การพัฒนาซอฟต์แวร์ระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์

นายสุขุม เจียมชูโรจน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2544

ISBN 974-03-0469-9

SOFTWARE IMPLEMENTATION OF OFF-LINE TRACEABLE ANONYMOUS DIGITAL CASH SYSTEM

MR. SUKUM JAIMCHUROJ

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2001

ISBN 974-03-0469-9

หัวข้อวิทยานิพนธ์	การพัฒนาซอฟต์แวร์ระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์
โดย	นายสุขุม เจียมชูโรจน์
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	อาจารย์สุวิทย์ นาคพีระยุทธ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ศาสตราจารย์ ดร.ประสิทธิ์ ประพิณมงคลการ)

..... อาจารย์ที่ปรึกษา
(อาจารย์สุวิทย์ นาคพีระยุทธ)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

สุขุม เจียมชูโรจน์ : การพัฒนาซอฟต์แวร์ระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ (SOFTWARE IMPLEMENTATION OF OFF-LINE TRACEABLE ANONYMOUS DIGITAL CASH SYSTEM) อ.ที่ปรึกษา: อาจารย์ สุวิทย์ นาคพีระยุทธ 79 หน้า. ISBN 974-03-0469-9.

วิทยานิพนธ์นี้นำเสนอต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ ระบบนี้จัดเป็นระบบการจ่ายเงินอิเล็กทรอนิกส์แบบ token-based ระบบหนึ่ง ทำงานด้วยซอฟต์แวร์ล้วน และเป็นระบบแบบออฟไลน์ ระบบนี้จะมีคุณสมบัติพื้นฐานของระบบรักษาความปลอดภัยข้อมูล และที่สำคัญคือให้ความเป็นส่วนตัว นอกจากนี้ระบบจะสามารถตามรอยผู้ทุจริตได้เมื่อเกิดการใช้เงินสดดิจิตอลซ้ำ

วิทยาการเข้ารหัสลับถูกนำมาใช้เพื่อให้ได้มาซึ่งความปลอดภัยและคุณสมบัติต่างๆของระบบ ซอฟต์แวร์ที่พัฒนาขึ้นประกอบด้วยสองโปรแกรมด้วยกัน ในส่วนโปรแกรมของธนาคารจะมีฐานข้อมูลเพื่อใช้ในการเก็บข้อความระบุตัวของผู้ใช้ และเก็บข้อมูลของเงินสดดิจิตอลที่ใช้แล้ว ในส่วนโปรแกรมของผู้ใช้จะมีฟังก์ชันเพื่อตอบสนองโพรโตคอลการไหลของเงิน ได้แก่ การซื้อเงิน การคืนเงิน การจ่าย/ทอนเงิน และการขึ้นเงิน เป็นต้น

เวลาคำนวณในการทำงานของซอฟต์แวร์ขึ้นอยู่กับค่าพารามิเตอร์ต่างๆที่กำหนดไว้ เช่น จำนวนของใบสั่งเงินที่ปกปิดในขั้นตอนการซื้อเงิน, จำนวนคู่ของข้อความระบุตัว, ขนาดของข้อความระบุตัว และขนาดของกุญแจที่ใช้ในการเข้ารหัสลับ เป็นต้น

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2544

4170591221 : MAJOR ELECTRICAL ENGINEERING
KEY WORD: CRYPTOGRAPHY / DIGITAL CASH / PRIVACY / TRACEABLE
SUKUM JAIMCHUROJ: SOFTWARE IMPLEMENTATION OF OFF-LINE TRACEABLE ANONYMOUS DIGITAL CASH SYSTEM. THESIS ADVISOR: SUVIT NAKAPEERAYUT. 79 pp. ISBN 974-03-0469-9.

This thesis proposes a prototype of the off-line traceable anonymous digital cash system. This system is a token-based electronic payment. It is a pure software and off-line system. It has basic properties of security system and privacy. Double spending can be traced.

This prototype uses cryptography to make a secure system and other properties for it. The software is composed of two programs. The bank's program has database for keeping user's identity string and digital cash. The user's program can function on various transaction protocols such as purchasing, refunding, paying/receiving, and claiming digital cash.

The processing time of this software depends on parameters such as number of blind money orders, number of identity string pairs, identity string size, and key length.

Department Electrical Engineering Student's signature .....
Field of study Electrical Engineering Advisor's signature .....
Academic year 2001 .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ อาจารย์ สุวิทย์ นาคพีระยุทธ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำ ข้อคิดเห็น และสนับสนุนอุปกรณ์เครื่องมือต่างๆ ในการทำวิจัยมาด้วยดีตลอด ผู้วิจัยจึงขอกราบขอบพระคุณมา ณ ที่นี้

ขอขอบคุณ เพื่อนพี่น้องนิสิตที่อยู่ภายในห้องปฏิบัติการวิจัยระบบโทรคมนาคม ที่ได้ช่วยเหลือ และเป็นกำลังใจที่ดียิ่งต่อผู้วิจัย

ท้ายนี้ผู้วิจัยขอกราบขอบพระคุณบิดามารดา ที่ให้การสนับสนุนแก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ช่วงเวลาที่ผ่านมาอินเทอร์เน็ตได้พัฒนาจนสามารถใช้งานได้ง่ายระดับหนึ่งแล้ว เป็นที่แน่นอนว่าจะต้องมีการใช้งานเครือข่ายอินเทอร์เน็ตกันในเชิงธุรกิจมากขึ้น เช่นการใช้ไปรษณีย์อิเล็กทรอนิกส์ (Electronic Mail หรือ E-mail) สำหรับการติดต่อสื่อสาร หรือการใช้เว็บไซต์เพื่อประชาสัมพันธ์องค์กร ซึ่งเป็นการใช้ประโยชน์ขั้นพื้นฐานที่เห็นได้ทั่วไปสำหรับองค์กรธุรกิจที่เริ่มต้นใช้งานเครือข่ายอินเทอร์เน็ต

แนวโน้มในขั้นตอนไป องค์กรธุรกิจก็จะใช้เครือข่ายอินเทอร์เน็ตในการขายสินค้าและบริการขององค์กร และตรงนี้เองที่เทคโนโลยีที่เกี่ยวข้องกับความปลอดภัยในการป้องกันหรือเก็บรักษาความลับของข้อมูลจะเข้ามามีบทบาท และเป็นตัวแปรสำคัญที่จะทำให้ธุรกิจในอินเทอร์เน็ตแพร่หลายมากกว่าที่เป็นอยู่ในปัจจุบัน

การนำเอาเครือข่ายอินเทอร์เน็ตมาใช้งานในเชิงพาณิชย์ หรือที่เรียกกันว่า การพาณิชย์อิเล็กทรอนิกส์(Electronic Comerce หรือ E-commerce) ได้นั้น สิ่งสำคัญที่สุดสิ่งหนึ่งที่จะช่วยให้การทำธุรกิจผ่านเครือข่ายอินเทอร์เน็ตเติบโตขึ้นได้ ก็คือ เรื่องความปลอดภัยหรือความเชื่อถือไว้วางใจได้ในเครือข่ายอินเทอร์เน็ตนั่นเอง

ปัจจุบันก็ได้มีการนำเสนอระบบการจ่ายเงินอิเล็กทรอนิกส์(Electronic Payment) ผ่านเครือข่ายอินเทอร์เน็ตขึ้นมาหลายระบบในขณะที่ก็มีหลายระบบที่กำลังพัฒนาอยู่ เพื่อให้ได้มาซึ่งความปลอดภัยหรือความเชื่อถือไว้วางใจได้ เช่น การพิสูจน์ตัวจริง (Authentication), บูรณภาพของข้อมูล (Integrity), การไม่สามารถปฏิเสธได้ (Non-Repudiation), การรักษาความลับของข้อมูล (Confidentiality), ความเป็นส่วนตัว (Privacy) ฯลฯ ขึ้นกับจุดประสงค์ของระบบนั้นๆ การจ่ายเงินในการซื้อสินค้าในรูปแบบทั่วๆไปที่มีกันอยู่ ไม่ว่าจะเป็นการจ่ายเงินสด, การจ่ายด้วยบัตรเครดิต, การจ่ายด้วยเช็ค, การโอนเงินเข้าบัญชี และอีกหลายๆวิธีการ บางวิธีก็สามารถนำมาใช้ได้ทางอินเทอร์เน็ต ในขณะที่บางวิธีการก็ใช้ไม่ได้กับเครือข่ายอินเทอร์เน็ต เนื่องจากเครือข่ายอินเทอร์เน็ตมีรูปแบบการติดต่อสื่อสารข้อมูลในเชิงอิเล็กทรอนิกส์ระหว่างคอมพิวเตอร์ ดังนั้นผู้ซื้อจึงไม่สามารถจ่ายเงินสดให้กับผู้ขายได้ การใช้บัตรเครดิตโดยเซ็นชื่อรับรองการใช้จ่ายในสลิปบัตรก็ทำไม่ได้ ผู้ให้บริการบัตรเครดิตจึงจำต้องมีการดัดแปลงรูปแบบการจ่ายกันใหม่ เพื่อให้สามารถใช้ได้กับเครือข่ายอินเทอร์เน็ต

ระบบการจ่ายเงินอิเล็กทรอนิกส์แบ่งตามชนิดของระบบได้เป็น 2 ประเภท คือระบบแบบ token-based หรือ cash-like ซึ่งในระบบนี้ผู้ใช้ต้องซื้อโทเค็น (token) จากธนาคารก่อนที่จะนำไป

ใช้ โดยโทเค็นจะเป็นตัวกลางในการซื้อขายคล้ายกับเหรียญเงินหรือธนบัตร ตัวอย่างระบบที่มีอยู่ได้แก่ DigiCash และ NetBill เป็นต้น และระบบแบบ credit/debit-based ซึ่งมีลักษณะคล้ายกับการใช้เช็คหรือบัตรเครดิต ตัวอย่างระบบที่มีอยู่ได้แก่ SET(Secure Electronic Transaction) และ CyberCash เป็นต้น [1 และ 2] ระบบการจ่ายเงินแบบอิเล็กทรอนิกส์ยังแบ่งได้เป็นแบบออนไลน์ คือ แต่ละการจ่ายเงินระหว่างผู้ใช้กับร้านค้าจะต้องมีการติดต่อไปยังธนาคารด้วยในขณะนั้น และแบบออฟไลน์ คือ ผู้ใช้สามารถทำการจ่ายเงินไปยังร้านค้าโดยไม่ต้องมีการติดต่อไปยังธนาคารขณะนั้น ร้านค้าสามารถตรวจสอบการจ่ายเงินได้เอง นอกจากนี้ระบบก็มีทั้งที่ใช้ซอฟแวร์กับเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป และแบบที่ต้องมีการใช้ฮาร์ดแวร์เฉพาะ อย่างไรก็ตามแต่ละระบบก็ยังมีสิ่งที่แตกต่างกันเช่น ในแง่ของระดับความปลอดภัย, ความสะดวกรวดเร็วในการใช้งาน, ความเป็นส่วนตัว เป็นต้น [3]

ในวิทยานิพนธ์นี้ได้นำเสนอการออกแบบและพัฒนาต้นแบบระบบเงินสดดิจิตอลแบบตามรอยได้ชนิดออฟไลน์ ซึ่งเป็นระบบจ่ายเงินแบบอิเล็กทรอนิกส์แบบ token-based ระบบหนึ่งทำงานด้วยซอฟต์แวร์ล้วน เป็นระบบแบบออฟไลน์ ระบบนี้จะมีคุณสมบัติพื้นฐานของระบบรักษาความปลอดภัยข้อมูล และที่สำคัญคือให้ความเป็นส่วนตัว (Privacy) ซึ่งระบบแบบ credit/debit-based ไม่สามารถให้ได้ และการตามรอยได้ (Traceable) เมื่อเกิดการทุจริตใช้เงินอิเล็กทรอนิกส์ซ้ำ

ในบทนี้จะกล่าวถึงวัตถุประสงค์ ขอบเขตของวิทยานิพนธ์ ขั้นตอนการดำเนินงาน ประโยชน์ที่ได้รับจากวิทยานิพนธ์ ภาพรวมของเนื้อหาในแต่ละบทของวิทยานิพนธ์

บทที่ 2 จะกล่าวถึงภาพรวมของวิทยาการเข้ารหัสลับ (Cryptography) ขั้นตอนวิธีและโพรโตคอลของวิทยาการเข้ารหัสลับ (Cryptography Algorithms and Protocols) ที่ใช้ในระบบเงินสดดิจิตอลแบบตามรอยได้ชนิดออฟไลน์

บทที่ 3 จะกล่าวถึงภาพรวมและการออกแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ ได้แก่คุณสมบัติของระบบต่างๆ รวมถึงคุณสมบัติที่ขาดไป ส่วนประกอบของระบบ สมมติฐานเบื้องต้นของระบบ การไหลของเงินอิเล็กทรอนิกส์และโพรโตคอลที่เกิดขึ้นระหว่างผู้ใช้และธนาคาร และสุดท้ายคือการตรวจจับทุจริตหรือใช้เงินอิเล็กทรอนิกส์ซ้ำ

บทที่ 4 จะกล่าวถึงต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ที่ถูกพัฒนาขึ้นด้วยซอฟต์แวร์ล้วน

บทที่ 5 กล่าวสรุปผลการวิจัย และข้อเสนอแนะสำหรับการวิจัยในอนาคต

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อออกแบบและพัฒนาระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์
2. พัฒนาซอฟต์แวร์ที่ใช้ในระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์

1.3 ขอบเขตของการวิจัย

1. พัฒนาระบบที่มีลักษณะดังนี้
    1.1. ทำงานด้วยซอฟต์แวร์ล้วนไม่ต้องการฮาร์ดแวร์เฉพาะ
    1.2. เป็นระบบออฟไลน์และสามารถระบุผู้ทุจริตใช้เงินอิเล็กทรอนิกส์ซ้ำได้
    1.3. ระบบให้ความเป็นส่วนตัวแก่ผู้ใช้ระบบ ถ้าไม่มีการทุจริตใช้เงินซ้ำระบบจะไม่สามารถระบุได้ว่าเจ้าของเงินอิเล็กทรอนิกส์คือใคร
    1.4. ความปลอดภัยของระบบจะอาศัยขั้นตอนวิธีของวิทยาการเข้ารหัสลับในการป้องกันการแก้ไขหรือปลอมแปลงของบุคคลอื่น
    1.5. เงินอิเล็กทรอนิกส์ จะมีอายุการใช้งานเพื่อแก้ปัญหาขนาดของฐานข้อมูล
2. พัฒนาโปรแกรมที่ใช้กับระบบตามข้อ 2 ประกอบด้วย
    2.1. ซอฟต์แวร์ของผู้ใช้
    2.2. ซอฟต์แวร์ของธนาคาร

1.4 วิธีดำเนินการวิจัย

1. ศึกษาวิทยาการเข้ารหัสลับ (Cryptography)
2. ศึกษาระบบจ่ายเงินอิเล็กทรอนิกส์แบบต่างๆ
3. ศึกษาขั้นตอนวิธีและโพรโตคอลของวิทยาการเข้ารหัสลับที่ใช้ในระบบเงินสดดิจิตอล
4. ออกแบบและพัฒนาระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์
5. พัฒนาโปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับผู้ใช้ และธนาคาร
6. ทดสอบและปรับปรุงระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์
7. สรุปผลการทดสอบและเขียนวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ที่ใช้งานได้จริง
2. เป็นทางเลือกของการพาณิชย์อิเล็กทรอนิกส์ที่สามารถรับประกันความเป็นส่วนตัว

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

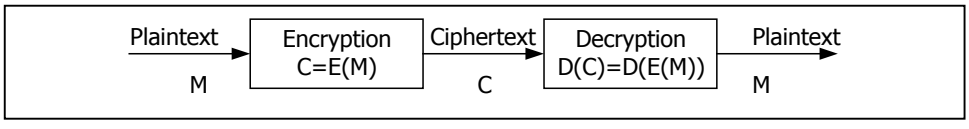
เนื้อหาในบทนี้จะกล่าวถึง วิทยาการเข้ารหัสลับในส่วนของภาพรวม วิทยาการเข้ารหัสลับที่ใช้ในระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ ในส่วนของขั้นตอนวิธีของวิทยาการเข้ารหัสลับ คือ RSA DES (Data Encryption Standard) และ MD5 และในส่วนของโพรโตคอลของวิทยาการเข้ารหัสลับ คือ การพิสูจน์ตัวจริงโดยใช้ RSA โพรโตคอล Secret Splitting โพรโตคอล Bit Commitment โพรโตคอล Blind Signature และโพรโตคอล Digital Cash

2.1 วิทยาการเข้ารหัสลับ (Cryptography)

การเข้ารหัสลับ (Encryption) กับการถอดรหัสลับ (Decryption) เป็นส่วนหนึ่งของศาสตร์และศิลป์ที่ใช้ในการเก็บรักษาข้อมูลให้เป็นความลับ ที่เรียกว่า วิทยาการเข้ารหัสลับ (Cryptography) และผู้ใช้ศาสตร์นี้เรียกว่า นักเข้ารหัสลับ (Cryptographer) ศาสตร์และศิลป์ที่ศึกษาเพื่อพยายามถอดรหัสข้อมูลที่เป็นความลับนี้ เรียกว่า วิทยาการถอดรหัสลับ (Cryptanalysis) และผู้ใช้ศาสตร์นี้เรียกว่า นักถอดรหัสลับ (Cryptanalyst) ส่วนศาสตร์ในแขนงคณิตศาสตร์ที่รวมทั้ง 2 ศาสตร์นี้เข้าด้วยกัน เรียกว่า Cryptology และผู้ใช้ศาสตร์นี้เรียกว่า Cryptologist

การเข้ารหัสลับ คือการแปลงข้อมูลให้อยู่ในรูปแบบที่ไม่สามารถที่จะอ่านได้ถ้าปราศจากกุญแจ จุดประสงค์หลักก็คือ ต้องการความเป็นส่วนตัวโดยการปกปิดข้อมูลจากผู้อื่นที่ไม่ต้องการให้รู้ การถอดรหัสลับ คือการแปลงข้อมูลที่ถูกเข้ารหัสลับไว้กลับเป็นข้อมูลที่สามารถอ่านได้

กำหนดให้ข้อมูล (Plaintext) คือข้อมูลที่ผู้ส่งต้องการส่งไปยังผู้รับ และไม่ต้องการถูกดักฟังโดยผู้อื่น ผู้ส่งสามารถเข้ารหัสลับข้อมูลก่อน จะได้ข้อมูลที่เข้ารหัสลับ (Ciphertext) จากนั้นจึงค่อยทำการส่งข้อมูลที่เข้ารหัสลับนี้ไป ทางผู้รับเมื่อได้รับข้อมูลที่เข้ารหัสลับมาก็จะทำการถอดรหัสลับข้อมูล จะได้ข้อมูลเดิมกลับมา ถ้ากำหนดให้ข้อมูล (Plaintext) คือ M ข้อมูลที่เข้ารหัสลับ (Ciphertext) คือ C ฟังก์ชันการเข้ารหัสลับ คือ E() และฟังก์ชันการถอดรหัสลับ คือ D() จะแสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 การเข้ารหัสลับและการถอดรหัสลับ

โปรแกรมประยุกต์ที่ใช้วิทยาการเข้ารหัสลับต่างๆนั้นมีหลายระดับด้วยกันขึ้นอยู่กับความซับซ้อนของการประยุกต์นั้นๆ โดยทั่วไปถ้าระบบมีความซับซ้อนมากขึ้น ความเร็วในการใช้การก็จะลดลง ตัวอย่างโปรแกรมประยุกต์อย่างง่ายๆ ได้แก่ ช่องการสื่อสารที่ปลอดภัย (Secure Communication) การพิสูจน์ตัวจริง (Authentication) และการแบ่งความลับ (Secret Sharing) เป็นต้น สำหรับโปรแกรมประยุกต์ที่ซับซ้อนยิ่งขึ้น ได้แก่ การพาณิชย์อิเล็กทรอนิกส์ (Electronic Commerce) ระบบการรับรอง (Certification) และ การกู้กุญแจ (Key Recovery) เป็นต้น

2.2 คุณสมบัติพื้นฐานของระบบความปลอดภัยข้อมูล

จุดประสงค์ของวิทยาการเข้ารหัสลับ คือ การทำให้เกิดความปลอดภัยแก่ข้อมูลของระบบ ซึ่งคุณสมบัติพื้นฐานของระบบความปลอดภัยข้อมูล จะประกอบด้วย 4 ข้อด้วยกัน [4] คือ

2.2.1 **การรักษาความลับของข้อมูล** (Confidentiality)

การรักษาความลับของข้อมูลที่ดี จะทำให้ผู้อื่นไม่สามารถลักลอบอ่านข้อมูลหรือนำข้อมูลไปใช้ได้ เนื่องจากข้อมูลจะถูกทำการเข้ารหัสลับไว้ก่อนทำการส่งออกไปยังผู้รับ ผู้ที่ลักลอบอ่านข้อมูลก็จะอ่านได้แต่เพียงข้อมูลที่เข้ารหัสลับแล้วซึ่งไม่เป็นประโยชน์แต่อย่างไร ผู้รับที่มีกุญแจที่ถูกต้องเท่านั้นจึงจะสามารถทำการถอดรหัสลับข้อมูลที่เข้ารหัสลับนี้เป็นข้อมูลที่อ่านได้อย่างถูกต้อง

2.2.2 **การพิสูจน์ตัวจริง** (Authentication)

ในระบบรักษาความปลอดภัยของข้อมูลนั้น การพิสูจน์ตัวจริงเป็นกระบวนการระบุตัวผู้ส่งต่อผู้รับ เพื่อให้ผู้รับแน่ใจได้ว่าผู้ที่กำลังติดต่ออยู่ด้วยเป็นตัวผู้ส่งจริง ไม่ใช่ผู้อื่นที่ปลอมแปลงเข้ามา กระบวนการนี้มักจะใช้ในตอนเริ่มต้นของการติดต่อระหว่างผู้ส่งและผู้รับ ก่อนที่จะทำการส่งข้อมูลระหว่างกัน

2.2.3 **บูรณภาพของข้อมูล** (Integrity)

ข้อมูลที่ถูกส่งออกมาจากผู้ส่งและถึงผู้รับโดยไม่มีการเปลี่ยนแปลง คือคุณสมบัติที่เรียกว่าบูรณภาพของข้อมูล เนื่องจากข้อมูลไม่ได้รับส่งจากมือผู้ส่ง ไปสู่มือของผู้รับโดยตรง แต่จะต้องเดินทางผ่านไปทางเครือข่ายสาธารณะ จึงต้องมีกระบวนการที่ช่วยให้เกิดความมั่นใจว่า ในระหว่างทางข้อมูลจะต้องไม่ถูกมือที่สามมาเปลี่ยนแปลงแก้ไข หรือในกรณีที่ข้อมูลถูกเปลี่ยนแปลงแก้ไขไปแล้ว จะต้องมีวิธีที่ตรวจสอบได้ เพื่อจะได้ไม่ต้องเชื่อถือข้อมูลนั้นอีกต่อไป

2.2.4 **การไม่สามารถปฏิเสธได้** (non-repudiation)

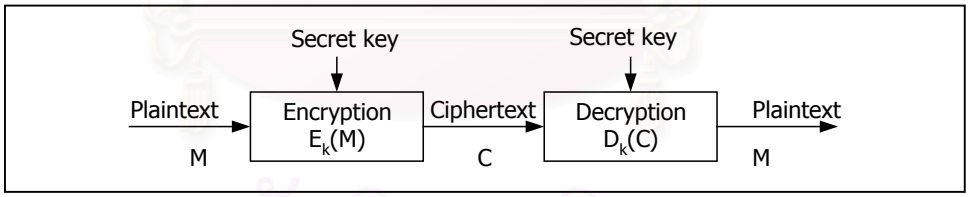
กระบวนการนี้จะทำให้ผู้ส่งไม่สามารถที่จะปฏิเสธได้ว่า ตัวเองเป็นผู้ส่งข้อมูลนี้

2.3 ขั้นตอนวิธีและกุญแจ

ขั้นตอนวิธีของวิทยาการเข้ารหัสลับ แบ่งออกได้ตามประเภทของกุญแจรหัสที่ใช้ ออกเป็น 2 ประเภท คือ ประเภทที่ใช้การเข้ารหัสลับแบบกุญแจลับ (Secret key) หรือเรียกว่า Symmetric Encryption และประเภทที่ใช้การเข้ารหัสลับแบบกุญแจสาธารณะ (Public key) หรือเรียกว่า Asymmetic Encryption อธิบายได้ดังนี้ [4 และ 5]

2.3.1 การเข้ารหัสลับแบบกุญแจลับ (Secret key หรือ Symmetric Encryption)

เทคนิคการเข้ารหัสลับแบบกุญแจลับนี้เกิดขึ้นก่อนการเข้ารหัสแบบกุญแจสาธารณะ โดยใช้หลักการที่ว่า ผู้รับและผู้ส่งจะถือกุญแจรหัสดอกเดียวกัน โดยที่ผู้ส่งจะใช้กุญแจรหัสเพื่อเข้ารหัสข้อมูลก่อนส่ง และผู้รับจะต้องใช้กุญแจรหัสที่มีรูปแบบเดียวกับที่ผู้ส่งถืออยู่ สำหรับถอดรหัสข้อมูล ซึ่งการเข้ารหัสข้อมูลแบบนี้ สามารถทำได้ทีละกลุ่มเรียกว่า Block Cipher คือจะทำการอ่านข้อมูลมาทีละกลุ่ม โดยมีความยาวของข้อมูลคงที่ แล้วทำการเข้ารหัสข้อมูลกลุ่มนั้น เมื่อได้ผลลัพธ์ออกมาเป็นข้อมูลที่เข้ารหัสแล้ว และมีความยาวคงที่เช่นเดียวกัน ส่วนอีกแบบหนึ่งคือ การเข้ารหัสแบบต่อเนื่อง หรือ Stream Cipher จะอ่านข้อมูลเข้ามาแล้วทำการเข้ารหัสข้อมูลทีละบิตต่อเนื่องกันไป เมื่อได้ผลลัพธ์ออกมาก็จะเป็นข้อมูลที่เข้ารหัสแล้วทีละส่วนต่อเนื่องกันไปเช่นกัน การเข้ารหัสลับแบบกุญแจลับหรือ Symmetric Encryption นี้ที่นิยมใช้กันมากนั้นมีอยู่หลายชนิดด้วยกัน ตัวอย่างเช่น DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm), RC2 และ RC4 เป็นต้น



รูปที่ 2.2 การเข้ารหัสลับและการถอดรหัสลับด้วยกุญแจลับ

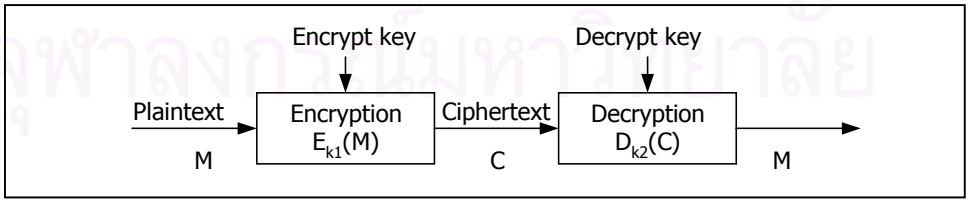
ข้อเสียของการเข้ารหัสลับแบบกุญแจลับ อยู่ที่ตัวกุญแจรหัส เนื่องจากผู้รับและผู้ส่งใช้กุญแจรหัสดอกเดียวกัน นั่นหมายความว่า ถ้ามีคนอื่นซึ่งถือกุญแจรหัสแบบเดียวกันนี้อยู่ ก็สามารถที่จะถอดรหัสข้อมูลได้อย่างง่ายดาย และยิ่งถ้ากุญแจรหัสจะต้องถูกส่งไปยังผู้รับข้อมูลปลายทางเพื่อใช้ในการถอดรหัสข้อมูลโดยผ่านเครือข่ายอินเทอร์เน็ตด้วยแล้ว โอกาสที่กุญแจรหัสจะถูกลักลอบนำไปใช้ถอดรหัสข้อมูลโดยบุคคลอื่น ก็ยิ่งมีความเสี่ยงสูงเป็นทวีคูณ จากข้อบกพร่องนี้เอง จึงได้มีการพัฒนาเทคนิคของการเข้ารหัสข้อมูลมาเป็นการเข้ารหัสแบบกุญแจสาธารณะ ที่ผู้ส่งและผู้รับข้อมูลจะถือกุญแจคนละดอกที่ไม่เหมือนกันในการเข้ารหัสและถอดรหัสข้อมูล

2.3.2 การเข้ารหัสลับแบบกุญแจสาธารณะ (Public key หรือ Asymmetric Encryption)

การเข้ารหัสลับแบบกุญแจสาธารณะหรือ Asymmetric Encryption ถูกคิดค้นขึ้นในปี ค.ศ. 1976 โดย Whitfield Diffie และ Martin Hellman เพื่อแก้ไขข้อเสียของการเข้ารหัสลับแบบกุญแจลับหรือ Symmetric Encryption ที่ใช้กุญแจลับดอกเดียวกันในการเข้ารหัสและถอดรหัสข้อมูล โดย Asymmetric Encryption จะเป็นการเข้ารหัสข้อมูลที่ใช้กุญแจ 2 ดอก คือผู้รับและผู้ส่งข้อมูลจะใช้กุญแจคนละดอกกัน โดยกุญแจรหัสนี้จะได้จากโปรแกรมการสร้างกุญแจรหัส มีด้วยกัน 2 ชุดคือ กุญแจสาธารณะ (Public key) และกุญแจส่วนตัว (Private key) คู่กุญแจนี้จะถูกเชื่อมโยงกันด้วยสูตรทางคณิตศาสตร์เพื่อให้เข้าคู่กันได้และไม่ซ้ำกับคู่อื่น ผู้รับและผู้ส่งจะสามารถติดต่อสื่อสารกันได้เพียงแค่รู้กุญแจสาธารณะของอีกฝ่ายหนึ่ง โดยผู้ส่งจะเข้ารหัสข้อมูลที่จะส่งให้ผู้รับที่ต้องการด้วยการใช้กุญแจสาธารณะของผู้รับ แล้วจึงส่งไปให้ผู้รับ ดังนั้นจะมีเพียงกุญแจส่วนตัวของผู้รับเท่านั้นที่สามารถถอดรหัสเอาข้อมูลออกมาได้

สิ่งที่น่าสนใจของ Asymmetric Encryption ก็คือ กุญแจสาธารณะ และกุญแจส่วนตัว จะใช้งานสลับกันได้ หมายถึงถ้าเรามีกุญแจคู่หนึ่ง เราสามารถใช้กุญแจดอกใดก็ได้ในการเข้ารหัสและใช้กุญแจดอกที่เหลือในการถอดรหัส และคุณสมบัติอีกอย่างก็คือ ถึงแม้ว่าจะทราบกุญแจสาธารณะในการเข้ารหัสข้อมูล แต่ก็เป็นการยากที่จะนำกุญแจนี้มาคำนวณย้อนกลับไปหากุญแจส่วนตัว และไม่สามารถถอดรหัสข้อมูลนั้นได้ ในการเข้ารหัสข้อมูล จึงสามารถเปิดเผยกุญแจสาธารณะได้ มีเพียงผู้รับข้อมูลเท่านั้นที่มีกุญแจส่วนตัวในการถอดรหัสข้อมูล

ข้อเสียของการใช้ Asymmetric Encryption ก็คือ การเข้ารหัสและการถอดรหัสจะเสียเวลานานกว่า Symmetric Encryption เนื่องจากมีความซับซ้อนสูงกว่า และหากเลือกใช้กุญแจลับที่มีความยาวน้อยเกินไป อาจมีผู้นำกุญแจสาธารณะและข้อมูลที่เข้ารหัสแล้วไปคำนวณย้อนกลับหากุญแจส่วนตัวได้ การเข้ารหัสลับแบบกุญแจสาธารณะหรือ Asymmetric Encryption ที่ใช้กันอย่างแพร่หลาย ได้แก่ RSA, Digital Signature Standard (DSS), Secure Electronic Transaction (SET) [6] เป็นต้น

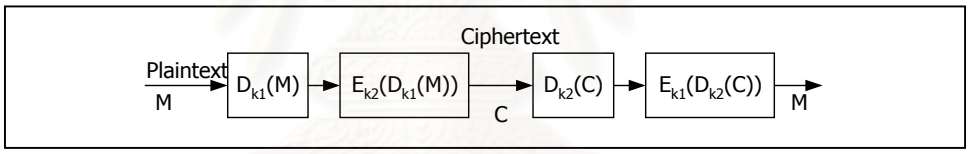


รูปที่ 2.3 การเข้ารหัสลับและการถอดรหัสลับด้วยกุญแจ 2 ดอกที่ต่างกัน

การใช้ระบบการเข้ารหัสลับแบบกุญแจสาธารณะเพื่อความเป็นส่วนตัว และการพิสูจน์ตัวจริง จะแตกต่างกันดังนี้ ในกรณีที่ผู้ส่งส่งข้อความไปให้ผู้รับ โดยใช้กุญแจสาธารณะของผู้รับในการ

เข้ารหัสลับ ก็จะสามารถส่งข้อความไปให้กับผู้รับอย่างเป็นความลับได้ แต่การส่งข้อความโดยวิธีนี้จะไม่รับประกันในเรื่องการพิสูจน์ตัวจริง เนื่องจากผู้อื่นที่รู้กุญแจสาธารณะของผู้รับ ก็จะสามารถปลอมข้อความเพื่อส่งให้ผู้รับได้เช่นกัน ส่วนในกรณีที่ผู้ส่งส่งข้อความไปให้ผู้รับ โดยใช้กุญแจส่วนตัวของผู้ส่งเองในการเข้ารหัสลับ ผู้รับก็จะสามารถใช้กุญแจสาธารณะของผู้ส่งในการถอดรหัส และผู้รับจะแน่ใจได้ว่าเป็นข้อความของผู้ส่งจริง แต่การส่งข้อความโดยวิธีนี้ก็จะไม่รับประกันว่าข้อความนี้จะเป็นความลับ เนื่องจากผู้ใดก็ตามที่รู้กุญแจสาธารณะของผู้ส่งก็จะสามารถถอดรหัสลับและรู้ข้อความได้

การจะทำให้ระบบการเข้ารหัสลับแบบกุญแจสาธารณะสามารถรับประกันได้ทั้งในเรื่องความเป็นส่วนตัว และการพิสูจน์ตัวจริง ทำได้โดยเมื่อผู้ส่งจะทำการเข้ารหัสลับข้อความที่ต้องการส่งให้ผู้รับด้วยกุญแจส่วนตัวของผู้ส่ง จากนั้นก็ทำการเข้ารหัสลับอีกชั้นหนึ่งด้วยกุญแจสาธารณะของผู้รับ ทางผู้รับจะทำการถอดรหัสลับด้วยกุญแจส่วนตัวของผู้รับ แล้วจึงใช้กุญแจสาธารณะของผู้ส่งในการถอดรหัสลับอีกชั้นหนึ่ง ก็จะได้ข้อความที่ต้องการจากผู้ส่งมา เมื่อกำหนดให้ Dk1( ) เป็นการเข้ารหัสลับด้วยกุญแจส่วนตัวของผู้ส่ง Ek2( ) เป็นการเข้ารหัสลับด้วยกุญแจสาธารณะของผู้รับ Dk2( ) เป็นการเข้ารหัสลับด้วยกุญแจส่วนตัวของผู้รับ Ek1( ) เป็นการเข้ารหัสลับด้วยกุญแจสาธารณะของผู้ส่ง ขั้นตอนต่างๆแสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 การใช้ระบบการเข้ารหัสลับแบบกุญแจกุญแจสาธารณะ
เพื่อความเป็นส่วนตัวและการพิสูจน์ตัวจริง

2.3.3 **ฟังก์ชันแฮชแบบทางเดียว** (One-way hash function) [7]

ฟังก์ชันแฮช (Hash Function; H) คือ ฟังก์ชันทางคณิตศาสตร์ที่แปลงข้อมูลขาเข้า (input; m) เป็นข้อมูลขาออกที่มีขนาดคงที่ค่าหนึ่ง ซึ่งเรียกว่า ค่าแฮช (Hash Value; h)

$$h = H(m) \tag{2-1}$$

ด้วยคุณสมบัติเพียงเท่านี้ฟังก์ชันแฮชก็ถูกนำไปใช้ทางคำนวณโดยทั่วไปได้อย่างหลากหลาย แต่เมื่อนำฟังก์ชันแฮชมาใช้ในวิทยาการเข้ารหัสลับ โดยทั่วไปจะต้องมีคุณสมบัติบางอย่างเพิ่มมากขึ้น

คุณสมบัติพื้นฐานสำหรับฟังก์ชันแฮชที่ใช้ในวิทยาการเข้ารหัสลับมีดังนี้

- ไม่จำกัดขนาดความยาวของข้อมูลขาเข้า

- ข้อมูลขาออกมีขนาดความยาวคงที่ค่าหนึ่ง
- สามารถคำนวณ H(x) ได้ง่าย สำหรับข้อมูลขาเข้าใดๆ
- H(x) เป็นฟังก์ชันแบบทางเดียว
- H(x) มีคุณสมบัติ Collision-free

ฟังก์ชันแฮชแบบทางเดียว คือ การคำนวณข้อมูลขาเข้าจากข้อมูลขาออกเป็นไปได้ยาก หรือเป็นไปไม่ได้ ถ้ากำหนดให้ x เป็นข้อมูลขาเข้าค่าหนึ่งซึ่งสามารถคำนวณหา y ข้อมูลขาเข้าอีกค่าหนึ่งที่มีค่าไม่เท่ากับ x และสามารถให้ค่าแฮชเท่ากัน H(x)=H(y) แสดงว่าฟังก์ชันแฮชนี้มีคุณสมบัติ Collision-free ที่ไม่ดี สำหรับฟังก์ชันแฮชที่มีคุณสมบัติ Collision-free ที่ดี คือ เป็นไปไม่ได้ที่คำนวณหาค่าข้อมูลขาเข้า 2 ค่าที่สามารถให้ข้อมูลขาออกหรือค่าแฮชเท่ากัน นอกจากนี้ฟังก์ชันแฮชที่ดีจะทำให้ประมาณครึ่งหนึ่งของบิตทั้งหมดของค่าแฮชเปลี่ยนโดยเฉลี่ย ถ้าการเปลี่ยนบิตข้อมูลเพียง 1 บิต

ค่าแฮช อาจเรียกอีกอย่างว่า Message Digest เนื่องจากสามารถใช้เป็นตัวแทนของข้อมูลที่มีขนาดยาวได้ ตัวอย่างของฟังก์ชันแฮช ได้แก่ MD2, MD4, MD5, และ SHA เป็นต้น

โดยทั่วไปการคำนวณของฟังก์ชันแฮชจะเร็วกว่าขั้นตอนวิธีการเข้ารหัสลับอื่นๆ จึงมักจะถูกใช้ในการคำนวณลายเซ็นอิเล็กทรอนิกส์ หรือการตรวจสอบบูรณภาพของข้อมูลเอกสารที่ถูกขั้นตอนวิธีของวิทยาการเข้ารหัสลับเป็นค่าแฮชของข้อมูลเอกสารนั้น ซึ่งจะมีขนาดเล็กเมื่อเทียบกับขนาดของข้อมูลเอกสาร นอกจากนี้ค่าแฮชยังสามารถที่จะเปิดเผยต่อสาธารณะได้โดยไม่ต้องเปิดเผยเนื้อหาของเอกสารที่ค่าแฮชคำนวณมาได้

2.4 ขั้นตอนวิธีของวิทยาการเข้ารหัสลับ (Cryptography Algorithms)

2.4.1 มาตรฐานการเข้ารหัสลับ DES [4 และ 7]

มาตรฐาน DES ย่อมาจาก Data Encryption Standard เป็นมาตรฐานซึ่งใช้เทคนิคในการเข้ารหัสลับแบบกุญแจลับ ซึ่งใช้กุญแจดอกเดียวกันในการเข้าและถอดรหัสลับ ความปลอดภัยของการเข้ารหัสแบบนี้จะอยู่ที่ว่า ทำอย่างไรถึงจะเก็บกุญแจรหัสให้เป็นความลับได้ โดยไม่ถูกลักลอบไปใช้ ในขณะที่รูปแบบหรือสูตรที่ใช้ในการเข้ารหัสลับเพื่อสร้างกุญแจรหัสจะไม่มีผลต่อความปลอดภัยของข้อมูล เพราะสูตรการสร้างกุญแจรหัสเดียวกัน สามารถสร้างกุญแจรหัสที่แตกต่างกันได้ ซึ่งขึ้นอยู่กับข้อมูลขาเข้าที่ใช้ในการสร้างกุญแจรหัสที่แตกต่างกัน

DES เป็นมาตรฐานการเข้ารหัสข้อมูลชนิดหนึ่งของ Symmetric Encryption ที่คิดค้นโดยบริษัทไอบีเอ็ม เมื่อปี ค.ศ. 1977 ต่อมาได้รับการปรับปรุงโดยกระทรวงกลาโหมของสหรัฐอเมริกาและกำหนดเป็นมาตรฐาน ANSI X3.92 และ X3.106 โดย DES ถูกออกแบบมาให้ทำงานโดยใช้ฮาร์ดแวร์เป็นตัวเข้ารหัสและถอดรหัส หรือจะใช้ซอฟต์แวร์ทำงานในลักษณะเดียวกันก็ได้ การ

ทำงานของ DES จะเป็นแบบ Block Cipher คือเข้ารหัสข้อมูลทีละกลุ่ม กลุ่มละ 64 บิต โดยใช้กุญแจลับขนาด 56 บิต ซึ่งการเข้ารหัสข้อมูล DES นี้ได้ผ่านการทดสอบจนเป็นที่ยอมรับว่า เป็นการเข้ารหัสข้อมูลที่ปลอดภัยมากที่สุดชนิดหนึ่ง

การเข้ารหัสของ DES สามารถเข้ารหัสข้อมูลได้ 2 วิธี คือ Electronic Codebook (ECB) ซึ่งจะเข้ารหัสข้อมูลทีละครั้ง ครั้งละ 64 บิต โดยใช้กุญแจลับขนาด 56 บิต ดังนั้นข้อมูล 64 บิตที่นำมาเข้ารหัสจะไม่เกี่ยวข้องกับข้อมูลส่วนอื่นๆที่เหลือเลย ส่วนอีกวิธีหนึ่ง คือ Cipher Block Chaining (CBC) จะทำการเข้ารหัสข้อมูลครั้งละ 64 บิตเช่นเดียวกัน และใช้กุญแจลับขนาด 56 บิต ต่างกันตรงที่ว่าข้อมูลที่นำมาเข้ารหัสจะถูก Exclusive OR (XOR) กับข้อมูล 64 บิตก่อนหน้านั้นเสียก่อนแล้วจึงนำมาเข้ารหัส ทำให้ข้อมูล 64 บิตที่มีข้อความเหมือนกัน ถูกเข้ารหัสข้อมูลแล้วได้ผลลัพธ์ออกมาไม่เหมือนกัน เป็นผลให้การเดาถอดรหัสข้อมูลยากยิ่งขึ้น

กุญแจลับขนาด 56 บิตนี้จะมีจำนวนรหัสที่เป็นไปได้ทั้งหมดประมาณ 72,000 ล้านล้านรหัส ซึ่งในการเข้ารหัสนั้นจะสุ่มรหัสใดรหัสหนึ่งมาเป็นกุญแจสำหรับการเข้ารหัสของ DES ทำให้การเดารหัสทำได้ยากมาก และนำรหัสลับที่เลือกไว้มาประมวลผล ผลการเข้ารหัสข้อมูลกลับไปกลับมาถึง 16 ครั้ง จึงถือว่าการเข้ารหัสของ DES เป็นการเข้ารหัสที่มีความปลอดภัยสูง ยากแก่การเดาถอดรหัส หากมีผู้ต้องการถอดรหัสก็จะต้องเดารหัสหลายหมื่นล้านล้านรหัสกว่าจะพบรหัสลับที่ถูกต้อง

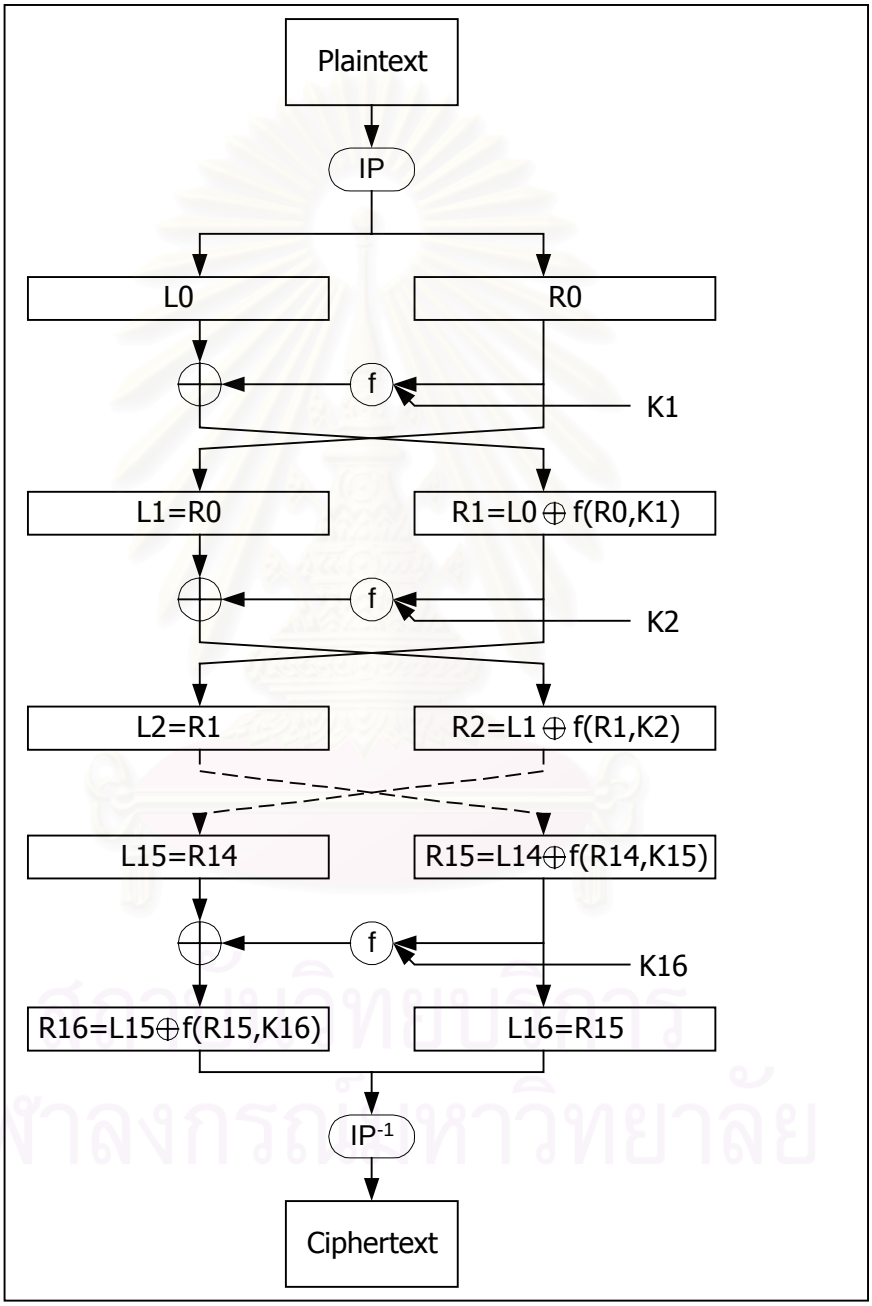
DES มีการทำงานที่รวดเร็วและสามารถทำในฮาร์ดแวร์ได้ แต่การจำหน่ายซอฟต์แวร์หรือฮาร์ดแวร์ที่ใช้การเข้ารหัสข้อมูลตามมาตรฐาน DES นอกสหรัฐอเมริกาเป็นสิ่งผิดกฎหมาย เนื่องจากรัฐบาลสหรัฐอเมริกาถือว่าเป็นผลิตภัณฑ์ควบคุม ห้ามนำออกจำหน่ายนอกประเทศ สำหรับงานเข้ารหัสข้อมูลที่ต้องการความปลอดภัยสูง จะใช้การเข้ารหัสข้อมูลที่เรียกว่า Triple-DES โดยมีหลักการทำงาน คือ ใช้ DES และกุญแจลับลำดับที่ 1 ทำการเข้ารหัสข้อมูลตามปกติ แล้วนำผลลัพธ์ที่ได้มาเข้ารหัสตามมาตรฐาน DES ครั้งที่ 2 โดยใช้กุญแจลับลำดับที่ 2 ในการเข้ารหัส จากนั้นจึงนำผลลัพธ์ที่ได้มาเข้ารหัส DES อีกครั้งหนึ่งเป็นครั้งที่ 3 โดยใช้กุญแจลับลำดับที่ 3 ในการเข้ารหัส ซึ่งกุญแจรหัสลับทั้ง 3 ตัวนี้จะมีไม่ส่วนเกี่ยวข้องกันเลย ทำให้ข้อมูลมีความปลอดภัยสูงขึ้นกว่าการเข้ารหัสตามปกติ ส่วนทางผู้รับก็จะถอดรหัสข้อมูลนี้ทีละขั้น ย้อนกลับไปจนได้ข้อมูลเดิมกลับมา

การทำงานของ DES จะเริ่มจากบล็อกข้อมูลขนาด 64 บิตถูกไปทำการสลับลำดับ จากนั้นบล็อกข้อมูลจะถูกแบ่งออกเป็นส่วนซ้ายและขวาขนาดครึ่งละ 32 บิต ข้อมูลทั้งสองส่วนจะถูกนำไปผ่านการคำนวณ 16 รอบที่เหมือนกันซึ่งแต่ละรอบการทำงานจะใช้ข้อมูลของกุญแจที่ต่างกันคือข้อมูลกุญแจจะถูกเลื่อนในระดับบิต และข้อมูลกุญแจเพียง 48 บิตจะถูกเลือกจากข้อมูลกุญแจขนาด 56 บิตมาใช้งานในแต่ละรอบ

ดังรูปที่ 2.5 บล็อกข้อมูลจะผ่าน IP (Initial Permutation) จากนั้นจะถูกแบ่งเป็น ส่วนซ้าย Li และส่วนขวา Ri แต่ละรอบที่ i จะใช้กุญแจ Ki ขนาด 48 บิต ผลลัพธ์ของฟังก์ชัน f จะถูก XOR อีกครั้ง ได้ผลลัพธ์ดังนี้

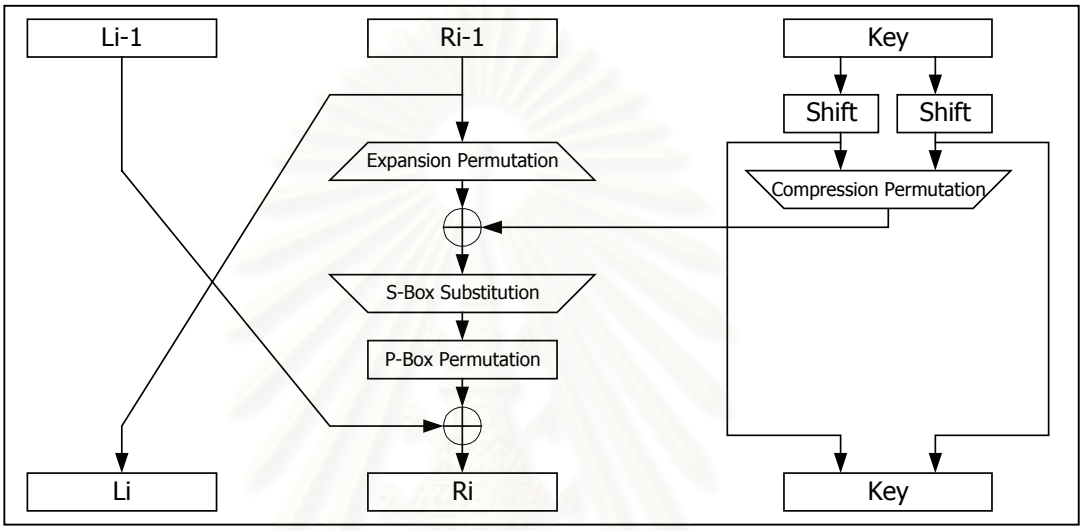
$$L_i = R_{i-1} \tag{2-2}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2-3}$$



รูปที่ 2.5 DES

ในแต่ละรอบดังรูปที่ 2.6 บิตของกุญแจจะถูกเลื่อน และหลังจากนั้นกุญแจซึ่งมีขนาด 54 บิต จะถูกเลือกมา 48 บิต ส่วนครึ่งขวาของข้อมูลขนาด 32 บิตจะถูกขยายเป็น 48 บิตโดยผ่าน Expansion Permutation แล้วมา XOR กับกุญแจย่อยซึ่งได้จาก Compression Permutation จากนั้นจะผ่าน S-Box จำนวน 8 บล็อก เพื่อสร้างบิตข้อมูลใหม่ขนาด 32 บิต แล้วจึงผ่าน P-Box Permutation อีกครั้ง กระบวนการทั้ง 4 ก็คือ ฟังก์ชัน f นั่นเอง ผลลัพธ์ขาออกที่ผ่านฟังก์ชัน f จะมา XOR กับครึ่งซ้ายของข้อมูล ก็จะได้เป็นครึ่งขวาใหม่ ส่วนครึ่งขวาเดิมจะกลายเป็นครึ่งซ้ายใหม่



รูปที่ 2.6 รอบการทำงานของ DES 1 รอบ

กุญแจลับ 64 บิตจะถูกใช้เพียง 54 บิต โดยตัดทุกๆบิตที่ 8 ทิ้ง (บิตที่8 จะใช้เป็น Parity checking และเป็น Least-significant bit ในแต่ละไบต์ของ Key) จากนั้นกุญแจขนาด 54 บิตนี้จะถูกสร้างเป็นกุญแจขนาด 48 บิตที่ต่างกัน ซึ่งจะใช้ในแต่ละรอบของทั้งหมด 16 รอบ ผลลัพธ์ที่ได้สุดท้ายทั้งส่วนซ้ายและขวาจะผ่าน IP^{-1} (Final Permutation) ซึ่งเป็นฟังก์ชันย้อนกลับของ IP ได้ $R_{16}L_{16}$ เป็นข้อมูลที่เข้ารหัสลับแล้วขนาด 64 บิต

Initial Permutation (IP)

กระบวนการนี้เป็นขั้นตอนแรกโดยจะเปลี่ยนบล็อกข้อมูลตามตารางที่ 2.1 ลักษณะของตารางจะเหมือนทุกๆตารางคือจะอ่านจากซ้ายไปขวา และบนลงล่าง ยกตัวอย่าง Initial Permutation จะย้ายบิตที่ 58 ของ Plaintext มาอยู่ที่บิตตำแหน่งที่ 1, จะย้ายบิตที่ 50 ของ Plaintext มาอยู่ที่บิตตำแหน่งที่ 2, จะย้ายบิตที่ 42 ของ Plaintext มาอยู่ที่บิตตำแหน่งที่ 3 เป็นต้น

ตารางที่ 2.1 Initial Permutation

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

Key Permutation

เริ่มแรกกุญแจขนาด 64 บิตจะถูกลดลงเหลือ 56 บิตโดยทิ้งทุกๆบิตที่ 8 ในกุญแจดังแสดงในตารางที่ 2.2 บิตเหล่านี้จะใช้เป็นParity Check เพื่อให้แน่ใจว่าไม่มีError หลังจากที่กุญแจขนาด 56 บิตถูกดึงออกมากุญแจย่อยขนาด 48 บิตจะถูกสร้างขึ้นมาสำหรับแต่ละรอบใน 16 รอบของ DES โดยกุญแจย่อย (Ki) จะถูกกำหนดตามนี้

ตารางที่ 2.2 Key Permutation

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

กุญแจขนาด 56 บิตจะถูกแบ่งเป็น 2 ครึ่งๆละ 28 บิต หลังจากนั้นทั้ง 2 ครึ่งจะถูกเลื่อนซ้ายแบบวงกลม 1 หรือ 2 บิต ขึ้นกับรอบ จำนวนบิตที่ Shift จะแสดงในตารางที่ 2.3

ตารางที่ 2.3 จำนวนบิตในการเลื่อนของกุญแจ

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

หลังจากถูกเลื่อนแล้ว กุญแจซึ่งมีขนาด 48 บิตจะถูกเลือกจาก 56 บิต ในขั้นนี้มีการสลับที่ด้วยเช่นกัน ดังนั้นจึงเรียกขั้นตอนนี้ว่า Compression Permutation แสดงดังตารางที่ 2.4 ยกตัวอย่างเช่น บิตที่ 33 ของกุญแจที่ถูกเลื่อนแล้วจะถูกย้ายมาอยู่ที่ตำแหน่งบิตที่ 35 ของข้อมูลขาออก , บิตที่ 18 ของกุญแจที่ถูกเลื่อนแล้วจะถูกละไป

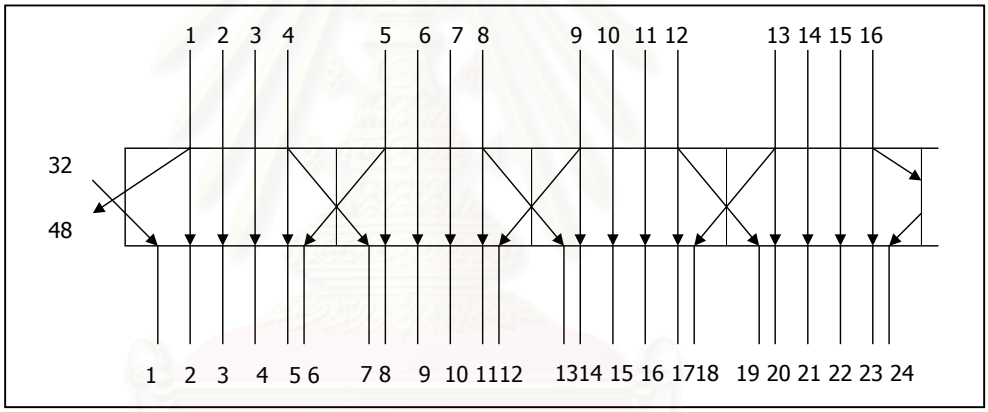
ตารางที่ 2.4 Compression Permutation

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

เนื่องจากการ Shift ทำให้ Subkey ที่ใช้แต่ละตัวต่างกันไป แต่ละบิตจะถูกใช้ในประมาณ 14 Subkeys ของทั้งหมด 16 Subkeys

The Expansion Permutation

ขั้นตอนนี้จะขยายครึ่งขวาของบล็อก (Ri) จาก 32 บิตเป็น 48 บิต โดยจะเปลี่ยนลำดับของบิตและใช้บิตซ้ำที่แน่นอน จุดประสงค์ของขั้นตอนนี้เพื่อให้ครึ่งขวาของข้อมูลมีขนาดเดียวกับกุญแจย่อย เพื่อนำมา XOR กัน



รูปที่ 2.7 Expansion Permutation

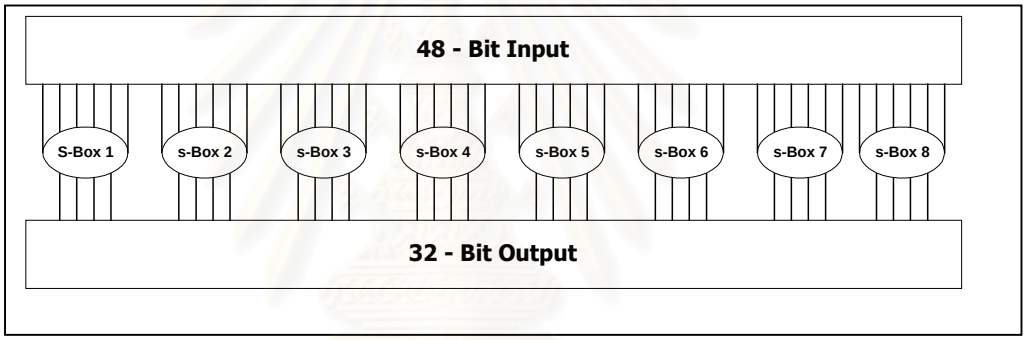
รูปที่ 2.7 แสดง Expansion Permutation บางครั้งจะเรียก E-Box ทุกๆบล็อกมีข้อมูลขาเข้า 4 บิต บิตที่ 1 และบิตที่ 4 แต่ละบิตจะไปแทน 2 บิตของข้อมูลขาออก ในขณะที่บิตที่ 2 และบิตที่ 3 แต่ละบิตจะไปแทน 1 บิตของข้อมูลขาออก ตารางที่ 2.5 แสดงตำแหน่งของข้อมูลขาออก ซึ่งขึ้นกับข้อมูลขาเข้า ยกตัวอย่าง บิตในตำแหน่งที่ 3 ของบล็อกข้อมูลขาเข้า จะย้ายไปอยู่ที่ตำแหน่งที่ 4 ของบล็อกข้อมูลขาออก และบิตในตำแหน่งที่ 21 ของ บล็อกข้อมูลขาเข้า จะย้ายไปอยู่ที่ตำแหน่งที่ 32 ของบล็อกข้อมูลขาออก เป็นต้น

ตารางที่ 2.5 Expansion Permutation

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

S-Box Substitution

หลังจากที่กุญแจย่อย XOR กับ บล็อกข้อมูลที่ถูกขยายแล้ว จะมาถึง Substitution box หรือ S-box ข้อมูลขนาด48 บิตนี้จะถูกแบ่งเป็น บล็อกย่อยขนาด 6 บิต 8 บล็อกย่อย แต่ละบล็อกจะถูกกระทำผ่าน S-Box คือ บล็อกแรกจะกระทำผ่าน S-Box 1 , บล็อกที่2 จะกระทำผ่าน S-Box 2 และ บล็อกที่3 จะกระทำผ่าน S-Box 3 เป็นต้น ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 S-Box Substitution

ดังแสดงในตารางที่ 2.6 แต่ละ S-box เป็นตารางซึ่งมี 4 แถว และ 16 ช่อง โดยที่ข้อมูลขาเข้าของแต่ละ S-box ซึ่งมีขนาด 6 บิต ดังนี้ b1, b2, b3, b4, b5, และ b6 โดย b1 และ b6 จะรวมเป็นเลข 2 บิต ซึ่งแทน 0-3 ในแต่ละแถวของตาราง ส่วน b2 ถึง b5 จะรวมเป็นเลข 4 บิต ซึ่งแทน 0-15 ในแต่ละช่องของตาราง

ยกตัวอย่าง สมมติว่า Input สำหรับ S-box ที่ 6 คือ 110011 ดังนั้น b1b6 ก็คือ 11 สอดคล้องกับแถวที่ 3 และ b2b3b4b5 ก็คือ 1001 สอดคล้องกับช่องที่ 9 ดังนั้น ดูจากตาราง S-box 6 ที่แถวที่ 3 และช่องที่ 9 จะได้ 14 ซึ่งก็คือ 1110 ดังนั้นข้อมูลขาเข้าเป็น 110011 ข้อมูลขาออกก็จะเป็น 1110

ผลลัพธ์ที่ได้จากขั้นตอนนี้คือ ข้อมูลขนาด 4 บิต 8 บล็อก หรือขนาด 32 บิต 1 บล็อก

ตารางที่ 2.6 S-Box

S-box 1:															
14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13
S-box 2:															
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10,
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5,
0,	14,	7,	11,	10,	4,	13,	1,	5,	8,	12,	6,	9,	3,	2,	15,
13,	8,	10,	1,	3,	15,	4,	2,	11,	6,	7,	12,	0,	5,	14,	9
S-box 3:															
10,	0,	9,	14,	6,	3,	15,	5,	1,	13,	12,	7,	11,	4,	2,	8,
13,	7,	0,	9,	3,	4,	6,	10,	2,	8,	5,	14,	12,	11,	15,	1,
13,	6,	4,	9,	8,	15,	3,	0,	11,	1,	2,	12,	5,	10,	14,	7,
1,	10,	13,	0,	6,	9,	8,	7,	4,	15,	14,	3,	11,	5,	2,	12
S-box 4:															
7,	13,	14,	3,	0,	6,	9,	10,	1,	2,	8,	5,	11,	12,	4,	15,
13,	8,	11,	5,	6,	15,	0,	3,	4,	7,	2,	12,	1,	10,	14,	9,
10,	6,	9,	0,	12,	11,	7,	13,	15,	1,	3,	14,	5,	2,	8,	4,
3,	15,	0,	6,	10,	1,	13,	8,	9,	4,	5,	11,	12,	7,	2,	14
S-box 5:															
2,	12,	4,	1,	7,	10,	11,	6,	8,	5,	3,	15,	13,	0,	14,	9,
14,	11,	2,	12,	4,	7,	13,	1,	5,	0,	15,	10,	3,	9,	8,	6,
4,	2,	1,	11,	10,	13,	7,	8,	15,	9,	12,	5,	6,	3,	0,	14,
11,	8,	12,	7,	1,	14,	2,	13,	6,	15,	0,	9,	10,	4,	5,	3
S-box 6:															
12,	1,	10,	15,	9,	2,	6,	8,	0,	13,	3,	4,	14,	7,	5,	11,
10,	15,	4,	2,	7,	12,	9,	5,	6,	1,	13,	14,	0,	11,	3,	8,
9,	14,	15,	5,	2,	8,	12,	3,	7,	0,	4,	10,	1,	13,	11,	6,
4,	3,	2,	12,	9,	5,	15,	10,	11,	14,	1,	7,	6,	0,	8,	13

ตารางที่ 2.6 S-Box (ต่อ)

S-box 7:															
4,	11,	2,	14,	15,	0,	8,	13,	3,	12,	9,	7,	5,	10,	6,	1,
13,	0,	11,	7,	4,	9,	1,	1,	13,	3,	5,	12,	2,	15,	8,	6,
1,	4,	11,	13,	12,	3,	7,	0,	4,	15,	6,	8,	0,	5,	9,	2,
6,	11,	13,	8,	1,	4,	10,	14,	1,	5,	0,	15,	14,	2,	3,	12
S-box 8:															
13,	2,	8,	4,	6,	15,	11,	1,	10,	9,	3,	14,	5,	0,	12,	7,
1,	15,	13,	8,	10,	3,	7,	4,	12,	5,	6,	11,	0,	14,	9,	2,
7,	11,	4,	1,	9,	12,	14,	2,	0,	6,	10,	13,	15,	3,	5,	8,
2,	1,	14,	7,	4,	10,	8,	13,	15,	12,	9,	0,	3,	5,	6,	11

P-Box Permutation

ข้อมูลขาออกขนาด 32 บิตที่ได้จาก S-box Substitution จะถูกสลับที่อีกผ่าน P-box ขั้นตอนนี้จะไม่มีการใช้บิต 2 ครั้ง และ ไม่มีการละทิ้งบิต จึงเรียกขั้นตอนนี้ว่า straight permutation แสดงในตารางที่ 2.7 ยกตัวอย่างเช่น บิตที่ 21 ของข้อมูลขาเข้าย้ายไปที่บิตที่ 4 ของข้อมูลขาออก

ในที่สุดแล้ว ผลลัพธ์ที่ได้จาก P-box permutation จะถูก XOR กับ ครึ่งซ้ายของบล็อกข้อมูลที่แบ่งไว้ตั้งแต่แรก ครึ่งซ้ายและครึ่งขวาจะสลับกันและเริ่มรอบต่อไป

ตารางที่ 2.7 P-Box

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

Final Permutation (IP^{-1})

ขั้นตอนนี้เป็นอินเวอร์สกับ Initial permutation ดังแสดงในตารางที่ 2.8 โดย $R_{16}L_{16}$จะถูกใช้เป็นข้อมูลขาเข้าสำหรับ final permutation

ตารางที่ 2.8 Final 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

สำหรับการถอดรหัสก็จะใช้ขั้นตอนวิธีเช่นเดียวกันกับตอนเข้ารหัส เพียงแต่ลำดับของกุญแจย่อย จะกลับกัน เช่นตอนเข้ารหัสใช้ $K_1,K_2,K_3,\ldots,K_{16}$ ดังนั้นตอนถอดรหัสก็จะต้องใช้ $K_{16},K_{15},K_{14},\ldots,K_1$ ตามลำดับ ในการเข้ารหัสด้วย DES นี้ นิยมใช้โหมดการทำงานแบบ CBC เนื่องจากมีความซับซ้อนมากกว่าโหมด ECB เล็กน้อย แต่ให้ความปลอดภัยสูงขึ้นมาก

ความเร็วของการเข้ารหัสลับโดย DES บนไมโครโพรเซสเซอร์ที่แตกต่างกัน แสดงดังตารางที่ 2.9

ตารางที่ 2.9 ความเร็วของ DES บนไมโครโพรเซสเซอร์

ไมโครโพรเซสเซอร์	ความเร็ว (หน่วย MHz)	จำนวนบล็อกข้อมูล (ต่อวินาที)
8080	4.7	370
68000	7.6	900
80286	6	1,100
68020	16	3,500
68030	16	3,900
80386	25	5,000
68030	50	9,600
68040	25	16,000
68040	40	23,000
80486	66	43,000

2.4.2 มาตรฐานการเข้ารหัสลับ RSA [5 และ 7]

RSA เป็นมาตรฐานการเข้ารหัสข้อมูลที่คิดค้นโดยบุคคล 3 คน คือ Ron Rivest, Adi Shamir และ Leonard Adleman ในปี 1978 ดังนั้นชื่อของ RSA จึงได้มาจากอักษรตัวแรกในนามสกุลของบุคคลทั้ง 3 คนนั่นเอง ปัจจุบันได้จัดตั้งเป็นบริษัท RSA Data Security ซึ่งมีผลิตภัณฑ์ด้านซอฟต์แวร์เกี่ยวกับการเข้ารหัสข้อมูลต่างๆ และบริการให้คำปรึกษาเกี่ยวกับความปลอดภัยข้อมูล

RSA จะประกอบด้วยกุญแจสาธารณะและกุญแจส่วนตัว ที่ใช้ในการเข้ารหัสและถอดรหัสข้อมูลตามลำดับ ซึ่งกุญแจสาธารณะนี้สามารถเปิดเผยได้ โดยกุญแจสาธารณะจะถูกแจกจ่ายไปให้บุคคลทั่วไปหรือบุคคลที่ผู้ส่งต้องการส่งข่าวสารให้ แต่กุญแจส่วนตัวจะเป็นความลับที่รู้เฉพาะผู้

ส่งเท่านั้น เนื่องจากคู่กุญแจนี้จะต้องเป็นคู่ของมันเองเท่านั้นจึงจะสามารถถอดรหัสได้ ดังนั้นตราบใดที่กุญแจส่วนตัวถูกเก็บไว้อย่างดี ก็จะป้องกันการดักฟัง หรือการปลอมแปลงข้อมูลได้

RSA ผ่านการทดสอบจนเป็นที่ยอมรับว่ามีความปลอดภัยในการเข้ารหัสข้อมูลสูงมาก เมื่อเลือกความยาวของกุญแจลับที่เหมาะสม เช่น ความยาวของกุญแจลับขนาด 512 บิตนั้นถือว่ายังไม่มีความปลอดภัยมากนัก หากต้องการความปลอดภัยสูงควรจะเป็น 1024 บิต เป็นต้น แม้คอมพิวเตอร์ในปัจจุบันมีความเร็วสูงขึ้นมาก ทำให้การกุญแจส่วนตัวของ RSA โดยไล่ไปทีละค่าสามารถทำได้ง่ายขึ้นในเวลาอันสั้น แต่ทว่าความเร็วของคอมพิวเตอร์ที่เพิ่มขึ้นนี้ ก็อาจจะเป็นตัวนำไปใช้เพิ่มความยาวของกุญแจก็ได้ ซึ่งก็ทำให้การเดากุญแจส่วนตัวยากขึ้นตามไปด้วยเช่นกัน

RSA มีหัวใจของระบบอยู่ที่ความยากในการแยกตัวประกอบของเลขจำนวนหนึ่งที่มีขนาดใหญ่มากๆ กุญแจของระบบจะสร้างมาจากฟังก์ชันของจำนวนเฉพาะ 2 จำนวน ซึ่งมีความยาวมาก คือ p และ q โดยผลคูณของทั้ง 2 จำนวนนี้คือ n

$$n = p \times q \tag{2-4}$$

จากนั้นจะเลือกค่ากุญแจสาธารณะ หรือใช้สัญลักษณ์ e ขึ้นมาค่าหนึ่ง โดย e จะต้องเป็นจำนวนเฉพาะใดๆ ที่มีค่าน้อยกว่า (p-1)(q-1) และไม่เป็นตัวประกอบของ (p-1)(q-1) เมื่อได้ค่ากุญแจสาธารณะ แล้วจำเป็นต้องหากุญแจส่วนตัวหรือ d ขึ้นมาอาศัยความสัมพันธ์ที่ว่า

$$d \times e = 1 \bmod [(p-1)(q-1)] \tag{2-5}$$

เมื่อต้องการเข้ารหัสลับข้อความให้เป็นข้อความที่ผ่านการเข้ารหัสลับแล้ว c จะทำได้ตามความสัมพันธ์

$$c = m^e \bmod n \tag{2-6}$$

เมื่อผู้รับได้รับข้อความ c แล้ว จะสามารถถอดรหัส ข้อความ c ได้ โดยอาศัยกุญแจสาธารณะ ตามความสัมพันธ์ที่ว่า

$$m = c^d \bmod n \tag{2-7}$$

จะเห็นได้ว่าการที่จะถอดรหัสลับได้นั้นจะต้องมีกุญแจสาธารณะเสมอ แต่การมีกุญแจสาธารณะ d ก็ไม่สามารถทำการเข้ารหัสลับได้

ตัวอย่าง สมมติให้ p=2 และ q=17 จะได้ว่า

$$n = p \times q = 2 \times 17 = 34$$

$$(p-1)(q-1) = 1 \times 16 = 16$$

เลือกค่า e โดยมีข้อกำหนดว่า e เป็น relatively prime กับ (p-1)(q-1)

ค่า e ที่เป็นไปได้สำหรับตัวอย่างนี้ คือ {1,3,5,7,11,13} สำหรับตัวอย่างนี้เลือก e=5 จากนั้นเลือกค่า d ตามความสัมพันธ์ตามสมการ (2-5)

$$(d \times 5) \bmod 16 = 1$$

ในตัวอย่างนี้เลือก d =13 สมมติให้ข้อความ m ที่ต้องการนำมาเข้ารหัสลับคือ 6 เราจะได้ว่า $c = m^e \bmod n = 6^5 \bmod 34 = 7776 \bmod 34 = 24$

การถอดรหัสข้อความ c ให้เป็นข้อความ m ทำได้ดังนี้

$$m = c^d \bmod n = 24^{13} \bmod 34$$

จะเห็นได้ว่าค่า 24^13 นั้นไม่สามารถคำนวณได้ด้วยเครื่องคิดเลขธรรมดา การคำนวณจึงจำเป็นต้องวิธีการกระจายโดยอาศัยความสัมพันธ์ทางคณิตศาสตร์ ดังต่อไปนี้

$$(xy) \bmod n = [(x \bmod n)(y \bmod n)] \bmod n \qquad (2\text{-}8)$$

จาก $24^{13} \bmod 34$ สามารถจัดรูปใหม่ได้เป็น

$$m = (24^3 \times 24^5 \times 24^5) \bmod 34$$
$$= [(24^3 \bmod 34)(24^5 \bmod 34)(24^5 \bmod 34)] \bmod 34$$
$$= (20 \times 28 \times 28) \bmod 34 = 6$$

จากตัวอย่างที่ได้กล่าวมาแล้วนั้นจะเห็นได้ว่าปัญหาในการเข้ารหัสลับและถอดรหัสลับแบบ RSA ก็คือการคำนวณด้วยตัวเลขที่มีความยาวมากๆ ซึ่งเครื่องคิดเลขหรือเครื่องคอมพิวเตอร์ธรรมดาอาจคำนวณโดยตรงไม่ได้ ดังนั้นจึงจำเป็นต้องศึกษาทฤษฎีการคำนวณทางคณิตศาสตร์สำหรับตัวเลขที่มีความยาวมากๆ

RSA สามารถถูกใช้สำหรับการเข้ารหัสลับและลายเซ็นอิเล็กทรอนิกส์ มีความแตกต่างกันเล็กน้อยดังนี้

- สำหรับการเข้ารหัสลับ

สมมติว่านาย ก ต้องการส่งข้อความ m ไปให้นาย ข นาย ก จะทำการสร้างข้อความที่เข้ารหัสลับ c ดังสมการ $c = m^e \bmod n$ โดย e และ n เป็นกุญแจสาธารณะของนาย ข จากนั้นนาย ก จึงทำการส่ง c ไปให้กับนาย ข ในการถอดรหัสลับนาย ข จะคำนวณดังสมการ $m = c^d \bmod n$ ซึ่ง d เป็นกุญแจส่วนตัวของนาย ข จากความสัมพันธ์ระหว่าง e และ d ทำให้ได้ข้อความ m กลับมา ดังนั้นนาย ข เป็นเพียงผู้เดียวที่สามารถถอดรหัสนี้ได้

- สำหรับลายเซ็นอิเล็กทรอนิกส์

สมมติว่านาย ก ต้องการส่งข้อความ m ไปให้นาย ข โดยที่นาย ข ต้องการแน่ใจได้ว่าข้อความนี้มาจากนาย ก นาย ก ทำการสร้างลายเซ็นอิเล็กทรอนิกส์ s ดังสมการ $s = m^d \bmod n$ ซึ่ง d เป็นกุญแจส่วนตัวของนาย ก จากนั้นนาย ก ทำการส่ง m และ s ไปให้กับนาย ข เพื่อตรวจสอบลายเซ็นอิเล็กทรอนิกส์ นาย ข จะคำนวณดังสมการ $m = s^e \bmod n$ ซึ่ง e และ n เป็นกุญแจสาธารณะของนาย ก เปรียบเทียบ m ที่ได้รับถ้าตรงกันก็แสดงว่า นาย ก ส่งข้อความ m นี้จริง

ในส่วนของซอฟต์แวร์นี้ ความเร็วในการเข้ารหัสลับโดย RSA นี้จะช้ากว่า DES ประมาณ 100 เท่า ตารางที่ 2.10 แสดงตัวอย่างความเร็วของ RSA

ตารางที่ 2.10 ความเร็วของ RSA สำหรับความยาวของโมดูลัสที่ต่างกันด้วยกุญแจสาธารณะขนาด 8 บิต (บนไมโครโพรเซสเซอร์ SPARC II) [4]

	512 บิต	768 บิต	1,024 บิต
Encrypt	0.03 วินาที	0.05 วินาที	0.08 วินาที
Decrypt	0.16 วินาที	0.48 วินาที	0.93 วินาที
Sign	0.16 วินาที	0.52 วินาที	0.97 วินาที
Verify	0.02 วินาที	0.07 วินาที	0.08 วินาที

2.4.3 MD5 (Message Digest) [4 และ 8]

MD2, MD4 และ MD5 เป็นขั้นตอนวิธีที่ถูกคิดค้นโดย Rivest มีจุดประสงค์เพื่อใช้สำหรับโปรแกรมประยุกต์ลายเซ็นอิเล็กทรอนิกส์ ซึ่งข้อมูลที่มีขนาดใหญ่จะถูกบีบอัดตามขั้นตอนวิธีการเข้ารหัสลับก่อนที่จะถูกเข้ารหัสลับด้วยกุญแจส่วนตัว ทั้ง 3 ขั้นตอนวิธีนี้จะรับข้อมูลขาเข้าที่มีขนาดความยาวไม่เจาะจงและให้ผลลัพธ์ขาออกเป็นข้อมูลขนาด 128 บิต โดยผลลัพธ์ที่ได้จะเป็นค่าแฮช (hash value) ซึ่งอาจจะเรียกว่า message digest หรือ fingerprint

MD5 ถูกคิดค้นในปี 1991 มีพื้นฐานการเข้ารหัสลับมาจาก MD4 มีการออกแบบและให้ข้อมูลขาออก 128 บิตเหมือนกัน แม้ว่าการคำนวณของ MD5 จะช้ากว่า MD4 เล็กน้อย แต่ความปลอดภัยของข้อมูลจะมากขึ้น

การทำงานของ MD5 มีขั้นตอนวิธีดังนี้

1. ข้อความจะถูกเติมท้ายด้วย Padding bit เพื่อให้ความยาวของข้อความหารด้วย 512 เหลือเศษ 448 หรือขาดอีก 64 บิต ถึงจะหารด้วย 512 ลงตัว Padding bit คือบิตขนาด 1 บิตมีค่า 1 และตามด้วยบิตค่า 0 จำนวนเท่าที่ต้องการ
2. จากนั้นต่อท้ายด้วยบิตบอกความยาว บิตบอกความยาวขนาด 64 บิต บอกความยาว b (ความยาวของข้อความก่อนเติมบิตต่อท้าย)

จาก 2 ขั้นตอนนี้จะทำให้ขนาดความยาวของข้อความคงที่เป็นจำนวนเท่าของ 512 บิต

3. ตั้งค่าเริ่มต้นให้กับตัวแปร 4 ตัวขนาดตัวละ 32 บิต (A, B, C, D) มีค่าเริ่มต้นในเลขฐาน 16 ดังนี้ (ไบต์ล่างมาก่อน) และทำการคัดลอกไปยังตัวแปร a,b,c,และd ตามลำดับ

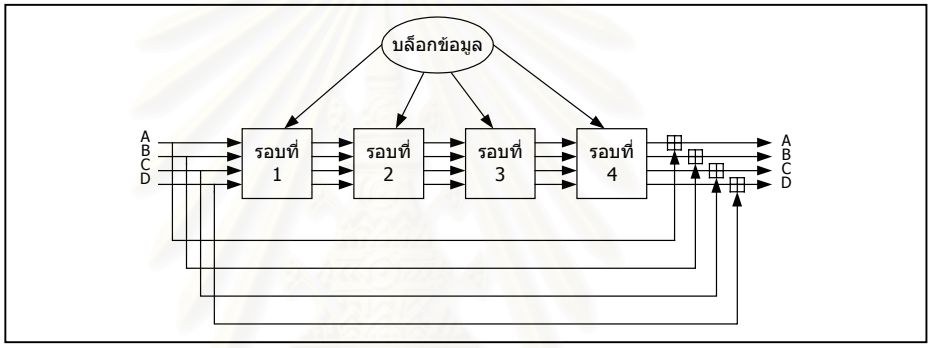
A=0x01234567

B=0x89abcdef

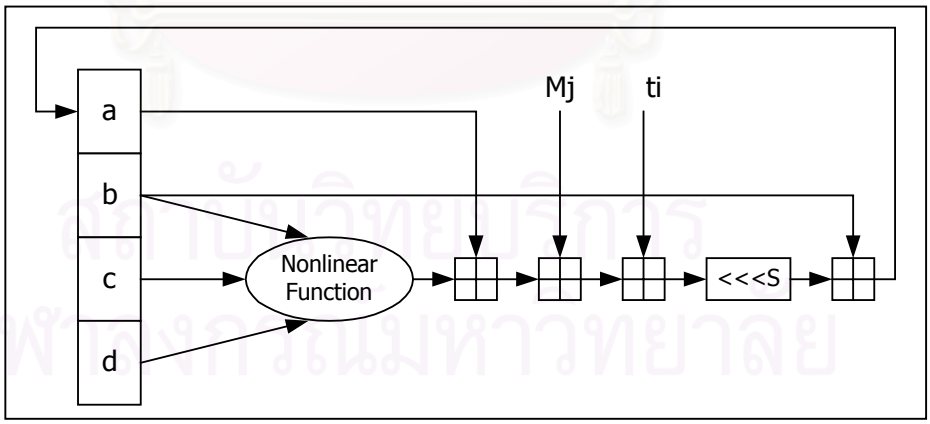
C=0xfedcba98

D=0x76543210

4. เริ่มรอบการทำงานหลัก โดยจำนวนรอบขึ้นอยู่กับจำนวนของบล็อกข้อมูลขนาด 512 บิต รอบการทำงานหลักของ MD5 จะมี 4 รอบ แต่ละรอบจะถูกใช้ในการคำนวณที่ต่างกัน 16 ครั้ง การคำนวณแต่ละครั้งจะใช้ฟังก์ชันแบบไม่เชิงเส้นโดยมีข้อมูลขาเข้าเป็น 3 ค่า จาก a,b,c,d และผลลัพธ์จะถูกเพิ่มไปยังตัวแปรที่ 4 หลังจากนั้นจะมีการหมุนผลลัพธ์ไปทางขวาตามจำนวนบิตของตัวแปร และเพิ่มให้กับ a,b,c หรือ d ตัวใดตัวหนึ่ง สุดท้ายผลลัพธ์ก็จะแทนที่ตัวแปรตัวหนึ่ง แสดงได้ดังรูปที่ 2.9 และรูปที่ 2.10



รูปที่ 2.9 รอบการทำงานหลักของ MD5



รูปที่ 2.10 การทำงานหลักของ MD5 1 รอบ

ฟังก์ชันแบบไม่เชิงเส้น แต่ละฟังก์ชันจะถูกใช้ในแต่ละรอบการทำงาน

$$F(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z) \quad (2\text{-}9)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge (\neg Z)) \quad (2\text{-}10)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z \quad (2\text{-}11)$$

$$I(X, Y, Z) = Y \oplus (X \vee (\neg Z)) \quad (2\text{-}12)$$

(หมายเหตุ $\oplus$ ¤×ÍXOR, $\wedge$ ¤×ÍAND, $\vee$ ¤×ÍOR áÅÐ$\neg$ ¤×ÍNOT)

เป็นที่น่าสังเกตว่า ถ้า X, Y และ Z เป็นอิสระต่อกัน ผลลัพธ์ของฟังก์ชันเหล่านี้ในแต่ละบิตจะเป็นอิสระต่อกันด้วย

ถ้า Mj แทนบล็อกย่อยที่ j ของข้อความ (มีค่า 0-15) และ <<<S แทนการเลื่อนแบบวงกลมไปทางซ้าย S บิต การทำงานทั้ง 4 เป็นดังนี้

$$FF(a, b, c, d, Mj, s, ti) \quad a = b + ((a + F(b, c, d) + Mj + ti) <<< s) \quad (2\text{-}13)$$

$$GG(a, b, c, d, Mj, s, ti) \quad a = b + ((a + G(b, c, d) + Mj + ti) <<< s) \quad (2\text{-}14)$$

$$HH(a, b, c, d, Mj, s, ti) \quad a = b + ((a + H(b, c, d) + Mj + ti) <<< s) \quad (2\text{-}15)$$

$$II(a, b, c, d, Mj, s, ti) \quad a = b + ((a + I(b, c, d) + Mj + ti) <<< s) \quad (2\text{-}16)$$

ทั้ง 4 รอบ (64 ขั้นตอน) เป็นดังนี้

รอบที่ 1

FF(a,b,c,d,M_0,7, 0xd76aa478)

FF(a,b,c,d,M_1,12, 0xe8c7b756)

FF(a,b,c,d,M_2,17, 0x242070db)

FF(a,b,c,d,M_3,22, 0xc1bdceee)

FF(a,b,c,d,M_4,7, 0xf57c0faf)

FF(a,b,c,d,M_5,12, 0x4787c62a)

FF(a,b,c,d,M_6,17, 0xa8304613)

FF(a,b,c,d,M_7,22, 0xfd469501)

FF(a,b,c,d,M_8,7, 0x698098d8)

FF(a,b,c,d,M_9,12, 0x8b44f7af)

FF(a,b,c,d,M_{10},17, 0xffff5bb1)

FF(a,b,c,d,M_{11},22, 0x895cd7be)

FF(a,b,c,d,M_{12},7, 0x6b901122)

FF(a,b,c,d,M_{13},12, 0xfd987193)

FF(a,b,c,d,M_{14},17, 0xa679438e)

FF(a,b,c,d,M_{15},22, 0x49b40821)

รอบที่ 2

GG(a,b,c,d,M_0,5, 0xf61e2562)

GG(a,b,c,d,M_6,9, 0xc040b340)

GG(a,b,c,d,M_{11},14, 0x265e5a51)

GG(a,b,c,d,M_0,20, 0xe9b6c7aa)

GG(a,b,c,d,M_5,5, 0xd62f105d)

GG(a,b,c,d,M_{10},9, 0x2441453)

GG(a,b,c,d,M_{15},14, 0xd8a1e681)

GG(a,b,c,d,M_4,20, 0xe7d3fbc8)

GG(a,b,c,d,M_9,5, 0x21e1cde6)

GG(a,b,c,d,M_{14},9, 0xc33707d6)

GG(a,b,c,d,M_3,14, 0xf4d50d87)

GG(a,b,c,d,M_8,20, 0x455a14ed)

GG(a,b,c,d,M_{13},5, 0xa9e3e905)

GG(a,b,c,d,M_2,9, 0xfcefa3f8)

GG(a,b,c,d,M_7,14, 0x676f02d9)

GG(a,b,c,d,M_{12},20, 0x8d2a4c8a)

รอบที่ 3

HH(a,b,c,d,M_5,4, 0xfffa3942)

HH(a,b,c,d,M_8,11, 0x8771f681)

HH(a,b,c,d,M_{11},16, 0x6d9d6122)

HH(a,b,c,d,M_{14},23, 0xfde5380c)

HH(a,b,c,d,M_1,4, 0xa4beea44)

HH(a,b,c,d,M_4,11, 0x4bdecfa9)

HH(a,b,c,d,M_7,16, 0xf6bb4b60)

HH(a,b,c,d,M_{10},23, 0xbebfbc70)

HH(a,b,c,d,M_{13},4, 0x289b7ec6)

HH(a,b,c,d,M_0,11, 0xeaa127fa)

HH(a,b,c,d,M_3,16, 0xd4ef3085)

HH(a,b,c,d,M_6,23, 0x4881d05)

HH(a,b,c,d,M_9,4, 0xd9d4d039)

HH(a,b,c,d,M_{12},11, 0xe6db99e5)

HH(a,b,c,d,M_{15},16, 0x1fa27cf8)

HH(a,b,c,d,M_2,23, 0xc4ac5665)

รอบที่ 4

II(a,b,c,d,M_0,6, 0xf4292244)

II(a,b,c,d,M_7,10, 0x432aff97)

II(a,b,c,d,M_{14},15, 0xab9423a7)

II(a,b,c,d,M_5,21, 0xfc93a039)

II(a,b,c,d,M_{12},6, 0x655b59c3)

II(a,b,c,d,M_3,10, 0x8f0ccc92)

II(a,b,c,d,M_{10},15, 0xffeff47d)

II(a,b,c,d,M_1,21, 0x85845dd1)

II(a,b,c,d,M_8,6, 0x6fa87e4f)

II(a,b,c,d,M_{15},10, 0xfe2ce6e0)

II(a,b,c,d,M_6,15, 0xa3014314)

II(a,b,c,d,M_{13},21, 0x4e0811a1)

II(a,b,c,d,M_4,6, 0xf7537e82)

II(a,b,c,d,M_{11},10, 0xbd3af235)

II(a,b,c,d,M_2,15, 0x2ad7d2bb)

II(a,b,c,d,M_9,21, 0xeb86d391)

ti เป็นค่าคงที่ มีค่าเท่ากับ จำนวนเต็มของ $2^{32} \times \text{abs}(\sin(i))$ โดย i มีหน่วยเป็นเรเดียน

สุดท้ายค่า a, b, c และ d จะเพิ่มไปยัง A, B, C และ D ตามลำดับ และขั้นตอนวิธีจะดำเนินต่อไปกับบล็อกถัดไป ผลลัพธ์สุดท้ายที่ได้ก็คือ A, B, C และ D ซึ่งนำมาต่อกัน

2.5 โพรโตคอลของวิทยาการเข้ารหัสลับ (Cryptography Protocols)

2.5.1 การพิสูจน์ตัวจริงโดยใช้การเข้ารหัสลับแบบกุญแจสาธารณะ

เป็นโพรโตคอลที่ใช้ในการทำให้ผู้อื่นแน่ใจว่าเป็นผู้ใช้จริงโดยอาศัยการเข้ารหัสลับแบบกุญแจสาธารณะ สำหรับระบบซึ่งประกอบด้วยธนาคารกับผู้ใช้ เริ่มแรกธนาคารจะมีกุญแจสาธารณะของผู้ใช้ทุกคนในระบบ โดยผู้ใช้แต่ละคนจะเก็บกุญแจส่วนตัวของตนเองไว้ เมื่อผู้ใช้ทำการเชื่อมต่อกับธนาคารแล้ว โพรโตคอลมีลำดับขั้นตอนดังนี้

1. ธนาคารทำการส่งข้อความสุ่มค่าหนึ่งไปให้ผู้ใช้
2. ผู้ใช้ทำการเข้ารหัสลับข้อความสุ่มนั้นด้วยกุญแจส่วนตัวของผู้ใช้ และส่งกลับไปให้ธนาคาร พร้อมกับข้อความระบุตัว

3. ธนาคารทำการค้นหากุญแจสาธารณะของผู้ใช้ในฐานข้อมูล และทำการถอดรหัสลับข้อมูลที่ได้รับด้วยกุญแจสาธารณะนั้น
4. ถ้าข้อความที่ถอดรหัสลับมาได้คือข้อความสุ่มเดิม ก็แสดงว่าเป็นผู้ใช้ตัวจริงและธนาคารจึงยอมให้ผู้ใช้เข้าสู่ระบบ

ไม่มีผู้ใดสามารถแอบอ้างเป็นผู้ใช้ได้ถ้าไม่ทราบกุญแจส่วนตัวของผู้ใช้ และที่สำคัญกุญแจส่วนตัวของผู้ใช้ไม่ได้ถูกส่งผ่านเครือข่ายไปยังธนาคาร แม้จะมีผู้ใดดักอ่านข้อมูลที่ส่งระหว่างผู้ใช้และธนาคาร ก็จะไม่สามารถได้ข้อมูลที่เป็นประโยชน์ในการหากุญแจส่วนตัวของผู้ใช้และแอบอ้างเป็นผู้ใช้ได้

2.5.2 Secret Splitting [4]

เป็นโพรโตคอลที่ใช้ในการแบ่งข้อความออกเป็นหลายๆส่วน โดยที่แต่ละส่วนจะไม่บอกอะไร แต่เมื่อนำทุกๆส่วนมารวมกันจะเปิดเผยข้อความเดิม จุดประสงค์ของโพรโตคอลนี้คือต้องการให้ผู้ใช้แต่ละคนเก็บข้อความไว้คนละส่วน โดยที่แต่ละคนไม่สามารถทราบข้อความเดิมได้จากข้อความส่วนของตนเองเพียงส่วนเดียว ตัวอย่างโพรโตคอลที่ง่ายที่สุดของ Secret Splitting ก็คือ การแบ่งข้อความออกเป็น 2 ส่วนสำหรับให้ผู้ใช้ 2 คนเก็บไว้คนละส่วน โดยมีธนาคารทำการแบ่งข้อความให้ มีลำดับขั้นตอนดังนี้

1. ธนาคารสร้างข้อความสุ่ม R ขึ้นมาโดยที่มีความยาวเท่ากับข้อความ M
2. ธนาคารทำการ Exclusive OR ระหว่าง R กับ M ได้ผลลัพธ์เป็น S ดังสมการ (2-17)

$$M \oplus R = S \tag{2-17}$$

3. ธนาคารให้ R กับผู้ใช้คนหนึ่ง และ S กับผู้ใช้อีกคนที่เหลือ
4. ผู้ใช้ทั้ง 2 คนจะสร้างข้อความเดิม M กลับมาได้จากสมการ (2-18)

$$R \oplus S = M \tag{2-18}$$

โพรโตคอลนี้สามารถนำไปใช้กับผู้ใช้มากกว่า 2 คนได้โดยง่าย ยกตัวอย่างเช่น ต้องการแบ่งข้อความให้กับผู้ใช้ 4 คน มีลำดับขั้นตอนดังนี้

1. ธนาคารสร้างข้อความสุ่มขึ้นมา 3 ค่า คือ R, S และ T โดยที่มีความยาวเท่ากับข้อความ M
2. ธนาคารทำการ Exclusive OR ระหว่าง M กับ ข้อความสุ่มทั้ง 3 ค่า ได้ผลลัพธ์เป็น U ดังสมการ (2-19)

$$M \oplus R \oplus S \oplus T = U \tag{2-19}$$

3. ธนาคารให้ R กับคนที่ 1, S กับคนที่ 2, T กับคนที่ 3 และ U กับคนที่ 4

4. ข้อความเดิม M จะสามารถคำนวณกลับมาได้โดยอาศัยข้อความทั้ง 4 คน จากสมการ (2-20)

$$R \oplus S \oplus T \oplus U = M \qquad (2\text{-}20)$$

2.5.3 Bit Commitment [4]

เป็นโพรโตคอลที่ผู้ใช้ต้องผูกมัดตัวเองกับข้อความแต่ไม่ต้องการเปิดเผยข้อความจนกระทั่งระยะเวลาหนึ่งผ่านไป ในทางตรงกันข้ามผู้อื่นก็ต้องการที่จะมั่นใจได้ว่า ผู้ใช้คนนี้จะไม่สามารถเปลี่ยนแปลงข้อความหลังจากที่ผู้ใช้ทำการผูกมัดตัวเองแล้วนี้

โพรโตคอล Bit Commitment โดยอาศัย Symmetric Encryption มีลำดับขั้นตอนดังนี้

1. ผู้ใช้คนที่ 1 ทำการสร้างข้อความสุ่ม R และส่งไปยังผู้ใช้คนที่ 2
2. ผู้ใช้คนที่ 2 สร้างข้อความที่ประกอบด้วยบิตข้อมูล b ที่ผู้ใช้คนที่ 2 ต้องการจะผูกมัดตัวเอง และข้อความสุ่ม R ทำการเข้ารหัสลับด้วยกุญแจลับ K ได้ $E_k(R, b)$ แล้วจึงส่งไปให้กับผู้ใช้คนที่ 1

สำหรับโพรโตคอลส่วนนี้ จะเห็นว่า ผู้ใช้คนที่ 1 จะไม่สามารถถอดรหัสข้อความนี้ได้ ทำให้ไม่ทราบค่า b เมื่อถึงเวลาที่ผู้ใช้คนที่ 2 ต้องเปิดเผย

3. ผู้ใช้คนที่ 2 ส่งกุญแจลับไปให้ผู้ใช้คนที่ 1
4. ผู้ใช้คนที่ 1 ทำการถอดรหัสก็จะทราบ b และตรวจสอบความถูกต้องได้จาก R

โพรโตคอล Bit Commitment โดยอาศัยฟังก์ชันแบบทางเดียว มีลำดับขั้นตอนดังนี้

1. ผู้ใช้สร้างข้อความสุ่ม R1 และ R2
2. ผู้ใช้สร้างข้อความที่ประกอบด้วย R1,R2 และบิตข้อมูล b ที่ต้องการผูกมัด
3. ผู้ใช้คำนวณค่าที่ได้จากฟังก์ชันแบบทางเดียวของข้อความ และส่งไปให้ผู้รับพร้อมกับค่าข้อความสุ่มค่าใดค่าหนึ่ง เช่น H(R1,R2,b),R1 เป็นต้น

ในขั้นตอนนี้ผู้ใช้ผูกมัดตัวเองกับบิตข้อมูล และการใช้ฟังก์ชันแบบทางเดียวในข้อ 3 ก็เพื่อป้องกันไม่ให้ผู้รับคำนวณกลับเพื่อหาค่า b เมื่อถึงเวลาที่ผู้ส่งต้องเปิดเผย

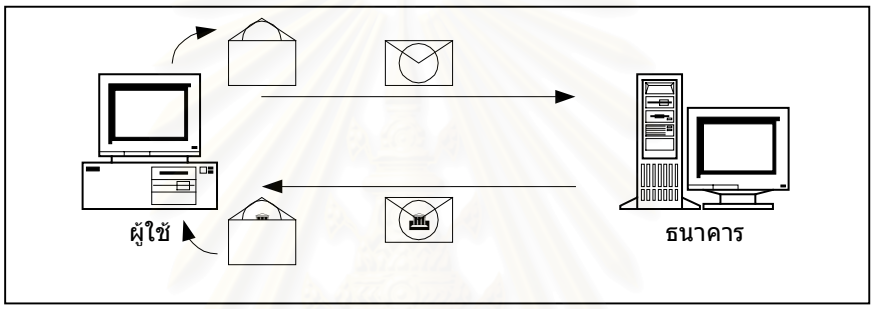
4. ผู้ใช้ส่งข้อความเดิม (R1,R2,b) ไปยังผู้รับ
5. ผู้รับคำนวณค่าฟังก์ชันแบบทางเดียวของข้อความที่ได้รับ และเทียบกับค่าที่ได้กับ R1 ที่ได้รับจากข้อ 3 ถ้าค่าตรงกันแสดงว่าบิตข้อมูลเป็นข้อความที่เก็บเอาไว้จริง

โพรโตคอลแบบหลังนี้ที่อาศัยฟังก์ชันแบบทางเดียวมีข้อดีกว่าโพรโตคอลที่อาศัย Symmetric Encryption คือ ผู้ที่ต้องการผูกมัดตัวเองกับข้อความไม่ต้องอาศัยข้อความสุ่มจากผู้รับ ผู้ใช้เพียงแค่ส่งข้อความที่ผูกมัดกับข้อความสุ่มตัวหนึ่งไปให้ผู้รับ ข้อความสุ่มจากผู้รับไม่มีความจำเป็นเพราะการผูกมัดตัวเองของผู้ใช้โดยอาศัยฟังก์ชันแบบทางเดียว ผู้ใช้ไม่สามารถคำนวณหาข้อ

ความ (R1,R2',b') ซึ่ง H(R1,R2',b') = H(R1,R2,b) เนื่องจากการส่ง R1 ไปยังผู้รับเป็นการผูกมัดตัวเองกับ b ถ้าผู้ใช้ไม่เก็บ R2 เป็นความลับ ผู้รับจะสามารถคำนวณ H(R1,R2,b) และ H(R1,R2,b') จากนั้นทำการเปรียบเทียบกับที่ได้รับจากผู้ส่ง ก็จะทราบบิตข้อมูล b

2.5.4 Blind Signature [9 10 และ 11]

เป็นโพรโตคอลที่ใช้ในการให้ผู้เซ็นรับรองเอกสารไม่เห็นเนื้อหาของเอกสาร ถ้าเปรียบเทียบเอกสารที่ใส่ซองไว้ คือเอกสารที่ไม่ต้องการให้ธนาคารเห็น ผู้ใช้ทำการส่งเอกสารที่ใส่ซองนี้ไปให้ธนาคาร ธนาคารทำการเซ็นเอกสารที่ยังใส่ซองอยู่นี้โดยที่มีกระดาษคาร์บอนอยู่ในซองด้วย ดังนั้นลายเซ็นของธนาคารก็จะติดลงบนเอกสาร จากนั้นจึงส่งให้กับผู้ใช้เปิดซองนำเอกสารออกมาก็จะได้เอกสารที่มีลายเซ็นของธนาคาร ดังรูปที่ 2.11



รูปที่ 2.11 Blind Signature

โพรโตคอล Blind Signature โดยอาศัย Asymmetric Encryption มีลำดับขั้นตอนดังนี้

สมมติว่าผู้ใช้มีข้อความ M ซึ่งเป็นข้อความที่ต้องการให้ธนาคารเซ็น หรือทำการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคาร, (n,e) เป็นกุญแจสาธารณะ, และ d เป็นกุญแจส่วนตัวของธนาคาร

1. ผู้ใช้สร้างข้อความสุ่ม R และคำนวณค่า M' ซึ่งเปรียบเหมือนกับเอกสารที่ใส่ซองแล้ว ดังสมการ (2-14) จากนั้นจึงส่งไปให้ธนาคาร

$$M' = MR^e \bmod n \qquad (2\text{-}21)$$

2. ธนาคารทำการเซ็นซองเอกสารนั้น ได้ค่า S' ซึ่งเปรียบเหมือนกับเอกสารที่ใส่ซองที่ได้รับการเซ็นแล้ว ดังสมการ (2-22) จากนั้นจึงส่งกลับไปให้ผู้ใช้

$$S' = (M')^d = (MR^e)^d \bmod n = RM^d \bmod n \qquad (2\text{-}22)$$

3. ผู้ใช้ก็จะได้ S ซึ่งเปรียบเหมือนกับเอกสารที่ถูกเซ็นแล้วที่ต้องการ โดยการหาร R ออก ดังสมการ (2-23)

$$S = (S')R^{-1} \bmod n = M^d \bmod n \qquad (2\text{-}23)$$

2.5.5 Traceable Anonymous Digital Cash [4 12 13 และ 14]

เป็นโพรโตคอลที่ให้ความเป็นส่วนตัวแก่ผู้ใช้ สามารถตรวจสอบได้ว่าเงินสดดิจิตอลเป็นเงินที่ถูกต้อง คือเงินที่ได้รับเซ็นโดยธนาคาร หรือได้รับการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคารและสามารถนำไปใช้ได้ในระบบ รวมถึงการตามรอยได้ว่าผู้ใช้หรือร้านค้าทุจริตในกรณีมีผู้ใช้เงินสดดิจิตอลซ้ำ โพรโตคอลมีลำดับขั้นตอนดังนี้

1. ผู้ใช้เตรียมใบสั่งเงินจำนวน n ใบสำหรับมูลค่าที่ต้องการ แต่ละใบสั่งเงินจะประกอบไปด้วยข้อความสุ่มที่แตกต่างกัน และข้อความระบุตัวจำนวน n คู่ $(I_1,I_2,\ldots,I_n)$ โดยแต่ละคู่จะสร้างได้ดังนี้

- ผู้ใช้จะสร้างข้อความที่ประกอบด้วย ชื่อ,ที่อยู่ และข้อมูลที่จำเป็นต่างๆที่ธนาคารต้องการ และแบ่งข้อความออกเป็น 2 ส่วนโดยใช้โพรโตคอล secret splitting ยกตัวอย่าง เช่น I_{10} จะประกอบด้วย I_{10L} และ I_{10R} เมื่อนำทั้ง 2 ส่วนมารวมกันจะเปิดเผยข้อความระบุตัวของผู้ใช้ (สำหรับคู่อื่น เช่น I_{10L} กับ I_{9L}จะไม่สามารถเปิดเผยข้อความได้) เป็นต้น
- จากนั้นผู้ใช้ต้องผูกมัดตัวเองกับแต่ละส่วนโดยใช้โพรโตคอล bit-commitment

ใบสั่งเงินจะประกอบด้วยส่วนต่างๆ ดังนี้

- มูลค่าของเงิน
- ข้อความสุ่ม
- ข้อความระบุตัว : $I_1 = (I_{1L}, I_{1R})$

$I_2 = (I_{2L}, I_{2R})$

....

$I_n = (I_{nL}, I_{nR})$

2. ผู้ใช้ทำการปกปิดใบสั่งเงินทั้ง n ใบโดยใช้โพรโตคอล blind signature และส่งใบสั่งเงินที่ปกปิดแล้วทั้งหมดนี้ไปยังธนาคาร
3. ธนาคารจะสั่งให้ผู้ใช้เปิดเผยใบสั่งเงิน n-1 ใบอย่างสุ่ม เพื่อตรวจสอบมูลค่า, ข้อความสุ่ม และ ข้อความระบุตัว ว่ามูลค่า,ข้อความสุ่ม และข้อความระบุตัวเป็นไปตามที่ตกลงหรือไม่ อาจเป็นไปได้ผู้ใช้พยายามโกงโดยใส่มูลค่าเงินไม่ตรงกัน เช่น ผู้ใช้เสี่ยงให้ใบสั่งเงินใบหนึ่งมีมูลค่ามากกว่าใบอื่น ถ้าบังเอิญใบนั้นธนาคารไม่ได้สั่งให้เปิดเผย ผู้ใช้ก็จะได้เงินที่มีมูลค่าเกินกว่าที่จะถูกธนาคารหัก เป็นต้น หรือก็อาจเป็นไปได้อีกเช่นกันที่ผู้ใช้ใส่ข้อความระบุตัวไม่ตรงกับผู้ใช้เอง ดังนั้นโพรโตคอลในขั้นตอนนี้จะช่วยให้สามารถตรวจสอบได้ว่าผู้ใช้พยายามทุจริตดังที่กล่าวมาหรือไม่ โอกาสในการทุจริตสำเร็จจะน้อยลง ถ้าจำนวนใบสั่งเงินที่ปกปิดเพิ่มขึ้น

4. ถ้าธนาคารตรวจสอบแล้วว่าผู้ใช้ไม่ได้พยายามโกง และข้อมูลใบสั่งเงินถูกต้อง ธนาคารจะเซ็นใบสั่งเงินที่เหลืออยู่ 1 ใบที่ยังไม่ได้เปิดเผย จากนั้นก็จะส่งกลับไปให้ผู้ใช้และหักเงินในบัญชีของผู้ใช้ตามมูลค่าใบสั่งเงินนั้น
5. ผู้ใช้ก็จะเปิดเผยใบสั่งเงินที่เซ็นโดยธนาคารออกก็จะได้เงินสดดิจิตอลที่มีลายเซ็นของธนาคารและนำไปใช้กับร้านค้า
6. ร้านค้าจะตรวจสอบลายเซ็นของธนาคารบนเงินสดดิจิตอลที่ผู้ใช้จ่าย เพื่อแน่ใจได้ว่าเงินสดดิจิตอลนี้เป็นเงินจริง โดยการใช้กุญแจสาธารณะของธนาคาร
7. ร้านค้าจะสั่งให้ผู้ใช้เปิดเผยข้อความระบุตัวครึ่งหนึ่ง คือไม่ส่วนซ้ายก็ส่วนขวาในแต่ละคู่ โดยสุ่มเซตของบิตขนาด n ตัวขึ้นมา ($b_1,b_2,\ldots b_n$) และส่งให้ผู้ใช้ เพื่อทำการเปิดเผยข้อความระบุส่วนซ้ายหรือขวาขึ้นกับว่าค่าของ b_i เป็น 0 หรือ 1
8. ผู้ใช้ตอบสนองเปิดเผยข้อความระบุตัวตามเซตของบิตที่ธนาคารส่งมา เนื่องจากโพรโตคอล Bit Commitment ทำให้ผู้ใช้ต้องเปิดเผยข้อความระบุตัวที่ถูกต้อง ไม่สามารถทุจริตโดยการเปิดเผยอย่างไม่ถูกต้องได้
9. ร้านค้านำเงินสดดิจิตอลนี้ส่งให้ธนาคาร
10. ธนาคารตรวจสอบลายเซ็นและฐานข้อมูลเพื่อแน่ใจได้ว่าไม่มีการใช้เงินนี้มาก่อน จากนั้นจะเก็บข้อความสุ่ม และข้อความระบุที่ถูกเปิดเผยมาครึ่งหนึ่งแล้วทั้งหมดลงในฐานข้อมูล
11. ถ้าธนาคารพบว่าข้อความสุ่มของเงินสดดิจิตอลนี้มีอยู่ในฐานข้อมูลแล้ว ธนาคารจะทำการตรวจสอบข้อความระบุตัวของเงินสดดิจิตอลที่ร้านค้าส่งมากับข้อความระบุตัวของเงินสดดิจิตอลที่เก็บในฐานข้อมูล ถ้าข้อความระบุตัวถูกเปิดเผยในครึ่งที่เหมือนกันทุกคู่ แสดงว่า ร้านค้าทำการคัดลอกเงินสดดิจิตอลนั้นและส่งให้ธนาคารอีกที แต่ถ้าข้อความระบุตัวถูกเปิดเผยในครึ่งที่ต่างกันแม้เพียงคู่เดียว แสดงว่า ผู้ใช้ทำการคัดลอกเงินสดดิจิตอลแล้วนำไปใช้อีกครั้ง เนื่องจากขั้นตอนที่ร้านค้าสุ่มเซตของบิตเพื่อให้ผู้ใช้เปิดเผยนั้นแตกต่างกัน ดังนั้นเมื่อนำข้อความระบุตัวครึ่งที่ถูกเปิดเผยในเงินสดดิจิตอลนั้นกับครึ่งที่ถูกเปิดเผยในฐานข้อมูลที่มีลำดับตรงกันมารวมกันก็จะสามารถเปิดเผยได้ว่าผู้ใช้ผู้ใดในระบบเป็นผู้ทุจริต

ผู้ใช้สามารถที่จะทำการคัดลอกเงินสดดิจิตอลไว้ได้ เมื่อนำไปใช้ครั้งแรกโพรโตคอลทุกขั้นตอนก็จะผ่านไปโดยไม่เกิดปัญหา ร้านค้าจะสุ่มเซตของบิตในขั้นตอนที่ 7 และผู้ใช้ก็จะทำการเปิดเผยครึ่งซ้ายหรือขวาของ I_i ในขั้นตอนที่ 8 เมื่อถึงขั้นตอนที่ 10 ธนาคารก็จะทำการเก็บข้อความทั้งหมดนี้ และข้อความสุ่มในฐานข้อมูล แต่เมื่อผู้ใช้นำเงินสดดิจิตอลที่คัดลอกไว้ไปใช้อีกครั้ง ร้านค้าซึ่งอาจจะเป็นร้านค้าเดิมหรือร้านอื่นก็จะสุ่มเซตของบิตที่แตกต่างกันขึ้นมาให้ผู้ใช้ในขั้นตอนที่ 7 ผู้

ใช้ก็ต้องทำการตอบสนอง ถ้าผู้ใช้ไม่ยอมตอบสนองก็จะเป็นการเตือนให้ร้านค้ารู้ว่ามีความผิดปกติเกิดขึ้น เมื่อร้านค้านำเงินสดดิจิตอลนี้ไปขึ้นเงินกับธนาคารในขั้นตอนที่ 10 ธนาคารจะพบว่าเงินสดดิจิตอลนี้มีข้อความสุ่มเหมือนกับเงินสดดิจิตอลในฐานข้อมูล จากนั้นก็จะทำการรวมครึ่งของข้อความระบุตัวที่แตกต่างกัน ซึ่งโอกาสที่ทุกครึ่งจะถูกเปิดเผยเหมือนกันมีแค่ 1 ใน 2n เมื่อทำการ Exclusive OR ครึ่งซ้ายและครึ่งขวาของเงินสดดิจิตอลที่ต่างกัน ธนาคารก็ทราบว่าใครเป็นผู้ทุจริต

ผู้ใช้จะไม่สามารถเปลี่ยนแปลงข้อความระบุตัวหรือข้อความสุ่ม เนื่องจากถูกเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคารดังในขั้นตอนที่ 6 โอกาสที่ผู้ใช้พยายามส่งเงินสดดิจิตอลที่มีข้อความระบุตัวไม่ใช่ผู้ใช้เองหรืออาจเป็นผู้ใช้อื่นไปให้ธนาคารได้มี 1 ใน n ในขั้นตอนที่ 3 ซึ่งธนาคารควรมีการกำหนดโทษสำหรับผู้ที่พยายามทุจริตนี้ที่รุนแรงมากพอ หรือ เพิ่มค่า n ในขั้นตอนที่ 1 ให้มากขึ้น

สำหรับร้านค้ามีโอกาสที่จะทุจริตเป็นไปได้ยากเช่นกัน ธนาคารจะทำการตรวจสอบเซตของบิตที่ใช้ในการเปิดเผยว่าซ้ำกันหรือไม่ ร้านค้าไม่สามารถกล่าวหาผู้ใช้ได้เนื่องจากมีเพียงผู้ใช้เท่านั้นที่สามารถเปิดเผยข้อความระบุครึ่งใดๆได้

ผู้ใช้จะได้ความเป็นส่วนตัวจากโพรโตคอล Blind Signature ในขั้นตอนที่ 2 ถึง 5 ธนาคารจะไม่สามารถตามรอยได้ว่าผู้ใช้เป็นใคร

ถ้าการติดต่อระหว่างผู้ใช้และร้านค้าถูกผู้อื่นดักฟัง และนำเงินดิจิตอลไปขึ้นก่อนร้านค้า ธนาคารก็จะรับ เมื่อร้านค้านำไปขึ้นต่อก็จะถูกกล่าวหาว่าเป็นผู้ทุจริต และถ้ามีผู้ขโมยเงินสดดิจิตอลของผู้ใช้ไปใช้ก่อนผู้ใช้ เมื่อผู้ใช้นำไปใช้อีกครั้ง ผู้ใช้ก็จะถูกกล่าวหาว่าเป็นผู้ทุจริตเช่นกัน ซึ่งไม่มีทางในการป้องกันได้ เนื่องจากผลของความเป็นส่วนตัวที่ได้รับจากเงินสดดิจิตอล ทั้งผู้ใช้และร้านค้าต้องเก็บรักษาเงินสดดิจิตอลนี้เช่นเดียวกับเงินที่ใช้ในชีวิตประจำวัน

บทที่ 3
การออกแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์

เนื้อหาในบทนี้จะกล่าวถึง ภาพรวมและการออกแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ ได้แก่คุณสมบัติของระบบต่างๆ รวมถึงคุณสมบัติที่ขาดไป ส่วนประกอบของระบบ สมมติฐานเบื้องต้นของระบบ การไหลของเงินสดดิจิตอลและโพรโตคอลที่เกิดขึ้นระหว่างผู้ใช้และธนาคาร และสุดท้ายคือการตรวจจับทุจริตหรือใช้เงินสดดิจิตอลซ้ำ รายละเอียดต่างๆอธิบายได้ดังต่อไปนี้

3.1 ภาพรวมของระบบ

โดยทั่วไปการใช้จ่ายเงินสดในชีวิตประจำวัน ผู้ใช้จะได้รับความเป็นส่วนตัว แต่ข้อเสียก็คือเงินสดที่ใช้จะต้องถูกทำขึ้นให้ยากแก่การปลอมแปลง ธนบัตรที่ใช้ถูกทำลายได้ง่าย นอกจากนี้ผู้ใช้ไม่สามารถทำการจ่ายเงินผ่านเครือข่ายอินเทอร์เน็ต การใช้บัตรเครดิตช่วยให้ไม่ต้องพกเงินสดแต่ก็ต้องยอมให้ถูกตามรอยได้ ผู้ใช้จะขาดความเป็นส่วนตัวไป แต่ระบบเงินสดดิจิตอลจะสามารถแก้ปัญหาที่กล่าวมานี้ได้

ระบบการจ่ายเงินแบบอิเล็กทรอนิกส์สามารถแบ่งตามการใช้งานได้ 2 ประเภท คือ ระบบแบบเครดิต (Credit-based) และระบบแบบเงินสด (Cash-based) ซึ่งต้นแบบของระบบในวิทยานิพนธ์นี้จัดเป็นระบบแบบเงินสด ในกรณีที่มีไม่มีการทุจริตใช้เงินสดดิจิตอลซ้ำ ระบบจะให้ความเป็นส่วนตัวแก่ผู้ใช้ได้เนื่องจากเงินสดดิจิตอลที่ใช้จะไม่ข้อมูลส่วนใดที่สามารถจะถูกเปิดเผยโดยธนาคารหรือผู้ใช้อื่นเพื่อระบุตัวผู้ใช้เงินสดดิจิตอลนี้ได้ เหมือนกับเงินที่ใช้ในชีวิตประจำวันที่ไม่สามารถระบุได้ว่าเป็นเงินของใคร ในขณะที่ระบบแบบเครดิตหรือเดบิตไม่สามารถให้ความเป็นส่วนตัวนี้ได้ ต้นแบบระบบถูกพัฒนาขึ้นเพื่อทำงานด้วยซอฟต์แวร์ล้วนบนเครื่องคอมพิวเตอร์ส่วนบุคคลทั่วไป ไม่ต้องอาศัยฮาร์ดแวร์เฉพาะเหมือนกับ Electronic Purse ที่ใช้บัตรสมาร์ตเป็นกระเป๋าพกเงิน และต้องการฮาร์ดแวร์เฉพาะในการอ่านข้อมูลจากบัตร ตัวอย่างระบบ เช่น Mondex ซึ่งถูกพัฒนาโดยธนาคาร NatWest [15] เป็นต้น ระบบจะทำงานแบบออฟไลน์ คือการจ่ายเงิน/ทอนเงินสดดิจิตอลระหว่างผู้ใช้ไม่จำเป็นต้องติดต่อไปยังธนาคารขณะทำงาน เป็นที่แน่นอนว่าระบบจะไม่สามารถป้องกันการใช้เงินซ้ำในระหว่างการจ่ายเงินได้ แต่ระบบสามารถตามรอยผู้ทุจริตได้เมื่อเงินสดดิจิตอลนั้นถูกนำมาขึ้นเงินกับธนาคาร เงินสดดิจิตอลสามารถถูกตรวจสอบได้ว่าเป็นเงินที่ได้รับการเซ็นจากธนาคารจริงโดยอาศัยกุญแจสาธารณะของธนาคาร ทำให้ไม่มีผู้ใดสามารถทำการแก้ไขข้อมูลของเงินสดดิจิตอลที่ได้รับการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคารนี้ได้ นอกจากนี้เงินสดดิจิตอลที่ใช้ในระบบนี้จะให้ความเป็นส่วนตัวแก่ผู้ใช้เมื่อมีการใช้งานโดยไม่มีการทุจริตใช้เงินซ้ำ แต่ในกรณีที่มีการทุจริตใช้เงินสดดิจิตอลซ้ำ ระบบจะสามารถตามรอย

หรือระบุได้ว่าผู้ใช้คนใดเป็นผู้ทุจริตเมื่อนำเงินสดดิจิตอลมาขึ้นเงินกับธนาคาร เนื่องจากเงินสดดิจิตอลจะมีข้อความระบุตัวของผู้ใช้ที่ซื้อเงินนั้นจากธนาคาร แต่ข้อความระบุตัวจะถูกทำการเข้ารหัสลับไว้ ในการใช้เงินสดดิจิตอลผู้ใช้จะต้องทำการเปิดเผยข้อความระบุตัวครึ่งหนึ่ง ดังนั้นถ้ามีการใช้เงินสดดิจิตอลซ้ำ ธนาคารจะสามารถตามรอยได้หรือระบุตัวผู้ใช้ได้จากข้อความระบุตัวทั้งสองครึ่งนั้น [16]

3.2 คุณสมบัติของระบบ

3.2.1 ความเป็นอิสระทางกายภาพ (Independence)

ความปลอดภัยของระบบจะไม่ขึ้นอยู่กับตำแหน่งทางกายภาพ เงินสดดิจิตอลสามารถถูกส่งเครือข่ายคอมพิวเตอร์ทั่วไป

3.2.2 ความปลอดภัย (Security)

เงินสดดิจิตอลจะถูกทำการเข้ารหัสลับด้วยขั้นตอนวิธีของวิทยาการเข้ารหัสลับ ทำให้ไม่สามารถทำการแก้ไขข้อมูลของเงิน หรือคัดลอกเพื่อนำมาใช้ซ้ำได้อีก ความปลอดภัยของระบบจะขึ้นอยู่กับความแข็งแรงของขั้นตอนวิธีที่ใช้ในการเข้ารหัสลับ ในต้นแบบระบบนี้ใช้ขั้นตอนวิธีต่างๆที่เป็นมาตรฐานสากล เช่น RSA, DES และ MD5 เป็นต้น

3.2.3 ความเป็นส่วนตัว (Privacy)

ระบบจะให้ความเป็นส่วนตัวแก่ผู้ใช้เงิน ถ้าไม่มีการทุจริตใช้เงินสดดิจิตอลซ้ำ ข้อมูลของเงินสดดิจิตอลจะไม่สามารถตามรอยได้ว่าเป็นเงินของผู้ใช้คนใด ไม่ว่าจะเป็นธนาคารหรือผู้ใช้อื่น เนื่องจากข้อมูลเงินสดดิจิตอลจะถูกเข้ารหัสลับไว้ด้วยโพรโตคอล Blind Signature

3.2.4 การจ่ายเงินแบบออฟไลน์ (Off-line Payment)

เมื่อผู้ใช้จ่ายด้วยเงินสดดิจิตอล โพรโตคอลที่เกิดขึ้นระหว่างผู้จ่ายเงินกับผู้รับเงินจะเป็นแบบออฟไลน์ คือผู้รับเงินไม่จำเป็นต้องติดต่อไปยังธนาคารเพื่อประมวลผลในขั้นตอนจ่ายเงินนี้ ผู้จ่ายเงินและผู้รับเงินสามารถทำการตรวจสอบได้ว่าเป็นเงินสดดิจิตอลที่ถูกต้องจริงโดยอาศัยกุญแจสาธารณะของธนาคาร ต้นแบบระบบนี้ ผู้รับเงินหรือร้านค้าทำการทอนเงินได้ด้วย ในกรณีที่ร้านค้าไม่มีเงินทอนหรือเงินทอนไม่พอ ก็จะแจ้งให้ผู้จ่ายทราบก่อนจะยืนยันการจ่ายเงิน ทำให้ผู้จ่ายเงินสามารถเลือกได้ว่าพอใจกับเงินทอนที่ร้านค้ามีให้หรือไม่ก่อนทำการยืนยันการจ่ายเงินไปยังร้านค้า

3.2.5 **การตามรอย** (Traceable)

แม้ว่าระบบจะให้ความเป็นส่วนตัว แต่ถ้ามีการทุจริตใช้เงินสดดิจิตอลซ้ำโดยผู้ใช้ ระบบจะสามารถตามรอยได้ว่าผู้ใช้คนใดเป็นคนทุจริต เนื่องจากข้อมูลระบุตัวจะถูกแบ่งเป็น 2 ส่วนโดยโพรโตคอล Secret Splitting และทำการเข้ารหัสลับด้วยโพรโตคอล Bit Commitment เมื่อผู้ใช้นำเงินดิจิตอลไปใช้จะต้องทำการเปิดเผยข้อมูลครึ่งหนึ่งที่เข้ารหัสลับไว้ ถ้าผู้ใช้ทำการคัดลอกเงินและนำไปใช้ซ้ำ เงินสดดิจิตอลจะถูกเปิดเผยข้อมูลอีกครึ่งหนึ่ง ทำให้ธนาคารสามารถตามรอยหรือระบุตัวผู้ใช้ได้ แต่ถ้าร้านค้านำไปใช้ซ้ำ เงินสดดิจิตอลที่ใช้ซ้ำจะมีข้อมูลครึ่งที่ถูกเปิดเผยเหมือนกัน ธนาคารก็จะทราบว่าร้านค้าทุจริตใช้เงินซ้ำ

3.2.6 **คุณสมบัติที่ขาดไปในระบบ**

ระบบจะขาดคุณสมบัติ การถ่ายโอน (Transferability) คือเงินสดดิจิตอลที่ถูกใช้แล้วจะไม่สามารถนำไปใช้จ่ายให้ผู้อื่นอีกได้ เงินสดดิจิตอลจะต้องส่งไปยังธนาคารเพื่อตรวจสอบว่าเป็นเงินที่ธนาคารเซ็นจริงหรือไม่ มีการทุจริตใช้เงินซ้ำหรือไม่ และเงินหมดอายุการใช้งานหรือยัง ถ้าทุกอย่างถูกต้อง ธนาคารก็จะทำการเพิ่มยอดเงินในบัญชีของเจ้าของเงินนั้น

3.3 **ส่วนประกอบของระบบ**

3.3.1 **ผู้ใช้งานระบบ**

3.3.1.1 **ธนาคาร**

ธนาคารมีหน้าที่หลักในระบบคือ การให้บริการซื้อและคืนเงินสดดิจิตอล นอกจากนี้ก็จะมีหน้าที่ในการพิสูจน์ตัวจริงของผู้ใช้ และตรวจสอบเงินสดดิจิตอลด้วยว่าถูกต้องหรือมีการใช้ซ้ำหรือไม่ โดยธนาคารจะมีฐานข้อมูลซึ่งใช้เก็บข้อมูลของผู้ใช้ และข้อมูลของเงินสดดิจิตอล เพื่อตรวจสอบการใช้เงินซ้ำ รวมถึงการตามรอยได้ว่าผู้ใช้ในระบบคนใดทุจริตใช้เงินซ้ำ

3.3.1.2 **ผู้ใช้**

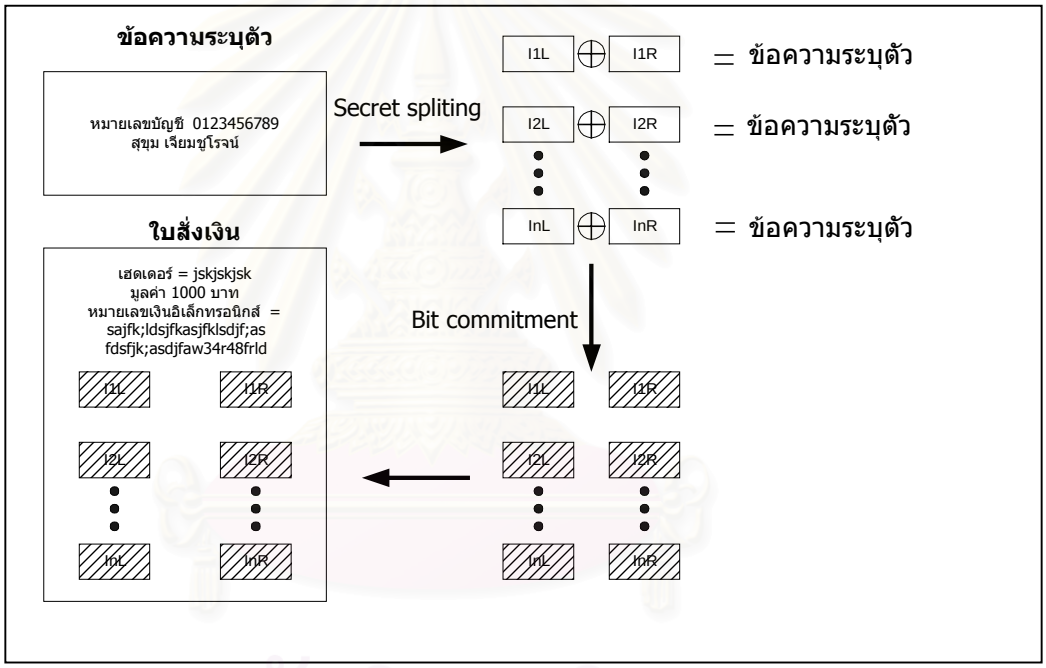
ผู้ใช้มีหน้าที่หลักในระบบคือ การซื้อเงินสดดิจิตอลจากธนาคาร การคืนเงินที่ยังไม่ได้ใช้ให้กับธนาคาร การจ่ายเงิน/ทอนเงินให้กับผู้ใช้อื่น และการขึ้นเงินที่ใช้แล้วกับธนาคาร

3.3.2 **เงินสดดิจิตอล** [17]

เงินสดดิจิตอลในระบบนี้ คือแฟ้มข้อมูลที่ผ่านโพรโตคอลของวิทยาการเข้ารหัสลับและมีรูปแบบตรงกับที่กำหนดตามการออกแบบ เงินสดดิจิตอลจะได้จากโพรโตคอลการซื้อเงินที่เกิดขึ้นระหว่างผู้ใช้กับธนาคาร เงินสดดิจิตอลนี้จะเก็บข้อมูลผู้ใช้ที่เข้ารหัสลับไว้อยู่ด้วย ข้อมูลระบุตัวของ

ผู้ใช้ที่เข้ารหัสลับไว้จะแบ่งเป็น 2 ส่วน ถ้าข้อความระบุตัวถูกเปิดเผยทั้ง 2 ส่วน จะสามารถระบุเจ้าของเงินได้

เงินสดดิจิตอลจะถูกสร้างจากใบสั่งเงิน ซึ่งขั้นตอนการสร้างใบสั่งเงินเป็นดังนี้ เริ่มต้นจากข้อความระบุตัว ซึ่งอาจจะประกอบด้วยหมายเลขบัญชี ชื่อและนามสกุล เป็นต้นจะผ่านโพรโตคอล Secret Splitting ทำให้ข้อมูลถูกแบ่งเป็น 2 ส่วนด้วยกัน คือ เซตของส่วนซ้าย $\{I_{1L},I_{2L},\ldots,I_{nL}\}$ และเซตของส่วนขวา $\{I_{1R},I_{2R},\ldots,I_{nR}\}$ โดยถ้านำสมาชิกของเซตของส่วนซ้ายและสมาชิกของเซตของส่วนขวาลำดับตรงกันมา Exclusive OR กันก็จะได้ข้อความระบุตัว เช่น I_{1L} XOR I_{1R} เป็นต้น จากนั้นข้อมูลแต่ละส่วนจะถูกนำมาผ่านโพรโตคอล Bit Commitment ชุดของข้อมูลที่เป็นผลลัพธ์นี้จะนำไปรวมกับ เฮดเดอร์ มูลค่าของเงิน และหมายเลขของเงินสดดิจิตอล ก็จะได้ใบสั่งเงินมา ลักษณะการสร้างใบสั่งเงินเป็นดังรูป 3.1



รูปที่ 3.1 ขั้นตอนการสร้างใบสั่งเงิน

จากนั้นเมื่อผู้ใช้ทำการซื้อเงินจากธนาคารโดยผ่านโพรโตคอล Blind Signature ใบสั่งเงินจะถูกเซ็นหรือได้รับการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคาร ทำให้ได้เป็นเงินสดดิจิตอลที่ถูกต้องนำไปใช้ได้ เงินสดดิจิตอลในระบบแบ่งเป็น 2 ประเภทดังนี้

3.3.2.1 เงินสดดิจิตอลที่ยังไม่ได้ใช้

เงินสดดิจิตอลที่ยังไม่ได้ใช้ คือ เงินสดดิจิตอลที่ยังไม่ได้เปิดเผยข้อความระบุตัวที่เข้ารหัสลับไว้ ผู้ใช้สามารถที่จะนำไปใช้จ่ายเงิน/ทอนเงินกับผู้ใช้อื่น หรือคืนเงินกับธนาคาร เงินสดดิจิตอล

ที่ยังไม่ได้ใช้จะถูกเข้ารหัสลับด้วย DES ทำให้เฉพาะผู้ที่มีรหัสผ่านของผู้ใช้เท่านั้นถึงสามารถนำเงินนี้ไปใช้ได้

3.3.2.2 **เงินสดดิจิตอลที่ใช้แล้ว**

เงินสดดิจิตอลที่ยังใช้แล้ว คือ เงินสดดิจิตอลที่ถูกเปิดเผยข้อความระบุตัวที่เข้ารหัสลับไว้แล้วครึ่งหนึ่ง ผู้ใช้สามารถนำไปใช้ขึ้นเงินกับธนาคาร เพื่อหักเข้าบัญชีของตนเองได้

3.3.3 **ฐานข้อมูล**

ฐานข้อมูลจะอยู่ที่ธนาคาร โดยจะเก็บข้อมูลของผู้ใช้, ข้อมูลของเงินสดดิจิตอลที่ใช้แล้ว และวันหมดอายุของเงินสดดิจิตอล

3.3.3.1 **ข้อมูลของผู้ใช้**

ประกอบด้วยหมายเลขบัญชี, ชื่อและนามสกุล, ยอดเงินในบัญชี, กุญแจสาธารณะของผู้ใช้ และข้อมูลระบุตัวผู้ใช้ เป็นต้น

3.3.3.2 **ข้อมูลของเงินที่ใช้แล้ว**

ประกอบด้วยชุดหมายเลขของเงินสดดิจิตอล, มูลค่าของเงิน, วันหมดอายุ, ชุดของส่วนที่เป็นความลับครึ่งหนึ่งของเงินสดดิจิตอล เป็นต้น

3.3.3.3 **วันหมดอายุของเงินสดดิจิตอล**

เนื่องจากขนาดของฐานข้อมูลจะใหญ่มากขึ้นเรื่อยๆ จึงต้องมีวันหมดอายุของเงินสดดิจิตอล เพื่อใช้ในการลบข้อมูลของเงินสดดิจิตอลที่ใช้แล้วในฐานข้อมูลออก

3.3.4 **ซอฟต์แวร์**

ซอฟต์แวร์เป็นส่วนติดต่อระหว่างผู้ใช้งานกับระบบ แบ่งเป็นส่วนของธนาคารและส่วนของผู้ใช้ โดยซอฟต์แวร์ทั้ง 2 ส่วนนี้ มีฟังก์ชันการติดต่อระหว่างกันเหมือนกัน แต่ฟังก์ชันอื่นจะต่างกันดังนี้

3.3.4.1 **ส่วนของธนาคาร**

ซอฟต์แวร์ในส่วนนี้จะติดต่อกับฐานข้อมูลได้ เพื่อใช้ในการบันทึก แก้ไขข้อมูลและนำข้อมูลมาเปรียบเทียบตรวจสอบความถูกต้องของเงินสดดิจิตอล นอกจากนี้ยังใช้ในการตามรอยผู้ทุจริตใช้เงินซ้ำ

3.3.4.2 **ส่วนของผู้ใช้**

ซอฟต์แวร์ของผู้ใช้จะมีฟังก์ชันหลัก คือการซื้อเงิน คืนเงิน จ่าย/ทอนเงิน และขึ้นเงิน นอกจากนี้จะฟังก์ชันการดูเงินสดดิจิตอลที่ยังไม่ได้ใช้ และเงินสดดิจิตอลที่ใช้แล้ว

3.4 **สมมติฐานเบื้องต้นของระบบ**

ผู้ใช้จะต้องเปิดบัญชีเงินฝากกับธนาคารไว้ โดยธนาคารจะเก็บข้อมูลของผู้ใช้ ส่วนผู้ใช้ก็จะต้องเก็บกุญแจสาธารณะของธนาคาร กุญแจสาธารณะและกุญแจส่วนตัวของผู้ใช้ และข้อมูลส่วนตัวไว้ ในส่วนของฐานข้อมูลของธนาคารก็จะมีการกำหนดวันหมดอายุของเงินไว้ เพื่อใช้ในการลบข้อมูลและจำกัดขนาดของฐานข้อมูล

3.5 **การไหลของเงินสดดิจิตอล** [18]

3.5.1 **การซื้อเงิน**

ผู้ใช้สามารถเปลี่ยนเงินในบัญชีของตนเองเป็นเงินสดดิจิตอล โดยฟังก์ชันการซื้อเงิน ซึ่งในขั้นตอนนี้ธนาคารจะทำการพิสูจน์ตัวจริงกับผู้ใช้ในตอนเริ่มต้นการเชื่อมต่อ ถ้าถูกตรวจสอบว่าเป็นผู้ใช้จริงในระบบ ผู้ใช้ก็จะสามารถระบุจำนวนเงินที่ต้องการซื้อ จากนั้นเงินสดดิจิตอลจะถูกส่งมายังผู้ใช้ และธนาคารก็จะหักเงินในบัญชีออกไปตามจำนวนเงินที่ระบุ

3.5.2 **การคืนเงิน**

เป็นไปได้ที่เงินสดดิจิตอลที่อยู่กับผู้ใช้ยังไม่ได้ใช้ และผู้ใช้ต้องการคืนเงินเนื่องจากเงินอาจจะใกล้หมดอายุ ผู้ใช้สามารถเชื่อมต่อไปยังธนาคารและใช้ฟังก์ชันคืนเงินได้ ถ้าเงินสดดิจิตอลถูกตรวจสอบแล้วว่าถูกต้อง ธนาคารก็จะเพิ่มเงินในบัญชีให้กับผู้ใช้ตามมูลค่าของเงินนั้น

3.5.3 **การจ่ายเงิน/การทอนเงิน**

เมื่อผู้ใช้ต้องการจ่ายเงินไปยังร้านค้าหรือผู้อื่น ผู้ใช้สามารถเชื่อมต่อไปยังร้านค้าและใช้ฟังก์ชันการจ่ายเงิน/ทอนเงิน โดยผู้ใช้จะสามารถเลือกเซตของเงินสดดิจิตอลที่ต้องการจ่ายได้ ในกรณีที่จ่ายไปมากกว่ามูลค่าที่ต้องการจ่ายจริงเนื่องจากไม่มีเศษ ฝั่งร้านค้าจะหาเงินสดดิจิตอลของตนเป็นเศษทอนให้กับผู้จ่าย และบอกไปยังผู้จ่ายว่ามีเศษทอนหรือไม่ และเท่าไร ถ้าผู้จ่ายตกลงก็จะทำการจ่ายเงิน/ทอนเงินกัน เงินสดดิจิตอลที่ถูกใช้ในการจ่ายเงิน/ทอนเงินแล้วนี้จะถูกเปิดเผยข้อมูลผู้ใช้ที่เข้ารหัสลับไว้แล้วครึ่งหนึ่งด้วย

3.5.4 การขึ้นเงิน

ผู้ใช้สามารถนำเงินสดดิจิตอลที่ใช้แล้วหักเข้าบัญชีตนเองได้โดยการเชื่อมต่อไปยังธนาคาร และใช้ฟังก์ชันการขึ้นเงิน เงินสดดิจิตอลจะถูกตรวจสอบว่าถูกต้องหรือไม่ มีการใช้ซ้ำหรือไม่ ถ้าทุกอย่างถูกต้องยอดเงินรวมก็จะถูกเพิ่มในบัญชีของผู้ใช้

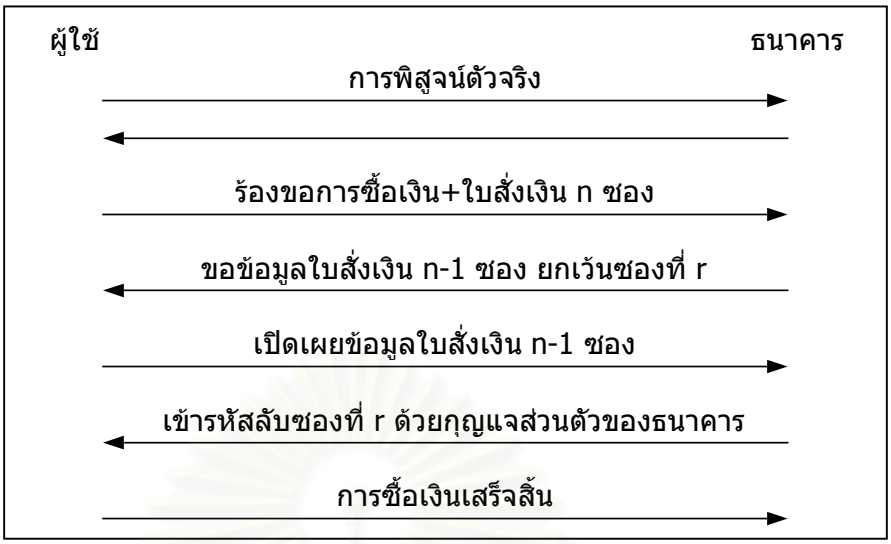
3.6 โพรโตคอลของการไหลของเงิน [18 และ 19]

โพรโตคอลของการไหลของเงินจะเกิดขึ้นได้ ก็ต่อเมื่อผู้ใช้ทำการเชื่อมต่อไปยังผู้ใช้อื่น หรือธนาคารเสร็จสิ้นแล้ว โดยโพรโตคอลต่างๆจะเกิดขึ้นตามฟังก์ชันในการเรียกใช้งานของผู้ใช้ดังนี้

3.6.1 การซื้อเงิน

โพรโตคอลของการซื้อเงินสดดิจิตอลนี้เกิดขึ้นระหว่างผู้ใช้กับธนาคาร เมื่อผู้ใช้ต้องการซื้อเงินสดดิจิตอลกับธนาคาร มีขั้นตอนดังนี้

1. ขั้นตอนการพิสูจน์ตัวจริง โดยผู้ใช้ทำการพิสูจน์ตัวจริงไปยังธนาคาร เพื่อให้ธนาคารทราบว่าเป็นผู้ใช้จริง
2. ขั้นตอนการซื้อเงินสดดิจิตอล โดยอาศัย Blind Signature
   2.1 ผู้ใช้ร้องขอการซื้อเงิน พร้อมทั้งส่งใบสั่งเงินที่ทำการปกปิดไว้แล้ว n ซอง ด้วยโพรโตคอล Secret Splitting และโพรโตคอล Bit Commitment จากนั้นส่งไปยังธนาคาร
   2.2 ธนาคารจะขอข้อมูลใบสั่งเงิน n-1 ซอง ยกเว้นซองที่ r
   2.3 ผู้ใช้ทำการเปิดเผยข้อมูลใบสั่งเงินทั้ง n-1 ซองที่ธนาคารต้องการ
   2.4 ธนาคารทำการตรวจสอบข้อมูลทั้ง n-1 ถ้าถูกต้อง ก็จะทำการเซ็นหรือเข้ารหัสลับซองที่ r ด้วยกุญแจส่วนตัวของธนาคาร
   2.5 ผู้ใช้นำเงินสดดิจิตอลออกจากซองหรือ Unblind ซองที่ r ก็จะได้เงินสดดิจิตอลที่ไปใช้
3. ธนาคารทำการหักเงินในบัญชีของผู้ใช้
4. เงินสดดิจิตอลที่ผู้ใช้ได้มาจะถูกเข้ารหัสลับด้วย DES โหมด CBC ทำให้เงินสดดิจิตอลที่ยังไม่ได้ใช้จะสามารถถูกนำมาใช้ได้เฉพาะผู้ที่มีรหัสผ่านของผู้ใช้เท่านั้น

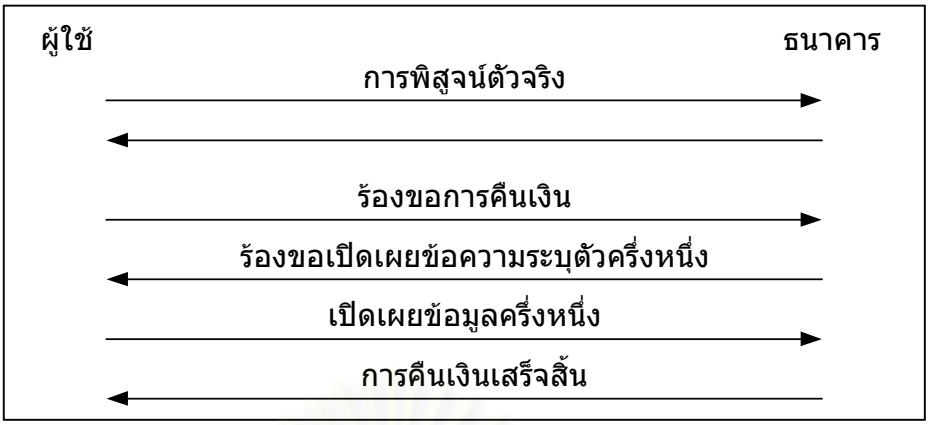


รูปที่ 3.2 โพรโตคอลการซื้อเงิน

3.6.2 การคืนเงิน

โพรโตคอลของการคืนเงินสดดิจิตอลนี้เกิดขึ้นระหว่างผู้ใช้กับธนาคาร เมื่อผู้ใช้ต้องการคืนเงินสดดิจิตอลกับธนาคาร อาจเนื่องจากเงินใกล้หมดอายุ ธนาคารจะทำการพิสูจน์ตัวจริง จากนั้นผู้ใช้จึงเลือกเงินสดดิจิตอลที่ต้องการจะคืน จากนั้นธนาคารจะร้องขอให้ผู้ใช้เปิดเผยข้อมูลระบุตัวผู้ใช้ครึ่งหนึ่ง ผู้ใช้ก็ทำงานส่งข้อมูลนั้นไปยังธนาคาร ธนาคารจะทำการตรวจสอบเงินสดดิจิตอลว่าถูกต้องและมีการใช้ซ้ำหรือไม่ ถ้าไม่ตรวจพบการทุจริต ผู้ใช้ส่งยอดเงินในบัญชีของผู้ใช้จะถูกเพิ่มไปตามมูลค่าของเงินสดดิจิตอลที่คืนมา ลำดับขั้นตอนที่เกิดขึ้นจะเป็นดังนี้

1. ขั้นตอนการพิสูจน์ตัวจริง โดยผู้ใช้ทำการพิสูจน์ตัวจริงไปยังธนาคาร เพื่อให้ธนาคารทราบว่าเป็นผู้ใช้จริง
2. ผู้ใช้ร้องขอการคืนเงินสดดิจิตอลที่ยังไม่ได้ใช้ พร้อมส่งเงินสดดิจิตอลที่ต้องการคืนไปยังธนาคาร
3. ธนาคารร้องขอให้เปิดเผยข้อความระบุตัวครึ่งหนึ่งโดยการสุ่มตัวเลข 0 หรือ 1 เพื่อให้เปิดเผยส่วนซ้ายหรือส่วนขวา จำนวนเท่ากับจำนวนคู่ของข้อความระบุตัว
4. ผู้ใช้เปิดเผยข้อความระบุตัวครึ่งที่ธนาคารต้องการ
5. ธนาคารทำการหักเงินให้กับบัญชีของผู้ใช้



รูปที่ 3.3 โพรโตคอลการคืนเงิน

3.6.3 การจ่ายเงิน/การทอนเงิน

โพรโตคอลของการซื้อเงินสดดิจิตอลนี้เกิดขึ้นระหว่างผู้จ่ายกับผู้รับ เมื่อผู้ใช้ต้องการจ่ายอิเล็กทรอนิกส์ไปยังร้านค้าหรือผู้ใช้อื่น ผู้รับจะทำการตรวจสอบเงินว่าถูกต้องหรือไม่ จากนั้นผู้จ่ายต้องทำการเปิดเผยข้อมูลผู้ใช้ที่เข้ารหัสลับไว้แล้วครึ่งหนึ่ง เงินสดดิจิตอลที่ถูกส่งมายังผู้รับก็จะเป็นเงินสดดิจิตอลที่ใช้แล้ว ในกรณีที่ผู้ใช้ต้องการจ่ายเงินแบบต้องการเงินทอน ทางร้านค้าหรือผู้รับจะทำการทอนเงินสดดิจิตอลที่ผู้จ่ายต้องการให้ใกล้เคียงมูลค่าที่ต้องทอนจริงมากที่สุด และถามการยืนยันไปยังผู้จ่ายอีกครั้ง ถ้าผู้จ่ายตกลง โพรโตคอลของการทอนเงินจากร้านค้าไปยังผู้จ่ายก็จะเหมือนกับโพรโตคอลของการจ่ายเงินจากผู้จ่ายไปยังร้านค้าหรือผู้รับ ลำดับขั้นตอนที่เกิดขึ้นจะเป็นดังนี้

1. ผู้ใช้ทำการร้องการจ่ายเงินไปยังร้านค้า โดยเลือกจำนวนเงินสดดิจิตอลที่ต้องการจ่ายพร้อมระบุจำนวนเงินที่ต้องการจ่ายจริง
2. ร้านค้าส่งข้อมูลเงินทอนให้กับผู้ใช้
3. ถ้าผู้ใช้พอใจกับเงินทอนที่ร้านค้ามีให้หรือไม่ต้องการเงินทอนก็ทำการยืนยันการจ่ายเงินนั้น
4. ร้านค้าร้องขอให้เปิดเผยข้อความระบุตัวครึ่งหนึ่งโดยการสุ่มตัวเลข 0 หรือ 1 เพื่อให้เปิดเผยส่วนซ้ายหรือส่วนขวา จำนวนเท่ากับจำนวนคู่ของข้อความระบุตัว
5. ผู้ใช้เปิดเผยข้อความระบุตัวครึ่งที่ธนาคารต้องการ
6. ถ้าไม่มีการทอนเงิน การจ่ายเงินเสร็จสิ้น ถ้ามีร้านค้าจะทำการร้องขอการจ่ายเงินไปยังผู้ใช้

7. การทอนเงินจะซ้ำกับขั้นตอนการจ่ายเงินของผู้ใช้ไปยังร้านค้า แต่ผู้ใช้จะเป็นฝ่ายร้องขอให้เปิดเผยข้อความระบุตัวครึ่งหนึ่ง และร้านค้าทำการเปิดเผยเงินสดดิจิตอลของตนเองให้กับผู้ใช้แทน

รูปที่ 3.4 โพรโตคอลการจ่ายเงิน/ทอนเงิน

3.6.4 การขึ้นเงิน

โพรโตคอลของการขึ้นเงินสดดิจิตอลนี้เกิดขึ้นระหว่างผู้ใช้กับธนาคาร เมื่อผู้ใช้ต้องการขึ้นเงินสดดิจิตอลกับธนาคาร ธนาคารจะทำการพิสูจน์ตัวจริง จากนั้นเงินสดดิจิตอลที่ใช้แล้วของผู้ใช้ก็จะถูกส่งไปยังธนาคาร เพื่อตรวจสอบความถูกต้องและการใช้ซ้ำ ถ้าไม่ตรวจพบการทุจริต ยอดเงินในบัญชีของผู้ใช้จะถูกเพิ่มไปตามมูลค่าของเงินสดดิจิตอลที่ใช้แล้วให้กับผู้ใช้ ลำดับขั้นตอนที่เกิดขึ้นจะเป็นดังนี้

1. ขั้นตอนการพิสูจน์ตัวจริง โดยผู้ใช้ทำการพิสูจน์ตัวจริงไปยังธนาคาร เพื่อให้ธนาคารทราบว่าเป็นผู้ใช้จริง
2. ผู้ใช้ร้องขอการขึ้นเงิน เงินสดดิจิตอลที่ใช้แล้วทั้งหมดจะถูกส่งไปยังธนาคาร
3. ธนาคารทำการตรวจสอบ ถ้าทุกอย่างถูกต้องก็จะทำการหักเงินให้กับบัญชีผู้ใช้

รูปที่ 3.5 โพรโตคอลการขึ้นเงิน

3.7 การตรวจจับทุจริตหรือใช้เงินสดดิจิตอลซ้ำ

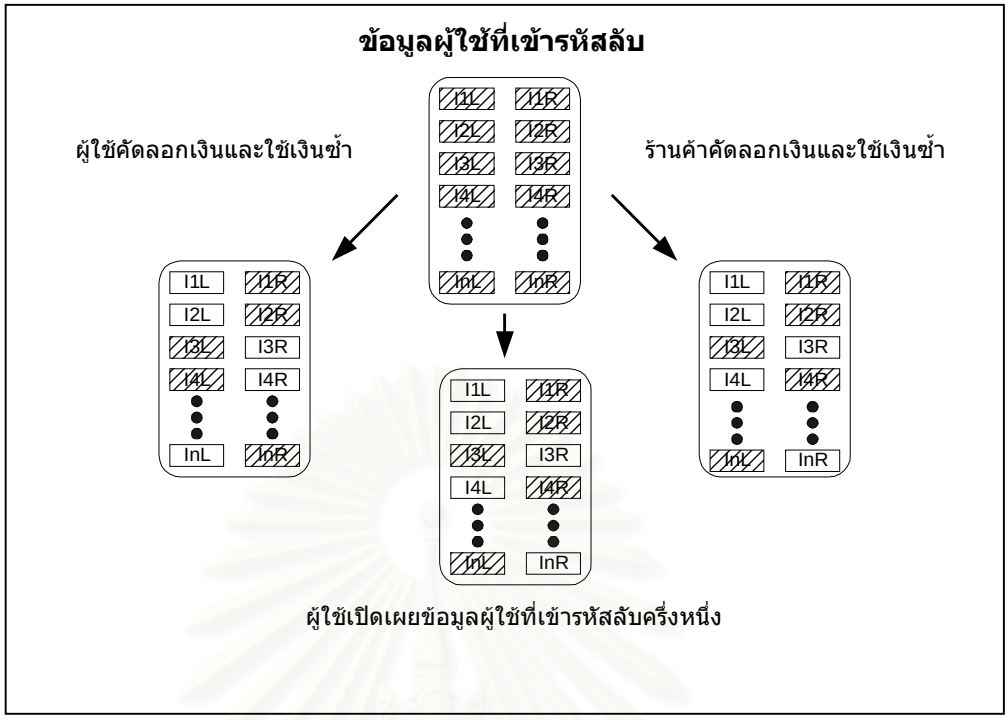
3.7.1 การตรวจสอบเงินสดดิจิตอล

เงินสดดิจิตอลจริงจะถูกเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคาร ซึ่งสามารถตรวจสอบได้โดยใช้กุญแจสาธารณะของธนาคาร ข้อมูลที่ถอดรหัสได้ ส่วนแรกจะถูกนำมาตรวจสอบเฮดเดอร์ว่าตรงกับที่กำหนดไว้หรือไม่ ถ้าตรงกัน แสดงว่าเป็นเงินสดดิจิตอลจริง ส่วนอื่นจะแสดงมูลค่าของเงิน ,หมายเลขของเงิน และข้อความระบุตัวที่ถูกแบ่งเป็น 2 ครึ่งด้วยโพรโตคอล Secret Splitting และเข้ารหัสลับไว้ด้วยโพรโตคอล Bit Commitment

3.7.2 การตามรอยเมื่อเกิดการใช้เงินสดดิจิตอลซ้ำ [8 และ 11]

ธนาคารจะสามารถทำการตรวจสอบการใช้เงินสดดิจิตอลซ้ำได้ และสามารถตามรอยได้ว่าผู้ใช้หรือร้านค้าเป็นผู้ทุจริตในโพรโตคอลการขึ้นเงินสดดิจิตอลที่ใช้แล้ว โดยอาศัยโพรโตคอล Secret Splitting และโพรโตคอล Bit Commitment ในกรณีที่ผู้ใช้ทุจริต โดยคัดลอกเงินสดดิจิตอลไว้ก่อนทำการจ่ายให้กับร้านค้า เซตของข้อมูลผู้ใช้ที่เข้ารหัสลับไว้จะถูกเปิดเผยครึ่งหนึ่งซึ่งร้านค้าจะเป็นผู้เลือกว่าให้ผู้ใช้เปิดเผยครึ่งใด เมื่อผู้ใช้นำไปใช้ซ้ำ เซตของข้อมูลผู้ใช้ที่เข้ารหัสลับไว้ก็จะถูกเปิดเผยครึ่งหนึ่งที่แตกต่างกัน เมื่อธนาคารตรวจสอบและพบข้อมูลผู้ใช้ที่เข้ารหัสลับไว้เพียงคู่เดียวที่ไม่ตรงกันก็จะสามารถระบุได้ว่าเป็นเงินของผู้ใช้ในระบบคนใด ในทางตรงกันข้ามกรณีที่ร้านค้าทุจริต เซตของข้อมูลผู้ใช้ที่เข้ารหัสลับไว้ก็จะถูกเปิดเผยครึ่งหนึ่งที่เหมือนกันทุกคู่ ธนาคารก็จะสามารถระบุได้ว่าร้านค้าหรือผู้ต้องการขึ้นเงินเป็นผู้ทุจริต

ดังรูปที่ 3.6 ข้อความระบุตัวที่เข้ารหัสลับไว้แบ่งเป็น 2 ส่วน คือเซตของส่วนซ้าย $\{I_{1L},I_{2L},\ldots,I_{nL}\}$ และเซตของส่วนขวา $\{I_{1R},I_{2R},\ldots,I_{nR}\}$ ถ้าผู้ใช้คัดลอกเงินและนำไปใช้ซ้ำ คู่ของข้อความระบุตัวจะถูกเปิดเผยครึ่งซ้ายและครึ่งขวาไม่ตรงกับการใช้ครั้งแรก เงินสดดิจิตอลที่ถูกใช้ครั้งแรกถูกเปิดเผย $\{I_{1L}, I_{2L}, I_{3R}, I_{4L},\ldots, I_{nR}\}$ ส่วนเงินสดดิจิตอลที่ถูกใช้ซ้ำ ถูกเปิดเผย $\{I_{1L}, I_{2L}, I_{3R}, I_{4R},\ldots, I_{nL}\}$



รูปที่ 3.6 การตามรอยเมื่อเกิดการทุจริตใช้เงินซ้ำ

จะเห็นว่ามีครึ่งที่เปิดเผยไม่เหมือนกัน คือ I_{4L} กับ I_{4R} ซึ่งถ้านำทั้งสองครึ่งมา Exclusive OR กันก็จะได้ข้อความระบุตัว ทำให้ธนาคารสามารถตามรอยหรือระบุตัวผู้ใช้ได้ ในกรณีที่ร้านค้าคัดลอกเงินและนำไปใช้ซ้ำ คู่ของข้อความระบุตัวที่เปิดเผยจะตรงกันทุกคู่ ธนาคารก็จะทราบได้ว่าร้านค้าที่กำลังขึ้นนี้กับธนาคารเป็นผู้ทุจริตใช้เงินซ้ำ

บทที่ 4
ต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์

เนื้อหาในบทนี้จะกล่าวถึง ส่วนประกอบของระบบและโพรโตคอลที่เกิดขึ้นระหว่างผู้ใช้และธนาคารในเชิงการพัฒนาซอฟต์แวร์ สมมติฐานเบื้องต้นของระบบ ต้นแบบซอฟต์แวร์ของธนาคารและผู้ใช้ รวมถึงประสิทธิภาพในการใช้งานจริง รายละเอียดต่างๆอธิบายได้ดังต่อไปนี้

4.1 ส่วนประกอบของระบบ

4.1.1 กุญแจ

กุญแจที่ใช้ในการเข้าและถอดรหัสลับในระบบนี้ จะประกอบด้วยกัน 2 ประเภทคือกุญแจสาธารณะและกุญแจส่วนตัว โดยคู่กุญแจสาธารณะและกุญแจส่วนตัวคู่หนึ่งจะถูกสร้างขึ้นและจัดเก็บต่างกัน คือสำหรับผู้ใช้จะเก็บแฟ้มข้อมูลกุญแจแฟ้มข้อมูลหนึ่งซึ่งเข้ารหัสลับด้วยรหัสผ่านของผู้ใช้เอง โดยแฟ้มข้อมูลกุญแจแฟ้มนี้จะประกอบไปด้วยกุญแจสาธารณะและกุญแจส่วนตัวของผู้ใช้ ส่วนกุญแจสาธารณะของธนาคารจะถูกจัดเก็บเป็นแฟ้มข้อมูลซึ่งมีกุญแจสาธารณะของธนาคารอย่างเดียวให้กับผู้ใช้ ในทำนองเดียวกันสำหรับธนาคาร ธนาคารก็จะมีแฟ้มข้อมูลกุญแจแฟ้มหนึ่งซึ่งเก็บทั้งกุญแจสาธารณะและกุญแจส่วนตัวของธนาคาร และเข้ารหัสลับด้วยรหัสผ่านของธนาคารเอง ฐานข้อมูลของธนาคารก็จัดเก็บเฉพาะแฟ้มข้อมูลกุญแจสาธารณะของผู้ใช้เท่านั้น

ในต้นแบบของระบบนี้คู่กุญแจสาธารณะและกุญแจส่วนตัวคู่หนึ่งจะจัดเก็บเป็นแฟ้มข้อมูล (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ หมายเลขบัญชี.acc) ส่วนกุญแจสาธารณะจะจัดเก็บแฟ้มข้อมูล (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ หมายเลขบัญชี.pub) ในส่วนของชื่อแฟ้มข้อมูลนี้ถ้าเป็นของผู้ใช้จะใช้หมายเลขบัญชีของผู้ใช้เป็นชื่อแฟ้มข้อมูล ส่วนของธนาคารเองใช้ (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ 0000000000.acc และ 0000000000.pub) คู่กุญแจนี้จะถูกสร้างโดยวิธี RSA และกุญแจมีความยาว 1024 บิต กุญแจที่สร้างขึ้นจะประกอบไปด้วยส่วนต่าง ๆ ดังตาราง

ตารางที่ 4.1 ส่วนประกอบสาธารณะ

ค่า n (Modulus)	ความยาวของ n (หน่วยเป็นบิต)
ค่า e	ความยาวของ e (หน่วยเป็นบิต)

ตารางที่ 4.2 ส่วนประกอบส่วนตัว

ค่า d	ความยาวของ d (หน่วยเป็นบิต)
ค่า p	ความยาวของ p (หน่วยเป็นบิต)
ค่า q	ความยาวของ q (หน่วยเป็นบิต)
ค่า u	ความยาวของ u (หน่วยเป็นบิต)
ค่า e1	ความยาวของ e1 (หน่วยเป็นบิต)
ค่า e2	ความยาวของ e2 (หน่วยเป็นบิต)

4.1.2 **แฟ้มข้อความระบุตัว**

แฟ้มข้อมูลนี้จะเก็บข้อความที่ระบุตัวผู้ใช้ได้ เช่น ชื่อและนามสกุลของผู้ใช้ ที่อยู่ หรือหมายเลขบัตรประชาชน เป็นต้น แฟ้มข้อความระบุตัวจะใช้ในโพรโตคอลการไหลของเงิน เพื่อสามารถทำให้ระบบสามารถทำการพิสูจน์ตัวจริง รวมถึงการตามรอยได้ถ้าเกิดการทุจริตใช้เงินซ้ำในระบบ

ในต้นแบบของระบบนี้แฟ้มข้อความระบุตัว จะเก็บหมายเลขบัญชี 10 หลัก และหมายเลขบัตรประจำตัวประชาชน 13 หลัก โดยจัดเก็บเป็นแฟ้มข้อมูลที่มีขนาด 23 ไบต์ (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ หมายเลขบัญชี.ids)

4.1.3 **เงินสดดิจิตอล**

เงินสดดิจิตอลเกิดจากใบสั่งเงินที่ผ่านโพรโตคอลการซื้อเงินแล้ว นั่นคือได้รับการเซ็นรับรองโดยธนาคาร หรือได้รับการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคารแล้ว เป็นเงินที่ถูกต้อง

ใบสั่งเงินจะถูกทำการสร้างทางฝั่งผู้ใช้ที่ต้องการซื้อเงินสดดิจิตอลจากธนาคาร ขั้นตอนการสร้างใบสั่งเงินแสดงได้ดังรูปที่ 4.1 มีลำดับขั้นตอนดังนี้

1. เริ่มต้นจากนำแฟ้มข้อความระบุตัวซึ่งประกอบด้วยหมายเลขบัญชี และหมายเลขบัตรประจำตัวประชาชน ขนาด 23 ไบต์ มาผ่านโพรโตคอล Secret Splitting

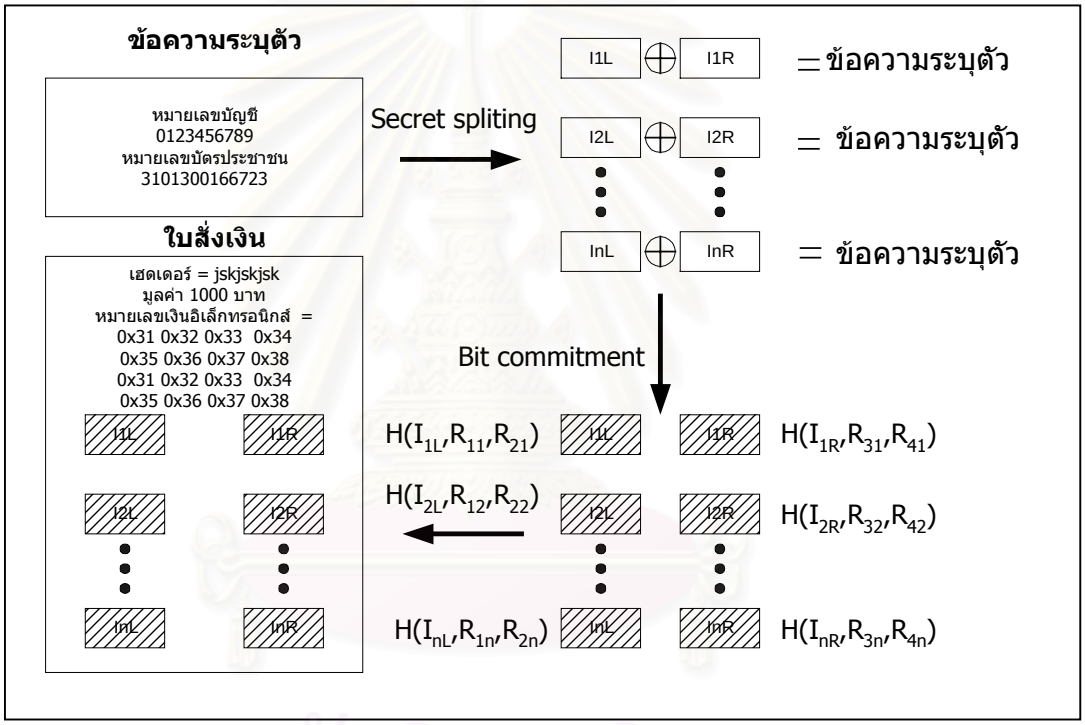
- ทำการสุ่มข้อความสุ่มที่มีขนาด 23 ไบต์เท่ากับขนาดของแฟ้มข้อความระบุตัว ได้เป็นเซตของข้อความสุ่ม คือ $\{I_{1R},I_{2R},\ldots,I_{nR}\}$ จากนั้นนำมา Exclusive OR กับ ข้อความระบุตัว จะได้ เซตของผลลัพธ์ คือ $\{I_{1L},I_{2L},\ldots,I_{nL}\}$

- ถ้านำสมาชิกของเซตข้อความสุ่ม และสมาชิกของเซตของผลลัพธ์ที่มีลำดับที่ตรงกันมาทำการ Exclusive OR กัน เช่น I_{1R} XOR I_{1L} ก็จะได้ข้อความระบุตัวคืนมา

2. จากนั้นสมาชิกของทั้ง 2 เซต จะถูกนำมาผ่านโพรโตคอล Bit Commitment โดยอาศัยฟังก์ชันแฮช

- เซตข้อความสุ่มจำนวน 4 เซต ดังนี้ คือ $\{R_{11},R_{12},\ldots,R_{1n}\}$, $\{R_{21},R_{22},\ldots,R_{2n}\}$, $\{R_{31},R_{32},\ldots,R_{3n}\}$ และ $\{R_{41},R_{42},\ldots,R_{4n}\}$ สมาชิกแต่ละตัวจะถูกนำมาผ่านฟังก์ชันแฮชกับสมาชิกของเซตข้อความสุ่มนี้ จะได้เซตของข้อมูลที่ทำการปกปิดแล้วดังนี้ เซตของส่วนซ้าย คือ $\{ (H(I_{1L},R_{11},R_{21}),R_{11}), (H(I_{2L},R_{12},R_{22}),R_{12}),\ldots, (H(I_{nL},R_{1n},R_{2n}),R_{1n})\}$ และเซตของส่วนขวา คือ $\{ (H(I_{1R},R_{31},R_{41}),R_{31}), (H(I_{2R},R_{32},R_{42}),R_{32}),\ldots, (H(I_{nR},R_{3n},R_{4n}),R_{3n})\}$ สมาชิกในแต่ละเซตจะประกอบด้วยค่าแฮช และข้อความสุ่มที่เปิดเผย 1 ค่า

3. เมื่อนำชุดของข้อมูลที่เป็นผลลัพธ์นี้จะนำไปรวมกับ เฮดเดอร์ที่มีขนาด 10 ไบต์ มูลค่าของเงินที่มีขนาด 4 ไบต์ และหมายเลขของเงินสดดิจิตอลขนาด 16 ไบต์ ก็จะได้ใบสั่งเงินมา



รูปที่ 4.1 ขั้นตอนการสร้างใบสั่งเงิน

เมื่อใบสั่งเงินผ่านโพรโตคอลการซื้อเงินเสร็จแล้ว ผู้ใช้จะได้เงินสดดิจิตอลที่ยังไม่ได้ใช้มา ซึ่งสามารถตรวจสอบว่าเป็นเงินที่ถูกต้องได้โดยใช้กุญแจสาธารณะของธนาคาร ถ้าได้ผลลัพธ์เป็นข้อมูลที่มีเฮดเดอร์ที่ถูกต้องก็แสดงว่าเป็นเงินจริง เมื่อผู้ใช้นำเงินสดดิจิตอลที่ยังไม่ได้ใช้ไปใช้ ข้อมูลที่ระบุตัวผู้ใช้ที่เป็นความลับจะถูกเปิดเผยครึ่งหนึ่ง และก็จะเรียกว่าเป็น เงินสดดิจิตอลที่ใช้แล้ว

เงินสดดิจิตอลที่ยังไม่ได้ใช้จะถูกเข้ารหัสด้วย DES โหมด CBC และใช้รหัสผ่านในการสร้าง Initialize Vector ขนาด 8 บิตด้วย ทำให้เฉพาะเจ้าของเงินสดดิจิตอลนี้ หรือผู้ที่ทราบรหัสผ่านที่ถูกต้องจึงจะสามารถนำเงินสดดิจิตอลนี้ไปใช้ได้

ในต้นแบบของระบบนี้ ใบสั่งเงินจะใช้เฮดเดอร์ที่มีขนาด 10 ไบต์ มูลค่าของเงินมีขนาด 4 ไบต์ หมายเลขของเงินสดดิจิตอลมีขนาด 16 ไบต์ และใช้ข้อความระบุตัว 10 คู่ด้วยกัน ข้อความระบุตัวขนาด 23 ไบต์ประกอบด้วยหมายเลขบัญชี 10 ไบต์ และหมายเลขบัตรประจำตัวประชาชน 13 ไบต์ สำหรับแฟ้มข้อมูลของเงินสดดิจิตอลที่ยังไม่ได้ใช้จะจัดเก็บเป็นแฟ้มข้อมูล (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ ตัวเลข.csh) และเงินสดดิจิตอลที่ใช้แล้วจะจัดเก็บเป็นแฟ้มข้อมูล (รูปแบบของแฟ้มข้อมูลที่จัดเก็บ คือ ตัวเลข.ush) โดยตัวเลขที่ใช้เป็นชื่อของแฟ้มข้อมูลจะเริ่มจากค่า 0 แต่ถ้าผู้ใช้มีแฟ้มข้อมูลชื่อนี้ ตัวเลขจะเพิ่มไปทีละหนึ่งจนกว่าจะไม่พบแฟ้มข้อมูลชื่อนี้ จากนั้นจึงใช้ตัวเลขค่าที่ได้เป็นชื่อของแฟ้มข้อมูลที่จะทำการบันทึก

4.1.4 ฐานข้อมูล

ฐานข้อมูลจะอยู่ที่ธนาคาร โดยจะเก็บข้อมูลของผู้ใช้, ข้อมูลของเงินสดดิจิตอลที่ใช้แล้ว และวันหมดอายุของเงินสดดิจิตอล

ในต้นแบบของระบบนี้ธนาคารจะมีฐานข้อมูล Microsoft Access 97 (*.mdb) คือ แฟ้มข้อมูล Bank.mdb ที่มีตารางการจัดเก็บข้อมูลดังนี้

ตารางที่ 4.3 ตาราง User สำหรับเก็บข้อมูลต่างๆเกี่ยวกับผู้ใช้งานระบบ

ชื่อสดมภ์	ชนิดข้อมูล	หมายเหตุ
Account_id (Primary key)	AutoNumber	เก็บดัชนีของระเบียนข้อมูล
Account_identity_string	Text	เก็บหมายเลขบัญชี และหมายเลขบัตรประจำตัวประชาชน
Account_balance	Number	เก็บยอดเงินในบัญชี
Public_key_path	Text	เก็บแฟ้มข้อมูลกุญแจสาธารณะ
Identity_string_path	Text	เก็บแฟ้มข้อมูลระบุตัวผู้ใช้

ตารางที่ 4.4 ตาราง Cash สำหรับเก็บข้อมูลต่างๆของเงินสดดิจิตอลที่ใช้แล้ว

ชื่อสดมภ์	ชนิดข้อมูล	หมายเหตุ
Cash_id (Primary key)	AutoNumber	เก็บดัชนีของระเบียนข้อมูล
Amount	Number	มูลค่าของเงิน
Unique_string	Text	หมายเลขของเงินสดดิจิตอล
Expire_date	Date/Time	วันหมดอายุ

ตารางที่ 4.5 ตาราง Cash_secret สำหรับเก็บข้อมูลส่วนที่เป็นความลับของเงินสดดิจิตอลที่ใช้แล้ว

ชื่อสดมภ์	ชนิดข้อมูล	หมายเหตุ
Cash_id	AutoNumber	เก็บดัชนีของระเบียนข้อมูล
L	Text	ข้อมูลความลับครึ่งซ้าย
R	Text	ข้อมูลความลับครึ่งขวา

ตารางที่ 4.6 ตาราง Expire สำหรับเก็บข้อมูลอายุของเงินสดดิจิตอล

ชื่อสดมภ์	ชนิดข้อมูล	หมายเหตุ
Expire_date	Date/Time	วันหมดอายุ
Before_date	Date/Time	วันเริ่มต้นที่จะกำหนดอายุเงินสดดิจิตอลให้เป็น Next_expire_date
After_date	Date/Time	วันสิ้นสุดที่จะคืนเงินสดดิจิตอลที่ยังไม่ได้ใช้
Next_expire_date	Date/Time	วันหมดอายุของเงินชุดถัดไป

4.1.5 ซอฟต์แวร์

ซอฟต์แวร์ที่ใช้ในระบบถูกพัฒนาโดยใช้โปรแกรมที่เขียนขึ้นบนเครื่องไมโครคอมพิวเตอร์ด้วยภาษา C++ และขั้นตอนวิธีการเข้ารหัสลับที่ใช้ในระบบ จะเรียกใช้จาก library “cl32.dll” [20]

4.2 สมมติฐานเบื้องต้นของระบบ

ในเบื้องต้นผู้ใช้ระบบจะต้องทำการเปิดบัญชีกับธนาคารเสียก่อน โดยธนาคารจะทำการเก็บข้อมูลของผู้ใช้ และทำการสร้างคู่กุญแจสาธารณะและกุญแจส่วนตัวให้กับผู้ใช้ ธนาคารจะมีฐานข้อมูลเพื่อเก็บข้อมูลของผู้ใช้ระบบ คือ หมายเลขบัญชี ยอดเงินในบัญชี กุญแจสาธารณะของผู้ใช้ และแฟ้มข้อมูลระบุตัวผู้ใช้ ซึ่งแฟ้มข้อมูลระบุตัวผู้ใช้นี้อาจจะประกอบไปด้วย ชื่อและนามสกุลของผู้ใช้ ที่อยู่ หรือหมายเลขบัตรประชาชน เป็นต้น นอกจากนี้ธนาคารก็จะมีฐานข้อมูลสำหรับเก็บข้อมูลของเงินสดดิจิตอลที่ใช้แล้ว และเก็บอายุเงินสดดิจิตอล

ในส่วนของธนาคารจะมีซอฟต์แวร์สำหรับธนาคารเพื่อใช้ในการติดต่อกับผู้ใช้และฐานข้อมูล แฟ้มข้อมูลกุญแจสาธารณะและกุญแจส่วนตัวของธนาคาร แฟ้มข้อมูลกุญแจสาธารณะและแฟ้มข้อมูลระบุตัวผู้ใช้ของผู้ใช้

ในส่วนของผู้ใช้ระบบก็จะมีซอฟต์แวร์สำหรับผู้ใช้ซึ่งเป็นโปรแกรมที่ใช้ในการติดต่อกับธนาคารหรือผู้ใช้อื่น แฟ้มข้อมูลกุญแจซึ่งเก็บกุญแจสาธารณะและกุญแจส่วนตัวของผู้ใช้ซึ่งจะถูก

เข้ารหัสลับด้วยรหัสผ่านของผู้ใช้ แฟ้มข้อมูลระบุตัวผู้ใช้ และแฟ้มข้อมูลกุญแจสาธารณะของธนาคาร

4.3 โพรโตคอลของการไหลของเงิน

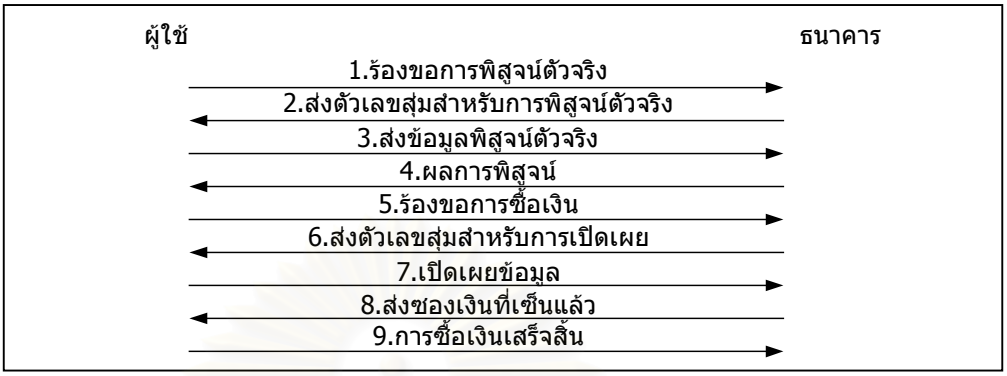
โพรโตคอลของการไหลของเงินจะเกิดขึ้นได้ ก็ต่อเมื่อผู้ใช้ทำการเชื่อมต่อไปยังผู้ใช้อื่น หรือธนาคารเสร็จสิ้นแล้ว รายละเอียดของแต่ละโพรโตคอลอธิบายได้ดังนี้

4.3.1 การซื้อเงิน

โพรโตคอลการซื้อเงินสดดิจิตอลเกิดจากการเชื่อมต่อระหว่างผู้ใช้ไปยังธนาคาร แสดงดังรูปที่ 4.2 และมีลำดับขั้นตอนดังนี้

1. ผู้ใช้ร้องขอการพิสูจน์ตัวจริงไปยังธนาคาร
2. ธนาคารจะสุ่มตัวเลขขึ้นมาขนาด VERIFY_RANDOM_BYTE ไบต์ (กำหนดไว้ 16 ไบต์) และส่งกลับไปให้ผู้ใช้
3. ผู้ใช้ทำการเข้ารหัสลับตัวเลขสุ่มนั้นด้วยกุญแจส่วนตัวของผู้ใช้เอง และส่งผลลัพธ์พร้อมกับข้อมูลระบุตัวผู้ใช้ไปให้ธนาคาร
4. ธนาคารทำการพิสูจน์ผู้ใช้กับฐานข้อมูลว่าถูกต้องหรือไม่จากนั้นจะส่งผลไปยังผู้ใช้
5. ถ้าผลการพิสูจน์ถูกต้อง ผู้ใช้จะทำการร้องขอการซื้อเงินสดดิจิตอลไปยังธนาคารตามที่ผู้ใช้ต้องการจะซื้อ โดยส่งใบสั่งเงินที่ผ่านโพรโตคอล Blind Signature แล้วจำนวน MONEY_ORDER ซอง (กำหนดไว้ 100 ซอง) ตัวอย่างเช่น ผู้ใช้ต้องการซื้อเงินสดดิจิตอลมูลค่า 100 บาท 1 ใบ และ 50 บาท 1 ใบ ระบบจะสร้างใบสั่งเงินที่ถูกใส่ซองไว้อย่างละ 100 ใบ รวมเป็น 200 ใบ เป็นต้น แล้วจึงส่งไปให้ธนาคาร
6. ธนาคารจะทำการส่งตัวเลขสุ่มระหว่าง 0 ถึง MONEY_ORDER-1 และส่งกลับไปให้ผู้ใช้ เพื่อให้ผู้ใช้ไม่ต้องเปิดเผยข้อมูลของซองที่มีค่าเท่ากับตัวเลขสุ่มนั้น แต่ซองที่เหลือจะต้องเปิดเผยมาทั้งหมด
7. ผู้ใช้ส่งข้อมูลเปิดเผยทุกซองไปยังธนาคาร ยกเว้นซองที่มีค่าเท่ากับตัวเลขสุ่ม
8. ธนาคารทำการตรวจสอบความถูกต้องของข้อมูลทุกซองว่าถูกต้อง ตัวอย่างเช่น ผู้ใช้ต้องการซื้อเงินสดดิจิตอลมูลค่า 100 บาท 1 ใบ ธนาคารจะได้รับข้อมูล 100 ซอง แต่จะได้รับการเปิดเผยจากผู้ใช้ 99 ซอง ธนาคารจะทำการตรวจสอบ 99 ซองที่เปิดเผยนั้นว่ามีมูลค่าเท่ากันจริงทุกซอง มีข้อมูลระบุตัวผู้ใช้จริง และยอดเงินของผู้ใช้มีมูลค่ามากพอ เป็นต้น จากนั้นธนาคารจะทำการเข้ารหัสลับด้วยกุญแจส่วนตัวของธนาคารกับซองที่เหลือ 1 ซอง และส่งกลับไปให้ผู้ใช้ พร้อมทั้งหักยอดเงินในบัญชีของผู้ใช้ตามมูลค่านั้น

9. ผู้ใช้ทำการ Unblind ข้อมูล 1 ซองที่เหลือ ก็จะได้เงินสดดิจิตอลที่ยังไม่ได้ใช้ พร้อมทั้งตรวจสอบความถูกต้องของเงินสดดิจิตอลนี้และส่งผลเสร็จสิ้นไปยังธนาคาร

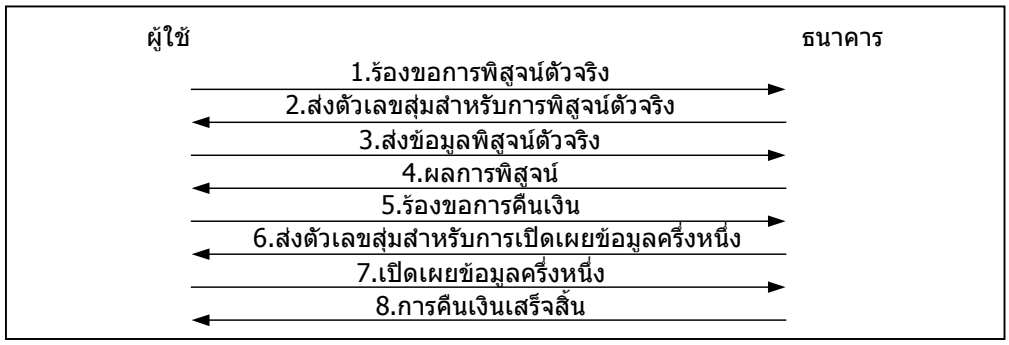


รูปที่ 4.2 โพรโตคอลการซื้อเงิน

4.3.2 การคืนเงิน

โพรโตคอลการคืนเงินสดดิจิตอลเกิดจากการเชื่อมต่อระหว่างผู้ใช้ไปยังธนาคาร แสดงดังรูปที่ 4.3 และมีลำดับขั้นตอนดังนี้

1. ผู้ใช้ร้องขอการพิสูจน์ตัวจริงไปยังธนาคาร
2. ธนาคารจะสุ่มตัวเลขขึ้นมาขนาด VERIFY_RANDOM_BYTE ไบต์ (กำหนดไว้ 16 ไบต์) และส่งกลับไปให้ผู้ใช้
3. ผู้ใช้ทำการเข้ารหัสลับตัวเลขสุ่มนั้นด้วยกุญแจส่วนตัวของผู้ใช้เอง และส่งผลลัพธ์พร้อมกับข้อมูลระบุตัวผู้ใช้ไปให้ธนาคาร
4. ธนาคารทำการพิสูจน์ผู้ใช้กับฐานข้อมูลว่าถูกต้องหรือไม่จากนั้นจะส่งผลไปยังผู้ใช้
5. ถ้าผลการพิสูจน์ถูกต้อง ผู้ใช้ทำการร้องขอการคืนเงินสดดิจิตอลไปยังธนาคาร
6. ธนาคารจะทำการสุ่มตัวเลขระหว่าง 0 หรือ 1 จำนวน IDENTITY_STRING ค่า (กำหนดไว้ 10 คู่) เพื่อสำหรับเปิดเผยข้อมูลระบุตัวผู้ใช้ครึ่งหนึ่ง คือ 1 จะเปิดเผยข้อมูลระบุตัวผู้ใช้ในส่วนซ้าย และ 0 จะเปิดเผยข้อมูลระบุตัวผู้ใช้ในส่วนขวา จากนั้นจะส่งตัวเลขชุดนี้ไปยังผู้ใช้
7. ผู้ใช้จะทำการเปิดเผยข้อมูลครึ่งหนึ่งตามค่าตัวเลขนั้น
8. เมื่อธนาคารตรวจสอบแล้วว่า เงินสดดิจิตอลเป็นเงินที่ถูกต้อง ยังไม่หมดอายุการใช้งาน และไม่มีการทุจริตใช้ซ้ำก็จะเพิ่มยอดเงินในบัญชีให้กับผู้ใช้ และส่งผลเสร็จสิ้นไปให้ผู้ใช้

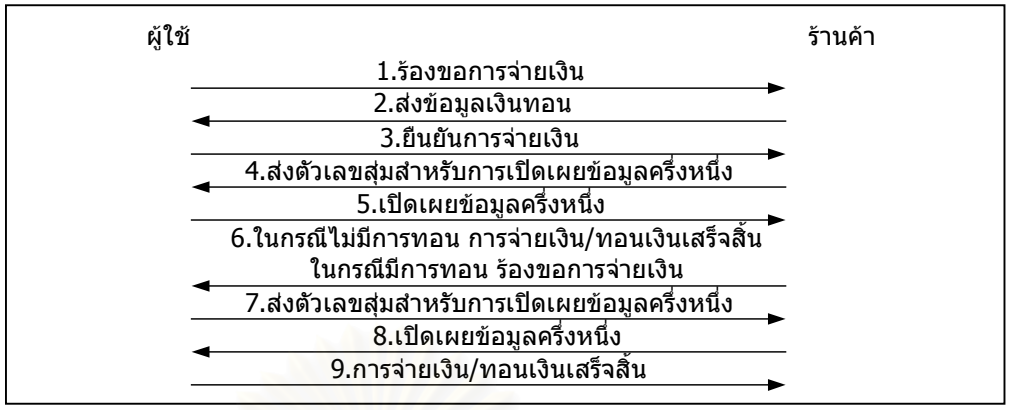


รูปที่ 4.3 โพรโตคอลการคืนเงิน

4.3.3 การจ่ายเงิน/ทอนเงิน

โพรโตคอลการคืนเงินสดดิจิตอลเกิดจากการเชื่อมต่อระหว่างผู้ใช้ไปยังผู้ใช้อื่นหรือร้านค้า แสดงดังรูปที่ 4.4 และมีลำดับขั้นตอนดังนี้

1. ผู้ใช้ร้องขอการจ่ายเงิน/ทอนเงินไปยังร้านค้า โดยผู้ใช้สามารถจ่ายเงินเกินมูลค่าจริงได้ ยกตัวอย่างเช่น ผู้ใช้ต้องการจ่ายเงิน 80 บาท แต่มีเงินสดดิจิตอลมูลค่า 100 บาท ก็สามารถจ่ายเงินสดดิจิตอลนี้ไป โดยระบุจำนวนเงินที่ต้องการจ่ายจริง 80 บาทไปด้วย เป็นต้น
2. ในกรณีผู้ใช้ต้องการเงินทอน ร้านค้าจะคำนวณเงินทอนจากเงินสดดิจิตอลที่ยังไม่ได้ใช้ของตนเอง เพื่อเป็นเงินทอนให้กับผู้ใช้ จากนั้นจะส่งข้อมูลเงินทอนที่คำนวณได้ไปยังผู้ใช้ ข้อมูลเงินทอนที่ร้านค้าส่งไปผู้ใช้อาจจะไม่มีหรือมีไม่ถึงมูลค่าที่ทอนจริงก็ได้
3. ผู้ใช้รับข้อมูลเงินทอนแล้วพิจารณาว่า จะยืนยันการจ่ายเงินที่ได้รับเงินทอนจากร้านค้ามูลค่าตามข้อมูลนี้หรือไม่ ถ้ายืนยันผู้ใช้จึงส่งการยืนยันการจ่ายเงินไปยังร้านค้า
4. ร้านค้ารับการยืนยันจากผู้ใช้แล้ว ก็จะทำการสุ่มตัวเลขระหว่าง 0 หรือ 1 จำนวน IDENTITY_STRING ค่า (กำหนดไว้ 10 คู่) เพื่อสำหรับเปิดเผยข้อมูลระบุตัวผู้ใช้ครึ่งหนึ่ง คือ 1 จะเปิดเผยข้อมูลระบุตัวผู้ใช้ในส่วนซ้าย และ 0 จะเปิดเผยข้อมูลระบุตัวผู้ใช้ในส่วนขวา จากนั้นจะส่งตัวเลขชุดนี้ไปยังผู้ใช้
5. ผู้ใช้จะทำการเปิดเผยข้อมูลครึ่งหนึ่งตามค่าตัวเลขนั้น
6. ในกรณีไม่มีการทอนเงิน ร้านค้าจะส่งผลการจ่ายเงิน/ทอนเงินเสร็จสิ้นมายังผู้ใช้ ส่วนในกรณีที่มีการทอนเงิน ร้านค้าจะทำการร้องขอการจ่ายเงินไปยังผู้ใช้
7. ผู้ใช้ทำการสุ่มตัวเลขเหมือนข้อที่ 4 และส่งไปยังร้านค้า
8. ร้านค้าทำการเปิดเผยข้อมูลครึ่งหนึ่งตามค่าตัวเลขนั้น
9. ผู้ใช้ตรวจสอบและส่งผลการจ่ายเงิน/ทอนเงินเสร็จสิ้นไปยังร้านค้า



รูปที่ 4.4 โพรโตคอลการจ่ายเงิน/ทอนเงิน

4.3.4 การขึ้นเงิน

โพรโตคอลการขึ้นนเงินสดดิจิตอลเกิดจากการเชื่อมต่อระหว่างผู้ใช้ไปยังธนาคาร แสดงดังรูปที่ 4.5 และมีลำดับขั้นตอนดังนี้

1. ผู้ใช้ร้องขอการพิสูจน์ตัวจริงไปยังธนาคาร
2. ธนาคารจะสุ่มตัวเลขขึ้นมาขนาด VERIFY_RANDOM_BYTE ไบต์ (กำหนดไว้ 16 ไบต์) และส่งกลับไปให้ผู้ใช้
3. ผู้ใช้ทำการเข้ารหัสลับตัวเลขสุ่มนั้นด้วยกุญแจส่วนตัวของผู้ใช้เอง และส่งผลลัพธ์พร้อมกับข้อมูลระบุตัวผู้ใช้ไปให้ธนาคาร
4. ธนาคารทำการพิสูจน์ผู้ใช้กับฐานข้อมูลว่าถูกต้องหรือไม่ จากนั้นจะส่งผลไปยังผู้ใช้
5. ผู้ใช้ร้องขอการขึ้นเงิน พร้อมทั้งส่งเงินสดดิจิตอลที่ใช้แล้ว ไปยังธนาคาร
6. ธนาคารทำการตรวจสอบเงินสดดิจิตอลว่าถูกต้องหรือไม่ ถ้าถูกต้องก็จะเพิ่มยอดเงินในบัญชีให้กับผู้ใช้ พร้อมทั้งส่งผลการขึ้นเงินเสร็จสิ้นไปยังผู้ใช้

รูปที่ 4.5 โพรโตคอลการขึ้นเงิน

4.4 ซอฟต์แวร์ของระบบ

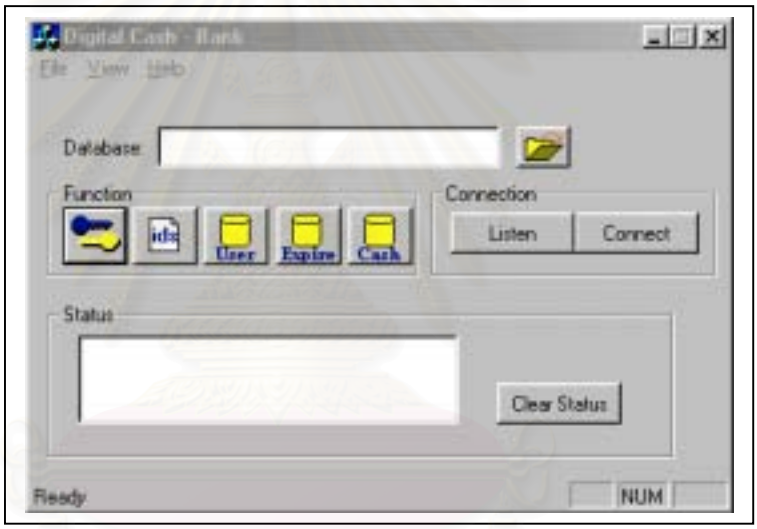
ซอฟต์แวร์เป็นส่วนติดต่อระหว่างผู้ใช้งานกับระบบ แบ่งเป็นส่วนของธนาคารและส่วนของผู้ใช้ โดยซอฟต์แวร์ทั้ง 2 ส่วนนี้ มีฟังก์ชันการติดต่อระหว่างกันเหมือนกัน หน้าต่างการติดต่อกับผู้ใช้งานและฟังก์ชันการทำงานของแต่ละส่วนเป็นดังนี้

4.4.1 ซอฟต์แวร์ส่วนของธนาคาร

ฟังก์ชันการทำงานของซอฟต์แวร์ส่วนนี้ ประกอบด้วย

1. การเลือกแฟ้มฐานข้อมูลที่จะติดต่อ

ธนาคารจะต้องเลือกแฟ้มฐานข้อมูลที่เป็น Microsoft Access 97 (*.mdb) ก่อนทำการเชื่อมต่อกับผู้ใช้เพื่อให้บริการซื้อ คืนและขึ้นเงินสดดิจิตอล ในต้นแบบของระบบนี้ใช้แฟ้มข้อมูล Bank.mdb ซึ่งมีโครงสร้างของฐานข้อมูลตามตารางที่ 4.3 –4.6



รูปที่ 4.6 หน้าจอหลักของซอฟต์แวร์สำหรับธนาคาร

2. การสร้างกุญแจสาธารณะ/กุญแจส่วนตัว และข้อความระบุตัว

ธนาคารจะมีฟังก์ชันที่ใช้ในการสร้างกุญแจสาธารณะ/กุญแจส่วนตัว โดยอาศัยการเข้ารหัสลับแบบ RSA ซึ่งกุญแจที่สร้างจะมีความยาว 1024 บิต ซึ่งจะสร้างเป็นแฟ้มข้อมูล (หมายเลขบัญชี.acc ซึ่งเก็บคู่กุญแจสาธารณะ/กุญแจส่วนตัว และ หมายเลขบัญชี.pub ซึ่งเก็บเฉพาะกุญแจสาธารณะ) และสร้างแฟ้มข้อมูลข้อความระบุตัว (หมายเลขบัญชี.ids)

3. การติดต่อกับซอฟต์แวร์ของผู้ใช้

ธนาคารจะมีฟังก์ชันในการรับฟัง ยกเลิกการรับฟัง ติดต่อและยกเลิกการติดต่อกับผู้ใช้ โปรแกรมจะสร้างซ็อกเก็ตเพื่อใช้ในการติดต่อกับผู้ใช้ ในต้นแบบของซอฟต์แวร์นี้จะมีปุ่มฟังก์ชันแสดงดังรูปที่ 4.6

4. การให้บริการซื้อ คืนและขึ้นเงินสดดิจิตอลแก่ผู้ใช้

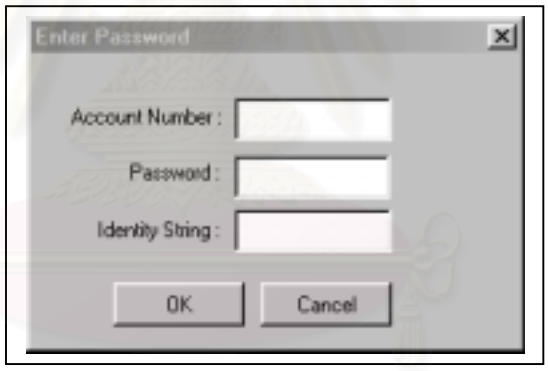
ในการใช้งานธนาคารจะสามารถให้บริการได้ก็ต่อเมื่อทำการติดต่อกับผู้ใช้แล้ว ในต้นแบบของซอฟต์แวร์นี้ฟังก์ชันในการให้บริการซื้อ คืนและขึ้นเงินสดดิจิตอล จะทำงานโดยอัตโนมัติ

5. การตรวจสอบและการตามรอย

ต้นแบบของซอฟต์แวร์นี้จะทำการตรวจสอบความถูกต้องของข้อมูล ในการให้บริการต่างๆ แก่ผู้ใช้ เช่น การตรวจสอบความถูกต้องของเฮดเดอร์ของข้อมูล เป็นต้น ตรวจสอบเงินสดดิจิตอลว่าเป็นเงินสดดิจิตอลจริง รวมทั้งการตามรอยเมื่อเกิดการทุจริตใช้เงินสดดิจิตอลซ้ำได้ว่าผู้ใช้ใดเป็นผู้กระทำ โปรแกรมจะทำการบันทึกข้อมูลผู้ทุจริตลงในแฟ้มข้อมูลข้อความ (ในระบบนี้ใช้ แฟ้มข้อมูล BlackList.txt ในการจัดเก็บ)

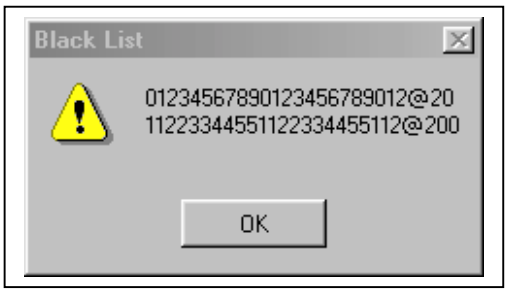
จากรูปที่ 4.6 ปุ่มฟังก์ชันจะประกอบไปด้วย

1. ปุ่มฟังก์ชันรูปคู่กุญแจ ซึ่งจะทำการสร้างแฟ้มข้อมูลคู่กุญแจสาธารณะและกุญแจส่วนตัว (หมายเลขบัญชี.acc), แฟ้มข้อมูลกุญแจสาธารณะ (หมายเลขบัญชี.pub) และแฟ้มข้อมูลระบุตัว (หมายเลขบัญชี.ids)

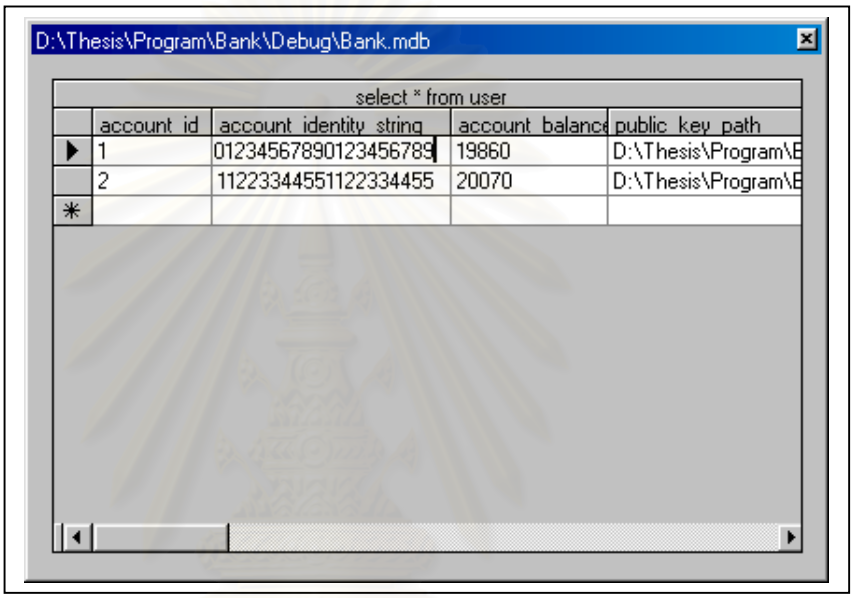


รูปที่ 4.7 หน้าต่างสำหรับกรอกข้อมูลที่ใช้ในการสร้างกุญแจและข้อความระบุตัว

2. ปุ่มฟังก์ชันรูปเอกสาร จะทำการแสดงข้อความระบุตัวของผู้ที่ทุจริตใช้เงินสดดิจิตอลซ้ำและมูลค่าของเงินสดดิจิตอลนั้น ดังรูปที่ 4.8 ข้อความระบุตัว คือ หมายเลขบัญชีและหมายเลขบัตรประจำตัวประชาชนรวม 23 หลัก และ “@20” แสดงมูลค่าของเงินสดดิจิตอลที่ถูกใช้ซ้ำ
3. ปุ่มฟังก์ชันรูปฐานข้อมูลทั้ง 3 ปุ่ม จะทำการแสดงข้อมูลของผู้ใช้ ,ข้อมูลการกำหนดอายุเงินสดดิจิตอล และข้อมูลเงินสดดิจิตอลที่ใช้แล้ว ตัวอย่างดังรูปที่ 4.7 แสดงตารางของข้อมูลผู้ใช้



รูปที่ 4.8 หน้าต่างแสดงข้อมูลของผู้ทุจริตใช้เงินซ้ำ



รูปที่ 4.9 หน้าต่างแสดงข้อมูลของผู้ใช้

4.4.2 ซอฟต์แวร์ส่วนของผู้ใช้

1. การติดต่อกับซอฟต์แวร์ของธนาคารหรือผู้ใช้อื่น

เช่นเดียวกันกับซอฟต์แวร์ของธนาคาร ต้นแบบซอฟต์แวร์ของผู้ใช้ก็จะมีฟังก์ชันในการสร้างซ็อกเก็ตเพื่อใช้ในรับฟัง ยกเลิกการรับฟัง ติดต่อและยกเลิกการติดต่อกับธนาคารหรือผู้ใช้อื่น

2. การซื้อ คืนและขึ้นเงินสดดิจิตอลกับธนาคาร

ผู้ใช้สามารถซื้อ คืนและขึ้นเงินสดดิจิตอลโดยการเชื่อมต่อกับธนาคารก่อน โดยแต่ละฟังก์ชันจะเรียกใช้โพรโตคอลต่างๆตามที่กล่าวมาแล้วข้างต้น

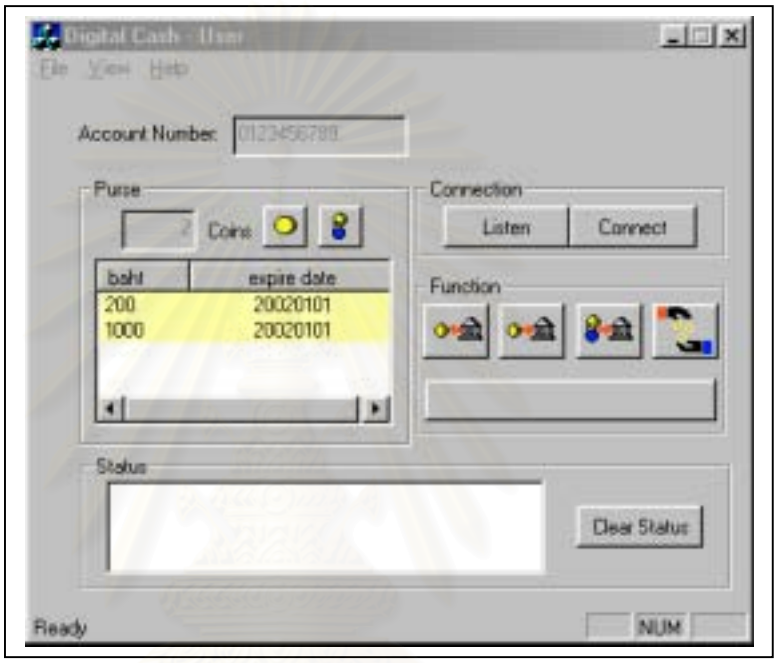
3. การจ่าย/ทอนเงินสดดิจิตอลกับผู้ใช้อื่นหรือร้านค้า

ผู้ใช้สามารถจ่ายเงินสดดิจิตอลให้กับผู้อื่นหรือร้านค้าได้ ต้นแบบของซอฟต์แวร์นี้จะสามารถทำการทอนเงินได้ ในกรณีที่ทางร้านค้ามีเงินสดดิจิตอลที่ยังไม่ได้ใช้สำหรับทอน

ไม่พอ โปรแกรมจะคำนวณเงินทอนให้ใกล้เคียงมูลค่าที่ต้องทอนจริงมากที่สุดจากเงินที่มีอยู่ ทั้งนี้ผู้ใช้จะได้ทราบมูลค่าของเงินทอนที่ร้านค้ามีให้ก่อนที่จะทำการยืนยันการจ่ายเงิน ในต้นแบบของซอฟต์แวร์นี้จะมีปุ่มฟังก์ชันต่างๆ แสดงดังรูปที่ 4.10

4. การตรวจสอบเงินสดดิจิตอล

ต้นแบบของซอฟต์แวร์นี้จะอาศัยกุญแจสาธารณะของธนาคารในการตรวจสอบเงินสดดิจิตอลได้ว่าเป็นเงินที่ธนาคารเซ็นจริงหรือไม่

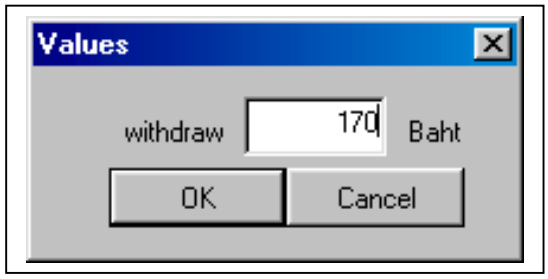


รูปที่ 4.10 หน้าจอหลักของซอฟต์แวร์สำหรับผู้ใช้

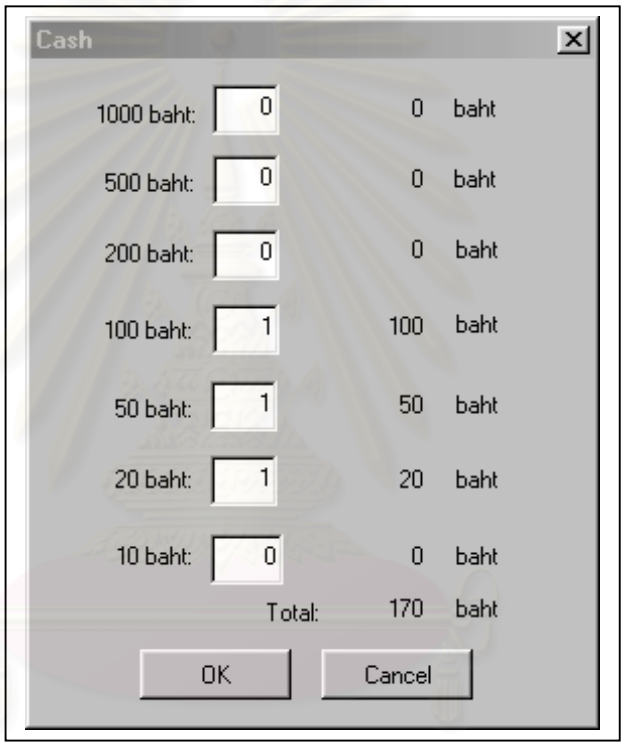
จากรูปที่ 4.10 ปุ่มฟังก์ชันจะประกอบไปด้วย

1. ปุ่มฟังก์ชันการซื้อเงิน

เมื่อผู้ใช้กดปุ่มฟังก์ชันนี้ จะมีหน้าต่างแสดงขึ้นมาเพื่อให้ผู้ใช้กรอกมูลค่าเงินที่ต้องการจะซื้อดังรูปที่ 4.11 หลังจากที่ผู้ใช้ทำการกรอกมูลค่าแล้ว หน้าต่างแสดงประเภทและจำนวนของเงินสดดิจิตอลแสดงดังรูปที่ 4.12 ตัวอย่างแสดงมูลค่า 170 บาท โปรแกรมจะเลือกเงินสดดิจิตอลมูลค่า 100 บาท 50 บาท และ 20 บาท ให้อย่างละใบ ผู้ใช้สามารถทำการแก้ไขได้เช่น ผู้ใช้อาจจะต้องการเงินสดดิจิตอลมูลค่า 10 บาท 2 ใบ แทนเงินสดดิจิตอลมูลค่า 20 บาท 1 ใบ เป็นต้น



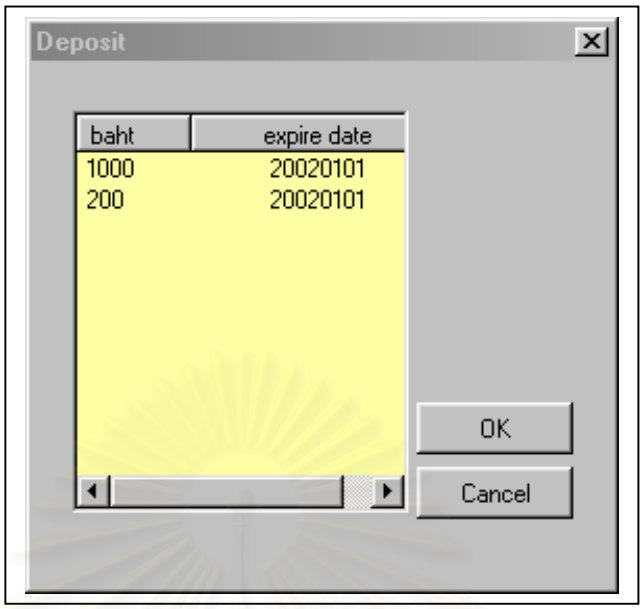
รูปที่ 4.11 หน้าต่างกรอกมูลค่าเงินที่ต้องการซื้อ



รูปที่ 4.12 หน้าต่างกรอกจำนวนเงินที่ต้องการซื้อ

2. ปุ่มฟังก์ชันการคืนเงิน

เมื่อผู้ใช้กดปุ่มฟังก์ชันนี้ จะมีหน้าต่างแสดงซึ่งแสดงเงินสดดิจิตอลที่ยังไม่ได้ใช้ทั้งหมดขึ้นมา ซึ่งจะมีข้อมูลของมูลค่าและวันหมดอายุแสดงดังรูปที่ 4.13 เงินสดดิจิตอลที่ยังไม่ได้ใช้จะมีรูปแบบแฟ้มข้อมูลเป็น “ตัวเลข.csh” ผู้ใช้สามารถเลือกเงินสดดิจิตอลที่ยังไม่ได้ใช้ที่แสดงอยู่ในรายการคืนธนาคารได้



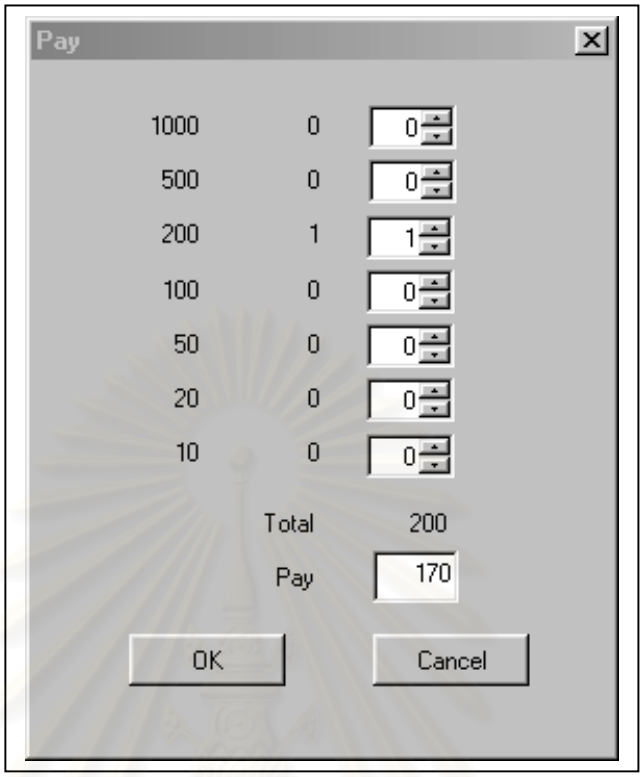
รูปที่ 4.13 หน้าต่างแสดงมูลค่าและวันหมดอายุของเงินที่ต้องการคืน

3. ปุ่มฟังก์ชันการขึ้นเงิน

เมื่อผู้ใช้กดปุ่มฟังก์ชันนี้ โปรแกรมจะทำการขึ้นเงินที่ใช้แล้วทั้งหมดไปให้กับธนาคาร เงินที่ใช้แล้วที่มีรูปแบบแฟ้มข้อมูลเป็น “ตัวเลข.ush” ผู้ใช้สามารถดูเงินสดดิจิตอลที่ใช้แล้ว และที่ยังไม่ได้ใช้ได้ จากรายการที่แสดงในส่วนของกระเป๋าเงิน (Purse) โดยการกดปุ่มฟังก์ชันรูปเหรียญจะแสดงเงินสดดิจิตอลที่ยังไม่ได้ใช้ และการกดปุ่มฟังก์ชันรูปเหรียญจะแสดงเงินสดดิจิตอลที่ใช้แล้ว แสดงดังรูปที่ 4.10

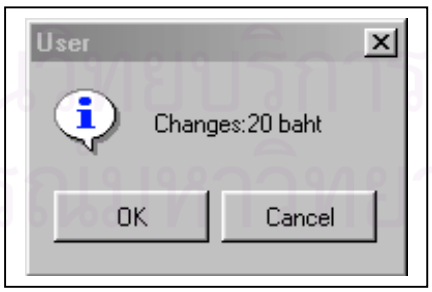
4. ปุ่มฟังก์ชันการจ่ายเงิน/ทอนเงิน

เมื่อผู้ใช้กดปุ่มฟังก์ชันนี้ หน้าต่างซึ่งแสดงประเภทของเงินสดดิจิตอลต่างๆขึ้นมาให้ผู้ใช้กำหนดว่าต้องการจ่ายเงินสดดิจิตอลแต่ละมูลค่าจำนวนเท่าใดให้กับร้านค้า รวมทั้งใส่มูลค่าที่ต้องการจ่ายจริงด้วย ในกรณีที่ค่าที่ต้องการจ่ายจริงน้อยกว่ามูลค่าที่จ่ายไปหรือผู้ใช้ต้องการเงินทอน ร้านค้าจะมีการส่งข้อมูลเงินทอนมาให้ผู้ใช้อีกทีว่ามีเงินทอนหรือไม่ และมูลค่าเท่าไรเพื่อให้ผู้ใช้ทำการยืนยันอีกครั้ง ถ้าผู้ใช้ตกลงเงินสดดิจิตอลก็จะถูกส่งไปให้กับร้านค้า รวมทั้งเงินสดดิจิตอลของร้านค้าที่ใช้เป็นเงินทอนก็จะถูกส่งมาให้กับผู้ใช้



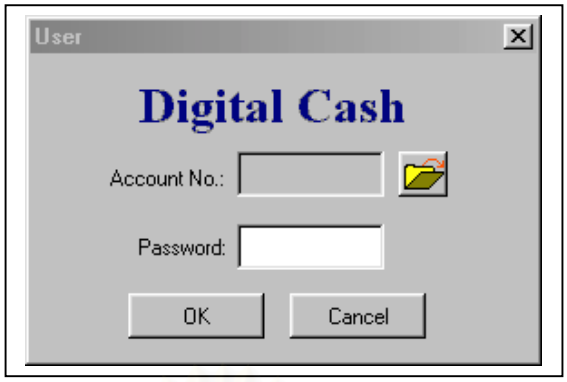
รูปที่ 4.14 หน้าต่างแสดงประเภทและจำนวนเงินสดดิจิตอลที่ต้องการจ่าย

จากรูปที่ 4.14 เมื่อผู้ใช้ต้องการจ่ายเงินสดดิจิตอลมูลค่า 170 บาท แต่ผู้ใช้มีเงินสดดิจิตอลมูลค่า 200 บาท 1 ใบ ร้านค้าส่งข้อความมาให้ผู้ทราบว่ามีเงินทอนหรือไม่ เพื่อให้ผู้ใช้ยืนยันการจ่ายเงินอีกที ดังรูปที่ 4.15 ข้อมูลเงินทอนที่ผู้ใช้ได้รับแสดงมูลค่า 20 บาท เมื่อผู้ใช้ตกลงยืนยันการจ่ายเงิน เงินสดดิจิตอลที่ใช้แล้วมูลค่า 200 บาทก็จะไปอยู่กับร้านค้า และเงินสดดิจิตอลที่ใช้แล้วมูลค่า 20 บาทก็จะไปอยู่กับผู้ใช้



รูปที่ 4.15 หน้าต่างแสดงข้อความเงินทอนที่ร้านค้ามีให้กับผู้ใช้

ในการใช้งานโปรแกรมส่วนของผู้ใช้ เริ่มแรกผู้ใช้จะต้องทำการเลือกแฟ้มข้อมูลคู่กุญแจสาธารณะและกุญแจส่วนตัว (หมายเลขบัญชี.acc) แล้วทำการใส่รหัสผ่านเพื่อเข้าสู่หน้าจอหลักของโปรแกรม หน้าต่างแรกของโปรแกรมแสดงดังรูปที่ 4.16



รูปที่ 4.16 หน้าต่างแรกของโปรแกรมส่วนของผู้ใช้

นอกจากปุ่มฟังก์ชันหลักทั้ง 4 ปุ่มที่ใช้ในการซื้อเงิน, คืนเงิน, ขึ้นเงิน และจ่าย/ทอนเงิน ที่หน้าจอหลักของโปรแกรมจะมีส่วนที่แสดงสถานะการทำงานของโปรแกรมขณะนั้นที่ส่วนของสถานะ (Status) ด้วย ทำให้ผู้ใช้สามารถทราบได้ว่าขณะนั้นโปรแกรมทำงานขั้นตอนใดอยู่

4.5 ความเร็วในการทำงานของซอฟต์แวร์

ในการใช้งานจริง ความเร็วของโพรโตคอลในการซื้อเงิน คืนเงิน จ่าย/ทอนเงิน และขึ้นเงินสดดิจิตอลของต้นแบบระบบนี้จะขึ้นอยู่กับพารามิเตอร์ต่างๆ ดังตารางที่ 4.7 ซึ่งถ้าเพิ่มค่าที่กำหนดในต้นแบบนี้ขึ้นจะทำให้ความปลอดภัยของระบบสูงขึ้น แต่ความเร็วของโพรโตคอลก็จะลดลงด้วยเช่นกัน

ตารางที่ 4.7 พารามิเตอร์ที่มีผลต่อความเร็วของโพรโตคอลต่างๆ

พารามิเตอร์	คำอธิบาย	ค่าที่กำหนดในต้นแบบ
MONEY_ORDER	จำนวนใบสั่งเงินที่ใช้ในโพรโตคอลการซื้อเงินสดดิจิตอลในส่วน Blind Signature	10 จำนวน
HEADER_BYTES	เฮดเดอร์สำหรับแฟ้มข้อมูล	10 ไบต์
IDENTITY_STRING	จำนวนคู่ของข้อความระบุตัวต่อใบสั่งเงิน	10 คู่
IDENTITY_STRING_BYTES	ขนาดของข้อความระบุตัว	23 ไบต์
BLIND_RANDOM_BYTES	ตัวเลขสุ่มสำหรับ Blind Signature	128 ไบต์
VERIFY_RANDOM_BYTES	ตัวเลขสุ่มสำหรับการพิสูจน์ตัวจริง	16 ไบต์
UNIQUE_STRING_BYTES	หมายเลขของเงินสดดิจิตอล	16 ไบต์

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้นำเสนอต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ เริ่มจากการศึกษาขั้นตอนวิธีและโพรโตคอลของวิทยาการเข้ารหัสลับ ทำการออกแบบต้นแบบของระบบ จากนั้นจึงพัฒนาซอฟต์แวร์ที่สอดคล้องกับที่ออกแบบไว้

ต้นแบบของระบบที่ออกแบบจะมีคุณสมบัติดังต่อไปนี้

1. ระบบทำงานด้วยซอฟต์แวร์ล้วนบนคอมพิวเตอร์ส่วนบุคคล และสามารถทำงานบนเครือข่ายคอมพิวเตอร์ทั่วไป
2. ความปลอดภัยของระบบขึ้นอยู่กับขั้นตอนวิธีของวิทยาการเข้ารหัสลับที่ใช้ ได้แก่ RSA, DES, และ MD5
3. เป็นระบบแบบเงินสดซึ่งจะให้ความเป็นส่วนตัวแก่ผู้ใช้
4. ระบบสามารถทำงานแบบออฟไลน์คือ ขั้นตอนการจ่าย/ทอนเงินระหว่างผู้ใช้ไม่จำเป็นต้องติดต่อไปยังธนาคาร
5. ระบบสามารถตามรอยผู้ทุจริตใช้เงินสดดิจิตอลซ้ำได้

โพรโตคอลการไหลของเงินสดดิจิตอล ประกอบด้วย 4 โพรโตคอลดังนี้

1. การซื้อเงิน –ผู้ใช้สามารถซื้อเงินสดดิจิตอลจากธนาคารได้ โดยอาศัยโพรโตคอล Blind Signature ในการปกปิดข้อความระบุ, โพรโตคอล Secret Splitting ในการแบ่งข้อความระบุตัวออกเป็น 2 ส่วน, และโพรโตคอล Bit Commitment ในการผูกมัดตัวเองกับเงินสดดิจิตอล
2. การคืนเงิน – ผู้ใช้สามารถคืนเงินสดดิจิตอลที่ยังไม่ได้ใช้กลับไปยังธนาคารได้ โดยต้องทำการเปิดเผยข้อความระบุตัวครึ่งหนึ่งให้กับธนาคารด้วย ผู้ใช้ต้องทำการเปิดเผยข้อมูลครึ่งหนึ่งที่ธนาคารต้องการเนื่องจากโพรโตคอล Bit Commitment ซึ่งการเปิดเผยข้อมูลครึ่งหนึ่งนี้จะเป็นประโยชน์ในการตามรอยผู้ทุจริตกรณีที่มีการใช้เงินซ้ำ
3. การจ่าย/ทอนเงิน – ผู้ใช้สามารถจ่ายเงินสดดิจิตอลไปยังร้านค้า โดยต้องทำการเปิดเผยข้อความระบุตัวครึ่งหนึ่งให้กับร้านค้า เช่นเดียวกับการทอนเงินของร้านค้า ก็ต้องทำการเปิดเผยข้อความระบุตัวครึ่งหนึ่งให้กับผู้ใช้ด้วย
4. การขึ้นเงิน – ผู้ใช้สามารถนำเงินสดดิจิตอลที่ใช้แล้วไปเปลี่ยนเป็นเงินสดกับธนาคารได้ ซึ่งธนาคารจะทำการตรวจสอบเงินสดดิจิตอลว่าเป็นเงินที่ถูกต้องและไม่การใช้ซ้ำ

ก็จะเพิ่มเงินในบัญชีผู้ใช้ แต่ถ้าตรวจพบการใช้ซ้ำ ธนาคารจะตามรอยผู้ทุจริตได้จากข้อความระบุตัวที่ถูกเปิดเผย

การซื้อเงิน คืนเงิน จ่าย/ทอนเงิน และขึ้นเงินสดดิจิตอล จะกระทำได้ก็ต่อเมื่อต้องทำการเชื่อมต่อระหว่างโปรแกรมก่อน และการซื้อเงิน คืนเงิน และขึ้นกับธนาคารนั้น จะมีโพรโตคอลการพิสูจน์ตัวจริงด้วย

ซอฟต์แวร์ของระบบถูกพัฒนาขึ้นด้วย Microsoft Visual C++ Version 6.0 ซึ่งประกอบด้วยกัน 2 โปรแกรม ดังนี้

1. โปรแกรมส่วนของธนาคาร ประกอบด้วยฟังก์ชันต่างๆ ดังนี้

– การสร้างคู่กุญแจสาธารณะและกุญแจส่วนตัว รวมทั้งข้อความระบุตัว
– การสร้างรายการแสดงผู้ทุจริตใช้เงินซ้ำ
– การดู, เพิ่ม, แก้ไข และลบข้อมูลในฐานข้อมูลที่เป็น Microsoft Access 97 ซึ่งใช้ในการเก็บข้อมูลของผู้ใช้ และเงินสดดิจิตอลที่ใช้แล้ว

2. โปรแกรมส่วนของผู้ใช้ ประกอบด้วยฟังก์ชันต่างๆ ดังนี้

- การซื้อเงิน คืนเงิน จ่าย/ทอนเงิน และขึ้นเงินสดดิจิตอล
- การดูเงินสดดิจิตอลที่ยังไม่ได้ใช้และที่ใช้แล้ว

ทั้งโปรแกรมในส่วนของธนาคารและส่วนของผู้ใช้จะมีปุ่มฟังก์ชันในการเชื่อมต่อระหว่างโปรแกรมเหมือนกัน และรายการสถานะที่แสดงข้อความระบุการทำงานในขั้นตอนต่างๆ ของโปรแกรมในขณะนั้น

สำหรับความเร็วในการใช้งานขึ้นอยู่กับการตั้งค่าพารามิเตอร์ต่างๆ ของระบบ ซึ่งพารามิเตอร์ที่สำคัญ ได้แก่ จำนวนใบสั่งเงินที่ปกปิด, จำนวนคู่ของข้อความระบุตัว, ขนาดของข้อความระบุตัว, และขนาดของกุญแจ เป็นต้น ถ้าปรับค่าพารามิเตอร์ขึ้นความปลอดภัยจะสูงขึ้น แต่ความเร็วในการใช้งานก็จะลดลงตาม

5.2 ข้อเสนอแนะสำหรับการวิจัยในอนาคต

1. ต้นแบบของระบบนี้สามารถให้ความเป็นส่วนตัว และการตามรอยได้เมื่อเกิดการทุจริตใช้เงินสดดิจิตอลซ้ำขึ้น แต่ระบบไม่มีคุณสมบัติการถ่ายโอน การประยุกต์ใช้เทคนิคที่นำเสนอกับเทคนิคอื่นๆของวิทยาการเข้ารหัสลับ จะเป็นแนวทางสามารถดำเนินการวิจัยต่อไปได้
2. มูลค่าที่ต่างกันของเงินสดดิจิตอลในระบบนี้ไม่มีผลต่อขนาดของเงินสดดิจิตอล เช่น เงินสดดิจิตอลมูลค่า 100 บาท มีขนาดแฟ้มข้อมูลเท่ากับ เงินสดดิจิตอลมูลค่า 10 บาท เป็นต้น การปรับเปลี่ยนให้ขนาดแฟ้มข้อมูลขึ้นอยู่กับมูลค่าเงินด้วยจะทำให้

ระบบสมบูรณ์มากขึ้น เช่น เงินสดดิจิตอลที่มีมูลค่าน้อย ก็อาจใช้จำนวนคู่ข้อความระบุตัวที่น้อยลง ถึงแม้ว่าความปลอดภัยจะลดลง แต่จะทำให้ขนาดของแฟ้มข้อมูลมีขนาดเล็กลง และความเร็วในการใช้งานสูงขึ้น เป็นต้น

3. ต้นแบบของระบบนี้ถูกพัฒนาขึ้นโดยใช้บนคอมพิวเตอร์ส่วนบุคคลเท่านั้น แต่เพื่อให้สะดวกกับผู้ใช้ในการใช้งานจริง โปรแกรมส่วนผู้ใช้อาจถูกนำไปพัฒนาบนพาล์มท็อปคอมพิวเตอร์เพื่อใช้แทนกระเป๋าเงิน ทำให้ผู้ใช้สามารถพกเงินสดดิจิตอลไปใช้กับร้านค้าโดยตรงได้

รายการอ้างอิง

1. Wayner, P. Digital Cash: Commerce on the Net. United States of America : Academic Press, 1996.
2. Sirbu, M.A. Credits and debits on the Internet. IEEE SPECTRUM. Vol 34 2 (February 1997): 23-29.
3. Ferreira, L., and Dahab, R. A Scheme for Analyzing Electronic Payment Systems. Computer Security Applications Conference.(1998):137-146.
4. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed. United Stated of America : John Wiley & Sons, 1996.
5. Denning, D. E. Cryptography and data security. United States of America : Addison-Wesley Publishing, 1982.
6. Sherif, M.H. SET and SSL : Electronic payments on the Internet. 3rd IEEE Symposium (1998): 353-358
7. RSA Laboratories. Frequently Asked Questions About Today's Cryptography, version 4.1 [Online]. RSA Data Security, 1995. Available from: http://www.rsasecurity.com/rsalabs/faq/index.html
8. Rivest, R. The MD5 Message Digest Algorithm. Request for Comments: 1321, MIT Laboratory for Computer Science and RSA Data Security, 1992.
9. Baumeler, E., and Rufer, S. Digital Money [Online]. Biel School of Engineering, Computer Science Department, 1999. Available from: http://www.hta-bi.bfh/~rfs/DigitalMoney/DigitalMoney.html
10. Law,L., Sabett,S., and Solinas, J. How to make a mint: the cryptography of anonymous electronic cash [Online]. National Security Agency Office of Information Security Research and Technology, 1996. Available from: http://www.swiss.ai.mit.edu/6805/articles/money/nsamint/nsamint.htm
11. Chaum, D., and Brands, S. 'Minting' electronic cash. IEEE Spectrum Vol 34 2 (February 1997) : 30-34
12. Gemmell, P.S. Traceable e-cash. IEEE Spectrum Vol 34 2 (February 1997) : 35-37.
13. Wang, H., and Zhang, Y. Untraceable Off-line Electronic Cash Flow in E-Commerce. Computer Science Conference (2001): 191-198.

14. Brands, S. Electronic Cash on the Internet. Network and Distributed System Security. (1995): 64-84.
15. Higgins, G.R.L. Electronic cash in global world. European Conference (1997): 86.
16. Varadharajan, V., and Mu, Y. On the Design of Secure Electronic Payment Schemes for Internet. Computer Security Applications Conference (1996): 78-87.
17. Nguyen, K.Q., Mu, Y., and Varadharajan, V. Secure and Efficient Digital Coins. Computer Security Applications Conference (1997): 9-15.
18. Schoenmakers, B. Basic Security of the ecash™ Payment System. [Online]. DigiCash, 1996. Available from: http://www.digicash.com
19. Lynch, D.C., and Lundquist, L. Digital money: the new era of Internet commerce. United Stated of America : John Wiley & Sons, 1996.
20. Gutmann, P. Cryptlib Security Toolkit Version 2.1 final beta [Computer Software]. (n.p.), 1999. Available from: http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

บรรณานุกรม

ภาษาอังกฤษ

1. Garfinkel, S. PGP : Pretty Good Privacy. 1st ed. United States of America : O'Reilly & Associates, 1995.
2. Gilbert, S.D. and McCarty, B. Visual C++ 6 Programming Blue Book. United States of America : Coriolis Group, 1999.
3. Robison, L. Teach Yourself Database Programming with Visual C++ 6. United States of America : Sams Publishing, 1999.

ภาษาไทย

1. นิรุธ อำนวยศิลป์. คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0. กรุงเทพมหานคร : ซัคเซส มีเดีย, 2542.
2. ยุทธนา ลีลาศวัฒนกุล. คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับ Database Programming. กรุงเทพมหานคร : ซัคเซส มีเดีย, 2544.

ภาคผนวก

ภาคผนวก ก

คลาสที่ใช้ระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์ ซึ่งถูกพัฒนาขึ้นโดยใช้ Microsoft Visual C++ Version 6.0

1. คลาสที่เกี่ยวกับการติดต่อกันระหว่างโปรแกรม ได้แก่

CclientSocket, ClisteningSocket, CMsg, CIPAddress

2. คลาสที่เกี่ยวกับการติดต่อกับฐานข้อมูล

- CDaoDatabase, CDaoRecordset, COleVariant
- CAdodc, CDataGrid (ActiveX control)

3. คลาสที่เกี่ยวกับโพรโตคอลการไหลของเงินสดดิจิตอล

- CDigitalCash, money_order_struct, money_order_secret_struct
- ส่วนของธนาคาร : CBankApp, CBankDoc, CBankView, CGridDlg, CPassword
- ส่วนของผู้ใช้ : cash_path_struct, CCash, CDepositBox, CLogin,CPayBox, CUserApp, CUserDoc, CUserView, CValues

4. แฟ้มข้อมูลที่ใช้ในการเข้ารหัสลับ

Capi.h

Cl32.dll

Cl32.lib

ภาคผนวก ข

Source Code ที่เพิ่มเติมในแฟ้มข้อมูล Crypt32.def, Capi.h, Cryptcab.c และ Crypt.c ซึ่งเป็นแฟ้มข้อมูลของ Source Code ที่ใช้ในการคอมไพล์ Cl32.dll เพื่อใช้ในต้นแบบระบบเงินสดดิจิตอลไม่ระบุชื่อแบบตามรอยได้ชนิดออฟไลน์นี้ เพื่อใช้ในการสร้างแฟ้มข้อมูลกุญแจสาธารณะ (*.pub) และใช้ในโพรโตคอล Blind Signature

Crypt32.def
CryptQueryRSA - เป็นฟังก์ชันที่ใช้ในการดึงพารามิเตอร์ของคู่กุญแจสาธารณะและกุญแจส่วนตัวซึ่งสร้างโดย RSA cryptBlindRSA - เป็นฟังก์ชันที่ใช้ในโพรโตคอล Blind Signature เพื่อทำการ Blind ข้อมูล cryptUnblindRSA - เป็นฟังก์ชันที่ใช้ในโพรโตคอล Blind Signature เพื่อทำการ Unblind ข้อมูล

Capi.h
CRET cryptQueryRSA(const CRYPT_HANDLE cryptHandle, CRYPT_PKCINFO_RSA * pRSAkey); CRET cryptBlindRSA(const CRYPT_CONTEXT cryptContext, void CPTR buffer, void CPTR random, const int length_b, const int length_r); CRET cryptUnblindRSA(const CRYPT_CONTEXT cryptContext, void CPTR buffer, void CPTR random, const int length_b, const int length_r);

Cryptcab.c

```
CRET cryptQueryRSA( const CRYPT_HANDLE cryptHandle,
                    CRYPT_PKCINFO_RSA * pRSAkey )  {
    CRYPT_CONTEXT context;
    CRYPT_INFO *cryptInfoPtr;
    int status;
    status = krnlSendMessage( cryptHandle,
                              RESOURCE_MESSAGE_GETDATA,
                              &context,
                              RESOURCE_MESSAGE_DATA_CONTEXT,
                              CRYPT_BADPARM1 );
    if( cryptStatusError( status ) )  return( status );
    getCheckInternalResource( context,
                              cryptInfoPtr,
                              RESOURCE_TYPE_CRYPT );
    if( checkBadPtrWrite( pRSAkey, sizeof( CRYPT_PKCINFO_RSA ) ) )
            unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM2 );
    if( cryptInfoPtr->capabilityInfo->cryptAlgo != CRYPT_ALGO_RSA )
            unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM1 );
     pRSAkey->endianness = 0;
     pRSAkey->isPublicKey = cryptInfoPtr->ctxPKC.isPublicKey;
     BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_n, pRSAkey->n );
     PRSAkey->nLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_n );
     BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_e, pRSAkey->e );
     PRSAkey->eLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_e );
     BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_d, pRSAkey->d );
     PRSAkey->dLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_d );
     BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_p, pRSAkey->p );
     PRSAkey->pLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_p );
```

Cryptcab.c (ต่อ)

```
        BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_q, pRSAkey->q );
        PRSAkey->qLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_q );
        BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_u, pRSAkey->u );
        PRSAkey->uLen = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_u );
        BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_exponent1, pRSAkey->e1 );
        PRSAkey->e1Len = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_exponent1 );
        BN_bn2bin( cryptInfoPtr->ctxPKC.rsaParam_exponent2, pRSAkey->e2 );
        PRSAkey->e2Len = BN_num_bits( cryptInfoPtr->ctxPKC.rsaParam_exponent2 );
        UnlockResourceExit( cryptInfoPtr, CRYPT_OK );
}
```

Crypt.c

```
int MulModN( CRYPT_INFO *cryptInfo,
             BYTE *buffer1,
             BYTE *buffer2,
             int noBytes1,
             int noBytes2 ) {
        BN_CTX *bnCTX;
        BIGNUM *n = cryptInfo->ctxPKC.rsaParam_n;
        BIGNUM *data, *m1, *m2;
        Int length1 = bitsToBytes( cryptInfo->ctxPKC.keySizeBits );
        Int length2 = bitsToBytes( cryptInfo->ctxPKC.keySizeBits );
        Int status = CRYPT_OK;
        If( noBytes1 != CRYPT_USE_DEFAULT && noBytes1 != length1 ) {
                if( noBytes1 > length1 && !*buffer1 ) {
                        buffer1++;
                        noBytes1--;
                 }
                 if( noBytes1 > length1 ) return( CRYPT_BADDATA );
```

Crypt.c (ต่อ)

```
                Length1 = noBytes1;
                }
        if( noBytes2 != CRYPT_USE_DEFAULT && noBytes2 != length2 ) {
                if( noBytes2 > length2 && !*buffer2 ) {
                        buffer2++;
                        noBytes2--;
                }
                if( noBytes2 > length2 ) return( CRYPT_BADDATA );
                length2 = noBytes2;
        }
        if( ( bnCTX = BN_CTX_new() ) == NULL ) return( CRYPT_NOMEM );
        data = BN_new();
        m1 = BN_new();
        m2 = BN_new();
        BN_bin2bn( buffer1, length1, m1 );
        BN_bin2bn( buffer2, length2, m2 );
        Zeroise( buffer1, length1 );
        BN_mod_mul( data, m1, m2, n, bnCTX )
        Length1 = BN_bn2bin( data, buffer1 );
        BN_clear_free( data );
        BN_clear_free( m1 );
        BN_clear_free( m2 );
        BN_CTX_free( bnCTX );
        if( noBytes1 == CRYPT_USE_DEFAULT ) length1 = CRYPT_OK;
        return( ( status == -1 ) ? CRYPT_PKCCRYPT : length1 );
}
```

Crypt.c (ต่อ)

```
int ChkInvR(CRYPT_INFO *cryptInfo,
            BYTE *buffer1,
            int noBytes1) {
        BN_CTX *bnCTX;
        BIGNUM *n = cryptInfo->ctxPKC.rsaParam_n;
        BIGNUM *data,*test,*one;
        int status = CRYPT_OK;
        unsigned char tmp[]={1};
        one = BN_new();
        BN_bin2bn( tmp, 1, one );
        if( ( bnCTX = BN_CTX_new() ) == NULL ) return( CRYPT_NOMEM );
        data = BN_new();
         test = BN_new();
        BN_bin2bn( buffer1, noBytes1, data );
        Zeroise( buffer1, noBytes1 );
        do {
                BN_gcd(test,data,n,bnCTX);
                BN_bn2bin( test, buffer1 );
                BN_add(data,data,one);
        }while( *buffer1 != 1);
        BN_sub(data,data,one);
        BN_bn2bin( data, buffer1 );
        BN_clear_free( data );
        BN_clear_free( test );
        BN_CTX_free( bnCTX );
        Return( ( status == -1 ) ? CRYPT_PKCCRYPT : noBytes1 );
}
```

Crypt.c (ต่อ)

```
CRET cryptBlindRSA( const CRYPT_CONTEXT cryptContext,
                    void CPTR buffer,
                    void CPTR random,
                    const int length_b,
                    const int length_r ) {
    CRYPT_INFO *cryptInfoPtr;
     Int status;
    Unsigned char* random_temp;
    GetCheckResource( cryptContext,
                    CryptInfoPtr,
                    RESOURCE_TYPE_CRYPT,
                    CRYPT_BADPARM1 );
    If( cryptInfoPtr->capabilityInfo->cryptAlgo != CRYPT_ALGO_RSA )
                UnlockResourceExit( cryptInfoPtr, CRYPT_BADPARM1 );
    if( checkBadPtrQ( buffer ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM2 );
    if( checkBadPtrQ( random ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM3 );
if( length_b != CRYPT_USE_DEFAULT )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM4 );
    if( needsKey( cryptInfoPtr ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_NOKEY );
    status = ChkInvR( cryptInfoPtr,random,length_r);
    if( cryptStatusError( status ) ) unlockResourceExit( cryptInfoPtr, status );
    random_temp = malloc(sizeof(unsigned char)*length_r);
    memcpy(random_temp, random, length_r);
```

Crypt.c (ต่อ)

```
If( cryptInfoPtr->capabilityInfo->encryptFunction != NULL )
                    status = cryptInfoPtr->capabilityInfo->
                              encryptFunction( cryptInfoPtr,
                                                  random_temp,
                                                  length_r );
          else
                    status = cryptInfoPtr->capabilityInfo->
                              signFunction( cryptInfoPtr,
                                                  random_temp,
                                                  length_r );
          if( cryptStatusError( status ) ) {
                    free(random_temp);
                    unlockResourceExit( cryptInfoPtr, status );
           }
          status = MulModN( cryptInfoPtr,
                              buffer,
                              random_temp,
                              length_b,
                              CRYPT_USE_DEFAULT);
          Free(random_temp);
          UnlockResourceExit( cryptInfoPtr, status );
}
int DivModN( CRYPT_INFO *cryptInfo,
                    BYTE *buffer1,
                    BYTE *buffer2,
                    int noBytes1,
                    int noBytes2 ) {
```

Crypt.c (ต่อ)

```
BN_CTX *bnCTX;
BIGNUM *n = cryptInfo->ctxPKC.rsaParam_n;
BIGNUM *data, *m1, *m2, *u;
int length1 = bitsToBytes( cryptInfo->ctxPKC.keySizeBits );
int length2 = bitsToBytes( cryptInfo->ctxPKC.keySizeBits );
int status = CRYPT_OK;
if( noBytes1 != CRYPT_USE_DEFAULT && noBytes1 != length1 ) {
        if( noBytes1 > length1 && !*buffer1 ) {
                buffer1++;
                noBytes1--;
        }
        if( noBytes1 > length1 ) return( CRYPT_BADDATA );
        length1 = noBytes1;
}
if( noBytes2 != CRYPT_USE_DEFAULT && noBytes2 != length2 ) {
        if( noBytes2 > length2 && !*buffer2 ) {
                buffer2++;
                noBytes2--;
        }
        if( noBytes2 > length2 ) return( CRYPT_BADDATA );
        length2 = noBytes2;
}
if( ( bnCTX = BN_CTX_new() ) == NULL ) return( CRYPT_NOMEM );
data = BN_new();
m1 = BN_new();
m2 = BN_new();
u = BN_new();
BN_bin2bn( buffer1, length1, m1 );
```

Crypt.c (ต่อ)

```
        BN_bin2bn( buffer2, length2, m2 );
        Zeroise( buffer1, length1 );
        u = BN_mod_inverse( m2, n, bnCTX );
        BN_mod_mul( data, m1, u, n, bnCTX )
        length1 = BN_bn2bin( data, buffer1 );
        BN_clear_free( data );
        BN_clear_free( m1 );
        BN_clear_free( m2 );
        BN_clear_free( u );
        BN_CTX_free( bnCTX );
        if( noBytes1 == CRYPT_USE_DEFAULT ) length1 = CRYPT_OK;
        return( ( status == -1 ) ? CRYPT_PKCCRYPT : length1 );
}
CRET cryptUnblindRSA( const CRYPT_CONTEXT cryptContext,
                      void CPTR buffer,
                      void CPTR random,
                      const int length_b,
                      const int length_r ) {
        CRYPT_INFO *cryptInfoPtr;
        int status;
        getCheckResource( cryptContext,
                          cryptInfoPtr,
                          RESOURCE_TYPE_CRYPT,
                          CRYPT_BADPARM1 );
        if( cryptInfoPtr->capabilityInfo->cryptAlgo != CRYPT_ALGO_RSA )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM1 );
        if( checkBadPtrQ( buffer ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM2 );
```

Crypt.c (ต่อ)

```
        if( checkBadPtrQ( random ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM3 );
        if( length_b != CRYPT_USE_DEFAULT )
                unlockResourceExit( cryptInfoPtr, CRYPT_BADPARM4 );
        if( needsKey( cryptInfoPtr ) )
                unlockResourceExit( cryptInfoPtr, CRYPT_NOKEY );
        status = DivModN( cryptInfoPtr,
                          buffer,
                          random,
                          length_b,
                          length_r);
        unlockResourceExit( cryptInfoPtr, status );
}
```

ประวัติผู้เขียนวิทยานิพนธ์

นายสุขุม เจียมชูโรจน์ เกิดเมื่อวันที่ 7 พฤศจิกายน พ.ศ.2518 ที่จังหวัดกรุงเทพมหานคร เข้าศึกษาในหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2537 สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2540 จากนั้น ได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่ห้องปฏิบัติการไฟฟ้าสื่อสาร สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541