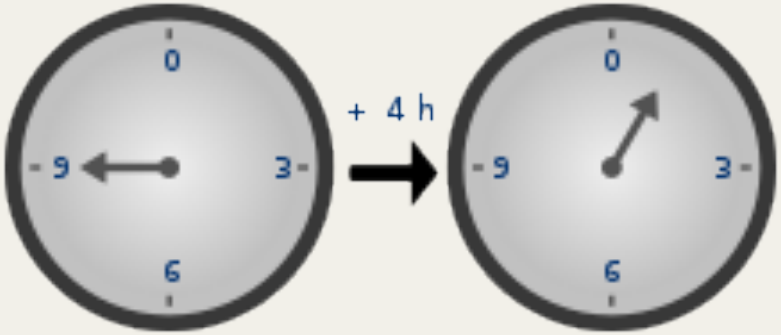
COM 5335 NETWORK SECURITY
LECTURE 3
FINITE FIELDS I

Scott CH Huang

Modular Arithmetic

- It's sometimes called the 'clock arithmetic'.
- It uses a finite number of values and loops back from either end:
  - *a (mod n) ≡ a+n (mod n) ≡ a+2*n (mod n)*



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
...

Modular Arithmetic

- Define modulo operator a mod n as the remainder when a is divided by n.
- We use the term congruence for ‘a ≡ b mod n’.
  - *It reads “a is congruent to b modulo n”.*
  - *When divided by n, a & b have same remainder*
  - *e.g. 100 ≡ 34 mod 11 ≡ 1 mod 11*
  - *-12 mod 7 ≡ -5 mod 7 ≡ 2 mod 7 ≡ 9 mod 7*
- b is also called the residue of a mod n

Modular Arithmetic Operations

- Include additions & multiplications
- Apply modulo to reduce answer within n.
- Basic properties
  - $a+b \bmod n \equiv (a \bmod n)+(b \bmod n) \bmod n$
  - $a*b \bmod n \equiv (a \bmod n)*(b \bmod n) \bmod n$

Modulo 5 Addition/ Multiplication Table

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Divisors

- A non-zero number b divides a if, for some m, we have a=m*b (a,b,m all integers)
  - *If we divide a by b, there's no remainder.*
- Denoted by b|a
- b is called a divisor of a
  - *e.g. each of 1,2,3,4,6,8,12,24 divides 24.*

Modular Arithmetic

- Modular arithmetic for integer n:
  - $Z_n = \{0, 1, \ldots, n-1\}$
  - $(Z_n, +, *)$ *forms a commutative ring (to be explained later)*
- Some Remarks
  - *If* $(a+b) \equiv (a+c) \bmod n$ *then* $b \equiv c \bmod n$.
  - *If* $(a*b) \equiv (a*c) \bmod n$ *then* $b \equiv c \bmod n$ *only if a is relatively prime to n.*

Greatest Common Divisor (GCD)

- A.k.a. the highest common factor (HCF).
- An elementary concept in number theory.
- GCD (a,b) of a and b is the largest number that divides both a and b .
  - *e.g. GCD(60,24) = 12*
- Numbers are relatively prime if their GCD = 1.
  - *e.g. GCD(8,15) = 1; 8 & 15 are relatively prime.*

Euclid's Algorithm (輾轉相除法)

- An efficient way to find the GCD(a,b)
- Based on the lemma that
  - *GCD(a,b) = GCD(b, a mod b)*
- Apply 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*

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)

Introduction to Finite Field

- Important in cryptography
  - *AES, Elliptic Curve, IDEA, XTR*
- Operations on “abstract elements”
  - *What constitutes a “number” and the type of operations varies considerably*
- Groups, rings, fields from abstract algebra

Groups

Niels Henrik Abel (1802-1829)

- (G,*): a set G of elements with operation '*' satisfying
  - *(G0) Closure:* $\forall a,b \in G \Rightarrow a*b \in G$
  - *(G1) Associativity:* $(a*b)*c = a*(b*c)$
  - *(G2) Identity:* $\exists e \text{ s.t. } e*a = a*e = a \quad \forall a \in G$
  - *(G3) Inverse:* $\exists\, a^{-1} \text{ s.t. } a*a^{-1} = a^{-1}*a = e$

- If commutativity also holds
  - *(G4). a*b = b*a then it is called an* ***abelian group***

Example 1

- G = {0,1,2,3}
- Operation: + (mod 4)
- (G,+) is an abelian group.

Example 2

- G = {0,1,2,3}
- Operator: * (mod 4)
- Is (G,*) a group? If not, which condition fails?

Example 3

- G = {1,2,3,4}
- Operator: * (mod 5)
- Is (G,*) a group? If not, which condition fails?

Example 4

- G = {1,2,3}
- Operator: * (mod 5)
- Is (G,*) a group? If not, which condition fails?

Example 5

- G = {1,2,3}
- Operator: * (mod 4)
- Is (G,*) a group? If not, which condition fails?

Rings

- (R,+,*) a set R of elements with two operations ‘+’ and ‘*’ satisfying the following conditions
  - *(R,+) is an abelian group. (G0-G4)*
  - *(R,*) is a semi-group, i.e. (G0,G1)*
    - Closure: a,b∈R => a*b ∈R
    - Associativity: (a*b)*c = a*(b*c)
  - *Distributivity: a*(b+c) = a*b + a*c*
- If ‘*’ is also commutative, it’s called a commutative ring.
- If the multiplicative identity exists, it’s called a ring with (unit) 1.
- Exercise: Is {0,1,2,3; (+, *) (mod 4)} a ring?

Example of Ring: Z_6

- Z_6={0,1,2,3,4,5}
- +: mod 6 addition
- *: mod 6 multiplication
- Additive identity = 0
- Multiplicative identity = 1

Z_6

- Additive inverse of 5?
  - *5+1=0, -5=1*
- Multiplicative inverse of 5?
  - *5*5=1, 5-1=5*
- Multiplicative inverse of 3?
  - *3 has no multiplicative inverse.*
- Elements of a ring may not have multiplicative inverse.

Fields

- A ring (R,+,*) satisfying:
  - *(R,+) is an abelian group*
  - *(R\{0},*) is an abelian group*
- In short, a field is a ***commutative division ring***.

- Exercise: Test if {0,1,2,3; (+, *) (mod 4)} is a field.
- Exercise: Test if {0,1,2,3, 4; (+, *) (mod 5)} is a field.

Finite Fields (Galois Fields)

Évariste Galois (1811-1832)

- Finite fields play a key role in cryptography
- The number of elements in a finite field must be some power of a prime p^n (big theorem!)
- Known as Galois fields
- Denoted by $GF(p^n)$
- Most important finite fields:
  - *GF(p)*
  - *$GF(2^n)$*

Finite Fields GF(p)

- GF(p) is the set of integers Z_p={0,1, … , p-1} with arithmetic operations modulo a prime p
- (Z_p,+,*) forms the finite field GF(p).
  - *Since each item has a multiplicative inverse*
- Division is “well-behaved”
  - *We can perform addition, subtraction, multiplication, and division in GF(p).*
- If p is prime, then Z_p is a field. Z_p=GF(p) .
- If n is not prime, then Z_n is not a field. Z_n is a commutative ring with 1.

GF(7) Multiplication Table

$\times$	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

Multiplicative Inverse of GF(7)

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

Finding Multiplicative Inverses in Z_p

- Finding the mult. inverse of 337 in Z1021
  - *Run Euclid's algorithm*
  - *1021-3*337=10*
  - *337-33*10=7*
  - *10-1*7=3*
  - *7-2*3=1*

Finding Multiplicative Inverses in Z_p (Backward Ver.)

- Run extended Euclid's algorithm (**backward substitution**)
  - *1=1*7+(-2)*3=1*7+(-2)(10-1*7)*
  - *=(-2)10+3(7)*
  - *=(-2)10+3(337-33*10)*
  - *=(3)337+(-101)10*
  - *=3(337)+(-101)(1021-3*337)*
  - *=(-101)1021+(306)337*
  - *337-1=306 mod 1021, multiplicative inverse.*

Euclid's Algorithm in C

- //Precondition: a,b > 0
- int gcd(int a, int b) {
  - *while (b != 0){*
  - *t = b;*
  - *b = a % b;*
  - *a = t;*
  - *return a ;*
  - *}*
- }

Extended Euclidean Algorithm

- There are both forward and backward substitution algorithms
- It can be used to find **Bézout's** identity
- Bézout's identity
  - *Let a and b be integers with gcd=d. Then, there exist integers x and y such that ax + by = d.*
  - *x,y are called Bézout's coefficients*
- Bézout's identity leads to multiplicative inverse, Chinese Remainder Theorem, Rabin cryptosystem and many other important cryptographic primitives.
- (Myth): *where there is cryptography, there is extended Euclidean algorithm.*

Standard Euclidean Algorithm Revisited

- Input: positive integers a,b
- Output: (quotients) $q_1,\ldots,q_k$ and (remainders) $r_0,\ldots,r_{k+1}$ s.t.

$$
\begin{aligned}
r_0 &= a \\
r_1 &= b \\
&\vdots \\
r_{i+1} &= r_{i-1} - q_i r_i \quad \text{and} \quad 0 \leq r_{i+1} < |r_i| \\
&\vdots
\end{aligned}
$$

Extended Euclidean Algorithm (Forward Ver.)

- I/O: same as standard Euclidean algorithm except there are two more output sequences: s_i and t_i

$$
\begin{array}{rclcrcl}
r_0 &=& a & & r_1 &=& b \\
s_0 &=& 1 & & s_1 &=& 0 \\
t_0 &=& 0 & & t_1 &=& 1 \\
&\vdots& & & &\vdots& \\
r_{i+1} &=& r_{i-1} - q_i r_i & \text{and} & 0 \le r_{i+1} &<& |r_i| \\
s_{i+1} &=& s_{i-1} - q_i s_i & & & & \\
t_{i+1} &=& t_{i-1} - q_i t_i & & & & \\
&\vdots& & & & &
\end{array}
$$

- Computation stops when $r_{k+1}=0$. $\gcd(a,b)=r_k$. Bézout's coefficients = s_k, t_k.

A Concrete Example

- Input a=240 b=46. Results: $-9 \times 240 + 47 \times 46 = 2$

i	quotient q_{i-1}	remainder r_i	s_i	t_i
0		240	1	0
1		46	0	1
2	240 ÷ 46 = 5	240 − 5 × 46 = 10	1 − 5 × 0 = 1	0 − 5 × 1 = −5
3	46 ÷ 10 = 4	46 − 4 × 10 = 6	0 − 4 × 1 = −4	1 − 4 × −5 = 21
4	10 ÷ 6 = 1	10 − 1 × 6 = 4	1 − 1 × −4 = 5	−5 − 1 × 21 = −26
5	6 ÷ 4 = 1	6 − 1 × 4 = 2	−4 − 1 × 5 = −9	21 − 1 × −26 = 47
6	4 ÷ 2 = 2	4 − 2 × 2 = 0	5 − 2 × −9 = 23	−26 − 2 × 47 = −120

Some Remarks

- If n is not prime, then Z_n is not a field.
- Given $x \in Z_n$, x^{-1} may not exist.
- Under what condition will x^{-1} exist?

Polynomial Arithmetic

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

- Consider
- Both + and * can be performed on polynomials if they can be performed on a0,…,an.
- Suppose a0,…,an∈R, R is a ring. Denote the set of polynomials by R[x].
- R[x] forms a ring, usually called the polynomial ring.
- Is {f∈ R[x] | deg(f)≤n } a ring?
- If F is a field. Is F[x] a field?

Ordinary Polynomial Arithmetic

- Z[x] arithmetics
  - *Let* $f(x) = x^3 + x^2 + 2$ *and* $g(x) = x^2 - x + 1$
  - $f(x) + g(x) = x^3 + 2x^2 - x + 3$
  - $f(x) - g(x) = x^3 + x + 1$
  - $f(x) * g(x) = x^5 + 3x^2 - 2x + 2$

Polynomial Arithmetic with Modulo Coefficients

- $Z_n[x]$ arithmetic
- In $Z_2[x]$
  - *Let* $f(x) = x^3 + x^2$ *and* $g(x) = x^2 + x + 1$
  - $f(x) + g(x) = x^3 + x + 1$
  - $f(x) * g(x) = x^5 + x^2$

Modular Polynomial Arithmetic

- Can we generalize $a \equiv b \pmod{n}$ to $a(x) \equiv b(x) \pmod{n(x)}$?
- We can consider modular +, * on polynomials too.
  - *If $f(x) = q(x) * g(x) + r(x)$*
  - *Interpret $r(x)$ as a remainder*
  - *$r(x) \equiv f(x) \bmod g(x)$*
- If have r(x)=0, we say g(x) divides f(x).
- The set of all polynomials R[x] modulo a fixed polynomial g(x) also forms a ring.
- We call this ring the quotient ring, denoted by R[x]/(g(x)) or R[x] mod g(x).

Quotient Rings

- Z_p is actually a quotient ring too.
- $Z_p = Z/(p)$ or Z mod p.
  - *c.f. R[x]/(g(x))*
- If $p \in Z$ is prime, then Z/(p) is a field.
- If $p(x) \in R[x]$ is prime (what does this mean???), then R[x]/(p(x)) is a field???

Irreducible Polynomials

- $g(x)$ is irreducible iff it has no divisors other than itself & 1.
- If $p(x) \in R[x]$ is irreducible, then $R[x]/(p(x))$ is a field.
- We can find the multiplicative inverse of any polynomial by running the extended Euclid's algorithm just like what we did earlier with integers.

Euclid’s Algorithm on Polynomials

- An efficient way to find the GCD(f(x),g(x))
- Based on the lemma that:
  - *GCD(f(x),g(x)) = GCD(g(x), f(x) mod g(x))*
- Euclid's Algorithm to compute GCD(f,g):
  - *A=f, B=g*
  - *while B>0*
    - R = A mod B
    - A = B, B = R
  - *return A*

Finite Field Construction

- To construct GF(p^n)
  - *Find an irreducible polynomial $p(x) \in Z_p[x]$*
  - *GF(p^n) can be constructed as $Z_p[x]/(p(x))$*
- This is just one of many equivalent constructions.
- Multiplicative inverses always exist. Why?

Example of $GF(2^3)$

- Find an irreducible polynomial $p(x)=x^3+x+1 \in Z_2[x]$
- $GF(8)=Z_2[x]/(p(x))$
- From now on, we use 1011 to represent $1x^3+0x^2+1x^1+1x^0$
- Everything is calculated mod 1011 (not modulo a number!!)
- Never regard these bit-strings as binary numbers and perform these operations on numbers !!!
- Example: 10001 = 111 mod 1011 because 10001=10*1011+111
- We only need 3 bits to represent each element. Why?

GF(2^3) Addition/Multiplication Table

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

	+	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

(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	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	$x+1$

(b) Multiplication

Computational Considerations

- Since coefficients are 0 or 1, we can always represent any polynomial as a bit string.
- Addition becomes XOR of these bit strings
- Multiplication can be done more easily
  - *Shift & XOR (to be explained in lec 4)*
- Modulo reduction can be done by repeatedly substituting highest power with remainder of an irreducible polynomial (also shift & XOR)