Jason Wilson

Final Writeup

Professor Arias

Friday, December 07, 2018

Abstract:

Creating a symmetric key encryption algorithm that uses truly randomly generated numbers to scramble the data. The program is designed to take an input of text, either by importing the text from a text file or based on a user input into a text box. The program will then generate truly random numbers that are based on user input, encrypt twice, generating a second key in the process, and output unintelligible characters and the entire key. Then the program has the ability to decrypt the encrypted string by using the generated key to reverse the encryption algorithm. All steps of the encryption/decryption process have the option to be saved to a file onto the systems drive. Using Java Swing, all process have been made easy to use by providing a simple user interface.

Introduction:

The motivation behind creating this encryption program was primarily to gain a better understanding of encryption and how it works. Homemade encryption programs should not be used over any of the current encryption standards when encryption data with any sensitivity. That aside, it has been a good way to better understand all of the lessons from class by applying each topic in its own way. This project has been a great way to learn more about the way that I problem solve and allows me to think about the inefficiencies that I may have had in this process, allowing me to try different approaches in the future. It has also been a good way for me to understand how I learn on my own.

This projects overall goal is to make a custom encryption program that generates truly random keys based on user input for a harder to break encryption. Truly random numbers are

important for an encryption program because it was found that random number generators programmed into systems are not completely secure. If the generator is simply an algorithm, that randomness can in some way be predicted and reversed. An interest in how creating numbers based on user input and the desire to learn the basics of encryption are more of what motivates the work for this project.

In this paper, you can expect to read what exactly the system does and how users will interact with it. As well as how the classes in the program interact with each other written in UML. This will all be apart of the detailed system description. Then, the paper will talk about the requirements of the system, what the program must achieve and how it must achieve it. Then, describe what other work has been done to address a similar problem. After that, a user manual about how to use the application. Then a conclusion, and finally, references.

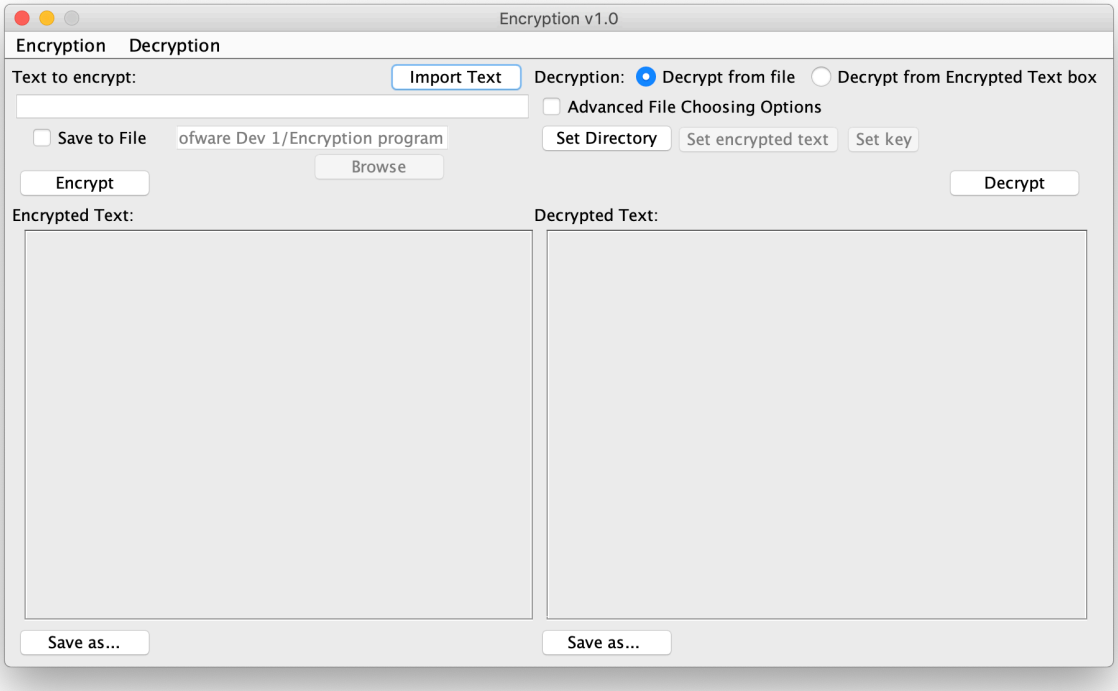
Detailed System Description:

The system both encrypts and decrypts data by generating and reading keys that correspond to the encrypted text. Each key that is generated is specific to the text that it was generated with. The key in this system is divided into two parts as the program encrypts with two keys and decrypts with two keys. The first key is generated by gathering user input with use of a key listener on a text box within the program. Every key pressed and released on the keyboard while the text box is selected will calculate the time the application started running, subtract that from the time the key was pressed or released and add the char value of the key that was pressed. Then the system takes all of those numbers and puts them into an array of long number and is sent to the encryption class to use once the encryption process is initiated. The second key is generated in the second stage of encryption. Once the characters char values have been shifted into new numbers by use of the first key, which will be talked about later, each individual number from the array is taken and shifted again. This shift is where the second key is generated, each number is carefully selected to create a value that will be printable on the ASCII table, excluding any ASCII values that have cause problems in the

decryption process. This second key is saved as an array list, and once the program saves the encryption data to use at another time, the two keys are combined into one file.

The Encrypt class is the class that handles the encryption of the data passed to it. Once the decrypted text and the first key are set the program is ready to begin encryption. Once the encryption starts, the program turns all of the characters in to their integer value equivalents and shifts their char value using the equation: (randomNumber1 * char value) + randomNumber2. Random number 1 and 2 change each time the program is looped for the entirety of the string, and of the program runs out of random numbers to use it will go back to the beginning of the list and continue on. This means the longer the first key is, the more secure the encryption will be. After that, the shifted char values are divided up into single digit number(ex.{1234,5678} turns into 1,2,3,4,-1,5,6,7,8. With -1 representing the comma dividing the two shifted values). Then 48 is added to each of those numbers and then are shifted by the second key to generate valid new char values that can be displayed. Once this step is complete, the encryption is over and the user will have the option to save to file. The encrypted text is completely unintelligible and can only be read again by reversing the encryption process.

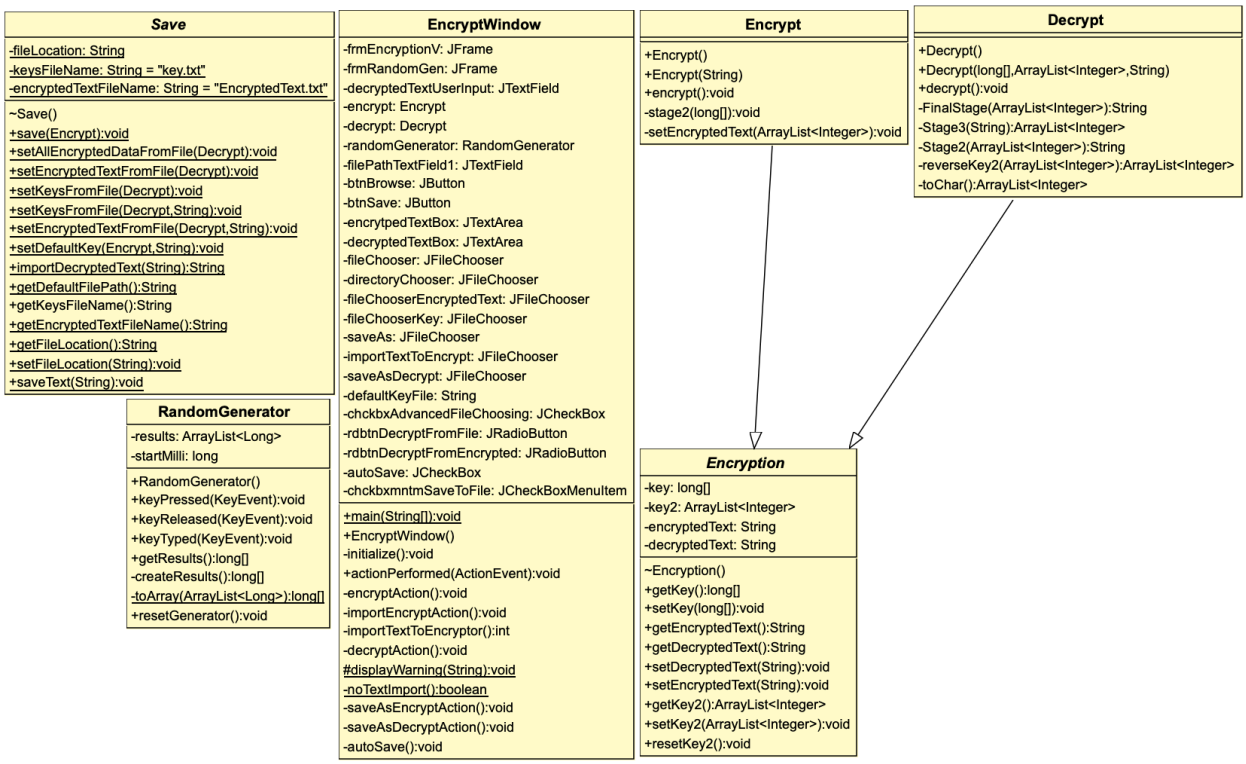
The Decrypt class is the class that is responsible for reversing the encryption when the proper key and encrypted text combo are loaded into it. The decryption program imports the encrypted text and keys and beings decrypting. First it takes all of the characters from the text file and converts them to their char values, respectively. Then it reverses each value by the second key. Then, using a string builder, it takes each value and parses it to a long value between each -1 value to recombine the number, preparing it for the use of the first key. Once that completes, equation used above is reversed and applied to each number using the values of the same key it was encrypted with. Finally, the char values are converted to their characters and displayed. The user can then view what was once encrypted, as decrypted text and export.



Referring to the image above, this is how one would interact with the program. Upon running the program, the user is presented with the window above. They can start by typing text to encrypt into the small dialog box labeled, “Text to encrypt:”. Then, once the text is inputted the user can click encrypt and the text from that text box will be encrypted into unintelligible text, displayed in the “Encrypted Text:” text box. The user also has the option to import text by using the import text button. Once clicked, the user will be prompted with an open dialogue where the user can specify which file to encrypt. Once okay is clicked in the open dialogue, the encryption will begin immediately. Once complete it will be displayed in the “Encrypted Text:” text box. The user has the option to check the “Save to file” check box which will save the encryption to the specified directory immediately once the encryption is complete. The default safe folder is the folder that the program is running from. There is also a “Save as…” option under the ““Encrypted Text:” text box. This will allow the user to select a directory to save the encryption and key to after the encryption is complete.

As far as decryption goes, the user can preform this action from the right side of the encryption window. By default, the decrypt from file selection is selected. This allows the user to decrypted a previously saved encryption from the program. Clicking the set directory option is used to choose a directory that contains both the files EncryptedText.txt and the associated key.txt files. If these files do not exist or are have different names, the user must choose the advanced file choosing options check box, where they can choose both the encrypted text file and key file individually. Once this is done the user can click the decrypt button and the program will decrypt the text. The second option for the decryption program is to decrypt from encrypted text box. This allows the user to immediately see the results of the decryption after encrypting it. It will take the text that is already in the “Encrypted text:” text box and decrypt it once the decrypt button is pressed. This is how any user would interact with this program.

Here are the UML diagrams describing how the classes interact with each other:



Requirements:

Problems that this system is addressing is the everlasting effort to secure the worlds data, or at least gain a better understanding of it and to create truly random numbers based on user input and use those numbers to encrypt data. Standard random number generators cannot be trusted as they can inevitably be predicted. When attempting to protect data of the user, created a number generator that generates truly random numbers is very important. The encryption must also display the encryption as text, meaning there are no arrays of numbers. Those must be converted into characters values. It must also be able to encrypt text data that is sent to it, no matter the length. Then, it must be able to completely decrypt the data with proper accuracy. Furthermore, it must be able to save its data to files allowing it to decrypt regardless if the application has been restarted or not. It must also be able to save the decrypted text result to a text file and import encrypted text and its key to be able to decrypt. It must also allow for these actions to be preformed in a painless and easy way with a user friendly window, prompting the user if they have done something wrong or if something could have gone wrong with the decryption during the decryption process.

Literature Survey:

As far as other works that people have done to address a similar problem. There are all of the previous encryption standards that are implemented heavily by the government and companies across the world in order to secure their data from data breaches. There are also a lot of small projects where people simply wish to encrypt data in a non serious way, but these encryptions, including mine, are only for a learning experience and should not be trusted unless heavily tested. In programs similar to mine, with regard to the generation of truly random numbers, they take similar approaches. They use mouse listeners that record the position of the mouse to use. Some ever use auditory inputs from the microphone, camera inputs, and keyboard inputs(like mine). In one instance, a company get random numbers from the

unpredictability of lava lamps. By taking pictures of a wall of lava lamps, they are able to generate secure random number for the encryption of data in an extremely unpredictable way.

User Manual: (Step by Step)

1. Run the application either by:
    1. Compiling it from the EncryptWindow class
    2. Launching the generated jar file for the application titled “Encryption Program.jar”
2. Encrypt data:
    1. Type text to encrypt into the text box labeled “Text to encrypt” and click the encrypt button when done. Or
    2. Click the import text button and browse for a text file to encrypt
        1. Encryption will commence once browse window is exited by clicking open
    3. Save the encrypted text to a file by clicking the save as button under the encrypted text text box
3. Decrypt data:
    1. Choose either “Decrypt from file” or “Decrypt from encrypted text box”
        1. If decrypt from file is chosen, click set directory to choose one that contains the files “EncryptedText.txt” and “key.txt” exactly
            1. If the encrypted data and keys are separated, or they are named something other than exactly what is listed above select the “Advanced File Choosing Options” and choose the encrypted text file and the key file, individually
        2. If Decrypt from encrypted text box is chosen
            1. Make sure the encrypted text text box has encrypted data in it
    2. Click “Decrypt” to begin decryption
    3. Save the decrypted data to file by clicking the “Save as… ” button, located under the decrypted text text box.
        1. Choose a directory to save to and add a file name and save

Conclusion:

The system that was created accomplished all of the goals that were set at the beginning of the process and meets all of the requirements. The encryption program encrypts data using truly random numbers that are generated through user input and displays the encrypted data as readable text. It can also encrypt and decrypt data of any length, although, more data results in longer runtimes. The program also decrypts data by successfully reversing the encryption process and displays the original text no matter the length. Through the use of the Save class created, the program is able to save both the encrypted data and decrypted data, as well as import data from text files for the encryption process and decryption process. Overall, this program meets all of the requirements that were set for it.

References:

https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf

https://www.howtogeek.com/183051/htg-explains-how-computers-generate-random-numbers/

https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide