การเปรียบเทียบประสิทธิภาพของลายเซ็นดิจิทัล และกระบวนการ MAC
Comparison of the performance of Digital Signature and MAC Algorithms

ปุญญิศา แก่นสวรรค์ (Punyisa Khansawan)* และ ศิรปัฐช์ บุญครอง (Sirapat Boonkrong)*

Received: September 13, 2022
Revised: November 7, 2022
Accepted: November 28, 2022

* ผู้นิพนธ์ประสานงาน: ปุญญิศา แก่นสวรรค์ (Punyisa Khansawan) อีเมล: m6303280@g.sut.ac.th

บทคัดย่อ

การติดต่อสื่อสารในยุคปัจจุบันนั้น สามารถเกิดขึ้นได้อย่างสะดวกและรวดเร็วด้วยการติดต่อสื่อสารผ่านเครือข่าย และสามารถจัดเก็บข้อมูลไว้ได้นานบนเครือข่ายหรืออุปกรณ์อิเล็กทรอนิกส์ แต่การสื่อสารหรือจัดเก็บข้อมูลบนเครือข่ายนั้นอาจเกิดความไม่ปลอดภัยของข้อมูล เช่น การสูญเสียความถูกต้องสมบูรณ์ของข้อมูล จึงควรนำกลไกการรักษาความปลอดภัยของข้อมูลมาใช้เพื่อตรวจสอบและรักษาความปลอดภัยของข้อมูล ในบทความวิจัยนี้ซึ่งมีวัตถุประสงค์เพื่อทดสอบว่าอัลกอริทึมที่มีความสามารถการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลและระบุตัวตนผู้สร้างข้อความใดที่มีประสิทธิภาพด้านเวลาในการทำงานมากที่สุด จึงได้ทำการศึกษาอัลกอริทึม ECDSA, HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128 และ CBC-MAC-AES256 ซึ่งเป็นอัลกอริทึมที่ใช้ในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลและระบุตัวตนผู้สร้างข้อความ และได้ทำการเปรียบเทียบประสิทธิด้านเวลาในการดำเนินการของทั้ง 5 อัลกอริทึม โดยใช้ข้อมูลที่มีขนาดต่างกัน 10 ขนาด ขนาดละ 20 ชุดข้อมูล และพิจารณาค่า Throughput

จากการศึกษา พบว่า อัลกอริทึม HMAC-SHA256 เป็นอัลกอริทึมที่มีค่า Throughput สูงที่สุด จึงมีประสิทธิภาพด้านเวลาในการทำงานมากที่สุด รองลงมาคือ HMAC-SHA512, ECDSA, CBC-MAC-AES128 และ CBC-MAC-AES256 ตามลำดับ นอกจากนี้ยังพบว่า อัลกอริทึม HMAC จะมีประสิทธิภาพในการทำงานที่มากขึ้นเมื่อข้อมูลมีขนาดใหญ่ขึ้นอีกด้วย

คำสำคัญ: ลายเซ็นดิจิทัล กระบวนการแฮช ความถูกต้องสมบูรณ์ของข้อมูล การระบุตัวตน

Abstract

Today's communications can happen at high speed and with convenience over computer networks. Moreover, the networks and their devices can store data for a long time. However, communicating or storing data over the network can lead to data insecurity, such as loss of data integrity. Therefore, information security mechanisms should be used to verify and secure information. This research article has an objective to test which algorithms capable of verifying data integrity and identifying the authors of the messages were the most time efficient. The algorithms that were studied included ECDSA, HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128, and CBC-MAC-AES256, all of which could be used to verify the integrity of the information and to identify the author of the message. We compared the execution time efficiency of the 5 algorithms using 10 different sizes of data, 20 data sets each and the throughput values were considered.

From the study, the HMAC-SHA256 algorithm was the least time-consuming so it was the most efficient. In addition, the algorithm was more efficient when the data became larger than well. This was then followed by HMAC-SHA512, ECDSA, CBC-MAC-AES128, and CBC-MAC-AES256 respectively.

* สำนักวิชาศาสตร์และศิลป์ดิจิทัล มหาวิทยาลัยเทคโนโลยีสุรนารี
* Institute of Digital Arts and Science, Suranaree University of Technology.

Keywords: Digital Signature, Hash Algorithm, Data Integrity, Identification.

1. บทนำ

ในยุคปัจจุบันมนุษย์ใช้การติดต่อสื่อสารผ่านเครือข่ายมากขึ้น ไม่ว่าจะเป็นการสื่อสารด้วยตัวอักษร ภาพนิ่ง เสียง หรือภาพเคลื่อนไหว ซึ่งการสื่อสารผ่านเครือข่ายนี้ช่วยให้มนุษย์เกิดความสะดวกและรวดเร็วในการแลกเปลี่ยนข้อมูล อีกทั้งยังสามารถจัดเก็บข้อมูลไว้ได้นานและไม่สูญหายไว้บนเครือข่ายหรืออุปกรณ์อิเล็กทรอนิกส์ แต่สื่อสารและจัดเก็บข้อมูลไว้บนเครือข่ายหรืออุปกรณ์อิเล็กทรอนิกส์นั้น สามารถเกิดความไม่ปลอดภัยของข้อมูลได้ เพราะในปัจจุบันเกิดการโจมตีข้อมูลประเภทต่าง ๆ อย่างมากมาย เช่น การโจมตีประเภท Modification ซึ่งการโจมตีประเภทนี้ เป็นการโจมตีที่เกี่ยวข้องกับการเปลี่ยนแปลงข้อมูล โดยข้อมูลนั้นอาจถูกแก้ไขหรือเปลี่ยนแปลงขณะส่ง-รับข้อมูลบนเครือข่าย โดยผู้โจมตี ทำให้เกิดปัญหาในการสื่อสาร เนื่องจากข้อมูลที่ผู้รับได้รับไม่ถูกต้องสมบูรณ์ตามที่ผู้ส่งส่งมา หรือข้อมูลถูกแก้ไขเปลี่ยนแปลงขณะจัดเก็บข้อมูลโดยผู้โจมตี ทำให้สูญเสียความถูกต้องสมบูรณ์ของข้อมูลได้ [1] ดังนั้น ในการสื่อสารและจัดเก็บข้อมูลผ่านเครือข่ายและอุปกรณ์อิเล็กทรอนิกส์นั้น ควรมีการนำกลไกการรักษาความปลอดภัยของข้อมูลมาใช้เพื่อตรวจสอบและรักษาความถูกต้องสมบูรณ์ของข้อมูลที่สื่อสารและจัดเก็บ เช่น การนำอัลกอริทึมลายเซ็นดิจิทัล ซึ่งเป็นอัลกอริทึมการตรวจสอบและระบุตัวตนผู้ส่งข้อความที่นิยมในปัจจุบัน โดยใช้ Private Key ของผู้ส่งร่วมกับกระบวนการแฮช (Hash Algorithm) ในการส่งข้อความไปยังผู้รับ [2], [3] หรือการนำอัลกอริทึม MAC ซึ่งเป็นกระบวนการแฮชที่มีการนำคีย์เข้ามาเป็นส่วนหนึ่งของการคำนวณค่าแฮช โดยคีย์ที่นำมาใช้คีย์เป็น Symmetric Key ที่ผู้ส่งและผู้รับมีคีย์ตัวเดียวกัน ซึ่ง MAC [4], [5] อัลกอริทึมนี้นอกจากจะใช้ในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลตามคุณสมบัติของกระบวนการแฮชแล้วนั้น ยังมีการใช้คีย์ในการเข้ารหัสร่วมในกระบวนการทำงานทำให้สามารถใช้เพื่อตรวจสอบตัวตนของผู้ส่งข้อความได้อีกด้วย

จะเห็นได้ว่าทั้งสองอัลกอริทึมดังกล่าวมีความสามารถในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูล เนื่องจากมีการใช้กระบวนการแฮช และยังสามารถตรวจสอบและระบุตัวตนผู้สร้างข้อความได้ เนื่องจากมีการใช้คีย์การเข้ารหัสเป็นส่วนหนึ่งในกระบวนการทำงานแต่เรายังไม่ทราบถึงประสิทธิภาพการทำงานของทั้งสองอัลกอริทึมดังกล่าวว่าหากเราต้องการอัลกอริทึมที่มีความสามารถในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลและตรวจสอบผู้สร้างข้อความโดยคำนึงถึงประสิทธิภาพด้านเวลาในการทำงาน อัลกอริทึมใดมีประสิทธิภาพมากกว่ากัน

ในการศึกษาครั้งนี้ ผู้วิจัยได้ทำการเปรียบเทียบประสิทธิภาพด้านเวลาในการทำงานของอัลกอริทึมลายเซ็นดิจิทัล ได้แก่ ECDSA ร่วมกับ SHA256 และ อัลกอริทึม MAC ได้แก่ HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128 และ CBC-MAC-AES256 ด้วยข้อมูลที่มีขนาดต่างกัน 10 ขนาด ขนาดละ 20 ชุดข้อมูล เพื่อให้ทราบประสิทธิภาพการทำงานที่เหมาะสมแก่การนำมาประยุกต์ใช้ในงานที่ต้องการการรักษาความถูกต้องสมบูรณ์ของข้อมูลและความสามารถในการตรวจสอบผู้สร้างข้อความในอนาคต ซึ่งในงานการศึกษาครั้งนี้ผู้วิจัยได้กำหนดขอบเขตของงานวิจัยโดยจะทำการเปรียบเทียบประสิทธิภาพด้านความเร็วในการประมวลผลในส่วนของการสร้างลายเซ็นดิจิทัลและค่าแฮชเท่านั้น

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

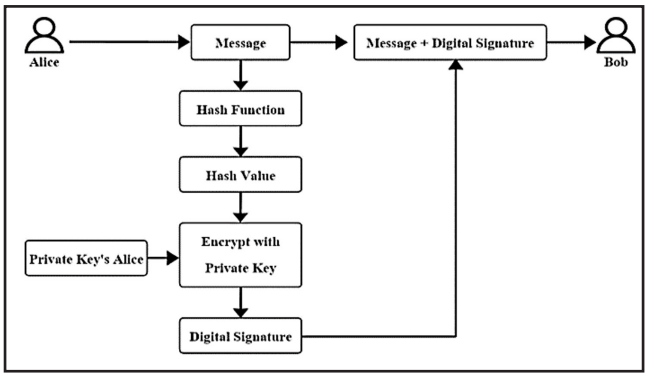
2.1 ลายเซ็นดิจิทัล (Digital Signature: DSA)

Digital Signature (DSA) เป็นอัลกอริทึมที่ใช้หลักการเข้ารหัสแบบอสมมาตร (Asymmetric Cryptography) ร่วมกับกระบวนการแฮช เพื่อนำมาใช้สำหรับการลงนามในเอกสารเพื่อรับรองความถูกต้องของแหล่งที่มาหรือผู้ส่ง โดยใช้คีย์ส่วนตัว (Private Key) ในการลงนามและส่งไปพร้อมกับข้อความต้นฉบับ ซึ่งผู้รับสามารถตรวจสอบได้โดยใช้คีย์สาธารณะของผู้ส่ง (Public Key) เพื่อพิสูจน์ว่าข้อความที่ได้รับเป็นของผู้ส่งจริง [2], [3], [6], [7]

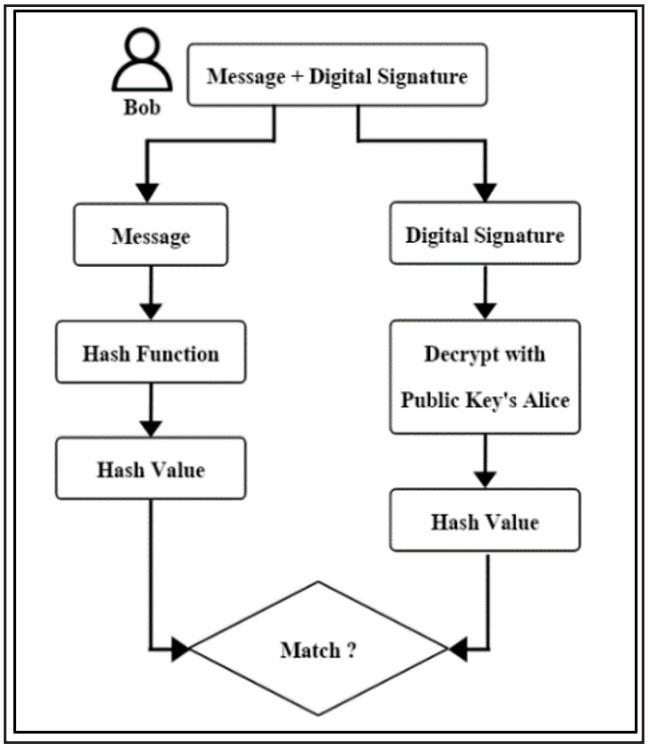
จากภาพที่ 1 ซึ่งแสดงกระบวนการทำงานของลายเซ็นดิจิทัล ผู้ส่งจะนำข้อความต้นฉบับไปเข้ากระบวนการแฮช หลังจากนั้นจะนำค่าแฮชที่ได้ไปเข้ากระบวนการเข้ารหัสแบบอสมมาตรด้วยคีย์ส่วนตัวของตนเองทำให้ได้ลายเซ็นดิจิทัล และสามารถส่งไปให้กับผู้รับพร้อมข้อความต้นฉบับได้

ในด้านของผู้รับนั้นสามารถตรวจสอบลายเซ็นดิจิทัลที่ได้รับได้โดยการนำข้อความต้นฉบับที่ได้รับไปเข้ากระบวนการแฮช

ซึ่งจะได้ค่าแฮช (1) และนำลายเซ็นดิจิทัลที่ได้รับไปถอดรหัสด้วยคีย์สาธารณะของผู้ส่ง จะได้รับค่าแฮช (2) จากนั้นนำค่าแฮช (1) และ (2) มาเปรียบเทียบกัน หากค่าแฮชทั้งสองตรงกัน หมายความว่าข้อความนี้ได้รับ มาจากผู้ส่งจริงและข้อมูลมีความถูกต้องสมบูรณ์ [7] ดังแสดงในภาพที่ 2



ภาพที่ 1 *กระบวนการลายเซ็นดิจิทัล*



ภาพที่ 2 *กระบวนการตรวจสอบลายเซ็นดิจิทัล*

ในบทความนี้ผู้วิจัยได้ใช้อัลกอริทึม Elliptic Curve Digital Signature Algorithm (ECDSA) ร่วมกับ SHA256 ในการทดสอบและประเมินประสิทธิภาพ โดยอัลกอริทึม ECDSA เป็นอัลกอริทึมลายเซ็นดิจิทัลที่ใช้อัลกอริทึม Elliptic Curve Cryptography (ECC) ในการสร้างลายเซ็นดิจิทัล เพื่อระบุตัวตน ซึ่งได้รับการพิสูจน์ว่ามีความปลอดภัย เป็นมาตรฐานที่ได้รับการรับรองโดย ANSI, IEEE และ NIST และเป็นที่นิยมใช้ในปัจจุบัน [6] - [9] นอกจากนี้ ECDSA ยังได้เปรียบอัลกอริทึมลายเซ็นดิจิทัลอื่น ๆ ในแง่ของเวลาในการดำเนินการและพื้นที่จัดเก็บ เนื่องจากมีการใช้คีย์ขนาดที่เล็กกว่าอัลกอริทึมอื่น ๆ แต่มีความปลอดภัยเท่ากัน [7], [10]

2.2 Message Authentication Code (MAC)

MAC อัลกอริทึม เป็นอัลกอริทึมที่ถูกใช้ระหว่างสองฝ่ายที่แชร์ความลับ เพื่อใช้ในการตรวจสอบความถูกต้องของข้อมูล เพื่อให้มั่นใจว่าข้อมูลไม่มีการเปลี่ยนแปลงและสามารถตรวจสอบผู้สร้างข้อความได้ เนื่องจาก MAC อัลกอริทึมมีคีย์สมมาตร ซึ่งเป็นคีย์ตัวเดียวกันที่รู้เฉพาะผู้ส่งและผู้รับเข้ามาเกี่ยวข้องกับกระบวนการทำงาน จึงสามารถสร้างความมั่นใจได้ว่าเป็นข้อมูลที่ถูกต้องสมบูรณ์โดยผู้ส่งที่ถูกต้อง ซึ่ง MAC อัลกอริทึมสามารถแบ่งได้เป็น 2 ประเภท คือ HMAC และ CBC-MAC [4], [11], [12], [13] .

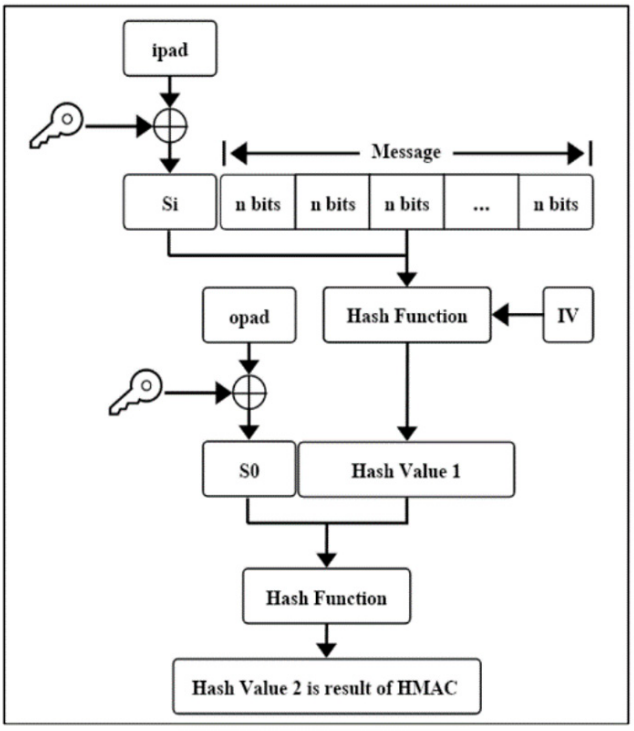
2.2.1 Hash-Based Message Authentication Code (HMAC).

HMAC เป็นอัลกอริทึมการทำงานโดยมีพื้นฐานบนกระบวนการแฮชและเพิ่มคีย์เข้ามาเป็นส่วนหนึ่งของกระบวนการทำงาน จึงทำให้ความสามารถในการตรวจสอบความถูกต้องของข้อความนั้นขึ้นอยู่กับคุณสมบัติของฟังก์ชันแฮชที่นำมาใช้งาน [12] และคีย์ที่ใช้ในการสร้างค่าแฮชและตรวจสอบค่าแฮชนั้นจะต้องใช้คีย์ตัวเดียวกันที่แชร์ร่วมกันระหว่างผู้สร้างและผู้รับเท่านั้น [4] โดยมีกระบวนการทำงานดังแสดงในภาพที่ 3

จากภาพที่ 3 แสดงให้เห็นถึงกระบวนการทำงานของ HMAC ซึ่งมีรูปแบบการทำงานโดยแบ่งข้อความต้นฉบับออกเป็นบล็อกขนาด n บิตตามกระบวนการแฮชที่เลือกใช้ เช่น SHA256, SHA512 และคีย์ที่ใช้จะต้องมีขนาด n บิต เท่ากับขนาดของบล็อก หากคีย์มีขนาดเล็กกว่าจะต้องทำการเติมบิต 0 ต่อท้ายเพื่อให้มีจำนวนเท่ากับ n บิต แต่หากคีย์มีขนาดใหญ่กว่าจะต้องนำคีย์ไปผ่านกระบวนการแฮชเพื่อให้คีย์มีขนาดเท่ากับ n บิต

จากนั้นนำคีย์ที่พร้อมใช้งานมาทำการ XOR กับ ipad จะได้ค่า Si ซึ่งจะนำค่า Si ไปต่อกับข้อความและนำไปใส่กระบวนการแฮช จะได้ค่าแฮชออกมา จากนั้นนำคีย์ที่

พร้อมใช้งานมาทำการ XOR กับ opad จะได้ค่า S0 และนำค่า S0 กับค่าแฮชที่ได้ไปเข้ากระบวนการแฮชเดียวกันกับขั้นตอนข้างต้นจะได้ผลลัพธ์เป็นค่าแฮชส่งไปให้ผู้รับ [11] , [12] ซึ่งผู้รับจะทำการตรวจสอบโดยการนำข้อความและคีย์ที่แชร์ร่วมกันกับผู้ส่งไปผ่านกระบวนการเช่นเดียวกันกับผู้ส่ง จากนั้นนำค่าแฮชที่ได้รับมาเปรียบเทียบกับค่าแฮชที่คำนวณได้ หากค่าแฮชทั้งสองตรงกัน หมายความว่า ข้อความที่ได้รับเป็นข้อความที่ถูกต้องสมบูรณ์ และมาจากผู้ส่งจริง [4], [5]

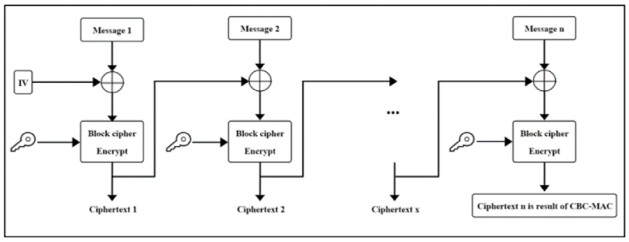


ภาพที่ 3 *กระบวนการทำงาน HMAC*

2.2.2 Cipher Block Chaining Message Authen-tication Code (CBC-MAC).

เป็นอัลกอริทึมการเข้ารหัสแบบบล็อกไซเฟอร์ (Block Cipher) โหมด CBC ซึ่งเป็นอัลกอริทึมที่เป็นที่รู้จักมากที่สุดในการอัลกอริทึมการตรวจสอบข้อความที่มีการเข้ารหัสแบบบล็อก [14], [15] โดยผู้รับและผู้ส่งมีคีย์ในการเข้ารหัสและถอดรหัสตัวเดียวกัน ซึ่งขนาดของคีย์และบล็อกที่ใช้ในการคำนวณค่า MAC ขึ้นอยู่กับอัลกอริทึมที่เลือกใช้ [3] และมีกระบวนการทำงานดังแสดงในภาพที่ 4

จากภาพที่ 4 เมื่อต้องการส่งข้อความผู้ส่งจะทำการเข้ารหัสข้อมูลด้วยคีย์การสนทนาและอัลกอริทึมที่เลือกใช้งาน เช่น AES128, AES256 เมื่อทำการเข้ารหัสเสร็จเรียบร้อย



ภาพที่ 4 *กระบวนการทำงานของ CBC-MAC*

ผู้ส่งสามารถนำค่าไซเฟอร์ในบล็อกสุดท้ายมาเป็นค่า MAC ดังนั้น ข้อความที่ผู้รับจะได้รับจะประกอบด้วยข้อความที่เข้ารหัส (Cipher Text) ค่า IV และค่า MAC ที่ได้จากค่าไซเฟอร์ในบล็อกสุดท้าย ซึ่งผู้รับจะทำการตรวจสอบความถูกต้องของข้อความที่ได้รับโดยการนำ Cipher Text ที่ได้รับมาทำการถอดรหัสให้ได้ข้อความต้นฉบับ จากนั้น นำข้อความที่ได้มาทำการเข้ารหัสด้วยคีย์การสนทนาและอัลกอริทึมที่เลือกใช้งานอีกครั้ง และนำค่า MAC ที่ได้มาตรวจสอบว่าตรงกับค่า MAC ที่ได้รับหรือไม่ ถ้าตรงกัน หมายความว่าข้อความที่ได้รับมาจากผู้ส่งจริงและข้อมูลมีความถูกต้องสมบูรณ์ [11], [14], [15].

2.3 งานวิจัยที่เกี่ยวข้อง

จากการศึกษางานวิจัยที่เกี่ยวข้องเกี่ยวกับการเปรียบเทียบประสิทธิภาพระหว่างอัลกอริทึมลายเซ็นดิจิทัลและ MAC อัลกอริทึม โดย Yoon และ Yoo [16] ได้ทำการศึกษาการแลกเปลี่ยนคีย์ระหว่างการใช้อัลกอริทึม Diffie-Hellman ร่วมกับอัลกอริทึมลายเซ็นดิจิทัล และการแลกเปลี่ยนคีย์โดยใช้อัลกอริทึม Diffie-Hellman ร่วมกับ MAC อัลกอริทึม ผลการวิจัยพบว่า ทั้งสองอัลกอริทึมมีความปลอดภัยในการแลกเปลี่ยนคีย์ แต่อัลกอริทึม Diffie-Hellman ร่วมกับ MAC อัลกอริทึมจะมีประสิทธิภาพมากกว่าในเรื่องของเวลาที่ใช้ในกระบวนการทำงานที่สามารถคำนวณคีย์ได้ถึง 20,000 ครั้งต่อวินาที ซึ่งมีความสามารถที่รวดเร็วกว่า อัลกอริทึม Diffie-Hellman ร่วมกับอัลกอริทึมลายเซ็นดิจิทัล ที่สามารถคำนวณคีย์ได้เพียง 2 ครั้งต่อวินาที เนื่องจากอัลกอริทึมลายเซ็นดิจิทัลมีกระบวนการทำงานที่ซับซ้อนกว่า MAC อัลกอริทึม นอกจากนี้ในงานวิจัยของ Werayut และ Nattakarn [11] ซึ่งได้ทำการเปรียบเทียบประสิทธิภาพการทำงานของ MAC อัลกอริทึม ได้แก่ HMAC-MD5, HMAC-SHA1 และ CBC-MAC-AES256 พบว่า HMAC ใช้เวลาในการทำงานน้อยกว่า CBC-MAC-AES

โดย HMAC-MD5 ใช้เวลาน้อยที่สุด ตามด้วย HMAC-SHA1 และ CBC-MAC-AES256 ตามลำดับ นอกจากนี้ยังพบว่า HMAC จะมีประสิทธิภาพการทำงานที่ดีขึ้นอย่างมาก เมื่อข้อมูลมีขนาดใหญ่ขึ้น

ในปัจจุบันอัลกอริทึม MD5 และ SHA1 ไม่เป็นที่นิยมใช้ เนื่องจากมีช่องโหว่ในเรื่องของการชนกันของค่าแฮช (Hash Value) และ Rainbow Table ดังนั้น ในบทความวิจัยนี้จึงได้ใช้ MAC อัลกอริทึม ได้แก่ HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128 และ CBC-MAC-AES256 ซึ่งเป็นอัลกอริทึมที่มีความปลอดภัยและยังมีการนำมาประยุกต์ใช้ในปัจจุบัน [8] เปรียบเทียบกับอัลกอริทึมลายเซ็นดิจิทัล ได้แก่ ECDSA เพื่อทดสอบประสิทธิภาพด้านเวลาในการทำงาน

3. วิธีดำเนินการวิจัย

ในการศึกษาและเปรียบเทียบประสิทธิภาพของ MAC อัลกอริทึม ได้แก่ HMAC-SHA256, HMAC-SHA512, CMC-MAC-AES128, CBC-MAC-AES256 และอัลกอริทึมลายเซ็นดิจิทัล ได้แก่ ECDSA ร่วมกับ SHA256 ซึ่งใช้วิธีการสร้างคีย์ด้วยอัลกอริทึม Elliptic Curve Cryptography (ECC) โดยพิจารณาเวลาที่ใช้ในการทำงานของแต่ละอัลกอริทึม ผู้วิจัยได้ทำการกำหนดขนาดของข้อมูลสำหรับการทดลองทั้งหมด 10 ขนาดข้อมูล ได้แก่ ข้อมูลขนาด 10, 20, 30, 40, 50, 60, 70, 80, 90 และ 100 ไบต์ ซึ่งเป็นขนาดของข้อความที่ใช้ในการสื่อสารกันโดยทั่วไป เช่น การสื่อสารผ่าน SMS ของโทรศัพท์มือถือ การสื่อสารผ่านแอปพลิชันส่งข้อความ เป็นต้น โดยการสุ่มตัวอักษรในแต่ละขนาดจำนวน 20 ชุดข้อมูล เป็นไฟล์ .txt

ผู้วิจัยได้ใช้เครื่องคอมพิวเตอร์แบบพกพา Intel® Core™ i7-8568U 1.8GHz หน่วยความจำหลัก 8GB DDR4 Memory หน่อยความจำสำรองขนาด 1TB HDD ระบบปฏิบัติการ Microsoft Windows 11 และใช้โปรแกรม Cygwin และฟังก์ชันของโปรแกรม OpenSSL ในการทดสอบ

3.1 การสร้างไฟล์

ในการสร้างไฟล์สุ่มตัวอักษร ผู้วิจัยได้ใช้คำสั่งดังแสดงในภาพที่ 5 โดย head -c 10 คือการกำหนดขนาดของตัวอักษรที่สุ่ม มีหน่วยเป็น Bytes

```
$ for i in {1..20}; do tr -dc
A-Za-z0-9 </dev/urandom | head
-c 10 > random10_$i.txt; done
```

ภาพที่ 5 *คำสั่งสร้างไฟล์สุ่มตัวอักษร*

3.2 การทดสอบประสิทธิภาพของ Digital Signature (DSA)

3.2.1 การสร้างคีย์

ในการทดสอบประสิทธิภาพการทำงานของลายเซ็นดิจิทัลผู้วิจัยได้ใช้อัลกอริทึม ECDSA ในการทดสอบประสิทธิภาพ โดยทำการสร้างคีย์ส่วนตัวและคีย์สาธารณะขนาด 256 บิตด้วยอัลกอริทึม ECC และจัดเก็บไว้ในไฟล์ .pem ซึ่งคำสั่งที่ใช้ในการสร้างคีย์ส่วนตัวดังแสดงในภาพที่ 6 และคำสั่งที่ใช้ในการสร้างคีย์สาธารณะดังแสดงในภาพที่ 7

```
$ openssl ecparam -name prime
256v1 -genkey -noout -out pri
vate-key.pem
```

ภาพที่ 6 *การสร้างคีย์ส่วนตัว*

```
$ openssl ec -in private-key.
pem -pubout -out public-key.pem
```

ภาพที่ 7 *การสร้างคีย์สาธารณะ*

```
$ for i in $(ls random10b_*.txt)
; do (time openssl dgst -sha256
-sign private-key.pem $i > $i.
bin) ; done
```

ภาพที่ 8 *การทดสอบลายเซ็นดิจิทัล*

3.2.2 การทดสอบ Digital Signature

ผู้วิจัยได้ใช้คำสั่ง -sign ร่วมกับ -sha256 และคีย์ส่วนตัวที่สร้างขึ้นในขั้นตอนที่ 3.2.1 ในการสร้างลายเซ็นดิจิทัลในแต่ละไฟล์ข้อมูล ($i) และจัดเก็บไว้ในไฟล์ชื่อ $i.bin โดยใช้คำสั่ง time ในการจับเวลาเพื่อวัดประสิทธิภาพการทำงาน ดังแสดงในภาพที่ 8

3.3 การทดสอบประสิทธิภาพของ HMAC

3.3.1 อัลกอริทึม HMAC-SHA256

ผู้วิจัยทำการทดสอบประสิทธิภาพของ HMAC-SHA256 โดยใช้คีย์ขนาด 128 บิตและใช้คำสั่ง time ในการจับเวลาเพื่อวัดประสิทธิภาพการทำงาน ดังแสดงในภาพที่ 9

```
$ for i in $(ls random10b_*.txt)
; do (time openssl sha256 -mac
hmac -macopt hexkey:0123456789
ABCDEF0123456789ABCDEF $i) ;
done
```

ภาพที่ 9 *การทดสอบ HMAC-SHA256*

3.3.2 อัลกอริทึม HMAC-SHA512

ผู้วิจัยทำการทดสอบประสิทธิภาพของ HMAC-SHA512 โดยใช้คีย์ขนาด 128 บิตและใช้คำสั่ง time ในการจับเวลาเพื่อวัดประสิทธิภาพการทำงาน ดังแสดงในภาพที่ 10

```
$ for i in $(ls random10b_*.txt)
; do (time openssl sha512 -mac
hmac -macopt hexkey:0123456789
ABCDEF0123456789ABCDEF $i) ;
done
```

ภาพที่ 10 *การทดสอบ HMAC-SHA512*

3.4 การทดสอบประสิทธิภาพของ CBC-MAC-AES

3.4.1 อัลกอริทึม CBC-MAC-AES128

ผู้วิจัยทำการทดสอบประสิทธิภาพของ CBC-MAC-AES128 โดยใช้คีย์ขนาด 128 บิต iv ขนาด 128 บิต ซึ่งคำสั่ง tail -c 32 คือการระบุบล็อกสุดท้ายของการเข้ารหัสด้วยอัลกอริทึม AES ในแต่ละขนาด input ซึ่งก็คือค่า MAC และใช้คำสั่ง time ในการจับเวลาเพื่อวัดประสิทธิภาพการทำงาน ดังแสดงในภาพที่ 11

```
$ for i in $(ls random10b_*.txt)
; do (echo -n $i ; time openssl
enc -e -aes-128-cbc -K 0123456
789ABCDEF0123456789ABCDEF -iv 0
00000000000000000000000000000
0 -in $i | tail -c 32 | od -An
-tx1) ; done
```

ภาพที่ 11 *การทดสอบ CBC-MAC-AES128*

3.4.2 อัลกอริทึม CBC-MAC-AES256

ผู้วิจัยทำการทดสอบประสิทธิภาพของ CBC-MAC-AES256 โดยใช้คีย์ขนาด 256 บิต iv ขนาด 128 บิต ซึ่งคำสั่ง tail -c 32 คือการระบุบล็อกสุดท้ายของการเข้ารหัสด้วยอัลกอริทึม AES ในแต่ละขนาด input ซึ่งก็คือค่า MAC และใช้คำสั่ง time ในการจับเวลาเพื่อวัดประสิทธิภาพการทำงาน ดังแสดงในภาพที่ 12

```
$ for i in $(ls random10b_*.txt)
; do (echo -n $i ; time openssl
enc -e -aes-256-cbc -K 0123456
789ABCDEF0123456789ABCDEF01234
56789ABCDEF0123456789ABCDEF -iv
00000000000000000000000000000
00 -in $i | tail -c 32 | od -An
-tx1) ; done
```

ภาพที่ 12 *การทดสอบ CBC-MAC-AES256*

จากวิธีดำเนินการวิจัยข้างต้น สามารถสรุปขนาดของข้อมูล อัลกอริทึมและขนาดของคีย์ที่นำมาทำการทดสอบและจำนวนรอบที่ทำการทดสอบในแต่ละขนาดข้อมูลได้ดังแสดงในตารางที่ 1

ตารางที่ 1 *ขนาดของข้อมูล อัลกอริทึมและขนาดคีย์ที่ใช้ในการทดสอบ*

ขนาดข้อมูล	จำนวนรอบ	อัลกอริทึม	ขนาดคีย์ (บิต)
10 - 100 ไบต์	ขนาดข้อมูลละ 20 รอบ (20 ชุดข้อมูล)	ECDSA	256
		HMAC-SHA256	128
		HMAC-SHA512	128
		CBC-MAC-AES128	128
		CBC-MAC-AES256	256

นอกจากนี้คำสั่งที่ใช้ในการทดสอบของอัลกอริทึม AES จะต้องมีการระบุคำสั่งบล็อกสุดท้ายของการเข้ารหัสในแต่ละขนาด input ซึ่งจะนำมาเป็นค่า MAC โดยสามารถสรุปคำสั่งที่ข้อมูลแต่ละขนาดใช้ได้ดังแสดงในตารางที่ 2

ตารางที่ 2 *ขนาดของ IV และ Option เพิ่มเติมของแต่ละขนาดข้อมูลในการทดสอบอัลกอริทึม AES*

ขนาดข้อมูล	IV (บิต)	Option เพิ่มเติม
10 - 30 ไบต์	128	tail -c 32
40 - 50 ไบต์	128	tail -c 64
60 - 90 ไบต์	128	tail -c 96
100 ไบต์	128	tail -c 128

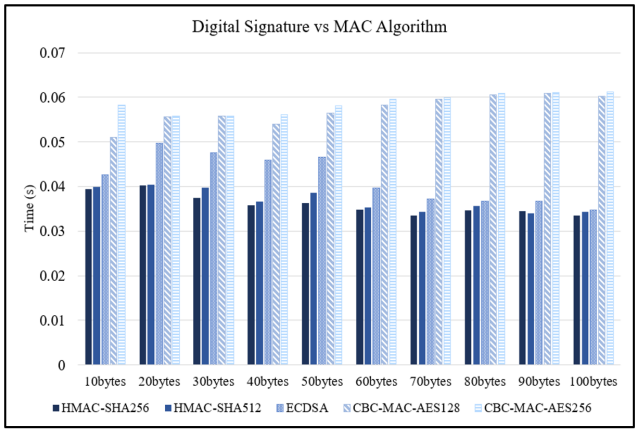
4. ผลการดำเนินงาน

ผลการเปรียบเทียบเวลาในการดำเนินการของอัลกอริทึม ECDSA, HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128 และ CBC-MAC-AES256 ดังแสดงในตารางที่ 3 โดยแสดงเวลาเฉลี่ยของข้อมูลแต่ละขนาด ขนาดละ 20 ชุดข้อมูลและมีหน่วยของเวลาเป็นวินาที (s)

จากตารางที่ 3 สามารถแสดงข้อมูลเปรียบเทียบเวลาในการดำเนินการของทั้ง 5 อัลกอริทึม โดยเฉลี่ยเวลาในการดำเนินการของข้อมูลแต่ละขนาด ขนาดละ 20 ชุดข้อมูลและมีหน่วยของเวลาเป็นวินาที (s) ในรูปแบบกราฟได้ดังภาพที่ 13

ตารางที่ 3 *เวลาในการดำเนินการโดยเฉลี่ยของข้อมูลแต่ละขนาดของ DSA และ MAC*

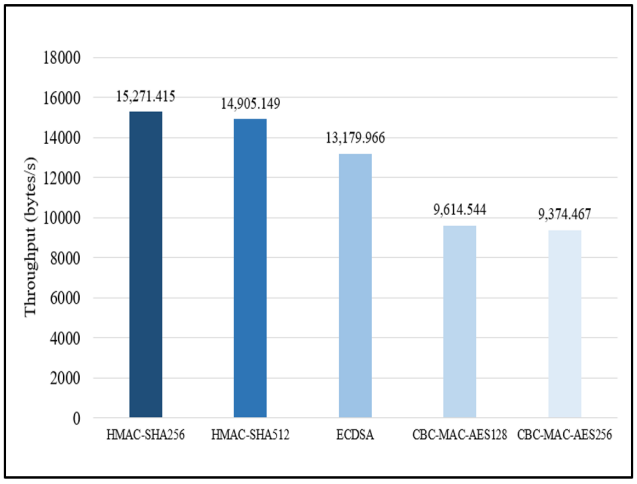
ขนาดข้อมูล (ไบต์)	เวลาที่ใช้ (วินาที)				
	EC DSA	HMAC-SHA 256	HMAC-SHA 512	CBC-MAC-AES 128	CBC-MAC-AES 256
10	0.0427	0.0394	0.0399	0.0511	0.0582
20	0.0497	0.0403	0.0405	0.0556	0.0558
30	0.0476	0.0375	0.0397	0.0559	0.0559
40	0.0459	0.0358	0.0367	0.0539	0.0562
50	0.0467	0.0363	0.0386	0.0564	0.0582
60	0.0396	0.0348	0.0353	0.0583	0.0595
70	0.0372	0.0336	0.0344	0.0595	0.0600
80	0.0366	0.0347	0.0357	0.0605	0.0609
90	0.0368	0.0345	0.0341	0.0609	0.0611
100	0.0347	0.0335	0.0343	0.0602	0.0613
รวม	**0.0417**	**0.0360**	**0.0369**	**0.0572**	**0.0587**



ภาพที่ 13 *เวลาในการดำเนินการโดยเฉลี่ยของข้อมูลแต่ละขนาดของ DSA และ MAC*

และเมื่อโดยพิจารณาจากค่า Throughput ดังแสดงในภาพที่ 14 พบว่า อัลกอริทึม HMAC-SHA256 มีความสามารถในการดำเนินการที่รวดเร็วกว่าอัลกอริทึมอื่น ๆ

เนื่องจากมีค่า Throughput สูงที่สุดคือ 15271.415 ไบต์ต่อวินาที รองลงมาคืออัลกอริทึม HMAC-SHA512 โดยใช้เวลาในการดำเนินการ 14905.149 ไบต์ต่อวินาที อัลกอริทึม ECDSA ใช้เวลา 13179.966 ไบต์ต่อวินาที อัลกอริทึม CBC-MAC-AES128 ใช้เวลา 9614.544 ไบต์ต่อวินาที และอันดับสุดท้าย คือ อัลกอริทึม CBC-MAC-AES256 โดยใช้เวลาในการดำเนินการ 9374.467 ไบต์ต่อวินาที



ภาพที่ 14 *ค่า Throughput ของ DSA และ MAC*

5. สรุป

จากการศึกษาและเปรียบเทียบประสิทธิภาพด้านเวลาในการทำงานของอัลกอริทึมลายเซ็นดิจิทัล ได้แก่ ECDSA และ MAC อัลกอริทึม ได้แก่ HMAC-SHA256, HMAC-SHA512, CBC-MAC-AES128 และ CBC-MAC-AES256 โดยใช้ข้อมูลที่มีขนาดต่างกัน 10 ขนาดข้อมูลในการทดสอบ ได้แก่ ข้อมูลขนาด 10, 20, 30, 40, 50, 60, 70, 80, 90 และ 100 ไบต์ ผลการทดสอบ โดยพิจารณาจากค่า Throughput ดังแสดงในภาพที่ 14 พบว่า อัลกอริทึมที่มีค่า Throughput สูงที่สุด ได้แก่ HMAC-SHA256 รองลงมาคือ HMAC-SHA512, ECDSA, CBC-MAC-AES128 และ CBC-MAC-AES256 ตามลำดับ ดังนั้น อัลกอริทึม HMAC จึงมีความสามารถในการดำเนินการที่รวดเร็วกว่าอัลกอริทึมลายเซ็นดิจิทัล และ CBC-MAC-AES ซึ่งสอดคล้องกับงานวิจัยของ Yoon และ Yoo [16] ที่ทำการ ศึกษาการแลกเปลี่ยนคีย์ระหว่างอัลกอริทึม Diffie-Hellman ร่วมกับอัลกอริทึมลายเซ็นดิจิทัล และอัลกอริทึม Diffie-Hellman ร่วมกับ MAC พบว่าเวลาในการดำเนินการของอัลกอริทึม Diffie-Hellman ที่ทำงานร่วมกับ MAC มีความเร็วกว่าเวลาในการดำเนินการของอัลกอริทึม Diffie-Hellman ที่ทำงานร่วมกับอัลกอริทึมลายเซ็นดิจิทัล นอกจากนี้ยังสอดคล้องกับงานวิจัยของ Werayut และ Nattakarn [11] ซึ่งได้ทำการเปรียบเทียบประสิทธิภาพการทำงานของ MAC อัลกอริทึมระหว่าง HMAC และ CBC-MAC-AES จากการศึกษาพบว่า HMAC ใช้เวลาในการทำงานน้อยกว่า CBC-MAC-AES และ HMAC จะมีประสิทธิภาพการทำงานที่ดีขึ้นเมื่อข้อมูลมีขนาดใหญ่ขึ้น เช่นเดียวกับผลการทดสอบของบทความวิจัยนี้ ซึ่งเมื่อดูจากภาพที่ 13 หากเปรียบเทียบเวลาในการดำเนินการโดยเฉลี่ยของข้อมูลขนาด 10 ไบต์และ 100 ไบต์ของอัลกอริทึม HMAC-SHA256 และ HMAC-SHA512 จะพบว่า ข้อมูลที่มีขนาดใหญ่ขึ้นเวลาในการดำเนินการโดยเฉลี่ยยิ่งลดลง ในขณะที่อัลกอริทึม CBC-MAC-AES128 และ CBC-MAC-AES256 พบว่า ข้อมูลที่มีขนาดใหญ่ขึ้นเวลาในการดำเนินการโดยเฉลี่ยก็เพิ่มขึ้นตามไปด้วย

ดังนั้น อัลกอริทึม HMAC-SHA256 ซึ่งเป็นอัลกอริทึมที่มีค่า Throughput สูงที่สุดจากทั้ง 5 อัลกอริทึมที่ทำการทดสอบ อีกทั้งในกระบวนการทำงานยังมีการใช้ SHA256 ซึ่งเป็นกระบวนการแฮชที่มีความปลอดภัยและได้รับความนิยมในปัจจุบัน จึงมีความเหมาะสมที่สุดในการนำมาใช้ในงานที่ต้องการประสิทธิภาพด้านเวลาในการทำงานในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลและตรวจสอบผู้สร้างข้อความ

สำหรับการศึกษาวิจัยในอนาคต ผู้วิจัยมีความเห็นว่าควรทำการเพิ่มขนาดของข้อมูลที่ใช้ในการทดสอบเพื่อให้เห็นความแตกต่างของเวลาที่ใช้ในแต่ละอัลกอริทึมได้อย่างชัดเจนยิ่งขึ้น และควรเพิ่มจำนวนรอบที่ใช้ในการทดสอบเพื่อให้ผลการดำเนินงานมีความแม่นยำและถูกต้องสมบูรณ์มากยิ่งขึ้น นอกจากนี้ ควรมีการศึกษาอัลกอริทึมอื่น ๆ ที่มีความสามารถในการตรวจสอบความถูกต้องสมบูรณ์ของข้อมูลและผู้สร้างข้อความเพิ่มเติม เช่น อัลกอริทึมลายเซ็นดิจิทัลกับ RSA อัลกอริทึม HMAC กับ SHA384 อัลกอริทึม GMAC เป็นต้น [3] เพื่อให้เห็นความแตกต่างของประสิทธิภาพด้านเวลาในการทำงานของอัลกอริทึมต่าง ๆ และสามารถเลือกอัลกอริทึมที่เหมาะสมที่สุดในการนำมาใช้ได้

6. เอกสารอ้างอิง

[1] J. Andress. *The Basics of Information Security:*

Understanding the Fundamentals of InfoSec in Theory and Practice. Syngress, 2014.

[2] S. Puangpronpitag, and N. Sriwiboon. "Solving the Problem of Public Key Distribution for Digital Signature." *J Sci Technol MSU*, Vol. 35, No. 5, 2016.

[3] E. Barker. "Recommendation for Key Management: Part 1 – General." *National Institute of Standards and Technology, NIST Special Publication (SP) 800-57 Part 1 Rev. 5*, May 2020. doi: 10.6028/NIST .SP.800-57 pt1r5.

[4] Q. Dang. "Recommendation for Applications Using Approved Hash Algorithms." *National Institute of Standards and Technology, NIST Special Publication (SP) 800-107 Rev. 1*, August. 2012. doi: 10.6028/ NIST.SP.800-107r1.

[5] National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code (HMAC)." *National Institute of Standards and Technology*, July. 2008. doi: 10.6028/NIST.FIPS. 198-1.

[6] D. Johnson, A. Menezes, and S. Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)." *International Journal of Information Security*, Vol. 1, No. 1, pp. 36–63, August, 2001.

[7] S. M. Swain, D. A. Pradhan, and S. K. Moharana. "A COMPARATIVE STUDY ON DIGITAL SIGNATURE SCHEMES," *International Journal of Current Science (IJCSPUB),* Vol. 12, No. 1, pp. 498-504, March, 2022.

[8] R. Singla, N. Kaur, D. Koundal, and A. Bharadwaj. "Challenges and Developments in Secure Routing Protocols for Healthcare in WBAN: A Comparative Analysis." *Wireless Pers Commun*, Vol. 122, No. 2, pp. 1767–1806, January, 2022.

[9] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. "Secure Two-party Threshold ECDSA from ECDSA Assumptions," *2018 IEEE Symposium on Security and Privacy (SP)*, May, pp. 980–997, 2018.

[10] M. Nursalman, P. R. R. Judie, and A. Ashshiddiqi. "Implementation of AES and ECDSA for Encrypted Message in Instant Messaging Application." *2020 6th International Conference on Science in Information Technology (ICSITech)*, pp. 165–170, 2020.

[11] W. Shutimarrungson, and N. Shutimarrungson. "Study on Performance of Hash Function of HMAC-MD5, HMAC-SHA-1 and CBC-MAC-AES Algorithms." *The 5th National Conference on Technology and Innovation Management (NCTIM 2019)*, Maha Sarakharm, Thailand, 5 March 2019, pp. 96-103, 2019.

[12] H. Krawczyk, M. Bellare, and R. Canetti. *RFC2104: HMAC: Keyed-Hashing for Message Authentication*. USA: RFC Editor, 1997.

[13] K. Alghathbar, and A. Hafez. "The Use of NMACA Approach in Building a Secure Message Authentication Code." *International Journal of Education and Information Technologies*, Vol. 3, pp. 187-195, 2009.

[14] J. Black, and P. Rogaway. "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions." *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, Berlin, Heidelberg, pp. 197–215, 2000.

[15] M. Bellare, J. Kilian, and P. Rogaway. "The Security of the Cipher Block Chaining Message Authentication Code." *Journal of Computer and System Sciences*, Vol. 61, No. 3, pp. 362–399, December, 2000.

[16] E.-J. Yoon, and K.-Y. Yoo. "An Efficient Diffie-Hellman-MAC Key Exchange Scheme." *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pp. 398–400, 2009.