Information Security
CSD-410 **Unit-II (Part-2)**



Instructor: **Dr. Lokesh Chouhan**

Assistant Professor

Computer Science and Engineering (CSE) Department

National Institute of Technology (NIT)

Hamirpur (H.P.) INDIA

E-Mail: Lokeshchouhan@gmail.com, lokesh@nith.ac.in

Mob: +91-898924399, 9827235155

Unit 2

Number Theory and Prime Numbers

Introduction

- will now introduce **finite fields**
- of increasing importance in cryptography
  - AES, Elliptic Curve, IDEA, Public Key
- concern operations on "numbers"
  - where what constitutes a "number" and the type of operations varies considerably
- start with concepts of groups, rings, fields from abstract algebra

Group

- a set of elements or "numbers"
- with some operation whose result is also in the set (closure)
- obeys:
  - associative law: $(a.b).c = a.(b.c)$
  - has identity e: $e.a = a.e = a$
  - has inverses a^{-1}: $a.a^{-1} = e$
- if commutative $a.b = b.a$
  - then forms an **abelian group**

Cyclic Group

- define **exponentiation** as repeated application of operator
  - example: $a^3 = a.a.a$
- and let identity be: $e=a^0$
- a group is **cyclic** if every element is a power of some fixed element
  - ie $b = a^k$ for some a and every b in group
- **a** is said to be a generator of the group

Ring

- a set of “numbers” with two operations (addition and multiplication) which are:
- an abelian group with addition operation
- multiplication:
  - has closure
  - is associative
  - distributive over addition: `a(b+c) = ab + ac`
- if multiplication operation is commutative, it forms a **commutative ring**
- if multiplication operation has inverses and no zero divisors, it forms an **integral domain**

Field

- a set of numbers with two operations:
  - abelian group for addition
  - abelian group for multiplication (ignoring 0)
  - ring

Modular Arithmetic

- define **modulo operator** `a mod n` to be remainder when a is divided by n
- use the term **congruence** for: `a ≡ b mod n`
  - when divided by *n,* a & b have same remainder
  - eg. 100 = 34 mod 11
- b is called the **residue** of a mod n
  - since with integers can always write: `a = qn + b`
- usually have `0 <= b <= n-1`

`-12 mod 7 ≡ -5 mod 7 ≡ 2 mod 7 ≡ 9 mod 7`

Modulo 7 Example

...

-21	-20	-19	-18	-17	-16	-15
-14	-13	-12	-11	-10	-9	-8
-7	-6	-5	-4	-3	-2	-1
0	**1**	**2**	**3**	**4**	**5**	**6**
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	32	33	34

...

Divisors

- say a non-zero number b **divides** a if for some m have $a=mb$ (a, b, m all integers)
- that is b divides into a with no remainder
- denote this $b \mid a$
- and say that b is a **divisor** of a
- eg. all of 1,2,3,4,6,8,12,24 divide 24

Modular Arithmetic Operations

- is 'clock arithmetic'
- uses a finite number of values, and loops back from either end
- modular arithmetic is when do addition & multiplication and modulo reduce answer
- can do reduction at any point, ie
  - `a+b mod n = [a mod n + b mod n] mod n`

Modular Arithmetic

- can do modular arithmetic with any group of integers: $Z_n = \{0, 1, \ldots, n-1\}$
- form a commutative ring for addition
- with a multiplicative identity
- note some peculiarities
  - if $(a+b) \equiv (a+c) \mod n$ **then** $b \equiv c \mod n$
  - but $(ab) \equiv (ac) \mod n$ **then** $b \equiv c \mod n$ only if a is relatively prime to n

Modular Arithmetic Operations

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

e.g.

$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$ $(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$ $(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$ $(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

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

Modulo 8 Multiplication

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is the largest number that divides evenly into both a and b
  - eg GCD(60,24) = 12
- often want **no common factors** (except 1) and hence numbers are **relatively prime**
  - eg GCD(8,15) = 1
  - hence 8 & 15 are relatively prime

Euclid's GCD Algorithm

- an efficient way to find the GCD(a,b)
- uses theorem that:
  - `GCD(a,b) = GCD(b, a mod b)`
- **Euclid's Algorithm** to compute GCD(a,b):
  - `A=a, B=b`
  - `while B>0`
    - `R = A mod B`
    - `A = B, B = R`
  - `return A`

```
A=10, B=45
While(45>0)
R=10mod45=10
A=45, B=10
While(10>0)
R=45mod10=5
A=10, B=5
While(5>0)
R=10mod5
A=5, B=0
Return A=5
```

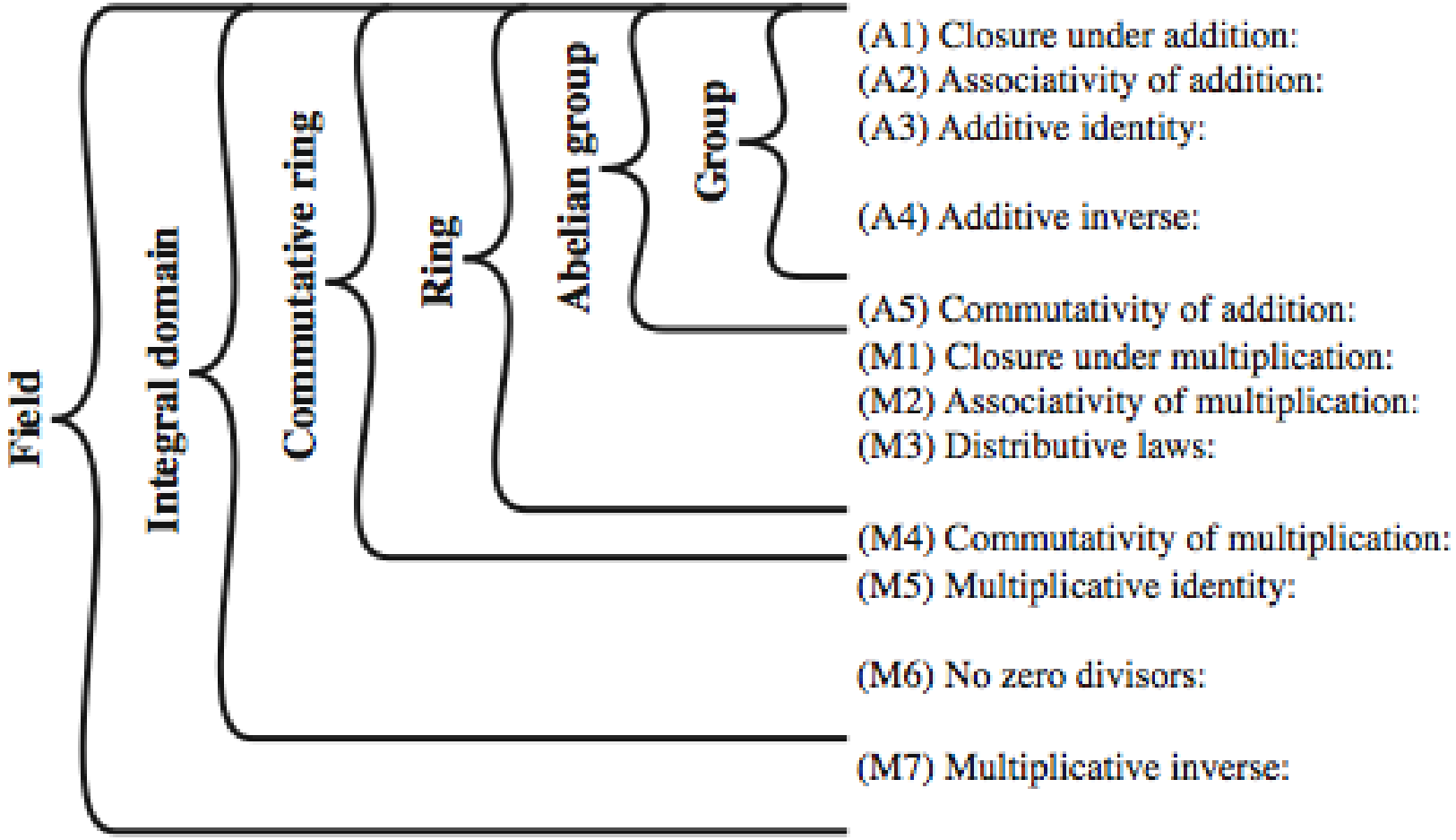
Example GCD(1970,1066)

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)

Field

- a set of numbers
- with two operations which form:
  - abelian group for addition
  - abelian group for multiplication (ignoring 0)
  - ring
- have hierarchy with more axioms/laws
  - group -> ring -> field

Group, Ring, Field



Galois Fields

- finite fields play a key role in cryptography
- can show **number of elements** in a finite field **must** be a power of a **prime p^n**
- known as **Galois fields**
- denoted **GF(p^n)**
- in particular often use the fields:
  - GF(p)
  - GF(2^n)

Galois Fields GF(p)

- **GF(p)** is the set of integers **{0,1, … , p-1}** with arithmetic operations **modulo prime p**
- these form a finite field
  - since have multiplicative inverses
- hence arithmetic is "well-behaved" and can do addition, subtraction, multiplication, and division without leaving the field GF(p)

Example GF(7)

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

Finding Inverses

- can extend Euclid's algorithm:

```
EXTENDED EUCLID(m, b)        m= 550, b=1759
1. (A1, A2, A3)=(1, 0, m);
   (B1, B2, B3)=(0, 1, b)
2. if B3 = 0
   return A3 = gcd(m, b); no inverse
3. if B3 = 1
   return B3 = gcd(m, b); B2 = b^-1 mod m
4. Q = A3 div B3
5. (T1, T2, T3)=(A1 - Q B1, A2 - Q B2, A3 - Q B3)
6. (A1, A2, A3)=(B1, B2, B3)
7. (B1, B2, B3)=(T1, T2, T3)
8. goto 2
```

Inverse of 550 in GF(1759)

$$550 \times x \equiv 1 \bmod 1759$$

Q	A1	A2	A3	B1	B2	B3
–	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	355	1

Polynomial Arithmetic

- can compute using polynomials

  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum a_i x^i$

  - nb. not interested in any specific value of x
  - which is known as the indeterminate
- several alternatives available
  - ordinary polynomial arithmetic
  - poly arithmetic with coords mod p
  - poly arithmetic with coords mod p and polynomials mod m(x)

Ordinary Polynomial Arithmetic

- add or subtract corresponding coefficients
- multiply all terms by each other
- eg

  let $f(x) = x^3 + x^2 + 2$ and

  $g(x) = x^2 - x + 1$

  - *Addition:* $f(x) + g(x) = x^3 + 2x^2 - x + 3$
  - Subtraction: $f(x) - g(x) = x^3 + x + 1$
  - *Multiplication:* $f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$

Polynomial Arithmetic with Modulo Coefficients

- when computing value of each coefficient do calculation modulo some value
  - forms a polynomial ring
- could be modulo any prime
- but we are most interested in mod 2
  - Ie, all coefficients are 0 or 1
  - eg. let $f(x) = x^3 + x^2$ and $g(x) = x^2 + x + 1$

- *Addition:* $f(x) + g(x) = x^3 + x + 1$
- *Multiplication:* $f(x) \times g(x) = x^5 + x^2$

Polynomial Division

- can write any polynomial in the form:
  - $f(x) = q(x)\, g(x) + r(x)$
  - can interpret $r(x)$ **as being a remainder**
  - $r(x) = f(x) \bmod g(x)$
- if have no remainder say $g(x)$ divides $f(x)$
- if $g(x)$ has no divisors other than itself & 1 say it is **irreducible** (or prime) polynomial
- arithmetic modulo an irreducible polynomial forms a field

Modular Polynomial Arithmetic

- can compute in field GF(2^n)
  - polynomials with coefficients modulo 2
  - whose degree is less than n
  - hence must reduce modulo an irreducible poly of degree n (for multiplication only)
- form a finite field
- can always find an inverse
  - can extend Euclid's Inverse algorithm to find

Example GF(2^3)

Table 4.7 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

(a) Addition

	+	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(b) Multiplication

	×	000 0	001 1	010 x	011 $x+1$	100 x^2	101 x^2+1	110 x^2+x	111 x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

Computational Considerations

- since coefficients are 0 or 1, can represent any such polynomial as a bit string
- **addition** becomes **XOR** of these bit strings
- **multiplication** is **shift & XOR**
  - cf long-hand multiplication
- modulo reduction done by repeatedly substituting highest power with remainder of irreducible poly (also shift & XOR)

Computational Example

- in GF(2^3) have (x^2+1) is 101_2 & (x^2+x+1) is 111_2
- **so addition is**
  - $(x^2+1) + (x^2+x+1) = x$
  - 101 XOR $111 = 010_2$
- and **multiplication is (shift & XOR)**
  - $(x+1).(x^2+1) = x.(x^2+1) + 1.(x^2+1)$
  - $= x^3+x+x^2+1 = x^3+x^2+x+1$
  - $011.101 = (101)$**<<**1 **XOR** $(101)<<0 =$
  - 1010 XOR $101 = 1111_2$
- polynomial modulo reduction (get q(x) & r(x)) is
  - (x^3+x^2+x+1) **mod** $(x^3+x+1) = 1.(x^3+x+1) + (x^2) = x^2$
  - 1111 mod $1011 =$ **1111 XOR $1011 = 0100_2$**

Polynomial GCD

- can find greatest common divisor for polys
  - ***c(x)*** **= GCD(*a(x)*, *b(x)*)**
    if ***c(x)*** is the poly of greatest degree which divides both *a(x), b(x)*
  - can adapt Euclid's Algorithm to find it:
  - EUCLID[*a*(*x*), *b*(*x*)]

1. A(*x*) = *a*(*x*); B(*x*) = *b*(*x*)
2. **2. if** B(*x*) = 0 **return** A(*x*) = gcd[*a*(*x*), *b*(*x*)]
3. R(*x*) = A(*x*) mod B(*x*)
4. A(*x*) ¨ B(*x*)
5. B(*x*) ¨ R(*x*)
6. **goto** 2

Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

```
2  3  5  7  11  13  17  19  23  29  31  37  41  43  47  53  59
61  67  71  73  79  83  89  97  101  103  107  109  113  127
131  137  139  149  151  157  163  167  173  179  181  191
193  197  199
```

Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: n=a x b x c
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
  - eg. $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in \mathrm{P}} p^{a_p}$$

Relatively Prime Numbers & GCD

- **two numbers a , b are relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and **1** is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1 \times 3^1 \times 5^2$ $18=2^1 \times 3^2$ hence $GCD(18,300)=2^1 \times 3^1 \times 5^0=6$

Fermat's Theorem

- $a^{p-1} = 1 \pmod{p}$
  - where p is prime and $\gcd(a,p)=1$
- also known as Fermat's Little Theorem
- also have: $a^{p} = a \pmod{p}$
- useful in **public key and primality testing**

Euler Totient Function ø(n)

- when doing arithmetic modulo n
- **complete set of residues** is: `0..n-1`
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Euler Totient Function ø(n)

- to compute ø(n) need to count number of residues to be excluded
- in general need prime factorization, but
  - for p (p prime) $\phi(p) = p-1$
  - for p.q (p,q prime) $\phi(p.q) = (p-1) \times (q-1)$
- eg.

$\phi(37) = 36$

$\phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\phi(n)} = 1 \pmod n$
  - for any a,n where gcd(a,n)=1
- eg.

  $a=3; n=10;\ \phi(10)=4;$ {1,3,5,7}relatively@10

  hence $3^4 = 81 = 1 \bmod 10$

  $a=2; n=11;\ \phi(11)=10;$

  hence $2^{10} = 1024 = 1 \bmod 11$
- also have: $a^{\phi(n)+1} = a \pmod n$

Primality Testing

- often need to find large prime numbers
- traditionally **sieve** using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternatively can use **statistical primality tests** based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property
- can use a slower deterministic primality test

Miller Rabin Algorithm

- a test based on prime properties that result from Fermat's Theorem
- algorithm is:

TEST (n) is:

1. Find integers k, q, $k > 0$, q odd, so that $(n-1)=2^k q$
2. Select a random integer a, $1<a<n-1$
3. **if** $a^q \bmod n = 1$ **then** return ("inconclusive");
4. **for** $j = 0$ **to** $k - 1$ **do**
   5. **if** ($a^{2^j q} \bmod n = n-1$)
      then return("inconclusive")
6. return ("composite")

Probabilistic Considerations

- if Miller-Rabin returns "composite" the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is $< {}^{1}/_{4}$
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1-4^{-t}$
  - eg. for t=10 this probability is > 0.99999
- could then use the deterministic AKS test

Prime Distribution

- prime number theorem states that primes occur roughly every (`ln n`) integers
- but can immediately ignore evens
- so in practice need only test `0.5 ln(n)` numbers of size `n` to locate a prime
  - note this is only the "average"
  - sometimes primes are close together
  - other times are quite far apart

Chinese Remainder Theorem

- used to speed up modulo computations
- if working modulo a product of numbers
  - eg. $\mathrm{mod}\ M = m_1 m_2 .. m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- can implement CRT in several ways
- to compute $A \pmod{M}$
  - first compute all $a_i = A \bmod m_i$ separately
  - determine constants c_i below, where $M_i = M/m_i$
  - then combine results to get answer using:

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) \pmod{M}$$

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{for } 1 \leq i \leq k$$

CRT Theorem

Theorem (Chinese Remainder Theorem). *Let $m_1, m_2, \ldots, m_r$ be a collection of pairwise relatively prime integers. Then the system of simultaneous congruences*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

has a unique solution modulo $M = m_1 m_2, \cdots m_r$, for any given integers $a_1, a_2, \ldots, a_r$.

CRT Theorem

Proof of CRT. Put $M = m_1 \cdots m_r$ and for each $k = 1, 2, \ldots, r$ let $M_k = \frac{M}{m_k}$. Then $\gcd(M_k, m_k) = 1$ for all k. Let y_k be an inverse of M_k modulo m_k, for each k. Then by definition of inverse we have $M_k y_k \equiv 1 \pmod{m_k}$. Let

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_r M_r y_r.$$

Then x is a simultaneous solution to all of the congruences. Since the moduli $m_1, \ldots, m_r$ are pairwise relatively prime, any two simultaneous solutions to the system must be congruent modulo M. Thus the solution is a unique congruence class modulo M, and the value of x computed above is in that class. □

CRT Example

- Find all integers x which leave a remainder of 1, 2, 3, and 4 when divided by 5, 7, 9, and 11, respectively.

CRT Example

We are asked to solve the system of congruences:

$$x \equiv 1 \pmod 5$$
$$x \equiv 2 \pmod 7$$
$$x \equiv 3 \pmod 9$$
$$x \equiv 4 \pmod{11}.$$

CRT Example

- $M = 5x\ 7\ x\ 9\ x\ 11 = 3465$
- $M_1 = M/5 = 693$,
- $M_2 = M/7 = 495$,
- $M_3 = M/9 = 385$, and
- $M_4 = M/11 = 315$.

CRT Example

- $M_1y_1=1 \pmod{m_1}= 693.y_1= 1 \pmod{5}$
- $y_1 = 2$,
- $y_2 = 3$,
- $y_3 = 4$, and
- $y_4 = 8$.

CRT Example

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_r M_r y_r.$$

x = 1. 693. 2 + 2. 495. 3 + 3. 385. 4 + 4. 315. 8

= 19056

x = [19056]M = 19056 mod M= 19056mod 3465

=[1731]M

CRT Question

- *There are certain things whose number is unknown. Repeatedly divided by 3, the remainder is 2; by 5 the remainder is 3; and by 7 the remainder is 2. What will be the number?*

- x=2 (mod 3)
- x=3 (mod 5)
- x=2 (mod 7)

Answer

- Thus we have solutions

23,128,233,…….

More Questions

1. Find all integers that leave a remainder of 1 when divided by 2, 3, and 5.

2. Find a solution to
   - x ≡ 88 (mod 6)
   - x ≡ 100 (mod 15)

Primitive Roots

- from Euler's theorem have $a^{\phi(n)} \bmod n = 1$
- consider $a^m = 1 \pmod n$, $\gcd(a,n) = 1$
  - must exist for $m = \phi(n)$ but may be smaller
  - once powers reach m, cycle will repeat
- if smallest is $m = \phi(n)$ then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group $\bmod p$
- these are useful but relatively hard to find

Powers mod 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Discrete Logarithms

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- that is to find i such that $b = a^i \pmod p$
- this is written as $i = \text{dlog}_a\ b \pmod p$
- if a is a primitive root then it always exists, otherwise it may not, eg.

  $x = \log_3 4 \bmod 13$ has no answer

  $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Discrete Logarithms mod 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Summary

- have considered:
  - prime numbers
  - Fermat's and Euler's Theorems & ø(n)
  - Primality Testing
  - Chinese Remainder Theorem
  - Primitive Roots & Discrete Logarithms

Summary

- have considered:
  - concept of groups, rings, fields
  - modular arithmetic with integers
  - Euclid's algorithm for GCD
  - finite fields GF(p)
  - polynomial arithmetic in general and in GF(2^n)