# Applied Cryptography and Computer Security

# CSE 664 Spring 2017

## Lecture 12: Introduction to Number Theory II

**Department of Computer Science and Engineering**
**University at Buffalo**

# Lecture Outline

- **This time we'll finish the intro to number theory**
- **What to expect:**
  - **congruences**
  - **Fermat and Euler's theorems**
  - **the Chinese remainder theorem**
  - **finding large primes**

# Congruences

- **A congruence is a statement about divisibility**
  - **such statements simplify reasoning about divisibility**
- **Definition**
  - **let $a$, $b$, $m > 0$ be integers**
  - **if $m$ divides $a - b$, then $a$ is congruent to $b$ modulo $m$ and we write $a \equiv b \pmod{m}$**
  - **if $m$ does not divide $a - b$, $a$ is not congruent to $b$ modulo $m$ and we write $a \not\equiv b \pmod{m}$**
  - **the formula $a \equiv b \pmod{m}$ is called a congruence**
  - **the integer $m$ is called the modulus**

# Congruences

- **Do not confuse $a \equiv b \pmod{m}$ with binary operator "mod"**
  - $a \equiv b \pmod{m}$ **if and only if** $(a \bmod m) = (b \bmod m)$
- **For each integer $a$, the set of all integers $b \equiv a \pmod{m}$ is called the congruence class or residue class of $a$ modulo $m$**
  - **example: the residue class of $27 \pmod{5}$ is** $\ldots, -13, -8, -3, 2, 7, 12, \ldots$
  - **each value is a representative of the class, and the smallest positive value is the standard representative**

# Congruences

- **The congruence relation has many similarities to equality**
  - **it, like equality, is an equivalence relation**
  - **reflexive:** $a \equiv a \pmod{m}$
  - **symmetric: if** $a \equiv b \pmod{m}$**, then** $b \equiv a \pmod{m}$
  - **transitive: if** $a \equiv b \pmod{m}$ **and** $b \equiv c \pmod{m}$**, then** $a \equiv c \pmod{m}$

# Congruences

- **Properties of congruence relations**
  - **let** $a \equiv b \pmod{m}$ **and** $c \equiv d \pmod{m}$
  - $a + c \equiv b + d \pmod{m}$
  - $a - c \equiv b - d \pmod{m}$
  - $ac \equiv bd \pmod{m}$
  - **let** $f$ **be a polynomial with integer coefficients, then if** $a \equiv b \pmod{m}$**,** $f(a) \equiv f(b) \pmod{m}$
  - **let** $d|m$**, then** $a \equiv b \pmod{m} \;\Rightarrow\; a \equiv b \pmod{d}$

# Congruences

- **Although addition, subtraction, and multiplication follow the usual rules, division does not always work as expected**
  - $ac \equiv bc \pmod{m}$ **does not always imply** $a \equiv b \pmod{m}$
  - **example:** $2 \cdot 3 = 6 \equiv 18 = 2 \cdot 9 \pmod{12}$**, but** $3 \not\equiv 9 \pmod{12}$
  - **we next investigate when this implication is true**

- **Theorem (division):**
  - **for integer** $a$**,** $b$**,** $c \neq 0$**, and** $m > 0$**, if** $gcd(c, m) = 1$**, then** $ac \equiv bc \pmod{m}$ **implies** $a \equiv b \pmod{m}$
  - **example:** $5 \cdot 3 = 15 \equiv 39 = 13 \cdot 3 \pmod{8}$**; both** $15 \equiv 39 \pmod{8}$ **and** $5 \equiv 13 \pmod{8}$

# Congruences

- **Theorem (multiplicative inverse):**
  - **if** $gcd(a, m) = 1$**, then there is a unique** $x$ **(**$0 < x < m$**) such that** $ax \equiv 1 \pmod{m}$**, i.e.,** $x$ **is** $a^{-1} (\mathrm{mod}\ m)$
  - **example:** $a = 3$**,** $m = 5$**;** $x \equiv 2 \equiv 3^{-1} \pmod{5}$
  - **the inverse is normally computed using the extended Euclidean algorithm, where** $ax + my = 1$

# Residue Sets

- A **complete set of residues** (CSR) modulo $m$ is a set $S$ of integers such that every integer is congruent to exactly one integer in that set $S$
  - the **standard CSR** modulo $m$ is $\{0, 1, \ldots, m-1\}$, i.e, $\mathbb{Z}_m$
- A **reduced set of residues** (RSR) modulo $m$ is a set $R$ of integers such that every integer **relatively prime to $m$** is congruent to exactly one integer in $R$
  - the **standard RSR** modulo $m$ is all $1 \leq r \leq m$ such that $gcd(r, m) = 1$
  - example: for $m = 12$, the standard RSR is $\{1, 5, 7, 11\}$
  - for a prime $p$, this set is $\{1, 2, \ldots, p-1\}$

# Linear Congruences

- **Now how do we solve congruences $ax \equiv b \pmod{m}$ for given $a$, $b$, $m$ and unknown $x$?**
  - **we first need to determine when they are solvable**
- **Theorem (solvability of linear congruence)**
  - $ax \equiv b \pmod{m}$ **has a solution if and only if** $gcd(a, m)$ **divides** $b$
  - **example:**
    - **solve** $165x \equiv 100 \pmod{285}$
    - **?**

## Linear Congruences

- **Theorem (solution to a linear congruence)**
  - **let** $g = gcd(a, m)$
  - **if** $g$ **divides** $b$**, then** $ax \equiv b \pmod{m}$ **has** $g$ **solutions**
  - **the solutions are:**

$$x \equiv \frac{b}{g} x_0 + t\frac{m}{g} \pmod{m}, \ \ t = 0, 1, \ldots, g-1$$

  - **here** $x_0$ **is any solution to** $\frac{a}{g} x_0 \equiv 1 (\bmod \frac{m}{g})$

# Linear Congruences

- **Example of a linear congruence**
  - **solve** $7x \equiv 3 \pmod{12}$
  - **first find** $g =$
  - **determine the number of solutions**
  - **determine** $x_0$
  - **find the solution**
  - **now solve** $8x \equiv 4 \pmod{12}$

# Groups

- A **group** $G$ is a set of elements together with a binary operation $\circ$ such that
  - the set is **closed under the operation** $\circ$, i.e., for every $a, b \in G$, $a \circ b$ is a unique element of $G$
  - the **associative law holds**, i.e., for all $a, b, c \in G$, $a \circ (b \circ c) = (a \circ b) \circ c$
  - the set has a **unique identity element** $e$ such that $a \circ e = e \circ a = a$ for every $a \in G$
  - every element has a **unique inverse** $a^{-1}$ in $G$ such that $a \circ a^{-1} = a^{-1} \circ a = e$

# Groups

- **A group is called commutative or abelian if $a \circ b = b \circ a$ for every pair $a, b \in G$**
- **Size of a group**
  - **a group is finite if it has only a finite number of elements**
  - **a group is infinite if it has an infinite number of elements**
  - **the number of elements of a finite group is called the order of the group**
- **Groups is a convenient way to represent sets by strings of symbols**

# Groups

- **Examples of groups**
  - **the set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ forms an infinite abelian group**
    - **addition is the binary operation**
    - **0 is the identity**
    - **$-a$ is the inverse of $a$**
  - **this set does not form a group with multiplication as the binary operation (lack of inverses)**

## Groups

- **Examples of groups**
  - **if $m \geq 2$ is an integer, a complete set of residues (CSR) modulo $m$ forms an abelian group**
    - **addition modulo $m$ is the binary operation**
    - **the residue class containing 0 is the identity**
    - **the inverse of the residue class containing $a$ is the residue class containing $-a$**
  - **this group is called the additive group modulo $m$**
  - **a CSR modulo $m$ does not form a group under multiplication**

# Groups

- **Examples of groups**
  - **recall that a reduced set of residues (RSR) includes all numbers relatively prime to $m$**
  - **for $m > 1$, a RSR modulo $m$ forms a group with multiplication modulo $m$ as operation**
  - **the identity element is the residue class containing 1**
  - **it is called the multiplicative group modulo $m$**
  - **what is the group order?**

# Euler's $\phi$ Function

- **Euler $\phi$ function**
  - **$\phi(m)$ is the size of RSR modulo $m$**
  - **$\phi$ is called the Euler Phi or totient function**
- **Properties of $\phi$**
  - **if $p$ is prime, $\phi(p) = p - 1$**
  - **$\phi$ is multiplicative: $\phi(ab) = \phi(a)\phi(b)$**
  - **thus, if $p \neq q$ are primes, $\phi(pq) = (p-1)(q-1)$**
  - **if $p$ is prime, $\phi(p^e) = p^e - p^{e-1}$**
  - **if $n = \prod_i p_i^{e_i}$, where $p_i$'s are distinct primes and $e_i \geq 1$,**
    $\phi(n) = \prod_i p_i^{e_i-1}(p_i - 1)$

# Fermat and Euler’s Theorems

- **Fermat’s “Little” Theorem**
  - **let $p$ be prime and $a$ be an integer which is not a multiple of $p$, then**

$$a^{p-1} \equiv 1 \pmod{p}$$

- **Euler’s Theorem**
  - **let $m > 1$ and $gcd(a, m) = 1$, then**

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

- **A Corollary of Euler’s Theorem**
  - **let $m$, $x$, $y$, and $g$ be positive integers with $gcd(g, m) = 1$**
  - **if $x \equiv y \pmod{\phi(m)}$, then $g^x \equiv g^y \pmod{m}$**

# Fermat and Euler's Theorems

- **Another corollary of Euler's theorem**
  - **we obtain an alternative way of computing** $a^{-1} (\text{mod } m)$
    - **recall that** $a \cdot a^{-1} \equiv 1 \pmod m$
  - **factoring out one** $a$ **gives us** $aa^{\phi(m)-1} \equiv 1 (\text{mod } m)$
  - **then** $a^{-1} \equiv$
  - **for a prime modulus** $p$**,** $a^{-1} \equiv$
  - **computing the inverse using this approach requires roughly the same number of bit operations as the extended Euclidean algorithm**

# More on Groups

- **If $a$ is an element of a finite group with identity 1, then there is a unique smallest positive integer $i$ with $a^i = 1$ (using multiplicative notation)**
  - **such $i$ is called the order of $a$ (different from the order of the group)**
- **The element $a$ has infinite order is there is no positive integer $i$ with $a^i = 1$**
- **A cyclic group is one that contains an element $a$ whose powers $a^i$ and $a^{-i}$ make up the entire group**
- **An element $a$ with such property is called a generator of the group**

# Cyclic Groups

- **Examples**
  - **the set of all integers with $+$ for the operation is a cyclic group of infinite order**
    - **the group is generated by 1**
    - **the "powers" of 1 are $0, \pm 1, \pm 2, \ldots$**
    - **every element $a \neq 0$ has infinite order**
  - **the integers modulo $m$ with $+$ operation form a cyclic group of order $m$, where the residue class of 1 is a generator**
  - **the multiplicative group modulo $m$, $\mathbb{Z}_m^*$, may or may not be cyclic depending on $m$**

# Cyclic Groups

- **Theorem:** **If $p$ is prime, then $\mathbb{Z}_p^*$ is cyclic.**
- **Example**
  - **consider multiplicative group over $\mathbb{Z}_7^*$**
  - **what is the order of 2?**
  - **what is the order of 3?**

# Fast Exponentiation

- **We'll need to compute $a^n$ often**
- **This can be done using only $O(\log_2 n)$ multiplications**

**power(**$a$**,** $n$**)** $\{$

$e = n$**;** $y = 1$**;** $z = a$**;**

**repeat** $\{$

**if (**$e$ **is odd)** $y = y \cdot z$**;**

**if (**$e \leq 1$**) return** $y$**;**

$z = z \cdot z$**;**

$e = e \gg 1$**;** $\longleftarrow e = \lfloor e/2 \rfloor$

$\}$

$\}$

# Fast Exponentiation

- To compute $a^n \bmod m$, we want to keep numbers small (smaller than $m$)

- We reduce them modulo $m$ after each multiplication

**power**($a$, $n$, $m$) $\{$
    $e = n$; $y = 1$; $z = a$;
    **repeat** $\{$
        **if** ($e$ **is odd**) $y = (y \cdot z) \ \% \ m$;
        **if** ($e \leq 1$) **return** $y$;
        $z = (z \cdot z) \ \% \ m$;
        $e = e \gg 1$;
    $\}$
$\}$

## Fast Exponentiation

- **Example: compute $3^6 \bmod 11$**
  - **set $e = 6$ (0110), $y = 1$; $z = 3$**
  - **execute the loop**
    - **iteration 1**
    - **iteration 2**
    - **iteration 3**
- **What's the complexity of fast exponentiation?**

## The Chinese Remainder Theorem

- **The Chinese Remainder Theorem (CRT) can be used to perform modular exponentiations even faster than in the above algorithm**

- **The main advantage of CRT:**
  - **it allows us to split up one large exponentiation into smaller exponentiations**

- **The main idea:**
  - **for a composite number $m$ with factors $p_1, p_2, \ldots$, it allows us to combine congruences of the form $x \equiv a_i \pmod{p_i}$ into a congruence $x \equiv a \pmod{m}$**

- **Main uses:**
  - **in public-key decryption and signing algorithms**

# The Chinese Remainder Theorem

- **The Chinese Remainder Theorem**
  - **we are given $n_1, \ldots, n_r$ positive integers pair-wise relatively prime (i.e., $gcd(n_i, n_j) = 1$ for any $i \neq j$)**
  - **let $n = n_1 \cdots n_r$**
  - **then $r$ congruences $x \equiv a_i \pmod{n_i}$ have common solutions modulo $n$**

- **The solution to such congruences is**

$$x \equiv \sum_{i=1}^{r} (n/n_i) b_i a_i \pmod{n}$$

  - **here $b_i \equiv (n/n_i)^{-1} (\text{mod } n_i)$**

# The Chinese Remainder Theorem

- **Example:**
  - **solve a system of congruences $x \equiv 1 \pmod{7}$, $x \equiv 3 \pmod{10}$, and $x \equiv 8 \pmod{13}$**

# Finding Large Primes

- In many constructions we rely on large primes
- How do we find them?
  - the probability that a randomly picked integer, say, 2000 bits long is prime is not great
- But even if we have a candidate, how do we test it?
  - Fermat's theorem says that if $p$ is prime and $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$
  - this theorem gives us a test for compositeness
  - if $p$ is odd, $p \not| a$, and $a^{p-1} \not\equiv 1 \pmod{p}$, then $p$ is not prime
  - how about the converse, a test for primality?

## Finding Large Primes

- **Unfortunately, the converse is not always true**
  - **consider** $p = 11 \cdot 13 = 341$ **and** $a = 2$; $2^{340} \equiv 1 \pmod{341}$
  - **it is, however, true for most** $p$ **and** $a$
- **The composite numbers that pass such "primality test" are called Carmichael numbers (pseudo-prime)**
  - **they result in** $a^{p-1} \equiv 1 \pmod{p}$ **for every integer a with** $gcd(a, p) = 1$
  - **there are infinitely many of them**
  - **they must be detected and avoided in cryptosystems like RSA**

# Finding Large Primes

- **But there is a true converse of Fermat's theorem**
- **Lucas-Lehmer test (rigorous primality test):**
  - **let $n > 3$ be odd**
  - **if for every prime $p$ that divides $n-1$ there exists $a$ such that $a^{n-1} \equiv 1 \pmod{n}$, but $a^{(n-1)/p} \not\equiv 1 \pmod{n}$, then $n$ is prime**
  - **using the test requires knowledge of factorization of $n-1$**
- **This theorem can be used iteratively to construct large, random primes**
  - **start with a rather small prime and make it several digits longer in each step**
  - **test for primality in each iteration**

# Finding Large Primes

- **Constructing large primes:**
  - **begin with a prime $p_1$ and let $i = 1$**
  - **repeat the following steps until $p_i$ is large enough**
    - **for a random small $k$ (9–10 digits), let $n = 2kp_i + 1$**
    - **if $2^{n-1} \not\equiv 1 (\bmod\ n)$, then $n$ is composite and try another $k$**
    - **otherwise, $n$ is probably prime, so try to prove it using Lucas-Lehmer test**
    - **if you succeed in finding the base $a$ to satisfy the test, then $n$ is proved prime and set $p_{i+1} = n$**
    - **otherwise try a new random $k$**

# Finding Large Primes

- **Using Lucas-Lehmer approach adds about 10 digits to the length of the prime in each step**
- **It is possible to construct large primes faster**
  - **we can double the size of the prime in one step**
  - **complete factorization of the candidate prime is not required**
- **Pocklington-Lehmer theorem allows us to do so**
  - **given prime $p_i$ set $n$ to $2Fp_i + 1$, where factorization of $F$ is not known**
  - **the idea is that if $p_i \geq \sqrt{n}$, then $n$ is prime**

# More on Primality Tests

- Given a large number $n$, can we test whether it is prime without other conditions?
- History of primality tests development
  - trying all numbers up to $\sqrt{n}$ works, but is inefficient
    - this algorithm has been known for over 2000 years
  - applying Fermat's theorem is efficient, but not always works
    - Carmichael numbers satisfy the test as well
    - this theorem was the basis for many efficient primality tests

# More on Primality Tests

- **History of primality tests development**
  - **In 1970s randomized polynomial-time algorithms have been developed**
    - **Miller-Rabin test determines composite numbers with probability at least $1 - 4^{-k}$ for a chosen $k$**
    - **Solovay-Strassen test determines composite numbers with probability at least $1 - 2^{-k}$**
  - **In 1983 Adleman, Pomerance, and Rumely achieved a breakthrough**
    - **they gave the first deterministic test that doesn't require exponential time**
    - **the algorithm runs in $(\log n)^{O(\log \log \log n)}$**

# More on Primality Tests

- **History of primality tests development**
  - **Finally, in 2004 Agrawal, Kayal, and Saxena proved that PRIMES is in P**
    - **their deterministic algorithm runs in $O((\log n)^{15/2})$ time or better**
    - **the algorithm is based on a generalization of Fermat's theorem**
- **History happens even now!**

## Summary

- **Congruences are statements about divisibility**
  - **their properties often coincide with our intuition, but they also differ**
- **Fermat and Euler's theorems**
  - **provide an alternative way of computing an inverse modulo a number**
  - **provide a compositeness test**
- **To find a large prime either**
  - **choose a value at random and test for primality**
  - **construct a prime from smaller values**
- **As of 2004, unconditional primality testing is in P**