Breakdown of security of HW 2

My first decision made in security for HW2 was using C++ instead of C because it has less security vulnerabilities since memory is automatically allocated to strings amongst other things. With C++ dealing with buffer overflows is easier than dealing with it in C. To avoid buffer overflow I used std:string instead of a char array this dynamically allocates memory for the string depending on the size of the input rather than statically sizing a string. This avoids both out of bounds array if the users input is too long or a buffer overflow. When taking user input I switched between using std:: cin, and std::getline. The reason for this switch is cin is really effective when taking in singular strings with no spaces. So, I only used cin when I wanted the user to give one word input like for encrypt or decrypt. I used getline when the user could input multiple lines, so it not only got the right space put also kept the new lines that the new line characters the user input. While getting input I also needed to check if it was valid.

For getting the input when I only wanted input that was a specific word I could use if statements to make sure the input matched the input, I wanted like to be a 'text' or 'file' input. If the input didn't match one of these, I encased the code in a while statement so it would throw an error, and the user would input until the strings matched one of the options. The hardest input to validate was getting the key. In C++ storing an input as an integer is friendly only if the user inputs an integer. If instead the user inputs a string it will crash the program. So instead we take the users input as a string, check that each character in the string is a digit, if not repeat the input process. Then we finally checked if the key is within our number range 0 to 94.

Lastly, I needed to create a secure way of logging into the program. To start I used cin to get the users password because I only wanted a one word response. Next, I created a function that would validate the user's password. I ran the password through the hash function I used, which was given in the C++ libraries I used, storing the output as the global variable password. Lastly, I checked the hash of the users password with the global hash to see if the password was right user continues if it is wrong the user is logged out of the program.