เครื่องมือทดสอบความแข็งแกร่งรหัสผ่าน
PASSWORD PENETRATION TOOLS

ธีรัตน์ ชินรัตนตรัย
นฤพัชร เชาว์อรัญ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

เครื่องมือทดสอบความแข็งแกร่งรหัสผ่าน

PASSWORD PENETRATION TOOLS

ธีรัตน์ ชินรัตนตรัย

นฤพัชร เชาว์อรัญ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

รายงานปีการศึกษา 2557

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องมือทดสอบความแข็งแกร่งรหัสผ่าน

PASSWORD PENETRATION TOOLS

ผู้จัดทำ

1. นายธีรัตน์ ชินรัตนตรัย รหัสนักศึกษา 54010643
2. นายนฤพัชร เชาว์อรัญ รหัสนักศึกษา 54010681

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

เครื่องมือทดสอบความแข็งแกร่งรหัสผ่าน

นายธีรัตน์ ชินรัตนตรัย 54010643
นายนฤพัชร เชาว์อรัญ 54010681
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2557

บทคัดย่อ

ในปัจจุบันความต้องการที่จะรักษาความปลอดภัยมีมากขึ้นจึงมีการใช้เครื่องมือหรือความสามารถของโปรแกรมในการเข้ารหัสลับต่าง ๆ เพื่อไม่ให้ผู้ที่ไม่เกี่ยวข้องสามารถเข้าถึงไฟล์ได้ โดยจะใช้รหัสผ่านเป็นตัวกำหนดสิทธิ์ ซึ่งยิ่งรหัสผ่านมีความซับซ้อนมากเท่าใดก็จะยิ่งทำให้ไฟล์มีความปลอดภัยมากขึ้น ปัญหาที่ตามมาคือยิ่งตั้งรหัสผ่านซับซ้อนมากเท่าใดก็จะยิ่งจำได้ยาก และหากลืมรหัสผ่านก็จะไม่สามารถเข้าถึงไฟล์ได้ ในปัจจุบันมีเครื่องมือที่ใช้หารหัสผ่านอยู่หลากหลายแต่ส่วนใหญ่นั้นสนับสนุนเพียงไฟล์นามสกุลเดียว และไม่สามารถใช้การประมวลผลแบบขนานได้ จากข้อจำกัดดังกล่าวผู้พัฒนาจึงมีความต้องการที่จะพัฒนาโปรแกรมเพื่อที่จะสามารถหารหัสผ่านได้หลายนามสกุลและสามารถใช้การประมวลผลแบบขนานเพื่อเพิ่มประสิทธิภาพได้

PASSWORD PENETRATION TOOLS

Mr. Thirat Chinrattanatrai 54010643
Mr. Naruepach Chowarun 54010681
Asst. Akkradach Watcharapupong Advisor
Academic Year 2014

ABSTRACT

Nowadays, the need of security is increasing. Encryption Tools and functions in application are used in order to protect person that does not have permission accesses files by using password to protect files. The more complex password, the more secure. The problem that comes up with complexity is it is hard to remember and cannot access files when forget their password. At the present, There are many password recovery tools but most of them still can recover only one specific format and cannot use multiprocessing. From those limitations, we want to develop program in order to recovery more than one format and be able to use multiprocessing in order to increase program performance.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยคำแนะนำและคำปรึกษาจาก อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา

ขอขอบคุณ นาย สุรพงศ์ เท่าเทียมตน และ นาย สรัล ศิริพันธ์โนน ที่ให้คำปรึกษาในเรื่องการเขียนโปรแกรมทำให้สามารถเรียนรู้ได้เร็วมากขึ้น รวมไปถึงเพื่อน ๆ ที่อยู่ด้วยกันมาจนจบ

ขอขอบคุณห้องวิจัย ISAG ที่เอื้อเฟื้อสถานที่ในการทำงานวิจัย และ เครื่องคอมพิวเตอร์ที่ใช้ในการทดลอง

ขอขอบคุณอาจารย์คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ประสิทธิ์ประสาทความรู้จนสามารถนำความรู้ไปประยุกต์ใช้ทำงานวิจัยได้

สุดท้ายนี้ทางคณะผู้จัดทำต้องขอบพระคุณเป็นอย่างสูง และ หวังเป็นอย่างยิ่งว่าปริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์แก่ผู้อ่านไม่มากก็น้อย

นฤพัชร เชาว์อรัญ
ธีรัตน์ ชินรัตนตรัย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญตาราง (ต่อ)

ตาราง **หน้า**

สารบัญรูปภาพ

รูป **หน้า**

สารบัญรูปภาพ (ต่อ)

รูป **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องจากในปัจจุบันความต้องการที่จะรักษาความปลอดภัยของข้อมูลในรูปแบบของไฟล์เอกสารมีมากขึ้น จึงมีการสร้างความสามารถที่จะเข้ารหัสลับไฟล์ต่าง ๆ ได้เพื่อไม่ให้ผู้ที่ไม่เกี่ยวข้องเข้าถึงไฟล์ได้ โดยจะใช้รหัสผ่านเป็นตัวกำหนดสิทธิ์ในการเข้าถึงข้อมูลและเนื่องจากวิทยาการเข้ารหัสลับมีความก้าวหน้าอย่างมากโดยการนำแฮชซิ่งฟังก์ชันมาใช้ร่วมกับการเข้ารหัสลับ

ดังนั้นหากลืมรหัสผ่านก็จะไม่สามารถเข้าถึงไฟล์ได้และการโจมตีที่ทำให้ได้มาซึ่งกุญแจที่ใช้ในการเข้ารหัสลับทำได้โดยยากจึงต้องใช้การโจมตีโดยการลองหารหัสลับโดยใช้รหัสทุกแบบที่เป็นไปได้เพื่อหาว่ารหัสผ่านไหนคือรหัสผ่านที่ถูกต้อง ซึ่งหากทดสอบกับทุกรหัสผ่านที่เป็นไปได้จะใช้เวลาในการประมวลผลมาก จึงมีการพัฒนารูปแบบการโจมตีเพื่อลดขอบเขตที่เป็นไปได้ให้น้อยลง เช่น Dictionary Attack เป็นการโจมตีโดยใช้คำจากในพจนานุกรมที่เตรียมไว้และการพัฒนาของหน่วยประมวลผลในปัจจุบันทำให้มีประสิทธิภาพในการประมวลผลมากขึ้นทำให้ลดเวลาในการประมวลผลลงไปได้และสามารถลดเวลาในการประมวลผลลงได้อีกโดยใช้สถาปัตยกรรมการประมวลผลแบบขนานแบ่งการประมวลผลไปที่โพรเซสเซอร์หลาย ๆ ตัว

ในปัจจุบันมีโปรแกรมหารหัสผ่านอยู่มากมายแต่ส่วนใหญ่จะสนับสนุนเพียงนามสกุลเดียวและไม่ได้มีรูปแบบการโจมตีที่ดี เช่น Mask Attack อีกทั้งยังไม่สามารถใช้การประมวลผลแบบขนานมาช่วยประมวลผล

ด้วยข้อจำกัดของความสามารถของโปรแกรมในปัจจุบัน จึงเป็นที่น่าสนใจที่จะศึกษาและพัฒนาการหารหัสลับของข้อมูลไฟล์เอกสารให้อยู่ในโปรแกรมเดียวกันเพื่อเพิ่มความสะดวกในการใช้งานและเพิ่มประสิทธิภาพโดยการประมวลผลแบบขนาน เพื่อเป็นประโยชน์ในการใช้งาน ศึกษาและพัฒนาต่อไปในอนาคต

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างเครื่องมือในการหารหัสผ่านได้หลายหลายนามสกุล
2. เพื่อพัฒนาโปรแกรมให้มีประสิทธิภาพสูงขึ้นโดยใช้การประมวลผลแบบขนาน

1.3 ขอบเขตของโครงงาน

1. สามารถหารหัสผ่านของไฟล์ได้หลายนามสกุล เช่น ZIP, RAR, PDF
2. สามารถเลือกรูปแบบวิธีการสำหรับการหารหัสผ่านได้
3. สามารถใช้การประมวลผลแบบขนานเพื่อทำให้ประสิทธิภาพดีขึ้นได้

1.4 วิธีการดำเนินการ

1. ศึกษาโครงสร้างของไฟล์ document
2. ทดลองโปรแกรมที่ใช้เดารหัสผ่าน
3. หาแนวทางพัฒนาโปรแกรม
4. ศึกษาภาษา Python เบื้องต้น
5. ศึกษาและทดลองการหารหัสผ่านของไฟล์นามสกุล .ZIP
6. ศึกษาและทดลองการหารหัสผ่านของไฟล์นามสกุล .RAR
7. หาแนวทางการปรับปรุงประสิทธิภาพของโปรแกรม
8. ศึกษาและทดลองการทำการประมวลผลแบบขนาน
9. ศึกษาและทดลองการหารหัสผ่านของไฟล์นามสกุล .PDF
10. ศึกษาและทดลองการโจมตีแบบ Dictionary
11. ศึกษาและทดลองการโจมตีแบบ Brute-Force
12. ศึกษาและทดลองการโจมตีแบบ Mask
13. นำส่วนต่างๆของโปรแกรมมารวมเข้าด้วยกัน
14. ทดลองและตรวจสอบหาข้อผิดพลาดต่างๆของโปรแกรม

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ความรู้ความเข้าใจของการใช้ภาษา Python
2. ความรู้เกี่ยวกับรูปแบบการโจมตีต่าง ๆ ให้ได้มาซึ่งรหัสผ่าน
3. การทำ Multiprocessing
4. โปรแกรมหารหัสผ่าน

1.6 ส่วนประกอบของรายงาน

รายงานเล่มนี้ประกอบด้วยส่วนประกอบ 5 ส่วน คือ

บทที่ 1 บทนำ กล่าวถึง ความเป็นมา วัตถุประสงค์ ขอบเขตการดำเนินงานของโครงการ วิธีการดำเนินงาน ผลประโยชน์ที่คาดว่าจะได้รับ และ ส่วนประกอบของรายงาน

บทที่ 2 ทฤษฎีที่เกี่ยว ข้อง กล่าวถึง ทฤษฎีพื้นฐานที่ใช้ในการทำโครงงานประกอบด้วย วิทยาการการเข้ารหัสลับ รูปแบบการโจมตีเพื่อให้ได้มาซึ่งรหัสลับและสถาปัตยกรรมการประมวลผลแบบขนาน

บทที่ 3 การออกแบบและพัฒนา กล่าวถึง รายละเอียดของโปรแกรมที่พัฒนา และ การทำงานของโปรแกรม

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึง รายละเอียดการทดลองของโปรแกรม ผลการทดลอง ประสิทธิภาพของโปรแกรม และ วิเคราะห์ผลการทดลอง

บทที่ 5 บทสรุป กล่าวถึง บทสรุปของโครงงาน ปัญหาและอุปสรรคต่าง ๆ ของโครงงาน แนวทางการแก้ไขและพัฒนาต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 Python

เป็นภาษาชั้นสูงที่ออกแบบมาให้โค้ดที่เขียนสามารถอ่านแล้วเข้าใจ โดยตัวภาษา Python สนับสนุนการเขียนโปรแกรมหลายรูปแบบ เช่น เชิงวัตถุ หรือ เชิงโครงสร้าง และ มีการจัดการเมมโมรี่ให้โดยอัตโนมัติ และ การจัดการชนิดตัวแปรตามข้อมูลที่บรรจุอยู่ และในปัจจุบัน ภาษา Python ถูกพัฒนาไปถึงเวอร์ชัน 3

Python เป็นภาษาสคริปจะทำงานโดยใช้ Interpreter อ่านโค้ดทีละบรรทัดแล้วทำการประมวลผลซึ่งจะต่างจาก compiler ที่ทำการอ่านโค้ดทุกบรรทัดแล้วประมวลผล

2.2 Race Condition

เป็นปัญหาที่เกิดขึ้นในการเขียนโปรแกรมแบบ Multi-threaded หรือ Multiprocessing โดยที่ Thread หรือ Process ต้องการที่จะใช้ข้อมูลตัวเดียวกันร้อม ๆ กันซึ่งจะทำให้ได้ค่าที่ผิดไป ไม่ว่าจะเป็นการเขียน หรือ อ่าน

ตาราง 2.1 ตัวอย่างปัญหา Race Condition

Thread 1	Thread 2	Data	ค่าที่ Thread 1 ใช้งาน	ค่าที่ Thread 2 ใช้งาน
		0		
อ่านค่า		0	0	
	อ่านค่า	0		0
เพิ่มค่า		0	1	0
	เพิ่มค่า	0		
เขียนค่า		1	1	
	เขียนค่า	1		1

ซึ่งจริง ๆ แล้วค่าที่ได้ในตอนสุดท้ายควรจะเป็น 2 แต่เนื่องจากเกิดปัญหา Race condition ทำให้ Thread 2 เขียนทับค่าที่ได้จาก Thread 1

2.3 Thread Safety

เป็นแนวคิดของการทำ Multi-threaded โดยที่โค้ดส่วนใด ๆ จะเป็น Thread safe ก็ต่อเมื่อสามารถรับประกันได้ว่าเมื่อใช้หลาย ๆ thread พร้อมกันจะไม่มีปัญหา Race Condition โดยสามารถแบ่งระดับของ Thread Safety ได้เป็น 3 ระดับดังนี้

- Thread safe รับประกันได้ว่าไม่มีปัญหา Race Condition เมื่อใช่หลาย Thread
- Conditionally safe แต่ละ Thread สามารถใช้ข้อมูลที่ต่างกันพร้อม ๆ กันได้ และ สามารถใช้ข้อมูลส่วนกลางที่ได้รับการป้องกันจากปัญหา Race Condition
- Not thread safe โค้ดไม่ควรถูกใช้งานจากหลาย ๆ Thread พร้อม ๆ กัน

ซึ่งสามารถแก้ไขปัญหา Race Condition เพื่อทำให้เป็น Thread safe แบ่งออกเป็น 2 วิธี

- หลีกเลี่ยงการใช้ข้อมูลร่วมกัน
- การใช้ synchronization

2.4 Lock

เป็นกระบวนการแก้ไขปัญหา Race condition ที่ง่ายที่สุด ซึ่งเป็นการแก้ไขโดยการใช้ synchronization โดยจะทำการ Lock ข้อมูลที่มี Thread อื่นใช้งานอยู่ เพื่อที่จะไม่ให้ Thread อื่นมาใช้งานข้อมูลนั้น ๆ โดยที่การ Lock ข้อเสียดังต่อไปนี้

1. เกิดการ Block ทำให้ Thread ต้องรอจนกว่า Thread ที่ใช้งานข้อมูลนั้นทำงานจนเสร็จแล้วปล่อย Lock
2. อาจจะทำให้เกิดปัญหา Dead Lock
3. ทำให้เป็นโปรแกรมแบบ serial เพราะทำได้ทีละ Thread หากต้องการใช้ข้อมูลเดียวกัน

2.5 Global Interpreter Lock (GIL)

เนื่องจากภาษา Python เป็นภาษาที่รันโดยใช้ interpreter ซึ่งตัว interpreter เองจะต้องจัดการข้อมูล และ Thread ซึ่งตัว interpreter จะใช้วิธี Lock เพื่อไม่ให้ใช้ข้อมูลเดียวกันจากหลาย ๆ Thread เพื่อป้องกันปัญหา Race condition ซึ่งจะเรียกว่า Global Interpreter Lock ซึ่งสามารถรับประกันได้ว่าเป็น Thread Safe ซึ่งจะทำให้ในทุก ๆ โปรแกรมที่เขียนโดยภาษา Python จะมีเพียง Thread เดียวที่ทำงานได้ในช่วงเวลาเดียวกันทำให้ไม่สามารถใช้ Multi-threaded เพื่อเพิ่มประสิทธิภาพของโปรแกรมได้ จึงจำเป็นต้องใช้ Multiprocessing แทน

2.6 วิทยาการรหัสลับ

เป็นวิทยาการสำหรับการรักษาความลับของของข้อมูลเพื่อไม่ให้ผู้ที่ไม่มีสิทธิสามารถเข้าถึงหรือใช้งานข้อมูลที่ทำการเข้ารหัสลับได้โดยจะแบ่งเป็น 2 ประเภทคือ

1. Symmetric algorithm
2. Asymmetric algorithm

โดยที่การเข้ารหัสลับของไฟล์ต่าง ๆ จะใช้วิธีเข้ารหัสลับแบบ Symmetric algorithm

2.6.1 Symmetric algorithm

เป็น algorithm ที่ใช้ Key เดียวกันทั้งการเข้ารหัสลับและถอดรหัสลับ

2.6.1.1 AES algorithm

เป็น Algorithm ที่คิดค้นโดย Jonan Daemen และ Vincent Rijment ในปี2000 และเผยแพร่ในปี 2001 โดย NIST ซึ่งจะเป็นแบบ Block Cipher ที่มีขนาด 128 bits และใช้ Key ที่มีความยาวต่างกัน คือ 128,192 และ 256 bits โดยจะมีขั้นตอนการเข้ารหัสลับดังนี้

1. Substitute byte
2. Shift Rows
3. Mix Columns
4. Add Round Key

และถอดรหัสลับดังนี้

1. Inverse Shift Rows
2. Inverse Substitute Bytes
3. Add Round key
4. Inverse Mix Columns

โดยที่จำนวนครั้งของการทำจะขึ้นอยู่กับขนาดของ Key ดังนี้

1. 128 bit keys ทำ 10 รอบ
2. 192 bit keys ทำ 12 รอบ
3. 256 bit keys ทำ 14 รอบ

ซึ่งแต่ละรอบจะใช้ Key ที่สร้างมาจาก Key ในตอนแรกที่ได้มานำมาทำตาม Key Expansion Algorithm โดยที่รอบสุดท้ายจะมีการทำ Mix Columns สำหรับการเข้ารหัสลับและไม่มีการทำ Inverse Mix Columns สำหรับการถอดรหัสลับ

2.6.2 Asymmetric algorithm

เป็น algorithm ที่ใช้ Key ต่างกันในการเข้าและถอดรหัสลับ

2.7 รูปแบบการโจมตีเพื่อให้ได้มาซึ่งรหัสลับ

รูปแบบการโจมตีต่าง ๆ ที่ใช้เพื่อให้ได้มาซึ่งรหัสผ่าน จะเป็นการแลกกันระหว่างเวลาและหน่วยคำจำ (Time-Memory Trade-Off) หรือ ระหว่างเนื้อที่และเวลา (Time-Space Trade-Off) ซึ่งการเลือกการโจมตีต่าง ๆ ก็ต้องพิจารณาว่าจะใช้แบบไหน เช่น หากมี Space ที่มากพอก็สามารถใช้ Dictionary Attack เพื่อลดเวลาได้ หรือ หากมีหน่วยความจำที่มากพอก็สามารถใช้ Brute-force Attack เพื่อลดเวลาได้

2.7.1 Brute-force Attack

เป็นการโจมตีโดยใช้รหัสลับจากการนำเซตของตัวอักษรที่สนใจมาคูณคาร์ทีเซียนกันโดยกำหนดความยาวของรหัสลับ ตัวอย่างเช่น ทดสอบโดยใช้เซตของตัวอักษรภาษาอังกฤษตัวพิมพ์เล็ก (a,b,c,...,x,y,z) ขนาดความยาว 8 ตัวจะได้รหัสลับที่เริ่มตั้งแต่ “aaaaaaaa” ถึง “zzzzzzzz”

โดยการใช้รูปแบบการโจมตีแบบนี้จะรับประกันผลสำเร็จ 100 % หากตัวอักษรของรหัสลับทั้งหมดอยู่ในเซตของตัวอักษรที่เลือกมาทำผลคูณคาร์ทีเซียนและมีขนาดเท่ากับที่กำหนด

อย่างไรก็ตามเมื่อรหัสลับนั้นมีความยาวมากขึ้นจะทำให้เซตของรหัสลับที่จะนำมาทดสอบเพิ่มขึ้นแบบชี้กำลังซึ่งการโจมมตีแบบนี้ทำให้เสียเวลาไปกับการทดสอบรหัสผ่านที่ไม่ถูกต้องจากเหตุผลข้างต้นทำให้การโจมตีแบบนี้ไม่ค่อยมีประสิทธิภาพ

2.7.2 Dictionary Attack

เป็นโจมตีโดยการอ่านมารหัสลับจากไฟล์ Dictionary ทีละบรรทัดโดยที่ไฟล์ Dictionary เป็น text file ที่เก็บคำศัพท์หรือประโยคที่มีความเป็นไปได้ที่จะเป็นรหัสลับบรรทัดละหนึ่งคำหรือประโยค ซึ่งการโจมตีแบบนี้จะต่างจาก Brute-force ที่จะไม่เก็บรหัสลับที่ได้จากการคูณคาร์ทีเซียนทั้งหมดแต่จะเก็บรหัสลับที่มีความเป็นไปได้ที่จะถูก โดยจะพิจารณาจากความนิยมของคำนั้นที่นำมาใช้เป็นรหัสลับ,เป็นคำที่มีความหมาย,เป็นเบอร์โทรศัพท์ และอื่น ๆ โดยการโจมตีแบบนี้จะไม่รับประกันว่าจะสำเร็จแต่อัตราการประสบผลสำเสร็จจะขึ้นอยู่กับคุณภาพของไฟล์ Dictionary

2.7.3 Mask Attack

เป็นการโจมตีที่คล้าย ๆ กับ Brute-Force Attack แต่จะมีความเฉพาะเจาะจงมากกว่าโดยผู้ใช้งานสามารถออกแบบรหัสลับที่จะนำมาทดสอบได้โดยจะต้องกำหนดเซตของตัวอักษรที่จะนำมาคูณคาร์ทีเซียนแต่ละหลักนั้นจะสามารถใช้เซตตัวอักษรที่ต่างกัน ซึ่งรูปแบบการโจมตีแบบนี้จะสามารถลดเซตของรหัสลับที่เป็นไปได้หากรู้ข้อมูลบางส่วนของรหัสลับตัวอย่างเช่น ต้องการรหัสลับ “Az9” ถ้าหากใช้รูปแบบการโจมตีแบบ Brute-force attack จะได้เซตของรหัสลับที่จะนำมาทดสอบขนาด 62*62*62 (238,328) ตัว แต่ถ้าผู้ใช้งานรู้ว่าตำแหน่งของรหัสลับแต่ละตัวอักษรอยู่ในเซตของตัวอักษรไหนก็จะสามารถลดเซตเหลือขนาด 26*26*10 (6,760) ตัว จากตัวอย่างจะเห็นได้ว่าขนาดของเซทของ

รหัสลับที่จะนำมาทดสอบมีขนาดเล็กกว่าซึ่งการโจมตีรูปแบบนี้มีประสิทธิภาพดีกว่าการโจมตีแบบ Brute-force

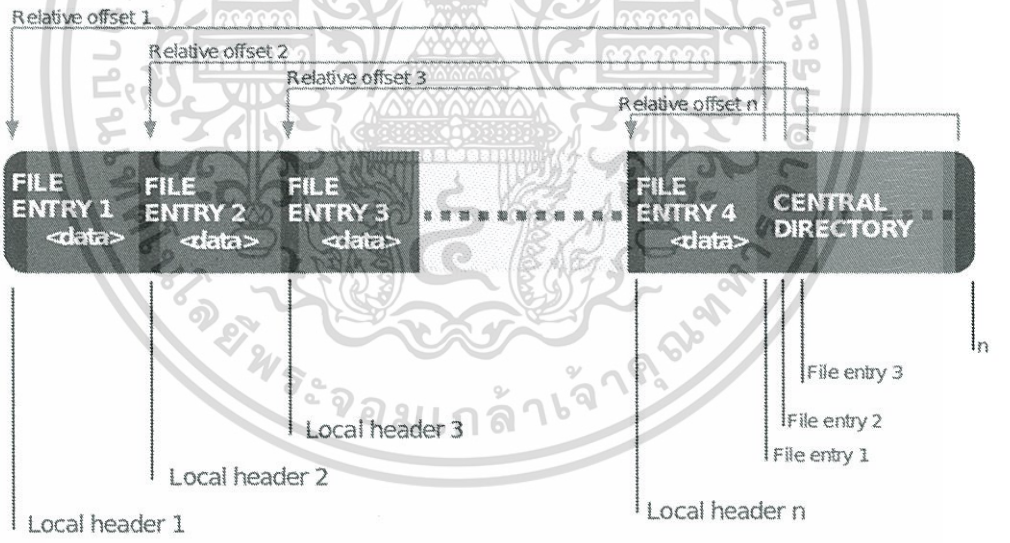
2.8 รูปแบบไฟล์ข้อมูลที่ใช้ในโครงงาน

2.8.1 ไฟล์นามสกุล ZIP

ไฟล์ ZIP เป็นไฟล์ที่สามารถเก็บไฟล์ต่าง ๆ หรือแฟ้มข้อมูลไว้ด้วยกัน ซึ่งสามารถทำการบีบอัดข้อมูลได้แบบ lossless และ สามารถตั้งรหัสผ่านเพื่อป้องกันผู้ที่ไม่มีสิทธิมาใช้งานได้โดย Algorithm ที่ใช้ในการเข้ารหัสลับ คือ AES และ กระบวนการตรวจสอบความถูกต้องของข้อมูล คือ CRC-32

2.8.1.1 โครงสร้างไฟล์นามสกุล ZIP

ไฟล์นามสกุล ZIP จะทำการเก็บข้อมูลของไฟล์ต่างๆ ที่อยู่ภายในไฟล์ ZIP ไว้ที่ Central directory แล้วทำการชี้ไปที่ Local Header ซึ่งเป็นที่เก็บข้อมูลของไฟล์ภายใน ซึ่งแสดงดังรูป โดยที่ไฟล์ ZIP จะใช้ข้อมูลขนาด 4 bytes เรียก Signature เพื่อทำการแยกความแตกต่างของโครงสร้างไฟล์ในทุกๆ Entries จะมี Signature ที่แตกต่างกัน



รูป 2.1 โครงสร้างของไฟล์นามสกุล ZIP

ตาราง 2.2 ส่วน Local Header File ของแต่ละไฟล์ในไฟล์นามสกุล ZIP

Offset	Bytes	Description
0	4	Local file header signature = 0x04034b50 (อ่านแบบ little-endian)
4	2	Version needed to extract (minimum)
6	2	General purpose bit flag
8	2	Compression method
10	2	File last modification time
12	2	File last modification date
14	4	CRC-32
18	4	Compressed size
22	4	Uncompressed size
26	2	File name length (*n*)
28	2	Extra field length (*m*)
30	n	File name
30+n	m	Extra field

ตาราง 2.3 ส่วน Data Descriptor ของแต่ละไฟล์ในไฟล์นามสกุล ZIP

Offset	Bytes	Description
0	0/4	Optional data descriptor signature = 0x08074b50
0/4	4	CRC-32
4/8	4	Compressed size
8/12	4	Uncompressed size

ตาราง 2.4 ส่วน Central Directory File Header ของไฟล์นามสกุล ZIP

Offset	Bytes	Description
0	4	Central directory file header signature = 0x02014b50
4	2	Version made by
6	2	Version needed to extract (minimum)
8	2	General purpose bit flag
10	2	Compression method
12	2	File last modification time
14	2	File last modification date
16	4	CRC-32
20	4	Compressed size
24	4	Uncompressed size
28	2	File name length (*n*)
30	2	Extra field length (*m*)
32	2	File comment length (*k*)
34	2	Disk number where file starts
36	2	Internal file attributes
38	4	External file attributes
42	4	Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header. This allows software reading the central directory to locate the position of the file inside the .ZIP file.
46	n	File name
46+n	m	Extra field
46+n+m	k	File comment

ตาราง 2.5 ส่วน End of Central Directory Record (ECOD) ของไฟล์นามสกุล ZIP

offset	Bytes	Description
0	4	End of central directory signature = 0x06054b50
4	2	Number of this disk
6	2	Disk where central directory starts
8	2	Number of central directory records on this disk
10	2	Total number of central directory records
12	4	Size of central directory (bytes)
16	4	Offset of start of central directory, relative to start of archive
20	2	Comment length (*n*)
22	n	Comment

2.8.2 โครงสร้างของไฟล์นามสกุล RAR

ไฟล์ RAR เป็นไฟล์ที่สามารถเก็บไฟล์ต่าง ๆ หรือแฟ้มข้อมูลไว้ด้วยกัน ซึ่งถูกพัฒนาโดย Eugene Roshal และเป็นลิขสิทธิ์ของ win.rar GmbH ซึ่งมีความสามารถเพิ่มขึ้นจากไฟล์ ZIP คือ สามารถ Recovery ได้ โดยสามารถตั้งรหัสผ่านเพื่อไม่ให้ผู้ที่ไม่มีสิทธิมาใช้งานได้โดยใช้ Algorithm ในการเข้ารหัสลับ คือ AES-256 และ กระบวนการตรวจสอบความถูกต้องของข้อมูล คือ CRC-32

ไฟล์ rar จะทำการแบ่งข้อมูลออกเป็น Block โดยจะมีเลข 4 ตัวเรียกว่า Magic number เพื่อใช้ในการอธิบาย Header

- 0x6152 - HEAD_CRC
- 0x72 - HEAD_TYPE
- 0x1A21 - HEAD_FLAGS
- 0x0007 - HEAD_SIZE

สำหรับไฟล์ RAR จะทำการแบ่งออกเป็น Block โดยจะมีฟิลด์ต่าง ๆ ดังนี้

ตาราง 2.6 โครงสร้างของ Block ภายในไฟล์นามสกุล RAR

Name	Bytes	Description
HEAD_CRC	2	CRC of total block or block part
HEAD_TYPE	1	Block type
HEAD_FLAGS	2	Block flags
HEAD_SIZE	2	Block size
ADD_SIZE	4	Optional field - added block size

โดยที่จะมีฟิลด์ ADD_SIZE ก็ต่อเมื่อ (HEAD_FLAGS & 0x8000) != 0 และ ขนาดของ Block จะมีขนาดเท่า HEAD_SIZE ถ้า (HEAD_FLAGS & 0x8000) == 0 และจะมีขนาด Block เท่ากับ HEAD_SIZE+ADD_SIZE เมื่อมีฟิลด์ ADD_SIZE

ตาราง 2.7 Header ของแต่ละ Block ในไฟล์นามสกุล RAR

Head Type Signifier	Description
HEAD_TYPE=0x72	marker block
HEAD_TYPE=0x73	archive header
HEAD_TYPE=0x74	file header
HEAD_TYPE=0x75	old style comment header
HEAD_TYPE=0x76	old style authenticity information
HEAD_TYPE=0x77	old style subblock
HEAD_TYPE=0x78	old style recovery record
HEAD_TYPE=0x79	old style authenticity information
HEAD_TYPE=0x7a	subblock
HEAD_TYPE=0x7b	terminator

รูปแบบของ Block ที่อยู่ในไฟล์นามสกุล RAR จะมีอยู่ 3 ชนิด คือ Marker Block, Archive Header และ File Header

ตาราง 2.8 โครงสร้างของ Block ชนิด Marker Block ภายในไฟล์นามสกุล RAR

Field Name	Bytes	Possibilities
HEAD_CRC	2	Always 0x6152
HEAD_TYPE	1	Header type: 0x72
HEAD_FLAGS	2	Always 0x1A21
HEAD_SIZE	2	Block size = 0x0007

ตาราง 2.9 โครงสร้างของ Block ชนิด Archive Header ภายในไฟล์นามสกุล RAR

Field Name	Bytes	Description
HEAD_CRC	2	CRC of fields HEAD_TYPE to RESERVED2
HEAD_TYPE	1	Header Type: 0x73
HEAD_FLAGS	2	Bit Flags (สามารถดูข้อมูลเพิ่มเติมได้ที่ตาราง 2.9).
HEAD_SIZE	2	Archive header total size including archive comments
RESERVED1	2	RESERVED
RESERVED2	4	RESERVED

ตาราง 2.10 ความหมายของ Bit Flag ภายใน Block ชนิด Archive Header ภายในไฟล์นามสกุล RAR

Flag (0x)	Description
0001	Volume attribute (archive volume)
0002	Archive comment present RAR 3.x uses the separate comment block and does not set this flag.
0004	Archive lock attribute
0008	Solid attribute (solid archive)
0010	New volume naming scheme ('volname.partN.rar')
0020	Authenticity information present RAR 3.x does not set this flag.
0040	Recovery record present
0080	Block headers are encrypted
0100	First volume (set only by RAR 3.0 and later)

ตาราง 2.11 โครงสร้างของ Block ชนิด File Header ภายในไฟล์นามสกุล RAR

Field Name	Bytes	Description
HEAD_CRC	2	CRC of fields from HEAD_TYPE to FILEATTR and file name
HEAD_TYPE	1	Header Type: 0x74
HEAD_FLAGS	2	Bit Flags (สามารถดูข้อมูลเพิ่มเติมได้ที่ตาราง 2.11)
HEAD_SIZE	2	File header full size including file name and comments
PACK_SIZE	4	Compressed file size

UNP_SIZE	4	Uncompressed file size
HOST_OS	1	Operating system used for archiving (สามารถดูข้อมูลเพิ่มเติมได้ที่ตาราง 2.13)
FILE_CRC	4	File CRC
FTIME	4	Date and time in standard MS DOS format
UNP_VER	1	RAR version needed to extract file (Version number is encoded as 10 * Major version + minor version.)
METHOD	1	Packing method (สามารถดูข้อมูลเพิ่มเติมได้ที่ตาราง 2.14)
NAME_SIZE	2	File name size
ATTR	4	File attributes
HIGH_PACK_SIZE	4	High 4 bytes of 64-bit value of compressed file size. Optional value, presents only if bit 0x100 in HEAD_FLAGS is set.
HIGH_UNP_SIZE	4	High 4 bytes of 64-bit value of uncompressed file size. Optional value, presents only if bit 0x100 in HEAD_FLAGS is set.
FILE_NAME	NAME_SIZE bytes	File name - string of NAME_SIZE bytes size
SALT	8	present if (HEAD_FLAGS & 0x400) != 0
EXT_TIME	variable size	present if (HEAD_FLAGS & 0x1000) != 0

ตาราง 2.12 ส่วน Bit Flag ภายใน Block ชนิด File Header ภายในไฟล์นามสกุล RAR

Flag (0x)	Description
01	File continued from previous volume
02	File continued in next volume
04	File encrypted with password
08	File comment present. RAR 3.x uses the separate comment block and does not set this flag.
10	Information from previous files is used (solid flag) (for RAR 2.0 and later)
Dictionary bits 7 6 5 (for RAR 2.0 and later)	สามารถดูข้อมูลเพิ่มเติมได้ที่ตาราง 2.12
100	HIGH_PACK_SIZE and HIGH_UNP_SIZE fields are present. These fields are used to archive only very large files (larger than 2Gb), for smaller files these fields are absent.
200	FILE_NAME contains both usual and encoded Unicode name separated by zero. In this case NAME_SIZE field is equal to the length of usual name plus encoded Unicode name plus 1. If this flag is present, but FILE_NAME does not contain zero bytes, it means that file name is encoded using UTF-8.
400	The header contains additional 8 bytes after the file name, which are required to increase encryption security (so called 'salt').
800	Version flag. It is an old file version, a version number is appended to file name as ';n'.
1000	Extended time field present.
8000	This bit always is set, so the complete block size is HEAD_SIZE + PACK_SIZE (and plus HIGH_PACK_SIZE, if bit 0x100 is set)

ตาราง 2.13 ความหมายของ Dictionary Bits ภายใน Block ชนิด File Header ภายในไฟล์นามสกุล RAR

Bits (7 6 5)	Description	Size (KB)
0 0 0	Dictionary Size	64
0 0 1	Dictionary Size	128
0 1 0	Dictionary Size	256
0 1 1	Dictionary Size	512
1 0 0	Dictionary Size	1024
1 0 1	Dictionary Size	2048
1 1 0	Dictionary Size	4096
1 1 1	file is a directory	N/A

ตาราง 2.14 ความหมายในส่วน HOST_OS ของ Block ชนิด File Header ภายในไฟล์นามสกุล RAR

Byte Indicator	Operating System
0	MS DOS
1	OS/2
2	Windows
3	Unix
4	Mac OS
5	BeOS

ตาราง 2.15 ความหมายในส่วน METHOD ของ Block ชนิด File Header ภายในไฟล์นามสกุล RAR

Byte Indicator	Method
0x30	Storing
0x31	Fastest Compression
0x32	Fast Compression
0x33	Normal Compression
0x34	Good Compression
0x35	Best Compression

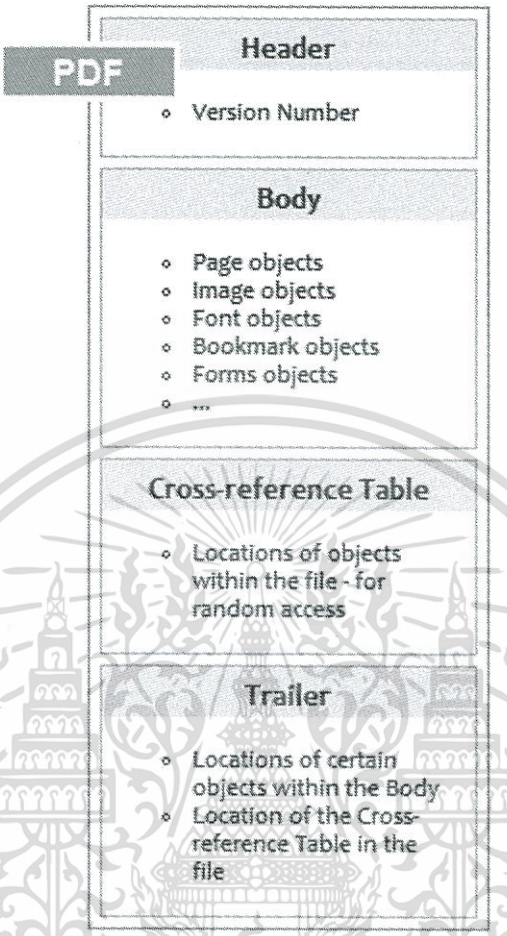
ตาราง 2.16 โครงสร้างของ Blok ชนิด Terminator

Field Name	Bytes	Possibilities
HEAD_CRC	2	Always 0x3DC4
HEAD_TYPE	1	Header type: 0x7b
HEAD_FLAGS	2	Always 0x4000
HEAD_SIZE	2	Block size = 0x0007

2.8.3 โครงสร้างของไฟล์นามสกุล PDF

ไฟล์ PDF หรือ Portable Document Format เป็นรูปแบบแฟ้มลักษณะหนึ่ง ที่พัฒนาโดยบริษัท Adobe System สำหรับแสดงเอกสารที่สามารถใช้งานได้ในทุกระบบปฏิบัติการ และยังคงลักษณะเอกสารเหมือนต้นฉบับ เอกสารในรูปแบบนี้สามารถจัดเก็บ ตัวอักษร รูปภาพ ลายเส้น ในลักษณะเป็นหน้าหนังสือ ตั้งแต่ หนึ่งหน้า หรือหลายพันหน้าได้ในแฟ้มเดียวกัน รูปแบบเป็นมาตรฐานที่เปิดให้คนอื่นสามารถเขียนโปรแกรมมาทำงานร่วมกันได้ นอกจากนี้ยังสามารถเข้ารหัสลับของไฟล์ได้ด้วย โดยจะมีอัลกอริทึมในการเข้ารหัสลับเป็นแบบ AES-256

ภายในไฟล์ PDF จะมีส่วนประกอบทั้งหมด 4 ส่วน ดังรูป



รูป 2.2 ส่วนต่างๆภายในไฟล์ PDF

ส่วน Header จะเป็นส่วนที่บ่งบอกว่า ไฟล์ดังกล่าวเป็นไฟล์ pdf และในส่วนนี้ก็บ่งบอกถึงเวอร์ชั่นของไฟล์ pdf นั้นๆอีกด้วย

ส่วน Body จะทำการเก็บ Object ที่อยู่ภายในไฟล์ pdf ซึ่งอาจประกอบไปด้วยข้อความต่างๆ, รูปภาพ, หรือมัลติมีเดียอื่นๆ ซึ่ง Object ที่ถูกเขียนลงในส่วนนี้จะถูกใช้ในการแสดงผลต่อไป

ส่วน Cross-reference Table จะเป็นส่วนที่อ้างอิงถึงทุกๆ Object ในไฟล์เอกสาร เหตุผลที่จำเป็นต้องมีส่วนนี้อยู่ เนื่องจากจะทำให้ไฟล์สามารถเข้าถึง Object ต่างๆ แบบ Random Access ได้ กล่าวคือ ไม่จำเป็นต้องอ่านไฟล์ pdf ทั้งหมด จึงจะสามารถระบุตำแหน่งของแต่ละ Object ได้ แต่ละ Object จะถูกนำเสนอด้วยรายการ 1 รายการภายในส่วน Cross-reference Table โดยที่แต่ละรายการจะมีความยาวเท่ากับ 1 byte เสมอ

ส่วน Trailer จะเป็นส่วนที่บ่งบอกว่า Application ที่ทำการอ่านไฟล์ สามารถหา Cross-reference Table ได้ที่ใดในไฟล์ และบ่งบอกถึง Object พิเศษอื่นๆ ทุก Application ที่จะทำการอ่านไฟล์ pdf จะต้องทำการเริ่มต้นอ่านจากท้ายไฟล์เป็นอันดับแรก

2.9 กระบวนการการตรวจสอบความถูกต้องของข้อมูล

หลังจากการประมวลผล จำเป็นต้องมีการตรวจสอบความถูกต้องของข้อมูล เพื่อให้มั่นใจว่าข้อมูลที่ได้นั้นไม่มีการปลอมแปลง หรือ เสียหาย

2.9.1 Cyclic redundancy check (CRC)

เป็นรูปแบบหนึ่งของแฮชฟังก์ชันซึ่งนำมาใช้ประโยชน์ในการตรวจสอบความถูกต้องของข้อมูล ซึ่งจะมีประสิทธิภาพดีกว่า การใช้บิตตรวจสอบ (Parity Bit) และ การหาผลรวม (Checksum) โดยจะใช้วิธีหาร โพลิโนเมียล (Polynomial) เพื่อหาเอาเศษ ซึ่ง CRC จะใช้ โพลิโนเมียลที่แตกต่างกันไป

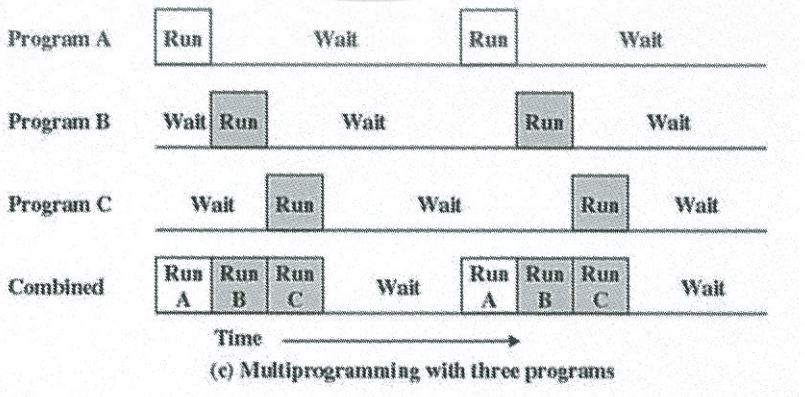
โดย โพลิโนเมียลที่ใช้ในไฟล์นามสกุล ZIP และ RAR คือ CRC32 โดยจะมีโพลิโนเมียลดังนี้

$$CRC32 = X^{32} + X^{26} + x^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{3} + X + 1 \quad (1.1)$$

2.10 สถาปัตยกรรมประมวลผลแบบขนาน

2.10.1 Multiprogramming

เป็นเทคนิคที่ใช้ Main memory ในการเก็บหลายๆ programs โดยที่เมื่อโปรแกรมหนึ่งที่กำลังทำงานอยู่ และต้องการทำ I/O operation CPU จะทำการเปลี่ยนไปทำงานอีกโปรแกรมหนึ่งแทน

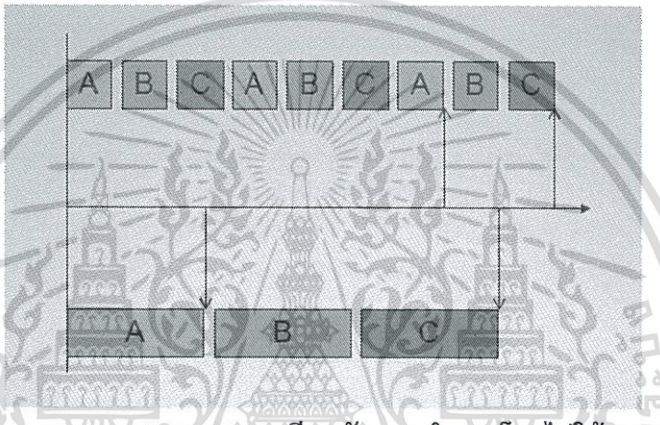


รูป 2.3 การทำงานประเภท Multiprogramming

จากรูปด้านบน จะสังเกตได้ว่ามี 3 โปรแกรมที่อยู่ใน Main memory แต่จะมีเพียงหนึ่งโปรแกรมเท่านั้นที่ถูกทำการ execution อยู่ ณ เวลาหนึ่ง ซึ่งวิธีการนี้จะทำให้ใช้ CPU ได้อย่างมีประสิทธิภาพมากขึ้น

2.10.2 Multitasking

เป็นเทคนิคที่ CPU จะถูกมอบหมายงานเป็นจำนวนมากกว่า 1 งานขึ้นไป โดยที่ CPU จะทำการสลับการทำงานไปเรื่อยๆ อย่างรวดเร็ว ทำให้เหมือนกับว่า ณ เวลาหนึ่งมีมากกว่า 1 หนึ่งโปรแกรมที่กำลังทำงานอยู่ ซึ่งวิธีนี้จะช่วยให้การใช้ CPU ได้มีประสิทธิภาพมากขึ้น และทำให้ใช้เวลาในการตอบสนองลดลง

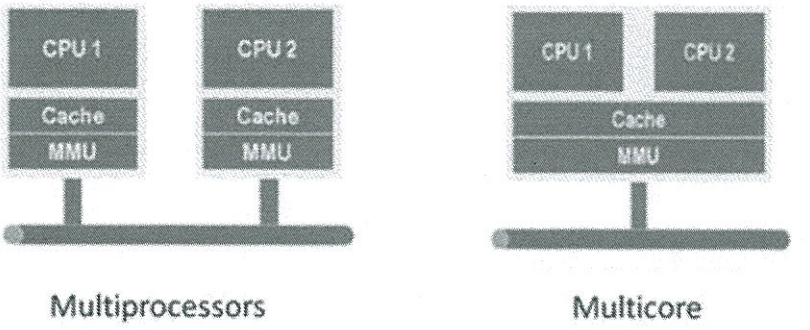


รูป 2.4 การทำงานของ multitasking เทียบกับการทำงานโดยไม่ใช้งาน multitasking

จากรูปจะสามารถอธิบายได้ว่า process ทั้งหมดจะถูกแบ่ง และทำงานด้วยระยะเวลาเท่าๆกัน สลับกันทำงานไปเรื่อยๆ

2.10.3 Multiprocessing

เป็นเทคนิคที่ใช้ processors มากกว่า 1 ตัวขึ้นไปในการทำงาน โดยเทคนิคนี้จะแบ่งงานให้กับ processor แต่ละตัวในการทำงาน ซึ่งหมายความว่า เราสามารถทำการประมวลผลแบบขนานได้ โดยที่ระบบปฏิบัติการจะทำหน้าที่เป็นตัวประสานการทำงานของซีพียูที่มากกว่าหนึ่งตัวนี้ให้ทำงานด้วยกันได้

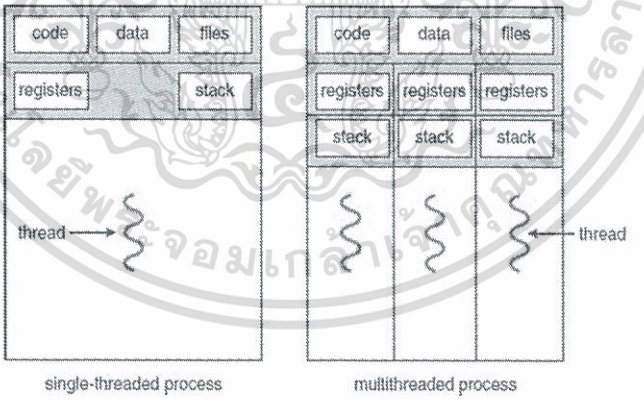


รูป 2.5 ความแตกต่างของ Multiprocessor และ Multicore

ความแตกต่างระหว่าง Multiprocessor และ Multicore จะสังเกตได้จากรูป 2.3 ก็คือ Multiprocessor จะเป็นการใช้หลาย processor ในการประมวณผล ส่วน Multicore จะเป็นการใช้ processor เพียงตัวเดียว แต่ภายใน processor นั้นจะประกอบด้วย CPU หลายตัว ตั้งแต่ 2 ตัวขึ้นไป จึงเรียกว่า Multicore

2.10.4 Multithreading

เป็นการนำ process มาแบ่งเป็นงานย่อยๆ (thread) โดยมีประโยชน์คือ เป็นการเรียกใช้ CPU ให้เกิดประโยชน์สูงสุด ทำให้การทำงานของโปรแกรมง่าย และมีประสิทธิภาพมากขึ้นและมีประโยชน์ ต่อระบบที่เป็น multicores เพราะสามารถเรียกใช้ threads หลายๆ ตัวได้พร้อมๆ กัน



รูป 2.6 การทำงานของ single-threaded process และ multithreaded process

บทที่ 3
การออกแบบและพัฒนา

3.1 ขอบเขตของโปรแกรมที่พัฒนา

1. โปรแกรมสามารถหารหัสผ่านของไฟล์ได้หลายนามสกุล เช่น ZIP, RAR, PDF
2. โปรแกรมสามารถเลือกรูปแบบวิธีการสำหรับการหารหัสผ่านได้
3. โปรแกรมสามารถใช้การประมวลผลแบบขนานเพื่อทำให้ประสิทธิภาพดีขึ้นได้

3.2 ข้อจำกัดของโปรแกรมที่พัฒนา

1. ไม่สามารถทำการหารหัสผ่านของไฟล์นามสกุล PDF ที่สร้างมาจากโปรแกรม Microsoft Office
2. ต้องทำการติดตั้งตัวแตกไฟล์นามสกุล RAR จึงจะสามารถหารหัสผ่านไฟล์นามสกุล RAR ได้
3. ไม่สนับสนุนไฟล์นามสกุล ZIP รูปแบบ multi-disk

3.3 เครื่องมือที่ใช้พัฒนา

3.3.1 สภาพแวดล้อมในการพัฒนา

- หน่วยประมวลผล AMD FX(tm) - 8350 Eight-Core Processor 4.00 GHz
- หน่วยความจำหลัก 16 GB
- ระบบปฏิบัติการ Windows 7 64-bit
- ฮาร์ดดิสก์ WDC WD1003FZEX-00MK2A0 ATA Device 1 TB
- กราฟฟิกการ์ด NVIDIA GeForce GT 95500GT

3.3.2 ซอฟแวร์ที่ใช้พัฒนา

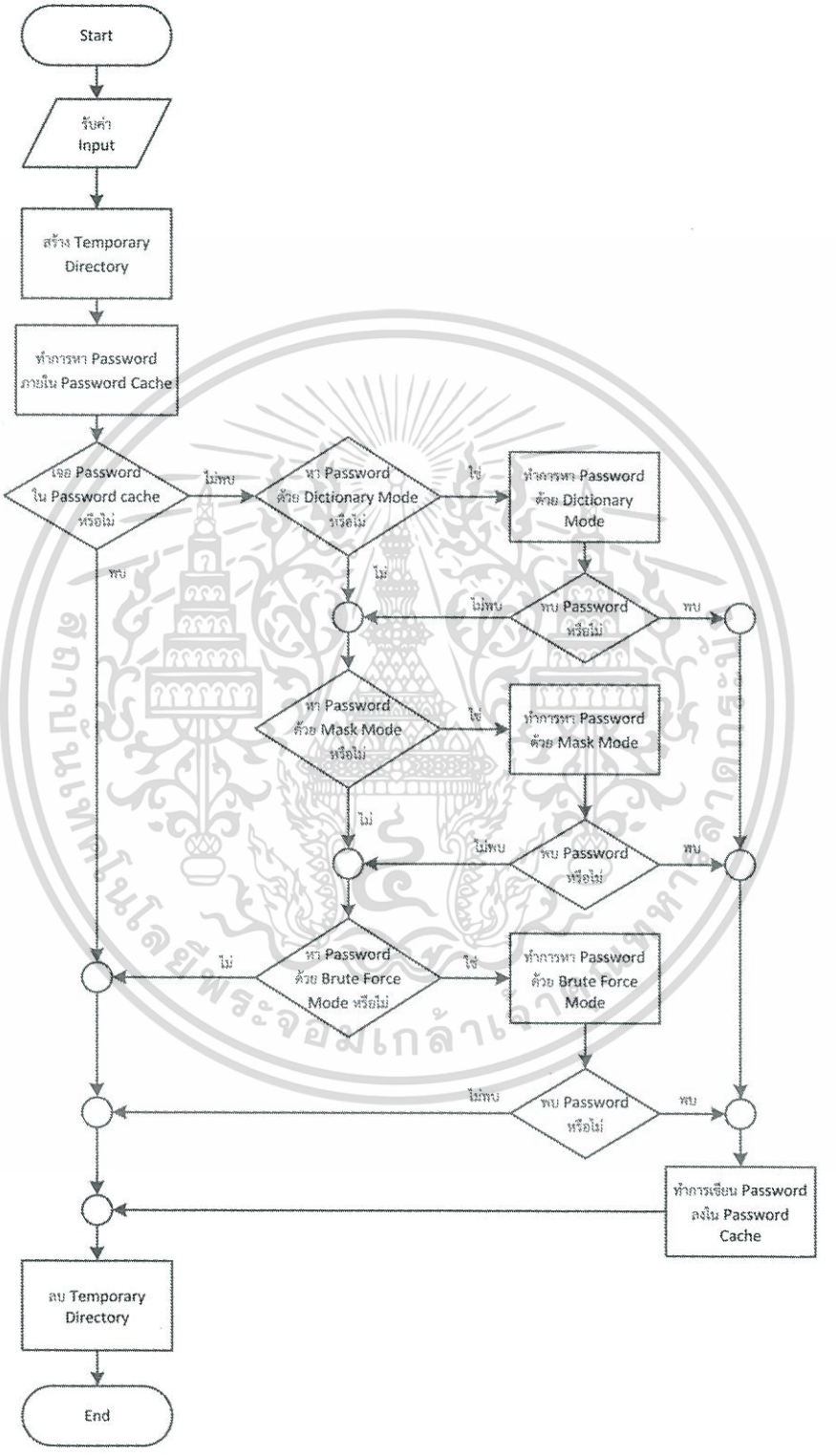
- Python version 2.7.8
- Pycharm community edition version 4.0.1
- unrar

3.3.3 ภาษาที่ใช้พัฒนา

- ภาษา Python

3.4 รูปแบบโครงสร้างของโปรแกรม

3.4.1 ภาพรวมของโปรแกรม



รูป 3.1 โครงสร้างของโปรแกรมโดยภาพรวม

3.4.2 ส่วนตรวจสอบชนิดของไฟล์

เป็นส่วนที่ทำหน้าที่ตรวจสอบชนิดของไฟล์ว่าเป็นไฟล์ที่โปรแกรมสนับสนุนหรือไม่ โดยการทำงานของส่วนนี้จะเป็นไปตามโปรแกรม 3.1 โดยฟังก์ชั่นจะทำการรับค่าที่อยู่ของไฟล์มา และนำที่อยู่ของไฟล์ไปทำการตรวจสอบตามฟังก์ชั่นของไลบรารี่เพื่อทำการตรวจสอบว่าเป็นไฟล์ที่ไลบรารี่รองรับ

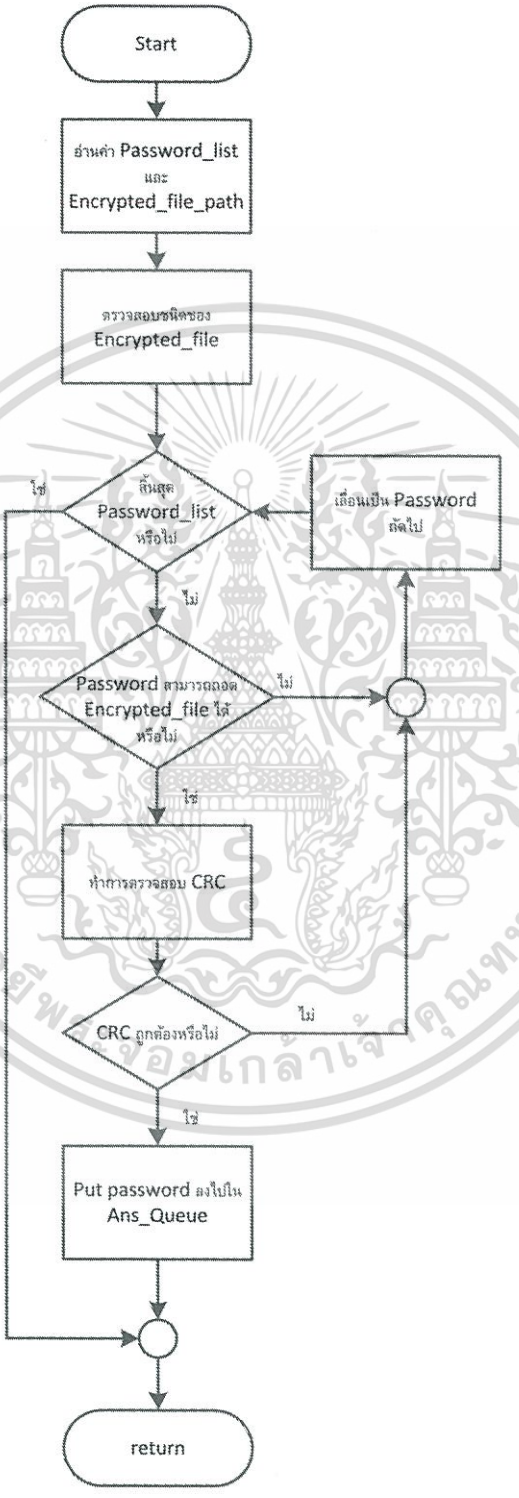
รูป 3.2 โครงสร้างในส่วนตรวจสอบชนิดไฟล์

โปรแกรม 3.1 ฟังก์ชั่น issupport

```
def issupport(file_dir):
    if zipfile.is_zipfile(file_dir) == True:
        return True
    elif rarfile.is_rarfile(file_dir) == True:
        return True
    else:
        try:
            PyPDF2.PdfFileReader(open(file_dir,"r+b"))
            return True
        except:
            return False
```

3.4.3 ส่วนค้นหารหัสลับ

ส่วนค้นหารหัสลับจะทำหน้าที่ทดสอบว่ารหัสที่ได้จากส่วนสร้างรหัสผ่านจากรูปแบบการโจมตีต่างๆทีละรหัส ซึ่งมีการทำงานตามโปรแกรมที่ 3.2



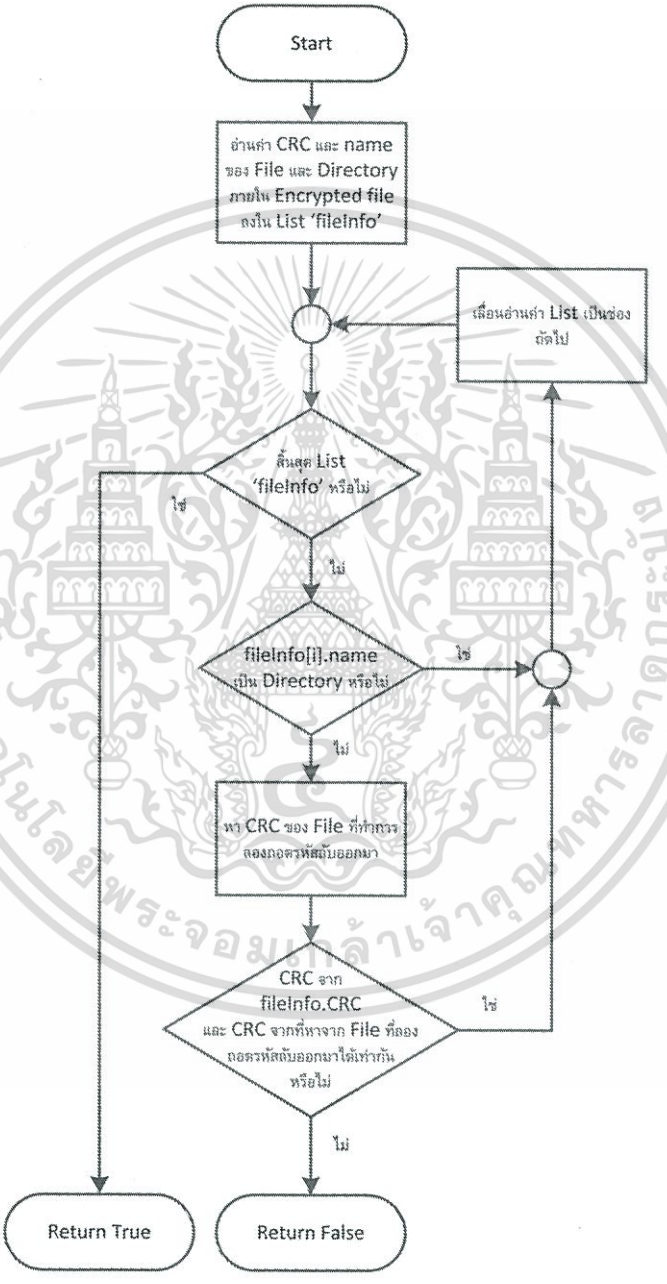
รูป 3.3 โครงสร้างในส่วนค้นหารหัสลับ

โปรแกรม 3.2 ตัวอย่างโค้ดในส่วนการหารหัสผ่านของไฟล์นามสกุล ZIP

```
        while start < stop:
            Try_pass = lines[start].strip('\n')
            try:
                My_File.extractall(path=temp_dir,
                                   pwd=Try_pass)
                if CRC.crc_checker(My_File,temp_dir):
                    Answer.put(Try_pass)
                    for i in other_Process:
                        os.kill(i,signal.SIGBREAK)
                    My_File.close()
                    return
                else:
                    start+=1
            except:
                start+=1
```

3.4.4 ส่วนตรวจสอบ CRC

ส่วนตรวจสอบ CRC ทำงานต่อจากส่วนค้นหารหัสลับเพื่อตรวจสอบความถูกต้องสมบูรณ์ ของไฟล์หลังจากทำการหารหัสลับ เนื่องจากในบางครั้งอาจจะเกิดความผิดพลาดโดยแจ้งว่ารหัสลับที่ค้นหานั้นถูกต้องทั้ง ๆ ที่ผิด (False Positive) โดยจะตรวจสอบค่า CRC32 ของไฟล์ก่อนและหลังการค้นหารหัสลับ ตามโปรแกรม 3.3



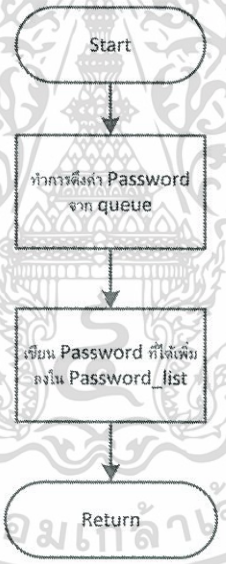
รูป 3.4 โครงสร้างในส่วนตรวจสอบ CRC

โปรแกรม 3.3 ฟังก์ชั่น crc

```
def crc(fileName):
    prev = 0
    if os.path.isfile(fileName):
        for eachLine in open(fileName,"rb"):
            prev = zlib.crc32(eachLine, prev)
        return prev & 0xFFFFFFFF
```

3.4.5 ส่วนสร้าง Password Cache

ส่วนเก็บ password ที่เคยค้นหาใน password cache ทำงานต่อจากส่วนค้นหารหัสผ่านเมื่อค้นพบรหัสลับที่ถูกต้องที่ไม่มีในไฟล์ password cache ก็จะทำการเก็บไว้ในไฟล์ password cache เพื่อใช้ในการค้นหาครั้งต่อไป ตามโปรแกรม 3.4 (หากระบบได้มีการสร้างไฟล์ password cache ไว้แล้ว ระบบจึงจะทำการค้นหารหัสผ่านภายใน password cache แต่ถ้าหากไม่มี ระบบจะทำการสร้างไฟล์ password cache ไว้ เพื่อการทำงานในครั้งถัดไป)



รูป 3.5 โครงสร้างในส่วน Password Cache

โปรแกรม 3.4 ฟังก์ชั่น writecache

```
def writecache(cache_path,password):
    file = open(cache_path,'a+')
    file.write(password)
    file.write("\n")
    file.close()
```

3.4.6 ส่วนสร้างชุดรหัสผ่านสำหรับทำการทดสอบด้วยวิธี Dictionary Attack

ส่วนสร้างชุดของรหัสผ่านที่จะทำการทดสอบโดยใช้ Dictionary Attack จะทำงานเมื่อถูกเรียกจากในส่วนการค้นหารหัสลับเพื่อที่จะทำการสร้างชุดรหัสลับที่จะทำการทดสอบโดยจะรับอินพุทเป็นที่อยู่ของไฟล์ Dictionary และทำการอ่าน String มาทั้งหมด และนำ string แต่ละบรรทัดที่ได้นำมาใส่ลงในเซทของรหัสผ่านที่จะทำการทดลอง ตามโปรแกรม 3.5

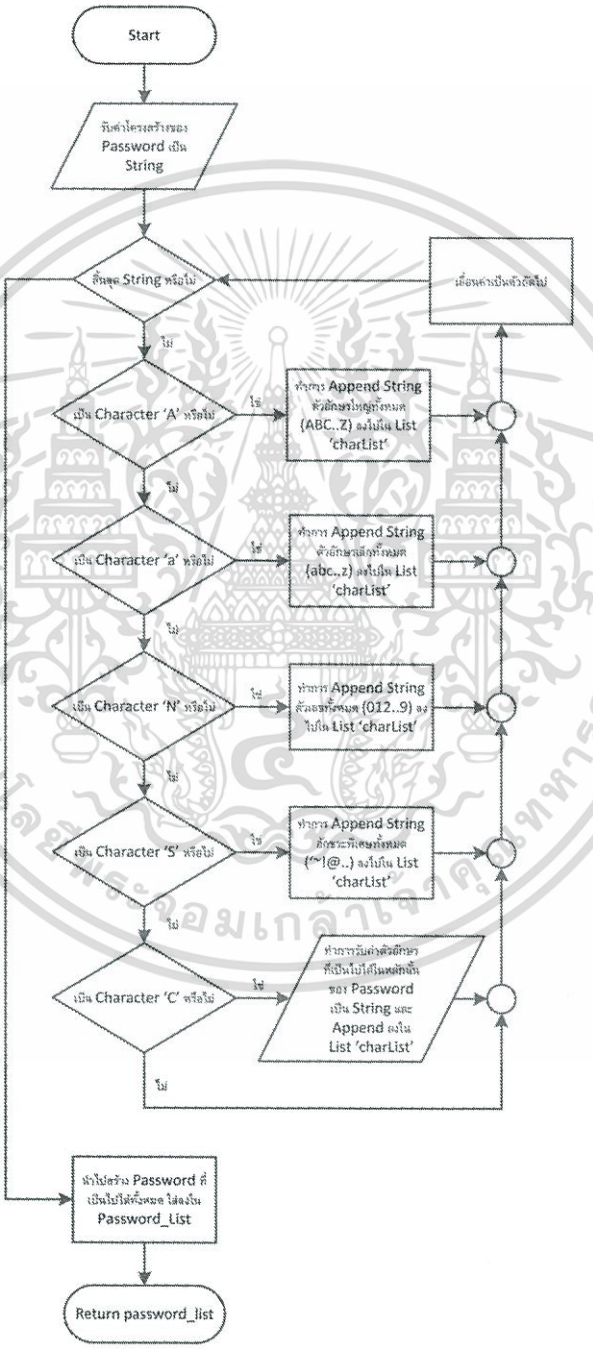
รูป 3.6 โครงสร้างในส่วนสร้างชุดรหัสผ่านด้วยวิธี Dictionary Attack

โปรแกรม 3.5 ฟังก์ชั่น create_dictlist

```
def create_dictlist(dict_dir):
    Dict = open(dict_dir)
    lines = Dict.readlines()
    Dict.close()
    return lines
```

3.4.1 ส่วนสร้างชุดรหัสผ่านสำหรับทำการทดสอบด้วยวิธี Mask Attack

ส่วนสร้างชุดของรหัสผ่านที่จะทำการทดสอบโดยใช้ Mask Attack โดยจะทำงานเมื่อถูกเรียกจากส่วนค้นหารหัสลับเพื่อที่จะทำการสร้างชุดรหัสลับเพื่อทำการทดสอบโดยจะรับอินพุทเป็นคำสั่งที่แสดงถึงเซทของตัวอักษรในแต่ละหลักของรหัสผ่าน ซึ่งจะทำการส่งต่อไปที่ฟังก์ชั่น create_set ภายในไฟล์ Mask.py ตามโปรแกรม 3.6 เพื่อนำไปสร้างเซทของรหัสผ่านทั้งหมด



รูป 3.7 โครงสร้างในส่วนสร้างชุดรหัสผ่านด้วยวิธี Mask Attack

โปรแกรม 3.6 ฟังก์ชั่น create_set

```
def create_set(command):
    passwordset = []
    for i in range(len(command)):
        if command[i] == 'A':
            passwordset.append(string.ascii_uppercase)
        elif command[i] == 'a':
            passwordset.append(string.ascii_lowercase)
        elif command[i] == 'S':
            passwordset.append(string.punctuation)
        elif command[i] == 'N':
            passwordset.append(string.digits)
        elif command[i] == 'C':
            bool_temp = False
            while(not bool_temp):
                try:
                    temp = raw_input("     + Enter custom
char for the %d bit : " % (i+1))
                    temp.encode('ascii')
                    passwordset.append(temp)
                    bool_temp = True
                except:
                    print "Invalid Input"
        else:
            print "Command Error"
    return passwordset
```

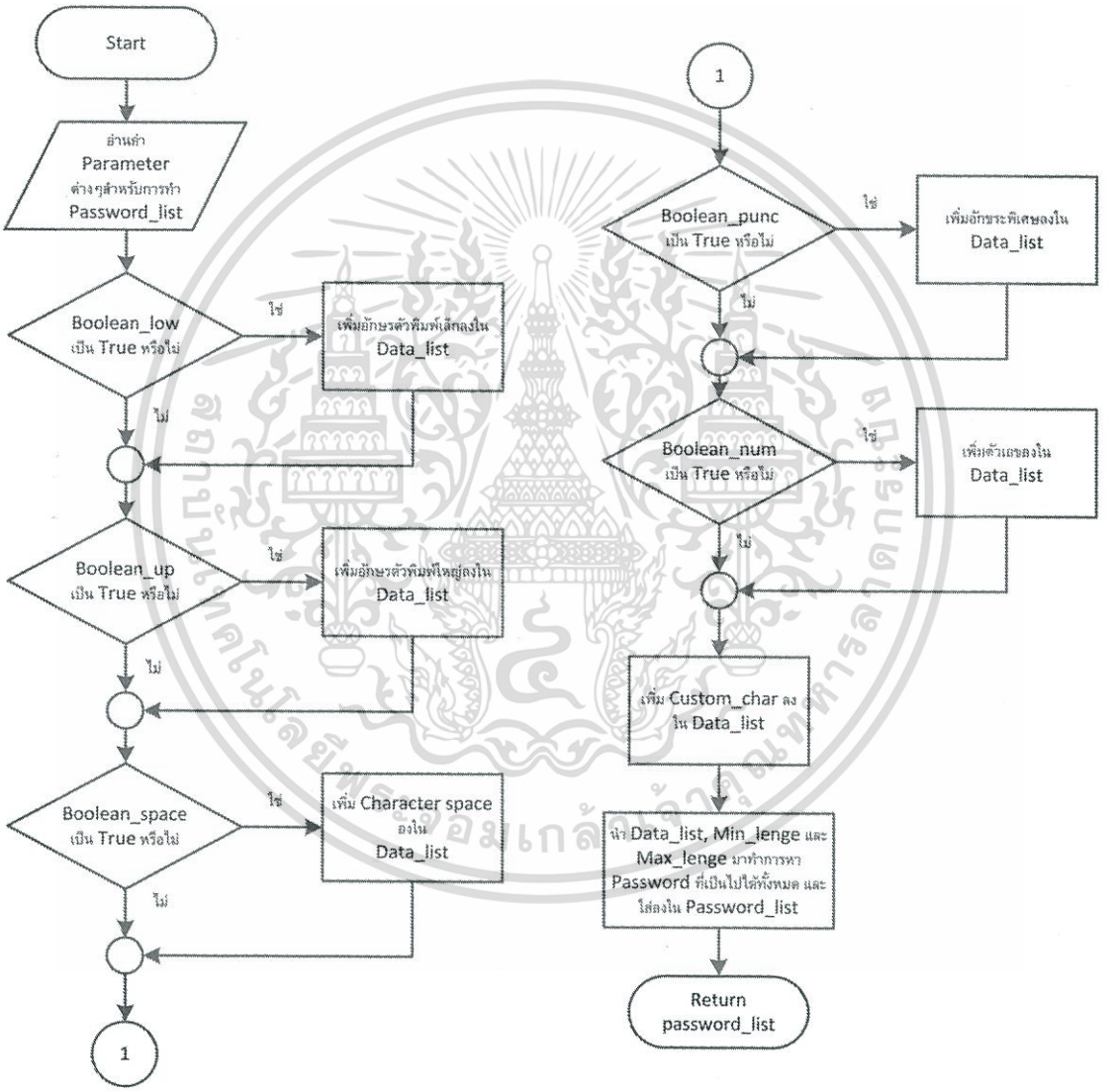
โปรแกรม 3.7 ฟังก์ชั่น create_masklist

```
def create_masklist(passwordset):
    wordlist = []
    temp = itertools.product(*passwordset)
    for i in temp:
        wordlist.append(''.join(i))

    return wordlist
```

3.4.2 ส่วนสร้างชุดรหัสผ่านสำหรับทำการทดสอบด้วยวิธี Brute Force Attack

ส่วนสร้างชุดของรหัสผ่านที่จะทำการทดสอบโดยใช้ Brute Force Attack จะทำงานเมื่อถูกเรียกจากส่วนของการค้นหารหัสลับ เพื่อที่จะทำการสร้างชุดรหัสลับที่จะใช้ในการทดสอบโดยจะรับอินพุทเป็นเซทของตัวอักษรที่มีความเป็นไปได้ที่จะอยู่ในรหัสลับและช่วงของความยาวของรหัสลับที่จะทำการทดสอบ โดยเซทของตัวอักษรจะถูกสร้างจากฟังก์ชั่น create_brutelist ภายในไฟล์ bruteforce.py ซึ่งจะมีรายละเอียดตามโปรแกรม 3.8



รูป 3.8 โครงสร้างในส่วนสร้างชุดรหัสผ่านด้วยวิธี Brute Force Attack

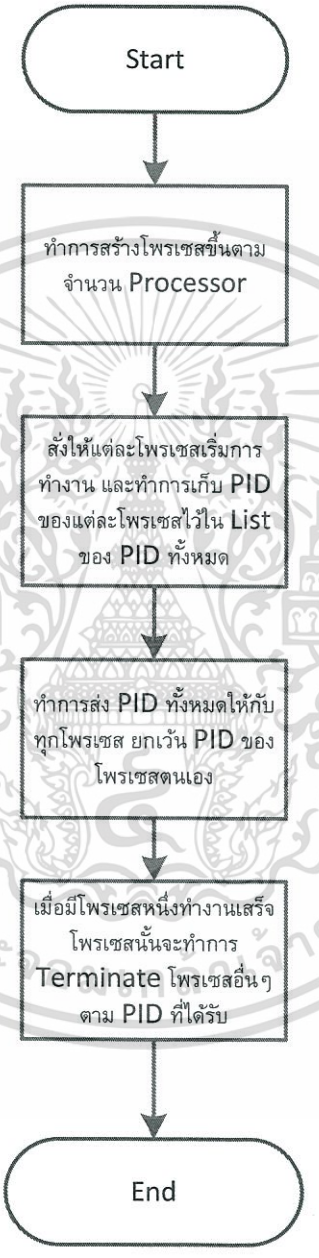
โปรแกรม 3.8 ฟังก์ชั่น create_brutelist

```
def create_brutelist
(Min,Max,bool_low,bool_up,bool_space,bool_punctuation,
bool_number,custom_char):
    #list for password list
    data_list = ""
    #lowwer case
    if bool_low == True:
        data_list = data_list + string.ascii_lowercase
    #upper case
    if bool_up == True:
        data_list = data_list + string.ascii_uppercase
    #space case
    if bool_space == True:
        data_list = data_list + " "
    #special character case
    if bool_punctuation == True:
        data_list = data_list + string.punctuation
    #number case
    if bool_number == True:
        data_list = data_list + string.digits
    #add custom char if any
    data_list = data_list + custom_char

    Max += 1
    word = []
    #each case in range Min,Max generate password long i
    for i in xrange(Min, Max):
        temp = itertools.product(data_list, repeat=i)
        #join character to word long = repeat
        for j in temp:
            word.append(''.join(j))
    return word
```

3.4.3 การสร้างโพรเซสสำหรับการทำ Multiprocessing

ในส่วนนี้ จะพูดถึงกระบวนการทำงานของส่วนสร้างโพรเซสเพื่อการทำ Multiprocessing ซึ่งสร้างโพรเซส รวมไปถึงการจัดการการทำงานต่างๆจะต้องมีกระบวนการทำที่เป็นขั้นเป็นตอน ซึ่งจะมีกระบวนการดังรูป



รูป 3.9 โครงสร้างในส่วนของการจัดการ Multiprocessing

โปรแกรม 3.9 ตัวอย่างโค้ดในส่วนของการจัดการ Multiprocessing

```
for i in range(Core):
    Start = ((lines/Core)*i)
    if i != Core-1:
        Stop = ((lines/Core)*(i+1))
    else:
        Stop = lines
    P = multiprocessing.Process(
                target=crackmask.Mask_Crack,
                args=(file_dir,tempdir[i],
                pidQ[i],pwd,Start,
                Stop,set_of_password))
    Work.append(P)


#Start Process
for i in range(Core):
    Work[i].start()
    all_PID.append(Work[i].pid)


#Put every PID to Queue
for i in range(Core):
    temp_pid = []
    for j in range(Core):
        if(i != j):
            temp_pid.append(all_PID[j])
        pidQ[i].put(temp_pid)


#join Process
for i in range(len(Work)):
    Work[i].join()
```

บทที่ 4
การทดลองและผลการทดลอง

4.1 คุณสมบัติของเครื่องคอมพิวเตอร์ที่ใช้ในการทดลอง

- หน่วยประมวลผล AMD FX(tm) - 8350 Eight-Core Processor 4.00 GHz
- หน่วยความจำหลัก 16 GB
- ระบบปฏิบัติการ Windows 7 64-bit
- ฮาร์ดดิสก์ WDC WD1003FZEX-00MK2A0 ATA Device 1 TB
- กราฟฟิกการ์ด NVIDIA GeForce GT 95500GT

4.2 การทดลองหารหัสผ่านของไฟล์นามสกุล ZIP

4.2.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

- **ตัวแปรต้น** ขนาดของไฟล์
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP
- **ตัวแปรควบคุม**

(a) จำนวนโพรเซสเซอร์ที่ใช้ในการประมวลผลคือ 1 โพรเซสเซอร์

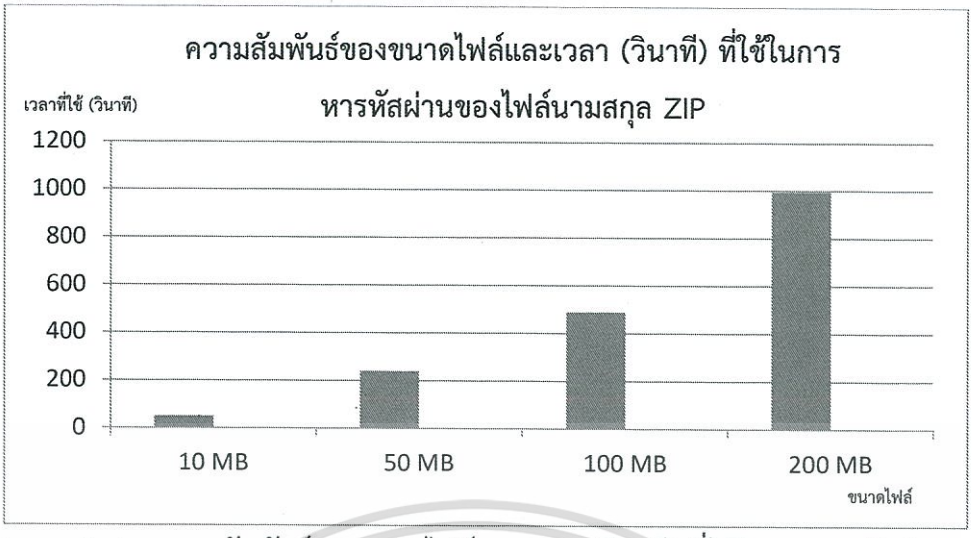
(b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000

- การทดลองนี้จะใช้ข้อมูลขนาดต่าง ๆ ซึ่งมีขนาดตั้งแต่ 1 MB ไปจนถึง 100 MB

4.2.1.1 ผลการทดลอง

ตาราง 4.1 ความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

ครั้งที่	10 MB	50 MB	100 MB	200 MB
1	50.53	240.20	493.33	996.29
2	50.07	240.47	492.39	995.09
3	50.34	240.71	480.68	995.07
เฉลี่ย	50.31	240.46	488.80	995.48



รูป 4.1 กราฟแสดงความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

4.2.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

- **ตัวแปรต้น** จำนวนโพรเซส
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP
- **ตัวแปรควบคุม**

(a) จำนวนโพรเซสที่ใช้คือ 8

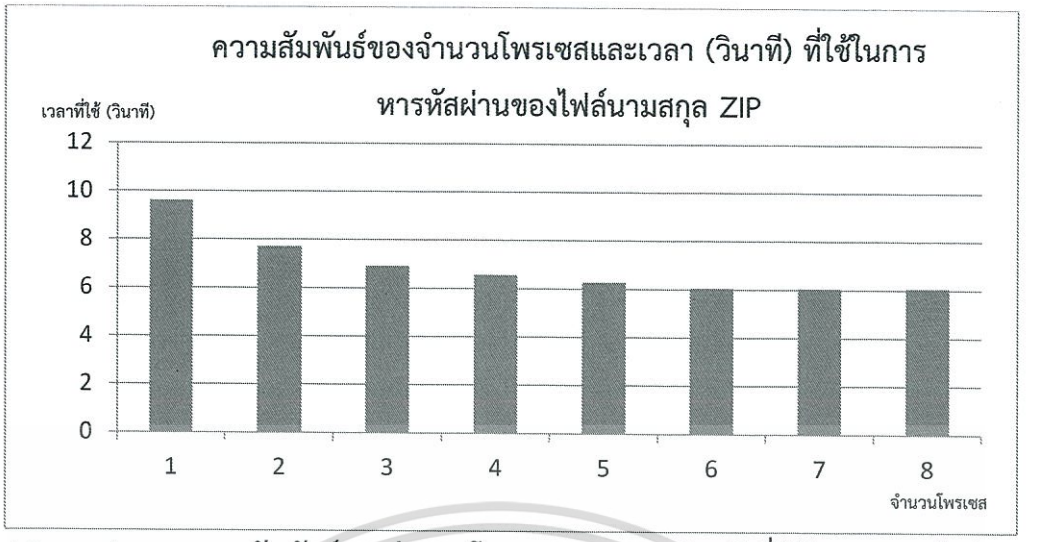
(b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000

- การทดลองนี้จะใช้โพรเซสตั้งแต่ 1 ไปจนถึง 8 โพรเซส

4.2.2.1 ผลการทดลอง

ตาราง 4.2 ความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

ครั้งที่	1 process	2 process	3 process	4 process	5 process	6 process	7 process	8 process
1	9.65	7.75	7.00	7.05	6.29	6.03	6.13	6.11
2	9.60	7.69	6.91	6.36	6.32	6.03	6.00	6.08
3	9.59	7.70	6.87	6.31	6.22	6.09	6.04	6.00
เฉลี่ย	9.61	7.71	6.92	6.573	6.27	6.05	6.05	6.06



รูป 4.2 กราฟแสดงความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

4.3 การทดลองหารหัสผ่านของไฟล์นามสกุล RAR

4.3.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

- **ตัวแปรต้น** ขนาดของไฟล์
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR
- **ตัวแปรควบคุม**

(a) จำนวนโพรเซสเซอร์ที่ใช้ในการประมวลผลคือ 1 โพรเซสเซอร์

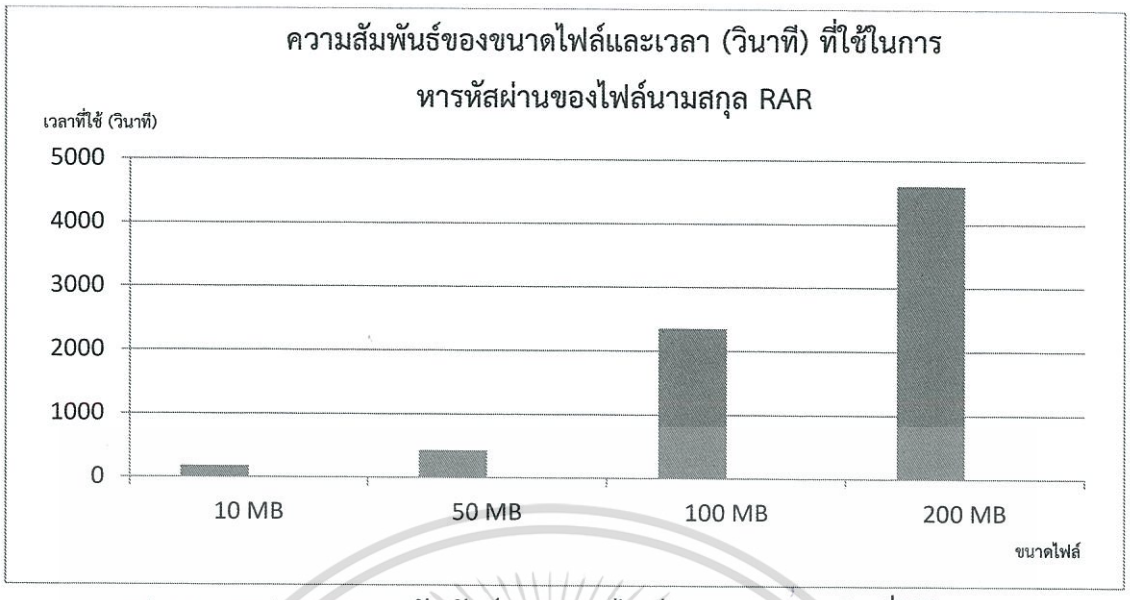
(b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000

- การทดลองนี้จะใช้ข้อมูลขนาดต่าง ๆ ซึ่งมีขนาดตั้งแต่ 10 MB ไปจนถึง 100 MB

4.3.1.2 ผลการทดลอง

ตาราง 4.3 ความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

ครั้งที่	10 MB	50 MB	100 MB	200 MB
1	177.58	430.047	2326.16	4622.33
2	177.13	429.382	2373.20	4585.64
3	177.53	429.348	2362.86	4604.44
เฉลี่ย	177.413	429.5923	2354.073	4604.137



รูป 4.3 กราฟแสดงความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

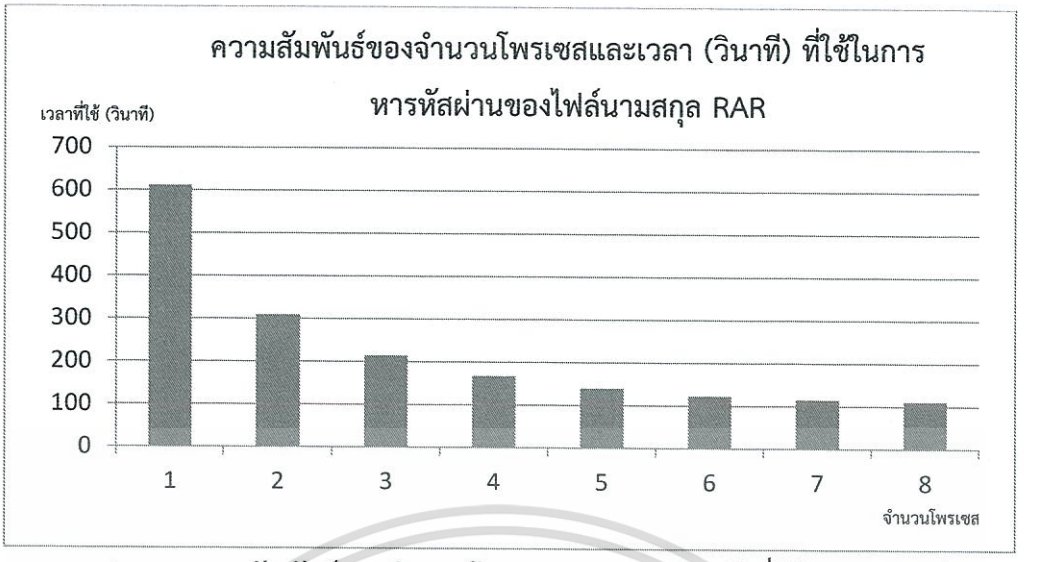
4.3.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

- **ตัวแปรต้น** จำนวนโพรเซส
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR
- **ตัวแปรควบคุม**
  (a) จำนวนโพรเซสที่ใช้คือ 8
  (b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000
- การทดลองนี้จะใช้โพรเซสตั้งแต่ 1 ไปจนถึง 8 โพรเซส

4.3.2.1 ผลการทดลอง

ตาราง 4.4 ความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

ครั้งที่	1 process	2 process	3 process	4 process	5 process	6 process	7 process	8 process
1	610.60	309.14	214.65	168.16	140.27	122.90	115.19	111.51
2	611.64	309.47	214.34	167.98	140.06	122.43	114.52	111.24
3	611.42	309.63	213.98	167.72	140.64	123.05	115.07	111.65
เฉลี่ย	611.22	309.41	214.32	167.95	140.32	122.79	114.92	111.46



รูป 4.4 กราฟแสดงความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

4.4 การทดลองหารหัสผ่านของไฟล์นามสกุล PDF

4.4.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

- **ตัวแปรต้น** ขนาดของไฟล์
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF
- **ตัวแปรควบคุม**

(a) จำนวนโพรเซสเซอร์ที่ใช้ในการประมวลผลคือ 1 โพรเซสเซอร์

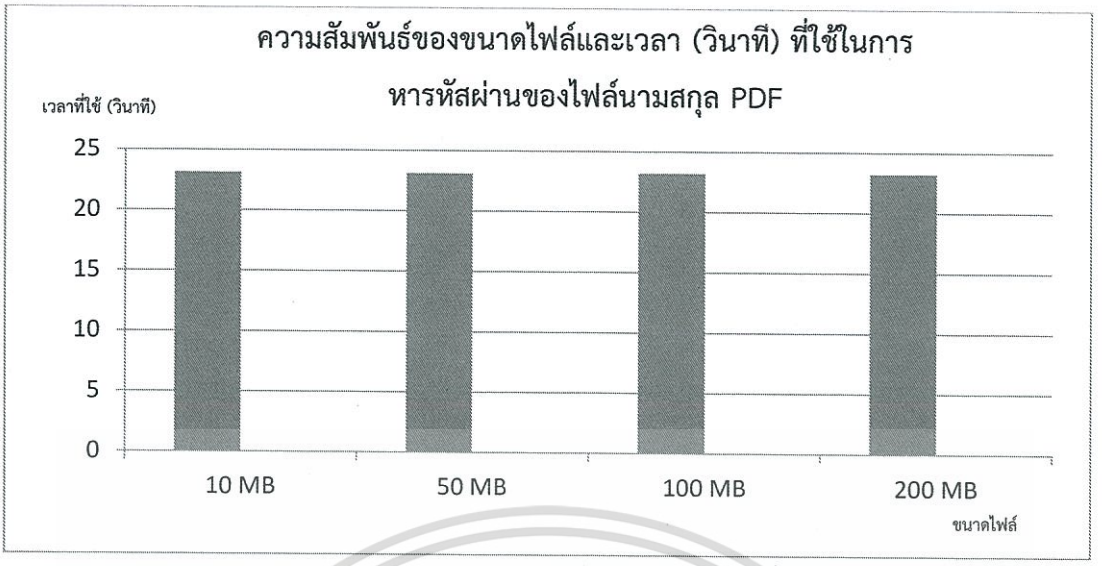
(b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000

- การทดลองนี้จะใช้ข้อมูลขนาดต่าง ๆ ซึ่งมีขนาดตั้งแต่ 1 MB ไปจนถึง 100 MB

4.4.1.1 ผลการทดลอง

ตาราง 4.5 ความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

ครั้งที่	10 MB	50 MB	100 MB	200 MB
1	23.09	23.08	23.24	23.25
2	23.16	23.11	23.19	23.14
3	23.15	23.15	23.10	23.23
เฉลี่ย	23.13	23.11	23.18	23.20



รูป 4.5 กราฟแสดงความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

4.4.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

- **ตัวแปรต้น** จำนวนโพรเซส
- **ตัวแปรตาม** เวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF
- **ตัวแปรควบคุม**

(a) จำนวนโพรเซสที่ใช้คือ 8

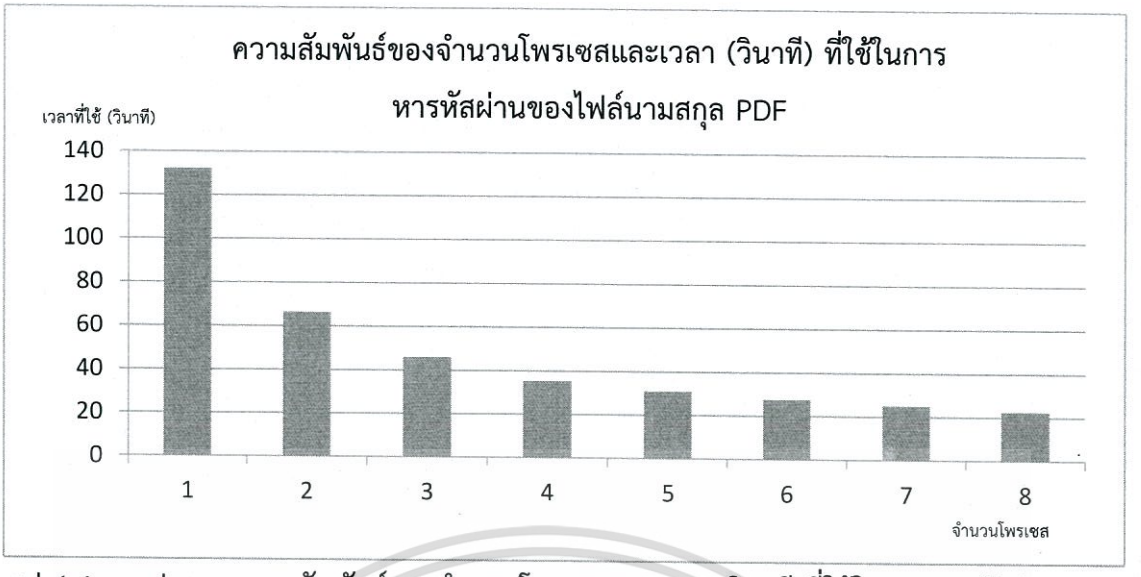
(b) ใช้รูปแบบการโจมตีแบบ Dictionary โดยรหัสผ่านที่ถูกต้องอยู่ในบรรทัดที่ 10000

- การทดลองนี้จะใช้โพรเซสตั้งแต่ 1 ไปจนถึง 8 โพรเซส

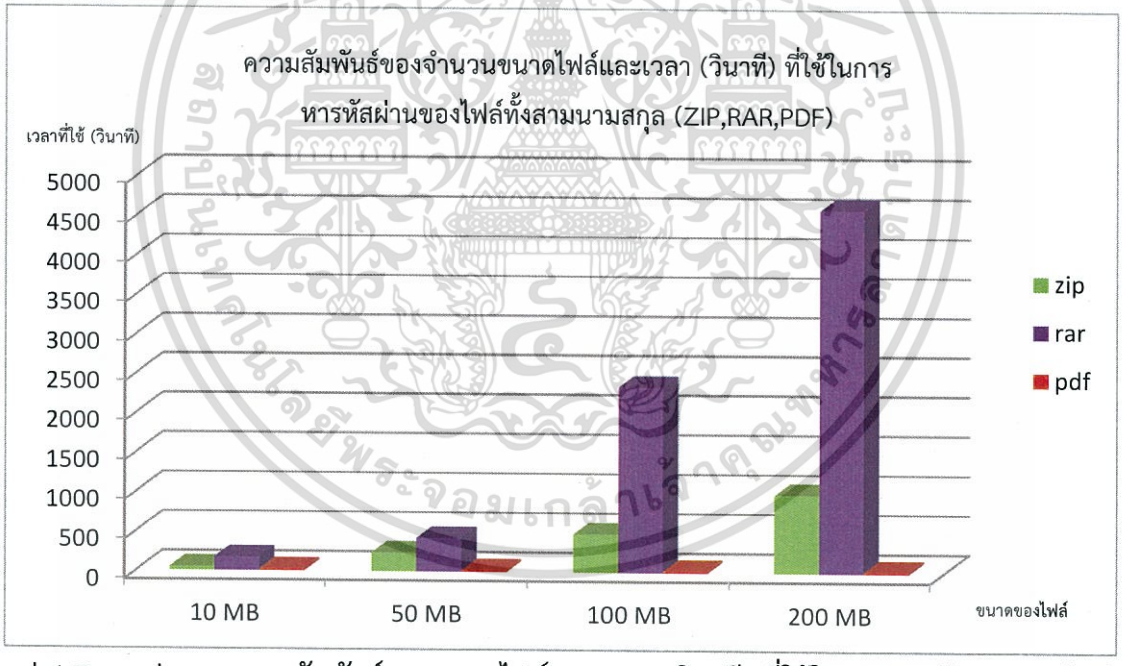
4.4.2.1 ผลการทดลอง

ตาราง 4.6 ความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

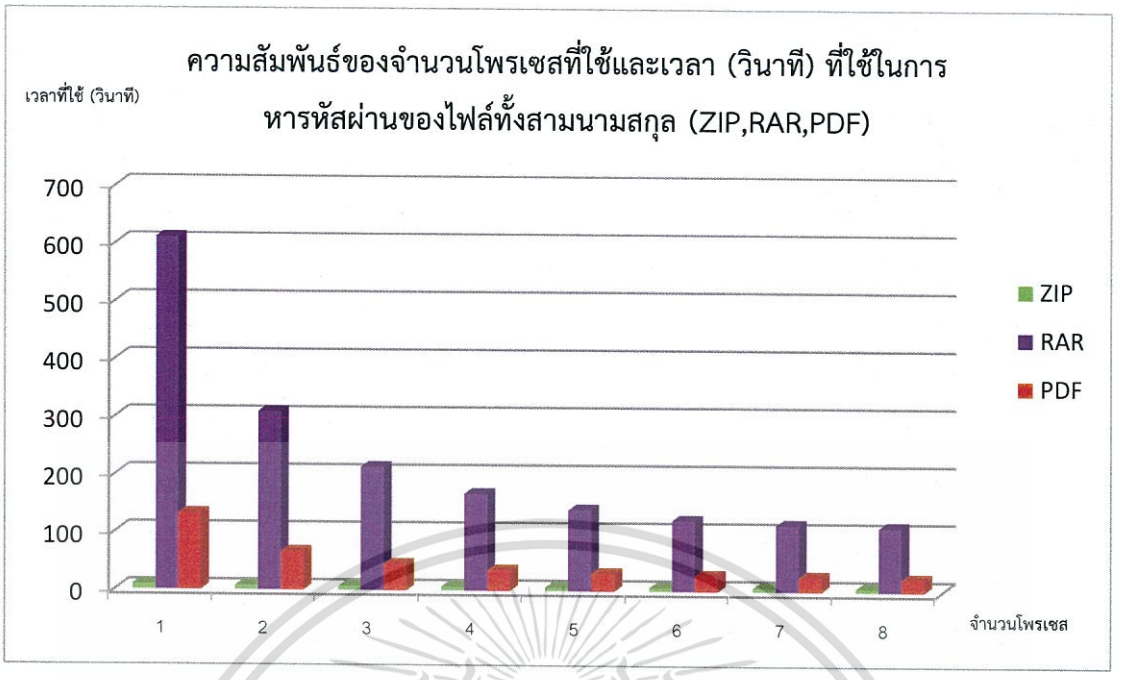
ครั้งที่	1 process	2 process	3 process	4 process	5 process	6 process	7 process	8 process
1	132.14	66.67	46.78	36.58	31.44	27.95	26.13	23.29
2	132.19	66.53	45.84	34.44	30.67	27.71	24.84	22.27
3	132.25	66.53	45.68	35.63	31.06	27.55	24.47	23.04
เฉลี่ย	132.19	66.58	46.1	35.55	31.05	27.73	25.15	22.87



รูป 4.6 กราฟแสดงความสัมพันธ์ของจำนวนโพรเซสและเวลา(วินาที)ที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF



รูป 4.7 กราฟแสดงความสัมพันธ์ของขนาดไฟล์และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์ทั้งสามนามสกุล (ZIP, RAR, PDF)



รูป 4.8 กราฟแสดงความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลา (วินาที) ที่ใช้ในการหารหัสผ่านของไฟล์ทั้งสามนามสกุล (ZIP, RAR, PDF)

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุปของโครงงาน

จากการทดลองในบทที่ 4 เราจะสามารถสรุปการทดลองได้ว่า

5.1.1 การทดลองหารหัสผ่านของไฟล์นามสกุล ZIP

5.1.1.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

จากการทดลองหารหัสผ่านของไฟล์นามสกุล ZIP โดยการจับเวลาที่ใช้ในการหารหัสผ่านของไฟล์ที่มีขนาดแตกต่างกัน ซึ่งจะได้ผลการทดลองตามตารางที่ 4.1 และจะสามารถสังเกตได้ว่า เวลาที่ใช้ในการทำการหารหัสผ่านจะมีความสัมพันธ์กับขนาดของไฟล์มีลักษณะใกล้เคียงกับสมการเส้นตรง โดยเวลาที่ใช้ในการหารหัสต่อไฟล์ขนาด 10 MB จะใช้เวลาประมาณ 50 วินาที และไฟล์ขนาด 200 MB จะใช้เวลาประมาณ 990 วินาที

5.1.1.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล ZIP

จากการทดลองหารหัสผ่านของไฟล์นามสกุล ZIP โดยการจับเวลาที่ใช้ในการหารหัสผ่าน ด้วยโพรเซสที่มีจำนวนแตกต่างกัน ตั้งแต่ 1 ถึง 8 โพรเซส ซึ่งจะได้ผลการทดลองตามตารางที่ 4.2 และจะสามารถสังเกตได้ว่า เวลาที่ใช้ในการหารหัสผ่านของไฟล์จะมีค่าแปรผกผันกับจำนวนของโพรเซสที่ใช้ และมีค่าลดลงเรื่อยๆ ตั้งแต่การใช้เวลาประมาณ 9 ถึง 10 วินาที ที่การใช้งานโพรเซส 1 โพรเซส ไปจนถึงใช้เวลาในการหารหัสผ่านประมาณ 6 วินาที ที่การใช้โพรเซสตั้งแต่ 6 ถึง 8 โพรเซส

5.1.2 การทดลองหารหัสผ่านของไฟล์นามสกุล PDF

5.1.2.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

จากการทดลองหารหัสผ่านของไฟล์นามสกุล PDF โดยการจับเวลาที่ใช้ในการหารหัสผ่านของไฟล์ที่มีขนาดแตกต่างกัน ซึ่งจะได้ผลการทดลองตามตารางที่ 4.5 และจะสามารถสังเกตได้ว่า เวลาที่ใช้ในการทำการหารหัสผ่านแทบจะไม่มีความสัมพันธ์กับขนาดของไฟล์เลย โดยที่เวลาที่ใช้ในการหารหัสผ่านของไฟล์ pdf จะคงที่ตลอด โดยการทดสอบจะใช้เวลา 0.0024 วินาทีต่อหนึ่งรหัสผ่าน

5.1.2.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล PDF

จากการทดลองหารหัสผ่านของไฟล์นามสกุล PDF โดยการจับเวลาที่ใช้ในการหารหัสผ่าน ด้วยโพรเซสที่มีจำนวนแตกต่างกัน ตั้งแต่ 1 ถึง 8 โพรเซส ซึ่งจะได้ผลการทดลองตามตารางที่ 4.6 และจะสามารถสังเกตได้ว่าเวลาที่ใช้จะมีค่าแปรผกผันกันกับจำนวนโพรเซสที่ใช้ โดยจะมีระยะเวลาที่ใช้ตั้งแต่ประมาณ 132 วินาที ที่มีการใช้งานโพรเซส 1 โพรเซส ไปจนถึงประมาณ 22 ถึง 23 วินาที ที่การใช้โพรเซสทั้งหมด 8 โพรเซส

5.1.3 การทดลองหารหัสผ่านของไฟล์นามสกุล RAR

5.1.3.1 การทดลองเพื่อหาความสัมพันธ์ของขนาดของไฟล์และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

จากการทดลองหารหัสผ่านของไฟล์นามสกุล PDF โดยการจับเวลาที่ใช้ในการหารหัสผ่านของไฟล์ที่มีขนาดแตกต่างกัน ซึ่งจะได้ผลการทดลองตามตารางที่ 4.3 และจะสามารถสังเกตได้ว่า เวลาที่ใช้ในการหารหัสผ่านของไฟล์ จะมีค่าแปรผันตามขนาดของไฟล์ เมื่อไฟล์ที่นำมาทดลองมีขนาดใหญ่ขึ้นเวลาที่ใช้ก็จะมากขึ้น ซึ่งการความสัมพันธ์ของเวลาที่ใช้ และขนาดของไฟล์จะมีความสัมพันธ์ที่ไม่สามารถบอกได้ว่าเพิ่มขึ้นแบบใด ซึ่งต้องทำการทดลองแบบละเอียดจึงจะสามารถหาความสัมพันธ์ของทั้งสองตัวแปรได้ โดยที่ ระยะเวลาที่ใช้ในการหารหัสผ่าน จะใช้เวลาตั้งแต่ 177 ถึง 178 วินาทีในการหารหัสผ่านของไฟล์ขนาด 10 MB ไปจนถึงใช้เวลาทั้งหมดประมาณ 4,600 วินาทีในการหารหัสผ่านของไฟล์ขนาด 200 MB

5.1.3.2 การทดลองเพื่อหาความสัมพันธ์ของจำนวนโพรเซสที่ใช้และเวลาที่ใช้ในการหารหัสผ่านของไฟล์นามสกุล RAR

จากการทดลองหารหัสผ่านของไฟล์นามสกุล RAR โดยการจับเวลาที่ใช้ในการหารหัสผ่าน ด้วยโพรเซสที่มีจำนวนแตกต่างกัน ตั้งแต่ 1 ถึง 8 โพรเซส ซึ่งจะได้ผลการทดลองตามตารางที่ 4.4 และจะสามารถสังเกตได้ว่า เวลาที่ใช้ในการหารหัสผ่าน จะมีค่าแปรผกผันกับจำนวนโพรเซสที่ใช้ โดยจะใช้เวลาตั้งแต่ประมาณ 610 ถึง 612 วินาทีจากการใช้โพรเซส 1 โพรเซส ไปจนถึงใช้เวลาประมาณ 111 ถึง 112 วินาที จากการใช้โพรเซสทั้งหมด 8 โพรเซส

5.2 ปัญหาและอุปสรรค

1. การพัฒนาระบบมีการพัฒนาด้วยภาษาไพทอน (Python) ซึ่งเป็นภาษาที่ใหม่ จึงต้องใช้เวลาในการเรียนรู้และพัฒนา
2. การค้นหา Library ที่เหมาะสมกับการนำมาใช้งานได้ยาก
3. แต่ละ Library จำเป็นจะต้องใช้เวลาในการศึกษาไม่มากก็น้อย
4. ต้องใช้เวลาอย่างมากในการหาและทดลอง Library ที่เหมาะกับการนำมาใช้งานกับโครงงาน
5. ในการเขียนโปรแกรมในส่วน Multiprocessing มีรายละเอียดมาก ทำให้ต้องเสียเวลาในการศึกษา และลองผิดลองถูก เพื่อหาวิธีที่เหมาะสมในการใช้งาน Multiprocessing
6. การทดสอบในแต่ละครั้งใช้เวลานาน

5.3 แนวทางการแก้ไขและพัฒนา

1. พัฒนารูปแบบการทดสอบรหัสผ่านของแต่ละนามสกุลให้ดียิ่งขึ้น
2. พัฒนาให้โปรแกรมสามารถใช้กราฟฟิกการ์ดในการประมวลผลได้
3. พัฒนาเพิ่มในส่วนของ ส่วนประสานงานกับผู้ใช้ (User Interface)
4. พัฒนาให้โปรแกรมสามารถค้นหารหัสผ่านของไฟล์ที่ถูกเข้ารหัสลับได้หลายนามสกุลมากขึ้น
5. พัฒนาให้โปรแกรมสามารถใช้การประมวลผลแบบ Clustering ได้
6. พัฒนาให้โปรแกรมใช้ทรัพยากรน้อยลง

บรรณานุกรม

Chrysanthou Yiannis. 2013. **Modern Password Cracking:A hands-on approach to creating an optimised and versatile attack.**

[Online].Available: http://www.ma.rhul.ac.uk/static/techrep/2013/MA 2013-07.pdf.

Sam Martin and Mark Tokutomi. **Password Cracking**

[Online].Available: http://www.cs.arizona.edu/~collberg/Teaching/466-566/2012/Resources/presentations/2012/topic7-final/report.pdf

United States National Institute of Standards and Technology (NIST). 2001. **Announcing the ADVANCED ENCRYPTION STANDARD (AES)**

[Online].Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

PKWARE Inc. **ZIP File Format Specification**

[Online].Available: https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT

Simson Garfinkel. **RAR File Format**

[Online].Available: http://www.forensicswiki.org/wiki/RAR

Dejan Lukan. **PDF File Format**

[Online].Available: http://resources.infosecinstitute.com/pdf-file-format-basic-structure/

Jeff Knupp, **Python's Hardest Problem**

[Online].Available:http://www.jeffknupp.com/blog/2012/03/31/pythons-hardest-problem/

Thread safety

[Online].Available: http://en.wikipedia.org/wiki/Thread_safety

Python (programming language)

[Online].Available:

http://en.wikipedia.org/wiki/Python_%28programming_language%29

Race condition definition

[Online].Available:

http://searchstorage.techtarget.com/definition/race-condition