

## Introduction

The point of this project is to create your own implementation of the RSA public key encryption algorithm. Go read the Wikipedia entry on RSA to get the details.

The project is divided into two stages:

1. **Due 04/08/09** The first stage is to write the auxiliary functions to implement:
  - (a) the extended euclidean algorithm
  - (b) computing the multiplicative inverse (if it exists) of an element in a modular ring
  - (c) exponentiation through repeated squaring
  - (d) the Chinese Remainder Theorem.
2. **Due 04/15/09** Turn in a complete RSA implementation.

## Specifics

Here are the specific details for each function that you must implement.

1. **Extended Euclidean Algorithm** You must write a function “my\_xgcd” which takes as input two integers,  $a$  and  $b$ , and returns a list  $[x, y]$  such that  $ax + by = \gcd(a, b)$ . If you were paying attention in class, then you should already be done with this one.
2. **Multiplicative Inverse** You must create a function called, “my\_mul\_inv” which takes two inputs,  $x$ , and  $m$ . The function must return  $y$  such that  $xy \equiv 1 \pmod{m}$ , or if no such  $y$  exists, then the function must return 0. Again, if you were paying attention in class, then this is trivial once you finished the previous function.
3. **Exponentiation** You must create a function called “my\_pow” that takes three inputs,  $x, n$ , and  $m$ . The function must return the value  $y$  such that  $y \equiv x^n \pmod{m}$ . Your function should use the method of repeated squaring so that it can run in a timely manner. (Your function must handle positive and negative exponents.)
4. **Chinese Remainder Theorem** You must create a function called, “my\_crt” which implements the Chinese Remainder Theorem. Your function will take as input a list whose elements are themselves lists of two integers, e.g.  $input\_list = [[a_1, m_1], [a_2, m_2], \dots, [a_n, m_n]]$ . The function returns a list of two integers,  $[x, M]$ , where  $x \equiv a_1 \pmod{m_1}, x \equiv a_2 \pmod{m_2}, \dots, x \equiv a_n \pmod{m_n}$ , and any other integer,  $y$ , that satisfies the same equivalences is equivalent to  $x$  modulo  $M$ . Or, if no solution exists, the function returns an empty list.
5. **RSA** You must create three functions which will be used to implement the RSA public key algorithm.
  - (a) **my\_RSA\_gen\_key\_pair** This function must create a public/private RSA key pair. It must return a list where the first element is the public key and the second element is the private key. The function may take a single optional input which you can use to seed the random number generator. You may choose the format of the public and private keys, but make sure that they are consistent

with your encrypt and decrypt functions. The prime numbers that you choose should be larger than  $2^{1024}$ .

For our purposes, the Public Key will be a list consisting of two integers. The first integer is the RSA modulus. The second integer is the RSA encryption exponent.

For our purposes, the Private Key will be a list consisting of three integers. The first two integers are the prime factors of the RSA modulus. The third is the RSA encryption exponent. Note that this is slightly different than the common implementation...this is deliberate so that you will write your own code rather than translating stuff from the web.

- (b) **my\_RSA\_encrypt** This function takes two inputs, a message,  $M$ , and a public key. It returns  $C$  which is the result of encrypting  $M$  with the public key. For our purposes, the message will always be an integer between 1 and  $2^{512}$ .
- (c) **my\_RSA\_decrypt** This function takes two inputs, an encrypted message,  $C$ , and a private key. It should return the message  $M$  that was encrypted with the corresponding public key.

## Grading

On or before each of the due dates listed in the introduction section, someone from your group must demonstrate your working functions to me. If your group does not have the functions working by their listed due dates, then I will deduct one third of a letter grade for each non-working function.

As before, the code you submit must be well documented. Please indicate which members of your group are responsible for which pieces of code inside the documentation.

## Honor Code

All of these routines are widely implemented and implementations of various forms are scattered around the internet. You must not simply copy someone else's code. Your group may examine pseudo-code for these algorithms that is listed on Wikipedia and elsewhere to understand how the algorithms work. However, in order to ensure that you truly understand the code you are writing, I insist that you use a "clean room" design philosophy. It works like this: you begin by searching the internet, math textbooks, etc. for all the information you think you need. You make sure that you completely understand the algorithms. Now, once you know that you understand the algorithms, you put yourself in a "clean room" to write the code.

This means that you don't use any other sources of information about the algorithms; You don't use Wikipedia, any notes, print outs, math textbooks, etc. You only use your brain, the brains of your team members, and whatever Sage or Python manuals you need to look up language details. You sit down and you write the code without using any outside math or algorithm references.

If, while writing the code, you discover that you don't actually understand the algorithm, you delete all of the code you have written in that session and you go back to searching and reading until you are ready to try again. (This is a method commonly used by companies that want to ensure that they can legally prove that intellectual property they develop doesn't infringe on other copyrights... and they will quite literally put their employees in a clean room with nothing in it. The employees must then ask a third party "legal witness" to supply them with whatever tools they need... for example, a computer with an operating system, compiler, etc. The witness can then testify that all the code created by the employees is original and there was no cut-and-pasting from other pieces of software.)

At the beginning of your program, put a comment attesting to your use of a "clean room" development method.