โปรแกรมการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจลับเชิงควอนตัม

A SIMULATION SOFTWARE FOR ERROR CORRECTION IN QUANTUM KEY DISTRIBUTION

โดย

นายจตุพิทย์ จึงศิรกุลวิทย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

A SIMULATION SOFTWARE FOR ERROR CORRECTION IN QUANTUM KEY DISTRIBUTION

BY

MR. JATUPIT CHEUNGSIRAKULWIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEER

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อปริญญานิพนธ์	โปรแกรมแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม
TITLE	A Simulation Software for Error Correction in Quantum Cryptography
นักศึกษา	นายจตุพิทย์ จึงศิรกุลวิทย์ รหัสประจำตัว 47010082
อาจารย์ที่ปรึกษา	รศ.อรลาภ แสงอรุณ ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2550

ปริญญานิพนธ์นี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(รศ. อรลาภ แสงอรุณ)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	โปรแกรมการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัม
ชื่อนักศึกษา	นายจตุพิทย์ จึงศิรกุลวิทย์ รหัสประจำตัว 47010082
อาจารย์ที่ปรึกษา	รศ.อรลาภ แสงอรุณ ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2550

บทคัดย่อ

ปริญญานิพนธ์นี้เสนอ การศึกษากระบวนกระจายกุญแจรหัสลับเชิงควอนตัม และการแก้ไขความผิดพลาดของกระจายกุญแจรหัสลับเชิงควอนตัม เนื่องจากสัญญาณรบกวนที่เกิดในเส้นทางการสื่อสารระหว่างทางผู้ส่งและทางผู้รับ โดยใช้การแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE และ เนื่องจากกระบวนแก้ไขความผิดพลาดได้กระทำผ่านทางช่องทางสาธารณะ ซึ่งอาจถูกดักจับข้อมูลได้ จึงได้ใช้กระบวนการขยายสภาวะส่วนตัว เพื่อลดความสำคัญของกุญแจรหัสลับบางส่วนลง จากนั้นเขียนโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมนั้น ผลจากการจำลองระบบพบว่า วิธีการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE สามารถแก้ไขความผิดพลาด ที่เกิดขึ้นได้ดี มีการเปิดเผยข้อมูลที่เกี่ยวกับกุญแจรหัสลับน้อย แต่ต้องใช้จำนวนรอบในการติดต่อสื่อสารที่มาก ไม่เหมาะกับการติดต่อสื่อสารที่ต้องการความเร็วสูง จึงเป็นแนวทางในการพัฒนาโพรโทคอลการแก้ไขความผิดพลาดให้มีประสิทธิภาพสูงขึ้นต่อไป

Thesis Title A Simulation Software for Error Correction in Quantum Cryptography

Student Mr. Jatupit Cheungsirakulwit ID. 47010082

Advisor Assoc. Prof. Ornlarp Sangaroon

Dr. Keattisak Sripimanwat

Graduate Level Bachelor Degree of Information Engineering

Department Information Engineering

Academic Year 2007

ABSTRACT

This project presents the study of Quantum Cryptography or Quantum Key Distribution (QKD) and the methods of error correcting in Quantum Key Distribution , which is degraded from various kinds of noise occurring in Quantum channel. we proposes the error correcting method by using cascade protocols to reduce a number of noises . As an error correcting process on public channel, some informations over public channel are able to recover. Privacy amplification that wipe out information by reducing the key length. From this studies , the methods of error correcting in Quantum Key Distribution with cascade method are expression by means of software simulation. The simulation results shows good performance at low QBER , it becomes an interesting topic to explore for further protocol development and improvement of error correcting in Quantum Key Distribution .

กิตติกรรมประกาศ

ในการจัดทำปริญญานิพนธ์เล่นนี้ให้บรรลุตามจุดประสงค์ที่วางไว้และเสร็จสิ้นตรงตามเวลานั้น ข้าพเจ้าต้องขอขอบพระคุณในความร่วมมือและการสนับสนุนจากหลายท่านดังนี้

ขอขอบคุณ รศ.อรลาภ แสงอรุณ อาจารย์ผู้ควบคุมปริญญานิพนธ์ ภาควิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และ ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์ อาจารย์ผู้ควบคุมปริญญานิพนธ์ร่วมและนักวิจัยจากศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ในการให้ความรู้ คำแนะนำและคำปรึกษาที่เป็นประโยชน์ต่อการทำปริญญานิพนธ์เล่มนี้ ทางผู้เขียนรู้สึกซาบซึ้งในความอนุเคราะห์และขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณนางวุฒิกรณ์ ตรัยศิลานันท์ นักศึกษาปริญญาโท ภาควิชาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ความรู้และความช่วยเหลือเกี่ยวกับวิทยาการรหัสลับเชิงควอนตัมเป็นอย่างดี

ขอขอบคุณทีมนักวิจัยภายในหน่วยปฏิบัติการวิจัยการสื่อสารเชิงแสงและควอนตัม (OQC) ภายใต้โครงการวิยาการรหัสลับเชิงควอนตัม (T54902) และผู้ช่วยนักวิจัยท่านอื่นๆ ภายในหน่วยปฏิบัติการวิจัยการสื่อสารเชิงแสงและควอนตัม ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ที่ให้ความช่วยเหลือเป็นอย่างดี

นอกจากนี้ยังมีท่านคณาจารย์ภาควิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้ คำแนะนำ รวมทั้งความเห็นต่างๆ ซึ่งคุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ทางผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่านไว้ ณ โอกาสนี้

จตุพิทย์ จึงศิรกุลวิทย์

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

ปัจจุบันเทคโนโลยีการสื่อสารผ่านระบบเครือข่ายอินเตอร์เน็ตที่ได้รับความนิยมและได้กลายเป็นสิ่งที่มีความจำเป็นในการดำเนินกิจกรรมต่างๆในชีวิตประจำวันของมนุษย์มากยิ่งขึ้น ดังเช่นการเปิดให้บริการข่าวสารแบบออนไลน์(Online) การรับส่งข้อมูล เกมออนไลน์ หรือบริการที่มีการทำธุรกรรมทางการเงินแก่ผู้ใช้บริการแบบออนไลน์ในหลากหลายประเภท ซึ่งการใช้บริการผ่านทางเครือข่ายสาธารณะนี้แม้จะมีข้อดีอยู่มาก ในการที่ทำให้การรับส่งข้อมูลนั้นมีความรวดเร็วมีความถูกต้องและมีต้นทุนที่ประหยัด แต่การติดต่อสื่อสารบางอย่างยังต้องการความปลอดภัยของข้อมูล ซึ่งในเครือข่ายสาธารณะนั้นแฝงไปด้วยผู้ประสงค์ร้ายที่เข้ามาใช้เพื่อหาประโยชน์ใส่ตนเช่น การโจมตีระบบทำให้ระบบเครือข่ายใช้งานไม่ได้ การเข้ามาขโมยข้อมูลระหว่างการใช้บริการ ซึ่งปัญหาเหล่านี้เป็นปัญหาสำคัญของการส่งข้อมูลผ่านเครือข่ายสาธารณะ ส่งผลให้เกิดความเสียหายต่อระบบเศรษฐกิจหรือความมั่นคงของประเทศตามมาหากข้อมูลทางอิเล็กทรอนิกส์ที่สำคัญ เช่น ข้อมูลทางการเงิน ข้อมูลทางทหาร มีผู้ขโมยและรับรู้ข้อความที่ทำการส่งเหล่านั้นเพื่อเป็นการป้องกันปัญหาเหล่านี้ วิทยาการรหัสลับจึงถูกนำมาใช้เพื่อรักษาความลับข้อมูลที่ส่งผ่านเครือข่ายสาธารณะไม่ให้บุคคลที่ไม่ได้รับอนุญาตหรือบุคคลที่สามรับรู้ข้อมูลระหว่างการส่งหรือนำข้อมูลที่ส่งไปใช้ในทางที่ผิดกฎหมาย ซึ่งวิทยาการรหัสลับที่ใช้อยู่ในปัจจุบันทั้งวิทยาการรหัสลับแบบสมมาตร(Systematic Cryptography) และวิทยาการรหัสลับแบบอสมมาตร(Asystematic Cryptography) อาศัยความซับซ้อนในการคำนวณเพื่อป้องกันและรักษาความลับข้อมูลที่ส่ง ถ้าการพัฒนาเทคโนโลยีการคำนวณมีความก้าวหน้าเพิ่มมากขึ้นเช่น การพัฒนาควอนตัมคอมพิวเตอร์ (Quantum Computer) ที่สามารถประมวลผลได้เร็วกว่าคอมพิวเตอร์ที่ใช้อยู่ในปัจจุบันด้วยเหตุนี้ทำให้วิทยาการรหัสลับที่ใช้อยู่ในปัจจุบันอาจจะไม่สามารถป้องกันและรักษาความลับข้อมูลได้อีกต่อไป วิทยาการรหัสลับเชิงควอนตัม (Quantum Cryptography) เสนอขึ้นเพื่อใช้ในการส่งกุญแจรหัสลับ (Secret Key) ระหว่างผู้ส่งและผู้รับก่อนจะนำกุญแจรหัสลับนี้มาใช้ร่วมกับวิทยาการรหัสลับในปัจจุบันเช่น One time pad เป็นต้น เพื่อเข้ารหัสลับและถอดรหัสลับข้อมูลที่ต้องการส่งผ่านระบบเครือข่ายสาธารณะ การส่งกุญแจรหัสลับในระบบวิทยาการรหัสลับเชิงควอนตัมอาศัยคุณสมบัติเชิงควอนตัมของแสง เพื่อใช้แทนกุญแจรหัสลับและใช้ทฤษฎีกลศาสตร์ควอนตัม (Quantum Mechanics) ช่วยยืนยันความปลอดภัยของระบบเมื่อมีบุคคลที่สามเข้ามาขโมยสถานะ

ควอนตัมของแสงเหล่านี้จะทำให้ผู้ส่งและผู้รับทราบทันทีถึงการเข้ามารบกวนระบบ ทำให้ผู้ส่งและผู้รับยกเลิกการส่งกุญแจรหัสลับได้ทันก่อนจะเกิดความเสียหายตามมาในภายหลังได้

1.1 แนวคิดและที่มา

วิทยาการรหัสลับเชิงควอนตัม (Quantum Cryptography) หรือการกระจายกุญแจรหัสลับเชิงควอนตัม (Quantum Key Distribution) นำเสนอครั้งแรกในปี ค.ศ. 1984 เรียกว่าโพรโทคอล BB84 [1] ซึ่งเป็นแนวคิดในการส่งกุญแจรหัสลับโดยอาศัยโพลาไรเซชัน (Polarization) ของแสงที่ไม่ตั้งฉากกันสองเวกเตอร์ฐาน (Basis) แทนกุญแจรหัสลับบิตตามระบบสื่อสารดิจิทัล (บิต "0" และบิต "1") และหลังจากนั้นในปี ค.ศ. 1992 C.H. Bennett และคณะ ได้แสดงระบบวิทยาการรหัสลับระบบแรกที่ใช้โพรโทคอล BB84 เพื่อส่งกุญแจรหัสลับผ่านอากาศระยะทาง 30 เซนติเมตร ด้วยอัตราเร็วในการส่งกุญแจรหัสลับ 10 บิตต่อวินาที [1] หลังจากนั้นวิทยาการรหัสลับเชิงควอนตัมกลายเป็นที่สนใจแก่นักวิจัยทั่วโลก ได้ทำการวิจัยเพื่อให้ระบบวิทยาการรหัสลับเชิงควอนตัมสามารถส่งกุญแจรหัสลับได้ระยะทางไกล อัตราเร็วในการส่งกุญแจรหัสลับสูงขึ้นและสามารถใช้ร่วมกับระบบเครือข่ายสื่อสารดิจิทัลในปัจจุบันได้ โดยนิตยสารและหน่วยงานที่สำคัญของโลก เช่น MIT นิตยสาร PC Magazine และ RAND ได้จัดให้วิทยาการรหัสลับเชิงควอนตัมเป็นหนึ่งในเทคโนโลยีที่น่าจับตามองในอนาคต ตัวอย่างงานวิจัยเกี่ยวกับระบบวิทยาการรหัสลับเชิงควอนตัมที่สำคัญของโลกได้แก่ หน่วยงาน Defense Advanced Research Projects Agency (DARPA) เป็นหน่วยงานสังกัดกระทรวงกะลาโหมของประเทศสหรัฐอเมริกาได้ให้ทุนสนับสนุนงานวิจัยเกี่ยวกับระบบเครือข่ายวิทยาการรหัสลับเชิงควอนตัม เรียกว่า "DARPA Quantum Network" โดยแรกเริ่มเป็นความร่วมมือระหว่างมหาวิทยาลัยบอสตัน (Boston University) มหาวิทยาลัยฮาร์วาด (Harvard University) และ BBN Technology ได้สร้างระบบเครือข่ายวิทยาการรหัสลับเชิงควอนตัมระบบแรกของโลกเมื่อปี ค.ศ. 2003 [2] [3] จากนั้นในปี ค.ศ. 2005 บริษัท QinetiQ ได้เข้าร่วมงานการวิจัยเพื่อสร้างเครือข่ายวิทยาการรหัสลับเชิงควอนตัมด้วย [4] นอกจากนี้สหภาพยุโรปมีโครงการวิจัยเกี่ยวกับวิทยาการรหัสลับเชิงควอนตัมที่ชื่อว่า Development of a Global Network for Secure Communication based on Quantum Cryptography (SECOQC) โดยเป็นความร่วมมือจากสถาบันวิจัยต่างๆ ภายในสหภาพยุโรปซึ่งลงทุนประมาณ 11 ล้านยูโรเพื่อดำเนินการกำหนดมาตรฐานระบบวิทยาการรหัสลับเชิงควอนตัมและบังคับใช้ภายในระยะเวลา 4 ปี [5] เป็นต้น ซึ่งในประเทศไทยนั้นได้มีการทำวิจัยเพื่อสร้างต้นแบบระบบวิทยาการรหัสลับเชิงควอนตัม โดยหน่วยปฏิบัติการวิจัยการสื่อสารเชิงแสงและควอนตัม (OQC) ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ภายใต้รหัสโครงการ TS4902 ชื่อโครงการ " ระบบวิทยาการรหัส

ลับเชิงควอนตัม" ได้กำลังศึกษาและทำการสร้างฮาร์ดแวร์ระบบวิทยาการรหัสลับเชิงควอนตัมผ่านอากาศ และซอร์ฟแวร์กระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมโดยใช้รหัส BCH [6][7] ดังนั้นเพื่อเป็นการแสดงการทำงานของกระบวนการรับ-ส่งกุญแจรหัสลับผ่านช่องสื่อสารเชิงควอนตัม (Quantum Channel) และกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมให้ละเอียดมากยิ่งขึ้น วิทยานิพนธ์เล่มนี้นำเสนอซอฟต์แวร์การทำงานของระบบวิทยาการรหัสลับเชิงควอนตัม การแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE และการขยายสภาวะส่วนตัว ซึ่งเป็นหนึ่งในขั้นตอนการทำงานที่สำคัญในระบบบวิทยาการรหัสลับเชิงควอนตัมเพื่อแก้ไขความผิดพลาดที่เกิดขึ้นและลดความสามารถในการสร้างกุญแจรหัสลับใหม่ของบุคคลที่สาม ซึ่งการพัฒนาโปรแกรมจำลองการทำงานนี้เป็นการพัฒนาร่วมกับหน่วยปฏิบัติการวิจัยการสื่อสารเชิงแสงและควอนตัม (OQC) ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ภายใต้โครงการ T54902 ระบบวิทยาการรหัสลับเชิงควอนตัม

1.2 วัตถุประสงค์

ปริญญานิพนธ์นี้เสนอการจำลองการทำงานกระบวนแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE ทำให้กุญแจรหัสลับที่ใช้มีความถูกต้องพร้อมที่จะนำไปใช้ในวิทยาการรหัสลับเชิงควอนตัม โดยมีวัตถุประสงค์ดังต่อไปนี้

1.2.1 เพื่อศึกษาวิทยาการรหัสลับเชิงควอนตัม

1.2.2 เพื่อศึกษากระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม

1.2.3 เพื่อศึกษาการทำงานของการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE

1.2.4 เพื่อจัดทำโปรแกรมจำลองการทำงานกระบวนการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

1.2.5 เพื่อแสดงการทำงานกระบวนการแก้ไขความผิดพลาดในระบบวิทยาการรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

1.3 ขอบเขตของโครงงาน

ปริญญานิพนธ์เล่มนี้ได้นำเสนอโปรแกรมจำลองการทำงานกระบวนการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE ซึ่งมีกระบวนการทำงานของโปรแกรมที่เขียนขึ้นจะมีการจำลองกระบวนการกระจายกุญแจรหัสลับซึ่งประกอบด้วย

การกำหนดขนาดของกุญแจรหัสลับที่จะรับส่งกัน การสร้างกุญแจรหัสลับ การรับส่งกุญแจรหัสลับ การตรวจสอบเวกเตอร์ฐาน จากนั้นจะเป็นกระบวนการแก้ไขความผิดพลาดตามหลักการทำงานของโพรโทคอล CASCADE และการขยายสภาวะส่วนตัวด้วยการนำกุญแจรหัสลับสองบิตที่อยู่ติดกันมาทำการ XOR กันเมื่อเสร็จสิ้นทุกกระบวนการแล้วจะได้ชุดกุญแจรหัสลับเพื่อนำไปใช้งานต่อไป

1.4 ขั้นตอนการดำเนินโครงงาน

การออกแบบจำลองการทำงานของโปรแกรมจำลองการทำงานกระบวนการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัมมีขั้นตอนการทำงานดังต่อไปนี้

1.4.1 ศึกษาวิทยาการรหัสลับเชิงควอนตัม โพรโทคอลที่ใช้ในการส่งกุญแจรหัสลับและการประยุกต์ระบบวิทยาการรหัสลับเชิงควอนตัม

1.4.2 ศึกษากระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมและ การทำงานของโพรโทคอล CASCADE

1.4.3 ออกแบบขั้นตอนและแผนภาพการทำงานของโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

1.4.4 ออกแบบแอพพลิเคชั่นของโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

1.4.5 เขียนโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

1.4.6 จัดทำเอกสารเกี่ยวกับวิทยาการรหัสลับเชิงควอนตัมที่ได้ทำการศึกษา และการใช้งานโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม

1.5 รายละเอียดของปริญญานิพนธ์

ปริญญานิพนธ์เล่มนี้แบ่งเนื้อหาออกเป็นบทย่อย 5 บทซึ่งรายละเอียดของแต่ละบทมีดังต่อไปนี้

บทที่ 1 แสดงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ ขอบเขตและขั้นตอนการทำโครงงาน

บทที่ 2 แสดงพื้นฐานวิทยาการรหัสลับเชิงควอนตัม กระบวนการกระจายกุญแจรหัสลับ กระบวนการกลั่นกุญแจรหัสลับ และโพรโทคอลแก้ไขความผิดพลาดในระบบวิทยาการรหัสลับเชิงควอนตัม เช่น โพรโทคอล BBBSS โพรโทคอล CASCADE

บทที่ 3 เสนอการออกแบบโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

บทที่ 4 แสดงขอบเขต ข้อจำกัด วิธีการใช้งาน และผลการทำงานของโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล CASCADE

บทที่ 5 แสดงบทสรุปของงาน ปัญหาและอุปสรรค์ที่เกิดขึ้น และข้อเสนอแนะแนวทางในการแก้ไขปัญหา

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในปัจจุบันเทคโนโลยีด้านการสื่อสารมีความก้าวหน้าและทันสมัยมากยิ่งขึ้นตามลำดับ อันเนื่องมาจากมีเครื่องมือที่ทันสมัยซึ่งได้รับการพัฒนา เพื่อรองรับและตอบสนองการใช้งานเทคโนโลยี ส่งผลให้เทคโนโลยีด้านนี้เข้ามามีบทบาทกับชีวิตประจำวันมากขึ้น เช่น เพื่อการแลกเปลี่ยนข่าวสาร เพื่อความบันเทิง หรือเพื่อติดต่องานต่างๆ ซึ่งการสื่อสารบางอย่างนั้นจำเป็นที่จะต้องมีการเก็บรักษาความปลอดภัยของข้อมูลในการสื่อสารนั้นด้วย ตัวอย่างเช่นการทำธุรกรรมทางการเงินของธนาคารหรือการสื่อสารทางการทหาร หากข้อมูลที่ส่งไม่เป็นความลับแล้วย่อมส่งผลกระทบต่อระบบเศรษฐกิจและความมั่นคงของชาติได้ จึงต้องทำให้การสื่อสารนั้นเป็นความลับกันภายในกลุ่มเครือข่ายที่เกี่ยวข้องเท่านั้น จึงได้มีการพัฒนาระบบรักษาความปลอดภัยข้อมูลหรือที่เรียกว่า วิทยาการรหัสลับ(Cryptography) ซึ่งวิทยาการรหัสลับที่มีอยู่ในปัจจุบันนั้นได้อาศัยกระบวนการทางคณิตศาสตร์ที่ซับซ้อนมาใช้ในการเก็บรักษาความปลอดภัยข้อมูล แต่ด้วยความก้าวหน้าทางเทคโนโลยีที่มีการพัฒนาอยู่เสมอ อาจจะทำให้ในอนาคตข้อมูลที่ถูกเก็บรักษาไว้ไม่มีความปลอดภัยอีกต่อไป จึงได้มีการวิจัยและศึกษาวิทยาการรหัสลับเชิงควอนตัม (Quantum Cryptography) ขึ้นมา ซึ่งวิทยาการรหัสลับเชิงควอนตัมนี้เป็นศาสตร์ที่ว่าด้วยเรื่องของกลศาสตร์ควอนตัม ที่จะนำมาประยุกต์ใช้งานร่วมกับวิทยาการรหัสลับ ทำให้ระบบรหัสลับเชิงควอนตัมเป็นระบบรหัสลับที่ได้รับการยอมรับว่ามีความปลอดภัยมากที่สุดในปัจจุบัน

2.1 พื้นฐานระบบวิทยาการรหัสลับ

ก่อนที่จะกล่าวถึงกระบวนการแก้ไขความผิดพลาดของกุญแจรหัสลับด้วยวิธีการของโพรโทคอล CASCADE นั้นขอกล่าวถึงวิทยาการรหัสลับที่ใช้อยู่ก่อนแล้ว ปัญหาที่เกิดขึ้นของวิทยาการรหัสลับที่ใช้ในปัจจุบัน และประวัติของวิทยาการรหัสลับเชิงควอนตัมพอสังเขป เพื่อใช้เป็นพื้นฐานการทำความเข้าใจในวิทยาการรหัสลับเชิงควอนตัมต่อไป เนื่องจากวิทยาการรหัสลับ (Cryptography) เป็นเรื่องที่ว่าด้วยการจัดการข้อมูลที่เป็นความลับให้มีความปลอดภัยไม่อนุญาตให้มีการเข้าถึงข้อมูลที่ถูกเก็บไว้ได้โดยง่าย โดยนำข้อมูลเข้ามาผ่านกระบวนการเข้ารหัสลับ (Encryption) ส่งผลให้ข้อมูลนั้นอยู่ในรูปของข้อมูลที่ไม่รู้ความหมายที่เรียกว่า "Cipher Text" และเก็บรักษาไว้หรือใช้ในการสื่อสารต่อไป ซึ่งผู้ที่จะเข้ามาใช้งานต้องเป็นผู้ที่สามารถเปลี่ยนข้อมูลที่

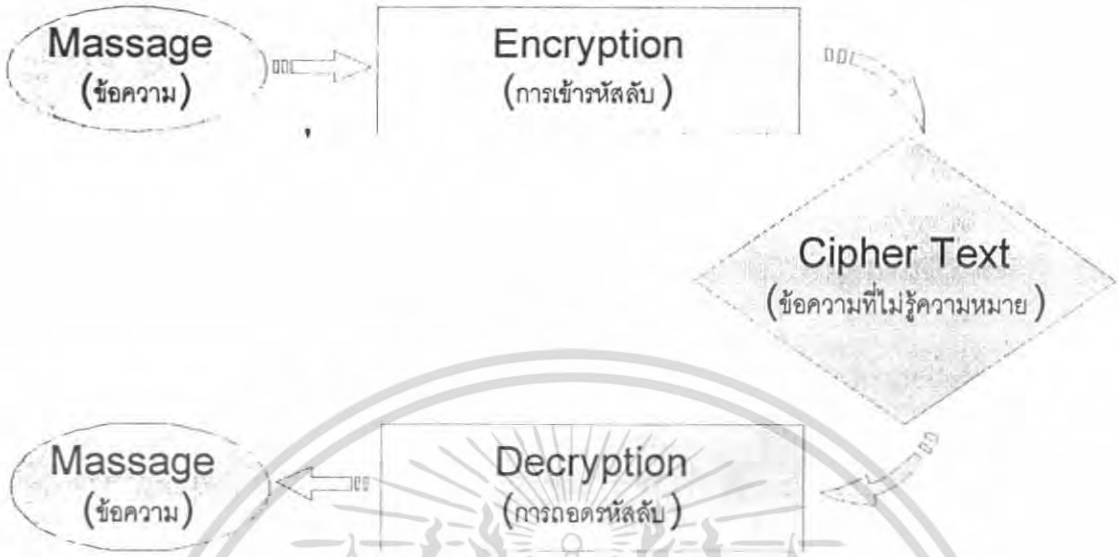
ไม่รู้ความหมายให้กลับมาอยู่ในรูปเดิมได้โดยใช้วิธีการถอดรหัสลับ (Decryption) ซึ่งการเปลี่ยนข้อมูลทั้งสองรูปแบบต้องใช้กุญแจรหัสลับ (Key) เป็นตัวกลางในการดำเนินการ โดยที่กุญแจรหัสลับต้องเป็นความลับที่อยู่ภายในกลุ่มที่ได้รับอนุญาตให้เข้าถึงได้เท่านั้น การทำงานของระบบวิทยาการรหัสลับ(Cryptography System)แสดงดังรูปที่ 2.1 วิทยาการรหัสลับที่ใช้อยู่ในปัจจุบันแบ่งได้เป็น 2 ประเภทดังต่อไปนี้คือ

2.1.1 วิทยาการรหัสลับแบบสมมาตร

วิทยาการรหัสลับแบบสมมาตร(Systematic Cryptography) เป็นกระบวนการเข้ารหัสลับรูปแบบหนึ่งที่มีการใช้กุญแจรหัสลับในการเข้ารหัสลับและถอดรหัสลับชุดเดียวกันได้แก่ รหัสลับแบบ One-Time Pad รหัสลับแบบ AES รหัสลับแบบ DES และรหัสลับแบบ 3DES เป็นต้น แต่ด้วยวิธีการที่ใช้การเข้ารหัสลับและถอดรหัสลับด้วยกุญแจรหัสลับเหมือนกัน ซึ่งการที่จะทำให้กุญแจรหัสลับของผู้ส่งและผู้รับเหมือนกันโดยไม่ถูกเปิดเผยต่อสาธารณะนั้นเป็นเรื่องยาก[6][7]

2.1.2 วิทยาการรหัสลับแบบอสมมาตร

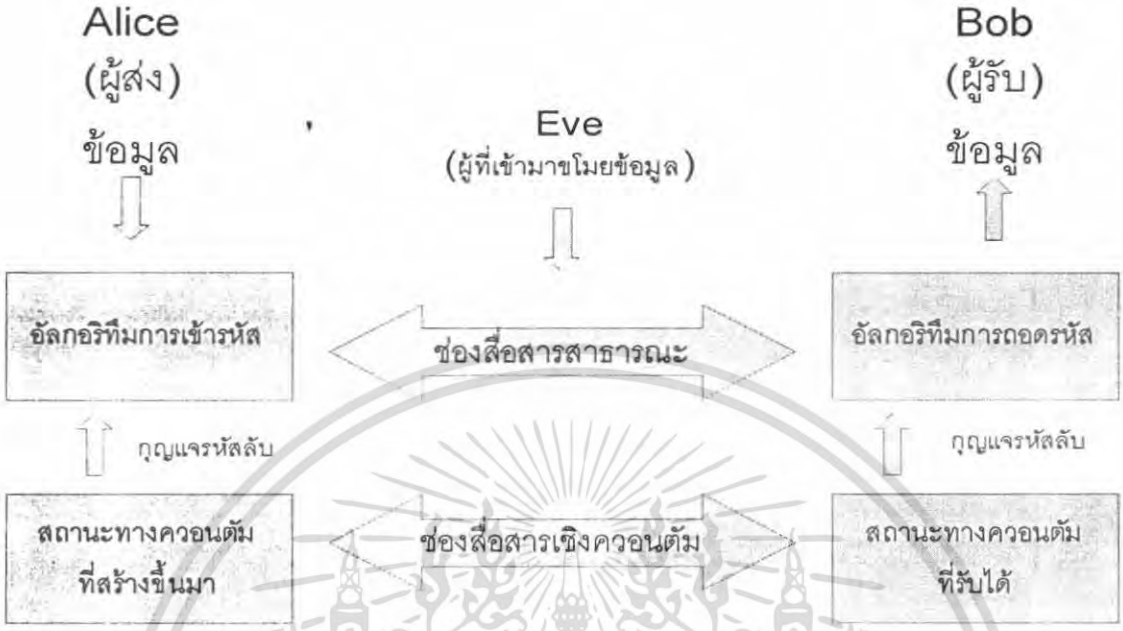
วิทยาการรหัสลับแบบอสมมาตร(Asystematic Cryptography) เป็นกระบวนการที่มีการใช้กุญแจรหัสลับต่างชุดกันในเข้ารหัสลับและถอดรหัสลับ วิธีของรหัสลับแบบอสมมาตรนี้ โดยทั่วไปจะใช้กุญแจรหัสลับที่เรียกว่า “กุญแจสาธารณะ (Public Key)” ซึ่งการนำกุญแจสาธารณะไปใช้ในการเข้ารหัสลับข้อมูลนั้น ผู้ส่งจะต้องใช้กุญแจสาธารณะให้ตรงกับกุญแจสาธารณะของผู้รับ ซึ่งเมื่อข้อมูลนั้นถูกเข้ารหัสลับเรียบร้อยและถูกส่งไปยังผู้รับ ฝ่ายผู้รับจะใช้กุญแจรหัสลับอีกชุดหนึ่งในการถอดรหัสลับ ซึ่งกุญแจรหัสลับนี้ถูกเรียกว่า “กุญแจรหัสลับส่วนตัว (Private Key)” โดยที่กุญแจรหัสลับส่วนตัวนี้จะมีเพียงบุคคลใดบุคคลหนึ่งที่เป็นเจ้าของเท่านั้น นั่นก็คือฝ่ายรับหรือผู้ที่รู้กุญแจรหัสลับส่วนตัวเท่านั้นที่จะถอดรหัสลับได้ ซึ่งวิทยาการในการเข้ารหัสลับหรือถอดรหัสลับที่จะใช้ในรูปแบบนี้คือ คุณสมบัติของจำนวนเฉพาะที่มีค่ามาก ซึ่งอาจต้องใช้เวลาในการคำนวณที่นานในการสร้างกุญแจรหัสลับที่จะนำมาใช้ถอดรหัสลับนี้ได้ ทำให้เชื่อได้ว่าวิทยาการรหัสลับแบบนี้จะสามารถรักษาข้อมูลให้เป็นความลับได้ เนื่องจากกว่าจะทำการถอดรหัสลับเพื่อให้ได้ข้อมูลชุดนั้นใช้เวลานาน จนทำให้ข้อมูลชุดนั้นไม่สามารถนำไปใช้ประโยชน์ได้อีกต่อไปจึงเรียกวิทยาการนี้ว่า "Public Key Cryptography"[6][7]



รูปที่ 2.1 กระบวนการเข้ารหัสลับและถอดรหัสลับของข้อมูล

2.2 ประวัติวิทยาการรหัสลับเชิงควอนตัม

การถือกำเนิดของวิทยาการรหัสลับเชิงควอนตัมนั้นสืบเนื่องมาจากวิทยาการรหัสลับที่ใช้อยู่ในปัจจุบัน เช่น วิทยาการรหัสลับแบบอสมมาตร หรือวิทยาการรหัสลับแบบสมมาตร ที่ยังสามารถใช้ได้อย่างมีประสิทธิภาพอยู่นั้น เมื่อมองถึงคุณสมบัติและรูปแบบเฉพาะของวิทยาการรหัสลับทั้งสองประเภทนี้ อาจกล่าวได้ว่าวิทยาการรหัสลับแบบสมมาตรนี้มีความปลอดภัยสูง ถ้ามีวิธีการส่งกุญแจลับที่ปลอดภัยสูง ส่วนวิทยาการรหัสลับแบบอสมมาตรนั้น เนื่องจากปัจจุบันเทคโนโลยีก้าวหน้าไปมากจึงง่ายต่อการพัฒนาเทคโนโลยีในการคำนวณให้มีประสิทธิภาพสูงขึ้นได้ เช่น ควอนตัมคอมพิวเตอร์ ให้สามารถนำมาใช้งานได้จริง ทำให้สามารถถอดรหัสลับได้ง่ายขึ้น ดังนั้นจึงมีการคิดค้นหากระบวนการส่งกุญแจรหัสลับรูปแบบอื่นที่ไม่อาศัยความซับซ้อนทางการคำนวณ หรือคุณสมบัติของจำนวนเฉพาะอีกต่อไป ส่งผลให้มีการนำเอาโฟตอนเดี่ยวของแสงที่มีขนาดเล็กมากมาเป็นเครื่องมือในการส่งกุญแจรหัสลับเหล่านี้ ซึ่งกระบวนการส่งกุญแจรหัสลับนี้เรียกว่า "วิทยาการรหัสลับเชิงควอนตัม " โดยการอาศัยคุณสมบัติเฉพาะตัวของโฟตอนเดี่ยวนั้นมาใช้งาน ซึ่งการกำเนิดของวิทยาการรหัสลับเชิงควอนตัมนั้น เริ่มจากบทความเรื่อง "Conjugate Coding" ที่เผยแพร่ในปี 1983 โดย Stephen Weisner และหลังจากนั้นอีกหนึ่งปีคือในปี 1984 โดย G. Bennett และ C.H. Brassard ได้เสนอโพรโทคอลในการส่งบิตสุ่มขึ้นเป็นโพรโทคอลแรก และได้ชื่อว่า "โพรโทคอล BB84" และหลังจากนั้นในปี 1992 G. Bennett และ C.H. Brassard [1] ได้แสดงระบบวิทยาการรหัสลับเชิงควอนตัมขึ้นมาเป็นระบบแรก โดยสามารถส่งบิตสุ่มได้ในระยะ



รูปที่ 2.2 ระบบวิทยาการรหัสลับเชิงควอนตัม

32 เซนติเมตร ด้วยอัตราเร็วในการส่งที่ต่ำมากเพียงไม่กี่บิตต่อวินาที ซึ่งเป็นจุดเริ่มต้นของนักวิจัยในการศึกษาและวิจัยเรื่องของวิทยาการรหัสลับเชิงควอนตัม

2.3 วิทยาการรหัสลับเชิงควอนตัม

ระบบวิทยาการรหัสลับเชิงควอนตัม (Quantum Cryptography) หรือ การกระจายกุญแจรหัสลับเชิงควอนตัม (Quantum Key Distribution: QKD) เป็นระบบในการส่งกุญแจรหัสลับ (Secret Key) โดยแทนด้วยสถานะทางควอนตัมของแสง ระหว่างผู้ส่ง (โดยทั่วไปจะเรียกว่า "*Alice*") และผู้รับ (โดยทั่วไปจะเรียกว่า "*Bob*") ผ่านทางช่องสื่อสารเชิงควอนตัม (Quantum Channel) เช่น เส้นใยนำแสง (Fiber Optic) หรือ อากาศ (Free Space) โดยอาศัยกฎทางควอนตัมฟิสิกส์ช่วยยืนยันความปลอดภัยของการส่ง ซึ่งกุญแจรหัสลับจะถูกใช้ในวิทยาการรหัสลับที่ใช้กันอยู่ในปัจจุบัน เช่น One-Time Pad เพื่อรักษาความปลอดภัยของข้อมูลที่ส่งผ่านระบบเครือข่ายสื่อสาร

โดยที่การทำงานของระบบวิทยาการรหัสลับเชิงควอนตัมนั้นมีการแบ่งการทำงานออกเป็นสองกระบวนการหลัก ได้แก่ กระบวนการส่งกุญแจรหัสลับผ่านทางช่องสื่อสารเชิงควอนตัม (Quantum Channel) ซึ่งในกระบวนการนี้หากมีบุคคลที่สาม (โดยทั่วไปเรียกว่า *Eve*) เข้ามาขโมยสถานะทางควอนตัมจำนวนมากจะทำให้ *Alice* และ *Bob* ทราบการบุกรุกของ *Eve* ได้ทันทีจาก

อัตราการผิดของกุญแจรหัสลับที่เพิ่มขึ้นทำให้สามารถยกเลิกกุญแจรหัสลับชุดนั้นได้ แต่ถ้าหาก *Eve* ขโมยจำนวนสถานะทางควอนตัมเพียงเล็กน้อยจะทำให้มีอัตราการผิดของกุญแจรหัสลับมีเพียงเล็กน้อยอาจจะทำให้ *Alice* และ *Bob* ไม่ทราบการเข้ามาขโมยสถานะทางควอนตัมของ *Eve* ได้ ก็ไม่ได้ทำการยกเลิกกุญแจรหัสลับชุดนั้นและเนื่องมาจากสาเหตุอื่นๆ เช่น สัญญาณรบกวนในช่องสื่อสารเชิงควอนตัม จึงต้องมีกระบวนการกลั่นกุญแจรหัสลับ (Secret Key Distillation) ที่กระทำผ่านทางช่องสื่อสารสาธารณะ ซึ่งกระบวนการกลั่นกุญแจรหัสลับนี้ได้แบ่งการทำงานเป็นสองกระบวนการย่อย ได้แก่ กระบวนการไกล่เกลี่ยความผิดพลาด (Reconciliation) เป็นกระบวนการแก้ไขกุญแจรหัสลับที่แตกต่างกันระหว่าง *Alice* และ *Bob* ให้กลับมีความเหมือนกัน โดย *Alice* จะทำการส่งข้อมูลบางอย่างเกี่ยวกับกุญแจรหัสลับของตนไปให้ยัง *Bob* ผ่านทางช่องสื่อสารสาธารณะ และ *Bob* จะใช้ข้อมูลนี้ร่วมกับกุญแจรหัสลับของตนเพื่อแก้ไขกุญแจรหัสลับที่แตกต่างให้เหมือนกุญแจรหัสลับของ *Alice* และ กระบวนการขยายสภาวะส่วนตัว (Privacy Amplification) เป็นกระบวนการลดความสำคัญของข้อมูลเกี่ยวกับกุญแจรหัสลับบางส่วนลง เนื่องจาก *Eve* อาจจะเข้ามาขโมยสถานะทางควอนตัมของแสงทางช่องสื่อสารเชิงควอนตัมและได้ข้อมูลเกี่ยวกับกุญแจรหัสลับเพียงเล็กน้อยทำให้ *Alice* และ *Bob* ไม่ทราบการเข้ามาขโมยสถานะทางควอนตัมของ *Eve* และข้อมูลระหว่างกระบวนการแก้ไขความผิดพลาดจากทางช่องสื่อสารสาธารณะหาก *Eve* ได้รับข้อมูลเกี่ยวกับกุญแจรหัสลับมากเกินไปก็อาจจะสามารถสร้างกุญแจรหัสลับใหม่ขึ้นมาได้ จึงทำให้ *Alice* และ *Bob* จำเป็นต้องลดความสามารถในการสร้างกุญแจรหัสลับใหม่ของ *Eve*

2.4 กระบวนการส่งกุญแจรหัสลับ

กระบวนการส่งกุญแจรหัสลับเป็นกระบวนการในการส่งกุญแจรหัสลับโดยกุญแจรหัสลับจะแทนด้วยสถานะควอนตัมของแสง เช่น สถานะโพลาไรเซชันของโฟตอนเดี่ยว โดยสถานะทางควอนตัมเหล่านี้จะถูกส่งผ่านช่องสื่อสารเชิงควอนตัมและใช้กฎความไม่แน่นอนของไฮเซนแบร์ก (Uncertainty Principle) ช่วยยืนยันความปลอดภัยของสถานะควอนตัมเหล่านี้ว่าจะไม่ถูกคัดลอกหรือสร้างขึ้นมาใหม่ หรือ หากมี *Eve* เข้ามาขโมยสถานะทางควอนตัมของแสงเหล่านี้ *Alice* และ *Bob* จะทราบทันทีว่ามีผู้บุกรุกเข้ามาในระบบจากอัตราการผิดพลาดที่สูงผิดปกติของกุญแจรหัสลับที่ทำการรับ-ส่ง ในขณะนั้นหากพบก็จะทำการทิ้งกุญแจรหัสลับที่ทำการส่งนี้ทั้งหมด และทำการส่งกุญแจรหัสลับใหม่ หรือ หาก *Eve* เข้ามาขโมยสถานะควอนตัมไปเพียงเล็กน้อย ซึ่งจะทำให้กุญแจรหัสลับมีความผิดพลาดเล็กน้อย ส่งผลให้ *Alice* และ *Bob* ไม่สามารถตรวจพบได้ว่ามี *Eve* เข้ามารบกวนระบบ เนื่องจากความผิดพลาดอาจเกิดจากสาเหตุอื่นๆ เช่น สัญญาณรบกวนทาง

ช่องสัญญาณทางควอนตัม เป็นต้น ส่งผลให้ *Eve* มีข้อมูลเกี่ยวกุญแจรหัสลับเพียงเล็กน้อย จึงได้มีกระบวนการขยายสภาวะส่วนตัวที่เป็นการลดความสำคัญของข้อมูลเหล่านี้จน *Eve* ไม่สามารถนำข้อมูลเกี่ยวกับกุญแจรหัสลับไปสร้างกุญแจรหัสลับใหม่ได้ และมั่นใจได้ว่ากุญแจรหัสลับนี้มีความปลอดภัยโดยตัวอย่างโพรโทคอลที่ใช้ในการกระจายกุญแจรหัสลับเชิงควอนตัมมีดังต่อไปนี้

2.5 โพรโทคอล BB84

โพรโทคอล BB84 เป็นโพรโทคอลตัวแรกที่ใช้ในระบบรหัสลับเชิงควอนตัมที่ได้มีการเสนอขึ้นในปี ค.ศ. 1984 โดย C.H. Bennett นักวิจัยจาก IBM และ Gilles Brassard นักวิจัยมหาวิทยาลัย Montreal ในระหว่างการประชุมวิชาการของสมาคม IEEE ที่ประเทศอินเดีย[8] ซึ่งเป็นแนวคิดการแทนกุญแจรหัสลับด้วยสถานะทางควอนตัม โดยอาศัยสถานะโพลาไรเซชันของโฟตอนเดี่ยว (Single Photon) สี่สถานะซึ่ง ได้แก่ 0 องศา (→) 90 องศา (↑) 45 องศา (↗) และ -45 องศา (↘) ในการแทนค่าบิตในระบบสื่อสารดิจิทัล ซึ่งสถานะโพลาไรเซชันในแนว 0 องศา (→) และ 90 องศา (↑) แทนด้วยเวกเตอร์ฐาน + และสถานะโพลาไรเซชันในแนว 45 องศา (↗) และ -45 องศา (↘) แทนด้วยเวกเตอร์ฐานได้ × โดยโพรโทคอลนี้กำหนดให้สถานะโพลาไรเซชันในแนว 90 องศา (↑) และ 45 องศา (↗) แทนบิต "1" และสถานะโพลาไรเซชัน 0 องศา (→) และ -45 องศา (↘) แทนบิต "0" ตามเลขบิตในระบบสื่อสารแบบดิจิทัลซึ่งการทำงานของโพรโทคอล BB84 จะทำการส่งบิตสุ่มโดยแทนด้วยสถานะโพลาไรเซชันดังกระบวนการต่อไปนี้

ทางช่องสื่อสารเชิงควอนตัมนั้น *Alice* จะทำการสุ่มบิตขึ้นมาสองบิตเพื่อเลือกสถานะโพลาไรเซชันหนึ่งในสี่สถานะ เพื่อส่งไปยัง *Bob* โดยบิตแรกแทนด้วยเวกเตอร์ฐานหนึ่งในสองเวกเตอร์ฐาน ซึ่งได้แก่ + และ × ซึ่งบิตที่สองจะแทนกุญแจรหัสลับบิตซึ่งประกอบด้วยบิต "0" หรือ บิต "1" และ *Bob* ก็จะทำการสุ่มเวกเตอร์ฐานหนึ่งในสองเวกเตอร์ฐาน เพื่อใช้รับสถานะโพลาไรเซชันที่ส่งมาโดย *Alice* แล้วเปลี่ยนเป็นบิตข้อมูล ซึ่งบิตที่รับได้ทั้งหมดจะถูกเรียกว่า "คีย์ดิบ" (Raw Key) หลังจากนั้น *Bob* จะบอก *Alice* ถึงเวกเตอร์ฐานที่ใช้ในการรับผ่านทางช่องสื่อสารสาธารณะและเมื่อ *Alice* ได้รับข้อความจาก *Bob* ถึงเวกเตอร์ฐานที่ใช้ในการรับเรียบร้อยแล้วนั้น *Alice* จะบอกเวกเตอร์ฐานที่ *Bob* ใช้รับที่ถูกต้องให้แก่ *Bob* และภายหลังจากกระบวนการนี้ *Alice* และ *Bob* จะมีกุญแจรหัสลับบิตที่ใกล้เคียงกัน ซึ่งจะมีเพียงบางบิตที่ต่างกันอยู่โดยบิตเหล่านี้จะถูกเรียกว่า "ชิพคีย์" (Sifted Key) ส่วนบิตที่เวกเตอร์ฐานในการส่งและรับต่างกันจะถูกลบทิ้งทั้งหมด ดังตัวอย่างที่แสดงในตารางที่ 2.1 แสดงวิธีสร้างคีย์ดิบในกระบวนการส่งกุญแจรหัสลับของโพรโทคอล BB84

ตารางที่ 2.1 ตัวอย่างวิธีการส่งบิตสุ่มโดยโพรโทคอลBB84

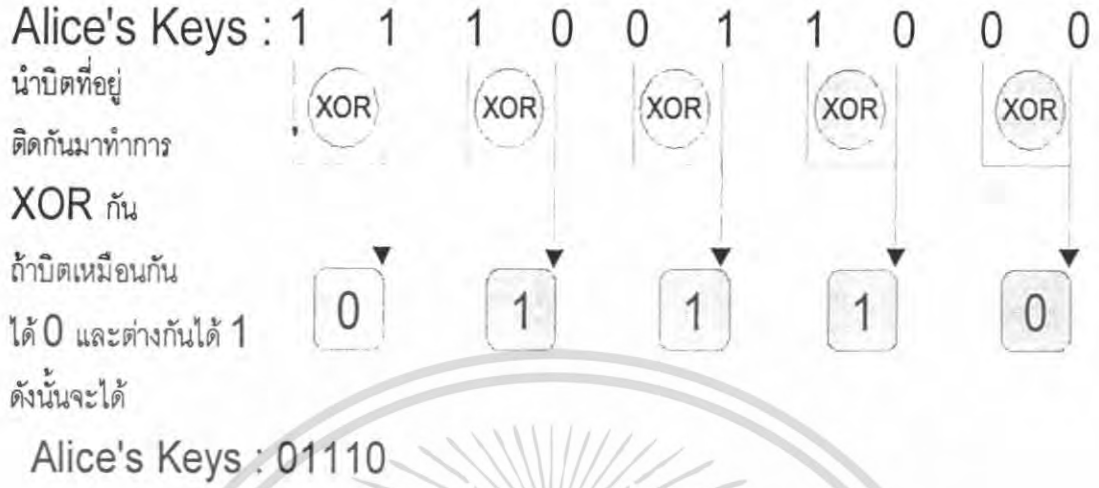
เวกเตอร์ฐาน / Alice	✚	✖	✚	✚	✖
บิต	1	1	1	0	0
สถานะโพลาไรเซชัน	↑	↗	↑	→	↘
เวกเตอร์ฐาน / Bob	✚	✚	✖	✚	✖
คีย์ดิบ (Raw Key)	1	0	1	0	0
ชิพคีย์ (Shifted Key)	1	-	-	0	0

2.6 กระบวนการกลั่นกุญแจรหัสลับ

กระบวนการกลั่นกุญแจรหัสลับ (Secret Key Distillation) เป็นกระบวนการที่มีจัดการบิตของ *Alice* และ *Bob* เพื่อนำบิตเหล่านี้มาสร้างเป็นกุญแจรหัสลับ (Secret Key) เพื่อนำไปใช้ในการเข้ารหัสลับและถอดรหัสลับข้อมูลระหว่างกัน ผ่านทางช่องทางสื่อสารสาธารณะซึ่งแบ่งเป็นกระบวนการย่อยที่สำคัญอยู่สองกระบวนการคือ กระบวนการแก้ไขความผิดพลาด (Error Correction) และกระบวนการขยายสภาวะส่วนตัว (Privacy Amplification) [9]

2.6.1 การแก้ไขความผิดพลาด

การแก้ไขความผิดพลาด(Error Correction)เป็นการแก้ไขความผิดพลาดที่เกิดขึ้นจากการส่งกุญแจรหัสลับผ่านทางช่องสื่อสารเชิงควอนตัม ซึ่งความผิดพลาดอาจเกิดขึ้นจากการที่ *Eve* เข้ามาขโมยกุญแจรหัสลับเพียงเล็กน้อยหรือเกิดขึ้นจากสัญญาณรบกวนภายในช่องสื่อสารเชิงควอนตัม ทำให้กุญแจรหัสลับระหว่าง *Alice* และ *Bob* แตกต่างกัน โดย *Alice* ทำการเปิดเผยข้อมูลเกี่ยวกับกุญแจรหัสลับของตนไปยัง *Bob* ผ่านทางช่องสื่อสารสาธารณะแล้ว *Bob* ใช้ข้อมูลส่วนนี้และกุญแจรหัสลับบิตของตนแก้ไขความผิดพลาดที่เกิดขึ้น เช่น *Alice* แบ่งกุญแจรหัสลับของตนออกเป็นบล็อกแล้วส่งพาริตี้บิต (Parity Bit) ของแต่ละบล็อกไปให้ *Bob* เพื่อทำการเปรียบเทียบกันซึ่งความแตกต่างของพาริตี้บิตจะบ่งบอกถึงความแต่ต่างระหว่างกัน เป็นต้น



รูปที่ 2.3 กระบวนการขยายสภาวะส่วนตัว

2.6.2 การขยายสภาวะส่วนตัว

การขยายสภาวะส่วนตัว(Privacy Amplification) เป็นกระบวนทำงานภายหลังจากการแก้ไขความผิดพลาดเพื่อลดความสามารถในการสร้างหรือสำเนากุญแจรหัสลับขึ้นมาใหม่จากข้อมูลที่ *Eve* อาจจะได้มาจากการเข้าไปขโมยโพลาไรเซชันของโพตอนเดี่ยวภายในช่องสื่อสารเชิงควอนตัม และข้อมูลที่ *Alice* และ *Bob* เปิดเผยระหว่างกระบวนการแก้ไขความผิดพลาด โดยวิธีของการลดความสำคัญของข้อมูลเกี่ยวกับกุญแจรหัสลับที่ *Eve* มีอยู่ ทำได้ดังเช่น การนำกุญแจรหัสลับบิตที่ติดกันสองตำแหน่งมารวมกันแบบมอดุโล 2 (Modulo-2) หรือ XOR เพื่อลดกุญแจรหัสลับลงครึ่งหนึ่งเนื่องจากข้อมูลที่ *Eve* มีอยู่นั้นจะเป็นข้อมูลที่ขาดหายไปบางส่วนทำให้ *Eve* ไม่สามารถสร้างกุญแจรหัสลับขึ้นมาใหม่เช่น ถ้าหากนำบิตตำแหน่งที่หนึ่งและตำแหน่งที่สองมาทำ XOR กันจะได้ผลลัพธ์เป็นกุญแจรหัสลับบิตใหม่ขนาดหนึ่งบิตดังรูปที่ 2.3 ซึ่ง *Eve* จะได้ข้อมูลเพียงหนึ่งตำแหน่งเท่านั้น จึงทำให้กุญแจรหัสลับบิตที่ *Eve* จะสามารถสร้างขึ้นมาใหม่หลังจากกระบวนการนี้จะมีความน่าจะเป็นระหว่างบิต "1" และบิต "0" อย่างละ 50% ดังนั้นโอกาสที่กุญแจรหัสลับบิตนั้นจะเป็นบิตที่ผิดเป็นไปได้ค่อนข้างสูง ทำให้กุญแจรหัสลับที่ใช้สื่อสารกันระหว่าง *Alice* และ *Bob* มีความปลอดภัยมากยิ่งขึ้น

ตารางที่ 2.2 พาริตี้บิตของข้อมูลขนาด 5 บิต

ข้อมูล (k) ขนาด 5บิต	ข้อมูลที่รวมพาริตี้คู่ (Even Parity)	ข้อมูลที่รวมพาริตี้คี่ (Odd Parity)
00000	000000	100000
01010	001010	101010
10101	110101	010101
11111	111111	011111

2.7 หลักการพื้นฐานของพาริตี้บิต

โดยทั่วไปการสื่อสารหรือการส่งสัญญาณข่าวสารทุกครั้งจะต้องเกิดสัญญาณรบกวนขึ้นมาด้วยทุกครั้งทำให้ข้อมูลข่าวสารเกิดความผิดเพี้ยนไป ซึ่งหลักการพาริตี้บิตเป็นกระบวนการหนึ่งเพื่อตรวจสอบหาความผิดพลาดที่เกิดขึ้นซึ่งมีหลักการคือผู้ส่งจะแบ่งบิตข้อมูลออกเป็นบล็อกๆละ k บิต จากนั้นทำการเพิ่มบิตตรวจสอบเข้าไปอีกหนึ่งบิตทำให้บล็อกมีขนาดเพิ่มเป็น k+1บิตซึ่งการเดิมบิตตรวจสอบนั้นจะเป็นบิต "1" หรือ บิต "0" ขึ้นอยู่กับจำนวนบิต "1"ที่อยู่ในบล็อก วางไว้หน้าหรือหลังก็ได้ เมื่อผู้รับได้รับข้อมูลชุดนั้นจะทำการหาค่าพาริตี้บิตเพื่อเปรียบเทียบกับพาริตี้บิตที่ได้จากผู้ส่ง ถ้าหากพาริตี้บิตมีความแตกต่างกันผู้รับจะทราบได้ทันทีว่าเกิดความผิดพลาดขึ้นภายในบล็อกข้อมูลนั้นและจำนวนบิตที่เกิดความผิดพลาดนั้นเป็นจำนวนกี่ แต่ผู้รับจะไม่ทราบตำแหน่งบิตที่ผิดและไม่สามารถแก้ไขบิตที่ผิดให้กลับมาถูกต้องได้ แต่ถ้าหากพาริตี้บิตที่ผู้รับและผู้ส่งมีความเหมือนกันอาจเป็นไปได้ว่าข้อมูลที่ผู้รับได้รับอาจจะมีความถูกต้องหรือมีจำนวนบิตที่เกิดความผิดพลาดนั้นเป็นจำนวนคู่ ทำให้ผู้รับไม่ทราบเลยว่ามีความผิดพลาดเกิดขึ้นกับข้อมูลชุดนั้นหรือไม่ ซึ่งตัวอย่างของการหาค่าพาริตี้บิตสามารถแสดงได้ดังตารางที่ 2.2 โดยที่การตรวจสอบความผิดพลาดด้วยพาริตี้บิตนั้นสามารถแบ่งออกเป็น 2 ประเภทคือ

2.7.1 **พาริตี้คู่** (Even Parity) เป็นการเพิ่มบิตตรวจสอบเข้าไปอีกหนึ่งบิตเพื่อทำให้ผลรวมของจำนวนบิต "1"ในบล็อกนั้นเป็นจำนวนคู่

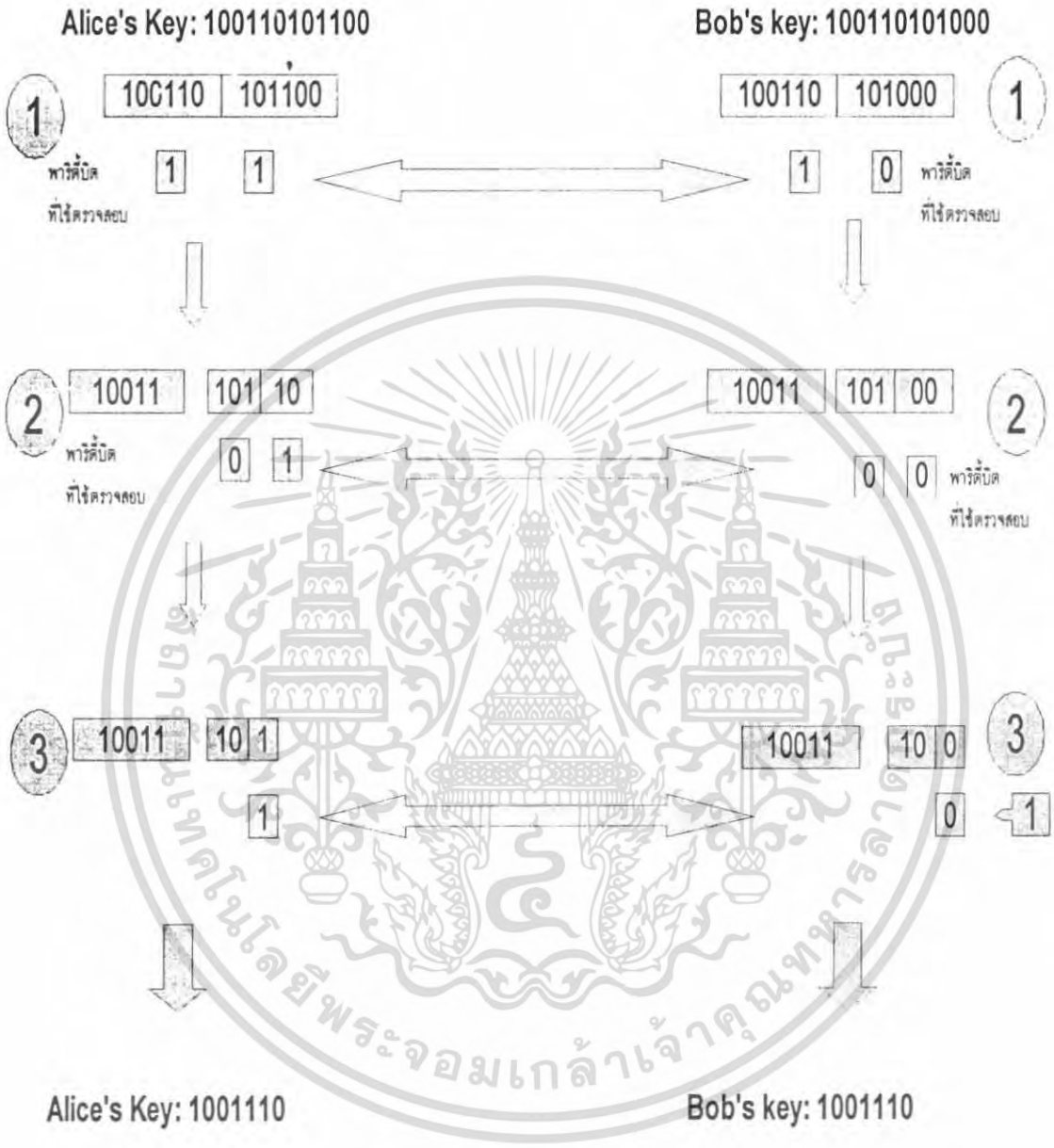
2.7.2 **พาริตี้คี่** (Odd Parity) เป็นการเพิ่มบิตตรวจสอบเข้าไปอีกหนึ่งบิตเพื่อทำให้ผลรวมของจำนวนบิต "1"ในบล็อกนั้นเป็นจำนวนคี่

2.8 ตัวอย่างโพรโทคอลแก้ไขความผิดพลาด

การส่งกุญแจรหัสลับผ่านช่องทางการสื่อสารเชิงควอนตัมโดยการแทนกุญแจรหัสลับด้วยสถานะควอนตัมของแสงเช่น โพลาไรเซชัน เป็นต้น การที่ *Eve* เข้ามาขโมยสถานะควอนตัมของแสง ความไม่เป็นอุดมคติของอุปกรณ์และสัญญาณรบกวนภายในช่องสื่อสารเชิงควอนตัม เป็นสาเหตุที่ทำให้สถานะควอนตัมของแสงเกิดการเปลี่ยนแปลงส่งผลให้กุญแจรหัสลับที่ส่งเกิดความผิดพลาดตามไปด้วย เพื่อที่จะทำให้กุญแจรหัสลับของทั้ง *Alice* และ *Bob* เหมือนกันมากที่สุดจึงต้องมีการแก้ไขความผิดพลาดที่เกิดขึ้น ดังตัวอย่างต่อไปนี้

2.8.1 โพรโทคอล BBBSS

ระบบวิทยาการรหัสลับที่เสนอโดย C.H. Bennett และคณะ ในปี ค.ศ. 1992 นั้นได้มีการนำเสนอโพรโทคอล BBBSS ซึ่งเป็นโพรโทคอลแรกที่ใช้ในกระบวนการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม [1] โดยที่การทำงานของโพรโทคอลนี้อาศัยพาริตี้บิตเพื่อใช้ในการตรวจหาตำแหน่งที่กุญแจรหัสลับบิตมีความผิดพลาดและแก้ไขความผิดพลาดที่เกิดขึ้นกับกุญแจรหัสลับบิตนั้น ซึ่งเริ่มจากการที่ผู้ส่งและผู้รับทำการแบ่งกุญแจรหัสลับของตนออกเป็นบล็อกและหาค่าพาริตี้บิตแล้วนำมาเปรียบเทียบกัน แล้วจะทำการตัดกุญแจรหัสลับบิตตำแหน่งสุดท้ายของบล็อกออกเพื่อลดข้อมูลเกี่ยวกับกุญแจรหัสลับที่บุคคลที่สามจะได้ไประหว่างการแก้ไขความผิดพลาด จากนั้นก็จะเก็บกุญแจรหัสลับที่เหลือในบล็อกนั้นไว้ ส่วนบล็อกที่พาริตี้บิตไม่ตรงกันก็จะทำการแบ่งกุญแจรหัสลับที่เหลือในบล็อกนั้นออกเป็นสองบล็อกแล้วเปรียบเทียบพาริตี้บิตของบล็อกเหล่านั้นใหม่ ซึ่งผู้ส่งและผู้รับจะทำเช่นนี้จนกว่าจะพบกุญแจรหัสลับที่ผิดแล้วจึงทำการแก้ไขความผิดพลาด โดยการทำการแก้ไขความผิดพลาดของโพรโทคอล BBBSS นี้แสดงดังรูปที่ 2.4 ตัวอย่างการแก้ไขความผิดพลาดของโพรโทคอล BBBSS เช่น *Alice* มีกุญแจรหัสลับคือ 100110101100 และ *Bob* กุญแจรหัสลับคือ 100110101100 โดยที่ขั้นที่ 1ทำการแบ่งกุญแจออกเป็นบล็อก ขนาดบล็อกละ 6 บิต แล้วทำการหาพาริตี้บิต จากนั้นนำพาริตี้บิตมาเปรียบเทียบกันและหลังจากนั้นจะทำการตัดกุญแจรหัสลับบิตสุดท้ายของแต่ละบล็อกทิ้งไปแล้วเก็บค่ากุญแจรหัสลับในบล็อกนั้นไว้ ขั้นที่ 2 หากบล็อกใดที่มีพาริตี้ไม่ตรงกันจะทำการแบ่งบล็อกนั้นออกเป็นสองบล็อกจากนั้นทำการนำพาริตี้บิตมาเทียบกันและทำการตัดกุญแจรหัสลับบิตสุดท้ายของบล็อกที่นำพาริตี้บิตมาเปรียบเทียบกันทิ้งไปก่อนจะเก็บกุญแจรหัสลับในบล็อกนั้นไว้ โดยที่หากพาริตี้ยังไม่ตรงกันก็จะทำซ้ำแบบเดิมอีกครั้ง และขั้นที่ 3 หากพบบิตที่ผิดก็จะทำการแก้ไขความผิดพลาดและตัดกุญแจรหัสลับบล็อกนั้นทิ้งไปแล้วทั้งสองฝ่ายก็จะได้กุญแจรหัสลับที่พร้อมนำไปใช้งาน



รูปที่ 2.4 การแก้ไขความผิดพลาดของโพรโทคอล BBBSS

ขั้นตอนที่ 1 *Alice* และ *Bob* แบ่งกุญแจรหัสลับบิตของตนออกเป็นบล็อกโดยขนาดของบล็อกจะขึ้นอยู่กับปริมาณความผิดพลาดที่เกิดขึ้นระหว่างการส่งทางช่องทางการสื่อสารเชิงควอนตัม (Quantum Bit Error Rate: QBER) และความน่าจะเป็นของการเกิดความผิดพลาดภายในบล็อกข้อมูลซึ่งความน่าจะเป็นในการเกิดการผิดของบิด k บิต ภายในบล็อกขนาด N บิตแสดงดังสมการ [10]

$$P_k(N) = \binom{N}{k} e^k (1-e)^{N-1} \tag{2.1}$$

โดยที่ $P_k(N)$ คือความน่าจะเป็นที่เกิดความผิดพลาดจำนวน k บิตในบล็อกขนาด N บิต

e คืออัตราความผิดพลาดของกุญแจรหัสลับ

โดยขนาดของบล็อกที่เลือกใช้ในกระบวนการแก้ไขความผิดพลาดขึ้นอยู่กับประสิทธิภาพของการแก้ไขความผิดพลาด ซึ่งต้องมีความสามารถในการแก้ไขความพลาดสูงสุดดังสมการ

$$\eta(N) = \frac{(1-P_1(N))(N-1)}{N} \tag{2.2}$$

โดยที่ η คือประสิทธิภาพของการแก้ไขความผิดพลาด

N คือขนาดของบล็อก

$P_1(N)$ คือความน่าจะเป็นของการเกิดความผิดพลาดขึ้นหนึ่งบิตภายในบล็อกขนาด N บิต

ขั้นตอนที่ 2 *Alice* และ *Bob* หาพาริตี้บิตในแต่ละบล็อกและทำการเปรียบเทียบพาริตี้บิตเหล่านี้ เมื่อพบพาริตี้บิตที่แตกต่างกัน *Alice* และ *Bob* จะทราบทันทีว่ากุญแจรหัสลับบิตมีความผิดพลาดหากผลการเปรียบเทียบพาริตี้บิตมีความเหมือนกัน *Alice* และ *Bob* จะเก็บกุญแจรหัสลับบิตเหล่านั้นไว้พร้อมกับทิ้งกุญแจรหัสลับบิตในตำแหน่งสุดท้ายของทุกบล็อกทิ้งไปเพื่อลดข้อมูลเกี่ยวกับกุญแจรหัสลับที่ *Eve* อาจจะได้รับระหว่างกระบวนการเปรียบเทียบพาริตี้บิต

ขั้นตอนที่ 3 หาก *Alice* และ *Bob* พบความผิดพลาดที่เกิดขึ้นภายในบล็อกกุญแจรหัสลับแล้ว ทั้ง *Alice* และ *Bob* จะทำการแบ่งบล็อกกุญแจรหัสลับนั้นออกเป็นสองส่วนเท่าๆ กันและทำการหาพาริตี้บิตของบล็อกย่อยพร้อมกับเปรียบเทียบพาริตี้บิตอีกครั้งเพื่อหากุญแจรหัสลับที่ผิดอยู่ในบล็อกย่อยใด หลังจากนั้น *Alice* และ *Bob* จะตัดกุญแจรหัสลับบิตตำแหน่งสุดท้ายของแต่ละบล็อกทิ้งไป

ขั้นตอนที่ 4 *Alice* และ *Bob* จะทำตามขั้นตอนที่ 3 จนกว่าจะพบตำแหน่งกุญแจรหัสลับบิตที่ผิดและแก้ไขบิตที่ผิดนั้นให้กลับมาถูกต้องพร้อมกับตัดกุญแจรหัสลับบิตในตำแหน่งสุดท้ายของบล็อกออก

ขั้นตอนที่ 5 *Alice* และ *Bob* จะนำกุญแจรหัสลับบิตที่ผ่านการแก้ไขความผิดพลาดและกุญแจรหัสลับบิตของบล็อกที่ไม่เกิดความผิดพลาดรวมเข้าด้วยกันอีกครั้งและทำการเพิ่มขนาดของบล็อกพร้อมกับสับเปลี่ยนตำแหน่งของกุญแจรหัสลับบิตที่ตนมีอยู่และทำซ้ำตามขั้นตอนที่ 1 จนเหลืออัตราความผิดพลาดในปริมาณที่ยอมรับได้

2.8.2 โพรโทคอล CASCADE

โพรโทคอล CASCADE เป็นโพรโทคอลที่ปรับปรุงจากโพรโทคอล BBBSS เนื่องจากในระหว่างกระบวนการแก้ไขความผิดพลาดของโพรโทคอล BBBSS นี้จะทำการตัดกุญแจรหัสลับบิตในตำแหน่งสุดท้ายของแต่ละบล็อกออก การตัดกุญแจรหัสลับบิตเหล่านี้ออกทำให้โพรโทคอล BBBSS นี้มีความปลอดภัย สามารถลดข้อมูลเกี่ยวกับกุญแจรหัสลับบิตที่ *Eve* จะได้จากการเข้ามาขโมยระหว่างการแก้ไขความผิดพลาดแต่จะส่งผลต่อประสิทธิภาพในการแก้ไขความผิดพลาดของโพรโทคอลและจำนวนกุญแจรหัสลับบิตที่เหลือภายหลังจากการแก้ไขความผิดพลาด [11] โพรโทคอล CASCADE เป็นโพรโทคอลการแก้ไขความผิดพลาดเช่นเดียวกับโพรโทคอล BBBSS ที่อาศัยพาริตี้บิตในการตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นแต่โพรโทคอลนี้จะไม่ตัดกุญแจรหัสลับบิตในตำแหน่งสุดท้ายของแต่ละบล็อกออกเพื่อเพิ่มประสิทธิภาพการแก้ไขความผิดพลาด โดยรายละเอียดของแต่ละขั้นตอนมีดังต่อไปนี้

ขั้นตอนที่ 1 การคำนวณหาขนาดของบล็อกและจำนวนรอบในการวนซ้ำ (Pass) เพื่อหาขนาดของบล็อกที่เหมาะสมในการเปรียบเทียบพาริตี้บิตและจำนวนรอบในการทำงานทั้งหมดโดยใช้ค่าของอัตราความผิดพลาดของกุญแจรหัสลับบิตที่ส่งผ่านทางช่องทางการสื่อสารเชิงควอนตัม (e) สามารถหาได้โดยทาง *Bob* ทำการสุ่มกุญแจรหัสลับขึ้นมาจำนวนหนึ่งแล้วส่งไปให้ทาง *Alice* โดยระบุตำแหน่งของกุญแจรหัสลับนั้นไปด้วย จากนั้นทาง *Alice* ก็จะทำการตรวจสอบกุญแจรหัสลับที่ส่งมาว่ามีความผิดพลาดเกิดขึ้นเท่าไรเมื่อนำจำนวนบิตที่เกิดความผิดพลาดนำมาคำนวณค่าเพื่อใช้แทนอัตราความผิดพลาดเชิงควอนตัมของกุญแจรหัสลับทั้งชุดได้ ซึ่งมีความสัมพันธ์ดังสมการ

$$e \approx \frac{m}{n} \tag{2.3}$$

โดยที่ e คืออัตราความผิดพลาดของกุญแจรหัสลับ

n คือจำนวนบิตทั้งหมดที่ถูกสุ่มขึ้นมา และ m คือจำนวนบิตที่พบความผิดพลาด

ตารางที่ 2.3 ความน่าจะเป็นของจำนวนบิตผิดในบล็อกและการกระจายแบบปัวส์ซอง

จำนวนบิตผิดภายในบล็อก	อัตราความผิดพลาดของกุญแจรหัสลับเท่ากับ 0.02 ขนาดของบล็อกเท่ากับ36	การกระจายโอกาสแบบปัวส์ซอง ค่าเฉลี่ยเท่ากับ ln(2)
0	0.4832	0.5001
1	0.3550	0.3466
2	0.1268	0.1201
3	0.0293	0.0277
4	0.0049	0.0048
5	0.0006	0.0007
6 และ จำนวนบิตผิดอื่นๆ	0.0001	0.0001

ข้อดีโพรโทคอลนี้คือขนาดของบล็อกจะมีขนาดต่างๆ ขึ้นอยู่กับอัตราความผิดพลาดที่เกิดจากการส่งกุญแจรหัสลับโดยขนาดของบล็อกเริ่มต้น (Initial Block) หรือ N_0 นั้นจะคำนวณมาจากการประมาณการผิดของบิตภายในบล็อกซึ่งความน่าจะเป็นที่จะเกิดการผิดของบิตจำนวน k บิตภายในบล็อกขนาด N บิต แสดงดังสมการ [10]

$$P_k(V) = \binom{N}{k} e^k (1-e)^{N-k} \qquad (2.4)$$

พิจารณาขนาดของบล็อกและอัตราการผิดของชิพคีย์ที่เสนอใน [11] ผลคูณของขนาดของบล็อกและอัตราความผิดพลาดของกุญแจรหัสลับ(e)มีค่าอยู่ระหว่าง [0.7,0.75] เช่น อัตราการผิดของกุญแจรหัสลับมีค่าเท่ากับ 0.02 จะใช้ขนาดของบล็อกเท่ากับ 36 บิต อัตราการผิดของกุญแจรหัสลับมีค่าเท่ากับ 0.04 ขนาดของบล็อกเท่ากับ 16 บิด เป็นต้น ซึ่งเมื่อทำการหาความน่าจะเป็นในการเกิดการผิดของบิตภายในบล็อกข้อมูลขนาด N บิต ตามสมการที่ (2.4) แสดงได้ดังตารางที่ 2.3 ซึ่งความน่าจะเป็นที่เกิดขึ้นนี้มีรูปแบบการกระจายโอกาสแบบปัวส์ซอง (Poisson Distribution) ที่ค่าเฉลี่ยเท่ากับ ln(2) [12] จากสมการที่ (2.4) และตารางที่ 2.3 ความน่าจะเป็นของบล็อกข้อมูลขนาด N บิดจะมีจำนวนบิตผิดภายในบล็อกจำนวนคู่ดังสมการ

$$P = \sum \frac{N!}{j!(N-j)!} e^{j}(1-e)^{j} \quad \text{โดยที่ } j = 0,2,4,... \qquad (2.5)$$

โดยที่ e คืออัตราการผิดของกุญแจรหัสลับบิต

P คือความน่าจะเป็นของการเกิดบิตผิดเป็นจำนวนคู่ภายในบล็อกขนาด N บิต ถ้าผลคูณของขนาดของบล็อกและอัตราการผิดของดิวบิตมีค่าประมาณ ln(2) หรือ $N \times e \approx \ln(2)$ ดังนั้นความน่าจะเป็นของการเกิดความผิดพลาดเป็นจำนวนคู่ภายในบล็อกขนาด N บิตจะมีค่าประมาณ 0.626 [12] ส่วนขนาดของบล็อกเริ่มต้น N_0 จะหาได้ดังสมการ

$$N_0 = \frac{\ln(2)}{e} \qquad (2.6)$$

โดยที่ N_0 คือขนาดของบล็อกเริ่มต้น

ขนาดของบล็อกถัดไปของโพรโทคอล CASCADE จะเพิ่มขนาดบล็อกเป็นสองเท่าของขนาดของบล็อกก่อนหน้าดังสมการ [11]

$$N_{i+1} = 2N_i \quad \text{โดยที่ } i = 0,1,2,.... \qquad (2.7)$$

โดยที่ N_{i+1} เป็นขนาดของบล็อกในรอบที่ $i+1$ และขนาดของบล็อกมากที่สุดแสดงดังสมการ

$$N_m \geq \frac{l}{4} \qquad (2.8)$$

โดยที่ l คือขนาดความยาวทั้งหมดของซิฟคีย์

N_m คือขนาดบล็อกสุดท้ายของรอบปกติและมีขนาดของบล็อกเกินกว่า $1/4$ ของความยาวทั้งหมดของซิฟคีย์

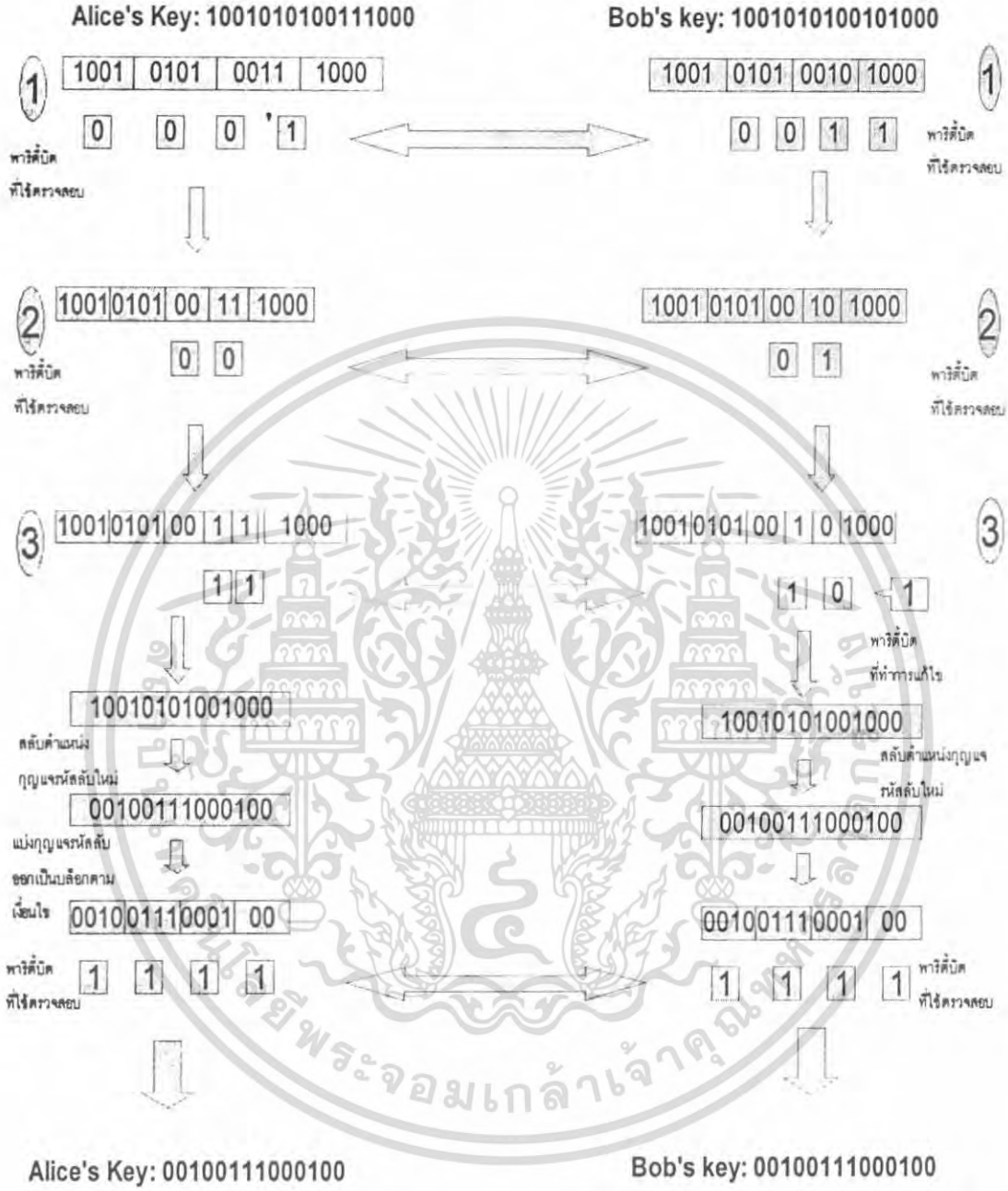
หลังจากได้ขนาดของบล็อกทั้งหมด จำนวนรอบทั้งหมดในการวนซ้ำจะมีค่าเท่ากับจำนวนของขนาดบล็อกทั้งหมดที่หาได้รวมกับจำนวนรอบที่เพิ่มขึ้นมาอีกสองรอบซึ่งขนาดของบล็อกของรอบในการวนซ้ำที่เพิ่มขึ้นมานี้จะมีขนาดเท่ากับขนาดของบล็อกสูงสุด โดยการเพิ่มจำนวนรอบของการวนซ้ำเพื่อเพิ่มโอกาสที่ *Alice* และ *Bob* จะแก้ไขความผิดพลาดที่ยังเหลืออยู่ให้ได้มากที่สุดซึ่งจะช่วยเพิ่มประสิทธิภาพการทำงานของโพรโทคอล

ขั้นตอนที่ 2 *Alice* และ *Bob* แบ่งกุญแจรหัสลับของตนออกเป็นบล็อกตามขนาดที่หาได้จากขั้นตอนแรกจากนั้น *Alice* และ *Bob* จะหาพาริตี้บิตในแต่ละบล็อก

ขั้นตอนที่ 3 *Alice* ส่งพาริตี้บิดของตนไปให้ *Bob* และเมื่อ *Bob* ทำการเปรียบเทียบพาริตี้บิตเหล่านี้แล้วหากพบพาริตี้บิตมีความแตกต่างกันทั้ง *Alice* และ *Bob* จะทราบทันที่ว่ามีกุญแจรหัสลับบิตแตกต่างกันอยู่ภายในบล็อกนั้นและจะทำการแบ่งบล็อกกุญแจรหัสลับบล็อกนั้นออกเป็นสองบล็อกๆ ละเท่าๆ กันแล้ว *Alice* และ *Bob* จะทำการหาค่าพาริตี้บิตและเปรียบเทียบพาริตี้บิตเหล่านี้อีกครั้ง ซึ่งกระบวนการนี้จะดำเนินซ้ำจนกว่า *Alice* และ *Bob* จะทราบตำแหน่งของบิตที่ผิดและทำการแก้ไขบิตนั้นแสดงดังรูปที่ 2.5 หากผลการเปรียบเทียบพาริตี้บิตมีความเหมือนกัน *Alice* และ *Bob* จะดำเนินการในรอบการวนซ้ำถัดไปจะกว่าจะดำเนินการเสร็จทั้งหมด

ขั้นตอนที่ 4 หาก *Alice* และ *Bob* เพิ่มขนาดของบล็อกตามขนาดของบล็อกที่ได้หาไว้ก่อนหน้าตามจำนวนรอบการวนซ้ำโดยอาจจะทำการสลับตำแหน่งของกุญแจรหัสลับบิตด้วยเพื่อเพิ่มประสิทธิภาพการหาความผิดพลาดและจะดำเนินการตามขั้นตอนที่ 2 ถึงขั้นตอนที่ 4 ใหม่จนกว่าจะสิ้นสุดจำนวนรอบในการวนซ้ำ (Pass) หรือจนกว่าจะเหลืออัตราการผิดของกุญแจรหัสลับเป็นที่ยอบรับได้โดยการทำงานของโพรโทคอล CASCADE [6][7]

ซึ่งกระบวนการทั้งหมดตามขั้นตอนการทำงานของโพรโทคอล CASCADE แสดงได้ดังรูปที่ 2.5 ตัวอย่างการแก้ไขความผิดพลาดของโพรโทคอล CASCADE เช่น *Alice* มีกุญแจรหัสลับคือ 1001010100111000 และ *Bob* กุญแจรหัสลับคือ 1001010100101000 โดยที่ขั้นที่ 1ทำการแบ่งกุญแจออกเป็นบล็อก ขนาดบล็อกละ 4 บิต แล้วทำการหาพาริตี้บิตจากนั้นนำพาริตี้บิตมาเปรียบเทียบกันแล้วเก็บค่ากุญแจรหัสลับในบล็อกที่พาริตี้ตรงกันไว้ ขั้นที่ 2หากบล็อกใดที่มีพาริตี้ไม่ตรงกันจะทำการแบ่งบล็อกนั้นออกเป็นสองบล็อกจากนั้นทำการนำพาริตี้บิตมาเทียบกันและทำการเก็บกุญแจรหัสลับในบล็อกที่พาริตี้บิตตรงกันไว้ โดยที่หากพาริตี้ยังไม่ตรงกันก็จะทำซ้ำแบบเดิมอีกครั้ง ขั้นที่ 3 พบบิตที่ผิดก็จะทำการแก้ไขความผิดพลาดและตัดกุญแจรหัสลับบล็อกนั้นทิ้งไปแล้วทั้งสองฝ่ายก็จะเพิ่มขนาดบล็อกขึ้นเป็นสองเท่าแล้วเทียบว่าขนาดใหม่ที่ได้นั้นมีขนาดมากกว่า1/4หรือไม่ถ้ายังไม่มากกว่าก็จะแบ่งกุญแจรหัสลับออกตามขนาดบล็อกที่ได้ แต่ถ้ามากกว่าจะทำการสลับตำแหน่งของกุญแจรหัสลับแล้วแบ่งเป็นบล็อกขนานเท่าบล็อกสุดท้ายที่ใช้แล้วทำซ้ำ 1และ2



รูปที่ 2.5 การแก้ไขความผิดพลาดของโพรโทคอล CASCADE

หลังจากที่ทำการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE เสร็จเรียบร้อยแล้ว กระบวนการตรวจสอบความผิดพลาดที่อาจจะยังเหลืออยู่จะถูกดำเนินต่อเพื่อยืนยันว่ากุญแจรหัสลับบิตที่ได้มีความเหมือนกันหรือมีความผิดพลาดเหลืออยู่น้อยที่สุด ซึ่งการหาโอกาสที่จะมีกุญแจรหัสลับบิตที่ผิดเหลืออยู่จะถูกจัดการโดย *Alice* และ *Bob* จะสุ่มกุญแจรหัสลับบิตที่ผ่านการแก้ไข

ความผิดพลาดแล้วมารวมกันและแบ่งเป็นบล็อกพร้อมกับหาค่าพาริตี้บิตและทำการเปรียบเทียบพาริตี้บิตเหล่านี้ หากพาริตี้บิตที่เปรียบเทียบทั้งหมด (w บิต) มีความเหมือนกัน *Alice* และ *Bob* จะมีค่าความน่าจะเป็นที่จะมีกุญแจรหัสลับที่ต่างกันเท่ากับ 2^{-w} [13] และบิตที่สุ่มมาทั้งหมดเหล่านี้จะถูกทิ้งไปเนื่องจากเปิดเผยให้แก่บุคคลที่สามได้รับทราบ นอกจากจะหาจำนวนกุญแจรหัสที่มีโอกาสเกิดการผิดพลาดอยู่แล้วจำนวนบิตที่เปิดเผยทั้งหมดจากการแก้ไขความผิดพลาดก็เป็นอีกหนึ่งสิ่งที่ถูกนำมาพิจารณาด้วยเช่นกัน ซึ่งหากจำนวนบิตที่เปิดเผยมีจำนวนมากโอกาสที่ Eve จะนำข้อมูลนี้ไปใช้สร้างหรือทำสำเนากุญแจรหัสลับบิตใหม่ขึ้นมาก็ยิ่งมีโอกาสสูงขึ้นเช่นกัน โดยจำนวนบิตที่เปิดเผยแสดงดังสมการ [6][7]

$$d = l(1+\xi)H(e) \tag{2.9}$$

โดยที่ d คือจำนวนบิตที่เปิดเผยทั้งหมด l คือจำนวนซิฟคีย์ (Sifted Key) และ ξ คือ Overhead Factor หรือจำนวนบิตที่เปิดเผยนอกเหนือจากพาริตี้บิตที่เปิดเผยจากกระบวนการแก้ไขความผิดพลาด

บทที่ 3
การออกแบบและจัดทำโครงงาน

จากการศึกษาในส่วนของการแก้ไขความผิดพลาดของกุญแจรหัสลับเชิงควอนตัมที่กระทำหลังจากที่ได้มีการส่งกุญแจรหัสลับผ่านทางช่องสัญญาณควอนตัมเรียบร้อยแล้ว เพื่อเป็นการแก้ไขกุญแจรหัสลับระหว่างผู้ส่งและผู้รับมีความใกล้เคียงกันมากที่สุดหรือให้ได้กุญแจรหัสลับที่ถูกต้อง จึงได้มีการจัดทำโปรแกรมเพื่อมาใช้ในส่วนของการแก้ไขความผิดพลาดขึ้น เพื่อใช้สำหรับแก้ไขความผิดพลาดของกุญแจรหัสลับทางควอนตัมที่สามารถเกิดขึ้นได้ระหว่างขั้นตอนการส่งกุญแจรหัสลับทางช่องสัญญาณทางควอนตัม ด้วยโพรโทคอล CASCADE เพื่อเป็นตัวอย่างแสดงการทำงานของกระบวนการต่างๆในระบบวิทยาการรหัสลับเชิงควอนตัม โดยมีอยู่สองกระบวนการคือ กระบวนการกระจายกุญแจรหัสลับเชิงควอนตัม และกระบวนการกลั่นกุญแจรหัสลับ เพื่อแสดงให้ผู้ที่สนใจสามารถเข้าใจกระบวนการทำงานอย่างชัดเจน ดังรายละเอียดการทำงานดังนี้

3.1 การออกแบบจำลองการทำงาน

จากการศึกษาวิทยาการรหัสลับเชิงควอนตัมนั้นทำให้สามารถนำมาออกแบบขั้นตอนการทำงานของโปรแกรมจำลองการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมที่เลือกใช้โพรโทคอล CASCADE นั้นได้ออกแบบการทำงานเป็นขั้นตอนย่อยสองส่วนคือการส่งกุญแจรหัสลับบิตด้วยโพรโทคอล BB84 การแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE และการขยายสภาวะส่วนตัว ซึ่งมีรายละเอียดของการทำงานของระบบดังต่อไปนี้

1. จากวิธีการส่งกุญแจรหัสลับด้วยโพรโทคอล BB84 ในหัวข้อที่ 2.5 สามารถนำมาจำลองการส่งกุญแจรหัสลับบิตด้วยโพรโทคอล ดังนี้

ขั้นตอนที่ 1 ผู้ส่งกำหนดขนาดของของกุญแจรหัสลับที่ส่งไปให้ผู้รับทราบ

ขั้นตอนที่ 2 ผู้ส่งทำงานสุ่มทำการสุ่มเวกเตอร์ฐานขึ้นมาและทำการเปลี่ยนเวกเตอร์ฐานไปเป็นกุญแจรหัสลับ ซึ่งจะแสดงออกมาในรูปสองบิตข้อมูลที่อยู่ติดกันโดยสองบิตนั้นจะแทนสถานะโพลาไรเซชันหนึ่งสถานะที่จะใช้ในการส่งให้ได้ตามขนาดของกุญแจรหัสลับที่ตกลงกันไว้

ขั้นตอนที่ 3 ผู้รับทำการสุ่มเวกเตอร์ฐานขึ้นมาให้มีขนาดเท่ากับขนาดของกุญแจรหัสลับที่ตกลงกันไว้เพื่อมาใช้รับสถานะโพลาไรเซชันที่ผู้ส่งจะส่งมาให้

ขั้นตอนที่ 4 ผู้ส่งทำการส่งสถานะโพลาไรเซชันไปยังผู้รับ

ขั้นตอนที่ 5 ผู้รับทำการแปลงสถานะโพลาไรเซชันที่รับได้ให้เป็นกุญแจรหัสลับ

ขั้นตอนที่ 6 ผู้ส่งและผู้รับทำการตรวจสอบเวกเตอร์ฐานว่าตรงกันหรือไม่ พร้อมทำการลบตำแหน่งที่ไม่ตรงกันทิ้งไป

2. จากวิธีการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE ในหัวข้อที่ 2.8.2 และการขยายสภาวะส่วนตัวในหัวข้อที่ 2.6.2 สามารถนำมาจำลองการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE และการขยายสภาวะส่วนตัวได้ดังต่อไปนี้

ขั้นตอนที่ 1 ผู้รับทำการสุ่มกุญแจรหัสลับขึ้นมาบางตำแหน่งแล้วส่งไปทางผู้ส่งพร้อมลบกุญแจรหัสลับในตำแหน่งนั้นทิ้งไป

ขั้นตอนที่ 2 ผู้ส่งทำการตรวจสอบคำนวนเปอร์เซ็นต์ความผิดพลาด และทำการลบกุญแจรหัสลับตำแหน่งที่ใช้เปรียบเทียบกับกุญแจรหัสลับชุดที่ถูกส่งมาตรวจสอบทิ้งไป

ขั้นตอนที่ 3 ทำการคำนวนหาขนาดของบล็อกเริ่มต้นและจำนวนรอบในการวนซ้ำแล้วส่งไปบอกผู้รับ

ขั้นตอนที่ 4 ทั้งสองฝ่ายแบ่งกุญแจรหัสลับออกเป็นบล็อกตามขนาดที่คำนวนได้จากนั้นทำการคำนวนหาพาริตี้บิตของแต่ละบล็อก

ขั้นตอนที่ 5 ผู้รับส่งพาริตี้บิตที่คำนวนได้ไปให้ผู้ส่งเพื่อทำการตรวจสอบว่าตรงกันหรือไม่ ถ้าตรงกันจะเก็บบล็อกนั้นไว้แต่ถ้าไม่ตรงกันผู้ส่งจะแจ้งกลับไปยังผู้รับ

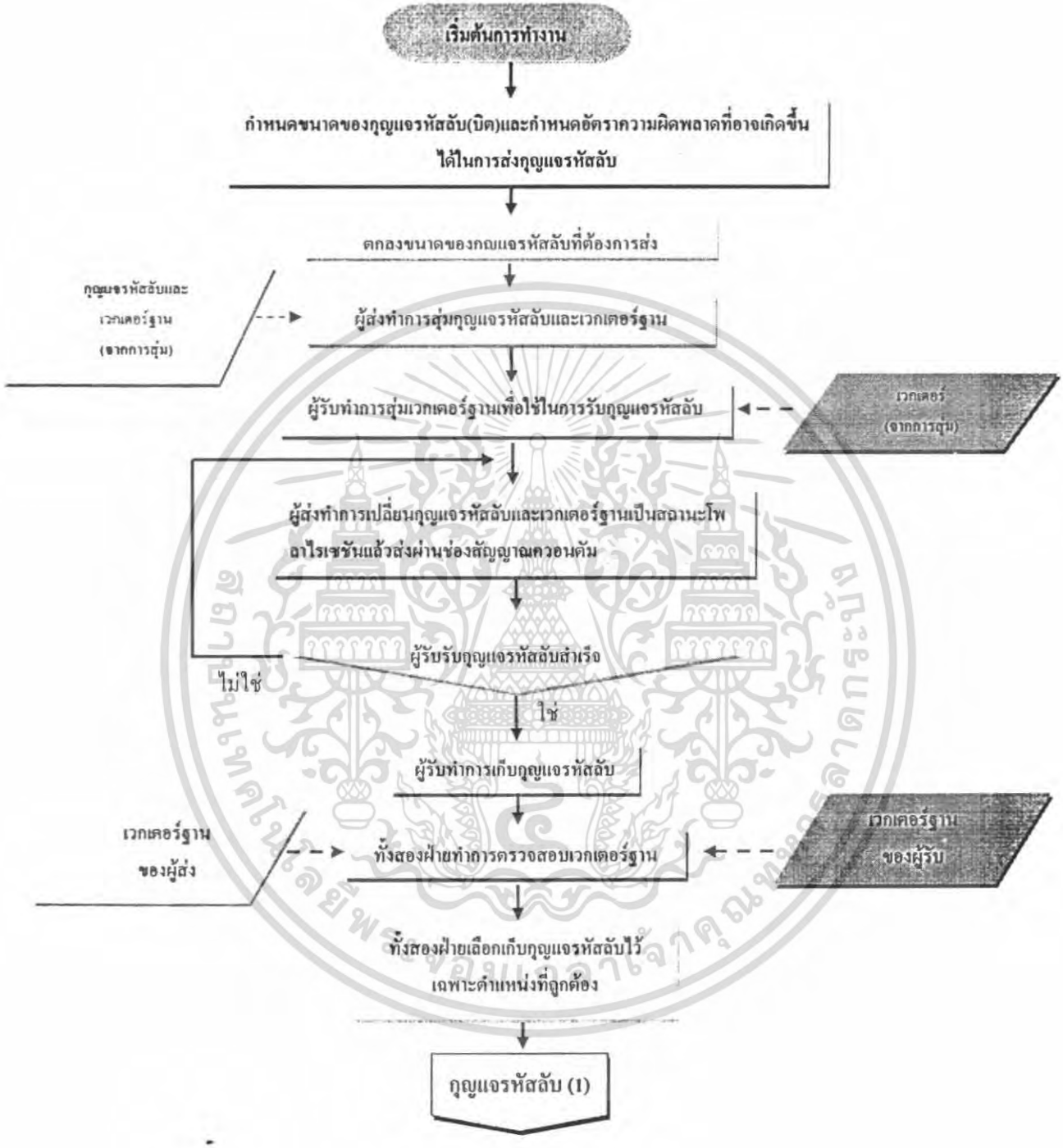
ขั้นตอนที่ 6 ทั้งสองฝ่ายทำการแบ่งบล็อกที่มีค่าพาริตี้บิตไม่ตรงกันออกเป็นสองส่วนแล้วคำนวนหาพาริตี้บิตอีกครั้ง

ขั้นตอนที่ 7 ทำซ้ำขั้นตอนที่ 6 จนได้บิตที่ผิดนั้นคือการแบ่งครั้งสุดท้ายจะต้องมีขนาดของบล็อกเท่ากับสองบิตเมื่อได้บล็อกที่ผิดก็ส่งจะทำการลบบล็อกนั้นทิ้งพร้อมแจ้งไปยังผู้รับให้ทำเช่นเดียวกัน

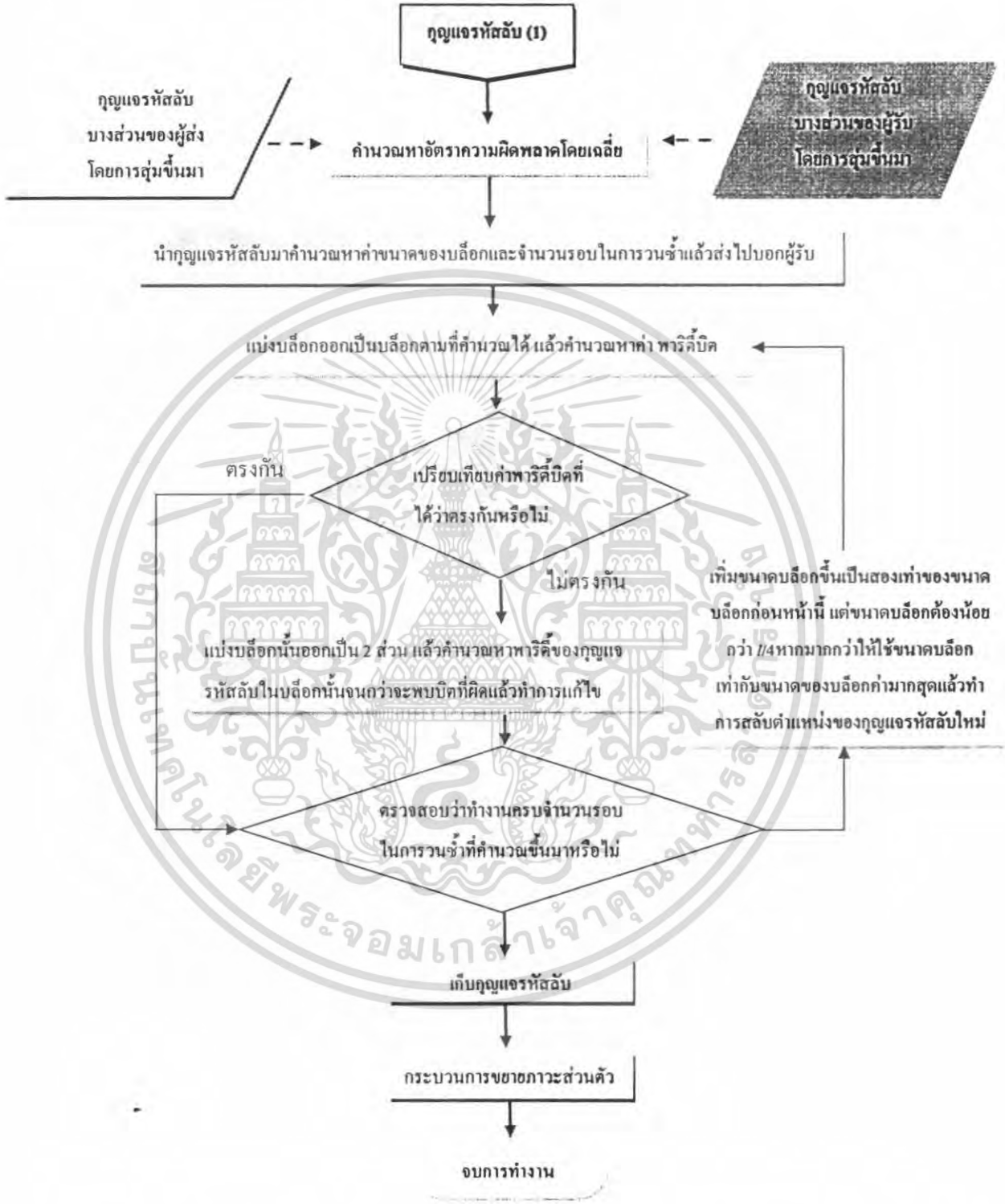
ขั้นตอนที่ 8 จากนั้นทั้งสองฝ่ายจะทำการแบ่งกุญแจรหัสลับออกเป็นบล็อกอีกครั้งโดยที่ขนาดของบล็อกจะเพิ่มเป็นสองเท่าของขนาดบล็อกก่อนหน้านี้แล้วทำซ้ำในขั้นตอนที่ 4 5 6 และ7 จนกว่าจะครบรอบในการวนซ้ำ ตามสมการที่ 2.8

ขั้นตอนที่ 9 จากนั้นทั้งผู้รับและผู้ส่งจะทำการขยายสภาวะส่วนตัว โดยที่จะนำกุญแจรหัสลับที่มีมาทำการXORกันเพื่อลดความสำคัญของกุญแจรหัสลับบางตำแหน่งลงไป

ซึ่งในขั้นตอนการทำงานย่อยข้างบนนี้เมื่อพิจารณาแล้วสามารถแบ่งออกเป็นสองส่วนคือ ส่วนที่หนึ่งคือการแสดงกระจายกุญแจรหัสลับเชิงควอนตัม โดยที่ในส่วนที่สองนั้นเป็นการจำลองเพื่อแสดงกระบวนการแก้ไขความผิดพลาดและกระบวนการขยายสภาวะส่วนตัวด้วย



รูปที่ 3.1 แผนภาพขั้นตอนการจำลองการส่งกุญแจรหัสลับบิตด้วยโพรโทคอล BB84



รูปที่ 3.2 แผนภาพขั้นตอนการทำงานของโปรแกรมการจำลองการทำงานของการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADEและการขยายสภาวะส่วนตัว

3.2 แผนภาพแสดงการทำงานของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม

จากการออกแบบในข้อ3.1นั้นสามารถนำมาเขียนเป็นแผนภาพการทำงานเป็นส่วนๆได้ดังนี้

- แผนภาพจำลองในส่วนของการส่งกุญแจรหัสลับบิตด้วยโพรโทคอล BB84 ดังรูปที่ 3.1
- แผนภาพจำลองในส่วนของการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE และการขยายสภาวะส่วนตัวดังรูปที่ 3.2

3.3 ขั้นตอนการแก้ไขความผิดพลาด

การจัดทำโปรแกรมในปริญญานิพนธ์เล่มนี้เลือกใช้ภาษา Visual Basic 2005 ขึ้นมาเพื่อเป็นภาษาสำหรับการเขียนโปรแกรมนี้ เนื่องจากภาษา Visual Basic 2005 สามารถพัฒนาโปรแกรมในส่วนที่ติดต่อกับผู้ใช้ (User's Interface) ได้ง่าย การใช้งานผู้ใช้จะควบคุมการทำงานของโปรแกรมได้จากตัวควบคุม ที่โปรแกรมยุคใหม่นิยมใช้ มีฟังก์ชันการใช้งานที่ง่ายมีรองรับการทำงานหลายรูปแบบที่นิยมใช้กัน จะทำให้การเขียนและออกแบบโปรแกรมเป็นไปได้อย่างรวดเร็ว ซึ่งรายละเอียดของโปรแกรมมีดังนี้

3.3.1 กำหนดความยาวของกุญแจรหัสลับ

ส่วนนี้มีหน้าที่ในการรับความยาวของกุญแจรหัสลับที่ผู้ใช้ต้องการส่ง แล้วโปรแกรมจะทำการตรวจสอบโดยอัตโนมัติว่าความยาวของกุญแจรหัสลับที่ผู้ใช้ใส่เข้ามาถูกต้อง หรือมีขนาดเกินกว่าที่โปรแกรมจะสามารถทำงานได้หรือไม่ โดยในโปรแกรมสามารถปรับเปลี่ยนขนาดของกุญแจรหัสลับได้ตามความต้องการ ซึ่งขนาดของกุญแจรหัสลับที่จะกำหนดจะต้องมีค่าอยู่ระหว่าง 100 ถึง 10,000 บิต ถ้าผู้ใช้ใส่ขนาดของกุญแจรหัสลับได้ถูกต้อง โปรแกรมก็จะดำเนินการต่อไป แต่หากขนาดของกุญแจรหัสลับผิดพลาด โปรแกรมก็จะนำขนาดของกุญแจรหัสลับที่ถูกกำหนดไว้มาใช้ โดยที่เมื่อโปรแกรมทำงานขึ้นมานั้นจะมีการแสดงขนาดกุญแจรหัสลับที่ถูกกำหนดไว้ขึ้นมาก่อน ในส่วนนี้ผู้ใช้ยังสามารถสั่งให้โปรแกรมทำการสุ่มขนาดกุญแจรหัสลับขึ้นมาได้อีกด้วย

3.3.2 กำหนดอัตราความผิดพลาดที่อาจเกิดขึ้นในระหว่างการส่งกุญแจรหัสลับ

ส่วนนี้มีหน้าที่เพื่อกำหนดอัตราความผิดพลาดที่อาจเกิดขึ้นได้ระหว่างการส่งกุญแจรหัสลับผ่านช่องสัญญาณควอนตัม ซึ่งส่วนนี้สามารถตรวจสอบความถูกต้องและผู้ใช้สามารถให้โปรแกรมสุ่มอัตราความผิดพลาดเองได้เช่นเดียวกับส่วนของการรับขนาดกุญแจรหัสลับที่ต้องการส่ง

3.3.3 *Alice* ทำการสุ่มเวกเตอร์ฐานและกุญแจรหัสลับ

ส่วนนี้โปรแกรมจะเรียกใช้งานฟังก์ชันสุ่มเพื่อใช้ในการสุ่มเวกเตอร์ฐานและกุญแจรหัสลับ จากนั้นจึงนำสิ่งที่สุ่มได้ไปเก็บในตัวแปรของ *Alice* โดยที่ "+0" แทนสถานะโพลาไรเซชันในแนว "0 องศา" "+1" แทนสถานะโพลาไรเซชันในแนว "90 องศา" "x0" แทนสถานะโพลาไรเซชันในแนว "-45 องศา"และ "x1" แทนสถานะโพลาไรเซชันในแนว "45 องศา"

3.3.4 *Bob* ทำการสุ่มเวกเตอร์ฐานเพื่อใช้ในการรับกุญแจรหัสลับ

ส่วน*Bob*จะทำการสุ่มเวกเตอร์ฐาน แล้วนำเวกเตอร์ฐานไปเก็บไว้ในตัวแปรของทาง *Bob* โดยที่ "+" แทนเวกเตอร์ฐาน "[+]" และ "x" แทนเวกเตอร์ฐาน "[x]"

3.3.5 *Alice* ทำการส่งกุญแจรหัสลับ

ส่วนนี้โปรแกรมจะทำการจำลองเสมือนกับว่ามีการส่งกุญแจรหัสลับผ่านทางช่องสัญญาณควอนตัมซึ่งจะทำให้กุญแจรหัสลับบางส่วนของทาง *Bob* ไม่ตรงกับกุญแจรหัสลับทาง *Alice*

แต่ *Bob* จะต้องทำการรับกุญแจรหัสลับให้ครบถ้วนตามที่ *Alice* ต้องการส่งโดยที่บิตไหนที่เวกเตอร์ฐานที่สุ่มมาใช้รับไม่สามารถใช้รับได้ก็จะแทนบิตนั้นด้วย "1" แล้วก็จะเริ่มการทำงานในขั้นตอนถัดไป

หลังจากนั้น *Bob* จะทำการเก็บกุญแจรหัสลับที่รับได้แล้วเปลี่ยนให้กุญแจรัสลับบางตำแหน่งเกิดความผิดพลาด (เพื่อจำลองว่าในการรับส่งกุญแจรหัสลับนั้นมีผิดพลาดเกิดขึ้น) และรอนำไปผ่านขั้นตอนถัดไป

3.3.6 การตรวจสอบเวกเตอร์ฐาน

ส่วนนี้โปรแกรมจะทำการนำเวกเตอร์ฐานที่ได้จากการสุ่มในช่วงต้นของทั้ง *Alice* และ *Bob* มาทำการเปรียบเทียบ โดยจะเลือกเก็บเฉพาะกุญรหัสแจลับตำแหน่งที่มีเวกเตอร์ฐานตรงกันเท่านั้น

3.3.7 คำนวณหาอัตราความผิดพลาดของกุญแจรหัสลับบิต

ส่วนนี้จะเรียกใช้งานฟังก์ชันที่สามารถทำการสุ่มกุญแจรหัสลับทาง *Bob* บางตำแหน่งขึ้นมาโดยที่ตำแหน่งที่ไม่ได้สุ่มขึ้นมาจะแสดงออกมาเป็นช่องว่างและส่งไปให้ *Alice*

หลังจากนั้น *Alice* จะทำการเปรียบเทียบกุญแจรหัสลับแล้วทำการคำนวณหาอัตราความผิดพลาดของกุญแจรหัสลับบิต (กุญแจรหัสลับที่ใช้ในครั้งนี้ทั้งทาง *Alice* และ *Bob* จะทำการลบกุญแจรหัสลับชุดนี้ทิ้งไป)

3.3.8 แก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE

ส่วนนี้มีการทำงานที่ค่อนข้างซับซ้อน มีการเรียกฟังก์ชันที่สามารถคำนวณหาพาริตี้ของทั้งทางผู้ส่ง *Alice* และทางผู้รับ *Bob* แล้ว *Bob* จะส่งค่าที่ได้ไปยัง *Alice* หลังจากนั้น *Alice* ก็จะนำค่าที่ได้ไปคำนวนในฟังก์ชันที่สามารถแก้ไขความผิดพลาดของกุญแจรหัสลับ ซึ่งเมื่อผ่านกระบวนการนี้กุญแจรหัสลับถูกแก้ไขความผิดพลาดเกือบทั้งหมดมีรายละเอียดดังนี้

ขั้นตอนที่ 1. *Alice* ทำการคำนวนหาขนาดของบล็อกเริ่มต้นและจำนวนรอบในการวนซ้ำทั้งหมดแล้วส่งไปบอก *Bob*

ขั้นตอนที่ 2. ทั้งสองฝ่ายแบ่งกุญแจรหัสลับออกเป็นบล็อกตามขนาดที่คำนวนได้จากนั้นทำการคำนวนหาพาริตี้บิตของแต่ละบล็อก

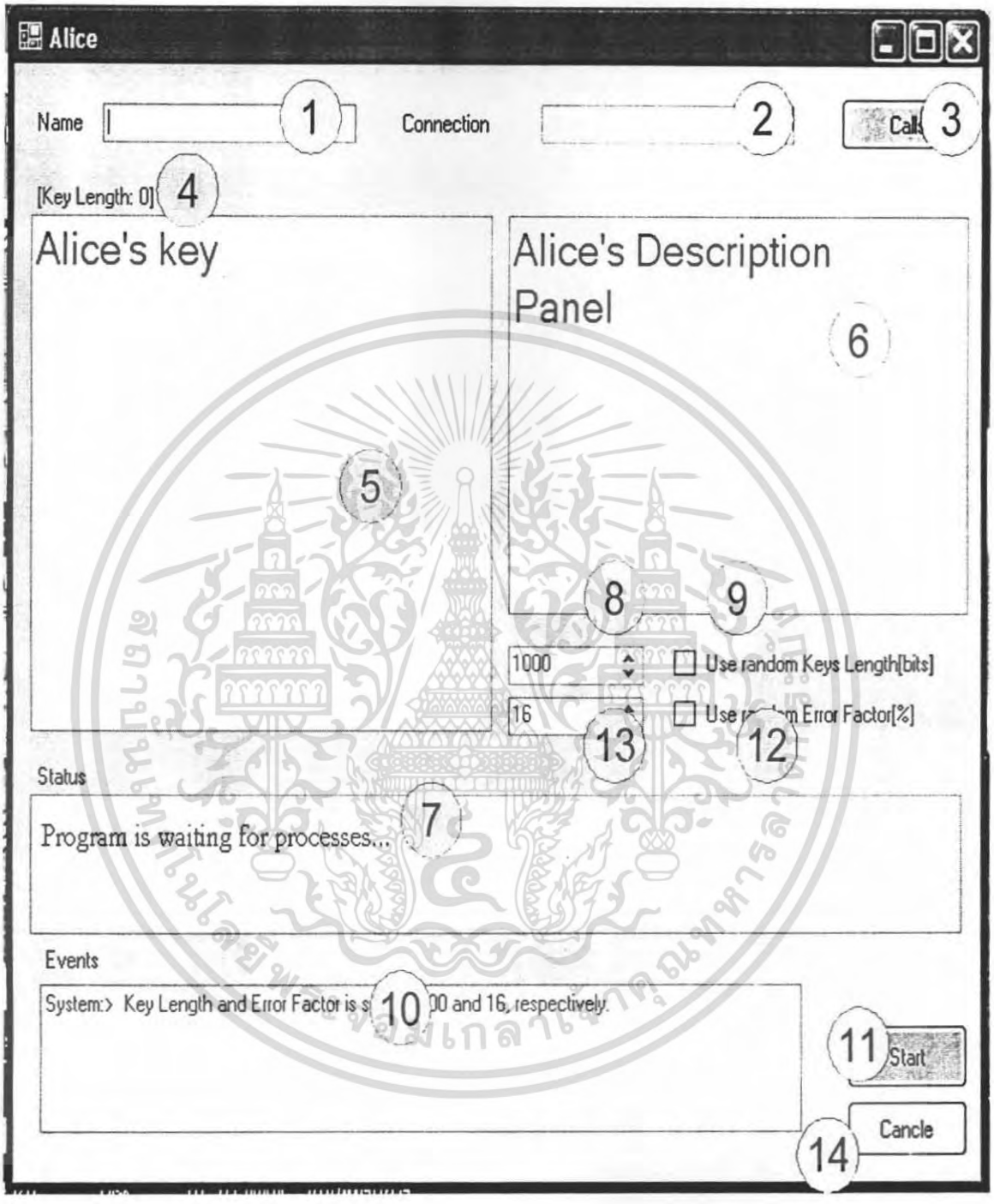
ขั้นตอนที่ 3. *Bob* ส่งค่าพาริตี้บิตที่คำนวนได้ไปให้ *Alice* เพื่อทำการตรวจสอบว่าตรงกันหรือไม่ถ้าตรงกันจะเก็บบล็อกนั้นไว้แต่ถ้าไม่ตรงกัน *Alice* จะแจ้งกลับไปยัง *Bob*

ขั้นตอนที่ 4. ทั้งสองฝ่ายทำการแบ่งบล็อกที่มีพาริตี้บิตไม่ตรงกันออกเป็นสองส่วนแล้วคำนวนหาพาริตี้บิตอีกครั้ง ทำซ้ำขั้นตอนที่ 2 และขั้นตอนที่ 3 จนได้บิตที่ผิดนั้นคือการแบ่งครั้งสุดท้ายจะต้องมีขนาดของบล็อกเท่ากับสองบิตเมื่อได้บล็อกที่ผิดและจะทำการแก้ไขบิดนั้นให้ถูกต้องแต่ในรอบสุดท้ายจะทำการลบบิตนั้น

ขั้นตอนที่ 5. จากนั้นทั้งสองฝ่ายจะทำการแบ่งกุญแจรหัสลับออกเป็นบล็อกอีกครั้งโดยที่ขนาดของบล็อกจะเพิ่มเป็นสองเท่าของขนาดบล็อกก่อนหน้านี้แล้วทำซ้ำในขั้นตอนที่ 1 ถึง ขั้นตอนที่ 4 จนกว่าจะถึงจำนวนรอบในการวนซ้ำรอบสุดท้าย

3.3.9 การขยายสภาวะส่วนตัว

ส่วนนี้ทั้ง *Alice* และ *Bob* จะทำหลังจากทำการแก้ไขความผิดพลาดเรียบร้อยแล้วเพื่อลดความสำคัญของกุญแจรหัสลับบางบิตที่มีการเปิดเผยในขณะทำการแก้ไขความผิดพลาด โดยที่จะนำกุญแจรหัสลับบิดทั้งหมดมาแบ่งเป็นบล็อกๆละสองบิดแล้วนำสองบิตในบล็อกนั้นมารวมกันแบบมอดุโล 2 (Modulo-2) หรือ XORกัน



รูปที่ 3.3 แอพพลิเคชันฟอร์มของ *Alice*

3.4 รายละเอียดของแอพพลิเคชันที่ออกแบบมาในการนำเสนอโครงงาน

แอพพลิเคชันของโปรแกรมที่ถูกออกแบบมีรายละเอียดของแต่ละแอพลิเคชั้น ดังนี้

3.4.1 แอพพลิเคชันฟอร์มของ *Alice* มีรายละเอียดดังรูปที่ 3.3 และแต่ละหมายเลขมีรายละเอียดดังนี้

หมายเลข 1 แสดงชื่อหรือหมายเลข IP ที่ต้องการใช้ในการติดต่อสื่อสาร (ในที่นี้คือ *Alice*)

หมายเลข 2 แสดงชื่อหรือหมายเลข IP ผู้ที่ต้องการจะติดต่อสื่อสารด้วย (ในที่นี้คือ *Bob*)

หมายเลข 3 กดเพื่อต้องการให้แสดงฟอร์มของคู่สนทนาขึ้นมา

หมายเลข 4 แสดงขนาดของกุญแจรหัสลับของ *Alice*

หมายเลข 5 แสดงเวกเตอร์ฐานและกุญแจรหัสลับที่สุ่มขึ้นมาของ *Alice* แล้วหลังจากนั้นจะแสดงเพียงกุญแจรหัสลับเท่านั้น

หมายเลข 6 แสดงค่าเวกเตอร์ฐานแล้วหลังจากนั้นจะแสดงค่าพารีตี้บิตของ *Alice* ที่ใช้

หมายเลข 7 แสดงค่าสถานะระหว่างการทำงานในขณะนั้นของโปรแกรม

หมายเลข 8 กำหนดและแสดงขนาดของกุญแจรหัสลับที่ตกลงกันไว้

หมายเลข 9 กำหนดขนาดของกุญแจรหัสลับโดยต้องสุ่มขึ้นมาเท่านั้น

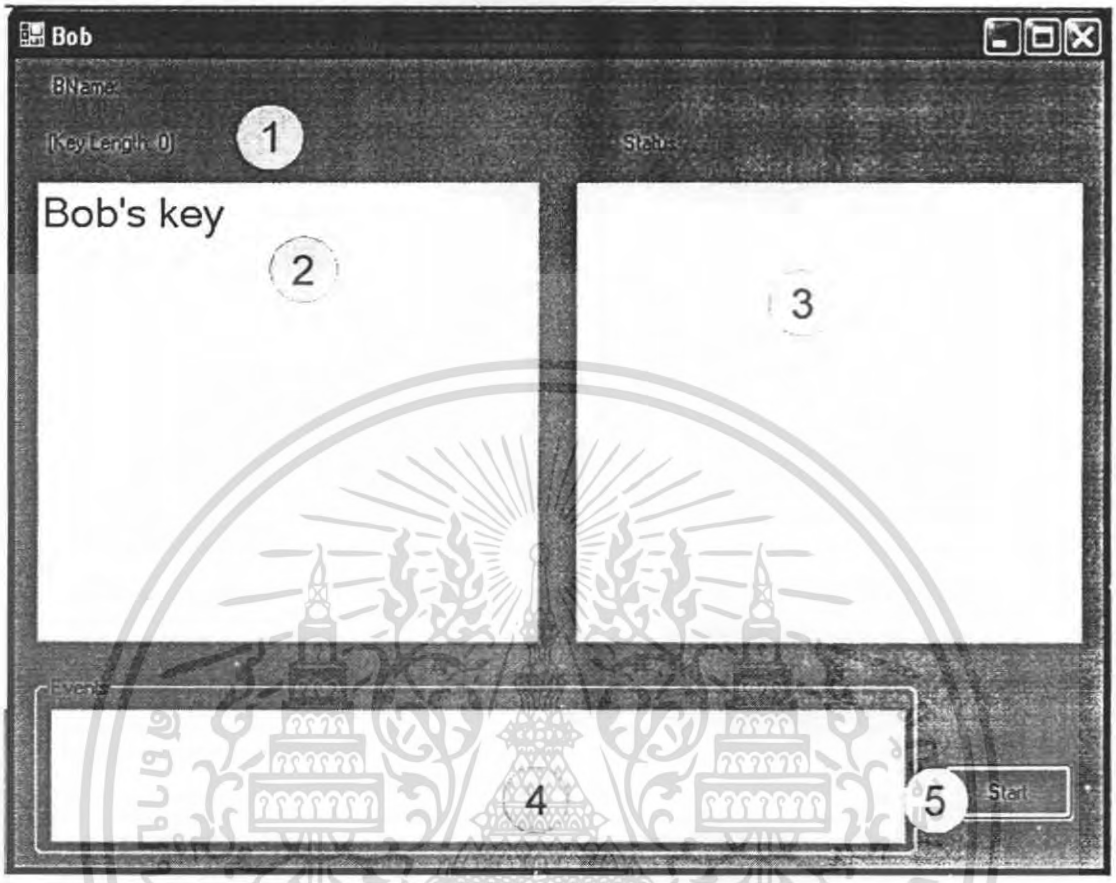
หมายเลข 10 แสดงประวัติการทำงานของโปรแกรม

หมายเลข 11 ปุ่มสั่งงานให้โปรแกรมเริ่มทำงานหรือทำงานในขั้นตอนการคำนวณค่าต่างๆต่อไป

หมายเลข 12 กำหนดค่าความผิดพลาดของกุญแจรหัสลับโดยการสุ่มขึ้นมาเท่านั้น

หมายเลข 13 กำหนดและแสดงค่าความผิดพลาดของกุญแจรหัสลับที่จะให้เกิดขึ้นกับกุญแจรหัสลับ

หมายเลข 14 ปุ่มยกเลิกการทำงานทั้งหมดของโปรแกรม



รูปที่ 3.4 แอพพลิเคชันฟอร์มของ *Bob*

3.4.2 แอพพลิเคชันฟอร์มของ *Bob* มีรายละเอียดดังรูปที่ 3.4 ซึ่งแต่ละหมายเลขมีรายละเอียดดังนี้

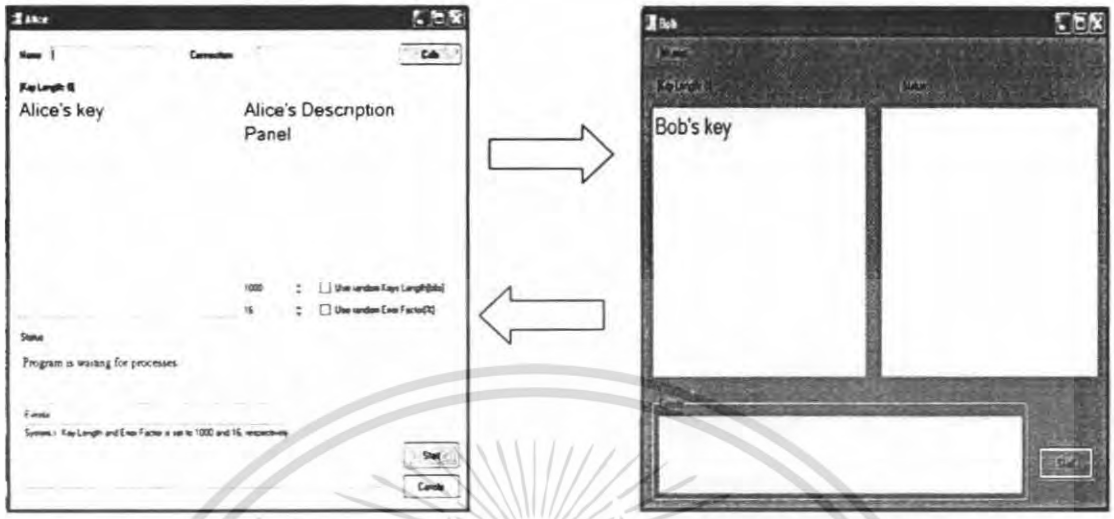
หมายเลข 1 แสดงขนาดของกุญแจรหัสลับของ *Bob*

หมายเลข 2 แสดงเวกเตอร์ฐานและกุญแจรหัสลับที่สุ่มขึ้นมาของ *Bob* แล้วหลังจากนั้นจะแสดงเพียงกุญแจรหัสลับเท่านั้น

หมายเลข 3 แสดงค่าสถานะการทำงานในขณะนั้นของโปรแกรม

หมายเลข 4 แสดงประวัติการทำงานของโปรแกรม

หมายเลข 5 ปุ่มที่ใช้สั่งงานให้โปรแกรมเริ่มทำงานหรือทำงานในขั้นตอนการคำนวณค่าต่างๆต่อไป



รูปที่ 3.5 ลักษณะการทำงานของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม

3.4.3 ลักษณะการทำงานของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม

ในการทำงานของโปรแกรมนั้นได้ออกแบบให้แอพพลิเคชันของ *Alice* และ *Bob* นั้นสามารถติดต่อสื่อสารกันผ่านตัวแปรที่ถูกสร้างขึ้นในโมดูล (Module) ซึ่งกระบวนการทำงานทั้งหมดของโปรแกรมจะกระทำผ่านตัวแปรสาธารณะนี้ทั้งหมด และใช้แอพพลิเคชันของ *Alice* เป็นตัวควบคุมการทำงานทั้งหมดของโปรแกรม ซึ่งมีลักษณะการทำงานดังรูปที่ 3.5

บทที่ 4
ผลการทดลอง

จากการออกแบบโปรแกรมจำลองการทำงานกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัม และการจำลองการทำงานของกระบวนการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE ที่ได้ทำการศึกษา เพื่อแสดงผลการทำงานและวิธีการใช้งานโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม ดังนั้นในบทนี้จึงนำเสนอวิธีการใช้งานและผลการจำลองการทำงานของกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัม ผลการจำลองการทำงานของกระบวนการแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE ที่ได้ทำการศึกษา ซึ่งรายละเอียดมีดังต่อไปนี้

4.1 วิธีการใช้และการแสดงผลการทำงานของโปรแกรม

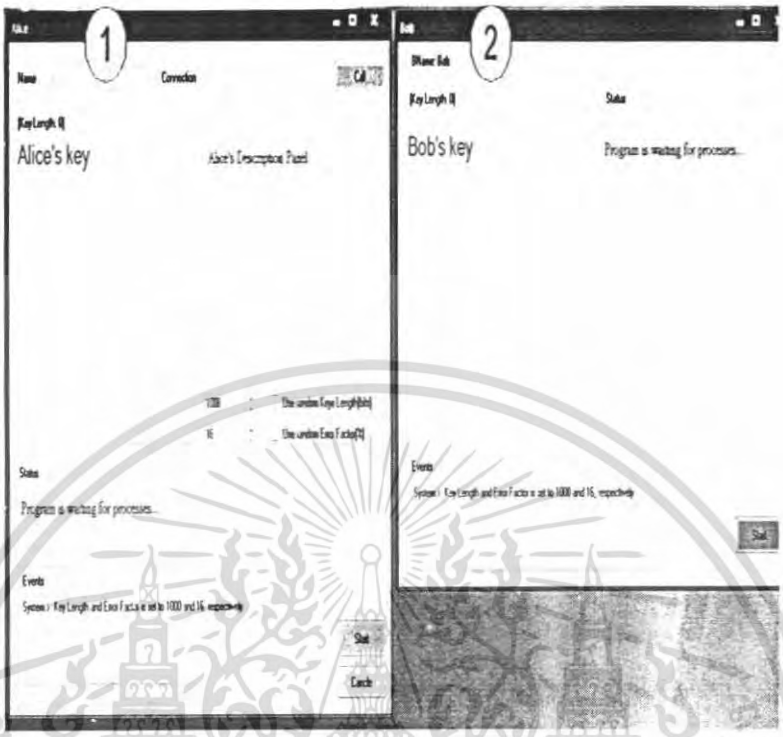
โปรแกรมที่สร้างขึ้นมานั้นเป็นการจำลองการทำงานกระบวนการต่างๆของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัม ซึ่งเขียนโดยใช้ภาษา Visual Basic 2005 ซึ่งโปรแกรมมีรูปแบบการแสดงผลที่เป็นแอพพลิเคชันฟอร์ม (Form Application) ที่สามารถใช้งานได้ง่าย ซึ่งการใช้งานโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมที่จัดทำขึ้นมานั้น มีขั้นตอนการใช้งานและการแสดงผลตามลำดับขั้นตอนต่างๆดังนี้

ขั้นตอนที่ 1 เมื่อทำการรันโปรแกรมขึ้นมา โปรแกรมจะแสดงหน้าต่างแอพพลิเคชันฟอร์มของ *Alice* ดังรูปที่ 4.1

ขั้นตอนที่ 2 ทำการใส่ชื่อหรือ IP ของตัวเองในหมายเลข 1 และชื่อหรือ IP ของบุคคลที่ต้องการจะติดต่อในหมายเลข 2 และทำการกดปุ่ม "Call" ในหมายเลข 3 ตามรูปที่ 4.2 แล้วหน้าต่างแอพพลิเคชันฟอร์มของ *Bob* จะแสดงขึ้นมาดังรูปที่ 4.3 ซึ่งจะมีการแสดงขึ้นมาสองแอพพลิเคชันฟอร์มคือ หมายเลข 1เป็นแอพพลิเคชันฟอร์มของ *Alice* และหมายเลข 2 เป็นแอพพลิเคชันฟอร์มของ *Bob*

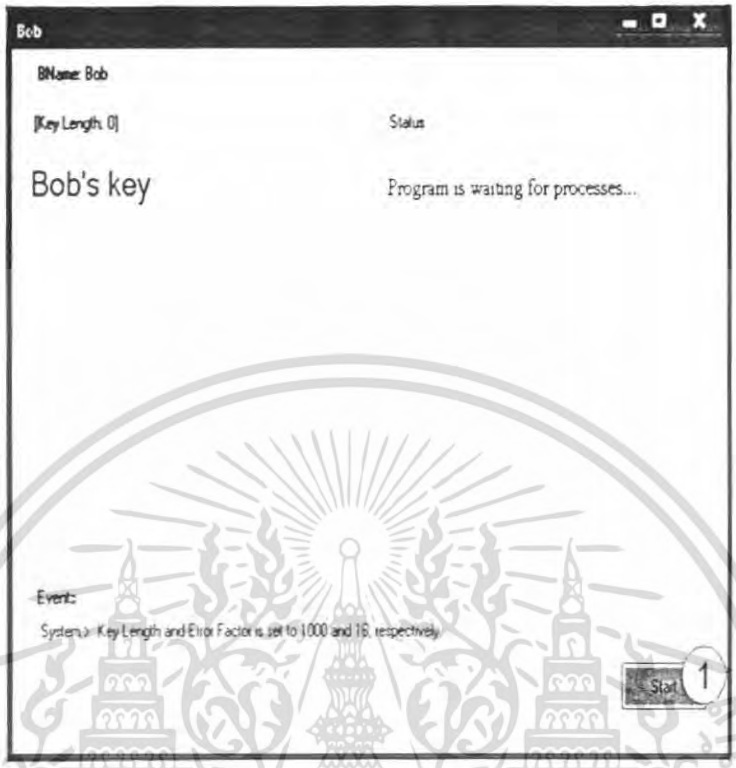
รูปที่ 4.1 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 1

รูปที่ 4.2 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 2



รูปที่ 4.3 แอพพลิเคชันฟอร์มของ *Alice* และ *Bob* ที่เป็นผลการทำงานในขั้นตอนที่ 2

รูปที่ 4.4 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 3



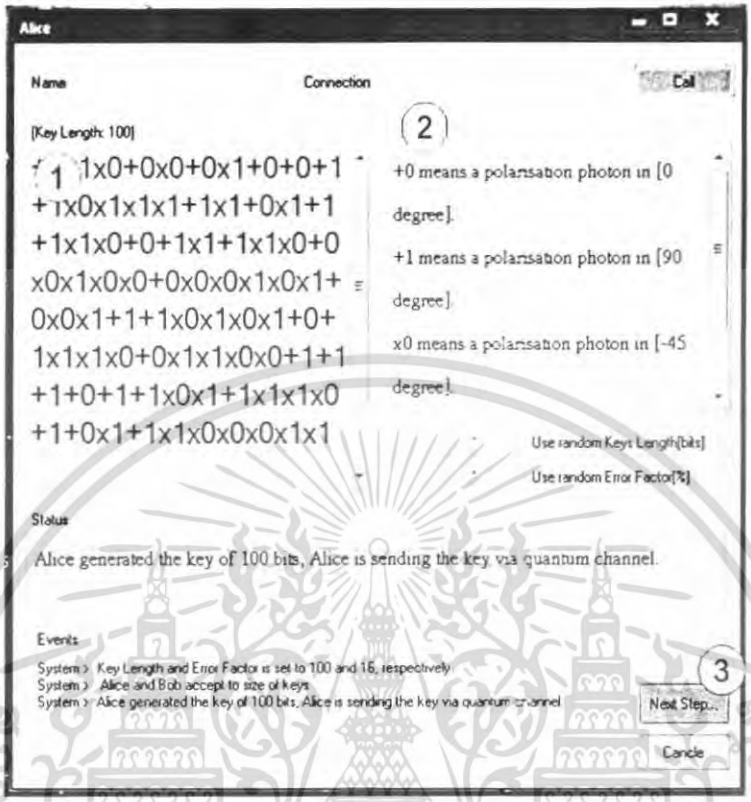
รูปที่ 4.5 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 3

ขั้นตอนที่ 3 ใส่ขนาดของกุญแจรหัสลับที่หมายเลข 1 และ ใส่ค่าความผิดพลาดที่หมายเลข 2 ซึ่งค่าทั้งสองนั้นจะกำหนดขึ้นเองหรือจะเลือกโดยให้โปรแกรมสุ่มขึ้นมาให้ โดยที่ถ้าจะสุ่มขึ้นมานั้นให้เลือกที่หมายเลข 3 และหมายเลข 4 จากนั้นให้กดที่ปุ่ม "Start" ที่หมายเลข 5 ในแอพพลิเคชันฟอร์มของ *Alice* จากนั้นที่แอพพลิเคชันฟอร์มของ *Bob* ให้กดที่ปุ่ม "Start" ที่หมายเลข 1 ดังรูปที่ 4.5

ซึ่งผลจากการทำในขั้นตอนที่ 3 นี้ที่แอพพลิเคชันฟอร์มของ *Alice* แสดงออกมมาดังรูปที่ 4.6 โดยที่หมายเลข 1 จะแสดงว่าระบบทำงานถึงขั้นตอนในการตกลงกันว่ากุญแจรหัสลับจะมีขนาดเท่าที่กำหนด และหมายเลข 2 จะแสดงข้อมูลที่ผู้รับได้รับตามขนาดที่ตกลงกันไว้ ส่วนแอพพลิเคชันฟอร์มของ *Bob* นั้นจะแสดงค่าขนาดกุญแจรหัสลับที่ตกลงกันไว้ที่หมายเลข 1 หมายเลข 2 จะแสดงว่าระบบทำงานถึงขั้นตอนว่าตกลงว่ากุญแจรหัสลับจะมีขนาดเท่าที่กำหนด ดังรูปที่ 4.7

รูปที่ 4.6 ผลการทำงานของแอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 3

รูปที่ 4.7 ผลการทำงานของแอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 3



รูปที่ 4.8 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 4

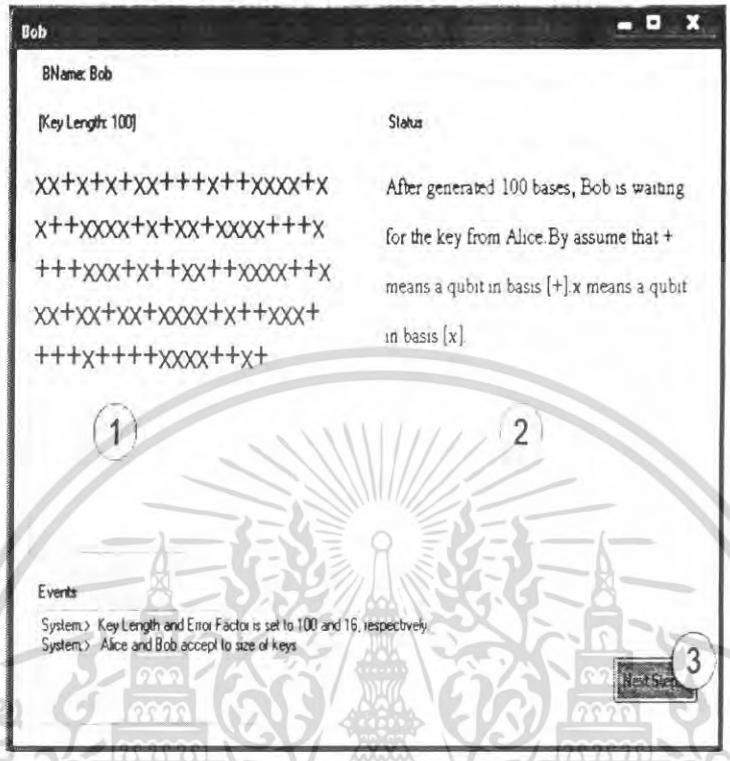
ขั้นตอนที่ 4 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 3 ของแอพพลิเคชันฟอร์ม *Alice* จะทำการสุ่มกุญแจรหัสลับและเวกเตอร์ฐานในการส่งกุญแจรหัสลับซึ่งค่าที่แสดงออกมาจะใช้สองบิตข้อมูลที่อยู่ติดกันแทนสถานะโพลาไรเซชันหนึ่งสถานะ ดังรูปที่ 4.8 ซึ่งค่าที่แสดงออกมาที่หมายเลข 1 นั้นความหมายของแต่ละบิตจะอธิบายโดยหมายเลข 2 ดังนี้

" +0 " แทนสถานะโพลาไรเซชัน "0 องศา" " +1 " แทนสถานะโพลาไรเซชัน "90 องศา",
" x0 " แทนสถานะโพลาไรเซชัน "-45 องศา" " x1 " แทนสถานะโพลาไรเซชัน "45 องศา"

ขั้นตอนที่ 5 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 3 ของแอพพลิเคชันฟอร์ม *Bob* เพื่อทำการสุ่มเวกเตอร์ฐานโดยที่จะแสดงผลขึ้นมาที่หมายเลข 1 ซึ่งค่าที่แสดงออกมาดังรูปที่ 4.9 ซึ่งความหมายของแต่ละบิตจะอธิบายโดยหมายเลข 2 ดังนี้

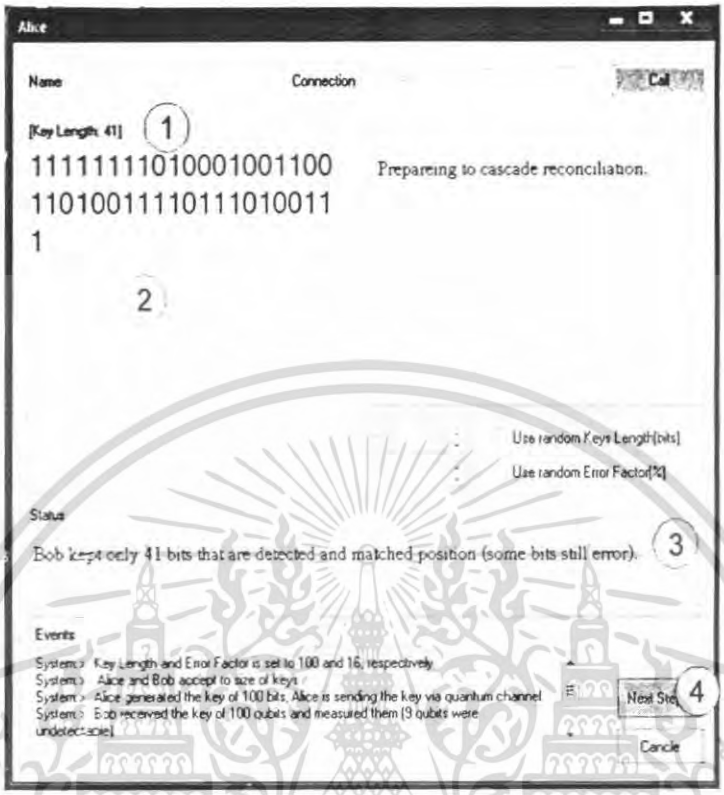
" + " ใช้ในการรับสถานะโพลาไรเซชัน "0 องศา" และ "90 องศา"

" x " ใช้ในการรับสถานะโพลาไรเซชัน "-45 องศา" และ "45 องศา"



รูปที่ 4.9 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 5

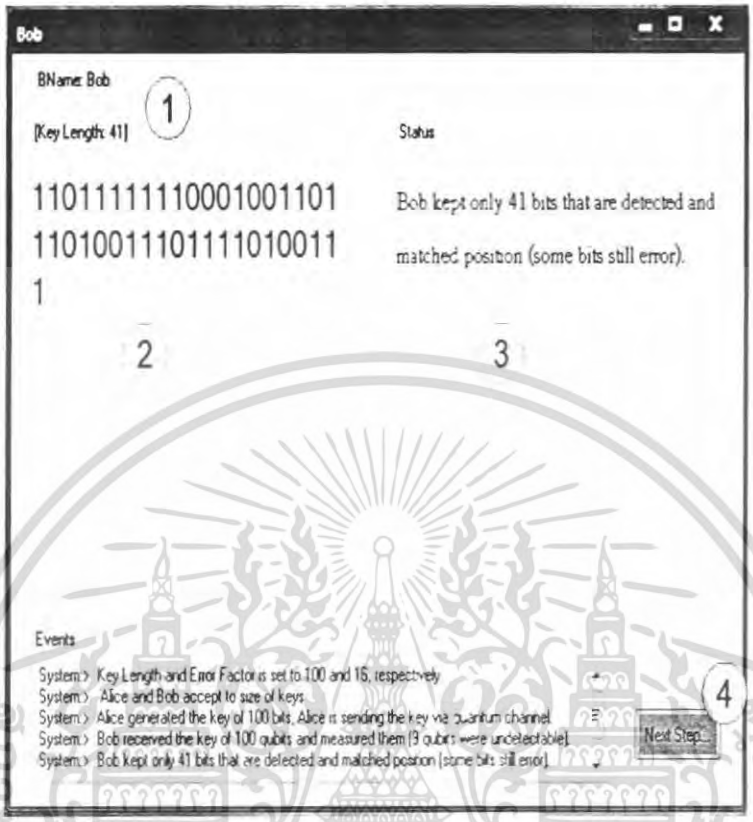
รูปที่ 4.10 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 6



รูปที่ 4.11 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 7

ขั้นตอนที่ 6 ให้กดที่ปุ่ม "Next Step" ของแอพพลิเคชันฟอร์ม *Alice* จะทำการส่งสถานะโพลาไรเซชันไปยังแอพพลิเคชันฟอร์มของ *Bob* แล้วให้กดที่ปุ่ม "Next Step" ที่หมายเลข 3 ของแอพพลิเคชันฟอร์ม *Bob* แล้วจะแสดงกุญแจรหัสลับบิตที่รับได้ทั้งหมดที่หมายเลข 1 ดังรูปที่ 4.10 และหมายเลข 2 จะบอกว่าส่งมาทั้งหมดกี่บิตและมีเวกเตอร์ฐานที่สุ่มขึ้นมาแล้วไม่ตรงกับแนวของสถานะโพลาไรเซชันกี่บิต

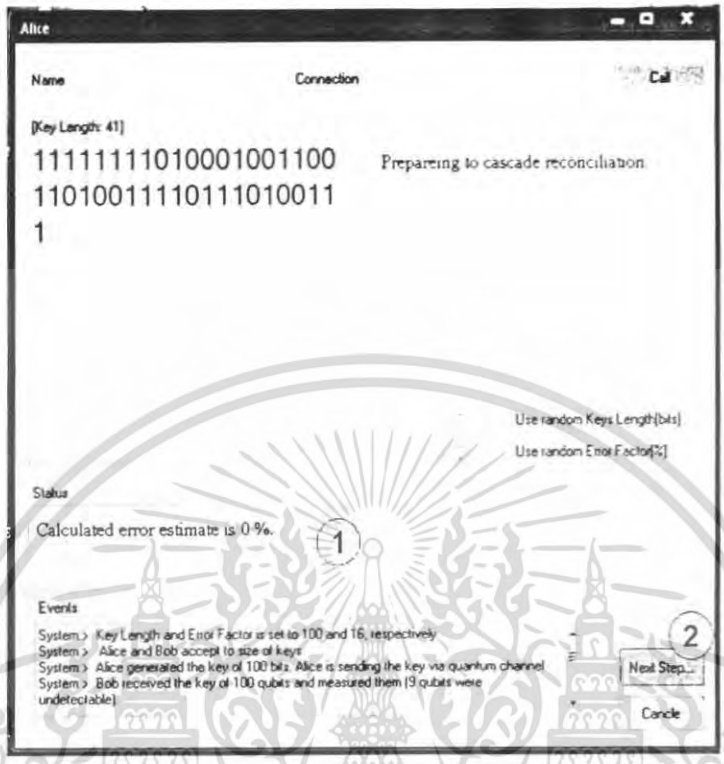
ขั้นตอนที่ 7 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 4 ของแอพพลิเคชันฟอร์ม *Alice* เพื่อทำการตรวจสอบเวกเตอร์ฐาน แล้วแสดงกุญแจรหัสลับบิตที่เหลือจากการตัดกุญแจรหัสลับบิตในตำแหน่งที่มีเวกเตอร์ฐานไม่ตรงกันไว้ที่หมายเลข 2 ส่วนหมายเลข 1 จะแสดงขนาดกุญแจรหัสลับที่เหลือ และหมายเลข 3 จะบอกว่าเหลือบิตที่เหลือจากการตรวจสอบเวกเตอร์ฐานกี่บิต ดังรูปที่ 4.11



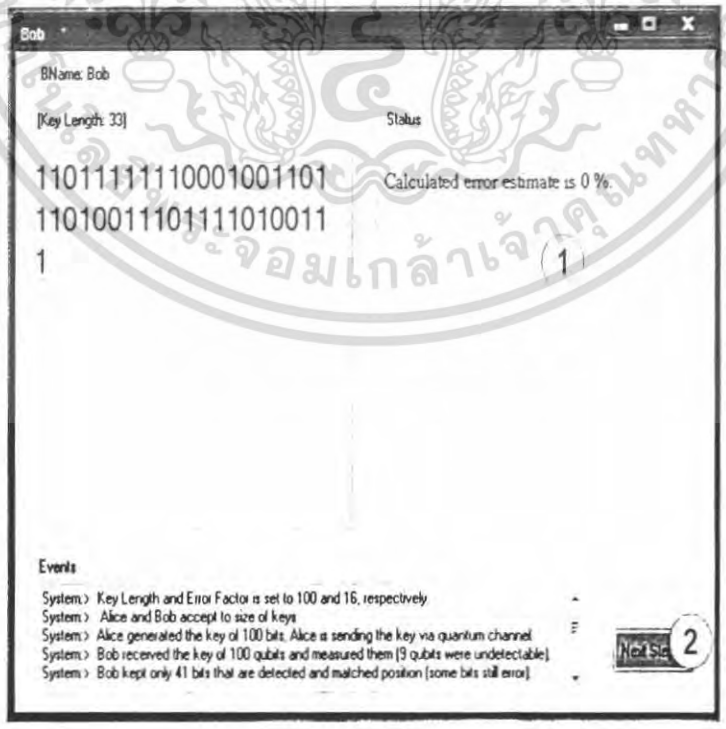
รูปที่4.12 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 8

ขั้นตอนที่ 8 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 4 ของแอพพลิเคชันฟอร์ม *Bob* เพื่อทำการตรวจสอบเวกเตอร์ฐาน แล้วแสดงกุญแจรหัสลับบิตที่เหลือจากการตัดกุญแจรหัสลับบิตในตำแหน่งที่มีเวกเตอร์ฐานไม่ตรงกันไว้ที่หมายเลข 2 ส่วนหมายเลข 1 จะแสดงขนาดกุญแจรหัสลับที่เหลือ และหมายเลข 3 จะบอกว่าเหลือบิตที่เหลือจากการตรวจสอบเวกเตอร์ฐานกี่บิตดังรูปที่ 4.12

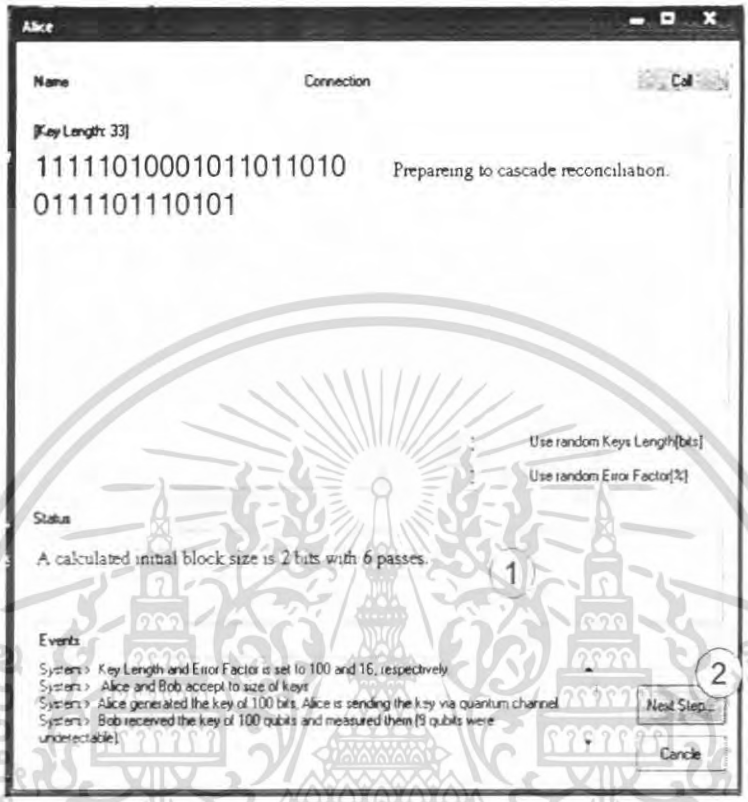
ขั้นตอนที่ 9 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 2 ของแอพพลิเคชันฟอร์ม *Alice* เพื่อทำการคำนวณค่าความผิดพลาดที่จะนำไปคำนวณค่าขนาดของบล็อกเริ่มต้นและจำนวนรอบในการทำการแก้ไขความผิดพลาด ซึ่งค่าที่ได้จะแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.13 และกดที่ปุ่ม "Next Step" ที่หมายเลข 2 ของแอพพลิเคชันฟอร์ม *Bob* เพื่อทำการคำนวณค่าความผิดพลาด ที่จะนำไปคำนวณค่าขนาดของบล็อกเริ่มต้นและจำนวนรอบในการทำการแก้ไขความผิดพลาด ซึ่งค่าที่ได้จะแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.14



รูปที่ 4.13 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 9



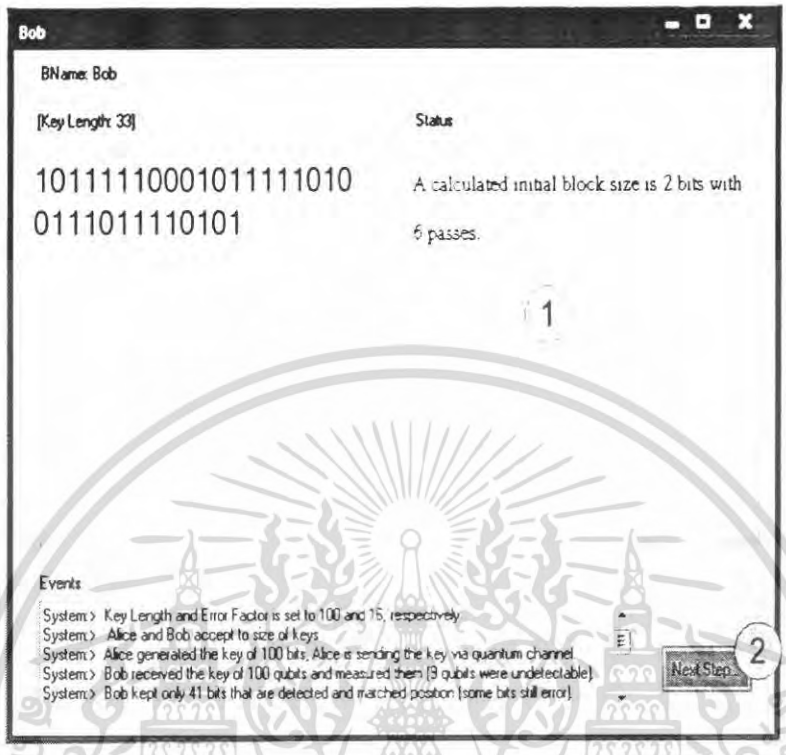
รูปที่ 4.14 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 9



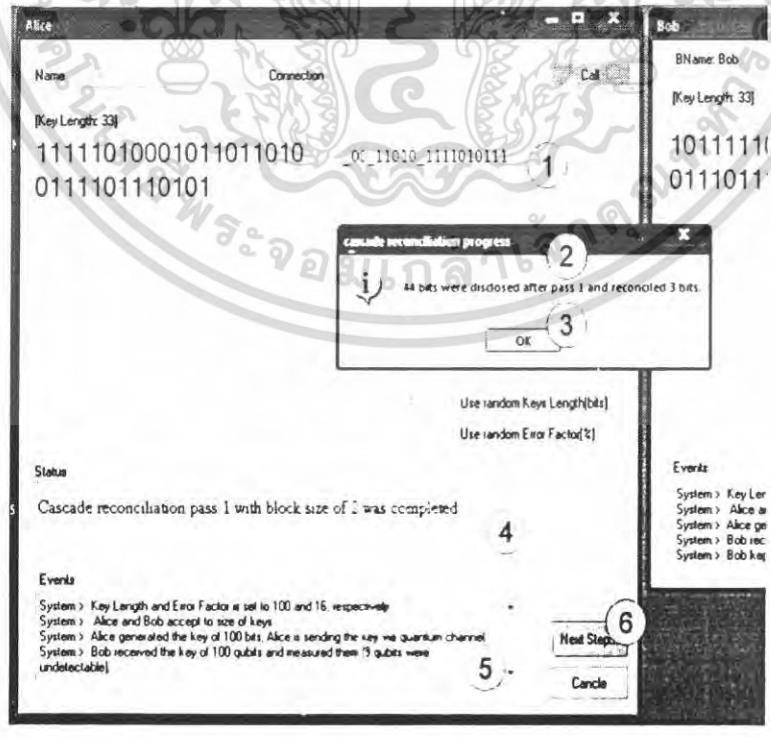
รูปที่ 4.15 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 10

ขั้นตอนที่ 10 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 2 ของแอพพลิเคชันฟอร์มของ *Alice* อีกครั้งเพื่อทำการคำนวณค่าขนาดของบล็อกเริ่มต้นและจำนวนรอบในการวนซ้ำเพื่อทำการแก้ไขความผิดพลาด และค่าที่ได้จะแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.15

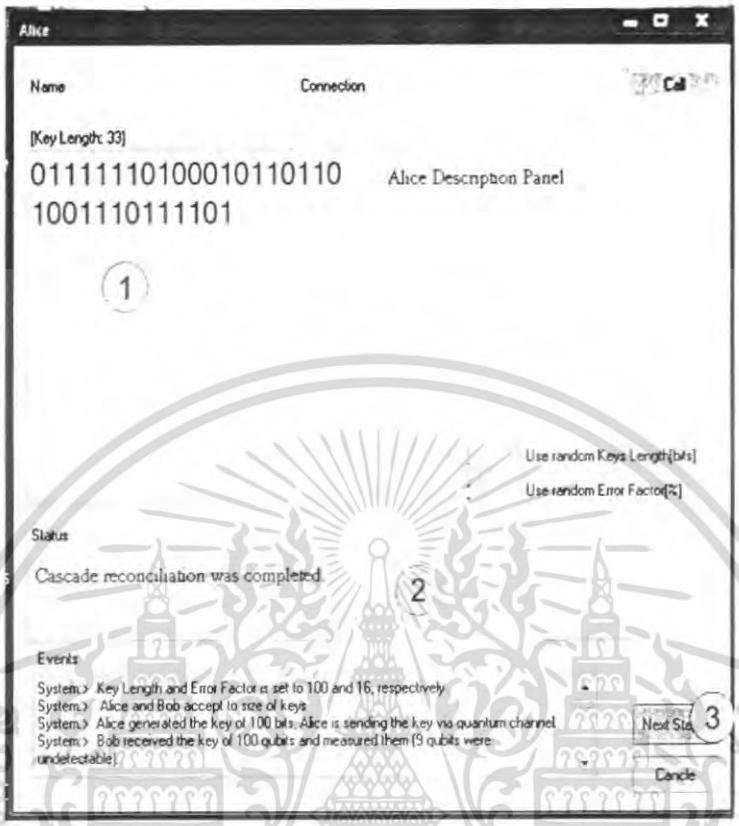
ขั้นตอนที่ 11 ให้และกดที่ปุ่ม "Next Step" ที่หมายเลข 2 ของแอพพลิเคชันฟอร์มของ *Bob* อีกครั้งเพื่อทำการคำนวณค่าขนาดของบล็อกเริ่มต้นและจำนวนรอบในการวนซ้ำเพื่อทำการแก้ไขความผิดพลาด และค่าที่ได้จะแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.16



รูปที่ 4.16 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 10



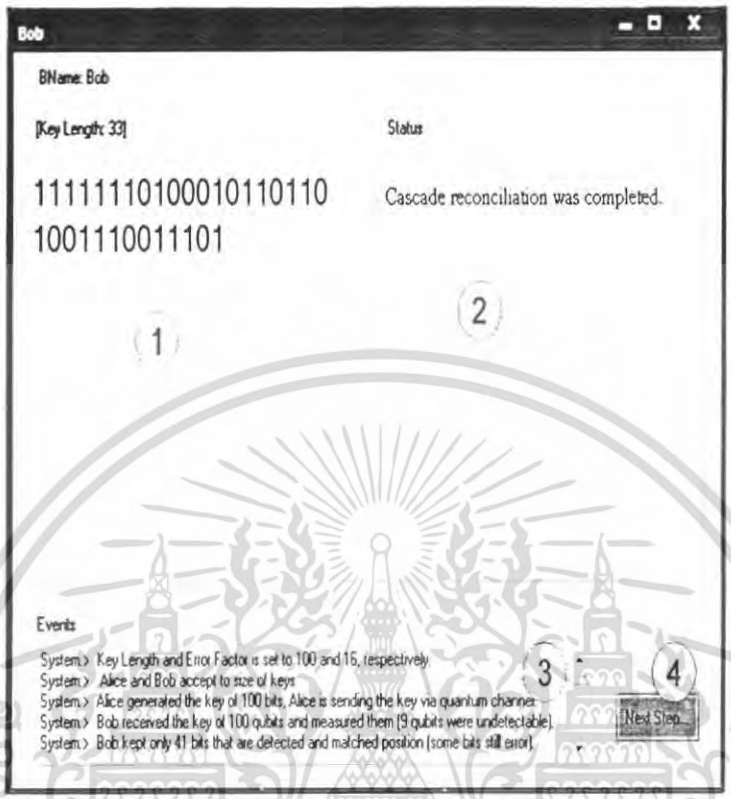
รูปที่ 4.17 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 12



รูปที่ 4.18 ผลของแอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 12

ขั้นตอนที่ 12 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 6 ของแอพพลิเคชันฟอร์มของ *Alice* เพื่อทำการแบ่งกุญแจรหัสลับบิตตามขนาดของบล็อกค่าเริ่มต้นแล้วหาพาริตี้บิตของแต่ละบล็อก จากนั้นนำพาริตี้บิตของ *Alice* และ *Bob* มาทำการเปรียบเทียบกันถ้าตรงกันให้แสดงบิตนั้นออกมา แต่ถ้าไม่ตรงกันให้แสดงเป็น " _ " ซึ่งผลจะแสดงขึ้นมาที่หมายเลข 1 จากนั้นก็จะเรียกฟังก์ชันเพื่อมาทำการแก้ไขความผิดพลาดและแสดงค่าออกมาเป็น Massage Box ที่จะบอกว่ามีบิตผิดพลาดกี่บิตที่หมายเลข 2 แล้วให้กดปุ่ม OK ที่หมายเลข 3 เพื่อประมวลผลในรอบต่อไป ซึ่งทุกรอบจะแสดงผลการแก้ไขในแต่ละรอบที่หมายเลข 4 จนครบตามจำนวนรอบที่คำนวณไว้และแสดงกุญแจรหัสลับที่เหลือจากการผ่านกระบวนการแก้ไขความผิดพลาด ซึ่งผลการทำงานในแต่ละรอบจะมีการอธิบายการทำงานไว้ที่หมายเลข 5 ดังแสดงในรูปที่ 4.17

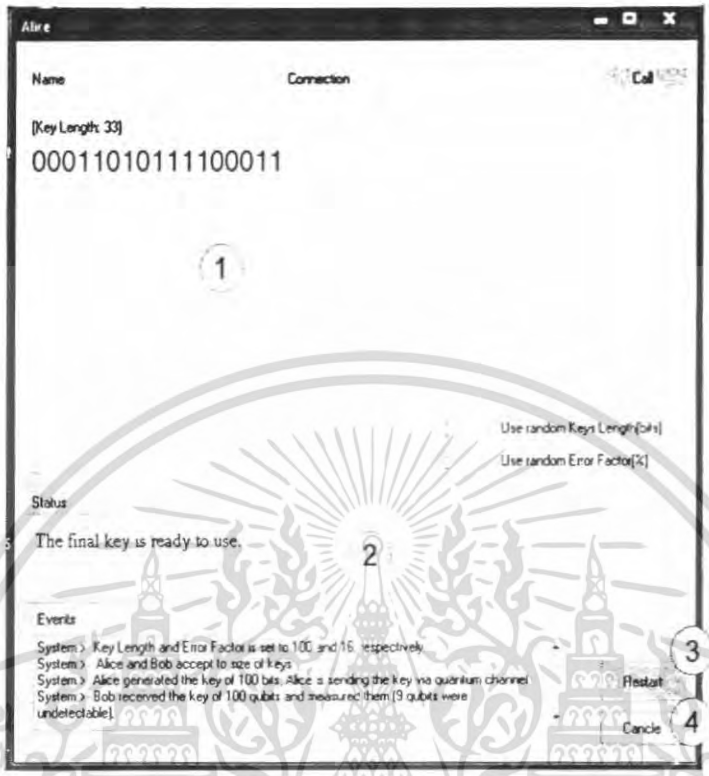
ซึ่งหลังจากการทำงานจนครบทุกรอบแล้วจะแสดงผลสุดท้ายออกมาดังรูปที่ 4.18 ซึ่งหมายเลข 1 จะแสดงกุญแจรหัสลับที่ได้มีการแก้ไขความผิดพลาดเรียบร้อยแล้ว หมายเลข 2 จะบอกว่าการทำงานทุกกระบวนการในการแก้ไขความผิดพลาดนั้นเรียบร้อยแล้ว



รูปที่ 4.19 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 13

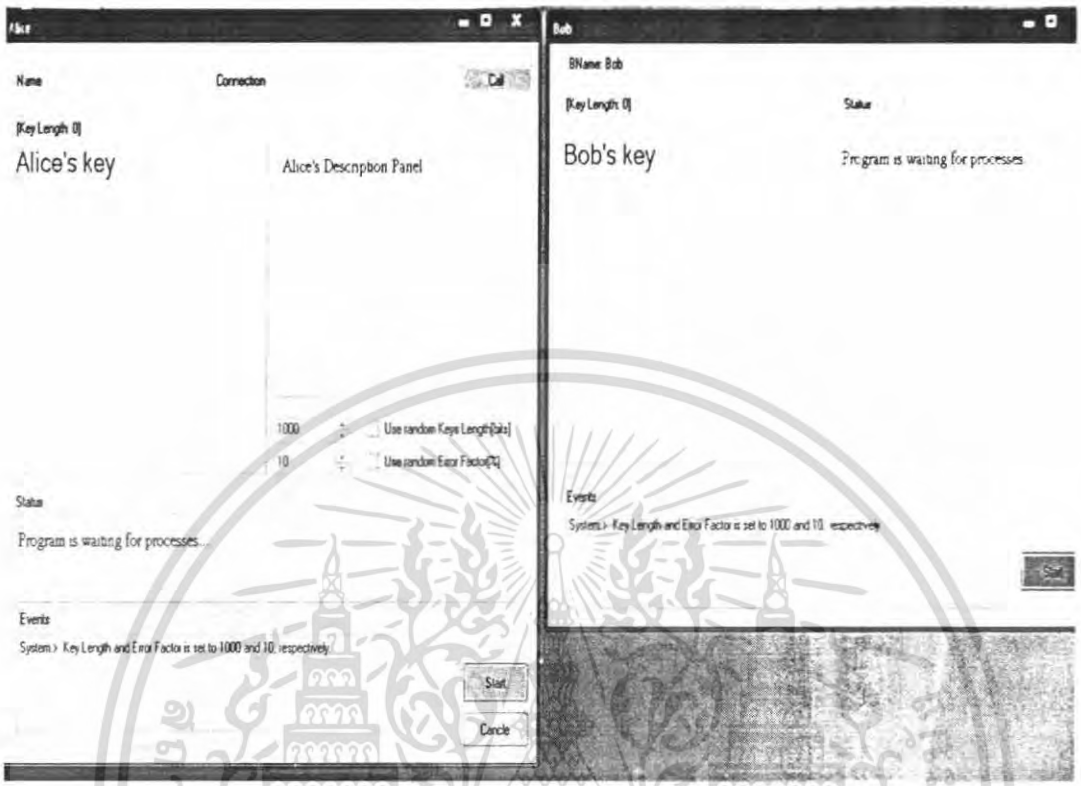
ขั้นตอนที่ 13 ให้กดที่ปุ่ม "Next Step" ที่หมายเลข 4 ของแอพพลิเคชันฟอร์มของ *Bob* เพื่อแสดงกุญแจรหัสลับที่ได้หลังจากการแก้ไขความผิดพลาดออกมาที่หมายเลข 1 ดังรูปที่ 4.19 หมายเลข 2 จะบอกว่าการทำงานทุกกระบวนการในการแก้ไขความผิดพลาดนั้นเรียบร้อยแล้ว และทุกรอบจะแสดงผลการแก้ไขในแต่ละรอบที่หมายเลข 3

ขั้นตอนที่ 14 ให้กดที่ปุ่ม "Next Step" ที่แอพพลิเคชันฟอร์มของ *Alice* เพื่อทำกระบวนการขยายสภาวะส่วนตัวกับกุญแจรหัสลับที่ *Alice* มีแล้วแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.20 ส่วนหมายเลข 2 จะบอกว่ากุญแจรหัสลับที่ได้นั้นพร้อมนำไปใช้งานได้แล้ว และปุ่ม Next Step จะเปลี่ยนเป็นปุ่ม "Restart" ดังหมายเลข 3 ซึ่งหากต้องการเริ่มการทำงานของโปรแกรมไปยังค่าเริ่มต้นใหม่อีกครั้งให้กดที่ปุ่มหมายเลข 3 จะแสดงผลออกมาดังรูปที่ 4.22 แต่หากต้องการปิดโปรแกรมก็กดที่หมายเลข 4 คือปุ่ม "Cancle"



รูปที่ 4.20 แอพพลิเคชันฟอร์มของ *Alice* ในขั้นตอนที่ 14

รูปที่ 4.21 แอพพลิเคชันฟอร์มของ *Bob* ในขั้นตอนที่ 15

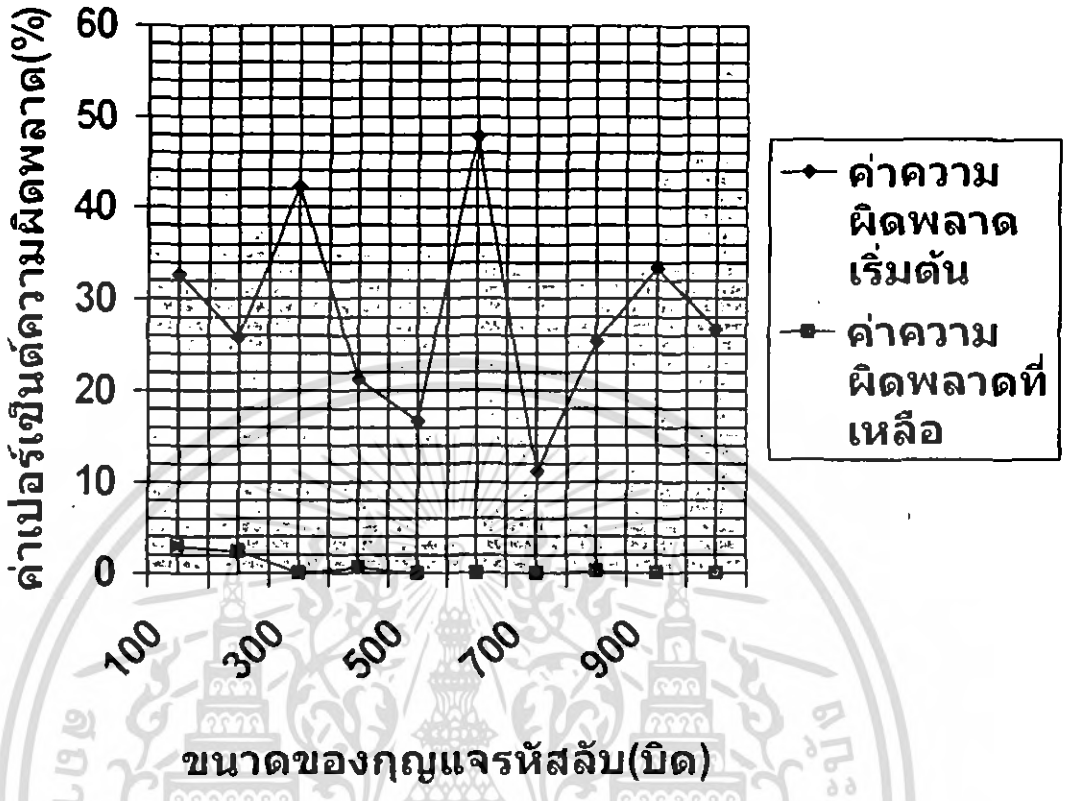


รูปที่ 4.22 แอพพลิเคชันฟอร์มของ *Alice* และ *Bob* เมื่อกดปุ่ม "Restart"

ขั้นตอนที่ 15 ให้กดที่ปุ่ม "Next Step" ที่แอพพลิเคชันฟอร์มของ *Bob* เพื่อทำกระบวนการขยายสภาวะส่วนตัวกับกุญแจรหัสลับที่ *Bob* มีแล้วแสดงออกมาที่หมายเลข 1 ดังรูปที่ 4.21 ส่วนหมายเลข 2 จะบอกว่ากุญแจรหัสลับที่ได้นั้นพร้อมนำไปใช้งานได้แล้ว และปุ่ม "Next Step" จะเปลี่ยนเป็นปุ่ม "Restart" ดังหมายเลข 3 ซึ่งหากต้องการเริ่มการทำงานของโปรแกรมไปยังค่าเริ่มต้นใหม่อีกครั้งให้กดที่ปุ่มหมายเลข 3 จะแสดงผลออกมาดังรูปที่ 4.22

4.2 ประสิทธิภาพของการทำงานของโปรแกรม

โดยที่การทำงานของโปรแกรมแก้ไขความผิดพลาดที่จักทำขึ้นมานั้นสามารถแสดงออกมาให้เห็นถึงประสิทธิภาพการทำงานของโปรแกรม โดยแสดงออกมาในรูปของกราฟเปรียบเทียบระหว่าง ค่า ความผิดพลาด ที่เกิดขึ้นก่อนทำการแก้ไขความผิดพลาดกับค่า ความผิดพลาด ที่เหลืออยู่ ซึ่งจากรูปแสดงให้เห็นว่าค่า ความผิดพลาด ลดน้อยลงหรือไม่มีเลย ดังแสดงในรูปที่ 4.23



รูปที่ 4.23 กราฟแสดงเปรียบเทียบค่าความผิดพลาดที่เกิดขึ้นทั้งก่อนและหลังจากการแก้ไขความผิดพลาด

บทที่5
สรุปผลการดำเนินงาน

บทนี้ได้กล่าวถึงบทสรุปจากการเรียนรู้วิทยาการรหัสลับเชิงควอนตัมและการจัดทำซอฟท์แวร์จำลองการทำงาน มีการกล่าวถึงปัญหาและอุปสรรคที่พบระหว่างการจัดทำปริญญานิพนธ์ แล้วยังมีการนำเสนอแนวทางในการพัฒนาและแก้ปัญหา ซึ่งมีรายละเอียดดังต่อไปนี้

5.1 สรุปผลการจัดทำโปรแกรมการ

ระบบวิทยาการรหัสลับเชิงควอนตัมเป็นระบบที่ใช้ส่งกุญแจรหัสลับระหว่างผู้ส่งและผู้รับ โดยใช้กฎกลศาสตร์ควอนตัมมาช่วยยืนยันความปลอดภัยของระบบ หากบุคคลที่สามหรือผู้ไม่พึงประสงค์เข้ามาขโมยกุญแจรหัสลับระหว่างการส่ง ผู้ส่งและผู้รับจะทราบทันทีถึงการเข้ามาขโมยกุญแจรหัสลับและสามารถยกเลิกการส่งกุญแจรหัสลับได้ทันก่อนเกิดความเสียหายตามมา ทำให้ระบบมีความปลอดภัยและกุญแจรหัสลับไม่ถูกเปิดเผย แต่ในระหว่างการส่งสัญญาณรบกวนความไม่เป็นอุดมคติของอุปกรณ์ภายในภาคส่งและภาครับและการเข้ามาขโมยสถานะควอนตัมของแสงของบุคคลที่สามภายในช่องสื่อสารเชิงควอนตัม เป็นสาเหตุทำให้สถานะควอนตัมเกิดการเปลี่ยนแปลงส่งผลให้กุญแจรหัสลับที่ส่งเกิดความผิดพลาดตามไปด้วย ดังนั้นการแก้ไขความผิดพลาดและการลดความสำคัญของข้อมูลเกี่ยวกับกุญแจรหัสลับที่บุคคลที่สามสามารถขโมยได้ จึงมีความสำคัญเพื่อทำให้กุญแจรหัสลับที่ส่งยังคงเป็นความลับ ซึ่งโพรโทคอลแก้ไขความผิดพลาดที่ใช้ในปัจจุบัน เช่น โพรโทคอล CADCADE โพรโทคอล BBBSS เป็นต้น สามารถที่จะแก้ไขความผิดพลาดได้เป็นอย่างดีและเปิดเผยข้อมูลเกี่ยวกับกุญแจรหัสลับน้อย ซึ่งในปริญญานิพนธ์นี้ได้นำเสนอโปรแกรมจำลองการทำงานของวิทยาการรหัสลับเชิงควอนตัม ที่มีการแสดงกระบวนการกระจายกุญแจรหัสลับด้วยโพรโทคอล BB84 และใช้การแก้ไขความผิดพลาดด้วยโพรโทคอล CASCADE นั้นสามารถแสดงการจำลองกระบวนการต่างๆเพื่อสาธิตกระบวนการทั้งสองนี้ ผลจากการประมวลผลของโปรแกรมจะแสดงออกมาในรูปบิตข้อมูล โดยที่ผลสุดท้ายหลังจากเสร็จสิ้นกระบวนการทำงานทั้งหมดของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมจะเป็นกุญแจรหัสลับที่พร้อมนำไปใช้ในการเข้ารหัสลับและถอดรหัสลับต่อไป และมีการกราฟแสดงประสิทธิภาพการทำงานของโปรแกรม ซึ่งจากกราฟแสดงให้เห็นว่าการทำงานของโปรแกรมการแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมด้วยโพรโทคอล

CASCADE นั้นสามารถแก้ไขความผิดพลาดที่เกิดขึ้นได้ดี แต่วิธีการของโพรโทคอลCASCADEนี้ต้องใช้จำนวนรอบในการติดต่อสื่อสารที่มากทำให้ไม่เหมาะกับการติดต่อสื่อสารที่ต้องใช้ความรวดเร็ว และจากการจัดทำโปรแกรมจำลองกระบวนการแก้ไขความผิดพลาดจากกระบวนการกระจายกุญแจรหัสลับเชิงควอนตัมนั้นทำให้กล่าวได้ว่าวิทยาการรหัสลับเชิงควอนตัมนั้นเป็นวิทยาการรหัสลับที่มีความปลอดภัยและเหมาะสมในการใช้เก็บความลับของข้อมูลที่ใช้ในการสื่อสารได้เป็นอย่างดี

5.2 ปัญหาและอุปสรรคที่พบในงาน

ในการทำโครงงานนี้ได้พบปัญหาในทางเทคนิคที่เกิดอันเนื่องมาจากโปรแกรมภาษา Visual Basic 2005 ต้องทำงานในระบบปฏิบัติการขั้นสูง ปัญหาที่เกิดขึ้นจึงเป็นปัญหา ที่เกิดขึ้นนอกเหนือไปจากตัวโปรแกรมเอง เช่นโปรแกรมไม่สามารถเปิดใช้งานได้กับคอมพิวเตอร์บางเครื่อง โปรแกรมแสดงผลทางด้านกราฟฟิกที่ผิดเพี้ยนไป เป็นต้น

และเนื่องจากโพรโทคอล CASCADE นั้นใช้พาริตี้บิตในการตรวจสอบดังนั้นโปรแกรมที่สร้างขึ้นเพื่อจำลองการทำงานในกระบวนการต่างๆทำให้การประมวลผลเกิดความผิดพลาดขึ้น

5.3 แนวทางในการพัฒนาและแก้ไขปัญหา

เนื่องจากปัญหาที่เกิดขึ้นนั้นเป็นปัญหาในทางเทคนิคที่เกิดจากโปรแกรมภาษา Visual Basic 2005 ที่สามารถทำการแก้ไขได้โดย ติดตั้ง Runtime Library ที่ชื่อว่า Microsoft .NET Framework ก่อนที่จะเปิดใช้งานโปรแกรม และปัญหาที่เกิดจากการทำงานของตัวโปรแกรมเองนั้นสามารถแก้ไขโดยการเพิ่มจำนวนรอบหรือมีการสลับตำแหน่งของกุญแจรหัสลับ ทำให้โปรแกรมมีประสิทธิภาพมากขึ้นแต่โปรแกรมจะใช้เวลามากขึ้นด้วยทำให้ไม่เหมาะกับการสื่อสารที่ต้องการความรวดเร็ว

ในส่วนของการพัฒนานั้นสามารถพัฒนาให้มีการจำลองการทำงานระหว่างเครื่องคอมพิวเตอร์สองเครื่องที่มีการแลกเปลี่ยนข้อมูลสื่อสารระหว่างกันดังเช่นในระบบวิทยาการรหัสลับทั่วไปและจากการศึกษาโพรโทคอล CASCADE ปัญหาของตัวโพรโดคอลที่เกิดขึ้นนั้น หากพัฒนาโพรโทคอลแก้ไขความผิดพลาดในระบบกระจายกุญแจรหัสลับ พร้อมด้วยรหัสแก้ไขความผิดพลาดที่มีประสิทธิภาพสูง อาจจะทำให้ประสิทธิภาพของโพรโทคอลแก้ไขความผิดพลาดที่ได้มีประสิทธิภาพเพิ่มสูงขึ้น

บรรณานุกรม

[1] C.H.Bennett, F.Bessette, G.Brassard, L.Salvail and J. Smolin, "Experimental Quantum Cryptography" *Journal of Cryptography* 5(1), pp.3-28, 1992.

[2] C. Elliott, D. Pearson, G. Troxel, "Quantum Cryptography in Practice" in Proceeding of the 2003 Conference on Applications, Technology, Architectures and Protocols for Computer Communication, pp. 227-238, 2003.

[3] C.Elliott, "Building the quantum network" New J. Phys.4 (July 2002) 46.

[4] C. Elliott and all, "Current Status of The DARPA Quantum Network", eprint. arxiv:quant-ph/0503058, 2005.

[5] **Europe Research.** (มิถุนายน 2547). Available URL: http://www.thestandard.com/article.php?story=20040517152322624

[6] วุฒิกรณ์ ตรัยศิลานันท์, เกียรติศักดิ์ ศรีพิมานวัฒน์ และ อรลาภ แสงอรุณ, "การแก้ไขความผิดพลาดจากการกระจายกุญแจรหัสลับเชิงควอนตัมแบบต่อเนื่องด้วยรหัส BCH" งานประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 30 (EECON30), หน้าที่ 1033-1036, 2550.

[7] W. Traisilanun, K. Sripimanwat, O. Sangaroon, "Secret Key Reconciliation using BCH Codes" International Symposium on Communications and Information Technology 2007 (ISCIT 2007) , pp. 1482-1485 ,2007

[8] C.H. Bennett , G. Brassard and A.K. Ekert, " Quantum Cryptography", Sci. Am., 257 pp.50-57. 1992.

[9] G. Brassard and L. Salvial, "Secret-Key Reconciliation by Public Discussion", D. IRO. Canada : University de Montreal. 1998.

[10] พิทักษ์ พานทอง, "การทำงานของระบบวิทยาการรหัสลับเชิงควอนตัม" วิทยานิพนธ์วิทยาศาสตรมหาบัณฑิต สาขาฟิสิกส์, มหาวิทยาลัยเกษตรศาสตร์, 2548

[11] A. Nakassis, J. Bienfang, and C. Williams, "Expeditious reconciliation for practical quantum key distribution," in Defense and Security Symposium: Quantum Information and Computation II, Proc. SPIE 5436, 28-35 (2004).

[12] **CASCADE** [Online] (12 มีนาคม 2549) Available URL;
http://www.cki.au.dk/experiment/qrypto/templates/initiator/ReeBB84/output2_1.htm
[13] K. Nguyen, G. Van Assche and N. J. Cerf, "Side-Information Coding with Turbo Codes and its Application to Quantum Key Distribution" in Proc. International Symposium on Information Theory and Its Applications,2004.

ภาคผนวก

Code Programs

- Alice's code Form

```
Public Class Form1
    Dim Form2 As New Form2
        Public Function Reconcile(ByRef startofblock As Integer, ByRef endofblock As Integer) As Integer
        Dim blocksize As Integer = (endofblock - startofblock) + 1
        Dim midofblock As Single = CSng((endofblock + startofblock) / 2)
        Dim endoffirst As Integer = CInt(midofblock)
        Dim startofsecond As Integer = CInt(midofblock) + 1
        Dim aliceparity As Integer = 0
        Dim bobparity As Integer = 0
        If blocksize = 2 Then
            endoffirst = startofblock
            startofsecond = endofblock
        End If
        StatusTextBox.Text = "The first block is (" & startofblock.ToString & ".." & endoffirst.ToString & "), and
the second block is (" & startofsecond.ToString & ".." & endofblock.ToString & ")."
        If startofblock = CInt(midofblock) Or CInt(midofblock) = endofblock Then
            If akey(startofblock) <> bkey(startofblock) Then
                bkey(startofblock) = akey(startofblock)
                disclosedbit = disclosedbit + 2
                correctedbit = correctedbit + 1
                StatusTextBox.Text = "The key address " & startofblock.ToString & " has been reconciled."
            ElseIf akey(endofblock) <> bkey(endofblock) Then
                bkey(endofblock) = akey(endofblock)
                disclosedbit = disclosedbit + 4
                correctedbit = correctedbit + 1
                StatusTextBox.Text = "The key address " & endofblock.ToString & " has been reconciled."
            End If
        Else
            aliceparity = 0
            bobparity = 0
            For i As Integer = startofblock To endoffirst
                aliceparity = (aliceparity + akey(i)) Mod 2
```

```
            bobparity = (bobparity + bkey(i)) Mod 2
        Next i
        disclosedbit = disclosedbit + 2
        StatusTextBox.Text = "The first block:      Alice's parity is " & aliceparity.ToString & ", while Bob's
parity is " & bobparity.ToString & "."
        If aliceparity <> bobparity Then
            StatusTextBox.Text = "The first block:      Exchange the mismatched parity which correspond to the
key address (" & startofblock.ToString & ".." & endoffirst.ToString & ")."
            Reconcile(startofblock, endoffirst)
        Else
            aliceparity = 0
            bobparity = 0
            For i As Integer = startofsecond To endofblock
                aliceparity = (aliceparity + akey(i)) Mod 2
                bobparity = (bobparity + bkey(i)) Mod 2
            Next i
            disclosedbit = disclosedbit + 2
            StatusTextBox.Text = "The second block: Alice's parity is " & aliceparity.ToString & ", while Bob's
parity is " & bobparity.ToString & "."
            If aliceparity <> bobparity Then
                StatusTextBox.Text = "The second block: Exchange the mismatched parity which correspond to
the key address (" & startofsecond.ToString & ".." & endofblock.ToString & ")."
                Reconcile(startofsecond, endofblock)
            End If
        End If
    End If
  End Function
  Private Sub KLNumericUpDown_ValueChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles KLNumericUpDown.ValueChanged
    ksize = KLNumericUpDown.Value
    ReDim akey(ksize)
    ReDim bkey(ksize)
    ReDim abase(ksize)
    ReDim bbase(ksize)
```

```
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
    End Sub
    Private Sub EFNumericUpDown_ValueChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles EFNumericUpDown.ValueChanged
        percent_error = EFNumericUpDown.Value
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        ksize = KLNumericUpDown.Value
        percent_error = EFNumericUpDown.Value
        progress = 0
        ReDim akey(ksize)
        ReDim bkey(ksize)
        ReDim abase(ksize)
        ReDim bbase(ksize)
        StatusTextBox.Text = "Program is waiting for processes..."
        BStatus = StatusTextBox.Text
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
        AKeyTxt.Text = "Alice's key"
        AParityTxt.Text = "Alice's Description Panel"
    End Sub
    Private Sub KLCheckBox_CheckStateChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles KLCheckBox.CheckStateChanged
        If KLCheckBox.Checked = True Then
            KLCheckBox.Enabled = False
            ksize = RandomClass.Next(KLNumericUpDown.Minimum, KLNumericUpDown.Maximum)
            ReDim akey(ksize)
            ReDim bkey(ksize)
            ReDim abase(ksize)
            ReDim bbase(ksize)
```

```
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
        KLNumericUpDown.Value = ksize
      Else
        KLCheckBox.Enabled = True
        ksize = KLNumericUpDown.Value
        ReDim akey(ksize)
        ReDim bkey(ksize)
        ReDim abase(ksize)
        ReDim bbase(ksize)
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
      End If
    End Sub
    Private Sub EFCheckBox_CheckStateChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles EFCheckBox.CheckStateChanged
      If EFCheckBox.Checked = True Then
        EFNumericUpDown.Enabled = False
        percent_error = RandomClass.Next(EFNumericUpDown.Minimum, EFNumericUpDown.Maximum)
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
        EFNumericUpDown.Value = percent_error
      Else
        EFNumericUpDown.Enabled = True
        percent_error = EFNumericUpDown.Value
        EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
      End If
    End Sub
    Private Sub StatusBox_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles StatusTextBox.TextChanged
      EventTextBox.Text = EventTextBox.Text & "System:>  " & StatusTextBox.Text & vbCrLf
    End Sub
```

```
    Private Sub ProgressButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ProgressButton.Click
        Dim akeystr As String = ""
        Dim bkeystr As String = ""
        Dim akey2(ksize) As Integer
        Dim bkey2(ksize) As Integer
        Dim errorbit As Integer
        Dim exmbit As Integer = 0
        Dim errorfound As Integer = 0
        Dim randaddress As Integer = 0
        Dim randvalue As Integer
        Dim k0 As Integer
        If progress = 0 Then
            KLNumericUpDown.Value = ksize
            EFNumericUpDown.Value = percent_error
            ProgressButton.Text = "Next Step..."
            Bprogress = ProgressButton.Text
            StatusTextBox.Text = " Alice and Bob accept to size of keys "
            BStatus = StatusTextBox.Text
            KLNumericUpDown.Enabled = False
            EFNumericUpDown.Enabled = False
            KLCheckBox.Enabled = False
            EFCheckBox.Enabled = False
            ksize = ksize
            progress = progress + 1
        ElseIf progress = 1 Then
            ksize = ksize
            For i As Integer = 0 To ksize - 1
                akey(i) = RandomClass.Next(2)
            Next i
            For i As Integer = 0 To ksize - 1
                If i <> 0 And i Mod 40 = 0 Then
                    akeystr = akeystr
                End If
```

```
        Next i
        For i As Integer = 0 To ksize - 1
          abase(i) = RandomClass.Next(2)
        Next i
        For i As Integer = 0 To ksize - 1
          If i <> 0 And i Mod 40 = 0 Then
            akeystr = akeystr
          End If
          If abase(i) = 0 Then
            akeystr = akeystr & "+"
          Else
            akeystr = akeystr & "x"
          End If
          akeystr = akeystr & akey(i)
        Next i
        StatusTextBox.Text = "Alice generated the key of " & ksize.ToString & " bits, Alice is sending the key
via quantum channel."
        BStatus = StatusTextBox.Text
        AKeyTxt.Text = akeystr
        AParityTxt.Text = "+0 means a polarisation photon in [0 degree]." & "                                    "
& "+1 means a polarisation photon in [90 degree]." & "                                    " & "x0 means a
polarisation photon in [-45 degree]." & "                                    " & "x1 means a polarisation photon in
[45 degree]."
        progress = progress + 1
      ElseIf progress = 3 Then
        ksize = KLNumericUpDown.Value
        For i As Integer = 0 To ksize - 1
          randvalue = RandomClass.Next(101)
          If randvalue < percent_error Then
            bkey(i) = (akey(i) + 1) Mod 2
          ElseIf randvalue > 90 Then
            bkey(i) = 2
            errorbit = errorbit + 1
```

```
            Else
              bkey(i) = akey(i)
            End If
          Next i
          For i As Integer = 0 To ksize - 1
            If i <> 0 And i Mod 40 = 0 Then
              akeystr = akeystr
              bkeystr = bkeystr
            End If
            If bkey(i) = 0 Or bkey(i) = 1 Then
              If bbase(i) = 0 Then
                akeystr = akeystr & akey(i).ToString
                bkeystr = bkeystr & bkey(i).ToString
              Else
                akeystr = akeystr & akey(i).ToString
                bkeystr = bkeystr & bkey(i).ToString
              End If
            ElseIf bkey(i) = 2 Then
              If bbase(i) = 0 Then
                akeystr = akeystr & "_" & akey(i).ToString
                bkeystr = bkeystr & "1"
              Else
                akeystr = akeystr & "_" & akey(i).ToString
                bkeystr = bkeystr & "0"
              End If
            End If
          Next i
          ksize = ksize
          xkeystr = bkeystr
          StatusTextBox.Text = "Bob received the key of " & ksize.ToString & " qubits and measured them (" &
errorbit.ToString & " qubits were undetectable)."
          BStatus = StatusTextBox.Text
        ElseIf progress = 4 Then
          ksize = ksize
```

```
For i As Integer = 0 To ksize - 1
  If bkey(i) = 2 Then
    akey(i) = 2
  End If
Next i
For i As Integer = 0 To ksize - 1
  If abase(i) <> bbase(i) Then
    akey(i) = 3
    bkey(i) = 3
  End If
  akey2(i) = akey(i)
  bkey2(i) = bkey(i)
Next i
Dim j As Integer = 0
Dim k As Integer = 0
Do While j < ksize
  If bkey2(j) = 2 Or bkey2(j) = 3 Then
    j = j + 1
    Continue Do
  Else
    akey(k) = akey2(j)
    bkey(k) = bkey2(j)
  End If
  j = j + 1
  k = k + 1
Loop
ksize = k
For i As Integer = 0 To ksize - 1
  If i <> 0 And i Mod 40 = 0 Then
    akeystr = akeystr
    bkeystr = bkeystr
  End If
  akeystr = akeystr & akey(i).ToString
  bkeystr = bkeystr & bkey(i).ToString
```

```
        Next i
        AKeyTxt.Text = akeystr
        xkeystr = bkeystr
        StatusTextBox.Text = "Bob kept only " & ksize.ToString & " bits that are detected and matched
position (some bits still error)."
        BStatus = StatusTextBox.Text
        AParityTxt.Text = "Prepareing to cascade reconciliation."
    ElseIf progress = 5 Then
        For i As Integer = 0 To ksize - 1
          akey2(i) = akey(i)
          bkey2(i) = bkey(i)
        Next
        exmbit = CInt((20 * ksize) / 100)
        For i As Integer = 0 To exmbit - 1
          randaddress = RandomClass.Next(ksize)
          If akey2(randaddress) <> bkey2(randaddress) Then
            akey2(randaddress) = 2
            bkey2(randaddress) = 2
            errorfound = errorfound + 1
          Else
            akey2(randaddress) = 2
            bkey2(randaddress) = 2
          End If
        Next i
        kerror = CDbl(errorfound / exmbit)
        For i As Integer = 0 To ksize - 1
          If i <> 0 And i Mod 40 = 0 Then
            akeystr = akeystr
            bkeystr = bkeystr
          End If
          If bkey2(i) = 0 Or bkey2(i) = 1 Then
            akeystr = akeystr & akey(i).ToString
            bkeystr = bkeystr & bkey(i).ToString
          ElseIf bkey2(i) = 2 Then
```

```
            akeystr = akeystr & akey(i).ToString
            bkeystr = bkeystr & bkey(i).ToString
          End If
        Next i
        ksize = ksize - exmbit
        AKeyTxt.Text = akeystr
        xkeystr = bkeystr
        StatusTextBox.Text = "Each side, " & exmbit.ToString & " bits are randomly sampled and " &
errorfound.ToString & " errors are detected."
        BStatus = StatusTextBox.Text
        StatusTextBox.Text = "Calculated error estimate is " & (kerror * 100).ToString & " %."
        BStatus = StatusTextBox.Text
        ksize = ksize + exmbit
        Dim j As Integer = 0
        Dim k As Integer = 0
        Do While j < ksize
          If bkey2(j) = 2 Or bkey2(j) = 3 Then
            j = j + 1
            Continue Do
          Else
            akey(k) = akey2(j)
            bkey(k) = bkey2(j)
          End If
          j = j + 1
          k = k + 1
        Loop
        ksize = ksize - exmbit
      ElseIf progress = 6 Then
        k0 = 0
        pass = 1
        If kerror = 0 Then
          k0 = ksize / 20
        Else
          k0 = CInt((0.5 * ((1 / (kerror)) + (1 / (4 * kerror)))))
```

```
        End If
        inbsize = k0
        Do
          pass = pass + 1
          k0 = CInt(k0 * 2)
        Loop While CInt(k0) < CInt((ksize / 4))
        pass = pass + 3
        For i As Integer = 0 To ksize - 1
          If i <> 0 And i Mod 40 = 0 Then
            akeystr = akeystr
            bkeystr = bkeystr
          End If
          akeystr = akeystr & akey(i).ToString
          bkeystr = bkeystr & bkey(i).ToString
        Next i
        AKeyTxt.Text = akeystr
        xkeystr = bkeystr
        StatusTextBox.Text = "Key Length of " & ksize.ToString & " remains at this step."
        BStatus = StatusTextBox.Text
        StatusTextBox.Text = "A calculated initial block size is " & inbsize.ToString & " bits with " &
pass.ToString & " passes."
        BStatus = StatusTextBox.Text
      ElseIf progress = 7 Then
        Dim alieeparity As Integer = 0
        Dim bobparity As Integer = 0
        Dim startofblock As Integer = 0
        Dim endofblock As Integer = 0
        Dim temp As Integer = 0
        Dim _aparity(ksize) As Integer
        Dim _bparity(ksize) As Integer
        Dim k As Integer = 0
        Dim aparstr As String = ""
        Dim bparstr As String = ""
        For i As Integer = 0 To pass - 1
```

```
aliceparity = 0
bobparity = 0
startofblock = 0
endofblock = startofblock + inbsize - 1
disclosedbit = 0
correctedbit = 0
k = 0
StatusTextBox.Text = "Start cascade reconciliation pass " & (i + 1).ToString & " with block size of "
& inbsize.ToString & "."
For j As Integer = 0 To ksize - 1
  _aparity(j) = 0
  _bparity(j) = 0
Next j
If pass - i <= 2 Then
  StatusTextBox.Text = "Shuffled key..."
  randvalue = RandomClass.Next(ksize)
  For j As Integer = 0 To ksize - 1
    temp = akey(randvalue)
    akey(randvalue) = akey(j)
    akey(j) = temp
    temp = bkey(randvalue)
    bkey(randvalue) = bkey(j)
    bkey(j) = temp
  Next j
End If
akeystr = ""
bkeystr = ""
For j As Integer = 0 To ksize - 1
  If j <> 0 And j Mod 40 = 0 Then
    akeystr = akeystr
    bkeystr = bkeystr
  End If
  akeystr = akeystr & akey(j)
  bkeystr = bkeystr & bkey(j)
```

```
Next j
AKeyTxt.Text = akeystr
xkeystr = bkeystr
While startofblock < ksize
  aliceparity = 0
  bobparity = 0
  For j As Integer = startofblock To endofblock
    aliceparity = (aliceparity + akey(j)) Mod 2
    bobparity = (bobparity + bkey(j)) Mod 2
  Next j
  _aparity(k) = aliceparity
  _bparity(k) = bobparity
  disclosedbit = disclosedbit + 2
  startofblock = startofblock + inbsize
  endofblock = endofblock + inbsize
  If endofblock > ksize - 1 Then
    endofblock = ksize - 1
  End If
  k = k + 1
End While
aparstr = ""
bparstr = ""
For j As Integer = 0 To k - 1
  If j <> 0 And j Mod 40 = 0 Then
    aparstr = aparstr
    bparstr = bparstr
  End If
  If _aparity(j) <> _bparity(j) Then
    aparstr = aparstr & "_" & _aparity(j)
    bparstr = bparstr & "_" & _bparity(j)
  Else
    aparstr = aparstr & _aparity(j)
    bparstr = bparstr & _bparity(j)
  End If
```

```
            Next j
            AParityTxt.Text = aparstr
            aparstr = aparstr.Replace("_", "")
            bparstr = bparstr.Replace("_", "")
            StatusTextBox.Text = "Alice's parity is " & aparstr.ToString & "."
            StatusTextBox.Text = "Bob's parity is  " & bparstr.ToString & "."
            For j As Integer = 0 To k - 1
              If _aparity(j) <> _bparity(j) Then
                startofblock = j * inbsize
                endofblock = ((j + 1) * inbsize) - 1
                StatusTextBox.Text = "The detected parity address " & j.ToString & " is mismatched."
                If endofblock < ksize - 1 Then
                  StatusTextBox.Text = "Exchange the mismatched parity which correspond to the key address
(" & startofblock.ToString & ".." & endofblock.ToString & ")."
                  Reconcile(startofblock, endofblock)
                Else
                  StatusTextBox.Text = "Exchange the mismatched parity which correspond to the key address
(" & startofblock.ToString & ".." & (ksize - 1).ToString & ")."
                  Reconcile(startofblock, ksize - 1)
                End If
              End If
            Next j
            StatusTextBox.Text = disclosedbit.ToString & " bits were disclosed after pass " & (i + 1).ToString &
" and reconciled " & correctedbit.ToString & " bits."
            StatusTextBox.Text = "Cascade reconciliation pass " & (i + 1).ToString & " with block size of " &
inbsize.ToString & " was completed."
            MessageBox.Show(disclosedbit.ToString & " bits were disclosed after pass " & (i + 1).ToString & "
and reconciled " & correctedbit.ToString & " bits.", "cascade reconciliation progress",
MessageBoxButtons.OK, MessageBoxIcon.Information, MessageBoxDefaultButton.Button1)
            If pass - i > 3 Then
              inbsize = inbsize * 2
            End If
          Next i
```

```
    StatusTextBox.Text = "Cascade reconciliation was completed."
    BStatus = StatusTextBox.Text
    AParityTxt.Text = "Alice Description Panel"
  ElseIf progress = 8 Then
    For i As Integer = 0 To ksize - 1
      If i <> 0 And i Mod 40 = 0 Then
        akeystr = akeystr
        bkeystr = bkeystr
      End If
    Next i
    For i As Integer = 0 To ksize - 1
      For j As Integer = i + 1 To ksize

        If akey(i) = 0 And akey(j) = 0 Then
          akey2(i) = 0
        ElseIf akey(i) = 1 And akey(j) = 1 Then
          akey2(i) = 0
        Else
          akey2(i) = 1
        End If
        If bkey(i) = 0 And bkey(j) = 0 Then
          bkey2(i) = 0
        ElseIf bkey(i) = 1 And bkey(j) = 1 Then
          bkey2(i) = 0
        Else
          bkey2(i) = 1
        End If
      Next
      akeystr = akeystr & akey2(i)
      bkeystr = bkeystr & bkey2(i)
      i = i + 1
    Next
    AKeyTxt.Text = akeystr
    xkeystr = bkeystr
```

```
            StatusTextBox.Text = "All operations were completed successfully."
            StatusTextBox.Text = "The final key is ready to use."
            AParityTxt.Text = " "
            ProgressButton.Text = "Restart"
            progress = progress + 1
        ElseIf progress = 9 Then
            ProgressButton.Text = "Start"
            KLNumericUpDown.Enabled = True
            EFNumericUpDown.Enabled = True
            KLCheckBox.Enabled = True
            EFCheckBox.Enabled = True
            KLNumericUpDown.Value = 1000
            EFNumericUpDown.Value = 10
            ALabel.Text = "[Key Length: 0]"
            Form1_Load(sender, e)
            progress = 0
        End If
    End Sub
    Private Sub AKeyTxt_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles AKeyTxt.TextChanged
        If progress > 0 Then
            ALabel.Text = "[Key Length: " & ksize.ToString & "]"
        End If
    End Sub
    Private Sub btnCall_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnCall.Click
        If TextBox1.TextLength = 0 Then
            MessageBox.Show(" กรุณากรอกชื่อบุคคลที่ต้องการจะติดต่อด้วยครับ ", "ต้องการติดต่อกับใคร",
MessageBoxButtons.OK, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1)
        Else
            BName = TextBox1.Text
            Form2.Show()
            TextBox1.Enabled = False
            TxtName.Enabled = False
```

```
        btnCall.Enabled = False
      End If
    End Sub
    Private Sub EventTextBox_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles EventTextBox.TextChanged
      BEven = EventTextBox.Text
    End Sub
    Private Sub btnCancle_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnCancle.Click
      Me.Close()
    End Sub
End Class
- Bob's code  Form
Public Class Form2
    Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      StatusTextBox.Text = "Program is waiting for processes..."
      EventTextBox.Text = "System:>  Key Length and Error Factor is set to " & ksize.ToString & " and " &
percent_error.ToString & ", respectively." & vbCrLf
      NamLabel.Text = " BName: " & CStr(BName)
      BLabel.Text = "[Key Length: 0]"
      BKeyTxt.Text = "Bob's key"
    End Sub
    Private Sub ProgressButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ProgressButton.Click
      If progress = 0 Then
        BLabel.Text = "[Key Length: 0]"
        EventTextBox.Text = CStr(BEven)
        BKeyTxt.Text = "Bob's key"
        ProgressButton.Text = "Start"
        StatusTextBox.Text = CStr(BStatus)
      ElseIf progress = 1 Then
        xkeystr = ""
        BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
```

```
        EventTextBox.Text = CStr(BEven)
        BKeyTxt.Text = " "
        ProgressButton.Text = CStr(Bprogress)
        StatusTextBox.Text = CStr(BStatus)
    ElseIf progress = 2 Then
        ksize = ksize
        ReDim bbase(ksize)
        For i As Integer = 0 To ksize - 1
            bbase(i) = RandomClass.Next(2)
        Next i
        For i As Integer = 0 To ksize - 1
            If i <> 0 And i Mod 40 = 0 Then
                xkeystr = xkeystr
            End If
            If bbase(i) = 0 Then
                xkeystr = xkeystr & "+"
            Else
                xkeystr = xkeystr & "x"
            End If
        Next i
        BKeyTxt.Text = xkeystr
        StatusTextBox.Text = "After generated " & ksize.ToString & " bases, Bob is waiting for the key from
Alice.By assume that + means a qubit in basis [+]." & "x means a qubit in basis [x]."
        BStatus = StatusTextBox.Text
        progress = progress + 1
    ElseIf progress = 3 Then
        BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
        EventTextBox.Text = CStr(BEven)
        BKeyTxt.Text = CStr(xkeystr)
        ProgressButton.Text = CStr(Bprogress)
        StatusTextBox.Text = CStr(BStatus)
        progress = progress + 1
    ElseIf progress = 4 Then
        BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
```

```
    EventTextBox.Text = CStr(BEven)
    BKeyTxt.Text = CStr(xkeystr)
    ProgressButton.Text = CStr(Bprogress)
    StatusTextBox.Text = CStr(BStatus)
    progress = progress + 1
  ElseIf progress = 5 Then
    BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
    EventTextBox.Text = CStr(BEven)
    BKeyTxt.Text = CStr(xkeystr)
    ProgressButton.Text = CStr(Bprogress)
    StatusTextBox.Text = CStr(BStatus)
    progress = progress + 1
  ElseIf progress = 6 Then
    BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
    EventTextBox.Text = CStr(BEven)
    BKeyTxt.Text = CStr(xkeystr)
    ProgressButton.Text = CStr(Bprogress)
    StatusTextBox.Text = CStr(BStatus)
    progress = progress + 1
  ElseIf progress = 7 Then
    BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
    EventTextBox.Text = CStr(BEven)
    BKeyTxt.Text = CStr(xkeystr)
    ProgressButton.Text = CStr(Bprogress)
    StatusTextBox.Text = CStr(BStatus)
    progress = progress + 1
  ElseIf progress = 8 Then
    BLabel.Text = "[Key Length: " & CStr(ksize) & "]"
    EventTextBox.Text = CStr(BEven)
    BKeyTxt.Text = CStr(xkeystr)
    ProgressButton.Text = CStr(Bprogress)
    StatusTextBox.Text = CStr(BStatus)
  ElseIf progress = 9 Then
    BKeyTxt.Text = CStr(xkeystr)
```

```
            ProgressButton.Text = "Restart"
            StatusTextBox.Text = "The final key is ready to use."
        End If
    End Sub
End Class
```

- code Module

```
Module Module1
    Public akey(10000) As Integer
    Public bkey(10000) As Integer
    Public abase(10000) As Integer
    Public bbase(10000) As Integer
    Public BName As String
    Public BEven As String
    Public BStatus As String
    Public Bprogress As String
    Public xkeystr As String
    Public randvalue As Integer
    Public ksize As Integer
    Public percent_error As Integer
    Public kerror As Single
    Public inbsize As Integer
    Public pass As Integer
    Public disclosedbit As Integer
    Public correctedbit As Integer
    Public progress As Integer
    Public RandomClass As New Random()
End Module
```