Cryptology
Assignment 3

Due 10:15 Friday, April 14, 2000

Write a program which chooses a public-private key pair for Rabin's cryptosystem, encrypts with the chosen key pair and decrypts with the chosen key pair. Use the definition given in class, where encryption is simply squaring modulo a composite number N. Your program should not restrict the prime factors of N to be congruent to 3 modulo 4, so you will have to be able to take square roots modulo any prime, even those congruent to 1 modulo 4. In order to determine which of the four possible square roots you get is the correct one, you may choose any arbitrary format you like for legitimate inputs. One possibility is to require that all bits with even index are zero, so the bits with odd index contain the information; an easier possibility might be to require that some subset of the high order bits all be zero, maybe 1/8 of the bits. Your program should choose the keys, encrypt a message using the public key, and decrypt it using the private key. (It is not necessary that your messages come from character input; they can just be numbers if you find that easier.) These should be three separate functions in your program, so that it would be easy to separate them into three separate programs.

To deal with the long numbers necessary, you may use the language C with the Large Integer Package due to Paul Leyland and Arjen Lenstra. You can get it via anonymous ftp from

`ftp.ox.ac.uk:/pub/math/freelip/freelip_1.1.tar.gz`.

After you `gunzip freelip_1.1.tar.gz` and `tar -xvf freelip_1.1.tar`, there will be several files, including `lipdoc.ps`, which contains the documentation. You may use any of the modular arithmetic routines (note that squaring is faster than multiplication, so you should do that when appropriate) and the randomization routines (we won't worry about their shortfalls in terms of cryptographic security). You may also use the routines `zchirem` and `zrandomprime`, described on pages 22 and 27 of `libdoc.ps`, respectively, for computing using the Chinese Remainder Theorem and for finding random primes. (If there are routines for finding square roots, please do not use them.)

If you prefer using Java, there is a class called BigInteger which should be efficient and easy to use. There is documentation at

`http://www.imada.sdu.dk/Technical/Manpages/jdk1.1.6/docs/api/java.math.BigInteger.html`

You may use any of the standard routines provided there, including the routines for generating random primes and computing modular inverses. There does not appear to be a routine for computing with the Chinese Remainder Theorem, but the hardest part of that is finding inverses.

A description of the Las Vegas algorithm I presented in the lectures for finding square roots modulo primes can be found in *Algorithmics: Theory and Practice*, by Brassard and Bratley, which is on reserve in the library.

Please turn in your program and some output, including the public and private keys, plus the input and output for the encryption and decryption. You should write a brief report, explaining how your program should be used, discussing how long your program took to run, and discussing the problem that the decryption is not unique. What confidence level did you choose for primality checking? (Choose primes which are about 512 bits long.)

Your report should also mention any major functions you used, such as one the one to do the symbolic exponentiation. You should also send me your program and any extra files via e-mail (they can just be attachments in `pine`).