Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

О.В. Голубева
С.Г. Ехилевский
Ю.Ф. Пастухов
Д.Ф. Пастухов
А.Ю. Зеленуха

Шифрование с помощью нелинейных функций

Учебно-методическое пособие к лекционным и практическим занятиям
для студентов специальности
1-98 01 01 Компьютерная безопасность

Новополоцк
ПГУ
2017

Одобрено и рекомендовано к изданию
методической комиссией факультета информационных технологий
В качестве учебно-методического пособия

Кафедра технологий программирования

Рецензенты:
А.Ф. Оськин, кандидат технических наук, доцент, доцент кафедры технологий программирования;
Р.П. Богуш, кандидат технических наук, доцент, заведующий кафедрой вычислительных систем и сетей

СОДЕРЖАНИЕ

1.Введение

Шифрование с использованием нелинейных функций преследует цель выработать эффективные способы шифрования и дешифрования, имея набор известных нелинейных математических функций, применяя несложные математические вычисления, и в то же время получить надёжную степень защиты шифрованного текста от атак злоумышленников.

Хорошо известно, что диапазон изменения целых чисел значительно меньше диапазона изменения действительных чисел с плавающей запятой. Например, для переменной типа int на языке C++ верхнее значение имеет порядок 10^{10}. В то же время верхнее значение переменной типа double превышает 10^{300}. Кроме того, чувствительность области действительных чисел составляет 10^{-16}. Т.е. там, где целочисленная переменная изменяется на 1, переменная типа double в этом же диапазоне изменится 10^{16} раз.

Богатство диапазона чисел двойной точности заслуживает пристального внимания, для того чтобы применить эти возможности для шифрования и дешифрования.

Кроме того применение для кодирования - декодирования большого набора взаимно - обратных пар нелинейных функций увеличивает криптографическую стойкость шифрованного текста.

2.Постановка задачи. Крипто стойкость шифрования нелинейными функциями.

Неимоверно большая чувствительность шифра относительно малых $10^{-10}-10^{-12}$ изменений параметров ключа $(left, raight, a), (left, raight, n)$ обеспечивает большую размерность пространства шифров. Если бы суперкомпьютер крипто аналитика подбирал ключи к фиксированному шифру нелинейными функциями и тройным ключом со скоростью $10^9 \frac{1}{сек}$, то понадобилось бы время взлома 30 лет. Такая большая размерность обеспечивается применением чисел двойной точности(double) для аргументов функций и нелинейных функций. Действительно, в диапазон изменения одного параметра ключа равный 1 можно разместить 10^6 различных дешифрованных текстов. Крипто стойкость шифрования нелинейными функциями обеспечивается двумя вескими причинами: 1) большая размерность пространства ключей, 2)большой набор нелинейных функций с “плавающей” областью определения – отрезком, позволяющей с одной стороны увеличить пространство ключей, а с другой стороны повысить крипто стойкость шифрования.

Постановка задачи: Выработать быстрые и крипто стойкие алгоритмы шифрования с использованием нелинейных функций для шифрования данных. В данной дипломной работе применялись 2 класса нелинейных

функций 1) прямые и обратные тригонометрические функции 2) показательные и логарифмические функции. Применяемые алгоритмы в программе можно использовать для хранения в базе данных паролей длиной до нескольких сотен – тысяч символов.

3. Структурная схема протокола обмена сеансовым ключом

Пусть два абонента договорились обмениваться данными с помощью протокола без передачи ключей. Опишем алгоритм протокола. Перед началом работы абоненты должны создать общий сеансовый ключ. Ключ состоит в знании каждым из абонентов трёх (девяти) символьных строк с длиной не более 16 символов – это запись десятичных чисел - параметров $left, raight, a$ ($left, raight, m, n$). Каждый из 16 символов указанной тройки чисел с помощью таблицы ASCII можно перевести, используя десятичную точку и разделительный знак – пробел между числами в определённый номер символа 0– 48;1– 49; 2– 50;3– 51; 4– 52; 5– 53; 6– 54; 7– 55;8– 56;9 – 57; “.” – 46; “ “ – 32 . Для создания совместного ключа нужно передать не более чем 16*3+3+2=53 символа(159символов – двухзначных натуральных чисел). Абоненты для секретной переписки конфиденциально выбирают достаточно большое простое число p такое, что p-1 разлагается на не очень большие простые множители. Каждый из абонентов независимо выбирает некоторое простое число взаимно простое с $\varphi(p) = p - 1$ функцией Эйлера числа p. Пусть числа абонентов a – A , b – B . Эти числа являются первыми секретными ключами абонентов. Вторые секретные ключи $\alpha - A, \beta - B$ находятся из уравнений

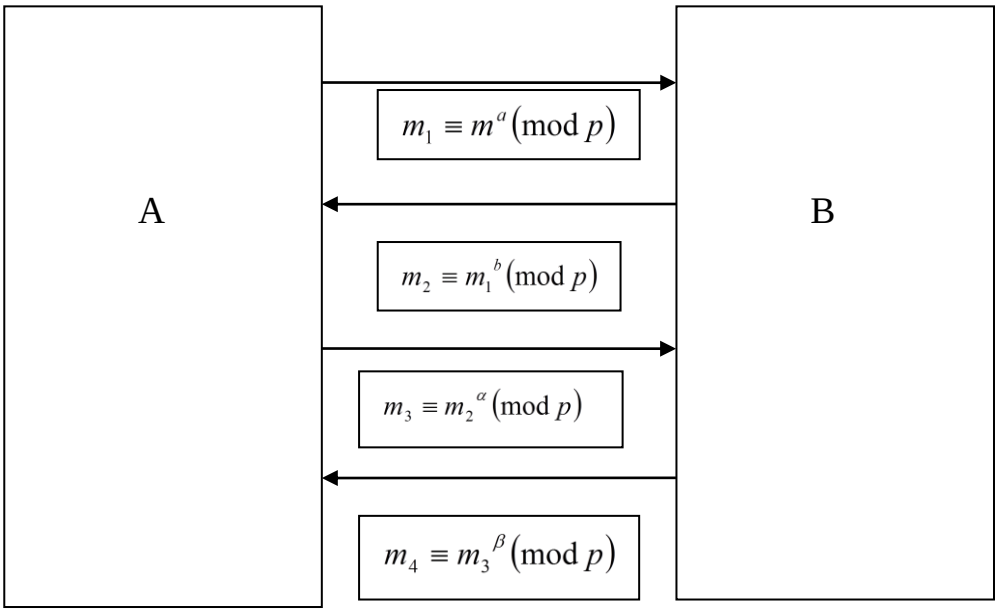
$$\alpha a \equiv 1 \bmod(\varphi(p)), \beta b \equiv 1 \bmod(\varphi(p)), 0 < \alpha < p-1, 0 < \beta < p-1.$$

Пересылаемые сообщения должны быть меньше p-1(достаточно выбрать p>59). Пусть A отправляет сообщение m(символы параметров ключей, которые задаёт A для B). Тогда для номера m каждого символа он посылает первым ключом число $m_1 \equiv m^a \pmod p$. B, получив m_1, зашифровывает его своим ключом $m_2 \equiv m_1^{\,b} \pmod p$.

Далее A шифрует по формуле $m_3 \equiv m_2^{\,\alpha} \pmod p$ и отправляет абоненту B, который теперь может расшифровать сообщение по формуле $m_4 \equiv m_3^{\,\beta} \pmod p$. Действительно,

$m_4 \equiv m^{b\beta\alpha a} \pmod p \equiv m^{k\varphi(p)+1} \pmod p \equiv m\left(m^{\varphi(p)}\right)^k \pmod p \equiv m \pmod p$, так как по теореме Эйлера – Ферма $m^{\varphi(p)} \equiv 1 \pmod p$.

Таким образом, можно построить структурную схему передачи 3 параметров – действительных чисел, используя алгоритм протокола без передачи ключа.



Структурная схема передачи параметров сеансового ключа.

После трёх этапной передачи по каналу связи сеансового ключа, абоненты могут передавать сообщения, используя методы шифрования нелинейными функциями.

4.Описание работы основной программы (с использованием тригонометрических и степенных функций).

Первая программа применяется для шифрования текстовых данных нелинейными функциями. Она использует 2 взаимно обратные функции. Прямая функция $y = \mathrm{arc}\sin(\mathrm{x})^{\mathrm{n}}, n > 0, 0 \le x \le 1$. Тогда обратной к данной функции является функция $x = \sin\left(y^{\frac{1}{n}} \right),\ m = 1$. Либо пара взаимно обратных тригонометрических функций для параметра $m = 2$ $y = arctg(x)^{\mathrm{n}}, n > 0, 0 \le x < \infty$. $x = tg\left(y^{\frac{1}{n}} \right),\ m = 1.$

Параметрами ключей являются числа $left, raight, m, n$.

Опишем подробнее алгоритм работы программы:

1) каждому английскому символу, цифрам и знакам клавиатуры соответствует некоторое целое число (номер) от 0 до255, с помощью

таблицы ASCII можно перевести любой символ клавиатуры в число, например, слово Polotsk шифруется 7 символами 80 111 108 111 116 115 107.

Всего основных символов в клавиатуре 256 с участием больших и малых букв английского шрифта, невидимые знаки (табуляция, начало и конец строки и т.д.) а также знаки типа - !,»,№,(,) и т.д., цифры 0,1,2,3,4,5,6,7,8,9 пронумерованы от 0 до 255. Мы можем перевести каждый символ после применения команды ASCII в действительное число, заключённое от 0 до 1 по формуле $x = n/255, 0 \le x \le 1$, где n -номер символа в ASCII. В программе x записывается в массиве res10[ii].

2) Далее с единичным отрезком мы проводим линейное преобразование, растягивающее его в несколько раз и с применением параллельного переноса, смещающее левую точку по формуле $res11[ii] = left + (raight - left)res10[ii]$

где left , raight левая и правая границы нового отрезка. В результате последнего преобразования все точки преобразованного отрезка имеют координаты $x_1 \in [left, raight]$

3) сначала работает обратная функция с вызовом res100[ii]=f1(res11[ii],left,-raight,m,n), в которой по порядку следуют аргументы: число x после преобразования подобия из массива res11[ii], left, raight- левая и правая границы растянутого интервала, номер тригонометрических функций (целое m=1 для синусов и арксинусов, если m=2, то используются тангенс и арктангенс), далее идёт действительная степень n (не более 30.0),значения которой записываются в массив re-s100[ii]

В прямой программе используются арктангенсы либо арксинусы

```
double f1(double x, double left, double raight, int m, double n)
{
double norm,z;
if (m==1)
{
norm = pow(asin(raight),double(n));
z= pow(asin(x),double(n))/norm;
return z;
}
else if(m==2)
{
norm = pow(atan(raight),double(n));
z= pow(atan(x),double(n))/norm;
return z;
}
```

4) полученные значения масштабируются (умножаются на число 1000000)

```
res101[ii]= res100[ii]*1000000.0;
```

5)в файл 1001.txt записывается шифр из массива res101[ii]. Массив res101[ii] и является шифром. Здесь применяется одна важная идея. Поскольку диапазон целых чисел значительно меньше диапазона действительных чисел с плавающей запятой двойной точности, то массив res101[ii] заполняется действительными числами двойной точности. При считывании чисел из массива res101[ii] может быть потеряна мантисса действительного числа. Это может даже привести к ошибке на этапе декодирования. Чтобы избежать снижения точности, отдельно в текстовый файл1001.txt записывается целая часть числа res101[ii](как переменная double), а мантисса res101[ii] преобразуется в целое число с нужным количеством знаков точности и записывается вторым по счёту (как переменная double).

```
remove("1001.txt");
file=fopen("1001.txt","w");
for(i=0;i<=nn-1;i++)
{
aa=int(res101[i]);
bb=int ((res101[i]  -aa)*1e8);
fprintf(file,"%.4lf\n", aa);
fprintf(file,"%.4lf\n", bb);
}
fclose(file);
```

6)открываем и копируем данные из текстового файла1001.txt , затем из каждой последующей пары целых чисел формируем целую часть и мантиссу текущего кодированного символа

```
int sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1001.txt","r");
if(file==NULL)
{
printf(" not open file");
}
else
{
for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
```

```
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
 printf(" coder1(%d)=%.12lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
```

Далее столбцы кода до записи в текстовый файл и после чтения из него сравниваются coder1 и coder2(должны совпадать).
Прочитанные данные записываются в файл res102[i] .

7) проводим пошагово обратные операции

```
res103[i]=res102[i]/1000000.0;
```

делим массив res102[i] на 1000000 результат записывает в массив res103[i].

8)вызываем обратную функцию, в которой используются прямые тригонометрические функции синус и тангенс совместно со степенными функциями

```
res104[i]=f2(res103[i],left,raight,m,n0);
double f2(double x, double left, double raight, int m, double n)
{
double norm,z;
if (m==1)
{
norm = pow(asin(raight),double(n));
z=sin(pow((x)*norm,1.0/double(n)));
}
else if(m==2)
{
 norm = pow(atan(raight),double(n));
z= tan(pow((x)*norm,1.0/double(n)));
}
return z;
}
```

9)проводим обратное линейное преобразование и возвращаемся к переменной x по формуле   res14[i]=(res104[i]-left)/(raight-left);

10)

```
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
printf("start(%d) = %d  decoder(%d) = %d\n",i,res[i], i,res105[i]);
}
 printf("\n");
```

11) проводим обратное преобразование ASCII

```
for(i=0;i<=nn-1;i++)
{
 printf("%c",res105[i] );
}
```

12) в результате исходная символьная фраза

char str[nn+1]="Polotsk State University 1234567890"программой возвращается в ту же фразу.

Таким образом, в программе использованы математические операции - запись чисел с двойной точностью в текстовой файл, композиция из преобразования подобия, нелинейных тригонометрических преобразований, степенной функции $x_1 = n/255, x_2 = a + (b-a)x_1,\ y_1 = \arcsin x_2, y_2 = y_1^{\,n}$.

Обратные преобразования

$$y_1 = y_2^{\,1/n}, x_2 = \sin y_1, x_1 = \frac{x_2 - a}{b-a},\ n = 255x_1$$

5. **Описание работы основной программы (с использованием показательных и логарифмических функций)**

Вторая программа, применяемая для шифрования текстовых данных нелинейными функциями. Она использует 2 взаимно - обратные функции. Прямая функция $y = \ln(x)/a;\ a > 0, 0 < left \le x \le raight$.Тогда обратной к данной функции является функция $x = \exp(ay)$.

Параметрами ключей являются числа $left, raight, a$.

Опишем подробнее алгоритм работы программы:

1) каждому английскому символу, цифрам и знакам клавиатуры соответствует некоторое целое число (номер) 0т 0 до255, с помощью таблицы ASCII можно перевести любой символ клавиатуры в число, например, числа 012345678 шифруются 10 символами 48 49 50 51 52 53 54 55 56 57.

Всего основных символов в клавиатуре 256 с участием больших и малых букв английского шрифта, невидимые знаки (табуляция, начало и конец строки и т.д.) а также знаки типа - &,[,],№,(,) и т.д., цифры 0,1,2,3,4,5,6,7,8,9 пронумерованы от 0 до 255. Мы можем перевести каждый символ после применения команды ASCII в действительное число, заключённое от 0 до 1 по формуле $x = n/255, 0 \le x \le 1$, где n номер символа в ASCII. В программе x записывается в массиве res10[ii].

2) Далее с единичным отрезком мы проводим линейное преобразование, растягивающее его с некоторым коэффициентом и с использованием параллельного переноса, смещающее левую точку по формуле

$$res11[ii] = left + (raight - left)res10[ii]$$

где left , raight левая и правая границы нового отрезка. В результате последнего преобразования все точки преобразованного отрезка имеют координаты $x_1 \in [left, raight]$

3) сначала работает обратная функция с

вызовом res100[ii]=f1(res11[ii],left,raight,a), в которой по порядку следуют аргументы: число x после преобразования подобия из массива res11[ii], left,raight- левая и правая границы растянутого интервала, далее идёт действительное число - параметр a>0, которое может меняться от 0 до 1000. Значения массива res11[ii] записываются в массив res100[ii]

В прямой программе используются натуральный логарифм

```
double f1(double x, double left, double raight, double a0)
{
double norm,z;
z= log(x)/double(a0);
return z;
}
```

4) полученные значения масштабируются (умножаются на число 1000)

```
res101[ii]= res100[ii]*1000.0;
```

5)в файл 1011.txt записываются шифр из массива res101[ii]. Массив res101[ii] и является шифром. Здесь применяется одна важная идея. Поскольку диапазон целых чисел значительно меньше диапазона действительных чисел с плавающей запятой двойной точности, то массив res101[ii] заполняется действительными числами двойной точности. При считывании чисел из массива res101[ii] может быть потеряна мантисса действительного числа. Это может привести к ошибке на этапе декодирования. Чтобы избежать снижения точности отдельно в текстовый файл1001.txt записывается целая часть числа res101[ii](как переменная double), а мантисса res101[ii] преобразуется в целое число с нужным количеством знаков точности и записывается вторым по счёту (как переменная double).

```
remove("1011.txt");
file=fopen("1011.txt","w");
for(i=0;i<=nn-1;i++)
{
aa=int(res101[i]);
bb=int ((res101[i]-aa)*1e8);
fprintf(file,"%.4lf\n", aa);
fprintf(file,"%.4lf\n", bb);
fclose(file);
printf(" \n");
```

6)открываем и копируем данные из текстового файла1011.txt , затем из каждой последующей пары целых чисел формируем целую часть и мантиссу

```
int sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1001.txt","r");
if(file==NULL)
```

```
{
printf(" not open file");
}
else
{
for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
 printf(" coder1(%d)=%.12lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
```

Далее столбцы кода до записи в текстовый файл и после чтения из него сравниваются coder1 и coder2(должны совпадать).

Прочитанные данные записываются в файл res102[i] .

7) проводим пошагово обратные операции res103[i]=res102[i]/1000.0; делим массив res102[i] на 1000 результат записывает в массив res103[i].

8)вызываем обратную функцию, в качестве которой используются прямая показательная функция

```
res104[i]=f2(res103[i],left,raight,m,n0);
double f2(double x, double left, double raight, double a0)
{
double z;
z=exp(a0*x);
return z;
}
```

9)проводим обратное линейное преобразование и возвращаемся к переменной x по формуле res14[i]=(res104[i]-left)/(raight-left);

10)

```
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
printf("start(%d) = %d  decoder(%d) = %d\n",i,res[i], i,res105[i]);
}
printf("\n");
```

11)проводим обратное преобразование ASCII

```
for(i=0;i<=nn-1;i++)
{
printf("%c",res105[i] );
}
```

12) в результате программой возвращается исходная символьная фраза char str[nn+1]= "Polotsk State University 1234567890" .

Таким образом, в программе использованы математические операции запись чисел с двойной точностью в текстовой файл, композиция, состоящая из преобразования подобия, нелинейных показательных и логарифмических преобразований. $x_1 = n/255, x_2 = a + (b-a)x_1,\ y = \ln x_2 / a0$.

Обратные преобразования

$$x_2 = \exp(ya0),, x_1 = \frac{x_2 - a}{b-a},\ n = 255x_1$$

6. **Описание работы вспомогательной программы создания общего сеансового ключа на базе протокола без передачи долговременного ключа.**

Пользуясь структурной схемой создания общего сеансового ключа, опишем работу программы. В программе выбраны параметры p=61; phi=p-1; a=13; b=7. Используется подпрограмма, позволяющая по заданному простому числу p, открытому первому ключу a найти второй закрытый ключ $\alpha : \alpha a \equiv 1 (\mathrm{mod}\ p)$:

```
int zakr(int a,int n)
{
int i,j,n1,n2,jj;
jj=0;
n1=10000;
n2=10000;
for(i=1;i<=n1;i++)
{
for(j=1;j<=n2;j++)
{
if(a*i-n*j==1)
{
jj=jj+1;
return i;
}
}
}
}
```

Параметр ключа – действительное число с 16 значащими цифрами и десятичной точкой, подлежащее шифрованию с помощью протокола без передачи долговременного ключа записывается в символьную строку:

```
char str[nn+1]="1223456786433.3456776654";
```

Далее программа посимвольно выводит строку на экран:

```
printf("\n");
for(i=0;i<=nn-1;i++)
```

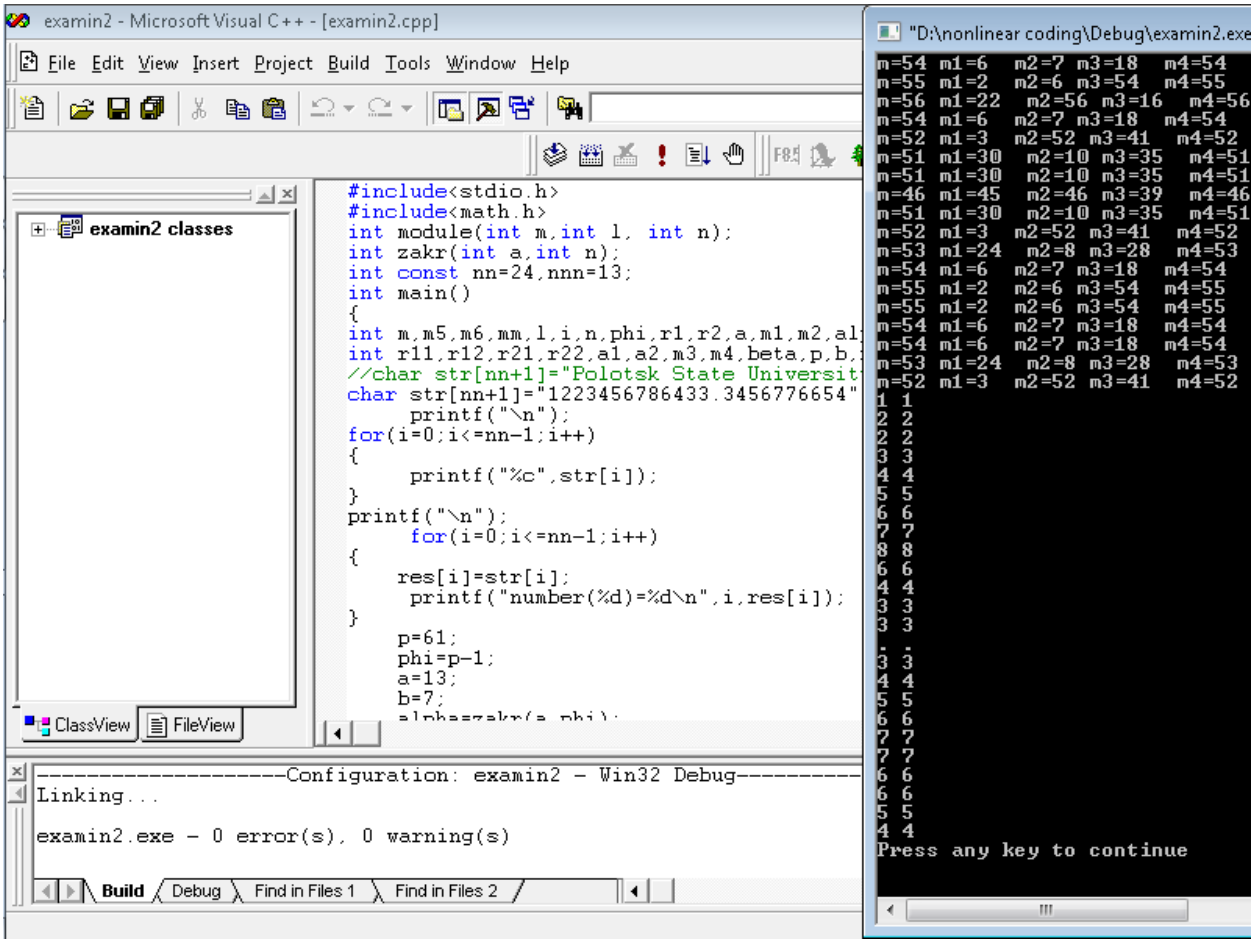
```
{
printf("%c",str[i]);
}
printf("\n");
```
Затем для каждого символа по циклу от 0 до nn-1 проводится 4 этапа шифрования ключа с 3 этапами передачи информации по каналу связи A-B, B-A, A- B:
```
for(i=0;i<=nn-1;i++)
{
res[i]=str[i];
printf("number(%d)=%d\n",i,res[i]);
}
p=61;
phi=p-1;
a=13;
b=7;
alpha=zakr(a,phi);
beta=zakr(b,phi);
printf("alpha=%d\n",alpha);
printf("beta=%d\n",beta);
for (i=0;i<=nn-1;i++)
{
m1=module(res[i],a,p);
m2=module(m1,b,p);
m3=module(m2,alpha,p);
m4=module(m3,beta,p);
res1[i]=m4;
printf("m=%d m1=%d  m2=%d m3=%d  m4=%d \n",res[i],m1,m2,m3,m4);
}
```
Проводится проверка совпадений символов начального и конечного m=m4
```
for(i=0;i<=nn-1;i++)
{
 printf("%c %c\n",str[i],res1[i]);
}
}
```
Сверяются числа – параметры ключа в начальной символьной строке и после 4 – раундного шифрования – дешифрования. Полный текст вспомогательной программы приведен в приложении.

В приведенном ниже скриншоте справа для каждого символа символьной строки"1223456786433.3456776654" указаны все 4 этапа шифрования. Согласно алгоритму протокола без передачи долговременного ключа равны значения m=m4, что мы и видим.

Кроме того в проекте программы Visual Studio C++ видна запись строки

char str[nn+1]="1223456786433.3456776654". Затем эта же строка повторяется первым столбцом до шифрования и вторым столбцом после шифрования и дешифрования. Полное совпадение результатов при тестировании вспомогательной программы. В данном случае использованы параметры; p=61; phi=p-1;a=13;b=7.

Для указанных параметров программа вычислила также закрытые ключи абонентов $\alpha = 37; \beta = 43$.



7. Нелинейное шифрование тригонометрическими функциями. Оценка пространства шифров и пространства ключей. Тестирование программы.

Тестирование программы, размерность пространства шифров

1)Зададим в программе параметры

eps=0.0;n0=30.0;n0=n0+eps;m=2;pi=2.0*asin(1.0);

left =3.0/4.0;raight=1.0;

Из рисунка 1 мы видим, что коды символов исходного текста до записи в текстовый файл 1001.txt coder1 и после чтения из него coder2 совпадают в 13 верных знаков. Нечётные строки в1001.txt совпадают с целой частью coder1 и coder2 , а чётные строки в1001.txt совпадают с мантиссой coder1 и coder2. Таким образом, при записи и чтении чисел из файла1001.txt не теряется точность чисел(double). Из рисунка 2 видно, что символы строки char str[nn+1]="Polotsk State University 0123456789" , преобразованной универсальной таблицей ASCII (переменная start(i)) и дешифрованные символы(переменная decoder(i)) полностью совпадают. Что доказывает однозначность шифрования и дешифрования. Кроме того полностью совпадают фразы

"Polotsk State University 0123456789 на рисунках 1 и 2 т.е. до шифрования и после шифрования и дешифрования.

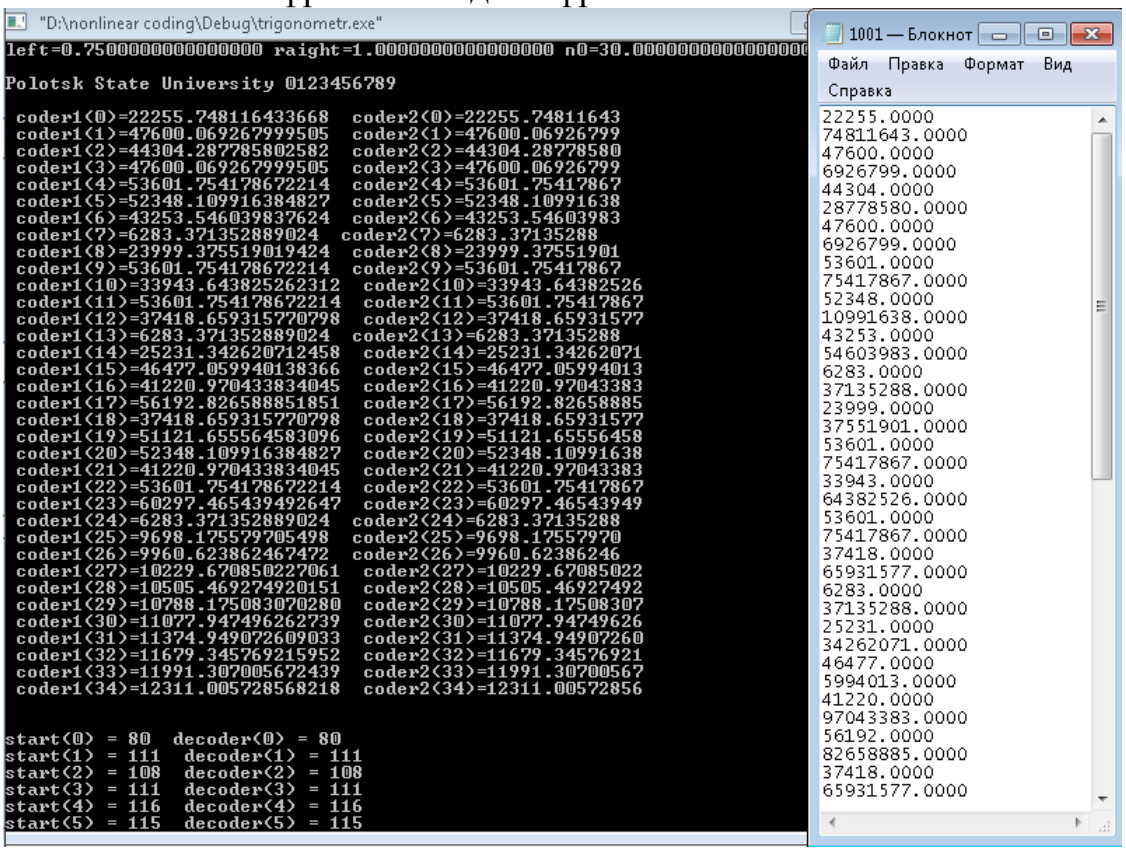


Рис.1

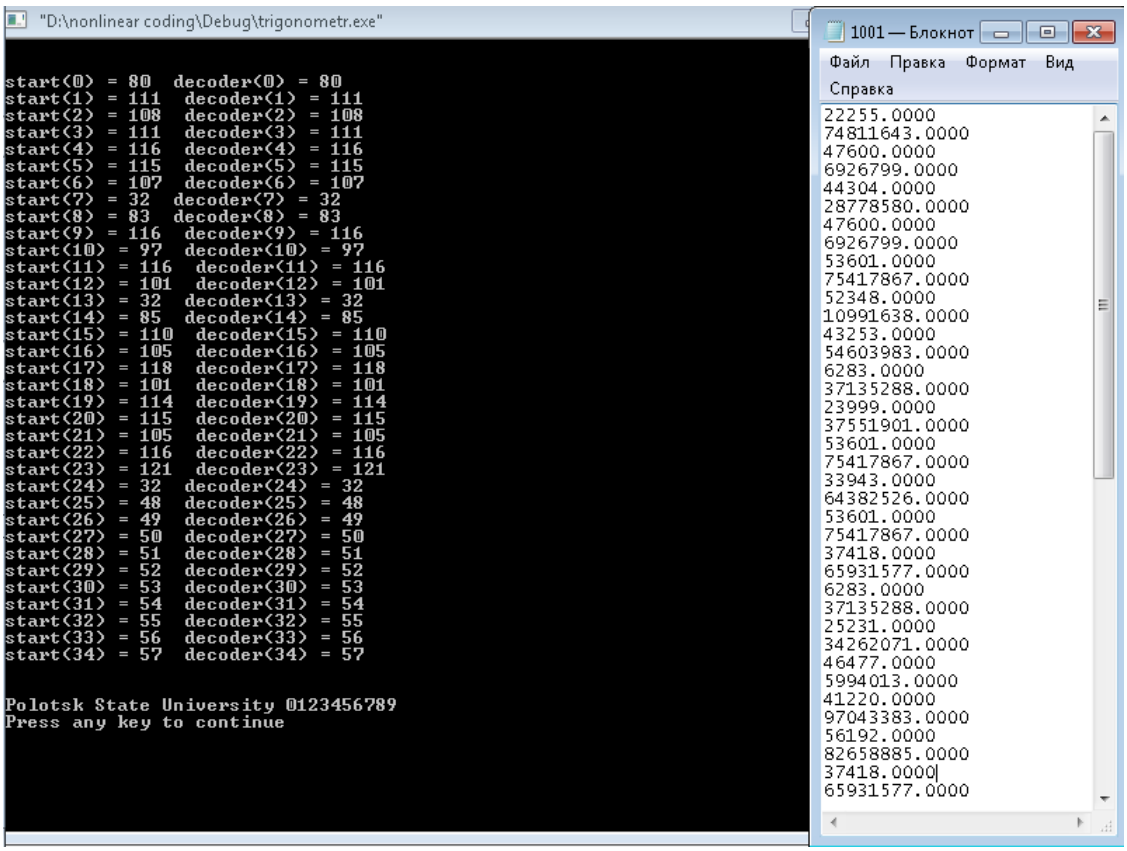


Рис.2

Оценим размерность пространства шифров по всем параметрам ключа, для чего увеличим степень n на 0.0000000001. В результате получаем шифр исходной фразы

"Polotsk State University 0123456789" на Рис.3 в консольном виде в проекте Visual Studio и в текстовом файле txt.1001. Сравнивая Рис.1 и Рис.3, по текстовым файлам txt.1001 видно, что мантиссы одних и тех же знаков , т.е. и шифров в целом различаются в 2 последних цифрах.

Поэтому чувствительность построенного в программе шифра для степени n составляет 10^{-10}. Учитывая диапазон применяемых степеней $n0 = 0 - 30$, получим $3*10^{11}$ различных ключей. Найдём чувствительность шифра по правой и левой границам $left, raight$. Прибавим число $eps = 10^{-10}$ к левой границе 0.75 по сравнению с Рис.1, оставляя остальные параметры программы неизменными eps=0.000000000001;

n0=30.0; m=2; left =3.0/4.0+eps;raight=1.0;

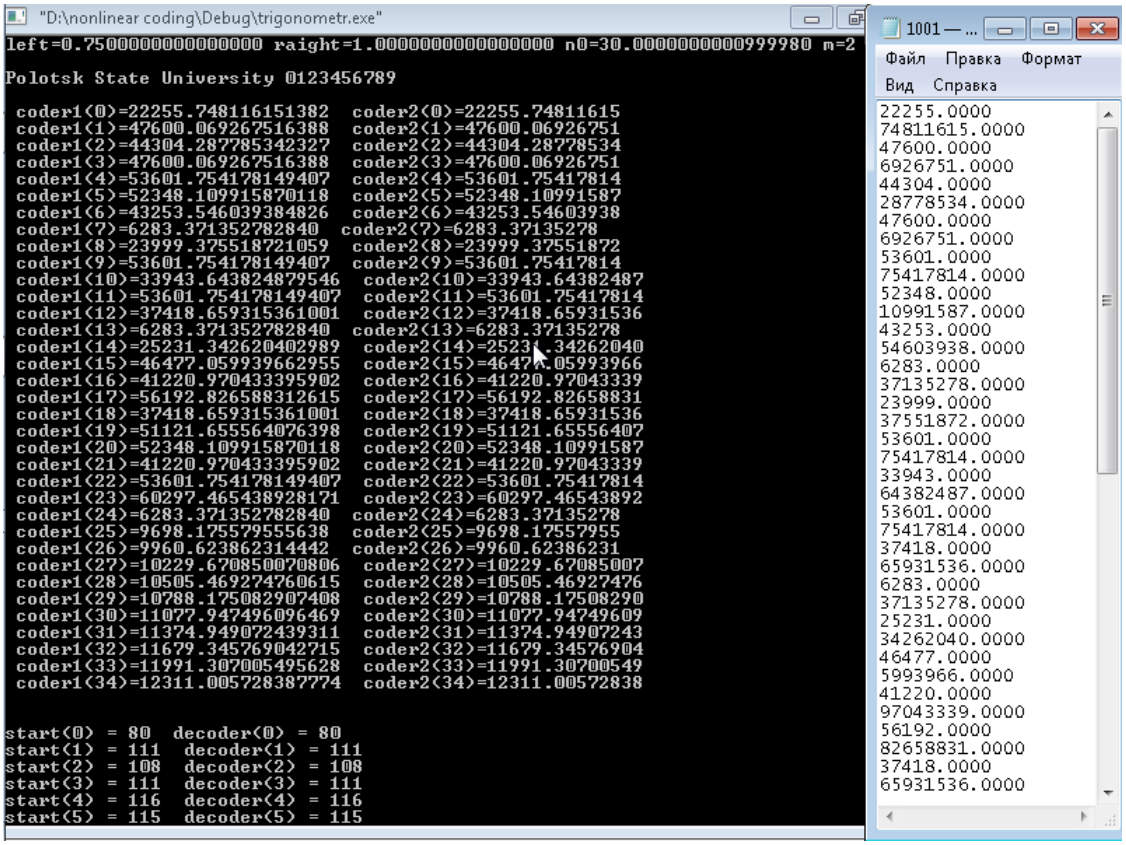

"D:\nonlinear coding\Debug\trigonometr.exe"
left=0.75000000000000000 raight=1.0000000000000000 n0=30.000000000999980 m=2
Polotsk State University 0123456789
coder1(0)=22255.748116151382 coder2(0)=22255.74811615
coder1(1)=47600.069267516388 coder2(1)=47600.06926751
coder1(2)=44304.287785342327 coder2(2)=44304.28778534
coder1(3)=47600.069267516388 coder2(3)=47600.06926751
coder1(4)=53601.754178149407 coder2(4)=53601.75417814
coder1(5)=52348.109915870118 coder2(5)=52348.10991587
coder1(6)=43253.546039384826 coder2(6)=43253.54603938
coder1(7)=6283.371352782840 coder2(7)=6283.37135278
coder1(8)=23999.375518721059 coder2(8)=23999.37551872
coder1(9)=53601.754178149407 coder2(9)=53601.75417814
coder1(10)=33943.643824879546 coder2(10)=33943.64382487
coder1(11)=53601.754178149407 coder2(11)=53601.75417814
coder1(12)=37418.659315361001 coder2(12)=37418.65931536
coder1(13)=6283.371352782840 coder2(13)=6283.37135278
coder1(14)=25231.342620402989 coder2(14)=25231.34262040
coder1(15)=46477.059939662955 coder2(15)=46477.05993966
coder1(16)=41220.970433395902 coder2(16)=41220.97043339
coder1(17)=56192.826588312615 coder2(17)=56192.82658831
coder1(18)=37418.659315361001 coder2(18)=37418.65931536
coder1(19)=51121.655564076398 coder2(19)=51121.65556407
coder1(20)=52348.109915870118 coder2(20)=52348.10991587
coder1(21)=41220.970433395902 coder2(21)=41220.97043339
coder1(22)=53601.754178149407 coder2(22)=53601.75417814
coder1(23)=60297.465438928171 coder2(23)=60297.46543892
coder1(24)=6283.371352782840 coder2(24)=6283.37135278
coder1(25)=9698.175579555638 coder2(25)=9698.17557955
coder1(26)=9960.623862314442 coder2(26)=9960.62386231
coder1(27)=10229.670850070806 coder2(27)=10229.67085007
coder1(28)=10505.469274760615 coder2(28)=10505.46927476
coder1(29)=10788.175082907408 coder2(29)=10788.17508290
coder1(30)=11077.947496096469 coder2(30)=11077.94749609
coder1(31)=11374.949072439311 coder2(31)=11374.94907243
coder1(32)=11679.345769042715 coder2(32)=11679.34576904
coder1(33)=11991.307005495628 coder2(33)=11991.30700549
coder1(34)=12311.005728387774 coder2(34)=12311.00572838
start(0) = 80 decoder(0) = 80
start(1) = 111 decoder(1) = 111
start(2) = 108 decoder(2) = 108
start(3) = 111 decoder(3) = 111
start(4) = 116 decoder(4) = 116
start(5) = 115 decoder(5) = 115
1001 — ...
Файл Правка Формат
Вид Справка
22255.0000
74811615.0000
47600.0000
6926751.0000
44304.0000
28778534.0000
47600.0000
6926751.0000
53601.0000
75417814.0000
52348.0000
10991587.0000
43253.0000
54603938.0000
6283.0000
37135278.0000
23999.0000
37551872.0000
53601.0000
75417814.0000
33943.0000
64382487.0000
53601.0000
75417814.0000
37418.0000
65931536.0000
6283.0000
37135278.0000
25231.0000
34262040.0000
46477.0000
5993966.0000
41220.0000
97043339.0000
56192.0000
82658831.0000
37418.0000
65931536.0000

Рис.3

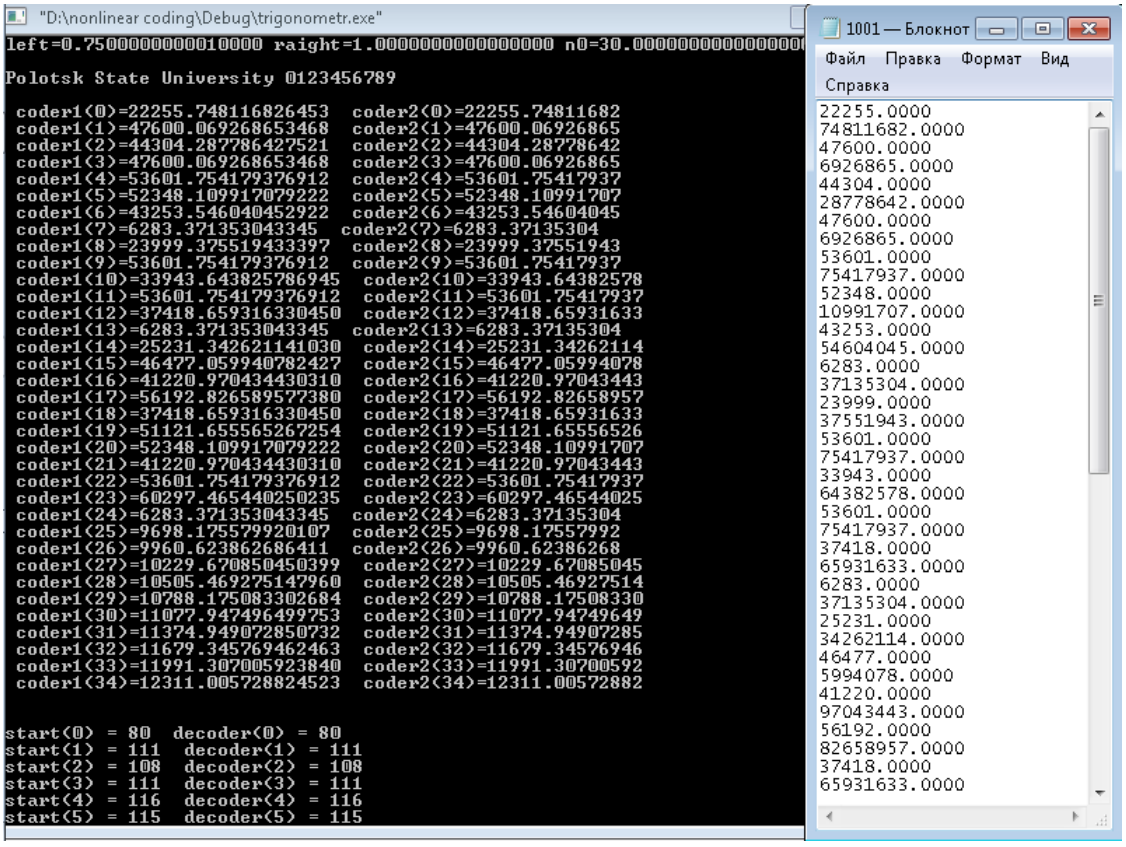

"D:\nonlinear coding\Debug\trigonometr.exe"
left=0.75000000000010000 raight=1.0000000000000000 n0=30.000000000000000
Polotsk State University 0123456789
coder1(0)=22255.748116826453 coder2(0)=22255.74811682
coder1(1)=47600.069268653468 coder2(1)=47600.06926865
coder1(2)=44304.287786427521 coder2(2)=44304.28778642
coder1(3)=47600.069268653468 coder2(3)=47600.06926865
coder1(4)=53601.754179376912 coder2(4)=53601.75417937
coder1(5)=52348.109917079222 coder2(5)=52348.10991707
coder1(6)=43253.546040452922 coder2(6)=43253.54604045
coder1(7)=6283.371353043345 coder2(7)=6283.37135304
coder1(8)=23999.375519433397 coder2(8)=23999.37551943
coder1(9)=53601.754179376912 coder2(9)=53601.75417937
coder1(10)=33943.643825786945 coder2(10)=33943.64382578
coder1(11)=53601.754179376912 coder2(11)=53601.75417937
coder1(12)=37418.659316330450 coder2(12)=37418.65931633
coder1(13)=6283.371353043345 coder2(13)=6283.37135304
coder1(14)=25231.342621141030 coder2(14)=25231.34262114
coder1(15)=46477.059940782427 coder2(15)=46477.05994078
coder1(16)=41220.970434430310 coder2(16)=41220.97043443
coder1(17)=56192.826589577380 coder2(17)=56192.82658957
coder1(18)=37418.659316330450 coder2(18)=37418.65931633
coder1(19)=51121.655565267254 coder2(19)=51121.65556526
coder1(20)=52348.109917079222 coder2(20)=52348.10991707
coder1(21)=41220.970434430310 coder2(21)=41220.97043443
coder1(22)=53601.754179376912 coder2(22)=53601.75417937
coder1(23)=60297.465440250235 coder2(23)=60297.46544025
coder1(24)=6283.371353043345 coder2(24)=6283.37135304
coder1(25)=9698.175579920107 coder2(25)=9698.17557992
coder1(26)=9960.623862686411 coder2(26)=9960.62386268
coder1(27)=10229.670850450399 coder2(27)=10229.67085045
coder1(28)=10505.469275147960 coder2(28)=10505.46927514
coder1(29)=10788.175083302684 coder2(29)=10788.17508330
coder1(30)=11077.947496499753 coder2(30)=11077.94749649
coder1(31)=11374.949072850732 coder2(31)=11374.94907285
coder1(32)=11679.345769462463 coder2(32)=11679.34576946
coder1(33)=11991.307005923840 coder2(33)=11991.30700592
coder1(34)=12311.005728824523 coder2(34)=12311.00572882
start(0) = 80 decoder(0) = 80
start(1) = 111 decoder(1) = 111
start(2) = 108 decoder(2) = 108
start(3) = 111 decoder(3) = 111
start(4) = 116 decoder(4) = 116
start(5) = 115 decoder(5) = 115
1001 — Блокнот
Файл Правка Формат Вид
Справка
22255.0000
74811682.0000
47600.0000
6926865.0000
44304.0000
28778642.0000
47600.0000
6926865.0000
53601.0000
75417937.0000
52348.0000
10991707.0000
43253.0000
54604045.0000
6283.0000
37135304.0000
23999.0000
37551943.0000
53601.0000
75417937.0000
33943.0000
64382578.0000
53601.0000
75417937.0000
37418.0000
65931633.0000
6283.0000
37135304.0000
25231.0000
34262114.0000
46477.0000
5994078.0000
41220.0000
97043443.0000
56192.0000
82658957.0000
37418.0000
65931633.0000

Рис.4

Сравнивая рисунок 1 и рисунок 4, мы видим по текстовым файлам txt.1001, что шифры мантисс шифров различаются в 2 последних знаках. Учтём эквивалентность вхождения в формулы переменных $0 \le left, raight \le 1$. Получаем за счёт каждой переменной $left, raight$ увеличения пространства шифров в 10^{12} раз. Имеем, что всё пространство шифров состоит из числа шифров $3*10^{11}*10^{12}*10^{12} = 3*10^{35}$.

Очевидно, что размерность пространства ключей (различных дешифрованных фраз при фиксированной входной текстовой фразе) меньше пространства шифрованных фраз. Оценим размерность пространства ключей. В декодированной фразе необходимо изменить как минимум 1 символ. Если параметр ключа изменяется на единичном отрезке, то необходимое изменение параметра составит $\sim 10^{-2}$. Один ключ содержит 3 независимых параметра. Следовательно, пространство ключей имеет размерность 10^6. Если при кодировании используется несколько ключей(3), то имеем 9 независимых изменяемых параметра с пространством ключей 10^{18}

Пусть суперкомпьютер подбирает ключи к шифру со скоростью $10^9 \frac{1}{сек}$ - порядок тактовой частоты современных процессоров. Тогда ему понадобится времени $t \sim 10^{18}/10^9 = 10^9 сек = 10^9/(3600*24*30*12) = 32 года$. Применением большого числа ключей можно увеличить пространство ключей и время перебора до времени существования звёздного скопления Млечный Путь $\approx 10^{13} лет$ (5 используемых ключей).

Поэтому применение тригонометрических функций оправдано, так как имеет огромное пространство ключей по трём переменным $left, raight, n$

8.Нелинейное шифрование показательными и логарифмическими функциями. Оценка пространства ключей. Тестирование программы.

Тестирование программы, размерность пространства ключей и пространства шифров

1)Зададим в программе параметры eps=0.0;left =1.0+eps;raight=10.0; a0=5.0.

Из рисунка 5 мы видим, что коды символов исходного текста до записи в текстовый файл 1011.txt coder1 и после чтения из него coder2 совпадают в 13 верных знаков. Нечётные строки в1011.txt совпадают с целой частью coder1 и coder2 , а чётные строки в1011.txt совпадают с мантиссой coder1 и coder2. Таким образом, при записи и чтении чисел из файла1001.txt не теряется точность чисел(double). Из рисунка 5 видно, что символы строки

char str[nn+1]="Polotsk State University 0123456789" , преобразованной универсальной таблицей ASCII (переменная start(i)) и дешифрованные символы(переменная decoder(i)) полностью совпадают. Что доказывает однозначность шифрования и дешифрования. Кроме того полностью совпадают фразы

"Polotsk State University 0123456789 на рисунках 5 и 6 т.е. до шифрования и после шифрования и дешифрования.

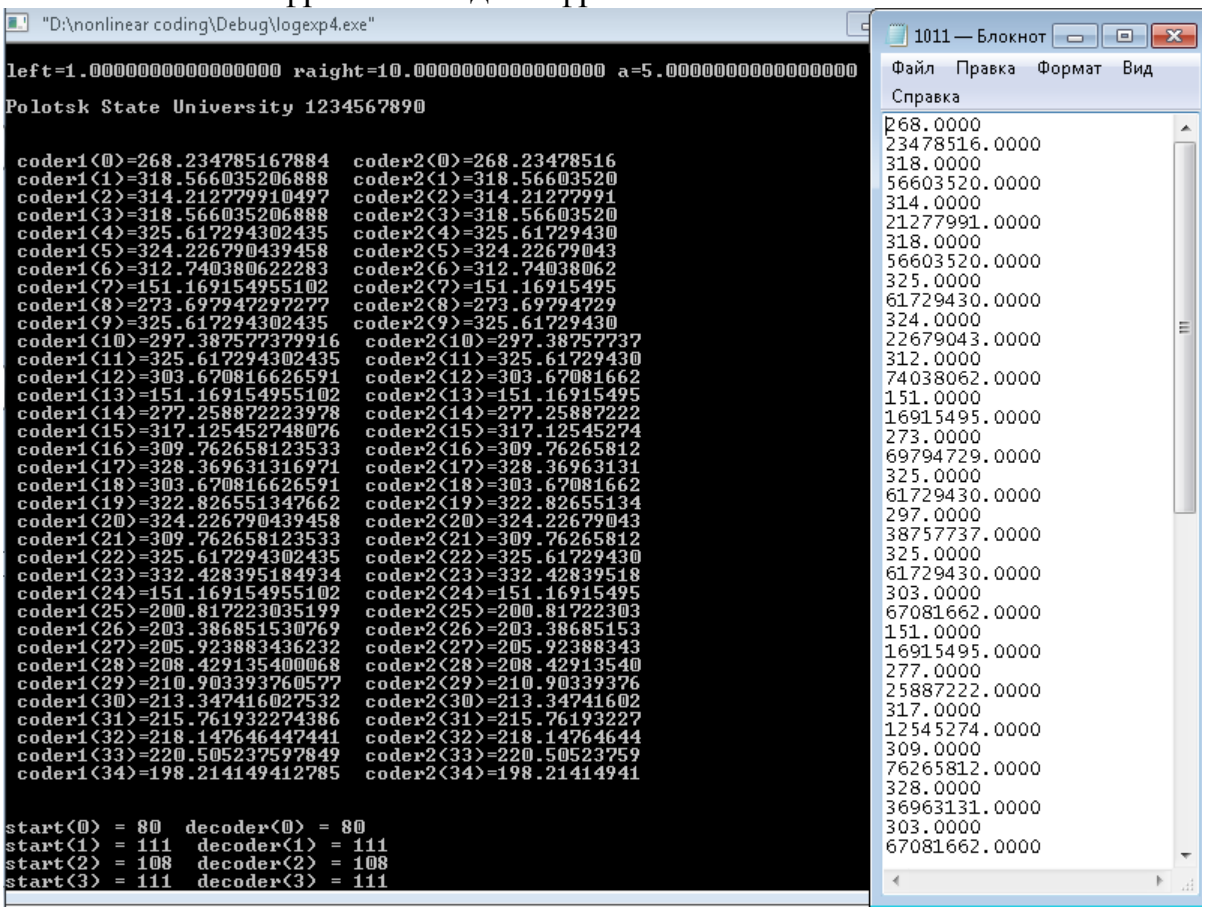


Рис.5

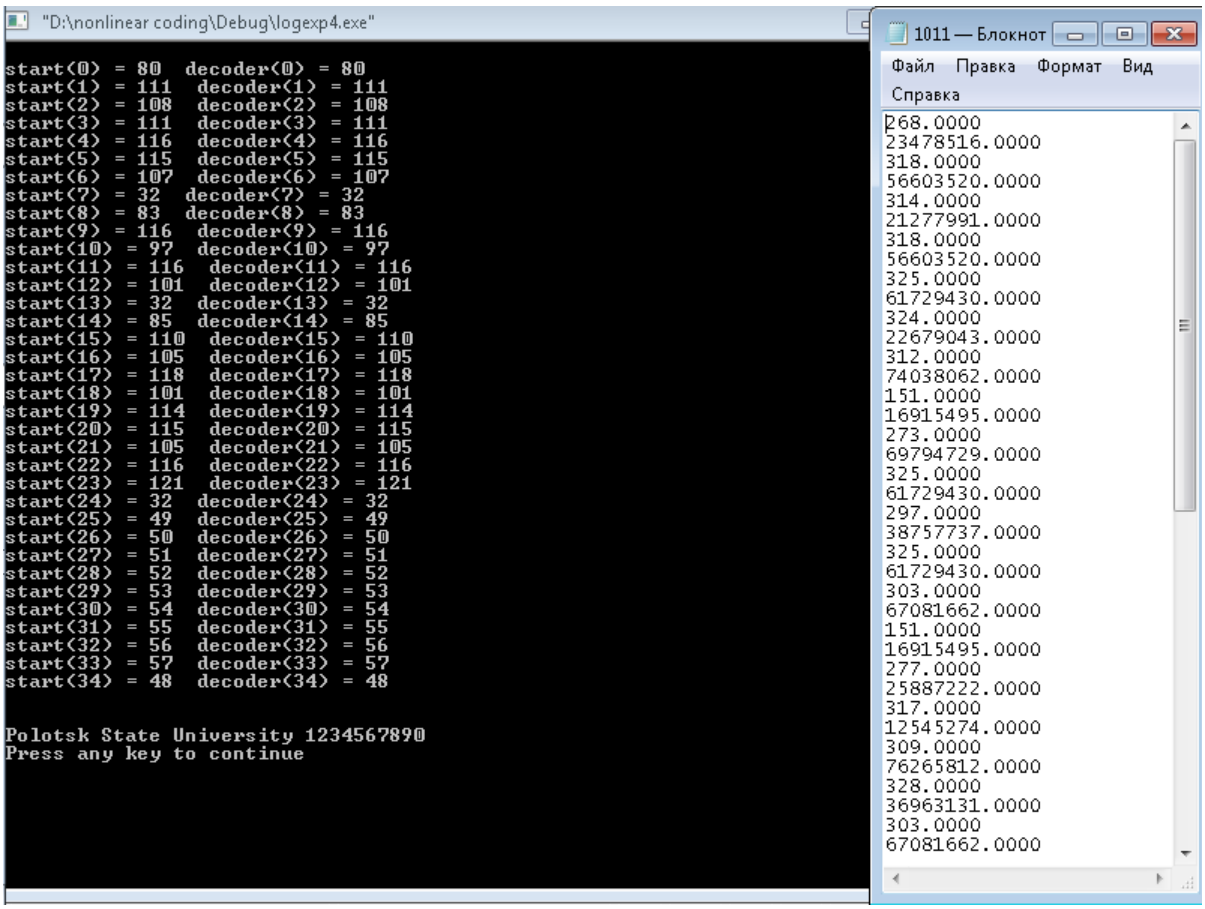


Рис.6

Оценим пространство шифров по всем параметрам ключа, для чего увеличим параметр a на 0.0000000002. В результате получаем шифр исходной фразы

"Polotsk State University 0123456789" на Рис.7 в консольном виде в проекте Visual Studio и в текстовом файле txt.1011. Сравнивая Рис.5 и Рис.7, по текстовым файлам txt.1011 видно, что мантиссы одних и тех же символов, т.е. и шифров в целом различаются в 1 последней цифре.

Поэтому чувствительность построенного в программе шифра для параметра a составляет $1/0.0000000002 = 5*10^9$. Учитывая диапазон применяемых степеней $=1-1000$, получим $5*10^{12}$ различных шифров. Найдём чувствительность шифра по правой и левой границам $left, raight$. Прибавим число $eps = 10^{-10}$ к левой границе 1.0 ($eps = 0$ на Рис.1). Остальные параметры программы неизменные.

```
eps=0.0000000001;
left =1.0+eps;
raight=10.0;
a0=5.0;
```

Результаты шифрования приведены на рис.8.

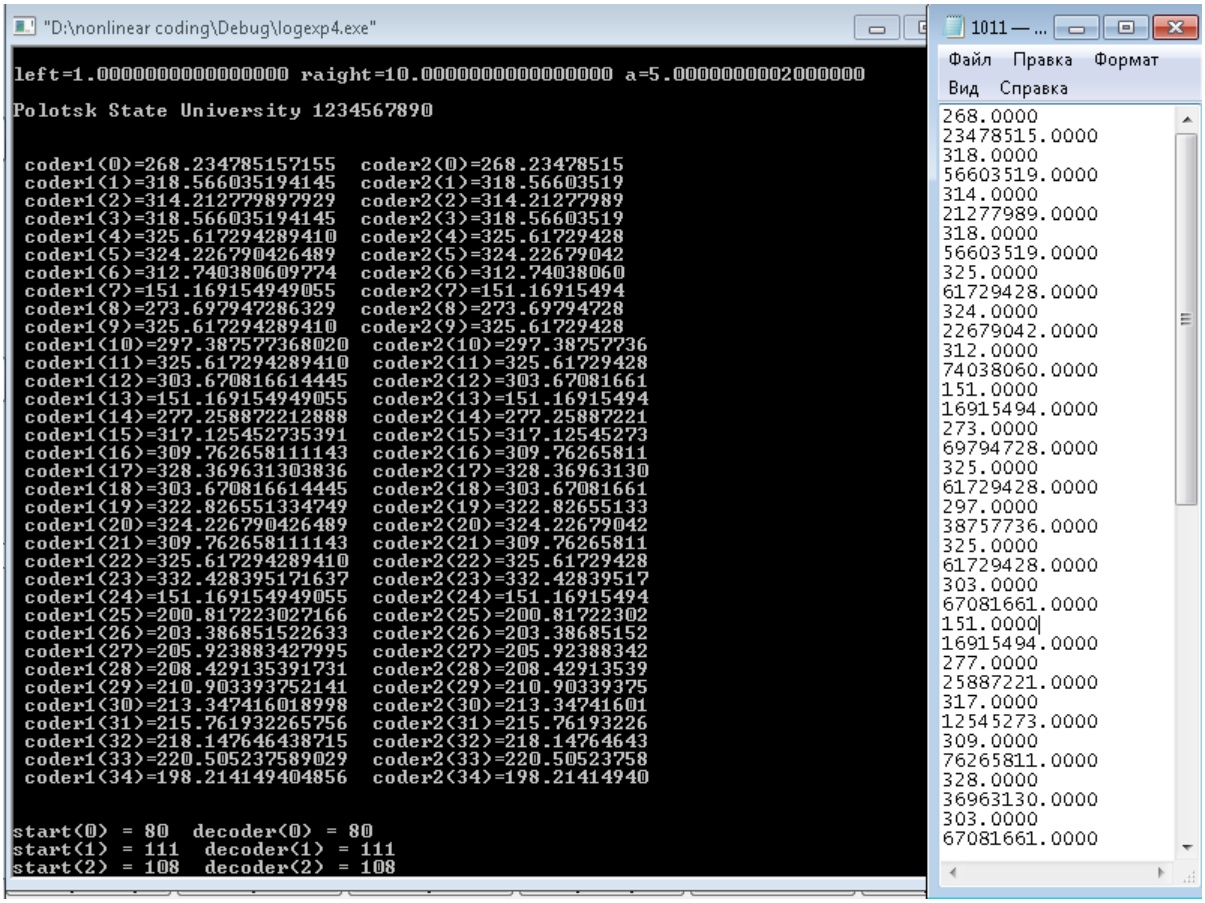


Рис.7

Сравнивая рисунок 5 и рисунок 8, мы видим по текстовым файлам txt.1011, что шифры мантисс шифров различаются в 1 последнем знаке. Учтём эквивалентность вхождения в формулы переменных $1 \le left, raight \le 10$. Получаем за счёт каждой переменной $left, raight$ увеличение пространства шифров в 10^{11}раз. Имеем, что всё пространство ключей состоит из числа шифров $5*10^{12}*10^{11}*10^{11} = 5*10^{34}$.

Очевидно, что размерность пространства ключей (различных дешифрованных фраз при фиксированной входной текстовой фразе) меньше пространства шифрованных фраз. Оценим размерность пространства ключей. В декодированной фразе необходимо изменить как минимум 1 символ. Если параметр ключа изменяется на единичном отрезке, то необходимое изменение параметра составит $\sim 10^{-2}$. Один ключ содержит 3 независимых параметра, следовательно, пространство ключей имеет размерность10^6. Если при кодировании используется несколько ключей(3), то имеем 9 независимых изменяемых параметра с пространством ключей10^{18}

Пусть суперкомпьютер подбирает ключи к шифру со скоростью $10^9 \frac{1}{сек}$ - порядок тактовой частоты современных процессоров. Тогда ему понадобится времени $t \sim 10^{18}/10^9 = 10^9 сек = 10^9/(3600*24*30*12) = 32 года$. Применением большого числа ключей можно увеличить пространство ключей и время перебора до времени существования звёздного скопления Млечный Путь $\approx 10^{13} лет$ (5 используемых ключей).

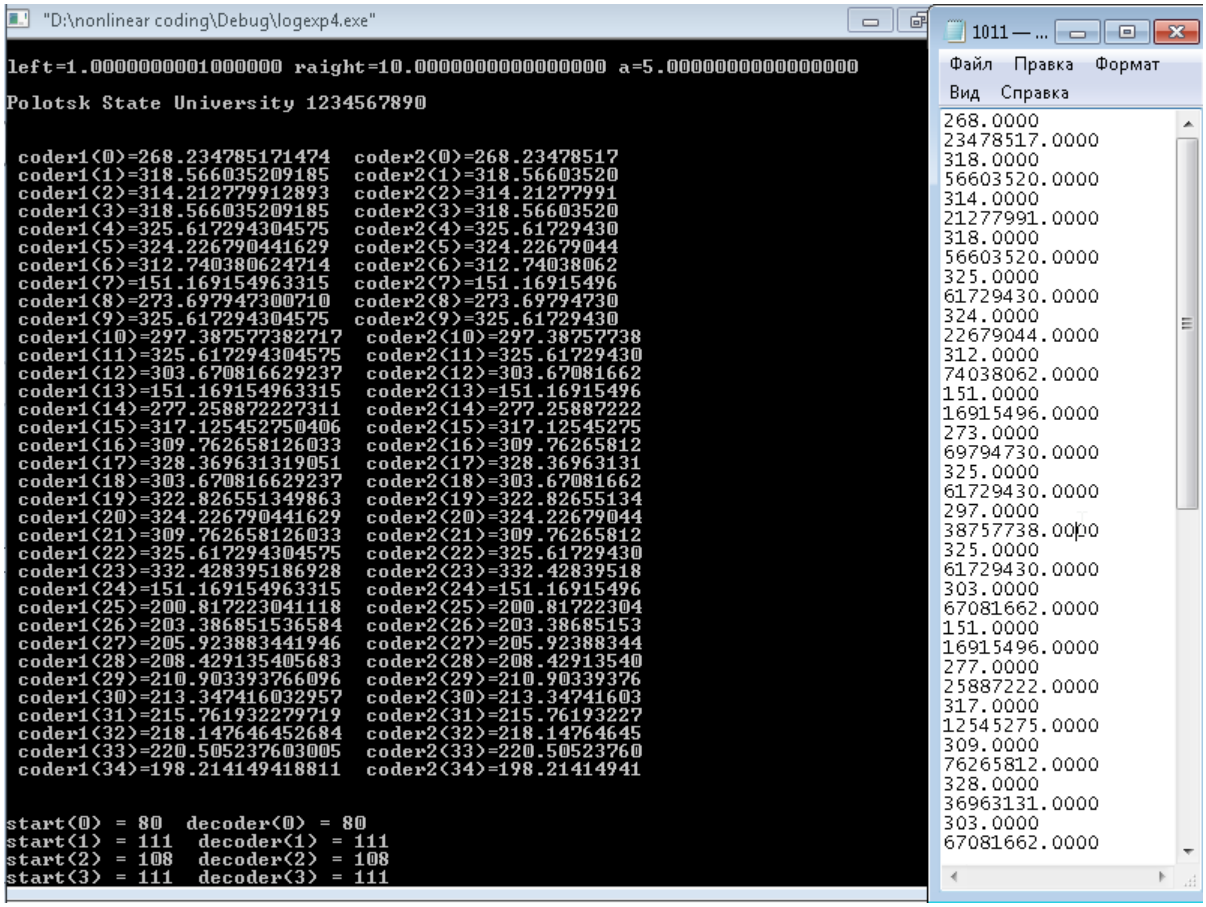


Рис.8

Поэтому применение показательных и логарифмических функций оправдано, так как имеет огромное пространство ключей по трём переменным $left, raight, a$.

Другими словами, взлом шифра без знания параметров ключа ($left, raight, m, n$)(9 параметров для тройного ключа) для суперкомпьютера невозможен из – за большой размерности пространства ключей.

9.Интерфейс программы. Тестирование.

Ядро интерфейса программы (основная программа) написана на языке C++, которая подключается к интерфейсу, использующую программную среду разработки Visual Studio 6.6. В среде Visual Studio 6.6 создаётся динамическая библиотека DLL, использующая как функции интерфейса, его входные и выходные параметры, так и основные программы для шифрования и дешифрования, написанные на языке C++6.0. Для конструкции интерфейса в Visual Studio 6.6 применяется уникальная библиотека XEFORT, написанная югославскими программистами.

На Рис.9 представлен удобный для пользователя при шифровании данных нелинейными функциями с уплотнённым шифром.

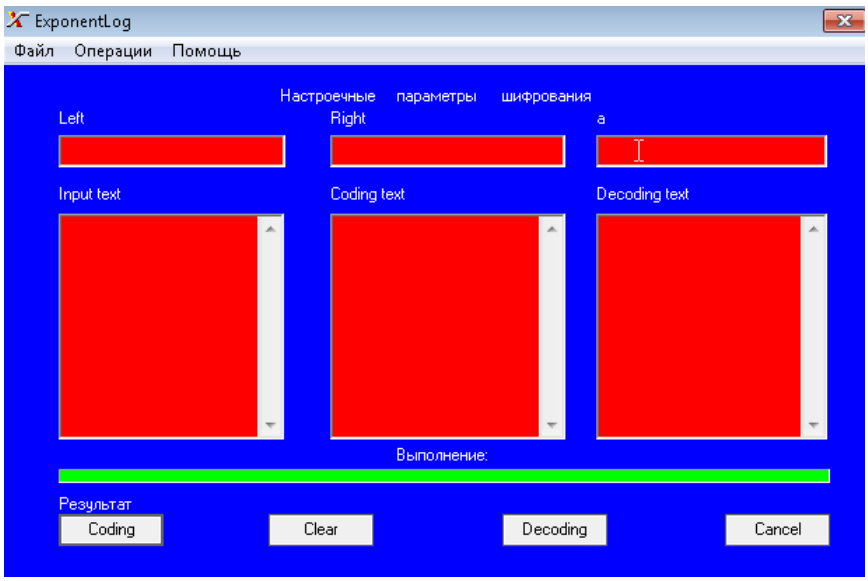


Рис.9

В интерфейсе 6 редактируемых окон – параметры ключа шифрования *Left*, *Right*, *a* левая и правая границы отрезка области определения и параметр показательной функции a. Все три параметра считываются как действительные числа с двойной точностью и десятичной точкой в качестве запятой. Ниже расположены 3 редактируемых текстовых окна – окно ввода исходного текста *Innput text* , в котором допустимыми символами являются все буквы английского текста (малые и большие), цифры, знаки препинания – .,<>?' "\|_-+=!@#$%^&*()[]{} и т.д., цифры, невидимые разделительные знаки табуляции, пробела, возврат каретки на новую строку и т.д., В среднее окно из основной программы загружается шифр *Coding text* исходного текста, в нижнее правое окно отображается декодируемый текст *Decoding text* .

Для большого количества символов кодируемого текста все 3 редактируемых нижних окна снабжены колесом прокрутки. В интерфейсе присутствует также линейка выполнения процесса в процентах (зелёным цветом). В нижней части интерфейса расположены 4 исполняемых статических окна. *Coding* - кодирование, *Decoding* - декодирование, *Clear* - очистить, *Cancel* - выход.

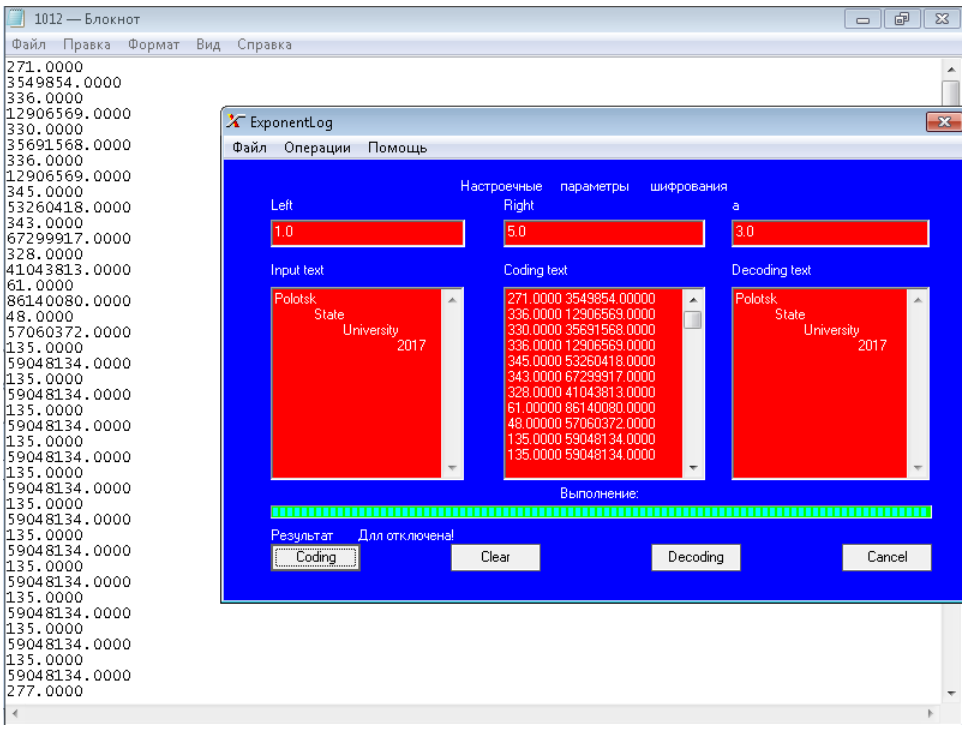


Рис.10

На Рис.10 показан пример тестирования интерфейса $Left = 1.0; Right = 5.0; a = 3.0$. В текстовое редактируемое окно *Innput text* вводимый текст *Polotsk State University* 2017, который расположен вдоль главной диагонали окна ”*каскадом*” для чего использовались многочисленные невидимые знаки пробела и табуляции при записи текста. В среднем нижнем окне *Coding text* в два столбца расположены шифры набранных символов – в первом столбце целая часть действительного числа, во втором столбце мантисса действительного числа, соответствующих каждому символу расположенные построчно. Основная программа создаёт также текстовый файл 12.*txt* с шифрованным текстом, с единственным отличием от интерфейса, заключающимся в том, что целая часть числа и его мантисса расположены друг за другом в один столбец. Сравнивая шифры на Рис.10 записанные в окно *Coding text* интерфейса и в текстовый файл 1012.*txt*, мы видим, что они полностью совпадают.

Видно также полное совпадение исходного текста *Innput text* и дешифрованного *Decoding text* текста, учитывая символы английского алфавита, цифровую символику и невидимые символы табуляции и пробелов.

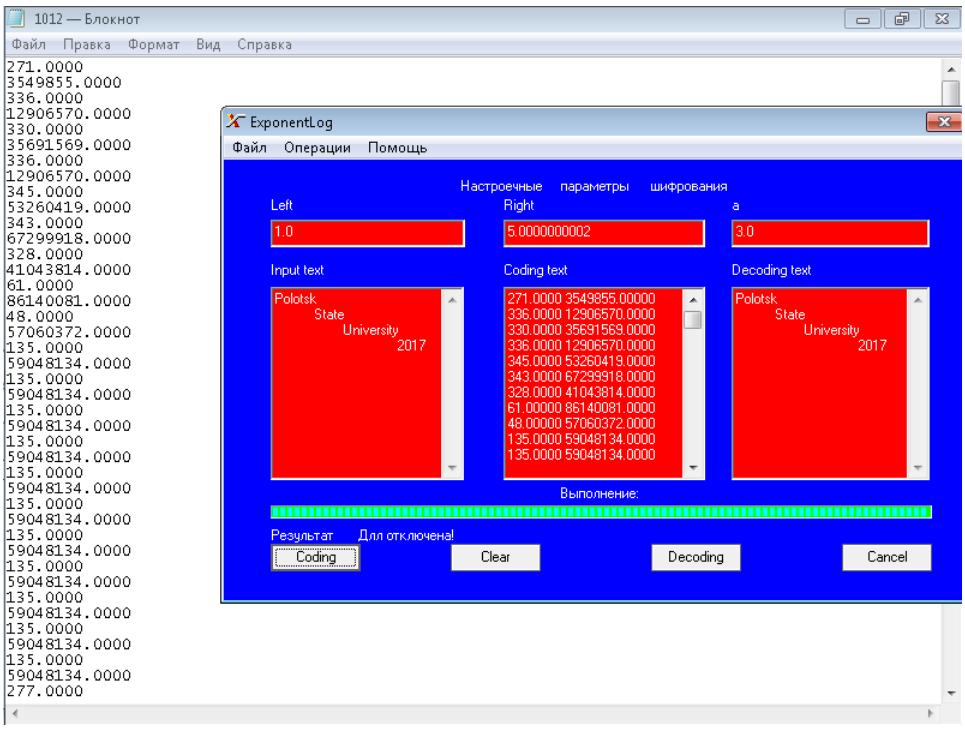


Рис.11

Рис.11 показывает результат второго примера тестирования интерфейса. А именно, сравнивая рисунки 10 и 11, мы можем сделать выводы о чувствительности уплотнённого шифра при использовании быстрорастущих нелинейных функций к изменению параметров сеансового ключа. Текстовые файлы1012.*txt* на Рис.10 и Рис.11 имеют шифры, которые различаются мантиссами шифров только в последнем знаке. Такой же вывод следует из сравнения шифров в окне *Coding text* Рис.10 и Рис.11. Но параметр a на Рис.10 и Рис.11. различается на 0.0000000002!

Мы можем определить чувствительность пространства шифра по формуле $sensivity\,coding = \frac{1}{a_2 - a_1} = \frac{1}{0.0000000002} = 5*10^9 \frac{шифр}{1интервал\,параметра}$

Действительно, мы имеем дело с очень плотной упаковкой шифров $\approx 10^9 \frac{шифр}{1интервал\,параметра}$ (“очень плотные или резонансные шифры”).

В то время как шифрование с использованием целых чисел позволяет применить единственный шифр на том же единичном интервале изменения параметра.

Сравнивая по отдельности шифры в текстовых файлах 1012.*txt* и окнах интерфейса *Coding text* на Рис.10 и Рис.12 мы видим, что при изменении параметра ключа a на 0.0000000005 мантиссы шифров изменились также в последнем знаке. Т.е. чувствительность шифра по всем 3 параметрам ключа $Left, Right, a$ по порядку величины одинакова и равна $\approx 10^9 \frac{шифр}{1интервал\,параметра}$.

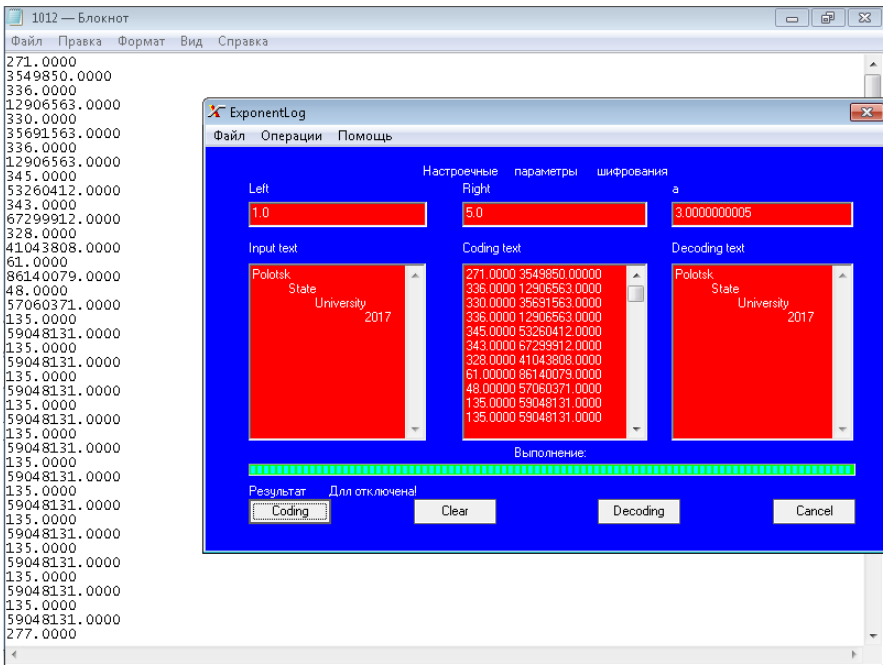


Рис.12.

Интерфейс – декодер

После пересылки по каналу связи текстового файла с шифрованным сообщением, адресат сообщения получает предварительно параметры ключа с помощью аутентификационного протокола описанного в пункте 2.2. Далее текстовый файл помещается в папку Debug программы декодера. Затем нажатием кнопки *Decoding* текст, загружённый в среднее нижнее окно, декодируется. Пример декодирования из текстового файла 12.*txt* фразы Polocku more than 1000 years с ключом $Left = 1.0; Right = 2.0; a = 5.0$ приведен на Рис. 13. На Рис.13 видно также, что

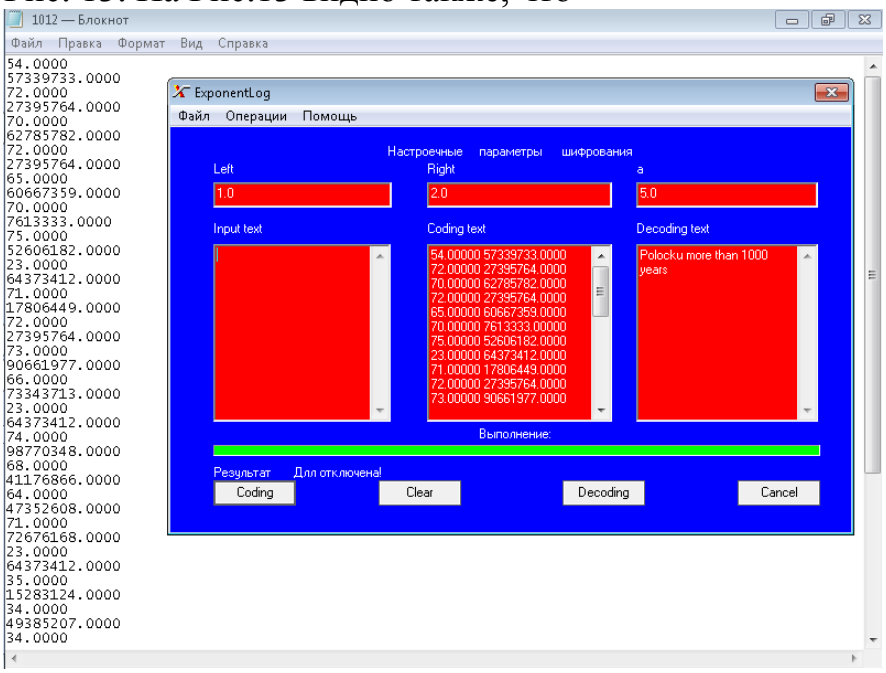


Рис.13

коды в текстовом файле 12.*txt* и в окне интерфейса *Coding text* полностью совпадают. Однако, если мы изменим параметры ключа на $Left = 1.0; Right = 5.0; a = 3.0000000001$ третий параметр всего на одну десяти миллиардную!, то из шрифта соответствующей фразе Polocku more than 1000 years, уже эту фразу при декодировании не получим. Но получим искажённую фразу Pomocku!more!than!1111!years.

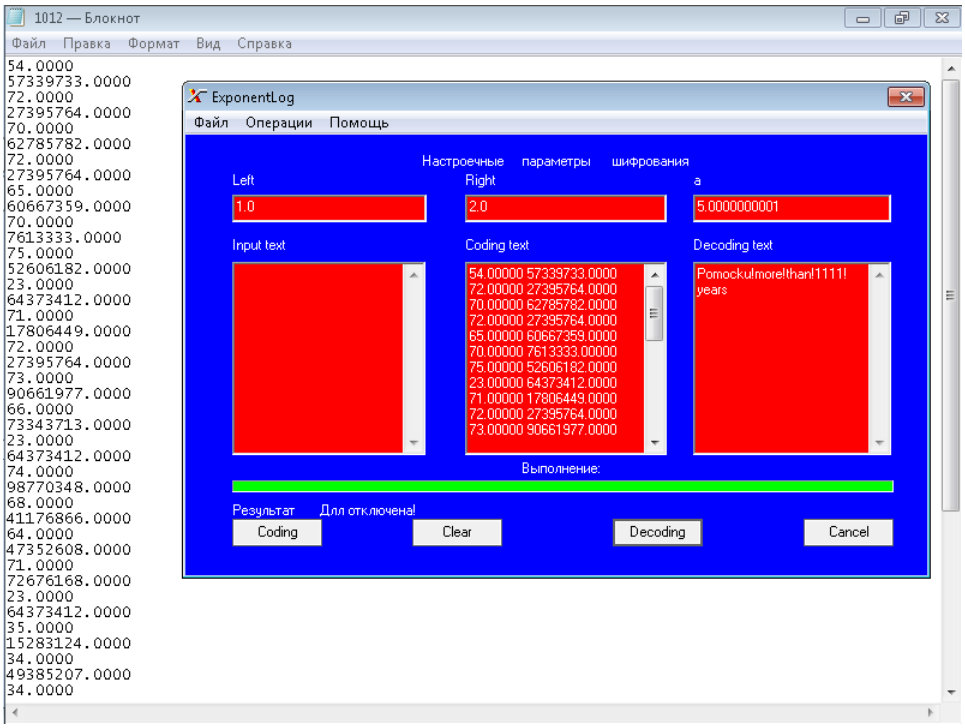


Рис.14

Сравнивая Рис.13 и Рис.14, мы видим, что использование нелинейных функций для шифрования возможно только с применением действительных чисел с двойной точностью.

10. Описание программ кодер – декодер с использованием тройного ключа

```
#include<stdio.h>
#include<iostream>
#include<fstream>
#include<algorithm>
#include<math.h>
using namespace std;
double f1(double x,double left, double raight ,double a0);
double f2(double x,double left, double raight, double a0);
int const n=28,nn=n;
int main()
```

```
{
int m,m5,m6,mm,l,i,ii,j,jj;
int r11,r12,r21,r22,a1,a2,m3,m4,beta,p,b,res[nn+1];
int  res105[nn+1] ,r;
double res100[nn+1], aa,bb,a01,a02, a03, eps,res103[nn+1],res104[nn+1],min,ll,x;
double res10[nn+1];
double left1,raight1, left3,raight3 ;
double left2,raight2,pi,res11[nn+1],res14[nn+1], res101[nn+1],res102[nn+1];
double res0104[nn+1], res011[nn+1], res0100[nn+1] , res0011[nn+1],
res00100[nn+1] ;
char str[nn+1]="Polocku more then 1000 years";
eps=0.0;
pi=2.0*asin(1.0);
left1 =1.0;
raight1=10.0;
left2 =2.0;
raight2=7.0;
left3 =3.0;
raight3=9.0;
a01=5.0;
a02=15.0;
a03=10.0;
        // в программе кодер – декодер используются три ключа с девятью
//независимо изменяемыми параметрами
left1,raight1,left2,raight2,left3,raight3,a01,a02,a03
for(i=0;i<=nn-1;i++)
{
 printf("%c",str[i] );
}
for(ii=0;ii<=nn-1;ii++)
{
res[ii]=str[ii];
res10[ii]=double(res[ii])/double(255);
res11[ii]=left1+(raight1-left1)*res10[ii];
res011[ii]=left2+(raight2-left2)*res10[ii];
res0011[ii]=left3+(raight3-left3)*res10[ii];
if(ii%3==0)
{
res100[ii]=f1(res11[ii],left1,raight1,a01);
res101[ii]= res100[ii]*1000.0;
}
else if (ii%3==1)
{
res0100[ii]=f1(res011[ii],left2,raight2,a02);
```

```
res101[ii]= res0100[ii]*1000.0;
}
 else if (ii%3==2)
{
res00100[ii]=f1(res0011[ii],left3,raight3,a03);
res101[ii]= res00100[ii]*1000.0;
}
// каждый первый, второй и третий символ входных данных шифруется
//своим ключом
printf("*(%d)=%d\n",ii,res[ii]);
}
FILE*file;
remove("1014.txt");
file=fopen("1014.txt","w");
for(i=0;i<=nn-1;i++)
{
aa=double(int(res101[i]));
bb=double(int ((res101[i]-aa)*1e8));
fprintf(file,"%.4lf\n", aa);
fprintf(file,"%.4lf\n", bb);
printf("a=%.1lf b=%.1lf c=%.16lf\n",aa,bb, res101[i] );
}
fclose(file);
double sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1014.txt","r");
if(file==NULL)
{
printf(" not open file");
          }
else
{
for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,2*nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
printf(" coder1(%d)=%.12lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
```

```
for(i=0;i<=nn-1;i++)
{
res103[i]=res102[i]/1000.0;
if(i%3==0)
{
res104[i]=f2(res103[i],left1,raight1,a01);
res14[i]=(res104[i]-left1)/(raight1-left1);
 }
if(i%3==1)
{
 res0104[i]=f2(res103[i],left2,raight2,a02);
res14[i]=(res0104[i]-left2)/(raight2-left2);
}
if(i%3==2)
{
res0104[i]=f2(res103[i],left3,raight3,a03);
 res14[i]=(res0104[i]-left3)/(raight3-left3);
}
 // каждый первый, второй и третий символ входных данных шифруется
//своим ключом

}
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
 printf("start(%d) = %d  decoder(%d) = %d\n",i,res[i], i,res105[i]);
 }
for(i=0;i<=nn-1;i++)
{
 printf("%c",res105[i] );
}
}
double f1(double x, double left, double raight, double a0)
{
double z;

z= log(x)/double(a0);
return z;
}

double f2(double x, double left, double raight, double a0)
{
double z;
z=exp(a0*x);
```

```
return z;
}
```

11.Программа декодера с тройным ключом

```
#include<stdio.h>
#include<iostream>
#include<fstream>
#include<algorithm>
#include<math.h>
using namespace std;
double f1(double x,double left, double raight ,double a0);
double f2(double x,double left, double raight, double a0);
int const n=28,nn=n;
int main()
{
int m,m5,m6,mm,l,i,ii,j,jj;
int r11,r12,r21,r22,a1,a2,m3,m4,beta,p,b,res[nn+1];
int  res105[nn+1] ,r;
double res100[nn+1], aa,bb,a01,a02,a03,eps,res103[nn+1],res104[nn+1],min,ll,x;
double res10[nn+1];
double left1,raight1, left3,raight3 ;
double left2,raight2,pi,res11[nn+1],res14[nn+1], res101[nn+1],res102[nn+1];
double res0104[nn+1], res00104[nn+1], res011[nn+1], res0100[nn+1],
res0011[nn+1], res00100[nn+1] ;
char str[nn+1]="Polocku more then 1000 years";
eps=0.0;
pi=2.0*asin(1.0);
left1 =1.0;
raight1=10.0;
left2 =2.0;
raight2=7.0;
left3 =3.0;
raight3=9.0;
a01=5.0;
a02=15.0;
a03=10.0;
FILE*file;
double sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1014.txt","r");
if(file==NULL)
{
```

```
printf(" not open file");
}
else
{
for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,2*nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
 printf(" coder2(%d)=%.8lf\n",i, res102[i] );
}
for(i=0;i<=nn-1;i++)
{
res103[i]=res102[i]/1000.0;
if(i%3==0)
{
res104[i]=f2(res103[i],left1,raight1,a01);
res14[i]=(res104[i]-left1)/(raight1-left1);
}
if(i%3==1)
{
 res0104[i]=f2(res103[i],left2,raight2,a02);
res14[i]=(res0104[i]-left2)/(raight2-left2);
}
if(i%3==2)
{
 res00104[i]=f2(res103[i],left3,raight3,a03);
res14[i]=(res00104[i]-left3)/(raight3-left3);
 }
}
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
printf(" decoder(%d) = %d\n", i,res105[i]);
 }
for(i=0;i<=nn-1;i++)
{
 printf("%c",res105[i] );
}
```

```
printf("\n");
}
double f1(double x, double left, double raight, double a0)
{
double z;
z= log(x)/double(a0);
return z;
}
double f2(double x, double left, double raight, double a0)
{
double z;
z=exp(a0*x);
return z;
}
```

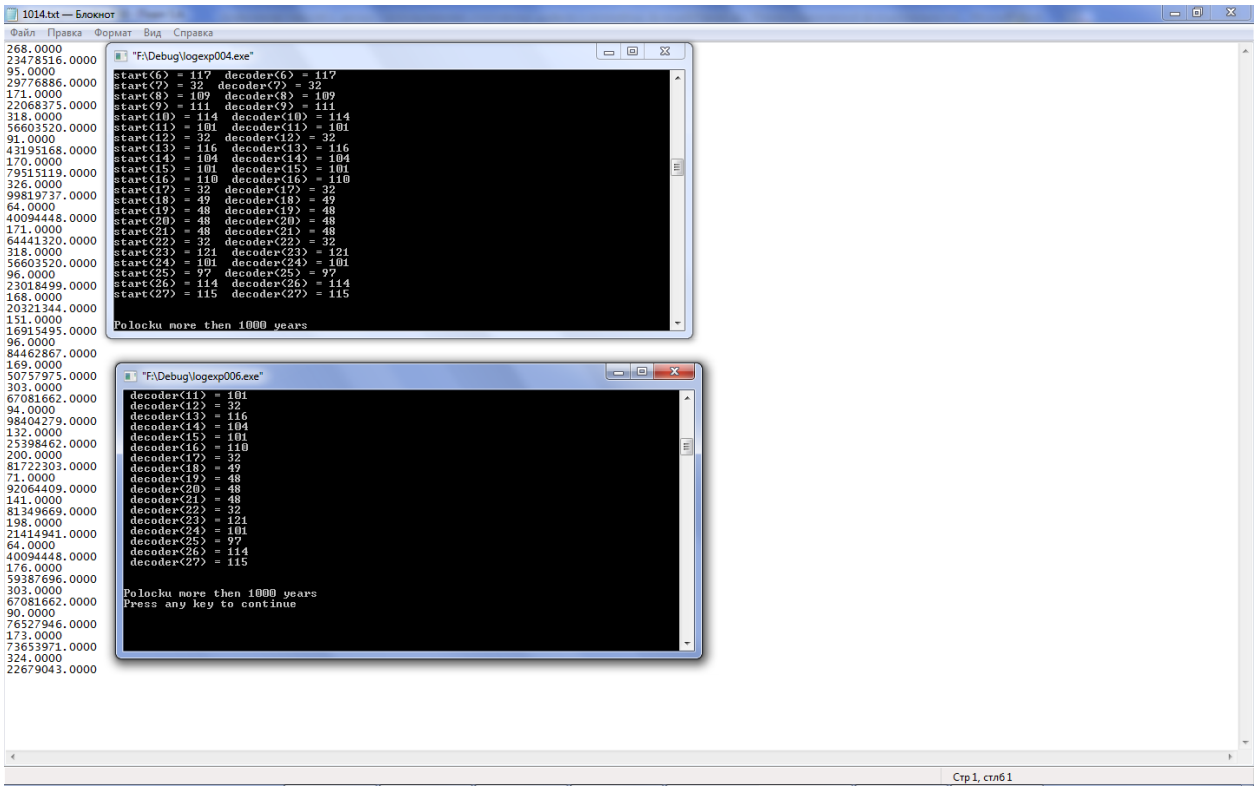


Рис.15

На Рис.15 видно, что обе программы кодер – декодер и отдельный декодер возвращают одну и ту и ту же символьную фразу Polocku more then 1000 years с помощью тройного ключа. Первая программа создаёт текстовый файл 1014.txt, записывает туда тройным ключом шифр, считывает из файла и декодирует каждый третий символ своим ключом. Из Рис. 15 1014.txt видно, что каждые первые, вторые, третьи строки 1014.txt шифруются своими ключами.

Отметим, что все три применяемых ключа являются контр ключами по отношению друг к другу. Действительно, пусть крипто аналитик расшифровал треть символьной фразы через три символа при благоприятных обстоятельствах, тогда однозначное дешифрование слова из 3 символов будет невозможным, так же как с первой русской буквы с могут начинаться слова сон, сок, сом, сам и т. д.

Литература

1.Лидовский В.В. Теория информации. – М.: Компания Спутник+.2004.- 111 с.

Приложения
1.Код основной программы 1 и библиотеки

```
#include<stdio.h>
#include<iostream>
#include<fstream>
#include<algorithm>
#include<stdlib.h>
#include<vector>
#include<iterator>
#include<math.h>
# define nnn 8000
using namespace std;
double f1(double x,double left, double raight , int m, double n);
double f2(double x,double left, double raight, int m, double n);
int const nn=35,nnnn=13,n=3;
int main()
{
int m,m5,m6,mm,l,i,ii,j,jj,n,phi,r1,r2,a,m1,m2,alpha,aa1,aa2,shifr[nn+1];
int
r11,r12,r21,r22,a1,a2,m3,m4,beta,p,b,res[nn+1],res1[nn+1],podp1[nnn+2],podp[nn
n+2];
int  res105[nn+1] ,r;
double aa, bb,n0;
double res100[nn+1],eps, res103[nn+1],res104[nn+1],min,ll,x;
double res10[nn+1],res20[nnnn+1],res200[nnnn+1];
double left,raight,pi,res11[nn+1],res14[nn+1], res101[nn+1],res102[nn+1];
char str[nn+1]="Polotsk State University 0123456789";
int k0;

FILE*file;
eps=0.000000000001;
n0=30.0;
m=2;
pi=2.0*asin(1.0);
left =3.0/4.0+eps;
raight=1.0;
printf("left=%.16lf raight=%.16lf n0=%.16lf m=%d\n",left,raight,n0,m);
printf("\n");
for(i=0;i<=nn-1;i++)
{
printf("%c",str[i] );//,int(res105[i]));
}
printf("\n");
```

```
printf("\n");
for(ii=0;ii<=nn-1;ii++)
{
res[ii]=str[ii];
res10[ii]=double(res[ii])/double(255);
res11[ii]=left+(raight-left)*res10[ii];
res100[ii]=f1(res11[ii],left,raight,m,n0);
 res101[ii]= res100[ii]*1000000.0;
}
remove("1001.txt");
file=fopen("1001.txt","w");
for(i=0;i<=nn-1;i++)
{
aa=int(res101[i]);
bb=int ((res101[i]  -aa)*1e8);
fprintf(file,"%.4lf\n", aa);
fprintf(file,"%.4lf\n", bb);
}
fclose(file);
int sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1001.txt","r");
if(file==NULL)
{
printf(" not open file");
}
else
{
 for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
printf(" coder1(%d)=%.12lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
for(i=0;i<=nn-1;i++)
{
res103[i]=res102[i]/1000000.0;
res104[i]=f2(res103[i],left,raight,m,n0);
```

```
res14[i]=(res104[i]-left)/(raight-left);
}
printf("\n");
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
 printf("start(%d) = %d  decoder(%d) = %d\n",i,res[i], i,res105[i]);
}
printf("\n");
for(i=0;i<=nn-1;i++)
{
printf("%c",res105[i] );
}
printf("\n");
}
double f1(double x, double left, double raight, int m, double n)
{
double norm,z;
if (m==1)
{
norm = pow(asin(raight),double(n));
z= pow(asin(x),double(n))/norm;
return z;
}
else if(m==2)
{
norm = pow(atan(raight),double(n));
z= pow(atan(x),double(n))/norm;
return z;
}
}
double f2(double x, double left, double raight, int m, double n)
{
double norm,z;
if (m==1)
{
norm = pow(asin(raight),double(n));
z=sin(pow((x)*norm,1.0/double(n)));
}
else if(m==2)
{
norm = pow(atan(raight),double(n));
z= tan(pow((x)*norm,1.0/double(n)));
}
```

```
        return z;
}
```

2.Код основной программы 2 и библиотеки

```
#include<stdio.h>
#include<iostream>
#include<fstream>
#include<algorithm>
#include<stdlib.h>
#include<vector>
#include<iterator>
#include<math.h>
# define nnn 8000
using namespace std;
double f1(double x,double left, double raight ,double a);
double f2(double x,double left, double raight, double a);
int const nn=35,nnnn=13,n=3;
int main()
{
int m,m5,m6,mm,l,i,ii,j,jj,n,phi,r1,r2,a,m1,m2,alpha,aa1,aa2,shifr[nn+1];
int
r11,r12,r21,r22,a1,a2,m3,m4,beta,p,b,res[nn+1],res1[nn+1],podp1[nnn+2],podp[nn
n+2];
int  res105[nn+1] ,r;
double res100[nn+1], aa,bb,a0,eps,res103[nn+1],res104[nn+1],min,ll,x;
double res10[nn+1],res20[nnnn+1],res200[nnnn+1];
double left,raight,pi,res11[nn+1],res14[nn+1], res101[nn+1],res102[nn+1];
char str[nn+1]="Polotsk State University 1234567890";
int k0;

FILE*file;
//eps=0.0;
eps=0.0000000001;
pi=2.0*asin(1.0);
left =1.0+eps;
raight=10.0;
a0=5.0;
printf("\n");
printf("left=%.16lf raight=%.16lf a=%.16lf \n",left,raight,a0);
printf("\n");
for(i=0;i<=nn-1;i++)
{
```

```
 printf("%c",str[i] );
}
printf(" \n");

for(ii=0;ii<=nn-1;ii++)
{
res[ii]=str[ii];
res10[ii]=double(res[ii])/double(255);
res11[ii]=left+(raight-left)*res10[ii];
 res100[ii]=f1(res11[ii],left,raight,a0);
 res101[ii]= res100[ii]*1000.0;
}
remove("1011.txt");
file=fopen("1011.txt","w");
for(i=0;i<=nn-1;i++)
{
aa=int(res101[i]);
bb=int ((res101[i]-aa)*1e8);
fprintf(file,"%.4lf\n", aa);
fprintf(file,"%.4lf\n", bb);
}
fclose(file);
printf(" \n");
int sad[2*nn+1];
char arr00[2*nn+1];
file=fopen("1011.txt","r");
if(file==NULL)
{
printf(" not open file");
}
else
{
 for(i=0; i<=2*nn-1;i++)
{
fgets(arr00,nn-1,file);
sad[i]=atoi(arr00);
}
}
fclose(file);
for(i=0;i<=nn-1;i++)
{
res102[i] = sad[2*i]+ sad[2*i+1]*1e-8;
printf(" coder1(%d)=%.12lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
```

```
for(i=0;i<=nn-1;i++)
{
res103[i]=res102[i]/1000.0;
res104[i]=f2(res103[i],left,raight,a0);
res14[i]=(res104[i]-left)/(raight-left);
}
printf("\n");
for (i=0;i<=nn-1;i++)
{
res105[i]=1+int(255.0*res14[i]);
 printf("start(%d) = %d  decoder(%d) = %d\n",i,res[i], i,res105[i]);
printf("\n");
for(i=0;i<=nn-1;i++)
{
 printf("%c",res105[i] );
}
printf("\n");
}
double f1(double x, double left, double raight, double a0)
{
double norm,z;
z= log(x)/double(a0);
return z;
}
double f2(double x, double left, double raight, double a0)
{
double z;
z=exp(a0*x);
return z;
}
```

3.Код вспомогательной программы

```
#include<stdio.h>
#include<math.h>
int module(int m,int l, int n);
int zakr(int a,int n);
int const nn=24,nnn=13;
int main()
{
int m,m5,m6,mm,l,i,n,phi,r1,r2,a,m1,m2,alpha,aa1,aa2,shifr[nn+1];
int r11,r12,r21,r22,a1,a2,m3,m4,beta,p,b,res[nn+1],res1[nn+1];
//char str[nn+1]="Polotsk State University";
```

```
char str[nn+1]="1223456786433.3456776654";
printf("\n");
for(i=0;i<=nn-1;i++)
{
printf("%c",str[i]);
}
printf("\n");
 for(i=0;i<=nn-1;i++)
{
res[i]=str[i];
 printf("number(%d)=%d\n",i,res[i]);
}
p=61;
phi=p-1;
a=13;
b=7;
alpha=zakr(a,phi);
beta=zakr(b,phi);
printf("alpha=%d\n",alpha);
printf("beta=%d\n",beta);
for (i=0;i<=nn-1;i++)
{
m1=module(res[i],a,p);
m2=module(m1,b,p);
m3=module(m2,alpha,p);
m4=module(m3,beta,p);
res1[i]=m4;
printf("m=%d m1=%d  m2=%d m3=%d  m4=%d \n",res[i],m1,m2,m3,m4);
}

for(i=0;i<=nn-1;i++)
{
printf("%c %c\n",str[i],res1[i]);
}
}
int module(int m,int l,int n)
{
int bac,i;
bac=1;
for (i=1;i<=l;i++)
{
bac=bac*m;
if(bac<n)
{
```

```
bac = bac;
}
else if(bac>=n)
{
bac = bac - int(double(n)*(int(double(bac)/double(n))));
}
}
return bac;
}
int zakr(int a,int n)
{
int i,j,n1,n2,jj;
jj=0;
n1=10000;
n2=10000;
for(i=1;i<=n1;i++)
{
for(j=1;j<=n2;j++)
{
if(a*i-n*j==1)
{
jj=jj+1;
return i;
}
}
}
}
```