Ministry of Education and Science of the Republic of Kazakhstan
Al-Farabi Kazakh National University
Faculty: “Mechanics and Mathematics”
Department: “Mathematical and computer modeling”

Project report

Theme: **RC4 and its cryptanalysis**

Done by: Akmuratova S.M., Kakibay A. K.

Checked by: Kudaibergenov A.K.

.

Almaty - 2020y.

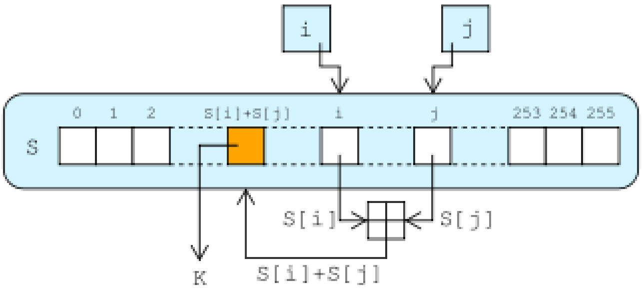
RC4 and its cryptosystem

RC4 is a stream cipher widely used in various information security systems. The RC4 algorithm is based on a pseudo-random bit generator. The key is written to the input of the generator, and pseudo-random bits are read at the output. The key length can range from 40 to 2048 bits. The generated bits have a uniform distribution.

The main advantages of the cipher:

- high speed operation;
- variable key size.

Algorithm:



1. Function is generating sequence of bits;(we denote it as k_i)
2. Then we do XOR to the sequence of bits and plaintext (here plaintext denote as m_i), after we can get cipher text (denote it as c);
   $$c_i = m_i \oplus k_i$$
   To decipher it do this algorithm again, then it will look like:
   $$m_i = (m_i \oplus k_i) \oplus k_i$$
3. To initialize RC4 we use "key-scheduling algorithm". In this algorithm we dive own key and key length. First, we fill the array box then this array is shuffled by permutations defined by the key. We need to be sure that box has the same value as was given during the initialization. Next,
   ```
   j = (key[i % key.Length] + box[i] + j) % 256
   ```
   will be performed, then just swap box[i] and box[j].
4. Next step is "pseudo-random generation algorithm". Here in one loop is defining one n-bits text from the keystream, then just swap box[y] and box[j]. After key will do XOR with plaintext.

RC4 and its cryptanalysis

First let us answer, what is cryptanalysis? Cryptanalysis is an attempt to decipher cipher text without key.

Now, let's look at cryptanalysis for RC4 cipher and show the weakness of this cipher and attacks on it, here we will consider two attacks, they are distinguishing attack and key recovery attack. Before starting explanation of attacks let us describe some weakness of RC4 cipher, and one of them is weakness of key-scheduling algorithm (KSA). Here are interior states with short-term cycles that generate some classes for secret keys, and by using a certain relation among the states can calculate the last internal states. In sample, here is deliberating that the state $S[0]$ in the array transfers and only with this state swap all further states. We can write the following relations as in algorithm of key-scheduling algorithm:

$i = 0, j = 0, k = 0$	$i = 1, j = 1, k = S[0]$	$i = 2, j = 2, k = S[0] + S[1]$
$j = 0 + S[0] + K[0] = 1$	$j = 1 + S[1] + K[1] = 2$	$j = 2 + S[2] + K[2] = 3$
$Swap(S[0], S[1])$	$Swap(S[1], S[2])$	$Swap(S[2], S[3])$
$S[0] = S[0] + S[1]$	$S[1] = S[1] + S[2]$	$S[2] = S[2] + S[3]$
$k = 0 + S[0]$	$k = S[0] + S[1]$	$k = S[0] + S[1] + S[2]$

These relations demonstrate that in RC4 S[i] with S[i+1] swap and one increment i and j, further weak keys can be found by using relations of probabilities.

Now let's talk about a low diffusion of bits in key-scheduling algorithm.

1. If the least significant bits of initial states = 0, then the least significant bits of keystreams will be 0 with the probability 1. We may range this for [2;32]-bit of initial states.

2. $S[i] \pmod{2^n} = 1 - K[i \pmod{l}]$ and $S[0] \pmod{2^n} = -K[0]$.
 So, j=i. It means: after 1[st] round, we get the even states of interval.

3. Assuming that K[0] (odd) and K[0]-2=K[1].Also,
 $S[0] \pmod{2^n} = (1 - K[0]) \pmod{2^n}$,
 here S[1] is even.
 $S[i] \pmod{2^n} = (1 - K[2])(i \bmod l)\ 3 < i < 255$, it means that states of interfal after one round be even.

Property of Low Diffusion:

The serious weakness of RC4 cipher is that the i-th bits depends from each other and when attacker changes the most important bits of initial values in key-scheduling algorithm then keystream's significant bits can be changed.

In algorithm of RC4 a secret key and initial bits can provide internal state like input for pseudo-random generation algorithm to generate output keystream, and all bits of output keystreams will be changed with probability near to 0.5 by complementing one bit of beginning state. This property called the torrential slide measure is one of the foremost fundamental properties of a secure cipher. But this property was made only for a small part of the bits of the initial value of the array. Since if we change the i-th bit, then most of the known bits will also change. But the insignificant part will not be changed.

Distinguishing Attack on RC4

Our first attack will be based on the weakness of our cipher, that is **nonrandom property of internal states**. Let us give some explanation for nonrandom property of internal states.

Our array S has 32-bit 256 elements and the pointer j has only one byte. And if we randomly select two indexes $i, j \in \{0,1\}^8$, then probability will be:

$$P(S[i] = S[j] = P(i = j) = 2^{-8}.$$

When $S[i] \neq S[j]\ (i \neq j)$, for 32-bit word this probability will be equal to 2^{-32}. And now we can prove that for each element for our array, after the algorithm is initialized, we have

$$P\left([S[i]]_0 = 0\right) = 0.5 + 2^{-8}, 0 < i < 256.$$

Statement 1. Let's assume that the index KSA - j will be uniformly distributed in the interval from 0 to N-1 and independent of index i. Also, if the two indices are not equal, then $S[i] + S[j]$ mod M will be uniform in the interval from 0 to N-1 and also independent. In this case, after the performance of KSA, for all elements of the array we will have:

$$P\left([S[i]]_0 = 0\right) = \frac{1}{2\left(1 + \frac{1}{2^n}\right)}, \qquad \text{where } 0 \le i < 2^n$$

This way we can find all the insignificant bits of the array are offset. If our key stream were array dependent, then we could use the offset for a distinctive attack. However the output of the key stream is the sum of the word from the array and the variable k. Including the variable k is the sum of the random elements of the array. We can also note that the least significant bit of the variable k will also be shifted, but it will be so small that it will be close to 0. And so we should use a combination of pins to eliminate the effect of the variable k and find the shifted linear connection. For example, a linear combination of two consecutive outputs may indicate the expected offset. In order to do this, let's take event E with this condition:

$k_{t+1} = k_t + s[y]$:

$$Output[t] = S[x] + k_t \ mod \ M,$$
$$Output[t] = S[y] + k_{t+1} \ mod \ M,$$

where x and y are randomly chosen indices and t = 0.

Now summing up:

$$\lfloor Output[1] \oplus Output[0] \rfloor_0 = [S[x]]_0$$

We can further formulate the following statement.

Statement 2. The probability will be

$$[S[x]]_0 = 0 = \frac{1}{2} * (1 + \frac{1}{2^{(2*n)}})$$

If S [x] is not updated at t = 0 then can be expand assumption for more than two consecutive output:

$$Output[0] = S[z] + k_t \ mod \ M$$

$$Output[1] = S[x] + k_{t+1} \ mod \ M$$

$$Output[2] = S[y] + k_{t+2} \ mod \ M$$

Algorithm:

Input: First 2 words of output suitable 2^{4*n} randomly selected secret key.

Output: To distinguish between cipher outputs and random source:

1. Generate $\mathrm{Output}^{k}[0]$ and $\mathrm{Output}^{K}[1]$
2. $S = \frac{\sum k(\lfloor \mathrm{Output}[0] \oplus \mathrm{Output}[1] \rfloor)}{2^{(2*n)}}$
3. $S \geq \frac{1}{2}$, the algorithm that was analyzed will be our RC4

Key Recovery Attack on RC4

We will try to prove that we can pick up the secret key by guessing each byte of the secret key individually and we will call three steps for recovering a key attack:

1. Guess secret key bytes individually;
2. Generate a suitable introducing differential initial variable
3. Confirm the assumption

Algorithm (For first byte of key):

Input: Two initial vectors IV1 and IV2.

Output: To distinguish between cipher outputs and random source:

1. Guess $SK[0] = \widehat{SK_0}$
2. Calculate $\lceil IV_1[0] \rceil 0 \ldots 7 = \widehat{-SK_0} \bmod 2^8$
3. Choice differential vectors $\Delta_{IV}[0] = \Delta$
4. Output keystream 2^8 words need to be generated, Output1[j] and Output2[j]

$$\begin{cases} IV_1[0] = IV_2[0] \oplus \Delta_{IV}[0] \\ IV_1[i] = IV_2[i] \ where \ 1 \leq i < 2^8 \end{cases}$$

5. The output differential vector is calculating like
$$\Delta_{Output}[j] = Output_1[j] \oplus Output_2[j]$$
6. In case when
$$\Delta_{Output}[j] = 0X00\,00\,00\,00, \overleftrightarrow{SK_0}$$
will be the least important byte of secret key with probability close to 1, else go to 1st step.

The attacks were considering weakness as:

- Non-randomness property of initial variable;
- Low diffusion property of key-scheduling algorithm and pseudo-random generation algorithm.

Conclusion:

We have considered the RC4 stream cipher and explained its cryptanalysis and gave some definitions with examples. The main goal was to follow the algorithm during of creating the program code. We also analyzed the main attacks on the RC4 cipher, and showed the algorithm for each attack and called the weakness of RC4 cipher. Moreover, we took to account that the attacks can be used only when we have access to change the input values. The base of cryptoanalysis was non-randomness of initial variable and on low diffusion property of key-scheduling algorithm and pseudo-random generation algorithm.

Code:

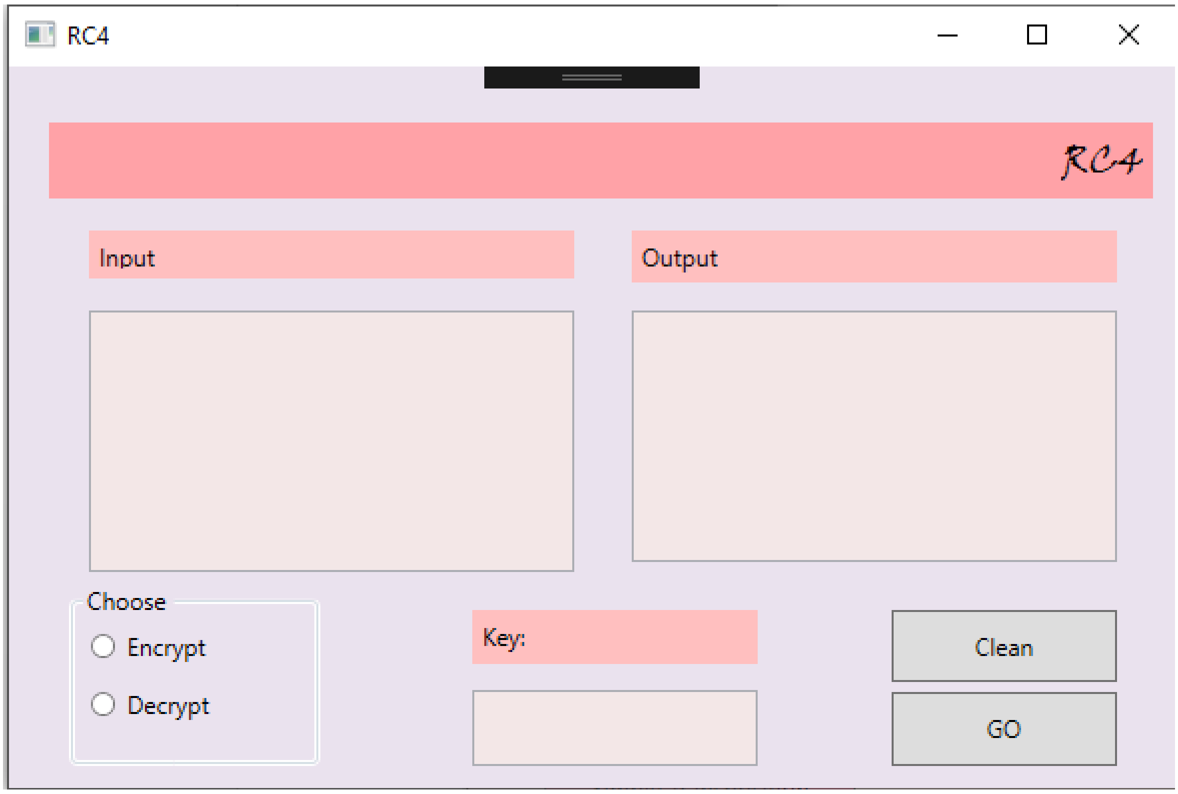
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SIW_ALLS
{
  class RCfourEng : RCfour
  {


    private const char LowerBound = 'A';
    private const char UpperBound = 'Z';
    private const int AlphabetSize = UpperBound - LowerBound + 1;

    bool m;

    public string EncrDecr(string input, string key)
    {
      StringBuilder result = new StringBuilder();
      int x, y, j = 0;
      int[] box = new int[256];
      for (int i = 0; i < 256; i++) //initialization of box
        box[i] = i;
      for (int i = 0; i < 256; i++)
      {
        j = (key[i % key.Length] + box[i] + j) % 256;
        x = box[i];
        box[i] = box[j];
        box[j] = x;
      }
      for (int i = 0; i < input.Length; i++)
```

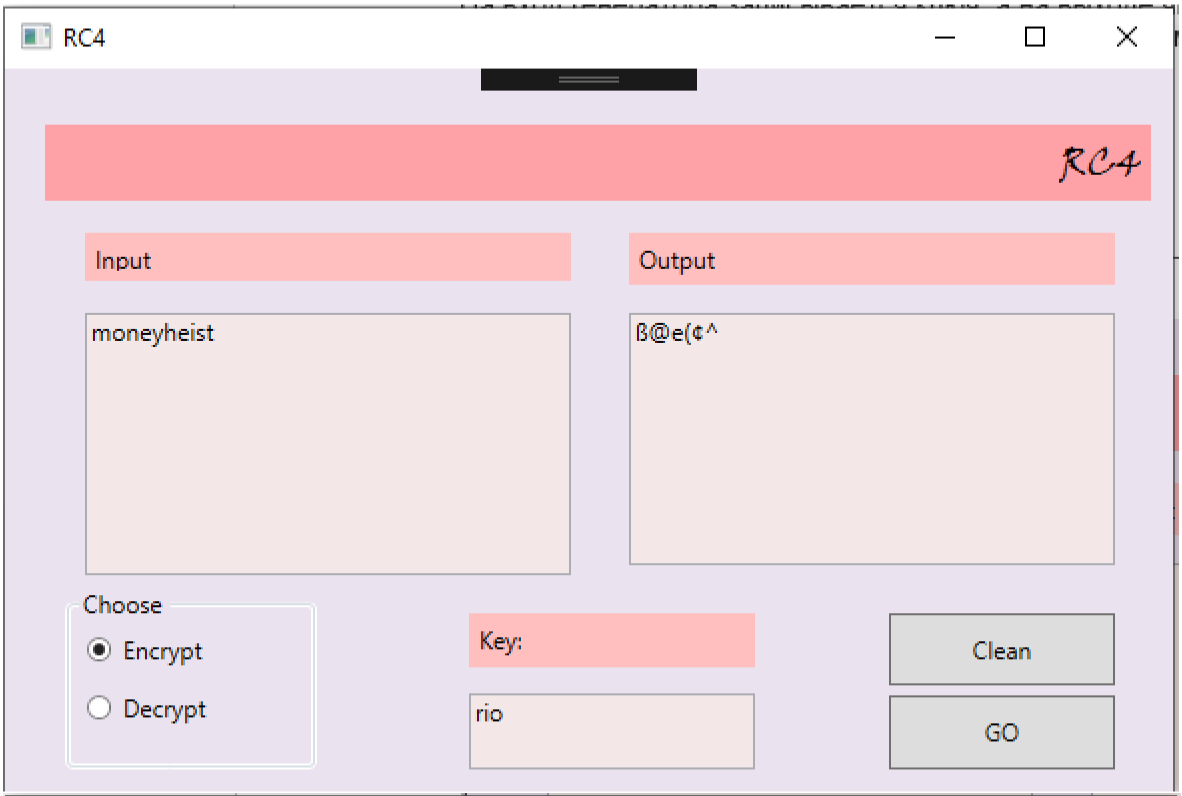
```csharp
            { //pseudo-random generation algorithm
                y = i % 256;
                j = (box[y] + j) % 256;
                x = box[y];
                box[y] = box[j];
                box[j] = x;
                result.Append((char)(input[i] ^ box[(box[y] + box[j]) % 256]));
            }
            return result.ToString();
        }



    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SIW_ALLS
{
    interface RCfour
    {
        string EncrDecr(string input, string key);

    }
}
```

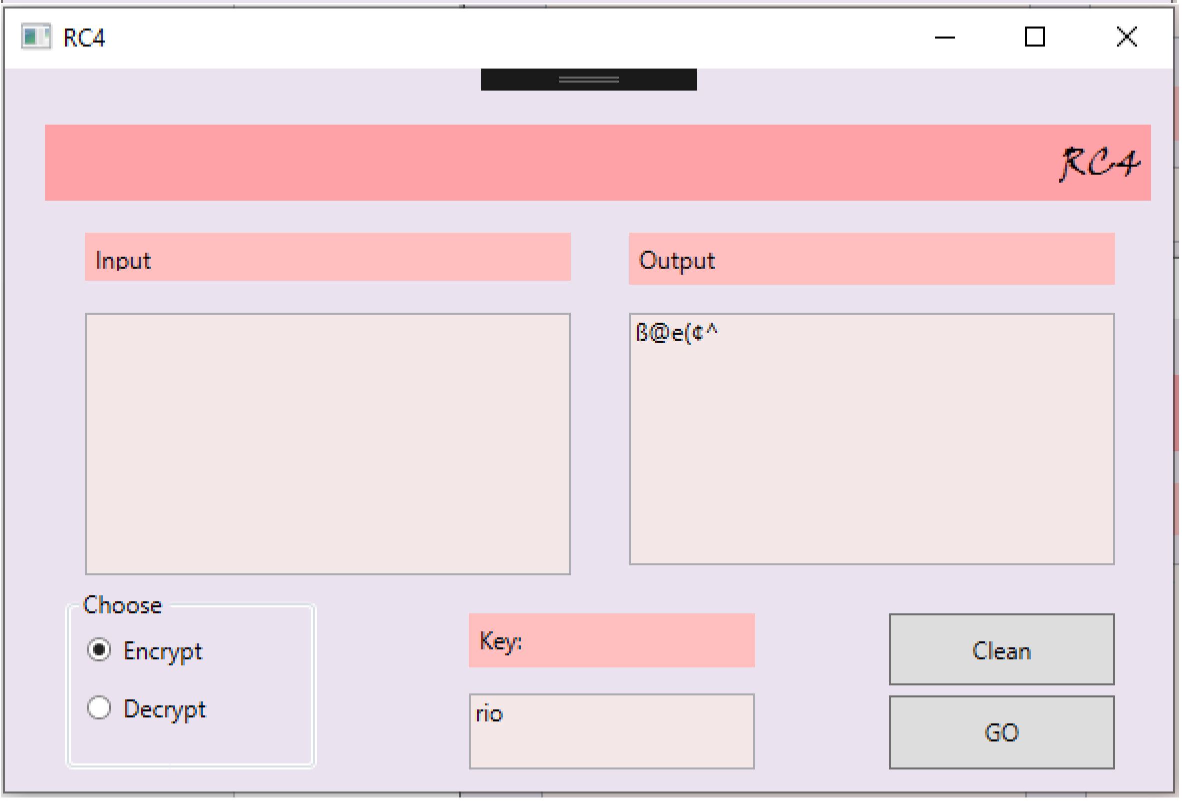
```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace SIW_ALLS
{
    /// <summary>
    /// Interaction logic for RC4.xaml
    /// </summary>
    public partial class RC4 : Window
    {
        public RC4()
        {
            InitializeComponent();
        }
        RCfour rc4 = new RCfourEng();
        private int wrong = 0;
        private void Go_Click(object sender, RoutedEventArgs e)
        {
            if ((bool)Encrypt.IsChecked)
            {
                Output.Text = rc4.EncrDecr(Input.Text, Key.Text);
            }
```

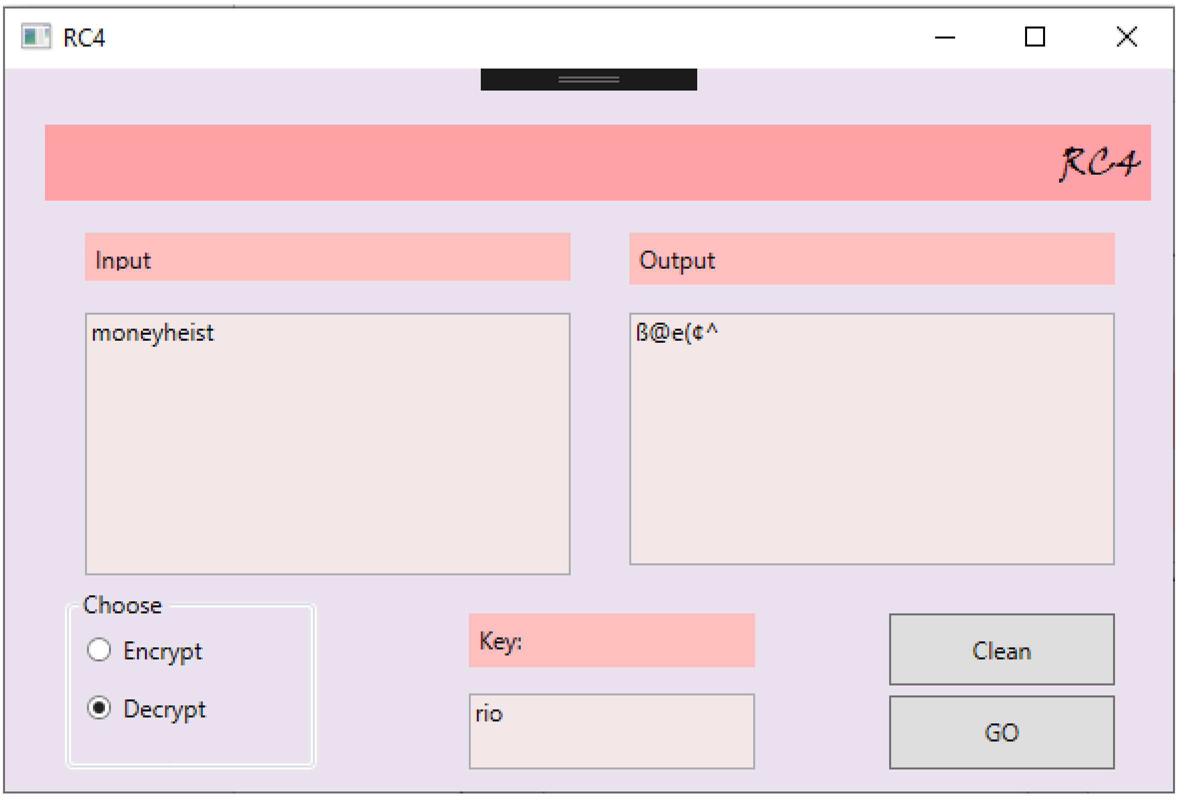
```
            else if ((bool)Decrypt.IsChecked)
            {
                Input.Text = rc4.EncrDecr(Output.Text, Key.Text);
            }

        }

        private void Clean_Click(object sender, RoutedEventArgs e)
        {

            Input.Text = "";
        }
    }
}
```

Results:

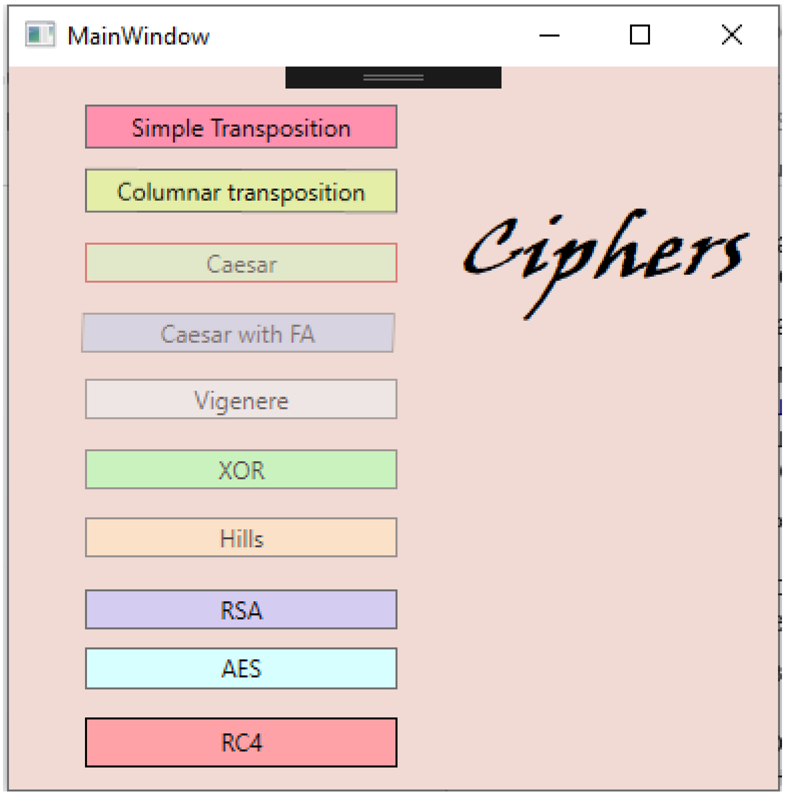



RC4
RC4
Input
Output
moneyheist
ß@e(¢^
Choose
Encrypt
Decrypt
Key:
rio
Clean
GO

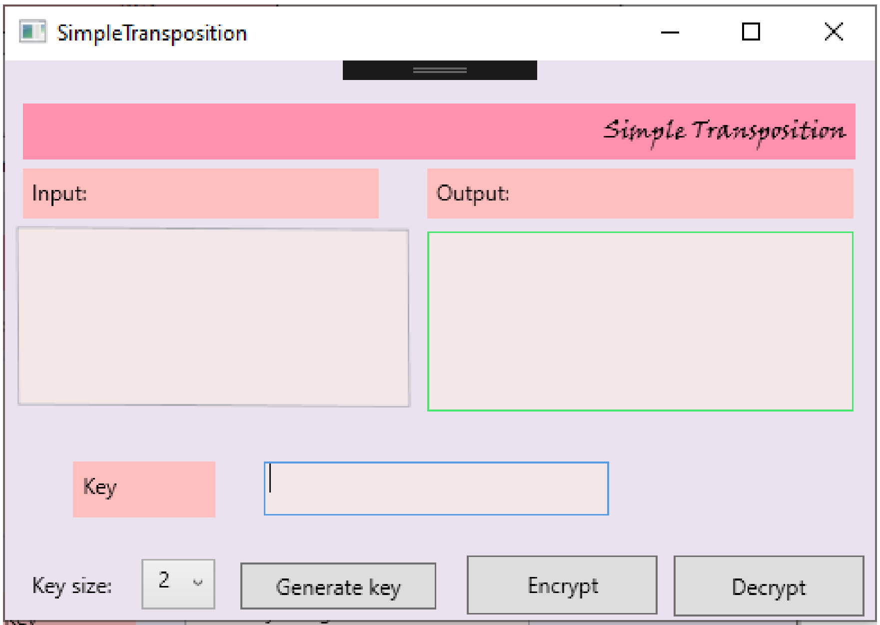

RC4
RC4
Input
Output
ß@e(¢^
Choose
Encrypt
Decrypt
Key:
rio
Clean
GO

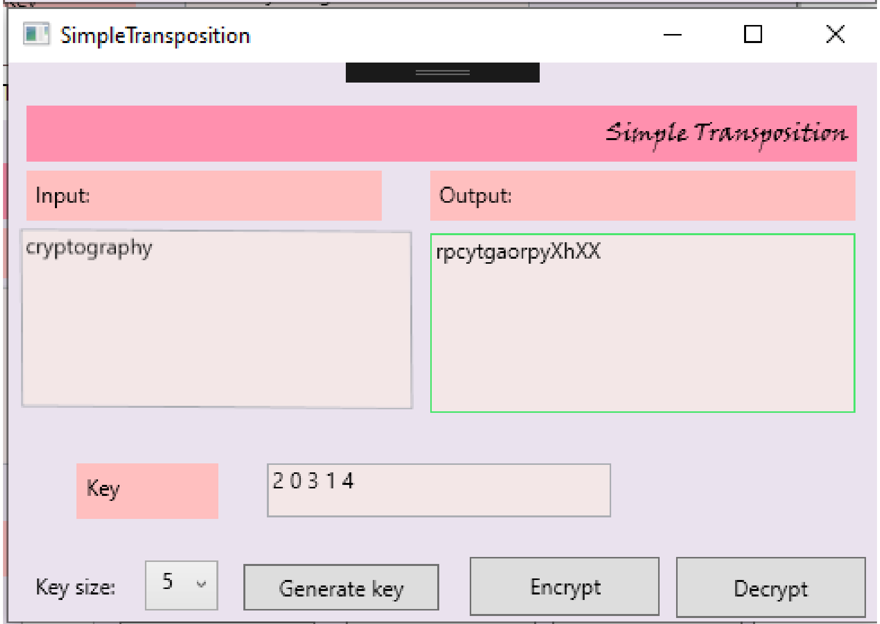


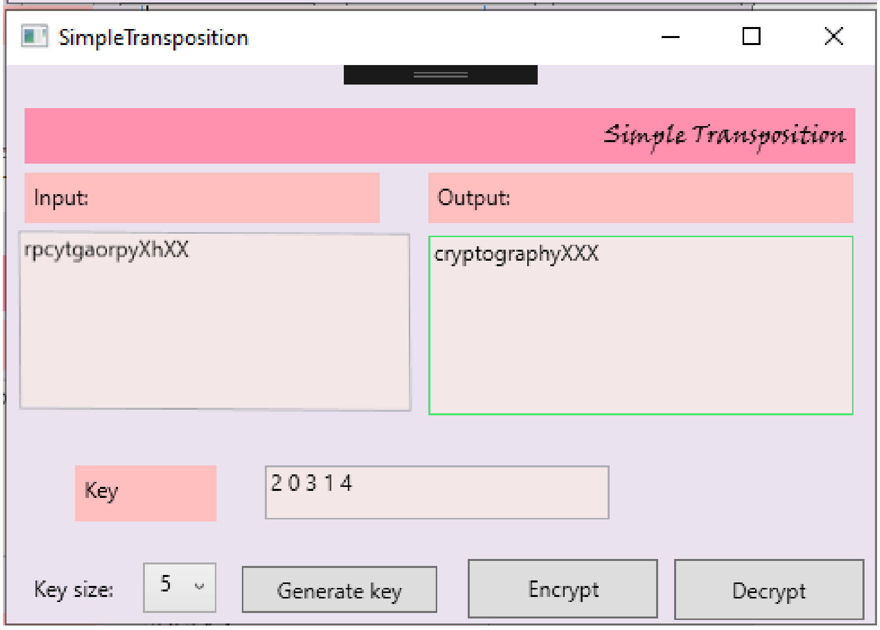
Other ciphers:



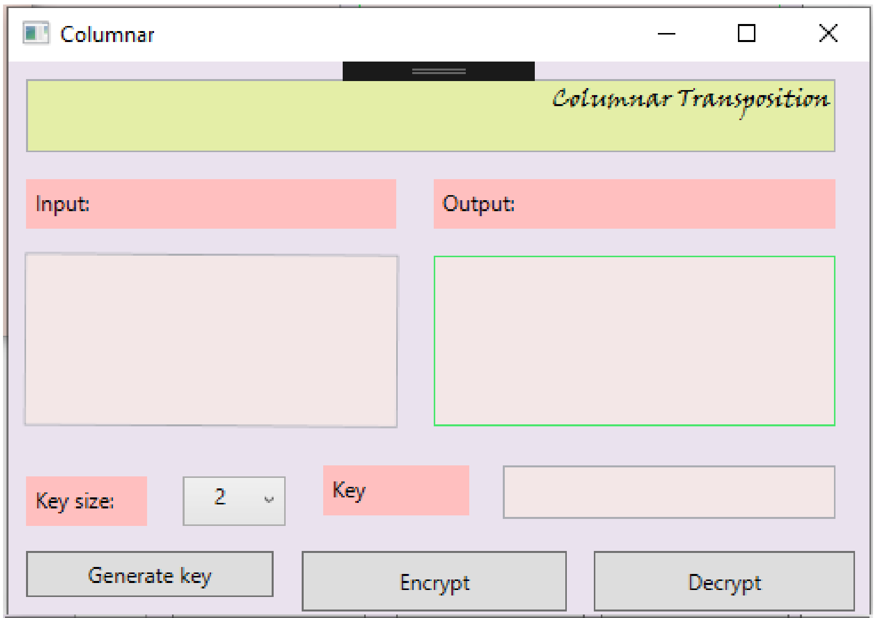
a. Simple transposition

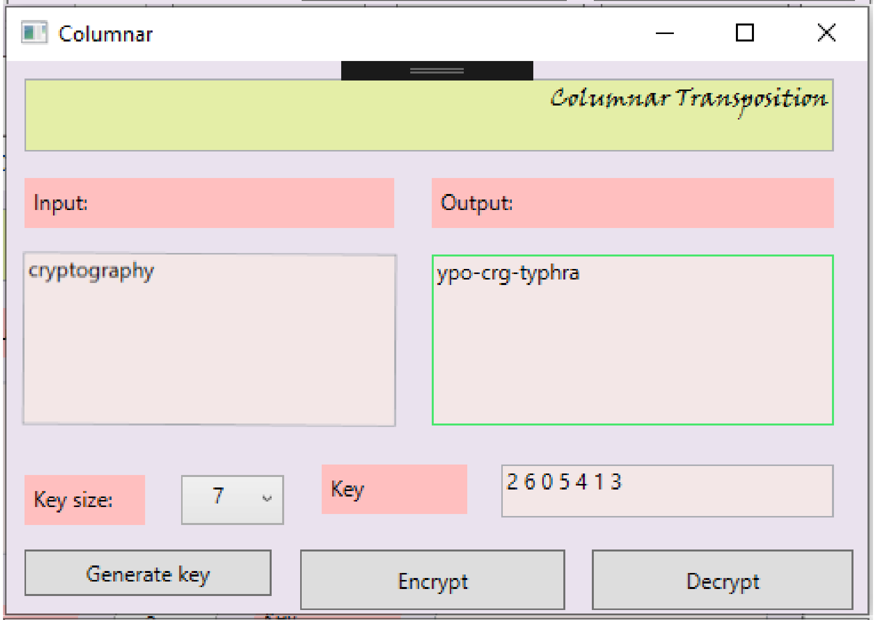


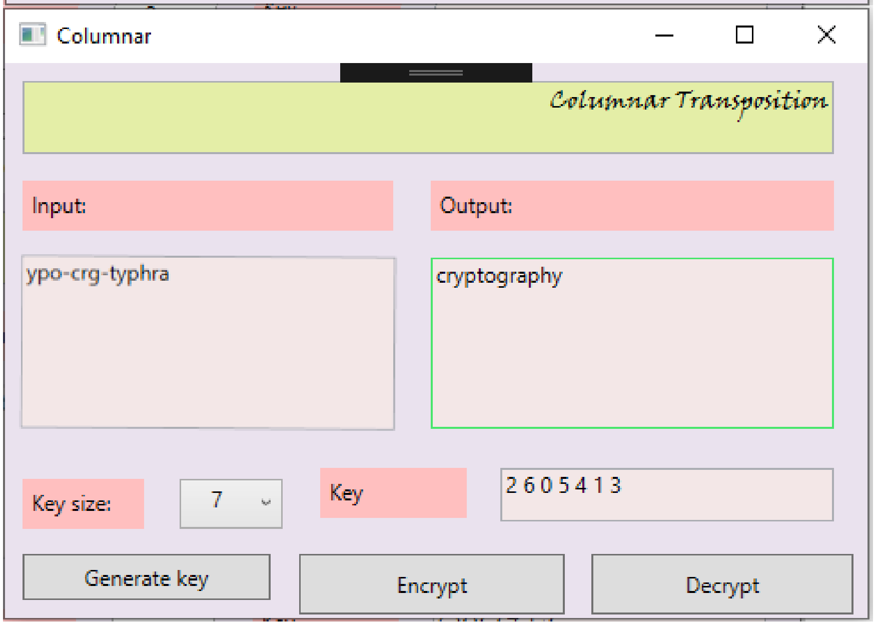




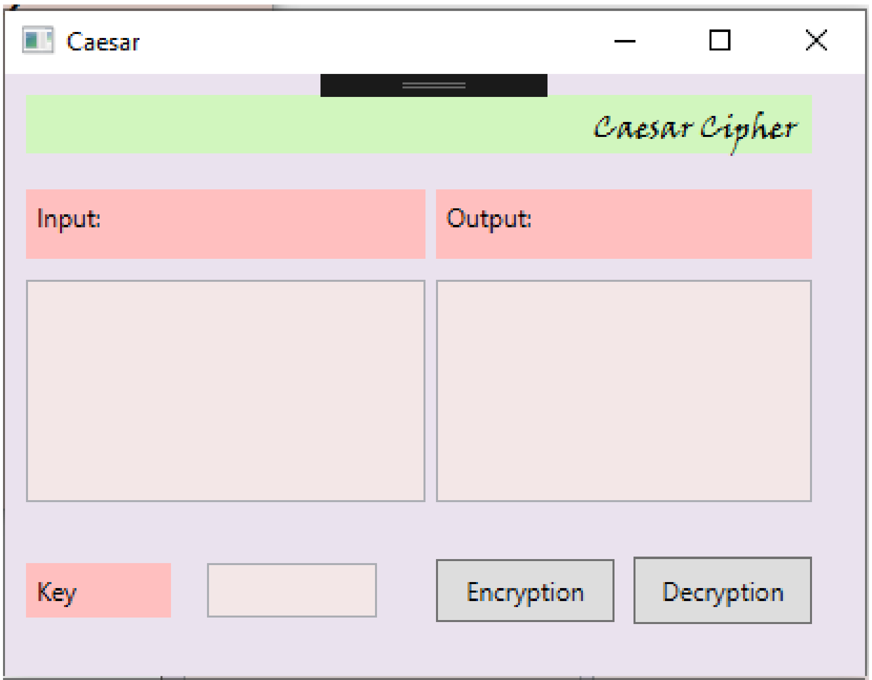
b. Columnar transposition

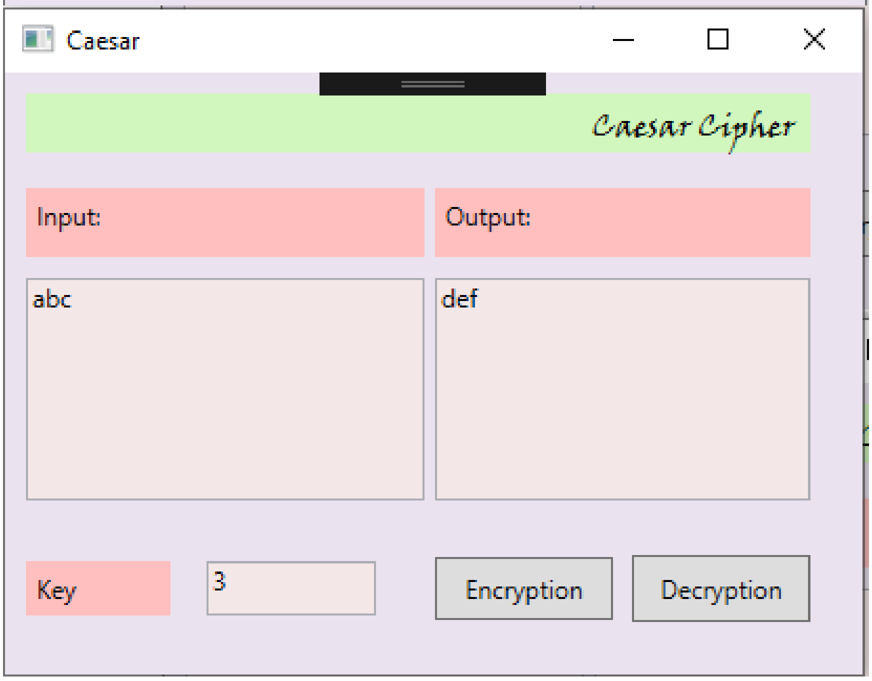


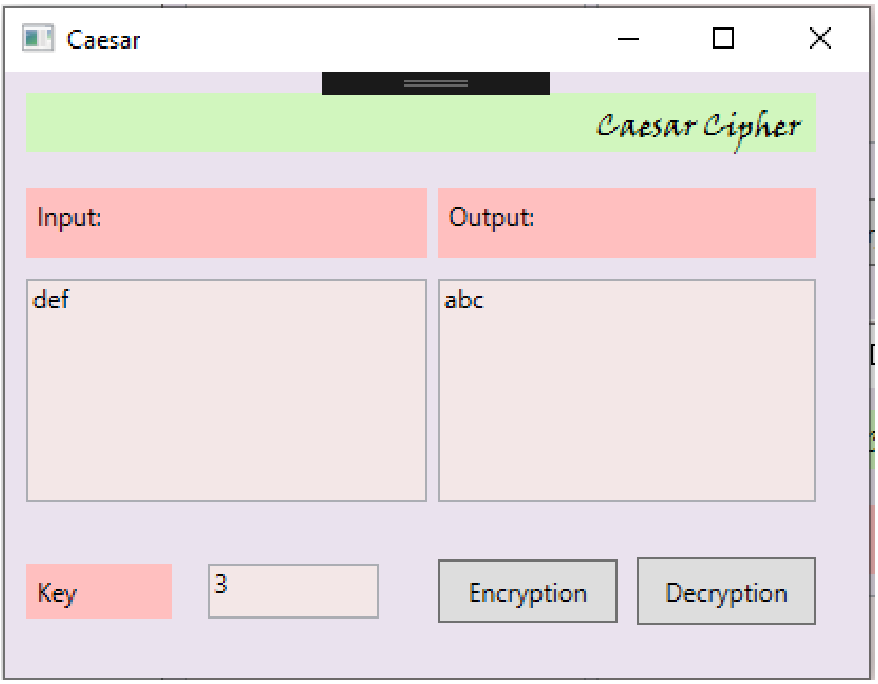




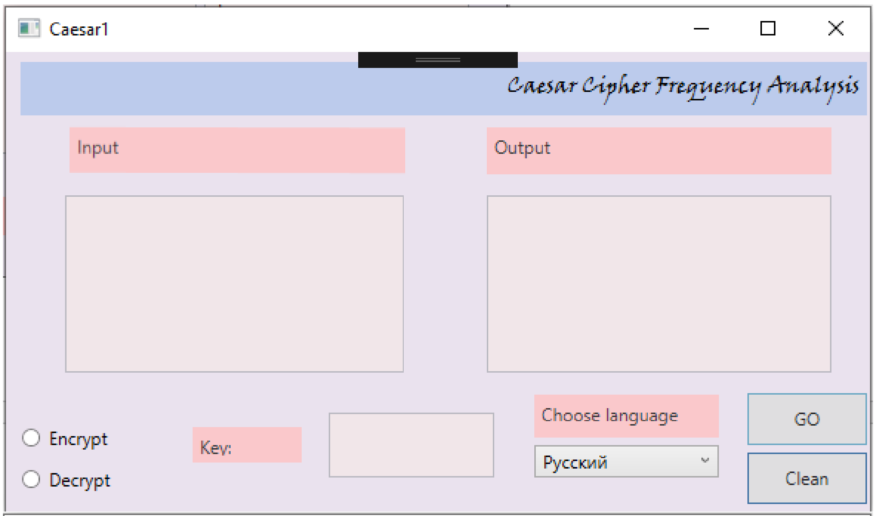
c. Caesar

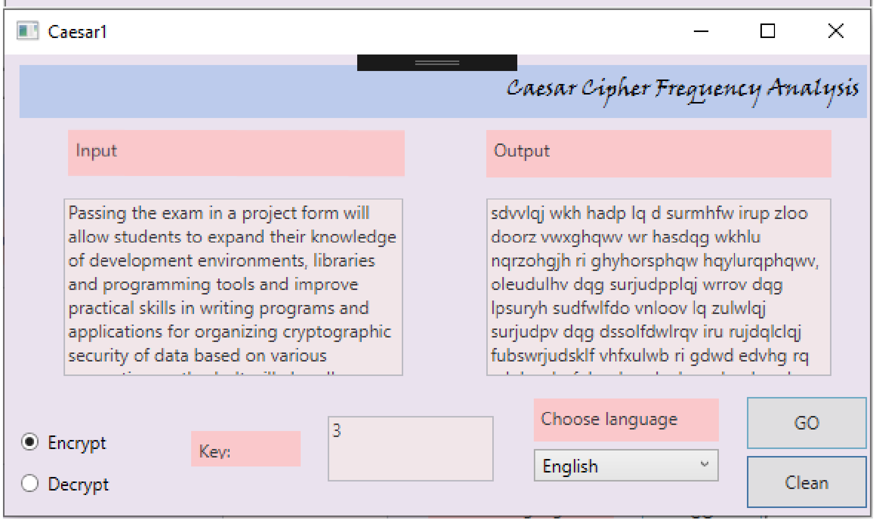

Caesar
Caesar Cipher
Input:
Output:
Key
Encryption
Decryption

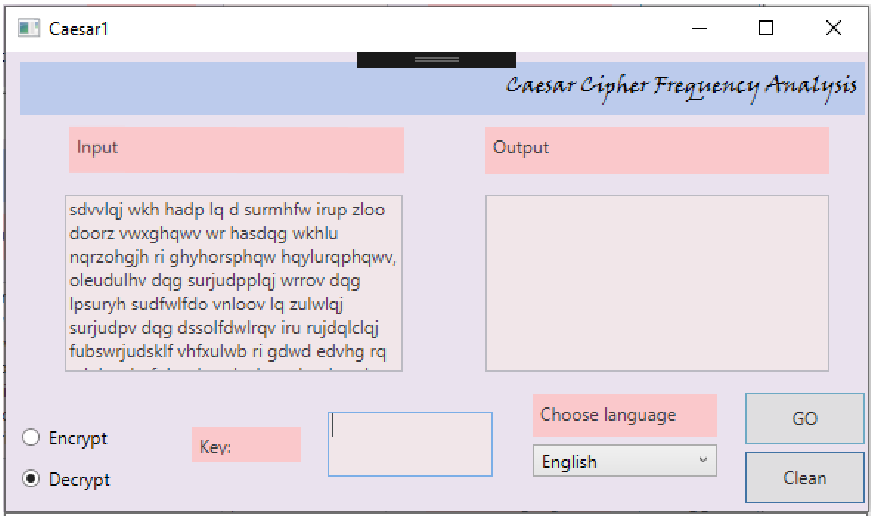

Caesar
Caesar Cipher
Input:
Output:
abc
def
Key
3
Encryption
Decryption

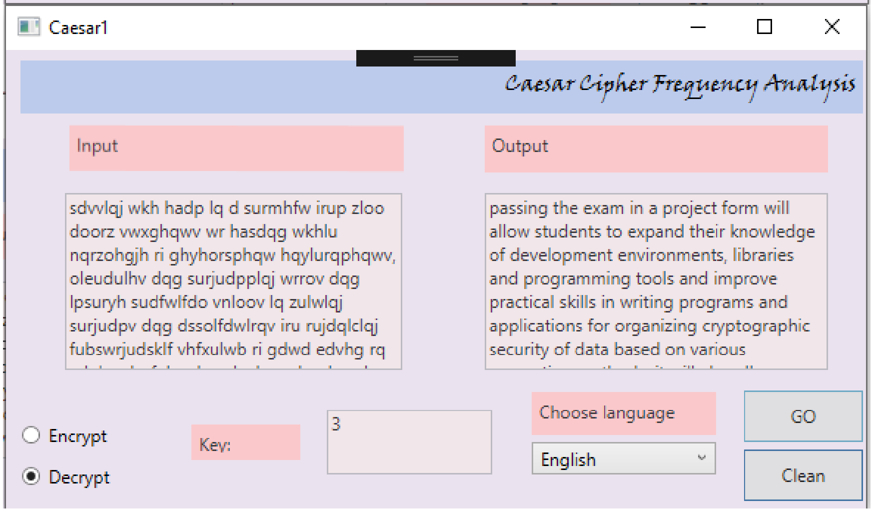


d. Caesar Frequent Analysis

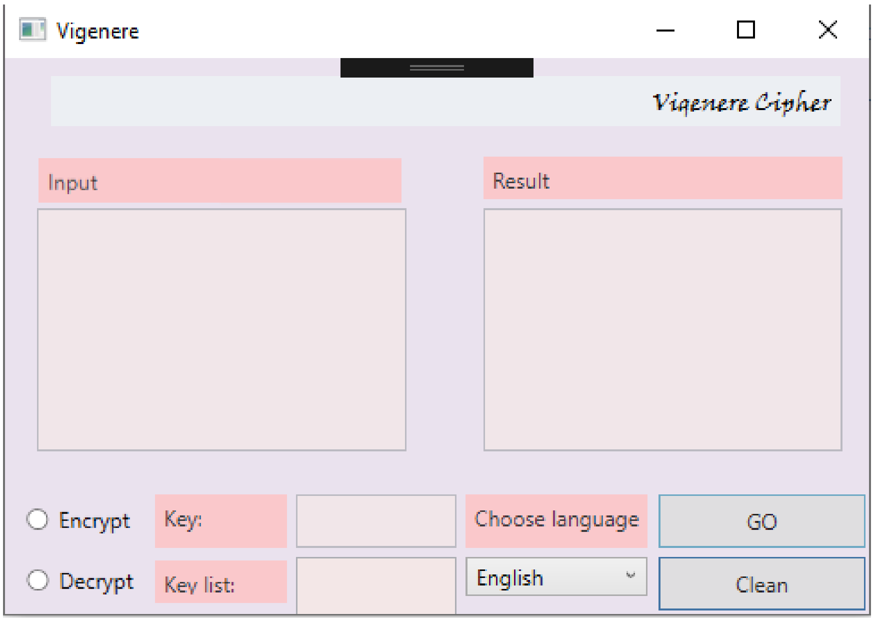


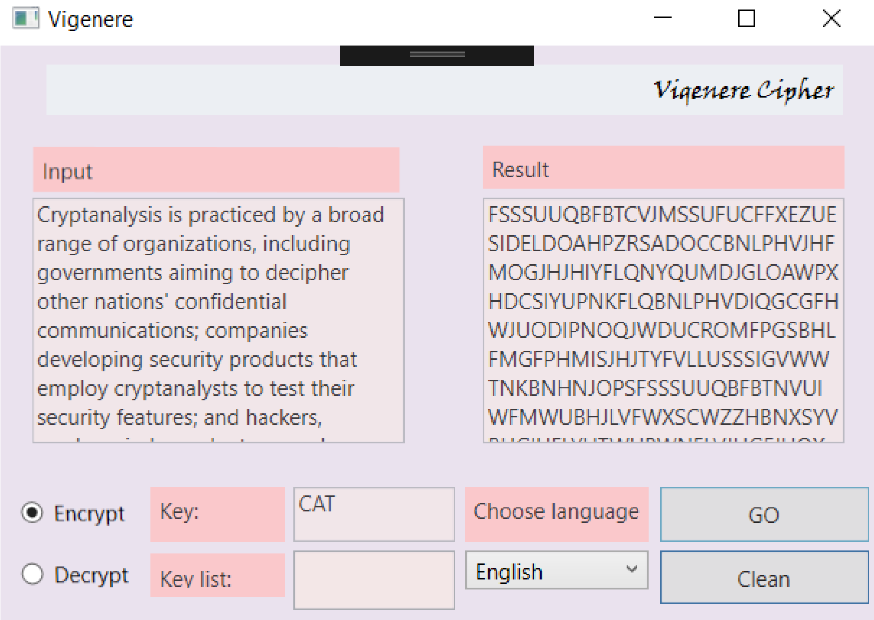


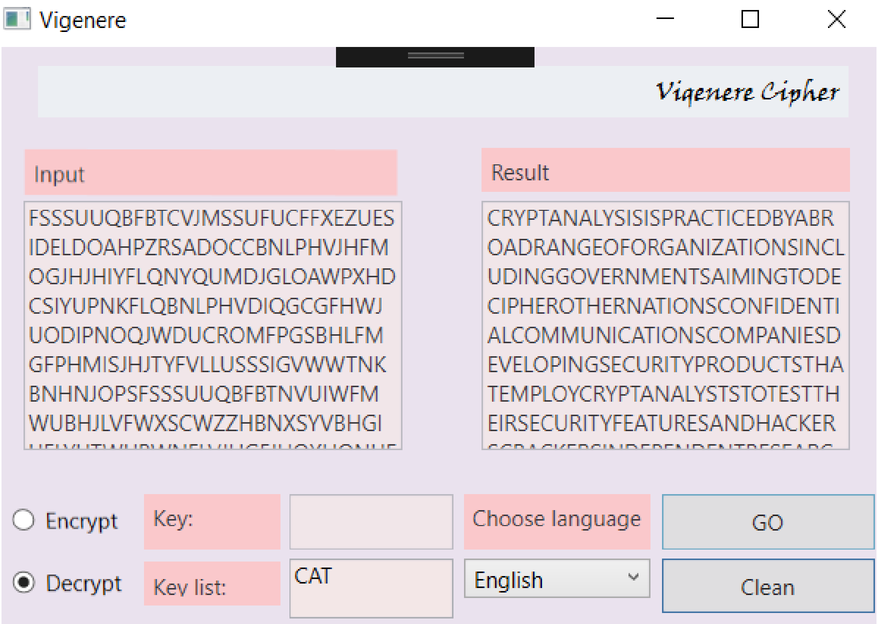




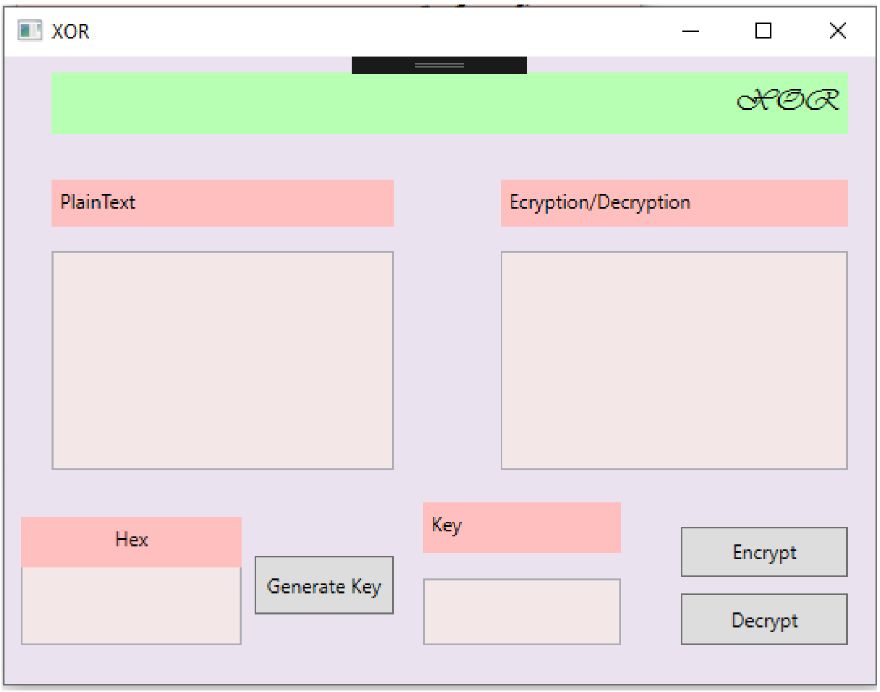
e. Vigenere with frequency analysis

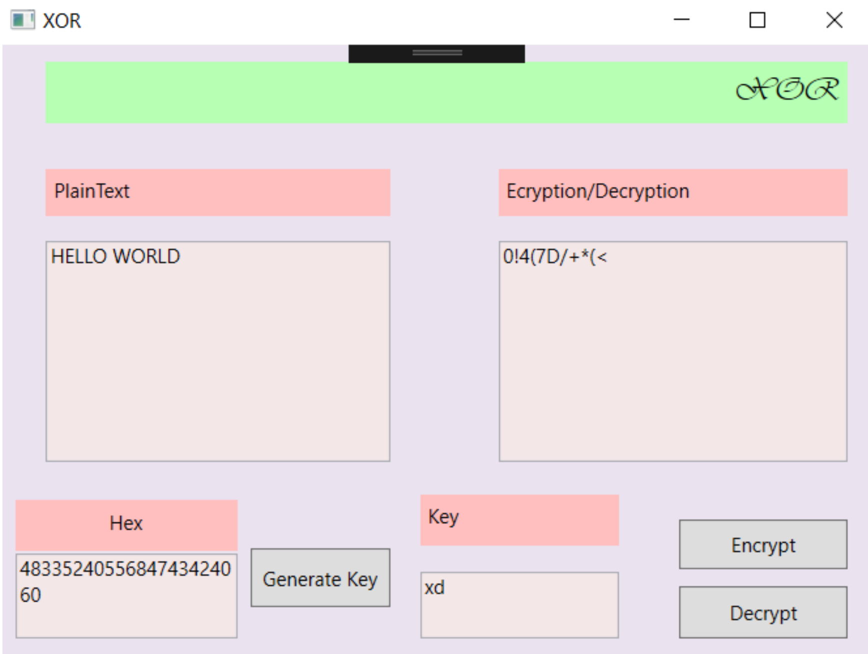



Vigenere
Vigenere Cipher
Input
Cryptanalysis is practiced by a broad
range of organizations, including
governments aiming to decipher
other nations' confidential
communications; companies
developing security products that
employ cryptanalysts to test their
security features; and hackers,
Result
FSSSUUQBFBTCVJMSSUFUCFFXEZUE
SIDELDOAHPZRSADOCCBNLPHVJHF
MOGJHJHIYFLQNYQUMDJGLOAWPX
HDCSIYUPNKFLQBNLPHVDIQGCGFH
WJUODIPNOQJWDUCROMFPGSBHL
FMGFPHMISJHJTYFVLLUSSSIGVWW
TNKBNHNJOPSFSSSUUQBFBTNVUI
WFMWUBHJLVFWXSCWZZHBNXSYV
Encrypt
Decrypt
Key:
CAT
Key list:
Choose language
English
GO
Clean

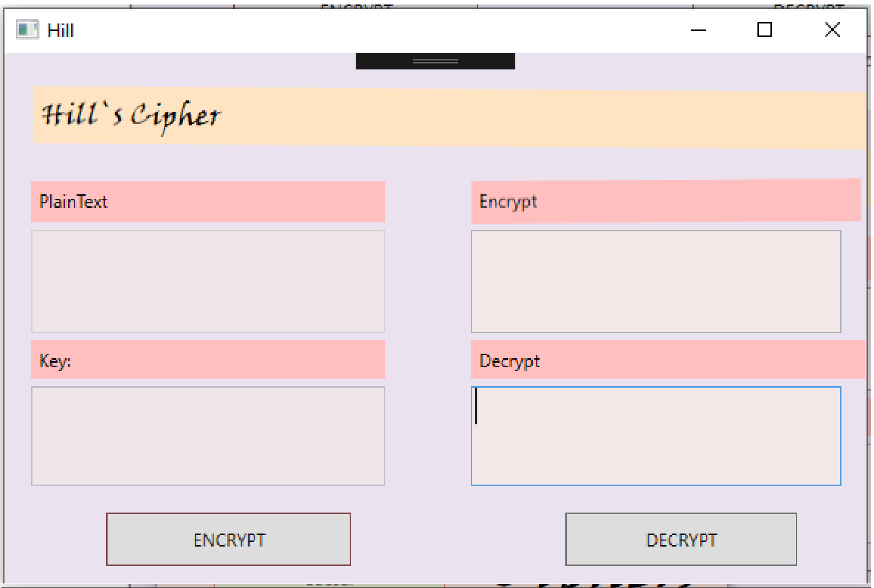

Vigenere
Vigenere Cipher
Input
FSSSUUQBFBTCVJMSSUFUCFFXEZUES
IDELDOAHPZRSADOCCBNLPHVJHFM
OGJHJHIYFLQNYQUMDJGLOAWPXHD
CSIYUPNKFLQBNLPHVDIQGCGFHWJ
UODIPNOQJWDUCROMFPGSBHLFM
GFPHMISJHJTYFVLLUSSSIGVWWTNK
BNHNJOPSFSSSUUQBFBTNVUIWFM
WUBHJLVFWXSCWZZHBNXSYVBHGI
Result
CRYPTANALYSISISPRACTICEDBYABR
OADRANGEOFORGANIZATIONSINCL
UDINGGOVERNMENTSAIMINGTODE
CIPHEROTHERNATIONSCONFIDENTI
ALCOMMUNICATIONSCOMPANIESD
EVELOPINGSECURITYPRODUCTSTHA
TEMPLOYCRYPTANALYSTSTOTESTTH
EIRSECURITYFEATURESANDHACKER
Encrypt
Decrypt
Key:
Key list:
CAT
Choose language
English
GO
Clean

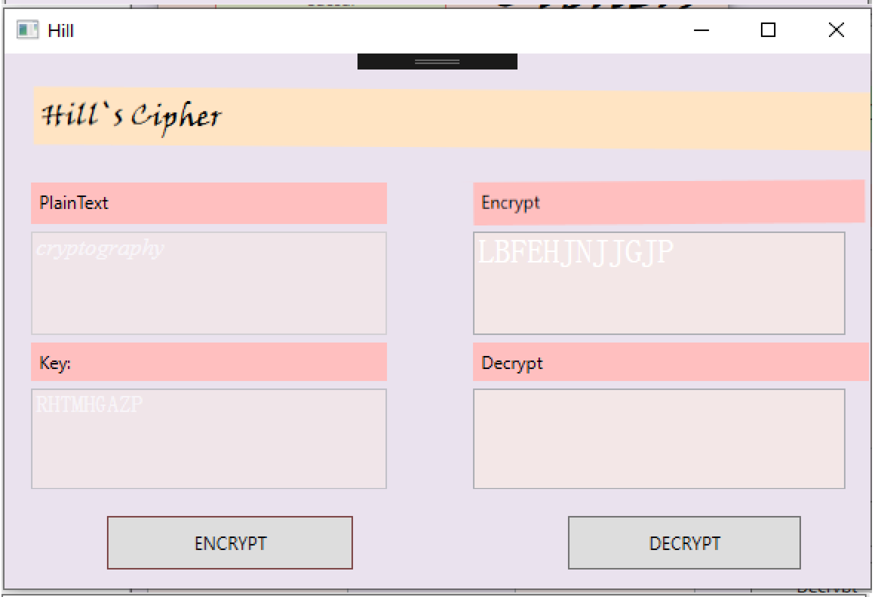
f. XOR

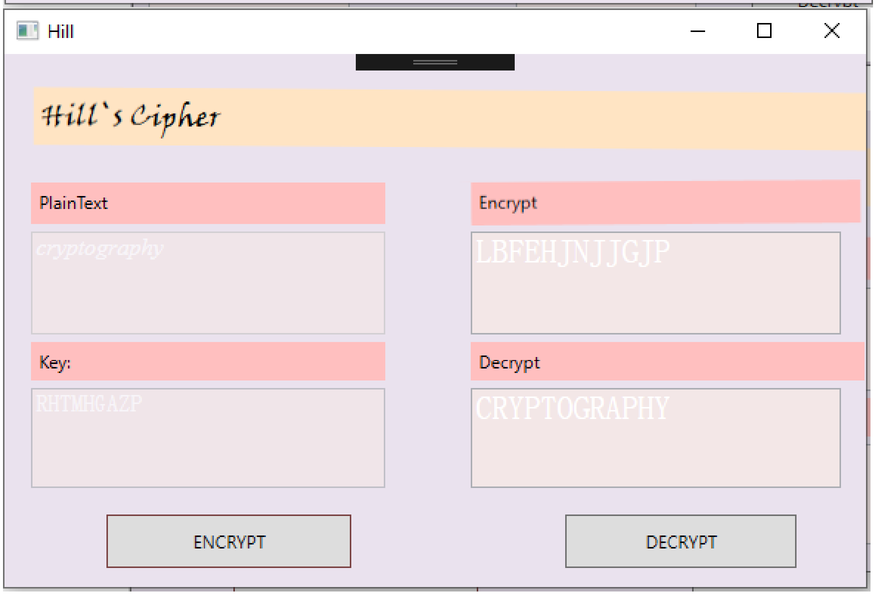

XOR
XOR
PlainText
Ecryption/Decryption
Hex
Generate Key
Key
Encrypt
Decrypt


XOR
XOR
PlainText
Ecryption/Decryption
HELLO WORLD
0!4(7D/+*(<
Hex
48335240556847434240
60
Generate Key
Key
xd
Encrypt
Decrypt

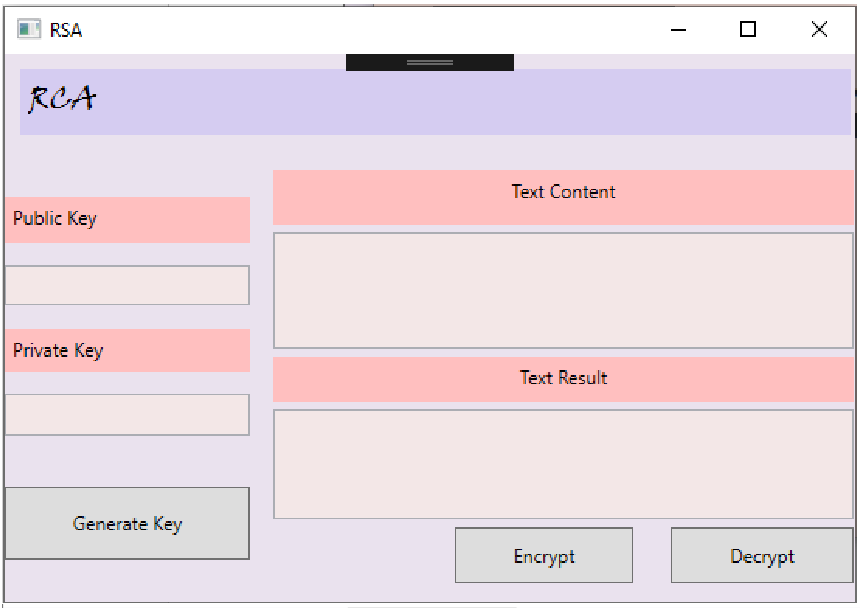
g. Hill

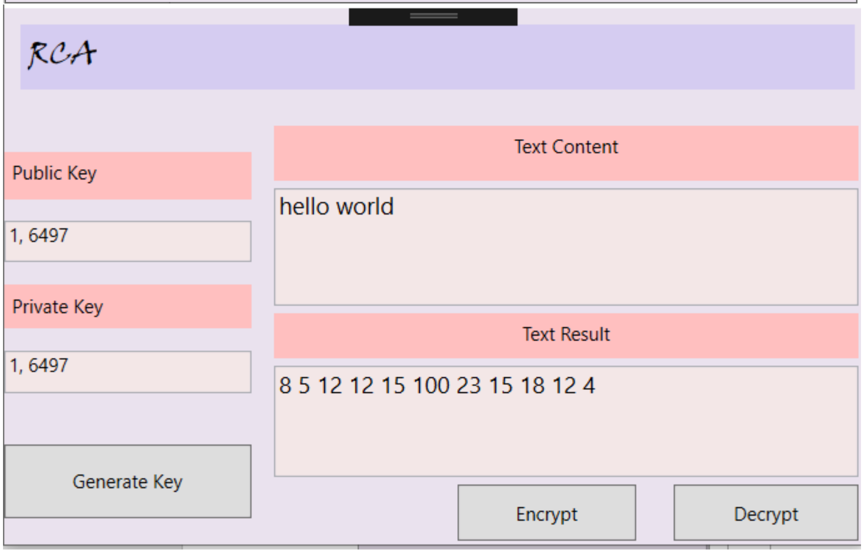

Hill
Hill`s Cipher
PlainText
Encrypt
Key:
Decrypt
ENCRYPT
DECRYPT

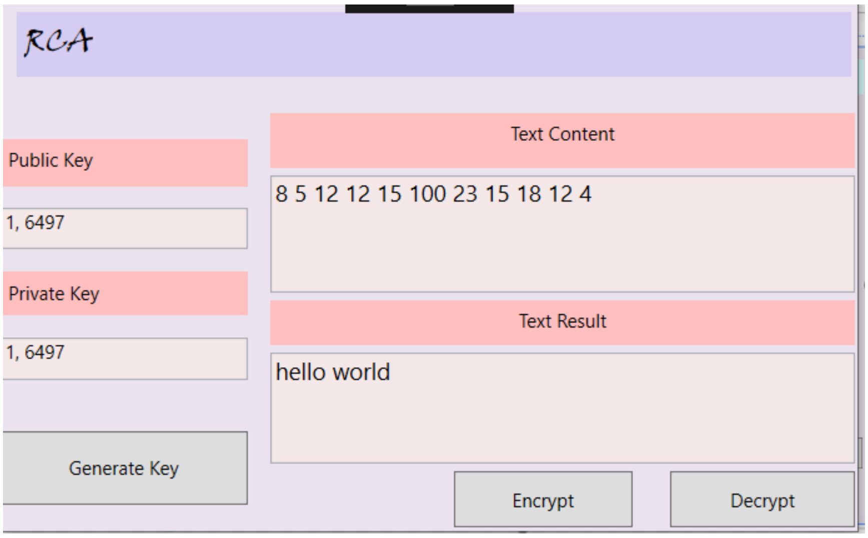

Hill
Hill`s Cipher
PlainText
cryptography
Encrypt
LBFEHJNJJGJP
Key:
RHTMHGAZP
Decrypt
ENCRYPT
DECRYPT


Hill
Hill`s Cipher
PlainText
cryptography
Encrypt
LBFEHJNJJGJP
Key:
RHTMHGAZP
Decrypt
CRYPTOGRAPHY
ENCRYPT
DECRYPT

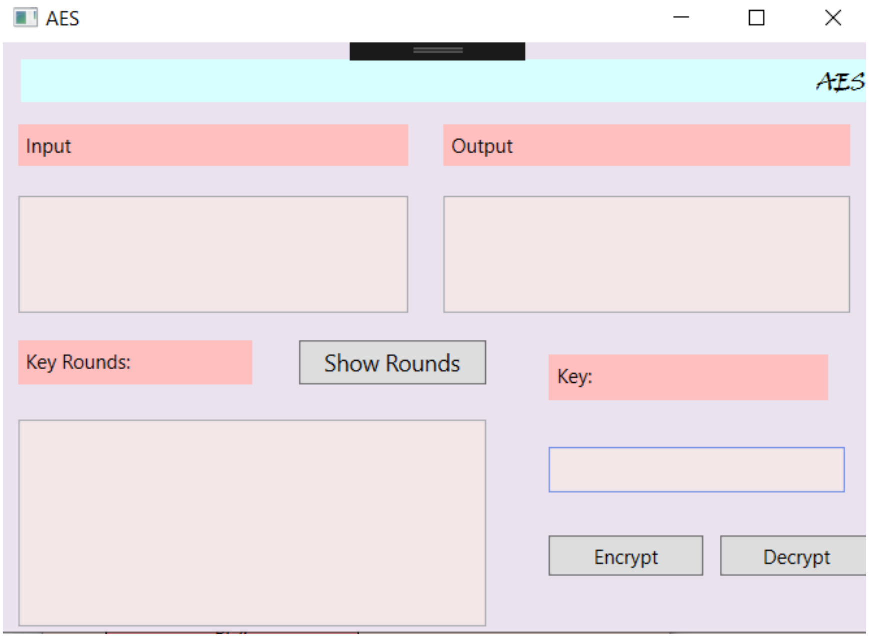
h. RSA

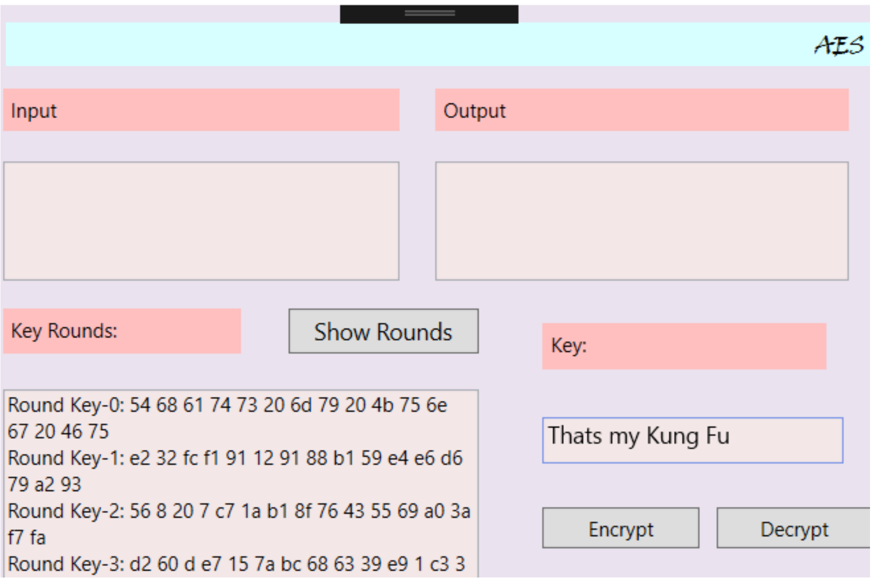

RSA
RCA
Text Content
Public Key
Private Key
Text Result
Generate Key
Encrypt
Decrypt

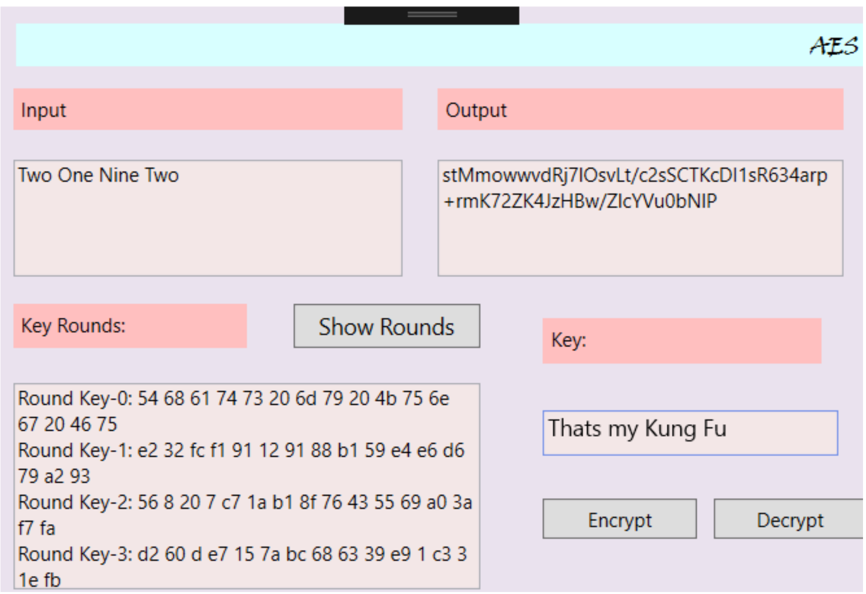

RCA
Text Content
Public Key
hello world
1, 6497
Private Key
Text Result
1, 6497
8 5 12 12 15 100 23 15 18 12 4
Generate Key
Encrypt
Decrypt

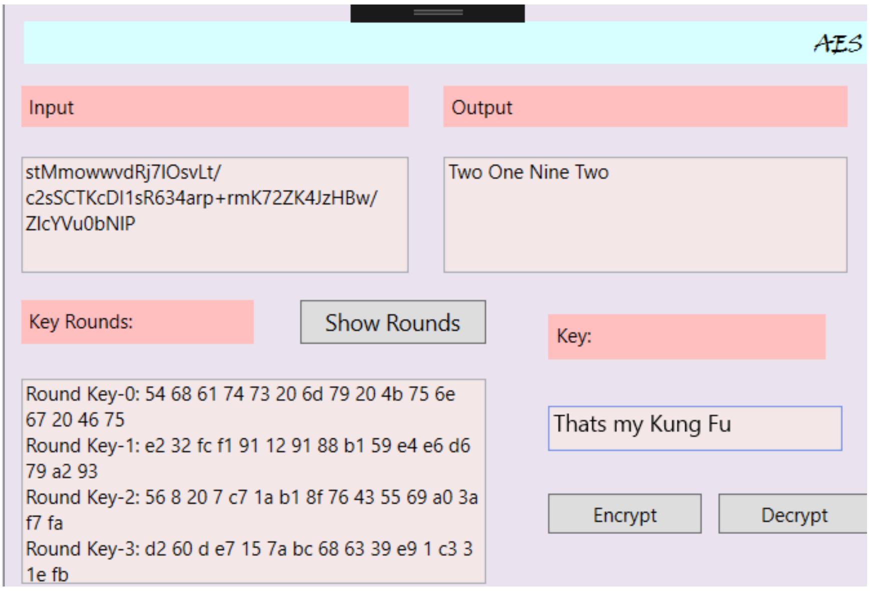

RCA
Text Content
Public Key
8 5 12 12 15 100 23 15 18 12 4
1, 6497
Private Key
Text Result
1, 6497
hello world
Generate Key
Encrypt
Decrypt

i. AES


AES
AES
Input
Output
Key Rounds:
Show Rounds
Key:
Encrypt
Decrypt


AES
Input
Output
Key Rounds:
Show Rounds
Key:
Round Key-0: 54 68 61 74 73 20 6d 79 20 4b 75 6e 67 20 46 75
Round Key-1: e2 32 fc f1 91 12 91 88 b1 59 e4 e6 d6 79 a2 93
Round Key-2: 56 8 20 7 c7 1a b1 8f 76 43 55 69 a0 3a f7 fa
Round Key-3: d2 60 d e7 15 7a bc 68 63 39 e9 1 c3 3
Thats my Kung Fu
Encrypt
Decrypt


AES
Input
Output
Two One Nine Two
stMmowwvdRj7IOsvLt/c2sSCTKcDI1sR634arp +rmK72ZK4JzHBw/ZlcYVu0bNIP
Key Rounds:
Show Rounds
Key:
Round Key-0: 54 68 61 74 73 20 6d 79 20 4b 75 6e 67 20 46 75
Round Key-1: e2 32 fc f1 91 12 91 88 b1 59 e4 e6 d6 79 a2 93
Round Key-2: 56 8 20 7 c7 1a b1 8f 76 43 55 69 a0 3a f7 fa
Round Key-3: d2 60 d e7 15 7a bc 68 63 39 e9 1 c3 3 1e fb
Thats my Kung Fu
Encrypt
Decrypt



References:

1. Orumiehchiha, Mohammad Ali, et al. "Cryptanalysis of RC4(n, m) Stream Cipher." *178.Pdf*, 2013, eprint.iacr.org/2013/178.pdf.