CHAPTER 11

Mathematical Background

Number Theory

Prime Numbers:

- A prime number is an integer greater than 1 whose only factors are 1 and itself.
- Cryptography, especially public key cryptography, uses large primes (512 bits and even larger).

Greatest Common Divisor (gcd):

- Two numbers are relatively prime when they share no factors in common other than 1.
- In other words, if the greatest common divisor of a and n is equal to 1. This is written as:

$$\gcd(a, n)=1$$

The number 15 and 38 are relatively prime, 15 and 55 are not.

Greatest Common Divisor

- Knuth describes the algorithm and some modern modifications in C.

```
Int gcd (int x, int y)
{
      int g;
      If (x<0) x= -x;
      If (y<0) y= -y;
      If (x+y==0) Error;
      g=y;
      While (x>0)
      {
                  g=x;
                  x=y%x;
                  y=g;
      }
      return(g);
}
```

Inverses Modulo a Number

- The multiplicative inverse of 3 is 1/3, because 3*1/3=1.
- In modulo world, the problem is more complicated.

$$4*x \equiv 1 \pmod 7.$$

This equation is equivalent to finding an x and k such that

$$4x=7k+1.$$ where both x and k are integers.

The general problem is finding an x such that

$$1= (a*x) \bmod n.$$

This is written as

$$a^{-1} \equiv x \pmod n.$$

- The inverse modulo problem is a lot more difficult to solve.
- Sometimes it has solutions, sometimes not. For example, the inverse of 5, modulo 14, is 3. On the other hand, 2 has no inverse modulo 14.

Prime Number Generation

- Public key algorithms need prime numbers.
- Let us answers some obvious questions:

1. If everyone need a different prime numbers, won't we run out?

   Answer: No. In fact, there are approximately 10^{151} primes 512 bits in length or less. For numbers near n, the probability that a random number is prime is approximately one in (ln n). So the total number of primes less than n is n/(ln n).

2. What if two people accidentally pick the same prime numbers?

   Answer: It won't happen. With over 10^{151} prime numbers to choose from, the odds of that happening are significantly less than odds of your computer spontaneously combusting at the exact moment you win the lottery.

3. If someone creates a database of all primes, won't he be able to use that database to break public key?

   Yes, but he can't do it. If you store 1GB of information on a drive weighing 1 gram, then a list of just the 512 bit primes would weigh so much that it would exceed the chandrasekhar limit and collapse into a black hole.

Prime Number Generation

- The wrong way to find primes is to generate random numbers and then try to factor them.
- The right way is to generate random numbers and test if they are prime.

Lehmann: A simple test was developed independently by lehmann. Here it test if p is prime.

1. Choose a random number a less than less than p.
2. Calculate $a^{(p-1)/2}$ mod p.
3. If $a^{(p-1)/2} \neq 1$ or -1 (mod p), then p is definitely not prime.
4. If $a^{(p-1)/2} \equiv 1$ or -1 (mod p), then the likelihood that p is not prime is no more than 50 percent.

Repeat this test t times. If the calculation equals 1 or -1, but does not always equal 1, then p is probably prime with an error rate of 1 in 2^t.

Robin-Miller

- This algorithm is very easy and used by everyone.
- Choose a random number p to test. Calculate b, where b is the number of times 2 divides p-1 (i.e. 2^b is the largest power of 2 that divides p-1). Then calculate m such that $p=1+2^b*m$.

1. Choose a random number , a, such that a is less than p.
2. Set j=0 and set $z=a^m \bmod p$.
3. If z=1, or z=p-1, then p passes the test and may be prime.
4. If j>0 and z=1, then p is not prime.
5. Set j=j+1. If j<b and $j \neq p-1$, set $z=z^2 \bmod p$ and go back to step[4]. If z=p-1, then p passes the test and may be prime.
6. If j=b and $z \neq p-1$, then p is not prime.

Practical Considerations

- In real world implementations, prime generation goes quickly.
  1. Generate a random n-bit number, p.
  2. Set the high order and low order bit to 1. (The high order bit ensures that the prime is of the required length and the low order bit ensures that it is odd).
  3. Check to make sure p is not divisible by any small primes: 3, 5, 7, 11, and so on. Many implementations test p for divisibility by all primes less than 256. The most efficient is to test for divisibility all primes less than 2000.
  4. Perform the Rabin –Miller test for some random a. If p passes, generate another random a and go through the test again. Choose a small value of a to make the calculations go quicker. If p fails one of the tests, generate another p and try again.