

Cryptography and Network Security Number Theory

Xiang-Yang Li

CS595-Cryptography and Network Security

Introduction to Number Theory

- Divisors
 - $b|a$ if $a=mb$ for an integer m
 - $b|a$ and $c|b$ then $c|a$
 - $b|g$ and $b|h$ then $b|(mg+nh)$ for any int. m,n
- Prime number
 - p has only positive divisors 1 and p
- Relatively prime numbers
 - No common divisors for p and q except 1

CS595-Cryptography and Network Security

GCD

- Greatest common divisor $\gcd(a,b)$
 - The largest number that divides both a and b
- Euclid's algorithm
 - Find the GCD of two numbers a and b , $a < b$
- Use fact if a and b have divisor d so does $a-b$, $a-2b \dots$

CS595-Cryptography and Network Security

Cont.

- GCD (a,b) is given by:
 - let $g_0=b$
 - $g_1=a$
 - $g_{i+1} = g_{i-1} \bmod g_i$
 - when $g_i=0$ then $\gcd(a,b) = g_{i-1}$
- The algorithm terminates in $O(\log b)$ rounds
 - Why?

CS595-Cryptography and Network Security

Properties

- For any two integers a and b
 - Exist integers m and n : $\gcd(a,b) = ma+bn$
 - Example:
 - $a=2, b=3$; we choose $m=-1, n=1$ so $-2+3=1$
 - $a=6, b=11$; we choose $m=2, n=-1$ so $2*6-11=1$
 - Simple proof?
- Integer a can be factored as
 - $a=p_1^{a_1} p_2^{a_2} p_3^{a_3} \dots p_n^{a_n}$ where p_i is prime number

CS595-Cryptography and Network Security

Modular Arithmetic

- Congruence
 - $a \equiv b \pmod n$ says when divided by n that a and b have the same remainder
 - It defines a relationship between all integers
 - $a \equiv a$
 - $a \equiv b$ then $b \equiv a$
 - $a \equiv b, b \equiv c$ then $a \equiv c$

CS595-Cryptography and Network Security

Cont.

□ addition

➤ $(a+b) \bmod n \equiv (a \bmod n) + (b \bmod n)$

□ subtraction

➤ $a-b \bmod n \equiv a+(-b) \bmod n$

□ multiplication

➤ $a*b \bmod n$

➤ derived from repeated addition

➤ Possible: $a*b \equiv 0$ where neither $a, b \equiv 0 \bmod n$

CS595-Cryptography and Network Security

Cont.

□ Division

➤ $a/b \bmod n$

➤ multiplied by inverse of b: $a/b = a*b^{-1} \bmod n$

➤ $b^{-1}*b \equiv 1 \bmod n$

➤ $3^{-1} \equiv 7 \bmod 10$ because $3*7 \equiv 1 \bmod 10$

➤ Inverse does not always exist!

- Only when $\gcd(b,n)=1$

CS595-Cryptography and Network Security

Addition and Multiplication

□ Integers modulo n with addition and multiplication form a commutative ring with the laws of

➤ Associativity

- $(a+b)+c \equiv a+(b+c) \bmod n$

➤ Commutativity

- $a+b \equiv b+a \bmod n$

➤ Distributivity

- $(a+b)*c \equiv (a*c)+(b*c) \bmod n$

CS595-Cryptography and Network Security

Galois Field

□ If n is constrained to be a prime number p then this forms a **Galois field modulo p** denoted **GF(p)** and all the normal laws associated with integer arithmetic work

□ Exponentiation

➤ $b = a^c \bmod p$

□ Discrete Logarithms

➤ find x where $a^x = b \bmod p$

CS595-Cryptography and Network Security

Inverses and Euclid's Extended GCD Routine

□ If $\gcd(a,n)=1$ then the inverse always exists

□ Can extend Euclid's algorithm to find inverse by keeping track of $g_i = u_i*n + v_i*a$

□ Extended Euclid's (or binary GCD) algorithm to find inverse of a number $a \bmod n$ (where $\gcd(a,n)=1$) is:

CS595-Cryptography and Network Security

Inverse

□ Inverse(a,n) is given by:

➤ $X=(x1,x2,x3)=(1,0,n)$; $Y=(y1,y2,y3)=(0,1,a)$

➤ If $y3=0$ return $x3=\gcd(a,n)$; no inverse

➤ If $y3=1$ return $y3=\gcd(a,n)$; $y2=a^{-1} \bmod n$

➤ $Q=[x3/y3]$

➤ $T=X-Q*Y$

➤ $X=Y$; $Y=T$

➤ Goto 2nd step

CS595-Cryptography and Network Security

Proof

- Assume we compute $\gcd(x_0, y_0)$
 - Let $X_i = (x_i, y_i)$; $0 \leq x_i - q_{i+1}y_i < |y_i|$
 - Then $X_i = M_i X_{i-1}$, where $M_i = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix}$
 - Assume the gcd algorithm terminates in n steps
 - We have $M_n M_{n-1} \dots M_1 X_0 = (\gcd(x_0, y_0), 0)^T$
 - Assume $M_n M_{n-1} \dots M_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
 - Then $ax_0 + by_0 = \gcd(x_0, y_0)$
 - The above algorithm is to keep track of a, b, c, d , and x_i, y_i values.

CS595-Cryptography and Network Security

Euler Totient Function

- If consider arithmetic modulo n , then a **reduced set of residues** is a subset of the complete set of residues modulo n which are relatively prime to n
 - eg for $n=10$,
 - the complete set of residues is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
 - the reduced set of residues is $\{1, 3, 7, 9\}$
- The number of elements in the reduced set of residues is called the **Euler Totient function $\phi(n)$**

CS595-Cryptography and Network Security

Euler's Theorem

- Let $\gcd(a, n) = 1$ then
 - $a^{\phi(n)} \pmod n = 1$
 - Proof: consider all reduced residues x_i
 - Then ax_i also form reduced residues set
 - Using $\prod ax_i = \prod x_i \pmod n$ and $\prod x_i$ has inverse
- Compute $\phi(n)$
 - If factoring of n is known
 - $\phi(n) = n \prod (1 - 1/p_i)$ where p_i is its prime factor
 - Otherwise
 - It is expensive!

CS595-Cryptography and Network Security

Fermat's Theorem

- Let p be a prime and $\gcd(a, p) = 1$ then
 - $a^{p-1} \pmod p = 1$
 - Proof: similar to the proof of Euler's theorem
 - But consider all integers in Z_p
- Generally, for any prime number p
 - $a^p \pmod p = a$

CS595-Cryptography and Network Security

Chinese Remainder Theorem

- Let m_1, m_2, \dots, m_k be pair-wise relative prime numbers
- Assume integer $A = a_i \pmod{m_i}$
- Then $A = \sum a_i C_i \pmod M$
 - Where $M = \prod m_i$; $M_i = M / m_i$
 - $C_i = M_i * (M_i^{-1} \pmod{m_i})$

CS595-Cryptography and Network Security

Primality Testing

- To check if exists integer a such that a/p
 - Primary school method
 - Two slow!
 - Check $n^{0.5}$ numbers
 - At least around $(n/\ln n) n^{0.5}$ numbers need be checked
- Any improvement?

CS595-Cryptography and Network Security

Simple Fact

- Equation $x^2 \equiv 1 \pmod p$ has only solutions $1, -1$
 - If p is prime number
 - Simple proof: $(x+1)(x-1) \equiv 0 \pmod p$
- So if we find another solution, then p can not be prime number!
 - Miller and Rabin 1975,1980
- Randomly chosen integer a
 - If $a^2 \equiv 1 \pmod p$ then p is not prime number
 - Integer a is called the witness
 - Otherwise p maybe, or maybe not a prime number

CS595-Cryptography and Network Security

Witness Algorithm

- Witness(a,n)
 - Let $b_k b_{k-1} \dots b_1 b_0$ be the binary code of $n-1$
 - Let $d=1$
 - For $i=k$ downto 0
 - $x=d; d=d*d \pmod n$
 - If $d=1$ and $x \neq 1$, and $x \neq n-1$
 - return TRUE
 - If $b_i=1$ then $d=d*a \pmod n$
 - Endfor
 - If $d \neq 1$ then return TRUE
 - Return FALSE

CS595-Cryptography and Network Security

Facts

- Analysis the result of witness
 - If returns TRUE, then n is not prime number
 - Find other solutions for $x^2 \equiv 1 \pmod n$
 - Otherwise, n maybe prime number
- Given odd n and random a
 - Witness fails with probability less than 0.5
- Run witness algorithm s times
 - If one time, it is TRUE
 - Then n is not prime number
 - Otherwise, $\Pr(n \text{ is prime}) > 1-2^{-s}$

CS595-Cryptography and Network Security

Randomized Methods

- Las Vegas Method
 - Always produces correct results
 - Runs in expected polynomial time
- Monte Carlo Method
 - Runs in polynomial time
 - May produce incorrect results with bounded probability
 - No-Biased Monte Carlo Method
 - Answer yes is always correct, but the answer no may be wrong
 - Yes-biased Monte Carlo Method
 - Answer no is always correct, but the answer yes may be wrong

CS595-Cryptography and Network Security

Witness Algorithm

- Witness Algorithm is based on Monte Carlo Method
 - It actually test compositeness, not primality
 - When it reports yes, the number is always composite
 - When it reports no, input may be composite, prime
 - Probability Result
 - $\Pr(\text{input=composite} \mid \text{ans=composite}) = 1$
 - $\Pr(\text{ans=no} \mid \text{input=composite}) < 1/2$
 - $\Pr(\text{input=composite} \mid \text{ans=no}) \leq 1/4$

CS595-Cryptography and Network Security

Time Complexity

- Each round of witness cost $O(\log n)$
 - Unit: integer multiplication and modular arithmetic
- So the primality testing cost $O(s \log n)$
 - The confidence is $1-2^{-s}$ if report prime
 - The confidence is 1 if report non-prime

CS595-Cryptography and Network Security

Primitive Root

- Order of integer
 - The order of a modulo n is the smallest positive k such that $a^k \equiv 1 \pmod{n}$
- Primitive Root
 - Integer a is a primitive root of n if the order of a modulo n is $\phi(n)$
 - Not all integers have primitive root
 - Example $n=pq$ for primes p and q
 - Prime p has $\phi(p-1)$ primitive roots

CS595-Cryptography and Network Security

Discrete Logarithms

- $Y \equiv g^x \pmod{p}$
 - Compute x
 - Time complexity $O(e^{(\ln p)^{1/3}(\ln \ln p)^{2/3}})$

CS595-Cryptography and Network Security

Quadratic Residue

- Quadratic Residue
 - Integer b is a quadratic residue of integer n if and only if $x^2 \equiv b \pmod{n}$ has a solution for x
 - Otherwise b is called quadratic nonresidue
- Given odd prime p ,
 - b is quadratic residue, iff $b^{(p-1)/2} \equiv 1 \pmod{p}$
 - b is quadratic nonresidue, iff $b^{(p-1)/2} \equiv -1 \pmod{p}$

CS595-Cryptography and Network Security

Complexity Theory

- The **input length** of a problem is the number n of symbols used to characterize it
- Function $f(n)$ is order $O(g(n))$ if
 - $f(n) \leq c \cdot |g(n)|$, for all $n \geq N_0$, for some c
- **Polynomial time algorithm (P)**
 - solves any instance of a particular problem with input length n in time $O(p(n))$, where p is a polynomial

CS595-Cryptography and Network Security

Cont.

- **Non-deterministic polynomial time algorithm (NP)** - is one for which any guess at the solution of an instance of the problem may be checked for validity in polynomial time.
- **NP-complete** problems - are a subclass of NP problems for which it is known that if any such problem has a polynomial time solution, then **all** NP problems have polynomial solutions.
- **Co-NP**: the complements of NP problems.

CS595-Cryptography and Network Security