Final Project — Phase Three

Max Olivier

March 8, 2009

Readings
There will be reading from Stinson here, specifically on solutions to the system $a * x \equiv b \pmod{n}$ and why there are solutions $\iff \gcd(a, n) | b$, that there are exactly $\gcd(a, n)$ solutions $\pmod{n}$, etc.
Main Ideas

- By now I'm sure you've guessed that the next phase in the project will be to make a decode_rand() function or your Mult_Cipher class. Before jumping into it do review the pages above carefully though because decrypting a message from a multiplicative cipher is actually considerably more difficult than decrypting one from an additive cipher.

Exercise 1 (Cryptololgy Program)
Add a decode_rand() function to your Mult_Cipher.cpp program that takes a pointer to a string as a parameter and decodes that string (though the returns type as with our other encode and decode functions should be void). The structure should be similar to the corresponding function that you wrote for you Add_Cipher program, and all the hints/reminders from the last phase of the project still apply. Things are slightly different with a multiplicative cipher though because, as you will recall from the reading, the congruence

$$a * x \equiv b \pmod{n}$$

has many more restriction on the solutions, and can possibly have multiple solutions for x. And since decoding a multiplicative cipher encoded message amounts to solving that congruence (with x as the possible key) we have more work to do here. So, when writing the program keep in mind that

- There are obviously many possible ways to go about making this function. But since, unlike with the additive cipher, there can be several possible keys for a given guess of which plaintext letter corresponds to the most frequently appearing ciphertext letter, I would suggest that you make a 2-D array with the valid keys in one row and 0 or 1 in the second with 1 corresponding to a valid key for this plaintext letter guess and 0 an invalid key. Use separate functions to determine whether a 0 or 1 should be placed for a given possible key. Note that if you use this approach you will also need a function to determine the number of numbers coprime to a given n so that you know the size of the array. Note that the syntax for passing a two dimensional array as a parameter into a C++ function is:
  function(int array[ ] [k])
  where k is the number of rows in the array. (In general to pass a multidimensional array into a function all the dimensions but the first must be specified).

- Keep this weeks reading in mind throughout the project!
- Feel free to use helper functions liberally since the decode_rand() function will get very large and cumbersome.