암호학과 네트워크 보안

- 9장 암호수학(2)-

이 하 늘(haneul@pel.sejong.ac.kr)
세종대학교 프로토콜공학연구실

목 차

- 보충
- 소인수분해
- 중국인의 나머지 정리
  (CRT; Chinese Remainder theorem)
- 2차 합동

보충

- 대치 암호
  - 다중문자 암호
    - Hill 암호
      - 암호화시 m개의 문자를 한번에 치환하는 암호방식
      - 행렬을 키로 이용하며, 크기는 m×m의 정사각형
      - $k = \begin{bmatrix} k_{11} & \cdots & k_{1m} \\ \vdots & \ddots & \vdots \\ k_{m1} & \cdots & k_{mm} \end{bmatrix}$
      - $\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \\ 4 \end{bmatrix}$
      - 키 행렬이 곱셈에 대한 역원을 가져야 복호화 가능
      - 암호 해독
        - m값과 최소 m블록에 대한 평/암호문 쌍을 알고 있다면 알려진 평문 공격 가능

목 차

- 보충
- 소인수분해
- 중국인의 나머지 정리
  (CRT; Chinese Remainder theorem)
- 2차 합동

소인수분해

- 산술 기본 정리
  - 1보다 큰 모든 양의 정수는 소수의 곱들의 형태로 표현됨

$p_1, p_2, \cdots, p_k$은 소수이고, $e1, e2, \cdots, ek$는 양의 정수일 때,

$$n = p_1^{e1} \times p_2^{e2} \times \cdots \times p_k^{ek}$$

- 최대 공약수
  - 소인수 분해를 통해서 최대 공약수 구하기 가능

$$a = p_1^{a1} \times p_2^{a2} \times \cdots \times p_k^{ak} \qquad \mathrm{b} = p_1^{b1} \times p_2^{b2} \times \cdots \times p_k^{bk}$$

$$\gcd(a, b) = p_1^{\min(a_1, b_1)} \times p_2^{\min(a_2, b_2)} \times \cdots \times p_k^{\min(a_k, b_k)}$$

소인수분해

- 산술 기본 정리
  - 최소 공배수
    - 소인수 분해를 통해서 최소 공배수 구하기 가능

$$a = {p_1}^{a1} \times {p_2}^{a2} \times \cdots \times {p_k}^{ak} \qquad \mathrm{b} = {p_1}^{b1} \times {p_2}^{b2} \times \cdots \times {p_k}^{bk}$$

$$lcm(a, b) = {p_1}^{\max(a_1, b_1)} \times {p_2}^{\max(a_2, b_2)} \times \cdots \times {p_k}^{\max(a_k, b_k)}$$

소인수분해

- 소인수분해 방법
  - 전수 나눔 소인수분해 방법
    - 2부터 $\sqrt{n}$까지 n을 나누는 모든 양의 정수를 찾아냄
    - n이 합성수라면 n을 나누는 소수 p가 존재
    - $28 = 2^2 \times 7$
  - 복잡도
    - 전수 나눔 소인수분해 방법은 $n < 2^{10}$인 경우에 효과가 있음
    - 2^{10}이상의 정수를 소인수분해 하는 것은 효율성이 떨어지고 실용성이 없음

소인수분해

- 소인수분해 방법
  - 전수 나눔 소인수분해 방법
    - 의사코드

```
Trial_Division_Factorization(n)
{
    a<-2
    while(a<=sqrt(n))
    {
        while(n mod a=0)
        {
            output a
            n=n/a
        }
        a <- a+1
    }
    if(n>1) output n
}
```

```python
import math

a=2

n=int(input())

while a<=math.sqrt(n):
    while n%a == 0:
        print(a)
        n=n/a

    a = a+1
if n>1:
    print(int(n))
```

```
24
2
2
2
3
>>>
```

소인수분해

- 소인수분해 방법
  - 페르마 방법
    - 정수 n을 두 개의 양의 정수 p와 q로 나누어 $n = p \times q$로 표현하여 인수분해를 하는 방법

$$x = p \times q$$

$$x = (\frac{p+q}{2})^2 - (\frac{p-q}{2})^2 \qquad a = \frac{p+q}{2}, \mathrm{b} = \frac{p-q}{2}$$

- $n = a^2 - b^2 = (a+b) \times (a-b) = p \times q$
- 예시
  - $28 = 14 \times 2$
  - $28 = 8^2 - 6^2 = (8+6) \times (8-6)$

소인수분해

- 소인수분해 방법
  - 페르마 방법
    - 의사코드

```
Fermat_Factorization(n){
  x<-sqrt(n)

  while(x<n){
    w<-x^2-n

    if(w is perfect square){
      y<-sqrt(w)
      a<-x+y
      b<-x-y

      return a,b
    }
    x<-x+1
  }
}
```

```python
import math

def factoring_fermat(n) :
    a = math.ceil(math.sqrt(n))
    b2 = a * a - n
    b = round(math.sqrt(b2))

    while b * b != b2 :
        a = a + 1
        b2 = a * a - n
        b = round(math.sqrt(b2))

    return a-b,a+b


n = int(input())

p,q=factoring_fermat(n)

print("p:%d q:%d" %(p,q))
```

```
24
p:4 q:6
>>> 
```

소인수분해

- 소인수분해 방법
  - Pollard $p-1$ 방법
    - 1974년 존 폴라드가 발견한 소인수분해 알고리즘
    - 전제조건
      - 어떤 수의 소인수들이 매우 많은 경우 적합한 방법
      - 특정 값 B보다 큰 소인수를 갖지 않음
    - 전제조건을 만족하는 소수인수 p를 찾는 방법

$$p = \gcd(2_{B!} - 1, n)$$

    - 복잡도
      - B-1번의 지수연산이 필요

소인수분해

- 소인수분해 방법
  - Pollard $p-1$ 방법
    - 의사코드

```
Pollard_(p-1)_Factorization(n,B){
  a<-2
  e<-2
  while(e<=B){
    a<-a^e mod n
    e<-e+1
  }
  p<-gcd(a-1,n)
  if 1<p<n return p
  return fail
}
```

```python
def gcd(a, b):
    if a<b:
        a,b=b,a

    while b != 0:
        n = a%b
        a = b
        b = n
    return a

def factor(n, B):
    a = 2
    for i in (2, B+1):
        a = (a^i) % n
    d = gcd(a-1, n)
    if 1 < d < n:
        return d
    elif d==1:
        return True
    else:
        return None

n = int(input())
B = int(input())

result = factor(n, B)

print(result)
```

```
36
2
2
>>>
```

소인수분해

- 소인수분해 방법
  - Pollard rho 방법
    - 1975년 존 폴라드가 발견한 두 번째 인수분해 방법
    - 전제 사실
      - 두 개의 정수 x_1과 x_2가 존재하고, p는 $x_1 - x_2$를 나누지만 n은 $x_1 - x_2$을 나누지 못한다고 가정
      - $p = \gcd(x_1 - x_2, n)$을 증명 할 수 있음
        - P가 $x_1 - x_2$을 나누기 때문에 $x_1 - x_2 = q \times p$
        - n은 $x_1 - x_2$을 나누지 못하기 때문에 q는 n을 나누지 못함
        - 이는 $\gcd(x_1 - x_2, n)$이 1이거나 n의 인수라는 뜻

소인수분해

- 소인수분해 방법
  - Pollard rho 방법
    - 과정
      1. 가상 수열 정의

         $f(x) = x^2 + 1$

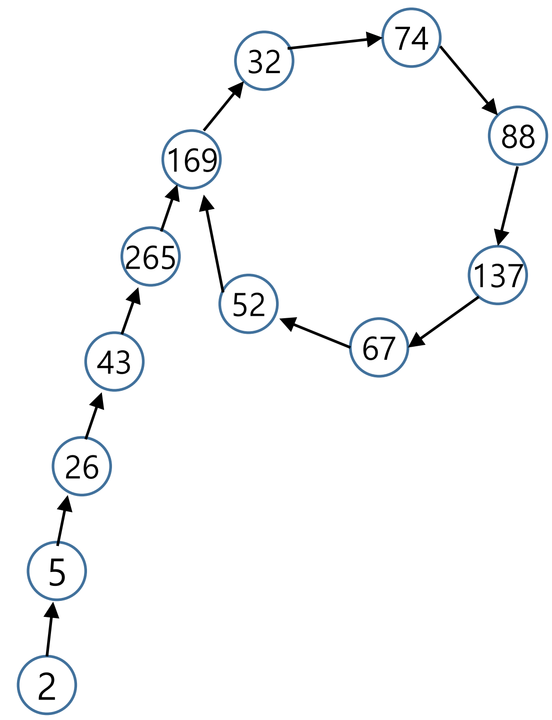
         $x_i = x_0, f(x_0), f(f(x_0)), f\left(f(f(x_0))\right), \cdots$
      2. n=24751043이라고 가정

         $x_i = 2, 5, 26, 677, 458330, 4286959, 21525579, \cdots$
      3. $p|n$인 p로 각 항을 나눈 나머지 구하기

         $24751043 = 317 \times 78079$

         P=317로 가정,

         $x_i \; mod \; p = 2, 5, 26, 43, 265, 169, 32, 74, 88, 137, 67, 52, 169 \cdots$



소인수분해

- 소인수분해 방법
  - Pollard rho 방법
    - 의사코드

```
Pollard_rho_Factorization(n,B){
  x<-2
  y<-2
  p<-1
  while(p=1){
    x<-f(x) mod n
    y<-f(f(y) mod n)mon n
    p<-gcd(x-y,n)
  }
  return p
}
```

```python
def gcd(a, b):
    if a<b:
        tmp = a
        a = b
        b= tmp

    while b != 0:
        n = a%b
        a = b
        b = n
    return a

def f(x, n):
        return (x*x%n + 1) % n

def factor(n, x0):
    x = x0
    y = x0
    p=1
    while p==1:
        x = f(x, n)
        y = f(y,n)
        y = f(y,n)
        p = gcd(abs(x - y), n)

    return p

n = int(input())

result = factor(n, 2)

print(result)
```

```
24751043
317
>>>
```

소인수분해

- 소인수분해 방법
  - 2차 체
    - $x^2\ mod\ n$의 값을 구함
    - 100자리 이상인 정수 인수분해 가능
  - 정수체 체
    - Hendric Lenstra와 Argin Lenstra가 고안한 소인수분해 방법
    - $x^2 \equiv y^2\ mod\ n$의 값을 구함
    - 120자리 이상인 정수를 인수분해 하려고 할 때 2차 체 보다 빠름

목 차

- 보충
- 소인수분해
- 중국인의 나머지 정리
  (CRT; Chinese Remainder theorem)
- 2차 합동

중국인의 나머지 정리

- 중국인의 나머지 정리
  - 모듈로 값들이 서로소만 된다면 연립방정식이 해를 갖는다는 것을 보이는 정리

$$x \equiv a_1 (mod\ m_1)$$
$$x \equiv a_2 (mod\ m_2)$$
$$\vdots$$
$$x \equiv a_k (mod\ m_k)$$

  - 과정
    1. 공통 모듈로로 사용할 $M = m_1 \times m_2 \times \cdots \times m_k$구하기
    2. $M_1 = \frac{M}{m_1}, M_2 = \frac{M}{m_2}, \cdots, M_k = \frac{M}{m_k}$를 구하기
    3. 곱에 대한 역원인 $M_1{}^{-1}$, $M_2{}^{-1}$, $\cdots$, $M_k{}^{-1}$을 구하기
    4. 다음 식을 따라 해를 구하기

$$x = \left(a_1 \times M_1 \times M_1{}^{-1} + a_2 \times M_2 \times M_2{}^{-1} + \cdots + a_k \times M_k \times M_k{}^{-1}\right) mod\ M$$

중국인의 나머지 정리

- 중국인의 나머지 정리
  - 예시

$$x \equiv 2(mod\ 3)$$
$$x \equiv 3(mod\ 5)$$
$$x \equiv 2(mod\ 7)$$

  - 과정

1. $M = 3 \times 5 \times 7 = 105$
2. $M_1 = \frac{105}{3} = 35, M_2 = \frac{105}{5} = 21, M_3 = \frac{105}{7} = 15$
3. 곱에 대한 역원은 ${M_1}^{-1} = 2,\ {M_2}^{-1} = 1,\ {M_3}^{-1} = 1$
4. $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) mod\ 105 = 23\ mod\ 105$

목 차

- 보충
- 소인수분해
- 중국인의 나머지 정리
  (CRT; Chinese Remainder theorem)
- 2차 합동

2차 합동

- 소수가 모듈로인 2차 합동
  - p가 소수이고 a는 p로 나누어 떨어지지 않는 정수일 때, $x^2 \equiv a(mod\ p)$형태를 갖는 방정식의 해를 구하는 방법
  - 2차 잉여(QR; Quadratic Residue)
    - 2차 합동 방정식이 두 개의 해를 가질 때의 a를 가리킴
  - 2차 비잉여(QNR; Quadratic NonResidue)
    - 2차 합동 방정식이 해를 갖지 않을 때의 a를 가리킴

2차 합동

- 소수가 모듈로인 2차 합동
  - 오일러의 판정기준
    - 한 정수가 모듈로 p로 QR인지 아닌지 알 수 있는 방법

    - 만약 $a^{(p-1)/2} \equiv 1 \ (mod\ p)$이면 모듈로 p로 a는 QR
    - 만약 $a^{(p-1)/2} \equiv -1 \ (mod\ p)$이면 모듈로 p로 a는 QNR

$${Z_{11}}^{*} = \{1,2,3,4,5,6,7,8,9,10\}$$

$$\text{QR set} = \{1,3,4,5,9\}$$

$$QNR\ set = \{2,6,7,8,10\}$$

2차 합동

- 소수가 모듈로인 2차 합동 방정식 풀기
  - $p = 4k + 3$인 경우(즉, $p = 3 \; mod \; 4$), a는 QR

$$x \equiv a^{\frac{p+1}{4}} mod \; p \text{이고 } x \equiv -a^{\frac{p+1}{4}} mod \; p$$

  - 예시
    - $x^2 \equiv 3(mod \; 23)$
    - $x^2 \equiv 2(mod \; 11)$
    - $x^2 \equiv 7 \; (mod \; 19)$

    - 3은 QR, 해는 $x \equiv \pm 16 \; mod \; 23$
    - 2는 QNR, 해는 없음
    - 7은 QR, 해는 $x \equiv \pm 11 \; mod \; 19$

2차 합동

- 모듈로가 합성수인 2차 합동 방정식 풀기
  - 모듈로가 소수인 2차 합동 방정식을 여러 번 풀이 하는 것으로 해결 가능

$$x^2 \equiv a \ mod \ (n)$$
$$n = \ p_1 \times p_2 \times \cdots \times p_k$$

$\longrightarrow$

$$x^2 \equiv a_1 \ (mod \ p_1)$$
$$x^2 \equiv a_2 \ (mod \ p_2)$$
$$\vdots$$
$$x^2 \equiv a_k \ (mod \ p_k)$$

$\longrightarrow$

$$x_1 \equiv \pm b_1 \ (mod \ p_1)$$
$$x_2 \equiv \pm b_2 \ (mod \ p_2)$$
$$\vdots$$
$$x_k \equiv \pm b_k \ (mod \ p_k)$$

2차 합동

- 모듈로가 합성수인 2차 합동 방정식 풀기
  - 예시
    - $x^2 \equiv 36(mod\ 77)$
      - $x^2 \equiv 36\ mod\ 7 \equiv 1(mod\ 7)$이고 $x^2 \equiv 36\ mod\ 11\ \ \equiv 3\ (mod\ 11)$
      - 7과 11이 4k+3의 형태이기 때문에 해를 구하기 가능
      - $x \equiv +1(mod\ 7), x \equiv -1(mod\ 7), x \equiv +5(mod\ 11), x \equiv -5(mod\ 11)$

Thanks!

이 하 늘(haneul@pel.sejong.ac.kr)