**Отчёт**

*Манух Анна 8 группа*

**Лабораторная №1**

Условие лабораторной работы: Ваша задача – реализация двух  
классических шифров согласно вашему варианту (первый шифр смотрите в  
таблице № 1, второй – в таблице №2).

|  |  |  |  |
| --- | --- | --- | --- |
| вар. | Шифр | Ключ | Открытый текст |
| 14. | Аффинный | 4 4 | КУДРАВЕЦ |
| 14. | Аффинный | 25 12 | ПЛЗЛНЁШТ |

Программный код:

public class Main {

private static final String alphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

enum Mode {

ENCRYPT, DECRYPT

}

private static String Affin(int a, int b, String Text, Mode mode)

{

String result = "";

if (mode == Mode.ENCRYPT) {

for (char c : Text.toCharArray()) {

int index = alphabet.indexOf(c);

result += alphabet.charAt((a \* index + b) % alphabet.length());

}

} else {

int a\_ = 0;

int pos = 0;

for (int i = 0; i < alphabet.length(); i++) {

pos = (a \* i) % alphabet.length();

if (pos == 1) {

a\_ = i;

}

}

for (int i = 0; i < Text.length(); i++) {

int indexInCipher = alphabet.indexOf(Text.charAt(i));

int index = (indexInCipher - b + alphabet.length()) % alphabet.length();

result += alphabet.charAt(a\_ \* index % alphabet.length());

}

}

return result;

}

public static void main(String[] args) {

String Encrypted = Affin(4, 4, "КУДРАВЕЦ", Mode.ENCRYPT);

String Decrypted = Affin(25, 12, "ПЛЗЛНЁШТ", Mode.DECRYPT);

String Decrypted2 = Affin(4, 4, "ОСУЁДЛЧЭ", Mode.DECRYPT);

System.out.println("зашифр: "+ Encrypted + "\n " + "расшифр: "+Decrypted+"\n"+"расшифр2: "+Decrypted2);

}

}

Результат:

зашифр: ОСУЁДЛЧЭ

расшифр: ПАРАЗИТЫ

расшифр2: КУДРАВЕЦ

**Лабораторная №2**

Условие лабораторной работы: Необходимо реализовать (программно или вручную) блочно-итерационных шифр, состоящий из применения 3 итераций следующей упрощенной *SP*-подстановки.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | Вид «Сложение» | Раундовые ключи – номера битов из общего ключа | Блок *S*1 | Блок *S*2 | Блок *P* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 14 | Круглый плюс | 1 – (10, 12, 2, 5, 8, 6, 9, 4) 2 – (2, 9, 10, 5, 1, 12, 6, 4) 3 – (7, 1, 2, 6, 12, 3, 9, 11) | 5 | 4 | Циклический сдвиг на 6 бит влево |

Задание реализовано вручную.

**Лабораторная №3**

Условие лабораторной работы: В рамках данной лабораторной работы вам необходимо реализовать генератор Геффе с заданными характеристическими многочленами и начальными состояниями.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | РСЛОС №1 (5) | РСЛОС №2 (7) | РСЛОС №3 (8) |
| 14 | 00100 10111 | 0111010 0011001 | 00000100 11100011 |

Программный код:

class Geffe\_Generator {

Integer[] a1 = {0, 0, 1, 0, 0};

Integer[] c1 = {1, 1, 1, 0, 1};

Integer[] a2 = {0, 1, 0, 1, 1, 1, 0};

Integer[] c2 = {1, 0, 0, 1, 1, 0, 0};

Integer[] a3 = {0, 0, 1, 0, 0, 0, 0, 0};

Integer[] c3 = {1, 1, 0, 0, 0, 1, 1, 1};

String a1\_copy = Arrays.toString(a1);

String a2\_copy = Arrays.toString(a2);

String a3\_copy = Arrays.toString(a3);

Integer n;

Integer[] GeffePosl;

public Geffe\_Generator(Integer n) {

Set<String> r = new LinkedHashSet<>();

this.n = n;

GeffePosl = new Integer[n];

Integer x1, x2, x3;

int k1 = 0, k2 = 0, k3 = 0;

int C1 = 0, C2 = 0, C3 = 0;

for (int i = 0; i < n; i++) {

String buf\_1 = Arrays.toString(a1);

String buf\_2 = Arrays.toString(a2);

String buf\_3 = Arrays.toString(a3);

x1 = Rslos(a1, c1);

x2 = Rslos(a2, c2);

x3 = Rslos(a3, c3);

if (Arrays.toString(a1).equals(a1\_copy)) {

System.out.println(i);

k1 = i;

C1++;

r.add(buf\_1);

}

if (Arrays.toString(a2).equals(a2\_copy)) {

k2 = i;

C2++;

r.add(buf\_2);

}

if (Arrays.toString(a3).equals(a3\_copy)) {

k3 = i;

C3++;

r.add(buf\_3);

}

GeffePosl[i] = ((x1 \* x2) + ((x1 + 1) % 2) \* x3) % 2;

System.out.println(Arrays.toString(a1));

}

//System.out.println(k1 + " и " + C1);

System.out.println("Период первого РСЛОС: " + (k1 / C1));

//System.out.println(k2 + " и " + C2);

System.out.println("Период второго РСЛОС: " + (k2 / C2));

//System.out.println(k3 + " и " + C3);

System.out.println("Период третьего РСЛОС: " + (k3 / C3));

System.out.println("Последовательность до зацикливания: ");

for (String s : r) {

System.out.println(s);

}

}

public Integer Rslos(Integer[] a, Integer[] c) {

int r = 0;

Integer res = a[0];

for (int i = 0; i < c.length; i++) {

r += (c[i] \* a[i]);

}

r %= 2;

if (c.length - 1 >= 0) System.arraycopy(a, 1, a, 0, c.length - 1);

a[a.length - 1] = r;

return res;

}

public String Count\_0\_1() {

int a = 0, b;

for (int i = 0; i < n; i++) {

a += GeffePosl[i];

}

b = n - a;

return b + " нулей, " + a + " единиц";

}

public String R() {

Integer[] R = new Integer[5];

StringBuilder res = new StringBuilder();

for (int i = 0; i < 5; i++) {

R[i] = 0;

res.append("r[").append(i + 1).append("]= ");

for (int j = 1; j < n - i - 1; j++) {

R[i] += (int) Math.pow(-1, (GeffePosl[j] + GeffePosl[j + i + 1]) % 2);

}

res.append(R[i]).append("\n");

}

return res.toString();

}

}

public class Main {

public static void main(String[] args) {

Geffe\_Generator G = new Geffe\_Generator(10000);

System.out.println("Количество нулей и единиц: " + G.Count\_0\_1());

System.out.println(G.R());

}

}

Результат:

Период первого РСЛОС: 30

Период второго РСЛОС: 23

Период третьего РСЛОС: 92

Последовательность до зацикливания:

[1, 0, 1, 0, 1, 1, 1]

[0, 0, 0, 1, 0]

[0, 0, 0, 1, 0, 0, 0, 0]

Количество нулей и единиц: 5169 нулей, 4831 единиц

r[1]= -542

r[2]= 119

r[3]= -1602

r[4]= 1191

r[5]= 320

**Лабораторная №4**

Условие лабораторной работы:Для выполнения всех заданий необходимо использовать "длинную" арифметику. Разрешается  
использовать любую готовую библиотеку или написать свою.  
Шаг 1. В условиях своего варианта для заданных чисел *p*, *q* и *e*, необходимо вычислить личный  
ключ *d*. Можно использовать либо расширенный алгоритм Евклида или малую теорему Ферма  
(частный случай теоремы Эйлера).  
Шаг 2. Для заданного сообщения *X*1, вычислить зашифрованное сообщение *Y*1, используя  
открытый ключ *e*.  
Шаг 3. Расшифровать сообщение *Y*1, используя найденный личный ключ *d*, сравнить результат  
с исходным сообщением *X*1.  
Шаг 4. Для заданного шифртекста *Y*2, вычислить исходный открытый текст *X*2, используя  
личный ключ *d*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № вар. | p | q | e | X1 | Y2 |
| 14 | 172382424535073 | 270521047700671 | 3751677126372540769365457709 | 17290178567521224714295393741 | 40212663200342538021234198502 |

Программный код:

class RSA:

def \_\_init\_\_(self,

p: int,

q: int,

e: int,

):

self.n = p \* q

self.e = e

self.d = self.\_generate\_private\_key(p, q)

def power(a, n):

return (1 if n == 0

else power(a \* a, n // 2) if n % 2 == 0

else a \* power(a, n - 1))

def \_generate\_private\_key(self, p: int, q: int) -> int:

phi = (p - 1) \* (q - 1)

return mod\_inverse(self.e, phi)

def encrypt(self, message: int) -> int:

return pow(message, self.e, self.n)

def decrypt(self, message: int) -> int:

return pow(message, self.d, self.n)

p = 172382424535073

q = 270521047700671

e = 3751677126372540769365457709

X1 = 17290178567521224714295393741

Y2 = 40212663200342538021234198502

rsa = RSA(p, q, e)

print(f'Исходное сообщение X1: {X1}')

Y1 = rsa.encrypt(X1)

print(f'Зашифрованное сообщение Y1: {Y1}')

X1\_dec = rsa.decrypt(Y1)

print(f'После расшифровки: {X1\_dec}')

print(f'Зашифрованное сообщение Y2: {Y2}')

X2\_dec = rsa.decrypt(Y2)

print(f'После расшифровки: {X2\_dec}')

Результат:

Исходное сообщение X1: 17290178567521224714295393741

Зашифрованное сообщение Y1: 38903819864462547331347418040

После расшифровки: 17290178567521224714295393741

Зашифрованное сообщение Y2: 40212663200342538021234198502

После расшифровки: 29157983802348781240587544540