บทที่ 10
ขั้นตอนวิธีทฤษฎีจำนวน
Number-Theoretic Algorithms

ทฤษฎีจำนวนเป็นคณิตศาสตร์ที่เกี่ยวกับคุณสมบัติของเลขจำนวนเต็ม ที่เคยถูกมองว่าดูดีแต่ไร้ประโยชน์ในสาขาวิชาคณิตศาสตร์ แต่ปัจจุบันขั้นตอนวิธีทฤษฎีจำนวนกลับถูกนำไปใช้อย่างมากในเรื่องการเข้ารหัสลับบนพื้นฐานของเลขจำนวนเฉพาะตัวโตๆ เพราะความปลอดภัยของข้อมูลที่เข้ารหัสลับเกิดจากคุณสมบัติของเลขจำนวนเฉพาะที่ไม่สามารถหาเลขจำนวนประกอบได้ ยิ่งจำนวนเฉพาะขนาดใหญ่ๆ ก็จะยิ่งทำให้เดารหัสลับได้ยาก ในบทนี้ เราจะศึกษาเรื่องพื้นฐานของทฤษฎีจำนวน ขั้นตอนวิธีจัดการกับเลขจำนวนเฉพาะขนาดใหญ่มากๆ ขั้นตอนวิธีของ Euclid ในการหาค่าตัวหารร่วมมากของเลข 2 จำนวน, ทฤษฎีเศษ, วิธีการเข้ารหัสลับแบบ RSA

10.1 ทบทวนทฤษฎีจำนวน

จำนวนเฉพาะ (prime number) และจำนวนประกอบ (composite number)

จำนวนเฉพาะคือเลขที่ไม่มีเลขใดหารมันได้ลงตัว ยกเว้น 1 กับตัวมันเอง เช่น 2, 3, 5, 7 ,.... สำหรับเลขจำนวนเต็มอื่นๆ ที่ไม่ใช่จำนวนเฉพาะจะเรียกว่าเป็นจำนวนประกอบ เราจะใช้สัญลักษณ์ $h|n$ เพื่อแทนว่า ตัวตั้ง n หารด้วย h ได้ลงตัว ตัวอย่างเช่น $4|20$ คือ 20 หาร 4 ลงตัว

ตัวหารร่วมมาก (Greatest Common Divisor)

ตัวหารร่วมมาก คือเลขที่มากที่สุดที่สามารถหารเลขตั้งแต่ 2 จำนวนได้ลงตัว ตัวอย่างเช่น เลข 2 จำนวน 24 และ 30

เลขที่หาร 24 ได้ลงตัว ได้แก่ 1, 2, 3, 4, 6, 12, 24

เลขที่หาร 30 ได้ลงตัว ได้แก่ 1, 2, 3, 5, 6, 10, 15, 30

เลขที่สามารถหาร 24 และ 30 ได้ลงตัว (common divisor)ได้แก่ 1, 2, 3, 6

เพราะฉนั้น ตัวหารร่วมมากของ (24, 30) จะเท่ากับ 6 เราจะเขียนว่า gcd(24, 30) = 6

ทฤษฎีบทการหาร (Division Theorem)

สำหรับ จำนวนเต็มใดๆ a และ จำนวนเต็มบวก n จะมีจำนวนเต็ม q และ b ได้เพียงจำนวนเดียวที่ทำให้ $0 \le b < n$ และ $a = qn+b$

จะเรียกค่า q ว่าผลหาร (quotient) มีค่า $q = \lfloor a/n \rfloor$ และเรียกค่า b ว่าเศษ (residue) หรือเขียนว่า b = a mod n ซึ่งจำนวนเต็ม a มีได้หลายจำนวนที่หาร n แล้วเหลือเศษ b ตัวอย่างเช่น

b =3, n =7 จะได้ว่า a = {…, -11, -4, 3, 10, 17,…} เราสามารถเขียนได้ว่า $a \equiv b \pmod n$ (เรียกว่า a congruence กับ b mod n)

จำนวนเฉพาะสัมพัทธ์(Relatively prime)

จำนวนตึ็ม a, b จะเรียกว่าเป็นrelatively prime ก็ต่อเมื่อตัวหารร่วมมากของมันเป็น 1 ตัวอย่าง 8 และ 15 เป็น relatively prime เพราะตัวประกอบของ 8 = 1, 2, 4, 8 และ ตัวประกอบของ 15 = 1, 3, 5, 15

10.2 ขั้นตอนวิธีของ Euclid

การหาค่าหารร่วมมาก ของเลข 2 จำนวน (3,185,325 และ 7,276,500) ทำได้โดยหาตัวประกอบของเลข 2 จำนวน แล้วดึงตัวประกอบร่วมออกมา เช่น

$3,185,325 = 3^4 \times 5^2 \times 11^5 \times 13^1$

$7,276,500 = 2^2 \times 3^3 \times 5^3 \times 7^2 \times 11^1$

$gcd(3,185,325 , 7,276,500) = 3^3 \times 5^2 \times 11^1 = 7,425$

เทคนิคข้างต้นไม่ง่ายสำหรับตัวเลขจำนวน 25 หลัก ซึ่งเวลาในการแก้ปัญหาด้วยการแยกตัวประกอบ จะเป็นแบบพหุนาม (polynomial-time) วิธีหาค่าหารร่วมมากที่มีประสิทธิภาพมากขึ้นโดยขั้นตอนวิธีของ Euclid

<u>ขั้นตอนวิธีของ Euclid</u>

<u>โจทย์</u> คำนวณค่าตัวหารร่วมมาก ของเลขจำนวนเต็มบวก 2 จำนวน a, b

```
public static int Euclid(int a, int b)  // รับค่าจำนวนเต็มบวก a, b
{  if  (b=0)
      { return a; }
   else
      { return Euclid(b, a mod b); }
} // end euclid
```

ตัวอย่าง 10.1 จงหาค่าหารร่วมมากของ (30, 21)

Eucild (30, 21) = Eucild (21, 9)

= Eucild (9, 3)

= Eucild (3, 0)

gcd (30,21) = 3

10.3 ขั้นตอนวิธี Extended Euclid

เราจะขยายขั้นตอนวิธีของ Euclid เพื่อคำนวณค่า สัมประสิทธิ x และ y จากสมการ

d = gcd(a,b) = ax + by

ขั้นตอนวิธี Extended Euclid

```
ExtendedEuclid(a,b)
{   if   (b = 0)
       {return (a, 1, 0);}
    (d', x', y') = ExtendedEuclid(b, a mod b)
    (d, x, y) = (d', y', x' – ⌊a/b⌋ y')
    return (d, x, y)
}//end ExtendedEuclid
```

ตัวอย่าง 10.2 จงหาค่า x, y จากสมการ

d = gcd(99, 78) = ax+by

<u>วิธีทำ</u>

	a	b	⌊a/b⌋	d	x	y = x' – ⌊a/b⌋ y'	y
1.	99	78	1	3	-11	3-(1x-11)	14
2.	78	21	3	3	3	-2-(3x3)	-11
3.	21	15	1	3	-2	1-(1x-2)	3
4.	15	6	2	3	1	0-(2x1)	-2
5.	6	3	2	3	0	1-(2x0)	1
6.	3	0	-	3	1		0

1. เรียก ExtendedEuclid โดยส่งค่า a = 99 และ b = 78 ➔ (d', x' ,y') = ExtendedEuclid(78, 21)

2. เรียก ExtendedEuclid โดยส่งค่า a = 78 และ b = 21 ➔ (d', x' ,y') = ExtendedEuclid(21, 15)

3. เรียก ExtendedEuclid โดยส่งค่า a = 21 และ b = 15 ➔ (d', x' ,y') = ExtendedEuclid(15, 6)

4. เรียก ExtendedEuclid โดยส่งค่า a = 15 และ b = 6 ➔ (d', x' ,y') = ExtendedEuclid(6, 3)

5. เรียก ExtendedEuclid โดยส่งค่า a = 6 และ b = 3 ➔ (d', x' ,y') = ExtendedEuclid(3, 0)

6. เรียก ExtendedEuclid โดยส่งค่า a = 3 และ b = 0 จะได้ค่า d = 3, x=1, y=0

คราวนี้ pop จาก stack ย้อนจากข้อ 5, 4, 3, 2, 1

5b. จะได้ค่า d = 3, x=0, y= 1-(2x0)=1,

4b. จะได้ค่า d = 3, x=1, y= 0-(2x1) = -2

3b. จะได้ค่า d = 3, x=-2, y= 1-(1x-2) = 3

2b. จะได้ค่า d = 3, x=3, y=**-2-(3x3)**=-11

1b. จะได้ค่า d = 3, x= -11, y= **3-(1x-11)**=14

จาก ExtendedEuclid จะได้ค่า x = -11 และ y = 14

ประสิทธิภาพของขั้นตอนวิธีExtendedEuclid จะใช้เวลาเท่ากับ O(log b)

10.4 ปัญหา Modular linear Equation

จงหาค่า x โดยกำหนดค่า a, b และ n จากสมการ

$$ax \equiv b \pmod{n} \quad \text{เมื่อ } a > 0 \text{ และ } n > 0$$

สมการนี้อาจหาคำตอบไม่ได้ หรือ สามารถหาคำตอบได้มากกว่า 1 ค่า

<u>ขั้นตอนวิธี ModularLinearEquation</u>

```
ModularLinearEquation (a, b, n)
{     (d, x', y') = ExtendedEuclid(a, n);
      if (d|b = true)        // ถ้า b หารด้วย d ลงตัว ➔ b%d = 0
          { x0 = x' * (b/d) mod n;
            for (i = 0; i<= d-1; ++i)
              { x = x0+ i*(n/d)) mod n;
                print (x); } //end for i
          } end if
      else
         { print ("no solution");  }
} //end ModularLinearEquation
```

ตัวอย่าง 10.3 จงหาค่า x จากสมการ

$$14x \equiv 30 \pmod{100}$$

<u>วิธีทำ</u>

	a	b	$\lfloor a/b \rfloor$	d	x	$y = x' - \lfloor a/b \rfloor y'$	y
1.	14	100	0	2	-7	1-(0x-7)	1
2.	100	14	7	2	1	0-(7x1)	-7
3.	14	2	7	2	0	1-(7x0)	1
4.	2	0	-	2	1		0

1. กำหนดให้ a = 14, b=30 และ n = 100
2. เรียก ExtendedEuclid จะได้ (d, x', y') =(2, -7, 1)
3. d|b เป็นจริงเพราะ 30/2 ลงตัว จะได้ x0 = (-7)(15) mod 100 = 95
4. ทำคำสั่ง for i 2 รอบ จะได้ค่า x = 95 และ 45 เป็นคำตอบ

ประสิทธิภาพ เวลาที่ใช้สำหรับเรียก ExtendedEuclid = O(log n) + การวนหาค่า x จาก loop for i เท่ากับ O(gcd(a,n)) รวมเป็น O(log n + gcd(a,n))

10.5 ปัญหา Modular Exponentiation

จะเป็นการคำนวณค่าของ $d = a^b \bmod n$ โดยกำหนดค่า a, b, n เป็นเลขจำนวนเต็มบวก การคำนวณค่าโดยปกติของเลขยกกำลังมากๆ เช่น $7^{560} \bmod 561$ จะทำให้เกิด over flow error ขั้นตอนวิธี ModularExponential จะสามารถคำนวณค่าดังกล่าวโดยไม่เกิด error

<u>ขั้นตอนวิธี ModularExponential</u>

```
ModularExponential(a, b, n)
{  c = 0;
   d = 1;
  กำหนดให้ <b[k], b[k-1], …, b[0]> เป็นค่าเลขฐาน 2 แทนค่าของ b
  for (i = k; i>=0; --i)
       {  c = 2*c;
          d = (d*d) mod n;
          if (b[i] == 1)
              { c= c+1;
                 d = (d*a) mod n;
              }  //end if
       } // end for
} // end ModularExponential
```

ตัวอย่าง 10.4 จงหาค่า $d = 7^{560} \bmod 561$

1. กำหนดให้ a = 7, b = 560 และ n = 561
2. หาค่าเลขฐาน 2 ของ b จะเท่ากับ 1000110000
3. วน loop for i = 9, 8,…0 ตามตาราง

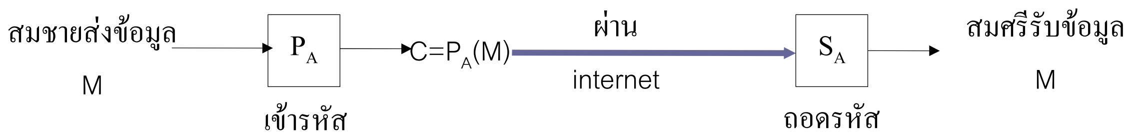
ModularExponential เพื่อคำนวณ ค่า $7^{560} \bmod 561$ ให้ a = 7, b = 560 และ n = 561										
i	9	8	7	6	5	4	3	2	1	0
b[i]	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

ได้ d = 1 และประสิทธิภาพ จะเท่ากับจำนวนบิตของเลขฐานสองของค่า b นั่นคือ O(log b)

10.6 การเข้ารหัสลับ RSA

การเข้ารหัสลับ RSA จะประกอบด้วย 2 รหัสลับ คือ รหัสสาธารณะ (public key) และรหัสลับส่วนตัว (secret key)

ตัวอย่าง สมศรีเปิดร้านขายของทางอินเตอร์เน็ต มีสมชายเป็นลูกค้า สมศรีจะให้รหัสลับ P_A กับลูกค้าทุกคน และเก็บรหัสลับ S_A ไว้ไม่ให้ใครรู้ เมื่อลูกค้าชื่อสมชายส่งข้อมูลเลขที่บัตรเครดิตด้วยการเข้ารหัสมาด้วยรหัสลับ P_A แล้วถูก hack ระหว่างทางบน internet คนที่ hack ได้ไปก็ไม่สามารถอ่านข้อมูลรู้เรื่อง ถ้าไม่มีรหัสลับ S_A เพื่อถอดข้อมูล ตามรูป 10.1



รูป10.1 แผนภาพระบบการเข้ารหัสลับ RSA

ขั้นตอนการสร้างรหัสลับ RSA

1. เลือกสุ่มจำนวนเฉพาะ p และ q ให้เลือกค่า p และ q มากๆ ประมาณว่าขนาด 512 บิต
2. คำนวณค่า $n = pq$
3. เลือกจำนวนเฉพาะค่าน้อยๆ e ที่เป็นจำนวนเฉพาะสัมพัทธ์กับ (p-1)(q-1) นั่นคือ

   $1 < e < (p-1)(q-1)$ และ $\gcd((p-1)(q-1),e) = 1$.
4. คำนวณค่า d ที่ทำให้

   $(de) \bmod (p-1)(q-1) = 1$.

   โดยใช้ขั้นตอนวิธี ModularLinear Equation
5. จะได้รหัสลับสาธารณะ(RSA public key) เป็น $P = (e, n)$ และ รหัสลับส่วนตัว(RSA secret key) เป็น $S = (d, n)$
6. การเข้ารหัส(encrypt) ข้อมูล M ➔ $P(M) = C = M^e \bmod n$.
7. การถอดรหัส(decrypt) จาก C ➔ $S(C) = C^d \bmod n = M$

ตัวอย่าง 10.5 จงหา RSA key กำหนดให้ p = 11, q =29 และ e = 3

ก. p =11, q= 29

ข. n = 11x29 = 319

ค. ϕ = (p-1)(q-1) =280

ง. เลือกให้ e = 3 ➔ ทดสอบจำนวนเฉพาะสัมพัทธ์(relatively prime)

gcd(280, 3) = 1

จ. คำนวณหาค่า d ที่

3d ≡ 1 mod 280 โดยเรียก ModularLinearEquation (3,1, 280)

a	b	⌊a/b⌋	d	x	y
3	280	0	1	-93	1-(-93x0)= 1
280	3	93	1	1	0-(93x1) = -93
3	1	3	1	0	1-(3x0) = 1
1	0	-	1	1	0

d = 187

ฉ. เราจะได้ RSA public key เป็น P = (3, 319)

ช. เราจะได้ RSA secret key เป็น S = (187, 319)

เข้ารหัส(encryption)**, ถอดรหัส**(decryption)

ถ้าข้อความที่ต้องการเข้ารหัส $M < n$.

เข้ารหัส โดยใช้ RSA public key P = (e,n) ได้ข้อความที่เข้ารหัสแล้วเป็น C

$P(M) = C = M^e \bmod n$.

แล้วส่ง ข้อความ C ไปในระบบสื่อสาร เมื่อปลายทางได้รับข้อความ C จะต้องทำการถอดรหัส โดยใช้ RSA secret key S = (d,n) ด้วยการคำนวณ

$S(E) = M = E^d \bmod n$

ตัวอย่าง 10.6 จงใช้ RSA public key (3,319) และ RSA secret key (187,319) เพื่อเข้ารหัส และถอดรหัสข้อความ

กำหนดให้ ข้อความ M = 100, RSA public key (3,319) และ RSA secret key (187,319)

$P(100) = 100^3 \bmod 319$

ModularExponential (100,3,319) // n = 319, a = 100, b =3

$3_{10} = 11_2$

i		1	0
b_i		1	1
c	0	1	3
d	1	100	254

$C= P(100) = 100^3 \bmod 319= 254$

$S(254) = 254^{187} \bmod 319$

ModularExponential (254,187,319) // n= 319, a = 254, b =187

$187_{10} = 11_2$

i		7	6	5	4	3	2	1	0
b_i		1	0	1	1	1	0	1	1
c	0	1	2	5	11	23	46	93	187
d	1	254	78	100	122	67	23	67	100

$S(254) = 254^{187} \bmod 319 = 100 = M$