	36	37	38	39	40	41
Selected for position	47	31	27	48	35	41	-	46	28	-	39	32	25
Key bit	43	44	45	46	47	48	49	50	51	52	53	54	55
Selected for position	-	37	34	43	29	36	38	45	33	26	42	-	30

3.5.7 กล่องเอส (S-Boxes)

Substitution จะกระทำกับ 8 กล่องเอส

กล่องเอส คือ ตารางที่ 6 บิทข้อมูลจะถูกแทนด้วย 4 บิทข้อมูล, อินพุทจำนวน 48 บิท จะถูกแบ่งออกเป็น 4

บลอค(Block ละ 6 บิท) ซึ่งจะถูกกำหนดเป็น B1 B2 ... B8

บลอค B1 จะถูกปฏิบัติงานบนกล่องเอส Si ดังแสดงในรูปที่ 7

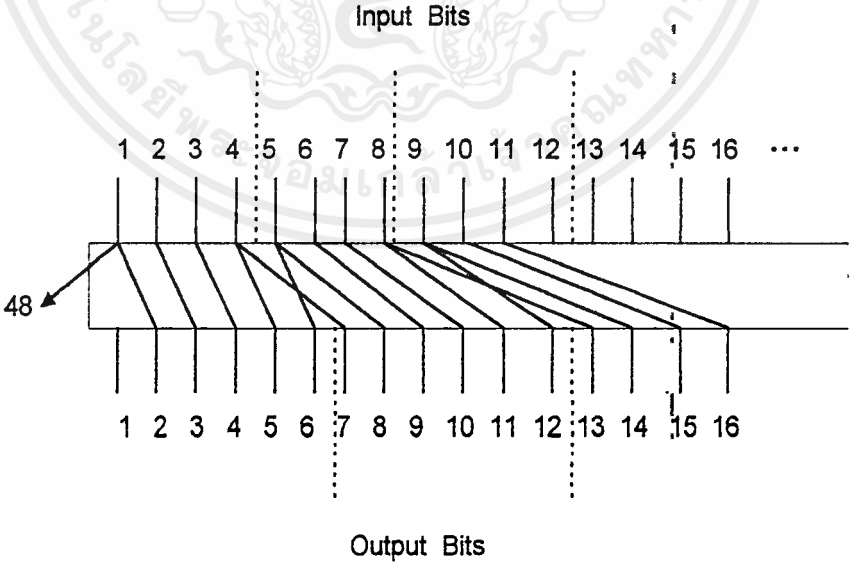
ตารางที่ 4 Si จะมี 4 แถว(row), 16 หลัก(col) สมมติว่าบลอค Bi คือ 6 บิต b1 b2 b3 b4 b5 b6 บิต b1 และ b6 จะหยิบไว้ด้วยกันจาก 2 บิต ไบนารี b1 b6 มีค่า Decimal 0-3 เรียกว่า r บิต b2 b3 b4 b5 มี 4 บิต ไบนารี 0-15 เรียกว่าค่า c ในการทำ Substitution ของ กล่องเอส จะเปลี่ยนจาก 6 บิต บลอค Bi เหลือ 4 บิต ผลลัพธ์ที่แสดงใน แถว r, หลัก c ของส่วน Si ของตาราง 4

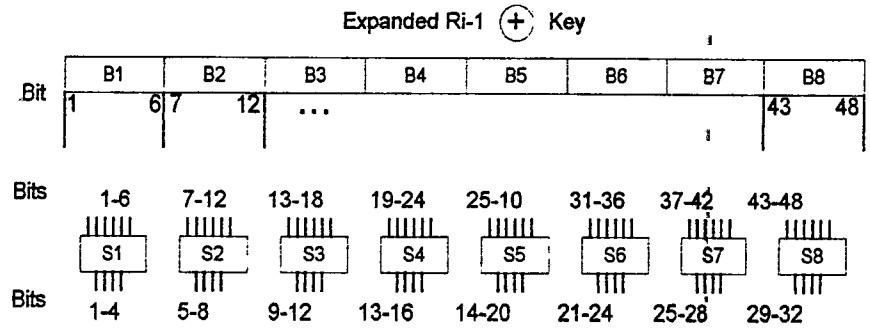
ตัวอย่าง บลอค B7 มี ข้อมูล 010011 จะทราบว่า r = 01 = 1 และ c = 1001 = 9 ที่ S7 จะพบแถว 1, หลัก 9 จะมีค่าเท่ากับ 3 = 0011 ใช้แทนค่า 010011

3.5.8 กล่องพี (P-Boxes)

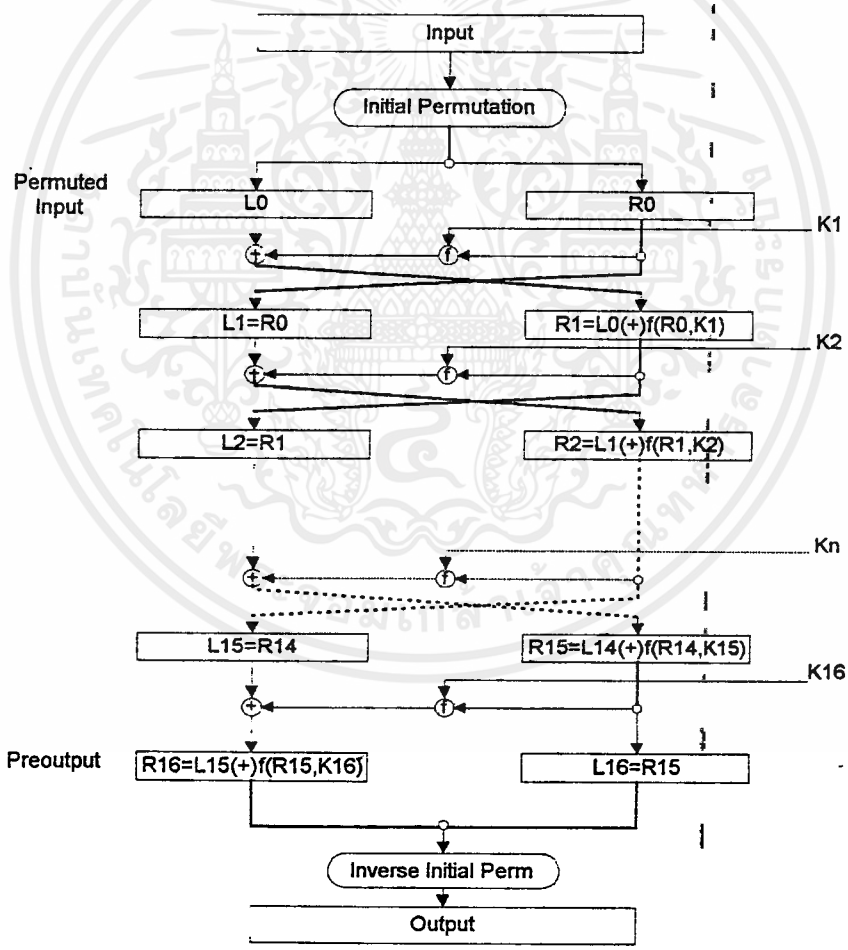
32 บิตของกล่องแอส จะไปที่ Permutation กล่องพี ในตารางที่ 5 แสดงตำแหน่งที่บิตจะย้ายเปลี่ยนที่ไป เช่น บิต 1 จะไปอยู่ที่ตำแหน่งบิต 16 ในขณะที่บิต 10 จะย้ายไปที่ตำแหน่ง 15

รูปที่ 3.6 (ภาพล่าง) Pattern of Expansion Permutation





รูปที่ 3.7 S-Boxes Operation on Eight 6-bit Block



รูปที่ 3.8 แสดง DES ที่สมบูรณ์

3.5.9 การถอดรหัสDES

Substitution และ Permutation ของ DES ดูเหมือนมีจำนวนที่ถูกเลือกแบบ Random นั่นคือไม่ปรากฏ Pattern ไปที่ตารางถอดรหัส ซึ่งหลายอย่างที่เปลี่ยนแปลงไป DES algorithm ใช้ได้ทั้งการเข้ารหัสและการถอดรหัส

ผลลัพธ์จะถูก เพราะว่าวงรอบ j derives มาจาก วงรอบ j-1

$Lj = Rj-1$ --------- (1)

$Rj = Lj-1 \oplus f(Rj-1, kj)$ --------- (2)

ทั้ง 2 สมการแสดงให้เห็นว่า ผลลัพธ์ของแต่ละ วงรอบ สัมพันธ์กับวงรอบก่อน และถ้าเขียนใหม่ สมการเหล่านี้ในรูปของ Rj-1 และ Lj-1 คือ

$Rj-1 = Lj$ --------- (3)

$Lj-1 = Rj \oplus f(Rj-1, kj)$ --------- (4)

(3) แทน (4) ได้

$Lj-1 = Rj \oplus f(Lj, kj)$ --------- (5)

สรุปแล้ว การเข้ารหัส คือ การทำ DES algorithm FORWARD

การถอดรหัส คือ การทำ DES algorith BACKWARD

แต่ต้องใช้ Function เดียวกันในการทำตลอด DES algorithm ส่วน Key ต้องเรียงลำดับย้อนกลับในการถอดรหัส คือ (k16, k15, k14,... k1)

ตารางที่ 4 S-BOX TABLE FOR THE DES

Row	0	1	2	3	4	5	6	Colum	8	9	10	11	12	13	14	15
S1																
0	14	4	13	1	2	15	11		4	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13		6	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2		23	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1		17	5	11	3	15	10	0	6	13
S2																	
0	15	1	3	14	6	11	3		8	9	7	2	13	12	0	5	15
1	3	13	4	7	15	2	8		0	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13		6	5	8	12	6	9	3	2	9
3	13	8	10	1	3	15	4		3	11	6	7	12	0	5	14	0

S3																	
	0	10	0	9	14	6	3	15	23	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	4	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	6	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	7	13	14	3	0	6	9	23	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	45	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	5	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	11	9	4	5	11	12	7	2	14
S5																	
	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	1	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	1	14	2	13	6	15	0	9	10	4	5	3
S6																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	6
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	2
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7																	
	0	4	11	2	14	15	0	8	13	1	12	9	7	5	10	6	7
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	2
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	8
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	11
S8																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	81
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	7
	2	7	11	4	1	9	12	14	2	0	6	10	13	25	3	5	12
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	9

ตารางที่ 5 PERMUTATION BOX P

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

3.5.10 การวิเคราะห์อัลกอริทึมDES

จะเห็นว่าโครงสร้างภายในของอัลกอริทึม DES จะเป็นวงจรการทำงานที่ซ้ำ ๆ กันถึง 16 รอบ ซึ่งประกอบด้วยการที่ข้อมูล การสับตำแหน่งข้อมูล และการเอ็กซ์คลูซีฟออร์ ทำให้การเข้าข้อมูลแบบนี้มีความซับซ้อนมากและไม่สามารถที่จะย้อนรอยกลับไปเพื่อหาข้อมูลแท้จริงได้ วิธีการที่จะค้นหาข้อมูลที่แท้จริงนั่นได้ คือ การค้นหาคีย์ที่ใช้สำหรับการเข้ารหัสโดยใช้วิธีการค้นแบบไล่เรียง (Exhaustive Search) เพราะเมื่อทราบคีย์ก็ถอดรหัสข้อมูลได้โดยในการค้นหาแบบนี้ได้จะต้องรู้ข้อมูลธรรมดาและข้อมูลที่เข้ารหัสที่คู่กันบางส่วน และคีย์ที่เป็นไปได้ทั้งหมดในการเข้ารหัสมีประมาณ 2^{56} หรือ $7*10^{16}$ คีย์ การค้นหาว่าคีย์ใดเป็นคีย์ที่แท้จริงก็จะต้องนำคีย์นั้นมาเข้ารหัสข้อมูลธรรมดาแล้วดูผลลัพธ์ว่าเหมือนกับข้อมูลเข้ารหัสที่มีอยู่แล้วหรือไม่ ถ้าเหมือนก็แสดงว่าคีย์ถูกต้อง แต่ถ้าไม่เหมือนก็ต้องเข้ารหัสด้วยคีย์อื่นต่อ ๆ ไป

ในปัจจุบันได้มีหน่วยงานต่าง ๆ ได้รับเอาวิธีการเข้ารหัสแบบนี้มาใช้อย่างแพร่หลาย ตัวอย่างเช่น Home Bos Office มีการเข้ารหัสที่เป็นสัญญาณของสายเคเบิ้ลทีวีด้วย อัลกอริทึม DES หรือ CIRRUS National Banking Network มีการใช้อุปกรณ์สำหรับการเข้ารหัสที่ใช้อัลกอริทึม DES มาใช้กับระบบถอนเงินอัตโนมัติ (Electronic Transter) และอัลกอริทึม DES ปรับเปลี่ยนให้เป็นมาตรฐาน (Inter-agency Electronic Fundation โดยรัฐบาลสหรัฐอเมริกา

การพัฒนาด้านเทคโนโลยีที่ก้าวหน้ามากขึ้นอาจจะมีผลกระทบต่ออัลกอริทึม DES จะเห็นว่าความเร็วของคอมพิวเตอร์เพิ่มขึ้นจาก 10 MFLOP (million floating operations per second) เป็น GFLOP (a billion of link point operations per second) นอกจากนี้การประมูลผลแบบขนาน (Parallel Processing) และการประมวลผลเวกเตอร์ (Vector Processing)

3.6 ทฤษฎีการเข้ารหัสแบบ RSA Algorithm

RSA คือ การเข้ารหัสชนิดเปิดเผยคีย์ได้ของระบบที่ใช้สำหรับการเข้ารหัสและยืนยันข้อมูล RSA ได้ถูกสร้างขึ้นโดย Ron Rivest, Adi Shamir และ Leonard Adleman ในปี 1977 เราจะศึกษาในเรื่องของ number และ factor

ซึ่งการเข้ารหัสและถอดรหัสเพื่อไม่ให้มีการก้าวก่ายในส่วนของไพร์เวทคีย์ (private key) ของ ผู้ใช้เองเท่านั้น แต่ละคนสามารถจะส่งข้อมูลที่ทำการเข้ารหัสหรือนำข้อมูลออกมาใช้ได้ในส่วนของพับบลิคคีย์(public keys)

ระบบ RSA เป็นระบบที่มีความซับซ้อนของแฟคเตอร์ริ่ง(factoring)แต่ถ้าทราบวิธีของ แฟคเตอร์ริ่ง แล้วจะสามารถแปรความหมายของข้อมูลที่เข้ารหัสโดย RSA ได้ไม่ยาก

RSA จะสนับสนุนและทำให้ DES มีประสิทธิภาพในการเข้ารหัสซึ่งจะใช้ร่วมกันในการรักษาความปลอดภัยของข้อมูลในระบบสื่อสาร

RSA จะมี 2 หน้าที่ที่สำคัญซึ่ง DES ไม่มีการเข้ารหัสข้อมูลบ่อยครั้งที่ RSA และ DES จะใช้งานร่วมกัน คือ เมื่อข้อมูลถูกเข้ารหัสโดยการสุ่ม DES คีย์

ก่อนที่จะส่งออกไปก็จะมีการนำ DES คีย์มาเข้ารหัสด้วย RSA และจะถูกส่งออกไปพร้อมกับข้อมูล (Ciphertext) ใน พับบลิคแชนแนล(Public Channel) ข้อมูลในส่วนนี้จะไม่เป็นความลับ คือ จะไม่มีการเข้ารหัส

คุณอาจสงสัยว่าทำไมไม่ใช้อย่างใดอย่างหนึ่งเลยในการเข้ารหัส เนื่องจาก RSA จะทำได้ดีกับข้อมูลที่สั้นๆ แต่ DES หรือการเข้ารหัสแบบอื่นๆ สามารถที่จะเข้ารหัสข้อมูลยาวๆ ได้ดีกว่าการเข้ารหัสแบบ RSA

บางครั้งการเข้ารหัสแบบ RSA ไม่จำเป็นอาจใช้การเข้ารหัสแบบ DES ก็เพียงพอ เช่นในระบบผู้ใช้คนเดียว(Singleuser) แต่ถ้าเป็นระบบผู้ใช้หลายคน(Multiuser) ระบบต้องการการเข้ารหัสที่มีความปลอดภัยในการส่งข้อมูลซึ่งอาจจะต้องใช้ RSA หรือ พับบลิคคีย์อื่นๆ ของระบบ

RSA จะใช้การคำนวณมาเกี่ยวข้องกับการเข้ารหัสซึ่งจะทำให้ผู้ที่ต้องการจะนำข้อมูลไปใช้โดยไม่มีสิทธิกับข้อมูลนั้น ก็ไม่สามารถที่จะนำข้อมูลนี้ไปใช้ได้หรืออาจต้องใช้เวลานานในการแปลความหมายของข้อมูล (ถ้าไม่ทราบคีย์หรือวิธีการในการถอดรหัส)

3.6.1 ลักษณะการทำงานของ RSA

การใช้ RSA จะทำการขยายข้อมูลเล็กๆ เท่านั้นสำหรับการเข้ารหัสนั้น ข้อมูลจะมีความยาวซึ่งจะอยู่ในบลอคที่ซับซ้อน ซึ่งความยาวจะเป็น Modulus (หลายๆ Application มีขนาด 512bit) สำหรับการแสดงข้อมูลจริง ซึ่งข้อมูลจะไม่มีการเข้ารหัสและไม่มีการขยาย

อย่างไรก็ดี คีย์ที่เข้ารหัสจะถูกแสดงต่อท้ายข้อมูลชุดนี้ ซึ่งจะมีความยาว 1 บลอค (Identiry และ Public Key) เพื่อป้องกันบุคคลอื่นใช้คีย์ปลอมในการถอดรหัสข้อมูล

3.6.2 การใช้ RSA ในการเข้ารหัส

RSA จะใช้ร่วมกับซีเครทคีย์(Secrect-Key) ของระบบ เช่น DES ขั้นตอนก็คือจะใช้ DES เข้ารหัสข้อมูลก่อน ซึ่งสิ่งเหล่านี้จะกลายเป็นข้อมูลทางดิจิตอลของ RSA สมมติว่าคุณต้องการส่งข้อมูลไปใช้ Jim คุณจะต้องเข้ารหัสข้อมูลก่อนด้วยวิธีการของ DES โดยจะสุ่ม DESคีย์ขึ้นมาแล้วทำการเข้ารหัสจากนั้นใช้พับบลิคคีย์ ของ Jim เข้ารหัส DESคีย์อีกครั้งหนึ่ง หลังจากเข้ารหัสเสร็จเรียบร้อยแล้ว ก็ทำการส่งข้อมูลทั้งสอง (RSA Digital) ไปใช้ Jim และ Jim ก็จะทำการถอดรหัส DESคีย์ ด้วยไพร์เวทคีย์ ของ Jim และใช้ DESคีย์ถอดรหัสข้อมูลอีกครั้งหนึ่ง

3.6.3 การใช้ RSA แสดงข้อมูล

สมมุติว่าคุณต้องการส่งข้อมูลไปใช้ Jim คุณจะต้องกำหนดขนาดของข้อมูล เรียกว่า ค่าแฮช(Hash Value) เพื่อที่จะสามารถสร้างบลอคในการส่งข้อมูลได้ถูกต้อง ซึ่งค่าแฮช จะเป็น Digital Signature ของข้อมูลแล้วก็ทำการเข้ารหัสข้อมูลด้วย RSAไพร์เวทคีย์ ของคุณ (Your Digital Signature) จากนั้นส่งไปใช้ Jim เมื่อได้รับข้อมูลและ Signature ก็จะถอดรหัส Signature ด้วยพลับบลิคคีย์ของคุณแล้วเขาก็จะหาข้อมูลโดยใช้แฮชฟังค์ชั่น(Hash Function) เหมือนกับที่คุณใช้แล้วก็จะทำการเปรียบเทียบข้อมูลกับ Signature ถ้ามีค่าเท่ากันแสดงว่าข้อมูลนั้นถูกต้อง แต่ถ้าไม่เท่ากันแสดงว่าข้อมูลอาจจะมาจากที่อื่นหรืออาจถูกดัดแปลงระหว่างการส่งมา Jim ก็จะ Reject ข้อมูลนั้น

การตรวจสอบไวรัสของวิธีการของ RSA จะคล้ายกับการตรวจสอบ Transmission Error คือ สามารถตรวจสอบการเปลี่ยนแปลงข้อมูลของ File ที่เก็บอยู่ใน Disk ซึ่งอาจถูกกระทำโดย Viruses

การตรวจสอบไวรัส (Viruses) ของ RSA จะมีการกำหนดสัญลักษณ์ของไฟล์ทุกไฟล์และจะมีการทำเครื่องหมายของการเปลี่ยนแปลงครั้งสุดท้ายที่เกิดขึ้น ถ้าหาก Signture Fail (ถูก Verify) มีการเปลี่ยนแปลงเกิดขึ้น RSA ก็จะฟ้องว่าข้อมูลมีการเปลี่ยนแปลง ซึ่งอาจจะถูกกระทำโดย Viruses แต่ในบางครั้งอาจถูกกระทำโดยสภาพแวดล้อมอื่นๆ เช่น Soruce Code หรือปัญหาทางฟิซิคอล ของ ฮาร์ดดิสก์(Harddisk) แต่ถ้าสงสัยว่าเกิดจากไวรัสก็ควร Runโปรแกรมตรวจสอบไวรัสตรวจสอบดู

3.6.4 การเข้ารหัสของ RSA

ทำงานกับ Arithmetic mod n วิธีการนี้ Plaintext Block จะถูกเปลี่ยนมาเป็น Unsigned Integer มี 2 Keys คือ d and e ซึ่งใช้ในการถอดรหัสและเข้ารหัส

Plaintext Block P จะถูกเข้ารหัสเป็น P^e mod n ดังนั้น Exponentiation จะถูกกระทำโดย mod n มันเป็นการยากมากที่หา Factor P^e มา Encrypted Plaintext เพื่อไม่ให้ผู้อื่นถอดรหัสได้

คีย์ถอดรหัส d จะถูกเลือกอย่างระมัดระวังคือ $(P^e)^d$ mod n = P ผู้รับที่แท้จริงจะทราบค่า d และจะสามารถถอดรหัสได้

3.6.5 รายละเอียดและวิธีการเข้ารหัส

RSA มี 2 คีย์ คือ d และ e ซึ่งทำงานเป็นคู่ของการเข้ารหัสและถอดรหัส Plaintext Massage P จะถูกเข้ารหัสเป็น Ciphertext C โดย

$$C = P^e \bmod n$$

Plaintext ถูกเรียกกลับคืนมาโดย

$$P = C^d \bmod n$$
$$= (P^e)^d \bmod n$$
$$= (P^d)^e \bmod n$$

การเลือกคีย์

คีย์เข้ารหัส ประกอบด้วยคู่ลำดับของ Integer (e, n) และ

คีย์ถอดรหัส ประกอบด้วยคู่ลำดับของ Integer (d, n)

การเลือกค่าของ n ควรเป็นค่าที่มากที่สุด ซึ่งเป็นผลคูณของ 2 Prime p และ q

โดยทั่วไป p และ q มีค่าประมาณ 100 หลัก ดังนั้น n จะมีค่าประมาณ 200 หลัก การเลือกค่า e นั้น ค่า e จะต้องเป็น relatively prime กับ (p-1)*(q-1) คือ จะต้องไม่มี common factor ร่วมกัน วิธีง่ายในการ Guarantee ว่า e คือ Relatively Prime กับ (P-1)*(q-1) คือ ให้เลือก e ที่เป็น Prime ซึ่งต้องมากกว่า (p-1) และ (q-1) ในที่สุด ก็จะเลือก d ได้ว่า

$$e*d \equiv 1 \bmod (p-1)*(q-1)$$

3.6.6 หลักการทางคณิตศาสตร์ที่ใช้ในการเข้ารหัสแบบ RSA

φ (n) = จำนวนเต็มบวกที่น้อยกว่า n และเป็น Relatively rime กับ n

ถ้า p คือ Prime แล้ว

$$\varphi(p) = (p-1)$$

มากไปกว่านั้น ถ้า n = p*q ขณะที่ p และ q เป็น Prime ทั้งคู่

$$\varphi(n) = \varphi(p)*\varphi(q) = (p-1)*(q-1)$$

แสดงในรูปแบบของ Euler และ Fermat

$$x^{\varphi(n)} \equiv 1 \bmod n$$

โดยที่ x = จำนวนเต็มใดๆ และ n กับ x เป็น Relatively Prime

สมมติว่าเราต้องการเข้ารหัส Pliantext ข้อมูล P โดยวิธีของ RSA Algorithm ดังนั้น $E(P) = P^e$ เราจะต้องแน่ใจว่าเราสามารถที่จะควบคุมข้อมูลได้ ค่า e, d เป็น inverses mod $\varphi(n)$

$$e * d \equiv 1 \bmod \varphi(n)$$

หรือ

$$e * d \equiv k * \varphi(n) + 1 \quad ; k = \text{จำนวนเต็ม} \qquad (0)$$

จาก Euler/Fermat ผลลัพธ์จะได้

$$P^{p-1} \equiv 1 \bmod p$$

และ (p-1) เป็น Factor ของ $\varphi(n)$

$$P^{k*\varphi(n)} \equiv 1 \bmod p$$

คูณด้วย P

$$P^{k*\varphi(n+1)} \equiv P \bmod p$$

ถ้าอยู่ในรูปของ q จะได้

$$P^{k*\varphi(n)+1} \equiv P \bmod q$$

นำผลลัพธ์ทั้งสองเข้ากับสมการ (0) จะได้

$$(P^e)^d = P^{e*d}$$
$$= P^{k*\varphi(n)+1}$$
$$\equiv P \bmod p$$
$$\equiv P \bmod q$$

ดังนั้นจะได้

$$(P^e)^d \equiv P \bmod n$$

และ a * b mod n = (a mod n) * (b mod n) mod n

ตัวอย่าง

P = 11, q = 13

n = p * q = 143

$\varphi(n)$ = (p-1) * (q-1) = 10 * 12 = 120

เลือก e ที่ต้องการและ e ต้องเป็น Relatively Prime กับ (p-1) x (q-1) เลือก e = 11

Inverse ของ 11 mod 120 คือ 11

11 * 11 = 121 = 1 mod 120 ซึ่งทั้ง key เข้ารหัสและถอดรหัสจะเหมือนกัน คือ

e = d = 11

ให้ P = massage ที่ถูกเข้ารหัส

ถ้า P = 7

$E(7) = 7^{11} \bmod 143 = 106$

$D(106) = 106^{11} \bmod 143 = 7$

3.6.7 การนำไปใช้งาน

e เป็นค่าที่ถูกเลือก ซึ่งเป็น Relatively Prime to (p-1) * (q-1)

e มักจะเป็น Prime ที่มากกว่า (P-1) หรือ (q-1)

d คือ inverses ของ e mod $\varphi(n)$

User ได้แบ่ง e และ n และเก็บ d เป็นความลับ p, q และ $\varphi(n)$ จะถูกตัดทิ้งไปได้ ในส่วนนี้สังเกตุว่าถ้ารู้ว่า n เป็น Product ของ 2 Primes และถ้า n มีค่ามาก เช่น 100 digit มันจะเป็นไไม่ได้ที่จะหา Prime p และ q หรือ Private Key d จาก e ดังนั้นรูปแบบนี้จึงมีความปลอดภัยให้สำหรับ d

รูปแบบนี้ยังไม่เคยทำการตรวจสอบความถูกต้องของ p และ q เว้นแต่ว่าพวกมันเป็น Prime ตั้งแต่นั้นจะกำหนดการพิจารณาบนลำดับของ 10 ยกกำลัง 50 Factor ที่เป็นไปได้

ฮิวริสติคอัลกอริทึม(Heuristic Algorithm) จาก Soloray และ Strassen [SOL 77] จะทำการศึกษาความเป็นไปได้ของจำนวนจำเพาะต่อระดับความน่าไว้ใจทุกๆจำนวนจำเพาะจะผ่าน 2 การทดสอบ ถ้า p เป็นจำนวนจำเพาะ และ r คือ จำนวนที่น้อยกว่า P

$$\gcd(p,r) = 1$$

และ

$$J(r,p) \equiv r^{(p-1)/2} \bmod p$$

ซึ่ง J (r,p) เป็น Jacobi Function กำหนดดังนี้

$$\begin{aligned} &= 1 && ; \text{if } r = 1 \\ J(r,p) &= J(r/2.p)*(-1)^{(p2-1)/8} && ; \text{if } r \text{ is even} \\ &= J(P \bmod r.r) * (-1)^{(r-1)*(p-1)/4} && ; \text{if } r \text{ is odd}, \neq 1 \end{aligned}$$

ถ้าจำนวนถูกสงสัยว่าเป็นจำนวนจำเพาะหรือไม่ให้ใช้วิธีการนี้ทดสอบได้

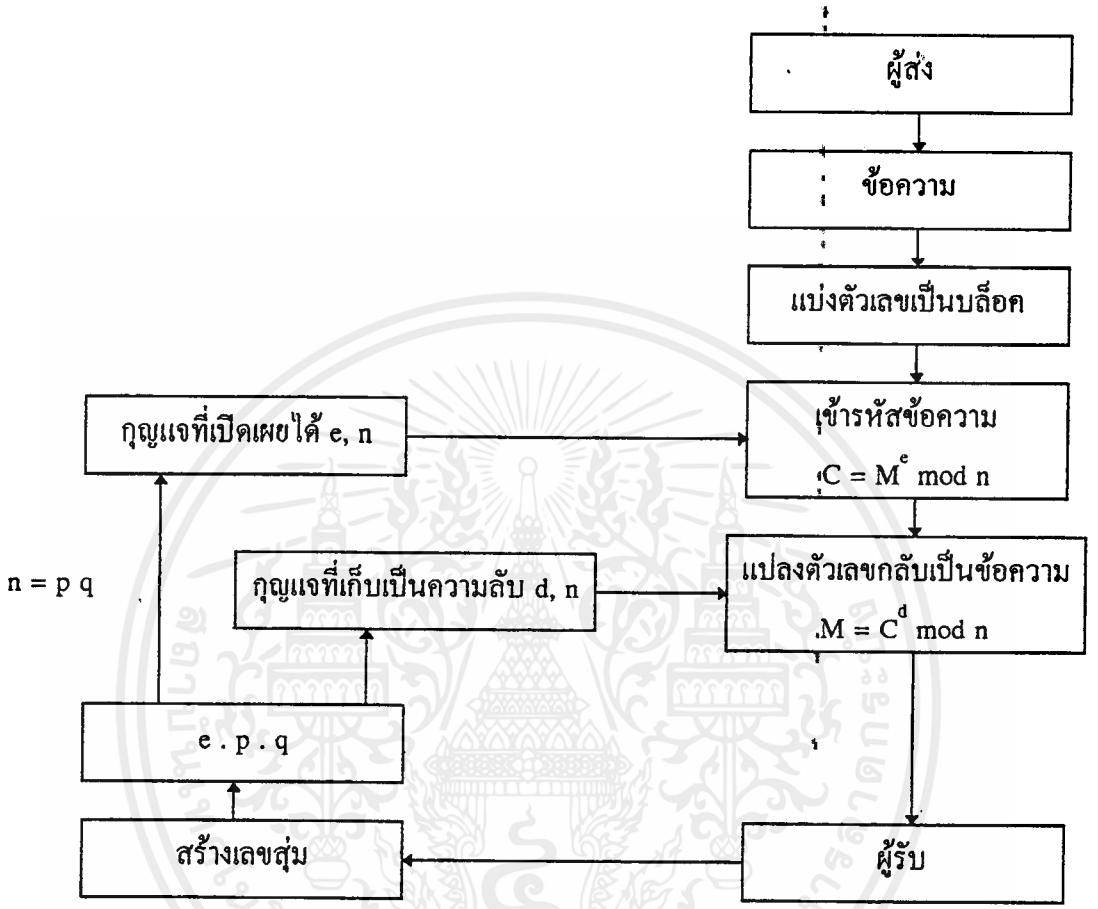
ถ้าจำนวนผ่านการทดสอบจะมีความเป็นไปได้ว่าจำนวนเป็นจำนวนจำเพาะอย่างน้อยที่สุด 1/2

ปัญหาที่เกี่ยวกับ RSAอัลกอริทึม คือการหา 2 จำนวนจำเพาะที่มีค่ามากที่สุดของ p และ q ด้วยวิธีการ Solovay และ Strassen Approch

พยายามเดาค่าจำนวนจำเพาะที่มากๆ ของ p แล้ว Generates Random Number r และคำนวณ gcd(p,r) และ J(r,p) ถ้าผลการ Test Fail จะสรุปว่า p ไม่ใช่จำนวนจำเพาะ และ Procedure จะหยุดการทำงาน

แต่ถ้าทั้งคู่ผ่านเป็นไปได้ว่า p ไม่ใช่จำนวนจำเพาะคือมากที่สุด 1/2 Procedure จะทำซ้ำด้วยค่าใหม่ จากนั้น r จะถูกสุ่มขึ้นมา ถ้า r ผ่านเป็นไปได้ว่าไม่เป็นจำนวนจำเพาะ p สามารถผ่านการทดสอบที่ 1/4 โดยทั่วไปหลังจากทำโปรเซส k ครั้ง โดยไม่ Fail เป็นไปได้ว่า p ไม่เป็นจำนวนจำเพาะ ที่ $(1/2)^k$

3.6.8 รูปแสดงการเข้ารหัสแบบRSA



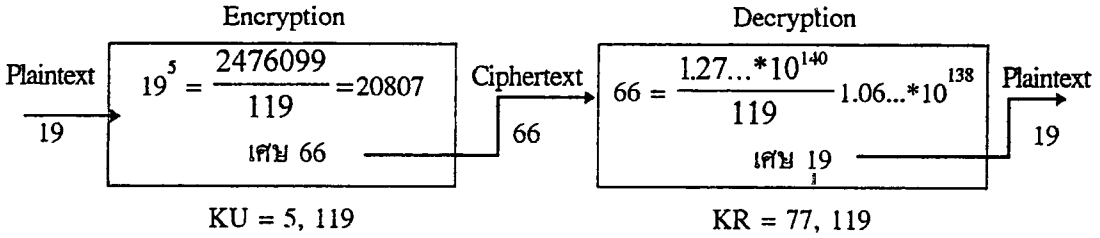
รูปที่ 3.9 ขั้นตอนการเข้ารหัส RSA

3.6.9 การทำ Fast Exponential ใน RSA

ในการเข้ารหัสและการถอดรหัสด้วยวิธีการของ RSA จะต้องทำการคำนวณจำนวนเต็ม (integer) ยกกำลังจำนวนเต็มและนำจำนวนดังกล่าวมาทำการ mod n ถ้าจำนวนเต็มที่ยกกำลังดังกล่าวมีค่ามากๆ เราสามารถใช้วิธีที่ช่วยในการลดจำนวนดังกล่าวแล้วนำมา mod n

เราสามารถใช้คุณสมบัติทางคณิตศาสตร์ :

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$$



รูปที่ 3.10 แสดงตัวอย่างของ RSA

ดังนั้น เราสามารถทำการลดจำนวนเต็มในระหว่างทำการ mod n ไปด้วย โดยที่จะกล่าวถึงต่อไป จะเป็นการทำให้หลักการที่กล่าวมาใช้ได้จริง

อีกสิ่งหนึ่งที่ต้องคำนึงถึง คือ ประสิทธิภาพในการคำนวณเลขยกกำลัง เนื่องจากว่า วิธีการของ RSA ต้องทำการคำนวณเลขยกกำลังจำนวนมากๆ ลองพิจารณาการคำนวณ x^{16} จะพบว่าเราต้องทำการคูณทั้งหมด 15 ครั้งดังนี้

$$x^{16} = x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x$$

การทำให้ได้มาซึ่งผลลัพธ์สุดท้าย เราสามารถทำการคูณ 4 ครั้ง โดยทำการหา x^2, x^4, x^8, x^{16} ตามลำดับ

ในการหาค่า a^m โดย a กับ m เป็นจำนวนเต็มบวก ถ้าจะคำนวณค่า m ให้เร็วต้องทำเป็นเลขฐานสอง $b_k, b_{k-1} \ldots b_0$ จะได้

$$m = \sum_{Bit \neq 0} 2i$$

ดังนั้น

$$a^m = a^{\left[\sum_{bi\neq 0} 2^i\right]} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \prod_{b_i \neq 0} \left[a^{(2^i)} \bmod n\right]$$

ดังนั้นเราสามารถพัฒนาขึ้นมาเป็นอัลกอริทึมสำหรับใช้ในการคำนวณ $a^b \bmod n$

```
C = 0 ; d = 1
for i = k down to 0
        do C = C * 2
           d = (d * d) mod n
           if b_i = 1 then C = C + 1
                           d = (d * a) mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
C	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

3.6.10 ตัวอย่างการเข้ารหัสแบบ อาร์ เอส เอ

ถ้าต้องการเข้ารหัสข้อความว่า "KING MONGKUT INSTITUTE OF TECHNOLOGY" โดยแทนตัวอักษรด้วยตัวเลขต่อไปนี้ A = 01, B = 02, C = 03, D = 04, E = 05, F = 06, G = 07, ... , Z = 26, BLANK = 00 จะแปลงข้อความเป็นตัวเลขได้ดังนี้

11091407001315140711212000091419200 9

20212005001506002005030814151215072 5

เลือกบล็อก p = 73, q = 151 จะได้ n = p q = 11023 ขั้นตอนการเข้ารหัสแบ่งข้อความที่เข้ารหัสเป็นบล็อกๆ ละ 4 หลัก จากนั้นหาโมดูโล n ดังนี้

$C_1 = 1109^{11} \bmod 11023 = 5902$

$C_2 = 1407^{11} \bmod 11023 = 8015$

$C_3 = 0013^{11} \bmod 11023 = 4385$

$C_4 = 1514^{11} \bmod 11023 = 8433$

$C_5 = 0711^{11} \bmod 11023 = 7923$

$C_6 = 2120^{11} \bmod 11023 = 1436$

$C_7 = 0009^{11} \bmod 11023 = 576$

$C_8 = 1419^{11} \bmod 11023 = 9127$

$C_9 = 2009^{11} \bmod 11023 = 3133$

$C_{10} = 2021^{11} \bmod 11023 = 9538$

$C_{11} = 2005^{11} \bmod 11023 = 7420$

$C_{12} = 0015^{11} \bmod 11023 = 10206$

$C_{13} = 0600^{11} \bmod 11023 = 4855$

$C_{14} = 2005^{11} \bmod 11023 = 7420$

$C_{15} = 0308^{11} \bmod 11023 = 9892$

$C_{16} = 1415^{11} \bmod 11023 = 10833$

$C_{17} = 1215^{11} \bmod 11023 = 5601$

$C_{18} = 0725^{11} \bmod 11023 = 3911$

จะสังเกตุได้ว่าขนาดของการแบ่งบล็อกของข้อความจะต้องน้อยกว่าค่า n เพราะเซ็ททั้งหมดของโมดูโล n จะอยู่ในช่วง [0, n-1] เมื่อจะถอดรหัสจะทำได้โดยวิธีเดียวกัน โดยใช้กุญแจถอดรหัสอีกตัวหนึ่ง คือ 5891 เพราะว่า

$5891 = 11 \bmod 10800$

$10800 = (73-1)(151-1)$

$11023 = 73 \times 151$

$M_1 = 5902^{5891} \bmod 11023 = 1109$

$M_2 = 8015^{5891} \bmod 11023 = 1407$

$M_3 = 4385^{5891} \bmod 11023 = 0013$

$M_4 = 8433^{5891} \bmod 11023 = 1514$

$M_5 = 7922^{5891} \bmod 11023 = 0711$

$M_6 = 1436^{5891} \bmod 11023 = 2120$

$M_7 = 576^{5891} \bmod 11023 = 0009$

$M_8 = 9127^{5891} \bmod 11023 = 1419$

$M_9 = 3133^{5891} \bmod 11023 = 2009$

$M_{10} = 9538^{5891} \bmod 11023 = 2021$

$M_{11} = 7420^{5891} \bmod 11023 = 2005$

$M_{12} = 10206^{5891} \bmod 11023 = 0015$

$M_{13} = 4855^{5891} \bmod 11023 = 0600$

$M_{14} = 7420^{5891} \bmod 11023 = 2005$

$M_{16} = 10833^{5891} \bmod 11023 = 1415$

$M_{17} = 5601^{5891} \bmod 11023 = 1215$

$M_{18} = 3911^{5891} \bmod 11023 = 0725$

จากตัวอย่างเดิม ถ้าเลือกค่า p และ q ที่มีจำนวนหลักมากๆ จะลดจำนวนครั้งของการเข้ารหัสซึ่งทำให้การเข้ารหัสเร็วขึ้น

เลือก p = 232039415285101600928 33 (จำนวนเฉพาะขนาด 23 หลัก)

q = 17357390175479470655 87 (จำนวนเฉพาะขนาด 23 หลัก)

จะได้ n = pq

= 40275986671936234424621110207

398375752479552 (44 หลัก)

W(n) = (p-1)(q-1)

= 40275986671933623442462111

0207398375752479552

จากคุณสมบัติ e d = 1 mod (p-1)(q-1) จะได้กุญแจเข้ารหัส e = 7

กุญแจถอดรหัส d = 57537123817051763463744 44

3163426251074970 79

ซึ่ง ed = 40275986671936234424621110207398 3

75752479553

จากข้อความเดิม "KING MONGKUT INSTITUE OF TECHNOLOGY" เมื่อแปลงเป็นตัวเลขโดยแทน A = 01, B = 02, C = 03, ..., Z = 26. แล้วจะได้ตัวเลข 72 หลัก

110914070013151407112120000914192 09

202120050015060020050308141512150725

จากจำนวนหลักของ n เท่ากับ 44 หลักเราสามารถแบ่งข้อความที่เป็นตัวเลขได้ 2 บล็อก โดยบล็อกแรกใช้ 40 หลัก ส่วนบล็อกที่ 2 ใช้ 32 หลัก แล้วเข้ารหัสได้ดังนี้

ข้อความบล็อกที่ 1 M_1 : 1109140700131514071121

200009141920092021

ทำการเข้ารหัสได้ดังนี้ C_1 : 9380655288089560794796

743877454715306769400

ข้อความบล็อกที่ $2M_2$: 200500150600200503

เมื่อเข้ารหัสโดย $C_2 = M_2^e \bmod n$

จะได้ข้อความที่เข้ารหัส

C_2 : 367470296989927969361939

65894295337378579167

บทที่ 4

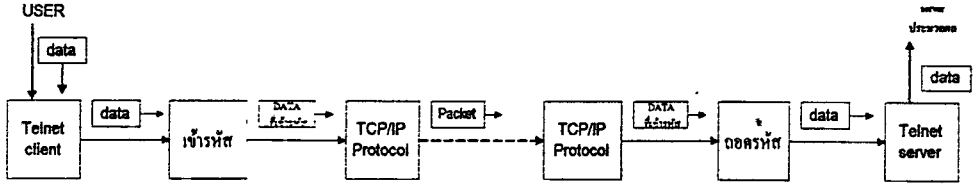
การออกแบบ

4.1 โครงสร้างและส่วนประกอบของโปรแกรม Secure Telnet

ก่อนที่เราจะทำการออกแบบโครงสร้างของโปรแกรม Secure Telnet เราจะมาดูโครงสร้างของเทลเนทธรรมดากันก่อนดังรูปที่ 4.1

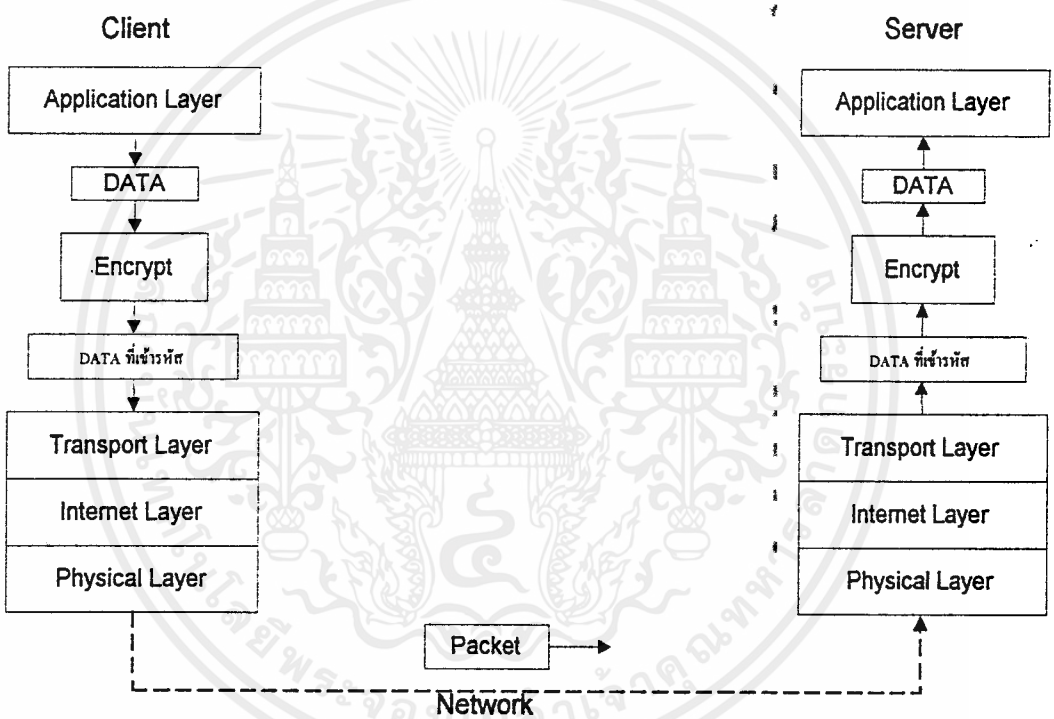
รูปที่ 4.1 โครงสร้างโดยรวมของการส่ง Data ของ Telnet ธรรมดา

จากรูปที่ 4.1 เป็นโครงสร้างที่แสดงถึงวิธีส่งข้อมูลจากเครื่องใช้บริการไปเครื่องให้บริการ ซึ่งเราจะเห็นว่าหลังจากข้อมูลผ่านกระบวนการ TCP/IP Protocol แล้ว จะได้เป็น Packet แบบ TCP/IP Protocol ส่งไปในเครือข่าย ซึ่งจากรูปจะดูเหมือนกับเครื่องใช้บริการกับเครื่องให้บริการต่อกันโดยตรงทั้ง 2 เครื่อง แต่ในความจริงแล้ว Packet จะต้องเดินทางผ่านเครื่องหลายเครื่องที่ไม่ใช่เครื่องให้บริการที่ต้องการติดต่อซึ่งถ้าเกิดมีผู้ใช้เครื่องที่ Packet ผ่านเหล่านั้น ต้องการจะดัก Packet ก็ย่อมจะทำได้และการถอด Packet ออกก็ทำได้ไม่ยากเช่นกัน เนื่องจาก โปรโตคอล TCP/IP เป็นโปรโตคอลที่เป็นมาตรฐาน ดังนั้น จึงทำให้ผู้ที่ดัก Packet ไปนั้นรู้ข้อมูลของผู้ใช้ได้ทันที ดังนั้น ในโครงงานนี้จึงทำการเข้ารหัสข้อมูลที่จะทำการส่งนี้เสีย จึงทำให้ Secure Telnet มีโครงสร้างใหม่ดังรูปที่ 4.2



รูปที่ 4.2 โครงสร้างของ Secure Telnet

จากรูปที่ 4.2 Secure Telnet นั้นจะนำข้อมูล จาก Telnet client ซึ่งอยู่ใน Application Layer ของโปรโตคอล TCP/IP จากนั้นจึงค่อยนำข้อมูลที่ได้เข้ารหัสนี้ส่งไปให้ชั้นต่อไปในโปรโตคอล TCP/IP ก็จะได้ Packet ส่งไปยังเครื่องให้บริการที่ต้องการติดต่อแล้วจากนั้น เมื่อ Packet ถึงปลายทางแล้ว ที่เครื่องให้บริการก็จะทำการแกะ Packet จนได้ข้อมูล ที่เข้ารหัสไว้แล้วก็จะทำการถอดรหัสแล้วส่งไปให้ Telnet process (Telnet server) เพื่อให้เครื่องให้บริการประมวลผลคำสั่งหรือข้อมูลที่ได้รับนั้น ซึ่งการส่งข้อมูลในชั้นต่าง ๆ ได้แสดงไว้ดังรูปที่ 4.3



รูปที่ 4.3 รูปแสดง Data ที่ Layer ต่างๆ และการเข้ารหัสใน Secure Telnet

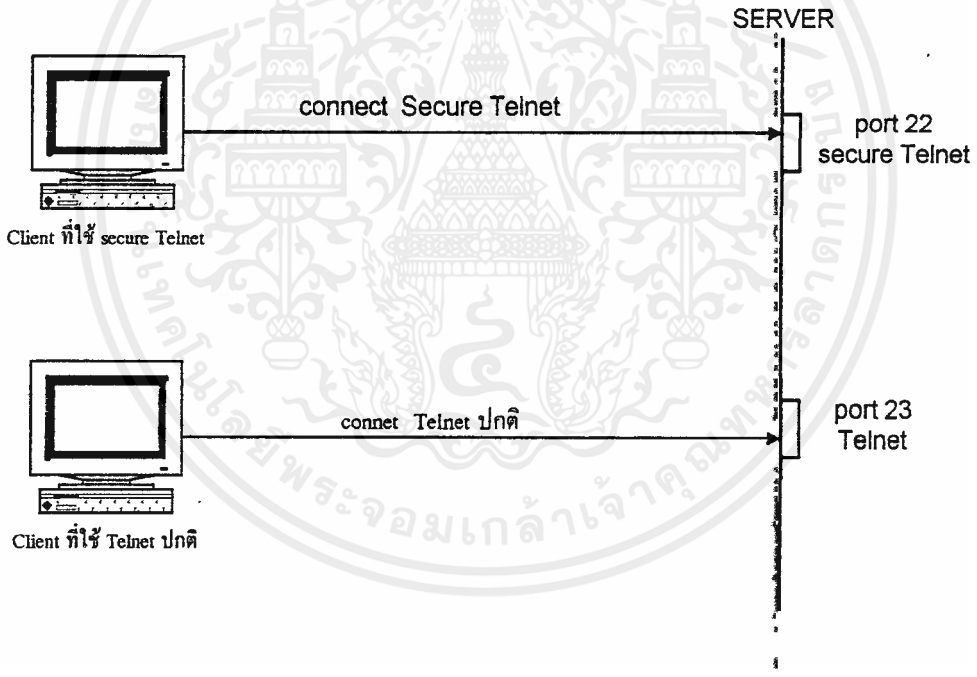
จากรูปที่ 4.2 และ 4.3 เป็นโครงสร้างโดยรวมของ Secure Telnet เรายังไม่ได้พูดถึงรายละเอียดของโปรแกรม Secure Telnet ที่เครื่องใช้บริการและเทลเนทที่เครื่องให้บริการกันเลย ในหัวข้อต่อไป

4.2 Telnet server (Telnet d)

ในการออกแบบ Telnet server นอกจากจะต้องออกแบบให้มีในส่วนการถอดรหัสข้อมูลแล้ว ยังมีสิ่งที่ต้องคำนึงถึงก็คือ Secure Telnet จะต้องรองรับการ login จาก Telnet ปกติได้อีกด้วย ซึ่งวิธีการที่ใช้เพื่อจะทำให้ Telnet server สามารถรองรับการ login จาก Telnet ปกติและ Secure Telnet ได้ มีวิธีดังต่อไปนี้

4.2.1 การแยก port บริการ

การแยก port บริการของ Secure Telnet กับเทลเนทปกติ เป็นวิธีหนึ่งที่ทำให้เครื่องให้บริการสามารถให้บริการได้ทั้งจากเทลเนทปกติและ Secure Telnet โดยในโครงงานนี้นั้นได้เลือก port เบอร์ 22 เป็น port ที่ใช้บริการ Secure Telnet ซึ่งแสดงไว้ดังรูปที่ 4.4



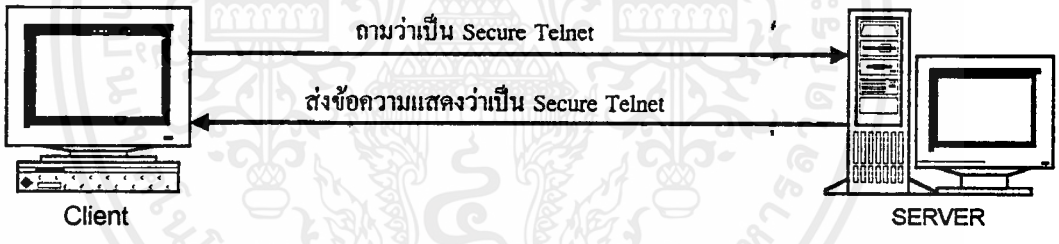
รูปที่ 4.4 แสดงการแยก port บริการของ Secure Telnet

การแยก port นี้ นอกจากจะทำให้ Telnet server สามารถรองรับการ login ได้ทั้งเทลเนทปกติและ Secure Telnet แล้ว ยังเป็นการช่วยเพิ่มความปลอดภัยของ Secure Telnet อีกด้วย เพราะถ้าผู้ดัก Packet ใช้โปรแกรมที่สร้างขึ้นมาเฉพาะ port บริการ Telnet (23) แล้วละก็ผู้ที่ดัก Packet ก็จะไม่เห็น Packet ที่ส่งกันด้วย Secure Telnet เลยทำให้ผู้ที่ดักิ Packet ไม่สามารถทดลองใส่ข้อมูลเพื่อวิเคราะห์วิธีการเข้ารหัสและคีย์ที่ใช้ในการเข้ารหัสได้ ซึ่งโปรแกรมที่ผู้บุกรุกระบบ

โดยใช้วิธีดัก Packet ก็นิยมดัก Packet เฉพาะ Port 23 เพราะทำให้ผู้บุกรุกระบบที่ดัก packet นั้น สามารถแยกแยะข้อมูลของผู้ใช้แต่ละคนได้โดยง่าย เพราะข้อมูลที่ได้จะเป็นข้อมูลของผู้ใช้ที่ใช้โปรแกรมเทลเนทอยู่เท่านั้น จึงทำให้การแยก port นี้ ทำให้ Secure Telnet มีความปลอดภัยสูงขึ้น

4.2.2 การแสดงความเป็น Secure Telnet ของ Telnet server

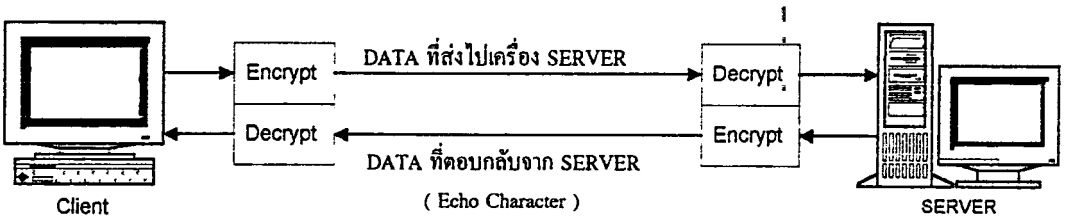
เนื่องจากโปรแกรม Secure Telnet นี้ได้ออกแบบให้รองรับการ login ด้วย Telnet ปกติได้ ดังนั้น จึงต้องมีการแสดงตัวว่าเป็น Secure Telnet ของ Telnet server เพื่อให้โปรแกรม Secure Telnet จากเครื่องใช้บริการทราบว่าเครื่องให้บริการนี้มีบริการ Secure Telnet หรือไม่ โดยการออกแบบได้ใช้ port บริการอีก port เพื่อเป็น port ที่ใช้แสดงตัวว่าเป็น Secure Telnet ของเครื่องให้บริการ โดยเริ่มต้นการติดต่อเครื่องใช้บริการจะติดต่อไปที่ port นี้ก่อน แล้วรอการตอบกลับจากเครื่องให้บริการ โดยถ้าเครื่องให้บริการเป็น Secure Telnet ก็จะส่งข้อความที่แสดงว่าเป็น Secure Telnet กลับไปให้เครื่องใช้บริการ เพื่อแสดงตัวว่าเป็น Secure Telnet ดังรูปที่ 4.5



รูปที่ 4.5 การแสดงตัวว่าเป็น Secure Telnet ของ Telnet server

4.2.3 การเปลี่ยนแปลงสถานะของการถอดรหัสข้อมูล

ในโครงงานนี้ได้ออกแบบการเข้ารหัสข้อมูลเฉพาะ Password ของผู้ใช้เท่านั้น จะไม่ทำการเข้ารหัสข้อมูลตลอดการใช้งาน เพราะการทำเช่นนั้น จะมีภาระในการเข้ารหัสและการถอดรหัสสูงทำให้การทำงานช้า เพราะการเข้ารหัสตลอดจนการใช้งานจะต้องทำการเข้ารหัสและถอดรหัส 2 ครั้ง คือ เข้ารหัสและถอดรหัสข้อมูลที่ส่งจากเครื่องใช้บริการไปยังเครื่องให้บริการ และเข้ารหัสถอดรหัสข้อมูลที่ส่งจากเครื่องให้บริการเพื่อตอบรับและส่ง Echo character กลับไปให้เครื่องใช้บริการ ซึ่งแสดงดังรูปที่ 4.6

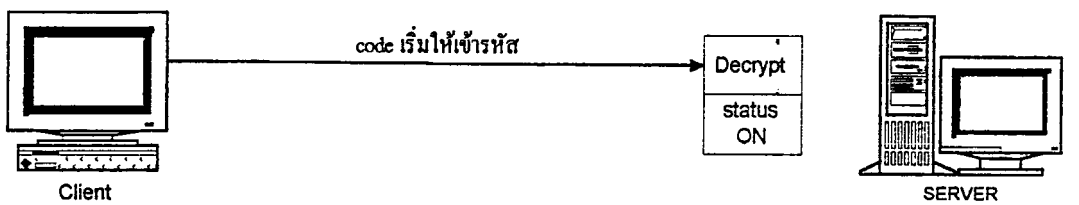


รูปที่ 4.6 แสดงการเข้ารหัสตลอดการใช้งาน

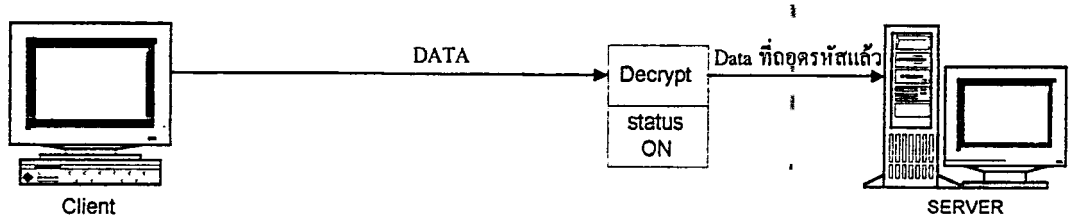
จากรูปที่ 4.6 เราจะเห็นว่าการเข้ารหัสตลอดการใช้งานนั้น มี Cost ในการเข้าและถอดรหัสสูง ดังนั้น เราจึงเลือกที่จะเข้ารหัสเฉพาะข้อมูลที่เป็น Password ของผู้ใช้เท่านั้น ซึ่งข้อมูลในส่วนนี้จะไม่มี Echo character เพื่อป้องกันไม่ให้บุคคลอื่นแอบดู Password จากจอภาพของผู้ใช้ ดังนั้น การเข้ารหัสและการถอดรหัสจึงทำเพียงส่วนที่เป็นการส่งข้อมูล Password จากเครื่องใช้บริการไปเครื่องให้บริการนั่นเอง ซึ่งแสดงดังรูปที่ 4.7

รูปที่ 4.7 แสดงการเข้ารหัสเฉพาะข้อมูลที่เป็น Password

ดังนั้น เมื่อเราไม่ได้ทำการเข้ารหัสตลอดการใช้งาน เราจึงต้องมีการเปลี่ยนสถานะโดยตกลง Code กันเอาไว้เพื่อบอกว่าข้อมูลไหนบ้างที่ Telnet server จะต้องถอดรหัสก่อนนำไปใช้ ซึ่งมีการทำงานดังรูปที่ 4.8

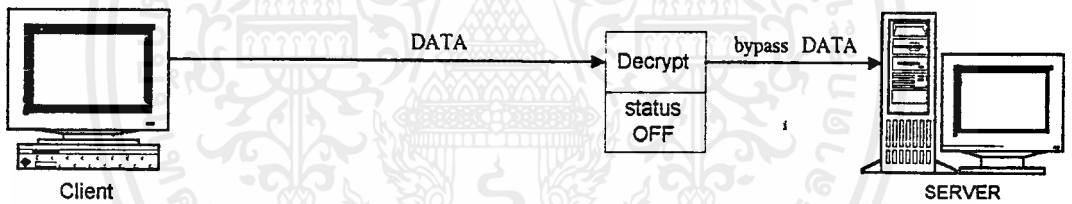


รูปที่ 4.8 (ก)



รูปที่ 4.8 (ข)

รูปที่ 4.8 (ค)



รูปที่ 4.8 (ง)

รูปที่ 4.8 (ก) เป็นรูปเมื่อ Telnet Client จะเข้ารหัสข้อมูลโดย Telnet client จะต้องส่ง Code บอกว่าข้อมูลที่จะส่งต่อไปเป็นข้อมูลที่เข้ารหัสให้ Telnet server ถอดรหัสด้วย โดย Status ของฟังก์ชั่นถอดรหัสจะเป็น ON

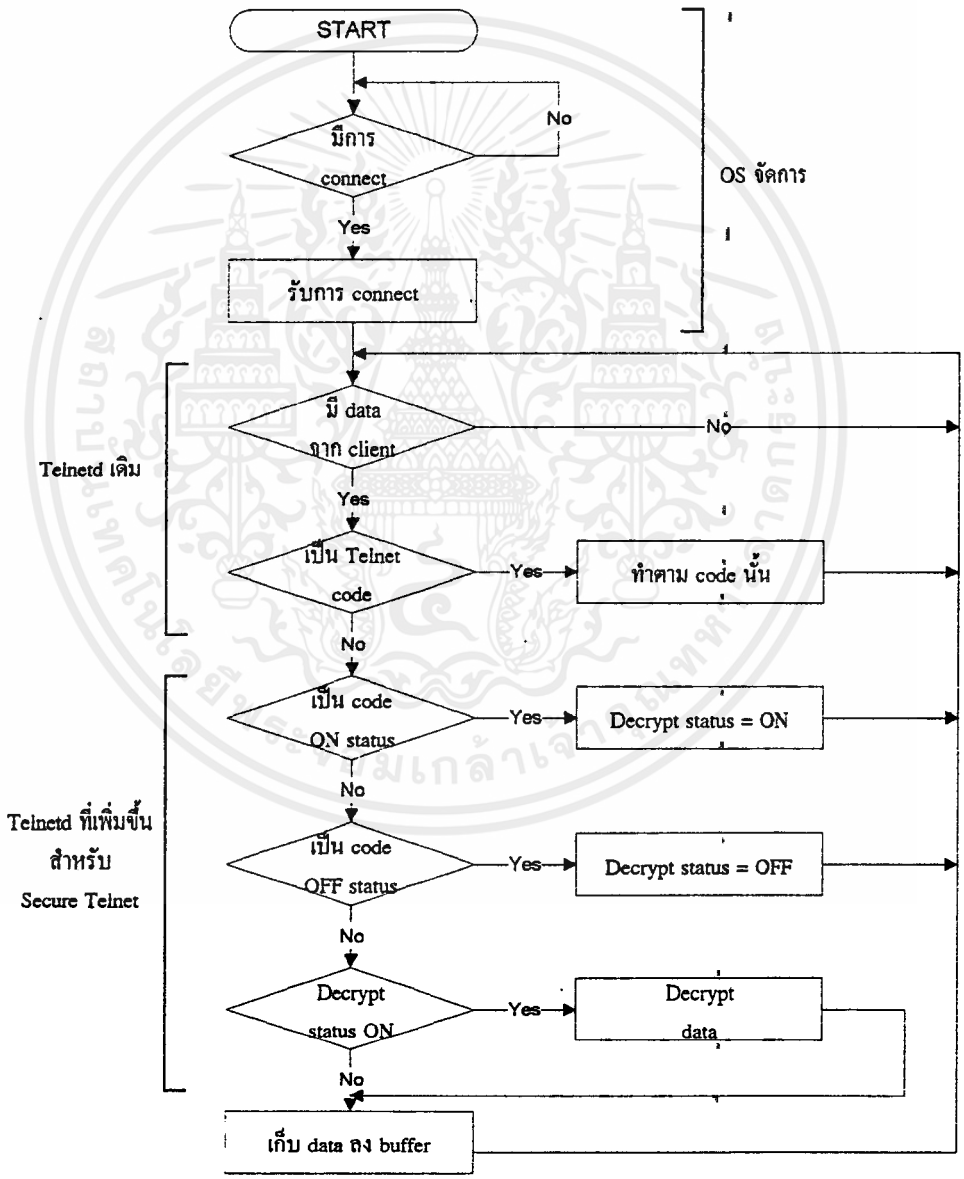
รูปที่ 4.8 (ข) เป็นรูปที่ Telnet client ส่งข้อมูลที่เข้ารหัสไปให้ Telnet server ฟังก์ชั่น ถอดรหัสที่ status เป็น ON อยู่ก็จะทำงานโดยจะถอดรหัสข้อมูล ก่อนที่จะนำไปประมวลผล

รูปที่ 4.8 (ค) เป็นรูปที่เมื่อ Telnet client ส่ง Data ที่เข้ารหัสเสร็จแล้ว จะเริ่มส่ง Data ที่เป็น Data ปกติก็จะส่ง Code ยกเลิกการเข้ารหัสข้อมูล เพื่อให้ Telnet server เลิกการถอดรหัสข้อมูล ซึ่งทำให้ฟังก์ชั่นถอดรหัสมี status เป็น off

รูปที่ 4.8 (ง) เป็นรูปแสดงการส่งข้อมูลจาก Telnet client มายัง Telnet server เมื่อฟังก์ชั่นถอดรหัสของ Telnet server เป็น off ซึ่ง Data ก็จะผ่านฟังก์ชั่น ถอดรหัสไปเลยโดยไม่มีการถอดรหัสข้อมูลแต่อย่างใด

4.2.4 การทำงานโดยรวมของ Telnet server ของ Secure Telnet

ในหัวข้อนี้จะนำหลักการของ Telnet server มารวมกัน โดยได้อธิบายเป็นแนวการทำงานดังรูปที่ 4.9



รูปที่ 4.9 แผนผังการทำงานของ Telnet server ใน Secure Telnet

4.3 Telnet client

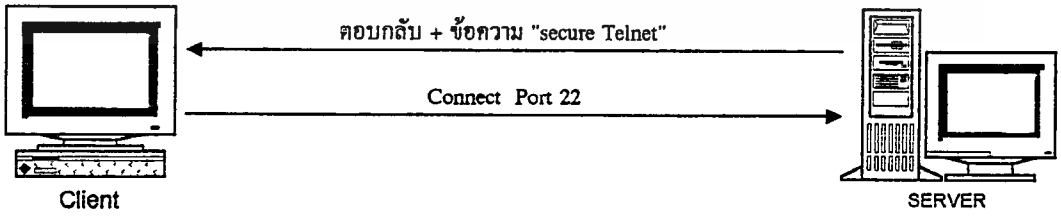
ในหัวข้อต่อไปนี้จะพูดถึงวิธีและการออกแบบ Telnet client เพื่อให้สามารถติดต่อขอ login กับเครื่องให้บริการที่มี Secure Telnet และไม่มีได้ และยังจะรวมไปถึงการออกแบบเพื่อให้ Telnet client ทำเทอร์มินอลเสมือนได้ด้วย

4.3.1 การตรวจสอบและติดต่อกับเครื่องให้บริการที่เป็น Secure Telnet

เริ่มจาก Telnet client จะติดต่อกับ port แสดงความเป็น Secure Telnet ของเครื่องให้บริการที่จะติดต่อด้วย ดังรูปที่ 4.10 (ก)

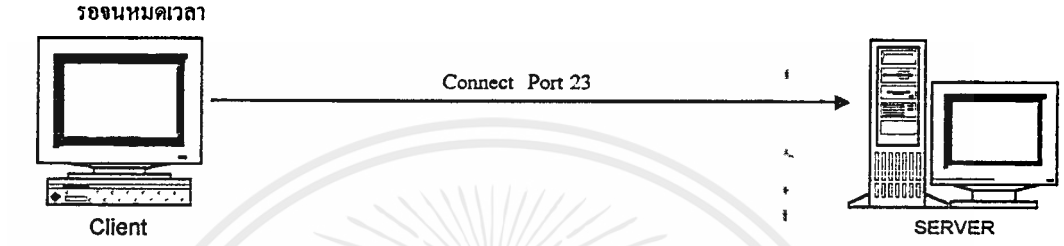
รูปที่ 4.10 (ก)

หลังจากนั้นเครื่องใช้บริการก็จะรอการตอบกลับจากเครื่องให้บริการ โดยถ้าตอบกลับมาภายในเวลาที่กำหนดและข้อมูลที่ตอบกลับมาเป็นข้อความแสดงความเป็น Secure Telnet ก็จะทำการสร้าง Connection กับ Port Secure Telnet (22) ต่อไปดังรูปที่ 4.10 (ข)

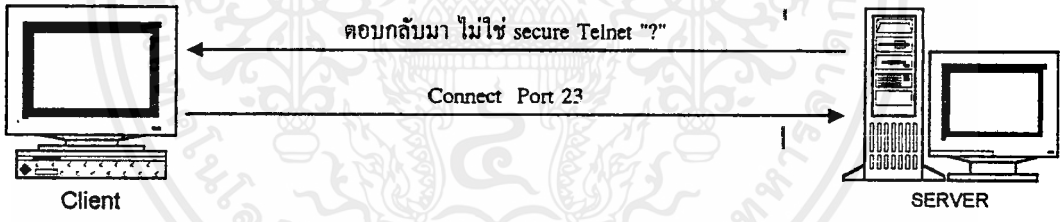


รูปที่ 4.10 (ข)

แต่ถ้าหมดเวลาแล้วเครื่องให้บริการไม่ตอบกลับมาหรือตอบกลับมาไม่ใช่ข้อความแสดงความเป็น Secure Telnet เครื่องใช้บริการก็จะสร้าง Connection กับ Port Telnet ปกติ (23) ดังรูปที่ 4.10 (ค) และ 4.10 (ง)



รูปที่ 4.10 (ค)



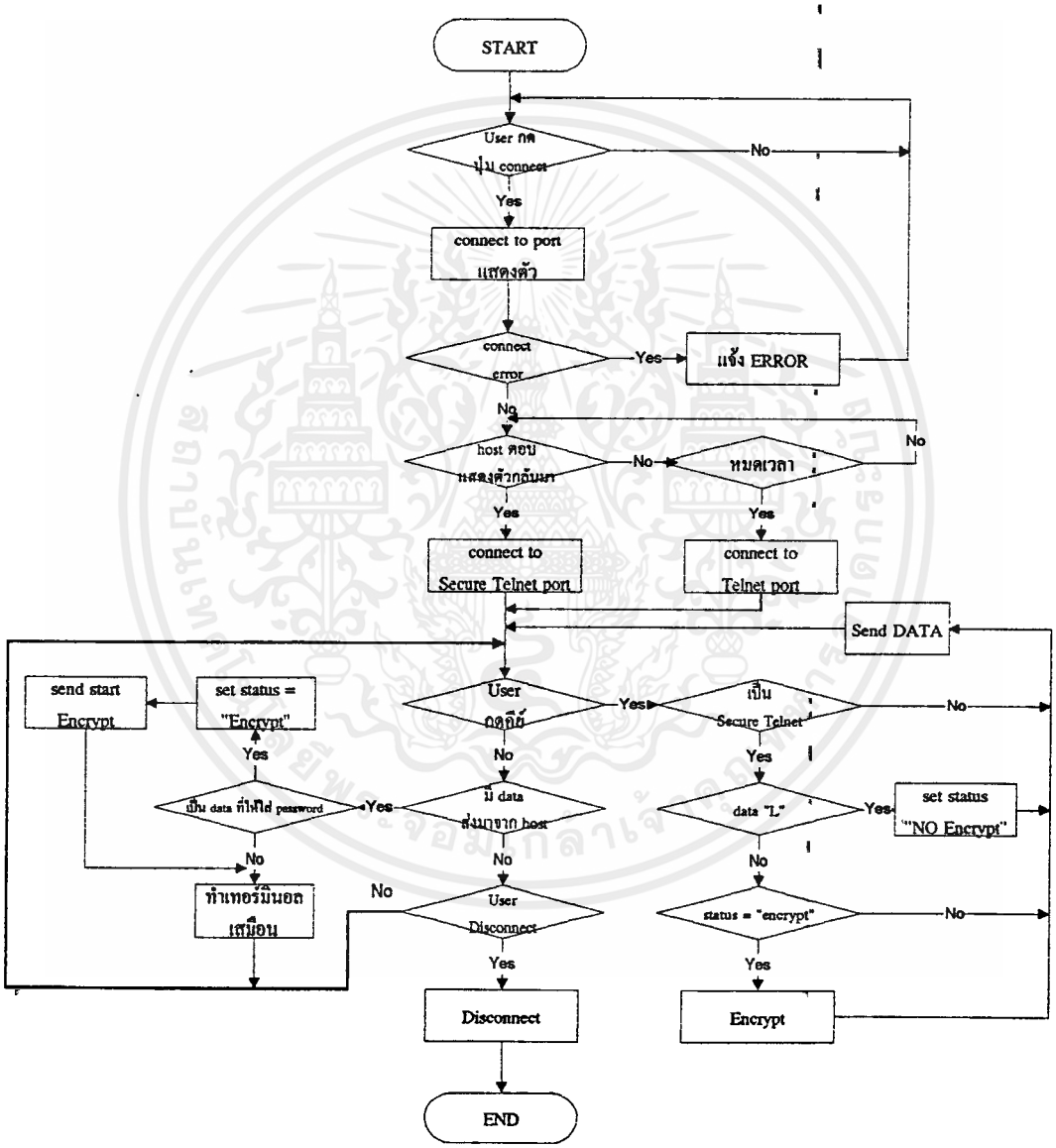
รูปที่ 4.10 (ง)

4.3.2 การแสดงผลโดยการทำเทอร์มินอลเสมือน

จากบทที่ 2 ในเรื่องการทำเทอร์มินอลเสมือนนั้นมีหลักการก็คือ เราจะต้องตีความ Code คำสั่งในการทำเทอร์มินอลเสมือนที่ส่งมาจากเครื่องให้บริการ แล้วทำมาทำให้เกิดผลบนจอของเครื่องใช้บริการซึ่งความหมายของ Code ต่าง ๆ ของเทอร์มินอลเสมือนมาตรฐาน VTIDO ซึ่งเป็นมาตรฐานที่ใช้ในโครงงานนี้ ได้แสดงไว้ในภาคผนวกด้านหลังแล้ว

4.3.3 การทำงานโดยรวมของ Telnet client ของ Secure Telnet

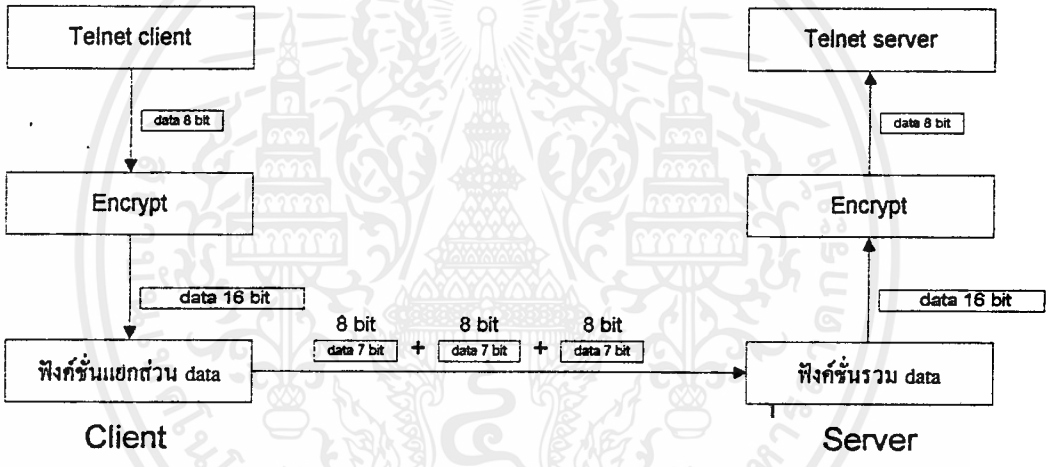
การทำงานของ Telnet client ของ Secure Telnet แสดงไว้เป็นแผนผังการทำงาน ดังรูปที่ 4.11



รูปที่ 4.11 แผนผังการทำงานของ Telnet client

4.4 การออกแบบส่วนการส่งข้อมูลที่เข้ารหัส

เนื่องจากคีย์ ที่ใช้ในการเข้ารหัสมีขนาด 16 บิต ซึ่งจะทำให้ข้อมูลขนาด 1 character หรือ 8 บิตนั้น เมื่อเข้ารหัสแล้ว จะมีขนาด 16 บิต ด้วยตามคีย์ ที่ใช้เข้ารหัส จึงทำให้เราต้องส่งข้อมูลอย่างน้อย 2 ไบต์ แต่ในความเป็นจริงแล้วเราไม่สามารถส่งข้อมูล 16 บิตนี้ได้โดยการส่งข้อมูลไป 2 ไบต์ ได้เพราะในข้อมูล 1 ไบต์นี้ เราไม่สามารถใได้ทั้งช่วง (0-255) เพราะในช่วงข้อมูล 0-255 นี้ มีส่วนที่เป็น code ของเทลเนท และ Command ของเทลเนท อยู่ ซึ่งได้แสดงไว้ในภาคผนวกด้านหลังแล้ว ดังนั้น เพื่อตัดปัญหานี้ เราจึงต้องแบ่งข้อมูล 16 บิตนี้ ออกเป็น 7+7+2 บิต คือ เราต้องส่งข้อมูล 3 ครั้ง ครั้งละไบต์ ดังรูปที่ 4.12



รูปที่ 4.12 แสดงการส่งข้อมูลที่เข้ารหัส

จากรูปที่ 4.12 เมื่อ Telnet client ส่งข้อมูลขนาด 8 บิตไปเข้ารหัสการจะได้ข้อมูลขนาด 16 บิต นำไปเข้าฟังก์ชั่นแยกส่วนของข้อมูล 16 บิตนั้น เพื่อส่งไปใน Network ทีละไบต์ ทั้งหมดก็เป็น 3 ไบต์ ต่อข้อมูล 1 ตัวอักษร เมื่อข้อมูลทั้ง 3 ไบต์ส่งไปถึงฟังก์ชั่นรวม ข้อมูลแล้วก็จะได้ข้อมูล 16 บิต ซึ่งยังเป็นข้อมูลที่เข้ารหัสอยู่ ดังนั้น จึงนำไปผ่านฟังก์ชั่นถอดรหัส ซึ่งก็จะได้ข้อมูล 8 บิต ที่เป็นข้อมูลจริง ๆ นำไปแปลความหมายด้วย Telnet server ต่อไป

บทที่ 5

การทดลองและผลการทดลอง

5.1 การทดสอบโปรแกรมในส่วนของ Telnet server

ในหัวข้อนี้จะพูดถึงการทดสอบฟังก์ชั่นต่างๆ ของ Telnet server

5.1.1 การทดสอบส่วนของการแสดงตัวว่าเป็น Secure Telnet

การทดสอบทำได้โดยทดลองติดต่อไปที่ port แสดงตัวของ Telnet server แล้วรอแสดงผลข้อมูลที่ตอบกลับมาดูว่าถูกต้องหรือไม่ ซึ่งมีผลการทดสอบ ดังนี้

Telnet AMIGA 26 ← ติดต่อกับ port แสดงตัว

Secure Telnet ← ข้อมูลแสดงตัวของ Telnet server

5.1.2 การทดสอบส่วนของการถอดรหัสของ Telnet server

การทดสอบทำโดยการให้เทลเนท ปกติติดต่อไปที่ Server Telnet ที่เครื่องให้บริการ แล้วลองพิมพ์ข้อมูลที่ทราบว่าเมื่อผ่านฟังก์ชั่นถอดรหัสแล้วจะได้ข้อมูลอะไรดูโดยใช้คีย์ขนาดเล็ก ๆ ทดสอบดู ซึ่งได้ผลดังนี้

จากเราทราบว่าข้อมูล “Test” เมื่อผ่านฟังก์ชั่นถอดรหัสแล้วจะได้

T → r

e → t

s → 0

t → 3

ดังนั้น เมื่อเราพิมพ์ “Test” ส่งไปให้เครื่องให้บริการ Echo character จะต้องส่งกลับมาเป็น “r t 0 3” การทำงานของฟังก์ชั่นก์ถอดรหัสที่ Telnet server จึงจะทำงานถูกต้อง

$ Telnet AMIGA 22

login r t 0 3 ← พิมพ์ Test

แสดงว่า ฟังก์ชั่นถอดรหัสทำงานถูกต้อง

5.2 การทดสอบโปรแกรมในส่วนของ Telnet client

ในหัวข้อนี้จะพูดถึงการทดสอบฟังก์ชั่นการทำงานต่าง ๆ ของ Telnet client

5.2.1 การทดสอบส่วนของฟังก์ชั่นเข้ารหัสของ Telnet client

การทดสอบทำได้โดยใช้ Telnet client นี้ ติดต่อไปที่ port ของเทลเนทปกติ (23) แล้วลองพิมพ์ข้อมูลที่เราทราบว่าเมื่อเข้ารหัสแล้ว จะได้ข้อมูลอะไรเข้าไป แล้วดู Echo character ที่ส่งกลับมาว่าเป็นข้อมูลอะไร ตรงกับข้อมูลที่ทราบหรือไม่ถ้าตรงก็แสดงว่าถูกต้อง

จากเราทราบว่าข้อมูล "Test" เมื่อเข้ารหัสแล้วจะได้

T → f

e →)

s → l

t → ä

ดังนั้น เมื่อเราพิมพ์คำว่า "Test" เข้าไปผลบนจอจะต้องได้ ถ้าไม่ได้แสดงว่าฟังก์ชั่นเข้ารหัสทำงานผิดพลาด

กดปุ่ม connect

login : "f)lä" ← พิมพ์ "Test"

แสดงว่าการทำงานของฟังก์ชั่นเข้ารหัสถูกต้อง

5.2.2 ทดสอบการตรวจสอบเครื่องให้บริการของ Telnet client

เป็นการทดสอบว่า Telnet client สามารถ login กับเครื่องให้บริการที่มี Secure Telnet และไม่มี Secure Telnet ได้หรือไม่

5.2.2.1 วิธีทดสอบว่า login กับเครื่องให้บริการที่มี Secure Telnet

ทำได้โดยลอง login เข้าไปที่เครื่องให้บริการที่มี Secure Telnet ดูและสังเกตผลว่าติดต่อกับ port อะไร ใช่ Port Secure Telnet (22) หรือไม่ และ status การเข้ารหัสของโปรแกรม Telnet client เป็นอย่างไร และเข้าใช้ระบบได้หรือไม่ ซึ่งได้ผลดังนี้

Host = AMIGA กดปุ่ม connect

Secure Telnet ← การแสดงตัวของ Host

(สถานะของ Telnet Client remote port = 22)

login : Test (status = NO Encrypt)

password : xxxx ← ใส่ password (status = Encrypt)

S _ เข้า shell ได้แล้ว แสดงว่า login ถูกต้อง

5.2.2.2 วิธีการทดสอบว่า login กับเครื่องให้บริการที่ไม่มี Secure Telnet

ทำโดยลอง login เข้าไปที่เครื่องให้บริการที่ไม่มี Secure Telnet ดู ถ้า login เข้าไปได้ แสดงว่า ทำงานได้ถูกต้อง

Host = DIAMOND กดปุ่ม Connect

(รอการแสดงตัว 5 วินาที)

(สถานะของ Telnet Client remote port เปลี่ยนเป็น 23 ; Telnet ปกติ)

login : Test (status = NO Encrypt)

password : xxxx (status = NO Encrypt)

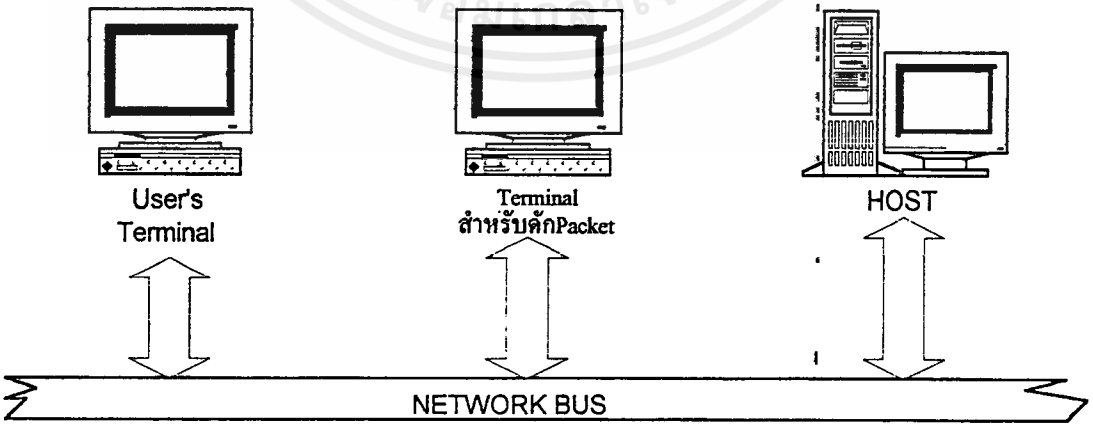
$ _ เข้า shell ได้แล้ว แสดงว่า login ถูกต้อง

5.3 การทดสอบความปลอดภัยของโปรแกรม Secure Telnet

ในหัวข้อนี้จะกล่าวถึงการเตรียมอุปกรณ์และวิธีทดสอบว่า Secure Telnet มีความปลอดภัยจริง

5.3.1 การเตรียมอุปกรณ์สำหรับทดสอบ

การทดสอบให้เชื่อมต่ออุปกรณ์ ดังรูป



รูปที่ 5.1 การเชื่อมต่ออุปกรณ์ที่ใช้ในการทดสอบ

ส่วนโปรแกรมที่จะรันในแต่ละเครื่องมีดังนี้

เครื่องให้บริการ (Host) ติดตั้งโปรแกรม Secure Telnet server เรียบร้อยแล้ว

เครื่องใช้บริการ (User's Terminal) ติดตั้งโปรแกรม Secure Telnet client เรียบร้อยแล้ว

เครื่อง Terminal สำหรับดัก Packet จะรันโปรแกรมสำหรับดัก Packet อยู่

5.3.2 ขั้นตอนทดสอบ

เราจะทดสอบโดยดัก Packet เมื่อใช้โปรแกรม Secure Telnet กับเทลเนทปกติมาเทียบกันดู โดยการใช้ Telnet ปกติ login เข้าไปใช้ระบบก่อนแล้วดูผลจากโปรแกรมดัก Packet แล้วจากนั้นก็ทดลองใช้โปรแกรม Secure Telnet ลอง login เข้าไปใช้ระบบเช่นกันแล้วดูผลจากโปรแกรมดัก Packet ซึ่งได้ผลดังต่อไปนี้

Host คือ 161.246.2.232 (AMIGA)

Client คือ 161.246.2.12

ผลของการดัก Packet จากการใช้ **เทลเนทปกติ**

161.246.2.232	→	161.246.2.12	" login : "
161.246.2.12	→	161.246.2.232	"T"
161.246.2.232	→	161.246.2.12	ACK , "T"
161.246.2.12	→	161.246.2.232	"e"
161.246.2.232	→	161.246.2.12	ACK , "e"
161.246.2.12	→	161.246.2.232	"s"
161.246.2.232	→	161.246.2.12	ACK , "s"
161.246.2.12	→	161.246.2.232	"t"
161.246.2.232	→	161.246.2.12	ACK , "t"
161.246.2.12	→	161.246.2.232	ch(13),ch(10)
161.246.2.232	→	161.246.2.12	ACK , ch(13) , ch(10)
161.246.2.12	→	161.246.2.232	"password : "
161.246.2.232	→	161.246.2.12	"T"
161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	"e"

161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	“s”
161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	“t”
161.246.2.12	→	161.246.2.232	ACK

จะพบว่า เห็น Password ว่าเป็น “Test”

ผลของการทดลองดัก Packet จากการใช้ **Secure Telnet**

161.246.2.232	→	161.246.2.12	“ login : “
161.246.2.12	→	161.246.2.232	“T”
161.246.2.232	→	161.246.2.12	ACK , “T”
161.246.2.12	→	161.246.2.232	“e”
161.246.2.232	→	161.246.2.12	ACK , “e”
161.246.2.12	→	161.246.2.232	“s”
161.246.2.232	→	161.246.2.12	ACK , “s”
161.246.2.12	→	161.246.2.232	“t”
161.246.2.232	→	161.246.2.12	ACK , “t”
161.246.2.12	→	161.246.2.232	ch(13),ch(10)
161.246.2.232	→	161.246.2.12	ACK , ch(13) , ch(10)
161.246.2.12	→	161.246.2.232	“password : “
161.246.2.232	→	161.246.2.12	“f” , ”t” , ”ä“
161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	“l” , “x” , “;”
161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	“+” , “n” , “┘“
161.246.2.12	→	161.246.2.232	ACK
161.246.2.232	→	161.246.2.12	“-” , ““” , “g”
161.246.2.12	→	161.246.2.232	ACK

จะพบว่าอ่าน password ไม่รู้เรื่องเลย

สรุปผลการทดสอบ จากผลการทดสอบจะเห็นว่า Telnet ปกติเมื่อผู้ใช้ใส่รหัสผ่าน (Password) จะเห็นในโปรแกรมดัก Packet แต่เมื่อใช้โปรแกรม Secure Telnet จากผลจะเห็นว่า ข้อมูลรหัสผ่านถูกเข้ารหัสโดย 1 ตัวอักษรจะได้ข้อมูลที่เข้ารหัสแล้วขนาด 3 ไบต์ และข้อมูลที่ได้อ่านไม่รู้เรื่องเลย หรือก็คือ แตกต่างไปจากข้อมูล Password เดิมอย่างสิ้นเชิง ซึ่งผลการทดสอบแสดงให้เห็นว่า โปรแกรม Secure Telnet มีความปลอดภัยในการใช้งานจริง

บทที่ 6
บทวิจารณ์และสรุป

6.1 บทวิจารณ์

ผลจากที่เรานำข้อมูลรหัสผ่านของผู้ใช้ไปเข้ารหัส ก็ทำให้ความปลอดภัยของการใช้งานโปรแกรมเทลเนทมีสูงขึ้นมาก ซึ่งเป็นไปตามวัตถุประสงค์ที่วางไว้ แต่โปรแกรม Secure Telnet ในโครงงานนี้ ก็ยังต้องมีสิ่งที่ต้องปรับปรุงเพื่อให้โปรแกรมมีความสมบูรณ์ยิ่งขึ้นอีกมาก เช่น ในเรื่องของความปลอดภัย แม้ว่าโปรแกรมจะให้ความปลอดภัยที่สูงขึ้น แต่ก็ยังไม่สูงเกินกว่าความพยายามของผู้บุกรุก เนื่องจาก ขนาดของคีย์ที่ใช้เข้ารหัสที่เล็กเกินไป หรือการที่ใช้คีย์ในการเข้ารหัสเพียงคีย์เดียวไม่เปลี่ยนแปลง และนอกจากนี้ในเรื่องความสะดวกของผู้ใช้โปรแกรมยังต้องปรับปรุงอีกมาก เช่น ควรมีการทำเทอร์มินอลเสมือนได้หลากหลายกว่านี้ หรือความสมบูรณ์ในการทำเทอร์มินอลเสมือน รวมไปถึงการเก็บชื่อเครื่องให้บริการที่ผู้ใช้เคยไปมาแล้ว เพื่อความสะดวกในการใช้งาน เป็นต้น

ในหัวข้อต่อไปจะกล่าวถึงแนวทางการปรับปรุงโครงงานให้ดีขึ้นมีความปลอดภัยและความสะดวกในการใช้งานให้ดีขึ้น และสามารถนำโครงงานนี้ไปใช้ได้จริง

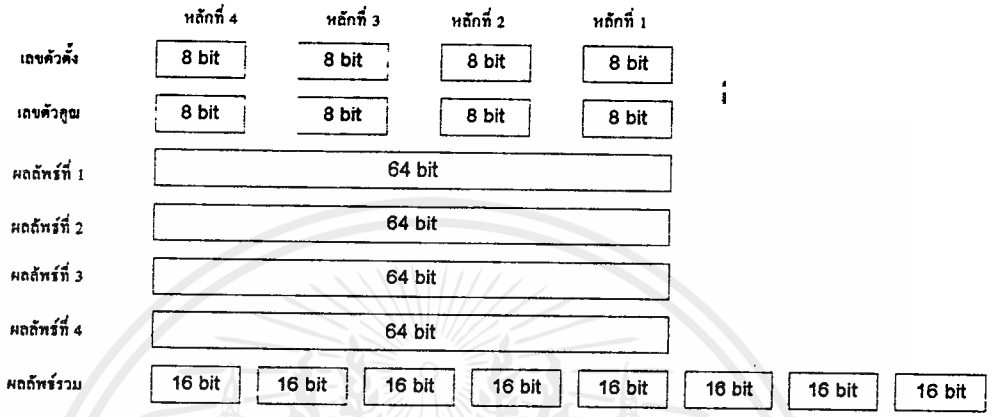
6.2 แนวทางการปรับปรุงโครงงาน

ต่อไปนี้จะกล่าวถึงแนวทางและวิธีปรับปรุงโครงงานให้ดีและมีความสมบูรณ์ยิ่งขึ้น ซึ่งมีแนวทาง ดังนี้

6.2.1 การปรับปรุงขนาดของคีย์ในการเข้ารหัส

จากทฤษฎีในการเข้ารหัสเราทราบว่าถ้าคีย์ในการเข้ารหัสมีขนาดใหญ่ขึ้นเท่าไร ความปลอดภัยของข้อมูลที่เข้ารหัสก็สูงขึ้น ซึ่งในทางปฏิบัติแล้ว คีย์ของการเข้ารหัสแบบ RSA ควรจะมีขนาดไม่ต่ำกว่า 48 บิต แต่ในโครงงานนี้ใช้เพียง 16 บิต ซึ่งสาเหตุสำคัญที่ใช้เพียง 16 บิต เพราะตัวแปรในภาษา C และ VB4 มีขนาดสูงสุดเพียง 64 บิต ซึ่งไม่พอเพียงที่จะรองรับผลของเลข 32 บิตกำลัง 2 หลายๆ ครั้งได้ ดังนั้น ในโครงงานจึงเลือกใช้คีย์ในการเข้ารหัสคีย์เพียง 16 บิต เพราะมีตัวแปรขนาด 64 บิต รองรับผลลัพธ์ได้นั่นเอง

แนวทางที่จะพัฒนาให้คีย์ที่ใช้ในการเข้ารหัสมีขนาดใหญ่กว่านี้ เราจะต้องใช้หลักการแยกคูณ และแยกหารหาเศษโดยจะต้องมีการเตรียมพื้นที่และตัวแปรที่ใช้ในการแยกคูณทีละส่วนดังรูปที่ 6.1



รูปที่ 6.1 แสดงโครงสร้างตัวแปรที่ใช้ในการคูณแยกส่วน 32 บิต

รูปที่ 6.1 แสดงการคูณเลขขนาด 32 บิตกับ 32 บิต โดยวิธีการคูณแยกส่วน โดยแบ่งเลขตัวตั้งออกเป็น 8 บิต 4 ชุด และแบ่งตัวคูณออกเป็น 8 บิต 4 ชุดเช่นกัน ส่วนผลลัพธ์ก็จะมีผลลัพธ์ที่ 1-4 เป็นขนาด 64 บิต และผลลัพธ์รวมแยกส่วนเป็น 16 บิต 8 ชุด ดังรูปที่ 6.1

หลักการคูณแยกส่วนมีวิธีดังนี้ คือ

1. นำตัวคูณหลักที่ 1 คูณตัวตั้งทั้ง 4 หลัก เก็บผลลัพธ์ไว้ในผลลัพธ์ที่ 1
2. ทำเช่นเดียวกับข้อ 1 แต่ใช้ตัวคูณหลักที่ 2, 3, 4 และเก็บผลลัพธ์ไว้ในผลลัพธ์ที่ 2, 3, 4 ตามลำดับ
3. นำผลลัพธ์ทั้ง 4 บวกกันโดยผลลัพธ์ที่ 2 เลื่อนไปทางซ้าย 8 บิต ผลลัพธ์ที่ 3 เลื่อนไป 16 บิต และผลลัพธ์ที่ 4 เลื่อนไป 24 บิต นำผลบวกที่ได้แยกเก็บในผลลัพธ์รวม ดังตัวอย่างในรูปที่ 6.2

เลขตัวตั้ง **F5 B3 A2 C7**

เลขตัวคูณ **F5 B3 A2 C7**

ผลลัพธ์ที่ 1 F5 B3 A2 C7 * C7 = 00 00 00 BE FE A3 88 B1

ผลลัพธ์ที่ 2 F5 B3 A2 C7 * A2 = 00 00 00 9B 7B AD 01 EE

ผลลัพธ์ที่ 3 F5 B3 A2 C7 * B3 = 00 00 00 AB CC 9A D1 25

ผลลัพธ์ที่ 4 F5 B3 A2 C7 * F5 = 00 00 00 EB 24 EA C8 73

BE FE A3 88 B1	ผลลัพธ์ที่ 1
BE FE A3 88 B1	ผลลัพธ์ที่ 2
AB CC 9A D1 25	ผลลัพธ์ที่ 3
EB 24 EA C8 73	ผลลัพธ์ที่ 4
EB D1 53 9D EF CA 76 B1	ผลลัพธ์รวม

$\therefore (F5\ B3\ A2\ C7)^2 = 0000\ 0000\ 0000\ 0000\ EBD1\ 539D\ EFCA\ 76B1$

รูปที่ 6.2 แสดงการหาค่าของ $(F5B3A2C7)^2$

จากนั้นเมื่อเราได้ผลลัพธ์ของการคูณแยกส่วนแล้วเราก็นำไปหารแยกส่วนเมื่อ Mod หาเศษดังรูปที่ 6.3

ผลลัพธ์จากการคูณ = 0000 0000 0000 0000 EBD1 539D EFCA 76B1

เลขที่จะนำมา MOD = A001

```
      1 744B 9083 CBA3
A001 EBD1 539D EFCA 76B1
      A001
      4BD0 5
      4600 7
       5CF E3
       5A0 09
```

2F 3A9

28 004

7 3A5D

6 E00B

5A52 E

5A00 9

52 5FC

50 008

2 5F4A

1 E003

7F47 7

7800 C

746 86

6E0 0B

66 7BB

64 00A

2 7B11

1 E003

ผลลัพธ์ของ $(F5B3A2C7)^2 \bmod A001 = 9B0E$

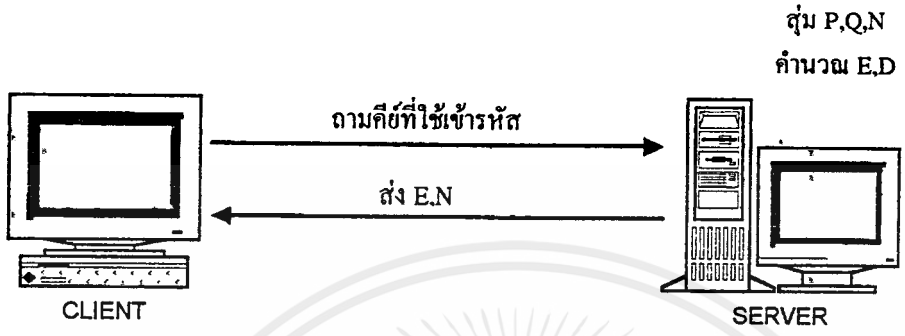
รูปที่ 6.3 การหารแยกส่วน

จากวิธีคูณแยกส่วนและหารแยกส่วนดังกล่าว จึงทำให้เราสามารถที่จะเพิ่มขนาดคีย์ที่ใช้เข้ารหัสจาก 16 บิต เป็น 64 บิตได้เลย ซึ่งเมื่อได้คีย์ขนาด 64 บิตแล้วก็จะทำให้ Secure Telnet มีความปลอดภัยเพิ่มขึ้นได้อีกหลายเท่าตัวเลยทีเดียว

6.2.2 การปรับปรุงให้การเข้ารหัสใช้คีย์มากกว่าหนึ่งคีย์

การทำให้การเข้ารหัสเปลี่ยนคีย์ที่ใช้เข้ารหัสได้ทำให้ความปลอดภัยมีสูงขึ้นอีกมาก ซึ่งการเปลี่ยนแปลงคีย์ที่ใช้เข้ารหัสนี้ อาจทำโดยสุ่มคีย์ขึ้นมาเลย แต่วิธีนี้ อาจจะช้าเพราะการ

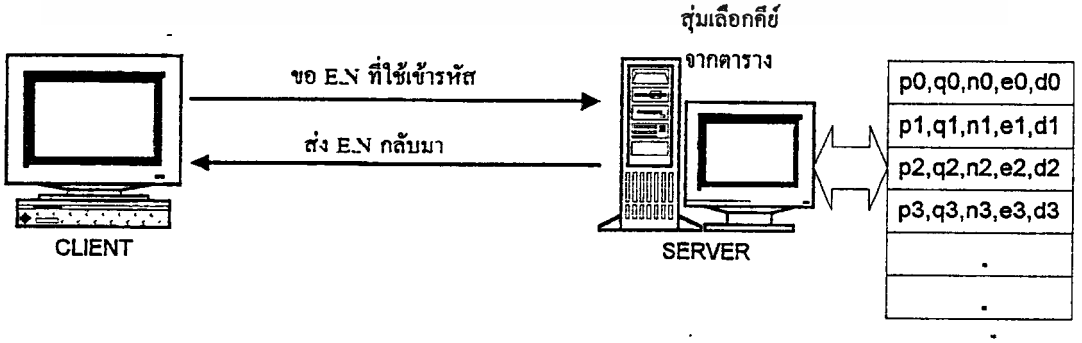
สุ่มเลขจำนวนเฉพาะขนาดใหญ่ 2 จำนวน นั้นใช้เวลามาก แต่วิธีนี้ก็จะให้ความปลอดภัยสูงสุด เพราะแม้แต่ผู้สร้างโปรแกรมเองก็ยังไม่ทราบว่าคีย์ใช้เข้ารหัสคืออะไร



รูปที่ 6.4 การถาม key เข้ารหัสแบบสุ่ม

จากรูปที่ 6.4 เครื่องใช้บริการเมื่อเริ่มติดต่อจะถามคีย์ที่ใช้เข้ารหัสจากเครื่องให้บริการ เครื่องให้บริการก็จะทำการสุ่ม p, q, n และคำนวณหาค่า e, d แล้วส่งคีย์ที่ใช้เข้ารหัส (e, n) กลับไปให้เครื่องใช้บริการเพื่อให้นำไปใช้ในการเข้ารหัสต่อไป

แต่การที่เราจะให้เครื่องให้บริการสุ่มและคำนวนหาคีย์ที่ใช้เข้ารหัสและถอดรหัสแบบนี้ทุกครั้ง จะเป็นการเสียเวลาในการเริ่มติดต่อมาโดยเฉพาะถ้าคีย์ที่มีขนาดใหญ่ๆ ดังนั้น เราอาจสร้างคีย์เข้าและถอดรหัสเอาไว้ก่อนแล้วเก็บไว้ในตาราง เมื่อมีการติดต่อก็ให้เครื่องบริการไปสุ่มเลือกคีย์เข้าและถอดรหัสออกมาจากตารางนั้น แล้วส่งไปให้เครื่องใช้บริการ ซึ่งวิธีนี้นอกจากจะเพิ่มความเร็วในการติดต่อแล้ว ความปลอดภัยก็ลดลงไม่มาก ถ้าเรามีจำนวนคีย์ที่มากในตาราง แสดงไว้ดังรูปที่ 6.5



รูปที่ 6.5 การถาม key โดยเลือกจากตาราง

6.3 บทสรุป

บทสรุป โครงงานนี้เป็นตัวอย่างการพัฒนาความปลอดภัยของโปรแกรมประยุกต์บนระบบอินเตอร์เน็ตโปรแกรมหนึ่งซึ่งก็คือโปรแกรมเทลเนท ซึ่งโปรแกรมเทลเนทนี้เป็นโปรแกรมหนึ่งที่เกี่ยวข้องกับข้อมูลที่เป็นความลับของผู้ใช้ ซึ่งจากผลการนำข้อมูลเหล่านี้ไปเข้ารหัสแล้ว ก็ทำให้โปรแกรมเทลเนท ที่พัฒนาโดยโครงงานนี้ มีความปลอดภัยขึ้น จึงทำให้การใช้อินเตอร์เนทยังมีโปรแกรมประยุกต์อื่นๆ ที่เกี่ยวข้องกับข้อมูลที่เป็นความลับของผู้ใช้อีก เช่น FTP ซึ่งพวกกระผมหวังว่าโครงงานนี้คงเป็นแนวทางให้ผู้ที่ต้องการพัฒนาความปลอดภัยในระบบอินเตอร์เนทนำไปใช้พัฒนาความปลอดภัยของโปรแกรมอื่นๆ ในระบบอินเตอร์เนทเพื่อให้การใช้งานอินเตอร์เนทมีความปลอดภัยขึ้นมากกว่าในปัจจุบัน

ภาคผนวก ก

CODE of VT100 Virtual Terminal

ANSI/VT100 Terminal Control

Many computer terminals and terminal emulators support color and cursor control through a system of escape sequences. One such standard is commonly referred to as ANSI Color. Several terminal specifications are based on the ANSI color standard, including VT100.

The following is a partial listing of the VT100 control set.

`<ESC>` represents the ANSI "escape" character, 0x1B. Bracketed tags represent modifiable decimal parameters; eg. `{ROW}` would be replaced by a row number.

Device Status

The following codes are used for reporting terminal/display settings, and vary depending on the implementation:

`Query Device Code` `<ESC>[c`

Requests a `Report Device Code` response from the device.

`Report Device Code` `<ESC>[{code}0c`

Generated by the device in response to `Query Device Code` request.

`Query Device Status` `<ESC>[5n`

Requests a `Report Device Status` response from the device.

`Report Device OK` `<ESC>[0n`

Generated by the device in response to a `Query Device Status` request; indicates that device is functioning correctly.

`Report Device Failure` `<ESC>[3n`

Generated by the device in response to a `Query Device Status` request; indicates that device is functioning improperly.

`Query Cursor Position` `<ESC>[6n`

Requests a `Report Cursor Position` response from the device.

`Report Cursor Position` `<ESC>[{ROW};{COLUMN}R`

Generated by the device in response to a `Query Cursor Position` request; reports current cursor position.

Terminal Setup

The `h` and `l` codes are used for setting terminal/display mode, and vary depending on the implementation. Line Wrap is one of the few setup codes that tend to be used consistently:

`Reset Device` `<ESC>c`

Reset all terminal settings to default.

`Enable Line Wrap` `<ESC>[7h`

Text wraps to next line if longer than the length of the display area.

`Disable Line Wrap` `<ESC>[7l`

Disables line wrapping.

Fonts

Some terminals support multiple fonts: normal/bold, swiss/italic, etc. There are a variety of special codes for certain terminals; the following are fairly standard:

`Font Set G0` `<ESC>(`

Set default font.

`Font Set G1` `<ESC>)`

Set alternate font.

Cursor Control

`Cursor Home` `<ESC>[{ROW};{COLUMN}H`

Sets the cursor position where subsequent text will begin. If no row/column parameters are provided (ie. `<ESC>[H`), the cursor will move to the *home* position, at the upper left of the screen.

`Cursor Up` `<ESC>[{COUNT}A`

Moves the cursor up by *COUNT* rows; the default count is 1.

`Cursor Down` `<ESC>[{COUNT}B`

Moves the cursor down by *COUNT* rows; the default count is 1.

`Cursor Forward` `<ESC>[{COUNT}C`

Moves the cursor *forward* by *COUNT* columns; the default count is 1.

`Cursor Backward` `<ESC>[{COUNT}D`

Moves the cursor *backward* by *COUNT* columns; the default count is 1.

`Force Cursor Position` `<ESC>[{ROW};{COLUMN}f`

Identical to `Cursor Home`.

`Save Cursor` `<ESC>[s`

Save current cursor position.

`Unsave Cursor` `<ESC>[u`

Restores cursor position after a `Save Cursor`.

`Save Cursor & Attrs` `<ESC>7`

Save current cursor position.

`Restore Cursor & Attrs` `<ESC>8`

Restores cursor position after a `Save Cursor`.

Scrolling

`Scroll Screen` `<ESC>[r`

Enable scrolling for entire display.

`Scroll Screen` `<ESC>[{start};{end}r`

Enable scrolling from row `{start}` to row `{end}`.

`Scroll Down` `<ESC>D`

Scroll display down one line.

`Scroll Up` `<ESC>M`

Scroll display up one line.

Tab Control

`Set Tab` `<ESC>H`

Sets a tab at the current position.

`Clear Tab` `<ESC>[g`

Clears tab at the current position.

`Clear All Tabs` `<ESC>[3g`

Clears all tabs.

Erasing Text

`Erase End of Line` `<ESC>[K`

Erases from the current cursor position to the end of the current line.

`Erase Start of Line` `<ESC>[1K`

Erases from the current cursor position to the start of the current line.

`Erase Line` `<ESC>[2K`

Erases the entire current line.

`Erase Down` `<ESC>[J`

Erases the screen from the current line down to the bottom of the screen.

`Erase Up` `<ESC>[1J`

Erases the screen from the current line up to the top of the screen.

`Erase Screen` `<ESC>[2J`

Erases the screen with the background color and moves the cursor to *home*.

Printing

Some terminals support local printing:

`Print Screen` `<ESC>[i`

Print the current screen.

`Print Line` `<ESC>[1i`

Print the current line.

`Stop Print Log` `<ESC>[4i`

Disable log.

`Start Print Log` `<ESC>[5i`

Start log; all received text is echoed to a printer.

Define Key

Set Key Definition `<ESC>[{key};"{string}"p`

Associates a *string* of text to a keyboard key. `{key}` indicates the key by its ASCII value in decimal.

Set Display Attributes

Set Attribute Mode `<ESC>[{attr1};...;{attrn}m`

Sets multiple display attribute settings. The following lists standard attributes:

0	Reset all attributes
1	Bright
2	Dim
4	Underscore
5	Blink
7	Reverse
8	Hidden
	Foreground Colors
30	Black
31	Red
32	Green
33	Yellow
34	Blue
35	Magenta
36	Cyan
37	White
	Background Colors
40	Black
41	Red
42	Green

43	Yellow
44	Blue
45	Magenta
46	Cyan
47	White

[Top | Status | Setup | Fonts | Cursor | Scrolling | Tabs | Erasing | Printing | Keyboard | Colors]

ภาคผนวก ข

NVT CODE and Telnet command code

NVT CODE

Name	Code	Meaning
NULL	0	No operation
Line Feed (LF)	10	Moves theprinter to the next line keeping the jsame horizontal position.
Carriage Return (CR)	13	Movesthe printer to the left margin of the current line.
Bell (BELL)	7	Produces a sound or a visible signal,
Back Space (BS)	8	Moves one character position back toward the left.
Horizontal Tab (HT)	9	Moves to the next tab position. How tab positions are determinede is not specified.
Vertical Tab (VT)	11	Moves to the next vertical tab position. How vertical tab positions are determined is not specified.
Form Feed (FF)	12	Moves to the top of next page, keeping the same horizontal position.

TELNET Commands

Command	Code	Function Requested
Interrupt Process (IP)	244	Terminate the current process.
About Output (AO)	245	Stop sending output for a process but allow the process to run to completion.
Are You There (AYT)	246	Requests a response from the server that the server is still active.
Erase Character (EC)	247	Delete the last character of output.
Erase Line (EL)	248	Delete the current line of output.
Break	243	NVT character BRK.

ภาคผนวก ค

โปรแกรม หาและทดสอบคีย์ของการเข้ารหัส RSA

```
{ Program by : Utain Saeteng }

program rsa;
uses crt;
var  ar              :  array[0..15] of boolean;
     ch1,ch2         :  char;
     i,j,k,e,d,m,fn,cip,a,b,n,p,q,point : integer;
     long            :  longint;

procedure fast_exponential(a,b : integer; var d : integer);
var c,i,temp : integer;
   long1 : longint;
begin
    point := 0;
    for i := 0 to 15 do
    begin
        temp := b mod 2;
        if temp = 1 then ar[point] := true else ar[point] := false;
        b := b div 2;
        point := point+1;
    end;

    c:=0;
    long1:=1;
    for point := 15 downto 0 do
       begin
           c := c*2;
           long1 := (long1*long1) mod n;
           if ar[point] = true then
             begin
                 c := c+1;
                 long1 := (long1*a) mod n;
             end;
       end;
   d := long1;
end;

function inv(e,fn : integer) : longint;
var gi,ui,vi : array[0..50] of longint;
   i,y,x : integer;
begin
    gi[0] := fn;
    gi[1] := e;
```

```
    ui[0] := 1;
    ui[1] := 0;
    vi[0] := 0;
    vi[1] := 1;
    i := 1;
    while gi[i] <> 0 do
        begin
            y := gi[i-1] div gi[i];
            gi[i+1] := gi[i-1] - (y*gi[i]);
            ui[i+1] := ui[i-1] - (y*ui[i]);
            vi[i+1] := vi[i-1] - (y*vi[i]);
            i := i+1;
        end;
    x := vi[i-1];
    if x >= 0 then inv := x else inv := x+fn;
end;

begin
    clrscr;
    ch1 := readkey;
    m := ord(ch1);

    p := 47;q := 23;
    n := p*q;
    fn := (p-1)*(q-1);
  { encrype }
    for k := 0 to 255 do
    begin
        m := k;
        e := 91;
        fast_exponential(m,e,cip);
        write('m= ',m,' e= ',e,' n= ',n,' cip= ',cip);
  { decrypt }
        long := inv(e,fn);
        d := long;
        fast_exponential(cip,d,m);
        writeln('   m = ',m);
        if k mod 25 = 0 then readln;
    end;
readln;
end.
```

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาโครงงานนี้ ซึ่งให้คำแนะนำแหล่งค้นคว้าและศึกษาข้อมูลรวมถึงตำราต่างๆที่จำเป็นต้องใช้ในโครงงานนี้ และขอขอบคุณ นายเกรียรติพงษ์ กนกบรรณกร ซึ่งเป็นธุระในการช่วยค้นหา tenet control for visual Basic ซึ่งหากไม่ได้บุคคลทั้งสองท่านแล้ว โครงงานนี้คงจะทำไม่สำเร็จ

หนังสืออ้างอิง

1. ยุทธนา สวนสุข, "การเข้ารหัสข้อมูลในระบบอินเทอร์เน็ต", วารสารอินเทอร์เน็ต-อินทราเน็ต, ปีที่1, ฉบับที่1 , 2539 , หน้า 65-73.

2. Charles P. Pfleeger, "Security in Computing", Prentice Hall, 538 p. , 1989.

3. Douglas E. Comer, "Internetworking with TCP/IP volume1", Prentice Hall, 474 p. , 1991.

4. Douglas E. Comer and David L. Stevens, "Internetworking with TCP/IP volume3", Prentice Hall, 498 p. , 1993.

5. William Stallings,Ph.D., "Network and Internetwork Security", Prentice Hall, 325 p. , 1995.

6. Michael Padovano, "Networking Applications on UNIX SystemV Release 4 ", Prentice Hall, 650 p. , 1993.