Lecture 11: Prime Numbers And Discrete Logarithms

Lecture Notes on "Computer and Network Security"

by Avi Kak (kak@purdue.edu)

February 14, 2017
1:53pm

Goals:

- Primality Testing
- Fermat's Little Theorem
- The Totient of a Number
- The Miller-Rabin Probabilistic Algorithm for Testing for Primality
- **Python and Perl Implementations for the Miller-Rabin Primality Test**
- The AKS Deterministic Algorithm for Testing for Primality
- Chinese Remainder Theorem for Modular Arithmetic with Large Composite Moduli
- Discrete Logarithms

CONTENTS

11.1: PRIME NUMBERS

- **Prime numbers are extremely important to computer security.** As you will see in the next lecture, public-key cryptography would not be possible without prime numbers.

- As stated in Lecture 12, an important concern in public-key cryptography is to test a randomly selected integer for its **primality**. That is, we first generate a random number and then try to figure out whether it is prime.

- An integer is prime if it has exactly two **distinct** divisors, the integer 1 and itself. That makes the integer 2 the **first prime**.

- We will also be very interested in two integers being **relatively prime** to each other. Such integers are also called **coprimes**. Two integers m and n are coprimes if and only if their Greatest Common Divisor is equal to 1. That is if $gcd(m, n) = 1$. Therefore, whereas 4 and 9 are coprimes, 6 and 9 are not. [See Lecture 5 for *gcd*.]

- Much of the discussion in this lecture uses the notion of **co-primes**, as defined above. The same concept used in earlier lectures was referred to as **relatively prime**. **But as mentioned above, the two mean the same thing.**

- Obviously, the number 1 is **coprime** to every integer.

11.2: FERMAT'S LITTLE THEOREM

- Our main concern in this lecture is with testing a randomly generated integer for its primality. As you will see in Section 11.5, the test that is computationally efficient is based directly on Fermat's Little Theorem. [This theorem also plays an important role in the derivation of the famous RSA algorithm for public-key cryptography that is presented in Section 12.2.3 of Lecture 12. Yet another application of this theorem will be in the speedup of the modular exponentiation algorithm that is presented in Section 12.5 of Lecture 12.]

- The theorem states that if p is a **prime number**, then for **every integer** a the following must be true

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

Another way of saying the same thing is that for any prime p and any integer a, $a^p - a$ will always be divisible by p. [Review the notation of modular arithmetic in Lecture 5 to fully understand what this theorem is saying. As stated in that lecture, $a^p \equiv a \ (mod\ p)$ means that $a^p \ mod\ p = a \ mod\ p$. For example, $8^3 \equiv 8 \ (mod\ 3)$ since $8^3 \ mod\ 3 = 2$ and, at the same time, $8 \ mod\ 3 = 2$.]