ผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของอาร์กอนสองไอ

นายพัชร กกสูงเนิน

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาการสารสนเทศมหาบัณฑิต

สาขาวิชาเทคโนโลยีดิจิทัลและนิเทศศาสตร์ดิจิทัล

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2566

EFFECT OF PARAMETERS ON HASHING SPEED OF ARGON2I

PATCHARA KOKSUNGNOEN

A Paper Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Information Science Program in Digital Technology and Communication

Suranaree University of Technology

Academic Year 2023

ผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของอาร์กอนสองไอ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับการค้นคว้าอิสระฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

ผู้ช่วยศาสตราจารย์ ดร.สุชา สมานชาติ
ประธานกรรมการ

รองศาสตราจารย์ ดร.ศิรปัฐช์ บุญครอง
กรรมการ (อาจารย์ที่ปรึกษา)

รองศาสตราจารย์ ดร.ธรา อั่งสกุล
กรรมการ

รองศาสตราจารย์ ดร.ธรา อั่งสกุล
คณบดีสำนักวิชาศาสตร์และศิลป์ดิจิทัล

นายพัชร กกสูงเนิน : ผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของอาร์กอนสองไอ (EFFECT OF PARAMETERS ON HASHING SPEED OF ARGON2I)
อาจารย์ที่ปรึกษา : รองศาสตราจารย์ ดร.ศิรปัฐช์ บุญครอง, 88 หน้า.

การค้นคว้าอิสระนี้ ทำการทดลอง การแฮชชิ่งด้วย Argon2i เพื่อค้นหาว่าการปรับค่าพารามิเตอร์ ได้แก่ ความยาวของรหัสผ่าน ความยาวของ Salt ขนาดของ Memory size (k) Iteration number (t) Parallelism (p) และ Tag length (l) ขนาดใด ที่ใช้เวลาในการแฮชชิ่งที่น้อยที่สุดที่ Argon2i สามารถทำได้ เพื่อแก้ปัญหาของ Argon2i ที่มีจุดอ่อนด้านเวลาในการแฮชชิ่งที่ต้องใช้เวลานาน และ พารามิเตอร์ที่สามารถปรับแก้ไขได้หลากหลาย ทำให้หากเลือกปรับค่าได้ไม่เหมาะสมจะทำให้เวลาในการแฮชชิ่งนานเกินไปจนทำให้ผู้ใช้ระบบไม่พอใจ นอกจากนั้น ผู้วิจัยยังได้นำค่าพารามิเตอร์เหล่านี้มาทดลอง Avalanche Effect กับ แฮชชิ่งอัลกอริธึมอื่น ๆ ได้แก่ MD5 SHA1 และ SHA256 เพื่อเปรียบเทียบและประเมินด้านความปลอดภัยของ Argon2i ที่ได้เวลาที่น้อยที่สุด ว่ามีความปลอดภัยที่เพียงพอต่อการใช้งานจริง เมื่อเทียบกับ อัลกอริธึม ต่าง ๆ ที่ใช้งานจริงในปัจจุบัน สุดท้าย ผู้วิจัยได้ทดลองปรับเปลี่ยนพารามิเตอร์ของ Argon2i เป็นรายตัว เพื่อหาว่าการปรับขนาดของ พารามิเตอร์ตัวไหนที่มีผลต่อความปลอดภัยมากที่สุด

จากการวิจัย สามารถสรุปได้ว่า Argon2i สามารถทำให้ได้เวลาที่น้อยที่สุดโดยปรับค่าพารามิเตอร์ดังนี้ Memory size ปรับค่าเท่ากับ 4000 KiB และ Iteration number ปรับค่าเท่ากับ 2 และ Parallelism (p) ปรับค่าเท่ากับ 8 ซึ่งได้มาจากค่า 2 เท่าของ Threads CPU และ รหัสผ่าน ให้มีความยาวเท่ากับ 28 ตัวอักษร และ ค่า Salt ให้มีความยาวเท่ากับ 24 ตัวอักษร และ Tag length (l) ให้ปรับเป็นขนาด 32 bits ซึ่งเป็นค่าจะผลทดลองที่ได้เวลาที่ต่ำที่สุด และ ส่วนของการประเมินด้านความปลอดภัยเมื่อเทียบกับการปรับพารามิเตอร์ด้วยความยาวรหัสผ่าน จะมีผลต่อความปลอดภัยของ Argon2i มากที่สุดและเมื่อเทียบกับ อัลกอริธึมอื่น ๆ Argon2i ที่ปรับค่าพารามิเตอร์ให้ได้เวลาที่น้อยที่สุด นั้นมีความปลอดภัยที่เทียบเท่ากับ ได้แก่ MD5 SHA1 และ SHA256

สาขาวิชาเทคโนโลยีดิจิทัลและนิเทศศาสตร์ดิจิทัล ลายมือชื่อนักศึกษา _____
ปีการศึกษา 2566 ลายมือชื่ออาจารย์ที่ปรึกษา _____

PATCHARA KOKSUNGNOEN : EFFECT OF PARAMETERS ON HASHING SPEED OF ARGON2I: THESIS ADVISOR : ASSOC. PROF. SIRAPAT BOONKRONG, Ph.D. 88 PP.

ARGON2/ARGON2I/HASHINGPASSWORD/AVALANCHEEFFECT/ ARGON2IASSESSMENT

Experiment with hashing using Argon2i to determine how to adjust parameters such as password length, salt length, memory size (k), iteration number (t), parallelism (p), and tag length (l) will help optimise the security of the system. The objective of this independent study is to minimise the hashing time of Argon2i, addressing its weakness in prolonged hashing time and the presence of numerous adjustable parameters. Inappropriately adjusting these settings can result in excessively long hashing times, leading to user dissatisfaction. Additionally, these parameters were tested for the avalanche effect and compared with other hashing algorithms, including MD5, SHA1, and SHA256, to conduct a minimal-time security assessment of Argon2i, ensuring it is secure enough for practical use. By comparing Argon2i with various algorithms currently in use and experiment with each parameter, we can determine which ones have the greatest impact on safety.

The results indicate that the lowest time for Argon2i can be achieved by adjusting the following parameters: setting memory size to 4000 KiB, iteration number to 2, and parallelism to 8, derived from a value of 2 times the CPU threads. The password must be 28 characters long, the salt value should be 24 characters long, and the tag length should be adjusted to 32 bits, as this value yields the lowest time. Evaluate the security of Argon2i by examining the impact of password length on parameterisation. This aspect has the most significant effect on the safety of Argon2i.

School of Digital Technology and Communication Student's Signature_____

Academic Year 2023 Advisor's Signature_____

กิตติกรรมประกาศ

การค้นคว้าอิสระ นี้เล่มนี้สำเร็จลุล่วงด้วยดี เนื่องด้วยความดูแล เอาใจใส่ และให้คำแนะนำอย่างดียิ่ง จากรองศาสตราจารย์ ดร.ศิรปัฐช์ บุญครอง อาจารย์ที่ปรึกษาการค้นคว้าอิสระ ที่กรุณาให้คำปรึกษา

ข้อเสนอแนะทางวิชาการ และแก้ไขปัญหาต่าง ๆ ในการทำการค้นคว้าอิสระ เป็นกำลังใจตลอดจนตรวจทาน และแก้ไขการค้นคว้าอิสระให้แก่ผู้วิจัยจนเสร็จสมบูรณ์

ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.สุชา สมานชาติ ประธานกรรมการค้นคว้าอิสระ และรองศาสตราจารย์ ดร.ธรา อั่งสกุล กรรมการสอบการค้นคว้าอิสระ ที่กรุณาเสียสละเวลาอันมีค่ามาพิจารณาและให้คำแนะนำการปรับปรุงการค้นคว้าอิสร และให้ความรู้ที่เป็นประโยชน์ต่อการค้นคว้าอิสระเป็นอย่างยิ่ง

ขอกราบขอบพระคุณคณาจารย์ทุกท่านในสำนักวิชาศาสตร์และศิลป์ดิจิทัล มหาวิทยาลัยเทคโนโลยีสุรนารีทุกท่าน ที่ประสิทธิ์ประสาทวิชา ความรู้ และประสบการณ์อันมีค่าที่ล้วนแต่เป็นประโยชน์ต่องานวิจัย

ขอขอบคุณเพื่อน ๆ พี่ๆ และน้อง ๆ ทุกท่านที่เป็นกำลังใจและให้ความช่วยเหลือในเรื่องการเรียนและการค้นคว้าอิสระฉบับนี้

ท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา ที่ให้การเลี้ยงดูอบรมและส่งเสริม การศึกษาเป็นอย่างดีมาตลอด และเป็นกำลังใจให้ข้าพเจ้าเสมอมาใน ทำให้การค้นคว้าอิสระ นี้สำเร็จ ลุล่วงไปได้ด้วยดี

คุณงามความดีอันใดที่เกิดจากการค้นคว้าอิสระเล่มนี้ผู้วิจัยขอมอบให้บิดา มารดา พี่น้อง และอาจารย์ที่เคารพทุกท่าน

พัชร กกสูงเนิน

สารบัญ

หน้า

บทคัดย่อ (ภาษาไทย) ..... ก
บทคัดย่อ (ภาษาอังกฤษ) ..... ข
กิตติกรรมประกาศ ..... ค
สารบัญ ..... ง
สารบัญตาราง ..... ช
สารบัญรูป ..... ซ

บทที่

1 บทนำ 1
1.1 ความสำคัญและที่มาของปัญหาการวิจัย 1
1.2 วัตถุประสงค์ของการวิจัย 4
1.3 ประโยชน์ที่คาดว่าจะได้รับ 4
1.4 ขอบเขตของการวิจัย 4
1.5 คำอธิบายศัพท์ 5
2 ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง 6
2.1 ความหมายของการพิสูจน์ตัวตน (Authentication) 6
2.1.1 ปัจจัยของการพิสูจน์ตัวตน (Factor of Authentication) 8
2.2 จัดเก็บรหัสผ่าน (Password Storage) 10
2.2.1 การจัดเก็บรหัสผ่านเป็นข้อความธรรมดา (Plaintext Password) 10
2.2.2 การจัดเก็บรหัสผ่านแบบแฮช (Hash password) 11
2.2.3 การจัดเก็บรหัสผ่านแบบแฮชสองครั้ง (Double hash password) 13
2.2.4 การจัดเก็บรหัสผ่านแบบเติมค่าซอลส์ (Salted hash password) 14
2.3 อาร์กอนสอง (Argon2) 17
2.3.1 Argon2d 17
2.3.2 Argon2i 18
2.3.3 Argon2id 19
2.4 อินพุต (input) ทั้งหมดของ Argon2i 19

สารบัญ (ต่อ)

หน้า

2.5 Avalanche effect 21
2.6 งานวิจัยที่เกี่ยวข้อง 21
2.6.1 งานวิจัยที่เกี่ยวข้องกับ Avalanche effect 22
2.6.2 งานวิจัยที่เกี่ยวข้องกับโจมตีฐานข้อมูลที่เก็บข้อมูลรหัสผ่าน 23
2.6.3 งานวิจัยที่เกี่ยวข้องกับการเลือกพารามิเตอร์ที่เหมาะสมกับ Argon2di 24
3 วิธีดำเนินการวิจัย 26
3.1 วิธีวิจัย 26
3.2 ศึกษาพารามิเตอร์ที่มีผลกระทบต่อเวลา 27
3.3 การเตรียมสภาพแวดล้อมสำหรับการทดลอง 28
3.4 การกำหนดพารามิเตอร์ 30
3.5 การประเมินผล 37
3.5.1 การประเมินค่าพารามิเตอร์ที่มีผลกระทบต่อเวลา 37
3.5.2 การประเมินด้านความปลอดภัย 38
4 ผลการวิจัยและการอภิปรายผล 40
4.1 ผลการทดลองในด้านของเวลาของพารามิเตอร์ 40
4.1.1 ผลการทดลองของพารามิเตอร์รหัสผ่าน 40
4.1.2 ผลการทดลองของพารามิเตอร์ Salt 41
4.1.3 ผลการทดลองของพารามิเตอร์ Memory size 42
4.1.4 หาพารามิเตอร์ Iteration number (t) 44
4.1.5 หาพารามิเตอร์ Parallelism (p) 46
4.1.6 หาพารามิเตอร์ Tag length (l) 47
4.2 การประเมินผล Avalanche effect 51
5 สรุปและข้อเสนอแนะ 55
5.1 สรุปผลการวิจัย 56
5.1.1 สรุปผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i 56
5.1.2 สรุปผลการประเมินด้านความปลอดภัยของ Argon2i 56
5.2 การประยุกต์ผลการวิจัย 57

สารบัญ (ต่อ)

หน้า

5.3 ข้อเสนอแนะในการวิจัยครั้งต่อไป 58
รายการอ้างอิง 59
ภาคผนวก ก. 60
ภาคผนวก ข. 75
ประวัติผู้เขียน 88

สารบัญตาราง

หน้า

ตารางที่ 2.1 แสดงค่าพารามิเตอร์ที่แนะนำโดยคุณ Bryan Burman 24
ตารางที่ 3.1 พารามิเตอร์ที่สามารถปรับค่าได้ของ Argon2i 26
ตารางที่ 3.2 แสดงอักษรพารามิเตอร์ของคำสั่ง Argon2 28
ตารางที่ 3.3 แสดงพารามิเตอร์ต่าง ๆ ที่ใช้ในการทดลอง 31
ตารางที่ 3.4 การปรับเปลี่ยนค่าเพื่อทดลอง Avalanche effect 37
ตารางที่ 3.5 แสดงขนาดของตัวอักษรแฮชซึ่ง output และ เลขฐานสอง 38
ตารางที่ 4.1 ผลสรุปค่าพารามิเตอร์ที่ใช้เวลาน้อยที่สุด 47
ตารางที่ 4.2 ผลสรุปค่าพารามิเตอร์ที่ใช้เวลามากที่สุด 47
ตารางที่ 4.3 แสดงการเพิ่มค่าพารามิเตอร์ไป 1 เท่าตัว 48

สารบัญรูป

หน้า

รูปที่ 2.1 กระบวนการ Registration 7
รูปที่ 2.2 กระบวนการ Authentication 7
รูปที่ 2.3 ตัวอย่างการเก็บข้อมูลรหัสผ่านโดยที่ไม่ผ่านกระบวนการใด ๆ เลย 10
รูปที่ 2.4 กระบวนการการจัดเก็บแบบแฮชชิ่งด้วย อัลกอริทึม MD5 11
รูปที่ 2.5 กระบวนการการจัดเก็บแบบแฮชชิ่งด้วย อัลกอริทึม SHA256 12
รูปที่ 2.6 ขั้นตอนการโจมตีด้วย Rainbow Table 12
รูปที่ 2.7 กระบวนการการจัดเก็บรหัสผ่านแบบแฮชชิ่งสองครั้ง 13
รูปที่ 2.8 กระบวนการการจัดเก็บรหัสผ่านแบบเติมค่าซอลท์ 14
รูปที่ 2.9 กระบวนการจัดเก็บค่า Salt 15
รูปที่ 2.10 กระบวนการยืนยันตัวตนของ Salted hash password 16
รูปที่ 2.11 วิธีการทดลอง Avalanche effect ตัวอย่างการศึกษางานวิจัยอื่น 22
รูปที่ 2.12 ผลการทดลอง Avalanche effect ตัวอย่างการศึกษางานวิจัยอื่น 23
รูปที่ 3.1 แสดงลำดับขั้นตอนของการทดลอง 25
รูปที่ 3.2 หน้าเว็บไซต์ multipass 28
รูปที่ 3.3 อัพเดทแพ็คเก็จของระบบ Linux 29
รูปที่ 3.4 การติดตั้ง argon2 29
รูปที่ 3.5 การรันเพื่อแฮชชิ่งด้วย Argon2i 30
รูปที่ 3.6 แสดงตัวอย่างในการกำหนดพารามิเตอร์ ในแต่ละกลุ่มตัวอย่าง 35
รูปที่ 3.7 แสดงตัวอย่างค่าเวลาการคำนวณที่ Argon2 แสดงผลออกมา 36
รูปที่ 4.1 เวลาที่ใช้ในการประมวลผลสำหรับรหัสผ่านขนาดต่าง ๆ 39
รูปที่ 4.2 เวลาที่ใช้ในการประมวลผลสำหรับ Salt ขนาดต่าง ๆ 40
รูปที่ 4.3 เวลาที่ใช้ในการประมวลผลสำหรับ Memory size เมื่อเพิ่มขนาดครั้งละ 4,000 KiB 41
รูปที่ 4.4 เวลาที่ใช้ในการประมวลผลสำหรับ Memory size เมื่อถูกเพิ่มขึ้นเท่าตัวต่อครั้ง 42
รูปที่ 4.5 เวลาที่ใช้ในการประมวลผลสำหรับ Iteration number เมื่อเพิ่มขนาดครั้งละ 6 t 43

สารบัญรูป (ต่อ)

หน้า

รูปที่ 4.6 เวลาที่ใช้ในการประมวลผลสำหรับ Iteration number เมื่อเพิ่มขึ้นเท่าตัวต่อครั้ง.....44
รูปที่ 4.7 เวลาที่ใช้ในการประมวลผลสำหรับ Parallelism (p) เมื่อเพิ่มขนาดครั้งละ 6 p.....45
รูปที่ 4.8 เวลาที่ใช้ในการประมวลผลสำหรับ Tag length เมื่อเพิ่มขึ้นครั้งละ 4 bit.....46
รูปที่ 4.9 ผลต่างของเวลากับการเพิ่มขึ้นของพารามิเตอร์ไป 1 เท่าตัว.....49
รูปที่ 4.10 ผลของ Avalanche Effect จากการเปลี่ยนแปลงค่าของแต่ละพารามิเตอร์ของ Argon2i.....50
รูปที่ 4.11 การเปรียบเทียบ Avalanche Effect ของแฮชฟังก์ชัน.....51
รูปที่ 4.12 เปรียบเทียบการเพิ่มขึ้นของเวลาเมื่อทำการเพิ่มค่าพารามิเตอร์ขึ้นตั้งแต่ 1.8 เท่าตัว.....52

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

เนื่องจากในปัจจุบันระบบต่าง ๆ ในโลกได้พบกับปัญหาความปลอดภัยในด้านการรั่วไหลของข้อมูลรหัสผ่านอยู่มาก ซึ่งรหัสผ่าน (Password) หมายถึงชุดตัวอักษรหรืออักขระที่ใช้ในการยืนยันตัวตน ซึ่งการยืนยันตัวตน (Authentication) หมายถึงกระบวนการในการตรวจสอบหรือยืนยันว่าบุคคลหรือเครื่องมือที่กำลังใช้งาน พยายามจะเข้าถึงระบบหรือบัญชีนั้นเป็นตัวตนจริงๆ และมีสิทธิ์ในการเข้าถึงข้อมูลหรือทรัพยากรที่ต้องการ ของผู้ใช้ในระหว่างกระบวนการตรวจสอบสิทธิ์เพื่อใช้งานระบบใดระบบหนึ่ง โดยทั่วไปรหัสผ่านจะใช้ควบคู่กับชื่อผู้ใช้ ได้รับการออกแบบมาให้มีเฉพาะผู้ใช้เท่านั้นที่สามารถรู้ถึงรหัสผ่าน ซึ่งรหัสผ่านนั้นอาจมีความยาวแตกต่างกันไปและอาจประกอบด้วยตัวอักษร ตัวเลข และอักขระพิเศษ (Bacon, 2021) โดยปัญหาด้านความปลอดภัยของรหัสผ่านสามารถเห็นได้จากตัวอย่างข่าว “พบข้อมูลรั่วไหลครั้งใหญ่ที่สุด จำนวนกว่า 8.4 พันล้านรายการ โดยเป็นไฟล์ TXT ขนาดใหญ่กว่า 100 GB ที่ใช้ชื่อว่า “Rockyou 2021” ถูกนำมาปล่อยบนฟอรั่มแฮกเกอร์ ซึ่งข้อมูลที่รั่วไหลดังกล่าวเป็นรหัสผ่าน มีความยาว 6-20 ตัวอักษร คาดว่าเป็นการรวบรวมรหัสผ่านที่ขโมยมาจากการละเมิดและการรั่วไหลของข้อมูลครั้งก่อน” (NT cyfence, 2021)

นอกจากนี้การยังมีการโจมตีรหัสผ่านประเภทต่าง ๆ ที่ผู้โจมตีสามารถโจมตีรหัสผ่านได้ด้วยวิธีต่าง ๆ ดังนี้

Brute Force Attack เป็นการโจมตีโดยการคาดเดาชื่อผู้ใช้งาน, รหัสผ่าน เพื่อเข้าสู่ระบบโดยไม่ได้รับอนุญาต เป็นการโจมตีที่ง่ายและมีอัตราความสำเร็จที่สูง (Cases, 2022) ผู้โจมตีจะใช้เครื่องมือเช่น ระบบที่ใช้สุ่มรหัสผ่านขึ้นมา เพื่อโจมตีเซิร์ฟเวอร์ของเป้าหมาย จุดประสงค์ของผู้โจมตีนั้นเพื่อหวังขโมยข้อมูลขัดขวางการใช้งาน ทำให้ติดไวรัสหรือระบบที่ผู้โจมตีสร้างขึ้นเพื่อดักจับรหัสผ่านหรือดูข้อมูลต่าง ๆ ของระบบ หรือการใช้งานในส่วนต่าง ๆ บนระบบที่ใช้งานอยู่ ซึ่งหากถูกโจมตีด้วยวิธีนี้กับระบบ อาจจะทำให้ความลับของรหัสผ่านผู้ใช้จะสามารถถูกละเมิดได้โดยง่าย หากไม่ได้มีการเข้ารหัสรหัสผ่านของผู้ใช้ และสามารถป้องกันได้ด้วยการ สร้างรหัสผ่านที่ยากและซับซ้อนขึ้น หรือการติดตั้งระบบที่ตรวจจับการเข้าสู่ระบบใน Ip address เดียวกันมากจนเกินไป และ เปลี่ยนข้อมูลผู้ใช้และรหัสผ่านเริ่มต้น ซึ่ง Ip address คือหมายเลขประจำเครื่องคอมพิวเตอร์แต่ละเครื่องในระบบเครือข่ายที่ใช้และแต่ละ ip จะไม่ซ้ำกัน (Kittitat, 2019)

Password Dictionary Attack การโจมตีด้วยรายการคำศัพท์ ผู้โจมตีใช้รายการรหัสผ่านที่เตรียมไว้ เช่น รหัสผ่านที่ถูกใช้บ่อย โดยหวังว่ารหัสผ่านของผู้ใช้นั้นจะตรงกันกับรหัสที่เตรียมไว้ หรือจะใช้รหัสผ่านที่เคยหลุดออกมาแล้วเช่นกับข่าวที่อ้างอิงไว้ข้างต้น การโจมตีด้วยพจนานุกรมเหมาะสมที่สุดสำหรับรหัสผ่านที่มาจากคำง่ายๆ เช่น 'cowboys' 'longhorns' 'password,' 'letmein,' 'iloveyou,' หรือ '123456') (Rapid7, 2022) ผู้โจมตีอาจใช้ วิธีนี้ร่วมกับ Brute Force Attack ได้

Password Spraying Attack การโจมตีด้วยการกระจายรหัสผ่าน พื้นฐานของการโจมตีแบบกระจายรหัสผ่านเกี่ยวข้องกับผู้ใช้ที่ใช้รหัสผ่านเพียงรหัสเดียวกับให้กับหลายระบบ การโจมตีแบบกระจายรหัสผ่านเกิดขึ้นในสองขั้นตอนคือ ผู้โจมตีได้ชื่อของชื่อผู้ใช้ไม่ว่าจะด้วยวิธีใดก็ตาม หลังจากนั้นผู้โจมตีจะเข้าสู่ระบบชื่อผู้ใช้ทั้งหมดโดยใช้รหัสผ่านเดียวกัน ผู้โจมตีจะทำซ้ำขั้นตอนนี้ด้วยรหัสผ่านใหม่จนกว่าจะยืนยันตัวตนได้สำเร็จ (crowdstrike, 2022)

Credential Stuffing คือการที่ผู้โจมตีได้ใช้ชื่อผู้ใช้ และ รหัสผ่านที่หลุดออกมาจากระบบใดระบบหนึ่งที่เคยถูกโจมตีสำเร็จมาแล้ว ซึ่งจะคล้ายกับกรณีข่าวเรื่องรหัสผ่านที่หลุดออกมาที่ได้กล่าวไว้ก่อนหน้านี้ เพื่อยืนยันตัวตนในระบบ เนื่องจากผู้ใช้ส่วนใหญ่มักจะใช้รหัสผ่านและชื่อผู้ใช้ซ้ำกันในหลาย ๆ ระบบ ซึ่งอ้างอิงจากบทสำรวจ TechRepublic ที่กล่าวว่าผู้ใช้ 53% ยอมรับว่าใช้รหัสผ่านเดียวกันในหลายระบบ (McDade, 2022) และนำข้อมูลเหล่านี้ไปกระจายเพื่อโจมตีกับระบบอื่น ๆ เพื่อหวังว่าผู้ใช้จะใช้รหัสผ่านเดิมที่เคยถูกโจมตีได้กับระบบอื่น ๆ ที่ยังไม่ถูกโจมตี (Mueller, 2021)

ผู้โจมตียังสามารถขโมยรหัสผ่านด้วยการ ดักจับข้อมูลระหว่างทางที่ผู้ใช้และระบบ ซึ่งการโจมตีนี้เรียกว่า Packet sniffing ซึ่งเป็นรูปแบบหนึ่งของการโจมตีด้วยการสอดแนมข้อมูลทุกประเภทไม่ว่าจะเป็นข้อมูลผู้รับ ผู้ส่ง และการตอบสนองของผู้ใช้ระบบ รวมไปถึง ข้อมูลชื่อผู้ใช้และรหัสผ่านของผู้ใช้ด้วย ดังนั้นผู้โจมตีสามารถอาศัยช่องโหว่นี้เพื่อดักจับข้อมูลรหัสผ่านของผู้ใช้และนำไปใช้เพื่อเข้าสู่ระบบได้

นอกจากนั้นยังมีการขโมยรหัสผ่าน ด้วยการโจมตีด้วยฟิชชิง (Phishing Attack) หมายถึงรูปแบบการหลอกลวงบนโลกออนไลน์ผ่านการแอบอ้างเป็นเว็บไซต์ต่าง ๆ ที่น่าเชื่อถือ เช่น เว็บไซต์ธนาคาร หรือบัญชีโซเชียลมีเดีย มาจากคำว่า “Fishing” ที่แปลว่า “ตกปลา” ดังนั้น Phishing จึงหมายถึง การปล่อยให้ปลามากินเหยื่อที่ล่อไว้ กล่าวคือ เมื่อผู้ใช้งานคลิกเข้าสู่ระบบเพื่อยืนยันตัวตน ผู้โจมตีจะสร้างเว็บไซต์ปลอมเหล่านั้นก็จะล้วงข้อมูลส่วนบุคคลของผู้ใช้งานไปทันที ไม่ว่าจะเป็นหมายเลขบัตรเครดิต เลขบัตรประจำตัวประชาชน ตลอดจนรหัสผ่านในการเข้าสู่บัญชีต่าง ๆ (PIMLAPAT, 2022) เพื่อหลอกให้ผู้ใช้เปิดเผยรหัสผ่านของตน และการแบ่งปันหรือแชร์ข้อมูลส่วนตัวบนโซเชียลมีเดีย อาจทำให้ผู้โจมตีสามารถใช้ข้อมูลดังกล่าวเพื่อเดารหัสผ่านได้

อย่างไรก็ตาม ยังมีวิธีการสามารถแก้ไขปัญหาการรั่วไหลของรหัสผ่านได้ด้วยการย่อยข้อมูล โดยใช้แฮชชิ่งฟังก์ชัน ยกตัวอย่างเช่น MD5 แต่การย่อยข้อมูลด้วย MD5 นั้นเป็นการย่อยข้อมูลที่ไม่ได้ถูกสร้างมาเพื่อที่จะจัดกับรหัสผ่านด้วยเฉพาะ จึงง่ายต่อการโจมตีด้วย ฐานข้อมูลการแฮชชิ่ง (Rainbow table) ที่หมายถึง ตารางที่ผู้โจมตีสร้างขึ้นมาเพื่อเปรียบเทียบผลลัพธ์การแฮชชิ่งของข้อความต่างๆ กับข้อความจริง โดยการคำนวณฟังก์ชันแฮชชิ่งของทุกรหัสผ่านที่อยู่ในตารางที่ผู้โจมตีระบุไว้ คล้ายกับการ Brute Force Attack ที่จะเป็นการสุ่มรหัสผ่านไปเรื่อย ๆ จนกว่าจะพบค่าแฮชชิ่งที่ตรงกับค่าแฮชชิ่งรหัสผ่านของผู้ใช้ ซึ่งผู้โจมตีสามารถรู้ถึงค่าแฮชชิ่งของรหัสผ่านผู้ใช้ได้ ด้วยวิธีใด ๆ ก็ตาม (Gillis, 2022) อาจเป็นการดักฟังข้อมูลรหัสผ่านด้วย Packet sniffing ระว่างผู้ใช้งานกับระบบ และหลังจากที่ผู้โจมตีพบ ค่าแฮชชิ่งของผู้ใช้แล้ว ผู้โจมตีก็จะนำค่าแฮชชิ่งไปเปรียบเทียบกับตาราง Rainbow table ที่ผู้โจมตีสร้างไว้ และหากเปรียบเทียบสำเร็จก็จะสามารถรู้ถึงรหัสผ่านจริงที่ยังไม่ผ่านการแฮชชิ่งของผู้ใช้ได้ ซึ่งคล้ายกับวิธีการแปลภาษาหนึ่งไปเป็นอีกภาษาหนึ่ง

รวมไปถึงปัญหา การชนกัน (Collisions) ของค่าแฮชชิ่งของ MD5 (Thompson, 2005) ที่หมายถึงผลลัพธ์ของค่าแฮชชิ่งที่ได้ออกมานั้นมีค่าเหมือนกัน ซึ่งจะเป็นช่องโหว่ของการแฮชรหัสผ่าน ใน กรณีที่ ผู้โจมตีนั้นโจมตีด้วยวิธีการ Brute Force Attack ซึ่งหมายถึง การโจมตีรหัสผ่านด้วยการสุ่มค่าขึ้นมาเรื่อย ๆ จนกว่าจะพบรหัสผ่านที่ถูกต้อง ดังนั้นการ Collisions ของค่าแฮชชิ่ง จึงเป็นการช่วยให้ผู้โจมตีสามารถ Brute Force ได้เร็วยิ่งขึ้น

อาร์กอนสองไอ (Argon2i) จะช่วยแก้ปัญหานี้ได้ เนื่องจากเป็นอัลกอริทึมที่ถูกสร้างมาสำหรับการแฮชชิ่ง รหัสผ่านโดยเฉพาะ ที่ยังไม่มีการค้นพบว่าเกิดการ การชนกัน (Collisions) ของค่าแฮชชิ่ง และยัง สามารถป้องกันการโจมตีแบบแลกเปลี่ยน (trade off attacks) (Biryukov, Dinu, & Khovratovich, 2017) ซึ่งเป็นการโจมตีแบบที่เปรียบเทียบในทุก ๆ สิ่งที่ระบบ Output ออกมา ยกตัวอย่างเช่น เวลา ขนาดของหน่วยความจำ ความร้อน เสียง หรือพลังงานที่ใช้ในคำนวณทั้งหมด ยกตัวอย่างเช่น ในแฮชชิ่งในแต่ละครั้งจะต้องผ่านระบบคอมพิวเตอร์ เพื่อใช้ในการคำนวณแฮชชิ่งทางคณิตศาสตร์ ซึ่งคอมพิวเตอร์เหล่านี้จะต้องใช้ทรัพยากรต่าง ๆ เช่น RAM หรือ หน่วยความจำระยะสั้นของคอมพิวเตอร์ และ คอมพิวเตอร์นั้นส่วนใหญ่จะ Spec หรือความสามารถในการคำนวณที่คล้ายกัน ผู้โจมตีอาจจะใช้เครื่องคอมพิวเตอร์รุ่นเดียวกันหรือ Spec ที่เหมือนกัน มาเปรียบเทียบในด้านต่าง ๆ หากผู้โจมตีสามารถรับรู้ถึงค่า RAM ได้ไม่ว่าจะด้วยวิธีใดก็ตาม และนำไปเปรียบเทียบการทำงานได้ว่าค่าแฮชชิ่งนั้นใช้ RAM ไปเท่าไหร่ในการคำนวณ และเช่นกันกับด้านของความร้อนที่ออกมาจากคอมพิวเตอร์ซึ่งเกิดจาก CPU หรือสมองหลักของคอมพิวเตอร์เป็นหลัก และหาก CPU ทำงานหนักก็จะเกิดความร้อนออกมาได้ ผู้โจมตีก็สามารถนำความร้อนนั้นไปเปรียบเทียบได้เช่นกัน

อย่างไรก็ตามการเลือกพารามิเตอร์เพื่อใช้กับ Argon2i เป็นสิ่งที่สำคัญเพื่อให้การแฮชชิ่งทำงานได้อย่างมีประสิทธิภาพในด้านของความเร็วและด้านของความปลอดภัยในการแฮชชิ่งที่เหมาะสม การเลือกค่าพารามิเตอร์ที่ไม่เหมาะสมอาจทำให้เกิดปัญหาด้านเวลาในการแฮชชิ่งและส่งผลต่อประสิทธิภาพของระบบโดยตรง ระบบต่าง ๆ แฮชชิ่งรหัสผ่านได้ช้า ระบบก็จะทำงานได้ช้าลงเช่นกันและอาจทำให้ผู้ใช้งานไม่พอใจในการบริการ เพราะผู้ใช้งานจะต้องรอให้ระบบทำงานเสร็จสิ้นก่อนจึงจะสามารถเข้าใช้งานได้

จากบทความเรื่อง“ผลกระทบของความเร็วในการโหลดหน้าเว็บที่ช้าต่อประสิทธิภาพของเว็บไซต์” (Green, 2016) กล่าวว่า เวลาในการโหลดหน้าเว็บที่ช้าจะเพิ่มอัตราการละทิ้งหน้าเว็บอย่างมากการตอบสนองหน้าเว็บล่าช้า 4 วินาทีส่งผลให้มีอัตราการละทิ้ง 25% และ การหน่วงเวลา 10 วินาทีมักจะทำให้ผู้ใช้ออกจากเว็บไซต์ทันที ซึ่งจะมีผลกระทบต่อรายได้และการค้าขาย ตัวอย่างจากเว็บไซต์ของ Amazon ระบุว่าการล่าช้าในการโหลด 1 วินาทีจะทำให้สูญเสียยอดขาย 1.6 พันล้านดอลลาร์ต่อปี

ดังนั้น การศึกษาผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i เป็นสิ่งสำคัญเพื่อป้องกันการเกิดปัญหาดังกล่าว และช่วยให้การใช้งานระบบต่าง ๆ เป็นไปได้อย่างราบรื่นและมีประสิทธิภาพสูงสุดเท่าที่เป็นไปได้

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อทดลองและค้นหา พารามิเตอร์ที่เหมาะสมกับ Argon2i กับระบบต่าง ๆ

1.2.2 เพื่อประเมินประสิทธิภาพทางด้านเวลา สำหรับ การทำงานของ Argon2i

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 แก้ไขปัญหาด้านเวลาในการแฮชชิ่งรหัสผ่านของ Argon2i ที่ใช้งานกับระบบต่าง ๆ

1.3.2 เพื่อเพิ่มประสิทธิภาพด้านเวลาและความปลอดภัยให้กับระบบต่าง ๆ

1.4 ขอบเขตของการวิจัย

1.4.1. ในด้านฮาร์ดแวร์ จะใช้คอมพิวเตอร์ประเภท X86 บนระบบปฏิบัติการ Windows 10 Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz RAM 32.0

1.4.2. ซอฟต์แวร์ จะใช้ Run ระบบจำลอง Linux ที่ Run ด้วยระบบ Hyper-V ของ Windows ด้วยโปรแกรม Multipass ซึ่งเป็นโปรแกรมที่ใช้เพื่อจำลองระบบปฏิบัติการ Linux

1.5 คำอธิบายศัพท์

1.5.1 Hash function หมายถึงการนำข้อมูลหรือข้อความไปย่อยข้อมูลที่ทำให้ไม่สามารถย้อนความออกมาได้

1.5.2 Salt หมายถึง ข้อความ คำเติมที่ใส่ไปเพื่อที่จะทำให้ผลลัพธ์แตกต่างจากปกติเพื่อป้องกัน การโจมตีจากตาราง Rainbows table

1.5.3 Rainbow table หมายถึง ตารางที่ผู้โจมตีสร้างขึ้นมาเพื่อเปรียบเทียบผลลัพธ์การ Hash ของข้อความต่างๆ กับข้อความจริง

1.5.4 Hyper-V หมายถึง ระบบสร้างเครื่องเสมือนหรือจำลองการทำงานของระบบปฏิบัติการอื่นและแอปพลิเคชันบนเครื่องคอมพิวเตอร์เครื่องเดียว บนระบบปฏิบัติการ Windows

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

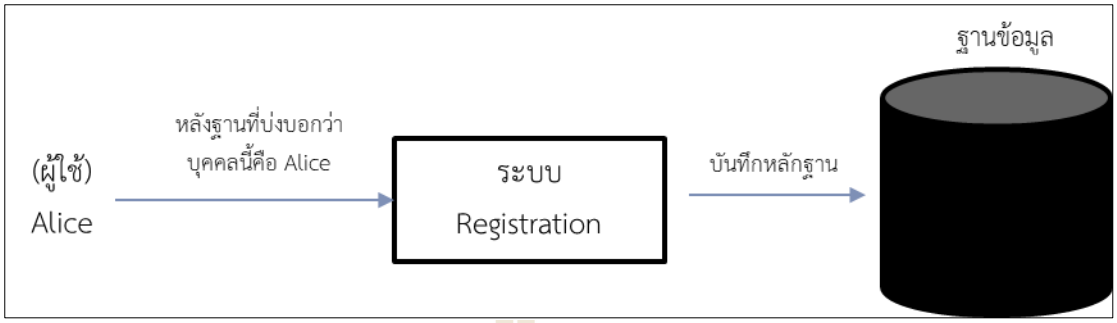
การศึกษาความหมาย วิธีการ และรูปแบบในกระบวนการพิสูจน์ตัวตน เช่น การจัดเก็บรหัสผ่านต่าง ๆ ที่ใช้งานจริง รวมถึงการศึกษากระบวนการเข้ารหัสข้อมูลด้วยอัลกอริทึมแฮชชิ่งต่าง ๆ ผู้วิจัยได้ทบทวนแนวคิด ทฤษฎี และผลงานวิจัยที่เกี่ยวข้องดังนี้

2.1 ความหมายของการพิสูจน์ตัวตน (Authentication)

การพิสูจน์ตัวตน (Authentication) นั้นมีนิยามที่หลากหลายไม่ว่า หมายถึง การตรวจสอบตัวตนของผู้ใช้ในกระบวนการ การเข้าสู่ระบบ ซึ่งผู้ใช้จะถูกระบุตัวตนโดยใช้ปัจจัยการพิสูจน์ตัวตนที่แตกต่างกัน (Mohammed, 2016) หรือ หมายถึง ขั้นตอนการยืนยันความถูกต้องของหลักฐาน (Identity) ที่แสดงว่าเป็นผู้ใช้หรือบุคคลที่กล่าวอ้างจริง ในทางปฏิบัติจะแบ่งออกเป็น 2 ขั้นตอน คือ การระบุตัวตน (Identification) คือขั้นตอนที่ผู้ใช้แสดงหลักฐานว่าตนเองคือใครเช่น ชื่อผู้ใช้ (username) และ การพิสูจน์ตัวตน (Authentication) คือขั้นตอนที่ตรวจสอบหลักฐานเพื่อแสดงว่าเป็นผู้ใช้ที่กล่าวอ้างจริงหลักฐานที่ผู้ใช้นำมากล่าวอ้าง (จิตต์เจริญธรรม, ปานจันทร์, & ลิ้มวิวัฒน์กุล, 2547) หรือคือ การยืนยันว่าผู้ใช้คือบุคคลที่กล่าวว่าเป็น และผู้ที่มีข้อมูลรหัสผ่านที่ได้รับอนุญาตจากระบบเท่านั้นที่จะสามารถเข้าถึงระบบได้ เมื่อผู้ใช้พยายามเข้าถึงข้อมูลบนระบบ ผู้ใช้จะต้องให้ข้อมูลรหัสผ่านที่เป็นความลับเพื่อพิสูจน์ตัวตนของตน (Magnusson, 2023)

โดยสรุปจากนิยามที่กล่าวมาข้างต้นนั้น การพิสูจน์ตัวตน หมายถึงการการพิสูจน์ตัวตนว่าผู้ใช้นั้นเป็นผู้ใช้ที่กล่าวอ้างจริงหรือไม่ ซึ่งการพิสูจน์ตัวตนนั้นจะมีกระบวนการทำงานดังนี้

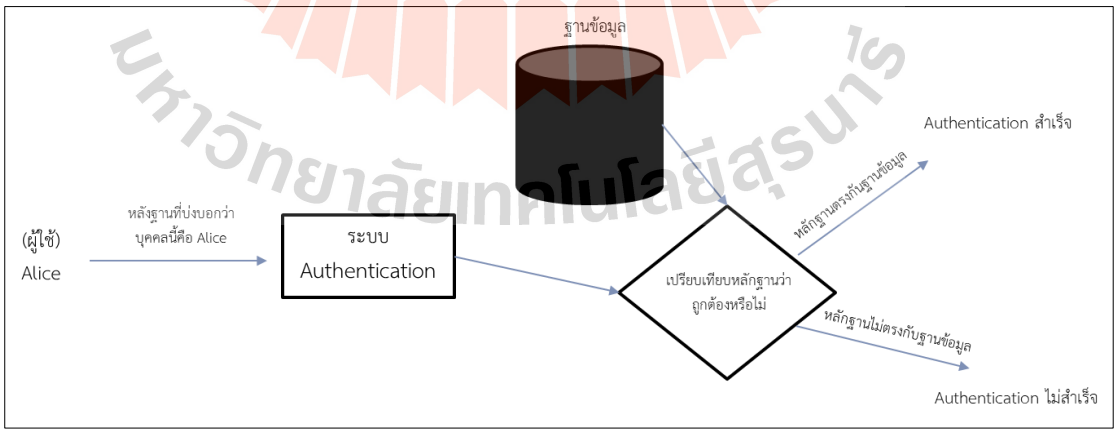
Registration การลงทะเบียน หมายถึงในพิสูจน์ตัวตนนั้น ผู้ใช้ต้องทำการมอบหลักฐานที่สามารถบ่งบอกได้ว่าในความเป็นจริงผู้ใช้ที่กล่าวอ้างนั้นเป็นใคร (จิตต์เจริญธรรม, ปานจันทร์, และลิ้มวิวัฒน์กุล, 2547) ได้แก่ สิ่งที่คุณรู้ (Something you know) สิ่งที่คุณมี (Something you have) และสิ่งที่คุณเป็น (Something you are) ซึ่งจะอธิบายในหัวข้อต่อไป และนำข้อมูลเหล่านี้เก็บไว้ในระบบก่อน เพื่อให้ระบบนั้นใช้เพื่อการพิสูจน์ตัวตนในข้อตอนต่อไป สามารถแสดงการ Registration ได้ในรูปภาพนี้



รูปที่ 2.1 กระบวนการ Registration

รูปที่ 2.1 แสดงให้เห็นกระบวนการของ Registration ที่เป็นการที่ผู้ใช้ได้นำข้อมูลหลักฐานการยืนยันตัวตนที่สามารถบ่งบอกได้ว่าในความเป็นจริงผู้ใช้ที่กล่าวอ้างนั้นเป็นใคร โดยผู้ใช้หรือในรูปนั้นแทนตนเองว่า Alice ได้ส่งหลักฐานการยืนยันตัวตน ยกตัวอย่างเช่น ชื่อผู้ใช้และรหัสผ่าน ที่ Alice สร้างขึ้นและมีเพียง Alice เท่านั้นที่รู้ ให้กับระบบ Registration หลังจากนั้น ระบบก็จะนำข้อมูลหลักฐานนั้นไปจัดเก็บที่ฐานข้อมูล และวิธีการจัดเก็บข้อความรหัสผ่านนั้นสามารถจัดเก็บได้หลากหลายวิธี โดยจะขออธิบายในภายหลัง

สำหรับ Authentication หรือ การพิสูจน์ตัวตน หรือจะสามารถเรียกวิธีการนี้ว่า Login ก็ได้ ซึ่งทำได้โดยการที่ผู้ใช้ระบบต้องถูกยอมรับจากระบบว่าสามารถเข้าสู่ระบบได้ ด้วยการตรวจสอบหลักฐานที่ผู้ใช้ได้เคยมอบให้กับระบบไว้ก่อนหน้านี้แล้วในขั้นตอน Registration ที่ผ่านมา เพื่อแสดงว่าเป็นผู้ใช้เป็นผู้ใช้นั้นจริง ๆ สามารถแสดงกระบวนการ พิสูจน์ตัวตน นี้ได้ด้วยภาพนี้



รูปที่ 2.2 กระบวนการ Authentication

รูปที่ 2.2 แสดงให้เห็นกระบวนการของ Authentication ที่เป็นการตรวจสอบหลักฐานการยืนยันตัวตนที่สามารถบ่งบอกได้ว่าในความเป็นจริงผู้ใช้ที่กล่าวอ้างนั้นเป็นใคร โดยการที่ ผู้ใช้หรือ Alice ได้ ทำการ Authentication หรือ Login เข้าสู่ระบบ หลังจากนั้นระบบ Login จะทำการเปรียบเทียบหลักฐานว่าถูกต้อง ตรงกันกับ ข้อมูลในฐานข้อมูลที่ผู้ใช้ได้บันทึกไว้ก่อนหน้านี้หรือไม่ หากการเปรียบเทียบนั้นถูกต้องระบบก็จะยอมรับการเข้าสู่ระบบของผู้ใช้ หรือหากเปรียบเทียบไม่ถูกต้องระบบก็จะไม่ยอมรับว่าผู้ใช้เป็นผู้ใช้ที่กล่าวอ้างนั้นจริง ๆ

ในส่วนของข้อมูลหลักฐานการยืนยันตัวตนที่สามารถบ่งบอกได้ว่าในความเป็นจริงผู้ใช้ที่กล่าวอ้างนั้นเป็นใคร นั้นถูกเรียกว่า ปัจจัยของการพิสูจน์ตัวตน และสามารถอธิบายได้ดังนี้

2.1.1 ปัจจัยของการพิสูจน์ตัวตน (Factor of Authentication)

กระบวนการการพิสูจน์ตัวตนนั้นมีหลากหลายวิธีด้วยกัน และทั้งหมดนี้จะเป็นการเปรียบเทียบด้วยหลักฐานการยืนยันตัวตนของผู้ใช้ที่บันทึกไว้ในระบบด้วยวิธี Registration และข้อมูลที่ผู้อ้างว่าเป็นผู้ใช้ยืนยันตัวตนกับระบบผ่านกระบวนการ Authentication ไม่ว่าจะเป็นการใช้อุปกรณ์อิเล็กทรอนิกส์ อย่างเช่น ระบบ RFID คือ การระบุเอกลักษณ์ด้วยคลื่นวิทยุ โดยย่อมาจาก (Radio Frequency Identification) หรือเรียกได้ว่าเป็นระบบเก็บข้อมูลทางอิเล็กทรอนิกส์ที่เพิ่มความสามารถในการคำนวณและการรักษาความปลอดภัยของข้อมูล และส่งคลื่นแม่เหล็กไฟฟ้าแทนการสัมผัส เป็นการนำคลื่นวิทยุมาเป็นคลื่นพาหะเพื่อใช้ในการสื่อสารข้อมูล (JIN, 2022) หรือการใช้ชีวมิติ อย่างเช่น การสแกนลายนิ้วมือหรือม่านตาของผู้ใช้ และ การใช้ ชื่อผู้ใช้ (Username) และ รหัสผ่าน (Password) ระบบ

การพิสูจน์ตัวตนเป็นกระบวนการที่ผู้ใช้อ้างว่าเป็นผู้ใช้ผู้นั้นจริง ๆ ซึ่งการรับรองความถูกต้องมีสามปัจจัยทั้งหมด 3 ปัจจัย ได้แก่

1. สิ่งที่คุณรู้ (Something you know) หมายถึง ข้อมูลที่จะมีเฉพาะคุณเท่านั้นที่จะรู้ได้ ซึ่งอาจจะเป็นข้อมูลส่วนตัว หรือ ข้อความที่คุณตั้งไว้สำหรับการพิสูจน์ตัวตนนั้น ๆ เช่น ข้อความรหัสลับ PIN (Personal Identification Number) คือข้อความรหัสลับที่ใช้ในการยืนยันตัวตนของบุคคลหรือผู้ใช้ในระบบต่าง ๆ เป็นทางการส่วนตัวและเป็นความลับที่ไม่ควรเปิดเผยกับผู้อื่น และ PIN จะมีความยาวระหว่าง 4-6 ตัวอักษร หรือ Username ชื่อผู้ใช้ และ Password รหัสผ่าน ที่คุณกำหนดไว้ในขั้นตอน Registration ซึ่งจะถูกเก็บไว้ในฐานข้อมูลและนำมาเปรียบเทียบกับผู้ใช้ เมื่อผู้ใช้ทำการยืนยันตัวตน

2. สิ่งที่คุณมี (Something you have) หมายถึง สิ่งที่เกี่ยวข้องกับคุณ วัตถุทางกายภาพหรือดิจิทัลที่คุณเท่านั้นที่สามารถเข้าถึงได้ ตัวอย่างเช่น บัตรเข้าใช้งาน เช่นบัตร ATM บัตรพนักงาน หรือ โทเค็นฮาร์ดแวร์ (Token Hardware) หมายถึง อุปกรณ์ที่ใช้ในระบบความปลอดภัยเพื่อยืนยัน

ตัวตนของผู้ใช้หรือรับรองความถูกต้องของการเข้าถึงข้อมูลหรือระบบ (Andress, 2014) โดยอุปกรณ์นี้จะสร้างรหัสผ่าน รหัสผ่านแบบครั้งเดียว One-Time Password (OTP) หมายความว่ารหัสผ่านที่สร้างขึ้นจะหมดอายุเมื่อมีการใช้งานและไม่สามารถใช้ได้อีกในการพยายามเข้าสู่ระบบครั้งต่อไป (Huseynov & Seigneur, 2017) โดยจะมีเพียงผู้ใช้โทเค็นฮาร์ดแวร์และระบบเท่านั้นที่รู้รหัสผ่าน OTP ซึ่งระบบอาจเก็บโทเค็นฮาร์ดแวร์รูปแบบเดียวกันไว้เพื่อเปรียบเทียบเมื่อผู้ใช้ต้องการยืนยันตัวตน โทเค็นฮาร์ดแวร์นั้นมีหลากหลายแบบไม่ว่าจะเป็น โทเค็นฮาร์ดแวร์แบบพกพาที่จะสร้างรหัส OTP อยู่ตลอดเวลา ทำให้รหัสผ่านนั้นเปลี่ยนแปลงอยู่ตลอด และ แบบที่อยู่ในแอปพลิเคชันบนสมาร์ทโฟน เช่น Google Authenticator ที่สามารถสร้างรหัส OTP ได้เช่นกัน ดังนั้นการยืนยันตัวตนวิธีนี้ไม่จำเป็นต้องเก็บข้อมูลรหัสผ่านไว้ที่ฐานข้อมูลเพื่อทำการยืนยันตัวตน แต่ก็ยังต้องมีการตกลง Token Hardware และ บันทึกประเภทหรือชนิด Token Hardware ไว้ในฐานข้อมูลเช่นกัน

3. สิ่งที่คุณเป็น (Something you are) หมายถึง ข้อมูลคุณลักษณะของ Biometric การรับรองความถูกต้องทางชีวภาพ หมายถึง กระบวนการรักษาความปลอดภัยที่ต้องอาศัยลักษณะทางชีววิทยาที่เป็นเอกลักษณ์ของแต่ละบุคคลเพื่อตรวจสอบว่าตนเป็นใคร ระบบการตรวจสอบความถูกต้องด้วยไบโอเมตริกซ์จะเปรียบเทียบลักษณะทางกายภาพหรือพฤติกรรมกับข้อมูลที่จัดเก็บในฐานข้อมูลในขั้นตอน Registration เช่น ลายนิ้วมือ ลักษณะใบหน้า รูปแบบม่านตา เสียงพูด หรือชีวมิติเชิงพฤติกรรม เช่น ความเร็วในการพิมพ์และการเคลื่อนไหวของเมาส์ หากตัวอย่างข้อมูลไบโอเมตริกทั้งสองตรงกันการรับรองความถูกต้องจะได้รับการยืนยัน (Hashemi-Pour, 2023) ดังนั้นการยืนยันด้วยวิธีนี้ต้องจัดเก็บข้อมูลชีวมวลแทนการจัดเก็บข้อมูลรหัสผ่านของผู้ใช้ เพื่อการเปรียบเทียบในการยืนยันตัวตน

ปัจจัยที่มักถูกใช้ในการระบุตัวตนนั้นคือสิ่งที่คุณรู้ (Something you know) ที่ใช้ ชื่อผู้ใช้และรหัสผ่าน (Password-based) อ้างอิงจากที่มีธุรกิจถึง 59% ที่ใช้การยืนยันตัวนี้ประเภทนี้ (descope, 2023) และ ในด้านของความคุ้นเคยของผู้ใช้เนื่องจากเป็นการยืนยันตัวตนที่ใช้กันมาเป็นเวลานาน หรือจะเป็นความง่ายในการใช้งาน ซึ่งการใช้รหัสผ่านนั้นค่อนข้างง่ายจากมุมมองทางเทคนิคเมื่อเทียบกับปัจจัยอื่นที่ ต้องใช้บัตรยืนยัน การใช้แอปพลิเคชันเสริมหรืออุปกรณ์เสริมอื่น ๆ อีกทั้งการใช้รหัสผ่านเพื่อเข้าถึงเว็บไซต์ ซึ่งเว็บไซต์ชั้นนำของโลก เช่น Facebook, Google หรือ Microsoft ก็ยังใช้วิธีนี้อยู่

อย่างไรก็ตาม การใช้รหัสผ่านก็ยังมีข้อเสียเปรียบที่สำคัญและข้อกังวลด้านความปลอดภัยที่เกี่ยวข้องกับการยืนยันตัวตนรหัสผ่าน ดังนี้

1. การตั้งรหัสผ่านที่ไม่รัดกุม ซึ่งผู้ใช้หลายคนเลือกรหัสผ่านที่ไม่รัดกุมซึ่งเดาหรือถอดรหัสได้ง่าย มักเกิดจากการขาดความตระหนักหรือความยากลำบากในการจดจำรหัสผ่านที่ซับซ้อนของผู้ใช้

2. การใช้รหัสผ่านซ้ำ ผู้ใช้มักจะใช้รหัสผ่านซ้ำในหลายบัญชี ซึ่งจะขยายผลกระทบของความเสียหาย หากบัญชีหนึ่งถูกโจมตีและบุกรุก บัญชีอื่น ๆ ก็มีความเสี่ยงเช่นกัน

3. ช่องโหว่ของการขโมยรหัสผ่านด้วยการโจมตีด้วยฟิชชิง (Phishing Attack)

4. การขโมยรหัสผ่าน โดยรหัสผ่านสามารถถูกขโมยได้ อาจเป็นการที่ผู้ใช้จดรหัสผ่านไว้ที่กระดาษแล้วถูกขโมย ทำหาย หรือการเก็บรหัสผ่านไว้กับระบบที่ไม่ปลอดภัย

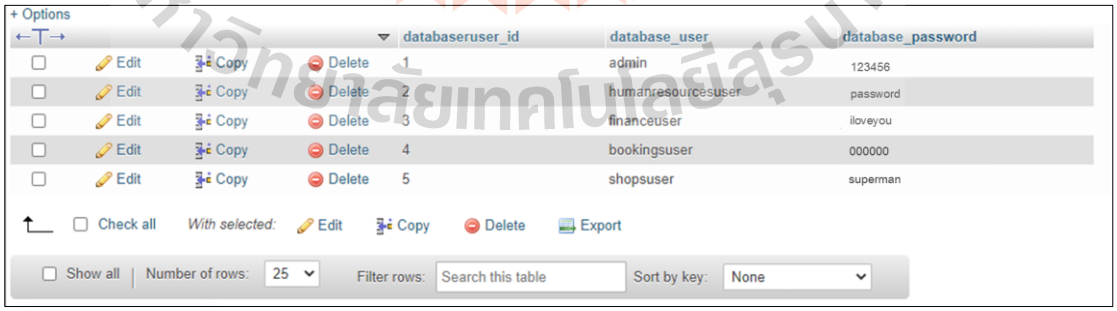
5. ลืมรหัสผ่าน หากผู้ใช้ลืมรหัสผ่าน ก็จำเป็นต้องใช้ระบบ การรีเซ็ตรหัสผ่าน ซึ่งอาจเป็นช่องโหว่หนึ่งที่ ผู้โจมตีสามารถโจมตีได้

2.2 จัดเก็บรหัสผ่าน (Password Storage)

เทคนิคและวิธีการที่ใช้ในการจัดเก็บรหัสผ่านสามารถจัด เก็บได้หลากหลายวิธีดังต่อไปนี้ (สมบูรณ์พัฒนากิจ และ บุญครอง, 2014)

2.2.1 การจัดเก็บรหัสผ่านเป็นข้อความธรรมดา (Plaintext Password)

การจัดเก็บรหัสผ่านไว้ในฐานข้อมูลโดยไม่มีการเข้ารหัสเลย เป็นชนิดที่ง่ายที่สุดและความปลอดภัยต่ำที่สุด เพราะหากผู้โจมตีสามารถเข้าถึงข้อมูลรหัสผ่านของผู้ใช้ได้ นั้นแปลว่าผู้โจมตีจะสามารถนำรหัสผ่านไปใช้งานได้เลย และรหัสผ่านของผู้ใช้ก็จะไม่เป็นความลับอีกต่อไป แสดงในตัวอย่างดังรูปที่ 2.3



รูปที่ 2.3 ตัวอย่างการเก็บข้อมูลรหัสผ่านโดยที่ไม่ผ่านกระบวนการใด ๆ เลย

รูปที่ 2.3 แสดงตัวอย่างการเก็บข้อมูลรหัสผ่านด้วยฐานข้อมูล MySQL ที่เป็นการเก็บแบบข้อความธรรมดาโดยที่ไม่ได้มีการคำนวณหรือประมวลผลใด ๆ กับรหัสผ่านของผู้ใช้

ทั้งหมดนี้แสดงให้เห็นว่าการจัดเก็บรหัสผ่านด้วยวิธีนี้นั้นไม่สามารถเก็บการรักษาความลับของข้อมูลรหัสผ่าน และ ทำให้ขาดในความมั่นคงปลอดภัยของข้อมูลสารสนเทศ (Information Security) หมายถึง ข้อมูลหรือรหัสผ่านของผู้ใช้ นั้นไม่ได้รับการป้องกันจากการเข้าถึงที่ไม่ได้รับอนุญาต การเปิดเผย แก้ไข หรือทำลายข้อมูล จากผู้ที่ไม่ได้รับอนุญาต โดยมีคุณสมบัติและองค์ประกอบหลักด้านความมั่นคงปลอดภัยสารสนเทศ คือ

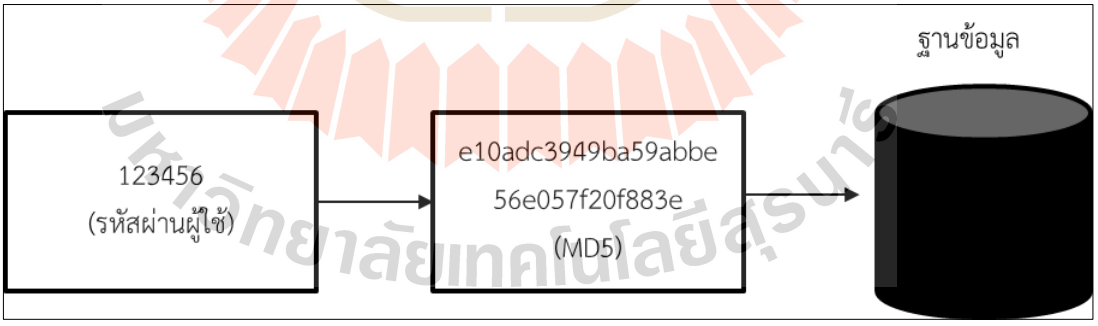
1. การรักษาความลับ (Confidentiality) ข้อมูล รหัสผ่านของผู้ใช้นั้นต้องเข้าถึงได้เฉพาะผู้ที่มีสิทธิ์ หรือได้รับอนุญาตเท่านั้น จะต้องไม่มีการเปิดเผยโดยมิชอบ หรือโดยบุคคลที่ไม่มีสิทธิ์หรือไม่ได้รับอนุญาต

2. การรักษาความถูกต้องครบถ้วน (Integrity) ข้อมูล รหัสผ่านของผู้ใช้นั้นต้อง มีความถูกต้อง จะมีการแก้ไข เปลี่ยนแปลง ได้เฉพาะผู้ที่มีสิทธิ์หรือตัวผู้ใช้เองเท่านั้น

3. สภาพความพร้อมใช้ (Availability) ข้อมูล รหัสผ่านของผู้ใช้นั้นต้องมีความพร้อมในการใช้งานอยู่เสมอ สามารถเข้าถึงได้เมื่อต้องการ เฉพาะผู้มีสิทธิ์ หรือตัวผู้ใช้เอง (acisonline, 2023)

2.2.2 การจัดเก็บรหัสผ่านแบบแฮช (Hash password)

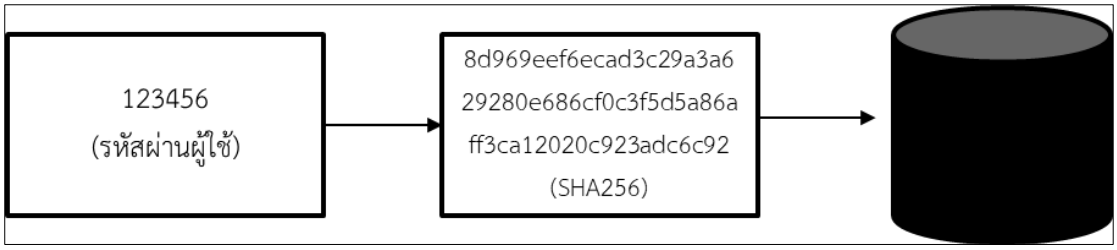
การจัดเก็บรหัสผ่านโดยเข้ารหัสผ่านของผู้ใช้งานทั้งหมดด้วยการแฮชชิ่ง ซึ่งเป็นกระบวนการทางคณิตศาสตร์ที่ใช้เพื่อแปลงข้อมูลเข้าสู่รหัส (หรือ "แฮชชิ่ง") เพื่อให้ข้อมูลดูเป็นตัวเลขหรือสตริงที่มีความยาวคงที่ อาจใช้แฮชชิ่งอัลกอริทึมรูปแบบที่เรียกว่า MD5 หรืออื่น ๆ (สมบูรณ์พัฒนากิจ และ บุญครอง, 2014) ดังรูป



รูปที่ 2.4 กระบวนการการจัดเก็บแบบแฮชชิ่งด้วย อัลกอริทึม MD5

รูปที่ 2.4 แสดงกระบวนการการจัดเก็บแบบแฮช โดยจะนำรหัสผ่านของผู้ใช้ไปแฮชชิ่ง ซึ่งในตัวอย่างใช้อัลกอริทึม MD5 แล้วจึงนำไปเก็บที่ฐานข้อมูลของระบบ

หรือจะใช้อัลกอริทึม SHA256 ซึ่งจะใช้วิธีเดียวกันกับ อัลกอริทึม MD5 แสดงดังรูป

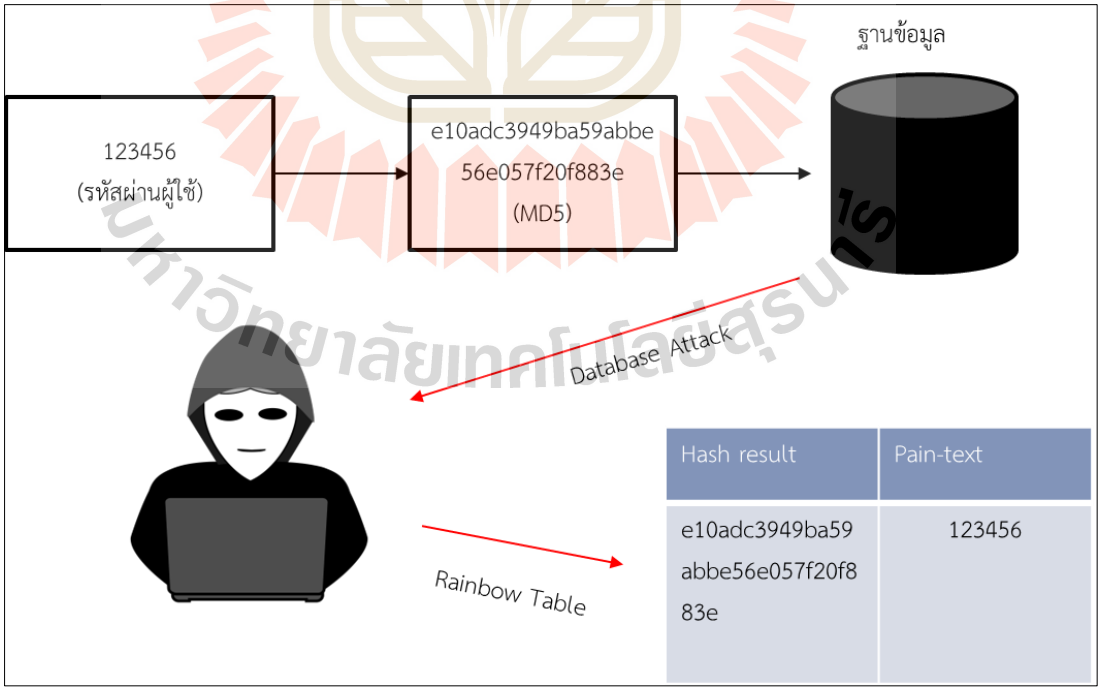


รูปที่ 2.5 กระบวนการการจัดเก็บแบบแฮชชิ่งด้วย อัลกอริทึม SHA256

รูปที่ 2.5 แสดงกระบวนการการจัดเก็บแบบแฮชชิ่ง โดยจะนำรหัสผ่านของผู้ใช้ไปแฮชชิ่ง ซึ่งในตัวอย่างใช้อัลกอริทึม อัลกอริทึม SHA256 แล้วจึงนำไปเก็บที่ฐานข้อมูลของระบบเช่นเดียวกับวิธีของอัลกอริทึม MD5

ซึ่งความหมายของแฮชชิ่งฟังก์ชันนั้นคือกระบวนการทางด้านคณิตศาสตร์ที่ใช้ในการย่อยข้อมูลให้เป็นในรูปแบบต่าง กันไปในแต่ละ อัลกอริทึมที่ใช้ ไม่ว่า จะเป็นไฟล์เอกสาร, Plaintext หรืออื่น ๆ โดยผลลัพธ์ที่ได้จากการย่อยนี้จะถูกเรียกว่า ค่าแฮช (hash value)

แต่วิธีนี้ก็ยังมีช่องโหว่ที่เรียกว่า Rainbow Table ที่เคยอธิบายไว้แล้วในบทที่หนึ่ง ซึ่งผู้โจมตีจะใช้วิธีการต่าง ๆ ในการเข้าถึงข้อมูลรหัสผ่านที่ถูกแฮชชิ่งแล้วนำไปเปรียบเทียบกับตารางรหัสผ่านที่ถูกสร้างขึ้นมา ทำให้ผู้โจมตีสามารถเปรียบเทียบค่ารหัสผ่านจริง ๆ ออกมาได้ ดังรูปที่ 2.6

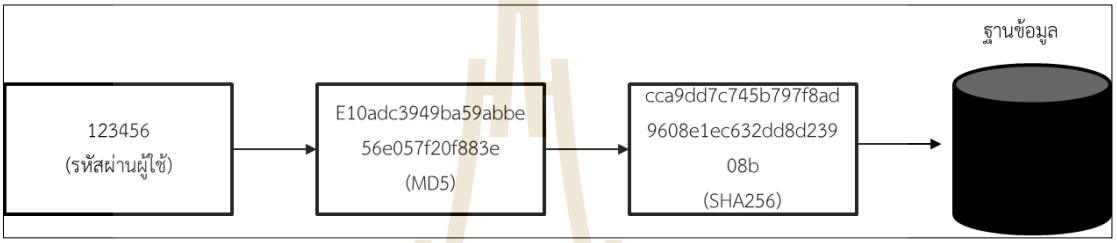


รูปที่ 2.6 ขั้นตอนการโจมตีด้วย Rainbow Table

รูปที่ 2.6 แสดงให้เห็นตัวอย่างวิธีการที่ผู้โจมตี โจมตีการแฮชชิ่งรหัสผ่านด้วยวิธีการเข้าถึงฐานข้อมูลของระบบและนำข้อมูลแฮชชิ่งรหัสผ่านมาเปรียบเทียบกับตาราง Rainbow Table

2.2.3 การจัดเก็บรหัสผ่านแบบแฮชสองครั้ง (Double hash password)

การจัดเก็บรหัสผ่านโดยเข้ารหัสผ่านด้วยการแฮชชิ่งเหมือนกันแต่ทำซ้ำอีกครั้งซึ่งอาจจะเปลี่ยน Algorithm หรือไม่ก็ได้ (สมบูรณ์พัฒนากิจ และ บุญครอง, 2014) ดังรูปที่ 2.7



รูปที่ 2.7 กระบวนการการจัดเก็บรหัสผ่านแบบแฮชชิ่งสองครั้ง

รูปที่ 2.7 แสดงให้เห็นถึงตัวอย่างการจัดเก็บรหัสผ่านแบบแฮชชิ่งสองครั้ง ซึ่งจะนำรหัสผ่านของผู้ใช้มาแฮชชิ่งด้วย MD5 และหลังจากนั้นก็นำผลลัพธ์ของการแฮชชิ่งนั้น ไปแฮชชิ่งด้วย SHA256 ในอีกรอบ แล้วจึงนำไปเก็บที่ฐานข้อมูลของระบบ

โดยข้อดีของการจัดเก็บรหัสผ่านแบบแฮชชิ่งสองครั้งนั้นสามารถแบ่งเป็น 5 ข้อได้ดังนี้ คือ

1. ความปลอดภัยที่เพิ่มขึ้น จากการแฮชชิ่งถึงสองครั้งของข้อความรหัสผ่าน ซึ่งหมายความว่าถึงแม้ผู้โจมตีจะค้นพบค่าแฮชชิ่งแรกด้วยวิธีใดก็ตาม พวกเขายังคงต้องถอดค่าแฮชชิ่งที่สองเพื่อค้นหารหัสผ่านที่แท้จริง สิ่งนี้จึงเป็นการเพิ่มความปลอดภัยให้กับการยืนยันตัวตนด้วยการแฮชชิ่งเพียงครั้งเดียว

2. การป้องกัน Rainbow Tables ผู้โจมตีมักใช้ฐานข้อมูลค่าแฮชชิ่ง เรียกว่า “Rainbow Tables”ที่ได้เคยอธิบายไว้ก่อนหน้านี้แล้ว เมื่อใช้แฮชชิ่งเพียงครั้งเดียว ผู้โจมตีสามารถเปรียบเทียบตารางเหล่านี้ได้ อย่างไรก็ตาม ด้วยการแฮชชิ่งสองครั้ง ผู้โจมตีจะต้องสร้าง Rainbow Tables ชุดใหม่สำหรับแฮชชิ่งที่สอง ซึ่งทำให้การโจมตีมีความซับซ้อนและใช้เวลานานกว่าการยืนยันตัวตนด้วยการแฮชชิ่งเพียงครั้งเดียว

3. การโจมตีแบบ Brute Force ที่ช้าลง การแฮชชิ่งสองครั้งจะทำให้การโจมตีแบบ Brute Force ช้าลง ซึ่งวิธีการโจมตีนี้ได้เคยอธิบายไว้แล้วในบทแรก และเนื่องจากการเดาแต่ละครั้งจะต้องแฮชชิ่งสองครั้ง ซึ่งหมายถึงระบบต้องทำงานช้าเป็น 2 เท่า ซึ่งทำให้ผู้โจมตีต้องใช้เวลานานเป็นสองเท่าในการตรวจสอบการเดาแต่ละครั้ง ทำให้การโจมตีมีประสิทธิภาพน้อยลง

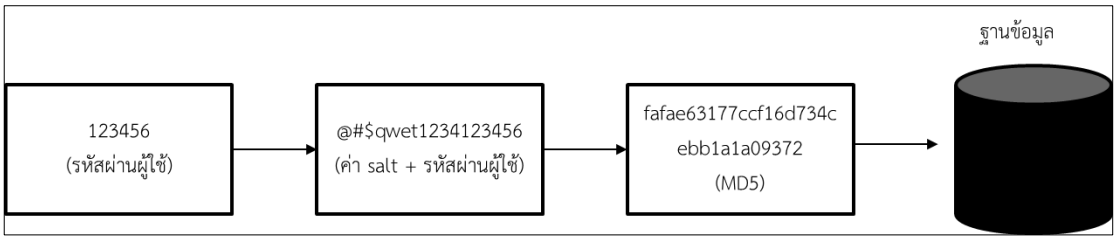
4. ป้องกันการชนกัน (Collisions) ฟังก์ชันแฮชบางตัวมีความเสี่ยงต่อการชนกัน (Collisions) ของค่าแฮชชิ่งซึ่งเคยได้อธิบายไว้แล้วในบทที่หนึ่ง การใช้ double hashing กับฟังก์ชันแฮชชิ่งที่แตกต่างกันสำหรับแต่ละขั้นตอนสามารถช่วยลดความเสี่ยงนี้ได้ แม้ว่าผู้โจมตีจะพบการ Collisions ของฟังก์ชันแฮชชิ่งแรก พวกเขาก็ยังจำเป็นต้องค้นหา Collisions ของฟังก์ชันแฮชชิ่งที่สองด้วย

5. การปรับใช้งานได้ง่ายกว่าในบางกรณี ในบางองค์กรนั้นอาจยังคงใช้ระบบเก่าที่ยังมีการยืนยันตัวตนแบบแฮชการแฮชชิ่งเพียงครั้งเดียว และการที่จะปรับปรุงให้มีความปลอดภัยมากยิ่งขึ้น อาจเป็นเรื่องที่ยากซับซ้อนและสิ้นเปลืองทรัพยากรขององค์กร แต่การปรับเปลี่ยนจากยืนยันตัวตนด้วยการแฮชชิ่งเพียงครั้งเดียวไปเป็นการแฮชชิ่งแบบสองครั้งนั้น ง่ายและประหยัดทรัพยากรขององค์กร กว่าวิธีอื่น ๆ เพราะไม่ต้องปรับแก้ไขฟังก์ชันการยืนยันตัวตนเดิมที่มีอยู่ เพียงแค่เพิ่มฟังก์ชันการแฮชชิ่งอีกชนิดเข้าไปในการยืนยันตัวตนเพียงเท่านั้น ซึ่งจะง่ายกว่าวิธีการอื่น ๆ เช่นการเปลี่ยนไปใช้อัลกอริทึมแฮชชิ่งชนิดอื่นที่ใหม่และทันสมัยกว่า ที่ต้องทำการปรับแก้ฟังก์ชันการยืนยันตัวตนและฐานข้อมูลรหัสผ่านใหม่ในระบบทั้งหมด

แต่ข้อเสียของวิธีนี้ก็คือเวลาในการคำนวณแฮชชิ่งที่นานกว่ารูปแบบอื่นที่กล่าวมาทั้งหมด เพราะว่าการจัดเก็บรหัสผ่านแบบแฮชชิ่งสองครั้ง จะเพิ่มเวลาในการแฮชชิ่งรหัสผ่านจากเดิมนานเป็น 2 เท่าและในด้านของความซับซ้อนของระบบที่เพิ่มขึ้น ที่ผู้ดูแลระบบจะต้องดูแลอัพเดทฟังก์ชันการแฮชชิ่งถึง 2 ฟังก์ชันด้วยกัน สิ่งนั้นทำให้การดูแลนั้นยุ่งยากและซับซ้อนกว่าการแฮชชิ่งเพียงครั้งเดียว

2.2.4 การจัดเก็บรหัสผ่านแบบเติมค่าซอลส์ (Salted hash password)

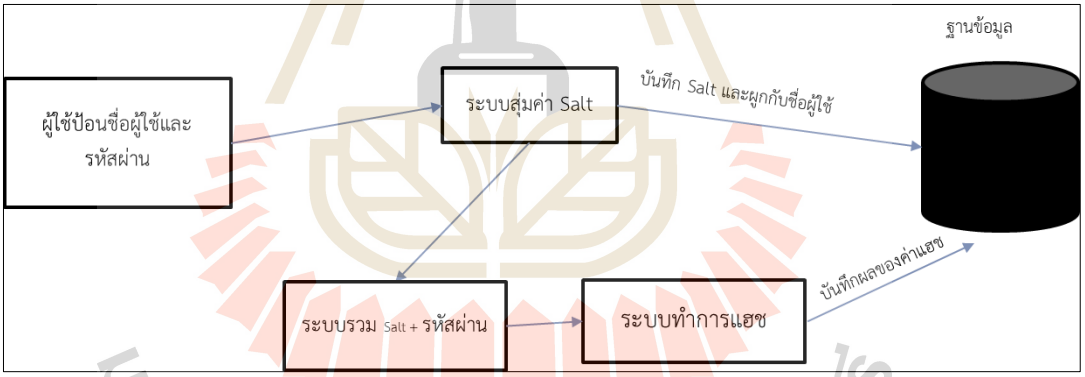
วิธีนี้ทำได้โดย ก่อนที่จะแฮชชิ่งรหัสผ่านจะเติมค่า Salt ให้กับรหัสผ่านเสียก่อน ซึ่งคือค่าสุ่มที่สร้างขึ้นและใช้ร่วมกับรหัสผ่านของผู้ใช้ก่อนที่จะถูกแฮชชิ่ง วัตถุประสงค์หลักของค่า Salt คือ เพื่อเพิ่มความปลอดภัยของรหัสผ่านที่แฮชชิ่งและป้องกันการโจมตีประเภทต่างๆ โดยเฉพาะอย่างยิ่งการโจมตีด้วยการเปรียบเทียบกับตาราง Rainbow Table โดยเฉพาะ (สมบูรณ์ พัฒนากิจ และ บุญครอง, 2014) ดังรูปที่ 2.8



รูปที่ 2.8 กระบวนการการจัดเก็บรหัสผ่านแบบเติมค่าซอลท์

รูปที่ 2.8 แสดงให้เห็นถึงตัวอย่างการจัดเก็บรหัสผ่านแบบเติมค่าซอลท์ ซึ่งจะนำรหัสผ่าน 123456 มาเติมค่า Salt ที่มีค่าเป็น @#$qwet ข้างหน้ารหัสผ่าน แล้วจะได้ค่าเป็น #$qwet123456 และหลังจากนั้นก็นำไปแฮชชิ่งด้วย MD5 แล้วจึงนำไปเก็บที่ฐานข้อมูลของระบบ ซึ่งในฐานข้อมูลจะเก็บค่า salt และค่าแฮชชิ่งไว้ โดยการทำงานนั้นจะมีกระบวนการดังนี้

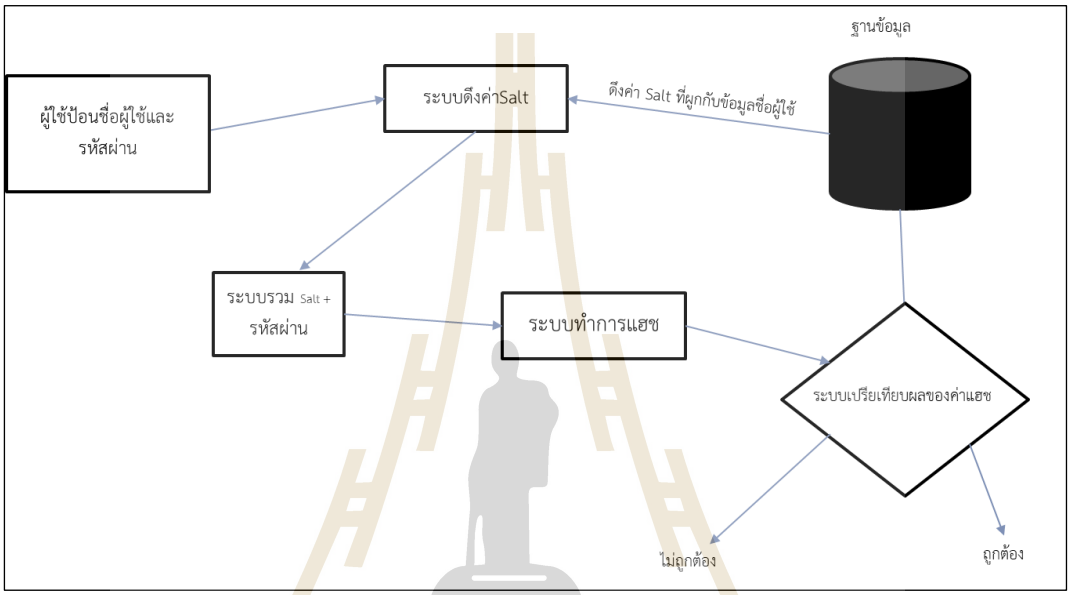
1. ผู้ใช้ตั้งรหัสผ่านกับระบบ ระบบจะทำการสุ่มค่า Salt ขึ้นมาและนำค่า Salt รวมกับรหัสผ่านของผู้ใช้ แล้วจึงคำนวณค่าแฮชชิ่ง และจัดเก็บทั้งข้อมูลค่าแฮชชิ่งและค่า Salt ไว้ในฐานข้อมูล สามารถอธิบายได้ดังรูป



รูปที่ 2.9 กระบวนการจัดเก็บค่า Salt

รูปที่ 2.9 แสดงกระบวนการจัดเก็บค่า Salt โดยเริ่มจากผู้ใช้ป้อนชื่อผู้ใช้และรหัสผ่านของผู้ใช้ ต่อจากนั้นระบบสุ่มค่า Salt ขึ้นมาและจัดเก็บไว้ในฐานข้อมูลโดยผูกไว้กับชื่อผู้ใช้ และในขนาดเดียวกันก็นำค่า Salt ที่สุ่มขึ้นมานั้นไปรวมกับรหัสผ่านและทำการแฮชรหัสผ่านตามกระบวนการแฮชชิ่งและเก็บไว้ในฐานข้อมูลเช่นกัน

1. การยืนยันตัวตนนั้นจะดึงรหัสผ่านที่ผู้ใช้กรอก และ ดึงข้อมูล salt ที่ผูกไว้กับผู้ใช้ในฐานข้อมูล นำมารวมกันและคำนวณค่าแฮชซิ่งเหมือนครั้งแรกแล้ว หลังจากนั้นระบบจึงตรวจสอบว่าค่าแฮชซิ่งที่ได้จากการคำนวณนั้นตรงกับค่าแฮชซิ่งแรกที่ผู้ใช้ได้ทำการตั้งไว้แต่แรกหรือไม่ สามารถอธิบายได้ดังรูป



รูปที่ 2.10 กระบวนการยืนยันตัวตนของ Salted hash password

รูปที่ 2.10 แสดงกระบวนการยืนยันตัวตนของ Salted hash password โดยเริ่มจากผู้อ้างว่าเป็นผู้ใช้ป้อน ชื่อผู้ใช้และรหัสผ่าน ต่อจากนั้นระบบดึงค่า Salt ที่ถูกจัดเก็บไว้ในฐานข้อมูลที่ผูกไว้กับชื่อผู้ใช้ และนำค่า Salt ที่ดึงข้อมูลมานั้นไปรวมกับรหัสผ่านและทำการแฮชรหัสผ่านตามกระบวนการแฮชซิ่งและเปรียบเทียบผลของค่าแฮชซิ่งตามกระบวนการยืนยันตัวตน

แต่ข้อเสียของวิธีนี้ก็คือจัดเก็บค่า Salt หรือก็คือจัดเก็บค่า Salt ไว้ในฐานข้อมูลนั้นถือเป็นความเสี่ยงอย่างหนึ่ง ถ้าหากเก็บเป็นข้อความธรรมดา ที่ไม่ได้มีการเข้ารหัสใด ๆ เพราะหากว่าผู้โจมตีนั้นสามารถเข้าถึงฐานข้อมูลได้ไม่ว่าด้วยวิธีใดก็ตาม ผู้โจมตีก็จะสามารถอ่านค่า Salt ได้

จากทั้งหมดนี้จะเห็นได้ว่า การเก็บรหัสผ่านเป็นเรื่องที่มีความเสี่ยงที่ผู้โจมตีสามารถใช้เทคนิคการผู้โจมตีอย่าง Rainbow table ได้ ดังนั้นการรักษาความลับของรหัสผ่าน เป็นสิ่งสำคัญที่สุดในการยืนยันตัวตนด้วยรหัสผ่าน จึงควรเข้ารหัสรหัสผ่านด้วยแฮชซิ่งฟังก์ชันที่ปลอดภัยต่อการโจมตีชนิดนี้

ซึ่ง Argon2i เป็นอัลกอริธึมการแฮชรหัสผ่านที่ออกแบบมาให้สามารถปรับเปลี่ยนการคำนวณค่า RAM และ CPU ได้เพื่อป้องกันการโจมตีแบบ trade off attacks รวมถึงการโจมตีด้วย Rainbow Table เนื่องจากเหตุผลดังนี้

1. การใช้ Salt ซึ่งใน Argon2i เป็นอัลกอริธึมที่มีฟังก์ชันการเติมค่า Salt ในตัวเองโดยที่ผู้ใช้ไม่จำเป็นต้องสร้างฟังก์ชันการเติม Salt ขึ้นมาด้วยตนเอง

2. การใช้พารามิเตอร์ที่หลากหลาย ซึ่งใน Argon2i เป็นอัลกอริธึมที่สามารถปรับเปลี่ยนค่า พารามิเตอร์ที่ระบบใช้ในการคำนวณค่าแฮชชิ่งได้ ไม่ว่าจะเป็นค่า Memory ค่า Parallelism ค่า iteration และ ค่าความยาวของ Output ค่าแฮชชิ่ง ซึ่งค่าเหล่านี้จะอธิบายในบทต่อไป ดังนั้นผู้โจมตีจะต้องใช้ทรัพยากรจำนวนมากเพื่อสร้าง Rainbow Table สำหรับอินพุตพารามิเตอร์ทั้งหมดที่แตกต่างกันออกไป

2.3 อาร์กอนสอง (Argon2)

อาร์กอนสองเป็นอัลกอริทึมที่ใช้ในการแฮชชิ่งฟังก์ชันรูปแบบใหม่ออกแบบมาสำหรับสถาปัตยกรรม x86 Processor บน Intel และ AMD ถูกสร้างขึ้นมาจากการแข่งขัน Password Hashing ที่เรียกว่า Password Hashing Competition (PHC) ที่ถูกจัดขึ้นเพื่อพัฒนาการแฮชรหัสผ่านให้ทันสมัยขึ้นและ Argon2 ถูกเลือกให้เป็นผู้ชนะการแข่งขัน Password Hashing ในเดือนกรกฎาคม ค.ศ. 2015 (Biryukov, Dinu, & Khovratovich, 2017)

โดยฟังก์ชันของ Argon2 จะมีหลัก ๆ 3 ฟังก์ชันนั้น คือ ฟังก์ชัน Argon2d ฟังก์ชัน Argon2i และ ฟังก์ชัน Argon2id

2.3.1 Argon2d

Argon2d เป็นฟังก์ชันที่เร็วที่สุดในกลุ่มของฟังก์ชัน Argon2 ทั้งหมด โดยที่ 'd' ย่อมาจาก data-depending memory ที่จะใช้ นำเข้า (input) เช่น รหัสผ่านของผู้ใช้ การกำหนดการใช้งานหน่วยความจำ (Memory) ซึ่งหมายความว่ารูปแบบการเข้าถึง Memory จะแตกต่างกันไปสำหรับแต่ละรหัสผ่านที่ Input จึงเหมาะสำหรับการสร้างสินทรัพย์ดิจิทัล (Cryptocurrencies) เนื่องจาก แฮชชิ่งเป็นบทบาทที่สำคัญในการดำเนินการและรักษาความปลอดภัยของสกุลเงินดิจิทัลหลายประเภทโดยเฉพาะในบริบทของ เทคโนโลยีบล็อกเชน (Blockchain Technology) หมายถึงบัญชีผู้ใช้แยกประเภทแบบกระจายที่บันทึกธุรกรรมทั้งหมดผ่านเครือข่ายคอมพิวเตอร์ แต่ละบล็อกข้อข้อมูลในบล็อกเชนประกอบด้วยรายการธุรกรรม และบล็อกเหล่านี้เชื่อมโยงเข้าด้วยกันเป็นลูกโซ่ การแฮชชิ่งใช้เพื่อสร้างตัวระบุที่ไม่ซ้ำกันและป้องกันการเปลี่ยนแปลงข้อมูล สำหรับแต่ละบล็อก

ยกตัวอย่างเช่น แฮชชิ่งอัลกอลทึม sha256 นั้นถูกใช้กับ Cryptocurrencies ที่ใหญ่ที่สุดอย่าง Bitcoin (READ, 2022) เพื่อป้องกันการเปลี่ยนแปลงข้อมูลใน Blockchain Technology และ Argon2d ก็สามารถใช้ใน Blockchain Technolog ได้เช่นกัน (Biryukov, Dinu, & Khovratovich, 2017) หรือเว็บไซต์ออฟไลน์ที่ไม่มีโอกาสโดนการโจมตีแบบ Trade off attacks ซึ่งได้เคยกล่าวไว้แล้วในบทที่หนึ่งนั้นก็คือการเทียบจากความต่างกันของเวลาจากตอบสนองของสิ่งที่ค้นหา เมื่อเราให้ข้อมูลที่แตกต่างกันเข้าไปเป็น input และ output ที่ออกมาจะมีการใช้เวลาในการ response ที่ต่างกันออกไป (Biryukov, Dinu, & Khovratovich, 2017)

2.3.2 Argon2i

Argon2i เป็นฟังก์ชันที่ช้าที่สุดในกลุ่มของฟังก์ชัน Argon2 ทั้งหมด โดยที่ 'i' ย่อมาจาก data-independent memory ซึ่งหมายความว่าฟังก์ชันนี้ไม่ต้องพึ่งพาข้อมูลนำเข้า (input) เช่น รหัสผ่านของผู้ใช้ เพื่อการกำหนดการใช้งานหน่วยความจำ (Memory)โดยใช้วิธีการที่เรียกว่าฟังก์ชันสุ่มหลอก หรือ pseudo-random function (PRF) เป็นฟังก์ชันทางคณิตศาสตร์ที่รับ Input และคีย์ลับ แล้วสร้าง Output ที่มีลักษณะคล้ายการสุ่มแต่มิใช่การสุ่มขึ้นมา จึงเหมาะสำหรับการแฮชชิ่งเพื่อจัดเก็บรหัสผ่าน (Biryukov, Dinu, & Khovratovich, 2017) แต่การแฮชชิ่งจะทำได้ช้าลงเนื่องจากใช้หน่วยความจำของระบบมากขึ้นจากการทำ pseudo-random function เพื่อเป็นการป้องกันการโจมตีจาก จาก Trade off attacks และรวมไปถึง Rainbow Table

อีกทั้ง Argon2i ยังรองรับการทำงานแบบขนาน (Parallelism) ที่ทำให้สามารถใช้ CPU Cores ที่ทำหน้าที่ในการประมวลผลข้อมูล, ถอดรหัส, และสั่งการคำสั่งต่าง ๆ ไปยังส่วนประกอบอื่น และ CPU Threads ที่เป็นช่องทาง (Channel) ในการส่งผ่านข้อมูลที่ CPU ประมวลผลเป็นที่เรียบร้อยแล้วไปยังที่ต่าง ๆ ทำให้สามารถใช้งานคอมพิวเตอร์แบบหลาย ๆ งาน (Multitasking) อย่างการคำนวณแฮชชิ่งที่ต้องใช้การคำนวณทางคณิตศาสตร์ที่ซับซ้อน เป็นต้น (l3uch, 2564) Argon2i นั้นสามารถให้ปรับเปลี่ยน Parallelism ได้ตามต้องการ หรือก็คือเราสามารถกำหนดการแบ่งส่วนการคำนวณให้กับ CPU ได้ตามที่ต้องการ เปลี่ยนค่าของ Parallelism จะเปลี่ยนผลลัพธ์ของอัลกอริทึม (Auernhammer, 2018)

รวมไปถึง Argon2i ยังสามารถปรับการคำนวณวนซ้ำ (Iteration) ซึ่งหมายถึงว่าเราสามารถกำหนดครั้งในการวนซ้ำในการคำนวณค่าแฮชชิ่งได้ขึ้น (Biryukov, Dinu, & Khovratovich, 2017) วัตถุประสงค์หลักของ Iteration นั้นคือการทำให้การคำนวณแฮชชิ่งนั้นช้าลง และทำให้ผู้โจมตีนั้นโจมตีได้ยากขึ้น เพราะการวนซ้ำหลายครั้งจะเพิ่มเวลาในการยืนยันตัวตนมากยิ่งขึ้นทำให้การ Brute Force Attack นั้นใช้เวลามากขึ้น ดังนั้นจึงเป็นเหตุผลที่ทำให้การโจมตีนั้นยากขึ้น (Security, 2021) ซึ่งจะคล้ายกับการยืนยันตัวตนด้วยการแฮชชิ่งสองครั้งที่เคยกล่าวไว้แล้ว ซึ่งก็คือเราสามารถกำหนดให้ Argon2i นั้นแฮชชิ่งซ้ำได้

เช่นเดียวกันกับฟังก์ชันอื่น ๆ อีก 2 ตัวของ Argon2 นั้นสามารถปรับขนาดของผลของค่าแฮชชิ่ง (Output) ได้ว่าจะออกมาเป็นกี่บิต ซึ่งจะแตกต่างจากอัลกอริทึมอื่น ๆ เช่น SHA256 ที่จะไม่สามารถเปลี่ยนแปลง Output ได้เลย มีผลต่อการที่ผู้โจมตีนั้นจะสร้าง Rainbow Table ได้ยากขึ้นเนื่องจากต้องคำนึงถึงขนาดของ Output ด้วย ในการสร้าง ทำให้เป็นไปได้ยากที่จะสร้าง Rainbow Table ที่โจมตี Argon2 ได้สำเร็จ

2.3.3 Argon2id

เป็นฟังก์ชันที่แบ่งครึ่งการทำงาน (Hybrid construction) ระหว่าง ฟังก์ชัน Argon2i และ ฟังก์ชัน Argon2d โดยครึ่งแรกจะแฮชชิ่ง Argon2i ก่อนและหลังจากนั้นอีกครึ่งหลังที่เหลือจะแฮชชิ่ง Argon2d (Biryukov, Dinu, & Khovratovich, 2017) ซึ่งหมายความว่า Argon2id นั้นเป็นฟังก์ชันที่ผสมผสานกันระหว่าง Argon2i และ Argon2d โดยที่ การกำหนดการใช้งานหน่วยความจำ (Memory) ที่ไม่ขึ้นกับ input (การทำงานแบบ Argon2i) สำหรับการวนซ้ำครั้งแรก จากนั้นจึงสลับไปใช้การกำหนดการใช้งานหน่วยความจำ (Memory) โดยอาศัยข้อมูล(การทำงานแบบ Argon2d) สำหรับการวนซ้ำครั้งต่อไป สลับกันไปเรื่อย ๆ (Alex Biryukov, 2021)

เนื่องจากคุณสมบัติที่ได้กล่าวมาข้างต้นของ Argon2i ที่ถูกสร้างขึ้นมาเพื่อแฮชชิ่งรหัสผ่านโดยเฉพาะ และ ผู้พัฒนามุ่งเน้นไปที่การป้องกันการโจมตีจาก Trade off attacks โดยตรงและรวมไปถึงการโจมตีที่ผู้วิจัยได้เคยกล่าวมาข้างต้น เช่น Rainbow table , Collisions หรือ Packet sniffing ได้ ดังนั้นในงานวิจัยนี้จึงมุ่งเน้นไปที่การศึกษาและวิจัย Argon2i เป็นหลัก

2.4 อินพุต (input) ทั้งหมดของ Argon2i

การทำงานของ Argon2i นั้นจำเป็นต้องได้รับ อินพุต (input) ต่าง ๆ เข้าไปเพื่อใช้ในการคำนวณทางคณิตศาสตร์ เพื่อให้ได้ผลลัพธ์เป็นค่าแฮชชิ่ง โดย อินพุต นั้นจะถูกเรียกว่าพารามิเตอร์ทั้งหมด 10 พารามิเตอร์ด้วยกันและสามารถอธิบาย ดังนี้

1. Plain Text Input (P) รหัสผ่านตั้งต้น หมายถึงข้อความรหัสผ่านของผู้ใช้ที่ต้องการแปลงเป็นค่าแฮชชิ่ง โดยที่พารามิเตอร์รหัสผ่านนั้นไม่มีขั้นต่ำของความยาวค่ารหัสผ่าน
2. Salt (S) ค่า Salt คือค่าที่เติมไปในรหัสผ่าน ซึ่งได้เคยอธิบายไว้ก่อนแล้ว โดยพารามิเตอร์ ชนิดนี้นั้นหมายถึงข้อความที่เราจะใช้เพื่อเติมเข้าไปในรหัสผ่าน ซึ่ง Argon2i จะมีฟังก์ชันที่สร้างมาเพื่อรวม ค่า Salt กับ รหัสผ่านเข้าด้วยกันอยู่แล้ว โดยที่ผู้ใช้ไม่จำเป็นมารวมกับรหัสผ่านด้วยตนเองโดยที่ความยาว ขั้นต่ำที่ Argon2 กำหนดของค่า Salt คือ “8 ตัวอักษร”

3. Memory size (m) ค่าหน่วยความจำ ซึ่งได้เคยอธิบายไว้ก่อนแล้วว่า Argon2i นั้นมีการกำหนดการใช้งานหน่วยความจำ (Memory) หรือก็คือค่าพารามิเตอร์นี้ใช้เพื่อการกำหนดให้ Argon2i นั้นใช้ Memory หรือ RAM ของเครื่องคอมพิวเตอร์เท่าไหร่สำหรับการคำนวณ โดยค่า Memory size นั้น สามารถเลือกหน่วยการคำนวณได้ 2 แบบด้วยกันคือ KiB(k) หรือ MB(m) ก็ได้ โดยที่ ขั้นต่ำที่ Argon2 กำหนด ของค่านี้คือ “256 KiB”

4. Iteration number (t) จำนวนการวนซ้ำ จากที่ได้เคยอธิบายไว้แล้วว่า Argon2i นั้นสามารถที่จะวนซ้ำเพื่อแฮชชิ่งค่าแฮชชิ่งเดิมที่แฮชชิ่งไว้แล้วได้ โดยพารามิเตอร์ชนิดนี้นั้นมีเพื่อการกำหนดการวนซ้ำเพื่อนำค่าแฮชชิ่งที่ได้นั้นมาคำนวณใหม่อีกกี่ครั้งได้ตามที่ผู้ใช้ต้องการ โดยที่ขั้นต่ำของค่านี้คือ “1 (t)”

5. parallelism (p) ระดับความขนานหรือจำนวน Threads พร้อมกันที่อัลกอริทึมจะใช้ในการแฮชชิ่ง จากที่ได้เคยอธิบายไว้แล้วว่า Argon2i นั้นสามารถที่จะปรับการแบ่งการทำงานของ CPU ของเครื่องคอมพิวเตอร์ได้ โดยพารามิเตอร์ชนิดนี้นั้นมีเพื่อการกำหนดการแบ่งการทำงานไปที่ CPU ที่จะทำการไปพร้อม ๆ ในแต่ละรอบการทำงานของ CPU กันเท่าไหร่ โดยที่ขั้นต่ำของค่านี้คือ “1 (P)”

6. Tag length ความยาวแท็ก จากที่ได้เคยอธิบายไว้แล้วว่า Argon2i นั้นสามารถที่จะปรับขนาดของ Output ที่จะออกมาได้ หรือก็คือ เราสามารถปรับขนาดของค่าแฮชชิ่งที่ออกมาได้ โดยพารามิเตอร์ชนิดนี้นั้นมีเพื่อกำหนดของขนาดของ Output ได้ซึ่งจะมีหน่วยเป็น “บิต” ซึ่งก็คือผลลัพธ์ค่าแฮชชิ่งว่าจะออกมาเป็นความยาวเท่าไหร่ โดยที่ขั้นต่ำของค่านี้คือ “32 บิต”

7. Version number (v) เลขเวอร์ชั่นของ Argon2i ซึ่งผู้พัฒนาได้สร้างขึ้นมาเพื่อให้ผู้ใช้สามารถกำหนดเวอร์ชั่นของ Argon2i ได้ด้วยตนเอง ซึ่ง ณ เวลาที่เขียนนี้จะมีให้เลือก 2 เวอร์ชั่นก็คือ เวอร์ชั่น 10 และ เวอร์ชั่น 13 เพิ่มความปลอดภัยในบริบทของการโจมตีที่เรียกว่า "side-channel attacks" หมายถึงการโจมตีช่องทางด้านข้างเป็นช่องโหว่ด้านความปลอดภัยที่มีจุดมุ่งหมายเพื่อรวบรวมข้อมูลจากการทำงานของระบบโดยการวัดหรือใช้ประโยชน์จากผลกระทบทางอ้อมของระบบหรือฮาร์ดแวร์ของระบบ เช่น กระแสไฟฟ้าที่ใช้ ความร้อน หรือเสียงการทำงานของคอมพิวเตอร์ คล้ายกับ Trade off attacks

8. Secret Value K ได้มาจาก Plain Text Input (P) และ Salt (S) ซึ่งเป็นค่าสุ่มที่สร้างขึ้นสำหรับการแฮชชิ่งพารามิเตอร์ ที่ใช้เพื่อการคำนวณ ฟังก์ชันสุ่มหลอก หรือ pseudo-random function (PRF) ค่าพารามิเตอร์นี้ใน Argon2i เวอร์ชัน 13 ที่เป็นล่าสุดในตอนนี้ไม่สามารถปรับเปลี่ยนได้

9. Associated data (X) คือข้อมูลเพิ่มเติมที่สามารถระบุเป็น Input ให้กับแฮชชิ่งฟังก์ชัน ใช้เพื่อเพิ่มข้อมูลเพิ่มเติมให้กับรหัสผ่าน หรือก็คือเราสามารถใส่ Tag ให้กับรหัสผ่านได้ เช่น การระบุว่ารหัสผ่านนี้เป็นของผู้ใช้คนไหน หรือ รหัสผ่านนี้ถูกสร้างเมื่อใด ค่าพารามิเตอร์นี้ใน Argon2i เวอร์ชัน 13 ที่เป็นล่าสุดในตอนนี้ไม่สามารถปรับเปลี่ยนได้

10. Type y เลือกชนิดของ Argon2 ที่ประกอบไปด้วย Argon2i Argon2d Argon2id ค่าพารามิเตอร์นี้ใช้เพื่อปรับเปลี่ยนฟังก์ชันการทำงานของ Argon2i ไปใช้เป็นชนิดอื่น

การวิจัยนี้จะทำการทดลองเพื่อหาค่า ค่าพารามิเตอร์ของ Argon2i ใช้เวลาในการประมวลผลน้อยที่สุดซึ่ง หลังจากที่เราได้ พารามิเตอร์ ที่มี ใช้เวลาในการประมวลผลน้อยที่สุดแล้ว ควรเป็นต้องพิสูจน์ว่า ค่าพารามิเตอร์เหล่านั้น ยังคงมีความปลอดภัยเพียงพอต่อการใช้งานหรือไม่โดยการทดสอบด้านความปลอดภัยออกมาเป็นค่าหนึ่ง แล้วนำผลทดลองเหล่านั้นไปเปรียบเทียบกับแฮชชิ่งอัลกอริทึมเดิมที่เคยมีมาก่อนแล้วว่ามีผลแตกต่างกันมากน้อยเพียงไหน เพื่อเป็นค่าพิสูจน์ว่า การใช้ค่าพารามิเตอร์เหล่านี้ มิได้ทำให้ความปลอดภัยนั้นด้อยไปกว่าการใช้แฮชชิ่งอัลกอริทึมเดิมที่เคยมีมาก่อนแล้ว ในงานวิจัยนี้จึงได้ยกการเปรียบเทียบผลต่างบิตของค่าแฮชชิ่ง Avalanche effect มาทดลองเพื่อหาประสิทธิภาพด้านความปลอดภัยของ Argon2i

2.5 Avalanche effect

นิยามของ The Avalanche effect คือการเปรียบเทียบคุณลักษณะที่สำคัญของฟังก์ชันแฮชชิ่ง โดยการเปลี่ยนแปลงเล็กน้อยในอินพุตของฟังก์ชันแฮชชิ่งจะส่งผลให้เอาต์พุตแตกต่างกันอย่างมีนัยสำคัญ โดยที่ Avalanche effect จะมุ่งเน้นไปที่การวิเคราะห์การสุ่มของค่าแฮชชิ่งและความสามารถของอัลกอริทึมกับการต้านทานการชนกันของค่าแฮชชิ่ง (Collision)

หรือในอีกนิยามนั้นคือ Avalanche Effects วิธีการนี้จะแสดงค่าแฮชชิ่งในระดับบิตเมื่อข้อมูลตั้งต้นหรือคีย์มีการเปลี่ยนแปลงเพียงเล็กน้อยหรือเพียง 1 บิต จะทำให้ค่าแฮชชิ่งที่ได้เปลี่ยนแปลงไปมากดังนั้นถ้าค่าแฮชชิ่งที่ได้จากอัลกอริทึมการเข้ารหัสใดๆ มีความแตกต่างกันมากจะส่งผลให้การคาดเดารหัสผ่านทำได้ยากซึ่งหมายถึงอัลกอริทึมนั้นจะมีความแข็งแรง มากด้วย (แซ่ตั้ง & บุญครอง, 2017)

โดยสรุปแล้วนั้น Avalanche effect คือการทดลองเพื่อวัดความแตกต่างกันระหว่างแค่แฮชชิ่ง 2 ค่า ที่เปลี่ยนแปลงข้อความตั้งต้น หรือ รหัสผ่านตั้งต้นไปเล็กน้อย เพียงแค่ 1 บิต นั้นจะสร้างความแตกต่างกันได้มากน้อยเพียงใด เพื่อการทดสอบในด้านความปลอดภัยในด้าน การต้านทานการชนกันของค่าแฮชชิ่ง (Collision) และ ความสามารถในการสุ่มเพื่อสร้างค่าแฮชชิ่ง ของอัลกอริทึมแฮชชิ่ง นั้น

ด้วยวิธีนี้จะทำให้สามารถประเมินความแข็งแกร่งของอัลกอริทึม Argon2i ที่ใช้พารามิเตอร์ ที่ทำให้การแฮชชิ่งมีเวลาที่สั้นที่สุดได้

2.6 งานวิจัยที่เกี่ยวข้อง

ผู้วิจัยได้ทำการศึกษางานวิจัยที่เกี่ยวข้องในเรื่องของ Avalanche effect และ การโจมตีฐานข้อมูลที่เก็บข้อมูลรหัสผ่าน และ งานวิจัยที่เกี่ยวข้องกับการเลือกพารามิเตอร์ที่เหมาะสมกับ Argon2di และมีรายละเอียดของงานวิจัยดังนี้

2.6.1 งานวิจัยที่เกี่ยวข้องกับ Avalanche effect

งานวิจัยที่เกี่ยวข้องกับการทดลอง Avalanche effect ที่ได้ยกมาอธิบายนิยามของ Avalanche effect ก่อนหน้านี้ (แซ่ตั้ง & บุญครอง, 2017) ได้ใช้วิธีการ เปลี่ยนแปลงข้อมูล 1 บิตจากข้อมูลตั้งต้น หรือ รหัสผ่านตั้งต้น สามารถแสดงได้ดังภาพ

ลำดับ	กระบวนการ	ข้อมูลตั้งต้น	ข้อมูลถูกเปลี่ยนแปลง 1 บิต
1	ข้อมูล	SECURITY	SECUSITY
2	Ascii	R -> 0 1 0 1 0 0 1 0	S -> 0 1 0 1 0 0 1 1
3	คำสั่งที่ใช้	openssl dgst md5 –out hashvalue.txt test.txt	openssl dgst md5 –out hashvalue2.txt test2.txt
4	ค่าแฮช (Hex)	219ce424cc367fba 8a5a0210792d4dc0	ee30b02869e3f479 acb45810cabc0dad
5	ค่าแฮช (Binary)	0010000110011100 1110010000100100 1100110000110110 0111111110111010 1000101001011010 0000001000010000 0111100100101101 0100110111000000	1110111000110000 1011000000101000 0110100111100011 1111010001111001 1010110010110100 0101100000010000 1100101010111100 0000110110101101
6	ผลต่างบิต	61	

รูปที่ 2.11 วิธีการทดลอง Avalanche effect ตัวอย่างการศึกษางานวิจัยอื่น (แซ่ตั้ง & บุญครอง, 2017)

รูปที่ 2.11 แสดงวิธีการทดลอง Avalanche effect ตัวอย่างการศึกษางานวิจัยอื่น ดังนี้ข้อมูลตั้งต้นหรือรหัสผ่านตั้งต้น คือ“SECURITY”เปลี่ยนเป็น“SECUSITY” จาก R เปลี่ยนเป็น S ซึ่ง R ระดับบิตคือ 01010010 และ S ระดับบิตคือ 01010011 มีการเปลี่ยน บิต สุดท้าย 1 บิตจาก 0 เป็น 1 นำเข้าสู่แฮชชิ่งฟังก์ชัน ได้ค่าแฮชออกมา อยู่ในรูปเลขฐาน 16 จากนั้นทำการแปลงค่าแฮชที่ได้ให้อยู่ในรูปฐาน 2 และ ได้ผลต่างของบิต เป็น 61 ตัว

ซึ่งผลของงานวิจัยตัวอย่างที่ได้ศึกษามานั้นได้ทำการทดลองกับอัลกอริทึมได้แก่ MD5 SHA-1 SHA-256 และ SHA-512 และผลได้ออกมารูปภาพนี้ดังนี้

	ผลต่างของบิต (เปอร์เซ็นต์)					
	Uppercase	Lowercase	Number	Symbols	Mixes	Average
MD5	39.84	46.09	53.90	51.56	47.65	47.81
SHA-1	46.87	47.50	48.12	54.37	50.00	49.37
SHA-256	47.26	49.22	48.05	51.95	48.82	49.06
SHA-512	48.43	45.11	50.00	50.00	49.02	48.51
Average	45.60	46.98	50.02	51.97	48.87	

รูปที่ 2.12 ผลการทดลอง Avalanche effect ตัวอย่างการศึกษางานวิจัยอื่น (แซ่ตั้ง & บุญครอง, 2017)

รูปที่ 2.12 ผลการทดลอง Avalanche effect ในตัวอย่างการศึกษางานวิจัยแสดงให้เห็นว่าอัลกอริทึมต่าง ๆ สามารถมีผลต่างในการแปลงข้อมูลเมื่อมีการเปลี่ยนแปลงตัวอักษรพิมพ์ใหญ่, ตัวอักษรพิมพ์เล็ก, ตัวเลข, สัญลักษณ์และแบบผสม และนำความแตกต่างนั้นมาเปลี่ยนเป็นเปอร์เซ็นต์และหาค่าเฉลี่ย จะเห็นได้ว่าผลต่ายที่มากที่สุดคือ SHA-1 และ น้อยที่สุดคือ MD5

2.6.2 งานวิจัยที่เกี่ยวข้องกับโจมตีฐานข้อมูลที่เก็บข้อมูลรหัสผ่าน

งานวิจัยที่เกี่ยวข้องกับวิธีการโจมตีฐานข้อมูลที่เก็บข้อมูลรหัสผ่านของระบบของที่ผู้ไม่ประสงค์ดีที่ใช้เพื่อเข้าถึงภายในระบบ ซึ่งมีดังต่อไปนี้ (Cases, 2022)

SQL injection เป็น รูปแบบการโจมตีของผู้โจมตีโดยอาศัยช่องโหว่ของระบบ ในฟังก์ชันการรับข้อมูล input ของผู้ใช้ที่ไม่ได้มีการตรวจสอบ input ให้มีเฉพาะข้อความจริง ๆ เท่านั้น ผู้โจมตีทำให้สามารถใส่ คำสั่งของฐานข้อมูล (SQL) ซึ่ง SQL เป็นภาษาที่ใช้เพื่อใช้งานฐานข้อมูลของระบบซึ่งเก็บข้อมูลรหัสผ่านของผู้ใช้อยู่ในนั้น ที่จะเปลี่ยนแปลงคำสั่งเดิมที่มีอยู่ให้เป็นไปในอีกทิศทางหนึ่งที่ผู้โจมตีต้องการ เช่น ผู้ใช้ปกติจะ Input ข้อมูลเป็นข้อความปกติ เช่น คำค้นหา ชื่อ

ผู้ใช้ หรือ รหัสผ่าน แต่ผู้โจมตีนั้นใช้เป็นข้อความชนิดที่ยกเลิกคำสั่ง SQL เดิมและเปลี่ยนคำสั่ง SQL ใหม่เข้าไป เพื่อที่จะสามารถดึงข้อมูลออกมาจากฐานข้อมูลได้ หรือแม้กระทั่ง ใช้คำสั่ง SQL เพื่อ เพิ่ม ลบ แก้ไข ข้อมูลได้ทั้งหมด (หลงหัน, 2014) ซึ่งหากถูกโจมตีด้วยวิธีนี้กับระบบฐานข้อมูล จะทำให้ความลับของรหัสผ่านผู้ใช้จะสามารถถูกละเมิดได้หากไม่ได้มีการเข้ารหัสรหัสผ่านของผู้ใช้ สามารถป้องกันได้ด้วยการใช้ตรวจสอบ Input ทุกที่ที่เข้ามาในระบบ ให้สามารถแยกแยะได้ว่า Input ประเภทไหนเป็นข้อความปกติ หรือเป็นคำสั่ง SQL

Stolen backup tapes คือการโจมตีด้วยการเข้าถึง Backup ข้อมูลจากระบบ ซึ่งในข้อมูลนั้นจะมีข้อมูลชื่อผู้ใช้และรหัสผ่านอยู่ในนั้น และเนื่องจากใน backup ข้อมูลนั้นส่วนใหญ่จะไม่มีการเข้ารหัสหรือการรักษาความปลอดภัยข้อมูล หรือถ้าหากมีก็ไม่ปลอดภัยเทียบเท่ากับระบบหลักที่ใช้งานอยู่ (Cases, 2022) ซึ่งหากถูกโจมตีด้วยวิธีนี้กับระบบฐานข้อมูลสำรอง (Backup) จะทำให้ความลับของรหัสผ่านผู้ใช้จะสามารถถูกละเมิดได้หากไม่ได้มีการเข้ารหัสรหัสผ่านของผู้ใช้ สามารถป้องกันได้ด้วยการตระหนักถึงช่องโหว่ของ Backup และเข้ารหัสหรือเพิ่มความปลอดภัยให้กับระบบฐานข้อมูลสำรอง (Backup)

2.6.3 งานวิจัยที่เกี่ยวข้องกับการเลือกพารามิเตอร์ที่เหมาะสมกับ Argon2di

งานวิจัยที่ได้พยายามศึกษาหาคำตอบสำหรับการเลือกพารามิเตอร์ที่เหมาะสม หมายถึงการทำให้ Argon2di นั้นมีความเร็วไม่เกิน 0.5 วินาที ซึ่งอ้างอิงจากผู้สร้าง Argon2 (Biryukov, Dinu, & Khovratovich, 2017) ว่ากระบวนการแฮชรหัสผ่านทั้งหมดในระบบนั้นไม่ควรที่จะใช้เวลาเกินจากนี้ โดยสามารถอธิบายได้ดังนี้ เป้าหมายของบทความนี้คือการสร้างสมดุลระหว่างการเพิ่มเวลาสูงสุดที่ใช้ในการคำนวณแฮชซึ่งเพื่อป้องกันไม่ให้ผู้โจมตีได้โจมตีด้วยการ brute force attack ที่ได้เคยกล่าวไว้แล้วก่อนหน้านี้ และ ลดเวลาในการคำนวณแฮชซึ่งให้ได้น้อยที่สุด เพื่อให้ระบบไม่ต้องหยุดชะงักเมื่อมีผู้ใช้ต้องการเข้าสู่ระบบ

ตารางที่ 2.1 แสดงค่าพารามิเตอร์ที่แนะนำโดยคุณ Bryan Burman (Burman, 2019)

พารามิเตอร์	ค่าที่แนะนำ
Parallelism (p)	ควรขึ้นอยู่กับจำนวน Threads ของ CPU คือจำนวน Threads x 2 เช่น CPU 8 Threads ค่า p จะเท่ากับ 16
Memory size (m)	ให้ใช้ค่าสูงสุดเท่าที่ทำได้ แต่ไม่ให้การคำนวณเกินเวลา 0.5 วินาที
Iteration number (t)	ในบทความไม่ได้ข้อสรุปชัดเจน แต่ในงานวิจัย Argon2 แนะนำใช้มากกว่า 2

ตารางที่ 2.1 แสดงให้เห็นค่าที่แนะนำของพารามิเตอร์ที่ทำการปรับเปลี่ยนเพื่อหาค่าที่เหมาะสม ได้แก่ Parallelism (p) ควรขึ้นอยู่กับจำนวน Threads ของ CPU คือ 2 เท่าของ Threads เช่น ฮาร์ดแวร์ที่ใช้คำนวณ ตาม Spec นั้นมี CPU 8 Threads ให้เราเลือกใช้ค่า Parallelism เป็น 16 (P)

ค่า Memory size (m) ให้ใช้ค่าสูงสุดเท่าที่ทำได้ แต่ไม่ให้การคำนวณเกินเวลา 0.5 วินาที เพราะตามอ้างอิงจากงานวิจัยของ Argon2 แนะนำให้ใช้เวลาไม่เกิน 0.5 วินาที (Biryukov, Dinu, & Khovratovich, 2017) และสุดท้าย ค่า Iteration number (t) ในบทความไม่ได้ข้อสรุปชัดเจน แต่ในงานวิจัย Argon2 แนะนำใช้มากกว่า 2 สิ่งที่น่าจะเป็นประโยชน์สูงสุดกับงานวิจัยนี้คือผลลัพธ์ของค่า Parallelism (p) ที่ต้องใช้เป็น 2 เท่าของจำนวน Threads ของ CPU นั้นเป็นหนึ่งในข้อสงสัยในการพิสูจน์ของงานวิจัยนี้ว่าเหตุผลนี้เป็นจริงแค่ไหน

ซึ่งการค้นคว้าอิสระนี้แตกต่างจากงานวิจัยดังกล่าวที่การใช้ฟังก์ชัน Argon2id ไม่ใช่ Argon2i และยังไม่มีข้อสรุปที่ชัดเจนอยู่หลายจุดด้วยกันรวมถึงยังมีพารามิเตอร์อีกหลายตัวที่ยังไม่ได้ทำการทดลองเปลี่ยนปรับเปลี่ยน

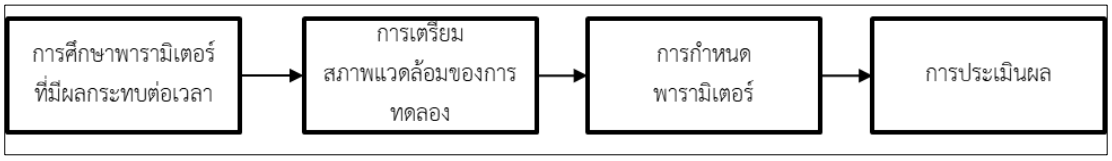
บทที่ 3
วิธีดำเนินการวิจัย

เนื่องจาก Argon2I เป็นอัลกอริทึมสมัยใหม่สำหรับการแฮชรหัสผ่านอย่างปลอดภัย (Portwood, 2016) และยังเป็นผู้ชนะการแข่งขัน Password Hashing Competition (PHC) เป็นการแข่งขันแบบเปิด ซึ่งเป็นกระบวนการแบบเดียวกับการแข่งขัน AES และ SHA-3 ของ NIST และเป็นวิธีที่มีประสิทธิภาพสูงสุดในด้าน Information Security มีความเร็วในการคำนวณการใช้งาน CPU สามารถป้องกันการโจมตีแบบ trade off attacks ที่ได้เคยกล่าวไว้ก่อนหน้านี้แล้ว ด้านความเรียบง่าย(Simplicity) ที่หมายถึงความชัดเจนโดยรวมของโครงร่างการคำนวณและความง่ายในการใช้งาน และด้านฟังก์ชันการทำงาน (Functionality) ที่หมายถึงความสามารถในการแปลงแฮชชิ่งที่มีอยู่เป็นรหัสผ่านดั้งเดิมก่อนที่จะถูกแฮชชิ่งได้ หรือหมายความว่าสามารถในการแปลค่าแฮชกลับคืนได้เพื่อใช้ในการตรวจสอบการทำงานของค่าแฮชชิ่ง (Competition, 2019) แต่ Argon2i ยังมีข้อจำกัดในด้านของเวลาในกระบวนการที่ต้องใช้เวลาอยู่พอสมควรหากไม่เลือกพารามิเตอร์ที่เหมาะสม และในกรณีที่ใช้งานกับระบบเว็บแอปพลิเคชันอาจจะทำให้ผู้ใช้งานไม่พอใจ

ดังนั้นการที่จะได้มาซึ่งเวลาที่เหมาะสมในการประมวลผลนั้นสามารถทำได้ด้วยการเลือกใช้พารามิเตอร์ที่ถูกต้องและเหมาะสมกับเครื่องคอมพิวเตอร์ที่ใช้ในการเป็นเว็บเซิร์ฟเวอร์ของระบบเว็บแอปพลิเคชัน ซึ่งเป็นเป้าหมายของโครงการวิจัยนี้

3.1 วิธีวิจัย

การวิจัยนี้จะทำการทดลองการเลือกพารามิเตอร์ที่เหมาะสมเพื่อลดเวลาการแฮชชิ่งด้วย Argon 2I เพื่อลดปัญหาในการประมวลผล ที่จะเกิดขึ้นหากจะนำไปใช้งานในระบบต่าง ๆ โดยกระบวนการทดลองจะดำเนินการดังระบุในรูปที่ 3.1



รูปที่ 3.1 แสดงลำดับขั้นตอนของการทดลอง

รูปที่ 3.1 แสดงลำดับขั้นตอนวิธีดำเนินการวิจัยโดย เริ่มตั้งแต่ ขั้นตอนที่ 1 คือ การศึกษาพารามิเตอร์ที่มีผลกระทบต่อเวลา ซึ่งในขั้นตอนนี้ ผู้วิจัยจะดำเนินการศึกษาหาข้อมูลเกี่ยวกับพารามิเตอร์ต่าง ๆ ที่มีผลต่อเวลาการประมวลผลของการคำนวณค่าแฮชชิ่งของ Argon2I , ขั้นตอนต่อไป คือ การออกแบบการทดลอง ซึ่งหมายถึงการกำหนดว่าจะใช้เครื่องมือใดในการทดลองและการติดตั้งโปรแกรมต่าง ๆ รวมถึงการเตรียมสภาพแวดล้อมของการทดลอง เช่น การทดลองบนเครื่องคอมพิวเตอร์ส่วนบุคคล การทดลองบนเครื่องคอมพิวเตอร์เสมือน หรือ การทดลองบนเครื่องแม่ข่าย เป็นต้น จากนั้น จะเป็นการกำหนดพารามิเตอร์ และการปรับค่าของพารามิเตอร์เหล่านั้น โดยในขั้นตอนนี้ ผู้วิจัยจะกำหนดพารามิเตอร์ที่จะใช้ทดลอง เพื่อหาผลลัพธ์และเวลาในการประมวลผลของ Argon2I ที่มีประสิทธิภาพในการด้านความเร็ว ให้สามารถทำความเร็วในการแฮชชิ่งให้ได้มากที่สุด และขั้นตอนสุดท้าย คือ การประเมินผล ซึ่งก็คือการกำหนด และการอธิบายวิธีการประเมิน ไปพร้อม ๆ กับ การแสดงผลและอภิปรายผลของการทดลอง

3.2 ศึกษาพารามิเตอร์ที่มีผลกระทบต่อเวลา

จากการศึกษาพารามิเตอร์ทั้งหมด (Biryukov, Dinu, & Khovratovich, 2017) ที่ Argon2I มีนั้น ปรากฏว่า มีพารามิเตอร์ที่สามารถปรับแก้ไขได้ ดังนี้

ตารางที่ 3.1 พารามิเตอร์ที่สามารถปรับค่าได้ของ Argon2i

พารามิเตอร์	ความหมาย
Salt (ความยาวตัวอักษร)	ค่าที่ถูกสุ่มขึ้นมาสำหรับเติมไปในรหัสผ่านที่เติมไปในรหัสผ่านตั้งต้น
Memory size (k)	ค่าหน่วยความจำที่ของคอมพิวเตอร์ที่คำนวณหน่วยเป็น KiB
Iteration number (t)	จำนวนการวนซ้ำในการแฮชชิ่ง
Parallelism (p)	ระดับความขนานหรือจำนวน Threads ของ CPU
Tag length (l)	ความยาวของค่าแฮชชิ่งหน่วยเป็น Byte

ตารางที่ 3.1 แสดงพารามิเตอร์ต่าง ๆ ของกระบวนการแฮชชิ่งแบบ Argon2 พร้อมทั้งความหมาย ของตัวโปรแกรม Argon 2 โดยพารามิเตอร์เหล่านี้ เป็นพารามิเตอร์ที่สามารถปรับตั้งค่าได้ทั้งหมด ซึ่งสามารถอธิบายเพิ่มเติม ได้ดังนี้

พารามิเตอร์ตัวแรกที่สามารถปรับเปลี่ยนค่าได้ ได้แก่ Salt คือค่าหนึ่ง ที่ถูกสร้างขึ้นมาแบบ random และจะถูกนำไป วางไว้อยู่ด้านหน้าหรือด้านหลังของรหัสผ่าน (Plaintext) แล้วนำรหัสผ่านและ Salt ที่รวมกันแล้วนั้น นำไปแฮชชิ่งทั้งหมด ยกตัวอย่างเช่น Plaintext = 123456 ค่า Salt = LV)wuS5h ดังนั้นค่าที่จะนำไปแฮชชิงคือ 123456LV)wuS5h

พารามิเตอร์ตัวต่อมา คือ ค่า Memory size คือค่าความจำที่ได้จาก RAM ของเครื่องคอมพิวเตอร์ที่เราอนุญาตให้ Argon2I นำไปใช้ได้ในการคำนวณแต่ละครั้ง ยกตัวอย่างเช่น คอมพิวเตอร์มี 8GB ก็จะสามารถกำหนดให้ ค่าเริ่มต้นที่ Argon2i กำหนดไว้อัตโนมัติหากไม่ได้แก้ไข คือ 4096KiB (ค่าเริ่มต้น)) และ ขึ้นต่ำคือ 256 KiB

พารามิเตอร์ตัวที่สาม คือ ค่า Iteration number หรือ จำนวนการวนซ้ำในการแฮชชิ่ง ซึ่งพารามิเตอร์ตัวนี้ หมายถึง ค่าที่สั่งให้โปรแกรมทำคำนวณหรือประมวลผล Argon2I ซ้ำเป็นจำนวนกี่ครั้งตามที่กำหนด ค่าเริ่มต้นที่ Argon2i กำหนดไว้อัตโนมัติหากไม่ได้แก้ไข คือ 3t (ค่าเริ่มต้น) และขึ้นต่ำคือ 1t

พารามิเตอร์ตัวที่สี่ คือ ค่า Parallelism ซึ่งหมายถึง ค่าระดับคำนวณแบบขนานกันของ CPU ซึ่งจะอนุญาตให้ Argon2I คำนวณแบบพร้อม ๆ กันแบบขนานได้กี่ Threads ของ CPU ที่เครื่องคอมพิวเตอร์เรามีตามที่เราระบุไว้ได้ ค่าเริ่มต้นที่ Argon2i กำหนดไว้อัตโนมัติหากไม่ได้แก้ไข คือ 1p (ค่าเริ่มต้น) และ ขึ้นต่ำคือ 1p เช่นกัน

พารามิเตอร์สุดท้าย คือ ค่า Tag length (l) ซึ่งพารามิเตอร์ตัวนี้ สามารถให้ผู้ใช้งานกำหนดความยาวของ Output ที่โปรแกรมคำนวณค่าแฮชชิ่งออกมาให้เป็นความยาวกี่ตัวอักษรตามที่ระบบค่าเริ่มต้นที่ Argon2i กำหนดไว้อัตโนมัติหากไม่ได้แก้ไข คือ 32l (ค่าเริ่มต้น) หรือ 62 ตัวอักษร และขึ้นต่ำคือ 4l หรือ หรือ 8 ตัวอักษร

3.3 การเตรียมสภาพแวดล้อมสำหรับการทดลอง

การวิจัยจะเป็นการวิจัยในระบบจำลอง Linux Ubuntu VM ผ่านระบบปฏิบัติการ Windows 10 Pro Version 21H2 ที่มีข้อกำหนดของเครื่อง ดังนี้

CPU: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz 4

GPU: GeForce GTX 1050 Mobile 2 GB จำนวน 4 Cores 4 Threads: โดยแบ่งให้ระบบ Hyper-V เพื่อ Run ระบบ Linux Ubuntu 1 Threads (1 Cores)

RAM: 32.0 GB และ แบ่งให้ระบบ Hyper-V เพื่อ Run ระบบ Linux Ubuntu 8 GB

ROM: 1000 GB และ แบ่งให้ระบบ Hyper-V เพื่อ Run ระบบ Linux Ubuntu 10 GB

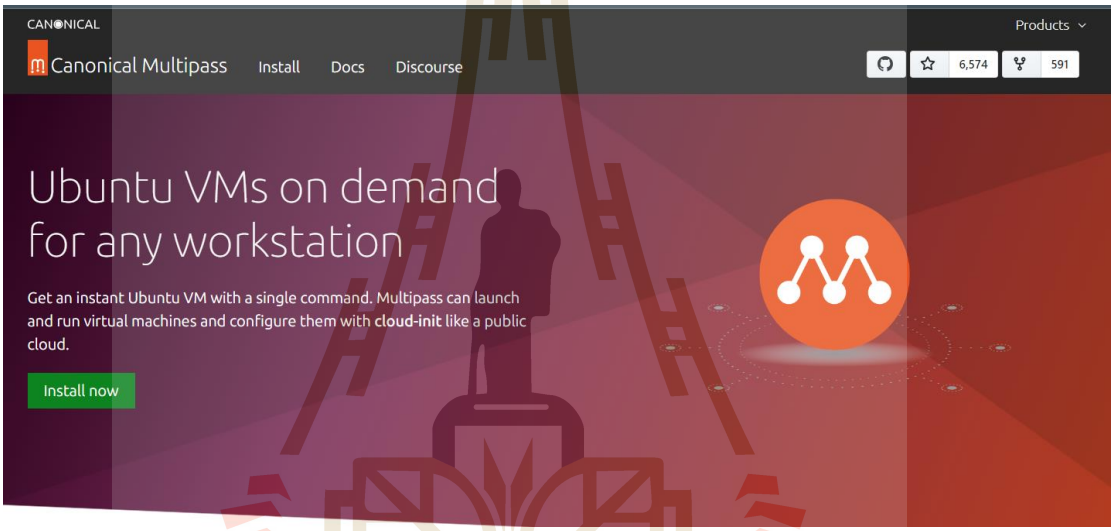
โปรแกรมที่ใช้ทั้งหมดมี 2 โปรแกรม ได้แก่

1. Argon2 จาก GitHub ชื่อ P-H-C/phc-winner-argon2 ถูกพัฒนาด้วยภาษา C URL : https://github.com/p-h-c/phc-winner-argon2 โดยช่องทางการดาวน์โหลดนี้เป็นช่องทางที่เป็นทางการ และเลือกใช้เวอร์ชัน 13 เนื่องจากเป็นเวอร์ชันล่าสุด

2. VM ที่ใช้เป็น Multipass Version 1.10.1 ซึ่งโปรแกรม Virtualization ที่ใช้ระบบของ Hyper-V บนวินโดวส์ ซึ่งทำงานคล้ายกับ Oracle VM VirtualBox แล้วจัดการเรื่องอิมเมจและการติดตั้งให้อัตโนมัติ เพื่อ Run ระบบ Linux

ขั้นตอนการติดตั้งโปรแกรมต่าง ๆ เพื่อเตรียมการทดลอง

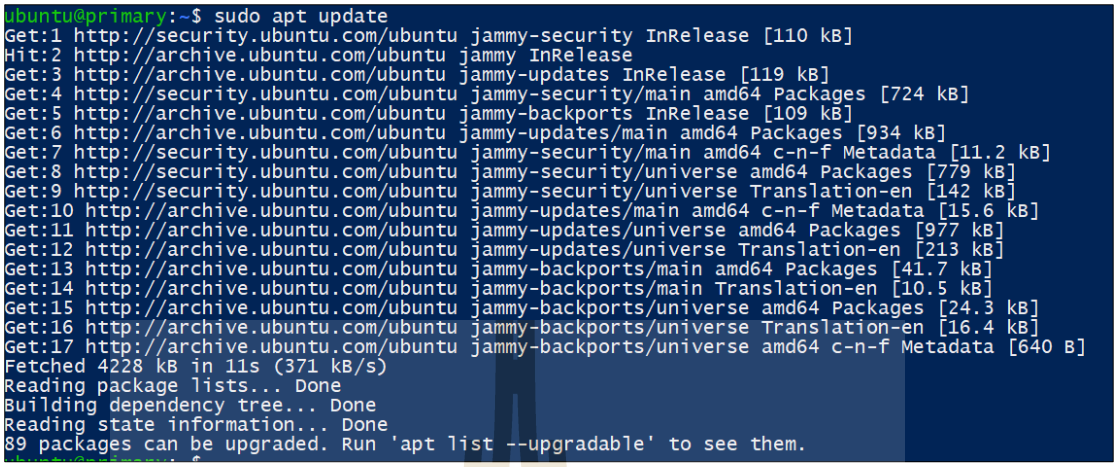
ติดตั้งระบบปฏิบัติการ Linux Ubuntu บน VM Hyper-V ของ Windows 10 ด้วยโปรแกรม multipass โดยสามารถดาวน์โหลดและติดตั้งได้ที่เว็บไซต์ multipass.run และติดตั้ง



รูปที่ 3.2 หน้าเว็บไซต์ multipass

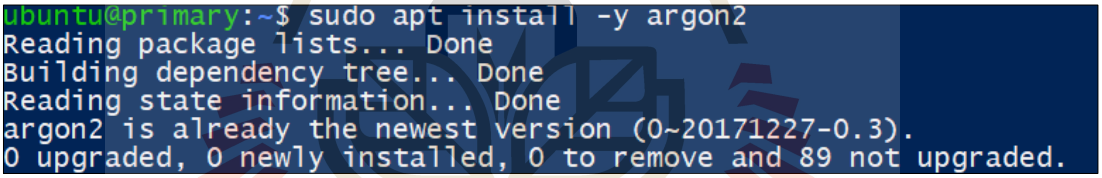
รูปที่ 3.2 แสดงหน้าเว็บเว็บไซต์ multipass.run สำหรับดาวน์โหลดโปรแกรม multipass เพื่อใช้งานระบบ Linux (multipass, 2023)

1. เปิดโปรแกรม multipass และติดตั้ง Argon2 ไปที่ Virtualization โดยใช้ code ดังนี้ใช้คำสั่ง ที่ PowerShell “sudo apt update” เพื่ออัพเดทแพ็คเก็จของระบบ Linux ให้เป็นปัจจุบันเพื่อให้สามารถติดตั้ง Argon2 เวอร์ชั่นล่าสุดได้



รูปที่ 3.3 อัพเดทแพ็คเก็จของระบบ Linux

รูปที่ 3.3 แสดงอัพเดทแพ็คเก็จของระบบ Linux ก่อนการการติดตั้ง Argon2i บน Linux ที่รันอยู่ในโปรแกรม multipass เพื่อตรวจหาอัพเดทล่าสุดและติดตั้งระบบให้เป็น เป็นปัจจุบัน และ “sudo apt install -y argon2” เพื่อการติดตั้ง Argon 2i



รูปที่ 3.4 แสดงการติดตั้ง argon2 บน Linux เพื่อใช้งาน Argon2i

3.4 การกำหนดพารามิเตอร์

ก่อนที่จะทำขั้นตอนในการกำหนดพารามิเตอร์นั้น จะต้องทำความเข้าใจกับคำสั่งของ argon2i ก่อนเพื่อปรับตั้งค่าพารามิเตอร์ รหัสผ่าน , ค่า Salt , Iteration number (t) , Parallelism (p) , Tag length (l) และ Memory size (k) โดยมีคำสั่งสำหรับการใช้งาน Argon2i ที่ผู้ใช้ต้องเข้าใจถึงอักษรย่อพารามิเตอร์ของคำสั่ง Argon2 ก่อนเพื่อใช้งาน

ตารางที่ 3.2 แสดงอักษรพารามิเตอร์ของคำสั่ง Argon2

อักษรย่อ	หมายถึง
-i	การเลือกใช้ Argon2i ในการแฮชชิ่ง
-d	การเลือกใช้ Argon2d ในการแฮชชิ่ง
-id	การเลือกใช้ Argon2id ในการแฮชชิ่ง
-t	การปรับค่า iterations
-m	การปรับค่า memory ที่ใช้หน่วยเป็น MB
-k	การปรับค่า memory ที่ใช้หน่วยเป็น KiB
-p	การปรับค่า parallelism
-l	การปรับค่าความยาวของค่า Output แฮชชิ่ง
-v	การเลือกใช้เวอร์ชั่นในการแฮชชิ่ง

ตารางที่ 3.2 แสดงให้เห็นอักษรย่อต่าง ๆ ที่ใช้เพื่อออกคำสั่งให้ Argon2 นั้นทำงานและกำหนดค่าพารามิเตอร์ต่าง ๆ ตามที่ผู้ใช้ต้องการ

ตัวอย่างการใช้คำสั่ง “echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32” เพื่อสั่งให้ Argon2I แฮชชิ่งรหัสผ่าน

```
ubuntu@primary:~$ echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32
Type:           Argon2i
Iterations:     2
Memory:         4000 KiB
Parallelism:    2
Hash:           ca8206ba004bf87310ce875f0ca69065e7cee726ed18b7ac0f4285a30d627590
Encoded:        $argon2i$v=19$m=4000,t=2,p=2$TzZKVjxIZEQ$yoIGugBL+HMQzodfDKaQZefO5ybtGLesDOKFow1idZA
0.008 seconds
Verification ok
```

รูปที่ 3.5 การรันเพื่อแฮชชิ่งด้วย Argon2i

รูปที่ 3.5 แสดงการรันเพื่อแฮชชิ่งด้วย Argon2i โดยสามารถอธิบายรายละเอียดของคำสั่งได้ดังนี้ ส่วนแรก “echo -n” เป็นคำสั่งของระบบ Linux เพื่อแสดงผลรับจาก Argon2 ส่วนต่อไปคือ “g#7AE9@?” หรือก็คือข้อความรหัสผ่านตั้งต้นที่จะใช้แฮชชิ่ง ส่วนต่อไป “argon2 "O6JV<HdD"” เป็นคำสั่งของ Argon2 ในการใช้ค่า Salt ซึ่งก็คือค่า "O6JV<HdD" ส่วนต่อไปคือ -i นั้นหมายถึง การเลือกใช้ Argon2i ในการแฮชชิ่ง ส่วนต่อไปคือ -k นั้นหมายถึง การปรับค่า memory ที่ใช้หน่วยเป็น KiB และใช้ค่าเท่ากับ 4,000 KiB ส่วนต่อไปคือ -t นั้นหมายถึง การปรับค่า iterations และ ใช้ค่าเป็น 2 ส่วนต่อไปคือ -p นั้นหมายถึง การปรับค่า parallelism และ ใช้ค่าเป็น 2 ส่วนต่อไปคือ -l นั้นหมายถึง การปรับค่าความยาวของค่า Output แฮชชิ่ง และ ใช้ค่าเป็น 32 Byte

ในการกำหนดพารามิเตอร์ ในแต่ละกลุ่มตัวอย่างจะถูกเปลี่ยนแปลงค่าต่าง ๆ ให้เป็นไปตามที่กำหนดไว้ตามตารางดังนี้

ตารางที่ 3.3 แสดงพารามิเตอร์ต่าง ๆ ที่ใช้ในการทดลอง

ค่าพารามิเตอร์	ช่วงการกำหนดค่า	เหตุผล	เพิ่มครั้งละ	เหตุผล
รหัสผ่าน	8-38 ตัวอักษร	เป็นช่วงความยาวการใช้งานรหัสผ่านส่วนใหญ่ อ้างอิงจาก (NT cyfence, 2021)	2 ตัวอักษร	หากเพิ่มครั้งละ 1 ตัวอักษร จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา
ค่า Salt	8 - 38 ตัวอักษร	ใช้จากค่าขั้นต่ำสุดที่ ค่า Salt จะปรับได้ไปจนถึงค่าที่ 38 ซึ่งเป็นค่าที่แนะนำจาก (Awati, 2021) ที่ควรมากกว่า 32 ขึ้นไป	2 ตัวอักษร	หากเพิ่มครั้งละ 1 ตัวอักษร จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา
Iteration number (t) แบบที่ 1	2 - 256 (t)	2t เป็นค่าที่ใกล้ขั้นต่ำสุด (1t) และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 16 เท่าตัว ระหว่าง 2 - 256	6 (t)	หากเพิ่มครั้งละ 1 t จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา
Iteration number (t) แบบที่ 2	2 - 256 (t)	เช่นเดียวกับ แบบที่ 1	1 เท่าตัว	เพื่อให้เห็นความแตกต่างแบบก้าวกระโดด
Parallelism (p) แบบที่ 1	2 - 256 (p)	2p เป็นค่าที่ใกล้ขั้นต่ำสุด (1p) และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 16 เท่าตัว ระหว่าง 2 - 256	6 (p)	หากเพิ่มครั้งละ 1 p จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา

ตารางที่ 3.3 แสดงพารามิเตอร์ต่าง ๆ ที่ใช้ในการทดลอง (ต่อ)

ค่าพารามิเตอร์	ช่วงการกำหนดค่า	เหตุผล	เพิ่มครั้งละ	เหตุผล
Parallelism (p) แบบที่ 2	2 - 256 (p)	เช่นเดียวกับ แบบที่ 1	1 เท่าตัว	เพื่อให้เห็นความแตกต่างแบบก้าวกระโดด
Tag length (l) แบบที่ 1	32 - 256 Byte	32 Byte เป็นค่าเริ่มต้นของ Argon2 และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 4 เท่าตัว ระหว่าง 32 - 256	4 Byte	หากเพิ่มครั้งละ 1 Byte จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา
Tag length (l) แบบที่ 2	32 - 4096 Byte	ใช้จากค่าเริ่มต้นของ Argon2 จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 8 เท่าตัว ระหว่าง 32 - 4096	1 เท่าตัว	เพื่อให้เห็นความแตกต่างแบบก้าวกระโดด
Memory size (k) แบบที่ 1	4,000 – 400,000 KiB	ใช้จากค่าที่ใกล้กับค่าเริ่มต้นของ Argon2 (4096 KiB) จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 8 เท่าตัว ระหว่าง 4000 - 400000	4,000 KiB	หากเพิ่มครั้งละ 1 KiB จะทำให้ตัวอย่างทดลองน้อยจนไม่ทำให้เห็นความแตกต่างด้านเวลา
Memory size (k) แบบที่ 2	4,000 – 4,096,000 KiB	ใช้จากค่าที่ใกล้กับค่าเริ่มต้นของ Argon2 (4096 KiB) จะปรับไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างก้าวกระโดดได้ 11 เท่าตัว ระหว่าง 4000 – 4,096,000	1 เท่าตัว	เพื่อให้เห็นความแตกต่างแบบก้าวกระโดดและเพิ่มขอบเขตถึงจุดสูงสุดเท่าที่จะทำได้

ตารางที่ 3.3 นี้แสดงถึง พารามิเตอร์ต่าง ๆ ที่ใช้ในการทดลอง โดยสามารถอธิบายได้ ดังนี้ อันดับแรก คือ รหัสผ่าน โดยการทดลองนี้จะใช้รหัสผ่านที่สุ่มขึ้นมาทั้งหมดมีความยาวตั้งแต่ 8-32 ตัวอักษรเพิ่มขึ้นครั้งละ 2 ตัวอักษร เหตุผลที่เลือกเพราะเป็นช่วงความยาวการใช้งานรหัสผ่านส่วนใหญ่อ้างอิงจากบทความของรหัสผ่านที่ถูกละเมิดออกมาของ (Awati, 2021) ที่เคยกล่าวไว้แล้ว

อันดับที่สอง คือ ค่า Salt ที่สุ่มขึ้นมาทั้งหมดกำหนดตั้งแต่ 8 - 32 ตัวอักษร โดยเพิ่มครั้งละ 2 ตัว เหตุผลที่เลือกเพราะผู้วิจัยต้องการใช้จากค่าขั้นต่ำสุดที่ ค่า Salt จะปรับได้ไปจนถึงค่าที่ 32 ซึ่งเป็นค่าที่แนะนำจาก (Awati, 2021) ในการกำหนดค่า Salt

อันดับที่สาม คือ พารามิเตอร์ Iteration number (t) จะมี 2 แบบ ซึ่งแบบที่ 1 จะกำหนดตั้งแต่ 2 - 256 โดยเพิ่มครั้งละ 6 ส่วนแบบที่ 2 กำหนดตั้งแต่ 2 - 256 โดยเพิ่มขึ้นครั้งละเท่าตัว เหตุผลที่เลือกเริ่มต้นที่ 2t เพราะเป็นค่าที่ใกล้ขั้นต่ำสุด(1t) และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้นั้นคือ 16 เท่าตัว โดยแบบที่ 1 จะเพิ่มขึ้นครั้งละ 6P ไปเรื่อย ๆ และ แบบที่ 2 จะ เพิ่มขึ้นแบบเท่าตัว

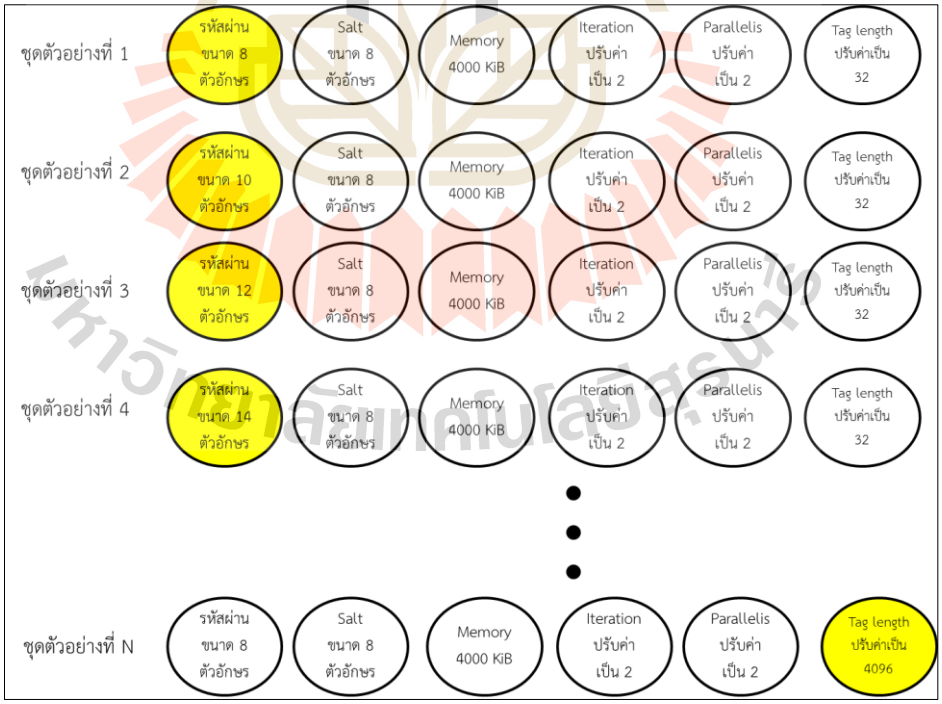
อันดับที่สี่ พารามิเตอร์ Parallelism (p) จะมี 2 แบบเช่นกัน ซึ่งแบบที่ 1 จะกำหนดตั้งแต่ 2 - 256 โดยเพิ่มครั้งละ 6 และ แบบที่ 2 กำหนดตั้งแต่ 2 - 256 ตัวอักษร โดยเพิ่มขึ้นเป็น 1 เท่าตัวเหตุผลที่เลือกเริ่มต้นที่ 2p เพราะเป็นค่าที่ใกล้ขั้นต่ำสุด(1p) และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 16 เท่าตัว

อันดับที่ห้า พารามิเตอร์ Tag length (l) จะมี 2 แบบเช่นกัน ซึ่งแบบที่ 1 กำหนดตั้งแต่ 32 - 256 โดยเพิ่มครั้งละ 4 Byte และแบบที่ 2 กำหนดตั้งแต่ 32 - 4096 Byte โดยเพิ่มขึ้นเป็นเท่าตัว และใช้ค่าเริ่มต้นเป็น 32 Byte เพราะเป็นค่าเริ่มต้นของ Argon2 และง่ายต่อการเพิ่มขึ้นแบบเท่าตัว จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 4 เท่าตัว ในแบบที่ 1 และ 8 เท่าตัว ในแบบที่ 2 เหตุผลที่ค่าสุดท้ายของแบบที่ 2 ต่างกันกับค่าสุดท้ายของแบบที่ 1 เพราะหลังจากที่ผู้วิจัยได้ทดลองแล้วพบว่าการเพิ่มค่า Tag length ไม่มีผลต่อเวลาจึงต้องการเพิ่มค่าสูงสุดเพิ่มให้เห็นชัดเจนขึ้น และ เหตุผลที่ไม่ใช้ค่าสูงสุดเป็น 16 เท่า เหมือนกับ Iteration และ Parallelism เพราะหากใช้ค่า Tag length ที่มีค่าขนาดมากกว่า 4096 Byte ขึ้นไปจะเป็นขนาดที่อาจจะเยอะเกินไปสำหรับการเก็บในข้อมูลในฐานข้อมูล เมื่อเทียบกับอัลกอริทึม อื่น ๆ ที่มี Output สูงสุด SHA512 มีขนาดเพียง 62 Byte เท่านั้น

อันดับที่หก พารามิเตอร์ Memory size (k) จะมี 2 แบบเช่นกัน ซึ่งแบบที่ 1 กำหนดตั้งแต่ 4000 – 400,000 KiB โดยเพิ่มครั้งละ 4000 KiB และแบบที่ 2 กำหนดตั้งแต่ 4000 – 4,096,000 KiB โดยเพิ่มขึ้นเป็นเท่าตัว เหตุผลที่เลือกใช้จากค่าเริ่มต้นเป็น 4000 KiB เพราะเป็นค่าที่ใกล้กับค่าเริ่มต้นของ Argon2 (4096 KiB) จะปรับได้ไปจนถึงค่าที่ผู้วิจัยคิดว่าจะมีผลให้เห็นค่าความแตกต่างอย่างละเอียดได้ 8 เท่าตัว สำหรับแบบที่ 1 และ 11 เท่าตัว สำหรับแบบที่สอง เหตุผลที่ไม่ใช้ค่าสูงสุดเป็น

16 เท่า เหมือนกับ Iteration และ Parallelism เพราะหากใช้ค่า ที่มีค่าขนาด เป็น 16 เท่า เพราะค่า Memory size จะสูงขึ้นมาก ๆ และจะเป็นการใช้กลุ่มตัวอย่างที่เยอะจนเกินไป และ เหตุผลที่ค่าสุดท้ายของแบบที่ 2 ต่างกันกับค่าสุดท้ายของแบบที่ 1 เพราะหลังจากที่ผู้วิจัยได้ทดลองแล้วพบว่าการเพิ่มค่า Memory size มีผลต่อเวลาชัดเจนที่สุดจึงต้องการเพิ่มค่าสูงสุดเพิ่มหาค่าเวลาที่นานที่สุดเท่าที่ฮาร์ดแวร์ที่ทำการทดลองนี้จะรับไหวคือ 11 เท่าของขนาดเริ่มต้น

การทดลองนี้จะดำเนินการ เปลี่ยนพารามิเตอร์ทีละชนิดพารามิเตอร์ตามที่กำหนดไว้ในตารางที่ 3.2 และทำการ Fix พารามิเตอร์อื่น ๆ ทั้งหมด และเลือกที่จะ ให้ค่าที่เป็นค่าที่ต่ำที่สุดของช่วงตามที่กำหนดไว้เพื่อให้สามารถควบคุมเวลาที่ใช้ในการทดลองได้ เพราะตัวอย่างการทดลองมีจำนวนที่เยอะกว่า 1,000 ตัวอย่างโดยประมาณ ดังนั้นการใช้ค่าที่ต่ำที่สุดจึงจำเป็นในการทดลองเพื่อให้การทดลองเป็นไปได้ด้วยความรวดเร็ว เพื่อค้นหาว่าค่าพารามิเตอร์ที่กำหนดไว้ตัวไหนนั้นมีผลต่อเวลามากที่สุด ยกตัวอย่างเช่น การทดลองหาพารามิเตอร์รหัสผ่าน จะทดลองเปลี่ยนค่า พารามิเตอร์ รหัสผ่านเพียงอย่างเดียว โดยที่ให้ค่าพารามิเตอร์อื่น ๆ คงที่ไว้ทั้งหมด หลังจากทดลองทุก Setting แล้วก็จะทำการทดลอง พารามิเตอร์ Salt โดยที่เปลี่ยนค่าพารามิเตอร์ตามที่กำหนดไว้ในตารางที่ 3.2 และ ให้ค่าพารามิเตอร์ รหัสผ่าน และตัวอื่น ๆ ไว้ ทั้งหมด สามารถอธิบายได้ดังรูปที่ 3.6



รูปที่ 3.6 แสดงตัวอย่างในการกำหนดพารามิเตอร์ ในแต่ละกลุ่มตัวอย่าง

รูปที่ 3.6 แสดงตัวอย่างในการกำหนดพารามิเตอร์ต่าง ๆ ในแต่ละกลุ่มตัวอย่าง โดยผู้วิจัยจะปรับพารามิเตอร์ต่าง ๆ ตามตารางที่ 3.2 ที่กำหนดไว้ โดยกลุ่มตัวอย่างที่ 1 จะเปลี่ยนพารามิเตอร์รหัสผ่านเป็นขนาด 8 ตัวอักษร และ ทำการ Fix ค่าพารามิเตอร์อื่น ๆ ทั้งหมดเพื่อทดลองหาค่ารหัสผ่านที่มีผลต่อเวลา หลังจากที่ทดลองรหัสผ่านเสร็จสิ้นแล้วก็จะทำการทดลองค่า Salt และตามด้วย Memory size

โดยจะทำการทดลองทั้งหมดนี้จะทำซ้ำเป็นจำนวน 5 ครั้งในแต่ละพารามิเตอร์ และคำนวณค่าเฉลี่ย ได้แก่ กรณีของพารามิเตอร์รหัสผ่าน จะใช้ช่วงความยาวรหัสผ่านตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 8 – 32 ตัวอักษร และทำการทดลองสุ่มรหัสผ่านโดยทำการทดลองซ้ำจนครบ 5 ครั้ง และในแต่ละครั้งรหัสผ่านจะไม่ซ้ำกัน

กรณีของพารามิเตอร์ Salt จะใช้ช่วงความยาวของค่า Salt ตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 8 - 32 ตัวอักษร และทำการทดลองสุ่ม Salt โดยทำการทดลองซ้ำจนครบ 5 ครั้ง และในแต่ละครั้งค่า Salt จะไม่ซ้ำกัน

กรณีของพารามิเตอร์ Iteration จะใช้ช่วงความยาวของค่า Iteration ตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 2 - 256 (t) ทั้งวิธีที่เพิ่มครั้งละ 6 และ วิธีที่เพิ่มขึ้นเท่าตัว และทำการทดลองสุ่มรหัสผ่านโดยทำการทดลองซ้ำจนครบ 5 ครั้งต่อ ซึ่งหมายความว่า วิธีที่เพิ่มครั้งละ 6 จะทดลองซ้ำจนครบ 5 ครั้ง และ วิธีที่เพิ่มขึ้นเท่าตัวจะทดลองซ้ำจนครบ 5 ครั้ง

กรณีของพารามิเตอร์ Parallelism จะใช้ช่วงความยาวของค่า Parallelism ตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 2 - 256 (p) ทั้งวิธีที่เพิ่มครั้งละ 6 และ วิธีที่เพิ่มขึ้นเท่าตัว และทำการทดลองสุ่มรหัสผ่านโดยทำการทดลองซ้ำจนครบ 5 เช่นเดียวกับ กรณีของพารามิเตอร์ Iteration

กรณีของพารามิเตอร์ Tag length จะใช้ช่วงความยาวของค่า Tag length ตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 32 - 256 Byte ของวิธีที่เพิ่มครั้งละ 4 และ 32 - 4096 Byte ของ วิธีที่เพิ่มขึ้นเท่าตัว และทำการทดลองสุ่มรหัสผ่านโดยทำการทดลองซ้ำจนครบ 5 เช่นเดียวกับ กรณีของพารามิเตอร์ Iteration

กรณีของพารามิเตอร์ Memory size จะใช้ช่วงความยาวของค่า Memory size ตามตารางที่กำหนดค่าไว้ข้างต้น นั้นคือ 4,000 – 400,000 KiB ของวิธีที่เพิ่มครั้งละ 4,000 – 4,096,000 KiB ของวิธีที่เพิ่มขึ้นเท่าตัว และทำการทดลองสุ่มรหัสผ่านโดยทำการทดลองซ้ำจนครบ 5 เช่นเดียวกับ กรณีของพารามิเตอร์ Iteration

3.5 การประเมินผล

งานวิจัยนี้มีขอบเขต ทั้งในการค้นหาค่าพารามิเตอร์ที่มีผลกระทบต่อเวลาและการประเมินด้านความปลอดภัยในการประมวลผลของ Argon2I สำหรับแฮชรหัสผ่าน โดยมีรายละเอียดดังนี้

3.5.1 การประเมินค่าพารามิเตอร์ที่มีผลกระทบต่อเวลา

โดยการประเมินนั้น สามารถประเมินได้จากการ Output ของโปรแกรม Argon2 ดังรูปที่ 3.3

รูปที่ 3.7 แสดงตัวอย่างค่าเวลาการคำนวณที่ Argon2 แสดงผลออกมา

รูปที่ 3.7 แสดงภาพตัวอย่างของเอาต์พุตของ Argon 2i หลังจากที่ได้ใส่พารามิเตอร์ตัวอย่างต่าง ๆ ได้แก่ข้อมูล Plain-text ที่ระบุให้มีค่า Hello, ข้อมูล Salt ที่ได้ใส่ค่า mysalt0123456789, ข้อมูลชนิดแฮชชิ่ง ใส่ค่า I ซึ่งก็คือ argon2i, ข้อมูล Memory size ใส่ค่า 65,536(KiB), ข้อมูล Iteration number ใส่ค่า 4, ข้อมูล Parallelism ใส่ค่า 2 และ โปรแกรมจะแสดงข้อมูลต่าง ๆ ออกมา รวมถึงเวลาในการทำงาน โดยตัวอย่างในรูปที่ 3.3 จะเห็นได้ว่า เวลาที่ใช้ในการประมวลผล หรือคำนวณค่าแฮชชิ่งนั้น คือ 0.241 วินาที และหลังจากได้เวลาที่ทดลองแล้วจึงทำการบันทึกและรวบรวมเพื่อหาค่าเฉลี่ยตามกระบวนการที่วางแผนไว้

การประเมินผลจะทำได้โดยการนำผลลัพธ์ของพารามิเตอร์ที่ทดลองในรูปแบบต่าง ๆ ที่กำหนดไว้ข้างต้น เพื่อหาเวลาแล้วได้ผลลัพธ์ของเวลาออกมาได้น้อยที่สุด หลังจากนั้นจะหาว่าค่าพารามิเตอร์ที่กำหนดไว้ค่าไหนหรือขนาดเท่าไหร่ที่ใช้เวลาน้อยที่สุด และนำค่าพารามิเตอร์เหล่านั้นมารวมกันและทำการทดลองในขั้นสุดท้ายเพื่อเป็นการยืนยันผลต่อไป โดยเป้าหมายของการวิจัยนี้ คือ การปรับพารามิเตอร์แบบใดบ้างที่มีผลต่อความเร็วกับการแฮชชิ่งด้วย Argon2i

3.5.2 การประเมินด้านความปลอดภัย

หลังจากที่ได้ค่าพารามิเตอร์ที่ใช้เวลาในแฮชชิ่ง Argon2i ที่น้อยที่สุดแล้วจะนำพารามิเตอร์เหล่านั้นมาปรับเปลี่ยนเพื่อมาหาค่า Avalanche effect เพื่อวัดความแข็งแกร่งของกระบวนการแฮชชิ่ง โดยจะประเมินด้วย 2 วิธีคือ

เปลี่ยนรายพารามิเตอร์แล้วทำการคำนวณ Avalanche effect ดังสามารถอธิบายได้ตามตารางดังนี้

ตารางที่ 3.4 การปรับเปลี่ยนค่าเพื่อทดลอง Avalanche effect

พารามิเตอร์	Password	Salt	Memory size	Iteration number	Parallelism (p)
รหัสตั้งต้น	r=IlH<7+ej[fv7u,A&Ob%H3x7J.A	gGvRivJNky7)gsu:Bhx8j7R.<G	4000	2	8
รหัสที่เปลี่ยนครั้งที่ 1	r=IlH<7+ej[fv7u,A&Ob%H3x7J.@	gGvRivJNky7)gsu:Bhx8j7R.<F	3999	3	9
รหัสที่เปลี่ยนครั้งที่ 2	r=IlH<7+ej[fv7u,A&Ob%H3x7J.C	gGvRivJNky7)gsu:Bhx8j7R.<A	3998	4	10
รหัสที่เปลี่ยนครั้งที่ 3	r=IlH<7+ej[fv7u,A&Ob%H3x7J.E	gGvRivJNky7)gsu:Bhx8j7R.<I	3997	5	11
รหัสที่เปลี่ยนครั้งที่ 4	r=IlH<7+ej[fv7u,A&Ob%H3x7J.I	gGvRivJNky7)gsu:Bhx8j7R.<Q	3996	6	12
รหัสที่เปลี่ยนครั้งที่ 5	r=IlH<7+ej[fv7u,A&Ob%H3x7J.Q	gGvRivJNky7)gsu:Bhx8j7R.<a	3995	7	13

ตารางที่ 3.4 แสดงการปรับเปลี่ยนค่าเพื่อทดลอง Avalanche effect ได้แก่ พารามิเตอร์ Password Salt Memory size Iteration number และ Parallelism (p) ซึ่งในการปรับเปลี่ยนค่าพารามิเตอร์สำหรับการประเมิน Avalanche Effect จะดำเนินการตามที่ได้กำหนดไว้ใน ตารางที่ 3.4

เปลี่ยนทุก ๆ พารามิเตอร์ ที่ใช้วิธีเหมือนกับกับวิธีที่ 1 ทั้งหมดแต่จะเป็นการเปลี่ยนพร้อมกันทั้งหมด แล้วทำการคำนวณ Avalanche effect เพื่อเป็นการประเมินความปลอดภัยของการแฮชชิ่งด้วย Argon2i กับพารามิเตอร์ที่ใช้เวลาที่น้อยที่สุด ซึ่งทำได้โดยการเปลี่ยนค่าพารามิเตอร์ไป 1 บิต และแฮชชิ่งอีกครั้งแล้วนำผลลัพธ์ค่าแฮชชิ่งของทั้งสองค่ามาเปรียบเทียบกันในแต่ละ 1 บิตว่ามีความแตกต่างกันเป็นกี่เปอร์เซ็น

โดยการเปลี่ยนนั้นจะเปลี่ยนค่าเพื่อทดลอง Avalanche effect จะทำการทดลองแฮชชิ่งอัลกอริทึมต่าง ๆ ได้แก่ Argon2i , MD5 , SHA1 และ SHA256 และทดสอบซ้ำ 5 ครั้ง โดยจะใช้รหัสผ่านเดียวกันกับ การทดลอง Argon2i

และหลังจากที่ได้แฮชชิ่ง output ออกมาแล้วนั้นก็ทำการเปรียบด้วยโดยการ นำค่าแฮชชิ่งไปแปลงเป็นเลขฐาน 2 หมายถึง การใช้เลขฐาน 2 เพื่อแสดงอักขระต่าง ๆ อักขระแต่ละตัวจะแสดงด้วยเลขฐาน 2 จำนวน 8 บิต (1 ไบต์) โดยแต่ละบิตแทน 0 หรือ 1 ขึ้นอยู่กับตำแหน่งภายในลำดับไบต์ ASCII มาตรฐานประกอบด้วย 7 บิตที่ฝังอยู่ภายในไบต์ขนาด 8 บิต โดยบิตสุดท้ายสงวนไว้เป็นบิต "parity" เพื่อตรวจสอบข้อผิดพลาดในการส่ง ซึ่งอนุญาตให้มีอักขระที่แตกต่างกันทั้งหมด 128 ตัว (Hopkins, 2023) และจึงนำมาเปรียบเทียบกันแล้วหาเปอร์เซ็นต์ความต่างกันของค่าแฮช เพื่อเป็นการประเมินความปลอดภัยของ Argon2i กับ แฮชชิ่งอัลกอริทึมที่ใช้กันอยู่ในปัจจุบัน สามารถแสดงขนาดตัวอักษรของ แฮชชิ่ง output ของ อัลกอริทึมต่าง ๆ และ ขนาดของ แฮชชิ่ง output เมื่อแปลงเป็น ASCII ได้ดังตารางนี้

ตารางที่ 3.5 แสดงขนาดของตัวอักษรแฮชชิ่ง output และ เลขฐานสอง

แฮชชิ่งอัลกอริทึม	แฮชชิ่ง output / ตัวอักษร	เลขฐานสอง / ตัวอักษร
SHA1	40	320
MD5	32	256
Argon2i	64	512
SHA256	64	512

ตารางที่ 3.5 แสดงขนาดของตัวอักษรแฮชชิ่ง output แบบข้อมูลฐานปกติ และ ข้อมูลในฐาน 2 ของอัลกอริทึมต่าง ๆ โดย SHA1 มีขนาดของแฮชชิ่ง output และ เลขฐานสอง เท่ากับ 40 และ 320 ตัวอักษร และ MD5 เท่ากับ 32 และ 320 ตัวอักษรตามลำดับ และ Argon2i และ SHA256 มีขนาดของแฮชชิ่ง output และ เลขฐานสอง เท่ากับ 128 และ 512 ตัวอักษร ตามลำดับ และมีขนาดที่เท่ากัน

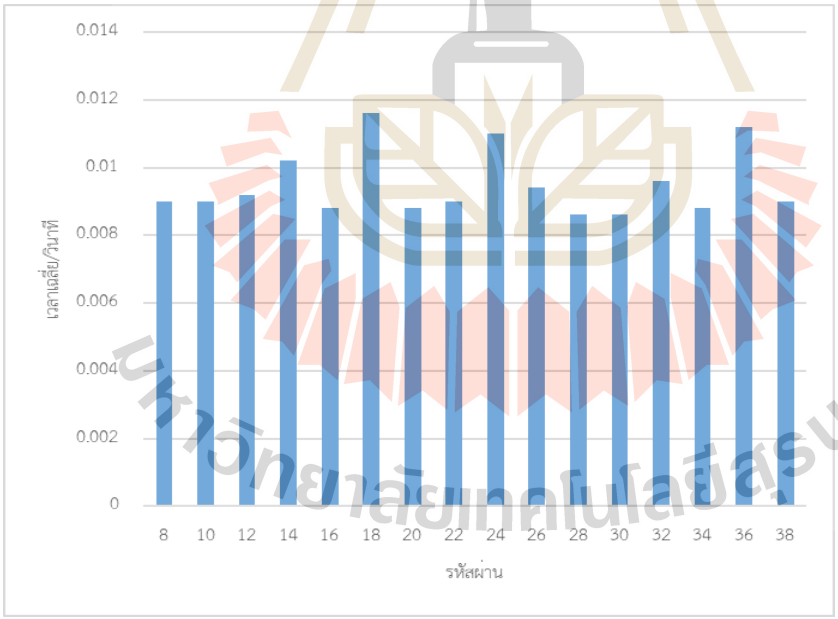
บทที่ 4
ผลการวิจัยและการอภิปรายผล

บทนี้จะนำเสนอผลการวิจัย ผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i ซึ่งจะประกอบไปด้วยผลการวิจัยของพารามิเตอร์ต่อความเร็วและการประเมินด้านความปลอดภัย และมีรายละเอียดดังนี้

4.1 ผลการทดลองในด้านของเวลาของพารามิเตอร์

การทดลองพารามิเตอร์จะประกอบไปด้วยผลของ รหัสผ่าน , ค่า Salt , Memory size (k) , Parallelism (p) , Tag length (l) และ Iteration number (t) โดยจะทดลองในด้านของเวลา ดังนี้

4.1.1 ผลการทดลองของพารามิเตอร์รหัสผ่าน



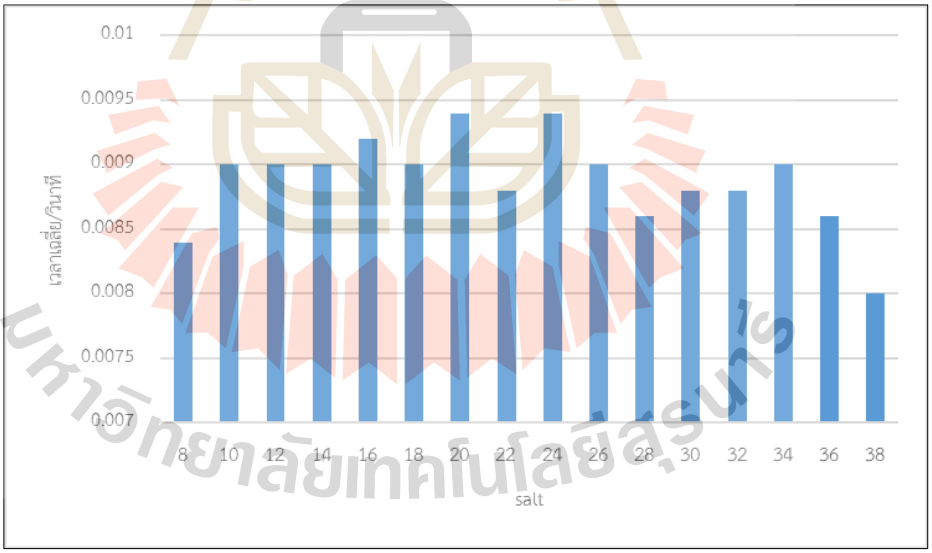
รูปที่ 4.1 เวลาที่ใช้ในการประมวลผลสำหรับรหัสผ่านขนาดต่าง ๆ

รูปที่ 4.1 แสดงผลการทดลองของพารามิเตอร์รหัสผ่านที่ถูกเปลี่ยนตั้งแต่ 8-38 ตัวอักษร จะเห็นได้ว่า ความยาว 8 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 10 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 12 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0092 วินาที และความยาว 14 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0102 วินาที และความยาว 16 ตัวอักษรได้เวลาเฉลี่ย

เท่ากับ 0.0088 วินาที และความยาว 18 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0116 วินาที และความยาว 20 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0088 วินาที และความยาว 22 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 24 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.011 วินาที และความยาว 26 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0094 วินาที และความยาว 28 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0086 วินาที และความยาว 30 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0086 วินาที และความยาว 32 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0096 วินาที และความยาว 34 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0088 วินาที และความยาว 36 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.0112 วินาที และ ความยาว 38 ตัวอักษรได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที ค่าที่ได้เวลาน้อยที่สุดคือ 28 ตัวอักษร และ มากสุดคือ 18 ตัวอักษร

ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของตัวอักษรรหัสผ่านไม่มีผลต่อเวลาการแฮช ซึ่ง เหตุว่าค่าที่แตกต่างกันของเวลาระหว่างค่าที่นานที่สุดและเร็วที่สุดคือ 0.003 วินาทีเท่านั้น และแนวโน้มของกราฟเป็นไปในทางคงที่ เนื่องจาก Argon2i นั้นมีคุณสมบัติพิเศษคือการคำนวณไม่ขึ้นอยู่กับ Input หรือ รหัสผ่านที่เข้ามาจึงไม่มีผลต่อเวลา

4.1.2 ผลการทดลองของพารามิเตอร์ Salt



รูปที่ 4.2 เวลาที่ใช้ในการประมวลผลสำหรับ Salt ขนาดต่าง ๆ

รูปที่ 4.2 แสดงผลการทดลองของพารามิเตอร์ Salt ที่ถูกเปลี่ยนตั้งแต่ 8-38 ตัวอักษรจะเห็นได้ว่า ความยาว 8 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0084 วินาที และความยาว 10 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 12 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009

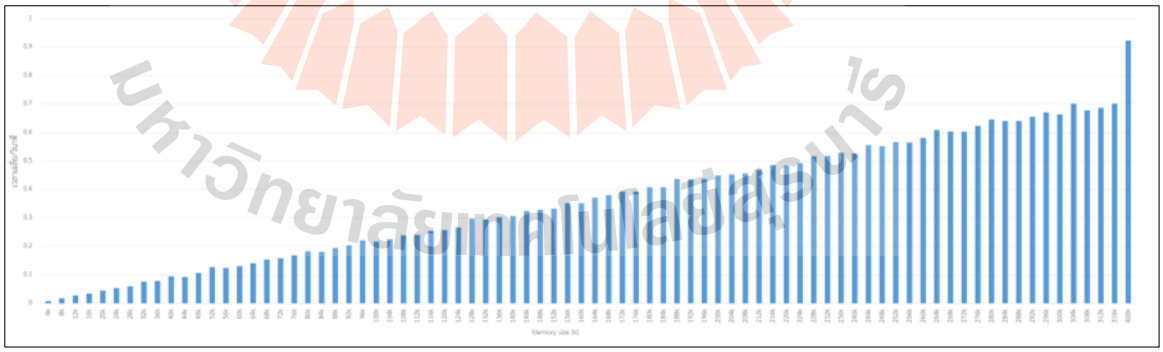
วินาที และความยาว 14 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 16 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0092 วินาที และความยาว 18 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 20 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0094 วินาที และความยาว 22 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0088 วินาที และความยาว 24 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0094 วินาที และความยาว 26 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 28 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0086 วินาที และความยาว 30 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0088 วินาที และความยาว 32 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0088 วินาที และความยาว 34 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.009 วินาที และความยาว 36 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.0086 วินาที และความยาว 38 ตัวอักษร ได้เวลาเฉลี่ยเท่ากับ 0.008 วินาที ค่าที่ได้เวลาน้อยที่สุดคือ 38 ตัวอักษร และ มากสุดคือ 24 ตัวอักษร

ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของตัวอักษร Salt ไม่มีผลต่อเวลาการแฮชชิ่ง เหตุว่าค่าที่แตกต่างกันของเวลาระหว่างค่าที่นานที่สุดและเร็วที่สุดคือ 0.0014 วินาที เท่านั้น และแนวโน้มของกราฟเป็นไปในทางคงที่

4.1.3 ผลการทดลองของพารามิเตอร์ Memory size

โดยจะแบ่งเป็น 2 ประเภทในการเพิ่มขึ้นของขนาด Memory size ได้แก่

4.1.3.1 เป็นการเพิ่มค่าพารามิเตอร์ Memory size ตั้งแต่ 4000 KiB ถึง 400,000 KiB เพิ่มครั้งละ 4,000 KiB

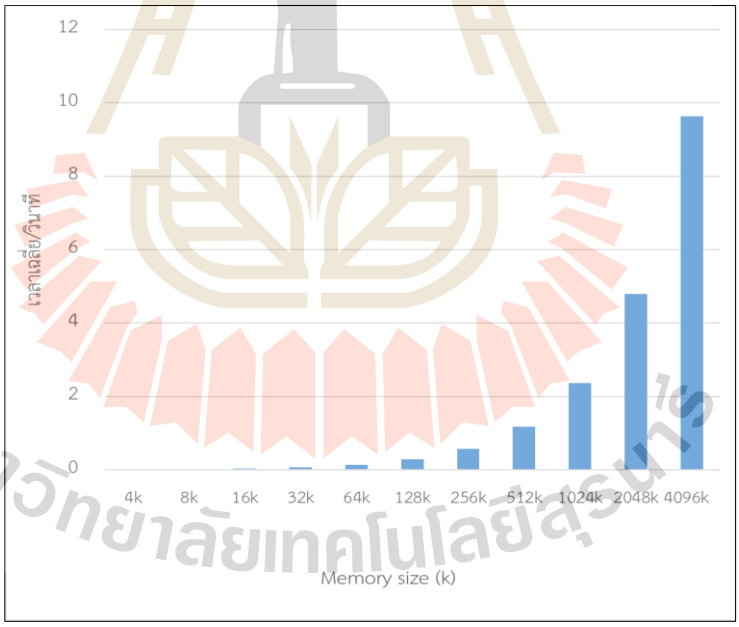


รูปที่ 4.3 เวลาที่ใช้ในการประมวลผลสำหรับ Memory size เมื่อเพิ่มขนาดครั้งละ 4,000 KiB

รูปที่ 4.3 แสดงผลการทดลองของพารามิเตอร์ Memory size ที่ถูกเปลี่ยนตั้งแต่ 4000 KiB - 400,000 KiB โดยแนวตั้งคือระยะเวลา / วินาที และแนวนอนคือค่า Memory size ที่มีหน่วยเป็น -k ถูกเพิ่มขึ้นครั้งละ 4000 KiB จะเห็นได้ค่าของเวลานั้นสูงขึ้นอย่างชัดเจน โดยค่าที่ได้เวลาน้อยที่สุดคือ 4,000 KiB และ มากสุดคือ 400,000 KiB และ ค่า 400,000 KiB จะทำให้มีค่าดีดขึ้นของกราฟมากที่สุดจนเป็นที่สังเกตุ ซึ่งสาเหตุอาจเป็นช่วงขนาด 400,000 KiB อาจมีจุดสำคัญอะไรบางอย่างกับ Aagon2i

ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Memory size ที่เพิ่มขึ้นมีผลชัดเจนต่อเวลาการแฮชชิ่งอย่างชัดเจน ค่าเพิ่มขึ้นของเวลาต่อรอบเฉลี่ยแล้วเพิ่มขึ้นถึง 0.0116 วินาทีต่อรอบการเพิ่มขึ้นครั้งละ 4,000 KiB แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาในทุก ๆ การเพิ่มขนาดพารามิเตอร์นี้ และ จากกราฟจะเห็นได้ว่ามีแนวโน้มที่จะสูงขึ้นตามขนาดของ Memory size อย่างชัดเจน

4.1.3.2 ผลการทดลองของพารามิเตอร์ Memory size (k) ตั้งแต่ 4000 KiB ถึง 4,096,000 KiB โดยเพิ่มขึ้นเป็นตัว



รูปที่ 4.4 เวลาที่ใช้ในการประมวลผลสำหรับ Memory size เมื่อถูกเพิ่มขึ้นเท่าตัวต่อครั้ง

รูปที่ 4.4 แสดงผลการทดลองของพารามิเตอร์ Memory size ที่ถูกเปลี่ยนตั้งแต่ 4000 KiB – 4,096,000 KiB โดยถูกเพิ่มขึ้นเท่าตัว ต่อครั้ง

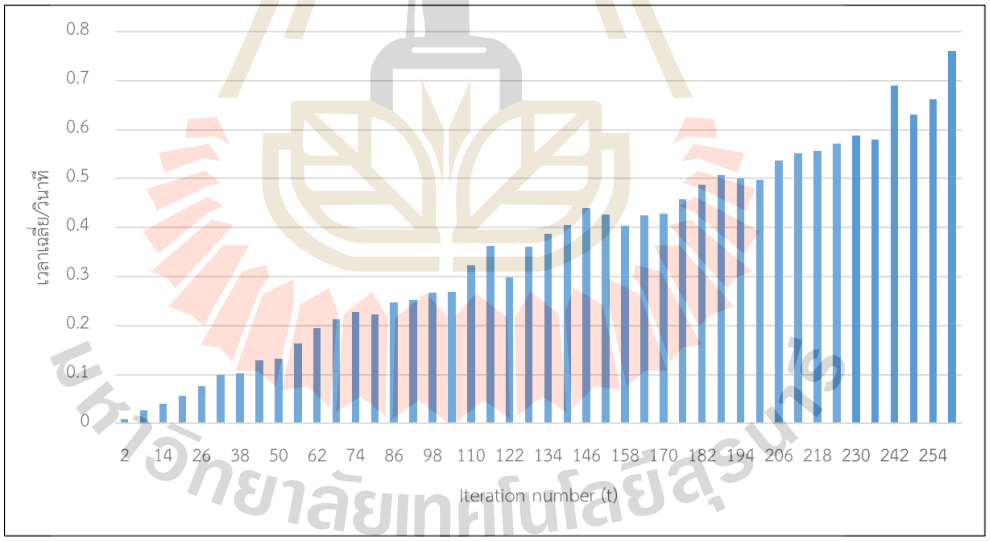
ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Memory size ที่เพิ่มขึ้นมีผลชัดเจนต่อเวลาการแฮชชิ่งอย่างชัดเจน การเพิ่มขึ้นของเวลาต่อรอบเฉลี่ยแล้วเพิ่มขึ้นถึง 0.531 วินาที ต่อรอบการเพิ่มขึ้นเท่าตัว แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาในทุก ๆ การเพิ่มขนาดพารามิเตอร์นี้ และ จากกราฟจะเห็นได้ว่ามีแนวโน้มที่จะสูงขึ้นตามขนาดของ Memory size อย่างชัดเจน และในกราฟนี้จะทำให้เห็นภาพการเพิ่มขึ้นที่ชัดกว่าในแบบที่ 1 เหตุผลเนื่องจากช่วงของค่าที่ทำการทดลองที่มีขนาดใหญ่กว่าแบบที่ 1

ทั้งหมดนี้ทำให้เห็นว่าการเพิ่มขึ้นของค่า Memory size จะทำให้เวลาในการแฮชนั้นเพิ่มขึ้นอย่างชัดเจน เพราะหากเพิ่มขนาดของค่า Memory size ให้มากขึ้น จะเป็นการให้ตัว Argon2i นั้นเรียกใช้งาน RAM ของคอมพิวเตอร์เพื่อใช้ในการแฮชที่มากขึ้น ดังนั้นจึงเป็นการเพิ่มภาระให้กับคอมพิวเตอร์ในการคำนวณค่าแฮช

4.1.4 หาพารามิเตอร์ Iteration number (t)

โดยจะแบ่งเป็น 2 ประเภทในการเพิ่มขึ้นของขนาด Iteration number (t) ได้แก่

4.1.4.1 การเพิ่มค่าพารามิเตอร์ Iteration number (t) ตั้งแต่ 2 ถึง 242 โดยเพิ่มครั้งละ 6

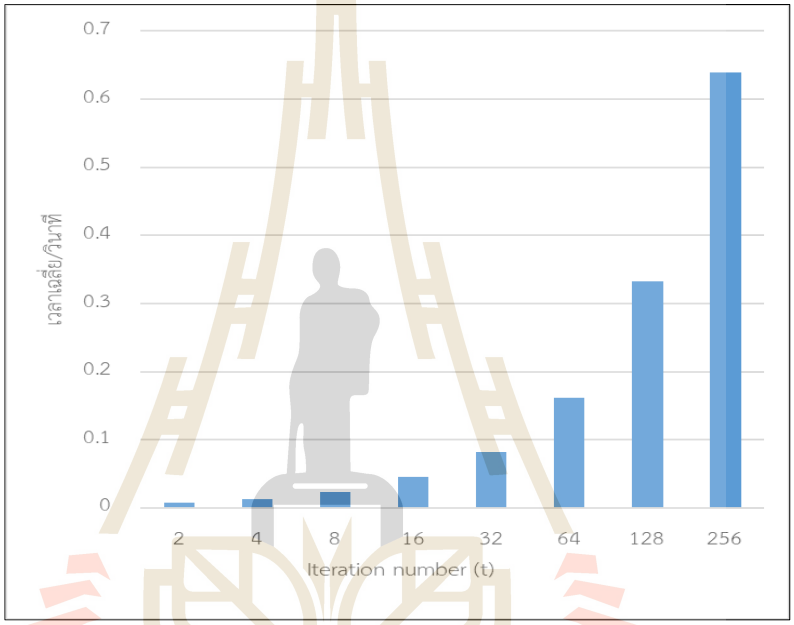


รูปที่ 4.5 เวลาที่ใช้ในการประมวลผลสำหรับ Iteration number เมื่อเพิ่มขนาดครั้งละ 6 t

รูปที่ 4.5 แสดงผลการทดลองของพารามิเตอร์ Iteration number (t) ที่ถูกเปลี่ยนตั้งแต่ 2 - 254 t โดยแนวตั้งคือระยะเวลา / วินาที และแนวนอนคือค่า Iteration number ที่มีหน่วยเป็น -t ถูกเพิ่มขึ้นครั้งละ 6 t ข้อสังเกตคือค่าจะมีการดีดขึ้นเล็กน้อยที่ ค่า t เท่ากับ 122 ,146 และ 242 และค่าเวลาที่น้อยที่สุดคือ t 2 และ มากที่สุดคือ t 242

ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Iteration number มีผลชัดเจนต่อเวลาการแฮชชิ่ง การเพิ่มขึ้นของเวลาต่อรอบเฉลี่ยแล้วเพิ่มขึ้นถึง 0.0152 วินาทีต่อรอบการเพิ่มขึ้นครั้งละ 6 t แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาในทุก ๆ การเพิ่มขนาดพารามิเตอร์นี้ และ จากกราฟจะเห็นได้ว่ามีแนวโน้มที่จะสูงขึ้นตามขนาดของ Iteration number อย่างชัดเจน

4.1.4.2 การเพิ่มค่าพารามิเตอร์ Iteration number (t) ตั้งแต่ 2 ถึง 256 ถูกเพิ่มขึ้นเท่าตัวต่อครั้ง



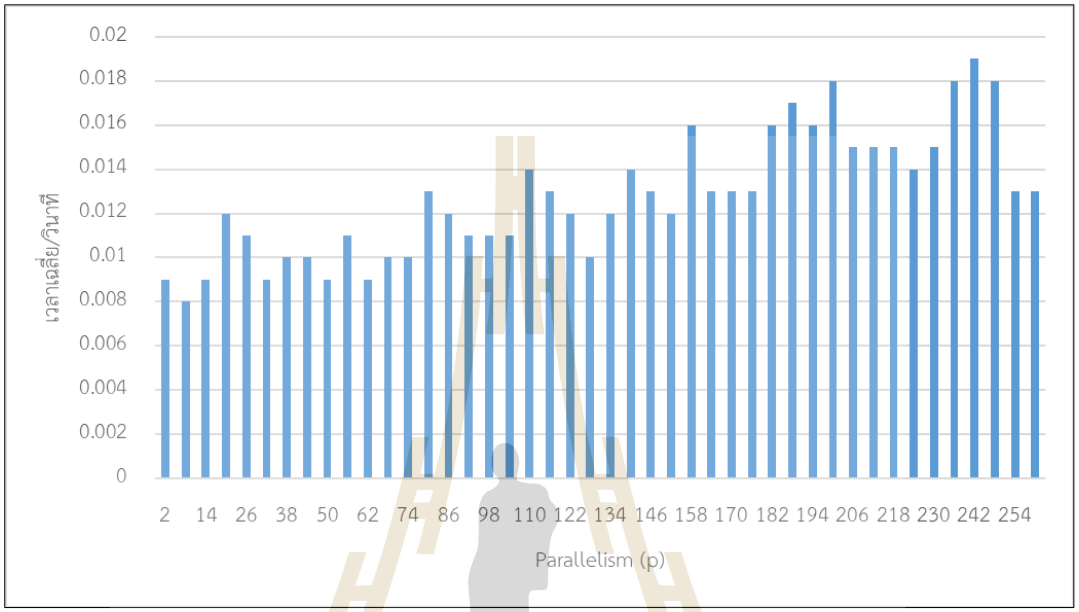
รูปที่ 4.6 เวลาที่ใช้ในการประมวลผลสำหรับ Iteration number เมื่อเพิ่มขึ้นเท่าตัวต่อครั้ง

รูปที่ 4.6 แสดงผลการทดลองของพารามิเตอร์ Iteration number (t) ที่ถูกเปลี่ยนตั้งแต่ 2 - 254 t โดยถูกเพิ่มขึ้นครั้งละ 6t จะเห็นได้ว่า กราฟนั้นแสดงผลการเพิ่มในรูปแบบ Exponential growth เช่นกัน ซึ่งผลของงานวิจัยนี้มีค่าเวลาที่เพิ่มขึ้นอย่างช้าในช่วงแรกและเพิ่มขึ้นอย่างรวดเร็วในช่วงท้าย เช่น ผลของเวลาของ -t 4 ได้เวลาเฉลี่ยที่ 0.013 วินาที ต่อจากนั้น -t 8 ได้เวลาเฉลี่ยที่ 0.023 วินาที ผลของเวลาของ -t 16 วินาที 0.046 และ ข้ามไปช่วงท้ายที่ ผลของเวลาของ -t 64 ได้เวลาเฉลี่ยที่ 0.161วินาที ผลของเวลาของ -t 128 ได้เวลาเฉลี่ยที่ 0.333 วินาที ผลของเวลาของ -t 256 ได้เวลาเฉลี่ยที่ 0.639 วินาที นั้นแสดงให้เห็นถึงลักษณะการเพิ่มขึ้นในรูปแบบ Exponential growth ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Iteration number มีผลชัดเจนต่อเวลาการแฮชชิ่ง การเพิ่มขึ้นของเวลาต่อรอบเฉลี่ยแล้วเพิ่มขึ้นถึง 0.0543 วินาทีต่อรอบการเพิ่มขึ้นเท่าตัว แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาในทุก ๆ การเพิ่มขนาดพารามิเตอร์นี้ และ จากกราฟจะเห็นได้ว่ามีแนวโน้มที่จะสูงขึ้นตามขนาดของ Iteration number อย่างชัดเจน

4.1.5 หาพารามิเตอร์ Parallelism (p)

โดยจะแบ่งเป็น 2 ประเภทในการเพิ่มขึ้นของขนาด Parallelism (p) ได้แก่

4.1.5.1 การเพิ่มค่าพารามิเตอร์ Parallelism (p) ตั้งแต่ 2 ถึง 242 p เพิ่มครั้งละ 6



รูปที่ 4.7 เวลาที่ใช้ในการประมวลผลสำหรับ Parallelism (p) เมื่อเพิ่มขนาดครั้งละ 6 p

รูปที่ 4.7 แสดงผลการทดลองของพารามิเตอร์ Parallelism (p) ที่ถูกเปลี่ยนตั้งแต่ 2 - 254 p โดย โดยแนวตั้งคือระยะเวลา / วินาที และแนวนอนคือค่า Parallelism (p) ที่มีหน่วยเป็น -p ถูกเพิ่มขึ้น เพิ่มครั้งละ 6 และ กราฟมีรูปแบบที่มีแนวโน้มขาขึ้น (Up Trend) ที่หมายถึง การที่จุดต่ำสุดและสูงสุดเพิ่มขึ้นเรื่อย ๆ จากการเพิ่มขนาด (traderocket, 2021) และ มีข้อสังเกตคือ ค่า p8, p32, p104, p128, p152, p224 ค่าเหล่านี้นั้นเป็นค่าที่เป็นจุดต่ำสุด ณ ช่วงขนาดย่อย ๆ ก่อนหน้าและค่าเหล่านี้เป็นค่าที่หารด้วย 8 ลงตัว ซึ่งเป็นค่า Threads ของ CPU ของเครื่องคอมพิวเตอร์ที่ทดลอง คือ 4 คูณด้วย 2 นั้นสนับสนุนข้อเท็จจริงของงานวิจัยที่ได้เคยกล่าวไว้ก่อนหน้านี้ ที่กล่าวว่าค่า Parallelism (p) ที่ทำให้การแฮช Argon2id มีเวลาที่สั้นที่สุดคือ ค่า Threads คูณ 2 (Cases, 2022) สามารถปรับไปใช้กับ Argon2i ได้เช่นกัน และ ค่าที่เวลาน้อยที่สุดคือ p 8 มากที่สุดคือ p 242

แต่ในเครื่องของการทดลองนี้ได้แบ่งให้ระบบ VM นั้นใช้เพียง 1 Threads เท่านั้น ซึ่งเป็นที่น่าสนใจอย่างยิ่งว่าเหตุใดจึงได้ผลทดลองออกมาได้ดังนี้

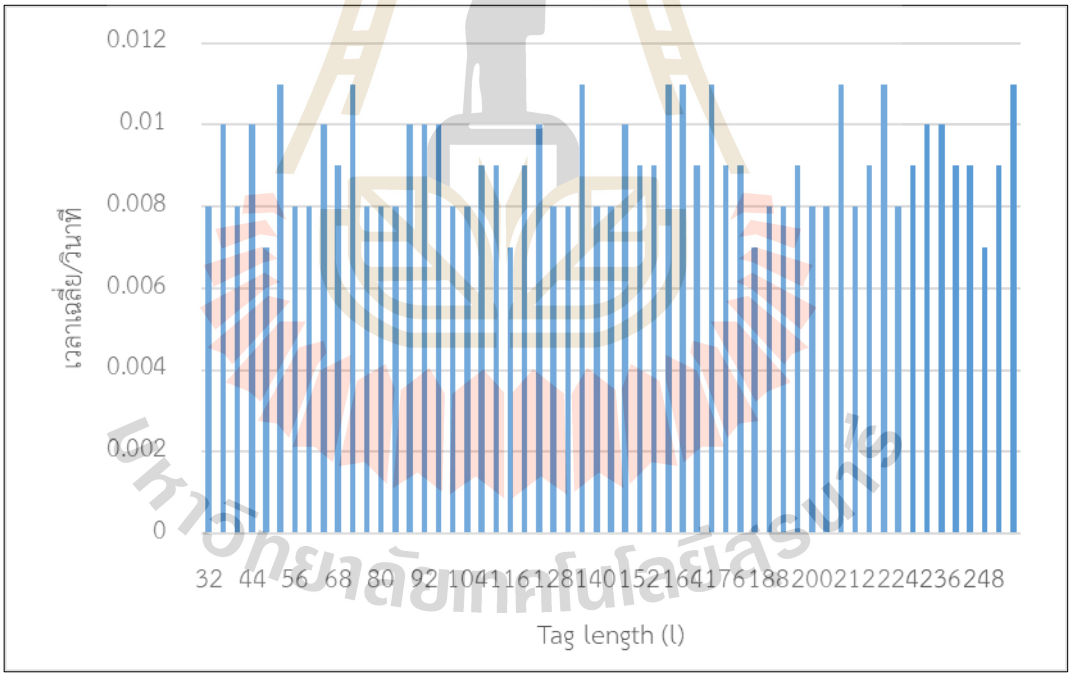
ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Parallelism ไม่มีผลต่อเวลาการแฮชชิ่งมากนัก การเพิ่มขึ้นของเวลาต่อรอบเฉลี่ยแล้วเพิ่มขึ้น 0.0002 วินาทีต่อรอบการเพิ่มขึ้นครั้งละ 6 p แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาเล็กน้อยในทุก ๆ การเพิ่มขนาดพารามิเตอร์นี้ และ จากกราฟจะเห็นได้ว่าอาจมีแนวโน้มที่จะสูงขึ้นเล็กน้อยแต่ก็ไม่มีผลต่อเวลามากนัก

เหตุผลที่ทำให้การเพิ่มค่า Parallelism (p) นั้นอาจจะมีผลทำให้เวลาในการแฮชชิ่งเพิ่มขึ้นเล็กน้อยเนื่องจาก การเพิ่มค่า Parallelism นั้นจะเป็นการแบ่งย่อยให้ CPU แบ่งงานให้เป็นงานย่อย ๆ มากขึ้น เปรียบสเมือนกับการที่เรา ยกของจากห้องหนึ่งไปอีกห้องหนึ่ง ในปริมาณที่เท่าเดิม ในรอบแรกเราแบ่งให้ของยกแค่ 2 ครั้งจึงย้ายหมดพอดี หรือ รอบที่สอง เราแบ่งให้ต้องยกของ 6 ครั้งจึงจะย้ายหมดพอดี รวมไปถึงการที่ Argon2i นั้นจะต้องดึง RAM มาใช้ในการคำนวณที่แบ่งย่อยออกมามากขึ้นอีก ดังนั้นการเพิ่มค่า Parallelism จึงอาจจะมีผลต่อเวลาในการแฮชชิ่งเล็กน้อย และการปรับค่า Parallelism จึงควรจะปรับให้ เหมาะสมกับความสามารถของ CPU ในการแบ่งแยกในการทำงานให้เหมาะสม

4.1.6 หาพารามิเตอร์ Tag length (l)

โดยจะแบ่งเป็น 2 ประเภทในการเพิ่มขึ้นของขนาด Tag length (l) ได้แก่

4.1.6.1 การเพิ่มค่าพารามิเตอร์ Tag length (l) ตั้งแต่ 32 ถึง 256 เพิ่มครั้งละ 6



รูปที่ 4.8 เวลาที่ใช้ในการประมวลผลสำหรับ Tag length เมื่อเพิ่มขึ้นครั้งละ 4 bit

รูปที่ 4.8 แสดงผลการทดลองของพารามิเตอร์ Tag length ที่ถูกเปลี่ยนตั้งแต่ 32 - 256 bit โดยแนวตั้งคือระยะเวลา / วินาที และแนวนอนคือค่า Tag length ที่มีหน่วยเป็น -l ถูกเพิ่มขึ้นครั้งละ 4 bit จากกราฟจะเห็นได้ว่าค่าเวลาไม่ได้แนวโน้มที่จะเพิ่มขึ้นเลย มีแต่เพียงค่าขึ้น ๆ ลง ๆ ในค่าสูงสุดต่ำสุดเท่า ๆ กันเท่านั้น และค่าที่ได้เวลาน้อยที่สุดคือ -l 32 และ มากที่สุดคือ -l 172

ซึ่งผลลัพธ์การทดลองนี้แสดงเห็นว่าขนาดของ Tag length ไม่มีผลต่อเวลาการแฮชชิ่งอย่างเห็นได้ชัด เหตุว่าค่าที่แตกต่างกันของเวลาระหว่างค่าที่นานที่สุดและเร็วที่สุดคือ 0.004 วินาทีเท่านั้น และ จากกราฟจะเห็นได้ว่ามีแนวโน้มที่เป็นค่าคงที่

ผลการทดลองนั้นแสดงให้เห็นถึงพารามิเตอร์ที่มีผลต่อเวลาน้อยที่สุดได้แก่

ตารางที่ 4.1 ผลสรุปค่าพารามิเตอร์ที่ใช้เวลาน้อยที่สุด

พารามิเตอร์	ขนาด	เวลา
รหัสผ่าน	28 ตัวอักษร	0.0086 วินาที
ค่า Salt	24 ตัวอักษร	0.0084 วินาที
Memory size (k)	-k 4000	0.008 วินาที
Iteration number (t)	-t 2	0.008 วินาที
Parallelism (p)	-p 8	0.009 วินาที
Tag length (l)	-l 32 Byte	0.008 วินาที

ตารางที่ 4.1 แสดงถึงผลการทดลองที่ได้เวลาที่น้อยที่สุดจากการทดลองทั้งหมด ได้แก่ รหัสผ่านขนาด 28 ตัวอักษร , ค่า Salt ขนาด 24 ตัวอักษร , Memory size (k) ขนาด -k 4000 ซึ่งเป็นขนาดเล็กที่สุด , Iteration number (t) ขนาด -t 2 ซึ่งเป็นขนาดเล็กที่สุด , Parallelism (p) -p 2 ซึ่งเป็นขนาดเล็กที่สุด , Tag length (l) -l 32 ซึ่งเป็นขนาดเล็กที่สุด และจะนำค่าเหล่านี้ไปใช้ในการคำนวณ Avalanche Effect ต่อไป และหากนำค่าเหล่านี้มารวมกันแล้วแฮชชิ่ง ได้ค่าเฉลี่ยคือ 0.00925 วินาที

ผลการทดลองนั้นแสดงให้เห็นถึงพารามิเตอร์ที่มีผลต่อความเร็วในการแฮชชิ่งได้แก่

ตารางที่ 4.2 ผลสรุปค่าพารามิเตอร์ที่ใช้เวลามากที่สุด

พารามิเตอร์	ขนาด	เวลา
รหัสผ่าน	18 ตัวอักษร	0.0116 วินาที
ค่า Salt	38 ตัวอักษร	0.0122 วินาที
Memory size (k)	-k 4096000	9.638 วินาที
Iteration number (t)	-t 254	0.662 วินาที
Parallelism (p)	-p 242	0.019 วินาที
Tag length (l)	-l 172	0.0112 วินาที

ตารางที่ 4.2 แสดงถึงผลทดลองที่ได้เวลาที่มากที่สุดจากการทดลองทั้งหมดได้แก่ รหัสผ่านขนาด 18 ตัวอักษร , ค่า Salt ขนาด 38 ตัวอักษร , Memory size (k) ขนาด -k 4096000 ซึ่งเป็นขนาดใหญ่ที่สุด , Iteration number (t) ขนาด -t 254 ซึ่งเป็นเกือบใหญ่ที่สุด , Parallelism (p) -p 242 ซึ่งเป็นขนาดเกือบใหญ่ที่สุด , Tag length (l) -l 172

ในการที่จะทำให้เห็นภาพการเปรียบเทียบของการเพิ่มขึ้นของเวลาในการแฮชชิ่งกับในแต่ละพารามิเตอร์ที่มีขนาดและลักษณะในการปรับค่าที่แตกต่างกันนั้น เพื่อที่จะเห็นภาพได้อย่างชัดเจน ผู้ทดลองจึงนำค่าพารามิเตอร์มาเพิ่มขึ้นจากค่าที่ทดลองค่าเริ่มต้น เป็นเท่าตัวขนาด 1 เท่าตัว เพื่อให้เห็นว่าการปรับ พารามิเตอร์ตัวไหนนั้นมีผลต่อเวลามากที่สุด

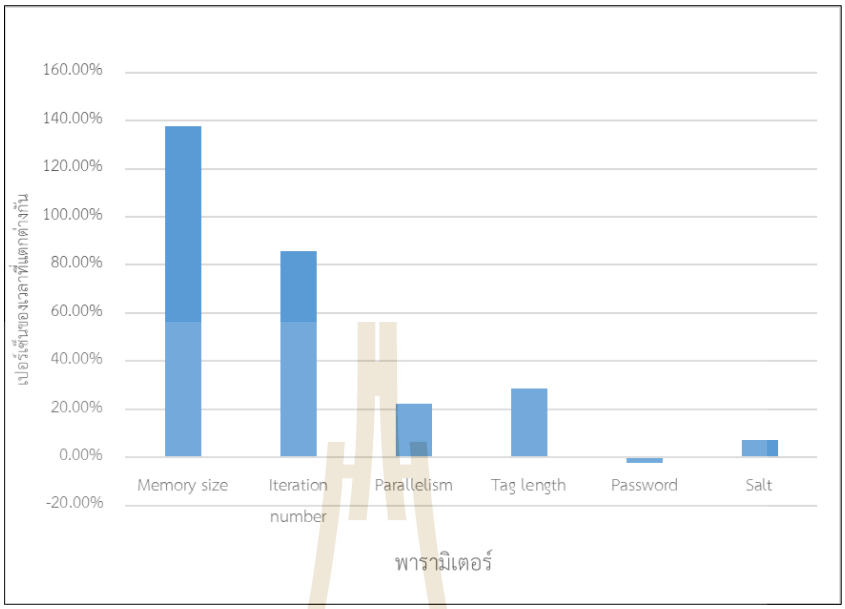
โดยค่าที่ได้นั้นจะนำมาจากค่าเวลาจากผลการทดลองที่ได้ทดลองมาก่อนหน้านี้แล้ว จึงนำค่าเหล่านั้นมาเลือกเฉพาะที่เป็นค่าเริ่มต้นและ ค่าที่เพิ่มค่าจากค่าเริ่มต้นไป 1 เท่าตัว แสดงให้เห็นค่าของพารามิเตอร์ได้ดังตารางนี้

ตารางที่ 4.3 แสดงการเพิ่มค่าพารามิเตอร์ไป 1 เท่าตัว

ค่าพารามิเตอร์	ค่าเริ่มต้น	ค่าที่เพิ่มไป 1 เท่าตัว
Password	8 ตัวอักษร	16 ตัวอักษร
Salt	8 ตัวอักษร	16 ตัวอักษร
Memory size	4000 KiB	8000 KiB
Iteration number	2t	4t
Parallelism	2p	4p
Tag length	32 byte	64 byte

ตารางที่ 4.3 แสดงการเลือกผลการทดลอง เฉพาะที่เป็นค่าเริ่มต้น และ ค่าที่เพิ่มค่าจากค่าเริ่มต้นไป 1 เท่าตัว จากค่าที่ได้ทดลองมาแล้ว เพื่อเป็นการเปรียบเทียบความแตกต่างระหว่างพารามอเตอร์ Password , Salt, Memory size, Iteration number, Parallelism และ Tag length ว่าผลต่อความเร็วในการแฮชชิ่งมากน้อยกว่ากับแค่ไหน โดย ค่าพารามิเตอร์ Salt ค่าเริ่มต้น คือ 8 ตัวอักษร ค่าที่เพิ่มไป 1 เท่าตัวคือ 16 ตัวอักษร และ ค่าพารามิเตอร์ Memory size ค่าเริ่มต้น คือ 4000 KiB ค่าที่เพิ่มไป 1 เท่าตัวคือ 8000 KiB และ ค่าพารามิเตอร์ Iteration number ค่าเริ่มต้น คือ 2t ค่าที่เพิ่มไป 1 เท่าตัวคือ 4t และ ค่าพารามิเตอร์ Parallelism ค่าเริ่มต้น คือ 2p ค่าที่เพิ่มไป 1 เท่าตัวคือ 4p และ ค่าพารามิเตอร์ Tag length ค่าเริ่มต้น คือ 32 byte ค่าที่เพิ่มไป 1 เท่าตัวคือ 64 byte

สรุปผลต่างของเวลากับการเพิ่มขึ้นของพารามิเตอร์ไป 1 เท่าตัว คำนวณออกมาเป็นเปอร์เซ็นต์ได้ดังกราฟนี้



รูปที่ 4.9 ผลต่างของเวลากับการเพิ่มขึ้นของพารามิเตอร์ไป 1 เท่าตัว

รูปที่ 4.9 แสดงให้เห็นถึงผลต่างขอเวลาระหว่างพารามิเตอร์ที่มีขนาดกัน 1 เท่าตัว ออกมาเป็นเปอร์เซ็นต์ความแตกต่างระหว่างค่าหนึ่งและค่าที่เพิ่มขึ้น 1 เท่าตัว โดยแนวตั้งคือเปอร์เซ็นต์ของเวลาที่ต่างกันระหว่างค่าเริ่มต้นและค่าที่เพิ่มขึ้น 1 เท่าตัว และแนวนอนคือค่าพารามิเตอร์ ต่าง ๆ ซึ่งจะเห็นได้ว่า การเพิ่มค่าพารามิเตอร์ Memory size (k) ไป 1 เท่าตัวนั้นทำให้เวลานั้นช้าลงไปถึง 137.50 เปอร์เซ็นต์ เป็นการเพิ่มขึ้นที่เยอะที่สุดเมื่อเทียบกับพารามิเตอร์ตัวอื่น และพารามิเตอร์รหัสผ่าน ที่ได้เปอร์เซ็นต์ เป็นติดลบ -2.22 % เพราะกลับได้เวลาที่เร็วขึ้น 2.22 % ทั้งนี้อาจจะมีปัจจัยอื่น ๆ ที่ทำให้เวลาในการแฮชช้ำเร็วขึ้น

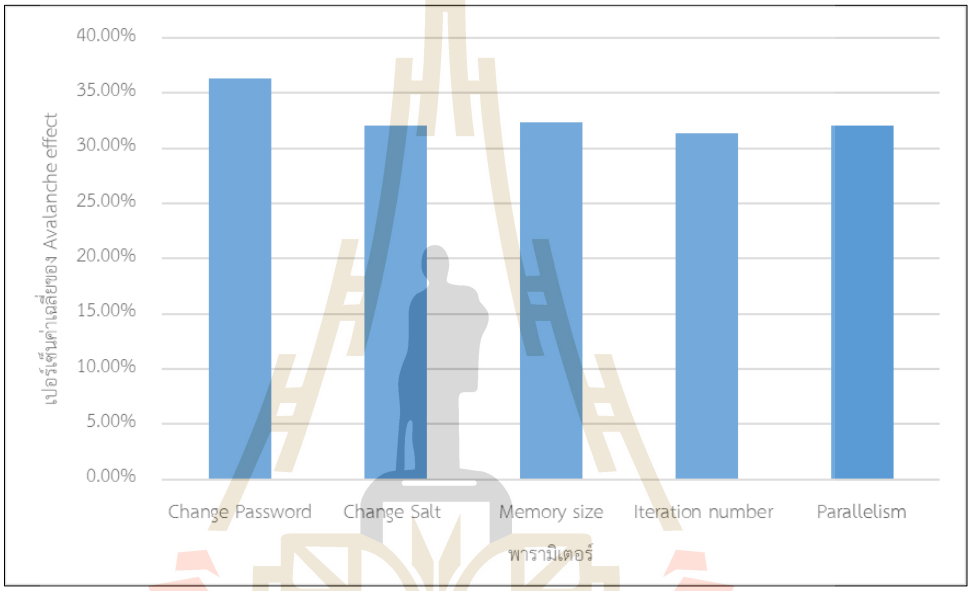
ข้อมูลทั้งหมดนี้แสดงให้เห็นว่าพารามิเตอร์ที่มีผลกระทบต่อเวลามากที่สุดคือ Memory size (k) ตามมาเป็น Iteration number (t) และส่วนที่มีผลเล็กน้อยคือ Parallelism (p) และไม่มีผลเลยคือ Tag length (l) รหัสผ่าน และ ค่า Salt ที่มีค่าการเปลี่ยนแปลงที่น้อยที่สุด นั้นแสดงให้เห็นว่า มีผลกระทบต่อเวลาน้อยหรืออาจไม่มีผลเลย

4.2 การประเมินผล Avalanche effect

ทำการประเมินแฮชชิ่ง Argon2i แบบเปลี่ยนรายพารามิเตอร์ และ อัลกอริทึมต่าง ๆ ได้แก่ Argon2i , MD5 , SHA1 และ SHA256

4.2.1 Argon2i แบบเปลี่ยนรายพารามิเตอร์

จะประเมิน Avalanche effect เป็นรายพารามิเตอร์ทั้งหมด และ ผลที่ได้ออกมาดังกราฟนี้เปรียบเทียบภาพรวมของการประเมิน Avalanche effect แบบรายพารามิเตอร์



รูปที่ 4.10 ผลของ Avalanche Effect จากการเปลี่ยนแปลงค่าของแต่ละพารามิเตอร์ของ Argon2i

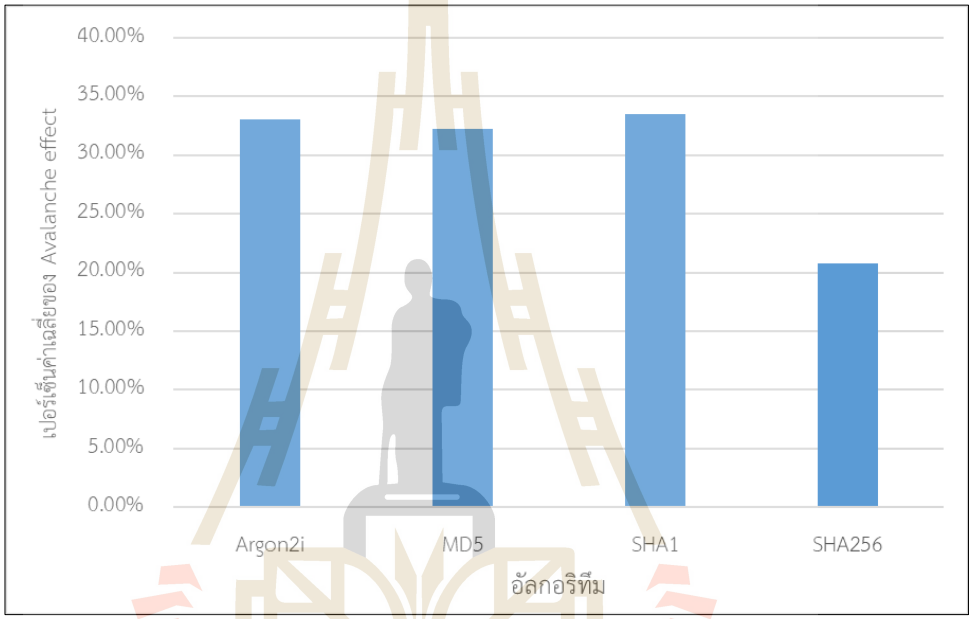
รูปที่ 4.10 แสดงผลของ Avalanche Effect จากการเปลี่ยนแปลงค่าของแต่ละพารามิเตอร์ของ Argon2i ได้แก่ Password ,Salt, Memory size, Iteration number, Parallelism ซึ่งจะแสดงผลลัพธ์เป็นแบบผลเฉลี่ย โดยแนวตั้งคือเปอร์เซ็นต์ความต่างกันของค่าแฮช และ แนวนอนคือค่าพารามิเตอร์ ต่าง ๆ ได้แก่ Password, Salt , Memory size, Iteration number (t) และ Parallelism (p) เปอร์เซ็นต์ความต่างกันของค่าแฮช เฉลี่ยที่ 36.29% 32.03% ,32.30% , 31.37% และ 32.03% ตามลำดับ และในกรณีที่เปลี่ยนพารามิเตอร์ทั้งหมดพร้อมกันจะอยู่ในการทดลอง 4.2.2

จะเห็นได้ว่าค่า Avalanche effect ของ Password จะสูงที่สุด และ Iteration number จะเป็นค่าที่ต่ำที่สุด นั้นแสดงให้เห็นว่า การเปลี่ยนพารามิเตอร์ Password เพียงอย่างเดียว จะทำให้ค่าแฮชมีความแข็งแกร่งมากขึ้นมากที่สุด เพราะ หาก Avalanche Effect นั้นมีค่าเปอร์เซ็นต์ความต่างกันของค่าแฮชที่สูงก็แปลว่าค่าแฮชชิ่งนั้นคุณสมบัติในการสุ่มที่สูง และ สามารถป้องกันการชน

กัน (Collisions) ของค่าแฮชยิ่งได้ดียิ่งขึ้น แต่ในทางกลับกันการเปลี่ยนพารามิเตอร์ Iteration number จะทำให้ค่าแฮชมีความแข็งแกร่งมากขึ้นน้อยที่สุด

4.2.2 Argon2i แบบเปลี่ยนทุกพารามิเตอร์ และ แฮชชิ่งอัลกอริทึมอื่น ๆ

จะประเมิน Avalanche effect ด้วยการเปลี่ยนพารามิเตอร์ทุกตัว และผลที่ได้ออกมาตามกราฟนี้

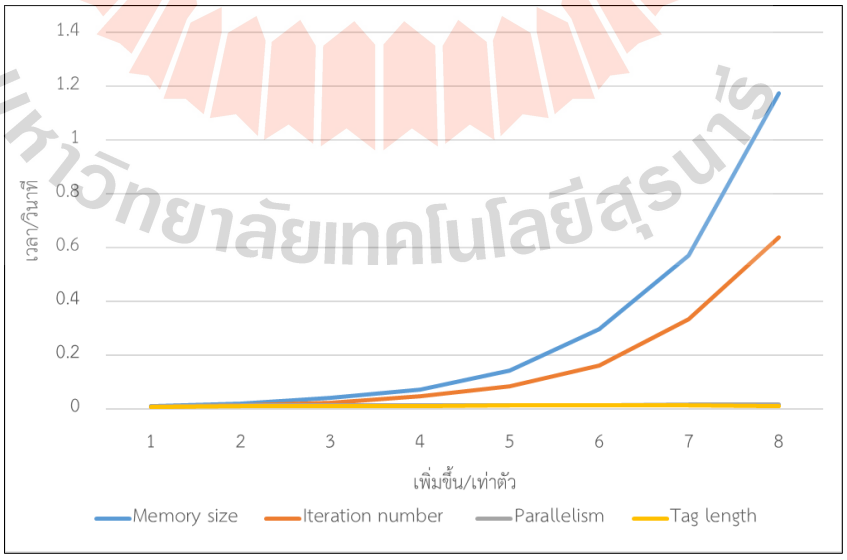


รูปที่ 4.11 การเปรียบเทียบ Avalanche Effect ของแฮชฟังก์ชัน

รูปที่ 4.11 แสดงให้เห็นผลของการทดลอง Avalanche effect เป็นรายอัลกอริทึมในรูปแบบกราฟ โดยแนวตั้งคือเปอร์เซ็นต์ความต่างกันของค่าแฮช และ แนวนอนคืออัลกอริทึมต่าง ๆ ได้แก่ Argon2i ที่ใช้วิธีเปลี่ยนไปพร้อม ๆ กันทุกพารามิเตอร์ตามวิธี ที่ 4.2.1 , MD5, SHA1 , SHA256 และได้ผลลัพธ์เท่ากับ 33.01% , 32.27%, 33.50%, และ 20.78% ตามลำดับ ซึ่งจะแสดงผลลัพธ์เป็นแบบผลเฉลี่ย

จะเห็นได้ว่าผลลัพธ์ Avalanche effect ของอัลกอริทึม SHA1 เมื่อแปลงค่าแฮชซึ่ง outputขนาด 40 ตัวอักษร หรือ 160 bit ได้เลขฐาน 2 ออกมาทั้งหมด 320 ตัวอักษร และเมื่อเปรียบเทียบกับ MD5 ที่มีขนาดของค่าแฮชซึ่ง output ที่มีขนาดที่ใกล้เคียงกันนั้นจะเห็นได้ว่า SHA1 นั้นได้ค่า Avalanche Effect เท่ากับ 33.50% ส่วน MD5 เท่ากับ 32.27% นั้นแปลว่า SHA1 มีความปลอดภัยที่มากกว่าMD5 อยู่พอสมควรเมื่อเปรียบเทียบกันด้วยขนาดตัวอักษรที่ใช้ในการเปรียบเทียบที่ใกล้เคียงกัน และ SHA256 เมื่อแปลงค่าแฮชซึ่ง output ขนาด 128 ตัวอักษร หรือ 256 bit ด้วยวิธีการ ASCII แล้วจะได้เลขฐาน 2 ออกมาทั้งหมด 512 ตัวอักษร และเมื่อเปรียบเทียบกับ Argon2i ที่มีขนาดของค่าแฮชซึ่ง output ที่มีขนาดที่เท่ากันนั้นจะเห็นได้ว่า Argon2i นั้นได้ค่า Avalanche Effect เท่ากับ 33.01% ส่วน SHA256 ทำได้เพียง 20.78% นั้นแปลว่า Argon2i มีความปลอดภัยมากกว่า SHA256 อยู่ถึง 12.23% และหากมองในภาพรวมโดยไม่คำนึงถึงขนาดของตัวอักษรของแฮชซึ่ง output เลยนั้น อัลกอริทึมที่ได้ค่า Avalanche effect มากที่สุดคือ SHA1 รองลงมาคือ Argon2i , MD5 และ SHA256 ตามลำดับ และหากมองในมุมมองที่ขนาดของตัวอักษรเท่า ๆ กันนี้ผู้วิจัยมองว่า Argon2i นั้นจะเป็นอัลกอริทึมที่ปลอดภัยที่สุดเนื่องจาก การเปรียบเทียบกันระหว่าง ตัวอักษร 512 ตัวโดยที่ไม่ซ้ำกันนั้นเป็นเรื่องที่เกิดขึ้นได้ยากกว่าการเปรียบเทียบด้วยขนาด 320 ตัวอักษร และหากมากในด้านการชนกัน (Collisions) ของค่าแฮช โอกาสของแฮชซึ่ง output มีมากกว่าก็มีโอกาสที่น้อยกว่า รวมไปถึง คุณสมบัติพิเศษของ Argon2i ที่เราสามารถเพิ่มลดได้อย่างอิสระ นั้นทำให้โอกาสของการชนกันนั้นน้อยลงอีก

และในส่วนของพารามิเตอร์ที่มีผลต่อเวลานั้นจะมีพารามิเตอร์ดังนี้ คือ Memory size (k), Iteration number (t), Parallelism (p) และ Tag length (l) ซึ่งสามารถอธิบายการเพิ่มขึ้นของแต่ละพารามิเตอร์ ได้ดังกราฟนี้



รูปที่ 4.12 เปรียบเทียบการเพิ่มขึ้นของเวลาเมื่อทำการเพิ่มค่าพารามิเตอร์ขึ้นตั้งแต่ 1-8 เท่าตัว

รูปที่ 4.12 แสดงให้เห็นถึงการเพิ่มขึ้นของเวลาเมื่อเราเพิ่มค่าพารามิเตอร์ขึ้นครั้งละ 1 เท่าตัวไปเรื่อย ๆ จนถึง 8 เท่าตัว ซึ่งจะเห็นว่ากราฟ ค่า Memory size เพิ่มขึ้นอย่างชัดเจนเพราะการเพิ่มค่า Memory size นั้นหมายถึงการดึงเอา RAM ของคอมพิวเตอร์มาใช้เพิ่มขึ้น จึงเป็นการใช้ทรัพยากรของคอมพิวเตอร์มากขึ้น ดังนั้นจึงทำให้การแฮชนั้นช้าลงโดยตรงที่สุด และ รองลงมาคือ Iteration number ที่เป็นการวนซ้ำเพื่อคำนวณแฮชทับไปเรื่อย ๆ ตามที่เรากำหนด ทำให้การเพิ่มค่า Iteration number นั้นเป็นการเพิ่มเวลาในการแฮชโดยตรงเช่นกัน ส่วน Parallelism และ Tag length นั้นแทบไม่มีการเปลี่ยนแปลงต่อเวลาเลย ด้วยเหตุผลดังนี้ คือ Parallelism นั้นเป็นการแบ่งให้ CPU ของคอมพิวเตอร์ทำการคำนวณให้ละเอียดขึ้นตามที่เรากำหนด ซึ่งไม่มีผลต่อเวลาในการคำนวณ และ Tag length นั้นเป็นการปรับความยาวของ Out put เท่านั้นจึงไม่มีผลต่อเวลาในการแฮช

โดยสรุปแล้วถึงว่าแม้แต่ Argon2i นั้นจะไม่ได้มีค่า Avalanche effect ที่มากที่สุด แต่ผลลัพธ์นั้นก็ไม่ได้ต่างจากอัลกอริทึมอื่น ๆ มากนัก และ ด้วยคุณสมบัติพิเศษ Argon2i ที่เราสามารถเพิ่มหรือลด แฮชซิ่ง output ได้ตามต้องการ จะช่วยลดโอกาสที่จะเกิดการชนกัน (Collisions) ของค่าแฮชได้มากขึ้น นั้นทำให้เห็นว่า แม้ว่าการใช้พารามิเตอร์ที่ใช้เวลาในการคำนวณที่น้อยที่สุด ก็ยังไม่ได้ลดความปลอดภัยของ Argon2i และยังคงมีความปลอดภัยมากกว่าอัลกอริทึม MD5, SHA1 , SHA256

บทที่ 5
สรุปและข้อเสนอแนะ

งานวิจัยนี้ถูกดำเนินการโดยสาเหตุมาจาก ปัญหาความปลอดภัยในระบบรหัสผ่าน ไม่ว่าจะเป็นการรั่วไหลของข้อมูลรหัสผ่านที่หมายถึงชุดตัวอักษรหรืออักขระที่ใช้ในการยืนยันตัวตน ที่หมายถึงการตรวจสอบว่าผู้ใช้งานเป็นตัวตนจริงและมีสิทธิ์ในการเข้าถึงข้อมูลหรือทรัพยากรในระบบที่ต้องการเข้าถึง และ การโจมตีจากผู้ไม่หวังดีใช้ เพื่อขโมยข้อมูลส่วนตัวของผู้ใช้ และ รหัสผ่าน

การโจมตีรหัสผ่านที่ผู้ไม่หวังดีใช้จะมีหลายวิธี เช่น Brute Force Attack ที่เป็นการคาดเดารหัสผ่านโดยการลองสุ่มทุกค่าเป็นไปจนกว่าจะพบรหัสถูกต้อง และ Password Dictionary Attack ที่ใช้รายการคำศัพท์หรือรหัสผ่านที่ถูกใช้บ่อยในการโจมตี รวมถึง Password Spraying Attack ที่โจมตีด้วยการกระจายรหัสผ่านที่มีความคาดเดาสูง และ Credential Stuffing ที่ใช้ข้อมูลรหัสผ่านที่หลุดมาจากระบบอื่นเพื่อโจมตีระบบอื่น ๆ ที่ใช้รหัสผ่านเดียวกัน และ การโจมตีด้วย Packet sniffing ที่ดักจับข้อมูลการสื่อสารระหว่างผู้ใช้และระบบ การโจมตีด้วยฟิชชิง (Phishing Attack) ที่ใช้เว็บไซต์ปลอมเพื่อหลอกผู้ใช้ให้เปิดเผยข้อมูลส่วนตัวของตน โดยมีเป้าหมายหลักในการขโมยรหัสผ่านและข้อมูลสำคัญอื่น ๆ ของผู้ใช้งานออนไลน์ ส่งเหล่านี้ต้องมีการป้องกันและเสริมความปลอดภัยอย่างเหมาะสมเพื่อป้องกันการโจมตีดังกล่าวอย่างมีประสิทธิภาพและปกติมากขึ้นในระบบของเรา

อย่างไรก็ตามนั้นเราสามารถแก้ไขปัญหาการรั่วไหลของรหัสผ่านได้ด้วยการย่อยข้อมูลโดยใช้แฮชชิ่งฟังก์ชัน ยกตัวอย่างเช่น MD5 แต่การใช้ MD5 ในการแฮชชิ่งนั้นไม่เหมาะสำหรับป้องกันข้อมูลรหัสผ่าน เนื่องจากมีความเสี่ยงที่ผู้โจมตีจะใช้ Rainbow table หรือช่องโหว่ช่องการชนกัน (Collisions) ของค่าแฮชชิ่งได้ ดังนั้นควรใช้วิธีการแฮชที่มีความปลอดภัยมากขึ้นเมื่อต้องการป้องกันข้อมูลสำคัญ

ซึ่งอาร์กอนสองไอ (Argon2i) จะช่วยแก้ปัญหานี้ได้ เพราะเป็นอัลกอริทึมที่มีประสิทธิภาพและปลอดภัยในการแฮชชิ่งรหัสผ่านที่ สามารถช่วยแก้ปัญหาจุดด้อยของ MD5 หรือ อัลกอริทึมอื่น ๆ เพราะสามารถป้องกันการโจมตีแบบแลกเปลี่ยน (trade off attacks) และ การชนกันของค่าแฮชชิ่ง

แต่ในการใช้ Argon2i นั้นจำเป็นต้องคำนึงถึงการเลือกพารามิเตอร์ที่เหมาะสม เพราะหากปรับค่าที่ไม่เหมาะสมนั้น อาจส่งผลต่อประสิทธิภาพของระบบโดยตรง เช่นการทำให้ระบบแฮชชิ่งช้าลง ซึ่งอาจทำให้ผู้ใช้งานรอนานและอาจมีผลกระทบทางธุรกิจ ดังนั้นเราควรให้ความสำคัญในการเลือกใช้พารามิเตอร์ ซึ่ง Argon2i นั้นจะมีพารามิเตอร์อยู่หลายตัวด้วยกัน ซึ่งพารามิเตอร์ที่สามารถปรับค่าได้ แก่พารามิเตอร์ รหัสผ่าน , ค่า Salt , Memory size (k) ,Iteration number (t) , Parallelism (p) ,Tag length (l)

ดังนั้น การทดลองนี้จะไปทำเพื่อศึกษาผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ argon2i ตามวัตถุระสงค์ของงานวิจัยนี้ที่คือ เพื่อทดลองและค้นหา พารามิเตอร์ที่เหมาะสมกับ Argon2i กับระบบต่าง ๆ โดยจะใช้วิธีเปลี่ยนพารามิเตอร์ทีละตัวแล้วบันทึกผลลัพธ์ของเวลานำมาเฉลี่ยกัน และ เพื่อประเมินประสิทธิภาพทางด้านเวลา สำหรับ การทำงานของ Argon2i โดยหลังจากที่ได้ผลเวลาที่น้อยที่สุดจึงนำพารามิเตอร์ที่ทำเวลาได้น้อยที่สุดเหล่านั้นมารวมกันเพื่อแฮชชิ่งบน argon2i แล้วจึงนำผลแฮชชิ่งนั้นไปทำการประเมินด้านความปลอดภัยด้วยกระบวนการ Avalanche effect

5.1 สรุปผลการวิจัย

ผลสรุปของผลการวิจัยนี้จะแบ่งตามวัตถุประสงค์ของงานวิจัยนี้ได้แก่

5.1.1 สรุปผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i

สรุปผลของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i จากผลการวิจัยในบทที่ 4 นั้นแสดงให้เห็นชัดเจนแล้วว่าการเพิ่มขนาดของพารามิเตอร์ รหัสผ่าน , Salt และ Tag length (l) นั้นไม่ได้มีผลต่อเวลาในการแฮชชิ่งเลย เพราะผลที่ออกมานั้นไม่ว่าจะเพิ่มขนาดของพารามิเตอร์เหล่านี้ เท่าไหร่ก็ไม่มีแนวโน้มที่ทำให้เวลาในการแฮชชิ่งเพิ่มขึ้นหรือลดเลย

และให้เห็นว่าพารามิเตอร์ที่มีผลกระทบต่อเวลามากที่สุดคือ Memory size (k) เพราะหากยิ่งปรับค่า Memory size เพิ่มขึ้นจะเป็นการเรียกใช้ค่า Memory หรือ RAM เพื่อมาใช้ในการคำนวณที่มากขึ้น และรองลงมาคือ Iteration number (t) ที่เป็นค่าที่ใช้เพื่อกำหนดให้ argon2i แฮชชิ่งซ้ำกี่ครั้งตามกำหนด จึงเป็นเหตุผลสำคัญที่ทำให้การแฮชชิ่งนั้นมีเวลาในการแฮชชิ่งที่สูงขึ้น และส่วนที่มีผลเล็กน้อยคือ Parallelism (p) ที่ได้ค่าที่เหมาะสมตามงานวิจัยที่ได้ศึกษามา (Burman, 2019) ว่าควรปรับค่าเท่ากับ 2 เท่าของค่า Treads ของ CPU ซึ่งจากผลวิจัยนี้แล้ว สามารถสรุปได้ว่าเป็นจริงสำหรับเครื่องคอมพิวเตอร์ที่ทำการทดลอง

5.1.2 สรุปผลการประเมินด้านความปลอดภัยของ Argon2i

สรุปการประเมินผล Avalanche effect ที่ทำการประเมิน 2 แบบ คือ การประเมินเป็นรายพารามิเตอร์ ที่การเปลี่ยนค่า Password จะทำให้ได้ค่า Avalanche Effect ที่ สูงที่สุด รองลงมาเป็นการเปลี่ยนค่า Memory size และ การเปลี่ยน Salt และ Parallelism จะได้ค่าที่เท่ากัน และ Iteration number จะเป็นค่าที่ต่ำที่สุด นั้นแสดงให้เห็นว่า การเปลี่ยนพารามิเตอร์ Password เพียงอย่างเดียว จะทำให้ค่าแฮชมีความแข็งแกร่งมากขึ้นมากที่สุด และ การประเมินกับอัลกอริทึมอื่น ๆ ที่ SHA1 ได้ค่าที่สูงที่สุดเพราะจำนวนค่าแฮชชิ่ง output ที่น้อยกว่า ทำให้มีโอกาสที่จะซ้ำกันได้น้อย

กว่า argon2i ซึ่งได้ค่าน้อยกว่าเพราะมีจำนวน ค่าแฮชชิ่ง output ที่มากกว่า รองลงมาคือ MD5 ที่มีค่าแฮชชิ่ง output ขนาดใกล้เคียงกับ SHA1 ที่มีเยอะกว่าเพียง 8 ตัวอักษร และ น้อยที่สุดคือ SHA256 ที่มีค่าแฮชชิ่ง output เท่ากับ argon2i ซึ่งจะเห็นได้ว่า หากเปรียบเทียบกับ แฮชชิ่ง output ที่ทำกันนั้น argon2i จะเป็นอัลกอริทึมที่ปลอดภัยที่สุด หรือ หากไม่หากเปรียบเทียบที่แฮชชิ่ง output ค่า Avalanche Effect ของ Argon2i ก็ไม่ได้น้อยไปกว่า SHA1 มากนัก

จากผลการทดลองที่ปรากฏออกมานั้น ผู้วิจัยแนะนำว่าควรให้ความสำคัญกับพารามิเตอร์ตามลำดับดังนี้ รหัสผ่าน ซึ่งในที่นี้นับเป็นพารามิเตอร์ชนิดหนึ่ง ซึ่งเป็นที่แน่นอนอยู่แล้วว่าควรให้ความสำคัญเพราะมีผลต่อความปลอดภัยของ Argon2i ที่สุดอ้างอิงจากการประเมินความปลอดภัยด้วย Avalanche Effect และ Memory Size เพราะเป็นพารามิเตอร์ที่ส่งผลต่อเวลามากที่สุดและมีผลต่อความปลอดภัยรองลงมา และ Iteration Number เพราะเป็นพารามิเตอร์ที่ส่งผลต่อเวลารองลงมา และ Parallelism เพราะเป็นพารามิเตอร์ที่ส่งผลต่อเวลาเล็กน้อยและมีผลต่อความปลอดภัยอยู่พอสมควร และ Salt เพราะเป็นพารามิเตอร์ที่ไม่ส่งผลต่อเวลาเลยและมีผลต่อความปลอดภัยเทียบเท่ากับพารามิเตอร์ Parallelism และ Tag Length (L) เพราะเป็นพารามิเตอร์ที่ไม่ส่งผลต่อเวลาเลย และ ผลของ Avalanche Effect ก็บ่งชี้ได้ว่า หากใช้ค่าพารามิเตอร์ที่ทำให้เวลาในการแฮชชิ่งน้อยที่สุด ก็ไม่ได้ทำให้ความปลอดภัยของ Argon2i น้อยไปกว่าอัลกอริทึมในการแฮชชิ่งอื่น ๆ ที่ถูกใช้กันอยู่ในปัจจุบันแต่อย่างใด

5.2 การประยุกต์ผลการวิจัย

จากผลการวิจัยทั้งหมดนี้สามารถนำไปประยุกต์ใช้งานจริงได้ กับฟังก์ชัน Registration และ Authentication ของระบบที่ต้องการใช้ argon2i เพื่อแฮชชิ่งรหัสผ่านของผู้ใช้และเก็บไว้ในระบบฐานข้อมูล และเพื่อให้ argon2i สามารถแฮชชิ่งได้เวลาที่รวดเร็วที่สุดและยังคงปลอดภัย สามารถทำได้โดยปรับค่าพารามิเตอร์ดังนี้ Memory size ปรับค่าเท่ากับ 4000 KiB และ Iteration number ปรับค่าเท่ากับ 2 และ Parallelism (p) ปรับค่าเท่ากับ 8 ซึ่งได้มาจากค่า Threads ของ CPU เครื่องหลักที่ทำการทดลอง ซึ่งไม่เกี่ยวกับที่แบ่งให้กับระบบ VM คูณด้วย 2 ตามงานวิจัยของ (Cases, 2022) และรหัสผ่าน , Salt และ Tag length (l) ถึงแม้ว่าจากข้อสรุปแล้วจะไม่ได้มีผลต่อเวลาในการแฮชชิ่งเลย แต่สามารถปรับค่า รหัสผ่าน ให้มีความยาวเท่ากับ 28 ตัวอักษร และ ค่า Salt ให้มีความยาวเท่ากับ 24 ตัวอักษร และ Tag length (l) ให้ปรับเป็นขนาด 32 bits ซึ่งเป็นค่าจะผลทดลองที่ได้เวลาที่ต่ำที่สุด

และหากต้องการให้เวลาในการแฮชชิ่งนั้นลดลงอีก จากผลการทดลองทั้งหมดเหล่านี้ผู้วิจัยแนะนำว่าควรเพิ่มประสิทธิภาพของ CPU เป็นหลัก และ รองลงมาคือ ประสิทธิภาพของ Memory

5.3 ข้อเสนอแนะในการวิจัยครั้งต่อไป

5.3.1 ควรที่จะให้ฮาร์ดแวร์การทดลองที่ตรงกับการใช้งานจริงที่สุด เช่นกับ เครื่องคอมพิวเตอร์แม่ข่าย ที่มีข้อกำหนดของเครื่อง (Specifications) หรือเป็นรุ่นใช้งานกับระบบจริง ๆ เพื่อให้สามารถหาค่าพารามิเตอร์ที่เหมาะสมกับฮาร์ดแวร์ที่รันบนระบบนั้น ๆ ได้มากที่สุด

5.3.2 ควรที่จะเพิ่มฮาร์ดแวร์ในทดลองที่มากกว่า 1 ฮาร์ดแวร์ เพื่อให้เห็นความแตกต่างระหว่างผลของเวลาในการแฮชชิ่งกับฮาร์ดแวร์ต่าง ๆ ว่ามีผลที่แตกต่างกันหรือไม่

5.3.3 ควรขยายขอบเขตของพารามิเตอร์ในการทดลอง เพื่อการศึกษาผลการทดลองที่กว้างและสามารถมองเห็นความแตกต่างของเวลาในการทดลองได้มากขึ้น

5.3.4 ควรเพิ่มการทดลองเพื่อหาจุดที่เหมาะหรือพอดีระหว่างการปรับพารามิเตอร์ให้เร็วที่สุดและการปรับค่าให้ปลอดภัยมากที่สุด

5.3.5 ควรเพิ่มการเพื่อวัดความปลอดภัยของค่าแฮชชิ่ง Argon2i ที่แฮชชิ่งได้เวลาที่น้อยที่สุด ให้มากกว่า Avalanche Effect ที่เน้นไปที่หารวัดคุณสมบัติการสุ่มและการชนกัน (Collision) ของค่าแฮชชิ่ง

รายการอ้างอิง

5 MIN READ. (2022). *10 อันดับ คริปโต ที่ครอง มูลค่าตลาด โลกมากที่สุด*. เข้าถึงได้จาก moneybuffalo.in.th: https://www.moneybuffalo.in.th/cryptocurrency/10-market-cap-of-crypto

Acisonline. (2023). *Information Security ความมั่นคงปลอดภัยของข้อมูลสารสนเทศ*. เข้าถึงได้จาก acisonline.net: https://www.acisonline.net/?p=8647

Alexander S. Gillis. (2022). *DEFINITION DEFINITION*. เข้าถึงได้จาก techtarget.com: https://www.techtarget.com/whatis/definition/rainbow-table

Andreas Auernhammer. (2018). *How to Hash and Verify Passwords With Argon2 in Go*. เข้าถึงได้จาก alexedwards.net: https://www.alexedwards.net/blog/how-to-hash-and-verify-passwords-with-argon2-in-go

Andrew Magnusson. (2023). *The Definitive Guide to Authentication*. เข้าถึงได้จาก strongdm.com: https://www.strongdm.com/authentication

Biryukov, A., Dinu, D., & Khovratovich, D. (2017). *Argon2: the memory-hard function for password hashing and other*. March 24. Retrieved from Rockyou 2021 รวมรหัสผ่านรั่วไหลครั้งใหญ่ที่สุด กว่า 8.4 พันล้านรายการ: https://www.cyfence.com/it-360/rock-you-2021-includes-the-biggest-password-leak-8-4-billion/

Bryan Burman. (2019). *How to Choose the Right Parameters for Argon2*. เข้าถึงได้จาก twelve21: https://www.twelve21.io/how-to-choose-the-right-parameters-for-argon2/

Cameron Hashemi-Pour. (2023). *DEFINITION biometric authentication*. เข้าถึงได้จาก techtarget. com/ : https: / / www. techtarget. com/ searchsecurity/ definition/ biometric-authentication

Charles R. Portwood. (2016). *PHP RFC: Argon2 Password Hash*. เข้าถึงได้จาก wiki.php.net: https://wiki.php.net/rfc/argon2_password_hash

Crime Cases. (2022). *6 Types of Database Attacks Hackers Use to Obtain Unauthorized Access*. เข้าถึงได้จาก salvationdata.com: https://www.salvationdata.com/crime-cases/6-types-of-database-attacks-hackers-use-to-obtain-unauthorized-access/

crowdstrike. (2022). *PASSWORD SPRAYING.* เข้าถึงได้จาก crowdstrike.com: https://www.crowdstrike.com/cybersecurity-101/password-spraying/

Daniel Dinu , Dmitry Khovratovich , Simon Josefsson Alex Biryukov. (2021). *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications.* เข้าถึงได้จาก datatracker.ietf.org: https://datatracker.ietf.org/doc/draft-irtf-cfrg-argon2/13/

Darshana Upadhyay, Nupur Gaikwad, และ Marzia Zaman. (2022). Investigating the Avalanche Effect of Various. *Natural Sciences and Engineering Research Council (NSERC)*, 112473-112486.

Defuse Security. (2021). *Making Password Cracking Harder: Slow Hash Functions.* เข้าถึงได้จาก crackstation.net: https://crackstation.net/hashing-security.htm

Descope. (2023). *What is Password-Based Authentication?* เข้าถึงได้จาก https://www.descope.com/: https://www.descope.com/learn/post/password-authentication

Donya Petchyodsri. (2022). *15 รหัสผ่าน ที่ความปลอดภัยยอดแย่แห่งปี 2022.* เข้าถึงได้จาก digitalmore: https://digitalmore.co/15-worst-passwords-of-2022/

Emin Huseynov, และ Jean-Marc Seigneur. (2017). *Computer and Information Security Handbook (Third Edition)*. เข้าถึงได้จาก sciencedirect.com: https://www.sciencedirect.com/topics/computer-science/hardware-token

F. Mohammed. (2016). A Review Of Authentication Methods. *In Proceedings Of International Journal of Scientific & Technology*, 245-249.

Jason Andress. (2014). *The Basics of Information Security (Second Edition)*. เข้าถึงได้จาก sciencedirect.com: https://www.sciencedirect.com/topics/computer-science/hardware-token

Jessica Hopkins. (2023). *Binary vs ASCII – Their Relationship, Differences, and Embedded Applications.* เข้าถึงได้จาก https://www.totalphase.com/: https://www.totalphase.com/blog/2023/05/binary-ascii-relationship-differences-embedded-applications/

Julien Piatek. (2017). *Hash functions for newbies.* เข้าถึงได้จาก medium.com: https://pjulien.medium.com/blockchain-for-newbies-1-hash-functions-1fb2563bc67c

Kanyawee Jin. (2022). *ระบบ RFID คือ อะไร? สุดยอดเทคโนโลยีคลื่นวิทยุ มีบทบาทอย่างไรกับแวดวงธุรกิจ?* เข้าถึงได้จาก https://www.zipeventapp.com /blog/ 2022/08/18/what-is-rfid/: https://www.zipeventapp.com/blog/2022/08/18/what-is-rfid/

Kittitat. (2019). *IP Address คืออะไร?* เข้าถึงได้จาก hostinglotus.com: https://www.hostinglotus.com/blog/2019/12/16/ip-address-คืออะไร/

l3 uch. (2564). *Core, Thread และ Clock Speed คืออะไร ?.* เข้าถึงได้จาก thaiware: https://tips.thaiware.com/1740.html

Madelyn Bacon. (2021). *password.* เข้าถึงได้จาก techtarget.com: https://www.techtarget.com/searchsecurity/definition/password

Mirren McDade. (2022). *5 Reasons To Avoid Password Reuse.* เข้าถึงได้จาก expertinsights.com: https://expertinsights.com/insights/5-reasons-you-should-never-reuse-passwords/

Multipass. (2023). *multipass*. เข้าถึงได้จาก multipass: https://multipass.run/

Neal Mueller. (2021). *Credential stuffing.* เข้าถึงได้จาก owasp.org: https://owasp.org/www-community/attacks/Credential_stuffing

New Naveen. (2020). *รู้จักการเติบโตแบบ exponential และ logistic จาก COVID-19.* เข้าถึงได้จาก medium.com: https://medium.com/

NT cyfence. (2021). *IT 360º ไอที ง่ายๆ รอบๆ ตัว*. เข้าถึงได้จาก cyfence: https://www.cyfence.com/it-360/rock-you-2021-includes-the-biggest-password-leak-8-4-billion/

Password Hashing Competition. (2019). *Password Hashing Competition*. เข้าถึงได้จาก password-hashing.net: https://www.password-hashing.net/

Pimlapat. (2022). *ฟิชชิ่ง (Phishing) คืออะไร? รู้จักภัย 8 ประเภทบนโลกออนไลน์*. เข้าถึงได้จาก primal.co.th: https://www.primal.co.th/th/seo/what-is-phishing/

Rahul Awati. (2021). *password salting.* เข้าถึงได้จาก techtarget.com: https://www.techtarget.com/searchsecurity/definition/salt

Rapid7. (2022). *What are Brute-Force and Dictionary Attacks?* เข้าถึงได้จาก rapid7.com: https://www.rapid7.com/fundamentals/brute-force-and-dictionary-attacks/

Support. (2020). *Brute Force Attack*. เข้าถึงได้จาก hostpacific.com: https://www.hostpacific.com/what-is-a-brute-force-attack/

Swetha vazhakkat. (2022). *Avalanche Effect in Cryptography.* เข้าถึงได้จาก geeksforgeeks.org: https://www.geeksforgeeks.org/avalanche-effect-in-cryptography/

Thompson, E. (2005). MD5 collisions and the impact on computer forensics. *Digital Investigation*, 36-40.

traderocket. (2021). *แนวโน้ม คืออะไร มี 3 แบบ Uptrend, Downtrend, Sideway.* เข้าถึงได้จาก traderocket.net: https://traderocket.net/what-is-trend-uptrend-downtrend-sideway/

University of Regina and The Pacific Institute for the Mathematical. (ม.ป.ป.). *What is the definition of linear growth?* เข้าถึงได้จาก mathcentral.uregina.ca: https://mathcentral.uregina.ca/qq/database/qq.09.06/s/rebecca1.html

Viki Green. (2016). *Impact of slow page load time on website performance*. เข้าถึงได้จาก https://medium.com/: https://medium.com/@vikigreen/impact-of-slow-page-load-time-on-website-performance-40d5c9ce568a

Winyou Adisaktrakoon. (2020). *จะง่ายไปไหน: Hash Function*. เข้าถึงได้จาก medium.com: https://medium.com/@winyou.info/จะง่ายไปไหน-hash-function-42bc705d8876

เชาวลิต สมบูรณ์พัฒนากิจ, และ ศิรปัฐช์ บุญครอง. (2014). การจัดเก็บรหัสผ่านอย่างปลอดภัยโดยใช้เทคนิคการปรับค่าซอลท์ที่เหมาะสม. *The Tenth National Conference on Computing and Information Technology*. ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ.

ปริญญา นาโท, และ ศิรปัฐช์ บุญครอง. (2017). การสร้างความปลอดภัยรหัสผ่านด้วยเทคนิกการแทรกซอลท์ร่วมกับแฮชฟังก์ชั่นโดยใช้อัลกอริทึม Bcrypy. *สารสาร มทร. อีสาน ฉบับวิทยาศาสตร์ และเทคโนโลยี*, 137-145.

ภูวดล แสงทอง. (ม.ป.ป.). *Information security*.

วนิดา แซ่ตั้ง, และ ศิรปัฐช์ บุญครอง. (2017). *การวิเคราะห์ความปลอดภัยของฟังก์ชั่นแฮช*. มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ: คณะเทคโนโลยีสารสนเทศ.

สิริพร จิตต์เจริญธรรม, เสาวภา ปานจันทร์, และ เลอศักดิ์ ลิ้มวิวัฒน์กุล. (11 มิถุนายน 2547). *การพิสูจน์ตัวตน (Authentication)*. เข้าถึงได้จาก tanasith: http://tanasith.blogspot.com/2011/06/blog-post_20.html

อานนท์ หลงหัน. (17 Dec 2014). *SQL Injection คืออะไร*. เข้าถึงได้จาก arit.rmutsv.ac.th: https://arit.rmutsv.ac.th/th/blogs/80-sql-injection-คืออะไร-757

ภาคผนวก ก.

ผลการทดลองของพารามิเตอร์ต่อความเร็วสำหรับการแฮชชิ่งของ Argon2i

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
1.หาพารามิเตอร์ "รหัสผ่าน" ที่เร็วที่สุด			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.929	
2	echo -n "VTw>2XwO" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.881	
3	echo -n "Kvkvcl2e" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.889	
4	echo -n "Pg>UY8V@" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.89	
5	echo -n "n)EwtN9V" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.934	0.9046
6	echo -n "q(eP|56o,<" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.901	
7	echo -n "XzrKn|K0KE" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.899	
8	echo -n "Dd$@Pj5r^&" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.936	
9	echo -n "pqLfDv3Xgf" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.916	
10	echo -n "Uv1qP%fx1G" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.916	0.9136
11	echo -n "V9o7.k*cBrdk" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.918	
12	echo -n "rH{L&n3uau34" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.878	
13	echo -n "Q6SYk}/SF(yi" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.877	
14	echo -n "eoE,V=x(%2W&" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.89	
15	echo -n "j#k89nHzT9k%" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.881	0.8888
16	echo -n "r1S0Sp]5D{uuw2" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.879	
17	echo -n "yyz,Eh)t)gf3eJ" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.886	
18	echo -n "q_b9trrzjY)CjN" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.888	
19	echo -n "Fa/b//wt5$9U1$" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.879	
20	echo -n "Oj~nTt8stQ<cXu" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.89	0.8844
21	echo -n "S-o4gd1{NAEPtxS}" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.868	
22	echo -n "YeW@,<D6yr$XKhz9" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.904	
23	echo -n "y0/u1)hlkT@^WdxX" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.881	
24	echo -n "nx5tx+CvB6{t$mB3" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.887	
25	echo -n "s+z12>MjUl?_[x}=" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.877	0.8834
26	echo -n "Spv>4{N(0&A.,6i/T7" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.877	
27	echo -n "U^H^d@/^k.,f<A7nGv" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.915	
28	echo -n "wu#1E4sr%JY5C_v#2A" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.874	
29	echo -n "lptNe*|4Q8<eb31-xg" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.87	
30	echo -n "O,hw6m]KgqUQk.@^vR" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.871	0.8814
31	echo -n "IEG]<rYx+pSQ|72[4]1U" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.842	
32	echo -n "StOw>Kk]ti27PRFWx0kG" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.827	
33	echo -n "kLvMZhrNqvauHVV0v6[$" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.818	
34	echo -n "CqBx,C9z?Gu^1Act_mGb" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.839	
35	echo -n "Ou3nu>2sU3vQCE~}G6j@" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.84	0.8332
36	echo -n "jM)Ty3Gp6Og3[73D2t@x2&" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.951	
37	echo -n "s%rlq2fAn|h@T,<wP4TYP^" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.95	
38	echo -n "idtu])1Dvh8@hMU5+KvNn-" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.918	
39	echo -n "SL@G_31=(x}j#br@9u%=Gn" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.963	
40	echo -n "cA>.){R(4{#yQF=OfH>s*k" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	1.03	0.9624
41	echo -n "MFLjX~U?^i$96MHo]%z3kPh8" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
42	echo -n "P0?T.Yc2BUhBUC@Jp])26A" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.835	
43	echo -n "SE6Jm+.x($c?Ew>Q+Z2zEHCU" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.821	
44	echo -n "Tr%MkzippRN%IRtx4IHl5xJa" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.811	

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
45	echo -n "mWT75uVCzr&uB>?uYuxd9rrD" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.807	0.8196
46	echo -n "U1{h|&,BxYvC($Vkkx5T5@t-xT" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.916	
47	echo -n "H1/T@k35krb{[vqu>6$Zr[%V6v" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.889	
48	echo -n "K[?F#O5UgIvQr?xt+z==rpx+8I" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.941	
49	echo -n "a.rRVT?^Q@(DjW2os]9~9zVva<" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.878	
50	echo -n "dLLSUaF_3hKq-EXn>YmxvXQx6t" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.962	0.9172
51	echo -n "N+9H8>)0$noQ(5BeSdL<ll9YyMLC" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.818	
52	echo -n "T^NbZouj&@t<<-3]Rc/k&GgHie5L" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.811	
53	echo -n "I$vjwKrWuhpUTnj]7Y?phhV>haAs" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.811	
54	echo -n "I3i|9T}k}%>tv(D(]?r}}T9LelG)" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.81	
55	echo -n "r=IlH<7+ej[fv7u,A&Ob%H3x7J.A" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.812	0.8124
56	echo -n "E4P?SL}Snkrluuu(ZH%r3+a/vd6-2u" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
57	echo -n "bq]j6_2nrNJj,_QUs.kV}bE#v.iM5f" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.846	
58	echo -n "n#S+<kpLxF-+f4J#.g_H&Fk9Niu_9u" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
59	echo -n "su~t3KsZ)&]G{Ut~9=2}[O)kn1Df99" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.816	
60	echo -n "Gjlf7U=o9#SG|Z)lrkPe8-axWQ7<R5" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.828	0.8266
61	echo -n "qu~}N_6{xwE<g2CdIR@hiQ[_uuAkvv9h" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.815	
62	echo -n "TXm}tk.<2@oxUus~DJPmhW38+Vuy>Nrr" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.829	
63	echo -n "UtOo&8.qbRKW.u4rc}%9i$lr>^JuX8c7" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
64	echo -n "Bk<VmQQJk]Fe/5x0x(l]G5{HzyXZe0bC" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
65	echo -n "ktrRG189xkI4k#kS(7.5FIlBM$&FW4xP" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.818	0.8214
66	echo -n "Hs7xIq?f5ioUxKE_DT5<[fkou%I.}VXxOg" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.817	
67	echo -n "ptYYu[I%~aZ+&S6s%pk&uW8rrP|b]x)EmZ" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.801	
68	echo -n "veaJb<ug{$C7SW$,5eB[3Uu2*K@.m19x9p" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.803	
69	echo -n "Cm*xo7AtKxi6n[x}-u7SV=(MOuplqeyB?V" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.894	
70	echo -n "FIiv9C0X(]IkLH]B2j_DN_<ti25$ufH&#T" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.828	0.8286

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
71	echo -n "WuBwPkkKudaNt,0t%Pf7xvrOp1h*e0uDkB>-" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.827	
72	echo -n "bw%Qe65rNQp<)u_vs(_yUDrdt[Isg8Uv~e%8" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.83	
73	echo -n "X<9..x?og<crevVpaux7HXBh.pu<9%87G#aA" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.811	
74	echo -n "MhrxyL&wA0Mp<~74gI{tvo{sbk~5tHS,L.YW" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.833	
75	echo -n "C0zrLE<qJyJzsmK>ekTMrk2}=hJ&,h-hrW10" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.813	0.8228
76	echo -n "jUN5BHJJhw5bpVcqm9uBsQLxPTXnSya5DNZp9M" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
77	echo -n "B9.I1Xlv?d1OEsb8ex4$~xj=6Dbryk@b+Y9B[S" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.848	
78	echo -n "JK#LS[{G+{$vxCZ>g1w@HtMP{^ym5$|i#Vk.43" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.86	
79	echo -n "b6gmtHfh&ZItvJrN#kS@sr2t@r=cj3%4n~-Utx" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	1.039	
80	echo -n "Qmvrr7a*dR4zEONWh016<xX530drDa757Bvh?f" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.885	0.8916
2.หาพารามิเตอร์ "Salt" ที่เร็วที่สุด			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
2	echo -n "g#7AE9@?" | argon2 "t#HVE22H" -i -k 400000 -t 2 -p 2 -l 32;	0.822	
3	echo -n "g#7AE9@?" | argon2 "Ln0Jf<M{" -i -k 400000 -t 2 -p 2 -l 32;	0.811	
4	echo -n "g#7AE9@?" | argon2 "K{0~M/n0" -i -k 400000 -t 2 -p 2 -l 32;	0.861	
5	echo -n "g#7AE9@?" | argon2 "i%V2*O(G" -i -k 400000 -t 2 -p 2 -l 32;	0.83	0.8296
6	echo -n "g#7AE9@?" | argon2 "jSOz9[5&w[" -i -k 400000 -t 2 -p 2 -l 32;	0.831	
7	echo -n "g#7AE9@?" | argon2 "T^iI1D~$RC" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
8	echo -n "g#7AE9@?" | argon2 "Ub{w+58~aI" -i -k 400000 -t 2 -p 2 -l 32;	0.814	
9	echo -n "g#7AE9@?" | argon2 "n:y5-NEx&q" -i -k 400000 -t 2 -p 2 -l 32;	0.816	
10	echo -n "g#7AE9@?" | argon2 "L.?[q3x8c>" -i -k 400000 -t 2 -p 2 -l 32;	0.842	0.8244
11	echo -n "g#7AE9@?" | argon2 "WmCA03*}fk)8" -i -k 400000 -t 2 -p 2 -l 32;	0.812	
12	echo -n "g#7AE9@?" | argon2 "TEpeRs*i6kEE" -i -k 400000 -t 2 -p 2 -l 32;	0.822	
13	echo -n "g#7AE9@?" | argon2 "y<Or>EKOGr4P" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
14	echo -n "g#7AE9@?" | argon2 "x=m9V?rU@:Xc" -i -k 400000 -t 2 -p 2 -l 32;	0.809	
15	echo -n "g#7AE9@?" | argon2 "sy6BJ)qK0i9," -i -k 400000 -t 2 -p 2 -l 32;	0.832	0.8202
16	echo -n "g#7AE9@?" | argon2 "JAB8r<x8kP3+Kp" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
17	echo -n "g#7AE9@?" | argon2 "j(A#TaHuP}334k" -i -k 400000 -t 2 -p 2 -l 32;	0.821	
18	echo -n "g#7AE9@?" | argon2 "H{r/jgxHU[>k4t" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
19	echo -n "g#7AE9@?" | argon2 "HSZ2Xm?=kz8CY5" -i -k 400000 -t 2 -p 2 -l 32;	0.816	
20	echo -n "g#7AE9@?" | argon2 "ykgAOmF,<?q3xW" -i -k 400000 -t 2 -p 2 -l 32;	0.811	0.8182
21	echo -n "g#7AE9@?" | argon2 "BHRY8F_zItL/G5k)" -i -k 400000 -t 2 -p 2 -l 32;	0.836	
22	echo -n "g#7AE9@?" | argon2 "OYI7PdBz8u<pa?xq" -i -k 400000 -t 2 -p 2 -l 32;	0.829	
23	echo -n "g#7AE9@?" | argon2 "w>AQG2Z|o_Ij/q5U" -i -k 400000 -t 2 -p 2 -l 32;	0.818	
24	echo -n "g#7AE9@?" | argon2 "NE{GWZk$XjUx4AB:" -i -k 400000 -t 2 -p 2 -l 32;	0.817	

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
25	echo -n "g#7AE9@?" | argon2 "yyo3c00xeZBx.LWL" -i -k 400000 -t 2 -p 2 -l 32;	0.817	0.8234
26	echo -n "g#7AE9@?" | argon2 "JM7,)w#{8e>w_tr5gY" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
27	echo -n "g#7AE9@?" | argon2 "F-5#u9JOc]xsSNXQ+h" -i -k 400000 -t 2 -p 2 -l 32;	0.828	
28	echo -n "g#7AE9@?" | argon2 "AonlFYth,^Nt2ujN:|" -i -k 400000 -t 2 -p 2 -l 32;	0.822	
29	echo -n "g#7AE9@?" | argon2 "c9k&_%meT{Rk99KWW^" -i -k 400000 -t 2 -p 2 -l 32;	0.817	
30	echo -n "g#7AE9@?" | argon2 "D{(j8TxX4$?nZ_A.99" -i -k 400000 -t 2 -p 2 -l 32;	0.826	0.8238
31	echo -n "g#7AE9@?" | argon2 "aUlJAr[hkJI<03,q|B^h" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
32	echo -n "g#7AE9@?" | argon2 "Rm,j:[x#NXzE1q7]rMuZ" -i -k 400000 -t 2 -p 2 -l 32;	0.818	
33	echo -n "g#7AE9@?" | argon2 "BsLmlEe8_OY>A.l/939m" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
34	echo -n "g#7AE9@?" | argon2 "HqYr,dqzvhv/|8Vu+>2t" -i -k 400000 -t 2 -p 2 -l 32;	0.825	
35	echo -n "g#7AE9@?" | argon2 "yM0Re_29t3NT?y),I;&h" -i -k 400000 -t 2 -p 2 -l 32;	0.829	0.823
36	echo -n "g#7AE9@?" | argon2 "quE*dV+oLof=M,.yTj<j89" -i -k 400000 -t 2 -p 2 -l 32;	0.808	
37	echo -n "g#7AE9@?" | argon2 "fy2GT~%x6+{~XY?7:)2K#}" -i -k 400000 -t 2 -p 2 -l 32;	0.804	
38	echo -n "g#7AE9@?" | argon2 "tQKOLy)|6YG81d5(I#QxHY" -i -k 400000 -t 2 -p 2 -l 32;	0.812	
39	echo -n "g#7AE9@?" | argon2 "Ggu)x8{6S/P_*m5D(snUxx" -i -k 400000 -t 2 -p 2 -l 32;	0.817	
40	echo -n "g#7AE9@?" | argon2 "N(C~c#zr&J7C~i9qoGdixQ" -i -k 400000 -t 2 -p 2 -l 32;	0.954	0.839
41	echo -n "g#7AE9@?" | argon2 "D^^3#3l_Imo0a6>Ib?[V_]E$" -i -k 400000 -t 2 -p 2 -l 32;	0.8	
42	echo -n "g#7AE9@?" | argon2 "Q2Y+s]ts$5{BUgqCrq.I6[Kb" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
43	echo -n "g#7AE9@?" | argon2 "N6:|BUtrpp>vx|=3L>{P5XM&" -i -k 400000 -t 2 -p 2 -l 32;	0.813	
44	echo -n "g#7AE9@?" | argon2 "X{u1-5kd&mRV}%))1m0&M+B+" -i -k 400000 -t 2 -p 2 -l 32;	0.811	
45	echo -n "g#7AE9@?" | argon2 "iB(hVzYT*<CEy*9{SLY0HPr9" -i -k 400000 -t 2 -p 2 -l 32;	0.818	0.8136
46	echo -n "g#7AE9@?" | argon2 "T7kAtK_-nH]tDP33x8taxx5+0A" -i -k 400000 -t 2 -p 2 -l 32;	0.838	
47	echo -n "g#7AE9@?" | argon2 "UqDXd(t6Ym%y[sT8946kV4n7td" -i -k 400000 -t 2 -p 2 -l 32;	0.809	
48	echo -n "g#7AE9@?" | argon2 "pqxirptt_6-X#[p6q/V3E:*RBA" -i -k 400000 -t 2 -p 2 -l 32;	0.827	
49	echo -n "g#7AE9@?" | argon2 "gGvRivJNky7)gsu:Bhx8j7R.<G" -i -k 400000 -t 2 -p 2 -l 32;	0.807	
50	echo -n "g#7AE9@?" | argon2 "DUBqTVXBIT)j6}25gU&kFQ^?@v" -i -k 400000 -t 2 -p 2 -l 32;	0.816	0.8194
51	echo -n "g#7AE9@?" | argon2 "VnMKAF6|{gys2DJ*{2gU|zP>5fA." -i -k 400000 -t 2 -p 2 -l 32;	0.833	
52	echo -n "g#7AE9@?" | argon2 "oJ*xBd*Vt7|@*=w|6xZ5xttjCYx." -i -k 400000 -t 2 -p 2 -l 32;	0.823	
53	echo -n "g#7AE9@?" | argon2 "prPsog6]3Vvc>uVjG/XNxf<W-/UN" -i -k 400000 -t 2 -p 2 -l 32;	0.818	
54	echo -n "g#7AE9@?" | argon2 "KzhBix~9a2c>|0zoQ5<a-_ir?-kK" -i -k 400000 -t 2 -p 2 -l 32;	0.825	
55	echo -n "g#7AE9@?" | argon2 "Q[wo:o$JO)KbzHrT>*6u8wCPH?GS" -i -k 400000 -t 2 -p 2 -l 32;	0.805	0.8208
56	echo -n "g#7AE9@?" | argon2 "dW0ZREhNfi8eS^+xjK64(M:Eg}=USC" -i -k 400000 -t 2 -p 2 -l 32;	0.81	
57	echo -n "g#7AE9@?" | argon2 "JdAPXkf0#.DKn%~bg.waz[1M?T7nuH" -i -k 400000 -t 2 -p 2 -l 32;	0.829	
58	echo -n "g#7AE9@?" | argon2 "Btb^j@NX]2$oBtum?a|V1q5dn}.<Ix" -i -k 400000 -t 2 -p 2 -l 32;	0.817	
59	echo -n "g#7AE9@?" | argon2 "QTn8sW)&{ZN6,.]W_m&]&LC=K-3957" -i -k 400000 -t 2 -p 2 -l 32;	0.816	

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
60	echo -n "g#7AE9@?" | argon2 "sum6xqaINlDucyx&pRbI6qB?3(W9B6" -i -k 400000 -t 2 -p 2 -l 32;	0.815	0.8174
61	echo -n "g#7AE9@?" | argon2 "RoqJ.N%=xpxxqr60As}9BSk{4qspd-3D" -i -k 400000 -t 2 -p 2 -l 32;	0.816	
62	echo -n "g#7AE9@?" | argon2 "Ir4q#?Q.$bOtJ/M_nTF[Ac+HUA_U/9t*" -i -k 400000 -t 2 -p 2 -l 32;	0.824	
63	echo -n "g#7AE9@?" | argon2 "Ff6]hjxa-Ef]uEoyvCkF~jFD+ytQA5VX" -i -k 400000 -t 2 -p 2 -l 32;	0.826	
64	echo -n "g#7AE9@?" | argon2 "Pj&kQZcxU1?E2/Dfd_tLm3gFGK*T3/br" -i -k 400000 -t 2 -p 2 -l 32;	0.83	
65	echo -n "g#7AE9@?" | argon2 "QJ5Zx&.Ik8Jj%-g>/iWY8I8k3=rDkXl#" -i -k 400000 -t 2 -p 2 -l 32;	0.815	0.8222
66	echo -n "g#7AE9@?" | argon2 "ZH4J[>{19rD$Cxppg(BmL&R)kGHsq1t)hw" -i -k 400000 -t 2 -p 2 -l 32;	0.819	
67	echo -n "g#7AE9@?" | argon2 "j?lN&v55WapFq.]iSYmoxir<8XY}W4KFpp" -i -k 400000 -t 2 -p 2 -l 32;	0.833	
68	echo -n "g#7AE9@?" | argon2 "e0j[UMzoU$OXOErn=S<ll($D|>ILF7l=k%" -i -k 400000 -t 2 -p 2 -l 32;	0.808	
69	echo -n "g#7AE9@?" | argon2 "WgQaYQqT9CAJdR7rW9cnZd49vK4sD4Yv5T" -i -k 400000 -t 2 -p 2 -l 32;	0.837	
70	echo -n "g#7AE9@?" | argon2 "D6Vk.[(OMe-X(xxwc&Nq0QSbD7ci5Z5jXx" -i -k 400000 -t 2 -p 2 -l 32;	0.849	0.8292
71	echo -n "g#7AE9@?" | argon2 "MUL.9Dw79E3FOQiF6f1$Q#l]gy0S7r|R{HTH" -i -k 400000 -t 2 -p 2 -l 32;	0.834	
72	echo -n "g#7AE9@?" | argon2 "k1qM/]cEzGqXox9H3~]W~G</Xq?2-MUTx%&1" -i -k 400000 -t 2 -p 2 -l 32;	0.812	
73	echo -n "g#7AE9@?" | argon2 "e7W~@D%kqm|vqy3Sim[tVL=QBKjHOq]~>&Hy" -i -k 400000 -t 2 -p 2 -l 32;	0.807	
74	echo -n "g#7AE9@?" | argon2 "t%/_{*J2yC>r_-f(ZT(Q+:ZqzxF+8{AzXN$6" -i -k 400000 -t 2 -p 2 -l 32;	0.814	
75	echo -n "g#7AE9@?" | argon2 "a0:vh&|G3.#BR{Hk9KkrTJbWif,GYP)i8{K#" -i -k 400000 -t 2 -p 2 -l 32;	0.826	0.8186
76	echo -n "g#7AE9@?" | argon2 "I-3tXy/]9z+[Rl4{EIr>Ciw>sA)hB3~Zze:r6%" -i -k 400000 -t 2 -p 2 -l 32;	0.812	
77	echo -n "g#7AE9@?" | argon2 "P_1gtaQN.|k]xVea|JT9[}aG7Li)Yk:yt.Hxp9" -i -k 400000 -t 2 -p 2 -l 32;	0.812	
78	echo -n "g#7AE9@?" | argon2 "u~G&{hAatF4Q)AvbBSw1ya>Gu3mxp#R^/ZY>p4" -i -k 400000 -t 2 -p 2 -l 32;	0.846	
79	echo -n "g#7AE9@?" | argon2 "CCrq859$>~s.]4SR0NMf5m064T)@Dc%nxVx2k." -i -k 400000 -t 2 -p 2 -l 32;	0.821	
80	echo -n "g#7AE9@?" | argon2 "DrWZ/BGkx7DWTU}*KiTk.{f>6kiPxrBK<n4Wut" -i -k 400000 -t 2 -p 2 -l 32;	0.821	0.8224
3.หาพารามิเตอร์ "Memory size (k)" ที่เร็วที่สุด แบบที่ 1			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.008
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 8000 -t 2 -p 2 -l 32;		0.018
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 12000 -t 2 -p 2 -l 32;		0.028

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 16000 -t 2 -p 2 -l 32;		0.034
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 20000 -t 2 -p 2 -l 32;		0.045
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 24000 -t 2 -p 2 -l 32;		0.054
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 28000 -t 2 -p 2 -l 32;		0.06
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 32000 -t 2 -p 2 -l 32;		0.077
9	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 36000 -t 2 -p 2 -l 32;		0.079
10	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 40000 -t 2 -p 2 -l 32;		0.095
11	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 44000 -t 2 -p 2 -l 32;		0.093
12	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 48000 -t 2 -p 2 -l 32;		0.107
13	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 52000 -t 2 -p 2 -l 32;		0.128
14	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 56000 -t 2 -p 2 -l 32;		0.124
15	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 60000 -t 2 -p 2 -l 32;		0.131
16	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 64000 -t 2 -p 2 -l 32;		0.141
17	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 68000 -t 2 -p 2 -l 32;		0.154
18	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 72000 -t 2 -p 2 -l 32;		0.158
19	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 76000 -t 2 -p 2 -l 32;		0.169
20	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 80000 -t 2 -p 2 -l 32;		0.183
21	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 84000 -t 2 -p 2 -l 32;		0.181
22	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 88000 -t 2 -p 2 -l 32;		0.195
23	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 92000 -t 2 -p 2 -l 32;		0.204
24	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 96000 -t 2 -p 2 -l 32;		0.221
25	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 100000 -t 2 -p 2 -l 32;		0.217
26	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 104000 -t 2 -p 2 -l 32;		0.225
27	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 108000 -t 2 -p 2 -l 32;		0.239
28	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 112000 -t 2 -p 2 -l 32;		0.241
29	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 116000 -t 2 -p 2 -l 32;		0.255
30	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 120000 -t 2 -p 2 -l 32;		0.258
31	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 124000 -t 2 -p 2 -l 32;		0.267
32	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 128000 -t 2 -p 2 -l 32;		0.298
33	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 132000 -t 2 -p 2 -l 32;		0.295
34	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 136000 -t 2 -p 2 -l 32;		0.303
35	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 140000 -t 2 -p 2 -l 32;		0.307
36	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 144000 -t 2 -p 2 -l 32;		0.325
37	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 148000 -t 2 -p 2 -l 32;		0.329
38	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 152000 -t 2 -p 2 -l 32;		0.333
39	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 156000 -t 2 -p 2 -l 32;		0.352
40	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 160000 -t 2 -p 2 -l 32;		0.352
41	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 164000 -t 2 -p 2 -l 32;		0.372
42	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 168000 -t 2 -p 2 -l 32;		0.38
43	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 172000 -t 2 -p 2 -l 32;		0.393
44	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 176000 -t 2 -p 2 -l 32;		0.393
45	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 180000 -t 2 -p 2 -l 32;		0.408
46	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 184000 -t 2 -p 2 -l 32;		0.408
47	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 188000 -t 2 -p 2 -l 32;		0.437
48	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 192000 -t 2 -p 2 -l 32;		0.436

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
49	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 196000 -t 2 -p 2 -l 32;		0.437
50	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 200000 -t 2 -p 2 -l 32;		0.449
51	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 204000 -t 2 -p 2 -l 32;		0.453
52	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 208000 -t 2 -p 2 -l 32;		0.456
53	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 212000 -t 2 -p 2 -l 32;		0.47
54	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 216000 -t 2 -p 2 -l 32;		0.486
55	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 220000 -t 2 -p 2 -l 32;		0.484
56	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 224000 -t 2 -p 2 -l 32;		0.493
57	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 228000 -t 2 -p 2 -l 32;		0.517
58	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 232000 -t 2 -p 2 -l 32;		0.518
59	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 236000 -t 2 -p 2 -l 32;		0.529
60	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 240000 -t 2 -p 2 -l 32;		0.527
61	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 244000 -t 2 -p 2 -l 32;		0.556
62	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 248000 -t 2 -p 2 -l 32;		0.552
63	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 252000 -t 2 -p 2 -l 32;		0.567
64	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 256000 -t 2 -p 2 -l 32;		0.565
65	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 260000 -t 2 -p 2 -l 32;		0.581
66	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 264000 -t 2 -p 2 -l 32;		0.609
67	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 268000 -t 2 -p 2 -l 32;		0.604
68	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 272000 -t 2 -p 2 -l 32;		0.603
69	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 276000 -t 2 -p 2 -l 32;		0.624
70	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 280000 -t 2 -p 2 -l 32;		0.646
71	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 284000 -t 2 -p 2 -l 32;		0.641
72	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 288000 -t 2 -p 2 -l 32;		0.641
73	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 292000 -t 2 -p 2 -l 32;		0.656
74	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 296000 -t 2 -p 2 -l 32;		0.671
75	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 300000 -t 2 -p 2 -l 32;		0.664
76	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 304000 -t 2 -p 2 -l 32;		0.702
77	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 308000 -t 2 -p 2 -l 32;		0.678
78	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 312000 -t 2 -p 2 -l 32;		0.687
79	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 316000 -t 2 -p 2 -l 32;		0.702
80	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 400000 -t 2 -p 2 -l 32;		0.923
4.หาพารามิเตอร์ "Memory size (k)" ที่เร็วที่สุดแบที่ 2			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.008
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 8000 -t 2 -p 2 -l 32;		0.019
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 16000 -t 2 -p 2 -l 32;		0.04
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 32000 -t 2 -p 2 -l 32;		0.07
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 64000 -t 2 -p 2 -l 32;		0.142
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 128000 -t 2 -p 2 -l 32;		0.297
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 256000 -t 2 -p 2 -l 32;		0.57
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 512000 -t 2 -p 2 -l 32;		1.175
9	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 1024000 -t 2 -p 2 -l 32;		2.375
10	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 2048000 -t 2 -p 2 -l 32;		4.791
11	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4096000 -t 2 -p 2 -l 32;		9.638
5.หาพารามิเตอร์ "Iteration number (t)" ที่เร็วที่สุด (+6)			

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.008
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 8 -p 2 -l 32;		0.027
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 14 -p 2 -l 32;		0.04
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 20 -p 2 -l 32;		0.056
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 26 -p 2 -l 32;		0.077
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 32 -p 2 -l 32;		0.099
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 38 -p 2 -l 32;		0.102
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 44 -p 2 -l 32;		0.129
9	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 50 -p 2 -l 32;		0.133
10	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 56 -p 2 -l 32;		0.163
11	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 62 -p 2 -l 32;		0.195
12	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 68 -p 2 -l 32;		0.213
13	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 74 -p 2 -l 32;		0.227
14	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 80 -p 2 -l 32;		0.223
15	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 86 -p 2 -l 32;		0.248
16	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 92 -p 2 -l 32;		0.252
17	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 98 -p 2 -l 32;		0.267
18	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 104 -p 2 -l 32;		0.269
19	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 110 -p 2 -l 32;		0.323
20	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 116 -p 2 -l 32;		0.363
21	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 122 -p 2 -l 32;		0.299
22	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 128 -p 2 -l 32;		0.36
23	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 134 -p 2 -l 32;		0.387
24	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 140 -p 2 -l 32;		0.406
25	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 146 -p 2 -l 32;		0.439
26	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 152 -p 2 -l 32;		0.426
27	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 158 -p 2 -l 32;		0.403
28	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 164 -p 2 -l 32;		0.425
29	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 170 -p 2 -l 32;		0.429
30	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 176 -p 2 -l 32;		0.458
31	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 182 -p 2 -l 32;		0.488
32	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 188 -p 2 -l 32;		0.507
33	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 194 -p 2 -l 32;		0.501
34	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 200 -p 2 -l 32;		0.498
35	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 206 -p 2 -l 32;		0.537
36	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 212 -p 2 -l 32;		0.551
37	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 218 -p 2 -l 32;		0.557
38	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 224 -p 2 -l 32;		0.572
39	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 230 -p 2 -l 32;		0.587
40	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 236 -p 2 -l 32;		0.579
41	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 242 -p 2 -l 32;		0.69
42	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 248 -p 2 -l 32;		0.63
43	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 254 -p 2 -l 32;		0.662
6.หาพารามิเตอร์ "Iteration number (t)" ที่เร็วที่สุด (^2)			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.007

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 4 -p 2 -l 32;		0.013
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 8 -p 2 -l 32;		0.023
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 16 -p 2 -l 32;		0.046
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 32 -p 2 -l 32;		0.082
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 64 -p 2 -l 32;		0.161
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 128 -p 2 -l 32;		0.333
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 256 -p 2 -l 32;		0.639
7.หาพารามิเตอร์ Parallelism (p)" ที่เร็วที่สุด			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.009
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 4 -l 32;		0.011
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 8 -l 32;		0.011
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 16 -l 32;		0.011
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 32 -l 32;		0.017
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 64 -l 32;		0.011
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 128 -l 32;		0.011
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 256 -l 32;		0.015
8.หาพารามิเตอร์ "Parallelism (p)" ที่เร็วที่สุด (+6)			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32;		0.009
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 8 -l 32;		0.008
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 14 -l 32;		0.009
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 20 -l 32;		0.012
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 26 -l 32;		0.011
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 32 -l 32;		0.009
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 38 -l 32;		0.01
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 44 -l 32;		0.01
9	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 50 -l 32;		0.009
10	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 56 -l 32;		0.011
11	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 62 -l 32;		0.009
12	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 68 -l 32;		0.01
13	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 74 -l 32;		0.01
14	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 80 -l 32;		0.013
15	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 86 -l 32;		0.012
16	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 92 -l 32;		0.011
17	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 98 -l 32;		0.011
18	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 104 -l 32;		0.011
19	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 110 -l 32;		0.014
20	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 116 -l 32;		0.013
21	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 122 -l 32;		0.012
22	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 128 -l 32;		0.01
23	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 134 -l 32;		0.012
24	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 140 -l 32;		0.014
25	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 146 -l 32;		0.013
26	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 152 -l 32;		0.012
27	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 158 -l 32;		0.016
28	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 164 -l 32;		0.013

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
29	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 170 -l 32;		0.013
30	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 176 -l 32;		0.013
31	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 182 -l 32;		0.016
32	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 188 -l 32;		0.017
33	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 194 -l 32;		0.016
34	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 200 -l 32;		0.018
35	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 206 -l 32;		0.015
36	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 212 -l 32;		0.015
37	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 218 -l 32;		0.015
38	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 224 -l 32;		0.014
39	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 230 -l 32;		0.015
40	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 236 -l 32;		0.018
41	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 242 -l 32;		0.019
42	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 248 -l 32;		0.018
43	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 254 -l 32;		0.013
9.หาพารามิเตอร์ "Tag length (l)" ที่เร็วที่สุด (+4)			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32		0.008
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 36		0.01
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 40		0.008
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 44		0.01
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 48		0.007
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 52		0.011
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 56		0.008
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 60		0.008
9	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 64		0.01
10	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 68		0.009
11	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 72		0.011
12	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 76		0.008
13	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 80		0.008
14	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 84		0.008
15	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 88		0.01
16	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 92		0.01
17	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 96		0.01
18	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 100		0.008
19	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 104		0.008
20	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 108		0.009
21	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 112		0.009
22	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 116		0.007
23	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 120		0.009
24	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 124		0.01
25	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 128		0.008
26	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 132		0.008
27	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 136		0.011
28	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 140		0.008
29	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 144		0.008

ลำดับ	Code	เวลา/วินาที	เวลาเฉลี่ย/วินาที
30	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 148		0.01
31	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 152		0.009
32	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 156		0.009
33	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 160		0.011
34	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 164		0.011
35	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 168		0.009
36	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 172		0.011
37	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 176		0.009
38	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 180		0.009
39	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 184		0.007
40	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 188		0.008
41	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 192		0.008
42	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 196		0.009
43	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 200		0.008
44	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 204		0.008
45	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 208		0.011
46	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 212		0.008
47	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 216		0.009
48	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 220		0.011
49	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 224		0.008
50	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 228		0.009
51	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 232		0.01
52	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 236		0.01
53	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 240		0.009
54	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 244		0.009
55	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 248		0.007
56	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 252		0.009
57	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 256		0.011
10.หาพารามิเตอร์ "Tag length (l)" ที่เร็วที่สุด (^2)			
1	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 32		0.007
2	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 64		0.009
3	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 128		0.008
4	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 256		0.008
5	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 512		0.012
6	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 1024		0.012
7	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 2048		0.009
8	echo -n "g#7AE9@?" | argon2 "O6JV<HdD" -i -k 4000 -t 2 -p 2 -l 4096		0.009

ภาคผนวก ข.

ผลทดลองของ Avalanche effect

<table>
<tr><th>Change Password</th><th>ค่าบิตของแฮช</th><th>%</th></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.A" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>01100101001110010110011000110010011001010011011100110101011 00
1100011011100110011001100010011001000111001011000100011010100
110001001110000011000100110010001100000011100100110001001 1001
100110000000110000011000110011011000110011001100010011100 10011
0100001100100011010101100101011001100011011000110010001100100
0110001011001010011100100110001001100100110010101100011001101
1000110100001110010011001101100001011000110011000100110101001
1010100110100011000010110001001100110001110010011100001100101
011001010011100101100011</td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>0011100100110100001100110011010100111001011000010011010101100
1100011100101100101001100000011100100110110001101100011011100
1101000011001001100100001110000011001000110110011000110110011
0001100000001110000011100100110111001101000011010101100110001 1
0001001100010011000100110110011001000110001001100010011001000
0110001001100000110010100110111011000010110000101100001011001
0000110111011000110110010000110001001101100110001100111001001
1010100111000001100110110001001100010011001000110001001100001
0110001100110010001101 10</td><td>32.03%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.C" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>0011000100110000001100000011000100110000001100010011010000110
0010011011100111001011001100011001001100100011000010110011000
1100010011000000111000011000110011010000110100001101000110001
1011000110011001000110111001110000011010001100010001101000110
0100011000110011001000110000001100110110001100110011001101110
0110001011000010011001100110001011001100110010100110011001110
0100110101011000010011100100111001011000110110011001100010011
0001100110111001110000110001100111000001101100011100100110111
0011000100110101001 10010</td><td>31.64%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.E" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>0011011001100110011001000011011101100100001100000110001001100
1010011100000110110011001010110001101100110001100010011001000
1100100011001001100010011000010011010000110011001101000011000
0011000010011010000111000001100010011100000110010011000010011
0000001101110011010100111001001100000011100101100011001100010
1100010001100100011011001100011001101010110000100111000001100
1000110100001101010011000000110101001101100110010100110111001
1100101100010011000110011001101100100011001100011000100110001
011001000110010101100101</td><td>34.77%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.I" | argon2</td><td>0110001000110100011001010011010100110000001101100011011000111
0010011000101100001011000110011010001100100001100000011010101
1000100011000100110100001101100011000101100100011000100110001</td><td>34.38%</td></tr>
</table>

<table>
<tr><th>Change Password</th><th>ค่าบิตของแฮช</th><th>%</th></tr>
<tr><td>"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>10011011101100110011000100110000101100101001100110011010100 11
000001100010001101010110010000111000001110000011100001100 1000
011100000110100001110000011011000111000011000010011000001 1001
100011010001100010011000100011100100110110011001010110010 0011
001100011000001100110001110000011010101100010001100110011 0000
0011010000110101001101 11</td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.Q" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>001110000011010100110000011000010110011000110011001101100 0111
000011001010110010100110101011000110110001000110110001100 1100
111001001101010110001101100101011001000110000100110001011 0001
100110011001100100110001000110001001101100011011100110010 0011
010001100001001100000011010101100001011001100011000100110 1010
011001100110110001100000011011101100100001100110011100101 1000
010011011000110111001100000011000100110010001101100110010 0001
110010011010100110101001110000011001101100001011000100011 0010
0011000000111000011001 00</td><td>48.63%</td></tr>
<tr><td></td><td>เฉลี่ย</td><td>36.29%</td></tr>
<tr><td>Change Salt</td><td></td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.A" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>011001010011100101100110001100100110010100110111001101010 1100
110001101110011001100110001001100100011100101100010001101 0100
110001001110000011000100110010001100000011100100110001001 1001
100110000001100000110001100110110001100110011000100111001 0011
010000110010001101010110010101100110001101100011001000110 0100
011000101100101001110010011000100110010011001010110001100 1101
100011010000111001001100110110000101100011001100010011010 1001
101010011010001100001011000100110011000111001001110000110 0101
0110010100111001011000 11</td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 4000 -t 2 -p 8 -l
32;</td><td>011001100011100100110111001100010011000000110110001110000 1100
110001110000011100100110101011001000011011000110010001100 1000
111000011001000011000101100110011001100011010100111001001 1000
100110101011001000110010001100110001110000011000000110001 0011
000000111000011000010110011000110101001100100110010000110 1110
110010100110101001110000011010000110001011001010011011001 1001
010011100101100010001110010011001100110010001100100011000 1001
100100011011001100011001100010110001001100101011000110011 0101
0110011001100001001101 10</td><td>29.88%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R</td><td>011000010011001001100101001110010011010001100110001101100 1100
011011001100011011001100100001101000110001101100010001100 1001
100001001110000110010101100100011000100110011000110011011 0001
101100011001100100011100100110101011001010011000100110010 0011</td><td>32.03%</td></tr>
</table>

Change Password	**ค่าบิตของแฮช**	**%**
.<A" -i -k 4000 -t 2 -p 8 -l 32;	0001011000110110000100111001001100110011011100111000001100110 1100101001100100011001100110110001101000110011001100011011001 0100110001011001000011000000110000001100100110011001100010001 1011000110001011000010110001101100101001100100011100100110100 00111001001100100110000 1	
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.@" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<I" -i -k 4000 -t 2 -p 8 -l 32;	0110010001100010001100100110011000110110001101110110000100111 0010110001000110111001101000011001000110011001100100011100100 1101010110010100110000001110000011010000110111011001100110001 1001101110110010100111001001101010110001000110110011000100011 0110001100100110001101100100001100100011000101100011001101100 0110101011000010110001000110010011000100011010100110101001101 0101100100001110000011010101100010011000100110011001100010001 1100100110100001100010011010100110111001100010110001001100110 01100010001100010110001 0	31.05%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.@" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<Q" -i -k 4000 -t 2 -p 8 -l 32;	0011011001100100001100000011011100110010011001000011000100110 1110011001000111000001100010110011000110100001100100011010100 1101110011100100111001001101000011010000110110001101000011000 0001100110011000101100010011000010110010100110110001100110011 0000001100000011100101100011001101100110001101100101001101100 1100101001100000011011000111001001110000110001000111001011001 0000110111011000010110010100111000001101110011100100110110001 1010100110100001110010011100100110111001110010011000100111001 00110100001101010110001 0	30.86%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.@" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<a" -i -k 4000 -t 2 -p 8 -l 32;	0110001000110111011001000110001000110110001110000011010101100 1010110001101100101011000100011011001100001001100000011001000 1101100110001100110100011001100011011101100001001101000110010 0001100010011100001100011011000010011010101100010001101110110 0110001110010011011000110100001101010011000000111001001110010 0110010001110010011001101100101011000010011000000111001001100 0001100011011001100011010100110010001101010110000100110110011 0010100110010011001100011001001100001011001000011001000110101 00110111011000110011001 0	36.33%
	เฉลี่ย	32.03%
Memory size		
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R	0110010100111001011001100011001001100101001101110011010101100 1100011011100110011001100010011001000111001011000100011010100 1100010011100000110001001100100011000000111001001100010011001 1001100000011000001100011001101100011001100110001001110010011 0100001100100011010101100101011001100011011000110010001100100	

<table>
<tr><th>Change Password</th><th>ค่าบิตของแฮช</th><th>%</th></tr>
<tr><td>.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>0110001011001010011100100110001001100100110010101100011001101
1000110100001110010011001101100001011000110011000100110101001
1010100110100011000010110001001100110001110010011100001100101
0110010100111001011000 11</td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 3999 -t 2 -p 8 -l
32;</td><td>0011010000110111011001000011100000110010011000010011010000111
0010110010000110010001100100110000100110111001110010011011001
1001100011011001100110001110010110010101100010001101100110001
1001101110110010001100101001100010011000100110101001100100011
0010011001100110000101100001001100000011001100110100001100110
0111001011000010011010100110101001101100110001000110100011000
1000110110001100110110000101100001011000110011001001100001011
0010100110100011000100110000100110010001101010011011100110011
001101010110011001100110</td><td>32.62%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 3998 -t 2 -p 8 -l
32;</td><td>0011001101100010011001010011011000110110001110000011011101100
1000110010100110000011000010011000101100101001100100011011101
1000110011000100110011001101110011001000110010011000110011001
1011000010110001000110001011001010011001000110111001101100011
1001011000010110000101100101001100010011001001100110001100110
0110000001100100011000001100101001101100110010101100110001100
0000111000011001100011011001100100011001000110001100110110001
1011000110011001100000011100001100100011001100110010100110110
001110010011001000110111</td><td>32.62%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 3997 -t 2 -p 8 -l
32;</td><td>0011000000111000001101010011100001100001011001100011100101100
1100011011100110011011001100011010000110001011000010011100000
1100010011011000111000001101000110010100111001001101100011000
0001101010011000100110110001100100110011000111000001101100011
0111001110000011000001100101011000110011001000110101001110010
0110110001110000110010001100011001100100011001000110010001110
0101100110001100000011001101100011001110000011011101100010001
1011000110100001100010011000001100001011001000011100000110111
001100110011010000110101</td><td>30.08%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 3996 -t 2 -p 8 -l
32;</td><td>0110011000110001011000110011011001100010011001100011010001100
0010110010000111001001101010011001000110101001110010011100100
1101100011000000110001011001100110010100110100001100010011011
1011000010110000101100100011001010110001101100101001101010011
0110011001010110001000110000001101000110001000110110001110010
1100100001101010011001001100100011000100011011001100101001100
1100110000011001100110001000110000001101010110010001100110001</td><td>32.32%</td></tr>
</table>

Change Password	ค่าบิตของแฮช	%
	1010101100010001110000110001101100100001101100110000101100011 011001010110010001100010	
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.@" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<F" -i -k 3995 -t 2 -p 8 -l 32;	0011001000110110011001100110001100110110001101000110010000110 0100110011000110101001101110011001001100011001110000011011000 1110010110001101100101011000110110010001100010001101000011100 0011000010110010101100100001101000011100100110101001100000011 0001001101010011000100110100001101010011001001100100011001010 0111001011001010011010101100010011001100110010000110000011000 1100110011001100000110001001100010001100010011100000110100001 1001100110010001101110110010000110110011000100011100000110001 011000110110011000110100	33.79%
	เฉลี่ย	32.30%
Iteration number (t)		
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 2 -p 8 -l 32;	0110010100111001011001100011001001100101001101110011010101100 1100011011100110011001100010011001000111001011000100011010100 1100010011100000110001001100100011000000111001001100010011001 1001100000011000001100011001101100011001100110001001110010011 0100001100100011010101100101011001100011011000110010001100100 0110001011001010011100100110001001100100110010101100011001101 1000110100001110010011001101100001011000110011000100110101001 1010100110100011000010110001001100110001110010011100001100101 011001010011100101100011	
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 3 -p 8 -l 32;	0011001101100101001110010011010101100110011000010011011100110 0110011000000110000001100100110010000110111011000100011011001 1001000011001101100100001100010011010001100110001100100110010 1001100010011100100110101001101110011011000110100001100010110 0110001100010110011001100010011001000110000100110000001101110 0110111011000110011100000110001001100100011100000110101001100 1000110010001100110011100100110111011000100011011101100101011 0000101100001001110010110011001100110001101010011011100110110 001100110011010000110000	32.32%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 4 -p 8 -l 32;	0110001001100001011001000011011101100010011001100011010001100 1000011010100110100001100010011100101100001001100000011000000 1101010110000100111000001100010011001101100010001100100011000 0001100010011000000110000001100010110010100110000001101100110 0010001100110011000000110010001100100011001000110110001101110 0111001001100110011100000110011011001010110001001100100011000 0101100100011000010011000000110111001100100011100001100011001	31.64%

Change Password	ค่าบิตของแฮช	%
	1000000110010011001000011011100110001001100110011011101100010 0110000101100100011000 11	
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 5 -p 8 -l 32;	0011011000111001001100100110000101100101001101000110010000110 1110011001000110100001101000011010001100101001100010011000001 1000110011100001100110001110000011010101100110001100000011001 0001100110011011101100010001101000011011000110101001100010011 0011011000110110010001100101011001000110001000110000001100010 0110110001101010011000000110101001110000110011000111000001101 1000111000011000100011010101100110001101100011000100110110001 1000001100100011000110011011101100101011001000011100101100001 001101110110001101100101	28.71%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 6 -p 8 -l 32;	0011010101100110001101000011100001100001001100100011000000110 0010011001001100100001100000011011100110110001100000011010001 1001010011011100110011001101110011011100110110001100010011011 1001100110011010100110101011001000011010101100101001100100110 0101001100100110010000110100011000010110010100110111001101000 0110010001100110110001000110011011000100011001101100101001101 1100110101001100010011100100110001001101010110011001100100011 0001100110010011000100110001001100010001100110011000000111001 011001010011000001100001	30.47%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 7 -p 8 -l 32;	0011100000110001001100110011000000110111011001000110001101100 1000011011000110011011001000011011000110000001101110011100101 1001000011100000110110001100010011011000110001011001100110011 0011001100110011000111000011001100011011000110000001101100011 1001011001010011000000110101001100010011001000110001001100000 0110000011000110011000000110111001110000011010000110010011000 1101100011011000010011001100110101001101100110011000110001011 0001001100100001100010011011000110111001100000011000101100010 001101000110001101100110	33.59%
	เฉลี่ย	31.37%
Parallelism (p)		
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 2 -p 8 -l 32;	0110010100111001011001100011001001100101001101110011010101100 1100011011100110011001100010011001000111001011000100011010100 1100010011100000110001001100100011000000111001001100010011001 1001100000011000001100011001101100011001100110001001110010011 0100001100100011010101100101011001100011011000110010001100100 0110001011001010011100100110001001100100110010101100011001101 1000110100001110010011001101100001011000110011000100110101001	

Change Password	ค่าบิตของแฮช	%
	1010100110100011000010110001001100110001110010011100001100101011001010011100101100011	
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000-t 2 -p 9 -l 32;	0011011100111000001100110011001101100001001101000011100100110000001110000001100100011010100111001001101000011100100110100011000010011001100110000001100010110001000111001001110000011000000110011011000100011100101100101011000100011001000110001001100000011001001100010001101100110011000110110001100110011001001100110001110010011000001100110001110000011001000110110011000010011100001100100001101110110000100110111001110000001100010011001100110110001101010110010100110010011000100011011000110111011000100011010100111001	31.05%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 2 -p 10 -l 32;	0011010000110111001101010110010000110010001100110011000000011000100110011011001100011100000110101011000100110011001100001001110000001101110011000001100110011000110011010100110001011001100011011100110010011001100011001100110010001100100110011000110011001100000011010000110111001101100110001001100101011000100011001101100100001101110011011101100101001101000110001100110011001101100101001110000011010101100011011000100011011000110101001110000110000100110101011001000011000100110001001100100011011100110111011000110011001100110011	32.23%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 2 -p 11 -l 32;	001100110011100000111001001110000011000001100011001101000011001001100010011001100110010000111000001101000011100000110110001101100011010000110011001101000011011001100010001110000110001100110111001101010110010101100100001100110011011100110000001100101001101000011000100110100011001000110010000110110001101100011001000111001001100000011011000111000001100000011011001100101011000010011011000110111001100110011100001100101001101100110010100110111001100110011011000110001001100000011011001100100001101110011100001100100	33.20%
echo -n "r=IlH<7+ej[fv7u,A&Ob%H 3x7J.A" | argon2 "gGvRivJNky7)gsu:Bhx8j7R .<G" -i -k 4000 -t 2 -p 12 -l 32;	0110001100111000011001100011100100110011001101000011011000110010001100100110000100110111001100100011001101100011001110010110010100110110001100010011100000111000011001100011011101100110001100100011000000110010101100001001110010011100000110110001100100011011000110011001100110000011000010011010101100100011000001001101110110010001100101001101100011001100110001001100000011001001100101001101010011010100110011011000110011001100110011011100110111011001000011011001100010011001100110010001100010001110010011000100110000	31.84%

<table>
<tr><th>Change Password</th><th>ค่าบิตของแฮช</th><th>%</th></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.A" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 13 -l
32;</td><td>01100101001101110011011001100110011001010011001000110010011000
0110110001101100110001101110110000101100101001101010110001000
1101110011000100110000001101110011000101100011011001000110000
1001100010011010100110000011000100011000100110010011000110011
0001001101100011000000110100011001100011000000110101001101110
1100101011000110011001100111000001101010011000001100010001101
1000110110011001000011001100111001001101000011000001100110001
1010001100011011001100011100000111000001101110011001101100101
0011100000011010001100101</td><td>31.84%</td></tr>
<tr><td></td><td>เฉลี่ย</td><td>32.03%</td></tr>
<tr><td>All Paramiter</td><td></td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.A" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<G" -i -k 4000 -t 2 -p 8 -l
32;</td><td>0110010100111001011001100011001001100101001101110011010101100
1100011011100110011001100010011001000111001011000100011010100
1100010011100000110001001100100011000000111001001100010011001
1001100000011000001100011001101100011001100110001001110010011
0100001100100011010101100101011001100011011000110010001100100
0110001011001010011100100110001001100100110010101100011001101
1000110100001110010011001101100001011000110011000100110101001
1010100110100011000010110001001100110001110010011100001100101
0110010100111001011000111</td><td></td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.@" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<F" -i -k 3999-t 3 -p 3 -l
32;</td><td>0011001100110111001101110110011000110111001101010011100100111
0000011001100110100001100000110000100110001001101100011001100
1100000011010101100011011001100110011000110100011001010110011
0001100110011010001100101001100000011001001100001011000110011
0001011000010011001100110000001101100011010000110001011001010
1100011001101010011011101100100001101100011100100110100001110
0001100110001101000011001100110010001101000011001101100100011
0010000111000001100110110010100110100001110010110001000110111
0110001101100010001110 01</td><td>34.18%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.C" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<A" -i -k 3998 -t 3 -p 3 -l
32;</td><td>0011000100110000001100000011001101100100001101010011000000110
0110011000101100001001101010011000001100110001101110011000000
1110000110000100110000011000110011010101100011001101110011001
0001101100011100000111000011001000011010000111001001110010110
0100001100110011100100110110001110000110010000110100001100000
1100010001101010110001000111001011001010110001100110111001110
0000110001001110010011100100110000001101110110010000111001011
0000101100110001100010011011000110000011001010011000000110100
0011000000110010000110110</td><td>32.81%</td></tr>
</table>

<table>
<tr><th>Change Password</th><th>ค่าบิตของแฮช</th><th>%</th></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.E" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<I" -i -k 3997 -t 4 -p 4 -l
32;</td><td>0110011001100101001100100011010100111000011001000110000100110
0010011001001100001001101010011001101100001011000100011000101
1001100110011001100010011000010011100100110110001100010110001
0011000100110010001100001011000110110011001100101001100100011
1001011000110110011001100001001100100011011001100011011000010
0111001001110010011100001100110001101000110011000110101011001
0001100101001101000011010000110111011001100110001101100110001
1100000111000001101010011100101100011011000110011010001100010
00110001001110010110010 1</td><td>35.55%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.I" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<Q" -i -k 3996 -t 5 -p 5 -l
32;</td><td>0011011101100100011000100011011100110010001101010110001001100
0100011000100111001001101100011001000110111011000010011011001
1000110110010101100100011001000011000101100101001110010110011
0001100110011011100110100001100100011000000110111011000110011
1000011001010011011100110111001100010011011000110010001110010
0110101001101110011011000111001001100010011011101100100001101
0100110111001100010011100000110111001110010011100000110001001
1010101100110011000100011011001100100011001100011010100110001
0110001101100001001100 00</td><td>32.62%</td></tr>
<tr><td>echo -n
"r=IlH<7+ej[fv7u,A&Ob%H
3x7J.Q" | argon2
"gGvRivJNky7)gsu:Bhx8j7R
.<a" -i -k 3995 -t 6 -p 6 -l
32;</td><td>0011100100110000011001100011011101100101001101010011001101100
1010011001000110000001101100011010001100101001100010011001101
1001010110001101100001001100000011010101100100001100000011011
0011001000110010000110010001101100011001000110111001101010110
0101001101000110010000110001011000010011100000110101001100100
1100100001101110011011101100001001110000011011100110001001100
0100111001001100000011100100110000001110000011001000111001001
1000101100100011001100110001101100011001101010110000101100100
0011001100111001001100 00</td><td>29.88%</td></tr>
<tr><td></td><td>เฉลี่ย</td><td>33.01%</td></tr>
<tr><td>r=IlH<7+ej[fv7u,A&Ob%H3
x7J.A</td><td>0011000000110001011000110110000100111000011001000011001101100
1100011010000110011001110000011011101100110001101110011000100
1100010011011000110001011001010011100000110100001110010110001
1011001100110010001100011001110000011000000110000011000010011
010000111001</td><td></td></tr>
<tr><td>r=IlH<7+ej[fv7u,A&Ob%H3
x7J.@</td><td>0011011000110110011000010011001100110111001100000011100101100
0110011001001100011001110000011100000110000001100110011000000
1110010110000101100001011001000011100101100101011000010011001
1001101100011011001100001001101010110001000111001001110010011
011000111000</td><td>27.73%</td></tr>
</table>

Change Password	ค่าบิตของแฮช	%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.C	0110011000110100001110010011000100110001011000100011010100110 1000011100000110101001101110011001100110101001100000110011001 1001010011001100110001001110010110000100110011001110010110001 0001101000110001101100100011000100110001000110010011000100110 010101100010	33.59%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.E	0011000001100110001100000011010100111001011000110011100000110 0100011000000110100011000110011011100111000011001000110011000 1110010011011000110110011000100011000100111000001101010011000 0001100010011001101100001001101000011011101100011011001000011 000101100110	35.94%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.I	0011010000110101001110010011011100110001011001010011100000110 1110011011000111001001100110011011001100010011001100011000000 1100010110010001100101001101110110011001100110001100010110011 0001101010110010000110011011001100110001001100010001100000011 010000110011	28.52%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.Q	0110000100110000001110010110010001100101001100110011100001100 0100110010000111001001110010110001000111001001100110011001100 1101100110001101100010001101100011010100110010011001100011001 0011001100110000100110000001100010110000100111001001100010110 010101100001	35.55%
	เฉลี่ย	32.27%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.A	0011000000110000011001000011100100110111011000100011010000110 1100011001001100010011001100110001000110101011000010110001100 1101100110001101100100011000110110010100110101001100010011100 1011000100011010100110000001101110110010100111001001101010110 0100001110010110010110001001100010011000110011000100111001010 011011101100011	
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.@	0011001101100101011000100110010000110100001110000011100100110 1010011000100110001011000100011100000110101001110010011100000 1100110110001101100010011000010011011001100010001100010011100 1001110010011000000110011001101010011011100110000001101110011 0111011001010011011000110100001110010110011000110101011001010 011100000110010	33.75%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.C	0011000000110100001110010110001100110110001101010110000100111 0010011000001100110001100100011010000110010001110000011000000 1101100011010101100001001100110011001000110001001100100110001 1001100100011001000111000001101000110001101100110001100010110 0010011000110011100001100110011000010011000000110011011000010 011100100110111	34.06%

Change Password	ค่าบิตของแฮช	%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.E	011001000011011000111001001101000011010000110011001101000011 0 11101100001001101100011001100111000011001100011000101100100 01 10010000111001001101000011010000110100001101110011000001100 00 10110010001100110001100100110010100110001011001000110010100 11 01100110001001100101001100000110000101100001001100100011010 10 110001001100001	33.75%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.I	011000110011000000110100011001000110001100110100001100010011 1 00001100010001100010011100001100001001100110110010001100100 01 10010101100100011000100011010001100100001101110011100001100 10 10011010001100011011000010110010000111000011000100110000101 10 00010011010101100101011000110110001001100101001101000011011 00 110010100110110	35.31%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.Q	001100100011100000110010001100000011011001100011001110000011 0 00101100100001100000011011001100011011000010110001001100010 01 10001100110011001101010011010101100010011001100011100001100 01 00011000100110111001100000011100101100001011001000011001101 10 00110011011001100100011000100011100100110100011000010011010 10 011001101100010	30.62%
	เฉลี่ย	33.50%
ทดสอบ SHA256		
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.A	0011010000110010001100000110000101100110001101010011011001100 1100011010100111001001110000011010001100100001100010011001001 1001000110011001100001001110010011011100110100011000010110001 1001100000011000101100011011000010011001100111001001110010110 0101001100110110010000111000001101100011100000111000001101000 0110001001110010110001001100110001101110011011000110100011001 1001100001001100110011010000110111001101110011000001100101001 1011001100100001101000110010101100101001100000011100000110011 0110010101100110001110 01	
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.@	0110000101100011001100010011010101100010001100110011011101100 1000011010100111000000111001001101110011000100110001001100010 1 1001100110001100110100001100110011000101100101011001000011010 1011000110011100101100110011000010110010000110100001101010011 0111001101000011010000110101011000010011001001100001001100010 0110101001101000011010101100100011001010110001001100001011000 1100110100001110000011011100111001001101110011100001100101001 1011000111000011001000110011000110101001100110011100100110101 0011001100110001001101 10	19.53%

Change Password	**ค่าบิตของแฮช**	**%**
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.C	00110001001100010011100001100110001100110011001101100101011000 0110110010000110111011000110011000101100010011000010011001101 10001100110101011001010110010101100001001101110011011000111000 00011100101100110001101100011001100111001001101100110000101100 00100110010000110001011000100110001100111001011001100110001000 01101100011001100110000011000100011010101100110001110000011100 0100110001001100100110001100110111001101100110010000110010001 10011011001000011001100111000011000100110000100110010011000100 0110001101100100000111001	21.88%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.E	00110011001100010011100000110001001100110110010000110000001100 00100110111001101000011011100110111001100110011010001100100000 1101010011001100111001011000100011010001100100001100010110001 00110011001100101001101100110011000111001011001010011001101100 00010011010101100100011000010011000100110011001100010011001100 1100110001100000011000000110000011001010110001100110100001101 0001100010011000110011011000110110011001010110001000110111011 0000100110010011001100011011100110001001100000011001100111001 001100100011011100111000	20.90%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.I	01100110001110010011011000110010001101000011011000110101001100 1110011001000110010001100100011011001100011011000100011100101 10001000110100001101100110001000111001001100100110000100111000 10011001100110111001100000011010101100110001100010011011001100 00100011000001100110011001000011000100110100001101100011100000 11001100110000101100010011001010011000000110110011001000011000 1101100011001101010110001001100101001101110110000100111000001 11001011000110110010001100011001100100110000101100010011001100 001101100110011000110001	20.90%
r=IlH<7+ej[fv7u,A&Ob%H3 x7J.Q	00110010001101010011100100111000001110000110010100110000011000 10000110011011001000011000001100110001101100011010001100100000 11011000110101011000110011001000110101001110000011011001100000 10011010001100110011000110011000001100001001101110011000000110 00100011011001100100011000010011001001100010011000100011010100 01110000110001100110111011001000011000100110011001101000110000 0100110110011001010011011101100001001101110110000101100001001 1100100110010001110000110001001100011001101100011011001100011 011000110011000101100010	20.70%

ประวัติผู้เขียน

ชื่อ - นามสกุล	นาย พัชร กกสูงเนิน
วัน เดือน ปีเกิด	6 สิงหาคม พ.ศ. 2541
สถานที่ทำงาน	สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีสุรนารี 111 ถ.มหาวิทยาลัย ต. สุรนารี อ.เมือง จ.นครราชสีมา 30000
ประวัติการศึกษา	พ.ศ. 2564 วิทยาการสารสนเทศบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีสุรนารี
ประสบการณ์ทำงาน	
พ.ศ. 2564 - ปัจจุบัน	เจ้าหน้าที่วิเคราะห์ระบบ สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีสุรนารี

ประวัติผู้เขียน

ชื่อ - นามสกุล	นาย พัชร กกสูงเนิน
วัน เดือน ปีเกิด	6 สิงหาคม พ.ศ. 2541
สถานที่ทำงาน	สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีสุรนารี 111 ถ.มหาวิทยาลัย ต. สุรนารี อ.เมือง จ.นครราชสีมา 30000
ประวัติการศึกษา	พ.ศ. 2564 วิทยาการสารสนเทศบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีสุรนารี
ประสบการณ์ทำงาน	
พ.ศ. 2564 - ปัจจุบัน	เจ้าหน้าที่วิเคราะห์ระบบ สถาบันวิจัยและพัฒนา มหาวิทยาลัยเทคโนโลยีสุรนารี