**Crypto Assign 5**

In this assignment we demonstrate the use of c = 3 as an encoding exponent in RSA. It is quite OK to do this since c is part of the public key anyway. We do need to be careful about one thing, however. The encoding exponent c must be coprime to C(n) so that the decoding exponent d ≡ c-1 (mod C(pq)) will be defined. This is a serious problem, because 1/3 of the primes p will satisfy

gcd(p – 1, 3) > 1. It’s simply a question of whether 3|(p – 1). So only 2/3 of the primes p will be suitable, and the prime search will be 3/2 as long, or 50% longer. The same is true for q of course.

Normally the encoding exponent c is roughly the size of n, and the probability that gcd(c, C(n)) > 1 is so incredibly small that we don’t even bother checking for it.

**Why use c = 3.**

If you are using a mobile phone to compute E ≡ Mc (mod n), the computation M3 (mod n) is less of a strain on its computing power than when c is a 300-digit number. We assume that there will be a full size computer at the receiving end, so the computation of M ≡ Ed (mod n) will be practical, even though d is likely to be a 300-digit number.

**1)** Find the smallest 150-digit primes p > R and q > S, where R and S are the 150-digit random numbers given below. All random numbers were generated for us on the website random.org.

R =

1783641067 3402257991 5357141142 3726990275 6700426613

7223687186 8435340097 7239452778 6774481895 9523570165

6778631793 3763105630 1597566295 4599897346 7529932163.

S =

7485273427 7911450910 8012294258 7999788736 1679312489

7195999683 4395315492 5671989910 1076581473 2692898441

7698081268 3605133154 8473067343 5936330595 5569274099.

**a)** Apply the (base 2) Fermat test to R, R+2, R+4,...until a probable prime p is found such that **3 does not divide p-1.** (Watch out, 3 will divide 1/3 of the numbers p-1.)

**b)** Same instructions for q. And 3 must not divide q-1.

**c)** Apply Rabin's test to p and q using the 4 "random" numbers r = 2, 3, 5, 7. If p fails, do everything over again starting at p+2. Similarly for q.

**2) a)** Compute n = pq and C(n) = LCM[p-1,q-1].

[Recall that LCM[u, v] = uv/gcd(u, v).]

You will set up an RSA system using the coding exponent c = 3.

**b)** Find the unique decoding exponent d = 3-1 (mod C(n)) = inverse(3,C(n)). Note that 3 does not divide C(n), so inverse(3,C(n)) will exist. The C++ code for inverse(u, v) is in the notes for your convenience.

**3)** Use the following "message" M. In a real world application M might consist of several AES keys concatenated together, and these would be represented in base 2k for some k, but base ten is more convenient for a homework assignment. We have made M a random 298-digit number, somewhat smaller than 300-digits, because it needs to be smaller than n = pq. Otherwise the decryption of E will be a different number M’ < M that is congruent to M modulo n. It is good practice when using RSA to reduce the message M modulo n first as a precaution. This is possible because n is part of the public key.

M =

7968676964 9407886671 8569964993 1393523249

3052895139 7458433313 6676341463 5656816617

5388440824 3356921203 7566144161 9811754277

3426855898 5843321177 7823851893 7443311869

4183224778 4261454402 4955562101 4882440899

6830393381 8924421750 6974674764 1554014204

1425360265 5058643390 6143337257 7700188914

8469682239 35567188

**a)** Compute E = M3 (mod n).

**b)** Compute Ed (mod n) and verify that it is M.