

# Modular Inverses for Public Key Cryptography

Dr. Demetrios Glinos  
University of Central Florida

CIS3360 - Security in Computing

# Readings

- "Computer Security: Principles and Practice", 3<sup>rd</sup> Edition, by William Stallings and Lawrie Brown
  - Appendix B.1, B.3

# Outline

- The Importance of Modular Inverses
- Prime Numbers
- What it means to be “relatively prime”
- Euclid’s GCD Algorithm
- Finding Modular Inverses
- Checking Candidate Modular Inverses
- The Extended Euclidean Algorithm

# The Importance of Modular Inverses

- Public key cryptosystems use two different keys (public and private)
- The keys are modular inverses of each other
- When we generate a new set of keys, we need to be able to choose one key at random and then determine the other key that is the first key's modular inverse
- We need to be able to find the modular inverse without guessing
  - guessing is too hard (too many choices) for real-world situations
  - this is what makes public key cryptosystems secure

# Prime Numbers

- Here, we are only talking about the *counting numbers* 1, 2, 3, ...
- A **prime number** is a counting number that is *evenly divisible* only by 1 and itself
  - *evenly divisible* means that the remainder is zero when the larger number is divided by the smaller of the two numbers
  - e.g., 12 is evenly divisible by 1, 2, 3, 4, and 6, but not by 5, 7, 8, 9, 10, or 11
- The first few prime numbers are: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, ...
- **Note 1:** “2” is the only even prime number; all other even numbers are evenly divisiby by 2, so they cannot be prime
- **Note 2:** all prime numbers other than 2 are odd, but not all odd number are prime
  - e.g., “15” is not prime, since it is divisible by 3 and 5
  - e.g., similarly for 9, 21, 25, 27, 33, and countless others

# What it means to be “relatively prime”

- Two counting numbers are **relatively prime** if their *greatest common divisor* is 1
- The **greatest common divisor (GCD)** is
  - something we calculate for *two input numbers*
  - it is *the largest number that divides evenly into both input numbers*
  - we write  **$GCD(a, b)$**  to mean the GCD of the two numbers  $a$  and  $b$

**Question:** Are the numbers 6 and 35 relatively prime?

**Answer:** Yes, even though neither 6 nor 35 are prime

6 is evenly divisible by 1, 2, 3, and 6

35 is evenly divisible by 1, 5, 7, and 35

→ The largest value that evenly divides both 6 and 35 is 1, so these two numbers are relatively prime

# Euclid's GCD Algorithm

- For large numbers, it is impractical to find a GCD by first finding all the factors of each input number and then checking for a number greater than 1 on both lists
- Uses the definition of **division**,  $a = (q)(b) + r$ , where
  - $a$  and  $b$  are the two given numbers
  - $q$  is the **quotient** and  $r$  is the **remainder**
- **Euclid's GCD Algorithm:**
  1. perform division using  $a$  and  $b$  to find  $q$  and  $r$
  2. then, assign  $a = b$  and  $b = r$
  3. repeat steps 1 and 2 until  $r = 0$
  4. GCD is the last non-zero remainder  $r$

**IMPORTANT:** For this class we will always divide the larger number by the smaller number, that is, we let "a" be the larger number

# Euclid's GCD Algorithm

## Euclid's GCD Algorithm:

1. perform division using a and b to find q and r
2. then, assign  $a = b$  and  $b = r$
3. repeat steps 1 and 2 until  $r = 0$
4. GCD is the last non-zero remainder r

Example: Let's use 6 and 35 again

Step 0: Let  $a = 35$  and  $b = 6$  ← we let the larger number be "a"

Step 1:  $35 = (5)(6) + 5$ , so  $q = 5$  and  $r = 5$

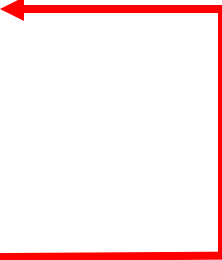
Step 2: assign  $a = 6$  and  $b = 5$

Repeat step 1:  $6 = (1)(5) + 1$ , so  $q = 1$  and  $r = 1$  ←

Repeat step 2: assign  $a = 5$  and  $b = 1$

Repeat step 1:  $5 = (5)(1) + 0$ , and *stop since  $r = 0$*

GCD is *last nonzero r*, which is 1 in this case



→ Since  $GCD(6, 35) = 1$ , the numbers 6 and 35 are relatively prime



# Finding Modular Inverses

- Recall the definition of a modular inverse:
  - Given counting numbers  $x$  and  $y$ , they are modular inverses with respect to a particular modulus  $n$  if  $(x)(y) \bmod n = 1$
  - Example (from before): 7 and 3 are inverses modulo 10,  
since  $(7)(3) \bmod 10 = 21 \bmod 10 = 1$
- Two ways to find modular inverses:
  - We can guess (aka “*trial and error*” or “*brute force attack*”) and then *check* to see if we are correct
  - Or, given  $x$  and  $n$ , we can use an algorithm to find  $y$ 
    - we can use the **Extended Euclidean Algorithm** to do this

# Review: Checking Candidate Modular Inverses

- Suppose we are given the number 7 and a modulus of 160 and we are asked to determine whether the number 23 is the modulo 160 inverse of 7.

**Question:** How do we proceed?

→ **Answer:** We just multiply it out using the definition of modular inverse;  
if the result is 1, then the two numbers are modular inverses

- Solution: We let  $x = 7$ ,  $y = 23$ , and  $n = 160$

We compute  $(7)(23) \bmod 160 = 161 \bmod 160 = 1$

→ *Since the result is 1, we conclude that 7 and 23 are inverses modulo 160*

# The Extended Euclidean Algorithm

- Used to **find** a modular inverse
- *Basic idea is to extend the standard Euclidean GCD algorithm by computing, for each standard Euclidean algorithm step 1 equation, a companion set of values*
  - those companion values will generate the desired solution
  - starts by numbering the equations, *starting with a counting index of 0*
- the companion set of values are called “**y**” values
  - y**<sub>0</sub> is always 0
  - y**<sub>1</sub> is always 1

for all other y values, we use the recurrence formula:  $y_i = y_{i-2} - (y_{i-1})(q_{i-2})$

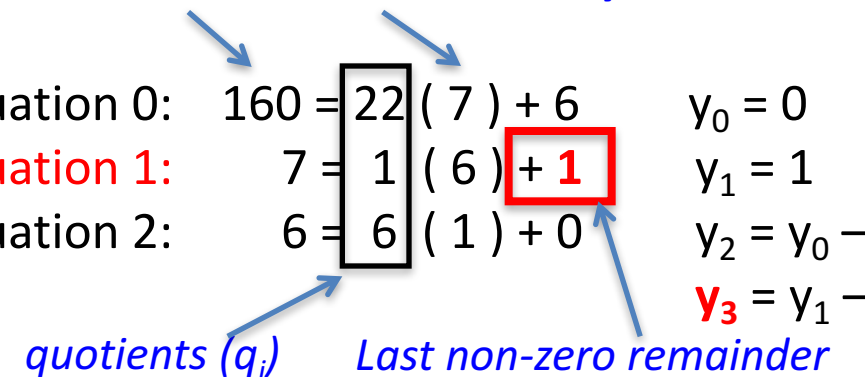
where **q**<sub>i-2</sub> is the Euclidean algorithm *quotient* for equation *i-2*

→ *The modular inverse is the y value for the equation whose counting index is 2 more than the index for the last nonzero Euclidean algorithm remainder*

# Extended Euclidean Algorithm Example

- We use the Euclidean Algorithm to find the *greatest common divisor (GCD)* and extend it to compute a *companion set of “y” values* (see below)
- FOR THIS TO WORK, ALWAYS DIVIDE THE LARGER NUMBER BY THE SMALLER NUMBER
- The *desired result* is the *y value whose index is 2 more than the index for the last non-zero remainder*.

Given:  $a = 160$  and  $b = 7$ , we find  $y$ , the mod  $a$  inverse of  $b$ , as follows:



Equation 0:  $160 = 22(7) + 6$        $y_0 = 0$

Equation 1:  $7 = 1(6) + 1$        $y_1 = 1$

Equation 2:  $6 = 6(1) + 0$        $y_2 = y_0 - (y_1)(q_0) = 0 - (1)(22) = -22$

$y_3 = y_1 - (y_2)(q_1) = 1 - (-22)(1) = +23$

quotients ( $q_i$ )      Last non-zero remainder

Here, the last non-zero remainder occurred in **Equation 1**, so the modular inverse is the 3<sup>rd</sup> y-value:  $y_3 = 23$  (same value we checked before)

**NOTE:** Start with  $y_0 = 0$  and  $y_1 = 1$  always; thereafter,  $y_i = y_{i-2} - (y_{i-1})(q_{i-2})$

# Extended Euclidean Algorithm Example 2

Given:  $a = 160$  and  $b = 43$ , we find  $y$ , the mod  $a$  inverse of  $b$ , as follows:

Equation 0:	$160 = 3 ( 43 ) + 31$	$y_0 = 0$
Equation 1:	$43 = 1 ( 31 ) + 12$	$y_1 = 1$
Equation 2:	$31 = 2 ( 12 ) + 7$	$y_2 = y_0 - ( y_1 )( q_0 ) = 0 - (1)(3) = -3$
Equation 3:	$12 = 1 ( 7 ) + 5$	$y_3 = y_1 - ( y_2 )( q_1 ) = 1 - (-3)(1) = +4$
Equation 4:	$7 = 1 ( 5 ) + 2$	$y_4 = y_2 - ( y_3 )( q_2 ) = -3 - (4)(2) = -11$
Equation 5:	$5 = 2 ( 2 ) + 1$	$y_5 = y_3 - ( y_4 )( q_3 ) = 4 - (-11)(1) = 15$
Equation 6:	$2 = 2 ( 1 ) + 0$	$y_6 = y_4 - ( y_5 )( q_4 ) = -11 - (15)(1) = -26$
		$y_7 = y_5 - ( y_6 )( q_5 ) = 15 - (-26)(2) = 67$

Here, the last non-zero remainder occurred in **Equation 5**, so the modular inverse is the 7<sup>th</sup>  $y$ -value, that is:  $y_7 = 67$

We check our result:  $(43)(67) \bmod 160 = 2881 \bmod 160 = 1$ , so our result is correct

*Q: If we had tried guessing, what are the chances we would have guessed 67?*