Assignment 5 Write-up

James Contini

2022 October 26

General information

Citations

None.

Description

Since there was no explicit description for this write-up I thought I would just take this time to talk about efficiency.

1 Efficiency

1.1 *pow mod()* discussion

This function takes four parameters, an out, a base, an exponent and a modulus. Using the fact that any integer, can be represented as a polynomial, we can calculate the exponent out of a sum of base two polynomials. Similar to how decimal numbers are represented in binary. In class the other day someone remarked how they thought this function was fast but I think what got lost in translation was the word efficient. Computing large powers modulo some number makes it so that the computer doesn't actually have to store those large numbers but instead kind of overflow through modulo. This makes pow mod faster than a normal power function solely because its dealing with numbers in modulo n which means they are ceiling restricted.

1.2 *make prime()* discussion

Ben mentioned something that helped speed up my code by about two times in the piazza which was that we can make sure all of our generated primes our odd by setting the first bit. This makes sense because its equivalent to saying 1 plus 2 to any number. It will always be odd. This was the only optimization I used except for the clang flag -Ofast but that didn't really improve anything noticeably.

1.3 *Conclusion*

The amount of time it takes to generate a large number that's actually viable in terms of security and the amount of time it takes to encrypt and decrypt these numbers makes RSA encryption very bad for things of high data. But signatures and similar smaller things work very well with this technique.