บทที่ 3

บล็อคไซเฟอร์และสตรีมไซเฟอร์

คำนำบท

ในบทนี้จะกล่าวถึงประเภทของอัลกอริธึ่มในการเข้ารหัสซึ่งแบ่งเป็นบล็อกไซเฟอร์อัลกอริธึ่มและสตรีมไซเฟอร์อัลกอริธึ่มและอธิบายถึงข้อแตกต่างของทั้งสองอัลกอริธึ่ม นอกจากนี้ยังได้กล่าวถึงเทคนิคของโหมดที่ใช้ในการเข้ารหัสทั้ง 4 โหมด คือ ECB, CBC , CFB และ OFB โดยจะอธิบายหลักการทำงานและข้อดีข้อเสียของแต่ละแบบพอสังเขปเพื่อให้ผู้ใช้มีความเข้าใจในการเข้ารหัสแต่ละแบบในแต่ะละสถานการณ์ และภาวะการใช้งานได้อย่างถูกต้อง

คริปโตกราฟฟิกอัลกอริธึ่ม [Meyer and Matyas,1982]

ปัญหาพื้นฐานในการทำการเข้ารหัสข้อมูลคือการที่จะทำอย่างไรให้การแปลงเพลนเท็กซ์ (Plaintext) ไปเป็นไซเฟอร์เท็กซ์ (Ciphertext)โดยที่จะไม่ถูกแปลงกลับได้โดยง่ายโดยผู้ที่ไม่รู้คีย์ (Key) หนึ่งในวิธีการแปลงนั้นก็คือ การใช้โค้ดซิสเท็ม (Code System) วิธีการก็คือจะมีโค้ดบุ๊ค (Code Book) หรือคู่มือการแปลงข้อความอยู่โดยเพลนเท็กซ์ที่ถูกส่งเข้ามาทำการแปลงจะถูกแปลงโดยใช้โค้ดบุ๊คนี้ทำให้ได้ไซเฟอร์เท็กซ์ออกมาจะเห็นว่าคู่ข้อมูลของเพลนเท็กซ์และไซเฟอร์เท็กซ์จะถูกจำกัดโดยขนาดของโค้ดบุ๊คนี้อีกวิธีการหนึ่งเรียกว่า ไซเฟอร์ซิสเท็ม (Cipher System) ซึ่งวิธีการนี้ในการใช้งานจะต้องประกอบด้วยคริปโตกราฟฟิกอัลกอริธึ่ม (Cryptographic Algorithm) และคริปโตกราฟฟิกคีย์ (Cryptographic Key)คริปโตกราฟฟิกอัลกอริธึ่มยังแบ่งออกได้เป็น 2 ประเภท คือ บล็อกไซเฟอร์อัลกอริธึ่ม (Block Cipher Algorithm) คือการที่ข้อมูลถูกเข้ารหัสและ ถอดรหัส เป็นบล็อกของข้อมูลซึ่งขนาดของบล็อกของข้อมูลถูกกำหนดไว้ล่วงหน้าแล้วโดยผู้สร้างอัลกอริธึ่มและอีกประการหนึ่งคือ สตรีมไซเฟอร์อัลกอริธึ่ม (Stream Cipher Algorithm)คืออัลกอริธึ่มที่ผู้ใช้สามารถกำหนดขนาดของข้อมูลที่ทำการเข้ารหัสและถอดรหัสได้เราสามารถจะใช้ทั้งบล็อกไซเฟอร์อัลกอริธึ่มและสตรีมไซเฟอร์อัลกอริธึ่มในการทำให้เกิดโหมดของการเข้ารหัสแบบต่างๆได้โดยอาศัยวิธีการฟีดแบ็ค (Feedback) หรือ เชนนิ่ง(Chaining) ซึ่งก็คือการเอาข้อมูลที่

การเอาข้อมูลที่เกิดขึ้นในอดีตมาเป็นตัวกำหนดข้อมูลในปัจจุบันซึ่งนอกจากจะทำให้เกิดความปลอดภัยของข้อมูลเพิ่มมากขึ้นแล้วยังสามารถใช้ในการทำการออ เทนธิเคชันได้ด้วย

ในกรณีของโค้ดบุ๊คจะเห็นว่าการแปลงเพลนเท็กซ์เป็นไซเฟอร์เท็กซ์จะถูกกำหนดและจำกัดโดยขนาดของโค้ดบุ๊คโดยถ้าโค้ดบุ๊คมีขนาดเล็กความปลอดภัยของข้อมูลก็จะน้อยเราจะสามารถมองได้ว่าคริปโตกราฟฟิกอัลกอริธึ่มคือ โค้ดบุ๊คที่มีขนาดใหญ่มากและมีรูปแบบการแปลงเพลนเท็กซ์เป็นไซเฟอร์ได้หลายรูปแบบมากโดยในแต่ละรูปแบบจะถูกกำหนดโดยคีย์ซึ่งรูปแบบของการแปลงจากเพลนเท็กซ์ไปเป็นไซเฟอร์เท็กซ์ก็จะมีน้อยซึ่งการแปลงข้อมูลจากเพลนเท็กซ์เป็นไซเฟอร์เท็กซ์นี้เรียกว่าการเข้ารหัส (Encryption) โดยในการแปลงแต่ละครั้งจะต้องสามารถทำการแปลงกลับจากไซเฟอร์เท็กซ์เป็นเพลนเท็กซ์ได้ด้วยคีย์เดียวกันซึ่งเรียกการทำเช่นนี้ว่าการถอดรหัส (Decryption)นอกจากคริปโตกราฟฟิกอัลกอริธึ่มสามารถแบ่งออกเป็นบล็อกไซเฟอร์และสตรีมไซเฟอร์แล้วเรายังสามารถแบ่งตามลักษณะของการใช้คีย์ได้ด้วยคือการเป็นไพรเวทคีย์อัลกอริธึ่มและพับบลิคคีย์อัลกอริธึ่มโดยไพรเวทคีย์อัลกอริธึ่มคืออัลกอริธึ่มที่คีย์ที่ใช้ในการทำการเข้ารหัสหรือถอดรหัสเป็นคีย์เดียวกันส่วนพับบลิคคีย์อัลกอริธึ่มคืออัลกอริธึ่มที่อนุญาตให้ใครก็ได้ในข่ายสื่อสารนั้นสามารถทำการเข้ารหัสข้อมูลส่งมาให้เราผ่านทางข่ายสื่อสารสาธารณะโดยใช้พับลิคคีย์ซึ่งเป็นคีย์ของเราซึ่งเปิดเผยต่อสาธารณชนดังนั้นทุกคนในข่ายสื่อสารนี้ก็สามารถจะรับไซเฟอร์เท็กซ์ชุดนี้ได้แต่จะมีเพียงเราเท่านั้นที่สามารถจะถอดรหัสได้โดยใช้คีย์อีกตัวหนึ่งซึ่งเป็นซิเคร็ทคีย์ (Secret Key) หรือคีย์ที่เป็นความลับ

<u>บล็อกไซเฟอร์</u> (Block Cipher)

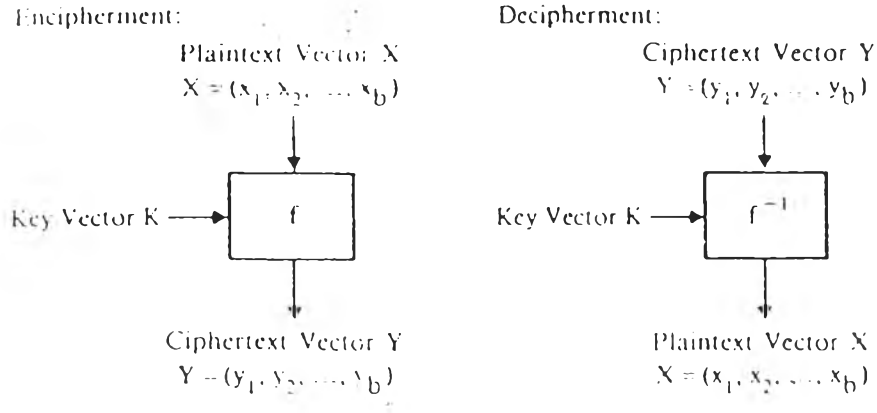
บล็อกไซเฟอร์จะแปลงกลุ่มข้อมูลอินพุทที่มีขนาดจำกัดจำนวนหนึ่งหรืออินพุทบล็อก(InputBlock)ไปเป็นกลุ่มของข้อมูลเอาท์พุทที่มีขนาดเท่ากันหรือเอาท์พุทบล็อก(Output Block) ฟังก์ชันที่ใช้ในการแปลงนี้จะต้องมีคุณสมบัติที่ทำให้ทุกๆบิทของเอาท์พุทบล็อกขึ้นอยู่กับอินพุทบล็อกและคีย์ขนาดของบล็อก หรือ บล็อกไซซ์ (Blocksize) จะถูกกำหนดขึ้นโดยพิจารณาถึงความสามารถในการรักษาความลับของข้อมูลไม่ควรให้บล็อกไซซ์มีขนาดใหญ่เกินไปเพราะจะทำให้ยากแก่การนำมาใช้งานจริงเนื่องจากรีจิสเตอร์ที่ต้องใช้เก็บข้อมูลในแต่ละบล็อกจะต้องมีขนาดใหญ่ไปด้วยแต่ถ้าให้เล็กเกินไปความสามารถในการรักษาความปลอดภัยก็จะน้อยเนื่องจากฝ่ายตรงข้าม(Opponent)ก็จะสามารถใช้วิธีที่เรียกว่าเมสเสจเอ็กซ์ซอสชัน(Message

Exhaustion)คือการที่มีข้อมูลของเพลนเท็กซ์และไซเฟอร์เท็กซ์ที่เป็นไปได้ทั้งหมดแล้วนำมาเปรียบเทียบกับไซเฟอร์เท็กซ์ที่รับมาได้ทำให้ได้รู้เพลนเท็กซ์จะเห็นว่าถ้าบล็อกไซซ์มีขนาดใหญ่ข้อมูลเพลนเท็กซ์-ไซเฟอร์เท็กซ์หรือดิคชันนารีนี้ก็จะมีขนาดใหญ่ด้วยทำให้ฝ่ายตรงข้ามอาจจะเห็นว่าไม่คุ้มที่พยายามเปิดเผย ความลับโดยวิธีนี้

อีกวิธีการหนึ่งในการพยายามเปิดเผยความลับที่เราจะต้องพิจารณาถึงเพื่อเป็นข้อมูลในการกำหนดขนาดของบล็อกไซซ์คือวิธีการวิเคราะห์ความถี่ในการเกิดบล็อก(Block Frequency Analysis) ถ้าบล็อกไซซ์มีขนาดเล็กความถี่ในการเกิดบล็อกซ้ำ ๆ กันก็จะสูง ซึ่งฝ่ายตรงข้ามเก็บข้อมูลตรงส่วนนี้ไว้เป็นสถิติก็จะทำให้สามารถเปิดเผยความลับได้

<u>คอนเวนชันแนลอัลกอริธึ่ม (Conventional Algorithms)</u>

เราจะพิจารณาถึงพื้นฐานบล็อกไซเฟอร์ที่ถูกออกแบบให้มีลักษณะเดียวกับDESอัลกอริธึ่มการเข้ารหัสและถอดรหัสจะเป็นไปตามรูป 3.1



รูป 3.1 คอนเวนชันแนลอัลกอริธึ่มของบล็อคไซเฟอร์ [Meyer and Matyas,1982]

จากรูป 3.1 เราสามารถเขียนเป็นสมการทางคณิตศาสตร์ได้ดังนี้คือ

$$f_K(X) = Y$$

$$f_K^{-1}(Y) = X$$

สำหรับการเข้ารหัสและถอดรหัสตามลำดับโดยที่

X คือ เพลนเท็กซ์

Y คือ ไซเฟอร์เท็กซ์

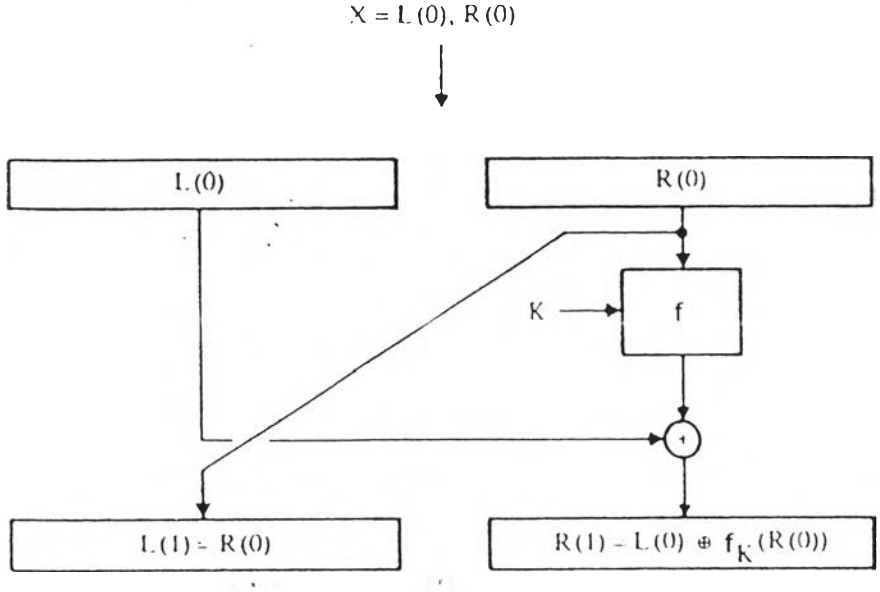
K คือ คีย์

f คือ ฟังก์ชันที่ใช้ในการเข้ารหัส

จากรูป 3.1 เราสมมติให้เพลนเท็กซ์ X ประกอบด้วย b บิท ซึ่งถูกแยกออกเป็น 2 บล็อก L(0) และ R(0) แต่ละบล็อกมีขนาด b/2 บิท เราเขียนในรูปของสมการได้ดังนี้

$$X = L(0), R(0)$$

ฟังก์ชัน f จะแปลง R(0) ไปเป็น $f_K(R(0))$ โดยคีย์ K ดังรูป 3.2



รูป 3.2 การแปลงของบล็อคข้อมูลขาเข้า L(0) และ R(0)
[Meyer and Matyas,1982]

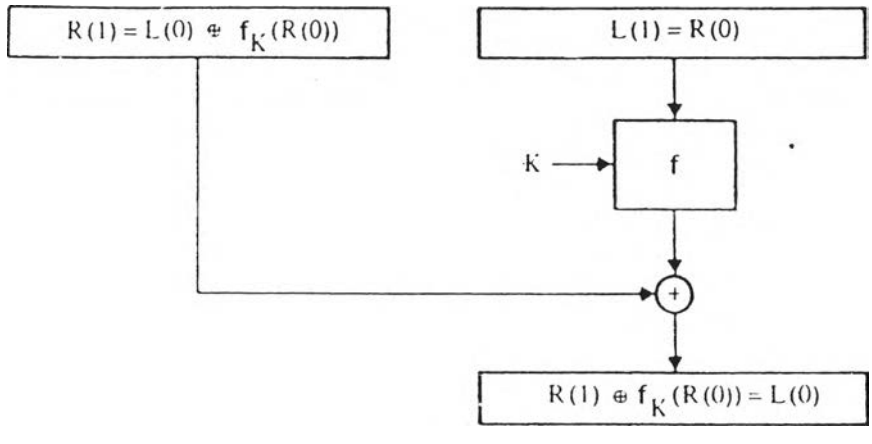
จากรูป 3.2

$$R(1) = L(0) \oplus f_K(R(0))$$

และเมื่อกำหนดให้ L(1) = R(0) แล้วก็จะถือว่าการเข้ารหัสเสร็จเรียบร้อย ณ จุดนี้ ถ้า L(1) และ R(1) คือไซเฟอร์เท็กซ์ของ L(0) และ R(0) ตามลำดับและจะมีวิธีอย่างไรหรือไม่ที่จะถอดรหัส L(1) และ R(1) นี้ให้เป็นเพลนเท็กซ์ L(0) และ R(0) โดยไม่ใช้ฟังก์ชันย้อนกลับของ f_k ทั้งนี้เพราะจะได้สะดวกในการใช้ฟังก์ชันเดียวกันทั้งการเข้ารหัสและถอดรหัส

เพื่อให้ได้ตามจุดประสงค์นี้เราลองพิจารณาดูไซเฟอร์เท็กซ์L(1)และR(0)เพราะว่า L(1) เท่ากับR(0)ดังนั้นจะเห็นได้ว่าครึ่งหนึ่งของเพลนเท็กซ์ถูกถอดรหัสออกมาแล้วโดยที่ยังไม่ได้ทำอะไรเลย เพลนเท็กซ์ที่เหลืออีกครึ่งหนึ่งคือ L(0) ก็สามารถหาออกมาได้ โดยการสร้าง $f_K(R(0))$ ขึ้นมาแล้วบวกแบบโมดูโล-2 เข้ากับ R1 ดังรูปที่ 3.3

$$R(1) \oplus f(R(0)) = [L(0) \oplus f(R(0))] \oplus f(R(0)) \qquad (3.5)$$



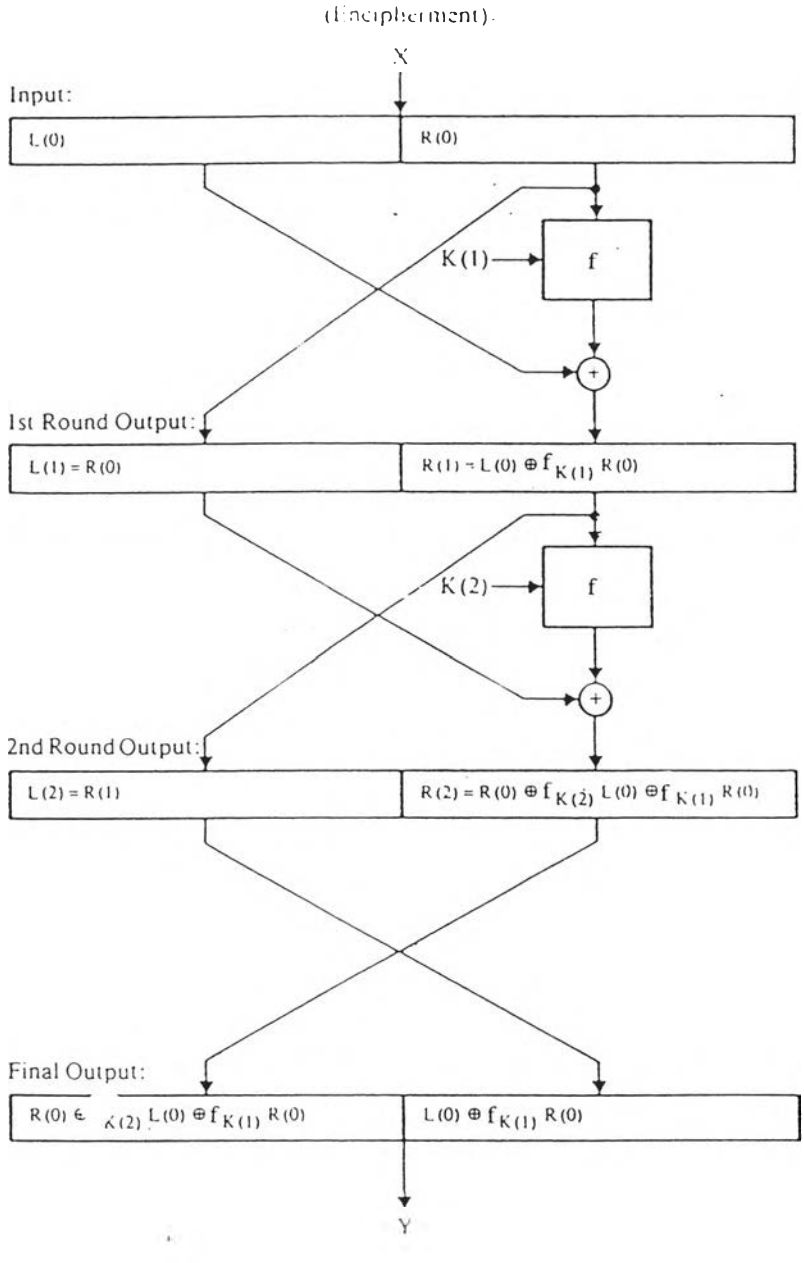
รูป 3.3 การแปลงเพื่อให้ได้ L(0) [Meyer and Matyas,1982]

จะเห็นว่าเราไม่จำเป็นต้องใช้ฟังก์ชันย้อนกลับของ f_k ก็จะสามารถถอดรหัสไซเฟอร์เท็กซ์

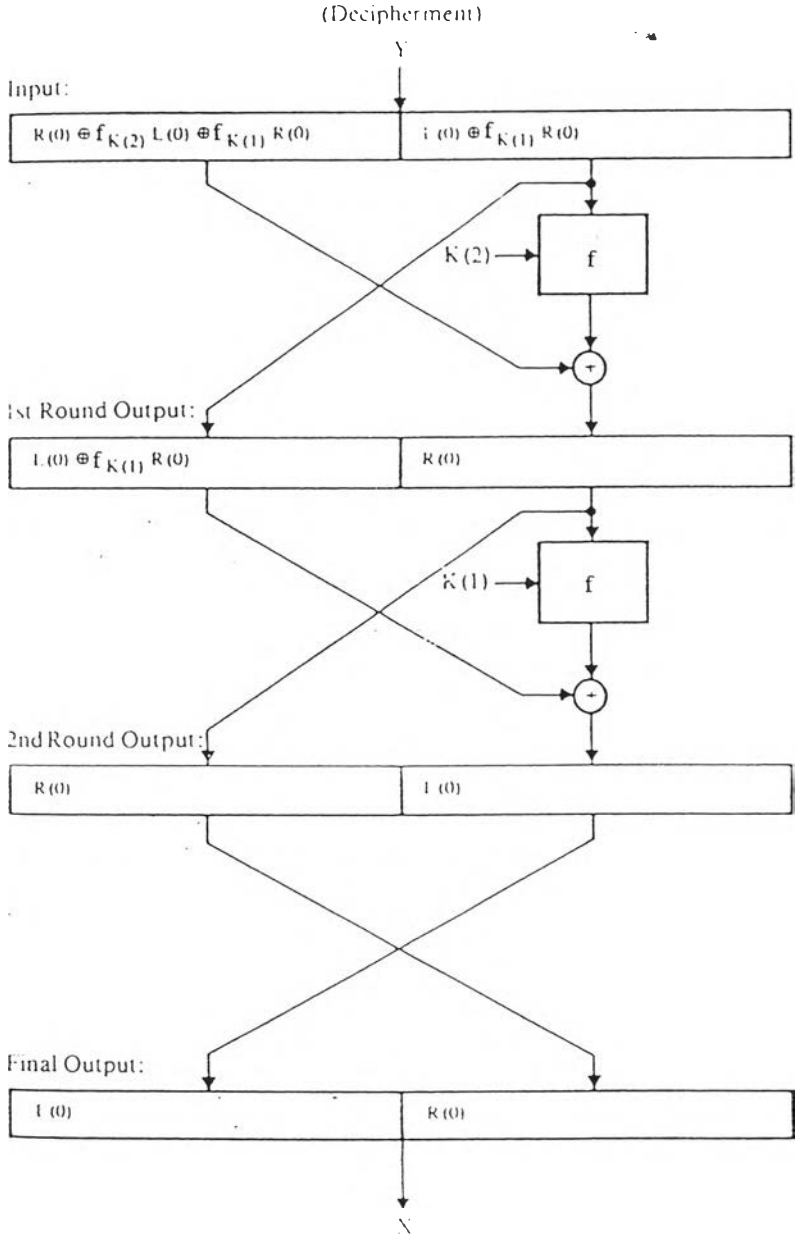
$$Y = [L(0) \oplus f_K(R(0))], R(0) \qquad (3.6)$$

เพื่อให้ได้เพลนเท็กซ์กลับมาได้จะเห็นว่าวิธีการนี้ความปลอดภัยในการรักษาความลับมีน้อยมาก เนื่องจากครึ่งหนึ่งของไซเฟอร์เท็กซ์ยังคงเป็นเพลนเท็กซ์ซึ่งยังไม่ได้ถูกเข้ารหัสอย่างไรก็ตามถ้าหากต้องการให้มีความปลอดภัยมากขึ้นก็ทำได้โดยการเข้ารหัสซ้ำ ๆ กันเป็น จำนวน n รอบโดย

ที่แต่ละรอบจะใช้คีย์ที่แตกต่างกันเพื่อเป็นการแสดงให้เห็นภาพที่ชัดเจน ของการเข้ารหัส n ครั้ง นั้น จะยกตัวอย่างประกอบในกรณีที่ n=2 ดังรูป 3.4



รูป 3.4 ตัวอย่างอัลกอริธึ่มในการเข้ารหัสด้วยบล็อคไซเฟอร์ 2 รอบ [Meyer and Matyas,1982]

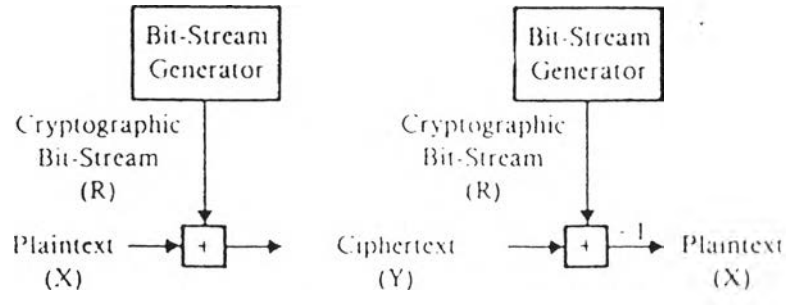


รูป 3.5 การถอดรหัสโดยใช้ฟังก์ชันเดียวกับการเข้ารหัส
[Meyer and Matyas,1982]

จะเห็นว่าข้อมูลจะถูกถอดรหัสออกมาได้ถูกต้อง ก็ต่อเมื่อเราใช้คีย์K(1),K(2)ในการเข้ารหัสและใช้คีย์ K(2),K(1) ในการถอดรหัสในกรณีที่จำนวนรอบของการเข้ารหัสเป็นค่า n ใด ๆ คีย์ที่ใช้ก็ K(1),K(2)....K(n-1),K(n) สำหรับการเข้ารหัสและ K(n),K(n-1)...K(2),K(1) ในการถอดรหัส

สตรีมไซเฟอร์ (Stream Cipher)

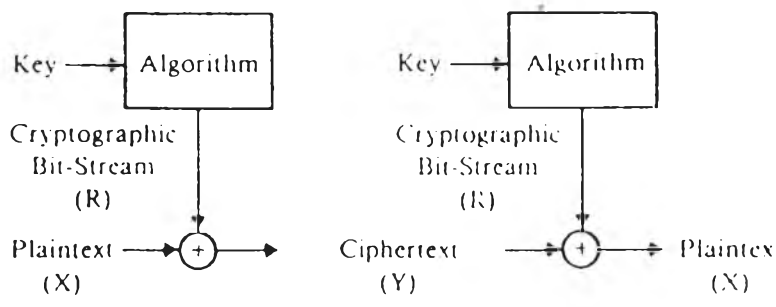
คือการเข้ารหัสแบบหนึ่งโดยมีหลักการดังรูป 3.6



รูป 3.6 แนวคิดของสตรีมไซเฟอร์ [Meyer and Matyas,1982]

จากรูปจะเห็นว่ามีตัวกำเนิดบิทสตรีม(Bit Stream Generator) ซึ่งจะสร้างกระแสของข้อมูลไบนารีซึ่งเรียกว่าคริปโตกราฟฟิกบิทสตรีม (Cryptographic Bit Stream) แล้วนำไปบวกแบบโมดูโล-2กับเพลนเท็กซ์เพื่อทำให้ได้ไซเฟอร์เท็กซ์หรือกับไซเฟอร์เท็กซ์เพื่อให้ได้เพลนเท็กซ์กลับมา ในการเข้ารหัสตัวกำเนิดบิทสตรีมสามารถกำเนิดคริปโตกราฟฟิกบิทสตรีมได้อย่างสุ่มคือ ไม่มีรูปแบบที่แน่นอนในการกำเนิดแล้วนำคริปโตกราฟฟิกบิทสตรีมนี้ไปบวกแบบโมดูโลกับเพลนเท็กซ์เพื่อให้ได้ไซเฟอร์เท็กซ์ออกมาจะเห็นว่าเราสามารถจะมองคริปโตกราฟฟิกบิทสตรีมนี้เป็นเสมือนคีย์ได้เลยคือความปลอดภัยของข้อมูลทั้งหมดขึ้นอยู่กับคริปโตกราฟฟิกบิทสตรีมเองในการถอดรหัสจะต้องนำไซเฟอร์เท็กซ์มาบวกแบบโมดูโล-2กับคริปโตกราฟฟิกบิทสตรีมชุดเดียวกันกับที่ใช้ในการเข้ารหัสแต่เนื่องจากคริปโตกราฟฟิกบิทสตรีมเกิดขึ้นอย่างสุ่มโอกาสที่ผู้ถอดรหัสจะสร้างขึ้นมาเองให้เหมือนกับในตอนเข้ารหัสจึงยากมากจึงต้องรับมาจากทางผู้เข้ารหัสและจำนวนบิทของมันก็เท่ากับไซเฟอร์เท็กซ์ดังนั้นจึงเป็นการไม่สะดวกและไม่มีประสิทธิภาพเนื่องจากผู้รับจะต้องรู้ค่าของทั้งไซเฟอร์เท็กซ์และคริปโตกราฟฟิกบิทสตรีมซึ่งต้องใช้เวลาในการใช้ช่องสื่อสารมากขึ้นด้วยเหตุนี้ตัวกำเนิดบิทสตรีมจะต้องถูกสร้างให้มีลักษณะของการเป็นอัล

กอริธึ่มทั้งนี้เพื่อให้ทั้งทางด้านเข้ารหัสและถอดรหัสจะได้สามารถต่างคนต่างสร้างขึ้นมาให้เหมือนกันได้โดย อาศัยการใช้คีย์เป็นตัวกำหนดการสร้างคริปโตกราฟฟิกบิทสตรีมตามรูป 3.7



รูป 3.7 การใช้วิธีการสร้างบิตสตรีมและการบวกแบบโมดูโล-2ในสตรีมไซเฟอร์ [Meyer and Matyas,1982]

จากรูป 3.7 การเข้ารหัสข้อมูลอินพุท X โดยการนำ X มาบวกแบบโมดูโล-2 กับ คริปโตกราฟฟิกบิทสตรีม R เพื่อที่จะทำให้ได้ไซเฟอร์เท็กซ์ Y

$$Y = X \oplus R \tag{3.7}$$

โดยใช้หลักการของการบวกแบบโมดูโล-2 เราสามารถจะทำให้ได้ค่า X กลับคืนมาโดยการบวกคริปโตกราฟฟิกบิทสตรีม R ตัวเดิมเข้ากับ Y

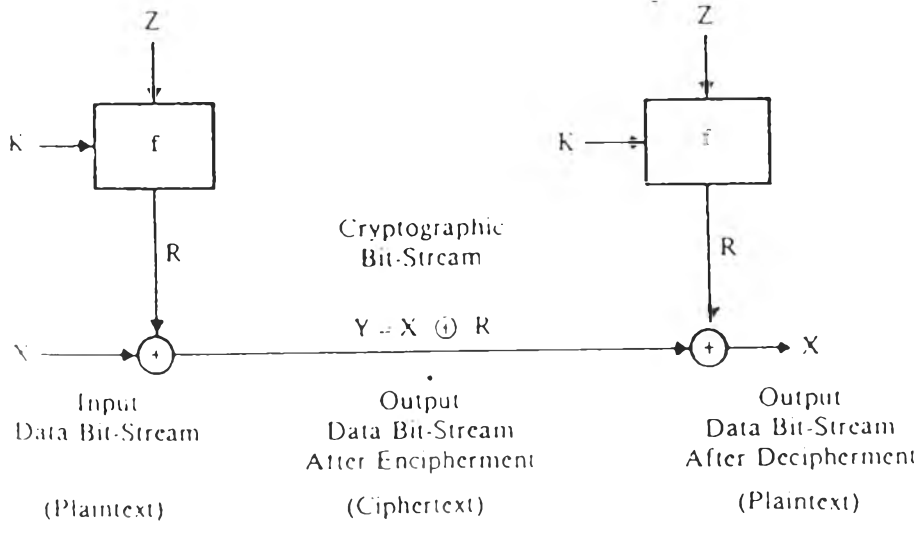
$$X = Y \oplus R$$
$$X = (X \oplus R) \oplus R \tag{3.8}$$

วิธีการเข้ารหัสโดยใช้การบวกแบบโมดูโล-2 ในรูป3.7 นี้ จะเป็นวิธีที่ง่ายมากไม่มีอะไรซับซ้อนอย่างไรก็ตามความปลอดภัยก็จะต่ำเช่นในกรณีที่ฝ่ายตรงข้ามรู้ว่าเราใช้การบวกแบบโมดูโล-2 และรู้ส่วนใดส่วนหนึ่งของเพลนเท็กซ์ X และของไซเฟอร์เท็กซ์ Y ที่ สอดคล้องกัน เขาสามารถจะรู้ค่าคริปโตกราฟฟิกบิทสตรีมได้โดย

$$X \oplus Y = X \oplus (X \oplus R)$$

$$X \oplus Y = R \qquad (3.9)$$

และเนื่องจากคีย์ K ในรูป 3.7 มีค่าคงที่ดังนั้น R ที่ได้ก็จะมีค่าคงที่ในทุกรอบของการเข้ารหัสดังนั้นในรอบต่อไปของการเข้ารหัสถ้าฝ่ายตรงข้ามรู้เพียงไซเฟอร์เท็กซ์ Y ก็จะสามารถรู้ถึงเพลนเท็กซ์ได้เลยตามสมการ $X = Y \oplus R$ โดยไม่จำเป็นต้องรู้คีย์ K จากที่กล่าวมาจะเห็นว่าถึงแม้ว่าคีย์K จะถูกเก็บเป็นความลับก็ไม่สามารถจะป้องกันข้อมูลได้เนื่องจากไม่มีการเปลี่ยนแปลงของคริปโตกราฟฟิกบิทสตรีมRดังนั้นถ้าหากเราทำให้มันมีการเปลี่ยนแปลงโดยการกำหนดข้อมูลขึ้นมาอีกจำนวนหนึ่งซึ่งจะเรียกว่า เวกเตอร์เริ่มต้น Z (Initializing Vector) เพื่อไปใช้ร่วมกับคีย์ K เป็นตัวกำเนิด คริปโตกราฟฟิกบิทสตรีมขึ้นมาโดยเวกเตอร์เริ่มต้น Z นี้จะเปลี่ยนแปลงไปทุกรอบของการ เข้ารหัส ก็จะทำให้ข้อมูลมีความปลอดภัยมากขึ้น ตามรูป 3.8



Legend: example of encipherment and decipherment

	Plaintext	0 1 0 1
⊕	Cryptographic Bit-Stream	0 0 1 1
	Ciphertext	0 1 1 0
	Ciphertext	0 1 1 0
⊕	Cryptographic Bit-Stream	0 0 1 1
	Recovered Plaintext	0 1 0 1

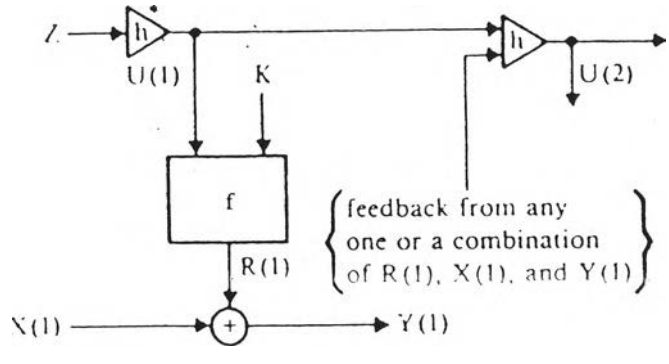
รูป 3.8 สตรีมไซเฟอร์ [Meyer and Matyas,1982]

เวกเตอร์เริ่มต้นที่จะทำให้เกิดคริปโตกราฟฟิกบิทสตรีมที่ทำให้เกิดความปลอดภัยแก่ข้อมูลต้องมีลักษณะอย่างใดอย่างหนึ่งดังนี้

1. เกิดขึ้นอย่างสุ่ม Random โดย Z จะต้องเกิดขึ้นอย่างสุ่มและมีโอกาสที่จะเกิดขึ้นซ้ำเดิมน้อยมาก

2. เกิดขึ้นอย่างสุ่มเทียม Pseudo-Random คือ Z จะถูกสร้างให้เกิดขึ้นอย่างสุ่มด้วยวิธีการที่กำหนดไว้ ซึ่งมีโอกาสที่ค่า Z จะเกิดซ้ำกันแต่ช่วงเวลาที่จะเกิดซ้ำกันต้องห่างกัน

ในระบบรักษาความปลอดภัยที่เป็นแบบสตรีมไซเฟอร์โดยทั่วไปแล้วไม่จำเป็นที่จะต้องเก็บเวกเตอร์เริ่มต้นไว้เป็นความลับดังนั้นฝ่ายตรงข้ามก็สามารถจะรู้ค่าได้แต่ในกรณีที่อัลกอริธึ่มและคีย์มีค่าคงที่นั้นคริปโตกราฟฟิกบิทสตรีมจะเปลี่ยนแปลงได้ก็ต่อเมื่อมีการเปลี่ยนแปลงของเวกเตอร์เริ่มต้นซึ่งสามารถกระทำได้โดยการใช้เวกเตอร์เริ่มต้นใหม่สำหรับทุกรอบของอัลกอริธึ่มซึ่งเวกเตอร์ใหม่แต่ละรอบนั้นได้มาจากการฟีดแบ็คข้อมูลมาจากคริปโตกราฟฟิกบิทสตรีม, เพลนเท็กซ์หรือไซเฟอร์ที่เกิดขึ้นในรอบก่อนหน้านี้ ดังรูป 3.9



รูป 3.9 การเข้ารหัสบล็อคแรกของเพลนเท็กซ์โดยใช้สตรีมไซเฟอร์
[Meyer and Matyas,1982]

<u>ข้อแตกต่างระหว่างบล็อกไซเฟอร์และสตรีมไซเฟอร์</u>

1. บล็อกไซเฟอร์จะทำการเข้ารหัสข้อมูลเป็นบล็อกซึ่งขนาดของบล็อกเป็นตัวหนึ่งในการกำหนดความปลอดภัยของระบบถ้าขนาดของบล็อกหรือจำนวนบิทใน 1 บล็อกมีน้อยระบบก็จะไม่ปลอดภัยในขณะที่สตรีมไซเฟอร์ไม่คำนึงถึงขนาดของบล็อกเพราะความปลอดภัยไม่ขึ้นอยู่กับขนาดของบล็อก

2. กรณีของบล็อกไซเฟอร์ค่าของทุกๆ บิทในไซเฟอร์เท็กซ์จะขึ้นอยู่กับเพลนเท็กซ์ในบล็อกที่อยู่ลำดับเดียวกันแต่ในสตรีมไซเฟอร์ทุก ๆ ไซเฟอร์เท็กซ์บิท Y(i) จะสัมพันธ์กับเพลนเท็กซ์ X(i) โดยมีความเกี่ยวข้องกันคือ

$$Y(i) = X(i)+R(i) \qquad (3.10)$$

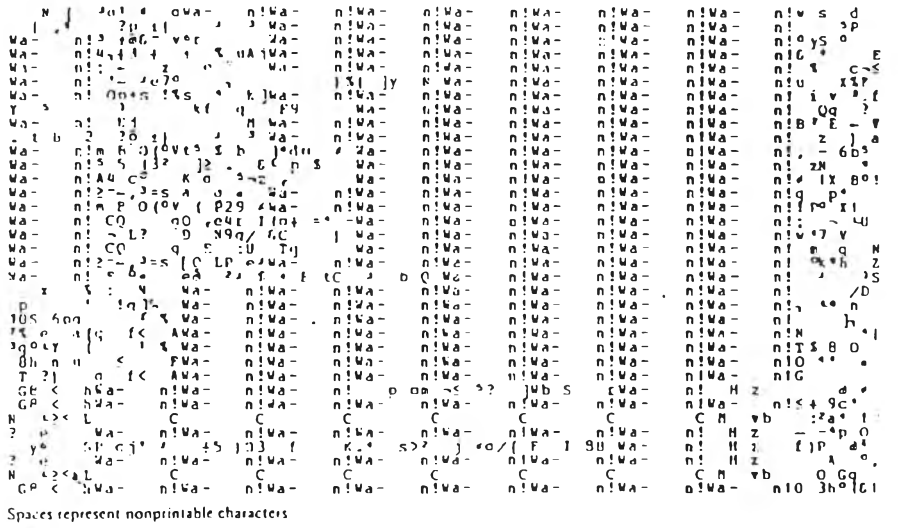
3. บล็อกไซเฟอร์อาจจะมีหรือไม่มีเวกเตอร์เริ่มต้นก็ได้เพราะว่าถ้าฝ่ายตรงข้ามรู้คู่ของเพลนเท็กซ์-ไซเฟอร์เท็กซ์ก็ไม่สามารถจะเปิดเผยข้อมูลได้แต่ในสตรีมไซเฟอร์จะต้องมีเวกเตอร์เริ่มต้น

การเชนนิ่งในบล็อกไซเฟอร์ (Block Cipher with Chaining)

เราสามารถเพิ่มประสิทธิภาพในการรักษาความปลอดภัยได้โดยการใช้เทคนิคการเชนนิ่ง (Chaining Technic) การเชนนิ่งคือ วิธีการที่ทำให้ไซเฟอร์เท็กซ์ไม่ขึ้นอยู่กับคีย์และเพลนเท็กซ์ในเฉพาะบล็อกใดบล็อกหนึ่งเท่านั้นแต่จะขึ้นอยู่กับเพลนเท็กซ์หรือไซเฟอร์เท็กซ์ทุกๆบล็อกก่อนหน้านี้ซึ่งประโยชน์ของการใช้การเชนนิ่งคือจะช่วยลดการซ้ำซ้อนหรือรูปแบบของข้อมูลในไซเฟอร์เท็กซ์ในกรณีที่ข้อมูลเพลนเท็กซ์มีความเป็นรูปแบบหรือซ้ำซ้อนมากการเข้ารหัสด้วยบล็อกไซเฟอร์โดยไม่มีการเชนนิ่งอาจจะไม่สามารถป้องกันการถอดรหัสโดยฝ่ายตรงข้ามซึ่งใช้วิธีการสังเกตุและวิเคราะห์ความถี่ในการเกิดบล็อกได้ดังรูป 3.10 และ 3.11 ดังนั้นจึงใช้วิธีการเชนนิ่งเพื่อขจัดปัญหาดังกล่าว

```
                                                                    0 00034000
                                                                    0 00035000
                                                                    0 00036000
                                                                    0 00037000
                DS    0H
                USING *,R15
                S     @PROLOG
                DC    AL1(16)
                DC    C'IKJSWLP01  75.325'
                DROP  @15
@PPOLOG         STM   @14,@12,12(@13)
                BALR  @09,0
@PSTART         DS    0H
                USING @PSTART,@09
                L     @00,@SIZDATD
                GETMAIN  R,LV=(0)
                LR    @07,@01
                USING @DATD,@07
                ST    @13,@SA00001+4
                LA    @00,@01,20(@13)
                ST    @07,8(,@13)
                LR    @13,@07
                MVC   @PC00001(12),0(@01)
@@PPREFLN       EQU   64
@@DPREFLN       EQU   32
PARMLEN         EQU   136
SIAMLEN         EQU   200
TIOELEN         EQU   136
IAMLEN          EQU   128
SVAMLEN         EQU   200
*                                   WAIT NO-TIME                    */ 0 00030000
*                                                                     0 00031000
*  /*****************************************************************/ 0 00032000
*  /*                                                               */ 0 00033000
*  /* FIRST LETS MAKE SURE WE ARE NOT ALREADY ACTIVE                */ 0 00034000
*  /*                                                               */ 0 00035000
*  /*****************************************************************/ 0 00036000
*                                                                     0 00037000
```

รูป 3.10 ตัวอย่างของเพลนเท็กซ์ที่มีความมีรูปแบบของข้อมูลมาก
[Meyer and Matyas,1982]

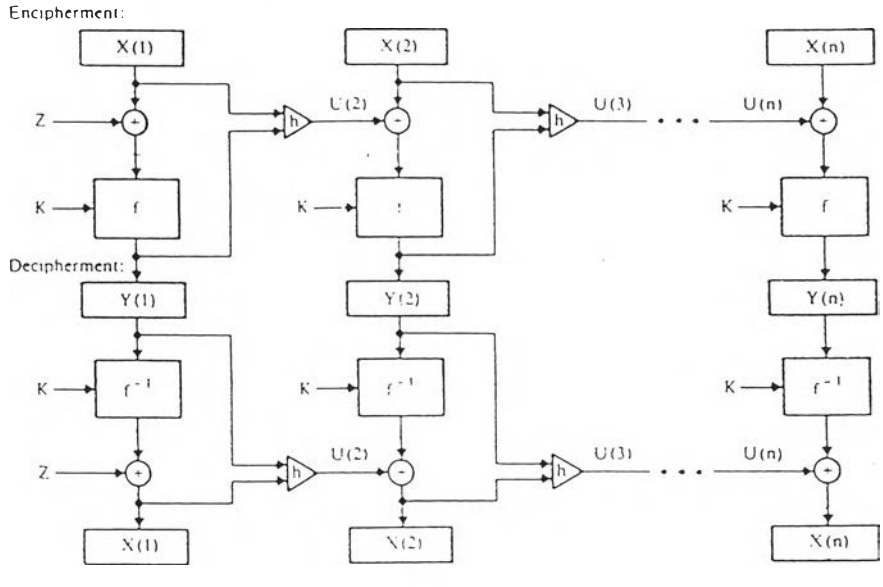


Spaces represent nonprintable characters

K = Hex "85CDCB1C9BD0851A" is the parity-adjusted key used for encipherment. Hexadecimal, or "Hex" for short, is a base-sixteen system for representing numbers. The numbers 0 through 15 are represented by digits 0 through 9, and letters A through F, respectively.

รูป 3.11 ไซเฟอร์เท็กซ์ที่ได้จากการเข้ารหัสเพลนเท็กซ์ในรูป 3.10 โดยใช้ DES ที่ไม่มีการเซนนิ่ง [Meyer and Matyas,1982]

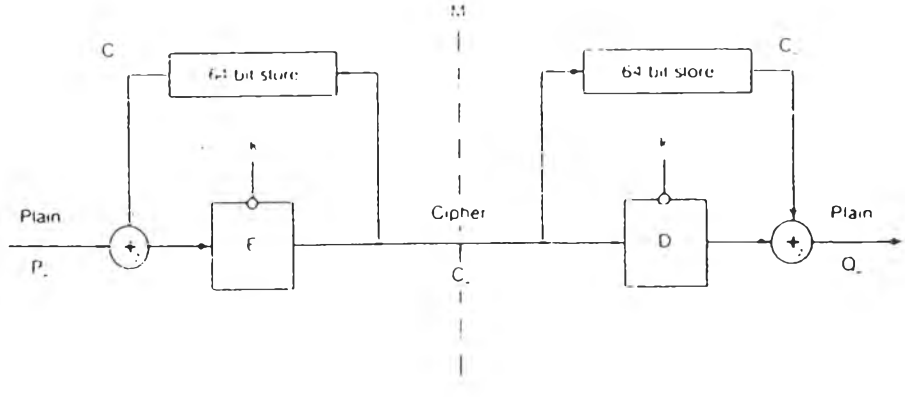
เทคนิคในการเซนนิ่งมีหลายแบบแต่มีหลักการเดียวกันคือการนำเอาเวกเตอร์เริ่มต้นมาทำการบวกแบบโมดูโล-2เข้ากับข้อมูลเพลนเท็กซ์หรือไซเฟอร์เท็กซ์ในรอบปัจจุบันก่อนจะทำการเข้ารหัสเพื่อให้ได้เวกเตอร์เริ่มต้นสำหรับการเข้ารหัสในรอบต่อไปดังนั้นจะเห็นว่าความมีรูปแบบหรือข้อนของเพลนเท็กซ์ก็จะไม่ทำให้ไซเฟอร์เท็กซ์มีความซ้ำซ้อนด้วยเนื่องจากเวกเตอร์เริ่มต้นจะถูกกำเนิดค่าขึ้นมาอย่างสุ่มหรือสุ่มเทียมตัวอย่างของการเซนนิ่งในบล็อคไซเฟอร์แสดงไว้ในรูป 3.12



รูป 3.12 ตัวอย่างของการเซนนิ่งในบล็อคไซเฟอร์โดยวิธีเพลนเท็กซ์ - ไซเฟอร์เท็กซ์ฟีดแบ็ค [Meyer and Matyas,1982]

อย่างไรก็ตามถ้าหากเวกเตอร์เริ่มต้นถูกสร้างขึ้นมาแบบสุ่มหรือสุ่มเทียมโดยไม่มีวิธีการที่แน่นอนนั้นในการถอดรหัสซึ่งก็จำเป็นต้องใช้เวกเตอร์เริ่มต้นนี้ด้วยจะรู้ได้อย่างไรว่าเวกเตอร์เริ่มต้นที่ใช้ในการเข้ารหัสในแต่ละรอบเป็นอะไรบ้างถ้าเป็นดังวิธีการนี้ก็คือนอกจากจะเก็บไซเฟอร์เท็กซ์ไว้ในหน่วยความจำแล้วยังต้องเก็บค่าเวกเตอร์ที่ใช้ในการเข้ารหัสด้วยซึ่งก็จะเป็นข้อเสียเปรียบเพราะจะต้องใช้เนื้อที่ในหน่วยความเพิ่มขึ้นมากวิธีที่จะหลีกเลี่ยงปัญหานี้ก็คือการใช้เทคนิคการเชนนิ่งที่สามารถสร้างเวกเตอร์เริ่มต้นขึ้นมาได้ในแต่ละรอบของการเข้ารหัสหรือถอดรหัสโดยอาศัยเพลนเท็กซ์ ไซเฟอร์เท็กซ์หรือ คีย์ในรอบก่อนหน้านี้เป็นส่วนประกอบ

ตัวอย่างหนึ่งของการใช้ไซเฟอร์เท็กซ์เป็นตัวสร้างเวกเตอร์เริ่มต้นซึ่งมีชื่อเรียกว่าไซเฟอร์บล็อกเชนนิ่ง(Cipher Block Chaining) CBC แสดงในรูปที่ 3.13



รูป 3.13 ไซเฟอร์บล็อกเชนนิ่ง (CBC) [Caelli et al.,1989]

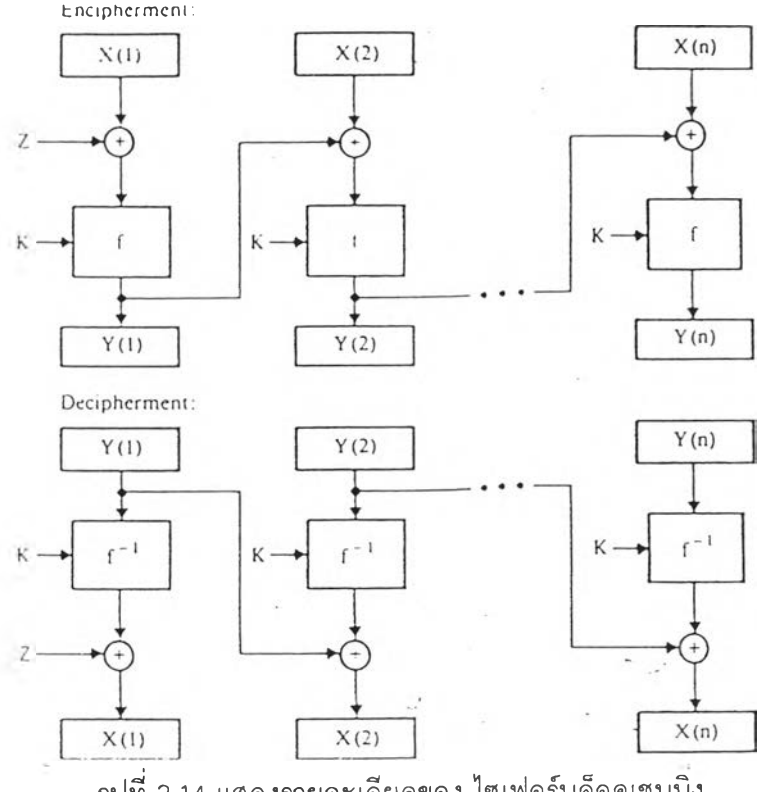
<u>คุณสมบัติการซินโครไนซ์ตัวเองของไซฟอร์บล็อกเชนนิ่ง</u>(Self Synchronizing in Cipher Block Chaining)

วิธีการเข้ารหัสที่ถูกเรียกว่ามีคุณสมบัติเซลฟ์ซินโครไนซ์ซึ่งคือ วิธีการที่ถ้าหากว่าข้อมูลเกิดผิดพลาดขึ้นในขณะเข้ารหัสข้อมูลหลังจากนั้นจะต้องถูกแก้ไขให้ถูกต้องโดยตัวของมันเองไม่ผิดพลาดไปทั้งหมดเพื่อความชัดเจนจะขอยกตัวอย่างในกรณีที่มีอุปกรณ์เข้ารหัส 2 ชุด ที่เหมือนกันทุกประการคือถ้ามีอินพุทที่เหมือนกันเอาท์พุทที่ได้ก็จะเหมือนกันและเราสมมติว่าถ้าเกิดข้อผิดพลาดขึ้นในอุปกรณ์ชุดที่ 1 ดังนั้นเอาท์พุทที่ได้ออกมาย่อมแตกต่างจากตัวที่2 แต่ใน

ช่วงเวลาหนึ่งต่อมาเอาท์พุทที่ได้กลับมาเหมือนกันอีกเราจะเรียกว่ามีคุณสมบัติเซลฟ์ซินโครไนซ์ซิ่ง ซึ่งในการเข้ารหัสแบบ CBC หรือไซเฟอร์บล็อกเชนนิ่งจะมีคุณสมบัตินี้

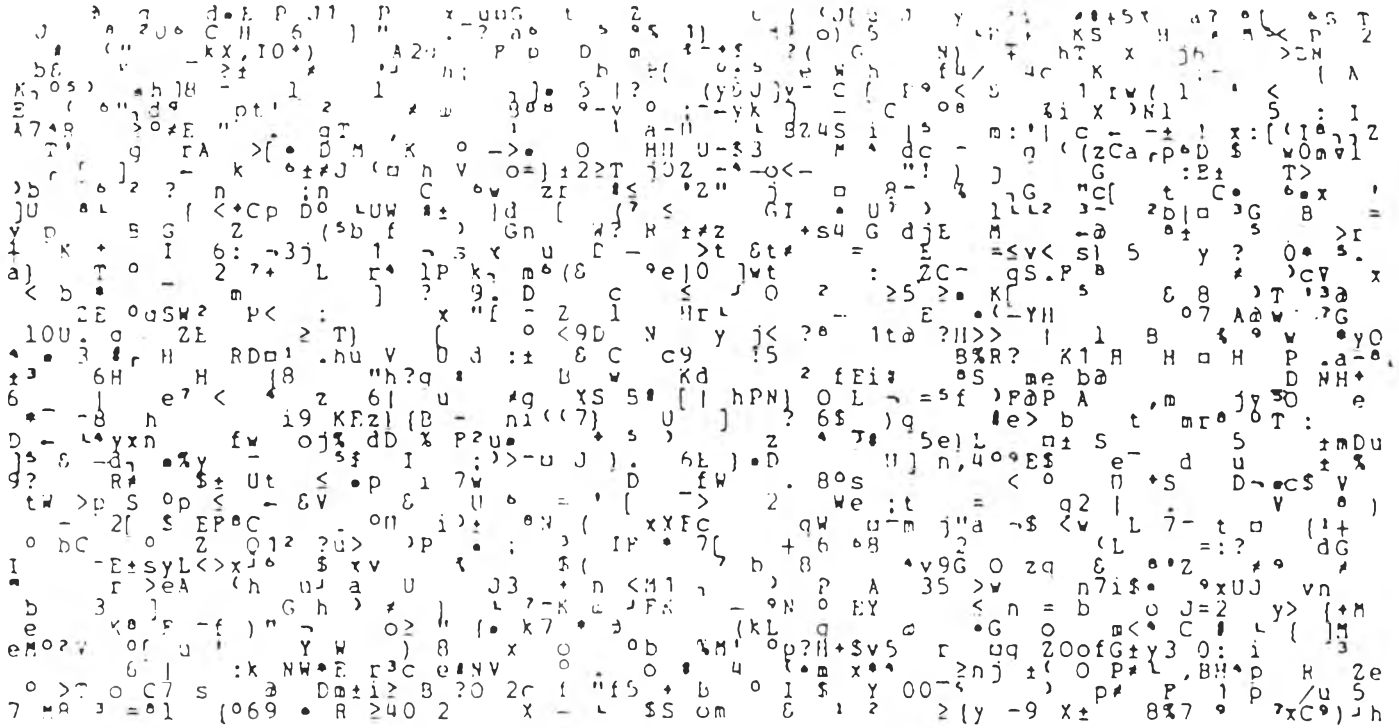
ข้อผิดพลาดที่เกิดขึ้นที่ไซเฟอร์เท็กซ์บล็อกY(i-1) จะทำให้การถอดรหัสกลับมาเป็นเพลนเท็กซ์บล็อก X(i-1) ผิดพลาดหมดทั้งบล็อกแต่ในบล็อก X(i) จะผิดพลาดเฉพาะแต่เพียงบิทที่ผิดในบล็อก Y(i-1) เท่านั้นเช่น กรณีที่ผิดที่บิท 3 และ บิท 7 ใน Y(i-1) เพลนเท็กซ์บล็อก X(i-1) จะผิดทั้งบล็อกแต่เฉพาะบิท 3 และบิท 7 ใน X(i) เท่านั้นที่ผิดพลาดและเพลนเท็กซ์บล็อกต่อไปคือ X(i+1),X(i+2)... จะไม่ถูกผลกระทบจาก การผิดพลาดที่ Y(i-1) ดังนั้นเพลนเท็กซ์ส่วนใหญ่จะสามารถถูกถอดรหัสกลับมาได้แม้ในกรณีที่ไซเฟอร์เท็กซ์มีการผิดพลาดซึ่งลักษณะของการมีเซลฟ์ซินโคนัสนี้เป็นข้อดีของไซเฟอร์บล็อกเชนนิ่งที่เหมาะสำหรับประยุกต์ใช้งานในการใช้รักษาความปลอดภัยของไฟล์ (File Security)เพราะในกรณีที่ไฟล์ที่ถูกเข้ารหัสและเก็บไว้ในหน่วยความจำสำรองเกิดเสียหายขึ้นมาบางส่วนเราก็ยังสามารถถอดรหัสไฟล์ส่วนที่เหลือได้ไม่เสียหายทั้งหมด

รายละเอียดของการเข้ารหัสแบบไซเฟอร์บล็อคเชนนิ่งแสดงในรูป3.14ซึ่งสามารถขจัดปัญหาการมีรูปแบบหรือความซ้ำซ้อนของข้อมูลโดยการเปรียบเทียบไซเฟอร์เท็กซ์ที่ได้จากการเข้ารหัสเพลนเท็กซ์ในรูป 3.10 ด้วยวิธีไซเฟอร์บล็อกเชนนิ่ง กับการเข้าแบบที่ไม่มีการเชนนิ่งซึ่งพอจะเห็นว่าความมีรูปแบบหรือความซ้ำซ้อนของข้อมูลลดลงอย่างมาก ดังแสดงในรูป 3.15



รูปที่ 3.14 แสดงรายละเอียดของ ไซเฟอร์บล็อคเชนนิง

[Meyer and Matyas,1982]



Spaces represent nonprintable characters.

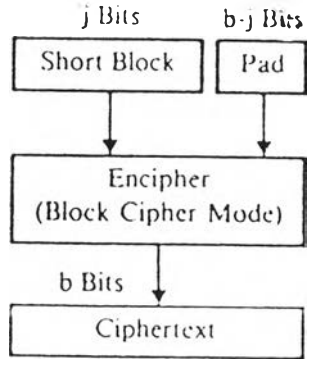
Z = Hex ' 5555555555555555 ' is the initializing vector.

K = Hex ' 85CDCB1C9BD0851A ' is the parity-adjusted key used for encipherment.

รูป 3.15 ไซเฟอร์เท็กซ์ที่ได้จากการเอนคริปต์เพลนเท็กซ์ในรูป 3.10 โดยใช้ไซเฟอร์บล็อคเชนนิ่ง [Meyer and Matyas,1982]

การเข้ารหัสข้อมูลที่ไม่ครบเต็มบล็อก (Short Block Encryption)

การเข้ารหัสแบบบล็อกไซเฟอร์ในแต่ละรอบของการเข้ารหัสจะต้องจัดข้อมูลออกเป็นบล็อกที่มีความยาวจำกัดจำนวนหนึ่ง (โดยทั่วไปมีขนาด 64 บิท) แต่ในกรณีที่บล็อกมีขนาดไม่ถึง 64 บิท เช่น บล็อกสุดท้ายของข้อมูลหรือบล็อกแรกของข้อมูลที่มีขนาดไม่ถึง 64 บิท เราจำเป็นต้องเพิ่มข้อมูลให้ครบเต็มจำนวนบล็อกโดยวิธีที่เรียกว่าแพดดิ้ง(Padding)เพื่อให้สามารถทำการเข้าหรือถอดรหัสได้ดังแสดงในรูป 3.16 โดยทั่วไปในการแพดดิ้งจะเป็นประโยชน์ในการประยุกต์ใช้เกี่ยวกับหนทางด้านเมสเสจแฮนดลิ่ง (Message Handling) ซึ่งต้องการข้อมูลที่เป็นบล็อกมีขนาดเท่าๆกันแต่ถ้าใช้ในการแพดดิ้งในงานด้านการรักษาความปลอดภัยของไฟล์อาจจะมีปัญหาเกิดขึ้นมาได้เนื่องจากหน่วยความจำสำรองไม่พอเพราะข้อมูลมีขนาดใหญ่ขึ้น



รูป 3.16 การเข้ารหัสข้อมูลที่ไม่ครบเต็มบล็อก [Meyer and Matyas,1982]

การเชนนิ่งใน สตรีมไซเฟอร์(Stream Ciphers with Chaining)

การเชนนิ่งในบล็อกไซเฟอร์สามารถใช้ในการแก้ปัญหาเรื่องของความมีรูปแบบของข้อมูลเพลนเท็กซ์ซึ่งก็จะปรากฏรูปแบบเช่นเดียวกับในไซเฟอร์เท็กซ์นอกจากนั้นก็ยังทำให้เกิดคุณสมบัติการแพร่กระจายความผิดพลาด (ErrorPropagation) ซึ่งเป็นคุณสมบัติที่ใช้ในการทำออเทนธิเคชันสำหรับในสตรีมไซเฟอร์นั้นความมีรูปแบบที่ปรากฏอยู่ในเพลนเท็กซ์ก็จะถูกกำจัดออกในขั้นตอนของการบวกแบบโมดูโล-2ระหว่างเพลนเท็กซ์กับคริปโตกราฟฟิกบิทสตรีมอยู่แล้วโดยไม่จำเป็นต้องมีการเชนนิ่งดังรูป3.8อย่างไรก็ตามการเชนนิ่งในสตรีมไซเฟอร์จะทำให้เกิดการ

ซิงโคไนซ์ตัวเอง(Self-Synchronization)ในกรณีที่ระบบไม่มีการปรับค่าเริ่มต้น(Re-initialize)หลังจากเกิดการผิดพลาด

<u>วิธีการเชนนิ่งที่มีคุณสมบัติการกระจายความผิดพลาด (Chaining Method with a Property of Error Propagation)</u>

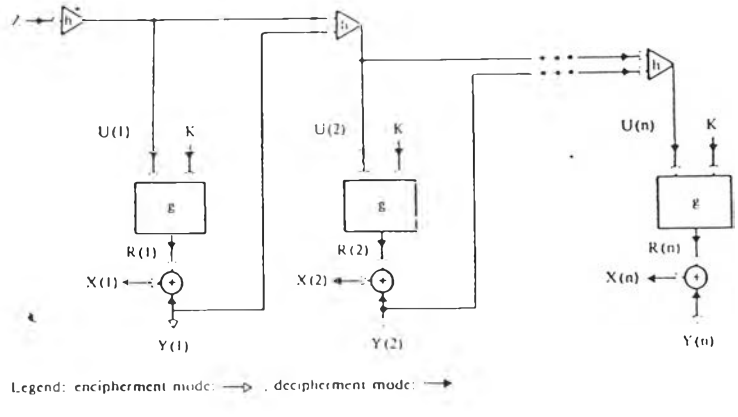
การกระจายความผิดพลาดจะเกิดขึ้นในบล็อกไซเฟอร์ได้เมื่อไรก็ตามที่แต่ละบิทในเพลนเท็กซ์บล็อกที่ถูกถอดรหัสกลับมาหรือX(i)เป็นฟังก์ชันของทุกๆบิทในไซเฟอร์เท็กซ์บล็อกY(1)ถึงY(i) สำหรับในสตรีมไซเฟอร์ที่มีสมการดังนี้

$$Y(i) = X(i) \oplus R(i) \qquad ; i \geq 1 \qquad (3.11)$$

$$X(i) = Y(i) \oplus R(i) \qquad ; i \geq 1 \qquad (3.12)$$

โดยที่ $R(i) = [r_1(i), r_2(i), \ldots, r_b(i)]$

เนื่องจากการบวกแบบโมดูโล-2 ใน 3.12 บิทที่ j ใน X(i) จะขึ้นอยู่กับบิทที่ j เท่านั้นในY(i)แต่ไม่ขึ้นอยู่กับบิทอื่นจะเห็นว่าจะยังไม่มีการกระจายความผิดพลาด แต่ถ้าเราต้องการให้เกิดการกระจายความผิดพลาดบิทที่ j ของ X(i) จะต้องขึ้นอยู่กับทุก ๆ บิทใน Y(1) ถึง Y(i-1) และเพื่อที่จะให้เกิดลักษณะดังกล่าวการป้อนกลับควรจะมาจากเพลนเท็กซ์ X,เวกเตอร์เริ่มต้น Z หรือผลรวมของทั้ง2อย่างเพิ่มเติมขึ้นมาจากการป้อนกลับที่มาจากไซเฟอร์เท็กซ์อย่างเดียว ตัวอย่างของสตรีมไซเฟอร์ที่มีคุณสมบัติของการ กระจายความผิดพลาดแสดงดังในรูป 3.17



รูป 3.17 สตรีมไซเฟอร์ที่มีคุณสมบัติของการกระจายความผิดพลาด
[Meyer and Matyas,1982]

ในกรณีที่เราป้อนกลับเฉพาะเพลนเท็กซ์อย่างเดียว คุณสมบัติในการกระจายความผิดพลาดเนื่องจากการผิดพลาดของไซเฟอร์เท็กซ์ก็ยังจะมีอยู่ดังแสดงได้ด้วยสมการ

$$Y(i) = X(i) \oplus g_K(Z) \quad ; i = 1 \qquad (3.13)$$

$$Y(i) = X(i) \oplus g_K(X(i-1)) \quad ; i > 1 \qquad (3.14)$$

และในกรณีของการถอดรหัส

$$X(i) = Y(i) \oplus g_K(Z) \quad ; i = 1 \qquad (3.15)$$

$$X(i) = Y(i) \oplus g_K(X(i-1)) \quad ; i > 1 \qquad (3.16)$$

จากสมการจะเห็นว่าแต่ละบิทในเพลนเท็กซ์บล็อกX(i) ที่ถอดรหัสกลับมาขึ้นอยู่กับแต่ละบิทในเวกเตอร์เริ่มต้น z และแต่ละบิทในไซเฟอร์เท็กซ์ Y(i) ถึง Y(i-1) โดย ความสัมพันธ์ที่เกี่ยวเนื่องกัน (RecursiveRelation) ดังในสมการ 2.24

จากรูป 3.17 สามารถเขียนเป็นสมการทางคณิตศาสตร์แสดงการเข้าและถอดรหัสได้ คือ

$$Y(i) = X(i) \oplus g_K(U(i)) \quad ; i \geq 1 \qquad (3.17)$$

$$X(i) = Y(i) \oplus g_K(U(i)) \quad ; i \geq 1 \qquad (3.18)$$

โดยที่

$$U(I) = \begin{cases} h(Z) & ; i = 1 \\ h(U(i-1), Y(i-1)) & ; i > 1 \end{cases} \qquad (3.19)$$

h^* คือฟังก์ชันที่ทำให้จำนวนบิทของ Z เท่ากับ U ในกรณีที่ไม่เท่ากัน

<u>วิธีการเชนนิ่งที่มีคุณสมบัติการซิงโครไนซ์ตัวเอง</u>

รูป 3.17 ก็สามารถจะทำให้เป็นสตรีมไซเฟอร์ที่สามารถซิงโครไนซ์ตัวเองได้ โดยการกำหนดให้ฟังก์ชัน h เป็นดังนี้

$$h((Ui-1) , Y(I-1)) = Y(I-1) \quad ; i > 1 \qquad (3.20)$$

นั่นก็คือการป้อนกลับไซเฟอร์เท็กซ์กลับเข้าทางอินพุทของอัลกอริธึ่มและโดยการกำหนดให้ $Y(0) \equiv Z$ จะทำให้ได้ว่า

$$g_K(U(i)) = g_K(Y(i-1)) \quad ; i \geq 1 \qquad (3.21)$$

ดังนั้นจากสมการที่ 3.17 และ 3.18 การเข้ารหัสและถอดรหัสสามารถแสดงได้ดังสมการ

$$Y(i) = X(i) \oplus g_K(Y(i-1)) \quad ; i \geq 1 \qquad (3.22)$$

และ

$$X(i) = Y(i) \oplus g_K(Y(i-1)) \quad ; i \geq 1 \qquad (3.23)$$

จากสมการ 3.23 ถ้าหากสมมติว่าเกิดข้อผิดพลาดขึ้นในไซเฟอร์เท็กซ์ Y(i-1) ก็จะเกิดผลกระทบกับทุกๆ บิทในบล็อกที่เป็นเอาท์พุทของ $g_K(Y(i-1))$และก็จะทำให้ทุกๆบิทในเพลนเท็กซ์ X(i)ที่ถูกถอดรหัสกลับมาผิดหมด ยิ่งไปกว่านั้นถ้าเกิดข้อผิดพลาดใน Y(i-1) ในตำแหน่งบิทใดก็จะทำให้เกิดข้อผิดพลาดขึ้นทันทีเพลนเท็กซ์ X(i-1) ที่ตำแหน่งเดียวกันด้วยข้อสรุปของสมการ 3.23 ก็คือถ้าหากเกิดความผิดพลาดขึ้นที่ไซเฟอร์เท็กซ์ Y(i-1) จะมีผลกระทบค่าเพลนเท็กซ์ที่ถูกถอดรหัสกลับมา X(i-1) และ X(i) เท่านั้น แต่จะไม่มีผลต่อ X(i+1), X(i+2)... ดังนั้นจึงกล่าวได้ว่าลักษณะการ เข้าและถอดรหัสแบบนี้มีการซินโครไนซ์ตัวเอง

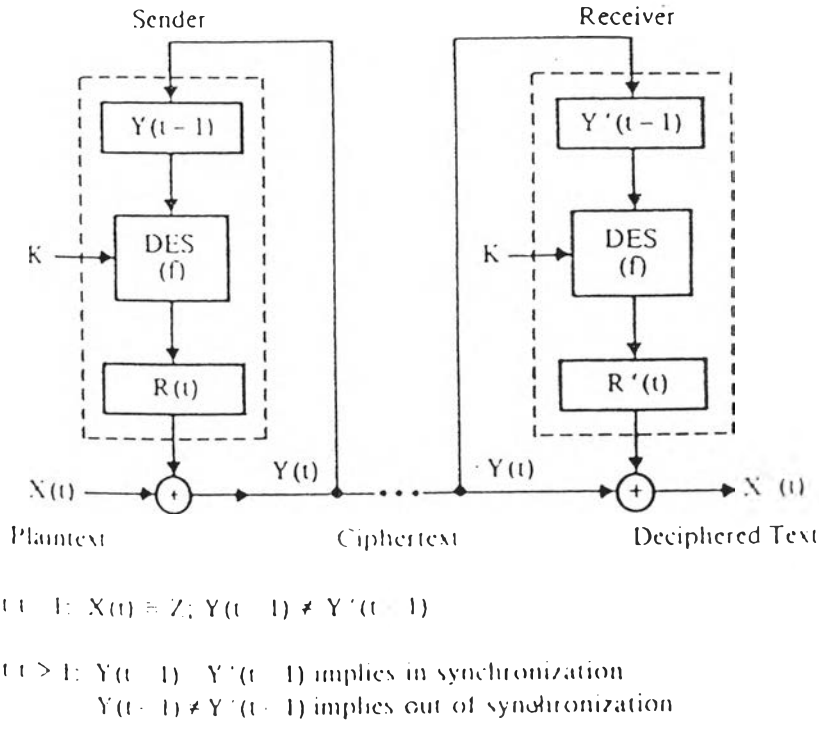
<u>ตัวอย่างของสตรีมไซเฟอร์ที่มีการซินโครไนซ์ตัวเอง</u>

ตัวอย่างของสตรีมไซเฟอร์ที่มีการซินโครไนซ์ตัวเอง คือ

1. ไซเฟอร์เท็กซ์ออโตคีย์ไซเฟอร์(Ciphertext Auto-Key Cipher)

ซึ่งแสดงในรูป 3.18คริปโตกราฟฟิกบิทสตรีมขนาด 64 บิทจะถูกสร้างขึ้นโดยการเข้ารหัสสข้อมูลที่อยู่ในอินพุทรีจิสเตอร์สมมติให้อินพุทรีจิสเตอร์ทางด้านส่งคือ Y และทางด้านรับคือ Y'และคริปโตกราฟฟิกบิทสตรีมจะถูกเก็บอยู่ในเอาท์พุทรีจิสเตอร์ R และ R' ของด้านส่งและด้านรับตามลำดับข้อมูลที่อยู่ใน Y , Y', Rและ R'ที่เวลา t คือ Y(t), $Y'(t)$, R(t)และ $R'(t)$ ก่อนที่การติดต่อสื่อสารจะเริ่มต้นเครื่องส่งและเครื่องรับจะถูกทำให้

ซินโครไนซ์กันก่อนเพราะว่าในขณะเริ่มแรก เราสมมติว่า Y(0) และ Y′(0) มีค่าไม่เท่ากัน



รูป 3.18 ไซเฟอร์เท็กซ์ออโตคีย์ไซเฟอร์ [Meyer and Matyas,1982]

ในขณะที่เวลา t=1 จะเกิดการซินโครไนซ์ขึ้นโดยการส่งเวกเตอร์เริ่มต้น 64 บิทไปที่ Y และ Y′ ทำให้ Y(1) = Y′(1)

เมื่อ t > 0 ; Y(t-1) จะถูกเข้ารหัสโดยใช้คีย์ K ทำให้ได้

$$R(t) = f_K(Y(t-1)) \qquad (3.24)$$

R(t) จะถูกเอ็กคลูซีพออร์กับ X(t) จะได้

$$Y(t) = R(t) \oplus X(t) \qquad (3.25)$$

แล้ว Y(t) จะถูกส่งไปยังเครื่องรับ โดยทางด้านเครื่องรับ Y(t) ที่รับเข้ามาคือ Y′(t-1)ซึ่งจะถูกถอดรหัส (โดยมีอัลกอริธึ่มเดียวกับการเข้ารหัสทางด้านเครื่องส่ง) โดยคีย์ K จะได้สมการ

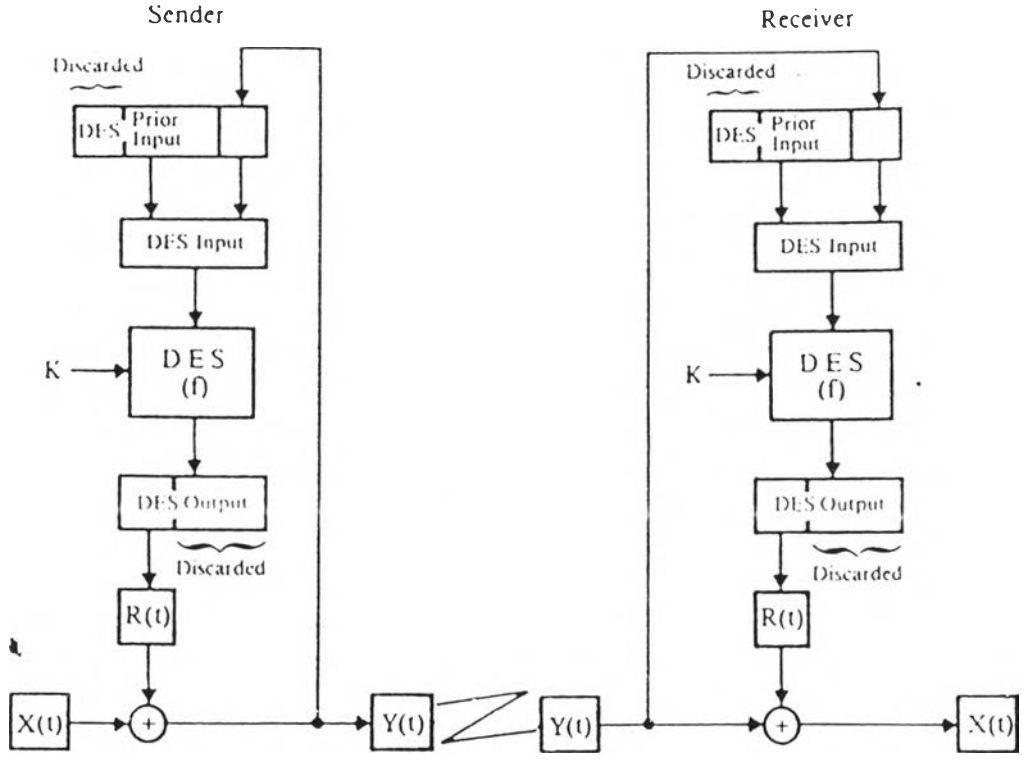
$$R'(t) = f_K(Y'(t-1)) \qquad (3.26)$$

$R'(t)$ จะถูกเอ็กซ์คูลซีพออร์กับไซเฟอร์เท็กซ์บล็อก $Y(t)$ เพื่อที่จะทำให้ได้เพลนเท็กซ์ $X'(t)$

$$X'(t) = Y(t) \oplus f_K(Y'(t-1)) \qquad (3.27)$$

2. ไซเฟอร์ฟีดแบคสตรีมไซเฟอร์ (Cipher Feedback Stream Cipher)

เป็นอัลกอริธึ่มหนึ่งของการเข้ารหัสข้อมูลโดยมีลักษณะการทำงานดังรูป 3.19



รูป 3.19 ไซเฟอร์ฟีดแบ็ค [Meyer and Matyas,1982]

บิทซ้ายสุด n ของเอาท์พุทจาก DES จะถูกบวกแบบโมดูโล-2กับ n บิทของเพลนเท็กซ์เพื่อสร้างไซเฟอร์เท็กซ์ขนาด n บิทโดยที่n คือจำนวนบิทที่จะทำการเข้ารหัสใน 1ครั้ง($1 \leq n \leq 64$) ไซเฟอร์เท็กซ์ n บิทนี้จะถูกย้อนกลับไปยังด้านอินพุทของ DES โดย การเลื่อนข้อมูลในอินพุทของ DES ไปทางซ้าย n บิทแล้วนำ n บิทที่ย้อนกลับมานี้เติมท้ายให้เต็มเพื่อเป็นข้อมูลใน

การผ่านไปยัง DES ในรอบต่อไปไซเฟอร์ฟีดแบคมีคุณสมบัติของการซินโครไนซ์ตัวเองเพื่อให้ง่ายแก่การเข้าใจจะขอยกตัวอย่างตามรูป3.18สมมติว่าจะเข้ารหัสข้อมูลทีละ8บิท(n=8)และสมมติว่าในไซเฟอร์เท็กซ์ Y(t1) ข้อมูลถูกเปลี่ยนค่าไป 1 บิท ทำให้เป็น $Y^*(t_1)$ หลังจากนั้นอินพุท ของ DESทั้งทางด้านส่งและรับจะเป็นไปดังรูป 3.20 ในกรณีนี้ ไซเฟอร์เท็กซ์ $Y^*(t_1)$, $Y(t_2)$, . . . , Y(8)จะถูกถอดรหัสออกมาผิดหมดเนื่องจากอินพุทของ DES ไม่ถูกต้อง หลังจากที่ไซเฟอร์เท็กซ์ที่ไม่โดนเปลี่ยนแปลงผ่านไป 8 บล็อก คือ $Y(t_2)$, $Y(t_3)$, . . . ,$Y(t_9)$ ข้อมูลทางอินพุทของ DES ทั้งทางด้านรับและส่งก็จะกลับมาค่าเท่ากันอีกครั้งหนึ่งซึ่งก็ทำให้ระบบซินโครไนซ์กันใหม่นั่นเอง

Iteration	DES Input At Sender	DES Input At Receiver
0	0 0 0 S1 S2 S3 S4 S5	0 0 0 S1 S2 S3 S4 S5
1	0 0 S1 S2 S3 S4 S5 $Y(t_1)$	0 0 S1 S2 S3 S4 S5 $\hat{Y}(t_1)$
2	0 S1 S2 S3 S4 S5 $Y(t_1)$ $Y(t_2)$	0 S1 S2 S3 S4 S5 $\hat{Y}(t_1)$ $Y(t_2)$
3	S1 S2 S3 S4 S5 $Y(t_1)$ $Y(t_2)$ $Y(t_3)$	S1 S2 S3 S4 S5 $\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$
4	S2 S3 S4 S5 $Y(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$	S2 S3 S4 S5 $\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$
5	S3 S4 S5 $Y(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$	S3 S4 S5 $\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$
6	S4 S5 $Y(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$	S4 S5 $\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$
7	S5 $Y(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$	S5 $\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$
8	$Y(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$ $Y(t_8)$	$\hat{Y}(t_1)$ $Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$ $Y(t_8)$
9	$Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$ $Y(t_8)$ $Y(t_9)$	$Y(t_2)$ $Y(t_3)$ $Y(t_4)$ $Y(t_5)$ $Y(t_6)$ $Y(t_7)$ $Y(t_8)$ $Y(t_9)$

รูป 3.20 แสดงข้อมูลทางด้านผู้ส่งและผู้รับของไซเฟอร์ฟีดแบ็คที่มีคุณสมบัติซินโครไนซ์ตัวเอง [Meyer and Matyas,1982]

โดยทั่วไปแล้วบิทที่ถูกเปลี่ยนแปลงในไซเฟอร์เท็กซ์จะทำให้เกิดการเปลี่ยนแปลง(หรือผิดพลาด) ในเพลนเท็กซ์ที่ถอดรหัสกลับมาที่ตำแหน่งนั้นกับอีก 64 บิทของเพลนเท็กซ์ที่ตามมาหลังจากนั้นก็จะซินโครไนซ์กันอย่างไรก็ตามถ้าหากว่าการเปลี่ยนแปลงในไซเฟอร์เท็กซ์เป็นการเพิ่มหรือลดจำนวนบิทลงไปจะทำให้เกิดสภาวะที่ไม่ซินโครไนซ์ (Out of Sync) เพลนเท็กซ์ที่ถอดรหัสกลับมาจะผิดพลาดหมดเนื่องจากขอบเขตของบล็อก (Block Boundary) ของข้อมูลทางด้านรับและส่งไม่ตรงกัน