รายงานวิจัยฉบับสมบูรณ์

โครงการ ระบบการรับส่งข้อความและรูปภาพที่เป็นความลับด้วยโทรศัพท์มือถือ สำหรับหน่วยงานที่ต้องการความมั่นคงของข้อมูล

Secure Mobile Messaging System for Organizations Requiring Information Safety

ณัฐนนท์ ลีลาตระกูล
ภารุจ รัตนวรพันธุ์
อุรีรัฐ สุขสวัสดิ์ชน
โกเมศ อัมพวัน
จักริน สุขสวัสดิ์ชน
สุนิสา ริมเจริญ
กรชวัล ชายผา
เอกภพ บุญเพ็ง
พีระศักดิ์ เพียรประสิทธิ์
พรภิรมย์ มั่นฤกษ์

โครงการวิจัยประเภทงบประมาณเงินรายได้ จากเงินอุดหนุนรัฐบาล (งบประมาณแผ่นดิน)
ประจำปีงบประมาณ พ.ศ. 2559
มหาวิทยาลัยบูรพา

รหัสโครงการ 2559A10802208
สัญญาเลขที่ 51/2559

รายงานวิจัยฉบับสมบูรณ์

โครงการ ระบบการรับส่งข้อความและรูปภาพที่เป็นความลับด้วยโทรศัพท์มือถือ สำหรับหน่วยงานที่ต้องการความมั่นคงของข้อมูล

Secure Mobile Messaging System for Organizations Requiring Information Safety

ณัฐนนท์ ลีลาตระกูล
ภารุจ รัตนวรพันธุ์
อุรีรัฐ สุขสวัสดิ์ชน
โกเมศ อัมพวัน
จักริน สุขสวัสดิ์ชน
สุนิสา ริมเจริญ
กรชวัล ชายผา
เอกภพ บุญเพ็ง
พีระศักดิ์ เพียรประสิทธิ์
พรภิรมย์ มั่นฤกษ์

คณะวิทยาการสารสนเทศ มหาวิทยาลัยบูรพา

ตุลาคม พ.ศ. ๒๕๕๙

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับทุนสนับสนุนการวิจัยจากงบประมาณเงินรายได้ จากเงินอุดหนุนรัฐบาล (งบประมาณแผ่นดิน) ประจำปีงบประมาณ พ.ศ. 2559 มหาวิทยาลัยบูรพา ผ่านสำนักงานคณะกรรมการการวิจัยแห่งชาติ เลขที่สัญญา 51/2559

Acknowledgment

This work was financially supported by the Research Grant of Burapha University through National Research Council of Thailand (Grant no. 51/2559).

บทคัดย่อ

งานวิจัยนี้นำเสนอระบบการรับส่งข้อความและรูปภาพที่เป็นความลับด้วยโทรศัพท์มือถือ สำหรับหน่วยงานที่ต้องการความมั่นคงของข้อมูล โดยคณะผู้วิจัยได้ทดสอบแชทแอปพลิเคชันบนสมาร์ทโฟนที่มีชื่อว่า “Signal” ซึ่งเป็นแอปพลิเคชันที่มีความปลอดภัยสูง และยังเป็นแอปพลิเคชันประเภท Open Source ซึ่งสามารถนำมาปรับเปลี่ยนให้เหมาะกับการส่งข้อความแชทภายในองค์กร (บางองค์กรต้องการความปลอดภัยในการติดต่อสื่อสารกันผ่านแชทแอปพลิเคชันบนสมาร์ทโฟน และเครื่องแม่ข่ายที่อยู่ในองค์กรนั้น ๆ) ทั้งนี้คณะผู้วิจัยได้เลือกโปรแกรมทดสอบที่มีมาตรฐานจากบริษัทโทรคมนาคมขนาดใหญ่ AT&T เพื่อทดสอบประสิทธิภาพด้านต่าง ๆ ของแอปพลิเคชัน เครื่องมือที่นำมาทดสอบคือ “application resource optimizer (ARO)” โดย ARO สามารถรายงานผลการทดสอบในด้านต่าง ๆ ได้อย่างครอบคลุม

นอกจากนี้งานวิจัยนี้ยังได้มีการพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) โดยนำการสแกนลายนิ้วมือ และการใส่ Pattern มาใช้เป็นรูปแบบในการยืนยันตัวตน เพื่อป้องกันผู้ไม่หวังดีหรือบุคคลอื่นที่ไม่ใช่เจ้าของสมาร์ทโฟน ไม่ให้เข้าถึงหรือเข้าใช้งานแชทแอปพลิเคชัน (Signal) โดยไม่ได้รับอนุญาตจากเจ้าของสมาร์ทโฟน

Abstract

This research presents a secure mobile messaging system for organizations requiring information safety. We test results of a highly secure mobile chat app, called “Signal”. Signal is an Open Source application, which we can modify suit to send chat messages within private organizations, that need more security. In this research, we use “ARO”, a mobile app testing application, developed by AT&T, to test Signal (regarding energy usage and security issues). We found that “ARO” can report testing result comprehensively.

In addition, we developed an additional authentication feature in a chat application (i.e., Signal). The feature includes fingerprint scanning and pattern input, aiming to protect impersonation breaches. Specifically, the feature helps protecting attackers from an access to the chat application (Signal) without permission from owners of smartphones.

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันหน่วยงานต่าง ๆ ในประเทศไทย เช่นกองทัพบกยังไม่มีแชทแอปพลิเคชันที่ใช้สำหรับการสื่อสารบนสมาร์ทโฟนที่ดีของหน่วยงานเอง (ทำให้ข้อความที่ต้องส่งผ่านแชทแอปพลิเคชันอื่น ต้องเดินทางผ่านเครือข่ายภายนอกองค์กร และมีความเสี่ยงต่อการถูกโจรกรรม) อย่างไรก็ตามมีผู้พัฒนาแชทแอปพลิเคชันที่มีความปลอดภัย และเป็น Open Source ที่มีประสิทธิภาพดีอยู่แล้ว ข้อดีของ Open Source คือผู้ที่นำแอปพลิเคชันมาพัฒนาต่อ สามารถตั้งระบบของตัวเองได้ไม่ต้องพึ่งพาเครื่องแม่ข่ายของผู้ให้บริการที่อยู่นอกองค์กร ทำให้ผู้ใช้รู้สึกปลอดภัยจากการคุกคาม เนื่องจากข้อมูลรั่วไหลออกไปภายนอกได้ยากขึ้น ผู้วิจัยจึง 1) ทดสอบแชทแอปพลิเคชันที่ได้เลือกมา ชื่อว่า Signal และ 2) พัฒนาส่วนการยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) เนื่องจากการนำแชทแอปพลิเคชัน (Signal) ไปใช้งานในกองทัพบก ข้อความการสนทนาต้องเก็บเป็นความลับ บุคคลที่ไม่ใช่เจ้าของสมาร์ทรวมถึงบุคคลที่ไม่ได้รับอนุญาต ไม่ควรสามารถที่จะเข้าใช้งานแชทแอปพลิเคชัน (Signal) ได้

การทดสอบ Signal จะเป็นการทดสอบภายใต้ข้อกำหนดตาม Best Practices ที่บริษัทโทรคมนาคมขนาดใหญ่ของสหรัฐ (AT&T) ได้แนะนำ โดยผู้วิจัยเน้นด้านของความปลอดภัยเป็นสำคัญ และจะพัฒนาส่วนยืนยันตัวตนเพิ่มเติม โดยใช้วิธีการยืนยันตัวตนในรูปแบบของการสแกนลายนิ้วมือ และการใส่ Pattern มาประยุกต์ใช้ในแชทแอปพลิเคชัน (Signal) เพื่อเป็นการเพิ่มความปลอดภัยก่อนการเข้าใช้งานแชทแอปพลิเคชัน (Signal)

ดังนั้นหากแชทแอปพลิเคชัน (Signal) นั้นผ่านการทดสอบโดย ARO ภายใต้ข้อกำหนดตาม Best Practices และผ่านการพัฒนาส่วนยืนยันตัวตนเพิ่มเติม หมายความว่าแชทแอปพลิเคชันนั้นมีความปลอดภัยในระดับที่เป็นมาตรฐาน เหมาะแก่การนำมาใช้งานได้ และอยู่ในระดับที่เชื่อถือได้ และรวมถึงการป้องกันผู้ไม่หวังดีหรือบุคคลอื่นที่ไม่ใช่เจ้าของสมาร์ทโฟน ไม่ให้เข้าถึงหรือเข้าใช้งานแชทแอปพลิเคชัน (Signal) โดยไม่ได้รับอนุญาตจากเจ้าของสมาร์ทโฟน

1.2 วัตถุประสงค์ของโครงการวิจัย

เพื่อหาวิธีการที่จะทำให้การรับส่งข้อความผ่านโทรศัพท์มือถือมีความปลอดภัยจากการถูกละเมิดมาตรการรักษาความปลอดภัยข้อมูลข่าวสาร และเป็นวิธีการที่มีมาตรฐานที่ยอมรับได้ ดังรายละเอียดต่อไปนี้

1. เพื่อศึกษาและส่งเสริมการสร้างองค์ความรู้ที่เป็นพื้นฐานของอุปกรณ์โทรศัพท์มือถือแบบสมาร์ทโฟนที่ใช้ระบบปฏิบัติการแอนดรอยด์ สำหรับใช้ในการพัฒนาต้นแบบอุปกรณ์สำหรับการรับส่งข้อมูลที่เป็นความลับ
2. เพื่อศึกษาและส่งเสริมการสร้างองค์ความรู้ที่เป็นพื้นฐานของอุปกรณ์สแกนลายนิ้วมือที่สามารถเชื่อมต่อกับโทรศัพท์มือถือแอนดรอยด์ สำหรับใช้ในการพัฒนาต้นแบบอุปกรณ์สำหรับการยืนยันตัวตนของผู้ถือโทรศัพท์
3. เพื่อศึกษาและส่งเสริมการสร้างองค์ความรู้ที่เป็นพื้นฐานของเครื่องแม่ข่ายให้บริการเครือข่ายส่วนตัวเสมือน สำหรับเพิ่มชั้นป้องกันบุคคลภายนอกไม่ให้รุกล้ำเข้ามาในระบบ
4. เพื่อศึกษาและส่งเสริมการสร้างองค์ความรู้ที่เป็นพื้นฐานของเครื่องแม่ข่ายที่สร้างรหัสผ่านแบบใช้ครั้งเดียว (One Time Password, OTP) สำหรับป้องกันการปลอมตัวตน และ ป้องกันการโจมตีแบบส่งข้อมูลซ้ำ
5. เพื่อศึกษาและส่งเสริมการสร้างองค์ความรู้ที่เป็นพื้นฐานของเครื่องแม่ข่ายให้บริการรับส่งข้อมูล (Messaging Management System, MMS)
6. เพื่อให้สามารถตรวจสอบและบ่งชี้ถึงช่องโหว่ด้านความปลอดภัยของการใช้โปรแกรมประยุกต์บนโทรศัพท์มือถือที่มีระบบปฏิบัติการแอนดรอยด์ และ ของเครื่องแม่ข่ายให้บริการรับส่งข้อมูล
7. เพื่อปรับปรุงประสิทธิภาพด้านความปลอดภัยของการใช้โปรแกรมประยุกต์บนโทรศัพท์มือถือที่มีระบบปฏิบัติการแอนดรอยด์ในการส่งข้อมูลที่เป็นความลับ
8. เพื่อปรับปรุงประสิทธิภาพด้านความปลอดภัยของเครือข่ายและเครื่องแม่ข่ายต่าง ๆ
9. เพื่อปรับปรุงประสิทธิภาพของเครือข่ายและเครื่องแม่ข่ายต่าง ๆ ในด้านการรองรับผู้ใช้จำนวนมาก
10. พัฒนาเทคโนโลยีที่ช่วยการส่งข้อมูลที่เป็นความลับได้อย่างปลอดภัยและมีประสิทธิภาพ ซึ่งตั้งอยู่บนฐานองค์ความรู้ (knowledge-based innovation and technology) และ สามารถนำไปใช้ประโยชน์ได้จริงอย่างเต็มศักยภาพ
11. ทำการทดสอบการใช้ได้จริงที่หน่วยงานทหารของไทย เช่น ที่หน่วยศูนย์กลางไซเบอร์กองทัพบก
12. เพื่อให้ผู้ที่สนใจสามารถนำแนวความคิดที่ได้นำเสนอ ไปทำการพัฒนาหรือประยุกต์ใช้ในหน่วยงานของตนเองต่อไป

1.3 ขอบเขตของโครงการวิจัย

1. ทดสอบแชทแอปพลิเคชันบนสมาร์ทโฟนที่มีชื่อว่า Signal Version 4.3.1
2. ทดสอบโดยใช้โปรแกรม ARO ที่พัฒนาจากบริษัท AT&T
3. ทดสอบในขณะที่แชทแอปพลิเคชันทำงานบนระบบปฏิบัติการแอนดรอยด์ 6.0
4. ในขณะทดสอบสมาร์ทโฟนต้องเปิดโหมดเป็นผู้พัฒนา เนื่องจากโปรแกรม ARO จะต้องทำการติดตั้งโปรแกรมที่ใช้ดัก Packet ที่รับ-ส่งบนสมาร์ทโฟนเพื่อนำไปวิเคราะห์
5. สมาร์ทโฟนที่ใช้ต้องสามารถเชื่อมต่อ 3G/LTE และ Wi-Fi ได้
6. สมาร์ทโฟนต้องยินยอมการขอสิทธิเข้าถึงต่าง ๆ ที่เป็นความต้องการของแชทแอปพลิเคชันได้
7. พัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน Signal Version 3.27.1
8. พัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) โดยใช้โปรแกรม Android studio
9. พัฒนาส่วนยืนยันตัวตนเพิ่มเติมในขณะที่แชทแอปพลิเคชัน (Signal) ทำงานบนระบบปฏิบัติการแอนดรอยด์ 6.0
10. ในขณะพัฒนาส่วนยืนยันตัวตนเพิ่มเติม สมาร์ทโฟนต้องเปิดโหมดเป็นผู้พัฒนา เนื่องจากโปรแกรม Android studio จะต้องทำการติดตั้งแอปพลิเคชันลงบนสมาร์ทโฟน สำหรับให้ผู้พัฒนาใช้งานหรือเพื่อทดสอบแอปพลิเคชัน

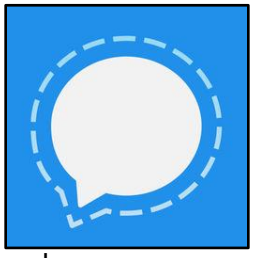
1.4 ประโยชน์ที่ได้รับ

1. ได้ระบบที่ใช้งานได้จริงในกองทัพและมีกระบวนการทำให้การรับส่งข้อความและรูปผ่านโทรศัพท์มือถืออย่างมีความปลอดภัยและมีมาตรฐานที่ยอมรับได้
2. กำลังพลผู้ปฏิบัติงานในการรับส่งข่าวสารมีความมั่นใจในการรักษาความลับของข้อมูลข่าวสาร
3. หน่วยสามารถนำข่าวสารที่รับส่งผ่านระบบมาดำเนินกรรมวิธีเพื่อนำไปใช้ในภารกิจของหน่วยทั้งในด้านเชิงรับและเชิงรุกได้อย่างมีประสิทธิภาพ
4. หน่วยสามารถกระจายข่าวสารที่มีชั้นความลับได้อย่างรวดเร็วและทั่วถึง
5. แหล่งข่าวมีความปลอดภัย เนื่องจากข้อมูลผู้ส่งไม่รั่วไหล
6. ระบบที่นำเสนอจะสามารถถูกใช้เป็นต้นแบบเพื่อพัฒนาต่อยอดได้

บทที่ 2
เนื้อเรื่อง

2.1 Signal (Private Messenger)

Signal - Private Messenger เป็นโปรแกรมประยุกต์หรือแอปพลิเคชันที่มีความปลอดภัยสูง (แม้แต่ Edwords Snowden ได้ใช้ App นี้เพื่อส่งข้อความอย่างปลอดภัย) ถูกพัฒนาโดยองค์กร Open Whisper System (OWS) เป็นซอฟต์แวร์ที่ไม่มีค่าใช้จ่าย และเป็นแอปพลิเคชันแบบ Open Source สำหรับสมาร์ทโฟนที่เป็นทั้ง Android, iOS นอกจากนี้ Signal ยังทำงานได้บนเดสก์ท็อป Signal ใช้การเข้ารหัสแบบต้นทางถึงปลายทาง ช่วยให้ผู้ใช้สามารถส่งข้อความตัวหนังสือ รูปภาพ และวิดีโอในแบบกลุ่ม และยังเข้ารหัสบทสนทนาทางโทรศัพท์ระหว่างผู้ใช้ Signal ได้ ถึงแม้ว่า Signal จะใช้หมายเลขโทรศัพท์ในการติดต่อ แต่จริง ๆ แล้วการโทร และข้อความต่าง ๆ จะใช้การเชื่อมต่อข้อมูลอินเทอร์เน็ต ดังนั้นผู้สนทนาทั้งสองฝ่ายต้องสามารถใช้งานอินเทอร์เน็ตได้บนอุปกรณ์มือถือของตน ด้วยเหตุผลนี้จึงไม่มีค่าใช้จ่ายในการส่งข้อความ SMS และ MMS เกิดขึ้นกับผู้ใช้งาน Signal



รูปที่ 2-1 Signal Icon

จากบทความของ PCMag “Only 6 Messaging Apps Are Truly Secure” ได้เขียนไว้ว่า ผู้ใช้โทรศัพท์มือถือต้องการ การรับส่งข้อความที่มีความปลอดภัยจากการทดสอบขององค์กร Electronic Frontier Foundation (EFF) ได้ให้การยอมรับแอปพลิเคชันที่มีความปลอดภัย 6 แอปพลิเคชัน

โดย EFF ได้ทดสอบแอปพลิเคชันต่าง ๆ ถึง 39 แอปพลิเคชัน มีทั้งเครื่องมือยอดนิยมอย่าง Apple, Google, Facebook, BlackBerry, Microsoft และ Yahoo พบว่าทั้ง 39 แอปพลิเคชัน มีการเข้ารหัสในระหว่างการรับส่งข้อมูล และส่วนมากไม่ผ่านการทดสอบด้านความปลอดภัยของ EFF โดยมี

เพียง 6 แอปพลิเคชันเท่านั้นที่ผ่านการทดสอบทั้งหมด 7 ข้อ และถูกรับรองว่าปลอดภัย คือ ChatSecure + Orbot, Cryptocat, RedPhone, Silent Phone, Silent Text และ TextSecure (Signal) ในขณะที่แอปพลิเคชันยอดนิยมต่าง ๆ ส่วนมากจะผ่านแค่เพียง 2 ข้อ โดยมีรายละเอียดดังตารางที่ 2-1

ตารางที่ 0-1 ตารางทดสอบของ EFF

All Tools	เข้ารหัสข้อมูลก่อนส่ง	เข้ารหัสโดยที่ Provider ไม่สามารถถอดรหัสได้	สามารถพิสูจน์ Identities ของ Contact ได้	การสื่อสารก่อนหน้าปลอดภัยเมื่อ Key ถูกขโมยไป	open source code	มีคู่มือ	Code มันมีการ audit แล้ว
AIM							
BlackBerry Messenger							
BlackBerry Protected							
ChatSecure + Orbot							
Cryptocat							
RedPhone							
Silent Phone							
Silent Text							
Signal (TextSecure)							

เว็บไซต์ Security-in-a-box ที่เป็นเว็บไซต์ด้านความปลอดภัยที่เกิดจากการร่วมกันระหว่าง Tactical Technology Collective และ Front Line Defenders องค์กร NGO ด้านความปลอดภัยของไอที และสิทธิมนุษยชนเผยแพร่บทความพิเศษที่เขียนโดย Maria Xynou และ Chris Walker ผู้เชี่ยวชาญของ Tactical Technology Collective โดยระบุว่าแอปพลิเคชันสนทนาเข้ารหัสอย่าง Signal นั้นยังเหนือว่า WhatsApp และแนะนำให้ทุกคนใช้ติดต่อสื่อสารกัน ทั้งสองคนให้เหตุผลไว้ 4 ประเด็นหลักคือ

2.1.1 ตัวแอปพลิเคชัน Signal เป็น Open Source ขณะที่ WhatsApp เป็นระบบปิด ทำให้ไม่มีทางรู้ว่าทาง WhatsApp นำโปรโตคอลเข้ารหัสไปใช้ (implement) อย่างไร

2.1.2 Signal เข้ารหัสข้อมูลการสนทนาทั้งหมดรวมถึงข้อมูลที่เก็บไว้บนเครื่อง แต่ WhatsApp ยังไม่มีคุณสมบัตินี้

2.1.3 แม้ในการแชททั้ง Signal และ WhatsApp ต่างต้องมีการยืนยันตัวตนผ่าน digital fingerprint แต่ในเรื่องของการใช้เสียง WhatsApp อิงกับ fingerprint แต่ Signal ต้องให้ผู้ใช้แต่ละฝ่ายอ่านคำสำคัญเพื่อยืนยันตัวตน (เพราะแม้แต่เสียงก็ปลอมกันได้) Signal จึงลดความเสี่ยงในการโจมตีแบบ MITM (Man-in-the-Middle) ได้ลงไปพอสมควร

2.1.4 ทั้งสองแอปพลิเคชันแม้จะเข้ารหัสข้อความแชทเหมือนกัน แต่ทั้งคู่ไม่ได้เข้ารหัสส่วนข้อมูลที่เป็น metadata อย่างเช่นติดต่อกับใคร เวลาไหน จากที่ไหน เจ้าของ Signal คือ Open Whisper Systems ไม่ได้มีโมเดลธุรกิจอิงกับการโฆษณา ผิดกับ WhatsApp ที่เจ้าของคือ Facebook และมักจะทำประวัติ (Profiling) เอาไว้สำหรับการโฆษณา เรื่องนี้ย่อมทำให้เกิดความเสี่ยงที่จะระบุตัวตนได้จากข้อมูลแวดล้อมอื่น ๆ เช่น คนที่ติดต่อ ตำแหน่ง สถานที่ อุปกรณ์ หรือแม้กระทั่ง IP Address ด้วยปัจจัยทั้งหมดนี้ ทำให้ทั้งสองยังเลือกที่จะแนะนำให้ใช้ Signal ในการสื่อสารต่อไป

2.2 Application Resource Optimizer (ARO)

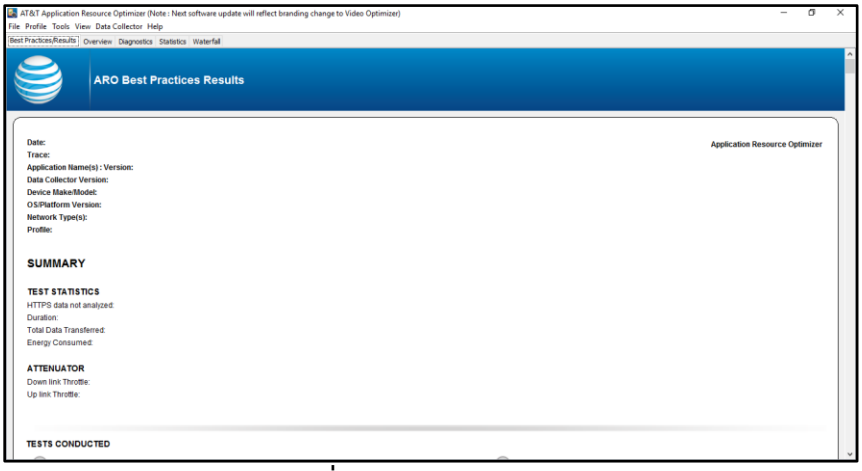
AT&T Application Resource Optimizer (ARO) เป็นเครื่องมือ Open Source สำหรับการนักพัฒนา และนักทดสอบที่ให้คำแนะนำเพื่อเพิ่มประสิทธิภาพการทำงานของแอปพลิเคชันบนมือถือ ทำให้การใช้งานแบตเตอรี่มีประสิทธิภาพมากขึ้น และลดการใช้ข้อมูล

ARO ประกอบด้วย 2 ส่วน คือ การรวบรวมข้อมูล (Data Collector) และการวิเคราะห์ข้อมูล (Data Analyzer) การรวบรวมข้อมูล จะรวบรวมข้อมูลที่เป็น วีดีโอ ข้อมูล ติดตามอุปกรณ์ ที่มีการโต้ตอบกับแอปพลิเคชัน การวิเคราะห์ข้อมูลวัดจากการติดตามจากเครื่อง รวมไปถึงไฟล์ PCAP (ไฟล์เป็น Wireshark, WinDump, Tcpdump ฯลฯ) จากเครื่องมือทดสอบอื่น ๆ กับแนวทางปฏิบัติที่ดีสำหรับมือ

ถือ 24 ข้อในเว็บไซต์ของบริษัท และระบุแนวทางเพื่อปรับปรุงประสิทธิภาพของแอปพลิเคชันที่พัฒนา ทั้งนี้แนวทางปฏิบัติที่ดี (Best Practices) ของมือถือจะช่วยปรับปรุงประสิทธิภาพของแอปพลิเคชันอย่างมาก เช่น การ Cache และการจัดการ connection เครือข่าย ช่วยลดเวลารับส่งข้อมูล และประหยัดการใช้พลังงาน

ARO ช่วยทดสอบตามมาตรฐาน ซึ่งเป็นการประกันคุณภาพ และ AT&T แนะนำให้ผู้พัฒนาดำเนินตามชุดการทดสอบตามมาตรฐาน เช่น เกณฑ์การทดสอบของ AQuA (http://www.appqualityalliance.org/resources)

รูปที่ 2-2 ARO Icon



รูปที่ 2-3 โปรแกรม ARO

2.2.1 ไฟล์ดาวน์โหลด (File Download)

Mobile App ที่ดีควรมีการจัดการ Cache เพื่อลดเนื้อหาที่ซ้ำกัน ควรบีบอัดข้อมูล และมีจัดการวิธีดาวน์โหลดไฟล์ เพื่อช่วยลดการใช้แบตเตอรี่ของแอปพลิเคชัน

2.2.1.1 Text File Compression ให้รายละเอียดเกี่ยวกับวิธีการต่าง ๆ สำหรับการบีบอัด และให้คำแนะนำในการใช้การบีบอัดไฟล์ข้อความเพื่อให้หน้าแอปพลิเคชันของคุณแสดงผลได้เร็วขึ้น

2.2.1.2 Duplicate Content อธิบายว่าเนื้อหาที่ซ้ำซ้อนจะมีผลต่อแอปพลิเคชัน และเสนอคำแนะนำในการพัฒนากลยุทธ์ Cache เพื่อลดเนื้อหาที่ซ้ำกัน

2.2.1.3 Cache Control อธิบาย Cache และการควบคุม Cache ที่ระบุไว้ใน Protocol HTTP 1.1 โดยกล่าวถึงเหตุผลที่ Cache มีความสำคัญ และให้คำแนะนำในการใช้Cache ในแอปพลิเคชัน

2.2.1.4 Cache Expiration อธิบายถึงส่วนประกอบของรูปแบบการหมดอายุที่ระบุไว้ใน Protocol HTTP 1.1 ให้ตัวอย่างของสิ่งที่อาจเกิดขึ้นเมื่อโมเดลหมดอายุ และไม่ได้ใช้อย่างถูกต้อง อีกทั้งให้คำแนะนำเกี่ยวกับวิธีใช้ประโยชน์จากกลไกการหมดอายุของ Cache ในแอปพลิเคชัน

2.2.1.5 Content Pre-fetching ดูว่าการดึงข้อมูลล่วงหน้าทำงานอย่างไรอธิบายประเด็นที่เกี่ยวข้องกับการจัดการเนื้อหา และให้คำแนะนำเกี่ยวกับวิธีใช้การดึงข้อมูลล่วงหน้าในแอปพลิเคชัน

2.2.1.6 Resize Images for Mobile อธิบายถึงข้อดี และข้อเสียของวิธีการต่าง ๆ ในการปรับขนาดแสดงภาพ และให้คำแนะนำในการพิจารณาว่าภาพใดที่ก่อให้เกิดความไร้ประสิทธิภาพและสามารถปรับขนาดสำหรับมือถือที่ได้

2.2.1.7 Image Compression ดูวิธีการปรับคุณภาพของภาพ และขนาดภาพเพื่อให้ประสบการณ์การใช้งานของผู้ใช้ได้ดียิ่งขึ้น

2.2.1.8 Image Metadata กล่าวถึงวิธีการลดข้อมูล metadata ของรูปภาพโดยไม่มีผลกระทบต่อคุณภาพของภาพ

2.2.2 การเชื่อมต่อ (Connections)

Mobile App ที่ดีควรมีการจัดการ Connection ของ TCP อย่างมีประสิทธิภาพ การเพิ่มประสิทธิภาพการใช้งานการส่งสัญญาณของอุปกรณ์ ทำให้คุณสามารถเพิ่มความเร็วในการส่งข้อมูล และลดการใช้แบตเตอรี่ของแอปพลิเคชันได้

2.2.2.1 Opening Connections อธิบายถึงผลกระทบที่การเปิดการเชื่อมต่อที่ไร้ประสิทธิภาพอาจมีผลกับแอปพลิเคชัน และเสนอคำแนะนำเกี่ยวกับวิธีเปิดการเชื่อมต่ออย่างมีประสิทธิภาพมากขึ้น

2.2.2.2 Multiple Simultaneous TCP Connections ดูว่าการเชื่อมต่อ TCP มีขึ้นอย่างไรเพื่อดูว่าเหตุใดการเชื่อมต่อแบบถาวรจึงมีประสิทธิภาพมากขึ้น ผู้พัฒนาแอปพลิเคชันควรตรวจสอบปัญหาเกี่ยวกับการเปิดการเชื่อมต่อแบบต่อเนื่องหลาย Connection พร้อมกัน และ AT&T เสนอคำแนะนำสำหรับการจัดการการเชื่อมต่ออย่างชาญฉลาดในแอปพลิเคชันบนอุปกรณ์เคลื่อนที่เพื่อจัดการกับปัญหานี้

2.2.2.3 Periodic Transfers อธิบายว่าการถ่ายโอนเป็นระยะ ๆ สามารถทำให้แอปพลิเคชันสิ้นเปลืองพลังงานได้อย่างไร และ AT&T เสนอคำแนะนำเกี่ยวกับวิธีใช้การถ่ายโอนข้อมูลเป็นระยะอย่างมีประสิทธิภาพเพื่อให้มีผลกระทบน้อยที่สุดต่อประสบการณ์ของผู้ใช้ และทรัพยากรของเครือข่าย

2.2.2.4 Screen Rotations ดูวิธีการใช้การหมุนหน้าจอในแอปพลิเคชันบนอุปกรณ์เคลื่อนที่ AT&T อธิบายปัญหาเกี่ยวกับการจัดการการหมุนหน้าจอ และให้คำแนะนำเกี่ยวกับวิธีจัดการการหมุนหน้าจอได้อย่างมีประสิทธิภาพมากขึ้น (บาง App ส่งข้อมูลผ่านเครือข่ายทุกครั้งทั้ง ๆ ที่ไม่จำเป็น)

2.2.2.5 Closing Connections อธิบายถึงปัญหาที่เกิดขึ้นเมื่อการเชื่อมต่อ TCP ไม่ได้ถูกปิดโดยเจตนา และให้คำแนะนำในการปิดการเชื่อมต่ออย่างมีประสิทธิภาพ

2.2.2.6 Offloading to Wi-Fi กล่าวถึงค่าใช้จ่าย และประโยชน์ของการใช้ Wi-Fi ใน C แทนที่จะใช้ 3G

2.2.3 HTML

เอกสารเหล่านี้มีคำแนะนำสำหรับการเพิ่มประสิทธิภาพการจัดการภายใน HTML ของผู้พัฒนา การเพิ่มประสิทธิภาพการเชื่อมต่อ HTTP ของผู้พัฒนาจะช่วยลดการใช้แบตเตอรี่ของแอปพลิเคชันได้อีกด้วย

2.2.3.1 HTTP 400 and 500 Status Codes อธิบายเกี่ยวกับลำดับชั้นของรหัสสถานะ HTTP ที่แตกต่างกันอธิบายปัญหาที่เกิดขึ้นโดยใช้รหัสสถานะ 400 และ 500 ที่พบมากที่สุด และแนะนำทางเลือกที่เป็นไปได้สำหรับพวกเขา

2.2.3.2 HTTP 300 Status Codes อธิบายเกี่ยวกับรหัสสถานะ HTTP ที่แตกต่างกัน อธิบายถึงปัญหาที่เกิดขึ้นโดยรหัสสถานะ 300 ที่พบมากที่สุด และแนะนำทางเลือกที่เป็นไปได้สำหรับพวกเขา

2.2.3.3 HTTP 1.0 Usage อธิบายถึงการปรับปรุงที่เพิ่มลงใน Protocol HTTP 1.1 และกล่าวถึงปัญหาที่แอปพลิเคชันที่ใช้ Protocol HTTP 1.0 รุ่นเก่าประสบอยู่

2.2.3.4 Unsecure SSL Version ให้รายละเอียดเกี่ยวกับ SSL Version ที่มีความปลอดภัยต่ำ และแจ้งเตือนแนะนำแนวทางในการปรับปรุงแก้ไข

2.2.3.5 Weak Cipher การเข้ารหัสในการรับส่งข้อมูลที่มีการเข้ารหัสด้วยรหัสง่าย ๆ ที่สามารถใช้เวลาถอดได้ไม่นานโดย ARO จะแจ้งเตือนแนะนำแนวทางในการปรับปรุงแก้ไข

2.2.3.6 Forward Secrecy เป็นตัวเลขที่ช่วยปกป้อง Session ก่อนหน้านี้หาก Session หนึ่งถูกแกะได้ นี่ก็ทำให้ Session ทั้งหมดก่อนหน้านี้ถูกแกะได้ จะมีการแนะนำ และแจ้งเตือนเพื่อป้องกันสิ่งเหล่านี้ที่จะเกิดขึ้นบนแอปพลิเคชันได้

2.2.3.7 HTTP Vs. HTTPS อธิบายประโยชน์ของการใช้ HTTPS สามารถป้องกันไม่ให้มีการอ่านข้อมูลที่มีการเข้ารหัสได้

2.2.4 หัวข้อทั่วไป (General Topics)

AT&T อธิบายข้อมูลพื้นฐานเกี่ยวกับ LTE และรุ่นพลังงาน 3G และการเพิ่มประสิทธิภาพแอปพลิเคชันของคุณสำหรับอุปกรณ์เคลื่อนที่

2.2.4.1 Private Data บอกเหตุผล และวิธีการปกป้องข้อมูลส่วนตัวของผู้ใช้ที่มีการส่งจากแอปพลิเคชันไปยังปลายทางได้อย่างไร

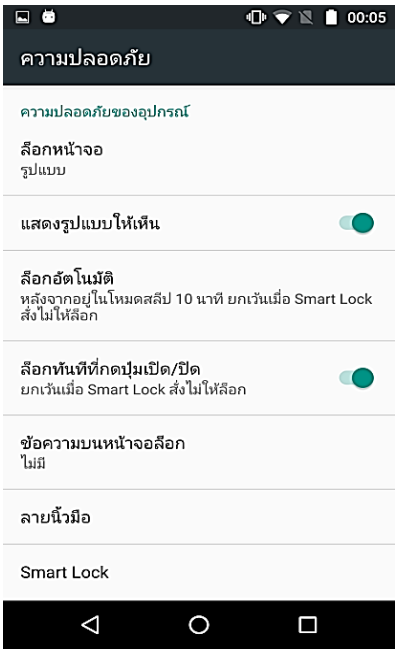
2.2.4.2 Accessing Peripherals ดูแอปพลิเคชันที่มีการขอเข้าถึงอุปกรณ์ต่อพ่วงประเภทต่าง ๆ และปัญหาที่เกี่ยวข้องกับการใช้งานอุปกรณ์เหล่านี้ และให้คำแนะนำในการใช้งานแอปพลิเคชันได้อย่างมีประสิทธิภาพ

2.2.4.3 Comparing LTE and 3G Energy Consumption อธิบายถึงความแตกต่างในการใช้พลังงานของ LTE และเครือข่าย 3G และแสดงให้ผู้ใช้เห็นว่า ARO แสดงความแตกต่างเหล่านี้เพื่อให้คุณทราบถึงประสิทธิภาพแอปพลิเคชันของคุณในทั้งสองเครือข่ายสัญญาณ

2.3 Fingerprint API

Google เป็นผู้พัฒนา Fingerprint API และให้ผู้ใช้ได้ใช้งานกันมาตั้งแต่ระบบปฏิบัติการแอนดรอยด์ 6.0 หรือ Marshmallow (API Level 23)

Fingerprint API นำ Fingerprint ของสมาร์ทโฟนนั้น ๆ มาประมวลผลโดยผู้ใช้ต้องเปิดการใช้งาน และเพิ่มลายนิ้วมือใน Settings ของสมาร์ทโฟนก่อน ดังแสดงในรูปที่ 2-4



รูปที่ 2-4 หน้าการเปิดใช้งาน และเพิ่มลายนิ้วมือใน Setting ของสมาร์ทโฟน

ขั้นตอนการทำ Fingerprint Authentication ด้วย Fingerprint API มีดังนี้

2.3.1 ให้ผู้ที่ต้องการศึกษาเริ่มสร้าง Project และเพิ่ม Fingerprint Authentication Permission ในไฟล์ AndroidManifest.xml ดังแสดงในรูปที่ 2-5

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.test.fingerpirnt.fingerprintdemo">

    <uses-permission
android:name="android.permission.USE_FINGERPRINT" />

    <application
       .....
    </application>

</manifest>
```

รูปที่ 2-5 เพิ่ม Fingerprint Authentication Permission ในไฟล์ AndroidManifest.xml**.** จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.2 ให้ผู้ที่ต้องการศึกษาประกาศตัวแปรของ FingerprintManager และ KeyguardManager ลงในไฟล์ MainActivity.java ดังแสดงในรูปที่ 2-6

```
public class MainActivity extends AppCompatActivity {

...
   private FingerprintManager mFingerprintManager;
   private KeyguardManager mKeyguardManager;
   ...

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mKeyguardManager = (KeyguardManager)
getSystemService(KEYGUARD_SERVICE);
        mFingerprintManager = (FingerprintManager)
getSystemService(FINGERPRINT_SERVICE);

...
```

รูปที่ 2-6 ประกาศตัวแปรของ FingerprintManager และ KeyguardManager ลงในไฟล์ MainActivity.java. จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.3 ผู้ที่ต้องการศึกษาทำการตรวจสอบ Fingerprint support โดยใช้ Method ดังแสดงในรูปที่ 2-7 โดยที่

isHardwareDetected() คือ การตรวจสอบสมาร์ทโฟนว่ามี sensor รองรับการสแกนลายนิ้วมือหรือไม่

isKeyguardSecure() คือ การตรวจสอบสมาร์ทโฟนว่าได้มีการเปิดใช้งานการ Lock Screen หรือไม่

hasEnrolledFingerprints() คือ การตรวจสอบสมาร์ทโฟนว่าได้มีการลงทะเบียนลายนิ้วมือไว้ในสมาร์ทโฟนหรือไม่

```
   ...

// Checking Device support fingerprint or not.
if (mFingerprintManager.isHardwareDetected()) {

    // Not setting lock screen with imprint.
    if (!mKeyguardManager.isKeyguardSecure()) {
        Toast.makeText(this,
                "Lock screen security not enabled in Settings",
                Toast.LENGTH_LONG).show();
        return;
    }

    // Not registered at least one fingerprint in Settings.
    if (!mFingerprintManager.hasEnrolledFingerprints()) {
        Toast.makeText(this,
                "Register at least one fingerprint in Settings",
                Toast.LENGTH_LONG).show();
        return;
    }

    ...
}
```

รูปที่ 2-7 Method สำหรับตรวจสอบ Fingerprint support ของสมาร์ทโฟน. จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.4 เมื่อผู้ที่ต้องการศึกษามีการเรียกใช้ isHardwareDetected() (ในข้อ 2.3.3) Android Studio จะบังคับให้เพิ่ม Code ส่วน Fingerprint permission ลงไปในไฟล์ MainActivity.java ดังแสดงในรูปที่ 2-8

```
// Checking fingerprint permission.
if (ActivityCompat.checkSelfPermission(this,
        Manifest.permission.USE_FINGERPRINT) !=
        PackageManager.PERMISSION_GRANTED) {
    Toast.makeText(this,
            "Fingerprint authentication permission not enabled",
            Toast.LENGTH_LONG).show();
    return;
}
```

รูปที่ 2-8 Code ที่ถูกบังคับให้เพิ่มเมื่อเรียกใช้ isHardwareDetected(). จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.5 ผู้ที่ต้องการศึกษาทำการ createKey เพื่อเข้าถึง secure storage ของสมาร์ทโฟน ดังแสดงในรูปที่ 2-9, 2-10 และ 2-11

```
//Alias for our key in the Android Key Store
private static final String KEY_NAME = "key_name";

private KeyStore mKeyStore;
private KeyGenerator mKeyGenerator;
```

รูปที่ 2-9 กำหนดค่าตั้งต้นสำหรับ createKey เพื่อเข้าถึง secure storage ของสมาร์ทโฟน. จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.6 ผู้ที่ต้องการศึกษาสร้าง Method ชื่อ generateKey() เพื่อสร้าง key ดังแสดงในรูปที่ 2-10

```
protected void generateKey() {
    try {
        mKeyStore = KeyStore.getInstance("AndroidKeyStore");
    } catch (Exception e) {
        e.printStackTrace();
    }

    try {
        mKeyGenerator = KeyGenerator.getInstance(
                KeyProperties.KEY_ALGORITHM_AES,
                "AndroidKeyStore");
    } catch (NoSuchAlgorithmException |
            NoSuchProviderException e) {
        throw new RuntimeException(
                "Failed to get KeyGenerator instance", e);
    }

    try {
        mKeyStore.load(null);
        mKeyGenerator.init(new
                KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_ENCRYPT |
                        KeyProperties.PURPOSE_DECRYPT)
                .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                .setUserAuthenticationRequired(true)
                .setEncryptionPaddings(
                        KeyProperties.ENCRYPTION_PADDING_PKCS7)
                .build());
        mKeyGenerator.generateKey();
    } catch (NoSuchAlgorithmException |
            InvalidAlgorithmParameterException
            | CertificateException | IOException e) {
        throw new RuntimeException(e);
    }

}
```

รูปที่ 2-10 Code ภายใน Method ชื่อ generateKey(). จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.7 ผู้ที่ต้องการศึกษาสร้าง Method ชื่อ initCipher() เพื่อเตรียมการเข้ารหัส ดังแสดงในรูปที่ 2-11

```
private boolean initCipher() {
    try {
        cipher = Cipher.getInstance(
                KeyProperties.KEY_ALGORITHM_AES + "/"
                        + KeyProperties.BLOCK_MODE_CBC + "/"
                        + KeyProperties.ENCRYPTION_PADDING_PKCS7);
    } catch (NoSuchAlgorithmException |
            NoSuchPaddingException e) {
        throw new RuntimeException("Failed to get Cipher", e);
    }

    try {
        mKeyStore.load(null);
        SecretKey key = (SecretKey) mKeyStore.getKey(KEY_NAME,
null);
        cipher.init(Cipher.ENCRYPT_MODE, key);
        return true;
    } catch (KeyPermanentlyInvalidatedException e) {
        return false;
    } catch (KeyStoreException | CertificateException |
UnrecoverableKeyException | IOException
            | NoSuchAlgorithmException | InvalidKeyException e) {
        throw new RuntimeException("Failed to init Cipher", e);
    }
}
```

รูปที่ 2-11 Code ภายใน Method ชื่อ initCipher(). จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.8 ผู้ที่ต้องการศึกษาสร้าง Class ชื่อ FingerprintHelper เพื่อใช้ตรวจสอบลายนิ้วมือของผู้ใช้ว่าตรงกับลายนิ้วมือที่ได้ลงทะเบียนไว้ในสมาร์ทโฟนหรือไม่ ดังแสดงในรูปที่ 2-12

```
public class FingerprintHelper extends
FingerprintManager.AuthenticationCallback {

    private CancellationSignal cancellationSignal;
    private Context context;

    public FingerprintHelper(Context context) {
        this.context = context;
    }

    public void startAuth(FingerprintManager manager,
                          FingerprintManager.CryptoObject
cryptoObject) {

        cancellationSignal = new CancellationSignal();

        if (ActivityCompat.checkSelfPermission(context,
                Manifest.permission.USE_FINGERPRINT) !=
                PackageManager.PERMISSION_GRANTED) {
            return;
        }
        manager.authenticate(cryptoObject, cancellationSignal, 0,
this, null);
    }

    public void stopListening() {
        if (cancellationSignal != null) {
            cancellationSignal.cancel();
            cancellationSignal = null;
        }
    }
```

รูปที่ 2-12 Code ภายใน Class ชื่อ FingerprintHelper. จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

```
    @Override
    public void onAuthenticationError(int errorCode, CharSequence
errString) {
        super.onAuthenticationError(errorCode, errString);
        Log.e("FingerprintHelper", "onAuthenticationError:" +
errString );
    }


    @Override
    public void onAuthenticationHelp(int helpCode, CharSequence
helpString) {
        super.onAuthenticationHelp(helpCode, helpString);
        Toast.makeText(context,
                "Authentication help\n" + helpString,
                Toast.LENGTH_LONG).show();
    }

    @Override
    public void
onAuthenticationSucceeded(FingerprintManager.AuthenticationResult
result) {
        super.onAuthenticationSucceeded(result);
        Toast.makeText(context,
                "Authentication succeeded.",
                Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAuthenticationFailed() {
        super.onAuthenticationFailed();
        Toast.makeText(context,
                "Authentication failed.",
                Toast.LENGTH_LONG).show();
    }
}
```

รูปที่ 2-13 Code ภายใน Class ชื่อ FingerprintHelper (ต่อ). จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.9 ผู้ที่ต้องการศึกษาเพิ่ม Code ใน onResume() และ onPause() ในไฟล์ MainActivity.java เพื่อใช้จัดการ FingerprintHelper ว่าจะให้หยุดทำงาน (เมื่อ onPause) หรือทำงานต่อ (เมื่อ onResume) ดังแสดงในรูปที่ 2-14

```
@Override
protected void onResume() {
    super.onResume();
    if (mFingerprintHelper != null) {
        mFingerprintHelper.startAuth(mFingerprintManager,
mCryptoObject);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (mFingerprintHelper != null) {
        mFingerprintHelper.stopListening();
    }
}
```

รูปที่ 2-14 FingerprintHelper เริ่มทำงานใน onResume() และหยุดทำงานใน onPause() ในไฟล์ MainActivity.java. จาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

2.3.10 ผู้ที่ต้องการศึกษาเพิ่ม Code สำหรับการเรียกใช้งาน ดังแสดงในรูปที่ 2-15

```
...
generateKey();
if (initCipher()) {
    mCryptoObject = new FingerprintManager.CryptoObject(cipher);
    mFingerprintHelper = new FingerprintHelper(this);
}
```

รูปที่ 2-15 Code สำหรับการเรียกใช้งานจาก https://developers.ascendcorp.com/เรามาลองทำ-fingerprint-authentication-ด้วย-fingerprint-api-กัน-a2511dbd6c11

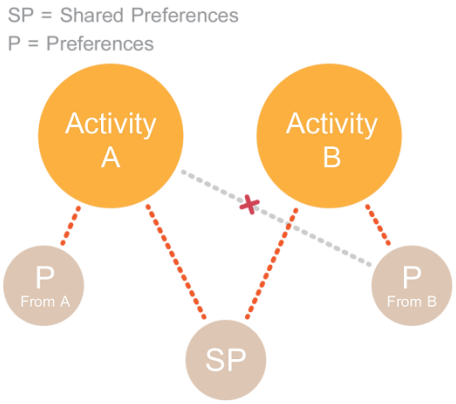
2.4 Shared Preferences

Shared Preferences เป็น Class ที่ใช้สำหรับเก็บข้อมูลชนิดต่าง ๆ โดยข้อมูลจะยังอยู่ในเครื่องแม้ว่าผู้ใช้จะปิดเครื่องไป เช่น Integer Boolean หรือ Float โดย Shared Preferences มีอยู่ 2 แบบคือ Shared Preferences และ Preferences

Shared Preference เป็นการเก็บข้อมูลตัวแปรที่สามารถดึงไปใช้งานที่ไหนก็ได้ภายใน App เช่น Activity A เก็บข้อมูลบางอย่าง แล้วให้ Activity B ดึงขึ้นมาใช้งาน เป็นต้น

Preferences เป็นการเก็บข้อมูลตัวแปรที่สามารถใช้งานได้เฉพาะที่ที่เรียกใช้เท่านั้น เช่น Activity A สร้าง Preferences เก็บค่าบางอย่างไว้ เมื่อ Activity B จะดึงค่าที่ Activity A เก็บไว้จะไม่สามารถทำได้

การเก็บข้อมูล และการดึงค่าไปใช้งานของ Shared Preferences ทั้ง 2 ประเภท สามารถอธิบายได้ ดังแสดงในรูปที่ 2-16



รูปที่ 2-16 การเก็บข้อมูล และการดึงค่าไปใช้งานของ Shared Preferences ทั้ง 2 ประเภท. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

โดยทั้งสอง Shared Preferences มีความแตกต่างกันเพียงคำสั่งที่เรียกใช้ ดังแสดงในรูปที่ 2-17 (สำหรับ Shared Preferences) และ 2-18 (สำหรับ Preferences)

```
SharedPreferences sp = getSharedPreferences(name, mode);
```

รูปที่ 2-17 คำสั่งที่เรียกใช้สำหรับ Shared Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

```
SharedPreferences sp = getPreferences(mode);
```

รูปที่ 2-18 คำสั่งที่เรียกใช้สำหรับ Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

การใช้งาน Shared Preferences แบ่งออกเป็น 2 ส่วน คือ การเก็บข้อมูลลงใน Shared Preferences และการอ่านข้อมูลจาก Shared Preferences ดังนี้

2.4.1 การเก็บข้อมูลลงใน Shared Preferences

การเก็บข้อมูลลงใน Shared Preferences จะต้องเรียกจาก Class SharedPreferences.Editor ดังแสดงในรูปที่ 2-19

```
SharedPreferences sp = getSharedPreferences("PREF_NAME", Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
```

รูปที่ 2-19 การเรียก Class SharedPreferences.Editor. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

PREF_NAME เป็นชื่อ Shared Preferences ซึ่งสามารถตั้งเป็นชื่ออะไรก็ได้ เพราะมันจะกลายเป็นชื่อไฟล์ที่เก็บลงในสมาร์ทโฟน ส่วน Context.MODE_PRIVATE คือการกำหนดรูปแบบของการสร้างไฟล์ ในที่นี้คือการสร้างไฟล์โดยให้ไฟล์นั้นสามารถเข้าถึงได้จาก App ที่มี UID เหมือนกันเท่านั้น การเก็บข้อมูลจากตัวแปร Integer ถูกแสดงในรูปที่ 2-20

```
int x = 100;

SharedPreferences sp = getSharedPreferences("PREF_NAME", Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.putInt("My_Value", x);
editor.commit();
```

รูปที่ 2-20 แสดงการเก็บข้อมูลจากตัวแปร Integer ลงใน Shared Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

การเก็บค่า Integer ลงใน Shared Preferences จะต้องกำหนด Keyword โดยในรูปที่ 2-20 นักพัฒนาเก็บข้อมูลลงใน Shared Preferences โดยใช้ Keyword My_Value ส่วนค่าที่เก็บคือค่าจากตัวแปร x ที่มีค่าเป็น 100

ต่อมาให้ใช้คำสั่ง Commit เพื่อบันทึกคู่ (Key, Value) (“My_Value”, 100) ไว้ใน Shared Preferences หากใช้คำสั่ง Put แล้วไม่ได้ใช้คำสั่ง Commit ต่อท้าย คู่ (Key, Value)จะไม่ถูกเก็บลงใน Shared Preferences

คำสั่งสำหรับ Put ข้อมูลเพื่อเตรียมเก็บลงใน Shared Preferences จะมีด้วยกันทั้งหมด ดังแสดงในรูปที่ 2-21

```
void putBoolean(String key, boolean value);
void putInt(String key, int value);
void putFloat(String key, float value);
void putLong(String key, long value);
void putString(String key, String value);
void putStringSet(String key, Set<String> value);
```

รูปที่ 2-21 คำสั่งทั้งหมดสำหรับ Put ข้อมูลเพื่อเตรียมเก็บลงใน Shared Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

2.4.2 การอ่านข้อมูลจาก Shared Preferences

การอ่านข้อมูลจาก Shared Preferences สามารถเรียกผ่าน Class Shared Preferences ได้ ดังแสดงในรูปที่ 2-22

```
SharedPreferences sp = getSharedPreferences("PREF_NAME", Context.MODE_PRIVATE);
int user_id = sp.getInt("userId", -1);
```

รูปที่ 2-22 การอ่านข้อมูลจาก Shared Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

จากรูปที่ 2-22 ค่า -1 มีไว้สำหรับกรณีไม่มี Keyword ดังกล่าวอยู่ใน Shared Preferences Method getInt จะ return ค่า -1 แทน ซึ่งผู้ใช้สามารถกำหนดค่าเองได้

คำสั่งสำหรับการอ่านข้อมูลจาก Shared Preferences จะมีด้วยกันทั้งหมด ดังแสดงในรูปที่ 2-23

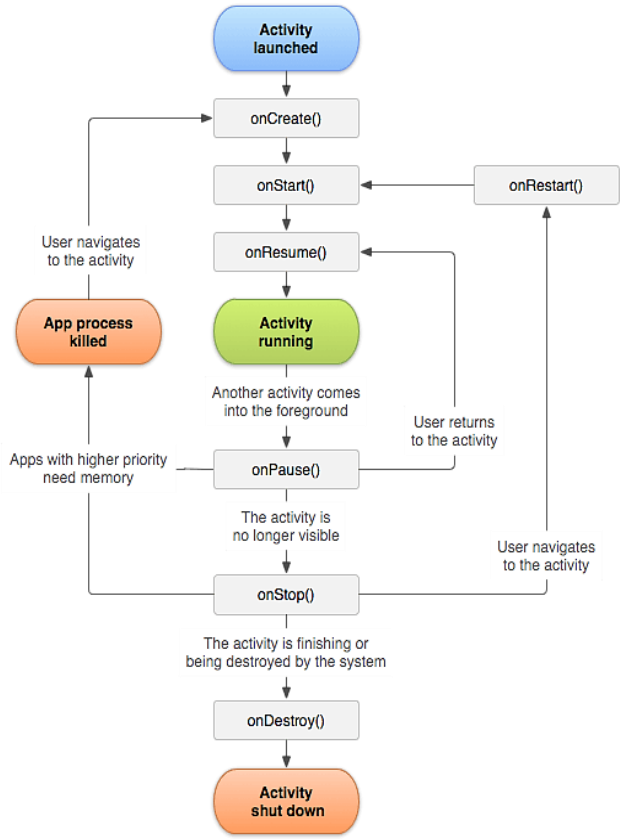
```
boolean getBoolean(String key, boolean defValue);
int getInt(String key, int defValue);
float getFloat(String key, float defValue);
long getLong(String key, long defValue);
String getString(String key, String defValue);
Set<String> getStringSet(String key, Set<String> defValue);
```

รูปที่ 2-23 คำสั่งทั้งหมดสำหรับการอ่านข้อมูลจาก Shared Preferences. จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html

สำหรับการอ่านข้อมูลจาก Shared Preferences จะมีลักษณะคล้ายคลึงกับการเก็บข้อมูลลงใน Shared Preferences คือ สามารถเรียกใช้ได้ทุกเมื่อแต่แตกต่างกันตรงที่ไม่ต้องมีการ Commit เพราะเป็นการอ่านข้อมูลออกมาใช้งานเท่านั้น

2.5 Activity Lifecycle

ภาพรวมการทำงาน Activity Lifecycle แสดงดังรูปที่ 2-24



รูปที่ 2-24 Activity Lifecycle Flowchart. จาก https://developer.android.com/reference/android/app/Activity.html

Activity Launched คือ สถานะเมื่อผู้ใช้เปิดเข้าใช้งาน App

Activity running คือ สถานะที่ผู้ใช้กำลังใช้งานอยู่บน App

Activity shut down คือ สถานะที่ Activity ถูกเลิกใช้งานแล้ว

App process killed คือ สถานะที่ App ถูก Kill โดยระบบ (เฉพาะกรณี Activity อยู่ในสถานะ Stop) เนื่องจากระบบต้องการ Memory หากผู้ใช้กลับเข้าสู่ App จะกลับไปเริ่มต้นทำงานใหม่ที่ Method onCreate()

onCreate() จะถูกแอนดรอยด์เรียกเมื่อ App เริ่มใช้งาน Activity นั้น แอนดรอยด์อาจ Create และ Destroy Activity อยู่เรื่อย ๆ ยกตัวอย่างเช่น เมื่อ User ทำการ Rotate Screen จะส่งผลให้ Activity ถูก Destroy และ Instance ใหม่ของ Activity เดิมก็จะถูก Create อีกครั้ง

onStart() จะถูกแอนดรอยด์เรียกหลังจาก onCreate() return หรือในกรณี Activity ของเราถูกสั่งให้ไปอยู่ใน Background (อาจโดยการสั่ง Launch Activity อื่นเหนือ Activity ของเรา) และถูก Stop onStart() จะถูกเรียกเมื่อเรากลับมาที่ Activity ของเราอีกครั้ง

onResume() จะถูกแอนดรอยด์เรียกหลังจาก onStart() return หรือเมื่อ Activity เปลี่ยนสถานะจาก Pause (ตอนมี Dialog มาบดบัง) กลับมาอยู่บน Foreground อีกครั้งโดย onResume() จะถูกเรียกก่อนที่แอนดรอยด์จะแสดง Layout ของ Activity บนหน้าจอ

onPause() จะถูกแอนดรอยด์เรียกเมื่อ Activity กำลังเปลี่ยนจากสถานะ Foreground ไปยัง Background หรือ Activity ถูกบดบังบางส่วน อาจเกิดจากการสั่ง Launch Activity อื่นขึ้นมาอยู่เหนือ Activity ที่กำลังแสดงผลอยู่ (แต่ยังสามารถมองเห็น Activity ที่ถูกบดบังได้)

onStop() จะถูกแอนดรอยด์เรียกหลังจาก onPause() หรือเมื่อ Activity ของเรากลายเป็น Background

onDestroy() จะถูกแอนดรอยด์เรียกเมื่อผู้ใช้ออกจาก App หรือก่อนที่ Activity จะปิดตัวลงอย่างถาวร เป็น Method ที่เราใช้ในการคืนค่า resources ภายใน Method นี้ เราควรจะสั่งปิด Process ใด ๆ ที่เราสั่ง Run ไว้ใน Background

onRestart() จะถูกแอนดรอยด์เรียกเมื่อ Activity อยู่ในสภาวะ onStop() อยู่ แล้วเมื่อเรากลับมาที่ Activity อีกครั้ง onRestart() จะถูกเรียก และแอนดรอยด์จะเรียก onStart() ต่อจาก onRestart() เสมอ ๆ

บทที่ 3

อภิปรายผลการทดลอง

บทนี้อธิบายถึงวิธีการดำเนินงาน และอภิปรายผลการทดลอง โดยแบ่งออกเป็น 2 ส่วน ได้แก่ ส่วนการทดสอบ App Signal โดยใช้ ARO และส่วนการยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal)

การทดสอบ App Signal โดยใช้ ARO ประกอบด้วยการศึกษา และวิเคราะห์ตัว App เริ่มจากติดตั้ง Signal ลงในสมาร์ทโฟน และกำหนดสภาพแวดล้อมการทดสอบให้อยู่ภายใต้การควบคุมของผู้วิจัย โดยการทดสอบ App Signal โดยใช้ ARO นั้นจะต้องมีระบบ Server ที่เป็นของผู้วิจัยเอง

ส่วนการพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) นั้นผู้วิจัยได้ใช้ระบบ Server สาธารณะของผู้พัฒนา Signal เลย โดยผู้วิจัยเริ่มจากการออกแบบ Layout ของส่วนการยืนยันตัวตน, การออกแบบ Wireframe ของส่วนการยืนยันตัวตนและส่วนการตั้งค่า และการออกแบบแอททิวิตี้ไดอะแกรมของส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal)

การทดสอบโดยใช้ ARO มีวิธีการดำเนินงานได้ดังนี้

3.1 การติดตั้ง Server และตั้งค่า Configuration

3.2 ตั้งค่าแอปพลิเคชัน Signal เชื่อมต่อกับ Server ของผู้วิจัย

3.3 การทดสอบแอปพลิเคชันตามหัวข้อทดสอบที่ได้กำหนดไว้

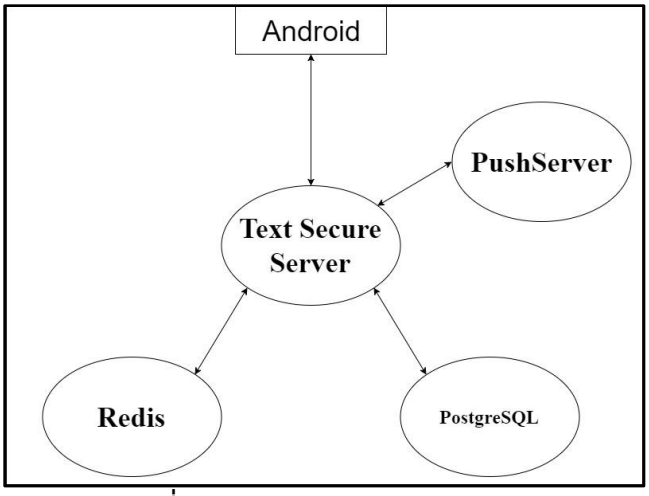
การพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) มีวิธีการดำเนินงานได้ดังนี้

3.4 การออกแบบ Layout ของส่วนการยืนยันตัวตน

3.5 การออกแบบ Wireframe ของส่วนการยืนยันตัวตน และส่วนการตั้งค่า

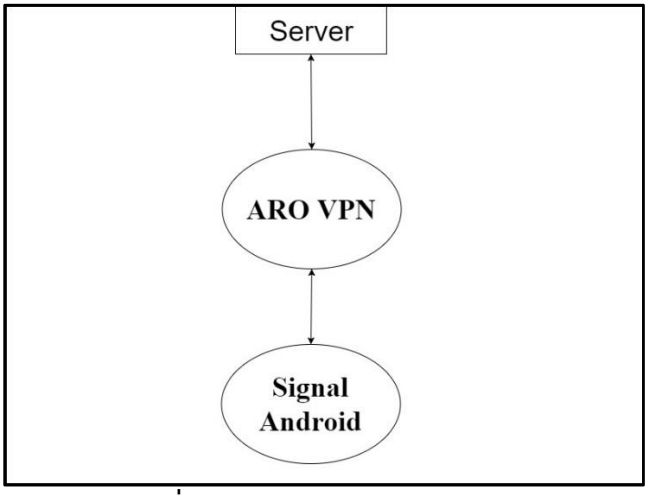
3.6 การออกแบบแอททิวิตี้ไดอะแกรมของส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal)

ส่วนประกอบโดยรวมทั้งหมดของระบบ Text Secure Server (Signal Server) ประกอบไปด้วย Redis, PostgreSQL, PushServer และ Android (App Signal) ดังแสดงในรูปที่ 3-1



รูปที่ 3-1 Signal Server Diagram

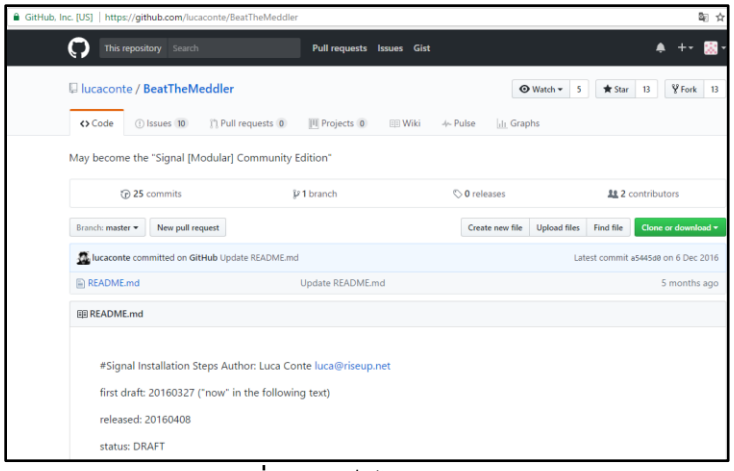
โดย ARO VPN จะดักจับข้อมูลระหว่าง Signal Android กับ Server เพื่อใช้ในการทดสอบโดย ARO ดังแสดงในรูปที่ 3-2



รูปที่ 3-2 ARO Testing Diagram

3.1 การติดตั้ง Server และตั้งค่า Configuration

3.1.1 ไปยัง https://github.com/lucaconte/BeatTheMeddler เพื่อศึกษางานที่มีผู้พัฒนาได้พัฒนาเบื้องต้นไว้บางส่วนแล้ว



รูปที่ 3-3 เว็บไซต์ GitHub

3.1.2 Download หรือ Clone (หากมี Git) Signal Server ซึ่งประกอบด้วย TextSecure-Server และ PushServer จากลิงค์ต่อไปนี้

https://github.com/WhisperSystems/TextSecure-Server และ

https://github.com/WhisperSystems/PushServer

3.1.3 สิ่งที่ Server ที่จะทำการติดตั้ง Signal Server จะต้องมีโปรแกรมต่าง ๆ ดังนี้ติดตั้งไว้ด้วย

- JAVA 1.7 ที่เป็นของ OpenJdk
- Redis
- PostgreSQL (สร้าง DB accountsdb และ messaged ด้วยคำสั่ง

  #sudo -u postgres createdb -O signal accountsdb

  #sudo -u postgres createdb -O signal messagedb)
- Twillio account สำหรับการยืนยันตัวตนเมื่อสมัครการใช้Signal โดยผ่าน SMS (ผู้วิจัยไม่ได้ติดตั้งเนื่องจากไม่ต้องการใช้บริการ)
- Amazon AWS S3 bucket service สำหรับการรับส่งไฟล์ (ผู้วิจัยไม่ได้ติดตั้งเนื่องจากไม่ต้องการใช้บริการ)
- GCM account จาก google developer console เพื่อการติดตามสำหรับแอปพลิเคชันที่นำไปใช้จริง (ผู้วิจัยไม่ได้ติดตั้งเนื่องจากไม่ต้องการใช้บริการ)
- Apple Push Notifications (APN) สำหรับ IOS (ไม่ติดตั้งเนื่องจากทดสอบเฉพาะ IOS)

3.1.4 ทำการเปลี่ยน APN Service ตาม Path เริ่มจาก PushServer ที่ได้ Download หรือ Clone มา ./org/whispersystems/pushserver/senders/APNSender.class ใน method “public void start ()”ดังแสดงในรูปที่ 3-2 เปลี่ยนจาก withProdouctionDestinetion() เป็น withSandboxDestination

```
//[LC] uncomment for production releases (with the right certificate)
// this.pushApnService = APNS.newService()
//                          .withCert(new ByteArrayInputStream(pushKeyStore), "insecure")
//                         .asQueued()
//                        .withProductionDestination().build();

this.pushApnService = APNS.newService()
        .withCert(new ByteArrayInputStream(pushKeyStore), "insecure")
        .asQueued()
        .withSandboxDestination().build();
```

รูปที่ 3-4 แก้ไข code ใน public void start()

3.1.5 ทำการ compile ด้วย Command Line หลังจากแก้ไข ด้วยคำสั่ง

#apt-get install -y maven

#cd PushServer

#mvn clean install

3.1.6 จะพบ Error ในลักษณะตามนี้

[ERROR] Failed to execute goal on project TextSecureServer: Could not resolve dependencies for project org.whispersystems.textsecure:TextSecureServer:jar:0.92: Failed to collect dependencies at org.whispersystems:websocket-resources:jar:0.3.2: Failed to read artifact descriptor for org.whispersystems:websocket-resources:jar:0.3.2: Could not find artifact org.whispersystems:parent:pom:0.3.2 in gcm-server-repository (https://raw.github.com/whispersystems/maven/master/gcm-server/releases/) -> [Help 1]

3.1.7 ต้องทำการ Bypass Error นี้ด้วยการ Clone WebSocket-Resource จากลิงค์
https://github.com/lucaconte/WebSocket-Resources.git
3.1.8 ทำการ compile ด้วย Command Line ดังนี้

```
#cd ../WebSocket-Resources
#mvn clean install
#mvn install:install-file -Dfile=./library/target/websocket-resources-0.3.2.jar -
DgroupId=org.whispersystems -DartifactId=websocket-resources -
Dversion=0.3.2 -Dpackaging=jar
```

3.1.9 ทำการ Bypass SMS service โดยแก้ไข 3 ไฟล์
ไฟล์ที่1
src/main/java/org/whispersystems/textsecuregcm/sms/TwiliOSmsSender.java ลบ code ที่อยู่ภายใน Method `deliverVoxVerification` `deliverSmsVerification` และให้ `getRandom` return ค่าคงที่ (100000)
ไฟล์ที่ 2
src/main/java/org/whispersystems/textsecuregcm/sms/SmsSender.java
ลบ code ที่อยู่ภายใน Method `deliverSmsVerification`
ไฟล์ที่ 3
src/main/java/org/whispersystems/textsecuregcm/controllers/
AccountController.java ใน Method createAccount() ภายใน if else ให้ลบ code ที่อยู่ภายในออก จะได้ดังนี้

```
if (testDevices.containsKey(number)) {
// noop
} else if (transport.equals("sms")) {
} else if (transport.equals("voice")) {}
```

ใน Method generateVerificationCode (Constant) แก้ไขค่า randomInt เป็น 100000 ได้ดังนี้ int randomInt = 100000;

3.1.10 จากนั้น compile TextSecureServer ด้วย คำสั่งดังนี้
#cd ..
#cd TextSecure-Server
#mvn install
หากพบปัญหาที่เกี่ยวกับ Test จะข้ามการ Test ไปด้วยการเพิ่ม -DskipTests จาก mvn install
3.1.11 สร้าง file config ตามนี้ config/textsesure.yml ภายในไฟล์ดังนี้

```
twilio:
  accountId: AC302d9ea2695e21cd17ce15bc510d28fd #fake
  accountToken: febf5ccba3b4051dd7e7d0901a0fd404 #fake
  numbers:
  -
     +66876157370 #fake
  localDomain: domain
push:
  host: localhost
  port: 9090
  username: whisper
  password: thepassword
s3:
   accessKey: ABCDEFGCUFYDHVM2LXXX #fake
   accessSecret: W0UfGDddfAbqYyCTIIbSQlDtreTGokOs0OTpL0SE #fake
   attachmentsBucket: thenameofyouts3buket #fake
directory:
  url: "redis://localhost:6379/0"
cache:
  url: "redis://localhost:6379/1"
server:
  applicationConnectors:
    - type: https
      port: 8080
      keyStorePath: ./ssl/example.keystore
```

```
    keyStorePassword: example
#keyStoreType : PKCS12
    validateCerts: false
adminConnectors:
- type: https
    port: 8081
    keyStorePath: ./ssl/example.keystore
    keyStorePassword: example
    #keyStoreType : PKCS12
    validateCerts: false
websocket:
  enabled: true
messageStore: # Postgres database configuration for message store
  driverClass: org.postgresql.Driver
  user: "whisper"
  password: "thepassword"
  url: "jdbc:postgresql://localhost:5432/messagedb"
database:
  driverClass: org.postgresql.Driver
  user: "whisper"
  password: "thepassword"
  url: "jdbc:postgresql://localhost:5432/accountdb"
  properties:
    charSet: UTF-8
#federation: # is disabled
logging:
  level: INFO
```

```
 appenders:
    - type: file
      currentLogFilename: /tmp/textsecureshserver.log
      archivedLogFilenamePattern: /temp/textsecureserver-%d.log.gz
      archivedFileCount: 5
    - type: console
redphone:
  authKey: 1234567890 #fake
```

3.1.12 สร้าง file config ตามนี้ config/pushserver.yml ภายในไฟล์ดังนี้

```
redis:
  url: redis://localhost:6379/2
authentication:
  servers:
    -
      name: whisper
      password: thepassword
gcm:
  xmpp: false
  apiKey: AIzaSyC8gPzceq2SPebZZWaD3I9OeqePyD9CUqk #real
  senderId: 412918270132 #real
  redphoneApiKey: AIddSyAsviyMy8jKe8chCEfr8NbeqGghy7oOCi4 #fake
server:
    applicationConnectors:
    - type: http
      port: 9090
      #keyStorePath: ./ssl/example.keystore
      #keyStorePassword: example
```

```
      #keyStoreType : PKCS12
      #validateCerts: false
    adminConnectors:
    - type: http
      port: 9091
      #keyStorePath: ./ssl/example.keystore
      #keyStorePassword: example
      #keyStoreType : PKCS12
      #validateCerts: false
    gzip:
        enabled: true
logging:
  level: INFO
  appenders:
    - type: file
      currentLogFilename: /tmp/pushserver.log
      archivedLogFilenamePattern: /tmp/pushserver-%d.log.gz
      archivedFileCount: 5
    - type: console
```

3.1.13 จากนั้นทำการสร้าง PostgreSQL Database ไว้ คำสั่งนี้จะเป็นการสั่งสำหรับติดตั้ง

#apt-get update

#apt-get install -y redis-server postgresql openjdk-7jre-headless

#/etc/init.d/postgresql start #sudo -u postgres psql --command

"CREATE USER signal WITH SUPERUSER PASSWORD 'thepassword';"

#sudo -u postgres createdb -O signal accountsdb

#sudo -u postgres createdb -O signal messagedb

3.1.14. ทำการ Migrate config ของ TextSecure เข้ากับ Database ตามคำสั่งดังนี้

#java -jar TextSecure-Server/target/ TextSecureServer-0.93_sin.jar accountsdb migrate config/textsecure.yml

#java -jar TextSecure-Server/target/ TextSecureServer-0.93_sin.jar messagedb migrate

3.1.15. การทำ SSL ไปยัง Server เพื่อแก้ไขให้ Signal รองรับ HTTPS ได้ ด้วยคำสั่งต่อไปนี้ (Code ด้านล่างเป็น Bash Command สามารถ copy ไปใส่ไฟล์ .sh แล้วสั่ง Run Script ได้เลย)

```
#/bin/bash
#This script creates root CA and server certificates to be used by the client and the server.
# rootCA.crt needs to be copied to the client to replace the system-wide root CA set
# example.keystore needs to be referenced by keyStorePath in the server's config file
#
#TO EXECUTE FROM THE OUTSIDE WITH:
#ALTNAME=DNS:signal.foo.org ./gencerts
#
#IF YOU HAVE A DOMAIN OR IP if U are in a lan
#ALTNAME=IP:<IP ของเครื่อง Server > ./gencerts
#
#
# Create private key for root CA certificate
openssl genrsa -out rootCA.key 4096

# Create a self-signed root CA certificate
openssl req -x509 -new -nodes  -days 3650 -out rootCA.crt -key rootCA.key
```

```
# Create server certificate key
openssl genrsa -out whisper.key 4096

# Create Certificate Signing Request
openssl req -new -key whisper.key -out whisper.csr

# Sign the certificate with the root CA

openssl x509 -req -in whisper.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial
-days 365 -out whisper.crt -extensions extensions -extfile <(cat <<-EOF
[ extensions ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
subjectAltName=IP:<IP ของเครื่อง Server > ./gencerts
EOF
)
# Export to host key and certificate to PKCS12 format which is recognized by Java
keytool
openssl pkcs12 -export -password pass:example -in whisper.crt -inkey whisper.key
-out keystore.p12 -name example -CAfile rootCA.crt

# Import the host key and certificate to Java keystore format, so it can be used
by dropwizard
keytool -importkeystore -srcstoretype PKCS12 -srckeystore keystore.p12 -
srcstorepass example -destkeystore example.keystore -deststorepass example

#whisper.store bust be placed into  Android client
```

```
# -- keytool -importcert -v -trustcacerts -file whisper.crt -alias IntermediateCA -
keystore whisper.store -provider
org.bouncycastle.jce.provider.BouncyCastleProvider -providerpath ../bcprov-
jdk15on-154.jar -storetype BKS -storepass whisper


#for iOS client you have to convert  whisper.crt in  DER format and install
whisper.cer in Signal-iOS
#openssl x509 -in whisper.crt -out whisper.cer -outform DER
```

3.1.16 ทำการสั่ง Run Server ตามคำสั่งต่อไปนี้

#/etc/init.d/postgresql start

#rm -rf /var/run/redis_6379.pid

#service redis_6379 start

#java -jar PushServer/target/Push-Server-0.12.0capsule-fat.jar server pushserver.yml &

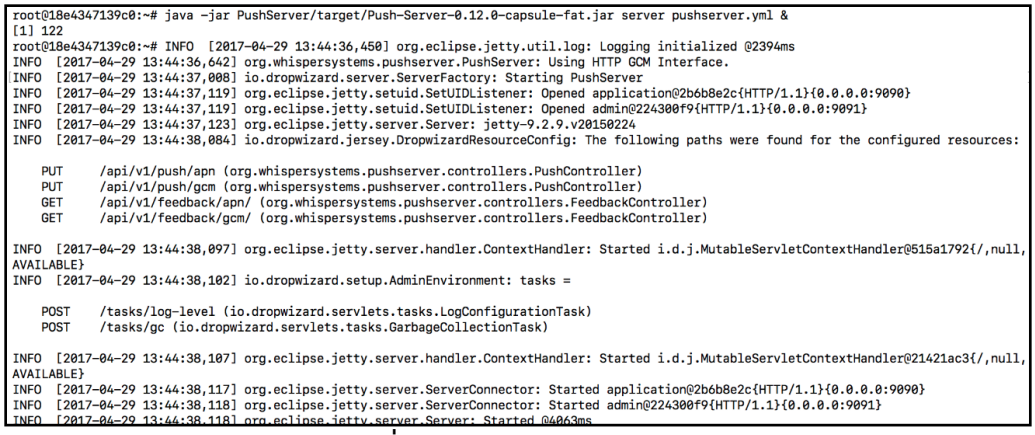
#java -jar TextSecure-Server/target/ TextSecureServer-0.93_sin.jar server textsecure.yml

```
root@18e4347139c0:~# /etc/init.d/postgresql start
 * Starting PostgreSQL 9.3 database server                              [ OK ]
root@18e4347139c0:~#
```

รูปที่ 3-5 Start PostgreSQL

```
root@18e4347139c0:~# rm -rf /var/run/redis_6379.pid
root@18e4347139c0:~# service redis_6379 start
Starting Redis server...
root@18e4347139c0:~#
```

รูปที่ 3-6 Start Redis

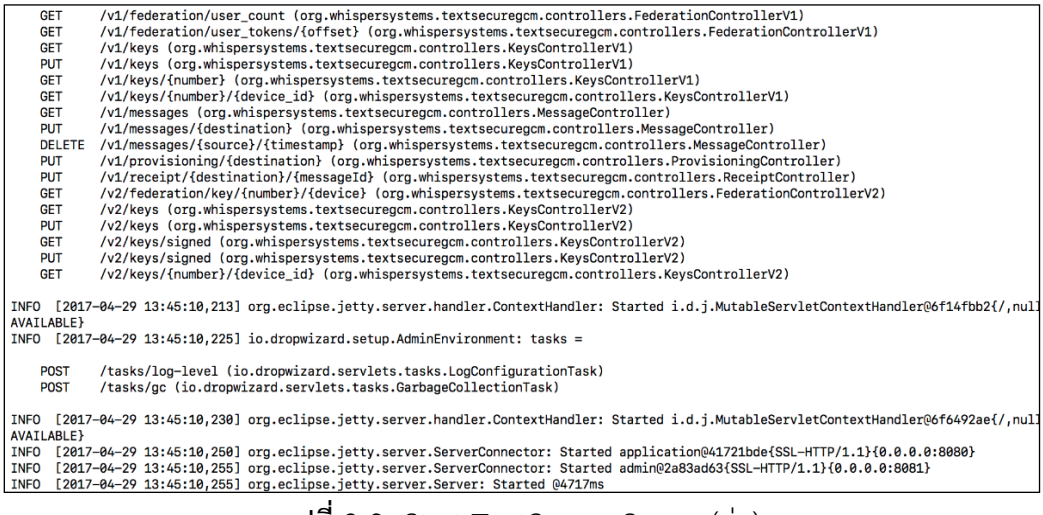


```
root@18e4347139c0:~# java -jar PushServer/target/Push-Server-0.12.0-capsule-fat.jar server pushserver.yml &
[1] 122
root@18e4347139c0:~# INFO  [2017-04-29 13:44:36,450] org.eclipse.jetty.util.log: Logging initialized @2394ms
INFO  [2017-04-29 13:44:36,642] org.whispersystems.pushserver.PushServer: Using HTTP GCM Interface.
INFO  [2017-04-29 13:44:37,008] io.dropwizard.server.ServerFactory: Starting PushServer
INFO  [2017-04-29 13:44:37,119] org.eclipse.jetty.setuid.SetUIDListener: Opened application@2b6b8e2c{HTTP/1.1}{0.0.0.0:9090}
INFO  [2017-04-29 13:44:37,119] org.eclipse.jetty.setuid.SetUIDListener: Opened admin@224300f9{HTTP/1.1}{0.0.0.0:9091}
INFO  [2017-04-29 13:44:37,123] org.eclipse.jetty.server.Server: jetty-9.2.9.v20150224
INFO  [2017-04-29 13:44:38,084] io.dropwizard.jersey.DropwizardResourceConfig: The following paths were found for the configured resources:

    PUT     /api/v1/push/apn (org.whispersystems.pushserver.controllers.PushController)
    PUT     /api/v1/push/gcm (org.whispersystems.pushserver.controllers.PushController)
    GET     /api/v1/feedback/apn/ (org.whispersystems.pushserver.controllers.FeedbackController)
    GET     /api/v1/feedback/gcm/ (org.whispersystems.pushserver.controllers.FeedbackController)

INFO  [2017-04-29 13:44:38,097] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHandler@515a1792{/,null,
AVAILABLE}
INFO  [2017-04-29 13:44:38,102] io.dropwizard.setup.AdminEnvironment: tasks =

    POST    /tasks/log-level (io.dropwizard.servlets.tasks.LogConfigurationTask)
    POST    /tasks/gc (io.dropwizard.servlets.tasks.GarbageCollectionTask)

INFO  [2017-04-29 13:44:38,107] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHandler@21421ac3{/,null,
AVAILABLE}
INFO  [2017-04-29 13:44:38,117] org.eclipse.jetty.server.ServerConnector: Started application@2b6b8e2c{HTTP/1.1}{0.0.0.0:9090}
INFO  [2017-04-29 13:44:38,118] org.eclipse.jetty.server.ServerConnector: Started admin@224300f9{HTTP/1.1}{0.0.0.0:9091}
INFO  [2017-04-29 13:44:38,118] org.eclipse.jetty.server.Server: Started @4063ms
```

รูปที่ 3-7 Start Push-Server

```
root@18e4347139c0:~# java -jar TextSecure-Server/target/TextSecureServer-0.93_sin.jar server textsecure.yml
INFO  [2017-04-29 13:45:07,498] org.eclipse.jetty.util.log: Logging initialized @1958ms
INFO  [2017-04-29 13:45:08,917] io.dropwizard.jersey.DropwizardResourceConfig: The following paths were found for the configured resources:

    GET     /v1/keepalive (org.whispersystems.textsecuregcm.controllers.KeepAliveController)
    GET     /v1/keepalive/provisioning (org.whispersystems.textsecuregcm.controllers.KeepAliveController)

INFO  [2017-04-29 13:45:09,032] io.dropwizard.jersey.DropwizardResourceConfig: The following paths were found for the configured resources:

    GET     /v1/keepalive (org.whispersystems.textsecuregcm.controllers.KeepAliveController)
    GET     /v1/keepalive/provisioning (org.whispersystems.textsecuregcm.controllers.KeepAliveController)

INFO  [2017-04-29 13:45:09,057] io.dropwizard.server.ServerFactory: Starting whisper-server
 _    _ _     _                  _____
| |  | | |   (_)                / ___|
| |  | | |__  _ ___ _ __   ___ _ __ \ `--.  ___ _ ____   _____ _ __
| |/\| | '_ \| / __| '_ \ / _ \ '__|`--. \/ _ \ '__\ \ / / _ \ '__|
\  /\  / | | | \__ \ |_) |  __/ |  /\__/ /  __/ |   \ V /  __/ |
 \/  \/|_| |_|_|___/ .__/ \___|_|  \____/ \___|_|    \_/ \___|_|
                   | |
                   |_|

INFO  [2017-04-29 13:45:09,069] io.dropwizard.server.DefaultServerFactory: Registering jersey handler with root path prefix: /
INFO  [2017-04-29 13:45:09,080] io.dropwizard.server.DefaultServerFactory: Registering admin handler with root path prefix: /
```

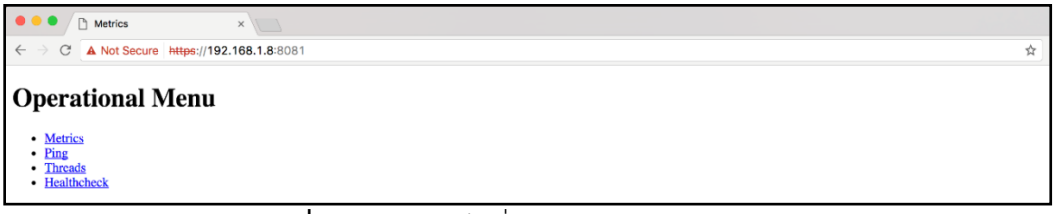
รูปที่ 3-8 Start TextSesure-Server



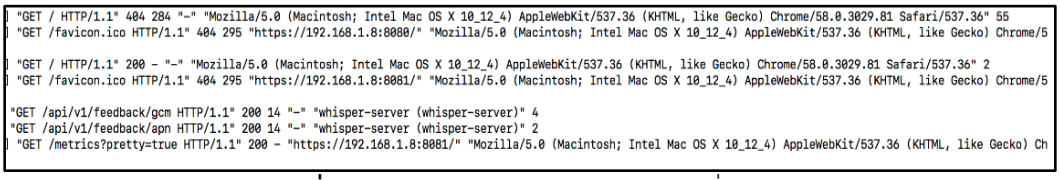
```
    GET     /v1/federation/user_count (org.whispersystems.textsecuregcm.controllers.FederationControllerV1)
    GET     /v1/federation/user_tokens/{offset} (org.whispersystems.textsecuregcm.controllers.FederationControllerV1)
    GET     /v1/keys (org.whispersystems.textsecuregcm.controllers.KeysControllerV1)
    PUT     /v1/keys (org.whispersystems.textsecuregcm.controllers.KeysControllerV1)
    GET     /v1/keys/{number} (org.whispersystems.textsecuregcm.controllers.KeysControllerV1)
    GET     /v1/keys/{number}/{device_id} (org.whispersystems.textsecuregcm.controllers.KeysControllerV1)
    GET     /v1/messages (org.whispersystems.textsecuregcm.controllers.MessageController)
    PUT     /v1/messages/{destination} (org.whispersystems.textsecuregcm.controllers.MessageController)
    DELETE  /v1/messages/{source}/{timestamp} (org.whispersystems.textsecuregcm.controllers.MessageController)
    PUT     /v1/provisioning/{destination} (org.whispersystems.textsecuregcm.controllers.ProvisioningController)
    PUT     /v1/receipt/{destination}/{messageId} (org.whispersystems.textsecuregcm.controllers.ReceiptController)
    GET     /v2/federation/key/{number}/{device} (org.whispersystems.textsecuregcm.controllers.FederationControllerV2)
    GET     /v2/keys (org.whispersystems.textsecuregcm.controllers.KeysControllerV2)
    PUT     /v2/keys (org.whispersystems.textsecuregcm.controllers.KeysControllerV2)
    GET     /v2/keys/signed (org.whispersystems.textsecuregcm.controllers.KeysControllerV2)
    PUT     /v2/keys/signed (org.whispersystems.textsecuregcm.controllers.KeysControllerV2)
    GET     /v2/keys/{number}/{device_id} (org.whispersystems.textsecuregcm.controllers.KeysControllerV2)

INFO  [2017-04-29 13:45:10,213] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHandler@6f14fbb2{/,null
AVAILABLE}
INFO  [2017-04-29 13:45:10,225] io.dropwizard.setup.AdminEnvironment: tasks =

    POST    /tasks/log-level (io.dropwizard.servlets.tasks.LogConfigurationTask)
    POST    /tasks/gc (io.dropwizard.servlets.tasks.GarbageCollectionTask)

INFO  [2017-04-29 13:45:10,230] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHandler@6f6492ae{/,null
AVAILABLE}
INFO  [2017-04-29 13:45:10,250] org.eclipse.jetty.server.ServerConnector: Started application@41721bde{SSL-HTTP/1.1}{0.0.0.0:8080}
INFO  [2017-04-29 13:45:10,255] org.eclipse.jetty.server.ServerConnector: Started admin@2a83ad63{SSL-HTTP/1.1}{0.0.0.0:8081}
INFO  [2017-04-29 13:45:10,255] org.eclipse.jetty.server.Server: Started @4717ms
```

รูปที่ 3-9 Start TextSesure-Server (ต่อ)

3.1.17 หลังจาก Server Start เสร็จแล้ว ทดสอบอย่างง่ายด้วยการเข้าผ่าน Browser https://ip:8081 เช่น https://192.168.1.8:8081



รูปที่ 3-10 หน้าเว็บที่แสดงว่า Server ใช้งานได้

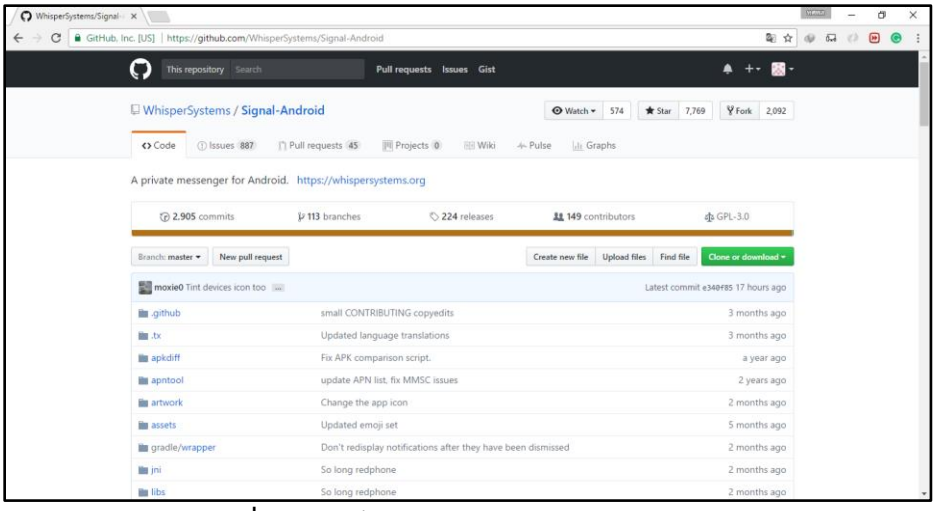


รูปที่ 3-11 Log จาก Server ว่ามีการเชื่อมต่อ

3.1.18 เมื่อทำงานได้ตามข้างต้น แสดงว่า Server พร้อมใช้งานแล้ว

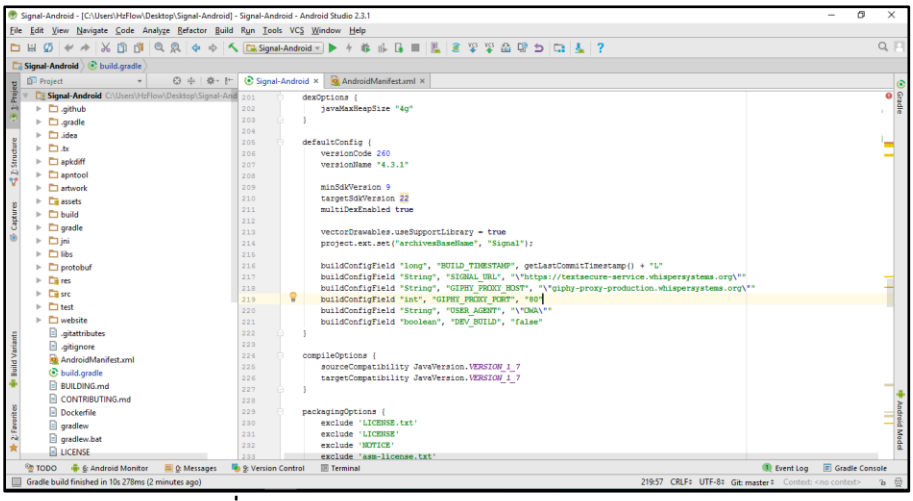
3.2 ตั้งค่าแอปพลิเคชันเชื่อมต่อกับ Server ที่ได้ถูกติดตั้งตามข้อ 3.1

3.2.1 ดาวน์โหลด Source file ของ Signal Android จาก ลิงค์https://github.com/WhisperSystems/Signal-Android



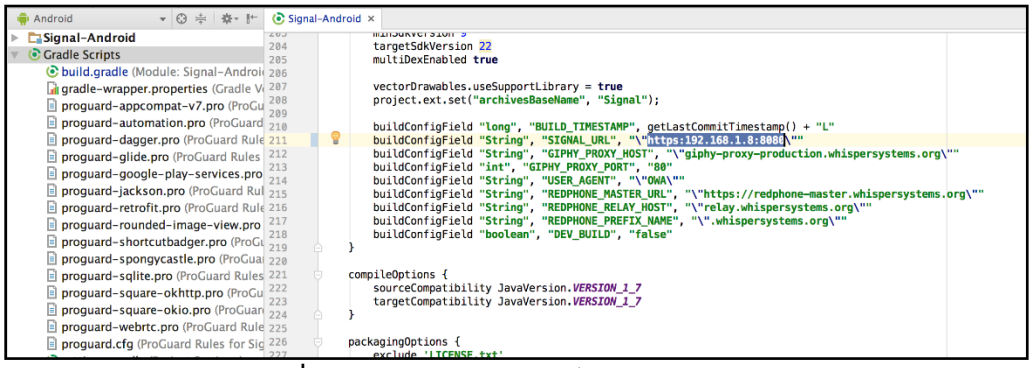
รูปที่ 3-12 เว็บ Source ของ Signal Android

3.2.2 เปิด Source ที่ Download หรือ Clone มาด้วย Android Studio



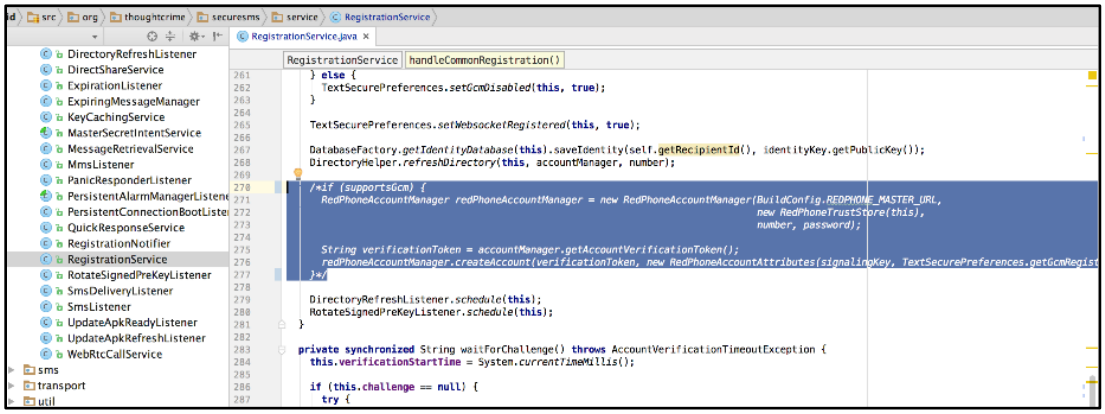
รูปที่ 3-13 เปิด Source ด้วย Android Studio

3.2.3 ทำการแก้ไข buildConfigField "String", "SIGNAL_URL", "\"https://textsecure-service.whispersystems.org\"" เป็น IP ของ Server ที่เราตั้งขึ้น



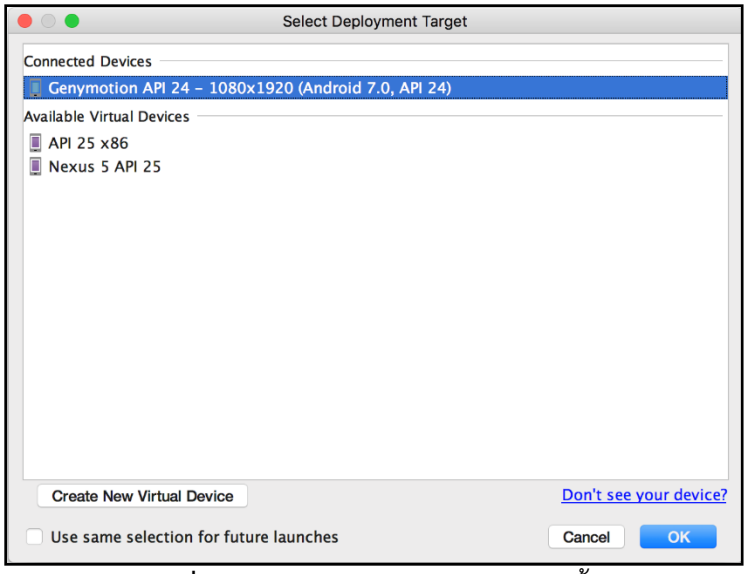
รูปที่ 3-14 SIGNAL_URL เป็น IP ของ Server

3.2.4 ไปยังไฟล์ RegistrationService.java ใน Folder src>org>thougtcrime>securesms>service>RegistrationService ปิด GCM โดยการ comment ดังแสดงในรูปที่ 3-15

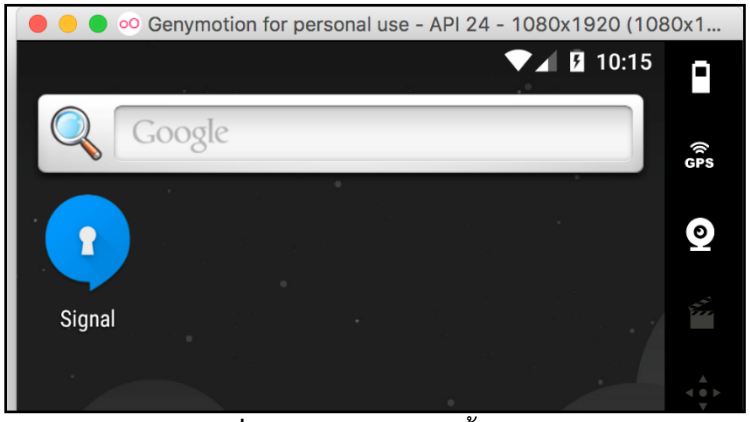


รูปที่ 3-15 ทำการ comment code เพื่อปิด GCM

3.2.5 Run Source ลง Emu หรือ สร้างเป็นไฟล์ .APK เพื่อติดตั้งบนสมาร์ทโฟนที่เป็นระบบปฏิบัติการแอนดรอยด์ 2 เครื่อง

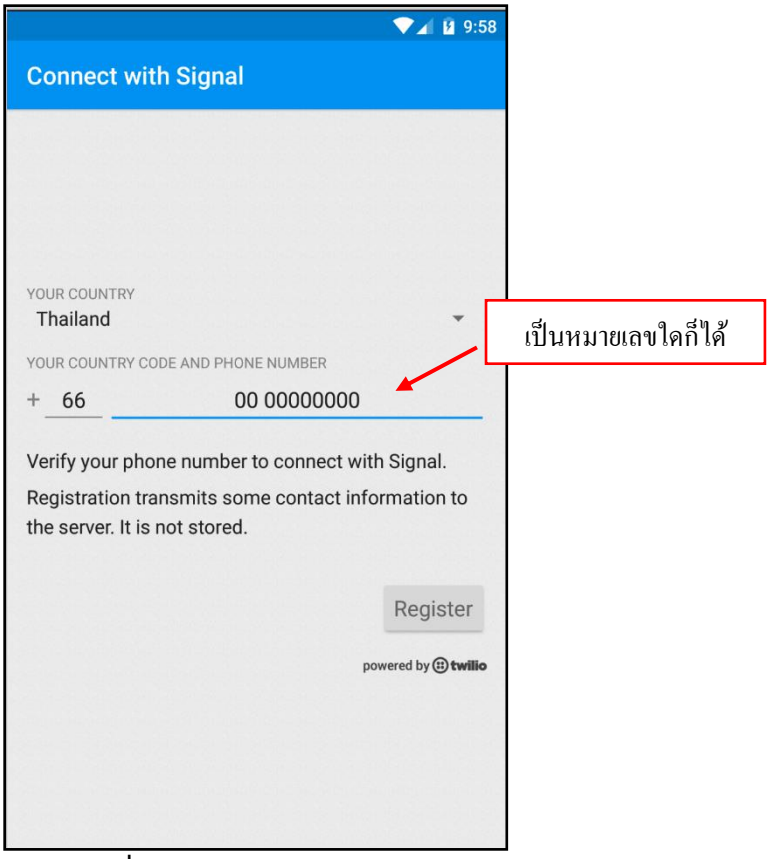


รูปที่ 3-16 เลือกอุปกรณ์ที่ต้องการติดตั้ง

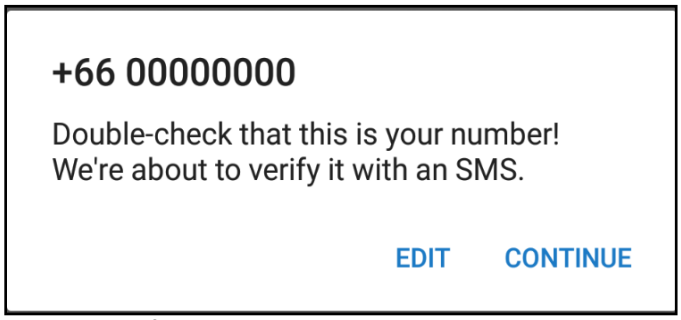


รูปที่ 3-17 หลังจากติดตั้งสำเร็จ

3.2.6 จากนั้นก็ทำการกรอกข้อมูลหมายเลขโทรศัพท์ลงไป

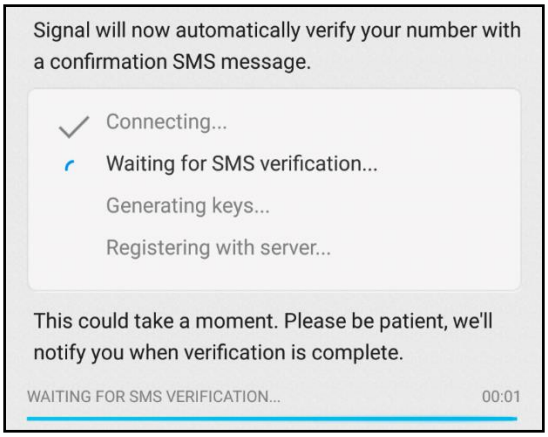


รูปที่ 3-18 กรอกหมายเลขผู้ใช้



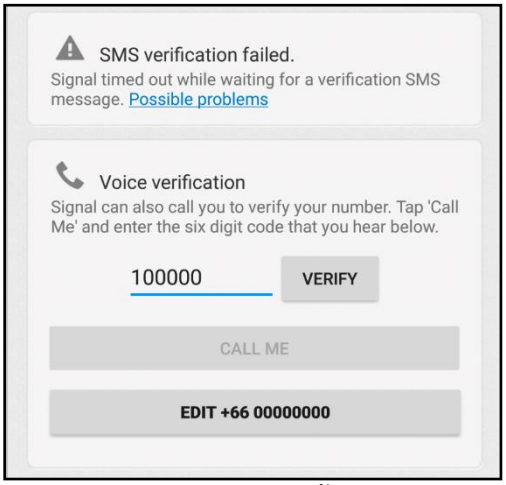
รูปที่ 3-19 กล่องข้อความยืนยันหมายเลข

3.2.7 จากนั้นระบบจะทำการส่ง SMS มายังหมายเลขที่กรอกไว้ แต่ในกรณีทดสอบ เราได้ทำการ Bypass การส่ง SMS เนื่องจากมีค่าใช้จ่าย จึงรอให้หมดเวลา (วิธี Bypass จากหัวข้อ 3.2 ข้อ 9)



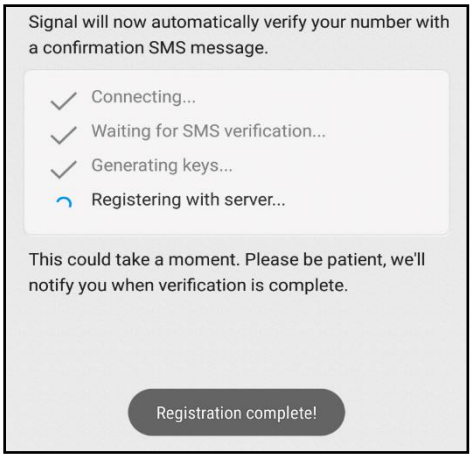
รูปที่ 3-20 รอคอย SMS เพื่อยืนยันหมายเลข

3.2.8 หลังจากหมดเวลา ระบบจะเปลี่ยนไปอีกหน้า เราสามารถแก้ไขหมายเลขเพื่อไปขั้นตอนที่ 6 อีกครั้ง หรือจริง ๆ แล้ว Signal ยอมให้ผู้ใช้กด “CALL ME” เพื่อฟังรหัสผ่านได้ โดย Signal จะโทรมาที่โทรศัพท์ตามเบอร์โทรศัพท์ที่ให้ไว้ในขั้นตอนที่ 7 แต่ในงานวิจัยนี้ ผู้วิจัยได้ละขั้นตอนนี้ไว้ จากนั้นผู้ใช้กรอกรหัสที่ได้ตั้งไว้ตอนแก้ไข Server คือ 100000

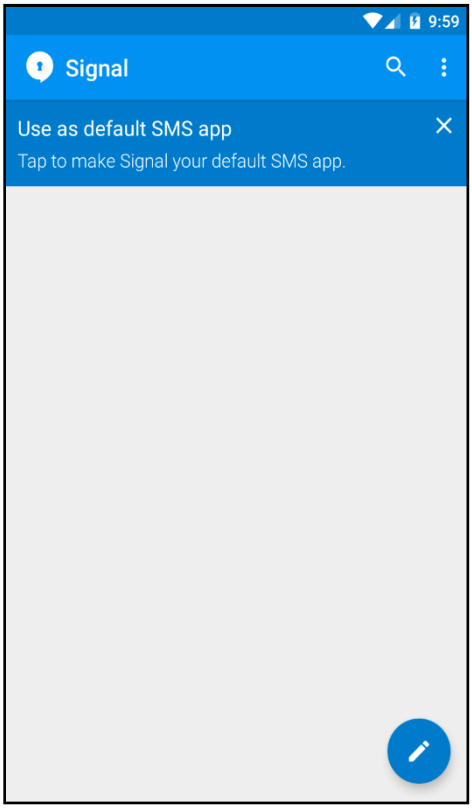


รูปที่ 3-21 กรอกรหัสที่ได้ตั้งไว้กับ Server

3.2.9 จากนั้นทำการรอ และเข้าสู่หน้าที่สามารถใช้งานได้

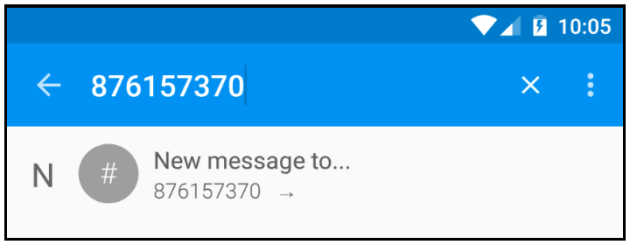


รูปที่ 3-22 ทำการสมัครเรียบร้อย

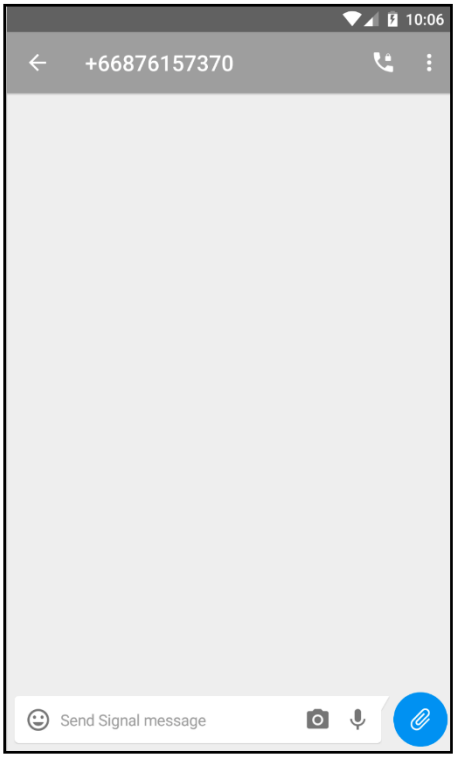


รูปที่ 3-23 หน้าจอพร้อมใช้งาน

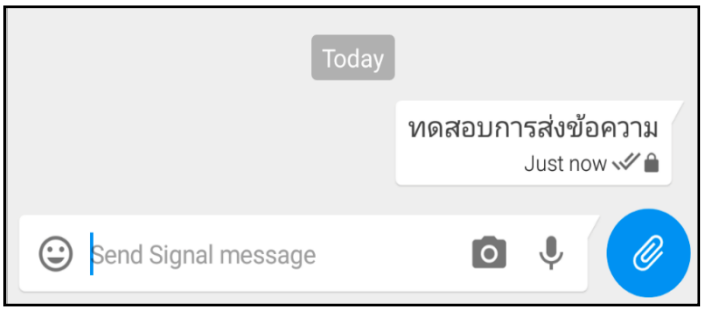
3.2.10 ทดสอบการใช้งานโดยการส่งข้อความไปหาอีกเครื่อง



รูปที่ 3-24 ค้นหาคู่แชทด้วยหมายเลขโทรศัพท์

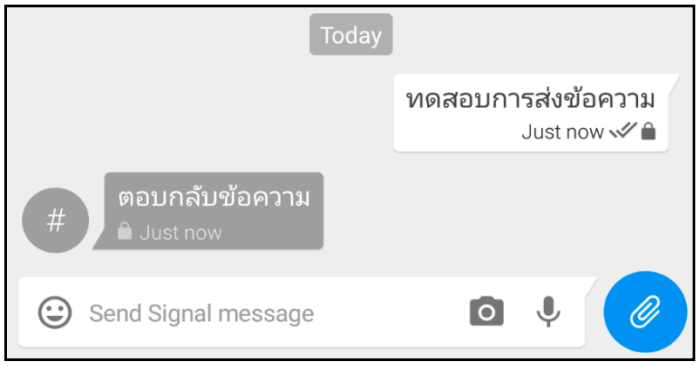


รูปที่ 3-25 เมื่อพบแล้วสามารถแชทได้เลย



รูปที่ 3-26 ส่งข้อความสำเร็จ

อธิบาย: ✓ หมายถึง ส่งข้อความสำเร็จ, ✓✓ หมายถึงผู้รับอ่านข้อความแล้ว



รูปที่ 3-27 เครื่องคู่แชทส่งข้อความตอบกลับ

3.2.11 แอปพลิเคชันบนแอนดรอยด์พร้อมที่จะใช้ในการทดสอบแล้ว

3.3 การทดสอบแอปพลิเคชันตามหัวข้อทดสอบที่ได้กำหนดไว้

3.3.1 ศึกษาหัวข้อต่าง ๆ ใน Best Practice ของ ARO

3.3.2 เลือกหัวข้อต่าง ๆ จาก Best Practice ที่เกี่ยวข้องกับแอปพลิเคชัน Signal

3.3.3 ออกแบบการทดสอบจากหัวข้อต่าง ๆ ที่ถูกเลือก ตามความเหมาะสม

3.3.4 ออกแบบตารางบันทึกผลการทดสอบ โดยมีวิธีการทดสอบอยู่ที่หัวตาราง

ตารางที่ 0-1 ตารางบันทึกผลการทดสอบ

ครั้งที่	ผลลัพธ์	รายละเอียด
1		
2		
3		
สรุป		ผ่าน/ไม่ผ่าน

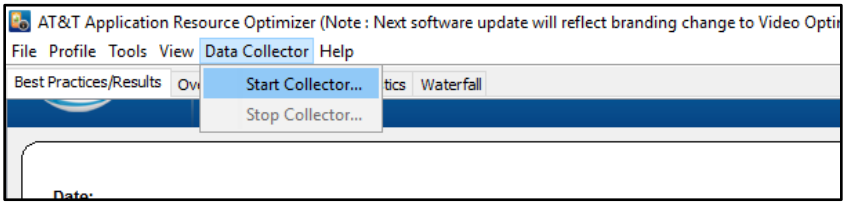
วิธีการทดสอบ : รับข้อความที่มีขนาดใหญ่จากคู่สนทนา

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ส่งสำเร็จ สามารเปิดอ่านได้	ไม่มีบอัดไฟล์ text
2	ส่งสำเร็จ สามารเปิดอ่านได้	-
3	ส่งสำเร็จ สามารเปิดอ่านได้	-
สรุป		ไม่ผ่าน

รูปที่ 3-28 ตัวอย่างตารางที่มีการบันทึกผล

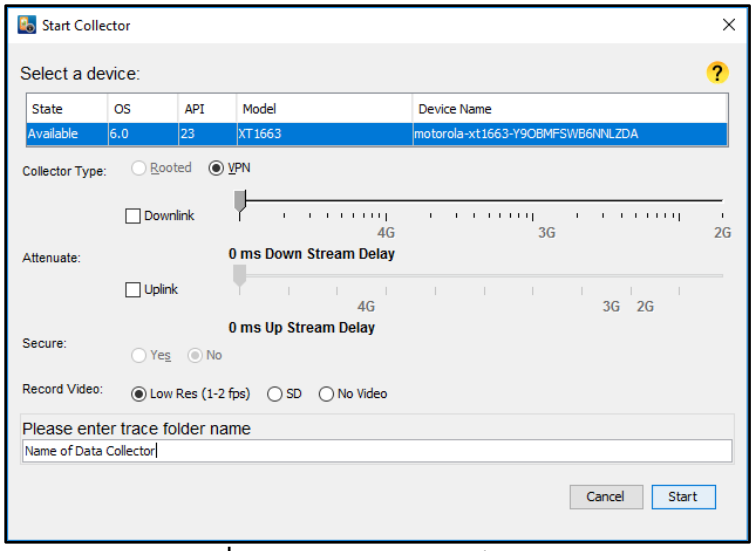
3.3.5 ทำการทดสอบ และบันทึกผลลงไปในตารางบันทึกผลการทดสอบ

3.3.6 เปิดโปรแกรม ARO ไปที่เมนู Data Collector>Start Collector.. (ต้องเชื่อมต่อสมาร์ทโฟนกับคอมพิวเตอร์ก่อน)

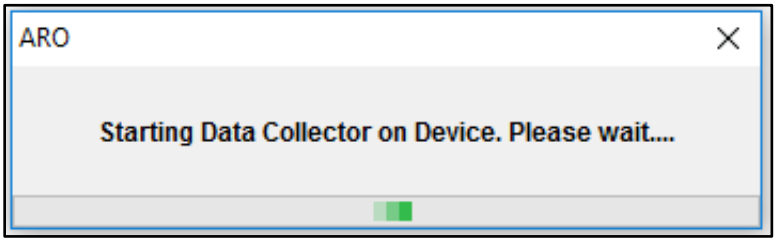


รูปที่ 3-29 การเริ่มเก็บข้อมูล

3.3.7 จะมีหน้าต่างโต้ตอบให้ทำการตั้งค่า และตั้งชื่อจากนั้นกด Start

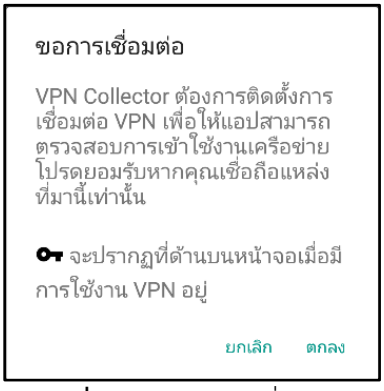


รูปที่ 3-30 หน้าต่างก่อนเก็บข้อมูล



รูปที่ 3-31 รอสักครู่

3.3.8 จากนั้น ARO จะมีการติดตั้งโปรแกรม VPN ใน Android และขออนุญาตในการเข้าถึงการเชื่อมต่อ

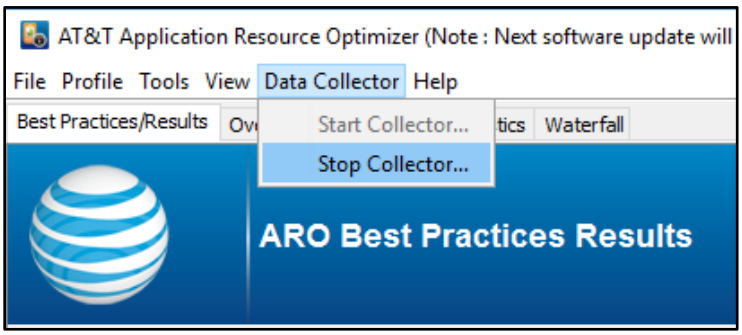


รูปที่ 3-32 ขอการเชื่อมต่อ

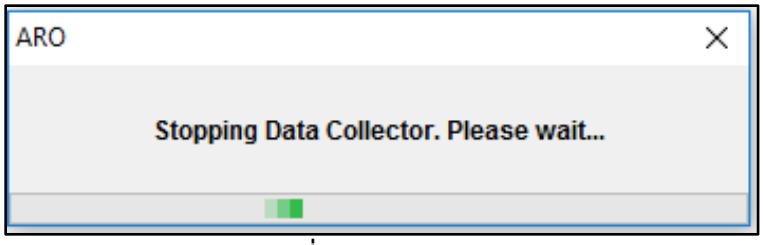
3.3.9 หลังจากนั้นเราก็สามารถเริ่มการทดสอบ Signal ตามการทดสอบที่ได้ออกแบบไว้

รูปที่ 3-33 พร้อมที่จะเก็บข้อมูล

3.3.10 หลักจากทำการทดสอบเสร็จ ไปที่เมนู Data Collector>Stop Collector..

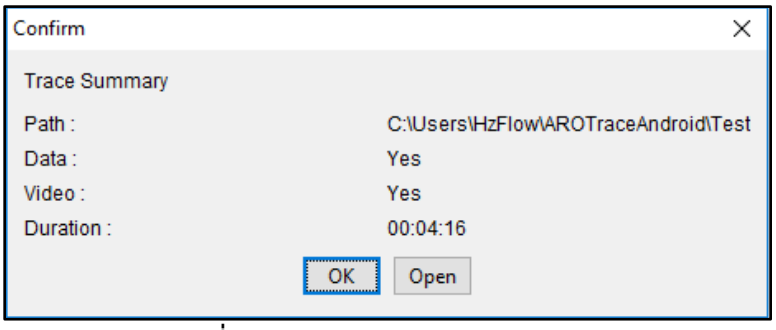


รูปที่ 3-34 Stop Collector



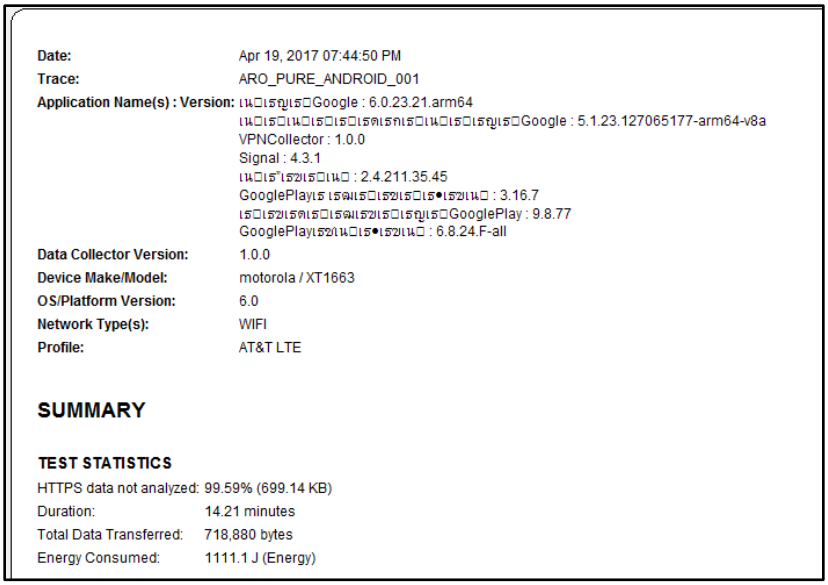
รูปที่ 3-35 รอสักครู่

3.3.11 จากนั้นจะมีกล่องโต้ตอบ (Dialog) ให้เลือกว่า จะเปิดข้อมูลที่เก็บ หรือ ปิดกล่องโต้ตอบด้วยการกด OK หรือ X



รูปที่ 3-36 หลังจาก Stop Collector

3.3.12 รูปที่ 3-37 แสดงหน้าจอเมื่อผู้ใช้เปิดดูผลการทดสอบที่เก็บไว้



รูปที่ 3-37 เมื่อเปิดดูข้อมูลการทดสอบที่เก็บไว้

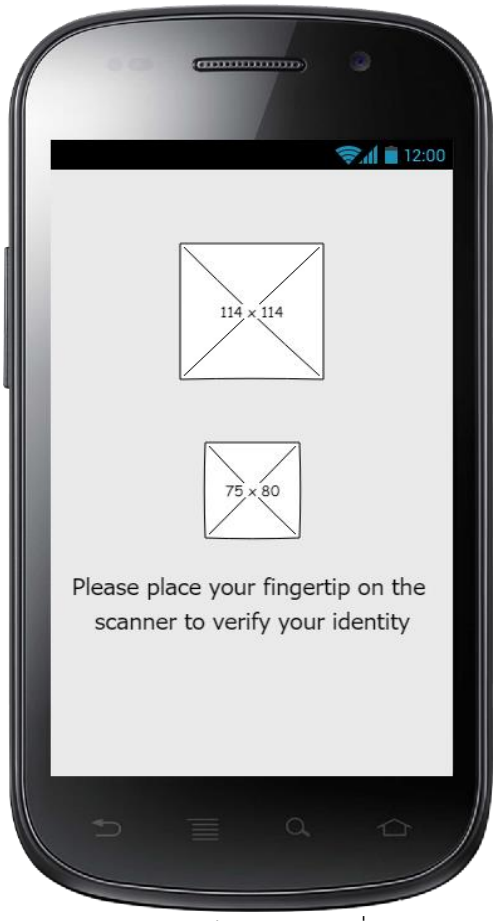
3.3.13 ทำการทดสอบ และบันทึกซ้ำเป็นรวมเป็นจำนวน 3 ครั้ง

3.3.14 วิเคราะห์ผลการทดสอบ

3.3.15 สรุปผลการทดสอบ

3.4 การออกแบบ Layout ของส่วนการยืนยันตัวตน

ก. หากผู้ใช้เลือกการยืนยันตัวตนโดยใช้ลายนิ้วมือ และลงทะเบียนลายนิ้วมือที่สมาร์ทโฟนแล้ว App จะแสดงหน้า Wait For Fingerprint Scan โดยในหน้านี้จะประกอบไปด้วย รูป Logo App Signal, รูปลายนิ้วมือ และข้อความตามลำดับ ดังแสดงในรูปที่ 3-38

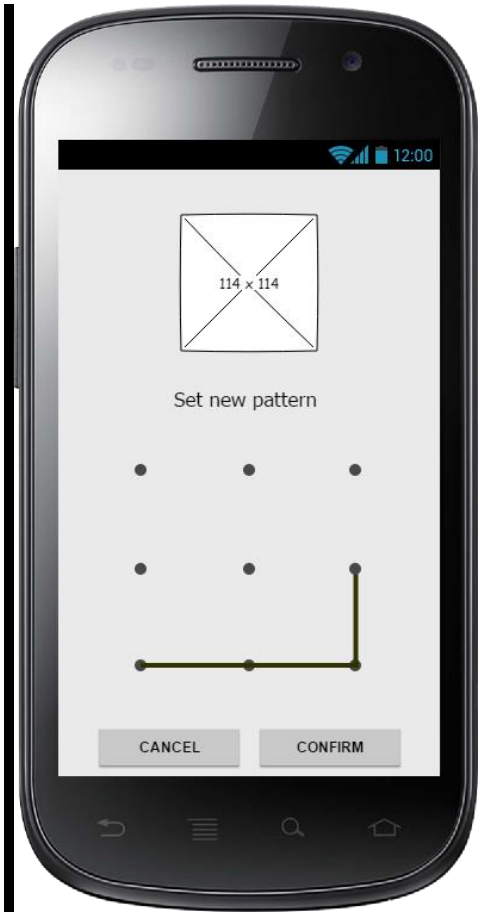


รูปที่ 3-38 “Wait For Fingerprint Scan” เป็น Layout ที่ระบุให้ผู้ใช้ยืนยันตัวตนโดยใช้ลายนิ้วมือ

ข. หากผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern App จะแสดงหน้า Pattern Input (สำหรับสร้าง Pattern) โดยในหน้านี้จะประกอบไปด้วย รูป Logo App Signal, ข้อความ “Set new pattern” และส่วนใส่ Pattern ตามลำดับ ดังแสดงในรูปที่ 3-39

รูปที่ 3-39 “Pattern Input” เป็น Layout สำหรับสร้าง Pattern

ผู้ใช้ใส่ Pattern ที่ผู้ใช้ต้องการ (จะต้องมีความยาวไม่น้อยกว่า 4 จุด แต่ไม่เกิน 9 จุด) จากนั้น App จะแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CANCEL คือ ยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และปุ่ม CONFIRM คือ จดจำ Pattern และดำเนินการต่อ ดังแสดงในรูปที่ 3-40



รูปที่ 3-40 Layout ของหน้า Pattern (ผู้ใช้ใส่ Pattern ครั้งแรก)

หลังจากผู้ใช้ปุ่ม CONFIRM เพื่อจดจำ Pattern และดำเนินการต่อ App จะเปลี่ยนข้อความใต้รูป Logo App Signal จาก “Set new pattern” เป็นข้อความ “Confirm Pattern”ดังแสดงในรูปที่ 3-41

รูปที่ 3-41 Layout ของหน้า Pattern (App แสดงข้อความ Confirm pattern)

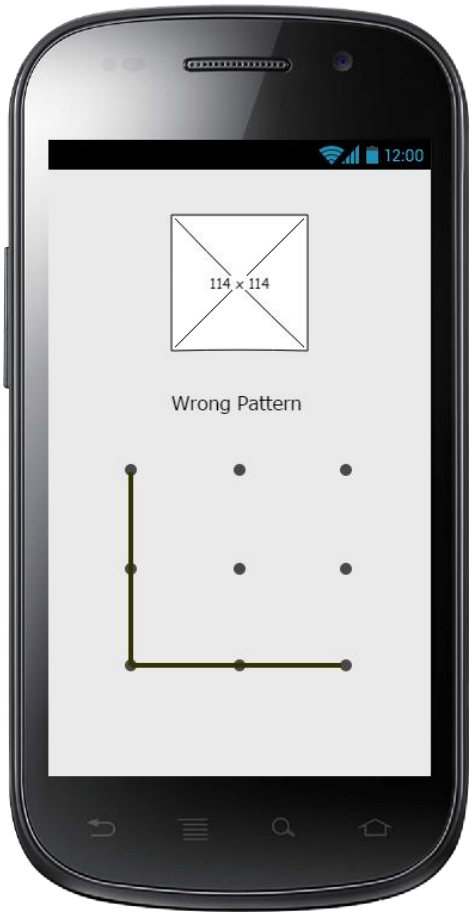
ผู้ใช้ใส่ Pattern อีกครั้ง โดยจะต้องเหมือนกับ Pattern ที่ผู้ใช้ใส่ก่อนหน้า เพื่อเป็นการยืนยันความถูกต้องตรงกัน เมื่อผู้ใช้ใส่ Pattern สำเร็จ App จะแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CANCEL คือ การยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และปุ่ม CONFIRM คือ การบันทึก Pattern ที่ผู้ใช้ใส่ลงใน Shared Preferences ดังแสดงในรูปที่ 3-42

รูปที่ 3-42 Layout ของหน้า Pattern (Pattern ที่ผู้ใช้ใส่ตรงกับ Pattern ที่จดจำไว้ก่อนหน้า)

ค. กรณีผู้ใช้ใส่ Pattern โดยมีความยาวไม่น้อยกว่า 4 จุด App จะแสดงข้อความ “Pattern must contain at least 4 dots!” แจ้งเตือนผู้ใช้ด้านล่างสุด ดังแสดงในรูปที่ 3-43

รูปที่ 3-43 Layout ของหน้า Pattern (App แสดงข้อความแจ้งเตือนผู้ใช้)

ง. กรณีผู้ใช้ใส่ Pattern ยืนยันตัวตนผิด ไม่ตรงกับ Pattern ที่ถูกบันทึกไว้ App จะแสดงข้อความ “Wrong Pattern” ใต้รูป Logo Signal ดังแสดงในรูปที่ 3-44



รูปที่ 3-44 Layout ของหน้า Pattern (ผู้ใช้ใส่ Pattern ไม่ตรงกับ Pattern ที่ถูกบันทึกไว้)

3.5 การออกแบบ Wireframe ของส่วนยืนยันตัวตน และส่วนการตั้งค่า

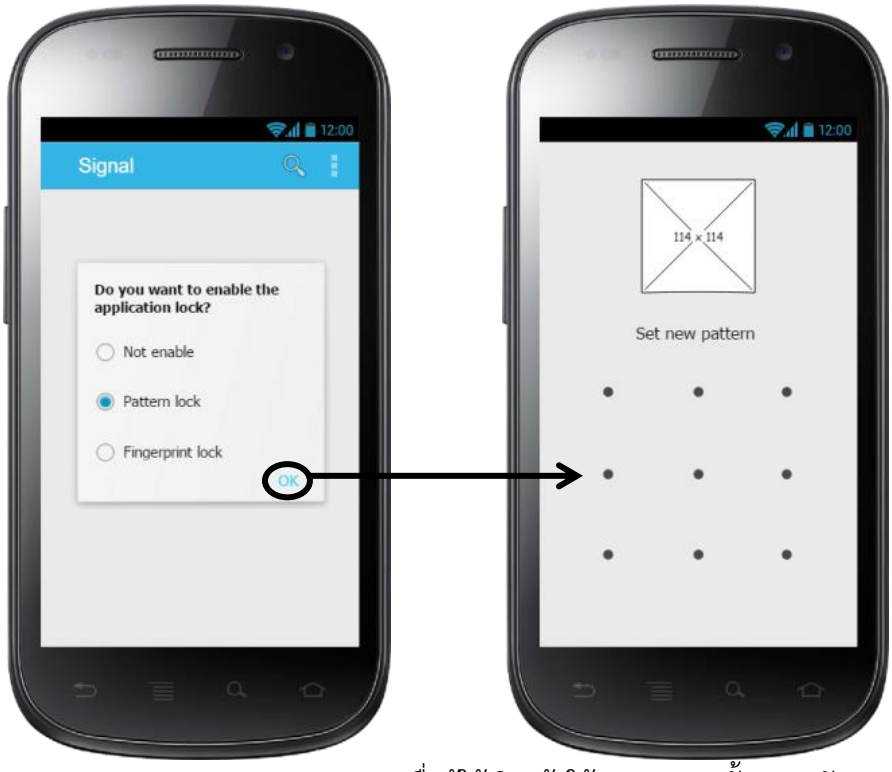
Wireframe ของส่วนยืนยันตัวตน และส่วนการตั้งค่า สามารถแบ่งเป็นเหตุการณ์ต่าง ๆ และกรณีย่อย ๆ ได้ดังนี้

3.5.1 เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App

ก. เมื่อเปิดเข้าใช้งาน App ครั้งแรก หากผู้ใช้เลือกตัวเลือก Not enable และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก และบันทึกตัวเลือกที่ผู้ใช้เลือกใน Shared Preferences และปิด Dialog ลง ผู้ใช้จะพบหน้าหลักของ App ดังแสดงในรูปที่ 3-45

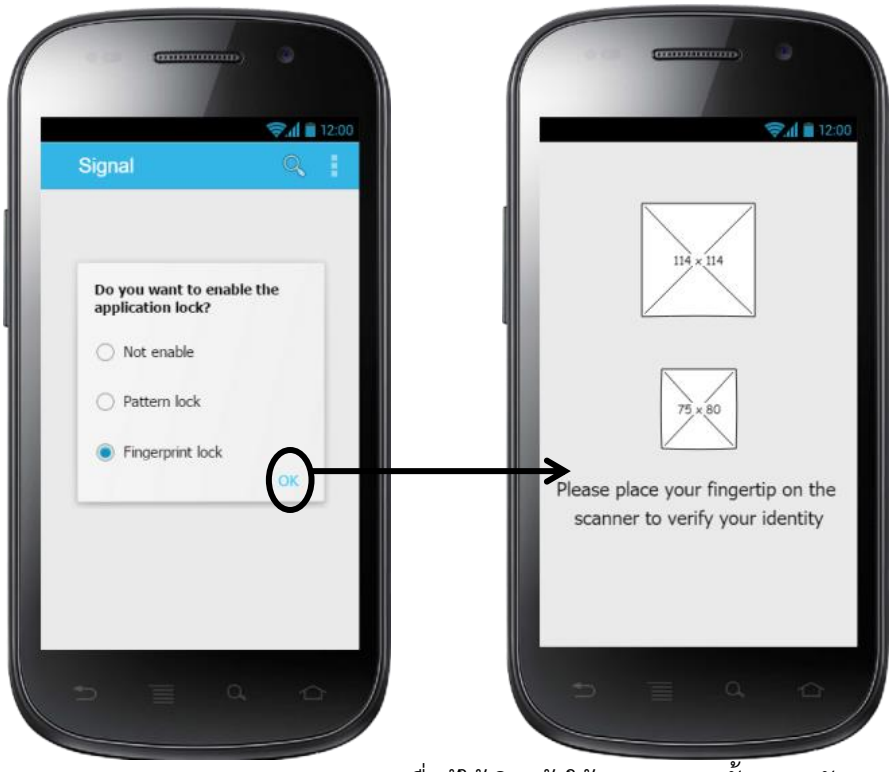
รูปที่ 3-45 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App

ข. เมื่อเปิดเข้าใช้งาน App ครั้งแรก หากผู้ใช้เลือกตัวเลือก Pattern lock และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก และบันทึกตัวเลือกที่ผู้ใช้เลือกใน Shared Preferences จากนั้นจะแสดงหน้า Pattern Input สำหรับให้ผู้ใช้สร้าง Pattern ที่ใช้ในการยืนยันตัวตน ดังแสดงในรูปที่ 3-46



รูปที่ 3-46 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App

ค. ในกรณีผู้ใช้ได้ลงทะเบียนลายนิ้วมือในสมาร์ทโฟน หลังจากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก และบันทึกตัวเลือกที่ผู้ใช้เลือกใน Shared Preferences จากนั้น App จะแสดงหน้า “Wait For Fingerprint Scan” สำหรับให้ผู้ใช้ยืนยันตัวตนโดยใช้ลายนิ้วมือ ดังแสดงในรูปที่ 3-47



รูปที่ 3-47 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App

ง. หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม OK แต่ App ไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App จะแสดง Dialog แจ้งเตือนผู้ใช้ว่า “Register at least one fingerprint in Settings” ดังแสดงในรูปที่ 3-48 หากผู้ใช้กดปุ่ม CONFIRM App จะแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ

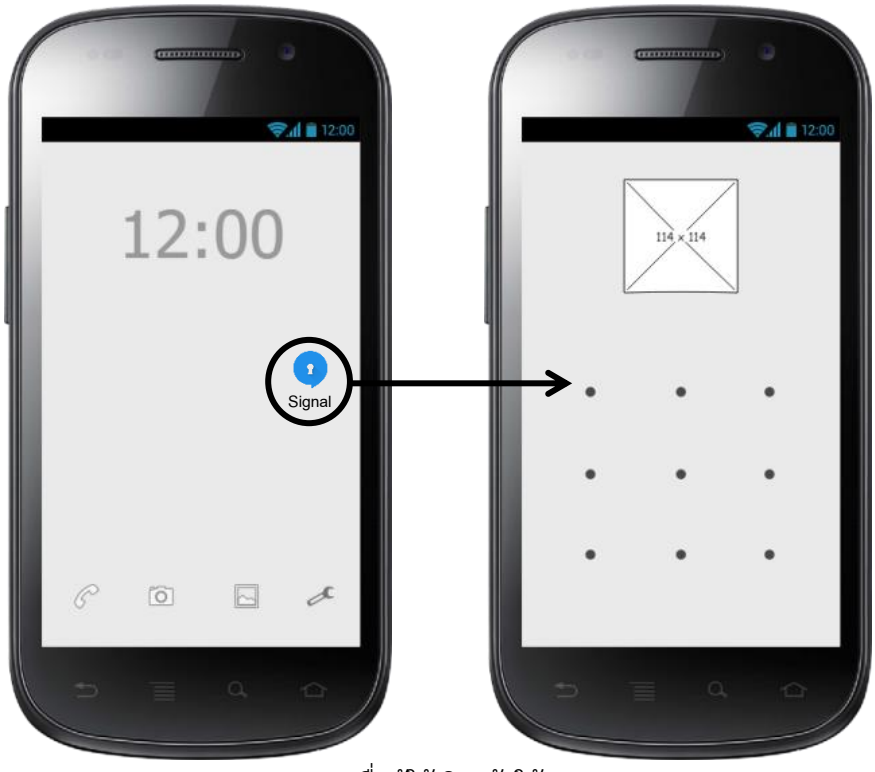
รูปที่ 3-48 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App

3.5.2 เปิดเข้าใช้งาน App จาก Icon ของ App Signal

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตน ผู้ใช้จะสามารถเปิดเข้าใช้งาน App ได้ทันทีโดยไม่ต้องผ่านการยืนยันตัวตน ดังแสดงในรูปที่ 3-49

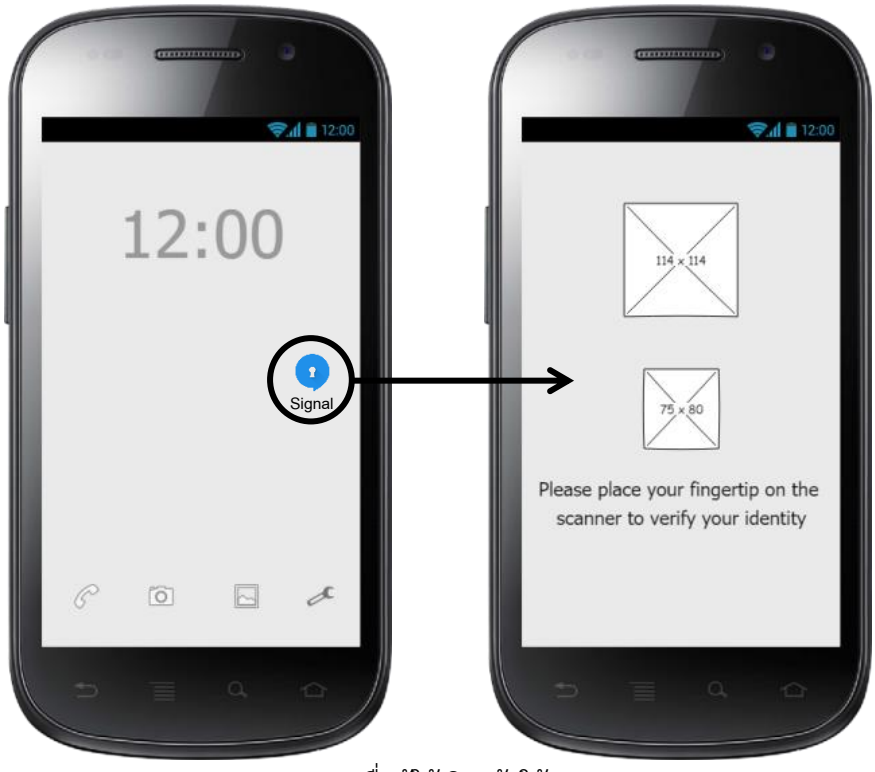
รูปที่ 3-49 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App จาก Icon ของ App Signal

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกดปุ่ม Icon App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-50



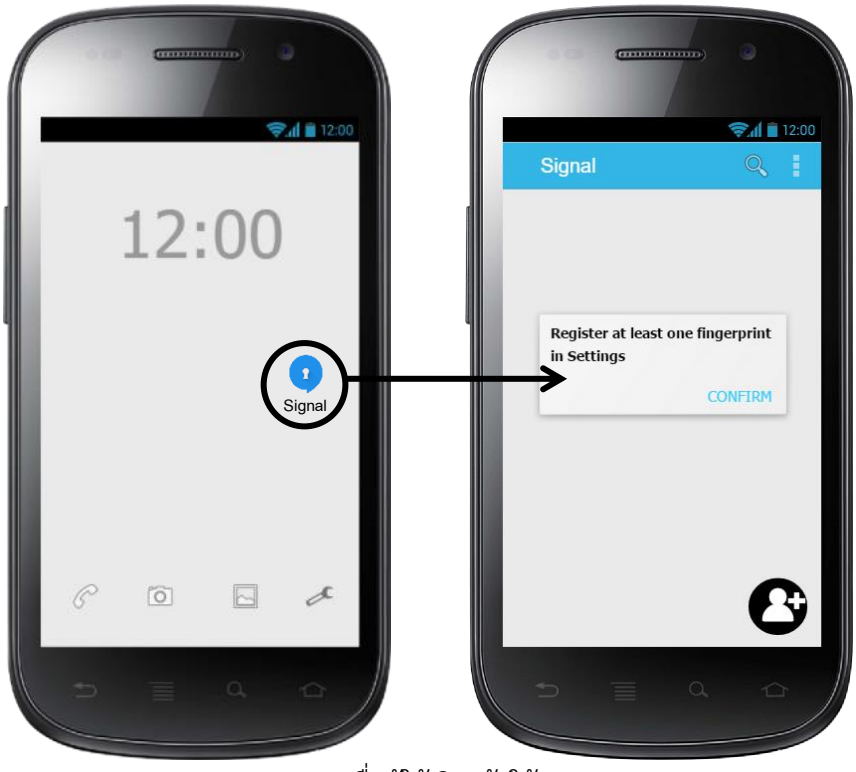
รูปที่ 3-50 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App จาก Icon ของ App Signal

ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-51



รูปที่ 3-51 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App จาก Icon ของ App Signal

ง. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint แต่ App ไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon App จะแสดง Dialog แจ้งเตือนผู้ใช้ ดังแสดงในรูปที่ 3-52 หากผู้ใช้กดปุ่ม CONFIRM App จะแสดงหน้าเมนู Setting ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ



รูปที่ 3-52 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App จาก Icon ของ App Signal

3.5.3 ผู้ใช้ปลด Lock จากหน้า Lock Screen แล้วเข้าหน้า App Signal (ที่เปิดค้างไว้ก่อนหน้าทันที)

กรณีกลับเข้าหน้า App จากหน้า Lock Screen สามารถเกิดขึ้นได้ หากผู้ใช้เลิกใช้ App เกินเวลาที่ตั้ง Lock Screen ไว้ หรือผู้ใช้กดปุ่ม Power Button ผู้วิจัยต้องการลดความเสี่ยงด้านความปลอดภัย ดังนั้นผู้วิจัยจึงออกแบบให้ผู้ใช้ต้องทำการยืนยันตัวตนอีกครั้งหนึ่งหลังผู้ใช้ปลด Lock Screen

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตนก่อนเข้า App เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้าของ App ที่เปิดค้างไว้ก่อนหน้า ดังแสดงในรูปที่ 3-53

รูปที่ 3-53 Wireframe และ Flow ของ App เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen แล้วเข้าหน้า App Signal (ที่เปิดค้างไว้ก่อนหน้าทันที)

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน App ดังแสดงในรูปที่ 3-54

รูปที่ 3-54 Wireframe และ Flow ของ App เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen แล้วเข้าหน้า App Signal (ที่เปิดค้างไว้ก่อนหน้าทันที)

ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน App ดังแสดงในรูปที่ 3-55

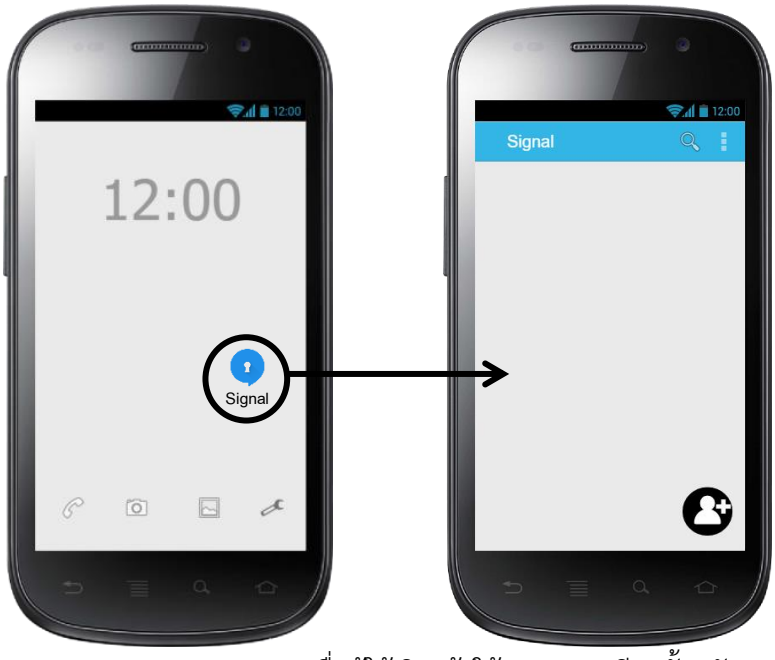
รูปที่ 3-55 Wireframe และ Flow ของ App เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen แล้วเข้าหน้า App Signal (ที่เปิดค้างไว้ก่อนหน้าทันที)

3.5.4 เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที

กรณีการเปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที สามารถเกิดขึ้นได้หากผู้ใช้เลิกใช้ App หรือเปิดเข้าใช้งาน App อื่นนานเกินระยะเวลา 5 นาที ผู้วิจัยต้องการลดความเสี่ยงด้านความปลอดภัย ดังนั้นผู้วิจัยจึงออกแบบให้ผู้ใช้ต้องทำการยืนยันตัวตนก่อนเข้าใช้งาน

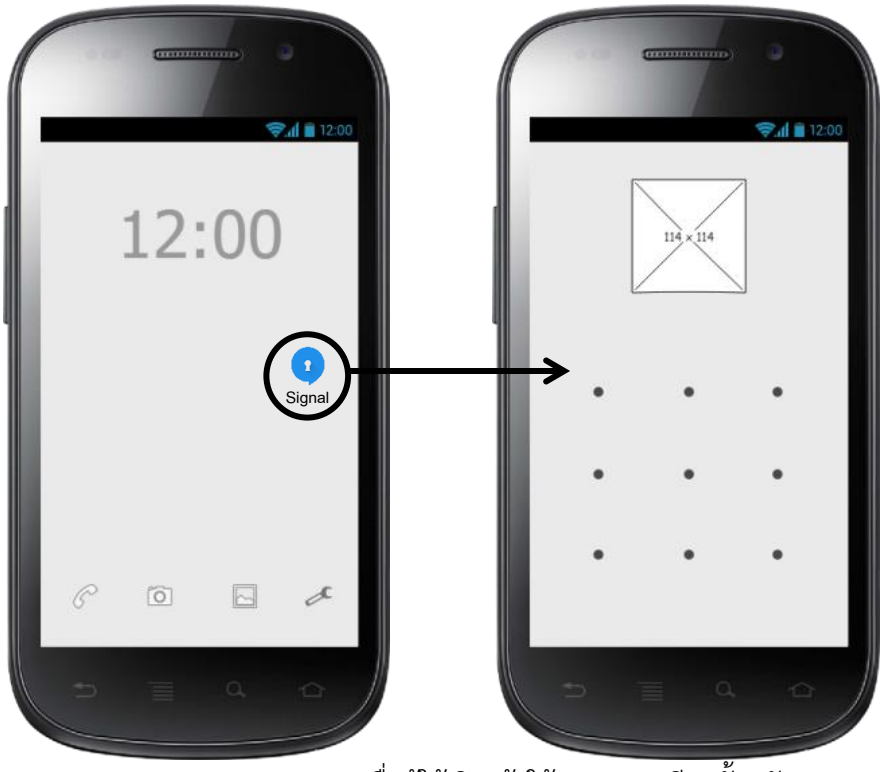
ผู้วิจัยพบว่า การกลับเข้า App Signal โดยใช้ปุ่ม Icon และโดยเลือกจาก Resent App จะให้พฤติกรรมต่างกันนั่นคือ หากผู้ใช้กลับเข้า App โดยใช้ปุ่ม Icon Android จะแสดงหน้าหลักของ App เสมอแต่หากผู้ใช้กลับเข้า App โดยเลือก App จาก Resent App Android จะแสดงหน้า Activity ที่เคยทำงานล่าสุด (Resent Activity) เสมอผู้จัดทำได้ออกแบบการยืนยันตัวตนให้มีพฤติกรรมเหมือนเดิมไม่ว่าผู้ใช้จะกลับเข้า App โดยใช้วิธีใด

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตน ผู้ใช้สามารถเปิดเข้าใช้งาน App ได้ทันทีโดยไม่ต้องผ่านการยืนยันตัวตน ไม่ว่าผู้ใช้จะกดเปิด App จาก Icon หรือจาก Resent App ก็ตาม ดังแสดงในรูปที่ 3-56



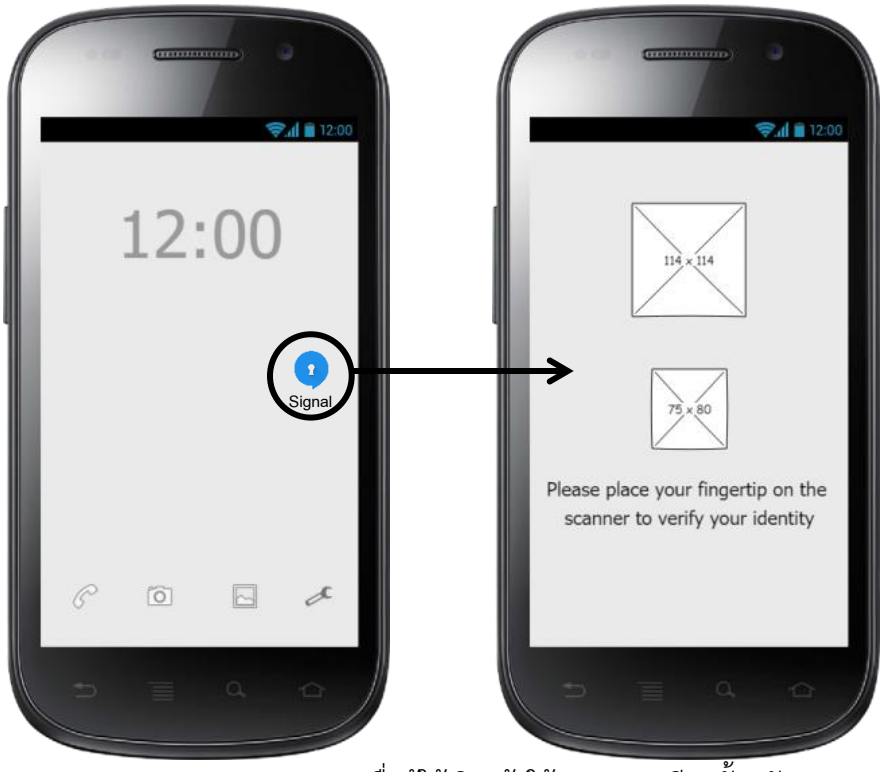
รูปที่ 3-56 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon หรือจาก Resent App ก็ตาม App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-57



รูปที่ 3-57 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที

ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon หรือจาก Resent App ก็ตาม App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-58



รูปที่ 3-58 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที

3.5.5 เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) ไม่เกินระยะเวลา 5 นาที

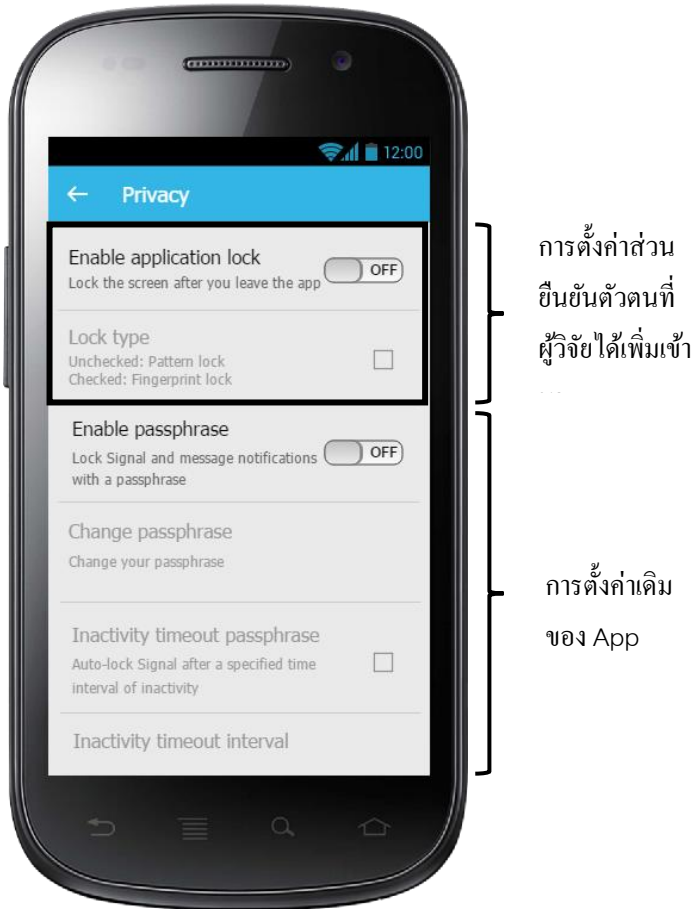
ผู้ใช้สามารถเปิดเข้าใช้งาน App อีกครั้ง ได้โดยไม่ต้องผ่านการยืนยันตัวตนใดๆ หาก App อยู่ในสถานะหยุด (Stop State) ไม่เกินระยะเวลา 5 นาที แม้ว่าผู้ใช้จะเคยได้เลือกให้มีการยืนยันตัวตนก่อนการเข้าใช้งาน App ก็ตาม ดังแสดงในรูปที่ 3-59

รูปที่ 3-59 Wireframe และ Flow ของ App เมื่อผู้ใช้เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) ไม่เกินระยะเวลา 5 นาที

3.5.6 การตั้งค่า

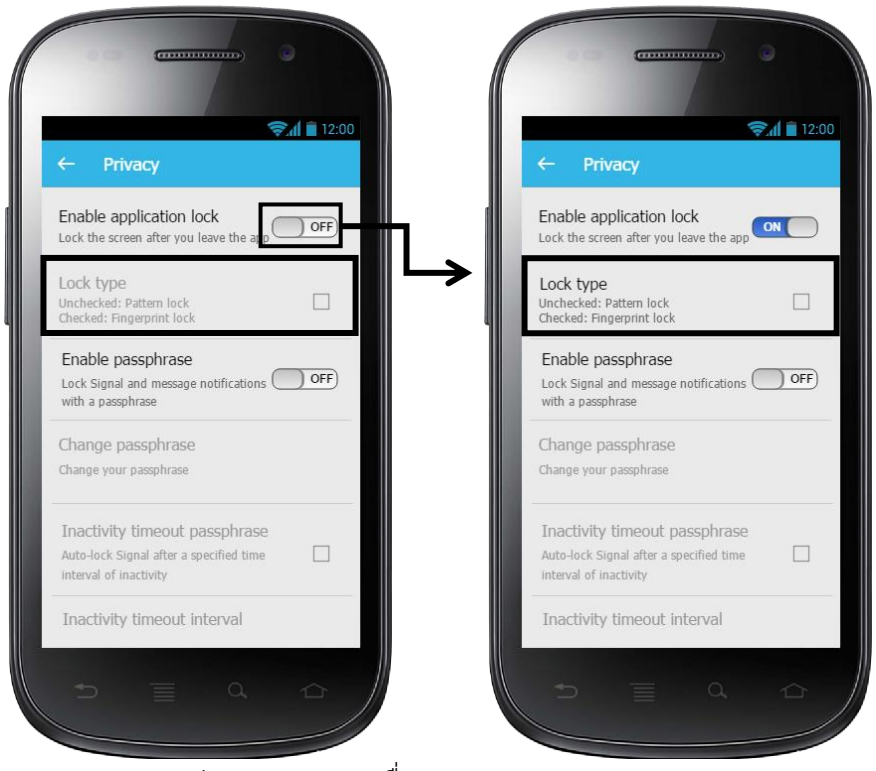
นอกเหนือจากการตั้งค่ายืนยันตัวตนเมื่อผู้ใช้เปิดเข้าใช้งาน App ครั้งแรก (หลังจากติดตั้ง App (ดังหัวข้อ 3.5.1)) ผู้ใช้สามารถตั้ง /เปลี่ยนชนิดการยืนยันตัวตนได้ภายหลัง โดยใช้เมนูดังรายละเอียดต่อไปนี้

เมนู Setting -> Privacy ของ App Signal มีปุ่มที่เกี่ยวกับงานวิจัยนี้ 2 ปุ่ม ได้แก่ ปุ่มแบบ Switch ชื่อ Enable application lock (ใช้สำหรับการเปิด-ปิดการยืนยันตัวตน) และปุ่มแบบ Checkbox ชื่อ Lock type (ใช้สำหรับการเปลี่ยนชนิดการยืนยันตัวตนระหว่างชนิด Pattern กับชนิด Fingerprint) ดังแสดงในรูปที่ 3-60



รูปที่ 3-60 หน้าเมนู Setting -> Privacy ของ App Signal

หากปุ่ม Enable application lock มีสถานะเป็น OFF ปุ่ม Lock type จะไม่สามารถใช้งานได้ และสีตัวหนังสือ Lock type จะเป็นสีเทา (ปุ่มถูก Disable) ในทางตรงกันข้าม หากปุ่ม Enable application lock มีสถานะเป็น ON ปุ่ม Lock type จะสามารถใช้งานได้ และสีตัวหนังสือ Lock type จะเป็นสีดำ (ปุ่มถูก Enable) ดังแสดงในรูปที่ 3-61



รูปที่ 3-61 สถานะของปุ่ม Lock type เมื่อ Enable / Disable application lock

กรณีที่สมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ หากผู้ใช้ตั้งค่าปุ่ม Enable application lock ให้มีสถานะเป็น ON แล้วก็ตาม ปุ่ม Lock type จะไม่สามารถใช้งานได้สีตัวหนังสือ Lock type จะเป็นสีเทา (ปุ่มถูก Disable) ดังแสดงในรูปที่ 3-62

รูปที่ 3-62 Wireframe และ Flow ของ App เมื่อผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก OFF เป็น ON

ก. เมื่อผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ OFF เป็น ON App แสดง Dialog ให้ผู้ใช้เลือกชนิดการยืนยันตัวตน หากผู้ใช้เลือกตัวเลือก Pattern lock และกดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าไม่ถูกเช็คเครื่องหมายถูก (ชนิด Pattern) และแสดงหน้าสร้าง Pattern ดังแสดงในรูปที่ 3-63

ข. เมื่อผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ OFF เป็น ON App แสดง Dialog ให้ผู้ใช้เลือกชนิดการยืนยันตัวตน หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้า Wait For Fingerprint Scan สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 3-64

ค. หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม CONFIRM แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App แสดง Dialog แจ้งเตือนผู้ใช้ หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ ดังแสดงในรูปที่ 3-65

ง. เมื่อผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ ON เป็น OFF App จะแสดง Dialog แจ้งเตือนผู้ใช้ หากผู้ใช้กดปุ่ม CANCEL App จะปิด Dialog แจ้งเตือนลง แต่หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น OFF และลบ Pattern ที่ผู้ใช้ได้บันทึกไว้สำหรับการยืนยันตัวตนออกจาก Shared Preferences ดังแสดงในรูปที่ 3-66

รูปที่ 3-63 Wireframe และ Flow ของ App เมื่อผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก OFF เป็น ON

รูปที่ 3-64 Wireframe และ Flow ของ App เมื่อผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก OFF เป็น ON

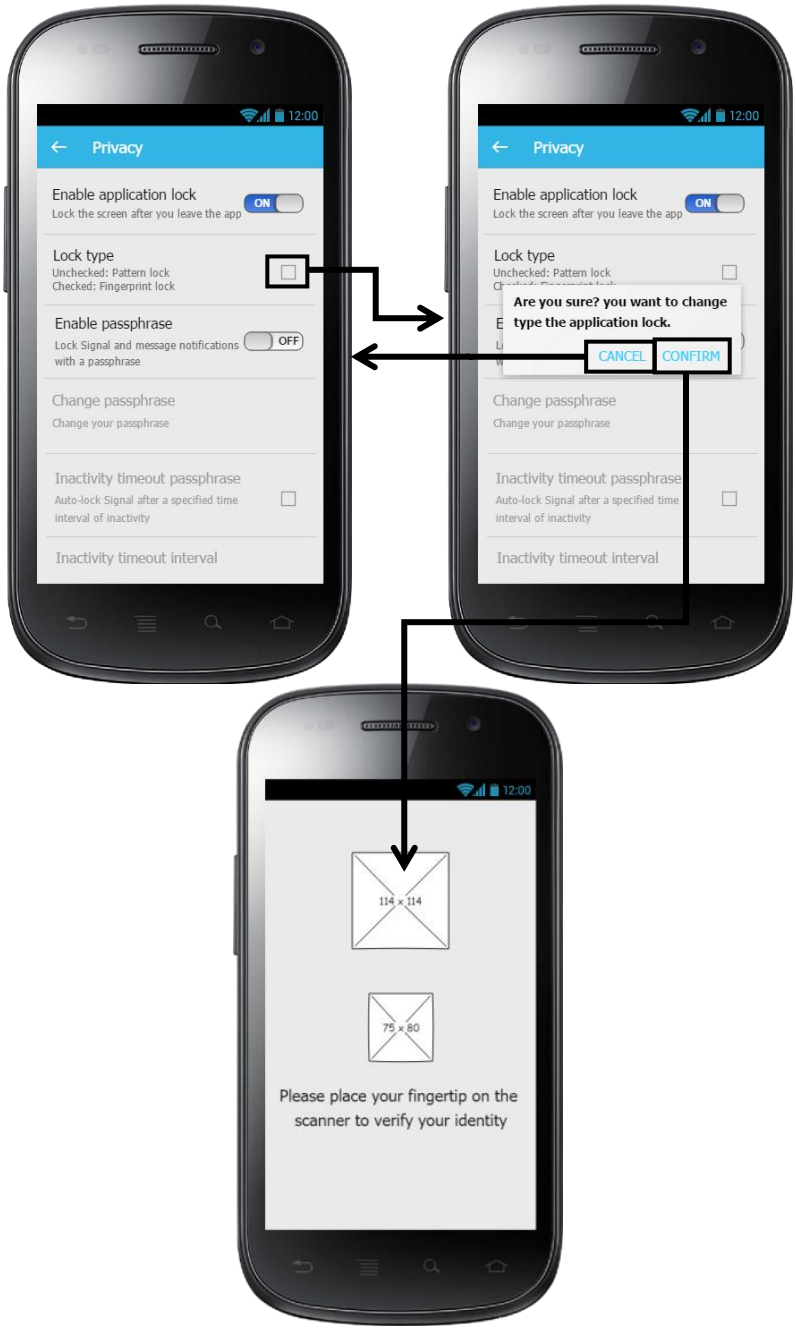
รูปที่ 3-65 Wireframe และ Flow ของ App เมื่อผู้ใช้เลือกตัวเลือก Fingerprint lock แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน

รูปที่ 3-66 Wireframe และ Flow ของ App เมื่อผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก ON เป็น OFF

ก. เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนชนิดการยืนยันตัวตนจากแบบ Pattern เป็นแบบ Fingerprint App จะแสดง Dialog แจ้งเตือน หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้า Wait For Fingerprint Scan สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 3-67 (ในกรณีนี้ สมาร์ทโฟนมีลายนิ้วมือที่ลงทะเบียนแล้ว อย่างน้อย 1 ลายนิ้วมือ)

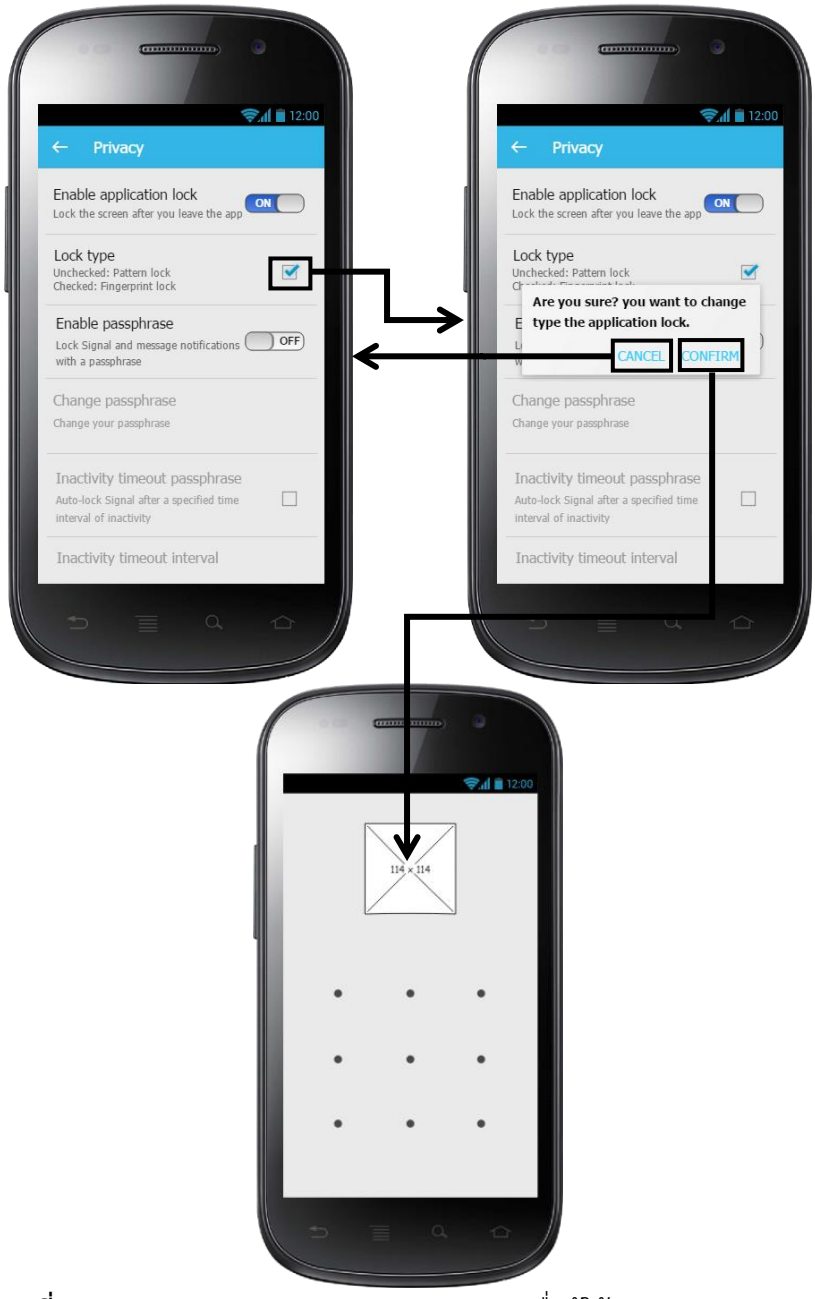
ข. เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนชนิดการยืนยันตัวตนจากแบบ Pattern เป็นแบบ Fingerprint แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App จะแสดง Dialog แจ้งเตือนผู้ใช้ หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ ดังแสดงในรูปที่ 3-68

ค. เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนประเภทการยืนยันตัวตนจากแบบ Fingerprint เป็นแบบ Pattern App จะแสดง Dialog แจ้งเตือน หากผู้ใช้กดปุ่ม CONFIRM App จะแสดงหน้า Pattern Input สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 3-69



รูปที่ 3-67 Wireframe และ Flow ของ App เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก)

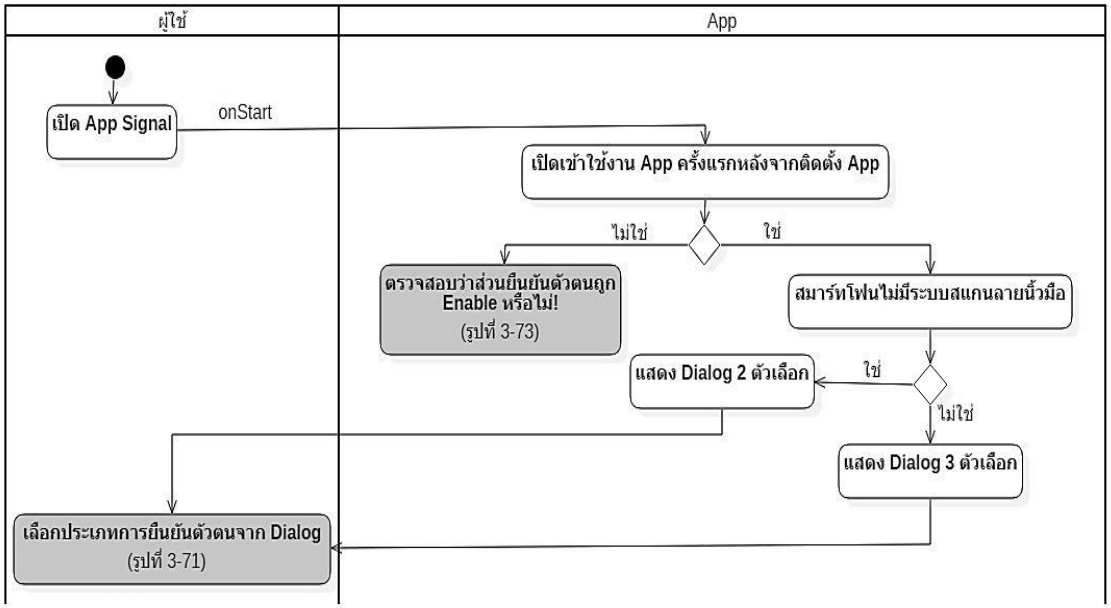
รูปที่ 3-68 Wireframe และ Flow ของ App เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน



รูปที่ 3-69 Wireframe และ Flow ของ App เมื่อผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มถูกเช็คเครื่องหมายถูก)

3.6 การออกแบบแอททิวิตี้ไดอะแกรมของส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal)

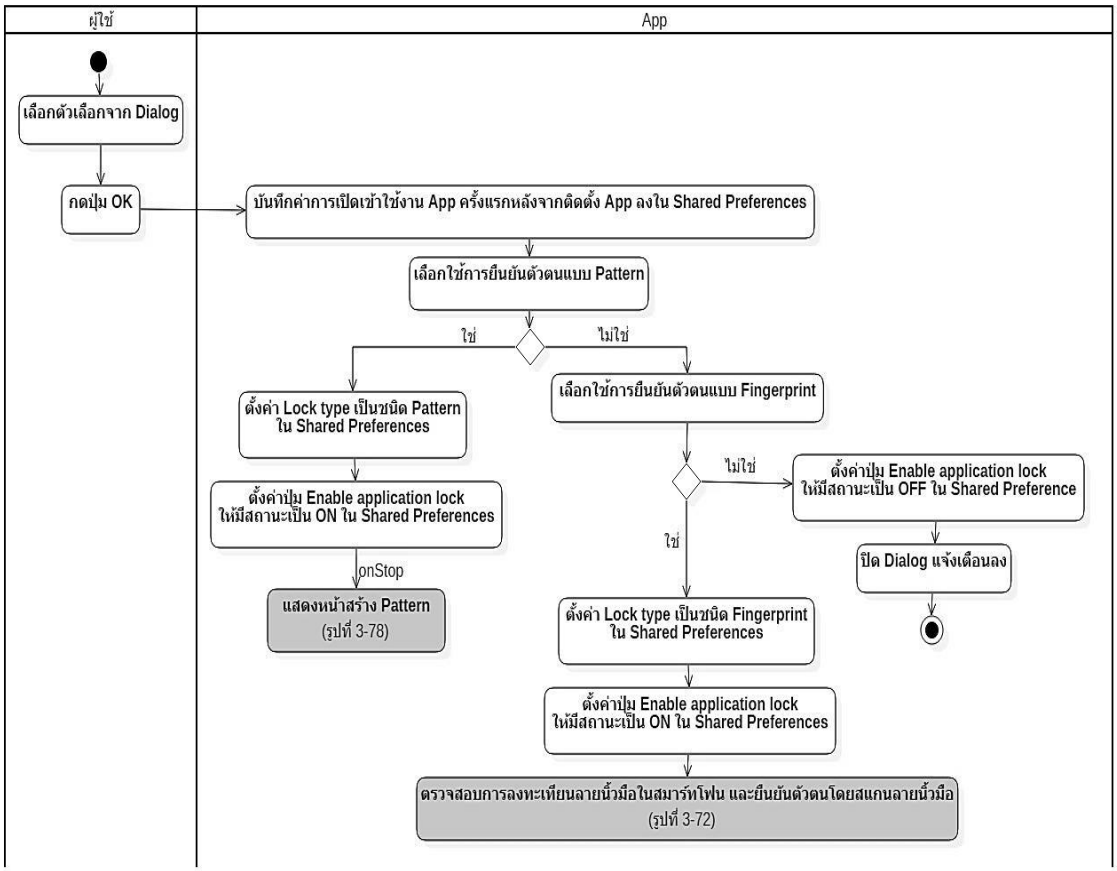
- ผู้ใช้เปิด App Signal จากนั้น App จะตรวจสอบเงื่อนไขว่า “เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App หรือไม่” หากไม่ใช่ App จะตรวจสอบว่าผู้ใช้ได้ Enable หรือ Disable ส่วนการยืนยันตัวตน, แต่ถ้าใช่ และสมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ App จะแสดง Dialog ว่า “Do you want to enable the application lock?” 2 ตัวเลือก (Not enable, Pattern lock) แต่ถ้าสมาร์ทโฟนมีระบบสแกนลายนิ้วมือจะแสดง Dialog ว่า “Do you want to enable the application lock?” 3 ตัวเลือก (Not enable, Pattern lock, Fingerprint lock) จากนั้นผู้ใช้ทำการเลือกประเภทการยืนยันตัวตนจาก Dialog ดังแสดงในรูปที่ 3-70



รูปที่ 3-70 แอททิวิตี้ไดอะแกรมผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากเพิ่งติดตั้ง App

- ผู้ใช้เลือกตัวเลือกจาก Dialog ที่อยู่ในรูปหน้า 3-70 และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก (หลังจากเพิ่งติดตั้ง App) ลงใน Shared Preferences และ App จะตรวจสอบเงื่อนไขว่า ถ้าผู้ใช้“เลือกใช้การยืนยันตัวตนแบบ Pattern” App จะตั้งค่า Lock type ใน Shared Preferences เป็นชนิด Pattern รวมถึงตั้งค่าปุ่ม Enable application lock ใน Shared Preferences

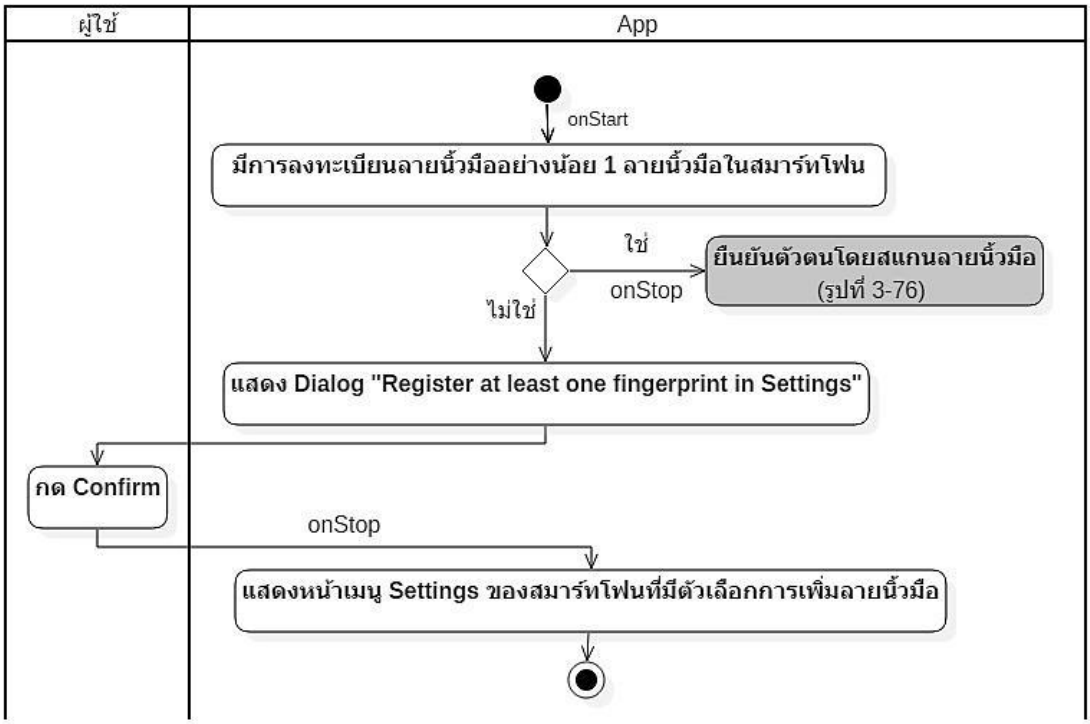
ให้มีสถานะเป็น ON และ App จะแสดงหน้าสร้าง Pattern แต่ถ้าผู้ใช้ “เลือกใช้การยืนยันตัวตนแบบ Fingerprint” App จะตั้งค่า Lock type ใน Shared Preferences เป็นชนิด Fingerprint รวมถึงตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON และตรวจสอบการลงทะเบียนลายนิ้วมือในสมาร์ทโฟน และยันยันตัวตนโดยสแกนลายนิ้วมือ แต่ถ้าผู้ใช้ “เลือก Not enable” App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น OFF และปิด Dialog แจ้งเตือนลง ดังแสดงในรูปที่ 3-71



รูปที่ 3-71 แอททิวิตี้ไดอะแกรมเลือกประเภทการยืนยันตัวตนจาก Dialog (ต่อจากรูปที่ 3-70)

- App ตรวจสอบเงื่อนไขว่าผู้ใช้ “มีการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟนหรือไม่” ถ้าไม่มี App จะแสดง Dialog แจ้งเตือนผู้ใช้ หากผู้ใช้กด Confirm App จะแสดงหน้าเมนู

Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ แต่ถ้ามี App จะให้ยืนยันตัวตนโดยการสแกนลายนิ้วมือ ดังแสดงในรูปที่ 3-72



รูปที่ 3-72 แอททิวิตี้ไดอะแกรมตรวจสอบการลงทะเทียนลายนิ้วมือในสมาร์ทโฟน และยืนยันตัวตนโดยสแกนลายนิ้วมือ (ต่อจากรูปที่ 3-71, 3-75, 3-81)

- App ตรวจสอบเงื่อนไขว่า “ส่วนยืนยันตัวตนถูก Enable หรือไม่” ถ้าไม่ถูก Enable App จะแสดงหน้า App Signal แต่ถ้าถูก Enable App จะทำงานต่างกันในกรณีดังต่อไปนี้

กรณีที่ 1 ผู้ใช้เพิ่งมาจากหน้า Lock Screen แล้วเข้า Activity นี้ทันที App จะตรวจสอบประเภทการยืนยันตัวตน และแสดงหน้ายืนยันตัวตน

กรณีที่ 2 ผู้ใช้เข้า Activity นี้โดยไม่ได้มาจากหน้า Lock Screen ทันที และ Activity นี้เป็นหน้าหลักของ App Signal ในกรณีนี้ App จะต้องตัดสินใจว่าจะให้ผู้ใช้ยืนยันตัวตนหรือไม่ โดยหากผู้ใช้เคยได้ออกจาก Activity แล้วกลับเข้ามาใหม่ (StopTime ไม่เท่ากับ 0) และ App ยังไม่ได้ถูก Lock อยู่ App จะ

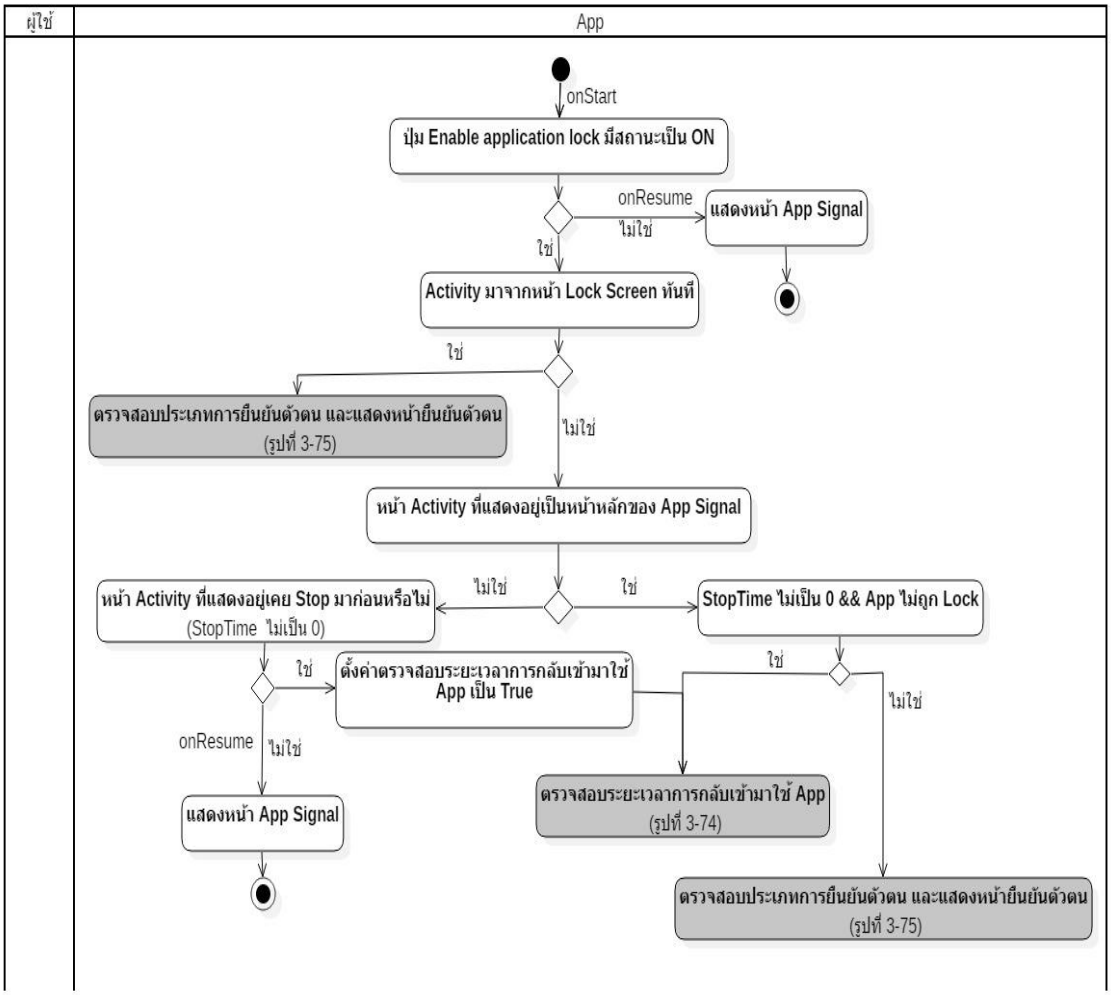
ตรวจสอบระยะเวลาการกลับเข้ามาใช้ App ต่อ (หากเกิน 5 นาที ผู้ใช้ต้องยืนยันตัวตน ดังแสดงในรูปที่ 3-74)

กรณีที่ 2 ข้างต้นครอบคลุมถึงเหตุการณ์ที่ผู้ใช้กดปุ่ม Icon จากหน้า Home เพื่อเข้าใช้ App อีกครั้ง ซึ่งการกดปุ่ม Icon นี้จะทำให้ผู้ใช้กลับเข้าหน้าหลักของ App ทุกครั้ง (โดยก่อนหน้านี้ผู้ใช้ต้องปลด Lock หน้า Activity อื่น ๆ ให้เสร็จก่อนแต่ผู้ใช้ดันกดปุ่ม Home ออกแล้วก็กลับเข้ามาใหม่) การที่กลับเข้าหน้าหลักจาก Activity อื่นที่ถูก Lock (StopTime ไม่เป็น 0) ก่อให้เกิดปัญหา App ไม่ Lock หน้าหลักนี้ ดังนั้นผู้วิจัยจึงต้องเพิ่มเงื่อนไขสำหรับตรวจสอบว่า App กำลังถูก Lock อยู่ที่หน้า Activity ใด ๆ ก่อนเข้าหน้าหลักหรือไม่ ในกรณีที่ 2 ดังได้อธิบายข้างต้น

กรณีที่ 3 ผู้ใช้เข้า Activity นี้โดยไม่ได้มาจากหน้า Lock Screen ทันที และ Activity นี้เป็นหน้าหลักของ App Signal เหมือนในกรณีที่ 2 แต่ผู้ใช้ยังไม่เคยออกจาก Activity นี้เลย (StopTime เท่ากับ 0 เช่น ในกรณีผู้ใช้เปิด App) หรือ App กำลังถูก Lock อยู่ App จะตรวจสอบประเภทการยืนยันตัวตน และแสดงหน้ายืนยันตัวตน

กรณีที่ 4 ผู้ใช้เข้า Activity นี้โดยไม่ได้มาจากหน้า Lock Screen ทันทีแต่ Activity นี้ไม่ใช่หน้าหลักของ App Signal App จะตรวจสอบเงื่อนไขต่อว่า “หน้า Activity ที่แสดงอยู่เคย Stop มาก่อนหรือไม่” ถ้าไม่เคยถูก Stop มาก่อน (StopTime เท่ากับ 0 ซึ่งเกิดทุกครั้งที่ App ออกจากหน้าหลักเข้าหน้า Activity อื่น ๆ App จะ initialize ค่า StopTime ให้เป็น 0 ทุกครั้ง) App จะแสดงหน้า App Signal โดยไม่ต้องยืนยันตัวตน แต่ถ้าเคยถูก Stop มาก่อน App จะตั้งค่าตรวจสอบระยะเวลาการกลับเข้ามาใช้ App เป็น True และตรวจสอบระยะเวลาการกลับเข้ามาใช้ App ดังแสดงในรูปที่ 3-73

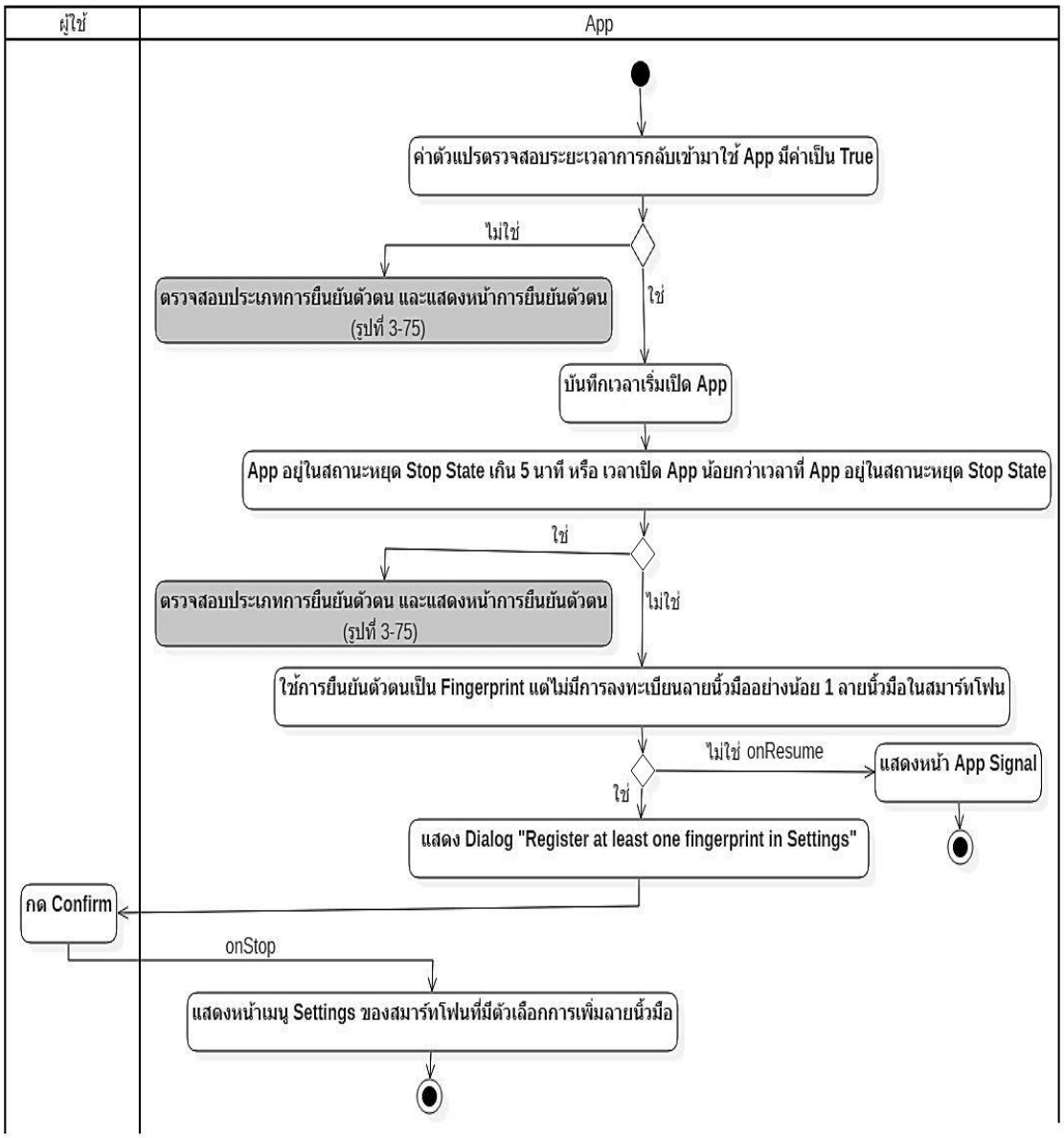
เหตุผลการออกแบบการทำงานของ App ในกรณีที่ 2, 3 และ 4 ให้ดูในหัวข้อที่ 4.2.6 - 4.2.9



รูปที่ 3-73 แอททิวิตี้ไดอะแกรมตรวจสอบว่าส่วนยืนยันตัวตนถูก Enable หรือไม่ (ต่อจากรูปที่ 3-70)

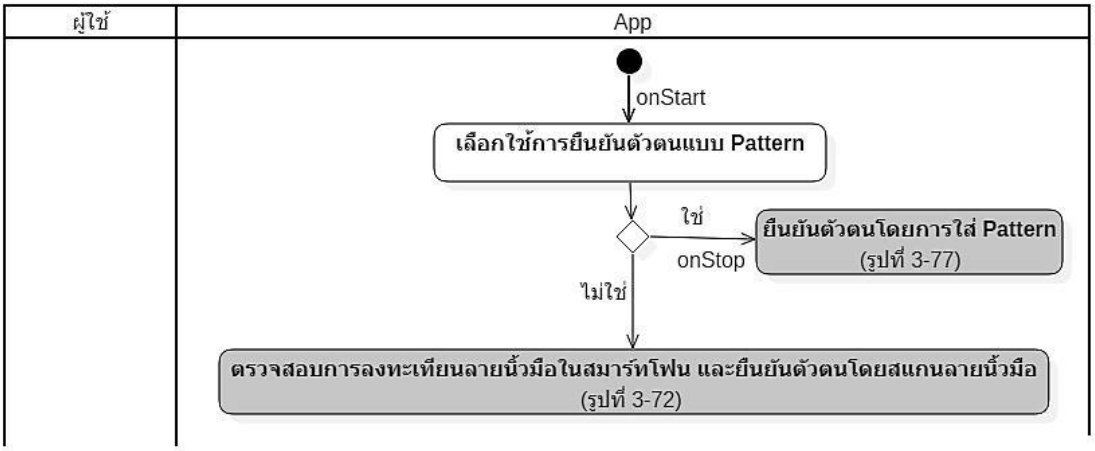
- ถ้า “ค่าตัวแปรตรวจสอบระยะเวลาการกลับเข้ามาใช้ App ว่ามีค่าเป็น False” App จะตรวจสอบประเภทการยืนยันตัวตน และแสดงหน้าการยืนยันตัวตน แต่ถ้าค่าเป็น True App จะบันทึกเวลาเริ่มเปิด App และตรวจสอบเงื่อนไขต่อว่า “App อยู่ในสถานะหยุด Stop State เกิน 5 นาที หรือเวลาการเปิด App น้อยกว่าเวลาที่ App อยู่ในสถานะหยุด Stop State หรือไม่” ถ้าหากใช่กรณีใดกรณีหนึ่ง App จะตรวจสอบประเภทการยืนยันตัวตน และแสดงหน้าการยืนยันตัวตน แต่ถ้าไม่ใช่ทั้ง 2 กรณี App จะตรวจสอบเงื่อนไขต่อว่า “ผู้ใช้เลือกใช้การยืนยันตัวตนเป็น Fingerprint แต่ไม่มีการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟนหรือไม่” (แม้ระยะเวลาการกลับเข้ามาใช้ App จะไม่เกิน 5

นาที จริง ๆ แล้ว App ไม่ต้องตรวจสอบอะไรอีก แต่ผู้วิจัยฯ ต้องการป้องกันกรณีผู้ใช้ลบลายนิ้วมือที่ไปลงทะเบียนก่อนกลับเข้าใช้ App ผู้วิจัยฯ จึงทำการตรวจสอบเงื่อนไขต่อเนื่องดังกล่าว) ถ้าไม่ใช่ App จะแสดงหน้า App Signal แต่ถ้าใช่ App จะแสดง Dialiog แจ้งเตือน หากผู้ใช้กด Confirm App จะแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือดังแสดงในรูปที่ 3-74



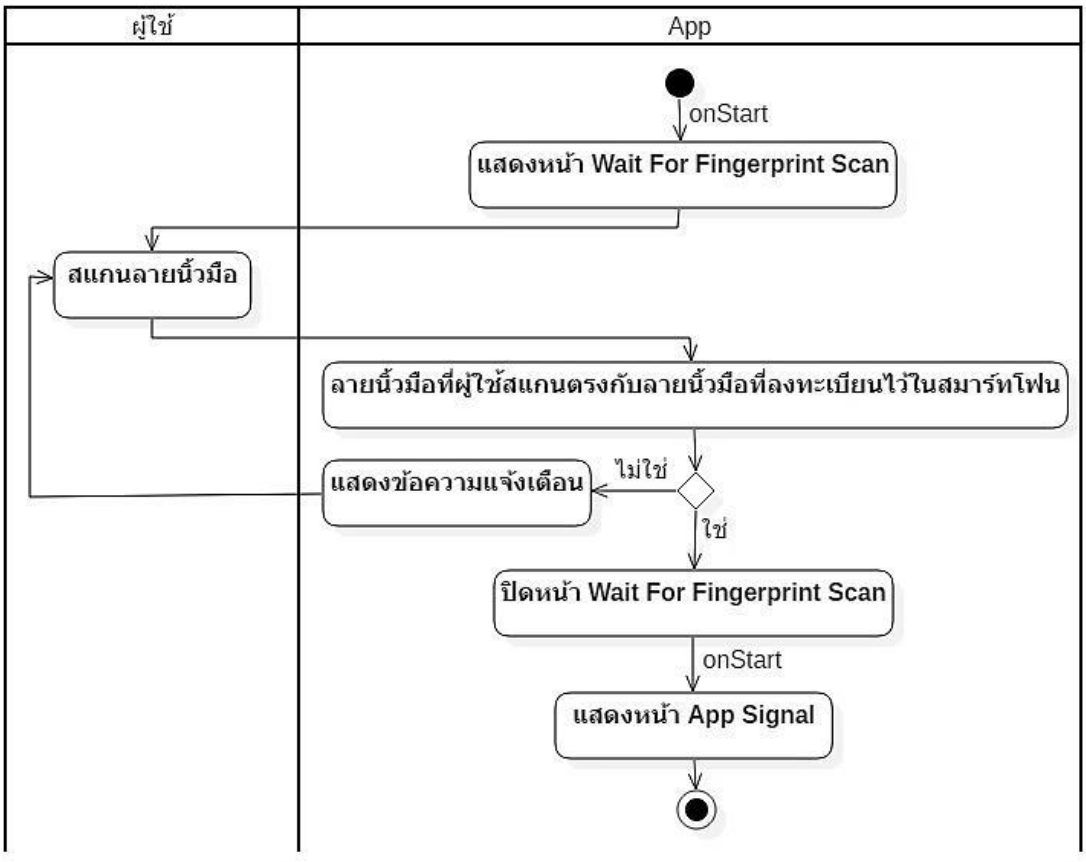
รูปที่ 3-74 แอททิวิตี้ไดอะแกรมตรวจสอบระยะเวลาการกลับเข้ามาใช้ App (ต่อจากรูปที่ 3-73)

- App ตรวจสอบเงื่อนไขว่าผู้ใช้ “เลือกใช้การยืนยันตัวตนแบบ Pattern หรือไม่” ถ้าใช่ App จะใช้การยืนยันตัวตนโดยการใส่ Pattern แต่ถ้าไม่ใช่ App จะตรวจสอบการลงทะเบียนลายนิ้วมือในสมาร์ทโฟน และยืนยันตัวตนโดยการสแกนลายนิ้วมือ ดังแสดงในรูปที่ 3-75



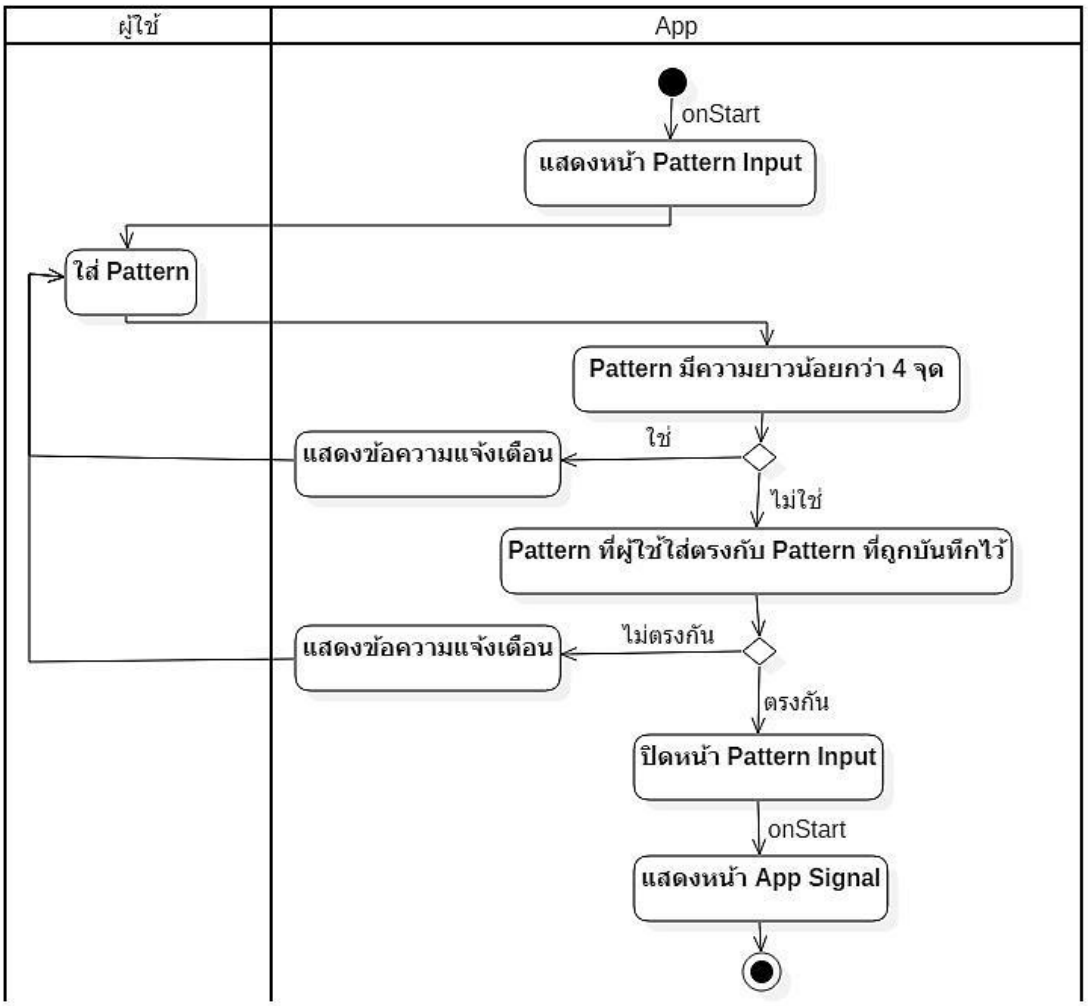
รูปที่ 3-75 แอททิวิตี้ไดอะแกรมตรวจสอบประเภทการยืนยันตัวตน และแสดงหน้าการยืนยันตัวตน (ต่อจากรูปที่ 3-73 หรือ 3-74)

- App แสดงหน้า Wait For Fingerprint Scan หลังจากผู้ใช้สแกนลายนิ้วมือ App จะตรวจสอบ “ลายนิ้วมือที่ผู้ใช้สแกนว่าตรงกับลายนิ้วมือที่ลงทะเบียนไว้ในสมาร์ทโฟนหรือไม่” ถ้าใช่ App จะปิดหน้า Wait For Fingerprint Scan ลง และแสดงหน้า App Signal แต่ถ้าไม่ใช่ App จะแสดงข้อความแจ้งเตือนและให้ผู้ใช้กลับไปสแกนลายนิ้วมือใหม่ ดังแสดงในรูปที่ 3-76



รูปที่ 3-76 แอททิวิตี้ไดอะแกรมยืนยันตัวตนโดยสแกนลายนิ้วมือ (ต่อจากรูปที่ 3-72, 3-79)

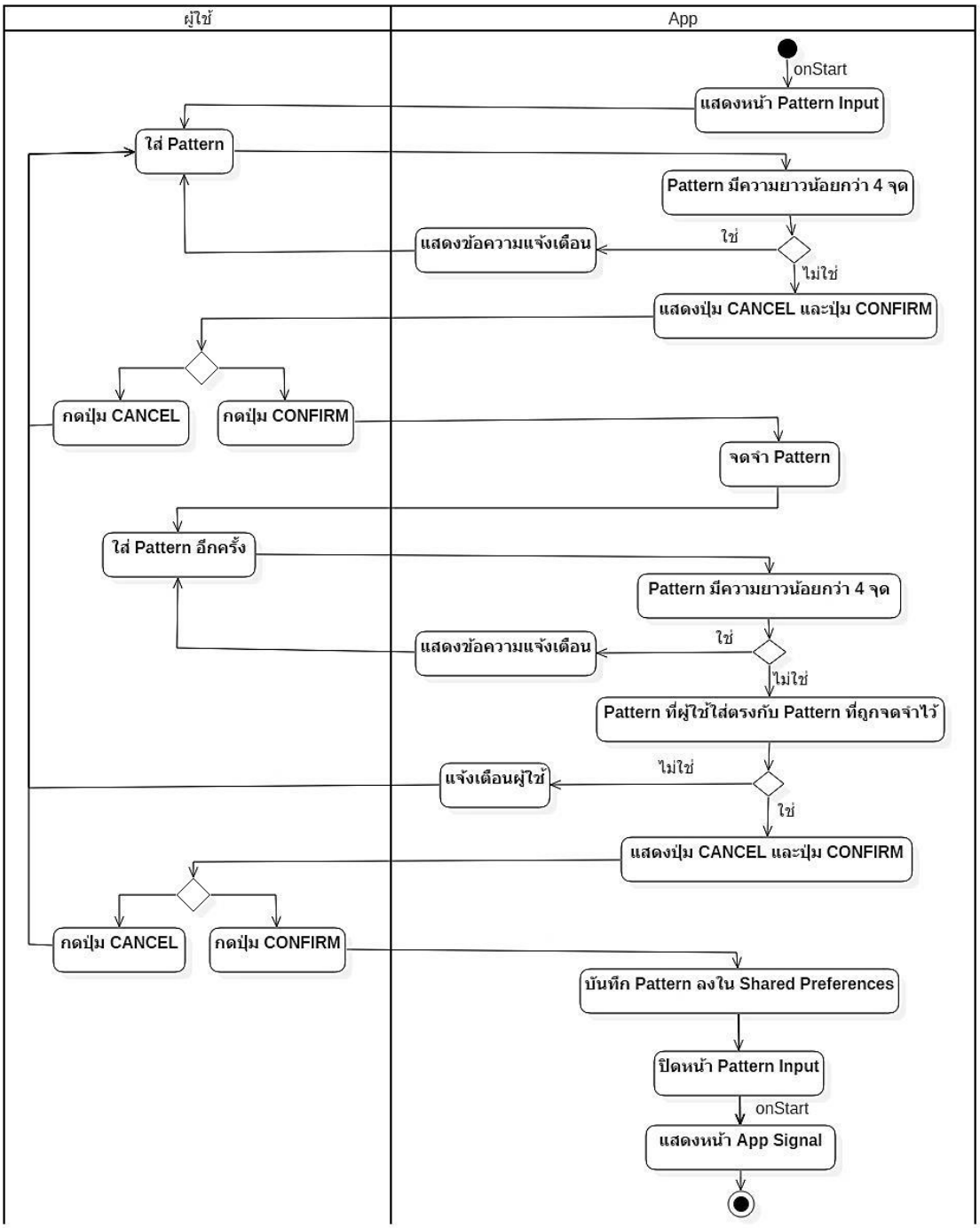
- App แสดงหน้า Pattern Input จากนั้นผู้ใช้ใส่ Pattern App จะตรวจสอบว่า “Pattern ที่ผู้ใช้ใส่ มีความยาวน้อยกว่า 4 จุด หรือไม่” ถ้าใช่ App จะแสดงข้อความแจ้งเตือน และให้ผู้ใช้กลับไปใส่ Pattern ใหม่ แต่ถ้าไม่ใช่ App จะตรวจสอบว่า “Pattern ที่ผู้ใช้ใส่ตรงกับ Pattern ที่ถูกบันทึกไว้หรือไม่ตรงกัน” ถ้าตรงกัน App จะปิดหน้า Pattern Input ลง และแสดงหน้า App Signal แต่ถ้าไม่ตรงกัน App จะแสดงข้อความแจ้งเตือน และให้ผู้ใช้กลับไปใส่ Pattern ใหม่อีกครั้ง ดังแสดงในรูปที่ 3-77



รูปที่ 3-77 แอททิวิตี้ไดอะแกรมยืนยันตัวตนโดยใส่ Pattern (มาจากรูปที่ 3-75, 3-82)

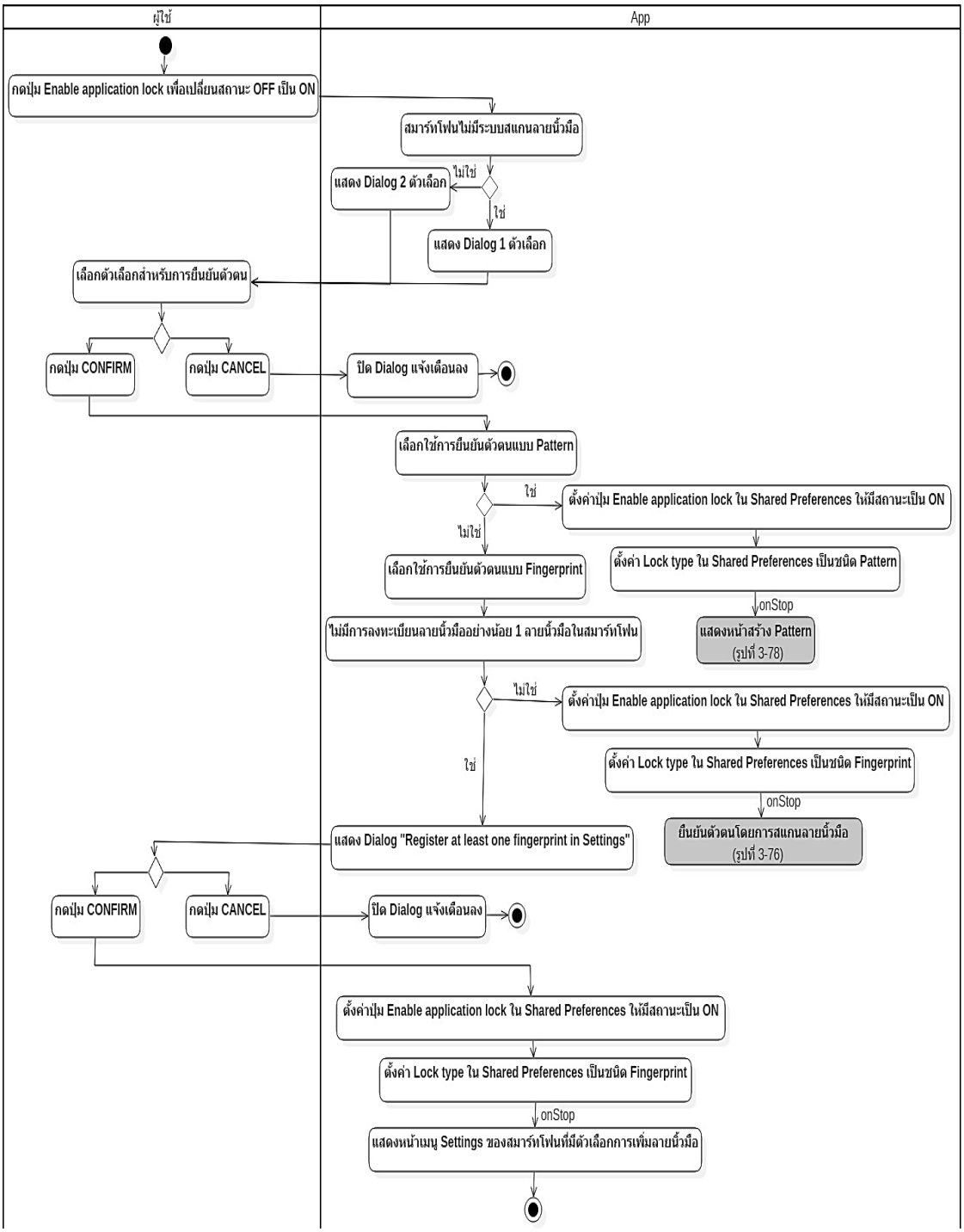
- App แสดงหน้า Pattern Input (สำหรับสร้าง Pattern) จากนั้นผู้ใช้ใส่ Pattern App จะตรวจสอบว่า “Pattern ที่ผู้ใช้ใส่มีความยาวน้อยกว่า 4 จุดหรือไม่” ถ้าใช่ App จะแสดงข้อความแจ้งเตือนและให้ผู้ใช้กลับไปใส่ Pattern ใหม่ แต่ถ้าไม่ใช่ App จะแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CANCEL และปุ่ม CONFIRM หากผู้ใช้กดปุ่ม CANCEL App จะยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และให้ผู้ใช้กลับไปใส่ Pattern ใหม่หรือหากผู้ใช้กดปุ่ม CONFIRM Pattern ที่ผู้ใช้ใส่จะถูกจดจำ และให้ผู้ใช้ใส่ Pattern อีกครั้ง จากนั้น App ก็จะตรวจสอบว่า “Pattern ที่ผู้ใช้ใส่มีความยาวน้อยกว่า 4 จุดหรือไม่” ถ้าใช่ App จะแสดงข้อความ

แจ้งเตือน และให้ผู้ใช้กลับไปใส่ Pattern ใหม่ แต่ถ้าไม่ใช่ App จะตรวจสอบว่า “Pattern ที่ผู้ใช้ใส่ตรงกับ Pattern ที่ถูกจดจำไว้หรือไม่” ถ้าไม่ใช่ App จะแจ้งเตือนผู้ใช้ และให้ผู้ใช้กลับไปใส่ Pattern ใหม่ตั้งแต่ขั้นตอนแรก แต่ถ้าใช่ App จะแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CANCEL และปุ่ม CONFIRM หากผู้ใช้กดปุ่ม CANCEL App จะยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และให้ผู้ใช้กลับไปใส่ Pattern ใหม่ตั้งแต่ขั้นตอนแรก หรือหากผู้ใช้กดปุ่ม CONFIRM Pattern ที่ผู้ใช้ใส่จะถูกบันทึกลงใน Shared Preferences รวมถึงปิดหน้า Pattern Input (สำหรับสร้าง Pattern) ลง และแสดงหน้า App Signal ดังแสดงในรูปที่ 3-78



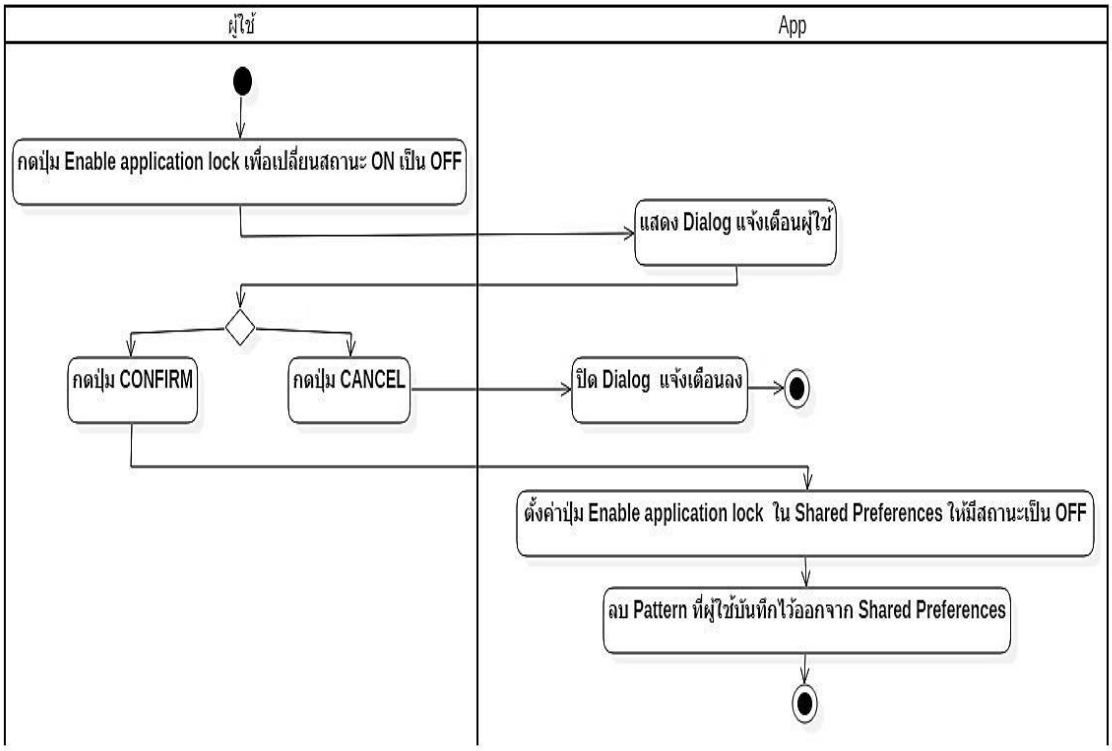
รูปที่ 3-78 แอททิวิตี้ไดอะแกรมแสดงหน้าสร้าง Pattern (ต่อจากรูปที่ 3-79)

- ผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ OFF เป็น ON จากนั้น App จะแสดง Dialog ว่า “Do you want to enable the application lock?” หากสมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ App จะแสดง 1 ตัวเลือก (Pattern lock) แต่ถ้าสมาร์ทโฟนมีระบบสแกนลายนิ้วมือ จะแสดง 2 ตัวเลือก (Pattern lock, Fingerprint lock) จากนั้นผู้ใช้ทำการเลือกประเภทการยืนยันตัวตน หากผู้ใช้กดปุ่ม CANCEL App จะปิด Dialog แจ้งเตือนลง หรือหากผู้ใช้กดปุ่ม CONFIRM App จะตรวจสอบว่าผู้ใช้ “เลือกใช้การยืนยันตัวตนแบบ Patternหรือไม่” ถ้าใช่ App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าไม่ถูกเช็คเครื่องหมายถูก (ชนิด Pattern) และแสดงหน้าสร้าง Pattern แต่ถ้าไม่ใช่แสดงว่าผู้ใช้ “เลือกใช้การยืนยันตัวตนแบบ Fingerprint” จากนั้น App จะตรวจสอบเงื่อนไขว่า “ไม่มีการลงทะเบียนลายนิ้วมืออย่างน้อง 1 ลายนิ้วมือ ในสมาร์ทโฟนใช่หรือไม่ใช่” ถ้าใช่ App จะแสดง Dialog แจ้งเตือน หากผู้ใช้กดปุ่ม CANCEL จะปิด Dialog แจ้งเตือนลง หรือหากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ แต่ถ้าไม่ใช่ มีการลงทะเบียนลายนิ้วมือแล้ว App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น ON รวมถึงตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และยืนยันตัวตนโดยใช้การสแกนลายนิ้วมือ ดังรูปที่ 3-79



รูปที่ 3-79 แอททิวิตี้ไดอะแกรมผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ OFF เป็น ON

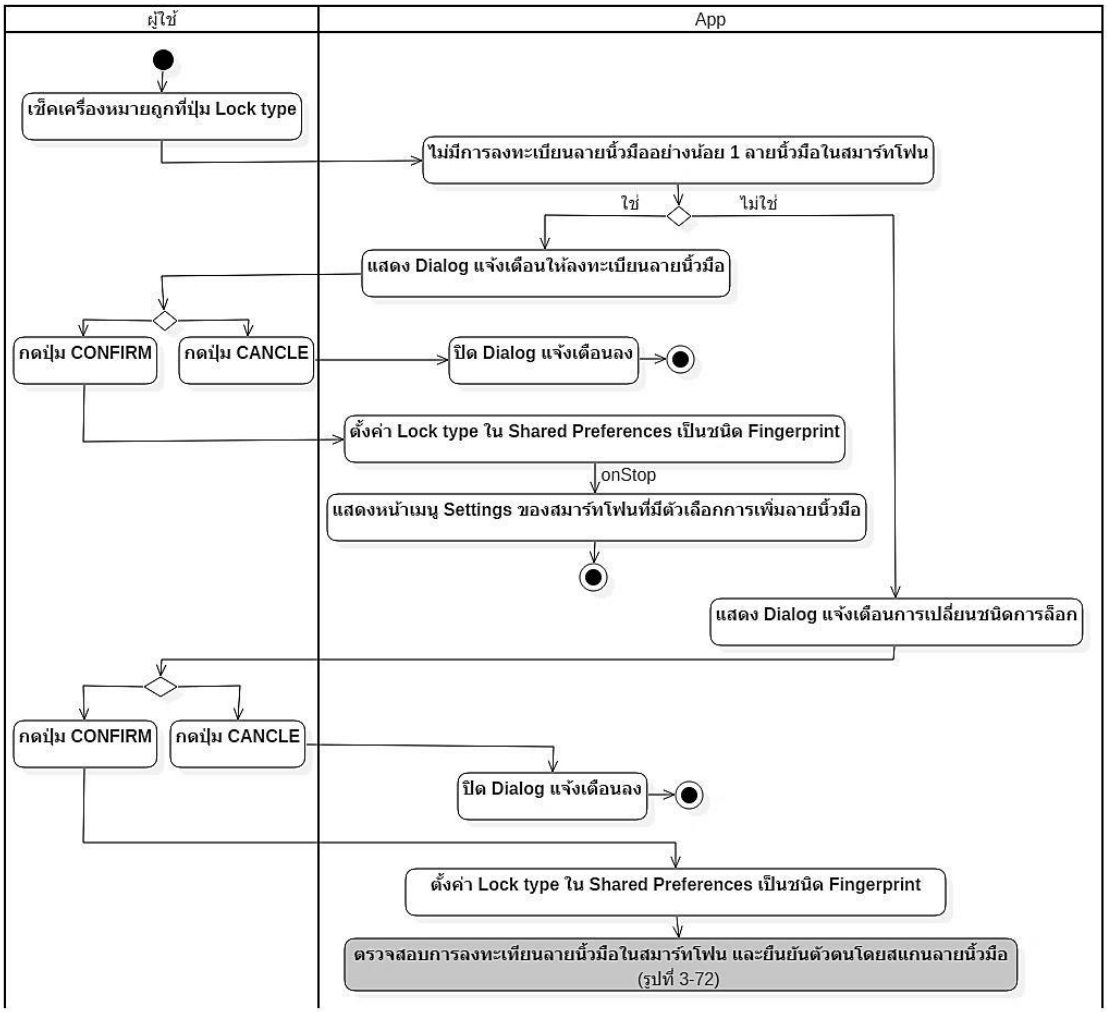
- ผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ ON เป็น OFF จากนั้น App จะแสดง Dialog แจ้งเตือนผู้ใช้ว่า “Are you sure you want to disable the application lock?” หากผู้ใช้กดปุ่ม CANCEL App จะปิด Dialog แจ้งเตือนลง แต่หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่าปุ่ม Enable application lock ใน Shared Preferences ให้มีสถานะเป็น OFF และลบ Pattern ที่ผู้ใช้ได้บันทึกไว้สำหรับการยืนยันตัวตนออกจาก Shared Preferences ดังแสดงในรูปที่ 3-80



รูปที่ 3-80 แอกทิวิตี้ไดอะแกรมผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ ON เป็น OFF

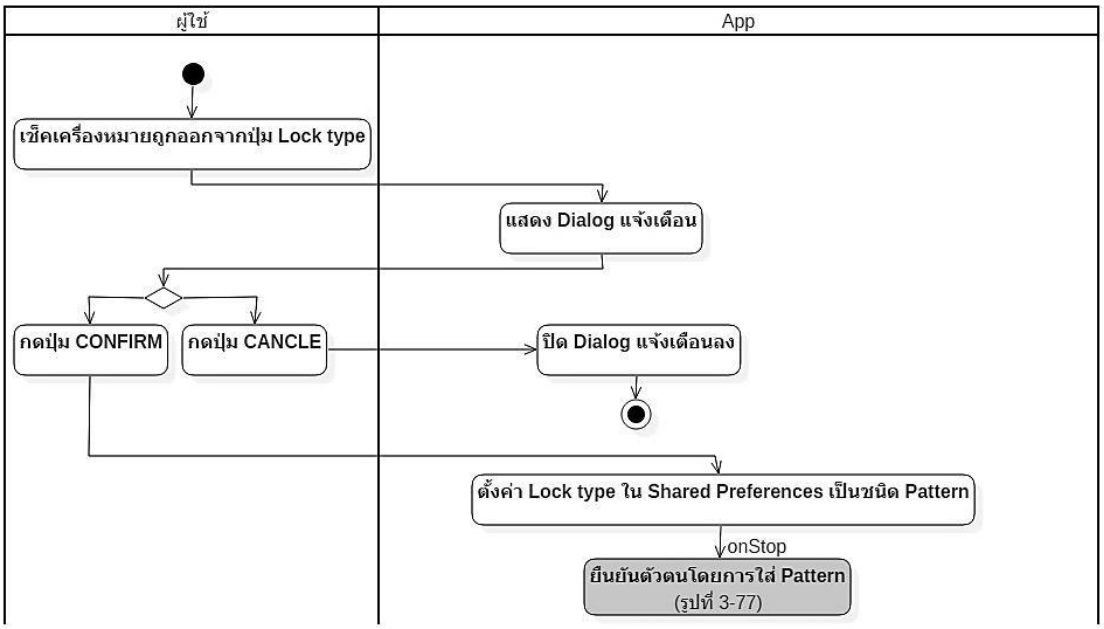
- ผู้ใช้เช็คเครื่องหมายถูกที่ปุ่ม Lock type จากนั้น App จะตรวจสอบเงื่อนไขว่า “ไม่มีการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือ ในสมาร์ทโฟนหรือไม่” ถ้าใช่ App จะแสดง Dialog แจ้งเตือนว่า “Register at least one fingerprint in Settings” หากผู้ใช้กดปุ่ม CANCEL จะปิด Dialog แจ้งเตือนลง หรือหากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และแสดงหน้าเมนู Setting ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่ม

ลายนิ้วมือ แต่ถ้ามีการลงทะเบียนลายนิ้วมือแล้ว App จะแสดง Dialog แจ้งเตือนว่า “Are you sure you want to change the application lock type.” พร้อมกับแสดงปุ่ม 2 ปุ่ม ได้แก่ปุ่ม CONFIRM และ ปุ่ม CANCEL หากผู้ใช้กดปุ่ม CANCEL App จะปิด Dialog แจ้งเตือนลง หรือหากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่า Lock type ใน Shared Preferences ว่าถูกเช็คเครื่องหมายถูก (ชนิด Fingerprint) และ App จะทำการตรวจสอบการลงทะเบียนลายนิ้วมือในสมาร์ทโฟน และยืนยันตัวตนโดยการสแกนลายนิ้วมือ ดังแสดงในรูปที่ 3-81



รูปที่ 3-81 แอททิวิตี้ไดอะแกรมผู้ใช้เช็คเครื่องหมายถูกที่ปุ่ม Lock type

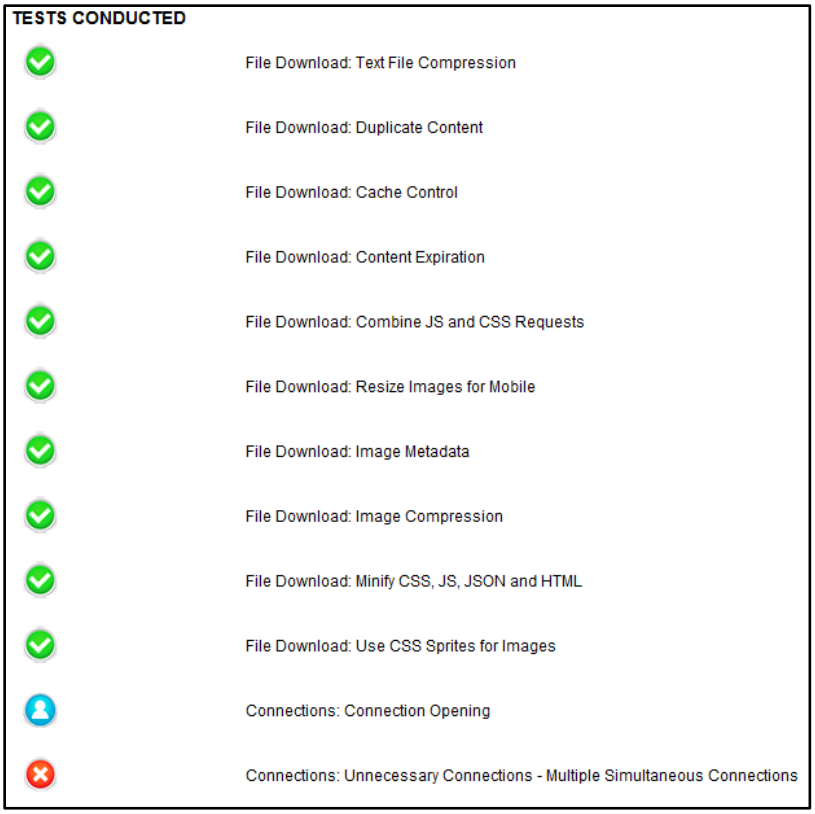
- ผู้ใช้เช็คเครื่องหมายถูกออกจากปุ่ม Lock type จากนั้น App จะแสดง Dialog แจ้งเตือนว่า “Are you sure you want to change the application lock type?” พร้อมกับแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CONFIRM และ ปุ่ม CANCEL หากผู้ใช้กดปุ่ม CANCEL App จะปิด Dialog แจ้งเตือนลง หรือหากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่า Lock type ใน Shared Preferences ว่าไม่ถูกเช็คเครื่องหมายถูก (ชนิด Pattern) และยืนยันตัวตนโดยการใส่ Pattern ดังแสดงในรูปที่ 3-82



รูปที่ 3-82 แอททิวิตี้ไดอะแกรมผู้ใช้เช็คเครื่องหมายถูกออกจากปุ่ม Lock type

3.7 ผลการทดสอบจาก ARO

เป็นการบันทึกวิเคราะห์ และสรุปผลการทดสอบจากการทดสอบจำนวน 3 ครั้งต่อเนื่องกัน โดยในส่วนของของอายุ Cache จะเริ่มนับอายุจากการทดลองครั้งที่ 3 เสร็จ หากมีครั้งใดที่ไม่ผ่าน การทดสอบนั้นจะถือว่าไม่ผ่านการทดสอบ และผลการทดสอบจาก ARO เป็นดังนี้



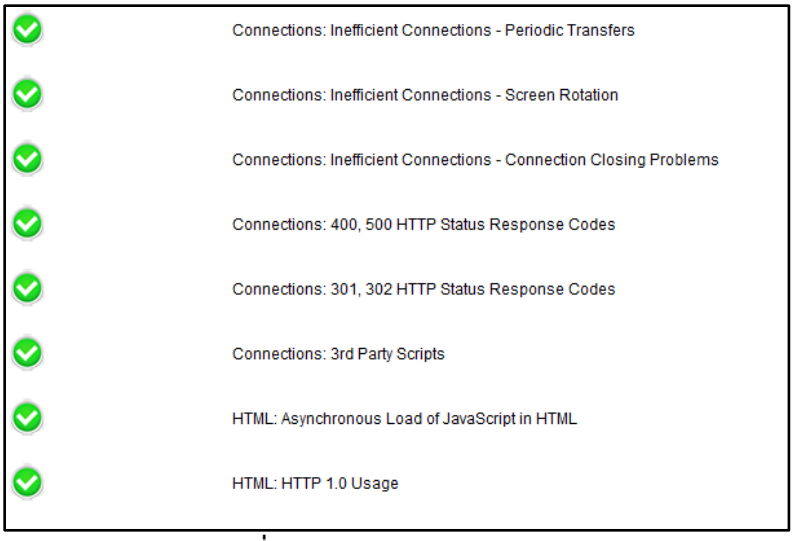
รูปที่ 3-83 ผลการทดสอบ

หมายถึง ผ่านการทดสอบ

หมายถึง ไม่ผ่านทดสอบ

หมายถึง ผ่านการทดสอบแต่มีคำแนะนำ

หมายถึง มีแจ้งเตือนแต่ยังผ่านการทดสอบ



รูปที่ 3-84 ผลการทดสอบ (ต่อ)

3.7.1 Text File Compression

การบีบอัด Text File ก่อนส่งเพื่อลดขนาดของ Text File

วิธีการทดสอบ : เนื่องจาก ผู้ใช้ไม่สามารถใช้ Signal ส่ง Text File ได้ เราจึงส่งข้อความที่มีขนาดใหญ่จากคู่สนทนา แม้ว่าในขั้นตอนนี้ ARO จะรายงานผลว่าผ่าน แต่ผู้วิจัยพบว่า Signal ไม่ได้บีบอัดข้อมูล Text ใด ๆ (ARO ตรวจจับการบีบอัดไม่ได้ เพราะ Signal เข้ารหัสข้อมูล)

ตารางที่ 3-2 Text File Compression

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ส่งสำเร็จ สามารเปิดอ่านได้	ไม่บีบอัดไฟล์ text
2	ส่งสำเร็จ สามารเปิดอ่านได้	ไม่บีบอัดไฟล์ text
3	ส่งสำเร็จ สามารเปิดอ่านได้	ไม่บีบอัดไฟล์ text
สรุป		**ไม่ผ่าน**

3.7.2 Duplicate Content

ไม่ Download เนื้อหาที่ซ้ำซ้อน (กรณีที่ Download มาแล้ว) หากไม่มีการเปลี่ยนแปลง

วิธีการทดสอบ : รับภาพมาหนึ่งภาพ เปิด 1 ครั้ง ปิดภาพออกจากแชท |
กลับเข้ามาแล้วเปิดอีกครั้ง

ตารางที่ 3-3 Duplicate Content

ครั้งที่	**ผลลัพธ์**	**รายละเอียด**
1	Signal App ไม่ Download ภาพซ้ำ เนื่องจาก Signal ได้เก็บภาพไว้ใน Cache (Database) (ทดลองโดย ผู้ทดสอบปิด Internet ขณะดูภาพเป็นครั้งที่ 2)	-
2	Signal App ไม่ Download ภาพซ้ำ เนื่องจาก Signal ได้เก็บภาพไว้ใน Cache (Database) (ทดลองโดย ผู้ทดสอบปิด Internet ขณะดูภาพเป็นครั้งที่ 2)	-
3	Signal App ไม่ Download ภาพซ้ำ เนื่องจาก Signal ได้เก็บภาพไว้ใน Cache (Database) (ทดลองโดย ผู้ทดสอบปิด Internet ขณะดูภาพเป็นครั้งที่ 2)	-
สรุป		**ผ่าน**

3.7.3 Cache Control

การจัดการด้าน Cache ว่าภายใน Signal มีการจัดการ Cache หรือไม่

วิธีการทดสอบ : ส่งข้อความและภาพไปมาหากัน

จากนั้นทำการปิด App แล้วเปิดใหม่ ดูว่า มี Cache อยู่หรือไม่

ตารางที่ 3-4 Cache Control

ครั้งที่	ผลลัพธ์	รายละเอียด
1	มีการเก็บข้อความและภาพต่าง ๆ ไว้ในเครื่อง เมื่อปิด App และเปิดขึ้นมาใหม่โดยไม่เชื่อต่อกับ Internet App สามารถแสดงข้อความและภาพได้	-
2	มีการเก็บข้อความและภาพต่าง ๆ ไว้ในเครื่อง เมื่อปิด App และเปิดขึ้นมาใหม่โดยไม่เชื่อต่อกับ Internet App สามารถแสดงข้อความและภาพได้	-
3	มีการเก็บข้อความและภาพต่าง ๆ ไว้ในเครื่อง เมื่อปิด App และเปิดขึ้นมาใหม่โดยไม่เชื่อต่อกับ Internet App สามารถแสดงข้อความและภาพได้	-
สรุป		ผ่าน

3.7.4 Cache Expiration

ตรวจสอบอายุของ Cache ว่ามีวันหมดอายุหรือไม่ และ Cache ควรจะมีวันหมดอายุ

วิธีการทดสอบ : ดูว่า message หรือรูปใน Cache มีอายุของ Cache หรือไม่ ในลักษณะคล้าย ๆ กับแอปพลิเคชัน Line

ตารางที่ 3-5 Cache Expiration

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีอายุของ Cache เนื่องจากเก็บข้อมูลต่าง ๆ ลงใน database ของเครื่องจนกว่าผู้ใช้จะลบเอง (หลังจาก ผู้ทดลองรอ 2 อาทิตย์พบว่า ข้อมูลดังกล่าวก็ยังอยู่ใน Database)	-
2	ไม่มีอายุของ Cache เนื่องจากเก็บข้อมูลต่าง ๆ ลงใน database ของเครื่องจนกว่าผู้ใช้จะลบเอง (หลังจาก ผู้ทดลองรอ 2 อาทิตย์พบว่า ข้อมูลดังกล่าวก็ยังอยู่ใน Database)	-
3	ไม่มีอายุของ Cache เนื่องจากเก็บข้อมูลต่าง ๆ ลงใน database ของเครื่องจนกว่าผู้ใช้จะลบเอง (หลังจาก ผู้ทดลองรอ 2 อาทิตย์พบว่า ข้อมูลดังกล่าวก็ยังอยู่ใน Database)	-
สรุป		**ไม่ผ่าน**

3.7.5 Content Pre-fetching

Signal ควรเรียกดูเนื้อหาถัดไปอัตโนมัติเมื่อผู้ใช้เลื่อนไปใกล้จะสิ้นสุดของเนื้อหาที่มีอยู่

วิธีการทดสอบ : เนื่องจาก App ไม่ได้ถูกออกแบบมาให้ Download ข้อความหรือภาพเก่าจากบัญชีผู้ใช้ในเครื่องเดิมสู้เครื่องใหม่ ดังนั้น การทดสอบโดยใช้ ARO จึงไม่จำเป็น อย่างไรก็ตาม ผู้วิจัยได้ทดลอง Scroll ในหน้าสนทนา ผู้ทำการทดลองจะสังเกตว่า App สามารถแสดงรูปหรือข้อความในอดีตที่ได้เคยถูก Download ไว้ในเครื่อง

3.7.6 Resize Images for Mobile

ตรวจสอบว่า Signal มีการทำ Resizing รูปภาพให้มีขนาดที่เหมาะสำหรับมือถือหรือไม่

วิธีการทดสอบ : ลองส่ง ภาพ / VDO ขนาดใหญ่ แล้วดูขนาดที่ฝั่ง Client ผู้รับ

ตารางที่ 3-6 Resize Images for Mobile

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ภาพขนาดเท่าเดิม เช่น ส่งด้วยขนาด 1440 X 1080 ปลายทางภาพที่ได้รับ ขนาดเดียวกัน ดังนั้นไม่มีการ resize	-
2	ภาพขนาดเท่าเดิม เช่น ส่งด้วยขนาด 1440 X 1080 ปลายทางภาพที่ได้รับ ขนาดเดียวกัน ดังนั้นไม่มีการ resize	-
3	ภาพขนาดเท่าเดิม เช่น ส่งด้วยขนาด 1440 X 1080 ปลายทางภาพที่ได้รับ ขนาดเดียวกัน ดังนั้นไม่มีการ resize	-
สรุป		**ไม่ผ่าน**

3.7.7 Image Compression

การบีบอัดภาพ Signal ควรจะมีการบีบอัดภาพก่อนส่ง

วิธีการทดสอบ : ทดลองส่งภาพหลาย ๆ ขนาด ตรวจสอบดูว่า มีการบีบอัดหรือไม่ หากมีการบีบอัด ตรวจสอบต่อว่า บีบอัดแล้วคุณภาพของภาพแย่ลงหรือไม่ (ไม่สามารถส่ง bmp ได้)

ตารางที่ 3-7 Image Compression

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ขนาดไฟล์ภาพเล็กลงเล็กน้อยเท่านั้น เนื่องจาก ตัด meta data ออกแต่ไม่พบการบีบอัดภาพ (สามารถส่งภาพ PNG กับ JPEG ได้เท่านั้น)	-
2	ขนาดไฟล์ภาพเล็กลงเล็กน้อยเท่านั้น เนื่องจาก ตัด meta data ออกแต่ไม่พบการบีบอัดภาพ (สามารถส่งภาพ PNG กับ JPEG ได้เท่านั้น)	-
3	ขนาดไฟล์ภาพเล็กลงเล็กน้อยเท่านั้น เนื่องจาก ตัด meta data ออกแต่ไม่พบการบีบอัดภาพ (สามารถส่งภาพ PNG กับ JPEG ได้เท่านั้น)	-
สรุป		**ไม่ผ่าน**

3.7.8 Image Metadata

Signal ควรจะต้องมีการตัด Metadata ที่ไม่จำเป็นออกไปก่อนทำการส่ง

วิธีการทดสอบ : ส่งภาพเพื่อตรวจสอบว่า Metadata ถูกตัดออกก่อนส่งหรือไม่

ตารางที่ 3-8 Image Metadata

ครั้งที่	ผลลัพธ์	รายละเอียด
1	มีการตัด metadata ก่อนส่ง เช่น ชื่อ เวลาที่ถ่ายภาพ สถานที่ถ่ายภาพ	-
2	มีการตัด metadata ก่อนส่ง เช่น ชื่อ เวลาที่ถ่ายภาพ สถานที่ถ่ายภาพ	-
3	มีการตัด metadata ก่อนส่ง เช่น ชื่อ เวลาที่ถ่ายภาพ สถานที่ถ่ายภาพ	-
สรุป		**ผ่าน**

3.7.9 Opening Connections

การเปิด Connection ไม่ควรมีมากเกินไป Signal ควรจะจัดกลุ่มข้อมูลก่อนแล้วจึงส่งข้อมูลไปใน Connection เดียว (ในหัวข้อ 3.7.10)

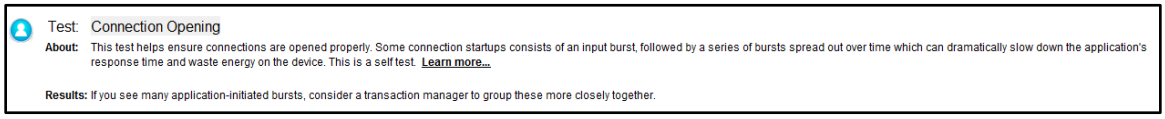
วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-9 Opening Connections

ครั้งที่	ผลลัพธ์	รายละเอียด
1	มีเพียงคำแนะนำ ไม่พบการแจ้งเตือนสำหรับการแก้ไข	-
2	มีเพียงคำแนะนำ ไม่พบการแจ้งเตือนสำหรับการแก้ไข	-
3	มีเพียงคำแนะนำ ไม่พบการแจ้งเตือนสำหรับการแก้ไข	-
สรุป		**ผ่าน(พร้อมคำแนะนำ)**

คำแนะนำ : ถ้าหากพบว่า App เปิด Connection พร้อม ๆ กันจำนวนมาก ให้พิจารณาที่ทำการรวม Connection เข้าไว้ด้วยกัน

If you see many application-initiated bursts, consider a transaction manager to group these more closely together.



รูปที่ 3-85 แนะนำ Opening Connections

3.7.10 Multiple Simultaneous TCP Connections

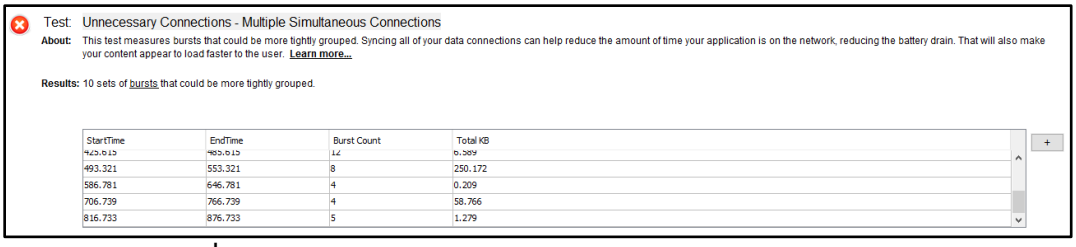
การเปิด Connection พร้อมกันมากเกินความจำเป็น Signal ควรจัดกลุ่มก่อนแล้วส่งไปใน Connection เดียว

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO ในขณะที่เปิด Chat กลุ่ม คุยกันหลาย ๆ คน load รูปพร้อม ๆ กันหลาย ๆ รูป

ตารางที่ 3-10 Multiple Simultaneous TCP Connections

ครั้งที่	ผลลัพธ์	รายละเอียด
1	พบการแจ้งเตือนของ IP Signal 1 ครั้ง	มีการส่ง packet 19 packet ไปที่ปลายทางแห่งเดียว โดย Signal ไม่ได้ส่งข้อความ และ รูปภาพต่าง ๆ ไปพร้อมกัน (เว้นระยะการส่ง)
2	พบการแจ้งเตือนของ IP Signal 2 ครั้ง	มีการส่ง packet 19 packet ไป แต่เป็นการส่งข้อความ รูปภาพไปที่ปลายทางหลายแห่ง จึงทำให้ทาง app ไม่สามารถตัดสินใจส่งรวมไปทีเดียวได้
3	ไม่พบการแจ้งเตือนของ app จาก ARO	เนื่องจาก network มีปัญหาขณะส่งข้อมูล ทำให้มีการรวบข้อมูลส่งไปด้วยกันทีเดียวจึงทำให้ไม่เกิดการแจ้งเตือน
สรุป		**ไม่ผ่าน (เนื่องจาก ครั้งที่ 1 และ 2)**

คำอธิบาย: ในกรณีที่สัญญาณการเชื่อมต่อดี Signal พยายามส่งข้อความทันที โดยไม่ Bundle แต่ จากการทดลองครั้งที่ 3 พบว่า สัญญาณการเชื่อมต่อมีปัญหา Signal จะ Bundle ข้อความแล้วส่งทีเดียว



รูปที่ 3-86 Multiple Simultaneous TCP Connections ไม่ผ่าน

3.7.11 Periodic Transfers

ตรวจสอบว่า Signal มีการเชื่อมต่อเพื่อส่ง Log ต่าง ๆ หรือ Ad ที่เป็นโฆษณาเป็นระยะ ๆ หรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-11 Periodic Transfers

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่พบการแจ้งเตือนใด ๆ จาก ARO	ARO ไม่พบการแจ้งเดือน ใด ๆ แสดงว่าไม่มีการเก็บ log หรือ ad ต่าง ๆ (ที่ต้องส่งข้อมูลเป็นระยะ ๆ)
2	ไม่พบการแจ้งเตือนใด ๆ จาก ARO	ARO ไม่พบการแจ้งเดือน ใด ๆ แสดงว่าไม่มีการเก็บ log หรือ ad ต่าง ๆ (ที่ต้องส่งข้อมูลเป็นระยะ ๆ)
3	ไม่พบการแจ้งเตือนใด ๆ จาก ARO	ARO ไม่พบการแจ้งเดือน ใด ๆ แสดงว่าไม่มีการเก็บ log หรือ ad ต่าง ๆ (ที่ต้องส่งข้อมูลเป็นระยะ ๆ)
สรุป		**ผ่าน**

3.7.12 Screen Rotations

การหมุนหน้าจอของ Signal หามีเนื้อหาเดิม ไม่ควรจะมีการเชื่อมต่อกับ Server เพื่อรับข้อมูลอีก

วิธีการทดสอบ : ลอง Rotate Screen ดู แล้วดูว่า มีการ Load เพิ่มหรือไม่

ตารางที่ 3-12 Screen Rotations

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการโหลดข้อมูลใหม่ และไม่มีการ open connection ใด ๆ	-
2	ไม่มีการโหลดข้อมูลใหม่ และไม่มีการ open connection ใด ๆ	-
3	ไม่มีการโหลดข้อมูลใหม่ และไม่มีการ open connection ใด ๆ	-
สรุป		**ผ่าน**

3.7.13 Closing Connections

ตรวจสอบว่า Signal ปิด Connection ของ Signal ว่าเมื่อไม่ใช้งานแล้วหรือไม่

วิธีการทดสอบ : ตรวจสอบดูว่า มี Connection เปิดค้างหรือไม่

ตารางที่ 3-13 Closing Connections

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่พบการเปิด connection ทิ้งไว้	ผ่านการตรวจสอบ ARO ไม่พบการเปิด connection ใด ๆ ค้างไว้
2	ไม่พบการเปิด connection ทิ้งไว้	ผ่านการตรวจสอบ ARO ไม่พบการเปิด connection ใด ๆ ค้างไว้
3	ไม่พบการเปิด connection ทิ้งไว้	ผ่านการตรวจสอบ ARO ไม่พบการเปิด connection ใด ๆ ค้างไว้
สรุป		**ผ่าน**

3.7.14 Offloading to Wi-Fi

ตรวจสอบว่า Signal สามารถสลับไปมาระหว่าง Wi-Fi กับ 3G ได้หรือไม่ หากมีทั้ง 3G/4G และ Wi-Fi Signal ควรเปลี่ยนไปใช้ Wi-FI

วิธีการทดสอบ : ทดลองใช้ 3G ก่อน จากนั้น ลองเปิด Wi-Fi ตรวจสอบว่า app ทำการ switch มาใช้ Wi-Fi หรือไม่

ตารางที่ 3-14 Offloading to Wi-Fi

ครั้งที่	ผลลัพธ์	รายละเอียด
1	app ทำการ switch จาก 3G/LTE มาเป็น Wi-Fi	-
2	app ทำการ switch จาก 3G/LTE มาเป็น Wi-Fi	-
3	app ทำการ switch จาก 3G/LTE มาเป็น Wi-Fi	-
สรุป		ผ่าน

3.7.15 HTTP 400 and 500 Status Codes

ตรวจสอบว่า Signal มีการเรียกใช้บริการ Web ที่ Return Code HTTP 400 หรือ 500 หรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-15 HTTP 400 and 500 Status Codes

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 400 - 500
2	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 400 - 500
3	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 400 - 500
สรุป		**ผ่าน**

3.7.16 HTTP 300 Status Codes

ตรวจสอบว่า Signal มีการเรียกใช้บริการ Web ที่ Return Code HTTP 300 หรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-16 HTTP 300 Status Codes

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 300
2	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 300
3	ไม่มีการตรวจพบ	ARO ไม่พบการเชื่อต่อใด ๆ ที่ทำให้เกิด code 300
สรุป		**ผ่าน**

3.7.17 HTTP 1.0 Usage

ตรวจสอบว่า Signal มีการเรียกใช้ API หรือการเชื่อมต่อไปยัง HTTP 1.0 หรือไม่ ซึ่งไม่ควรมี

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-17 HTTP 1.0 Usage

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ตรวจไม่พบการส่งข้อมูลแบบ HTTP 1.0
2	ไม่มีการตรวจพบ	ARO ตรวจไม่พบการส่งข้อมูลแบบ HTTP 1.0
3	ไม่มีการตรวจพบ	ARO ตรวจไม่พบการส่งข้อมูลแบบ HTTP 1.0
สรุป		**ผ่าน**

3.7.18 Unsecure SSL Version

ตรวจสอบว่า Signal มีการใช้ SSL Version ที่ถูกระบุว่าไม่ปลอดภัยหรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-18 Unsecure SSL Version

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ Unsecure SSL Version
2	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ Unsecure SSL Version
3	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ Unsecure SSL Version
สรุป		ผ่าน

3.7.19 Weak Cipher

ตรวจสอบว่า Signal มีการเข้ารหัส SSL ที่มีความปลอดภัยต่ำ ถูกถอดรหัสได้ง่ายหรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-19 Weak Cipher

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ weak cipher
2	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ weak cipher
3	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ weak cipher
สรุป		ผ่าน

3.7.20 Forward Secrecy

ตรวจสอบความปลอดภัยของ Signal ว่าป้องกันเรื่อง Forward Secrecy ซึ่งหาก Signal มีการป้องกันแล้ว เหตุการณ์การถูกถอดรหัส Session หนึ่งได้แล้วจะทำให้เข้าไปในอีก Session ก่อนหน้านั้นทั้งหมดได้ จะไม่เกิดขึ้น

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-20 Forward Secrecy

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ support forward
2	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ support forward
3	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ support forward
สรุป		ผ่าน

3.7.21 HTTP Vs. HTTPS

ตรวจสอบว่า API ต่าง ๆ ของ Signal ใช้ HTTPS ทั้งหมดหรือไม่

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-21 HTTP Vs. HTTPS

ครั้งที่	ผลลัพธ์	รายละเอียด
1	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ HTTP
2	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ HTTP
3	ไม่มีการตรวจพบ	ARO ตรวจไม่พบ HTTP
สรุป		**ผ่าน**

3.7.22 Private Data

ตรวจสอบว่า Signal ได้ขอข้อมูลส่วนตัวอะไรบ้างจากผู้ใช้ และมีการป้องกันข้อมูลเหล่านั้นอย่างไรเมื่อมีการส่งไปยัง Server

วิธีการทดสอบ : นอกจากการใช้ ARO ทดสอบผู้วิจัยได้ตรวจสอบจาก manifest ของ android และได้สังเกตขั้นตอน Register Signal เข้ารหัสข้อมูลก่อนส่งหรือไม่

ตารางที่ 3-22 Private Data

ครั้งที่	ผลลัพธ์	รายละเอียด
1	มีการขอ data ต่าง ๆ ดังนี้ profile, เบอร์โทรศัพท์, การเข้าถึง log, การขอใช้พื้นที่ภายในเครื่อง, การขอที่อยู่, การอ่าน SMS, การสมัครมีการส่งข้อมูล เบอร์โทรเป็นแบบเปิดเผย	-
2	มีการขอ data ต่าง ๆ ดังนี้ profile, เบอร์โทรศัพท์, การเข้าถึง log, การขอใช้พื้นที่ภายในเครื่อง, การขอที่อยู่, การอ่าน SMS, การสมัครมีการส่งข้อมูล เบอร์โทรเป็นแบบเปิดเผย	-
3	มีการขอ data ต่าง ๆ ดังนี้ profile, เบอร์โทรศัพท์, การเข้าถึง log, การขอใช้พื้นที่ภายในเครื่อง, การขอที่อยู่, การอ่าน SMS, การสมัครมีการส่งข้อมูล เบอร์โทรเป็นแบบเปิดเผย	-
สรุป		**ไม่ผ่าน**

คำอธิบาย : ผลจากการตรวจสอบไฟล์ AndroidManifest ดูได้จาก รูปที่ 3-87

ผลจากการสังเกตขั้นตอน Registation จาก Log ของ Server ดูได้จากรูปที่ 3-88

```
<uses-permission android:name="org.thoughtcrime.securesms.ACCESS_SECRETS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.WRITE_PROFILE"/>
<uses-permission android:name="android.permission.BROADCAST_WAP_PUSH"
                tools:ignore="ProtectedPermissions"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

รูปที่ 3-87 จากไฟล์ AndoridManifest

```
"GET /v1/accounts/sms/code/+66000000000 HTTP/1.1" 200 - "-" "okhttp/3.6.0" 63
"GET /v1/accounts/voice/code/+66000000000 HTTP/1.1" 200 - "-" "okhttp/3.6.0" 18
"PUT /v1/accounts/code/100000 HTTP/1.1" 204 - "-" "okhttp/3.6.0" 127
"PUT /v2/keys/ HTTP/1.1" 204 - "-" "okhttp/3.6.0" 206
"PUT /v1/directory/tokens HTTP/1.1" 200 15 "-" "okhttp/3.6.0" 29
"PUT /v1/directory/tokens HTTP/1.1" 200 15 "-" "okhttp/3.6.0" 7
"GET /v1/websocket/?login=+66000000000&password=dd89Xn/AlaaN2vq4TSEyJiEa HTTP/1.1"
```

รูปที่ 3-88 Log จาก Server ที่แสดงให้เห็นว่าหมายเลขโทรศัพท์ไม่มีการเข้ารหัส

3.7.23 Accessing Peripherals

ตรวจสอบการขอใช้ Hardware ต่าง ๆ ของ Signal ว่าเมื่อมีการใช้งาน หลังจากเสร็จสิ้นการใช้งาน Signal ทำการปิดอุปกรณ์เหล่านั้นให้ด้วยหรือไม่

วิธีการทดสอบ : ตรวจใน manifest ว่าควรจะปิด GPS, Bluetooth ฯลฯ เมื่อไม่ใช้งานแล้ว

ตารางที่ 3-23 Accessing Peripherals

ครั้งที่	ผลลัพธ์	รายละเอียด
1	มีการขอใช้ GPS Bluetooth แต่ทั้งนี้ทั้งนั้น ทาง signal จะไม่เปิดใช้เอง หากผู้ใช้ประสงค์จะใช้งาน signal จึงจะขออนุญาต ในการเปิดอีกครั้ง และหากจะปิดผู้ใช้ต้องปิดที่ android settings ด้วยตัวเอง เนื่องจาก signal โดยการทำงานให้ android	-
2	มีการขอใช้ GPS Bluetooth แต่ทั้งนี้ทั้งนั้น ทาง signal จะไม่เปิดใช้เอง หากผู้ใช้ประสงค์จะใช้งาน signal จึงจะขออนุญาต ในการเปิดอีกครั้ง และหากจะปิดผู้ใช้ต้องปิดที่ android settings ด้วยตัวเอง เนื่องจาก signal โดยการทำงานให้ android	-
3	มีการขอใช้ GPS Bluetooth แต่ทั้งนี้ทั้งนั้น ทาง signal จะไม่เปิดใช้เอง หากผู้ใช้ประสงค์จะใช้งาน signal จึงจะขออนุญาต ในการเปิดอีกครั้ง และหากจะปิดผู้ใช้ต้องปิดที่ android settings ด้วยตัวเอง เนื่องจาก signal โดยการทำงานให้ android	-
สรุป		**ไม่ผ่าน**

สรุป : ไม่ผ่านเนื่องจาก Signal ไม่ปิด GPS ให้อัตโนมัติ ขณะไม่ใช้งาน

3.7.24 Comparing LTE and 3G Energy Consumption

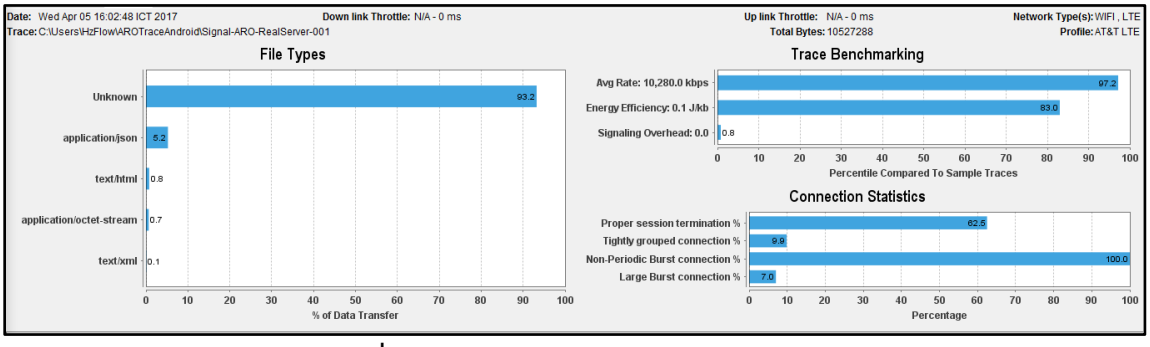
เปรียบเทียบการใช้พลังงาน ต่าง ๆ ระหว่าง LTE และ 3G โดยจะสรุปผลต่าง ๆ ออกมาให้ผู้ตรวจสอบเปรียบเทียบ ว่า Signal เหมาะสมกับการใช้งานแบบใดมากกว่า เพื่อวิเคราะห์ประสิทธิภาพการส่งข้อมูล และรวมไปถึงการใช้พลังงานที่ลดลง

วิธีการทดสอบ : วิเคราะห์จาก Trace results ใน ARO

ตารางที่ 3-24 Comparing LTE and 3G Energy Consumption

ครั้งที่	ผลลัพธ์	รายละเอียด
1	-	ARO ไม่รายงานข้อมูลในส่วนของการใช้ 3G จึงไม่สามารถสรุปได้
2	-	ARO ไม่รายงานข้อมูลในส่วนของการใช้ 3G จึงไม่สามารถสรุปได้
3	-	ARO ไม่รายงานข้อมูลในส่วนของการใช้ 3G จึงไม่สามารถสรุปได้
สรุป		**ไม่สามารถสรุปได้**

จากข้อมูลที่ได้ (รูปที่ 3-89) จะเห็นจากด้านขวาบนว่า Network เป็น WIFI,LTE และไม่สามารถสร้างการเปรียบเทียบจาก 3G ได้ (แม้ขณะทดสอบจะใช้ 3G)



รูปที่ 3-89 การเปรียบเทียบการใช้พลังงาน

File Types : ชนิดของไฟล์ที่ ARO มีการตรวจจับได้ โดยจะคิดเป็น % จากขนาดของไฟล์ที่มีการรับส่ง

- Unknown เป็นไฟล์ภาพที่ Signal มีการเข้ารหัสทำให้ ARO ไม่สามารถระบุ Type มีค่า 93.2 %

- Application/json เป็น Type JSON ที่ Signal ใช้ในการติดต่อระหว่าง App กับ Server มีค่า 5.2 %

Trace Benchmarking : ภาพรวมของเปรียบเทียบ คิดเป็น % โดยเทียบจาก Sample Trace

- Avg Rate คือค่าเฉลี่ยของขนาดข้อมูลที่รับส่งในขณะทดสอบ มีค่า 10,280 kbps| (kb ต่อ วินาที)

- Energy Efficiency คือการใช้พลังงานในขณะทดสอบ มีค่า 0.1 J/kb (joule ต่อ kb)

- Signaling Overhead คือ ต้นทุนหรือปริมาณ (Overhead) ข้อมูลที่ถูกส่งสำหรับการทำ Signaling

Connection Statistics : ภาพรวมสถิติของ Connection

- Proper session termination (การจบ Session ทีเหมาะสม) มีค่า 62.5 %

- Tightly grouped connection (การเชื่อมต่ออย่างหนาแน่) มีค่า 9.9 %

- Non-Periodic Burst connection (การเชื่อมต่อที่ลักษณะการส่งข้อมูลเป็นแบบ Burst แต่ไม่ได้ส่งเป็นระยะ ๆ) มีค่า 100 %

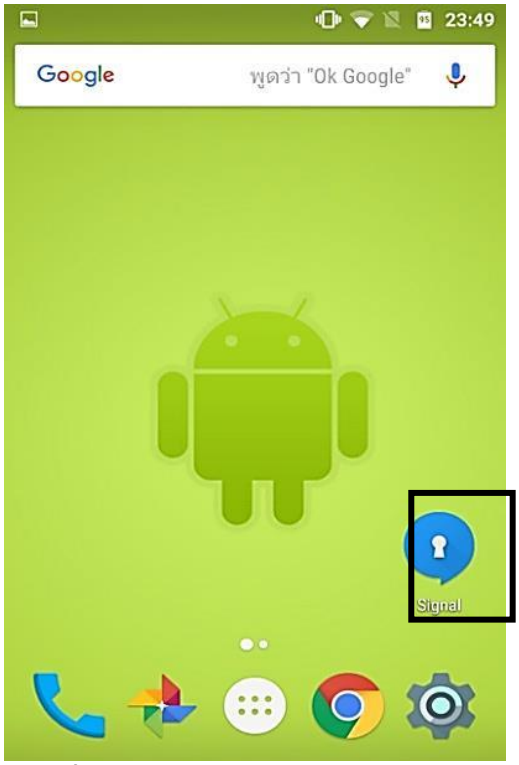
- Large Burst connection (การเชื่อมต่อที่มีขนาดใหญ่) มีค่า 7.0 %

3.8 ผลการพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal)

ผลการพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) สามารถแบ่งเป็นเหตุการณ์ต่าง ๆ และกรณีย่อย ๆ ได้ดังนี้

3.8.1 เปิดใช้งาน App ครั้งแรกหลังจากติดตั้ง App

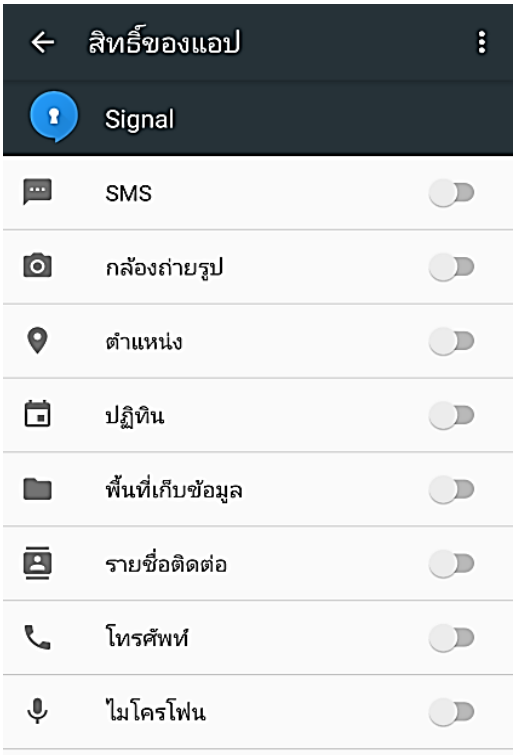
- ผู้ใช้กดเปิด App Signal จาก Icon หน้า Home ดังแสดงในรูปที่ 3-89



รูปที่ 3-19 หน้า Home ของสมาร์ทโฟน

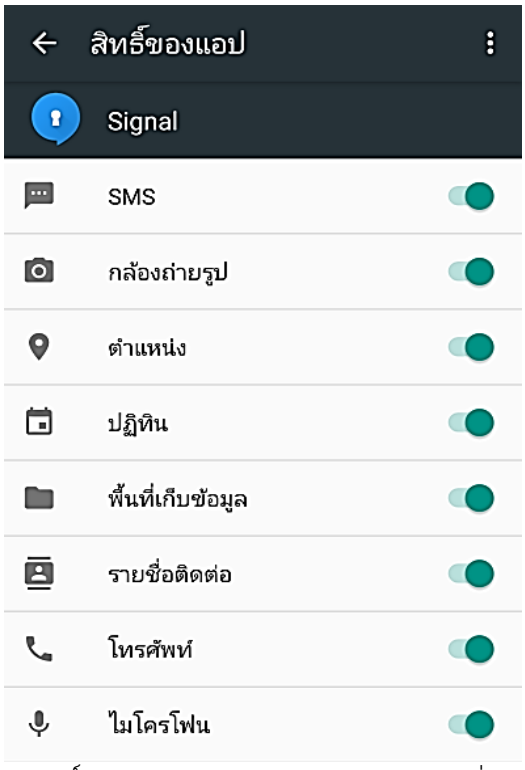
- สมาร์ทโฟนแสดง Dialog แจ้งเตือน ให้ผู้ใช้กดปุ่มตกลงเพื่อไปหน้าเพิ่มสิทธิ์การอนุญาตของ App Signal ดังแสดงในรูปที่ 3-90, 3-91

รูปที่ 3-20 Dialog แจ้งเตือนผู้ใช้หลังกดเปิด App Signal



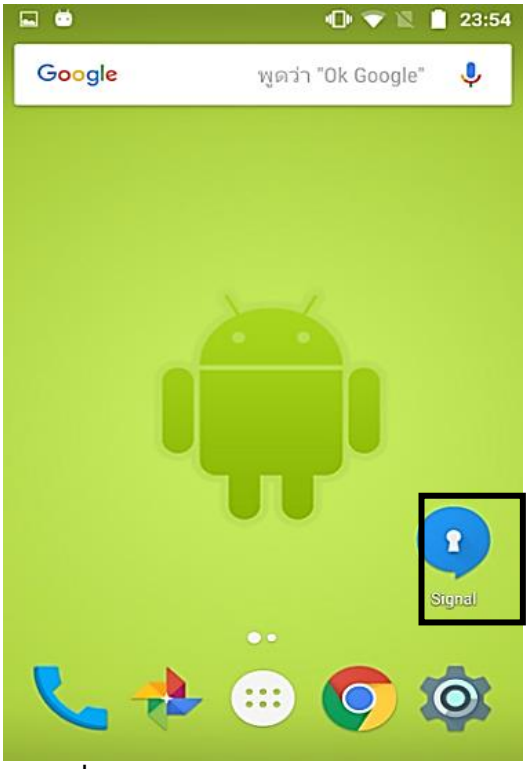
รูปที่ 3-91 หน้าสิทธิ์การอนุญาตของแอปพลิเคชัน Signal

- ให้ผู้ใช้ทำการเพิ่มสิทธิ์การอนุญาตให้ App Signal ดังแสดงในรูปที่ 3–92



รูปที่ 3-92 หน้าสิทธิ์การอนุญาตของ App Signal หลังเพิ่มสิทธิ์การอนุญาต

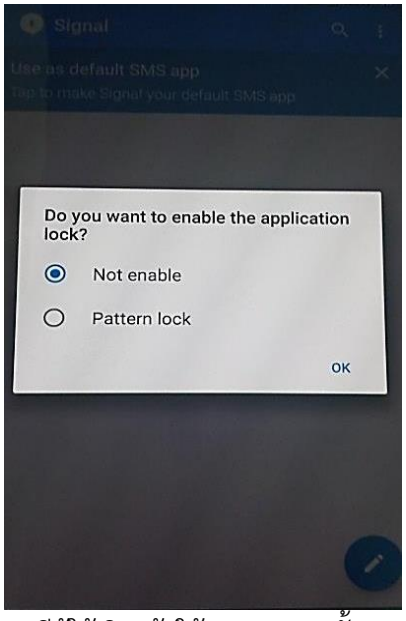
- ผู้ใช้กดเปิด App Signal จาก Icon หน้า Home อีกครั้ง ดังแสดงในรูปที่ 3–93



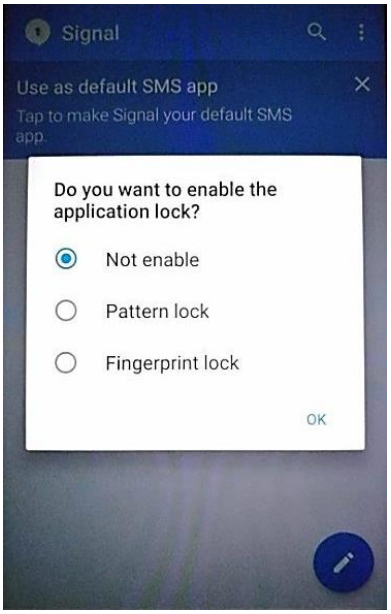
รูปที่ 3-93 หน้า Home ของสมาร์ทโฟน

- กรณีเปิดเข้าใช้งาน App ครั้งแรก หลังจากติดตั้ง App App จะแสดง Dialog แจ้งเตือนให้ผู้ใช้เลือกประเภทการยืนยันตัวตน ดังแสดงในรูปที่ 3-94 หรือ 3-95

- สำหรับสมาร์ทโฟนที่ไม่มีระบบสแกนลายนิ้วมือ App จะแสดง Dialog ให้ผู้ใช้เลือกประเภทการยืนยันตัวตนก่อนการเข้าใช้งาน App เพียง 2 ตัวเลือก ดังแสดงในรูปที่ 3-94

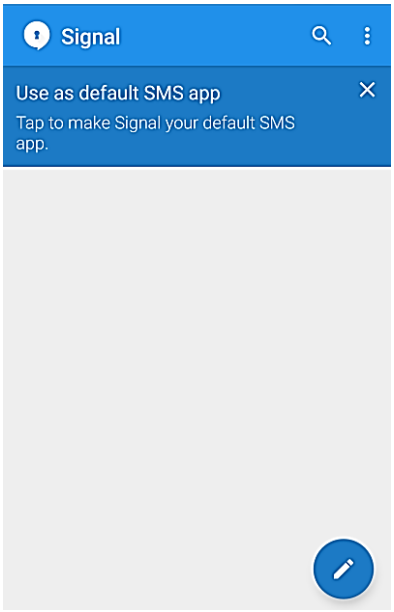


รูปที่ 3-94 Dialog แจ้งเตือน (กรณีผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App แต่สมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ)



รูปที่ 3-95 Dialog แจ้งเตือน (กรณีผู้ใช้เปิดเข้าใช้งาน App ครั้งแรกหลังจากติดตั้ง App)

ก. หากผู้ใช้เลือกตัวเลือก Not enable และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก, บันทึกตัวเลือกที่ผู้ใช้เลือกลงใน Shared Preferences และปิด Dialog ลง ผู้ใช้จะพบหน้าหลักของ App ดังแสดงในรูปที่ 3-96

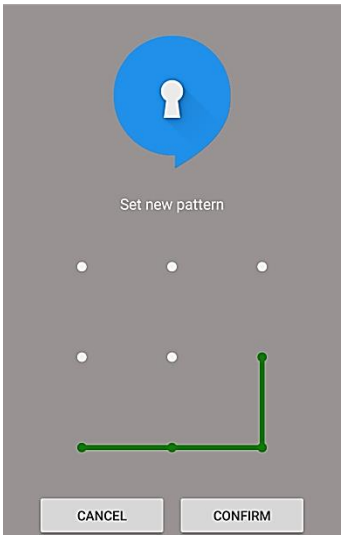


รูปที่ 3-96 หน้าหลักของ App

ข. หากผู้ใช้เลือกตัวเลือก Pattern lock และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก และบันทึกตัวเลือกที่ผู้ใช้เลือกลงใน Shared Preferences จากนั้น App จะแสดงหน้า Pattern Input สำหรับให้ผู้ใช้สร้าง Pattern ที่ใช้ในการยืนยันตัวตน ดังแสดงในรูปที่ 3-97

รูปที่ 3-97 หน้าสร้าง Pattern

ผู้ใช้ใส่ Pattern ที่ผู้ใช้ต้องการ (จะต้องมีความยาวไม่น้อยกว่า 4 จุด แต่ไม่เกิน 9 จุด) จากนั้น App จะแสดงปุ่ม 2 ปุ่ม ได้แก่ ปุ่ม CANCEL คือ ยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และปุ่ม CONFIRM คือ จดจำ Pattern และดำเนินการต่อ ดังแสดงในรูปที่ 3-98

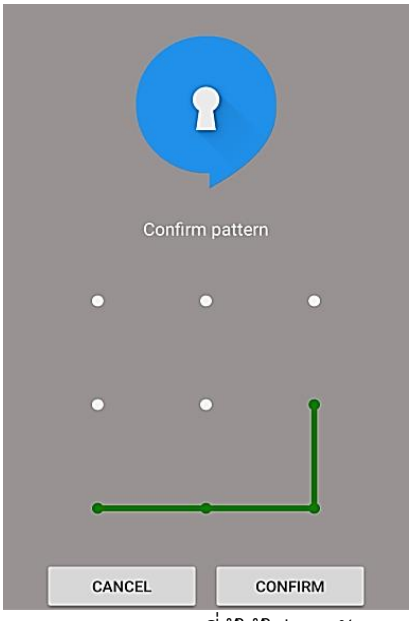


รูปที่ 3-98 หน้าสร้าง Pattern (ผู้ใช้ใส่ Pattern ครั้งแรก)

หลังจากผู้ใช้กดปุ่ม CONFIRM เพื่อจดจำ Pattern และดำเนินการต่อ App จะเปลี่ยนข้อความใต้รูป Logo App Signal จาก “Set new pattern” เป็นข้อความ “Confirm Pattern” ดังแสดงในรูปที่ 3-99

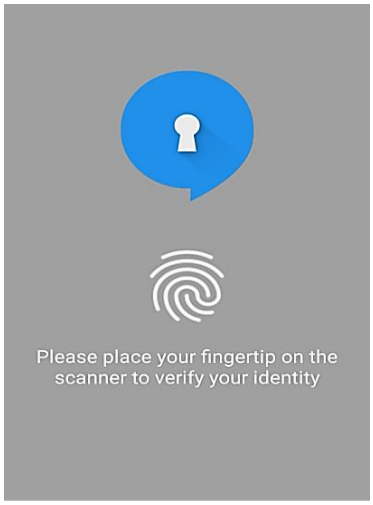
รูปที่ 3-99 หน้าสร้าง Pattern (App แสดงข้อความ Confirm pattern)

ให้ผู้ใช้ใส่ Pattern อีกครั้ง โดยจะต้องเหมือนกับ Pattern ที่ผู้ใช้ใส่ก่อนหน้า เพื่อเป็นการยืนยันความถูกต้องตรงกัน เมื่อผู้ใช้ใส่ Pattern สำเร็จ App จะแสดงปุ่ม 2 ปุ่ม (ดังแสดงในรูปที่ 3-100) ได้แก่ ปุ่ม CANCEL คือ การยกเลิก Pattern ที่ผู้ใช้ได้ใส่ไว้ และปุ่ม CONFIRM คือ การบันทึก Pattern ที่ผู้ใช้ใส่ลงใน Shared Preferences และปิดหน้าสร้าง Pattern ลงแล้วแสดงหน้าหลักของ App



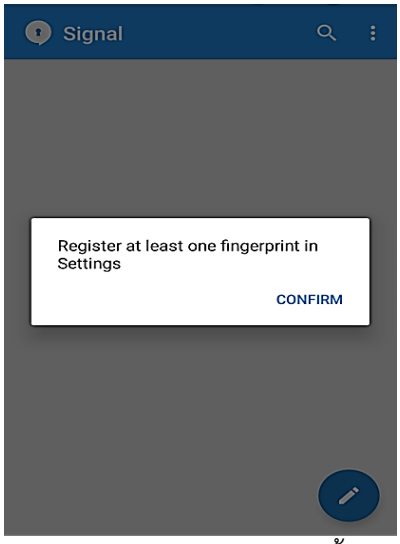
รูปที่ 3-100 หน้าสร้าง Pattern (Pattern ที่ผู้ใช้ใส่ตรงกับ Pattern ที่จดจำไว้ก่อนหน้า)

ค. กรณีที่สมาร์ทโฟนมีการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน หลังจากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม OK App จะบันทึกค่าการเปิดเข้าใช้งาน App ครั้งแรก และบันทึกตัวเลือกที่ผู้ใช้เลือกลง Shared Preferences จากนั้น App จะแสดงหน้า “Wait For Fingerprint Scan” สำหรับให้ผู้ใช้ยืนยันตัวตนโดยใช้ลายนิ้วมือ ดังแสดงในรูปที่ 3-101

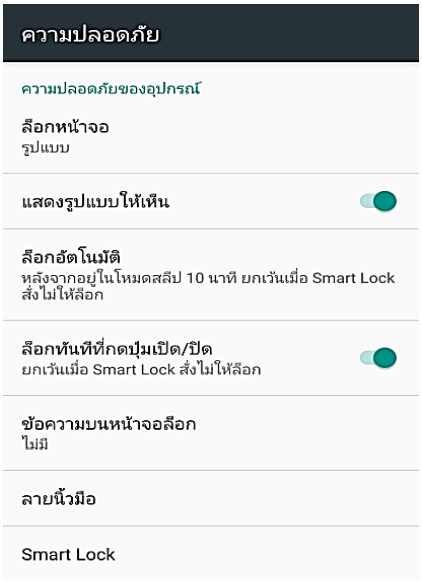


รูปที่ 3-101 หน้า “Wait For Fingerprint Scan” สำหรับให้ผู้ใช้ยืนยันตัวตนโดยใช้ลายนิ้วมือ

ง. หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม OK แต่ App ไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App จะแสดง Dialog แจ้งเตือนผู้ใช้ ดังแสดงในรูปที่ 3-102 หากผู้ใช้กดปุ่ม CONFIRM App จะแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ ดังแสดงในรูปที่ 3-103



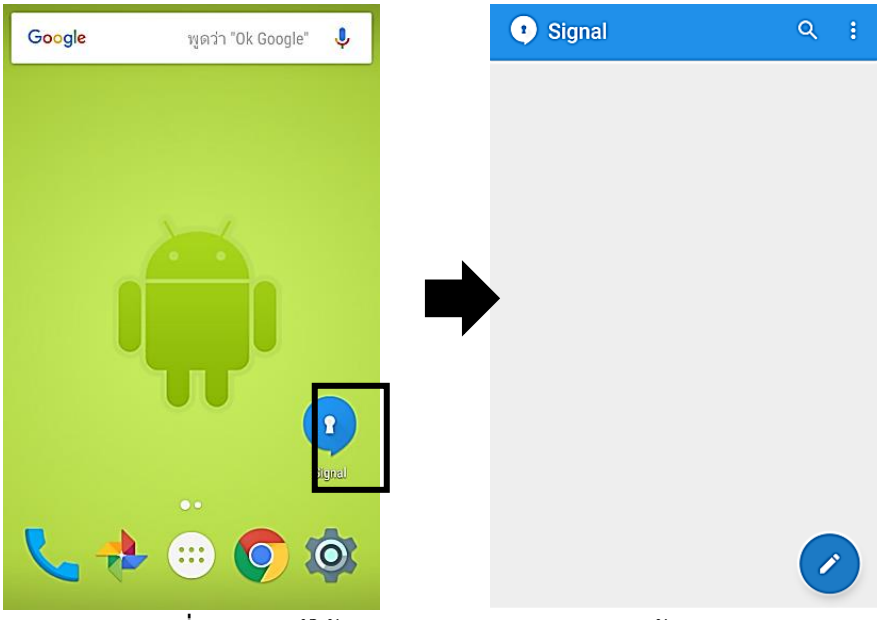
รูปที่ 3-102 Dialog แจ้งเตือนผู้ใช้กรณีไม่พบลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน



รูปที่ 3-103 หน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ

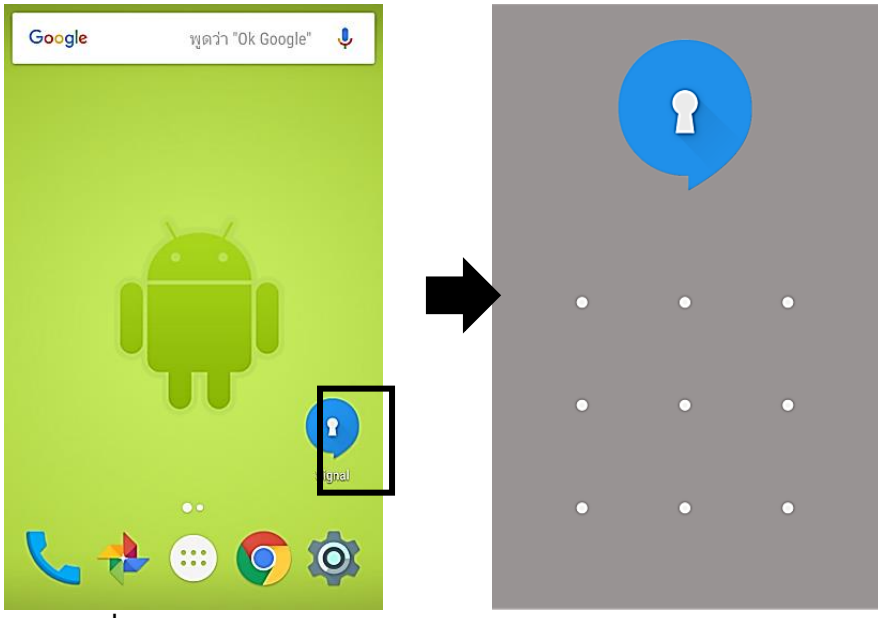
3.8.2 เปิดเข้าใช้งาน App จาก Icon ของ App Signal

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตน ผู้ใช้จะสามารถเปิดเข้าใช้งาน App ได้ทันทีโดยไม่ต้องผ่านการยืนยันตัวตน ดังแสดงในรูปที่ 3-104



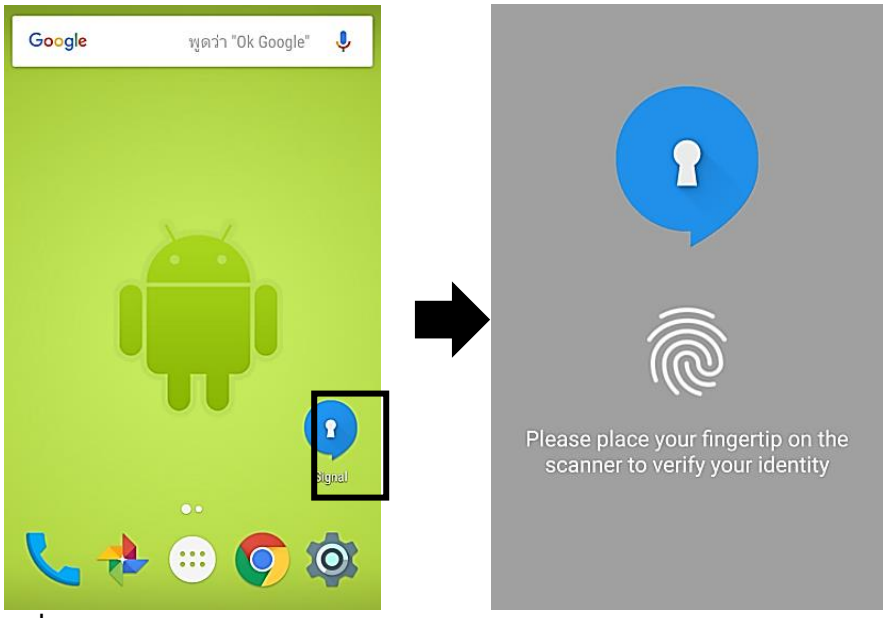
รูปที่ 3-104 ผู้ใช้เปิด App และ App แสดงหน้าหลัก

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกดปุ่ม Icon App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-105



รูปที่ 3-105 ผู้ใช้เปิด App และ App จะแสดงหน้า Pattern Input

ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-106

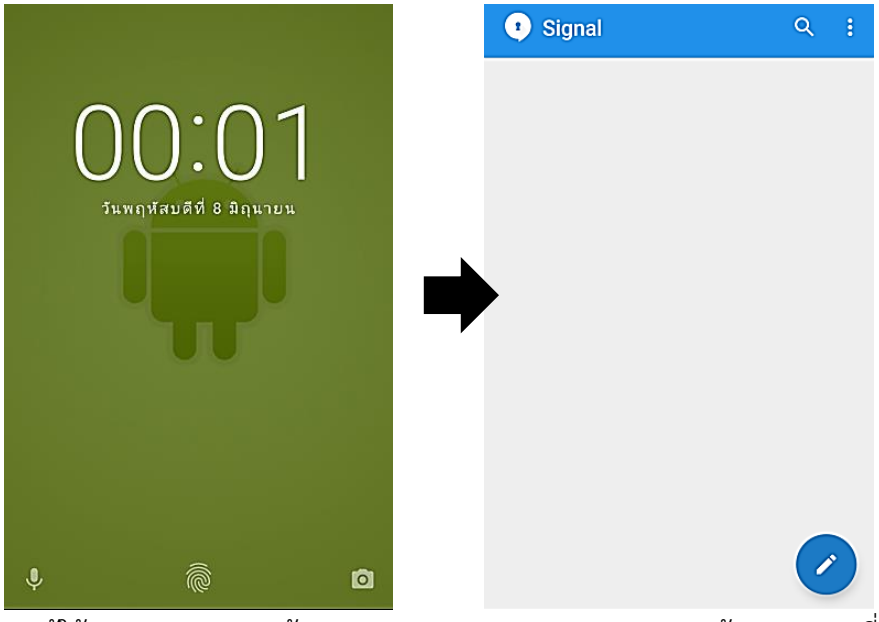


รูปที่ 3-106 ผู้ใช้เปิด App และ App จะแสดงหน้า Wait For Fingerprint Scan

3.8.3 ผู้ใช้ปลด Lock จากหน้า Lock Screen แล้วเข้าหน้า App Signal (ที่เปิดค้างไว้ก่อนหน้าทันที)

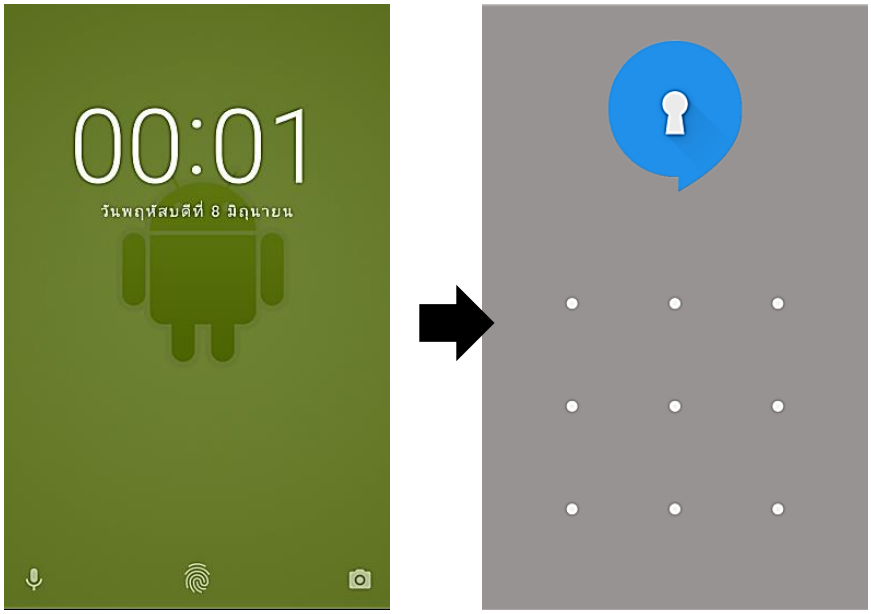
กรณีการกลับเข้าหน้า App จากหน้า Lock Screen สามารถเกิดขึ้นได้ หากผู้ใช้เลิกใช้ สมาร์ทโฟนเกินเวลาที่ตั้ง Lock Screen ไว้ หรือผู้ใช้กดปุ่ม Power Button ปิดสมาร์ทโฟนแล้วกดเปิดใหม่ ผู้วิจัยต้องการลดความเสี่ยงด้านความปลอดภัย ดังนั้นผู้วิจัยจึงออกแบบให้ผู้ใช้ต้องทำการยืนยันตัวตนอีกครั้งหนึ่งหลังผู้ใช้ปลด Lock Screen

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตนก่อนเข้า App เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้าของ App ที่เปิดค้างไว้ก่อนหน้า ดังแสดงในรูปที่ 3-107



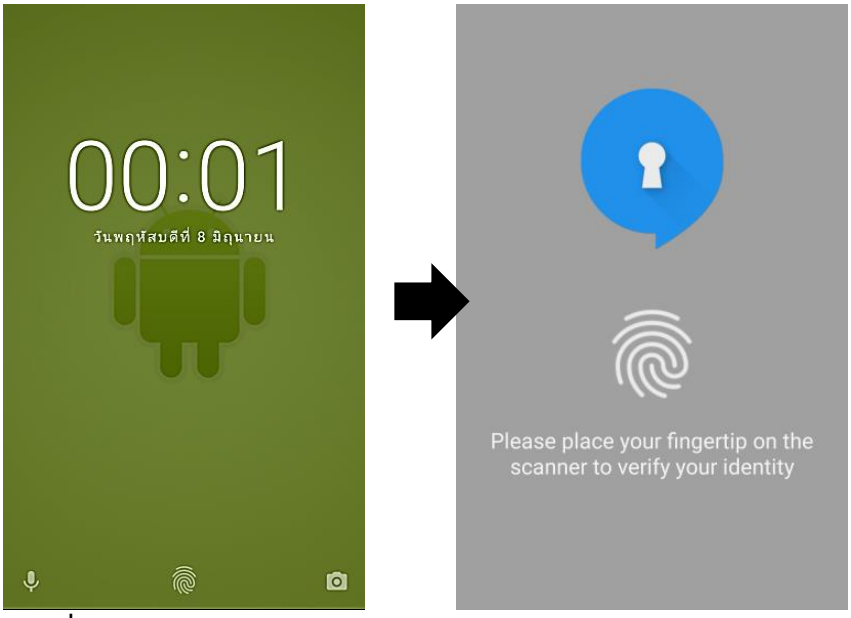
รูปที่ 3-107 ผู้ใช้ปลด Lock จากหน้า Lock Screen และ App จะแสดงหน้าของ App ที่เปิดค้างไว้

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน App ดังแสดงในรูปที่ 3-108



รูปที่ 3-108 ผู้ใช้ปลด Lock จากหน้า Lock Screen และ App จะแสดงหน้า Pattern Input

ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้ปลด Lock จากหน้า Lock Screen App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน App ดังแสดงในรูปที่ 3-109



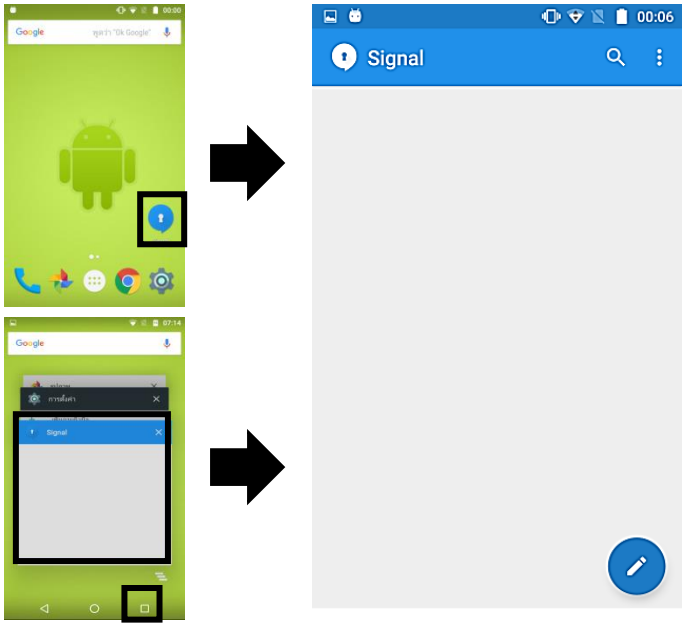
รูปที่ 3-109 ผู้ใช้ปลด Lock จากหน้า Lock Screen และ App จะแสดงหน้า Wait For Fingerprint Scan

3.8.4 เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที

กรณีการเปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) เกินระยะเวลา 5 นาที สามารถเกิดขึ้นได้หากผู้ใช้เลิกใช้ App หรือเปิดเข้าใช้งาน App อื่นนานเกินระยะเวลา 5 นาที ผู้วิจัยต้องการลดความเสี่ยงด้านความปลอดภัย ดังนั้นผู้วิจัยจึงออกแบบให้ผู้ใช้ต้องทำการยืนยันตัวตนก่อนเข้าใช้งาน

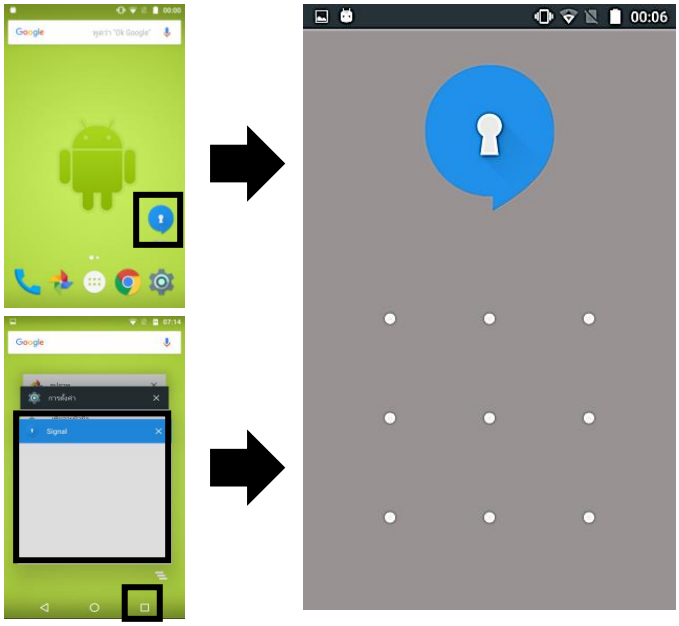
ผู้วิจัยพบว่า การกลับเข้า App Signal โดยใช้ปุ่ม Icon และโดยเลือกจาก Resent App จะให้พฤติกรรมต่างกันนั่นคือ หากผู้ใช้กลับเข้า App โดยใช้ปุ่ม Icon Android จะแสดงหน้าหลักของ App เสมอแต่หากผู้ใช้กลับเข้า App โดยเลือก App จาก Resent App Android จะแสดงหน้า Activity ที่เคยทำงานล่าสุด (Resent Activity) เสมอผู้จัดทำได้ออกแบบการยืนยันตัวตนให้มีพฤติกรรมเหมือนเดิมไม่ว่าผู้ใช้จะกลับเข้า App โดยใช้วิธีใด

ก. กรณีผู้ใช้เลือกไม่ใช้การยืนยันตัวตน ผู้ใช้จะสามารถเปิดเข้าใช้งาน App ได้ทันทีโดยไม่ต้องผ่านการยืนยันตัวตน ดังแสดงในรูปที่ 3-110



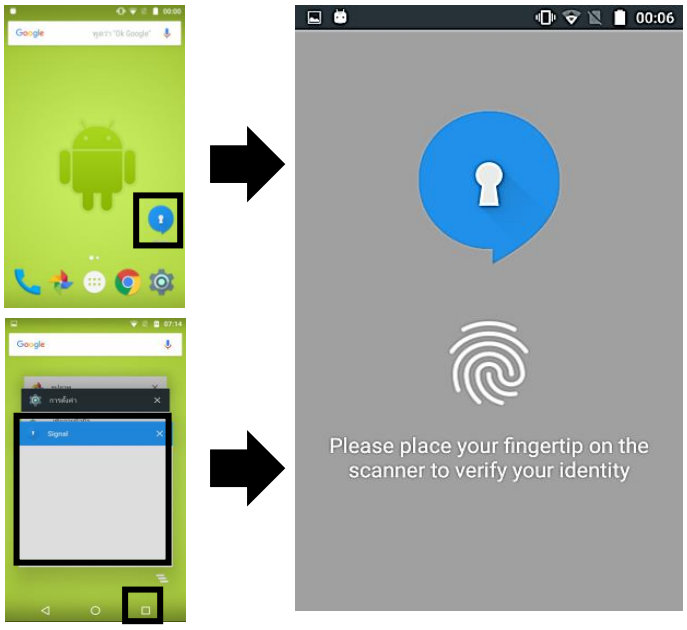
รูปที่ 3-110 ผู้ใช้เปิด App และ App จะแสดงหน้าหลัก

ข. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Pattern เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon หรือจาก Resent App App จะแสดงหน้า Pattern Input ให้ผู้ใช้ยืนยันตัวตนโดยใส่ Pattern ก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-111



รูปที่ 3-111 ผู้ใช้เปิด App และ App จะแสดงหน้า Pattern Input

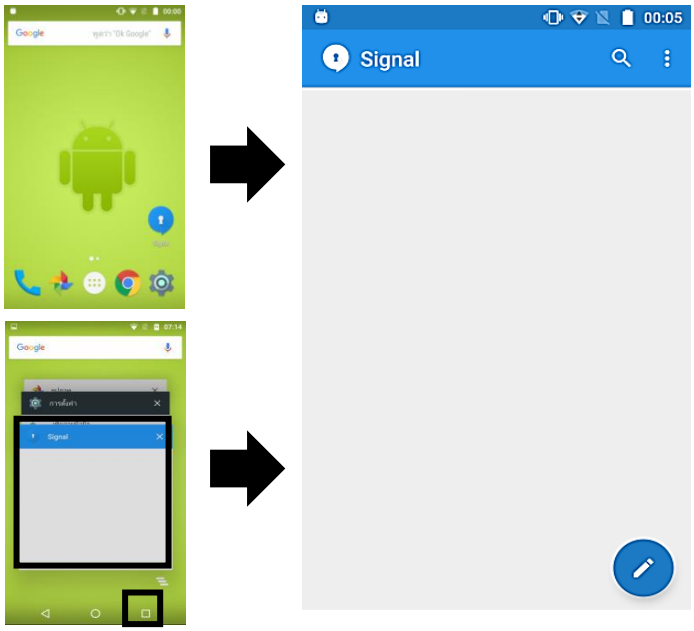
ค. กรณีผู้ใช้เลือกการยืนยันตัวตนโดยใช้ Fingerprint เมื่อผู้ใช้เปิดเข้าใช้งาน App โดยกด Icon หรือจาก Resent App App จะแสดงหน้า Wait For Fingerprint Scan ให้ผู้ใช้ยืนยันตัวตนก่อนการเข้าใช้งาน ดังแสดงในรูปที่ 3-112



รูปที่ 3-112 ผู้ใช้เปิด App และ App จะแสดงหน้า Wait For Fingerprint Scan

3.8.5 เปิดเข้าใช้งาน App อีกครั้งหลังจาก App อยู่ในสถานะหยุด (Stop State) ไม่เกินระยะเวลา 5 นาที

ผู้ใช้สามารถเปิดเข้าใช้งาน App อีกครั้ง ได้โดยไม่ต้องผ่านการยืนยันตัวตนใดๆ หาก App อยู่ในสถานะหยุด (Stop State) ไม่เกินระยะเวลา 5 นาที แม้ว่าผู้ใช้จะเคยได้เลือกให้มีการยืนยันตัวตนก่อนการเข้าใช้งาน App ก็ตาม ดังแสดงในรูปที่ 3-113



รูปที่ 3-113 ผู้ใช้เปิด App และ App จะแสดงหน้าหลัก

3.8.6 กรณีหน้าหลักไปยังหน้า Activity อื่น ๆ

หากผู้ใช้อยู่ในหน้าหลักเกิน 5 นาที หรือไม่เกิน 5 นาที ก็ตาม เมื่อผู้ใช้เปิดหน้า Activity อื่น ๆ ที่ไม่ใช่หน้าหลัก ผู้ใช้ไม่ต้องทำการยืนยันตัวตนใด ๆ เพราะหลีกเลี่ยงไม่ให้ผู้ใช้เกิดความรำคาญจากการยืนยันตัวตนบ่อย ๆ (StopTime ของหน้า Activity ที่ไม่ใช่หน้าหลักจะถูกตั้งเป็น 0 เสมอ ขณะผู้ใช้สร้าง Activity นี้)

3.8.7 กรณีหน้า Activity อื่น ๆ กลับไปยังหน้าหลัก

หากผู้ใช้อยู่ในหน้า Activity อื่น ๆ ที่ไม่ใช่หน้าหลักเกิน 5 นาที เมื่อผู้ใช้กดย้อนกลับไปยังหน้าหลัก ผู้ใช้จะต้องทำการยืนยันตัวตน แต่ถ้าหากไม่เกิน 5 นาที ผู้ใช้สามารถกดย้อนกลับมายังหน้าหลักได้ โดยไม่ต้องผ่านการยืนยันตัวตนใด ๆ (StopTime ของหน้าหลักไม่ได้ถูกตั้งค่าเป็น 0 ขณะที่ผู้ใช้กลับมายังหน้าหลักอีกครั้ง)

3.8.8 กรณีหน้า Activity อื่น ๆ ที่ไม่ใช่หน้าหลักไปยังหน้า Activity อื่น ๆ ที่ไม่ใช่หน้าหลัก

หากผู้ใช้อยู่ในหน้า Activity อื่น ๆ เกิน 5 นาที หรือไม่เกิน 5 นาที ก็ตามเมื่อผู้ใช้กดเปิดหน้า Activity อื่น ๆ อีกหน้า ผู้ใช้จะสามารถเปิดได้โดยไม่ต้องผ่านการยืนยันตัวตนใด ๆ (เพื่อหลีกเลี่ยงไม่ให้ผู้ใช้รำคาญ)

3.8.9 กรณีหน้า Activity อื่น ๆ กลับไปยังหน้า Activity อื่น ๆ

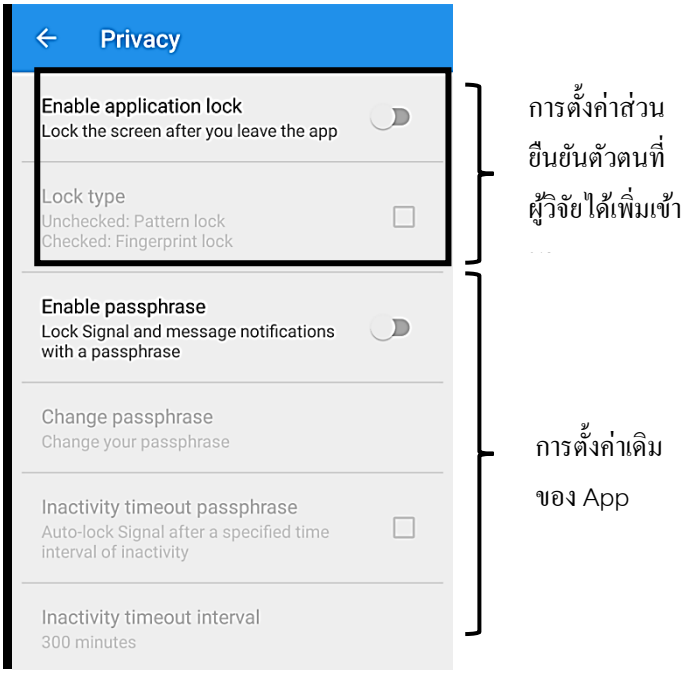
หากอยู่ในหน้า Activity อื่น ๆ เกิน 5 นาที เมื่อผู้ใช้กดย้อนกลับไปยังหน้า Activity อื่น ๆ ผู้ใช้จะต้องทำการยืนยันตัวตน แต่ถ้าหากไม่เกิน 5 นาที ผู้ใช้สามารถกดย้อนกลับมายังหน้า Activity อื่น ๆ ได้โดยไม่ต้องผ่านการยืนยันตัวตนใด ๆ

สังเกตว่า 1) พฤติกรรมการเข้า Activity หน้าหลัก จะไม่เหมือนการเข้า Activity หน้าอื่น ๆ เนื่องจาก App จะให้ผู้ใช้ยืนยันตัวตน หากเพิ่งเข้าหน้าหลัก (StopTime เท่ากับ 0) แต่ App จะไม่ได้ให้ผู้ใช้ยืนยันตัวตน หากผู้ใช้เพิ่งเข้าหน้า Activity อื่น ๆ นั้น

2) พฤติกรรมการ“กลับ”เข้า Activity หน้าหลัก และ “กลับ”เข้า Activity อื่น ๆ ที่ไม่ใช่หน้าหลัก จะเหมือนกัน (กล่าวคือ หากระยะเวลาการกลับเข้ามาใช้ App เกิน 5 นาที ผู้ใช้จะต้องยืนยันตัวตน)

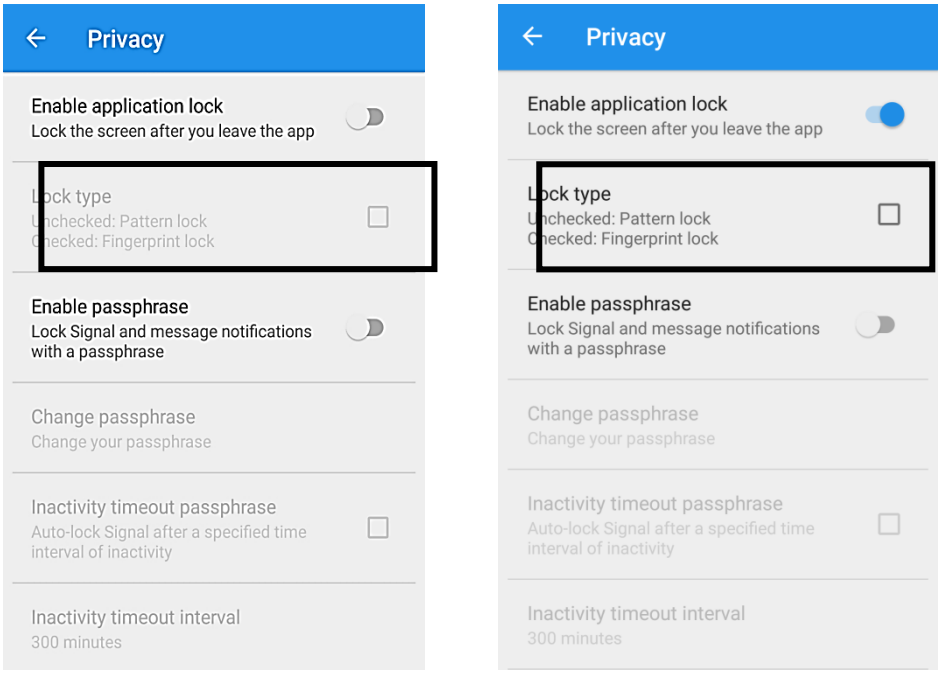
3.8.10 การตั้งค่า

เมนู Settings -> Privacy ของ App Signal มีปุ่มที่เกี่ยวกับงานวิจัยนี้ 2 ปุ่ม ได้แก่ ปุ่มแบบ Switch ชื่อ Enable application lock (ใช้สำหรับการเปิด-ปิดการยืนยันตัวตน) และปุ่มแบบ Checkbox ชื่อ Lock type (ใช้สำหรับการเปลี่ยนชนิดการยืนยันตัวตนระหว่างชนิด Pattern กับชนิด Fingerprint) ดังแสดงในรูปที่ 3-114



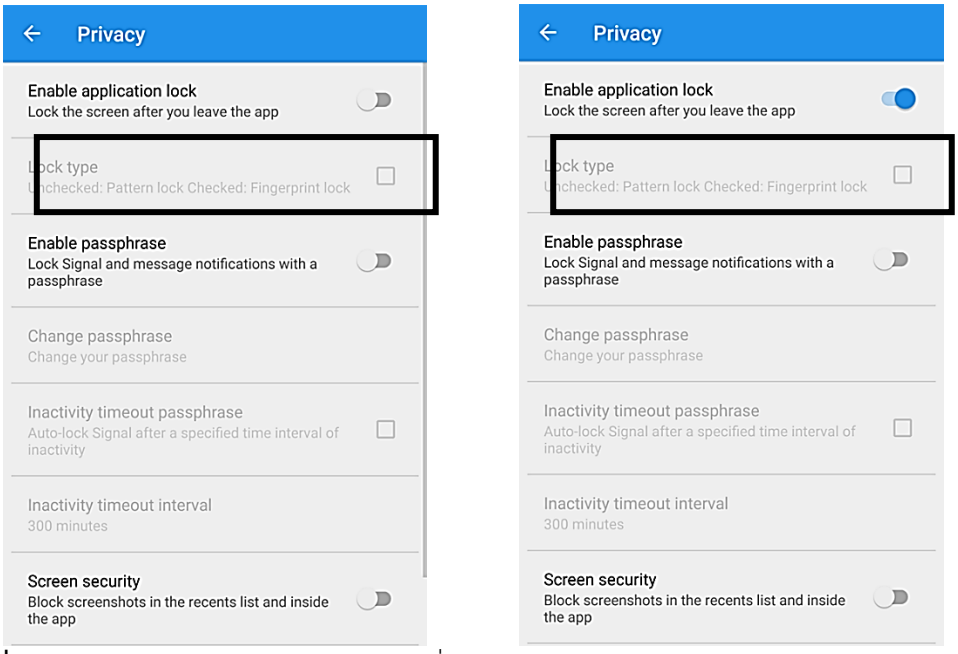
รูปที่ 3-114 หน้าเมนู Setting -> Privacy ของ App Signal

หากปุ่ม Enable application lock มีสถานะเป็น OFF ปุ่ม Lock type จะไม่สามารถใช้งานได้ และสีตัวหนังสือ Lock type จะเป็นสีเทา (ปุ่มถูก Disable) ในทางตรงกันข้าม หากปุ่ม Enable application lock มีสถานะเป็น ON ปุ่ม Lock type จะสามารถใช้งานได้ และสีตัวหนังสือ Lock type จะเป็นสีดำ (ปุ่มถูก Enable) ดังแสดงในรูปที่ 3-115



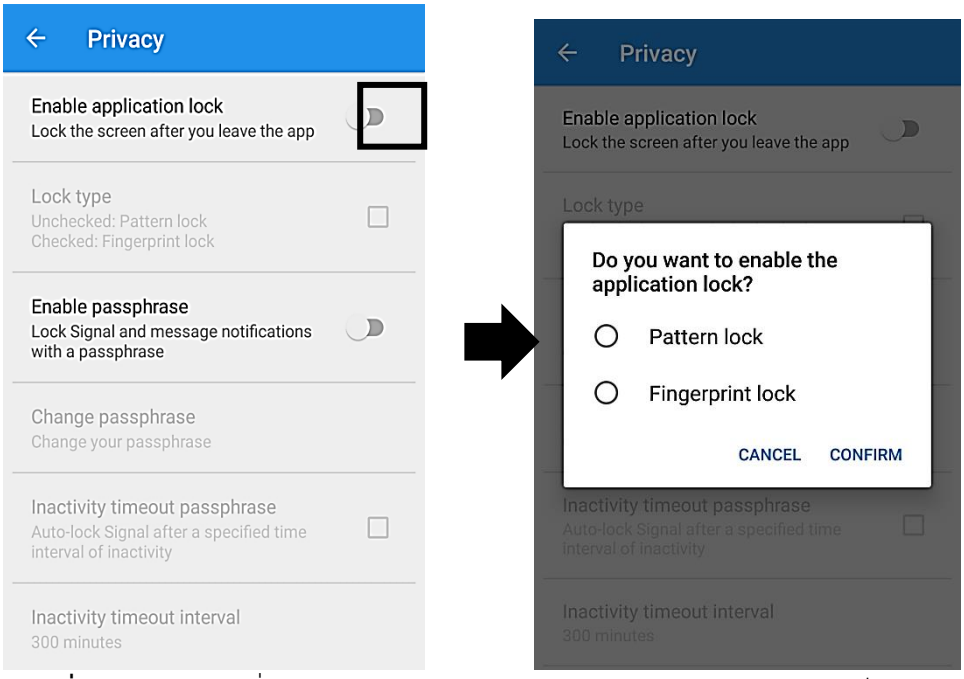
รูปที่ 3-115 สถานะของปุ่ม Lock type เมื่อ Enable / Disable application lock

สำหรับสมาร์ทโฟนที่ไม่มีระบบสแกนลายนิ้วมือ หากผู้ใช้ตั้งค่าปุ่ม Enable application lock ให้มีสถานะเป็น ON แล้วก็ตาม ปุ่ม Lock type จะไม่สามารถใช้งานได้สีตัวหนังสือ Lock type จะเป็นสีเทา (ปุ่มถูก Disable) ดังแสดงในรูปที่ 3-116



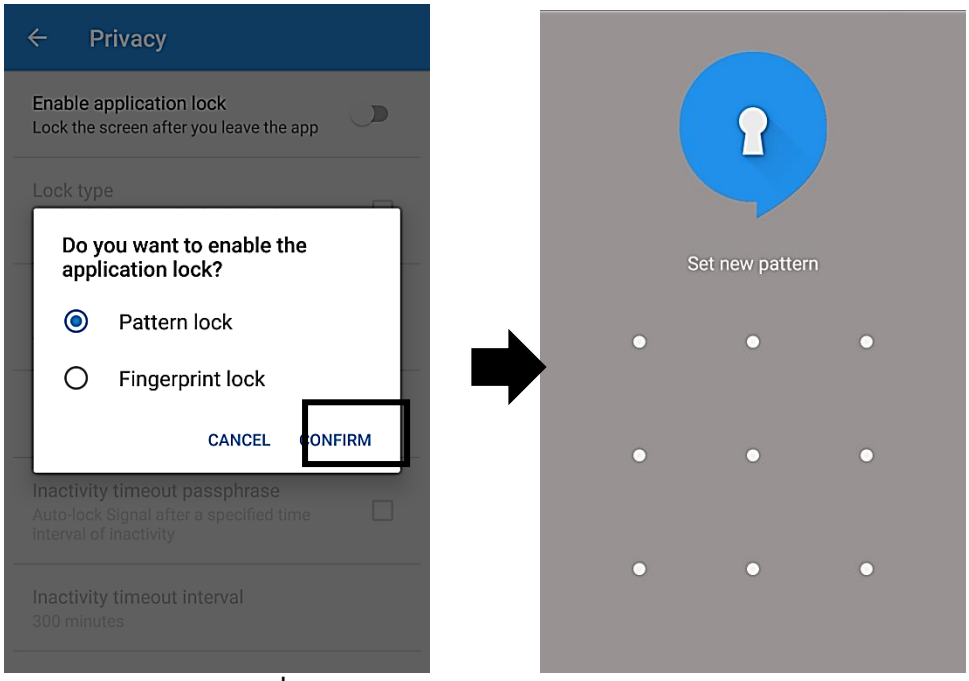
รูปที่ 3-116 สถานะของปุ่ม Lock type เมื่อ Enable / Disable application lock (กรณีที่สมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ)

- กรณีผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ OFF เป็น ON App แสดง Dialog ให้ผู้ใช้เลือกชนิดการยืนยันตัวตน ดังแสดงในรูปที่ 3-117 (ในกรณีที่สมาร์ทโฟนไม่มีระบบสแกนลายนิ้วมือ Dialog จะมีเพียง Pattern lock ให้ผู้ใช้เลือก)



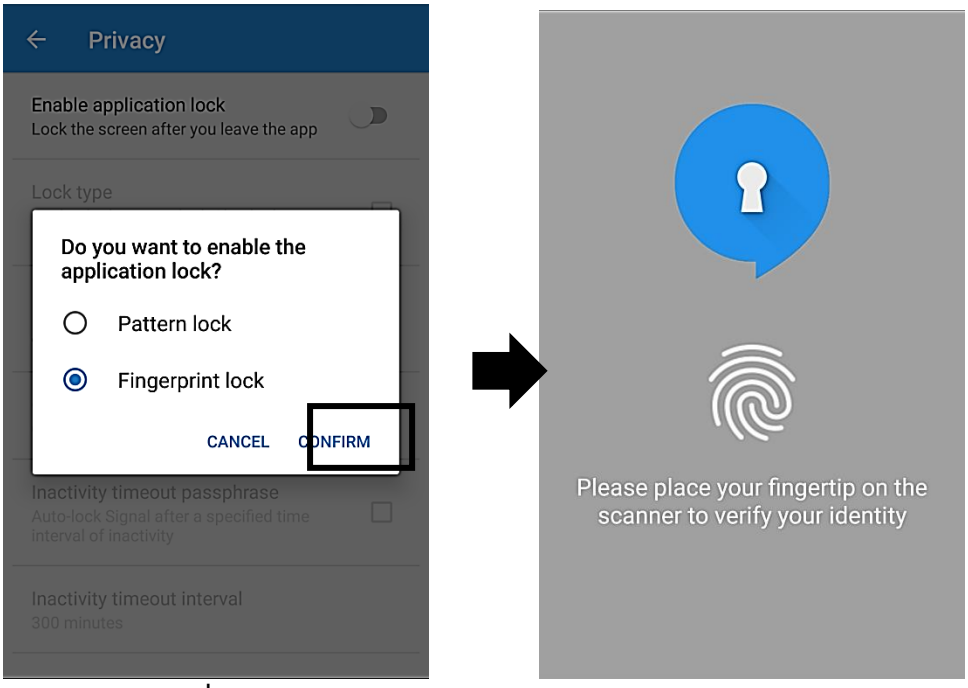
รูปที่ 3-117 ผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก OFF เป็น ON

ก. หากผู้ใช้เลือกตัวเลือก Pattern lock และกดปุ่ม CONFIRM App จะทำการตั้งค่าใช้ Pattern ใน Shared Preference และแสดงหน้าสร้าง Pattern ดังแสดงในรูปที่ 3-118



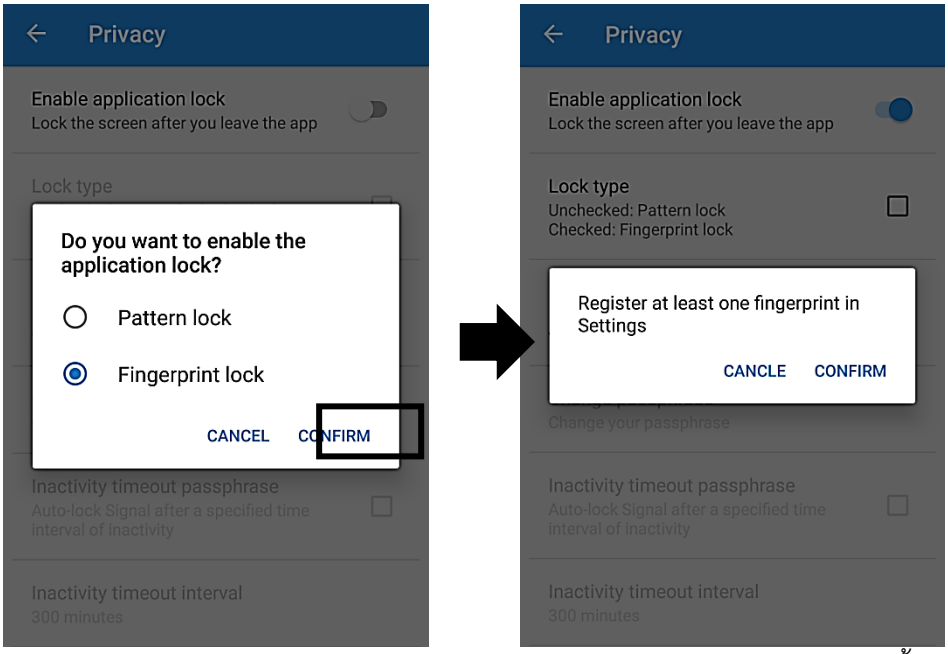
รูปที่ 3-118 ผู้ใช้เลือกตัวเลือก Pattern lock

ข. หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม CONFIRM App จะทำการตั้งค่าใช้ Fingerprint ใน Shared Preference และแสดงหน้า Wait For Fingerprint Scan สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 3-119



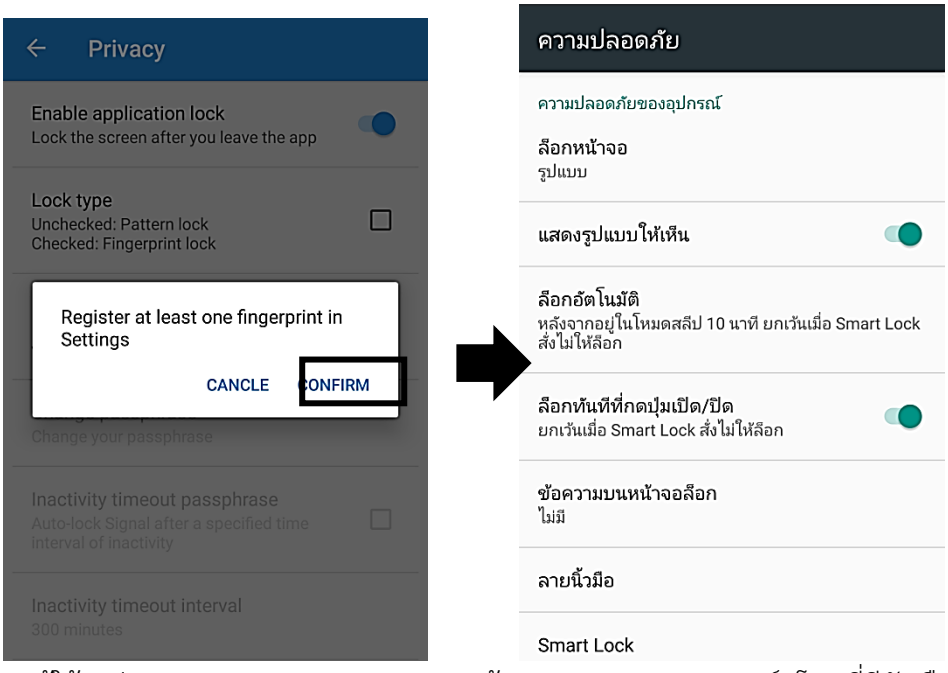
รูปที่ 3-119 ผู้ใช้เลือกตัวเลือก Fingerprint lock

ค. หากผู้ใช้เลือกตัวเลือก Fingerprint lock และกดปุ่ม CONFIRM แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App แสดง Dialog แจ้งเตือนผู้ใช้ ดังแสดงในรูปที่ 3-120



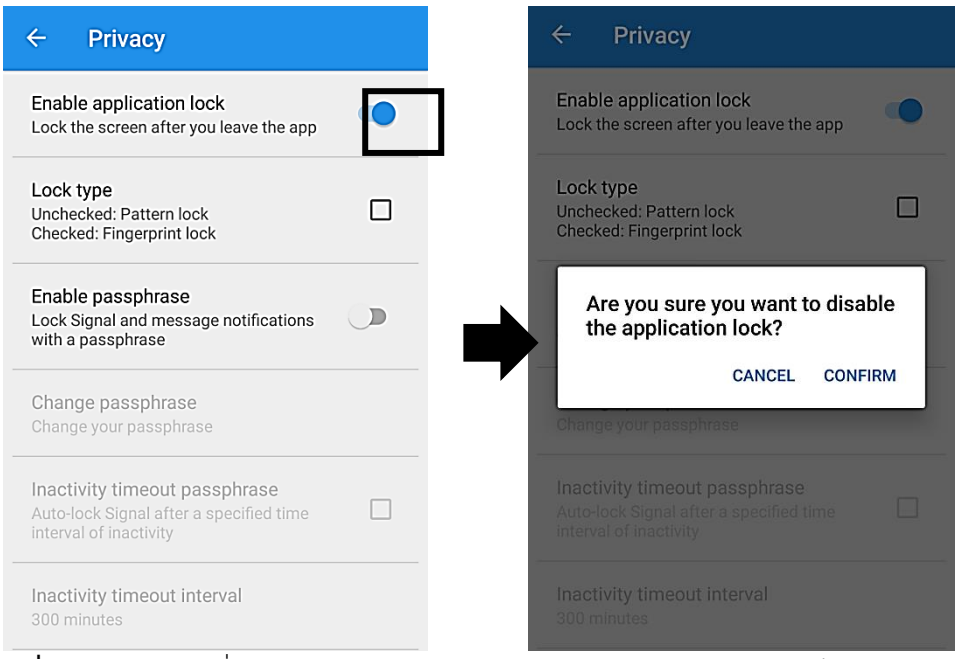
รูปที่ 3-120 ผู้ใช้เลือกตัวเลือก Fingerprint lock แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน

ง. หากผู้ใช้กดปุ่ม CONFIRM App จะตั้งค่าใช้ Fingerprint ใน Shared Preference และแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ ดังแสดงในรูปที่ 3-121



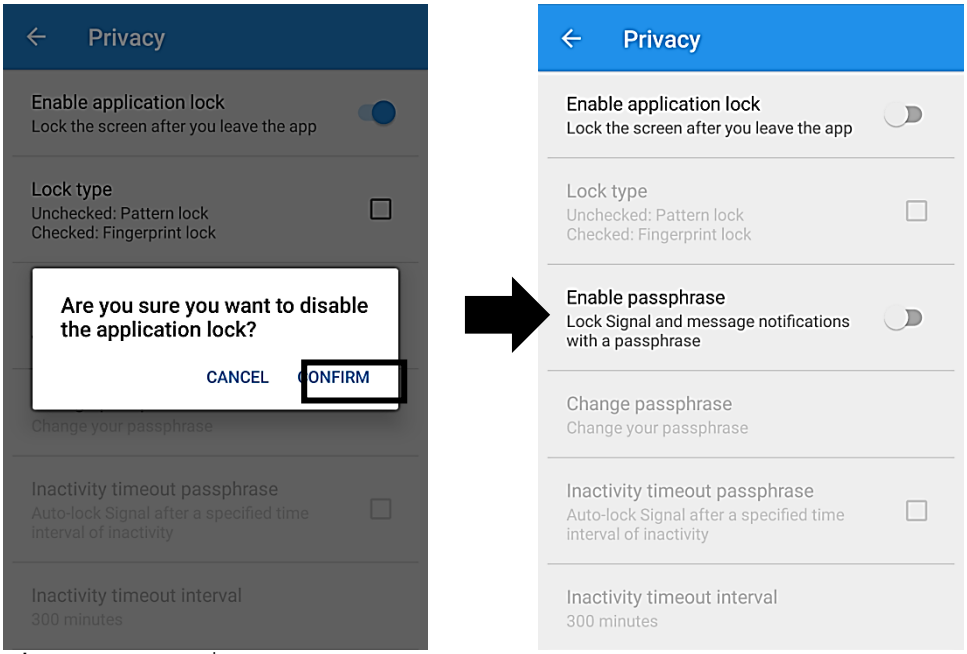
รูปที่ 3-121 ผู้ใช้กดปุ่ม CONFIRM และ App แสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ

- กรณีผู้ใช้กดปุ่ม Enable application lock เพื่อเปลี่ยนสถานะ ON เป็น OFF App แสดง Dialog แจ้งเตือนผู้ใช้ ดังแสดงในรูปที่ 3-122



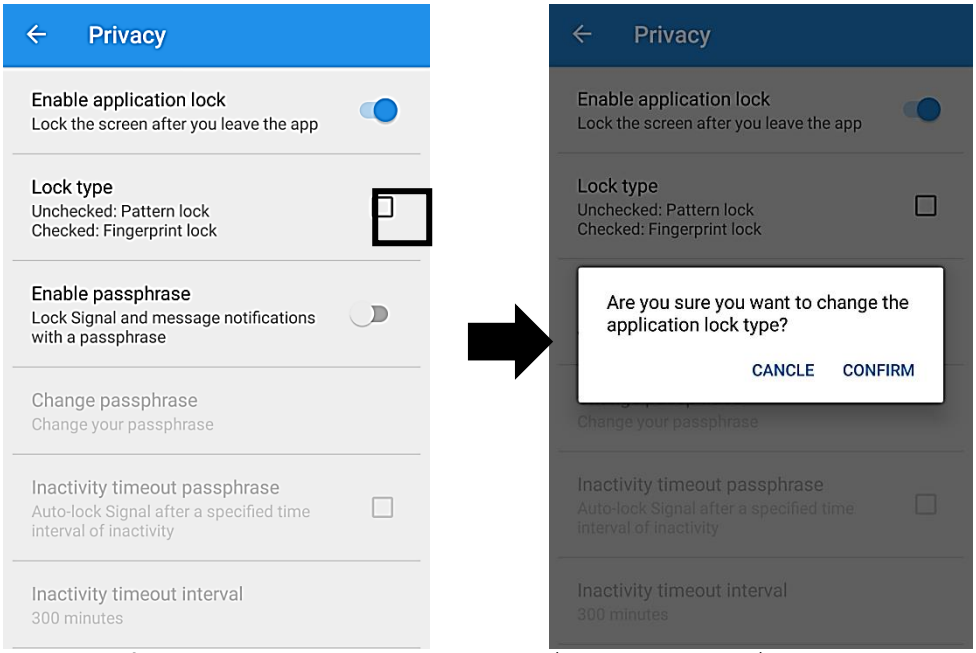
รูปที่ 3-122 ผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก ON เป็น OFF

- หากผู้ใช้กดปุ่ม CONFIRM App จะทำการตั้งค่าปุ่ม Enable application lock ให้มีสถานะเป็นOFF ใน Shared Preferences และลบ Pattern ที่ผู้ใช้ได้บันทึกไว้สำหรับการยืนยันตัวตนออกจาก Shared Preferences ดังแสดงในรูปที่ 3-123



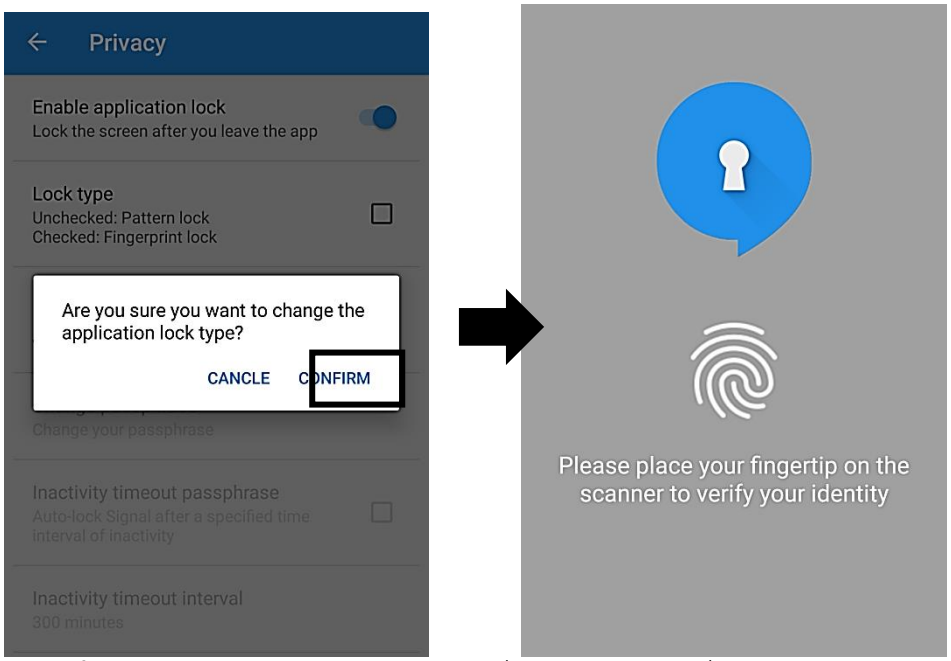
รูปที่ 3-123 ผู้ใช้เปลี่ยนสถานะปุ่ม Enable application lock จาก ON เป็น OFF

- กรณีผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนชนิดการยืนยันตัวตนจากแบบ Pattern เป็นแบบ Fingerprint App จะแสดง Dialog แจ้งเตือน ดังแสดงในรูปที่ 3-124



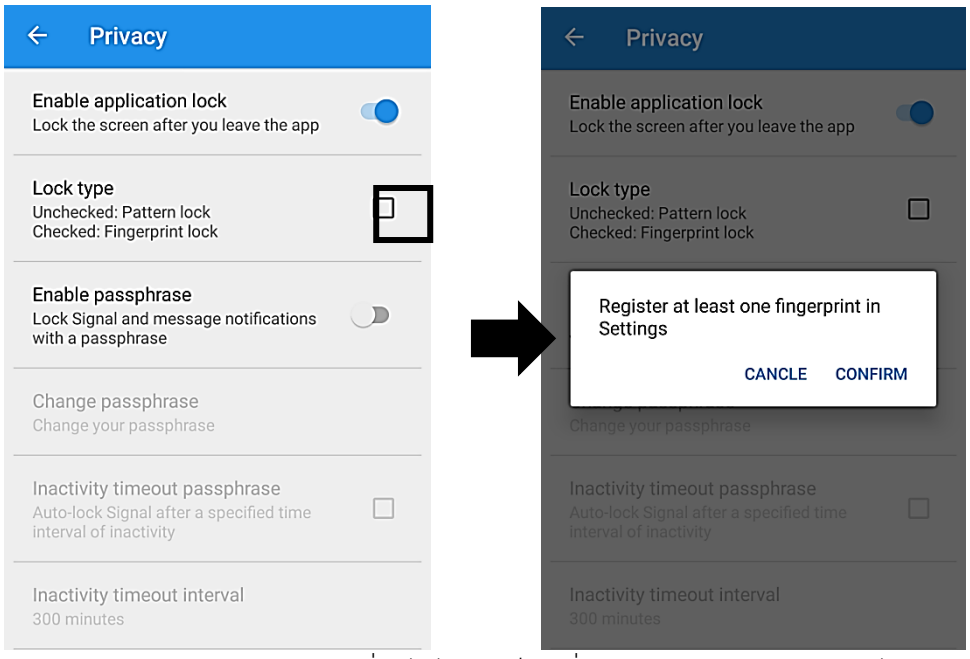
รูปที่ 3-124 ผู้ใช้กดปุ่ม Lock type ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก

- หากผู้ใช้กดปุ่ม CONFIRM ใน Dialog ในรูปที่ 3-125 App จะตั้งค่าใช้ Fingerprint ใน Shared Preference และแสดงหน้า Wait For Fingerprint Scan สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 4-44 (ในกรณีนี้ สมาร์ทโฟนมีลายนิ้วมือที่ลงทะเบียนแล้ว อย่างน้อย 1 ลายนิ้วมือ)



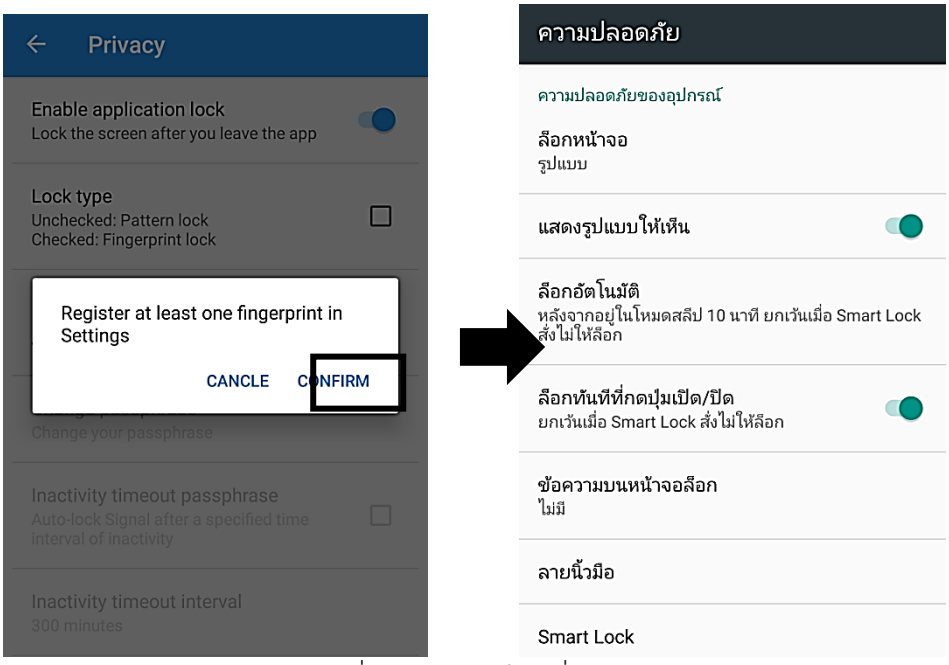
รูปที่ 3-125 ผู้ใช้กดปุ่ม Lock type ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก

- กรณีผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนชนิดการยืนยันตัวตนจากแบบ Pattern เป็นแบบ Fingerprint แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน App จะแสดง Dialog แจ้งเตือนผู้ใช้ ดังแสดงในรูปที่ 3-126



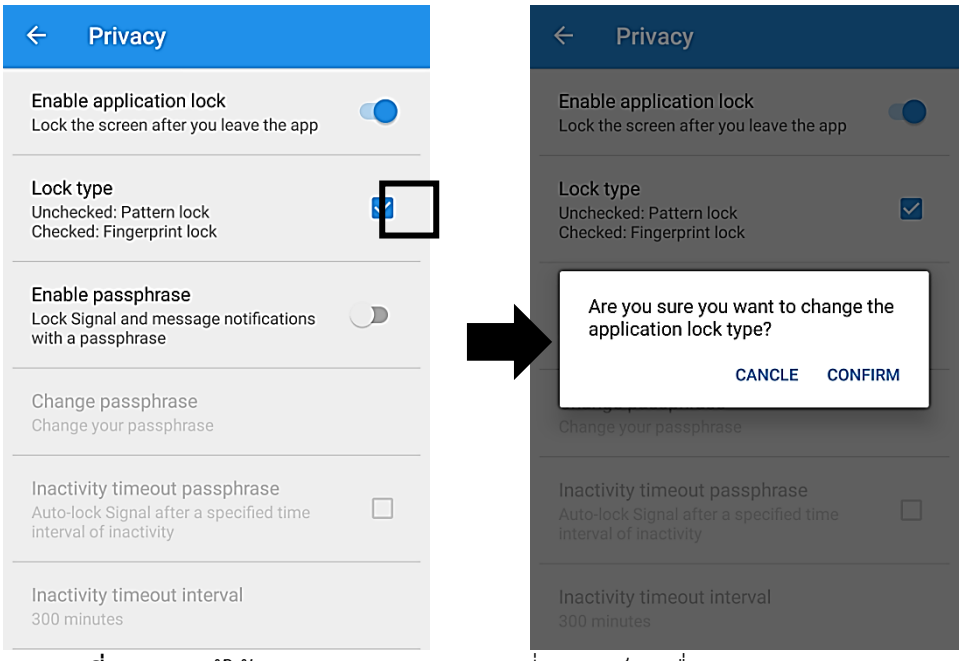
รูปที่ 3-126 ผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก) แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน

- หากผู้ใช้กดปุ่ม CONFIRM ในรูปที่ 4-45 App จะตั้งค่าใช้ Fingerprint ใน Shared Preference และแสดงหน้าเมนู Settings ของสมาร์ทโฟนที่มีตัวเลือกการเพิ่มลายนิ้วมือ ดังแสดงในรูปที่ 3-127



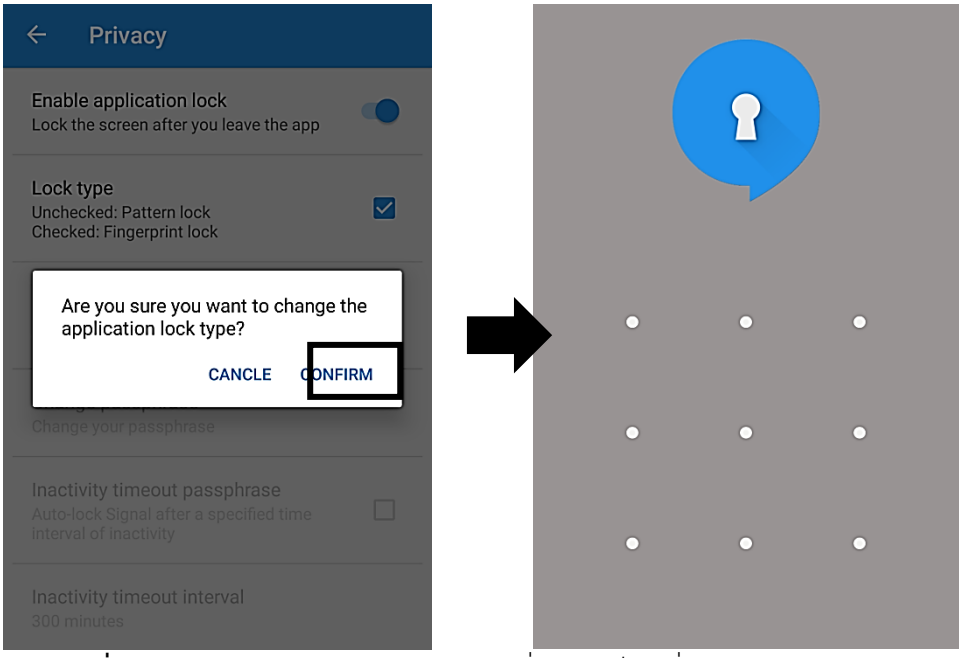
รูปที่ 3-127 ผู้ใช้กดปุ่ม Lock type ขณะที่ปุ่มไม่ได้ถูกเช็คเครื่องหมายถูก แต่สมาร์ทโฟนไม่พบการลงทะเบียนลายนิ้วมืออย่างน้อย 1 ลายนิ้วมือในสมาร์ทโฟน

- กรณีผู้ใช้กดปุ่ม Lock type (ขณะที่ปุ่มถูกเช็คเครื่องหมายถูก) เพื่อต้องการเปลี่ยนประเภทการยืนยันตัวตนจากแบบ Fingerprint เป็นแบบ Pattern App จะแสดง Dialog แจ้งเตือน ดังแสดงในรูปที่ 3-128



รูปที่ 3-128 ผู้ใช้กดปุ่ม Lock type ขณะที่ปุ่มถูกเช็คเครื่องหมายถูก

- หากผู้ใช้กดปุ่ม CONFIRM ใน Dialog ในรูปที่ 3-128 App จะแสดงหน้า Pattern Input สำหรับให้ผู้ใช้ยืนยันตัวตน ดังแสดงในรูปที่ 3-129 แต่ในกรณีที่ผู้ใช้ยังไม่เคยสร้าง Pattern App จะแสดงหน้าสร้าง Pattern แทน



รูปที่ 3-129 ผู้ใช้กดปุ่ม Lock type ขณะที่ปุ่มถูกเช็คเครื่องหมายถูก

บทที่ 4

สรุปผลการทดลอง

ในบทนี้ ผู้วิจัยจะสรุปผลการทดสอบแชทแอปพลิเคชันโดยใช้ ARO และการพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) อันประกอบด้วย สรุปผลการดำเนินงาน ปัญหาและอุปสรรค ประโยชน์ที่ได้รับ ข้อเสนอแนะ และแนวทางสำหรับพัฒนางานวิจัยนี้เพื่อให้ผู้ที่สนใจการทดสอบแอปพลิเคชันหรือพัฒนาแอปพลิเคชันเพิ่มเติมสามารถพัฒนาต่อไปได้

4.1 สรุปผลการดำเนินงาน

1. สามารถติดตั้ง Signal Server ที่ใช้ภายในได้โดยไม่ต้องผ่านเครื่องแม่ข่ายอื่น ๆ
2. สามารถสร้างแอปพลิเคชัน Signal Android ที่ใช้งาน Signal Server ที่ติดตั้งเองได้
3. สามารถใช้งานโปรแกรม ARO ทดสอบแอปพลิเคชันได้
4. สามารถบันทึกผลการทดสอบด้วย ARO ได้
5. ARO สามารถนำผลการทดสอบออกมาสรุปได้
6. สามารถพัฒนาส่วนยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) ได้
7. แชทแอปพลิเคชัน (Signal) สามารถใช้งานได้เป็นปกติ

4.2 ข้อจำกัดของงานวิจัย

1. การนำ Open Source มาพัฒนาต้องใช้เวลาในการทำความเข้าใจ
2. มีการ Update Source ในขณะทำงานวิจัยทำให้บางอย่างมีผลกระทบ
3. ข้อจำกัดในการใช้งาน SMS Service เนื่องจากผู้ให้บริการมีค่าใช้จ่าย (ผู้ใช้บริการไม่ต้องจ่าย)
4. การทดสอบต้องใช้สัญญาณการเชื่อมต่อที่เสถียร
5. Background Process ของ Android มีผลต่อการทดสอบเล็กน้อย
6. การปิด Service ต่าง ๆ ใน Android ไม่สามารถปิดได้หมด
7. ไม่สามารถเรียกใช้การยืนยันตัวตนแบบ Pattern lock ของทางแอนดรอยด์ได้จึงทำให้ความปลอดภัยลดลง

4.3 ข้อเสนอแนะ

1. ทดสอบ Signal ที่ทำงานบนระบบปฏิบัติการ iOS
2. พัฒนาแอปพลิเคชันปรับปรุงในด้านที่ไม่ผ่านการทดสอบ
3. พัฒนา Signal Server ให้ใช้ Service การส่งภาพโดยส่งผ่าน Server ของผู้วิจัยเอง
4. ใช้เครื่องมือทดสอบอื่น ๆ ที่มีมาตรฐานเพิ่มเติมด้วย
5. ควรให้การเชื่อมต่อสัญญาณมีความเสถียรมากขึ้น
6. พัฒนาส่วนการยืนยันตัวตนเพิ่มเติมในแชทแอปพลิเคชัน (Signal) บนระบบปฏิบัติการ iOS

บทที่ 5

ผลผลิต

5.1 ประโยชน์ที่ได้รับ

1. สามารถติดตั้ง Signal Server เป็นของผู้อ่านที่สนใจได้
2. สามารถสร้าง Signal Android ที่เชื่อมต่อกับ Signal Server ใด ๆ ที่เขาให้บริการได้
3. สามารถใช้ ARO เพื่อทดสอบแอปพลิเคชันได้
4. สามารถเข้าใจผลของการทดสอบจาก ARO ได้
5. สามารถนำผลการทดสอบไปอ้างอิงความปลอดภัย และคุณภาพของแอปพลิเคชันได้
6. สามารถป้องกันหากผู้ไม่หวังดีหรือบุคคลอื่นที่ไม่ใช่เจ้าของสมาร์ทโฟน ต้องการที่จะเข้าใช้งานแชทแอปพลิเคชัน (Signal)

5.2 ผลสำเร็จและความคุ้มค่าของการวิจัย

ผลสำเร็จเบื้องต้น (P) คือ การพัฒนาองค์ความรู้พื้นฐานเกี่ยวกับการใช้การสแกนลายนิ้วมือกับโทรศัพท์ที่ระบบปฏิบัติการแอนดรอยด์ องค์ความรู้พื้นฐานเกี่ยวกับการติดตั้งระบบเครือข่ายเสมือน ระบบส่งข้อความ และระบบส่งรหัสผ่านแบบใช้ครั้งเดียว

ผลสำเร็จกึ่งกลาง (I) คือ การถ่ายทอดองค์ความรู้ และเทคนิคการพัฒนาโปรแกรมและเครื่องแม่ข่ายสำหรับการรับและส่งข้อความลับ

ผลสำเร็จตามเป้าประสงค์ (G) คือ องค์ความรู้ที่ได้ ได้นำไปถ่ายทอดให้กับหน่วยงานที่สนใจในการรับส่งข้อความที่เป็นความลับ

บรรณานุกรม

[1] ARO [ออนไลน์] เข้าถึงได้จาก https://developer.att.com/application-resource-optimizer (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[2] ARO Best-Practices [ออนไลน์] เข้าถึงได้จาก https://developer.att.com/application-resource-optimizer/docs/best-practices (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[3] ATT ARO Analysis Guide [ออนไลน์] เข้าถึงได้จาก https://developer.att.com/static-assets/documents/aro/ATT_ARO_Analysis_
Guide_4-0.pdf (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[4] PC Mag Only 6 Messaging Apps Are Truly Secure [ออนไลน์] เข้าถึงได้จาก http://www.pcmag.com/article2/0,2817,2471658,00.asp (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[5] Signal Android [ออนไลน์] เข้าถึงได้จาก https://github.com/WhisperSystems/Signal-Android/wiki (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[6] Signal News Topics [ออนไลน์] เข้าถึงได้จาก https://www.blognone.com/topics/signal (วันที่ค้นข้อมูล: 20 เมษายน 2560)

[7] Signal Setup Server [ออนไลน์] เข้าถึงได้จาก https://github.com/lucaconte/BeatTheMeddler/blob/a16faea568c41150c10eb4162b74b94faebbfbb7/README.md (วันที่ค้นข้อมูล: 10 มกราคม 2560)

[8] Signal VS WhatApps [ออนไลน์] เข้าถึงได้จาก https://www.blognone.com/node/81380 (วันที่ค้นข้อมูล: 20 เมษายน 2560)

[9] Android Fingerprint Authentication [ออนไลน์] เข้าถึงได้จาก http://www.techotopia.com/index.php/An_Android_Fingerprint_Authentication_Tutorial (วันที่ค้นข้อมูล: 24 พฤษภาคม 2560)

[10] Fingerprint API Tutorial [ออนไลน์] เข้าถึงได้จาก http://wanttobeanandroiddev.blogspot.com/2016/04/android-fingerprint-api-tutorial.html (วันที่ค้นข้อมูล: 24 พฤษภาคม 2560)

[11] Activity Lifecycle พื้นฐาน Android ที่ Developer ควรรู้ [ออนไลน์] เข้าถึงได้จาก http://www.artit-k.com/android-activity-lifecycle/ (วันที่ค้นข้อมูล: 24 พฤษภาคม 2560)

[12] เก็บค่าตัวแปรให้ถาวรแบบง่ายๆได้ด้วย Shared Preferences [ออนไลน์] เข้าถึงได้จาก http://www.akexorcist.com/2014/09/android-shared-preferences.html (วันที่ค้นข้อมูล: 24 พฤษภาคม 2560)

[13] LockPattern [ออนไลน์] เข้าถึงได้จาก https://github.com/pro100svitlo/LockPattern (วันที่ค้นข้อมูล: 24 พฤษภาคม 2560)