การเข้ารหัสลับ Cryptographic

เทคโนโลยีการเข้ารหัสลับ

เทคโนโลยีการเข้ารหัสลับหรือวิทยาการการเข้ารหัสลับนั้น เริ่มเป็นที่รู้จักกันมาตั้งแต่สมัยโรมัน พัฒนาจากแนวคิดเกี่ยวกับ พื้นฐานในการรักษาความปลอดภัยของข้อมูลอิเล็กทรอนิกส์และพัฒนาเรื่อยมาจนถึงปัจจุบัน ซึ่งกลายเป็นกระบวนการทางคณิตศาสตร์ ในการเข้ารหัสลับ (Cryptographic Algorithms) โดยการสร้างสิ่งที่อยู่ในรูปตัวอักษร อักขระ ตัวเลข หรือสัญลักษณ์ใดๆ ขึ้นมา และเรียกสิ่งนั้นว่า “กุญแจ (key)” และใช้ “กุญแจ (key)” เป็นกลไกสำคัญในการ “เข้ารหัส” และ “ถอดรหัส”

“การเข้ารหัส (encryption)” และ “การถอดรหัส (decryption)” “การเข้ารหัส” หมายถึง การแปลงข้อความหรือข้อมูลอิเล็กทรอนิกส์รูปหนึ่งที่อ่านได้ (plaintext) ให้อยู่ในอีกรูปแบบหนึ่ง ที่เปลี่ยนแปลงไปจากเดิมซึ่งอ่านไม่ได้ (ciphertext) ส่วน “การถอดรหัส” หมายถึงการแปลงข้อมูลอิเล็กทรอนิกส์จากรูปแบบที่เปลี่ยนแปลงไปจากเดิม (ciphertext) ให้กลับไปอยู่ในรูปของข้อความหรือข้อมูลอิเล็กทรอนิกส์รูปแบบเดิมก่อนการเปลี่ยนแปลง (plaintext) สำหรับกระบวนการข้างต้นในการแปลงข้อมูลอิเล็กทรอนิกส์ที่อ่านได้เป็นข้อมูลอิเล็กทรอนิกส์ที่อ่านไม่ได้จะนี้เรียกว่า “การเข้ารหัส (Encryption) และการแปลงข้อมูลอิเล็กทรอนิกส์กลับให้อยู่ในรูปของข้อมูลอิเล็กทรอนิกส์ที่อ่านได้เรียกว่า “การถอดรหัส (Decryption)”

“กุญแจ (key)” คำว่า “กุญแจ (key)” ซึ่งเป็นกลไกสำคัญในการเข้ารหัสหรือถอดรหัสนั้น จะสร้างขึ้นด้วยกระบวนการทางคณิตศาสตร์ที่คำนวณโดยอัตโนมัติ และได้ผลลัพธ์ซึ่งอาจจะอยู่ในรูปของ อักษร อักขระ ตัวเลข หรือสัญลักษณ์ใดๆ ก็ได้ เช่น

D3K7EF8C9FE98A4B58CB2A57FD814BF78BC3D98B15FE8A
4FA8EB33C2F5D569FFB4A0012CF16EDA45CEF79AA5F1D3
AF7D9B46CF711CE84DEA011BF8A2D75F9CA701AD4B8A9F

คำว่า “กุญแจ” ในที่นี้จึงต่างไปจาก “กุญแจ” เป็นดอกๆ สำหรับใช้ไขแม่กุญแจทั่วไป แต่เหตุที่เรียกว่า “กุญแจ” อาจจะด้วยวัตถุประสงค์ในการสร้างขึ้นมาในการใช้เข้ารหัสโดยแปลงข้อความหรือตัวหนังสือที่อ่านเข้าใจได้ (plaintext) ให้อยู่ในรูปของข้อมูลอิเล็กทรอนิกส์ซึ่งอ่านไม่ได้หรืออ่านไม่เข้าใจ (ciphertext)และในการใช้ เพื่อถอดรหัสโดยทำหน้าที่ในการแปลงข้อความที่อ่านไม่ได้หรืออ่านไม่เข้าใจวัตถุประสงค์ของสิ่งๆนั้นให้อยู่ในรูปของข้อความที่อ่านได้ หรือสามารถเข้าใจได้ถึงวัตถุประสงค์ของสิ่งๆ นั้น การทำงานของ “กุญแจ” โดยทำให้ข้อความนั้นเป็นความลับจึงคล้ายกับการปิดไม่ให้บุคคลอื่นได้รับรู้หรือเข้าถึงหรือเข้าใจ และสามารถใช้ไขความลับของข้อความนั้นได้คล้ายกับการเปิดออกอ่านได้ จึงน่าจะเป็นที่มาของการใช้คำว่า “กุญแจ”

ความต้องการของเทคโนโลยีการเข้ารหัสข้อมูล

การระบุตัวบุคคลได้ (Authenticity)

คือการที่เราสามารถที่จะระบุตัวตนของผู้ที่การเข้าถึงข้อมูลภายในระบบได้

การรักษาความลับ (Confidentiality)

คือความสามารถในการที่จะรักษาความลับที่ไม่ให้ผู้อื่นที่ไม่มีสิทธิ์เข้าถึงข้อมูลภายในระบบได้

การรักษาความลับ (Integrity)

คือความสามารถในการรักษาความถูกต้องและสมบูรณ์ของข้อมูล

การป้องกันการปฏิเสธความรับผิดชอบ (Non-repudiation)

คือความสามารถในการป้องกันการปฏิเสธความรับผิดชอบของการเข้าถึงข้อมูลภายในระบบ

กุญแจคู่ (key pairs)

“กุญแจคู่” จะประกอบด้วยกุญแจสองข้างที่สร้างขึ้นมาพร้อมกันด้วยกระบวนการทางคณิตศาสตร์ที่เรียกว่า “ระบบรหัสแบบอสมมาตร” โดยกุญแจข้างหนึ่งเรียกว่า “กุญแจส่วนตัว (private key)” ส่วนอีกข้างเรียกว่า “กุญแจสาธารณะ (publickey)” เหตุที่เรียกต่างกันเพราะลักษณะการทำงานของกุญแจทั้งสองข้างที่ต่างกัน กล่าวคือ “กุญแจส่วนตัว” นั้น ใช้ในการสร้างลายมือชื่อดิจิทัลเพื่อระบุหรือยืนยันตัวบุคคล ส่วน “กุญแจสาธารณะ” นั้นใช้ในการตรวจสอบลายมือชื่อ ดิจิทัล กุญแจทั้งสองข้างซึ่งสร้างขึ้นมาพร้อมกันนี้จึงเป็นกุญแจที่มีความสัมพันธ์กันในเชิงตรรกะซึ่งต้องใช้ควบคู่กันเสมอ

-“กุญแจส่วนตัว (private key)” หมายความถึง กุญแจที่ใช้ในการสร้างลายมือชื่อดิจิทัล

-“กุญแจสาธารณะ (public key)” หมายความถึง กุญแจที่ใช้ในการตรวจสอบลายมือชื่อดิจิทัล

ความยาวของกุญแจรหัส (Key Length)

Key Length in Bits	Number of Possible Keys
1	2
2	4
4	16
8	256
16	65,536
40	1,099,511,627,776
56	72,057,594,037,927,900
112	5,192,296,858,534,830,000,000,000,000,000
168	3.74144E+50
256	1.15792E+77
512	1.3408E+154

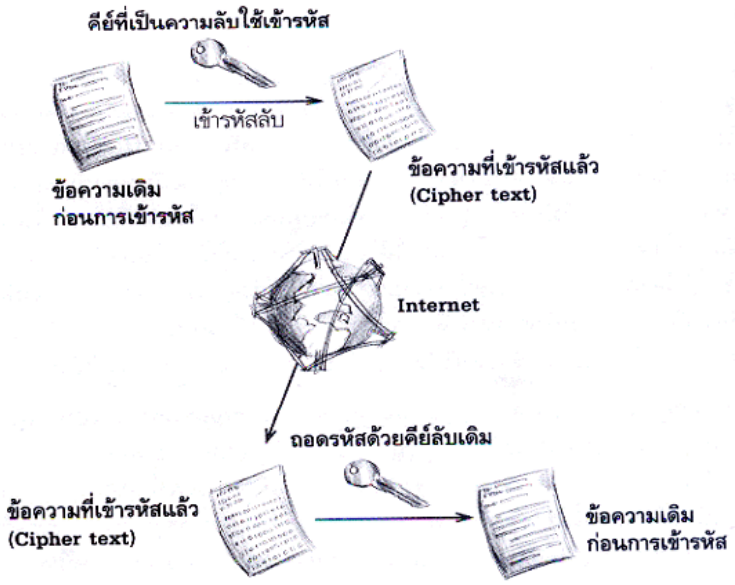
ตารางที่1 ความยาวบิตของกุญแจรหัสและจำนวนของกุญแจรหัสที่อาจเกิดขึ้นได้จากความยาวบิตนั้น

เหมือนกับเรื่องของรหัสผ่าน ที่ยิ่งรหัสผ่านมีความยาวมากเท่าใด ก็ยิ่งเป็นการยากขึ้นเท่านั้นที่จะคาดเดารหัสผ่านได้ ในเรื่องความยาวของกุญแจรหัสก็เช่นกัน ยิ่งมีความยาวมากขึ้น ก็ยิ่งทำให้เป็นการยากขึ้นต่อการค้นหาความเป็นไปได้ของรหัสผ่าน (Exhaustive Search) ทุกๆ บิตที่เพิ่มขึ้น ทำให้ต้องใช้เวลาในการสืบค้นมากขึ้นเป็นสองเท่า เพราะฉะนั้นแล้ว การเพิ่มจำนวนบิตของรหัสผ่านเพียงไม่กี่บิต ก็มีผลทำให้กระบวนการค้นหารหัสผ่านอย่างละเอียด (Exhaustive Search) ต้องใช้เวลาเพิ่มมากยิ่งขึ้นเท่านั้น จากตารางที่1 แสดงให้เห็นว่า ยิ่งจำนวนบิตของกุญแจรหัสมีความยาวมากขึ้น ความเป็นได้ของการเกิดกุญแจรหัสก็ยิ่งมาก ทำให้ยากต่อการค้นหาหรือคาดเดารหัสผ่าน

ประเภทระบบการเข้ารหัส

อย่างไรก็ตาม วิทยาการการเข้ารหัสนั้นสัมพันธ์อย่างยิ่งกับกลไกการทำงานของ “กุญแจ” ที่สร้างขึ้นให้อยู่ในรูปของข้อมูลอิเล็กทรอนิกส์เพราะในการเข้ารหัสแต่ละครั้งอาจใช้กุญแจเพียงแค่ข้างเดียวหรือหลายข้างต่างวัตถุประสงค์กันไป จึงทำให้สามารถแยกประเภทของการเข้ารหัสตามจำนวนกุญแจที่นำมาใช้ได้ ดังนี้

ระบบการเข้ารหัสแบบกุญแจสมมาตร (Symmetric Key Cryptosystem)

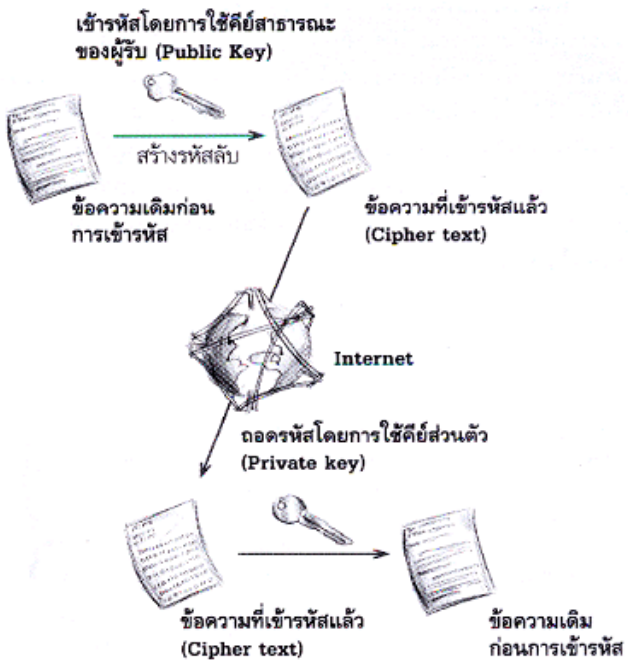


รูปที่1 รูปการเข้ารหัสแบบ Secret Key Encryption

การเข้ารหัสแบบสมมาตรนี้อาจจะเป็นการเข้ารหัสอย่างง่าย เช่น กำหนดเพียงให้เลื่อนพยัญชนะออกไปอีก 1 ตำแหน่งกล่าวคือคำว่า “กฎหมาย” หากเลื่อนตำแหน่งพยัญชนะไป 1 ตัว ก็จะปรากฏเป็นดังนี้ “ขฏพาย” แทนคำว่า “กฎหมาย”จะเป็นการนำข้อมูลอิเล็กทรอนิกส์แบบธรรมดาเข้ารหัสโดยการแปลงข้อมูลนั้นให้อยู่ในรูป ที่ไม่สามารถอ่านได้ด้วยการใช้กุญแจดอกเดียวกันหรือสูตรเดียวกันผ่านกระบวนการทางคณิตศาสตร์ทั้งในการเข้ารหัสและถอดรหัสเพื่อ แปลงข้อมูลอิเล็กทรอนิกส์ที่อ่านไม่ได้ให้เป็นข้อมูลอิเล็กทรอนิกส์ที่อ่านได้ ดังนั้นเมื่อใช้กุญแจในการเข้ารหัสแล้วก็ต้องส่งมอบกุญแจนั้นให้กับผู้รับอีกฝ่าย ซึ่งต้องใช้กุญแจดอกเดียวกันในการถอดรหัส และต้องมีการเก็บรายละเอียดเกี่ยวกับกุญแจไว้เป็นความลับเพื่อความปลอดภัยของข้อมูลอิเล็กทรอนิกส์ กรณีที่ไม่ประสงค์ ให้บุคคลที่สามหรือบุคคลอื่นได้ล่วงรู้อันอาจนำกุญแจไปใช้ในทางมิชอบโดยการเปิดเผยข้อมูลให้สาธารณะชนได้รับรู้ อย่างไรก็ตาม ระบบการเข้ารหัสแบบสมมาตรมีข้อดี คือ อาจตกลงให้มีการเข้ารหัสแบบง่ายๆ เช่น การเลื่อนพยัญชนะ หรือกรณีที่มีการใช้เทคโนโลยีซับซ้อนขึ้น การเข้ารหัสแบบนี้ก็จะช่วยให้สามารถเข้ารหัสและถอดรหัสได้รวดเร็ว แต่ก็มีข้อเสียเพราะข้อตกลงให้มีการเข้ารหัสแบบง่ายๆ อาจทาให้บุคคลอื่นล่วงรู้ได้ง่าย และในกรณีที่มีการใช้ระบบกุญแจก็จะประสบปัญหาในด้าน

การบริหารจัดการกุญแจเพราะในการใช้กุญแจเพื่อเข้ารหัส และถอดรหัสนั้นจะต้องใช้กุญแจอันเดียวกัน ผู้สร้างกุญแจจึงต้องแจ้งให้บุคคลอื่นทราบเพื่อใช้ในการถอดรหัส ซึ่งการจะทำสำเนาให้บุคคลหลายคนเพื่อใช้ร่วมกันก็อาจจะก่อให้เกิดปัญหาในการระบุตัวบุคคล การแสดงความผูกพันหรือความรับผิดที่เกิดขึ้นจากการทำธุรกรรมในครั้งนั้น ดังนั้น โดยทั่วไปในการใช้กุญแจในระบบสมมาตรจึงมักมีการสร้างกุญแจขึ้นแบบสำหรับคนสองคนใช้ร่วมกัน ดังนั้นหากมีหลายคน ถ้าไม่ต้องการให้กุญแจซ้ำกันก็ต้องให้กุญแจหลายดอก เป็นจำนวนมากเพื่อความคล่องตัว และสะดวกในการใช้งานสำหรับกรณีที่ต้องติดต่อสื่อสารกับคนเป็นจำนวนมาก เช่น คนสี่คนติดต่อกันจะต้องใช้กุญแจคนละ 3แบบ รวมทั้งสิ้นมีคู่กรณีได้ 6 คู่ รวมกุญแจทั้งสิ้น 6 แบบ ถ้าคน 100 คน จะต้องใช้กุญแจจำนวนมากซึ่งก็จะเกิดปัญหามากมายติดตามมาเช่นกันในการบริหารจัดการกุญแจซึ่งมีเป็นจำนวนมาก

ระบบการเข้ารหัสแบบกุญแจอสมมาตร (Asymmetric Key Cryptosystem)



รูปที่2 รูปการเข้ารหัสแบบ Public Key Encryption

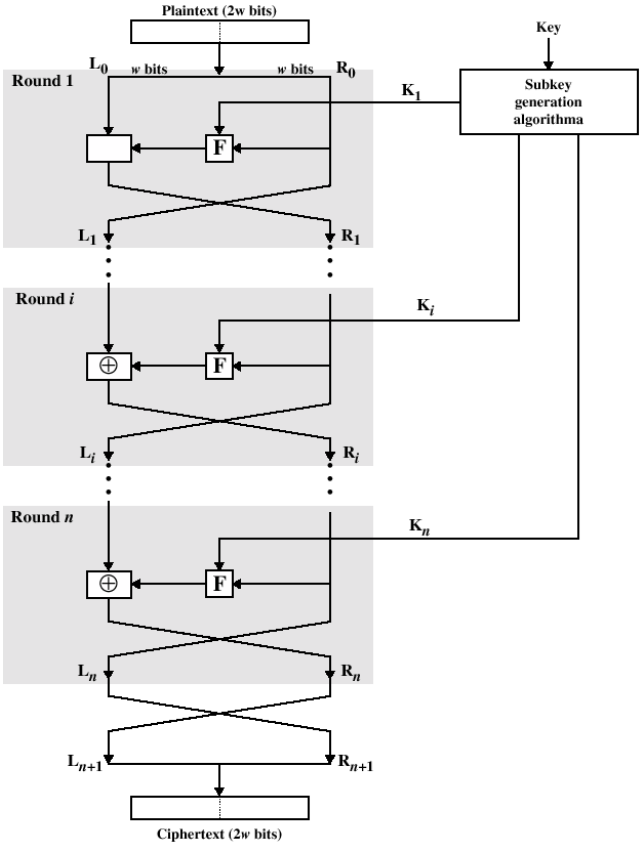
ระบบการเข้ารหัสแบบอสมมาตรเป็นการเข้า และถอดรหัสโดยใช้กุญแจสองดอก กุญแจข้างหนึ่งใช้เข้ารหัส อีกข้างหนึ่งหรืออีกดอกหนึ่งใช้ในการถอดรหัส ข้างที่ใช้ในการเข้ารหัสต้องเก็บไว้เป็นความลับ ส่วนข้างที่ใช้ในการถอดรหัสไม่จำต้องเก็บไว้เป็นความลับแต่อย่างใด (หรือจะใช้กลับกันก็ได้แล้วแต่วัตถุประสงค์) กุญแจที่สร้างขึ้นจะสร้างขึ้นพร้อมกันเรียกว่า “กุญแจคู่ (Key Pair)” ข้างที่ใช้ในการเข้ารหัสเรียกว่า “กุญแจส่วนตัว (Private Key)” ส่วนอีกข้างใช้ในการถอดรหัสเรียกว่า “กุญแจสาธารณะ (Public Key)” และโดยทั่วไปกุญแจทั้งสองข้างแม้สร้างขึ้นมาพร้อมกันแต่ก็จะมีลักษณะไม่เหมือนกัน ยิ่งไปกว่านั้น การเข้ารหัส

และถอดรหัส หากใช้กุญแจข้างเดียวกัน จะไม่ได้ผลต้องใช้อีก ข้างหนึ่งซึ่งสร้างขึ้นมาพร้อมกันเท่านั้นจึงจะบรรลุวัตถุประสงค์ในการนำไปใช้ได้

Feistel Cipher Structure

ในการเข้ารหัสข้อมูลทุกรูปแบบที่ใช้อัลกอริทึมในกลุ่มที่เป็นแบบ Conventional นี้ จะมีโครงสร้างในการเข้ารหัสโดยรวม เป็นแบบที่นำเสนอโดย Horst Feistel แห่ง IBM ในปี 1973 ซึ่งแสดงในรูปที่3 โดยอินพุตจะป้อนเข้าสู่อัลกอริทึมในการเข้ารหัสในลักษณะที่เป็นกลุ่มของข้อมูลที่มีความยาวเท่า ๆ กัน หรือเรียกว่า Block โดยจะมีความยาวของกลุ่มข้อมูลเท่ากับ 2w โดยใช้คีย์ Key K จากนั้นกลุ่มข้อมูลจะแบ่งออกเป็น 2 ส่วนเท่าๆ กัน คือ L0 และ R0 จากนั้นข้อมูลทั้ง 2 ส่วน จะป้อนเข้าสู่การประมวลผล โดยหลักของการเข้ารหัสในแบบนี้ จะอาศัยการประมวลผลซ้ำ ๆ กัน เพื่อสร้างความซับซ้อน โดยในแต่ละรอบของการประมวลผลอินพุต Li-1 และ Ri-1 จะได้มาจากผลของการประมวลผลก่อนหน้า และคีย์ย่อย Ki ก็จะได้มาจากคีย์ K ซึ่งโดยทั่วไปแล้ว คีย์ย่อยที่จะใช้ในแต่ละรอบของกรประมวลผลจะต้องไม่ซ้ำกัน

ในแต่ละรอบของการประมวลผลจะประกอบด้วยการใช้ Round Function F ทำกับข้อมูล R และนำผลลัพธ์ทีได้ไป XOR กับข้อมูล L โดย Round Function ที่ใช้จะต้องเป็นฟังก์ชันเดียวกันทั้งหมด แต่เปลี่ยนคีย์ไปเรื่อย ๆ เท่านั้น ซึ่งโดยทั่วไปแล้ว ความซับซ้อน หรือ ความยากในการแกะรหัส จะขึ้นอยู่กับ การออกแบบในส่วนต่าง ๆ โดยมีข้อพิจารณาในการออกแบบอัลกอริทึมในแต่ละส่วนดังนี้



รูปที่3 รูปการเข้ารหัสแบบ Public Key Encryption

- ขนาดของบล็อกข้อมูล บล็อกข้อมูลที่มีขนาดใหญ่ จะมีความปลอดภัยมากขึ้น แต่จะทำให้ความเร็วในการเข้าและถอดรหัสลดลงด้วย ปกติจะถือว่าบล็อกที่มีขนาด 64 บิต ถือว่ามีความเหมาะสม

- ขนาดของคีย์ ขนาดคีย์ที่มีขนาดใหญ่ จะมีความปลอดภัยมากขึ้น แต่จะทำให้ความเร็วในการเข้าและถอดรหัสลดลงด้วย ปัจจุบันในอัลกอริทึมสมัยใหม่จะถือว่าคีย์ที่มีความยาว 128 ถือว่ามีความปลอดภัยเพียงพอ

- จำนวนครั้งของการประมวลผล ยิ่งทำมากรอบจะยิ่งถอดรหัสได้ยากขึ้น ปกติจะถือว่าประมาณ 16 รอบ มีความเหมาะสม

- อัลกอริทึมที่ใช้ในการสร้างคีย์ย่อย ยิ่งอัลกอริทึมซับซ้อนจะยิ่งทำให้การแกะรหัสยากขึ้น

- ฟังก์ชัน Round ยิ่งฟังก์ชันซับซ้อนจะยิ่งทำให้การแกะรหัสยากขึ้น

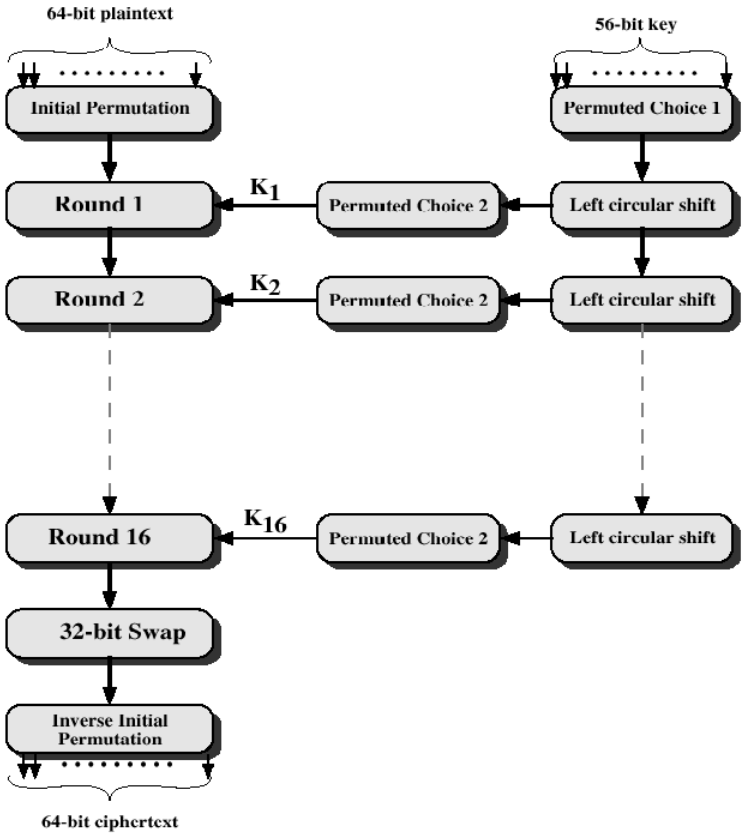
สำหรับการถอดรหัสแล้ว จะมีกระบวนการเดียวกับการเข้ารหัส แต่จะใช้คีย์ย่อยย้อนกลับ

Conventional Encryption Algorithm

ในการเข้ารหัสแบบ Block Cipher นั้น มีอัลกอริทึมอยู่หลายตัวที่มีการออกแบบและใช้งานในปัจจุบัน โดยอัลกอริทึมที่มีความสำคัญ และใช้งานมากที่สุด คือ DES (Data Encryption Standard) และ 3DES (Triple Data Encryption Standard) ซึ่งถือว่าเป็นรากฐานของระบบเข้ารหัสที่ใช้ในปัจจุบัน อย่างไรก็ตาม เนื่องจาก DES และ 3DES ได้มีการใช้งานมาระยะหนึ่งแล้ว และเริ่มจะมีความปลอดภัยน้อยลง จึงได้มีความพยายามออกแบบอัลกอริทึมในการเข้ารหัสใหม่ในชื่อ AES (Advanced Encryption Standard) ซึ่งจะกล่าวถึงในที่นี้ด้วย

Data Encryption Standard

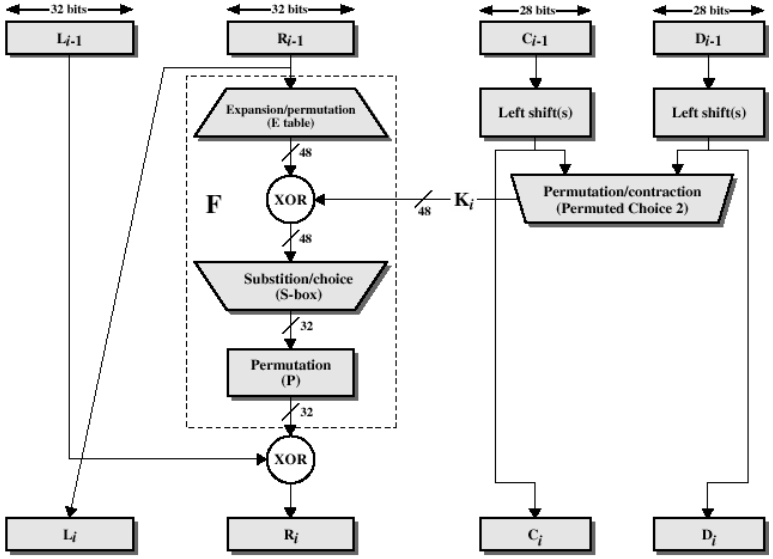
DES เป็นอัลกอริทึมแบบ Block Cipher ที่มีการใช้งานมากที่สุด โดย DES เป็นมาตรฐานของ NIST (National Institute of Standard and Technology) โดยประกาศใช้งานเมื่อปี 1977 โดยใช้ชื่อรหัสว่า FIPS PUB 64 (Federal Information Processing Standard 46) และในปี 1994 ได้ปรับปรุงมาตรฐานเป็น FIPS PUB 46-2 โดยอัลกอริทึมที่ใช้รู้จักกันในชื่อของ DEA (Data Encryption Algorithm)



รูปที่3 Data Encryption Standard Algorithm

อัลกอริทึมการทำงานของ DES แสดงไว้ในรูปที่3 โดย DES จะใช้บล็อกข้อมูลขนาด 64 บิต และใช้คีย์ขนาด 56 บิต โดยหากข้อมูลมีขนาดใหญ่กว่า 64 บิต ก็จะแบ่งออกเป็นบล็อกละ 64 บิต จากรูปทางด้านฝั่งซ้าย จะแสดงการนำบล็อกข้อมูลมากระทำ โดยแบ่งออกเป็น 3 ช่วงย่อย โดยช่วงแรกจะเป็นการนำเอาบล็อกข้อมูลมาผ่านการสลับบิตขั้นต้น (Initial Permutation) ซึ่งจะสลับบิตทั้งหมดเสียใหม่ จากนั้นจะเข้าสู่ช่วงที่ 2 โดยประกอบด้วยการทำฟังก์ชัน Round จำนวน 16 ครั้ง และช่วงสุดท้าย จะประกอบด้วยการสลับกลุ่มข้อมูล 32 บิตซ้ายและขวา จากนั้นก็จะนำมาผ่านการสลับบิตย้อนกลับ (Reverse Initial Permutation) อีกครั้ง ก็จะได้ออกมาเป็น Ciphertext ที่มีความยาวเท่ากับ Plaintext ที่เข้าไป คือ 64 บิต

สำหรับทางด้านฝั่งขวา จะแสดงกระบวนการในการสร้างคีย์ย่อย โดยเริ่มต้นคีย์หลักที่มีความยาว 56 บิต จะผ่านฟังก์ชันการสลับบิต 1 จากนั้นจะนำผลลัพธ์ที่ได้ไปใช้ในการสร้างคีย์ย่อยจำนวน 16 คีย์ โดยในแต่ละครั้งของการสร้างคีย์ย่อยนั้น จะมีการทำ Circular Shift และนำผลลัพธ์ที่ได้ไปผ่านฟังก์ชันการสลับบิต 2 จากทั้งหมดที่ได้กล่าวมา คงจะเห็นภาพรวมของการเข้ารหัสแบบ DES สำหรับรายละเอียดของการเข้ารหัสในแต่ละรอบนั้น ได้แสดงไว้ในรูปที่4



รูปที่4 แต่ละรอบในการทำงานของ DES Algorithm

จากรูป จะเห็นได้ว่าในแต่ละรอบการทำงานนั้น จะมีการแบ่งข้อมูลขนาด 64 บิตที่ได้จากผลของการทำงานในรอบก่อนหน้าออกเป็นข้อมูล 32 บิต 2 ชุด โดยจะเรียกว่าชุด L และชุด R โดยสามารถเขียนเป็นสมการของการสร้างข้อมูลชุด L และ R ในรอบที่ I ได้ดังนี้

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F\,(R_{i-1}, K_1)$$

สำหรับกระบวนการในการถอดรหัส DES ก็จะมีลักษณะเช่นเดียวกับการเข้ารหัสทุกประการ ซึ่งถือเป็นสิ่งสำคัญ โดยจะเป็นอัลกอริทึมเดียวกัน เพียงแต่เปลี่ยนอินพุตเป็น Ciphertext และฟังก์ชันที่สร้างคีย์จะต้องสร้างออกมาในลำดับที่ย้อนกลับกันเท่านั้น ทั้งนี้เนื่องจากการเข้ารหัสนี้โดยภาพรวมแล้ว จะเป็นการสลับบิตข้อมูลไปมา โดยผ่านทางตารางที่มีรูปแบบตายตัว ซึ่งเป็นกระบวนการที่ย้อนกลับได้ และผ่านฟังก์ชัน XOR ซึ่งเป็นกระบวนการที่ย้อนกลับได้เช่นเดียวกัน ดังนั้นการออกแบบอัลกอริทึมที่เหมาะสม ก็จะทำให้การถอดรหัสทำได้ง่าย (หากทราบคีย์) แต่การแกะจะทำได้ยาก เพราะข้อมูลมีการเปลี่ยนไปมาก

The Strength of DES

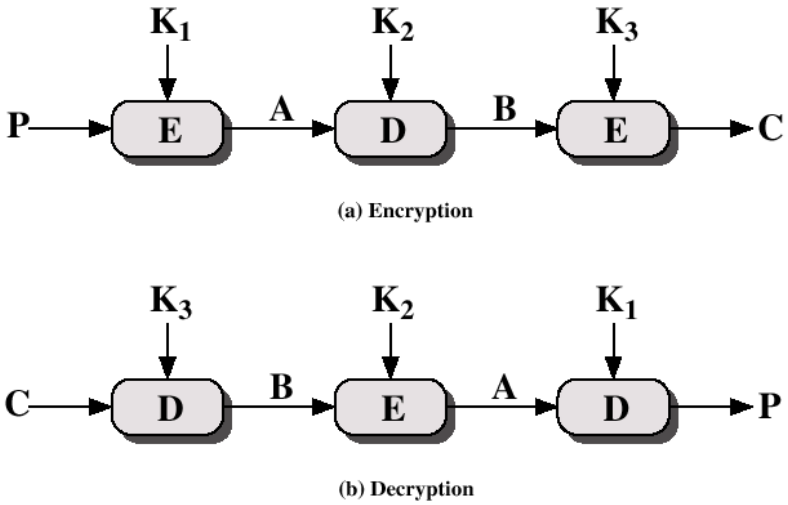
ในการพิจารณาถึงความแข็งแกร่งของ DES นั้น เราจะพิจารณากันใน 2 ด้าน คือ ด้านของตัวอัลกอริทึมเอง และด้านความยาวของคีย์ สำหรับเรื่องของอัลกอริทึมนั้น หลังจากที่ DES ได้ประกาศใช้ออกมาก็ได้มีผู้พยายามจะหาจุดอ่อนของ DES อยู่มาก จนอาจกล่าวได้ว่า DES เป็นอัลกอริทึมการเข้ารหัสที่มีผู้ศึกษาค้นคว้ามากที่สุดในโลกก็ว่าได้ แต่จนถึงบัดนี้ก็ยังไม่มีผู้ที่ค้นหาจุดอ่อนของ DES ได้เลย ดังนั้นจึงอาจถือได้ว่ายังเป็นอัลกอริทึมที่ยังไม่มีจุดอ่อน

แต่จุดที่น่าสนใจมากกว่า คือ ความยาว 56 บิตของ DES เพียงพอหรือไม่ เนื่องจาก DES เกิดมาในช่วงที่คอมพิวเตอร์ยังไม่มีความเร็วมากนัก แต่หลังจากนั้นก็ได้มีการพัฒนาความสามารถของคอมพิวเตอร์อย่างรวดเร็ว ทำให้ความเป็นไปได้ในการแกะคีย์ขนาด 56 บิตมีความเป็นไปได้มากขึ้น ในปี 1998 มีเหตุการณ์หนึ่งที่ต้องบันทึกไว้ และถือได้ว่าเป็นเหตุการณ์ที่ทำให้อัลกอริทึม DES ถึงจุดจบอย่างเป็นทางการ เหตุการณ์ที่ว่านั้นเกิดจากหน่วยงานหนึ่งที่ชื่อว่า EFF (Electronic Frontier Foundation) ได้สร้างเครื่องคอมพิวเตอร์ขึ้นมาเครื่องหนึ่งเพื่อทำหน้าที่ในการแกะรหัส DES โดยเฉพาะ โดยใช้ชื่อว่า “DES Cracker” โดยใช้เงินทุนไม่ถึง 250,000 เหรียญ โดยสามารถแกะคีย์ขนาด 56 บิตได้ในเวลา 3 วันเท่านั้น ยิ่งไปกว่านั้น EFF ได้เผยแพร่ผลงานของตนเองสู่สาธารณะ ทำให้ทุกคนสามารถสร้างได้

อย่างไรก็ตาม ในการแกะรหัสนั้นจะเริ่มจากการใส่ Ciphertext เข้าไปจากนั้นก็ใส่ Key เข้าไปทีละตัวอย่าง ซึ่งจะทำให้เกิดเป็น Plaintext ออกมา แต่ Plaintext จะเป็น Plaintext ที่ถูกต้อง คือ ตรงกับต้นฉบับก็ต่อเมื่อ Key ที่ใส่เข้าไปเป็นคีย์ที่ถูกต้อง คราวนี้จะรู้ได้อย่างไรว่า Plaintext ที่ได้เป็น Plaintext ที่ถูกต้อง ซึ่งหมายถึง Key ที่ถูกต้องด้วย นั่นก็คือ การพิจารณาดูเนื้อความ เช่น หากต้นฉบับเป็นภาษาอังกฤษ Plaintext ที่ถูกต้องก็ต้องเป็นภาษาอังกฤษด้วย ดังนั้นก็จะใช้วิธีดูไปเรื่อย ๆ ว่าหากผลลัพธ์ที่แกะได้ออกมาเป็นภาษาที่อ่านได้ ก็หมายความว่า Key ที่ใช้เป็น Key ที่ถูกต้องแล้ว แต่หากไฟล์ที่เข้ารหัสเป็นไฟล์แบบอื่น เช่น ไฟล์ที่มีตัวเลขอย่างเดียว หรือ ไฟล์ที่ผ่านการบีบข้อมูลมาแล้ว ก็จะยิ่งหา Key ที่ถูกต้องได้ยากยิ่งขึ้น หรืออาจทำไม่ได้เลยก็ได้

Triple DES

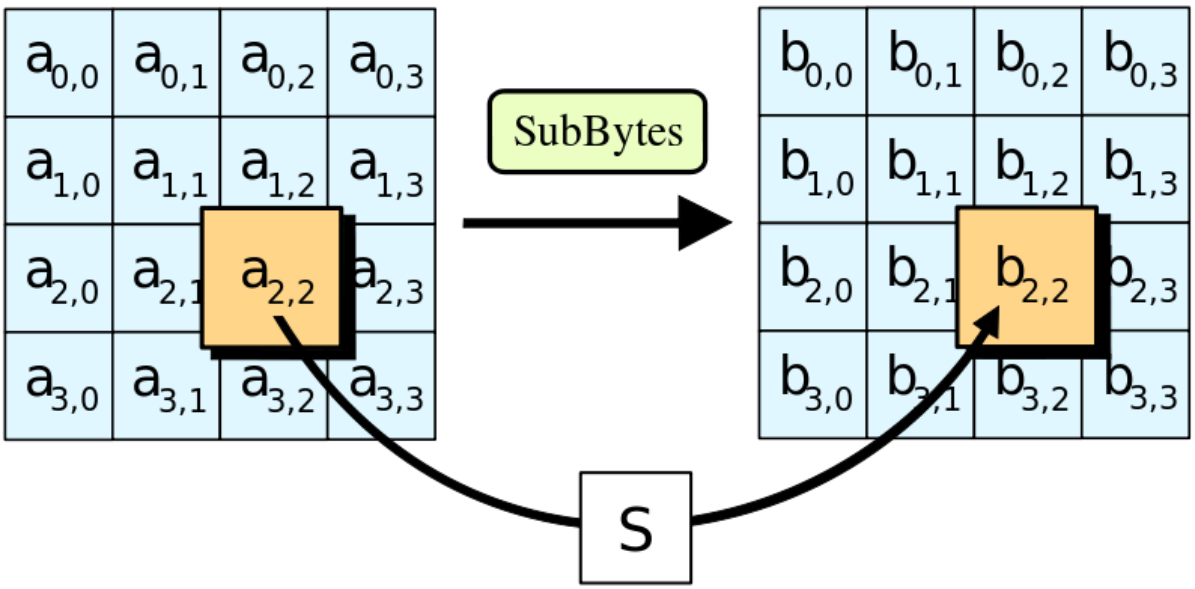
อัลกอริทึม 3DES ได้รับการเสนอครั้งแรกโดย Tuchman โดยเริ่มแรกเป็นมาตรฐานของ ANSI หมายเลข X9.17 ในปี 1985 จากนั้น NIST ได้นำมาเป็นส่วนหนึ่งของมาตรฐานหมายเลข FIPS PUB 46-3 ในปี 1999 โดย 3DES จะใช้อัลกอริทึมเดียวกับ DES แต่จะใช้คีย์จำนวน 3 คีย์และทำ DES จำนวน 3 ครั้ง ดังรูปที่5



รูปที่5 Triple DES Algorithm

ดังนั้นคีย์ทั้งหมดจะมีความยาวเท่ากับ 168 บิต อย่างไรก็ตาม FIPS PUB 46-3 ยอมให้ใช้คีย์เพียง 2 คีย์ คือ กำหนดให้คีย์ K1 เท่ากับ K3 ได้ ดังนั้นความยาวคีย์จะเหลือเท่ากับ 112 บิต กล่าวโดยสรุป 3DES เป็นการปรับปรุง DES ให้มีความปลอดภัยมากขึ้น สามารถใช้งานร่วมกับ DES ได้ อย่างไรก็ตามการทำ DES จำนวน 3 ครั้ง ถือได้ว่ามีความปลอดภัยมากพอสำหรับช่วงเวลาที่จะมีการพัฒนาอัลกอริทึมในการเข้ารหัสตัวต่อไป นั่นก็คือ AES

Advanced Encryption Standard



รูปที่6 AES Algorithm

แม้ว่า 3DES เป็นอัลกอริทึมที่มีความปลอดภัย เพราะใช้คีย์ที่มีความยาวถึง 168 บิต ทำให้ยากต่อการแกะรหัส และใช้ อัลกอริทึมเดียวกับ DES ซึ่งรู้จักกันทั่วไป ดังนั้นจึงถือว่ามีความปลอดภัยมากที่สุดในปัจจุบัน ซึ่งจนถึงปัจจุบันก็ยังไม่มีใครแกะ3DES ได้สำเร็จ ดังนั้นหากจะพิจารณาในด้านความปลอดภัยเพียงอย่างเดียว 3DES ถือเป็นตัวเลือกที่เหมาะสม แต่เนื่องจากรูปแบบอัลกอริทึมเป็นอัลกอริทึมที่ออกแบบมาให้ง่ายต่อการสร้างด้วยฮาร์ดแวร์มากกว่าซอฟต์แวร์ ประกอบกับการทำ DES ถึง 3 ครั้ง ทำให้การเข้ารหัสข้อมูลจำนวนมาก ๆ มีความล่าช้า อีกประการหนึ่งคือ บล็อกข้อมูลขนาด 64 บิต ถือว่าเล็กเกินไปหน่อยในปัจจุบัน

ด้วยเหตุดังกล่าวในระยะยาว จึงมีความต้องการอัลกอริทึมใหม่ ที่มีความเหมาะสมมากขึ้น โดยในปี 1997 ได้มีการประกาศให้นักพัฒนาเสนออัลกอริทึมเข้ามาให้ NIST พิจารณา โดยได้ตั้งชื่ออัลกอริทึมใหม่นี้ว่า AES ซึ่งมีข้อกำหนดเบื้องต้นว่า จะต้องมีความปลอดภัยอย่างน้อยเท่ากับ 3DES มีการใช้บล็อกข้อมูลขนาด 128 บิต และสามารถใช้ความยาวคีย์ได้ตั้งแต่ 128 บิต 192 บิต และ 256 บิต โดยมีความเร็วในการทำงานที่ดี และใช้หน่วยความจำในการทำงานน้อย โดยในรอบแรกมีผู้เสนอเข้ามา 15 อัลกอริทึม และได้คัดเหลือ 5 อัลกอริทึมในรอบที่ 2 คือ MARS, RC6™, Rijndael, Serpent,Twofish และล่าสุดในเดือนตุลาคม 2543 ที่ผ่านมานี่เอง ทาง NIST ก็ได้ประกาศผู้ชนะ ออกมา

อัลกอริทึม AES ที่ได้รับการตัดสินให้ชนะเลิศนี้ สร้างขึ้นโดย Joan Daemen และ Vincent Rijmen ซึ่งเป็นนักวิจัยชาวเบลเยี่ยม โดยอัลกอริทึมนี้เดิมทีใช้ชื่อว่า Rijndael (Rijmen & Daemen) อัลกอริทึมนี้จะยังคงเป็นแบบ Block Cipher โดยใช้บล็อกข้อมูลขนาด 128 บิต 196 บิต และ 256 บิต โดยสามารถใช้คีย์ได้ยาวถึง 128 บิต 196 บิต และ 256 บิต โดยอัลกอริทึมนี้ได้รับการออกแบบให้มีการทำงานที่เหมาะสมกับโพรเซสเซอร์รุ่นใหม่ ๆ และสามารถใช้งานกับ Smart Card ได้ เพราะใช้หน่วยความจำน้อย

อัลกอริทึม Rijndael จะใช้ฟังก์ชัน Round ที่สามารถเลือกได้ว่าจะทำ 10,12 หรือ 14 ครั้ง โดยมีการทำงานอยู่ 4 การทำงานย่อย คือ Byte Sub ก็คือการใช้ S-Boxes ในการสลับข้อมูลระหว่าง 2 บล็อก ShiftRow คือการสลับข้อมูลระหว่างแถว Mix Column คือการ Shift ข้อมูลในแต่ละ Column และสุดท้าย คือ Key Addition คือการนำมาบวกกับคีย์ ซึ่งการทำงานทั้งหมด เป็นการทำงานที่ง่าย มีจำนวนครั้งของการทำงานน้อย ทำงานได้เร็ว และใช้หน่วยความจำน้อย

Other Symmetric Block Cipher

ที่ผ่านมาเราได้ศึกษาอัลกอริทึมในการเข้ารหัสของ NIST เพราะเป็นองค์กรหลักที่ทำหน้าที่ด้านนี้ แต่นอกจาก NIST แล้วยังมีผู้อื่นที่สร้างอัลกอริทึมในการเข้ารหัสในแบบ Block Cipher ขึ้นมาเช่นกัน โดยลักษณะของอัลกอริทึมส่วนใหญ่ จะยังคงเป็นแบบที่ใช้โครงสร้างของ Feistel ทั้งนี้เนื่องจากโครงสร้างนี้เป็นโครงสร้างที่รู้จักกันดีอยู่แล้ว ทำให้เมื่อต้องการวิเคราะห์การเข้ารหัส สามารถทำได้ง่าย เพราะหากมีการใช้โครงสร้างที่ต่างไปโดยสิ้นเชิง อาจมีจุดอ่อนของโครงสร้าง ซึ่งอาจไปปรากฏในภายหลังได้

IDEA

อัลกอริทึม IDEA (International Data Encryption Algorithm) เป็น Symmetric Block Cipher พัฒนาโดย Xuejia Lai และ James Massey แห่ง Swiss Federal Institute of Technology ในปี 1991 อัลกอริทึม IDEA ใช้คีย์ขนาด 128 บิต ซึ่งจะต่างจาก DES ทั้งในส่วนของฟังก์ชัน Round และส่วนของฟังก์ชันที่ใช้ในการสร้างคีย์ย่อย

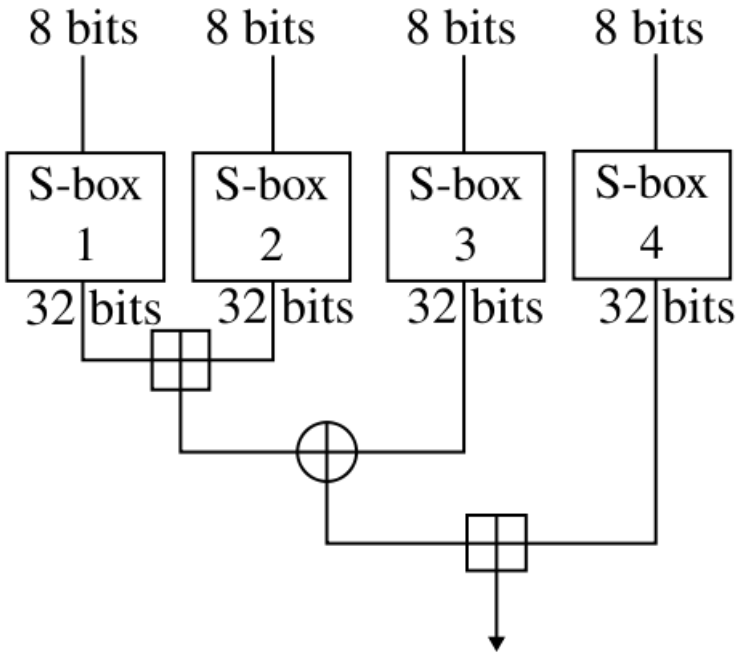
ในส่วนของฟังก์ชัน Round ใน IDEA จะไม่ใช้ S-Boxes โดยจะใช้ฟังก์ชันคณิตศาสตร์ XOR การบวก และการคูณประกอบกัน ซึ่งเป็นผลให้ฟังก์ชันดังกล่าว มีความซับซ้อนมาก ทำให้ยากต่อการวิเคราะห์ ในส่วนของคีย์ย่อยจะใช้ Circular Shift ในการสร้างคีย์ย่อย โดยจะมีการประมวลผลจำนวน 8 ครั้งด้วยกัน และเนื่องจาก IDEA เป็นอัลกอริทึมแรก ๆ ที่ใช้คีย์ขนาด 128 บิต เพื่อใช้แทน DES จึงไม่ได้รับความสนใจมากนัก และได้รับการต่อต้านจากนักวิเคราะห์ค่อนข้างมาก IDEA มีการใช้งานใน PGP (เป็นตัวเลือกหนึ่ง) และมีการใช้ในผลิตภัณฑ์ต่าง ๆ หลายตัว

Algorithm	Key Size	Number of Rounds	Mathematical Operations	Application
DES	56 bits	16	XOR, fixed S-boxes	SET, Kerberos

Triple DES	112 or 168 bits	48	XOR, fixed S-boxes	Financial key management, PGP, S/MIME
IDEA	128 bits	8	XOR, addition, Multiplication	PGP
Blowfish	Variable to 448 bits	16	XOR, variable S-boxes, addition	
RC5	Variable to 2048 bits	Variable to 255	Addition, subtraction, XOR, rotation	
CAST-128	40 to 128 bits	16	Addition, subtraction, XOR, rotation, fixed S-boxes	PGP

ตารางที่2 เปรียบเทียบ Algorithm ต่างๆ

Blowfish



รูปที่7 Blowfish Algorithm

อัลกอริทึม Blowfish พัฒนาโดย Bruce Schneier ในปี 1993 โดยเขาเป็นเป็นที่ปรึกษาอิสระและผู้เชี่ยวชาญด้านการเข้ารหัส ซึ่ง Blowfish ได้รับการต้อนรับอย่างดี ในฐานะตัวเลือกหนึ่งของ DES อัลกอริทึม Blowfish ได้รับการออกแบบมาเพื่อให้สร้างได้ง่าย และมีความเร็วในการทำงานสูง เป็นอัลกอริทึมที่ใช้พื้นที่ในการทำงานน้อยมาก เพียง 5K ก็สามารถทำงานได้ สิ่งที่น่าสนใจใน Blowfish คือ การใช้คีย์ที่มีการเปลี่ยนค่าความยาวได้ โดยยาวได้มากถึง 448 บิต แต่ในทางปฏิบัติมักใช้กันที่ 128 บิตก็ถือว่าเพียงพอแล้ว ในอัลกอริทึมนี้มีการวนรอบทั้งหมด 16 ครั้ง

อัลกอริทึม Blowfish มีการใช้ S-Boxes และ XOR เช่นเดียวกับ DES แต่ยังมีการใช้การบวกร่วมด้วย แต่สิ่งที่ต่างจาก DES คือ ใน DES จะใช้ S-Boxes แบบความยาวคงที่ แต่ใน Blowfish จะสามารถเปลี่ยนค่าความยาวได้ สำหรับส่วนของการสร้างคีย์ย่อยนั้น Blowfish มีการนำเอาอัลกอริทึม Blowfish เองมาใช้ในการสร้างคีย์ย่อยและ S-Boxes ด้วย โดยจะต้องมีการวนทั้งหมด 521 รอบในการทำงานเพื่อสร้างคีย์ย่อยและ S-Boxes และนั่นทำให้ Blowfish ไม่เหมาะที่จะใช้กับงานที่จะต้องมีการเปลี่ยนค่าSecret Key บ่อย ๆ

อัลกอริทึม Blowfish เป็นหนึ่งในอัลกอริทึมการเข้ารหัสที่มีความปลอดภัยสูง เนื่องจากการสร้างส่วนของคีย์ย่อย และ S-Boxes สร้างขึ้นมาจากอัลกอริทึม Blowfish เอง ซึ่งทำให้การแกะรหัสจะทำได้ยากมาก ปัจจุบันมีการใช้งานในผลิตภัณฑ์บางตัว

RSA

การเข้ารหัสแบบ RSA เป็นอัลกอลิธึมที่ถูกอธิบายเมื่อพ.ศ. 2520 โดย รอน ริเวสต์ (Ron Rivest) , อาดี ชามีร์ (Adi Shamir) และเล็น เอเดิลแมน (Len Adleman) ที่ MIT โดยที่ RSA นั้นเป็นตัวย่อมาจากนามสกุลของทั้ง 3 คน (Rivest-Shamir-Adleman) การเข้ารหัสแบบ RSA ได้จดสิทธิบัตรโดยสถาบัน MIT ในสหรัฐอเมริกาเมื่อปี พ.ศ. 2526 และได้สิ้นสุดลงเมื่อปี พ.ศ. 2543 เพราะเป็นผลงานที่เคยถูกตีพิมพ์เผยแพร่แล้วก่อนที่จะจดสิทธิบัตร

การสร้าง public key

• ขั้นที่ 1 สุ่มจำนวนเฉพาะ

ให้ p และ q เป็น 11 และ 23 ตามลำดับ

• ขั้นที่ 2 หาค่า n

n = pq

n = 11(23) = 253

• ขั้นที่ 3 หาค่า totient

• ขั้นที่ 4 เลือกค่า e

ให้ e = 3

• ขั้นที่ 5 หาค่า d

จะได้ d = 147

ตัวอย่างการเข้ารหัส

ให้ m = 97 (สมมุติว่าเปลี่ยนมาจากข้อความ M แล้ว)

นำไปคำนวณในสมการ

จะได้ c = 102

เพราะฉะนั้นจะได้ว่าข้อความที่เข้ารหัสแล้วจะมีค่าเป็นตัวเลขคือ 102

การถอดรหัส

การถอดรหัสจะมีขั้นตอนคล้ายกับการเข้ารหัส แต่จะมีการนำค่า d ที่หาไว้มาใช้ เมื่อได้ค่า d, e, n มาแล้ว ให้นำค่าเหล่านี้มาถอดรหัสได้ดังต่อไปนี้

- ขั้นที่ 1

นำค่า c ที่เป็นตัวเลขที่เข้ารหัสแล้วมาคำนวณในสมการ

โดยที่ m เป็นตัวเลขที่ยังไม่ได้เข้ารหัส

- ขั้นที่ 2

เมื่อได้ตัวเลข m มาแล้วให้นำไปเปลี่ยนเป็นตัวอักษร M เพื่อให้ได้ข้อความที่ยังไม่ได้เข้ารหัส (plain text)

ตัวอย่างการถอดรหัส

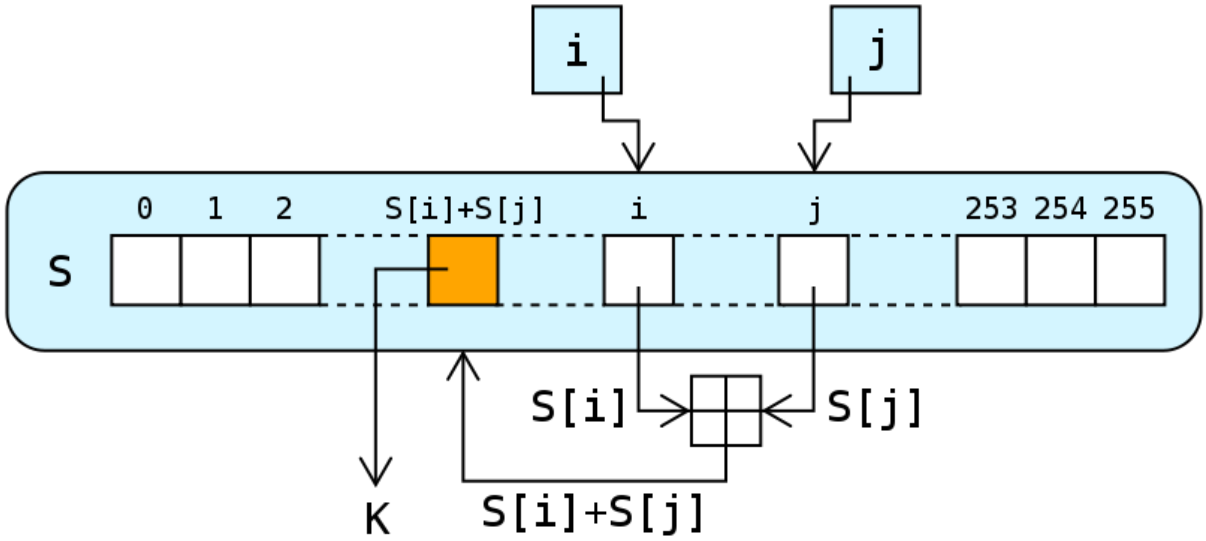
จากตัวอย่างการเข้ารหัสที่ได้กล่าวไว้แล้ว เราจะนำค่าตัวเลขที่เข้ารหัสไว้แล้วและค่า d, e, n ที่คำนวณไว้แล้วมาถอดรหัสได้ด้วยวิธีการดังต่อไปนี้

นำค่า c=102 มาคำนวณในสมการ

จะได้ m=97

เมื่อถอดรหัสแล้วจะได้ค่า m =97 ซึ่งตรงกับค่าตัวเลขที่ยังไม่ได้เข้ารหัส

RC4

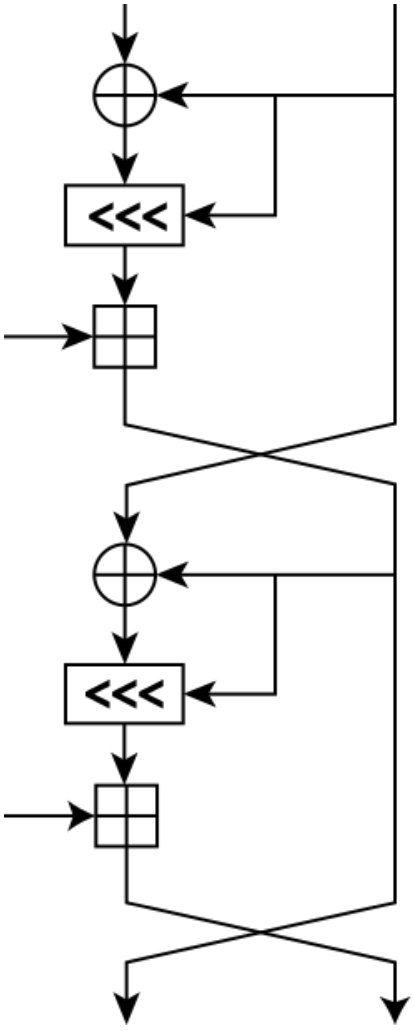


รูปที่8 RC4 Algorithm

เป็นกรรมวิธีในการเข้ารหัสที่ใช้กุญแจรหัสขนาด 40 บิต ซึ่งถือว่าน้อยเกินไปสำหรับระบบรักษาความมั่นคงปลอดภัยสารสนเทศ (มีฮาร์ดแวร์ที่ใช้ในการถอดรหัสแบบ RC4 ได้ภายในเวลา 5 ช.ม.) ดังนั้นแล้ว RC4 จึงถูกนำมาใช้แค่เพียงในบางกรณีเท่านั้น เช่น ถ้ามีกฎหมายหรือกฎข้อบังคับว่าห้ามใช้กุญแจรหัสที่มีความยาวเกินไป เป็นต้น ถือว่าไม่เป็นการดีเท่าไร ที่การเข้ารหัสแบบ RC4 ขนาด 40 บิตนี้ ถูกนำมาใช้จริงในหลายๆ ระบบ รวมถึงระบบการเข้ารหัสบนเครือข่ายไร้สายประเภท WEP ตามมาตรฐาน IEEE 802.11 ด้วย ซึ่งในความเป็นจริงสำหรับเครือข่ายไร้สายแล้ว ก็ยังมีการเข้ารหัสในรูปแบบอื่นที่มีความปลอดภัยกว่า เช่น WPA2

เป็นต้น ผู้ใช้จึงควรระมัดระวัง และไม่ควรเลือกใช้การเข้ารหัสแบบ WEP ถ้าสามารถเลือกระบบการเข้ารหัสแบบอื่นได้

RC5



รูปที่9 RC5 Algorithm

อัลกอริทึม RC5 พัฒนาขึ้นมาโดย Ron Rivest แห่ง RSA ซึ่งเป็นหนึ่งในผู้ร่วมค้นคิดอัลกอริทึม RSA อันเลื่องชื่อ (เป็นตัว R ใน RSA) RC5 พัฒนาขึ้นมาในปี 1994 โดยได้รับประกาศเป็นมาตรฐานใน RFC 2040 โดยมีลักษณะเด่นของการออกแบบดังนี้

- มีความเหมาะสมต่อการสร้างทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์ เพราะใช้เพียงการทำงานพื้นฐานเท่านั้น

- มีความเร็วสูง โดย RC5 มีการทำงานเป็น Word Oriented และมีอัลกอริทึมที่ง่าย

- สามารถปรับปรุงในเข้ากับโพรเซสเซอร์ที่มีขนาด Word ความยาวต่าง ๆ เพราะ RC5 สามารถปรับความยาวได้

- สามารถปรับจำนวนรอบการทำงานได้

- สามารถใช้คีย์ที่มีความยาวต่าง ๆ ได้

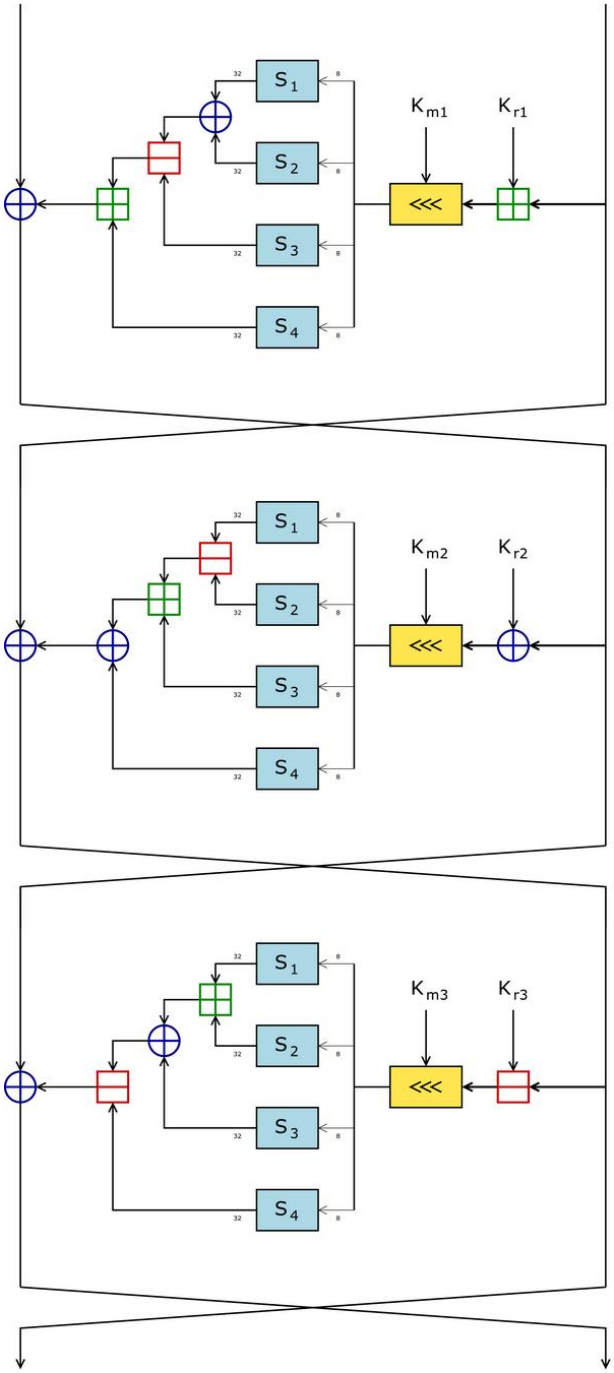
- มีโครงสร้างที่ง่าย ทำให้การวิเคราะห์หาจุดอ่อนสามารถทำได้ง่าย

- ต้องการหน่วยความจำในการทำงานน้อย

- มีความปลอดภัยสูง หากมีการกำหนดค่าการทำงานที่เพียงพอ

RC5 มีการใช้งานในผลิตภัณฑ์ของ RSA Data Security, Inc.

CAST-128



รูปที่10 CAST-128 Algorithm

อัลกอริทึม CAST ได้รับการพัฒนาขึ้นโดย Carlisle Adams และ Stafford Tavares แห่ง Entrust Techonology ในปี 1997 โดยอัลกอริทึมนี้ เป็นส่วนหนึ่งของ CAST Project โดยประกาศเป็นมาตรฐานใน RFC 2144 โดยมีขนาดของคีย์ตั้งแต่ 40 จนถึง 128 บิต โดยเพิ่มทีละ 8 บิต อัลกอริทึม CAST เป็นผลจากการวิจัยและพัฒนาอย่างยาวนาน และได้รับการวิเคราะห์อย่างกว้างขวางก่อนนำมาใช้งาน โดยมีการใช้งานในหลายผลิตภัณฑ์ ซึ่งรวมทั้ง PGP ด้วย

อัลกอริทึม CAST ในส่วนของการสร้างคีย์ย่อย มีการใช้ S-Boxes ที่มีความยาวคงที่ แต่มีความยาวมากกว่า DES โดยใช้ฟังก์ชันแบบ Non-Linear ทำให้การแกะรหัสทำได้ยาก นอกจากนั้นในส่วนของฟังก์ชัน Round ก็ต่างจากอัลกอริทึมอื่นด้วย เพราะใน CAST ในแต่ละรอบของการทำงาน จะใช้ฟังก์ชัน Round ที่แตกต่างกันไป ซึ่งยิ่งทำให้การแกะสามารถทำได้ยากมาก

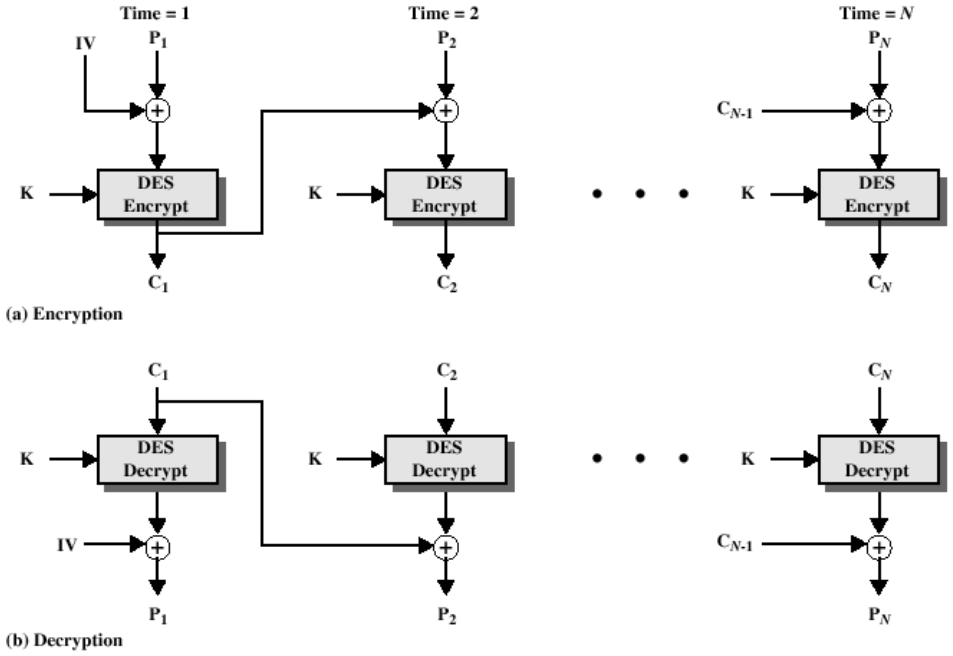
Cipher Block Modes of Operation

ในการเข้ารหัสแบบ Symmetric Block Cipher นั้น จะมีการประมวลผลข้อมูลเป็นชุด เช่น ใน DES และ 3DES นั้นจะประมวลผลข้อมูลครั้งละ 64 บิต ดังนั้นในกรณีที่ข้อมูลมีขนาดยาวมากกว่านั้น ก็มีความจำเป็นต้องแบ่งข้อมูลออกเป็นชุด ๆ แล้วประมวลผลทีละชุดไปเรื่อย ๆ จนหมดข้อมูล (ในกรณีที่ข้อมูลชุดสุดท้ายเหลือไม่ถึง 64 บิต ก็จะมีการเติมให้ครบ 64 บิต) ในการประมวลผลข้อมูลเป็นชุด ๆ นี้ การทำงานในแบบที่ง่ายที่สุด จะมีชื่อว่า Electronic Codebook หรือ ECB ซึ่งหลักการทำงานก็คือ จะแบ่งข้อมูลออกเป็นบล็อกละ 64 บิต โดยในการเข้ารหัสข้อมูลแต่ละชุด จะใช้คีย์เดียวกันทั้งหมด

การทำงานในแบบ ECB นี้ แม้ว่าจะเป็นการทำงานที่ง่าย แต่จะมีข้อเสีย คือ ในกรณีที่บล็อกข้อมูล 64 บิตเป็นข้อมูลเดียวกัน จะทำให้เกิดเป็น Ciphertext ที่มีลักษณะเหมือนกัน ดังนั้นกรณีที่ข้อมูลมีลักษณะเป็นโครงสร้างมาก ๆ เช่น ข้อมูลในลักษณะที่เป็น Record หรือข้อมูลที่มีลักษณะซ้ำกันบ่อย ๆ การทำงานในแบบ ECB อาจไม่ปลอดภัยเพียงพอ ดังนั้นในข้อมูลลักษณะดังกล่าว จึงควรใช้โหมดการทำงานอื่น

Cipher Block Chaining Mode

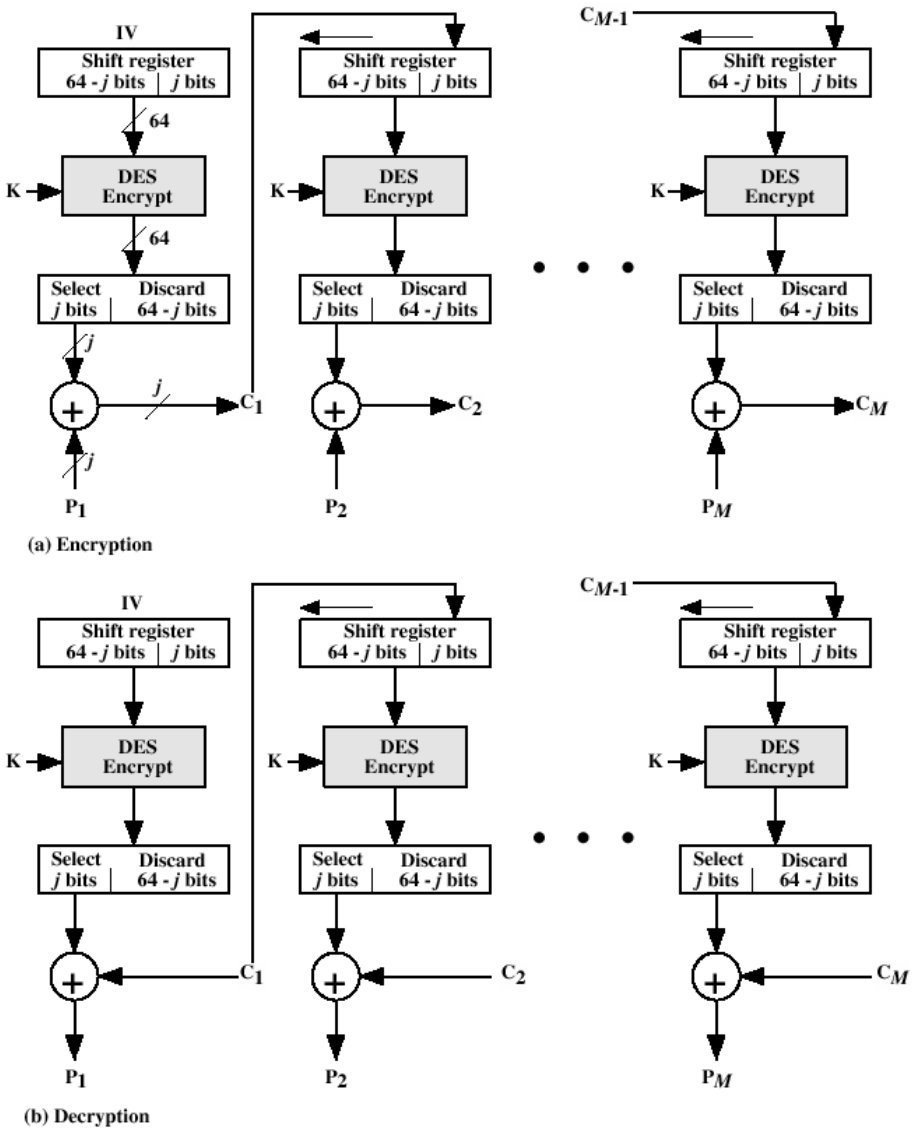
การทำงานในแบบ Cipher Block Chaining หรือ CBC แสดงในรูปที่11 โดยอินพุตที่จะป้อนเข้าสู่อัลกอริทึมการเข้ารหัสนั้น จะเกิดจากผลลัพธ์ของการทำ XOR ระหว่าง Plaintext Block ปัจจุบันกับ Ciphertext Block จากการทำงานครั้งก่อนหน้า โดยใช้คีย์เดียวกัน ในทุกการเข้ารหัส ซึ่งจะเป็นผลให้อินพุตถูกเปลี่ยนก่อนที่จะเข้าสู่กระบวนการเข้ารหัส โดยการเปลี่ยนนี้จะเกิดขึ้นในลักษณะของลูกโซ่ไปเรื่อย ๆ ซึ่งทำให้ผลลัพธ์ของการเข้ารหัสไม่มีส่วนสัมพันธ์โดยตรงกับข้อมูลก่อนเข้ารหัส



รูปที่11 การทำงานในแบบ CBC Mode

สำหรับการถอดรหัส ก็จะใช้กระบวนการที่กลับกันดังแสดงในรูปที่11b สำหรับ IV ที่แสดงในภาพนั้น หมายถึง Initialize Vector ซึ่งเป็นชุดข้อมูลที่จะใช้เฉพาะการ XOR กับข้อมูลในบล็อกแรกเท่านั้น โดยข้อมูล IV นี้จะต้องตรงกันทั้งผู้รับและผู้ส่ง เพราะในการถอดรหัสข้อมูล จำเป็นจะต้องใช้ IV มา XOR กับผลลัพธ์ของการถอดรหัส เพื่อให้ได้ข้อมูลเดิมกลับมา ดังนั้นในการทำงานจริงข้อมูล IV อาจเป็นข้อมูลที่ผู้ส่งกำหนดขึ้นเสมือนกับเป็นคีย์อย่างหนึ่ง หรืออาจเป็นข้อมูลที่ผู้ส่งสร้างขึ้นมาจากการสุ่ม จากนั้นก็ส่งให้ฝั่งรับโดยใช้วิธี ECB ในครั้งหนึ่งก่อน จากนั้นจึงค่อยใช้ IV ในการทำงานในโหมด CBC ต่อไป สำหรับข้อมูล IV นี้ควรจะมีการเปลี่ยนแปลงทุกครั้งที่มีการส่งข้อมูล 1 ชุด เพราะหากฝ่ายตรงข้ามได้ข้อมูล IV นี้ไปได้ ก็จะสามารถทำกระบวนการถอดรหัส CBC ได้เช่นเดียวกัน

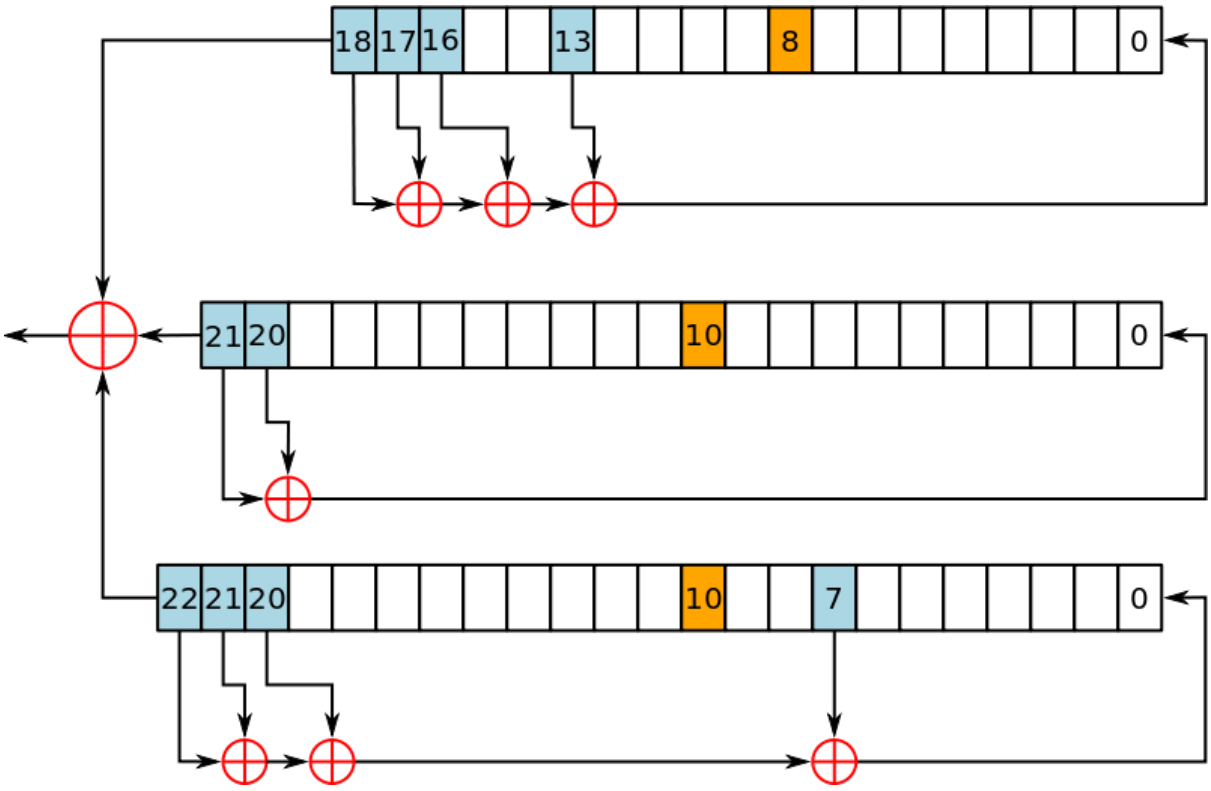
Cipher Feedback Mode



รูปที่12 การทำงานในแบบ CFB Mode

สำหรับการทำงานในแบบ CFB นี้ จะมีการทำงานในรูปแบบของ Streme Cipher โดยหากต้องการให้ทำงานในแบบ Byte Oriented จะเลือก j=8 ซึ่งเมื่อมีข้อมูลเข้ามา 1 ไบต์ จะสามารถประมวลผล และสามารถส่งข้อมูลที่เข้ารหัสไปได้ในทันที แต่การทำงานในโหมดนี้ จะมีการทำงานล่าช้า เพราะการทำงานครั้งละ 64 บิต แต่มีการนำไปใช้งานเพียง 8 บิตเท่านั้น

Stream Cipher



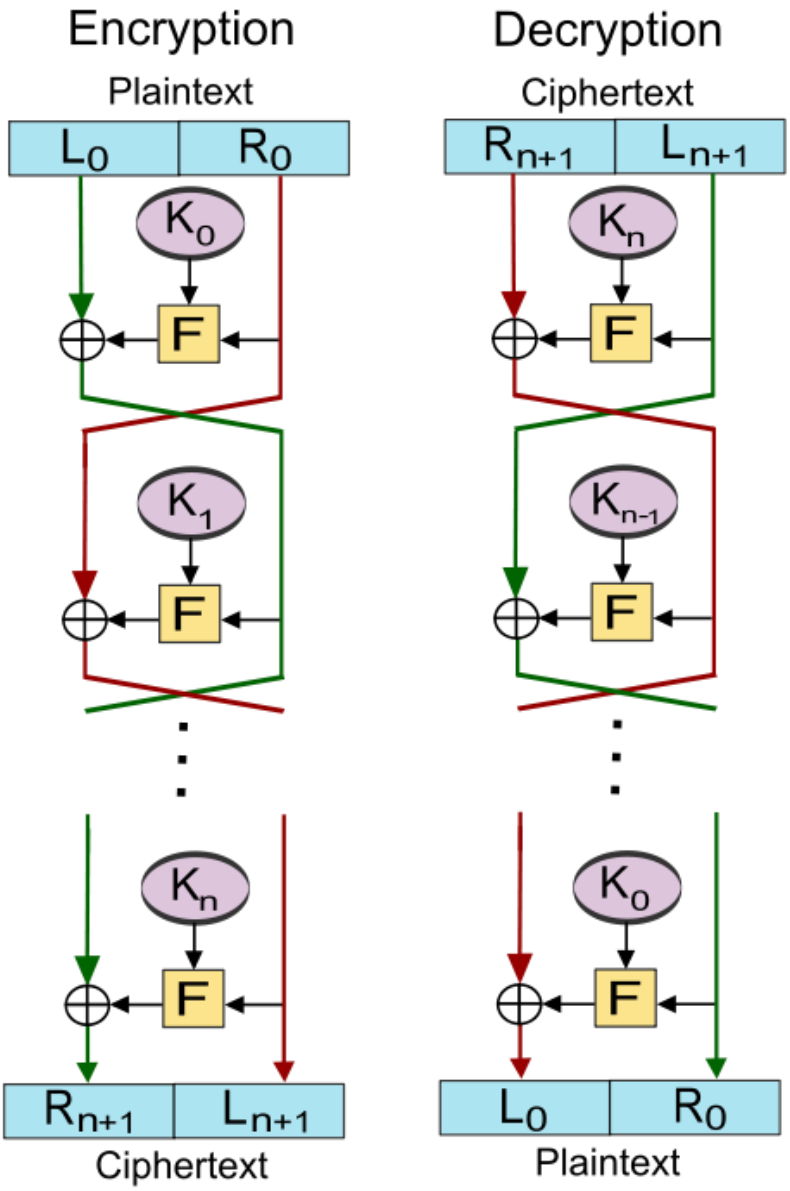
รูปที่13 Stream Cipher Algorithm

ในโลกความเป็นจริง การเข้ารหัสแบบ One Time Pad นั้นนับว่าเป็นกระบวนการที่ลำบาก โดยเฉพาะในแง่ของความเร็ว ที่ข้อมูลสำหรับการเข้ารหัสจะมีขนาดใหญ่เท่าๆ กับตัวข้อมูลเอง ในแง่ของการลดภาระในการส่งกุญแจที่ต้องมีขนาดเท่ากับตัวข้อมูล จึงมีการสร้างระบบเข้ารหัสที่เรียกว่า stream cipher ขึ้นมา

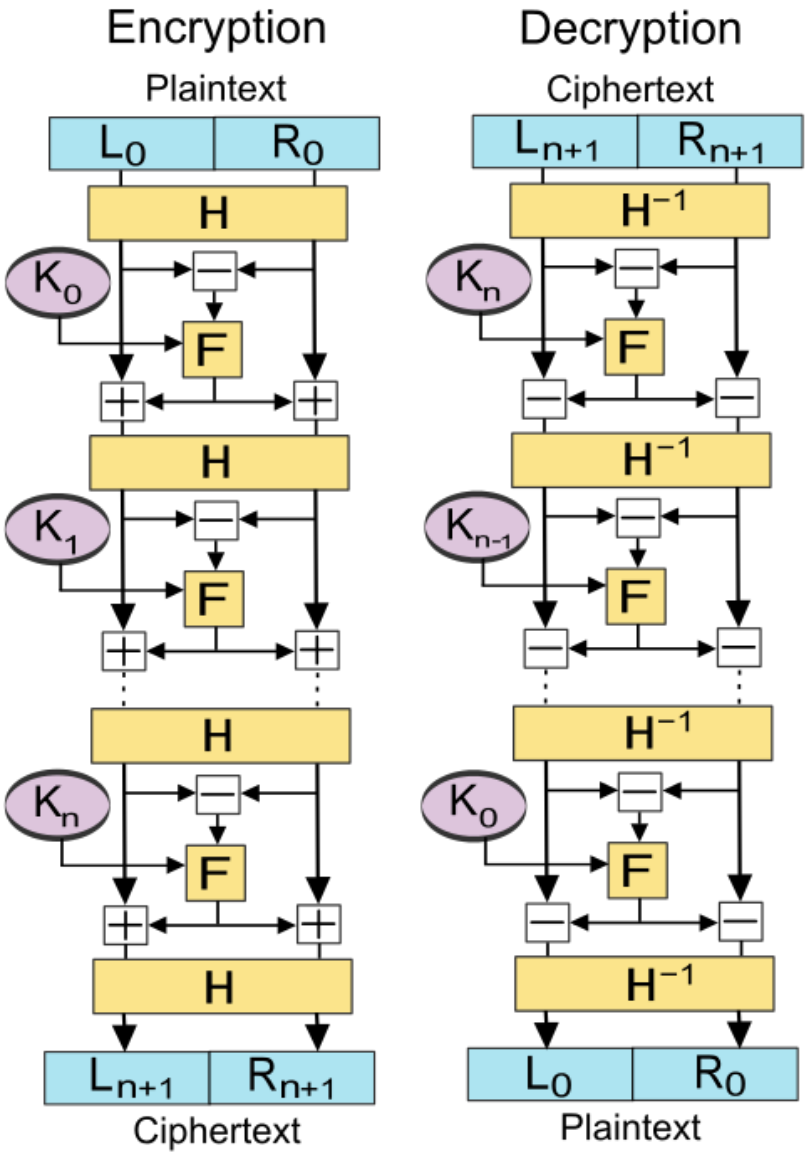
stream cipher โดยหลักการแล้วมันคือการใช้ค่าจากฟังก์ชั่น pseudo random number generator (PRNG) เพื่อสร้างข้อมูลไบนารีขนาดยาวขึ้นมา โดยค่าเริ่มต้นของ PRNG นั้นเป็นค่าความลับที่ผู้รับและผู้ส่งแชร์ระหว่างกัน ข้อมูลนั้นมักมีขนาดเล็ก แต่สามารถสร้างไบนารีแบบสุ่มที่มีขนาดใหญ่ขึ้นมาเพื่อ XOR กับข้อมูลทั้งหมดที่ต้องการเข้ารหัสได้ ตัวอย่างการใช้งานในรูปแบบนี้ได้แก่การเข้ารหัสแบบ A5/1 ในโทรศัพท์ GSM ที่ใช้ฟังก์ชั่นในตระกูล LFSR ที่เป็นฟังก์ชั่น PRNG ตัวหนึ่ง หรือใน WEP ของระบบแลนไร้สายที่ใช้ฟังก์ชั่น RC4 เป็นตัวสร้างไบนารีเช่นกัน

การเข้ารหัสแบบ SSL RC4_128 เป็นการเข้ารหัสแบบแบบ stream cipher ที่เว็บขนาดใหญ่ล้วนใช้งานจากความได้เปรียบด้านความเร็ว และการส่งข้อมูลที่เป็นชุดที่วิ่งอยู่บนโปรโตคอล TCP ที่ช่วยลดปัญหาข้อมูลสูญหายระหว่างทางไปแล้ว

Block Cipher



รูปที่14 Feistel ciphers Algorithm



รูปที่15 Lai-Massey ciphersAlgorithm

การทำ stream cipher แม้จะแก้ปัญหากุญแจเข้ารหัสมีขนาดใหญ่ไปได้ แต่การที่กุญแจเข้ารหัส ซึ่งกลายเป็นค่าเริ่มต้นสำหรับ PRNG นั้นมีขนาดเล็ก ทำให้ความปลอดภัยโดยรวมนั้นลดลงอย่างหลีกเลี่ยงไม่ได้ เช่นหากค่าเริ่มต้นของฟังก์ชั่น PRNG ความเป็นไปได้เพียง 2^32 ค่า (ขนาดกุญแจมีขนาด 32 บิต) ผู้ดักฟังข้อมูลไปได้ ก็อาจจะไล่ค่าที่เป็นไปได้เรื่อยๆ แล้วดูว่าค่าใดที่ให้ข้อมูลไบนารีแบบสุ่มแล้วนำไป XOR กับข้อมูลเข้ารหัสแล้วได้ค่าที่มีความหมายแสดงว่าได้เจอกุญแจที่ถูกต้องแล้ว แม้การขยายขนาดกุญแจไปเรื่อยๆ ในทุกวันนี้หากข้อมูลที่ส่งไประหว่างทางมีการตรวจสอบความถูกต้อง กุญที่ถอดรหัสแล้วสามารถตรวจสอบความถูกต้องได้ก็มีความเป็นไปได้สูงว่าจะมีเพียงกุญแจเดียว จากกุญแจที่เป็นไปได้จำนวนมหาศาล กระบวนการเช่นนี้ทำให้ความปลอดภัยโดยรวมของ stream cipher นั้นต่ำกว่า one time pad อยู่ ในโลกความเป็นจริง

การเข้ารหัสแม้จะเป็น stream cipher มักจะใช้กุญแจขนาดใหญ่กว่า 128 บิตขึ้นไปเพื่อรับประกันได้ว่าจะไม่มีใครสามารถไล่กุญแจทุกกุญแจที่เป็นไปได้ภายในช่วงเวลาหลายสิบปีข้างหน้า

Stream cipher สามารถใช้เข้ารหัสข้อมูลขนาดใหญ่ได้ แต่การใช้งานทุกวันนี้ เรามักใช้งานเข้ารหัสกับข้อมูลที่ต้องส่งผ่านเครือข่าย เป็นไปได้ที่ข้อมูลอาจจะเสียหายหรือสูญหายระหว่างทาง การใช้ข้อมูลที่ไม่ต้องส่งข้อมูลไปบนเครือข่ายเองบางครั้งก็ต้องการเข้าถึง “ตรงกลาง” ของข้อมูล การเข้าถึงข้อมูลที่เข้ารหัสเป็น stream ขนาดใหญ่ ทำให้กระบวนการถอดรหัสทำได้ช้า เพราะต้องสร้างข้อมูลไบนารีจากจุดเริ่มต้นไปจนถึงจุดจุดที่เราต้องการอ่านข้อมูล จึงจะสามารถถอดรหัสออกมาได้

สำหรับข้อมูลที่ต้องการใช้งานเป็นส่วนๆ ได้ จึงมีการเข้ารหัสแบบ block cipher มาอีกแบบหนึ่งให้เลือกใช้งาน ในตระกูล block cipher นั้นอัลกอริทึมที่ได้รับความนิยมสูงตัวแรกๆ คือ DES หรือ Data Encryption Standard ที่ถูกออกแบบโดย Horst Feisel ที่ทำงานอยู่ในไอบีเอ็มช่วงปี 1977 และได้รับบรรจุเป็นกระบวนการเข้ารหัสมาตรฐานสำหรับรัฐบาลกลางสหรัฐฯ นับแต่นั้นมา ตัวอัลกอริทึม DES นั้นออกแบบให้เข้ารหัสข้อมูลทีละ 64 บิต ด้วยกุญแจขนาด 56 บิต โดยตัวฟังก์ชั่นเองยังไม่สามารถพิสูจน์ได้ว่ามีความปลอดภัยจริงหรือไม่ แต่อายุกว่าสามสิบปีของ DES ก็ยังมีความปลอดภัยตามที่ออกแบบเอาไว้ นั่นคือผู้ที่จะแฮกได้ต้องทดสอบกุญแจถึง 2^56 รูปแบบเพื่อหากุญแจที่ถูกต้องในการถอดรหัส DES นับเป็นตัวอย่างหนึ่งของฟังก์ชั่นความปลอดภัยที่ปลอดภัยจากประสบการณ์ที่ผ่านมา (empirically secure)

กุญแจ 2^6 ที่เคยใหญ่พอในยุคของช่วงปี 1970 ซึ่งคอมพิวเตอร์ยังทำงานได้ช้ากว่าทุกวันนี้มาก ทำให้หลายคนยังเชื่อในประสิทธิภาพของฟังก์ชั่น DES เอง แล้วพัฒนามันด้วยการเข้ารหัสซ้ำๆ ไปเป็นจำนวนสามรอบ กลายเป็นฟังก์ชั่น 3DES หรือ Triple DES ที่มีขนาดกุญแจ 168 บิต ทุกวันนี้กระบวนการจ่ายเงินผ่านบัตรเครดิตสมาร์ตการ์ดต่างๆ ยังคงใช้การเข้ารหัสแบบ 3DES เป็นส่วนประกอบสำคัญในการรักษาความปลอดภัย

แต่ระหว่างที่ DES ถูกพิสูจน์ว่าไม่ปลอดภัยนั้น สถาบันมาตรฐานและเทคโนโลยีของสหรัฐฯ (National Institute of Standards and Technology - NIST) ก็จัดประกวดมาตรฐานการเข้ารหัส และอัลกอริทึมของ Joan Daemen และ Vincent Rijmen นักคณิตศาสตร์ชาวเบลเยียมก็ชนะการประกวดด้วยอัลกอริทึมที่ปรับปรุงจากอัลกอริทึมเข้ารหัสของตัวเองที่ชื่อว่า Rijndael จนกระทั่งชนะการประกวดและเวอร์ชั่นที่พัฒนาแล้วของ Rijndael ก็กลายเป็นอัลกอริทึม AES ในทุกวันนี้ ในช่วงสิบปีมานี้ มีแนวโน้มที่ระบบการเข้ารหัสใหม่ๆ จะใช้ AES มาเป็นอัลกอริทึมหลักมากขึ้นเรื่อยๆ กระบวนการเข้ารหัส AES ถูกบรรจุเข้าไปในมาตรฐาน เช่น SSL, WPA เพื่อรับประกันว่าจะทนทานต่อการเจาะรหัส และเนื่องจากความนิยมที่เพิ่มขึ้นซีพียูรุ่นใหม่ๆ เริ่มมีชุดคำสั่งเร่งการทำงานของ AES เพิ่มเข้ามา ความนิยมของการเข้ารหัสแบบ AES จึงน่าจะได้รับความนิยมสูงขึ้นเรื่อยๆ ในอนาคต

Block Cipher Operation Mode

กระบวนการเข้ารหัสเป็นบล็อคนั้นมีปัญหาเช่นเดียวกับการเข้ารหัสแบบอื่นๆ คือการใช้กุญแจเดิมๆ ซ้ำๆ นั้นจะให้ทำให้ได้ข้อมูลแบบเดิมออกมาเรื่อยๆ แม้ว่าฟังก์ชั่นการเข้ารหัสแบบใหม่ๆ จะทำให้การวิเคราะห์ว่าข้อมูลที่แท้จริงเป็นอะไรนั้นทำได้ยาก แต่หลายครั้งการที่แฮกเกอร์เห็นว่ามีข้อมูลแบบเดิม (ที่ไม่รู้ว่าเป็นอะไร) เดินทางซ้ำๆ บ่อยเพียงใดก็สามารถเปิดเผยข้อมูลบางอย่างได้แล้ว กระบวนการเข้ารหัสเป็นบล็อคที่พื้นฐานที่สุดนั้น เราเรียกว่าโหมด Electronic Code Book (ECB) การทำงานของมันคือการแบ่งข้อมูลออกเป็นบล็อคๆ ตามอัลกอริทึมที่เลือกใช้ เช่น 128 บิตสำหรับ AES หรือ 64 บิตสำหรับ DES แล้วเข้ารหัสด้วยกุญแจที่กำหนด

กระบวนการนี้แม้จะมองไม่ออกว่าข้อมูลเป็นอะไร แต่เมื่อมองข้อมูลในภาพรวมแล้วเราก็อาจจะเห็นได้ว่าข้อมูลที่แท้จริงคืออะไร กระบวนการแบบ ECB จึงไม่แนะนำนัก

เพื่อให้ข้อมูลทุกบล็อคแตกต่างกันออกไปอย่างแท้จริง มีการเสนอกระบวนการ Cipher Block Chaining (CBC) และ Cipher Feedback (CFB) ขึ้นมา โดยหลักการแล้วทั้งสองแบบคล้ายกันคือใช้ข้อมูลที่เข้ารหัสแล้วของบล็อคก่อนหน้าเพื่อเป็นส่วนประกอบของกุญแจเข้ารหัสของบล็อคถัดไป กระบวนการเช่นนี้ทำให้แม้จะส่งข้อมูลเดิมซ้ำๆ ไปเรื่อยๆ ข้อมูลที่ถอดรหัสแล้วก็จะเปลี่ยนไปไม่รู้จบ

ปัญหาของทั้ง CBC และ CFB คือหากมีบล็อคใดบล็อคหนึ่งเสียหายไประหว่างทาง ฝั่งถอดรหัสจะไม่สามารถถอดรหัสบล็อคถัดไปได้อีกด้วย ทำให้ข้อมูลเสียหายไปพร้อมกันสองบล็อค แต่ใช้บล็อคถัดไปที่ได้รับถูกต้องแต่ไม่สามารถถอดรหัสได้เพื่อไปคำนวณกุญแจสำหรับอีกสองบล็อคถัดจากบล็อคที่เสียหายได้ ข้อจำกัดอีกประการคือ CBC และ CFB นั้นไม่สามารถเข้ารหัสแบบขนานได้สะดวก เพราะแต่ละบล็อคต้องรอบล็อกก่อนหน้าเสมอ ทำให้การกระจายงานในซีพียูหลายคอร์ทุกวันนี้ทำได้ยาก

เพื่อให้กระบวนการเข้ารหัสสามารถทำไปขนานกันได้ทั้งการเข้ารหัสและการถอดรหัส รวมถึงไม่ต้องเสียข้อมูลมากเกินความจำเป็นในกรณีที่มีข้อมูลเสียหาย โหมดการเข้ารหัสแบบ Counter (CTR) อาศัยการติดตัวเลขที่เปิดเผย หรือเรียกว่า nonce ไปกับข้อมูล ตัวเลขนี้จะเปลี่ยนไปเรื่อยๆ ในแต่ละบล็อค เมื่อนำค่า nonce มารวมกับกุญแจลับเพื่อถอดรหัสแล้ว การส่งข้อมูลเดิมๆ ก็จะเปลี่ยนไปเรื่อยๆ ตาม nonce

กระบวนการเข้ารหัสแบบ CTR ถูกดัดแปลงไปใช้งานในมาตรฐานเข้ารหัส WEP ที่ตัวเลขขนาด 24 บิตเรียกว่า initial vector (IV) จะถูกส่งติดไปกับทุกแพ็กเก็ต เพื่อใช้ประกอบกับกุญแจลับในการเข้ารหัส แต่ตัวเลขขนาด 24 บิตนี้นับว่ามีขนาดเล็กมาก และสิ่งที่เกิดขึ้นคือการ์ดแลนไร้สายบางรุ่นพยายามใช้ค่า IV ในแบบสุ่ม แต่การสุ่มที่ดีกลับทำให้ความน่าจะเป็นที่จะใช้ค่า IV นี้ซ้ำกันเกิดขึ้นได้ง่ายในทุกๆ 4096 แพ็กเก็ตเท่านั้น (ปัญหาสุ่มเลขซ้ำนี้เรียกว่า birthday paradox ที่ระบุว่าในกลุ่มคนเพียง 57 คนจะมีวันเกิดวันเดือนเดียวกันด้วยความน่าจะเป็น 0.99) เมื่อเกิดการใช้ค่า IV ซ้ำกัน ค่าไบนารีที่ได้จาก RC4 ก็จะเหมือนกันทุกประการ และแฮกเกอร์ก็จะสามารถคาดเดาได้ว่าข้อมูลภายในเป็นอะไรเมื่อพบข้อมูลที่ใช้ค่า IV เดิมซ้ำกันมากขึ้นเรื่อยๆ แบบเดียวกับการใช้ One Time Pad ซ้ำๆ กัน