Discrete Structures	**Problem Set 5**
CS 280 Spring 2005	**Due Wednesday, March 2, 2005**

Reading: Rosen Sections 2.5. recall our first prelim time: Tuesday, March 8, 7:30-9pm. More on the prelim, and practice questions will be available on Wednesday, March 2nd.

(1) A standard way to generate pseudo-random numbers in computers is by a linear congruence generator. (Pseudo-random means that the numbers should behave randomly enough, but we can still generate them somehow without tossing coins.) Here is how linear congruence generators work. You select a prime p, and two natural numbers a and b, and start by any natural number x_0. The sequence generated will be $x_1, x_2, \ldots$. Where we define $x_{n+1} = (ax_n + b) \bmod p$ for all $n \geq 0$. For example, the prime $p = 2^{31} - 1$ is widely used in generators. The good news advertised by the manufacturers is that with good choice of a and b there are $p - 1$ different numbers generated before repetition begins. You do not have to prove this.

However, ideally good random numbers should be unpredictable. What this means is this: even if we know what $x_0, x_1, \ldots, x_k$ is for some value k, it should be hard or impossible to tell what x_{k+1} will be. Show that this generator does not have this property if the prime p is known.

More precisely, assume a prime p is given, but we do not know either b or a. How many entries in the sequence do we need to know to be able to determine the next entry? Prove that your answer is correct.

(2) We define $\phi(n)$ for a natural number n as the number of integers $1 \leq m < n$ that are relative prime to n. For example $\phi(5) = 4$, and $\phi(6) = 2$ (as only 1 and 5 are relatively prime). Note that $\phi(p) = p - 1$ for all primes p. You do not have to prove this (do you see why?)

(a) Let p be a prime, and k a natural number. What is $\phi(p^k)$? Prove your answer.

(b) Let p and q be two different primes. What is $\phi(pq)$? Prove your answer.

(3) In class I mentioned that we need a fast method for exponentiation modulo n. The method `Exponentiate`(a, x, n) computes $a^x \bmod n$, and works as follow.

```
Exponentiate(a, x, n)
If x = 1 output a modulo n
Else if x odd then
   Let b=Exponentiate(a, x − 1, n)
   compute a · b modulo n
   Output the result
Else if x even
   Compute b = Exponentiate(a, x/2, n)
   Output b^2  modulo n
```

Let $f(x)$ be the number of multiplications performed by this code for with input x. So $f(1) = 0$ as with $x = 0$ we perform no multiplications, $f(2) = 1$ as with $x = 2$ the code involves a single squaring operations. Prove that $f(x) \leq 2 \log_2 x$ for all $x \geq 1$.

Introdcution to questions 4-5. Recall that the security of the RSA code depends on the hardness of factoring numbers. So the RSA code uses a modulus $n = pq$ precursive ublishing n and the encryption exponent e, and keeping as secret p, q and the decryption exponent d. If you somehow figure out how to factor n as a product of primes $n = pq$, then given e, we can get the decryption exponent d by solving the modulo equation

$$ed \equiv 1 \bmod (p-1)(q-1)$$

We have seen how to do this using $O(\log n)$ divisions by Euclid's algorithm if $(p-1)(q-1)$ is given, however we claimed that it is hard to do just knowing n. Of course, we can always factor by brute force (say trying all numbers below $\sqrt{n}$ to see which of them divides n), but this will involve $O(\sqrt{n})$ divisions. We call an algorithm *polynomial* if the number of basic operations it performs (such as divisions, multiplications, etc), is a polynomial in the binary size of the number, that is, it is bounded by $O(\log^c n)$ for some constant c. Now the question is if there exists such a polynomial time algorithm to decode messages, given the encoded message C and the public information n, and e. However, to make sure it is hard to decode, we need to be careful with how we select primes. The next two questions explore this problem.

(4) Suppose your company needs k separate public key systems. They decide to use RSA, and do now they need to generate $2k$ primes p_i and q_i for $i = 1, 2, .., k$, and use modulus $n_i = p_i q_i$ as the modulus for the encryption system i.

However, they it takes a while finding so many primes, so they propose that maybe it would be enough to use $\approx \sqrt{k}$ of them. Here is how they are thinking. Suppose you find ℓ primes $p_1, p_2, \ldots, p_\ell$. We could take each pair of these primes, and produce their product $n_{ij} = p_i p_j$. From ℓ primes, this way we create $\ell(\ell-1)/2$ pairs. Now $\ell \approx \sqrt{k}$, so for example, if $\ell = 10$ then all possible pairs would give us $k = 45$ numbers that are products of pairs of primes. Quite a bit of saving from the $2k = 90$ that we would have to generate by the original method. Now the select an ecryption, descryption pairs (e, d) for each modulus b, and when they send an encrypted message, they need to know which modulus n was used.

Show that this method of generating RSA encryption schemes is insecure, by giving a polynomial time algorithm that given the public information of the $k = \ell(\ell-1)/2$ different modulus, and the k encryption keys, finds the descryption keys.

(5) Your friend wants advice for generating a good modulus for RSA coding. So he noticed that if one of the two two primes is small, like $n = pq$ where $p = 2, 3, 5, etc$ then finding the prime factorization gets pretty easy: using bruce force to find the factorization

of n one finds the small prime p pretty fast, and then we get $q = n/p$, and we know that q must be prime, as assumed by this system. To make things better, your friend decided to use twin primes, $q = p + 2$ for defining the modulus.

Show that this is a bad idea by giving a polynomial time algorithm that n that is known to be a product of two twin primes, recovers the primes p and q.

(6) After solving problems 4 and 5 you are worried that the RSA system may not be secure enough, and want to invent a crypto system that uses a product of 3 primes. So assume $n = pqr$ where p, q and r are distinct primes. Your system will be analogous to the reglar RSA system. The messages m will be integers $0 \leq m < n$, will use an exponent e to encrypt messages, so the encrypted form of message m will be $c \equiv m^e \bmod n$. Show that if you select e such that $gcd(e, (p-1)(q-1)(r-1)) = 1$, and you know the primes p, q and r, then you can find a decryption key d such that $c^d \equiv m \bmod n$, for all messages m. Show how to find this decryption exponent, and prove that your method works.

(7 optional) To find the description key for RSA, we may want to factor $n = pq$ as the product of two primes to get $(p-1)(q-1)$, which we can then use to find the decryption key d. Show that factoring n as pq is equivalent to finding $(p-1)(q-1)$. More formally, show that given a two integers n and m such that n is the product of two primes pq and $m = (p-1)(q-1)$ you can recover p and q in polynomial time.