

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.

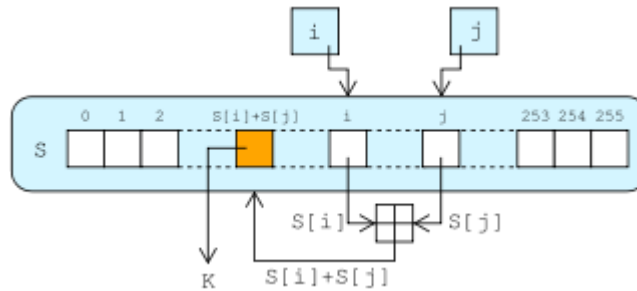
RC4 and its cryptosystem

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 give our 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,
$$j = (\text{key}[i \% \text{key.Length}] + \text{box}[i] + j) \% 256$$
will be performed, then just swap $\text{box}[i]$ and $\text{box}[j]$.
4. Next step is “pseudo-random generation algorithm”. Here in one loop is defining one n-bits text from the keystream, then just swap $\text{box}[y]$ and $\text{box}[j]$. 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. $S[i] \pmod{2^n} = 1 - K[i \pmod{1}]$ and $S[0] \pmod{2^n} = -K[0]$.
So, $j=i$. It means: after 1st 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.
 $S[i] \pmod{2^n} = (1 - K[2])(i \pmod{1})$ $3 < i < 255$, it means that states of interval 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

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

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 $S[i] + S[j] \pmod{M}$ 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:

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

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:

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

$$\begin{aligned} \text{Output}[t] &= S[x] + k_t \bmod M, \\ \text{Output}[t] &= S[y] + k_{t+1} \bmod M, \end{aligned}$$

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

Now summing up:

$$[\text{Output}[1] \oplus \text{Output}[0]]_0 = [S[x]]_0$$

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:

$$\text{Output}[0] = S[z] + k_t \bmod M$$

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

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

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 $\text{Output}^k[0]$ and $\text{Output}^k[1]$
2. $S = \frac{\sum k(|\text{Output}[0] \oplus \text{Output}[1]|)}{2^{(2*n)}}$
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 $\text{SK}[0] = \widehat{SK}_0$
2. Calculate $[\text{IV}_1[0]]_0 \dots 7 = \widehat{-SK}_0 \bmod 2^8$
3. Choice differential vectors $\Delta_{IV}[0] = \Delta$
4. Output keystream 2^8 words need to be generated, $\text{Output1}[j]$ and $\text{Output2}[j]$

$$\begin{cases} \text{IV}_1[0] = \text{IV}_2[0] \oplus \Delta_{IV}[0] \\ \text{IV}_1[i] = \text{IV}_2[i] \text{ where } 1 \leq i < 2^8 \end{cases}$$

5. The output differential vector is calculating like

$$\Delta_{Output}[j] = Output_1[j] \oplus Output_2[j]$$

6. In case when

$$\Delta_{Output}[j] = 0X00\ 00\ 00\ 00, \widehat{SK_0}$$

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 cryptanalysis 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++)
```

```

        { //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);
    }
}

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:

The screenshot shows a Windows application window titled "RC4". The interface is designed for performing RC4 encryption and decryption. At the top, there is a red header bar with the text "RC4" in a stylized font. Below this, the interface is divided into two main sections: "Input" and "Output", each with a corresponding text area. At the bottom, there is a "Choose" section with two radio buttons: "Encrypt" and "Decrypt". To the right of the "Choose" section is a "Key:" label followed by a text input field. Further right are two buttons: "Clean" and "GO".

RC4

RC4

Input

moneyheist

Output

B@e(\$^

Choose

☒ Encrypt

☐ Decrypt

Key:

rio

Clean

GO

RC4

RC4

Input

Output

B@e(\$^

Choose

☒ Encrypt

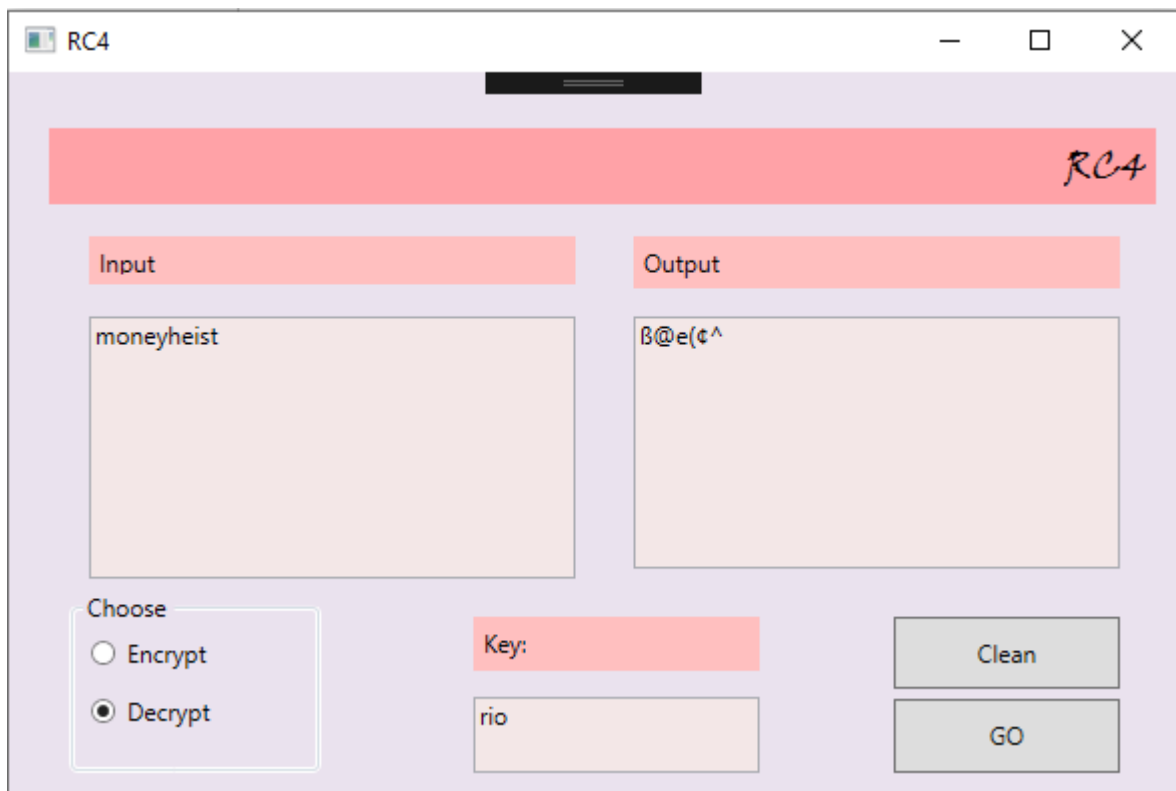
☐ Decrypt

Key:

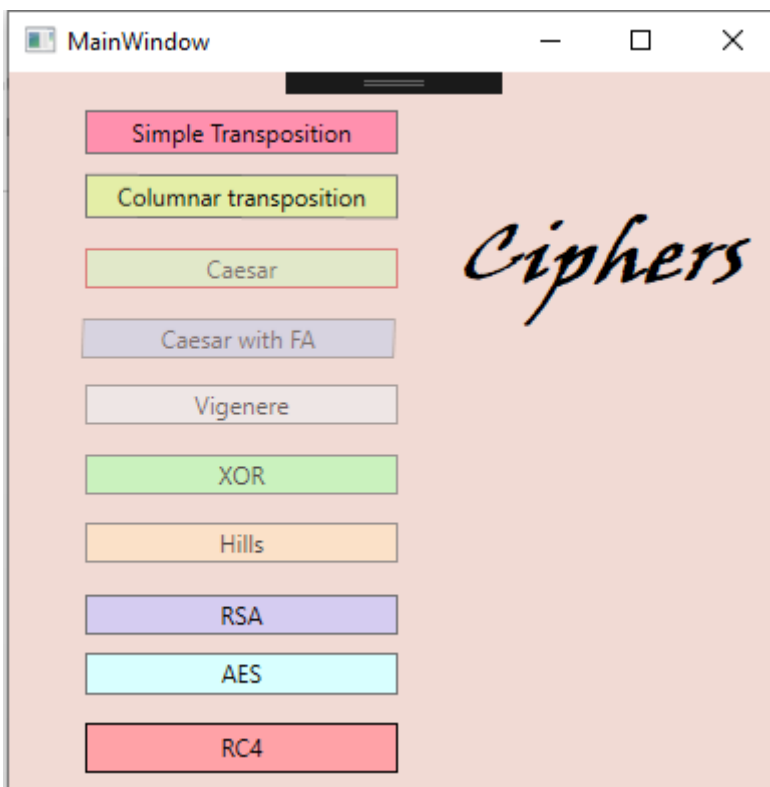
rio

Clean

GO



Other ciphers:



a. Simple transposition

The image displays three sequential screenshots of a software application titled "SimpleTransposition". The application window has a title bar with standard minimize, maximize, and close buttons. The main interface is divided into several sections:

- Header:** A pink bar with the text "Simple Transposition" in a cursive font.
- Input/Output Labels:** Two orange labels, "Input:" and "Output:", are positioned above their respective text areas.
- Text Areas:** Two large, light-colored rectangular boxes for entering and displaying text.
- Key Section:** An orange label "Key" is followed by a text input field.
- Controls:** A "Key size:" dropdown menu (currently set to "2"), a "Generate key" button, and "Encrypt" and "Decrypt" buttons.

The three screenshots illustrate the workflow:

- Initial State:** The "Input:" field is empty, the "Output:" field is empty, the "Key" field is empty, and the "Key size:" dropdown is set to "2".
- Encryption:** The "Input:" field contains the text "cryptography". The "Key" field contains the sequence "2 0 3 1 4". The "Key size:" dropdown is now set to "5". The "Output:" field displays the encrypted result: "rpcytgaorpyXhXX".
- Decryption:** The "Input:" field contains the encrypted text "rpcytgaorpyXhXX". The "Key" field still contains "2 0 3 1 4" and the "Key size:" dropdown remains at "5". The "Output:" field displays the decrypted result: "cryptographyXXX".

b. Columnar transposition

The image displays three sequential screenshots of a software application titled "Columnar Transposition". Each window has a title bar with standard minimize, maximize, and close buttons. The interface includes a header area with the title "Columnar Transposition", two input/output text areas, a key generation section, and three action buttons: "Generate key", "Encrypt", and "Decrypt".

First Screenshot: The application is in its initial state. The "Input:" text area is empty. The "Output:" text area is empty. The "Key size:" is set to 2, and the "Key" field is empty.

Second Screenshot: The "Input:" text area contains the text "cryptography". The "Output:" text area contains the encrypted text "ypo-crg-typhra". The "Key size:" is set to 7, and the "Key" field contains the sequence "2 6 0 5 4 1 3".

Third Screenshot: The application is in the decryption state. The "Input:" text area contains the encrypted text "ypo-crg-typhra". The "Output:" text area contains the decrypted text "cryptography". The "Key size:" is set to 7, and the "Key" field contains the sequence "2 6 0 5 4 1 3".

c. Caesar

Caesar

Caesar Cipher

Input: Output:

Key: Encryption Decryption

Caesar

Caesar Cipher

Input: Output:

abc def

Key: 3 Encryption Decryption

Caesar

Caesar Cipher

Input:

def

Output:

abc

Key

3

Encryption

Decryption

d. Caesar Frequent Analysis

Caesar1

Caesar Cipher Frequency Analysis

Input

Output

☐ Encrypt

Key:

Choose language

Русский

GO

Clean

Caesar1

Caesar Cipher Frequency Analysis

Input

Output

☒ Encrypt

Key:

3

Choose language

English

GO

Clean

Caesar1

Caesar Cipher Frequency Analysis

Input

sdvvlqj wkh hadp lq d surmhfw irup zloo
doorz vwxghqvv wr hasdqg wxhlu
nqrzohgjh ri ghyhorsphqw hqylurqphqvv,
oleudulhv dqg surjudpplaj wrrov dqg
lpsuryh sudfwlfd vnloov lq zulwlqj
surjudpv dqg dssolfdwlrqv iru rujdqqlqj
fubswrjudsklf vhfxfulwb ri gdwd edvhg rq

Output

☐ Encrypt
☒ Decrypt

Key:

Choose language

English

GO

Clean

Caesar1

Caesar Cipher Frequency Analysis

Input

sdvvlqj wkh hadp lq d surmhfw irup zloo
doorz vwxghqvv wr hasdqg wxhlu
nqrzohgjh ri ghyhorsphqw hqylurqphqvv,
oleudulhv dqg surjudpplaj wrrov dqg
lpsuryh sudfwlfd vnloov lq zulwlqj
surjudpv dqg dssolfdwlrqv iru rujdqqlqj
fubswrjudsklf vhfxfulwb ri gdwd edvhg rq

Output

passing the exam in a project form will
allow students to expand their knowledge
of development environments, libraries
and programming tools and improve
practical skills in writing programs and
applications for organizing cryptographic
security of data based on various

☐ Encrypt
☒ Decrypt

Key:

3

Choose language

English

GO

Clean

e. Vigenere with frequency analysis

Vigenere

Vigenere Cipher

Input

Result

☐ Encrypt
☐ Decrypt

Key:

Choose language

English

GO

Clean

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
HDCSIYUPNKFLQBPNLPHVDIQGCGFH
WJUODIPNOQJWDUCROMFPGSBHL
FMGFPHMISJHJTYFVLLUSSSIGVWW
TNKBNHNJOPSFSSSUUQBFTNVUI
WFMWUBHJLVFWXSCWZZHBNXSIV

☒ Encrypt

Key: CAT

Choose language

GO

☐ Decrypt

Key list:

English

Clean

Vigenere

— □ ×

Vigenere Cipher

Input

FSSSUUQBFBTCVJMSSUFUCFFXEZUES
IDELDOAHPZRSADOCCBNLPHVJHFM
OGJHJHIYFLQNYQUMDJGLOAWPXHD
CSIYUPNKFLQBPNLPHVDIQGCGFWJ
UODIPNOQJWDUCROMFPGSBHLFM
GFPHMISJHJTYFVLLUSSSIGVWWTNK
BNHNJOPSFSSSUUQBFTNVUIWFM
WUBHJLVFWXSCWZZHBNXSIVBHGI

Result

CRYPTANALYSISISPRACTICEDBYABR
OADRANGE OF ORGANIZATIONS INCL
UDING GOVERNMENTS AIMING TO DE
CIPHER OTHER NATIONS CONFIDENTI
AL COMMUNICATIONS COMPANIES D
EVELOPING SECURITY PRODUCTS THA
TEMPLOY CRYPTANALYSTS TO TEST TH
EIR SECURITY FEATURES AND HACKER
S

☐ Encrypt

Key:

Choose language

GO

☒ Decrypt

Key list: CAT

English

Clean

f. XOR

XOR

XOR

PlainText

Ecrption/Decryption

Hex

Key

Generate Key

Encrypt

Decrypt

XOR

XOR

PlainText

HELLO WORLD

Ecrption/Decryption

0!4(7D/+*(<

Hex

4833524055684743424060

Key

xd

Generate Key

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

Public Key

Private Key

Generate Key

Text Content

Text Result

Encrypt

Decrypt

RCA

Public Key

1, 6497

Private Key

1, 6497

Generate Key

Text Content

hello world

Text Result

8 5 12 12 15 100 23 15 18 12 4

Encrypt

Decrypt

RCA

Public Key

1, 6497

Private Key

1, 6497

Generate Key

Text Content

8 5 12 12 15 100 23 15 18 12 4

Text Result

hello world

Encrypt

Decrypt

i. AES

AES

AES

Input

Output

Key Rounds:

Show Rounds

Key:

Encrypt

Decrypt

AES

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

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 1e fb

Two One Nine Two

stMmowwvdRj7IOsvLt/c2sSCTKcDI1sR634arp
+rmK72ZK4jzHBw/ZlcYVu0bNIP

Thats my Kung Fu

Encrypt

Decrypt

AES

Input

stMmowwvdRj7lOsvLt/
c2sSCTKcDI1sR634arp+rmK72ZK4JzHBw/
ZlcYVu0bNIP

Output

Two One Nine Two

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

EncryptDecrypt

References:

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