Computer Security

# Number Theory

**Hyoungshick Kim**

Department of Software

College of Software

Sungkyunkwan University

# Notation

From here on:

- $N$ denotes a positive integer.
- $p$ denotes a prime.

Notation: $\mathbb{Z}_N = \{0, 1, 2, \ldots, N-1\}$

We can do addition and multiplication modulo $N$

# Modular arithmetic

Examples: let $N = 12$

$9 + 8 = 5$ in $\mathbb{Z}_{12}$

$5 \times 7 = 11$ in $\mathbb{Z}_{12}$

$5 - 7 = 10$ in $\mathbb{Z}_{12}$

Arithmetic in $\mathbb{Z}_N$ works as you expect,
e.g., $x \cdot (y+z) = x \cdot y + x \cdot z$ in $\mathbb{Z}_N$

*distributive law*

# Greatest common divisor

**Def**: For integers x,y, **gcd(x, y)** is the greatest common divisor of x,y

Example: gcd( 12, 18 ) = 6

**Fact**: For all integers x,y, there exist integers a,b such that

$$a \cdot x + b \cdot y = \gcd(x,y) \qquad 2 \cdot 12 + -1 \cdot 18 = 6$$

a,b can be found efficiently using the extended Euclid alg.

If gcd(x,y)=1 we say that x and y are **relatively prime**

# Modular inversion

Over the rational numbers, the inverse of 2 is ½ .
What about $\mathbb{Z}_N$?

**Def**: The **inverse** of x in $\mathbb{Z}_N$ is an element y in $\mathbb{Z}_N$ s.t. x·y=1 in $\mathbb{Z}_N$
y is denoted $x^{-1}$ .

Example: Let *N* be an odd integer. The inverse of 2 in $\mathbb{Z}_N$ is (*N*+1)/2

$$2\cdot(N+1)/2 = N+1 = 1 \text{ in } \mathbb{Z}_N$$

# Modular inversion

Which elements have an inverse in $\mathbb{Z}_N$ ?

**<u>Lemma</u>**: x in $\mathbb{Z}_N$ has an inverse if and only if gcd(x,N) = 1

Proof:

gcd(x,N)=1 $\Rightarrow$ $\exists$ a,b: $a \cdot x + b \cdot N = 1 \Rightarrow a \cdot x = 1$ in $\mathbb{Z}_N$

$\Rightarrow$ $x^{-1} = a$ in $\mathbb{Z}_N$

gcd(x,N) > 1 $\Rightarrow$ $\forall$a: gcd( a·x, N ) > 1 $\Rightarrow$ $a \cdot x \neq 1$ in $\mathbb{Z}_N$

# More notation

**Def:** $\mathbb{Z}_N^*$ = (set of invertible elements in $\mathbb{Z}_N$ ) =

= { x∈ $\mathbb{Z}_N$ : gcd(x,N) = 1 }

Examples:

1. for prime p, $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \ldots, p-1\}$
2. $\mathbb{Z}_{12}^*$ = { 1, 5, 7, 11}

For x in $\mathbb{Z}_N^*$ , we can find $x^{-1}$ using the extended Euclidean algorithm.

# Euclid's rule for GCD

- How do you find the greatest common divisor of two integers?
  - Select the largest factor that can divide both
  - A brute force approach requires exponential time
- Ancient Greece mathematician Euclid discovered the rule:
  - If x and y are positive integers with $x \geq y$ then

    $$\gcd(x,y) = \gcd(y, x \bmod y)$$

  - When $x \bmod y = 0$, then y is the gcd
- We can swap parameters each time to keep largest as 1st parameter

# Euclid's algorithm

- This rule leads to the following algorithm

<u>Function Euclid</u> ($a$,$b$)
Input: Two integers $a$ and $b$ with $a \geq b \geq 0$ ($n$-bit integers)
Output: gcd($a$,$b$)

if $b$=0: return $a$
else: return Euclid($b$, $a$ mod $b$)

# Example of Euclid's algorithm

- As an example of the running of Euclid, consider the computation of gcd(30, 21):

$$
\begin{aligned}
\text{Euclid}(30, 21) &= \text{Euclid}(21, 9) \\
&= \text{Euclid}(9, 3) \\
&= \text{Euclid}(3, 0) \\
&= 3
\end{aligned}
$$

# C code

```c
#include <stdio.h>

int gcd_algorithm(int x, int y)
{
   if (y == 0)
      return x;

      return gcd_algorithm(y, (x % y));
}
```

```c
int main(void)
{
   int num1, num2, gcd;
   printf("\nEnter two numbers: ");
   scanf("%d%d", &num1, &num2);

   if (num1<0) num1 = -num1;
   if (num2<0) num2 = -num2;
   gcd = gcd_algorithm(num1, num2);

   if (gcd)
      printf("\nThe GCD of %d and %d is %d\n", num1, num2, gcd);
   else
      printf("\nInvalid input!!!\n");
   return 0;
}
```

# Extended Euclid's Algorithm

function extended-Euclid ($a$, $b$)

Input: Two positive integers $a$ and $b$ with $a \geq b \geq 0$ ($n$-bits)

Output: Integers $x$, $y$, $d$ such that $d = \gcd(a, b)$
and $ax + by = d$

if $b = 0$: return $(1, 0, a)$

$(x', y', d)$ = extended-Euclid($b$, $a$ mod $b$)

return $(y', x' - \mathrm{floor}(a/b)y', d)$

# C code

```
int gcdExtended(int a, int b, int *x, int *y)
{
   // Base Case
   if (b == 0)
   {  *x = 1;
      *y = 0;
      return a;
   }
   int x1, y1; // To store results of recursive call
   int gcd = gcdExtended(b, a %b, &x1, &y1);

   // Update x and y using results of recursive
   // call
   *x = y1;
   *y = x1 - (a/b) * y1;

   return gcd;
}
```

```
int main()
{
   int x, y;
   int a = 30, b = 21;
   int g = gcdExtended(a, b, &x, &y);
   printf("gcd(%d, %d) = %d, x = %d, y = %d",
         a, b, g, x, y);
   return 0;
}
```

```
gcd(30, 21) = 3, x = 3, y = -2

...Program finished with exit code 0
Press ENTER to exit console.
```

# Multiplicative inverses

- Two numbers *a* and *b* are relatively prime if gcd(*a*,*b*) = 1
- If *a* and *N* are relatively prime, then we know the multiplicative inverse exists (e.g. 4 mod 7)
  - If *a* and *N* are relatively prime, then $ax + Ny = 1$
- Extended-Euclid(*a*,*N*) can be used:
  - Returns integers *x*, *y*, *d* such that $d = \gcd(a, b)$ and $ax + by = d$
  - First, it must return $d$=1 as the gcd to confirm that *a* and *N* are relatively prime
- $Ny = 0 \pmod{N}$ for all integers *y*
- Thus, $ax \equiv 1 \pmod{N}$ // *ax* is congruent to 1 modulo *N*
- Then *x* is the multiplicative inverse of *a* modulo *N*

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> ($a$, $b$)
if $b = 0$: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x' -$ floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

# Multiplicative inverse of 20 Mod 79

function extended-Euclid ($a$, $b$)
if $b$ = 0: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| | | | | | | | |
| | | | | | | | |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> ($a$, $b$)
if $b = 0$: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| 19 | 1 | | | | | | |
| | | | | | | | |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> (*a*, *b*)
if *b* = 0: return (1, 0, *a*)
(*x'*, *y'*, *d*) = extended-Euclid(*b*, *a* mod *b*)
return (*y'*, *x'* - floor(*a*/*b*)*y'*, *d*)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| 19 | 1 | | | | | | |
| 1 | 0 | | | | | | |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> (*a*, *b*)
if *b* = 0: return (1, 0, *a*)
(*x′*, *y′*, *d*) = extended-Euclid(*b*, *a* mod *b*)
return (*y′*, *x′* - floor(*a*/*b*)*y′*, *d*)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| 19 | 1 | | | | | | |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

function extended-Euclid ($a$, $b$)
if $b = 0$: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| 19 | 1 | 1 | 0 | 1 | | | |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

function extended-Euclid ($a$, $b$)
if $b$ = 0: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | | | | | | |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid </u>($a$, $b$)
if $b$ = 0: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | 0 | 1 | 1 | | | |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

function extended-Euclid ($a$, $b$)
if $b = 0$: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | | | | | | |
| 20 | 19 | 0 | 1 | 1 | 1 | -1 | 1 |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

function extended-Euclid ($a$, $b$)
if $b$ = 0: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | 1 | -1 | 1 | | | |
| 20 | 19 | 0 | 1 | 1 | 1 | -1 | 1 |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> (*a*, *b*)
if *b* = 0: return (1, 0, *a*)
(*x'*, *y'*, *d*) = extended-Euclid(*b*, *a* mod *b*)
return (*y'*, *x'* - floor(*a*/*b*)*y'*, *d*)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | 1 | -1 | 1 | -1 | 4 | 1 |
| 20 | 19 | 0 | 1 | 1 | 1 | -1 | 1 |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

# Multiplicative inverse of 20 Mod 79

<u>function extended-Euclid</u> ($a$, $b$)
if $b = 0$: return (1, 0, $a$)
($x'$, $y'$, $d$) = extended-Euclid($b$, $a$ mod $b$)
return ($y'$, $x'$ - floor($a/b$)$y'$, $d$)

| $a$ | $b$ | $x'$ | $y'$ | $d$ | ret 1 | ret 2 | ret 3 |
|---|---|---|---|---|---|---|---|
| 79 | 20 | 1 | -1 | 1 | -1 | 4 | 1 |
| 20 | 19 | 0 | 1 | 1 | 1 | -1 | 1 |
| 19 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | | | | 1 | 0 | 1 |

**$ax + Ny = 1 = 20(4) + 79(-1)$**
**Thus $x = a^{-1} = 4$**

# How can we choose a random prime?

- A number p is *prime* if
  - p is an integer and p > 1
  - The only (positive) factors of p are 1 and p.

- If n > 1 is not prime, it is *composite*
  - n has a positive factor other than 1 or n.

**Q. Can you develop a practical algorithm?**

# Brute-force algorithm

- Input: integer $N > 1$
- Output: $(x,y)$ such that $x > 1$, $y > 1$ and $xy = N$

  or "Prime" if $N$ is prime

- For $x = 2$ to $N-1$
  - If $x$ evenly divides $N$
    - Return$(x, N/x)$
- Return("Prime")

# Slightly Better Algorithm

- Input: integer N > 1
- Output: (x,y) such that x > 1, y > 1 and xy = N

  or "Prime" if N is prime

- For x = 2 to ~~N-1~~ $\sqrt{N}$
  - If x evenly divides N
    - Return(x, N/x)
- Return("Prime")

*If N has 200 digits*
*$\sqrt{N}$ has 100 digits*
*If inner loop takes .1 ns*
*Algorithm take 3 x $10^{82}$ years*

*Time proportional to the value of N, not the number of digits*

# PRIMES is in P

PRIMES is in P

Manindra Agrawal, Neeraj Kayal and Nitin Saxena*

Department of Computer Science & Engineering
Indian Institute of Technology Kanpur
Kanpur-208016, INDIA

August 6, 2002

**A. We can develop an efficient algorithm in $O(n^{12})$.**

**Best: $O(n^6)$.**

# Practical (randomized) algorithm

Suppose we want to generate a large random prime

say, prime p of length 1024 bits (i.e., $p \approx 2^{1024}$ )

Step 1: Choose a random integer $p \in [\ 2^{1024}\ ,\ 2^{1025}-1\ ]$

Step 2: Test if $2^{p-1} = 1$ in $\mathbb{Z}^*_p$

If so, output p and stop. If not, goto step 1 .

Simple algorithm (not the best)

# Fermat's theorem (1640)

**<u>Thm</u>:** Let p be a prime

$$\forall\, x \in \mathbb{Z}^*_p : \quad x^{p-1} = 1 \text{ in } \mathbb{Z}_p$$

Example: p=5. $3^4 = 81 = 1$ in $\mathbb{Z}_5$

So: $x \in \mathbb{Z}^*_p \Rightarrow \; x \cdot x^{p-2} = 1 \; \Rightarrow \; x^{-1} = x^{p-2}$ in $\mathbb{Z}_p$

another way to compute inverses, but less efficient than Euclidian algorithm

# Questions?


Sungkyunkwan
Campus
Social
Graduate
campus
Sciences
University
Seoul
Natural
Suwon
School
Korean
established
university
edit
constructed
English
Professors
Confucian
Building
Global
Institute
Advanced
Humanities
lecturers
Education
Engineering
located
Joseon
schools
Science
Administration
Academic
ASLE-Korea
also
two
Zone
Samsung
symbol
trees
heart
now
one