ระบบพิสูจน์และยืนยันตัวตนดิจิทัลของประเทศไทย

ประมาณปลายๆปีนี้ประเทศไทยจะมีระบบการพิสูจน์และยืนยันตัวตนทางดิจิทัลออกมาให้เราๆท่านๆใช้กัน (กำหนดการที่ผมได้ยินมา ซึ่งน่าจะยังไม่ยืนยันและยังไม่เป็นทางการ)

ซึ่งโปรเจคนี้มีตัวย่อที่ทีมทำงานเรียกกันเองว่า NDID

เว็บหลักอย่างเป็นทางการ หรือ เฟสบุ๊คเพจของโครงการ

ผมเองอาจจะอธิบายได้ไม่ดีเท่าไหร่นัก เลยอยากแนะนำให้ดูวีดีโอที่พูดในงาน TEDx ที่เชียงใหม่ (~15 นาที) โดย ดร.ภูมิ ภูมิรัตน ซึ่งเป็นหนึ่งในคณะออกแบบทางเทคนิคของระบบนี้

ภาพรวมของระบบ

ระบบนี้มีเป้าหมายเพื่อนำตัวตนของของประชาชนขึ้นระบบดิจิทัล ซึ่งข้อดีก็คือการที่เราจะสามารถได้รับการแจ้งเตือนแบบ real time เมื่อมีการนำ “ตัวตน” ของเราไปทำธุรกรรม

และเพื่อขอความยินยอม (consent) หรือเพื่อให้เราปฏิเสธธุรกรรมทันทีหากธุรกรรมที่แจ้งเตือนนั้นเราไม่ได้เป็นคนขอทำ

(ซึ่งหนึ่งในกรณีตัวอย่างที่รู้จักกันก็คือเคสคุณฉิชา ที่กว่าจะรู้ตัวว่ามีการทำธุรกรรมเรื่องก็บานปลายไปมากแล้ว)

ผู้เกี่ยวข้องที่ต้องเชื่อมต่อกับระบบนี้ แบ่งเป็นสามกลุ่มคือ

- RP (Relying Party)
  - องค์กรที่บุคคลต้องการทำธุรกรรมด้วย
- IdP (Identity Provider)
  - องค์กรที่ทำหน้าที่พิสูจน์และยืนยันตัวตนให้กับบุคคลที่กำลังจะทำธุรกรรมกับ RP
  - รวมถึงการขอ consent หรือปฏิเสธการทำธุกรรมจากบุคคลนั้นๆ
- AS (Authorative source)
  - องค์กรที่ทำหน้าเก็บรักษาข้อมูลของบุคคล และจะส่งมอบข้อมูลให้ RP หากบุคคลยินยอม

ซึ่งองค์กรหนึ่งๆสามารถสวมบทบาทมากกว่า 1 บทบาทได้ เช่น ธนาคารหนึ่งๆอาจจะเป็นบทบาททั้งสามบทบาทเลย

ตัวอย่างการใช้งาน

ผมต้องการเปิดบัญชีกับธนาคาร x ที่ไม่เคยมีบัญชีมาก่อน แต่ผมมีบัญชีกับธนาคาร y

ซึ่งตอนที่ผมเปิดบัญชีกับธนาคาร y นั้น ทางธนาคาร y ได้ทำการพิสูจน์และยืนยันตัวตนของผมเรียบร้อยแล้ว

เมื่อผมขอเปิดบัญชีกับธนาคาร x เพื่อความสะดวก ธนาคาร x สามารถอาศัยขั้นตอนพิสูจน์และยืนยันตัวตนที่ผมได้ทำไปแล้วกับธนาคาร y ได้

ซึ่งธนาคาร y จะสามารถให้คำยืนยันได้ก็ต่อเมื่อ “ได้รับความยินยอมจากผม” ซึ่งอาจจะมาในรูปแบบของการให้ผมล็อกอินเข้าแอพลิเคชั่นของธนาคาร y ไปกดยืนยัน (หรือจะโทรหาก็ได้ อันนี้แล้วแต่ธนาคารจะเลือกใช้)

จากกรณีนี้ ธนาคาร x คือ RP ธนาคาร y คือ IdP

และถ้าหากธนาคาร x ต้องการข้อมูลของผมเพิ่มเติมนอกเหนือจากการยืนยันตัวตน ก็สามารถระบุไปในคำขอ consent ได้

ตัวอย่างเช่นธนาคาร x ต้องการ bank statement จากธนาคาร y ซึ่งคำขอนี้จะถูกระบุมาอย่างชัดเจนให้ผมอ่านก่อนที่จะเลือก consent หรือ reject

ซึ่งกรณีนี้แปลว่าธนาคาร y รับบทบาท AS ไปในตัว

เท่าที่ผมทราบมา(ยังไม่ยืนยัน) การใช้งานที่ระบบนี้(และกฎหมาย)รองรับ ช่วงแรกๆจะเป็นการขอยืนยันตัวตนเพื่อเปิดบัญชีธนาคาร

หลังจากนั้นจะเป็นการยืนยันตัวตนและขอข้อมูลทางการเงิน(เครดิตบูโร)เพื่อขอสินเชื่อบุคคล

หลังจากนั้นจะพยายามขยายการใช้งานให้แพร่หลายทั่วประเทศซึ่งจะไม่จำกัดแค่ธุรกิจการเงิน โดยระบบนี้จะเป็นเหมือนโครงสร้างพื้นฐานให้ธุรกิจต่างๆมาต่อยอดบนระบบนี้ได้

แน่นอนว่าการติดต่อกับภาครัฐก็เป็นหนึ่งในเป้าหมาย

สิ่งที่คนทั่วไปเห็น

ตัวอย่างการใช้งานด้านบน เมื่อมองจากมุมมองผู้ใช้อาจจะเป็นประมาณนี้

- ผู้ใช้เข้าเว็บไซต์ธนาคาร x เพื่อขอเปิดบัญชีออนไลน์ โดยระบุเลขบัตรประชาชน
- ผู้ใช้เลือกการยืนยันตัวตนผ่านระบบ NDID แทนการยื่นเอกสารแบบเก่า
- ผู้ใช้เลือกธนาคาร y เป็น IdP
    - ธนาคาร y ส่งคำขอไปหาลูกค้าของตนที่มีเลขบัตรประชาชนดังกล่าว
- ผู้ใช้เปิดแอพลิเคชั่นของธนาคาร y เพื่อกดยินยอมให้ธนาคาร y ยืนยันตัวตน
    - ธนาคาร y อาจจะเลือกยืนยันตัวตนโดยให้กด PIN หรือแสกนใบหน้า/ลายนิ้วมือ
- ธนาคาร x ได้รับคำยืนยันว่าผู้ใช้คือบุคคลนั้นจริง จึงดำเนินการเปิดบัญชีได้

ทั้งนี้ขอเน้นย้ำว่าข้อมูลอ่อนไหวต่างๆเช่นเลขบัตรประชาชนจะ*ไม่มีการเก็บบันทึกไว้ในระบบ*

แต่ธนาคาร x และธนาคาร y จะใช้ช่องทางอื่นเพื่อส่งข้อมูลส่วนนี้หากันโดยตรง โดยช่องทางที่ว่าจะมีการเข้ารหัสลับ (Encrypt) เพื่อไม่ให้ถูกอ่านได้ระหว่างทาง

และระบบจะมีการจัดเก็บหลักฐานต่างๆตามทฤษฎีการเข้ารหัส (cryptography) เพื่อใช้ตรวจสอบหากมีข้อพิพาทเกิดขึ้น (อ่านเพิ่มเติมได้ที่บล็อกอธิบายทางเทคนิค)

บทบาทเพิ่มเติม

นอกเหนือจาก RP, IdP, AS แล้ว ความจริงในระบบนี้ยังมีบทบาท proxy ด้วย ซึ่งผู้ใช้งานทั่วไปอาจจะมองไม่เห็นบทบาทนี้

บทบาทนี้เกิดขึ้นเพื่อให้ผู้เกี่ยวข้องที่ไม่มีทรัพยากรมากพอจะทำการต่อเชื่อมกับ NDID โดยตรง สามารถเชื่อมต่อระบบได้อย่างง่ายขึ้นโดยทำการต่อเชื่อมกับ proxy แทน

หนึ่งใน proxy เท่าที่ผมทราบคือ SET (ตลาดหลักทรัพย์แห่งประเทศไทย) ซึ่งกำลังเตรียมความพร้อมเป็น proxy ให้ บลจ.ต่างๆ

หมายเหตุ

ระบบนี้มีการประชุมเพื่อออกแบบระบบในเชิงเทคนิคจากการร่วมมือของ

- ฝ่ายไอทีจากธนาคารใหญ่ๆทั่วประเทศ
- บริษัท NCB (เครดิตบูโร)
- ตัวแทนหน่วยงานรัฐบาลต่างๆ
- ผู้ทรงคุณวุฒิอีกจำนวนหนึ่ง

(อ้างอิงจากผู้เข้าร่วมประชุมเท่าที่ผมจำได้ ตกหน่วยงานไหนไปขออภัยด้วยครับ)

ซึ่งผู้ก่อตั้งบริษัทที่ผมทำงานอยู่เข้ามามีส่วนรวมในระบบนี้ผ่านที่ปรึกษากระทรวงการคลัง ผมและเพื่อนร่วมงานเลยมีโอกาสได้เข้าไปช่วยทำงานด้วย

ผู้รับผิดชอบจัดทำระบบนี้อย่างเป็นคือบริษัท National Digital ID จำกัด

โดยได้ยินว่ามีคณะกรรมการบริหารประกอบด้วยผู้บริหารจากธนาคารใหญ่ๆประมาณยี่สิบคน (น่าจะมีท่านอื่นๆอีก แต่ผมอยู่ฝ่ายเทคนิคเลยไม่รู้มากนัก)

โดยที่ระบบนี้จะทำเป็นโครงการ opensource (ดูซอร์สโค้ดโครงการได้ที่นี่) ซึ่งผมเขียนบล็อกอธิบายสถาปัตยกรรมระบบเชิงเทคนิคไว้ที่นี่

เรื่องทางเทคนิคของ NDID

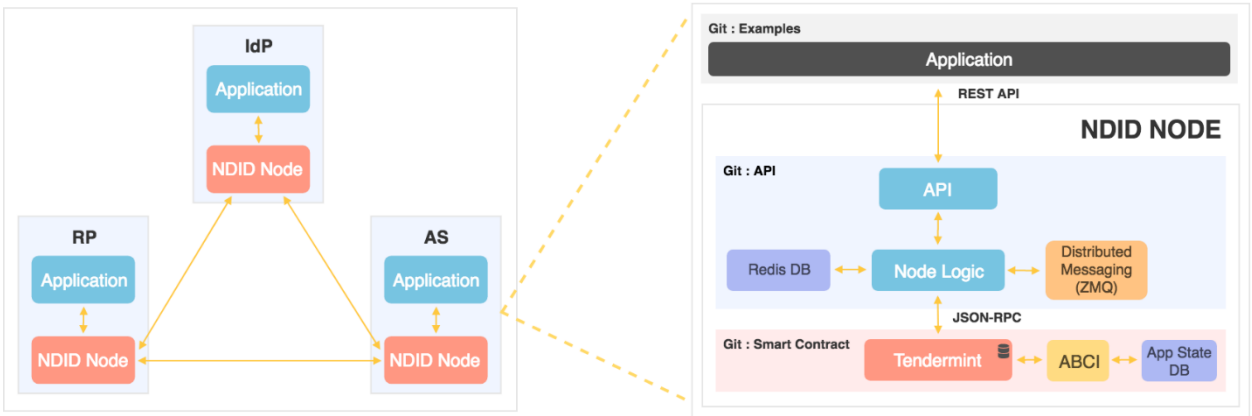
Saturday, October 27, 2018, 10:35 AM

NDID, Government, Identity, Blockchain, Tendermint, Nodejs, Golang, Redis, ZMQ, Opensource

จากโพสต์ที่แล้วที่อธิบายคร่าวๆว่า NDID คืออะไร โพสต์นี้จะอธิบายเชิงเทคนิคว่า NDID มีส่วนประกอบอะไรบ้าง แต่ละส่วนทำงานอย่างไร

ข้อมูลโดยละเอียดหาได้ที่เว็บหลักของข้อมูลทางเทคนิค

ภาพรวมของระบบ NDID



ภาพจาก NDID installation guideline

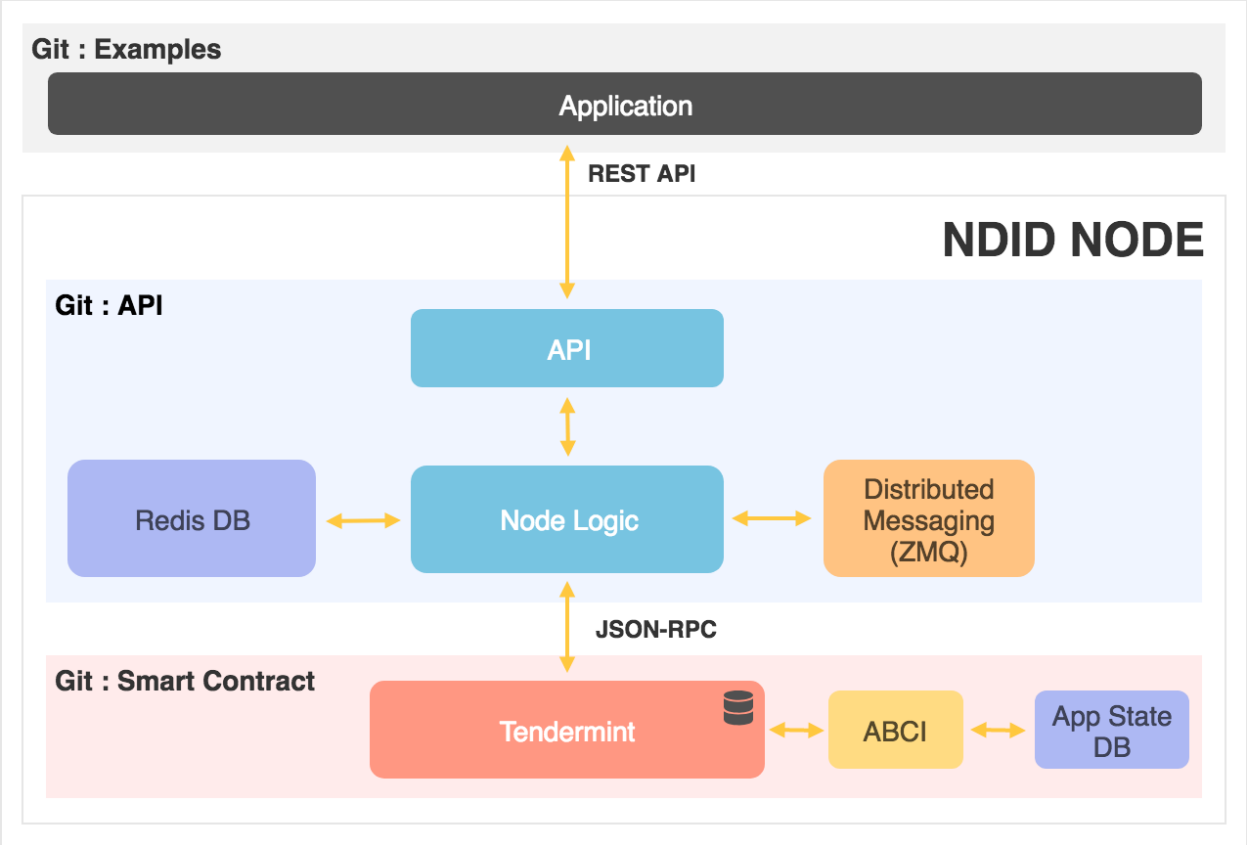
รูปทางด้านซ้ายคือการเชื่อมต่อของสมาชิกแต่ละราย ซึ่งจะเห็นว่าส่วนที่ติดต่อกันคือส่วนของ NDID Node ซึ่ง NDID Node คือซอฟต์แวร์ส่วนที่บริษัท National Digital ID จำกัด จัดทำขึ้นและอนุญาตให้สมาชิกนำไปใช้ได้ (มีแบบ docker ให้ด้วย)

การพัฒนาระบบนี้นั้นจัดทำเป็นแบบ opensource ดูซอร์สโค้ดทั้งหมดที่นี่

ซึ่งเนื่องจากระบบนี้มีภาระผูกพันทางกฎหมาย การจะเข้าเป็นหนึ่งในสมาชิกจึงต้องมีกระบวนการสมัครและเซ็นต์สัญญา รวมถึงต้องยอมรับการ audit ต่างๆตามที่บริษัท NDID กำหนดด้วย

และด้านขวาคือส่วนประกอบของแต่ละโมดูลใน NDID Node

Architecture ของระบบ



ภาพจาก NDID installation guideline

Repositories หลักของระบบนี้ที่แนะนำให้สมาชิกนำไปรันประกอบด้วย

- API ซึ่งเป็นส่วนที่ติดต่อกับแอพลิเคชั่นของสมาชิกผ่านทาง REST API
    - เขียนด้วย Nodejs
- Smart-contract เป็นส่วนที่ sync ข้อมูลสาธารณะกันระหว่างสมาชิกทั้งหมดด้วย blockchain
    - เขียนด้วย golang
    - **ไม่มีการเก็บ sensitive data บน blockchain**

ซึ่งระหว่าง API และ Smart-contract จะเชื่อมกันด้วย JSON-RPC over HTTP และ Websocketนอกเหนือจากสอง repositories หลักด้านบนแล้ว ใน Github ของบริษัทฯยังประกอบด้วย

- test ที่เอาไว้รัน end-to-end ก่อนที่จะ release แต่ละเวอร์ชั่น
- examples ซึ่งจำลองการทำงานเบื้องต้นของแอพลิเคชั่นของสมาชิก
    - ใช้เพื่อง่ายในการสื่อสารให้เห็นภาพในช่วงแรกๆของการพัฒนาระบบ

ซึ่งโพสต์นี้จะอธิบายเฉพาะส่วนของสอง repositories หลักเท่านั้น

คู่ key ที่สมาชิกต้องเก็บรักษา

เมื่อสมาชิกทำการเซ็นต์สัญญากับบริษัท NDID แล้ว สมาชิกจะต้องทำการสร้างคู่ private/public key จำนวน 3 คู่

1. คู่ key ที่ใช้สำหรับประมวลผลภายในของ blockchain และใช้แทนสิทธ์ในการ vote ว่า block หนึ่งๆเป็น block ที่ถูกต้องใน blockchain หรือไม่ (จะเรียกว่า tendermint key)
2. คู่ key ที่ใช้เพื่อยืนยันว่า transaction ในระบบมาจากสมาชิกนี้จริงๆ และใช้เพื่อ encrypt/decrypt ข้อมูลที่ส่งผ่านทาง message queue (จะเรียกว่า node key)
3. คู่ key ที่ไม่ใช้เพื่อการอื่นนอกจากใช้เพื่อเปลี่ยน node key ในข้อ 2 (จะเรียกว่า master key) หรือเพื่อเปลี่ยน master key เองเท่านั้น

สมาชิกจะส่ง public key ของทั้งสามคู่ให้บริษัทเพื่อนำเข้าระบบและประกาศให้สมาชิกอื่นๆทราบ ส่วน private key ทั้งสามคู่สมาชิกต้องเก็บรักษาเป็นความลับ

API Repository

เป็นส่วนที่ติดต่อกับแอพลิเคชั่นของสมาชิกด้วย REST API พัฒนาด้วย Nodejs

- ทำหน้าที่เช็คความถูกต้องของข้อมูลที่สมาชิกส่งลงมาก่อนส่งต่อลงไปยัง blockchain
- ทำหน้าที่จัดการการคำนวณทาง crytography ต่างๆ
- ทำหน้าที่จัดการ Distributed Message สำหรับส่ง sensitive data ระหว่างสมาชิก
- ทำหน้าที่จัดการ cache เพื่อให้ process สามารถทำงานต่อได้หลังจาก restart

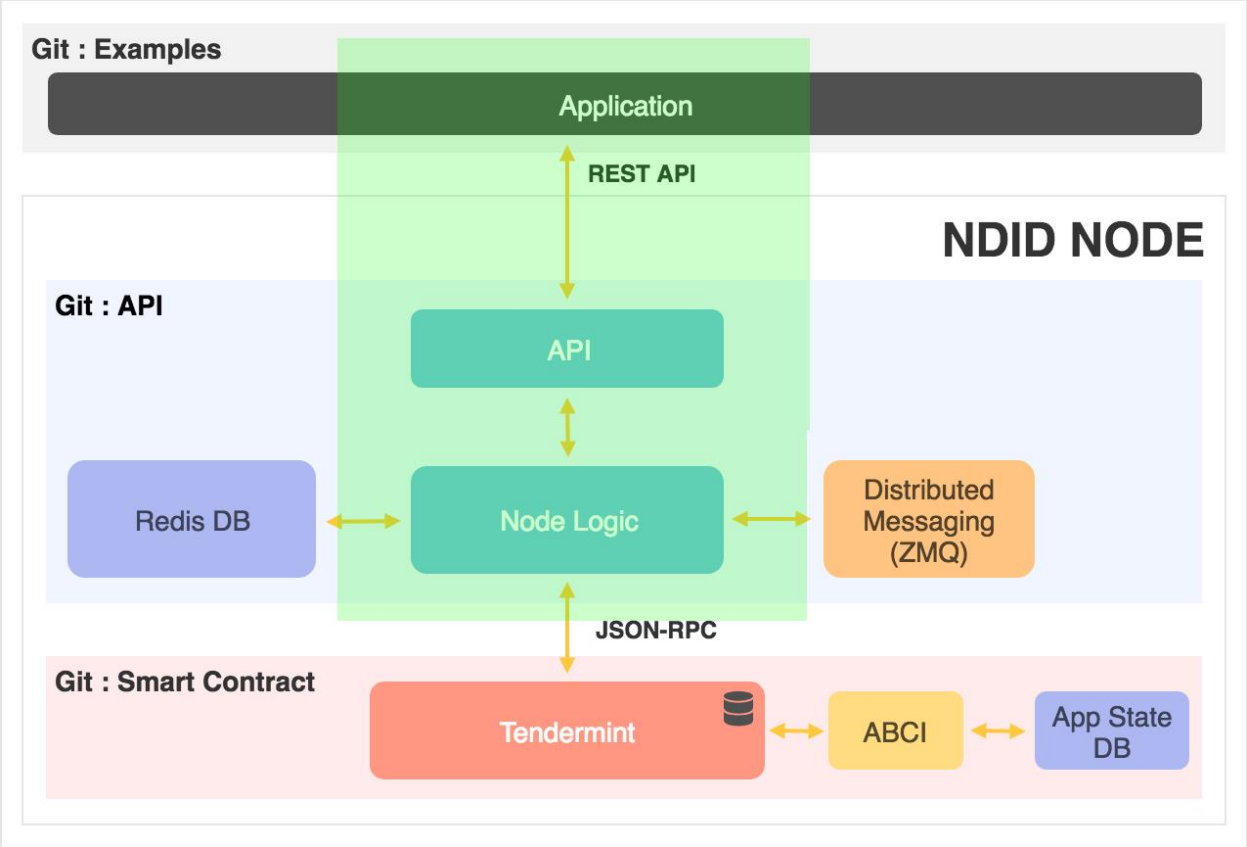
หน้าตาของ Standard API สามารถดูได้ที่นี่

Process ในส่วนนี้จะรันอยู่บนเครื่องของสมาชิก และสมาชิกจะต้องตั้งค่าไฟวอลล์ให้เครื่องรับการเชื่อมต่อ REST API จากเครื่องใน network ที่ไว้ใจได้เท่านั้น (นั่นคือเครื่องที่รันแอพลิเคชั่นของสมาชิก)

ส่วนของที่ API Process เชื่อมต่อกับ Internet จะมีแค่ส่วนของ Distributed Message (ZMQ) ซึ่งอาจจะรันบนเครื่องเดียวกับ API หรือแยกคนละเครื่องก็ได้

สำหรับผลลัพธ์ของ transaction ที่ใช้เวลานาน API process จะส่งคืนผลลัพธ์เหล่านั้นให้สมาชิกทาง callback ที่สมาชิกประกาศไว้กับ process ซึ่งหมายความว่าสมาชิกต้องทำการตั้ง HTTP server ด้วย (แต่เป็น server ภายในที่ถูกเรียกจาก API process เท่านั้น ไม่จำเป็นต้องออก Internet)

API และ Node Logic



ภาพดัดแปลงจาก NDID installation guideline

เป็น HTTP server ที่ให้แอพลิเคชั่นของสมาชิกติดต่อมาเมื่อต้องการทำธุรกรรมผ่าน NDID และสมาชิกจะต้องการตั้งค่า URL เพื่อให้ process ติดต่อกลับไปในกรณีต่างๆ เช่น

- มีการเปลี่ยนแปลงเกิดขึ้นกับธุรกรรมที่สมาชิกเกี่ยวข้อง
- ส่งข้อมูลที่ต้องการให้สมาชิกทำการสร้าง digital signature ด้วย private key ของสมาชิก
- ส่งข้อมูลที่ได้รับจากสมาชิกอื่นผ่านทาง message queue เพื่อให้สมาชิก decrypt ด้วย private key ของตน

Application -> API process

เมื่อได้รับ request จากสมาชิก Node logic จะทำการตรวจสอบรูปแบบข้อมูลว่าตรงตาม API standard ที่ประกาศไว้หรือไม่

หากข้อมูลนั้นถูกต้อง Node Logic จะทำการจัดรูปแบบข้อมูลสาธารณะให้ถูกต้องตามรูปแบบที่จะจัดเก็บลง blockchain และส่งคืนให้สมาชิกทำการสร้าง digital signature เพื่อยืนยัน

เมื่อได้ข้อมูลที่ยืนยันตัวสมาชิกได้จาก digital signature แล้ว Node logic จะทำการส่งข้อมูลไปยัง blockchain เพื่อบันทึกไว้เป็นหลักฐานว่ามีธุรกรรมเกิดขึ้น

หลังจากนั้นหากต้องมีการติดต่อส่ง sensitive data ไปยังสมาชิกรายอื่น Node Logic จะนำ public key ของสมาชิกปลายทางมา encrypt ข้อมูลแล้วส่งต่อทาง message queue

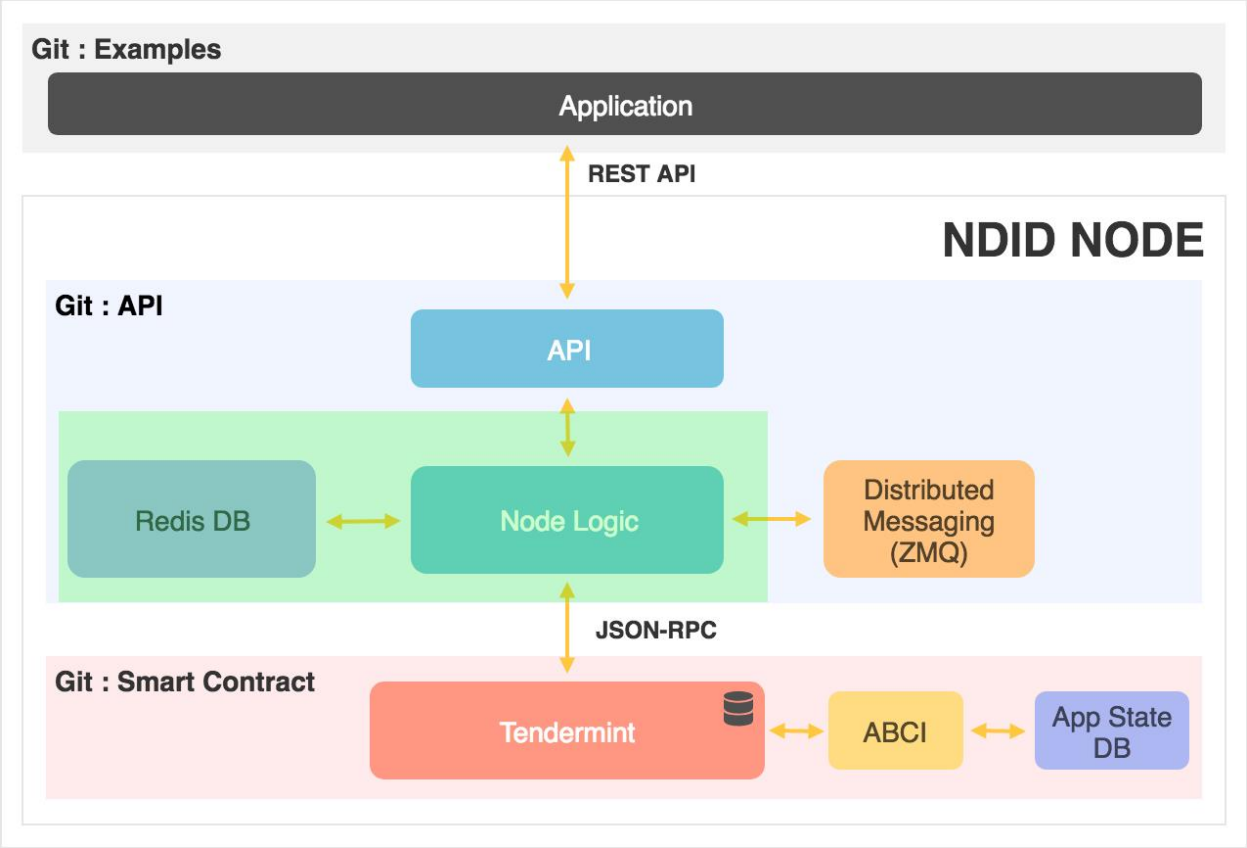
Application <- API process

เมื่อ Node logic ได้รับข้อมูลผ่านทาง message queue แล้ว Node logic จะส่งต่อให้สมาชิกเพื่อทำการ decrypt และอ่านข้อมูล หลังจาก decrypt เก็บพักไว้ใน short term cache (Redis DB)

หาก Node logic มองเห็นเปลี่ยนแปลงจาก blockchain ถึงธุรกรรมที่เกี่ยวข้องกับสมาชิก และมีข้อมูลที่เกี่ยวข้องหลังจาก decrypt อยู่ใน short term cache แล้ว Node logic จะทำการตรวจสอบความถูกต้องของข้อมูล รวมถึงตรวจสอบว่าข้อมูลที่ได้มาจากสมาชิกรายได้ผ่านทาง digital signature

หากข้อมูลถูกต้องทั้งหมด จะทำการส่งต่อให้สมาชิกในรูปแบบที่ได้ประกาศเป็น standard API ไว้

Redis DB



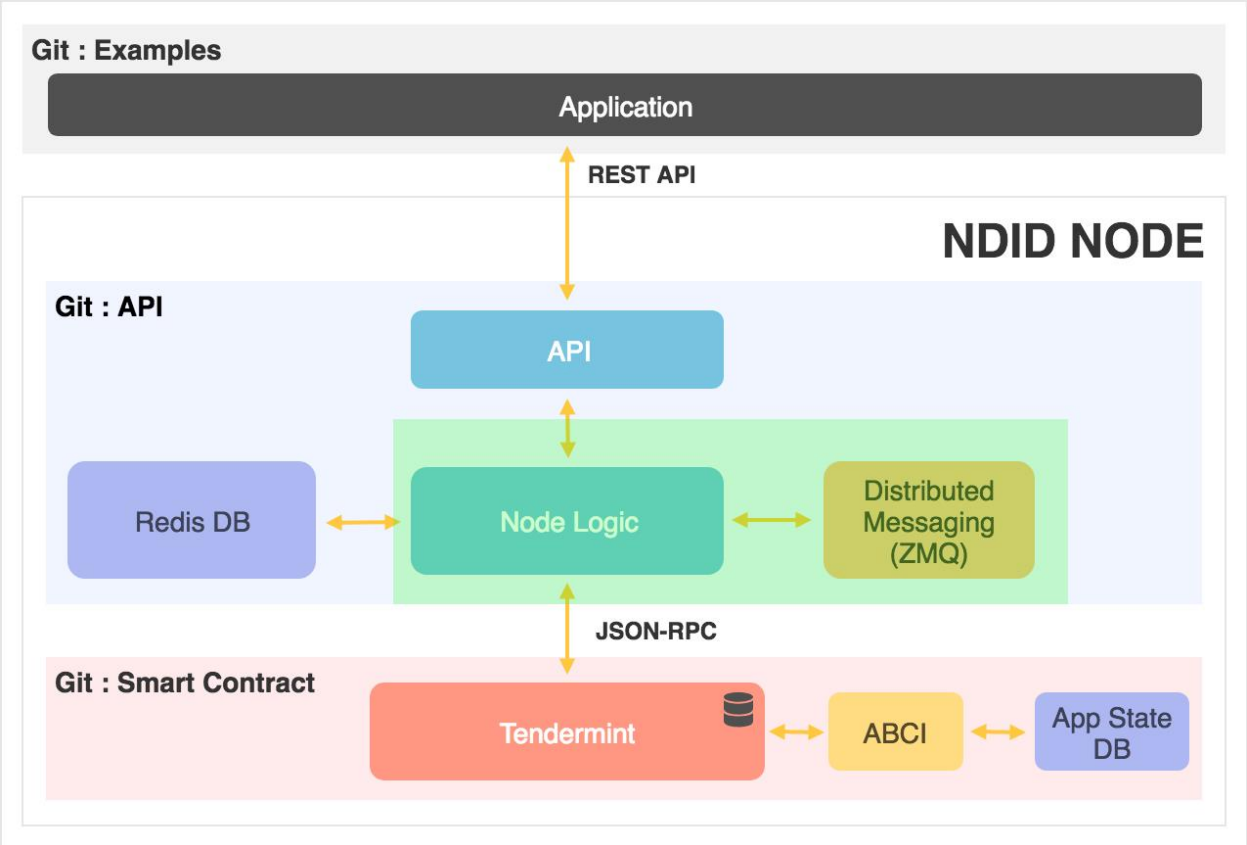
ภาพดัดแปลงจาก NDID installation guideline

ในการทำงานของ Node logic จำเป็นต้องใช้ข้อมูลจากทั้งสองช่องทางนั่นคือ message queue และ blockchain แต่ข้อมูลจากทั้งสองช่องทางอาจจะมาถึงไม่พร้อมกัน จึงต้องมี cache เอาไว้พักข้อมูลที่มาถึงก่อนเอาไว้

รวมถึงเก็บข้อมูลที่ได้จากสมาชิกไว้ใช้ประมวลผลในขั้นตอนถัดๆไป

- short term cache
  - ใช้เพื่อพักข้อมูลที่เกี่ยวข้องจากช่องทางใดช่องทางหนึ่ง
  - ใช้เพื่อเก็บสถานะของ process ว่าประมวลผลถึงขั้นใด สำหรับกรณีที่ process เกิดการ crash แล้ว restart จะได้ยังสามารถทำงานต่อจากจุดเดิมได้
- long term cache
  - ใช้เพื่อพักข้อมูลที่ได้รับและส่งออกผ่านทาง message queue เพื่อเก็บไว้เป็นหลักฐานตามกฎหมาย
  - สมาชิกจะต้องดึงข้อมูลออกและนำไปเก็บในที่ปลอดภัยแล้วจึงสั่งลบข้อมูลในส่วนนี้ออกจาก cache
  - การดึงข้อมูลและลบ สามารถสั่งผ่าน REST API ได้

Distributed Message (ZMQ)



ภาพดัดแปลงจาก NDID installation guideline

- ใช้เพื่อส่ง sensitive data ระหว่างสมาชิก
- เป็นการส่งแบบ P2P
- มีการสร้าง digital signature ของผู้ส่ง และข้อมูลทั้งหมดถูก encrypt ด้วย public key ของผู้รับ
- ข้อมูลใดๆที่ส่งผ่าน message queue จะมีการบันทึกค่า hash ลง blockchain ไว้เป็นหลักฐาน

ย้ำอีกครั้งว่าข้อมูลที่ถูกบันทึกลง blockchain คือ hash ของ sensitive data ไม่ใช่ sensitive data โดยตรง

Smart-contract repository

เป็นส่วนที่จัดการข้อมูลใน blockchain เพื่อให้สมาชิกทั้งระบบมีข้อมูลที่ตรงกัน

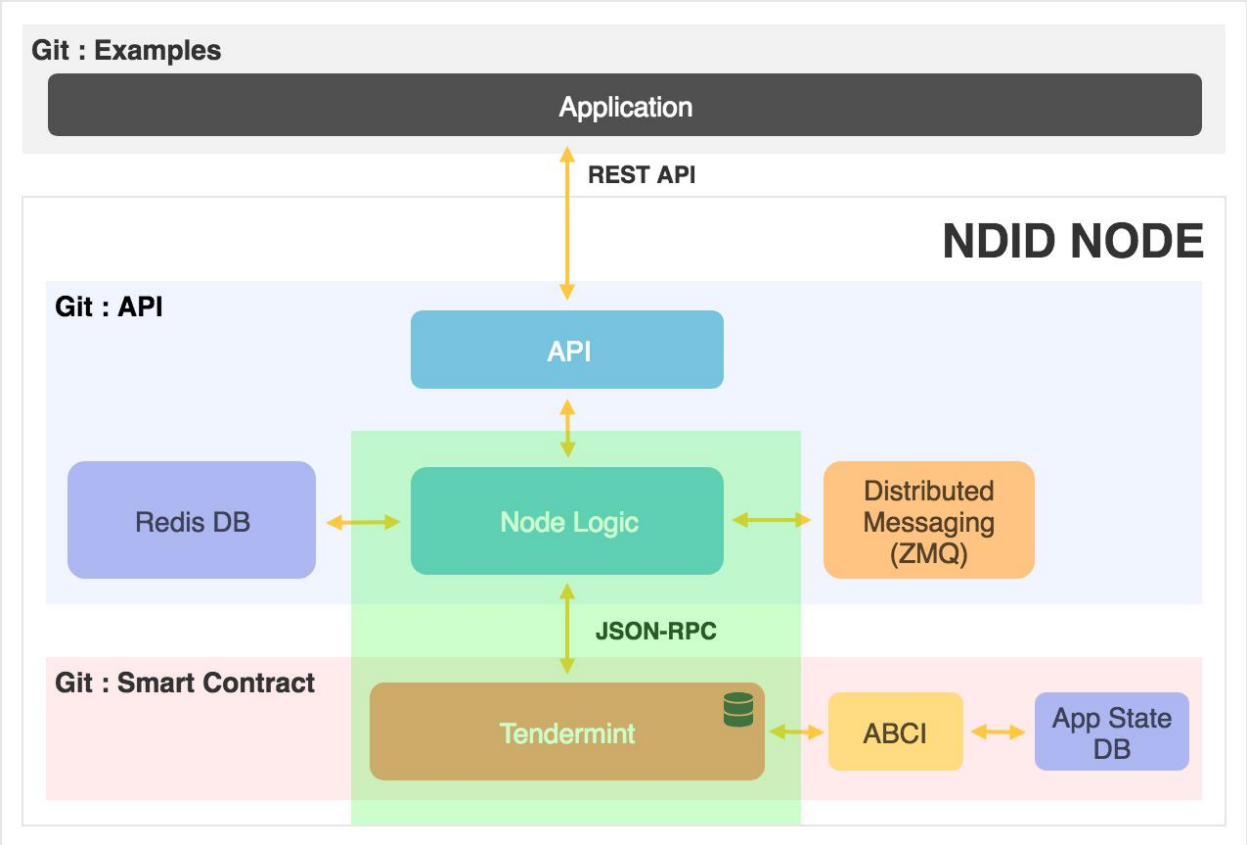
ข้อมูลในส่วนนี้จะเป็นข้อมูลเปิดเผยและ**ไม่มีการเก็บ plain sensitive data ใดๆ ลง blockchain ทั้งสิ้น**

การจะอ่านข้อมูลจาก blockchain นี้ เพียงแค่รู้ genesis file และ chain id ก็สามารถอ่านข้อมูลได้ แต่การจะตั้ง blockchain node ที่สามารถสร้าง transaction ใดๆลง blockchain เพื่อแก้ไขข้อมูล จะต้องได้รับการ approve จากบริษัท NDID ก่อนเท่านั้น (นั่นคือต้องมีการสมัครสมาชิก)

(ยังไม่มีการสรุปว่าบริษัทฯจะเปิดเผย genesis file และ chain id เป็นสาธารณะหรือไม่)

โดย blockchain ที่เลือกนำมาใช้ในระบบนี้คือ [tendermint]

Tendermint



ภาพดัดแปลงจาก NDID installation guideline

Tendermint เป็น light-weight consensus engine ของ blockchain ที่เขียนด้วย golang

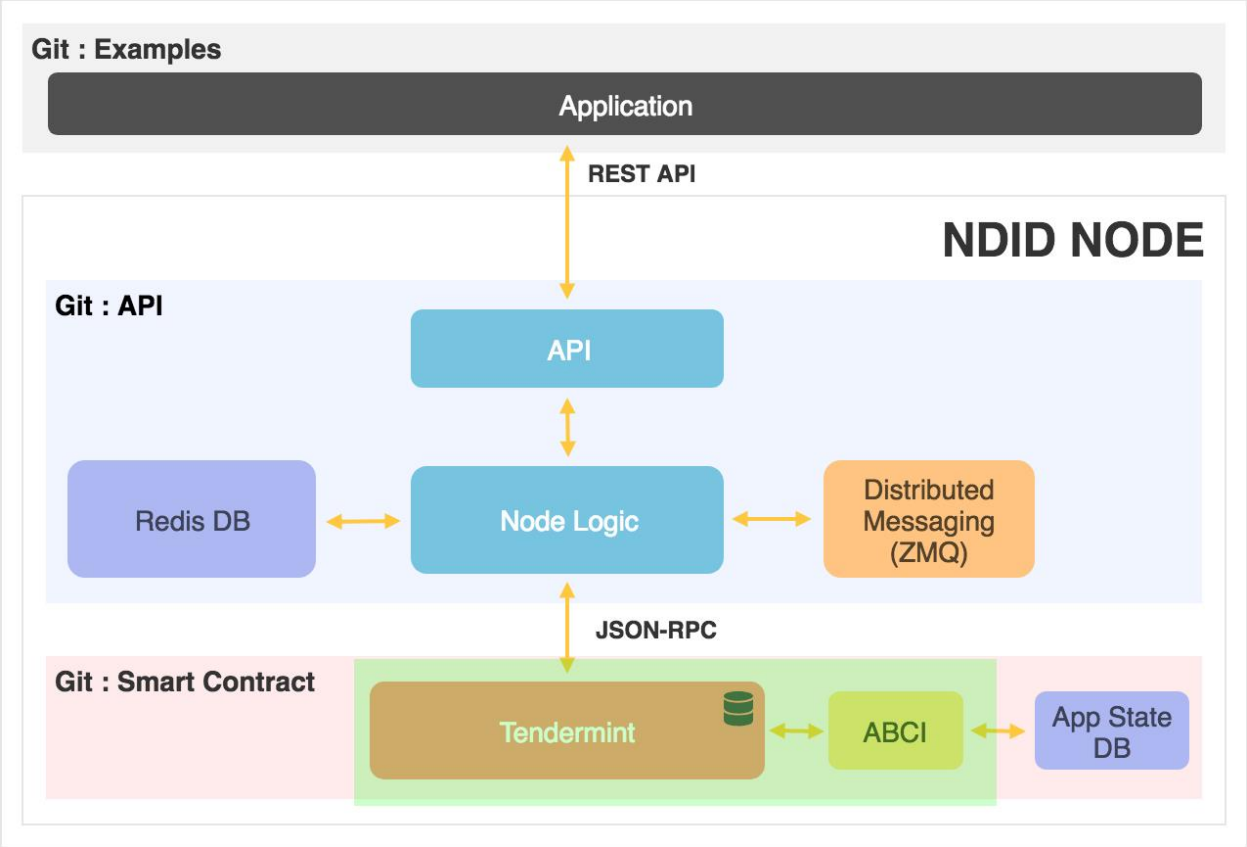
ซึ่งตัว tendermint เองนั้น *ไม่ได้*จำกัดประเภทของ transaction ให้มีเพียงแค่การโอนสิทธิ์ใน Token เหมือนที่อาจจะเห็นในสกุลเงิน crypto อื่นๆ

แต่ช่วยเรื่องการกระจาย transaction ที่เกิดขึ้นและทำการ vote เพื่อ validate block แต่ละ block รวมถึงการ check ว่าสมาชิกแต่ละรายมีข้อมูลที่ตรงกัน และ disconnect สมาชิกที่ข้อมูลไม่ตรง (ไม่มีการ fork)

และสำหรับแต่ละ transaction ที่ Node logic ใน API process ส่งให้ tendermint จะติดต่อกับส่วน ABCI เพื่อตัดสินใจว่า transaction valid หรือไม่ และจะจัดการแก้ไขข้อมูลตาม transaction อย่างไร

- Voting power
    - แต่ละ tendermint node (แยกด้วยคู่ tendermint key) จะมีค่า voting power
    - การที่แต่ละ block จะได้รับการยอมรับ จะต้องได้รับการ vote >= 2/3 ของผลรวมของ voting power ทั้งระบบ
    - สมาชิกแต่ละรายจะได้รับ voting power ตามหลักเกณฑ์ที่บริษัท NDID กำหนด
- Node ID
    - เป็นตัวชี้ถึงองค์กรแต่ละราย โดยแต่ละ ID จะมี public key ประกาศอยู่บน blockchain

ABCI (Application-blockchain interface)

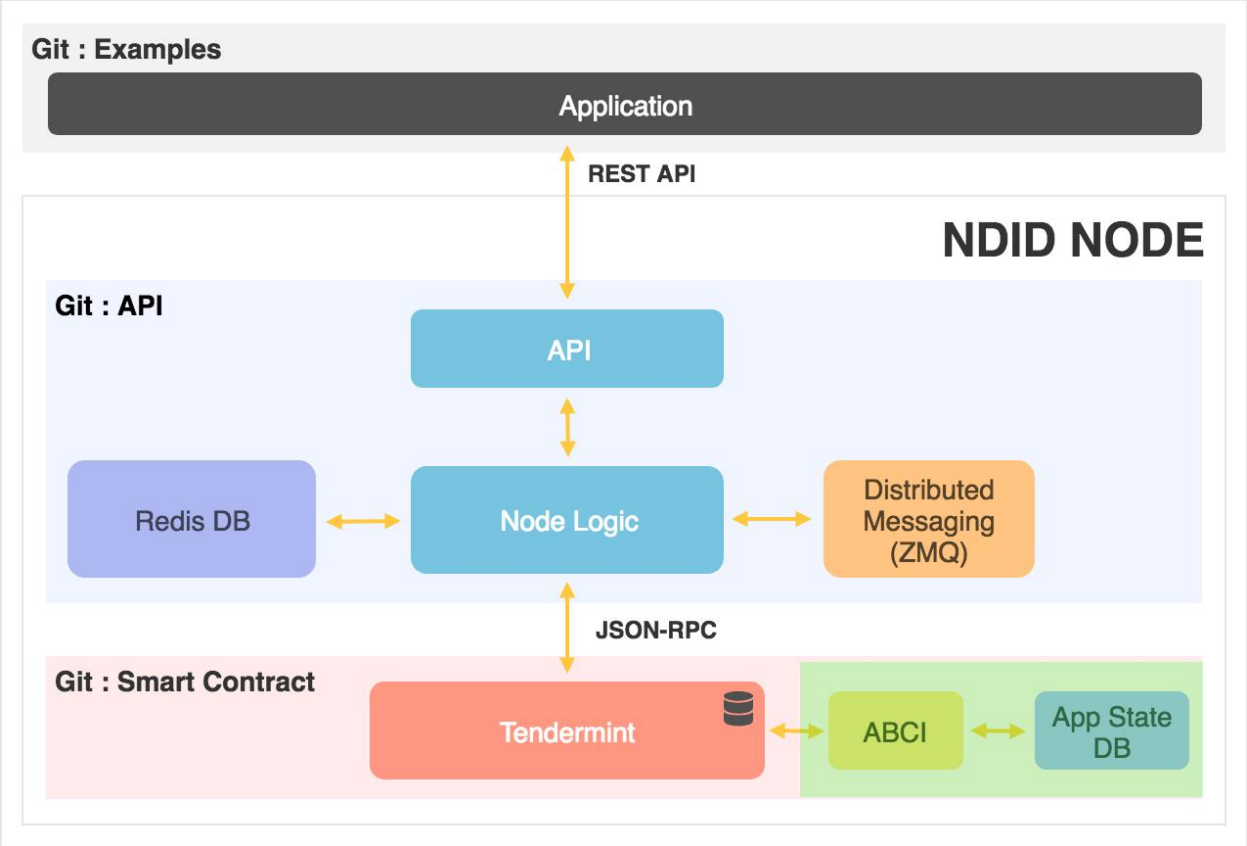


ภาพดัดแปลงจาก NDID installation guideline

เป็นส่วน logic ของ blockchain ที่ติดต่อกับ tendermint เพื่อให้ข้อมูลบน blockchain ของสมาชิกทุกรายนั้นตรงกัน

- ตรวจสอบความถูกต้องของ transaction ที่ทำลง blockchain
- คำนวณ hash ของ data ณ ปัจจุบันเพื่อเปรียบเทียบกับสมาชิกในระบบ
- update และ query ข้อมูลจาก blockchain (levelDB)

App State DB



ภาพดัดแปลงจาก NDID installation guideline

เป็นส่วนที่เก็บข้อมูลใน blockchain โดยใช้งาน levelDB ซึ่งเป็น key-value store

ซึ่ง abci จัดการให้ข้อมูลในนี้เก็บเป็น AVL tree สามารถ query ข้อมูลบนแต่ละ block ได้ ไม่จำเป็นต้อง query เฉพาะ block ล่าสุด

ตัวอย่างการทำงาน

- Onboard ที่ IDP

    - การที่ IdP หนึ่งๆจะสามารถยืนยัน consent ให้ user ใดๆได้ user คนนั้นต้องไปทำธุรกรรมเพื่ออนุญาตก่อน (ดูหมายเหตุ)
        - ในกรณีเป็น IdP แรก จะต้องผ่านการพิสูจน์และยืนยันตัวตนอย่างเข้มงวด
        - ถ้าเกิดมีข้อพิพาทจากความสะเพร่าของ IdP จะมีบทลงโทษจากบริษัท NDID
    - ถ้าเป็น IdP ที่สองเป็นต้นไป การ onboard จะต้องได้รับคำยืนยันจาก user ผ่านทางหนึ่งใน IdP ที่เคย onboard แล้ว
- RP สร้าง request
    - ส่งข้อมูลที่เกี่ยวข้องให้ API process
    - API process คำนวณ hash ของ sensitive data ต่างๆเพื่อบันทึกลง blockchain เป็นหลักฐาน
        - บันทึกลง blockchain พร้อม digital signature เพื่อยืนยันตัวตนสมาชิก
    - นำข้อมูลที่เหลือ encrypt ด้วย public key ของ IdP ปลายทางแล้วส่งทาง zmq
- IDP ได้รับ
    - API process ได้รับข้อมูลทาง zmq
    - API process เรียก callback ไปหา IdP application เพื่อให้ decrypt ด้วย private key
    - API process ตรวจสอบข้อมูลที่ได้ ว่าอยู่ในรูปแบบที่ถูกต้อง และคำนวณ hash เทียบกับข้อมูลใน blockchain
    - หากตรงกันจึงเรียก callback ไปหา IdP application เพื่อให้ขอ consent จาก user
    - เมื่อได้รับ consent จาก user จะส่งหลักฐานการ consent ให้ API process
    - API process คำนวณ hash ของ sensitive data ต่างๆในหลักฐานนั้นเพื่อบันทึกลง blockchain เป็นหลักฐาน
        - บันทึกลง blockchain พร้อม digital signature เพื่อยืนยันตัวตนสมาชิก
    - นำข้อมูลที่เหลือ encrypt ด้วย public key ของ RP แล้วส่งทาง zmq
- RP ได้รับ
    - API process ได้รับข้อมูลทาง zmq
    - API process เรียก callback ไปหา RP application เพื่อให้ decrypt ด้วย private key
    - API process ตรวจสอบข้อมูลที่ได้ ว่าอยู่ในรูปแบบที่ถูกต้อง และคำนวณ hash เทียบกับข้อมูลใน blockchain
        - รวมถึงตรวจสอบความถูกต้องของหลักฐานว่าได้ consent จาก user จริงๆ
    - ถ้าข้อมูลทั้งหมดถูกต้องจึง callback ไปบอก RP พร้อมหลักฐานต่างๆ (หาก RP ต้องการตรวจซ้ำ)
    - RP สามารถทำธุรกรรมต่อไปโดยเปรียบเสมือนว่าได้พิสูจน์และยืนยันตัวตนของ user แล้ว
    - (เพิ่มเติม) ถ้ามีการขอข้อมูล
        - API process จะนำหลักฐานและข้อมูล request ทั้งหมดแล้ว encrypt ด้วย public key ของ AS แล้วส่งทาง zmq

หมายเหตุ

ระบบนี้เริ่มต้นออกแบบด้วยการแบ่งระดับความเชื่อใจใน IdP และความสะดวก 3 ระดับ

Mode 1

- RP,AS ต้องอาศัยความเชื่อใจใน IdP มากที่สุด
- แค่ user เป็นลูกค้าของ IdP นั้นๆ IdP ก็สามารถให้ลูกค้ายืนยันตัวตนได้
- RP เลือกว่าจะให้ user ยืนยันด้วย IdP ไหน

Mode 2

- RP,AS ต้องอาศัยความเชื่อใจใน IdP ปานกลาง
- แค่ user เป็นลูกค้าของ IdP นั้นๆ IdP ก็สามารถให้ลูกค้ายืนยันตัวตนได้
- RP เลือกว่าจะให้ user ยืนยันด้วย IdP ไหน
- user ต้องมีการประกาศเลือก IdP ที่จะได้รับการแจ้งเตือนผ่านทางระบบ onboard
- ต่างจาก mode 1 ที่ธุรกรรมที่เกิดขึ้นต้อง “แจ้งเตือน” ไปยัง IdP ที่ user ได้ประกาศเลือกเอาไว้บนระบบ
    - แต่อาจจะได้รับคำยืนยันจาก IdP อื่นที่ไม่ได้ onboard ไว้

Mode 3

- RP,AS ต้องอาศัยความเชื่อใจใน IdP น้อยที่สุด
- IdP *ต้อง*ได้รับอนุญาตจาก user ผ่านทางระบบ onboard เพื่อได้รับสิทธิในการยืนยันตัวตน
- RP เลือกว่าจะให้ user ยืนยันด้วย IdP ไหน แต่ต้องเป็นหนึ่งใน IdP ที่ user อนุญาตแล้วเท่านั้น
- ต่างจาก mode 1,2 ที่ธุรกรรมที่เกิดขึ้นต้อง “ได้รับคำยืนยัน” จาก IdP ที่ user ได้ประกาศเลือกเอาไว้บนระบบ

ในขั้นต้นระบบจะเปิดใช้ด้วย mode 1 แต่บริษัท NDID จะผลักดันให้สมาชิกค่อยๆโอนย้ายไปยัง mode 3 ทั้งหมด และยกเลิกการรองรับ mode 1 ในที่สุด

ส่วน mode 2 ปัจจุบันที่ประชุมลงความเห็นว่าไม่ควรรองรับ เนื่องจากความสะดวกไม่ต่างกับ mode 3 มากนัก (ต้องมีการ onboard) ดังนั้นให้ทำการยืนยันกับ IdP ที่เลือกไว้แล้วเลยจะเหมาะสมกว่า (ใช้ mode 3)