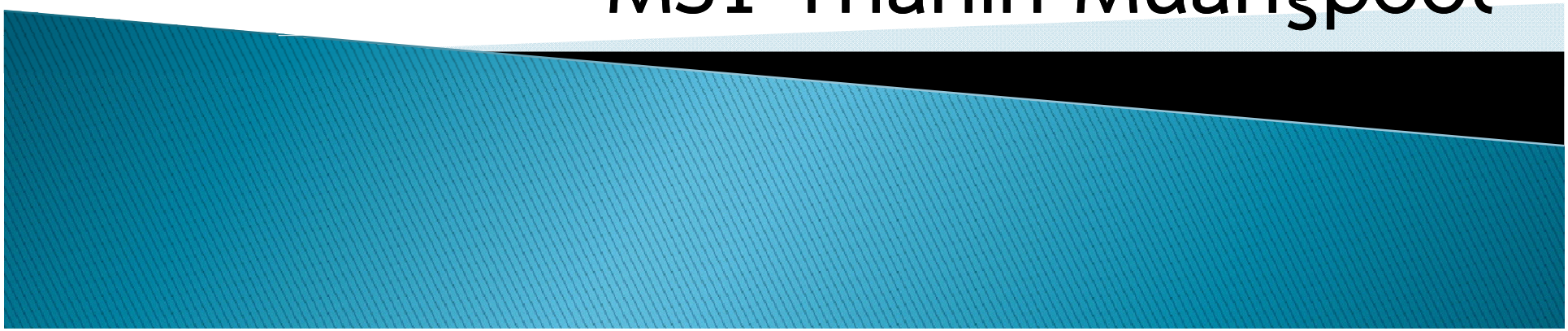
Encryption Algorithm

MS1 Thanin Muangpool



What is a Encryption ?

- คือ การนำข้อมูล Plain Text ที่ได้จากผู้ส่งมา Encryption หรือการสร้างรหัสลับขึ้นมาด้วย Secret Key กลายเป็นข้อมูล Cipher Text จากนั้นส่งไปให้ปลายทางรับข้อมูล ผู้ได้รับข้อมูลจะทำการ Decryption ด้วย Secret Key เช่นกัน โดยทั้ง 2 ขบวนการต้องอาศัย Algorithm เข้ามาช่วย วิธีการ หรือ Algorithm สามารถแบ่งออกเป็น 2 กลุ่ม ดังนี้

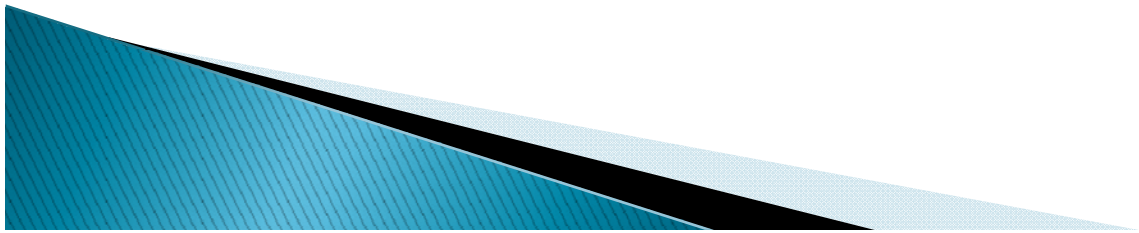
  1. Symmetric Encryption

  2. Asymmetric Encryption

- จุดประสงค์ของการเข้ารหัสและถอดรหัส
  - ผู้ที่ไม่ได้รับอนุญาตจะไม่สามารถเข้าถึงข้อมูลได้ หรือ เข้าถึงข้อมูลได้ยาก

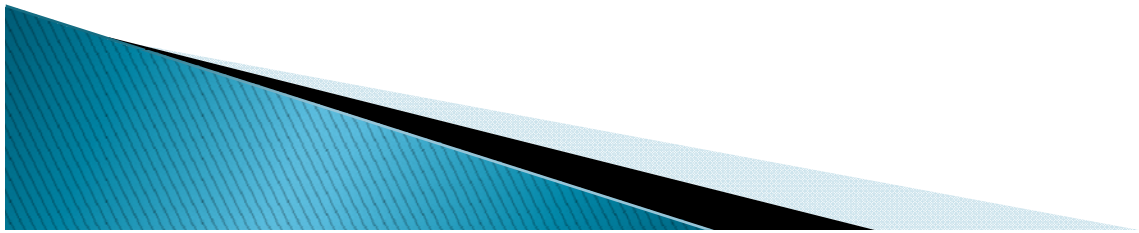
Symmetric Key Cryptography

- การเข้ารหัสโดยใช้กุญแจดอกเดียวเรียกว่า "การเข้ารหัสแบบสมมาตร" (Symmetric Key Cryptography) เนื่องจากใช้ Key ตัวเดียวกันในการเข้ารหัสและถอดรหัส นอกจาก DES และ AES แล้วอัลกอริทึมในการเข้ารหัสแบบสมมาตรอย่างอื่นก็มีเช่น Blowfish และ IDEA แต่อาจจะไม่เป็นที่นิยมมากนัก การเข้ารหัสแบบสมมาตรถึงแม้จะใช้อัลกอริทึมที่แข็งแกร่งอย่าง AES และใช้ Key ที่มีความยาวตั้งแต่ 128 bit ขึ้นไปแล้วก็ยังมีข้อด้อยอยู่ตัวอย่างเช่น



Symmetric Key Cryptography

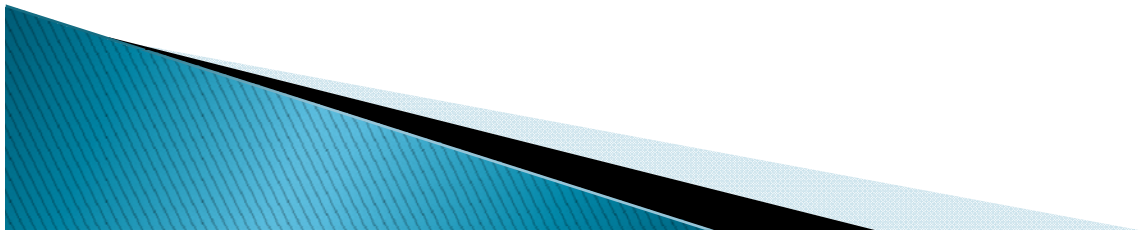
- การส่ง Key ไปยังผู้รับเพื่อใช้ในการถอดรหัส หาก Key ถูกดักจับได้แล้ว การเข้ารหัสก็จะไม่มีความหมายอะไรเลยเพราะผู้ดักฟังที่ดักจับได้ Key ไปก็สามารถที่จะดักจับ Cipher Text แล้วถอดรหัสได้
- หากมีจำนวนผู้ใช้มากขึ้น ซึ่งผู้ใช้แต่ละคู่จะต้องใช้คีย์ที่แตกต่างจากคู่สื่อสารอื่น จะทำให้จำนวนคีย์ที่ต้องใช้ทั้งหมดมีจำนวนมากเช่น ผู้ใช้ N คนจะต้องใช้คีย์ทั้งหมดเท่ากับ N x (N-1) / 2



Symmetric Key Cryptography

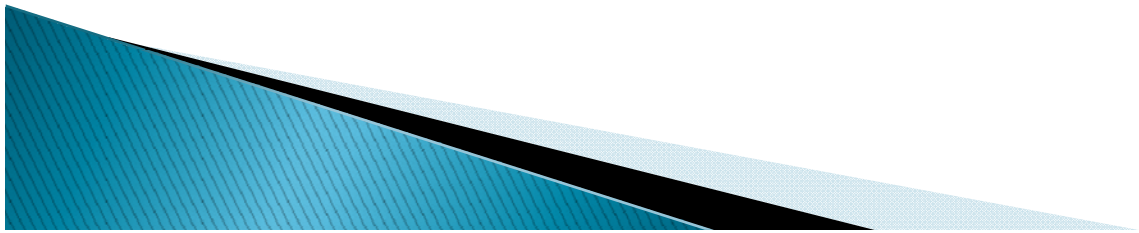
DES (Data Encryption Standard)

DES เป็นการเข้ารหัสแบบ Block Cipher ที่พัฒนามาจากอัลกอริทึม Lucifer ของ IBM โดย Lucifer ได้รับการพัฒนาเพิ่มความสามารถและเปลี่ยนชื่อเป็น DES แล้วได้รับการนำเสนอต่อ US NIST (US National Institute of Standards and Technology) ให้กลายเป็นมาตรฐานของการเข้ารหัส

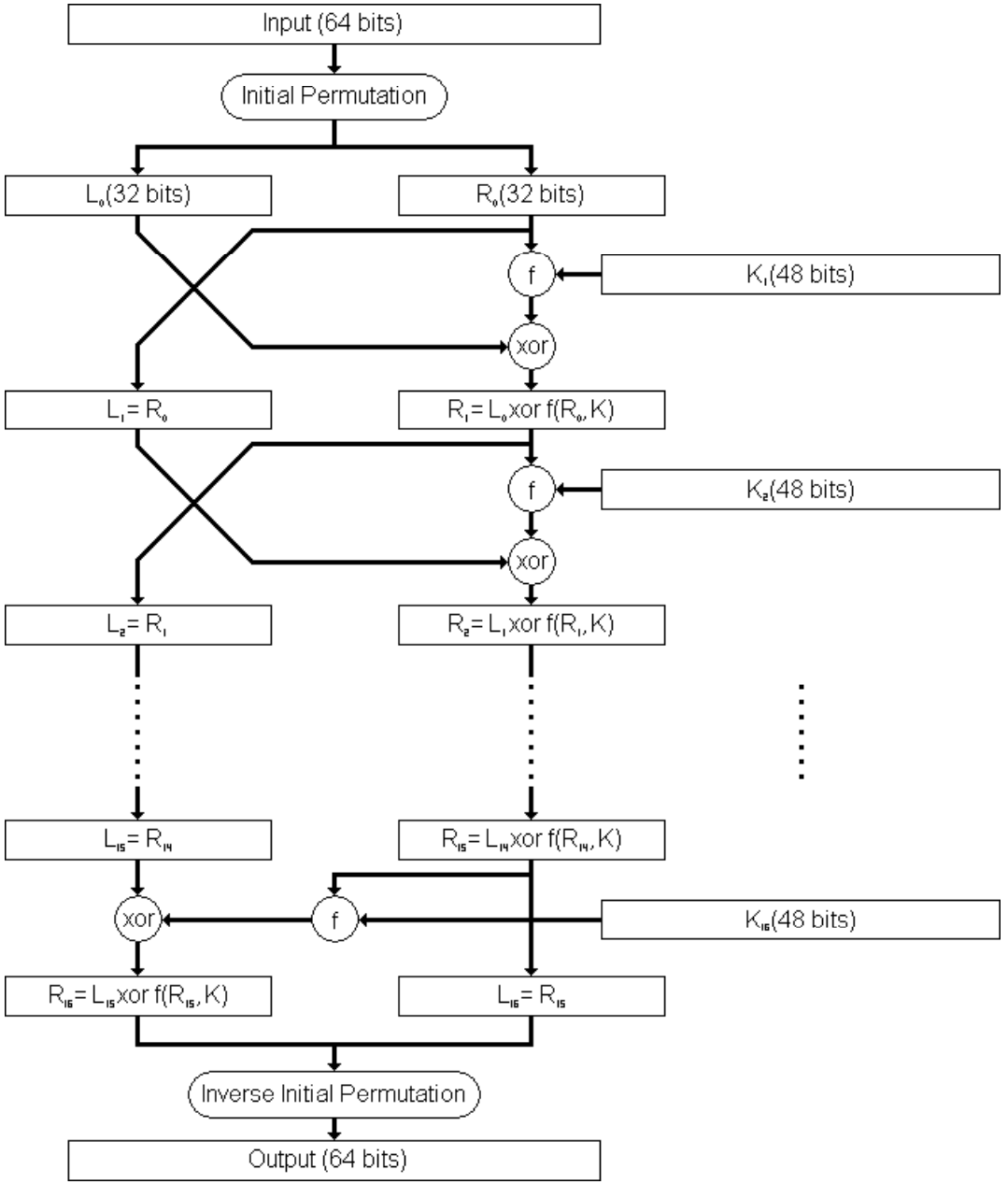


DES (Data Encryption Standard)

การเข้ารหัสข้อมูลแบบ DES เป็นการเข้ารหัสโดยกระทำกับกลุ่มของข้อมูลขนาด 64 บิต ลำดับแรกข้อมูล 64 บิตนี้จะถูกสลับตำแหน่ง (สลับบิต) จากนั้นจะถูกแบ่งเป็น 2 ส่วนได้แก่ส่วนทางซ้ายและส่วนทางขวา (ส่วนละ 32 บิต) ขั้นตอนต่อไปจะใช้ฟังก์ชั่นทางคณิตศาสตร์ (ฟังก์ชั่น f) ข้อมูลจากส่วนซ้ายหรือขวาจะถูกนำมารวมกันกับ Key โดยจะทำซ้ำกันอย่างนี้เป็นจำนวนทั้งสิ้น 16 รอบ เมื่อเสร็จสิ้นขั้นตอนนี้ (รอบที่ 16) ผลลัพธ์ที่ได้จากทั้งส่วนทางซ้ายและขวาก็จะถูกนำมารวมกันเป็นข้อมูลขนาด 64 บิตอีกครั้งหนึ่ง และนำไปสลับตำแหน่งในขั้นตอนสุดท้าย



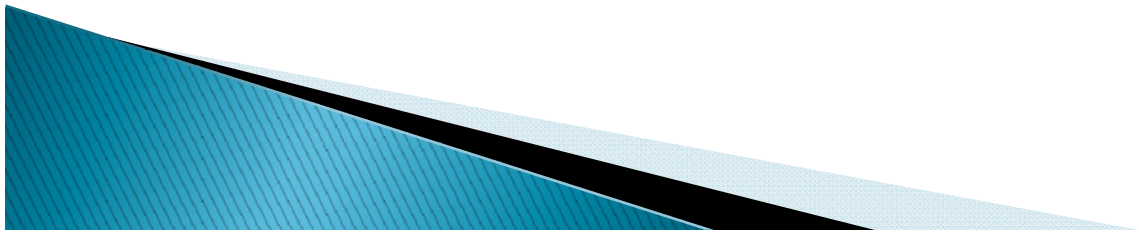
กระบวนการทำงานของ DES



DES (Data Encryption Standard)

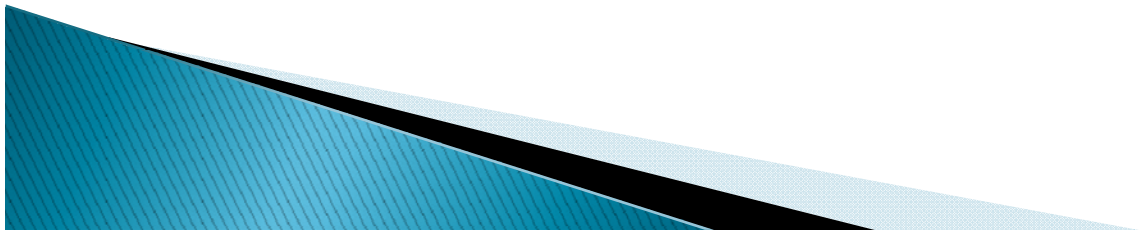
การทำงานของฟังก์ชั่น f ในแต่ละรอบ จะเป็นการเลื่อนบิตของ Key ซึ่งจะเลือกใช้เพียง 48 บิตจากทั้งสิ้น 56 บิต ข้อมูลในส่วนทางขวา (32 บิต) จะถูกขยายให้กลายเป็นข้อมูลขนาด48 บิต

จากนั้นจะนำมารวมกับกุญแจขนาด 48 บิต (ที่เลือกมา) การรวมกันในขั้นตอนนี้จะใช้การ XOR ผลลัพธ์ขนาด 48 บิตที่ได้จะถูกนำไปทำการแทนที่อีก 8 ครั้ง ผลลัพธ์จากการแทนที่จะเหลือข้อมูลเพียง 32 บิตเท่านั้น หลังจากนั้นก็จะต้องทำการสลับตำแหน่งกันอีกครั้ง

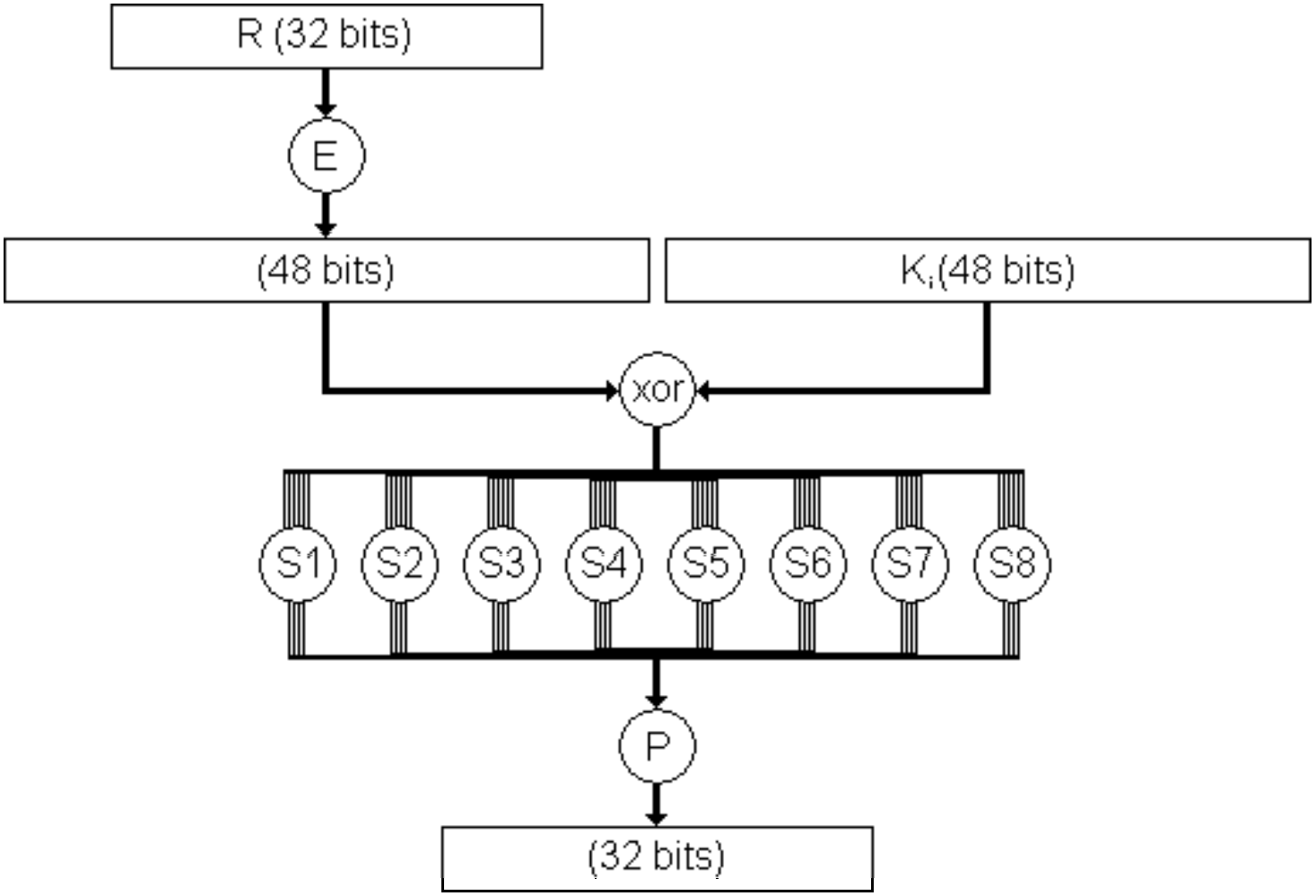


DES (Data Encryption Standard)

หนึ่งรอบของการทำฟังก์ชั่น f จะประกอบด้วยขบวนการข้างต้น 4 ครั้ง ข้อมูลในส่วนทางซ้ายจะต้องผ่านขบวนการเดียวกัน ผลลัพธ์ที่ได้จากทั้งส่วนทางซ้ายและขวาจะถูกนำมารวมกันแบบ XOR เมื่อเสร็จสิ้นขั้นตอนนี้แล้ว ผลลัพธ์ที่ได้จะถูกใช้เป็นข้อมูลส่วนทางขวาของรอบใหม่ และข้อมูลของส่วนทางขวาเดิมก็จะกลายเป็นข้อมูลส่วนซ้ายของวงรอบใหม่



การทำงานของฟังก์ชั่น f ของ DES



DES (Data Encryption Standard)

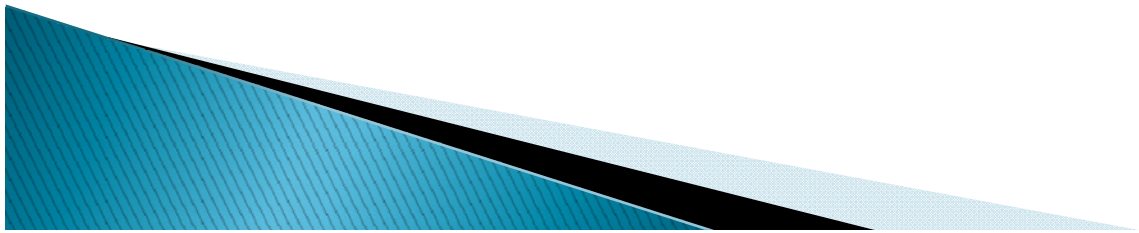
บริษัทแห่งหนึ่งต้องการเบรค DES เพื่อสร้างความแข็งแกร่งให้กับ RSA จึงจัดให้มีการประกวดการเบรคขึ้นโดยให้รางวัล 10,000 US$ สำหรับผู้ชนะในแต่ละรอบ

- บริษัท Distribution.net ใช้เวลา 41 วันก็ทำการเบรค DES ได้สำเร็จ
- บริษัท EFF สามารถ เบรค ได้ภายในเวลา 56 ชั่วโมง

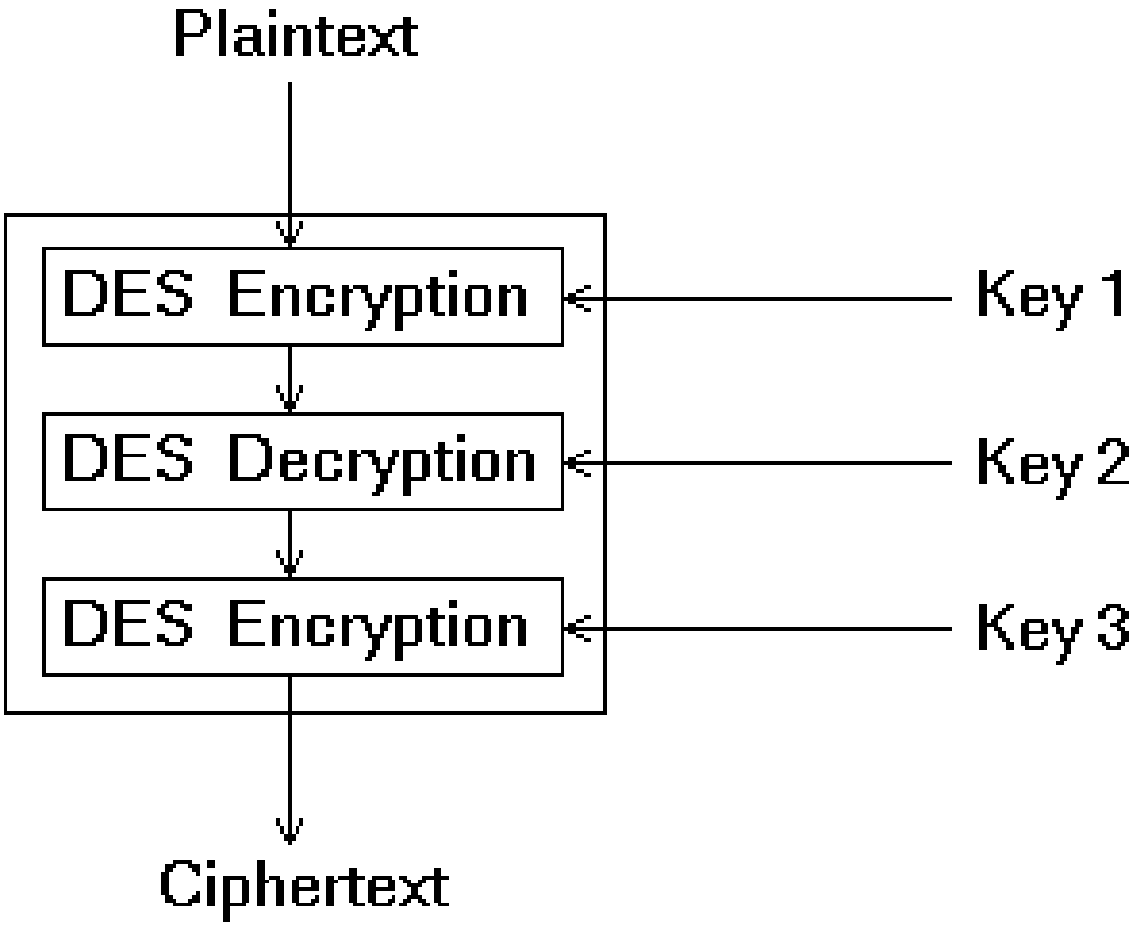
Distribution.net และบริษัท EFF ก็จับมือกันและใช้คอมพิวเตอร์กว่า 100,000 เครื่องทั่วโลกมาแคร็ก DES ซึ่งก็สามารถทำได้ ในเวลา 22 ชั่วโมง 15 นาที จึงเป็นต้นเหตุทำให้มีการขยาย Key ของ DES จาก 64 Bit ให้เป็น 128 Bit เพื่อจะได้ใช้เวลาในการแคร็กนานขึ้น

Tripple-DES (3DES)

Triple-DES เป็นการเข้ารหัสที่ถูกสร้างมาเพื่อแก้ปัญหาความอ่อนแอของ DES โดย Triple-DES จะช่วยเสริมความปลอดภัยให้การเข้ารหัสมีปลอดภัยมากขึ้น โดยการใช้อัลกอริทึม DES เป็นจำนวนสามครั้งเพื่อทำการเข้ารหัส โดยในแต่ละครั้งจะใช้กุญแจในการเข้ารหัสที่แตกต่างกันออกไป ดังนั้นจำนวนกุญแจที่ใช้ใน Triple-DES จึงมีทั้งสิ้น 3 ดอก (ความยาวดอกละ 56 บิต) ด้วยความแข็งแกร่งนี้จึงทำให้ Triple-DES เป็นอีกหนึ่งในมาตรฐานในการเข้ารหัสในปัจจุบัน



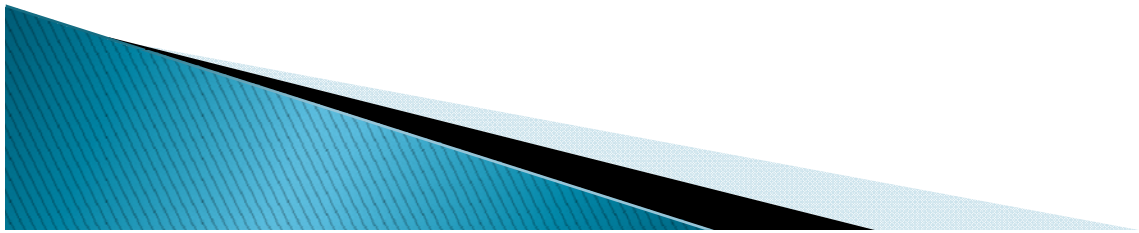
กระบวนการทำงานของ Tripple-DES



AES (Advance Encryption Standard)

AES (Advance Encryption Standard) เป็นการเข้ารหัสที่พัฒนาขึ้นมาเพื่อใช้ทดแทน DES หลังจากที่ DES ถูกเบรคได้โครงการพัฒนา AES ได้เริ่มต้นเมื่อปี 1997 โดย NIST หลังจากนั้น (ในปี 1998) NIST ก็ให้นักวิทยาการหัสลับทั่วโลกส่งอัลกอริทึมเข้ามาเพื่อคัดเลือกโดยกำหนดให้ 128 Bit เป็นมาตรฐานของ และ 256 Bit

อัลกอริทึมต่าง ๆ ถูกคัดเลือกเข้ามาทั้งสิ้น 15 อัลกอริทึม และมีอยู่ 5 อัลกอริทึมที่ผ่านเข้ารอบชิง จนผลสุดท้ายอัลกอริทึมของ Rijndeal ได้รับการตัดสินให้ชนะเพราะเร็วกว่าและใช้อัลกอริทึมที่ธรรมดากว่า แต่ได้ความปลอดภัยเท่ากัน จากนั้นจึงได้กลายเป็น RFC 3826 เมื่อปี 2004 ข้อกำหนดในมาตรฐานล่าสุดอนุญาตให้ใช้ AES เข้ารหัสข้อมูลโดยใช้ Key ที่มีขนาดต่าง ๆ ได้ ซึ่งได้แก่ 128 Bit, 192 Bit และ 256 Bit



AES (Advance Encryption Standard)

วงรอบการทำงานของ AES แบ่งเป็น 3 ส่วนหลัก ๆ ได้แก่ Initial Round, Rounds และ Final Round และในแต่ละส่วนก็มีกระบวนการย่อยต่าง ๆ ดังนี้

(1) Initial Round

- AddRoundKey

(2) Rounds

- SubBytes: เป็น non-linear substitution ซึ่งแต่ละไบต์จะถูกแทนที่ด้วยไบต์ที่ได้จาก lookup table (รูปที่ 7)

- ShiftRows: เป็นการเลื่อนไบต์ในแต่ละแถว ซึ่งจะทำเฉพาะแถวที่ 2, 3 และ 4

AES (Advance Encryption Standard)

- MixColumns: เป็นการผสมรวม 4 ไบต์ภายในคอมลัมน์

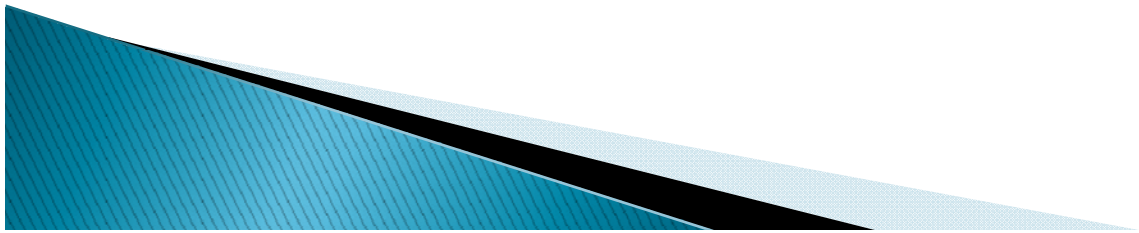
- AddRoundKey เป็นการนำ Cipher Text และ Key (ที่มาจาก key schedule) ผสมรวมกลายเป็น Cipher Text ใหม่

(3) Final Round (no MixColumns)

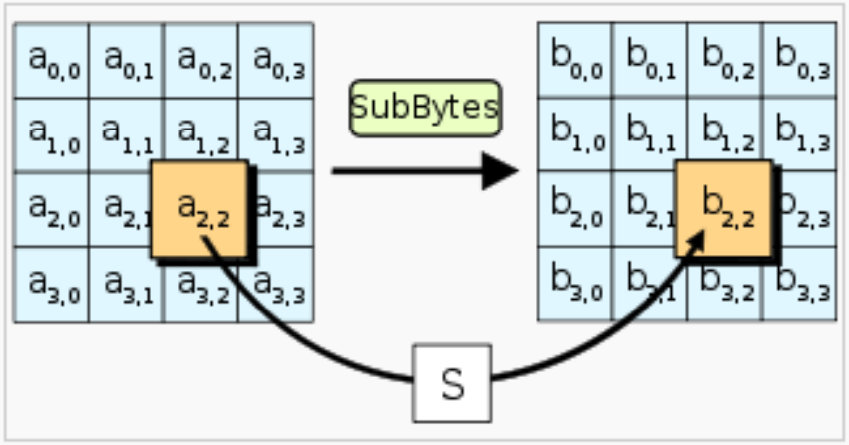
- SubBytes

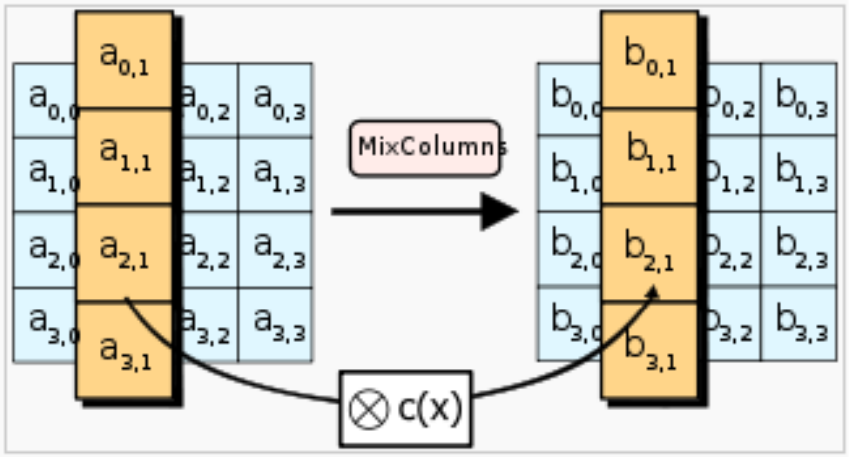
- ShiftRows

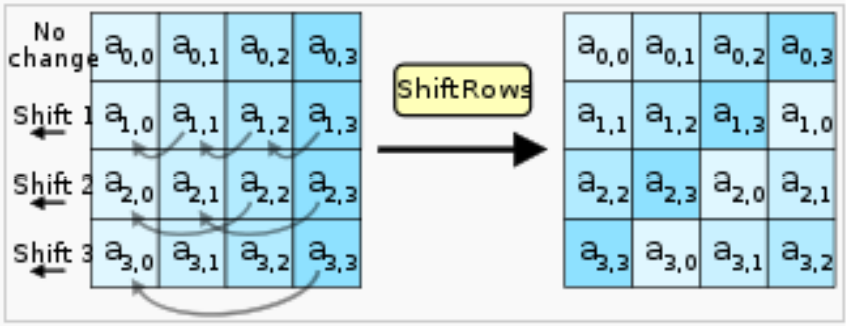
- AddRoundKey

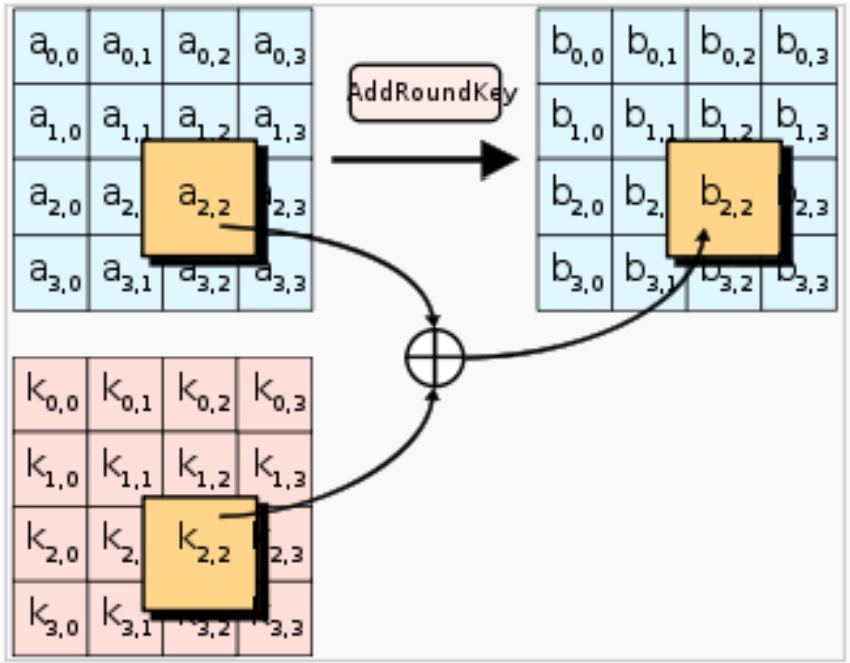


กระบวนการ SubBytes, ShiftRows, MixColumns และ AddRoundKey

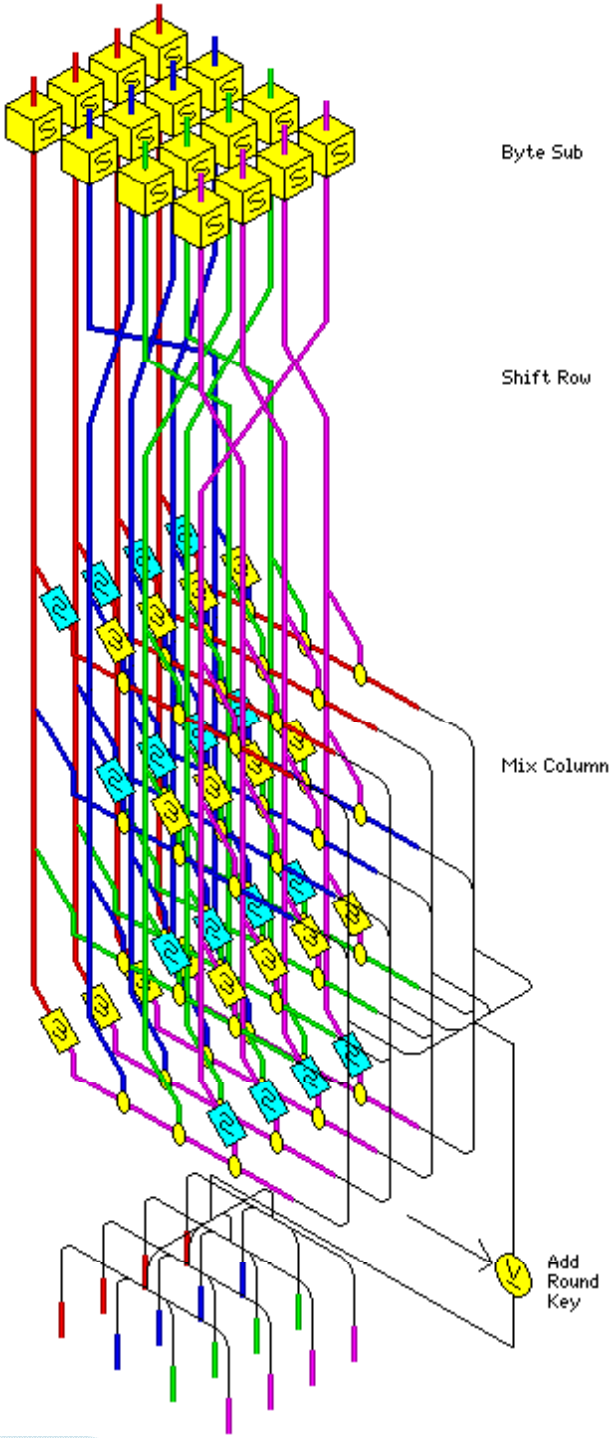








วงรอบการทำงานของ AES



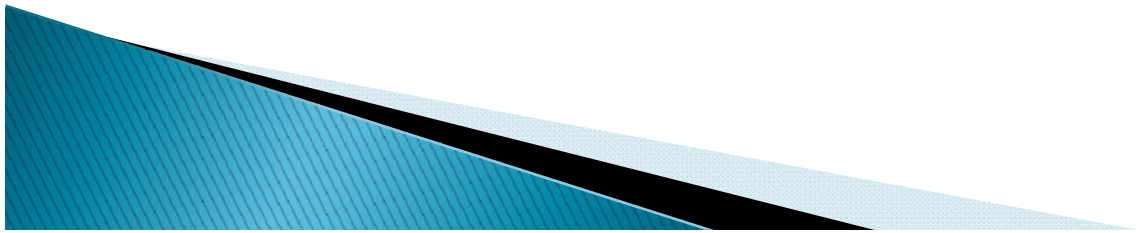
Asymmetric Key Cryptography

- การเข้ารหัสแบบอสมมาตร (Asymmetric Key Cryptography) บางตำราอาจใช้คำว่า Asymmetric Key Encryption หรือ Public Key Encryption หรือใช้คำว่า Public Key me Infrastructure (PKI) หรือ Public-Key Cryptography
- การเข้ารหัสแบบนี้ถูกคิดค้นโดย Whit Diffie และ Marty Hellman ตั้งแต่ปี 1976 โดยถูกสร้างมาเพื่อเป็นทางเลือกในการส่งข้อมูลที่เป็นความลับ เพราะการเข้ารหัสแบบสมมาตร (ใช้กุญแจดอกเดียว) จะมีปัญหาเรื่องการถูกดักจับ Key และปัญหาเกี่ยวกับการจัดการ Key ที่มีอยู่เป็นจำนวนมากเมื่อใช้ในระบบใหญ่ การเข้ารหัสแบบอสมมาตรจะใช้ Key สองอัน โดยหากเราเข้ารหัสด้วย Key อันหนึ่งจะต้องทำการถอดรหัสด้วย Key อีกอันหนึ่งที่เหลือ

Asymmetric Key Cryptography

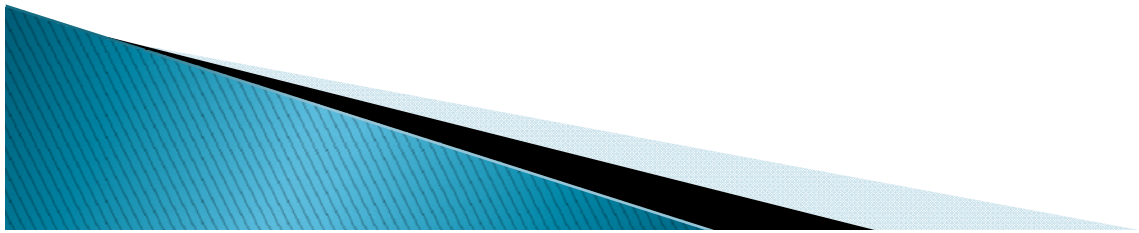
ตัวอย่างเช่น

- หากเข้ารหัสด้วย Key1 จะต้องถอดรหัสด้วย Key2 เท่านั้น
- หากเข้ารหัสด้วย Key2 จะต้องถอดรหัสด้วย Key1 เท่านั้น
- หากเข้ารหัสด้วย Key1 แล้วถอดรหัสด้วย Key1 จะไม่สามารถถอดรหัสได้
- หากเข้ารหัสด้วย Key2 แล้วถอดรหัสด้วย Key2 จะไม่สามารถถอดรหัสได้



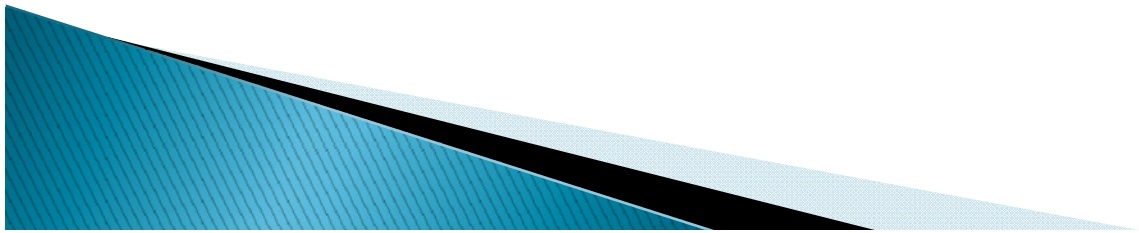
Asymmetric Key Cryptography

- การประยุกต์ใช้งานทำได้โดย เก็บ Key อันหนึ่งไว้กับตัวเองเรียกว่า Private Key ส่อนอีก Key หนึ่งสามารถที่จะแจกจ่ายให้ผู้อื่นได้ดังนั้น Key นี้จึงถูกเรียกว่า Public Key เมื่อผู้อื่นต้องการที่จะส่งข้อมูลที่เป็นความลับมายังเจ้าของ Private Key จะต้องทำการเข้ารหัสข้อมูลนั้นด้วย Public Key ของผู้รับ ดังนั้นจึงทำให้ผู้ที่มี Private Key เท่านั้นที่จะถอดรหัสข้อมูลได้ ส่วนการส่งข้อมูลที่เข้ารหัสด้วย Private Key ไปยังผู้อื่น ผู้ใดก็ตามที่มี Public Key (ซึ่งมีอยู่หลายคน) จะสามารถถอดรหัสข้อมูลได้



Asymmetric Key Cryptography

RSA เป็นอัลกอริทึมในการเข้าหัสแบบอสมมาตร ถูกสร้างขึ้นมาเมื่อปี1978 โดย Ron **R**ivest, Adi **S**hamir และ Leonard **A**dleman ตั้งแต่คิดค้นมายังไม่มีใครสามารถเบรคอัลกอริทึมนี้ได้ และ RSA ได้ถูกนำมาใช้อย่างแพร่หลายในด้าน e-commerce



กระบวนการทำงานของ RSA

(1) เลือก p และ q ซึ่งเป็นจำนวนเฉพาะที่มีค่าต่างกัน

(2) ให้ n = pq

(3) ให้ m = (p-1)(q-1)

(4) เลือกค่า e ที่ 1 < e < m ซึ่งหารร่วมมากของ m กับ e มีค่าเป็น 1 (สามารถหาค่า e ได้โดยการส่มค่าจำนวนเต็มบวกพร้อมกับทดสอบว่าหารร่วมมากของ m กับ e มีค่าเป็น 1)

(5) หาค่า d ที่ทำให้ ed mod m = 1

(6) Public key คือ (e,n)

(7) Private key คือ (d,n)

(8) ให้ M คือข้อความที่ยังไม่ถูกเข้ารัหส (ในรูปแบบของตัวเลข) M < n

(9) การเข้ารหัส => C = M^e mod n

(10) การถอดรหัส => M = C^d mod n

ตัวอย่างการเข้ารหัสและถอดรหัสด้วย RSA

(1) เลือก p และ q ซึ่งเป็นจำนวนเฉพาะที่มีค่าต่างกัน

p = 7

q = 17

(2) ให้ n = pq

ดังนั้น n = 7*17 = 119

(3) ให้ m = (p-1)(q-1)

ดังนั้น m = 6*16 = 96

(4) เลือกค่า e ที่ 1 < e < m ซึ่งหารร่วมมากของ m กับ e มีค่าเป็น 1

(สามารถหาค่า e ได้โดยการสุ่มค่าจำนวนเต็มบวกพร้อมกับทดสอบว่าหารร่วมมากของ m กับ e มีค่าเป็น 1)

เลือก e = 5 และทดสอบหารร่วมมากของ 96 กับ 5 แล้วได้ 1

(5) หาค่า d ที่ทำให้ ed mod m = 1

ได้ค่า d = 77 เพราะ 5*77 mod 96 ได้ 1

(6) Public key คือ (e,n)

ดังนั้น Public key คือ (5,119)

(7) Private key คือ (d,n)

ดังนั้น Private key คือ (77,119)

(8) ให้ M คือข้อความที่ยังไม่เข้ารหัส (ในรูปแบบของตัวเลข) M < n

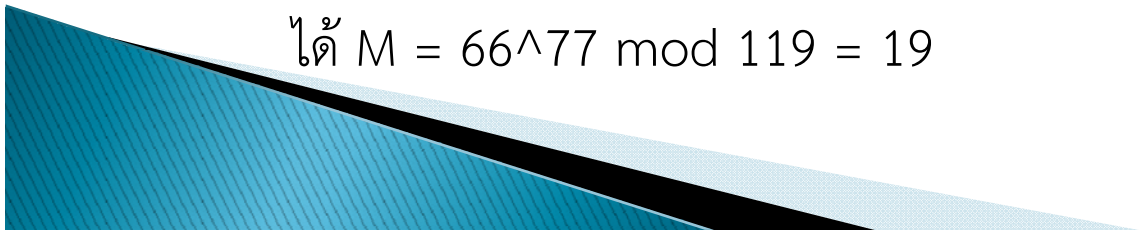
ให้ข้อความที่ยังไม่เข้ารหัส M = 19

(9) การเข้ารหัส => C = M^e mod n

ได้ C = 19^5 mod 119 = 66

(10) การถอดรหัส => M = C^d mod n

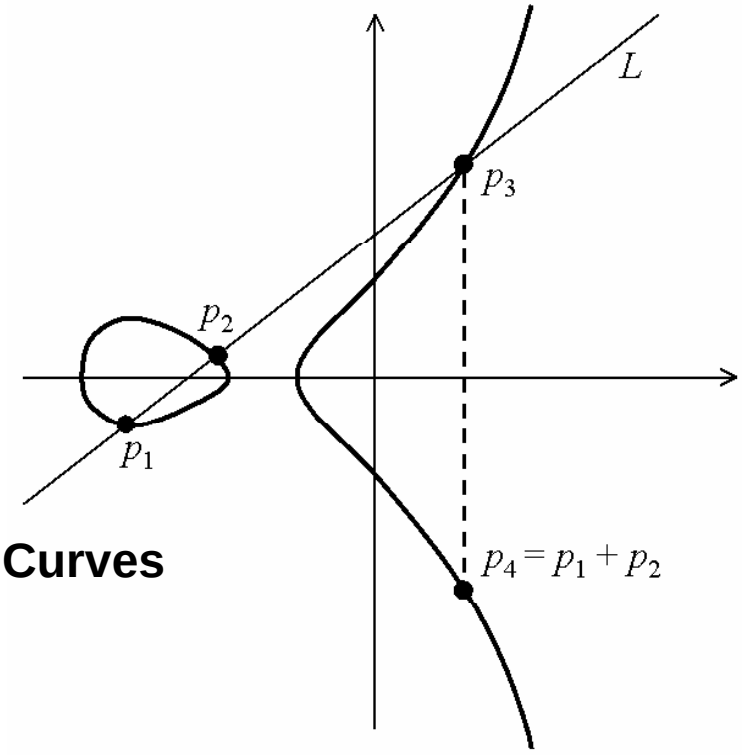
ได้ M = 66^77 mod 119 = 19



Asymmetric Key Cryptography

ECC ย่อมาจาก Elliptic Curves Cryptography ได้รับการนำเสนอโดย Neal Koblitz และ Victor S. Miller ในปี 1985 โดยอัลกอริทึมการเข้ารหัส ECC นี้ได้รับการพัฒนาจากสมการของเส้นโค้งของวงรี

$$y^2 = x^3 + ax + b$$

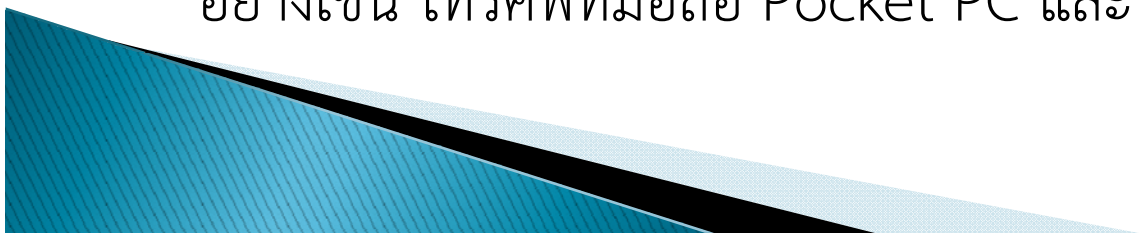


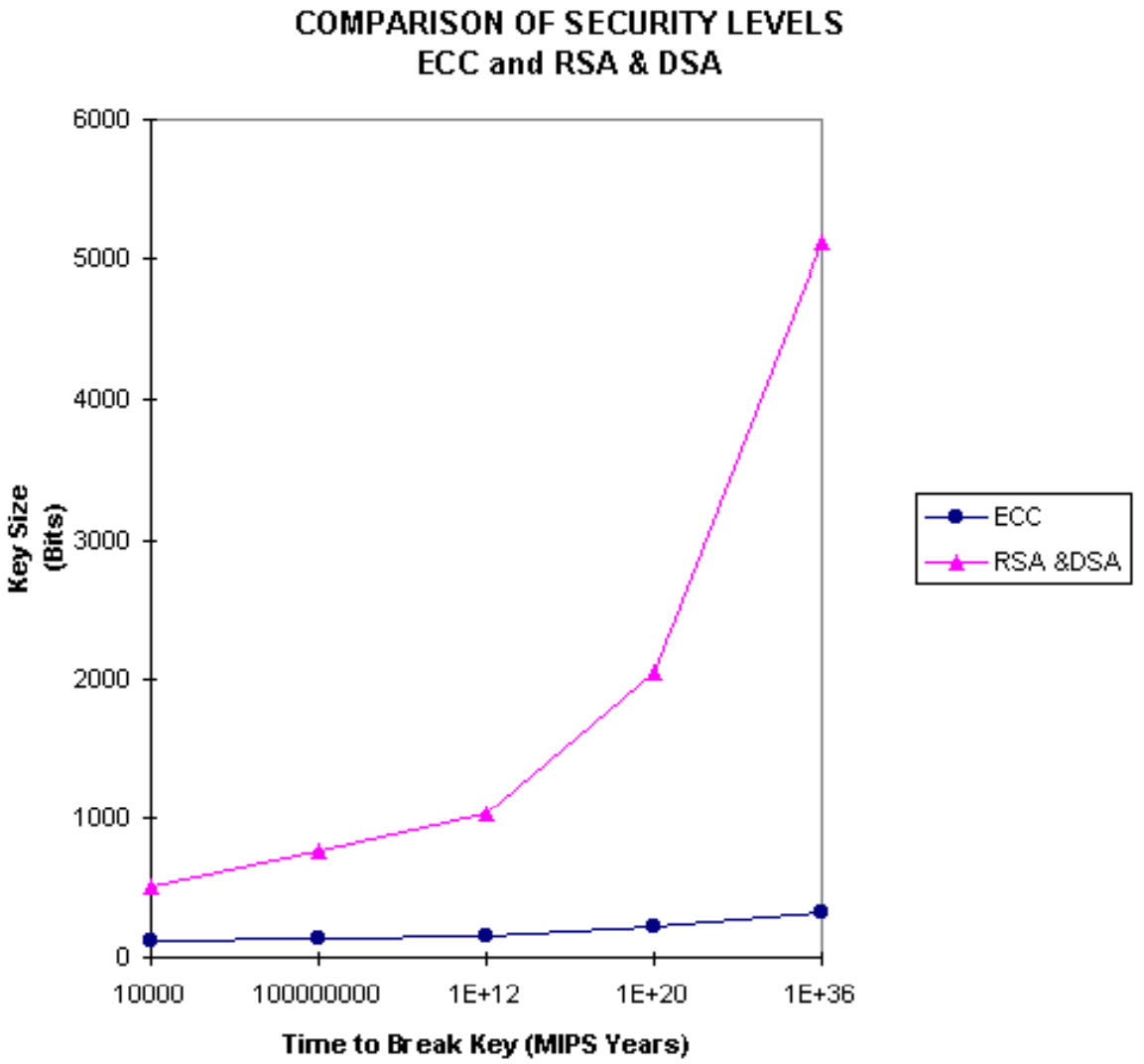
กราฟแสดงความสัมพันธ์ของสมการ Elliptic Curves

Asymmetric Key Cryptography

ECC มีข้อดีที่เหนือกว่า RSA คือจะใช้คีย์ที่สั้นกว่าแต่สามารถให้ความปลอดภัยเท่ากับ RSA ได้ หรือหากใช้คีย์ที่ยาวเท่ากับคีย์ของ RSA จะมีความปลอดภัยสูงกว่า คือหากต้องการที่จะเบรคจะใช้เวลาในการ Brute Force นานกว่า RSA

เนื่องจาก ECC ใช้ Key ที่มีขนาดเล็กกว่า RSA มาก และมีความสามารถในการคำนวณที่รวดเร็ว ใช้พลังงานต่ำและใช้หน่วยความจำน้อย ดังนั้น ECC จึงเหมาะสำหรับการใช้งานในอุปกรณ์เคลื่อนที่ขนาดเล็ก อย่างเช่น โทรศัพท์มือถือ Pocket PC และ PDA เป็นต้น





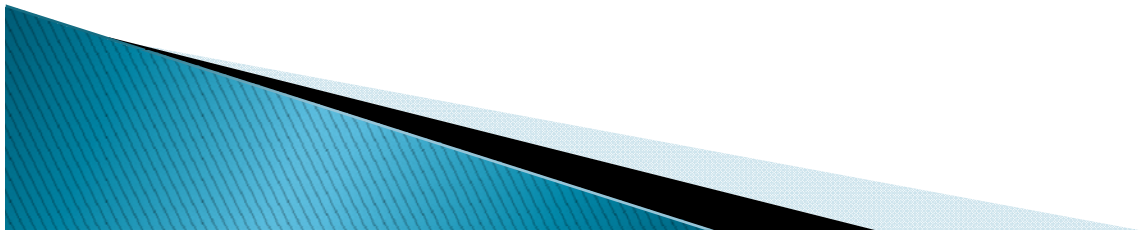
ตารางเปรียบตารางที่ใช้ในการแคร็ก ECC เทียบกับ RSA

Hash Function หรือ One-Way Function

- Hash Function มีประโยชน์อย่างมหาศาลใน Electrical Digital Signature เนื่องจาก Function นี้สามารถนำมาใช้หาลักษณะเฉพาะของข้อมูลที่ต้องการจะส่งได้ หรือเรียกอีกย่างหนึ่งว่าสามารถนำมาใช้หา Finger Print ของข้อมูลที่แตกต่างกันอกไปตามลักษณะของข้อมูลนั้น ๆ
- การเข้ารหัสแบบ Hash (Cryptographic hash) หมายถึง การแปลงรูปแบบของข้อมูลที่รับเข้ามาให้เป็นข้อมูลที่ถูกย่อย (Message Digest) ไม่ว่าข้อมูลต้นฉบับจะมีขนาดเล็กหรือใหญ่เท่าใดก็ตามก็จะถูกย่อยให้อยู่ในรูปแบบที่มีขนาดคงที่ ดังนั้นจึงไม่สามารถทำกระบวนการย้อนกลับเพื่อให้กลายเป็นข้อมูลต้นฉบับได้ จะทำได้เพียงแค่ตรวจสอบว่าข้อมูลที่ให้มาแต่ละครั้งเหมือนกันหรือไม่

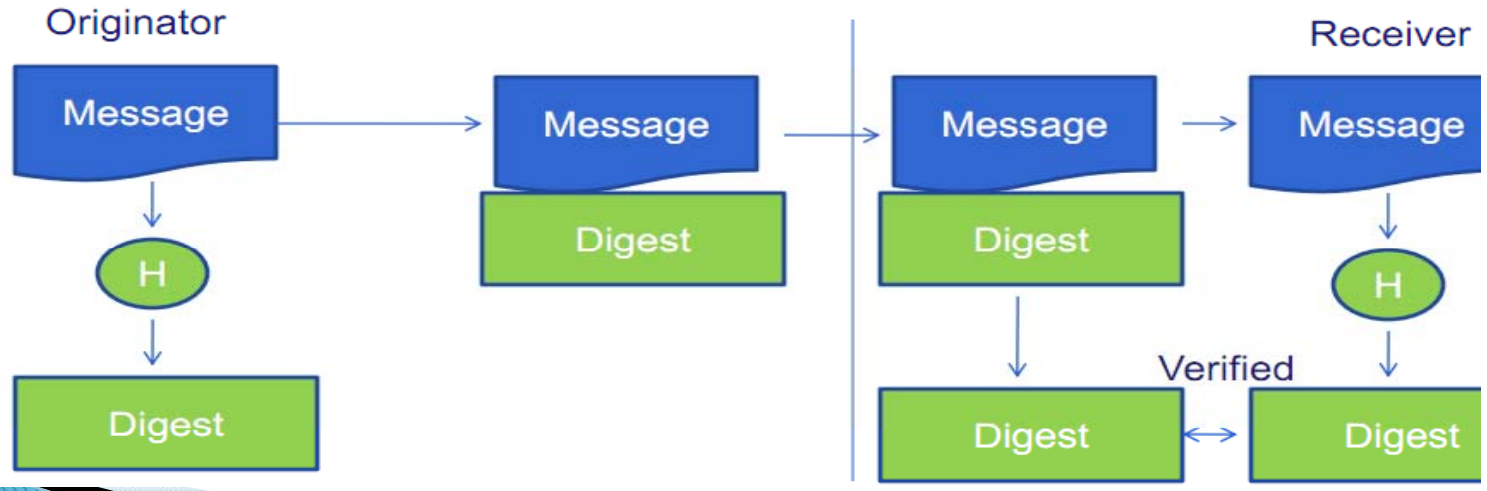
รูปแบบของ Hash Function

- Hash Function ที่แพร่หลายมี 2 รูปแบบ คือ
  - **MD (Message Digest)** เป็น Hash Function ที่ได้รับความนิยมมาก โดยจะนำ Message Digest ที่มีขนาด 128 Bits มา Hash เป็น Digest
  - **SHA-1 (Secure Hashing Algorithm)** เป็นระบบที่ได้รับการรับรองให้ใช้เป็นมาตรฐานโดยรัฐบาลสหรัฐฯ ระบบนี้จะให้ Message Digest ที่มีขนาด 160 Bits ซึ่งมีขนาดใหญ่กว่า MD แต่จะทำงานช้ากว่า โดยปกติแล้ว SHA-1 จะให้ความปลอดภัยที่สูงกว่าระบบ MD



หลักการของ Hash Function

- หลักการคือผู้ส่งจะนำข้อความ Plain Text มาเข้า Hash Function เพื่อให้ได้ Digest แล้วนำมารวมกับ Message และส่งไปให้กับผู้รับ
- ผู้รับทำการแยก Digest และ Plan Text ออกจากกันและจะนำ Plan Text ที่ได้รับเพื่อนำมา Hash Function เพื่อตรวจสอบว่า Digest ที่ได้ตรงกันกับที่ได้รับมาหรือไม่



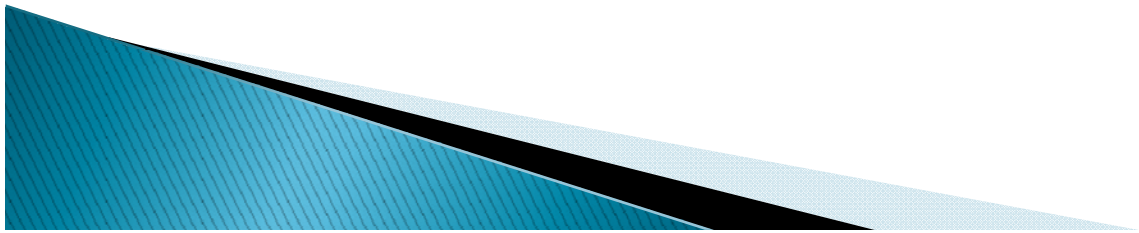
Hash Function

ตัวอย่างเช่นผู้ใช้ Thanin ตั้งรหัสผ่านเป็นคำว่า abc123 หากเก็บรหัสผ่านลงบน Database โดยตรงจะทำให้ผู้ใดก็ตามที่เข้าถึงฐานข้อมูลได้ทราบรหัสผ่านที่เก็บไว้ (ผู้ที่เข้าถึงฐานข้อมูลได้เช่นผู้ดูแลระบบ ผู้ดูและฐานข้อมูล และแฮกเกอร์ที่เจาะเข้ามาทางเว็บไซต์ด้วยวิธีการพิเศษ ตัวอย่างเช่น SQL Injection)

หากทำการย่อยรหัสผ่านด้วยฟังก์ชั่น Hash เช่นใช้ MD5 ย่อยรหัสผ่าน abc123 ได้เป็น e99a18c428cb38d5f260853678922e03 แล้วจึงเก็บค่า Hash นั้นลงใน Database จะทำให้การเปิดดูรหัสผ่านใน Database โดยตรง ไม่พบรหัสผ่าน abc123 แต่จะพบเพียงค่า Hash (e99a18c428cb38d5f260853678922e03)

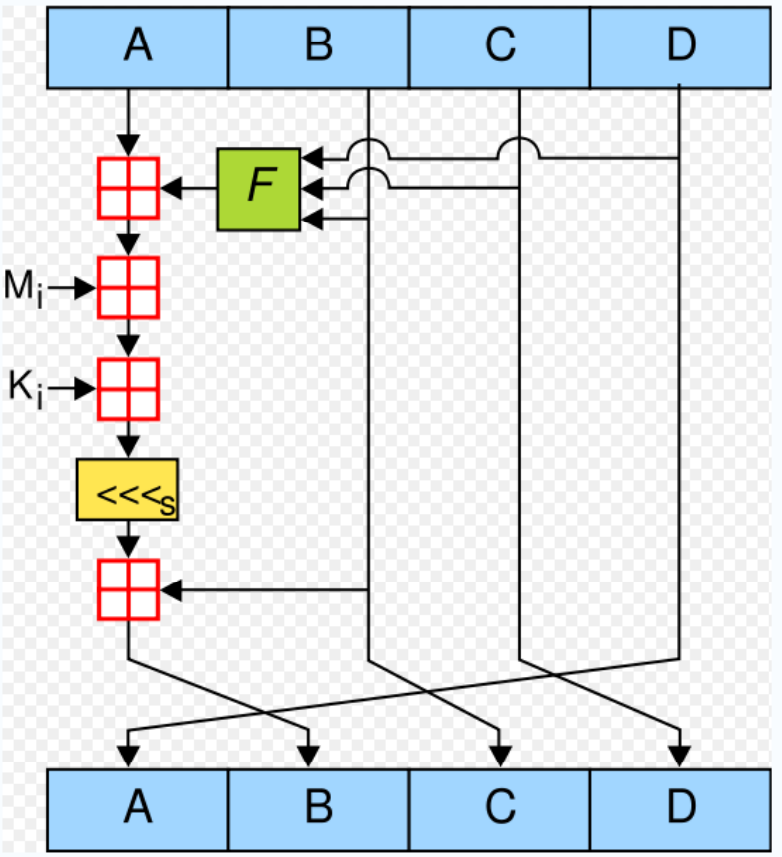
Hash Function

ซึ่งเป็นการป้องกันการเปิดเผยรหัสผ่านและไม่สามารถใช้ค่า Hash เพื่อคำนวณย้อนกลับไปเป็นรหัสผ่านได้ ในการตรวจสอบสิทธิ์ผู้ใช้แต่ละครั้งสำหรับการล็อกอินก็สามารถทำได้โดยนำรหัสผ่านที่ผู้ใช้ส่งผ่านฟอร์มล็อกอินเข้ามา แล้วนำไปผ่านฟังก์ชั่น Hash เช่น MD5 จากนั้นก็นำค่า Hash ที่ได้มาเทียบกับค่า Hash ที่เก็บไว้ใน Database หากมีค่าตรงกันก็แสดงว่ารหัสผ่านถูกต้อง ไฟล์รหัสผ่านของ Linux (/etc/shadow) ก็ Hash รหัสผ่านด้วย MD5 เช่นกัน นอกจากนั้นก็ยังพบเห็นการประยุกต์ใช้การแฮชรหัสผ่านใน Web Application ต่าง ๆ เช่น Moodle และ Mambo



Hash Function

อัลกอริทึม MD5 คิดค้นโดย Ron Rivest ซึ่งเป็น 1 ใน 3 คนที่คิดค้น RSA



MD5

- ถึงแม้ MD5 จะได้รับความนิยมอย่างมาก และได้มีการนำมาใช้แพร่หลายเช่น นำมาใช้สร้าง Digital Signature ในระบบ e-commerce อย่างไรก็ตาม MD5 ก็ถูกเบรคได้โดยนักคณิตศาสตร์หญิงชาวจีน (Professor Dr. Xiaoyun Wang) ในปี 2004 โดยใช้เครื่องซูเปอร์คอมพิวเตอร์ IBM P690 และใช้เวลาแคร็ก 1 ชั่วโมงก็สามารถเบรคได้ หลังจากนั้นก็มีคนอ้างว่าสามารถใช้เครื่องคอมพิวเตอร์ Notebook ความเร็ว 1.6 GHz เบรก MD5 ได้ภายในเวลา 8 ชั่วโมง
- ตัวอย่างการใช้งานคำสั่ง md5 บน Linux

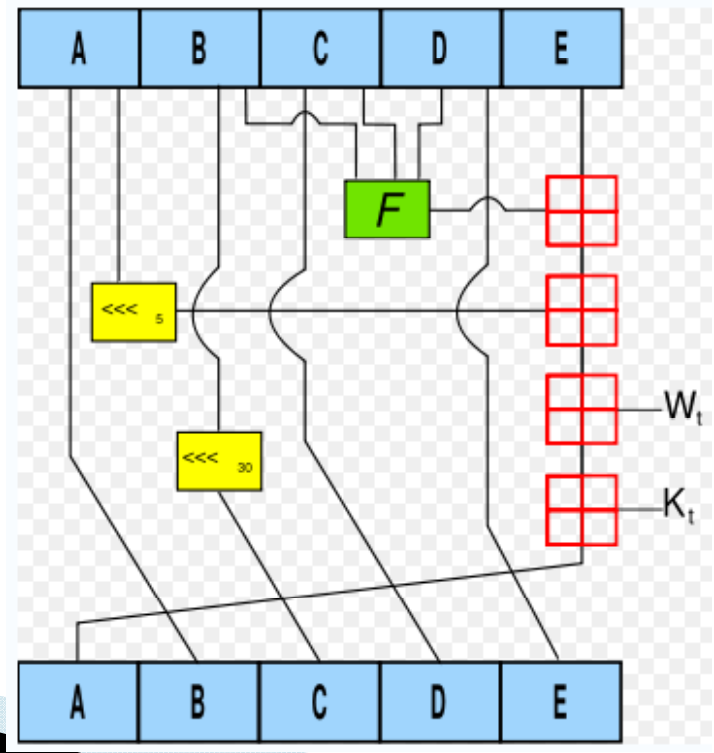
$ **md5 exim-4.43.tar.bz2**

MD5 (exim-4.43.tar.bz2) = f8f646d4920660cb5579becd9265a3bf

$

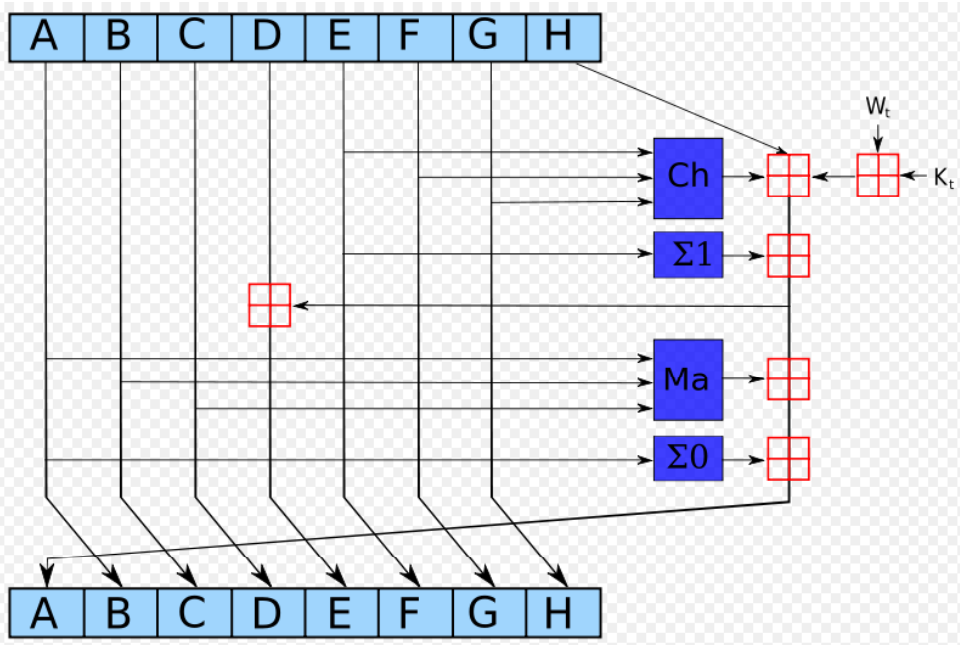
SHA

SHA0 และ SHA1 ได้ถูกพัฒนาให้มีความแข็งแรงกว่า MD5 โดยได้พัฒนาจาก MD5 เดิมให้ Output มีความเป็น Random สูงกว่า และมี Collision น้อยกว่าเพื่อลดโอกาสในการถูกแคร็กได้ อัลกอริทึมของ SHA1



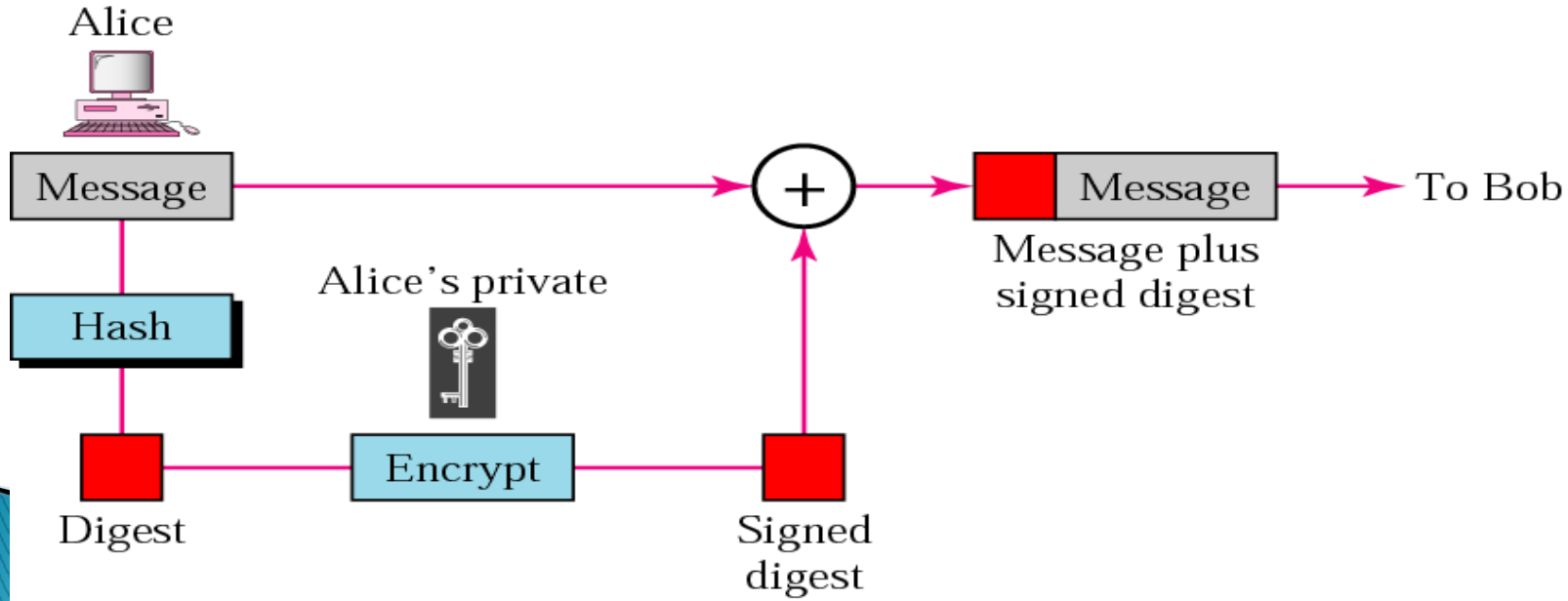
SHA

SHA0 และ SHA1 ก็ถูกเบรคได้โดยนักคณิตศาสตร์หญิงชาวจีน (Professor Dr. Xiaoyun Wang) คนเดียวกันกับที่เคยเบรค MD5 ได้ ดังนั้นปัจจุบันนี้ความหวังจึงอยู่ที่ SHA2 ซึ่งยังไม่มีใครเบรคได้ อัลกอริทึมของ SHA2



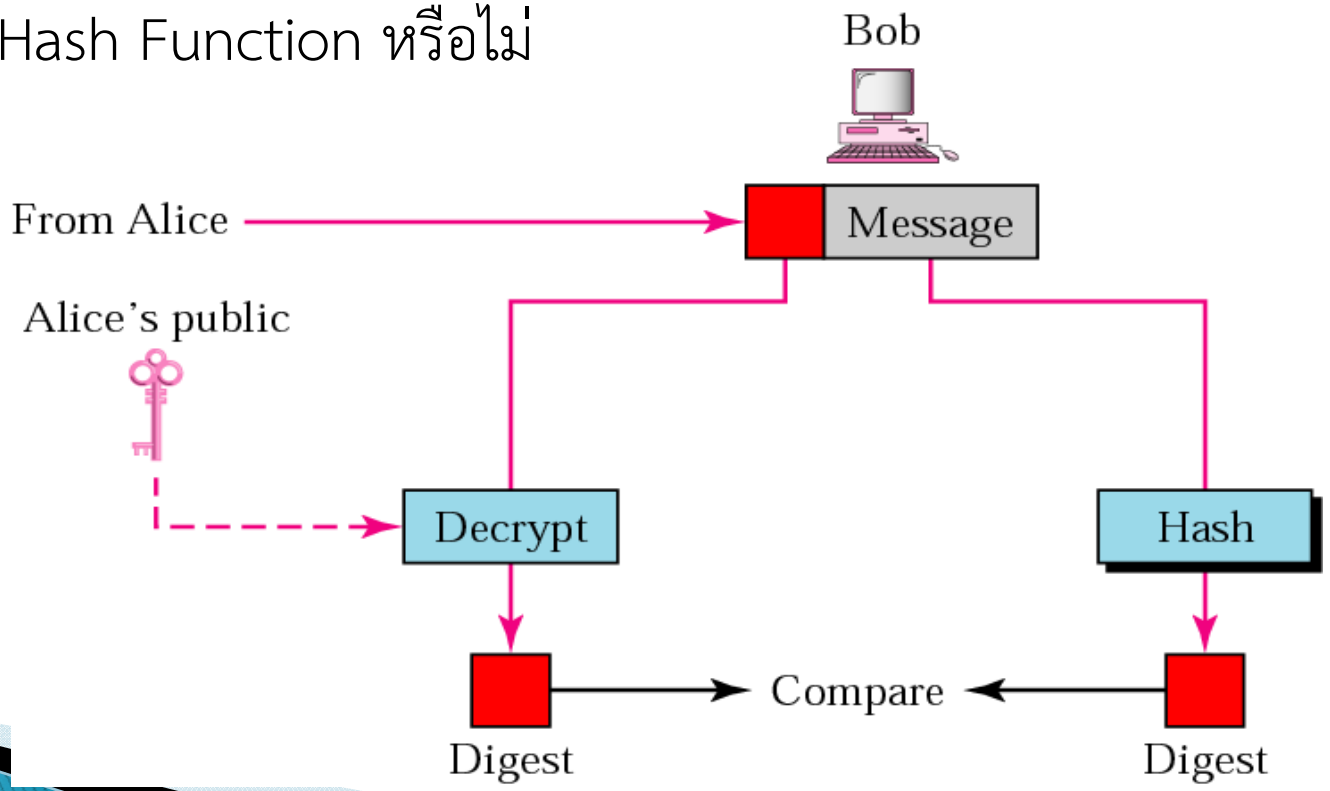
Digital Signature

- หลักการคือผู้ส่งจะนำข้อความ Plain Text มาเข้า Hash Function เพื่อให้ได้ Digest แล้วนำ Digest มาเข้ารหัสด้วยวิธีการของ Asymmetric Key จะได้ข้อมูลเป็น Signed Digest ซึ่งจะนำมารวมกับ Message และส่งไปให้กับผู้รับ



Digital Signature

- ผู้รับทำการแยก Signed Digest และ Plan Text ออกจากกันและนำ Plan Text ที่ได้รับเพื่อนำมา Hash Function เพื่อหา Digest และนำ Signed Digest มาถอดรหัสเพื่อตรวจสอบว่า Digest ที่ได้ตรงกันกับ Digest ของ Hash Function หรือไม่

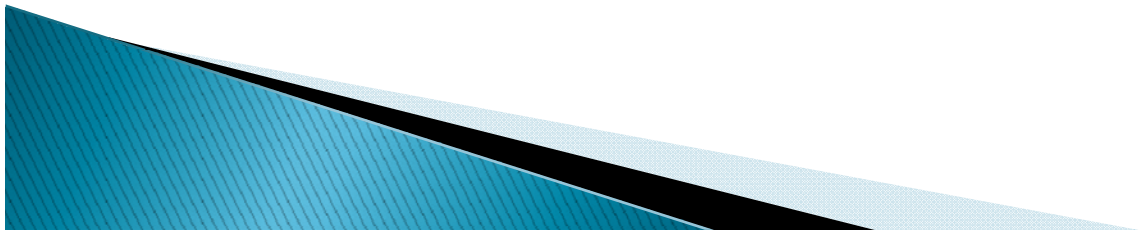


หลักการ Hasn Digital Signature

- นำข้อความที่ต้องการลงลายมือชื่อ (M) เป็นข้อมูลรับเข้าของ Hash Function (H) จะได้ผลลัพธ์เป็น Hash Code ที่มีความปลอดภัยคือ H(M) ที่มีความหมายคงที่
- นำ Hash Code มาเข้ารหัสสลับด้วยกุญแจส่วนตัวของผู้ส่ง
- ผู้ส่งส่งข้อความและลายมือชื่อพร้อมกันไปให้ผู้รับ
- ผู้รับนำข้อความที่ได้มาคำนวณหา Hash Code
- ผู้รับถอดรหัสสลับลายมือชื่อที่ได้รับด้วยกุญแจสาธารณะของผู้ส่ง
- ถ้า Hash Code มีค่าเท่ากับลายมือชื่อถอดรหัสสลับแล้วจะยอมรับลายมือชื่อนั้นว่าเป็นของจริง เนื่องจากผู้ส่งรู้กุญแจส่วนตัวแต่เพียงผู้เดียวและผู้ส่งเท่านั้นที่สามารถสร้างลายมือชื่อที่ถูกต้องได้ Valid Signature

ประโยชน์ของลายเซ็นดิจิตอล (Digital Signatures)

- ใช้เพื่อเพิ่มความปลอดภัย ช่วยยืนยันตัวจดหมายว่าส่งมาจากผู้ส่งนั้นจริง
- ใช้หลักการในการเปลี่ยนข้อความทั้งหมดให้เหลือเพียงข้อความสั้น ๆ เรียกว่า“**Message digest**” ซึ่งจะถูกสร้างขึ้นด้วยกระบวนการเข้ารหัสยอดนิยมที่เรียกว่า One-way hash function
- จะใช้ message digest นี้ในการเข้ารหัสเพื่อเป็นลายเซ็นดิจิตอล(Digital Signature) โดยจะแจก Public key ไปยังผู้ที่ต้องการติดต่อ



การตรวจสอบลายเซ็นดิจิตอล

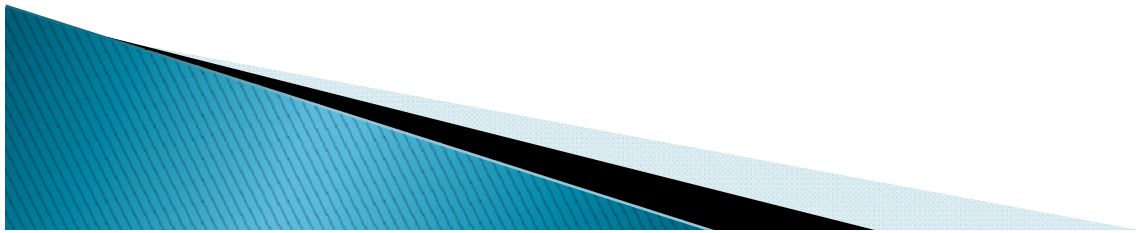
ตัวอย่าง

- ทิมต้องการส่งข้อความไปให้แอน ทิมก็นำข้อความที่ต้องการส่งมาคำนวณหา message digest ของข้อความ

- ทิมนำ message digest ที่ได้มาเข้ารหัสด้วยคีย์ส่วนตัวของทิม จะได้ผลลัพธ์ออกมาเป็นลายเซ็นดิจิตอล

- ทิมส่งข้อความต้นฉบับที่ไม่ได้เข้ารหัส พร้อมกับลายเซ็นดิจิตอลของตนเองไปให้แอน

- แอนได้รับข้อความก็จะนำคีย์สาธารณะของทิมมาถอดรหัสลายเซ็นดิจิตอลของทิม ได้ออกมาเป็น message digest ที่ทิมคำนวณไว้

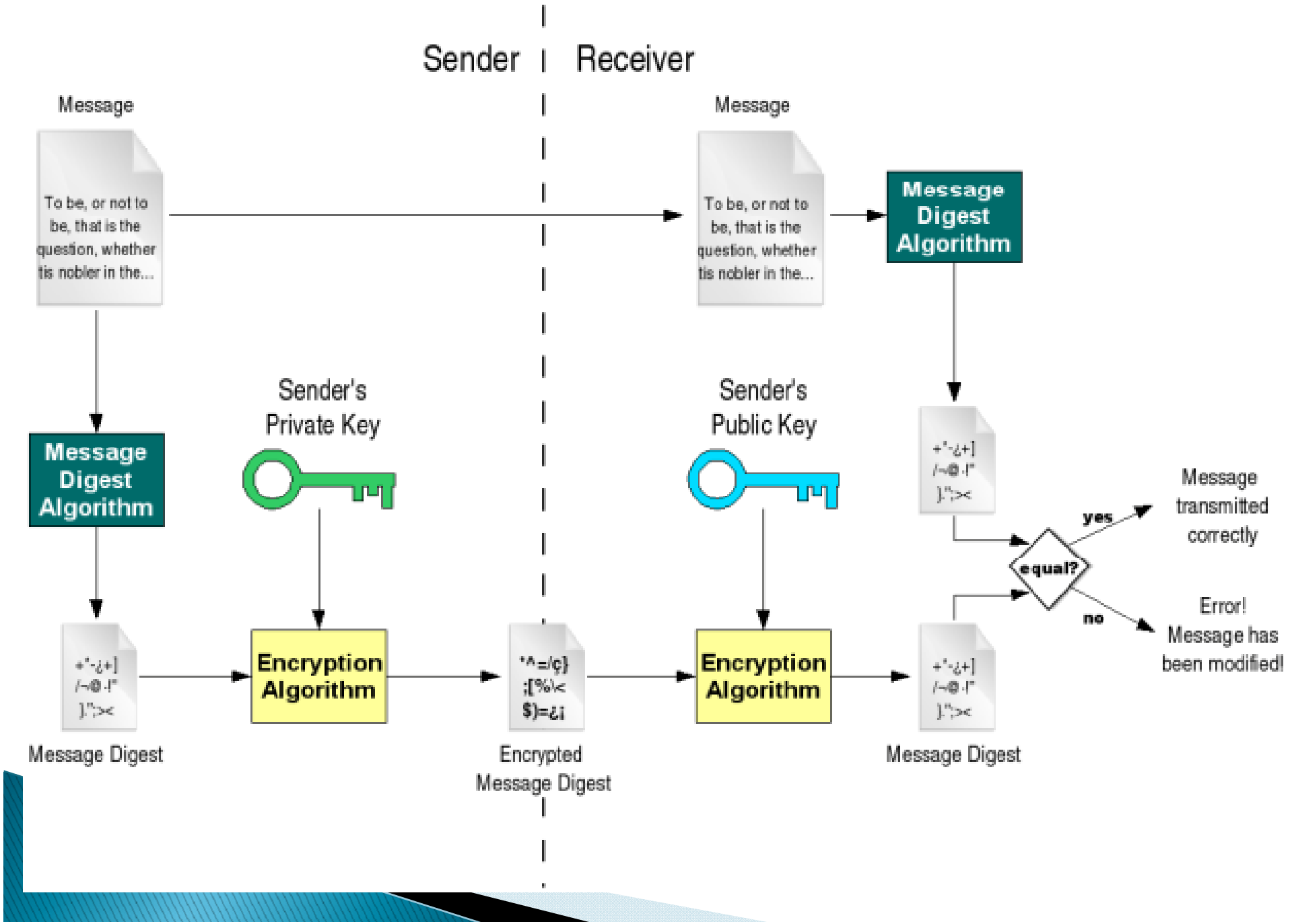


การตรวจสอบลายเซ็นดิจิตอล

- แอนมั่นใจได้ว่าข้อความที่ได้รับนั้นส่งมาโดยทิมจริง ๆ เพราะถอดรหัสลายเซ็นของทิมได้

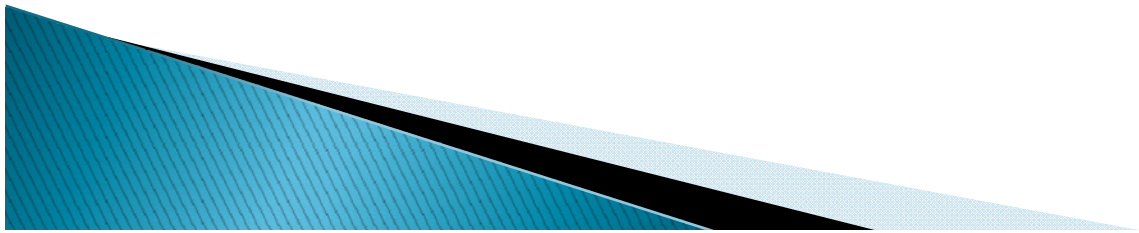
- แอนใช้แฮชฟังก์ชันตัวเดียวกับที่ทิมใช้ (ต้องตกลงกันไว้ก่อน) มาคำนวณหา message digest จากข้อความที่ทิมส่งมาเพื่อเปรียบเทียบกัน

- ถ้า message digest ที่ได้จากการคำนวณทั้งสองตรงกัน แสดงว่าลายเซ็นดิจิตอลเป็นของทิมจริง และไม่มีผู้ใดมาเปลี่ยนแปลงแก้ไขแต่อย่างใดก่อนจะมาถึงแอน


Sender
Receiver
Message
To be, or not to be, that is the question, whether tis nobler in the...
Message
To be, or not to be, that is the question, whether tis nobler in the...
Message Digest Algorithm
Message Digest Algorithm
Sender's Private Key
Sender's Public Key
+*-¿+]
/~@.!"
}";><
Encryption Algorithm
*^=/ç}
;[%\<
$}=¿i
Encryption Algorithm
+*-¿+]
/~@.!"
}";><
+*-¿+]
/~@.!"
}";><
equal?
yes
no
Message transmitted correctly
Error! Message has been modified!
Message Digest
Encrypted Message Digest
Message Digest

สรุปเกี่ยวกับขั้นตอนของ Digital Signatures ได้ดังนี้

1. นำข้อมูลต้นฉบับ (ซึ่งอาจจะมีขนาดใหญ่) มาทำการ Hash (ด้วยแฮชชิ่งอัลกอริทึมอย่างใดอย่างหนึ่งเช่น MD5 หรือ SHA1) ได้เป็นข้อมูลก้อนเล็ก ๆ เรียกว่า Message Digest
2 .นำ Message Digest มาเข้ารหัสด้วย Private Key ของผู้ส่ง ได้เป็น "Digital Signatures"
3 .ส่งข้อมูลต้นฉบับ (อาจจะมีขนาดใหญ่) ซึ่งอยู่ในรูปของ Plain Text ไปให้ผู้รับ โดยทำการแนบ Digital Signatures ไปด้วย (มีการส่งข้อมูลไปยังผู้รับ 2 ชิ้นคือ (a) ข้อมูลต้นฉบับ และ (b) Digital Signatures)



สรุปเกี่ยวกับขั้นตอนของ Digital Signatures ได้ดังนี้

4 .ผู้รับเมื่อได้รับข้อมูลแล้วก็ทำการตรวจสอบข้อมูลที่ได้รับ โดยการนำ Digital Signatures มาถอดรหัสโดยใช้ Public Key ของผู้ส่ง ได้เป็น Message Digest

5 .ผู้รับนำข้อมูลต้นฉบับมาทำการ Hash (ด้วยอัลกอริทึมเดียวกันกับที่ผู้ส่งใช้เช่น MD5 หรือ SHA1) ได้เป็น Message Digest อีกอันหนึ่ง

6 .นำ Message Digest ทั้งสองมาเปรียบเทียบกัน หากมีค่าเหมือนกันก็แสดงว่าข้อมูลต้นฉบับถูกส่งมาจากผู้ส่งจริงและไม่มีการเปลี่ยนแปลงข้อมูลระหว่างทาง

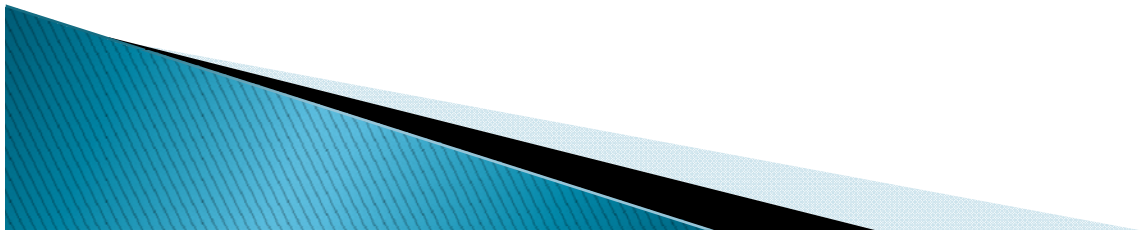
Digital Signature

ข้อสังเกตุ

- สิ่งที่ทำให้มั่นใจได้ว่าข้อมูลถูกส่งมาจากผู้ส่งจริงคือการที่ผู้รับสามารถถอด Digital Signatures โดยใช้ Public Key ของผู้ส่งได้ แสดงว่าข้อมูลนั้นถูกเข้ารหัสโดยใช้ Private Key ของผู้ส่งจริง ซึ่งผู้ที่มี Key นี้มีอยู่คนเดียวเท่านั้นคือผู้ส่ง
- สิ่งที่ทำให้มั่นใจได้ว่าข้อมูลไม่ถูกเปลี่ยนแปลงระหว่างทางคือ การเปรียบเทียบค่า Hash ทั้งสองแล้วพบว่าตรงกัน โดยค่าแฮชตัวหนึ่งมาจากการนำข้อมูล Plain Text ที่ได้รับมาแฮช ส่วนค่าแฮชอีกตัวหนึ่งมาจากการถอดรหัส Digital Signatures ที่สร้างจากข้อมูลต้นฉบับ ดังนั้นหากทั้งสองนี้ตรงกันก็แสดงว่ามาจากข้อมูลเดียวกัน

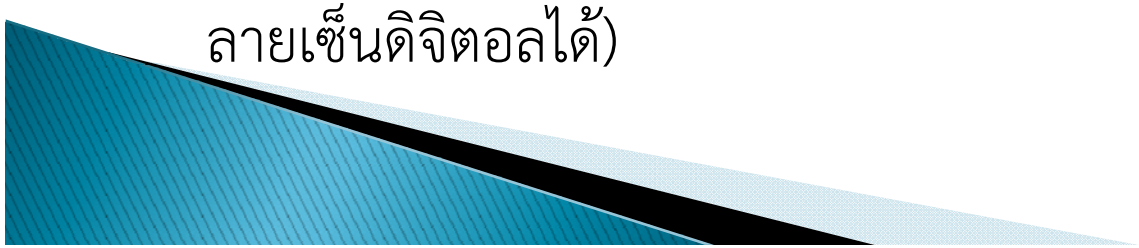
Digital Signature

ซึ่งสามารถที่จะส่ง Data, Digital Signature, และ Public Key ไปพร้อมกันได้ โดยส่ง Public Key ในรูปแบบของ Certificate (Public Key ที่ถูกรับรองโดย CA แล้ว) นอกจากนั้นการส่งข้อมูลลับและลงลายเซ็นดิจิตอลก็สามารถทำได้เช่นกัน โดยการนำขั้นตอนของ Digital Signatures ทั้ง 6 ขั้นตอนมา Apply โดยแก้ไขขั้นตอนที่ 3 และ 5 ดังนี้



ขั้นตอนที่ 3 (ปรับใหม่) นำข้อมูลต้นฉบับ (อาจจะมีขนาดใหญ่) ซึ่งเป็น Plain Text มาเข้ารหัสด้วย Public Key ของผู้รับ ให้กลายเป็น Cipher Text จากนั้นจึงส่ง Cipher Text ไปให้ผู้รับ โดยทำการแนบ Digital Signatures ไปด้วย (มีการส่งข้อมูลไปยังผู้รับ 2 ชิ้นคือ (a) ข้อมูลที่เป็น Cipher Text และ (b) Digital Signatures)

ขั้นตอน 5 (ปรับใหม่) ผู้รับนำข้อมูลที่เป็น Cipher Text มาถอดรหัสด้วย Pivate Key ของตนเอง ได้เป็นข้อมูลต้นฉบับแบบ Plain Text จากนั้นมาทำการ Hash (ด้วยอัลกอริทึมเดียวกันกับที่ผู้ส่งใช้เช่น MD5 หรือ SHA1) ได้เป็น Message Digest อีกอันหนึ่ง (เมื่อนำขั้นตอนที่ 3 และ 5 ที่ปรับใหม่ไปใช้ร่วมกับข้อที่ 1,2,4,6 เดิมก็จะสามารถส่งข้อมูลลับพร้อมลงลายเซ็นดิจิตอลได้)

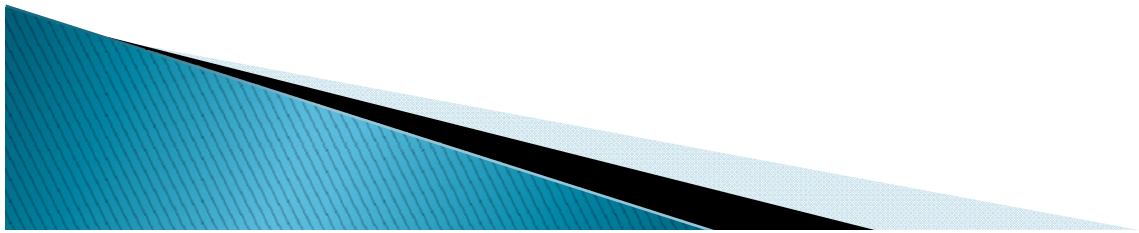


CA (Certificate Authority)

การใช้วิธีการเข้ารหัสแบบอสมมาตร (Asymmetric Key Cryptography) ที่จำเป็นจะต้องมีการแจกจ่าย Public Key ไปยังผู้ร่วมสื่อสารทุก ๆ รายนั้นมีจุดอ่อนจากการโจมตีแบบ MITM: Man In The Middle โดยแฮกเกอร์จะทำตัวเป็นผู้อยู่ตรงกลางของการสื่อสารแล้วรอจังหวะที่ผู้สื่อสารข้อมูลมีการรับส่งและแลกเปลี่ยน Key กัน โดยแฮกเกอร์จะทำการส่ง Public Key ของแฮกเกอร์เองไปยังผู้รับ ดังนั้นแฮกเกอร์จึงสามารถที่จะปลอมแปลงข้อมูลได้โดยใช้ Private Key ของตนเองในการทำ Digital Signatures ทำให้ผู้รับหลงเชื่อคิดว่าข้อมูลนั้นถูกส่งมายังผู้ส่งจริง หรือบางครั้งแฮกเกอร์ก็ปลอมแปลงทั้ง Private Key และ Public Key

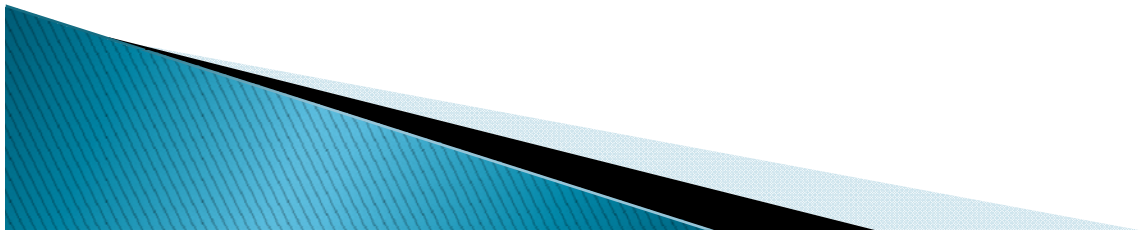
CA (Certificate Authority)

ตัวอย่างของการโจมตี Asymmetric Key Cryptography โดยใช้วิธี MITM ที่เห็นได้บ่อยคือการถอดรหัสข้อมูลที่ส่งทาง https (เช่นรหัสผ่านของ hotmail, Gmai และเว็บไซต์ e-commerce อื่น ๆ) ด้วยโปรแกรมในชุดของ BackTrack หรือใช้โปรแกรม Cain โดยแฮกเกอร์จะดักรอขั้นตอนการส่ง Pulic Key ผ่านทาง https เมื่อ Public Key ของเว็บไซต์ (เช่น hotmail) ถูกส่งมาถึงแฮกเกอร์ เขาจะทำการสร้าง Private Key และ Public Key ของตนเองขึ้นมา แล้วส่ง Public Key ไปให้เหยื่อ



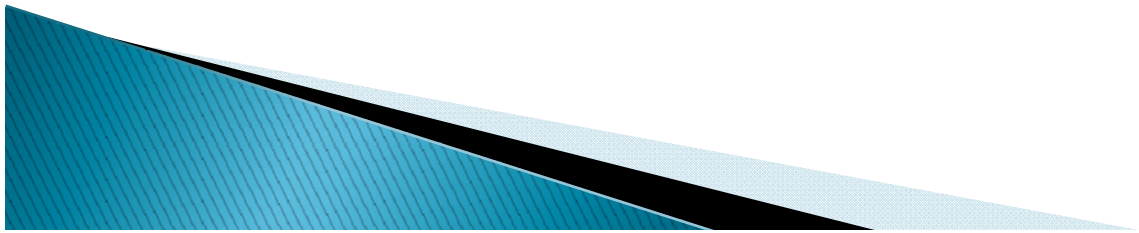
CA (Certificate Authority)

เมื่อเหยื่อต้องการส่งข้อมูลมายังเว็บไซต์ก็จะเข้ารหัสด้วย Public Key ของเว็บไซต์เพื่อให้เว็บไซต์ถอดได้แต่เพียงผู้เดียว แต่ Public Key นั้นที่แท้จริงเป็น Public Key ของแฮกเกอร์ จึงทำให้แฮกเกอร์ถอดรหัสได้ และได้ข้อมูลที่ แฮกเกอร์ที่มี Public Key ของเว็บไซต์อยู่แล้วก็จะเอาข้อมูลนั้นมาดำเนินการเข้ารหัสด้วย Public Key จริงของเว็บไซต์ และส่งให้เว็บเซิร์ฟเวอร์ต่อเพื่อให้การสื่อสารครบวงจรเพื่อที่เหยื่อจะได้ไม่รู้สึกถึงความผิดปกติ



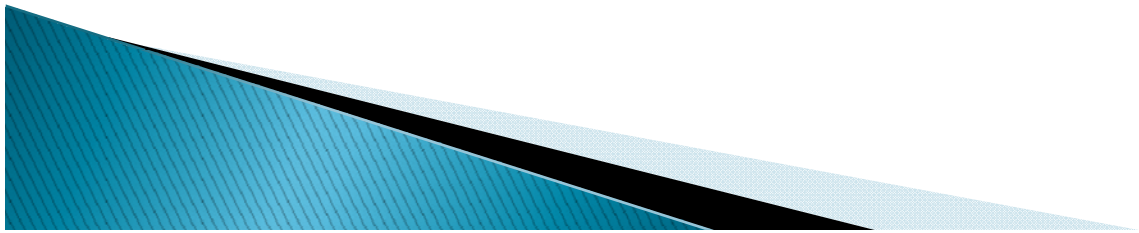
CA (Certificate Authority)

เพื่อแก้ไขปัญหาที่ผู้รับไม่สามารถที่จะตรวจสอบได้ว่า Public Key ที่ตนเองได้รับนั้นเป็นของผู้ส่งจริงหรือไม่ จึงได้มีการพัฒนาโครงสร้างพื้นฐานของ Asymmetric Key Cryptography ให้ปลอดภัยขึ้น เรียกว่า Public Key Infrastructure (PKI) โดยกำหนดให้มีหน่วยงานกลางเป็นผู้รับรอง Public Key ของแต่ละคน / เว็บไซต์ / เซิร์ฟเวอร์ หน่วยงานกลางดังกล่าวมีชื่อเรียกว่า CA (Certificate Authority)



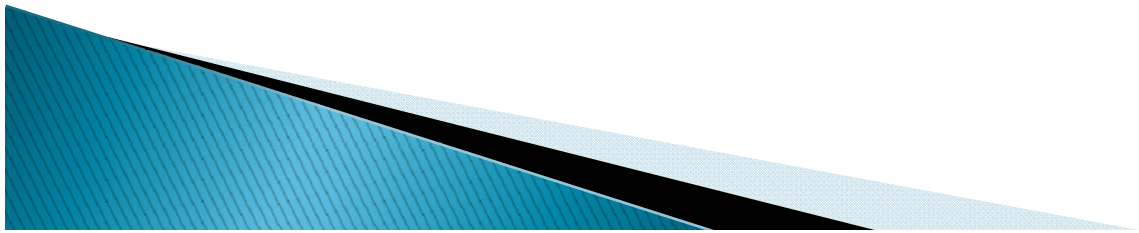
CA (Certificate Authority)

โดย CA จะได้รับการรับรองโดย Root CA อีกครั้งหนึ่ง ผู้รับส่งข้อมูลทุกรายจะต้องมี Public Key ของ Root CA ติดตั้งไว้บนระบบ เช่นบน Windows XP ก็จะมี Public Key ของ Root CA ทุกราย ซึ่งเราสามารถเปิดดู Public Key ของ Root CA ได้โดยใช้ Internet Explorer (เมนู Tools->Internet Options->Content->Certificates->Trusted Root Certification Authority)

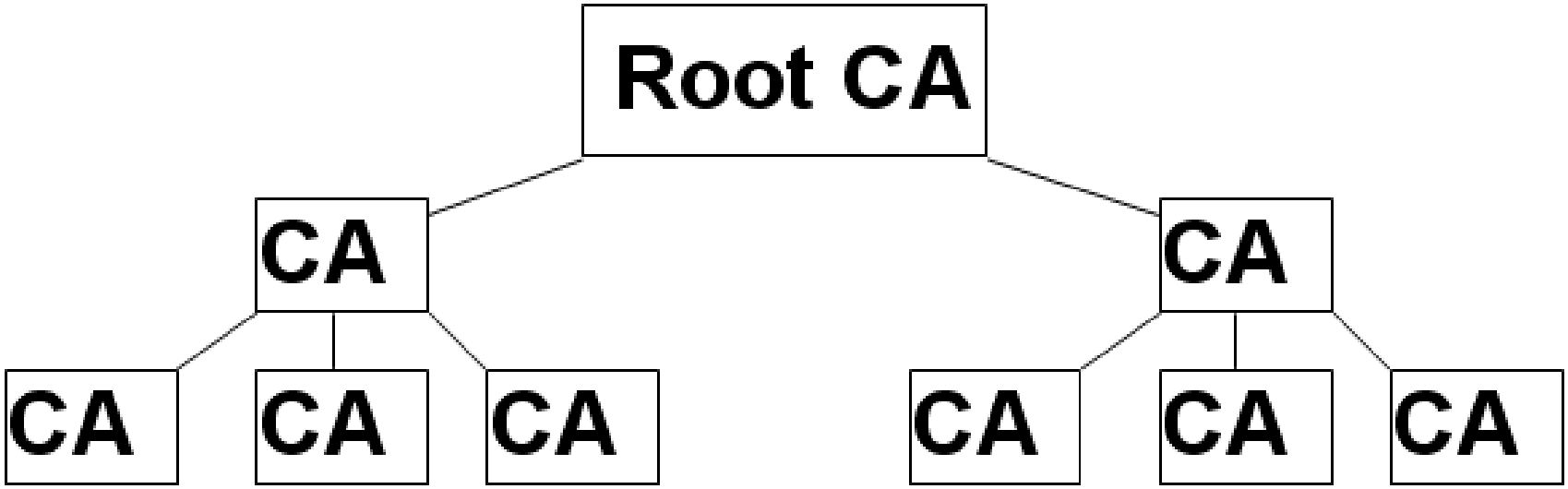


CA (Certificate Authority)

CA จะนำ Public Key ของผู้ส่ง (หรือผู้ใช้ / เว็บไซต์ ที่ขอให้ CA รับรอง) มารับรองด้วย Digital Signatures ของ CA (ทำการนำข้อมูลที่สำคัญเช่น Public Key และชื่อเว็บไซต์ ของผู้ส่งมาแฮช แล้วเข้ารหัสด้วย Private Key ของ CA กลายเป็น Digital Signatures) แล้วนำข้อมูลต้นฉบับ (Public Key และชื่อเว็บไซต์) มาผนวกเข้ากับ Digital Signatures ดังกล่าว กลายเป็นสิ่งที่เรียกว่า Certificate

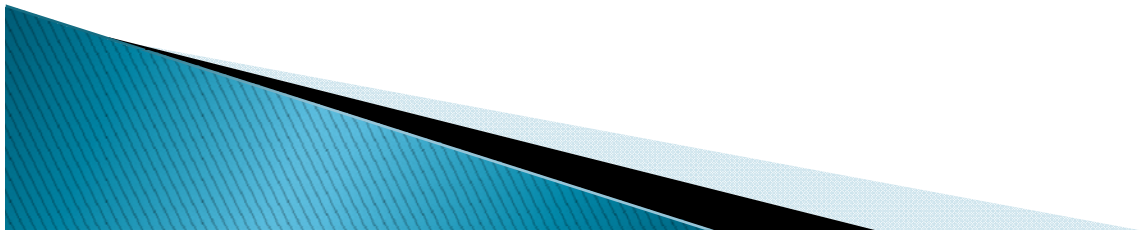


การรับรองเป็นลำดับชั้นของ CA (Certificate Authority)



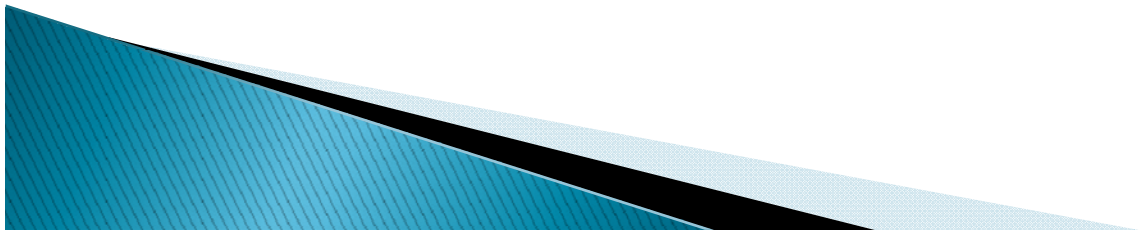
CA (Certificate Authority)

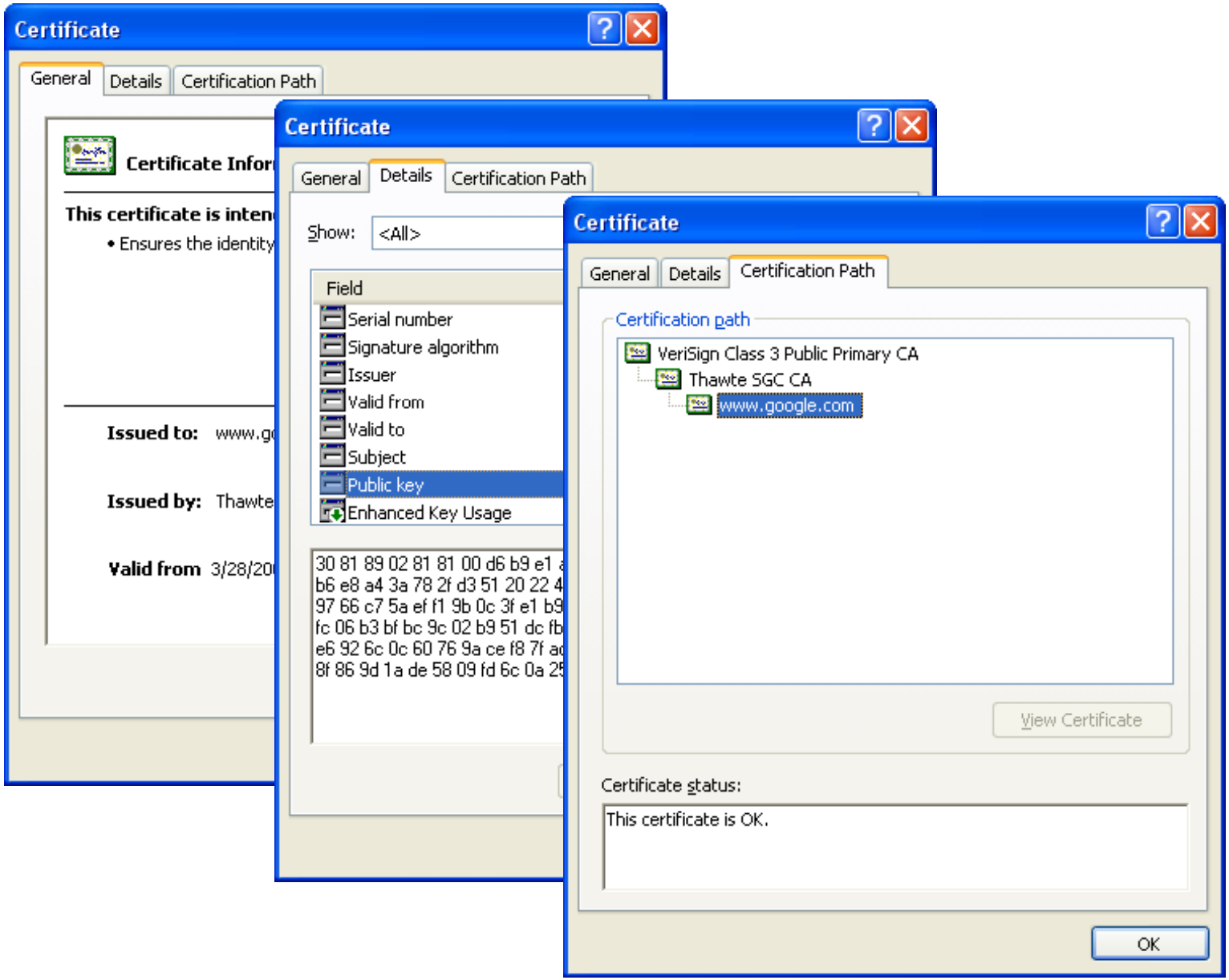
ตัวอย่างเช่น Public Key ของเว็บไซต์ www.google.com ได้รับการรับรองโดย CA ชื่อ Thawte SGC ซึ่ง Thawte SGC ทำการรับรองโดยใช้ Certificate (มี Digital Signatures ที่รับรองโดย CA และมีข้อมูล Public Key และชื่อของเว็บไซต์ www.google.com อยู่ภายใน) ส่วน Thawte SGC ก็ถูกรับรองด้วย Certificate ที่ออกโดย Root CA ชื่อ VeriSign อีกทอดหนึ่งโดยใช้กลไกแบบเดียวกันกับที่ Thawte SGC รับรอง www.google.com



CA (Certificate Authority)

บราวเซอร์สามารถใช้ Public Key ของ VeriSign (Root CA) ที่มีอยู่บน Windows ถอดรหัสทำให้แน่ใจได้ว่า Public Key ของ Thawte SGC เป็นของจริง และมั่นใจได้ว่า Public Key ของ www.google.com เป็นของจริงได้โดยใช้วิธีการในทำนองเดียวกัน



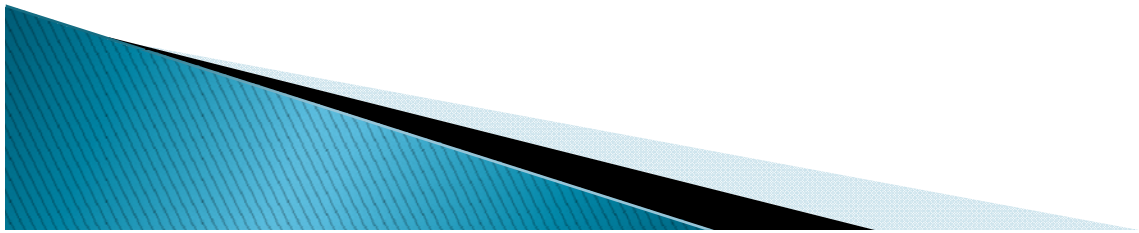


Certificate ของ www.google.com

โพรโทคอลในการพิสูจน์ตัวตน

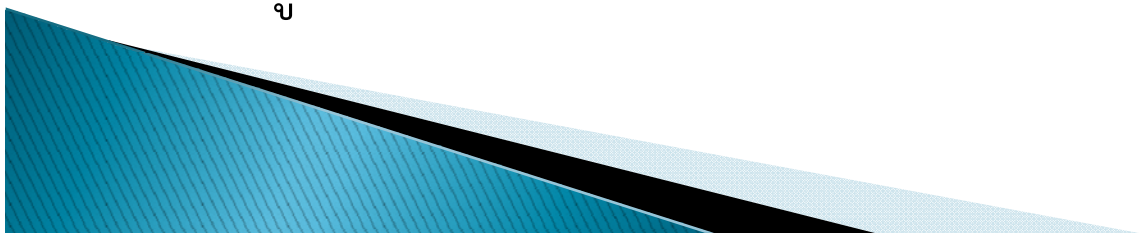
ในระบบเครือข่ายแบบเปิดหรืออินเทอร์เน็ต การพิสูจน์ตัวตนถือได้ว่าเป็นกระบวนการเริ่มต้นและมีความสำคัญที่สุดในการปกป้องเครือข่ายให้ปลอดภัย โพรโทคอลในการพิสูจน์ตัวตน คือโพรโทคอลการสื่อสารที่มีกระบวนการพิสูจน์ตัวตนรวมอยู่ในชุดโพรโทคอล ได้แก่

- Secure Socket Layer (SSL)
- Secure Shell (SSH)
- Internet Security (IPSEC)
- Kerberos



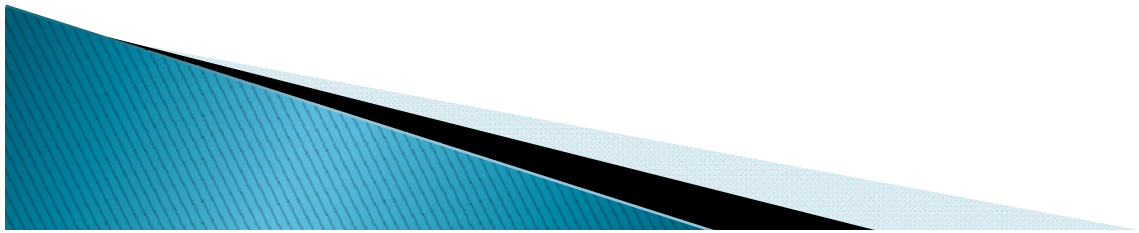
Secure Socket Layer (SSL)

Secure Sockets Layer (SSL) เริ่มพัฒนาโดย Netscape Communications เพื่อใช้ใน Protocol Application Layer คือ Hypertext Transfer Protocol (HTTP) ซึ่งเป็นการสื่อสารผ่านเว็บให้ปลอดภัย พัฒนาในช่วงต้นของยุคการค้าอิเล็กทรอนิคส์กำลังได้รับความนิยมในโลกอินเทอร์เน็ต SSL ทำให้เกิดการสื่อสารอย่างปลอดภัยระหว่าง Client and Server โดยการอนุญาตให้มีกระบวนการพิสูจน์ตัวตนร่วมกับการใช้งาน Digital Signature สำหรับการรักษาความถูกต้องของข้อมูลและการเข้ารหัสข้อมูลเพื่อป้องกันความเป็นส่วนตัวระหว่างการสื่อสารข้อมูล



Secure Socket Layer (SSL)

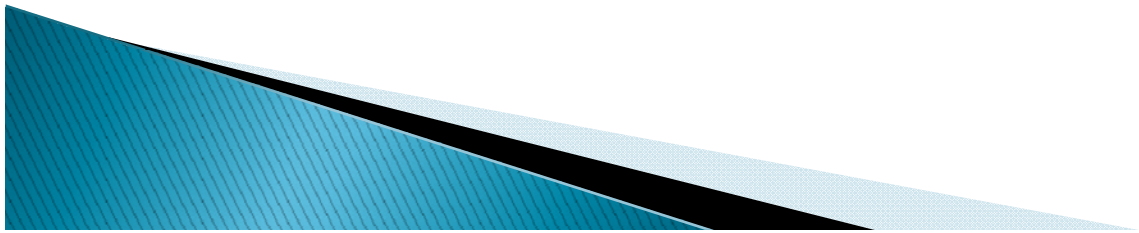
Protocol SSL อนุญาตให้สามารถเลือกวิธีการในการเข้ารหัส วิธีสร้าง Digest and Digital Signature ได้อย่างอิสระก่อนการสื่อสารจะเริ่มต้นขึ้น ตามความต้องการของทั้ง Web Server and Bowser ทั้งนี้เพื่อเพิ่มความยืดหยุ่นในการใช้งาน เปิดโอกาสให้ทดลองใช้วิธีการในการเข้ารหัสวิธีใหม่ รวมถึงลดปัญหาการส่งออกวิธีการเข้ารหัสไปประเทศที่ไม่อนุญาต



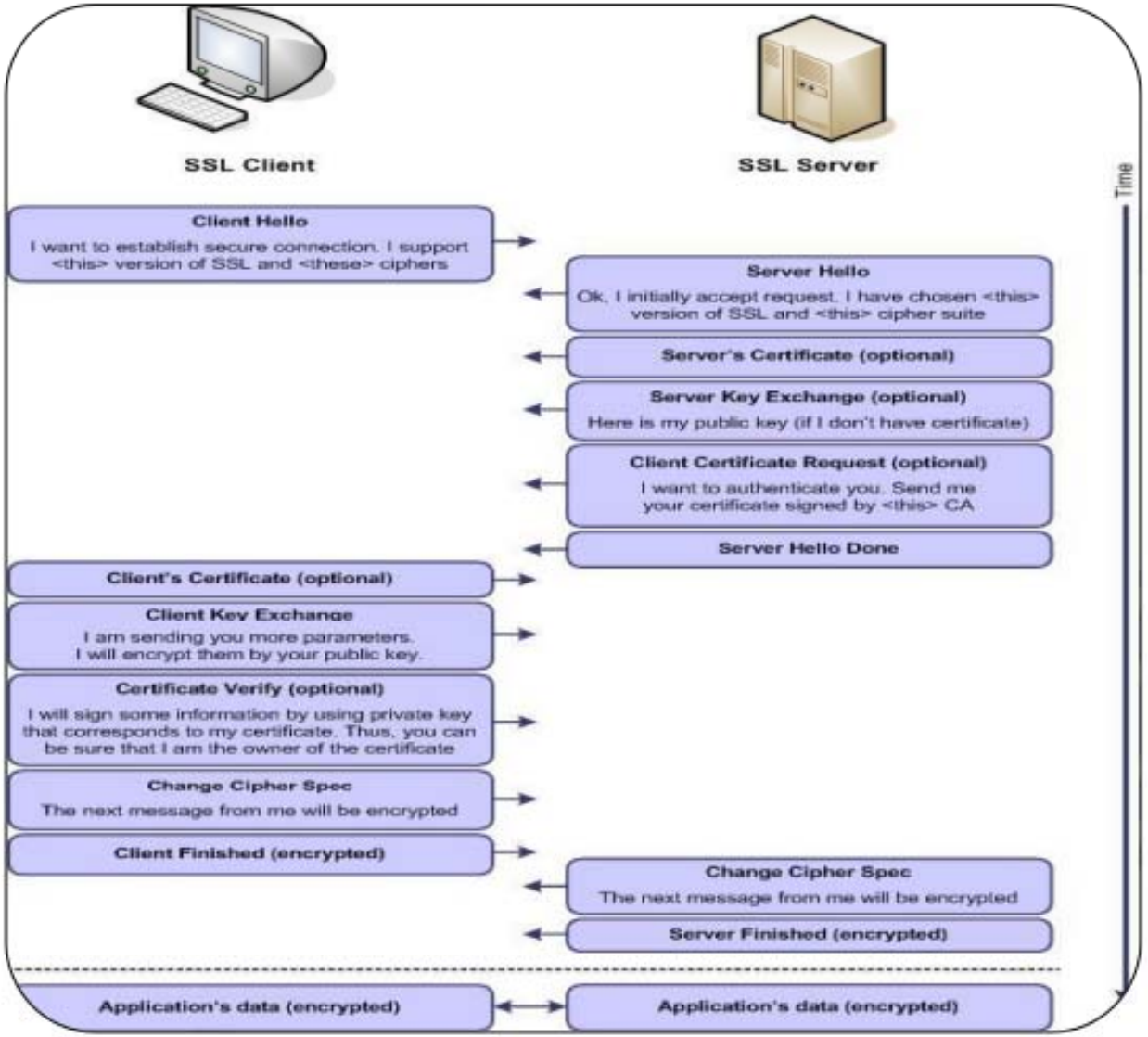
Secure Socket Layer (SSL)

กระบวนการในการเริ่มต้นการสื่อสารผ่านชั้น SSL แบ่งเป็น 4 ขั้นตอนคือ

- ประกาศชุดวิธีการเข้ารหัส ไดเจสต์ และลายเซ็นดิจิตอลที่สนับสนุนของทั้งไคลเอ็นต์และเซิร์ฟเวอร์
- การพิสูจน์ตัวตนของเซิร์ฟเวอร์ต่อไคลเอ็นต์
- การพิสูจน์ตัวตนของไคลเอ็นต์ต่อเซิร์ฟเวอร์ ถ้าจำเป็น
- ไคลเอ็นต์และเซิร์ฟเวอร์ตกลงชุดวิธีการเข้ารหัส การสร้างไดเจสต์ และการใช้ลายเซ็นดิจิตอล



Secure Socket Layer (SSL)



(http[illegible]ww.websecureplus.com/knowledges/secure-socket-layer-ssl-protocol/

Secure Shell (SSH)

SSH Secure Shell Client คือโปรแกรมสำหรับการเข้าถึงเครื่องคอมพิวเตอร์จากระยะไกล (Remote Login) แบบที่มีการคำนึงถึงความปลอดภัยผ่าน Protocol SSH โดยมีการเข้ารหัสลับ Username and Password ที่จะส่งออกไป รวมทั้งสามารถเข้ารหัสลับในรูปแบบ File Transfer ได้อีกด้วย พอร์ตมาตรฐานของ SSH คือพอร์ต 22

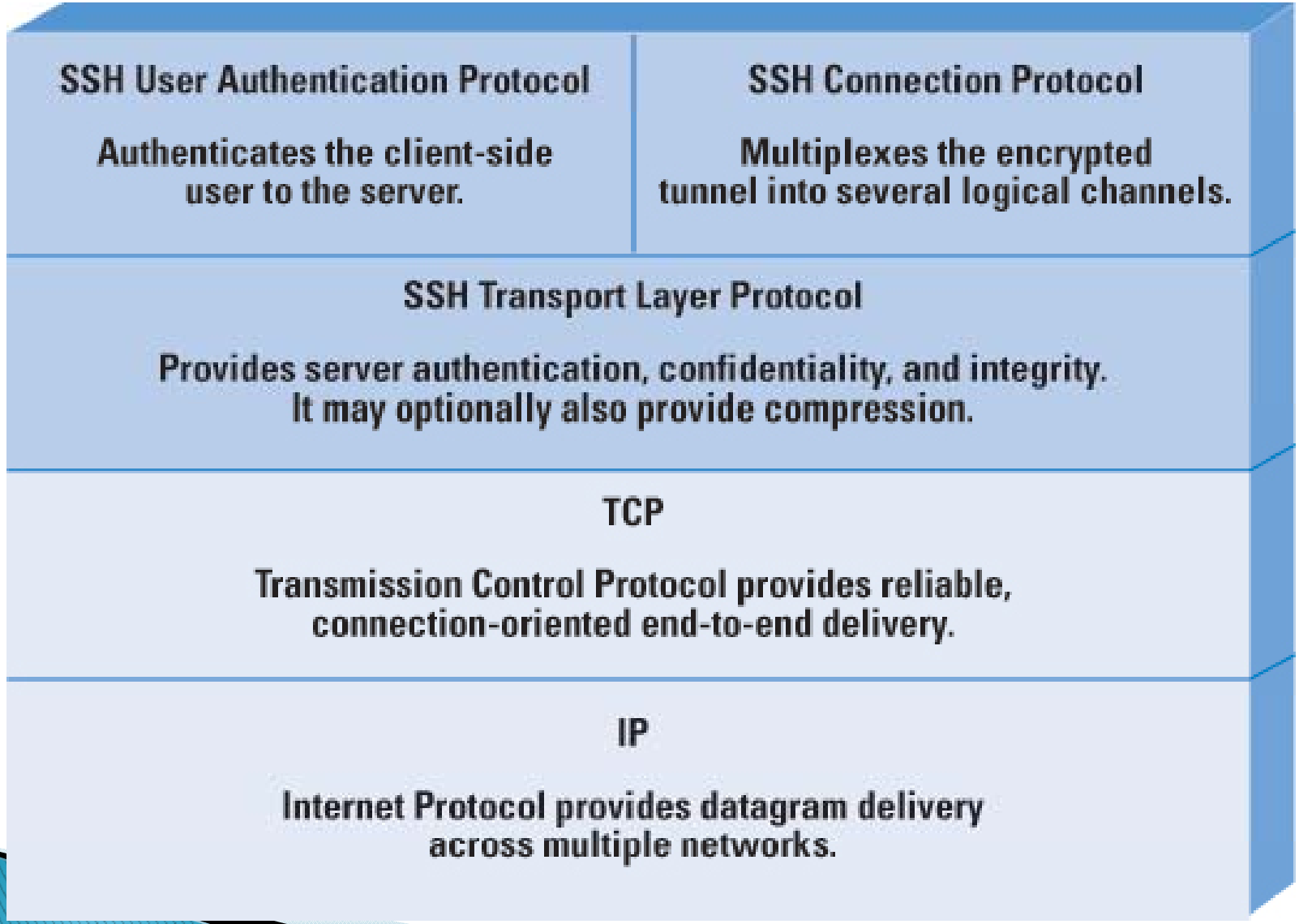
Program Application SSH ประเภท Client and Server ได้รับความนิยมจากหลายระบบปฏิบัติการ จนกลายมาเป็นทางเลือกสำหรับการเข้าสู่ระบบจากระยะไกลและใช้ช่องทางเฉพาะที่มีการเข้ารหัสลับ (X tunneling) และ รวดเร็ว จึงได้กลายเป็นหนึ่งในโปรแกรมประยุกต์ที่แพร่หลายมากที่สุดสำหรับเทคโนโลยีการเข้ารหัสลับที่ไม่ได้ใช้ระบบฝังตัว (Embedded Systems)

Secure Shell (SSH)

Protocol SSH จะทำงานอยู่บน Protocol TCP โดยจะถูกแบ่งออกเป็น 3 โปรโตคอลย่อยได้แก่

- **ชั้นทรานสปอร์ตโปรโตคอล** : เตรียมไว้สำหรับการตรวจสอบ Server, ปกป้องความลับของข้อมูล และ การคงสภาพของข้อมูล อาจจะเตรียมไว้สำหรับการบีบอัดข้อมูล ซึ่งเป็นการลดขนาดของข้อมูลเพื่อประหยัดพื้นที่หรือเวลาการในส่งข้อมูลด้วย
- **โปรโตคอลการพิสูจน์ตัวจริงของผู้ใช้งาน** : เป็นการพิสูจน์ตัวจริง สำหรับฝั่งผู้ใช้งานกับเซิร์ฟเวอร์
- **โปรโตคอลการเชื่อมต่อ** : มีไว้สำหรับร่วมเส้นทางการสื่อสารแบบ Logical หลายเส้นทางไปในเส้นทางพื้นฐานเส้นทางเดียว

Secure Shell (SSH)



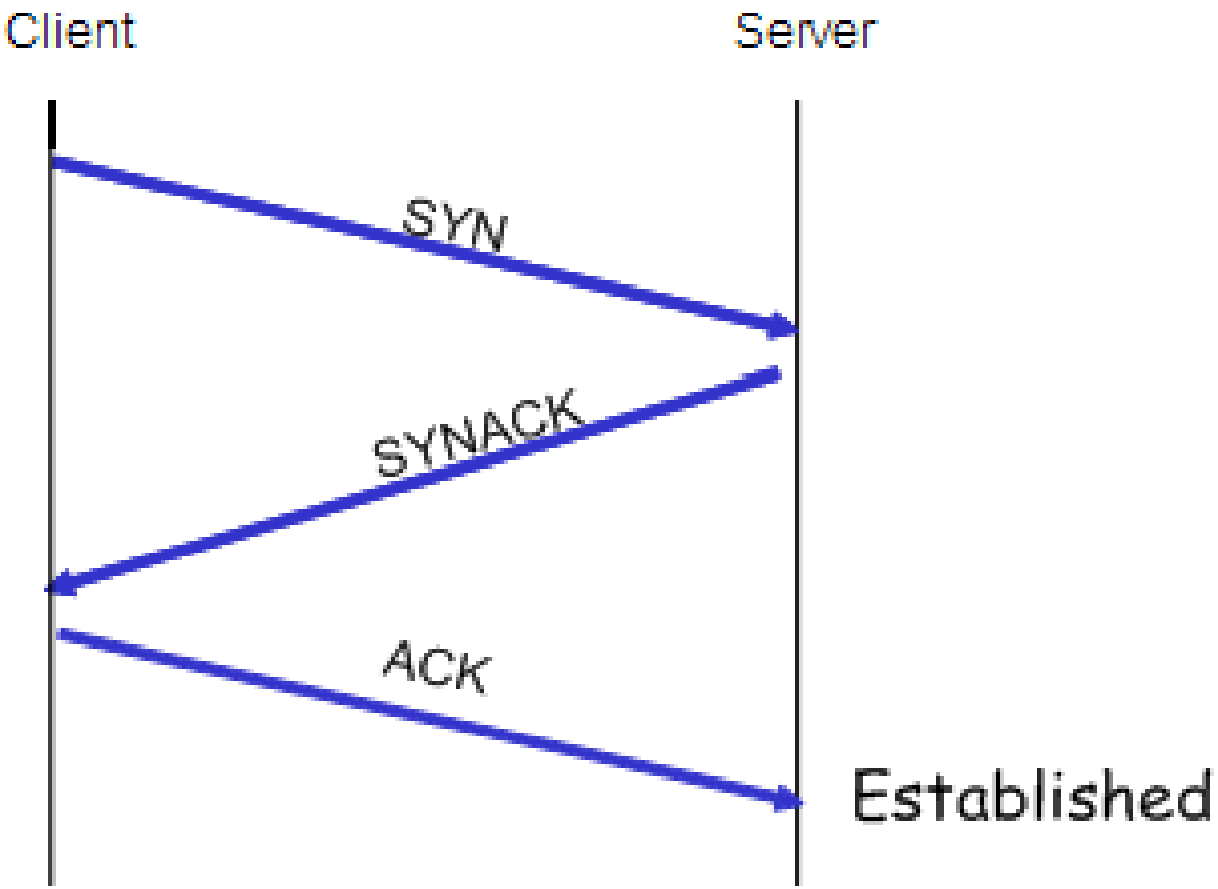
http://www.msit.mut.ac.th/[illegible]/phpfile/show.php?Qid=8006

Secure Shell (SSH)

การแลกเปลี่ยนข้อมูล บน SSH ชั้นทรานสปอร์ตโปรโตคอลมีขั้นตอนการทำงานดังนี้

- **Establish TCP Connection** สร้างการเชื่อมต่อผ่านทาง Protocol TCP เครื่อง Client ทำการสร้างการเชื่อมต่อไปยังเครื่อง Server หลังจากทำการเชื่อมต่อกันแล้ว เครื่อง Client ก็จะทำการแลกเปลี่ยนข้อมูลกับเครื่อง Server โดยการใช้ Three-Way Handshake
  - เครื่อง Client ส่ง Synchronization (SYN) ขอสร้างการเชื่อมต่อไปยังเครื่อง Server
  - เครื่อง Server ตอบกลับโดยส่ง Synchronization and Acknowledgement (SYN, ACK) กลับไปยังเครื่อง Client
  - เครื่อง Client ตอบกลับว่าได้รับแล้ว Acknowledgement (ACK) ไปทางเครื่อง Server อีกครั้งหนึ่ง

Secure Shell (SSH)



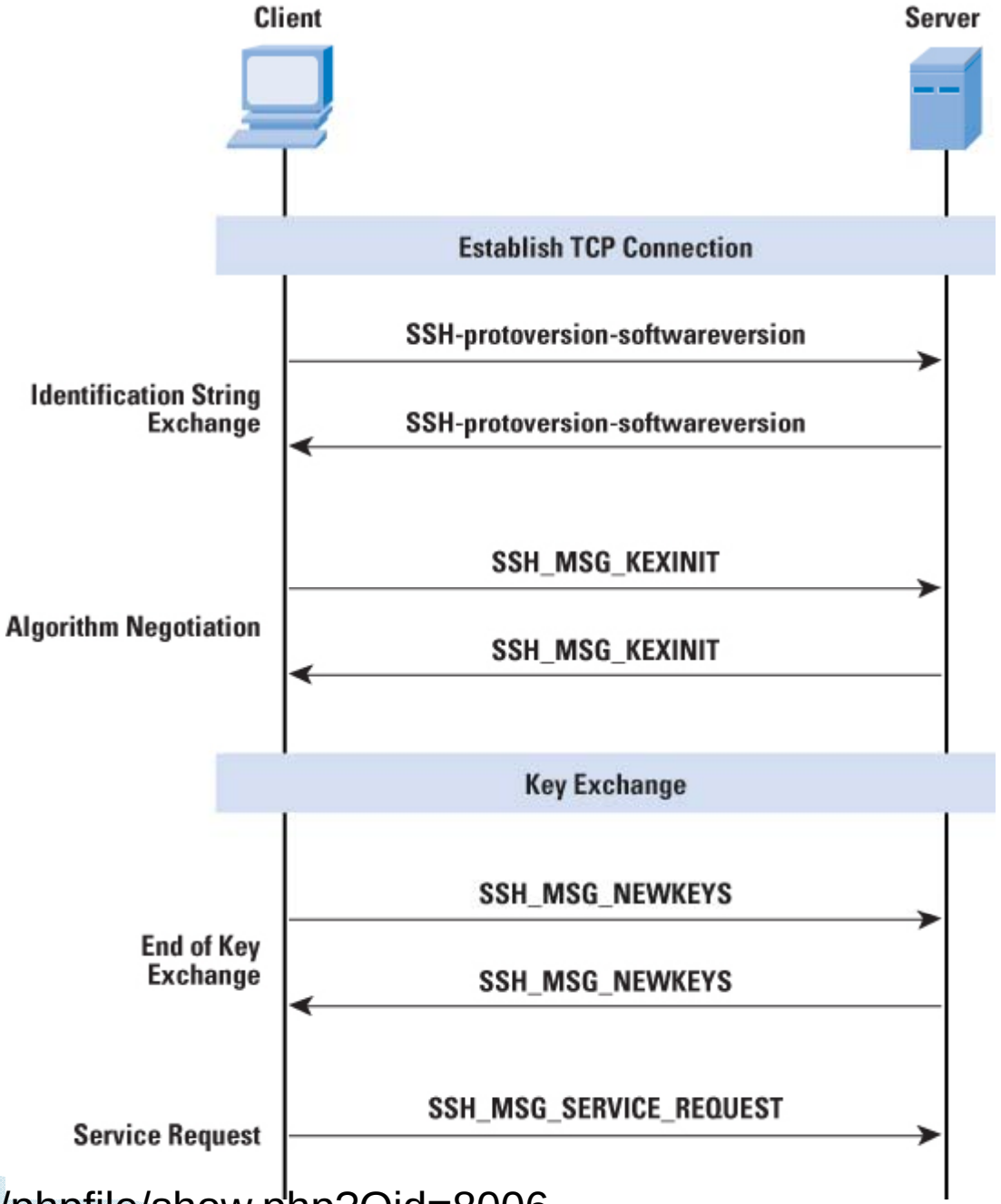
Secure Shell (SSH)

- **Identification String Exchange** เป็นการแลกเปลี่ยนชื่อและรุ่นของซอฟต์แวร์ที่ให้บริการโปรโตคอล SSH
- **Algorithm Negotiation** คือการเจรจาเรื่องอัลกอริทึ่มทีจะใช้ โดยทั้งสองฝั่งจะทำการส่ง อัลกอริทึ่มระหว่างกัน หลังจากนั้นจะทำการเลือกอัลกอริทึ่มที่รองรับการใช้งานบนเครื่องเซิร์ฟเวอร์และ เครื่องไคเอนต์ ซึ่งอัลกอริทึ่มหลักแบ่งออกเป็น 3 ส่วน
  - Encryption การเข้ารหัสลับ เช่น AES128-CBC, 3DES-CBC, Blowfish-CBC, Cast128-CBC, Twofish-CBC, Arcfour
  - Authentication การพิสูจน์ตัวจริง เช่น HMAC-MD5, HMAC-SHA1
  - Compression การบีบอัดข้อมูล เช่น none (ไม่มีการใช้งาน), zlib

Secure Shell (SSH)

- **Key Exchange** การแลกเปลี่ยนโดยใช้เทคนิคการกระจายและแลกเปลี่ยน Session Keys โดยใช้ Public parameter and Private parameter ซึ่งจะต้องเก็บ Private parameter ไว้เป็นความลับในขณะที่สามารถส่ง Public parameter ให้แก่ผู้อื่น
- **End of Key Exchange** สิ้นสุดการแลกเปลี่ยนกุญแจ โดยทั้งสองฝ่ายจะแลกเปลี่ยนพารามิเตอร์สาธารณะซึ่งกัน
- **Service Request** ทางฝั่งเครื่องไคเอนต์ จะส่งแพ็กเกจร้องขอใช้งานเซอร์วิส ของ โปรโตคอลการพิสูจน์ตัวตนของผู้ใช้งาน หรือ โปรโตคอลการเชื่อมต่อ

Secure Shell (SSH)

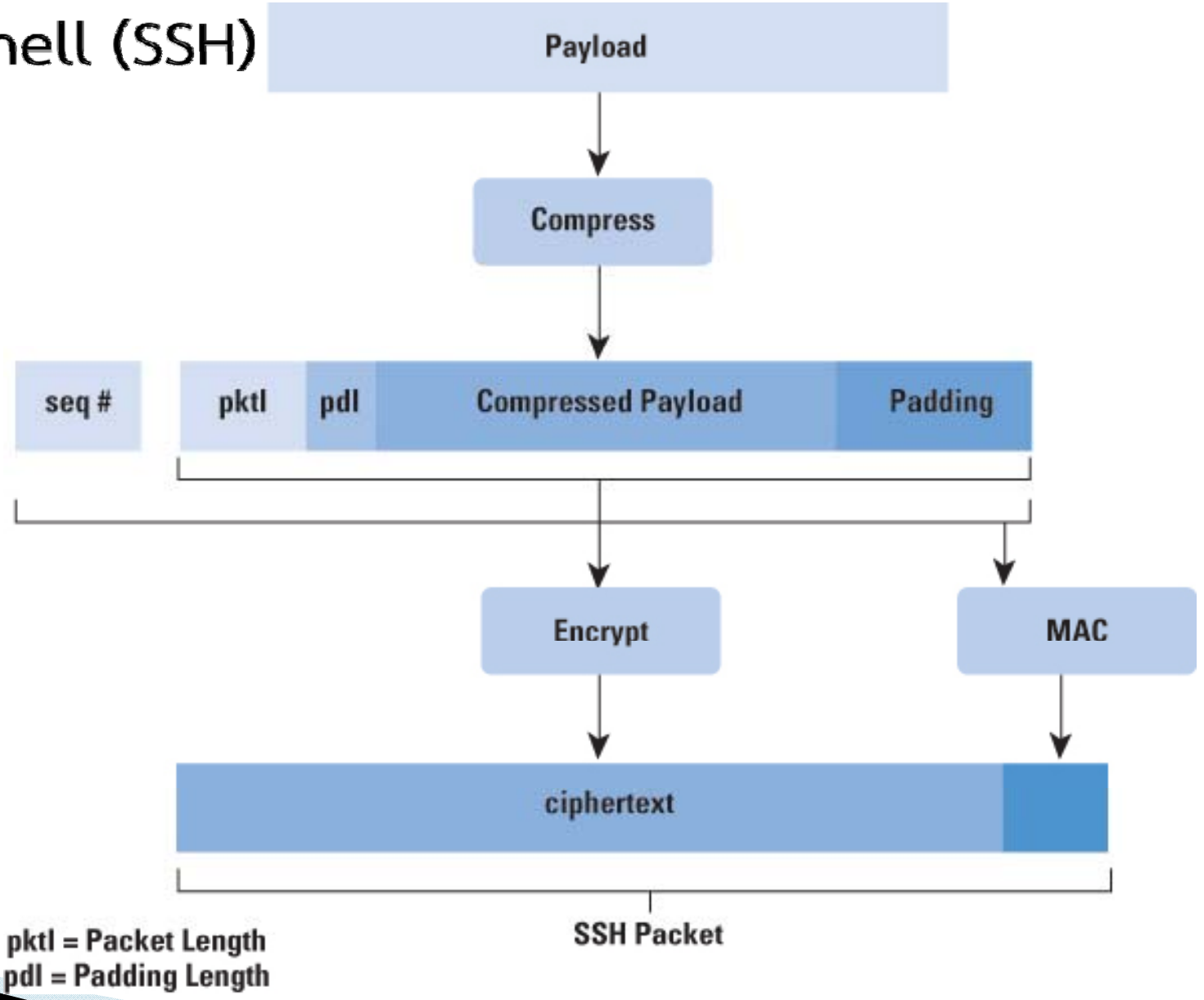


http://www.msit.mut.ac.th/ne[illegible]ph/phpfile/show.php?Qid=8006

เมื่อช่องทางการเชื่อมต่อถูกสร้างขึ้นจนเกิดการแลกเปลี่ยนข้อมูลระหว่างเครื่องไคเอนต์และเครื่องเซิร์ฟเวอร์ เราจะเรียกข้อมูลนี้ว่าแพ็กเกจ โดยจะอยู่ในส่วนพื้นที่ข้อมูลของทีซีพีเซกเมนต์ (Data field of a TCP segment) ซึ่งจะมีรูปแบบดังนี้

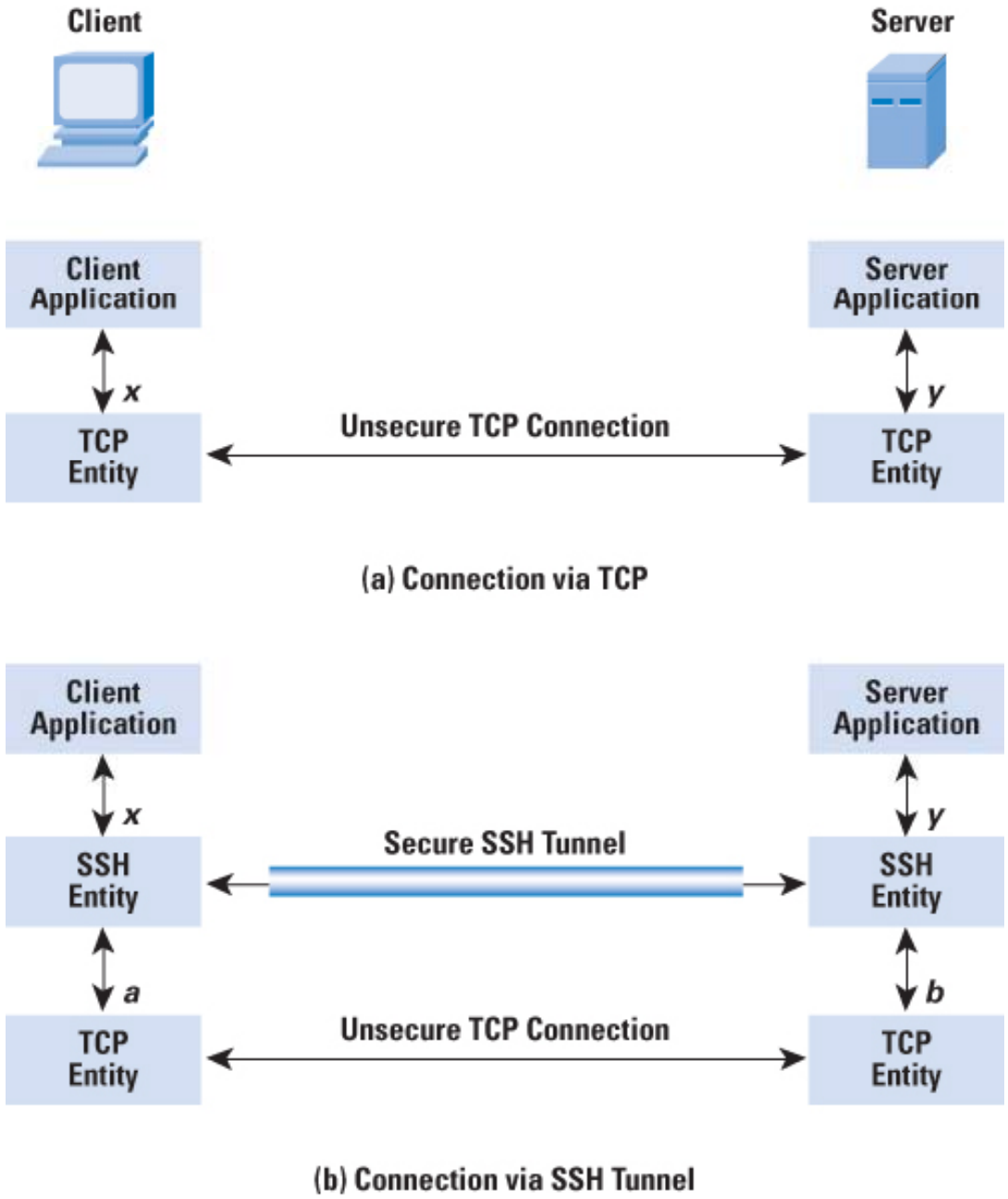
- Packet length: ความยาวของแพ็กเกจมีขนาดเป็นไบต์ ซึ่งไม่ได้รวมพื้นที่ความยาวของแพ็กเกจและพื้นที่รหัสพิสูจน์ตัวจริงข้อความ
- Padding length: ความยาวของแพ็ดดิ้งในพื้นที่ของพื้นที่แพ็ดดิ้งสุ่ม
- Payload: ข้อมูลที่ถือว่าเป็นเนื้อหาจริงของแพ็กเกจ
- Random padding: แพ็ดดิ้งสุ่ม เป็นพื้นที่สุ่มเพื่อที่ขนาดของแพ็กเกจโดยร่วมจะได้ไม่แน่นอน
- Message Authentication Code (MAC): รหัสพิสูจน์ตัวจริงข้อความใช้ในการตรวจความคงสภาพของข้อมูล โดยจะมีการนำ sequence number ซึ่งเป็นพื้นที่ในโปรโตคอลทีซีพีมาร่วมใช้งานด้วย

Secure Shell (SSH)



http://www.msit.mut.ac.th/newweb/phpfile/show.php?Qid=8006

Secure Shell (SSH)

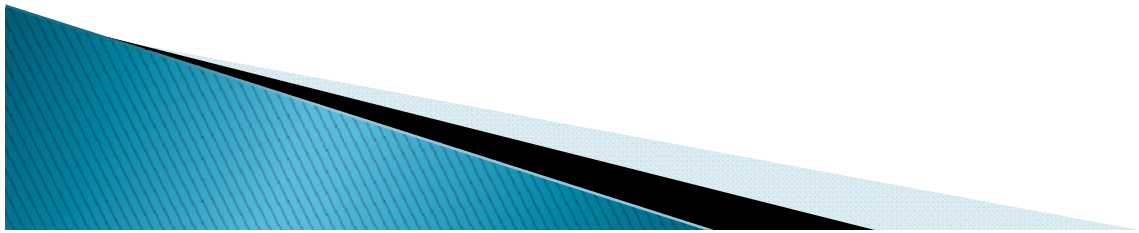


http://www.msit.mut.ac.th/ne[illegible]ph/phpfile/show.php?Qid=8006

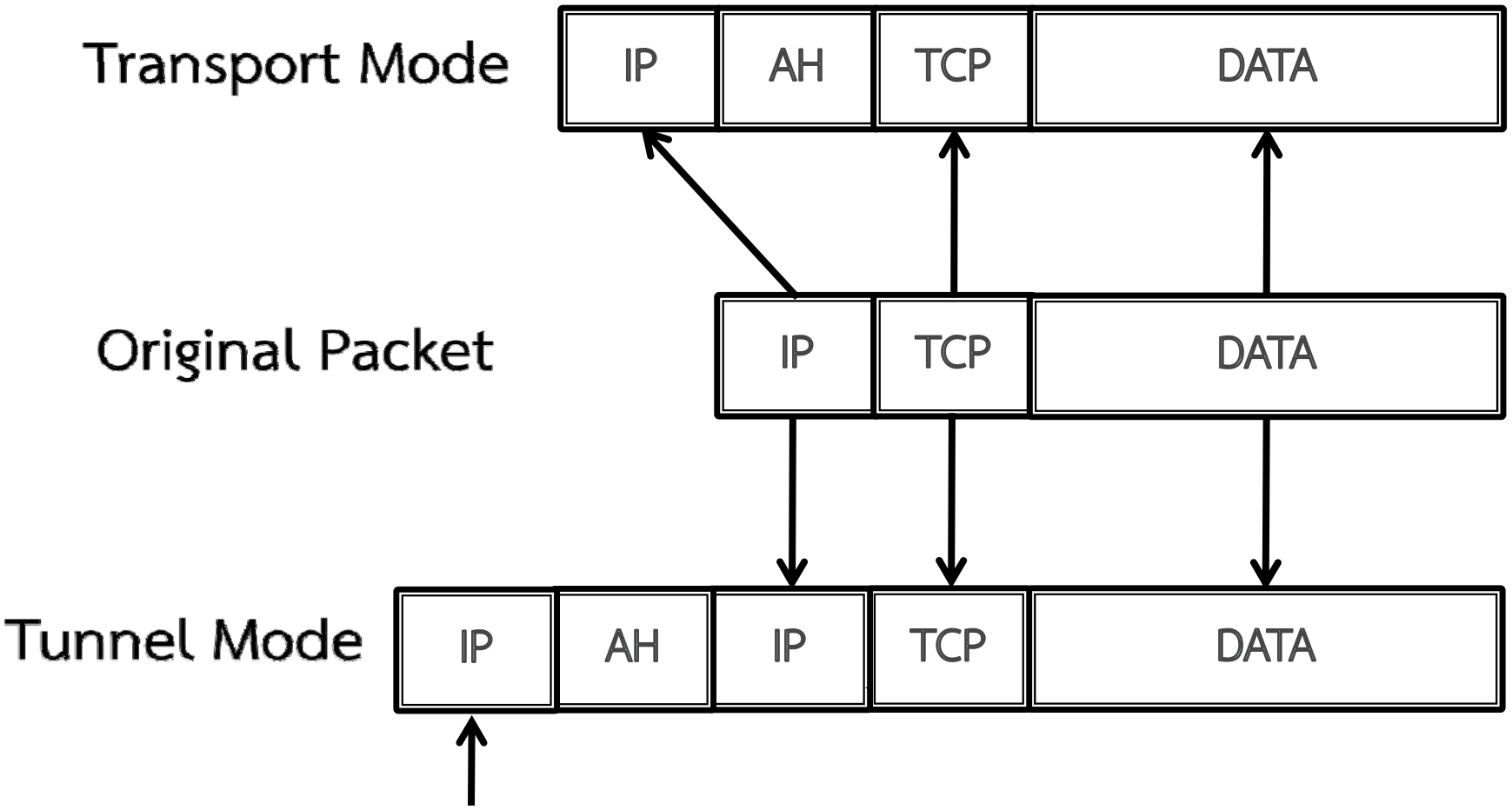
Internet Protocol Security (Ipsec)

IPsec เป็นส่วนเพิ่มขยายของ Internet Protocol (IP) ในชุด Protocol TCP/IP พัฒนาเพื่อเป็นส่วนหนึ่งของมาตรฐานของ IPv6 ซึ่งเป็น Protocol ที่พัฒนาเพื่อใช้แทน IPv4 ที่ใช้ในปัจจุบันและกำหนดหมายเลข RFC เป็น RFC24011

IPsec ใช้โพรโทคอล 2 ชุดคือ Authentication Header (AH) และ Encapsulated Security Payload (ESP) เพื่อรองรับการพิสูจน์ตัวตน (Authentication) การรักษาความถูกต้องของข้อมูล (Integrity) และการรักษาความลับ (Confidentiality) ในระดับชั้นของ IP



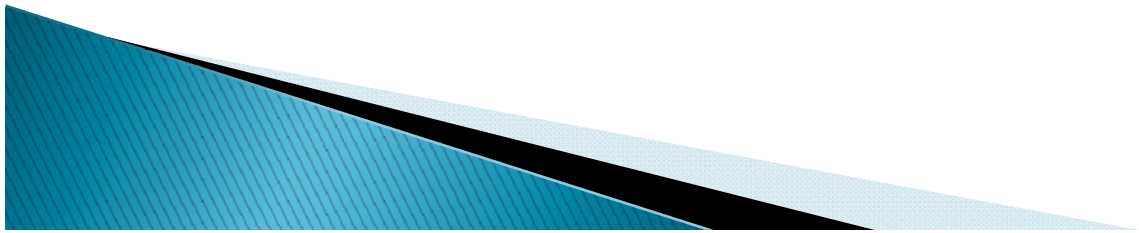
Secure Shell (SSH)



Internet Protocol Security (Ipsec)

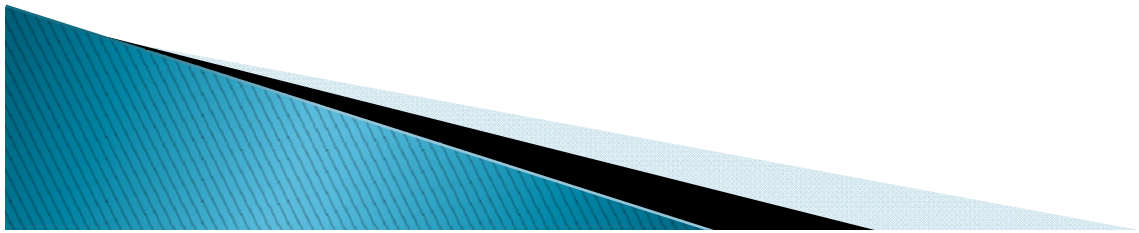
การทำงานของ IPsec แบ่งเป็น 2 โหมดการทำงานได้แก่

- **Tunnel mode** เป็นการนำส่วนแพ็กเก็ตเดิมทั้งหมดมาครอบด้วย IP โพรโทคอลชุดใหม่ที่เป็นไปตามชุดโพรโทคอล IPsec สังเกตได้จากมีการเพิ่มเฮดเดอร์ IP และ AH เข้าไปข้างหน้าแพ็กเก็ตชุดเดิม
- **Transport mode** นำเฉพาะข้อมูลของโพรโทคอล IP ซึ่งจะประกอบด้วยข้อมูลของชั้น Transport (TCP หรือ UDP) และชั้นแอพพลิเคชัน โดยเพิ่มโพรโทคอล AH และเพิ่มข้อมูลใน IP เดิมให้เหมาะสมตามมาตรฐาน IPsec



Internet Protocol Security (Ipsec)

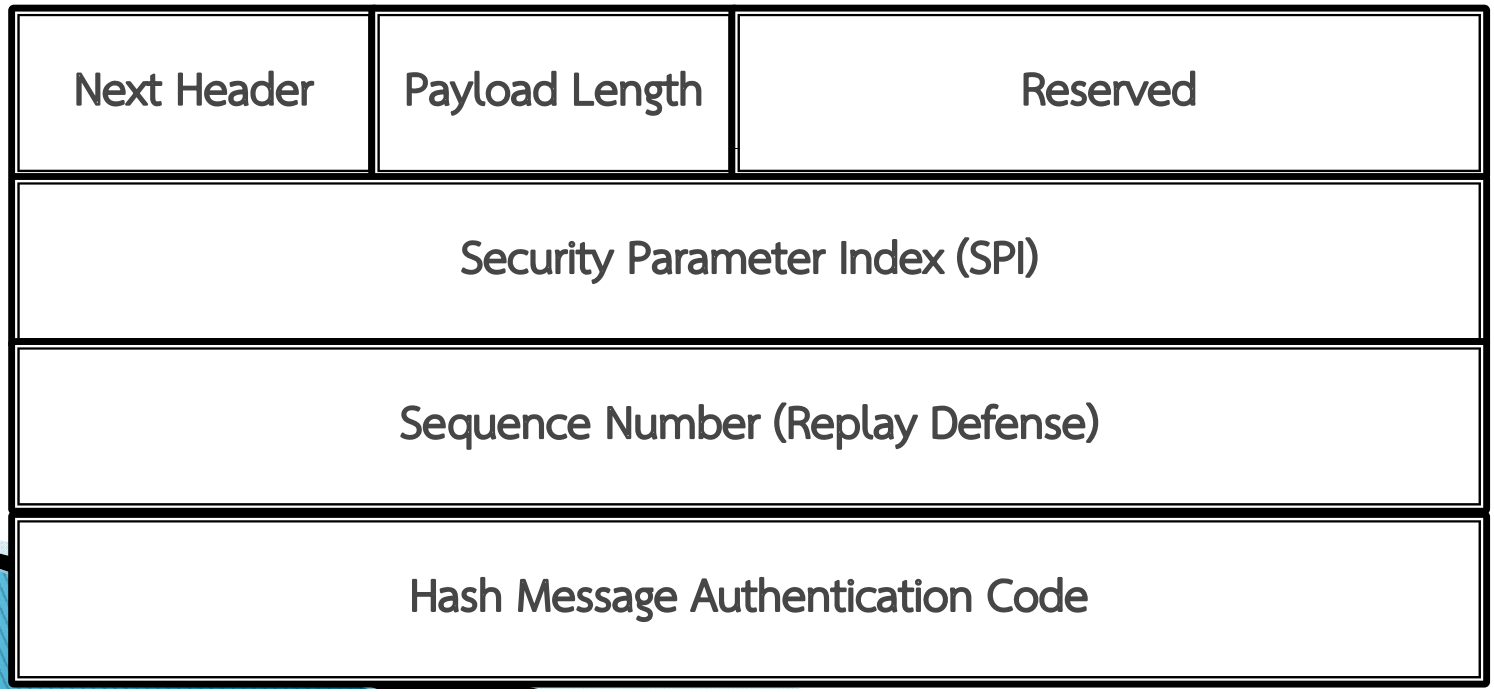
การรักษาความถูกต้องของข้อมูลของ IP ดาตาแกรม (IP Datagram) ในชุดโพรโทคอล IPsec ใช้ Hash Message Authentication Codes หรือ HMAC ด้วยฟังก์ชันแฮชเช่น MD5 หรือ SHA-1 ทุกครั้งที่มีการส่งแพ็กเก็ตจะมีการสร้าง HMAC และใช้การเข้ารหัสไปด้วยทุกครั้ง เพื่อให้ปลายทางสามารถตรวจสอบได้ตามหลักการลายเซ็นดิจิตอลว่าต้นทางเป็นผู้ส่งแพ็กเก็ตนั้นมาจริง ส่วนการรักษาความลับของข้อมูลนั้น จะใช้การเข้ารหัส IP ดาตาแกรมด้วยวิธีการเข้ารหัสด้วยกุญแจสมมาตร ด้วยวิธีการมาตรฐานที่เป็นรู้จักกันดีเช่น 3DES AES หรือ Blowfish เป็นต้น



Internet Protocol Security (Ipsec)

ชุด Protocol IPsec ประกอบด้วย Authentication Header (AH) และ Encapsulated Security Payload (ESP)

- AH หรือ Authentication Header ทำหน้าที่รักษาความถูกต้องของ IP ดาตาแกรม โดยการคำนวณ HMAC กับทุก IP ดาตาแกรม



Internet Protocol Security (Ipsec)

Header ของ AH มีขนาด 24 ไบต์ อธิบายได้ดังนี้

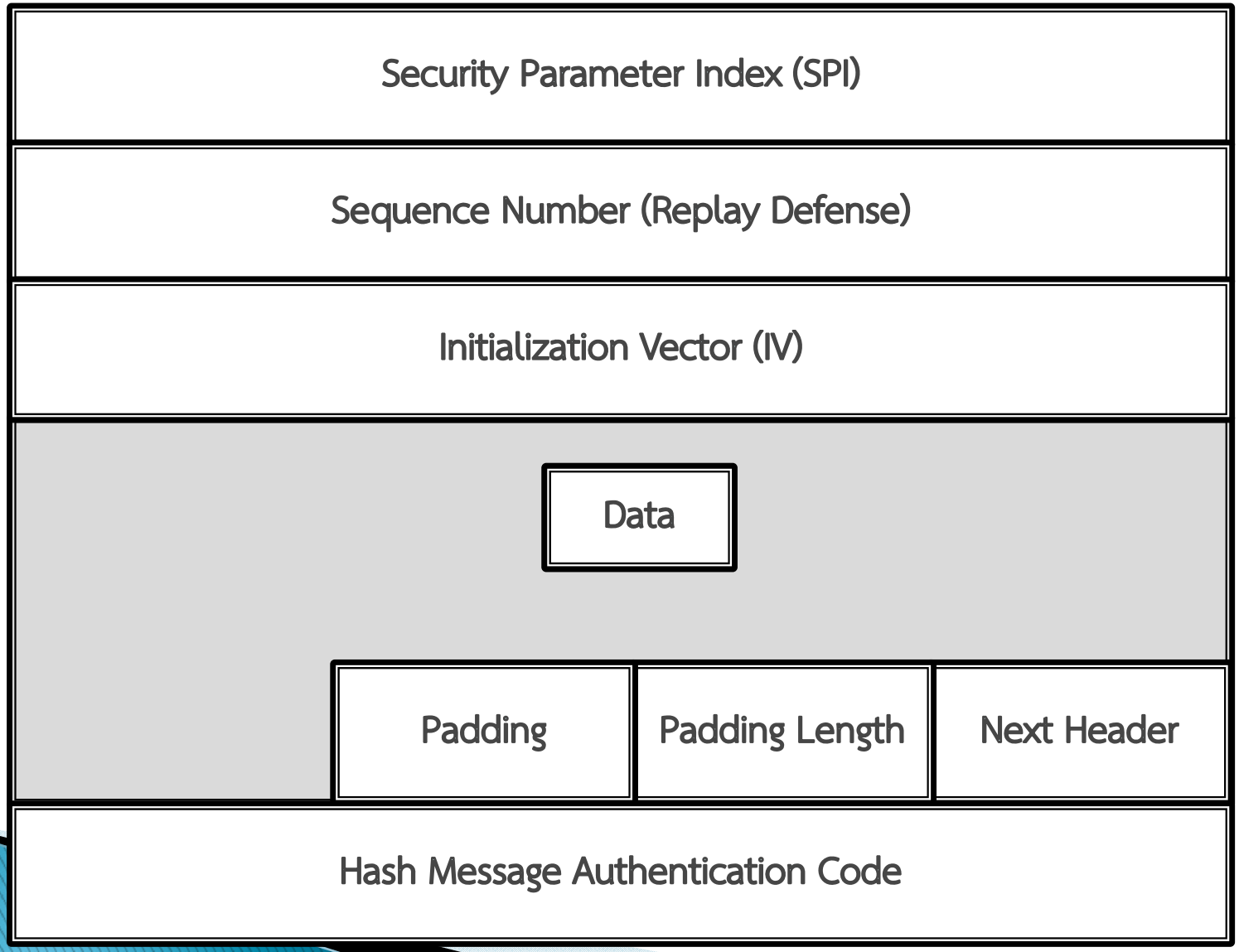
- Next Header ใช้เพื่อบอกรูปแบบในการใช้งาน IPsec ระหว่าง Tunnel mode ค่าจะเป็น 4 ส่วน Transport modeค่าจะเป็น 6
- Payload length บอกความยาวของข้อมูลที่ต่อท้ายHeader ตามด้วย Reserved จำนวน 2 ไบต์
- Security Parameter Index (SPI) กำหนด Security Association สำหรับใช้ในการถอดรหัส Packet เมื่อถึงปลายทาง
- Sequence Number ขนาด 32 บิตใช้บอกลำดับของ Packet
- Hash Message Authentication Code (HMAC) เป็นค่าที่เกิดจาก Hash Function เช่น MD5 หรือ SHA-1 เป็นต้น

Internet Protocol Security (Ipsec)

ESP หรือ Encapsulated Security Payload ใช้สำหรับรักษาความถูกต้องของ Packet โดยใช้ HMAC และการเข้ารหัสร่วมด้วย Header ประกอบด้วย

- Security Parameter Index (SPI) กำหนด Security Association (SA) ระบุ ESP ที่สอดคล้องกัน
- Sequence Number ระบุลำดับของ Packet
- Initialization Vector (IV) ใช้ในกระบวนการเข้ารหัสข้อมูล ป้องกันไม่ให้ 2 Packet มีการเข้ารหัสที่ซ้ำกันเกิดขึ้น
- Data คือข้อมูลที่เข้ารหัส
- Padding เป็นการเติม Data เพื่อให้ครบจำนวนไบต์ที่เข้ารหัสได้
- Padding Length บอกความยาวของ Padding ที่เพิ่ม
- Next Header กำหนด Header ถัดไป
- HMAC ค่าที่เกิดจาก Hash Function ขนาด 96 บิต

Internet Protocol Security (Ipsec)

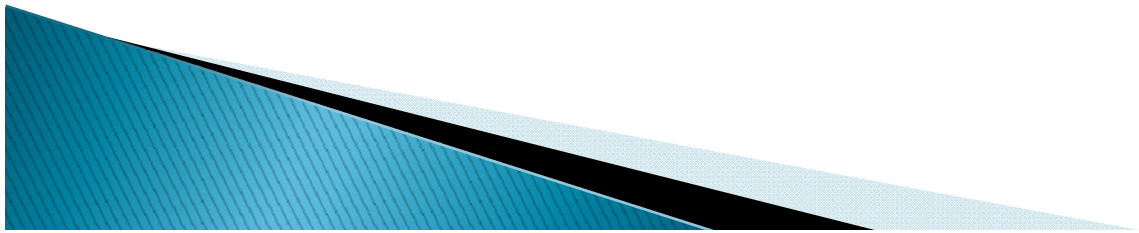


Kerberos

การพิสูจน์ตัวตนแบบ Kerberos พัฒนาขึ้นโดย Massachusetts Institute of Technology (MIT)

ระบบ Kerberos ประกอบขึ้นจากสองส่วนหลักได้แก่

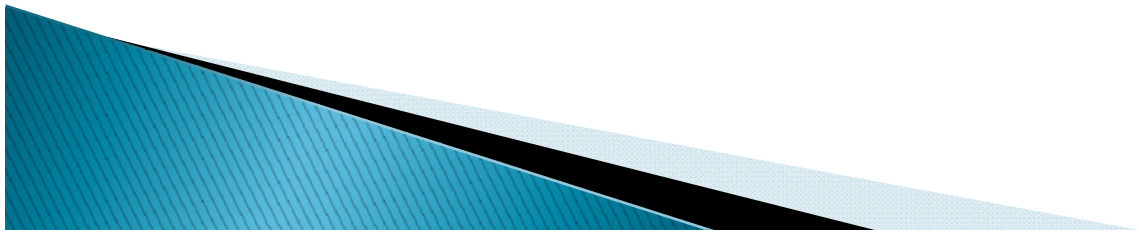
- **Ticket** ใช้สำหรับการพิสูจน์ตัวตนของผู้ใช้ในระบบ และการเข้ารหัสข้อมูล
- **Authenticator** ใช้ในการตรวจสอบ Ticket ว่าเป็นผู้ใช้คนเดียวกันที่ใช้ Ticket เป็นใบเบิกทางเข้าสู่ระบบและเป็นผู้ใช้ที่ระบบสร้างให้อย่างถูกต้อง



Kerberos

Kerberos เซิร์ฟเวอร์ มีสองส่วนบริการในการใช้งานคือ

- **Authentication service (AS)** สำหรับการพิสูจน์ตัวตนของผู้ใช้กับ Kerberos เซิร์ฟเวอร์ก่อนการเข้าใช้บริการ
- **Ticket Granting Service (TGS)** เป็นบริการที่ออก Ticket เพื่อให้ผู้ใช้นำไปใช้กับเซิร์ฟเวอร์ที่ต้องการ



Kerberos

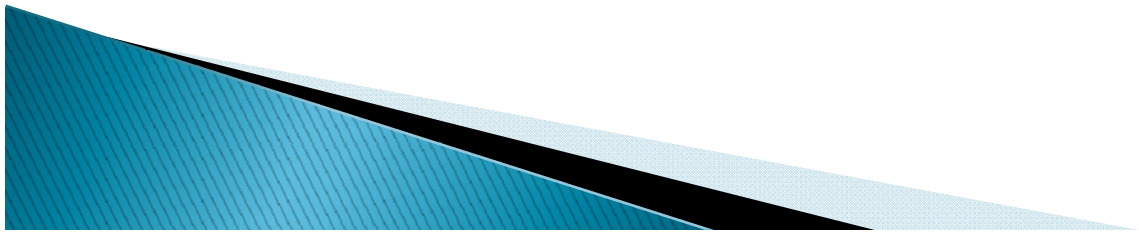
กระบวนการใช้งานระบบ Kerberos มีลำดับดังนี้

- ผู้ใช้จะทำการพิสูจน์ตัวตนครั้งแรกกับ Authentication service ของ Kerberos ซึ่งจะได้กุญแจสมมาตรซึ่งจะใช้ในการเข้ารหัสข้อมูลในการติดต่อสื่อสาร
- ก่อนผู้ใช้จะเข้าไปใช้บริการใด ๆ ในระบบได้ต้องมี Ticket ก่อน ด้วยการติดต่อไปที่ Ticket Granting Service เพื่อให้ออก Ticket ที่เหมาะสมกับการเข้าไปใช้บริการบนเซิร์ฟเวอร์ในระบบได้

-ผู้ใช้นำ Ticket สำหรับไปใช้กับการร้องขอการติดต่อการบริการจากเซิร์ฟเวอร์ในระบบ

Kerberos

ปัญหาสำคัญของการใช้ระบบ Kerberos คือการขยายระบบ เนื่องจากเซิร์ฟเวอร์ Kerberos ต้องเก็บกุญแจของผู้ใช้ทุกคนที่เข้ามาในระบบ ถ้าระบบใหญ่มากขึ้น มีการกระจายตัวมากกว่าหนึ่งจุด ย่อมส่งผลเสียต่อการใช้งานระบบโดยรวม แต่การนำระบบ Kerberos มาใช้จะเพิ่มความสะดวกในการพิสูจน์ตัวตนได้มากขึ้น มักเรียกการใช้งาน Kerberos ว่าเป็นระบบ Single Sign-On แบบหนึ่ง คือการเข้าถึงการใช้บริการของระบบทั้งหมดได้ด้วยการพิสูจน์ตัวตนเพียงครั้งเดียว



Questions and Answers

MS1 Thanin Muangpool

Thank you