วิทยาการเข้ารหัสลับ (Cryptography)

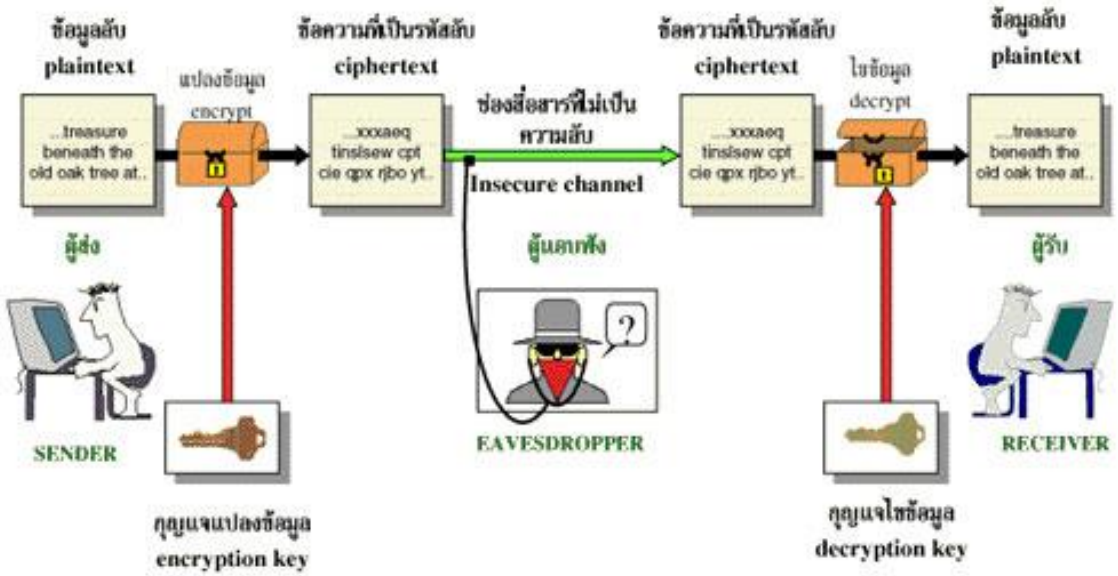
กล่าวนำ

มนุษย์ได้คิดค้นวิธีการรักษาความลับของเรามาตั้งนาน นับตั้งแต่สมัยจูเลียส ซีซาร์ จนกระทั่งถึงปัจจุบันที่ใช้คอมพิวเตอร์มาช่วยเข้ารหัสลับและถอดรหัสลับ การเข้ารหัสแบบซีซ่าร์ ทำได้โดยการนำตัวอักษรที่อยู่ถัดไปอีกสองตำแหน่งมาแทนที่ ยกตัวอย่างเช่น ถ้าต้องการเข้ารหัสคำว่า HELLO เราก็นำตัวอักษรที่ถัดจากตัว H ไปอีกสองตัว นั่นคือตัว J มาแทน ตัว E แทนด้วย G ตัว L แทนด้วย N ตัว O แทนด้วย Q ดังนั้นข้อความ HELLO จึงถูกแปลงให้เป็นคำว่า JGNNQ

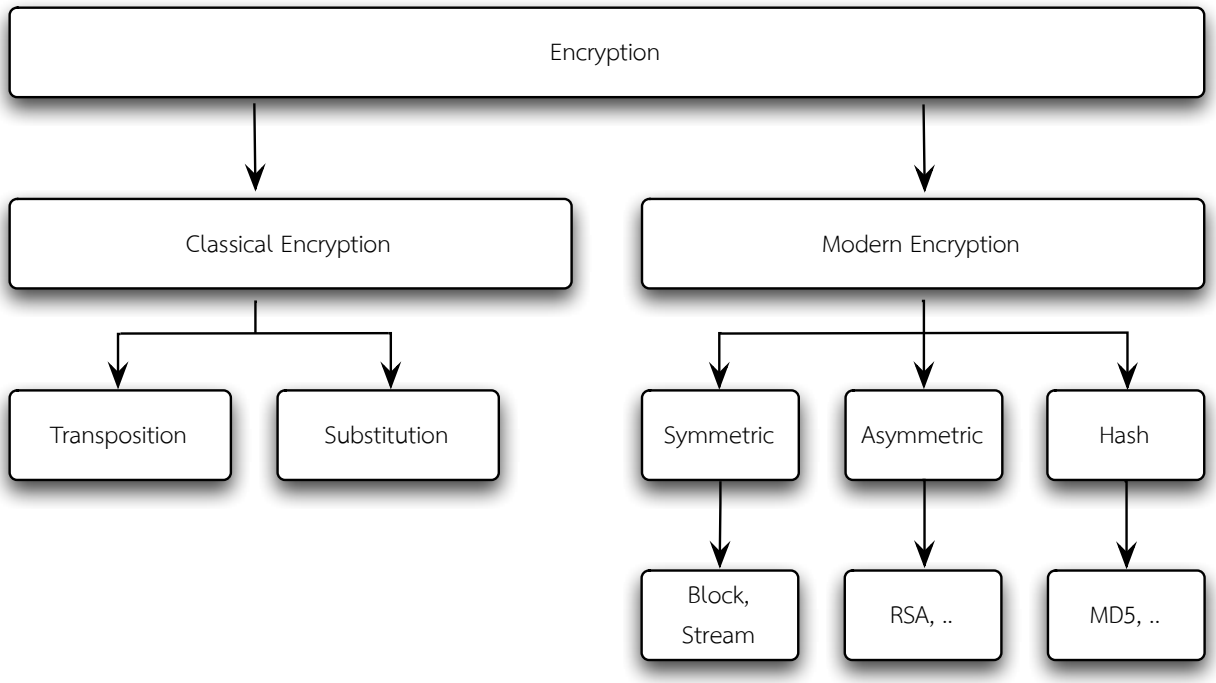
การเข้ารหัสลับแตกต่างกับวิทยาการอำพรางข้อมูล ข้อมูลที่ถูกอำพรางนั้นจะไม่ถูกเปลี่ยนแปลง ในขณะที่การเข้ารหัสลับจะเปลี่ยนแปลงข้อมูล วิทยาการเข้ารหัสลับสมัยใหม่ (Modern Cryptography) เป็นวิชาการที่ใช้แนวทางคณิตศาสตร์เพื่อแปลงข้อความปกติให้กลายเป็นข้อความลับ โดยให้เฉพาะคู่สนทนาที่ต้องการสามารถอ่านเข้าใจได้เท่านั้น ขั้นตอนวิธีของการเข้ารหัสลับสมัยใหม่ ได้แก่ Data Encryption Standard, Advanced Encryption Standard หรือ One-Time Padding ฯลฯ

การเข้ารหัสข้อมูล (Cryptography)

การเข้ารหัสข้อมูลโดยพื้นฐานแล้วจะเกี่ยวข้องกับวิธีการทางคณิตศาสตร์เพื่อใช้ในการป้องกันข้อมูลหรือข้อความตั้งต้นที่ต้องการส่งไปถึงผู้รับ ข้อมูลตั้งต้นจะถูกแปรเปลี่ยนไปสู่ข้อมูลหรือข้อความอีกรูปแบบหนึ่งที่ไม่สามารถอ่านเข้าใจได้โดยใครก็ตามที่ไม่มีกุญแจสำหรับเปิดดูข้อมูลนั้น เราเรียกกระบวนการในการแปรรูปของข้อมูลตั้งต้นว่า "การเข้ารหัสข้อมูล" (Encryption) และกระบวนการในการแปลงข้อความที่ไม่สามารถอ่านและทำความเข้าใจให้กลับไปสู่ข้อความดั้งเดิมว่าการถอดรหัสข้อมูล (Decryption)



หากแบ่ง Cryptography ตามยุคสมัยแล้วเราสามารถที่จะแบ่งได้เป็น 2 ยุคคือ ยุคประวัติศาสตร์ (หรือที่เรียกว่ายุค Classic) และยุคปัจจุบัน (Modern)

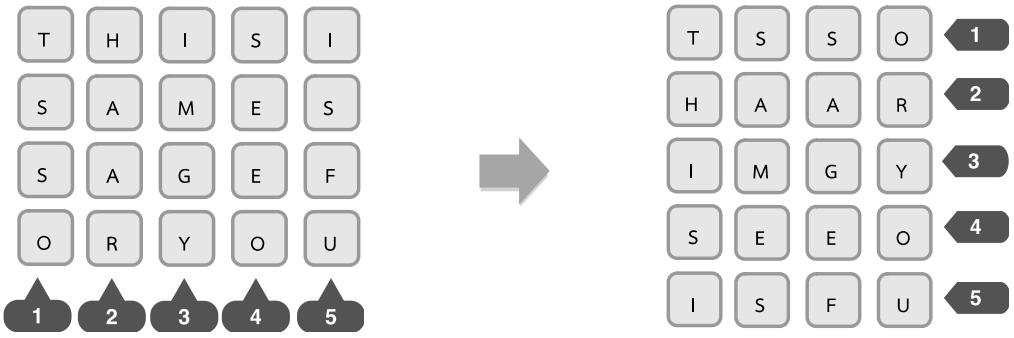


Classical Encryption : Transposition Ciphers

แบบไม่ใช้ Key : ไม่มีการสลับลำดับของหลัก
วิธีการ : เปลี่ยนหลักให้เป็นแถว แล้วทำการเรียงข้อมูลตามแถว

<u>ตัวอย่าง การเข้ารหัส</u>

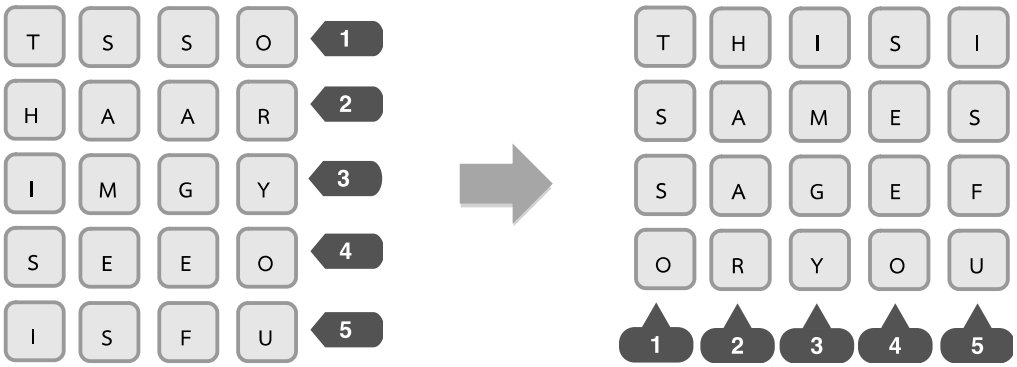
Plain Text : THIS IS A MESSAGE FOR YOU และใช้ 5 หลัก



Cipher Text: TSSOHAARIMGYSEEOISFU

การถอดรหัส

วิธีการ : เปลี่ยนแถวให้เป็นหลัก แล้วทำการเรียงข้อมูลตามแถว



Cipher Text: TSSOHAARIMGYSEEOISFU

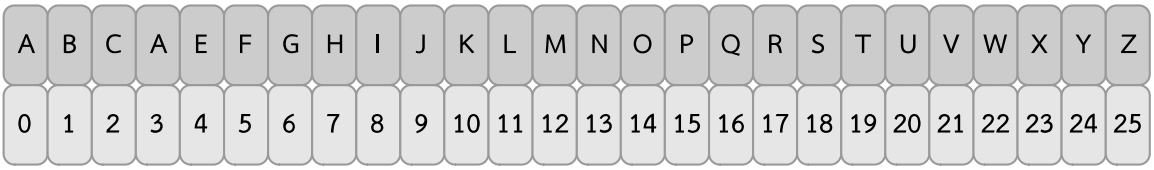
Plain Text : THIS IS A MESSAGE FOR YOU

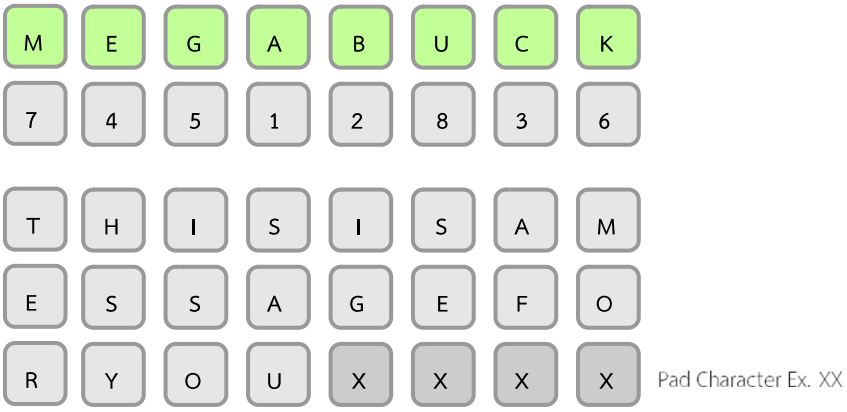
ตัวอย่าง การเข้ารหัส

แบบใช้ Key ช่วย : มีการจัดลำดับหลักตาม Key เช่น MEGABUCK

วิธีการ : จัดลำดับหลักตาม Key เช่น MEGABUCK แล้วทำการเรียงข้อมูลตามหลัก

ตัวอย่าง Plain Text : THIS IS A MESSAGE FOR YOU





Cipher Text : SAUIGXAFXHSYISOMOXTERSEX

การถอดรหัส

วิธีการ : จัดลำดับหลักตาม Key เช่น MEGABUCK แล้วทำการเรียงข้อความตามแถว

A	B	C	A	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Cipher Text: SAUIGXAFXHSYISOMOXTERSEX

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6

S	I
A	G
U	X

Plain Text : THIS IS A MESSAGE FOR YOU

Classical Encryption : Substitution Ciphers

Caesar Substitution Ciphers : เป็นการแทนค่าแต่ละตัวอักษรด้วยสัญลักษณ์เพียงตัวเดียว เป็นวิธีที่ง่ายที่สุด ใช้มาตั้งแต่สมัยจูเลียส ซีซาร์ ในการเข้ารหัสเนื้อความจดหมายส่งไปให้ทัพทหารระหว่างการรบ

ตัวอย่าง การเข้ารหัส

วิธีการ : ทำได้โดยการนำตัวอักษรที่อยู่ถัดไปอีก 3(Key) ตำแหน่งมาแทนที่

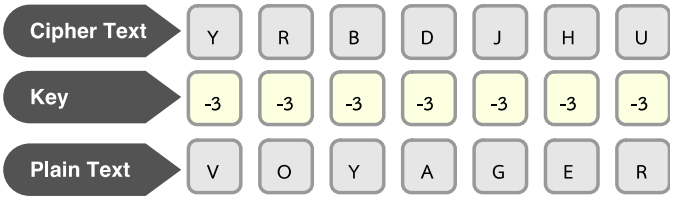
Plain Text	V	O	Y	A	G	E	R
Key	+3	+3	+3	+3	+3	+3	+3
Cipher Text	Y	R	B	D	J	H	U

Plain Text = P+K : V +3 ตำแหน่ง Y

Plain Text : VOYAGER ➡ **Cipher Text** : YRBDJHU

การถอดรหัส

วิธีการ : ทำได้โดยการนำตัวอักษรที่อยู่ก่อนหน้า 3(Key) ตำแหน่งมาแทนที่



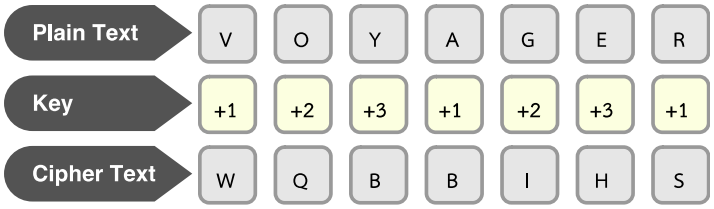
เลื่อนกลับ **1** ตำแหน่ง **: .....**

เลื่อนกลับ **2** ตำแหน่ง **: .....**

เลื่อนกลับ **3** ตำแหน่ง **VOYAGER ;** จะเจอคำที่สามารถอ่านได้

ตัวอย่างแบบมี KEY

วิธีการ : ทำได้โดยการนำตัวอักษรที่อยู่ถัดไป Key ตำแหน่งมาแทนที่ (Key : 1, 2, 3)

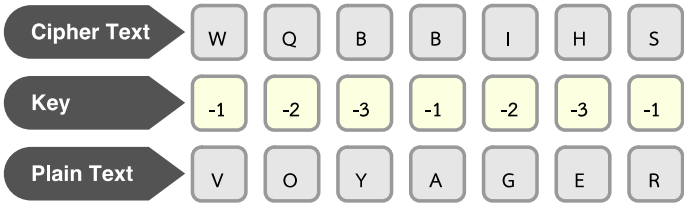


Cipher Text = P+K : V +1 ตำแหน่ง W

Plain Text : VOYAGER **Cipher Text :** WQBBIHS

การถอดรหัส

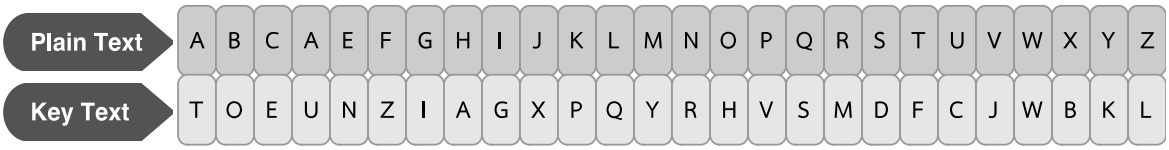
วิธีการ : ทำได้โดยการนำตัวอักษรที่อยู่ก่อนหน้า 3(Key) ตำแหน่งมาแทนที่



Plain Text = C - K : W -1 ตำแหน่ง V

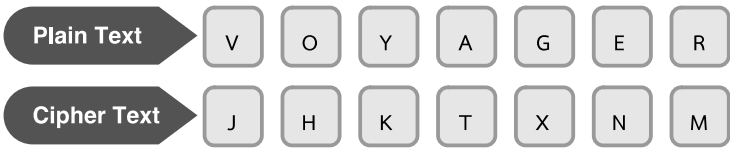
Plain Text : WQBBIHS **Plain Text** : VOYAGER

MonoAlphabetic Substitution Ciphers : เป็นการแทนค่าแต่ละตัวอักษรด้วยสัญลักษณ์เพียงตัวเดียวเช่นกัน แต่เป็นอย่างอิสระหรือไม่มีเหตุผลว่า ทำไมต้องเป็นแบบนี้



<u>การเข้ารหัส</u>

วิธีการ : ทำได้โดยการนำตัวอักษรที่ตรงกับ Key มาแทนที่อักษรเดิม



Cipher Text = V เมื่อเปรียบเทียบกับ Key ทำการแทนค่าอักษร J แทน

Plain Text : VOYAGER ➡ **Cipher Text :** JHKTXNM

<u>การถอดรหัส</u>

วิธีการ : ทำได้โดยการนำตัวอักษรที่ตรงกับ Plain Text มาแทนที่อักษรเดิม

Plain Text = Cipher Text ➡ “J” เมื่อเปรียบเทียบกับตารางแล้วได้ Plain Text คือ “V”

Cipher Text : JHKTXNM **Plain Text** : VOYAGER

Vigenere Substitution Ciphers : Vigenere Cipher เป็นการเข้ารหัสแบบซีเคร็ทคีย์ (Secret Key) หรือ Symmetric Key Cryptography ที่อาศัยพื้นฐานเดียวกันกับ Caesar หลักการของ Vigenere Cipher คือ จะใช้ Key ที่เป็นคำมาเรียงต่อๆกัน แล้วเข้ารหัสโดยสร้าง Caesar Cipher จากตัวอักษรที่ปรากฏอยู่ใน Key

<u>ตัวอย่างการเข้ารหัส</u>

วิธีการ : ทำได้โดยการนำเอาค่าตำแหน่งของตัวอักษรภาษาอังกฤษนั้นๆ มาบวกกันได้ค่าจำนวนหนึ่งออกมา แล้วนำค่าที่ได้ไปหาตัวอักษรตามตำแหน่งที่บวกได้มาแทนค่าตัวอักษรเดิม

เรามี **Plaintext : ATTACK** และเลือกใช้ **Keyword : LEMON**
นำ Plaintext มาเรียงคู่กับ Keyword ให้ได้ความยาวเท่ากันดังนี้

Plain Text : ATTACK
Key : LEMONL
Cipher Text : LXFOPV

ตัวอักษรตัวแรก - A จะถูกเข้ารหัสด้วย Caesar Cipher Key L
ตัวอักษรตัวที่ 2 - T จะถูกเข้ารหัสด้วย Caesar Cipher Key E
ตัวอักษรตัวที่ 3 - T จะถูกเข้ารหัสด้วย Caesar Cipher Key M
และเรียงต่อไปเรื่อยๆ จนกว่าจะครบประโยค

A	B	C	A	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Plain Text	A	T	T	A	C	K
Key	L	E	M	O	N	L
Cipher Text	L	X	F	O	P	V

ตัวอย่างเช่น ต้องการเข้ารหัสตัว “A”
Cipher Text = P + K ; P=Plain Text, K=Key
Cipher Text = 0 + 11 = 11 ; กรณีบวกกันเกิน 25 ให้เริ่มนับ A=26, B=27, ...
Cipher Text = L

<u>การถอดรหัส</u>
วิธีการ : กระบวนการย้อนกลับเหมือนกับของ **Caesar** แต่ต้องรู้ **Keyword**

Ciphertext : LXFOPV และ Keyword : LEMON
นำ Cipher Text มาเรียงคู่กับ Keyword ให้ได้ความยาวเท่ากันดังนี้

Ciphertext : LXFOPV
Key : LEMONL
Plaintext : ATTACK

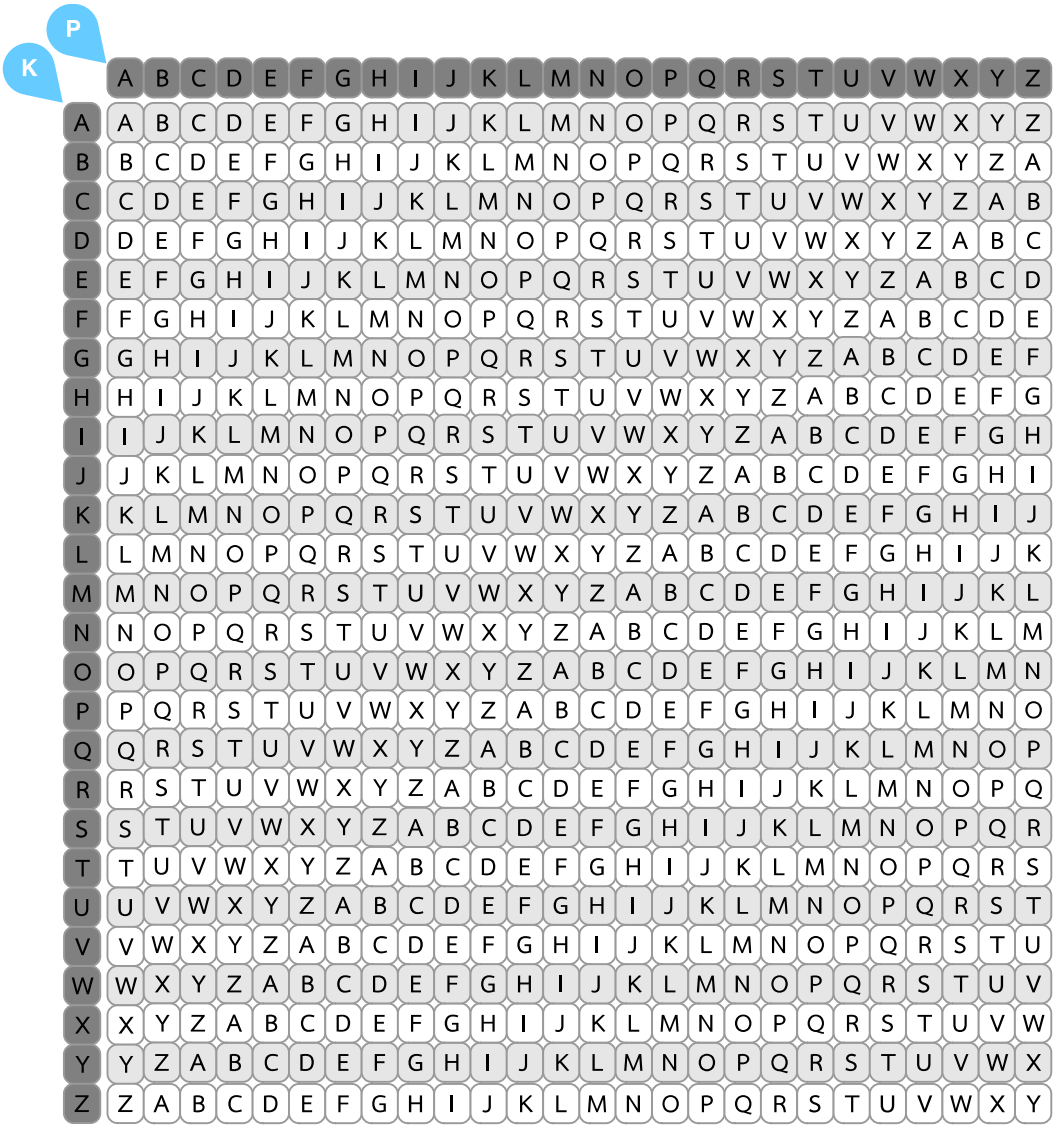
ตัวอักษรตัวแรก - L จะถูกถอดรหัสด้วย **Caesar Cipher Key L**
ตัวอักษรตัวที่ 2 - X จะถูกถอดรหัสด้วย **Caesar Cipher Key E**
ตัวอักษรตัวที่ 3 - F จะถูกถอดรหัสด้วย **Caesar Cipher Key M**
และเรียงต่อไปเรื่อยๆ จนกว่าจะครบประโยค

A	B	C	A	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Cipher Text	L	X	F	O	P	V
Key	L	E	M	O	N	L
Plain Text	A	T	T	A	C	K

ตัวอย่างเช่น ต้องการถอดรหัสตัว “L”
Plain Text = K - C ; K=Key, C=Cipher Text
Plain Text = 11 - 11 = 0 ; ถอดรหัส F จะได้ 12-31 = -19
Plain Text = A

Vigenere Table จากตัวอย่างข้างต้นหากมีการเข้ารหัสแล้วมีค่ามากกว่า 25 อาจเป็นผลให้เกิดข้อผิดพลาดในการถอดรหัส คือได้ Plain Text ที่ไม่ถูกต้อง ดังนั้น Vigenere Square Or Vigenere Table เป็นตารางที่จะนำมาทำการ Encrypt และ Decrypt อีกวิธีหนึ่งที่สามารถทำได้อย่างถูกต้อง โดยไม่สนใจว่าค่าตำแหน่งจะมากกว่า 25 หรือเป็นค่าเท่าไร เพราะวิธีนี้จะใช้การเปรียบเทียบจากตาราง



K \ P	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

ตัวอย่าง

ต้องการเข้ารหัสข้อความ NETWORK โดยใช้ Key คือ IPSEC จงแสดงวิธีการ Encrypt และ Decrypt โดยวิธีการ Vigenere Square

Plain Text: NETWORK

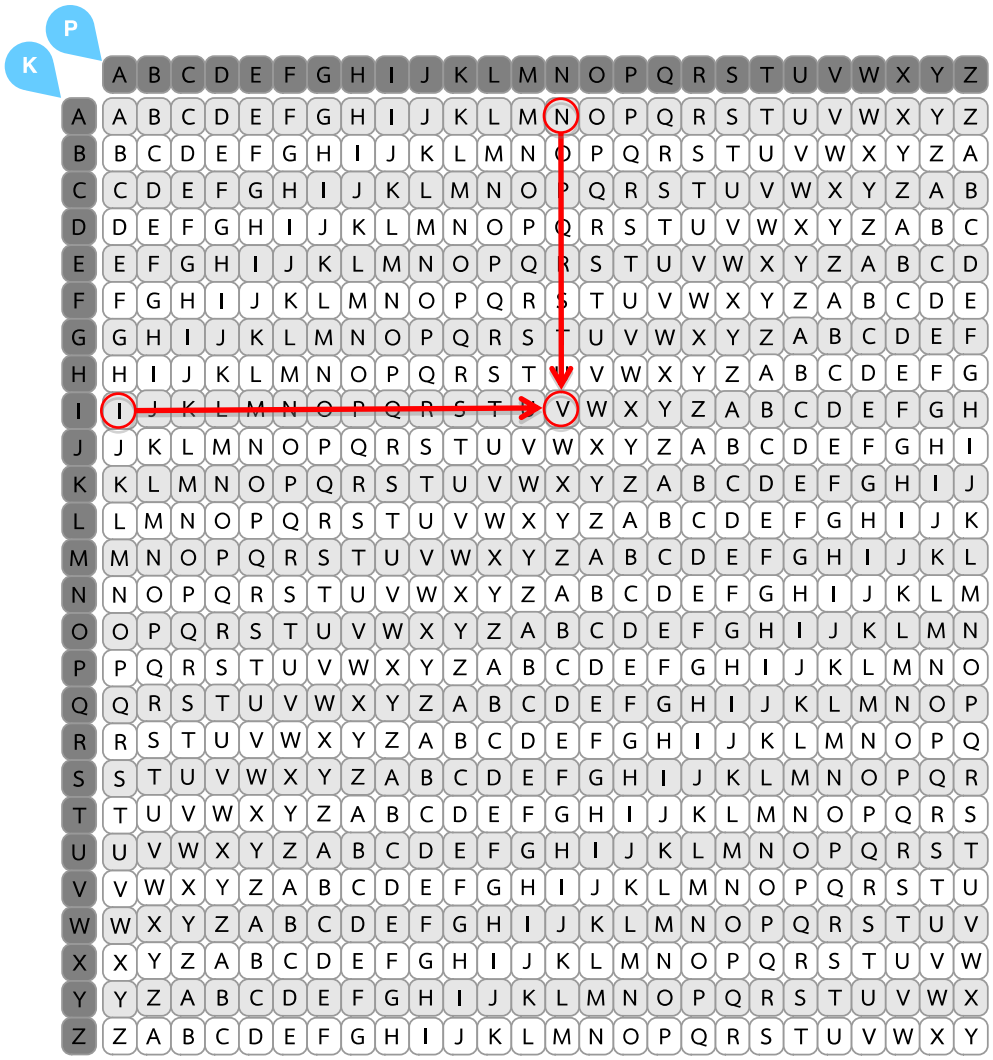
Key: IPSECIP

Cipher Text: VTLBQZZ

วิธีการเข้ารหัส

ทำการเข้ารหัส ตัวแรกคือ “N” ให้ดูตามแถวที่ตำแหน่ง “N”

ในแนวคอลัมน์ให้ดูไปที่ Key “I” หลังจากนั้นลากเส้นตรงทั้งสองมาตัดกันที่ตัวอักษรใด ก็จะได้ Cipher Text ออกมา คือ “V”



ทำลักษณะนี้ให้ครบทุกตัวก็จะได้ Cipher Text: VTLBQZZ

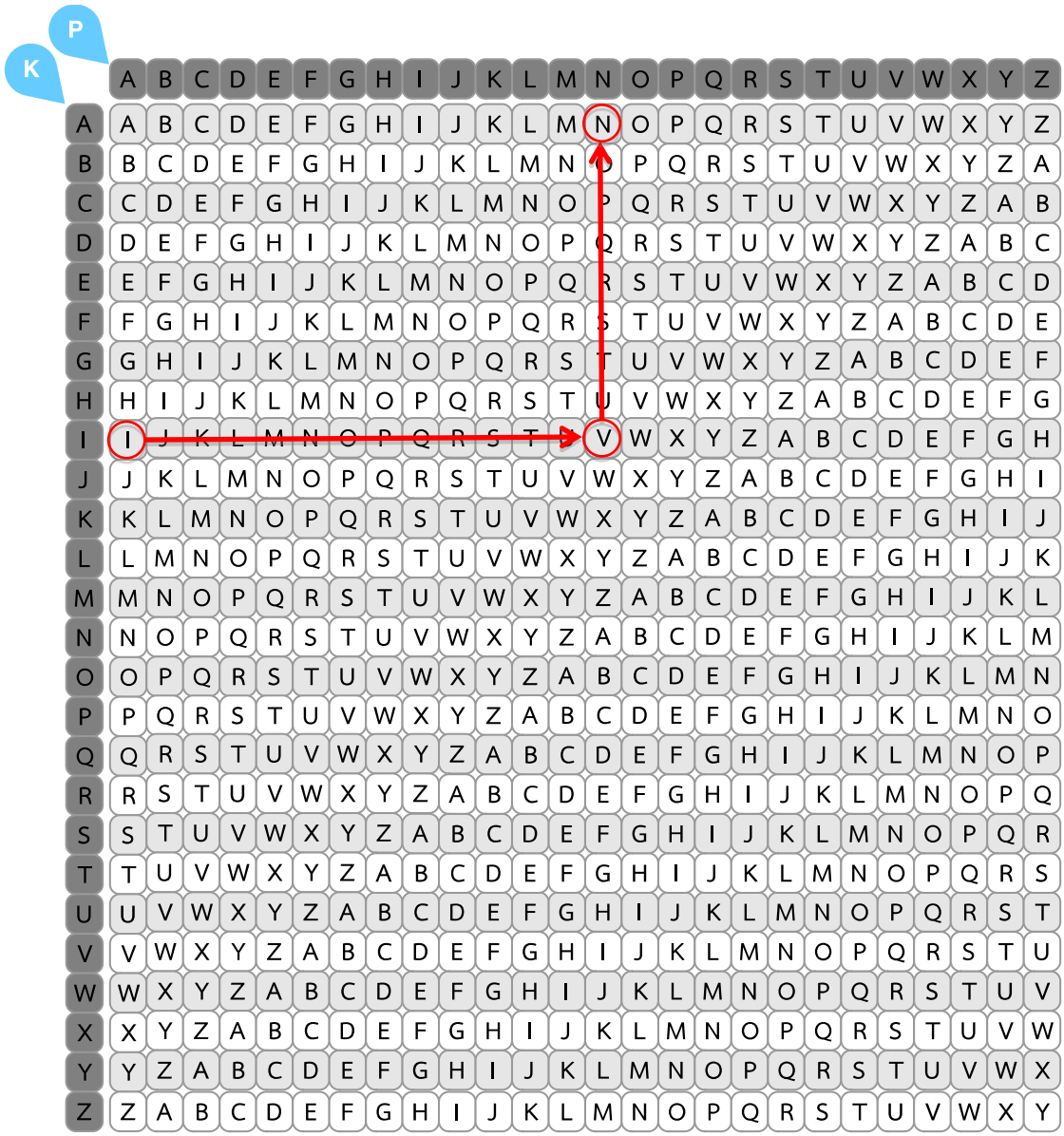
วิธีการถอดรหัส

Cipher Text: VTLBQZZ

Key: IPSECIP

Plain Text: NETWORK

ทำการถอดรหัสตัวแรก ในแนวคอลัมน์ Key “I” ลากเส้นตรงไปหา Cipher Text “V” แล้วลากเส้นตรงขึ้นไปบนสุด ตรงกับตัวอักษร “N” อักษรนั้นคือ Plain Text ที่ถอดรหัสนั่นเอง



อัลกอริทึมแบบสมมาตร (Symmetric Key Algorithms) อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้าและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อค (Block Algorithms) ซึ่งจะทำการเข้ารหัสทีละบล็อค (1 บล็อคประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำการเข้ารหัสทีละไบต์อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้าและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อค (Block Algorithms) ซึ่งจะทำการเข้ารหัสทีละบล็อค (1 บล็อคประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำการเข้ารหัสทีละไบต์

อัลกอริทึมแบบอสมมาตร (Asymmetric Key Algorithms) อัลกอริทึมนี้จะใช้กุญแจสองตัวเพื่อทำงาน ตัวหนึ่งใช้ในการเข้ารหัสและอีกตัวหนึ่งใช้ในการถอดรหัสข้อมูลที่เข้ารหัสมาโดยกุญแจตัวแรก อัลกอริทึมกลุ่มสำคัญในแบบอสมมาตรนี้คือ อัลกอริทึมแบบกุญแจสาธารณะ (Public keys Algorithms) ซึ่งใช้กุญแจที่เรียกกันว่า กุญแจสาธารณะ (Public keys) ในการเข้ารหัสและใช้กุญแจที่เรียกกันว่า กุญแจส่วนตัว (Private keys) ในการถอดรหัสข้อมูลนั้น กุญแจสาธารณะนี้สามารถส่งมอบให้กับผู้อื่นได้ เช่น เพื่อนร่วมงานที่เราต้องการติดต่อด้วย หรือแม้กระทั่งวางไว้บนเว็บไซต์เพื่อให้ผู้อื่นสามารถดาวน์โหลดไปใช้งานได้ สำหรับกุญแจส่วนตัวนั้นต้องเก็บไว้กับผู้เป็นเจ้าของกุญแจส่วนตัวเท่านั้นและห้ามเปิดเผยให้ผู้อื่นทราบโดยเด็ดขาด

อัลกอริทึมแบบกุญแจสาธารณะยังสามารถประยุกต์ใช้ได้กับการลงลายมือชื่ออิเล็กทรอนิกส์ (ซึ่งเปรียบเสมือนการลงลายมือชื่อของเราที่ใช้กับเอกสารสำนักงานทั่วไป) การลงลายมือชื่อนี้จะเป็นการพิสูจน์ความเป็นเจ้าของและสามารถใช้ได้กับการทำธุรกรรมต่างๆ บนอินเทอร์เน็ต เช่น การซื้อสินค้า เป็นต้น วิธีการใช้งานคือผู้เป็นเจ้าของกุญแจส่วนตัวลงลายมือชื่อของตนกับข้อความที่ต้องการส่งไปด้วยกุญแจส่วนตัว แล้วจึงส่งข้อความนั้นไปให้กับผู้รับ เมื่อได้รับข้อความที่ลงลายมือชื่อมา ผู้รับสามารถใช้กุญแจสาธารณะ (ที่เป็นคู่ของกุญแจส่วนตัวนั้น) เพื่อตรวจสอบว่าเป็นข้อความที่มาจากผู้ส่งนั้นหรือไม่

ปัญหาของอัลกอริทึมแบบสมมาตร

อัลกอริทึมแบบสมมาตรมีความสำคัญไม่ด้อยไปกว่าอัลกอริทึมแบบอสมมาตร ทั้งนี้เนื่องจากอัลกอริทึมแบบแรกทำงานได้รวดเร็วกว่าและง่ายต่อการใช้งานกว่าแบบหลัง อย่างไรก็ตามอัลกอริทึมแบบสมมาตรยังมีปัญหาที่สำคัญ 3 ประการ ซึ่งเป็นข้อจำกัดในการใช้งานอัลกอริทึมนี้

- ในการใช้งานอัลกอริทึมนี้ สองกลุ่มที่ต้องการแลกเปลี่ยนข้อมูลกัน (เช่น องค์กร ก และ ข) จำเป็นต้องแลกเปลี่ยนกุญแจลับกันก่อน (ซึ่งอาจหมายถึงส่งมอบกุญแจลับให้กับอีกกลุ่มหนึ่ง) การแลกเปลี่ยนกุญแจนั้นอาจทำได้อย่างยุ่งยากและไม่สะดวก
- ทั้งสองกลุ่มต้องรักษากุญแจลับนั้นไว้เป็นอย่างดี ห้ามเปิดเผยให้ผู้อื่นล่วงรู้โดยเด็ดขาด การที่กุญแจถูกเปิดเผยออกไปสู่ผู้อื่น (จะโดยกลุ่มใดกลุ่มหนึ่งก็ตาม) และอีกกลุ่มหนึ่งไม่ได้รับทราบปัญหานี้ อาจก่อให้เกิดปัญหากับกลุ่มที่ไม่ทราบนี้ได้ เช่น กลุ่มนี้อาจส่งข้อความที่เป็นความลับไปให้กับอีกกลุ่มหนึ่ง แต่ข้อความนี้อาจถูกเปิดเผยได้โดยใช้กุญแจลับที่ล่วงรู้โดยผู้อื่น
- สำหรับสองกลุ่มที่ต้องการติดต่อกัน จำเป็นต้องใช้กุญแจลับเป็นจำนวน 1 กุญแจเพื่อติดต่อกัน สมมติว่ามีผู้ที่ต้องติดต่อกันเป็นจำนวน n กลุ่ม จำนวนกุญแจลับทั้งหมดที่ต้องแลกเปลี่ยนกันคิดเป็นจำนวนทั้งหมด C2n หรือเท่ากับ n (n-1)/2 กุญแจ ซึ่งจะเห็นได้ว่าจำนวนกุญแจมีมากมายเกินไป ซึ่งอาจก่อให้เกิดปัญหาด้านการรักษาความปลอดภัยให้กับกุญแจเหล่านี้

อัลกอริทึมแบบกุญแจสาธารณะ (ซึ่งเป็นแบบอสมมาตร) ช่วยแก้ปัญหาเหล่านี้ได้ทั้งหมด ผู้ใช้ที่ถือกุญแจส่วนตัวและต้องการให้บุคคลอื่นที่ตนติดต่อด้วยส่งเอกสารหรือข้อความที่เข้ารหัสมาหาตน สามารถเผยแพร่กุญแจสาธารณะของตนไว้บนเว็บไซต์หรือในที่สาธารณะซึ่งผู้อื่นสามารถเข้ามาดาวน์โหลดไปใช้งานได้ วิธีการใช้งานคือให้

บุคคลอื่นที่มาดาวน์โหลดกุญแจไปนั้นทำการเข้ารหัสข้อความที่ต้องการส่งด้วยกุญแจสาธารณะ แล้วจึงส่งข้อความที่เข้ารหัสไปให้กับผู้เป็นเจ้าของกุญแจสาธารณะ โดยวิธีนี้จะไม่มีผู้อื่นสามารถเปิดดูข้อความที่เข้ารหัสนั้นได้ยกเว้นผู้ที่ถือกุญแจส่วนตัว (ที่เป็นคู่ของกุญแจสาธารณะนั้น) จึงจะสามารถเปิดข้อความนี้ดูได้

การเผยแพร่กุญแจสาธารณะในสถานที่ต่างๆ ได้ทำให้ลดความยุ่งยากในการแลกเปลี่ยนกุญแจกันซึ่งเป็นปัญหาข้อแรกของการเข้ารหัสแบบสมมาตร สำหรับปัญหาที่ว่าทั้งสองกลุ่มจะต้องรักษากุญแจลับไว้เป็นอย่างดีนั้น วิธีการของกุญแจสาธารณะจะทำให้ผู้ที่ต้องรับผิดชอบเหลือเพียงผู้เดียว กล่าวคือ ผู้ถือกุญแจส่วนตัว ซึ่งห้ามให้ผู้อื่นล่วงรู้โดยเด็ดขาด

สำหรับปัญหาที่สามที่ว่าจำนวนกุญแจลับที่จำเป็นต้องใช้มีมากมายเกินไป วิธีการของกุญแจสาธาณะจะใช้จำนวนกุญแจที่ประหยัดกว่า เนื่องจากกุญแจสาธารณะ 1 กุญแจของกลุ่มๆ หนึ่งจะสามารถเผยแพร่ให้กับกี่กลุ่มก็ได้ที่เราต้องการติดต่อด้วย (แทนที่จะเป็น 1 กุญแจลับต่อสองกลุ่มที่ต้องการติดต่อกัน) ดังนั้นถ้ามีกลุ่มที่ต้องติดต่อกันจำนวน n กลุ่ม จำนวนกุญแจส่วนตัวที่ต้องระวังรักษาก็คือ n กุญแจ ซึ่งจะเห็นได้ว่าลดลงไปได้เป็นจำนวนมาก

ข้อเสียที่สำคัญของระบบกุญแจสาธารณะที่สำคัญคือ ต้องใช้เวลาในการคำนวณการเข้าและถอดรหัส เมื่อเทียบกับระบบกุญแจสมมาตร และอาจใช้เวลาเป็นพันเท่าของเวลาที่ใช้โดยระบบกุญแจสมมาตร

ความแข็งแกร่งของอัลกอริทึมสำหรับการเข้ารหัส

ความแข็งแกร่งของอัลกอริทึมหมายถึงความยากในการที่ผู้บุกรุกจะสามารถถอดรหัสข้อมูลได้โดยปราศจากกุญแจที่ใช้ในการเข้ารหัส ซึ่งจะขึ้นอยู่กับปัจจัยดังนี้

- การเก็บกุญแจเข้ารหัสไว้อย่างเป็นความลับ ผู้เป็นเจ้าของกุญแจลับหรือส่วนตัวต้องระมัดระวังไม่ให้กุญแจสูญหายหรือล่วงรู้โดยผู้อื่น
- ความยาวของกุญแจเข้ารหัส ปกติกุญแจเข้ารหัสจะมีความยาวเป็นบิต ยิ่งจำนวนบิตของกุญแจยิ่งมากยิ่งทำให้การเดาเพื่อสุ่มหากุญแจที่ถูกต้องเป็นไปได้ยากยิ่งขึ้น (เช่น กุญแจขนาด 1 บิต จะสามารถแทนตัวเลขได้ 2 ค่าคือ 0 กับ 1 กุญแจขนาด 2 บิต จะเป็นไปได้ 4 ค่าคือ 0, 1, 2, 3 เป็นต้น)
- ความไม่เกรงกลัวต่อการศึกษาหรือดูอัลกอริทึมเพื่อหารูปแบบของการเข้ารหัส อัลกอริทึมที่ดีต้องเปิดให้ผู้รู้ทำการศึกษาในรายละเอียดได้โดยไม่เกรงว่าผู้ศึกษาจะสามารถจับรูปแบบของการเข้ารหัสได้
- การมีประตูลับในอัลกอริทึม อัลกอริทึมที่ดีต้องไม่แฝงไว้ด้วยประตูลับที่สามารถใช้เป็นทางเข้าไปสู่อัลกอริทึม แล้วอาจใช้เพื่อทำการถอดรหัสข้อมูลได้ ประตูลับนี้ทำให้ไม่จำเป็นต้องใช้กุญแจในการถอดรหัส
- ความไม่เกรงกลัวต่อปัญหาการหาความสัมพันธ์ในข้อมูลที่ได้รับ กล่าวคือเมื่อผู้บุกรุกทราบข้อมูลบางอย่างที่เป็นข้อมูลตั้งต้นซึ่งยังไม่ได้เข้ารหัส รวมทั้งมีข้อมูลที่เข้ารหัสแล้ว (ของข้อมูลตั้งต้นนั้น) ผู้บุกรุกอาจจะสามารถหาความสัมพันธ์ระหว่างข้อความทั้งสองนั้นได้ ซึ่งจะเป็นวิธีการในการถอดรหัส

ข้อมูลได้ ปัญหานี้เรียกกันว่า Known plaintext attack (คำว่า plaintext หมายถึงข้อความตั้งต้นที่ยังไม่ได้ผ่านการเข้ารหัส)

คุณสมบัติของข้อความตั้งต้น คุณสมบัตินี้อาจใช้เป็นช่องทางในการถอดรหัสข้อมูลได้ อัลกอริทึมที่ดีต้องไม่ใช้คุณสมบัติของข้อความเป็นกลไกในการเข้ารหัสข้อมูล

คำแนะนำในการเลือกใช้อัลกอริทึมคือให้ใช้อัลกอริทึมที่ได้มีการใช้งานมาเป็นระยะเวลานานแล้ว ทั้งนี้เนื่องจากหากปัญหาของอัลกอริทึมนี้มีจริง ก็คงเกิดขึ้นมานานแล้วและก็คงเป็นที่ทราบกันแล้ว นั่นคืออย่างน้อยที่สุดจวบจนกระทั่งถึงปัจจุบัน ก็ยังไม่มีการบุกรุกที่ทำให้อัลกอริทึมนั้นไม่สามารถใช้งานได้อย่างปลอดภัยเป็นที่ประจักษ์ ดังนั้นจึงไม่ควรใช้อัลกอริทึมใหม่ๆ ที่เพิ่งได้มีการนำเสนอกันสู่สาธารณะ เพราะอาจมีช่องโหว่แฝงอยู่และยังไม่เป็นที่ทราบในขณะนี้

ความยาวของกุญแจที่ใช้ในการเข้ารหัส

ความยาวของกุญแจเข้ารหัสมีหน่วยนับเป็นบิต หนึ่งบิตในคอมพิวเตอร์เป็นตัวเลขฐานสองที่ประกอบด้วยค่า 0 และ 1 กุญแจที่มีความยาว 1 บิต ตัวเลขที่เป็นไปได้เพื่อแทนกุญแจนั้น จึงอาจมีค่าเป็น 0 หรือ 1 กุญแจที่มีความยาว 2 บิต ตัวเลขที่เป็นไปได้จึงเป็น 0, 1, 2 และ 3 ตามลำดับ กุญแจที่มีความยาว 3 บิต ตัวเลขที่เป็นไปได้จะอยู่ระหว่าง 0 ถึง 7 ดังนั้นเมื่อเพิ่มความยาวของกุญแจทุกๆ 1 บิต ค่าที่เป็นไปได้ของกุญแจจะเพิ่มขึ้นเป็นสองเท่าตัว หรือจำนวนกุญแจที่เป็นไปได้จะเพิ่มขึ้นเป็น 2 เท่าตัวนั่นเอง

ฉะนั้นจะเห็นได้ว่ากุญแจยิ่งมีความยาวมาก โอกาสที่ผู้บุกรุกจะสามารถคาดเดากุญแจที่ตรงกับหมายเลขที่ถูกต้องของกุญแจจะยิ่งยากมากขึ้นตามลำดับ ในการที่ผู้บุกรุกลองผิดลองถูกกับกุญแจโดยใช้กุญแจที่มีหมายเลขต่างๆ กัน เพื่อหวังที่จะพบกุญแจที่ถูกต้องและสามารถใช้ถอดรหัสข้อมูลได้ การลองผิดลองถูกนี้เราเรียกกันว่า Key search หรือการค้นหากุญแจนั่นเอง ทฤษฎีได้กล่าวไว้ว่าการลองผิดลองถูกนี้โดยเฉลี่ยจะต้องทดลองกับกุญแจเป็นจำนวนครึ่งหนึ่งของกุญแจทั้งหมดก่อนที่จะพบกุญแจที่ถูกต้อง

ความยาวของกุญแจที่มีขนาดเหมาะสมจึงขึ้นอยู่กับความเร็วในการค้นหากุญแจของผู้บุกรุกและระยะเวลาที่ต้องการให้ข้อมูลมีความปลอดภัย ตัวอย่างเช่น ถ้าผู้บุกรุกสามารถลองผิดลองถูกกับกุญแจเป็นจำนวน 10 กุญแจภายในหนึ่งวินาทีแล้ว กุญแจที่มีความยาว 40 บิต จะสามารถป้องกันข้อมูลไว้ได้ 3,484 ปี ถ้าผู้บุกรุกสามารถลองได้เป็นจำนวน 1 ล้านกุญแจในหนึ่งวินาที (เทคโนโลยีปัจจุบันสามารถทำได้) กุญแจที่มีความยาว 40 บิตจะสามารถป้องกันข้อมูลไว้ได้เพียง 13 วันเท่านั้น (ซึ่งอาจไม่เพียงพอสำหรับในบางลักษณะงาน) ด้วยเทคโนโลยีในปัจจุบันหากผู้บุกรุกสามารถทดลองได้เป็นจำนวน 1,000 ล้านกุญแจในหนึ่งวินาที กุญแจขนาด 128 บิตจะสามารถป้องกันข้อมูลไว้ได้ 1022 ปี ดังนั้นด้วยลักษณะงานทั่วไปกุญแจขนาด 128 บิตจะพอเพียงต่อการรักษาความลับของข้อมูลเอาไว้ได้

อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร

อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตรในปัจจุบันมีเป็นจำนวนมาก ข้างล่างนี้จะนำเสนอเพียงจำนวนหนึ่งซึ่งเป็นอัลกอริทึมที่เป็นที่รู้จักกันดีในวงการของการเข้ารหัสข้อมูล

อัลกอริทึม DES

DES ย่อมาจาก Data Encryption Standard อัลกอริทึมนี้ได้รับการรับรองโดยรัฐบาลสหรัฐอเมริกาในปี ค.ศ. 1977 ให้เป็นมาตรฐานการเข้ารหัสข้อมูลสำหรับหน่วยงานของรัฐทั้งหมด ในปี 1981 อัลกอริทึมยังได้รับการกำหนดให้เป็นมาตรฐานการเข้ารหัสข้อมูลในระดับนานาชาติตามมาตรฐาน ANSI (American National Standards) อีกด้วย

DES เป็นอัลกอริทึมแบบบล็อกซึ่งใช้กุญแจที่มีขนาดความยาว 56 บิตและเป็นอัลกอริทึมที่มีความแข็งแกร่ง แต่เนื่องด้วยขนาดความยาวของกุญแจที่มีขนาดเพียง 56 บิต ซึ่งในปัจจุบันถือได้ว่าสั้นเกินไป ผู้บุกรุกอาจใช้วิธีการลองผิดลองถูกเพื่อค้นหากุญแจที่ถูกต้องสำหรับการถอดรหัสได้

ในปี 1998 ได้มีการสร้างเครื่องคอมพิวเตอร์พิเศษขึ้นมาซึ่งมีมูลค่า 250,000 เหรียญสหรัฐ เพื่อใช้ในการค้นหากุญแจที่ถูกต้องของการเข้ารหัสข้อมูลหนึ่งๆ ด้วย DES และพบว่าเครื่องคอมพิวเตอร์นี้สามารถค้นหากุญแจที่ถูกต้องได้ภายในระยะเวลาไม่ถึงหนึ่งวัน

อัลกอริทึม Triple-DES

Triple-DES เป็นอัลกอริทึมที่เสริมความปลอดภัยของ DES ให้มีความแข็งแกร่งมากขึ้นโดยใช้อัลกอริทึม DES เป็นจำนวนสามครั้งเพื่อทำการเข้ารหัส แต่ละครั้งจะใช้กุญแจในการเข้ารหัสที่แตกต่างกัน ดังนั้นจึงเปรียบเสมือนการใช้กุญแจเข้ารหัสที่มีความยาวเท่ากับ 56*3 = 168 บิต Triple-DES ได้ถูก ใช้งานกับสถาบันทางการเงินอย่างแพร่หลาย รวมทั้งใช้งานกับโปรแกรม Secure Shell (ssh) ด้วย

การใช้อัลกอริทึม DES เพื่อเข้ารหัสเป็นจำนวนสองครั้งด้วยกุญแจสองตัว (56*2=112 บิต) ยังถือได้ว่าไม่ปลอดภัยอย่างพอเพียง

อัลกอริทึม Blowfish

Blowfish เป็นอัลกอริทึมที่มีความรวดเร็วในการทำงาน มีขนาดเล็กกระทัดรัด และใช้การเข้ารหัสแบบบล็อค ผู้พัฒนาคือ Bruce Schneier อัลกอริทึมสามารถใช้กุญแจที่มีขนาดความยาวตั้งแต่ไม่มากนักไปจนถึงขนาด 448 บิต ซึ่งทำให้เกิดความยืดหยุ่นสูงในการเลือกใช้กุญแจ รวมทั้งอัลกอริทึมยังได้รับการออกแบบมาให้ทำงานอย่างเหมาะสมกับหน่วยประมวลผลขนาด 32 หรือ 64 บิต Blowfish ได้เปิดเผยสู่สาธารณะและไม่ได้มีการจดสิทธิบัตรใดๆ นอกจากนั้นยังใช้ในโปรแกรม SSH และอื่นๆ

อัลกอริทึม IDEA

IDEA ย่อมาจาก International Data Encryption Algorithm อัลกอริทึมนี้ได้รับการพัฒนาในประเทศสวิสเซอร์แลนด์ที่เมือง Zarich โดย James L. Massey และ Xuejia Lai และได้รับการตีพิมพ์เผยแพร่ในปี ค.ศ. 1990 อัลกอริทึมใช้กุญแจที่มีขนาด 128 บิต และได้รับการใช้งานกับโปรแกรมยอดฮิตสำหรับการเข้ารหัสและลงลายมือชื่ออิเล็กทรอนิกส์ในระบบอีเมล์ที่มีชื่อว่า PGP ต่อมา IDEA ได้รับการจดสิทธิบัตรทางด้านซอฟต์แวร์โดยบริษัท Ascom-Tech AG ในประเทศสวิสเซอร์แลนด์ ซึ่งทำให้การนำไปใช้ในงานต่างๆ เริ่มลดลง ทั้งนี้เนื่องจากติดปัญหาเรื่องลิขสิทธิ์นั่นเอง

อัลกอริทึม RC4

อัลกอริทึมนี้เป็นอัลกอริทึมแบบสตรีม (ทำงานกับข้อมูลทีละไบต์) ซึ่งได้รับการพัฒนาขึ้นมาโดย Ronald Riverst และถูกเก็บเป็นความลับทางการค้าโดยบริษัท RSA Data Security ในภายหลังอัลกอริทึมนี้ได้รับการเปิดเผยใน Usenet เมื่อปี ค.ศ. 1994 และเป็นที่ทราบกันว่าเป็นอัลกอริทึมที่มีความแข็งแกร่งโดยสามารถใช้ขนาดความยาวของกุญแจที่มีขนาดตั้งแต่ 1 บิตไปจนกระทั่งถึงขนาด 2048 บิต

อัลกอริทึม Rijndael (หรืออัลกอริทึม AES)

อัลกอริทึมนี้ได้รับการพัฒนาโดย Joan Daemen และ Vincent Rijmen ในปี 2000 อัลกอริทึมได้รับการคัดเลือกโดยหน่วยงาน National Institute of Standard and Technology (NIST) ของสหรัฐอเมริกาให้เป็นมาตรฐานในการเข้ารหัสชั้นสูงของประเทศ อัลกอริทึมมีความเร็วสูงและมีขนาดกะทัดรัดโดยสามารถใช้กุญแจที่มีความยาวขนาด 128, 192 และ 256 บิต

อัลกอริทึม One-time Pads

อัลกอริทึมนี้ได้รับการยอมรับว่าเป็นอัลกอริทึมที่ไม่มีใครสามารถเจาะความแข็งแกร่งของอัลกอริทึมได้ อัลกอริทึมใช้กุญแจที่มีขนาดความยาวซึ่งอาจจะมากกว่าขนาดความยาวของข้อความที่ต้องการเข้ารหัส กุญแจจะถูกสร้างออกมาแบบสุ่มและโดยปกติจะถูกใช้งานแค่เพียงครั้งเดียวแล้วทิ้งไป แต่ละไบต์ของข้อความที่ต้องการส่งไปจะถูกเข้าและถอดรหัสโดยหนึ่งไบต์ (ชนิดไบต์ต่อไบต์) ของกุญแจที่ถูกสร้างขึ้นมาใช้งาน เนื่องจากกุญแจที่ถูกใช้งานแต่ละครั้งจะไม่ซ้ำกันและถูกสร้างขึ้นมาแบบสุ่ม จึงเป็นการยากที่จะค้นหากุญแจที่ถูกต้องได้

ข้อจำกัดของอัลกอริทึมนี้ คือขนาดของกุญแจที่อาจมีขนาดยาวกว่าข้อความที่ต้องการส่ง ซึ่งส่งผลให้การส่งมอบกุญแจที่มีขนาดใหญ่ทำได้ไม่สะดวกนัก รวมทั้งการสร้างกุญแจให้มีความสุ่มสูงไม่ใช่เป็นสิ่งที่ทำได้ง่ายนัก อย่างไรก็ตามอัลกอริทึมนี้ก็ยังมีการใช้งานในระบบเครือข่ายที่ต้องการความปลอดภัยสูง

อัลกอริทึมสำหรับการเข้ารหัสแบบกุญแจสาธารณะ (หรือการเข้ารหัสแบบอสมมาตร)

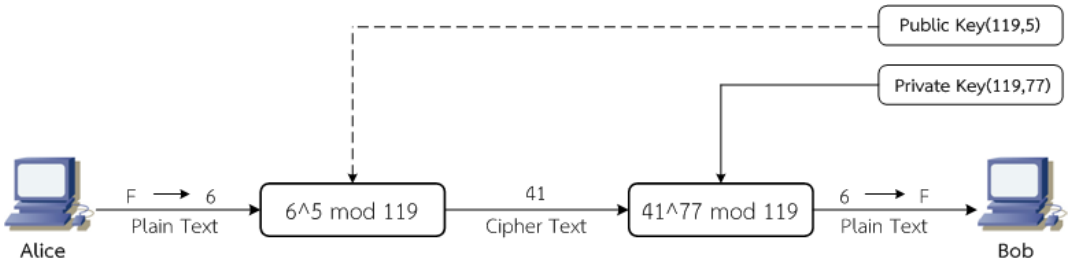
อัลกอริทึมแบบกุญแจสาธารณะแบ่งตามลักษณะการใช้งานได้เป็น 2 ประเภทที่รู้จักกันดีมีดังนี้คือ

- ใช้สำหรับการเข้ารหัส
- ใช้สำหรับการลงลายมือชื่ออิเล็กทรอนิกส์

อัลกอริทึม RSA

อัลกอริทึม RSA ได้รับการพัฒนาขึ้นที่มหาวิทยาลัย MIT ในปี 1977 โดยศาสตราจารย์ 3 คน ซึ่งประกอบด้วย Ronald Rivest, Adi Shamir และ Leonard Adleman ชื่อของอัลกอริทึมได้รับการตั้งชื่อตามตัวอักษรตัวแรกของนามสกุลของศาสตราจารย์ทั้งสามคน อัลกอริทึมนี้สามารถใช้ในการเข้ารหัสข้อมูลรวมทั้งการลงลายมือชื่ออิเล็กทรอนิกส์ด้วย

วิธีการนี้สามารถใช้ได้ทั้งกับการเข้ารหัสข้อมูลและลายเซ็นต์ดิจิตอล ข้อมูลที่เข้ารหัสด้วยไพรเวทคีย์ จะถูกถอดรหัสได้โดยใช้พับลิกคีย์ที่เป็นคู่กันเท่านั้น เช่น ถ้าอลิสใช้ไพรเวทคีย์ของตัวเองในการเข้ารหัส ใครก็ตามที่มีพับลิกคีย์ของเธอก็สามารถถอดรหัสข้อมูลนั้นได้(แสดงดังรูป) แสดงหลักฐานการเข้าและถอดรหัสแบบพับลิกคีย์เอนคริพชัน การสื่อสารแบบนี้จะสร้างความเชื่อมั่นในข้อมูลแบบทางเดียว RSA สามารถใช้ประโยชน์ได้หลายด้าน เช่น การเข้ารหัสข้อมูล และการแจกจ่ายซีเครทคีย์ก็ได้



รูปแสดง การเข้ารหัสแบบพับลิกคีย์แบบ RSA

จากรูป RSA Encrypt & Decrypt สมมติให้คู่ลำดับของกุญแจส่วนตัวเป็น (119,77) และคู่ลำดับของกุญแจสาธารณะเป็น (119,5) เมื่อผู้ส่งต้องการที่จะส่งตัวอักษร F ซึ่งตัวอักษรนี้จะสามารถแทนด้วยตัวเลข 6 (F เป็นพยัญชนะตัวที่ 6) ในการเข้ารหัสข้อมูลสามารถคำนวณได้ว่า C = 6^5 mod 119 = 41 ผลลัพท์ที่ได้มานี้จะเป็นตัวเลขที่ผ่านการเข้ารหัสข้อมูลแล้ว เมื่อผู้รับได้รับตัวเลขนี้แล้ว จะทำการถอดรหัสข้อมูลซึ่งสามารถคำนวณได้ดังนี้ M = 41^77 mod 119 = 6 ซึ่งตัวเลขนี้จะหมายถึงตัวอักษร F นั่นเอง

ขั้นตอนการคำนวณของ RSA

- พื้นฐานเกี่ยวกับ จำนวนจำเพาะและการ Mod

- ขั้นตอนในการเลือกพับลิกคีย์และไพรเวทคีย์
- การเข้าและถอดรหัส

<u>พื้นฐานเกี่ยวกับ จำนวนจำเพาะและการ Mod</u>

ตัวประกอบ หมายถึง จำนวนนับที่สามารถหารจำนวนนับได้ลงตัว เช่น
ตัวประกอบของ 3 คือ 1 และ 3
ตัวประกอบของ 15 คือ 1, 3, 5 และ 15

จำนวนจำเพาะ หมายถึง จำนวนนับที่มีตัวประกอบเพียง 2 ตัว คือ 1 และตัวของมันเอง เช่น 2, 3, 5, 7, 11.. เป็นต้น

Mod หมายถึง การหารเอาเฉพาะเศษ เช่น 10 mod 3 ได้ 1

หารร่วมมาก (ห.ร.ม.) ตัวหารร่วมที่มากทีสุด (ห.ร.ม.) ตัวหารร่วมที่มากที่สุดของจำนวนใดๆ ตั้งแต่ 2 จำนวนขึ้นไปหมายถึง จำนวนที่มีค่ามากที่สุดที่สามารถหารจำนวนทั้งหมดเหล่านั้นได้ลงตัว
- วิธีการหา ห.ร.ม. โดยการแยกตัวประกอบ มีวิธีการดังนี้ แยกตัวประกอบของจำนวนทุกจำนวนที่ต้องการหาร ห.ร.ม. เลือกตัวประกอบที่ซ้ำกันของทุกจำนวนมาคูณกัน ห.ร.ม. คือ ผลคูณที่ได้
- ตัวอย่าง จงหา ห.ร.ม. ของ 56 , 84 และ 140 วิธีทำ 56 = 2 * 2 * 2 * 7, 84 = 2 * 2 * 3 * 7, 104 = 2 * 2 * 5 * 7 เลือกตัวที่ซ้ำกัน ที่อยู่ทั้ง 56, 84 และ 104 ตัวที่ซ้ำกันเอามาซ้ำละ 1 ตัว คือ มีเลข 2 เลข 2 และ เลข 7 ดังนั้น ห.ร.ม. คือ 2 * 2 * 7 = 28

<u>ขั้นตอนในการเลือกพับลิกคีย์และไพรเวทคีย์</u>

ในการเลือกคู่ลำดับของเลขไม่ว่าจะเป็น n,d และ e นั้นจะสามารถใช้ขั้นตอนต่างๆต่อไปนี้

1. เลือกตัวเลขจำนวนเฉพาะ (Prime Number) ที่มีค่ามากๆ คือ p และ q
2. คำนวณโดยให้ n=p*q
3. ให้ m=(p-1)(q-1)
4. เลือกค่า e (มีค่าน้อยกว่า n) โดยที่ e และ m เป็นจำนวนที่ไม่มีตัวประกอบร่วมกัน
5. เลือกค่า d โดยที่ (e*d) mod m = 1

<u>ตัวอย่าง</u>

(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

แทนค่า 5.d mod 96 =1

วิธีการโดยนำ (96+1)/5 ว่าลงตัวหรือไม่ หากไม่ลงตัวทวีคูณอีกเท่าตัวและนำ 5 มาหารอีกจะได้ (96+96+1)/5 ทำไปเรื่อยๆจนมาหยุดที่ 96 บวกกัน 4 ครั้ง ได้ (96+96+96+96+1)/5 ซึ่งจะได้ค่า d=77 เพราะ 5*77 mod 96 ได้ 1

(6) Public key คือ (n,e) ดังนั้น Public key คือ (119,5)
(7) Private key คือ (n,d) ดังนั้น Private key คือ (119,77)

<u>การเข้าและถอดรหัส</u>

(1) ให้ M คือข้อความที่ยังไม่เข้ารหัส (ในรูปแบบของตัวเลข) M < n
ให้ข้อความที่ยังไม่เข้ารหัส M = 19

(2) การเข้ารหัส => C = M^e mod n
ได้ C = 19^5 mod 119 = 66

(3) การถอดรหัส => M = C^d mod n
ได้ M = 66^77 mod 119 = 19

อัลกอริทึม DSS

DSS ย่อมาจาก Digital Signature Standard อัลกอริทึมนี้ได้รับการพัฒนาขึ้นมาโดย National Security Agency ในประเทศสหรัฐอเมริกาและได้รับการรับรองโดย NIST ให้เป็นมาตรฐานกลางสำหรับการลงลายมือชื่ออิเล็กทรอนิกส์ในประเทศสหรัฐอเมริกา

อัลกอริทึมสำหรับสร้างเมสเซสไดเจสต์ (Hash Function)

เมสเซสไดเจสต์ (Message Digest) หรือเรียกสั้นๆ ว่าไดเจสต์ แปลว่าข้อความสรุปจากเนื้อหาข้อความตั้งต้น โดยปกติข้อความสรุปจะมีความยาวน้อยกว่าความยาวของข้อความตั้งต้นมาก จุดประสงค์สำคัญของอัลกอริทึมนี้คือ การสร้างข้อความสรุปที่สามารถใช้เป็นตัวแทนของข้อความตั้งต้นได้ โดยทั่วไปข้อความสรุปจะมีความยาวอยู่ระหว่าง 128 ถึง 256 บิต และจะไม่ขึ้นกับขนาดความยาวของข้อความตั้งต้น

<u>คุณสมบัติที่สำคัญของอัลกอริทึมสำหรับสร้างไดเจสต์มีดังนี้</u>

1. ทุกๆ บิตของไดเจสต์จะขึ้นอยู่กับทุกบิตของข้อความตั้งต้น
2. ถ้าบิตใดบิตหนึ่งของข้อความตั้งต้นเกิดการเปลี่ยนแปลง เช่น ถูกแก้ไข ทุกๆ บิตของไดเจสต์จะมีโอกาสร้อยละ 50 ที่จะแปรเปลี่ยนค่าไปด้วย ซึ่งหมายถึงว่า 0 เปลี่ยนค่าเป็น 1 และ 1 เปลี่ยนเป็น 0 คุณสมบัติข้อนี้สามารถอธิบายได้ว่าการเปลี่ยนแปลงแก้ไขข้อความตั้งต้นโดยผู้ไม่ประสงค์ดีแม้ว่าอาจแก้ไขเพียงเล็กน้อยก็ตาม เช่น เพียง 1 บิตเท่านั้น ก็จะส่งผลให้ผู้รับข้อความทราบว่าข้อความที่ตนได้รับไม่ใช่ข้อความตั้งต้น (โดยการนำข้อความที่ตนได้รับเข้าอัลกอริทึมเพื่อทำการคำนวณหาไดเจสต์ออกมา แล้วจึงเปรียบเทียบไดเจสต์ที่คำนวณได้กับไดเจสต์ที่ส่งมาให้ด้วย ถ้าต่างกัน แสดงว่าข้อความที่ได้รับนั้นถูกเปลี่ยนแปลงแก้ไข)
3. โอกาสที่ข้อความตั้งต้น 2 ข้อความใดๆ ที่มีความแตกต่างกัน จะสามารถคำนวณได้ค่าไดเจสต์เดียวกันมีโอกาสน้อยมากคุณสมบัติข้อนี้ทำให้แน่ใจได้ว่า เมื่อผู้ไม่ประสงค์ดีทำการแก้ไขข้อความตั้งต้น ผู้รับข้อความที่ถูกแก้ไขไปแล้วนั้นจะสามารถตรวจพบได้ถึงความผิดปกติที่เกิดขึ้นอย่างแน่นอน

อย่างไรก็ตามในทางทฤษฎีแล้ว มีโอกาสที่ข้อความ 2 ข้อความที่แตกต่างกันจะสามารถคำนวณแล้วได้ค่าไดเจสต์เดียวกัน ปัญหานี้เรียกกันว่าการชนกันของไดเจสต์(Collision) อัลกอริทึมสำหรับสร้างไดเจสต์ที่ดีควรจะมีโอกาสน้อยมากๆ ที่จะก่อให้เกิดปัญหาการชนกันของไดเจสต์

อัลกอริทึมสำหรับสร้างไดเจสต์ยอดนิยมมีดังนี้

อัลกอริทึม MD2

ผู้พัฒนาคือ Ronald Rivest อัลกอริทึมนี้เชื่อกันว่ามีความแข็งแกร่งที่สุดในบรรดาอัลกอริทึมต่างๆ ที่ Rivest พัฒนาขึ้นมา (ความแข็งแกร่งพิจารณาได้จากคุณสมบัติสามประการข้างต้น) ข้อเสียของอัลกอริทึมนี้คือใช้เวลามากในการคำนวณไดเจสต์หนึ่งๆ MD2 จึงไม่ค่อยได้มีการใช้งานกันมากนัก MD2 สร้างไดเจสต์ที่มีความยาว 128 บิต

อัลกอริทึม MD4

ผู้พัฒนาคือ Rivest เช่นเดียวกับ MD2 อัลกอริทึมนี้พัฒนาขึ้นมาเพื่อแก้ปัญหาความล่าช้าในการคำนวณของ MD2 อย่างไรก็ตามในภายหลังได้พบว่าอัลกอริทึมมีข้อบกพร่องที่เกี่ยวข้องกับคุณสมบัติข้อที่สามโดยตรง กล่าวคือปัญหาการชนกันของไดเจสต์มีโอกาสเกิดขึ้นได้ไม่น้อย ซึ่งผู้บุกรุกอาจใช้ประโยชน์จากจุดอ่อนนี้เพื่อทำการแก้ไขข้อความตั้งต้นที่ส่งมาให้ได้ MD4 ผลิตไดเจสต์ที่มีขนาด 128 บิต

อัลกอริทึม MD5

Rivest เป็นผู้พัฒนาเช่นกันโดยพัฒนาต่อจาก MD4 เพื่อให้มีความปลอดภัยที่สูงขึ้น ถึงแม้จะเป็นที่นิยมใช้งานกันอย่างแพร่หลาย ทว่าในปี 1996 ก็มีผู้พบจุดบกพร่องของ MD5 (เช่นเดียวกับ MD4) จึงทำให้ความนิยมเริ่มลดลงMD5 ผลิตไดเจสต์ที่มีขนาด 128 บิต

อัลกอริทึม SHA

SHA ย่อจาก Secure Hash Algorithm อัลกอริทึม SHA ได้รับแนวคิดในการพัฒนามาจาก MD4 และได้รับการพัฒนาขึ้นมาเพื่อใช้งานร่วมกับอัลกอริทึม DSS (ซึ่งใช้ในการลงลายมือชื่ออิเล็กทรอนิกส์) หลังจากที่ได้มีการตีพิมพ์เผยแพร่อัลกอริทึมนี้ได้ไม่นาน NIST ก็ประกาศตามมาว่าอัลกอริทึมจำเป็นต้องได้รับการแก้ไขเพิ่มเติมเล็กน้อยเพื่อให้สามารถใช้งานได้อย่างเหมาะสม SHA สร้างไดเจสต์ที่มีขนาด 160 บิต

อัลกอริทึม SHA-1

SHA-1 เป็นอัลกอริทึมที่แก้ไขเพิ่มเติมเล็กน้อยจาก SHA การแก้ไขเพิ่มเติมนี้เป็นที่เชื่อกันว่าทำให้อัลกอริทึม SHA-1 มีความปลอดภัยที่สูงขึ้นSHA-1 สร้างไดเจสต์ที่มีขนาด 160 บิต

อัลกอริทึม SHA-256, SHA-384 และ SHA-512

NIST เป็นผู้นำเสนออัลกอริทึมทั้งสามนี้ในปี 2001 เพื่อใช้งานร่วมกับอัลกอริทึม AES (ซึ่งเป็นอัลกอริทึมในการเข้ารหัสแบบสมมาตร) อัลกอริทึมเหล่านี้สร้างไดเจสต์ที่มีขนาด 256, 384 และ 512 บิต ตามลำดับ

นอกจากอัลกอริทึมสำหรับการสร้างไดเจสต์ที่กล่าวถึงไปแล้วนั้น อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร เช่น DES สามารถใช้ในการสร้างไดเจสต์เช่นกัน วิธีการใช้งานอัลกอริทึมแบบสมมาตรเพื่อสร้างไดเจสต์คือให้เลือกกุญแจลับสำหรับการเข้ารหัสขึ้นมา 1 กุญแจโดยวิธีการเลือกแบบสุ่ม และต่อมาใช้กุญแจนี้เพื่อเข้ารหัสข้อความตั้งต้น แล้วใช้เฉพาะบล็อกสุดท้ายที่เข้ารหัสแล้วเพื่อเป็นไดเจสต์ของข้อความทั้งหมด (ไม่รวมบล็อคอื่นๆ ที่เข้ารหัสแล้ว) อัลกอริทึมแบบสมมาตรสามารถสร้างไดเจสต์ที่มีคุณภาพดี แต่ข้อเสียคือต้องใช้เวลาในการคำนวณไดเจสต์มาก

ไดเจสต์เป็นเครื่องมือที่สำคัญที่สามารถใช้ในการตรวจสอบว่าไฟล์ในระบบที่ใช้งานมีการเปลี่ยนแปลงแก้ไขหรือไม่ (ไม่ว่าจะโดยเจตนาหรือไม่ก็ตาม) บางครั้งการเปลี่ยนแปลงแก้ไขอาจถูกกระทำโดยผู้ที่ไม่มีสิทธิ์ เช่น ผู้บุกรุก เป็นต้น วิธีการใช้ไดเจสต์เพื่อตรวจสอบไฟล์ในระบบคือให้เลือกใช้อัลกอริทึมหนึ่ง เช่น MD5 เพื่อสร้างไดเจสต์ของไฟล์ในระบบและเก็บไดเจสต์นั้นไว้อีกที่หนึ่งนอกระบบ ภายหลังจากระยะเวลาหนึ่งที่กำหนดไว้ เช่น 1 เดือน ก็มาคำนวณไดเจสต์ของไฟล์เดิมอีกครั้งหนึ่ง แล้วเปรียบเทียบไดเจสต์ใหม่นี้กับไดเจสต์ที่เก็บไว้นอกระบบว่าตรงกันหรือไม่ ถ้าตรงกัน ก็แสดงว่าไฟล์ในระบบยังเป็นปกติเช่นเดิม

ไดเจสต์ยังเป็นส่วนหนึ่งของการลงลายมือชื่ออิเล็กทรอนิกส์ กล่าวคือการลงลายมือชื่ออิเล็กทรอนิกส์ในปัจจุบันจะใช้การลงลายมือชื่อกับไดเจสต์ของข้อความตั้งต้นแทนการลงลายมือชื่อกับข้อความตั้งต้นทั้งข้อความ

บทสรุป

ความรู้พื้นฐานสำหรับการเข้ารหัสข้อมูล โดยกล่าวถึงจุดประสงค์ 3 ประการของการเข้ารหัสข้อมูล อัลกอริทึมในการเข้ารหัสแบบสมมาตรและอสมมาตรในแบบต่างๆ ที่แพร่หลายอยู่ในปัจจุบัน การพิจารณาความแข็งแกร่งหรือความปลอดภัยของอัลกอริทึมที่เลือกใช้งาน รวมทั้งอัลกอริทึมสำหรับการสร้างเมสเซสไดเจสต์ ซึ่งวิทยาการเข้ารหัสจะช่วยเพิ่มความลับของข้อมูลและทำให้การสื่อสารข้อมูลในระบบเครือข่ายมีความปลอดภัยมากยิ่งขึ้นด้วย