# CS 413
# Information Security

*Course Instructor*

Naushad Siddiqui

Head of IT Audit, Sui Southern Gas Company

Visiting Faculty, NED University

Director & Faculty Member, ISACA Karachi

# CS413 Information Security

Fall 2020

**Week 03**

## Agenda

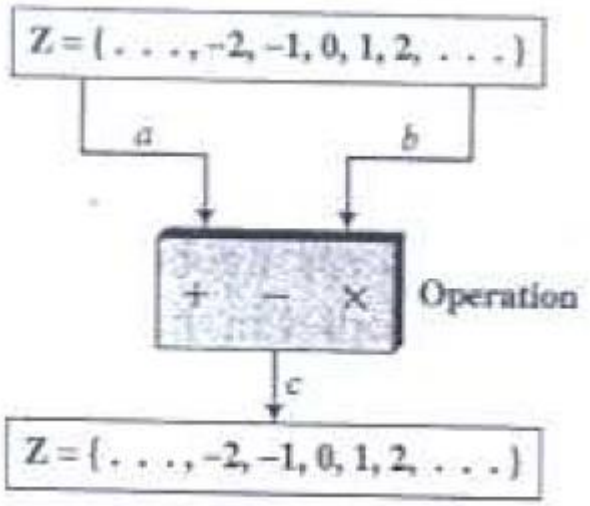
- Integer Arithmetic
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Arithmetic
- Affine Cipher

# Mathematics of Cryptography

# Integer Arithmetic

## *Basic Concepts Revision*

- Set of Integers
- Three Binary Operations
  Addition, Subtraction, Multiplication




- Division does not fit into this category as it produces TWO outputs

# Integer Arithmetic (2)

## *Basic Concepts Revision (2)*

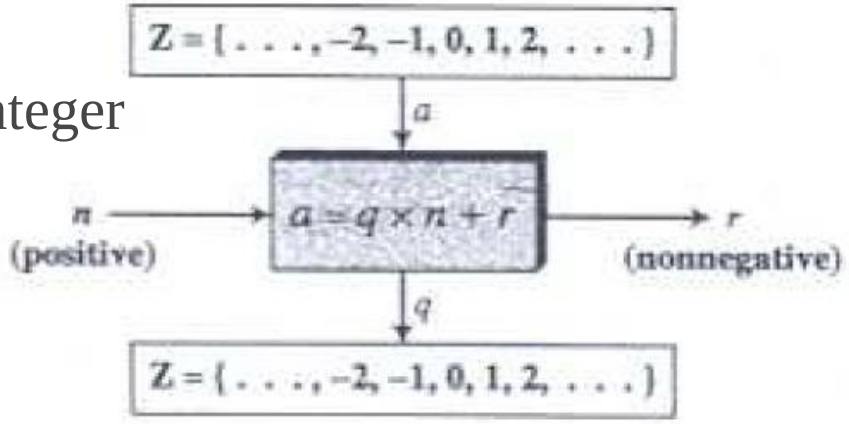
### *Integer Division*

- If we divide a by n

$$a = qn + r \qquad 0 \leq r < n; q = \lfloor a/n \rfloor$$

'a' is dividend; 'q' is quotient, 'n' is divisor; 'r' remainder

- Restrictions
  a) Divisor be a positive integer
  b) Quotient be a non-negative integer
- Division Algorithm for Integers




- Examples

# Integer Arithmetic (3)

## *Basic Concepts Revision (3)*

### *Divisibility*

- When ‘a’ is not ‘0’ and we let r = 0, then **a = q x n**
- We can say that ‘n’ is DIVISIBLE by ‘a’ and we write **a | n**
- When ‘r’ is not zero, then ‘n’ is NOT DIVISIBLE by ‘a **a ∤ n**
- **Properties of Divisibility**

**Property 1:** if $a|1$, then $a = \pm 1$.
**Property 2:** if $a|b$ and $b|a$, then $a = \pm b$.
**Property 3:** if $a|b$ and $b|c$, then $a|c$.
**Property 4:** if $a|b$ and $a|c$, then $a|(m \times b + n \times c)$, where $m$ and $n$ are arbitrary integers.

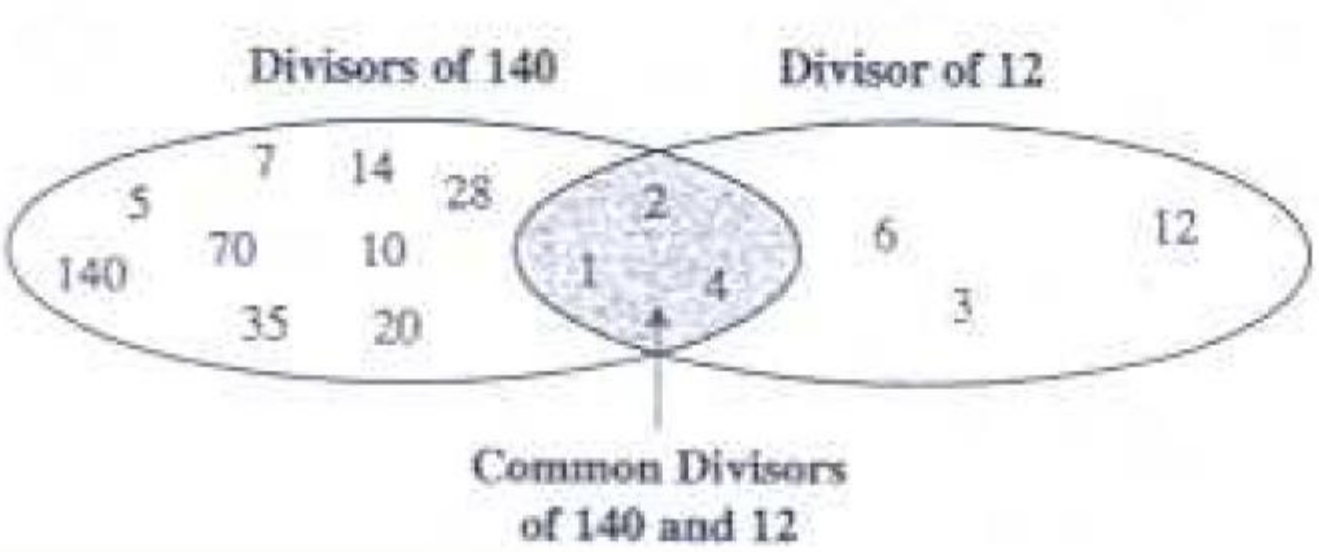
### *All Divisors*

- A positive integer can have more than one divisors
  - Integer ‘1’ has only ONE divisor i.e. itself
  - Any positive integer has at least two divisors, ‘1’ and itself

# Integer Arithmetic (4)

## *Greatest Common Divisor (GCD)*

- Greatest common divisors are needed in Cryptography
- Two positive integers can have many common divisors
- gcd (a, b) of ‘a’ and ‘b’ is the largest number that divides evenly into both ‘a’ and ‘b’



# Integer Arithmetic (5)

## *Coprime*

- In number theory, two integers ‘a’ and ‘b’ are relatively prime, mutually prime, or coprime if the only positive integer that evenly divides both of them is 1.
- It means ‘a’ is prime to ‘b’ or ‘a’ is coprime with ‘b’. Consequently, any prime number that divides one of ‘a’ or ‘b’ does not divide the other.
- Any prime number is a coprime number of every other integer by definition; hence, any integer has an infinite number of coprime numbers.
- ***Example***: 14 (2 × 7) and 9 (3 × 3) are coprime, yet neither is prime.

# Euclidean Algorithm

- Euclidean algorithm is a simple procedure for determining gcd of two positive integers. Use the notation gcd (a, b) to mean the greatest common divisor of ‘a’ and ‘b’.
- As discussed earlier, two integers ‘a’ and ‘b’ are relatively prime if their only common positive integer factor is 1, i.e. gcd (a, b) = 1

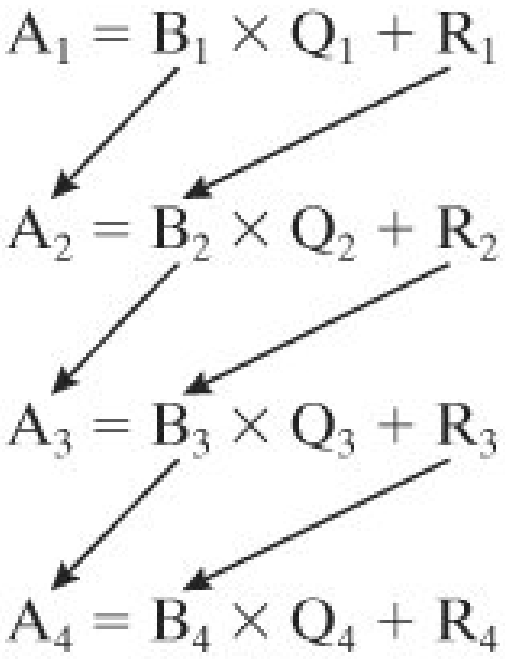
- The Euclidean Algorithm is based on the following theorem

For any nonnegative integers ‘a’ and ‘b’
gcd (a, 0) = a
gcd (a, b) = gcd (b, r)

where ‘r’ is the remainder of dividing ‘a’ by ‘b’

# Euclidean Algorithm (2)

- In other words, the theorem can be stated as
- gcd (a, b) = gcd (b, a mob b)




**Example:**

gcd (36, 10) = gcd (10, 6) = gcd (6, 4) = gcd (4,2) = gcd (2, 0) = 2

# Euclidean Algorithm (3)

***Example gcd(1970, 1066)***

| | |
|---|---|
| 1970 = 1 x 1066 + 904 | gcd(1066, 904) |
| 1066 = 1 x 904 + 162 | gcd(904, 162) |
| 904 = 5 x 162 + 94 | gcd(162, 94) |
| 162 = 1 x 94 + 68 | gcd(94, 68) |
| 94 = 1 x 68 + 26 | gcd(68, 26) |
| 68 = 2 x 26 + 16 | gcd(26, 16) |
| 26 = 1 x 16 + 10 | gcd(16, 10) |
| 16 = 1 x 10 + 6 | gcd(10, 6) |
| 10 = 1 x 6 + 4 | gcd(6, 4) |
| 6 = 1 x 4 + 2 | gcd(4, 2) |
| 4 = 2 x 2 + 0 | gcd(2, 0) |

**So GCD is ‘2’**

# Euclidean Algorithm (4)

***Examples and Class Activity*** *(from book)*

***Euclidean Algorithm – programming assignment***

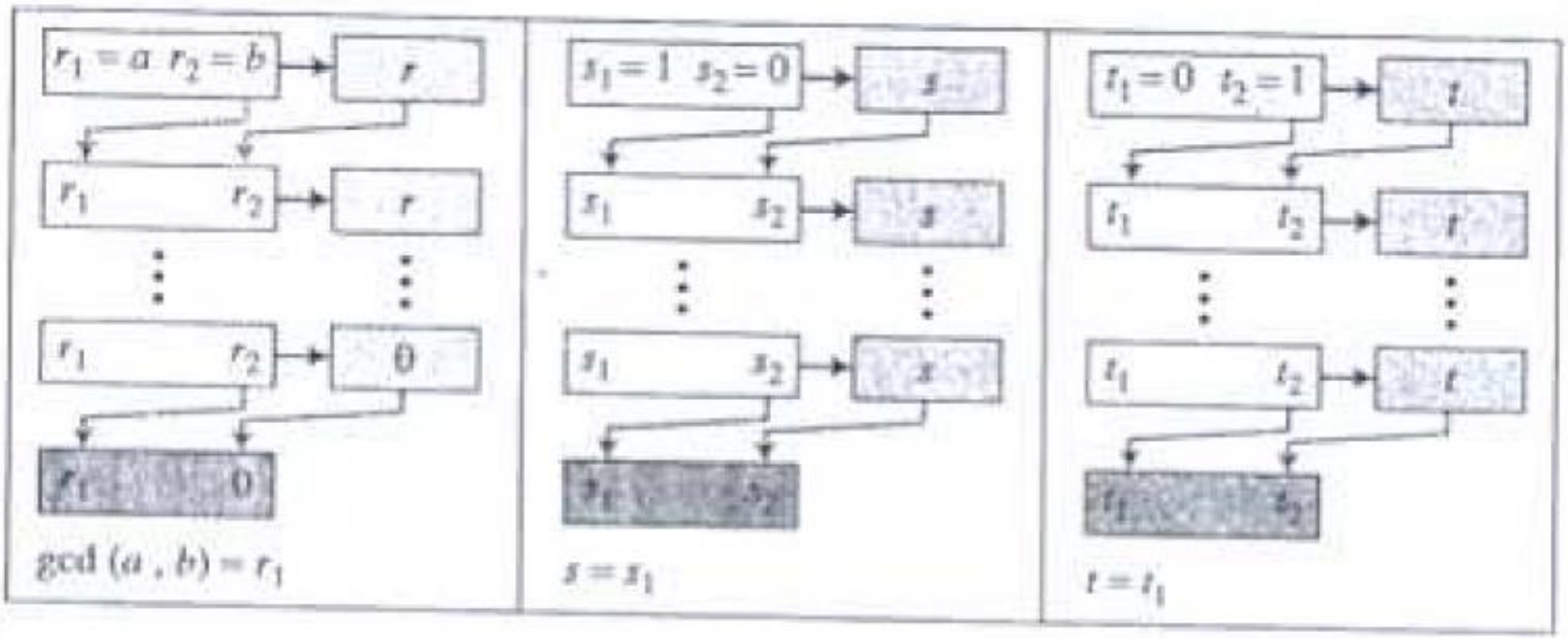
*Draw a flowchart and an algorithm and write a Python program to find the GCD of two numbers.*

# Extended Euclidean Algorithm

- Extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the GCD of integers ‘a’ and ‘b’, also the coefficients of Bézout's identity, which are integers ‘s’ and ‘t’ such that **s x a + t x b = gcd (a, b)**
- GCD is the only number that can simultaneously satisfy this equation and divide the inputs. It allows one to compute also, with almost no extra cost, the quotients of ‘a’ and ‘b’ by their GCD.
- It is particularly useful when ‘a’ and ‘b’ are coprime. It can be used to find out the modular multiplicative inverse.

# Extended Euclidean Algorithm (2)

## *Extended Euclidean Process*




r = r1 – q x r2 s = s1 – q x s2 t = t1 – q x t2

# Extended Euclidean Algorithm (3)

***Examples and Class Activity*** *(from book)*

# Modular Arithmetic

- As discussed earlier, we know that any positive integer ‘n’ and any nonnegative integer ‘a’, if we divide ‘a’ by ‘n’, we get an integer quotient ‘q’ and an integer remainder ‘r’.

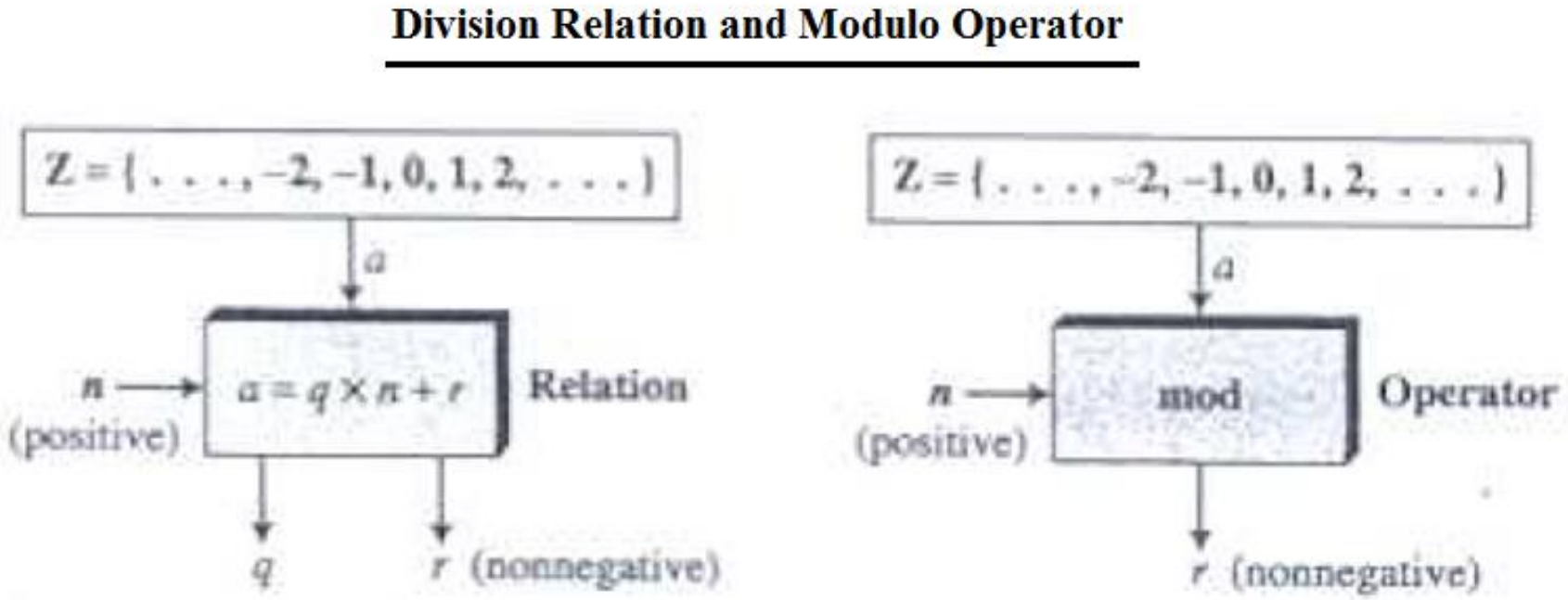
  **a = q x n + r**

- In Modular Arithmetic we are only interested in the remainder (or residue) after division by some modulus, and results with the same remainder are regarded as equivalent.

- **Modulo operator** “a mod n” to be remainder when ‘a’ is divided by ‘n’
- ‘n’ is called ***Modulus***
- Output ‘r’ is ***Residue***

  0 <= r <= n-1

# Modular Arithmetic (2)



# Modular Arithmetic (3)

## *Modular Arithmetic Operations*

- 'clock arithmetic’
- Modular arithmetic is where we perform arithmetic operations within the confines of some set of integers mod n.
- It uses a finite number of values, and loops back from either end where needed.
- When reducing, we "usually" want to find the positive remainder after dividing by the modulus.
- For positive numbers, this is simply the normal remainder. For negative numbers we have to "overshoot" (ie find the next multiple larger than the number) and "come back" (ie add a positive remainder to get the number); rather than have a "negative remainder".

# Modular Arithmetic (4)

## *Modulo 8 Addition Example*

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 |
| 3 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 4 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 |
| 5 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |
| 6 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 |
| 7 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

# Modular Arithmetic (5)

## *Set of Residues $Z_n$*

We know that result of ‘a’ mod ‘n’ is a non-negative integer, less than n.

In modular arithmetic with any group of integers modulo operation creates a set of residues. In other words,

$Z_n = \{0, 1, \ldots, n-1\}$

For example:

- $Z_2 = \{0, 1\}$
- $Z_5 = \{0, 1, 2, 3, 4\}$
- $Z_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

# Modular Arithmetic (6)

## *Congruence*

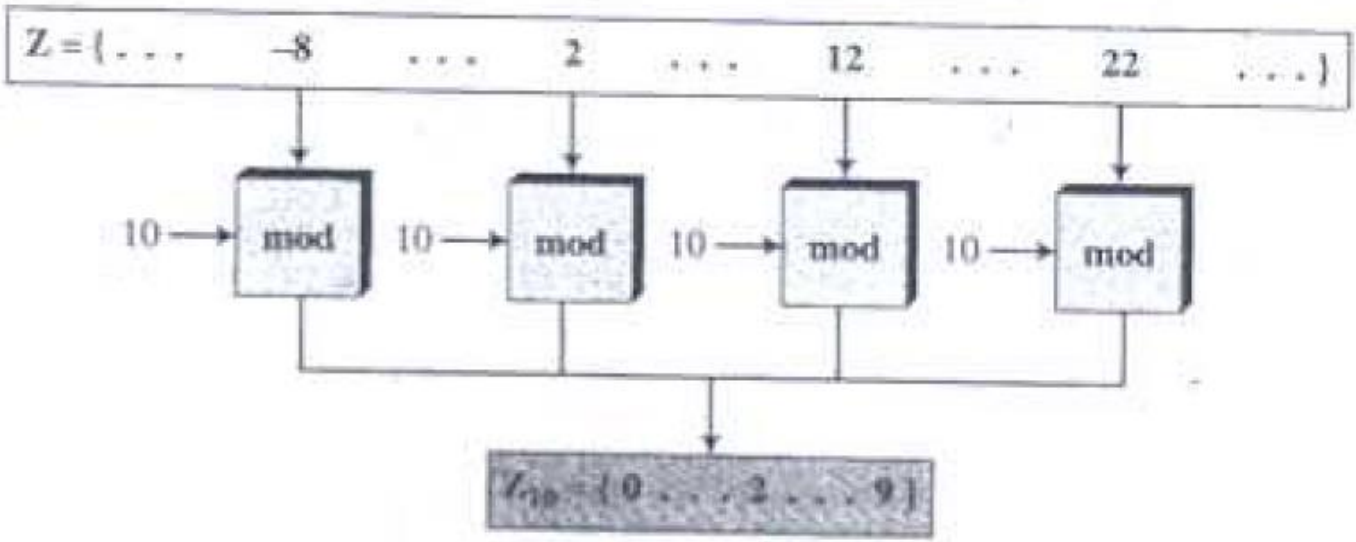
- In cryptography, we use the concept of 'Congruence' instead of equality
- Mapping from Z to $Z_n$ is not one-to-one
- Infinite members of Z can map to one member of $Z_n$
- Two integers 'a' and 'b' are said to be congruent modulo 'n', if
  (a mod n) = (b mod n)
- OR a = b mod n i.e. when divided by *n*, a & b have same remainder

- E.g. 2 mod 10 = 12 mod 10 = 22 mod 10 = 2

- We use symbol 

# Modular Arithmetic (7)

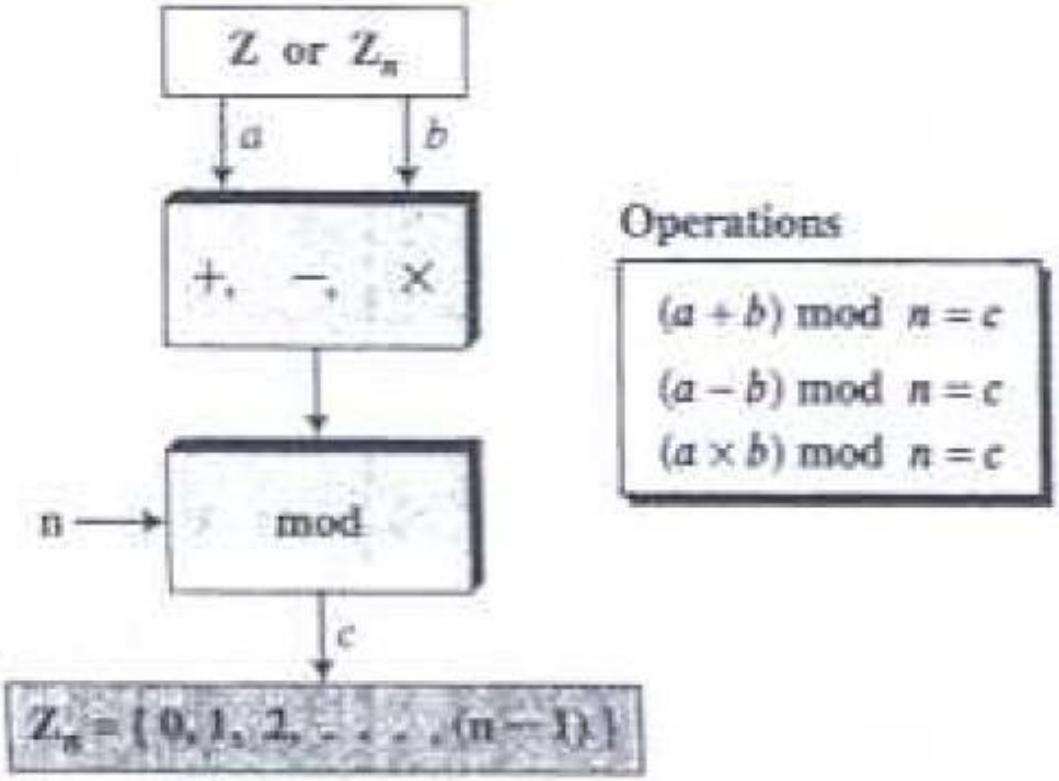
## *Congruence*




Congruence Relationship

# Modular Arithmetic (8)

## *Operations in $Z_n$*



# Modular Arithmetic (9)

## *Operations in $Z_n$*

***Examples and Class Activity*** *(from book)*

# Modular Arithmetic (10)

## *Additive Inverse*

- In $Z_n$, two numbers ‘a’ and ‘b’ are said to be additive inverse if

  $a + b = 0 \bmod n$

  i.e. the sum of a and b is Congruent to 0 module n

***Examples***

- All additive inverse pairs in $Z_{10}$

  (0,0), (1,9), (2,8), (3,7), (6,4), (5,5)

# Modular Arithmetic (11)

## *Multiplicative Inverse*

- In $Z_n$, two numbers ‘a’ and ‘b’ are said to be multiplicative inverse if

  a x b = 1 mod n

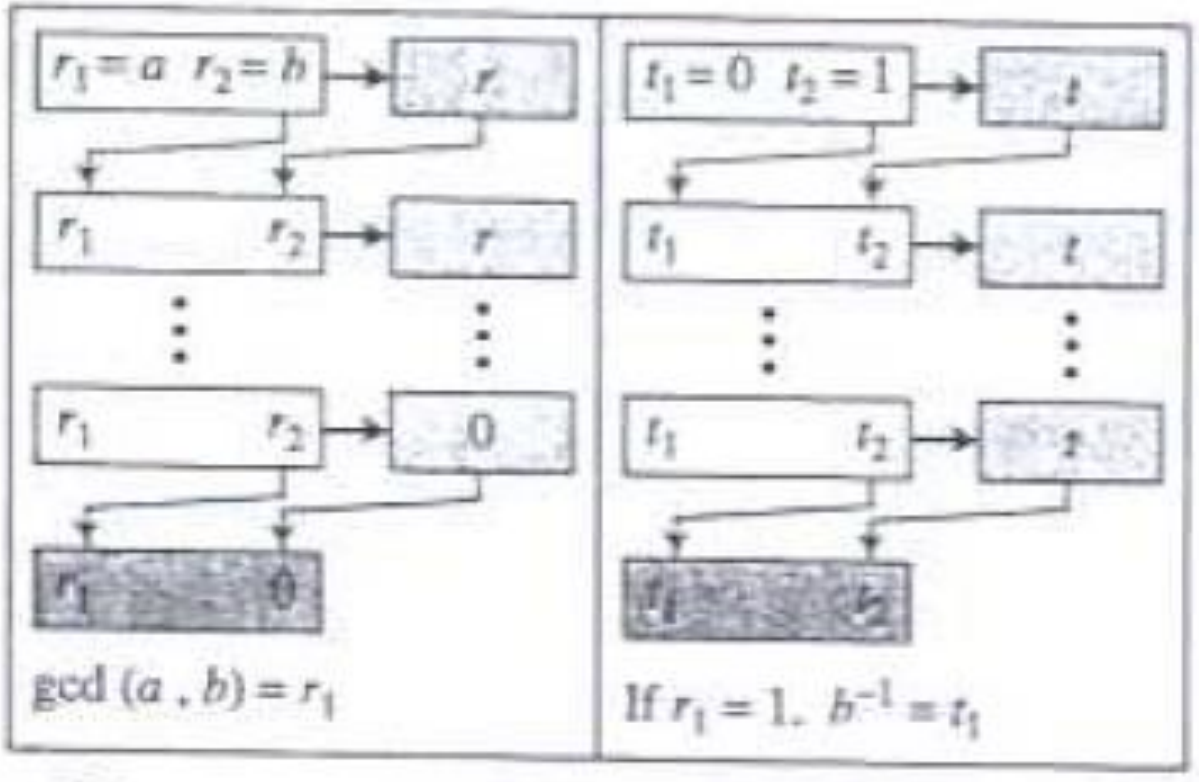
- In modular arithmetic, an integer may or may not have a multiplicative inverse

- In $Z_{10}$, the multiplicative inverse of 3 is 7

  OR in other words 3 x 7 mod 10 = 1

# Modular Arithmetic (12)

***Using Extended Euclidian Method to Find Multiplicative Inverse***



# Modular Arithmetic (13)

***Using Extended Euclidian Method to Find Multiplicative Inverse***

## *Examples and Class Activity* *(from book)*

*Finding MI of 11 in modulus 26*

| $q$ | $r_1$ | $r_2$ | $r$ | $t_1$ | $t_2$ | $t$ |
|---|---|---|---|---|---|---|
| 2 | 26 | 11 | 4 | 0 | 1 | −2 |
| 2 | 11 | 4 | 3 | 1 | −2 | 5 |
| 1 | 4 | 3 | 1 | −2 | 5 | −7 |
| 3 | 3 | 1 | 0 | 5 | −7 | 26 |
| | 1 | 0 | | −7 | 26 | |

The gcd (26, 11) is 1, which means that the multiplicative inverse of 11 exists. The extended Euclidean algorithm gives $t_1 = -7$. The multiplicative inverse is $(-7) \bmod 26 = 19$. In other words, 11 and 19 are multiplicative inverse in $Z_{26}$. We can see that $(11 \times 19) \bmod 26 = 209 \bmod 26 = 1$.

# Affine Ciphers

# Affine Cipher

- An encipherment scheme (or algorithm) of the form

  $E(x) = (ax + b) \text{ MOD } 26$

  is called an affine cipher. Here x is the numerical equivalent of the given plaintext letter, and a and b are (appropriately chosen) integers.
- Recall that the numerical equivalents of the letters are as follows:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

# Affine Cipher (2)

- The function E(x) = (ax + b) MOD 26 defines a valid affine cipher if a is relatively prime to 26, and b is an integer between 0 and 25, inclusive.
- Note that if a = 1, then E(x) = (x + b) MOD 26 is simply a Caesar (+b) shift cipher.

# Affine Cipher (3)

## *Encipherment Example*

Encipher ITS COOL using E(x) = (5x + 8) MOD 26

Solution:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

- Using the above table and the formula, gives:

| plain | I | T | S | C | O | O | L |
|---|---|---|---|---|---|---|---|
| $x$ | 8 | 19 | 18 | 2 | 14 | 14 | 11 |
| $5x + 8$ | 48 | 103 | 98 | 18 | 78 | 78 | 63 |
| $(5x + 8)$ MOD 26 | 22 | 25 | 20 | 18 | 0 | 0 | 11 |
| cipher | W | Z | U | S | A | A | L |

# Affine Cipher (4)

## *Decipherment*

- If y = E(x) = (ax+b) MOD 26, then we can "solve for x in terms of y" and so determine $E^{-1}(y)$.
- That is, if $y \equiv (ax + b) \pmod{26}$, then $y - b \equiv ax \pmod{26}$, or equivalently $ax \equiv (y - b) \pmod{26}$
- Using our earlier results, we see that if we multiply both sides by $a^{-1} \pmod{26}$, then $x \equiv a^{-1}(y - b) \pmod{26}$
- So our decipherment function is

  $E^{-1}(y) = a^{-1}(y - b) \text{ MOD } 26$

# Affine Cipher (5)

## *Decipherment Example*

Decipher HPCCXAQ if the encipherment function is

$E(x) = (5x + 8) \text{ MOD } 26$

Solution:

| A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

- Since $5x \equiv 1 \pmod{26}$ is solved with $x \equiv 21 \pmod{26}$,
  we see $5^{-1} \pmod{26} = 21$
  Therefore, $E^{-1}(y) = 21\ (y - 8) \text{ MOD } 26$
- Using this formula, gives:

| cipher | H | P | C | C | X | A | Q |
|---|---|---|---|---|---|---|---|
| $y$ | 7 | 15 | 2 | 2 | 23 | 0 | 16 |
| $y - 8$ | -1 | 7 | -6 | -6 | 15 | -8 | 8 |
| $21(y - 8)$ | -21 | 147 | -126 | -126 | 315 | -168 | 168 |
| $21(y - 8)$ MOD 26 | 5 | 17 | 4 | 4 | 3 | 14 | 12 |
| plain | F | R | E | E | D | O | M |

End of Week 03