RSA Encryption/Decryption

CS 342 Design Report

This program performs several operations related to RSA Encryption/Decryption algorithm. Task in this project include creating a Public/Private key pair, "Blocking" a large message into mineable size chunks and vice versa, perform the encryption, decryption process and more. Detailed information about the project is in the pdf of project[1]. The Authors of the project are : Dhrumil Patel, dpate85@uic.edu, and Kena Patel, kpate241@uic.edu. We are currently pursuing Bachelors degree in Computer Science at University of Illinois at Chicago.

The purpose of the project is to gain experience in industrial standard encryption and decryption algorithm. First of all, the programs needs to create a private and public key. The public key will be used by the other client, whose message we will be decrypting. And we will need the private key to decrypt the message. The keys need to be in certain xml format and should be generated using prime numbers. The next step is to use the Message Blocking process to translate message in a single numeric value by summing the ASCII numeric value for each character multiplied by 128^(pos) where pos is the character's position in the message. If the message is too large, RSA will block the message into smaller chunks and then encode the message by applying the algorithm described above. Detailed information about Blocking a message is in the project description. The next step is to implement the Encryption and Decryption process by utilizing those, Blocking and Keys, methods. The same math is used for encryption as well as decryption. The process will be described later in this document.

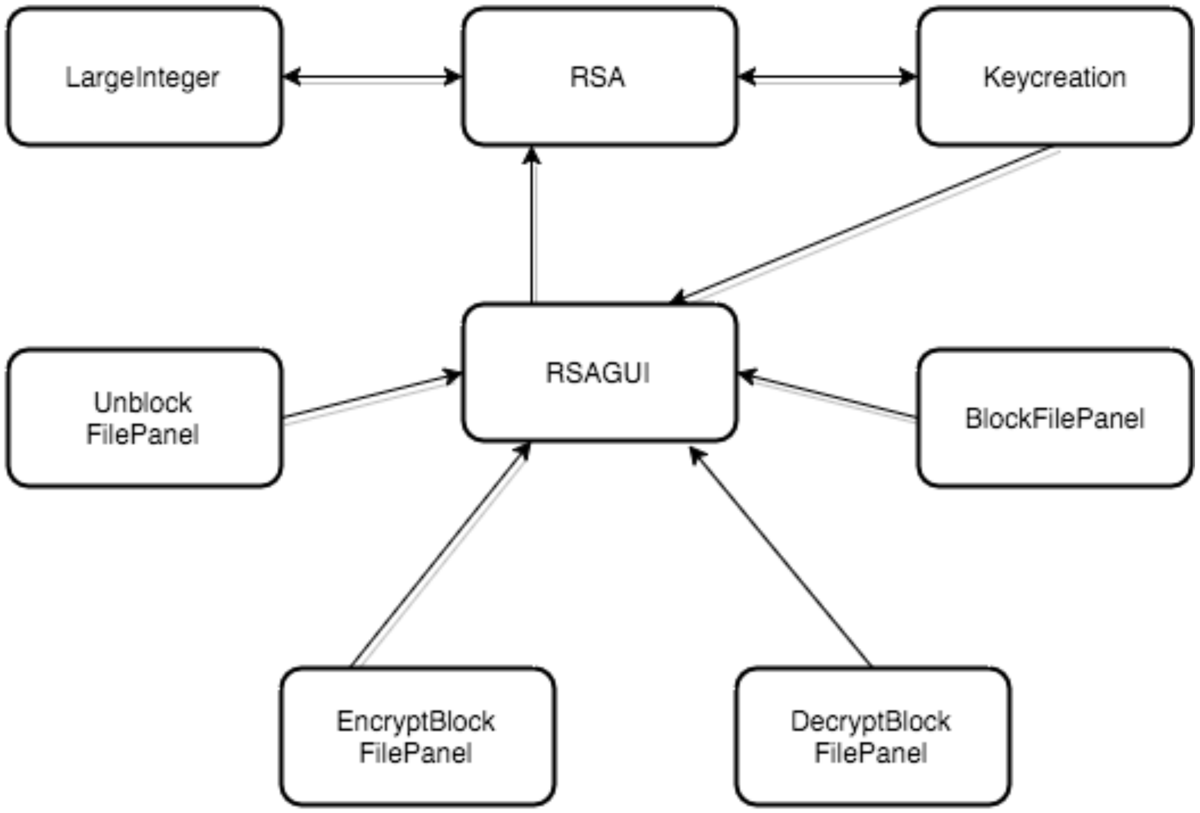
To achieve such functionality along with maintaining a sustainable design, we decided to make class for each task. Each class is related to other and follows a logical pattern. Brief description is listed in the table below:

Filename	**Brief Description**
BlockFilePanel	Used for blocking the file. Prompts the user to enter block size, filename, and the name of block file. Handles block file input errors. Utilized in graphical user interface.
DecryptBlockFilePanel	Decrypts the block file. Utilized for GUI.
EncryptBlockFilePanel	Encrypt the block file. Utilized in graphical user interface.
KeyCreation	Create private and public keys. Follows the algorithm described in project description.
LargeInteger	The huge-unsigned-integer class. This class handles all the math required to block a file, unblock a file.

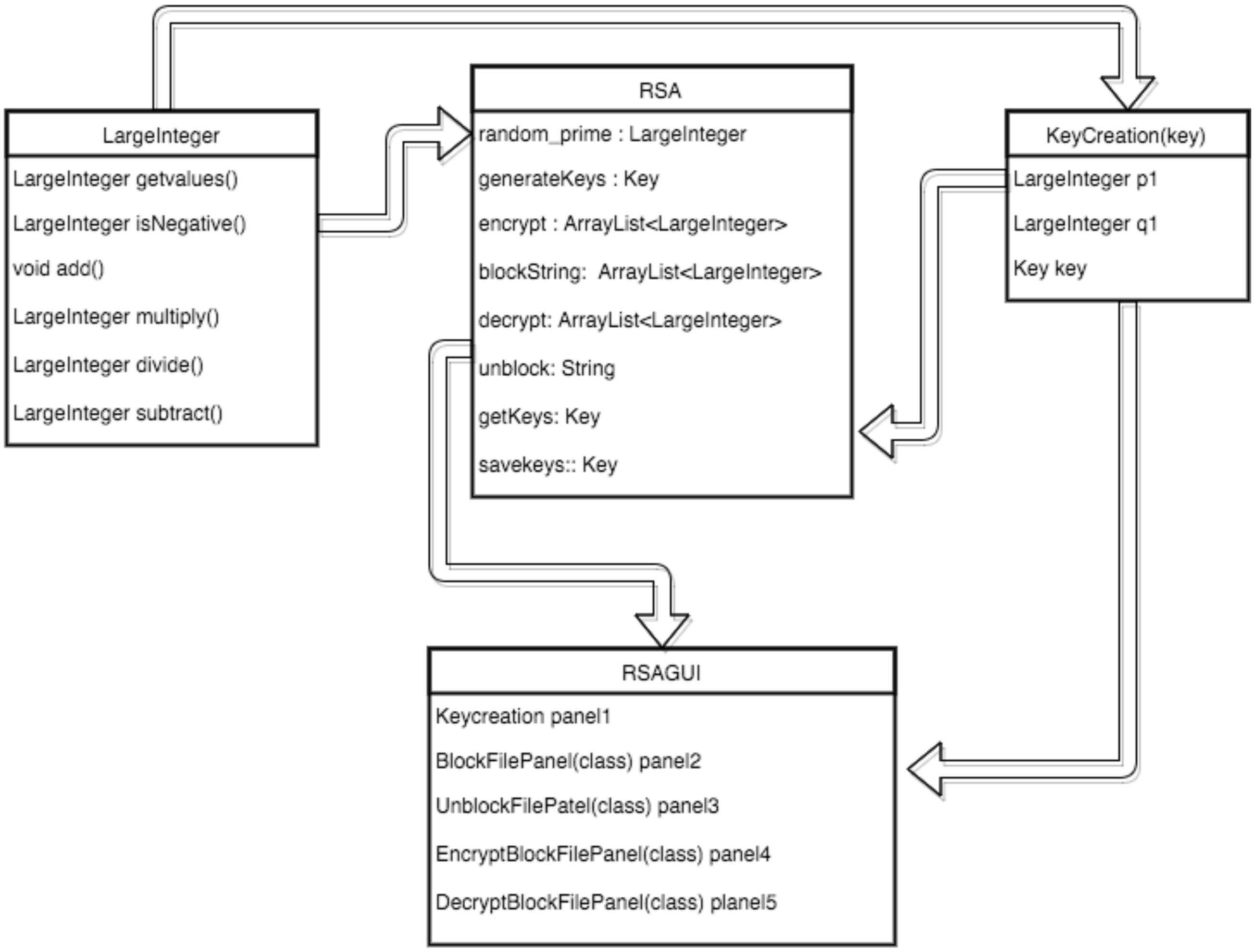
[1] https://www.cs.uic.edu/pub/CS342/AssignmentsS16/proj3s16.pdf

Filename	Brief Description
RSA	This file contains all the methods used for encrypting, decrypting, blocking, etc. It relies LargeInteger class to perform the math for creating the tasks described above. It also checks of the number is prime or not. It has the main function. Can be used instead of using the GUI. Executes user commands via I/O
UnblockFilePanel	Used for unblocking the file. Utilized in GUI
RSAGui	Graphical User Interface for the RSA Encryption/Decryption. All the xxPanel classes are connected to this file. Whenever a user presses a certain button, relative xxPanel is called via actionlistner and the task is performed.

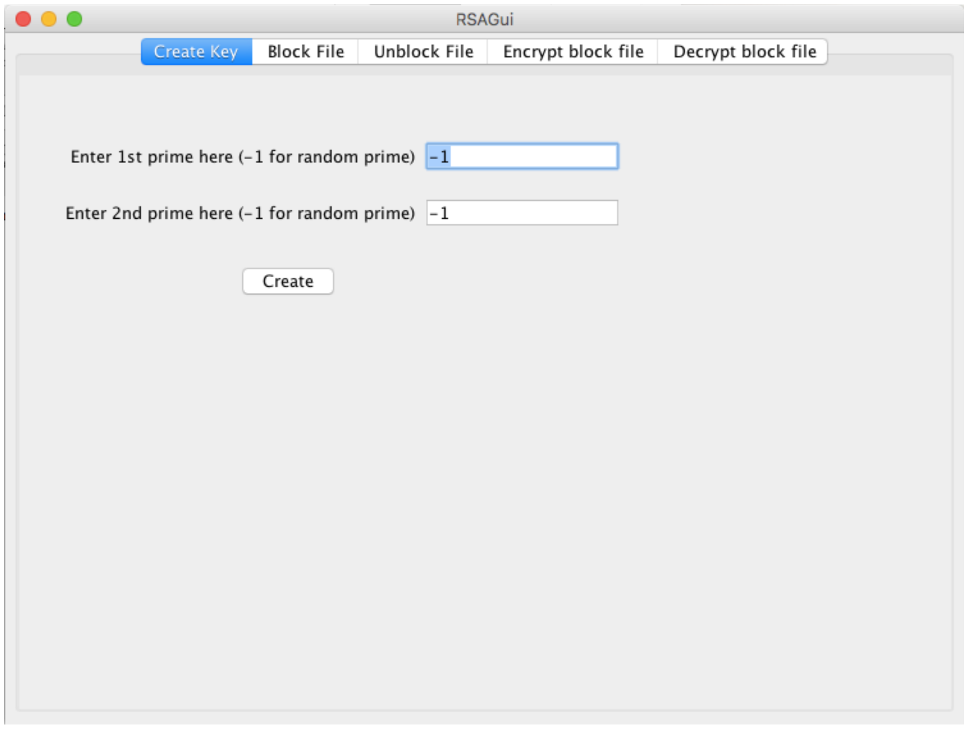
The connection between the files is in the digram below:



- RSA: this class uses all the classes together. the main function of this class takes in the prime number. It then passes the prime number to keyCreation class and LargeInteger class where all the operation occur.
- KeyCreation: It takes in the prime values. Applies the algorithm described in the project description and returns the file numbers which later gets converted to xml file.
- LargeInteger: This class does all the math operation for blocking, unblocking, encrypting and decrypting.
- RSAGUI : As mentioned before, this class just gets all the panel together, When a button is clicked, the appropriate class is called. All the panels have action listener implemented.


LargeInteger
LargeInteger getvalues()
LargeInteger isNegative()
void add()
LargeInteger multiply()
LargeInteger divide()
LargeInteger subtract()
RSA
random_prime : LargeInteger
generateKeys : Key
encrypt : ArrayList<LargeInteger>
blockString: ArrayList<LargeInteger>
decrypt: ArrayList<LargeInteger>
unblock: String
getKeys: Key
savekeys:: Key
KeyCreation(key)
LargeInteger p1
LargeInteger q1
Key key
RSAGUI
Keycreation panel1
BlockFilePanel(class) panel2
UnblockFilePatel(class) panel3
EncryptBlockFilePanel(class) panel4
DecryptBlockFilePanel(class) planel5

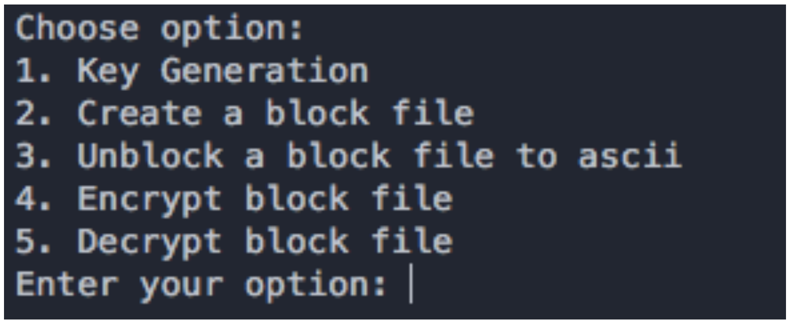
For the GUI, we've used rather simple design which has buttons related to separate tasks and it first opens up at prompt to enter prime numbers. As you can see in the pic, the buttons perform the tasks according to their name. They are self explanatory. If the user does not want to enter the prime number, -1 is set as default text field so random prime number is generated by the program itself. In case of the user entering not a prime number, the program shows an error saying the "The provided number is not a prime number" and thus it will not create the private and public key.



Enter 1st prime here (-1 for random prime) -1

Enter 2nd prime here (-1 for random prime) -1

When you click on the "xxxx File" button, there's a text saying create. When you click that text, the program will prompt the required information needed to perform the action desired via dialog. For example, when you click the "Block File" button, you'll see a text create. After clicking it, the program will show a dialog which will ask you to enter the "ASCII text file name" the "Block Size" and at last the "Block filename". It is the user's responsibility to enter correct information. But to keep error checking, in case the user enters a file which doesn't exist, the program will show a message "Error occurred during processing." This error checking is included in all methods.



```
Choose option:
1. Key Generation
2. Create a block file
3. Unblock a block file to ascii
4. Encrypt block file
5. Decrypt block file
Enter your option:
```

The program can also be used in shell. It works similarly to the GUI. Selecting an option will prompt the user to enter the information for performing related task.

The program is basically used as

- We first create a private/public key set by entering prime numbers.
- Share the public key to the client whose message you want to receive and decrypt
- The client creates a text file and blocks the message into a blocked file with desired block size
- The client uses public key shared by us to encrypt the blocked file.
- The client then sends the encrypted blocked file to us
- We decrypt the encrypted blocked file using our private key to get the original blocked file the client created.
- We now unblock the file created to get the original text message made by the client.

The benefit of the program is the user interface. It's very simple and easy to use. At the development level, the classes provide an accessible control over the program. If the graphical interface needs to be change, only a class has to change instead of changing the whole program.The code has meaningful names and the function names are self explanatory. The instructions are straight forward in the GUI as well as in the shell. In terms of risk, there are many things that could go wrong. For example, keeping the keys in same place is a security risk. If the LargeInteger class is changed with irrational math ops, then the conversion will always be false. Same goes with keyCreation class.