

Lecture 5: Finite Fields (PART 2)

PART 2: Modular Arithmetic

Theoretical Underpinnings of AES and ECC

Lecture Notes on “Computer and Network Security”

by Avi Kak (kak@purdue.edu)

April 26, 2009

©2009 Avinash Kak, Purdue University

Goal of this part:

- To review modular arithmetic
- To present Euclid’s gcd algorithms
- To present the prime finite field Z_p
- To show how Euclid’s gcd algorithm can be extended to find multiplicative inverses

Goal for the upcoming part:

- To review polynomial arithmetic.

5.1: Modular Arithmetic Notation

- Given **any** integer a and a **positive** integer n , and given a division of a by n that leaves the remainder between 0 and $n - 1$, both inclusive, we define

$$a \text{ mod } n$$

to be **the remainder**. Note that the remainder **must** be between 0 and $n - 1$, both ends inclusive, even if that means that we must use a negative quotient when dividing a by n .

- We will call two integers a and b to be **congruent modulo n** if

$$(a \text{ mod } n) = (b \text{ mod } n)$$

- Symbolically, we will express such a **congruence** by

$$a \equiv b \pmod{n}$$

- We say a non-zero integer a is a **divisor** of another integer b provided there is no remainder when we divide b by a . That is, when $b = ma$ for some integer m .

- When a is a divisor of b , we express this fact by $a \mid b$.

5.2: Examples of Congruences

- Here are some congruences modulo 3:

$$\begin{array}{rcl} 7 & \equiv & 1 \pmod{3} \\ -8 & \equiv & 1 \pmod{3} \\ -2 & \equiv & 1 \pmod{3} \\ 7 & \equiv & -8 \pmod{3} \\ -2 & \equiv & 7 \pmod{3} \end{array}$$

- One way of seeing the above congruences (for mod 3 arithmetic):

... 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 ...
... -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 ...

where the top line is the output of **modulo 3** arithmetic and the bottom line the set of **all** integers.

- Obviously, then, **modulo n** arithmetic maps all integers into the set $\{0, 1, 2, 3, \dots, n - 1\}$.

5.3: Modular Arithmetic Operations

- The following equalities are easily shown to be true:

$$\begin{aligned} [(a \bmod n) + (b \bmod n)] \bmod n &= (a + b) \bmod n \\ [(a \bmod n) - (b \bmod n)] \bmod n &= (a - b) \bmod n \\ [(a \bmod n) \times (b \bmod n)] \bmod n &= (a \times b) \bmod n \end{aligned}$$

with the ordinary meaning to be ascribed to the arithmetic operators.

- To prove any of the above equalities, you write a as $mn + r_a$ and b as $pn + r_b$, where r_a and r_b are the **residues** (the same thing as **remainders**) for a and b , respectively. You substitute for a and b on the right hand side and show you can now derive the left hand side. Note that r_a is $a \bmod n$ and r_b is $b \bmod n$. You also need to make use of equalities such as

$$(p + q) \bmod n = (p \bmod n) + (q \bmod n)$$

- For **arithmetic modulo n** , let Z_n denote the set

$$Z_n = \{0, 1, 2, 3, \dots, n - 1\}$$

Z_n is obviously the set of **remainders** in arithmetic modulo n .
It is officially called the **set of residues**.

5.4: Properties of the set Z_n

Commutativity:

$$\begin{aligned}(w + x) \bmod n &= (x + w) \bmod n \\ (w \times x) \bmod n &= (x \times w) \bmod n\end{aligned}$$

Associativity:

$$\begin{aligned}[(w + x) + y] \bmod n &= [w + (x + y)] \bmod n \\ [(w \times x) \times y] \bmod n &= [w \times (x \times y)] \bmod n\end{aligned}$$

Distributivity of Multiplication over Addition:

$$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$$

Existence of Identity Elements:

$$\begin{aligned}(0 + w) \bmod n &= (w + 0) \bmod n \\ (1 \times w) \bmod n &= (w \times 1) \bmod n\end{aligned}$$

Existence of Additive Inverses:

for each $w \in Z_n$ there exists a z such that $w + z = 0 \bmod n$

5.5: So What is Z_n ?

- Z_n is a commutative ring. Why? [See the lecture notes for the previous lecture for why.]
- Actually, Z_n is more than a commutative ring, but not quite an integral domain. What do I mean by that? [Because Z_n contains a multiplicative identity element. Commutative rings are not required to possess multiplicative identities.]
- Why is Z_n not an integral domain? [Even though Z_n possesses a multiplicative identity, it does NOT satisfy the other condition of integral domains which says that if $a \times b = 0$ then either a or b must be zero. Consider modulo 8 arithmetic. We have $2 \times 4 = 0$, which is a clear violation of the second rule for integral domains.]
- Why is Z_n not a field?
- Is Z_n a group? If so, what is the group operator? [The group operator is the modulo n addition.]
- Is Z_n an abelian group?
- Is Z_n a ring?

5.6: Asymmetries between Modulo Addition and Modulo Multiplication over Z_n

- For every element of Z_n , there exists an **additive inverse** in Z_n . But there does not exist a **multiplicative inverse** for every element of Z_n .
- Shown below are the additive and the multiplicative inverses for **modulo 8** arithmetic:

Z_8	=	0	1	2	3	4	5	6	7
addit. inv.	=	0	7	6	5	4	3	2	1
multi. inv	=	-	1	-	3	-	5	-	7

- Note that the **multiplicative inverses** exist for only those elements of Z_n that are **relatively prime** to n . Two integers are relatively prime to each other if the integer 1 is their only one common positive divisor. More formally, two integers a and b are relative prime to each other if $\gcd(a, b) = 1$ where \gcd denotes the **Greatest Common Divisor**.

- The following property of **modulo n addition** is the same as for ordinary addition:

$$(a + b) \equiv (a + c) \pmod{n} \quad \text{implies} \quad b \equiv c \pmod{n}$$

But a similar property is **NOT** obeyed by **modulo n multiplication**. That is

$$(a \times b) \equiv (a \times c) \pmod{n} \quad \text{does not imply} \quad b \equiv c \pmod{n}$$

unless a and n are **relatively prime** to each other.

- That the **modulo n addition** property stated above should hold true for all elements of Z_n follows from the fact that the **additive inverse** $-a$ exists for every $a \in Z_n$. So we can add $-a$ to both sides of the equation to prove the result.
- To prove the same result for **modulo n multiplication**, we will need to multiply both sides of the second equation above by the multiplicative inverse a^{-1} . But, as you already know, not all elements of Z_n possess multiplicative inverses.
- Since the existence of the multiplicative inverse for an element a of Z_n is predicated on a being **relatively prime** to n and since the answer to the question whether two integers are relatively prime

to each other depends on their **greatest common divisor** (gcd), let's explore next the world's most famous algorithm for finding the gcd of two integers.

- This algorithm is by Euclid who is considered to be the father of geometry. He was born around 325 BC.

5.7: Euclid's Method for Finding the Greatest Common Divisor of Two Integers

Euclid's GCD is based on the following observations:

- $\gcd(a, a) = a$
- *if $b|a$ then $\gcd(a, b) = b$*
- $\gcd(a, 0) = a$ *since it is always true that $a|0$*
- Assuming without loss of generality that a is larger than b , it can be shown that

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

This is the heart of Euclid's algorithm (now over 2000 years old).

5.8: Steps in a Recursive Invocation of Euclid's Algorithm

```
gcd( b1, b2 ):
= gcd( b2, b1 mod b2 )      = gcd( b2, b3 )
= gcd( b3, b2 mod b3 )      = gcd( b3, b4 )
= gcd( b4, b3 mod b4 )
.....
.....
.....

until b_{m-1} mod b_m == 0

then gcd(b1, b2) = b_m
```

Note that the algorithm works for any two non-negative integers b_1 and b_2 regardless of which is the larger integer. If the first integer is smaller compared to the second integer, the first iteration will swap the two.

5.9: Example of Euclid's Algorithm in Action

$$\begin{aligned}\gcd(70, 38) \\ &= \gcd(38, 32) \\ &= \gcd(32, 6) \\ &= \gcd(6, 2) \\ &= \gcd(2, 0)\end{aligned}$$

$$\text{Therefore, } \gcd(70, 38) = 2$$

Another Example (for relatively prime pair of integers):

$$\begin{aligned}\gcd(8, 17) : \\ &= \gcd(17, 8) \\ &= \gcd(8, 1) \\ &= \gcd(1, 0)\end{aligned}$$

$$\text{Therefore, } \gcd(8, 17) = 1$$

When the smaller of the two numbers is 1 (which happens when the two starting numbers are relatively prime), there is no need to go to the last step in which the smaller of the two numbers is 0.

5.10: An Example of Euclid's Algorithm for Moderately Large Numbers

$$\begin{aligned} \gcd(40902, 24140) & \\ &= \gcd(24140, 16762) \\ &= \gcd(16762, 7378) \\ &= \gcd(7378, 2006) \\ &= \gcd(2006, 1360) \\ &= \gcd(1360, 646) \\ &= \gcd(646, 68) \\ &= \gcd(68, 34) \\ &= \gcd(34, 0) \end{aligned}$$

Therefore, $\gcd(40902, 24140) = 34$

5.11: Proof of Euclid's Algorithm

The proof of Euclid's algorithm is based on the observations

- that we can obviously write $a = qb + r$ since we can do so for any two integers
- it must be case that **every common divisor** of a and b must also divide the remainder r .
- that implies, the all **common** divisors for a and b are the same as those for b and r .
- since $\gcd(a, b)$ is one of those common divisors, then it must be the case that $\gcd(a, b) = \gcd(b, r)$.

5.12: Implementation of Euclid's Algorithm in Python

```
def GCD(a,b):  
    if a < b: a,b = b,a  
    while b != 0:  
        remainder = a % b  
        a, b = b, remainder  
    return a
```

5.13: Prime Finite Fields

- Earlier we showed that, in general, Z_n is a commutative ring.
- The main reason for why, in general, Z_n is only a commutative ring and **not** a finite field is because not every element in Z_n is guaranteed to have a multiplicative inverse.
- In particular, as shown before, an element a of Z_n does **not** have a multiplicative inverse if a is not **relatively prime** to the modulus n .
- What if we choose the modulus n to be a **prime number**? (A prime number has only two divisors, one and itself.)
- For prime n , every element $a \in Z_n$ will be **relatively prime** to n . That implies that there will exist a **multiplicative inverse** for every $a \in Z_n$ for prime n .

- Therefore, Z_p is a **finite field** if we assume p denotes a **prime number**. Z_p is sometimes referred to as a **prime finite field**. Such a field is also denoted $GF(p)$, where GF stands for “Galois Field”.

5.14: What Happened to the Main Reason for Why Z_n Could Not be an Integral Domain?

- Earlier, when we were looking at how to characterize Z_n , we said that, although it possessed a **multiplicative identity**, it could not be an **integral domain** because Z_n allowed for the equality $a \times b = 0$ even for non-zero a and b . (Recall, 0 means the **additive identity element**.)
- If we have now decided that Z_p is a finite field for prime p because every element in Z_p has a unique multiplicative inverse, are we sure that we can now also guarantee that if $a \times b = 0$ then either a or b must be 0?
- Yes, we have that guarantee because $a \times b = 0$ for general Z_n occurs only when non-zero a and b are factors of the modulus n . When n is a prime, its only factors are 1 and n . So with the elements of Z_n being in the range 0 through $n - 1$, the only time we will see $a \times b = 0$ is when either a is 0 or b is 0.

5.15: Finding Multiplicative Inverses for Elements of Z_p

- In general, to find the multiplicative inverse of $a \in Z_n$, we need to find the element b such that

$$a \times b = 1 \text{ mod } n$$

- Based on the discussion so far, we can say that the multiplicative inverses exist for all $a \in Z_n$ for which we have

$$\gcd(a, n) = 1$$

Obviously, when n equals a prime p , this condition will always be satisfied by all elements of Z_p .

- In general, it can be shown that when a and n are **any** pair of positive integers, the following must always hold for some integers x and y (that may be positive or negative or zero):

$$\gcd(a, n) = x \times a + y \times n \quad (1)$$

This is known as the **Bezout's Identity**. For example, when $a = 16$ and $n = 6$, we have $\gcd(16, 6) = 2$. We can certainly write: $2 = (-1) \times 16 + 3 \times 6 = 2 \times 16 + (-5) \times 6$.

This shows that x and y do not have to be unique in Bezout's identity for given a and n .

5.16: Proof of Bezout's Identity

We will now prove that for a given pair of positive integers a and b , we have

$$\gcd(a, b) = ax + by \quad (2)$$

for some positive or negative integers x and y .

- First define a set S as follows

$$S = \{am + bn \mid am + bn > 0, m, n \in N\} \quad (3)$$

where N is the set of all integers. That is,

$$N = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\} \quad (4)$$

- Note that, by its definition, S can only contain *positive* integers. When $a = 8$ and $b = 6$, we have

$$S = \{2, 4, 6, 8, \dots\} \quad (5)$$

It is interesting to note that several pairs of (m, n) will usually result in the same element of S . For example, with $a = 8$ and $b = 6$, the element 2 of S is given rise to by the following pairs of $(m, n) = (1, -1), (-2, 3), (4, -5), \dots$

- Now let d denote the *smallest* element of S .

- Let's now express a in the following form

$$a = qd + r, \quad 0 \leq r < d \quad (6)$$

Obviously then,

$$\begin{aligned} r &= a \text{ mod } d \\ &= a - qd \\ &= a - q(am + bn) \\ &= a(1 - qm) + b(-n) \end{aligned}$$

We have just expressed the residue r as a linear sum of a and b . But that is only possible if r equals 0. If r is not 0 but actually a non-zero integer *less than* d that it must be, that would violate the fact that d is the smallest positive linear sum of a and b .

- Since r is zero, it must be the case that $a = qd$ for some integer q . Similarly, we can prove that b is sd for some integer s . This proves that d is a common divisor of a and b .
- But how do we know that d is the GCD of a and b ?
- Let's assume that some other integer c is also a divisor of a and b . Then it must be the case that c is a divisor of all linear combinations of the form $ma + nb$. Since d is of the form $ma + nb$, then c must be a divisor of d . This fact applies to any arbitrary common divisor c of a and b . That is, *every* common divisor c of a and b must also be a divisor of d .
- Hence it must be the case that d is the GCD of a and b .

5.17: Finding Multiplicative Inverses using Bezout's Identity

- Given an a that is relatively prime to n , we must obviously have $\gcd(a, n) = 1$. Such an a and n must satisfy the following constraint for some x and y :

$$x \times a + y \times n = 1 \tag{7}$$

Let's now consider this equation **modulo n** . Since y is an integer, obviously $y \times n \bmod n$ equals 0. Thus, it must be the case that, considered **modulo n** , x equals a^{-1} , the multiplicative inverse of a modulo n .

- Eq. (7) shown above gives us a strategy for finding the multiplicative inverse of an element a :
 - We use the same Euclid algorithm as before to find the $\gcd(a, n)$,
 - but now at each step we write the expression in the form $a \times x + n \times y$ **for the remainder**
 - eventually, before we get to the remainder becoming 0, when the remainder becomes 1 (which will happen only when a and n are relatively prime), x will automatically be the multiplicative inverse we are looking for.

5.18: Revisiting Euclid's Algorithm for the Calculation of GCD

- Earlier in Section 5.8 we showed the following steps for a straightforward application of Euclid's algorithm for finding $\text{gcd}(b_1, b_2)$:

```

gcd( b1, b2 ):
= gcd( b2, b1 mod b2 )      = gcd( b2, b3 )
= gcd( b3, b2 mod b3 )      = gcd( b3, b4 )
= gcd( b4, b3 mod b4 )      = gcd( b4, b5 )
  ....
  ....
  ....

until ( b_{m-1} mod b_m ) == 0   in gcd( b_m, b_{m-1} mod b_m )
then gcd = b_m

```

- Next, let's make explicit the arithmetic operations required for carrying out the recursion at each step. This is shown on the next page.

5.19: Making Explicit the Arithmetic Operations in GCD Recursion

Making explicit the arithmetic operations required for the computation of the remainders on the previous page:

$$\begin{array}{l|l}
 \text{gcd}(b_1, b_2) : & \\
 \\
 = \text{gcd}(b_2, b_1 \bmod b_2) = \text{gcd}(b_2, b_3) & b_3 = b_1 - q_1.b_2 \\
 \\
 = \text{gcd}(b_3, b_2 \bmod b_3) = \text{gcd}(b_3, b_4) & b_4 = b_2 - q_2.b_3 \\
 \\
 = \text{gcd}(b_4, b_3 \bmod b_4) = \text{gcd}(b_4, b_5) & b_5 = b_3 - q_3.b_4 \\
 \dots & \dots \\
 \dots & \dots \\
 = \text{gcd}(b_{\{m-1\}}, b_m) & b_m = b_{\{m-2\}} - q_{\{m-2\}}.b_{\{m-1\}}
 \end{array}$$

until b_m is either 0 or 1.

if $b_m = 0$ and $b_{\{m-1\}}$ exceeds 1, then there does NOT exist a multiplicative inverse for b_1 in arithmetic modulo b_2

if $b_m = 1$, then there exists a multiplicative inverse for b_1 in arithmetic modulo b_2

Examples:

$$\begin{array}{l}
 \text{gcd}(4,2) = \text{gcd}(2,0) \\
 \text{therefore } 4 \text{ has no multiplicative inverse modulo } 2
 \end{array}$$

$$\begin{array}{l}
 \text{gcd}(3,7) = \text{gcd}(7,3) = \text{gcd}(3,1) \\
 \text{therefore there exists a multiplicative inverse for } 3 \text{ modulo } 7.
 \end{array}$$

5.20: What Conclusions Can We Draw From the Remainders?

- The final remainder is always 0. By remainder we mean the second argument in the recursive call to $gcd()$ at each step.
- If the next to the last remainder is greater than 1, this remainder is the GCD of b_1 and b_2 . Additionally, b_1 and b_2 are **NOT** relatively prime. **In this case, neither can have a multiplicative inverse modulo the other.**
- If the next to the last remainder is 1, the two input integers, b_1 and b_2 , are relatively prime. In this case, b_1 possesses a multiplicative inverse modulo b_2 .

5.21: Rewriting GCD Recursion in the Form of Derivations for Remainders

- We will now rewrite the equations by placing the remainders on the left hand side of the equations.

Note that before we get to the final remainder of 0, we are supposed to make sure that the remainder that comes just before the last is 1 (that is presumably the GCD of the two numbers if they are relatively prime):

$\text{gcd}(b_1, b_2)$:

$$b_3 = b_1 - q_1.b_2$$

$$\begin{aligned} b_4 &= b_2 - q_2.b_3 \\ &= b_2 - q_2.(b_1 - q_1.b_2) \\ &= b_2 - q_2.b_1 + q_1.q_2.b_2 \\ &= -q_2.b_1 + (1 + q_1.q_2).b_2 \end{aligned}$$

$$\begin{aligned} b_5 &= b_3 - q_3.b_4 \\ &= (b_1 - q_1.b_2) - q_3.(-q_2.b_1 + (1 + q_1.q_2).b_2) \\ &= b_1 + q_2.q_3.b_1 - q_1.b_2 - q_3.(1 + q_1.q_2).b_2 \\ &= (1 + q_2.q_3).b_1 - (q_1 - q_1.q_2 - q_3).b_2 \end{aligned}$$

.
.
.
.

$$b_m = (\dots\dots).b_1 \sim\sim\sim + \sim\sim\sim (\dots\dots).b_2$$

- Stop when b_m is 1 (that will happen when b_1 and b_2 are coprime). Otherwise, stop when b_m is 0, in which case there is no multiplicative inverse for b_1 modulo b_2 .
- If you stopped because b_m is 1, then the multiplier of b_1 in the expansion for b_m is the multiplicative inverse of b_1 modulo b_2 .
- When the above steps are implemented in the form of an algorithm, we have the **Extended Euclid's Algorithm**

5.22: Two Examples that Illustrate the Extended Euclid's Algorithm

Let's find the multiplicative inverse of 32 modulo 17:

$$\begin{array}{lcl}
 \gcd(32, 17) & & \\
 = \gcd(17, 15) & | \text{ residue } 15 & = 1x32 - 1x17 \\
 = \gcd(15, 2) & | \text{ residue } 2 & = 1x17 - 1x15 \\
 & | & = 1x17 - 1x(1x32 - 1x17) \\
 & | & = (-1)x32 + 2x17 \\
 = \gcd(2, 1) & | \text{ residue } 1 & = 1x15 - 7x2 \\
 & | & = 1x(1x32 - 1x17) \\
 & | & \quad - 7x((-1)x32 + 2x17) \\
 & | & = 8x32 - 15x17
 \end{array}$$

Therefore the multiplicative inverse of 32 modulo 17 is 8.

Let's now find the multiplicative inverse of 17 modulo 32:

$$\begin{array}{lcl}
 \gcd(17, 32) & & \\
 = \gcd(32, 17) & | \text{ residue } 17 & = 1x17 + 0x32 \\
 = \gcd(17, 15) & | \text{ residue } 15 & = 1x32 - 1x17 \\
 = \gcd(15, 2) & | \text{ residue } 2 & = 1x17 - 1x15 \\
 & | & = 1x17 - 1x(1x32 - 1x17) \\
 & | & = 2x17 - 1x32 \\
 = \gcd(2, 1) & | \text{ residue } 1 & = 15 - 7x2 \\
 & | & = (1x32 - 1x17) \\
 & | & \quad - 7x(2x17 - 1x32) \\
 & | & = (-15)x17 + 8x32 \\
 & | & = 17x17 + 8x32 \\
 & | & \text{(since the additive} \\
 & | & \text{inverse of 15 is 17 mod 32)}
 \end{array}$$

Therefore the multiplicative inverse of 17 modulo 32 is 17.

5.23: The Extended Euclidean Algorithm in Python

This function of two arguments returns the multiplicative inverse of the second argument, assuming the **first** argument to be the **modulus**. Obviously, the multiplicative inverse exists only when $\text{gcd}(a, b) = 1$. When the multiplicative inverse does not exist, it prints out “NO INVERSE” and also prints out the value of the gcd.

```
def mult_inverse(a,b):
    a1, a2 = 0L, a
    b1, b2 = 1L, b
    while b2 != 0:
        q = a2 // b2
        t = a1 - b1 * q, a2 - b2 * q
        a1, a2 = b1, b2
        b1, b2 = t
        print a1, a2, " ", b1, b2
    if a2 != 1:
        return "NO INVERSE: GCD of the two args is " + str(a2)
    else:
        if a1 < 0: a1 = a1 + a
        return a1
```