Министерство образования Республики Беларусь

Учреждение образования

«Полоцкий государственный университет»

Д.Ф. Пастухов, Ю.Ф. Пастухов, Смоляк А.И.

Шифрование гиперболическими функциями

Учебное пособие к лекционным и практическим занятиям

для студентов специальности

1-40 01 01 Программное обеспечение информационных технологий

1-98 01 01 Компьютерная безопасность

Новополоцк

ПГУ

2018

ББК 32.811

Рецензенты:
А.А. Козлов, кандидат физико-математических наук, доцент,
заведующий кафедрой высшей математики
Р.П. Богуш, кандидат технических наук, доцент,
заведующий кафедрой вычислительных систем и сетей

Полоцкого государственного университета

Пастухов Д.Ф., Пастухов Ю.Ф., Смоляк А.И.

Шифрование гиперболическими функциями: учебное пособие

/Д.Ф. Пастухов, Ю.Ф. Пастухов, - Новополоцк: ПГУ, 2018. -33 с.

Материал учебного пособия соответствует требованиям государственного образовательного стандарта по математике и криптографии. В пособии содержатся методы шифрования текстовых данных гиперболическими функциями.

Для студентов университетов, программистов (специальность – программное обеспечение информационных технологий), а также для студентов технических вузов.

Содержание

Предисловие

Предлагаемое учебное пособие создано при чтении авторами курсов лекций двух предметов Математические методы в криптографии и Протоколы аутентификации в течение ряда лет в Полоцком государственном университете студентам специальности 1-40 01 01 (Программное обеспечение информационных технологий).

Богатство действительных чисел с двойной точностью ещё не нашло широкого распространения в криптографии. Однако применение чисел и функций с двойной точностью обладает рядом неоспоримых преимуществ при их использовании в качестве ключей и шифров. Прежде всего - это большая плотность действительных чисел – ключей даже на единичном интервале действительной оси. Это и большая чувствительность шифра к малому изменению действительного параметра ключа. Одним из главных преимуществ является также существенно большая машинная бесконечность в области чисел с двойной точностью (10^{300}), что существенно больше, чем 10^{10} для целочисленной бесконечности. При вычислениях с двойной точностью машинная бесконечность фактически не достигается, но она легко достигается при расчётах в области целых чисел, и переполнение потолка целых чисел неминуемо приведёт к ошибкам, если переполнение не учесть в алгоритмах.

Использование гиперболических функций для шифрования связано с тем, что гиперболический синус и косинус совпадает с тригонометрическим синусом в окрестности нуля, но отличается от них при конечных аргументах, при больших аргументах гиперболический синус и косинус фактически неразличимы между собой и экспонентой, но при конечных аргументах невозможно подобрать экспоненциальную функцию точно аппроксимирующую гиперболический синус или косинус. Кроме того и шифры гиперболического синуса и гиперболического косинуса различаются между собой. Все указанные особенности приведут криптоаналитика к неудачной попытке объяснить шифры применением известных элементарных функций, т.е. тригонометрических и экспонент.

Также в данном учебном пособии применен протокол аутентификации без передачи ключа и используется только для передачи параметров ключа между абонентами перед сеансом связи, т.е. реализована идея протокола профессора Лидовского В.В.

Пастухов Д.Ф., Пастухов Ю.Ф.

1. Структурная схема протокола без передачи ключа

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 каждого из абонентов трёх или более символьных строк длиной не более 16 символов – это десятичная запись параметров $left, raight, a$. 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(2 значных числа). Абоненты для секретной переписки конфиденциально выбирают достаточно большое простое число p, такое что

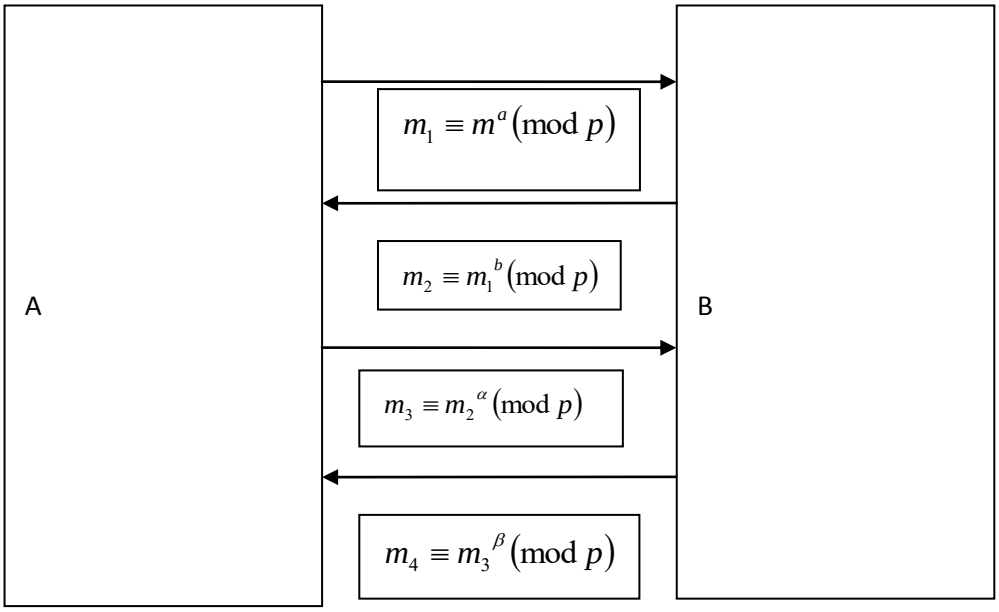
p-1 разлагается на не очень большие простые множители.

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

$m_1 \equiv m^a \pmod p$.В получив m_1 ,зашифровывает его своим ключом $m_2 \equiv m_1^{\ b} \pmod p$. А далее шифрует по формуле $m_3 \equiv m_2^{\ \alpha} \pmod p$ и отправляет 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 в протоколе без передачи долговременных ключей, α, β – закрытые ключи абонентов A,B, а a, b – их открытые ключи.

После создания сеансового ключа, абоненты могут передавать сообщения, используя методы шифрования гиперболическими функциями описанных в пункте 2.

2. Математическая постановка задачи шифрования – дешифрования с использованием гиперболических функций.

1)Гиперболический косинус. $y(x) = ch(x) = \frac{e^x + e^{-x}}{2}, x \geq 0$. Обозначим переменную $z = e^x$, преобразуем уравнение, используя переменную z

$1/z = e^{-x} \Rightarrow 2y = z + 1/z \Leftrightarrow z^2 + 1 - 2zy = 0$ - квадратное уравнение относительно переменной z. Решая последнее квадратное уравнение, получим корни

$z_{1,2} = y \pm \sqrt{y^2 - 1}, z_2 = y + \sqrt{y^2 - 1} = ch(x) + \sqrt{ch^2(x) - 1}$, тогда получим, что дешифрованный текст x зависит от шифра текста $y(x)$ по формуле

$$x = \ln(z) = \ln\left(y + \sqrt{y^2 - 1}\right) = \ln\left(ch(x) + \sqrt{ch^2(x) - 1}\right). \quad (1)$$

Для дешифрования выбран наибольший корень уравнения.

2)Гиперболический синус. $y(x) = sh(x) = \frac{e^x - e^{-x}}{2}, x \geq 0$. Обозначим переменную $z = e^x$, преобразуем уравнение, используя переменную z $1/z = e^{-x} \Rightarrow 2y = z - 1/z \Leftrightarrow z^2 - 1 - 2zy = 0$ - квадратное уравнение относительно переменной z. Решая последнее квадратное уравнение, получим корни

$z_{1,2} = y \pm \sqrt{y^2 + 1}, z_2 = y + \sqrt{y^2 + 1} = sh(x) + \sqrt{sh^2(x) + 1}$, тогда получим, что дешифрованный текст x зависит от шифра текста $y(x)$ по формуле

$$x = \ln(z) = \ln\left(y + \sqrt{y^2 + 1}\right) = \ln\left(sh(x) + \sqrt{sh^2(x) + 1}\right). \quad (2)$$

Для дешифрования выбран наибольший корень уравнения.

Отметим, что формулы(1) и (2) можно объединить в общую формулу (3)

$$x = \ln(z) = \ln\left(y + \sqrt{y^2 + l}\right), l = \begin{cases} 1, y(x) = sh(x) \\ -1, y(x) = ch(x) \end{cases} \Leftrightarrow y(x) = \frac{e^x - le^{-x}}{2}, l = -1,1 \quad (3)$$

Отметим, что в формуле(3) для шифрования-дешифрования гиперболическими функциями нами были использованы их правые ветви для взаимно однозначного шифрования – дешифрования.

3. Описание основной программы шифрования и дешифрования данных (с использованием гиперболических функций)

Параметрами ключей являются действительные числа a, b, c ,k- параметр функции шифрования-дешифрования. - левая b - правая граница интервала, c - коэффициент масштабирования (растяжения функции).

Используются следующие библиотеки

```
#include<stdio.h>

#include<math.h>

#include<iostream>

#include<fstream>

#include<algorithm>
```

```
using namespace std;
```

// Функция шифрования $f(x,k)$ содержит два аргумента. Первый аргумент x – действительное неотрицательное число, преобразованное шифруемым символом. Второй целочисленный аргумент принимает значение $k=1$ для гиперболического синуса и $k=2$ для гиперболического косинуса. m- число символов в сообщении.

```
double f(double x, int k){
if(k==1){return (exp(x)-exp(-x))/2.0;}
if(k==2){return (exp(x)+exp(-x))/2.0;}}
int const m=20;// количество символов в строке
main(){
int i,aa,bb,k,l, res7[m]; целочисленные переменные
double x, res1[m], res2[m] , res3[m] ,a1,b1,c1;
double  res4[m], res5[m] , res6[m] ;
double a2,b2,c2,a3,b3,c3,res101[m], res102[m]; действительные переменные
char str[m+1]=("Smolak Igor 2018 FIT"); // строка из 20 символов
k=2;l=1;//использование гиперболического косинуса для шифрования
for(i=1;i<=k+1;i++) {l=l*(-1);} // функция знака
a1=2.0;b1=3.0;c1=4.0;// параметры первого ключа
a2=7.0;b2=10.0;c2=5.0;//параметры второго ключа
a3=0.0;b3=3.0;c3=6.0;//параметры третьего ключа
for(i=0;i<=m-1;i++){
res1[i]=double(str[i])/255.0;
```

Каждый строковый символ сообщения str[i] переводится в действительное число и нормируется на единицу.

```
if(i%3==0){
```

res2[i]=a1+(b1-a1)*res1[i];// однозначное отображение нормированного символа в точку интервала $[a,b]$.

res3[i]=c1*f(res2[i],k); }// вычисление шифр - функции f(res2[i],k) и её однородное растяжение в c1 раз

else if(i%3==1){

res2[i]=a2+(b2-a2)*res1[i];

res3[i]=c2*f(res2[i],k); } // шифрование всех вторых символов сообщения вторым ключом

else if(i%3==2){

res2[i]=a3+(b3-a3)*res1[i];

res3[i]=c3*f(res2[i],k);}// шифрование всех третьих символов сообщения третьим ключом

printf("i=%d %c %d normal=%.16lf coder=%.16lf \n",i,str[i], str[i],res1[i],res3[i]);}

for(i=0;i<=m-1;i++) {

printf("%.16lf \n", res3[i]);

res101[i]= res3[i]; }

FILE*file;

 int sad0[2*m+1];

remove("1001.txt"); // очистка текстового файла 1001.txt

file=fopen("1001.txt","w");// открытие текстового файла1001.txt

for(i=0;i<=m-1;i++){

aa=int(res3[i]);// выделение целой части масштабированной шифр- функции

bb=int ((res3[i]-aa)*1e9);// перевод дробной части масштабированной шифр-функции в целое число с сохранением 9 первых знаков после запятой.

printf("coder1(%d)=%.12lf\n",i,res3[i]);

fprintf(file,"%d\n", aa);// запись целой части шифра в текстовый файл 1001 txt

sad0[2*i]=aa;

```
fprintf(file,"%d\n", bb); )// запись мантиссы   шифра в текстовый файл 1001 txt

sad0[2*i+1]=bb;}

fclose(file);//закрытие файла

int sad[2*m+1];

char arr00[2*m+1];

file=fopen("1001.txt","r");//чтение файла 1001.txt

if(file==NULL){

printf(" not open file");}

else{

for(i=0; i<=2*m-1;i++){

fgets(arr00,2*m-1,file);

sad[i]=atoi(arr00);// извлечение шифра – символа из 2*m строк файла 1001.txt

printf(" coder2(%d)=%d coder1(%d)=%d\n", i, sad[i],i, sad0[i]);}}

fclose(file);

for(i=0;i<=m-1;i++){

res102[i] = double(sad[2*i])+ double(sad[2*i+1])*1e-9;// возврат из целой
частиsad[2*i]) и целого числа sad[2*i+1]) для мантиссы шифра действительного значения
символа шифра. Значения массивов res102[i] и res3[i] совпадают.

printf(" coder1(%d)=%.8lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );}

for(i=0;i<=m-1;i++){

if(i%3==0){// обратное дешифрование каждого первого символа

res4[i]= res102[i]/c1;//обратное масштабирование

res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l));//возврат нормированного символа-
сообщения

res6[i]= 255.0*(res5[i]-a1)/(b1-a1);}// возврат дешифрованного символа

 else if(i%3==1){ //обратное дешифрование каждого второго символа
```

res4[i]= res102[i]/c2; // res4[i]= res3[i]/c2

res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l)) ;// res5[i]= res2[i]

res6[i]= 255.0*(res5[i]-a2)/(b2-a2);} // res6[i]= res1[i]* 255.0

else if(i%3==2){// обратное дешифрование каждого третьего символа

res4[i]= res102[i]/c3;

res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l));

res6[i]= 255.0*(res5[i]-a3)/(b3-a3);}

res7[i]= int(res6[i])+1;// перевод действительного значения символа в целое число с учётом отбрасывания мантиссы действительного числа компилятором при взятии операции int.

printf("i=%d text(%d)=%c decoder(%d)=%c \n",i,i,str[i],i,res7[i]);}}// печать дешифрованной символьной строки и её сравнение с исходным текстом.

Параметрами ключей являются числа a, b, c, k .

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, 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 res1[i].

2) Далее с единичным отрезком мы проводим линейное преобразование, растягивающее его в несколько раз и с применением параллельного переноса, смещающее левую точку по формуле

$$\text{res2[i]} = a + (b - a)res1[i]$$

Где a , b левая и правая границы нового отрезка. В результате последнего преобразования все точки преобразованного отрезка имеют координаты $x_i \in [a, b]$

3) Сначала работает прямая функция с

вызовом res3[i]=c*f(res2[i],k). В которой по порядку

следуют аргументы: число $x = \text{res2[i]}, \text{x} \in [\text{a}, \text{b}]$, c- коэффициент подобия,

k номер гиперболической функций целое k=1 для гиперболического синуса

и k=1 для гиперболического косинуса.

4) Поскольку диапазон целых чисел значительно меньше диапазона действительных чисел с плавающей запятой двойной точности, то массив res101[i] (res101[i]= res3[i]) заполняется действительными числами двойной точности, здесь применяется одна важная идея. В файл 1001.txt записываются сначала целые части действительных чисел шифра – все нечётные строки текстового файла sad[2*i+1]. Затем мантиссы шифра переводятся в целые числа (первые 9 знаков) и заполняются все чётные строки текстового файла sad[2*i]. Иначе при считывании чисел из файла 1001.txt может быть потеряна мантисса действительного числа. Это во много раз может снизить размерность пространства шифрующих ключей и надёжность дешифрования данных.

5)Открываем и копируем данные текстового файла1001.txt в массив sad[2*i], затем из каждой соседней пары целых чисел формируем целую часть и мантиссу шифра, создаём массив по формуле res102[i] = double(sad[2*i])+ double(sad[2*i+1])*1e-9

Далее столбцы кода до записи в текстовый файл и после чтения из него сравниваются res3[i]; и res102[i] (должны совпадать).

Прочитанные данные записываются в файл res102[i] .

6) Проводим обратное масштабирование res4[i]= res102[i]/c

7)вызываем обратную функцию дешифрования

res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+L))

$$x = \ln(z) = \ln\left(y + \sqrt{y^2 + L}\right), L = \begin{cases} 1, y(x) = sh(x) \\ -1, y(x) = ch(x) \end{cases} \Leftrightarrow y(x) = \frac{e^x - Le^{-x}}{2}, L = -1,1$$

9)проводим обратное линейное преобразование и возвращаемся к переменной x по формуле

res6[i]= 255.0*(res5[i]-a)/(b-a))

10) переводим действительный массив дешифрованного символа в целочисленный массив res7[i]= int(res6[i])

11) делаем обратное преобразование ASCII

for(i=0;i<=nn-1;i++)

{

printf("i=%d %.c \n",i,str[i],res1[i],res7[i]);}

12) в результате исходная символьная фраза

char str[m+1]=("Smolak Igor 2018 FIT") программой возвращается в эту же фразу.

Таким образом, в программе использованы математические операции запись чисел с двойной точностью в текстовой файл, композиция из преобразования подобия, нелинейных гиперболических функций, однородного сжатия функции

$$x_1 = n/255, x_2 = a + (b-a)x_1,\ \ y_1 = \frac{\exp(x_2) - l * \exp(-x_2)}{2}, y_2 = cy_1.$$

Обратные преобразования:

$$y_1 = y_2 / c,\ x_2 = \ln(y_1 + \sqrt{y_1^{\,2} + l}), x_1 = \frac{x_2 - a}{b-a},\ n = 255x_1, (l = 1 \leftrightarrow sh(x), l = -1 \leftrightarrow ch(x))$$

4. **Описание работы вспомогательной программы для создания общего сеансового ключа на базе протокола без передачи 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 $\alpha : \alpha a \equiv 1 (\mathrm{mod}\, \varphi(p)), \alpha = 37$:

```
#include <stdio.h>

#include <math.h>

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;}}}}
main(){
int a,n;
a=13;
n=60;
printf("zakr key(a=%d,p=%d)=%d \n",a,n+1,zakr(a,n));}
```

Параметр ключа – действительное число с 16 значащими цифрами и десятичной точкой, подлежащее шифрованию методом 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 D.

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 дештфрования.

Полное совпадение результатов при тестировании вспомогательной программы. В данном случае использованы параметры;

p=61; phi=p-1;a=13;b=7.

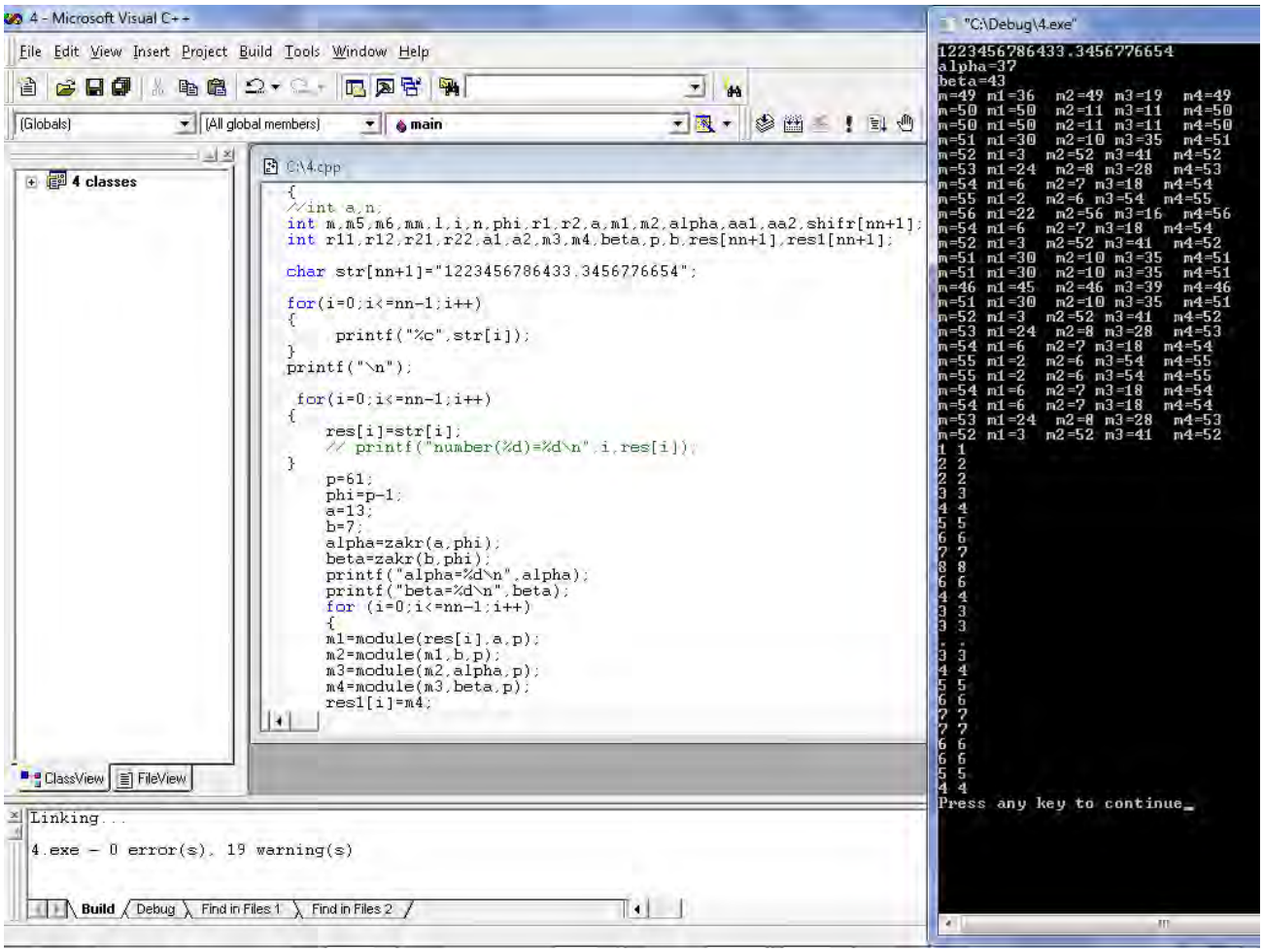


Рис.1 Иллюстрация работы протокола обмена данных без передачи ключей.

С указанными параметрами программа вычислила секретные ключи абонентов $\alpha = 37; \beta = 43$.

Подпрограмма для решения уравнения $y \equiv m^l \pmod n$ по известным целым числам m, n, l приведена ниже

```
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;}
```

5.Нелинейное шифрование гиперболическими функциями. Оценка пространства ключей. Тестирование программы.

Тестирование программы, размерность пространства ключей

1)Зададим в программе параметры a1=0.0;b1=6.0; c1=4.0; a2=1.0;b2=9.0; c2=5.0; a3=3.0;b3=4.0; c3=7.0;k=1;

На рисунке 2 видно, что каждый символ m_i сообщения таблицей ASCII преобразуется в число, которое затем преобразуется в действительное число и нормируется на единицу, последний столбец представляет код символа, причём смысл имеют только первые 9 знаков после запятой.

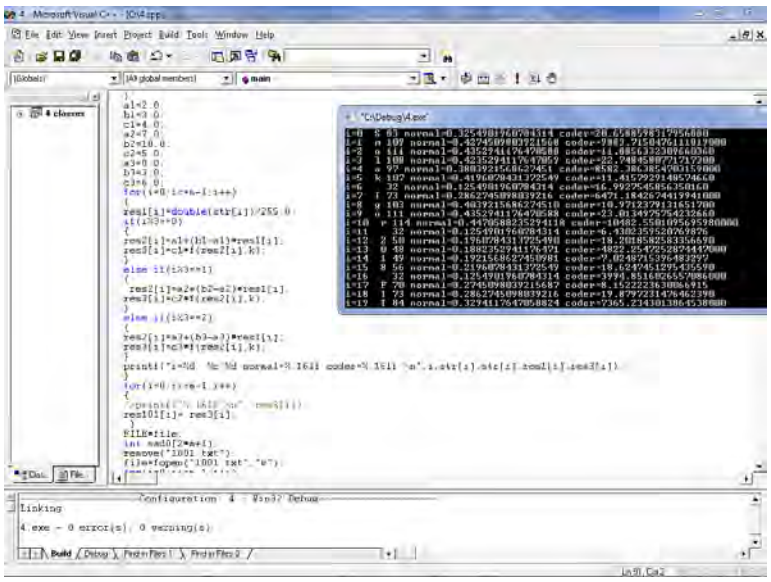


Рис.2

Из рисунка 3 видно, что коды символов исходного текста до записи в текстовый файл 1001.txt coder1, отдельная запись целой части кода символа и 9 первых знаков его мантиссы, а также после их чтения из файла 1001.txt (в coder2) полностью совпадают. Нечётные строки в1001.txt совпадают с целой частью coder1(до записи в текстовый файл) и coder2(после прочтения из текстового файла), а чётные строки в1001.txt совпадают с 9 знаками мантиссы coder1 и coder2. Из рисунка 3 также видно, что после прочитанный файл 1001.txt и образованный из него код совпадает в целой части и в мантиссе до записи в файл. Таким образом, при записи и чтении чисел из файла1001.txt не теряется точность чисел(double).

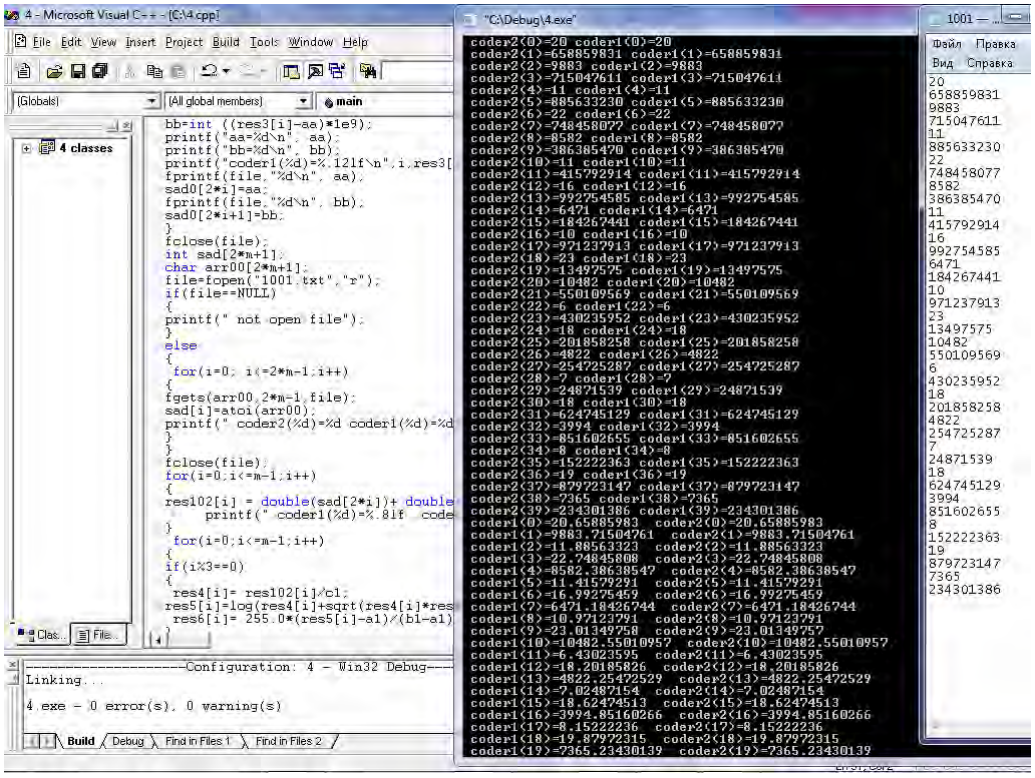


Рис.3

Из рисунка 4 видно, что символы строки char str[m+1]=("Smolak Igor 2018 FIT"), преобразованной универсальной таблицей ASCII (переменная text(i)) и дешифрованные символы (переменная decoder(i)) полностью совпадают. Что доказывает однозначность шифрования и дешифрования. Кроме того полностью совпадают фразы

("Smolak Igor 2018 FIT"), на рисунках 2 и 4, т.е. до шифрования и после шифрования и дешифрования.

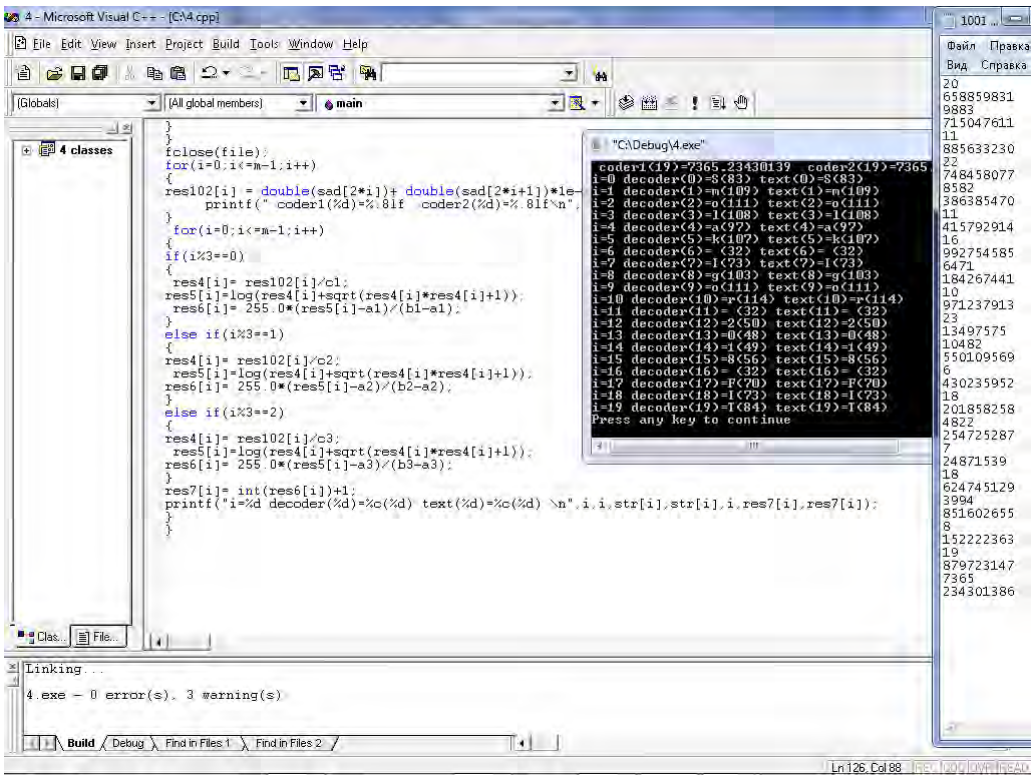


Рис.4

6. Обоснование информационной безопасности алгоритма шифрования – дешифрования с использованием гиперболических функций.

Оценим пространство ключей по всем параметрам ключа. Изменение значения любого параметра ключа или функции шифрования криптоаналитиком может привести к неверному дешифрованию символа. Применение сильно нелинейных функций, а также преобразование подобия делает крайне чувствительным результат шифрования (шифра) при малейшем изменении любого параметра функции шифрования. Очевидно, что поскольку в таблице ASCII всего 256 символов, получить большее разрешение, чем 1/256 по области изменения любого параметра шифрования мы не можем. Учтём, что ключей в алгоритме шифрования 3, каждый ключ имеет 3 независимых параметра. Кроме того, число используемых функций две – гиперболический синус и гиперболический косинус.

Тогда пространство ключей имеет мощность

$$N = 2*256^{3*3} \approx 9{,}44*10^{21} \text{шифров}$$

Пусть компьютер криптоаналитика перебирает параметры шифра и сравнивает полученный результат с анализируемым шифром на тождественность со скоростью $n = 10^6$ шифр/с.

Тогда время перебора составит

$$t = \frac{N}{n} \approx 9{,}44 * 10^{21-6} = 9{,}44 * 10^{15} c \approx 299490519 \text{ лет}$$

Что сравнимо(7%) с возрастом Земли $T_E = 4{,}5 * 10^9$ лет

Добавление четвёртого ключа образует пространство с размерностью и временем перебора ключей значительно превосходящих возраст Земли $t = 5 * 10^{15}$ лет, что превышает возраст Земли в миллион раз!

На рисунке 5 использованы все равные 9 параметров ключей a1,a2,a3,b1,b2,b3,c1,c2,c3, но разные функции гиперболический косинус(k=2) и гиперболический синус(k=1).Как видно из Рис.5, различаются целые и дробные значения шифра, т.е. различаются те части, подлежащие записи в текстовый файл. Показано, что данные функции дают действительно не совпадающие шифры, но при этом одна и та же исходная фраза char str[m+1]=("Smolak Igor 2018 FIT") совпадает с декодированной фразой одними ключами, но разными функциями.

Предположим, что криптоаналитик рассекретил параметры двух из трёх ключей, тогда при дешифровании известной фразы ("Smolak Igor 2018 FIT") он получит результат на Рис.6 (искусственно при дешифровании был изменен параметр b2 на 0. 1). Т.е. получена фраза в правом столбце, которую даже невозможно отобразить клавишами клавиатуры, и тем более неосмысленную.

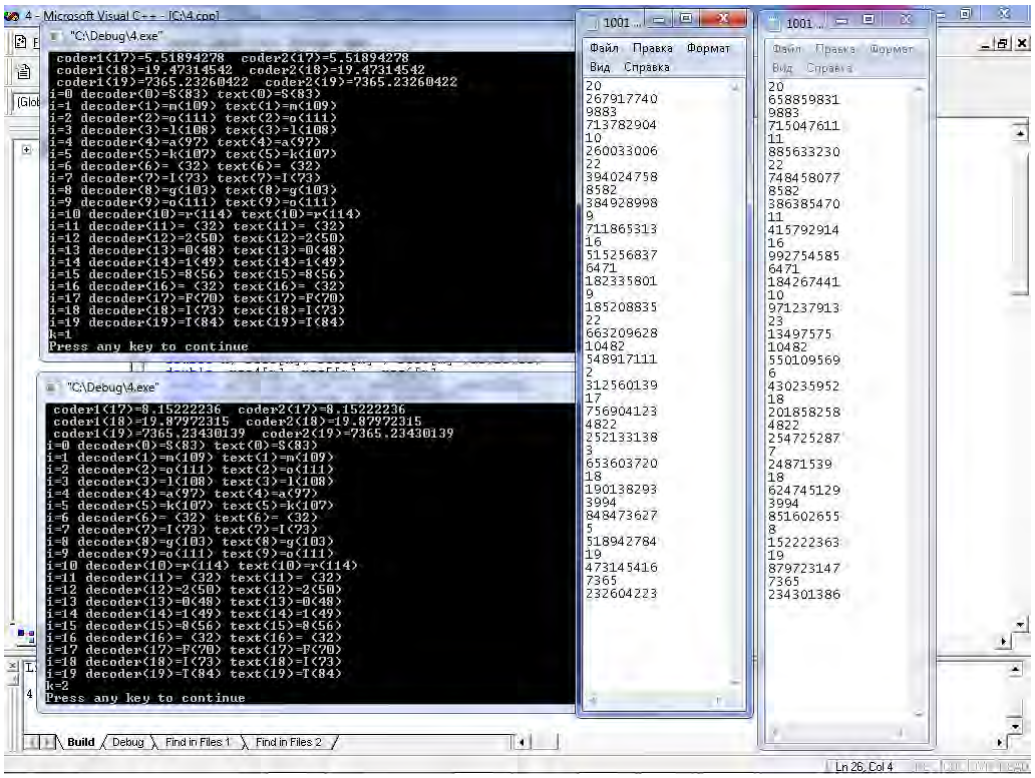


Рис.5

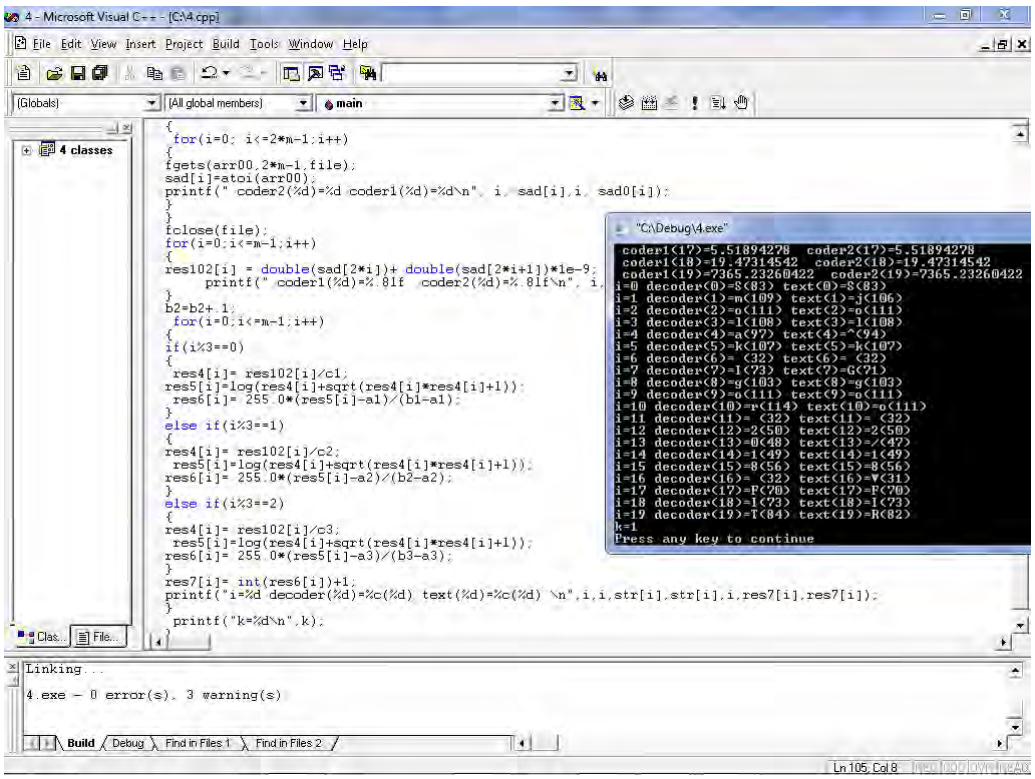


Рис.6

7. Интерфейс программы

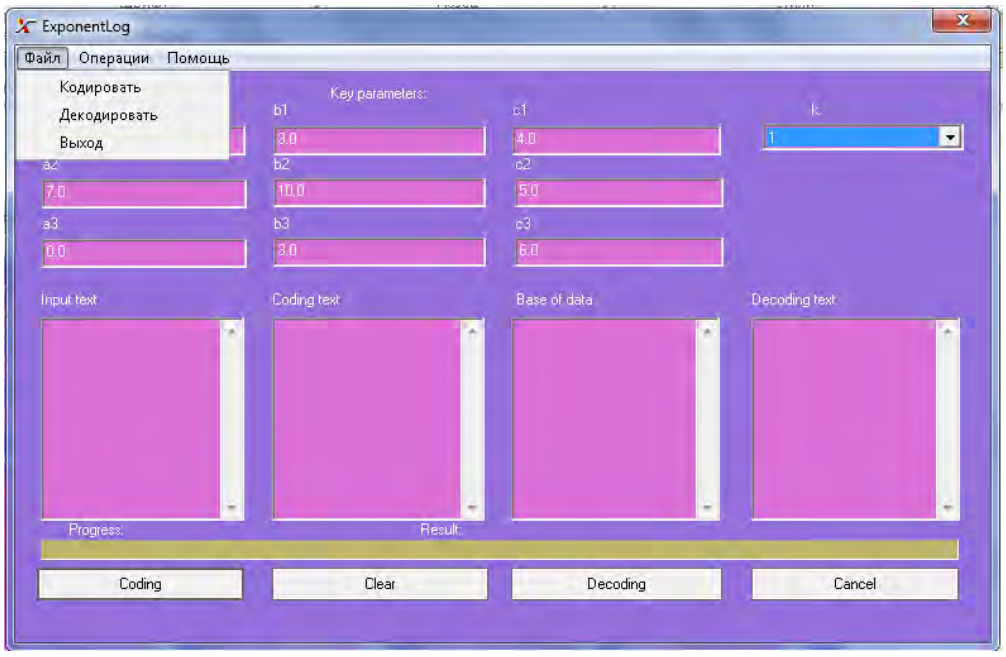


Рис.7

Интерфейс имеет стоку меню с падающим списком: 1)файл с выбором строки (кодировать, декодировать, выход); 2)операции с выбором строки (очистить); 3)помощь (о программе) (Рис.7).

Верхнее поле интерфейса содержит 9 параметров 3 ключей a1,b1,c1,a2,b2,c2,a3,b3,c3. Вводимые параметры ключа представляют собой действительные числа с двойной точностью. Параметр кодирующей функции принимает два значения для гиперболического синуса k=1, а для гиперболического косинуса k=2. Функциональные клавиши кодировать(coding), очистить(clear), декодировать(decoding), завершение работы(cancel).

Исходный текст сообщения использует все малые и большие символы латинской клавиатуры, а также все символы русской клавиатуры, числа, знаки табуляции и т.д. всего 256 символов в активное окно input text. Нажать клавишу кодировать. В пассивное окно coding text отображается массив кода res3[j] в виде действительных чисел с двойной точностью. В активном окне Base of data передаётся целочисленный код из массива sad[j], построчно выписывается целые части кода каждого символа во все нечётные строки, затем в чётные строки записываются все мантиссы, переведённые в целые числа с 9 знаками. Пассивное окно Decoding text остаётся незаполненным (Рис.8). Для декодирования необходимо нажать клавишу декодировать(decoding), после нажатия(decoding) декодированный текст отображается в крайнем правом окне decoding text(Рис.9). В нижней части интерфейса присутствует линейка относительного выполнения операции кодирования (декодирования).

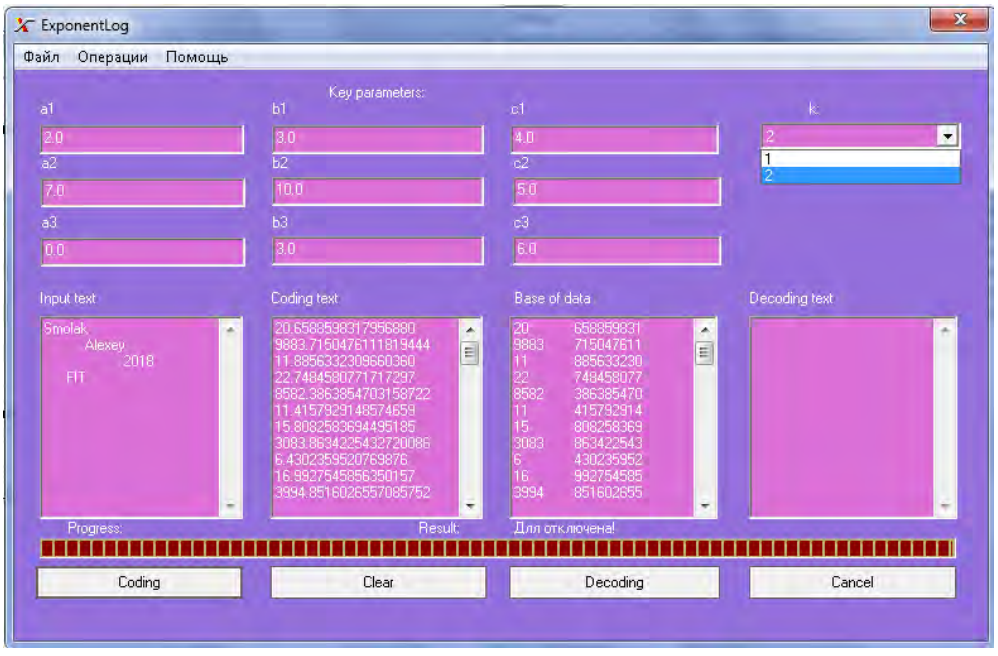


Рис.8

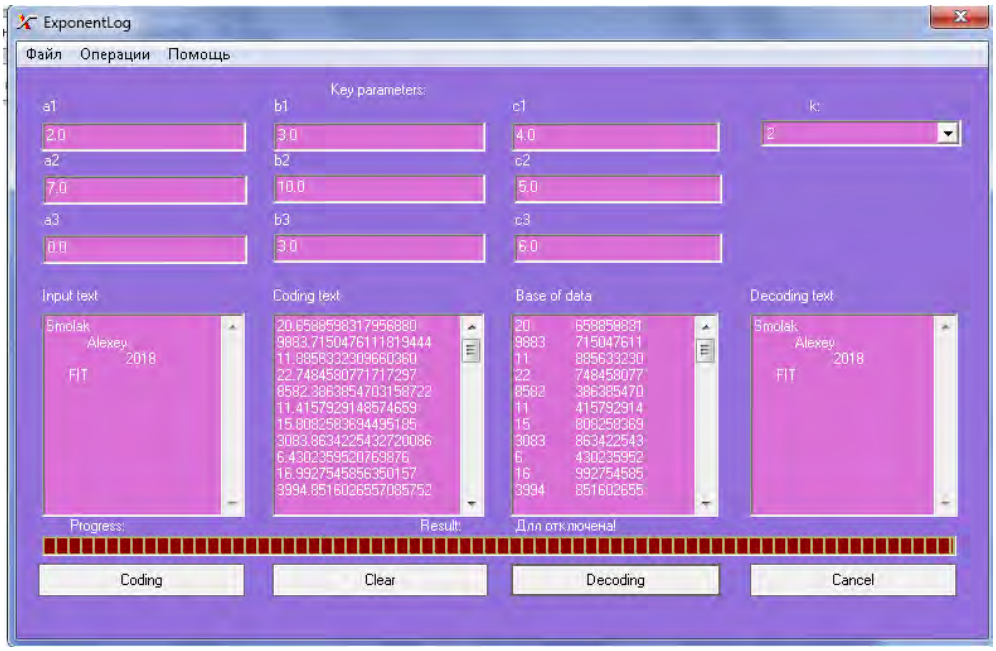


Рис.9

Интерфейс имеет дополнительно автономный режим декодирования. Для автономного декодирования предварительно в окно Base of data нужно ввести код сообщения, нажать клавишу декодировать для выхода декодированного текста в окно decoding text(Рис.10). Однако если ввести тот же код сообщения что и на Рис.11 ,но если изменить параметр одного из 9 ключей всего лишь на 0.1(значение c3),то получим полностью бессмысленное сообщение как видно из Рис.11.

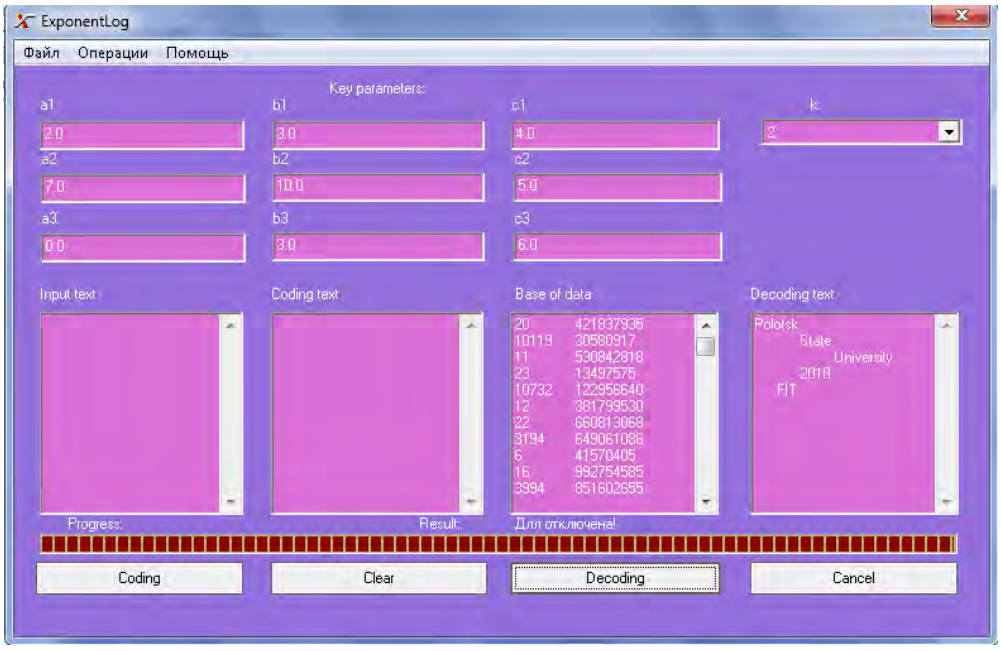


Рис.10

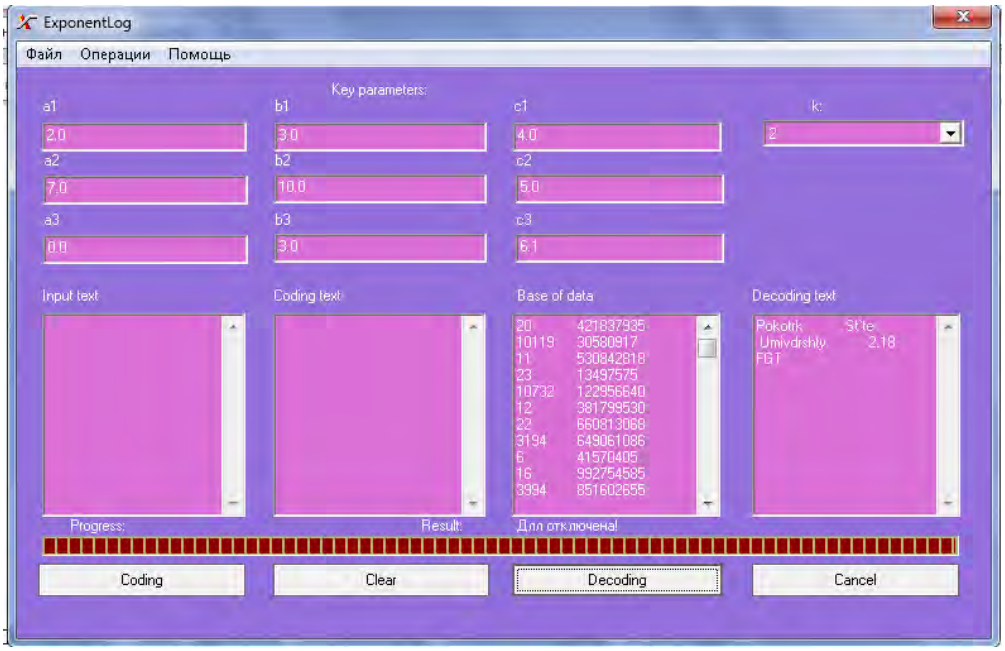


Рис.11

Учитывая, что время перебора параметров ключей сопоставимо с возрастом планеты Земля, то попытки злоумышленника, имея перехваченный код и программу интерфейса, но без знания ключей кода обречены на полную неудачу.

8.Код основной программы

```
#include<stdio.h>
#include<math.h>
#include<iostream>
#include<fstream>
#include<algorithm>
using namespace std;
double f(double x,int k)
{
if(k==1)
{
return (exp(x)-exp(-x))/2.0;
}
if(k==2)
{
return (exp(x)+exp(-x))/2.0;
}
}
int const m=20;
main()
{
int i,aa,bb,k,l, res7[m];
double x, res1[m], res2[m] , res3[m] ,a1,b1,c1;
double  res4[m], res5[m] , res6[m];
double a2,b2,c2,a3,b3,c3,res101[m], res102[m];
char str[m+1]=("Smolak Igor 2018 FIT");
k=2;
```

```
l=1;
for(i=1;i<=k+1;i++)
{
l=l*(-1);
}
a1=2.0;
b1=3.0;
c1=4.0;
a2=7.0;
b2=10.0;
c2=5.0;
a3=0.0;
b3=3.0;
c3=6.0;
for(i=0;i<=m-1;i++)
{
res1[i]=double(str[i])/255.0;
if(i%3==0)
{
res2[i]=a1+(b1-a1)*res1[i];
res3[i]=c1*f(res2[i],k);
}
else if(i%3==1)
{
 res2[i]=a2+(b2-a2)*res1[i];
res3[i]=c2*f(res2[i],k);
}
else if(i%3==2)
```

```
{
res2[i]=a3+(b3-a3)*res1[i];
res3[i]=c3*f(res2[i],k);
}
printf("i=%d  %d  normal=%c coder= %.16lf %.16lf\n",i,str[i],str[i],res1[i],res3[i]);
}
for(i=0;i<=m-1;i++)
{
printf("%.16lf \n", res3[i]);
res101[i]= res3[i];
 }
FILE*file;
int sad0[2*m+1];
remove("1001.txt");
file=fopen("1001.txt","w");
for(i=0;i<=m-1;i++)
{
aa=int(res3[i]);
bb=int ((res3[i]-aa)*1e9);
printf("aa=%d\n", aa);
printf("bb=%d\n", bb);
printf("coder1(%d)=%.12lf\n",i,res3[i]);
fprintf(file,"%d\n", aa);
sad0[2*i]=aa;
fprintf(file,"%d\n", bb);
sad0[2*i+1]=bb;
}
fclose(file);
```

```
int sad[2*m+1];
char arr00[2*m+1];
file=fopen("1001.txt","r");
if(file==NULL)
{
printf(" not open file");
}
else
{
 for(i=0; i<=2*m-1;i++)
{
fgets(arr00,2*m-1,file);
sad[i]=atoi(arr00);
printf(" coder2(%d)=%d coder1(%d)=%d\n", i, sad[i],i, sad0[i]);
}
}
fclose(file);
for(i=0;i<=m-1;i++)
{
res102[i] = double(sad[2*i])+ double(sad[2*i+1])*1e-9;
printf(" coder1(%d)=%.8lf  coder2(%d)=%.8lf\n", i, res101[i], i, res102[i] );
}
 for(i=0;i<=m-1;i++)
{
if(i%3==0)
{
 res4[i]= res102[i]/c1;
res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l));
```

```
 res6[i]= 255.0*(res5[i]-a1)/(b1-a1);
}
else if(i%3==1)
{
res4[i]= res102[i]/c2;
 res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l));
res6[i]= 255.0*(res5[i]-a2)/(b2-a2);
}
else if(i%3==2)
{
res4[i]= res102[i]/c3;
 res5[i]=log(res4[i]+sqrt(res4[i]*res4[i]+l));
res6[i]= 255.0*(res5[i]-a3)/(b3-a3);
}
res7[i]= int(res6[i])+1;
printf("i=%d decoder(%d)=%c text(%d)=%c \n",i,i,str[i],i,res7[i]);
}
}
```

9.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;
}
}
}
}
```

Шифрование гиперболическими функциями

Пастухов Д.Ф. кандидат физико-математических наук

Пастухов Ю.Ф. кандидат физико-математических наук

Смоляк А.И.

Новополоцк

ПГУ

2018