Cryptography and Network Security Chapter 8

Fifth Edition

by William Stallings

Lecture slides by Lawrie Brown

(with edits by RHB)

Chapter 8 – Introduction to Number Theory

The Devil said to Daniel Webster: "Set me a task I can't carry out, and I'll give you anything in the world you ask for."

Daniel Webster: "Fair enough. Prove that for n greater than 2, the equation $a^n + b^n = c^n$ has no non-trivial solution in the integers."

They agreed on a three-day period for the labor, and the Devil disappeared.

At the end of three days, the Devil presented himself, haggard, jumpy, biting his lip. Daniel Webster said to him, "Well, how did you do at my task? Did you prove the theorem?'

"Eh? No . . . no, I haven't proved it."

"Then I can have whatever I ask for? Money? The Presidency?'

"What? Oh, that—of course. But listen! If we could just prove the following two lemmas—"

***—The Mathematical Magpie*, Clifton Fadiman**

Outline

- will consider:
  - prime numbers
  - Fermat's and Euler's Theorems & ø(n)
  - Primality Testing
  - Chinese Remainder Theorem
  - Primitive Roots & Discrete Logarithms

Prime Numbers

- prime numbers only have divisors of 1 and self
  - 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
- prime numbers are central to number theory
- 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: `n = a x b x c`
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number `n` is when its written as a product of primes
  - eg. $91 = 7 \times 13$; $3600 = 2^4 \times 3^2 \times 5^2$

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

Relatively Prime Numbers & GCD

- two numbers `a,b` are **relatively prime (coprime)** if they have **no common divisors** apart from `1`
  - eg. `8` and `15` are relatively prime since factors of `8` are `1,2,4,8` and of `15` are `1,3,5,15` and `1` is the only common factor
- conversely, can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300 = 2^1 \times 3^1 \times 5^2$; $18 = 2^1 \times 3^2$ hence $\mathrm{GCD}(18,300) = 2^1 \times 3^1 \times 5^0 = 6$

Fermat's Theorem

- $a^{p-1} = 1 \pmod p$
  - where `p` is prime and $\mathrm{GCD}(a,p) = 1$
- also known as Fermat's Little Theorem
- also have: $a^p = a \pmod p$
- useful in public key crypto and primality testing

Fermat's Theorem ... sketch proof.

Consider $\{1, 2, \ldots, p-1\}$ with p prime.

Consider $\{1 \times a \bmod p, 2 \times a \bmod p, \ldots, (p-1) \times a \bmod p\}$. This permutes $\{1, 2, \ldots, p-1\}$ since all of $\{1, 2, \ldots, p-1\}$ coprime to p.

In both sets, multiply all the elements together $\bmod p$.

So $a^{p-1} \times (p-1)! \bmod p = (p-1)! \bmod p$

Since $(p-1)!$ is coprime to p, we can cancel it, getting Fermat's Theorem:

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

Euler Totient Function ø(n)

- when doing arithmetic modulo n
- **complete set of residues** is: 0..n-1
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n = 10, complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Euler Totient Function ø(n)

- to compute ø(n) need to count number of residues to be excluded
- in general need prime factorization, but
  - for p (p prime) ø(p)=p-1
  - for p.q (p,q prime) ø(p.q)=(p-1)x(q-1)
- eg.

  ø(37) = 36

  ø(21) = (3-1)x(7-1) = 2x6 = 12

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{ø(n)} = 1 \pmod n$
  - for any a,n where GCD(a,n) = 1
- eg.

  a = 3 ; n = 10 ; ø(10) = 4;

  hence $3^4 = 81 = 1 \bmod 10$

  a = 2 ; n = 11 ; ø(11) = 10;

  hence $2^{10} = 1024 = 1 \bmod 11$
- also have: $a^{ø(n)+1} = a \pmod n$

Euler's Totient Function $\phi(n)$ ... for $n = p.q$

Euler's Totient Function $\phi(n)$ is defined to be the number of integers between 1 and n which are coprime to n. (Obviously, the biggest of them is no greater than $n-1$.)

For a prime p, $\phi(p) = p-1$

For a product of two primes, p and q, the integers between 1 and $p.q$ which are **NOT** coprime to $p.q$ are:

1. multiples of p ($q-1$ of them),
2. multiples of q ($p-1$ of them).

So: $\phi(p.q) = (p.q-1) - [(q-1) + (p-1)] = pq - p - q + 1$

i.e.: $\phi(pq) = (p-1).(q-1) = \phi(p).\phi(q)$

Euler's Theorem ... sketch proof.

Suppose a and n are coprime.

Consider $\{1, x_2, \ldots, x_{\phi(n)}\}$ with each x_i coprime to n.

Consider $\{1\times a \bmod n, x_2\times a \bmod n, \ldots, x_{\phi(n)}\times a \bmod n\}$. This permutes $\{1, x_2, \ldots, x_{\phi(n)}\}$ since $\{1, x_2, \ldots, x_{\phi(n)}\}$ all coprime to n.

In both sets, multiply all the elements together $\bmod n$.

So $a^{\phi(n)}\times\prod_i x_i \bmod n = \prod_i x_i \bmod n$

Since $\prod_i x_i$ is coprime to n, we cancel it, getting Euler's Theorem:

$$a^{\phi(n)} = 1 \pmod n$$

Primality Testing

- often need to find large prime numbers
- traditionally **sieve** using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternatively can 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 property
- can use a slower deterministic primality test

Miller Rabin Algorithm

- a test based on prime properties that result from Fermat's Theorem
- algorithm is:

  `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`$^{\text{q}}$ `mod n = 1` **then** return ("inconclusive");
  4. **for** `j = 0` **to** k − 1 **do**
     5. **if** (`a`$^{2^{\text{j}}\text{q}}$ `mod n = n - 1`)
        then return("inconclusive")
  6. return ("composite")

Miller-Rabin ... rationale.

Testing a number $n = 2^k q + 1$ (k is maximal, q is odd).
Pick $1 < a < n - 1$.

IF n is prime, **THEN** $a^{n-1} = 1 \pmod n$ i.e., $a^{2^k q} = 1 \bmod n$
(by Fermat's Theorem). So a **proper suffix** of the sequence:

$$a^q \bmod n, a^{2q} \bmod n, a^{2^2 q} \bmod n, \ldots, a^{2^k q} \bmod n,$$

must be $1 \bmod n$.

IF the first is $1 \bmod n$ **THEN** the rest are too. **IF** a later element is $1 \bmod n$ **THEN** then its predecessor is $-1 \bmod n \equiv n - 1 \bmod n$

So Miller-Rabin says **MAYBE** if either of these is seen.

Probabilistic Considerations

- if Miller-Rabin returns "composite" the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is < $^1/_4$
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1 - (1/4)^t$
  - eg. for t=10 this probability is > 0.99999
- could then use the deterministic AKS test

Prime Distribution

- prime number theorem states that primes occur roughly every (`ln n`) integers
- but can immediately ignore evens
- so in practice need only test `0.5 ln(n)` numbers of size `n` to locate a prime
  - note this is only the "average"
  - sometimes primes are close together

    eg. 1,000,000,000,061 and 1,000,000,000,063 both prime
  - other times are quite far apart

    eg. (1001!+2), (1001!+3) … (1001!+1001) all composite

Chinese Remainder Theorem

- used to speed up modulo computations
- if working modulo a product of numbers
  - eg. `mod M` = $m_1 m_2 .. m_k$
- Chinese Remainder theorem lets us work in each modulus m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus `M`

Chinese Remainder Theorem.

Have k mutually coprime numbers $m_1, m_2, \ldots m_k$.
Let $M = m_1.m_2. \ldots .m_k$.

Then, provided $0 \le A \le M$, the number A is uniquely determined by the k-tuple $(a_1, a_2, \ldots, a_k) = (A \bmod m_1, A \bmod m_2, \ldots, A \bmod m_k)$.

Arithmetic operations done co-ordinate-wise:

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

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

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

N.B. Division doesn't work!

Chinese Remainder Theorem

given the $a_i = A \bmod m_i$, to compute $A \pmod M$

- first compute all $a_i = A \bmod m_i$ separately
- determine constants c_i below, where $M_i = M/m_i$
- then combine results to get answer A using:

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

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

Chinese Remainder Theorem Inversion Formula.

We can check that $A = (\sum_1^k a_i c_i) \bmod M$ has the expected property, namely that $A \bmod m_i = a_i$.

♠ = $A \bmod m_i = (((\sum_{j=1}^k a_j c_j) \bmod M) \bmod m_i) = (\sum_{j=1}^k a_j c_j \bmod m_i)$

[because m_i is a factor of M]. So

♠ = $(\sum_{j=1}^k a_j [M_j \times (M_j^{-1} \bmod m_j)] \bmod m_i)$ = ♠♠

Now there are two cases.

$i = j$ and

$i \neq j$

Case $i = j$: then ♠♠ = $(\sum_{j=1}^k a_j [M_j \times (M_j^{-1} \bmod m_j)] \bmod m_i) =$
$(a_i \bmod m_i) \times [(M_i \bmod m_i) \times (M_i^{-1} \bmod m_i)] = (a_i \bmod m_i)$
[because $((M_i \bmod m_i) \times (M_i^{-1} \bmod m_i) \bmod m_i) = 1$]

Case $i \neq j$: then ♠♠ = $(\sum_{j=1}^k a_j [M_j \times (M_j^{-1} \bmod m_j)] \bmod m_i) =$
$(a_j \bmod m_i) \times [(M_j \bmod m_i) \times (((M_j^{-1} \bmod m_j) \bmod m_i)] \bmod m_i) = 0$
[because m_i is a factor of M_j, so $(M_j \bmod m_i) = 0$]

So the sum of the terms is just $(a_i \bmod m_i)$ as required.

Primitive Roots

- by Euler's theorem we have $a^{\varnothing(n)} \bmod n = 1$
- consider $a^k \pmod n$ with $\mathrm{GCD}(a, n) = 1$, for various k, and look for m, where $a^m = 1 \pmod n$
  - works for $m = \varnothing(n)$ but may work for a smaller m too
  - once the powers of a reach m, the cycle will repeat
- if smallest value is $m = \varnothing(n)$ then a is called a **primitive root**
- if we do this for $n = p$ where p is prime, then successive powers of a "generate" the multiplicative group $\bmod p$
- these are useful but relatively hard to find

Powers mod 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Discrete Logarithms

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number `b` modulo `p`
- that is to find `i` such that $b = a^i \pmod p$
- this is written as $i = \text{dlog}_a\, b \pmod p$
- if `a` is a primitive root `mod p` then dlog_a always exists, otherwise it may not, eg.
  - $x = \log_3 4 \bmod 13$ has no answer
  - $x = \log_3 3 \bmod 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem (which is good for cryptography, of course)

Discrete Logarithms mod 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9