УДК: 004.9

ШИФРОВАНИЕ ПРИ ПОМОЩИ ПРОТОКОЛА ДИФФИ-ХЕЛЛМАНА

Тужилин Р.С.

Федеральное государственное бюджетное образовательное учреждение высшего образования «Брянский государственный университет имени академика И. Г. Петровского», г. Брянск, Россия, e-mail: roma.gradoff@yandex.ru

Рассматривается ведущая роль криптографии в защите информации в современном мире и в интернет пространстве в частности; рассмотрен один из существующих и активно применяемых протоколов шифрования, алгоритм которого позволяет создавать защищённые каналы связи между двумя и более пользователями. Выделены основные проблемы защиты информации, передаваемой по средствам сети интернет. Разработан программный продукт, наглядно демонстрирующий работу рассматриваемого протокола.
Ключевые слова: шифрование, криптография, канал связи, секретный ключ, криптоанализ, аутентификация, шифр, атака, шифротекст.

ENCRYPTION WITH THE DIFFY-HELLMAN PROTOCOL

Tuzhilin R.S.

Federal State-Funded Educational Institution of Higher Education "Bryansk State University named after Academician I.G. Petrovsky", Bryansk, Russia, e-mail: roma.gradoff@yandex.ru

The leading role of cryptography in the protection of information in the modern world and in the Internet space in particular is considered; considered one of the existing and actively used encryption protocols, the algorithm of which allows you to create secure communication channels between two or more users. The main problems of protecting information transmitted over the Internet are highlighted. A software product has been developed that clearly demonstrates the operation of the protocol in question.
Keywords: encryption, cryptography, communication channel, secret key, cryptanalysis, authentication, cipher, attack, ciphertext.

ВВЕДЕНИЕ

Вопрос защиты информации, с появлением различных средств для связи, становится всё более актуальным. В современном мире, данные, передаваемые при помощи Интернета, могут быть как личного характера, так и иметь стратегическое значение. Личная переписка человека, попав к третьим лицам, может нанести вред частной жизни и привести к негативным последствиям для самой личности. Стратегической информацией являются: статистические сведения, разведывательные данные, копии секретных документов, информация о количестве боевых единиц техники, схемы и карты с указанием расположения военных объектов, команды для управления различными объектами инфраструктуры

(атомные электростанции, дамбы и т.д.), засекреченные списки военнослужащих, с указанием их личных данных и т.д. Попадание подобных сведений к потенциальному врагу подставит под удар государство, его суверенитет и ухудшит оборонную силу страны.

Для связи сразличными объектами инфраструктуры и военными частями, как правило, используются специальные каналы связи, по которым данные передаются в зашифрованном виде при помощи различных аппаратных и программных комплексов.

Изучением и созданием различных алгоритмов, а так же основанных на них аппаратных и программных комплексов для шифрования занимается наука криптография. Криптография – наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним), целостности данных (невозможности незаметного изменения информации), аутентификации (проверки подлинности авторства или иных свойств объекта), а также невозможности отказа от авторства. Являясь одной из самых старейших наук, криптография существует более нескольких тысяч лет. Изначально криптография изучала методы шифрования информации - обратимого преобразования открытого (исходного) текста на основе секретного алгоритма или ключа в шифрованный текст (шифротекст). Традиционная криптография образует раздел симметричных криптосистем, в которых зашифровывание и расшифровывание проводится с использованием одного и того же секретного ключа. Помимо этого раздела современная криптография включает в себя асимметричные криптосистемы, системы электронной цифровой подписи (ЭЦП), хеш-функции, управление ключами, получение скрытой информации, квантовую криптографию.

С развитием технологий для передачи данных, появилось множество способов передачи и получения информации по средствам проводных и беспроводных сетей, спутниковой и иных видов связи, будь то мобильные сети различных поколений, оптоволоконное соединение, передача информации при помощи оптических средств (лазер). В основе каждого из методов лежит принцип передачи при помощи виртуально или физически устанавливаемого между клиентами канала связи. Для защиты передаваемых сведений используется шифрование, а так же различные методы аутентификации, например двухфакторная или биометрическая.

На данный момент существует большое количество различных алгоритмов шифрования. И задача каждого из них – предотвратить или сделать бессмысленным перехват информации, ввиду сложности и длительности процесса её дешифровки.

Актуальность данной научной работы заключается в том, что в настоящее время повсеместно используются различные средства, комплексы и алгоритмы для шифрования и защиты передаваемой информации.

Объект исследования – это криптографический протокол по схеме Диффи-Хеллмана.

Предмет исследования – это программный продукт, реализующий и демонстрирующий работу криптографического протокола Диффи-Хеллмана.

Цель исследования – изучение криптографического протокола по схеме Диффи-Хеллмана и создание программного продукта, реализующего и демонстрирующего работу криптографического протокола Диффи-Хеллмана.

Для достижения этой цели были поставлены следующие задачи:

- изучить теоретические сведения, необходимые для реализации поставленной задачи;
- ознакомиться с криптографическим протоколом Диффи-Хеллмана;
- создать программный продукт, наглядно демонстрирующий работу криптографического протокола Диффи-Хеллмана.
- выполнить отладку программного продукта и устранить выявленные ошибки.

ОСНОВНАЯ ЧАСТЬ

Криптографический протокол - это абстрактный или конкретный протокол, включающий набор криптографических алгоритмов. В основе протокола лежит набор правил, регламентирующих использование криптографических преобразований и алгоритмов в информационных процессах.

Функции криптографических протоколов:

1. аутентификация источника данных;
2. аутентификация сторон;
3. конфиденциальность данных;
4. невозможность отказа;
5. невозможность отказа с доказательством получения;
6. невозможность отказа с доказательством источника;
7. целостность данных;
8. обеспечение целостности соединения без восстановления;
9. обеспечение целостности соединения с восстановлением;
10. разграничение доступа.

Протоколы шифрования и дешифрования - это класс протоколов, в которых содержится некоторый симметричный или асимметричный алгоритм шифрования и дешифрования. Алгоритм шифрования выполняется на передаче отправителем сообщения, в

результате чего сообщение преобразуется из открытой формы в шифрованную. Алгоритм дешифрования выполняется на приёме получателем, в результате чего сообщение преобразуется из шифрованной формы в открытую. Так обеспечивается свойство конфиденциальности. Для обеспечения свойства целостности передаваемых сообщений симметричные алгоритмы шифрования и дешифрования, обычно, совмещаются с алгоритмами вычисления имитозащитной вставки (ИЗВ) на передаче и проверки ИЗВ на приёме, для чего используется ключ шифрования. При использовании асимметричных алгоритмов шифрования и дешифрования свойство целостности обеспечивается отдельно путем вычисления электронной цифровой подписи (ЭЦП) на передаче и проверки ЭЦП на приёме, чем обеспечиваются также свойства безотказности и аутентичности принятого сообщения.

Одним из способов создания защищённого канала связи является шифрование данных, с последующей дешифровкой при помощи секретного ключа. Данный подход к защите передаваемой информации является повсеместно встречающимся в современном мире. Примером реализации такого подхода является протокол Диффи-Хеллмана. При помощи специального алгоритма, он позволяет получить каждому из пользователей канала связи секретный ключ для шифровки и дешифровки сообщений, при этом сам ключ по каналу связи передаваться не будет, что исключает возможность его перехвата при атаке на канал связи.

Протокол обеспечивает хорошую стойкость от пассивных атак, ввиду особенностей самого алгоритма получения секретного ключа. Процесс вычисления секретного ключа для шифрования данных, передаваемых между двумя пользователями, происходит поэтапно и имеет следующий вид:

- задаются публично доступные переменные P и G с известными значениями;
- каждый из пользователей задаёт своё число A и B соответственно;
- при помощи числа G и чисел A или B заданных пользователями, происходят следующие вычисления: Пользователь-1 выполняет операцию возведения в степень A числа G и операцию поиска остатка от деления по переменной P (G^AmodP), аналогично делает и Пользователь-2: G^BmodP;
- далее происходит обмен данными, полученными в ходе вычисления: Пользователь-1 получает результат Пользователя-2 (RezB), а Пользователь-2 - результат Пользователя-1(RezA);
- после обмена результатами предыдущего этапа, выполняются следующие вычисления: Пользователь-1 производит возведение RezB в степень A и поиск остатка от деления по

переменной P ($RezB^{A}modP$), пользователь B производит схожую операцию, но с переменной RezA ($RezA^{B}modP$).

После выполнения алгоритма, каждый из пользователей получает секретный ключ в виде числа, являющимся результатом вычислений на последнем этапе. Секретный ключ получается идентичным для каждого пользователя. Это позволяет шифровать и дешифровать сообщения внутри канала связи, не совершая обмена секретными ключами. Из этого следует, что при взломе канала связи, перехваченные сообщения не смогут быть расшифрованы, так как ключ для дешифрации не передаётся между пользователями.

Для эффективного применения данного протокола к группе пользователей, необходимо соблюдение несколько основных принципов:

- передача ключа должна начинаться с «пустого» ключа G. Особенность состоит в повышении текущего значения показателя каждого участника один раз;
- любое промежуточное значение может быть раскрыто публично, но окончательное значение представляет собой секретный ключ, который никогда не должен быть публично раскрыт. Таким образом, каждый пользователь получает свою копию тайного ключа.

Данный протокол обладает недостатком - при наличии в канале связи третьего лица, которое перехватывает начальные передаваемые данные при их первичной передаче, третье лицо может расшифровать трафик, передаваемый по подобному каналу связи. По этой причине данный протокол, как правило, используется совместнос другими методами защиты и обеспечения конфиденциальности: двухфакторной аутентификацией, биометрической аутентификацией и т.д. Практическое применение данного протокола в совокупности с вышеуказанными методами аутентификации для доступа к каналу связи доказало свою эффективность, в связи с чем он используется уже более 50 лет для передачи различной ценной информации.

Для наглядной демонстрации работы данного протокола был создан программный продукт. Основные функции данного программного продукта реализованы на языке программирования высокого уровня C# на основе алгоритма протокола Диффи-Хеллмана.

Особенность криптографического протокола такова, что входные числа могут быть любыми, соответственно их длина может варьироваться от однозначного числа, до пятизначного числа и более. При работе с такими числами при операции возведения в степень даже двухзначного числа в степень трёхзначного, происходит выход за диапазон допустимой длины числа для типов данных, определенных в языке программирования C#

(Int32, Int64, double, long double и т.д.). По этой причине была задействована библиотека System.Numerics, содержащая объект BigInteger, который можно использовать для задания переменных неограниченной или неопределенной длины. С заданными таким способом переменными, можно производить те же арифметические операции, что и с любыми другими целочисленными переменными.

РЕЗУЛЬТАТЫ

В приложении реализованы несколько функций, а именно: задание переменных неограниченной длины, возведение в степень, поиск остатка от деления, случайное задание значений для ускорения ввода исходных данных, защита вводных полей от ввода неподходящих значений (букв, знаков), защита от ввода числа, начинающегося с нуля, цветовая индикация процесса выполнения расчётов. Исходный код программного продукта и его функций (см. Приложение 2) включён в работу.

Тестирование программного продукта происходило в процессе использовании основных функций приложения. После запуска программы, на экране пользователя появляется главное окно программы, которое можно увидеть на Рисунке 1.1 (см. Приложение 1).

Далее пользователю необходимо ввести начальные значения для четырех переменных (воспользуемся для этого кнопками для генерации случайных значений). Так же обратим внимание на появившийся в нижней части окна программы индикатор выполнения процесса вычислений, показанный на Рисунке 1.2 (см. Приложение 1).

Пользователь может не заполнить одно из полей или начать заполнение одного из полей числом, начинающимся на нуль. Каждый из этих вариантов является ошибкой, о чём будет выдано соответствующие уведомление при пустом поле, что можно увидеть на Рисунке 1.3 (см. Приложение 1) и при начале числа с цифры нуль, что демонстрируется на Рисунке 1.4 (см. Приложение 1).

При правильном вводе всех входных данных, при нажатии на кнопку «Вычислить публичные значения клиентов A и B» будет выведен промежуточный результат вычислений и записан в поля «Публичное значение kA» и «Публичное значение kB» соответственно. Так же изменится цвет индикатора хода вычислений, что показано на Рисунке 1.5 (см. Приложение 1).

При успешном выполнении предыдущего этапа вычислений, кнопка «Вычислить симметричные ключи для клиентов A и B» активируется, что показано на Рисунке 1.6 (см.

Приложение 1), при нажатии на неё происходит конечное вычисление, цвет индикатора меняет цвет, «мигающий» индикатор активируется.

ЗАКЛЮЧНИЕ И ВЫВОДЫ

После выполнения алгоритма, каждый из пользователей получает секретный ключ в виде числа, являющимся результатом вычислений на последнем этапе. Секретный ключ получается идентичным для каждого пользователя. Это позволяет шифровать и дешифровать сообщения внутри канала связи, не совершая обмена секретными ключами. Из этого следует, что при взломе канала связи, перехваченные сообщения не смогут быть расшифрованы, так как ключ для дешифрации не передаётся между пользователями.

Данный протокол обладает недостатком - при наличии в канале связи третьего лица, которое перехватывает начальные передаваемые данные при их первичной передаче, третье лицо может расшифровать трафик, передаваемый по подобному каналу связи. По этой причине данный протокол, как правило, используется совместно с другими методами защиты и обеспечения конфиденциальности: двухфакторной аутентификацией, биометрической аутентификацией и т.д. Практическое применение данного протокола в совокупности с вышеуказанными методами аутентификации для доступа к каналу связи доказало свою эффективность, в связи с чем он используется уже более 50 лет для передачи различной ценной информации.

Из вышеуказанного можно сделать вывод, что данный протокол является высокоэффективным средством, для шифрования и дешифрования сходящего и входящего трафика, с минимизацией рисков, заключающихся в перехвате данных или же дешифрации информации, зашифрованной таким способом.

Список литературы

1. Бабаш, А. В. История криптографии. Часть I / А.В. Бабаш, Г.П. Шанкин. - М.: Гелиос АРВ, 2002. - 240 с
2. Википедия. Криптография [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F
3. Википедия. Протокол Диффи-Хеллмана [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D0%B8%D1%84%D1%84%D0%B8_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0
4. Мао, В. Современная криптография: теория и практика.: Пер. с английского. М.: Издательский дом «Вильямс», 2005. — 768 с.
5. Масленников, М. Практическая криптография: моногр. / М. Масленников. - М.: БХВ-Петербург, 2003. - 464 с.
6. Петров, С.В. Информационная безопасность: Учебное пособие / С.В. Петров, И.П. Слинькова, В.В. Гафнер. — М.: АРТА, 2016. — 296 с.
7. Семененко, В.А. Информационная безопасность: Учебное пособие / В.А. Семененко. — М.: МГИУ, 2017. — 277 с.
8. Чипига, А.Ф. Информационная безопасность автоматизированных систем / А.Ф. Чипига. — М.: Гелиос АРВ, 2017. — 336 с.
9. Шаньгин, В.Ф. Информационная безопасность и защита информации / В.Ф. Шаньгин. — М.: ДМК, 2017. — 702 с.

Приложение 1

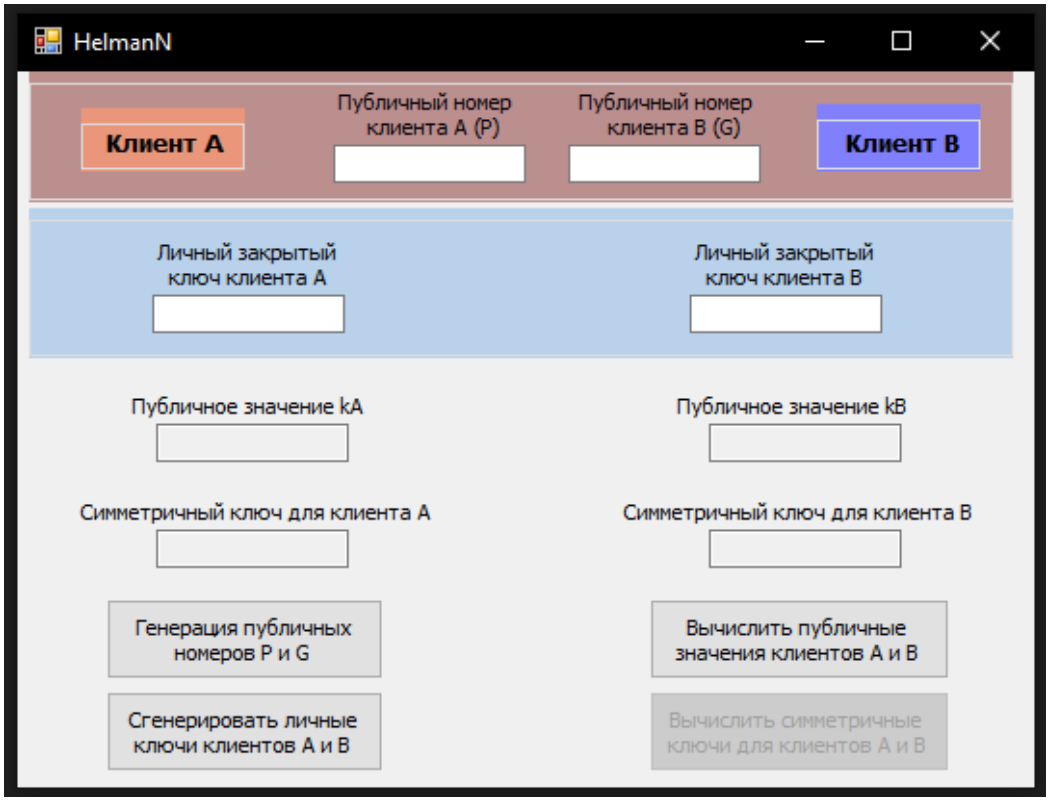

Рисунок 1.1 - Главное окно программы

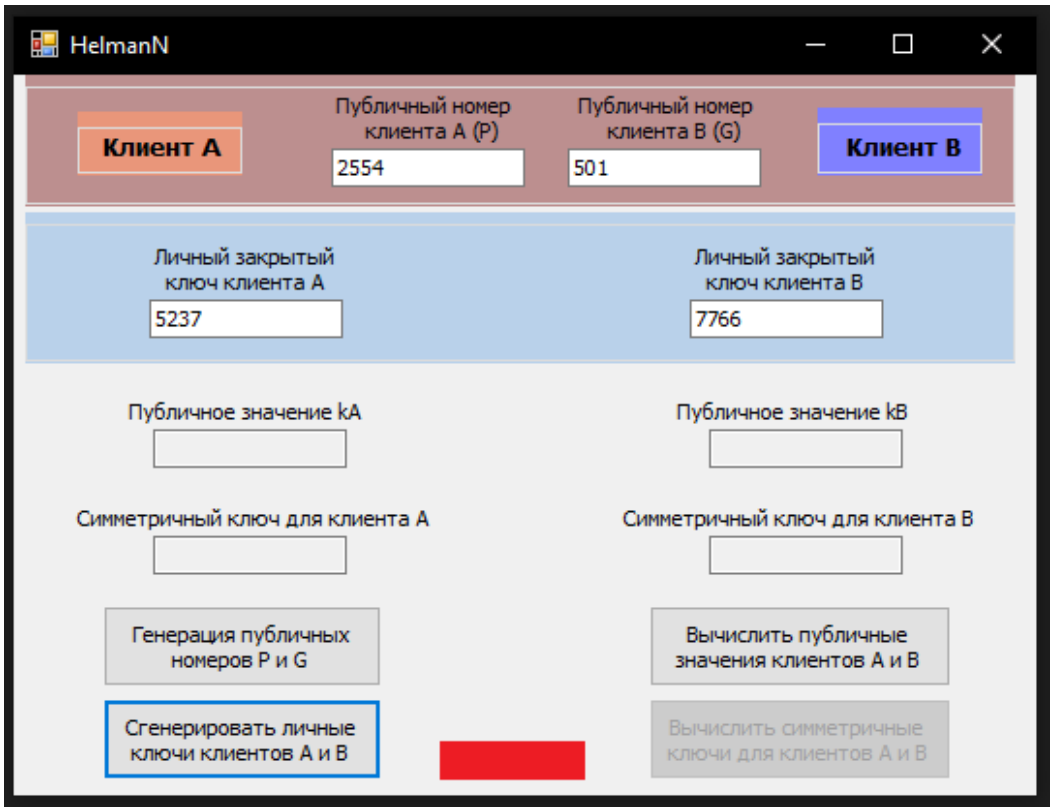

Рисунок 1.2 - Индикация прогресса выполнения вычислений

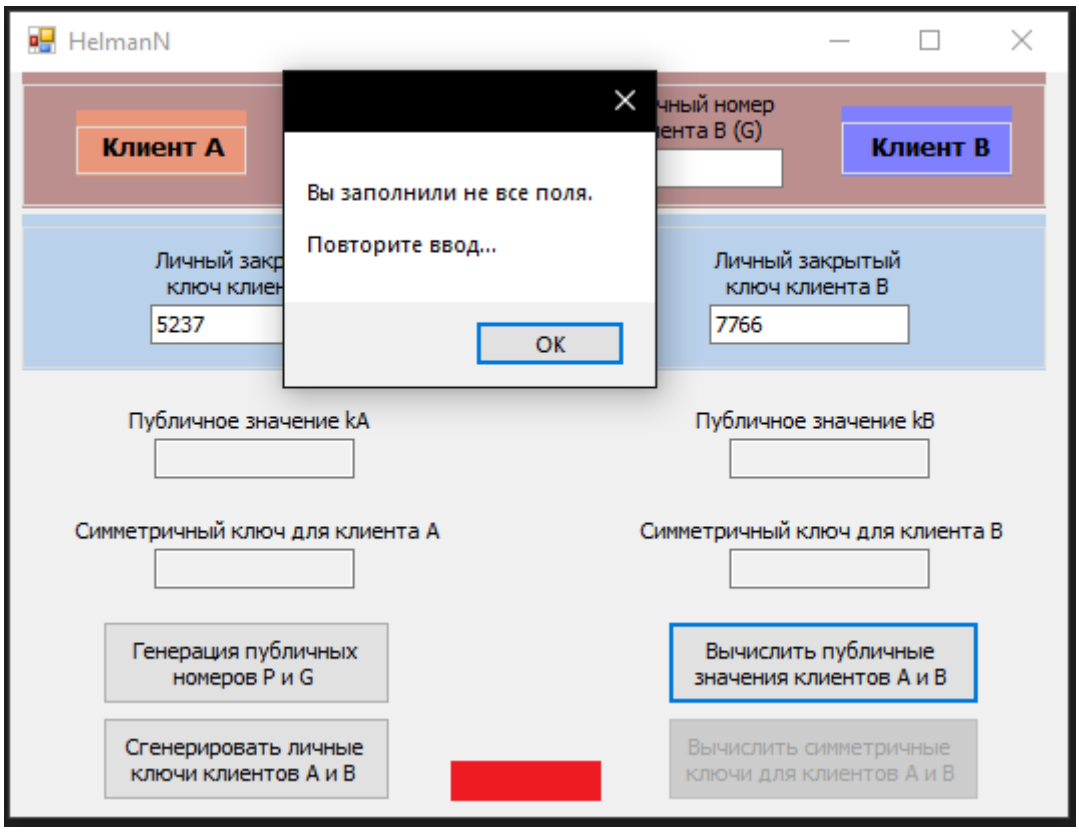

Рисунок 1.3 - Уведомление о незаполненном поле

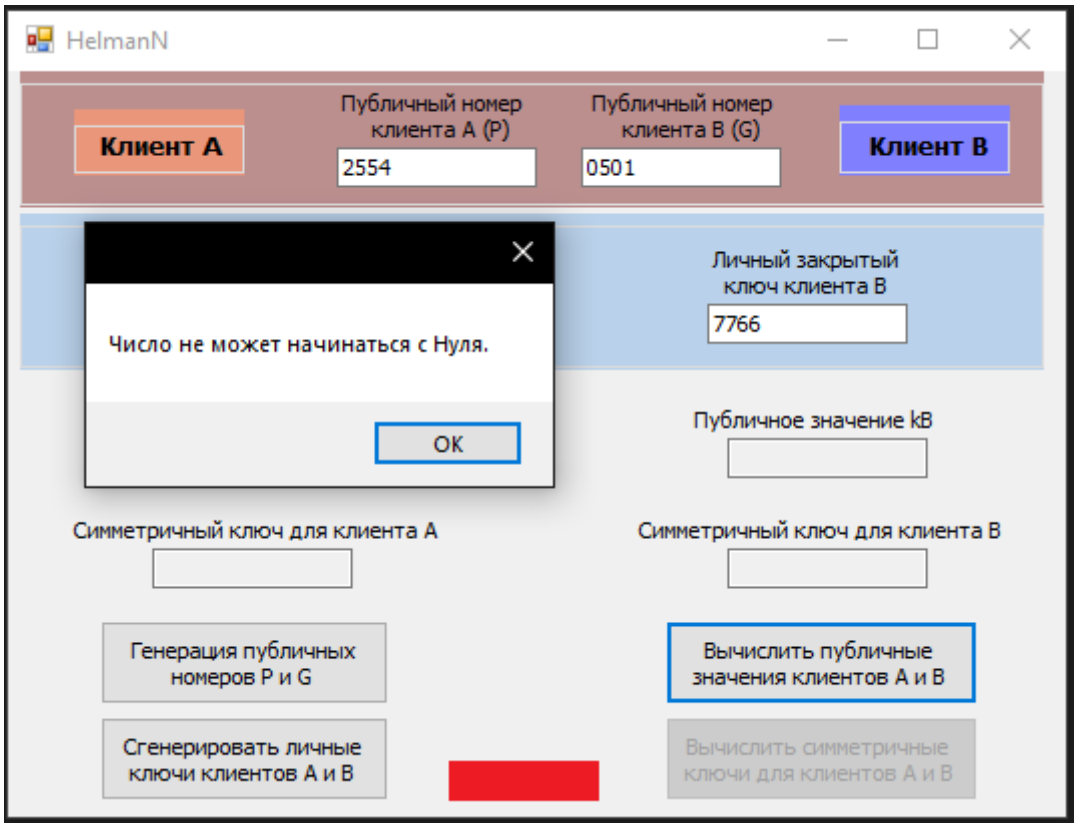

Рисунок 1.4 – Уведомление, что число не может начинаться с нуля

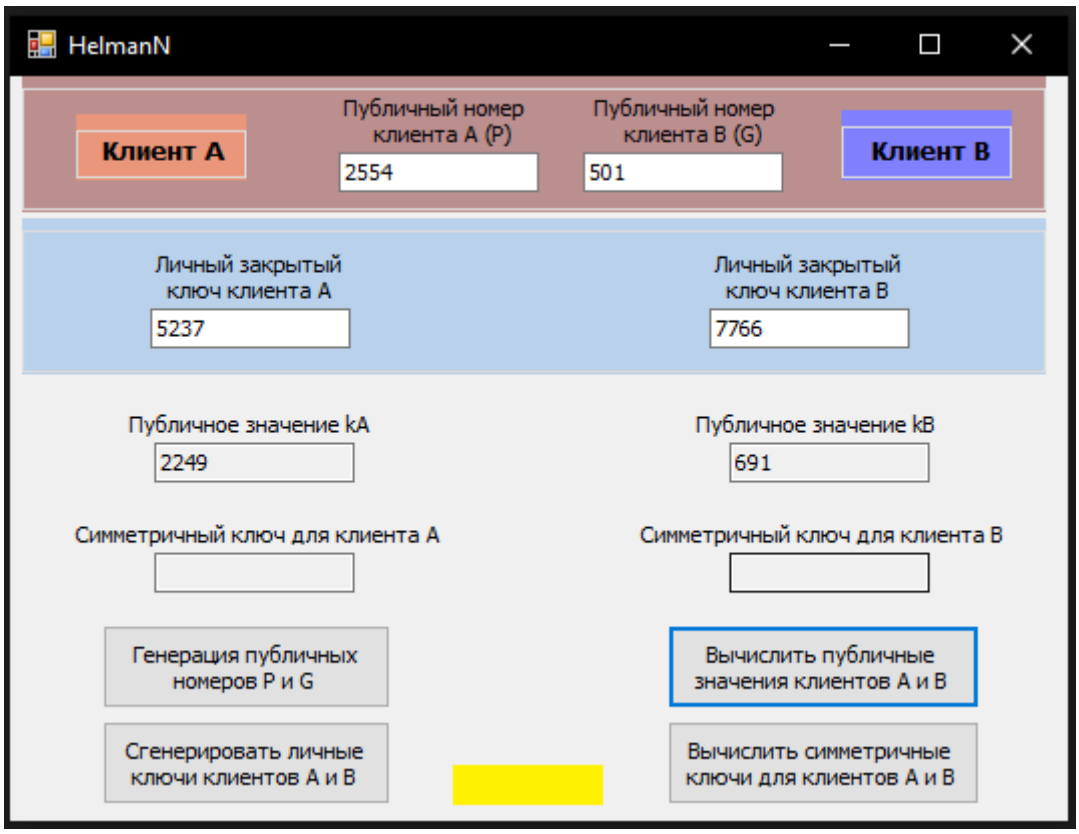

Рисунок 1.5 - Вывод промежуточных значений вычислений

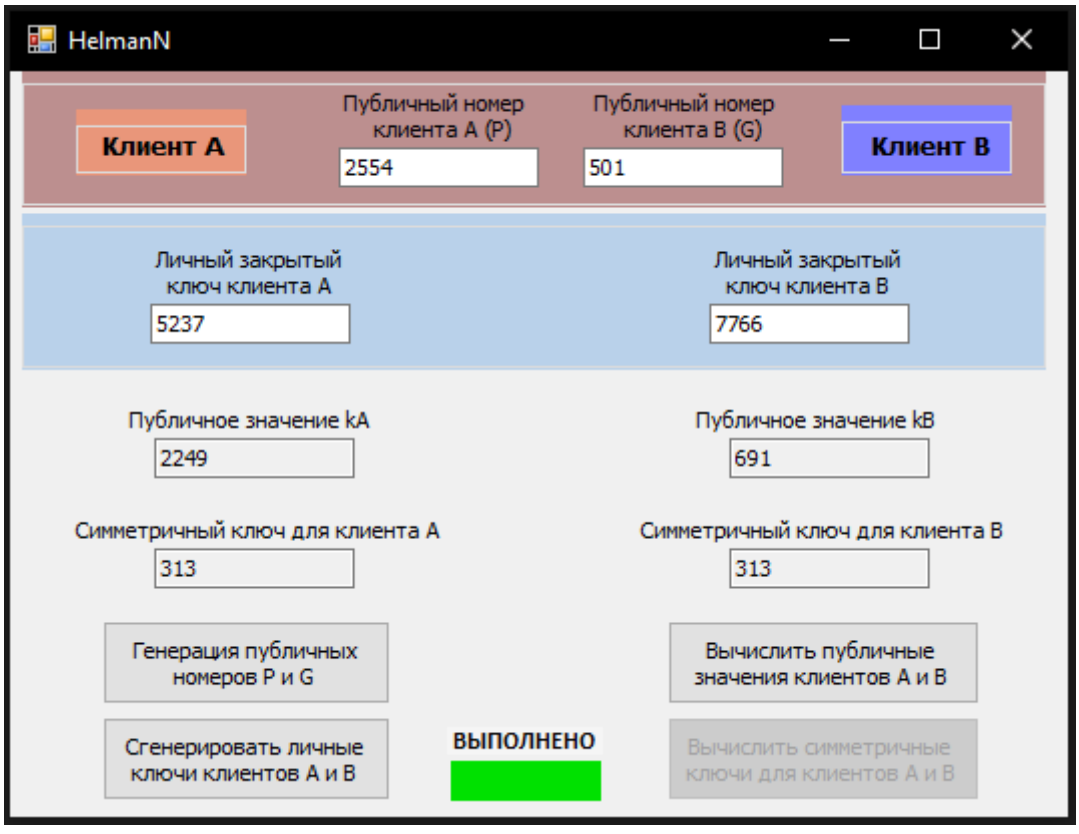

Рисунок 1.6 - Завершение вычислений

Приложение 2

Ниже приведён фрагмент кода, реализующий создание переменных неограниченной длины:

```
using System.Numerics;
...
BigInteger StepenA = G;
BigInteger StepenB = G;
BigInteger OstatokA;
BigInteger OstatokB;
BigInteger OpenkA=Convert.ToInt32(PublkB.Text);
BigInteger OpenkB=Convert.ToInt32(PublkA.Text);
…
BigInteger StepFinalKeyA=OpenkA;
BigInteger StepFinalKeyB=OpenkB;
BigInteger FinalA;
BigInteger FinalB
```

Ниже приведён фрагмент кода, реализующий операции возведение в степень и поиска остатка от деления:

```
int P = Convert.ToInt32(PublNumP.Text);
int G = Convert.ToInt32(PublNumG.Text);
int A = Convert.ToInt32(PrivKeyA.Text);
int B = Convert.ToInt32(PrivKeyB.Text);
BigInteger StepenA = G;
BigInteger StepenB = G;
for (int i = 1; i < A; i++)//возведение G в степень A
{
    StepenA = StepenA * G;
}
for (int i = 1; i < B; i++)//возведение G в степень B
{
    StepenB = StepenB * G;
}
OstatokA = StepenA % P;//поиск остатка от деления
OstatokB = StepenB % P;//поиск остатка от деления
```

Ниже приведён фрагмент кода, реализующий случайное задание исходных значений:

```
private void button1_Click(object sender, EventArgs e)
{
...
Random rnd = new Random();
int P = rnd.Next(1,9999);
int G = rnd.Next(1,9999);
PublNumP.Text = Convert.ToString(P);
PublNumG.Text = Convert.ToString(G);
}

...
private void button3_Click(object sender, EventArgs e)
{
...
Random rnd = new Random();
int SecA = rnd.Next(1, 9999);
int SecB = rnd.Next(1, 9999);
PrivKeyA.Text = Convert.ToString(SecA);
PrivKeyB.Text = Convert.ToString(SecB);
}
```

Ниже приведён фрагмент кода, реализующий функциию защиты вводных полей от ввода неподходящих значений (букв, знаков):

```
private void PrivKeyA_KeyPress(object sender, KeyPressEventArgs e)
{
char number = e.KeyChar;
if (!Char.IsDigit(number))
{
    e.Handled = true;
}
}
private void PrivKeyB_KeyPress(object sender, KeyPressEventArgs e)
{
char number = e.KeyChar;
if (!Char.IsDigit(number))
{
    e.Handled = true;
}
}
private void PublNumP_KeyPress(object sender, KeyPressEventArgs e)
{
char number = e.KeyChar;
if (!Char.IsDigit(number))
{
    e.Handled = true;
}
```

```
}
private void PublNumG_KeyPress(object sender, KeyPressEventArgs e)
{
char number = e.KeyChar;
if (!Char.IsDigit(number))
{
   e.Handled = true;
}
}
```

Ниже приведён фрагмент кода, реализующий функции проверки введённого числа и защиты от введения числа, начинающегося с нуля:

```
…
if(textP[0]=='0' || textG[0] == '0'|| textA[0] == '0'|| textB[0] == '0')
{
MessageBox.Show("Число не может начинаться с Нуля.");
int a= 0; int b = 1; a = b / a;
}
...
```

Ниже приведён фрагмент кода, реализующий функцию цветовой индикации процесса выполнения расчётов:

```
…
PicBoxG.Visible = false;
PicBoxY.Visible = false;
PicBoxR.Visible = true;
...
PicBoxG.Visible = false;
PicBoxY.Visible = true;
PicBoxR.Visible = false;
...
PicBoxG.Visible = true;
PicBoxY.Visible = false;
PicBoxR.Visible = false;
...
```

Ниже приведён фрагмент кода, реализующий функцию «моргания» индикатора со словом «ВЫПОЛНЕНО» для дополнительной демонстрации окончания расчётов:

```
…
Timer.Stop();
LBState.Visible = false;
...
```

```
Timer.Start();
...
private void Timer_Tick(object sender, EventArgs e)
{
if(LBState.Visible==true)
{
LBState.Visible = false;
}
Else
{
LBState.Visible = true;
}
}
```

Ниже приведён исходный код программного продукта:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Numerics;
using System.IO;
using System.Threading;

namespace HellmaN
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         MathFinalKeyAB.Enabled = false;
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Timer.Stop();
         LBState.Visible = false;

         PicBoxG.Visible = false;
         PicBoxY.Visible = false;
         PicBoxR.Visible = true;

         Random rnd = new Random();
         int P = rnd.Next(1,9999);
         int G = rnd.Next(1,9999);
         PublNumP.Text = Convert.ToString(P);
```

```
    PublNumG.Text = Convert.ToString(G);
}

private void button3_Click(object sender, EventArgs e)
{
    Timer.Stop();
    LBState.Visible = false;

    PicBoxG.Visible = false;
    PicBoxY.Visible = false;
    PicBoxR.Visible = true;

    Random rnd = new Random();
    int SecA = rnd.Next(1, 9999);
    int SecB = rnd.Next(1, 9999);
    PrivKeyA.Text = Convert.ToString(SecA);
    PrivKeyB.Text = Convert.ToString(SecB);
}


private void button2_Click(object sender, EventArgs e)
{
    try
    {
        Timer.Stop();
        LBState.Visible = false;

        string textP = PublNumP.Text;
        string textG = PublNumG.Text;
        string textA = PrivKeyA.Text;
        string textB = PrivKeyB.Text;

        if(textP[0]=='0' || textG[0] == '0'||textA[0]=='0'||textB[0]=='0')
        {
            MessageBox.Show("Число не может начинаться с Нуля.");
            int a= 0; int b = 1; a = b / a;
        }


        int P = Convert.ToInt32(PublNumP.Text);
        int G = Convert.ToInt32(PublNumG.Text);
        int A = Convert.ToInt32(PrivKeyA.Text);
        int B = Convert.ToInt32(PrivKeyB.Text);

        BigInteger StepenA = G;
        BigInteger StepenB = G;

        for (int i = 1; i < A; i++)//возведение G в степень A
        {
            StepenA = StepenA * G;
        }
```

```
        for (int i = 1; i < B; i++)//возведение G в степень B
        {
            StepenB = StepenB * G;
        }

        BigInteger OstatokA;
        BigInteger OstatokB;

        OstatokA = StepenA % P;
        OstatokB = StepenB % P;

        PublkA.Text = Convert.ToString(OstatokA);
        PublkB.Text = Convert.ToString(OstatokB);

        PicBoxG.Visible = false;
        PicBoxY.Visible = true;
        PicBoxR.Visible = false;

        MathFinalKeyAB.Enabled = true;
    }
    catch
    {
        MessageBox.Show("Вы заполнили не все поля.\n\nПовторите ввод...");
    }
}

private void button4_Click(object sender, EventArgs e)
{
    int A = Convert.ToInt32(PrivKeyA.Text);
    int B = Convert.ToInt32(PrivKeyB.Text);
    int P = Convert.ToInt32(PublNumP.Text);

    BigInteger OpenkA=Convert.ToInt32(PublkB.Text);
    BigInteger OpenkB=Convert.ToInt32(PublkA.Text);

    BigInteger StepFinalKeyA=OpenkA;
    BigInteger StepFinalKeyB=OpenkB;

    BigInteger FinalA;
    BigInteger FinalB;

    for (int i = 1; i < A; i++)//возведение OpenkA в степень A
    {
        StepFinalKeyA = StepFinalKeyA*OpenkA;
    }

    for (int i = 1; i < B; i++)//возведение OpenkB в степень B
    {
        StepFinalKeyB = StepFinalKeyB * OpenkB;
    }

    FinalA = StepFinalKeyA % P;
```

```
    FinalB = StepFinalKeyB % P;

    FinalKeyA.Text = Convert.ToString(FinalA);
    FinalKeyB.Text = Convert.ToString(FinalB);

    PicBoxG.Visible = true;
    PicBoxY.Visible = false;
    PicBoxR.Visible = false;

    MathFinalKeyAB.Enabled = false;

    Timer.Start();
}

private void PrivKeyA_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;
    if (!Char.IsDigit(number))
    {
        e.Handled = true;
    }
}

private void PrivKeyB_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;
    if (!Char.IsDigit(number))
    {
        e.Handled = true;
    }
}

private void PublNumP_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;
    if (!Char.IsDigit(number))
    {
        e.Handled = true;
    }
}

private void PublNumG_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;
    if (!Char.IsDigit(number))
    {
        e.Handled = true;
    }
}

private void Timer_Tick(object sender, EventArgs e)
{
```

```
            if(LBState.Visible==true)
            {
                LBState.Visible = false;
            }
            else
            {
                LBState.Visible = true;
            }
        }
    }
}
```