Asgn 5 Writeup

Mika Peer Shalem

Mpeersha

What I learned:

To begin with, the assignment helped me understand the importance of sharing an encrypted message in an online space. As well as the history of cryptography and the process of getting to the RSA system.

I learned about the different types of key cryptography. For example, in the assignment, we used symmetric keys. This means that the key for encrypting is also used for decrypting, while asymmetric cryptography means that there are two different keys. I understand more about the different possible permissions for files and keys. When we created the private key, we had to adjust the permissions so only the owner could read it and write to it.

I continued practicing ways to optimize my code. The original pseudo-code worked, but it was not as effective as it could have been. It generated large public keys slowly, so finding ways to change my code to improve performance was a big part of the assignment. For example, at the make_prime function, I generated both odd and even numbers even though no even number (in the range specified in the assignment) is a prime number. Instead of testing to see if even numbers are prime, I immediately discard them to save time.

I learned how to have multiple main functions in one assignment. Once again, I had to figure out how to handle multiple files of functions and the connection between them. The three main functions can work by themselves, but all connect to the same idea. Thus, I improved my testing skills and found ways to ensure each function fulfilled its purpose and worked as intended. I learned which functions are more integrable for the program and the order in which I should start coding. I started with writing the Keygen main function but realized that it depends on so many RSA library functions and Numtheory functions that I decided to switch the order of operations. It was interesting to analyze the list of functions used in the program and organize them by their usability. The skill is important not only in computer science courses but also in other fields in life. It is essential to learn how to organize the tasks ahead and see the ways they interact with one another.

How I tested my code:

I had two main ways to test my code. The first one was to print the output of each function I wrote. So, after writing one of the functions in the Numtheory program, I tested it with multiple inputs. When I used small inputs, I calculated the result by hand and compared the program's output to my result. However, as the size of the inputs increased, I used online calculators to compare my results. If I found any difference in the results, I added more print statements in my functions and made sure that I did each step correctly. This helped me figure out that the operations in the functions change the original value of the arguments. So, I had to create copies of the arguments before proceeding with the calculations.

Afterward, I tested my Keygen, Encrypt, and Decrypt with the provided dist functions. I did not have a way to test my Keygen function as it relied on random number generation. However, I was able to compare the outputs from the keygen-dist and my function. This way, I ensured that I use the right amount of bits and used the correct format for the print statements. I also called my keygen program and used the encrypt-dist and decrypt-dist functions to make sure that each of the components I set in Keygen worked correctly.

Testing the encrypt and decrypt functions went smoothly. I generated a key with either my Keygen program or the provided one, then I encrypted the same message using both my encrypt and the provided encrypt program. I saved the outputs to two files (using the command line operation > to send the output to a file). When the output was short, I used the "cat" command to display the two outputs one after the other and read it character by character to see if I noticed any difference. For larger outputs, I used the "diff" command to see if there was any difference between the files.

At first, I used the same method for the decrypt program. However, I realized that when the contents of a file changed on my Virtual Machine, the file had a pop-up and had to be reloaded to see the new data. So, I saved the original (unencrypted) message in a file and saved the output from the decrypt program on the same file. Then, if I got the message that the data in the file had changed, I knew that I needed to improve my decrypt program. If the message stayed the same, I knew my program works.

Citations

- Toturial on GMP - https://home.cs.colorado.edu/~srirams/courses/csci2824-spr14/gmpTutorial.html
- Fchmod https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-fchmod-change-mode-file-directory-by-descriptor
- Fopen - https://www.tutorialspoint.com/c_standard_library/c_function_fopen.htm
- Fileno - https://www.ibm.com/docs/en/zos/2.1.0?topic=functions-fileno-get-file-descriptor-from-open-stream
- Files - https://www.cs.utah.edu/~germain/PPS/Topics/C_Language/file_IO.html#:~:text=The%20basic%20steps%20for%20using,if%20the%20variable%20%3D%3D%20NULL.
- GMP lib manual - https://gmplib.org/manual/Integer-Functions
- RSA algorithm step by step video - https://www.youtube.com/watch?v=j2NBya6ADSY&ab_channel=Programmingw%2FProfessorSluiter