

COMPUTER SECURITY

NUMBER THEORY

Willy Sudiarto Raharjo

Teknik Informatika
Universitas Kristen Duta Wacana

Basic Concepts

- A number a is a divisor of b ($a \mid b$)
 - ◆ Example: 7 is a divisor of 35 $\rightarrow 7 \mid 35$
- Quotient is the result of division ($a \text{ div } b$)
 - ◆ Example: $35 \text{ div } 4 = 8$
- Modulo is the remainder of the division of two numbers
 - ◆ Example: $35 \text{ mod } 4 = 3$
- Prime is a number which has exactly two positive divisors
 - ◆ 1 and the number itself
- Integer > 1 that is not prime $\rightarrow$ composite
- So what is 1?

Basic Lemmas

- Lemma 1

If $a \mid b$ and $b \mid c$ then $a \mid c$

- Proof

- ◆ If $a \mid b$, then there's integer s such as $as = b$

- ◆ If $b \mid c$, then there's integer t such as $bt = c$

- ◆ $c = bt$

$$c = (as)t$$

$$c = a(st)$$

Basic Lemmas

- Lemma 2

Let n be a positive number > 1

Let d be smallest divisor of $n > 1$ then d is prime

- We use proof by contradiction (assume d is not prime)
- If d is not prime, it has divisor e such that $1 < e < d$.
- From lemma 1: if $e \mid d$ and $d \mid n$ then $e \mid n$ and $e < d$
- This lead to contradiction $\rightarrow d$ is smallest divisor of n
- Assumption is false $\rightarrow d$ must be prime

Small Primes

```
void simplePrime (unsigned long n) {  
    int i, j, prima;  
    for (i = 1; i < n; i++){  
        prima = 0;  
        for (j = 1; j <= i; j++) {  
            if (i % j == 0) {  
                prima++;  
            }  
        }  
        if (prima == 2)  
            printf("%d\n", i);  
    }  
}
```

Sieve of Eratosthenes

```
void sieveOfEratosthenes (unsigned long n) {  
    unsigned long arrPrime[n];  
    int i, j;  
    for (i = 2; i < n; i++){  
        arrPrime[i] = i;  
    }  
    i = 2;  
    while (pow(i,2) <= n) {  
        j = pow(i,2);  
        while (j <= n){  
            arrPrime[j] = 0;  
            j += i;  
        }  
        do {  
            i++;  
        } while (arrPrime[i] != i);  
    }  
    for (i = 2; i < n; i++) {  
        if (arrPrime[i] != 0)  
            printf("%d\n", arrPrime[i]);  
    }  
}
```

SimplePrime (1000)

```
991
997

real    0m0.005s
user    0m0.001s
sys     0m0.003s
```

SmallPrime (1000)

```
991
997

real    0m0.003s
user    0m0.001s
sys     0m0.000s
```

SimplePrime (1000000)

```
999979
999983

real    32m24.797s
user    32m8.118s
sys     0m3.721s
```

SmallPrime (1000000)

```
999979
999983

real    0m0.453s
user    0m0.073s
sys     0m0.164s
```

Computations Module a Prime

- Basic rule
 - ◆ Compute the number as integer normally
 - ◆ The result is then modulo by p
 - ◆ Any integer taken modulo p is always in range $0, \dots, p-1$
- Applicable to additions, subtractions, and multiplication
 - ◆ $7 + 10 = 2 \pmod{5}$
 - ◆ $9 - 6 = 3 \pmod{5}$
 - ◆ $8 * 3 = 4 \pmod{5}$
- What about division?

Division on numbers modulo p

$\frac{a}{b} \pmod{p}$ is a number c such that $c * b = a \pmod{p}$

Division can be used to proof that two numbers a and b are coprime which will be used in many cryptographic algorithm

GCD (Greatest Common Divisor)

gcd of two numbers a and b is the largest k such that
 $k \mid a$ and $k \mid b$

in other words:

$\text{gcd}(a,b)$ is the largest number that divides both a and b

gcd can be used to detect if both a and b are coprime or not

Coprime

Two numbers (a and b) are called coprime if the only positive integer that evenly divides both of them is 1

OR

$$\gcd(a, b) = 1$$

6 has a factor of 1, 2, 3, and 6

35 has a factor of 1, 5, 7, and 35

So 6 and 35 are coprime

9 and 27 are not coprime since they are divisible by 3

Euclid Algorithm

```
function GCD
input      : a: positive integer
            b: positive integer

output    : k: greatest common divisor of a and b
  assert  $a \geq 0 \wedge b \geq 0$ 
  while  $a \neq 0$  do
     $(a, b) \leftarrow (b \bmod a, a)$ 
  od
  Return b
```

Practice

- $\gcd(4, 28)$
- $\gcd(7, 54)$
- $\gcd(22, 95)$
- $\gcd(81, 33)$

LCM (Least Common Multiple)

lcm of two numbers a and b is the smallest number that is both a multiple of a and a multiple of b

$$\text{lcm}(a, b) = \frac{ab}{\text{gcd}(a, b)}$$

Practice

- $\text{lcm}(50, 42)$
- $\text{lcm}(38, 29)$
- $\text{lcm}(13, 78)$
- $\text{lcm}(60, 34)$

Extended Euclidean Algorithm

- Idea:
 - ◆ Compute $\gcd(a,b)$
 - ◆ Find two integers x and y such that $ax + by = \gcd(a,b)$
- Example: $a = 120, b = 23$
 - ◆ $\gcd(120, 23) = 1$
 - ◆ $120x + 23y = 1$
 - ◆ Result: $x = -9, y = 47$
 - ◆ We can use this later with modular multiplicative inverse
 - $-9 * 120 = 1 \pmod{23}$
 - $47 * 23 = 1 \pmod{120}$

Extended Euclidean

For an integer a to be invertible modulo b , it is necessary that a and b be coprime, so if a GCD greater than 1 is found, one may conclude that such a modular inverse does not exist.

Extended Euclidean can be used to solve modular multiplicative inverse as well

Practice

- Suppose we have : $de \approx 1 \pmod{p}$
- Find d when :
 - ◆ $e = 3$ and $p = 40$
 - ◆ $e = 7$ and $p = 60$

Extended Euclidean Algorithm

```
extendedEuclidean(int a, int b){
    int oldA, oldB, tempX, tempY, quotient, modula;
    oldA = a; oldB = b;
    if (a >= 0 && b >= 0){
        int x, y, lastX, lastY;
        x = 1; y = 0; lastX = 0; lastY = 1;
        while (b != 0) {
            quotient = a / b;
            modula = a % b;
            a = b; b = modula;
            tempX = lastX;
            tempY = lastY;
            lastX = x - quotient * lastX;
            x = tempX;
            lastY = y - quotient * lastY;
            y = tempY;
        }
        return (x, y)
    }
}
```

Modular multiplicative inverse

Modular multiplicative inverse of an integer a modulo m is an integer x such that

$$a^{-1} \approx x \pmod{m}$$

equivalent to

$$ax \approx aa^{-1} \approx 1 \pmod{m}$$

Modular multiplicative inverse

- Suppose we are to find modular multiplicative inverse x of 3 modulo 11.
 - ◆ $3^{-1} \approx x \pmod{11}$
- This is the same as finding x such that
 - ◆ $3x \approx 1 \pmod{11}$
- Solution
 - ◆ $3(4) = 12 \approx 1 \pmod{11}$
- Therefore, the modular multiplicative inverse of 3 modulo 11 is 4.

Extended Euclidean on Modular Multiplicative Inverse

Basic formula: $ax + by = \gcd(a, b)$

modular multiplicative inverse : $ax \approx 1 \pmod{m}$

By lemma 1: $m \mid ax - 1 \Rightarrow m$ is a divisor of $ax - 1$

$$ax - 1 = qm \approx ax - qm = 1$$