Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Лабораторная работа 1

Проверил: Чербу Ольга

Выполнил: Чобану Артём

Кишинев 2022

**Алгоритм RC6**

Шифруемое слово: **aclanguage\_input**

Ключ – **abcdefgh**

Скриншоты работы программы:

Text

Description automatically generated

Код программы:

using System;  
using System.Numerics;  
using System.Text;  
  
const uint w = 32;  
const uint r = 20;  
const uint Pw = unchecked(0xb7e15163);  
const uint Qw = unchecked(0x9e3779b9);  
  
var input = "aclanguage\_input";  
var key = "abcdefhg";  
  
Console.WriteLine($"Input string: {input}");  
Console.WriteLine($"Input key: {key}");  
  
var inputBytes = Encoding.UTF8.GetBytes(input);  
byte[] keyBytes = Encoding.UTF8.GetBytes(key);  
  
var S = ExpandKeys(keyBytes);  
  
Console.WriteLine("Encrypting:");  
var encrypted = Encrypt(inputBytes, S);  
  
Console.WriteLine("\nDecrypting:");  
byte[] decrypted = Decrypt(encrypted, S);  
  
Console.WriteLine("\nResult:");  
string decrypted\_text = Encoding.UTF8.GetString(decrypted);  
Console.WriteLine(decrypted\_text);  
  
Console.ReadKey();  
  
uint[] ExpandKeys(byte[] key)  
{  
 var c = (uint)Math.Max(key.Length, 1) / (w / 8);  
 var L = ByteArrayToIntArray(key);  
  
 uint[] S = new uint[2 \* r + 4];  
 S[0] = Pw;  
  
 for (int x = 1; x < S.Length; x++)  
 {  
 S[x] = S[x - 1] + Qw;  
 }  
  
 var v = 3 \* Math.Max(c, 2 \* r + 4);  
  
 uint A = 0, B = 0, i = 0, j = 0;

for (int s = 1; s <= v; s++)  
 {  
 A = S[i] = BitOperations.RotateLeft(S[i] + A + B, 3);  
 B = L[j] = BitOperations.RotateLeft(L[j] + A + B, (int)(A + B));  
 i = (i + 1) % (2 \* r + 4);  
 j = (j + 1) % c;  
 }  
  
 return S;  
}  
  
uint[] Encrypt(byte[] inputBytes, uint[] S)  
{  
 var input = ByteArrayToIntArray(inputBytes);  
  
 var A = input[0];  
 var B = input[1];  
 var C = input[2];  
 var D = input[3];  
  
 Console.WriteLine($"A: {A}, B: {B}, C: {C}, D:{D} (From input)");  
  
 B += S[0];  
 D += S[1];  
  
 int shift = 5; // log2 (32)  
  
 for (int i = 1; i <= r; i++)  
 {  
 var t = BitOperations.RotateLeft(B \* (2 \* B + 1), shift);  
 var u = BitOperations.RotateLeft(D \* (2 \* D + 1), shift);  
  
 A = BitOperations.RotateLeft(A ^ t, (int)u) + S[2 \* i];  
 C = BitOperations.RotateLeft(C ^ u, (int)t) + S[2 \* i + 1];  
  
 var temp = A;  
 A = B;  
 B = C;  
 C = D;  
 D = temp;  
  
 Console.WriteLine($"A: {A}, B: {B}, C: {C}, D:{D}");  
 }  
  
 A += S[2 \* r + 2];  
 C += S[2 \* r + 3];  
  
 return new[] { A, B, C, D };  
}  
  
byte[] Decrypt(uint[] input, uint[] S)  
{  
 var A = input[0];  
 var B = input[1];  
 var C = input[2];  
 var D = input[3];  
  
 C -= S[2 \* r + 3];  
 A -= S[2 \* r + 2];  
  
 Console.WriteLine($"A: {A}, B: {B}, C: {C}, D:{D} (From input)");  
  
 int shift = 5;  
  
 for (var i = r; i >= 1; i--)  
 {  
 var temp = D;  
 D = C;  
 C = B;  
 B = A;  
 A = temp;  
  
 var u = BitOperations.RotateLeft(D \* (2 \* D + 1), shift);  
 var t = BitOperations.RotateLeft(B \* (2 \* B + 1), shift);  
  
 C = BitOperations.RotateRight(C - S[2 \* i + 1], (int)t) ^ u;  
 A = BitOperations.RotateRight(A - S[2 \* i], (int)u) ^ t;  
  
 Console.WriteLine($"A: {A}, B: {B}, C: {C}, D:{D}");  
 }  
 D -= S[1];  
 B -= S[0];  
  
 var outputArr = IntArrayToByteArray(new[] { A, B, C, D });  
  
 return outputArr;  
}  
  
static byte[] IntArrayToByteArray(uint[] integerArray)  
{  
 byte[] result = new byte[integerArray.Length \* 4];  
  
 for (int i = 0; i < result.Length; i++)  
 {  
 result[i] = (byte)((integerArray[i / 4] >> i % 4 \* 8) & 0xff);  
 }  
  
 return result;  
}  
  
static uint[] ByteArrayToIntArray(byte[] bytes)  
{  
 uint[] result = new uint[bytes.Length / 4];  
  
 for (int i = 0, counter = 0; i < result.Length; i++)  
 {  
 var current = (bytes[counter++] & 0xff) |  
 ((bytes[counter++] & 0xff) << 8) |  
 ((bytes[counter++] & 0xff) << 16) |  
 ((bytes[counter++] & 0xff) << 24);  
  
 result[i] = (uint)current;  
 }  
  
 return result;  
}

Выводы:

Алгоритм RC6 очень компактен, и может быть реализован в менее чем 256 байтов кода для установки ключа, блока шифрования и блока дешифрования.

Даннные алгоритма RC6 могут поместиться в кэш памяти, так как он не использует справочные таблицы во время гифрования.

Кроме того, RC6 полностью параметризируется, что делает его эффективным и гибким.