Chapter 5. Number theory, 정수론

5.1 Divisors 약수

Definition Let n and $d \neq 0$ be integers. We say that 'd *divides* n,' if there exists an integer q (*quotient* 몫) such that $n = dq$. We call d a *divisor* (약수) or *factor* (인수) of n. If d divides n, we write $d|n$.

← The quotient-remainder theorem says, for each integer n, there exists unique q and r (*remainder* 나머지), such that $n = dq + r$ and $0 \leq r < d$. Thus, $d|n$, if and only if $r = 0$.

← If n and d are positive integers and $d|n$, then $d \leq n$.

Theorem 5.1 Let m, n and d be integers:

(a) If $d|m$ and $d|n$, then $d|(m+n)$.
(b) If $d|m$ and $d|n$, then $d|(m-n)$.
(c) If $d|m$, then $d|mn$.

Definition An integer greater than 1 whose only positive divisors are itself and 1 is called *prime* (소수). An integer greater than 1 that is not prime is called *composite* (합성수).

Note that 1 is not a prime number. 20 세기 중반까지도 소수의 지위를 유지하고 있었지만, 18 세기부터 점차 1 은 소수에서 제외되기 시작하였다. 1 은 인수가 되기에 부적절하기 때문이다. 1 을 곱하더라도 수의 값이 변하지 않기 때문에 소인수 분해의 uniqueness 특성을 위배하게 된다.

기원전 300 년 즈음에 Euclid 가 밝힌 대로, 무한히 많은 소수가 있다. 오랜 시간동안 소수는 순수수학을 제외한 여타 분야에서는 관심을 받지 못하였다. 하지만, 1970 년대에 public-key cryptography (공개키 암호 시스템)의 개념이 소개된 이후로 소수는 RSA cryptosystem algorithm(암호화 체계)의 이론적인 토대를 제공하였고, 활발한 연구가 진행되어 왔다.

소수의 중요성이 인식되면서, 가장 큰 소수를 찾는 노력이 계속되어 왔으며, 익명의 후원자는 Electronic Frontier Foundation 을 통해 소수에 포상금을 걸었다. 천만 이상 자리 소수에 10만 달러, 일억 이상 자리 소수에 15만 달러, 그리고 10억 이상 자리의 소수에는 25만 달러의 상금이 걸려있다. $2^p - 1$ (p prime) 형태의 소수는 *Mersenne* 소수로 알려져 있다. 현재, 2016년 1월 현재까지 밝혀진 가장 큰 Mersenne 소수는 $2^{74,207,281} - 1$이고 22,338,618 자릿수를 갖는다.

소수를 찾는 일은 많은 연산량을 필요로 하기 때문에 1996년부터 cloud computing 을 통한 초대형 Mersenne 소수 공동 project, GIMPS(great internet Mersenne prime search)가 시작되었다. 소수 찾기 어려움은 어떤 수가 소수인지 정확하게 예측할 수 없다는데 있다. GIMPS 의 방법은 소수가 될 만한 대형수(*e.g.*, Mersenne prime)를 추측하여 후보로 선정하고 정말 소수인지를 확인하기 위하여 약수일 것 같은 모든 수로 나누어 보는 것이다. 개인용 PC 가 분산형 super computer 의 일부가 되어 project 에 참여할 수 있으며, 현재 세계 천만대 이상의 processor 가 154조 FLOPS 연산처리하고 있다. GIMPS 는 2009년에 소수 $2^{43,112,609} - 1$을 발견하여 첫번째 소수에 걸린 상금을 차지하였다.

Mersenne 은 $n = 2, 3, 5, 7, 13, 19, 31, 67, 127, 257$일 때 $2^n - 1$이 소수가 되며, n이 257 이하인 Mersenne 수 가운데 다른 소수가 없다고 주장하였다. 하지만, 1903년 $2^{67} - 1$이 합성수임이 밝혀지고 $2^{257} - 1$ 역시 합성수로 증명되었다. 수학적으로 소수가 아닌 것은 증명되었지만 약수를 찾는데는 27 년이 걸렸다. 현재까지 485개의 Mersenne 소수가 발견되었다.

Mersenne primes: $2^2 - 1 = 3$, $2^3 - 1 = 7$, $2^5 - 1 = 31$, $2^7 - 1 = 127$, $2^{13} - 1 = 8191$,

$2^{31}-1=2{,}148{,}483{,}647,\ 2^{61}-1,$

$2^{127}-1=170{,}141{,}183{,}460{,}469{,}231{,}731{,}687{,}303{,}715{,}884{,}105{,}717$ (39자리수로 1876년 발견된 종이, 연필, 두뇌만을 사용하여 찾아낸 가장 큰 소수)

합성수: $2^4-1=3\times5,\ 2^6-1=3^2\times7,\ 2^8-1=3\times5\times17,\ 2^9-1=7\times73,$

$2^{10}-1=3\times11\times31,\ 2^{11}-1=23\times89,\ 2^{12}-1=3^2\times5\times7\times13$

Example 5.1 $n_1=23,\ n_2=34,\ n_3=43,$ and $n_4=451=11\cdot41$

← Suppose $n>1$ is composite. Then, $\exists$ a divisor $d>0,\ d\neq1,$ such that $d|n$. Since d is a divisor, $d\leq n$ and $d\neq n$, we have $d<n$. Thus, to tell if $n>0$ is composite, it suffices to check if any of $\{2,3,\dots,n-1\}$ divides n.

Theorem 5.2 A positive integer $n>1$ is composite, if and only if n has a divisor d satisfying

(5.1) $\quad 2\leq d\leq\sqrt{n}$

(Proof) If n has a divisor, then it is composite, by definition. Suppose that n is composite. Let's show that n has a divisor d satisfying $2\leq d\leq\sqrt{n}$. Since n is composite, it has a divisor $d',\ 2\leq d'<n$.

(a) If $d'\leq\sqrt{n}$, choose $d=d'$.

(b) Suppose $d'>\sqrt{n}$. Since $d'|n,\ \exists$ a unique $q\in Z$ such that $n=d'q$. This implies $q|n$. Claim that $q\leq\sqrt{n}$. (We can prove this by contradiction.) Then, we choose $q=d$.

Algorithm 5.1: This algorithm determines whether an integer $n>1$ is prime. If prime, it returns 0. Otherwise, it returns a divisor d satisfying $2\leq d\leq\sqrt{n}$.

```
import math
def is_prime(n):
        bound = math.floor(math.sqrt(n))
        for d in range(2, bound+1):
                if n%d == 0:
                        return d
        return 0
```

← When a composite number n is an input to this algorithm, the output divisor is always a prime.

Example 5.2 $n_1=43,\ n_2=451,\ n_3=1274$

$1274=2\cdot637,\ 637=7\cdot91,\ 91=7\cdot13,$ so that $1274=2\cdot7^2\cdot13$

Theorem 5.3 *Fundamental theorem of arithmetic* (*Unique factorization theorem*) Any integer $n>1$ can be written as a product of primes. Moreover, if the primes are written in non-decreasing order, the factorization is unique: *i.e.*

(5.2) $\quad n=p_1p_2\cdots p_i,$

where p_k are prime with $p_1\leq p_2\leq\cdots\leq p_i$ (*prime factorization*, 소인수분해).

Theorem 5.4 The number of primes is infinite.

(Proof) Show that if p is a prime, then there exists a prime greater than p. Suppose that there is a finite number of primes, say $p_1, p_2, \cdots, p_n$ less than or equal to p. Put

$$m = p_1 p_2 \cdots p_n + 1$$

$$m = p_i q + 1 \text{ and } q = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_n$$

Thus, p_i does not divide m, for all $1 \le i \le n$. If m is prime, m is a prime greater than p. If not, let p' be a factor of m. Then, $p' \neq p_i$ and $p' > p$, for all $1 \le i \le n$. Therefore, the conclusion follows.

Definition Let m and n be integers with not both m and n zero. A *common divisor* (공약수) of m and n is an integer that divides both m and n. The *greatest common divisor* (최대공약수), written $gcd\,(m,n)$, is the largest common divisor of m and n.

Example 5.3 $gcd(30, 105)$, where $30 = 2 \cdot 3 \cdot 5$ and $105 = 3 \cdot 5 \cdot 7$

Theorem 5.5 Let m and n be integers, $m > 1$ and $n > 1$, with prime factorizations

(5.3) $$m = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k} \text{ and } n = p_1^{b_1} p_2^{b_2} \cdots p_k^{b_k}$$

(if the prime p_i is not a factor of m, then we let $a_i = 0$ or $b_i = 0$). Then,

(5.4) $$gcd\,(m,n) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_k^{\min(a_k,b_k)}$$

Example 5.4 When $82320 = 2^4 \cdot 3 \cdot 5 \cdot 7^3$ and $950796 = 2^2 \cdot 3^2 \cdot 7^4 \cdot 11$

$$gcd(82320, 950796) = 2^2 \cdot 3 \cdot 5^0 \cdot 7^3 \cdot 11^0 = 4116$$

Definition Let m and n be positive integers. A *common multiple* (공배수) of m and n is an integer that is divisible by both m and n. The *least common multiple* (최소공배수), written $lcm\,(m,n)$, is the smallest positive common multiple of m and n.

Example 5.5 $lcm(30, 105) = 2 \cdot 3 \cdot 5 \cdot 7$

Theorem 5.6 Let m and n be integers, $m > 1$ and $n > 1$, with prime factorizations

$$m = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k} \text{ and } n = p_1^{b_1} p_2^{b_2} \cdots p_k^{b_k}$$

(if the prime p_i is not a factor of m, then we let $a_i = 0$). Then,

(5.5) $$lcm\,(m,n) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \cdots p_k^{\max(a_k,b_k)}$$

Theorem 5.7 For any positive integers m and n,

(5.6) $$gcd(m,n) \cdot lcm(m,n) = mn$$

$\leftarrow$ $max(a_i, b_i) + min(a_i, b_i) = a_i + b_i$

5.2 Representation of numbers 수의 표현 방법

수는 어떤 기저(base)를 선택하는가에 따라 표현 방법이 달라진다. 공학적으로는 binary (base-2), octal (base-8), 그리고 hexadecimal (base-16) 표현 방법을 많이 사용하고 있다.

Example 5.6 $3854 = 3\times10^3 + 8\times10^2 + 5\times10^1 + 4\times10^0 = (1111\ 0000\ 1110)_2 = (7416)_8 = (\mathrm{F0E})_{16}$

Binary representation of integers

Given a positive integer n, its binary representation is given by

(5.7) $n = 1\cdot 2^k + b_{k-1}2^{k-1} + \cdots + b_0 2^0$

← Since $2^k \le n < 2^{k+1}$, $k+1 \le \log_2 n + 1 < k+2$, so that we need $(k+1) = \lfloor \log_2 n + 1 \rfloor$-bits to represent a positive integer n.

Algorithm 5.2: Converting an integer $(c_n c_{n-1}\cdots c_0)_b$ from base b to decimal.

```
def b2dec(c, b):
        # convert (c)_b to corresponding decimal value
        # input, c, in list format
        dec_val, power = 0, 1
        n = len(c)
        for i in range (n-1, -1, -1):
                dec_val = dec_val + c[i]*power
                power = power*b
        return dec_val
```

Example 5.7 Tracing of algorithm 5.2 with $(1101)_2$

$s = [1, 0, 1, 1],\ *.\mathrm{b2dec}(s, 2)$

Example 5.8 (a) Hexadecimal to decimal, $\mathrm{B4F}_{16} = 11\cdot 16^3 + 4\cdot 16 + 15$

(b) Decimal to binary, 130_{10}

130	
65	0
32	1
16	0
8	0
4	0
2	0
1	0

- total $\lfloor \log_2 130 + 1 \rfloor = 8$ [bits]
- $130_{10} = 1000\ 0010_2$

Algorithm 5.3: Converting a decimal integer m into base-b integer $(c_n c_{n-1} \cdots c_0)_b$.

```
def dec2b(m, b):
        # convert decimal integer, m, into base-b
        # output in 'string' format (reverse order)
        c = ''                  # Null (empty) string
        while m>0:
                c = c + (str)(m%b) # modulo-b
                m = m // b
        return c
```

Example 5.9 Tracing of algorithm 5.3 with $(130)_{10}$

← $*.\mathrm{dec2b}(130, 2)$

$m = 130 \rightarrow 130(\mathrm{mod}\ 2) = 0,\ c = '0'$, and $\lfloor 130/2 \rfloor = 65$

$\rightarrow 65(\mathrm{mod}\ 2) = 1,\ c = '01'$, and $\lfloor 65/2 \rfloor = 32$ ...

Example 5.10 (a) Decimal to hexadecimal, $(20385)_{10}$

(b) Binary addition, $(10011011)_2 + (01011011)_2$

(c) Hexadecimal addition, $(84F)_H + (42EA)_H$

Repeated squaring

Consider computing a^{29}. Given $(29)_{10} = (11101)_2 = 1 + 4 + 8 + 16$,

$$a^{29} = a \cdot a^4 \cdot a^8 \cdot a^{16}$$

Table 5.1 Computing a^{29} with repeated squaring

x	current value of n	$n\ (mod\ 2)$	result	$\lfloor n/2 \rfloor$
a	29	1	a	14
a^2	14	0	unchanged	7
a^4	7	1	$a \cdot a^4 = a^5$	3
a^8	3	1	$a^5 \cdot a^8 = a^{13}$	1
a^{16}	1	1	$a^{13} \cdot a^{16} = a^{29}$	0

Algorithm 5.4: Repeated squaring to compute a^n.

```
def a_pow_n(a, n):
        # return a^n, where a: base, n: power
        y, x = 1, a
        while n>0:
                if n%2 == 1:
                        y = y*x
                x = x*x
                n = n//2
        return y
```

← $*.\mathrm{a\_pow\_n}(572, 20)$:

← Python supports a library computing the power: $\mathrm{pow}(572, 20)$

Theorem 5.8 If a, b, and z are positive integers, then

(5.8) $ab\ (mod\ z) = \big((a\ (mod\ z)) \cdot (b\ (mod\ z))\big)\ (mod\ z)$

(Proof) Let $w = ab\ (mod\ z)$, $x = a\ (mod\ z)$, and $y = b\ (mod\ z)$. Then,

$ab = q_1 z + w$, $a = q_2 z + x$, and $b = q_3 z + y$

$w = ab - q_1 z = qz + xy \ \Rightarrow\ w = xy\ (mod\ z)$

Example 5.11 $572^{29}(mod\ 713)$

In order to compute $a^{29}(mod\ z)$, we compute successively

$a(mod\ z)$

$a^2(mod\ z) = \big(a(mod\ z) \cdot a(mod\ z)\big)(mod\ z)$, $a^4(mod\ z) = \big(a^2(mod\ z) \cdot a^2(mod\ z)\big)(mod\ z)$

$a^5\ (mod\ z) = (a\ (mod\ z) \cdot a^4(mod\ z))$

$a^8(mod\ z) = (a^4(mod\ z) \cdot a^4(mod\ z))\ (mod\ z)$

$a^{13}\ (mod\ z) = (a^5(mod\ z) \cdot a^8(mod\ z))\ (mod\ z)$

$a^{16}(mod\ z) = (a^8(mod\ z) \cdot a^8(mod\ z))\ (mod\ z)$

$a^{29}\ (mod\ z) = (a^{13}(mod\ z) \cdot a^{16}(mod\ z))\ (mod\ z)$

Algorithm 5.5: Compute $a^n\ (mod\ z)$.

```
def a_to_n_modz(a, n, z):
        # return a^n(mod z), where a: base, n: power
        y = 1
        x = a%z
        while n>0:
                if n%2 == 1:
                        y = (y*x)%z
                x = (x*x)%z
                n = n//2
        return y
```

← $*$. a_to_n_modz$(572, 29, 713)$:

← Compare to $\text{pow}(572, 29)\%\ 713$

5.3 Euclidean algorithm

Theorem 5.8 If a is a non-negative integer, b is a positive integer, and $r = a(mod\ b)$, then

(5.9) $gcd(a, b) = gcd(b, r)$

(Proof) Let $a = bq + r$ with $0 \le r < b$. We claim that set of common divisors of a and b is equal to the set of common divisors of b and r.

If $c|b$ and $c|a$, then $c|bq$ and $c|(a - bq)$. Also, if $c|b$ and $c|r$, then $c|bq$ and $c|(bq + r)$.

Example 5.12 $gcd\ (105, 30)$

Since $105(\text{mod } 30) = 15$, $gcd(105, 30) = gcd(30, 15) = 15$

Algorithm 5.6: Euclidean algorithm: find $gcd(a, b)$.

```
def gcd(a, b):
        # return gcd of a and b
        if a < b:
                a, b = b, a          # swap a and b
        while b != 0:
                r = a%b
                a, b = b, r
        return a
```

← $*.\gcd(504, 396)$:

Example 5.13 $gcd\ (504, 396)$

Analysis of Euclidean algorithm

Let the operational time be the number of modulus operations in algorithm 5.6. Table 5.2 lists the number of modulus operations required for some small values of a and b.

Table 5.2 Number of modulus operations in algorithm 5.6

a\b	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0		0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	2	1	2	1	2	1	2	1	2	1	2
3	0	1	2	1	2	3	1	2	3	1	2	3	1	2
4	0	1	1	2	1	2	2	3	1	2	2	3	1	2
5	0	1	2	3	2	1	2	3	4	3	1	2	3	4
6	0	1	1	1	2	2	1	2	2	2	3	3	1	2
7	0	1	2	2	3	3	2	1	2	3	3	4	4	3
8	0	1	1	3	1	4	2	2	1	2	2	4	2	5
9	0	1	2	1	2	3	2	3	2					
10	0	1	1	2	2	1	3	3	2					
11	0	1	2	3	3	2	3	4	4					
12	0	1	1	1	1	3	1	4	2					
13	0	1	2	2	2	4	2	3	5					

In Table 5.2, we can find the input pair (a, b), $a > b$, with a as small as possible, that requires n modulus operations for $n = 0, 1, \dots, 5$. Those pairs are shaded in the table or can be listed as,

$$\{n: a, b\} = \{(0: 1, 0), (1: 2, 1), (2: 3, 2), (3: 5, 3), (4: 8, 5), (5: 13, 8), \dots\}$$

Note that values of b construct a Fibonacci sequence. Based on this observation, we can conjecture that, when the Euclidean algorithm with (a, b) requires n, $n \geq 1$, modulus operations, then

$a \geq f_{n+2}$ and $b \geq f_{n+1}$

Theorem 5.9 Suppose that, for a pair (a, b), $a > b$, the Euclidean algorithm requires n, $n \geq 1$, modulus operations, then

(5.10) $a \geq f_{n+2}$ and $b \geq f_{n+1}$

Theorem 5.10 If a and b are non-negative integers, not both zero, there exist integers s and t such that

(5.11) $gcd(a,b) = sa + tb$

Example 5.14 $gcd\ (273, 110)$

$\gcd(273,110) = \gcd(110,53) = \gcd(53,4) = \gcd(4,1) = 1$

$1 = 53 - 4 \cdot 13,\ 4 = 110 - 2 \cdot 53,$ and $53 = 273 - 2 \cdot 110$

Thus, $1 = 53 - 4 \cdot 13 = 53 - (110 - 2 \cdot 53) \cdot 13 = 27 \cdot 53 - 13 \cdot 110$

$= 27 \cdot (273 - 2 \cdot 110) - 13 \cdot 110 = 27 \cdot 273 - 67 \cdot 110$

Computing an inverse of modulo operation

$gcd(n, \phi) = 1$인 두 정수 $n > 0,\ \phi > 1$가 있다고 가정하자. $ns\ (\mathrm{mod}\ \phi) = 1,\ 0 < s < \phi$,를 만족하는 s를 $n\ (\mathrm{mod}\ \phi)$의 역원이라 한다. 이는 다음에 설명할 RSA 공개키 암호 시스템에서 중요한 역할을 한다.

$gcd(n, \phi) = 1$이므로, Theorem 5.10 에 따라 $s'n + t'\phi = 1$를 만족하는 s'와 t'이 존재한다. 그러면, $s'n = -t'\phi + 1$이므로,

(5.12) $ns'\ (\mathrm{mod}\ \phi) = 1$

다음과 같이 s를 선택하자.

$$s = s'\ (\mathrm{mod}\ \phi)$$

s가 $0 < s < \phi$ 조건을 만족함을 알 수 있다. s는 0이 될 수 없다. 만약 0이면, $\phi | s'$이 되어 (5.12) 식을 만족할 수 없게 된다. 따라서,

(5.13) $s' = q\phi + s$

이고,

$$ns = n(s' - q\phi) = -t'\phi + 1 - \phi nq = \phi(-t' - nq) + 1$$

이므로, 역원의 조건인 $ns\ (\mathrm{mod}\ \phi) = 1,\ 0 < s < \phi$이 만족되는 s가 구해진다.

Example 5.15 $n = 110,\ \phi = 273,\ gcd(273, 110) = 1,$ and $-67 \cdot 110 + 27 \cdot 273 = 1$:

Given $s' = -67$ and $t' = 27$, choose $s = s'(\mathrm{mod}\ \phi) = (-67)\ (\mathrm{mod}\ 273) = 206.$

따라서, $110\ (\mathrm{mod}\ 273)$의 역원은 206이다: 즉, $110 \cdot 206\ (\mathrm{mod}\ 273) = 1.$

Example 5.16 $n = 11,\ \phi = 47,\ gcd(47, 11) = gcd(11, 3) = gcd(3, 2) = gcd(2, 1) = 1,$ and $4 \cdot 47 + (-17) \cdot 11 = 1$:

Given $s' = -17$ and $t' = 4$, choose $s = (-17)\ (\mathrm{mod}\ 47) = 30.$

따라서, $11\ (\mathrm{mod}\ 47)$의 역원은 30이다: 즉, $11 \cdot 30\ (\mathrm{mod}\ 47) = 1.$

5.4 RSA public-key cryptosystem 공개키 암호 시스템

RSA 암호 시스템은 두 개의 **키**를 사용한다. 여기서 키(key)란 메시지(message)를 열고 잠그는 상수(constant)를 의미한다. 일반적으로 많은 공개키 알고리즘의 **공개키**(public key)는 모두에게 알려져 있으며 메시지를 암호화(encrypt)하는데 쓰인다. 암호화된 메시지는 **개인키**(비밀키, private key)를 가진 자만이 복호화(decrypt)하여 열어볼 수 있다.

RSA 암호 시스템은 소인수 분해의 난해함에 기반하여, 공개키 만을 가지고는 개인키를 쉽게 짐작할 수 없도록 디자인되어 있다. 예를 들어, B 가 A 에게 메시지를 전하고자 할 때, B 는 A 가 알려 준 공개키를 사용하여 메시지를 봉인(암호화)하고, A 에게 전달한다. A 는 자신 만이 알고 있는 열쇠(개인키)를 사용하여 그 메시지를 복호화 한다. 중간에 제 3 자가 그 메시지를 가로채더라도 비밀키를 가지고 있지 않으므로 메시지를 복호화 할 수 없다.

암호화 시스템에서는 모든 message 가 ASCII code 와 같은 숫자로 표시된다. 예를 들어, blank space 를 1 로 표시하고 알파벳 A~Z 를 2~27 등으로 표시한다면, message “SEND MONEY”는 20|06|15|05|01|14|16|15|06|26로 표현된다. 이를 하나로 묶어 하나의 숫자를 구성한다.

$a = 20061505011416150626$

Workflow of RSA system

A(수신자)와 B(송신자)가 보안이 보장되지 않는 통신 환경에서 비밀 메시지를 주고받고 싶다고 가정하자. B 가 A 에게 암호화 된 메시지를 보내려면 A 의 공개키를 사용해야 한다. 이때, A 는 다음 과정을 거쳐 자신만의 공개키와 비밀키를 생성한다.

(a) 암호화 시스템을 사용하는 각 사용자는 두 개의 소수 p와 q를 선택한다. 일반적으로 소수 p와 q는 100 자릿수 이상인 수로 선택한다. 그리고 다음 계산을 수행한다.

(5.12) $z = pq$ and

(5.13) $\phi = (p-1)(q-1)$

(b) $gcd(n, \phi) = 1$ (*i.e.* n과 ϕ는 서로 소) 조건을 만족하는 정수 n (usually prime)을 선택한다.

(c) 숫자 z와 n이 A 의 공개키 가 된다.

(d) 다음 조건을 만족하는 비밀키 $s(0 < s < \phi)$를 계산한다.

(5.14) $ns \ (mod\ \phi) = 1$

A 는 공개키 $< z, n >$을 B 에게 공개하고, B 는 이 공개키를 사용하여 자신의 메시지를 암호화 한다.

(e) Message $a(0 \le a < z)$를 공개키로 암호화 하여 아래와 같은 방법으로 암호문 c (encrypted message)값을 계산하여 전송한다.

(5.15) $c = a^n (mod\ z)$

(f) Message 를 수신한 수신자 A 는 자신만이 갖고 있는 비밀키를 사용하여 다음과 같이 암호문을 복호화 한다.

(5.16) $a = c^s (mod\ z)$

암호화 수준(encryption level, 복잡도)은 z값을 두 개의 소수 p와 q로 인수분해(factoring) 하는 과정이 얼마나 어려운가 하는 정도에 달려있다. 이러한 공개키 기반 암호화 시스템 (cryptosystem)에서 사용되는 기본적인 개념은 다음 식으로 표현될 수 있다.

(5.17) $a^u(mod\ z) = a$, for all $0 \le a < z$ and $u\ (mod\ \phi) = 1$

(5.14)식에 따라, 복호화 과정은 다음 수식으로 정리할 수 있다.

(5.18) $c^s(mod\ z) = \left(a^n(mod\ z)\right)^s(mod\ z) = a^{ns}(mod\ z) = a$

Example 5.16 (a) Choose $p = 23$, $q = 31$, and $n = 29$. Then, $z = pq = 713$ and $\phi = 660$.

(b) Given $s = 569$ as a secret key, since

$\gcd(n, \phi) = \gcd(660, 29) = 1$ and $4 \cdot 660 - 91 \cdot 29 = 1$

$\Rightarrow s' = -91$ and $s = s'(mod\ 660) = 569$

$ns\ (mod\ \phi) = 29 \cdot 569\ (mod\ 660) = 1$

(c) Open $(z, n) = (713, 29)$ as public key.

(d) To send a message $a = 572$ to the receiver who has public key, the sender computes and sends

$c = a^n(mod\ z) = 572^{29}(mod\ 713) = 113$

$\leftarrow 572^{29}(mod\ 713) = \left(572(mod\ 713) \cdot 572^4(mod\ 713) \cdot 572^8(mod\ 713) \cdot 572^8(mod\ 713) \cdot$ *57216mod 713 (mod 713)*

(e) The receiver computes

$c^s(mod\ z) = 113^{569}(mod\ 713) = 572$