การโจมตีคีย์ของอาเอฟไอดีด้วยคูด้ากราฟิกโพรเซสเซอร์

Attacking RFID Key Using CUDA Graphics Processors

มาลีวัลย์ แซ่ก๊วย ปกรณ์ วัฒนจตุรพร

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทคัดย่อ

ความปลอดภัยในการรักษาความลับของข้อมูลและการยืนยันตนในอาเอฟไอดีขึ้นอยู่กับอัลกอริทึมการเข้ารหัสลับในระบบ ความปลอดภัยของระบบอาเอฟไอดีไมแฟร์คลาสสิคซึ่งเป็นระบบที่ได้รับความนิยมและใช้งานอย่างแพร่หลายมาจากการปกปิดกระบวนการเข้ารหัส การโจมตีเพื่อค้นหาคีย์ที่ถูกต้องขนาด 48 บิตด้วยการบูทฟอท (Brute force) จึงเป็นไปได้ยากและใช้เวลานาน ในปัจจุบันนี้ มีการนำกราฟิกโพรเซสเซอร์หรือจีพียู (GPU: Graphics Processing Unit) มาใช้เพิ่มประสิทธิภาพการประมวลผลแบบขนาน บทความนี้นำเสนอผลการเปรียบเทียบการโจมตีเพื่อค้นหาคีย์ที่ถูกต้องด้วยการใช้ซีพียูและด้วยการใช้จีพียูเพื่อทำการโจมตีแบบบูทฟอทและการโจมตีโดยอาศัยจุดอ่อนของอัลกอริทึม ผลการทดลองแสดงให้เห็นถึงประสิทธิภาพการโจมตีด้วยวิธีบูทฟอทเพิ่มขึ้น 51.47 เท่าและโดยอาศัยจุดอ่อนของอัลกอริทึมเพิ่มขึ้น 76.01 เท่า

คำสำคัญ : กราฟิกโพรเซสเซอร์ อัลกอริทึมคริปโตวัน อาเอฟไอดี

Abstract

Security of an RFID data confidentiality and user authentication depends on a cryptography algorithm used in the system. The security of the well-known Mifare Classic RFID is based on the closure of its cryptography algorithm. Traditional brute-force attack on its 48 bits key was once impractical. Recently, Graphics Processing Unit (GPU) based computing has been employed to parallel processing and increase traditional processing performance. This paper presents comparative results of using CPU and using GPU to find a correct key using a brute force attack and using an attack on the cryptography algorithm. The experimental results clearly show the improvement of the attack performance by the magnitude of 51.47 when using the brute force method and 76.01 when attacking on the cryptography algorithm.

Keywords : Graphics Processing Unit, Crypto-1 algorithm, RFID

1. บทนำ

เทคโนโลยีอาเอฟไอดี (RFID: Radio Frequency Identification) เป็นระบบที่ถูกนำมาใช้ในกิจการต่าง ๆ อย่างแพร่หลาย เช่น ระบบควบคุมการเข้าออกอาคาร บัตรโดยสารรถไฟใต้ดิน เกษตรกรรม ปศุสัตว์ และการขนส่งสินค้า เป็นต้น อาเอฟไอดีของไมแฟร์คลาสสิค (Mifare Classic) เป็นหนึ่งในอาเอฟไอดีที่ได้รับความนิยมนำมาใช้เป็นจำนวนมาก จึงเป็นเหตุผลที่มีการศึกษาวิจัยจำนวนมากถึงประเด็นความปลอดภัยในการเข้ารหัสของอาเอฟไอดีไมแฟร์คลาสสิค

กระบวนการเข้ารหัสของไมแฟร์คลาสสิคเป็นอัลกอริทึมเฉพาะของบริษัทผู้ผลิตและไม่มีการเผยแพร่

ข้อมูล ทว่าในปี ค.ศ .2007 Nohl และ Plötz [1] เผยแพร่ความอ่อนแอของอัลกอริทึมคริปโตวัน (Crypto 1) ที่ใช้ในขั้นตอนการยืนยันตนของอาเอฟไอดีของไมแฟร์คลาสสิค จากนั้นจึงมีงานวิจัยจำนวนมากที่มีจุดประสงค์เพื่อศึกษาวิธีโจมตีการเข้ารหัสของอัลกอริทึมคริปโตวันด้วยหลากหลายวิธีการรวมถึงการใช้เอฟพีจีเอ (FPGA: Field-Programmable Gate Array) ช่วยในการประมวลผล นอกจากการใช้เอฟพีจีเอแล้ว ปัจจุบันนี้มีแนวคิดในการนำจีพียู มาช่วยในการประมวลผล ซึ่งโครงสร้างของจีพียูมีความเหมาะสมสำหรับการแก้ปัญหาการคำนวณข้อมูลที่ต้องการการประมวลผลคำสั่งเดียวกันแต่มีข้อมูลต่างชุดกัน ทำให้การคำนวณแต่ละส่วนสามารถประมวลผลแบบขนานได้

บทความนี้จึงนำเสนอผลการประยุกต์ใช้จีพียูเพื่อเพิ่มประสิทธิภาพการประมวลผลการโจมตีอัลกอริทึมคริปโตวัน โดยเลือกใช้การประมวลผลบนสถาปัตยกรรมคูด้า (CUDA: Compute Unified Device Architecture) [2] ซึ่งเป็นเทคโนโลยีที่ได้รับการพัฒนาโดยบริษัทเอ็นวีเดีย (NVIDIA) สำหรับการใช้จีพียูเพื่อการทำงานแบบขนานมาใช้ทดสอบการโจมตีอัลกอริทึมคริปโตวันในครั้งนี้

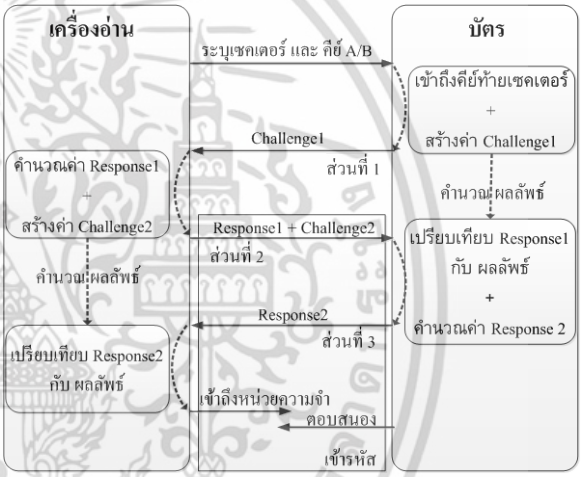
บทความนี้นำเสนอทฤษฎีที่เกี่ยวข้อง การทดลอง และผลการทดลองตามลำดับต่อไปนี้ ส่วนที่ 2 อธิบายขั้นตอนการทำงานและขั้นตอนการยืนยันตนของอาเอฟไอดีไมแฟร์ ส่วนที่ 3 อธิบายคุณลักษณะของคูด้าจีพียู ส่วนที่ 4 อธิบายการทดลอง ข้อมูลที่ใช้ทดสอบ และสภาพแวดล้อมของการทดสอบ ผลการทดลองและสรุปผลการทดลองแสดงไว้ในส่วนที่ 5 และส่วนที่ 6 ตามลำดับ

2. อาเอฟไอดีไมแฟร์

อาเอฟไอดีของไมแฟร์คลาสสิค (Mifare Classic) เป็นอาเอฟไอดีที่ได้รับความนิยมนำมาใช้เป็นจำนวนมาก ดังนั้นนักพัฒนาและวิจัยจำนวนมากจึงศึกษาประเด็นความปลอดภัยในการเข้ารหัสของอาเอฟไอดีไมแฟร์คลาสสิค ซึ่งมีชื่อว่าคริปโตวัน กระบวนการทำงานของคริปโตวันนั้นไม่ได้รับการเปิดเผยจากผู้ผลิต การทำงานและกระบวนการโจมตีอัลกอริทึมนั้นจึงได้จากการสังเกตและวิเคราะห์การติดต่อระหว่างเครื่องอ่านและอาเอฟไอดี รายละเอียดการวิเคราะห์ สันนิษฐานการทำงานของอัลกอริทึม และการโจมตีแสดงไว้ใน [3–8]

2.1 ขั้นตอนการยืนยันตนของอาเอฟไอดีไมแฟร์

ขั้นตอนการยืนยันตนโดยสังเขปของอาเอฟไอดีไมแฟร์ เริ่มเมื่อเครื่องอ่านส่งคำร้องขอการอ่านหรือเขียนข้อมูลไปยังบัตรโดยระบุตำแหน่งเซคเตอร์ที่ต้องการเข้าถึง พร้อมทั้งระบุคีย์ที่ใช้ไปยังบัตร (คีย์ A หรือ คีย์ B) บัตรจะทำการเข้าถึงอีอีพีรอม (EEPROM) และอ่านค่าคีย์ที่อยู่ในท้ายเซคเตอร์ที่เครื่องอ่านระบุมา จากนั้นจะเริ่มขั้นตอนการยืนยันตนระหว่างเครื่องอ่านและบัตรซึ่งมีทั้งหมด 3 ขั้นตอนตามมาตรฐาน ISO 9798-2 ตามรูปที่ 1



รูปที่ 1 ขั้นตอนการยืนยันตนของอาเอฟไอดีไมแฟร์

(1) บัตรจะทำการสร้างค่าขนาด 32 บิต เรียกว่า Challenge 1 ส่งไปยังเครื่องอ่าน

(2) เครื่องอ่านจะนำค่า Challenge1 ที่ได้รับมาคำนวณร่วมกับคีย์ 48 บิตและเลขระบุตัวตนของบัตร 32 บิตที่เครื่องอ่านบันทึกไว้ก่อนขั้นตอนการยืนยันตน ผลลัพธ์ที่ได้เรียกว่า Response 1 นอกจากนั้นแล้วเครื่องอ่านยังสร้างค่าขนาด 32 บิตเรียกว่า Challenge2 เครื่องอ่านจะทำการเข้ารหัส Response1 และเข้ารหัส Challenge2 จากนั้นจึงส่งค่าที่ถูกเข้ารหัสทั้งสองค่าไปยังบัตร

(3) บัตรจะนำค่า Challenge1 ที่มีในขั้นตอนที่ 1 มาทำการคำนวณด้วยขั้นตอนเหมือนที่เครื่องอ่านคำนวณแล้วนำค่าที่คำนวณได้มาเปรียบเทียบกับค่า

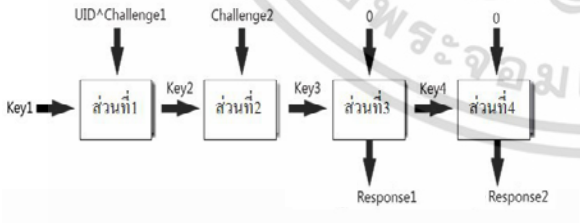
Response1 ที่ถูกเข้ารหัสที่ได้รับจากเครื่องอ่าน ถ้าตรงกันก็นำค่านั้นมาคำนวณต่อ ซึ่งจะเรียกเรียกค่านี้ว่า Response2 จากนั้นทำการเข้ารหัสและส่งไปยังเครื่องอ่าน

เมื่อผ่านการทำงาน 3 ขั้นตอนที่ระบุข้างต้น เครื่องอ่านจะทำการคำนวณด้วยวิธีเดียวกับบัตรจากนั้นนำผลลัพธ์มาตรวจสอบกับค่า Response2 ที่ถูกเข้ารหัสที่ได้รับจากบัตร หากค่าคีย์ตรงกับเครื่องอ่านจะถือว่าขั้นตอนการยืนยันตนสมบูรณ์ เครื่องอ่านสามารถอ่านหรือเขียนข้อมูลลงบัตรได้

2.2. โครงสร้างการเข้ารหัสด้วยอัลกอลิทึมคริปโตวัน

โครงสร้างของอัลกอลิทึมคริปโตวันจะมีพื้นฐานเป็น 48 บิต LFSR (Linear Feedback Shift Register) ซึ่งในการคำนวณแต่ละครั้งจะได้ค่าเอาท์พุทเรียกว่า KS (Key Stream) 1 บิต จากการเลือกบิตจำนวน 20 บิตจากบิตที่ 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, และ 47 ค่าคีย์ 48 บิตถูกนำมาผ่านฟิลเตอร์ฟังก์ชั่นและทำการปรับปรุงสถานะ LFSR โดยทำการชิพค่าคีย์ไปทางขวาและเติมบิตที่ 47 ด้วยค่าที่ได้จากการเอ็กคลูซีพออร์บิตจากค่าคีย์จำนวน 18 บิต คือ บิตที่ 0, 5, 9, 10, 12, 14, 15, 17, 19, 24, 25, 27, 29, 35, 39, 41, 42, 43 และค่า Input/Feedback

โดยสรุปอัลกอลิทึมสามารถแบ่งได้ 4 ส่วนดังรูปที่ 2 โดยแต่ละส่วนการคำนวณจะทำให้ค่าคีย์เริ่มต้นมีค่าเปลี่ยนแปลงไปและได้ KS รวม 32 บิต

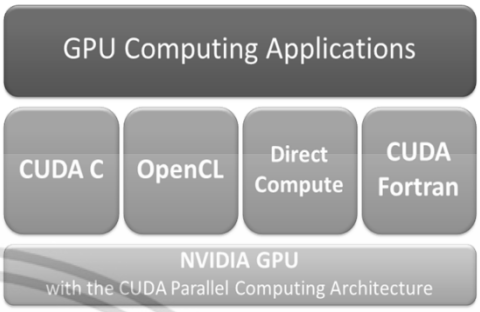


รูปที่ 2 โครงสร้างการเข้ารหัสของอัลกอลิทึมคริปโตวัน

3. คูด้ากราฟิกโพรเซสเซอร์

3.1 ความเป็นมาของคูด้า

ในเดือนพฤศจิกายน ปี ค.ศ. 2006 บริษัทเอ็นวีเดียได้เปิดตัวคูด้าซึ่งเป็นสถาปัตยกรรมการประมวลผลแบบขนานเพื่อวัตถุประสงค์การประมวลผลทั่วไป ด้วยรูปแบบการประมวลผลแบบขนานและมีชุดคำสั่งเฉพาะสำหรับสั่งงาน โดยจะช่วยให้สามารถเรียกใช้จีพียู เพื่อช่วยเพิ่มประสิทธิภาพการประมวลผลของซีพียูได้

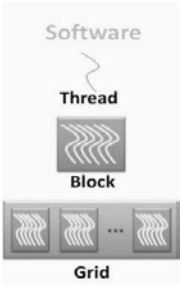


รูปที่ 3 สถาปัตยกรรมคูด้า

3.2 สถาปัตยกรรมคูด้า

คูด้าสามารถใช้ภาษาซีซึ่งเป็นภาษาระดับสูงในการพัฒนาซอฟแวร์ได้ นอกจากนี้ยังสามารถใช้ภาษาอื่นหรือเอพีไอ (API) ที่สนับสนุนการทำงานของคูด้า เช่น โอเพ่นซีแอล (OpenCL) คูด้าฟอร์แทรน (CUDA Fortran) และไดเร็กคอมพิวเตอร์ (Direct Compute) โดยสถาปัตยกรรมคูด้าแสดงดังรูปที่ 3

การเขียนโปรแกรมแบบขนานบนคูด้า สามารถทำได้โดยการแบ่งปัญหาออกเป็นปัญหาย่อยที่สามารถแก้ไขได้ โดยแต่ละปัญหาย่อยเป็นอิสระจากกันและสามารถคำนวณพร้อมกันแบบขนานได้โดยแบ่งการคำนวณออกเป็นเทรดและสามารถกำหนดให้เทรดทั้งหมดแบ่งออกเป็นบล็อกเพื่อให้เหมาะสมกับการคำนวณได้ และบล็อกทั้งหมดยังสามารถแบ่งออกเป็นกริด (Grid) ได้อีกด้วย โดยเทรดในบล็อกสามารถกำหนดค่าเป็น 1 มิติ 2 มิติ หรือ 3 มิติ บล็อกในกริดก็สามารถกำหนดให้เป็น 1 มิติ 2 มิติ หรือ 3 มิติได้เช่นกัน ทำให้ง่ายต่อการอ้างถึงในการคำนวณบางประเภท เช่น เมตตริก เวกเตอร์ หรือปริมาตรเป็นต้น ดังรูปที่ 4



รูปที่ 4 การแบ่งเทรดเป็นบล็อกและกริด

4. การทดลอง

บทความนี้ทำการโจมตีอัลกอลิทึมคริปโตวันด้วยวิธีการบูทฟอท ซึ่งเป็นวิธีการพื้นฐานในการโจมตีทุกอัลกอลิทึมและยังใช้เป็นฐานอ้างอิงต่อความทนทาน โดยหากอัลกอลิทึมถูกโจมตีด้วยวิธีการบูทฟอทและได้รับผลลัพธ์ที่ถูกต้องในระยะเวลาอันสั้นแสดงว่าเป็นอัลกอลิทึมที่อ่อนแอไม่เหมาะสมในการนำมาใช้งาน

ในอดีตอัลกอลิทึมคริปโตวันนับว่าเป็นอัลกอริทึมที่มีความทนทานต่อการโจมตีด้วยวิธีการบูทฟอท เนื่องจากต้องใช้เวลานานในการโจมตี แต่ด้วยเทคโนโลยีในปัจจุบันที่มีการนำจีพียูมาใช้ประมวลผลแบบขนานทำให้ลดเวลาการคำนวณหรือการโจมตีอย่างมีนัยสำคัญ บทความนี้จึงทดสอบการประยุกต์ใช้งานกราฟฟิกโพรเซสเซอร์เพื่อการโจมตีอัลกอริทึมดังกล่าวนี้ เวลาที่ใช้ในการประมวลผลระหว่างการประมวลผลด้วยซีพียูจะถูกเปรียบเทียบกับเวลาที่ใช้ในการประมวลผลด้วยจีพียู

4.1 ข้อมูลที่ใช้ทดสอบ

การทดสอบทำโดยการสุ่มค่าต่าง ๆ ที่เกี่ยวข้องกับขั้นตอนการยืนยันตนระหว่างเครื่องอ่านและบัตรขึ้น 3 ค่า คือเลขระบุตัวตนของบัตร Challenge1 ที่ถูกเข้ารหัส และ Challenge2 ที่ถูกเข้ารหัส จากนั้นจึงคำนวณหาค่า Response1 ที่ถูกเข้ารหัสและ Response2 ที่ถูกเข้ารหัส

4.2 สภาพแวดล้อมของการทดสอบ

การทดสอบกระทำโดยใช้ซีพียูเป็นโพรเซสเซอร์ Intel® Core ™ i5 – 2500 CPU (4 Processors) ความถี่สัญญาณนาฬิกา 3.3 GHz และจีพียู NVIDIA GeForce GTX 460 ระบบปฏิบัติการวินโดส์เซเวน (Windows 7) 32 บิท โดยเปรียบเทียบเวลาที่ใช้ในการโจมตีด้วยซีพียูกับเมื่อใช้จีพียู

4.3 การทดลองส่วนที่ 1 : การโจมตีด้วยวิธีบูทฟอท

เนื่องจากคีย์ของอัลกอริทึมคริปโตวันมีความยาว 48 บิต ดังนั้นจำนวนคีย์ทั้งหมดที่เป็นไปได้เท่ากับ 2^{48} หรือเท่ากับ 281,474,976,710,656 จำนวน ซึ่งเป็นข้อมูลจำนวนมากและไม่สามารถทำการทดสอบได้อย่างมีประสิทธิภาพ ดังนั้นจึงทำการทดสอบกับคีย์จำนวน 131,072 คีย์ ผลลัพธ์ที่ได้จะถูกเทียบเป็นสัดส่วนเพื่อหาเวลาที่ต้องใช้ในกรณีแย่ที่สุดของคีย์จำนวน 2^{48} คีย์

4.3.1 บูทฟอทด้วยหน่วยประมวลผลกลาง

การทดสอบกระทำโดยใช้โค้ดภาษาซีที่ถูกพัฒนาโดย I.C. Wiener โดยแทนค่าคีย์ในตำแหน่ง Key1 ในรูปที่ 2 เริ่มจากค่า 0 เพิ่มขึ้นทีละ 1 จากนั้นเปรียบเทียบค่า Response 1ที่ถูกเข้ารหัส และ Response2 ที่ถูกเข้ารหัส จนพบค่าคีย์ที่ให้ค่า Response1 ที่ถูกเข้ารหัส และ Response2 ที่ถูกเข้ารหัสที่ถูกต้อง และบันทึกเวลาทั้งหมดที่ใช้ในการประมวลผล

4.3.2 บูทฟอทด้วยกราฟิกโพรเซสเซอร์

สำหรับการบูทฟอทด้วยจีพียูดำเนินการโดยพัฒนาซอฟแวร์ประยุกต์ใช้กราฟฟิกโพรเซสเซอร์คำนวณในส่วนที่มีการคำนวณซ้ำกันหลายครั้งแต่มีการเปลี่ยนแปลงแค่อินพุท คือการแทนค่าคีย์ลงในคริปโตวันอัลกอลิทึมเพื่อหาค่าคีย์ที่ถูกต้อง ซอฟแวร์ที่พัฒนาขึ้นสามารถคำนวณพร้อมกันครั้งละ 256 บล็อค บล็อคละ 512 เทรด ซึ่งเปรียบเสมือนว่าทำการประมวลผลพร้อมกันโดยใช้จีพียูครั้งละ 131,072 เทรด แต่ละเทรดคำนวณด้วยค่าคีย์ที่แตกต่างกันเพื่อหาคำตอบพร้อมกันแบบขนานภายในจีพียู โดยซีพียูจะส่งค่าคีย์เริ่มต้นที่ต้องการหาขนาด 48 บิต ค่าต่างๆที่ใช้ในการคำนวณขนาด 64 บิต ส่งไปยังจีพียู เมื่อคำนวณเสร็จจีพียูจะส่งค่าแฟลกซ์ขนาด 1 บิตเพื่อแสดงสถานะการหาคำตอบและค่าคีย์ที่เป็นคำตอบขนาด 48 บิต กลับมายังซีพียู

4.4 การทดลองส่วนที่ 2 : การโจมตีด้วยวิธีบูทฟอทโดยอาศัยจุดอ่อนของอัลกอริทึม

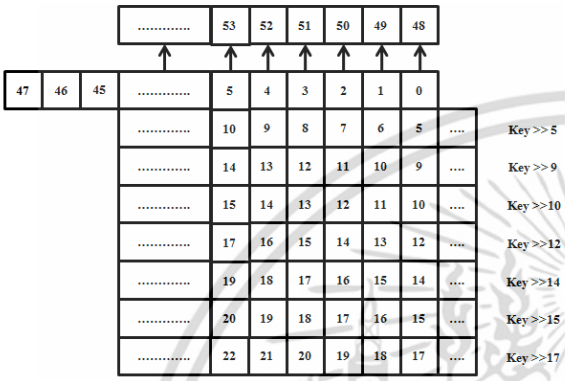
ในการทดลองส่วนนี้ทดสอบกับคีย์จำนวน 131,072 คีย์ ผลลัพธ์ที่ได้จะถูกเทียบเป็นสัดส่วนเพื่อหาเวลาที่ต้องใช้ในกรณีแย่ที่สุดของคีย์จำนวน 2^{48} คีย์ เช่นเดียวกับการทดลองส่วนที่ 1 โดยมีการปรับปรุงอัลกอลิทึมในการคำนวณเพื่อลดเวลาในการคำนวณลงโดยมีการปรับปรุงส่วนต่าง ๆ ดังนี้

4.4.1 เปลี่ยนจุดโจมตี

แทนค่าคีย์ในตำแหน่ง Key3 ในรูปที่ 2 โดยใช้วิธีการบูทฟอท เริ่มจากค่า 0 เพิ่มขึ้นทีละ 1 จากนั้นเปรียบเทียบค่า

Response1 ที่ถูกเข้ารหัสและ Response2 ที่ถูกเข้ารหัสจนกว่าจะเจอค่าคีย์ที่ให้ค่า Response1 ที่ถูกเข้ารหัส และ Response2 ที่ถูกเข้ารหัสที่ถูกต้อง จากนั้นจึงคำนวณย้อนกลับไปเพื่อหาค่าคีย์ Key1 ซึ่งจะช่วยลดเวลาในการคำนวณลงได้เนื่องจากต้องทำการคำนวณส่วนที่ 1 และส่วนที่ 2 แค่ครั้งเดียว

4.4.2 คำนวณผลลัพธ์จากการเปลี่ยนค่าคีย์ล่วงหน้า



รูปที่ 5 การคำนวณผลลัพธ์จากการเปลี่ยนค่าคีย์ล่วงหน้า

การคำนวณผลลัพธ์จากการเปลี่ยนค่าคีย์ล่วงหน้าทำได้โดยการนำค่าคีย์มาชิพขวาจำนวน 0, 5, 9, 10, 12, 14, 15, 17, 19, 24, 25 , 27, 29, 35, 39, 41, 42 และ 43 ครั้งแล้วนำค่าทั้ง 18 ค่ามาเก็บในตัวแปร จากนั้นนำค่าทั้งหมดมาเอ็กซ์คลูซิฟออร์กันบิตคำตอบที่ได้ตำแหน่งบิตที่ 0, 1, 2, 3 จะเป็นค่าผลลัพธ์ที่ได้จากคำนวณผลลัพธ์จากการเปลี่ยนค่าคีย์ครั้งที่ 1, 2, 3, 4 ของค่าคีย์ดังกล่าว ดังนั้นหากเรานำค่าคีย์ชิพขวาไป 4 บิตแล้วนำผลลัพธ์ที่คำนวณได้ไปแทนที่ในบิตที่ 47, 46, 45, 44 ของคีย์ แล้วทำการคำนวณแบบเดิมจำนวน 16 รอบจะทำให้เราสามารถหาบิตใหม่ของคีย์ล่วงหน้าได้ 64 บิท

4.4.3 คำนวณผลลัพธ์ KS ล่วงหน้า

เมื่อคำนวณผลลัพธ์จากการเปลี่ยนค่าคีย์ล่วงหน้าได้จำนวณ 64 บิต เราก็สามารถนำค่าที่คำนวณได้มาทำการคำนวณหาค่าผลลัพธ์ KS ล่วงหน้าได้เช่นกัน โดยมีหลักการคล้ายกับการคำนวณหาผลลัพธ์จากการเปลี่ยนค่าคีย์ล่วงหน้าคือ ทำการชิพคีย์ไปทางขวา 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45 และ 47 ครั้งและนำค่าที่ได้เก็บไว้ในตัวแปรโดยในตำแหน่งบิตที่ว่างไปไม่ครบ 48 บิตให้นำค่าจากการเปลี่ยนค่าคีย์ล่วงหน้าที่คำนวณได้มาเติมใส่ในตำแหน่งที่ถูกต้อง

เมื่อนำค่าที่ได้มาคำนวณจะได้ผลการคำนวณครั้งละ 64 บิตซึ่งค่าแต่ละบิตแทนการคำนวณแต่ละครั้งเปรียบเสมือนการคำนวณแต่ละครั้งสามารถคำนวณหาค่าผลลัพธ์ KS ได้ล่วงหน้าครั้งละ 64 บิต

4.4.4 ปรับปรุงการบูทฟอทด้วยหน่วยประมวลผลกลาง

ส่วนนี้ปรับปรุงวิธีการหาค่าคีย์ดังที่อธิบายในส่วนที่ 4.4.1 โดยลดเวลาที่ใช้คำนวณส่วนที่ 3 และ 4 จากรูปที่ 2 ด้วยการคำนวณตามวิธีที่ 4.4.2 และ 4.4.3

4.4.4 ปรับปรุงการบูทฟอทด้วยกราฟิกโพรเซสเซอร์

ซอฟทแวร์ส่วนนี้คำนวณพร้อมกันครั้งละ 256 บล็อค บล็อคละ 512 เทรด โดยใช้จีพียูในการหาค่า Key3 ที่ถูกต้องโดยลดเวลาที่ใช้ส่วนที่ 3 และ 4 ในรูปที่ 2 ด้วยการคำนวณตามวิธีที่ 4.4.2 และ 4.4.3 จากนั้นส่งค่า Key3 มายังซีพียูเพื่อคำนวณย้อนกลับไปหาค่าคีย์ Key1 โดยค่าที่ส่งไปกลับระหว่างซีพียูและจีพียูเหมือนกับส่วนที่ 1

5. ผลการทดลอง

ผลการทดลองโดยใช้ซีพียูและจีพียูแสดงในตารางที่1

ตารางที่ 1 ผลการทดลอง

	ซีพียู (มิลลิวินาที)	จีพียู(มิลลิวินาที)
บูทฟอท	2728.862793	103.135017
บูทฟอทโดยอาศัยจุดอ่อนของอัลกอริทึม	212.327789	53.340919

และในส่วนของการประยุกต์ใช้จีพียูมีการบันทึกผลการทดลองแยกเป็นส่วนแสดงดังตารางที่ 2

ตารางที่ 2 ผลการทดลองการประยุกต์ใช้จีพียู

	บูทฟอท (มิลลิวินาที)	บูทฟอทโดยอาศัยจุดอ่อนขออัลกอริทึม (มิลลิวินาที)
กำหนดตัวแปลในซีพียู	0.000930	0.001515

กำหนดค่าเริ่มต้นบริบทในจีพียู	49.861168	50.414013
จองหน่วยความจำในจีพียู	0.206931	0.189344
คัดลอกค่าจากซีพียูไปยังจีพียู	0.195241	0.094625
คำนวนหาคำตอบ	52.825439	2.579362
คัดลอกค่าจากจีพียูไปยังซีพียู	0.043966	0.061074

ผลการทดลองจากตารางที่ 1 แสดงให้เห็นถึงเวลาที่ใช้ในการประมวลผลข้อมูลจำนวน 131,072 ค่า เมื่อเทียบสัดส่วนในกรณีที่เป็นคีย์ทั้งหมด 2^{48} คีย์ พบว่าในกรณีที่แย่ที่สุดเมื่อโจมตีด้วยวิธีการบูทฟอท ซีพียูใช้เวลาประมาณ 185.82 ปี และจีพียูใช้เวลาประมาณ 3.61 ปี (เนื่องจากเวลาในส่วนการกำหนดตัวแปลในซีพียู การกำหนดค่าเริ่มต้นบริบทในจีพียู และการจองหน่วยความจำในจีพียูจะใช้เวลาในครั้งแรกของการรันโปรแกรมเท่านั้น) เพื่อการค้นหาคีย์ที่ถูกต้อง สำหรับการบูทฟอทโดยอาศัยจุดอ่อนของอัลกอริทึม ซีพียูใช้เวลาประมาณ 14.46 ปี และ จีพียูใช้เวลาประมาณ 0.19 ปี หรือ 67.98 วัน

เมื่อนำจีพียูมาประยุกต์ใช้ในการประมวลผลพบว่าประสิทธิภาพการโจมตีด้วยวิธีการบูทฟอทเพิ่มขึ้น 51.47 เท่า และการโจมตีโดยอาศัยจุดอ่อนของอัลกอริทึมเพิ่มขึ้น 76.01 เท่า

6. สรุป

บทความนี้นำเสนอผลการเปรียบเทียบการโจมตีเพื่อค้นหาคีย์ที่ถูกต้องด้วยการใช้ซีพียูและด้วยการใช้จีพียูเพื่อทำการโจมตีแบบบูทฟอทและการโจมตีโดยอาศัยจุดอ่อนของอัลกอริทึม ประสิทธิภาพที่เพิ่มขึ้นนี้มาจากความสามารถในการประมวลผลแบบขนานจากจำนวนโพรเซสเซอร์จำนวนมากที่มีอยู่ในจีพียู ดังนั้นหากการประมวลผลมีการกระบวนการประมวลผลที่เหมือนกันโดยข้อมูลที่ใช้ต่างกันและเป็นอิสระต่อกัน การใช้จีพียูจึงเป็นตัวเลือกที่สามารถเพิ่มประสิทธิภาพการทำงานของระบบได้

เป็นที่น่าสังเกตุว่าจีพียูที่ใช้ในการทดสอบครั้งนี้ เป็นจีพียูระดับกลางที่ผู้ใช้สามารถเลือกใช้งานได้ด้วยงบที่จำกัด หากมีการใช้จีพียูประสิทธิภาพสูง เช่น จีพียูของเอ็นวีเดียรุ่นเทสลา (Tesla) ประสิทธิภาพการประมวลผลมีโอกาสจะเพิ่มสูงขึ้นกว่านี้หลายเท่า

จากมุมมองของระดับความสามารถของอัลกอริทึมในการเข้ารหัสเพื่อรักษาความปลอดภัยของข้อมูลหรือเพื่อการยืนยันตัวตนนั้น ระบบการเข้ารหัสจำเป็นต้องคำนึงถึงเทคโนโลยีปัจจุบันที่สามารถลดเวลาในการประมวลผลได้อย่างมีนัยสำคัญ จากตัวอย่างที่นำเสนอเห็นได้ว่าอัลกอริทึมคริปโตวันที่ใช้ในขั้นตอนการยืนยันตนของอาร์เอฟไอดีไมแฟร์คลาสสิคที่ได้รับความนิยมเป็นอย่างมากและมีการใช้งานอย่างแพร่หลายนั้นอาจจะมีระดับความทนทานต่อการโจมตีไม่เพียงพอโดยเฉพาะอย่างยิ่งต่อเทคโนโลยีที่ได้รับการพัฒนาไปอย่างรวดเร็วเช่นในปัจจุบันนี้

7. เอกสารอ้างอิง

[1] K. Nohl and H. Plötz, “Mifare: Little Security despite Obscurity,” Presented in the 24th Congress of the Chaos Computer Club, Berlin, 2007.

[2] J. Sanders and E. Kandrot, CUDA by Example, Addison-Wesley, Upper Saddle River, NJ, 2010.

[3] K. E. Penri-Williams, “Implementing an RFID Mifare Classic Attack,” M.S. thesis, Department of Telecommunication & Networks, City University London, London, UK, 2009.

[4] F.D. Garcia, G.K. Gans, R. Muijrers, P. Rossum, R. Verdult, R.W. Schreur, and B. Jacobs, “Dismantling MIFARE Classic,” ESORICS 2008, LNCS, vol. 5283, pp. 97–114, 2008.

[5] Gerhard de Koning Gans, J.H. Hoepman, and F.D. Garcia, “A Practical Attack on the MIFARE Classic,” CARDIS 2008, LNCS, vol. 5189, pp. 267-282, 2008.

[6] F.D. Garcia, P. van Rossum, R. Verdult, and R.W. Schreur, “Wirelessly Pickpocketing a Mifare Classic Card,” the 30th IEEE Symposium on Security and Privacy (S&P 2009), pp.3-15, 2009.