Lock N Key

A G.C.K program

George, Chis, Seth

Our objective in this project is to make a program that is user friendly, which allows the user to navigate the systems interface to encrypt and or decrypt messages with the use of a key word generated by the user. The program then encrypts or decrypt based on the operation selected by the user, using that key.

Problems that we have run into with this program have been the utilization of the ASCII table and our own table in the encryption process. User interface errors from the user making a selection other than that which is the menu option, other issues that arose while in the building process was the implementation of using numbers and symbols in the key word, although after some thought it was best to allow this as it would allow for a stronger encryption over all.

Solutions that we came up with to fix the few problems that we ran into are that we limited the inputs the what was requested in the menu, anything entered other than the menu option will be rejected and the user will be asked to reselect a menu option. This was implemented on all of your option related menus so that the user is directed through the system as outlined in the program flow-chart.

The message and keyword are searched through our table. When the element of the Table matches the char of the message, the index of the Table is push back into a vector for the message. The same process is used for the keyword.

The encryption process:

Ei = (Mi + Ki + (ML% TL) + ( I % TL)) % TL.

The decryption process:

Di = (Ei – Ki + (TL \* TL) – (EL % TL + (I % TL))) % TL

The program uses combinatorics, sequences, algorithm and modular arithmetic. The combinatorics in the program is from the 95 possible choices for each encrypted position which can be expressed with 95msg length as the number of possible combinations. The sequence implemented in the program is our table which is then converted into the encrypted table with our encryption algorithm. Algorithm is implemented in the form of searching our table with a match to the user message and key, then inserting the index value to a new vector. Modular arithmetic is used in our encryption and decryption set to modulus 95 which is our table size.

Cryptography and Modular Arithmetic – chapter 5

Combinatorics – chapter 8

Algorithm – Chapter 10  
Sequence, sums, and series – chapter 11

We are only limited by the amount of characters that can be entered in a single line by the user.

To add more lines to encrypt or decrypt we can add a vector that can ask the user how many lines or messages they want to encrypt or decrypt. Once they answer we can create the string vector and get the users input to push back those messages into the vector. Once they finish inputting the messages we can encrypt or decrypt them and print them out or we can ask the user which message they would like to see.

Int ask\_number\_msg

Vector <string> msg

String tmpInt choice

Output: What would you like to do?

1. Encrypt

2. Decrypt

Input: choice

If they choose to encrypt

Output: How many messages would you like to Encrypt?

Input: ask\_number\_msg

For (int i = 0; I < ask\_number\_msg; i++)

Cout << Encrypt Message (i + 1):

Cin >> tmp

msg.pushback(tmp)

if they choose to decrypt

Output: How many messages would you like to Decrypt?

Input: ask\_number\_msg

For (int i = 0; I < ask\_number\_msg; i++)

Cout << Decrypt Message (i + 1):

Cin >> tmp

msg.pushback(tmp)