การศึกษาและประยุกต์ใช้การตรวจจับความน่าจะเป็นสูงสุด สำหรับการกู้ข้อมูลในระบบจัดเก็บข้อมูลแบบกระจาย

THE STUDY AND IMPLEMENTATION OF MAXIMUM LIKELIHOOD DECODER FOR DATA RECOVERY ON DISTRIBUTED DATA STORAGE

ภาคภูมิ แสงประสิทธิโชค

PAKPOOM SANGPRASITTICHOK

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2562

KMITL-2019-EN-M-010-135

การศึกษาและประยุกต์ใช้การตรวจจับความน่าจะเป็นสูงสุด สำหรับการกู้ข้อมูลในระบบจัดเก็บข้อมูลแบบกระจาย

THE STUDY AND IMPLEMENTATION OF MAXIMUM LIKELIHOOD DECODER FOR DATA RECOVERY ON DISTRIBUTED DATA STORAGE

ภาคภูมิ แสงประสิทธิโชค

PAKPOOM SANGPRASITTICHOK

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2562

KMITL-2019-EN-M-010-135

THE STUDY AND IMPLEMENTATION OF MAXIMUM LIKELIHOOD DECODER FOR DATA RECOVERY ON DISTRIBUTED DATA STORAGE

PAKPOOM SANGPRASITTICHOK

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN TELECOMMUNICATIONS ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2019

KMITL-2019-EN-M-010-135

หัวข้อวิทยานิพนธ์	การศึกษาและประยุกต์ใช้การตรวจจับความน่าจะเป็นสูงสุดสำหรับการกู้ข้อมูลในระบบจัดเก็บข้อมูลแบบกระจาย
นักศึกษา	นายภาคภูมิ แสงประสิทธิโชค
รหัสประจำตัว	58601350
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
พ.ศ.	2562
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.ณัฐกานต์ พุทธรักษ์

บทคัดย่อ

ในปัจจุบันความต้องการและปริมาณการใช้ข้อมูลเพิ่มมากขึ้นอย่างรวดเร็ว จึงจำเป็นต้องมีการสำรองเก็บข้อมูลไว้เพื่อป้องกันข้อมูลสูญหาย แต่เมื่อปริมาณข้อมูลเพิ่มมากขึ้นหน่วยความจำสำหรับเก็บข้อมูลชิ้นเดียวจึงไม่เพียงพอ จึงเกิดหน่วยเก็บข้อมูลแบบกระจาย (Distributed storage) ซึ่งเป็นการนำหน่วยความจำหลายๆ ตัวมาช่วยกันเก็บข้อมูล แต่เมื่อหน่วยความจำใดๆ ในหน่วยเก็บข้อมูลแบบกระจายเสียหายไป จะส่งผลให้ข้อมูลนั้นๆ สูญหายไปด้วย

วิทยานิพนธ์ฉบับนี้ได้ศึกษาระบบเก็บข้อมูลแบบกลุ่มเมฆซึ่งเป็นระบบเก็บข้อมูลแบบกระจายชนิดหนึ่งที่มีลักษณะแยกเก็บข้อมูลไปในหน่วยความจำต่างๆ ภายในกลุ่มเมฆ ซึ่งให้บริการอยู่บนเครือข่ายอินเตอร์เน็ต เพื่อให้ข้อมูลที่ถูกเก็บนั้นมีความน่าเชื่อถือ และพัฒนาระบบให้ดียิ่งขึ้น รหัสอีเรเซอร์จึงถูกคิดค้นขึ้นมาและนำมาประยุกต์ใช้กับระบบเก็บข้อมูลแบบกลุ่มเมฆ ซึ่งเป็นการเก็บข้อมูลโดยใช้พื้นที่บนโลกออนไลน์ โดยข้อมูลของผู้ใช้จะถูกแยกส่งไปเก็บตามหน่วยเก็บข้อมูลต่างๆ กระจายบนอินเตอร์เน็ต ทั้งนี้ในงานวิจัยนี้ได้ทำการศึกษาและนำรหัสอีเรเซอร์ คือ รหัส CGR (Complete Graph of Ring) ซึ่งมีคุณสมบัติเป็นรหัสระยะห่างสูงสุด (Maximum Distance Separable code : MDS code) ชนิดหนึ่งมาทดสอบกับระบบเก็บข้อมูลแบบกลุ่มเมฆที่อ้างอิงจากระบบเก็บข้อมูลแบบกระจาย (Distributed storage system) และนำเสนอการประยุกต์ใช้วิธีการถอดรหัสด้วยวิธีตรวจจับความน่าจะเป็นสูงสุด (Maximum Likelihood Decoder : MLD) ที่ได้รับการพัฒนาและประยุกต์ใช้บนเครื่องรับ เพื่อแก้ไขบิตข้อมูลที่ผิดพลาดจากการส่งผ่านช่องสัญญาณ โดยในงานวิจัยนี้ได้ใช้ข้อมูล 3 ชนิดที่แตกต่างกัน คือข้อมูลรูปภาพแบบไบนารี่ ข้อมูลรูปภาพแบบนอน-ไบนารี่ และข้อมูลเอกสาร ซึ่งผลการทดลองแสดงให้เห็นว่าเครื่องรับที่ใช้งานอัลกอริทึม MLD นั้นมีสมรรถนะที่ดีกว่าจากการที่มีค่าอัตราการผิดพลาดบิตข้อมูล (Bit Error Rate : BER) ที่น้อยลงนั่นเอง

Thesis	THE STUDY AND IMPLEMENTATION OF MAXIMUM LIKELIHOOD DECODER FOR DATA RECOVERY ON DISTRIBUTED DATA STORAGE
Student	Mr. Pakpoom Sangprasittichok
Student ID.	58601350
Degree	Master of Engineering
Program	Telecommunications Engineering
Year	2019
Thesis Advisor	Asst. Prof. Dr. Nattakan Puttarak

ABSTRACT

Nowadays, huge and a large amount of data is transferred and stored through network. It is backed up in order to protect data loss. However, the amount of data is rapidly increased, only one disks might not enough to handle the growing data. So, the distributed storage is proposed to combine multiple disks to be a group of disks to store such data. Nevertheless, when some of disks is broken, the data inside that disk will be lost.

This thesis studies about cloud storage system, which is a distributed storage system that stores data in many disks on the Internet. In order to achieve more reliable data and improve the storage system, erasure code is invented and applied to cloud storage system where in the user's data will be stored online through the network. This research studies and applies a complete graph of rings (CGR) code, which is a maximum distance separable (MDS) code, and implements in cloud storage system. Moreover, the maximum likelihood decoder (MLD) is also implemented on the receiver to detect and correct errors due to a noisy channel. In this research, 3 types of data, which are the binary images, non-binary images and files, are randomly stored on the distributed cloud storage. The experimental result shows that the receiver operated with the MLD outperforms by giving the lower BER.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ผศ.ดร.ณัฐกานต์ พุทธรักษ์ ที่ให้ความช่วยเหลือ ให้คำปรึกษา แนะนำและตรวจสอบแก้ไขข้อบกพร่องตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า ซึ่งทำให้สามารถทำวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

ขอกราบขอบพระคุณ คณาจารย์ประจำภาควิชาวิศวกรรมโทรคมนาคมทุกท่านที่ได้อบรมสั่งสอนและมอบความรู้อันเป็นประโยชน์อย่างยิ่งกับข้าพเจ้า

ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และพี่ของข้าพเจ้าที่ให้การสนับสนุนและให้กำลังใจมาตลอด

หากวิทยานิพนธ์ฉบับนี้มีข้อบกพร่องประการใด ข้าพเจ้ายินดีรับข้อเสนอแนะ และขออภัยมา ณ ที่นี้ด้วย

ภาคภูมิ แสงประสิทธิโชค

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

บทแรกของงานวิจัยนี้จะกล่าวในส่วนของบทนำ ซึ่งประกอบด้วย ที่มาและความสำคัญ ความมุ่งหมายและวัตถุประสงค์ ขอบเขตของงานวิจัย และระเบียบงานวิจัย

1.1 ที่มาและความสำคัญ

ในช่วงอดีตที่ผ่านมา มีความต้องการและปริมาณการใช้งานข้อมูลต่างๆ ที่เพิ่มมากขึ้นอย่างรวดเร็ว เพื่อป้องกันข้อมูลสูญหายจากการรับส่งหรือระหว่างการเก็บข้อมูลนั้น ทำให้ผู้ใช้งานจำเป็นต้องมีการสำรองข้อมูลในหน่วยเก็บข้อมูลสำรอง แต่เมื่อเวลาผ่านไปเมื่อข้อมูลเริ่มมีปริมาณมากขึ้นหน่วยความจำสำหรับเก็บข้อมูลชิ้นเดียวจึงไม่เพียงพอ จึงได้มีการนำหน่วยความจำสำหรับเก็บข้อมูลหลายๆ ตัวมาช่วยกันเก็บข้อมูลซึ่งเรียกว่าหน่วยเก็บข้อมูลแบบกระจาย (Distributed storage) เช่น ระบบ RAID (Redundant Array of Independent Disks) ในรูปแบบต่างๆ โดย RAID0 คือการนำหน่วยความจำหลายตัวมาช่วยกันเก็บข้อมูล แต่จะไม่มีการสำรองข้อมูล RAID1 มีการเพิ่มหน่วยความจำขึ้นมาอีกชุดหนึ่งเพื่อช่วยสำรองข้อมูล RAID5 ใช้การแยกเก็บข้อมูลกระจายในหน่วยความจำหลายๆ ตัวและมีการสำรองข้อมูลกระจายเก็บในหน่วยความจำทุกตัว ซึ่งทำให้หน่วยความจำสามารถเสียหายได้สูงสุดหนึ่งตัวโดยที่ข้อมูลไม่สูญหายและสามารถกู้คืนมาได้ และ RAID6 มีการทำงานเหมือน RAID5 แต่มีการเพิ่มสำรองข้อมูลไฟล์ในการกระจายในหน่วยความจำมากขึ้น ซึ่งทำให้มีความสามารถป้องกันหน่วยความจำเสียหายพร้อมกันได้ถึงสองตัว

จากที่กล่าวมาข้างต้นระบบ RAID0 และ RAID1 นั้นมีข้อดีข้อเสียที่แตกต่างกันกล่าวคือถ้าต้องการสำรองข้อมูลในระบบ RAID1 จะทำให้สิ้นเปลืองพื้นที่สำหรับสำรองข้อมูลนั้นๆ อีกหนึ่งชุด หรือถ้าต้องการพื้นที่เก็บข้อมูลที่เพิ่มขึ้นเหมือนใน RAID0 ข้อมูลจะสูญหายไปเลยหากหน่วยความจำนั้นเสียหาย กระบวนการเก็บข้อมูลโดยใช้รหัสอีเรเซอร์ (Erasure code) จึงถูกนำมาใช้งานกับระบบ RAID ซึ่งทำให้เกิดระบบ RAID5 และ RAID6 ซึ่งมีลักษณะการแบ่งข้อมูลออกเป็นหลายๆ ชุดกระจายเก็บในหน่วยความจำหลายๆ ตัว โดยเพิ่มข้อมูลตรวจสอบ 1 ชุด สำหรับระบบ RAID5 และข้อมูลตรวจสอบ 2 ชุด สำหรับระบบ RAID6 ซึ่งทำให้สามารถเพิ่มพื้นที่เก็บข้อมูลให้มากยิ่งขึ้น อีกทั้งยังมีการสำรองข้อมูลเพียงเล็กน้อยในการกู้ข้อมูลกลับมาได้ในกรณีที่หน่วยความจำเสียหายอีกด้วย [1]

หน่วยเก็บข้อมูลแบบกระจายอาจจะคุ้มค่าเมื่อถูกนำมาใช้งานในอุตสาหกรรมขนาดใหญ่ แต่ในอุตสาหกรรมขนาดเล็กซึ่งไม่ต้องการพื้นที่เก็บข้อมูลมากนัก แต่ยังต้องการความปลอดภัยของข้อมูล หน่วยเก็บข้อมูลแบบกระจายอาจจะไม่เหมาะมากนัก ดังนั้นต่อมาจึงมีผู้ให้บริการหน่วยเก็บข้อมูลแบบกลุ่มเมฆ (Cloud storage system) สำหรับผู้ใช้งานอินเตอร์เน็ตหรืออุตสาหกรรมขนาดเล็ก ซึ่งรหัสอีเรเชอร์ก็เป็นรหัสหนึ่งที่ถูกนำมาใช้ป้องกันข้อมูลภายในหน่วยเก็บข้อมูลแบบกลุ่มเมฆเสียหาย โดยรหัสอีเรเชอร์จะทำการเข้ารหัสข้อมูลโดยเพิ่มข้อมูลตรวจสอบเข้าไป และแยกเก็บไปในหน่วยความจำต่างๆ หลายๆ ตัวภายในหน่วยเก็บข้อมูลแบบกลุ่มเมฆนั้นๆ

อย่างไรก็ตาม การใช้งานรหัสอีเรเชอร์ทำให้ระบบหรือฮาร์ดดิสก์แต่ละตัวต้องสูญเสียพื้นที่เพื่อนำไปเก็บข้อมูลที่เพิ่มขึ้นมา (Redundancy) ซึ่งข้อมูลเหล่านี้นำมาสร้างรหัสอีเรเชอร์เพื่อปกป้องข้อมูลทั้งหมดในกรณีที่ฮาร์ดดิสก์เกิดการเสียหาย และข้อมูลเมื่อมาถึงผู้รับก็จะถูกลดทอนด้วยสัญญาณรบกวนจากช่องสัญญาณ ทำให้สัญญาณข้อมูลที่ได้รับมีความผิดเพี้ยนและเกิดความผิดพลาดในฝั่งรับเช่นเดิม ดังนั้นในงานวิจัยนี้จึงสนใจและศึกษาเกี่ยวกับอัลกอริทึมในการช่วยเพิ่มสมรรถนะในการกู้ข้อมูลที่เก็บไว้ในระบบจัดเก็บข้อมูลแบบกลุ่มเมฆที่ใช้รหัสอีเรเชอร์มาช่วยในการป้องกันข้อมูลโดยใช้กระบวนการตรวจจับความน่าจะเป็นสูงสุด (Maximum likelihood detector) ในฝั่งรับ ซึ่งจะไม่มีการเพิ่มข้อมูลสำหรับตรวจสอบใดๆ ทั้งสิ้นเพื่อให้การจัดการพื้นที่ในอุปกรณ์จัดเก็บข้อมูลเป็นไปอย่างคุ้มค่าและมีประสิทธิภาพที่สุด

1.2 ความมุ่งหมายและวัตถุประสงค์

1. ศึกษากระบวนการทำงานของระบบเก็บข้อมูลแบบกระจาย และกระบวนการป้องกันข้อมูลสูญหายภายในระบบเก็บข้อมูลแบบกระจายนั้นโดยใช้รหัสอีเรเชอร์ และจำลองสถานการณ์ให้ระบบเก็บข้อมูลภายในนั้นเสียหายในจำนวนที่รหัสอีเรเชอร์สามารถกู้ข้อมูลคืนได้ เพื่อให้สามารถวิเคราะห์สรุปผลและนำไปสู่การใช้งานในสภาวะจริงได้

2. ศึกษาการกระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุด (Maximum likelihood decoder) และลักษณะของข้อมูลที่จะนำมาทดลอง เช่น ข้อมูลภาพแบบไบนารี่ (Binary image) ข้อมูลภาพแบบนอน-ไบนารี่ (Non-binary image) และข้อมูลแบบเอกสาร (Text file) ซึ่งนำไปสู่การใช้งานอัลกอริทึมในสภาวะที่มีประสิทธิภาพสูงที่สุด

3. วิเคราะห์และสังเคราะห์ข้อมูลหลายๆ รูปแบบ เช่น ข้อมูลภาพแบบไบนารี่ ข้อมูลภาพแบบนอน-ไบนารี่ และข้อมูลแบบเอกสาร กับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด เพื่อชี้ให้เห็นถึงข้อดีและข้อเสีย เพื่อเป็นประโยชน์และสามารถนำไปประยุกต์ใช้ได้ในสภาพแวดล้อมจริง

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้มุ่งศึกษาค้นคว้าถึงปัจจัยของการเก็บรักษาข้อมูลในระบบเก็บข้อมูลแบบกระจายที่นำไปสู่การป้องกันและกู้ข้อมูลคืนในกรณีที่ข้อมูลสูญหายไปทั้งในระหว่างเก็บในระบบเก็บข้อมูลแบบกระจาย หรือระหว่างรับ/ส่งข้อมูลไปยังผู้ใช้งาน โดยวิเคราะห์อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดจากการประยุกต์ใช้กับข้อมูล 3 ประเภท คือ ข้อมูลภาพไบนารี่ ข้อมูลภาพนอน-ไบนารี่ และข้อมูลแบบเอกสาร

1.4 ระเบียบงานวิจัย

วิทยานิพนธ์เล่มนี้ประกอบด้วยเนื้อหาหลักซึ่งแบ่งได้ทั้งหมด 5 บท ประกอบด้วย บทแรกซึ่งเป็นบทนำจะกล่าวถึงที่มาและปัญหาซึ่งนำไปสู่การศึกษาและทดลองของงานวิจัยชิ้นนี้ บทที่สองจะนำเสนอข้อมูลพื้นฐานและทฤษฎีที่เกี่ยวข้องกับงานวิจัยและงานวิจัยต่างๆ ที่เกี่ยวข้องกับงานวิจัยนี้ บทที่สามจะกล่าวถึงการออกแบบกระบวนการและอัลกอริทึมที่ใช้ในการทดลอง บทที่สี่แสดงผลการทดลองของกระบวนการทดลองทั้งหมดรวมทั้งการวิเคราะห์ผลการทดลอง และบทสุดท้ายจะสรุปผลการทดลองของงานวิจัยในวิทยานิพนธ์นี้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง โดยในส่วนแรกจะกล่าวถึงทฤษฎีและข้อมูลพื้นฐานที่เกี่ยวข้องกับงานวิจัย ซึ่งประกอบด้วยระบบเก็บข้อมูลแบบกระจาย ระบบเก็บข้อมูลแบบกลุ่มเมฆ รหัสอีเรเซอร์ และอัลกอริทึมการถอดรหัส และในส่วนต่อมาจะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับการแก้ไขข้อมูลผิดพลาดและงานวิจัยที่มีการประยุกต์ใช้กระบวนการตรวจจับความน่าจะเป็นสูงสุด

2.1 ระบบเก็บข้อมูลแบบกระจาย

ระบบเก็บข้อมูลแบบกระจาย (Distributed storage system) เป็นระบบเก็บข้อมูลขนาดใหญ่ชนิดหนึ่ง โดยข้อมูลของผู้ใช้งานจะถูกกระจายเก็บไปในหน่วยความจำหลายๆ ตัว กรณีที่หน่วยความจำภายในระบบเก็บข้อมูลนั้นเสียหาย ข้อมูลภายในนั้นจะสูญหายไปด้วย จึงต้องมีวิธีป้องกันข้อมูลภายในระบบเก็บข้อมูลมีเสถียรภาพและความปลอดภัย (ตัวอย่างเช่น รหัสอีเรเซอร์)

ระบบ RAID (Redundant Arrays of Independent Disks) เป็นระบบเก็บข้อมูลแบบกระจายชนิดหนึ่งซึ่งมีความสามารถในการป้องกันข้อมูลสูญหายในกรณีที่หน่วยความจำเสียหายได้ โดยระบบ RAID นี้มีหลายรูปแบบซึ่งขึ้นอยู่กับการตั้งค่าและวิธีการป้องการข้อมูลเช่น การสำเนาข้อมูลซ้ำ (Replication) และการใช้รหัสอีเรเซอร์ (Erasure code) โดยกระบวนการที่กล่าวมานี้สามารถกู้ข้อมูลคืนมาในกรณีที่หน่วยความจำเสียหายได้ การตั้งค่าระบบ RAID มีตัวอย่างดังต่อไปนี้ [1]

1. RAID0

เป็นการนำเอาหน่วยความจำมากกว่า 1 ตัวมาช่วยเก็บข้อมูล เพื่อให้หน่วยความจำสามารถเก็บข้อมูลทีมีขนาดใหญ่มากขึ้นได้ โดยระบบ RAID นี้จะไม่มีการสำรองข้อมูลไว้เลย ในกรณีที่หน่วยความจำเสียหายข้อมูลจะไม่สามารถกู้คืนกลับมาได้ แต่ได้มาซึ่งความเร็วในการเข้าถึงข้อมูลที่สูง

2. RAID1

ข้อมูลที่เหมือนกันจะถูกเก็บในหน่วยความจำทั้ง 2 ตัว หรือเรียกได้ว่าเป็นการทำสำเนาข้อมูล (Disk mirroring) โดยความเร็วในการเขียนข้อมูลจะต่ำเนื่องจากต้องเขียนข้อมูลลงในหน่วยความจำทั้ง 2 ตัวเหมือนๆ กัน อีกทั้งยังสิ้นเปลืองหน่วยความจำที่นำมาทำสำเนาข้อมูลอีกด้วย

3. RAID2

เป็นระบบ RAID ที่ใช้รหัสแฮมมิง (Hamming code) ในการกู้ข้อมูลในกรณีที่หน่วยความจำเสียหาย โดยจะมีอัตราส่วน 4:3 คือใช้หน่วยความจำสำหรับเก็บข้อมูล 4 ตัว กับหน่วยความจำสำหรับเก็บข้อมูลตรวจสอบจำนวน 3 ตัว ซึ่งน้อยกว่าการทำสำเนาข้อมูลของ RAID1 จำนวน 1 ตัว

4. RAID3

เป็นระบบ RAID ที่มีลักษณะคล้ายกับ RAID2 โดยเพิ่มหน่วยความจำขึ้นมาอีกหนึ่งตัวเพื่อเก็บข้อมูลตรวจสอบ RAID3 นี้เหมาะกับการส่งผ่านข้อมูลจำนวนไม่มากนัก เพราะในกรณีที่ข้อมูลจำนวนมากจะทำให้เกิดคอขวด (Bottleneck) ที่หน่วยความจำสำหรับเก็บข้อมูลตรวจสอบ

5. RAID4

มีลักษณะโดยรวมเหมือน RAID3 คือเพิ่มหน่วยความจำขึ้นมาอีกหนึ่งตัวสำหรับเก็บข้อมูลตรวจสอบ เพียงแต่ RAID4 ข้อมูลจะถูกพิจารณาในรูปแบบบล็อก (Block) แทน RAID3 ที่ข้อมูลถูกพิจารณาในรูปแบบบิต แต่มีข้อเสียเช่นเดียวกับ RAID3 คือกรณีที่มีข้อมูลจำนวนมาก จะทำให้เกิดคอขวดที่หน่วยความจำสำหรับเก็บข้อมูลตรวจสอบ

6. RAID5

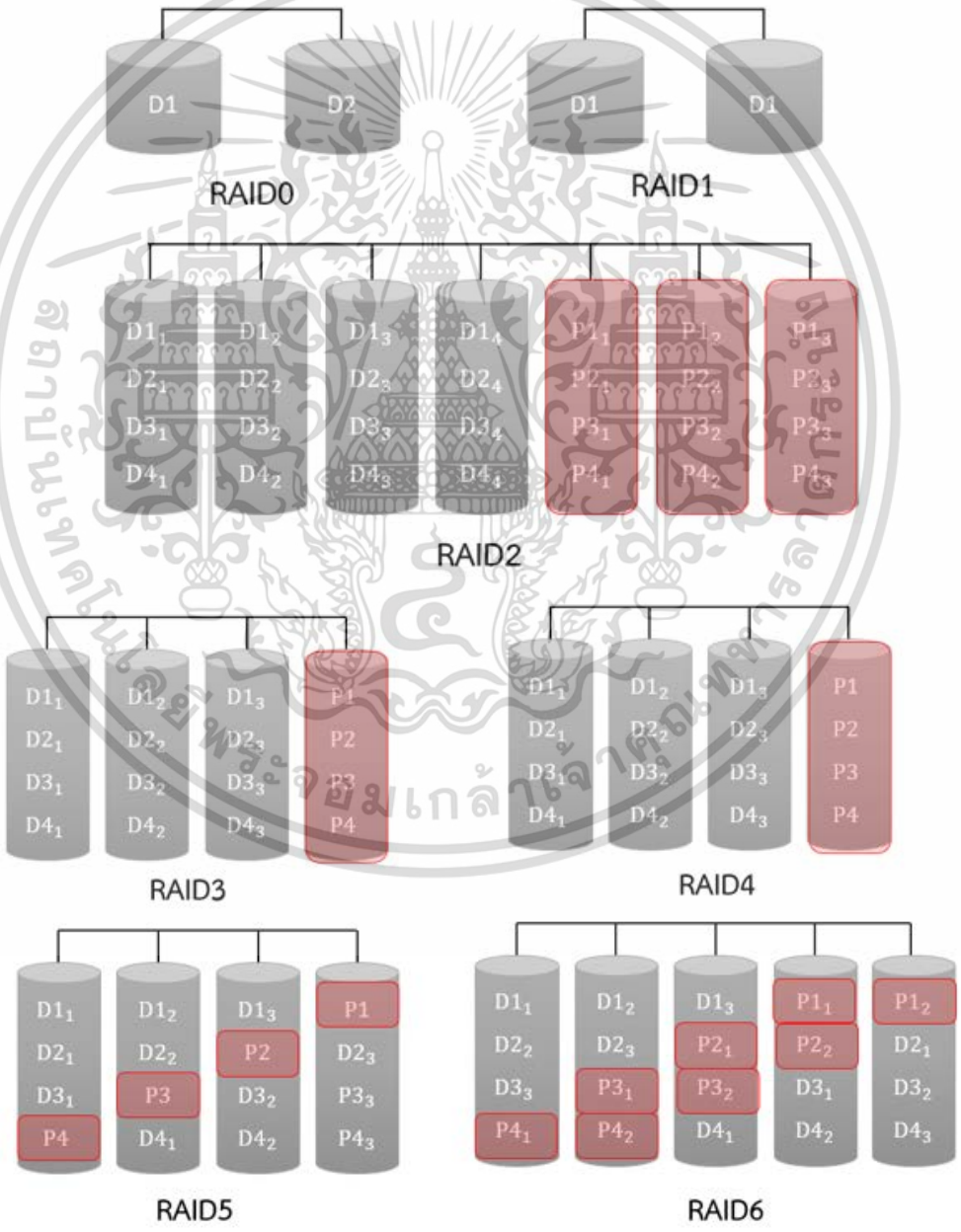
ข้อมูลจะถูกพิจารณาในรูปแบบบล็อกเช่นเดียวกับ RAID4 เพียงแต่ว่าข้อมูลตรวจสอบจะถูกกระจายเก็บในหน่วยความจำทุกๆ ตัวเพื่อแก้ปัญหาคอขวดจากระบบ RAID3 และ RAID4 ซึ่งทำให้ความเร็วในการเข้าถึงข้อมูลสูงกว่า โดย RAID5 นี้มีความสามารถกู้คืนข้อมูลได้ในกรณีที่หน่วยความจำภายในระบบเก็บข้อมูลเสียหายเพียงหนึ่งตัว จากการใช้งานข้อมูลจากหน่วยความจำที่เหลือมาพิจารณาข้อมูลในหน่วยความจำที่เสียหายไป

7. RAID6

มีลักษณะการทำงานเหมือน RAID5 ทุกประการ เพียงแต่เพิ่มข้อมูลตรวจสอบจำนวน 2 ชุด กระจายเก็บในหน่วยความจำทุกๆ ตัว ซึ่งทำให้สามารถกู้คืนข้อมูลได้กรณีที่หน่วยความจำภายในระบบเก็บข้อมูลเสียหายได้สูงสุดถึง 2 ตัว

จากที่กล่าวมาข้างต้นสามารถสรุปได้ดังนี้คือ RAID0 เป็นการนำหน่วยความจำมากกว่า 1 ตัวมาช่วยกันเก็บข้อมูล แต่จะไม่มีการสำรองข้อมูลในหน่วยความจำเลย กรณีที่หน่วยความจำเสียหายข้อมูลภายในจะสูญหายไปด้วย RAID1 เป็นการนำหน่วยความจำอีกหนึ่งชุดมาทำสำเนาข้อมูล (Disk

mirroring) แต่จะสิ้นเปลืองทรัพยากรในการนำหน่วยความจำมาสำรองข้อมูลนั้นๆ RAID2 RAID3 RAID4 เป็นการเพิ่มข้อมูลตรวจสอบเข้าไปในหน่วยความจำสำหรับเก็บข้อมูลตรวจสอบ โดยใช้กระบวนการและการเพิ่มหน่วยความจำสำหรับข้อมูลตรวจสอบที่แตกต่างกัน RAID5 เป็นการเพิ่มข้อมูลตรวจสอบ 1 ชุดกระจายเก็บไปในหน่วยความจำทุกตัว โดยช่วยลดปัญหาคอขวดที่หน่วยความจำสำหรับเก็บข้อมูลตรวจสอบได้ โดยมีความสามารถในการกู้ข้อมูลคืนได้กรณีที่หน่วยความจำเสียหายหนึ่งตัว สำหรับ RAID6 จะมีกระบวนการทำงานเหมือน RAID5 แต่เพิ่มข้อมูลตรวจสอบ 2 ชุดกระจายเก็บไปในหน่วยความจำทุกตัว ซึ่งสามารถกู้ข้อมูลคืนได้ในกรณีที่หน่วยความจำเสียหายสูงสุด 2 ตัว ซึ่งสรุปได้ดังรูปที่ 2.1



รูปที่ 2.1 รูปแบบการเก็บข้อมูลของ RAID0, RAID1, RAID2, RAID3, RAID4, RAID5, RAID6

2.2 ระบบเก็บข้อมูลแบบกลุ่มเมฆ

การประมวลผลบนกลุ่มเมฆ (Cloud computing) เป็นเทคโนโลยีซึ่งใช้ทรัพยากรเช่น ฮาร์ดแวร์และซอฟต์แวร์ของผู้ให้บริการผ่านที่ต่างๆ บนอินเตอร์เน็ต ซึ่งจะช่วยประหยัดค่าใช้จ่าย เนื่องจากทรัพยากรต่างๆ นั้นถูกจัดสรรให้ผู้ใช้บริการอย่างเหมาะสม มีความน่าเชื่อถือเนื่องจากในระบบประมวลผลบนกลุ่มเมฆนั้นมีกระบวนการป้องกันข้อมูลในกรณีที่ข้อมูลสูญหาย และสามารถเข้าถึงได้ง่ายเนื่องจากระบบต่างๆ อยู่บนอินเตอร์เน็ต

2.2.1 รูปแบบการให้บริการของการประมวลผลบนกลุ่มเมฆ

ประกอบไปด้วย [2]

1. Infrastructure as a service (IaaS)

ระบบนี้ใช้หลักการ Virtual Machine เพื่อรองรับการใช้ซอฟต์แวร์และแอปพลิเคชันบนระบบบนกลุ่มเมฆ โดยมีการจัดการทรัพยากรและพื้นที่จัดเก็บข้อมูลตามความต้องการของผู้ใช้งาน ตัวอย่างเช่น Telstra, Microsoft Azure และ amazon web services

2. Platform as a service (PaaS)

ระบบที่ให้บริการด้านแพลตฟอร์มสำหรับพัฒนาซอฟต์แวร์หรือแอปพลิเคชันบนเซิร์ฟเวอร์ของผู้ให้บริการ ยกตัวอย่างเช่น Microsoft SQL Azure, Amazon RDS และ Wordpress

3. Software as a service (SaaS)

ระบบที่ให้บริการด้านซอฟต์แวร์และแอปพลิเคชันกับผู้ใช้บริการผ่านทางอินเตอร์เน็ต ยกตัวอย่างเช่น Gmail และ Microsoft office

4. Storage as a service (StaaS)

ระบบที่ให้บริการหน่วยความจำสำหรับเก็บข้อมูลให้กับผู้ใช้บริการผ่านทางอินเตอร์เน็ต ยกตัวอย่างเช่น Onedrive, Google drive และ Dropbox

2.2.2 รูปแบบการใช้งานของระบบประมวลผลบนกลุ่มเมฆ

ประกอบไปด้วย

1. Private Cloud

ระบบที่ทำงานอยู่บนกลุ่มเมฆแบบส่วนตัวซึ่งถูกพัฒนาสำหรับบริษัทหรือองค์กร เพื่อใช้งานภายในองค์กรเท่านั้น ซึ่งผู้ใช้บริการสามารถควบคุมและปรับปรุงระบบความปลอดภัยได้

2. Community Cloud

เป็นระบบ private cloud ชนิดหนึ่ง ซึ่งถูกพัฒนาสำหรับบริษัทหรือองค์กรต่างๆ ที่มีความต้องการหรือความสนใจร่วมกันสามารถเข้าใช้งานร่วมกันได้

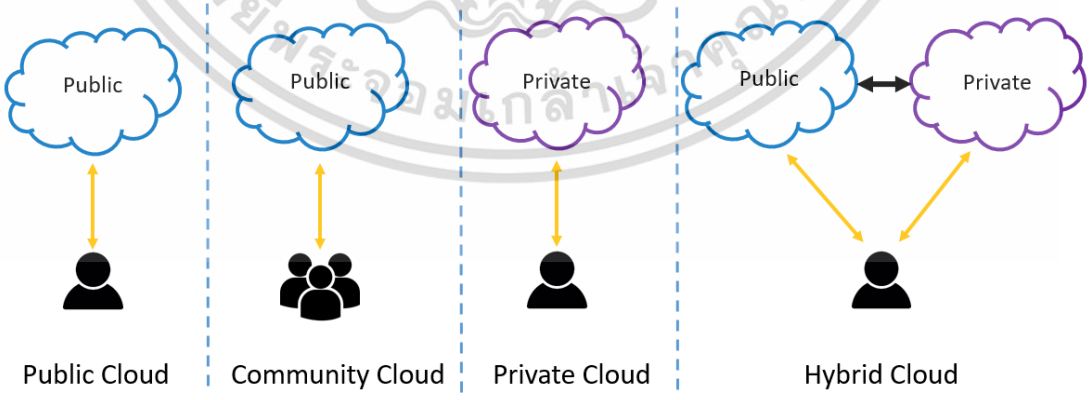
3. Public Cloud

ระบบที่ทำงานอยู่บนกลุ่มเมฆแบบสาธารณะซึ่งถูกพัฒนาให้ทุกคนสามารถใช้งานร่วมกันได้ผ่านทางอินเตอร์เน็ต โดยผู้ใช้บริการมีสิทธิการใช้งานที่จำกัดซึ่งถูกกำหนดโดยผู้ให้บริการ

4. Hybrid Cloud

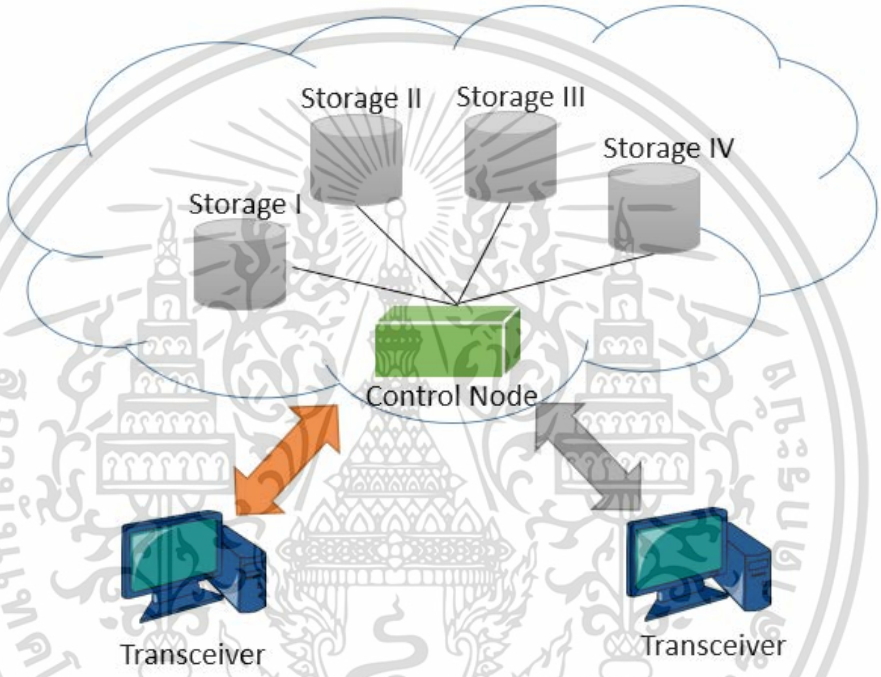
เป็นการผสมกันระหว่างระบบ private cloud และ public cloud ซึ่งระบบที่ทำงานบนกลุ่มเมฆนี้มีการเชื่อมต่อเพื่อให้สามารถย้ายข้อมูลหรือแอปพลิเคชันจากระบบกลุ่มเมฆหนึ่งไปยังอีกระบบกลุ่มเมฆหนึ่งได้

ซึ่งรูปแบบการใช้งานทั้ง 4 แบบดังที่กล่าวด้านบนนี้สามารถสรุปได้ดังรูปที่ 2.2



รูปที่ 2.2 รูปแบบการใช้งานของระบบประมวลผลบนกลุ่มเมฆ

ซึ่งในงานวิจัยนี้ทางผู้วิจัยจะสนใจระบบเก็บข้อมูลแบบกลุ่มเมฆ ซึ่งระบบเก็บข้อมูลแบบกลุ่มเมฆนี้เป็นระบบเก็บข้อมูลแบบกระจายชนิดหนึ่งซึ่งหน่วยเก็บข้อมูลต่างๆ นั้นกระจายไปหลายๆ แห่งบนเครือข่ายอินเตอร์เน็ต เมื่อผู้ใช้งานต้องการเก็บไฟล์ข้อมูลไปยังหน่วยเก็บข้อมูลแบบกลุ่มเมฆนี้ ข้อมูลจะถูกส่งไปยังโหนดควบคุมบนอินเตอร์เน็ตก่อน หลังจากนั้นโหนดควบคุมจะกระจายข้อมูลไปเก็บในหน่วยความจำอื่นๆ ต่อๆ ไป ในส่วนของผู้รับที่ต้องการดึงหรืออ่านไฟล์จากหน่วยเก็บข้อมูลแบบกลุ่มเมฆนี้ โหนดควบคุมจะทำการรวบรวมไฟล์ข้อมูลจากหน่วยความจำอื่นๆ มาเพื่อให้ได้ไฟล์ข้อมูลดั้งเดิมแล้วส่งต่อไปยังผู้รับ [3] ดังรูปที่ 2.3

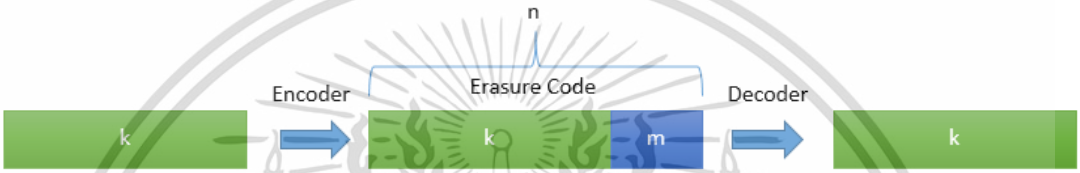


รูปที่ 2.3 แบบจำลองหน่วยเก็บข้อมูลแบบกลุ่มเมฆอย่างง่าย

เนื่องจากระบบเก็บข้อมูลแบบกลุ่มเมฆเป็นระบบเก็บข้อมูลที่มีความน่าเชื่อถือสูง การป้องกันข้อมูลสูญหายภายในระบบเก็บข้อมูลนี้จึงมีความสำคัญเป็นอย่างมาก วิธีการโดยทั่วไปคือการสำรองข้อมูลอีกหนึ่งชุดเผื่อในกรณีที่ข้อมูลชุดเดิมเสียหายหรือสูญหาย เช่น RAID1 แต่เนื่องจากวิธีนี้จะสิ้นเปลืองทรัพยากรเป็นอย่างมาก จึงมีการนำรหัสอีเรเซอร์มาเข้า/ถอดรหัสข้อมูลก่อน ซึ่งจะอธิบายรายละเอียดในหัวข้อถัดไป โดยที่รหัสอีเรเซอร์เป็นวิธีการที่นิยมในการป้องกันข้อมูลสูญหายภายในระบบเก็บข้อมูลบนกลุ่มเมฆในปัจจุบัน กล่าวคือเป็นการเพิ่มข้อมูลตรวจสอบเพื่อใช้กู้ข้อมูลคืนในกรณีที่ข้อมูลในหน่วยความจำแบบกลุ่มเมฆตัวใดๆ สูญหายไป

2.3 รหัสอีเรเซอร์

รหัสอีเรเซอร์เป็นรหัสบล็อก (Block code) ชนิดหนึ่งที่นำมาใช้กับช่องสัญญาณแบบลบเลือน (Erasure channel) โดยการนำข้อมูล (Data) จำนวน k ชุด และเพิ่มส่วนข้อมูลตรวจสอบ (Parity) จำนวน m ชุด ซึ่งจะได้ข้อมูลทั้งหมด n ชุด ดังรูปที่ 2.4 และนำข้อมูลจำนวน n ชุดนั้นกระจายไปเก็บในหน่วยความจำทั้งหมด n ตัวต่างๆ กัน (เรียกว่า การเข้ารหัส) โดยเมื่อมีหน่วยความจำเสียหายบางตัว ระบบจะสามารถกู้ข้อมูลได้จากการนำข้อมูลที่แยกเก็บในหน่วยความจำอื่นๆ มาถอดรหัสเพื่อกู้ข้อมูลนั้นๆ กลับมาได้ ข้อดีในการใช้รหัสอีเรเซอร์คือช่วยลดค่าใช้จ่ายในการลงทุนสร้างหรือเพิ่มหน่วยความจำเพื่อมาเก็บข้อมูลสำรองซ้ำอีก



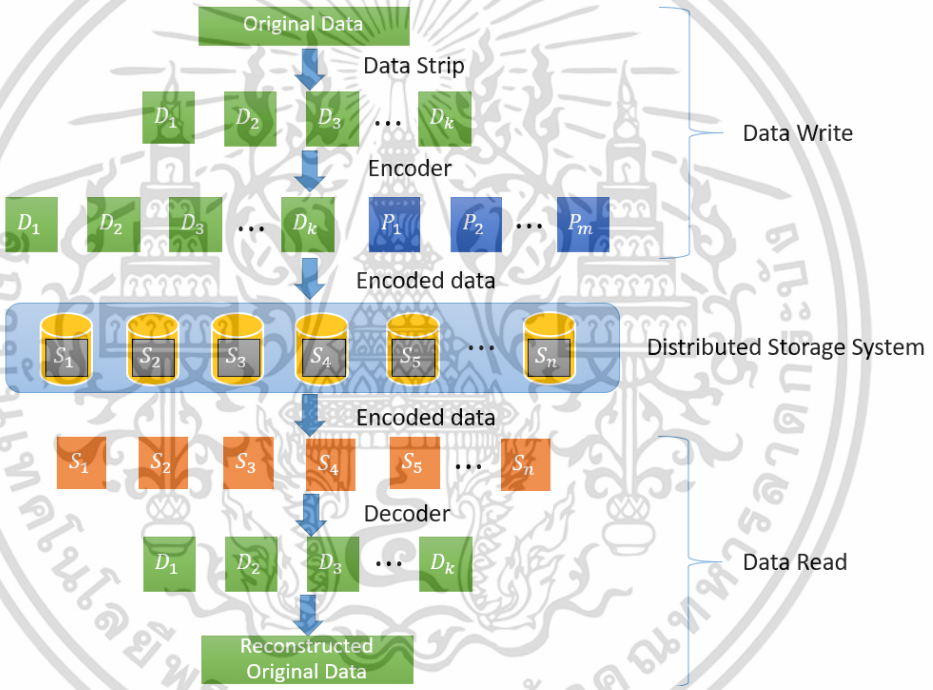
รูปที่ 2.4 โครงสร้างรหัสอีเรเซอร์

รหัสอีเรเซอร์ที่มีความสามารถกู้ข้อมูลที่เสียหายได้สูงสุดเท่ากับจำนวนข้อมูลตรวจสอบที่เพิ่มเข้าไปนั้นถือว่าเป็นรหัสที่มีคุณสมบัติ Singleton bound หรือที่เรียกกันว่ารหัสระยะห่างสูงสุด (Maximum distance separable code : MDS code) ซึ่งถือว่าเป็นรหัสที่เหมาะสมที่สุด (Optimal code) และเป็นรหัสอีเรเซอร์ประเภทที่นำมาใช้งานกับงานเก็บข้อมูลมากที่สุด ยกตัวอย่างเช่น รหัสรีด-โซโลมอน (Reed-Solomon) [4] รหัส EVENODD [5] หรือรหัส CGR (Complete Graph of Rings) [6]

รหัสอีเรเซอร์ใช้ในการเข้ารหัสก่อนเก็บลงหน่วยความจำแบบกระจายชนิด RAID5 คือ ระบบเก็บข้อมูลที่มีการเพิ่มข้อมูลตรวจสอบหนึ่งชุด โดยสามารถกู้ข้อมูลคืนได้กรณีที่หน่วยความจำเสียหายได้หนึ่งตัว และ RAID6 คือระบบเก็บข้อมูลที่เหมือนกับ RAID5 แต่มีการเพิ่มข้อมูลตรวจสอบ 2 ชุด ซึ่งสามารถกู้ข้อมูลคืนได้กรณีที่หน่วยความจำเสียหายได้พร้อมกัน 2 ตัว [7] รหัสอีเรเซอร์ที่นิยมใช้ในระบบเก็บข้อมูลคือรหัสรีด-โซโลมอน (Reed-Solomon code) ซึ่งเป็นรหัสที่เป็นที่รู้จักกันอย่างกว้างขวาง และยังเป็นรหัสระยะห่างสูงสุดที่มีความสามารถกู้ข้อมูลคืนได้เท่ากับจำนวนข้อมูลตรวจสอบที่เพิ่มเข้าไป รหัสรีด-โซโลมอนนี้ถูกใช้เข้ารหัสข้อมูลก่อนส่งไปเก็บในระบบเก็บข้อมูลชนิด RAID5 และ RAID6 [8]

รูปแบบการเก็บข้อมูลในระบบเก็บข้อมูลที่มีการใช้รหัสอีเรเซอร์นั้นมีกระบวนการดังรูปที่ 2.5 คือ

1. การเก็บข้อมูล ข้อมูลจากผู้ส่งจะถูกแบ่งเป็นจำนวน k ส่วน และถูกเข้ารหัสด้วยรหัสอีเรเซอร์โดยมีการเพิ่มข้อมูลตรวจสอบจำนวน m ส่วนเข้าไปตามรูปแบบของรหัสที่ใช้ ซึ่งจะได้ข้อมูลหลังการเข้ารหัสจำนวน n ส่วน แยกเก็บในระบบเก็บข้อมูลแบบกระจายหรือระบบเก็บข้อมูลแบบกลุ่มเมฆ
2. การอ่านข้อมูล ข้อมูลจะถูกรวบรวมจากระบบเก็บข้อมูลแบบกระจายหรือระบบเก็บข้อมูลแบบกลุ่มเมฆมาเพื่อถอดรหัส ในกรณีที่ข้อมูลนั้นอยู่ในเงื่อนไขที่สามารถถอดรหัสได้ จะได้ข้อมูลดั้งเดิมกลับมาและส่งกลับไปยังผู้รับ [9]



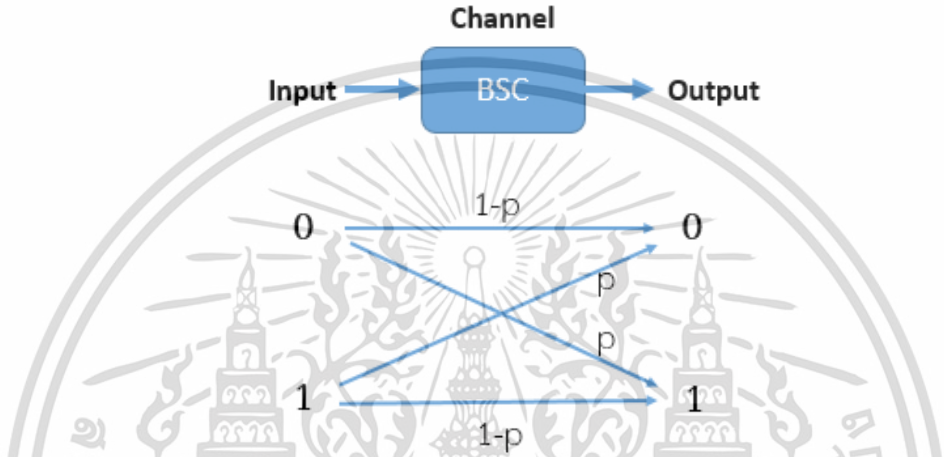
รูปที่ 2.5 รูปแบบการเก็บข้อมูลด้วยกระบวนการเข้า/ถอดรหัสอีเรเซอร์

2.4 อัลกอริทึมการถอดรหัส

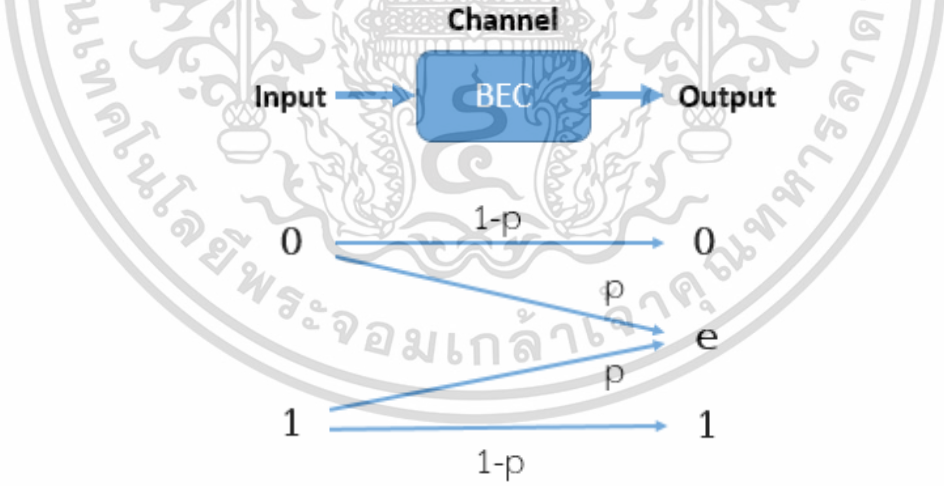
รูปแบบการรับส่งข้อมูลทางระบบสื่อสารประกอบไปด้วยข้อมูลผ่านการเข้ารหัสก่อนเข้าสู่ช่องสัญญาณของระบบสื่อสาร หลังจากข้อมูลผ่านช่องสัญญาณมาข้อมูลจะถูกถอดรหัสเพื่อให้ได้ข้อมูลดั้งเดิมกลับคืนมาดังรูปที่ 2.6

รูปที่ 2.6 รูปแบบการรับส่งข้อมูลผ่านช่องสัญญาณ

ช่องสัญญาณในงานวิจัยนี้จะประกอบด้วยช่องสัญญาณ 2 ชนิดคือ Binary Symmetric Channel (BSC) และ Binary Erasure Channel (BEC) ดังรูปที่ 2.7 และรูปที่ 2.8 ตามลำดับ



รูปที่ 2.7 รูปแบบช่องสัญญาณแบบ Binary Symmetric Channel



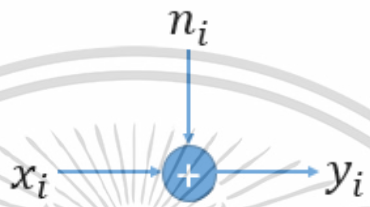
รูปที่ 2.8 รูปแบบช่องสัญญาณแบบ Binary Erasure Channel

โดยที่

1. Binary Symmetric Channel (BSC) เป็นช่องสัญญาณที่มีข้อมูลอินพุตและข้อมูลเอาท์พุตเป็นไบนารี่ โดยข้อมูลอินพุตเมื่อผ่านช่องสัญญาณจะมีความน่าจะเป็นที่บิตข้อมูลจะถูกเปลี่ยนมีค่า p ($input, output \in \{0,1\}$)

2. Binary Erasure Channel (BEC) เป็นช่องสัญญาณในการสื่อสารที่มีลักษณะคล้ายกับ BSC ซึ่งนิยมใช้ในแบบจำลองการสื่อสารแบบเข้ารหัส ซึ่งข้อมูลอินพุตสามารถถูกลบเลือน (e) ได้ด้วยความน่าจะเป็น p ($input \in \{0,1\}, output \in \{0, e, 1\}$) [10]

เมื่อกำหนดให้ข้อมูลถูกส่งในช่องสัญญาณแบบสัญญาณรบกวนเกาส์เซียนขาวบวก (Additive white Gaussian noise: AWGN) โดยมี x_i คือค่าข้อมูลแบบไบนารี่ที่ส่งผ่านช่องสัญญาณ และได้ข้อมูลหลังผ่านช่องสัญญาณคือ $y_i = x_i + n_i$ เมื่อ n_i คือสัญญาณรบกวนตามรูปที่ 2.9



รูปที่ 2.9 ข้อมูลในช่องสัญญาณแบบสัญญาณรบกวนเกาส์เซียนขาวบวก

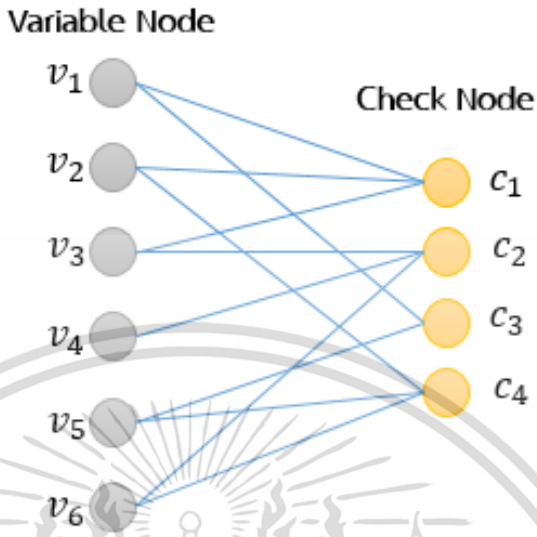
การเข้ารหัสข้อมูลก่อนส่งผ่านช่องสัญญาณของระบบสื่อสารนั้นช่วยป้องกันข้อมูลผิดพลาดที่เกิดระหว่างการรับส่งข้อมูล เมื่อข้อมูลอาจมีความผิดพลาดบางส่วนหลังผ่านช่องสัญญาณ การถอดรหัสข้อมูลนั้นๆ จะช่วยแก้ไขข้อมูลที่ผิดพลาดกลับคืนมาได้ สำหรับรูปแบบการถอดรหัสนั้นมีหลากหลายรูปแบบซึ่งทางผู้วิจัยจะขอยกตัวอย่างการถอดรหัสที่นิยมใช้ในปัจจุบันคือ

Message Passing Algorithm (MPA) เป็นกระบวนการถอดรหัสแบบวนซ้ำ (Iterative decoding) ที่นิยมใช้ในการถอดรหัสแอลดีพีซี ซึ่งเป็นรหัสบล็อกชนิดหนึ่ง [11] โดยรหัสชนิดนี้ใช้ลักษณะการเข้า/ถอดรหัสด้วยรูปแบบของกราฟสองส่วน (Bipartite graph หรือ Tanner graph) ซึ่งประกอบด้วยโหนดตัวแปร (Variable node) และโหนดตรวจสอบ (Check node) เชื่อมต่อเข้าหากัน เมื่อกำหนดให้ H คือเมตริกซ์ขนาด $m \times n$ จากกราฟสองส่วนจะได้โหนดตัวแปรคือ $V \in \{v_1, v_2, v_3 \dots, v_n\}$ และโหนดตรวจสอบคือ $C \in \{c_1, c_2, c_3 \dots, c_m\}$ เมื่อยกตัวอย่างเมตริกซ์ H ดังแสดงด้านล่าง

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

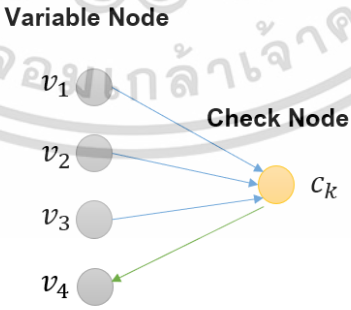
เมื่อกำหนดให้แถวแนวนอนของเมตริกซ์ H แทนด้วยโหนดตัวแปร และแถวแนวตั้งแทนด้วยโหนดตรวจสอบ จะสามารถเขียนกราฟเส้นเชื่อมในรูปแบบของกราฟสองส่วนได้จากการลากเส้นเชื่อมจากโหนดตัวแปรไปยังโหนดตรวจสอบในกรณีที่ค่าในเมตริกซ์ H ในตำแหน่งที่แถวและหลักนั้นๆ มีค่า

บิตเป็น 1 และไม่มีเส้นเชื่อมระหว่างสองโหนดนั้นๆ ในตำแหน่งที่แถวและหลักนั้นๆ มีค่าบิตเป็น 0 จากตัวอย่างเมตริกซ์ *H* จะสามารถวาดกราฟเส้นเชื่อมในรูปแบบของกราฟสองส่วนได้ดังรูปที่ 2.10

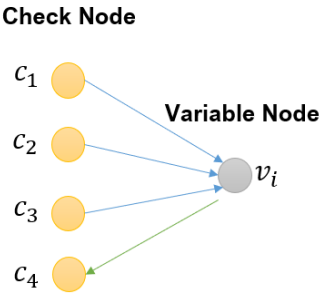


รูปที่ 2.10 กราฟสองส่วนตามเมตริกซ์ *H*

สำหรับกระบวนการทำงานของ MPA ในขั้นตอนแรกแต่ละโหนดตัวแปรจะได้รับค่าข้อมูลอินพุตมาจากช่องสัญญาณของระบบสื่อสาร หลังจากนั้นจะส่งข้อมูลต่อไปยังโหนดตรวจสอบ c_k ข้างเคียงเพื่อตรวจสอบข้อมูลดังรูปที่ 2.11 หลังจากนั้นโหนดตรวจสอบจะส่งข้อมูลกลับมายังโหนดตัวแปร v_i ดังรูปที่ 2.12 ซึ่งโหนดตัวแปรจะทำการส่งข้อมูลไปตรวจสอบกับโหนดตรวจสอบถัดไปเรื่อยๆ จนกระทั่งสิ้นสุดกระบวนการ [12] โดยกระบวนการถอดรหัสของ MPA แบ่งได้ 2 ชนิด คือ การถอดรหัสแบบหยาบ (hard decision) โดยใช้ Bit-Flip Algorithm และการถอดรหัสแบบละเอียด (soft decision) โดยใช้ Sum-Product Algorithm [13]



รูปที่ 2.11 กราฟเส้นเชื่อมย่อยแสดงข้อมูลที่ผ่านโหนดตรวจสอบ c_k มายังโหนดตัวแปร v_4



รูปที่ 2.12 กราฟเส้นเชื่อมย่อยแสดงข้อมูลที่ผ่านโหนดตัวแปร v_i มายังโหนดตรวจสอบ c_4

2.4.1 Bit-Flip Algorithm

Bit-Flip Algorithm เป็นอัลกอริทึมที่ใช้หลักการถอดรหัสแบบหยาบ (hard decision) เมื่อข้อมูลแบบไบนารี่ถูกส่งออกจากช่องสัญญาณมายังโหนดตัวแปร โหนดตัวแปรจะส่งข้อมูลไปยังโหนดตรวจสอบข้างเคียงเพื่อตรวจสอบข้อมูล ซึ่งโหนดตรวจสอบจะทำการตรวจสอบข้อมูลนั้นๆ โดยการนำข้อมูลทั้งหมดมาทำการ XOR (Exclusive-OR) กันดังสมการที่ 2.1

$$c_k = v_1 \oplus v_2 \oplus v_3 \oplus \ldots \oplus v_l \tag{2.1}$$

โดยที่

k คือ จำนวนลำดับของโหนดตรวจสอบ โดย $1 \leq k \leq m$ เมื่อ m คือจำนวนโหนดตรวจสอบ

$v_1, v_2, v_3 \ldots, v_l$ คือ ข้อมูลโหนดตัวแปรที่มีเส้นเชื่อมไปยังโหนดตรวจสอบ c_k

หากโหนดตรวจสอบพบว่าข้อมูลที่ได้จากโหนดตัวแปรนั้นมีข้อผิดพลาด โหนดตรวจสอบจะทำการตรวจสอบและปรับเปลี่ยนค่าบิตข้อมูลของโหนดตัวแปรโดยการนำข้อมูลของโหนดตรวจสอบทั้งหมดมา XOR กันดังสมการที่ 2.2

$$v_i = c_1 \oplus c_2 \oplus c_3 \oplus \ldots \oplus c_j \tag{2.2}$$

เมื่อ

i คือ จำนวนลำดับของโหนดตัวแปร โดย $1 \leq i \leq n$ เมื่อ n คือจำนวนโหนดตัวแปร

$c_1, c_2, c_3 \ldots, c_j$ คือ ข้อมูลโหนดตรวจสอบที่มีเส้นเชื่อมไปยังโหนดตัวแปร v_i

หลังจากโหนดตัวแปรปรับเปลี่ยนค่าบิตข้อมูลแล้ว ระบบจะส่งข้อมูลจากโหนดตัวแปรไปตรวจสอบกับโหนดตรวจสอบอีกครั้ง กระบวนการนี้จะทำงานไปเรื่อยๆ จนกระทั่งได้ค่าข้อมูลที่มีความผิดพลาดน้อยที่สุด Bit-Flip Algorithm เป็นอัลกอริทึมที่มีกระบวนการตรวจสอบของโหนดตัวแปรและโหนดตรวจสอบที่ง่าย ซึ่งทำให้มีความซับซ้อนในการคำนวณต่ำดังที่กล่าวมาข้างต้น แต่ประสิทธิภาพของการถอดรหัสนั้นยังไม่ดีเท่าที่ควร [13]

2.4.2 Sum-Product Algorithm

Sum-Product Algorithm หรือที่รู้จักกันคือวิธีการถอดรหัสด้วยความน่าจะเป็นสูงสุด (Maximum likelihood decoder) ซึ่งเป็นอัลกอริทึมที่ใช้หลักการถอดรหัสแบบละเอียด (soft decision) มีหลักการทำงานเหมือนกับการถอดรหัสแบบหยาบ (hard decision) เพียงแต่ข้อมูลจะพิจารณาว่าเป็นค่าบิต 0 หรือ 1 นั้นคำนวณมาจากโอกาสความน่าจะเป็นสูงสุด

เมื่อกำหนดให้ $P_i = P_r\,[v_i = 1|y]$ คือเงื่อนไขความน่าจะเป็นของโหนดตัวแปร v_i มีค่า 1 ที่ให้ค่าเอาท์พุต y ซึ่งจะได้ $P_r[v_i = 0|y] = 1 - P_i$ และกำหนดให้ q_{ij} คือข้อมูลที่ถูกส่งจากโหนดตัวแปร v_i ไปยังโหนดตรวจสอบ c_j โดยค่า $q_{ij}(0)$ และ $q_{ij}(1)$ คือค่าความน่าเชื่อถือที่ว่าบิต y_i จะมีค่าเป็นบิต 0 หรือบิต 1 ตามลำดับ โดยกำหนด $q_{ij}(1) = P_i$ และ $q_{ij}(0) = 1 - P_i$ ซึ่ง $q_{ij}(0) + q_{ij}(1) = 1$ และในทางกลับกันเมื่อกำหนด r_{ji} คือข้อมูลที่ส่งจากโหนดตรวจสอบ c_j มายังโหนดตัวแปร v_i โดยค่า โดยค่า $r_{ji}(0)$ และ $r_{ji}(1)$ คือค่าความน่าเชื่อถือที่ว่าบิต y_i จะมีค่าเป็นบิต 0 หรือบิต 1 ตามลำดับ ซึ่ง $r_{ji}(0) + r_{ji}(1) = 1$ ความน่าจะเป็นของข้อมูลที่โหนดตัวแปรส่งมายังโหนดตรวจสอบสามารถหาได้จากสมการที่ 2.3 และ 2.4

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in V_j \neq i} (1 - 2q_{i'j}(1)) \tag{2.3}$$

$$r_{ji}(1) = 1 - r_{ji}(0) \tag{2.4}$$

เมื่อ V_j คือเซ็ตของโหนดตัวแปรทั้งหมดที่มีเส้นเชื่อมกับโหนดตรวจสอบ c_j

ความน่าจะเป็นของข้อมูลที่โหนดตรวจสอบส่งมายังโหนดตัวแปรสามารถหาได้จากสมการ

$$q_{ij}(0) = k_{ij}(1 - P_i) \prod_{j' \in C_i \neq j} r_{j'i}(0) \tag{2.5}$$

$$q_{ij}(1) = k_{ij} P_i \prod_{j' \in C_i \neq j} r_{j'i}(1) \tag{2.6}$$

เมื่อ C_i คือเซ็ตของโหนดตรวจสอบทั้งหมดที่มีเส้นเชื่อมกับโหนดตัวแปร v_i

และค่า k_{ij} คือค่าคงที่ที่ทำให้สมการ $q_{ij}(0) + q_{ij}(1) = 1$

ซึ่งจะสามารถคำนวณแต่ละโหนดตัวแปรได้จากสมการที่ 2.7 และ 2.8

$$Q_i(0) = k_i(1 - P_i) \prod_{j' \in C_i} r_{ji}(0) \tag{2.7}$$

$$Q_i(1) = k_i P_i \prod_{j' \in C_i} r_{ji}(1) \tag{2.8}$$

เมื่อ $Q_i(0)$ และ $Q_i(1)$ คือความน่าจะเป็นที่ข้อมูลที่โหนดตัวแปร v_i นั้นจะมีค่าเป็น 0 และ 1 ตามลำดับ ซึ่งสามารถพิจารณาค่าข้อมูลที่โหนดตัวแปร v_i ได้จากสมการที่ 2.9

$$v_i = \begin{cases} 1, when\ Q_i(1) > Q_i(0) \\ 0, when\ Q_i(0) > Q_i(1) \end{cases} \tag{2.9}$$

ซึ่งอัลกอริทึมนี้จะทำงานจนกระทั่งพบข้อมูลบิตผิดพลาดที่น้อยที่สุด [14]

2.5 งานวิจัยที่เกี่ยวข้อง

ระบบเก็บข้อมูลแบบกระจาย (Distributed storage system) เป็นระบบเก็บข้อมูลที่ใช้หน่วยความจำหลายๆ ตัวมาช่วยกันเก็บข้อมูล เพื่อให้มีประสิทธิภาพและความปลอดภัยในการเก็บข้อมูล มีวิธีการป้องกันข้อมูลภายในหน่วยความจำสูญหายได้หลายวิธีดังงานวิจัยต่อไปนี้ การเก็บข้อมูลซ้ำ (Replication) ซึ่งเป็นการเก็บข้อมูลซ้ำไว้ในหน่วยความจำอีกชุดหนึ่ง ตัวอย่างเช่น Google File System [15] Dynamo [16] และ Windows Azure Storage [17] และการใช้รหัสอีเรเซอร์ (ตัวอย่างเช่น รหัสรีด-โซโลมอน (Reed-Solomon) [4] รหัส EVENODD [5] หรือรหัส CGR (Complete Graph of Rings) [6]) ในการเข้ารหัสข้อมูลก่อนเก็บไปยังหน่วยความจำ แต่มีข้อเสียคือความเร็วในการรับส่งข้อมูล (IO speed) จะต่ำกว่าเนื่องจากข้อมูลจะต้องผ่านการเข้า/ถอดรหัสก่อนนำมาใช้งาน ตัวอย่างเช่น OceanStore [18] ซึ่งเป็นระบบเก็บข้อมูลแบบกระจายที่ข้อมูลจะถูกเข้ารหัสด้วยรหัสอีเรเซอร์ (รหัสรีด-โซโลมอน) จากนั้นกระจายเก็บในหน่วยความจำได้ทุกแห่ง ซึ่งในระบบเก็บข้อมูลนั้นมีโอกาสที่จะเสียหายได้ตลอดเวลา และ Total Recall [19] ซึ่งเป็นระบบอัตโนมัติที่ตรวจสอบและคำนวณกระบวนการสำรองข้อมูล โดยคาดการณ์จากพฤติกรรมของระบบเก็บข้อมูลในอดีตเพื่อให้ระบบมีประสิทธิภาพสูงที่สุด สรุปได้ดังตารางที่ 2.1

การถอดรหัสด้วยความน่าจะเป็นสูงที่สุด (Maximum likelihood decoder) เป็นวิธีการถอดรหัสที่เป็นที่นิยม เนื่องจากมีความซับซ้อนต่ำ และให้ผลที่ดี ซึ่งมีงานวิจัยจำนวนมากที่ใช้วิธีการนี้ ยกตัวอย่างเช่น งานวิจัยชิ้นแรกที่ใช้การถอดรหัสด้วยความน่าจะเป็นสูงที่สุดในการตรวจสอบอาการชักกระตุกของเด็กแรกเกิดจากการกระมวลผลวิดีโอ ซึ่งอาการชักกระตุกจะทำให้ส่วนต่างๆ ของร่างกายเช่น แขนและขา เคลื่อนไหวซ้ำไปมาๆ ซึ่งงานวิจัยนี้จะตรวจสอบการเคลื่อนไหวของในส่วน

นั้นๆ เพื่อวินิจฉัยอาการชักกระตุกในเบื้องต้น ซึ่งยังมีความคลาดเคลื่อนในกรณีที่งานวิจัยนี้ไม่สามารถจำแนกอาการชักกระตุกกับอาการเคลื่อนไหวปกติได้ [20] งานวิจัยชิ้นต่อมาใช้การถอดรหัสด้วยความน่าจะเป็นสูงที่สุดในการแบ่งกลุ่มเพื่อบอกปริมาณของทรัพยากรจากภาพถ่ายทางอากาศ โดยในกรณีที่ภาพถ่ายมีจุดภาพที่คล้ายคลึงกันมากจะทำให้การแบ่งกลุ่มนั้นอาจมีประสิทธิภาพไม่มากเท่าที่ควร [21] และงานวิจัยชิ้นสุดท้ายที่ใช้กระบวนการถอดรหัสด้วยความน่าจะเป็นสูงที่สุดสำหรับตรวจจับสัญญาณสหสัมพันธ์ไขว้ (Cross-correlation) เพื่อลดค่าหน่วงเวลาให้น้อยลง (low latency) จึงทำให้การประยุกต์ใช้ในงานอินเทอร์เน็ตของทุกสิ่งแบนด์แคบ (Narrow band Internet of things : NB-IoT) ประหยัดพลังงานมากยิ่งขึ้นแต่มีความซับซ้อนในการสร้าง ซึ่งแตกต่างจากการตรวจจับสัญญาณสหสัมพันธ์ตัวเอง (Auto-correlation) ที่มีความซับซ้อนน้อยกว่า [22] จากที่กล่าวมากระบวนการถอดรหัสด้วยความน่าจะเป็นสูงที่สุดเป็นกระบวนการที่ไม่ซับซ้อนแต่มีประโยชน์มากมายจึงถูกนำมาประยุกต์ใช้งานกับระบบติดต่อสื่อสารที่หลากหลายสรุปได้ดังตารางที่ 2.2

ตารางที่ 2.1 งานวิจัยเกี่ยวกับการกู้ข้อมูลในระบบเก็บข้อมูลแบบกระจาย

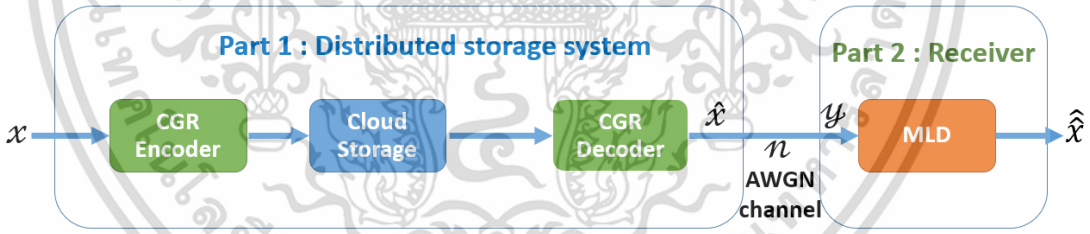
กระบวนการ	ตัวอย่าง	ผลงาน	ข้อเสีย
Replication	Google File System [15] Dynamo [16] Windows Azure [17]	เก็บข้อมูลซ้ำในหน่วยความจำอีกหนึ่งชุด กรณีที่หน่วยความจำเสียหายจะสามารถนำข้อมูลจากหน่วยความจำอีกชุดหนึ่งมาใช้งานได้	สิ้นเปลืองทรัพยากรหน่วยความจำที่นำมาเก็บข้อมูลซ้ำ
Erasure Code	OceanStore [18] Total Recall [19]	ข้อมูลถูกเข้ารหัสก่อนนำไปเก็บในหน่วยความจำ กรณีที่หน่วยความจำเสียหายจะสามารถถอดรหัสจากข้อมูลในหน่วยความจำที่เหลือกู้ข้อมูลกลับคืนมาได้	ความเร็วในการรับส่งข้อมูล (IO speed) ต่ำ

ตารางที่ 2.2 งานวิจัยที่ใช้กระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุด

งานวิจัย	ผลงาน	ข้อเสีย
Maximum-Likelihood Detection of Neonatal Clonic Seizures by Video Image Processing [20]	ใช้ MLD เพื่อตรวจสอบอาการชักกระตุกของเด็กแรกเกิดโดยใช้การประมวลผลภาพทางวีดีโอ	มีความผิดพลาดในการจำแนกการเคลื่อนไหวปกติเป็นอาการชักกระตุก
Class quantification of aerial images using Maximum Likelihood Estimation [21]	ใช้ MLD แบ่งกลุ่มบอกปริมาณของทรัพยากรธรรมชาติผ่านทางภาพถ่ายทางอากาศ	กรณีที่จุดภาพมีความคล้ายคลึงกันจะทำให้การแบ่งกลุ่มบอกปริมาณนั้นไม่มีประสิทธิภาพเท่าที่ควร
Maximum-Likelihood Detection for Energy-Efficient Timing Acquisition in NB-IoT [22]	ใช้ MLD สำหรับตรวจจับสัญญาณสหสัมพันธ์ไขว้เพื่อลดค่าหน่วยเวลาให้น้อยลง เพื่อให้ NB-IoT ประหยัดพลังงานมากยิ่งขึ้น	มีความซับซ้อนมากกว่าเมื่อเทียบกับการตรวจจับสัญญาณสหสัมพันธ์ตัวเอง

บทที่ 3
การศึกษาระบบจัดเก็บข้อมูลแบบกระจายและอัลกอริทึมการตรวจจับความน่าจะเป็นสูงสุด

บทนี้เป็นบทที่กล่าวถึงการศึกษาระบบจัดเก็บข้อมูลแบบกระจายและอัลกอริทึมที่ใช้ในการวิจัยซึ่งคืออัลกอริทึมการตรวจจับความน่าจะเป็นสูงสุด ซึ่งในงานวิจัยนี้มีแบบจำลองการทดลองแสดงดังรูปที่ 3.1 โดยจะกำหนดให้ข้อมูลดั้งเดิม (x) ถูกเข้ารหัสด้วยรหัส CGR ก่อนเพื่อเก็บในหน่วยเก็บข้อมูลแบบกลุ่มเมฆ เมื่อข้อมูลในหน่วยเก็บข้อมูลแบบกลุ่มเมฆถูกใช้งาน ข้อมูลนั้นจะถูกถอดรหัส CGR ก่อนส่งไปยังผู้ใช้งาน ($\hat{x}$) ระหว่างข้อมูลถูกส่งไปยังผู้รับจะผ่านช่องสัญญาณที่มีสัญญาณรบกวนแบบเกาส์เซียนขาวบวก (AWGN : Additive White Gaussian Noise) ซึ่งถูกรบกวนจากสัญญาณรบกวน (n) ทำให้เกิดข้อมูลรับที่มีความผิดพลาด (y) โดยในส่วนของเครื่องรับจะเพิ่มกระบวนการ MLD มาช่วยแก้ไขข้อมูลผิดพลาดนั้นได้ข้อมูลในส่วนของฝั่งรับ ($\hat{x}$) โดยข้อมูลที่สนใจนำมาทดลองมี 3 ประเภท คือ ข้อมูลภาพแบบไบนารี่ ข้อมูลภาพแบบนอน-ไบนารี่ และข้อมูลแบบเอกสาร



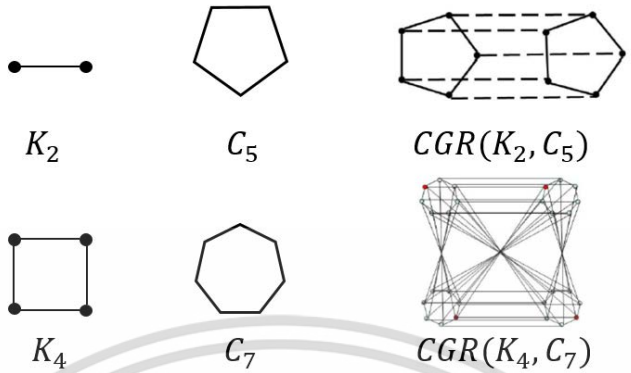
รูปที่ 3.1 แบบจำลองการทดลอง

3.1 ระบบจัดเก็บข้อมูลแบบกระจาย

3.1.1 รหัส CGR สำหรับระบบจัดเก็บข้อมูลแบบกระจาย

ทางผู้วิจัยได้ศึกษารหัส CGR (Complete Graph of Ring code) [6] ซึ่งเป็นรหัสอีเรเซอร์ที่พัฒนาบนระบบเก็บข้อมูลแบบกลุ่มเมฆ เนื่องจากรหัสนี้สามารถสร้างให้อยู่ในรูปของกราฟเส้นเชื่อมและสามารถนำมาเขียนให้อยู่ในรูปของกราฟสองส่วน (Bipartite graph) ที่เป็นโครงสร้างในการเข้ารหัสรูปแบบเดียวกับรหัสแอลดีพีซี (low-density parity check code) ซึ่งง่ายต่อความเข้าใจ อีกทั้งยังเป็นรหัสระยะห่างสูงสุดหรือรหัสเอ็มดีเอส (Maximum distance separable code) ชนิดหนึ่ง

เขียนแทนด้วยสัญลักษณ์ (K_{v_1}, C_{v_2}) โดย $v_2 = v_1 + 3$ เมื่อ v_1 คือจำนวนของวงกราฟเส้นเชื่อม และ v_2 คือจำนวนจุดหรือขอบของกราฟเส้นเชื่อมในแต่ละวงดังรูปที่ 3.2



รูปที่ 3.2 ตัวอย่างกราฟ $CGR(K_2, C_5)$ และ $CGR(K_4, C_7)$ [6]

ตัวอย่างเช่น รหัส CGR (K_2, C_5) สามารถเขียนให้อยู่ในรูปของรหัสแถวลำดับ (Array code) ซึ่งจะได้ แถวลำดับดังรูปที่ 3.3 โดยแต่ละหลักเปรียบได้กับจำนวนของหน่วยความจำ และแต่ละแถวเปรียบได้กับจำนวนชุดของข้อมูลและข้อมูลตรวจสอบ สำหรับรหัส CGR (K_2, C_5) นี้ประกอบด้วยข้อมูลเก็บในหน่วยความจำจำนวน 2 ตัวและเพิ่มข้อมูลตรวจสอบในหน่วยความจำอีก 3 ตัว ซึ่งจะต้องใช้หน่วยความจำรวมทั้งหมด 5 ตัวในการเก็บข้อมูลชุดนี้ โดยรหัสนี้มีคุณสมบัติคือสามารถกู้ข้อมูลจากกรณีหน่วยความจำเสียหายพร้อมกันได้ถึง 3 ตัว ซึ่งเท่ากับจำนวนหน่วยความจำที่ใช้สำหรับเก็บข้อมูลตรวจสอบ จึงเรียกได้ว่ารหัสนี้มีคุณสมบัติ Singleton bound

0	1	2	3	4
6	7	8	9	5
$2 \oplus 3$	$3 \oplus 4$	$4 \oplus 0$	$0 \oplus 1$	$1 \oplus 2$
$7 \oplus 8$	$8 \oplus 9$	$9 \oplus 5$	$5 \oplus 6$	$6 \oplus 7$
$4 \oplus 9$	$0 \oplus 5$	$1 \oplus 6$	$2 \oplus 7$	$3 \oplus 8$

รูปที่ 3.3 รหัสแถวลำดับของ $CGR(K_2, C_5)$

เพื่อให้เข้าใจกระบวนการกู้คืนข้อมูลได้ทั้งหมดกรณีที่หน่วยความจำเสียหาย 1-3 ตัว และไม่สามารถกู้คืนข้อมูลได้กรณีที่หน่วยความจำเสียหายมากกว่าหรือเท่ากับ 4 ตัว (ซึ่งไม่อยู่ในเงื่อนไขของรหัส CGR(K_2, C_5)) ผู้วิจัยได้ยกตัวอย่างกรณีที่หน่วยความจำเสียหาย (ค่าในแต่ละหลักหายไป) ดังตัวอย่างต่อไปนี้

1. หน่วยความจำเสียหาย 1 ตัว

ยกตัวอย่างให้หลักที่ 1 หายไป ซึ่งทำให้ข้อมูล 0 และ 6 หายไปด้วย แต่จะสามารถกู้ข้อมูลคืนได้โดย ข้อมูล 0 สามารถหาได้จากการทำ XOR ของข้อมูล 5 กับ ข้อมูลตรวจสอบในแถวที่ 5 หลักที่ 2 และข้อมูล 6 สามารถหาได้จากการทำ XOR ของข้อมูล 1 กับ ข้อมูลตรวจสอบในแถวที่ 5 หลักที่ 3

2. หน่วยความจำเสียหาย 2 ตัว

ยกตัวอย่างให้หลักที่ 1 และหลักที่ 2 หายไป ซึ่งทำให้ข้อมูล 0, 1, 6 และ 7 หายไปด้วย แต่จะสามารถกู้คืนข้อมูลได้โดย ข้อมูล 7 สามารถหาได้จากการทำ XOR ของข้อมูล 2 กับ ข้อมูลตรวจสอบในแถวที่ 5 หลักที่ 4 ข้อมูล 1 สามารถหาได้จากการทำ XOR ของข้อมูล 2 กับ ข้อมูลตรวจสอบในแถวที่ 3 หลักที่ 5 และข้อมูล 0 และ 6 สามารถหาได้เช่นเดียวกับกรณีหน่วยความจำเสียหาย 1 ตัว

3. หน่วยความจำเสียหาย 3 ตัว

ยกตัวอย่างให้หลักที่ 1, 2 และหลักที่ 3 หายไป ซึ่งทำให้ข้อมูล 0, 1, 2, 6, 7 และ 8 หายไปด้วย แต่จะสามารถกู้คืนข้อมูลได้โดย ข้อมูล 6 สามารถหาได้จากการทำ XOR ของข้อมูล 5 กับ ข้อมูลตรวจสอบในแถวที่ 4 หลักที่ 4 ข้อมูล 7 สามารถหาได้จากการทำ XOR ของข้อมูล 6 กับ ข้อมูลตรวจสอบในแถวที่ 4 หลักที่ 5 ข้อมูล 8 สามารถหาได้จากการทำ XOR ของข้อมูล 3 กับ ข้อมูลตรวจสอบในแถวที่ 5 หลักที่ 5 ข้อมูล 2 สามารถหาได้จากการทำ XOR ของข้อมูล 7 กับ ข้อมูลตรวจสอบในแถวที่ 5 หลักที่ 4 ข้อมูล 1 สามารถหาได้จากการทำ XOR ของข้อมูล 2 กับ ข้อมูลตรวจสอบในแถวที่ 3 หลักที่ 5 ข้อมูล 0 สามารถหาได้จากการทำ XOR ของข้อมูล 1 กับ ข้อมูลตรวจสอบในแถวที่ 3 หลักที่ 4

4. หน่วยความจำเสียหาย 4 ตัว

ยกตัวอย่างให้หลักที่ 1, 2, 3 และหลักที่ 4 หายไป ซึ่งทำให้ข้อมูล 0, 1, 2, 3, 6, 7, 8 และ 9 หายไปด้วย ซึ่งเหลือเพียงข้อมูล 4 และ 5 โดยเมื่อพิจารณาข้อมูลตรวจสอบที่เหลือในหน่วยความจำนั้นไม่เพียงพอต่อการกู้ข้อมูลที่หายไปกลับคืนมาได้ จึงทำให้เมื่อหน่วยความจำเสียหาย 4 ตัว จากระบบเก็บข้อมูลที่มีหน่วยความจำจำนวน 5 ตัว อยู่นอกเงื่อนไขในการกู้ข้อมูลด้วยรหัส รหัส CGR (K_2, C_5)

3.1.2 การตั้งค่าการทดลอง

ทางผู้วิจัยได้จำลองระบบเก็บข้อมูลแบบกระจายซึ่งประกอบด้วยหน่วยความจำขนาด 5 ตัว โดยภายในระบบเก็บข้อมูลแบบกระจายนี้มีการใช้งานรหัส CGR (K_2, C_5) เป็นรหัสอีเรเซอร์ในการป้องกันข้อมูลเสียหายภายในระบบเก็บข้อมูลนี้ โดยข้อมูลหนึ่งชุดจะถูกแบ่งเป็น 2 ส่วน (k=2) และเพิ่มข้อมูลตรวจสอบ 3 ส่วน (m=3) ซึ่งสามารถป้องกันหน่วยความจำเสียหายพร้อมกันได้สูงที่สุด 3 ตัว ทางผู้วิจัยได้สนใจนำข้อมูลชนิดไบนารี่เก็บลงในหน่วยเก็บข้อมูลแบบกระจายที่มีการใช้งานรหัส CGR (K_2, C_5) ต่อมาจำลองให้หน่วยความจำภายในระบบเก็บข้อมูลนี้เสียหาย 0-5 ตัว จากนั้นให้รหัส CGR (K_2, C_5) ถอดรหัสข้อมูลของหน่วยความจำที่เสียหายนั้นกลับมาซึ่งสามารถสรุปได้ดังตารางที่ 3.1

ตารางที่ 3.1 ค่าข้อมูลที่นำมาทดลองกับระบบเก็บข้อมูลแบบกระจาย

จำนวนข้อมูลไบนารี่ที่นำมาทดสอบ (บิต)	รหัสอีเรเซอร์ที่ใช้ในระบบเก็บข้อมูลแบบกระจาย	หน่วยความจำเสียหาย (ตัว)
100,000 200,000 300,000 400,000 500,000	รหัส CGR (K_2, C_5)	0-5

3.2 ข้อมูลภาพแบบไบนารี่กับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

3.2.1 กระบวนการทำงานของอัลกอริทึม

การถอดรหัสด้วยวิธีความน่าจะเป็นสูงสุด (Maximum Likelihood Decoder : MLD) เป็นวิธีการหนึ่งในการหาค่าโอกาสเป็นไปได้สูงที่สุด โดยถ้ากำหนดให้ X เป็นตัวแปรสุ่มที่มีเหตุการณ์คือ $X = \{x_1, x_2, x_3, \dots, x_n\}$ เมื่อ n คือเหตุการณ์รวมทั้งหมด และ μ คือค่าเหตุการณ์ที่เราพิจารณา ซึ่งจะได้สมการ Likelihood ตามสมการที่ 3.1

$$P(x_1, x_2, \dots, x_n ; \mu) = \prod_{i=1}^{n} p(x_i ; \mu) \tag{3.1}$$

เมื่อ $P(X)$ คือ joint probability mass function

เมื่อพิจารณาข้อมูลใน Binary symmetric channel ซึ่ง $x_i = \{0,1\}$ สามารถเปรียบให้ข้อมูลมีการแจกแจงแบบแบร์นูลลี่ (Bernoulli Random Variable) ได้ดังสมการที่ 3.2

$$P(x_1, x_2, \dots, x_n ; \mu) = \prod_{i=1}^{n} p(x_i ; \mu) = \prod_{i=1}^{n} \mu^{x_i} (1-\mu)^{1-x_i} \tag{3.2}$$

เมื่อการแจกแจงแบบแบร์นูลลี่มีค่าดังสมการที่ 3.3 และ 3.4

$$P(x = 1) = \mu \tag{3.3}$$

$$P(x = 0) = 1 - \mu \tag{3.4}$$

เมื่อ $0 < \mu < 1$

หาค่าสูงสุดของสมการที่ 3.2 โดยการใส่ค่าลอการิทึม (logarithm)

$$\begin{aligned} \log p(X ; \mu) &= \log \textstyle\prod_{i=1}^{n} \mu^{x_i} (1-\mu)^{1-x_i} \\ &= \textstyle\sum_{i=1}^{n} \log\{\mu^{x_i}(1-\mu)^{1-x_i}\} \\ &= \textstyle\sum_{i=1}^{n} [\log \mu^{x_i} + \log(1-\mu)^{1-x_i}] \\ \log p(X ; \mu) &= \textstyle\sum_{i=1}^{n} [x_i \log \mu + (1-x_i)\log(1-\mu)] \end{aligned} \tag{3.5}$$

หาค่าสูงสุดของสมการที่ 3.5

$$\underset{\mu}{\operatorname{argmax}} \log p(X ; \mu) = \underset{\mu}{\operatorname{argmax}} \textstyle\sum_{i=1}^{n} [x_i \log \mu + (1-x_i)\log(1-\mu)] \tag{3.6}$$

หาอนุพันธ์ลำดับที่ 1 ของสมการที่ 3.6

$$\begin{aligned} \frac{\partial}{\partial \mu} \log p(X ; \mu) &= \textstyle\sum_{i=1}^{n} \frac{\partial}{\partial \mu} [x_i \log \mu + (1-x_i)\log(1-\mu)] \\ &= \textstyle\sum_{i=1}^{n} x_i \frac{\partial}{\partial \mu} \log \mu + \sum_{i=1}^{n} (1-x_i) \frac{\partial}{\partial \mu} \log(1-\mu) \\ \frac{\partial}{\partial \mu} \log p(X ; \mu) &= \frac{1}{\mu} \textstyle\sum_{i=1}^{n} x_i - \frac{1}{1-\mu} \sum_{i=1}^{n} (1-x_i) \end{aligned} \tag{3.7}$$

กำหนดให้สมการที่ 3.7 มีค่าเท่ากับศูนย์เพื่อหาค่าสูงสุด

$$\begin{aligned} 0 &= \frac{1}{\mu} \textstyle\sum_{i=1}^{n} x_i - \frac{1}{1-\mu} \sum_{i=1}^{n} (1-x_i) \\ \frac{1-\mu}{\mu} &= \frac{\sum_{i=1}^{n}(1-x_i)}{\sum_{i=1}^{n} x_i} \\ \frac{1}{\mu} - 1 &= \frac{\sum_{i=1}^{n} 1}{\sum_{i=1}^{n} x_i} - 1 \\ \frac{1}{\mu} &= \frac{n}{\sum_{i=1}^{n} x_i} \\ \hat{\mu}_{Max} &= \frac{1}{n} \textstyle\sum_{i=1}^{n} x_i \end{aligned} \tag{3.8}$$

การประยุกต์ใช้กระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบไบนารี่ คือ เมื่อกำหนดบิตรอบข้าง 8 บิต ของบิต C ตามรูปที่ 3.4 เมื่อ $x_i = \{0,1\}$

x_1	x_2	x_3
x_4	C	x_5
x_6	x_7	x_8

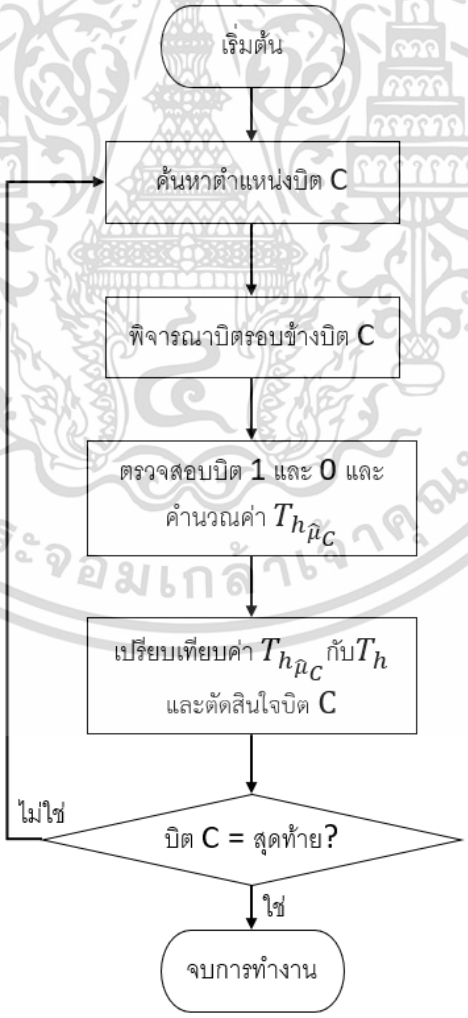
รูปที่ 3.4 ภาพจำลองการพิจารณาบิตรอบข้างของ MLD

สามารถหาค่าความน่าจะเป็นที่บิต C จะมีค่าเป็น 0 และ 1 ได้จากสมการที่ 3.9 และ 3.10

$$\hat{\mu}_{C=1} = \frac{1}{8}\sum_{i=1}^{n} x_i \quad (3.9)$$

$$\hat{\mu}_{C=0} = 1 - \hat{\mu}_{C=1} \quad (3.10)$$

อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบไบนารี่สามารถสรุปได้ดังผังงาน (Flowchart) ดังรูปที่ 3.5



รูปที่ 3.5 ผังงานอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบไบนารี่

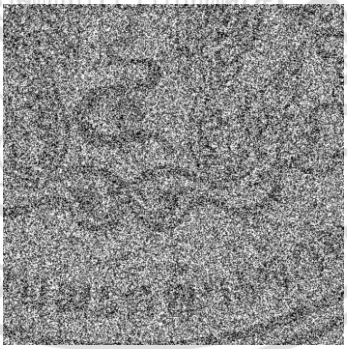
ค่าขีดเริ่มเปลี่ยน (Threshold : T_h) คือค่าที่ใช้เปรียบเทียบกับ $\hat{\mu}_C$ โดยจะตัดสินใจพิจารณาแก้ไขบิต C ให้เท่ากับ 1 เมื่อ $\hat{\mu}_{C=1}$ หรือ C ให้เท่ากับ 0 เมื่อ $\hat{\mu}_{C=0}$ โดยเปรียบเทียบกับค่าขีดเริ่มเปลี่ยน (Threshold) ของ MLD ผู้วิจัยหลีกเลี่ยงค่า $T_h = 1/2$ โดยสัญชาตญาณเนื่องจากค่าดังกล่าวเปรียบได้เหมือนการสุ่มค่า C ดังนั้นในงานวิจัยนี้จึงเลือกใช้ค่า $T_h > 1/2$, $T_h \geq 3/4$ และ $T_h > 3/4$ โดยเมื่อค่า $T_h > 1/2$ ซึ่งหมายถึงค่าความน่าจะเป็นของบิต C $\hat{\mu}_{C=1}$ หรือ $\hat{\mu}_{C=0}$ มีค่ามากกว่า $1/2$ จึงเปลี่ยนค่าบิต C ตามค่านั้นๆ โดยถ้า $\hat{\mu}_C$ ไม่เข้าเงื่อนไขดังกล่าว จะคงค่าบิต C เป็นค่าเดิม

3.2.2 การตั้งค่าการทดลอง

ทางผู้วิจัยได้นำข้อมูล 3 ชุดแตกต่างกันซึ่งประกอบด้วย ข้อมูลไบนารี่บิตจากการสุ่มขนาด 600x600 บิตดังรูปที่ 3.6 และภาพขาวดำชนิด Bitmap ขนาด 600x600 จุดภาพ จำนวน 50 ภาพ ที่แตกต่างกัน 2 ชนิด คือ ภาพขาวดำที่มีเส้นขอบเบาบาง และภาพขาวดำที่มีเส้นขอบหนา ดังรูปที่ 3.7 และรูปที่ 3.8 มาปรับให้อยู่ในรูปแบบของ bitmap โดยทดลองในสถานการณ์จำลองที่หน่วยเก็บข้อมูลแบบกลุ่มก้อนเมฆที่ใช้รหัส CGR (K_2, C_5) โดยข้อมูลหนึ่งชุดจะถูกแบ่งเป็น 2 ส่วน (k=2) และเพิ่มข้อมูลตรวจสอบ 3 ส่วน (m=3) ซึ่งสามารถป้องกันหน่วยความจำเสียหายพร้อมกันได้สูงที่สุด 3 ตัว ผ่านช่องสัญญาณที่มีสัญญาณรบกวนแบบเกาส์เซียนขาวบวก (AWGN : Additive White Gaussian Noise) มายังผู้รับซึ่งใช้ MLD ในการแก้ไขข้อมูลผิดพลาดเบื้องต้น ซึ่งอัลกอริทึม MLD สามารถเลือกใช้ค่าขีดเริ่มเปลี่ยนให้เหมาะสมกับสภาพของช่องสัญญาณได้ โดยในการทดลองจะเลือกใช้ MLD ที่มีค่า $T_h > 1/2$, $T_h \geq 3/4$ และ $T_h > 3/4$ โดยเลือกใช้ข้อมูลทั้งหมด 4 รูปแบบคือ ภาพจากข้อมูลไบนารี่แบบสุ่ม ภาพขาวดำที่มีเส้นขอบเบาบาง 50 รูป ภาพขาวดำที่มีเส้นขอบหนา 50 รูป และภาพขาวดำทั้งสองแบบสุ่มเลือกมาทั้งหมด 50 รูป ซึ่งสรุปได้ดังตารางที่ 3.2

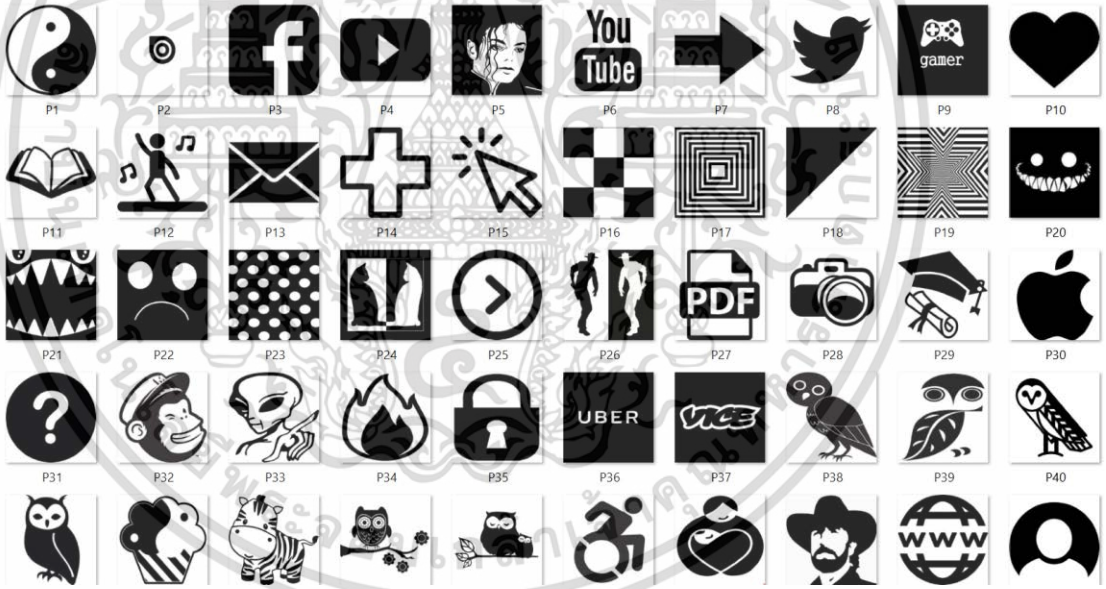
ตารางที่ 3.2 ค่าข้อมูลรูปแบบไบนารี่ที่นำมาทดลองกับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

รูปแบบข้อมูลรูปภาพ	หน่วยความจำเสียหาย (ตัว)	ค่าขีดเริ่มเปลี่ยนของ MLD (T_h)
ภาพจากข้อมูลไบนารี่แบบสุ่ม	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$
ภาพขาวดำที่มีเส้นขอบเบาบาง (แบบที่ 1)	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$
ภาพขาวดำที่มีเส้นขอบหนา (แบบที่ 2)	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$
ภาพขาวดำแบบที่ 1+2	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$



รูปที่ 3.6 ภาพตัวอย่างจากข้อมูลไบนารี่แบบสุ่มขนาด 600x600 จุดภาพ

รูปที่ 3.7 ภาพขาวดำที่มีเส้นขอบเบาบาง



รูปที่ 3.8 ภาพขาวดำที่มีเส้นขอบหนา

3.2.3 การศึกษาอัตราส่วนของข้อมูลไบนารี่ 0 และ 1 กับผลกระทบของอัลกอริทึม

ข้อมูลรูปภาพขาวดำแบบไบนารี่นั้นมีส่วนประกอบเกิดจากบิต 0 แทนสีดำ และบิต 1 แทนสีขาวมาประกอบกันเป็นรูปภาพ โดยในหัวข้องานวิจัยชิ้นนี้จะทำการศึกษาอัตราส่วนของบิต 0 หรือบิต 1 ต่อบิตทั้งหมดของรูปภาพ เพื่อศึกษาประสิทธิภาพของอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดดังที่ผู้วิจัยได้เสนอในหัวข้อที่ 3.2 เพื่อให้สามารถเลือกค่าพารามิเตอร์ของอัลกอริทึมให้เหมาะสมกับชนิดของภาพและสภาพแวดล้อมเพื่อให้อัลกอริทึมนี้เกิดประสิทธิภาพสูงที่สุด

คำนวณอัตราส่วนของบิต 1 ต่อบิตทั้งหมดของรูปภาพได้จากสมการ

$$Ratio_{\frac{1}{all}} = \sum_{i=1}^{n} \frac{x_i}{n} \tag{3.11}$$

เมื่อ $x_i = \{0,1\}$ และ n คือจำนวนบิตของรูปภาพทั้งหมด

ผู้วิจัยได้นำรูปภาพขาวดำแบบไบนารี่ขนาด 600x600 จุดภาพ จำนวน 120 ภาพ ดังรูปที่ 3.9 จากนั้นแบ่งกลุ่มของรูปภาพตามอัตราส่วนของบิต 1 ต่อบิตของรูปภาพทั้งหมด จากนั้นข้อมูลถูกส่งผ่านช่องสัญญาณซึ่งมีสัญญาณรบกวนแบบเกาส์เซียนขาวบวก (AWGN : Additive White Gaussian Noise) ที่ SNR 0-20 dB มายังผู้รับ โดยผู้รับใช้อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดด้วยค่าขีดเริ่มเปลี่ยน (T_h) ของ MLD ที่ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$ และ $T_h > \frac{3}{4}$ เพื่อแก้ไขข้อมูลผิดพลาดที่มาจากช่องสัญญาณดังตารางที่ 3.3 โดยทางผู้วิจัยจะเก็บผลการทดลองมาเพื่อวิเคราะห์ค่าพารามิเตอร์, อัตราส่วนของบิต 1 ต่อบิตทั้งหมดของรูปภาพ และคุณภาพของช่องสัญญาณ เพื่อหาค่าที่เหมาะสมเพื่อประสิทธิภาพสูงสุด

ตารางที่ 3.3 ค่าพารามิเตอร์ต่างๆสำหรับทดสอบผลของอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

อัตราส่วนบิต 1 ต่อบิตของรูปภาพทั้งหมด (รูปภาพขนาด 600x600 pixel)	คุณภาพของช่องสัญญาณ (SNR)	ค่าขีดเริ่มเปลี่ยนของ MLD (T_h)
1. $0 < Ratio_{\frac{1}{all}} \leq 0.2$ 2. $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$ 3. $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$ 4. $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ 5. $0.8 < Ratio_{\frac{1}{all}} \leq 1$	0-20 dB	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$

รูปที่ 3.9 ภาพขาวดำแบบไบนารี่

3.3 ข้อมูลภาพแบบนอน-ไบนารี่กับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

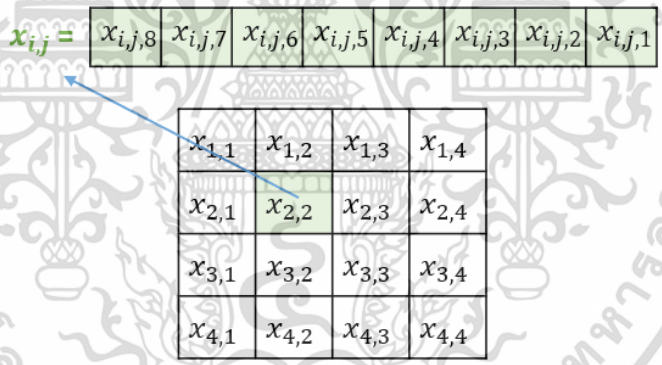
3.3.1 กระบวนการทำงานของอัลกอริทึม

จากหัวข้อที่ 3.2 อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดถูกใช้งานในการแก้ไขบิตในข้อมูลภาพแบบไบนารี่ กล่าวคือที่จุดภาพในแต่ละจุดนั้นมีค่า 0 หรือ 1 เมื่อกำหนดให้ภาพแบบไบนารี่มีขนาด $i \times j$ จุดภาพ และแต่ละจุดภาพมีค่า $x_{i,j} = \{0,1\}$ ดังรูปที่ 3.10

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$

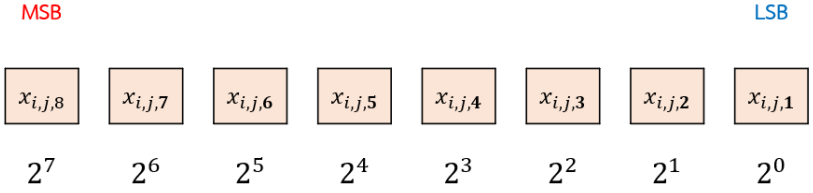
รูปที่ 3.10 ภาพจำลองจุดภาพของข้อมูลภาพแบบไบนารี่

สำหรับข้อมูลภาพแบบนอน-ไบนารี่นั้นยกตัวอย่างเช่น Grayscale ชนิด 8 บิต หรือ RGB ชนิด 8 บิต ซึ่งข้อมูลแต่ละจุดภาพจะประกอบด้วยบิต 0 หรือ 1 จำนวน 8 บิต ดังรูปที่ 3.11



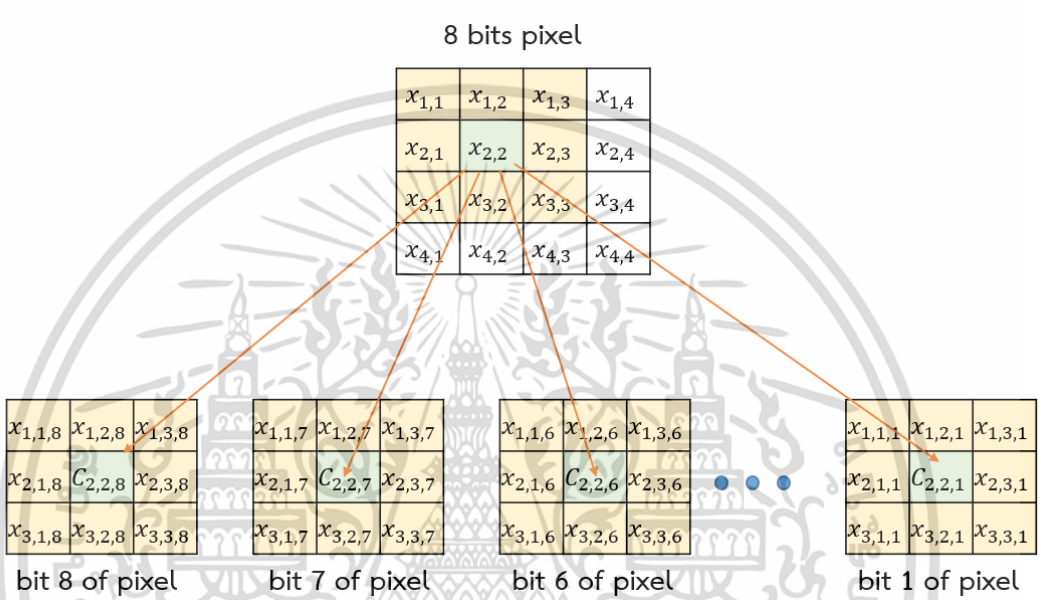
รูปที่ 3.11 ภาพจำลองจุดภาพของข้อมูลภาพแบบนอน-ไบนารี่

จากรูปที่ 3.11 จากข้อมูลทั้ง 8 บิตนั้นเมื่อคำนวณเป็นค่าตัวเลขฐาน 10 จะมีค่าระหว่าง 0-255 โดยค่าแต่ละบิตมีค่าดังรูปที่ 3.12



รูปที่ 3.12 รายละเอียดแต่ละบิตของข้อมูล 8 บิต

เมื่อพิจารณาข้อมูลจุดภาพของภาพแบบนอน-ไบนารี่แล้วจะพบว่าที่จุดภาพที่ใกล้เคียงกัน จะมีค่าบิตที่ใกล้เคียงกัน กล่าวคือลักษณะของภาพแบบนอน-ไบนารี่นั้นมีโครงสร้างพื้นฐานคล้ายๆกับภาพแบบไบนารี่ ซึ่งข้อมูลบิตแต่ละบิตมีความสัมพันธ์ซึ่งกันและกัน ซึ่งทางผู้วิจัยได้ประยุกต์ใช้กระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับภาพแบบนอน-ไบนารี่โดยใช้หลักการเดียวกับภาพแบบไบนารี่ดังสมการที่ 3.8 หัวข้อที่ 3.2 โดยจะพิจารณาบิตรอบข้างของภาพแบบนอน-ไบนารี่ทีละบิต ดังรูปที่ 3.13



รูปที่ 3.13 ภาพจำลองการพิจารณาบิตรอบข้างของ MLD กับข้อมูลภาพแบบนอน-ไบนารี่

จากสมการที่ 3.8 ในหัวข้อที่ 3.2 เมื่อกำหนดให้บิตรอบข้างบิตที่พิจารณา คือ $x_1, x_2, x_3, \dots, x_n$ สามารถเขียนใหม่ได้ดังสมการที่ 3.12

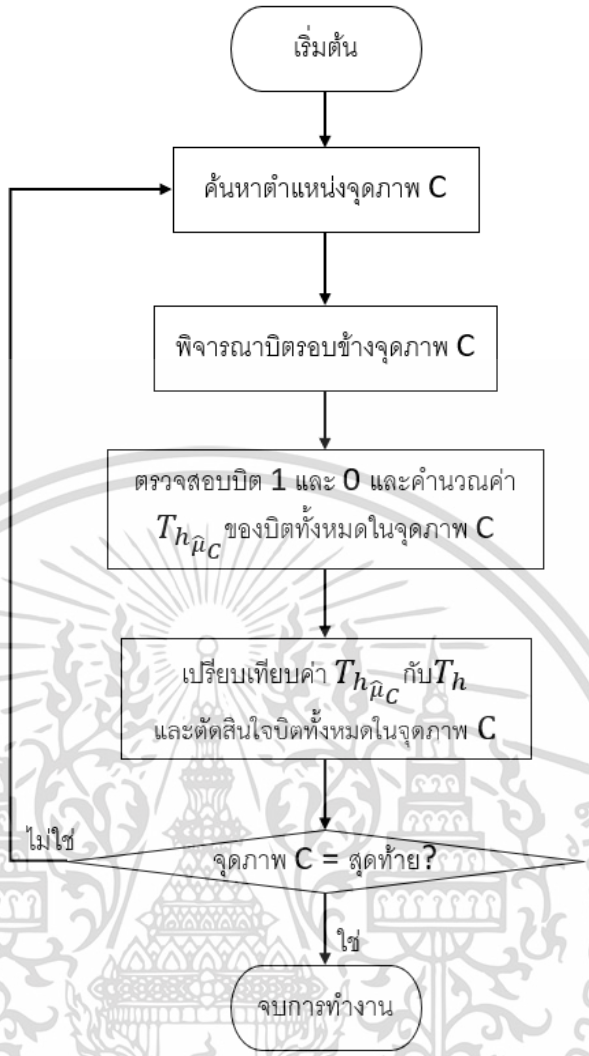
$$\hat{\mu}_{k_{Max}} = \frac{1}{n}\sum_{i=1}^{n} x_{i,k} \tag{3.12}$$

จากสมการที่ 3.12 เมื่อหาความน่าจะเป็นของค่าบิต k เมื่อ $k = \{1, 2, 3, \dots, 8\}$ บนจุดภาพที่จะพิจารณาว่ามีค่าเป็นบิต 0 และ 1 ได้ดังสมการที่ 3.13 และ 3.14

$$\hat{\mu}_{k_{C=1}} = \frac{1}{8}\sum_{i=1}^{n} x_{i,k} \tag{3.13}$$

$$\hat{\mu}_{k_{C=0}} = 1 - \hat{\mu}_{k_{C=1}} \tag{3.14}$$

อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบนอน-ไบนารี่สามารถสรุปได้ดังผังงาน (Flowchart) ดังรูปที่ 3.14



รูปที่ 3.14 ผังงานอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบนอน-ไบนารี่

ในส่วนของค่าขีดเริ่มเปลี่ยน (Threshold : T_h) ซึ่งจะมีลักษณะการพิจารณาคล้ายกับหัวข้อที่ 3.2 เพียงแต่จะพิจารณาแต่ละบิตแยกกันในแต่ละจุดภาพ เมื่อกำหนดให้จุดภาพแต่ละจุดประกอบด้วยไบนารี่บิต k ทั้งหมด 8 บิต โดยค่า T_h คือค่าที่ใช้เปรียบเทียบกับ $\hat{\mu}_{k_C}$ โดยจะตัดสินใจพิจารณาแก้ไขจุดภาพ C ตำแหน่งบิตที่ k ให้เท่ากับ 1 เมื่อ $\hat{\mu}_{k_{C=1}}$ หรือพิจารณาแก้ไขจุดภาพ C ตำแหน่งบิตที่ k ให้เท่ากับ 0 เมื่อ $\hat{\mu}_{k_{C=0}}$ โดยเปรียบเทียบกับค่าขีดเริ่มเปลี่ยน (Threshold) ของ MLD ซึ่งในส่วนนี้จะใช้ค่าขีดเริ่มเปลี่ยนเดียวกับหัวข้อที่ 3.2 ซึ่งประกอบด้วย $T_h > 1/2$, $T_h \geq 3/4$ และ $T_h > 3/4$

3.3.2 การตั้งค่าการทดลอง

ทางผู้วิจัยได้นำข้อมูลภาพนอน-ไบนารี่ 2 ชนิดซึ่งประกอบไปด้วย ข้อมูลภาพชนิด Grayscale และข้อมูลภาพชนิด RGB จำนวน 30 ภาพ ขนาด 600x600 จุดภาพ ดังรูปที่ 3.15 และ 3.16 ตามลำดับ ซึ่งลักษณะข้อมูลภาพชนิด Grayscale คือจุดภาพแต่ละจุดจะประกอบด้วยข้อมูลไบนารี่จำนวน 8 บิต ดังรูปที่ 3.17 ซึ่งแต่ละบิตแสดงถึงความเข้มของสีดำ และลักษณะข้อมูลภาพชนิด RGB คือจุดภาพแต่ละจุดจะประกอบด้วยข้อมูลไบนารี่จำนวน 8 บิต ทั้งหมด 3 ชุด ดังรูปที่ 3.18 ซึ่งแต่ละบิตในข้อมูลแต่ละชุดนั้นแสดงถึงความเข้มของสีแดง สีเขียว และสีน้ำเงิน ตามลำดับ

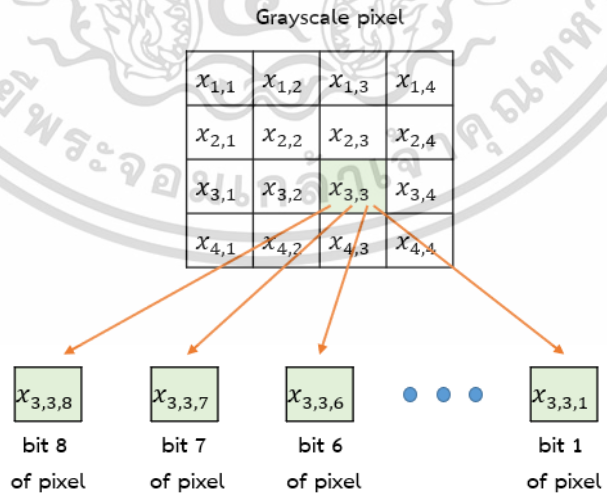
โดยทดลองในสถานการณ์จำลองที่หน่วยเก็บข้อมูลแบบกลุ่มเมฆที่ใช้รหัส CGR (K_2, C_5) โดยข้อมูลหนึ่งชุดจะถูกแบ่งเป็น 2 ส่วน (k=2) และเพิ่มข้อมูลตรวจสอบ 3 ส่วน (m=3) ซึ่งสามารถป้องกันหน่วยความจำเสียหายพร้อมกันได้สูงที่สุด 3 ตัว ผ่านช่องสัญญาณที่มีสัญญาณรบกวนแบบเกาส์เซียนขาวบวก (AWGN : Additive White Gaussian Noise) มายังผู้รับซึ่งใช้ MLD ในการแก้ไขข้อมูลผิดพลาดเบื้องต้น ซึ่งอัลกอริทึม MLD สามารถเลือกใช้ค่าขีดเริ่มเปลี่ยนให้เหมาะสมกับสภาพของช่องสัญญาณได้ โดยในการทดลองจะเลือกใช้ MLD ที่มีการพิจารณาค่าพารามิเตอร์ 2 ตัวคือ ค่าขีดเริ่มเปลี่ยนคือ $T_h > 1/2$, $T_h \geq 3/4$ และ $T_h > 3/4$ และจำนวนบิตที่นำมาคำนวณ (นับจากบิตด้านซ้ายสุดหรือบิตที่ 8) ซึ่งสรุปได้ดังตารางที่ 3.4

ตารางที่ 3.4 ค่าข้อมูลรูปแบบนอน-ไบนารี่ที่นำมาทดลองกับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

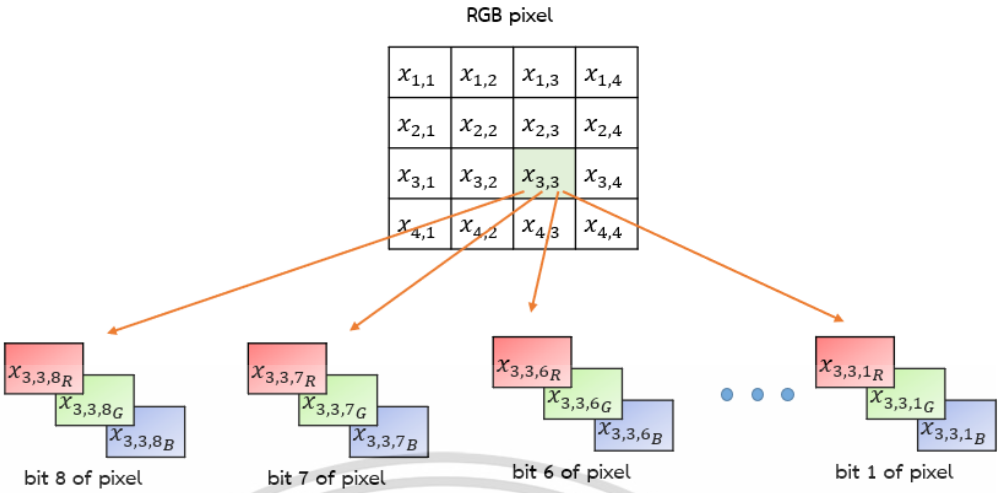
รูปแบบข้อมูลรูปภาพ	หน่วยความจำเสียหาย (ตัว)	ค่าขีดเริ่มเปลี่ยนของ MLD (T_h)	จำนวนบิตที่นำพิจารณาจาก MLD (บิต)
รูปภาพชนิด Grayscale	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$	1-8
รูปภาพชนิด RGB	0-3	1. $T_h > \frac{1}{2}$ 2. $T_h \geq \frac{3}{4}$ 3. $T_h > \frac{3}{4}$	1-8

รูปที่ 3.15 ภาพนอน-ไบนารี่ชนิด Grayscale

รูปที่ 3.16 ภาพนอน-ไบนารี่ชนิด RGB



รูปที่ 3.17 ภาพจำลองข้อมูลรูปภาพชนิด Grayscale



รูปที่ 3.18 ภาพจำลองข้อมูลรูปภาพชนิด RGB

3.4 ข้อมูลแบบเอกสารกับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

3.4.1 กระบวนการทำงานของอัลกอริทึม

อัลกอริทึมนี้ได้รับการพัฒนาต่อยอดจาก ngram [23] โดยที่ ngram คือแบบจำลองที่ใช้คำนวณค่าความน่าจะเป็นของกลุ่มตัวอักษร (Character Sequence) ที่เกิดขึ้นรวมเป็นคำ หรือความน่าจะเป็นของคำที่เรียงกัน (Word Sequence) ที่เกิดขึ้นรวมกันเป็นประโยค

เมื่อกำหนดให้ W แทนเซตของคำศัพท์ในฐานข้อมูล ซึ่งประกอบไปด้วยคำศัพท์ $W = \{w_1, w_2, w_3, \dots, w_n\}$ เมื่อ n คือจำนวนคำศัพท์ทั้งหมดในฐานข้อมูลนั้น โดยเมื่อพิจารณาคำศัพท์ลำดับที่ x ในฐานข้อมูล W ซึ่งประกอบไปด้วยตัวอักษร $w_x = \{w_{x,1}, w_{x,2}, w_{x,3}, \dots, w_{x,m}\}$ เมื่อ m คือจำนวนตัวอักษรทั้งหมดในคำนั้น และ D แทนคำที่พิจารณาแก้ไขข้อมูล ซึ่งประกอบไปด้วยตัวอักษร $D = \{d_1, d_2, d_3, \dots, d_n\}$ และ D_c แทนคำที่ถูกพิจารณาแก้ไขข้อมูลแล้ว สามารถหาความน่าจะเป็นของคำที่พิจารณา D จากทฤษฎีของเบย์ (Bayes's Theorem) ดังสมการที่ 3.15 และเมื่อเปรียบเทียบกับคำในฐานข้อมูล W สามารถหา $(p(D|W_x))$ ได้จากสมการที่ 3.16

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{3.15}$$

$$p(D|W_x) = \frac{p(D \cap W_x)}{p(W_x)} \tag{3.16}$$

พิจารณา $p(D \cap W_x)$ คือส่วนที่ข้อมูลทับซ้อนกันซึ่งจะได้ $p(\overline{D \oplus W_x})$ ดังสมการ 3.17

$$p(D|W_x) = \frac{p(\overline{D \oplus W_x})}{p(W_x)} \tag{3.17}$$

เนื่องจากความน่าจะเป็นของคำที่พิจารณาแก้ไขข้อมูลกับคำในฐานข้อมูลมี ความถี่ที่พบแตกต่างกัน จึงนำความถี่ที่พบมาพิจารณาในการคำนวณได้ดังสมการที่ 3.18

$$p_f(D|W_x) = p(f_{w_x})p(D|W_x) \tag{3.18}$$

ต่อมาหาโอกาสความน่าจะเป็นสูงสุดของคำที่ถูกพิจารณาแก้ไขข้อมูล D_c ได้จากสมการที่ 3.18

$$D_c = \text{argmax}[p_f(D|W_1, W_2, W_3, \dots W_n)] \tag{3.19}$$

เมื่อตัวอย่างคำศัพท์ในฐานข้อมูลแสดงดังตารางที่ 3.5 ข้อความที่ส่งจำนวน 18 ไบต์ และข้อความหลังผ่านสัญญาณรบกวนแสดง ดังรูปที่ 3.19 และ 3.20 ตามลำดับ

ตารางที่ 3.5 ตัวอย่างข้อมูล Ngram1

คำศัพท์ W	ความถี่ที่พบ f_{w_x}	ความน่าจะเป็นที่พบ $p(f_{w_x})$
a	20	0.25
as	8	0.10
is	15	0.19
it	11	0.14
this	10	0.13
that	12	0.15
storage	4	0.05

T	h	i	s		i	s		a		s	t	o	r	a	g	e	.

รูปที่ 3.19 ตัวอย่างข้อมูลเอกสารต้นแบบ

T	h	s	s		i	#		a		s	t	3	v	a	g	e	.

รูปที่ 3.20 ตัวอย่างข้อมูลเอกสารหลังผ่านสัญญาณรบกวน

พิจารณาสัญลักษณ์แบ่งคำซึ่งประกอบด้วย “space”, “.”, “?”, “!”, “:”, “;” เป็นต้น จากข้อมูลหลังผ่านสัญญาณรบกวนดังรูปที่ 3.17 จะสามารถแบ่งคำได้ 4 คำคือ Thss , i# , a และ st3vage โดยเมื่อพิจารณาความน่าจะเป็นของคำทั้ง 4 เทียบกับฐานข้อมูลดังตารางที่ 3.5 ซึ่งได้ดังตารางที่ 3.6

ตารางที่ 3.6 ตัวอย่างการเปรียบเทียบคำศัพท์จากฐานข้อมูล

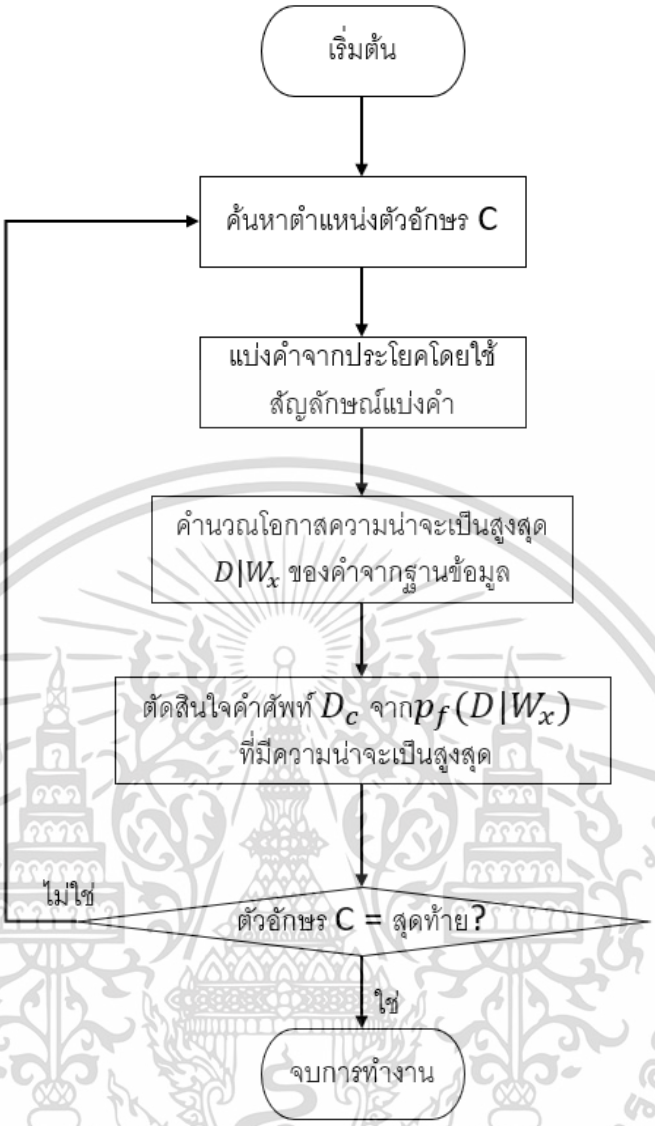
คำศัพท์ที่พิจารณา (D)	คำศัพท์จากฐานข้อมูล (W)	ความน่าจะเป็นที่พบ $p(f_{w_x})$	$p(D\|W_x)$	$p_f(D\|W_x)$
Thss	this	0.13	3/4	0.0975
	that	0.15	2/4	0.075
i#	as	0.10	0	0
	is	0.19	1/2	0.095
	it	0.14	1/2	0.07
a	a	0.25	1	0.25
St3vage	storage	0.05	5/7	0.035

จากตารางที่ 3.6 คำศัพท์ทั้ง 4 ตัวจะเลือกแก้ไขคำศัพท์เป็นคำที่มีโอกาสความน่าจะเป็นที่มีค่าสูงสุดซึ่งจะได้คำหลังจากการแก้ไขข้อมูลผิดพลาดแล้วดังรูปที่ 3.21

T	h	i	s		i	s		a		s	t	o	r	a	g	e	.

รูปที่ 3.21 ตัวอย่างข้อมูลเอกสารหลังแก้ไขข้อมูลผิดพลาด

อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลแบบเอกสารสามารถสรุปได้ดังผังงาน (Flowchart) ดังรูปที่ 3.22



รูปที่ 3.22 ผังงานอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลแบบเอกสาร

3.4.2 การตั้งค่าการทดลอง

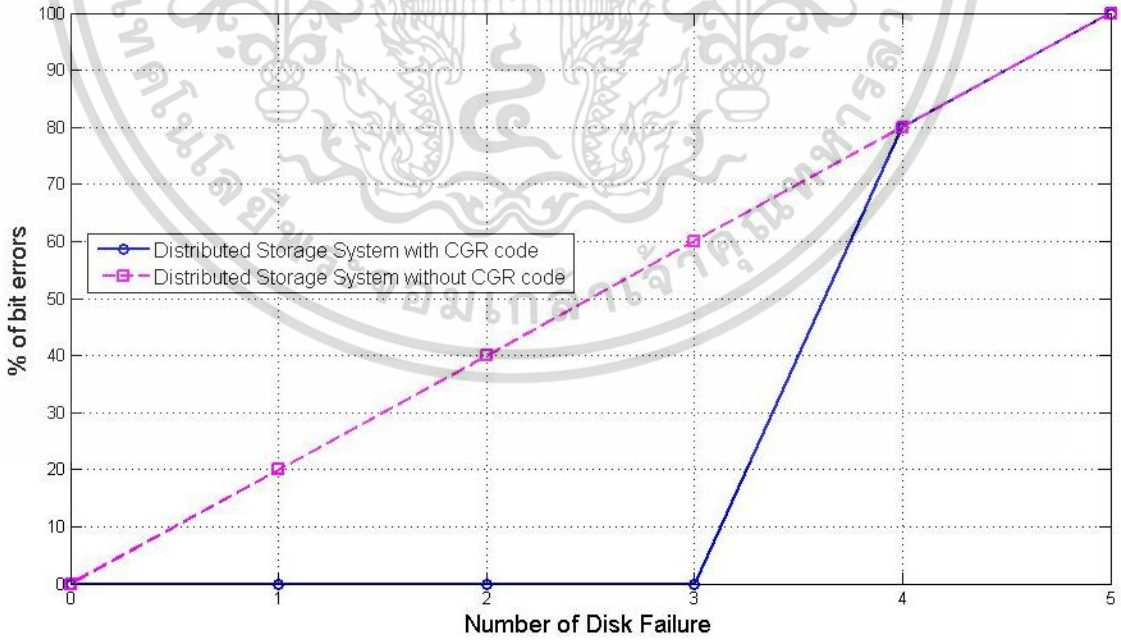
ทางผู้วิจัยได้นำข้อความจากบทความภาษาอังกฤษจากเครือข่ายอินเตอร์เน็ตทั้งหมดประมาณ 5,000 คำ โดยเก็บข้อความ และความถี่ที่พบคำนั้น มาสร้างฐานข้อมูลคำศัพท์สำหรับพิจารณาแก้ไขข้อมูลในอัลกอริทึมที่ทดลอง ซึ่งทางผู้วิจัยได้นำบทความ 5 บทความ แต่ละบทความมีประมาณ 200 คำ ส่งผ่านช่องสัญญาณที่มีสัญญาณรบกวนแบบเกาส์เซียนขาวบวก (AWGN : Additive White Gaussian Noise) มายังผู้รับซึ่งใช้ MLD ในการแก้ไขข้อมูลผิดพลาดเบื้องต้น

บทที่ 4
ผลการทดลอง

บทนี้จะกล่าวถึงผลการศึกษาและการทดลองของงานวิจัยจากบทที่ 3 ซึ่งประกอบด้วยผลการศึกษาระบบเก็บข้อมูลแบบกระจายที่เข้ารหัสด้วยรหัส CGR ผลการทดลองของข้อมูลภาพแบบไบนารี่ ผลการทดลองข้อมูลภาพแบบนอน-ไบนารี่ และผลการทดลองข้อมูลแบบเอกสารกับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

4.1 ผลการศึกษาระบบจัดเก็บข้อมูลแบบกระจาย

จากการศึกษาระบบจัดเก็บข้อมูลแบบกระจายโดยสร้างข้อมูลไบนารี่ขนาด 100,000 บิต 200,000 บิต 300,000 บิต 400,000 บิต และ 500,000 บิตเก็บในหน่วยเก็บข้อมูลแบบกระจายที่มีการใช้งานรหัส CGR (K_2, C_5) เป็นรหัสอีเรเชอร์ในการป้องกันข้อมูลเสียหายภายในระบบเก็บข้อมูลนี้ จากนั้นจำลองให้หน่วยความจำในระบบเก็บข้อมูลแบบกระจายเสียหายจำนวน 0-5 ตัว ซึ่งได้ผลการทดลองดังรูปที่ 4.1 เมื่อพล็อตกราฟแกน y ในรูปแบบเปอร์เซ็นต์ของค่าบิตผิดพลาด และแกน x คือหน่วยความจำที่เสียหาย



รูปที่ 4.1 ค่าเปอร์เซ็นต์ของข้อมูลผิดพลาดเมื่อหน่วยความจำเสียหาย 0-5 ตัว ในระบบเก็บข้อมูลแบบกระจายที่ใช้รหัส CGR (K_2, C_5)

จากผลการทดลองรูปที่ 4.1 แสดงให้เห็นว่าเมื่อเก็บข้อมูลในหน่วยเก็บข้อมูลแบบกระจายที่ใช้งานรหัส CGR (K_2, C_5) กรณีที่หน่วยความจำเสียหาย 1-3 ตัว สามารถกู้ข้อมูลคืนกลับมาได้ทั้งหมด 100% เนื่องจากอยู่ในเงื่อนไขของคุณสมบัติรหัส CGR (K_2, C_5) แต่กรณีที่หน่วยความจำเสียหาย 4 ตัว และ 5 ตัว นั้นจะไม่สามารถกู้ข้อมูลของหน่วยความจำที่เสียหายไปกลับมาได้ทั้งหมด ซึ่งสามารถสรุปได้ดังตารางที่ 4.1

ตารางที่ 4.1 สรุปผลการศึกษาระบบเก็บข้อมูลแบบกระจายเมื่อใช้รหัส CGR (K_2, C_5)

รหัสอีเรเชอร์	จำนวนหน่วยความจำที่เสียหาย (ตัว)	ผลการทดลอง
รหัส CGR (K_2, C_5)	1	สามารถกู้ข้อมูลคืนได้ทั้งหมด
	2	สามารถกู้ข้อมูลคืนได้ทั้งหมด
	3	สามารถกู้ข้อมูลคืนได้ทั้งหมด
	4	ไม่สามารถกู้ข้อมูลหน่วยความจำที่เสียหายไปกลับคืนมาได้
	5	ไม่สามารถกู้ข้อมูลหน่วยความจำที่เสียหายไปกลับคืนมาได้

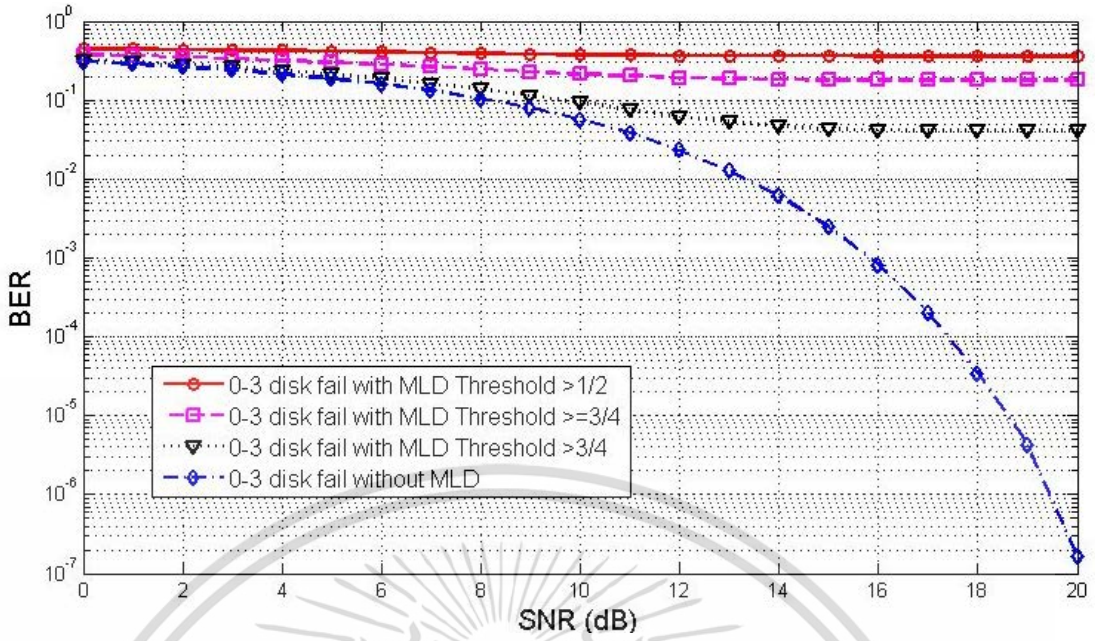
4.2 ผลการทดลองข้อมูลภาพแบบไบนารี่กับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

4.2.1 ผลการทดลองข้อมูลภาพแบบไบนารี่กับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

จากรูปภาพสามชุดต่างๆ กัน ขนาด 600x600 จุดภาพ ซึ่งประกอบไปด้วย

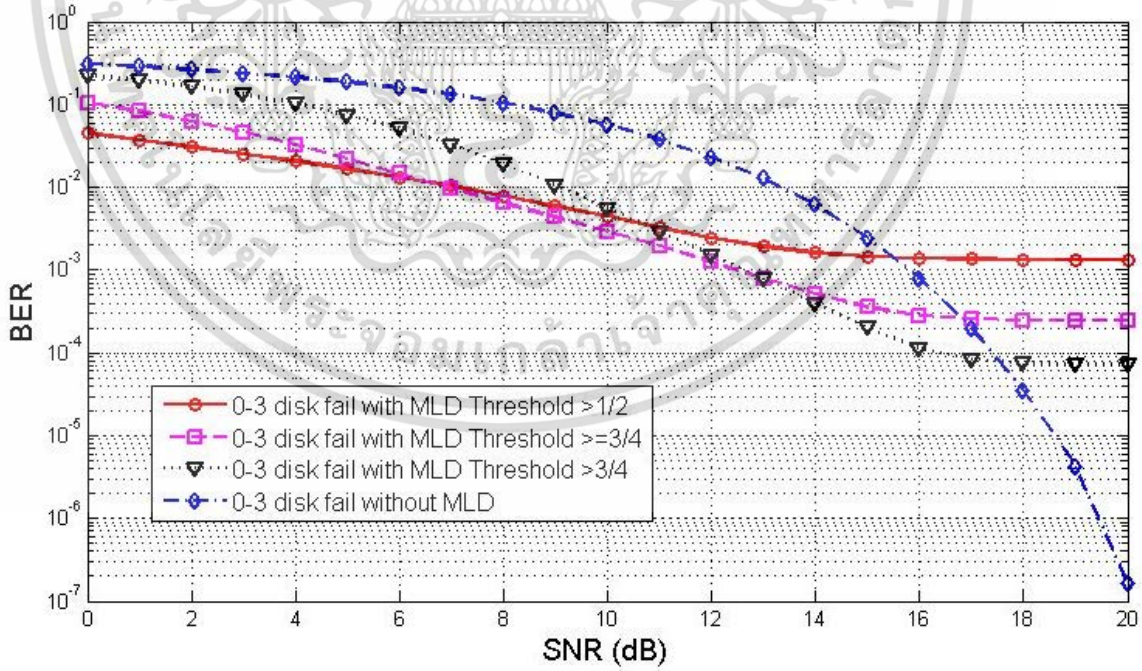
1. รูปภาพจากข้อมูลไบนารี่แบบสุ่มขนาด 600x600 จุดภาพ จำนวน 50 รูป
2. รูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 1 จำนวน 50 รูป
3. รูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 2 จำนวน 50 รูป
4. รูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 1 และ 2 มารวมกันแล้วสุ่มเลือกมาทั้งหมด 50 รูป

ในเงื่อนไขที่หน่วยเก็บข้อมูลเสียหายพร้อมกัน 0-3 ตัว จากทั้งหมด 5 ตัว และที่เครื่องรับใช้อัลกอริทึม MLD ที่มีค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$ และ $T_h > \frac{3}{4}$ ในเงื่อนไขที่จะตรวจสอบและแก้ไขข้อมูลผิดพลาด ซึ่งได้ผลดังรูปที่ 4.2, 4.3, 4.4 และ 4.5 ตามลำดับ



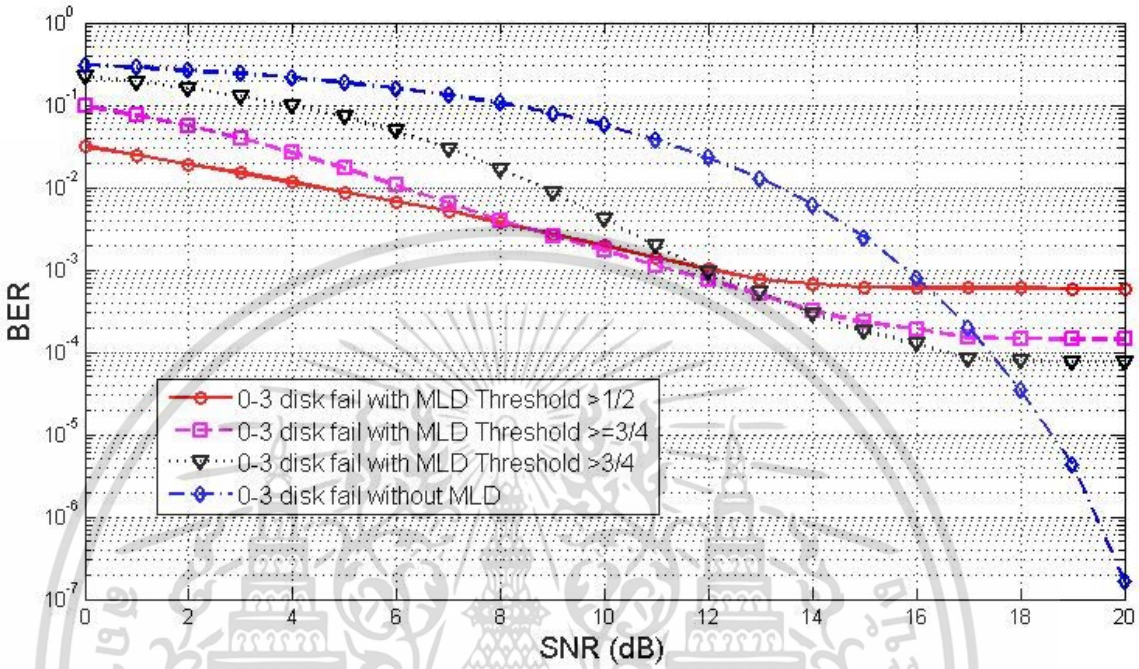
รูปที่ 4.2 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่แบบสุ่มขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$

ผลการทดลองรูปที่ 4.2 แสดงให้เห็นว่า MLD ไม่มีผลในการเพิ่มสมรรถนะแก้ไขความผิดพลาดกับข้อมูลไบนารี่แบบสุ่ม เนื่องจากข้อมูลแต่ละบิตไม่มีความสัมพันธ์กัน



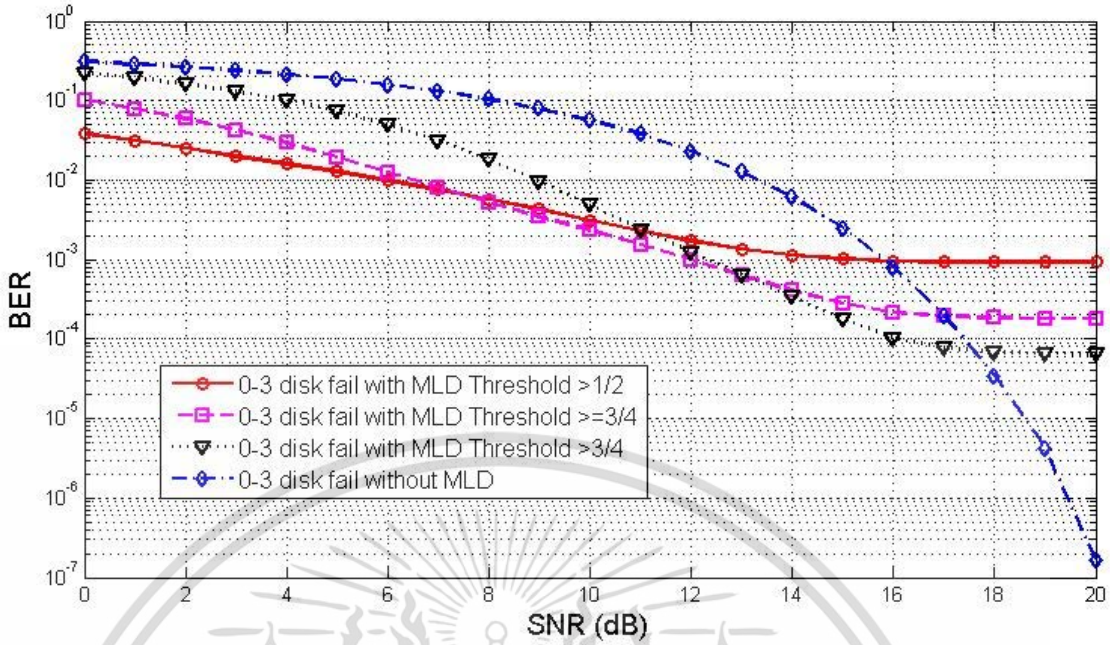
รูปที่ 4.3 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 1 เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$

ผลการทดลองรูปที่ 4.3 แสดงให้เห็นว่า MLD ให้ผลลัพธ์ในการเพิ่มสมรรถนะของการแก้ไขความผิดพลาดที่ดีกว่ากับข้อมูลภาพแบบไบนารี่แบบที่ 1 เมื่อ $T_h > \frac{1}{2}$ ที่ SNR < 16 dB , $T_h \geq \frac{3}{4}$ ที่ SNR < 17 dB และ $T_h > \frac{3}{4}$ ที่ SNR < 18 dB



รูปที่ 4.4 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 2 เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$

ผลการทดลองรูปที่ 4.4 แสดงให้เห็นว่า MLD ให้ผลลัพธ์ในการเพิ่มสมรรถนะของการแก้ไขความผิดพลาดที่ดีกว่ากับข้อมูลแบบไบนารี่แบบที่ 2 เมื่อ $T_h > \frac{1}{2}$ ที่ SNR < 16 dB , $T_h \geq \frac{3}{4}$ ที่ SNR < 17 dB และ $T_h > \frac{3}{4}$ ที่ SNR < 18 dB ซึ่งมีผลลัพธ์ที่ใกล้เคียงกับผลการทดลองจากรูปที่ 4.3



รูปที่ 4.5 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ขนาด 600x600 จุดภาพ แบบที่ 1 และ 2 เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$

ผลการทดลองรูปที่ 4.5 ใช้ข้อมูลภาพแบบไบนารี่ แบบที่ 1 และ 2 โดยจะสุ่มเลือกมาจำนวน 50 ภาพ มาทดลองกับ MLD แสดงให้เห็นว่า MLD ให้ผลลัพธ์ในการเพิ่มสมรรถนะของการแก้ไขความผิดพลาดที่ดีกว่า เมื่อ $T_h > \frac{1}{2}$ ที่ SNR < 16 dB , $T_h \geq \frac{3}{4}$ ที่ SNR < 17 dB และ $T_h > \frac{3}{4}$ ที่ SNR < 18 dB ซึ่งมีผลลัพธ์ที่ใกล้เคียงกับผลการทดลองจากรูปที่ 4.3 และ 4.4

จากผลการทดลองทั้งหมด แสดงให้เห็นว่าอัลกอริทึม MLD นั้นมีประสิทธิภาพที่ดีกับข้อมูลรูปภาพ เนื่องจากข้อมูลมีความสัมพันธ์กันระหว่างบิตรอบข้าง โดยค่า T_h ของอัลกอริทึม MLD นั้นมีความสามารถแก้ไขข้อมูลผิดพลาดที่ดีในช่องสัญญาณที่มี SNR แตกต่างกันซึ่ง

1. $T_h > \frac{1}{2}$ มีความสามารถแก้ไขข้อมูลที่ดีเมื่อ SNR อยู่ระหว่าง 0 – 8 dB
2. $T_h \geq \frac{3}{4}$ มีความสามารถแก้ไขข้อมูลที่ดีเมื่อ SNR อยู่ระหว่าง 8 – 12 dB
3. $T_h > \frac{3}{4}$ มีความสามารถแก้ไขข้อมูลที่ดีเมื่อ SNR มากกว่า 12 dB

จากที่กล่าวมาข้างต้นสรุปได้ดังตารางที่ 4.2 และสามารถพิจารณาต่อได้ว่าเมื่อเลือกใช้ T_h ของอัลกอริทึม MLD โดยพิจารณาจากคุณภาพของช่องสัญญาณเพื่อให้มีประสิทธิภาพสูงที่สุด ในข้อมูลภาพที่ไม่ได้ใช้งานอัลกอริทึม MLD นั้นต้องใช้ SNR ถึง 16dB แต่ในข้อมูลรูปภาพที่ใช้งานอัลกอริทึม MLD ที่ค่า $T_h \geq \frac{3}{4}$ หรือ $T_h > \frac{3}{4}$ นั้นใช้ SNR ที่มีค่าเพียงแค่ 12 dB ที่อัตราบิตผิดพลาดมีค่า 10^{-3} หรือในการรับส่งข้อมูลจำนวน 1,000 บิต พบว่ามีบิตผิดพลาด 1 บิต ซึ่งหมายความว่าในการ

ส่งข้อมูลรูปภาพนี้สามารถลด SNR ลงได้ถึง 4 dB เมื่อใช้อัลกอริทึม MLD ที่มีค่า $T_h \geq \frac{3}{4}$ หรือ $T_h > \frac{3}{4}$ เพื่อให้ได้รับคุณภาพของข้อมูลคงเดิม

ตารางที่ 4.2 สรุปผลการทดลองข้อมูลภาพไบนารี่กับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

ประเภทของข้อมูล	ค่าขีดเริ่มเปลี่ยนของ MLD (T_h)	คำอธิบาย
รูปภาพจากข้อมูลไบนารี่แบบสุ่ม	ทุกค่า T_h	อัลกอริทึม MLD ไม่เหมาะสมกับการแก้ไขข้อมูลรูปภาพจากข้อมูลไบนารี่แบบสุ่ม
รูปภาพไบนารี่ขาวดำ	$T_h > \frac{1}{2}$	แก้ไขข้อมูลที่ดีเมื่อ SNR อยู่ระหว่าง 0–8 dB
	$T_h \geq \frac{3}{4}$	แก้ไขข้อมูลที่ดีเมื่อ SNR อยู่ระหว่าง 8–12 dB
	$T_h > \frac{3}{4}$	แก้ไขข้อมูลที่ดีเมื่อ SNR มากกว่า 12 dB

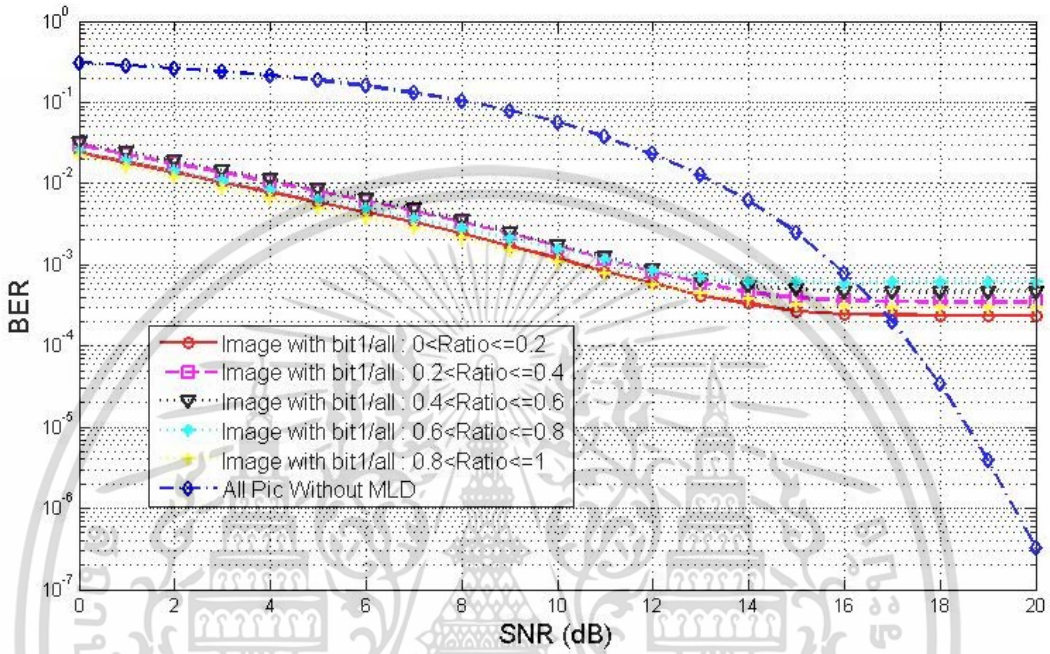
4.2.2 ผลการทดลองการศึกษาอัตราส่วนของข้อมูลไบนารี่ 0 และ 1 กับผลกระทบของอัลกอริทึม

จากการทดลองด้วยรูปภาพขาวดำแบบไบนารี่ขนาด 600x600 จุดภาพ จำนวน 120 ภาพ ที่มีอัตราส่วนของบิต 1 ต่อบิตทั้งหมด โดยบิต 0 แทนบิตสีดำและบิต 1 แทนบิตสีขาว ซึ่งถูกแบ่งเป็น 5 กลุ่มดังนี้

1. $0 < Ratio_{\frac{1}{all}} \leq 0.2$
2. $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$
3. $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$
4. $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$
5. $0.8 < Ratio_{\frac{1}{all}} \leq 1$

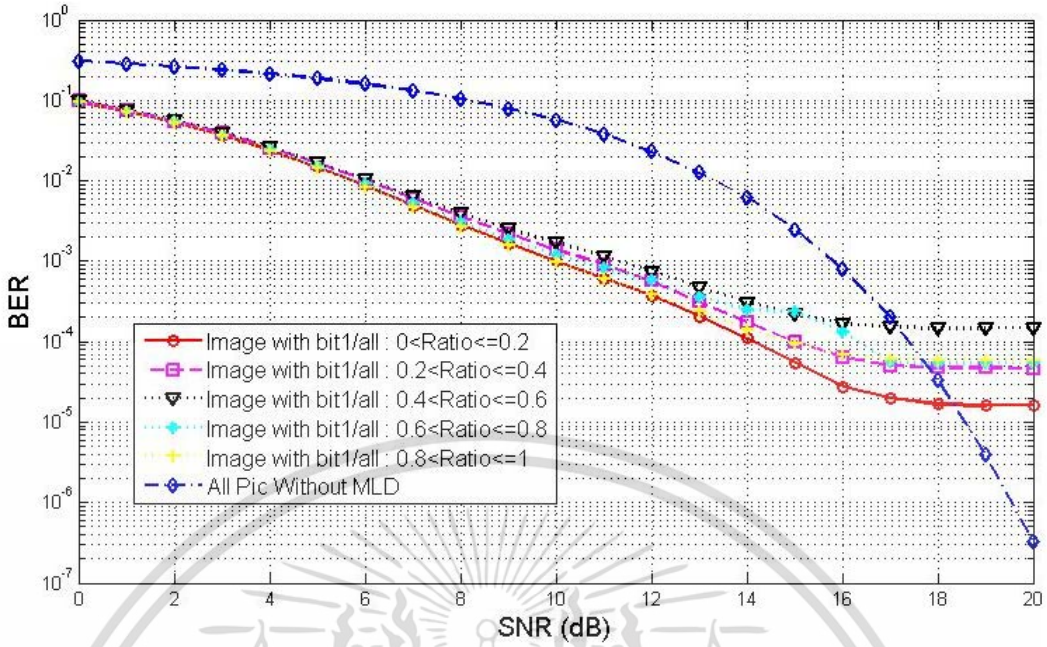
ผ่านช่องสัญญาณที่มีสัญญาณรบกวนแบบเกาส์เซียนขาวบวกที่มี SNR ระหว่าง 0-20 dB โดยในส่วนของผู้รับได้มีการใช้อัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดด้วยค่าขีดเริ่มเปลี่ยน (T_h) ของ MLD ที่ค่า $T_h > 1/2$, $T_h \geq 3/4$ และ $T_h > 3/4$ เพื่อแก้ไขข้อมูลผิดพลาดที่มาจากช่องสัญญาณ โดยในส่วนนี้ผู้วิจัยจะพล็อตกราฟเพื่อวิเคราะห์ข้อมูลเป็น 2 ส่วนคือ ในส่วนแรกจะพล็อตกราฟ T_h ของอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุดแต่ละค่า โดยใช้ข้อมูลรูปภาพที่มี

อัตราส่วนของบิตในค่าที่ต่างกันว่าให้คุณภาพของข้อมูลแตกต่างกันหรือไม่ ดังรูปที่ 4.6, 4.7 และ 4.8 และในส่วนที่สองจะพล็อตกราฟรูปภาพที่มีอัตราส่วนของบิตแต่ละค่า โดยใช้ข้อมูล T_h ของอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด เพื่อพิจารณาคุณภาพของข้อมูลที่ค่า T_h ที่ต่างกัน แสดงดังรูปที่ 4.9, 4.10, 4.11, 4.12 และ 4.13



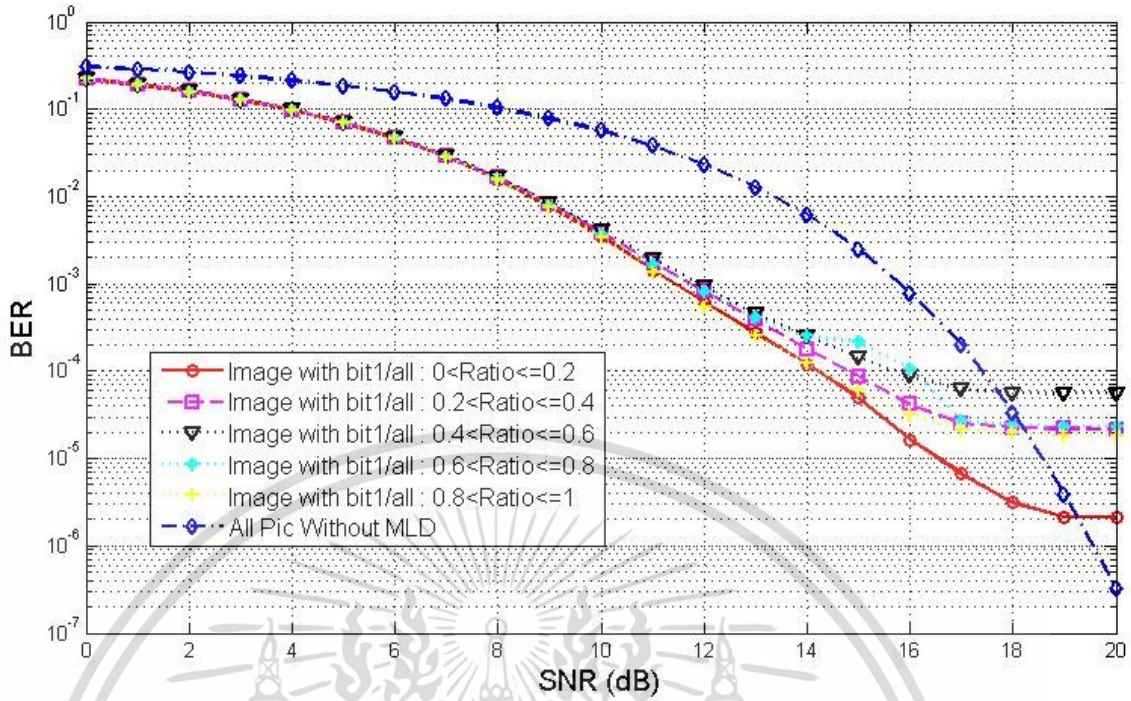
รูปที่ 4.6 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ $T_h > \frac{1}{2}$ กับรูปภาพที่มีอัตราส่วนของบิตในค่าต่างๆ

ผลการทดลองรูปที่ 4.6 แสดงให้เห็นว่าข้อมูลรูปภาพแบบไบนารี่ที่ค่า $T_h > \frac{1}{2}$ กับรูปภาพที่มีอัตราส่วน $0 < Ratio_{\frac{1}{all}} \leq 0.2$, $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$, $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$, $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ และ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ให้ค่า BER ที่ใกล้เคียงกันที่ทุกค่า SNR



รูปที่ 4.7 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ $T_h \geq \frac{3}{4}$ กับรูปภาพที่มีอัตราส่วนของบิตในค่าต่างๆ

ผลการทดลองรูปที่ 4.7 แสดงให้เห็นว่าข้อมูลรูปภาพแบบไบนารี่ที่ค่า $T_h \geq \frac{3}{4}$ กับรูปภาพที่มีอัตราส่วน $0 < Ratio_{\frac{1}{all}} \leq 0.2$, $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$, $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$, $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ และ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ให้ค่า BER ที่ใกล้เคียงกันที่ SNR ค่าต่ำๆ ($SNR < 10dB$) และที่ $0 < Ratio_{\frac{1}{all}} \leq 0.2$ และ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ซึ่งเป็นรูปภาพที่มีอัตราส่วนบิต 1 ต่อบิต 0 และรูปภาพที่มีอัตราส่วนบิต 0 ต่อบิต 1 ที่มีค่าใกล้เคียงกันจะให้ค่าผลลัพธ์ของการทดลองที่ดีกว่า (BER ต่ำ) นอกจากนั้นยังพบว่าที่ SNR มีค่าสูง ($SNR > 12dB$) รูปภาพที่มีอัตราส่วนของบิต 1 ต่อบิต 0 ใกล้เคียงกัน คือ $Ratio_{\frac{1}{all}}$ 0.4-0.6 ให้ค่า BER ที่สูงที่สุด (สมรรถนะในการแก้ไขบิตผิดพลาดต่ำที่สุด) เนื่องจากบิตตัวเดียวมีความสัมพันธ์กันสูงกว่าอัตราส่วนอื่นๆ



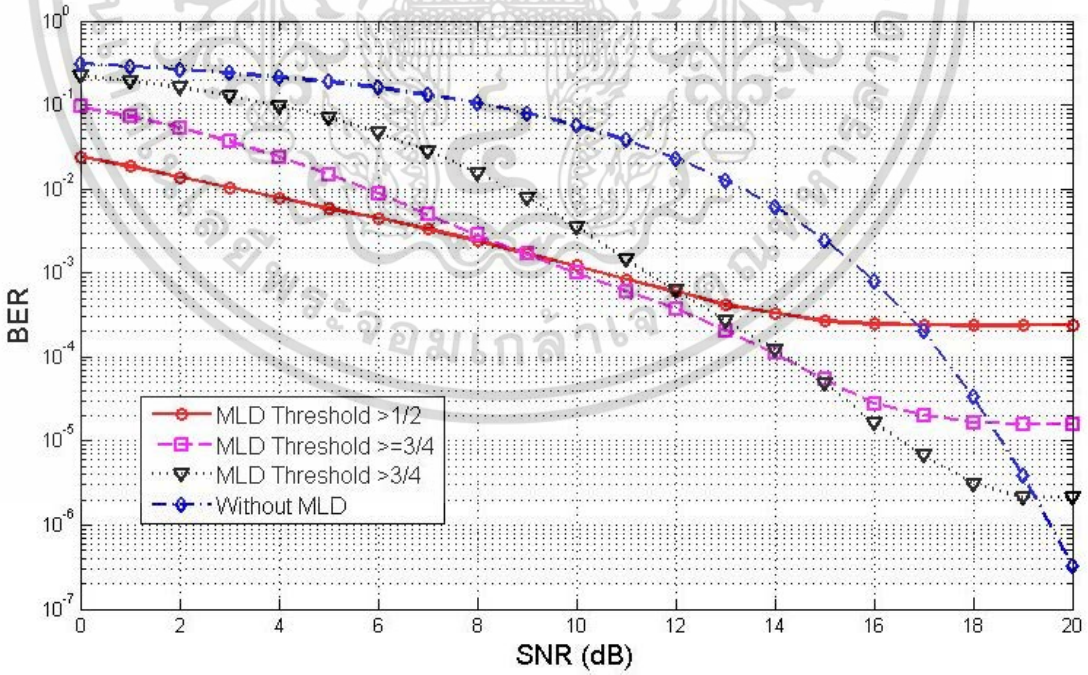
รูปที่ 4.8 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ $T_h > \frac{3}{4}$ กับรูปภาพที่มีอัตราส่วนของบิตในค่าต่างๆ

ผลการทดลองรูปที่ 4.8 แสดงให้เห็นว่าข้อมูลรูปภาพแบบไบนารี่ที่ค่า $T_h \geq \frac{3}{4}$ กับรูปภาพที่มีอัตราส่วน $0 < Ratio_{\frac{1}{all}} \leq 0.2$, $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$, $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$, $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ และ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ให้ค่า BER ที่ใกล้เคียงกันที่ SNR ค่าต่ำๆ ($SNR < 10dB$) และที่ $0 < Ratio_{\frac{1}{all}} \leq 0.2$ และ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ซึ่งเป็นรูปภาพที่มีอัตราส่วนบิต 1 ต่อบิต 0 และรูปภาพที่มีอัตราส่วนบิต 0 ต่อบิต 1 ที่มีค่าใกล้เคียงกันจะให้ค่าผลลัพธ์ของการทดลองที่ดีกว่า (BER ต่ำ) ซึ่งมีผลลัพธ์ที่ใกล้เคียงกับผลการทดลองจากรูปที่ 4.7

จากผลการทดลองดังรูปที่ 4.6, 4.7 และ 4.8 สามารถวิเคราะห์ค่าความสัมพันธ์ระหว่าง T_h กับชนิดของรูปภาพที่มีอัตราส่วนของบิตที่ต่างกันได้ดังนี้

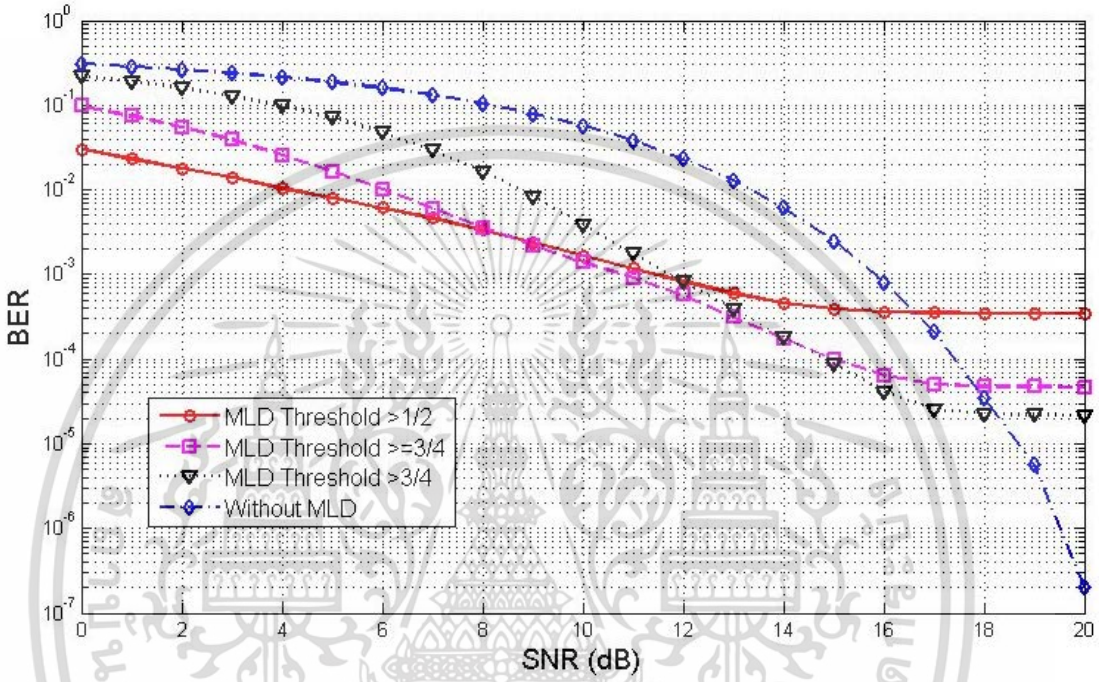
1. เมื่อกำหนดค่า $T_h > \frac{1}{2}$ ชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆนั้นไม่มีผลต่อประสิทธิภาพของอัลกอริทึม

2. เมื่อกำหนดค่า $T_h \geq \frac{3}{4}$ ชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆ นั้นไม่มีผลต่อประสิทธิภาพของอัลกอริทึมเมื่อ SNR อยู่ระหว่าง 0-6 dB แต่ที่ช่วง SNR มากกว่า 6 dB ชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆ นั้นมีผลต่อประสิทธิภาพของอัลกอริทึมเพียงเล็กน้อย จนกระทั่งถึง SNR สูงกว่า 14 dB ซึ่งเห็นว่าชนิดของรูปภาพที่มีอัตราส่วนของบิต $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$ หรือข้อมูลรูปภาพที่อัตราส่วนของบิตขาวกับบิตดำมีค่าใกล้เคียงกันนั้นให้ BER มีค่าสูงที่สุด (สมรรถนะในการแก้ไขบิตผิดพลาดต่ำ) เมื่อเทียบกับชนิดของรูปภาพที่มีอัตราส่วนของบิตอื่น
3. เมื่อกำหนดค่า $T_h > \frac{3}{4}$ ชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆ นั้นไม่มีผลต่อประสิทธิภาพของอัลกอริทึมเมื่อ SNR อยู่ระหว่าง 0-10 dB แต่ที่ช่วง SNR มากกว่า 10 dB ชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆ นั้นมีผลต่อประสิทธิภาพของอัลกอริทึมเพียงเล็กน้อย จนกระทั่งถึง SNR สูงกว่า 14 dB ซึ่งเห็นว่าชนิดของรูปภาพที่มีอัตราส่วนของบิต $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$ หรือข้อมูลรูปภาพที่อัตราส่วนของบิตขาวกับบิตดำมีค่าใกล้เคียงกันนั้นให้ BER มีค่าสูงที่สุด (สมรรถนะในการแก้ไขบิตผิดพลาด) เมื่อเทียบกับชนิดของรูปภาพที่มีอัตราส่วนของบิตอื่นๆ เนื่องจากบิตมีความสัมพันธ์กันสูงกับบิตตัวเดียว



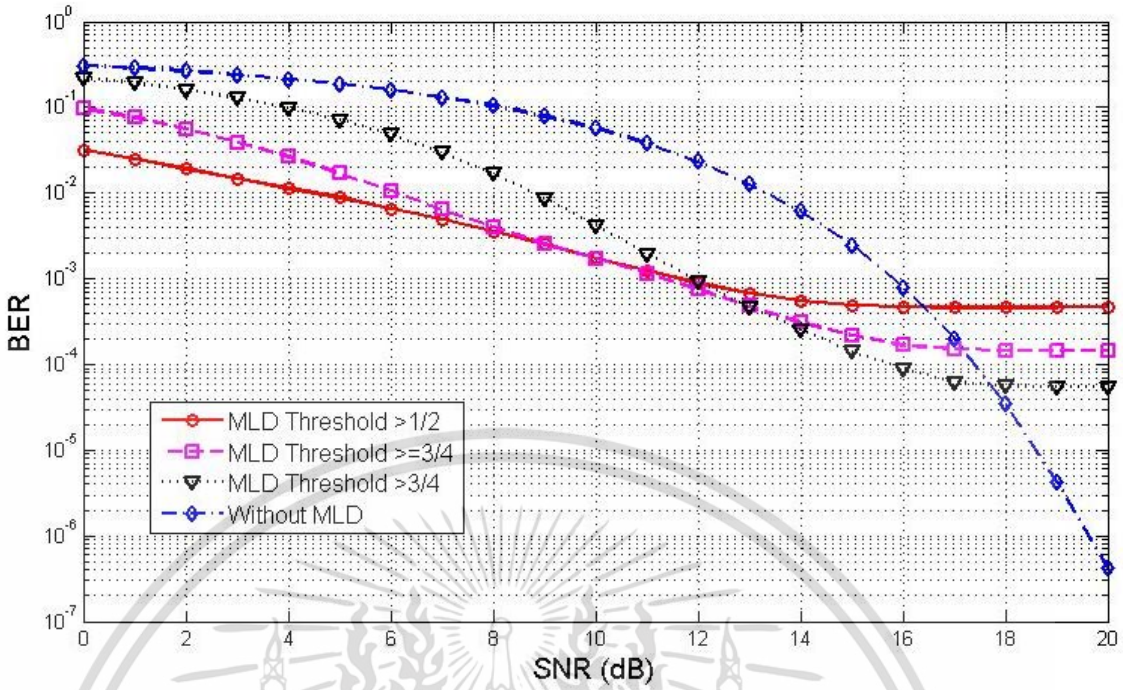
รูปที่ 4.9 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ที่มีอัตราส่วนบิตคือ $0 < Ratio_{\frac{1}{all}} \leq 0.2$ กับ T_h ค่าต่างๆ

ผลการทดลองรูปที่ 4.9 แสดงผลการทดลองที่ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ เมื่ออัตราส่วนของบิตในภาพมีค่า $0 < Ratio_{\frac{1}{all}} \leq 0.2$ ให้ผลลัพธ์คือที่ค่า $T_h > \frac{1}{2}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 17 dB, $T_h \geq \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18 dB, $T_h > \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 19 dB



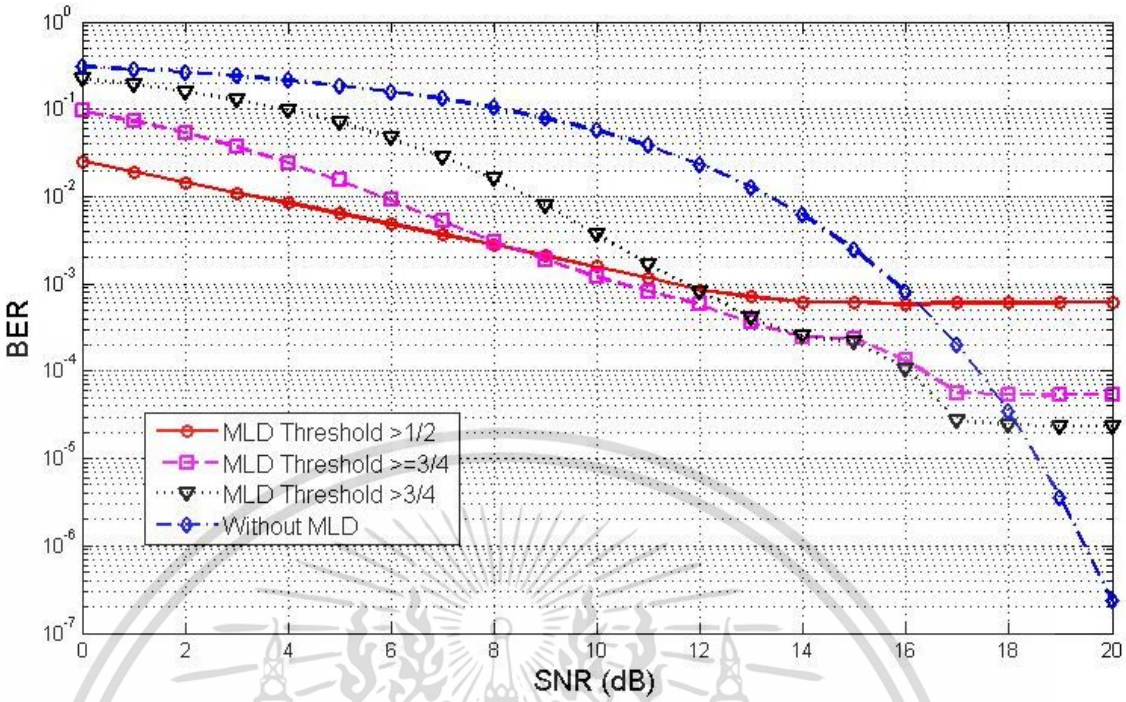
รูปที่ 4.10 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ที่มีอัตราส่วนบิตคือ $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$ กับ T_h ค่าต่างๆ

ผลการทดลองรูปที่ 4.10 แสดงผลการทดลองที่ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ เมื่ออัตราส่วนของบิตในภาพมีค่า $0.2 < Ratio_{\frac{1}{all}} \leq 0.4$ ให้ผลลัพธ์คือที่ค่า $T_h > \frac{1}{2}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 16.5 dB, $T_h \geq \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18 dB, $T_h > \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18.5 dB



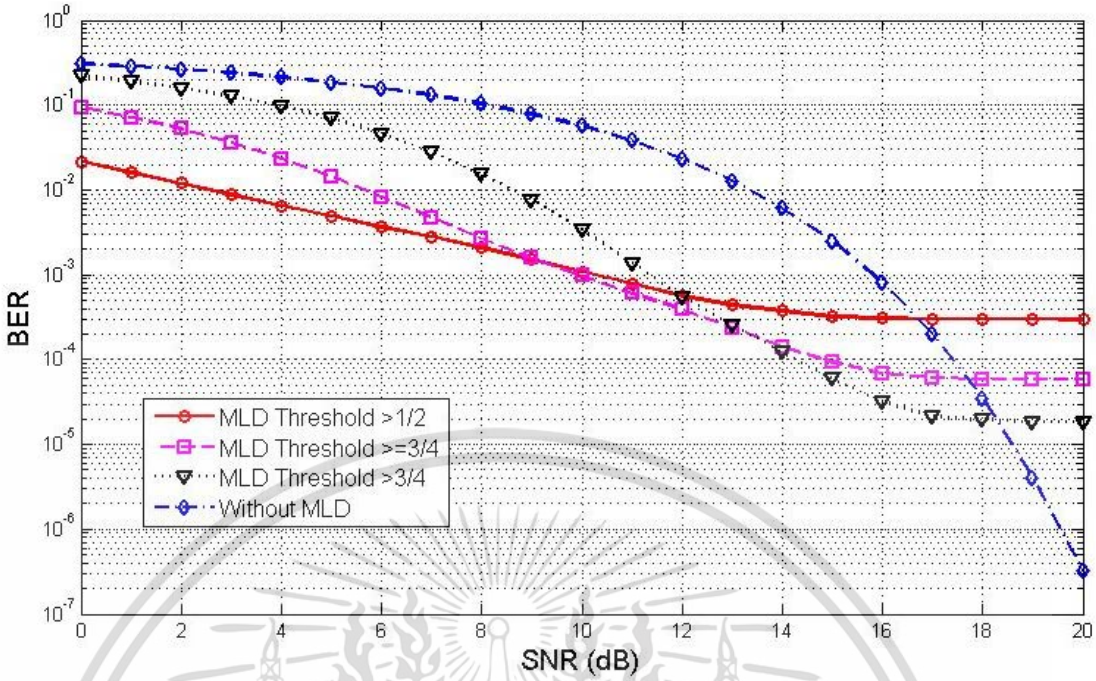
รูปที่ 4.11 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ที่มีอัตราส่วนบิตคือ $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$ กับ T_h ค่าต่างๆ

ผลการทดลองรูปที่ 4.11 แสดงผลการทดลองที่ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ เมื่ออัตราส่วนของบิตในภาพมีค่า $0.4 < Ratio_{\frac{1}{all}} \leq 0.6$ ให้ผลลัพธ์คือที่ค่า $T_h > \frac{1}{2}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 16.5 dB, $T_h \geq \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 17 dB, $T_h > \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 17.5 dB



รูปที่ 4.12 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ที่มีอัตราส่วนบิตคือ $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ กับ T_h ค่าต่างๆ

ผลการทดลองรูปที่ 4.12 แสดงผลการทดลองที่ค่า $T_h > \frac{1}{2}, T_h \geq \frac{3}{4}, T_h > \frac{3}{4}$ เมื่ออัตราส่วนของบิตในภาพมีค่า $0.6 < Ratio_{\frac{1}{all}} \leq 0.8$ ให้ผลลัพธ์คือที่ค่า $T_h > \frac{1}{2}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 16 dB, $T_h \geq \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18 dB, $T_h > \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18.5 dB



รูปที่ 4.13 ค่า BER จากการถอดรหัสข้อมูลรูปภาพแบบไบนารี่ที่มีอัตราส่วนบิตคือ $0.8 < Ratio_{\frac{1}{all}} \leq 1$ กับ T_h ค่าต่างๆ

ผลการทดลองรูปที่ 4.13 แสดงผลการทดลองที่ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ เมื่ออัตราส่วนของบิตในภาพมีค่า $0.8 < Ratio_{\frac{1}{all}} \leq 1$ ให้ผลลัพธ์คือที่ค่า $T_h > \frac{1}{2}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 17 dB, $T_h \geq \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18 dB, $T_h > \frac{3}{4}$ ให้ผลการทดลองที่ดีกว่ากรณีที่ไม่ใช้ MLD (BER ต่ำกว่า) เมื่อ SNR < 18.5 dB

จากผลการทดลองดังรูปที่ 4.9, 4.10, 4.11, 4.12 และ 4.13 สามารถวิเคราะห์ข้อมูลชนิดของรูปภาพที่มีอัตราส่วนของบิตต่างๆได้คือ

1. ที่ค่า $T_h > \frac{1}{2}$ นั้นที่ช่องสัญญาณที่มีค่า SNR 0 dB ถึง 20dB ให้ BER ที่ใกล้เคียงกัน กล่าวคือมีค่าตั้งแต่ $3x10^{-2}$ ถึง 10^{-4} ซึ่งค่า T_h ค่านี้เหมาะสมกับข้อมูลรูปภาพที่ต้องผ่านช่องสัญญาณที่มีสัญญาณรบกวนสูง (SNR ต่ำ ; SNR < 8 dB)
2. ที่ค่า $T_h \geq \frac{3}{4}$ นั้นที่ช่องสัญญาณที่มีค่า SNR 0 dB ถึง 20dB ให้ค่า BER $9x10^{-2}$ ถึง 10^{-5} ซึ่งค่า T_h นี้เหมาะสมกับข้อมูลรูปภาพที่ต้องผ่านช่องสัญญาณที่ไม่ทราบความเข้มของสัญญาณรบกวน (ไม่ทราบ SNR) เนื่องจากค่า BER ของ $T_h \geq \frac{3}{4}$ นั้นให้ค่า BER ที่มีค่ากึ่งกลางระหว่างค่า $T_h > \frac{1}{2}$ และ $T_h > \frac{3}{4}$

3. ที่ค่า $T_h > \frac{3}{4}$ นั้นที่ช่องสัญญาณที่มีค่า SNR 0 dB ถึง 20dB ให้ค่า BER แตกต่างกันประมาณ 10^{-1} ถึง 10^{-6} ซึ่งค่า T_h เหมาะสมกับข้อมูลรูปภาพที่ต้องผ่านช่องสัญญาณที่มีความเข้มของสัญญาณรบกวนต่ำ (SNR สูง ; SNR > 12 dB)

ซึ่งจากผลการทดลองที่กล่าวมาข้างต้นสามารถสรุปได้ดังตารางที่ 4.3

ตารางที่ 4.3 สรุปผลการทดลองการศึกษาอัตราส่วนของข้อมูลไบนารี่ 0 และ 1 กับผลกระทบของอัลกอริทึม

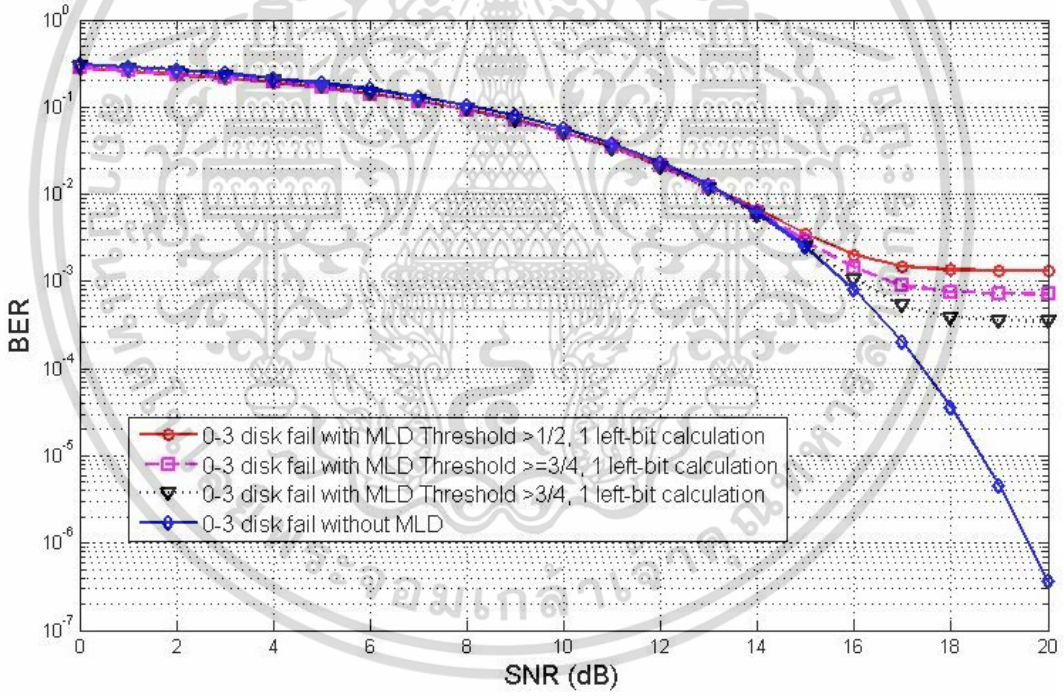
หัวข้อการทดลอง	ค่าพารามิเตอร์	คำอธิบาย
ค่าขีดเริ่มเปลี่ยนของ MLD (T_h)	$T_h > \frac{1}{2}$	เหมาะสมกับช่องสัญญาณที่มี SNR ต่ำ (SNR < 8 dB)
	$T_h \geq \frac{3}{4}$	เหมาะสมกับช่องสัญญาณที่ไม่ทราบ SNR
	$T_h > \frac{3}{4}$	เหมาะสมกับช่องสัญญาณที่มี SNR สูง (SNR > 12 dB)
อัตราส่วนของข้อมูลไบนารี่ 0 และ 1	$0 < Ratio_{\frac{1}{all}} \leq 0.2$	คุณภาพข้อมูลหลังผ่าน MLD ดีที่สุดเมื่อเทียบกับข้อมูลชนิดอื่น (BER ต่ำที่สุด)
	$0.2 < Ratio_{\frac{1}{all}} \leq 0.4$	คุณภาพข้อมูลหลังผ่าน MLD ปานกลางเมื่อเทียบกับข้อมูลชนิดอื่น (BER ปานกลาง)
	$0.4 < Ratio_{\frac{1}{all}} \leq 0.6$	คุณภาพข้อมูลหลังผ่าน MLD แย่ที่สุดเมื่อเทียบกับข้อมูลชนิดอื่น (BER สูงที่สุด)
	$0.6 < Ratio_{\frac{1}{all}} \leq 0.8$	คุณภาพข้อมูลหลังผ่าน MLD ปานกลางเมื่อเทียบกับข้อมูลชนิดอื่น (BER ปานกลาง)
	$0.8 < Ratio_{\frac{1}{all}} \leq 1$	คุณภาพข้อมูลหลังผ่าน MLD ดีที่สุดเมื่อเทียบกับข้อมูลชนิดอื่น (BER ต่ำที่สุด)

4.3 ผลการทดลองข้อมูลภาพแบบนอน-ไบนารี่กับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

จากการทดลองข้อมูลรูปภาพทั้งสองชนิดขนาด 600x600 จุดภาพ ดังต่อไปนี้

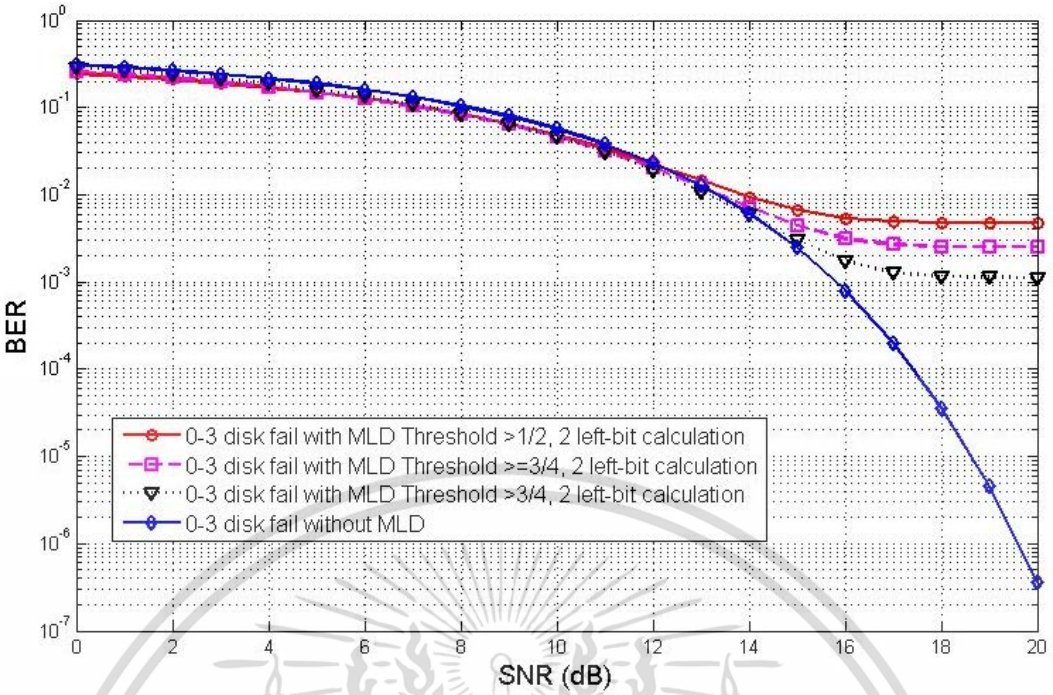
1. รูปภาพแบบนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ จำนวน 30 รูป
2. รูปภาพแบบนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ จำนวน 30 รูป

ในเงื่อนไขที่หน่วยเก็บข้อมูลเสียหายพร้อมกัน 0-3 ตัว จากทั้งหมด 5 ตัว และที่เครื่องรับใช้อัลกอริทึม MLD ที่มีค่าพารามิเตอร์ขีดเริ่มเปลี่ยนคือ $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และจำนวนบิตที่ถูกนำมาพิจารณาคือ 1-8 บิต สำหรับรูปภาพแบบนอน-ไบนารี่ชนิด Grayscale ได้ผลดังรูปที่ 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21 และรูปภาพแบบนอน-ไบนารี่ชนิด RGB ได้ผลดังรูปที่ 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29



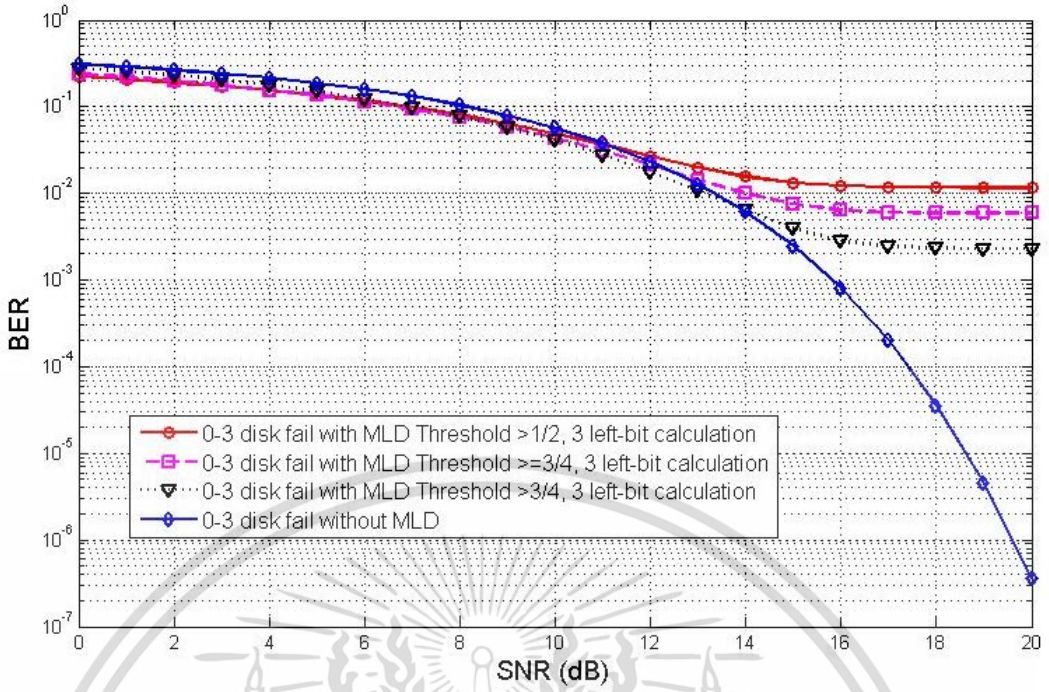
รูปที่ 4.14 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 1 บิต

ผลการทดลองรูปที่ 4.14 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ที่ข้อมูลแต่ละจุดภาพมีความสัมพันธ์กันน้อย ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 1 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 14 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



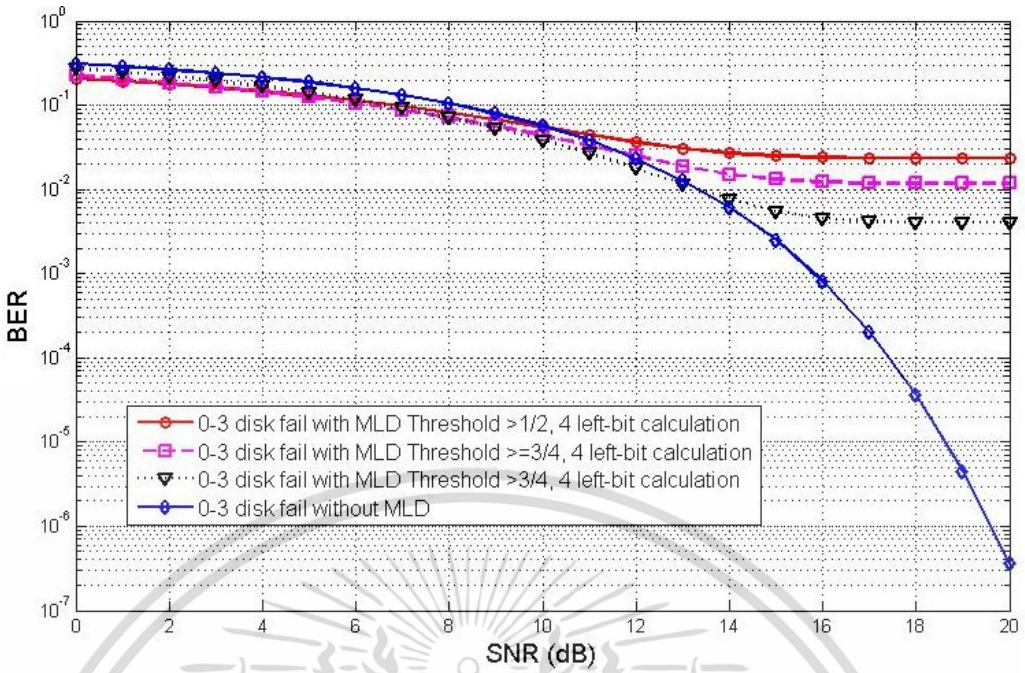
รูปที่ 4.15 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 2 บิต

ผลการทดลองรูปที่ 4.15 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 2 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 13 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



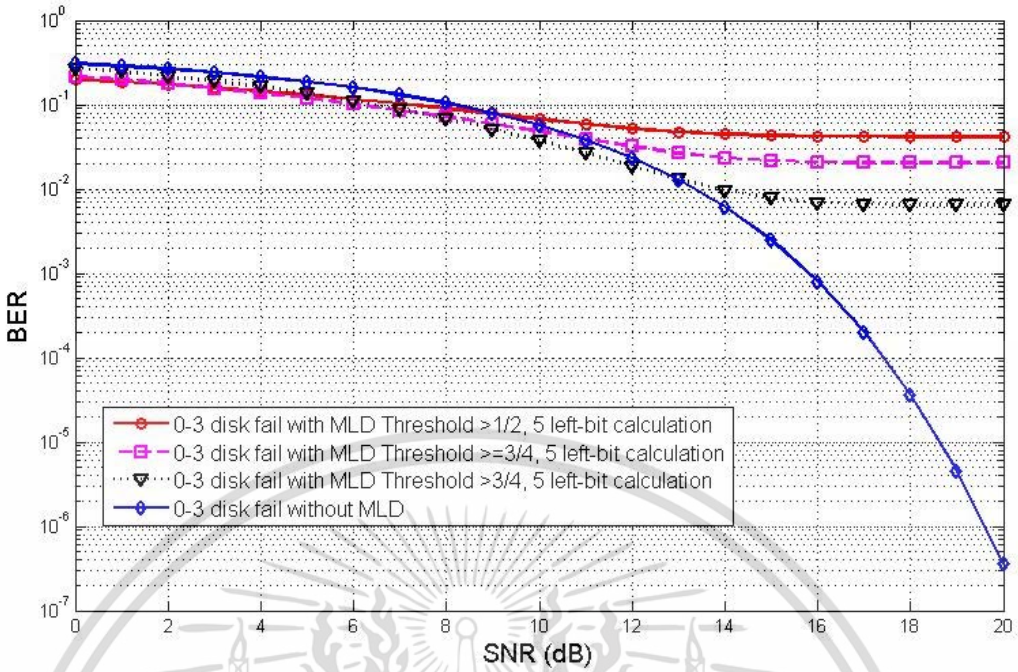
รูปที่ 4.16 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 3 บิต

ผลการทดลองรูปที่ 4.16 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 3 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 12 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



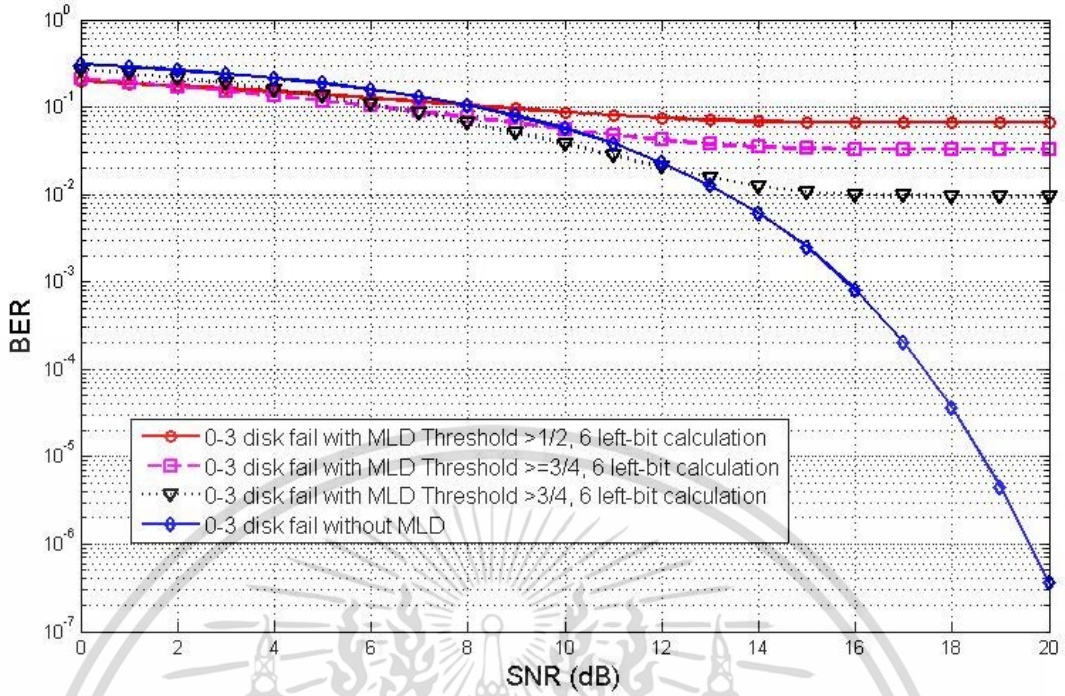
รูปที่ 4.17 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 4 บิต

ผลการทดลองรูปที่ 4.17 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 4 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 11 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



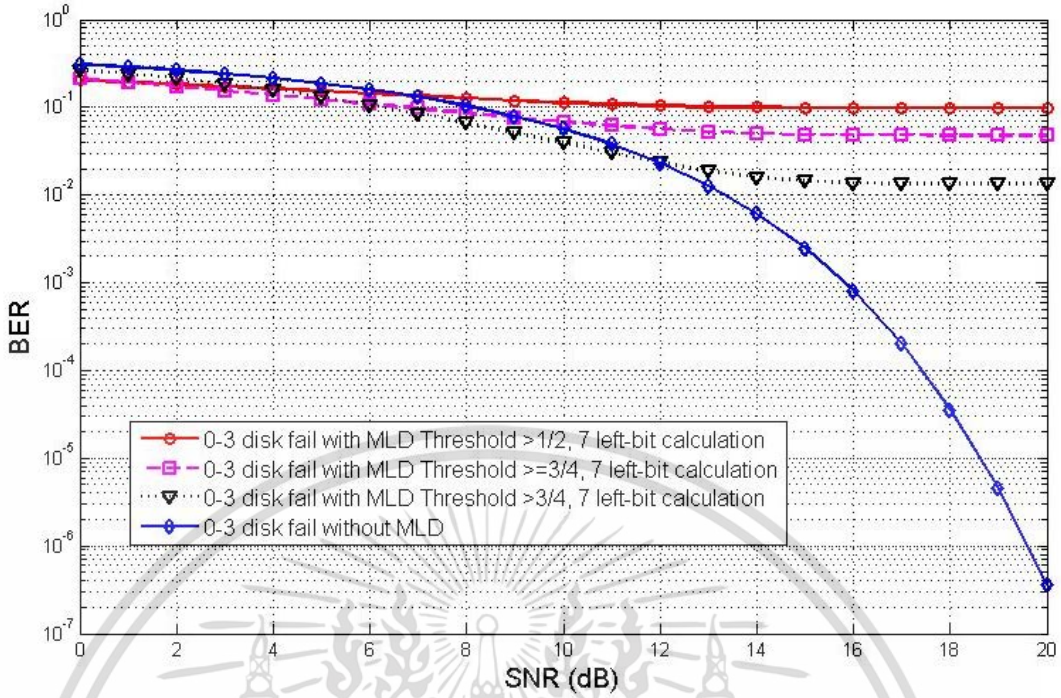
รูปที่ 4.18 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 5 บิต

ผลการทดลองรูปที่ 4.18 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 5 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 9 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



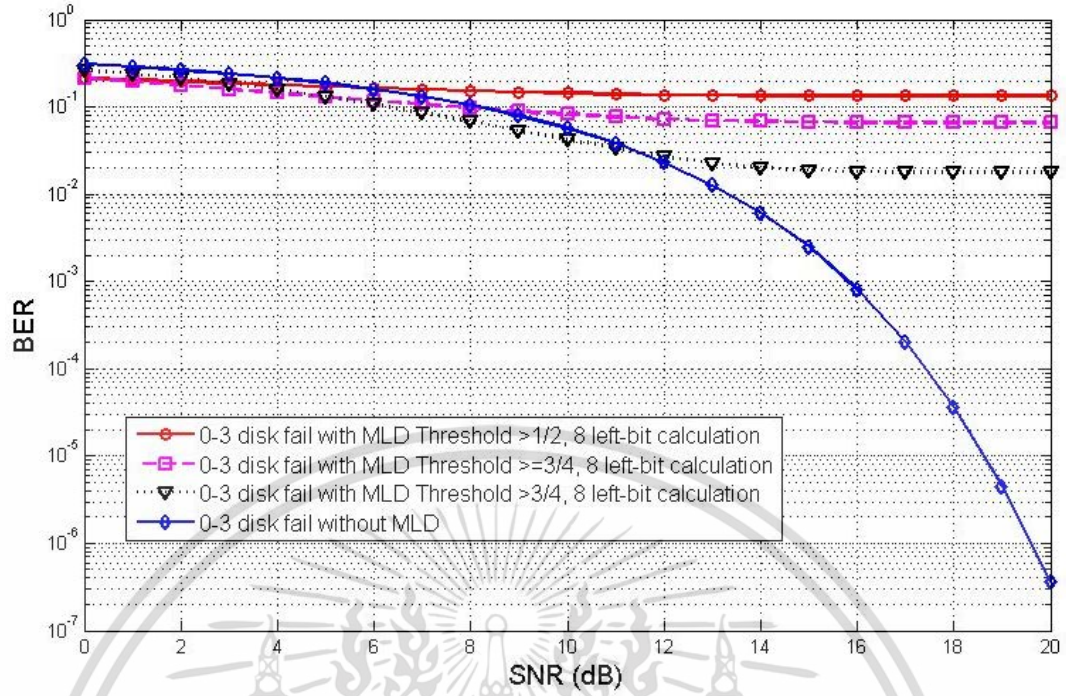
รูปที่ 4.19 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 6 บิต

ผลการทดลองรูปที่ 4.19 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 6 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 9 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



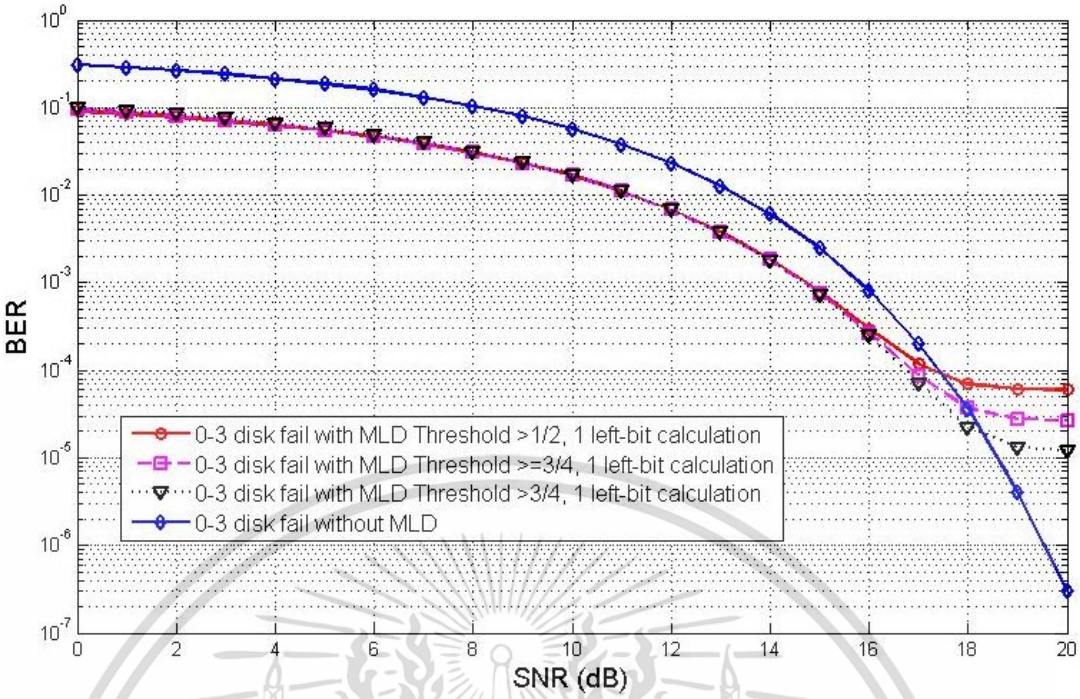
รูปที่ 4.20 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 7 บิต

ผลการทดลองรูปที่ 4.20 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 7 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 8 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



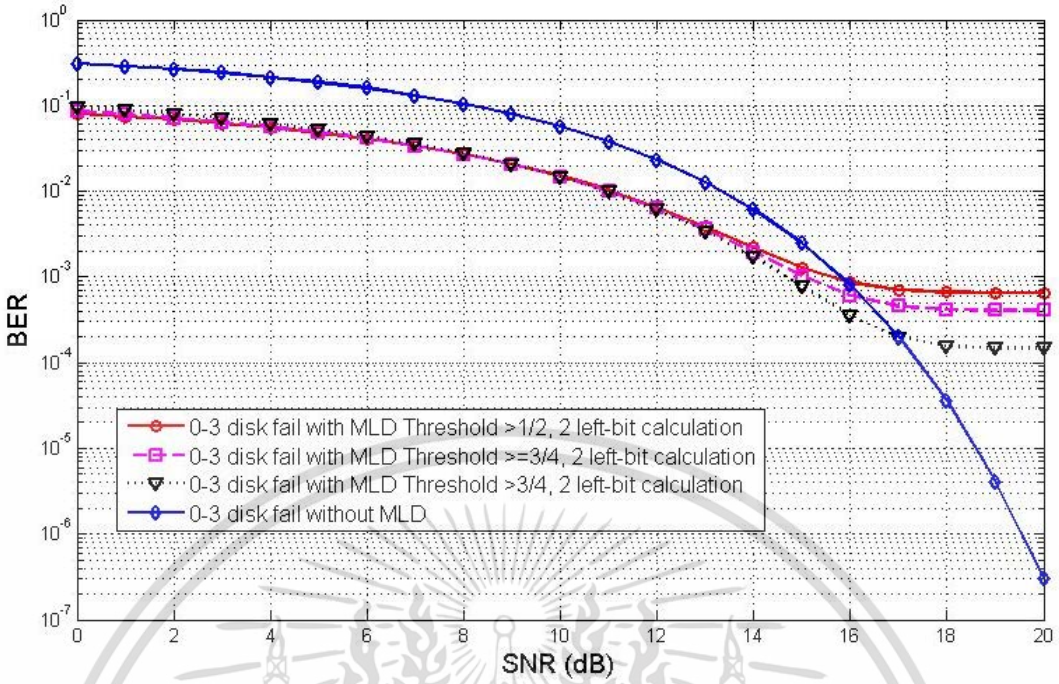
รูปที่ 4.21 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด Grayscale ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 8 บิต

ผลการทดลองรูปที่ 4.21 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ Grayscale ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 8 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER จะดีกว่า (BER ต่ำกว่า) เมื่อ SNR > 7 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



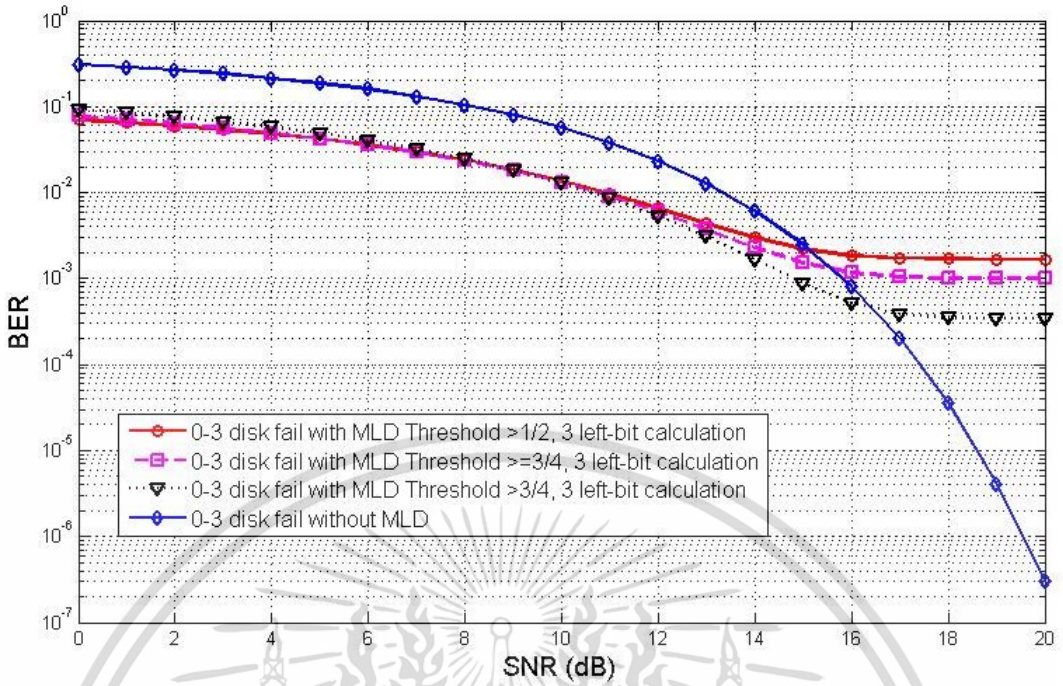
รูปที่ 4.22 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 1 บิต

ผลการทดลองรูปที่ 4.22 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 1 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 16 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



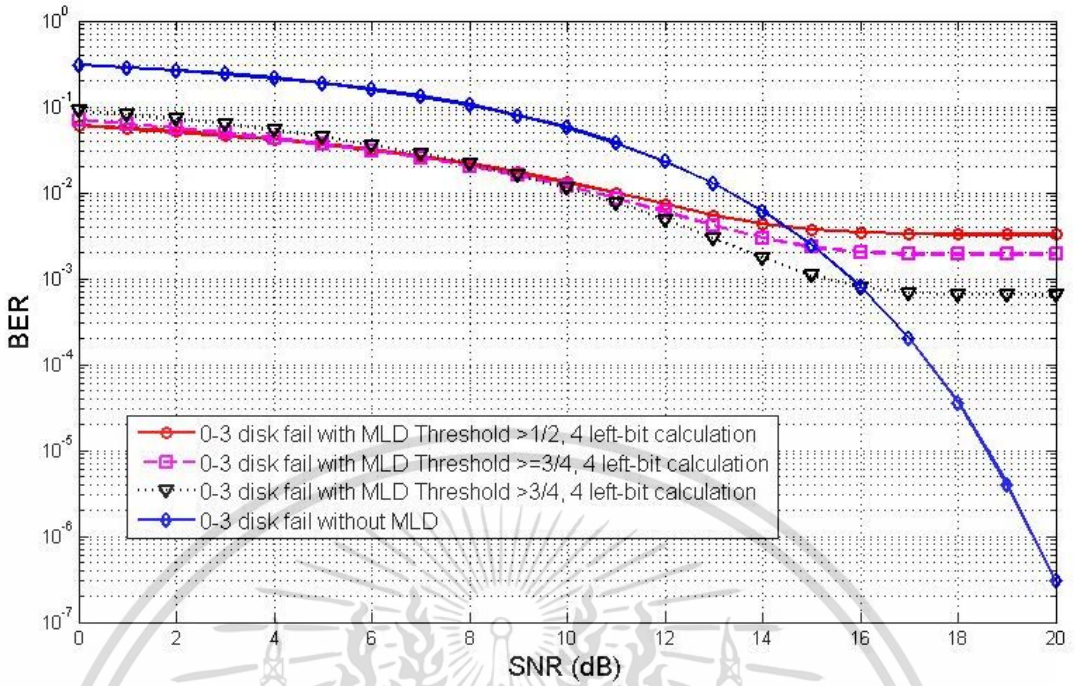
รูปที่ 4.23 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 2 บิต

ผลการทดลองรูปที่ 4.23 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 2 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 14 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



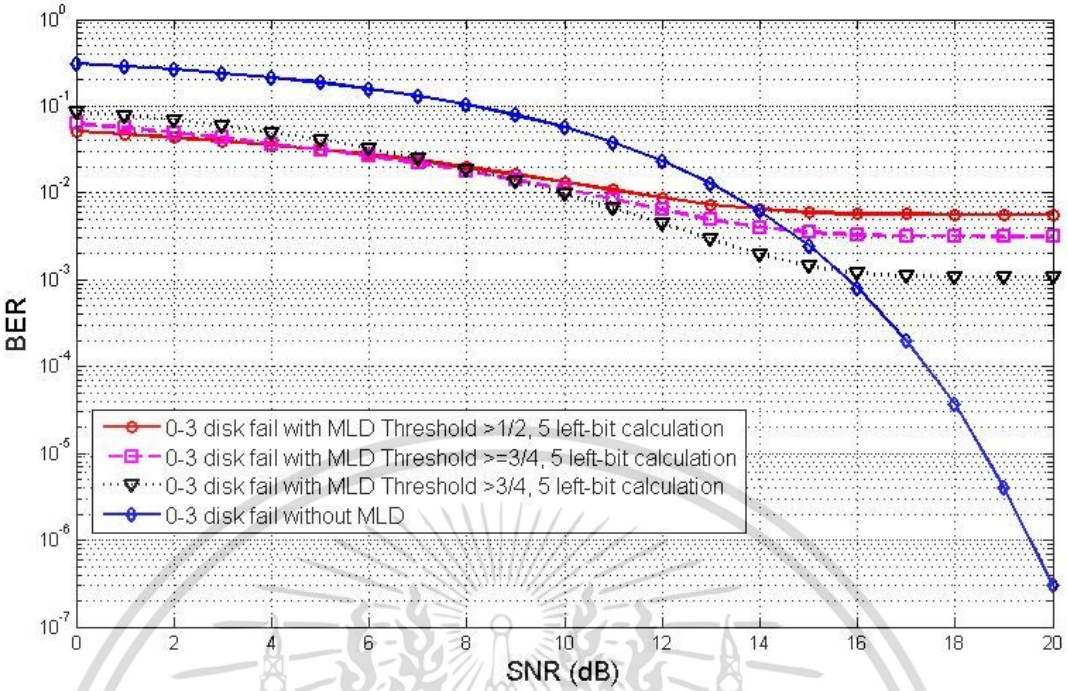
รูปที่ 4.24 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 3 บิต

ผลการทดลองรูปที่ 4.24 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 3 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 12 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



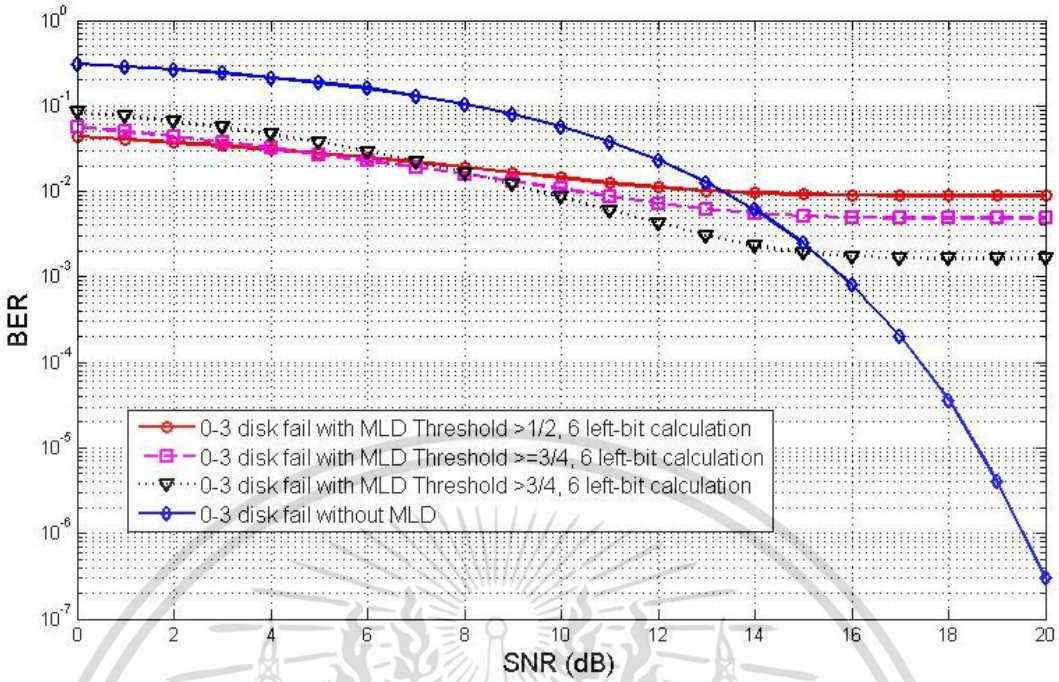
รูปที่ 4.25 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 4 บิต

ผลการทดลองรูปที่ 4.25 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 4 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 10 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลาง และ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



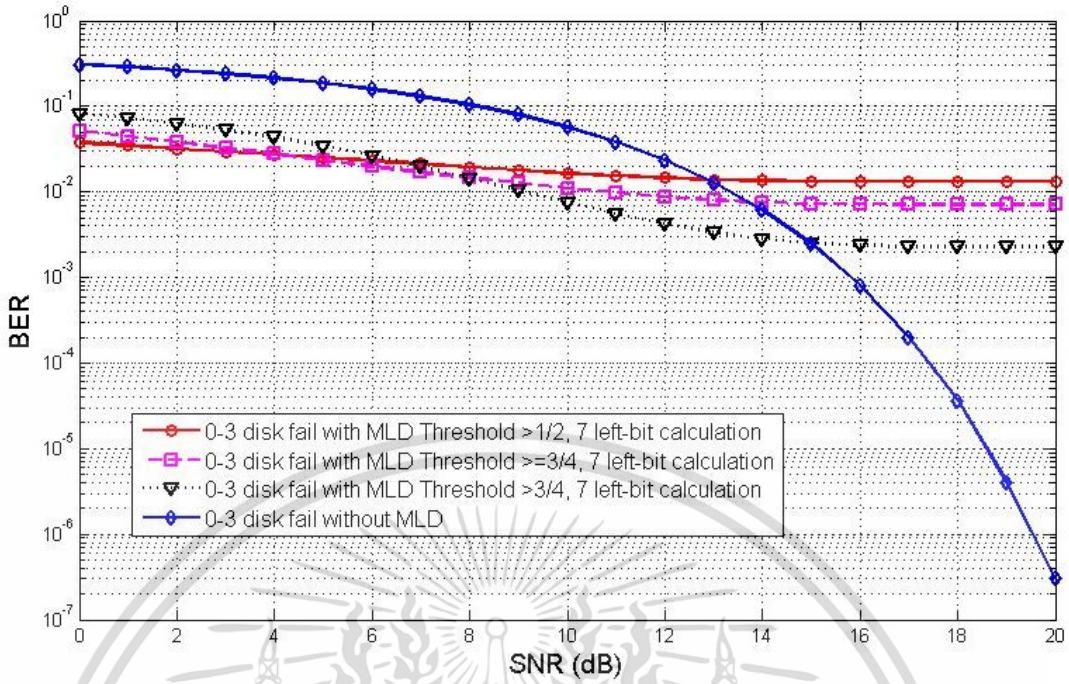
รูปที่ 4.26 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 5 บิต

ผลการทดลองรูปที่ 4.26 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 5 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 8 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



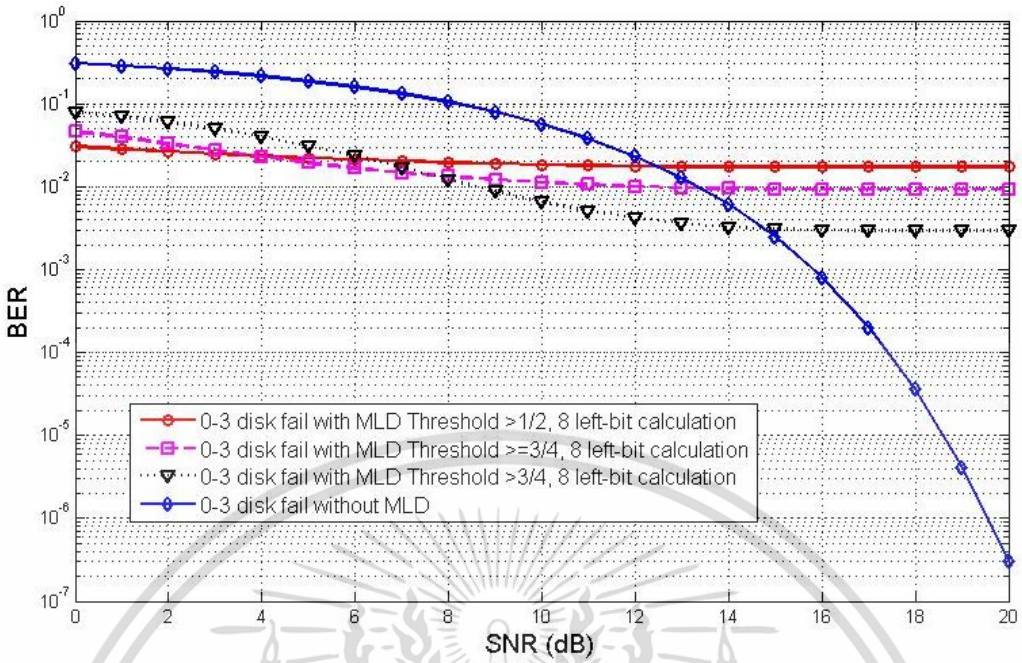
รูปที่ 4.27 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 6 บิต

ผลการทดลองรูปที่ 4.27 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 6 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 7 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



รูปที่ 4.28 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 7 บิต

ผลการทดลองรูปที่ 4.28 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 7 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 6 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)



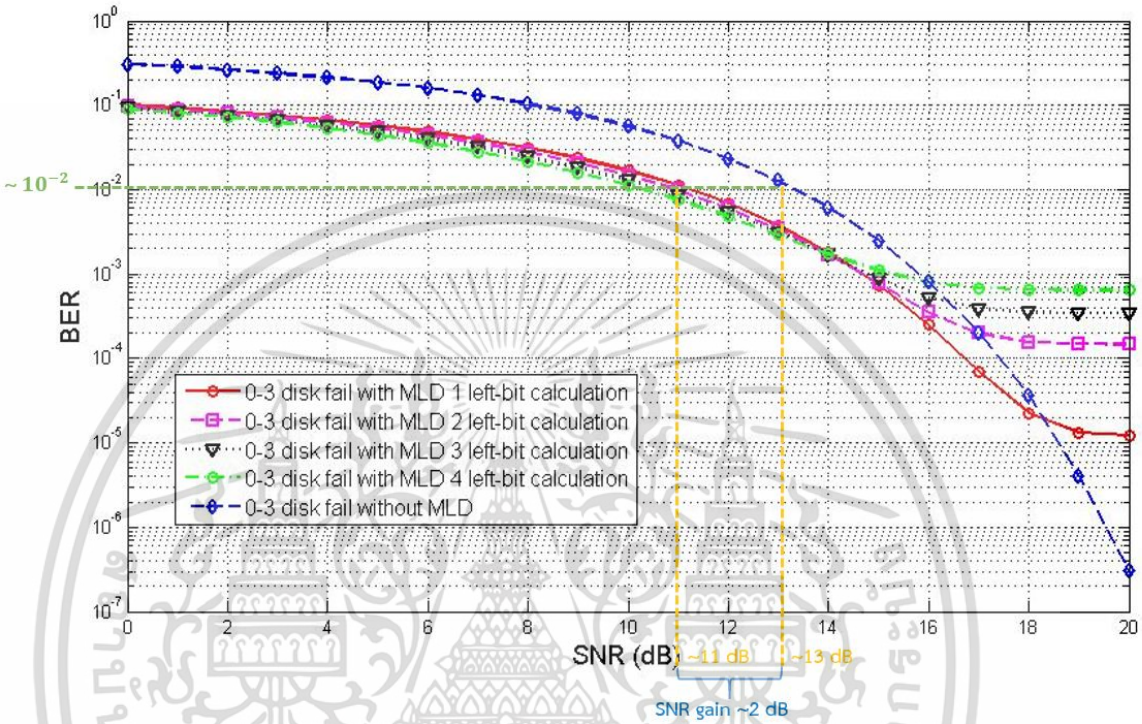
รูปที่ 4.29 ค่า BER จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB ขนาด 600x600 จุดภาพ เมื่อค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$, $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 8 บิต

ผลการทดลองรูปที่ 4.29 เป็นผลการทดลอง MLD กับภาพนอน-ไบนารี่แบบ RGB ซึ่งตั้งค่าอัลกอริทึม MLD ที่พิจารณาบิต 8 บิตนับจากบิต MSB ซึ่งให้ผลลัพธ์คือค่า BER ของ T_h จะเริ่มแตกต่างกันเมื่อ SNR > 5 dB ซึ่ง $T_h > \frac{1}{2}$ จะให้ BER ที่แย่ที่สุด (BER สูง) $T_h \geq \frac{3}{4}$ จะให้ BER ที่ปานกลางและ $T_h > \frac{3}{4}$ จะให้ BER ที่ดีที่สุด (BER ต่ำ)

จากผลการทดลองทั้งหมดแสดงให้เห็นว่า ข้อมูลรูปภาพแบบนอน-ไบนารี่นั้นซึ่งมีความซับซ้อนของข้อมูลสูงกว่าข้อมูลรูปภาพแบบไบนารี่นั้นให้ประสิทธิภาพการแก้ไขข้อมูลผิดพลาดของอัลกอริทึม MLD ที่ต่ำกว่า ซึ่งสามารถสรุปได้ได้ดังนี้

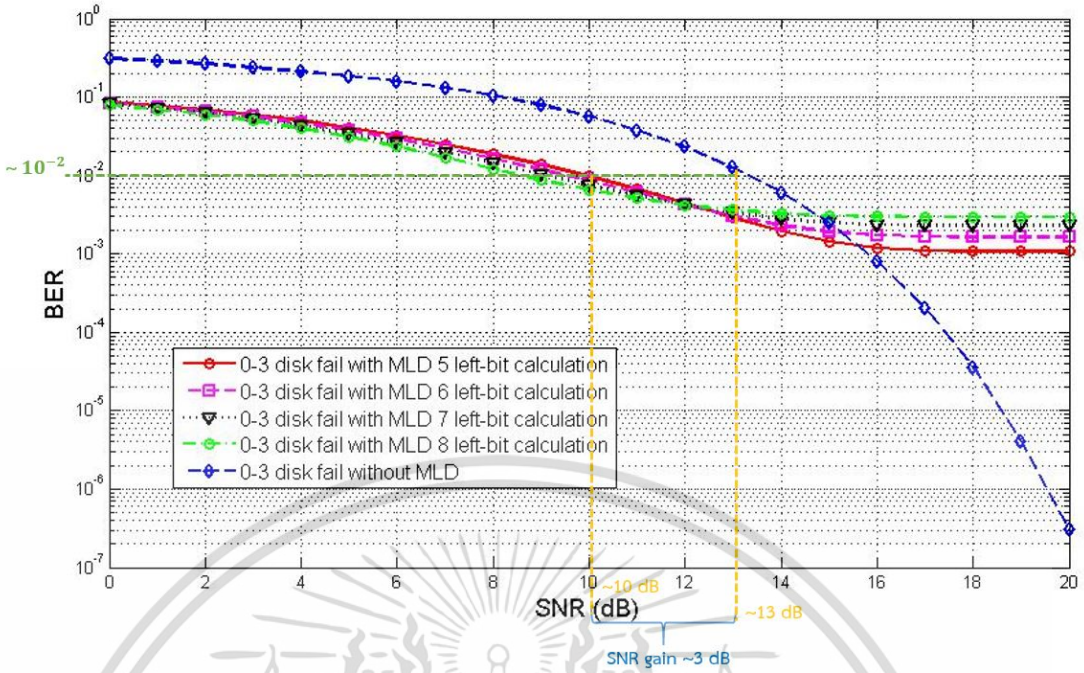
1. รูปภาพนอน-ไบนารี่ชนิด Grayscale จากผลการทดลองดังรูปที่ 4.14, 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21 แสดงให้เห็นว่าภาพที่มีความสัมพันธ์กันของบิตรอบข้างต่ำ (จุดภาพแต่ละจุดที่ใกล้เคียงกันมีความเข้มสีที่แตกต่างกันมาก) ให้ประสิทธิภาพแก้ไขข้อมูลผิดพลาดที่น้อยมาก (BER ลดลงน้อยกว่า 10^{-3})
2. รูปภาพนอน-ไบนารี่ชนิด RGB จากผลการทดลองดังรูปที่ 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28, 4.29 แสดงให้เห็นว่าภาพที่มีความซับซ้อนของบิตในแต่ละจุดภาพสูง แต่ยังคงมีความสัมพันธ์กันของบิตรอบข้างอยู่บ้าง (จุดภาพแต่ละจุดที่ใกล้เคียงกันมีความเข้มสีที่แตกต่างกันไม่มาก) ให้ประสิทธิภาพแก้ไขข้อมูลผิดพลาดในระดับปานกลาง (BER ลดลงประมาณ 7×10^{-2})

จากการทดลองสรุปได้ว่าเมื่อกำหนดจำนวนบิตที่นำมาพิจารณาอัลกอริทึม MLD ที่แตกต่างกัน (1-8 บิตโดยเริ่มต้นจากบิตด้านซ้าย) จะให้ประสิทธิภาพแก้ไขข้อมูลผิดพลาดที่แตกต่าง เมื่อพิจารณาอัลกอริทึมที่มีค่า $T_h > \frac{3}{4}$ ซึ่งให้ประสิทธิภาพแก้ไขข้อมูลผิดพลาดที่เหมาะสมที่สุดสำหรับข้อมูลภาพนอน-ไบนารี่ดังรูปที่ 4.30 และ 4.31



รูปที่ 4.30 ประสิทธิภาพ MLD เมื่อค่า BER ที่ 10^{-2} จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB เมื่อค่า $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 1-4 บิต

ผลการทดลองรูปที่ 4.30 ผู้วิจัยได้พล็อตกราฟรูปภาพ RGB ทีค่า $T_h > \frac{3}{4}$ โดยประกอบด้วยข้อมูล MLD ที่ตั้งค่าการพิจารณาบิต 1-4 บิตนับจากบิต MSB ซึ่งได้ผลคือที่ BER เท่ากับ 10^{-2} พบว่ากรณีที่ใช้ MLD ใช้ SNR ประมาณ 11 dB ซึ่งต่างจากกรณีที่ไม่ใช้ MLD ซึ่งใช้ SNR ประมาณ 13 dB (SNR gain = 2dB) แต่จะให้ผลลัพธ์ที่แย่กว่ากรณีที่ไม่ใช้ MLD เมื่อ SNR มากกว่า 16 dB



รูปที่ 4.31 ประสิทธิภาพ MLD เมื่อค่า BER ที่ 10^{-2} จากการถอดรหัสรูปภาพนอน-ไบนารี่ชนิด RGB เมื่อค่า $T_h > \frac{3}{4}$ และบิตที่นำมาพิจารณาจำนวน 5-8 บิต

ผลการทดลองรูปที่ 4.31 ผู้วิจัยได้พล็อตกราฟรูปภาพ RGB ที่ค่า $T_h > \frac{3}{4}$ โดยประกอบด้วยข้อมูล MLD ที่ตั้งค่าการพิจารณาบิต 5-8 บิตนับจากบิต MSB ซึ่งได้ผลคือที่ BER เท่ากับ 10^{-2} พบว่ากรณีที่ใช้ MLD ใช้ SNR ประมาณ 10 dB ซึ่งต่างจากกรณีที่ไม่ใช้ MLD ซึ่งใช้ SNR ประมาณ 13 dB (SNR gain = 3 dB) แต่จะให้ผลลัพธ์ที่แย่กว่ากรณีที่ไม่ใช้ MLD เมื่อ SNR มากกว่า 13 dB

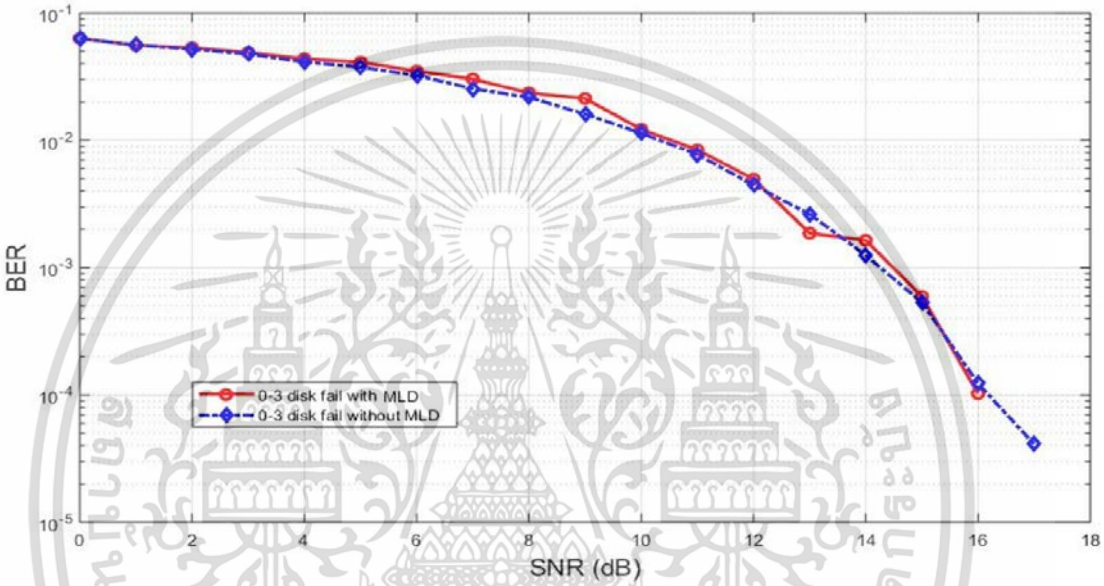
จากผลการทดลองข้างต้น สามารถพิจารณาการตั้งค่าพารามิเตอร์ของอัลกอริทึม MLD ที่ค่า $T_h > \frac{3}{4}$ จากรูปที่ 4.30 และ 4.31 ที่อัตราบิตผิดพลาดมีค่า 10^{-2} หรือในการรับส่งข้อมูลจำนวน 1,000 บิต พบว่ามีบิตผิดพลาด 1 บิตนั้น ในกรณีข้อมูลภาพนอน-ไบนารี่ไม่ได้ใช้อัลกอริทึม MLD ใช้ SNR 13 dB แต่เมื่อใช้บิตจำนวน 1-4 บิต มาพิจารณาแก้ไขข้อมูลผิดพลาดด้วยอัลกอริทึม MLD พบว่าใช้ SNR ประมาณ 11 dB ซึ่งสามารถลด SNR ในการส่งข้อมูลได้ 2 dB (จากรูปที่ 4.30) และเมื่อใช้บิตจำนวน 5-8 บิต มาพิจารณาแก้ไขข้อมูลผิดพลาดด้วยอัลกอริทึม MLD พบว่าใช้ SNR ประมาณ 10 dB ซึ่งสามารถลด SNR ในการส่งข้อมูลได้ 3 dB (จากรูปที่ 4.31) ซึ่งมากกว่าเมื่อใช้บิตจำนวน 1-4 บิตมาพิจารณา MLD แต่เมื่อพิจารณาที่ SNR ที่สูงขึ้น MLD เมื่อพิจารณาบิตจำนวน 3-8 บิตให้ค่า BER ที่เทียบเท่ากับกรณีที่ไม่ใช้ MLD ที่ SNR เท่ากับ 16 dB มีเพียง MLD เมื่อพิจารณาบิตจำนวน 1 และ 2 บิต ซึ่งให้ค่า BER เทียบเท่ากับกรณีที่ไม่ใช้ MLD ที่ SNR เท่ากับ 18 dB และ 17 dB ตามลำดับ ซึ่งสรุปได้ดังตารางที่ 4.4

ตารางที่ 4.4 สรุปผลการทดลองข้อมูลภาพแบบนอน-ไบนารี่กับอัลกอริทึมการถอดรหัสด้วยความน่าจะเป็นสูงสุด

ประเภทของข้อมูล	ความสัมพันธ์ของบิตรอบข้าง	ค่าพารามิเตอร์ของ MLD	คำอธิบาย
รูปภาพชนิด Grayscale	ต่ำ	ขีดเริ่มเปลี่ยน (T_h)	อัลกอริทึม MLD ที่ T_h ทุกค่า ไม่เหมาะสมกับการแก้ไขข้อมูลรูปภาพที่มีความสัมพันธ์ของบิตรอบข้างต่ำ
		จำนวนบิตที่มาพิจารณา	
รูปภาพชนิด RGB	ปานกลาง	ขีดเริ่มเปลี่ยน (T_h)	ค่า $T_h > \frac{1}{2}$, $T_h \geq \frac{3}{4}$ และ $T_h > \frac{3}{4}$ ให้ประสิทธิภาพแก้ไขข้อมูลน้อย ปานกลาง ดีที่สุดตามลำดับ เมื่อ SNR มีค่ามากขึ้น
		จำนวนบิตที่มาพิจารณา	จำนวนบิตที่นำมาพิจารณาใน MLD น้อย จะให้ประสิทธิภาพแก้ไขข้อมูลได้ดีกว่าเมื่อ SNR มีค่ามากขึ้น (จำนวนบิต $\propto \frac{1}{\text{ประสิทธิภาพการแก้ไขข้อมูล}}$)

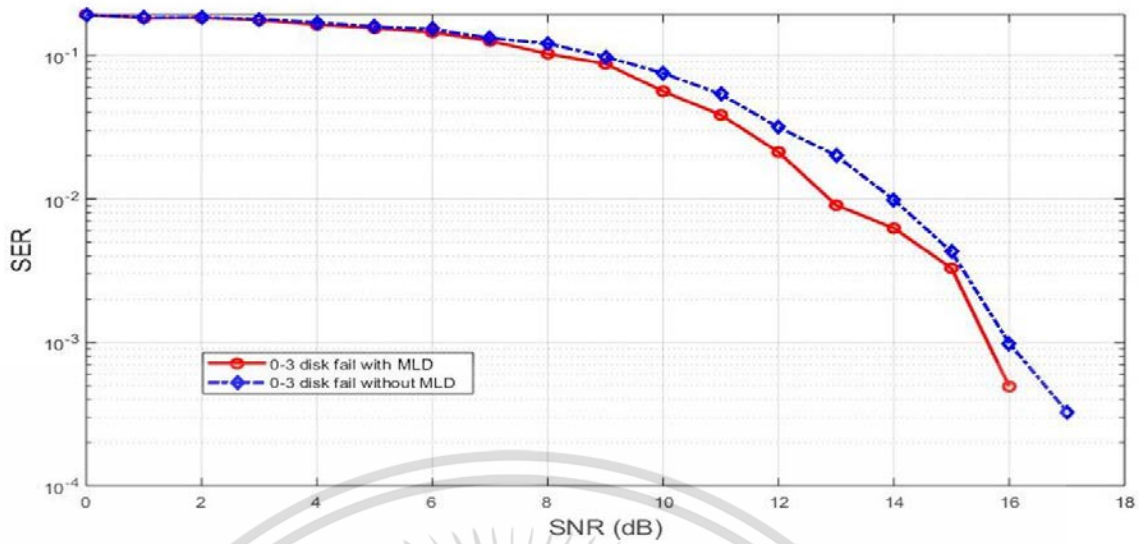
4.4 ผลการทดลองข้อมูลแบบเอกสารกับอัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุด

จากการทดลองข้อมูลบทความทั้งหมด 5 บทความ บทความละประมาณ 200 คำ โดยใช้ฐานข้อมูลจากบทความภาษาอังกฤษทางอินเตอร์เน็ตทั้งหมดประมาณ 5,000 คำ ซึ่งพล็อตกราฟอัตราบิตผิดพลาด (BER) และอัตราความผิดพลาดของสัญลักษณ์ (SER) ได้ผลดังรูปที่ 4.32 และ 4.33 ตามลำดับ



รูปที่ 4.32 ค่า BER จากการถอดรหัสข้อความจำนวน 5 บทความ

ผลการทดลองรูปที่ 4.32 แสดงผลการถอดรหัสข้อความจำนวน 5 บทความ โดยพล็อตกราฟ BER เทียบกับ SNR ที่ค่าต่างๆ ซึ่งพบว่าจากกราฟกรณีที่ใช้/ไม่ใช้ MLD ให้ผล BER ที่มีค่าใกล้เคียงกัน เนื่องจากในกรณีที่อัลกอริทึมแก้ไขคำที่ถูกให้กลายเป็นคำที่ผิดไปเพียงหนึ่งคำ โดยคำนั้นประกอบด้วยตัวอักษรหลายตัว และตัวอักษรแต่ละตัวประกอบบิตจำนวน 8 บิต จะทำให้ค่า BER เพิ่มขึ้นอย่างมาก



รูปที่ 4.33 ค่า SER จากการถอดรหัสข้อความจำนวน 5 บทความ

ผลการทดลองรูปที่ 4.33 แสดงผลการถอดรหัสข้อความจำนวน 5 บทความ โดยพล็อตกราฟ SER (Symbol error rate) เทียบกับ SNR ที่ค่าต่างๆ ซึ่งพบว่าจากกราฟ SER กรณีที่ใช้ MLD มีค่าดีกว่าเมื่อ SNR > 8 dB

จากผลการทดลองข้างต้นพบว่า เมื่อพิจารณาค่า BER เทียบกับ SNR ที่ค่าต่างๆ ดังรูปที่ 4.32 พบว่าในกรณีที่ใช้ MLD และไม่ใช้ MLD ในการแก้ไขไฟล์ข้อความหลังผ่านสัญญาณรบกวนนั้นให้ผลลัพธ์ที่ไม่แตกต่างกัน เนื่องจากข้อความหนึ่งคำประกอบด้วยตัวอักษรหลายตัว และในตัวอักษรแต่ละตัวประกอบด้วยบิตจำนวน 8 บิต โดยกรณีที่แก้ไขข้อความหนึ่งๆ ผิดไปจากข้อความต้นแบบ จะส่งผลให้ค่า BER เพิ่มขึ้นมาก และเมื่อพิจารณาค่า SER เทียบกับ SNR ดังรูปที่ 4.33 จะพบว่าค่า SER หลังการใช้ MLD ให้ผลลัพธ์ที่ดีกว่าเล็กน้อยในกรณีที่ไม่ใช้ MLD เนื่องจากที่ค่า SNR ที่สูง ไฟล์ข้อความจะมีความถูกต้องที่สูงอยู่แล้ว ซึ่ง MLD อาจจะแก้ไขไฟล์ข้อความนั้นจากถูกต้องเป็นผิด ซึ่งส่งผลให้ค่า BER และ SER มีค่าสูง

จากการประยุกต์ใช้ ngram มาช่วยในการแก้ไขไฟล์ข้อความพบว่าเมื่อสัญลักษณ์แบ่งคำถูกสัญญาณรบกวนเปลี่ยนเป็นตัวอักษรอื่น MLD จะพิจารณาคำนั้นผิดพลาดไป ตัวอย่างเช่น เมื่อคำสอง ซึ่งประกอบด้วย give (4 ตัวอักษร), space และ up (2 ตัวอักษร) ถูกสัญญาณรบกวนทำให้สัญลักษณ์ space ผิดเพี้ยนกลายเป็นตัวอักษร ดังรูปที่ 4.34 จะทำให้คำสองคำนั้นถูกอัลกอริทึมพิจารณาเป็นคำหนึ่งคำที่มีขนาด 7 ตัวอักษร

g	i	v	e	B	u	p

รูปที่ 4.34 ตัวอย่างข้อมูลเอกสารที่มีสัญลักษณ์แบ่งคำผิดเพี้ยน

บทที่ 5
สรุปผลการทดลอง

บทสุดท้ายนี้จะกล่าวถึงบทสรุปของงานวิจัยชิ้นนี้ รวมทั้งสรุปผลการทดลองโดยพิจารณาจากผลการทดลองในบทที่ 4

5.1 สรุปผลการทดลอง

งานวิจัยนี้ผู้วิจัยได้ศึกษาและประยุกต์ใช้กระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุด มาทดลองแก้ไขข้อมูลที่ผ่านช่องสัญญาณที่ถูกรบกวนจากสัญญาณรบกวน ซึ่งข้อมูลที่นำมาทดลองในงานวิจัยนี้ประกอบด้วยข้อมูลภาพแบบไบนารี่ ข้อมูลภาพแบบนอน-ไบนารี่ และข้อมูลไฟล์ข้อความเอกสาร โดยตั้งค่าการทดลองให้ข้อมูลนั้นถูกเก็บอยู่ในระบบเก็บข้อมูลแบบกลุ่มเมฆซึ่งเป็นระบบจัดเก็บข้อมูลแบบกระจายชนิดหนึ่งและมีการป้องกันข้อมูลสูญหายด้วยรหัสอีเรเซอร์คือรหัส CGR โดยทดลองสุ่มให้หน่วยความจำภายในกลุ่มเมฆนั้นเสียหายในเงื่อนไขที่รหัส CGR สามารถกู้ข้อมูลคืนได้ ซึ่งข้อมูลจะถูกส่งผ่านช่องสัญญาณที่ถูกสัญญาณรบกวนมายังผู้รับ ที่มีการใช้อัลกอริทึม MLD ในส่วนของเครื่องผู้รับและทดสอบกับไฟล์ข้อมูลต่างประเภทกัน ซึ่งจากผลการทดลองสามารถสรุปได้ดังนี้

1. อัลกอริทึม MLD เหมาะสมกับข้อมูลรูปภาพที่มีความสัมพันธ์กันระหว่างบิตที่สูง และค่าขีดเริ่มเปลี่ยน (T_h) ที่มีค่าสูงขึ้นเหมาะกับช่องสัญญาณที่ค่า SNR สูง
2. ข้อมูลภาพที่มีความซับซ้อนของบิตต่ำ (รูปภาพแบบไบนารี่) จะให้ผลการแก้ไขข้อมูลที่ดีกว่าข้อมูลภาพที่มีความซับซ้อนของบิตสูง (รูปภาพแบบนอน-ไบนารี่)
3. สำหรับข้อมูลภาพแบบนอน-ไบนารี่หากจำนวนบิตจาก MSB บิตที่นำมาพิจารณาใน MLD น้อยจะทำให้มีผลการแก้ไขข้อมูลที่ดีกว่าในกรณีที่ช่องสัญญาณมีค่า SNR สูง
4. ข้อมูลไฟล์ข้อความเมื่อใช้ MLD ในการแก้ไขข้อมูลพบว่าค่า BER ไม่ดีกว่ากรณีที่ไม่ใช้ MLD แต่เมื่อพิจารณา SER จะพบว่าให้ผลการแก้ไขข้อมูลที่ดีขึ้นเล็กน้อย เนื่องจากในแต่ละตัวอักษรประกอบด้วยบิตจำนวน 8 บิต ในการแก้ไขตัวอักษรที่ถูกต้องเป็นตัวอักษรที่ผิด จะเกิด BER ที่สูงขึ้นด้วย

5.2 ข้อเสนอแนะ

5.2.1 ปัญหาที่พบในงานวิจัยนี้

งานวิจัยชิ้นนี้ศึกษากระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุด (MLD) เพื่อนำมาประยุกต์ใช้ในการแก้ไขข้อมูลผิดพลาดในข้อมูลหลากหลายชนิด พบว่ากระบวนการถอดรหัสด้วยความน่าจะเป็นสูงสุดนี้มีข้อจำกัดในการนำมาใช้งาน ซึ่งข้อมูลแต่ละประเภทนั้นจะมีโครงสร้าง การจัดเรียงข้อมูล และความสัมพันธ์ของข้อมูลที่ต่างกัน เช่น ข้อมูลภาพเป็นข้อมูลที่จุดภาพจะมีความสัมพันธ์กัน หรือข้อมูลเอกสารซึ่งข้อมูลจะถูกเก็บในรูปแบบของรหัสแอสกี (ASCII code)

5.2.2 งานที่จะทำในอนาคต

ศึกษาโครงสร้าง การจัดเรียงข้อมูล และความสัมพันธ์ของข้อมูลประเภทต่างๆ วิเคราะห์และค้นคว้าหากระบวนการที่สามารถจัดการกับข้อมูลที่เฉพาะเจาะจงมากยิ่งขึ้น เพื่อเพิ่มประสิทธิภาพการแก้ไขข้อมูลให้มากขึ้น

เอกสารอ้างอิง

[1] N. Puttarak. 2011. “Coding for storage: disk arrays, flash memory, and distributed storage networks” Ph.D.dissertation, Lehigh University.

[2] G. Kulkarni, R. Waghmare, R.Palwe, V. Waykule, H. Bankar, K.Koli. 2012. “Cloud Storage Architecture” 76-81. TSSA. Bali, Indonesia.

[3] J. Wu, L. Ping, X. Ge, Y. Wang, J. Fu. 2010. “Cloud Storage as the Infrastructure of Cloud Computing”. 380-383. ICICCI. Kuala Lumpur, Malaysia.

[4] J. S. Plank. 1997. “A Tutorial on Reed-Solomon Coding for Fault Tolerance in RAID-like Systems” 995-1012. Software Practice and Experience 27(9). New York, USA.

[5] M. Blaum, J. Brady, J. Bruck, J. Menon. 1995. “EVENODD: An efficient scheme for tolerating double disk failures in RAID architecture” 192-202. IEEE Transactions on Computers.

[6] N. Puttarak, P. Kaewprapha, B. Chong, J. Li. 2009. “A New Class of MDS Erasure Codes Based on Graphs” 1-6. IEEE Global Telecommunications Conference. Honolulu, HI, USA.

[7] J. S. Plank. 2013. “Erasure Codes for Storage Systems : A Brief Primer” 44-50. ;login: The USENIX Magazine Vol.38 No.6.

[8] M. A. Song, I. F. Lan, S. Y. Kuo. 2005. “An Area-Efficient Architecture of Reed-Solomon Codec for Advanced RAID System”. ICPADS’05. Fukuoka, Japan.

[9] W. Kechao, J Zongfu, Z. Ming kui, Y. Jingwei. 2011. “Research of Network Storage System based on Erasure Code” 1470-1474. EMEIT. Harbin, China.

[10] N. Traore, S. Kant, T. Lindstrom Jensen, I. Land, J. Dahl, L. Kriestensen. 2016. “Message Passing Algorithm and Linear Programming Decoding for LDPC and Linear Block Codes” Aalborg University.

[11] R. Gallager. 1962. “Low Density Parity Check codes” 21-28. IEEE Trans. Information Theory 8 No.1.

[12] K. Sunil, P. Jayaraj, K. P. Soman. 2012. “Message Passing Algorithm : A Tutorial Review” 12-24. IOSR Journal of Computer Engineering Vol.2 Issue 3.

[13] V. A. Chandrasetty, S. M. Aziz. 2009. “A Reduced Complexity Message Passing Algorithm with Improved Performance for LDPC Decoding” 19-24. ICCIT. Dhaka, Bangladesh.

[14] T. Ta. “A Tutorial on Low Density Parity-Check Codes”, The University of Texas at Austin

[15] S. Ghemawat, H. Gobioff, and S.-T. Leung. 2003. “The Google File System” 29-43. SOSP. New York, USA.

[16] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. 2007. “Dynamo: Amazon’s Highly Available Key-Value Store” 205-220. SOSP. Washington, USA.

[17] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav,J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. Fahim ul Haq, M. Ikram ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas. 2011. “Windows Azure storage: A highly available cloud storage service with strong consistency. In Symposium on Operating Systems Principles” 143-157. SOSP. Cascais, Portugal.

[18] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, 2000. “Oceanstore: An Architecture for Global-Scale Persistent Storage” 190-201. ASPLOC. MA, USA.

[19] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G.M. Voelker, 2004. “Total Recall: System Support for Automated Availability Management” 337-350. NSDI. California, USA.

[20] L. Cattani, G. Ntonfo, F.Lofino, G. Ferrari, R. Raheli, F. Pisani. 2014. “Maximum-Likelihood Detection of Neonatal Clonic Seizures by Video Image Processing” 1-5. ISMICT. Firenze, Italy.

[21] S. S. Wanarse, T. G. Patil, S. S. Patankar, J. V. Kulkarni. 2014. “Class quantification of aerial images using Maximum Likelihood Estimation” 345-347. ICNSC. Guntur, India.

[22] H. Kroll, M. Korb, B. Weber, S. Willi, Q. Huang. 2017. “Maximum-Likelihood Detection for Energy-Efficient Timing Acquisition in NB-Iot” 1-5. WCNCW. San Francisco, CA, USA.

[23] J. Dumoulin. 2012. “Smoothing of ngram language models of human chats” 1-4. SCIS/ISIS. Kobe, Japan.

ภาคผนวก ก

อัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบไบนารี่

Step 1 : Identify size of image Matrix P : n,m

: Identify surrounding bit of $C_{i,j}$: X

Step2 : Start correct bit of $C_{i,j}$ where $0 < i \le n$ and $0 < j \le m$

for $0 < i \le n$

for $0 < j \le m$

switch i, j **do**

case I : $i = 1$ (bit of C is at the first row)

case I - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i+1,j}\ P_{i+1,j+1}]$

case I - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i+1,j-1}\ P_{i+1,j}]$

case I - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i+1,j-1}\ P_{i+1,j}\ P_{i+1,j+1}]$

case II : $i = n$ (bit of C is at the last row)

case II - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i-1,j}\ P_{i-1,j+1}]$

case II - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i-1,j-1}\ P_{i-1,j}]$

case II - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i-1,j+1}]$

case III : $1 < i < n$ (otherwise)

case III - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i-1,j}\ P_{i-1,j+1}\ P_{i+1,j}\ P_{i+1,j+1}]$

case III - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i+1,j-1}\ P_{i+1,j}]$

case III - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i-1,j+1}\ P_{i+1,j-1}\ P_{i+1,j}\ P_{i+1,j+1}]$

end switch

$\mu_{C=1} = \frac{1}{r}\sum_{i=1}^{r} X[i]$, for any $r \in X$

$\mu_{C=0} = 1 - \mu_{C=1}$

$C_{i,j} \leftarrow \{1 \mid \text{compare } \mu_{C=1} \text{ with } T_h, 0 \mid \text{compare } \mu_{C=0} \text{ with } T_h\}$

end for

end for

อัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุดกับข้อมูลภาพแบบนอน-ไบนารี่

Step 1 : Identify size of image Matrix P : n, m, b

: Where n : row, m : column, b : number of bit

: Identify surrounding bit of $C_{i,j,k}$: X

Step2 : Start correct bit of $C_{i,j,k}$ where $0 < i \le n$, $0 < j \le m, 0 < k \le b$

for $0 < i \le n$
for $0 < j \le m$
for $0 < k \le b$
switch *i, j* **do**
case I : $i = 1$ (bit of C is at the first row)
case I - A : $j = 1$ **step** : $X = [P_{i,j+1,k}\ P_{i+1,j,k}\ P_{i+1,j+1,k}]$
case I - B : $j = m$ **step** : $X = [P_{i,j-1,k}\ P_{i+1,j-1,k}\ P_{i+1,j,k}]$
case I - C : $1 < j < m$ **step** : $X = [P_{i,j-1,k}\ P_{i,j+1,k}\ P_{i+1,j-1,k}\ P_{i+1,j,k}\ P_{i+1,j+1,k}]$
case II : $i = n$ (bit of C is at the last row)
case II - A : $j = 1$ **step** : $X = [P_{i,j+1,k}\ P_{i-1,j,k}\ P_{i-1,j+1,k}]$
case II - B : $j = m$ **step** : $X = [P_{i,j-1,k}\ P_{i-1,j-1,k}\ P_{i-1,j,k}]$
case II - C : $1 < j < m$ **step** : $X = [P_{i,j-1,k}\ P_{i,j+1,k}\ P_{i-1,j-1,k}\ P_{i-1,j,k}\ P_{i-1,j+1,k}]$
case III : $1 < i < n$ (otherwise)
case III - A : $j = 1$ **step** : $X = [P_{i,j+1,k}\ P_{i-1,j,k}\ P_{i-1,j+1,k}\ P_{i+1,j,k}\ P_{i+1,j+1,k}]$
case III - B : $j = m$ **step** : $X = [P_{i,j-1,k}\ P_{i-1,j-1,k}\ P_{i-1,j,k}\ P_{i+1,j-1,k}\ P_{i+1,j,k}]$
case III - C : $1 < j < m$ **step** : $X = [P_{i,j-1,k}\ P_{i,j+1,k}\ P_{i-1,j-1,k}\ P_{i-1,j,k}\ P_{i-1,j+1,k}\ P_{i+1,j-1,k}\ P_{i+1,j,k}\ P_{i+1,j+1,k}]$
end switch

$\mu_{k_{C=1}} = \frac{1}{r}\sum_{i=1}^{r} X_k[i]$, for any $r \in X$

$\mu_{k_{C=0}} = 1 - \mu_{k_{C=1}}$

$C_{i,j,k} \leftarrow \{1 \mid \text{compare } \mu_{k_{C=1}} \text{ with } T_h, 0 \mid \text{compare } \mu_{k_{C=0}} \text{ with } T_h\}$

end for
end for
end for

อัลกอริทึมการตรวจจับด้วยความน่าจะเป็นสูงสุดกับข้อมูลแบบเอกสาร

Step 1 : Identify word of corpus : $W = \{W_1, W_2, W_3, ..., W_n\}$

: Identify letter of text files : $C = \{C_1, C_2, C_3, ..., C_m\}$

: Identify interested word : D

: Identify letter of text files with MLD : $T = \{T_1, T_2, T_3, ..., T_m\}$

Step2 : Start correct word of D where $0 < i \le m$,

for $0 < i \le m$

switch C_i **do**

case I : $C_i \neq$ Special character {, : ; ! . ? - ( ) space}

step : $D \leftarrow \{C_i\}$

case II : $C_i =$ Special character {, : ; ! . ? - ( ) space}

step : compare D with W

find $\operatorname{argmax}(D / W_x) = p(f_{w_x}) \frac{1}{n} \sum_{i=1}^{n} (d_i \oplus w_{x,i})$

$T_{(p+1):(i-1)} \leftarrow \{W_x\}$ Where : p is the previous position of special character

clear D

end switch

end for

ภาคผนวก ข

โค้ดแมทแลป (Matlab code) สำหรับอัลกอริทึมถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลรูปภาพแบบไบนารี่

```
function Result = MLD(DataMatrix,ratio)
    [n,m] = size(DataMatrix);
    D = DataMatrix;
    for i=1:n
        for j=1:m
            if i==1
                if j == 1
                    C = [D(i,j+1) D(i+1,j) D(i+1,j+1)];
                elseif j == m
                    C = [D(i,j-1) D(i+1,j-1) D(i+1,j)];
                else
                    C = [D(i,j-1) D(i,j+1) D(i+1,j-1) D(i+1,j) D(i+1,j+1)];
                end
            end
            if i==n
                if j == 1
                    C = [D(i,j+1) D(i-1,j) D(i-1,j+1)];
                elseif j == m
                    C = [D(i,j-1) D(i-1,j-1) D(i-1,j)];
                else
                    C = [D(i,j-1) D(i,j+1) D(i-1,j-1) D(i-1,j) D(i-1,j+1)];
                end
            end
            if (1<i)&&(i<n)
                if j == 1
                    C = [D(i,j+1) D(i-1,j) D(i-1,j+1) D(i+1,j) D(i+1,j+1)];
                elseif j == m
                    C = [D(i,j-1) D(i-1,j-1) D(i-1,j) D(i+1,j-1) D(i+1,j)];
                else
```

```
                C = [D(i,j-1) D(i,j+1) D(i-1,j-1) D(i-1,j) D(i-1,j+1) D(i+1,j-1) D(i+1,j)
D(i+1,j+1)];
                end
            end
            one = 0;
            zero = 0;
            [r,s] = size(C);
            one = sum(C,2);
            zero = s - one;
            if one >= s*ratio
                D(i,j) = 1;
            elseif zero >= s*ratio
                D(i,j) = 0;
            end
        end
    end
    Result = D;
end
```

โค้ดแมทแลป (Matlab code) สำหรับอัลกอริทึมถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลรูปภาพแบบนอน-ไบนารี่

```
function Result = MLD8bit(DataMatrix,ratio,bitmeasure)
    [n,m] = size(DataMatrix);
    y = reshape(transpose(DataMatrix),[1 n*m]);
    D = de2bi(y,8,'left-msb');
    for i=1:n*m
        clear C;
        if i<=m
            if i==1
                for j=1:8
                    C(:,j) = [D(i+1,j) D(i+m,j) D(i+m+1,j)];
                end
            elseif i==m
                for j=1:8
                    C(:,j) = [D(i-1,j) D(i+m,j) D(i+m-1,j)];
                end
            else
                for j=1:8
                    C(:,j) = [D(i-1,j) D(i+1,j) D(i+m,j) D(i+m-1,j) D(i+m+1,j)];
                end
            end
        end
        if (((n-1)*m)+1<= i)&&(i<=n*m)
            if i==((n-1)*m)+1
                for j=1:8
                    C(:,j) = [D(i+1,j) D(i-m,j) D(i-m+1,j)];
                end
            elseif i==(n*m)
                for j=1:8
                    C(:,j) = [D(i-1,j) D(i-m,j) D(i-m-1,j)];
                end
```

```
            else
                for j=1:8
                    C(:,j) = [D(i-1,j) D(i+1,j) D(i-m-1,j) D(i-m,j) D(i-m+1,j)];
                end
            end
        end
        if (m<i)&&(i<((n-1)*m)+1)
            if mod(i,m)==1
                for j=1:8
                    C(:,j) = [D(i-m,j) D(i-m+1,j) D(i+1,j) D(i+m,j) D(i+m+1,j)];
                end
            elseif mod(i,m)==0
                for j=1:8
                    C(:,j) = [D(i-m,j) D(i-m-1,j) D(i-1,j) D(i+m,j) D(i+m-1,j)];
                end
            else
                for j=1:8
                    C(:,j) = [D(i-m-1,j) D(i-m,j) D(i-m+1,j) D(i-1,j) D(i+1,j) D(i+m-1,j) D(i+m,j)
D(i+m+1,j)];
                end
            end
        end
        [r,s] = size(C);
        C_Check_one = sum(C,1);
        C_Check_zero = r - C_Check_one;
        for j=1:bitmeasure
            if C_Check_one(1,j) >= r*ratio
                D(i,j) = 1;
            elseif C_Check_zero(1,j) >= r*ratio
                D(i,j) = 0;
            end
        end
    end
```

```
    Result = D;
end
```

โค้ดแมทแลป (Matlab code) สำหรับอัลกอริทึมถอดรหัสด้วยความน่าจะเป็นสูงสุดกับข้อมูลแบบเอกสาร

```
function Result = Textcollectionngram1(Sentence,bag)
    Char = num2cell(char(Sentence));
    [~,b]=size(Char);
    count =0;
    [~,totalword] = size(bag.Vocabulary);
    ngram1 = topkngrams(bag,totalword,'NGramLengths',1);
    W_ngram1 = ngram1(:,1).Variables;
    W_Check_ngram1 = lower(num2cell(char(W_ngram1(:,1))));
    [ngram1_Row,ngram1_Column] = size(W_Check_ngram1);
    Tem = cell(0);
    Check_Tem = zeros(ngram1_Row,ngram1_Column);
    for i=1:b
        if
(Char{1,i}==',')||(Char{1,i}=='.')||(Char{1,i}==':')||(Char{1,i}==';')||(Char{1,i}=='!')||(Char{1,i}=='?')
||(Char{1,i}=='-')||(Char{1,i}==' ')||(Char{1,i}=="")||(Char{1,i}=='(')||(Char{1,i}==')')
            [~,SizeTem] = size(Tem);
            if (SizeTem ~= 0) && (SizeTem <= ngram1_Column)
                [~,Tem_Column] = size(Tem);
                for Tem_number = (Tem_Column+1):ngram1_Column
                    Tem(1,Tem_number) = {' '};
                end
                for j=1:ngram1_Row
                    Check_Tem(j,:) = strcmpi(Tem(1,:),W_Check_ngram1(j,:));  % Don't care
about Upper& Lower <---
                end
                Check_Tem_c = sum(Check_Tem,2);
                Check_Upper = isstrprop(Tem(1,1),'Upper');
                Check_Upper_All = sum(cell2mat(isstrprop(Tem(1,:),'Upper')),2);
                Max_Check_Tem = max(Check_Tem_c);
                Position_max_ngram1 = find(Check_Tem_c == Max_Check_Tem);
```

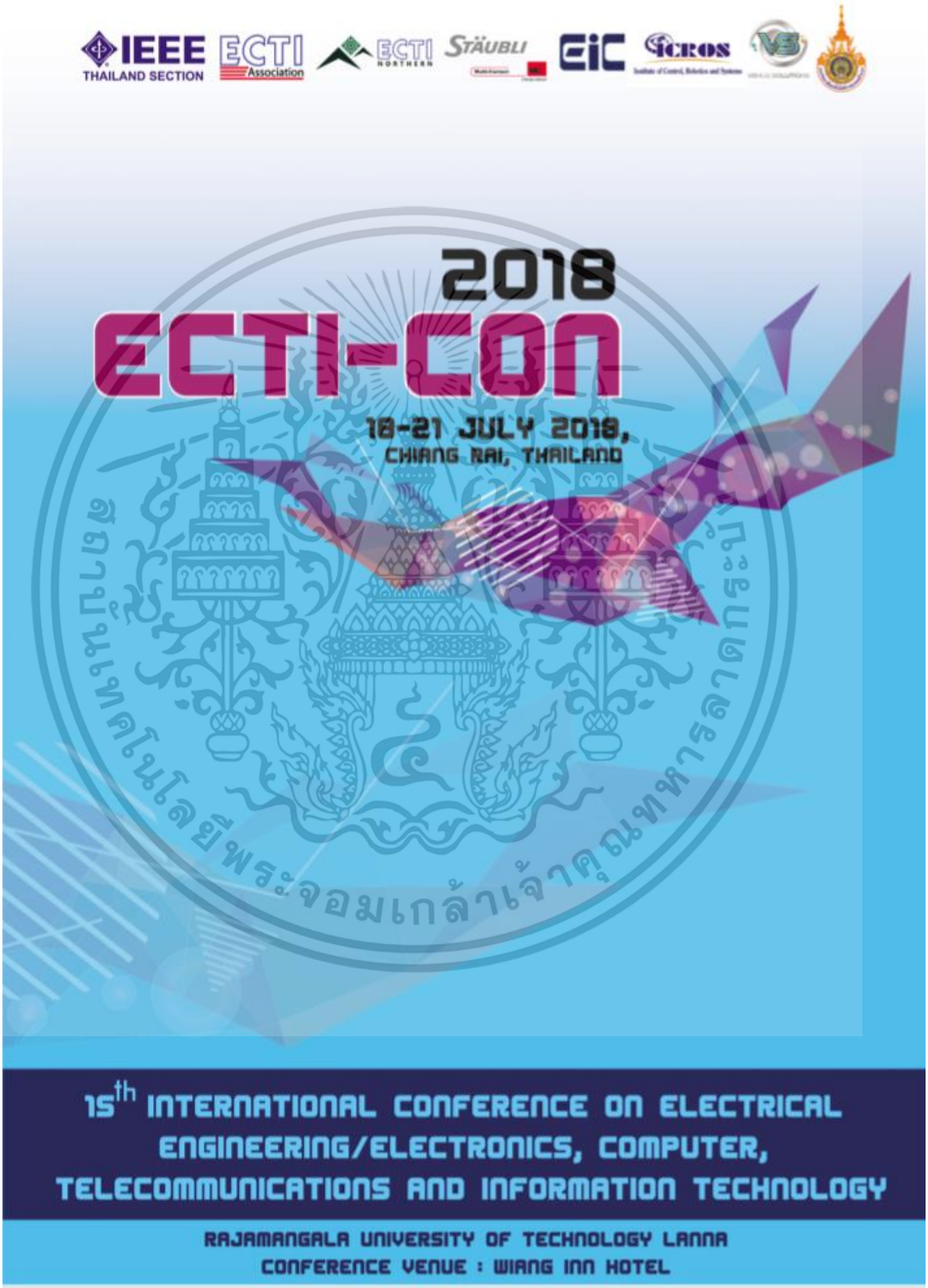
```
            [Position_max_ngram1_Row,~] = size(Position_max_ngram1);
            for a=1:Position_max_ngram1_Row
                Position_max_ngram1(a,2) =
table2array(ngram1(Position_max_ngram1(a,1),2));  % <---
            end
            [~,Position_ngram] = max(Position_max_ngram1(:,2));
            if Max_Check_Tem == ngram1_Column
                for k = ngram1_Column:-1 : 1
                    if Tem{1,k} == ' '
                        Tem = Tem(:,1:k-1);
                    end
                end
                [~,SizeTem] = size(Tem);
                if Check_Upper{1,1} == 1
                    Tem{1,1} = upper(Tem{1,1});
                    if Check_Upper_All > (SizeTem/3) && SizeTem >= 3
                        Tem = upper(Tem);
                    end
                    Char(1,(count+1):(count+SizeTem)) = Tem(1,1:SizeTem);
                else
                    Char(1,(count+1):(count+SizeTem)) = lower(Tem(1,1:SizeTem));
                end
                Tem = cell(0);
            else
                if Check_Upper{1,1} == 1
                    Tem = W_Check_ngram1(Position_max_ngram1(Position_ngram,1),:);
                    Tem{1,1} = upper(Tem{1,1});
                    for k = ngram1_Column:-1 : 1
                        if Tem{1,k} == ' '
                            Tem = Tem(:,1:k-1);
                        end
                    end
                    [~,SizeTem] = size(Tem);
```

```
                if Check_Upper_All > (SizeTem/3)
                    Tem = upper(Tem);
                end
                Char(1,(count+1):(count+SizeTem)) = Tem(1,1:SizeTem);
            else
                Tem = W_Check_ngram1(Position_max_ngram1(Position_ngram,1),:);
                for k = ngram1_Column:-1 : 1
                    if Tem{1,k} == ' '
                        Tem = Tem(:,1:k-1);
                    end
                end
                [~,SizeTem] = size(Tem);
                Tem = lower(Tem);
                Char(1,(count+1):(count+SizeTem)) = Tem(1,1:SizeTem);
            end
            clear Tem_Column Check_Tem_c Check_Upper Max_Check_Tem
Position_max_ngram1 Position_max_ngram1_Row Position_ngram1
            Check_Tem = zeros(ngram1_Row,ngram1_Column);
            Tem = cell(0);
        end

        else
            Tem = cell(0);
        end
        count = i;
    else
        Tem(1,i-count) = Char(1,i);
    end
end
Result = num2str(cell2mat(Char(1,:)));
end
```

ภาคผนวก ค

งานประชุมวิชาการ ECTI-CON 2018 International Conference



2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology

An Implementation of Maximum Likelihood Decoder for Error-File Recovery on Cloud Storage System

Pakpoom Sangprasittichok
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand
pakpoom.sangp@gmail.com

Nattakan Puttarak
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand
nattakan.pu@kmitl.ac.th

***Abstract*—Cloud storage system is a distributed storage system which stores data in many disks on the Internet. In order to achieve more reliable data and improve performance of the storage system, an erasure code is invented and applied to cloud storage system wherein the user's data will be stored online through the Internet. This paper applies the complete graph of ring (CGR) codes [7], which is a maximum distance separable (MDS) code, and implements in cloud storage system. Moreover, the maximum likelihood decoder (MLD) is also applied and implemented on the receiver to detect and correct errors due to noisy channel. In this paper, the grey-scale images/files are randomly stored on distributed cloud storage, then sent via communications channel, and finally read by receivers. The result shows that the receiver operated with the MLD outperforms by giving the lower BER.**

Keywords—cloud storage system; erasure code; maximum likelihood decoder (MLD)

I. INTRODUCTION

Huge and a large amount of data is stored and transferred through network every day. To protect the data loss during transmission and storage, in the past users might have to duplicate data and store in backup hard disks. However, in the present to store data in only one disk might not handle such growing and increasing data. So, a distributed storage is proposed to combine different disks such as RAID system shown in Fig.1. In RAID0, data will be stored in many disks without any redundancy data. RAID1 uses a duplicate disk as the backup to store the same data as the original data. In the other hand, RAID5 uses a group of disks to store data and also provides parity data stored in each disk, so this system can protect data if one disk is failure. RAID6 is similar to RAID5 except it can protect two disk failure in the same time without data loss [1].

To protect data loss in distributed data storage, there is a trade-off between storage overhead and fault tolerance. Erasure code is one efficient technique applied in this system. The maximum distance separable (MDS) code, an optimal erasure code which achieves a singleton bound property, is preferable used for fault tolerance such as Reed-Solomon code [1] and EVENODD code [3], or the complete graph of ring (CGR) code [7].

However, distributed storage system is suitable in a large company. Whereas a small company, which does not use storage too much but still need security, is not suitable. Service provider provides cloud storage system for the Internet users and small companies. Erasure code is also applied for data protection in cloud storage system (e.g. Microsoft Azure [2], Hadoop Distributed File System (HDFS)), in order to protect data by providing the fault tolerance capability.

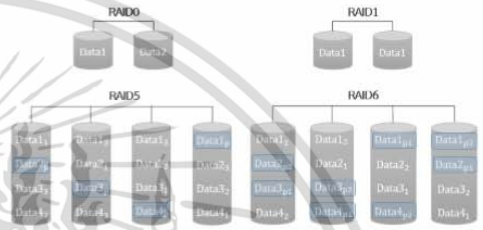


Fig. 1. Diagram of RAID0, RAID1, RAID5, RAID6

In communications system, a maximum likelihood decoder (MLD) is a decoding method by selecting codeword with the maximum possible probability. In the other word, it is the least probability of error data. There are many research applied this method in various applications and implementations. For example; in [4] MLD is used to detect neonatal clonic seizures using video image processing. This research detects the clonic seizures, which cause periodic movement of parts of body (e.g. hands, legs), to early diagnose, but there are still in some case that is not able to recognize all clonic seizures using MLD algorithm. In [5], researchers use MLD to classify quantification of aerial image. But, in case of where the aerial image has very similar pixels, the classification may not be as effective as it should be. Moreover, MLD with cross-correlation is applied to reduce latency on NB-IoT, in order to save more energy. The disadvantage is that it is more complex in implementation than the one with auto-correlation [6]. As mentioned above, MLD is a simple and useful method with many fruitful benefits. It can be applied with various implementations of communications system.

This research applies and implements the MLD on the cloud storage to detect and correct errors due to system transmission. When user wants to download data from cloud storage system, data will be transmitted via a noisy channel so it may cause some data loss. To provide the reliability and efficiency of data transmission without adding any redundancy

978-1-5386-3555-1/18/$31.00 ©2018 IEEE

or additional function on transmitters, we propose the MLD algorithm applied on a receiver to reduce bit error rate (BER) and improve the system performance.

The organization of this paper is as follows. Section II introduces all related backgrounds such as cloud storage, MLD, and CGR codes. In Section III, our algorithm and design framework are shown and described, Section IV gives an experimental results, and finally in section V, the conclusion of this research is given.

II. BACKGROUND

A. Cloud Storage System Overview

To simply describe algorithm which is applied with cloud storage system, we present diagram of transmitter that can upload files to store into cloud storage, and diagram of receiver that can download files from cloud storage shown in Fig.2.

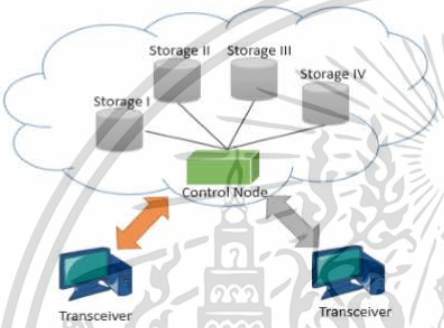


Fig. 2. Diagram of basic cloud storage system

When users want to upload or store files into cloud storage system, files will be transmitted to a control node and then a control node will distribute them to a group of storages. To read or download files, a control node also retrieves such files from all storages and send to receivers. Since, there are many data loss in cloud storage system due to disk failure, erasure codes will be applied for data loss protection. All encoding and decoding process are always implemented on the control node. So, to download/read files, control node will decode all codewords into original files (or data) and then send to receiver.

The communications model used in this paper is separated into 2 parts shown in Fig.3. The first part is a cloud storage system which is a distributed storage, and encoded data by CGR code [7] will be stored into different storages. The second part is a communications systems. Stored data from cloud storage system is sent to receiver via noisy channel. There might be some error data arrived at receiver and degraded a system performance. To reduce errors, the MLD algorithm is implemented into a receiver to correct some bit errors in order to provide better performance.

From Fig.3, an original data (x) is encoded by CGR code before all codewords will be stored into cloud storage system. Before sending codewords through a channel, they will be decoded. The data sent to the receiver via a noisy channel is affected by the additive white Gaussian noise (AWGN) (n). So, all received data (y) might include some undesirable errors. In part of a receiver, MLD algorithm is implemented to detect and correct errors. The preferable data required at a receiver is the same data as we stored in the system at first.

Fig. 3. Block diagram of cloud stotage system

B. Complete Graph of Ring Code

The complete graph of ring (CGR) code [7] is one of a maximum distance separable (MDS) codes constructed based on the structure of graphs. It can be represented as (K_{v1}, C_{v2}), where v_1 is a number of ring-graph and v_2 is a number of all edges of graph, and $v_2 = v_1 + 3$.

The (K_2, C_5) can be written in term of an array code as shown in Fig.4. Each column represents as a number of storage and each row represents as a symbol of data or redundancy. The (K_2, C_5) contains 2 disks of data and 3 disks of redundancy (totally 5 disks). This code is capable of recovering up to 3 simultaneous disk failure, which is equal to the number of memory used to store the redundancy. In addition, this code can be easily set in the form of tanner graph (or bipartite graph), which is the same format as a low-density parity-check (LDPC) code [8].

0	1	2	3	4
6	7	8	9	5
$2 \ominus 3$	$3 \oplus 4$	$4 \oplus 0$	$0 \oplus 1$	$1 \oplus 2$
$7 \ominus 8$	$8 \oplus 9$	$9 \oplus 5$	$5 \oplus 6$	$6 \oplus 7$
$4 \ominus 9$	$0 \oplus 5$	$1 \oplus 6$	$2 \oplus 7$	$3 \oplus 8$

Fig. 4. Array Code of CGR (K_2, C_5)

C. Maximum Likelihood Decoder

The maximum likelihood decoder (MLD), which is in the memoryless channel, is a method of estimating maximum probability. If X is a discrete random variable with $X = \{x_1, x_2, x_3, ..., x_n, \}$, where n is the total event and μ is the parameter to be estimated. Likelihood function is defined as follows.

$$P(x_1, x_2, ..., x_n; \mu) = \prod_{i=1}^{n} f(x_i; \mu), \quad (1)$$

where $P(X)$ is a joint probability mass function.

When considering the data in binary symmetric channel (BSC), where $x_i = \{0, 1\}$. It can be compared to the Bernoulli random variable as shown in Eq. (2).

$$P(x_1, x_2, ..., x_n; \mu) = \prod_{i=1}^{n} f(x_i; \mu) = \prod_{i=1}^{n} \mu^{x_i} (1-\mu)^{1-x_i}, \quad (2)$$

where Bernoulli random variable is shown below.

$$P(x = 1) = \mu, P(x = 0) = 1 - \mu, \text{ when } 0 < \mu < 1.$$

After that, we find the maximum of Eq. (2) by taking log.

$$\underset{\mu}{\operatorname{argmax}} \log P(X;\mu) = \underset{\mu}{\operatorname{argmax}} \sum_{i=1}^{n} [x_i \log\mu + (1 - x_i)\log(1 - \mu)] \quad (3)$$

We find the maximum of Eq. (3) by letting its first derivative equal zero, $\left(\frac{\partial}{\partial u} \log P(X;\mu) = 0\right)$, which is

$$\hat{\mu}_{Max} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (4)$$

III. Algorithm and Designed Framework

A. Our Algorithm

We define the probability of the 8 bits around bit C as shown in Fig.5, where $x_i = \{0,1\}$

x_1	x_2	x_3
x_4	C	x_5
x_6	x_7	x_8

Fig. 5. Model of surrounding 8 bits

The probability of C whether it is 0 or 1 can be designed as

$$\hat{\mu}_{C=1} = \frac{1}{8} \sum_{i=1}^{8} x_i \quad (5)$$

$$\hat{\mu}_{C=0} = 1 - \hat{\mu}_{C=1} \quad (6)$$

The threshold (T_h) is the value compared with $\hat{\mu}_C$. We consider bit flipping when $\hat{\mu}_{C=1}$ or $\hat{\mu}_{C=0}$ from Eq. (5), and Eq. (6) compared with T_h of MLD. We avoid the value of $T_h \le \frac{1}{2}$ by intuition since it is randomly selected value of C. So, in this work we set the value of T_h as $T_h > 1/2$, $T_h \ge 3/4$ and $T_h > 3/4$. If $T_h > 1/2$, it means that the probability of bit C $\hat{\mu}_{C=1}$ or $\hat{\mu}_{C=0} > 1/2$ and bit C will be flipped, otherwise we keep C as the original received bit. The algorithm is shown in Algorithm1.

TABLE I. EXPERIMENTAL SETUP

Type of images	*Number of Disk Failures*	*Threshold of MLD (T_h)*
Random binary bits images	0-3	$>1/2, \ge 3/4, > 3/4$
B/W images with light borders (Type 1)	0-3	$>1/2, \ge 3/4, > 3/4$
B/W images with thick borders (Type 2)	0-3	$>1/2, \ge 3/4, > 3/4$
B/W image (Type1+2)	0-3	$>1/2, \ge 3/4, > 3/4$

B. Designed Framework

We select 3 different sets of data which are random binary bits with size 600x600 bits and black/white bitmap images with light/thick borders size 600x600 bits for 50 images each. In this simulation scenario, we apply the CGR (K_2, C_5) code (which is a MDS array code that can recovery up to 3 simultaneous disk failure) to cloud storage system via the AWGN channel to the MLD receiver. The MLD can select threshold probability, $T_h > 1/2$, $T_h \ge 3/4$ or $T_h > 3/4$, by using 2 different sets of data in Table I.

IV. Results

In a situation when the storages simultaneously fail up to 0-3 disks from 5 disks, the MLD receiver with the threshold probability $T_h > 1/2$, $T_h \ge 3/4$, and $T_h > 3/4$ to make a decision for bit flipping will detect and correct errors.

Fig.6 shows the bit error rate (BER) of MLD receiver with various threshold T_h in each type of stored files/images. The results show that the MLD algorithm gives good performance with image data because any bit of image data has a relation with its neighbors. Consider the threshold, we can notice that $T_h > 1/2$ achieves lower BER compared with the system without MLD, when SNR is between 0-8 dB, $T_h \ge 3/4$ gives good performance, when SNR is between 8-12 dB, $T_h > 3/4$ gives lower BER, when SNR is higher than 12 dB. However, when the SNR is higher than 18 dB, the system with MLD will give worse performance, since at the more reliable channel many received bits are already good bits and need not to be corrected, but our algorithm might make a wrong decision. So, we can choose the value of T_h based on the quality of the channel to achieve the best efficiency. Fig.6 (d) shows that the receiver without MLD algorithm needs 16 dB of SNR, while the one with MLD ($T_h \ge 3/4$, $T_h > 3/4$) needs only 12 dB of SNR to achieve BER at 10^{-3}. The different threshold levels give different error-correction capability in a noisy channel based on the different SNR.

Algorithm : 1 Maximum likelihood decoder for binary image

Step 1 : Identify size of image Matrix P : n, m
: Identify surrounding bit of $C_{i,j}$: X
Step2 : Start correct bit of $C_{i,j}$ where $0 < i \le n$ and $0 < j \le m$
for $0 < i \le n$
for $0 < j \le m$
switch i, j **do**
case I : $i = 1$ (bit of C is at the first row)
case I - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i+1,j}\ P_{i+1,j+1}]$
case I - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i+1,j-1}\ P_{i+1,j}]$
case I - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i+1,j-1}\ P_{i+1,j}\ P_{i+1,j+1}]$
case II : $i = n$ (bit of C is at the last row)
case II - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i-1,j}\ P_{i-1,j+1}]$
case II - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i-1,j-1}\ P_{i-1,j}]$
case II - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i-1,j+1}]$
case III : $1 < i < n$ (otherwise)
case III - A : $j = 1$ **step** : $X = [P_{i,j+1}\ P_{i-1,j}\ P_{i-1,j+1}\ P_{i+1,j}\ P_{i+1,j+1}]$
case III - B : $j = m$ **step** : $X = [P_{i,j-1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i+1,j-1}\ P_{i+1,j}]$
case III - C : $1 < j < m$ **step** : $X = [P_{i,j-1}\ P_{i,j+1}\ P_{i-1,j-1}\ P_{i-1,j}\ P_{i-1,j+1}\ P_{i+1,j-1}\ P_{i+1,j}\ P_{i+1,j+1}]$
end switch
$\mu_{C=1} = \frac{1}{r} \sum_{k=1}^{r} X[k]$, for any $r \in X$
$\mu_{C=0} = 1 - \mu_{C=1}$
$C_{i,j} \leftarrow \{1 \mid \text{compare } \mu_{C=1} \text{ with } T_h, 0 \mid \text{compare } \mu_{C=0} \text{ with } T_h\}$
end for
end for

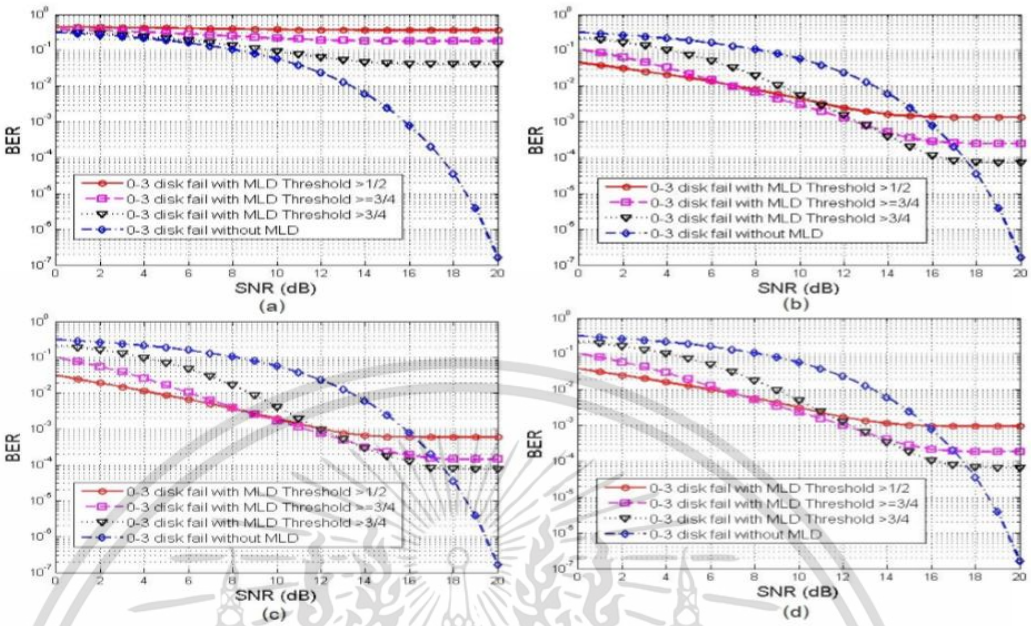


Fig. 6. Bit error rate of MLD receiver with various threshold T_h in each type of stored files/images: (a) random binary bits image (b) B/W images with light borders (Type 1) (c) B/W images with thick borders (Type 2) (d) B/W image (Type1+2)

V. Conclusion

This paper proposes an implementation of MLD to recover data errors on cloud storage system. The MLD algorithm is applied in a receiver, so there is no redundancy adding in the original data. Our implementation uses the binary bitmap image as experimental data stored in the cloud storage, and the receiver with MLD, which the threshold probability are $T_h > 1/2, T_h \geq 3/4$ and $T_h > 3/4$, can detect and correct errors. Each threshold level gives different error-correction capability depended on the different SNR. In the low level of SNR (0-8 dB), the suitable threshold is $T_h > 1/2$, then the threshold $T_h \geq 3/4$ gives a better BER at the higher level of SNR (8-12 dB). The BER of $T_h > 3/4$ is better than the one of $T_h \geq 3/4$ when the SNR > 14 dB. However, this algorithm is not preferable when SNR > 18 dB, since flipped bits may give more errors.

VI. Future Work

All data applied to the MLD algorithm should be black and white image, so the system has lower complexity in decoding process than the color image or non-binary image. To improve and apply this algorithm to correct errors in non-binary images is still the open and challenge problem.

References

[1] J. S. Plank, "A Tutorial on Reed-Solomon Coding for FaultTolerance in RAID-like Systems," Software Practice and Experience, 27(9), pp. 995-1012, Sept. 1997.

[2] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav,J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. Fahim ul Haq, M. Ikram ul Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, and L. Rigas. Windows Azure storage: A highly available cloud storage service with strong consistency. In Symposium on Operating Systems Principles, 2011.

[3] M. Blaum, J. Brady, J. Bruck, J. Menon. EVENODD: An efficient scheme for tolerating double disk failures in RAID architecture, 1995.

[4] L,Cattani, G.Ntonfo, F.Lofino, G.Ferrari, R.Raheli, and F.Pisani. Maximum-Likelihood Detection of Neonatal Clonic Seizures by Video Image Processing, 2014.

[5] S.Wanarse, Tejas.G, S.Patankar,and J.Kulkarni. Class Quantification of Aerial Images using Maximum Likelihood Estimation, 2014.

[6] H.Kroll, M.Korb, B.Weber, S.Willi, and Q.Huang. Maximum-Likelihood Detection for Energy-Efficient Timing Acquisition in NB-IoT, 2017.

[7] N. Puttarak, P. Kaewprapha, B. Chong, and J. Li. A New Class of MDS Erasure Codes Based on Graphs, 2009.

[8] R. G. Gallager, Low-density parity-check codes, Cambridge, MA: MIT Press, 1963.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายภาคภูมิ แสงประสิทธิโชค
วัน เดือน ปีเกิด	1 เมษายน 2535
ที่อยู่	429-430 หมู่5 ต.รางหวาย อ.พนมทวน จ.กาญจนบุรี 71170
ประวัติการศึกษา	2557 วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม
	สถาบันเทคโนโลยีพระจอมเกล้าเข้าคุณทหารลาดกระบัง
ความชำนาญเฉพาะด้าน	1.) ระบบโทรคมนาคม
	2.) ระบบการสื่อสารไร้สาย 2G, 3G และ 4G
	3.) ระบบเก็บข้อมูล
ผลงานทางวิชาการ	“An Implementation of Maximum Likelihood Decoder for Error-file Recovery on Cloud Storage System” ในงานประชุมวิชาการ ECTI-CON 2018 International Conference, pp. 74-77, July 2018.
ประสบการณ์การทำงาน	
พ.ศ.2557 – พ.ศ.2562	ตำแหน่ง Data Core Mobile Engineer
	บริษัท ทรูคอร์ปอเรชั่น
พ.ศ.2562 - ปัจจุบัน	ตำแหน่ง Cloud Core Network Engineer
	บริษัท หัวเว่ยเทคโนโลยี่ (ประเทศไทย)