Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Аттестационная работа 1

Проверил:  
O.Cerbu

Выполнил:  
студент группы IA1802   
Заведеева Анастасия

Кишинев 2020

**Содержание**

[**Описание сети Фейстеля.**](#_puw7vn6cxgee) **3**

[**RC2**](#_lxg8w8zgwbts) **4**

[Encrypt](#_f68xd5q5veck) 4

[Decrypt](#_sc5qt4puo58t) 5

[**TEA**](#_xgdq8y3hk3gs) **6**

[Нечетная часть](#_wleyz8695177) 6

[Четная часть (правая часть = п.8, левая часть = правой сверху)](#_i54k9sbgtn0i) 7

[Результат](#_dj59hws5c2jx) 7

[Decrypt](#_3kv70ehk9xna) 8

[Четная часть](#_5yiy7cry5t46) 8

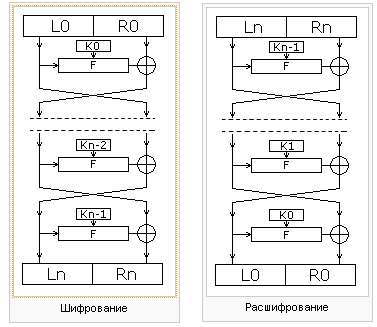
[Нечетная часть](#_aknbre3qxxgq) 9

# Описание сети Фейстеля.

Сеть Фейстеля - это многократно повторяющиеся преобразования состояния блока данных - для очередной ячейки сети входными значениями (для преобразования) служат выходные значения предыдущей ячейки (то есть данные преобразованные ранее). Является симметричным алгоритмом. Эту схему использует большая часть блочных шифров, в том числе новейшие шифры Blowfish и Twofish.

Сеть Фейстеля - это система блочного шифрования, то есть если данные больше блока - то мы разрезаем их на блоки фиксированной длины, если последний фрагмент окажется короче блока - дополняем его до нужной длинны любым способом (например, незначащими нулями)

### 

****

# **RC2**

Шифруемое слово - **composer**

Ключ - **az** - Anastasia Zaveeeva

## **Encrypt**

a => 01100011 01101111 b => 01101101 01110000

c => 01101111 01110011 d => 01100101 01110010

subkey => 01000001 01011010

1. a + subkey => (

01100011 01101111 +

01000001 01011010

10100100 11001001

1. c & d =>

01101111 01110011 &

01100101 01110010 =

01100101 01110010

1. (not d) & b =>

10011010 10001101 &

01101101 01110000 =

00001000 00000000

1. (c ^ d) + (not d ^ b) =

01100101 01110010 +

00001000 00000000 =

01101101 01110010

1. pt1 + pt4 =

10100100 11001001 +

01101101 01110010 =

00000001 00010010 00111011

1. 6. Сдвиг влево на 4:

00000001 00010010 00111011 <<< 4 =>

0001 00010010 00111011 0000

1. b => 01101101 01110000 c => 01101111 01110011d => 01100101 01110010 a => 00010001 00100011 10110000
2. Конкатенация b,c,d,a:

01101101 01110000 01101111 01110011 01100101 01110010 00010001 00100011 10110000

Результат в ascii: **mposer#°**

Результат в hex: **6d706f736572100000**

## **Decrypt**

1. Сдвиг вправо на 4:

000100010010001110110000 >>> 4 =

000000010001001000111011

1. c & d =>

01101111 01110011 &

01100101 01110010 =

01100101 01110010

1. (not d) & b =>

10011010 10001101 &

01101101 01110000 =

00001000 00000000

1. (c ^ d) + (not d ^ b) =

01100101 01110010 +

00001000 00000000 =

01101101 01110010

1. a - (b ^ (not d) + (c ^ d)) =

00000001 00010010 00111011 -

01101101 01110010 =

10100100 11001001

1. pt5 - AZ = (

10100100 11001001 -

01000001 01011010 =

01100011 01101111

1. Конкатенация a,b,c,d: 01100011 01101111 01101101 01110000 01101111 01110011 01100101 01110010

Результат в ascii: **composer**

Результат в hex: **636f6d706f736400**

# 

# 

# 

# **TEA**

Message: **composer**

01100011 01101111 01101101 01110000 01101111 01110011 01100101 01110010

Left: 01100011 01101111 01101101 01110000

Right: 01101111 01110011 01100101 01110010

Ключи:

K0: **azid**

K1: **ball**

K2: **cash**

K3: **open**

**Encrypt**

### **Нечетная часть**

1. Правую часть сдвигаем влево на 4 бита:

01101111 01110011 01100101 01110010 << 4 =

1111 01110011 0110 0101 01110010 0000

1. К результату из п.1 добавляем K0:

11110111 00110110 01010111 00100000 +

01100001 01111010 01101001 01100100 =

01011000 10110000 11000000 10000100

1. К правой части добавляем I\*delta (I = 1)

01101111 01110011 01100101 01110010 + 1 \*

**10011110 00110111 01111001 10111001** =

00001101 10101010 11011111 00101011

1. Результат из п2. XOR результат из п.3

01011000 10110000 11000000 10000100 XOR

00001101 10101010 11011111 00101011 =

01010101 00011010 00011111 10101111

1. Правую часть сдвигаем вправо на 5 бит:

011 01111011 10011011 00101011 10010 >> 5 =

00000011 01111011 10011011 00101011

1. К результату из п.5 добавляем K1:

00000011 01111011 10011011 00101011 +

01100010 01100001 01101100 01101100 =

01100101 11011101 00000111 10010111

1. Результат из п4. XOR результат из п.6

01010101 00011010 00011111 10101111 XOR

01100101 11011101 00000111 10010111 =

00110000 11000111 00011000 00111000

1. К левой части прибавляется результат из п.7

01100011 01101111 01101101 01110000 +

00110000 11000111 00011000 00111000 =

10010100 00110110 10000101 10101000

### **Четная часть (правая часть = п.8, левая часть = правой сверху)**

1. 1. Правую часть сдвигаем влево на 4 бита:

10010100 00110110 10000101 10101000 << 4 =

0100 00110110 10000101 10101000 0000

1. К результату из п.1 добавляем K2:

01000011 01101000 01011010 10000000 +

01100011 01100001 01110011 01101000 =

10100110 11001001 11001101 11101000

1. К правой части добавляем I\*delta (I = 1)

10010100 00110110 10000101 10101000 + 1 \*

**10011110 00110111 01111001 10111001** =

00110010 01101101 11111111 01100001

1. Результат из п2. XOR результат из п.3

10100110 11001001 11001101 11101000 XOR

00110010 01101101 11111111 01100001 =

10010100 10100100 00110010 10001001

1. Правую часть сдвигаем вправо на 5 бит:

100 10100001 10110100 00101101 01000 >> 5 =

00000100 10100001 10110100 00101101

1. К результату из п.5 добавляем K3:

00000100 10100001 10110100 00101101 +

01101111 01110000 01100101 01101110 =

01110100 00010010 00011001 10011011

1. Результат из п4. XOR результат из п.6

10010100 10100100 00110010 10001001 XOR

01110100 00010010 00011001 10011011 =

11100000 10110110 00101011 00010010

1. К левой части прибавляется результат из п.7:

01101111 01110011 01100101 01110010 +

11100000 10110110 00101011 00010010 =

01010000 00101001 10010000 10000100

### **Результат**

10010100 00110110 10000101 10101000 01010000 00101001 10010000 10000100

Результат в ascii: **6¨P)**

Результат в hex: **943685a850299000**

## **Decrypt**

### **Четная часть**

10010100 00110110 10000101 10101000 - left

01010000 00101001 10010000 10000100 - right

1. Правую часть сдвигаем влево на 4 бита:

0101 0000001 01001100 100001000 0100 << 4 =

0000001 01001100 100001000 0100 0000

1. К результату из п.1 добавляем K2:

00000010 10011001 00001000 01000000 +

01100011 01100001 01110011 01101000 =

01100101 11111010 01111011 10101000

1. К правой части добавляем I\*delta (I = 1)

01010000 00101001 10010000 10000100 + 1 \*

**10011110 00110111 01111001 10111001** =

11101110 01100001 00001010 00111101

1. Результат из п2. XOR результат из п.3

01100101 11111010 01111011 10101000 XOR

11101110 01100001 00001010 00111101 =

10001011 10011011 01110001 10010101

1. Правую часть сдвигаем вправо на 5 бит:

010 10000001 01001100 10000100 00100 >> 5 =

00000 010 10000001 01001100 10000100

1. К результату из п.5 добавляем K3:

00000010 10000001 01001100 10000100 +

01101111 01110000 01100101 01101110 =

01110001 11110001 10110001 11110010

1. Результат из п4. XOR результат из п.6

10001011 10011011 01110001 10010101 XOR

01110001 11110001 10110001 11110010 =

11111010 01101010 11000000 01100111

1. 8. К левой части прибавляется результат из п.7:

10010100 00110110 10000101 10101000 +

11111010 01101010 11000000 01100111 =

10001110 10100001 01000110 00001111

**10001110 10100001 01000110 00001111** -> oser

### **Нечетная часть**

1. Правую часть сдвигаем влево на 4 бита:

01010000 00101001 10010000 10000100 << 4 =

0000 00101001 10010000 10000100 0000

1. К результату из п.1 добавляем K0:

00000010 10011001 00001000 01000000 +

01100001 01111010 01101001 01100100 =

01100100 00010011 01110001 10100100

1. К правой части добавляем I\*delta (I = 1)

01010000 00101001 10010000 10000100 + 1 \*

**10011110 00110111 01111001 10111001** =

11101110 01100001 00001010 00111101

1. Результат из п2. XOR результат из п.3

01100100 00010011 01110001 10100100 XOR

11101110 01100001 00001010 00111101 =

10001010 01110010 01111011 10011001

1. Правую часть сдвигаем вправо на 5 бит:

010 10000001 01001100 10000100 00100 >> 5 =

00000010 10000001 01001100 10000100

1. К результату из п.5 добавляем K1:

00000010 10000001 01001100 10000100 +

01100010 01100001 01101100 01101100 =

01100100 11100010 10111000 11110000

1. Результат из п4. XOR результат из п.6

10001010 01110010 01111011 10011001 XOR

01100100 11100010 10111000 11110000 =

11101110 10010000 11000011 01101001

1. К левой части прибавляется результат из п.7:

10010100 00110110 10000101 10101000 +

11101110 10010000 11000011 01101001 =

10000010 11000111 01001001 00010001

**10000010 11000111 01001001 00010001** -> comp