Cryptography and Network Security
Chapter 4

Fifth Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 4 – Basic Concepts in Number Theory and Finite Fields

The next morning at daybreak, Star flew indoors, seemingly keen for a lesson. I said, "Tap eight." She did a brilliant exhibition, first tapping it in 4, 4, then giving me a hasty glance and doing it in 2, 2, 2, 2, before coming for her nut. It is astonishing that Star learned to count up to 8 with no difficulty, and of her own accord discovered that each number could be given with various different divisions, this leaving no doubt that she was consciously thinking each number. In fact, she did mental arithmetic, although unable, like humans, to name the numbers. But she learned to recognize their spoken names almost immediately and was able to remember the sounds of the names. Star is unique as a wild bird, who of her own free will pursued the science of numbers with keen interest and astonishing intelligence.

— ***Living with Birds***, Len Howard

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 basic number theory concepts

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|a`
- and say that `b` is a **divisor** of `a`
- eg. all of 1,2,3,4,6,8,12,24 divide 24
- eg. 13 | 182; –5 | 30; 17 | 289; –3 | 33; 17 | 0

Properties of Divisibility

- If *a|1, then a = ±1.*
- *If a|b and b|a, then a = ±b.*
- *Any b /= 0 divides 0.*
- *If a | b and b | c, then a | c*
  - *e.g.* 11 | 66 and 66 | 198 x 11 | 198
- If *b|g and b|h, then b|(mg + nh)*

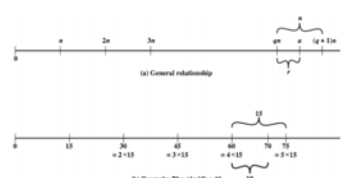
  for arbitrary integers m and n

  e.g. b = 7; g = 14; h = 63; m = 3; n = 2

  hence 7|14 and 7|63

Division Algorithm

- if divide a by n get integer quotient *q* and integer remainder *r* such that:
  - *a = qn + r* where *0 <= r < n; q = floor(a/n)*
- remainder *r* often referred to as a **residue**



Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is the largest integer that divides evenly into both a and b
  - eg GCD(60,24) = 12
- define gcd(0, 0) = 0
- often want **no common factors** (except 1) define such numbers as **relatively prime**
  - eg GCD(8,15) = 1
  - hence 8 & 15 are relatively prime

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)
```

GCD(1160718174, 316258250)

Dividend	Divisor	Quotient	Remainder
a = 1160718174	b = 316258250	q1 = 3	r1 = 211943424
b = 316258250	r1 = 211943424	q2 = 1	r2 = 104314826
r1 = 211943424	r2 = 104314826	q3 = 2	r3 = 3313772
r2 = 104314826	r3 = 3313772	q4 = 31	r4 = 1587894
r3 = 3313772	r4 = 1587894	q5 = 2	r5 = 137984
r4 = 1587894	r5 = 137984	q6 = 11	r6 = 70070
r5 = 137984	r6 = 70070	q7 = 1	r7 = 67914
r6 = 70070	r7 = 67914	q8 = 1	r8 = 2516
r7 = 67914	r8 = 2516	q9 = 31	r9 = 1078
r8 = 2516	r9 = 1078	q10 = 2	r10 = 0

Modular Arithmetic

- define **modulo operator** "`a mod n`" to be remainder when a is divided by n
  - where integer *n* is called the **modulus**
- *b* is called a **residue** of *a* mod *n*
  - since with integers can always write: `a = qn + b`
  - usually chose smallest positive remainder as residue
    - ie. `0 <= b <= n-1`
  - process is known as **modulo reduction**
    - `eg. -12 mod 7 = -5 mod 7 = 2 mod 7 = 9 mod 7`
- *a* & *b* are **congruent** if: `a mod n = b mod n`
  - when divided by *n,* a & b have same remainder
  - eg. 100 = 34 mod 11

Modular Arithmetic Operations

- can perform arithmetic with residues
- uses a finite number of values, and loops back from either end

  Z_n = {0, 1, . . . , (*n* – *1)}*
- 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 Operations

1. `[(a mod n) + (b mod n)] mod n = (a + b) mod n`
2. `[(a mod n) – (b mod n)] mod n = (a – b) mod n`
3. `[(a mod n) x (b mod n)] mod n = (a x b) mod n`

e.g.

[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 = 2 (11 + 15) mod 8 = 26 mod 8 = 2

[(11 mod 8) – (15 mod 8)] mod 8 = –4 mod 8 = 4 (11 – 15) mod 8 = –4 mod 8 = 4

[(11 mod 8) x (15 mod 8)] mod 8 = 21 mod 8 = 5 (11 x 15) mod 8 = 165 mod 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

Modular Arithmetic Properties

Property	Expression
Commutative laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive inverse (−w)	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

Euclidean Algorithm

- an efficient way to find the GCD(a,b)
- uses theorem that:
  - GCD(a,b) = GCD(b, a mod b)
- Euclidean Algorithm to compute GCD(a,b) is:

```
Euclid(a,b)
  if (b=0) then return a;
  else return Euclid(b, a mod b);
```

Extended Euclidean Algorithm

- calculates not only GCD but x & y:
  ax + by = d = gcd(a, b)
- useful for later crypto computations
- follow sequence of divisions for GCD but assume at each step i, can find x &y:
  r = ax + by
- at end find GCD value and also x & y
- if GCD(a,b)=1 these values are inverses

Finding Inverses

```
EXTENDED EUCLID(m, b)
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)

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

Group

- a set of elements or "numbers"
  - may be finite or infinite
- 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

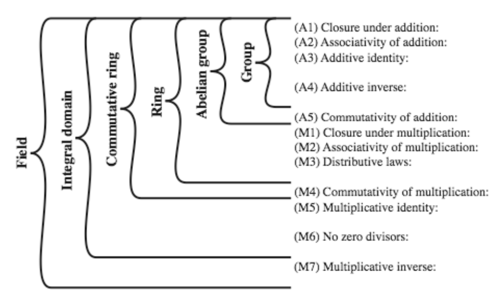
Ring

- a set of "numbers"
- with two operations (addition and multiplication) which form:
- an abelian group with addition operation
- and 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 an identity and no zero divisors, it forms an **integral domain**

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



Finite (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
  - find inverse with Extended Euclidean algorithm
- hence arithmetic is "well-behaved" and can do addition, subtraction, multiplication, and division without leaving the field GF(p)

GF(7) Multiplication Example

×	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

Polynomial Arithmetic

- can compute using polynomials

  $f(x) = a_n x^n + a_{n-1}x^{n-1} + … + a_1 x + a_0 = \sum a_i x^i$
  - 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 $g(x) = x^2 – x + 1$

  $f(x) + g(x) = x^3 + 2x^2 – x + 3$

  $f(x) – g(x) = x^3 + x + 1$

  $f(x) \text{ x } g(x) = x^5 + 3x^2 – 2x + 2$

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

    $f(x) + g(x) = x^3 + x + 1$

    $f(x) \text{ x } g(x) = x^5 + x^2$

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

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))
  if (b(x)=0) then return a(x);
  else return
          Euclid(b(x), a(x) mod b(x));
```

- all foundation for polynomial fields as see next

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	001	010	011	100	101	110	111
	+	0	1	x	$x+1$	x^2	x^2+1	x^2+x	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	001	010	011	100	101	110	111
	×	0	1	x	$x+1$	x^2	x^2+1	x^2+x	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
  - $(x+1).(x^2+1) = x.(x^2+1) + 1.(x^2+1)$
    $= x^3+x+x^2+1 = x^3+x^2+x+1$
  - 011.101 = (101)<<1 XOR (101)<<0 =
    1010 XOR 101 = 1111_2
- polynomial modulo reduction (get q(x) & r(x)) is
  - $(x^3+x^2+x+1) \bmod (x^3+x+1) = 1.(x^3+x+1) + (x^2) = x^2$
  - 1111 mod 1011 = 1111 XOR 1011 = 0100_2

Using a Generator

- equivalent definition of a finite field
- a **generator** g is an element whose powers generate all non-zero elements
  - in F have 0, g^0, g^1, …, g^{q-2}
- can create generator from **root** of the irreducible polynomial
- then implement multiplication by adding exponents of generator

Summary

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