Bitcoin: A Peer-to-Peer Electronic Cash System

บิทคอยน์:ระเงินสดอิเล็คทรอนิคส์แบบ Peer-to-Peer

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Translation in Thai by Peeraphat Hankongkaew
han@blockchain-review.co.th
www.blockchain-review.co.th

Abstract. เงินสดอิเล็คทรอนิคส์ที่เป็นแบบ peer-to-peer ที่แท้จริงนั้นจะสามารถทำให้การจ่ายเงินออนไลน์จากคนหนึ่งไปอีกคนเกิดขึ้นได้โดยไม่จำเป็นต้องมีสถาบันทางการเงิน เช่น ธนาคาร มาเป็นตัวกลางในการจัดการ Digital Signature เป็นส่วนหนึ่งที่จะทำให้การจ่ายเงินออนไลน์รูปแบบนี้เกิดขึ้นได้ แต่มันคงเป็นไปไม่ได้ถ้าสุดท้ายระบบยังต้องการความน่าเชื่อถือจากตัวกลางต่างๆ เพื่อตรวจสอบและป้องกันข้อผิดพลาดอย่าง Double spending (การใช้เงินซ้ำ) เราขอนำเสนอวิธีการแก้ไขปัญหา Double spending โดยใช้เครือข่ายแบบ peer-to-peer ซึ่งสามารถทำได้โดยเครือข่ายนี้จะทำการบันทึกเวลาในทุกๆธุรกรรมการโอนเงินที่เกิดขึ้นว่าเกิดขึ้นที่เวลาใด รวมถึงทำการ hash ธุรกรรมนี้ธุรกรรมที่โดนเข้ารหัสแล้วเข้าไปต่อกับห่วงโซ่ของธุรกรรมอื่นๆเป็นทอดๆ การจะบันทึกธุรกรรมเหล่านี้ได้จะต้องผ่านการ proof-of-work เท่านั้น ห่วงโซ่ที่ที่ยาวที่สุดไม่ใช่การแค่พิสูจน์ว่ามีคนตรวจสอบกระบวนการแล้วเท่านั้น แต่ยังบอกถึงว่าห่วงโซ่นั้นถูกสร้างจากพลังงานที่มากที่สุดที่มากจากการประมวลผลของ CPU อีกด้วย แต่มันเป็นประกอบไปด้วยบันทึกของธุรกรรมที่ยาวที่สุดคือผลงานที่มาจากกระบวนการ proof-of-work ของ node (คอมพิวเตอร์ในเครือข่าย) หลาย node ที่รวมตัวกันและใช้ CPU ในแก้โจทย์เพื่อยืนยันความถูกต้องของธุรกรรมที่เกิดขึ้น ตราบใดที่กำลัง CPU ของ node ส่วนใหญ่ไม่ได้ถูกนำมาใช้โจมตีระบบ ห่วงโซ่นี้ก็จะยาวไปเรื่อยๆ และจะทำให้ผู้ที่คิดจะโจมตีระบบไม่สามารถสร้างห่วงโซ่ตามทัน ระบบของเครือข่ายนี้ไม่จำเป็นที่จะต้องมีโครงสร้างการทำงานที่ซับซ้อน วิธีการสื่อสารภายในเครือข่ายจะเป็นแบบ best effort (ส่งข้อมูลแบบไม่สนใจว่าจะถึงผู้รับ) แต่ละ node สามารถออกจากเครือข่าย และกลับเข้ามาเมื่อไรก็ได้ เพราะว่าห่วงโซ่ที่ยาวที่สุดจะคอยบอกเสมอว่าเกิดอะไรขึ้นบ้างในช่วงที่ Node เหล่านั้นหายไป

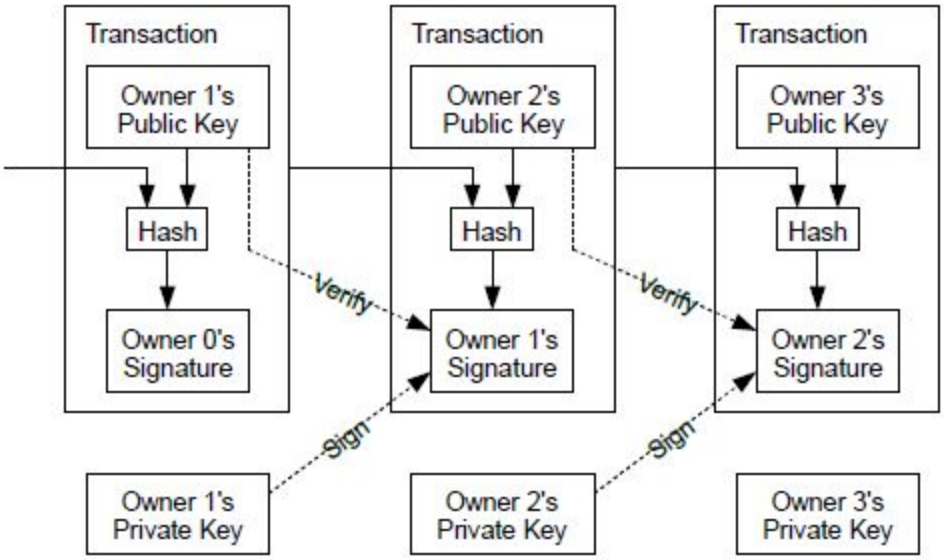
1. บทนำ

โดยทั่วการค้าขายแลกเปลี่ยนในอินเตอร์เน็ตจำเป็นจะต้องพึ่งพาสถาบันทางการเงินเป็นเสมือนบุคคลที่สามที่มีเครดิต หรือมีความน่าเชื่อถือ ในการยืนยันความถูกต้องของธุรกรรมออนไลน์นั้นๆ แม้ว่าระบบตัวกลางนี้จะทำงานได้ดีสำหรับธุรกรรมที่เกิดขึ้นส่วนใหญ่ แต่มันก็ยังมีจุดอ่อนว่าระบบนี้ยังต้องการ trust (เครดิต หรือความเชื่อถือในเชิงของการเงิน) เข้ามาเกี่ยวข้อง การใช้จ่ายผ่านตัวกลางเหล่านี้ มีโอกาสที่ธุรกรรมอาจเกิดความผิดพลาด หรือถูกยกเลิกได้ เนื่องจากสถาบันการเงินที่เป็นตัวกลางอาจจะพบเจอกับปัญหาความขัดแย้งระหว่างตัวกลางกันเอง (เช่น การจ่ายเงินไม่ถูกอนุมัติ เช็คเด้ง หรือ บัตรเครดิตหมดอายุ) การมีตัวกลางนั้นทำให้เกิดค่าใช้จ่ายและต้นทุนในการทำธุรกรรมที่เพิ่มขึ้น และค่าใช้จ่ายนี้ทำให้ผู้ใช้งานไม่สามารถทำธุรกรรมในขนาดเล็กๆได้ (เนื่องจากต้นทุนในการทำธุรกรรมแพงกว่าจำนวนเงินในการทำธุรกรรม) และมันยังมีต้นทุนที่เราอาจจะต้องเสียไป จากโอกาสที่การจ่ายเงินของเราจะไม่สำเร็จ ทำให้เป็นการยากที่เราจะใช้จ่ายเงินให้กับบริการการจ่ายเงินออนไลน์เพื่อแลกกับบริการการจ่ายเงินที่มีโอกาสที่ธุรกรรมอาจจะผิดพลาดรวมถึง ไม่สามารถย้อนกลับคืนได้ (ถ้าเราใช้บริการอย่างหนึ่ง และจ่ายด้วยเชค แล้วเช็คนั้นเด้ง แต่เราได้รับบริการมาแล้ว อาจจะทำให้เราต้องสูญเสียเงิน เวลา และเครดิต) เมื่อการทำธุรกรรมของเรามีสิทธ์ที่จะไม่สำเร็จ ไม่ถูกยืนยัน หรือไม่ถูกอนุมัติโดยตัวกลางที่ดำเนินธุรกรรมนั้น ความน่าเชื่อถือของตัวกลางจึงเป็นเรื่องที่สำคัญมากขึ้นไปอีก แม้แต่ผู้ค้าขายเองก็ต้องพบเจอปัญหามากขึ้น เมื่อลูกค้าสอบถามหาข้อมูลต่างๆที่มากเกินความจำเป็นเพื่อเป็นสิ่งที่ยืนยันความน่าเชื่อถือ (เช่น การขอข้อมูลส่วนตัวของผู้ค้าขาย ซึ่งเป็นสิ่งที่สามารถนำไปต่อยอดเพื่อก่ออาชญากรรมได้) เนื่องจากผู้ซื้อมองว่าการโกงนั้นอาจเกิดขึ้นได้ ซึ่งต้นทุนที่อาจจะเพิ่มขึ้นจากการที่ระบบจ่ายเงินออนไลน์อาจจะผิดพลาดนั้นสามารถถูกแก้ไขได้โดยการใช้เงินสดของสกุลเงินจริงที่จับต้องได้แทน แต่การใช้เงินจริง หรือวิธีไหนก็ไม่สามารถกำจัดตัวกลางได้ในการจ่ายเงินได้ (ตัวกลางสำหรับเงินจริง คือ รัฐบาลที่กำหนดมูลค่าเงิน) และเมื่อยังกำจัดตัวกลางไม่ได้ เราก็จะต้องเสียเงินค่าทำธุรกรรมอย่างไม่จำเป็น อิสระในการทำธุรกรรม และการจับจ่ายเงินออนไลน์จึงถูกจำกัด

สิ่งที่เราต้องการคือระบบการจ่ายเงินแบบอิเล็กทรอนิกส์ที่ตั้งอยู่บน cryptographic แทนที่ใช้ความน่าเชื่อถือตัวกลางทางการเงิน ซึ่งมันจะทำให้ทำให้บุคคลสองคนทำธุรกรรมต่อกันได้โดยไม่ต้องมีตัวกลางมาจัดการ ธุรกรรมที่ถูกยืนยันแล้วจะไม่สามารถถูกย้อนคืนหรือถูกยกเลิกได้ ซึ่งจะช่วยปกป้องผู้ขายจากการหลอกลวงในรูปแบบต่างๆ และระบบการใช้งานสัญญาขายแบบอัตโนมัติแบบไม่มีคนกลาง (escrow) นั้นก็จะถูกพัฒนาขึ้นอย่างง่ายดายเพื่อปกป้องผู้ซื้อ ในเอกสารนี้ เราได้นำเสนอวิธีการป้องกันการใช้เงินซ้ำ (Double spending) โดยการตั้งตัวอยู่ในระบบ peer-to-peer ที่จะลงบันทึกเวลาการเกิดของธุรกรรมนั้น และจะสร้างโจทย์ที่ต้องใช้กำลัง CPU ในการแก้ปัญหาเพื่อยืนยันการเกิดขึ้นจริงและเกิดขึ้นในลำดับเวลาที่ถูกต้อง ไม่ได้ถูกแก้ไข หรือถูกนำมาแทรกแซงจากผู้ต้องการโจมตีระบบ ระบบจะปลอดภัย ถ้า node ส่วนใหญ่ยังใช้กำลังของ CPU เพื่อใช้เพื่อยืนยันการทำธุรกรรม ไม่ใช่ถูกใช้เพื่อหันมาโจมตีระบบเอง

2. ธุรกรรม (Transactions)

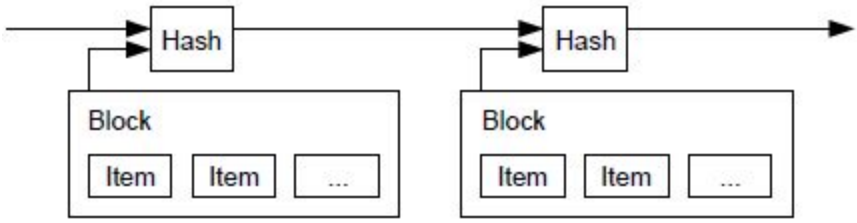
ถ้าเราจะกำหนดให้เหรียญเงินอิเล็คทรอนิคส์นั้นเป็นเหมือนห่วงโซ่ของลายเซ็นดิจิทัล (digital signature) โดยเจ้าของเหรียญแต่ละคนจะส่งเหรียญไปให้คนถัดไปโดยการลงนามแบบดิจิทัล (digitally sigining) ไปในเลข hash ของธุรกรรมที่ผ่านมาและกุญแจสาธารณะ (public key) ของคนถัดไปโดนเพิ่มสิ่งเหล่านี้ลงที่ส่วนสุดท้ายของเหรียญ ซึ่งผู้รับเงินจะสามารถตรวจสอบลายเซ็นเพื่อตรวจสอบห่วงโซ่ของความเป็นเจ้าของ (chain of ownership)



ปัญหาคือผู้รับเงินไม่สามารถตรวจสอบได้ว่า เจ้าเหรียญของไม่ได้นำเหรียญนั้นไปใช้ซ้ำ (double-spend) ซึ่งวิธีที่ทั่วไปในการแก้ปัญหานี้คือการใช้ ตัวกลางในการตรวจสอบที่เชื่อถือได้แบบรวมศูนย์ (trusted central authority) หรือ Mint (ตัวกลาง) ในการตรวจสอบทุก ๆ ธุรกรรมว่ามีการใช้ซ้ำ (double-spend) หรือไม่ ซึ่งในทุกๆธุรกรรมเหรียญนั้นจะต้องกลับไปสู่ตัวกลางในการที่จะสร้างเหรียญใหม่ และเฉพาะเหรียญที่ถูกสร้างจากตัวกลางเท่านั้นที่เราจะเชื่อได้ว่ามันไม่ได้ถูกนำไปใช้ซ้ำ (Double-spend)

ซึ่งปัญหาของวิธีนี้คือทุกอย่างของระบบการเงินจะขึ้นอยู่กับองค์กรตัวกลางที่ทำงานอยู่ ซึ่งทุก ๆ ธุรกรรมจะต้องผ่านตัวกลางนี้เหมือนกับระบบธนาคาร ซึ่งเราต้องการระบบที่ผู้รับเงินสามารถรู้ได้ว่าไม่ได้นำเงินไปใช้ก่อนหน้านี้ ด้วยเหตุผลนี้ เราจะนับธุรกรรมแรกสุดเป็นธุรกรรมแรก ซึ่งเราจะไม่สนใจว่าธุรกรรมนี้จะถูกใช้ซ้ำหรือไม่ ซึ่งวิธีเดียวที่เราจะยืนยันได้ว่าไม่มีธุรกรรมที่หายไปคือการตรวจสอบธุรกรรมทั้งหมด โดยในระบบที่ใช้รูปแบบตัวกลาง ตัวกลางจะคอยดูการทำธุรกรรมทั้งหมดและตัดสินใจว่าอันไหนมาก่อน ในทางกลับกันการจะทำให้มันสำเร็จได้โดยไม่ต้องใช้ตัวกลางที่มีความเชื่อถือ การทำธุรกรรมทั้งหมดจะต้องถูกประกาศแก่สารธารณะ [1] และเราจึงต้องการระบบที่เพื่อให้ผู้เข้าร่วมสามารถตรวจสอบธุรกรรมจากประวัติธุรกรรมได้ทันทีที่ได้รับ โดยผู้รับต้องตรวจในขณะที่ทำธุรกรรมแต่ละครั้ง โดยโหนดส่วนใหญ่จะต้องเห็นด้วยว่านี่เป็นการได้รับเหรียญครั้งแรก+

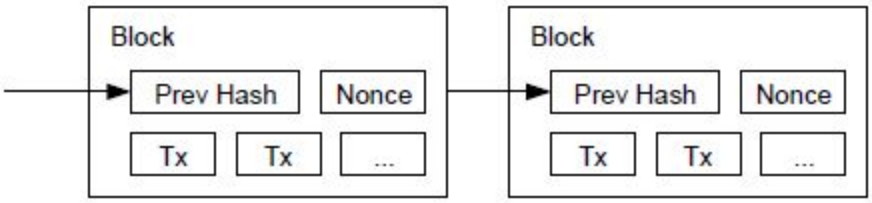
3. เวลาที่บันทึก (Timestamp Server)



ในการแก้ปัญหานี้เราจะใช้ประโยชน์จากจาก timestamp server (เวลาที่ถูกบันทึก) โดย timestamp server ที่บันทึกจะเอามาจากเลข Hash ของ block รายการที่ถูกบันทึกเวลาไว้และจะกระจายเลขนี้ออกไป เหมือนกับระบบของ newspaper หรือ Usenet post (คล้ายกับระบบโพสข้อความที่ทุกคนจะเห็นข้อมูล) [2-5] โดย timestamp จะแสดงให้เห็นว่าข้อมูลนั้นมีเวลาบันทึกไว้ ซึ่งในการที่จะได้เลข Hash นี้ timestamp แต่ละอันจะต้องมี timestamp อันก่อนหน้าเก็บไว้ในรูปแบบเลข hash ของสายโซ่ ซึ่ง timestamp แต่ละอันจะมี timestamp ก่อนหน้านี้เก็บไว้

4. Proof-of-Work

ในการสร้างระบบ distributed (ระบบกระจาย) timestamp server ที่เป็นแบบ peer to peer เราต้องใช้ระบบ proof of-work ซึ่งคล้ายกับ Adam Back's Hashcash [6] (ระบบ Proof of work ที่ใช้ในอีเมล์เพื่อป้องกัน Ddos) แทนที่จะใช้รูปแบบ newspaper หรือ Usenet posts โดย Proof of work จะถูกนำมาใช้ในส่วนของการตรวจสอบค่าที่ถูก Hash เช่นอัลกอริทึ่ม SHA-256 ที่ค่า Hash จะเริ่มต้นด้วยเลข 0 โดย work หรือกำลังประมวลผลที่ใช้จะเพิ่มขึ้นแบบ Exponential ในทุกๆเลข 0 ในการดำเนินการตรวจสอบ



ซึ่งสำหรับเครือข่าย timestamp เราจะสร้าง Proof of work โดยเพิ่มจำนวน Nonce ใน block จนกว่าจะเจอค่าที่ทำให้ Block hash กลายเป็นเลข 0 ซึ่งกำลังประมวลผลที่ CPU ใช้ก็จะมากขึ้นจนกว่าจะเพียงพอกับ Proof of work โดย Block จะไม่สามารถเปลี่ยนแปลงได้โดยปราศจากการประมวลผลซ้ำ เมื่อ Block จะถูกเชื่อมกับสายโซ่หลังจากประมวลผลแล้ว กำลังประมวลผลที่ใช้ในการเปลี่ยนแปลง Block นั้นจะถูกเพิ่มรวมไปกับการประมวลผล Block หลังจากนั้น

ระบบ Proof of work นั้นยังช่วยแก้ปัญหาในเรื่องการหาเสียงส่วนใหญ่ที่จะเป็นสิ่งที่ตัดสินใจในระบบด้วย ถ้าการคัดเลือกเสียงส่วนใหญ่นั้นเป็นระบบ 1 ip 1 vote มันอาจจะมีโอกาสที่ใครซักคนจะสร้าง ip จำนวนมากขึ้นมาเพื่อควบคุมระบบ แต่ระบบ Proof of work นั้นเป็นระบบ 1 CPU 1 vote เสียงส่วนใหญ่ในระบบนั้นจะอ้างอิงกับห่างโซ่ที่ยาวที่สุด ที่ถูกสร้างจากการประมวลผลของ Proof of work ที่มากที่สุดเช่นกัน ถ้ากำลังขุดของ CPU ยังถูกควบคุมโดย node ที่ซื่อสัตย์ ห่วงโซ่ที่ถูกต้องก็เป็นห่วงโซ่ที่สามารถสร้างสายโซ่ที่ยาวที่สุดเมื่อเทียบกับคนอื่นๆที่พยายามสร้างสายโซ่

ในการที่จะแก้ไข Block ที่ผ่านไปแล้ว attacker จะต้องทำการประมวลผล Proof of work ใน Block นั้นใหม่และต้องสร้าง Block ให้เร็วและจนมีห่วงโซ่ที่ยาวกว่าห่วงโซ่ที่ node ที่ซื่อสัตย์สร้างขึ้นซึ่งเราจะอธิบายในภายหลังถึงความเป็นไปได้ที่ Attacker จะสร้าง Block ได้ทันนั้นจะน้อยลง Exponential ในทุกๆ Block ที่เพิ่มขึ้นมาใน Chain และด้วยการที่ความเร็วในการประมวลผล Proof of work ในแต่ละ Hardware นั้นจะแตกต่างกันไปและในระยะยาวกำลังประมวลผลจะเพิ่มขึ้นเรื่อยๆ ค่าความยากในการทำ Proof of work จะถูกกำหนดโดนดูจากค่าเฉลี่ยของจำนวน Block ที่ถูกสร้างขึ้นในแต่ละชั่วโมง ถ้ามันถูกสร้างเร็วเกินไป ค่าความยากก็จะเพิ่มขึ้น

5. Network

ระบบ Network ของ Bitcoin จะทำตามขั้นตอนดั้งนี้

1. เมื่อธุรกรรมเกิดขึ้นมันจะถูกกระจายไปยังทุกๆ โหนด
2. แต่ละ Node จะนำธุรกรรมใหม่ๆเหล่านี้ลงไปเก็บใน Block
3. แต่ละโหนดจะประมวลผลด้วย Proof of work เพื่อหาคำตอบของค่าใน Difficult ใน Block
4. เมื่อ Node สามารถหาคำตอบด้วย proof of work แล้ว node จะกระจาย Block ของตัวเองไปยังทุก node
5. Node ที่เหลือจะยอมรับใน Block ที่ส่งมาถ้าทุกธุรกรรมใน Block นั้นถูกตรวจสอบแล้วว่ามันไม่เคยถูกใช้มาก่อน (Double spending)
6. Node ที่ยอมรับ Block ที่ส่งมาจะสร้าง Block นั้นๆลงบนห่วงโซ่ของตัวเอง โดยใช้เลย Hash ของ Block ที่ส่งมาใช้เป็นเลข previous hash (ที่จะถูกใช้ในการสร้าง Block ถัดไป)

Node ในระบบนั้นจะเชื่อว่าสายโซ่ที่ยาวที่สุดจะเป็นสายโซ่ที่ถูกต้องเสมอและจะสร้าง Block ถัดไปๆจาก สายโซ่นั้นๆ ถ้ามี Node 2 แห่งที่สามารถสร้าง Block ได้พร้อมๆกันและกระจาย Block ไปยัง Node อื่นๆ ซึ่งอาจจะมี Node บางแห่งที่จะได้รับข้อมูลจากจาก Block แห่งใดแห่งนึงก่อน ซึ่งซึ่งในกรณีนี้อาจจะทำให้ Node แต่ละ Node อาจจะมีข้อมูลที่ไม่เหมือนกัน ซึ่งความแตกต่างของข้อมูลของแต่ละ Node นี้จะหายไปเมื่อ Node ใด Node หนึ่งสามารถสายโซ่ได้ยาวกว่า Node ที่มีข้อมูลอีกชุดหนึ่ง และ Node ส่วนใหญ่ก็จะทำงานตาม Node ที่มีสายโซ่ที่ยาวที่สุด

เวลาที่มีธุรกรรมใหม่เกิดขึ้นและกระจายไปในระบบนั้น ธุรกรรมเหล่านั้นไม่จำเป็นต้องกระจายไปถึง Node ทุก Node ตราบใดที่มันยังถูกส่งไปยัง Node ส่วนใหญ่ ถ้ามี Block ใดที่ Node ได้รับแล้วมีการส่งมาซ้ำ Node ก็จะปฏิเสธ Block นั้นๆ แต่ถ้ามีกรณีที่ Node ไม่ได้รับ Block ที่ส่งมา Node นั้นจะขอ Block ที่เขาไม่ได้รับมาจาก Node อื่นๆในตอนที่พวกเขาได้รับ Block ถัดไป เพราะ Node เมื่อ Node รับ Block ใหม่มา Node นั้นจะรู้ว่ามี block ที่หายไป

6. Incentive (แรงจูงใจ, รางวัล)

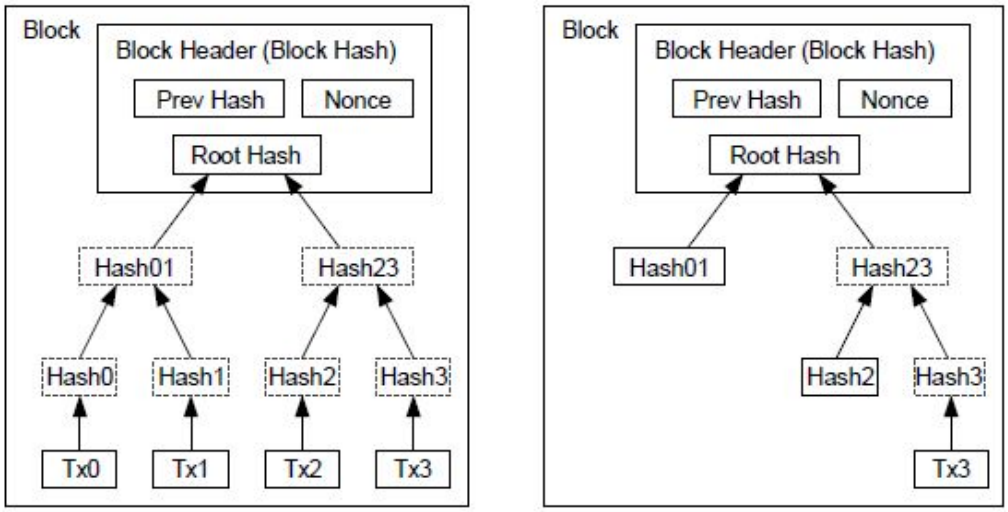
โดยปกติแล้วธุรกรรมแรกใน Block จะเป็นธุรกรรมพิเศษที่จะมอบเหรียญที่ถูกสร้างขึ้นใหม่ให้แก่ผู้ที่สามารถสร้าง Block ได้ การมอบเหรียญที่สร้างขึ้นใหม่นี้เป็นการสร้างแรงจูงใจให้ node ทั้งหลายยังคงสนับสนุนระบบต่อไปและยังเป็นวิธีที่จะแจกจ่ายเหรียญไปในระบบอีกด้วย เนื่องจากในระบบนี้ไม่มีตัวกลางที่จะผลิตเหรียญเพิ่มขึ้น การเพิ่มขึ้นของเหรียญที่ถูกสร้างขึ้นใหม่นี้จะคล้ายคลึงกับการขุดทองที่นักขุดจะต้องมีต้นทุนในการการเพิ่มทองใหม่ๆที่ไม่เคยเจอมาก่อนเข้าไปในระบบ ซึ่งในกรณีนี้คือเวลาของ CPU และพลังงานไฟฟ้าที่จะต้องใช้เป็นต้นทุน

ซึ่งรางวัลนี้ยังรวมถึงค่าธรรมเนียมของธุรกรรมอีกด้วย โดยถ้ามูลค่าของธุรกรรมนั้นมีค่าน้อยกว่าจำนวนเงินที่ใส่เข้ามาในธุรกรรมตอนแรก จำนวนเงินที่เป็นส่วนต่างนั้นคือค่าธรรมเรียมที่จะถูกเพิ่มเป็นรางวัลเพิ่มเติมที่จะอยู่ใน Block ที่มีธุรกรรม ถ้าเหรียญที่ใหม่ๆที่ถูกสร้างขึ้นมาถูกสร้างจนหมดแล้ว รางวัลที่เหลืออยู่ก็คือค่าธรรมเนียมของธุรกรรมต่างๆ และระบบก็จะไม่มีการสร้างเหรียญใหม่ๆอีก ซึ่งรางวัลเหล่านี้เป็นสิ่งที่ช่วยกระตุ้นให้ node ทั้งหลาย

ยังทำงานอย่างซื่อสัตย์ ถ้ามี Attacker ที่มีกำลัง Cpu มากกว่า node อื่นๆที่ทำงานถูกต้องเขาก็ต้องเลือกว่าเขาจะโกงทุกคนในระบบโดยการทำให้ธุรกรรมของเขาไม่เคยเกิดขึ้น หรือใช้กำลังประมวลผลที่เขามีในการสร้างเหรียญใหม่ๆ ซึ่งเขาอาจจะค้นพบว่าเขาจะได้ผลประโยชน์มากกว่าถ้าเขาช่วยเหลือระบบ เช่นถ้าเขานำกำลังที่มีมากกว่าคนทุกคนมาช่วยระบบและได้เหรียญใหม่ๆ เขาจะได้กำไรมากกว่านำกำลังเหล่านั้นไปทำให้แค่นำเงินของเขาที่เคยใช้ไปกลับคืนมา

7. Reclaiming Disk Space

เมื่อธุรกรรมสุดท้ายนั้นถูกใส่ลงใน Block แล้ว ธุรกรรมที่ถูกใช้ไปแล้วอาจจะถูกนำออกไปได้เพื่อประหยัดพื้นที่ในการจัดเก็บข้อมูล ซึ่งการที่จะทำสิ่งนี้ได้โดยไม่ได้ที่ให้เลข Hash ของ Block เปลี่ยนนั้น ธุรกรรมต่างๆจะต้องถูก Hash ในรูปแบบของ Markle Tree [7][2][5] (การคู่ข้อมูลแล้ว Hash ขึ้นมาเป็นลำดับขั้นสูงขึ้นเรื่อย ๆ แบบ Tree) ที่เฉพาะ Root (ส่วนแรกของการเก็บข้อมูลแบบ Tree) เท่านั้นที่จะมีเลข hash ของ Block สำหรับ Block เก่าๆนั้นอาจจะสามารถบีบอัดข้อมูลได้โดยนำตัดกิ่งของข้อมูลออกจาก Tree ซึ่งกิ่งนั้นไม่จำเป็นต้องมีเลข Hash เก็บไว้



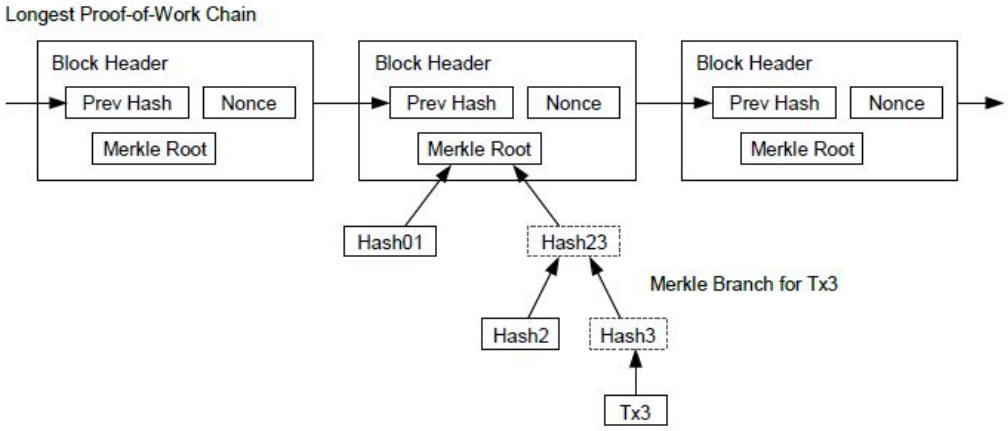
Transactions Hashed in a Merkle Tree

After Pruning Tx0-2 from the Block

ในส่วน Header (ข้อมูลส่วนแรกที่ถูกเก็บใน Block) ของ Block ที่ไม่มีธุรกรรมนั้นจะมีขนาดประมาณ 80 bytes ถ้าทุก Block นั้นถูกสร้างขึ้นในเวลาทุกๆ 10 นาทีก็เท่ากับ 80 bytes * 6 * 24 * 365 = 4.2MB ต่อปีด้วยระบบคอมพิวเตอร์ที่ปกติจะมีขนาดของ RAM ที่ 2GB ในปี 2008 และตามกฎของ Moore (กฎของมัวร์อธิบายถึง ปริมาณของทรานซิสเตอร์บนวงจรรวม โดยจะเพิ่มเป็นเท่าตัวประมาณทุก ๆ สองปี) ซึ่งประมาณการณ์ไว้ว่ามันจะเพิ่มขึ้นปีละ 1.2 GB ทำให้ขนาดพื้นที่ของการเก็บข้อมูลนั้นไม่ใช่ปัญหาแม้ว่า Header ของ Block จะถูกเก็บใน Memory

8. Simplified Payment Verification (การตรวจสอบธุรกรรม)

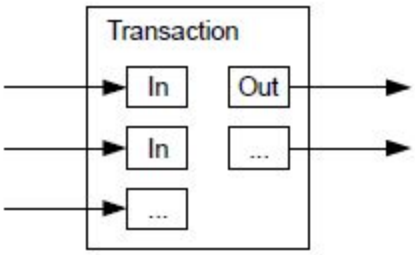
การตรวจสอบธุรกรรมนั้นสามารถทำได้โดยไม่จำเป็นต้องเป็น Full node (node ที่มีฟังชั่นทุกอย่างของระบบครบถ้วน) ผู้ใช้งานเพียงแค่เก็บสำเนา Header ของห่วงโซ่ที่ยาวที่สุด ซึ่งผู้ใช้งานสามารถหาได้จากการดึงข้อมูลจาก node ใน Network จนกว่าเขาจะได้ห่วงโซ่ที่ยาวที่สุด และได้กิ่งของ Markle ที่เชื่อมธุรกรรมไปยัง Block ที่มีการบันทึกเวาไว้ (timestamped) โดยปกติผู้ใช้งานจะไม่สามาระตรวจสอบธุรกรรมด้วยตัวเองได้ แต่เมื่อเชื่อมข้อมูลไปยังห่วงโซ่ ผู้ใช้งานจะสามารถเห็นได้ว่า Node มีการรับธุรกรรมและสร้าง Block เพิ่มทำให้ผู้ใช้งานสามารถยืนยันได้ว่าเนตเวิร์คมีการยืนยันธุรกรรมแล้ว



การตรวจสอบและยืนยันธุรกรรมนั้นจะสามารถเชื่อถือได้ตราบเท่าที่ node ที่ซื่อสัตย์ยังควาบคุมเน็ตเวิร์คอยู่ แต่ระบบก็อาจจะอ่อนแอลงได้ถ้าถูกโจมตีจาก Attacker ที่กำลังมาก ในขณะที่ node สามารถตรวจสอบธุรกรรมได้ด้วยตนเอง วิธีที่ Attacker สามารถหลอกระบบได้คือสร้างธุรกรรมปลอมๆขึ้นมาตราบเท่าที่พวกเขามีกำลังส่วนมากในระบบ และวิธีที่จะป้องกันได้คือการที่ node อื่นๆคอยแจ้งเตือน node อื่นๆเมื่อเขาได้ Block ที่ไม่ถูกต้อง เป็นการกระตุ้นผู้ใช้งานให้ download ข้อมูล block ทั้งหมดเพื่อคอยแจ้งเตือนคนอื่นๆเมื่อเกิดธุรกรรมขึ้นเพื่อป้องกันความผิดพลาด ธุรกิจที่มีจำนวนการจ่ายเงินมากๆอาจจะต้องมี Node เป็นของตัวเองเพื่อความปลอดภัยและการตรวจสอบธุรกรรมที่รวดเร็วขึ้น

9. Combining and Splitting Value การรวมและการแบ่งมูลค่า

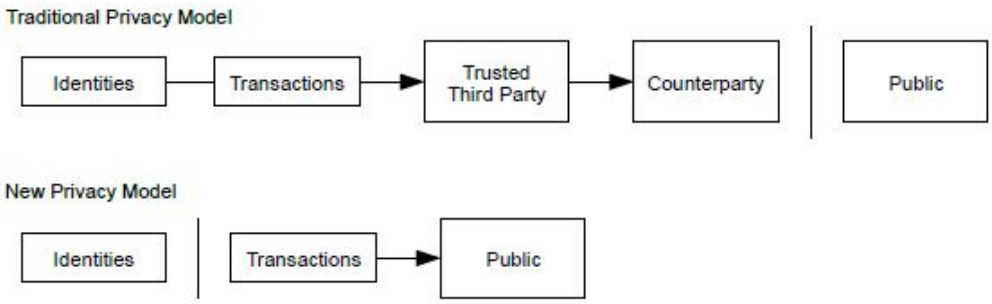
ถึงแม้ว่าระบบสามารถจัดการเหรียญและธุรกรรมได้อย่างอิสระ แต่มันคงจะดูแปลกๆถ้าเราต้องแตกธุรกรรมออกเป็นหลายๆธุรกรรมในเงินทุกส่วนที่เราจะส่ง การที่เราจะทำให้ค่าของเหรียญสามารถแตกธุรกรรมหรือรวมธุรกรรมได้ ธุรกรรมจะต้องมีจำนวนผู้ส่งและผู้รับที่หลากหลาย ปกติแล้วมันมีความเป็นไปได้ที่จะมีเงินที่มีค่ามากในธุรกรรมที่ผ่านมาแล้วหรืออาจจะเป็นเงินก้อนที่เกิดจากการรวมเงินก้อนเล็กๆหลายๆจำนวนซึ่งปกติจะมีธุรกรรมอย่างมากสองธุรกรรม คือส่วนที่เราจ่ายเงิน และส่วนที่เราจ่ายเป็นเงินทอน



ซึ่งวิธีนี้เป็นวิธีแบบ fanout เมื่อธุรกรรมขึ้นกับธุรกรรมย่อยๆจำนวนมาก และธุรกรรมเหล่านั้นยังขึ้นกับธุรกรรมอื่นๆจำนวนมากเช่นกัน ก็จะไม่เกิดปัญหาขึ้น การแตกธุรกรรมจึงไม่ใช่สิ่งที่จำเป็น

10. Privacy ความเป็นส่วนตัว

ในรูปแบบธนาคารทั่วไปในเรื่องความเป็นส่วนตัวนั้นจะมีข้อจำกัดอยู่เนื่องจากตัวกลางผู้ให้บริการนั้นมีสิทธิที่จะเข้าถึงข้อมูลของเรา การประกาศทุกธุรกรรมสู่สารธารณะจึงดูเหมือนว่าเป็นสิ่งที่ขัดแย้งกับความเป็นส่วนตัว แต่ด้วยวิธีนี้การที่จะคงความเป็นส่วนตัวนั้นยังสามารถทำได้โดยการนำข้อมูลบางส่วนออกไปจากกระบวนการของระบบ และวิธีนั้นคือการทำให้ Public key นั้นกลายเป็นสิ่งที่ไม่มีการระบุตัวตนลงไป (เนื่องจากไม่มีการใส่ข้อมูลส่วนตัวใดๆลงใน public key) ถ้ามองจากมุมมองข้างนอกทุกคนจะสามารถเห็นได้ว่ามีใครบางคนกำลังส่งเงินให้กับอีกคนหนึ่ง แต่ public key จะไม่มีการระบุข้อมูลส่วนตัวที่จะเชื่อมโยงธุรกรรมไปบุคคลใดๆ วิธีนี้มันก็คล้ายๆกับวิธีที่การรักษาข้อมูลของตลาดหลักทรัพย์ ว่าจะมีการเทรดหรือแลกเปลี่ยนเมื่อไหร่แต่จะไม่บอกว่าใครเป็นกลุ่มคนนั้น



และเพื่อความปลอดภัยที่มากขึ้น ผู้ใช้งานควรสร้าง Public key (address) ใหม่ทุกครั้งในทุกๆการโอนเพื่อไม่ให้มีข้อมูลที่ละเชื่อมโยงมาสู่ผู้ใช้ แต่มันก็จะมีข้อมูลเชื่อมโยงบางอย่างที่ไม่สามามารถปกปิดได้เช่นธุรกรรมที่มีจำนวนเงินจากหลายๆแห่ง ซึ่งมันอาจจะมีข้อมูลที่เชื่อมโยงว่าเงินเหล่านั้นมาจากคนๆเดียวกัน ซึ่งความเสี่ยงนั้นคือหากมีข้อมูลที่เปิดเผยว่าคนๆนั้นเป็นเจ้าของ Public key มันอาจจะเชื่อมโยงไปยังธุรกรรมอื่นๆอีกว่ามันเป็นของคนๆนั้น (เช่นเรารู้ว่า Address 1 อย่างมีใครเป็นเจ้าของเราอาจจะสืบต่อไปว่าเขาได้เงินจากใครหรือส่งเงินให้ใครบ้าง)

11. Calculations (การคำนวณ)

ถ้าเรามองว่าการโจมตีของ Attacker คือการที่เขาสามารถสร้างห่วงโซ่ได้เร็วกว่าห่วงโซ่ที่ถูกต้อง แม้ว่าเขาจะทำสำเร็จ มันก็ไม่ได้แปลว่าเขาจะสามารถเปลี่ยนแปลงและแก้ไขระบบได้ตามใจชอบ เช่น Attacker ก็ไม่สามารถเสกเงินขึ้นมาจากอากาศหรือหรือขโมยเงินที่ไม่เคยเป็นของ Attacker ได้ เนื่องจาก node อื่นๆจะไม่ยอมรับธุรรกรรมที่ไม่ถูกต้อง (เช่นเราเขียนว่ามีเงินเข้ากระเป๋าเราหรือเอาเงินจากคนอื่นเข้ากระเป๋าเราแต่ Digital signature นั้นไม่ถูกต้อง Node อื่นก็จะไม่รับ) สิ่งเดียวที่ Attacker สามารถทำได้คือการที่สามารถแก้ไขว่าธุรกรรมและเงินที่เขาเพิ่งใช้ไปนั้นไม่เคยเกิดขึ้นมาก่อน

ซึ่งการแข่งขันในการสร้างห่วงโซ่ที่ถูกต้องและห่วงโซ่ของ Attacker จะเป็นลักษณะของ Binomial Random Walk ที่ทุกครั้งที่ห่วงโซ่ที่ถูกต้องสร้าง Block ได้ความน่าจะเป็นจะ +1 และถ้าห่วงโซ่ของ Attacker สร้าง Block ได้ความน่าจะเป็นจะ -1

โอกาสที่ Atacker จะสร้างห่วงโซ่ได้ทันโดยดูจากจำนวน block ที่ขาดนั้นขึ้นอยู่กับ Gambler's Ruin problem สมมติว่ามีนักพนันที่มีชิปไม่จำกัดและเขาพยายามพนันนับครั้งไม่ถ้วนเพื่อให้ถึงจุด Breakeven หรือก็คือจุดที่ Acttacker จะสร้างห่วงโซ่ทันห่วงโซ่ที่ถูกต้อง[8]

p = ความน่าจะเป็นที่ Node ที่ซื่อสัตย์จะเจอ Block ถัดไป แล้ว
q = ความน่าจะเป็นที่ Attacker จะเจอ Block ถัดไป
qz = ความน่าจะเป็นที่ Attacker จะสร้าง Block ทันห่วงโซ่ที่ถูกต้องโดย z คือจำนวน Block ที่ตามกัน

ถ้าเราตั้งสมมติฐานว่า p>q หมายความว่า ความน่าจะเป็นจะตกลงแบบ Exponential ในทุก Block ที่ Acttacker ต้องตามให้ทัน ซึ่งมันทำให้ Attacker นั้นเสียเปรียบ ถ้าเขาไม่ได้ดวงดีสามารถสร้าง Block ได้มากกว่าห่วงโซ่ที่ถูกต้องตั้งแต่แรก โอกาสที่เขาจะตามทันจะน้อยมากเมื่อเขาตามหลังห่วงโซ่ที่ถูกต้อง ถ้าเราลองพิจารณาว่าผู้รับจะต้องรอนานเท่าไหร่ถึงจะมั่นใจได้ว่าธุรกรรมที่มาจากผู้ส่งจะเปลี่ยนแปลงไม่ได้ ถ้าเราลองคิดว่า Attacker คือผู้ส่งเงินที่ต้องการทำให้ผู้ที่รับเงินเชื่อว่าเขาส่งเงินแล้วหลังจากนั้น ค่อยทำให้ธุรกรรมนั้นถูกเขียนทับหลังจากเวลาผ่านไป แม้ผู้รับจะรู้ตัวหลังจากนั้นแต่ Attacker ก็จะหวังว่ามันคงจะสายเกินไป

ถ้าผู้รับสร้าง Public key ใหม่และเอามันให้แก่ผู้ส่งในทันทีก่อนที่จะมีการ Sign (การยืนยันธุรกรรมจาผู้ส่ง) มันจะช่วยป้องกันความเสี่ยงที่ผู้ส่งจะสร้างห่วงโซ่ปลอมๆเตรียมไว้ หากเขาโชคดีพอที่จะสร้างห่วงโซ่ได้ทันและคิดจะส่งห่วงโซ่ได้ทัน เมื่อธุรกรรมถูกส่งไปแล้ว attacker ที่เป็นผู้ส่งก็จะเริ่มสร้างห่วงโซ่อีกสายหนึ่งที่มีข้อมูลไม่เหมือนกับธุรกรรมที่เขาส่งไปในตอนต้น เมื่อผู้รับรอจนธุรกรรมของเขาถูกต่อไป z block ซึ่งเขาไม่รู้ว่า attacker นั้นสามารถสร้าง block ของตัวเองไปถึงเท่าไหร่แล้ว ซึ่งเราจะลองสมมติว่าถ้า Block ที่ถูกสร้างอย่างถูกต้องนั้นใช้เวลาในการสร้างตามค่าเฉลี่ย ประสิทธิภาพในการสร้าง Block ของ Attacker จะเป็นการแจกแจงความน่าจะเป็นแบบปัวซอง (Poisson distribution)

$$\lambda = z\frac{q}{p}$$

ในการที่จะได้ค่าความน่าจะเป็นที่ attacker จะสร้าง block ตามได้ทัน เราต้องคูณค่า poisson ในทุกกระบวนการที่ attacker สร้างด้วยความน่าจะเป็นที่เขาจะสามารถสร้าง block ตามได้ทันจากจุดๆนั้น

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

จัดเรียงสูตรเพื่อป้องกันการบวกกันของค่า infinite ในการกระจาย

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

เปลี่ยนเป็นภาษา C

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

คำนวณค่า แล้วเราจะเห็นว่าความเป็นไปได้นั้นจะตกลงแบบ exponential ด้วยค่า z

```
q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012

q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

หาคำตอบถ้า P น้อยกว่า 0.1%

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

12. Conclusion สรุป

เราพยายามสร้างระบบสำหรับธุรกรรมอิเล็คทรอนิคส์ที่ไม่ต้องใช้ความน่าเชื่อถือใดๆ เราเริ่มจากการทำโครงของเหรียญที่สร้างจาก Digital signature (ลายเซ็นต์ดิจิทัล) ซึ่งทำให้มันมีจุดแข็งในด้านการเป็นเจ้าของ (ownership) แต่ระบบนี้คงจะไม่สำเร็จหากไม่สามารถป้องกัน Double spending ได้ ซึ่งในการแก้ไขในจุดนี้เราจึงขอนำเสนอเครือข่ายแบบ peer-to-peer ที่ใช้ระบบ Proof of work ในการบันทึกธุรกรรมแบบสารธารณ ซึ่งมันจะกลายเป็นระบบที่ที่มีกำลังประมวลผลมากจน Attacker ไม่สามารถทำอะไรได้ถ้า Node ที่ซื่อสัตย์ยังควบคุมกำลังของ CPU ส่วนใหญ่อยู่

ระบบนี้จะมีความแข็งแรงแม้จะมีโครงสร้างที่ซับซ้อน node ทั้งหมดจะทำงานด้วยกันจากการประสานงานเล็กๆ การระบุตัวตนจะไม่เกิดขึ้นเพราะว่าข้อความหรือธุรกรรมที่ถูกส่งออกไปนั้นไม่ได้พุ่งไปที่จุดใดจุดหนึ่ง และข้อความแจะถูกส่งออกไปให้ถึง node มากที่สุดเท่าที่จะทำได้ node สามารถออกหรือเข้าร่วมระบบเมื่อไหร่ก็ได้ตามที่ต้องการ เพราะห่วงโซ่ที่เกิดจาก proof of work จะเป็นสิ่งที่บอกว่าเกิดอะไรขึ้นบ้างในช่วงที่ node หายไป ระบบนี้จะโหวตตามกำลัง CPU โดยดูจากการที่ node รับ Block ที่ถูกต้องและต่อเข้ากับห่วงโซ่และปฏิเสธ Block ที่มีข้อมูลไม่ถูกต้อง ซึ่งกฎและรางวัลต่างๆจะถูกใช้งานในรูปแบบ Consensus (ฉันทามติ)

อ้างอิง

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.
[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal
trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.
[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no
2, pages 99-111, 1991.
[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping,"
In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.
[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference
on Computer and Communications Security, pages 28-35, April 1997.
[6] A. Back, "Hashcash - a denial of service counter-measure,"
http://www.hashcash.org/papers/hashcash.pdf, 2002.
[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and
Privacy, IEEE Computer Society, pages 122-133, April 1980.
[8] W. Feller, "An introduction to probability theory and its applications," 1957.

Technical Term

Peer-to-peer: การส่งต่อข้อมูลหากันและกันเป็นเครือข่ายแบบไม่มีตัวกลาง เหมือน BitTorrent
Digital Signature: การเข้ารหัสโดยคอมพิวเตอร์ที่ยากจะเลียนแบบและสามารถยืนยันความถูกต้องของที่มาของข้อมูลได้ เปรียบเสมือนการตรวจสอบลายเซ็นบนเอกสารสำคัญ
hash: คือการเข้ารหัสเพื่อปกปิดข้อมูล โดยการจะไขรหัสเพื่อหาข้อมูลตั้งต้นก่อนการเข้ารหัสนั้น ทำได้ยาก
Node: คอมพิวเตอร์ในเครือข่าย
Best effort: ส่งข้อมูลแบบไม่สนใจว่าจะถึงผู้รับ
Trust: เครดิต หรือความเชื่อถือ
Double spending: การใช้เงินซ้ำที่เป็นหัญหาเมื่อเงินกลายเป็นข้อมูลดิจิทัลเพราะว่ามันอาจจะถูกเคยใช้ไปแล้ว
Cryptographic: การเข้ารหัสด้วย public key และ private key ทำให้เกิดการแลกเปลี่ยนข้อมูลซึ่งกันและกันโดยข้อมูลจะถูกรักษาไว้อย่างปลอดภัย
Mint: ตัวกลาง
Block: กล่องที่เก็บธุรกรรม
Attacker: คนที่คิดจะโกงหรือปลอมแปลงระบบ
Consensus: ฉันทามติ
proof-of-work: การพิสูจน์ด้วยผลงาน เป็นการแก้ไขโจทย์จากการเข้ารหัสที่ต้องใช้เวลาในการทำงานของ CPU

Note:เนื่องจากภาษาที่ใช้ใน White paper เป็นภาษาวิชาการที่ค่อนข้างเข้าใจยากทางทีมงานของเราจึงไม่ได้แปลแบบตรงตัวแต่พยายามจับใจความและแปลให้รูปประโยคดีขึ้นโดยที่ใจความยังคงเดิมเท่าที่ทำได้