Cryptography and Network Security: Principles and Practice (5e)

by William Stallings

Chapter 8

More Number Theory

Chapter 8. More Number Theory

8.1 Prime Numbers

8.2 Fermat's and Euler's Theorems

- Fermat's Theorem, Euler's Totient Function, Euler's Theorem

8.3 Testing for Primality

- Miller-Rabin Algorithm, A Deterministic Primality Algorithm, Distribution of Primes

8.4 The Chinese Remainder Theorem

8.5 Discrete Logarithms

- The Powers of an Integer Modulo n, Logarithms for Modular Arithmetic, Calculation of Discrete Logarithms

Appendix:

- Square Root of 1, Modular Linear Equations, Subgroup, Factorization

Prime Numbers

- central concept to number theory
- prime numbers only have divisors of 1 and itself
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
  - eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- list of prime number less than 200 is:

  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Prime Factorisation

- to **factor** a number **n** is to write it as a product of other numbers: $\mathbf{n = n_1 \cdot n_2 \cdot n_3}$
- factoring relatively hard compared to multiplying the factors together
- the **prime factorisation** of a number **n** is to write it as a product of primes
  - eg. $\mathbf{91=7\cdot 13\ ;\ 3600=2^4\cdot 3^2\cdot 5^2}$

$$a = \prod_{p \in \mathrm{P}} p^{a_p}$$

GCD by Prime Factorization

- It is easy to determine the greatest common divisor by comparing their prime factors and by using least powers
  - eg. $300=2^1\cdot3^1\cdot5^2$ and $18=2^1\cdot3^2$
  - hence $GCD(18,300)=2^1\cdot3^1\cdot5^0=6$

Chapter 8. Introduction to Number Theory

8.1 Prime Numbers

8.2 Fermat's and Euler's Theorems

Fermat's Theorem, Euler's Totient Function, Euler's Theorem

8.3 Testing for Primality

Miller-Rabin Algorithm, A Deterministic Primality Algorithm, Distribution of Primes

8.4 The Chinese Remainder Theorem

8.5 Discrete Logarithms

The Powers of an Integer Modulo n, Logarithms for Modular Arithmetic, Calculation of Discrete Logarithms

Appendix:

Square Root of 1, Modular Linear Equations, Subgroup, Factorization

Fermat's (Little) Theorem

- $a^{p-1} = 1 \pmod{p}$
  - where p is prime and $\gcd(a,p)=1$
  - useful in public key and primality testing
- Recall the congruence:

$$x = y \pmod{n}$$

  - when divided by n, x and y have same remainder

$$
\begin{aligned}
a &= 7, p = 19 \\
7^2 &= 49 \equiv 11 \pmod{19} \\
7^4 &\equiv 121 \equiv 7 \pmod{19} \\
7^8 &\equiv 49 \equiv 11 \pmod{19} \\
7^{16} &\equiv 121 \equiv 7 \pmod{19} \\
a^{p-1} &= 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}
\end{aligned}
$$

Fermat's (Little) Theorem

- Proof:

  Consider $Z^+_p = \{1,2,\ldots,p-1\}$ and

  $X = \{1a \bmod p, 2a \bmod p, \ldots, (p-1)a \bmod p\}$

  None in X is equal to 0 as p does not divide a

  No two in X are equal (by contradiction).

  Therefore, $X = Z^+_p$ (though elements in different orders).

  Multiply the numbers in both sets and taking mod p

  $1a \cdot 2a \cdot \ldots \cdot (p-1)a \bmod p = a^{p-1} \cdot (p-1)! \bmod p = (p-1)! \bmod p$

- also $a^p = a \pmod p$ not requiring $\gcd(a,p)=1$

Euler Totient Function **ø(n)**

- **complete set of residues**: **`0..n-1`**, i.e., $\mathbf{Z_n}$
  - when doing arithmetic modulo n
- **reduced set of residues**: those residues which are relatively prime to n, i.e., $\mathbf{Z^*_n}$
  - e.g., for n=10,
  - complete set $\mathbf{Z_n}$ of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set $\mathbf{Z^*_n}$ of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Computation of ø(n)

- count number of residues to be excluded
- Special cases
  - for n = p with prime p **ø(p) = p-1**
  - for n = p·q (p, q prime) **ø(pq) =(p-1)·(q-1)**
    - **by p·q – 1 – (p – 1) – (q – 1)**
  - eg **ø(37) = 36**
    ø(21) = (3–1)·(7–1) = 2·6 = 12
- In general, need prime factorization,

$$\phi(n) = n\prod_{p|n}(1-\frac{1}{p})$$

where ***p*** runs over all primes dividing ***n***

Euler's Theorem

- $a^{ø(n)} = 1 \pmod{n}$
  - for **any** **a,n** where **gcd(a,n)=1**
  - a generalisation of Fermat's if n prime, i.e., **ø(n)=n-1**

- eg.

  a=**3**; ***n***=**10**; **ø(10)=4;**

  hence $3^{ø(10)} = 3^4 = 81 = 1 \pmod{10}$

  a=**2**; ***n***=**11**; **ø(11)=10;**

  hence $2^{ø(11)} = 2^{10} = 1024 = 1 \pmod{11}$

Proof:

- (the same line of reasoning as applied to Fermat's):

  Let $\mathbf{R = \{x_1, .., x_{ø(n)}\}}$
  integers ≤ n and relatively prime to n

  Let $\mathbf{S = \{x_1 \cdot a \bmod n, .., x_{ø(n)} \cdot a \bmod n\}}$

Chapter 8. Introduction to Number Theory

8.1 Prime Numbers

8.2 Fermat's and Euler's Theorems

Fermat's Theorem, Euler's Totient Function, Euler's Theorem

8.3 Testing for Primality

Miller-Rabin Algorithm, A Deterministic Primality Algorithm, Distribution of Primes

8.4 The Chinese Remainder Theorem

8.5 Discrete Logarithms

The Powers of an Integer Modulo n, Logarithms for Modular Arithmetic, Calculation of Discrete Logarithms

Appendix:

Square Root of 1, Modular Linear Equations, Subgroup, Factorization

Primality Testing for large primes

- traditionally, **sieve** using **trial division**
  - i.e. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternative to use statistical primality tests based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the properties
- a slower deterministic primality test

Simple Pseudoprimality Test

- If $a \in Z_n^+ = \{1,2,\ldots,n-1\}$ and $a^{n-1} \neq 1 \pmod n$ then n is not prime (i.e., composite)
  - Usually, only try a=2 (or a=3 in addition)
- Otherwise, we guess that n is a prime
  - Unfortunately, we may be wrong (pseudoprime)
  - Carmichael numbers: composite numbers that satisfy the Fermat's theorem
    - Very rare, the first three are 561, 1105, and 1729. There are only 255 of them less than 100,000,000

Properties of Primes

- Using properties to test if a number is prime
  - Fermat's Theorem (the first property)
- Theorem (Square root of 1, the second property)
  - If **p** is an odd prime and **e**≥ 1, then the equation $x^2 = 1 \pmod{p^e}$, in particular, $x^2 = 1 \pmod{p}$, has only two trivial solutions, namely $x = 1$, $x = -1$.

- For example, four square roots of 1 (mod 8), which are 1, 7, 3, 5 (i.e., +1, -1, +3, -3)
- because

$1^2 \bmod 8 = 1$

$7^2 \bmod 8 = 49 \bmod 8 = 1$

$3^2 \bmod 8 = 9 \bmod 8 = 1$

$5^2 \bmod 8 = 25 \bmod 8 = 1$

Therefore, we know 8 is not a prime

Miller Rabin Primality Test

- "Square Root of 1 Test" + "Fermat's theorem"
  - if $n>2$ is prime, $(n-1)=2^k q$ for some $k > 0$, q *is* odd;
  - **For the sequence:**
    $a^q, a^{2q}, a^{4q}, \ldots, a^{2^{k-1}q}, a^{2^k q} = a^{n-1} = 1 \pmod{n}$
  - each number in the sequence is the square of the previous and the last is 1, there must be +1 or -1 (i.e., **n-1**) when **(mod n)** before the last

Miller Rabin Primality Test

TEST (n) is:

1. Find integers k, q, $k > 0$, q odd, so that $(n-1) = 2^k q$;
2. Select a random integer a, $1 < a < n-1$;
3. **if** $a^q \bmod n = 1$ **then** return (“maybe prime");
4. **for** $j = 0$ **to** $k - 1$ **do**
5. **if** ($a^{2^j q} \bmod n = n-1$) **then** return(" maybe prime ");
6. return ("composite");

The probability of a pseudo-prime < ¼ (error rate)

Probabilistic Considerations

- if Test(**n**) returns
  - "composite", **n** is definitely not prime
  - "maybe prime", **n** is a pseudo-prime (error): $Pr < \frac{1}{4}$
- If repeating the test with several randomly chosen bases **a** for $0<a<n$ then the chance **n** is prime after s tests is
  - $Pr(n \text{ prime after } s \text{ tests}) = 1 - (\frac{1}{4})^{s}$
  - eg. for $s = 10$ this probability is > 0.99999

Prime Distribution

- Prime number theorem:

  $\pi(n)$: #of primes $\leq n$

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1$$

- prime number theorem states that primes occur roughly every (`ln n`) integers
- immediately ignore evens, only need test `ln(n)/2` numbers of size **n** to locate a prime
  - this is only the "average"
  - sometimes primes are close together
  - other times primes are quite far apart

Chapter 8. Introduction to Number Theory

8.1 Prime Numbers

8.2 Fermat's and Euler's Theorems

Fermat's Theorem, Euler's Totient Function, Euler's Theorem

8.3 Testing for Primality

Miller-Rabin Algorithm, A Deterministic Primality Algorithm, Distribution of Primes

8.4 The Chinese Remainder Theorem

8.5 Discrete Logarithms

The Powers of an Integer Modulo n, Logarithms for Modular Arithmetic, Calculation of Discrete Logarithms

Appendix:

Square Root of 1, Modular Linear Equations, Subgroup, Factorization

Chinese Remainder Theorem

The Importance of CRT

- RSA – correctness

- RSA – used to speed up modulo computations
  - if working modulo a product of numbers

    $\mathbf{M = m_1 m_2 .. m_k}$
  - CRT lets us work in each moduli $\mathbf{m_i}$ separately
    - computational cost is proportional to size,

The Historic Problem

- Around 100 AD, the Chinese mathematician Sun-Tse solved the problem of finding those integers that leave remainder 2, 3, and 2 when divided by 3, 5, and 7 respectively.

- One such solution is x = 23; all solutions are of the form 23+105k for arbitrary integers k.

Chinese Remainder Theorem

- Let $m_1, m_2, ..., m_k$ be pairwise relatively prime and $M = m_1 m_2 .. m_k$,
  any integer A in Z_M
  one-to-one corresponds to a k-tuple
  $(a_1, a_2, \ldots, a_k)$ whose elements a_i are in Z_{m_i},
  that is,

$$A \leftrightarrow (a_1, a_2, \ldots, a_k)$$

One-to-one Correspondence

- $\rightarrow$: from **A(mod M)** to **a_i(mod m_i)**
  Simply compute $a_i = A \bmod m_i$ separately
- $\leftarrow$: back to **A(mod M)** from **a_i(mod m_i)**
  compute c_i then A where $M_i = M/m_i$.

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{for } 1 \le i \le k$$

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i\right) \pmod{M}$$

$c_i = 0 \pmod{m_j}$ as M_i has a factor m_j if $i \neq j$

$c_i = M_i * (M_i^{-1} \bmod m_i) = 1 \pmod{m_i}$

What is A? if $A = 2 \pmod 5$, $A = 3 \pmod{13}$

- First, $a_1=2$, $m_1=5=M_2$, $a_2=3$, $m_2=13=M_1$
- $M_1^{-1} \bmod m_1 = 13^{-1} \bmod 5 = 2 \pmod 5$
- $M_2^{-1} \bmod m_2 = 5^{-1} \bmod 13 = 8 \pmod{13}$
- $c_1= M_1(M_1^{-1} \bmod m_1) =13 * 2 = 26$
- $c_2= M_2(M_2^{-1} \bmod m_2) =5 * 8 = 40$
- $A = a_1c_1 + a_2c_2 = 2*26 + 3*40 = 42 \pmod{65}$

Modular Reduction by CRT

- One-to-one correspondence
  - a system of equations
    modulo a set of pairwise relatively prime moduli
  - an equation modulo their product.

$$A \longleftrightarrow (a_1, a_2, \ldots, a_k)$$
$$B \longleftrightarrow (b_1, b_2, \ldots, b_k)$$

which enables modular reduction

$$(A + B) \bmod M \longleftrightarrow ((a_1 + b_1) \bmod m_1, \ldots, (a_k + b_k) \bmod m_k)$$
$$(A - B) \bmod M \longleftrightarrow ((a_1 - b_1) \bmod m_1, \ldots, (a_k - b_k) \bmod m_k)$$
$$(A \times B) \bmod M \longleftrightarrow ((a_1 \times b_1) \bmod m_1, \ldots, (a_k \times b_k) \bmod m_k)$$

Chapter 8. Introduction to Number Theory

8.1 Prime Numbers

8.2 Fermat's and Euler's Theorems

Fermat's Theorem, Euler's Totient Function, Euler's Theorem

8.3 Testing for Primality

Miller-Rabin Algorithm, A Deterministic Primality Algorithm, Distribution of Primes

8.4 The Chinese Remainder Theorem

8.5 Discrete Logarithms

The Powers of an Integer Modulo n, Logarithms for Modular Arithmetic, Calculation of Discrete Logarithms

Appendix:

Square Root of 1, Modular Linear Equations, Subgroup, Factorization

Discrete logarithms

- inverse problem to exponentiation in Z^{+}_{p} is to find the **discrete logarithm** of a number in Z^{+}_{p}
- fundamental to public-key algorithms
  - Diffie-Hellman key exchange
  - the digital signature algorithm (DSA)
- share the properties of normal logarithms
  - The logarithm of a number is the power to which some positive base (except 1) must be raised in order to equal that number

Discrete logarithms

- If working with modulo arithmetic and the base is a generator, an integral discrete logarithm exists
  - Given **g** is a generator (primitive root) of $\mathbf{Z}^{+}_{p}$ with a prime p, find **x** such that $\mathbf{y = g^{x} \ (mod\ p)}$ for $\mathbf{y} \in \mathbf{Z}^{+}_{p}$
  - this is written as $\mathbf{x = dlog_{g,p}(y)}$
    - the discrete logarithm of **y** for the base **g**, **mod p**
- If the base is not a generator, dlog may not exist.

  $x = dlog_{3,13}(4)$ has no answer while

  $x = dlog_{2,13}(3) = 4$ as 3 is not a generator of $\mathbf{Z}^{+}_{13}$ and 2 is

Hard problem

- while exponentiation is relatively easy, finding discrete logarithms is generally **hard**, in fact is as hard as factoring a number
- problem that is "easy" one way, "hard" the other
  - easy: raising a number to a power
  - hard: finding what power a number is raised to giving the desired answer
- Problems with such asymmetry are rare, but are of critical usefulness in modern cryptography

Refresh: Generators (Primitive Roots) for Z^+_p

- For a group G, an element **g** is a generator of G if every element in G is a power of **g**
- for $\mathbf{g} \in Z^+_p = \{1,..,p\text{-}1\}$ with prime p, $\mathbf{g}^{p-1} = 1$ (mod p) by Fermat's theorem. If m = p-1 is the smallest m such that $\mathbf{g}^m = 1$ (mod p), then **g** is a generator of Z^+_p
- 2 is not a generator of Z^+_7 as $2^3=1$ (mod 7) while 3 is as successive powers of 3 are 3,2,6,4,5,1 which form Z^+_7

Finding generator(primitive roots) for $\mathbf{Z}^{+}_{p}$ efficiently

- Determine distinct prime factors of **p-1**, $p_1,..,p_k$.
- Select a random number g in $\mathbf{Z}^{+}_{p}$,

$$g^{(p-1)/p_i} \bmod p \quad for\ i = 1,..,k$$

- If the above k results are all different from 1, then g is a primitive root (generator)
  - The reason is that the size (order) of a subgroup is a factor of the size (order) of the group (Lagrange)

Avoiding prime factorization in finding a prime with generators

- In order to construct the group $Z^+_p = \{1,2,\ldots,p-1\}$ with p being a prime,
- we can run Miller-Rabin algorithm to randomly select k primes, $p_1,\ldots,p_k$, such that $p = p_1 * p_2 * \ldots * p_k + 1$ is a prime