การจัดเก็บข้อมูลไบนารีในฐานข้อมูลในรูปแบบ Base128

จรรยาภรณ์ ประสมสัตย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม คณะวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
พ.ศ. 2557

Storing Binary Data to Database using Base128 encoding

Chanyaphon Prosomsat

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering

Department of Computer and Telecommunication Engineering

Faculty of Engineering, Dhurakij Pundit University

2014

หัวข้อวิทยานิพนธ์ การจัดเก็บข้อมูลไบนารีในฐานข้อมูลในรูปแบบ Base128
ชื่อผู้เขียน จรรยาภรณ์ ประสมสัตย์
อาจารย์ที่ปรึกษา อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์
สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม
ปีการศึกษา 2557

บทคัดย่อ

งานวิจัยนี้พัฒนาวิธีการจัดเก็บข้อมูลสตริง ให้มีประสิทธิภาพมากขึ้น โดยวิธีที่พัฒนามีการใช้พื้นที่ในการจัดเก็บข้อมูลลดลงด้วยการแปลงเป็นข้อมูลแบบ Base128 ซึ่งจะช่วยลดความเสี่ยงที่อาจทำให้ฐานข้อมูลเสียหาย และง่ายต่อการค้นหาข้อมูล โดยการใช้เทคนิคการแปลงข้อมูลไบนารีให้เป็นข้อมูลสตริง ด้วยอินเด็กซ์ของข้อมูล แต่ยังคงใช้เวลาประมวลทั้งหมดมากกว่า BLOB อยู่ประมาณร้อยละ 80 โดยทำการทดสอบด้วยการอ่านและเขียนไฟล์ต่างๆ ลงฐานข้อมูล MySQL เปรียบเทียบกับวิธีการต่างๆ ได้แก่ BLOB, Base32, Base64 และ Base128 รวมทั้งทดสอบประสิทธิภาพการเข้ารหัสแบบ AES กับวิธีการต่างๆ ด้วย

ผลการทดสอบพบว่า การจัดเก็บข้อมูลแบบ Base128 ใช้พื้นที่การจัดเก็บข้อมูลใกล้เคียงกับการจัดเก็บข้อมูลแบบ BLOB ในขณะที่ใช้เวลาในการแปลงบันทึกและการคืนกลับข้อมูลน้อยที่สุดเมื่อเทียบกับการจัดเก็บข้อมูลแบบ Base64 และ Base32

Thesis Title	Storing Binary Data to Database using Base128 encoding
Author	Chanyaphon Prosomsat
Thesis Advisor	Chiyaporn Khemapatapan, Ph.D
Department	Computer and Telecommunication Engineering
Academic Year	2014

ABSTRACT

This research aims to develop an efficiency method of storing binary data which is developed on Web browser application with database system. The proposed method aims to reduce storage size by converting binary data into Base128 format. Storing Base128 string data in database will reduce the risk of database damage and help to find information in database. The conversion will use a binary value to index Base128 string. The testing results between various encoding: BLOB, Base32, Base64 and Base128 are compared.

The results show that storing data using Base128 consumes storage size nearby storing data using BLOB. However, Base128 encoding will consume time to convert, store and restore less than Base64 and Base32 encoding.

กิตติกรรมประกาศ

การจัดทำวิทยานิพนธ์ฉบับนี้สำเร็จสมบูรณ์ลงได้ ด้วยความเมตตากรุณาจาก อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ข้อคิดเห็น คำแนะนำที่เป็นประโยชน์ต่องานวิจัย ที่สละเวลาอันมีค่า คอยให้คำแนะนำให้คำปรึกษา ตลอดจนแนวทางในการแก้ไขปัญหาต่างๆ และเอาใจใส่ข้าพเจ้ามาโดยตลอด ข้าพเจ้ารู้สึกซาบซึ้งเป็นอย่างยิ่ง จึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

ขอขอบพระคุณกรรมการสอบ วิทยานิพนธ์ ซึ่งสละเวลาเพื่อเป็นกรรมการสอบวิทยานิพนธ์ และให้คำแนะนำที่เป็นประโยชน์ต่องานวิจัย และขอบคุณเจ้าหน้าที่ทุกท่านที่ช่วยดำเนินเรื่องต่างๆ ให้เป็นอย่างดี

ขอขอบพระคุณคณาจารย์ทุกท่านที่ได้ประสิทธิประสาทวิชาความรู้ จนข้าพเจ้าประสบความสำเร็จในการศึกษา ขอขอบพระคุณเพื่อนร่วมรุ่น พี่ๆ น้องๆ ทุกๆ คน รวมถึงคณะเจ้าหน้าที่ประจำหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต และคณะเจ้าหน้าที่บัณฑิตวิทยาลัย ทุกท่าน ซึ่งไม่อาจกล่าวนามได้ทั้งหมดในที่นี้ ที่ได้ให้กำลังใจและช่วยเหลือข้าพเจ้ามาโดยตลอด

ท้ายสุดนี้ คุณความดีและกุศลที่พึงบังเกิดมีจากการจัดทำวิทยานิพนธ์ของข้าพเจ้า ซึ่งสามารถก่อให้เกิดความรู้และข้อคิดอันควรค่าแก่การศึกษา หรือปฏิบัติให้เกิดประโยชน์ต่อส่วนรวม ข้าพเจ้าขอมอบระลึกบูชาคุณแด่ บิดา มารดา ครู อาจารย์ ผู้มีพระคุณ ตลอดจน ผู้แต่งหนังสือหรือตำราทุกท่าน ที่ข้าพเจ้าใช้อ้างอิงในวิทยานิพนธ์ฉบับนี้ ข้าพเจ้ามีความซาบซึ้งในความกรุณาอันดียิ่งจากทุกท่าน และขอกราบขอบพระคุณมา ณ โอกาสนี้ หากมีข้อบกพร่องประการใด ข้าพเจ้าขอน้อมรับไว้แต่เพียงผู้เดียว

จรรยาภรณ์ ประสมสัตย์

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ที่มาของงานวิจัย

ปัจจุบันการใช้งานคอมพิวเตอร์และการใช้บริการอินเตอร์เน็ต(Internet)มีจำนวนมากขึ้นอย่างรวดเร็ว ทำให้ข้อมูลที่อยู่ในระบบมีมากขึ้นส่งผลให้การบันทึกหรือจัดเก็บข้อมูลจากสื่อต่างๆ ต้องใช้ปริมาณความจุข้อมูลเพิ่มขึ้นตามไปด้วย แต่การจะเพิ่มหรือพัฒนาอุปกรณ์บันทึกข้อมูลให้มีขนาดความจุสูงขึ้นนั้นทำได้ไม่ง่ายนัก เนื่องจากต้องใช้การลงทุนสูง และการค้นคว้าวิจัยด้านอุปกรณ์การจัดเก็บข้อมูลนั้นกระทำได้ยาก กอปรกับข้อมูลในฐานข้อมูลเป็นสิ่งที่สำคัญมาก หากเกิดความเสียหายหรือข้อผิดพลาดใดๆ ขึ้น จะทำให้ส่งผลเสียใหญ่หลวงต่อระบบได้ ซึ่งการจัดเก็บข้อมูลในปริมาณมากนั้น สามารถส่งผลทำให้เกิดผลทำให้เกิดความเสียหายต่อฐานข้อมูลนั้นๆ ได้

ที่ผ่านมาข้อมูลในฐานข้อมูลส่วนใหญ่ สามารถแบ่งประเภทได้เป็น 3 ประเภท คือ ข้อมูลประเภทไฟล์ซึ่งอยู่ในรูปแบบของข้อมูลไบนารี ข้อมูลประเภทข้อความ และข้อมูลประเภทตัวเลข ซึ่งข้อมูลประเภทไบนารีนั้น เป็นข้อมูลที่มีความยุ่งยากในการจัดเก็บข้อมูลมากที่สุดและสามารถส่งผลให้ฐานข้อมูลเกิดความเสียหายต่อการจัดเก็บในระบบฐานข้อมูลได้ โดยปกติสามารถจัดเก็บข้อมูลไบนารีในรูปแบบของการเก็บที่อยู่ไฟล์ (Path) ของไฟล์ข้อมูล และทำการจัดเก็บข้อมูลจริงไว้ในรูปแบบของไฟล์แทนโดยแยกเก็บไว้ที่เซิร์ฟเวอร์ วิธีการนี้เรียกวิธีการเก็บแบบพาท ส่วนวิธีการนี้มีข้อด้อย คือ การบริหารจัดการข้อมูลเพิ่มขึ้นเนื่องจากข้อมูลไม่ได้เก็บเป็นก้อนเดียวกันต้องแยกข้อมูลไฟล์เก็บไว้บนเซิร์ฟเวอร์ นอกจากนี้ยังสามารถจัดเก็บในรูปแบบของข้อมูลไบนารีขนาดใหญ่ (BLOB : binary large objects) ก็ได้ คือ การเก็บข้อมูลไบนารีลงฐานข้อมูลโดยตรง ซึ่งช่วยให้การจัดการกับข้อมูลเบ็ดเสร็จในระบบฐานข้อมูล อย่างไรก็ตามการจัดเก็บข้อมูลแบบ BLOB ยังมีความเสี่ยงของความเสียหาย ของระบบฐานข้อมูลได้ เนื่องจากเป็นการจัดเก็บข้อมูลไบนารีลงฐานข้อมูลโดยตรง โดยมิได้ทำการแยกบันทึกข้อมูลที่เซิร์ฟเวอร์ (Server) ซึ่งอาจมีรหัสแอสกีหรือรหัสมาตรฐานของสหรัฐอเมริกาเพื่อการแลกเปลี่ยนสารสนเทศ (ASCII : American Standard Code for Information Interchange) ที่ทำให้ระบบฐานข้อมูลทำงานผิดพลาดได้ เช่น รหัสแอสกีที่ถูกกำหนดให้เป็นอักขระควบคุม ซึ่งส่งผลทำให้ข้อมูลเสียหายโดยไม่สามารถอ่านไฟล์กลับคืนมาได้ อีกทั้งยังมีข้อจำกัดในเรื่องของขนาดไฟล์ที่จะบันทึกลงฐานข้อมูลได้เนื่องจากแต่ละฐานข้อมูลมีข้อจำกัดในการเก็บข้อมูลประเภทไบนารี เช่น ฐานข้อมูล MySQL มีข้อจำกัดข้อมูลประเภทไบนารีประเภท BLOB ที่ขนาด 64 kB

ส่วนอีกหนึ่งวิธีการเก็บข้อมูลไบนารีแบบตัวอักษร(String) หรือข้อมูลไบนารีประเภทข้อความเป็นการเก็บข้อมูลไบนารีที่ผ่านการเข้ารหัสเพื่อแปลงเป็นข้อมูลประเภทตัวอักษรมีข้อดีคือสามารถจัดเก็บข้อมูลได้ไม่จำกับขนาดและสามารถจะทำการค้นหาข้อมูลในข้อมูลประเภทไบนารีที่อยู่ในรูปแบบข้อมูลตัวอักษรได้ ข้อเสียคือการจัดเก็บมีผลให้ขนาดการจัดเก็บข้อมูลเพิ่มขึ้น ซึ่งวิธีนี้ไม่เป็นที่นิยมใช้ในปัจจุบันอันเนื่องมาจากสาเหตุที่กล่าวมาข้างต้น แต่การมีข้อเสียในส่วนนี้นั้นก็สามารถทำให้มองในแง่มุมที่ดีได้ คือสามารถนำมาปรับปรุงแก้ไข พัฒนาต่อยอดให้ดีขึ้นหรือประยุกต์ใช้ข้อดีที่มีอยู่โดยนำไปใช้งานกับส่วนอื่นต่อไปได้ เช่น การค้นหาข้อความในข้อมูลได้

จากเหตุผลข้างต้นที่กล่าวมาจึงทำการพัฒนาปรับปรุงแก้ไขการแปลงข้อมูลไบนารีโดยใช้หลักการแปลงข้อมูลไบนารีให้อยู่ในรูปแบบตัวอักษร ผ่านกระบวนการเข้ารหัสข้อมูลแบบไบนารีแปลงเป็นข้อมูลประเภทตัวอักษร โดยใช้วิธีการสร้างจาวาคลาสขึ้นมาเรียกว่า Base128 เพื่อลดขนาดของข้อมูลไบนารีให้ได้ใกล้เคียงขนาดจริงให้มากที่สุด โดยมีแนวทางงานวิจัยที่พัฒนามาจากการเข้ารหัสข้อมูลแบบ Base64 ซึ่งเป็นมาตรฐานของการเข้ารหัสข้อมูลไบนารีเป็นข้อมูลสตริงที่ใช้งานกันทั่วไป

1.2 วัตถุประสงค์ของงานวิจัย

1. เพื่อออกแบบและพัฒนาวิธีการจัดเก็บข้อมูลไบนารีให้มีประสิทธิภาพมากขึ้น
2. เพื่อการจัดเก็บข้อมูลลดลงด้วยการแปลงเป็นข้อมูลแบบ Base128 ที่อยู่ในรูปของสตริง
3. เพื่อเปรียบเทียบระหว่างการจัดเก็บข้อมูลในรูปแบบ BLOB Base32, Base64 และ Base128

1.3 ขอบเขตของงานวิจัย

ขอบเขตของวิทยานิพนธ์ฉบับนี้ มีขอบเขตของงานวิจัยดังนี้

1. วิธีการที่พัฒนาสามารถใช้เทคนิคการแปลงข้อมูลไบนารีให้เป็นข้อมูลสตริงได้
2. วิธีการที่พัฒนาสามารถวัดประสิทธิภาพโดยเปรียบเทียบระหว่างการจัดเก็บข้อมูลในรูปแบบ BLOB, Base32, Base64 และ Base128
3. วิธีการที่พัฒนาสามารถลดการจัดเก็บข้อมูลให้ใกล้เคียงกับการจัดเก็บข้อมูลแบบ BLOB เมื่อเปรียบเทียบระหว่าง Base32, Base64 และ Base128
4. กระบวนการในการพัฒนาเป็นที่ยอมรับได้ในเชิงวิศวกรรม

1.4 เครื่องมือที่ใช้พัฒนางานวิจัย

1. เครื่องคอมพิวเตอร์ CPU 2.8 GHz, Memory 2 GB
2. MySQL
3. Eclipse
4. JAVA Language (JSP)
5. Tomcat Server

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้สร้างวิธีการจัดเก็บข้อมูลไบนารีให้อยู่ในรูปแบบสตริงในรูปแบบ Base128
2. ได้ทราบถึงความแตกต่างระหว่างการจัดเก็บข้อมูลในรูปแบบ BLOB Base32 Base64 และ Base128
3. ได้ทราบถึงวิธีการลดปริมาณการจัดเก็บข้อมูลให้ให้ใกล้เคียงกับการจัดเก็บข้อมูลแบบ BLOB
4. นำวิธีการที่ไม่นิยมใช้ในการการบันทึกข้อมูลเพื่อนำกลับมาใช้ใหม่เพื่อจะได้เป็นที่นิยมเพิ่มขึ้น
5. ลดปัญหาข้อมูลขนาดใหญ่ (Big data problem) ที่กำลังจะเกิดขึ้นในอนาคตอันใกล้นี้

บทที่ 2
แนวคิด ทฤษฎี และผลงานวิจัยที่เกี่ยวข้อง

2.1 การบันทึกข้อมูลไบนารี

งานวิจัยในนี้ มีวัตถุประสงค์ที่จะเข้ารหัสข้อมูลชนิดไฟล์ให้อยู่ในไฟล์แบบสตริงก่อนเก็บลงฐานข้อมูล โดยมุ่งเน้นการลดขนาดของข้อมูลไบนารี เมื่อบันทึกข้อมูลลงฐานข้อมูลให้ได้ขนาดใกล้เคียงกับขนาดข้อมูลจริงให้มากที่สุด เมื่อเปรียบเทียบกับการเข้ารหัสข้อมูลแบบ Base32 และ Base64 ซึ่งก่อให้เกิดผลคือ ขนาดของข้อมูลไบนารีมีขนาดเล็กลง การจัดเก็บข้อมูลมีความเร็วและยืดหยุ่นขึ้น และสามารถค้นหาข้อมูลไบนารีต่างๆโดยค้นหาจากข้อความที่เป็นเนื้อหาของไฟล์ ข้อมูลนั้นๆ แทนการค้นหาภาพโดยใช้อินเด็กซ์ (Index) ซึ่งทำให้มีความสะดวกและรวดเร็วยิ่งขึ้น ซึ่งเทคนิคที่ใช้ในงานวิจัยนี้คือ การแทนข้อมูลไบนารีที่เป็นชนิดไบต์ (Byte) เป็นชนิดสตริงมีข้อดีคือ การจัดเก็บข้อมูลมีความเร็วในการจัดเก็บข้อมูลมากขึ้น ขนาดของข้อมูลใกล้เคียงกับข้อมูลจริง ความเร็วในการแสดงภาพมีความเร็วมากขึ้น และสามารถค้นหาข้อมูลไบนารีด้วยข้อความ แต่วิธีการนี้จะมีข้อเสียบางประการนั่นคือขนาดข้อมูลจะมีขนาดใหญ่ขึ้นกว่าขนาดข้อมูลจริง เหมาะกับการใช้งานในด้านข้อมูลที่เก็บเป็นไฟล์ไม่ว่าจะเป็นข้อมูลชนิด .png .jpg .doc .xls .xlsx .pdf และอื่นๆ เป็นต้น ต่อไปเป็นความรู้เบื้องต้นเกี่ยวกับข้อมูล

2.1.1 หน่วยข้อมูล (DATA UNITS)

บิต (bit) เลขฐานสองหนึ่งหลักซึ่งมีค่าเป็น 0 หรือ 1

ตัวอักษร (character) กลุ่มของบิตสามารถแทนค่าตัวอักษรได้ ในชุดอักขระ ASCII 1 ไบต์ (8 บิต) แทนตัวอักษร 1 ตัว

เขตข้อมูล หรือฟิลด์ (field) เขตข้อมูลซึ่งประกอบด้วยกลุ่มตัวอักษรที่แทนข้อเท็จจริง

ระเบียน (record) ระเบียน คือโครงสร้างข้อมูลที่แทนตัววัตถุชิ้นหนึ่ง

แฟ้ม (file) ตารางที่เป็นกลุ่มของระเบียนที่มีโครงสร้างเดียวกัน

ฐานข้อมูล (database) กลุ่มของตาราง (และความสัมพันธ์)

2.1.2 ชนิดของข้อมูล (DATA TYPES)

ค่าตรรกะ (Boolean values) ซึ่งมีเพียงสองค่าคือ จริง กับ เท็จ

จำนวนเต็ม (integers) หมายถึง เลขที่ไม่มีเศษส่วน หรือทศนิยม

จำนวนจริง (floating-point numbers) หมายถึง จำนวนใดๆ ทั้งจำนวนเต็มและจำนวนทศนิยม

ตัวอักษร (characters) หมายถึง ข้อมูลประเภทตัวอักษรเพียงตัวเดียว

สายอักขระ (strings) หมายถึง กลุ่มตัวอักษรที่ประกอบกันขึ้นเป็นข้อความ

วันที่และเวลา (date/time) หมายถึง ข้อมูลที่แทนค่าวันที่และเวลา

ไบนารี (binary) หมายถึง ข้อมูลที่เก็บในคอมพิวเตอร์ อาจเป็นแฟ้มโปรแกรม ภาพ หรือวิดีโอ

2.2 ตารางอักขระ ASCII

ตัวเลขเป็นภาษาของคอมพิวเตอร์ เมื่อต้องการติดต่อสื่อสารกับโปรแกรม (และกับคอมพิวเตอร์เครื่องอื่น) คอมพิวเตอร์จะแปลงอักขระและสัญลักษณ์เป็นตัวแทนสัญลักษณ์และอักขระที่เป็นตัวเลข

ในช่วงทศวรรษ 1960 ความต้องการที่จะทำให้การสื่อสารดังกล่าวเป็นมาตรฐานจึงทำให้เกิดโค้ดที่เรียกว่า American Standard Code for Information Interchange (ASCII) (อ่านว่าแอสกี) โดยตาราง ASCII ประกอบด้วยตัวเลข 128 ตัวซึ่งจะใช้แทนอักขระ ทั้งนี้ ASCII ให้วิธีที่คอมพิวเตอร์สามารถเก็บและแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์เครื่องอื่นและโปรแกรมอื่นๆ ได้

ข้อความที่มีการจัดภาพแบบ ASCII จะไม่มีข้อมูลที่มีการจัดภาพแบบ เช่น ตัวหนา ตัวเอียง หรือแบบอักษร เมื่อใช้ Microsoft Notepad หรือบันทึกแฟ้มเป็นข้อความธรรมดาใน Microsoft Office Word แล้ว ASCII จะถูกใช้งาน เช่น การอ่านโฆษณาสำหรับการเปิดรับสมัครงานที่นายจ้างถามถึงประวัติย่อในภาพแบบ ASCII ซึ่งหมายความว่าไม่ว่าจะส่งประวัติย่อในข้อความอีเมล์ โทรสาร หรือสำเนาที่พิมพ์ก็ตาม ผู้จ้างต้องการประวัติย่อที่ไม่มีลักษณะการจัดภาพแบบพิเศษ ข้อความที่มีการจัดภาพแบบ ASCII จะทำงานได้ดีกับซอฟต์แวร์การสแกนประเภทการรู้จำอักขระด้วยแสง (OCR) ที่บริษัทขนาดใหญ่หลายแห่งมักจะใช้เพื่อสแกนประวัติย่อ

2.2.1 แทรกอักขระ ASCII ลงในเอกสาร

นอกจากการพิมพ์อักขระด้วยแป้นพิมพ์ เราสามารถใช้โค้ดอักขระของสัญลักษณ์เป็นทางลัด หรือเมื่อสัญลักษณ์ไม่พร้อมใช้งานบนแป้นพิมพ์ที่กำลังใช้งานอยู่หากต้องการแทรกอักขระ ASCII จากตารางที่ 2.1 ให้กดแป้น ALT ค้างไว้พร้อมกับพิมพ์เลขฐานสิบที่ระบุไว้ ตัวอย่างเช่น เมื่อต้องการแทรกสัญลักษณ์องศา (?) ให้กด ALT ค้างไว้ในขณะที่พิมพ์ 0176 บนแป้นพิมพ์ตัวเลข อักขระ ASCII ที่พิมพ์ได้ ตัวเลข 32–126 ถูกกำหนดให้ใช้แทนอักขระที่คุณจะพบบนแป้นพิมพ์ และจะปรากฏขึ้นเมื่อคุณดูหรือสั่งพิมพ์เอกสาร ตัวเลข 127 ใช้แทนคำสั่ง DELETE

ตารางที่ 2.1 อักขระ ASCII ที่พิมพ์ได้

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0010 0000	32	20	(ช่องว่าง)
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

ตารางที่ 2.2 อักขระควบคุม ASCII ที่พิมพ์ไม่ได้

ฐานสอง	ฐานสิบ	ฐานสิบหก	อักขระ	ความหมาย
0000 0000	0	00	(ว่าง)	NUL - null character
0000 0001	1	01	☺	SOH - start of heading
0000 0010	2	02	☻	STX - start of text
0000 0011	3	03	♥	ETX - end of text
0000 0100	4	04	♦	EOT - end of transmission
0000 0101	5	05	♣	ENQ - enquiry
0000 0110	6	06	♠	ACK - acknowledge
0000 0111	7	07	•	BEL - bell
0000 1000	8	08	◘	BS - backspace
0000 1001	9	09	○	HT - horizontal tabulation
0000 1010	10	0A	◙	LF - line feed
0000 1011	11	0B	♂	VT - vertical tabulation
0000 1100	12	0C	♀	FF - form feed
0000 1101	13	0D	♪	CR - carriage return
0000 1110	14	0E	♫	SO - shift out
0000 1111	15	0F	☼	SI - shift in
0001 0000	16	10	►	DLE - data link escape
0001 0001	17	11	◄	DC1 - device control one
0001 0010	18	12	↕	DC2 - device control two
0001 0011	19	13	‼	DC3 - device control three
0001 0100	20	14	¶	DC4 - device control four
0001 0101	21	15	§	NAK - negative acknowledge
0001 0110	22	16	▬	SYN - synchronous idle
0001 0111	23	17	↨	ETB - end of transmission block
0001 1000	24	18	↑	CAN - cancel
0001 1001	25	19	↓	EM - end of medium
0001 1010	26	1A	→	SUB - substitute
0001 1011	27	1B	←	ESC - escape
0001 1100	28	1C	∟	FS - file separator
0001 1101	29	1D	↔	GS - group separator
0001 1110	30	1E	▲	RS - record separator
0001 1111	31	1F	▼	US - unit separator
0111 1111	127	7F	⌂	DEL - delete

นอกเหนือจาก ASCII ตารางของอักขระอื่นซึ่งใหม่กว่านี้เรียกว่า Unicode เนื่องจาก Unicode เป็นตารางที่ใหญ่กว่ามาก จึงมีอักขระถึง 65,536 ตัวเทียบกับ ASCII ที่มีเพียง 128 ตัว หรือ ASCII เพิ่มเติมที่มีอักขระทั้งสิ้น 256 ตัว ความจุที่มากขึ้นนี่เองที่ทำให้อักขระส่วนใหญ่ของภาษาต่างๆ สามารถรวมอยู่ในอักขระชุดเดียวได้

2.3 การเข้ารหัสแบบ Base16 Base32 และ Base64

2.3.1 Base16

Base16 เป็นการเข้ารหัสที่เป็นมาตรฐาน การเข้ารหัสเลขฐานสิบหกจึงเรียกว่า "Base16" หรือ "hex" 16 ตัวอักษรของ US-ASCII ถูกนำมาใช้ทำให้ 4 บิตเป็นตัวแทนต่อตัวอักษรที่พิมพ์ การเข้ารหัสเป็นกลุ่ม 8 บิต (octets) ของบิตอินพุต เป็นสตริงเอาท์พุทของตัวอักษรที่เข้ารหัส 2 การดำเนินการจากทางซ้ายไปขวา 8 บิตจะมาจากการป้อนข้อมูล จาก 8 บิตทำการตัดแบ่งเป็น 2 กลุ่มแต่ละกลุ่มประกอบด้วย 4 บิตซึ่งเป็นแปลงเป็นตัวอักษรตัวเดียวใน Base16 ตัวอักษร แต่ละกลุ่ม 4 บิตจะใช้เป็นดัชนีในอาร์เรย์ของ 16 ตัวอักษรตัวพิพม์ ตัวอ้างอิงโดยดัชนี

จะอยู่ในสตริงเอาท์พุทดังตารางที่ 2.3 อย่างไรก็ตาม Base16 ไม่มีตัวอักษรพิเศษเหมือนกับ Base32 และ Base64

ตารางที่ 2.3 อินเด็กซ์ของข้อมูลสตริง Base16

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	0	4	4	8	8	12	C
1	1	5	5	9	9	13	D
2	2	6	6	10	A	14	E
3	3	7	7	11	B	15	F

2.3.2 Base32

คือการเข้ารหัสด้วยตัวอักษรระหว่าง A–V และตัวเลขจาก 0–9 สำหรับการเข้ารหัสข้อมูลไบต์โดยใช้ชุดของสัญลักษณ์ที่สามารถใช้ประมวลผลโดยระบบคอมพิวเตอร์ได้ ชุดตัวอักษรประกอบไปด้วยชุดสัญลักษณ์ที่สร้างขึ้นโดยมีความแตกต่างกัน 32 ตัวอักษรเช่นเดียวกับที่อัลกอริทึมสำหรับการเข้ารหัสสตริงโดยใช้อักขระ 8 บิตเป็นตัวอักษร Base32 นี้ใช้สัญลักษณ์ Base32 5 บิตจะเป็นตัวแทนต่อตัวอักษร (ตัวที่ 33 จะเป็นตัวอักษรพิเศษคือ "= " ถูกนำมาใช้ เพื่อใช้ในฟังก์ชันการประมวลผลพิเศษ)

ข้อดี

ผลจากการเข้ารหัสสามารถนำไปสร้างชื่อไฟล์ได้เนื่องจากไม่มีสัญลักษณ์ "/"

ตัวอักษรที่ใช้ไม่มีตัวอักษรต้องห้ามที่จะใช้กับ URL

ตารางที่ 2.4 ร้อยละของข้อมูลไบนารีระหว่าง Base32 และ Base64

	Base64	Base32
8-bit	133%	160%
7-bit	117%	140%

ตารางที่ 2.5 อินเด็กซ์ของข้อมูลสตริง Base32

Value	Symbol	Value	Symbol	Value	Symbol	Value	Symbol
0	0	9	9	18	I	27	R
1	1	10	A	19	J	28	S
2	2	11	B	20	K	29	T
3	3	12	C	21	L	30	U
4	4	13	D	22	M	31	V
5	5	14	E	23	N		
6	6	15	F	24	O		
7	7	16	G	25	P		
8	8	17	H	26	Q	*pad*	=

ตารางที่ 2.6 การเข้ารหัสด้วย Base32

Bit pattern	0	1	1	0	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1	1	0	0	1	1	0	0	1	0
Index	12					9					11					23					6					pad			
Base32-encoded	C					9					B					N					6					=			

การประมวลผลพิเศษ (Special processing) คือการแทนที่ตัวอักษรของตัวอักษรที่น้อยกว่า 40 บิต เมื่อถูกแบ่งเป็นกลุ่มละ 5 บิต กลุ่มบิตสุดท้ายจะมีไม่ถึง 5 บิต ดังนั้นกลุ่มบิตที่มีไม่ถึง 5 บิตจะถูกประมวลผลพิเศษโดยการแทนด้วย "="

2.3.3 Base64

คือ กลุ่มของเลขฐานสองที่อยู่ในภาพแบบการเข้ารหัสที่คล้ายกันที่เป็นตัวแทนของข้อมูลไบนารีในภาพแบบสตริง ASCII โดยการแปลงสัญลักษณ์ 65 ตัวอักษรของ US-ASCII จะถูกนำมาใช้ทำให้เป็น 6 บิต (ตัวอักษรที่ 65 เป็นตัวอักษรพิเศษคือ "=" ถูกนำมาใช้เพื่อบ่งฟังก์ชันการประมวลผลพิเศษ) การเข้ารหัสเป็นกลุ่ม 24 บิตของบิตอินพุตจะถูกแบ่งเป็น 4 ตัวอักษรโดยการเข้ารหัสจากซ้ายไปขวาซึ่งแปลเป็นตัวอักษรตัวเดียวใน Base 64 ตัวอักษร แต่ละกลุ่ม 6 บิตจะใช้เป็นดัชนีในอาร์เรย์ของ 64 ตัวอักษร โดยดัชนีจะอยู่ในสตริงเอาท์พุท Base64 ซึ่งมักใช้เมื่อมีความจำเป็นในการเข้ารหัสข้อมูลไบนารีที่จะต้องมีการจัดเก็บและถ่ายโอนผ่านสื่อที่ถูกออกแบบมาเพื่อจัดการกับข้อมูลต้นฉบับเดิม เพื่อให้แน่ใจว่าข้อมูลที่ยังคงเหมือนเดิมโดยไม่มีการดัดแปลงใน

ระหว่างการส่งข้อมูล Base64 เป็นที่นิยมใช้งานไม่ว่าจะเป็นการสื่อสารผ่านอีเมล์และการจัดเก็บข้อมูลที่ซับซ้อนใน XML

ตารางที่ 2.7 การเข้ารหัสด้วย Base64

Text content	M								a								n							
ASCII	77 (0x4d)								97 (0x61)								110 (0x6e)							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19								22								5							
Base64-encoded	T								W								F							

ตารางที่ 2.8 อินเด็กซ์ของข้อมูลสตริง Base64

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

2.4 การจัดการข้อมูลบนเว็บ

งานวิจัยในนี้ มีการพัฒนาโปรแกรมด้วยภาษาจาวาและทำการจัดเก็บข้อมูลลงฐานข้อมูล MySQL โดยเก็บข้อมูลชนิด varchar(1024) มุ่งเน้นที่จะสร้างไฟล์นามสกุล .jar เพื่อเป็นไลบารี่

(Library) กลางให้เรียกใช้ได้กับโปรเจ็คต่าง ๆ ได้ มีข้อดีคือโปรเจ็คต่างๆ สามารถเรียกใช้งานได้ถ้ามีไฟล์ Base128.jar มีข้อเสียอยู่เล็กน้อยคือใช้งานได้เฉพาะโปรแกรมที่สร้างด้วยภาษาจาวา

ตัวอย่างขั้นตอนการสร้าง .jar

เรามาเริ่มจาก HelloWorld กันดีกว่า โดยผมจะแยก folder ที่เก็บ source code คือ src และ folder ที่เก็บ class ที่ได้จากการ compile ว่า classes นะครับ

1. สร้าง folder 'src' และ 'src\pleX'
2. โปรแกรม HelloWorld ซึ่งอยู่ใน package pleX

```
package pleX;
public class HelloWorld {
public static void main(String[] args) {
 System.out.println("Hello World");
   }
   }
```

3. สร้าง folder '\classes'
4. จัดการ compile และ run ดังนี้

```
javac -sourcepath src -d classes src\pleX\HelloWorld.java
java -cp classes pleX.HelloWorld
```

ผลลัพธ์ที่ได้ก็คือ => Hello World

5. คราวนี้ลองมาสร้าง Jar file กัน ดังนี้
6. พิมพ์ข้อความ "Main-Class: pleX.HelloWorld" ใน text file ที่ชื่อว่า myManifest (โดยไม่ต้องมีเครื่องหมาย ")
7. สร้าง folder '\jar'
8. จากนั้นสั่ง

```
jar cfm jar\HelloWorld.jar myManifest -C classes .
```

เราก็จะได้ HelloWorld.jar มาใช้งานแล้ว

9. วิธีเรียก jar file ทำได้ดังนี้

```
java -jar jar\HelloWorld.jar
```

2.5 การเข้ารหัสข้อมูล

การเข้ารหัสข้อมูลโดยพื้นฐานแล้ว เกี่ยวข้องกับวิธีการทางคณิตศาสตร์ เพื่อใช้ในการป้องกันข้อมูลที่ต้องการส่งไปถึงผู้รับ ข้อมูลตั้งต้นจะถูกแปรเปลี่ยนไปสู่ข้อมูลอีกภาพแบบหนึ่งที่ไม่สามารถอ่านเข้าใจได้โดยใครก็ตามที่ไม่มีกุญแจสำหรับเปิดดูข้อมูลนั้น ซึ่งจะเรียกกระบวนการในการแปรภาพของข้อมูลว่า การเข้ารหัสข้อมูล (Encryption) และเรียกกระบวนการในการแปลงข้อความที่ไม่สามารถอ่านและทำความเข้าใจให้กลับไปสู่ข้อความดั้งเดิมว่า การถอดรหัสข้อมูล (Decryption) อัลกอริทึมที่ใช้ในการเข้าและถอดรหัสข้อมูลสามารถแบ่งออกตามลักษณะของกุญแจที่ใช้ได้ 2 ประเภทหลัก คือ

2.5.1 อัลกอริทึมแบบสมมาตร

อัลกอริทึมนี้ส่วนใหญ่กุญแจที่ใช้ในการเข้าและถอดรหัสลับมักจะเป็นตัวเดียวกัน เรียกว่า กุญแจลับ (Secret Key) อัลกอริทึมนี้ยังสามารถแบ่งย่อยออกเป็น 2 แบบ ได้แก่ แบบบล็อก (Block Algorithms) ซึ่งจะทำการเข้ารหัสข้อมูลทีละบล็อกและแบบสตรีม (Stream Algorithms) ซึ่งจะทำการเข้ารหัสข้อมูลทีละไบต์(Byte) ผู้ใช้งานจำเป็นจะต้องเห็นชอบและยอมรับที่จะใช้กุญแจตัวเดียวกันเสียก่อนที่จะมีการติดต่อสื่อสารระหว่างกัน เนื่องจากการแลกเปลี่ยนกุญแจนั้นอาจเกิดความยุ่งยากและไม่สะดวก ความปลอดภัยของอัลกอริทึมสมมาตรนี้จะขึ้นอยู่กับตัวกุญแจลับเป็นสำคัญ บุคคลใดที่ได้กุญแจลับนี้มา ไม่ว่าจะโดยวิธีการใด สามารถที่จะทำการเข้ารหัสลับข้อความต้นฉบับหรือทำการถอดรหัสลับข้อความรหัสใดๆ ก็ได้สำหรับสองกลุ่มที่ต้องการติดต่อกันจำเป็นต้องใช้กุญแจลับเป็นจำนวน 1 กุญแจเพื่อติดต่อกัน สมมติว่ามีผู้ที่ต้องติดต่อกันเป็นจำนวน n กลุ่ม จำนวนกุญแจลับทั้งหมดที่ต้องแลกเปลี่ยนกันคิดเป็นจำนวนทั้งหมดคือn (n-1)/2 กุญแจ ซึ่งจะเห็นได้ว่าจำนวนกุญแจมีมากมายเกินไป ซึ่งอาจก่อให้เกิดปัญหาด้านการรักษาความปลอดภัยให้กับกุญแจเหล่านี้ แต่อัลกอริทึมแบบสมมาตรนั้นมีข้อดี คือสามารถทำงานได้รวดเร็วกว่าและง่ายต่อการใช้งานมากกว่าอัลกอริทึมแบบอสมมาตร

2.5.2 อัลกอริทึมแบบอสมมาตร

อัลกอริทึมประเภทนี้ใช้กุญแจในการทำงานด้วยกัน 2 ตัว ตัวหนึ่งเรียกว่า กุญแจสาธารณะ (Public keys) ใช้ในการเข้ารหัสข้อมูล และอีกตัวหนึ่งเรียกว่า กุญแจส่วนตัว (Private keys) ใช้ในการถอดรหัสข้อมูลที่เข้ารหัสแล้ว โดยที่กุญแจสาธารณะสามารถส่งมอบให้กับผู้อื่นเพื่อให้ผู้อื่นสามารถนำไปใช้งานได้ สำหรับกุญแจส่วนตัวนั้นต้องเก็บไว้กับผู้เป็นเจ้าของกุญแจส่วนตัวเท่านั้นและห้ามเปิดเผยให้ผู้อื่นทราบโดยเด็ดขาดโดยวิธีนี้จะไม่มีผู้อื่นสามารถเปิดดูข้อความที่เข้ารหัสนั้นได้ยกเว้นผู้ที่ถือกุญแจส่วนตัว ที่เป็นคู่ของกุญแจสาธารณะนั้น การเผยแพร่กุญแจสาธารณะในสถานที่ต่าง ๆ ได้ ทำให้ลดความยุ่งยากในการแลกเปลี่ยนกุญแจกันซึ่งเป็น

ปัญหาของอัลกอรึทึมแบบสมมาตร และวิธีการของอัลกอริทึมแบบอสมมาตรใช้จำนวนกุญแจที่ประหยัดกว่า เนื่องจากกุญแจสาธารณะ 1 กุญแจของกลุ่ม ๆ หนึ่งจะสามารถเผยแพร่ให้กับกี่กลุ่มก็ได้ ที่เราต้องการติดต่อ ดังนั้นถ้ามีกลุ่มที่ต้องติดต่อกันจำนวน n กลุ่ม จำนวนกุญแจส่วนตัวที่ต้องระวังรักษาก็คือ n กุญแจ ซึ่งจะเห็นได้ว่าลดลงไปได้เป็นจำนวนมาก

2.6 ความแข็งแกร่งของอัลกอริทึม

สำหรับการเข้ารหัสความแข็งแกร่งของอัลกอริทึม หมายถึงความยากในการที่ผู้บุกรุกจะสามารถถอดรหัสข้อมูลได้โดยปราศจากกุญแจที่ใช้ในการเข้ารหัส ซึ่งจะขึ้นอยู่กับปัจจัยดังนี้

2.6.1 การเก็บกุญแจเข้ารหัสไว้อย่างเป็นความลับ ผู้เป็นเจ้าของกุญแจลับหรือส่วนตัวต้องระมัดระวังไม่ให้กุญแจสูญหายหรือล่วงรู้โดยผู้อื่น

2.6.2 ความยาวของกุญแจเข้ารหัส ปกติกุญแจเข้ารหัสจะมีความยาวเป็นบิต ยิ่งจำนวนบิตของกุญแจยิ่งมาก ยิ่งทำให้การเดาเพื่อสุ่มหากุญแจที่ถูกต้องเป็นไปได้ยาก แต่หากความยาวของกุญแจเข้ารหัสมีมากเกินไป จะทำให้การทำงานช้าลงไปด้วย จึงต้องดูให้เหมาะสม

2.6.3 ความไม่เกรงกลัวต่อการศึกษาหรือดูอัลกอริทึมเพื่อหาภาพแบบของการเข้ารหัส อัลกอริทึมที่ดีต้องเปิดให้ผู้รู้ทำการศึกษาในรายละเอียดได้โดยไม่เกรงว่าผู้ศึกษาจะสามารถจับภาพแบบของการเข้ารหัสได้

2.6.4 การมีประตูลับในอัลกอริทึม อัลกอริทึมที่ดีต้องไม่แฝงไว้ด้วยประตูลับที่สามารถใช้เป็นทางเข้าไปสู่อัลกอริทึม แล้วอาจใช้เพื่อทำการถอดรหัสข้อมูลได้ ประตูลับนี้ทำให้ไม่จำเป็นต้องใช้กุญแจในการถอดรหัส

2.6.5 ความไม่เกรงกลัวต่อปัญหาการหาความสัมพันธ์ในข้อมูลที่ได้รับ กล่าวคือเมื่อผู้ไม่ประสงค์ดีทราบข้อมูลบางอย่างที่เป็นข้อมูลตั้งต้น ซึ่งยังไม่ได้เข้ารหัส รวมทั้งมีข้อมูลที่เข้ารหัสแล้วของข้อมูลตั้งต้นนั้น ผู้ไม่ประสงค์ดีอาจสามารถหาความสัมพันธ์ระหว่างข้อความทั้งสองนั้นได้ ซึ่งจะเป็นวิธีการในการถอดรหัสข้อมูลได้

2.6.6 คุณสมบัติของข้อความตั้งต้น คุณสมบัตินี้อาจใช้เป็นช่องทางในการถอดรหัสข้อมูลได้ อัลกอริทึมที่ดีต้องไม่ใช้คุณสมบัติของข้อความเป็นกลไกในการเข้ารหัสข้อมูล

คำแนะนำในการเลือกใช้อัลกอริทึมคือ ให้ใช้อัลกอริทึมที่ได้มีการใช้งานมาเป็นระยะเวลานานแล้ว เนื่องจากหากอัลกอริทึมนี้มีปัญหาจริง คงเกิดขึ้นมาและเป็นที่ทราบกันนานแล้ว ดังนั้นจึงไม่ควรใช้อัลกอริทึมใหม่ ๆ ที่เพิ่งได้มีการนำเสนอกันสู่สาธารณะ เพราะอาจมีช่องโหว่แฝงอยู่และยังไม่เป็นที่ทราบในขณะนี้

2.7 อัลกอริทึม AES

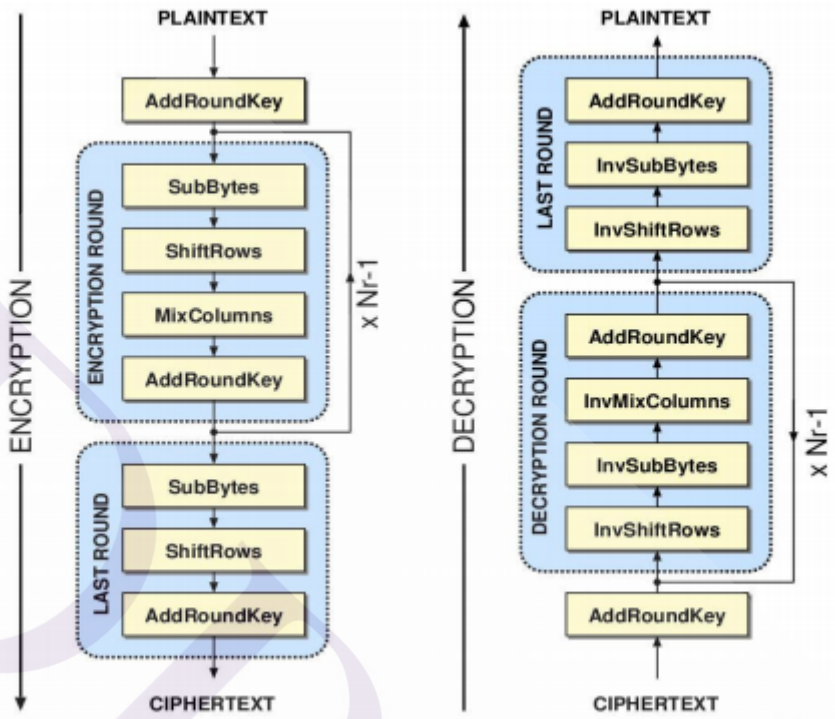
[8]อัลกอริทึม AES เป็นอัลกอริทึมแบบสมมาตรที่มีการคิดค้นพัฒนาขึ้นโดยชาวเบลเยี่ยม 2 คน จากอัลกอริทึมที่ใช้ชื่อเดิมว่า ราจินเดล (Rijndael) เป็นอัลกอริทึมที่ใช้เข้าและถอดรหัสข้อมูลที่มาตรฐานของ NIST (The US National Institute of Standards and Technology) ให้การยอมรับซึ่งเป็นหน่วยงานหนึ่งของกระทรวงพาณิชย์สหรัฐ ที่สนับสนุนและรักษามาตรฐานการวัด รวมถึงโครงการสำหรับการเข้าร่วมและช่วยเหลืออุตสาหกรรมและวิทยาศาสตร์ เพื่อพัฒนาและใช้มาตรฐานเหล่านี้ อัลกอริทึม AES ถือเป็นอัลกอริทึมแบบสมมาตร ที่ได้รับการยอมรับจากองค์กรต่าง ๆ และได้รับความนิยมจากคนทั่วโลกมากที่สุด ทั้งนี้เนื่องมาจากความรวดเร็วของการประมวลผลในกระบวนการเข้าและถอดรหัสลับ โดยได้รับการออกแบบให้มีการทำงานที่เหมาะสมกับโพรเซสเซอร์ (Processor) รุ่นใหม่ ๆ ความต้องการหน่วยความจำเพื่อใช้ในการเข้าและถอดรหัสลับที่น้อย รวมทั้งความง่ายในการนำไปประยุกต์ใช้งานทั้งในภาพของซอฟต์แวร์และฮาร์ดแวร์ได้อย่างกว้างขวาง เช่น สมาร์ตการ์ด (Smart Card) เป็นต้น

อัลกอริทึม AES จะเข้ารหัสลับข้อมูลเป็นกลุ่ม กลุ่มละ 128 บิต กุญแจที่ใช้ในการเข้าและถอดรหัสมีขนาด 128 192 หรือ 256 บิต ขึ้นอยู่กับผู้ใช้งาน ขนาดของกุญแจจะเป็นตัวกำหนดจำนวนรอบของการประมวลผล ซึ่งอาจแสดงในภาพของบิตหรือคำก็ได้โดยกุญแจขนาด n บิต จะมีขนาดความยาวของกุญแจ (Nk) เท่ากับ n/32 คำ ขนาดของกุญแจจะเป็นตัวกำหนดจำนวนรอบของการประมวลผล (Nr) ซึ่งตามมาตรฐานอัลกอริทึม AES ได้กำหนดไว้ว่า กุญแจที่มีขนาด 256 บิตจะมีการวนรอบในการประมวลผล14 รอบ ดังแสดงในตารางที่ 2.8

ตารางที่ 2.9 ภาพย่อที่แสดงถึงการทำงานในอัลกอริทึม AES-256

Mode	ขนาดความยาวของกุญแจ(Nk)	จำนวนรอบของการประมวลผล(Nr)
AES-256	8	14

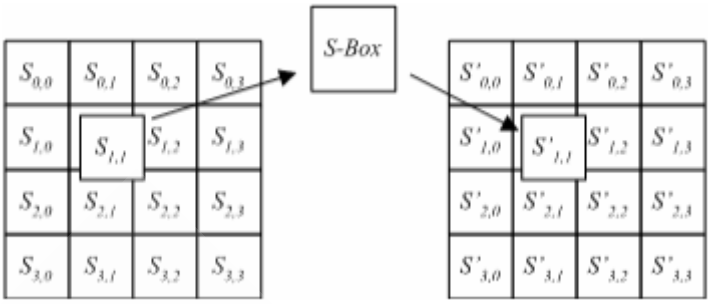
การเข้ารหัสลับอัลกอริทึม AES มีโครงสร้างพื้นฐานดังภาพที่ 2.1 ภาพฝั่งซ้ายแสดงการเข้ารหัสข้อมูล และภาพฝั่งขวาแสดงการถอดรหัสข้อมูล



ภาพที่ 2.1 โครงสร้างพื้นฐานการเข้าและถอดรหัสของอัลกอริทึม AES

จากภาพอธิบายโครงสร้างการเข้าและถอดรหัสแบบ AES โดยจะพิจารณาชุดข้อมูลครั้งละ 128 บิต และจะถูกจัดให้อยู่ในภาพของอะเรย์ (Array) ของไบต์ 2 มิติเรียกว่า สเตท (State) โดยสเตท นี้จะประกอบด้วยไบต์จำนวน 4แถวเสมอ ซึ่งจะถูกแก้ไขในทุกวงรอบของการประมวลผลและในวงรอบสุดท้าย สเตทนี้จะถูกคัดลอกกลับไปเป็นผลลัพธ์การดำเนินงาน ในแต่ละรอบของการประมวลผล จะประกอบไปด้วยกระบวนการย่อย 4 กระบวนการดังนี้ คือ SubstituteBytes, ShiftRows, MixColumns และ AddRoundKey

2.7.1 กระบวนการย่อย SubstituteBytes อัลกอริทึม AES-256 ใช้เมทริกซ์ขนาด16 ×16 ของชุดข้อมูลที่มีลักษณะเป็นไบต์เราจะเรียกตารางชุดนี้ว่า S-box ซึ่งเป็นตารางการสลับตำแหน่ง ของตัวเลข 8-bit ทั้ง 256 จำนวน จะทำการแทนที่ไบต์ข้อมูลที่อยู่ในสเตทโดยใช้ตางรางการแทนที่ S-box ที่กำหนด ดูภาพที่ 2.2และ 2.3 ประกอบ



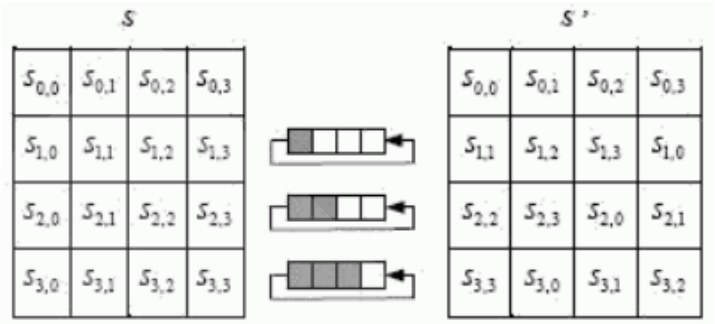
ภาพที่ 2.2 กระบวนการย่อย SubstituteBytes

การแทนที่โดยใช้ S-box ในอัลกอริทึม AES-256 นั้น ค่าแต่ละไบต์ของสเตทจะเกี่ยวโยงกับ ไบต์ที่กำหนดขึ้นมาใหม่ดังนี้คือ โดยบิตที่ 1 ถึง 4 (b1b2b3b4)จะใช้เป็นตัวกำหนดตำแหน่งเชิงตัวเลขฐานสิบหก 0-F ในแต่ละแถวของ S-box ในขณะที่บิต 5 ถึง 8(b5b6b7b8) จะใช้กำหนดตำแหน่งเชิงตัวเลขฐานสิบหก 0-F ในแต่ละคอลัมน์ของ S-box ตัวอย่างเช่น ค่าสเตท S1,1 เป็น 95 จะใช้อ้างถึงแถวนอนที่ 9 และแถวตั้งที่ 5 ของตาราง S-box ซึ่งตรงกับค่า 2A ดังนั้น {95} ถูกแปลงไปให้เป็น {2A}ดังแสดงในภาพที่ 2.3

		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

ภาพที่ 2.3 การแทนที่ S-box ในอัลกอริทึม AES

2.7.2 กระบวนการย่อย ShiftRows จะทำการเลื่อนไบต์เป็นวงกลมไปทางซ้าย n ไบต์ โดยในแถวที่ 1, 2,3 และ 4 จะมีค่า n เท่ากับ 0, 1, 2 และ 3 ตามลำดับ ดังแสดงในภาพที่ 2.4



ภาพที่ 2.4 กระบวนการย่อยShiftRows

2.7.3 กระบวนการย่อย MixColumns นำข้อมูลแต่ละคอลัมน์นำมาคูณเมทริกซ์ที่กำหนดไว้ ดังแสดงในภาพที่ 2.5 และนำคำตอบที่ได้ใส่เข้าไปแทนที่ในตำแหน่งเดิม

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad for \quad 0 \le c \le N_b$$

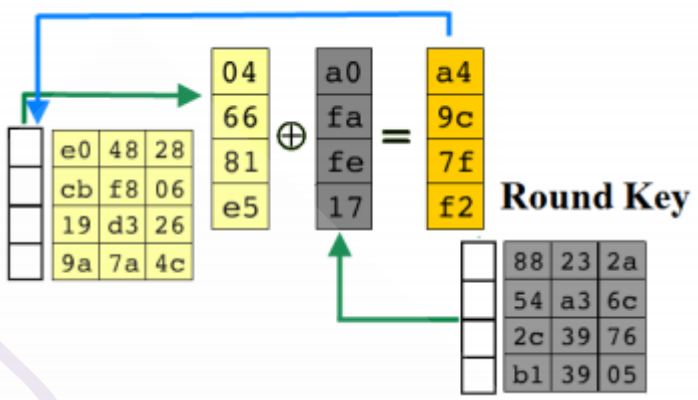
ภาพที่ 2.5 กระบวนการย่อย MixColumns

และผลลัพธ์ที่ได้จากการคูณกันในเมทริกซ์กับค่าคงที่สามารถเขียนในภาพของสมการได้ ดังภาพที่ 2.6

$$S'_{0,C} = (02_{16} \bullet S_{0,C}) \oplus (03_{16} \bullet S_{1,C}) \oplus S_{2,C} \oplus S_{3,C}$$
$$S'_{1,C} = S_{0,C} \oplus (02_{16} \bullet S_{1,C}) \oplus (03_{16} \bullet S_{2,C}) \oplus S_{3,C}$$
$$S'_{2,C} = S_{0,C} \oplus S_{1,C} \oplus (02_{16} \bullet S_{2,C}) \oplus (03_{16} \bullet S_{3,C})$$
$$S'_{3,C} = (03_{16} \bullet S_{0,C}) \oplus S_{1,C} \oplus S_{2,C} \oplus (02_{16} \bullet S_{3,C})$$

ภาพที่ 2.6 สมการการคูณกันของเมทริกซ์กับค่าคงที่ในกระบวนการย่อย MixColumns

2.7.4 กระบวนการย่อย AddRoundKey จะทำการ XOR เฉพาะระหว่างสเตทของกลุ่มข้อมูลกับสเตทของกลุ่มกุญแจย่อยและนำคำตอบที่ได้ใส่เข้าไปแทนที่ในตำแหน่งเดิม ดังภาพที่ 2.7



ภาพที่ 2.7 กระบวนการย่อย AddRoundKey

กุญแจรหัสจะถูกนำมาใช้ในขั้นตอน AddRoundKey เท่านั้น ด้วยเหตุนี้การเข้ารหัสจึงเริ่มและจบด้วยขั้นตอน AddRoundKey ส่วนขั้นตอนอื่น ๆ สามารถทำงานย้อนกลับได้โดยไม่ต้องใช้กุญแจ จากนั้นจึงจะเริ่มเข้าสู่กระบวนการประมวลผลเป็นจำนวน 14 รอบสำหรับอัลกอริทึม AES-256 ซึ่งแต่ละรอบของการประมวลผลจะเป็นการดำเนินการกระบวนการย่อยทั้ง 4 ที่กล่าวมาข้างต้นตามลำดับ อย่างไรก็ตามในรอบสุดท้ายของการประมวลผล ทั้ง 3 โหมดการทำงาน จะไม่มีกระบวนการย่อย MixColumns รวมอยู่ด้วย เมื่อผ่านกระบวนการประมวลผลตามรอบที่กำหนดแล้วผลลัพธ์สุดท้ายที่ได้ก็คือ ข้อความรหัสสำหรับขั้นตอนการเตรียมกลุ่มกุญแจย่อยในภาพสเตทของอัลกอริทึม AES-256จะทำโดยใช้กระบวนการขยายกุญแจ (Key Expansion) ซึ่งประกอบไปด้วย 2 กระบวนการย่อยคือ กระบวนการย่อย Rotwordจะทำการเลื่อนไบต์ในแต่ละคำเป็นวงกลมไปทางซ้าย 1 ไบต์ และ กระบวนการย่อย Subword จะทำการแทนที่ไบต์ข้อมูลที่อยู่ในสเตทของกลุ่มกุญแจย่อย โดยอ้างอิงจาก S-box ตัวเดียวกันกับที่ใช้ในกระบวนการย่อย SubstituteBytes ผลลัพธ์ที่ได้จากกระบวนการข้างต้นจะนำมา XOR เฉพาะกับอะเรย์ของคำที่เป็นค่าคงที่ เรียกว่า Rcon [i] ซึ่ง i คือลำดับของคำ [3]สำหรับกุญแจขนาด 256 บิต สามารถขยายเป็นตารางกุญแจได้ถึง 60คำ(w) ฉะนั้น ตารางกุญแจลับ จะประกอบด้วย อะเรย์ตั้งแต่ W[0] - W[59] ค่า W[0] - W[7] ตั้งให้มีค่าเท่ากับกุญแจลับที่ผู้ใช้กำหนดมา สำหรับการหาค่าอื่นที่เหลือ ให้แบ่งออกเป็น 3กลุ่มดังนี้กลุ่มแรกเป็นกลุ่มที่ w[i] ที่เป็นจำนวนเท่าของ Nk = 8 ได้แก่ w[8], w[16], w[24],… กลุ่มที่สองคือกลุ่มที่เป็นจำนวนเท่าของค่าบล็อกของข้อมูลได้แก่ w[12], w[20], w[28],… ให้ทำเพียงแค่กระบวนการย่อย

Subword เท่านั้น และกลุ่มสุดท้ายคือที่เหลือ ซึ่งสามารถสรุปการทำงานได้โดยใช้รหัสเทียม (Pseudo Code) ดังที่แสดงในภาพที่ 2.8 จากนั้นจึงนำไปจัดทำเป็นตารางกุญแจ (Key schedule) จากนั้นจึงนำไปผ่านกระบวนการ XOR กับสเตทในแต่ละรอบของขั้นตอน AddRoundKey

```
KeyExpansion(byte key [4*Nk],word w[Nb (Nr+1)], Nk)
begin
        word temp
        i = 0

while(i < Nk)
        w[i]= word(key[4*i] , key[4*i+1], key[4*i+2], key[4*i+3])
i = i+1
end while

        i = Nk

while(i < Nb * (Nr+1))

        temp = w[i-1]

        if( i mod Nk = 0)
                temp = SubWord(Rotword(w[i-1])) xor Rcon[i/Nk ] xor w[i-Nk]
        else if (Nk > 6 and I mod Nk = 4)
                temp = SubWord(w[i-1]) xor w[i-Nk]
        else
                temp = w[i-1] xor w[i-Nk]
        end if

        w[i] = w[i-Nk] xor  temp
        i = i+1

end while

end
```

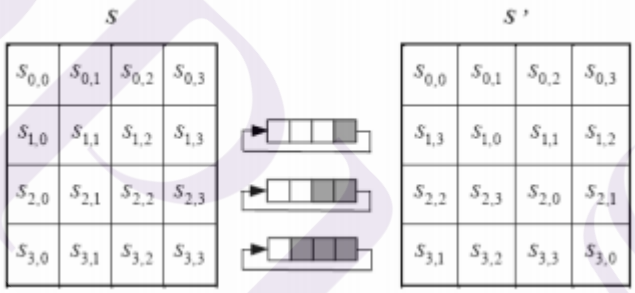
ภาพที่ 2.8 กระบวนการเตรียมกลุ่มกุญแจของอัลกอริทึม AES-256

ในส่วนของกระบวนการถอดรหัสลับอัลกอริทึม AES นั้นสามารถทำได้โดยการใช้กระบวนการย้อนกลับ(Inverse) ดังนี้ InvSubstituteBytes, InvShiftRows, InvMixColumns และ InvAddRoundKey [3]InvSubstituteBytesคือ การทำงานคล้ายกระบวนการย่อย SubstituteBytesแต่จะใช้ตาราง InverseS-Boxดังภาพที่ 2.9 ในการแทนที่ไบต์ข้อมูลที่อยู่ในสเตทแทนตารางS-Box

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

ภาพที่ 2.9 ตาราง InverseS-box ใช้ในกระบวนการ InvSubstituteBytes 13

InvShiftRows คือ การย้อนกลับโดยการเลื่อนไบต์เป็นวงกลมไปทางขวา n ไบต์ โดยในแถวที่ 1, 2, 3 และ 4 จะมีค่า n เท่ากับ 0, 1, 2 และ 3 ตามลำดับ ดังภาพที่ 2.10



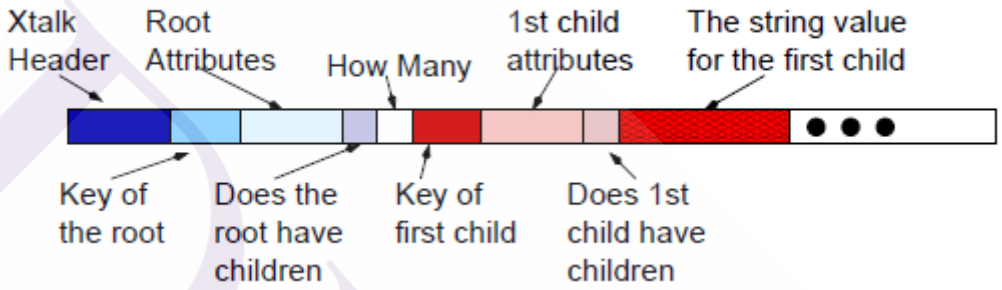
ภาพที่ 2.10 InvShiftRows โดยการเลื่อนกลับ ในอัลกอริทึม AES

InvMixColumns คือ การย้อนกลับด้วยการคูณด้วยเมทริกซ์ย้อนกลับของเมทริกซ์ที่ได้นำไปคูณกับคอลัมน์นั้นAddRoundKey คือ การย้อนกลับโดย XOR คีย์ของรอบนั้นอีกครั้ง

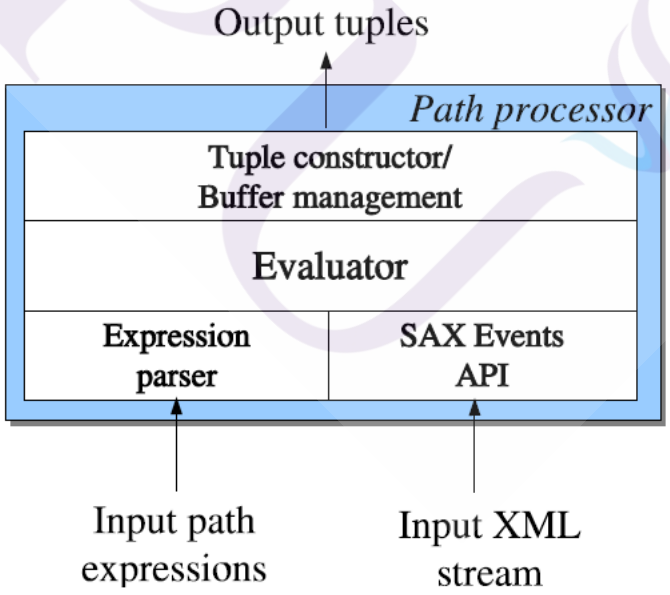
2.8 งานวิจัยที่เกี่ยวข้อง

2.8.1 การประเมินการเข้ารหัส Binary XML สำหรับการใช้การประมวลผลแบบ Fast Stream Based XML (An Evaluation of Binary XML Encoding Optimizations for Fast Stream Based XML Processing) [5] โดยงานวิจัยนี้มีวัตถุประสงค์เพื่อประเมินผลกระทบของการใช้การเข้ารหัสของ binary XML ด้วยกานใช้ fast stream-based XQuery แทนการประมวลผลด้วย binary format เดิม และทำการเปรียบเทียบกับมาตรฐานของ XML ที่มีอยู่ โดยงานวิจัยนี้ได้ทำการศึกษาถึง

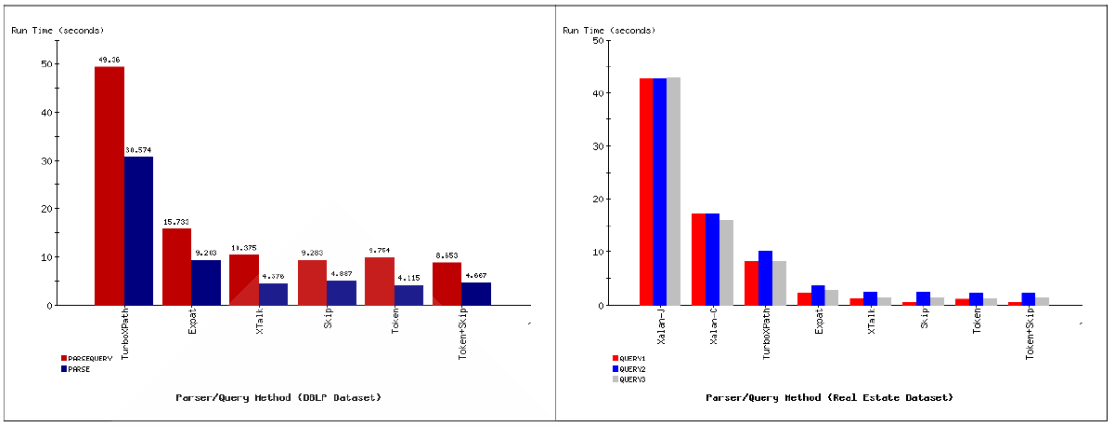
ผลกระทบของการใช้งานการประมวลผลด้วย XML ตัวอย่างเช่น XML database systems และ web services ที่ใช้งานบนเครือข่ายที่มีความรวดเร็ว โดยทำการทดลองบนระบบที่มีคอขวด และใช้วิธีการเข้ารหัสที่หลากหลาย เช่น Trivial Binary Encodings, Tokenization และEmbedded Indexes and Skip-To Pointers ซึ่งการประเมินผลกระทบของ XML นี้ ทำให้ทราบว่าการเพิ่มประสิทธิภาพการเข้ารหัสในการแยก XML และการประยุกต์ใช้แอพพลิเคชั่น คือ การใช้ XQuery ที่ใช้หน่วยความจำที่เพิ่มขึ้น เพื่อเพิ่มประสิทธิภาพการเข้ารหัส โดยการเปรียบเทียบการเข้ารหัสต่างๆ ในงานวิจัยนี้นั้น Trivial Binary Encoding มีประสิทธิภาพดีที่สุด



ภาพที่ 2.11 ภาพแบบ XTalk serialization format



ภาพที่ 2.12 โครงสร้างของ Retrieval phase



ภาพที่ 2.13 ผลลัพธ์จากการทดสอบ ภาพซ้าย DBLP-QUERY และภาพขวา RE-QUERY

2.8.2 การสร้าง Base62x โดยการเลือกใช้ตัวอักษรและตัวเลขจากBase64 (Base62x: An Alternative Approach to Base64 for non-Alphanumeric Characters)[6] ระบบคอมพิวเตอร์ไม่มีตัวอักษรพิมพ์แค่ 62 ตัวอักษรและตัวเลข (0-9, az และ AZ) แต่ยังอักขระที่ไม่ใช่ตัวเลขและตัวอักษรอื่น ๆ (เช่น *, \, ^,%) Base64 เป็นการเข้ารหัสภาพแบบที่ใช้กันทั่วไปที่แสดงถึงข้อมูลไบนารีในภาพแบบสตริง ASCII แต่สำหรับ Base64 ที่ไม่ใช่ตัวอักษรและตัวเลขตัวอักษร (เช่น * \ ^,%) เป็นปัญหาสำหรับชื่อไฟล์และURL ที่ ตัวอย่างเช่น "+" ซึ่งอาจเกิดข้อผิดพลาดได้ง่ายและอยู่ใน URL "/" ไม่ได้รับอนุญาตให้ใช้ตั้งชื่อไฟล์ เพื่อแก้ไขปัญหานี้งานวิจัยนี้จึงนำเสนอใหม่ วิธี Base62x เป็นวิธีการทางเลือกในการ Base64 วิธีที่เสนอมีประสิทธิภาพมากขึ้นในการพัฒนาโปรแกรมและอย่างเข้ากันได้กับการใช้งานสัญลักษณ์ที่สำคัญเช่น ระบบไฟล์ที่อยู่ IP และการส่งผ่านที่ปลอดภัยไม่ ASCII ผ่านทางอินเทอร์เน็ตได้โดยใช้ชุดตัวอักษรดังตารางต่อไปนี้

ตารางที่ 2.10 อินเด็กซ์ของข้อมูลสตริง Base62x

Value	*Encoding*	*Value*	*Encoding*	*Value*	*Encoding*
0	0	22	M	44	i
1	1	23	N	45	j
2	2	24	O	46	k
3	3	25	P	47	l
4	4	26	Q	48	m
5	5	27	R	49	n
6	6	28	S	50	o
7	7	29	T	51	p
8	8	30	U	52	q
9	9	31	V	53	r
10	A	32	W	54	s
11	B	33	X	55	t
12	C	34	Y	56	u
13	D	35	Z	57	v
14	E	36	a	58	w
15	F	37	b	59	y
16	G	38	c	60	z
17	H	39	d	61	x1
18	I	40	e	62	x2
19	J	41	f	63	x3
20	K	42	g		
21	L	43	h	(tag)	x

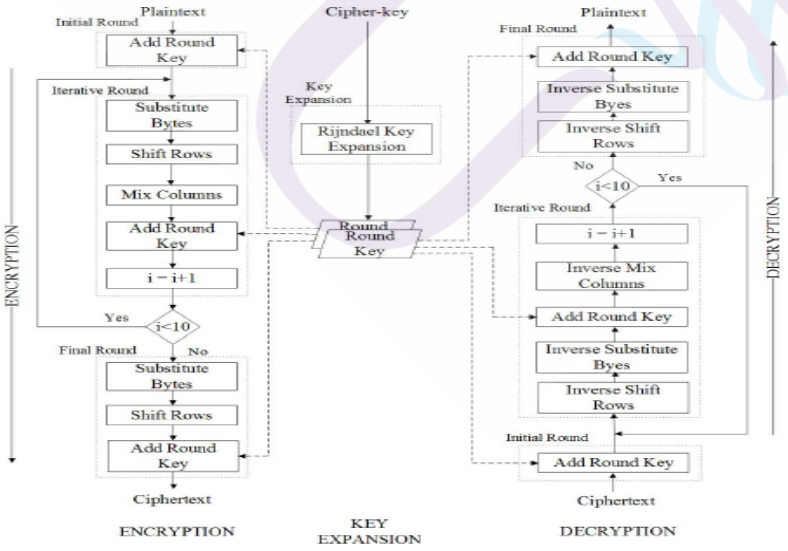
2.8.3 การปรับเปลี่ยนขั้นตอนวิธีการเข้ารหัส Vigenere และการดำเนินงานแบบไฮบริดด้วย Base64 และ AES (Modified Vigenere Encryption Algorithm and its Hybrid Implementation with Base64 and AES) [7] การรักษาความปลอดภัยเป็นกลไกป้องกันการเข้าถึงข้อมูล โดยไม่ได้ตั้งใจหรือไม่ได้รับอนุญาตหรือการเปลี่ยนแปลง เพื่อความปลอดภัยของข้อมูลดังกล่าวกระบวนการของการเข้ารหัสถูกนำมาใช้ การเข้ารหัสลับคือการเปลี่ยนแปลงข้อมูลที่สำคัญผ่านขั้นตอนวิธีที่จะทำให้ข้อมูลไม่สามารถอ่านได้ด้วยวิธีการของคนหรือคอมพิวเตอร์ สำคัญอยู่ที่ความสามารถเข้าถึงได้ ในการนี้วิธีการผสมที่ใช้สำหรับการใช้ขั้นตอนวิธีการเข้ารหัส ชนิดที่แตกต่างกันของขั้นตอนวิธีเช่นตัวเลขแทนเป็นต้น ขั้นตอนวิธีสมมาตรจะใช้ความปลอดภัยของระบบดีขึ้นอย่างมากผ่านการวิจัยที่มีชื่อเสียงหลายขั้นตอนวิธีการเข้ารหัสข้อมูลการปรับปรุงบางส่วน ขั้นตอนวิธีการเข้ารหัสข้อมูลและจัดเรียงไว้ในที่เหมาะสมคำสั่งที่ถูกเลือกให้เป็นตัวชี้วัดสำหรับการวัดประสิทธิภาพการทำงานของขั้นตอนวิธีการและการดำเนินการของขั้นตอนวิธีการที่นำเสนอแสดงให้เห็นถึงผลกระทบอย่างมากเมื่อเทียบกับขั้นตอนวิธีการเข้ารหัส Vigenered

การเข้ารหัสข้อมูลโดยวิธี AES วิธีเข้ารหัสข้อมูลแบบ AES ถูกกำหนดโดย National Institute of Standards and Technology ในปี ค.ศ.2001 ให้เป็นวิธีเข้ารหัสข้อมูลมาตรฐานของประเทศสหรัฐอเมริกา โดยใช้วิธีเข้ารหัสแบบ Rijndael ที่เสนอโดย Joan Daemen กับ Vincent Rijme

วิธี AES จะเข้ารหัสข้อมูลครั้งละ 128 bit (16 byte) โดยใช้กุญแจขนาด 128,192 และ 256 bit ในทางทฤษฎี จะเลือกขนาดข้อมูลที่ใช้เข้ารหัสและขนาดของกุญแจเป็นพหุคูณของ 32 โดยขนาดของข้อมูลและของกุญแจต่ำสุดเป็น 128 bit และขนาดของข้อมูลสูงสุดเป็น 256 bit การเข้ารหัส/ถอดรหัสจะทำบน matrix ขนาด 4X4 byte (แต่ถ้าเพิ่มขนาดข้อมูลขึ้น จะต้องเพิ่มจำนวนหลักใน matrix) มีการทำงานแบ่งเป็น 2 ขั้นตอน คือ

1. การสร้าง roundkey ที่ใช้ในการเข้ารหัส/ถอดรหัสจากกุญแจที่กำหนดไว้ การทำงานส่วนนี้ทำครั้งเดียวและได้ผลลัพธ์เดียวกันถ้าใช้กุญแจชุดเดียวกัน ที่ขนาดข้อมูลและขนาดของกุญแจอย่างเดียวกัน โดย roundkey ที่ถูกสร้างจะมีขนาดขึ้นกับขนาดของกุญแจที่เลือกไว้ เช่น กุญแจขนาด 128 bit จะได้ roundkey ขนาด 176 Byte, กุญแจขนาด 192 bit จะได้ roundkey ขนาด 208 Byte, กุญแจขนาด 256 bit จะได้ roundkey ขนาด 240 Byte

2. การเข้ารหัส/ถอดรหัสข้อมูล ซึ่งทำงานหลายรอบขึ้นกับขนาดของกุญแจที่ระบุ เช่น กุญแจขนาด 128 bit จะทำงาน 10 รอบ, กุญแจขนาด 192 bit จะทำงาน 12 รอบ, กุญแจขนาด 256 bit จะทำงาน 14 รอบ โดยมีการทำ 4 ส่วนหลักดังภาพที่ 2.12



ภาพที่ 2.14 กระบวนการของ AES (Advanced Encryption Standard)

1. SubBytes เป็นการแทนข้อมูลแต่ละ byte ด้วยค่าจากตารางค่าคงที่ตัวหนึ่งจาก 256 ตัวที่มีอยู่ โดยเลือกค่าที่ใช้จากค่าของข้อมูลเดิม เช่น ถ้าข้อมูลที่จะถูกแทนที่มีค่า 0 จะนำค่าจากตารางช่องที่ 0 มาแทนที่, ถ้าข้อมูลที่จะถูกแทนที่มีค่า 1 จะนำค่าจากตารางช่องที่ 1 มาแทนที่,..., ถ้าข้อมูลที่จะถูกแทนที่มีค่า 255 จะนำค่าจากตารางช่องที่ 25มาแทนที่ โดยที่ตารางที่ใช้ในการถอดรหัสและตารางที่ใช้ในการเข้ารหัสจะต่างกัน

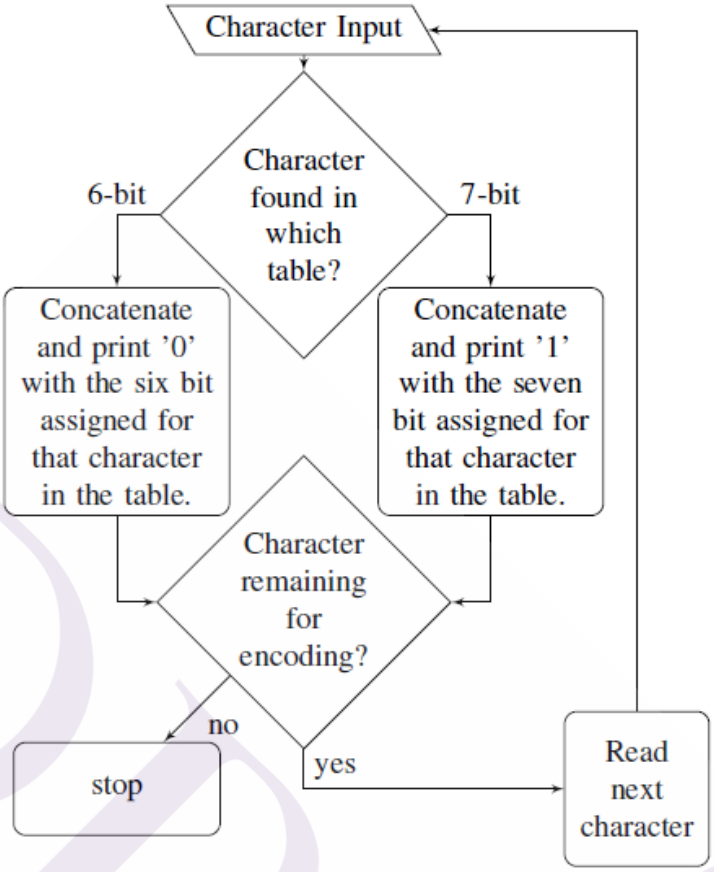
2. ShiftRows เป็นการเปลี่ยนตำแหน่งข้อมูลใน matrix สำหรับการเข้ารหัสข้อมูลจะถูกเลื่อนไปทางซ้าย โดยที่ข้อมูลแถวที่ 1 จะไม่ถูกเลื่อน, ข้อมูลแถวที่ 2 จะเลื่อนไป 1 หลัก, ข้อมูลแถวที่ 3 จะเลื่อนไป 2 หลัก และข้อมูลแถวที่ 4 จะเลื่อนไป 3 หลัก

สำหรับการถอดรหัสการเลื่อนตำแหน่งจะกลับทิศทางกัน คือเลื่อนไปทางขวา ส่านจำนวนการเลื่อนในแต่ละหลักจะเหมือนกับการเข้ารหัส

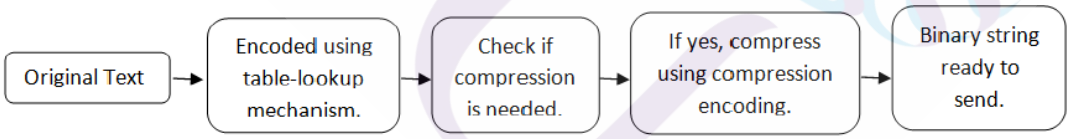
3. MixColumns เป็นการนำข้อมูลแต่ละหลักใน matrix ไปคูณกับ matrix ค่าคงที่ที่กำหนด

4. AddRoundKey ขั้นตอนนี้จะนำข้อมูลทำ XOR (exclusive OR) กับ roundkey ที่ถูกสร้างมาก่อนแล้ว โดยเลือก roundkey ที่จะใช้กับข้อมูลตามตำแหน่งของข้อมูล และรอบของการทำงาน เช่น ข้อมูลที่แถวที่ 3 หลักที่ 2 และยังไม่เริ่มทำงานในวงรอบ จะใช้ roundkey แถวที่ 3 หลักที่ 2 (0*4+2) แต่ข้อมูลตำแหน่งเดียวกัน ที่ทำงานในวงรอบที่ 1จะใช้ roundkey แถวที่ 3 หลักที่ 6 (1*4+2) เป็นต้น

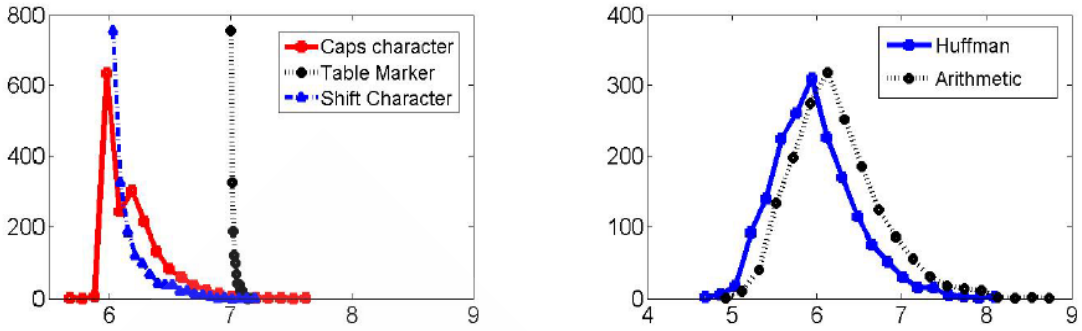
2.8.4 การเข้ารหัสแบบใหม่สำหรับการส่งข้อความหลายภาษาอย่างมีประสิทธิภาพ (New Encoding Schemes for Efficient Multilingual Text Messaging) ระบบการส่งข้อความ (SMS) โดยใช้โทรศัพท์มือเป็นสิ่งที่นิยมมาก อย่างไรก็ตามการเข้ารหัสในปัจจุบัน มีข้อจำกัดคือสามารถส่งข้อความตัวอักษรในภาษาอังกฤษได้เพียง 160 ตัวอักษรเท่านั้น แต่ในภาษาอื่นเช่นภาษาอินเดียสามารถส่งได้เพียงแค่ 70 ตัวอักษรเท่านั้น งานวิจัยนี้จึงเป็นการพัฒนาการส่งข้อความในภาษาอื่นให้ได้จำนวนมากขึ้น โดยพยายามส่งให้ถึง 160 ตัวอักษรต่อการส่งข้อความ 1 ครั้ง



ภาพที่ 2.15 การเข้ารหัสโดยใช้ Table marker algorithm



ภาพที่ 2.16 Path ของการส่งข้อความ



ภาพที่ 2.17 กราฟความถี่ของภาษาอังกฤษ

งานวิจัยนี้ทำให้ทราบว่าชนิดของข้อความมีประสิทธิภาพมากขึ้นจากการใช้อัลกอริทึมกับการใช้งาน Table marker การทำงานในส่วนต่างๆ ทั้งการเข้ารหัส ถอดรหัส มีความสอดคล้องกัน งานวิจัยนี้ทำให้เห็นถึงภาพแบบการเข้ารหัสที่สามารถลดจำนวนของข้อความลงได้ โดยไม่จำเป็นต้องทำการบีบอัดข้อมูลทุกครั้งที่ทำการส่งข้อความ

บทที่ 3
ระเบียบวิธีวิจัย

3.1 แนวทางการวิจัยและพัฒนา

งานวิจัยนี้ได้ทำการวิจัยพัฒนาการแปลงข้อมูลไบนารีในภาพแบบตัวอักษรให้มีคุณภาพมากขึ้น โดยใช้แนวคิดการลดปัญหาของขนาดไฟล์หลังจากการแปลงหรือบันทึกข้อมูลซึ่งการลดขนาดของข้อมูลนี้ ได้รับความสนใจในการพัฒนาการแปลงข้อมูลค่อนข้างน้อย ดังที่กล่าวไปแล้วในบทที่ 2 (สรุปงานวิจัยที่เกี่ยวข้อง) ซึ่งแนวคิดที่จะพัฒนาการลดขนาดของข้อมูลหลังจากบันทึกข้อมูลประเภทไบนารีนั้น ผู้วิจัยได้เลือกวิธีการเพิ่มวิธีการเข้ารหัสข้อมูลจากเดิมที่มีวิธีการเก็บข้อมูลแบบตัวอักษรส่วนใหญ่จะมีการใช้การเข้ารหัสอยู่ 2 วิธีการคือ การเข้ารหัสข้อมูลไบนารีแบบ Base32 และ Base64 การเข้ารหัสข้อมูลไบนารีทั้ง 2 แบบนี้ทำให้เราทราบว่า การเพิ่มจำนวนการเข้ารหัสจะทำให้ขนาดข้อมูลหลังจากการเข้ารหัสมีขนาดลดลงได้ แต่การใช้วิธีการนี้จะแลกมาด้วยระยะเวลาในการประมวลผลเพิ่มขึ้น ซึ่งจุดนี้เองเป็นจุดสำคัญสำคัญลำดับสองรองจากขนาดข้อมูลที่ลดลง เพราะหากข้อมูลลดลงแต่ใช้เวลานานเกินขอบเขตที่สามารถยอมรับได้ การพัฒนางานวิจัยนี้ถือว่ามีคุณภาพไม่เพียงพอไม่สามารถพัฒนาให้ไปถึงจุดที่สามารถใช้งานจริงได้

การวิเคราะห์การเพิ่มขนาดของการเข้ารหัสเป็นประเด็นที่สำคัญเนื่องจากต้องหาจุดสมดุลของระยะเวลาที่เพิ่มขึ้นด้วย กล่าวคือ หากเพิ่มการเข้ารหัสจาก Base64 เป็น Base256 หรือขนาดอื่น โดยอิงจากผลของการเพิ่มอักขระในการแปลงข้อมูลเพิ่มจะทำให้ขนาดของการแปลงข้อมูลลดลง แต่การอักขระในการแปลงไปถึง 256 อักขระนั้น เป็นใช้การวิธีการแปลงขนาดเดียวกับกับ BLOB ซึ่งวิธีการแบบ BLOB เองที่ใช้อักขระ 256 อักขระ มักพบปัญหาสำคัญคือ ข้อมูลหลังจากการจัดเก็บนั้นสูญหายได้ดังที่กล่าวมาแล้วในบทที่ 1 และเมื่อพิจารณาแบบ Base192 พบว่าวิธีการนี้ทำให้เกิดความยุ่งยากซับซ้อนมาก เนื่องจากอักขระที่ต้องใช้มาแทนมีจำนวนมากทำให้ไม่มีจำนวนอักขระที่เพียงพอ จึงอาจต้องนำวิธีการประมวลผลในเชิงคณิตศาสตร์มาช่วยในการประมวลผล จากปัญหาที่เกิดขึ้นทำให้โอกาสที่จะพัฒนาวิธีการนี้ในระยะเวลาอันใกล้มีความเป็นไปได้น้อย

ด้วยเหตุผลที่กล่าวมาทั้งหมดผู้วิจัยได้ทำการวิเคราะห์สังเคราะห์ที่ได้โดยเลือกใช้วิธีการ Base128 เพราะมีความเป็นไปได้ที่จะเกิดผลสำเร็จ โดยมุ่งเน้นการลดขนาดของข้อมูล และยังใช้เวลาในการเข้ารหัสและถอดรหัสอยู่ภายในระยะเวลาที่สามารถยอมรับได้ มาพัฒนาการแปลง

ข้อมูลไบนารีให้อยู่ในภาพแบบตัวอักษรเพื่อให้มีประสิทธิภาพมากขึ้น ช่วยลดขนาดของข้อมูลหลังการเข้ารหัสลงเมื่อเปรียบกับการเข้ารหัสแบบ Base32 และ Base64 และแก้ปัญหาพื้นที่ของการเก็บข้อมูลไม่เพียงพอหรือลดปัญหาบิ๊กดาต้า

3.1.1 ศึกษาและรวบรวมข้อมูล

1) จัดหา วิเคราะห์และศึกษาวิธีการจัดเก็บข้อมูลไบนารี่แบบ Blob Base32 และ Base64

2) ศึกษาวิธีการที่จะลดขนาดของข้อมูลไบนารี่ที่อยู่ในภาพแบบสตริง

3.1.2 การออกแบบวิธีการจัดเก็บโดยสร้างตารางจัดเก็บผลการทดลองในฐานข้อมูล MySQL

3.2 เครื่องมือที่ใช้ในการวิจัย

1) โปรแกรม Eclipse Editor เป็นโปรแกรมที่ใช้เป็นเครื่องมือในการเขียนโปรแกรมเพื่อทดสอบวิธีการ โปรแกรม MySQL ใช้ในการจัดเก็บข้อมูลและ Toad for MySQL ใช้สำหรับแสดงฐานข้อมูลที่จัดเก็บ

2) ภาษาที่ใช้ในการเขียนโปรแกรมคือ ภาษา JAVA และ สร้างระบบด้วย JSP

3.3 แผนการดำเนินงาน

3.3.1 จัดหาข้อมูลและเตรียมเครื่องมือในการพัฒนาโปรแกรม

3.3.2 ศึกษาข้อมูลที่ได้จัดเตรียมเพื่อเป็นแนวทางในการพัฒนาโปรแกรม

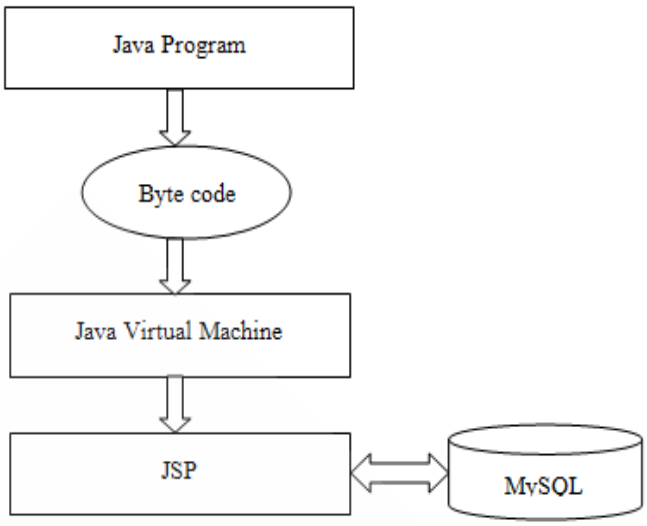
3.3.3 พัฒนาโปรแกรมตามวิธีการที่ได้ศึกษา

3.3.4 ทำการทดสอบแนวทางที่ได้ทำการพัฒนา

3.3.5 สรุปผลการทดลอง

3.4 การออกแบบ

งานวิจัยชิ้นนี้ได้มีการออกแบบให้ทำงานอยู่บนระบบเว็บแอพพลิเคชั่นโดยพัฒนาด้วยภาษา JSP (Java Server Pages) ให้สามารถใช้งานได้กับระบบปฏิบัติการทุกแพลตฟอร์ม (Platform) โดยสร้างจาวาคลาสเพื่อเป็นไลบารี่กลางในการเรียกใช้ ดังภาพที่ 3.1

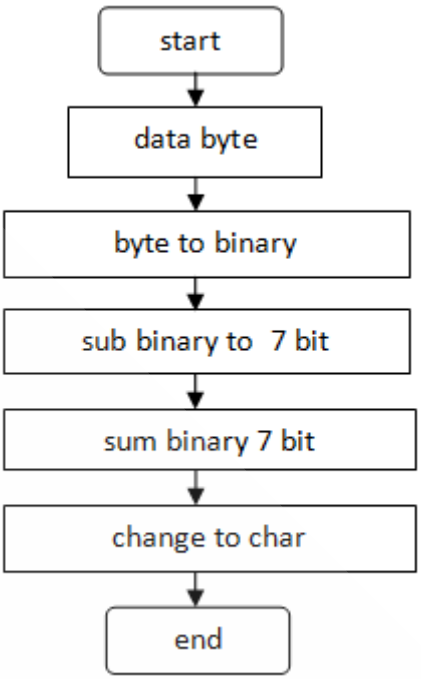


ภาพที่ 3.1 ภาพรวมของโปรแกรม

การทำงานบนเว็บเบราเซอร์ โดยการเลือกภาพหรือไฟล์โดยกดปุ่มอัพโหลด ระบบจะแสดงสถิติการเปรียบเทียบการเข้ารหัสระหว่าง Base32 Base64 และBase128

3.5 แนวคิดการทำงานของโปรแกรม

ก่อนที่จะทำการออกแบบผู้วิจัยมีแนวคิดที่จะจัดเก็บข้อมูลไบนารี่ในภาพแบบสตริง เทคนิคที่ใช้คือการแปลงข้อมูลชนิด Byte [4] เป็น binary ทำการตัด binary ครั้งละ 7 bit [6] แล้วแปลงค่าจาก binary เป็นเลขฐานสิบ และนำเลขฐานสิบไปแทนค่าด้วยอินเด็กซ์ของข้อมูลสตริง ดังภาพที่ 3.2



ภาพที่ 3.2 ขั้นตอนวิธีการทำงาน

ตารางที่ 3.1 อินเด็กซ์ของข้อมูลสตริง

Value	Char	Value	Char	Value	Char	Value	Char	Value	Char	Value	Char
0	A	23	X	46	u	69	*	92	ง	115	ฟ
1	B	24	Y	47	v	70	+	93	จ	116	ภ
2	C	25	Z	48	w	71	,	94	ฉ	117	ม
3	D	26	a	49	x	72	:	95	ช	118	ย
4	E	27	b	50	y	73	;	96	ซ	119	ร
5	F	28	c	51	z	74	<	97	ฌ	120	ล
6	G	29	d	52	1	75	>	98	ญ	121	ว
7	H	30	e	53	2	76	?	99	ฎ	122	ศ
8	I	31	f	54	3	77	@	100	ฏ	123	ษ
9	J	32	g	55	4	78	[	101	ฐ	124	ส
10	K	33	h	56	5	79	]	102	ฒ	125	ห

ตารางที่ 3.1 (ต่อ)

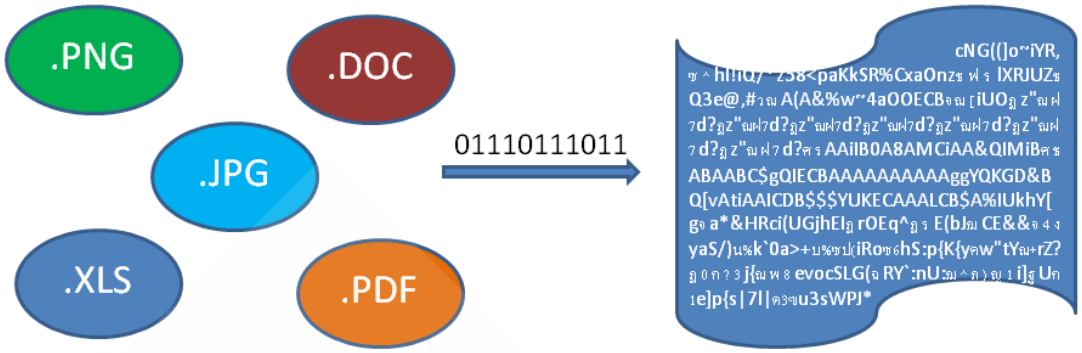
11	L	34	i	57	6	80	^	103	ณ	126	อ
12	M	35	j	58	7	81	_	104	ด	127	ฮ
13	N	36	k	59	8	82	{	105	ต		
14	O	37	l	60	9	83	|	106	ถ		
15	P	38	m	61	0	84	}	107	ท		
16	Q	39	n	62	!	85	~	108	ธ		
17	R	40	o	63	#	86	ก	109	น		
18	S	41	p	64	$	87	ข	110	บ		
19	T	42	q	65	%	88	ฃ	111	ป		
20	U	43	r	66	&	89	ค	112	ผ		
21	V	44	s	67	(	90	ฅ	113	ฝ		
22	W	45	t	68	)	91	ฆ	114	พ		

3.6 การแปลงข้อมูล

การแปลงข้อมูลในงานวิจัยนี้ เป็นส่วนสำคัญที่สุด ที่จะช่วยลดระยะเวลาในการจัดเก็บและช่วยลดขนาดของข้อมูลไบนารีที่ต้องจัดเก็บในฐานข้อมูล ดังตารางที่ 3.2

ตารางที่ 3.2 การแปลงข้อมูล

Bit pattern	0 1 1 0 0 0 1	0 0 1 0 1 0 1	1 1 0 1 1 1 0	0 1 1 0 0 1 0	0 0 0 0 1
Index	49	21	98	50	1
Base128-encoded	x	V	จ	y	B



ภาพที่ 3.3 แสดงผลของการแปลงข้อมูลไบนารี่

จากภาพที่ 3.3 เป็นการแสดงผลของการแปลงข้อมูลไบนารีให้อยู่ในภาพแบบตัวอักษร โดยใช้วิธีการของ Base128 จะเห็นว่าจากข้อมูลที่เป็น 0111011 จะถูกแทนที่ด้วยตัวอักษรตามตารางที่ 3.1 จากสายของข้อมูลที่เป็นไบนารีก็จะถูกแทนที่ด้วยตัวอักษร โดยจากข้อมูลแบบไบนารีที่อยู่ในภาพแบบรหัสแอสกี้จะถูกแปลงเป็นสายของข้อมูลบิท(Bit pattern) จากนั้นก็ทำการแบ่งกลุ่มบิทครั้งละครั้งละ 7 บิทเพื่อจะแทนที่ตัวอักษร 1 ตัวอักษรทำซ้ำกระบวนการจนครบตามความยาวของข้อมูล ผลลัพท์ที่ได้คือจากข้อมูลไบนารีก็จะถูกเข้ารหัสเป็นข้อมูลตัวอักษรแทน

3.7 ไฟล์ทดสอบ

การทดสอบประสิทธิภาพของระบบใช้ไฟล์เพื่อทดสอบระบบดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 ไฟล์ทดสอบ

ประเภทข้อมูล	ขนาดข้อมูล(KB)	ประเภทข้อมูล	ขนาดข้อมูล(KB)
bigDoc1.doc	4,652	mediumPdf2.pdf	148
bigDoc2.doc	1,222	smallPdf1.pdf	35
mediumDoc1.doc	723	smallPdf2.pdf	51
mediumDoc2.doc	877	bigPpt1.ppt	7,130
smallDoc1.doc	53	bigPpt2.ppt	5,157
smallDoc2.doc	77	mediumPpt1.ppt	1,222

ตารางที่ 3.3 (ต่อ)

ประเภทข้อมูล	ขนาดข้อมูล(KB)	ประเภทข้อมูล	ขนาดข้อมูล(KB)
bigJpg1.JPG	3,186	mediumPpt2.ppt	1,619
bigJpg2.JPG	2,640	smallPpt1.ppt	897
mediumJpg1.jpg	534	smallPpt2.ppt	354
mediumJpg2.jpg	760	bigXls1.xls	7,789
smallJpg1.jpg	5	bigXls2.xls	4042
smallJpg2.jpg	12	meduimXls1.xls	327
bigPdf1.pdf	4,952	meduimXls2.xls	726
bigPdf2.pdf	2,756	smallXls1.xlsx	12
mediumPdf1.pdf	241	smallXls2.xls	21

จากตารางคือไฟล์ที่ใช้ทดสอบวิธีการที่นำเสนอเท่านั้น สามารถที่จะใช้ไฟล์อื่นในการทดสอบนอกจากไฟล์ที่กล่าวมาได้

3.8 สมการที่ใช้วัดประสิทธิภาพ

การทดสอบประสิทธิภาพได้เลือกวิธีการทดสอบจากเวลาที่เครื่องคอมพิวเตอร์ตอบสนองต่อคำสั่งที่สั่งให้ทำ (Response Time) ซึ่งวิธีการพื้นฐานที่ใช้ในการวัดประสิทธิภาพ (System Performance) และ เวลาเฉลี่ยที่ CPU ใช้ Access ข้อมูลในตำแหน่งต่าง ๆ ของหน่วยความจำ (Memory access time) มีหน่วยวัดเป็น Nano second, Milli second เป็นต้นในเชิงสถาปัตยกรรมคอมพิวเตอร์

ประสิทธิภาพของระบบทดสอบโดยสมการคำนวณหาระยะเวลาที่ใช้ในการบันทึกข้อมูลมีดังนี้

$$t_w = t_e + t_i \qquad (1)$$

เมื่อ t_w คือ เวลาบันทึก

t_e คือ เวลาเข้ารหัสและแปลงข้อมูล

t_i คือ เวลาจัดเก็บลงฐานข้อมูล

สมการคำนวณหาเวลาที่ใช้ในการดึงข้อมูลกลับมา ประกอบด้วยเวลาของการถอดรหัส รวมกับเวลาของการเรียกข้อมูลจากฐานข้อมูลดังสมการ

$$t_r = t_q + t_d \qquad (2)$$

เมื่อ t_r คือ เวลาที่ใช้ในการดึงข้อมูลกลับมา

t_q คือ เวลาเรียกข้อมูลจากฐานข้อมูล

t_d คือ เวลาแปลงข้อมูลกลับ

สมการคำนวณหาเวลาที่ใช้ในการเข้ารหัสด้วยวิธีการ AES ประกอบด้วยเวลาของการเข้ารหัสรวมกับระยะเวลาที่ใช้ในการบันทึกข้อมูลดังสมการ

$$t_{AES} = t_{eAES} + t_i \qquad (3)$$

เมื่อ t_{AES} คือ เวลาที่ใช้ในการเข้ารหัสด้วยวิธีการ AES

t_{eAES} คือ เวลาที่ใช้ในการเข้ารหัสด้วยวิธีการ AES

t_i คือ เวลาจัดเก็บลงฐานข้อมูล

3.9 ตัวอย่างคำสั่ง

3.9.1 ตัวอย่างคำสั่ง JSP

- คำสั่งจัดเก็บข้อมูลไบนารี่ลงฐานข้อมูลแบบ BLOB

```
st=connection.prepareStatement("insert into FILE_DATA(FILE_NAME
,BASE_TYPE,file_blob,file_seq,file_no)"+"values(?,?,?,?,?)");//คำสั่ง insert
st.setString(1,ff.getName());//ชื่อไฟล์
st.setString(2,"blob");//ประเภทของภาพแบบจัดเก็บ
st.setBinaryStream(3, (InputStream)in, (int)(ff.length()));//ข้อมูลไบนารี่
st.setString(4,"1"); //ลำดับที่ของการจัดเก็บ
st.setInt(5, fileNo);//ครั้งที่ในการจัดเก็บ
int s = st.executeUpdate();
```

- คำสั่งจัดเก็บข้อมูลไบนารี่ลงฐานข้อมูลแบบ Base32

```
int initlength = 1024;//กำหนดขนาดข้อมูลเพื่อจัดเก็บลงฐานข้อมูล
int initlength1 = 1024;
int start = 0;
int j = 1;
String testBase32 = new Base32().encodeToString(baos); //แปลงข้อมูลไบนารี่เป็นข้อมูลสตริงในภาพแบบ Base32
int  total = (int) Math.ceil((double) testBase32.length() / (double) initlength);
for (int i = 0; i < total; i++) {//ลูปการบันทึกข้อมูลลงฐานข้อมูล
if(initlength1<testBase32.length()){
String strQuery1 = " INSERT INTO   FILE_DATA ( FILE_NAME  ,BASE_TYPE , FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','32','"+testBase32.substring(start, initlength1)+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();
}  else {
initlength1 = testBase32.length();
String strQuery1 =  " INSERT INTO   FILE_DATA ( FILE_NAME ,BASE_TYPE,FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','32','"+testBase32.substring(start, initlength1)+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();
break;
}
start = initlength1 ;
initlength1 = initlength1 + initlength;
j++;
st.close(); }
```

- คำสั่งจัดเก็บข้อมูลไบนารี่ลงฐานข้อมูลแบบ Base64

```
String testBase64 = Base64.encode(baos);
initlength = 1024;
initlength1 = 1024;
start = 0;
j = 1;
total = (int) Math.ceil((double) testBase64.length() / (double) initlength);
for (int i = 0; i < total; i++) {
if(initlength1<testBase64.length()){
String strQuery1 =  " INSERT INTO   FILE_DATA ( FILE_NAME
,BASE_TYPE,FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','64','"+testBase64.substring(start, initlength1)
+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();     }
else {
initlength1 = testBase64.length();
String strQuery1 =  " INSERT INTO   FILE_DATA (
FILE_NAME,BASE_TYPE,FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','64','"+testBase64.substring(start,
initlength1)+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();
break;
      }
start = initlength1 ;
initlength1 = initlength1 + initlength;
j++;
st.close();}
```

- คำสั่งจัดเก็บข้อมูลไบนารี่ลงฐานข้อมูลแบบ Base128

```
String testBase128 =  Base128.encode(baos);
```

```
initlength = 1024;
initlength1 = 1024;
start = 0;
j = 1;
total = (int) Math.ceil((double) testBase128.length() / (double) initlength);
for (int i = 0; i < total; i++) {
if(initlength1<testBase128.length()){
String strQuery1 =  " INSERT INTO   FILE_DATA (FILE_NAME,
BASE_TYPE,FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','128','"+testBase128.substring(start,
initlength1)+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();
} else {
initlength1 = testBase128.length();
String strQuery1 =  " INSERT INTO   FILE_DATA ( FILE_NAME,BASE_TYPE,
FILE_DTL,FILE_SEQ,FILE_NO) "+
"  VALUES ('"+ff.getName()+"','128','"+testBase128.substring(start,
initlength1)+"','"+j+"','"+fileNo+"')";
st = connection.prepareStatement(strQuery1);
st.executeUpdate();
break;
}   start = initlength1 ;
initlength1 = initlength1 + initlength;
j++;
st.close();
}
```

3.9.2 ตัวอย่างคำสั่ง JAVA ตัวอย่างเป็นตัวอย่างการเข้ารหัสด้วยภาพแบบ Bas128

```
public static String encode(byte[] baos) {//รับค่าเป็นข้อมูลไบนารี่
//ทำการแปลงข้อมูลจากไบนารี่ให้อยู่ในภาพแบบบิท
```

```
String [] bi =  new String[baos.length];
StringBuffer resultString = new StringBuffer();
StringBuffer results = new StringBuffer();
for (int i = 0; i < baos.length; i++) {
resultString.append(String.format("%8s",Integer.toBinaryString(baos[i] &
0xFF)).replace(' ', '0'));
 }
String mynewstring = resultString.toString();
System.out.println(mynewstring.substring(0, 32));
//ขั้นตอนการแบ่งข้อมูลบิทครั้งละ 7 บิทเพื่อแทนด้วยตัวอักษรตามตารางที่ 3.1
int initlength = 7;
int initlength1 = 7;
int start = 0;
int total = (int) Math.ceil((double) mynewstring.length() / (double) initlength);
for (int i = 0; i < total; i++) {
if(initlength1<mynewstring.length()){
String encodedImage12=  mynewstring.substring(start, initlength1);
int dec =0;
dec = Integer.parseInt(encodedImage12, 2);
results.append(ts.charAt(dec) );
}
else {
initlength1 = mynewstring.length();
String encodedImage12= mynewstring.substring(start, initlength1);
int dec =0;
dec = Integer.parseInt(encodedImage12, 2);
results.append(ts.charAt(dec) );
break;
}
start = initlength1 ;
```

```
initlength1 = initlength1 + initlength;
}
return results.toString();
}
```

3.9.3 ตัวอย่างคำสั่ง JAVA ตัวอย่างเป็นตัวอย่างการเข้ารหัสแบบ AES

```
public class AES {
static String IV = "AAAAAAAAAAAAAAAA";
static String plaintext = "test text 123\0\0\0"; /*Note null padding*/
static String encryptionKey = "0123456789abcdef";
public static void main(String [] args) {
try {
System.out.println("==Java==");
System.out.println("plain:   " + plaintext);
byte[] cipher = encrypt(plaintext, encryptionKey);
System.out.print("cipher:  ");
for (int i=0; i<cipher.length; i++)
System.out.print(new Integer(cipher[i])+" ");
System.out.println("");
String decrypted = decrypt(cipher, encryptionKey);
System.out.println("decrypt: " + decrypted);
} catch (Exception e) {
e.printStackTrace();}   }
public static byte[] encrypt(String plainText, String encryptionKey) throws
Exception {
Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");
SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"),
"AES");
cipher.init(Cipher.ENCRYPT_MODE, key,new IvParameterSpec
(IV.getBytes("UTF-8")));
return cipher.doFinal(plainText.getBytes("UTF-8"));  }
```

```
public static String decrypt(byte[] cipherText, String encryptionKey) throws
Exception{
Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");
SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"),
"AES");
cipher.init(Cipher.DECRYPT_MODE, key,new
IvParameterSpec(IV.getBytes("UTF-8")));
return new String(cipher.doFinal(cipherText),"UTF-8");
}
}
```

3.10 ฐานข้อมูลที่ใช้จัดเก็บผลการทดลอง

การจัดเก็บข้อมูลจะทำการจัดเก็บชื่อไฟล์ที่ในใช้ในกาทดสอบ ครั้งที่ ประเภทของการทดสอบและเวลาที่ใช้ในการประมวลผลของแต่ละวิธีการ

ตารางที่ 3.4 ฐานข้อมูลที่ใช้จัดเก็บผลการทดลอง

FILE_NAME	FILE_BLOB	FILE_BASE32	FILE_BASE64	FILE_BASE128	FILE_SEQ	FILE_TYPE
bigDoc1.doc	1402	18528	15025	17682	0	I
bigDoc2.doc	112	4077	3928	4040	1	I
mediumDoc1.doc	82	2550	1795	1713	2	I
mediumDoc2.doc	166	2376	2363	2736	3	I
smallDoc1.doc	2	229	121	122	4	I
smallDoc2.docx	189	209	166	172	5	I
bigJpg1.JPG	199	8932	8934	7661	6	I
bigJpg2.JPG	254	7902	7404	6545	8	I
mediumJpg1.jpg	49	1619	1329	1464	9	I
mediumJpg2.jpg	53	2152	1797	1856	10	I

ตารางที่ 3.4 (ต่อ)

FILE_NAME	FILE_BLOB	FILE_BASE32	FILE_BASE64	FILE_BASE128	FILE_SEQ	FILE_TYPE
smallJpg1.png	2	12	12	11	11	I
smallJpg2.jpg	11	48	30	40	12	I
bigPdf1.pdf	447	14095	13850	12454	13	I
bigPdf2.pdf	176	7990	8761	7138	14	I
mediumPdf1.pdf	15	651	528	603	15	I
mediumPdf2.pdf	12	362	334	338	16	I
smallPdf1.pdf	5	84	76	78	17	I
smallPdf2.pdf	0	158	110	114	18	I
bigPpt1.ppt	444	23998	19298	18379	19	I
bigPpt2.ppt	309	14464	13919	12609	20	I
mediumPpt1.ppt	86	3498	4327	3820	21	I
mediumPpt2.ppt	102	5276	4523	4078	22	I

3.11 ขั้นตอนการวัดประสิทธิภาพ

1. เลือกไฟล์ที่จะเปรียบเทียบประสิทธิภาพในการจัดเก็บ

2. ทำการกดปุ่ม อัพโหลดเมื่อกดปุ่มอัพโหลดระบบจะทำการแปลงไฟล์ข้อมูลไบนารี่ให้อยู่ในภาพแบบสตริงในภาพแบบวิธีการต่างๆ ระหว่างประมวลผลระบบก็จะทำการจัดเก็บเวลาของการแปลงข้อมูลและการจัดเก็บลงฐานข้อมูล ดังภาพที่ 3.2 ส่วนขั้นตอนการเขียนโปรแกรมเปรียบเทียบการจัดเก็บโดยผ่านการเข้ารหัสและขั้นตอนการเขียนโปรแกรมเปรียบเทียบการเรียกข้อมูลกลับคืนมาก็ใช้วิธีการทดสอบแบบเดียวกัน

1. เลือกไฟล์ที่ต้องการทดสอบ

2. ระบบทำการประมวลผลด้วยวิธีการต่าง ๆ

3. แสดงผลขนาดไฟล์ที่ผ่านการแปลงให้อยู่ในภาพแบบสตริง และเวลาที่ใช้ในการจัดเก็บลงฐานข้อมูล

```
----------------------------------------
FILE SIZE : 742912
Base32 SIZE :1188664
Base64 SIZE :990552
Base128 SIZE :849043
----------------------------------------
เวลาที่ใช้ในการจัดเก็บลงฐานข้อมูล :
BLOB :223
Base32 :3036
Base64 :1498
Base128 :2843
----------------------------------------
```

ภาพที่ 3.4 ขั้นตอนในการเปรียบเทียบการวัดประสิทธิภาพการจัดเก็บ

บทที่ 4
ผลการทดสอบระบบ

บทนี้จะแสดงผลการทดสอบระบบที่ผู้วิจัยได้ทำการพัฒนาการแปลงข้อมูลไบนารีให้อยู่ในภาพแบบตัวอักษรโดยการเข้ารหัสแบบ Base128 โดยผลการทดสอบนี้จะทำให้ทราบถึงข้อดีและข้อเสียของงานวิจัยนี้ได้ชัดเจนมากยิ่งขึ้นสามารถวิเคราะห์และประเมินผลลัพธ์ที่ได้จากการทดสอบนี้เพื่อนำไปพัฒนาให้ดียิ่งขึ้นหรือปรับปรุงแก้ไขในส่วนของข้อผิดพลาดที่เกิดขึ้นจากวิธีการนี้ต่อไป ซึ่งงานวิจัยนี้ได้ทำการออกแบบการทดสอบวิธีการด้วยการจำลองการใช้วิธีการผ่านเว็บเบราเซอร์ด้วยการเขียนวิธีการจากภาษาจาวา ผลการทดสอบจะแบ่งเป็น 5 แบบ

1. การทดสอบขนาดของข้อมูลที่ใช้ในการจัดเก็บ
2. การทดสอบเวลาที่ใช้ในการเข้ารหัสกับเวลาที่ใช้ในการบันทึกข้อมูลลงฐานข้อมูล
3. การทดสอบเวลาที่ใช้ในการถอดรหัสกับเวลาที่ดึงข้อมูลกลับคืนมา
4. การทดสอบเวลาในการเข้ารหัสแบบ AES

ทั้งนี้การทดสอบระบบได้ทำการเปรียบเทียบกับวิธีการอื่นคือ วิธีการแบบ BLOB ,Base32 และ Base64 โดยการทดสอบวิธีการนั้นได้ทำการทดสอบในสภาพแวดล้อมเดียวกัน เมื่อทำการทดสอบประสิทธิภาพของระบบ และทำการวิเคราะห์เปรียบเทียบกับข้อมูลอื่นผู้วิจัยได้ทำการเก็บข้อมูลเป็นเชิงสถิติเพื่อเปรียบเทียบและแสดงผลลัพธ์ให้ชัดเจนยิ่งขึ้น

4.1 ผลการทดสอบขนาดของข้อมูลที่ใช้ในการจัดเก็บ

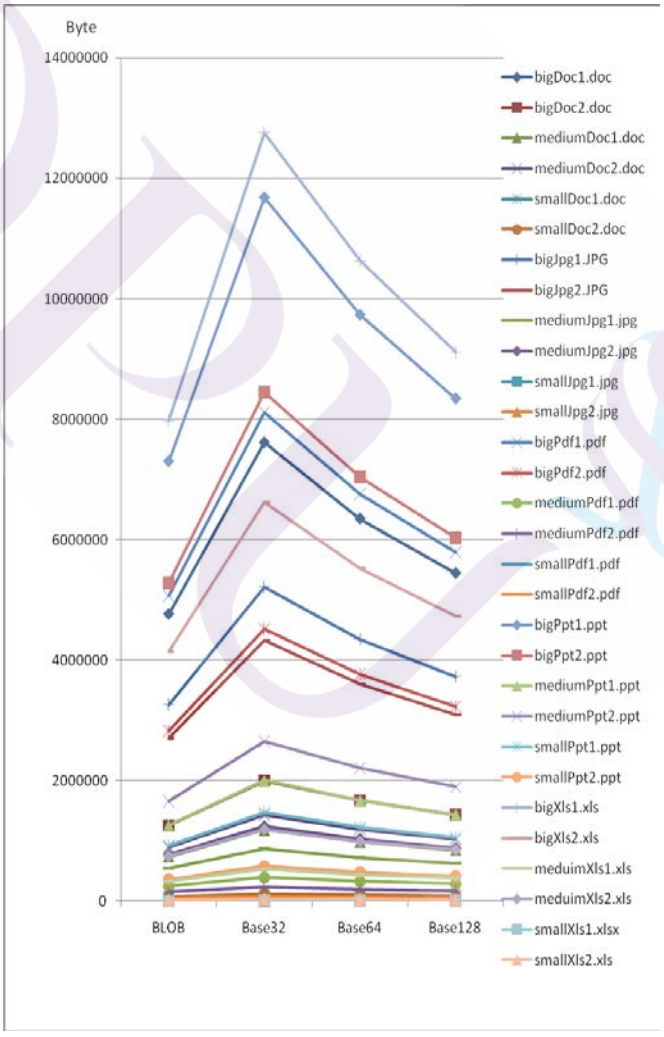
จากการทดสอบและใช้งานจริงกับชุดไฟล์ทดสอบดังแสดงในตารางที่ 4.1 ผ่านระบบที่จำลองขึ้นมาเพื่อทดสอบ โดยทำการทดสอบผ่านเว็บแอพพลิเคชั่นโดยเลือกไฟล์ข้อมูล ซึ่งผลการทดสอบขั้นต้นคือการวัดขนาดของข้อมูลที่ต้องทำการจัดเก็บลงในฐานข้อมูลเปรียบเทียบกับการเข้ารหัส และแปลงเป็นข้อมูลแบบสตริงวิธีต่างๆโดยได้ผลดังแสดงในภาพที่ 4.1 โดยจะพบว่าขนาดข้อมูลที่ใช้การเข้ารหัสและแปลงแบบ Base128 มีขนาดมากกว่าการจัดเก็บข้อมูลแบบ BLOB เพียงเล็กน้อย

ตารางที่ 4.1 ผลของขนาดข้อมูลในแต่ละไฟล์เมื่อผ่านการเข้ารหัสด้วยวิธีการต่างๆ

File Type	BLOB	Base32	Base64	Base128
bigDoc1.doc	4763136	7621018	6350848	5443584
bigDoc2.doc	1250653	2001045	1667538	1429318
mediumDoc1.doc	739840	1183744	986454	845532
mediumDoc2.doc	897536	1436058	1196715	1025756
smallDoc1.doc	54272	86836	72363	62026
smallDoc2.doc	74444	119111	99259	85079
bigJpg1.JPG	3261474	5218359	4348632	3727399
bigJpg2.JPG	2702772	4324436	3603696	3088883
mediumJpg1.jpg	546252	874004	728336	624288
mediumJpg2.jpg	777835	1244536	1037114	888955
smallJpg1.jpg	4917	7868	6556	5620
smallJpg2.jpg	12230	19568	16307	13978
bigPdf1.pdf	5069864	8111783	6759819	5794131
bigPdf2.pdf	2821845	4514952	3762460	3224966
mediumPdf1.pdf	246513	394421	328684	281730
mediumPdf2.pdf	150922	241476	201230	172483
smallPdf1.pdf	35471	56754	47295	40539
smallPdf2.pdf	51255	82008	68340	58578
bigPpt1.ppt	7301120	11681792	9734827	8344138
bigPpt2.ppt	5280768	8449229	7041024	6035164
mediumPpt1.ppt	1251328	2002125	1668438	1430090
mediumPpt2.ppt	1657856	2652570	2210475	1894693
smallPpt1.ppt	918016	1468826	1224022	1049162
smallPpt2.ppt	362496	579994	483328	414282
bigXls1.xls	7975424	12760679	10633899	9114771

ตารางที่ 4.1 (ต่อ)

File Type	BLOB	Base32	Base64	Base128
bigXls2.xls	4138496	6621594	5517995	4729710
meduimXls1.xls	334336	534938	445782	382099
meduimXls2.xls	742912	1188660	990550	849043
smallXls1.xlsx	11617	18588	15490	13277
smallXls2.xls	21504	34407	28672	24576



ภาพที่ 4.1 ผลของขนาดข้อมูลในแต่ละไฟล์เมื่อผ่านการเข้ารหัสด้วยวิธีการต่างๆ

4.2 ผลการทดสอบเวลาที่ใช้ในการแปลงและบันทึกข้อมูลลงฐานข้อมูล

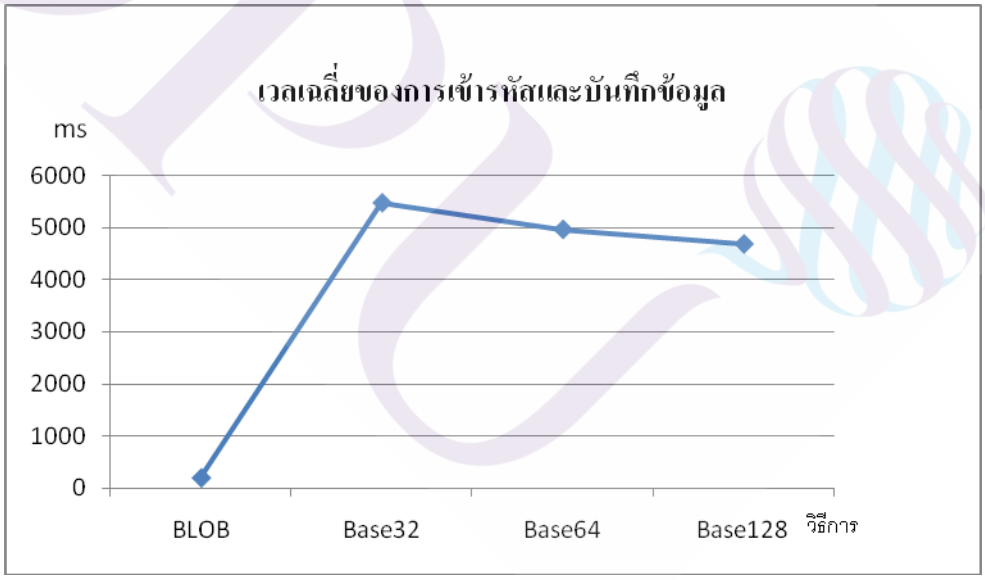
ผลการทดสอบที่ได้ในด้านของระยะเวลาที่ใช้ในการบันทึกข้อมูลดังสมการที่ (1) ผลที่ได้จากการทดสอบแสดงได้ดังภาพที่ 4.2 เมื่อนำผลมาเปรียบเทียบกันแล้ว จะเห็นได้ว่า วิธีการของ Base32 Base 64 ที่แกน X ใช้เวลาในการจัดเก็บน้อยกว่า Base128 เนื่องจากขั้นตอนในการเข้ารหัสของ Base128 มีขั้นตอนที่ซับซ้อนกว่า Base32 และ Base64 และการบันทึกข้อมูลแบบ BLOB ใช้เวลาน้อยสุดเนื่องจาก $t_e = 0$

ตารางที่ 4.2 ข้อมูลเวลาของการเข้ารหัสข้อมูลและบันทึกในแต่ละไฟล์

File Type	BLOB	Base32	Base64	Base128
bigDoc1.doc	1402	18528	15025	17682
bigDoc2.doc	112	4077	3928	4040
mediumDoc1.doc	82	2550	1795	1713
mediumDoc2.doc	166	2376	2363	2736
smallDoc1.doc	2	229	121	122
smallDoc2.doc	189	209	166	172
bigJpg1.JPG	199	8932	8934	7661
bigJpg2.JPG	254	7902	7404	6545
mediumJpg1.jpg	49	1619	1329	1464
mediumJpg2.jpg	53	2152	1797	1856
smallJpg1.jng	2	12	12	11
smallJpg2.jpg	11	48	30	40
bigPdf1.pdf	447	14095	13850	12454
bigPdf2.pdf	176	7990	8761	7138
mediumPdf1.pdf	15	651	528	603
mediumPdf2.pdf	12	362	334	338
smallPdf1.pdf	5	84	76	78
smallPdf2.pdf	0	158	110	114
bigPpt1.ppt	444	23998	19298	18379
bigPpt2.ppt	309	14464	13919	12609

ตารางที่ 4.2 (ต่อ)

File Type	BLOB	Base32	Base64	Base128
mediumPpt1.ppt	86	3498	4327	3820
mediumPpt2.ppt	102	5276	4523	4078
smallPpt1.ppt	56	2823	2373	2258
smallPpt2.ppt	23	1055	844	738
bigXls1.xls	1398	26600	22666	20625
bigXls2.xls	367	11037	11345	10434
meduimXls1.xls	18	1345	761	906
meduimXls2.xls	47	1875	2118	1770
smallXls1.xls	3	29	26	32
smallXls2.xls	48	137	66	94
Average Time	202.5667	5470.367	4960.967	4683.667



ภาพที่ 4.2 เวลาเฉลี่ยของการเข้ารหัสและบันทึกข้อมูลลงฐานข้อมูล

จากผลการทดสอบในด้านเวลาของการเก็บบันทึกข้อมูลเมื่อเปรียบเทียบในแต่ละวิธีการนั้นเราสามารถเห็นความแตกต่างของแต่ละวิธีการได้อย่างชัดเจน

4.3 ผลการทดสอบเวลาที่ใช้ในการถอดรหัสกับเวลาที่ดึงข้อมูลกลับคืนมา

ส่วนของการแปลงข้อมูลไบนารีมาเป็นข้อมูลสตริงนั้น ถือเป็นส่วนสำคัญในการพัฒนางานวิจัยนี้ ซึ่งปัญหาและความยากของการพัฒนาระบบการเก็บบันทึกข้อมูลไบนารีชนิดต่าง ๆให้มีประสิทธิภาพมากยิ่งขึ้นนั้น โดยมุ่งเน้นในเรื่องการลดขนาดของความจุข้อมูลลงฐานข้อมูลเป็นลำดับแรก

จากผลการทดสอบการแปลงข้อมูลไบนารีของงานวิจัยนี้ เมื่อเทียบกับวิธีการ Base32 โดยเฉลี่ยจะมีขนาดลดลง 30% ถ้าเทียบกับวิธีการ Base64 โดยเฉลี่ยจะมีขนาดลดลง 20% และถ้าเทียบกับการเก็บข้อมูลแบบ BLOB เฉลี่ยแล้วจะมีขนาดเพิ่มขึ้นจากเดิม 14% ถือว่ามีขนาดใกล้เคียงกันกับขนาดจริงของไฟล์ จะเห็นได้ว่า การพัฒนาให้ระบบสามารถย่อขนาดของข้อมูลให้ลดลงหรือคงขนาดของข้อมูลเดิมไว้นั้น เป็นส่วนที่กระทำได้ยากมาก ซึ่งทำให้พื้นที่ในการเก็บบันทึกข้อมูลไบนารีต่าง ๆ ลงในฐานข้อมูลมีขนาดใกล้เคียงกับขนาดจริงให้มากที่สุด แต่อย่างไรก็ตามด้วยวิธีการยังมีส่วนที่ต้องปรับปรุงต่อไป

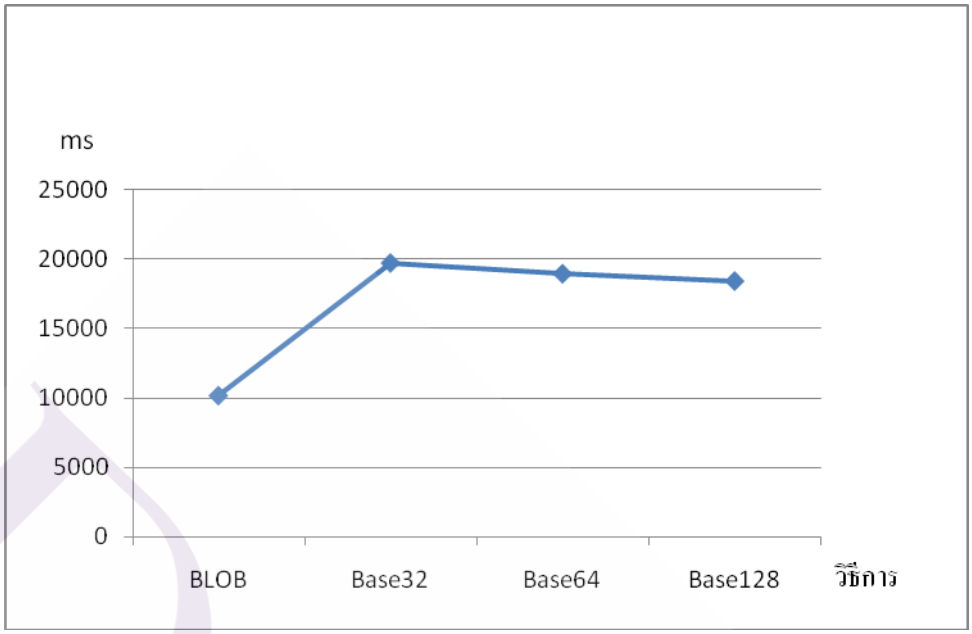
ในส่วนของการดึงข้อมูลกลับมาประกอบด้วยเวลาของการถอดรหัสรวมกับเวลาของการเรียกข้อมูลจากฐานข้อมูล ดัง สมการที่ (2) ผลที่ได้จากการทดสอบแสดงได้ดังภาพที่ 4.5

ตารางที่ 4.3 ข้อมูลเวลาของการดึงข้อมูลกลับมาในแต่ละไฟล์

File Type	BLOB	Base32	Base64	Base128
bigDoc1.doc	11023	39285	37157	37488
bigDoc2.docx	9994	17670	15291	15709
mediumDoc1.doc	10309	14452	14109	13196
mediumDoc2.doc	10528	15187	13959	14207
smallDoc1.doc	10267	10104	9742	10850
smallDoc2.docx	10110	10753	9731	10258
bigJpg1.JPG	9905	26712	25322	25722
bigJpg2.JPG	9756	22950	23715	21913
mediumJpg1.jpg	9250	12182	11983	12073
mediumJpg2.jpg	9480	13566	13542	12968
smallJpg1.png	9465	11126	10775	10198

ตารางที่ 4.3 (ต่อ)

File Type	BLOB	Base32	Base64	Base128
smallJpg2.jpg	9695	10002	9740	10166
bigPdf1.pdf	10379	37045	35500	34526
bigPdf2.pdf	10395	24741	22879	23262
mediumPdf1.pdf	9655	11856	10252	10564
mediumPdf2.pdf	9439	11078	10983	10402
smallPdf1.pdf	11471	10809	10632	10339
smallPdf2.pdf	9545	9975	10709	10054
bigPpt1.ppt	10652	48417	46293	43970
bigPpt2.ppt	12316	39971	37106	33218
mediumPpt1.ppt	10578	16703	16292	17924
mediumPpt2.ppt	10331	19123	18227	18392
smallPpt1.ppt	10185	14998	13835	14536
smallPpt2.ppt	9600	12387	11747	11361
bigXls1.xls	11018	51021	47720	46501
bigXls2.xls	9903	30571	31290	27072
meduimXls1.xls	9904	11851	14265	11077
meduimXls2.xls	10286	14354	13904	13025
smallXls1.xls	10380	10376	10399	10030
smallXls2.xls	9713	10996	10304	10542
Average Time	10184.4	19675.367	18913.433	18384.767



ภาพที่ 4.3 เวลาเฉลี่ยที่ใช้ในการดึงและแปลงข้อมูลกลับมา

จากภาพแสดงให้เห็นว่าเวลาในการดึงข้อมูลของวิธีการ BLOB ใช้เวลาน้อยสุด ส่วนวิธีการ Base128 ใช้เวลาในการดึงข้อมูลกลับได้ใกล้เคียงกับเวลาของ Base64 และวิธีการแบบ Base32 ใช้เวลามากที่สุด

4.4 ผลการทดสอบเวลาในการเข้ารหัสแบบ AES

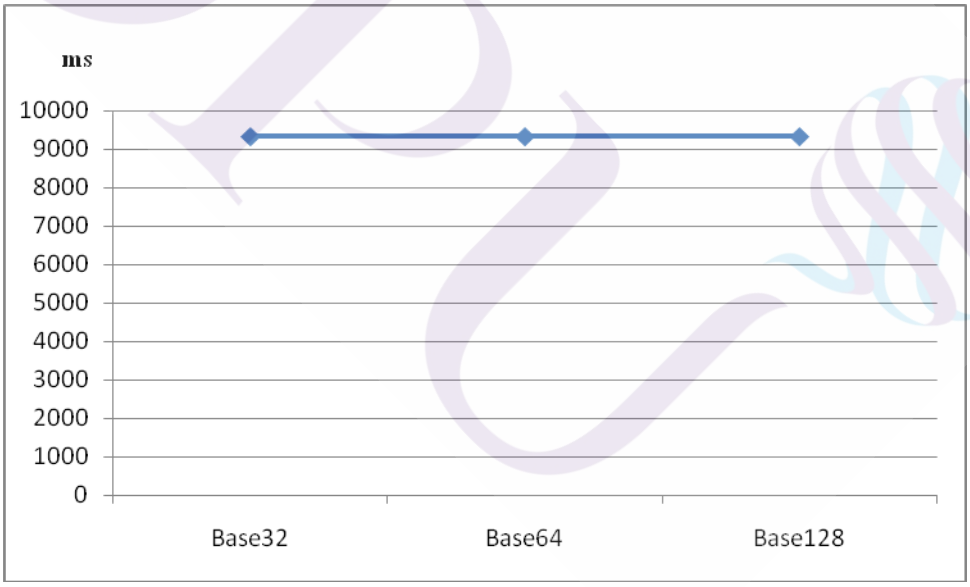
ในส่วนการทดสอบเวลาในการเข้ารหัสแบบ AES ประกอบด้วยเวลาของการเข้ารหัสแบบ AES รวมกับเวลาของบันทึกข้อมูลจากฐานข้อมูล ดัง สมการที่ (3) ผลที่ได้จากการทดสอบแสดงได้ดังภาพที่ 4.4

ตารางที่ 4.4 ข้อมูลเวลาการเข้ารหัสในแต่ละไฟล์ในแต่ละไฟล์

File Type	Base32	Base64	Base128
bigDoc1.doc	12693	11361	11151
bigDoc2.doc	12640	10809	10348
mediumDoc1.doc	10792	10568	10994
mediumDoc2.doc	10658	10691	10353
smallDoc1.doc	10474	10139	10622
smallDoc2.doc	9995	10307	10398
bigJpg1.JPG	11215	11484	10695
bigJpg2.JPG	10018	9778	10235
mediumJpg1.jpg	10177	10213	9747
mediumJpg2.jpg	10171	9927	10809
smallJpg1.jng	9669	10814	10573
smallJpg2.png	9857	10475	10218
bigPdf1.pdf	11283	11065	10755
bigPdf2.pdf	10804	10102	10686
mediumPdf1.pdf	9656	10208	10648
mediumPdf2.pdf	10644	10174	10047
smallPdf1.pdf	9680	11119	11095
smallPdf2.pdf	10942	10755	9785
bigPpt1.ppt	11466	11776	11766
bigPpt2.ppt	11043	10876	11171
mediumPpt1.ppt	10478	10311	10983
mediumPpt2.ppt	10270	10421	9905

ตารางที่ 4.4 (ต่อ)

File Type	Base32	Base64	Base128
smallPpt1.ppt	9780	10308	9863
smallPpt2.ppt	9685	9997	10707
bigXls1.xls	10503	10397	10041
bigXls2.xls	11731	11220	10337
meduimXls1.xls	9999	10824	13297
meduimXls2.xls	10325	11153	10361
smallXls1.xls	9784	10207	10087
smallXls2.xls	19756	9416	9644
Average Time	10872.933	10872.933	10872.933



ภาพที่ 4.4 เวลาเฉลี่ยในการเข้ารหัสแบบ AES

จากภาพแสดงให้เห็นว่าเวลาการเข้ารหัสแบบ AES นั้นในแต่ละวิธีการใช้เวลาใกล้เคียงกันโดยการทดสอบไม่ได้นำวิธีแบบBLOB มาเปรียบเทียบด้วยเนื่องจากวิธีการที่ใช้ในการเข้ารหัสใช้วิธีการเข้ารหัสข้อมูลที่อยู่ในภาพแบบตัวอักษร

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

ในบทนี้จะเป็นการอภิปรายเพื่อสรุปผลที่ได้จากการทดสอบงานวิจัย รวมทั้งข้อจำกัดของระบบที่พบจากการทดสอบระบบ และข้อเสนอแนะสำหรับแนวทางในการพัฒนางานวิจัยนี้ต่อไปเพื่อแก้ข้อบกพร่องของระบบให้มีประสิทธิภาพมากขึ้น

5.1 สรุปผลการทดลอง

วิธีการแปลงเพื่อบันทึกข้อมูลปัจจุบันมีวิธีการอยู่ 3 วิธีการคือ วิธีการแบบ BLOB ,Path และวิธีการเก็บแบบตัวอักษร โดยแต่ละวิธีมีข้อดีและข้อเสียแตกต่างกันดังที่กล่าวในบทที่ 1 (บทนำ) ซึ่งจากปัญหาจากการแปลงแบบตัวอักษรคือการแปลงแล้วได้ขนาดใหญ่กว่าข้อมูลจริงทำให้ไม่นิยมใช้เหมือนกับวิธีอื่น งานวิจัยฉบับนี้ผู้วิจัยได้ทำการศึกษาและพัฒนาจุดอ่อนสำคัญของการวิธีการแบบตัวอักษร โดยพยายามทำให้ข้อมูลหลังจากทำการบันทึกแล้วมีขนาดใกล้เคียงกับข้อมูลจริงที่สุด หรือไม่มากกว่าข้อมูลก่อนทำการบันทึกจนเกินไป เพื่อตอบโจทย์การมีพื้นที่การบันทึกข้อมูลบนเซิร์ฟเวอร์อย่างจำกัดและปริมาณการใช้งานหรือบันทึกบนเซิร์ฟเวอร์ที่มีเพิ่มขึ้น โดยไม่ส่งผลกระทบอื่น ๆ ที่ทำให้ระบบเกิดความผิดพลาด เช่น ระยะเวลาในการประมวลผลไม่มากจนเกินไปหรือไม่นานจนเกินไป หรือเวลาที่เรียกข้อมูลกลับคืนมาไม่มากจนเกินไป

งานวิจัยนี้ได้เลือกใช้วิธีการเข้ารหัสข้อมูลไบนารีแบบ Base128 อันเนื่องจากมีอักขระมากกว่า Base64 ถึง 2 เท่าและมีไม่มากจนเกินไปจนไม่สามารถหาอักขระมาใช้ในการแทนที่ข้อมูลไบนารีได้ การใช้ Base128 มีโอกาสทำให้ข้อมูลลดลง เมื่อเปรียบเทียบกับ Base64 ผู้วิจัยได้ทำการพิจารณาว่าโอกาสที่จะทำให้ข้อมูลลดลงนั้นมีโอกาสสูง ทั้งนี้ผู้วิจัยได้พิจารณา Base256 แล้วว่าจำนวนตัวอักษรที่ใช้ในการแทนข้อมูลไบนารีมีเท่ากับ BLOB ซึ่งทำให้เกิดความเสียหายของข้อมูลได้ดังที่กล่าวมาแล้วในบทที่ 3 แต่จะใช้ Base192 หรืออื่น ๆ ที่มากกว่าก็จะทำให้ไม่มีอักขระเพียงพอที่จะใช้ในการแทนที่ข้อมูลไบนารีได้ ผู้วิจัยได้ทำการประเมินแล้วว่าโอกาสที่จะพัฒนา Base128 นั้นมีความเป็นไปได้ที่สุดและมีผลดีที่สุดที่สามารถให้ข้อมูลลดลงได้ โดยการมีการใส่ตัวอักขระภาษาไทย อังกฤษพิมพ์ใหญ่ อังกฤษพิมพ์เล็ก ตัวเลข สัญลักษณ์ต่างๆ เพื่อให้ครบ 128 ตัวอักษร ในการพัฒนาวิธีการผู้วิจัยได้เลือกใช้ภาษาจาวาในการพัฒนา เนื่องจากเป็นภาษาที่สามารถรองรับได้ทุกระบบปฏิบัติการ

การวิเคราะห์ผลจากผลงานวิจัยขนาดของเวลาที่ลดลงเมื่อได้จากการเข้ารหัสและถอดรหัส เนื่องจากการใช้ตัวอักขระในการแทนที่ข้อมูลไบนารีมีจำนวนเพิ่มขึ้น ขนาดข้อมูลมีขนาดลดลงมากกว่าวิธีการแบบ BASE32, BASE64 และมากกว่าวิธีการแบบ BLOB อยู่ 14 % และเวลาที่ใช้ในการเข้ารหัสและถอดรหัสลดลงเมื่อเทียบกับวิธีการแบบ BASE32 และ BASE64 โดยทดสอบที่สภาวะแวดล้อมเดียวกัน เนื่องจากตัวอักษรที่ใช้ในการแทนที่มีจำนวนเพิ่มขึ้น จากผลของการพิจารณาของงานวิจัยชิ้นนี้ เป็นไปตามเป้าหมายที่กำหนด ทำให้งานวิจัยชิ้นนี้ดัดแปลงหรือพัฒนาต่อยอดได้

5.2 ข้อจำกัดของระบบที่พบจากการทดสอบระบบ

ข้อจำกัดในเรื่องของการทดสอบระบบที่ใช้เวลาในการทดสอบนานเนื่องจากข้อจำกัดของอุปกรณ์ที่ใช้ในการทดสอบ เทคนิคการเขียนโปรแกรมที่ไม่มีประสิทธิภาพมากพอ

5.3 ข้อเสนอแนะ

งานวิจัยที่จะพัฒนาต่อไปคือการทดสอบประสิทธิภาพการค้นหาข้อมูลในระบบฐานข้อมูลที่มีการแปลงเป็นข้อมูลสตริงด้วยรหัส BASE64 BASE128 ซึ่งรวมทั้งวิธีการเข้ารหัสข้อมูลหลังทำการแปลงและเข้ารหัสแล้ว [5],[7] ก่อนการจัดเก็บข้อมูล

บรรณานุกรม

บรรณานุกรม

ภาษาไทย

ศุภชัย ทองสุข. (2557) *การท ารหัสลับแบบ AES บนหน่วยประมวลผลหลายแกน.* สืบค้น 10 มีนาคม 2557, จาก www.cp.eng.chula.ac.th/~piak/thesis/supachai_complete_thesis2.pdf

ภาษาอังกฤษ

Changeset 5814. (2014) *Base64 utility methods.* section 6.2,. Available from : http://www.w3.org/html/wg/drafts/html/master/webappapis.html#atob

Congfu Xu, Yafang Chen, Kevin Chiew. (2010) *An Approach to Image Spam Filtering Based on Base64 Encoding and N-Gram Feature Extraction.* International Conference on Tools with Artificial Intelligence. pp. 171-176,.

Dev. (2006). *base64 Image.* Available from : http://base64image.org/ [2013,January 15].

Gurpreet Singh, Supriya, (2013) *Modified Vigenere Encryption Algorithm and its Hybrid Implementation with Base64 and AES.* International Conference on Advanced Computing, Networking and Security,pp. 232-237, 2013.

Mackenzie, Charles E, (1980). *ASCII.* Available from : http://en.wikipedia.org/ wiki/ASCII [2013, December 10].

S. Josefsson, (2006) *The Base16, Base32, and Base64 Data Encodings.* Availablefrom : http://www.ietf.org/rfc/rfc4648.txt[2013,December 15].

Zhenxing Liu , Lu Liu , Richard Hill1, Yongzhao Zhan. (2011) *Base62x: An Alternative Approach to Base64 for non-Alphanumeric Characters.* Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD).pp. 2667-2670.

ภาคผนวก

ภาคผนวก ก

บทความที่ได้รับการนำเสนอในที่ประชุม

The 10th National Conference on Computing and Information Technology

Proceedings of NCCIT 2014

The 10th National Conference on Computing and Information Technology

8th-9th May 2014
Anusana Laguna Phuket, Thailand

www.nccit.net

Faculty of Information Technology
King Mongkut's University of Technology North Bangkok

บทความวิจัย

การประชุมทางวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ครั้งที่ 10

8-9 พฤษภาคม 2557
โรงแรมอนันตรา ลากูน่า ภูเก็ต

คณะเทคโนโลยีสารสนเทศ

Friday May 9th, 2014		
NCCIT2014 Room 1		
Machine Learning in Prediction		
Time	**Title/Author**	**Page**
09.00-09.15 NCCIT2014-269	**Interval Type-2 Fuzzy Systems Trained by Hybrid Heuristic Algorithm for Stock Exchange of Thailand** *Adisak Sangsongfa and Phayung Meesad*	450
09.15-09.30 NCCIT2014-195	**Forecasting Agricultural Product Prices using Moving Average and Artificial Neural Network Techniques** *Saowaluk Pinsakmanaboon and Pudsadee Boonsrimuang*	456
09.30-09.45 NCCIT2014-149	**Apply Linear Regression Analysis to Estimate Leaf Area** *Saroopsiri Thapanand, Barames Onkaewtaf and Mahasak Ketcham*	462
09.45-10.00 NCCIT2014-56	**Support Vector Machines For Derivatives Price Prediction** *Siam Pimpun and Boonsam Narasril*	466
10.00-10.15 NCCIT2014-24	**Stock Forecasting with Candlestick Chart and Hidden Markov Models** *Supakit Nootyaskool, Monchai Thanapanna and Wichai Puttasakhalid*	472
10.15-10.45	*Coffee Break*	
Data mining Application		
10.45-11.00 NCCIT2014-227	**Spatial-Temporal Analysis and Visualization of Twitter Data on Google Earth** *Sompon Vejchasarn and Veera Muangsin*	478
11.00-11.15 NCCIT2014-152	**Complex Networks and Online VDO Classification** *Suwanan Jongpo, Nawalop Thongpoo, Sawaeng Datchuay and Nawapon Wongpongjun*	484
11.15-11.30 NCCIT2014-159	**Hand Written Thai Character Recognition for Mobile Phone** *Thongpong Rawkhon and Porchet Thaungtanan*	491
11.30-11.45 NCCIT2014-37	**A Feature - Oriented Traceability for Software Product Line Evolution using Vector Space Model** *Ketoka Tipsuksa and Somsak Vannakorn*	497
11.45-12.00 NCCIT2014-142	**Storing Binary Data to Database using Base128 encoding** *Chaisiphon Prayoonrat and Chaiyaphon Siriamphaiboon*	504
12.00-13.00	*Lunch*	

[illegible] Natural Resources and Environmental Management and Information Technology [illegible]

การจัดเก็บข้อมูลไบนารีในฐานข้อมูลในรูปแบบ Base128
Storing Binary Data to Database using Base128 encoding

[illegible]

[illegible]

บทคัดย่อ

[illegible]

[illegible]

[illegible]

คำสำคัญ: Base128, Base64, BLOB, Database

Abstract

[illegible]

[illegible]

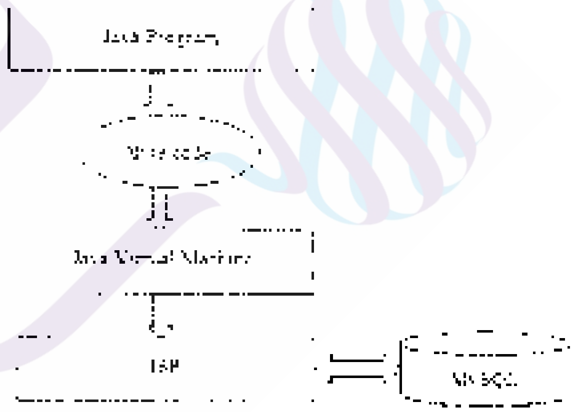
Keyword: Base128, Base64, BLOB, Database

1. บทนำ

[illegible]

2. งานวิจัยที่เกี่ยวข้อง

3. ระเบียบวิธีวิจัย



ภาพที่ 1 [illegible]

[illegible]

เมื่อ [illegible] คือ จดทับที่ n

[illegible] คือ ตำแหน่งข้อมูลในข้อมูล

[illegible] คือ ตัวอักษรในชุดข้อมูล

และสมการดังกล่าวจะแสดงการทำให้ได้ในการเข้ารหัสข้อมูลแต่ละตัวอักษรในชุดข้อมูลที่ต้องการ

[illegible]

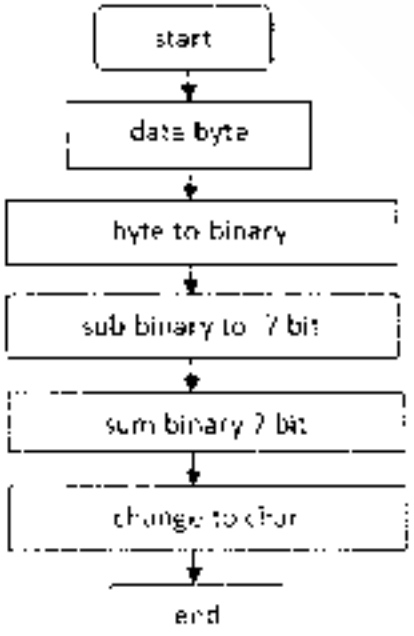
เมื่อ [illegible] คือ ตัวอักษรที่ใช้ในการถอดรหัสข้อมูลนั้น

[illegible] คือ ตำแหน่งข้อมูลในชุดข้อมูล

[illegible] คือ ตัวอักษรในชุดข้อมูล

3.3 การเข้ารหัส

การเข้ารหัสนี้จะนำข้อมูลที่ต้องการเข้ารหัสมาแปลงเป็นรหัสไบนารี่ ดังรูปที่ 2 ซึ่งจะนำข้อมูลที่ต้องการเข้ารหัสมาแปลงเป็นรหัสไบนารี่ 8 บิต จากนั้นนำรหัสไบนารี่ 8 บิตมาแปลงเป็นรหัสไบนารี่ 7 บิต แล้วนำรหัสไบนารี่ 7 บิตมาแปลงเป็นตัวอักษรตามตารางที่ 2



รูปที่ 2 ขั้นตอนการเข้ารหัส

ตารางที่ 1 ตัวอักษรในข้อมูลชุดหลัก

Value	Char	Value	Char	Value	Char	Value	Char	Value	Char
0	A	27	b	54	2	81	)	108	l
1	B	28	c	55	3	82	*	109	m
2	C	29	d	56	4	83	+	110	n
3	D	30	e	57	5	84	,	111	o
4	E	31	f	58	6	85	-	112	p
5	F	32	g	59	7	86	.	113	q
6	G	33	h	60	8	87	/	114	r
7	H	34	i	61	9	88	:	115	s
8	I	35	j	62	;	89	;	116	t
9	J	36	k	63	<	90	=	117	u
10	K	37	l	64	>	91	?	118	v
11	L	38	m	65	@	92	[	119	w
12	M	39	n	66	[	93	]	120	x
13	N	40	o	67	\	94	^	121	y
14	O	41	p	68	]	95	_	122	z
15	P	42	q	69	^	96	`	123	{
16	Q	43	r	70	_	97	a	124	|
17	R	44	s	71	`	98	b	125	}
18	S	45	t	72	a	99	c	126	~
19	T	46	u	73	b	100	d	127	
20	U	47	v	74	c	101	e		
21	V	48	w	75	d	102	f		
22	W	49	x	76	e	103	g		
23	X	50	y	77	f	104	h		
24	Y	51	z	78	g	105	i		
25	Z	52	{	79	h	106	j		
26	a	53	|	80	i	107	k		

3.4 การแปลข้อมูล

การแปลงข้อมูลในการวิจัยนี้ เป็นส่วนสำคัญที่สุด เพราะจะต้องมีการแปลงข้อมูลในการเข้ารหัสและถอดรหัส ดังรูปที่ 3 ซึ่งจะแสดงตัวอย่างข้อมูลในการเข้ารหัสและถอดรหัสตามตารางที่ 2

ตารางที่ 2 การแปลงข้อมูล

[illegible]															
[illegible]															
[illegible]															

3.5 ไฟล์ทดสอบ

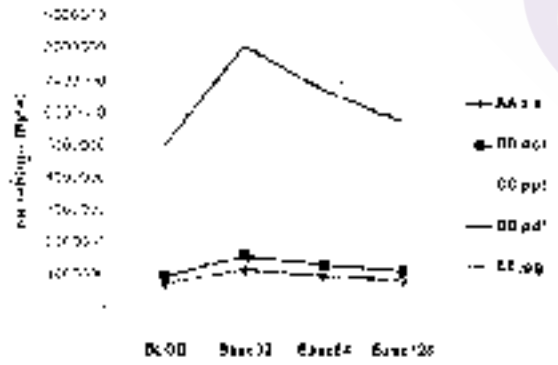
การทดสอบการประสิทธิภาพของการใช้ไฟล์ ที่ทดสอบ ขนาดต่างกัน ดังนี้

ตารางที่ 3 ไฟล์ทดสอบ

ประเภทข้อมูล	ขนาดข้อมูล
AA.xls	256 KB
BB.doc	962 KB
CC.ppt	2,774 KB
DD.pdf	1952 KB
EE.jpg	760 KB

4. ผลการทดลอง

การทดลองการใช้งานเครื่องมือวิเคราะห์ประสิทธิภาพของการเข้ารหัสและถอดรหัสข้อมูล ด้วยวิธีการเข้ารหัสแบบสมมาตร ได้แก่ Base32 Base64 และ Base128 ซึ่งเป็นการทดสอบเพื่อวิเคราะห์ประสิทธิภาพของการเข้ารหัสและถอดรหัสข้อมูลในรูปแบบต่างๆ ที่ใช้ในการทดสอบ ดังตารางที่ 3 โดยผลการทดสอบจะแสดงในรูปของกราฟเส้น ดังภาพที่ 3 ซึ่งแสดงผลการทดสอบการเข้ารหัสข้อมูลด้วย Base32 Base64 และ Base128 ตามลำดับ



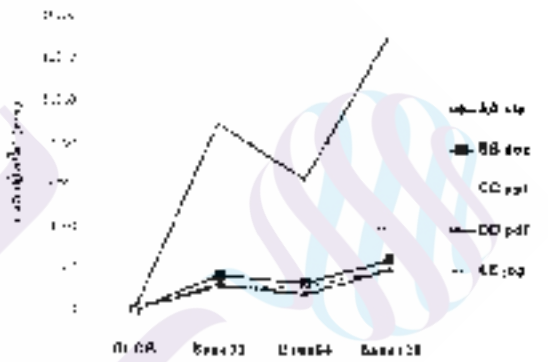
ภาพที่ 3 ผลการทดสอบข้อมูลที่ใช้ในการเข้ารหัส

จากผลการทดสอบการเข้ารหัสข้อมูลด้วยวิธีการต่างๆ ดังกล่าวข้างต้น ผู้วิจัยได้นำข้อมูลผลการทดสอบมาวิเคราะห์ เพื่อหาค่าเฉลี่ยของการเข้ารหัสข้อมูลในแต่ละวิธีการ ดังตารางที่ 4 ซึ่งแสดงผลการทดสอบการเข้ารหัสข้อมูลด้วยวิธีการต่างๆ ดังกล่าวข้างต้น ผู้วิจัยได้นำข้อมูลผลการทดสอบมาวิเคราะห์ เพื่อหาค่าเฉลี่ยของการเข้ารหัสข้อมูลในแต่ละวิธีการ ดังตารางที่

4.1 ส่วนของการบันทึก

ผลการทดสอบที่ได้ในส่วนของการบันทึกข้อมูลแสดงดังตารางที่

ผลที่ได้จากการทดสอบแสดงให้เห็นว่าวิธีการของ Base32 Base64 ที่ขนาด X ใช้เวลาในการเข้ารหัสน้อยกว่า Base128 เนื่องจากขั้นตอนในการเข้ารหัสของ Base128 มีขั้นตอนที่ซับซ้อนกว่า Base32 และ Base64 และการที่จะถอดรหัสข้อมูลจาก Base32 ได้ต้องใช้ข้อมูลเนื้อหาของ



ภาพที่ 4 กราฟที่ใช้ในการแสดงผลการเข้ารหัสข้อมูลด้วยวิธีการต่างๆ

จากผลการทดสอบในส่วนของการเข้ารหัสข้อมูลนั้น ผู้วิจัยได้นำข้อมูลมาวิเคราะห์ เพื่อหาค่าเฉลี่ยของการเข้ารหัสข้อมูลในแต่ละวิธีการ ดังตารางที่

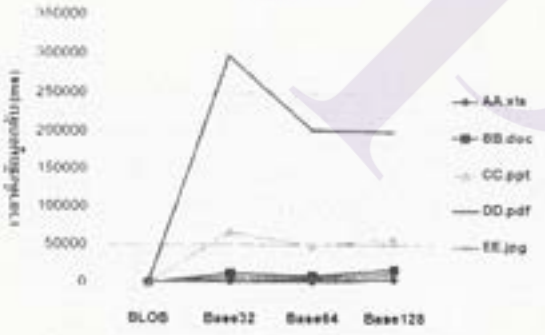
4.2 ส่วนของการแปลงข้อมูล

ส่วนของการแปลงข้อมูลในการเข้ารหัสข้อมูลนั้น ผู้วิจัยได้นำข้อมูลมาวิเคราะห์ เพื่อหาค่าเฉลี่ยของการเข้ารหัสข้อมูลในแต่ละวิธีการ ดังตารางที่ ซึ่งแสดงผลการทดสอบการเข้ารหัสข้อมูลด้วยวิธีการต่างๆ ดังกล่าวข้างต้น ผู้วิจัยได้นำข้อมูลมาวิเคราะห์ เพื่อหาค่าเฉลี่ยของการเข้ารหัสข้อมูลในแต่ละวิธีการ ดังตารางที่

จากผลการทดสอบการแปลงข้อมูลไบนารีของงานวิจัยนี้ เมื่อเทียบกับวิธีการ Base32 โดยเฉลี่ยจะมีขนาดลดลง 30% ถ้าเทียบกับวิธีการ Base64 โดยเฉลี่ยจะมีขนาดลดลง 20% และถ้าเทียบกับการเก็บข้อมูลแบบ BLOB เฉลี่ยแล้วจะมีขนาดเพิ่มขึ้นจากเดิม 14% ถือว่ามีขนาดใกล้เคียงกันกับขนาดจริงของไฟล์ จะเห็นได้ว่า การพัฒนาให้ระบบสามารถย่อขนาดของข้อมูลให้ลดลง หรือคงขนาดของข้อมูลเดิมไว้นั้น เป็นส่วนที่กระทำได้ยากมาก ซึ่งทำให้พื้นที่ในการเก็บบันทึกข้อมูลไบนารีต่าง ๆ ลงในฐานข้อมูลมีขนาดใกล้เคียงกับขนาดจริงให้มากที่สุด แต่อย่างไรก็ตามด้วยวิธีการยังมีส่วนที่ต้องปรับปรุงต่อไป

4.3 ส่วนของการดึงข้อมูลกลับมา

ในส่วนของการดึงข้อมูลกลับมาประกอบด้วยเวลาของการถอดรหัสรวมกับเวลาของการเรียกข้อมูลจากฐานข้อมูล ดังสมการที่ (2) ผลที่ได้จากการทดสอบแสดงได้ดังภาพที่ 5



ภาพที่ 5: เวลาที่ใช้ในการดึงและแปลงข้อมูลกลับมา

จากภาพแสดงให้เห็นว่าเวลาในการดึงข้อมูลของวิธีการ BLOB ใช้เวลาน้อยสุดเนื่องจาก $t_d = 0$ ส่วนวิธีการ Base128 ใช้เวลาในการดึงข้อมูลกลับได้ใกล้เคียงกับเวลาของ Base64 และบางชนิดไฟล์ภาพใช้เวลาน้อยกว่า เช่น ชนิด .pdf เป็นต้น

5. สรุปผลและเสนอแนะ

งานวิจัยนี้ได้ทำการแก้ปัญหาการลดขนาดของข้อมูลไบนารีหรือไฟล์ข้อมูลเมื่อจัดเก็บข้อมูลในภาพแบบสตริง โดยใช้วิธีสร้างคลาส Base128 ซึ่งได้ผลลัพธ์จากการลดขนาดของข้อมูลอยู่ที่ 30% เมื่อเทียบกับ Base64 ดีขึ้นจากเดิมเมื่อคิดในด้านเวลาบันทึกข้อมูล ลดความเสี่ยงข้อมูลสูญหายระหว่างการบันทึกข้อมูล เนื่องจากเก็บข้อมูลครั้งเดียวในภาพแบบสตริงแตกต่างจากวิธีเดิมที่อ่านค่าการเก็บข้อมูลทีละไบต์ ซึ่งระหว่างการอ่านค่าการเก็บทีละไบต์อาจเกิดการสูญหาย (loss) ของข้อมูลระหว่างการจัดเก็บได้ทำให้ข้อมูลที่เก็บลงฐานข้อมูลเกิดการเสียหายได้

งานวิจัยที่จะพัฒนาต่อไปคือการทดสอบประสิทธิภาพการค้นหาข้อมูลในระบบฐานข้อมูลที่มีการแปลงเป็นข้อมูลสตริงด้วยรหัส BASE64 BASE128 ซึ่งรวมทั้งวิธีการเข้ารหัสข้อมูลหลังทำการแปลงและเข้ารหัสแล้ว [5],[7] ก่อนการจัดเก็บข้อมูล

เอกสารอ้างอิง

[1] Mackenzie, Charles E, "ASCII," [Online] 1980. Available from : http://en.wikipedia.org/wiki/ASCII [2013, December 10].

[2] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings," [Online] 2006. Availablefrom : http://www.ietf.org/rfc/rfc4648.txt[2013,December 15].

[3] Dev , "base64 Image ," [Online] 2006. Available from : http://base64image.org/ [2013,January 15].

[4] Changeset 5814, "Base64 utility methods ," section 6.2, [Online] 2014. Available from : http://www.w3.org/html/wg/drafts/html/master/webappapis.html#atob [2013,December 15].

[5] Congfu Xu, Yafang Chen, Kevin Chiew, " An Approach to Image Spam Filtering Based on Base64 Encoding and N-Gram Feature Extraction, " International Conference on Tools with Artificial Intelligence. pp. 171-176, 2010.

[6] Zhenxing Liu , Lu Liu , Richard Hill1, Yongzhao Zhan, "Base62x: An Alternative Approach to Base64 for non-Alphanumeric Characters," Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD).pp. 2667-2670, 2011.

[7] Gurpreet Singh, Supriya, " Modified Vigenere Encryption Algorithm and its Hybrid Implementation with Base64 and AES ," International Conference on Advanced Computing, Networking and Security.pp. 232-237, 2013.

ประวัติผู้เขียน

ชื่อ-นามสกุล	จรรยาภรณ์ ประสมสัตย์
ประวัติการศึกษา	พ.ศ. 2548 วิศวกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยธุรกิจบัณฑิตย์
ตำแหน่งและสถานที่ทำงานปัจจุบัน	บริษัท Stone Apple Consulting