



## ระบบการคืนกุญแจที่ปลอดภัยและผ่านการรับรอง

**Authenticated secure key recovery system**

นายสุวิทย์ สาโยส รหัส 64366751

นายมินทร์บдинทร์ ออนุเคราะห์ รหัส 64367499

โครงสร้างปริญญาในพนธน์เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตร์  
บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาฯวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2567



## ใบรับรองปริญญาบัณฑิต

ชื่อหัวข้อโครงการ

ระบบการถ่ายทอดความรู้ที่ปลอดภัยและผ่านการรับรอง

ผู้ดำเนินงานโครงการ

นายสุวิทย์ สายโสด

รหัส 64366751

นายมินทร์บดินทร์ อนุเคราะห์

รหัส 64367499

ที่ปรึกษาโครงการ

ดร. สุรเดช จิตประไพกุลศาลา

สาขาวิชา

วิศวกรรมคอมพิวเตอร์

ภาควิชา

วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา

2567

คณะกรรมการค่าสตัน มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญาบัณฑิตนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

ที่ปรึกษาโครงการ

(ดร. สุรเดช จิตประไพกุลศาลา)

กรรมการ

(รศ.ดร. พงศ์พันธ์ กิจสน�โยธิน)

กรรมการ

(อาจารย์ภาณุพงษ์ สอนคุณ)

ชื่อหัวข้อโครงการ	ระบบการกู้คืนกุญแจที่ปลอดภัยและผ่านการรับรอง	
ผู้ดำเนินงานโครงการ	นายสุวิทย์ สายโถ	รหัส 64366751
	นายมนทร์บดินทร์ อนุเคราะห์	รหัส 64367499
ที่ปรึกษาโครงการ	ดร. สุรเดช จิตประ ไพบูลศาลา	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์	
ปีการศึกษา	2567	

### บทคัดย่อ

โครงการนี้มีจุดมุ่งหมายในการพัฒนาระบบการกู้คืนกุญแจที่มีความปลอดภัยสูงและได้รับการรับรอง ความถูกต้อง ซึ่งมีความสำคัญอย่างมากในระบบที่ต้องมีการรับส่งข้อมูลลับหรือการสื่อสารที่ต้องการความปลอดภัย โครงการนี้มุ่งเน้นในการต่อยอดจาก SFM-KRS (Secure Framework for Multi-Party Key Recovery System) โดยเพิ่มการตรวจสอบลิทเทิร์แบบ PKCE-like challenge ทั้งในขั้นตอนการขอ กู้คืนกุญแจและการส่งข้อมูลไปยังตัวแทนต่าง ๆ เพื่อให้มั่นใจในความปลอดภัยผ่านการยืนยันตัวตน หลายขั้นตอน (multi-way authentication) ซึ่งเป็นการพิสูจน์ว่าการยืนยันตัวตนแบบ certificated authentication และ token-based authentication ระบบที่พัฒนานี้จะช่วยลดความเสี่ยงในการโจมตีจากผู้ไม่ประสงค์ดีที่พยายามเข้าถึงกุญแจโดยไม่ได้รับอนุญาต อีกทั้งยังช่วยเพิ่มความมั่นใจในกระบวนการกู้คืนกุญแจที่มีความปลอดภัยสูงยิ่งขึ้น

<b>Project title</b>	Authenticated secure key recovery system	
<b>Name</b>	Mr. Suwit Saiso	ID. 64366751
	Mr. Minbodin Anukroh	ID. 64367499
<b>Project advisor</b>	Suradet Jitprapaikulsarn, Ph.D	
<b>Major</b>	Computer Engineering	
<b>Department</b>	Electric and Computer Engineering	
<b>Academic year</b>	2567	

---

### Abstract

This project aims to develop a highly secure and authenticated key recovery system, which is crucial for systems that require secure communication and the transfer of sensitive information. The project builds upon the Secure Framework for Multi-Party Key Recovery System (SFM-KRS) by introducing a PKCE-like challenge mechanism. This challenge is applied both during the key recovery request process and when sending information to various agents, ensuring security through multi-way authentication, which integrates certificate-based authentication and token-based authentication. The developed system is designed to mitigate the risks posed by unauthorized entities attempting to access recovery keys, thereby increasing the security and trustworthiness of the key recovery process.

## กิตติกรรมประกาศ

ผู้ดำเนินโครงการขอรับขอบพระคุณ ดร.สุรเดช จิตประไฟต์ คณบดี คณะมนุษยศาสตร์ มหาวิทยาลัยนเรศวร สำเร็จสมบูรณ์ไปได้ด้วยดี ขอขอบพระคุณ อาจารย์ภาณุพงศ์ สอนคอม และ รศ.ดร.พงศ์พันธ์ กิจสนาน้อยิน อาจารย์ประจำภาควิชา ไฟฟ้า และคุณพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร ซึ่งเป็นผู้ทรงคุณวุฒิที่ให้เกียรติเป็นกรรมการการ สอบโครงการในครั้งนี้ สุดท้ายนี้ผู้ดำเนินโครงการหวังเป็นอย่างยิ่งว่า ปริญญานิพนธ์ ฉบับนี้จะมีประโยชน์ในการนำไปใช้ต่อยอดและ พัฒนาแก่ผู้สนใจในภายภาคหน้า จึงขอขอบคุณ ปริญญานิพนธ์ ให้ทุกท่านเพื่อเป็นประโยชน์ต่อผู้ที่สนใจทุกท่านที่ สนใจในอนาคตต่อไป

คณะผู้ดำเนินโครงการวิศวกรรม

นายสุวิทย์ สายโสด

นายมินทร์บดินทร์ อนุเคราะห์

ตุลาคม 2567

## สารบัญ

	หน้า
ใบรับรองปริญญาอิเล็กทรอนิกส์	ก
บทคัดย่อภาษาไทย	ข
บทคัดย่อภาษาอังกฤษ	ค
กิตติกรรมประกาศ	ง
สารบัญ	จ
สารบัญตาราง	ฉ
สารบัญรูป	ฉ
 บทที่ 1 บทนำ	 1
1.1 ความเป็นมาและความสำคัญของโครงการ	1
1.2 วัตถุประสงค์ของโครงการ	1
1.3 ประโยชน์ที่คาดว่าจะได้รับ	2
1.4 ขอบเขตการทำงาน	2
1.5 ขั้นตอนการดำเนินงาน	3
1.6 แผนการดำเนินงาน	4
1.7 รายละเอียดงบประมาณตลอดโครงการ	4
บทที่ 2 หลักการและทฤษฎี	5
2.1 ระบบการคืนกุญแจ (Key Recovery System - KRS) และ SFM-KRS	5
2.2 ระบบคืนกุญแจที่มีการยืนยันตัวตนแบบหลายปัจจัย (Authenticated Secure Key Recovery System)	6
2.3 การยืนยันตัวตนหลายขั้นตอน (Multi-Way Authentication)	9
2.4 ระบบ PKI และการยืนยันตัวตนด้วยใบรับรอง (Certificated Authentication)	12
2.5 การใช้ Public และ Private Key	12
2.6 PKCE (Proof Key for Code Exchange)	13
2.7 PKCE-like Challenge	14

## สารบัญ (ต่อ)

	หน้า
2.8 การกระจายกุญแจและการรวมกุญแจ (Key Splitting and Reconstruction)	16
2.9 One-way encryption	18
2.10 คุณสมบัติที่สำคัญของฟังก์ชันแฮช	19
2.11 การเข้ารหัสแบบสมมาตร (Symmetric Encryption)	20
2.12 การเข้ารหัสแบบ非对称加密 (Asymmetric Encryption)	21
2.13 ภาษา Python	23
2.14 Docker	23
2.15 ไลบรารี Cryptography ใน Python	24
2.16 ไลบรารี Flask ใน Python	25
2.17 ไลบรารี Requests ใน Python	26
2.18 ไลบรารี Socket ใน Python	26
<b>บทที่ 3 วิธีดำเนินโครงการ</b>	<b>28</b>
3.1 การศึกษาและวิเคราะห์ข้อมูลเกี่ยวกับโครงการ	28
3.2 ภาพรวมโครงการ	29
3.3 การออกแบบระบบ	31
3.4 การสร้างจำลองโดยใช้ Docker	33
3.5 การสร้าง PKI Structure	35
3.6 การทำงานของ Sender	37
3.7 การทำงานของ Receiver/Requester	44
3.8 การทำงานของ KRC	48
3.9 การทำงานของ KRA	53
3.10 สมมติฐานของการทำงานของระบบ	54
3.11 การพัฒนาและทดสอบระบบ	56
3.12 มาตรการความปลอดภัยในระบบ	59
3.13 การทดสอบการถูกสืบกุญแจ กรณีอาจน์ล็อก	60
3.14 การแสดงผล	62

## สารบัญ (ต่อ)

	หน้า
<b>บทที่ 4 ผลการทดลองและวิเคราะห์</b>	64
4.1 ผลการทดสอบการทำงานของโปรแกรม	64
4.2 ผลการประเมินการทดสอบ	67
<b>บทที่ 5 บทสรุปและข้อเสนอแนะ</b>	78
5.1 สรุปผลการทดลอง	78
5.2 สรุปผลตามวัตถุประสงค์ที่คาดว่าจะได้รับ	78
5.3 ปัญหาและการปรับปรุง	78
5.4 ความรู้และเครื่องมือที่จำเป็นต่อการพัฒนาต่อ	79
5.5 ข้อเสนอแนะ	79
เอกสารอ้างอิง	80
ประวัติผู้ดำเนินโครงการ	83

## สารบัญตาราง

ตารางที่	หน้า
1.1 แผนการดำเนินงาน	4
2.1 ข้อได้เปรียบทอง AS-KRS เมื่อเทียบกับ SFM-KRS	7
2.2 ตารางเปรียบเทียบข้อดีและข้อเสียของรูปแบบการตรวจสอบสิทธิ์ที่นิยมใช้ในปัจจุบัน	10
3.1 รายละเอียดของ API สำหรับรับข้อมูล Plaintext	37
3.2 รายละเอียดของ API สำหรับทดสอบระบบ	45

## สารบัญรูป

รูปที่		หน้า
2.1	ระบบ SFM-KRS แบบเดิม	6
2.2	การพัฒนา SFM-KRS ด้วยการเพิ่ม Authentication ในระหว่างการสื่อสาร	9
2.3	PKI	13
2.4	OAuth 2.0 with PKCE	14
2.5	Authorization with PKCE-like Challenge between Receiver and KRC	15
2.6	Authorization with PKCE-like Challenge between KRC and KRAs	16
2.7	XOR-based key splitting	18
2.8	One-way encryption	19
2.9	การเข้ารหัสแบบกุญแจสมมาตร(Symmetric Key Cryptography)	21
2.10	การเข้ารหัสแบบกุญแจ非对称(Symmetric Key Cryptography)	22
2.11	ภาษา Python	23
2.12	Docker	24
3.1	โครงสร้างของ Docker container	32
3.2	ภาพรวมการทำงานของระบบ AS-KRS	33
3.3	Docker Network Simulation	35
3.4	PKI Struture	37
3.5	การเตรียมค่าส่วนประกอบย่อยของ KRF	40
3.6	ขั้นตอนการทำงานของ Sender	43
3.7	ขั้นตอนการทำงานของ Receiver	47
3.8	การอุดรหัส KRF โดย KRC	49
3.9	การถือครองส่วนประกอบของกุญแจลับโดย KRA	50
3.10	การถือครองกุญแจลับโดย KRC	51
3.11	ขั้นตอนการทำงานของ KRC	52
3.12	ขั้นตอนการทำงานของ KRA	54
4.1	หน้าเริ่มต้นของ Sender	64
4.2	หน้าเริ่มต้นของ Receiver	65

## สารบัญรูป(ต่อ)

รูปที่	หน้า
4.3 หน้าเริ่มต้นของ KRC	65
4.4 ตัวอย่างหน้าเริ่มต้นของ KRA1 – KRA5	65
4.5 ตัวอย่างการแสดงผลหลังการส่งข้อความจาก Sender	66
4.6 ตัวอย่างการแสดงผลหลังการรับข้อความจาก Sender ของ Receiver	66
4.7 ตัวอย่างการแสดงผลหลังการรับข้อความโดยที่ session key corrupt	67
4.8 ตัวอย่างการแสดงผลกรณีหัสข้อความโดยที่ session key ชำรุดหรือสูญหาย	67
4.9 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจ และถอดรหัสข้อความสุดท้ายด้วยกุญแจใหม่ของ Receiver	68
4.10 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRC	68
4.11 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA1	69
4.12 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA2	69
4.13 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA3	70
4.14 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA4	70
4.15 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA5	71
4.16 ตัวอย่างการแสดงผลการสนทนากลังการทำการกู้คืนกุญแจของ Sender	71
4.17 ตัวอย่างการแสดงผลการสนทนากลังการทำการกู้คืนกุญแจของ Receiver	72
4.18 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจ และถอดรหัสข้อความสุดท้ายด้วยกุญแจ	72
4.19 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRC กรณีมีอเจนต์ล้ม	73
4.20 ตัวอย่างการแสดงผลการพยายามติดต่ออเจนต์ที่ล้ม(KRA1) ของ KRC กรณีมีอเจนต์ล้ม	73
4.21 ตัวอย่างการแสดงผลการล้มของ KRA1 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)	74
4.22 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA2 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)	74
4.23 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRA3 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)	74

## สารบัญรูป(ต่อ)

รูปที่		หน้า
4.24	ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ KRA4 กรณีมีอเจนต์ล้ม <sup>กู้คืนกุญแจของ KRA1 ล้ม</sup>	75
4.25	ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ KRA5 กรณีมีอเจนต์ล้ม <sup>กู้คืนกุญแจของ KRA1 ล้ม</sup>	75
4.26	ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ Receiver กรณีมีอเจนต์ล้ม <sup>ทุกตัว</sup>	76
4.27	ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ KRC กรณีมีอเจนต์ล้ม <sup>ทุกตัว</sup>	76
4.28	ตัวอย่างการแสดงผลการล้มของ KRA1 – KRA5 กรณีมีอเจนต์ล้ม <sup>ทุกตัว</sup>	77

## บทที่ 1

### บทนำ

#### 1.1 ที่มาและความสำคัญของโครงการ

ในยุคดิจิทัล การรักษาความปลอดภัยของข้อมูลมีความสำคัญอย่างยิ่ง โดยเฉพาะในการมีข้อมูลมีการเข้ารหัสเพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต อย่างไรก็ตาม ปัญหานั่นที่พบได้บ่อยคือการสูญหายของกุญแจเข้ารหัส (Encryption Key) ซึ่งอาจทำให้ไม่สามารถกู้คืนหรือเข้าถึงข้อมูลที่สำคัญได้

Key Recovery System (KRS) เป็นกลไกที่ใช้ในการกู้คืนกุญแจเข้ารหัสที่สูญหาย ซึ่งมีบทบาทสำคัญในหลายบริบท เช่น

- การกู้คืนกุญแจขององค์กรที่ต้องการปกป้องข้อมูลสำคัญ
- การทำธุกรรมออนไลน์ที่ต้องการความปลอดภัยสูง
- การกู้คืนระบบหลังจากการถูกโจมตีทางไซเบอร์ หรือภัยพิบัติ

อย่างไรก็ตาม ระบบกู้คืนกุญแจในปัจจุบันยังคงมีข้อจำกัดหลายประการ เช่น การขาดกลไกยืนยันตัวตนที่แข็งแกร่ง หรือมีช่องโหว่ที่อาจถูกผู้ไม่หวังดีโจมตีได้ โครงการนี้จึงมีเป้าหมายในการพัฒนาระบบ Key Recovery System ที่มีความปลอดภัยและมีประสิทธิภาพสูงขึ้น โดยใช้กระบวนการ การยืนยันตัวตนหลายชั้นตอน (Multi-Factor Authentication: MFA) และ PKCE-like challenge เพื่อเพิ่มความปลอดภัยในการยืนยันตัวตนของทุกฝ่ายที่เกี่ยวข้องในกระบวนการกู้คืนกุญแจ ซึ่งจะช่วยลดความเสี่ยงจากการถูกโจมตีและเพิ่มความน่าเชื่อถือของระบบ

#### 1.2 วัตถุประสงค์ของโครงการ

1.2.1 พัฒนาระบบกู้คืนกุญแจที่มีความปลอดภัยสูง โดยใช้การยืนยันตัวตนหลายชั้นตอน

1.2.2 เพิ่มกลไก PKCE-like challenge เพื่อป้องกันการปลอมแปลงคำร้องขอ กู้คืนกุญแจ

1.2.3 พัฒนาการยืนยันตัวตนด้วย certificate-based authentication และ token-based authentication เพื่อเพิ่มระดับความปลอดภัย

### 1.3 ผลที่คาดว่าจะได้รับ

- 1.3.1 ได้ระบบกู้คืนกุญแจที่มีความปลอดภัยสูงกว่าระบบเดิม ลดความเสี่ยงจากการโจมตี
- 1.3.2 ระบบสามารถใช้งานกลไก PKCE-like challenge เพื่อเพิ่มความน่าเชื่อถือในการยืนยันตัวตน
- 1.3.3 สามารถนำไปประยุกต์ใช้ในองค์กรที่ต้องการความปลอดภัยสูง เช่น สถาบันการเงิน หรือหน่วยงานภาครัฐ
- 1.3.4 มีประสิทธิภาพในการกู้คืนกุญแจโดยใช้เวลาอ้อยดลงเมื่อเทียบกับระบบเดิม

### 1.4 ขอบเขตการทำงาน

โครงการนี้ครอบคลุมถึงการพัฒนา ระบบการกู้คืนกุญแจที่ปลอดภัย (Authenticated Secure Key Recovery System) โดยมุ่งเน้นการออกแบบและพัฒนา มาตรการป้องกัน เพื่อรับรองการกู้คืนกุญแจในสภาพแวดล้อมที่มีหลายฝ่ายเกี่ยวข้อง ได้แก่ ผู้ส่ง (Sender), ผู้รับ (Receiver), ตัวแทนกู้คืนกุญแจ (Key Recovery Center - KRC), และผู้ช่วยกู้คืนกุญแจ (Key Recovery Agent - KRA)

ระบบถูกออกแบบให้มี กลไกการยืนยันตัวตนหลายขั้นตอน รวมถึง PKCE-like challenge เพื่อป้องกันการปลอมแปลงคำขอ กู้คืนกุญแจ และใช้ การเข้ารหัสแบบ RSA และ AES เพื่อรักษาความปลอดภัยของข้อมูลระหว่างการรับ-ส่งกุญแจ อย่างไรก็ตาม โครงการนี้มุ่งเน้นที่การออกแบบและพัฒนาระบวนการป้องกัน โดย ไม่มีการดำเนินการทดสอบ โจรตีจิริ เช่น Replay Attack, Man-in-the-Middle Attack (MITM), หรือ Brute Force Attack

ระบบจะถูกพัฒนาและทดสอบภายใน สภาพแวดล้อมจำลอง ซึ่งประกอบด้วย เครื่องข่ายจำลองของผู้ใช้งานหลายฝ่าย โดยใช้ Docker เพื่อจัดการคอนเทนเนอร์ของแต่ละองค์ประกอบ การจำลอง PKI structure โดยมีการให้แต่ละฝ่ายสร้างคู่กุญแจ public และ private key ของตนเองและแชร์ public key ของตนไว้ในสูญญากาศจำลองเพื่อให้แต่ละฝ่ายสามารถเข้าถึง public key ของแต่ละฝ่ายได้ และ Flask สำหรับการสื่อสารระหว่างบริการต่าง ๆ ระบบจะรองรับการ แสดง log การทำงาน ผ่านเว็บ GUI เพื่อให้ผู้ใช้สามารถติดตามกระบวนการกู้คืนกุญแจได้แบบเรียลไทม์ แต่จะไม่มีการแสดงผลลัพธ์จากการทดสอบความปลอดภัย เนื่องจากระบบนี้ยังไม่ได้ดำเนินการทดสอบการโจมตีจริง

ขอบเขตของโครงการจึงมุ่งเน้นไปที่

- การออกแบบและพัฒนา ระบบการกู้คืนกุญแจที่ปลอดภัย
- การเพิ่มมาตรการป้องกัน PKCE-like challenge และ การเข้ารหัส
- การทำงานร่วมกันของ Sender, Receiver, KRC และ KRA

- การพัฒนาเว็บ GUI สำหรับแสดง log กระบวนการภัยคุกคามแจ้ง
- ไม่มีการทดสอบการโจมตีความปลอดภัยจริง (เช่น MITM, Replay, Brute Force)

โครงการนี้มุ่งเน้นการพัฒนาเกลไกป้องกันสำหรับระบบภัยคุกคามแจ้งมากกว่าการทดสอบด้านความปลอดภัย โดยเป็นพื้นฐานที่สามารถนำไปพัฒนาเพิ่มเติมเพื่อรับการทดสอบความปลอดภัยในอนาคต

## 1.5 ขั้นตอนการดำเนินงาน

**1. วางแผนโครงการ** กำหนดวัตถุประสงค์และเป้าหมายของโครงการ เริ่มจากการกำหนดขอบเขตของโครงการอย่างชัดเจน โดยการวางแผนวัตถุประสงค์และเป้าหมายในการทำงาน เช่น เพื่อแก้ไขปัญหา เมื่อมีการกำหนดเป้าหมายแล้ว ควรวิเคราะห์ความเป็นไปได้ของโครงการและทรัพยากรที่ต้องใช้ในการดำเนินงาน

**2. วางแผน จัดทำอุปกรณ์และออกแบบการทำงาน** เกี่ยวข้องกับการจัดทำทรัพยากรต่างๆ เช่น อุปกรณ์และเทคโนโลยีที่จำเป็นสำหรับการดำเนินโครงการ และการออกแบบการทำงานหรือระบบต่างๆ ให้เหมาะสมกับเป้าหมายที่กำหนดไว้ จะต้องพิจารณาถึงความเหมาะสมของอุปกรณ์และวิธีการดำเนินงานเพื่อให้เกิดประสิทธิภาพสูงสุด

**3. ดำเนินการ** การดำเนินการตามแผนที่วางไว้จะเริ่มขึ้น การดำเนินการนี้ต้องปฏิบัติตามขั้นตอนที่กำหนดไว้อย่างรอบคอบและติดตามผลอย่างต่อเนื่อง รวมถึงตรวจสอบว่าแต่ละขั้นตอนได้ดำเนินการไปตามที่วางแผนไว้หรือไม่

**4. ทดสอบและปรับปรุง** ปรับปรุงและแก้ไขเพื่อให้ระบบหรือโครงการสามารถดำเนินงานได้อย่างสมบูรณ์ โดยอาจต้องกลับไปปรับปรุงส่วนต่างๆ ของโครงการตามที่พบข้อบกพร่อง และทำการทดสอบซ้ำจนกว่าจะแน่ใจว่าระบบทำงานได้ตามที่คาดหวัง

**5. สรุปผลการทดลองและจัดทำรูปเล่มปริญญานิพนธ์** สรุปผลลัพธ์ที่ได้รับจากการดำเนินงานทั้งหมดและการจัดทำรายงานหรือรูปเล่มปริญญานิพนธ์ ซึ่งจะประกอบด้วยการอธิบายถึงกระบวนการดำเนินงาน ผลลัพธ์ที่ได้ บทวิเคราะห์ข้อดีข้อเสีย รวมถึงข้อเสนอแนะสำหรับการปรับปรุงในอนาคต

## 1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

รายละเอียด	ปี 2567			ปี 2568		
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1. วางแผน โครงการ						
2. วางแผน จัดทำ อุปกรณ์และ ออกแบบ การทำงาน						
3. ดำเนินการ						
4. ทดสอบ และ ปรับปรุง						
5. สรุปผล การทดลอง และ จัดทำ รูปเล่ม บริษัทฯ นิพนธ์						

## 1.7 รายละเอียดงบประมาณตลอดโครงการ

1. ค่าเอกสาร 2,000 บาท

รวมเป็นเงิน 2,000 บาท (สองพันบาทถ้วน)

## บทที่ 2

### หลักการและทฤษฎีเบื้องต้น

ระบบการกู้คืนกุญแจที่ปลอดภัยและผ่านการรับรอง (**Authenticated Secure Key Recovery System - AS-KRS**) เป็นหัวใจสำคัญของการรักษาความปลอดภัยในระบบที่เกี่ยวข้องกับการเข้ารหัสและการสื่อสารข้อมูลที่ต้องการการปกป้อง โครงงานนี้พัฒนาต่อจาก SFM-KRS โดยเพิ่มกระบวนการยืนยันตัวตนหลายปัจจัย และใช้การตรวจสอบสิทธิ์แบบ PKCE-like Challenge เพื่อป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

#### 2.1 ระบบการกู้คืนกุญแจ (Key Recovery System - KRS) และ SFM-KRS

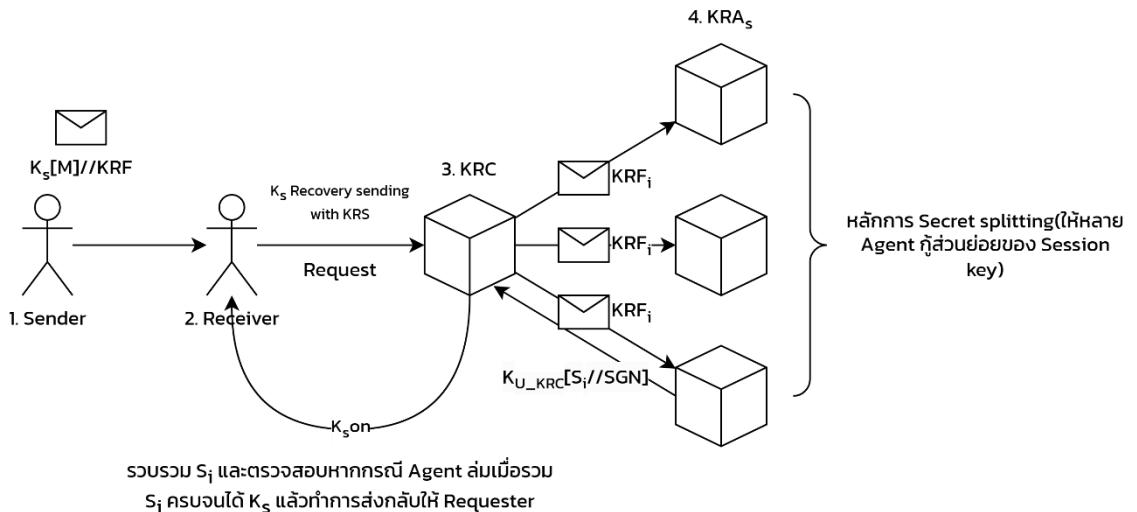
ระบบการกู้คืนกุญแจ (Key Recovery System - KRS) เป็นกระบวนการที่ช่วยให้ผู้ใช้หรือหน่วยงานที่มีสิทธิ์สามารถกู้คืนกุญแจเข้ารหัสได้ในกรณีที่กุญแจสูญหายหรือไม่สามารถเข้าถึงได้ กระบวนการนี้มีความสำคัญอย่างยิ่งในระบบที่ต้องการความปลอดภัยของข้อมูล โดยระบบต้องได้รับการออกแบบให้มีความปลอดภัยสูงเพื่อลดความเสี่ยงจากการถูกขโมยกุญแจหรือการเข้าถึงโดยไม่ได้รับอนุญาต

หนึ่งในระบบการกู้คืนกุญแจที่ได้รับการยอมรับคือ SFM-KRS (Secure and Flexible Multiple-Agent Key Recovery System) ซึ่งใช้แนวคิดของ Public Key Infrastructure (PKI) และ Key Recovery Agents (KRA) ในการกระจายกุญแจออกเป็นส่วนย่อยและจัดเก็บไว้กับตัวแทนหลายคนฝ่ายละบันที่ทำหน้าที่แยกกุญแจออกเป็นหลายส่วนและเก็บแยกกัน เพื่อให้ไม่สามารถกู้คืนกุญแจได้จากฝ่ายเดียว เมื่อต้องการกู้คืนกุญแจจะต้องมีหลายฝ่ายร่วมกันในการประกอบกุญแจให้สมบูรณ์ ซึ่งช่วยลดความเสี่ยงจากการถูกขโมยกุญแจทั้งหมด

อย่างไรก็ตาม SFM-KRS ยังมีข้อจำกัดที่สำคัญ ได้แก่

- ความเสี่ยงจากการโชมย Private Key หาก Private Key ของฝ่ายใดฝ่ายหนึ่งถูกขโมย อาจทำให้ระบบถูกบุกรุกได้
- ขาดการตรวจสอบสิทธิ์แบบหลายขั้นตอน ระบบใช้การตรวจสอบจาก PKI เพียงอย่างเดียว ซึ่งอาจไม่เพียงพอในการป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต
- เสี่ยงต่อการโจมตีแบบ Man-in-the-Middle (MITM) PKI เพียงอย่างเดียวอาจไม่สามารถป้องกัน MITM ได้หากไม่มีการตรวจสอบตัวตนเพิ่มเติม

ในโครงการนี้ SFM-KRS ถูกใช้เป็นโครงสร้างหลักของระบบ โดยมีการพัฒนาเพิ่มเติมเพื่อเพิ่มความปลอดภัยในการกู้คืนกุญแจ โดยเฉพาะการนำระบบตรวจสอบสิทธิ์ที่เข้มงวดขึ้น มาใช้ เพื่อลดความเสี่ยงจากการโจมตีของผู้ไม่หวังดีและเพิ่มความมั่นใจในการใช้งานระบบกู้คืนกุญแจอย่างปลอดภัย



รูปที่ 2.1 ระบบ SFM-KRS แบบเดิม

## 2.2 ระบบกู้คืนกุญแจที่มีการยืนยันตัวตนแบบหลายทางบัญชาย (Authenticated Secure Key Recovery System)

เพื่อแก้ไขข้อจำกัดของ SFM-KRS ระบบที่พัฒนาขึ้นมาใหม่คือ Authenticated Secure Key Recovery System (AS-KRS) ซึ่งเพิ่มความปลอดภัยในการกู้คืนกุญแจโดยใช้การยืนยันตัวตนแบบหลายขั้นตอน (Multi-Way Authentication) และ PKCE-like Challenge เพื่อลดความเสี่ยงจากการโจมตีและการเข้าถึงโดยไม่ได้รับอนุญาต

### 2.2.1 โครงสร้างของระบบ AS-KRS

ระบบ Authenticated Secure Key Recovery System (AS-KRS) ถูกพัฒนาขึ้นจากพื้นฐานของ SFM-KRS โดยเพิ่มความปลอดภัยด้วยการตรวจสอบสิทธิ์แบบหลายขั้นตอน เพื่อให้มั่นใจว่าผู้ร้องขอ กู้คืนกุญแจ เป็นบุคคลที่ได้รับอนุญาตจริง ๆ ระบบนี้ประกอบด้วยองค์ประกอบหลักดังต่อไปนี้:

- Key Recovery Center (KRC): ศูนย์กลางที่รับผิดชอบในการจัดการกระบวนการกู้คืนกุญแจ

- Key Recovery Agents (KRA): ตัวแทนที่ได้รับมอบหมายให้เก็บกุญแจย่อย (Key Recovery Fragments - KRF)
  - Authenticated User: ผู้ใช้งานที่ได้รับการยืนยันตัวตนหลายขั้นตอนก่อน
- ระบบนี้ออกแบบให้สามารถกู้คืนกุญแจต้องผ่านการยืนยันตัวตนหลายขั้นตอนก่อน ลดความเสี่ยงจากการโจมตีที่อาจเกิดขึ้นกับระบบกู้คืนกุญแจแบบเดิม

### 2.2.3 กระบวนการกู้คืนกุญแจใน AS-KRS

#### 2.2.3.1 การตรวจสอบสิทธิ์ด้วย PKCE-like Challenge

เมื่อผู้ใช้ร้องขอ กู้คืนกุญแจ ระบบจะส่ง Challenge Code ไปยังอุปกรณ์ของผู้ใช้ จากนั้นผู้ใช้ต้องตอบกลับด้วย Code Verifier เพื่อพิสูจน์ตัวตนของตนเอง ซึ่งช่วยลดความเสี่ยงจากการโจมตีแบบ Man-in-the-Middle (MITM) และการขโมย Private Key

#### 2.2.3.2 การร้องขอ กุญแจจาก KRA

- KRC จะร้องขอ กุญแจย่อย (KRF) จาก KRA หลายแห่ง
- KRA จะตรวจสอบสิทธิ์ของ KRC ก่อนส่งข้อมูลเพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต
  - การรวมกุญแจกลับคืน
  - KRC จะใช้ KRF ที่ได้รับจาก KRA มาประกอบเป็นกุญแจสมบูรณ์ เพื่อให้ผู้ใช้สามารถใช้งานได้

ตารางที่ 2.1 ข้อได้เปรียบของ AS-KRS เมื่อเทียบกับ SFM-KRS

คุณสมบัติ	SFM-KRS	AS-KRS(พัฒนาใหม่)
ใช้ PKI	✓	✓
มีการตรวจสอบสิทธิ์หลายขั้นตอน	X	✓
ป้องกัน MITM	X	✓
ป้องกันการขโมย Private Key	X	✓
ใช้ PKCE-like Challenge	X	✓

## 2.2.4 การปรับปรุงด้านความปลอดภัยของ AS-KRS

SFM-KRS แบบดั้งเดิมทำงานโดยเน้นการใช้ PKI (Public Key Infrastructure) ในการเข้ารหัสและการกู้คืนกุญแจ โดยมี KRA หลายตัวที่ช่วยเก็บกุญแจส่วนย่อย อย่างไรก็ตาม ข้อจำกัดหลักของระบบแบบเดิมคือ

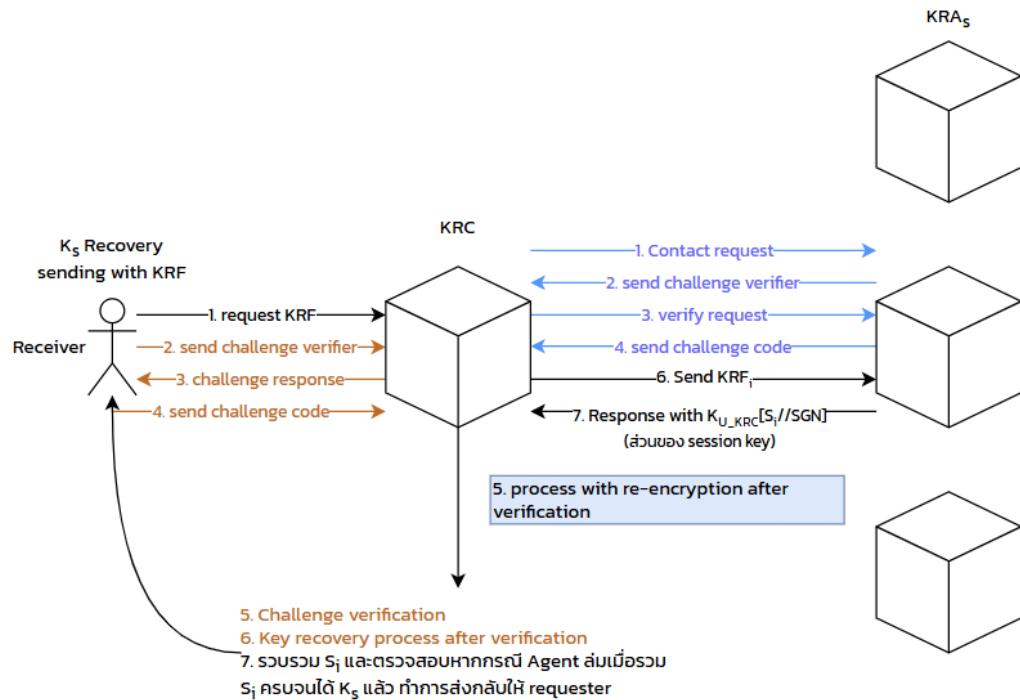
- การใช้งาน PKI เพียงอย่างเดียว: แม้ว่า PKI จะมีความปลอดภัยสูงในการเข้ารหัสข้อมูล แต่ยังมีช่องโหว่ที่อาจถูกโจมตีได้ เช่น Man-in-the-Middle (MITM) หรือการขโมย Private Key หาก Private Key ของฝ่ายหนึ่งถูกขโมย ระบบ PKI จะไม่สามารถป้องกันได้
- ขาดการยืนยันตัวตนอย่างละเอียด: ระบบ PKI แบบเดิมมีเพียงการตรวจสอบว่ากุญแจที่ใช้เป็นของผู้ส่ง/ผู้รับที่ถูกต้องเท่านั้น แต่ไม่มีการตรวจสอบเพิ่มเติม ซึ่งอาจทำให้เกิดช่องโหว่ด้านความปลอดภัยได้

เพื่อแก้ไขข้อจำกัดเหล่านี้ AS-KRS จึงนำ PKCE-like Challenge มาใช้ ซึ่งเป็นรูปแบบหนึ่งของ token-based authentication ที่ช่วยให้มั่นใจได้ว่าผู้ขอคืนกุญแจเป็นบุคคลที่ได้รับอนุญาตจริง ๆ

- เพิ่มการยืนยันตัวตนหลายชั้น: PKCE-like Challenge ช่วยให้กระบวนการยืนยันตัวตนมีความเข้มงวดมากขึ้น โดยกำหนดให้ผู้ใช้ต้องพิสูจน์ตัวตนผ่าน Challenge Code และ Code Verifier
- ลดโอกาสในการโจมตีแบบ Man-in-the-Middle: ด้วยการใช้ Token Authentication กระบวนการยืนยันตัวตนจะมีการสร้างและแลกเปลี่ยนโทเค็นที่แตกต่างกันในแต่ละครั้ง ทำให้ยากต่อการปลอมแปลง

ระบบนี้ยังช่วยให้การควบคุมการเข้าถึงและการกู้คืนกุญแจจาก KRA (Key Recovery Agent) มีความปลอดภัยมากขึ้น โดยที่ KRC (Key Recovery Center) ทำหน้าที่แจกจ่าย KRF ให้กับ KRA และรวมรวมกุญแจเหล่านี้เมื่อจำเป็นต้องทำการกู้คืน

ระบบ Authenticated Secure Key Recovery System (AS-KRS) ถูกพัฒนาขึ้นเพื่อแก้ไขข้อจำกัดของ SFM-KRS โดยเพิ่มกระบวนการตรวจสอบสิทธิ์แบบหลายชั้นตอน และใช้ PKCE-like Challenge เพื่อลดความเสี่ยงจากการถูกโจมตี MITM และการขโมย Private Key ระบบนี้จะช่วยให้กระบวนการกู้คืนกุญแจมีความปลอดภัยมากขึ้น และมั่นใจได้ว่าผู้ที่ร้องขอคืนกุญแจเป็นผู้มีสิทธิ์ที่ถูกต้อง



รูปที่ 2.2 การพัฒนา SFM-KRS ด้วยการเพิ่ม Authentication ในระหว่างการสื่อสาร

### 2.3 การยืนยันตัวตนหลายขั้นตอน (Multi-Way Authentication)

การยืนยันตัวตนหลายขั้นตอน (Multi-Way Authentication) เป็นกระบวนการที่ใช้หลายปัจจัยในการยืนยันตัวตนของผู้ใช้งาน โดยการใช้มากกว่าหนึ่งวิธีในการตรวจสอบสิทธิ์ เช่น รหัสผ่าน การยืนยันด้วยบิรบรอง โทคين หรือรหัส OTP กระบวนการนี้มีหน้าที่ตรวจสอบว่าผู้ใช้งานมีสิทธิ์เข้าถึงข้อมูล โดยผ่านการยืนยันตัวตนที่หลากหลาย เพื่อลดความเสี่ยงจากการถูกแฮกหรือการโจมตีแบบ Man-in-the-Middle (MitM)

ในการออกแบบระบบการคุ้มครองแบบปลอดภัย (Authenticated Secure Key Recovery System - AS-KRS) กระบวนการยืนยันตัวตนหลายขั้นตอนนี้จะถูกใช้ในการตรวจสอบสิทธิ์ก่อนที่จะอนุญาตให้ทำการคุ้มครอง และจะมีหลายขั้นตอนในการตรวจสอบสิทธิ์แต่ละฝ่าย เช่น Sender, Receiver, Key Recovery Center (KRC), และ Key Recovery Agent (KRA) ซึ่งจะต้องผ่านการยืนยันตัวตนอย่างเข้มงวดเพื่อให้แน่ใจว่าผู้ที่ขอคุ้มครองและมีสิทธิ์และได้รับอนุญาตจริง ๆ โดยใช้เทคโนโลยีการยืนยันตัวตนหลายชั้น เช่น การใช้โทคินหรือการยืนยันผ่านวิธีการต่าง ๆ ที่มีความปลอดภัยสูง

Multi-Way Authentication เป็นการป้องกันที่สำคัญในการลดความเสี่ยงจากการถูกโจมตีหรือการเข้าถึงข้อมูลโดยผู้ที่ไม่ได้รับอนุญาต โดยเฉพาะในการถูกคืนกุญแจ ซึ่งหากไม่มีการยืนยันตัวตนที่หลากหลาย อาจทำให้ระบบเสี่ยงต่อการถูกโจมตีแบบ Man-in-the-Middle หรือการแอบอ้างตัวตนได้

การยืนยันตัวตนหลากหลายขั้นตอน (Multi-Way Authentication) นี้จะนำมาระบุกต์ใช้ในระบบ AS-KRS โดยจะเพิ่มการยืนยันตัวตนในทุกขั้นตอนของกระบวนการถูกคืนกุญแจ ดังแต่การร้องขอ กุญแจจากผู้ใช้งาน การร้องขอ กุญแจยื่นจาก KRA ไปจนถึงการรวมกุญแจกลับคืนที่ KRC โดยจะใช้การยืนยันตัวตนที่มีความหลากหลายเพื่อเพิ่มความปลอดภัย เช่น การใช้โทเค็นหรือการยืนยันตัวตนผ่านวิธีการอื่น ๆ ที่มีความปลอดภัยสูง

**ตารางที่ 2.1 ตารางเปรียบเทียบข้อดีและข้อเสียของรูปแบบการตรวจสอบลิทีที่นิยมใช้ในปัจจุบัน**

รูปแบบการตรวจสอบลิที	ข้อดี	ข้อเสีย
Password-based Authentication (การตรวจสอบลิทีด้วยรหัสผ่าน)	ง่ายต่อการใช้งาน ไม่ต้องใช้ ชาร์ดแวร์เพิ่มเติม	รหัสผ่านอาจถูกแฮกหรือคาด เค่าได้ง่าย ต้องใช้การจัดการ รหัสผ่านที่ดีเพื่อความ ปลอดภัยสูงสุด
Multi-factor Authentication (MFA) (การตรวจสอบลิที หลากหลายปัจจัย)	เพิ่มความปลอดภัย โดยการใช้ หลากหลายปัจจัยในการตรวจสอบ ลดโอกาสของการโจมตีจาก การขโมยรหัสผ่าน	ใช้งานซับซ้อนมากขึ้น ต้อง พึ่งพาอุปกรณ์เพิ่มเติม เช่น โทรศัพท์ หรือโทเค็น
Biometric Authentication (การตรวจสอบลิทีทางชีว มิติ)	สะดวกและรวดเร็ว มี เอกลักษณ์เฉพาะตัว ยากต่อ การปลอมแปลง	ข้อมูลชีวมิติคือข้อมูลที่ไม่ สามารถเปลี่ยนแปลงได้ หาก ถูกละเมิดแล้วจะมีความเสี่ยง สูง ต้องใช้อุปกรณ์พิเศษ สำหรับการตรวจจับ
Token-based Authentication (เช่น OAuth, JWT)	ไม่ต้องเก็บข้อมูลรับรองระยะ ยาว เช่น รหัสผ่าน มีความ ปลอดภัยสูงในการตรวจสอบ ข้ามโดเมน	หากโทเค็นถูกขโมย ก็อาจถูก นำไปใช้งานได้ ต้องใช้การ จัดการโทเค็นให้ดี

Certificate-based Authentication (การตรวจสอบลิฟท์ด้วยใบรับรอง)	มีความปลอดภัยสูงมาก ใช้งานได้กับระบบที่มีความสำคัญสูง	ข้อบังคับในการจัดการใบรับรอง ต้องพึ่งพาโครงสร้างพื้นฐานของ PKI (Public Key Infrastructure)
Knowledge-based Authentication (KBA) (การตรวจสอบลิฟท์ด้วยความรู้)	ง่ายต่อการใช้งาน โดยไม่ต้องใช้อุปกรณ์เสริม ไม่ต้องจำรหัสผ่านยากๆ	ข้อมูลสามารถน้ำหนาหรือคาดเดาได้ เช่น คำถ้ามเกี่ยวกับชีวิตส่วนตัว
Simultaneous Authentication of Equals (SAE)	ความปลอดภัยสูง โดยเฉพาะในระบบเครือข่ายไร้สาย ลดโอกาสของการโจมตีด้วยการ截听 (eavesdropping)	การใช้งานยังไม่แพร่หลายมาก ข้อบังคับกว่าการใช้รหัสผ่านธรรมดา

### ระดับความปลอดภัยจากการใช้งาน PKI structure ร่วมกับ PKCE-like challenge

- PKI structure แบบดั้งเดิมมีความสามารถในการป้องกันการโจมตีทางประเภท เช่น
  - การโจมตีแบบตักฟัง (Eavesdropping):** ข้อมูลที่ถูกเข้ารหัสด้วย PKI จะไม่สามารถถูกตักฟังได้ง่าย เพราะผู้โจมตีต้องมี private key ของฝ่ายที่ต้องการเข้าถึงข้อมูล
  - การโจมตีแบบ replay attack:** PKI มีการป้องกันไม่ให้มีการใช้ข้อมูลซ้ำในการโจมตี (เช่น ส่งแพ็กเก็ตเดิม ๆ เพื่อโจมตีระบบ)
- อย่างไรก็ตาม ระบบ PKI ดั้งเดิมยังมีความเสี่ยงต่อการโจมตีทางรูปแบบ เช่น
  - การโจมตีแบบ Man-in-the-Middle:** หากผู้โจมตีสามารถโยนห์หรือเลียนแบบ private key ของผู้ใช้งาน การโจมตีแบบนี้จะสามารถเกิดขึ้นได้
  - การโจมตีโดยกุญแจส่วนตัว (Private Key):** หาก private key ถูกหักโยนห์ ผู้โจมตีสามารถถอดรหัสข้อมูลทั้งหมดได้
- การเพิ่ม PKCE-like challenge เข้ามาช่วยยกระดับความปลอดภัย โดยการเพิ่มการยืนยันตัวตนแบบ token-based authentication ซึ่งช่วยลดความเสี่ยงในการโจมตีหลายรูปแบบ เช่น
  - ลดความเสี่ยงในการโจมตี Man-in-the-Middle:** ด้วยการยืนยันตัวตนซ้ำในระหว่างกระบวนการส่งข้อมูล การโจมตี MITM จะทำได้ยากยิ่งขึ้น
  - ลดความเสี่ยงจากการโจมตีโดย private key:** เมื่อผู้โจมตีจะได้ private key แต่ยังต้องผ่านกระบวนการยืนยันตัวตนผ่าน token ก่อน จึงจะสามารถเข้าถึงข้อมูลได้

ถึงแม้ PKCE-like challenge จะเพิ่มความปลอดภัย แต่ยังคงมีช่องโหว่ว่างประเภท เช่น การโจมตีแบบ Phishing ซึ่งผู้ใช้งานอาจถูกหลอกให้เปิดเผยข้อมูลส่วนตัว

#### **2.4 ระบบ PKI และการยืนยันตัวตนด้วยใบรับรอง (Certified Authentication)**

Public Key Infrastructure (PKI) เป็นระบบที่ใช้ในการสร้างและจัดการใบรับรองดิจิทัล (Digital Certificates) ซึ่งช่วยให้สามารถยืนยันตัวตนและการเข้าถึงข้อมูลได้อย่างปลอดภัย PKI ประกอบด้วยองค์ประกอบสำคัญหลายส่วน ได้แก่ กุญแจสาธารณะ (Public Key), กุญแจส่วนตัว (Private Key), และใบรับรองดิจิทัลที่ออกโดย Certificate Authority (CA) ซึ่งมีบทบาทในการรับรองความถูกต้องของผู้ใช้และความน่าเชื่อถือของข้อมูล

การใช้ PKI ช่วยให้การยืนยันตัวตนของผู้ส่งและผู้รับข้อมูลสามารถทำได้โดยไม่ต้องส่งกุญแจลับผ่านเครือข่าย ซึ่งลดความเสี่ยงจากการถูกโจมตี เช่น การโจมตี Man-in-the-Middle (MitM) ในระบบการส่งข้อมูลระหว่างผู้ใช้

ในโครงการนี้ PKI ใช้ในการยืนยันตัวตนของแต่ละฝ่าย เช่น Sender, Receiver, Key Recovery Center (KRC), และ Key Recovery Agent (KRA) โดยมีการใช้ใบรับรองดิจิทัลเพื่อรับรองตัวตนของแต่ละฝ่าย และเพื่อให้การรับส่งข้อมูลระหว่างฝ่ายต่าง ๆ เป็นไปอย่างปลอดภัยและได้รับการตรวจสอบอย่างถูกต้อง

#### **2.5 การใช้ Public และ Private Key**

ในโครงการนี้ กุญแจสาธารณะ (Public Key) จะถูกใช้ในการเข้ารหัสข้อมูลที่ส่งระหว่างฝ่ายต่าง ๆ ส่วนกุญแจส่วนตัว (Private Key) จะถูกใช้ในการถอดรหัสข้อมูลที่ได้รับ โดยที่ Public Key ของแต่ละฝ่ายจะถูกเผยแพร่เพื่อให้สามารถเข้ารหัสข้อมูลได้ แต่ Private Key จะถูกเก็บรักษาไว้อย่างปลอดภัยเพื่อป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

การใช้ PKI และการจัดการ Public/Private Key มีบทบาทสำคัญในการรับประกันความปลอดภัยของระบบ เนื่องจากช่วยป้องกันการถูกโจมตีจากผู้ที่ไม่ได้รับอนุญาต

ระบบการยืนยันตัวตนด้วยใบรับรองดิจิทัล (Certificate-based Authentication) นี้ช่วยให้มั่นใจได้ว่าแต่ละฝ่ายที่มีการติดต่อกันผ่านระบบ AS-KRS จะได้รับการยืนยันตัวตนอย่างถูกต้อง

และเป็นมาตรฐาน ชี้งลดความเสี่ยงจากการ โจมตีต่าง ๆ เช่น การแอบอ้างตัวตนจากผู้ไม่หวังดีและ การ โจมตีแบบ Man-in-the-Middle



รูปที่ 2.3 PKI

ที่มา: <https://www.etda.or.th/getattachment/f3232119-eb34-41e1-8040-f7d46775fdd4/How-PKI-Works.aspx>

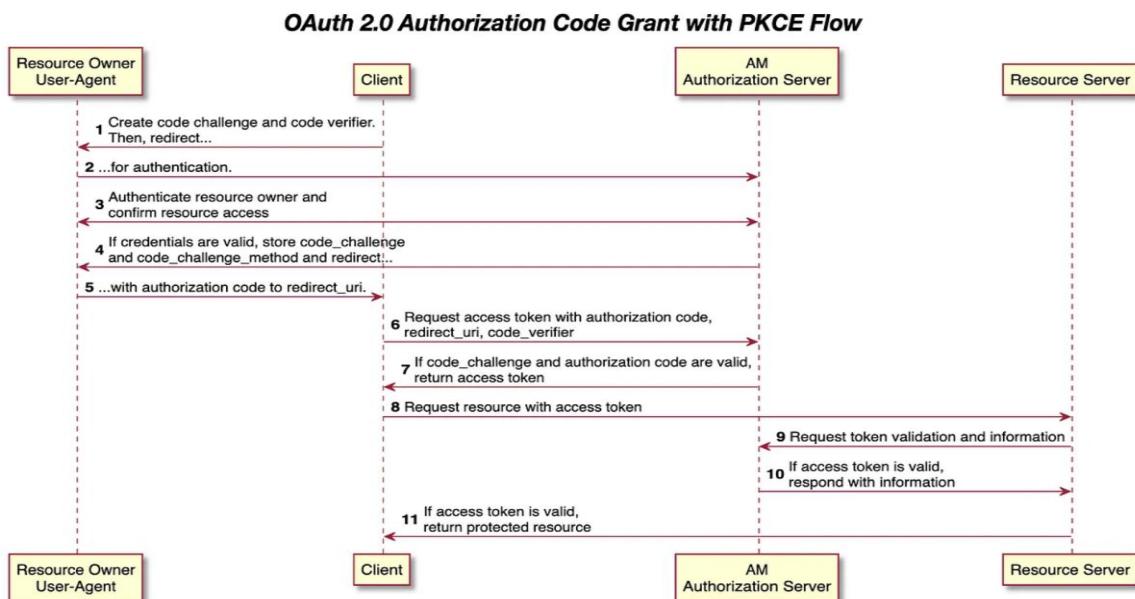
## 2.6 PKCE (Proof Key for Code Exchange)

PKCE (Proof Key for Code Exchange) เป็นกลไกการยืนยันตัวตนที่ใช้ในระบบ OAuth 2.0 เพื่อป้องกันการ โจมตีแบบ Interception โดยใช้กระบวนการ Challenge-Response ในการแลกเปลี่ยนโทเค็น โดยผู้ใช้จะต้องส่งรหัสท้าทาย (Code Challenge) ไปยังเซิร์ฟเวอร์ และเมื่อเซิร์ฟเวอร์ตอบกลับ จะต้องส่งรหัสยืนยัน (Code Verifier) ที่ตรงกับรหัสท้าทายเพื่อให้การยืนยันตัวตนสมบูรณ์

ในโครงการนี้ หลักการ PKCE ถูกนำมาใช้ในกระบวนการ PKCE-like Challenge เพื่อยืนยันตัวตนในการขอคุ้นคุ้นและ การส่งข้อมูลไปยัง Key Recovery Agent (KRA) โดยกระบวนการนี้จะช่วยป้องกันการ โจมตีที่อาจเกิดขึ้นระหว่างการแลกเปลี่ยนคุ้นคุ้นและช่วยยืนยันตัวตนของแต่ละฝ่ายอย่างมั่นคง

PKCE-like Challenge จะใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ถูกส่งระหว่างฝ่ายต่าง ๆ เช่น Sender, Receiver, KRC, และ KRA โดยช่วยให้มั่นใจได้ว่าในการแลกเปลี่ยนข้อมูลระหว่างฝ่ายต่าง ๆ จะไม่มีการแอบแฝงตัวของผู้โจมตีและไม่มีการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

การใช้ PKCE-like Challenge จะเสริมความปลอดภัยให้กับระบบการกู้คืนกุญแจ (AS-KRS) โดยสามารถยืนยันตัวตนและป้องกันการโจมตีที่อาจเกิดขึ้นจากการแอบแฝงตัวของผู้โจมตีในระหว่างการแลกเปลี่ยนข้อมูล ทำให้การขอ กู้คืนกุญแจและการส่งข้อมูลไปยัง KRA เป็นไปอย่างปลอดภัยและมั่นคง



รูปที่ 2.4 OAuth 2.0 with PKCE

ที่มา:

[https://miro.medium.com/v2/resize:fit:1400/format:webp/1\\*ktOVkcY3Zjh2kmaAPoSg.png](https://miro.medium.com/v2/resize:fit:1400/format:webp/1*ktOVkcY3Zjh2kmaAPoSg.png)

## 2.7 PKCE-like Challenge

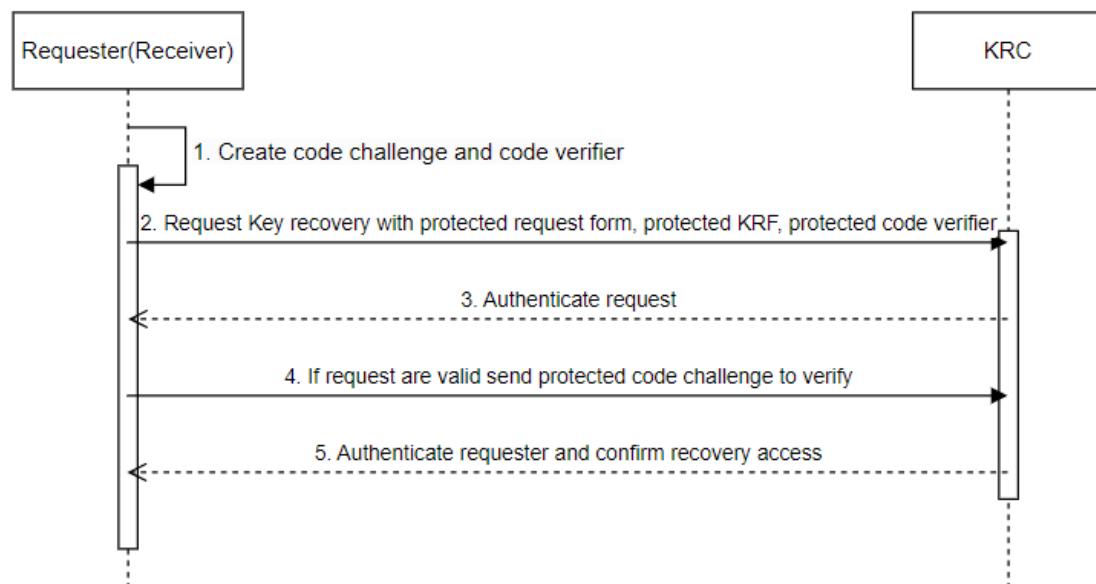
PKCE-like Challenge เป็นการปรับใช้กระบวนการ PKCE (Proof Key for Code Exchange) เพื่อใช้ในการตรวจสอบสิทธิ์ในระบบกู้คืนกุญแจ โดยกระบวนการทำงานยังคงเป็นแบบ Challenge-Response ซึ่งเริ่มต้นจากฝ่ายที่ขอ กู้คืนกุญแจ (เช่น Receiver หรือ Key Recovery Center - KRC) ที่จะสร้างโค้ดท้าทาย (Challenge Code) และส่งไปยังฝ่ายที่ทำการยืนยันตัวตน (เช่น Key Recovery Agent - KRA) เมื่อ KRA ได้รับโค้ดท้าทายแล้ว จะตอบกลับด้วยโค้ดตอบกลับ (Response Code) จากนั้นฝ่ายที่ขอ กู้คืนกุญแจจะต้องส่ง โค้ดยืนยัน (Verifier Code) ที่ตรงกับโค้ดท้าทาย หลังจากทำการเช็ค เพื่อให้การยืนยันตัวตนสมบูรณ์

PKCE-like Challenge นี้ถูกนำมาใช้เพื่อเพิ่มความปลอดภัยในการรักษาคืนกุญแจและช่วยป้องกันการโจมตีประเภทการปลอมแปลงหรือการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตในระหว่างกระบวนการรักษาคืนกุญแจ ซึ่งช่วยให้การยืนยันตัวตนของแต่ละฝ่ายในระบบมีความมั่นคงยิ่งขึ้น

การทำงานของ PKCE-like Challenge นี้จะช่วยให้ฝ่ายต่าง ๆ ที่เกี่ยวข้อง เช่น Receiver, KRC และ KRA สามารถตรวจสอบสิทธิ์ในการเข้าถึงข้อมูลหรือกุญแจได้อย่างมั่นใจ โดยการใช้กระบวนการที่คล้ายกับ PKCE ซึ่งเดิมถูกออกแบบมาเพื่อป้องกันการโจมตีจากการ截จับโทเค็น (Token Interception) ในระบบ OAuth 2.0 โดยจะมีการส่งโค้ดท้าทายและโค้ดยืนยันที่ผ่านกระบวนการเชื่อมและการตรวจสอบการเข้ารหัสที่ถูกต้อง

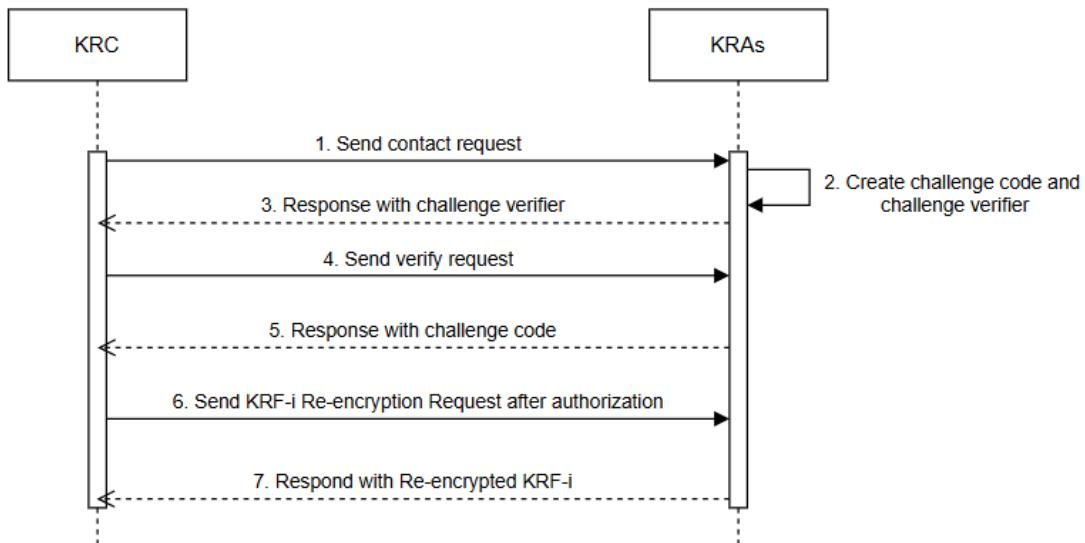
ในโปรเจคนี้ PKCE-like Challenge ถูกนำมาใช้เพื่อยืนยันตัวตนระหว่างฝ่ายต่าง ๆ ก่อนการแลกเปลี่ยนข้อมูลหรือการรักษาคืนกุญแจ ซึ่งช่วยให้มั่นใจได้ว่าแต่ละฝ่ายที่มีการแลกเปลี่ยนข้อมูลหรือกุญแจยังคงเป็นฝ่ายที่ได้รับอนุญาตจริง ๆ โดยกระบวนการนี้จะช่วยเสริมความปลอดภัยในการรักษาคืนกุญแจและป้องกันการโจมตีประเภทการปลอมแปลงหรือการเข้าถึงข้อมูลที่ไม่ได้รับอนุญาตในระบบ

#### Authorization with PKCE-like Challenge between Receiver and KRC



รูปที่ 2.5 Authorization with PKCE-like Challenge between Receiver and KRC

### Authorization with PKCE-like challenge between KRC and KRAs



รูปที่ 2.6 Authorization with PKCE-like Challenge between KRC and KRAs

### 2.8 การกระจายกุญแจและการรวมกุญแจ (Key Splitting and Reconstruction)

การกระจายกุญแจ (Key Splitting) เป็นกระบวนการที่ใช้ในการแบ่งกุญแจหลักออกเป็นหลายส่วนย่อย เพื่อเพิ่มความปลอดภัยในการเก็บรักษา กุญแจหลัก โดยที่แต่ละฝ่ายจะถือส่วนหนึ่งของ กุญแจ และการรวมกุญแจ (Key Reconstruction) คือการนำกุญแจย่อยเหล่านั้นกลับมาร่วมกันเพื่อสร้างกุญแจหลักขึ้นมาใหม่เมื่อจำเป็นต้องใช้

การกระจายกุญแจช่วยลดความเสี่ยงจากการโจมตี โดยการที่ฝ่ายเดียวไม่สามารถเข้าถึงกุญแจทั้งหมดได้ ซึ่งจะทำให้การขโมยข้อมูลจากการเข้าถึงกุญแจหลักทำได้ยากขึ้น การรวมกุญแจเกิดขึ้นเมื่อฝ่ายต่าง ๆ ต้องนำส่วนที่ถืออยู่มาร่วมกันเพื่อสร้างกุญแจหลัก โดยการกระทำนี้จะช่วยให้สามารถกู้คืนข้อมูลได้อย่างปลอดภัยเมื่อจำเป็น

ในโครงการนี้ใช้เทคนิค XOR-based Key Splitting เพื่อแบ่งกุญแจหลักออกเป็นหลายส่วนย่อย และให้แต่ละฝ่ายถือส่วนหนึ่งของกุญแจ การใช้ XOR-based Key Splitting ช่วยให้การกู้คืนกุญแจสามารถทำได้โดยการรวมกุญแจย่อยจากฝ่ายที่เกี่ยวข้อง และกระบวนการรวมกุญแจจะเกิดขึ้นเมื่อ KRA (Key Recovery Agent) แต่ละคนร่วมส่วนของกุญแจที่ตนเองถืออยู่และต้องผ่านการตรวจสอบสิทธิ์ร่วมกับ KRC (Key Recovery Center) เพื่อทำการรวมกุญแจให้สมบูรณ์

หนึ่งในแนวทางที่ใช้ใน AS-KRS คือการแบ่งกุญแจออกเป็นส่วนย่อย (Key Recovery Fragments - KRF) และกระจายไปยัง KRA หลายแห่งเพื่อลดความเสี่ยงในการโจมตี

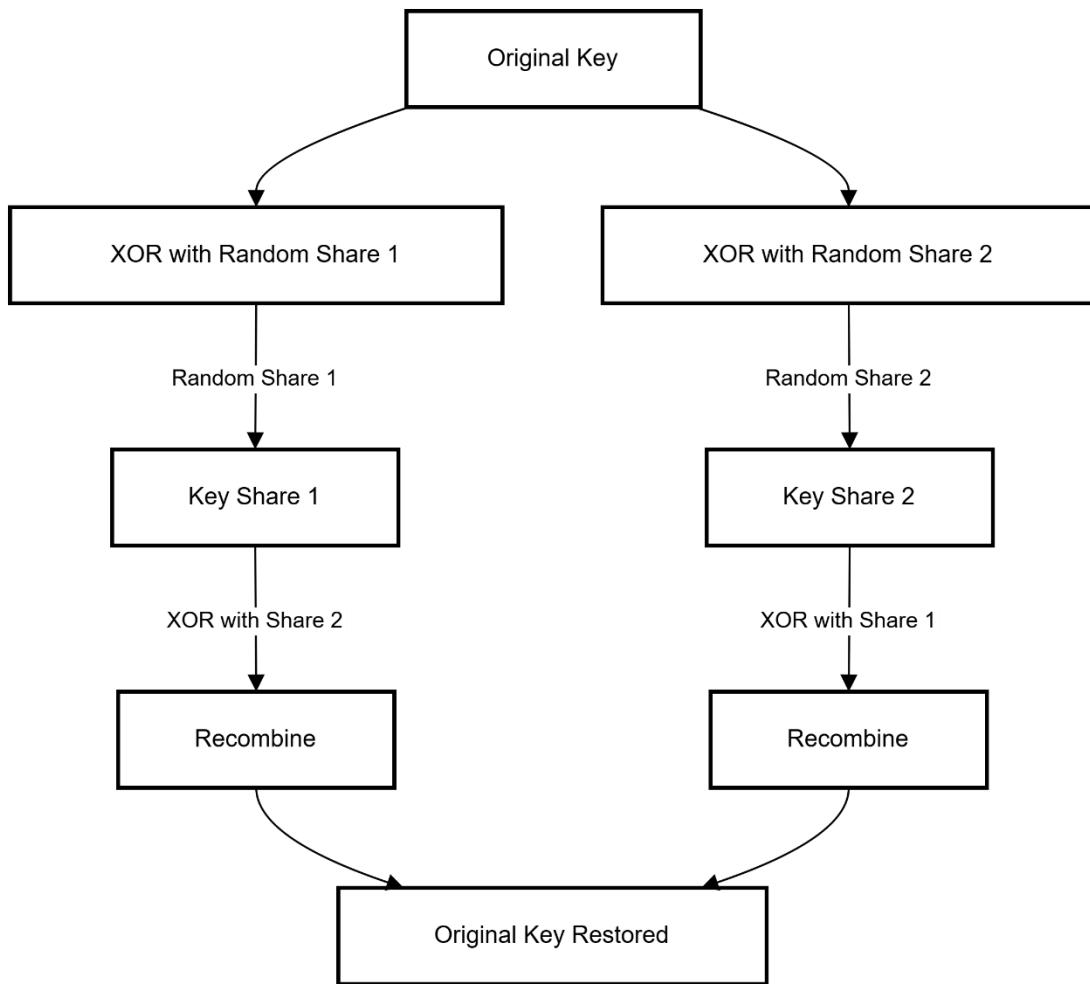
#### ข้อดีของ Key Sharding

- ลดโอกาสที่กุญแจทั้งหมดจะถูกขโมย
- เพิ่มความปลอดภัยในการกู้คืนกุญแจ เนื่องจากต้องมีหลายฝ่ายร่วมกันประกอบกุญแจ
- ลดภาระของ KRC ในการเก็บกุญแจทั้งหมด

#### ข้อเสียของ Key Sharding

- หาก KRA บางส่วนถูกโจมตี อาจทำให้การกู้คืนกุญแจได้ยากขึ้น
- ต้องมีการจัดการสิทธิ์ในการเข้าถึงแต่ละ KRA อย่างรัดกุม

กระบวนการกระจายและรวมกุญแจนี้ช่วยเพิ่มความปลอดภัยและประสิทธิภาพในการกู้คืนกุญแจในระบบการกู้คืนกุญแจ (Key Recovery System) ทั้งยังป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตจากการที่ฝ่ายเดียวมีอำนาจเข้าถึงกุญแจทั้งหมด



รูปที่ 2.7 XOR-based key splitting

## 2.9 One-way encryption

การเข้ารหัสแบบทางเดียว (One-way encryption) หรือที่เรียกว่าฟังก์ชันแฮช (Hashing) เป็นเทคนิคที่ใช้ในการแปลงข้อมูลให้กลายเป็นค่าแฮชที่ไม่สามารถย้อนกลับไปหาข้อมูลต้นฉบับได้ วิธีการนี้ใช้ในกรณีที่ไม่จำเป็นต้องเก็บข้อมูลต้นฉบับ แต่ต้องการตรวจสอบความถูกต้อง เช่น ในการเก็บรหัสผ่านผู้ใช้หรือการตรวจสอบความถูกต้องของข้อมูล

ฟังก์ชันแฮชมีลักษณะสำคัญที่ทำให้ไม่สามารถย้อนกลับไปหาข้อมูลต้นฉบับได้ และยังช่วยเพิ่มความปลอดภัยในการจัดเก็บข้อมูลสำคัญ

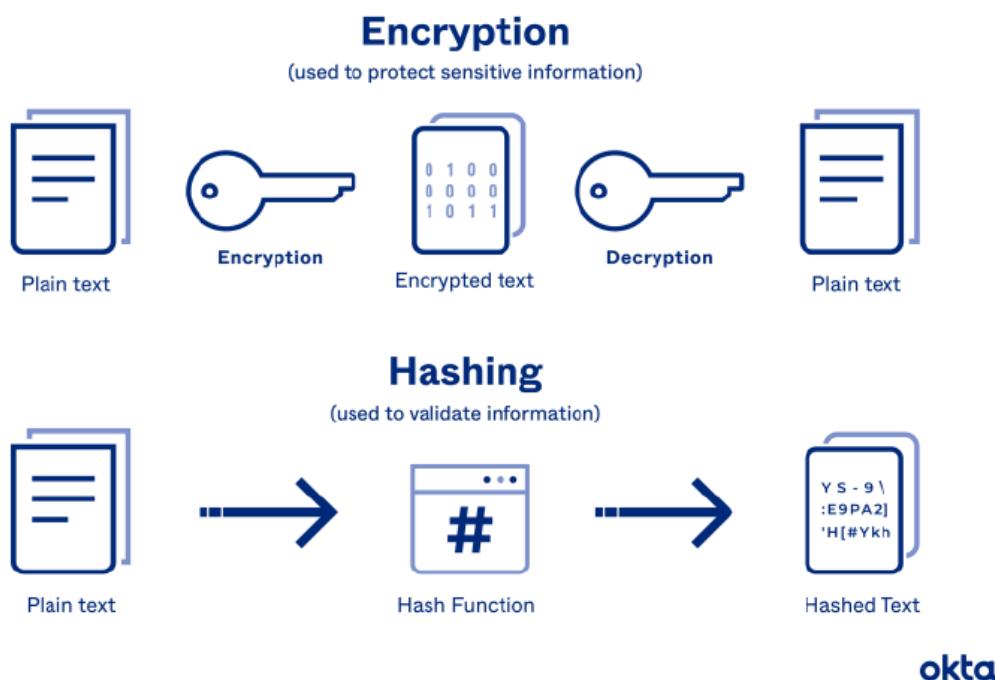
## 2.10 คุณสมบัติที่สำคัญของฟังก์ชันแฮช

Pre-image Resistance: การหาข้อมูลต้นฉบับจากค่าแฮชต้องยากหรือเกือบจะเป็นไปไม่ได้ การถอดรหัสจากค่าแฮชจะต้องใช้วิถีทางมากหรือมีความยากสูง

Collision Resistance: ต้องไม่มีการสร้างข้อมูลที่แตกต่างกันแต่มีค่าแฮชเหมือนกัน วิธีทางคณิตศาสตร์ไม่สามารถสร้างข้อมูลสองชุดที่มีค่าแฮชเหมือนกันได้

Second Pre-image Resistance: ไม่ควรสามารถหาข้อมูลใด ๆ ที่มีค่าแฮชเหมือนกับข้อมูลที่กำหนดไว้ล่วงหน้า

การเข้ารหัสแบบทางเดียวมักใช้ในหลายกรณี เช่น ในการเก็บรักษารหัสผ่านของผู้ใช้ โดยรหัสผ่านจะถูกเข้ารหัสเก็บไว้ในรูปแบบค่าแฮช และเมื่อผู้ใช้ล็อกอิน รหัสผ่านที่ป้อนจะถูกแฮชแล้ว เปรียบเทียบกับค่าแฮชที่เก็บไว้ในระบบ หากตรงกันก็จะสามารถเข้าสู่ระบบได้ นอกจากนี้ยังมีการใช้ฟังก์ชันแฮชในการเชื่นถ้อยเชิงดิจิทัลและการยืนยันความถูกต้องของข้อมูล เช่น การแนบเอกสารในอีเมลหรือข้อมูลทางการแพทย์



รูปที่ 2.8 One-way encryption

ที่มา: <https://www.okta.com/sites/default/files/styles/tinypng/public/media/image/2021-03/hashing-vs-encryption.png?itok=OSCh13sC>

ฟังก์ชันแซฟท์นิยมใช้ในการเข้ารหัสแบบทางเดียว ได้แก่ SHA-256, SHA-3, bcrypt, scrypt, และ PBKDF2 ซึ่งเป็นอัลกอริธึมแซฟท์ที่มีความปลอดภัยสูงและสามารถป้องกันการโจมตีด้วยการทำให้ค่าแซฟท์ซับซ้อนขึ้น รวมถึงการเพิ่มการสุ่มในกระบวนการแซฟท์ เพื่อลดความเสี่ยงในการพบค่าแซฟท์เหมือนกันสำหรับข้อมูลที่แตกต่างกัน

## 2.11 การเข้ารหัสแบบสมมาตร (Symmetric Encryption)

การเข้ารหัสแบบสมมาตร (Symmetric Encryption) เป็นวิธีการเข้ารหัสที่ใช้กุญแจเดียวในการเข้ารหัสและถอดรหัสข้อมูล โดยทั้งผู้ส่งและผู้รับจะใช้กุญแจเดียวกันในการดำเนินการ ซึ่งทำให้การเข้ารหัสและถอดรหัสมีความเร็วสูงและประสิทธิภาพดี แต่ในขณะเดียวกันก็มีความเสี่ยงในการถูก破解เจาะรั้วไว้ไปยังบุคคลที่ไม่ได้รับอนุญาต

การใช้กุญแจสมมาตรหมายความว่าสำหรับการเข้ารหัสข้อมูลที่ต้องการความเร็วและประสิทธิภาพในการดำเนินการ เช่น การส่งข้อมูลจำนวนมากหรือการสื่อสารในช่วงเวลาสั้น ๆ อย่างไรก็ตาม หากกุญแจถูกไขหรือรั่วไว้ไปยังบุคคลที่ไม่ได้รับอนุญาต จะสามารถถอดรหัสข้อมูลได้โดยง่าย

### 2.11.1 การใช้กุญแจชั่วคราว (Session Key)

ในโครงการนี้ใช้ Session Key ซึ่งเป็นกุญแจสมมาตรที่ถูกสร้างขึ้นเพื่อใช้ในการเข้ารหัสข้อมูลในช่วงเวลาหนึ่งหรือภายใต้เงื่อนไขใดๆ ก็ตาม ไม่สามารถนำมารีเซ็ตได้ จึงต้องรักษาความลับของ Session Key อย่างเคร่งครัด

การใช้ Session Key ช่วยเพิ่มความปลอดภัยในการสื่อสารระหว่างผู้ส่งและผู้รับ โดยป้องกันการดักจับข้อมูลในระหว่างการส่งข้อมูล ตัวอย่างเช่น เมื่อ Sender เข้ารหัสข้อมูลที่ส่งไปยัง Receiver จะใช้ Session Key ใน การเข้ารหัสข้อมูล เมื่อ Receiver ต้องการถอดรหัสข้อมูลที่ได้รับ จะต้องใช้ Session Key ที่ได้รับคืนจาก Key Recovery Center (KRC) เพื่อถอดรหัสข้อมูล

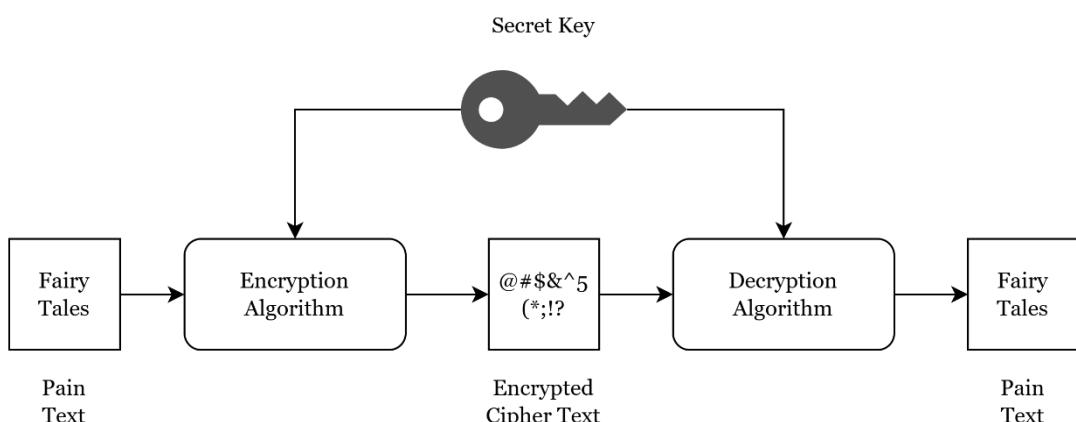
### 2.11.2 การแบ่งและจัดเก็บ Session Key

เพื่อเพิ่มความปลอดภัยในการถูกค้น Session Key จะถูกแบ่งออกเป็นหลายส่วนหรือที่เรียกว่า split key และจัดเก็บโดย Key Recovery Agent (KRA) หลายแห่ง การกระจายกุญแจเจ้าของช่วยลดความเสี่ยงจากการโจมตี โดยผู้โจมตีจะต้องได้รับข้อมูลจากหลาย KRA ซึ่งทำให้การโจมตีมีความซับซ้อนขึ้นและยากต่อการทารุณ

การกระจายและจัดเก็บ Session Key ในลักษณะนี้ช่วยให้การถูกคืบกู้แจ้งได้อย่างปลอดภัยมากขึ้น เนื่องจากต้องมีการร่วมมือจากหลาย KRA เพื่อสร้างกุญแจหลักขึ้นมาใหม่

### 2.11.3 ความเร็วในการเข้ารหัส

การใช้ Session Key ช่วยเพิ่มประสิทธิภาพในการเข้ารหัสและถอดรหัสข้อมูล เนื่องจากการใช้กุญแจสมมาตรในการเข้ารหัสทำให้การส่งข้อมูลขนาดใหญ่สามารถทำได้เร็วขึ้น โดยไม่สูญเสียความปลอดภัยในการสื่อสาร การใช้ Session Key จึงเป็นวิธีที่เหมาะสมในการเพิ่มประสิทธิภาพการทำงานของระบบเข้ารหัสแบบสมมาตรในโครงการนี้



รูปที่ 2.9 การเข้ารหัสแบบกุญแจสมมาตร(Symmetric Key Cryptography)

## 2.12 การเข้ารหัสแบบสมมาตร (Asymmetric Encryption)

การเข้ารหัสแบบสมมาตร (Asymmetric Encryption) หรือที่รู้จักกันในชื่อว่า "Public Key Cryptography" เป็นเทคนิคการเข้ารหัสที่ใช้กุญแจคู่ในการเข้ารหัสและถอดรหัสข้อมูล ซึ่งประกอบด้วย กุญแจสาธารณะ (Public Key) และ กุญแจส่วนตัว (Private Key) โดยกุญแจสาธารณะใช้ในการเข้ารหัสข้อมูล ขณะที่กุญแจส่วนตัวจะใช้ในการถอดรหัสข้อมูล

การเข้ารหัสแบบสมมาตรเหมาะสมในการใช้งานที่ต้องการให้ข้อมูลสามารถถูกเข้ารหัสโดยผู้ใดก็ได้ โดยที่เฉพาะเจ้าของกุญแจส่วนตัวเท่านั้นที่จะสามารถถอดรหัสข้อมูลได้ เทคนิคนี้ช่วยเพิ่มความปลอดภัยในการส่งข้อมูลผ่านช่องทางที่ไม่ปลอดภัย เช่น อินเทอร์เน็ต เนื่องจากกุญแจสาธารณะสามารถเผยแพร่ได้โดยไม่ต้องกลัวการถูกโจรกรรม

ในโครงการนี้ การเข้ารหัสแบบสมมาตรถูกใช้ในการเข้ารหัสข้อมูลสำคัญ เช่น Key Recovery File (KRF) ที่ใช้ในการถูกคืนกุญแจ โดยข้อมูลที่สำคัญจะถูกเข้ารหัสด้วย กุญแจสาธารณะ ของฝ่ายที่

เกี่ยวข้อง เช่น KRA หรือ KRC ซึ่งทำให้สามารถรับรองได้ว่าผู้ที่สามารถอ่านครหัสได้คือผู้ที่ถือกุญแจส่วนตัวที่ตรงกัน

#### 2.12.1 การใช้งาน AES ในการเข้ารหัส KRF

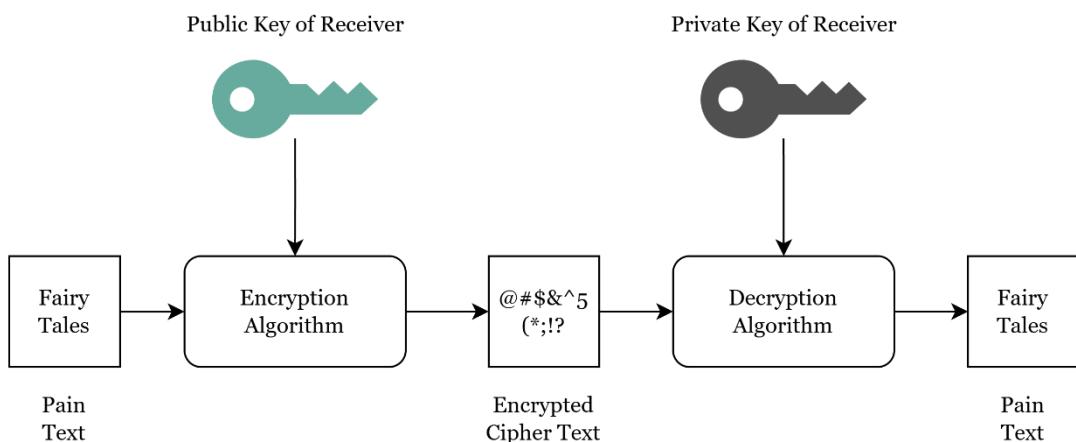
ในโครงการนี้ การเข้ารหัสแบบสมมาตรลูกน้ำมาร่วมกับการเข้ารหัสแบบสมมาตร (AES) ในการจัดการกับ Key Recovery File (KRF) โดยใช้กุญแจสาธารณะในการเข้ารหัส AES Key ซึ่งใช้ในการเข้ารหัสข้อมูล KRF เพื่อเพิ่มความปลอดภัยในการเก็บรักษาข้อมูลที่สำคัญและเพื่อให้แน่ใจว่าข้อมูลที่เข้ารหัสจะสามารถอ่านครหัสได้โดยผู้ที่มีสิทธิ์เท่านั้น

เมื่อ Sender สร้าง KRF ขึ้นมา จะทำการเข้ารหัส KRF ด้วย AES Key ก่อนแล้วจึงนำ AES Key ไปเข้ารหัสอีกครั้งด้วย กุญแจสาธารณะ ของฝ่ายที่เกี่ยวข้อง เช่น KRA หรือ KRC ซึ่งการกระทำนี้ช่วยเพิ่มชั้นความปลอดภัยในการป้องข้อมูลสำคัญจากการลูกโจมตี

#### 2.12.2 ความสำคัญของการเข้ารหัสแบบสมมาตร

การเข้ารหัสแบบสมมาตรช่วยในการป้องกันการโจมตีประเภทต่างๆ เช่น การแอบแฝงตัว การดักจับข้อมูล และการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต เนื่องจากกุญแจสาธารณะและกุญแจส่วนตัวไม่สามารถใช้แทนกันได้ ซึ่งทำให้การโจมตีในช่องทางที่เปิดเผย เช่น การสื่อสารผ่านอินเทอร์เน็ต มีความเสี่ยงน้อยลงมาก

การใช้ Public Key ใน การเข้ารหัสช่วยให้สามารถเปิดเผยข้อมูลสำคัญอย่าง KRF ให้แก่ทุกฝ่ายได้โดยไม่ต้องกลัวข้อมูลจะถูกเปิดเผยอย่างไม่ถูกต้อง และยังคงรับประกันได้ว่าผู้ที่สามารถอ่านครหัสได้คือผู้ที่ถือ Private Key ที่ตรงกันเท่านั้น



รูปที่ 2.10 การเข้ารหัสแบบกุญแจสมมาตร(Asymmetric Key Cryptography)

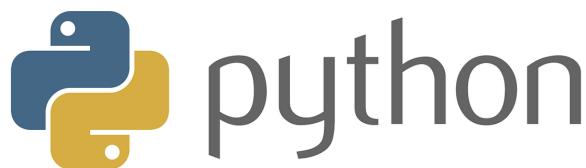
## 2.13 ภาษา Python

Python เป็นภาษาการเขียนโปรแกรมที่มีความยืดหยุ่นและทรงพลัง ด้วยไลบรารีที่หลากหลายที่ช่วยให้สามารถพัฒนาระบบที่มีความซับซ้อนและมีความปลอดภัยสูงได้ ภาษา Python รองรับการพัฒนาในหลายด้าน เช่น การพัฒนาเว็บแอปพลิเคชัน การเข้ารหัสข้อมูล และการจัดการข้อมูลที่สำคัญ จึงเป็นตัวเลือกที่เหมาะสมในการพัฒนาโครงการที่ต้องการรักษาความปลอดภัยสูง

ในโครงการนี้ Python ถูกใช้เป็นภาษาหลักสำหรับการพัฒนา โดยไลบรารีต่าง ๆ เช่น cryptography, Flask, และ requests ถูกใช้ในการจัดการการเข้ารหัสข้อมูล การพัฒนาเซิร์ฟเวอร์ สำหรับการสื่อสารที่ปลอดภัย และการจัดการกับการเชื่อมต่อผ่านเครือข่าย

- cryptography: ใช้สำหรับการจัดการกับกระบวนการเข้ารหัสและจดหมายหัวข้อมูลในระบบรวมถึงการจัดการคีย์และการใช้งานอัลกอริธึมการเข้ารหัสที่ปลอดภัย
- Flask: ใช้ในการพัฒนาเว็บแอปพลิเคชันสำหรับการจัดการเซิร์ฟเวอร์ การส่งข้อมูล และการตรวจสอบการเชื่อมต่ออย่างปลอดภัย
- requests: ใช้สำหรับการจัดการการส่งข้อมูลผ่านเครือข่าย ซึ่งรวมถึงการส่งข้อมูลที่เข้ารหัสไปยังผู้รับ

การใช้ Python ในโครงการนี้ช่วยให้การพัฒนาเป็นไปได้อย่างรวดเร็ว และมีเครื่องมือที่เหมาะสมสำหรับการสร้างระบบที่มีความปลอดภัยและมีประสิทธิภาพในการจัดการกับข้อมูลที่สำคัญ เช่น การเข้ารหัสและการตรวจสอบสิทธิ์ด้วย PKI และ PKCE



รูปที่ 2.11 ภาษา Python

ที่มา: [https://miro.medium.com/v2/resize:fit:1400/1\\*m0H6-tUbW6grMlezlb52yw.png](https://miro.medium.com/v2/resize:fit:1400/1*m0H6-tUbW6grMlezlb52yw.png)

## 2.14 Docker

Docker เป็นแพลตฟอร์มที่ช่วยให้นักพัฒนาสามารถสร้าง ทดสอบ และนำแอปพลิเคชันไปใช้งานในรูปแบบของคอนเทนเนอร์ที่แยกจากกัน ซึ่งทำให้สามารถจำลองสภาพแวดล้อมของระบบได้

อย่างมีประสิทธิภาพ โดยไม่ต้องกังวลเกี่ยวกับปัญหาการตั้งค่าสภาพแวดล้อมที่ซับซ้อน การใช้งาน Docker ช่วยให้การจัดการแอปพลิเคชันในรูปแบบคอนเทนเนอร์เป็นไปอย่างราบรื่นและมีประสิทธิภาพ

ในโครงการนี้ Docker ถูกใช้เพื่อจำลองสภาพแวดล้อมที่แยกจากกันสำหรับแต่ละหน่วยงานในระบบ เช่น Sender, Receiver, KRC, และ KRA โดยแต่ละหน่วยงานทำงานในคอนเทนเนอร์แยกจากกัน ทำให้สามารถทดสอบการทำงานของระบบในสภาพแวดล้อมที่แยกออกจากกันได้อย่างมีประสิทธิภาพและปลอดภัย การแยกการทำงานในคอนเทนเนอร์เหล่านี้ช่วยให้การพัฒนาและทดสอบโครงการเป็นไปได้อย่างราบรื่น โดยที่ไม่ต้องกังวลเกี่ยวกับปัญหาที่อาจเกิดจากการตั้งค่าสภาพแวดล้อมที่ซับซ้อนหรือข้อจำกัดของเครื่องมือในการทดสอบ



รูปที่ 2.12 Docker

ที่มา: [https://upload.wikimedia.org/wikipedia/commons/7/70/Docker\\_logo.png](https://upload.wikimedia.org/wikipedia/commons/7/70/Docker_logo.png)

## 2.15 ไลบรารี Cryptography ใน Python

ไลบรารี Cryptography เป็นเครื่องมือที่ใช้ในการเข้ารหัสและถอดรหัสข้อมูล โดยรองรับทั้งการเข้ารหัสแบบสมมาตรและ非对称加密 ทำให้เป็นไลบรารีที่เหมาะสมสำหรับการจัดการการเข้ารหัสในระบบที่ต้องการความปลอดภัยสูง ไลบรารีนี้มีฟังก์ชันที่ครอบคลุมการสร้างกุญแจ การเข้ารหัสและถอดรหัสข้อมูล การเชื่อมต่อท้อง และการจัดการใบรับรองต่างๆ ที่สามารถใช้งานในระบบที่ต้องการความปลอดภัย

ในโครงการนี้ ไลบรารี Cryptography ถูกใช้ในการจัดการกับกุญแจที่ใช้ในการเข้ารหัสข้อมูลในระบบ เช่น การเข้ารหัสและถอดรหัส AES Key ที่ใช้ในการเข้ารหัส KRF (Key Recovery File) รวมถึงการจัดการ PKI (Public Key Infrastructure) ที่ใช้ในกระบวนการตรวจสอบสิทธิ์และการคุ้มครอง

นอกจากนี้ ไลบรารีนี้ยังรองรับการสร้างและจัดการกุญแจแบบ非对称加密 และสามารถสร้างกุญแจสาธารณะและกุญแจส่วนตัวที่ใช้ในระบบได้อย่างปลอดภัย การใช้ไลบรารีนี้ช่วยให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพ โดยการเข้ารหัสข้อมูลและการแลกเปลี่ยนกุญแจระหว่าง

Sender, Receiver, KRC, และ KRA สามารถทำได้อย่างปลอดภัย เพิ่มความมั่นใจในการจัดการข้อมูลและลดความเสี่ยงจากการโจรตีทางด้านความปลอดภัย

### เหตุผลในการเลือกใช้

cryptography ถูกเลือกใช้ในโครงการนี้เนื่องจากมีคุณสมบัติที่ครบครันและปลอดภัยในการจัดการการเข้ารหัสและการจัดการกุญแจในระบบที่ซับซ้อน เช่น การสร้างกุญแจ AES สำหรับการเข้ารหัส KRF และการสร้างกุญแจอสมมาตรสำหรับการจัดการกับการตรวจสอบสิทธิ์ที่ปลอดภัยกว่า

เมื่อเปรียบเทียบกับไลบรารีอื่น ๆ เช่น PyCryptodome หรือ M2Crypto ซึ่งก็สามารถทำการเข้ารหัสได้เหมือนกัน cryptography มีความได้เปรียบในด้านความง่ายในการใช้งานและการรักษาความปลอดภัยที่ดีที่สุด ด้วยการสนับสนุนจากองค์กรที่เชื่อถือได้และมีการอัปเดตอย่างสม่ำเสมอ

## 2.16 ไลบรารี Flask ใน Python

Flask เป็นเฟรมเวิร์กเว็บที่เบาและใช้งานง่าย เหมาะสมสำหรับการพัฒนาเว็บแอปพลิเคชันขนาดเล็กถึงกลาง โดยสามารถขยายและปรับแต่งให้เข้ากับความต้องการของระบบได้อย่างสะดวกและรวดเร็ว ในโครงการนี้ Flask ทำหน้าที่เป็นเซิร์ฟเวอร์ที่จัดการการรับส่งข้อมูลระหว่างฝ่ายต่าง ๆ เช่น Sender, Receiver, KRC, และ KRA โดยใช้ HTTP ในการส่งข้อมูลที่เข้ารหัสไปยังปลายทางอย่างปลอดภัย

การใช้ Flask ในโครงการนี้ช่วยในการจัดการเส้นทางการเข้ามต่อ (Endpoints) และการจัดการข้อมูลต่าง ๆ ที่อาจเกิดขึ้นในการส่งข้อมูล การใช้ HTTP ทำให้การรับส่งข้อมูลเป็นไปอย่างมีประสิทธิภาพ และ Flask สามารถจัดการกับการส่งข้อมูลที่เข้ารหัสได้อย่างปลอดภัยระหว่างหน่วยงานในระบบ เช่น การตรวจสอบสิทธิ์ในการเข้าถึงข้อมูลที่เข้ารหัสและการจัดการกับคำขอจากผู้ใช้หรือระบบอื่น ๆ ผ่าน API ของ Flask ได้อย่างเหมาะสม

### เหตุผลในการเลือกใช้

Flask ถูกเลือกในโครงการนี้เนื่องจากมีคุณสมบัติที่ตอบโจทย์ความต้องการของระบบที่ต้องการความยืดหยุ่นและการปรับแต่งที่ง่าย ในกรณีที่ต้องพัฒนาเว็บเซิร์ฟเวอร์หรือระบบที่เกี่ยวข้องกับการรับส่งข้อมูลเข้ารหัสแบบ HTTPS โดยไม่ต้องการความซับซ้อนเกินไป

เปรียบเทียบกับฟอร์มเว็บก็ว่าอีกนิด ๆ เช่น Django ซึ่งเป็นฟอร์มเว็บที่มีฟังก์ชันมากมายและเหมาะสมกับโปรเจคขนาดใหญ่, Flask ได้รับเลือกเนื่องจากมีขนาดเบาและง่ายต่อการเรียนรู้ ซึ่งช่วยให้สามารถพัฒนาและทดสอบได้รวดเร็วโดยไม่ต้องตั้งค่ามากมาย

## 2.17 ไลบรารี Requests ใน Python

requests เป็นไลบรารีที่ช่วยให้การส่ง HTTP/HTTPS requests ง่ายและสะดวก โดยไม่ต้องเขียนโค้ดที่ซับซ้อนหรือจัดการกับการตั้งค่าต่าง ๆ ด้วยตัวเอง

ในโครงการนี้ requests ถูกใช้เพื่อส่งข้อมูลที่เข้ารหัสไปยัง Receiver หรือหน่วยงานอื่น ๆ ในระบบ เช่น KRC หรือ KRA โดยที่ไลบรารีนี้ทำให้การสื่อสารผ่าน HTTP มีความปลอดภัยและใช้งานง่าย ไม่ว่าจะเป็นการส่งข้อมูล JSON หรือการจัดการกับการตอบกลับจากเซิร์ฟเวอร์

### เหตุผลในการเลือกใช้

requests ถูกเลือกใช้ในโครงการนี้ เพราะมี API ที่ใช้งานง่ายและการตั้งค่าที่ไม่ซับซ้อน ทำให้การส่งข้อมูลเข้ารหัสผ่าน HTTP สามารถทำได้อย่างปลอดภัยและรวดเร็ว ซึ่งช่วยให้การสื่อสารระหว่าง Sender และ Receiver หรือ KRC และ KRA ในระบบทำได้ง่ายดาย

หากเปรียบเทียบกับ ไลบรารีอื่น ๆ เช่น urllib ที่เป็นไลบรารีมาตรฐานของ Python, requests มีฟังก์ชันที่ครบครันมากกว่าและรองรับการทำงานต่าง ๆ ได้มากกว่า เช่น การจัดการกับเวลาหมดอายุของคำขอ (Timeouts) หรือการส่งข้อมูล JSON อย่างสะดวก ทำให้เป็นทางเลือกที่เหมาะสมในโปรเจคนี้

## 2.18 ไลบรารี Socket ใน Python

socket เป็นไลบรารีใน Python ที่ช่วยให้สามารถสร้างการเชื่อมต่อระหว่างเซิร์ฟเวอร์และลูกค้า (Client-Server) โดยใช้โปรโตคอล TCP หรือ UDP สำหรับการส่งข้อมูลระหว่างคอมพิวเตอร์

ในโครงการนี้ socket ถูกใช้สำหรับการสร้างการเชื่อมต่อระหว่าง Sender และ Receiver ในระบบ เพื่อส่งข้อมูลที่เข้ารหัสผ่าน TCP/IP protocol โดยไลบรารีนี้ช่วยให้การสื่อสารระหว่างฝ่ายต่าง ๆ ในระบบสามารถทำได้อย่างมีประสิทธิภาพและปลอดภัย

การใช้ socket ในการเชื่อมต่อระหว่างเซิร์ฟเวอร์และผู้ใช้งานช่วยให้ระบบสามารถส่งข้อมูลขนาดใหญ่ได้อย่างรวดเร็วและปลอดภัย โดยการควบคุมการเชื่อมต่อแบบเต็มรูปแบบและจัดการการส่งข้อมูลแบบเป็นลำดับ

### เหตุผลในการเลือกใช้

socket ถูกเลือกในโครงการนี้ เพราะมันเป็นไลบรารีที่รองรับการเชื่อมต่อระหว่างเครื่องได้อย่างสะดวกและปลอดภัย โดยใช้โปรโตคอล TCP/IP ซึ่งเหมาะสมกับการส่งข้อมูลที่ต้องการ การเชื่อมต่อที่เชื่อถือได้และมั่นคง การใช้ socket ช่วยให้เราควบคุมกระบวนการส่งข้อมูลได้อย่างเต็มที่โดยไม่ต้องพึ่งพาไลบรารีภายนอกเพิ่มเติม

ถึงแม้ว่าไลบรารีอื่น ๆ เช่น Twisted หรือ asyncio ที่สามารถทำการสื่อสารผ่านเครือข่ายได้แต่ socket เป็นตัวเลือกที่ดีที่สุดในกรณีนี้ เนื่องจากมันเป็นไลบรารีที่อยู่ในมาตรฐานของ Python และทำให้สามารถจัดการกับการเชื่อมต่อแบบเต็มรูปแบบได้โดยไม่ต้องพึ่งพาไลบรารีที่ซับซ้อนเกินไป

## บทที่ 3

### วิธีดำเนินโครงการ

ในบทนี้จะกล่าวถึงกระบวนการและขั้นตอนในการดำเนินโครงการเพื่อพัฒนาระบบ Authenticated Secure Key Recovery System ซึ่งได้รับการออกแบบให้สามารถคืนกุญแจได้อย่างปลอดภัยผ่านการตรวจสอบสิทธิ์ด้วย PKCE-like challenge และใช้แนวทางที่ช่วยเพิ่มความมั่นคง ปลอดภัยต่อข้อมูล โดยระบบนี้พัฒนาบนโครงสร้างที่รองรับการทำงานร่วมกันระหว่างหลายส่วนประกอบ โดยมีองค์ประกอบที่เกี่ยวข้องในระบบดังนี้ (1) ผู้ส่ง (Sender), (2) ผู้รับ (Receiver), (3) ศูนย์กลางการคืนกุญแจ (Key Recovery Center: KRC) และ (4) เอเจนต์ในการคืนกุญแจ (Key Recovery Agent: KRA)

ระบบทำงานอยู่บนสภาพแวดล้อมจำลองที่ใช้ Docker เพื่อให้สามารถแยกกระบวนการทำงานของแต่ละองค์ประกอบได้อย่างชัดเจนและมีประสิทธิภาพ โดยในการเริ่มต้นกระบวนการทำงานทุกองค์ประกอบของระบบจะต้องมีการคุณแจคุ่ของตัวเองคือ กุญแจส่วนตัว private key (Kr) และ กุญแจสาธารณะ (Ku) เพื่อใช้ในการสื่อสารระหว่างกันอย่างมั่นคงปลอดภัย บนโครงสร้างพื้นฐานของกุญแจสาธารณะ (Public Key Infrastructure: PKI)

#### 3.1 การศึกษาและวิเคราะห์ข้อมูลเกี่ยวกับโครงการ

การศึกษาและวิเคราะห์ข้อมูลเกี่ยวกับโครงการนี้มีวัตถุประสงค์เพื่อทำความเข้าใจถึง ความปลอดภัย, ประสิทธิภาพ, และ ความสามารถในการขยายระบบ เพื่อให้มั่นใจว่าการคืนกุญแจสามารถดำเนินการได้อย่างปลอดภัยและรวดเร็วในกรณีที่เกิดสถานการณ์ฉุกเฉิน เช่น เอเจนต์บางส่วนล้ม หรือ มีความจำเป็นต้องเข้าถึงข้อมูลที่ถูกเข้ารหัส โดยมีรายละเอียดการวิเคราะห์ดังต่อไปนี้

##### 3.1.1 การวิเคราะห์ความปลอดภัยของระบบคืนกุญแจ

- ใช้ PKCE-like challenge เป็นกลไกการตรวจสอบสิทธิ์เพื่อป้องกัน replay attack และ man-in-the-middle attack
- มีการเข้ารหัสหลายชั้น (multi-layer encryption) เพื่อป้องกัน Key Recovery Field (KRF) ที่สั่งระหว่าง Sender, Receiver, KRC และ KRA

- ใช้ Public Key Infrastructure (PKI) ในการจัดการคู่กุญแจเพื่อให้มั่นใจว่าการแลกเปลี่ยนข้อมูลเป็นไปอย่างปลอดภัย

### 3.1.2 การวิเคราะห์กระบวนการตรวจสอบและยืนยันตัวตน

- ใช้ PKCE-like challenge ในกระบวนการตรวจสอบสิทธิ์ของ Receiver และ KRA เพื่อบังคับการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต
- เปรียบเทียบกับ OAuth 2.0 PKCE Flow และปรับปรุงให้เหมาะสมกับการคุ้มครองคุณเจในระบบ

### 3.1.3 การประเมินประสิทธิภาพของระบบในการคุ้มครองคุณเจเมื่อเกิดการล้มของ KRA

- มีการจำลองสถานการณ์ KRA บางส่วนล่ม และวิเคราะห์ว่าสามารถรวบรวมคุณเจที่เหลืออยู่เพียงพอสำหรับการถอดรหัสข้อมูลได้หรือไม่
- ระบบถูกออกแบบให้สามารถรองรับ การเชื่อมต่อใหม่ของ KRA โดยไม่ต้องเริ่มกระบวนการทั้งหมดใหม่

### 3.1.4 การศึกษาความสามารถในการขยายระบบ (Scalability)

- พิจารณาความสามารถของระบบในการรองรับ จำนวน KRA ที่เพิ่มขึ้น
- ออกแบบ KRC ให้สามารถกระจายโหลดได้อัตโนมัติ
- ใช้โครงสร้างเครือข่ายที่รองรับการทำงานในระบบขนาดใหญ่ โดยใช้ Docker Network

### 3.1.5 การศึกษาและทดสอบความเร็วในการส่งข้อมูล

- ทดสอบประสิทธิภาพของการส่งข้อมูล KRF ในสภาวะต่าง ๆ รวมถึงการเข้ารหัสและถอดรหัส
- ใช้ Flask, Requests และ Socket ในการทดสอบการส่งข้อมูลผ่าน HTTP และ TCP เพื่อเปรียบเทียบ ความเร็วและความน่าเชื่อถือ

โดยสรุป การศึกษาและวิเคราะห์นี้ครอบคลุม การวิเคราะห์ความปลอดภัย, ประสิทธิภาพของระบบและความสามารถในการขยายตัว เพื่อให้โครงงานสามารถรองรับสถานการณ์จริงที่ต้องการระบบคุ้มครองคุณเจที่มี ความปลอดภัยสูง, มีประสิทธิภาพ และสามารถปรับขยายได้

## 3.2 ภาพรวมโครงงาน

โครงงานนี้พัฒนาระบบ Authenticated Secure Key Recovery System ซึ่งเป็นระบบที่ต่อจาก Secure and Flexible Multiple-Agent Key Recovery System (SFM-KRS) โดยมีการเพิ่ม

การตรวจสอบสิทธิ์แบบ PKCE-like challenge เพื่อเพิ่มระดับความปลอดภัยและป้องกันการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

### 3.2.1 องค์ประกอบหลักของระบบ

ระบบประกอบด้วย 4 องค์ประกอบหลัก ซึ่งแต่ละองค์ประกอบมีหน้าที่เฉพาะดังนี้

#### 3.2.1.1 Sender

- ทำหน้าที่ เข้ารหัสข้อมูล ก่อนส่งไปยัง Receiver
- สร้างและส่ง Key Recovery Field (KRF) ไปยัง Receiver (Requester)

#### 3.2.1.2 Receiver

- ทำหน้าที่ จดครหัสข้อมูล และร้องขอการกู้คืนข้อมูล Session key ที่หายหรือชำรุด
- ส่งคำขอคืนไปยัง Key Recovery Center (KRC)

#### 3.2.1.3 Key Recovery Center (KRC)

- ตรวจสอบสิทธิ์ของ Receiver ผ่าน Request และยืนยันตัวตนด้วย PKCE-like challenge
- ประสานงานกับ KRA เพื่อร่วมรวม KRF และคืนกุญแจให้กับ Receiver

#### 3.2.1.4 Key Recovery Agents (KRA)

- ทำหน้าที่ ให้บริการ KRF ตามสิทธิ์ของ Receiver
- ส่ง KRF กลับไปยัง KRC เมื่อได้รับคำร้องขอที่ได้รับการตรวจสอบสิทธิ์แล้ว

### 3.2.2 สภาพแวดล้อมของระบบ

- แยกแต่ละองค์ประกอบให้อยู่ใน Docker Containers เพื่อให้สามารถจำลองการทำงานจริงได้อย่างมีประสิทธิภาพ
- กำหนด IP และ Port ของแต่ละ Container เพื่อจำลองการทำงานในเครือข่ายที่เสมือนจริง
- รองรับการขยายตัวของระบบ โดยสามารถเพิ่มจำนวน KRA ได้ตามความต้องการ ระบบนี้ถูกออกแบบมาเพื่อให้ การกู้คืนกุญแจสามารถทำได้อย่างปลอดภัยและมีประสิทธิภาพ โดยมีการเพิ่ม PKCE-like challenge เพื่อยืนยันตัวตนของ Receiver และใช้โครงสร้างที่สามารถทำงานบนเครือข่ายที่มีการแยกส่วน (isolation) ผ่าน Docker ซึ่งช่วยให้สามารถควบคุม, ทดสอบ, และปรับขยายระบบ ได้ง่ายขึ้น

### 3.3 การออกแบบระบบ

การออกแบบระบบเป็นขั้นตอนสำคัญที่ช่วยให้เข้าใจภาพรวมของโครงสร้างและฟังก์ชันการทำงานของ Authenticated Secure Key Recovery System เพื่อให้สามารถพัฒนาและปรับปรุงระบบให้มีประสิทธิภาพและความปลอดภัยสูงสุด

#### 3.3.1 การออกแบบโครงสร้างระบบ (System Architecture)

โครงสร้างของระบบถูกออกแบบให้ใช้ Client-Server Model โดยแต่ละองค์ประกอบของระบบมีบทบาทและหน้าที่เฉพาะ เพื่อให้สามารถทำงานร่วมกันได้อย่างปลอดภัยและรองรับการขยายตัวของระบบ

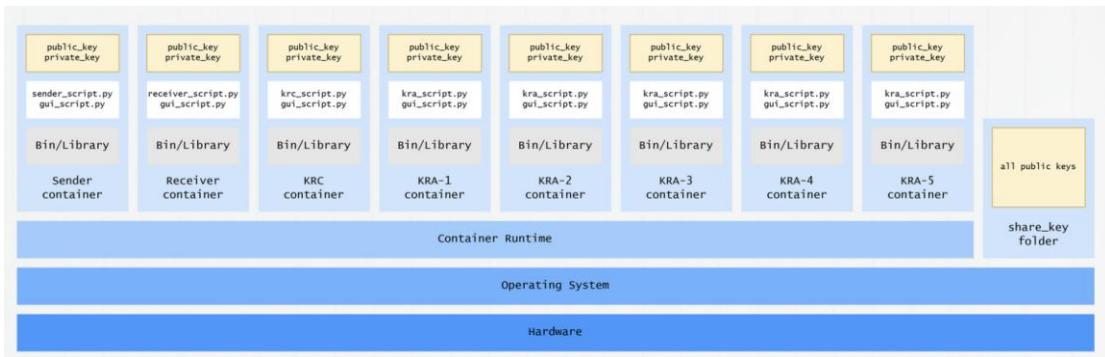
##### 3.3.1.1 องค์ประกอบหลักของระบบ

- Sender: ทำหน้าที่เข้ารหัสข้อมูลและส่ง Key Recovery Fragments (KRF) ไปยัง Receiver
- Receiver: ทำหน้าที่รับขอคืนข้อมูลและยืนยันตัวตนผ่าน KRC
- Key Recovery Center (KRC): ตรวจสอบสิทธิ์ของ Receiver และประสานงานกับ KRA
- Key Recovery Agents (KRA): จัดการ KRF และให้บริการเมื่อได้รับคำขอจาก KRC

##### 3.3.1.2 เทคโนโลยีที่ใช้ในโครงสร้างระบบ

- ใช้ Docker Containers สำหรับแต่ละองค์ประกอบ (Sender, Receiver, KRC, KRA) เพื่อจำลองสภาพแวดล้อมที่ปลอดภัยและรองรับการขยายตัวของระบบ
- ใช้ Flask และ Socket เป็น Backend API สำหรับการติดต่อระหว่างองค์ประกอบต่าง ๆ
- ใช้ RSA (Public Key) สำหรับเข้ารหัสกุญแจ และ AES สำหรับเข้ารหัสข้อมูลภายใน Session
- ใช้ Docker Network เพื่อให้ Containers สามารถสื่อสารกันได้อย่างปลอดภัย

โดยโครงสร้างของระบบ Docker container จะมีรูปแบบดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างของ Docker container

### 3.3.2 การออกแบบกระบวนการทำงานของระบบ (Workflow Design)

กระบวนการทำงานของระบบถูกออกแบบโดยใช้ Activity Diagram เพื่อแสดงลำดับของการทำงานในแต่ละขั้นตอนหลัก

#### 3.3.2.1 ลำดับขั้นตอนการทำงานหลัก

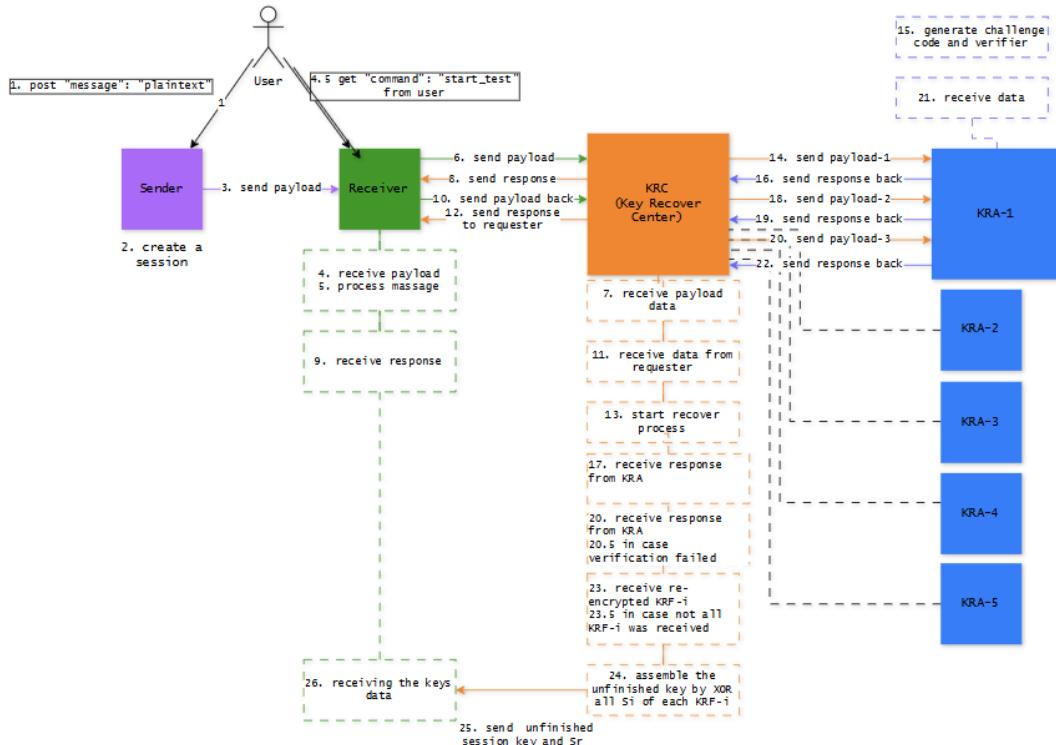
1. Receiver ส่งคำขอคืนกุญแจไปยัง KRC
2. KRC ตรวจสอบสิทธิ์ของ Receiver ด้วย Request information และยืนยันตัวตนด้วยPKCE-like Challenge
3. KRC ติดต่อกับ KRA เพื่อร้องขอการ re-encrypt KRF โดยมีการยืนยันตัวตนของ KRAด้วยPKCE-like Challenge
4. KRA ตรวจสอบคำขอ ทำการยืนยันตัวตนและส่ง re-encrypt KRF<sub>i</sub> กลับไปยัง KRC
5. KRC รวม KRF<sub>i</sub> ที่ได้รับและสร้างกุญแจที่เกือบสมบูรณ์และ Sr ให้กับ Receiver
6. Receiver ทำการรวม และสร้างกุญแจที่สมบูรณ์เพื่อใช้งาน ต่อไป

#### 3.3.2.2 มาตรการรักษาความปลอดภัยใน Workflow

- ตรวจสอบและยืนยันตัวตนทุกครั้งก่อนอนุญาตให้เข้าถึงข้อมูล
- บันทึกข้อมูลและการทำงานในแต่ละขั้นตอนเพื่อป้องกันการละเมิดสิทธิ์
- ใช้การเข้ารหัสหลายชั้นเพื่อป้องกันการดักฟังและการโจมตีแบบ Man-in-the-Middle

การออกแบบระบบนี้ช่วยให้ Authenticated Secure Key Recovery System มีความปลอดภัย, มีโครงสร้างที่สามารถขยายตัวได้ง่าย และสามารถทำงานได้อย่างมีประสิทธิภาพในสภาพแวดล้อมที่ต้องการความปลอดภัยสูง

โดยขั้นตอนการทำงานของระบบโดยภาพรวมจะมีขั้นตอนของระบบดังรูปที่ 3.2



รูปที่ 3.2 ภาพรวมการทำงานของระบบ AS-KRS

### 3.4 การสร้างจำลองโดยใช้ Docker

ระบบถูกจำลองโดยใช้ Docker เพื่อสร้างสภาพแวดล้อมเสมือนจริงที่ช่วยให้สามารถทดสอบและควบคุมการทำงานของแต่ละองค์ประกอบของระบบได้อย่างอิสระ โดยแต่ละองค์ประกอบของระบบ (Sender, Receiver, KRC และ KRA) จะทำงานอยู่ภายใต้ Container แยกจากกัน ซึ่งช่วยเพิ่มความปลอดภัยและทำให้การพัฒนาเป็นระบบมากขึ้น

#### 3.4.1 ขั้นตอนการสร้างจำลองระบบ

##### 3.4.1.1 สร้าง Dockerfile สำหรับแต่ละองค์ประกอบ

แต่ละองค์ประกอบของระบบจะมี Dockerfile เฉพาะที่กำหนดสภาพแวดล้อมของตัวเอง เช่น

- กำหนด Python Version และ ติดตั้งไลบรารีที่จำเป็น เช่น Flask, Cryptography
- ตั้งค่าตัวแปรสภาพแวดล้อม (Environment Variables) เพื่อจัดการการทำงานของ Container
- กำหนด EntryPoint เพื่อรันเซิร์ฟเวอร์หรือกระบวนการที่เกี่ยวข้องโดยอัตโนมัติ

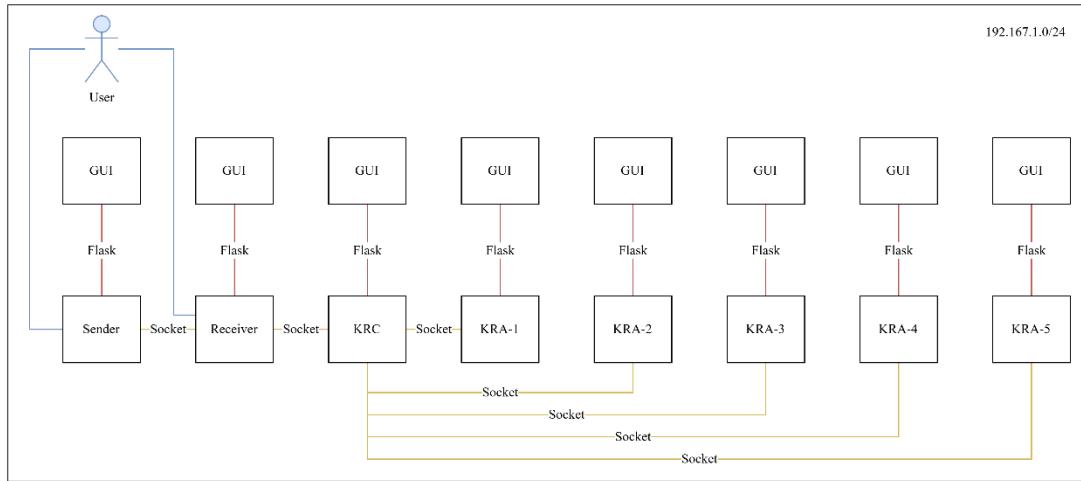
3.4.1.2 กำหนด Network และจัดการ Containers ด้วย Docker Compose ใช้ Docker Compose เพื่อกำหนดการเชื่อมต่อระหว่าง Containers และจัดการระบบได้ง่ายขึ้น โดย

- สร้าง Docker Network เพื่อให้แต่ละ Container สามารถสื่อสารกันได้ภายในเครือข่ายเดียวกัน
- กำหนด IP Address และ Port สำหรับแต่ละ Container เพื่อให้การติดต่อระหว่างองค์ประกอบของระบบมีความชัดเจน
- ใช้ Compose File (docker-compose.yml) เพื่อกำหนดลำดับการเริ่มต้นของแต่ละองค์ประกอบ

3.4.2 ประโยชน์ของการใช้ Docker ใน การพัฒนา การใช้ Docker ทำให้สามารถ

- ทดสอบระบบได้ในเครือข่ายเดียวจริง โดยเลียนแบบสถานการณ์ใช้งานจริง
- แยกส่วนการทำงานของแต่ละองค์ประกอบ ทำให้มั่นใจได้ว่าแต่ละ Container ทำงานได้อย่างอิสระและปลอดภัย
- พัฒนาและปรับปรุงระบบได้ง่ายขึ้น เนื่องจากสามารถอัปเดตโค้ดแต่ละส่วนได้โดยไม่กระทบกับส่วนอื่น ๆ
- ควบคุมทรัพยากรและการใช้งานของระบบ ได้อย่างมีประสิทธิภาพ ลดความซับซ้อนในการจัดการสิ่งแวดล้อมการพัฒนา

การใช้ Docker ทำให้การพัฒนาและทดสอบ Authenticated Secure Key Recovery System เป็นไปอย่างมีประสิทธิภาพ สามารถจำลองสถานการณ์ที่แตกต่างกันได้อย่างยืดหยุ่น และช่วยเพิ่มความปลอดภัยของระบบ



รูปที่ 3.3 Docker Network Simulation

### 3.5 การสร้าง PKI Structure

การสร้าง PKI (Public Key Infrastructure) Structure เป็นขั้นตอนสำคัญที่ช่วยรับรองความปลอดภัยของระบบ Authenticated Secure Key Recovery System โดยใช้ public key และ private key ในการเข้ารหัสและตรวจสอบความถูกต้องของแต่ละองค์ประกอบในระบบ กระบวนการนี้ช่วยป้องกันการดักฟังข้อมูลและตรวจสอบความถูกต้องของคุณสื่อสาร ได้อย่างปลอดภัย

#### 3.5.1 การสร้างคู่กุญแจ RSA

ระบบใช้ RSA key pair ขนาด 2048 บิต สำหรับแต่ละองค์ประกอบของระบบ ได้แก่ Sender, Receiver, KRC และ KRA โดยใช้ไลบรารี cryptography.hazmat.primitives.asymmetric.rsa ใน การสร้างคู่กุญแจ

- Sender: ใช้ public key อีกฝ่าย เพื่อเข้ารหัส session key และข้อมูลใน KRF ก่อน ส่งไปยัง Receiver
- Receiver: ใช้ private key ของตนเอง ในการถอดรหัส session key หลังจากได้รับ จาก Sender หรือ KRC
- KRC และ KRA: ใช้ public-private key pair ในการตรวจสอบสิทธิ์และจัดการ Key Recovery Fragments (KRFs)

#### 3.5.2 การจัดเก็บและปกป้อง Private Key เพื่อความปลอดภัย Private Key ของแต่ละ องค์ประกอบจะถูกเก็บและป้องกันดังนี้

- จัดเก็บใน Container ของตนเอง และไม่แชร์ให้กับ Container อื่น

- เข้ารหัสไฟล์ Private Key ด้วย AES-256 ก่อนทำการจัดเก็บ
- กำหนดสิทธิ์การเข้าถึง เลพาะ Container ที่เกี่ยวข้อง เช่น
  - Sender สามารถเข้าถึงเฉพาะ Private Key ของตนเอง
  - Receiver สามารถเข้าถึง Private Key สำหรับอุดหนัติที่ได้รับ
  - KRC มีสิทธิ์เข้าถึง Public Key ของทุกฝ่าย แต่ไม่สามารถเข้าถึง Private Key ของ KRA ได้

### 3.5.3 การเผยแพร่ Public Key

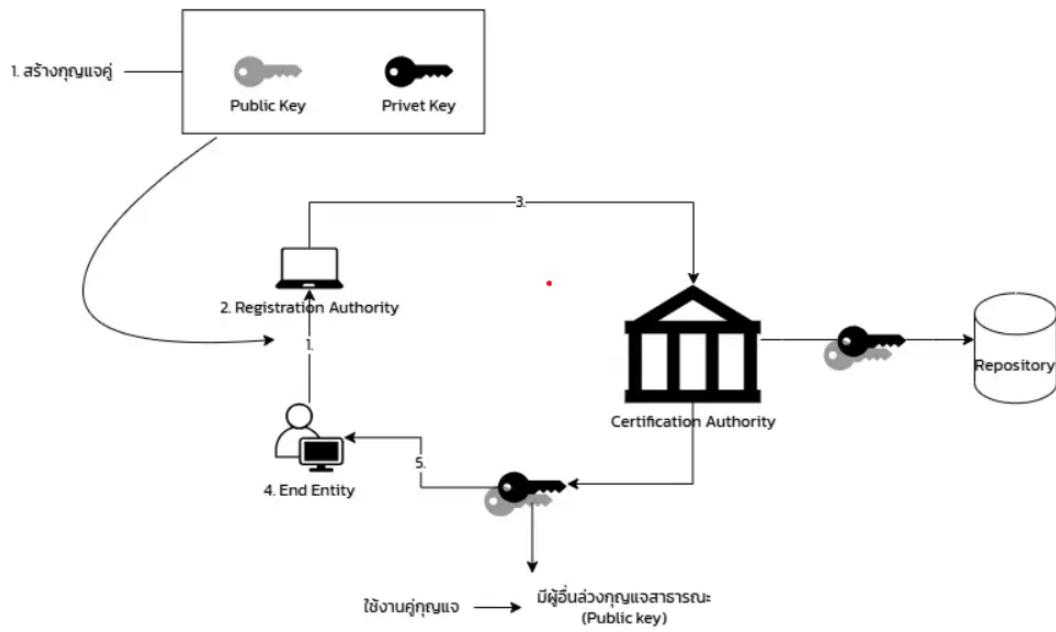
Public Key ของแต่ละองค์ประกอบ จะถูกแชร์ไปยังไฟล์เดอร์ shared ตามที่กำหนดโดยใช้ไฟล์ JSON หรือระบบ Public Key Directory เพื่อให้สามารถแลกเปลี่ยน Public Key ได้อย่างปลอดภัย

- Sender และ Receiver จะใช้ Public Key ของกันและกัน ในการเข้ารหัสข้อมูลที่ต้องการสื่อสาร
- KRC และ KRA ต่างก็สร้างและจัดเก็บ Public Key ของตนเอง โดยฝ่ายที่ต้องการใช้งานสามารถขอ Public Key จากไฟล์เดอร์ shared ได้
- ทุกฝ่ายเข้าถึง Public Key ได้จาก shared folder เพื่อใช้ในการเข้ารหัสและยืนยันตัวตน โดยไม่ต้องผ่านตัวกลาง

การสร้าง PKI Structure เป็นองค์ประกอบสำคัญของระบบที่ช่วย

- เพิ่มความมั่นใจในการเข้ารหัสข้อมูล
- ป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต
- ทำให้แน่ใจว่าเฉพาะผู้ที่มีสิทธิ์เท่านั้นที่สามารถอุดหนัติข้อมูลได้

กระบวนการนี้ช่วยเพิ่มความปลอดภัยของระบบ Authenticated Secure Key Recovery System ให้แข็งแกร่งและลดความเสี่ยงจากการโจมตีที่เกี่ยวข้องกับการปลอมแปลงข้อมูล



รูปที่ 3.4 PKI Struture

### 3.6 การทำงานของ Sender

Sender มีหน้าที่ในการเข้ารหัสข้อความและการสร้าง KRF (Key Recovery Field) ซึ่งประกอบไปด้วยการแบ่งกุญแจหลักเป็นกุญแจย่อย เพื่อใช้ในกระบวนการกรุ๊ปกุญแจเมื่อ Receiver ต้องการ

#### 3.6.1 การรับข้อความ plaintext เพื่อใช้ส่งต่อไปยัง Receiver

ตารางที่ 3.1 รายละเอียดของ API สำหรับรับข้อความ Plaintext

Method	POST
Host	http://<SENDER_IP>:5000/send_message
Content-type	application/json
Body	{ "message": "<plaintext_message>", "receiver": "<optional_receiver_ip>" }
Response	{ "response": "<Receiver's response>" }

จากตารางที่ 3.1 ใช้วิธี (Method) แบบ Post ผ่านเส้นทาง (Path) /send\_message โดยใช้โพร็อกซี่ หรือโดเมนคือ http://<SENDER\_IP>:5000 โดย sender IP มีการกำหนดไว้โดย docker-compose.yml ไว้ที่ 192.168.1.11 การส่งชุดข้อมูลอยู่ในรูปแบบ

application/json เป็นรูปแบบที่สามารถส่งไฟล์ไปสู่บริการได้ ซึ่งการส่งจะส่งพร้อมพารามิเตอร์และไฟล์โดยใช้คีย์ชื่อว่า message และข้อมูลที่รับค่าเป็นข้อมูล plaintext ของผู้ใช้เมื่อสำเร็จจะได้รับการตอบกลับเป็นรูปแบบ JSON โดย

จะแสดงสถานะของ response

### 3.6.2 การเข้ารหัสข้อมูล

- Sender ใช้ session key ในการเข้ารหัสข้อมูลหลัก ซึ่งใช้ในการต่อสารกับ Receiver
- Sender ใช้ Public Key ของ Receiver ในการเข้ารหัส session key และข้อมูลกุญแจ บอย Sr ก่อนใส่ไว้ใน KRF สำหรับ Receiver
- Sender สร้างและใช้กุญแจ AES ในการเข้ารหัส KRF ที่ใช้ในการถือคืนกุญแจ
- Sender ใช้ Public Key ของ KRC ใช้ในการเข้ารหัสของข้อมูล TT<sub>i</sub> ทั้งหมดและ Other Information ก่อนที่จะใส่ไว้ใน KRF และเข้ารหัสกุญแจ AES ที่ใช้ในการถอดรหัส KRF เพื่อป้องกันการเข้าถึงโดยผู้ที่ไม่ได้รับอนุญาต
- Sender ใช้ Public Key ของแต่ละ KRA<sub>i</sub> สำหรับเข้ารหัสข้อมูล KRF<sub>i</sub> ก่อนใส่ไว้ใน KRF เพื่อให้แต่ละ KRA<sub>i</sub> ถอดรหัสของตนเองได้เท่านั้น

### 3.6.3 การสร้าง KRF

Sender สร้าง KRF (Key Recovery Field) โดยการแบ่ง session key ที่ใช้เข้ารหัสข้อมูลเป็นส่วนย่อย ๆ และกระจายไปยัง KRA (Key Recovery Agents) เพื่อใช้ในการกู้คืนในกรณีที่ Receiver ไม่สามารถดูครหัสข้อมูลได้ โดยจะมีส่วนสุดท้าย Sr ที่ไว้ใช้สำหรับผู้ขอคืนเท่านั้นที่เข้าถึงได้ เพื่อป้องกันการสมรู้ร่วมคิดของ KRA หรือ KRC ที่จะเข้าถึงกุญแจฉบับสมบูรณ์

ภายใน KRF ประกอบด้วย (1) KRF ย่อย ๆ ( $KRF_i$ ), (2) ข้อมูลสำหรับกู้คืนค่า  $S_i$  กรณีที่เอกสารตัวล้ม ( $TT_i$ ), (3) กุญแจย่อของผู้ขอคืน ( $Sr$ ) และ (4) Other Information โดย  $KRF_i$  จะมีจำนวน  $n$  ชิ้น (เมื่อ  $n$  คือจำนวนเอกสารทั้งหมดที่ใช้ในการกู้คืนกุญแจ)

ภายใน  $KRF_i$  จะประกอบด้วย (1) ส่วนประกอบกุญแจลับ ( $S_i$ ) และ (2) Share Group Number (SGN)

แต่ละส่วนประกอบมีความสำคัญดังนี้  $S_i$  คือส่วนประกอบของกุญแจลับของแต่ละ KRA,  $Sr$  คือ ส่วนของกุญแจลับสำหรับผู้ขอคืน (ผู้รับ Receiver หรือหน่วยงานรัฐ Government) SGN เป็นค่าสำหรับการระบุตัวตนของแต่ละ KRA ที่อยู่ในกลุ่มการกู้คืน  $TT_i$  เป็นค่าสำหรับการกู้คืนค่า  $S_i$  ของแต่ละ KRA<sub>i</sub> ในกรณีที่เอกสารตัว  $i$  ได้ ๆ นั้นล้มหรือติดต่อไม่ได้ และ Other Information เก็บค่าอื่น ๆ ที่จำเป็นสำหรับการระบุตัวตนหรือค่าสำหรับการตรวจสอบเพื่อเพิ่มความมั่นคงของระบบ เช่น session ID หรือ timestamp เป็นต้น

#### 3.6.3.1 การ Split Key

Sender ใช้วิธีการ XOR-based Secret Sharing ในการแบ่ง session key ให้เป็นกุญแจย่อย ๆ ที่ต้องการจำนวนขึ้นต่ำในการประกอบกลับเป็น session key เดิม โดยแบ่ง session key ออกเป็น  $n+1$  ส่วนย่อย และกระจายไปยัง KRA  $n$  ราย และส่วนย่อยสุดท้ายสำหรับผู้ขอคืนเอง เพื่อให้สามารถกู้คืนกุญแจได้อย่างปลอดภัยหากต้องการโดยมีหลักการดังต่อไปนี้

- จำนวนตัวเลขจำนวน  $n$  ตัว สำหรับ  $n$  เอกสารที่ใช้ในการกู้คืน เช่น  $S_1, S_2, \dots, S_n$  สำหรับ  $Agent_1, Agent_2, \dots, Agent_n$  ตามลำดับ
- จำนวน  $Sr$  สำหรับ ผู้ขอคืนด้วย  $S_R = S_1 \oplus S_2 \dots \oplus S_n \oplus S_n \oplus K_S$  (โดย  $K_S$  คือกุญแจลับ Session key)
- จำนวนตัวเลข  $R$  สำหรับทุก ๆ ส่วนย่อยของกุญแจลับ ( $R_i$ )

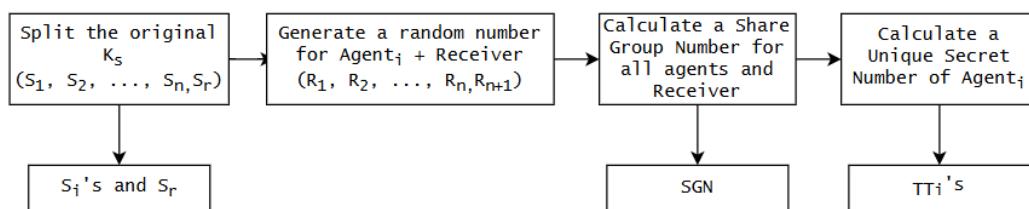
- คำนวณหาค่าความลับ สำหรับกลุ่มการกู้คืนกุญแจ SGN ด้วยการทำ XOR ค่า  $R_i$

$$SGN = R_1 \oplus R_2 \dots \oplus R_n \oplus R_{n-1}$$

3.6.3.2 คำนวณหาค่าพิเศษ (TT) สำหรับทุก ๆ เอเจนต์ ( $TT_i$ ) เพื่อใช้ในการกู้คืนส่วนประกอบของกุญแจในกรณีที่มีเอเจนต์ในกลุ่มการกู้คืนล้ม

$$TT_i = S_1 \oplus SGN$$

โดยการเตรียมค่าส่วนประกอบย่อยของ KRF จะมีขั้นตอนดังรูปที่ 3.5



รูปที่ 3.5 การเตรียมค่าส่วนประกอบย่อยของ KRF

### 3.6.3.3 การเข้ารหัสข้อมูล KRF

หลังจากที่แบ่งกุญแจออกเป็นส่วนย่อย ๆ โดยใช้เทคนิค XOR split แล้ว ในขั้นตอนนี้จะทำการเข้ารหัสข้อมูล KRF เพื่อให้แน่ใจว่ามีการส่งข้อมูลอย่างปลอดภัย โดยใช้กุญแจ AES key เข้ารหัสและเข้ารหัสกุญแจนี้ด้วยกุญแจสาธารณะของ KRC เพื่อเข้ารหัสอีกรั้ง

1. การสร้างข้อมูล  $KRF_i$ : Sender จะเข้ารหัสแต่ละส่วนย่อย ( $KRF_i$ ) ที่ได้จากการ split key ด้วยการใช้กุญแจสาธารณะของแต่ละ KRA (ซึ่งมีการเตรียมไว้ล่วงหน้า) ตามจำนวนส่วนย่อยของกุญแจ  $n_1$  โดยใช้ฟังก์ชันการเข้ารหัส RSA ที่มีการตั้งค่าให้มีความปลอดภัยสูง

2. การบรรจุข้อมูล KRF: หลังจากเข้ารหัสข้อมูล KRF<sub>i</sub> สำเร็จ ข้อมูลที่ได้จะถูกบรรจุไว้ในรูปแบบของ KRF structure ที่ประกอบด้วย

1. Other information: session id และ timestamp ที่ถูกเข้ารหัสไว้สำหรับ KRC

$$\text{EncryptedOtherInformation} = Ku_{KRC}[sessionID||timestamp]$$

2. แต่ละ KRF-i ที่ถูกเข้ารหัสไว้ สำหรับ KRA ที่กำหนดไว้ โดย KRF-i ประกอบด้วย Si และ SGN (shared group number)

$$\text{Encrypted KRF}_i = Ku_{agent_i}[Si||SGN]$$

3. Sr ถูกเข้ารหัสไว้ สำหรับ Requester (Receiver or Government)

$$\text{Encrypted Sr} = Ku_{\frac{\text{Receiver}}{\text{Government}}}[Sr]$$

4. แต่ละ TT<sub>i</sub> (unique secret number) ที่ถูกเข้ารหัสไว้ สำหรับ KRC เพื่อไว้ใช้กรณีอ่อนต์ล์ม โดยที่ TT<sub>i</sub> = Si  $\oplus$  SGN

$$\text{Encrypted TT}_i = Ku_{KRC}[TT_i]$$

จะได้ KRF ดังนี้

$$KRF = [encrypted KRF'_i s || encrypted TT'_i s || encrypted Sr || encrypted OtherInformation]$$

### 3.6.3.4 การส่งข้อมูล KRF ไปยัง Receiver

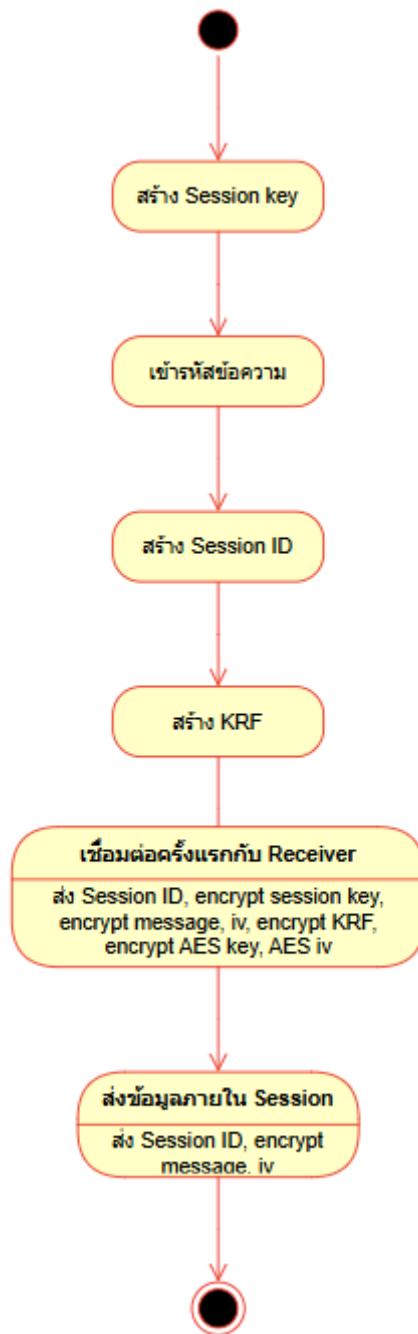
เมื่อสร้าง KRF สำเร็จ ข้อมูล KRF จะถูกส่งต่อไปยัง Receiver โดยผ่านการเข้ารหัส และมาตรการความปลอดภัยเพิ่มเติมก่อนทำการส่ง

1. การเข้ารหัสขั้นสุดท้าย: เพื่อให้แน่ใจว่าไม่มีบุคคลภายนอกสามารถดักจับ ข้อมูล KRF ได้ จะมีการเข้ารหัสข้อมูล KRF ทั้งหมดอีกครั้งด้วยกุญแจ AES ที่ถูกสร้างมาโดยเฉพาะและเข้ารหัสกุญแจนี้กุญแจสาธารณะของ KRC

$$\text{Encrypted KRF} = AES_{key}[KRF]$$

$$\text{Encrypted AES}_{key} = Ku_{KRC}[AES_{key}]$$

2. การส่งข้อมูล: Sender จะทำการส่งข้อมูลที่ถูกเข้ารหัสนี้ไปยัง Receiver ซึ่งจะไม่สามารถใช้กุญแจส่วนตัวของตนในการรอดรหัสเพื่อเข้าถึงข้อมูล KRF ที่อยู่ภายในได้โดยตรง และเป็นการบังคับให้ทำการใช้งานผ่านกระบวนการอย่างถูกต้อง
3. การบันทึกการส่งข้อมูล: เพื่อการติดตามและตรวจสอบ ในโโค้ดจะมีการบันทึก ข้อมูลเกี่ยวกับเวลาที่ทำการส่ง KRF ไปยัง Receiver และ session id เพื่อใช้ ตรวจสอบและระบุเหตุการณ์ในภายหลัง โดยขั้นตอนการทำงานของ Sender จะมีรูปแบบ Activity Diagram ดังรูปที่ 3.3



รูปที่ 3.6 ขั้นตอนการทำงานของ Sender

### 3.7 การทำงานของ Receiver/Requester

Receiver มีบทบาทในการถอดรหัสและอ่านข้อมูล หรือทำการร้องขอคืนกุญแจจาก KRC ในกรณีที่กุญแจสูญหายหรือชำรุด เมื่อได้รับการยืนยันตัวตนและข้อมูลที่เกี่ยวข้อง นอกจากนี้อาจเป็นหน่วยงานรัฐหรือหน่วยงานที่เกี่ยวข้องที่อาจสามารถทำการร้องขอคืนกุญแจไปตามกฎหมาย ได้ ผู้รับจะได้รับ  $Ks[M]$  ข้อมูลที่เข้ารหัสด้วยกุญแจลับ session key, encrypted KRF,  $AES_{key}[KRF]$ , encrypted session key  $Ku_{receiver}[\text{session key}]$ , encrypted  $AES_{key}$ ,  $Ku_{KRC}[AES_{key}]$ , iv,  $iv_{AES}$  และ session ID ในการติดต่อครั้งแรก และ  $Ks[M]$ , iv, session ID ในการติดต่อครั้งต่อ ๆ ไปใน session เดียวกัน กระบวนการทำงานของ Receiver มีดังนี้

#### 3.7.1 การเข้าถึงข้อมูล

Receiver จะใช้ private key ของตัวเองในการถอดรหัสข้อมูลที่ได้รับจาก Sender โดยข้อมูลที่ได้รับจะถูกเก็บไว้ในไฟล์ที่เข้าถึงได้เฉพาะ Receiver เท่านั้น และใช้ข้อมูลนั้น เช่น session key ในการถอดรหัสข้อมูลที่ได้รับ ต่อไป

#### 3.7.2 การขอคืนกุญแจ

Receiver ส่งคำขอการคืนกุญแจไปยัง KRC โดยมีข้อมูลที่ประกอบด้วย Request (session ID timestamp และ PKCE-like challenge token (challenge verifier) ที่ใช้ในการยืนยันตัวตน) และทำการเข้ารหัสด้วย Public key ของ KRC

$$\begin{aligned} & \text{EncryptedRequest} \\ &= Ku_{KRC}[\text{sessionID} || \text{sessionID} || \text{timestamp} || \text{challenge verifier}] \end{aligned}$$

และการส่งไปพร้อมกับ AES-key[KRF] และ  $Ku(KRC)[AES-key]$  ที่ถูกเข้ารหัสไว้ก่อนหน้าสำหรับ KRC เพื่อทำการขอคืนกุญแจกับ KRC

$$\begin{aligned} & \text{ข้อมูลที่ส่งไป KRC =} \\ & [ \text{encrypted Request} || \text{encrypted KRF} || \text{encrypted AES}_{key} || iv_{AES} ] \end{aligned}$$

### 3.7.3 การยืนยันตัวตนตามคำขอภัยคืน

หลังจากได้รับการตอบรับจาก KRC Receiver จะดำเนินการส่ง challenge code ไปยัง KRC เพื่อยืนยันตัวตน โดย KRC จะใช้ challenge code นี้ในการยืนยันว่า request มาจาก Receiver ตัวจริงเท่านั้น โดยเปรียบเทียบ challenge code กับ challenge verifier ที่ส่งไปก่อนหน้า โดยก่อนส่งจะมีการเข้ารหัส challenge code นี้ด้วย Public key ของ KRC เช่นเดียวกัน

$$\text{Encrypted ChallengeCode} = Ku_{KRC}[\text{challenge code}]$$

### 3.7.4 การสร้างกุญแจใหม่หลังรับข้อมูลจาก KRC

หลังจากได้รับข้อมูลจาก KRC Receiver จะดำเนินการสร้างกุญแจ session key ใหม่จาก Sr (ชิ้นส่วนบ่อสายของกุญแจลับ session key) ที่เข้ารหัสไว้สำหรับ Requester โดยเฉพาะ โดยที่ Requester จะต้องใช้ private key ของตนในการถอดรหัส และทำการ XOR กับ ข้อมูลกุญแจที่ได้จากการรวมกุญแจจาก KRC ที่ถูกส่งมาพร้อมกับ Sr ขั้นตอนนี้มีไว้ป้องกันไม่ให้มีการรั่วไหลของกุญแจจาก KRC

### 3.7.5 การรับคำสั่งทดสอบระบบ

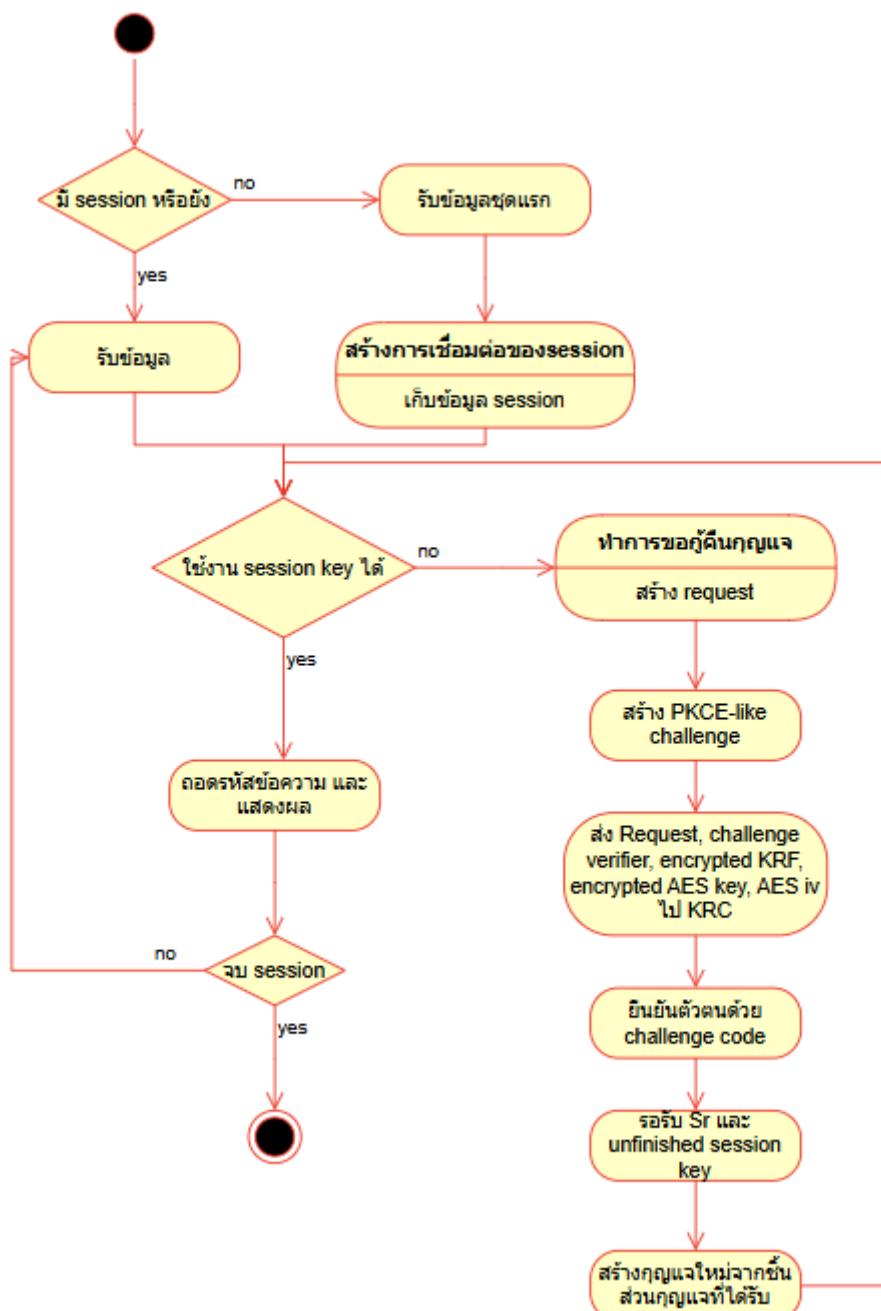
ตารางที่ 3.2 รายละเอียดของ API สำหรับทดสอบระบบ

Method	POST
Host	http://<RECEIVER_IP>:5050/manual_test
Content-type	application/json
Body	{ "command": "start test" } → Starts to recover the session key { "command": "corrupt" } → Corrupts the session key
Response	200 OK: { "message": <recovery response JSON> } 404 Not Found: { "message": "No session found" }

	400 Bad Request: { "message": "Invalid command" }
--	---

จากตารางที่ 3.2 ใช้วิธี (Method) แบบ Post ผ่านเส้นทาง (Path) /manual\_test โดยใช้โ kosten หรือโ kosten คือ `http://<RECEIVER_IP>:5050` โดย receiver IP มีการกำหนดไว้โดย `docker-compose.yml` ไว้ที่ 192.168.1.12 การส่งชุดข้อมูลอยู่ในรูปแบบ `application/json` เป็นรูปแบบที่สามารถส่งไฟล์ไปสู่บริการได้ ซึ่งการส่งจะส่งพร้อมพารามิเตอร์และไฟล์โดยใช้คีย์ชื่อว่า `command` และข้อมูลที่รับค่าเป็นข้อมูลคำสั่งโดยเฉพาะ `start test` เพื่อเป็นการออกคำสั่งการรีเริ่มการทำงานของกุญแจลับจาก Receiver ไปยัง KRC หรือ คำสั่ง `corrupt` เพื่อเป็นการทำลายกุญแจลับเพื่อเป็นการแสดงจำลองสถานะการณ์ของกุญแจลับที่มีการสูญหายหรือชำรุดเพื่อให้ผู้ใช้ได้เห็นเมื่อสำเร็จผู้ใช้จะได้รับการตอบกลับเป็นรูปแบบ JSON โดยจะแสดงสถานะของค่า response เช่น 200 เพื่อเป็นการแสดงการตอบรับคำสั่งได้อย่างราบรื่น 404 เมื่อ receiver ยังไม่ได้มีการเปิด session สนทนาก็ หรือ 400 เมื่อคำสั่งอยู่ในพิศรูปแบบหรือคำสั่งไม่ถูกต้อง

โดยขั้นตอนการทำงานของ Receiver จะมีรูปแบบ Activity Diagram ดังรูปที่ 3.4



รูปที่ 3.7 ขั้นตอนการทำงานของ Receiver

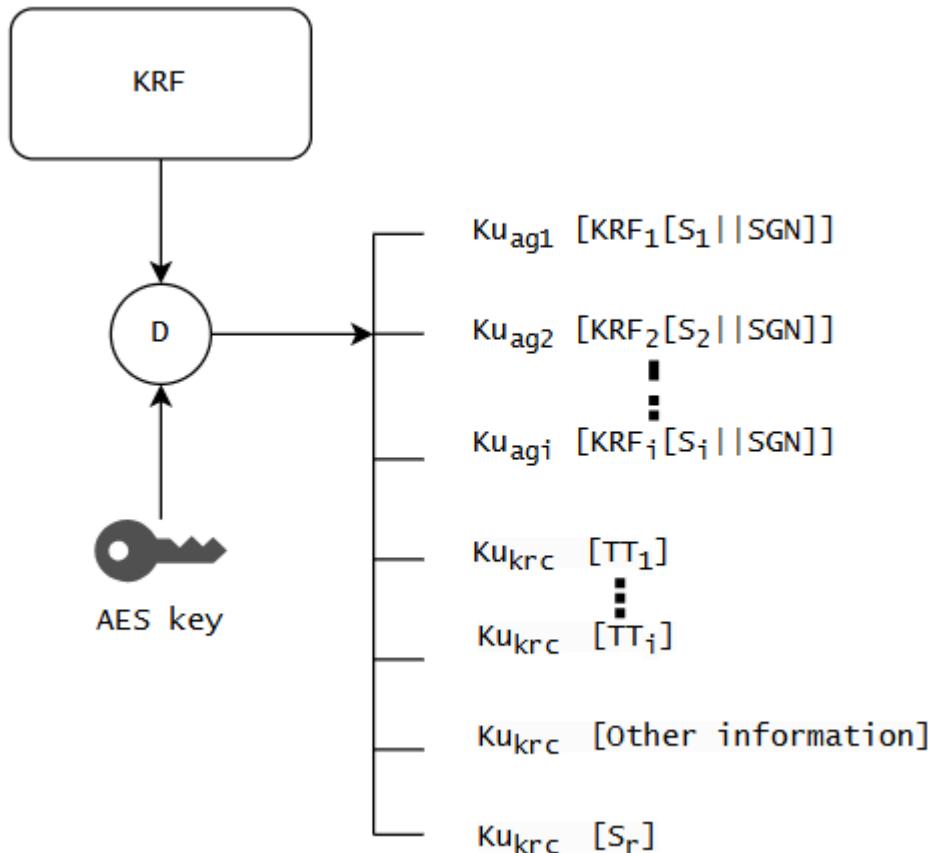
### 3.8 การทำงานของ KRC

KRC ทำหน้าที่เป็นศูนย์กลางในการจัดการการถือคืนกุญแจและตรวจสอบความถูกต้องของคำขอจาก Receiver และทำงานร่วมกันกับ KRA เพื่อทำการถือคืนกุญแจลับ โดยที่ KRC เป็นศูนย์กลางในการรวบรวมส่วนประกอบย่อยของกุญแจลับที่ได้รับจาก KRA และทำหน้าที่การถือรวมส่วนประกอบนี้ส่งกลับไปยังผู้รับและการตรวจสอบยืนยันตัวตนของทั้งผู้รับและ KRA ที่เกี่ยวข้อง ส่วน KRA ทำหน้าที่ในการถือคืนส่วนประกอบของกุญแจลับและส่งไปให้ KRC โดยมีขั้นตอนการทำงานดังต่อไปนี้

#### 3.8.1 การรับคำขอการถือคืน

- KRC ถอดรหัส AES<sub>key</sub> ด้วยกุญแจส่วนตัวของตนเอง Kr<sub>KRC</sub> จะได้รับกุญแจ AES ที่ใช้ในการถอดรหัส KRF
- KRC ถอดรหัส KRF ด้วยกุญแจ AES ร่วมกันกับ iv<sub>AES</sub> จะได้รับส่วนประกอบของ KRF (KRF<sub>i</sub>) คือ Ku<sub>agent-i</sub>[KRF<sub>i</sub>] จำนวน n ชิ้น (TT<sub>i</sub>) คือ Ku<sub>KRC</sub>[TT<sub>i</sub>] จำนวน n ชิ้น (Sr) คือ Ku<sub>receiver</sub>[Sr] จำนวน 1 ชิ้น และ Other information คือ Ku<sub>KRC</sub> จำนวน 1 ชิ้น
- KRC ถอดรหัสข้อมูล Other Information ด้วยกุญแจส่วนตัวของตนเอง Kr<sub>KRC</sub> จะได้รับข้อมูล session ID และ timestamp ของ Sender
- KRC ถอดรหัส Request ด้วยกุญแจส่วนตัวของตนเอง Kr<sub>KRC</sub> จะได้รับข้อมูล session ID, timestamp และ challenge verifier ของ Receiver และทำการตรวจสอบว่า session ID และ timestamp ที่ได้รับจาก Receiver ว่าตรงกันและตรวจสอบเวลาเพื่อให้แน่ใจว่าคำขอมีความถูกต้อง โดยเทียบกับข้อมูลของ Sender ใน KRF ในส่วนของ Other information ที่ได้รับมา

โดยการถอดรหัส KRF โดย KRC จะมีส่วนประกอบดังรูปที่ 3.8



รูปที่ 3.8 การผลิตรหัส KRF โดย KRC

### 3.8.2 การยืนยันตัวตนของผู้ขอการกู้คืนกัญแจ

KRC ใช้ PKCE-like challenge ใน การตรวจสอบ challenge code ที่ได้รับจาก Receiver เพื่อให้แน่ใจว่าเป็นผู้ร้องขอตัวจริง โดยนำ challenge code มาเข้า Hash function และเทียบกับ challenge verifier ที่ได้รับมาก่อนหน้านี้

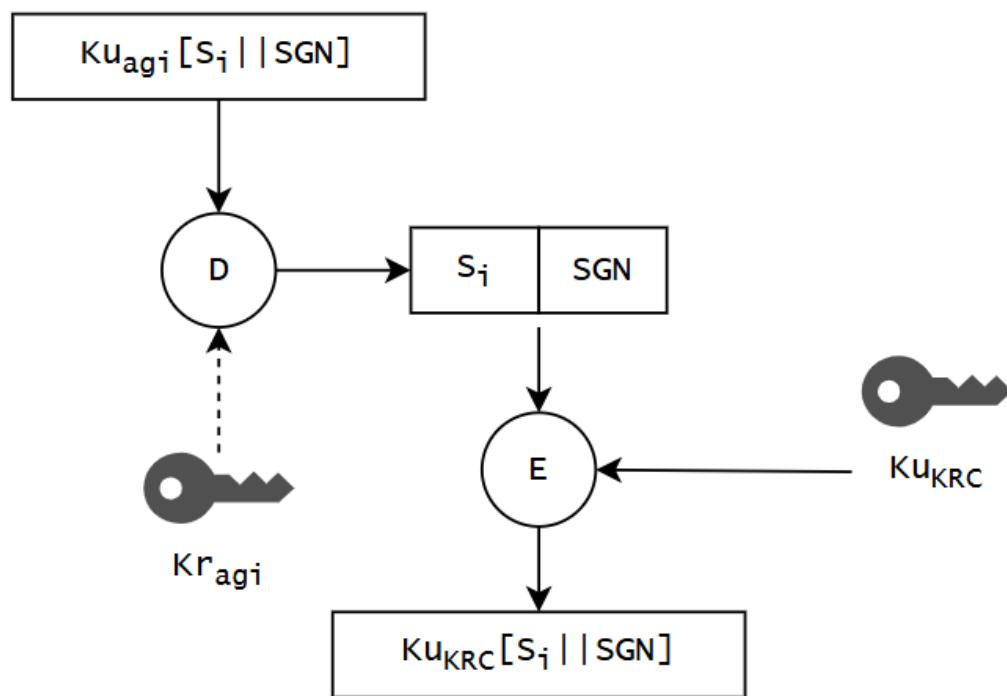
### 3.8.3 การแจกจ่ายข้อมูลไปยัง KRA และรับกลับของข้อมูล

กระบวนการนี้จะเกิดขึ้นก็ต่อเมื่อ KRC ได้ทำการรับการยืนยันตัวตนของผู้ร้องขอการกู้คืน กัญแจลับได้สำเร็จ (ผู้รับหรือหน่วยงานที่รับผิดชอบในการตรวจสอบข้อมูลสำเร็จการยืนยันตัวตน) เท่านั้น โดยมีขั้นตอนการทำงานดังต่อไปนี้

- KRC ทำการส่งคำขอการกู้คืนกัญแจย่องไปยัง KRA โดยมีการใช้ PKCE-like challenge เพื่อตรวจสอบตัวตนของ KRA แต่ละราย เหมือนกับการยืนยันตัวตนจาก Receiver
- KRC ทำการส่งข้อมูล KRF<sub>i</sub> ให้ Agent<sub>i</sub>

- $\text{Agent}_i$  ถอดรหัสด้วยกุญแจส่วนตัวของตนเอง  $Kr(\text{agent}_i)$  จะได้รับ  $S_i$  และ  $SGN$
- $\text{Agent}_i$  เข้ารหัส  $KRF_i$  ใหม่ด้วย กุญแจสาธารณะของ KRC  $Ku_{KRC}$  จะได้  $Ku_{KRC}[KRF_i]$
- $\text{Agent}_i$  ส่ง  $Ku_{KRC}[KRF_i]$  ให้ KRC
- KRC ถอดรหัสด้วย  $Kr_{KRC}$  จะได้รับ  $S_i$  และ  $SGN$
- KRC ทำการตรวจสอบ  $SGN$  ว่าเป็นอิเจนต์ในกลุ่มการคุ้มครองจริง
- KRC ทำการจัดเก็บข้อมูลชั่วคราวไว้สำหรับทำการรวมกุญแจและส่งกลับไปให้ผู้ร้องขอ

โดยการคุ้มครองส่วนประกอบของกุญแจลับโดย KRA จะมีขั้นตอนดังรูปที่ 3.9



รูปที่ 3.9 การคุ้มครองส่วนประกอบของกุญแจลับโดย KRA

#### 3.8.4 การแก้ปัญหากรณีอิเจนต์ล้ม

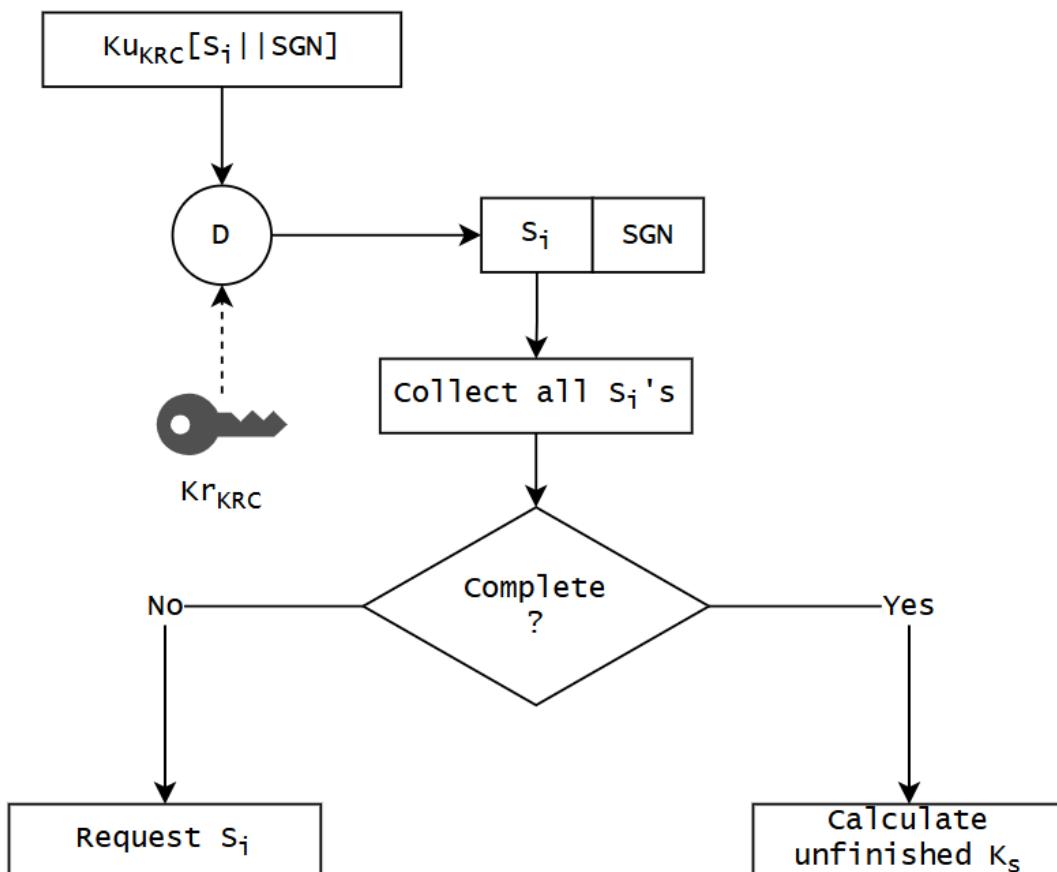
ในกรณีที่ KRA ล้มเหลว ส่งผลให้ KRC รวบรวม  $S_i$  ได้ไม่ครบ ดังนั้นจะเป็นหน้าที่ของ KRC ในการตรวจสอบการตอบสนองและเก็บบันทึกการตอบสนองของ KRA ที่เหลือ โดยในกรณีนี้ KRC จะทำการแจ้งเตือนและดำเนินการตามขั้นตอนการจัดการปัญหา โดยหลังจากพยาบาลติดต่ออิเจนต์ล้มเหลว 3 ครั้ง จะมีการใช้มาตรการ การใช้ข้อมูล  $TT_i$  ของอิเจนต์นั้น ๆ

ในการช่วยกู้คืนข้อมูลที่หายไป โดยต้องอาศัยค่า SGN ที่ได้รับจาก KRA ตัวอื่น ๆ ที่ใช้ในการเพื่อเป็นการยืนยันกลุ่มการกู้คืนและช่วยกู้คืนค่า Si ที่คาดหายไปด้วยการ

$$Si = TT_i \oplus SGN$$

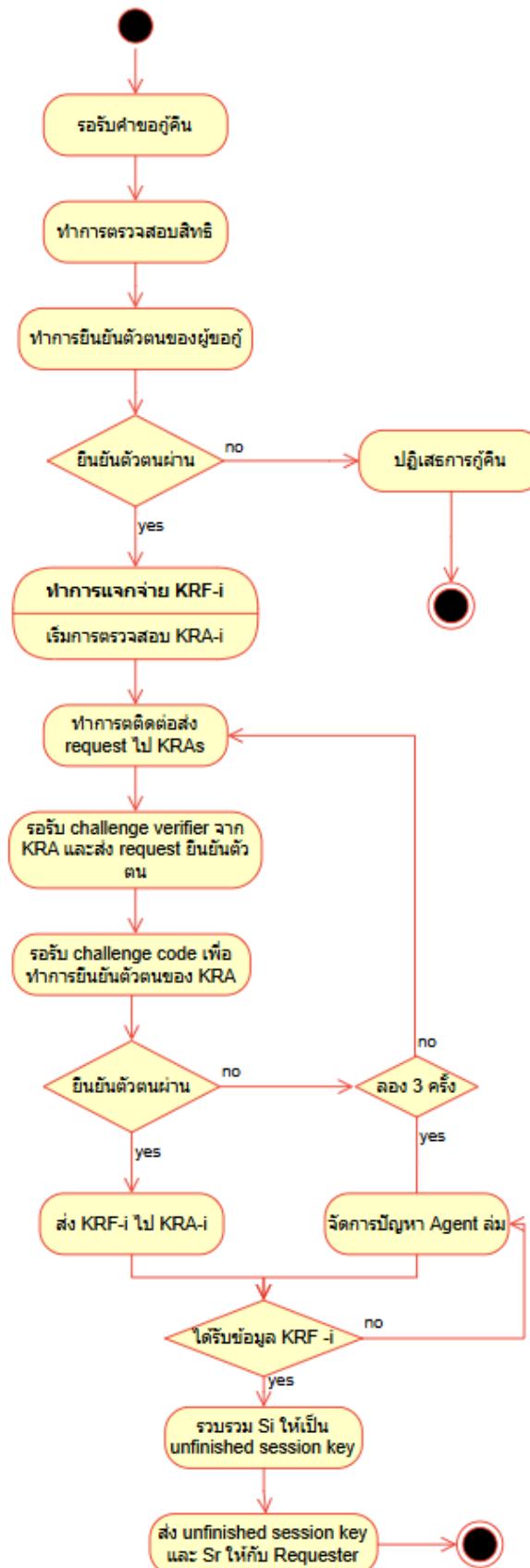
### 3.8.5 การรวมกุญแจ

KRC จะรวมกุญแจย่อยจาก KRA ที่ตอบสนองหรือถูกกู้คืนจากการณีอ่อนต์ล่ม และนำกุญแจย่อยเหล่านี้มาประกอบกลับเป็น unfinished session key และเข้ารหัสด้วย public key ของผู้ร้องขอ (Receiver)  $Ku_{receiver}$  เพื่อส่งคืนให้กับ Receiver พร้อมกับ  $Sr$  ที่มีการเข้ารหัสไว้ให้ Receiver จาก Sender มา ก่อนหน้านี้แล้ว เพื่อให้ Receiver เป็นผู้ถือครองกุญแจย่อยองนี้โดยเฉพาะ โดยขั้นตอนการกู้คืนกุญแจลับของ KRC จะมีขั้นตอนดังรูปที่ 3.10



รูปที่ 3.10 การกู้คืนกุญแจลับโดย KRC

โดยขั้นตอนการทำงานของ KRC จะมีรูปแบบ Activity Diagram ดังรูปที่ 3.5



รูปที่ 3.11 ขั้นตอนการทำงานของ KRC

### 3.9 การทำงานของ KRA

KRA (Key Recovery Agent) มีหน้าที่สำคัญในการตอบสนองต่อคำขอคุ้นกุญแจจาก KRC โดยให้การสนับสนุนในการคุ้นกุญแจย่อย (KRF<sub>i</sub>) เพื่อช่วยให้ระบบสามารถรวบรวมกุญแจที่สมบูรณ์และปลอดภัยได้

#### 3.9.1 การยืนยันตัวตนให้กับ KRC

KRA จะต้องยืนยันตัวตนต่อ KRC ผ่านการใช้ PKCE-like challenge mechanism โดย KRA จะสร้างและส่ง challenge verifier ให้กับ KRC ที่ต้องการทำคุ้นกุญแจไปก่อนแล้ว KRA ต้องทำการตอบสนองต่อ challenge verifier ด้วยการส่ง challenge code กลับไปยัง KRC เพื่อพิสูจน์ตัวตน เพื่อให้ KRC ทำการยอมรับ KRF<sub>i</sub> ให้

#### 3.9.2 การคุ้นกุญแจย่อยของกุญแจ

หลังจากการยืนยันตัวตนของ KRA สำเร็จ KRA จะทำการถอดรหัสข้อมูล KRF<sub>i</sub> ที่ถูกแบ่งมาจาก KRF จาก KRC ด้วยกุญแจส่วนตัวของตน Ku<sub>agent-i</sub> จากนั้นจะเข้ารหัสส่วนย่อยของกุญแจโดยใช้กุญแจสาธารณะของ KRC Ku<sub>KRC</sub> และส่งข้อมูลกลับไปยัง KRC เพื่อนำไปใช้ในการรวบรวมกุญแจในขั้นตอนต่อไป

#### 3.9.3 การช่วยคุ้นกุญแจเมื่อมีอุบัติเหตุ

หากมี KRA บางรายไม่สามารถตอบสนองได้ KRA ตัวอื่น ๆ ที่ยังทำงานได้จะต้องทำการส่งข้อมูลที่เหลือกลับไปก่อน และข้ามตัว KRA ที่ล้มเพื่อให้ KRC สามารถใช้ข้อมูลเหล่านั้นในการรวบรวมและคุ้น session key โดยใช้มาตรการกรณี KRA ล้มของ KRC ได้สำเร็จ โดยที่ KRC จะพยายามใช้ข้อมูลสำรอง TT<sub>i</sub> หรือดำเนินการที่จำเป็น (Retry on KRA) ตามกรณีเพื่อให้การคุ้นกุญแจสำเร็จลุล่วง โดยขั้นตอนการทำงานของ KRA จะมีรูปแบบ Activity Diagram ดังรูปที่ 3.6



รูปที่ 3.12 ขั้นตอนการทำงานของ KRA

### 3.10 สมมติฐานของการทำงานของระบบ

ในการพัฒนา Authenticated Secure Key Recovery System มีการกำหนดสมมติฐานในการทำงานของระบบเพื่อให้แน่ใจว่าการออกแบบและการทำงานของระบบสามารถดำเนินไปตามที่คาดหมาย โดยสมมติฐานที่ใช้เป็นพื้นฐานในการพัฒนามีดังนี้

### 3.10.1 ความปลอดภัยของคู่กุญแจ (Key Pairs) และการเข้ารหัส

- คู่กุญแจ (Public/Private Key) ของ Sender, Receiver, KRC และ KRA ได้รับการปกป้องอย่างปลอดภัย และไม่มีการรั่วไหลของ private key ไปยังบุคคลที่ไม่ได้รับอนุญาต
- ใช้ RSA-2048 และ AES-256 เป็นมาตรฐานการเข้ารหัสข้อมูล และสมมติว่าไม่มีช่องโหว่ที่ทำให้การเข้ารหัสนี้ถูกเจาะได้ง่าย และถือว่ามีความปลอดภัยเพียงพอ สำหรับการปกป้องข้อมูลที่ถูกส่งและจัดเก็บ
- Session key จะถูกคืนได้ก็ต่อเมื่อมีการยืนยันตัวตนและการตรวจสอบสิทธิ์ที่ถูกต้อง และมีการรวบรวม KRF (Key Recovery Fragments) ครบตามที่กำหนด

### 3.10.2 ความถูกต้องของ PKCE-like Challenge และการยืนยันตัวตน

- ระบบใช้ PKCE-like challenge เพื่อป้องกันการปลอมแปลงคำขอคืนกุญแจ โดยสมมติว่าไม่มีบุคคลที่สามารถดักฟังหรือปลอมแปลง challenge ได้ โดยถือว่าวิธีนี้สามารถป้องกันการโจมตี Replay Attack หรือ MITM Attack ได้ในระดับที่เหมาะสม
- KRC เป็นหน่วยงานที่เชื่อถือได้ (Trusted Authority) และเป็นผู้ตรวจสอบความถูกต้องของ request และ challenge จาก KRA
- สมมติว่า PKCE-like challenge สามารถป้องกัน Replay Attack ได้อย่างมีประสิทธิภาพ
- ไม่มีบุคคลที่สามารถดักฟังหรือแก้ไขข้อมูล PKCE-like challenge ระหว่าง KRA และ KRC ได้

### 3.10.3 เนื่องจากเครือข่ายและการสื่อสารระหว่างองค์ประกอบของระบบ

- สมมติว่า ทุกองค์ประกอบของระบบสามารถเชื่อมต่อถึงกันได้โดยไม่มีข้อจำกัดด้านเครือข่าย เช่น Firewall หรือ NAT
- ระบบใช้การเข้ารหัสแบบ End-to-End เพื่อให้แน่ใจว่าข้อมูลที่ส่งระหว่าง Sender, Receiver, KRC และ KRA ไม่สามารถถูกดักฟังได้
- ไม่มี Insider Attack หรือบุคคลภายในที่ไม่หวังดีพยายามเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

### 3.10.4 การทำงานของ KRC และ KRA

- KRC สามารถเข้าถึง KRF-i ได้ก็ต่อเมื่อได้รับอนุญาตจาก KRA และต้องผ่านกระบวนการตรวจสอบสิทธิ์
- KRA ทำหน้าที่ควบคุมการคืนกุญแจ โดยต้องมีการตรวจสอบการร้องขอจาก KRC ก่อนที่จะอนุญาตให้ดำเนินการคืน

- KRF-i (Key Recovery Fragments) ถูกจัดเก็บและป้องกันไม่ให้ถูกดัดแปลงโดยบุคคลที่ไม่ได้รับอนุญาต

#### 3.10.5 การทดสอบและการแสดงผลของระบบ

- ระบบไม่มีการทดสอบการโจมตีจริง เช่น Brute Force, Replay Attack หรือ MITM Attack แต่มีการออกแบบมาตรการป้องกันไว้ในกระบวนการการทำงาน
- ระบบสามารถบันทึกและแสดง message log ของกระบวนการทำงานได้ เพื่อให้ผู้ใช้สามารถตรวจสอบและติดตามสถานะของระบบ
- สมมติว่าการแสดงผลของระบบจะช่วยให้สามารถวิเคราะห์การทำงานได้โดยไม่ต้องอาศัยการทดสอบเชิงรุก

### 3.11 การพัฒนาและทดสอบระบบ

การพัฒนาและทดสอบระบบเป็นขั้นตอนสำคัญเพื่อให้มั่นใจว่าระบบทำงานได้อย่างถูกต้อง มีประสิทธิภาพ และมีความปลอดภัยตามที่ออกแบบไว้ Authenticated Secure Key Recovery System ถูกพัฒนาและทดสอบในสภาพแวดล้อมที่จำลองขึ้นผ่าน Docker โดยมีการควบคุมและตรวจสอบพฤติกรรมของแต่ละองค์ประกอบในระบบ

#### 3.11.1 การพัฒนาระบบ

การพัฒนาระบบใช้ Python เป็นภาษาโปรแกรมหลัก และทำงานภายใต้ Docker Container ซึ่งแต่ละองค์ประกอบของระบบแยกการทำงานออกจากกัน และสื่อสารผ่าน Docker Network เพื่อเพิ่มความปลอดภัยและความคุ้มครองข้อมูล ได้อย่างเป็นระบบ

เทคโนโลยีที่ใช้ในการพัฒนา:

- Flask: ใช้เป็น Web Framework สำหรับจัดการ API และการสื่อสารระหว่าง Sender, Receiver, KRC และ KRA กับ Web GUI ของตนและรับฟังคำสั่งพิเศษ
- Cryptography: ใช้สำหรับการเข้ารหัสข้อมูล การจัดการ PKI (Public Key Infrastructure) และการตรวจสอบความปลอดภัย
- Sockets: ใช้สำหรับการรับ-ส่งข้อมูลระหว่าง Sender, Receiver, KRC และ KRA อย่างปลอดภัย

โครงสร้างระบบถูกออกแบบให้รองรับการขยายตัวและปรับปรุงได้ง่าย โดยแยกโภคต์ออกเป็นโมดูลต่าง ๆ เพื่อให้สามารถพัฒนาและจัดการระบบได้อย่างเป็นระเบียบ

### 3.11.2 การทดสอบการทำงานของระบบ

เพื่อให้มั่นใจว่าระบบทำงานได้ตามข้อกำหนด มีการทดสอบระบบภายใต้สถานการณ์ต่างๆ โดยมุ่งเน้นไปที่ การตรวจสอบความปลอดภัยและความถูกต้องของกระบวนการกู้คืนกุญแจ ซึ่งรวมถึง:

- ทดสอบการเชื่อมต่อระหว่าง Container และการรับ-ส่งข้อมูลภายในเครือข่าย Docker Network
- ทดสอบ PKCE-like challenge
  - ตรวจสอบกระบวนการ PKCE-like challenge ที่ส่งไปยัง KRC จาก Receiver และ KRA1 – KRA5
  - ตรวจสอบว่า PKCE-like challenge ทำการเปรียบเทียบ challenge code และ challenge verifier ได้อย่างถูกต้องและปลอดภัย
- ทดสอบการรวม session key โดย KRC
  - ตรวจสอบกระบวนการ Key Recovery Field (KRF-i) ที่ส่งไปยัง KRA และการรวม session key จาก KRF<sub>i</sub>
  - ตรวจสอบว่า KRC สามารถแจกจ่าย session key ให้กับ Receiver ได้อย่างถูกต้องและปลอดภัย

นอกจากนี้ ยังมีการตรวจสอบ log และ debug mode เพื่อดูรายละเอียดการทำงานของระบบ และช่วยในการวิเคราะห์ข้อผิดพลาดที่เกิดขึ้น

### 3.11.3 การปรับปรุงตามผลการทดสอบ

จากผลการทดสอบ หากพบข้อผิดพลาดหรือปัญหาที่อาจเกิดขึ้น มีการปรับปรุงระบบในด้านต่างๆ เช่น:

- ปรับปรุงกระบวนการกู้คืนกุญแจ
  - ลดความล่าช้าหรือปัญหาที่เกี่ยวข้องกับการจัดการ session key
  - เพิ่มประสิทธิภาพการรวม KRF-i จาก KRA ให้เร็วขึ้น
- ปรับปรุงโพรเจกต์ PKCE-like challenge
  - ตรวจสอบว่าการเข้ารหัสและการตรวจสอบสิทธิ์มีความปลอดภัยเพียงพอ

- เพิ่ม log และ debug mode
  - เพิ่มการบันทึกข้อมูลที่สำคัญ เช่น ข้อผิดพลาดในการสื่อสารและความล้มเหลวของการคืนกุญแจ
  - ช่วยให้สามารถติดตามปัญหาและแก้ไขข้อผิดพลาดได้รวดเร็วยิ่งขึ้น

กระบวนการพัฒนาและทดสอบระบบช่วยให้ Authenticated Secure Key Recovery System ทำงานได้อย่างมีเสถียรภาพและปลอดภัย โดยผ่านการทดสอบภายใต้เงื่อนไขที่แตกต่างกัน การปรับปรุงระบบตามผลการทดสอบช่วยเพิ่มประสิทธิภาพ ลดความผิดพลาด และทำให้ระบบมีความปลอดภัยสูงขึ้น

### 3.12 มาตรการความปลอดภัยในระบบ

ระบบ Authenticated Secure Key Recovery System ได้ออกแบบและนำมาตรการความปลอดภัยมาใช้ เพื่อป้องกันการเข้าถึงที่ไม่ได้รับอนุญาตและการโจมตีที่อาจเกิดขึ้น เมื่อว่าจะยังไม่มีการทดสอบโจมตีจริง แต่ระบบได้รวมกลไกการรักษาความปลอดภัยไว้ด้วยกันแล้ว ขั้นตอนออกแบบและพัฒนา

#### 3.12.1 การใช้ PKCE-like Challenge ในการป้องกันคำขอคู่คืนกู้ยู

PKCE-like challenge ถูกนำมาใช้เพื่อป้องกันการปลอมแปลงคำขอคู่คืนกู้ยู และมีหลักการดังนี้

- ใช้ค่า challenge และ verifier เพื่อยืนยันความถูกต้องของคำขอคู่คืน
- มีการใช้Nonce หรือ Timestamp เพื่อลดความเสี่ยงจากการใช้ข้อมูลซ้ำ
- มีมาตรการป้องกัน Brute Force Attack เช่น การจำกัดจำนวนครั้งของการตรวจสอบ challenge

#### 3.12.2 การเข้ารหัสข้อมูลเพื่อรักษาความลับ

เพื่อให้แน่ใจว่าข้อมูลสำคัญในระบบได้รับการปกป้อง ระบบใช้การเข้ารหัสข้อมูลแบบ Asymmetric และ Symmetric ได้แก่

- RSA-2048: ใช้สำหรับเข้ารหัส session key และข้อมูลสำคัญที่ต้องส่งระหว่าง KRC และ KRA
- AES-256: ใช้สำหรับเข้ารหัส Key Recovery Fragments (KRFs) และข้อมูลที่ต้องการป้องกันภายในระบบ
- มาตรการป้องกัน Private Key ร่วมกัน
  - Private Key ถูกเข้ารหัสก่อนจัดเก็บ
  - จำกัดสิทธิ์การเข้าถึงไฟล์ Private Key เพื่อป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต

#### 3.12.3 มาตรการป้องกันการโจมตี

แม้ว่าจะยังไม่มีการทดสอบการโจมตีจริง ระบบได้ออกแบบให้รองรับการป้องกันภัยคุกคามที่อาจเกิดขึ้น ดังนี้

- ป้องกัน Man-in-the-Middle Attack (MITM)
  - ใช้ PKCE-like challenge เพื่อให้มั่นใจว่าผู้ส่งคำขอเป็นผู้ใช้ที่ได้รับอนุญาต
  - ใช้การเข้ารหัสแบบ end-to-end เพื่อป้องกันการดักฟังข้อมูล
- ป้องกัน SQL Injection และ Command Injection
  - API ได้รับการออกแบบให้มีการตรวจสอบ Input และไม่สามารถใช้คำสั่งอันตรายได้
- ป้องกันการโจมตีแบบ Brute Force
  - มี Rate Limiting เพื่อลดโอกาสที่ผู้โจมตีจะพยายามสุ่มค่าจำนวนมาก
  - ระบบสามารถล็อกคำขอที่ผิดปกติหรือซ้ำซ้อนภายในระยะเวลาสั้น ๆ

มาตรการความปลอดภัยในระบบ Authenticated Secure Key Recovery System ถูกออกแบบมาเพื่อป้องกันการโจมตีที่อาจเกิดขึ้น โดยเน้นที่ การเข้ารหัสข้อมูล, การป้องกันการปลอมแปลงคำขอ, และการควบคุมการเข้าถึง แม้ว่าจะยังไม่มีการทดลองโจมตีจริง แต่ระบบได้เพิ่มกลไกเหล่านี้เพื่อให้แน่ใจว่าสามารถป้องกันภัยคุกคามได้อย่างมีประสิทธิภาพ

### 3.13 การทดสอบการกู้คืนกุญแจ กรณีอ่อนตัว

เพื่อให้มั่นใจว่าระบบ Authenticated Secure Key Recovery System สามารถกู้คืน session key ได้เมื่อในกรณีที่ Key Recovery Agent (KRA) บางส่วนล่มหรือไม่สามารถให้บริการได้ มีการทดสอบสถานการณ์ต่าง ๆ เพื่อประเมินความสามารถของ Key Recovery Coordinator (KRC) ในการรวบรวมและกู้คืนกุญแจสำหรับ

#### 3.13.1 การทดสอบการกู้คืนกุญแจ

เป้าหมายของการทดสอบนี้คือ การตรวจสอบว่า KRC สามารถรวบรวม session key ได้แม้ว่าจะมี KRA ล้มบางส่วน

- จำลองกรณีที่ KRA บางตัวไม่ตอบสนอง
  - ทดสอบว่า KRC ยังสามารถรวบรวม Key Recovery Field ( $KRF_i$ ) ที่เหลืออยู่ และสร้าง session key ได้สำเร็จ

- ทดสอบ fallback mechanism
  - ตรวจสอบว่ามี แผนสำรอง (fallback mechanism) ในกรณีที่ KRA หลายตัวล้ม เช่น การใช้ TT<sub>i</sub> เพื่อใช้ในการดำเนินการต่อไป
- ทดสอบการกำหนดจำนวน KRA ขั้นต่ำที่ต้องใช้ในการกู้คืนกุญแจ
  - ตรวจสอบว่าระบบสามารถกำหนด threshold ของ KRA ที่จำเป็นต้องใช้ในการกู้คืนกุญแจ เพื่อให้แน่ใจว่าการกู้คืนทำงานได้แม้ว่า KRA บางส่วนล้ม

### 3.13.2 การทดสอบการกระจายกุญแจย่อย

ระบบจะต้องกระจาย Key Recovery Field (KRF<sub>i</sub>) ไปยัง KRA ต่าง ๆ อย่างถูกต้อง และสามารถร่วมกับมาได้ในภายหลัง

- ตรวจสอบกระบวนการแจกจ่าย KRF<sub>i</sub> ไปยัง KRA
  - ตรวจสอบว่าระบบกระจาย KRF<sub>i</sub> ไปยัง KRA ได้อย่างถูกต้องโดยไม่มีการสูญหายหรือเกิดข้อผิดพลาด
- ทดสอบการรับและคืนค่า KRF<sub>i</sub> จาก KRA
  - ทดสอบว่าทุก KRA สามารถ คืนค่า KRF<sub>i</sub> ได้อย่างถูกต้องเมื่อมีการร้องขอจาก KRC
- ทดสอบการร่วม session key โดย KRC
  - ตรวจสอบว่า KRC สามารถร่วม KRF<sub>i</sub> ที่ได้รับกลับมาและกู้คืน session key ได้ถูกต้อง

### 3.13.3 การจัดการข้อมูลสำรอง

หาก KRA หลายตัวล้ม ระบบต้องสามารถใช้ข้อมูลสำรอง เพื่อให้สามารถกู้คืน session key ได้สำเร็จ

- ทดสอบการใช้ TT<sub>i</sub> แทน KRA<sub>i</sub> ที่ล้ม
  - ตรวจสอบว่าระบบสามารถเลือกใช้ TT<sub>i</sub> ที่มีอยู่เพื่อทดแทน KRA<sub>i</sub> ที่ล้ม ได้
- ตรวจสอบความสามารถของ KRC ในการทำงานร่วมกับ KRA ที่เหลืออยู่
  - ทดสอบว่าระบบสามารถ กำหนด threshold ของ KRA ที่เหลืออยู่ และ ดำเนินการกู้คืน session key ได้สำเร็จ

จากการทดสอบ Authenticated Secure Key Recovery System พบว่า KRC สามารถถูกคืน session key ได้แม่ KRA บางตัวจะล่ม โดยระบบมีการออกแบบ fallback mechanism ให้รองรับกรณีลูกค้า อย่างไรก็ตาม ยังคงต้องมีการปรับปรุง การสำรองข้อมูล KRF และการตรวจสอบ KRA ที่เหลืออยู่ เพื่อให้มั่นใจว่าระบบสามารถถูกคืน session key ได้ในทุกสถานการณ์

### 3.14 การแสดงผล

การแสดงผลในระบบ Authenticated Secure Key Recovery System มีความสำคัญอย่างยิ่งในการให้ผู้ใช้หรือผู้ดูแลระบบสามารถตรวจสอบสถานะและกระบวนการการทำงานของระบบได้อย่างชัดเจน โดยระบบจะมีการแสดงผล Message Log ผ่าน Web GUI เพื่อให้สามารถติดตามและวิเคราะห์การทำงานของระบบแบบเรียลไทม์ได้

#### 3.14.1 การแสดงผลกระบวนการถูกคืนกุญแจ

ระบบจะมีการบันทึกและแสดง Message Log ที่เกี่ยวข้องกับการถูกคืนกุญแจ (Session Key) เพื่อให้ผู้ใช้สามารถตรวจสอบได้ว่าการถูกคืนสำเร็จหรือไม่

- ระบบจะแสดง ขั้นตอนการถูกคืน Session Key ในลำดับที่เกิดขึ้นจริง
- หากการถูกคืนสำเร็จ ระบบจะแสดง ข้อความยืนยัน ว่า Session Key ได้ถูกถูกคืนเรียบร้อย
- หากเกิดปัญหา เช่น KRA ไม่สามารถตอบสนองได้ ระบบจะแสดง ข้อความแจ้งเตือน และระบุว่ากำลังใช้ Fallback Mechanism

#### 3.14.2 การแสดงสถานะการทำงานของระบบ

ระบบจะมีการแสดง Message Log เพื่อตรวจสอบสถานะขององค์ประกอบต่าง ๆ ในระบบ Authenticated Secure Key Recovery System ดังนี้

##### 3.13.2.1. การแสดง Log ของกระบวนการถูกคืนกุญแจ

- ระบบจะแสดง ขั้นตอนการดำเนินการของ PKCE-like challenge และการตรวจสอบ Session Key
- ทุกการดำเนินการจะถูกบันทึกเป็น Message Log และแสดงผลใน GUI

##### 3.13.2.2. การแสดงสถานะของ KRA (Key Recovery Agent)

- ระบบจะแสดงสถานะของ KRA ว่า *Online* หรือ *Offline* พร้อมข้อความแจ้งเตือนที่ชัดเจน
- หาก KRA ไม่ตอบสนอง ระบบจะแสดงคำเตือน และระบุว่ากำลังใช้วิธีสำรอง (Fallback Mechanism)

### 3.13.2.3. การแสดงสถานะของ KRC (Key Recovery Center)

- ระบบจะแสดงสถานะของ KRC ในแต่ละขั้นตอน เช่น
  - การเริ่มต้นคำขอคืนกุญแจ
  - การตรวจสอบสิทธิ์ของผู้ใช้
  - การประมวลผลและรวม KRF-i จากหลายแหล่ง
- หากมีข้อผิดพลาดในการทำงานของ KRC ระบบจะแสดง ข้อความแจ้งเตือน และแสดงรายละเอียดของข้อผิดพลาด

การแสดงผลในระบบ Authenticated Secure Key Recovery System เน้นไปที่การให้ผู้ใช้ติดตามกระบวนการทำงานผ่าน Message Log บน Web GUI โดยมีรายละเอียดเกี่ยวกับ

- สถานะของการคืนกุญแจ
- สถานะขององค์ประกอบหลัก เช่น KRA และ KRC
- Message Log ของกระบวนการทำงาน

ระบบจะเน้นให้ข้อมูลแบบเรียลไทม์ เพื่อช่วยให้ผู้ดูแลสามารถตรวจสอบและแก้ไขปัญหาได้อย่างมีประสิทธิภาพ

## บทที่ 4

### ผลการทดลองและวิเคราะห์

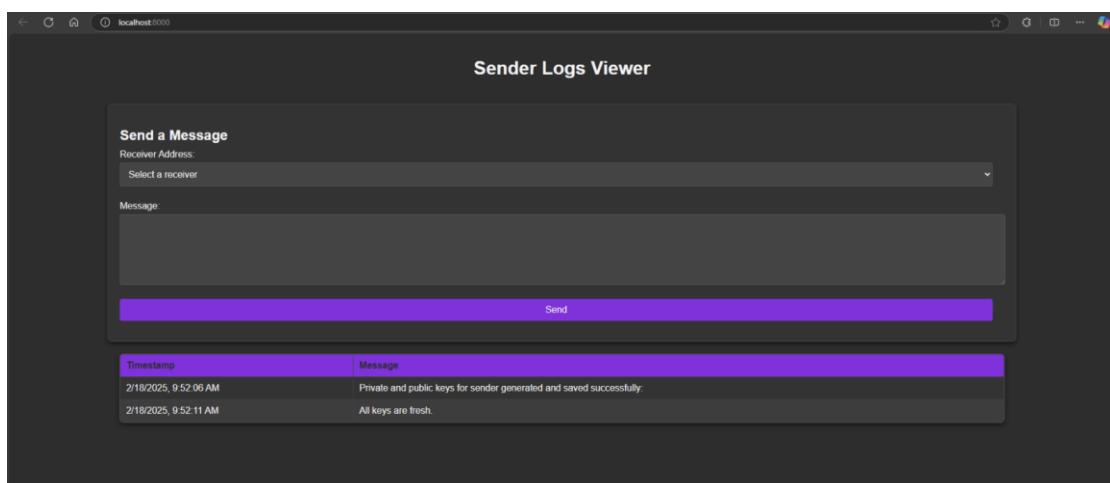
จากบทที่ 3 ได้กล่าวถึงการทำงานของโปรแกรม โดยบทที่ 4 เป็นการแสดงผลการทดลองโปรแกรม และ ผลการประเมิน โดยมีรายละเอียดดังนี้

#### 4.1 ผลการทดสอบการทำงานของโปรแกรม

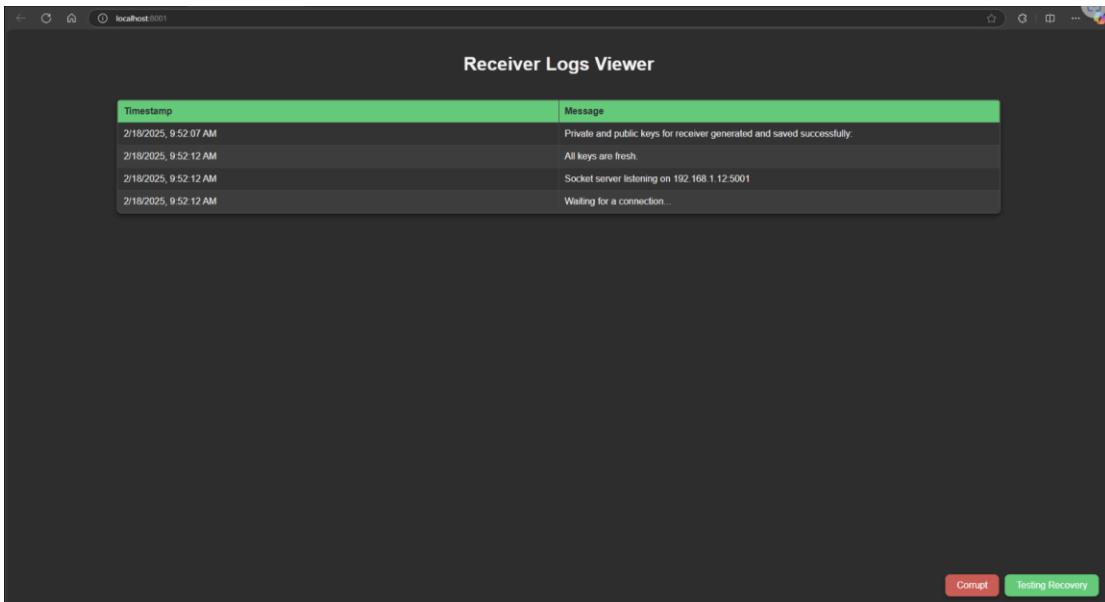
แสดงผลบน Web GUI ของแต่ละฝ่าย Sender, Receiver, KRC และ KRA1 – KRA5

##### 4.1.1 หน้าการเริ่มต้นพร้อมใช้งาน

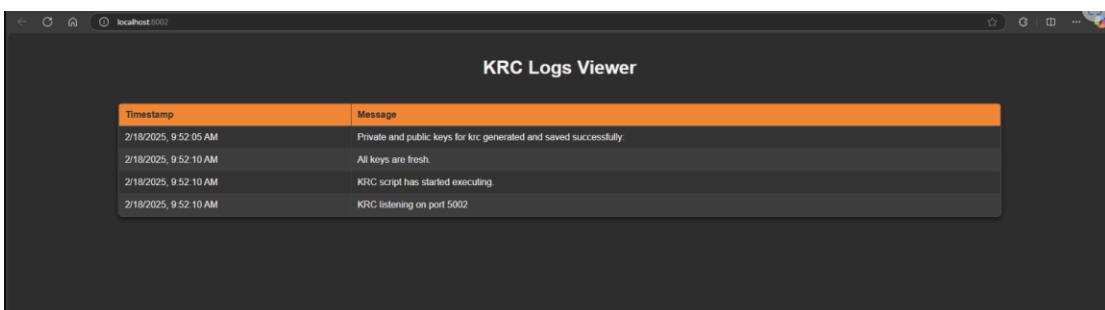
การแสดงผลเริ่มต้นของแต่ละฝ่าย



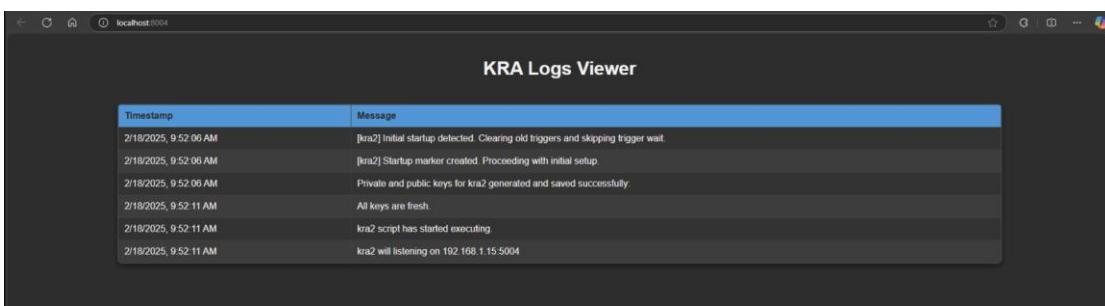
รูปที่ 4.1 หน้าเริ่มต้นของ Sender



รูปที่ 4.2 หน้าเริ่มต้นของ Receiver



รูปที่ 4.3 หน้าเริ่มต้นของ KRC

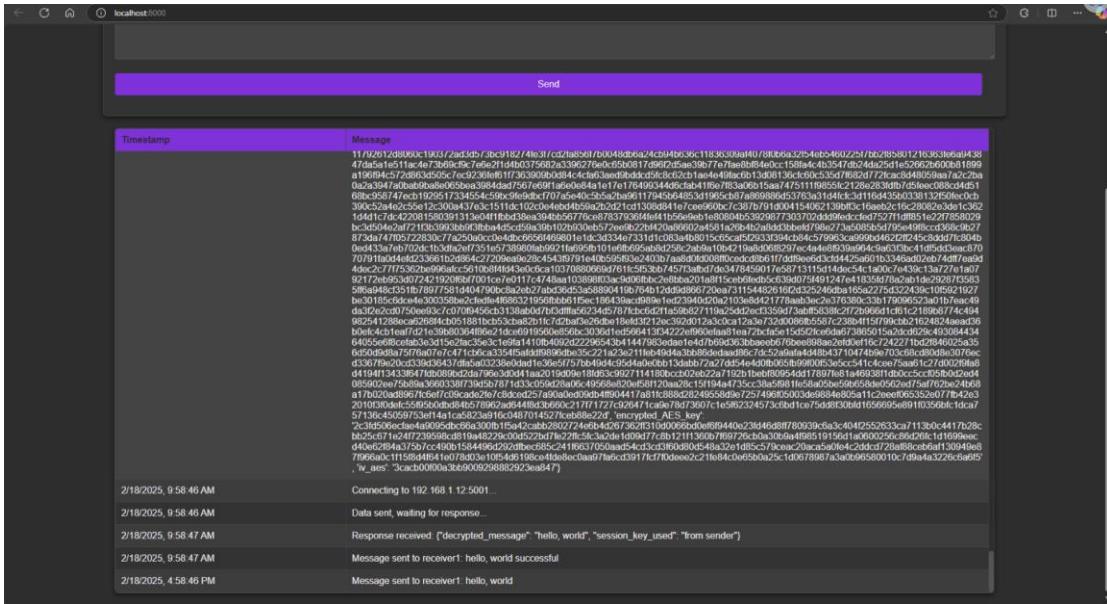


รูปที่ 4.4 ตัวอย่างหน้าเริ่มต้นของ KRA1 – KRA5

#### 4.1.2 หน้าการเริ่มต้น

แสดงผลหลังการส่งข้อความครั้งแรกของ Sender โดยจะมีตัวเลือกให้ผู้ใช้งานนี้

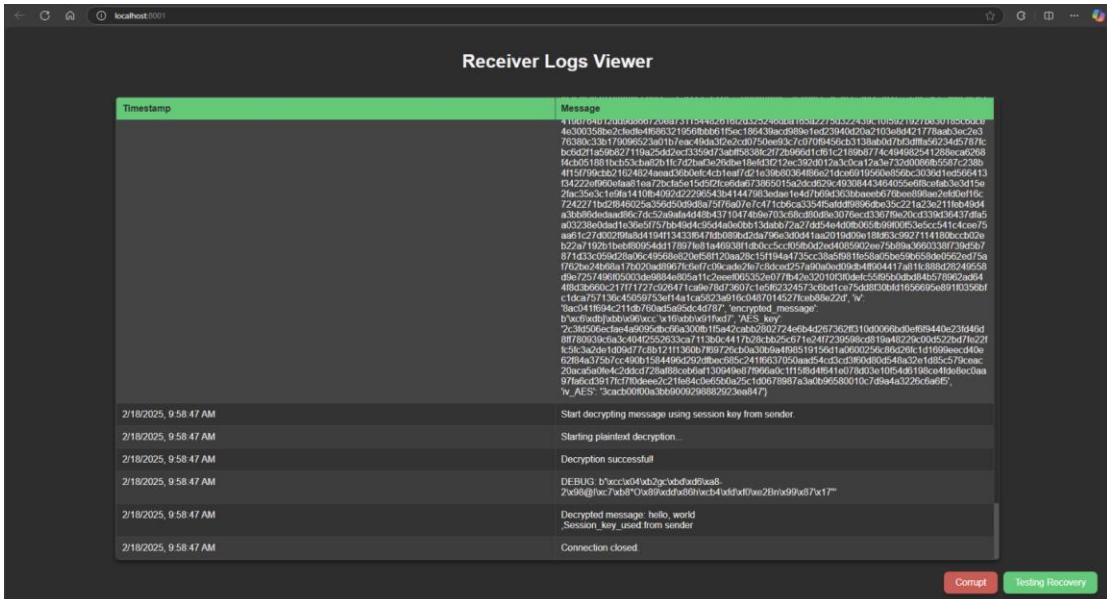
- Receiver Address: เพื่อเลือกเป้าหมายในการส่ง
- Message: ช่องว่างสำหรับการเขียน Plaintext ที่ต้องการส่งไป
- Send: ปุ่มกดส่งเพื่อยืนยันการส่งข้อความ



รูปที่ 4.5 ตัวอย่างการแสดงผลหลังการส่งข้อความจาก Sender

แสดงผลหลังการรับข้อมูลความถี่ของ Receiver โดยจะมีตัวเลือกให้ผู้ใช้งานนี้

- Corrupt: เพื่อจำลองสถานการณ์ session key เสียหายหรือชำรุด
  - Testing Recovery: เพื่อเริ่มสถานการณ์การรื้อลงของคู่กุญแจจาก KRC



รูปที่ 4.6 ตัวอย่างการแสดงผลหลังการรับข้อมูลจาก Sender ของ Receiver

#### 4.1.2 หน้าการสัมภารณ์ session key ของ Receiver เสียงหาย

4.1.3.1 แสดงผลหลังการส่งข้อมูลความของ Sender โดยจะมีการตอบกลับจาก Receiver ที่เป็น GARBAGE OUTPUT

```

localhost:~ % 
Send

Timestamp          Message
2/18/2025, 9:58:46 AM Connecting to 192.168.1.12:5001...
2/18/2025, 9:58:46 AM Data sent, waiting for response...
2/18/2025, 9:58:47 AM Response received: {"decrypted_message": "hello, world", "session_key_used": "from sender"}
2/18/2025, 9:58:47 AM Message sent to receiver1: hello, world successful
2/18/2025, 4:58:46 PM Message sent to receiver1: hello, world
2/18/2025, 10:03:26 AM Waiting for Input message...
2/18/2025, 10:03:26 AM Address for receiver: receiver1 received.
2/18/2025, 10:03:26 AM Input message: try to send message.
2/18/2025, 10:03:26 AM Debugging: No key changes detected.
2/18/2025, 10:03:26 AM Debugging: Keys have changed: False
2/18/2025, 10:03:26 AM sending to existing session
2/18/2025, 10:03:26 AM Start encrypting plaintext.
2/18/2025, 10:03:26 AM Encrypted plaintext successfully.
2/18/2025, 10:03:26 AM Payload prepared: {"session_id": "8671b91875e413a9558-3c83f2747e82", "iv": "185a6678379d3f7300a7d42834dc3745", "encrypted_message": "9ff11220565c22a6e611922b677862b3c1fb0a8f"}
2/18/2025, 10:03:26 AM Connecting to 192.168.1.12:5001...
2/18/2025, 10:03:26 AM Data sent, waiting for response...
2/18/2025, 10:03:27 AM Response received: {"decrypted_message": "GARBAGE OUTPUT 33222:8a031b4384bd0fb961cb5cb030b5682", "session_key_used": "from sender"}
2/18/2025, 10:03:27 AM Message sent to receiver1: try to send message. successful
2/18/2025, 5:03:26 PM Message sent to receiver1: try to send message.


```

รูปที่ 4.7 ตัวอย่างการแสดงผลหลังการส่งข้อความโดยที่ session key corrupt

#### 4.1.3.2 แสดงผลหลังการรับข้อความของ Receiver โดยที่ session key corrupt และถอดรหัสข้อความไม่ได้

```

localhost:~ %
Receiver Logs Viewer

Timestamp          Message
2/18/2025, 10:03:27 AM Start decrypting message using session key from sender.
2/18/2025, 10:03:27 AM Starting plaintext decryption...
2/18/2025, 10:03:27 AM Decryption successful.
2/18/2025, 10:03:27 AM Decryption output is corrupted (garbage data detected).
2/18/2025, 10:03:27 AM DEBUG: b7qjvdx_d_hv8cxvb8xlebx3d3x8f42482845: encrypted message
2/18/2025, 10:03:27 AM Decrypted message: GARBAGE OUTPUT 33222:da031b4384bd0fb961cb5cb030b5682
,Session_key_used from sender
2/18/2025, 10:03:27 AM Connection closed.


```

รูปที่ 4.8 ตัวอย่างการแสดงผลถอดรหัสข้อความโดยที่ session key ชำรุดหรือสูญหาย

## 4.2 ผลการประเมินการทดสอบ

แสดงผลบน Web GUI ของแต่ละฝ่าย Sender, Receiver, KRC และ KRA1 – KRA5 ในช่วงการทำการร้องขอคืนกุญแจ และหลังทำการคืนกุญแจ

#### 4.2.1 หน้าการแสดงผลหลังจากเริ่มกระบวนการร้องขอคุณกุญแจ

#### 4.2.1.1 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกัญแจของ Receiver

รูปที่ 4.9 ตัวอย่างการแสดงผลหลังการทำกรุ๊ปคิ้นกุญแจ และผลลัพธ์ที่สืบความสุดท้ายด้วยคิ้นกุญแจใหม่ของ Receiver

4.2.1.2 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRC

KRC Logs Viewer	
Timestamp	Message
2/18/2025, 10:44:49 AM	Comparing challenge verifier b'xe3y3xfe Jfx8bxd xd8xvxf0 _w00t7fxd Pwdxbsx0 7vb1 vdfl9 xd2xb4 xbe xde xd2Px0 7' from KRA with challenge code b'xd0t xdex0 f0xdWx0 7wxj x02 fxwid xdex0 xdax xd0x0 xdax xd0x0 xd0x0 4hxt0' from KRA
2/18/2025, 10:44:49 AM	KRA-5 verification succeeds.
2/18/2025, 10:44:49 AM	Payload to send: [encrypted_b6f / 18972cb5c21feadfb4c96b3648917af02b301151a3037e35de878282379d7c3da0c7c5597fb1b3c7bc8bf3d116e93fb53e8fb53752c2fc00b880db79969e1515d6babd7f8c495fc9449d02178d1209fe761085074951069801000e8223250296ebef5d570345b9957b06dc3dc956aac20675374bc2722aa7f7a1d327fc9343608ba16e7929ae4eaaf5c9b3b947859125359f79cb357b45ec59ddce35e8t479d4a322984a14ab3d355d83k3d88b50c7a90d2f15149243152759d7c7430f35825ea567b2ea544cb56cb6914deed7e085fa517ea9b0edd264490d88a1a952163a9b92874de5b0b093cb3', type: 'KRF_releve']
2/18/2025, 10:44:49 AM	send data to KRA-5
2/18/2025, 10:44:49 AM	Response received from KRA-5.
2/18/2025, 10:44:49 AM	Closing connection after receiving data from KRA-5.
2/18/2025, 10:44:49 AM	Extracting KRF-5 from KRA.
2/18/2025, 10:44:49 AM	Decrypting re-encrypted KRF-I: b"("9dab6857a4f5d8663d82e0581e0facb1a796049d098520d738ba228e8fe6", "SGN": "141661b17ade6e2b5c646bfa036346abdb89e47286b8943631c027905")
2/18/2025, 10:44:49 AM	KRF-I list successfully decrypted and parsed.
2/18/2025, 10:44:49 AM	KRF-I list successfully completed without problems
2/18/2025, 10:44:49 AM	Beginning Phase 4: Assembling session key.
2/18/2025, 10:44:49 AM	Beginning key assembly.
2/18/2025, 10:44:49 AM	Session key successfully assembled: b'wea9xb6b6x88x85xb6b6x10xed9 y xf2 x11 xb4 xe1u_ xc5 xe1Ezvday xd1x8fr v82 x91'
2/18/2025, 10:44:49 AM	Beginning Phase 5: Encrypting and sending session key.
2/18/2025, 10:44:49 AM	Key parts successfully sent.
2/18/2025, 10:44:49 AM	Closing connection after finishing recovery process.

รูปที่ 4.10 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยเจของ KRC

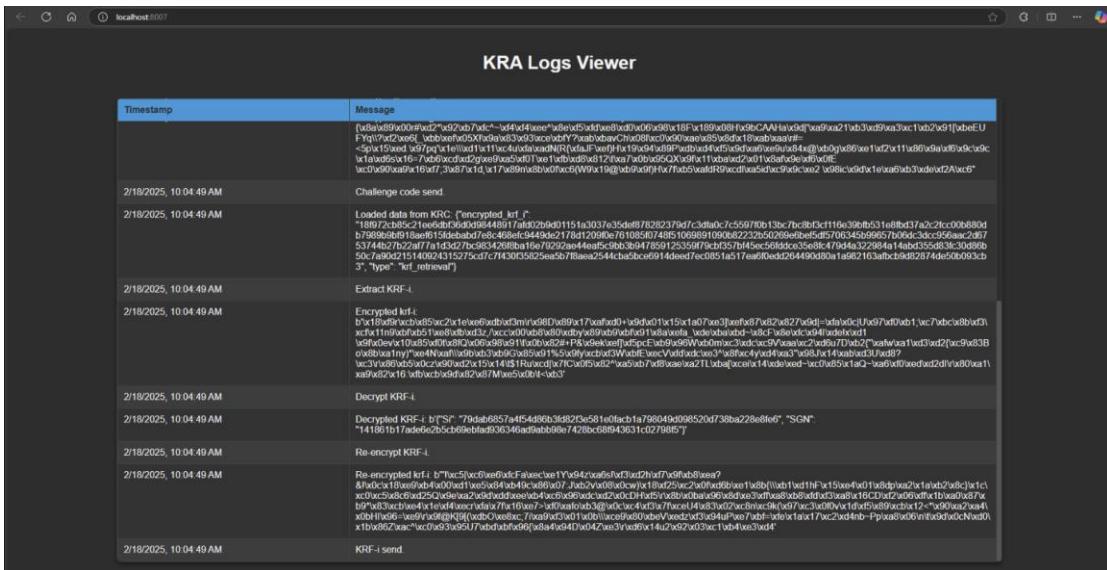
4.2.1.3 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRA1 – KRA5

รูปที่ 4.11 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยของ KRA1

รูปที่ 4.12 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกู้ณูของ KRA2

รูปที่ 4.13 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยของ KRA3

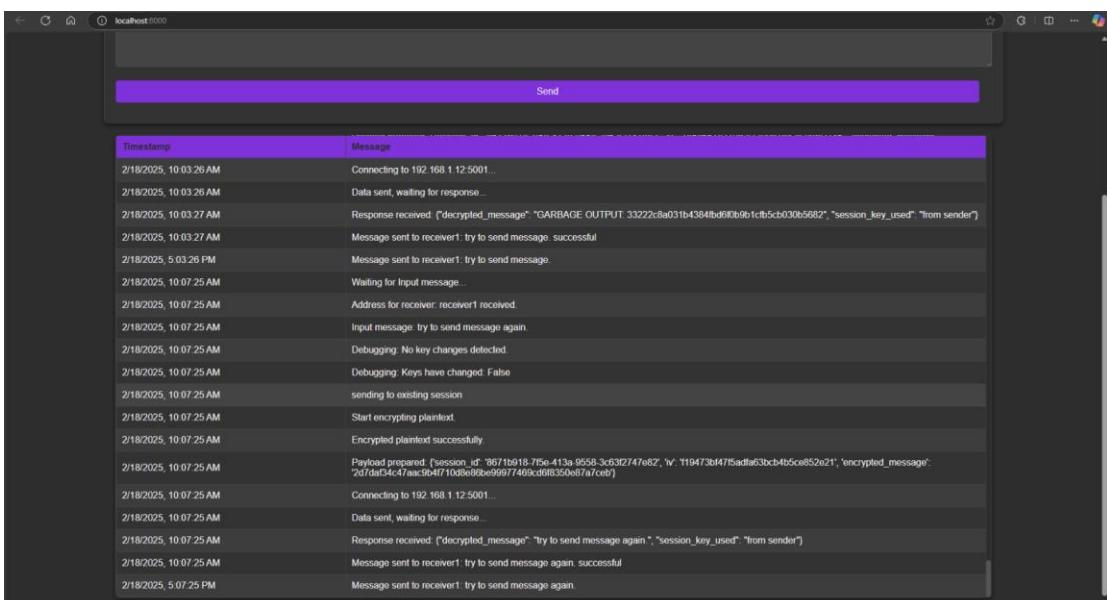
รูปที่ 4.14 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยของ KRA4



รูปที่ 4.15 ตัวอย่างการแสดงผลหลังการทำการคุ้นคุ้นของ KRA5

#### 4.2.2 หน้าการสนทนาหลังจากทำการร้องขอคืนกุญแจสำเร็จ

#### 4.2.2.1 แสดงผลการสันทนาหลังจากคืนกู้ของ Sender



รูปที่ 4.16 ตัวอย่างการแสดงผลการสนทนาระบบทามากับคุณกัญแจของ Sender

#### 4.2.2.2 แสดงผลการสันทนาหลังจากกู้คืนกุญแจของ Receiver

The screenshot shows the 'Receiver Logs Viewer' interface with a green header bar. The main area displays log entries in a table format:

Timestamp	Message
2/18/2025, 10:07:25 AM	Start decrypting message using session key from sender
2/18/2025, 10:07:25 AM	Starting plaintext decryption...
2/18/2025, 10:07:25 AM	Decryption successful
2/18/2025, 10:07:25 AM	DEBUG: b'hex3d4tvk2qcvlded9x88-2v36@fwc.7wb80x89udhdw0hwcb7vdxt0xe28mud99x87x17'
2/18/2025, 10:07:25 AM	Decrypted message: try to send message again ,Session_key used from sender
2/18/2025, 10:07:25 AM	Connection closed

At the bottom right, there are two buttons: 'Compt' and 'Testing Recovery'.

รูปที่ 4.17 ตัวอย่างการแสดงผลการสันทนาหลังการทำการกู้คืนกุญแจของ Receiver

#### 4.2.3 หน้าการแสดงผลหลังจากเริ่มกระบวนการร้องขอกู้คืนกุญแจ กรณีมีอเจนต์ล้ม

##### 4.2.3.1 แสดงผลหลังจากเริ่มกระบวนการร้องขอ กู้คืนกุญแจของ Receiver

The screenshot shows the 'Receiver Logs Viewer' interface with a green header bar. The main area displays log entries in a table format:

Timestamp	Message
2/18/2025, 10:17:55 AM	Session key successfully recovered and stored
2/18/2025, 10:17:55 AM	Start decrypting message using session key from KRC
2/18/2025, 10:17:55 AM	Starting plaintext decryption...
2/18/2025, 10:17:55 AM	Decryption successful
2/18/2025, 10:17:55 AM	DEBUG: b'hex3d4tvk2qcvlded9x88-2v36@fwc.7wb80x89udhdw0hwcb7vdxt0xe28mud99x87x17'
2/18/2025, 10:17:55 AM	Decrypted message: try to send message again ,Session_key used from KRC
2/18/2025, 10:17:55 AM	Message sent to Receiver: start test

At the bottom right, there are two buttons: 'Compt' and 'Testing Recovery'.

รูปที่ 4.18 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจ และถอดรหัสข้อความสุดท้ายค้างกุญแจใหม่ของ Receiver กรณีมีอเจนต์ล้ม

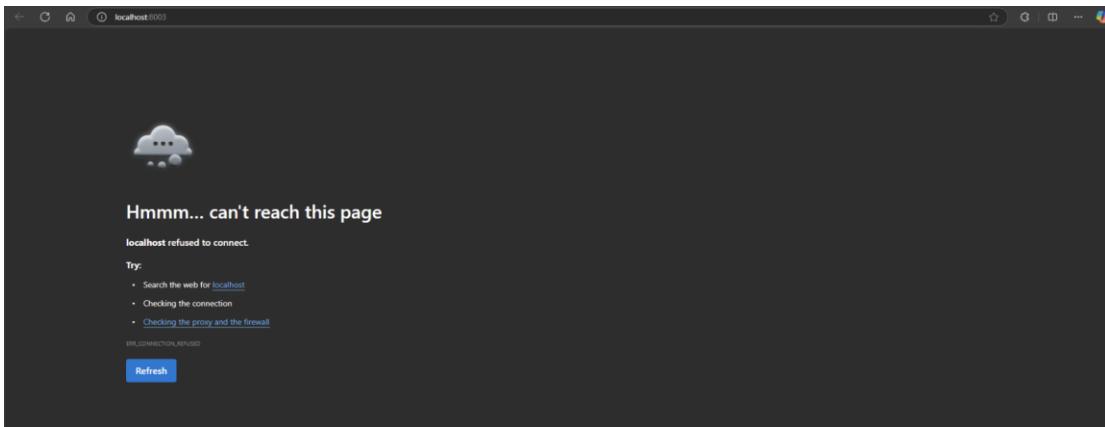
#### 4.2.3.2 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRC

รูปที่ 4.19 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกุญแจของ KRC กรณีมีเงินติดลบ

KRC Logs Viewer	
Timestamp	Message
2/1/2025, 11:27:38 AM	Error communicating with KRA-1: 'type'
2/1/2025, 11:27:38 AM	Retrying communication with KRA-1 (Attempt 1/3)...
2/1/2025, 11:27:41 AM	Error sending to KRA-1: [Error 113] No route to host
2/1/2025, 11:27:41 AM	Error receiving from KRA-1: 'NoneType' object has no attribute 'recv'
2/1/2025, 11:27:41 AM	Invalid response type from KRA-1. Expected 'challenge_response', got: None
2/1/2025, 11:27:41 AM	Retrying communication with KRA-1 (Attempt 2/3)...
2/1/2025, 11:27:44 AM	Error sending to KRA-1: [Error 113] No route to host
2/1/2025, 11:27:44 AM	Error receiving from KRA-1: 'NoneType' object has no attribute 'recv'
2/1/2025, 11:27:44 AM	Invalid response type from KRA-1. Expected 'challenge_response', got: None
2/1/2025, 11:27:44 AM	Retrying communication with KRA-1 (Attempt 3/3)...
2/1/2025, 11:27:47 AM	Error sending to KRA-1: [Error 113] No route to host
2/1/2025, 11:27:47 AM	Error receiving from KRA-1: 'NoneType' object has no attribute 'recv'
2/1/2025, 11:27:47 AM	Invalid response type from KRA-1. Expected 'challenge_response', got: None
2/1/2025, 11:27:47 AM	KRA-1 failed after 3 retries.
2/1/2025, 11:27:47 AM	KRA-1 permanently failed due to exception. Marking index as None
2/1/2025, 11:27:47 AM	prepare data to send KRA-2
2/1/2025, 11:27:47 AM	Payload to send: [type: 'challenge start']
2/1/2025, 11:27:47 AM	send data to KRA-2
2/1/2025, 11:27:47 AM	Response received from KRA-2.
2/1/2025, 11:27:47 AM	Closing connection after receiving data from KRA-2

รูปที่ 4.20 ตัวอย่างการแสดงผลการพยากรณ์ดicit'อ่อนนต์ที่ล่ำ(KRA1) ของ KRC กรณีมีอ่อนนต์ล่ำ

4.2.3.3 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRA1 – KRA5



รูปที่ 4.21 ตัวอย่างการแสดงผลการล้มของ KRA1 กรณีมีอุจจาระหล่อหลอม (KRA1 ล้ม)

รูปที่ 4.22 ตัวอย่างการแสดงผลหลังการทำการกู้คืนกลุ่มของ KRA2 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)

รูปที่ 4.23 ตัวอย่างการแสดงผลหลังการทำรายการกู้เงินกู้ของ KRA3 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)

รูปที่ 4.24 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยของ KRA4 กรณีมีอ่อนตัวลง(KRA1 ล้ม)

รูปที่ 4.25 ตัวอย่างการแสดงผลหลังการทำการคุ้นเคยของ KRA5 กรณีมีอเจนต์ล้ม(KRA1 ล้ม)

4.2.4 หน้าการแสดงผลหลังจากเริ่มกระบวนการรีวอร์ช์องขอคุ้นเคยๆ แล้ว กรณีอาจมีเงื่อนไขล้มทุกตัว

4.2.4.1 แสดงผลหลังจากการเริ่มกระบวนการ การร้องขอคืนกุญแจของ Receiver

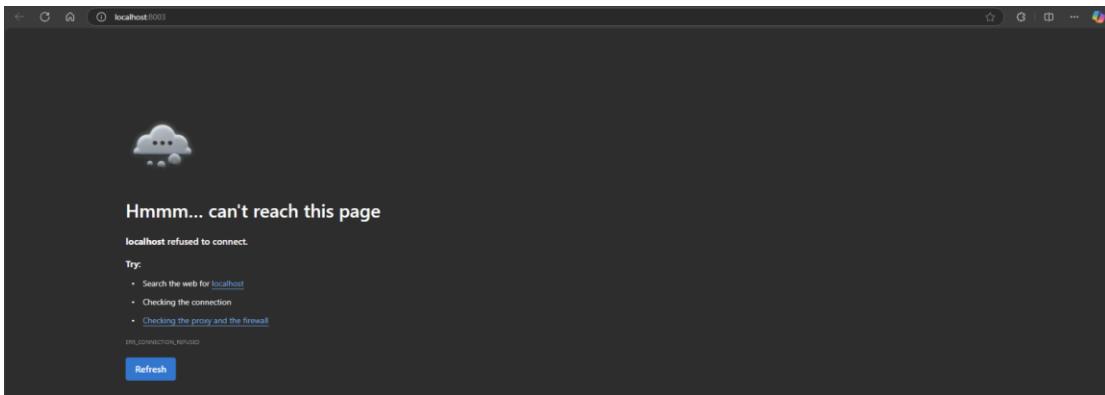
รูปที่ 4.26 ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ Receiver กรณีมีอเจนต์ล้มทุกตัว

4.2.4.2 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRC

Timestamp	Message
2/18/2025, 10:22:53 AM	Retrying communication with KRA-5 (Attempts: 3/5)
2/18/2025, 10:22:54 AM	Error sending to KRA-5: [Error 113] No route to host
2/18/2025, 10:22:54 AM	Error receiving from KRA-5. 'NoneType' object has no attribute 'recv'
2/18/2025, 10:22:54 AM	Invalid response type from KRA-5. Expected 'challenge_response', got: None
2/18/2025, 10:22:54 AM	KRA-5 failed after 3 refines
2/18/2025, 10:22:54 AM	KRA-5 permanently failed due to exception. Marking index as None.
2/18/2025, 10:22:54 AM	Not all KRF-i parts were collected, initiating failure handling
2/18/2025, 10:22:54 AM	Handling overall KRA failure for missing KRF-i parts.
2/18/2025, 10:22:54 AM	[WARNING] All KRF-i parts are missing. Reconstruction is not possible
2/18/2025, 10:22:54 AM	KRF-i list successfully completed without problems.
2/18/2025, 10:22:54 AM	Beginning Phase 4: Assembling session key.
2/18/2025, 10:22:54 AM	Beginning key assembly.
2/18/2025, 10:22:54 AM	[ERROR] KRF-1 is missing. Cannot assemble session key.
2/18/2025, 10:22:54 AM	[ERROR] KRF-2 is missing. Cannot assemble session key.
2/18/2025, 10:22:54 AM	[ERROR] KRF-3 is missing. Cannot assemble session key.
2/18/2025, 10:22:54 AM	[ERROR] KRF-4 is missing. Cannot assemble session key.
2/18/2025, 10:22:54 AM	[ERROR] KRF-5 is missing. Cannot assemble session key.
2/18/2025, 10:22:54 AM	[ERROR] Session key assembly failed due to missing or invalid KRF-i parts.
2/18/2025, 10:22:54 AM	Beginning Phase 5: Encrypting and sending session key.
2/18/2025, 10:22:54 AM	Closing connection after finishing recovery process.

รูปที่ 4.27 ตัวอย่างการแสดงผลหลังการทำการคืนกุญแจของ KRC กรณีมีอเจนต์ล้มทุกด้วย

4.2.4.3 แสดงผลหลังจากเริ่มกระบวนการการร้องขอคืนกุญแจของ KRA1 – KRA5 กรณีมีอเจนต์ล้มทุกตัว



รูปที่ 4.28 ตัวอย่างการแสดงผลการล้มของ KRA1 – KRA5 กรณีมีอเจนต์ล้มทุกตัว

## บทที่ 5

### บทสรุปและข้อเสนอแนะ

#### 5.1 สรุปผลการทดลอง

โครงการนี้มุ่งเน้นการพัฒนาระบบกู้คืนกุญแจที่มีความปลอดภัยสูง โดยใช้การยืนยันตัวตนหลายขั้นตอนและกลไกป้องกันการปลอมแปลงคำร้องขอกู้คืนกุญแจ เพื่อเพิ่มความมั่นคงปลอดภัยในการเข้าถึงข้อมูลที่เข้ารหัส ระบบที่พัฒนาขึ้นเรียกว่า Authenticated Secure Key Recovery System (AS-KRS) ซึ่งเป็นการต่อขยายจากระบบ SFM-KRS และออกแบบมาให้สามารถทำงานได้ในสภาพแวดล้อมเครือข่ายที่มีหลายฝ่ายที่เกี่ยวข้อง

##### 5.1.1 การพัฒนาระบบยืนยันตัวตนหลายขั้นตอน

เพื่อเพิ่มระดับความปลอดภัยของระบบ AS-KRS ได้มีการใช้การยืนยันตัวตนหลายปัจจัย (Multi-Factor Authentication - MFA) โดยรวมเอาการยืนยันตัวตนด้วย certificate-based authentication และ token-based authentication ซึ่งช่วยลดความเสี่ยงจากการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต

##### 5.1.2 การใช้กลไก PKCE-like Challenge

มีการเพิ่มกลไก PKCE-like challenge เพื่อป้องกันการปลอมแปลงคำร้องขอกู้คืนกุญแจ วิธีการนี้ช่วยให้สามารถตรวจสอบความถูกต้องของคำร้องขอ ก่อนดำเนินการกู้คืนกุญแจได้อย่างมีประสิทธิภาพ ลดโอกาสในการโจมตีแบบ replay attack และ man-in-the-middle attack

#### 5.2 สรุปผลตามวัตถุประสงค์ที่คาดว่าจะได้รับ

ระบบ AS-KRS ที่พัฒนาขึ้นสามารถตอบโจทย์วัตถุประสงค์ของโครงการได้อย่างมีประสิทธิภาพ โดยสามารถกู้คืนกุญแจได้อย่างปลอดภัย ลดความเสี่ยงจากการถูกโจมตี และสามารถนำไปประยุกต์ใช้ในองค์กรที่ต้องการความปลอดภัยสูงได้

#### 5.3 ปัญหาและการปรับปรุง

- ระบบอาจมีความชักช้อนในการติดตั้งและใช้งานในองค์กรที่ไม่มีโครงสร้างพื้นฐานด้านความปลอดภัยที่เพียงพอ
- การใช้ PKCE-like Challenge อาจเพิ่มระยะเวลาการตรวจสอบสิทธิ์ ทำให้เกิดความล่าช้าในการดำเนินงาน
- ควรปรับปรุงการออกแบบ UI/UX ให้ใช้งานได้สะดวกยิ่งขึ้น

#### 5.4 ความรู้และเครื่องมือที่จำเป็นต่อการพัฒนาต่อ

- เทคโนโลยีด้าน Public Key Infrastructure (PKI) : ระบบโครงสร้างพื้นฐานกุญแจสาธารณะที่ใช้สำหรับการเข้ารหัสและยืนยันตัวตนผ่านใบรับรองดิจิทัล ช่วยให้สามารถตรวจสอบความถูกต้องของคู่กุญแจและป้องกันการปลอมแปลงข้อมูลได้
- การพัฒนา Multi-Factor Authentication (MFA) : ระบบการยืนยันตัวตนที่ใช้มากกว่าหนึ่งปัจจัย เช่น รหัสผ่าน, โทเค็น หรือการยืนยันตัวตนทางชีวภาพ เพื่อเพิ่มระดับความปลอดภัยให้กับกระบวนการรักษาความปลอดภัย
- การใช้งาน Token-based authentication และ Certificate-based authentication :
  - Token-based authentication ใช้โทเค็นที่ออกโดยเซิร์ฟเวอร์เพื่อให้ผู้ใช้สามารถเข้าถึงระบบโดยไม่ต้องส่งข้อมูลรับรองซ้ำ ลดความเสี่ยงจากการโจมตีแบบ session hijacking
  - Certificate-based authentication ใช้ใบรับรองดิจิทัลเพื่อยืนยันตัวตนของผู้ใช้หรืออุปกรณ์ เพิ่มความมั่นใจว่าข้อมูลรับรองไม่สามารถปลอมแปลงได้ง่าย
- การวิเคราะห์และตรวจสอบความปลอดภัยของระบบกุญแจ เช่น การทดสอบการเจาะระบบ (Penetration Testing) และการตรวจสอบการทำงานของโปรโตคอลความปลอดภัย เพื่อให้แน่ใจว่าระบบสามารถป้องกันการโจมตีจากภายนอกได้

#### 5.5 ข้อเสนอแนะ

- ควรดำเนินการทดสอบระบบในสถานการณ์การใช้งานจริงเพื่อประเมินประสิทธิภาพและความปลอดภัยของระบบเพิ่มเติม
- พิจารณานำเทคโนโลยีอื่น เช่น biometric authentication มาใช้ร่วมกับการยืนยันตัวตนหลายปัจจัย เพื่อเพิ่มความปลอดภัยและความสะดวกในการใช้งาน
- ควรเพิ่มกระบวนการตรวจสอบจากทั้งสองฝ่ายในการทำ PKCE-like challenge เพื่อให้สอดคล้องกับแนวทาง Zero Trust ซึ่งมีความสำคัญมากขึ้นในปัจจุบัน และช่วยเพิ่มความปลอดภัยในการยืนยันตัวตนและป้องกันการปลอมแปลงคำขอคู่กุญแจ
- ควรศึกษาผลกระทบของกลไก PKCE-like Challenge ต่อประสิทธิภาพของระบบ และพัฒนาให้สามารถรองรับการใช้งานในระบบขนาดใหญ่ได้อย่างเหมาะสม
- พัฒนาแนวทางการรักษาความปลอดภัย เช่นการป้องกันการโจมตีทางไซเบอร์ โดยไม่ลดทอนความปลอดภัยของระบบ

## เอกสารอ้างอิง

กนกรรณ กันยะมี และ จันทร์บูรณ์ สพิติวิชวงศ์ (2553). ระบบการคุ้มกันแบบหลายเลนที่มีความมั่นคงและยืดหยุ่นสูง. JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY | VOL 1 | ISSUE 2 | JUL-DEC 2010.

กนกรรณ กันยะมี และจำรุญ จันทร์กุญชร. (2567). การเพิ่มความมั่นคงในการบริหารจัดการคุณแจ้ง สำหรับการคุ้มกันแบบหลายเลนที่อาศัยศูนย์กลางในการคุ้มกันและ วารสารวิชาการมหาวิทยาลัยอีสเทิร์นแอดิชัน ฉบับวิทยาศาสตร์และเทคโนโลยี ปีที่ 18 ฉบับที่ 2 ประจำเดือน พฤษภาคม-สิงหาคม 2567

Diffie, W., & Hellman, M. (1976). New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6), 644–654.

Rivest, R. L., Shamir, A., & Adelman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21(2), 120-126.

Shamir, A. (1979). How to Share a Secret. Communications of the ACM, 22(11), 612–613.

Rivest, R. (1992). The MD5 Message-Digest Algorithm. IETF RFC 1321.

Bellare, M., & Rogaway, P. (1995). Security of Cryptographic Hash Functions. Fast Software Encryption.

Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley.

Schneier, B. (1996). "Key Splitting and Secret Sharing" Applied Cryptography. Wiley.

RSA Laboratories. (1997). RSA Data Security, Inc. Public Key Cryptography Standards (PKCS), Version 1.5.

Daemen, J., & Rijmen, V. (2002). The Design of Rijndael: AES - The Advanced Encryption Standard. Springer.

Adams, C., & Lloyd, S. (2003). Understanding PKI: Concepts, Standards, and Deployment Considerations. Addison-Wesley.

## ເອກສາຮ້າງອິງ(ຕ່ອ)

- Schneier, B. (2005). "AES Encryption: Theory and Practice" Cryptanalysis of the Advanced Encryption Standard.
- Stigler, J. (2008). "Introduction to Hash Functions" Cryptographic Techniques in Computer Science.
- National Institute of Standards and Technology (NIST). (2012). Secure Hash Standard (SHS). NIST FIPS PUB 180-4.
- OAuth 2.0. (2015). OAuth 2.0 Authorization Framework. IETF RFC 7636.
- IETF RFC 7636. (2015). Proof Key for Code Exchange by OAuth Public Clients. IETF.
- Stallings, W. (2017). Man-in-the-Middle Attacks and Countermeasures. Pearson Education.
- Stallings, W. (2017). Public Key Cryptography. Cryptography and Network Security: Principles and Practice. Pearson.
- Parecki, A. (2017). OAuth 2.0 for Browser-Based Apps. O'Reilly Media.
- W3C. (2019). OAuth 2.0 and PKCE: A Secure Alternative to Client Authentication. World Wide Web Consortium.
- Grinberg, M. (2020). Flask Web Development. O'Reilly Media, Inc.
- IBM (2021). Size considerations for public and private keys. © Copyright IBM Corporation 2015, 2021 Retrieved December 12, 2024, from  
<https://www.ibm.com/docs/en/zos/2.4.0?topic=certificates-size-considerations-public-private-keys>
- Van Rossum, G. (2023). Python Programming Language. Python.org. Retrieved from  
<https://www.python.org/doc/>
- Docker Inc. (2023). What is Docker? Docker Documentation. Retrieved from  
<https://www.docker.com/what-is-docker>
- Python Cryptography Toolkit. (2023). Cryptography Documentation. Cryptography.io. Retrieved from <https://cryptography.io/en/latest/>

## ເອກສາຮ້າງອິງ(ຕ່ອ)

Requests Documentation. (2023). Requests: HTTP for Humans. Requests. Retrieved from <https://requests.readthedocs.io/en/latest/>

Python Software Foundation. (2023). socket — Low-level networking interface. Python Documentation. Retrieved from <https://docs.python.org/3/library/socket.html>

Micro Focus Enterprise (2023). Certificate and Key Formats. Retrieved December 12, 2024, from <https://www.microfocus.com/documentation/enterprise-developer/ed60/ED-Eclipse/BKCJCJCERTS001.html>

Cryptography (2024) Asymmetric Utilities. © Copyright 2013-2025, Individual Contributors. Retrieved December 20, 2024, from

<https://cryptography.io/en/latest/hazmat/primitives/asymmetric/utils/>

Authors in Security Domain. (ປີ້ມໍຮະບູ). SFM-KRS: A Secure and Flexible Key Recovery System for Multiple Agents. ນິຕະສາຮວິ່ນຍັດຕ້ານຄວາມປລອດກັບຂໍ້ມູນຄ.

Cybersecurity Journal. (ປີ້ມໍຮະບູ). Multi-Factor Authentication: A Survey and Implementation. Cybersecurity Journal.

"PKCE: Proof Key for Code Exchange" by Auth0. (ປີ້ມໍຮະບູ). Retrieved from <https://auth0.com/docs/security/advanced-security/pkce>

"PKCE and OAuth 2.0" by OAuth.com. (ປີ້ມໍຮະບູ). Retrieved from <https://oauth.com/2-0/pkce/>

## ประวัติผู้ดำเนินโครงการ



ชื่อ นายสุวิทย์ สายโส  
ภูมิลำเนา 66 หมู่ 1 ต.บ้านดี อ.หล่มสัก จ.เพชรบูรณ์  
67110

### ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนเมตตาวิทยา
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขา  
วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์  
มหาวิทยาลัยนเรศวร

E-mail: [suwit.saiso@gmail.com](mailto:suwit.saiso@gmail.com)



ชื่อ นายมนทร์บдинทร์ อนุเคราะห์  
ภูมิลำเนา 66/2 หมู่ 3 ต.ปากน้ำ อ.สوارคโลก จ.  
สุโขทัย 64110

### ประวัติการศึกษา

- จบระดับมัธยมศึกษาจาก BBZ Volklingen , Germany
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขา  
วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์  
มหาวิทยาลัยนเรศวร

E-mail: [minbodin.anukroh95@hotmail.com](mailto:minbodin.anukroh95@hotmail.com)

