

CIS 400/628  
Homework #3 & #4  
Spring 2005  
Due: 24 Feb 2005 & 3 Mar 2005

All of the following programs need only handle numbers that fit into long integers (64 bit quantities). Real implementations of these algorithms must handle arbitrarily long integers.

You may work in teams of two for this assignment.

Homework #3 comprises items 1-3, and is due at 11:59 pm on 24 Feb 2005. Homework #4 comprises items 4 & 5, and is due at 11:59 pm on 3 Mar 2005.

1. Write a program that takes two integers (a and b) and (pseudo-) randomly chooses a number between them (inclusive). For example, if I entered 1 and 100 as input, it would choose a number between 1 and 100. If I entered  $2^{30}$  and  $(2^{32} - 1)$ , it would choose one in that range. You may use whatever random number generator is available on your computer (note that this is too weak for a real cryptosystem).
2. Write a program to choose a 32-bit (likely) prime number. Note that this is a fairly small prime, by RSA standards, but will work for our purposes. Document what primality test you use.
3. Write a program that implements the extended version of Euclid's algorithm. The output should comprise the gcd of the two input values,  $d = \text{gcd}(a, b)$ , and the values of  $s$  and  $t$  such that  $\text{gcd}(a, b) = s \cdot a + t \cdot b$ . If you wish, you may include debugging output for each step of the algorithm.
4. Write a program to compute  $m^e \pmod{n}$ . Note that  $e$  will be a 32-bit number, and  $m$  will be at most 6-ASCII characters long (i.e., it will fit in 48 bits of a 64-bit word). You should consider the "tricks" from the "HOW DO WE COMPUTE  $m^e \pmod{n}$ " slide in the notes.
5. Graduate students: compose the above programs to implement RSA encryption and decryption. This is not cryptanalysis, but rather building the basic key selection, encryption, and decryption blocks for RSA. You should have write two programs: `rsa_keygen` and `rsa_crypt`. `rsa_keygen` should print out  $e, n, d, p, q,$  and  $\phi(n)$ . Note that normally, one would only publish  $e$  and  $n$ . `rsa_crypt` takes as inputs the message  $m$  (again, at most 48 bits),  $n$ , and either  $d$  or  $e$  depending on which way this encryption is going. The result should be the en-/deciphered text, in hexadecimal characters (e.g, 0x04ab72c15d01).

Undergraduate students: Explain how you would combine the programs from steps 1-4 to implement RSA. You do not have to write this program, only explain how you would.

We will distribute sample data for you to do some simple testing on your programs.