CS 3510 - Honors Algorithms
Homework 7
Assigned April 11
Due Tuesday, April 18

1. The *least common multiple* $lcm(a_1, \ldots, a_n)$ of a set of n integers $a_1, a_2, \ldots, a_n$ is the least non-negative integer that is a multiple of each of the a_i. Show how to compute $lcm(a_1, \ldots, a_n)$ efficiently using the two argument *gcd* operation as a subroutine. Be sure to argue correctness and also give the complexity in terms of the number of calls to *gcd*.

2. Let $p = 17$, $q = 19$ and $e = 25$. Find the public key (e, N) and the associated private key (d, N) of an RSA scheme. Encrypt a message $M < N$ and then decrypt the encoded message.

3. a) Recall that in the RSA public-key scheme, a user publishes his public key $P = (N, e)$ and keeps private his secret key d. In a digital signature algorithm there are two algorithms *Sign* and *Verify*. The algorithm *Sign* takes a "form" M and, using a user's secret key, produces a signature $\sigma(M)$. *Verify* takes a user's public key, a signature $\sigma(M)$ purported to be theirs, and the form M, and retur ns "true" if $\sigma(M)$ could have been created by the user "signing" the form M, a nd returns "false" otherwise.

   a) Why would we want digital signatures? (One paragraph at most!)

   b) An RSA signature consists of $Sign(d, M) = M^d (\text{mod N})$, where d is an RSA secret ke y and N is the first half of the public key. Show that anyone who knows the public key (N, e) can perform $Verify((N, e), M^d, M)$ (i.e, they can check that a signature reall y was created by the private key of the intended user. Notice that someone pretending to be you could not create this signature without your help. Give an implementation and prove it s correctness.

4. This question is intended to show that you have to be careful using signatures an d encryption. Suppose that someone has sent you an RSA-encrypted message $X = M^e (\text{mod N})$ where M is the original message and (N, e) is your public key. A hacker intercepts X and th en computes $Y = r^e X (\text{mod N})$ where r is a random number and e is your public key. Th e hacker then asks you to sign form Y claiming it is an encrypted petition for your fav orite cause. You sign Y with RSA, as described by the last question. Show that the hacker can easily recover M from the signature you returned to them.

5. In class we saw a randomized primality test that distinguished { primes and Carmichael numbers} from {ordinary composites}. The goal now is to modify the algorithm so that it gives a randomized test for distinguishing primes from Carmichael numbers.

For an integer s, we can write $s-1$ in the form $2^t u$, where u is odd. Instead of simply computing a^{s-1} (mod s), as before, we evaluate this in steps. First we compute a^u (mod s), and then we continually square (t times). We know that if $a^{s-1} \neq 1$ (mod s) then s is composite. Suppose now that $a^{s-1} = 1$ (mod s). Somewhere along the way, while computing a^u (mod s), $\mathrm{a}^{2\mathrm{u}}$ (mod s), $\ldots a^{2^t u}$ (mod s) we ran into a 1 for the first time. If $a^u \neq 1$ (mod s), t hen look at the last time in our list that we have $a^{2^i u} \neq 1$ (mod s). If that value is not -1 (mod s), then we have found a non-trivial square root of 1 modulo s (i.e., a number that is not 1 or -1 (mod s) but whose square is 1).

a) Prove that a non-trivial square root of s can only exist if s is composite.

b) Show that if we combine this test with the Fermat test from class, then at least 3/4 of the possible values of a between 1 and $s-1$ will uncloak a composite s, even if it is a Carmichael number.

c) Explain how to use these facts to design a randomized primality test that is correct for all primes and correct with very high probability for all composites.