МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность Программное обеспечение информационной безопасности мобильных систем

**Отчёт по практическим занятиям**

по дисциплине “Основы информационной безопасности”

Выполнил:

студент 2 курса 7 группы

специальности ПОИБМС Кореневский Кирилл

Преподаватель:

Ржеутская Надежда Викентьевна

Минск 2023

**Практическое занятие №1. “Решение задачи разработки средств защиты для обеспечения максимальной эффективности объекта в условиях несанкционированного доступа”**

**Цель**: научиться решать задачи разработки средств защиты для обеспечения максимальной эффективности объекта в условиях несанкционированного доступа.

**Теоретические сведения**

Все методы защиты информации по характеру проводимых действий можно разделить на:

– законодательные (правовые);

– организационные;

– технические;

– комплексные.

Принципиальным вопросом при определении уровня защищенности объекта является выбор критериев. Рассмотрим один из них ‑ широко известный критерий "эффективность - стоимость".

Пусть имеется информационный объект, который при нормальном (идеальном) функционировании создает положительный эффект (экономический, политический, технический и т.д.). Этот эффект обозначим через *Е0*. Несанкционированный доступ к объекту уменьшает полезный эффект от его функционирования (нарушается нормальная работа, наносится ущерб из-за утечки информации и т.д.) на величину *ΔЕ*. Тогда эффективность функционирования объекта с учетом воздействия несанкционированного доступа:

|  |  |
| --- | --- |
|  | (1) |

Относительная эффективность:

|  |  |
| --- | --- |
|  | (2) |

Уменьшение эффективности функционирования объекта приводит к материальному ущербу для владельца объекта. В общем случае материальный ущерб есть некоторая неубывающая функция от ΔЕ:

|  |  |
| --- | --- |
|  |  |

Будем считать, что установка на объект средств защиты информации уменьшает негативное действие несанкционированного доступа на эффективность функционирования объекта. Обозначим снижение эффективности функционирования объекта при наличии средств защиты через ΔЕ3, а коэффициент снижения негативного воздействия несанкционированного доступа на эффективность функционирования объект ‑ через К, тогда:

|  |  |
| --- | --- |
|  |  |

где К≥1.

Выражения (1) – (2) примут вид:

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |

Поскольку затраты на установку средств защиты можно рассматривать как ущерб владельцу объекта от возможности осуществления несанкционированного доступа, то суммарный ущерб объекту:

|  |  |
| --- | --- |
|  |  |

Если эффективность функционирования объекта имеет стоимостное выражение (доход, прибыль и т.д.), то UΣ непосредственно изменяет эффективность:

|  |  |
| --- | --- |
|  |  |

**Условие задачи**

Решить задачу разработки средств защиты для обеспечения максимальной эффективности объекта в условиях несанкционированного доступа в соответствии с вариантом.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | *E*0 | *E* | *K* | *C* |
| 11 | 19000 | 17000 | 4 | 800 |

**Решение**

0,89



0,93

Вывод: Я научился решать задачи разработки средств защиты для обеспечения максимальной эффективности объекта в условиях несанкционированного доступа и выяснил, что с данными значениями моего варианта защита эффективна, т.к. > .

**Практическое задание №2. «Разработка политики информационной безопасности бизнес-компании»**

**Цель**: Разработать проект политики информационной безопасности бизнес-компании.

**Введение**

Сегодня невозможно представить практически ни одну сферу деятельности без средств вычислительной техники и телекоммуникаций. Информационные технологии предлагают все новые и новые сервисы. Через Интернет становятся доступными электронные платежные системы, персональные финансовые порталы, электронные биржи и т. д. Внедряя новые услуги, компании укрепляют свое положение на рынке. С другой стороны, вопросы информационной безопасности всегда, как минимум, на шаг отстают от нововведений. Достаточно посмотреть, сколько версий должно миновать, пока тот или иной программный продукт не приобретет репутацию надежного. Эти две очевидные тенденции приводят к необходимости здорового консерватизма в построении информационных систем корпоративного уровня. Нужен взвешенный подход, учитывающий риски и стоимость отдачи от вложенных средств. Информационные технологии все шире применяются в качестве инструмента ведения бизнеса. И понятие риска является краеугольным как для бизнеса, так и для системы организации информационной безопасности. Но при этом следует понимать, что стандарты и передовые практики обычно не рассматривают конкретные программы и детальные методики, а концентрируются на основных принципах и примерах. С точки зрения рисков следует иметь в виду два аспекта:

- Во-первых, любую систему безопасности можно взломать, имея достаточно ресурсов и времени. Поэтому риски могут быть идентифицированы и уменьшены, но никогда не сведены полностью на нет.

- Во-вторых, все организации разные, поэтому процесс смягчения рисков для каждой имеет свои уникальные черты.

**Описание структуры**

Риэлторское агентство - это организация, специализирующаяся на предоставлении услуг в сфере недвижимости. Оно играет важную роль в процессе купли-продажи, аренды и управления недвижимостью. Структура риэлторского агентства может варьироваться в зависимости от его размера и организационной модели, но в общем случае включает следующие основные элементы:

* Директор/владелец: Ответственное лицо, принимающее стратегические решения и управляющее деятельностью агентства. Он может быть одним из основателей или назначенным руководителем.
* Отдел продаж: Занимается привлечением клиентов, оценкой и продажей недвижимости. Включает в себя риэлторов, агентов по продажам и других специалистов, работающих напрямую с клиентами.
* Отдел аренды: Занимается арендой недвижимости. Он отвечает за поиск арендаторов, заключение договоров аренды, управление арендным парком и связанные с этим аспекты.
* Отдел управления недвижимостью: Этот отдел осуществляет услуги по управлению имуществом от имени собственников. Включает в себя управляющих объектами недвижимости, бухгалтерию и других специалистов, занимающихся административными и финансовыми аспектами управления.
* Отдел маркетинга и рекламы: Занимается разработкой и реализацией маркетинговых стратегий, рекламой недвижимости, созданием презентаций и рекламных материалов.
* Юридический отдел: Специалисты в области недвижимости, юристы и нотариусы, которые предоставляют правовую поддержку и консультации по сделкам с недвижимостью, составляют и проверяют договоры и обеспечивают законность сделок.
* Административный отдел: Включает в себя администраторов, секретарей и другой персонал, обеспечивающий поддержку офисной работы, ведение документации, прием и распределение звонков и т.д.

Кроме того, риэлторское агентство может иметь специализированные подразделения или отделы, такие как отдел коммерческой недвижимости, отдел новостроек или отдел оценки недвижимости, в зависимости от фокуса своей деятельности.

Структура риэлторского агентства может быть гибкой и подстраиваться под конкретные потребности и стратегии организации. Важно, чтобы каждый отдел работал в тесном взаимодействии друг с другом, обеспечивая эффективное и качественное обслуживание клиентов и достижение целей агентства в сфере недвижимости.

**Оценка рисков**

Политика безопасности строится на основе анализа рисков, которые признаются реальными для информационной системы организации. Когда риски проанализированы, стратегия защиты определена, тогда составляется программа, реализация которой должна обеспечить информационную безопасность. Под эту программу выделяются ресурсы, назначаются ответственные, определяется порядок контроля выполнения программы.

При оценивании рисков учитывается:

* ценность ресурсов;
* оценка значимости угроз;
* эффективность существующих и планируемых средств защиты.

Анализ рисков состоит в том, чтобы выявить существующие риски и предложить способ их оценки.

В самом широком смысле мера риска может рассматриваться в дальнейшем как описание видов неблагоприятных действий, воздействию которых может подвергнуться система или организация, и вероятностей того, что эти действия могут произойти. Результат этого процесса должен указать риэлторскому агенству степень риска, связанного с определенными ценностями. Этот результат важен, потому что он является основой для дальнейшего выбора средств защиты и решений по минимизации риска.

**Существует вероятность возникновения следующих нарушений информационной безопасности:**

• получение неправомерного доступа к информации, другими словами, нарушение конфиденциальности;

• утрата сведений, вызванная разрушением носителя информации или стиранием данных;

• внесение изменений при прямом доступе к базе данных или через интерфейс системы;

• отказ функционала, связанный с получением доступа к информации;

• получение доступа к базе данных – полное или частичное;

• некорректное функционирование информационной системы вследствие несанкционированного изменения модулей.

**Вот несколько примеров угроз риэлторского агентства:**

Риэлторское агентство может стать **целью хакеров**, которые могут попытаться получить несанкционированный доступ к базам данных клиентов или финансовой информации. Кибератаки могут включать в себя вредоносные программы, фишинговые атаки, атаки на слабые места в сетевой инфраструктуре и другие методы.

**Утечка конфиденциальных данных** клиентов или бизнес-партнеров может возникнуть в результате кибератак, недостаточной безопасности систем или ошибок персонала. Утечка может привести к утрате доверия клиентов, юридическим проблемам и финансовым потерям.

Риэлторское агентство может подвергаться **физическим угрозам**, таким как кражи оборудования, взлом офиса или повреждение сетевой инфраструктуры. Это может привести к нарушению работы, потере данных или нарушению безопасности информации.

Злоумышленники могут использовать методы **социальной инженерии**, чтобы обмануть сотрудников и получить доступ к конфиденциальной информации. Например, они могут представиться сотрудниками агентства или внешними контрагентами и попросить предоставить им доступ к системам или конфиденциальным данным.

**Технические сбои**, отказы оборудования или программного обеспечения могут привести к временной недоступности информации или нарушению работы риэлторского агентства. Это может повлиять на клиентское обслуживание и вызвать финансовые потери.

**Разработка мер защиты**

Способы усиления защиты сведений в информационных системах риэлторской компании:

*Установка и обновление антивирусного программного обеспечения*: Риэлторская компания должна установить надежное антивирусное программное обеспечение на всех компьютерах и серверах, а также регулярно обновлять его для обнаружения и предотвращения новых угроз.

*Применение сильных паролей и авторизация с двухфакторной аутентификацией*: Сотрудники должны использовать сложные пароли, содержащие цифры, буквы и специальные символы. Кроме того, рекомендуется включить двухфакторную аутентификацию для дополнительного слоя защиты.

*Шифрование данных*: Чувствительные данные, хранящиеся на серверах или передаваемые по сети, должны быть зашифрованы. Использование современных алгоритмов шифрования поможет предотвратить несанкционированный доступ к информации.

*Регулярное создание резервных копий данных*: Риэлторская компания должна регулярно создавать резервные копии всех важных данных и хранить их в безопасном месте. Это поможет восстановить информацию в случае утери или повреждения данных.

*Обновление программного обеспечения и патчей безопасности*: Компания должна следить за выходом обновлений программного обеспечения и патчей безопасности, и своевременно их устанавливать. Это поможет исправить уязвимости, которые могут быть использованы злоумышленниками.

*Обучение сотрудников по безопасности*: Риэлторская компания должна проводить регулярные обучающие программы по безопасности информации для своих сотрудников. Это поможет повысить осведомленность о возможных угрозах и обучить сотрудников правилам безопасного поведения.

*Ограничение доступа к данным*: Компания должна регулировать доступ сотрудников к информации, предоставляя разные уровни доступа в зависимости от их должностей и необходимости работы. Также необходимо периодически обновлять права доступа.

*Мониторинг и регистрация событий*: Важно установить систему мониторинга и регистрации событий, которая будет отслеживать подозрительную активность в сети и системе. Это поможет выявить атаки или необычное поведение и принять меры вовремя.

*Защита физической инфраструктуры*: Риэлторская компания должна обеспечить физическую безопасность своих серверов, сетевого оборудования и других информационных ресурсов. Это может включать контроль доступа в помещение, видеонаблюдение и использование защитных механизмов, таких как замки и сигнализация.

*Регулярная аудитория безопасности*: Риэлторская компания должна проводить регулярные аудиты безопасности, включающие сканирование уязвимостей, проверку соответствия политикам безопасности, анализ журналов событий и тестирование на проникновение. Это поможет выявить слабые места в системе и принять меры для их устранения.

**Вывод**

В данной лабораторной работе были приведены примеры угроз информационной безопасности, связанных с риэлторской компанией, а также разработаны меры защиты, которые могут быть применены для обеспечения безопасности информационной инфраструктуры.

Информационная безопасность является важным аспектом для организаций, особенно в сфере, где хранятся и обрабатываются чувствительные данные клиентов и бизнес-информация. Риэлторская компания, занимающаяся продажей и арендой недвижимости, также подвержена ряду угроз, которые могут привести к серьезным последствиям, включая утрату данных, нарушение доверия клиентов и финансовые потери.

**Практическое занятие 3**

**Тема «Настройка Брандмауэра Windows»**

**Цель**

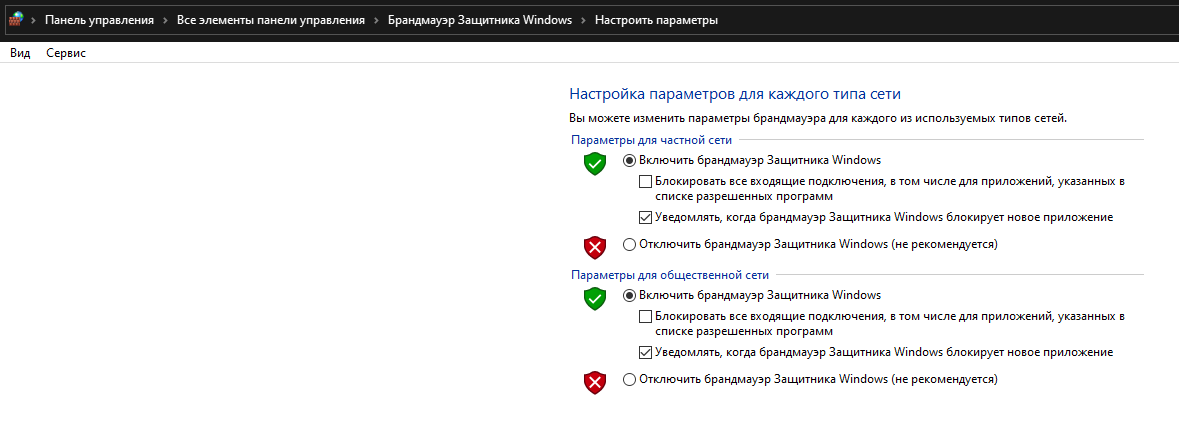
Овладеть навыками настройки и использования Брандмауэра Windows.

**Ход работы**

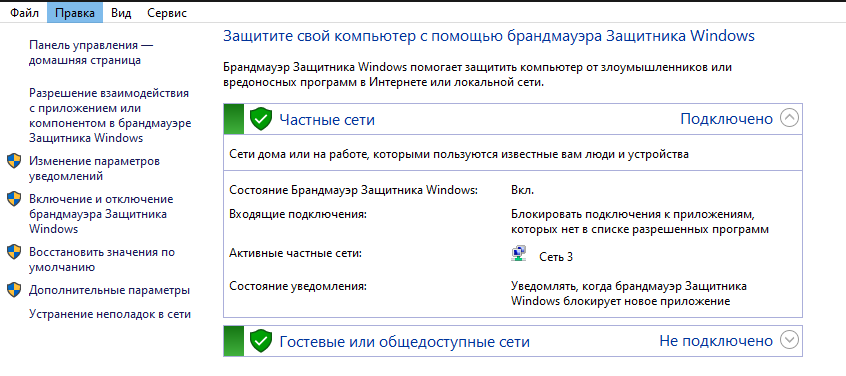
Брандмауэр – это защитный экран между глобальным интернетом и локальной компьютерной сетью организации. Он выполняет функцию проверки и фильтрации данных, поступающих из интернета. В зависимости от настроек брандмауэр может пропустить их или заблокировать (например, если обнаружит «червей», вирусы и хакерскую атаку).

Нужно различать сетевой брандмауэр (или, по-другому, сетевой экран) и брандмауэр, встроенный в операционную систему Windows. В первом случае решение устанавливается на границе (физической или логической) компьютерной инфраструктуры организации и защищает все ПК, подключенные к локальной сети – как программное, так и программно-аппаратное решение. Во втором случае это программа, работающая для защиты отдельно взятого компьютера пользователя.

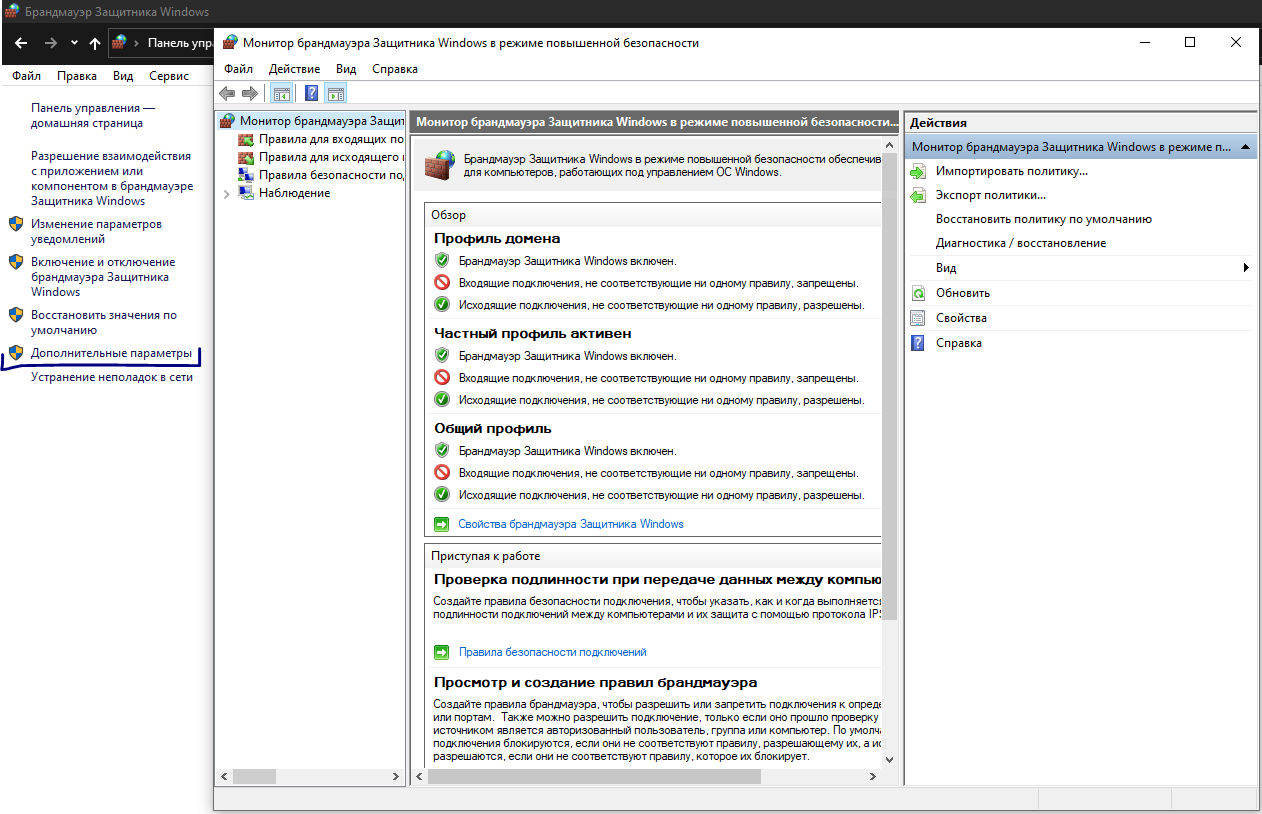
Для начала включим Брандмауэр Защитника Windows, перейдя в Панель управления и выбрав соответствующий пункт:



Включенный Брандмауэр выглядит следующим образом:

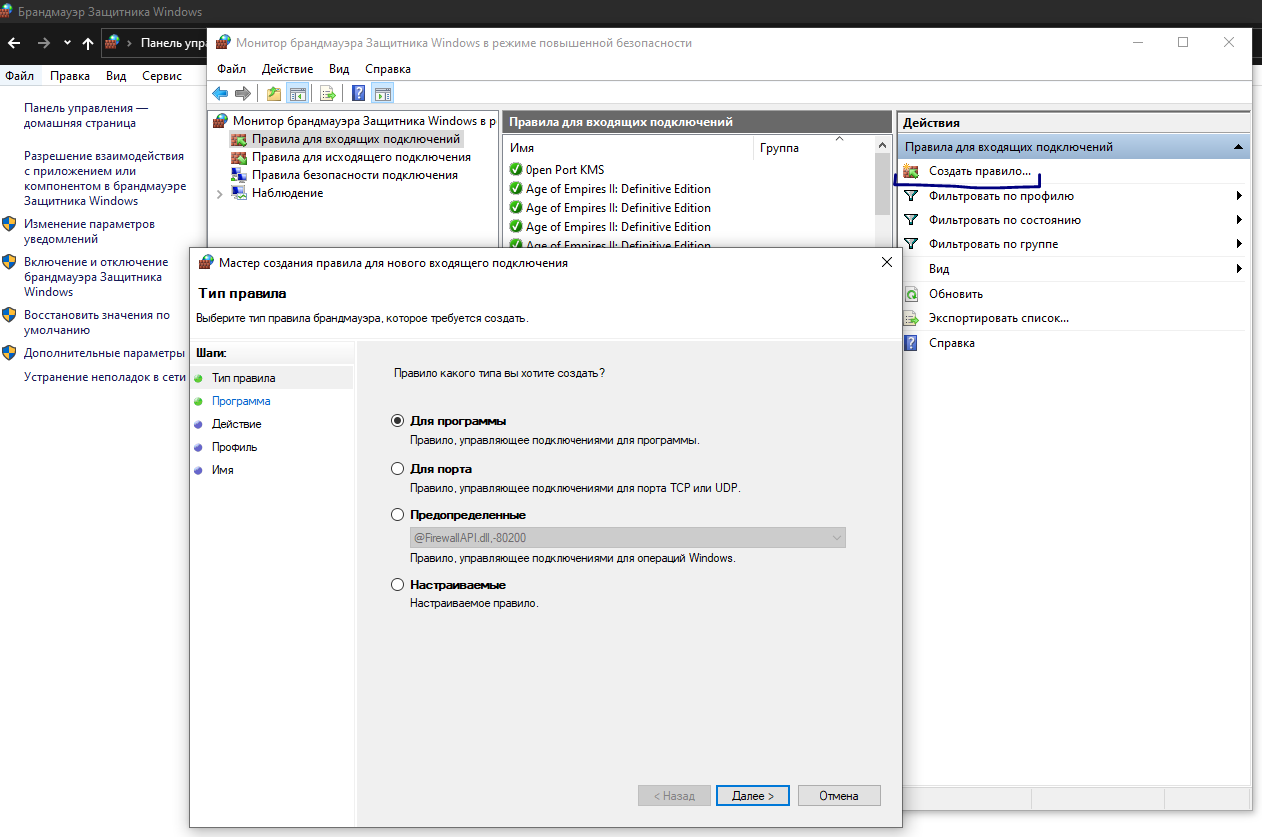


Перейдем во вкладку дополнительные параметры:

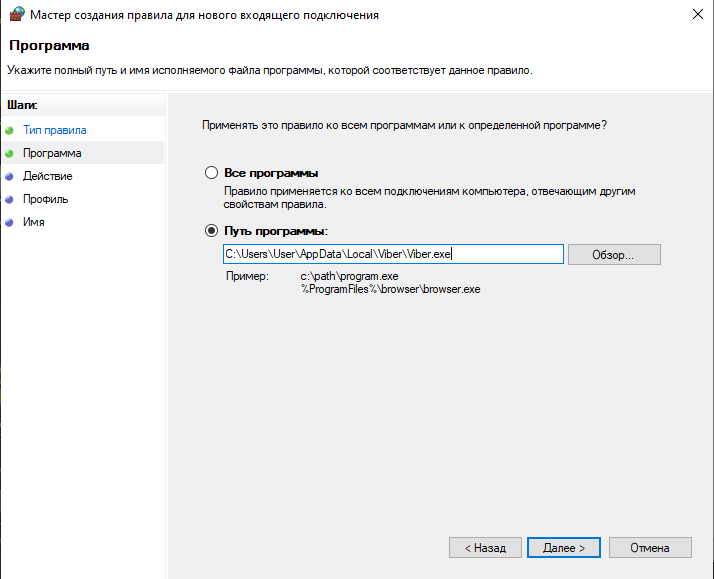


1. Создать правила для **входящих подключений** (с помещением в электронный конспект копий экрана с пояснениями промежуточных действий):
   1. Для 1 программы (по выбору) на **блокировку** подключения

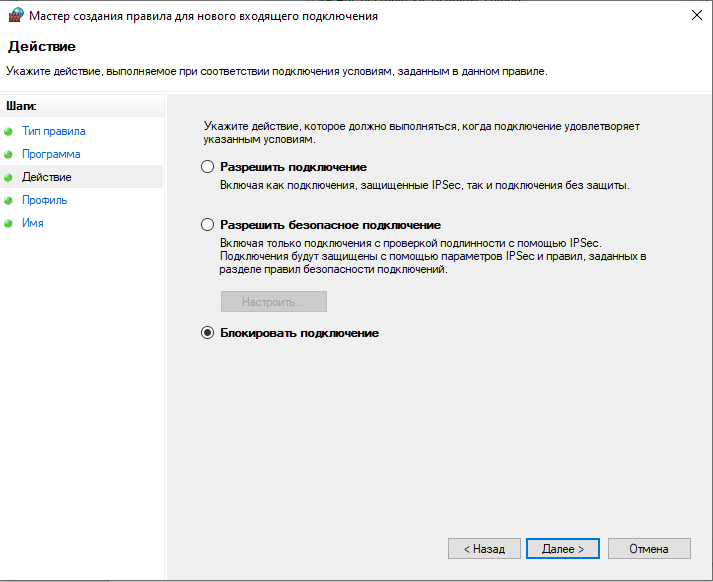
В левом меню выберем пункт «Правила для входящих подключений», в правом контекстном меню – пункт «Создать правило». В появившемся окне выбираем тип – «Для программы»:



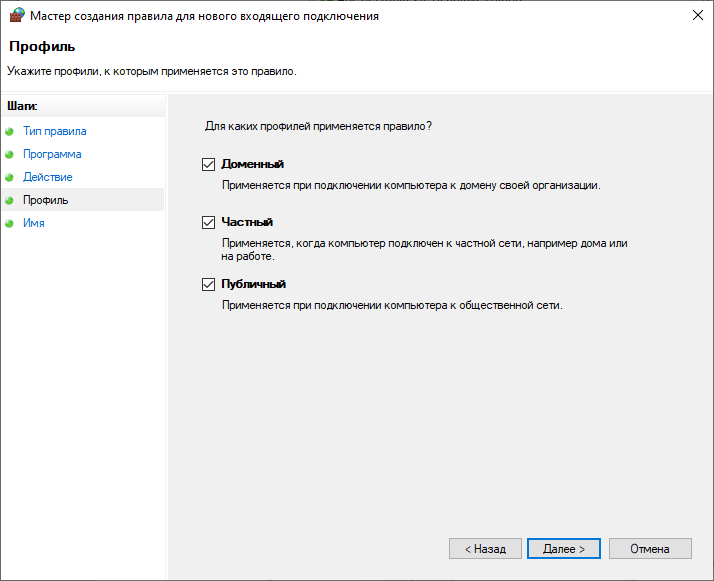
Указываем путь к программе:



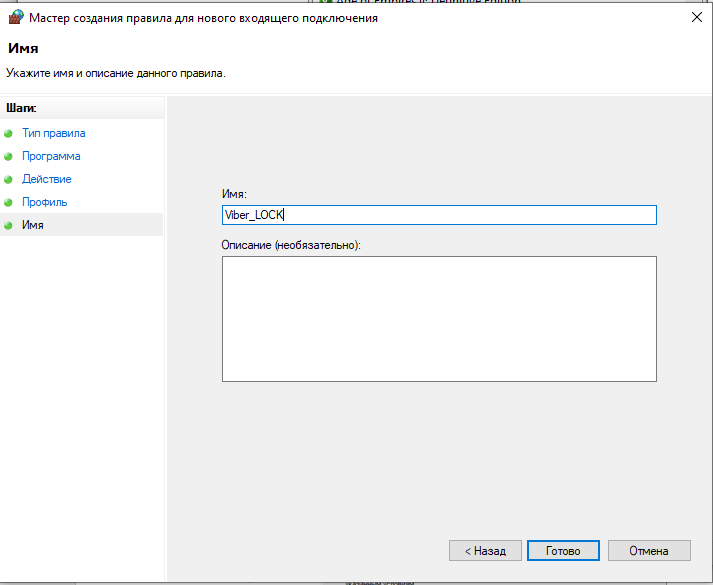
Выбираем действие «Блокировать подключение»:



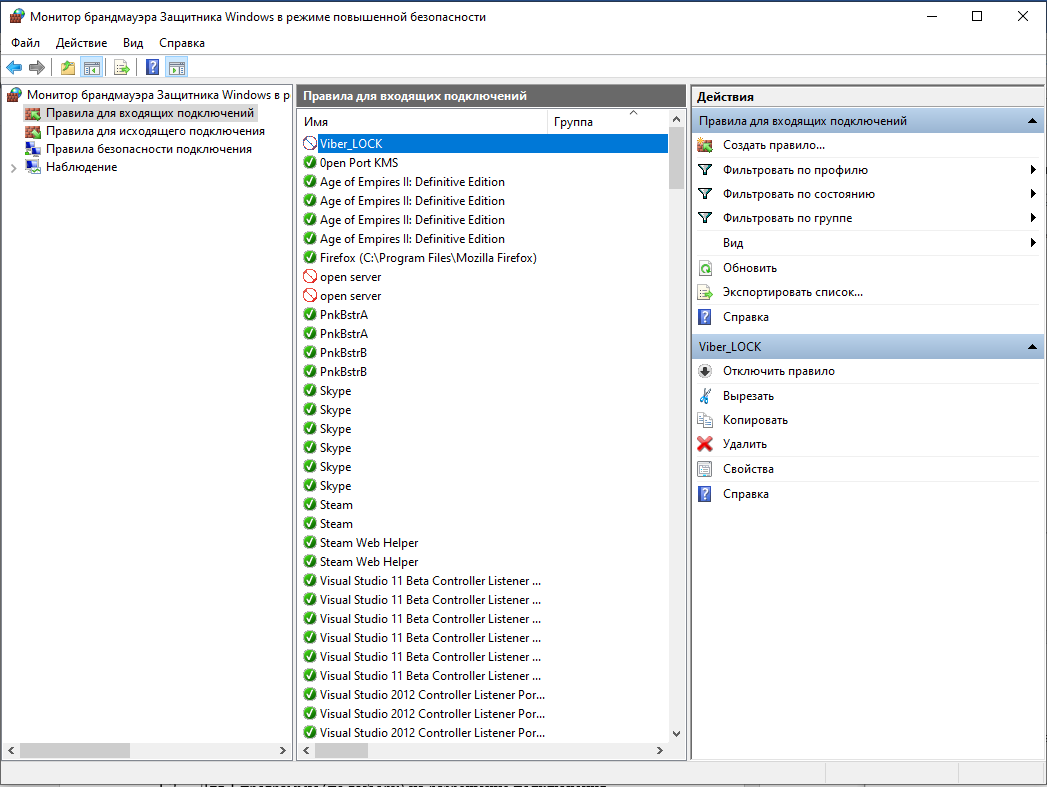
Оставляем по умолчанию:



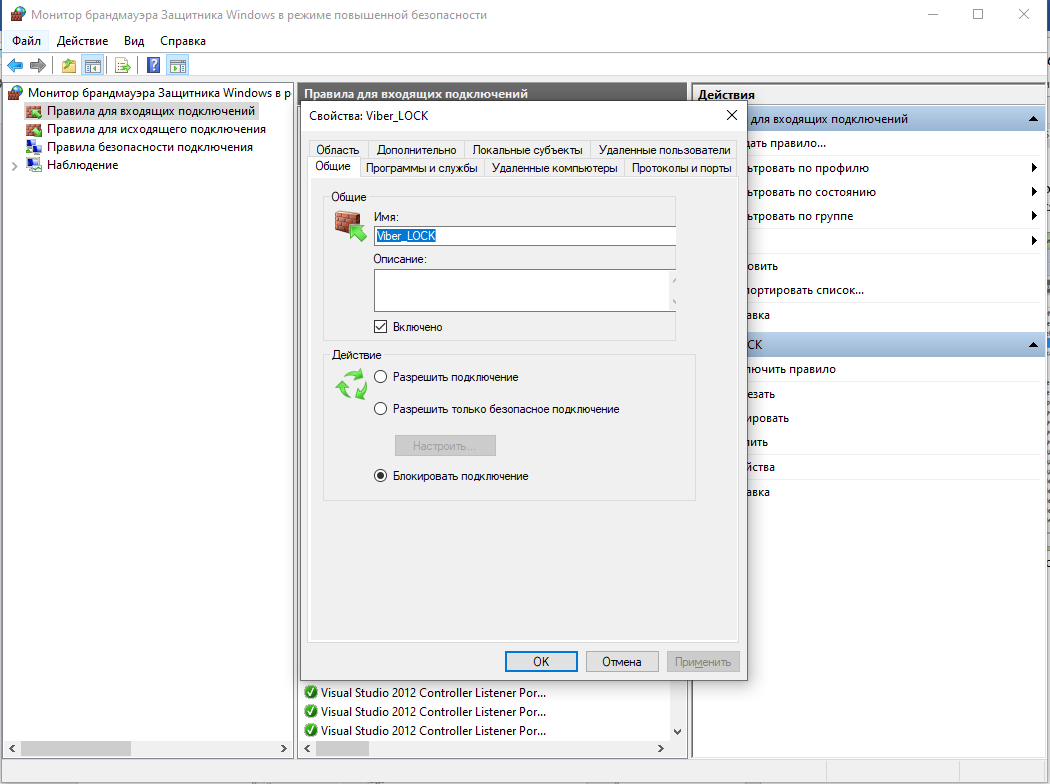
Назначим правилу имя:



В общем списке появилось созданное правило:

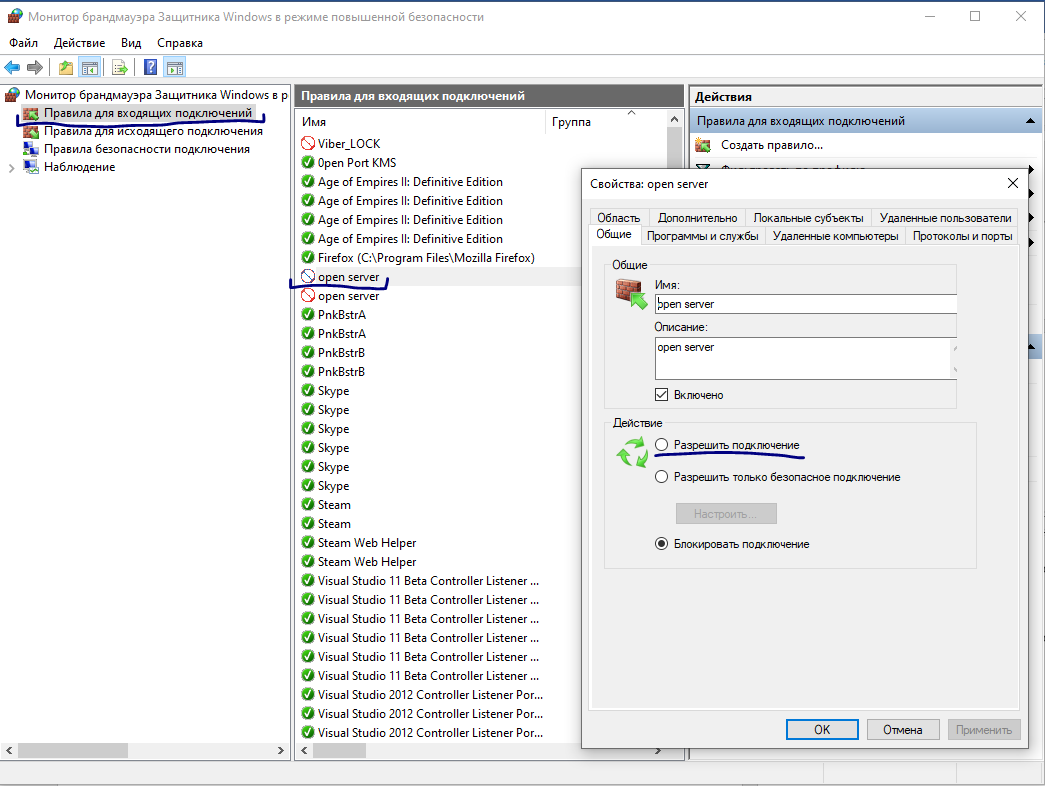


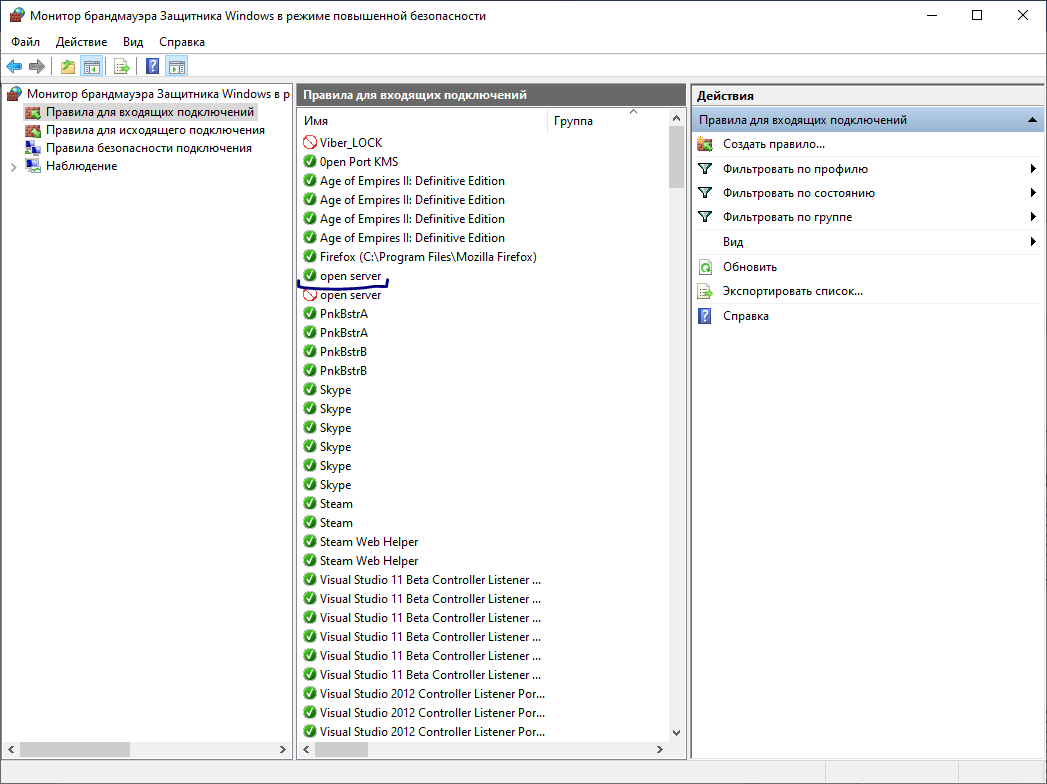
При двойном нажатии можно просмотреть свойства правила:



* 1. Для 1 программы (по выбору) на **разрешение** подключения

Создадим разрешение через свойства правила:



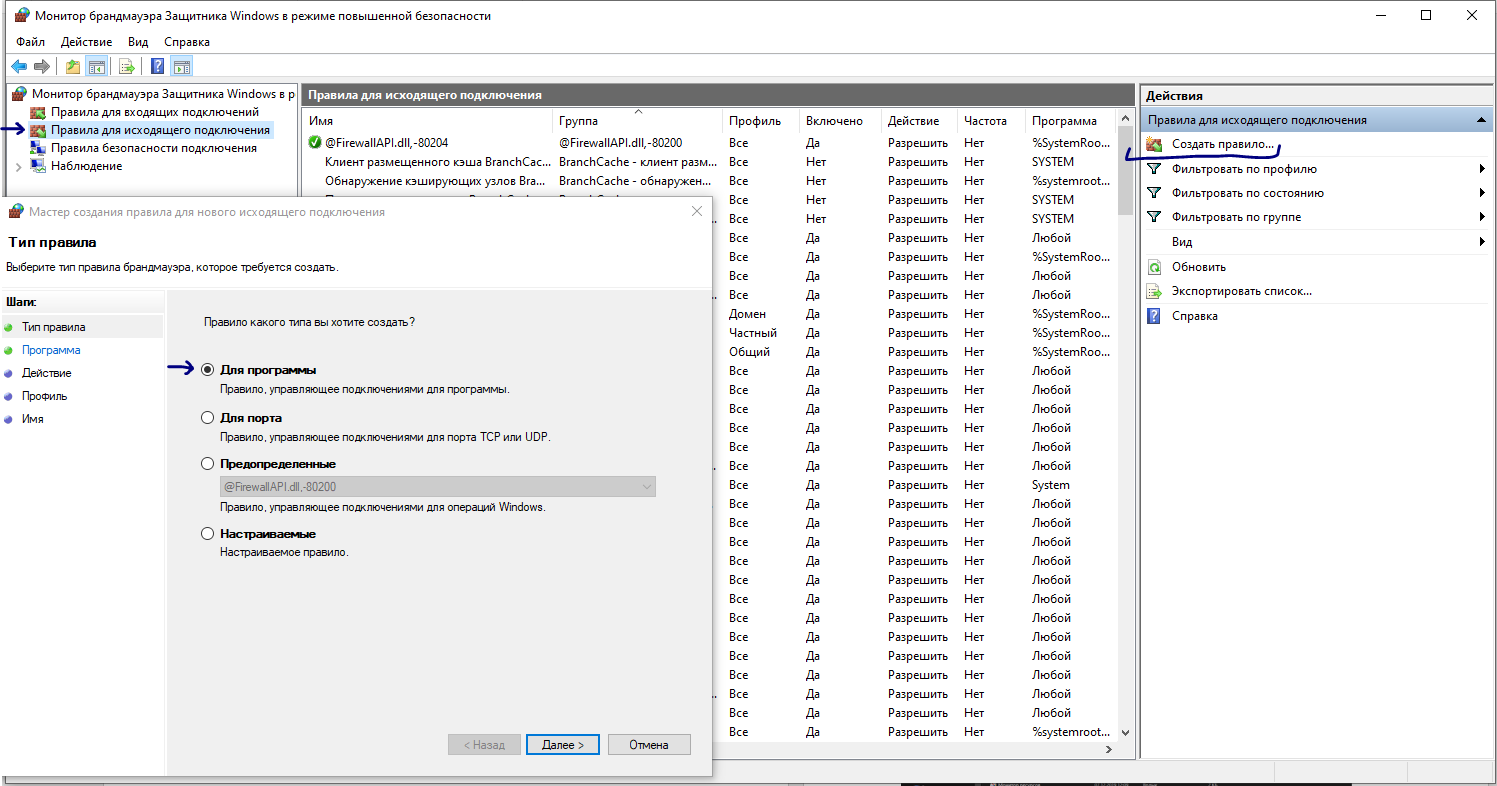


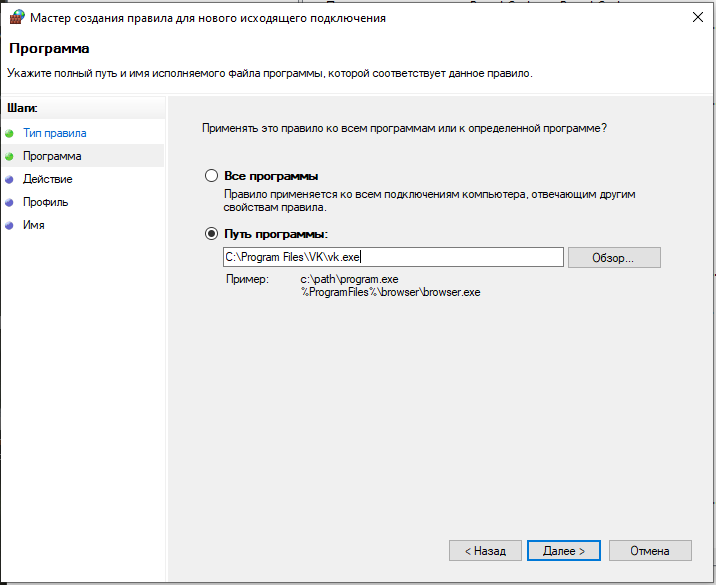
1. Создать правила **для исходящих подключений** (с помещением в электронный конспект копий экрана с пояснениями действий):

*Исходящие подключение: с персонального ПК подключение к серверу.*

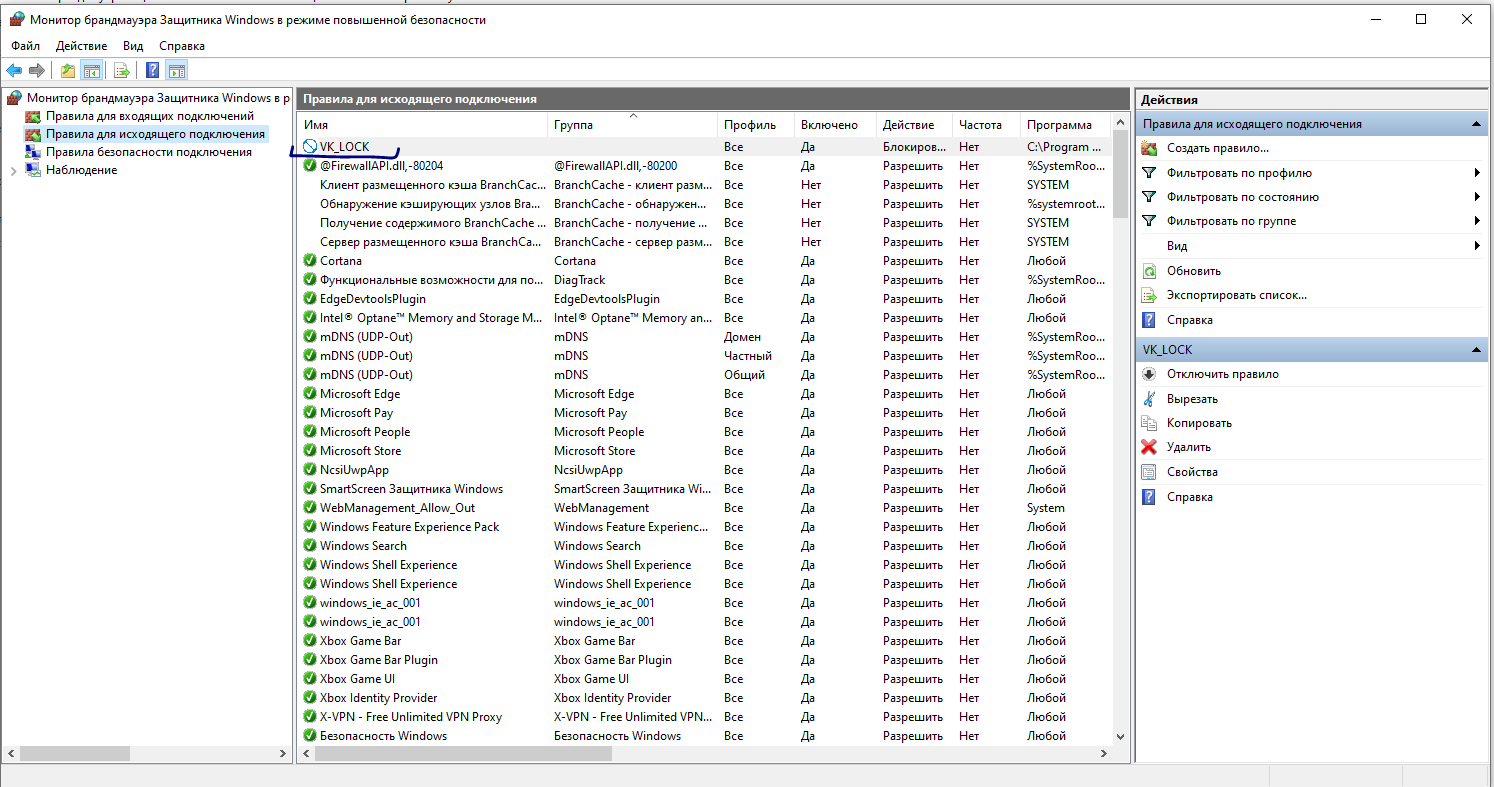
*Входящее подключение: на персональный ПК с сервера.*

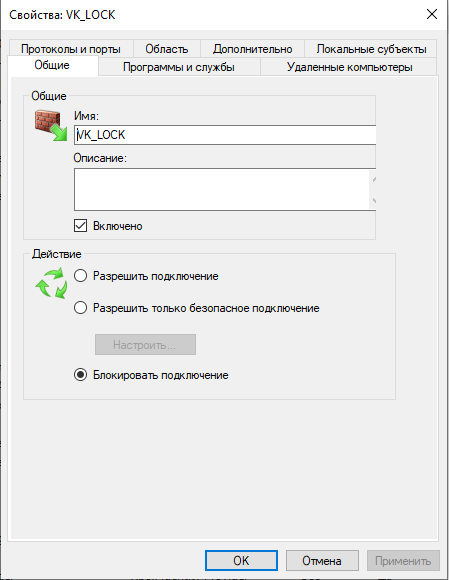
* 1. Для 1 программы (по выбору) на **блокировку** подключения



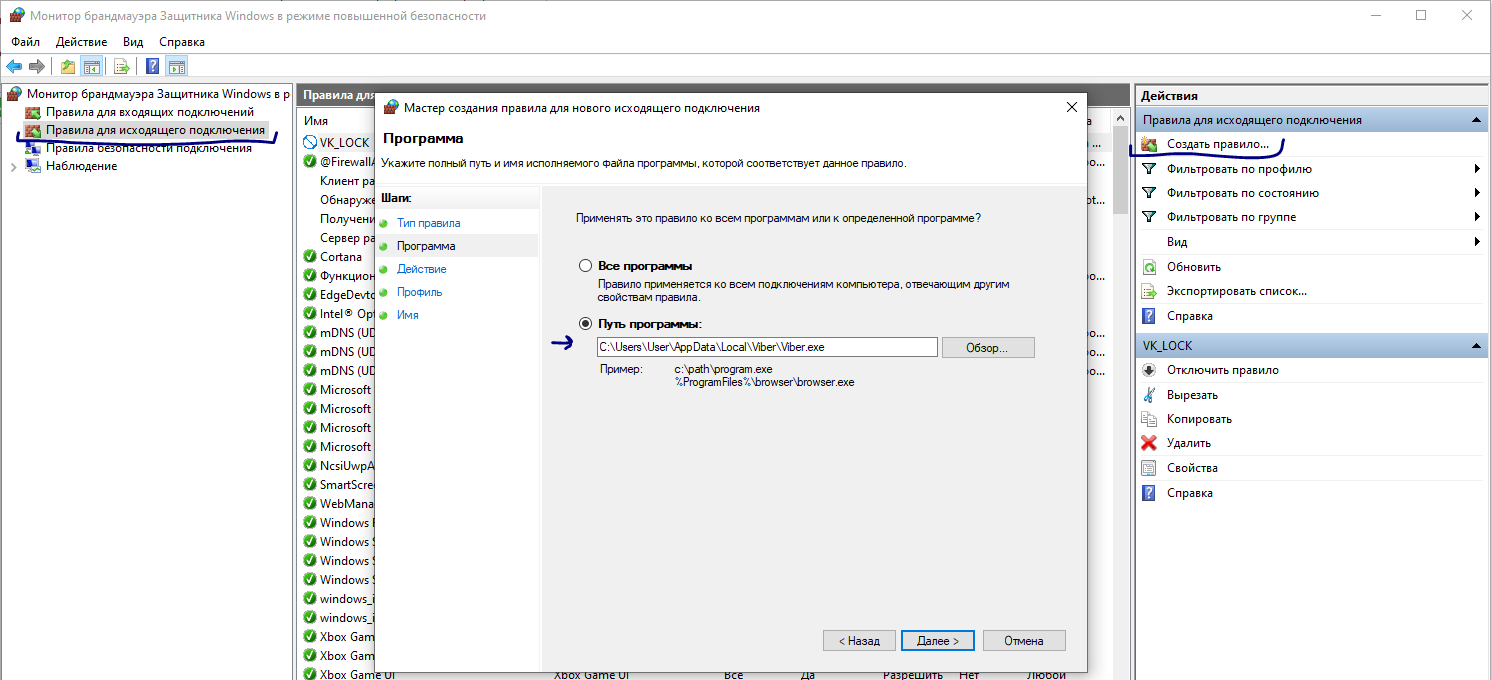


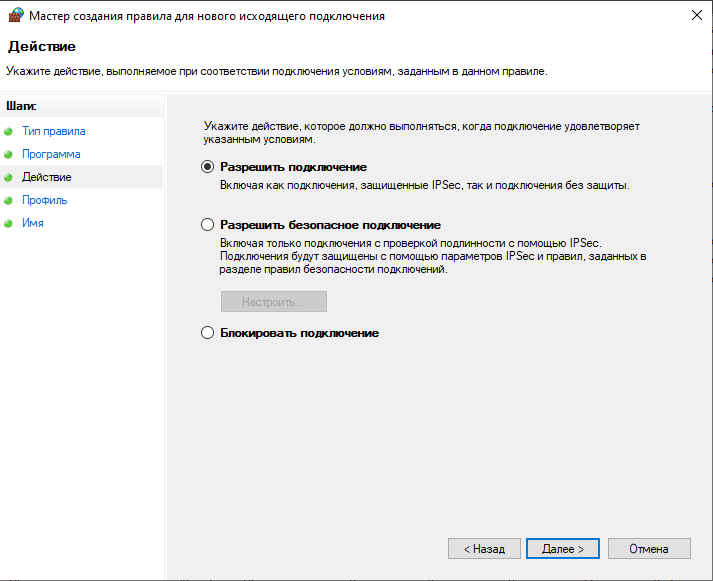
Стандартно задаем имя правила, применяем его. Правило появится в общем списке, двойным щелчком можно просмотреть его свойства:



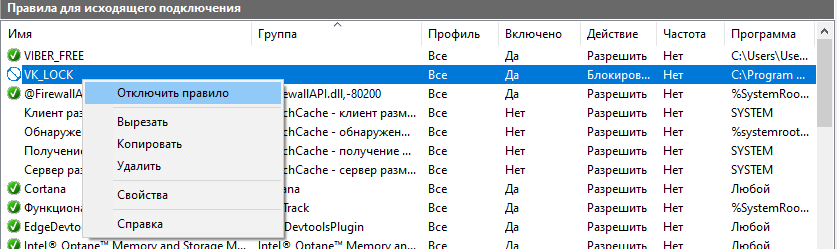


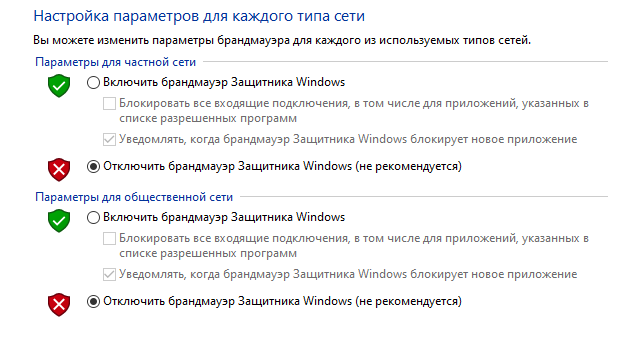
* 1. Для 1 программы (по выбору) на **разрешение** подключения



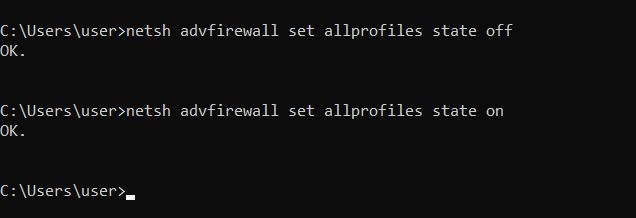


1. Вернуть настройки Брандмауэра в исходное состояние до начала выполнения практического задания.

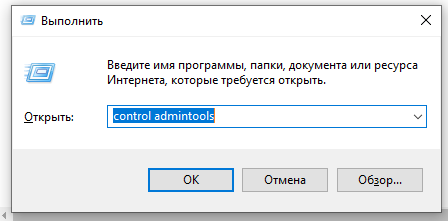


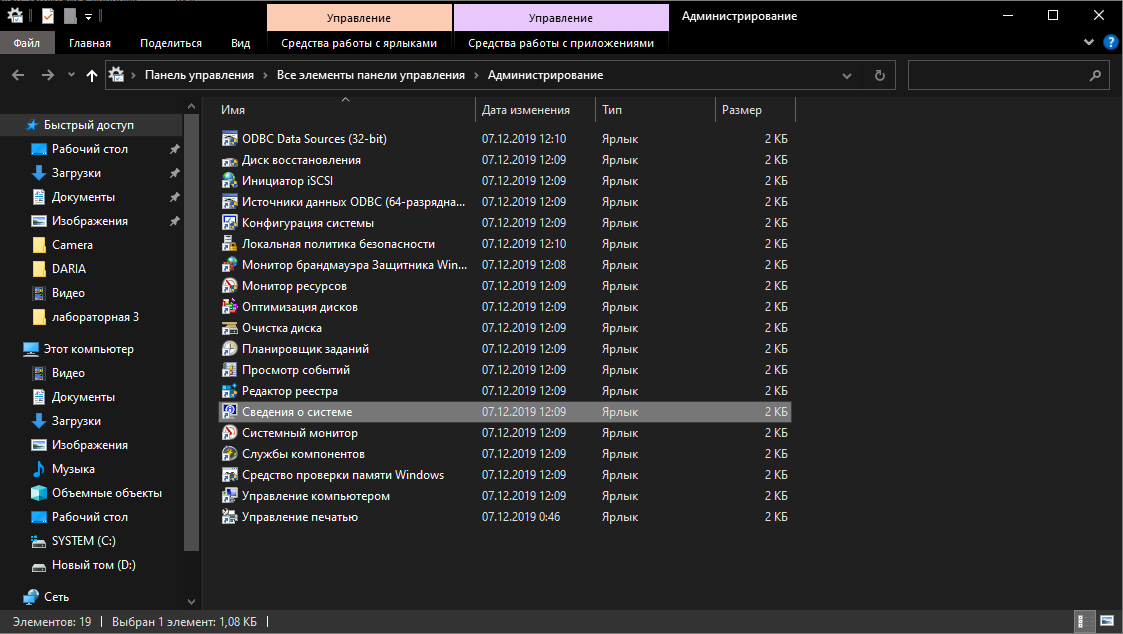


Выключение/включение брандмауэра из командной строки для всех профилей:

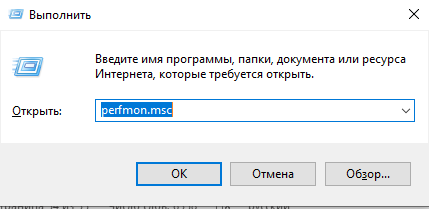


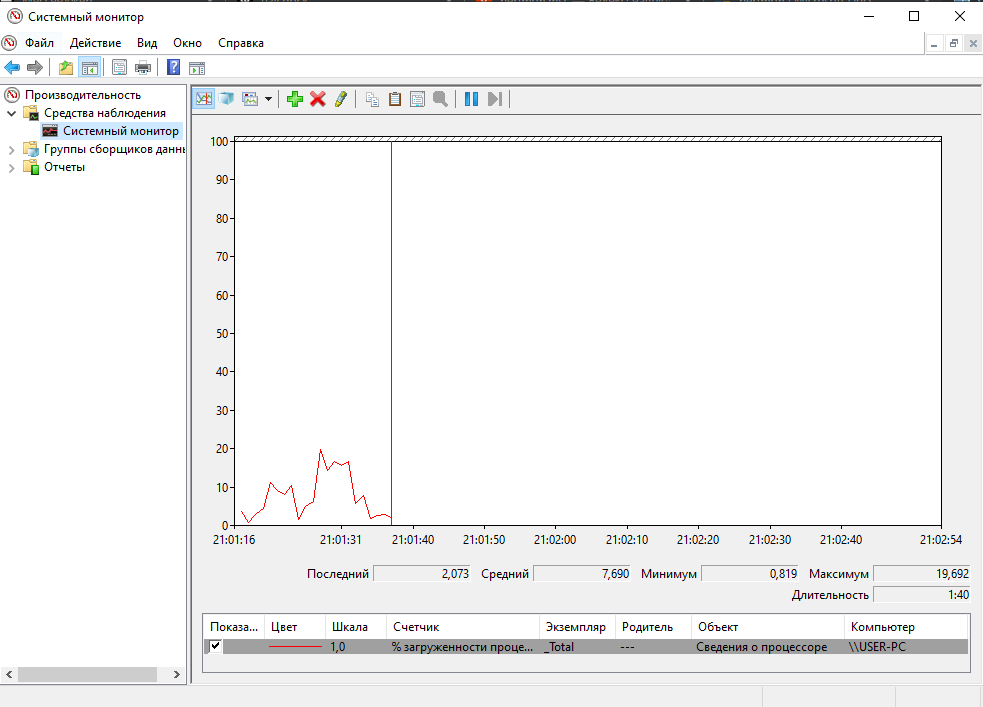
1. Администрирование – Win + R – **control admintools**: ведет к папке в панели управления, содержащей средства для системного администрирования.





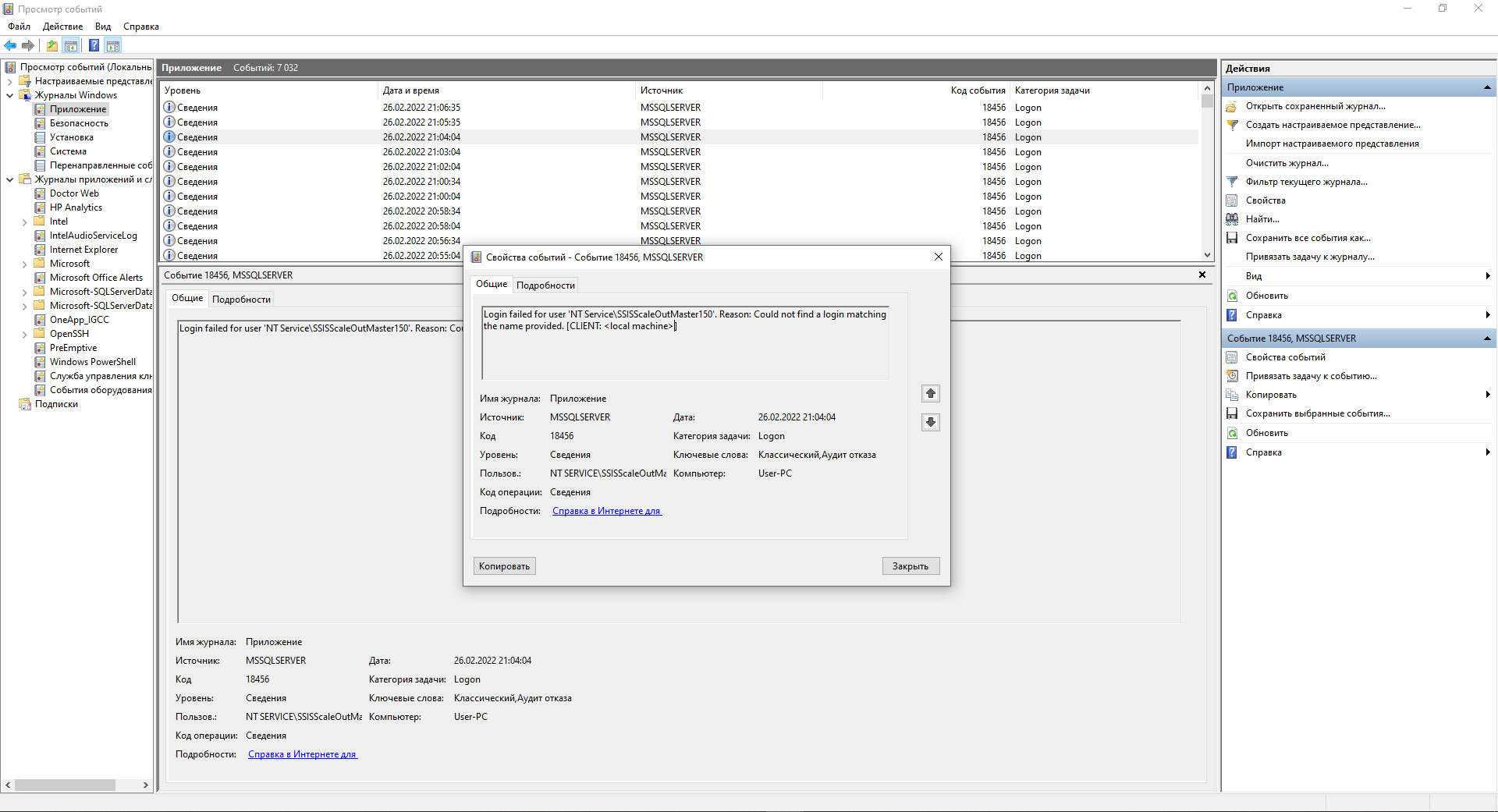
1. Win + R - **perfmon.msc** – системный монитор Windows; имеет несколько режимов отображения и позволяет выводить показания счетчиков производительности в режиме реального времени, а также сохранять данные в лог-файлы для последующего изучения:



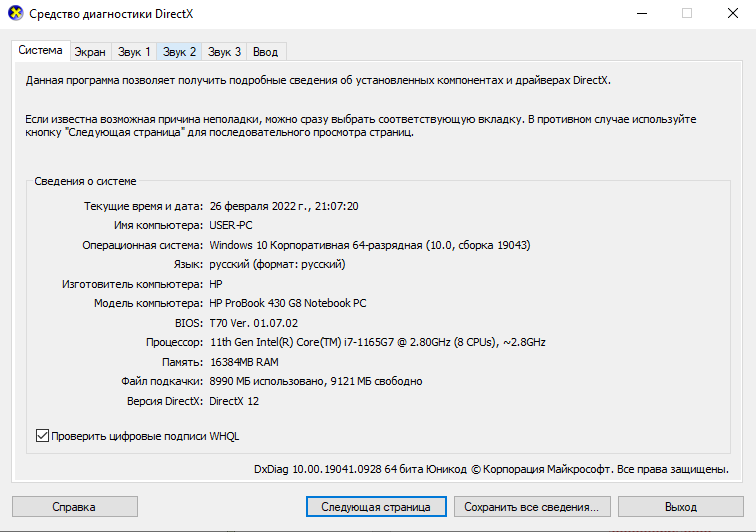


1. Win + R – **eventvwr.msc** – просмотр событий (Event Viewer). При выборе какого-либо события, в нижней части будет отображаться информация о нем. Эта информация может помочь найти решение проблемы в Интернете и стоит понимать, какое свойство что означает:

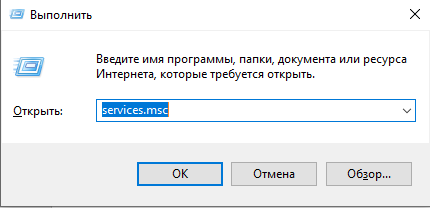
*Имя журнала* – имя файла журнала, куда была сохранена информация о событии. *Источник* – название программы, процесса или компонента системы, которое сгенерировало событие (если вы видите здесь Application Error), то имя самого приложение вы можете увидеть в поле выше. *Код* – код события, может помочь найти информацию о нем в Интернете. Правда, искать стоит в англоязычном сегменте по запросу Event ID + цифровое обозначение кода + название приложения, вызывавшего сбой (поскольку коды событий для каждой программы уникальны). *Код операции* – как правило, здесь всегда указано «Сведения», так что толку от этого поля мало. *Пользователь и компьютер* – сообщает о том, от имени какого пользователя и на каком компьютере был запущен процесс, вызвавший событие.

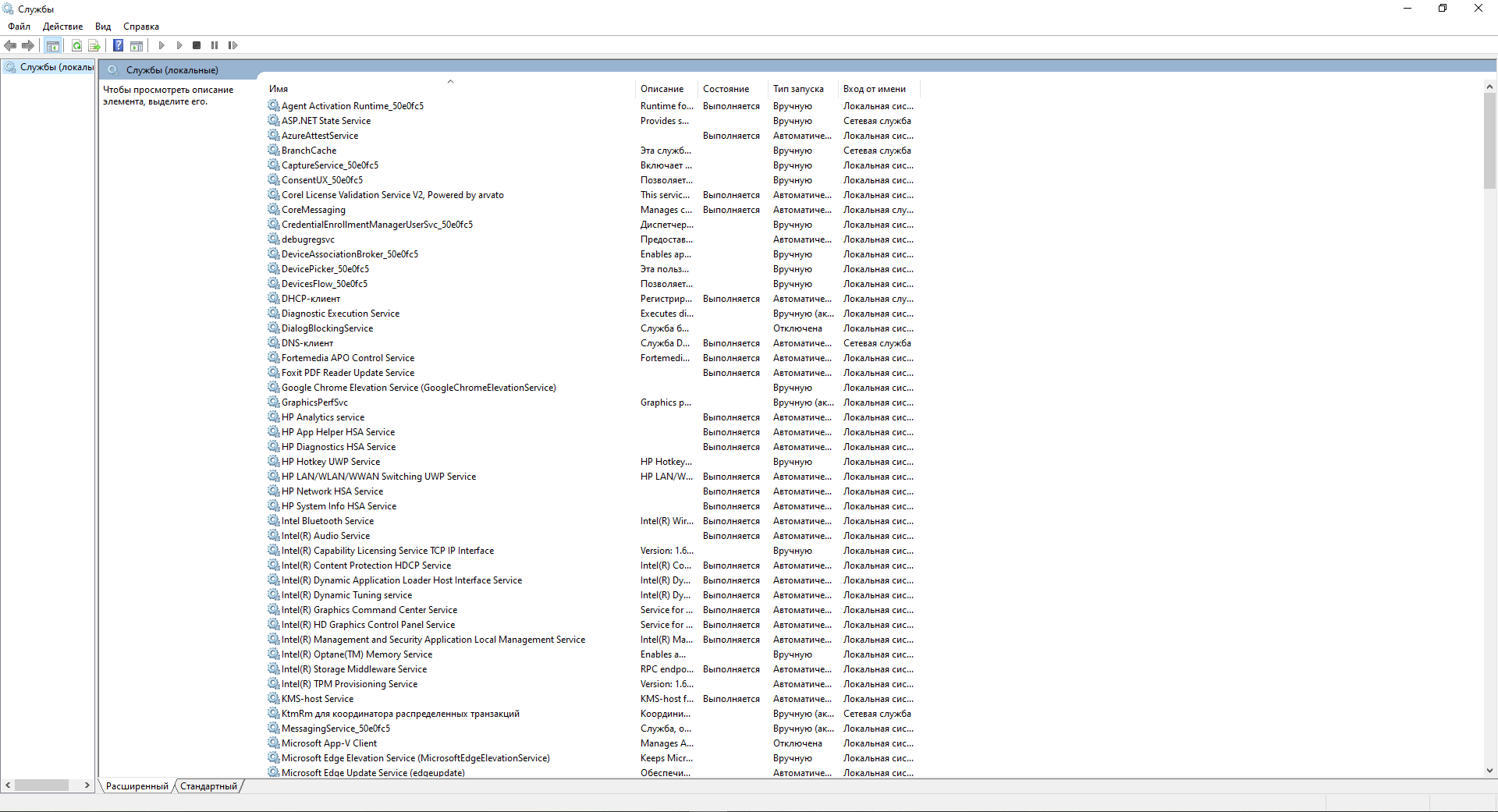


1. Средство диагностики Win + R – **dxdiag**: Это средство используется для сбора информации об устройствах для устранения неполадок со звуком и видео DirectX. Такую информацию может попросить сотрудник службы поддержки, или вы можете опубликовать ее на форуме для получения помощи.



1. Диспетчер системных служб Win + R – **services.msc**:





**Службы Windows** – это специальные программы, которые запускаются в фоновом (скрытом) режиме, отвечающие за корректное выполнение других программ, и, в частности, функций Windows.

**Вывод:** Я овладел навыками настройки и использования Брандмауэра Windows. Создал правила для входящих и исходящих подключений на разрешение и блокировку подключения.

**Практическое задание №4.1**

**Тема «Криптографическая защита информации»**

**Цель**

Овладеть основными криптографическими алгоритмами симметричного шифрования.

**Ход работы**

Симметричное шифрование – это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для восстановления информации.

Зашифровать сообщение с использованием шифра Цезаря, Трисемуса, Плейфейра и Вижинера и полученного секретного ключа (по номеру варианта и ключевому слову «Защита»). В качестве сообщения использовать свою Фамилию Имя Отчество.

1. Зашифровать сообщение «Кореневский Кирилл Русланович» шифром Цезаря с ключом 10.

using System;

public class CaesarCipher

{

//символы русской азбуки

const string alfabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

private string CodeEncode(string text, int k)

{

var fullAlfabet = alfabet;

var letterQty = fullAlfabet.Length;

var retVal = "";

for (int i = 0; i < text.Length; i++)

{

var c = text[i];

var index = fullAlfabet.IndexOf(c);

if (index < 0)

{

//если символ не найден, то добавляем его в неизменном виде

retVal += c.ToString();

}

else

{

var codeIndex = (letterQty + index + k) % letterQty;

retVal += fullAlfabet[codeIndex];

}

}

return retVal;

}

//шифрование текста

public string Encrypt(string plainMessage, int key)

=> CodeEncode(plainMessage, key);

//дешифрование текста

public string Decrypt(string encryptedMessage, int key)

=> CodeEncode(encryptedMessage, -key);

}

class Program

{

static void Main(string[] args)

{

var cipher = new CaesarCipher();

Console.Write("Введите текст: ");

var message = Console.ReadLine();

Console.Write("Введите ключ: ");

var secretKey = Convert.ToInt32(Console.ReadLine());

var encryptedText = cipher.Encrypt(message, secretKey);

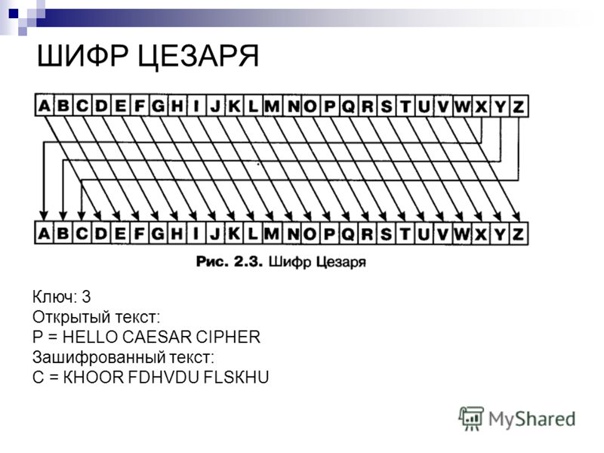
Console.WriteLine("Зашифрованное сообщение: {0}", encryptedText);

Console.WriteLine("Расшифрованное сообщение: {0}", cipher.Decrypt(encryptedText, secretKey));

Console.ReadLine();

}

Ответ: СХЧМФМЙШСПР СПЧПТТ ЧЪШТЗФХЙПЮ



1. Зашифровать сообщение «Кореневский Кирилл Русланович» шифром Цезаря с ключевым словом «Защита» и ключом 10.

using System;

public class CaesarCipher

{

//символы русской азбуки

const string alfabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

const string alfabet2 = "ВГДЕЖЙЗАЩИТБКЛМНОПРСУФХЦЧШЬЫЪЭЮЯ";

public string CodeEncode(string text)

{

var retVal = "";

for (int i = 0; i < text.Length; i++)

{

var c = text[i];

var index = alfabet.IndexOf(c);

if (index < 0)

{

//если символ не найден, то добавляем его в неизменном виде

retVal += c.ToString();

}

else

{

var codeIndex = index;

retVal += alfabet2[codeIndex];

}

}

return retVal;

}

}

class Program

{

static void Main(string[] args)

{

var cipher = new CaesarCipher();

Console.Write("Введите текст: ");

var message = Console.ReadLine();

var encryptedText = cipher.CodeEncode(message);

Console.WriteLine("Зашифрованное сообщение: {0}", encryptedText);

Console.ReadLine();

}

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| код | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| исх.текст | *А* | *Б* | *В* | *Г* | *Д* | *Е* | *Ж* | *З* | *И* | *Й* | *К* | *Л* | *М* | *Н* | *О* | *П* |
| шифр | В | Г | Д | Е | Ж | Й | К | Л | М | Н | **З** | **А** | **Щ** | **И** | **Т** | **Б** |
| код | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| исх.текст | *Р* | *С* | *Т* | *У* | *Ф* | *Х* | *Ц* | *Ч* | *Ш* | *Щ* | *Ъ* | *Ы* | *Ь* | *Э* | *Ю* | *Я* |
| шифр | О | П | Р | С | У | Ф | Х | Ц | Ч | Ш | Ъ | Ы | Ь | Э | Ю | Я |

Ответ: ЗТОЙИЙДПЗМН ЗМОМАА ОСПАВИТДМЦ

1. Зашифровать сообщение «КОРЕНЕВСКИЙ КИРИЛЛ РУСЛАНОВИЧ» шифром Трисемуса с ключевым словом «Защита» – таблица Трисемуса заполняется с помощью ключевого слова, повторяющиеся буквы которого отбрасываются. Затем таблица дополняется не вошедшими в нее буквами алфавита по порядку как в системе Цезаря с ключевым словом. При шифровании буква открытого текста заменяется буквой, расположенной ниже нее в том же столбце.

using System;

class TrisemusCipher

{

static char[,] CreateTrisemusMatrix()

{

char[,] matrix = new char[4, 8]

{

{ 'З', 'А', 'Щ', 'И', 'Т', 'Б', 'В', 'Г' },

{ 'Д', 'Е', 'Ж', 'Й', 'К', 'Л', 'М', 'Н' },

{ 'О', 'П', 'Р', 'С', 'У', 'Ф', 'Х', 'Ц' },

{ 'Ч', 'Ш', 'Ь', 'Ы', 'Ъ', 'Э', 'Ю', 'Я' }

};

return matrix;

}

static string Encrypt(string plaintext, char[,] matrix)

{

string encryptedText = "";

foreach (char c in plaintext)

{

if (char.IsLetter(c))

{

char upperChar = char.ToUpper(c);

bool found = false;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 8; j++)

{

if (matrix[i, j] == upperChar)

{

int nextRow = (i + 1) % 4;

encryptedText += matrix[nextRow, j];

found = true;

break;

}

}

if (found)

break;

}

}

else

{

encryptedText += c;

}

}

return encryptedText;

}

static void Main()

{

string plaintext = "КОРЕНЕВСКИЙ КИРИЛЛ РУСЛАНОВИЧ";

char[,] matrix = CreateTrisemusMatrix();

string encryptedText = Encrypt(plaintext, matrix);

Console.WriteLine(encryptedText);

}

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| З | А | Щ | И | Т | Б | В | Г |
| Д | Е | Ж | Й | К | Л | М | Н |
| О | П | Р | С | У | Ф | Х | Ц |
| Ч | Ш | Ь | Ы | Ъ | Э | Ю | Я |

Ответ: УЧЬПЦПМЫУЙС УЙЬЙФФ ЬЪЫФЕЦЧМЙЗ

1. Зашифровать сообщение «КОРЕНЕВСКИЙ КИРИЛЛ РУСЛАНОВИЧ» шифром Плейфейра с ключевым словом «Защита»

using System;

using System.Linq;

using System.Text;

public class PlayfairCipher

{

private const int RowSize = 8;

private const int ColSize = 4;

private char[,] table;

public PlayfairCipher(string keyword)

{

this.table = this.GenerateTable(keyword);

}

private char[,] GenerateTable(string keyword)

{

char[,] table = new char[RowSize, ColSize];

string alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

string key = string.Join("", (keyword + alphabet).Distinct());

for (int i = 0; i < RowSize; i++)

for (int j = 0; j < ColSize; j++)

table[i, j] = key[i \* ColSize + j];

return table;

}

public string Encrypt(string openText)

{

StringBuilder cipherText = new StringBuilder();

for (int i = 0; i < openText.Length; i += 2)

{

int row1, col1, row2, col2;

this.FindPosition(openText[i], out row1, out col1);

this.FindPosition(openText[i + 1], out row2, out col2);

if (row1 == row2)

{

cipherText.Append(this.table[row1, (col1 + 1) % ColSize]);

cipherText.Append(this.table[row2, (col2 + 1) % ColSize]);

}

else if (col1 == col2)

{

cipherText.Append(this.table[(row1 + 1) % RowSize, col1]);

cipherText.Append(this.table[(row2 + 1) % RowSize, col2]);

}

else

{

cipherText.Append(this.table[row1, col2]);

cipherText.Append(this.table[row2, col1]);

}

}

return cipherText.ToString();

}

private void FindPosition(char c, out int row, out int col)

{

for (int i = 0; i < RowSize; i++)

for (int j = 0; j < ColSize; j++)

if (this.table[i, j] == c)

{

row = i;

col = j;

return;

}

row = col = -1;

}

}

public class Program

{

static void Main(string[] args)

{

var cipher = new PlayfairCipher("ЗАЩИТА");

string openText = " КОРЕНЕВСКИЙКИРИЛЛРУСЛАНОВИЧЪ";

string cipherText = cipher.Encrypt(openText);

Console.WriteLine(cipherText);

}

}

Ответ:

КО РЕ НЕ ВС КИ ЙК ИР ИЛ ЛР УС ЛА НО ВИ ЧЪ

ОУ ПЖ ЛЙ ГР НЗ ДН ЩС АН МП ЦО ПБ КС ГЩ ШЫ

|  |  |  |  |
| --- | --- | --- | --- |
| **З** | **А** | **Щ** | **И** |
| **Т** | Б | В | Г |
| Д | Е | Ж | Й |
| К | Л | М | Н |
| О | П | Р | С |
| У | Ф | Х | Ц |
| Ч | Ш | Ъ | Ы |
| Ь | Э | Ю | Я |

Шифрование происходит по следующим правилам:

1. Если обе буквы биграммы исходного текста не лежат в одной строке или в одном столбце, тогда находят буквы в углах прямоугольника, определяемого данной парой букв. Первой буквой биграммы шифртекста становится буква, расположенная в той же строке, что и первая буква исходной биграммы, и в том же столбце, что и вторая буква открытого текста. Вторая буква биграммы шифртекста находится на пересечении строки, содержащей вторую букву, и столбца, содержащего первую букву открытого текста.
2. Если обе буквы биграммы открытого текста принадлежат одной строке таблицы, то первой и второй буквами биграммы шифртекста считаются буквы, лежащие справа, соответственно, от первой и второй букв биграммы открытого текста. При этом считается, что таблица циклически замкнута по строкам, то есть конец любой строки связан с ее началом. Поэтому если буквы биграммы расположены в одной строке и одна из них находится в последнем столбце таблицы, то для шифртекста берется буква из первого столбца этой строки.
3. Если обе буквы биграммы открытого текста принадлежат одному столбцу таблицы, то первой и второй буквами биграммы шифртекста считаются буквы, лежащие, соответственно, под первой и под второй буквами биграммы открытого текста. При этом считается, что таблица циклически замкнута по столбцам, то есть конец любого столбца замыкается на его начале. Поэтому если буквы биграммы расположены в одном столбце и одна из них находится в последней строке таблицы, то для шифртекста берется буква из первой строки этого столбца.
4. Зашифровать сообщение «КОРЕНЕВСКИЙ КИРИЛЛ РУСЛАНОВИЧ» шифром Вижинера с ключевым словом «Защита»

using System;

using System.Text;

public class VigenereCipher

{

private const string Alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

private string keyword;

public VigenereCipher(string keyword)

{

this.keyword = keyword.ToUpper();

}

public string Encrypt(string input)

{

StringBuilder output = new StringBuilder();

input = input.Replace(" ", "").ToUpper();

for (int i = 0; i < input.Length; i++)

{

int charPosition = Alphabet.IndexOf(input[i]);

int keywordPosition = Alphabet.IndexOf(keyword[i % keyword.Length]);

int newPosition = (charPosition + keywordPosition) % Alphabet.Length;

output.Append(Alphabet[newPosition]);

}

return output.ToString();

}

}

public class Program

{

public static void Main()

{

var cipher = new VigenereCipher("Защита");

string message = "КОРЕНЕВСКИЙ КИРИЛЛ РУСЛАНОВИЧ";

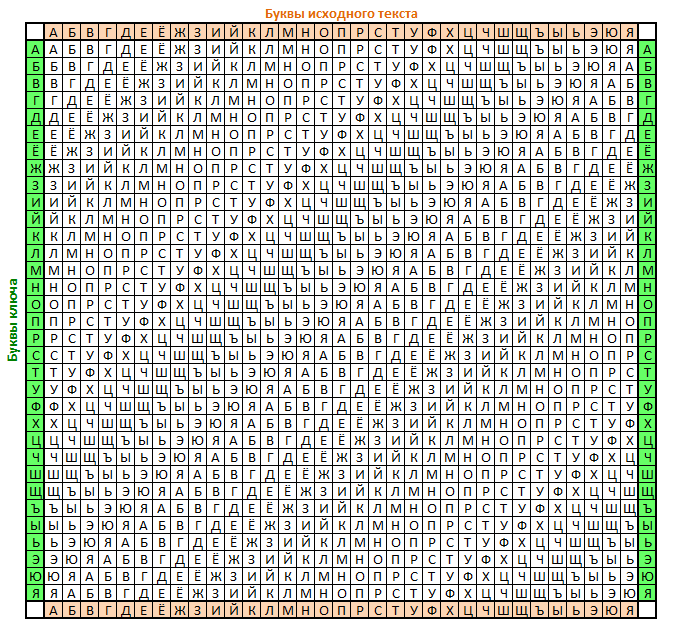
string cipherText = cipher.Encrypt(message);

Console.WriteLine(cipherText);

}

}

Ответ: СОЙНЯЕЙСГРЫ КПРБУЭ РЪСДИЯОЙИР



|  |  |  |
| --- | --- | --- |
| 10. | Боитдиултоьтдгь-псеоснояшмяил\_ьу-бу\_дччуч\_ | Расшифровать с помощью метода простой перестановки. Таблица 6х7 |

using System;

class SimpleTranspositionDecryptor

{

public static string Decrypt(string ciphertext, int numRows, int numCols)

{

int totalChars = numRows \* numCols;

// Добавляем пробелы, чтобы заполнить таблицу до полного размера

while (ciphertext.Length < totalChars)

{

ciphertext += " ";

}

char[,] matrix = new char[numRows, numCols];

int index = 0;

// Заполняем таблицу символами из зашифрованного текста

for (int col = 0; col < numCols; col++)

{

for (int row = 0; row < numRows; row++)

{

matrix[row, col] = ciphertext[index];

index++;

}

}

string plaintext = "";

// Считываем символы по строкам для получения расшифрованного текста

for (int row = 0; row < numRows; row++)

{

for (int col = 0; col < numCols; col++)

{

plaintext += matrix[row, col];

}

}

return plaintext.TrimEnd();

}

}

class Program

{

static void Main()

{

string ciphertext = "Боитдиултоьтдгь-псеоснояшмяил\_ьу-бу\_дччуч\_";

int numRows = 6;

int numCols = 7;

string plaintext = SimpleTranspositionDecryptor.Decrypt(ciphertext, numRows, numCols);

Console.WriteLine("Расшифрованный текст: " + plaintext);

}

}

Ответ: Будешьдолгомучиться-что-нибудьполучится\_\_\_

Вывод: я овладел основными криптографическими алгоритмами симметричного шифрования.

**Практическое задание №4.2**

**Тема «Криптографическая защита информации»**

**Цель**

Овладение основными криптографическими алгоритмами асимметричного шифрования.

**Ход работы**

**Алгоритм Диффи-Хеллмана**

Обмен ключами с помощью протокола Диффи-Хеллмана (DH) – это метод безопасного обмена криптографическими ключами по общедоступному каналу. Это один из первых протоколов с открытым ключом, который изначально был концептуализирован Ральфом Мерклем и назван в честь Уитфилда Диффи и Мартина Хеллмана. DH является одним из первых практических примеров обмена открытыми ключами. Сегодня DH используется для многих приложений, таких как, например, Proton Mail, SSH, GPG и так далее.

Диффи-Хеллман работает по принципу неполного обмена ключом шифрования по сети. У каждой стороны есть открытый ключ (который может видеть каждый, включая предполагаемого хакера) и закрытый ключ (его может видеть только пользователь компьютера). У первого пользователя нет доступа к личному ключу второго. У второго – к первому. Положим, что оба используем общий алгоритм шифрования, и им нужно каким-то образом восстановить общий ключ шифрования, не передавая слишком много информации через сеть.

На практике обмен ключами по алгоритму Диффи‑Хеллмана происходит по следующей схеме.

1. Два участника обмена договариваются о двух числах. **Один выбирает большое простое число, а другой – целое число, меньшее числа первого участника**. Переговоры они могут вести открыто, и это никак не отразится на безопасности.
2. **Каждый из двух участников, независимо друг от друга, генерирует другое число, которое они будут хранить в тайне**. Эти числа выполняют роль секретного ключа. Далее в вычислениях используются секретный ключ и два предыдущих целых числа. Результат вычислений посылается участнику обмена, и он играет роль открытого ключа.
3. **Участники обмена обмениваются открытыми ключами.** Далее они, **используя собственный секретный ключ и открытый ключ партнера, конфиденциально вычисляют ключ сессии**. Каждый партер вычисляет один и тот же ключ сессии.
4. **Ключ сессии может использоваться как секретный ключ для другого алгоритма шифрования, например, DES**. Никакое третье лицо, контролирующее обмен, не сможет вычислить ключ сессии, не зная один из секретных ключей.

using System;

using System.Numerics;

public class DiffieHellman

{

// Приватный ключ

private BigInteger privateKey;

// Публичный ключ

private BigInteger publicKey;

// Общий ключ, который генерируется каждым пользователем на основе их приватного ключа и публичного ключа другого пользователя

private BigInteger sharedKey;

public DiffieHellman(BigInteger generator, BigInteger prime)

{

Random random = new Random();

// Генерация приватного ключа как случайного числа

privateKey = new BigInteger(random.Next());

publicKey = BigInteger.ModPow(generator, privateKey, prime);

}

public BigInteger PublicKey

{

get { return publicKey; }

}

// Генерация общего ключа на основе публичного ключа другого пользователя, своего приватного ключа и простого числа

public void GenerateSharedKey(BigInteger otherPublicKey, BigInteger prime)

{

sharedKey = BigInteger.ModPow(otherPublicKey, privateKey, prime);

}

public BigInteger SharedKey

{

get { return sharedKey; }

}

}

public class Program

{

public static void Main()

{

BigInteger g = 5;

BigInteger p = 23;

var alice = new DiffieHellman(g, p);

var bob = new DiffieHellman(g, p);

// Генерация общего ключа для каждого пользователя на основе публичного ключа другого пользователя

alice.GenerateSharedKey(bob.PublicKey, p);

bob.GenerateSharedKey(alice.PublicKey, p);

// Вывод на консоль общих ключей, сгенерированных каждым пользователем

Console.WriteLine("Shared key Alice: " + alice.SharedKey);

Console.WriteLine("Shared key Bob: " + bob.SharedKey);

}

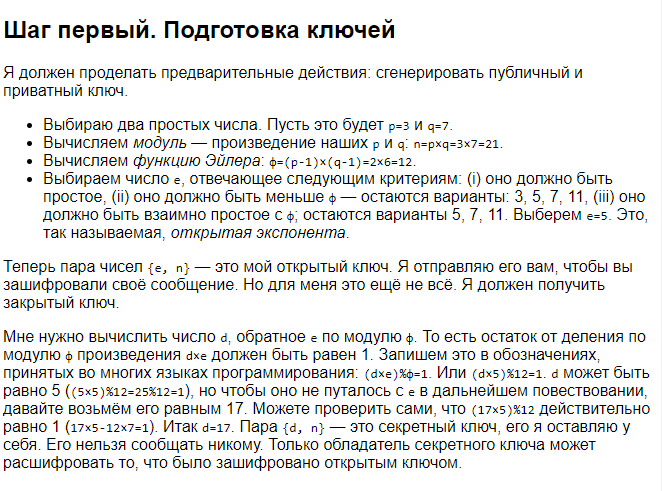
}

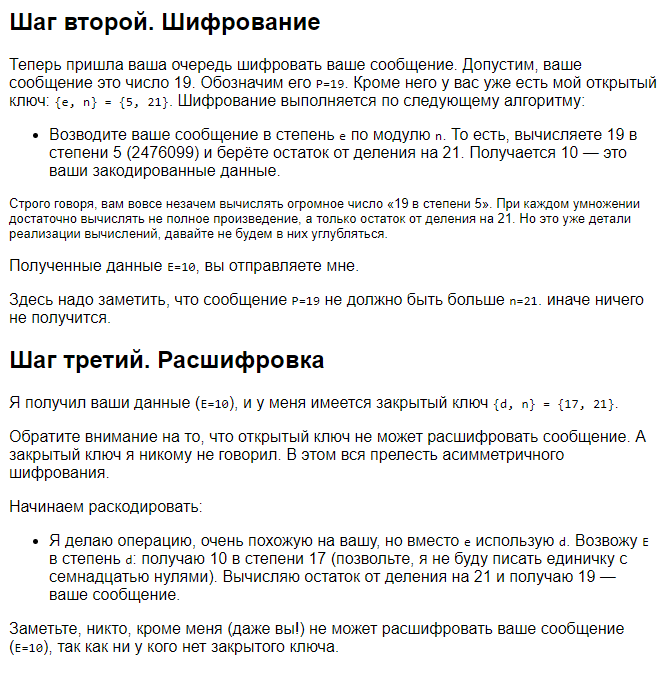
**RSA-шифрование**

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи **факторизации больших целых чисел.** Его криптостойкость и следующая отсюда всюду применимость в современных системах основывается на сложности разложения на множители больших чисел, а именно - на исключительной трудности задачи определить секретный ключ на основании открытого, так как для этого потребуется решить задачу о существовании делителей целого числа.

Весь алгоритм приведен в таблице:

|  |  |  |
| --- | --- | --- |
| Этап | Описание операции | Результат операции |
| Генерация ключей | Выбрать два простых различных числа | p=3557,  q=2579 |
| Вычислить модуль (произведение) | n = p \cdot q = 3557 \cdot 2579 = 9173503 |
| Вычислить функцию Эйлера | \varphi(n) = (p-1) (q-1) = 9167368 |
| Выбрать открытую экспоненту | e = 3 |
| Вычислить секретную экспоненту | d = e^{-1} \mod \varphi(n)  d = 6111579 |
| Опубликовать открытый ключ | \{e, n\} = \{3,9173503 \} |
| Сохранить закрытый ключ | \{d, n\} = \{6111579, 9173503 \} |
| Шифрование | Выбрать текст для зашифровки | m = 111111 |
| Вычислить шифротекст | \begin{align} c &= E(m) \\  &= m^e \mod n \\  &= 111111^3   \mod 9173503 \\  &= 4051753 \end{align} |
| Расшифрование | Вычислить исходное сообщение | \begin{align} m &= D(c) = \\   &= c^d \mod n \\   &= 4051753^{6111579} \mod 9173503 \\   &= 111111 \end{align} |





using System;

using System.Numerics;

using System.Security.Cryptography;

public class RSACryptography

{

public static void Main()

{

// Генерация ключей

using RSA rsa = RSA.Create();

RSAParameters publicKey = rsa.ExportParameters(includePrivateParameters: false);

RSAParameters privateKey = rsa.ExportParameters(includePrivateParameters: true);

BigInteger originalNumber = BigInteger.Parse("12345678901234567890");

BigInteger encryptedNumber = Encrypt(originalNumber, publicKey);

// Расшифровка числа

BigInteger decryptedNumber = Decrypt(encryptedNumber, privateKey);

// Вывод результатов

Console.WriteLine("Original number: " + originalNumber);

Console.WriteLine("Encrypted number: " + encryptedNumber);

Console.WriteLine("Decrypted number: " + decryptedNumber);

}

public static BigInteger Encrypt(BigInteger number, RSAParameters publicKey)

{

using RSA rsa = RSA.Create();

rsa.ImportParameters(publicKey);

byte[] numberBytes = number.ToByteArray(); // Преобразование числа в массив байтов

byte[] encryptedBytes = rsa.Encrypt(numberBytes, RSAEncryptionPadding.OaepSHA256); // Шифрование массива байтов

BigInteger encryptedNumber = new BigInteger(encryptedBytes); // Преобразование зашифрованных байтов в число

return encryptedNumber;

}

public static BigInteger Decrypt(BigInteger encryptedNumber, RSAParameters privateKey)

{

using RSA rsa = RSA.Create();

rsa.ImportParameters(privateKey);

byte[] encryptedBytes = encryptedNumber.ToByteArray(); // Преобразование числа в массив байтов

byte[] decryptedBytes = rsa.Decrypt(encryptedBytes, RSAEncryptionPadding.OaepSHA256); // Расшифровка массива байтов

BigInteger decryptedNumber = new BigInteger(decryptedBytes); // Преобразование расшифрованных байтов в число

return decryptedNumber;

}

}

**Шифрование Эль-Гамаля**

Схема Эль-Гамаля – криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Фактически здесь используется схема Диффи-Хеллмана, чтобы сформировать общий секретный ключ для двух абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ. Для каждого следующего сообщения секретный ключ вычисляется заново.

## **Генерация ключей**

1. Генерируется случайное простое число ~p длины ~n [битов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82).
2. Выбирается случайный примитивный элемент ~g.
3. Выбирается случайное целое число ~x такое, что ~1 < x < p-1.
4. Вычисляется ~y = g^x\,\bmod\,p.
5. Открытым ключом является тройка \left( p,g,y \right), закрытым ключом — число ~x.

**Шифрование**

1. Сообщение ~M шифруется следующим образом:
2. Выбирается сессионный [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) — случайное целое число ~k такое, что ~1 < k < p - 1
3. Вычисляются числа a = g^k\,\bmod\,p и b = y^k M\,\bmod\,p.
4. Пара чисел \left( a, b \right) является [шифротекстом](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82" \o "Шифротекст).
5. Нетрудно видеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

**Расшифрование**

1. Зная закрытый ключ ~x, исходное сообщение можно вычислить из шифротекста \left( a, b \right) по формуле:

M = b(a^x)^{-1}\,\bmod\,p.

1. При этом нетрудно проверить, что

~(a^x)^{-1}\equiv g^{-kx}\pmod{p}

1. и поэтому

~b(a^x)^{-1}\equiv (y^kM)g^{-xk}\equiv (g^{xk}M) g^{-xk}\equiv M \pmod{p}.

1. Для практических вычислений больше подходит следующая формула:

M = b(a^x)^{-1}\,\bmod\,p = b \cdot a^{(p-1-x)}\,\bmod\,p 

**Вывод**

**Асимметричное шифрование**– **это метод шифрования данных, предполагающий использование двух ключей – открытого и закрытого**. Открытый (публичный) ключ применяется для шифрования информации и **может передаваться по незащищенным каналам**. Закрытый (приватный) ключ применяется для расшифровки данных, зашифрованных открытым ключом. **Открытый и закрытый ключи – это очень большие числа, связанные друг с другом определенной функцией,** но так, что, зная одно, крайне сложно вычислить второе.

Асимметричное шифрование используется для защиты информации при ее передаче, также на его принципах построена работа электронных подписей.

Схема передачи данных между двумя субъектами (А и Б) с использованием открытого ключа выглядит следующим образом:

* Субъект А генерирует пару ключей, открытый и закрытый (публичный и приватный).
* Субъект А передает открытый ключ субъекту Б. Передача может осуществляться по незащищенным каналам.
* Субъект Б шифрует пакет данных при помощи полученного открытого ключа и передает его А. Передача может осуществляться по незащищенным каналам.
* Субъект А расшифровывает полученную от Б информацию при помощи секретного, закрытого ключа.

В такой схеме перехват любых данных, передаваемых по незащищенным каналам, не имеет смысла, поскольку восстановить исходную информацию возможно только при помощи закрытого ключа, известного лишь получателю и не требующего передачи.

using System;

using System.Numerics;

// Реализация алгоритма RSA

public class RSA

{

// Определение приватных и публичных ключей

private BigInteger e; // Открытая экспонента

private BigInteger n; // Модуль для публичного и приватного ключа

private BigInteger d; // Секретная экспонента

public RSA(BigInteger p, BigInteger q)

{

// Вычисление модуля (произведение)

n = p \* q;

// Вычисление функции Эйлера

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

// Выбор открытой экспоненты

e = 3;

// Вычисление секретной экспоненты

d = BigInteger.ModPow(e, 1, phi);

}

// Шифрование

public BigInteger Encrypt(BigInteger message)

{

return BigInteger.ModPow(message, e, n);

}

// Расшифрование

public BigInteger Decrypt(BigInteger cipher)

{

return BigInteger.ModPow(cipher, d, n);

}

}

// Реализация алгоритма Эль-Гамаля

public class ElGamal

{

// Определение приватного и публичного ключей

private BigInteger g; // Примитивный элемент

private BigInteger p; // Простое число

private BigInteger x; // Приватный ключ

private BigInteger h; // Часть публичного ключа

public ElGamal(BigInteger p, BigInteger g, BigInteger x)

{

this.p = p;

this.g = g;

this.x = x;

// Вычисление h

h = BigInteger.ModPow(g, x, p);

}

// Шифрование

public (BigInteger y, BigInteger s) Encrypt(BigInteger message)

{

// Выбор сессионного ключа

BigInteger k = 2; // Пример

// Вычисление чисел y и s

BigInteger y = BigInteger.ModPow(g, k, p);

BigInteger s = BigInteger.ModPow(h, k, p) \* message % p;

return (y, s);

}

// Расшифрование

public BigInteger Decrypt(BigInteger y, BigInteger s)

{

// Вычисление исходного сообщения

BigInteger message = BigInteger.ModPow(y, p - 1 - x, p) \* s % p;

return message;

}

}

class Program

{

static void Main(string[] args)

{

// RSA

Console.WriteLine("RSA:");

var rsa = new RSA(61, 53);

BigInteger rsaMessage = 123;

Console.WriteLine($"Original message: {rsaMessage}");

var rsaCipher = rsa.Encrypt(rsaMessage);

Console.WriteLine($"Encrypted message: {rsaCipher}");

var rsaDecipher = rsa.Decrypt(rsaCipher);

Console.WriteLine($"Decrypted message: {rsaDecipher}");

// ElGamal

Console.WriteLine("\nElGamal:");

var elGamal = new ElGamal(23, 5, 6);

BigInteger elGamalMessage = 15;

Console.WriteLine($"Original message: {elGamalMessage}");

var (y, s) = elGamal.Encrypt(elGamalMessage);

Console.WriteLine($"Encrypted message: y={y}, s={s}");

var elGamalDecipher = elGamal.Decrypt(y, s);

Console.WriteLine($"Decrypted message: {elGamalDecipher}");

}

}

**Источники**

1. Методическое пособие к практическому занятию №6 по дисциплине «Основы защиты информации»

**Практическое задание № 5**

**Тема «Криптографическая защита информации»**

**Цель**: изучить и закрепить умение реализации ЭЦП на примере RSA.

**Теоретические сведения**

Реализация элементов ЭЦП RSA

Протоколы ЭЦП с одной стороны относят к протоколам аутентификации, т.к. гарантируют, что сообщение поступило от достоверного отправителя, а с другой стороны к протоколам контроля целостности, т.к. гарантируют, что сообщение пришло в неискаженном виде. Более того, получатель в дальнейшем может использовать ЭЦП как доказательство достоверности сообщения третьим лицам (арбитру) в том случае, если отправитель впоследствии попытается отказаться от него.

Говоря о схеме цифровой подписи, обычно имеют в виду следующую классическую ситуацию:

• отправитель знает содержание сообщения, которое он подписывает;

• получатель, зная открытый ключ проверки подписи, может проверить правильность подписи полученного сообщения в любое время без какого-либо разрешения и участия отправителя;

• безопасность схемы подписи гарантируется.

Электронная цифровая подпись – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе (Федеральный закон "Об электронной цифровой подписи").

При создании цифровой подписи по классической схеме отправитель:

• применяет к исходному сообщению T хеш-функцию h(T) и получает хеш-образ r сообщения;

• вычисляет цифровую подпись s по хеш-образу r с использованием своего закрытого ключа;

• посылает сообщение T вместе с цифровой подписью s получателю.

Получатель, отделив цифровую подпись от сообщения, выполняет следующие действия:

• применяет к полученному сообщению T хеш-функцию h(T) и получает хеш-образ r сообщения;

• расшифровывает хеш-образ r’ из цифровой подписи s с использованием открытого ключа отправителя;

• проверяет соответствие хеш-образов r и r’ и если они совпадают, то отправитель действительно является тем, за кого себя выдает, и сообщение при передаче не подверглось искажению.

Как видно из этой схемы, порядок использования ключей обратный тому, который используется при передаче секретных сообщений. Вначале отправитель использует свой закрытый ключ, а затем получатель применяет открытый ключ отправителя.

Разновидности ЭЦП

Кроме классической схемы ЭЦП различают еще несколько специальных:

• схема "конфиденциальной" (неотвергаемой) подписи – подпись не может быть проверена без участия сгенерировавшего ее лица;

• схема подписи "вслепую" ("затемненной" подписи) - отправитель не знает подписанного им сообщения;

• схема "мультиподписи" - вместо одного отправителя сообщение подписывает группа из нескольких участников;

• схема "групповой" подписи - получатель может проверить, что подписанное сообщение пришло от члена некоторой группы отправителей, но не знает, кем именно из членов группы оно подписано. В тоже время, в случае необходимости, отправитель может быть определен;

• и др.

**Задание для выполнения**

1) Объясните последовательность выполнения процедур генерации и проверки ЭЦП.

2) Опишите последовательность действий участников протокола при отправке и проверке ЭЦП.

3) Опишите схему протокола ЭЦП на основе алгоритма RSA.

4) \*На базе алгоритма RSA получить ЭЦП (в проекте можно использовать существующие криптографические алгоритмы). Удостовериться, что ЭЦП принадлежит именно этому сообщению.

**Реализация**

using System;

using System.Security.Cryptography;

using System.Text;

class Program

{

static void Main()

{

string message = Console.ReadLine();

byte[] data = Encoding.UTF8.GetBytes(message);

using (RSA rsa = RSA.Create())

{

try

{

RSAParameters privateKey = rsa.ExportParameters(true);

RSAParameters publicKey = rsa.ExportParameters(false);

byte[] signature = SignData(data, privateKey);

bool isVerified = VerifyData(data, signature, publicKey);

Console.WriteLine("Исходное сообщение: " + message);

Console.WriteLine("Подпись: " + Convert.ToBase64String(signature));

Console.WriteLine("Подпись верна: " + isVerified);

}

catch (CryptographicException e)

{

Console.WriteLine(e.Message);

}

}

}

static byte[] SignData(byte[] data, RSAParameters privateKey)

{

using (RSA rsa = RSA.Create())

{

rsa.ImportParameters(privateKey);

// Создание подписи

byte[] signature = rsa.SignData(data, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

return signature;

}

}

static bool VerifyData(byte[] data, byte[] signature, RSAParameters publicKey)

{

using (RSA rsa = RSA.Create())

{

rsa.ImportParameters(publicKey);

// Проверка подписи

bool isVerified = rsa.VerifyData(data, signature, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);

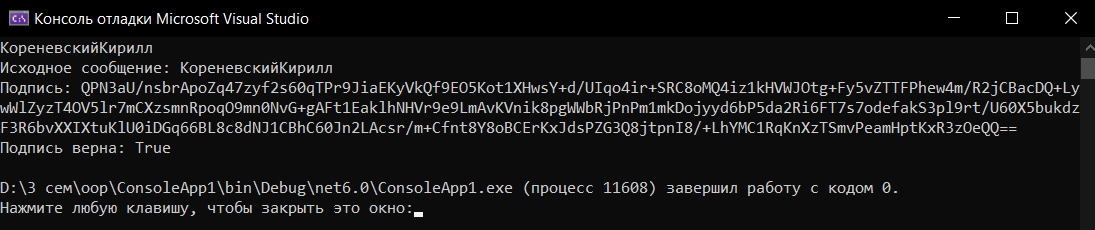
return isVerified;

}

}

}

**Результат**



Контрольные вопросы:

Дайте определение понятию "электронная цифровая подпись".

Электронная цифровая подпись – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.

Опишите последовательность действий участников протокола при отправке и проверке ЭЦП.

При создании цифровой подписи по классической схеме отправитель:

• применяет к исходному сообщению T хеш-функцию h(T) и получает хеш-образ r сообщения;

• вычисляет цифровую подпись s по хеш-образу r с использованием своего закрытого ключа;

• посылает сообщение T вместе с цифровой подписью s получателю.

Получатель, отделив цифровую подпись от сообщения, выполняет следующие действия:

• применяет к полученному сообщению T хеш-функцию h(T) и получает хеш-образ r сообщения;

• расшифровывает хеш-образ r’ из цифровой подписи s с использованием открытого ключа отправителя;

• проверяет соответствие хеш-образов r и r’ и если они совпадают, то отправитель действительно является тем, за кого себя выдает, и сообщение при передаче не подверглось искажению.

Опишите схему протокола ЭЦП на основе алгоритма RSA.



Перечислите специальные схемы ЭЦП.

Кроме классической схемы ЭЦП различают еще несколько специальных:

• схема "конфиденциальной" (неотвергаемой) подписи – подпись не может быть проверена без участия сгенерировавшего ее лица;

• схема подписи "вслепую" ("затемненной" подписи) - отправитель не знает подписанного им сообщения;

• схема "мультиподписи" - вместо одного отправителя сообщение подписывает группа из нескольких участников;

• схема "групповой" подписи - получатель может проверить, что подписанное сообщение пришло от члена некоторой группы отправителей, но не знает, кем именно из членов группы оно подписано.

Перечислите недостатки алгоритма цифровой подписи RSA:

Вычислительная сложность: Алгоритм RSA является вычислительно сложным и требует большого количества операций с большими числами. Это может быть проблематично при работе с большими объемами данных или на устройствах с ограниченными вычислительными ресурсами.

Размер ключа: Ключи RSA обычно имеют большую длину, чем другие алгоритмы подписи, такие как DSA или ECDSA. Больший размер ключа требует больше вычислительных ресурсов и времени для генерации ключей, выполнения операций подписи и проверки подписи.

Относительная медленность: Из-за своей вычислительной сложности алгоритм RSA может быть медленнее в сравнении с некоторыми другими алгоритмами подписи. Это может быть проблемой в случаях, где требуется высокая скорость обработки подписей, например, при обработке больших объемов транзакций.

Отсутствие нерепудиации: Алгоритм RSA не предоставляет встроенной поддержки для нерепудиации, то есть способности доказать, что отправитель не может отказаться от своей подписи. Для обеспечения нерепудиации может потребоваться использование дополнительных механизмов, таких как цепочки доверия или протоколы доказательства.

**Вывод**

В ходе практической работы я изучил и закрепил умение реализации ЭЦП на примере RSA**.**

**Практическое задание № 6**

**Тема «Теория чисел»**

**Цель**: получение основных сведений из курса теории чисел

**Теоретические часть**

Ниже рассматриваются: *N* – множество натуральных чисел, *Z* – множество рациональных чисел. Множество целых чисел *Z* – счетное, состоит из элементов 0; ±1; ±2; …; ± *n*,…. На нем определены две алгебраические операции – сложение и умножение. Эти операции обладают следующими свойствами (для любых ):

1. ассоциативность: ; ;

2. коммутативность: ; ;

3. существует нейтральный элемент – 0 и 1 соответственно:



4.  – закон дистрибутивности;

5. для каждого целого  существует единственное противоположное, то есть такое целое *b*, что *a* + *b* = *b* + *a* = 0.

*Теорема 2.1* (*О делении с остатком*). Для любых целых чисел *a* и *b*, , существует единственные целые числа *q* и  , такие, что .

В этом равенстве  называют остатком, а  – частным (неполным частным – при ) от деления *a*  на  При *r* = 0 величины *b* и *q* называют делителями или множителями числа *а*. Читатель со школьной скамьи умеет находить частное и остаток методом деления уголком.

*Следствие.* Пусть  – натуральное число,  Для всякого целого числа *a*  и максимального целого  с условием  существуют единственные целые  такие, что 

Такое равенство записывают сокращённо  или  (если *b* известно по контексту) и называют записью числа *a* в *b* – ичной позиционной системе счисления или системе счисления по основанию *b*. Нам кажется естественной привычная десятичная позиционная система записи целых чисел . В различных ситуациях более удобными оказываются другие основания. К примеру, во всех компьютерах на микроуровне вычисления проводятся в двоичной системе счисления. Для перехода к ней с десятичной применяют промежуточную – 16 - ричную систему счисления.

*Лемма 2.1.* Если в равенстве  все слагаемые – целые числа и все, кроме может быть одного, делятся на целое , то и это исключенное слагаемое делится на .

**Определение 2.1*.***Если целые числа  делятся на целое , то *d*  называют их *общим делителем*.

В дальнейшем речь идет только о положительных целых делителях.

**Определение 2.2.** Максимальный из общих делителей целых чисел  называется их *наибольшим общим делителем* и обозначается через НОД ().

*Теорема 2.2.* Если *,* то НОД *(a, b)*=НОД *(b, c).*

Теорема 2.2 позволила Евклиду (примерно 2300 лет тому назад) обосновать следующий факт.

*Теорема 2.3.* Наибольший общий делитель целых чисел  *a* и *b*   равен последнему отличному от нуля остатку цепочки равенств:

*;*

*;*

*…………………*

**

**

то есть  *=* НОД *.*

Теорема 2.3 формулирует алгоритм Евклида нахождения наибольшего общего делителя целых чисел. Его вариантом является следующий – второй способ вычисления наибольшего общего делителя по алгоритму Евклида – вычисляем последовательно разности  до получения последней ненулевой разности, которая и совпадает с НОД *(a, b).*

**Пример 2.1.** помощью алгоритма Евклида найти НОД (72, 26).

**Решение**. В соответствии с теоремой 2.2   ; . Следовательно, НОД (72, 26) = 2.

*Теорема 2.4.* Если *d* = НОД *(a, b)*, то существуют такие целые *u* и *v*, что выполняется следующее соотношение (Безу): *d = au+ bv.*

**Пример 2.2.** Из примера 2.1 следует, что



Такой способ получения соотношения Безу для конкретных целых чисел называется расширенным алгоритмом Евклида. Он состоит из двух этапов: собственно алгоритма Евклида - прогонки вниз и прогонки вверх – последовательного выражения остатков в каждом из шагов предыдущего этапа (с соответствующим приведением подобных на каждом шаге).

**Определение 2.3.** Натуральное число ** называется *простым*, если оно делится только на1 и на себя.

*Теорема 2.5.* Всякое натуральное число ** либо является простым числом, либо имеет простой делитель.

Заметим, что из соотношения  натуральных чисел, больших единицы, следует, что, либо *p,* либо *q* принадлежит отрезку . Легко видеть, что наименьший натуральный делитель ** натурального числа ** является простым числом. Исторически первый метод проверки натурального числа ** на простоту заключается в делении его на простые числа, не превосходящие , носит название “решета Эратосфена”. К настоящему времени разработан достаточно большой цикл алгоритмов проверки числа на простоту.

*Теорема 2.6 (Евклид).* Простых чисел бесконечно много.

Значение простых чисел в том, что они по теореме 2.5 являются составными кирпичиками всех натуральных чисел.

**Определение 2.4.** Целые числа *a*  и  *b* называются *взаимно простыми,* еслиНОД .

*Теорема 2.7* (*Критерий взаимной простоты целых чисел*). Целые числа  *a* и *b* взаимно просты тогда и только тогда, когда существуют такие целые u и v, что выполняется равенство .

**Следствие.** НОД** тогда и только тогда, когдаНОД иНОД .

Важным в теории чисел и ее приложениях является следующее свойство взаимно простых целых чисел.

*Лемма 2.2.* Пусть произведение целых чисел *ab* делится на целое число *с* и НОД . Тогда *b* делится на *с*.

*Теорема 2.8**(Основная теорема арифметики)*. Всякое целое число ** однозначно раскладывается в произведение простых множителей

*.*

Если в этом равенстве собрать одинаковые множители, то получим каноническое разложение целого числа: .

**Пример 2.3.** Приведем примеры канонических разложений целых чисел:

а) 196 = 2⋅98 = 2⋅2⋅49 = 22⋅72;

б) 212-1 = 4095 = 32⋅5⋅7⋅13.

*Теорема 2.9.* Пусть *-* натуральное число*,* . Для любых целых чисел *a* и *b* следующие условия равносильны:

*1) a и b имеют одинаковые остатки от деления на *

*2) a – b делится на m, то есть a – b = mq для подходящего целого q;*

*3) a = b + mq для некоторого целого q.*

**Определение 2.5.**Целые числа *а* и *b* называются сравнимыми по модулю *m*, если они удовлетворяют одному из условий теоремы 2.9.Этот факт обозначают формулой ** илии называют данную формулу сравнением.

**Пример 2.4.** -57(mod 4) 11(mod 4) 23(mod 4) 3(mod 4).

**Пример 2.5.** Если  то всякое целое число сравнимо по модулю *m* со своим остатком от деления на *m*. Это следует из определения 2.5 и второго условия теоремы 2.9. Ведь *a*–*r* делится на *m*.

Основные свойства сравнений:

**1.** Пусть *.* Тогда  для всякого целого *c*, то есть к обеим частям сравнения можно добавить (или вычесть из обеих частей) одно и то же число.

**2.** Сравнения можно почленно складывать и вычитать: если **, *,* то  

**3.** Сравнения можно почленно перемножать: если ** *,* то **.

**4.** Сравнения можно почленно возводить в любую натуральную степень: если *,* то **.

**5.** Если в сравнении ** числа *a*, *b*, *m* имеют общий множитель *d*, то на него сравнение можно сократить: **.

**6.** Сравнение можно сократить на общий множитель, взаимно простой с модулем: если **, НОД (*d*, *m*) = 1, то из сравнения  следует сравнимость  и  по модулю .

**7.** Сравнение можно умножить на любой целый множитель: если **, то  для всякого целого *t*.

**8.** Рефлексивность: ** для любого целого *а* и всякого натурального *m* >1.

**9.** Симметричность: если **, то **.

**10.** Транзитивность: если **, **, то .

*Теорема 2.10*(*Малая теорема Ферма*). Пусть *p –* простое число и целое число *a* не делится на . Тогд*а .*

Теория сравнений и малая теорема Ферма позволяют быстро находить остаток от деления большого числа на простое число.

**Пример 2.6.** Найдем остаток от деления  на 31.

**Решение.** **. Поэтому в силу свойства 4 сравнений . Двоичная запись: 29=11101. Следовательно, для любого натурального *a* величина . Далее, . Поэтому . Тогда . Следовательно, . Таким образом, остаток от деления  на 31 равен 4.

**Практическая часть**

1.Найти канонические разложения чисел *а* и *b*.

2. Найти НОД  пользуясь a) алгоритмом Евклида, б) разложением чисел на простые множители.

3. С помощью расширенного алгоритма Евклида найти целые *u*, *v*, удовлетворяющие соотношению Безу: *au* + *bv* = НОД .

4. Найти остаток от деления данного числа на простое.

****

**Решение**

1. а = 7957549 = 52699 \* 151 = 349 \* 151 \* 151 = × 349;

б = 23118553 = 153103 \* 151 = 541 \* 283 \* 151;

1. а) 23118553 = 7957549 ∙ 2 + 7203455;

7957549 = 7203455 ∙ 1 + 754094;

7203455 = 754094 ∙ 9 + 416609;

754094 = 416609 ∙ 1 + 337485;

416609 = 337485 ∙ 1 + 79124;

337485 = 79124 ∙ 4 + 20989;

79124 = 20989 ∙ 3 + 16157;

20989 = 16157 ∙ 1 + 4832;

16157 = 4832 ∙ 3 + 1661;

4832 = 1661 ∙ 2 + 1510;

1661 = 1510 ∙ 1 + 151;

1510 = 151 ∙ 10 + 0;

НОД равен предыдущему остатку от деления

НОД (7957549; 23118553) = 151

б) 7957549 = 349 \* 151 \* 151;

23118553 = 541 \* 283 \* 151;

Общие множители чисел: 151;

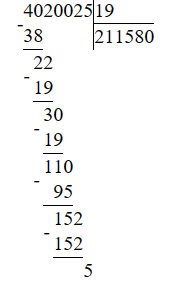
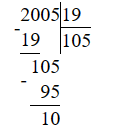
НОД (7957549; 23118553) = 151

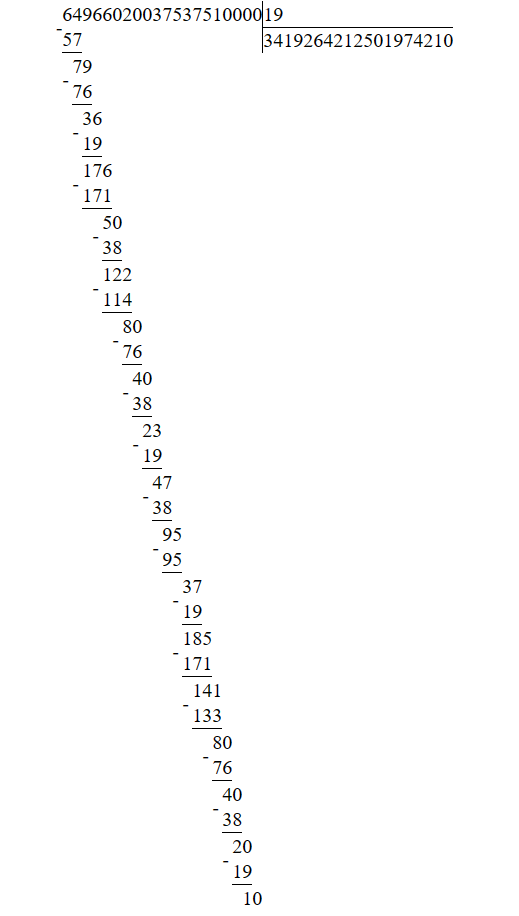
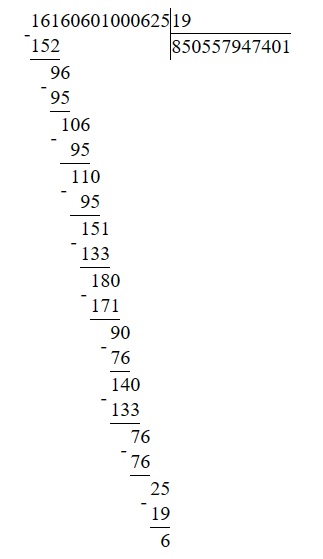
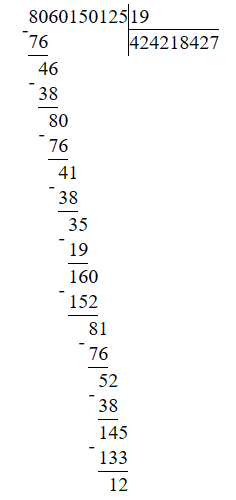
1. *au* + *bv =* 7957549 *\* u +* 23118553 *\* v =* НОД (7957549; 23118553) = 151

u = -969629, v = 9.

7957549 \* (-969629) + 23118553 \* 9 = 151

1.  на 19





2005 mod 19 = 10

mod 19 = 5

mod 19 = 12

mod 19 = 6

mod 19 = 2

mod 19 = 10 – “зациклились”

Длина цикла – 5

1999 = 399 \* 5 + 4, значит число , при делении на 19, даёт такой же остаток как и число , то есть 6.

**Вывод**

Получение основных сведений из курса теории чисел

**Практическое задание № 7**

**Тема «Настройка антивирусов»**

**Цель**: Овладение навыками настройки и использования различных антивирусов.

**Теоретические сведения**

При заражении компьютера вредоносными программами важно их обнаружить. Для этого следует знать об основных признаках проявления вредоносных программ. К ним можно отнести следующие:

 прекращение работы или неправильная работа ранее успешно функционировавших программ;

 медленная работа компьютера;

 невозможность загрузки операционной системы;

 исчезновение файлов и каталогов или искажение их содержимого;

 изменение даты и времени модификации файлов;

 изменение размеров файлов;

 неожиданное значительное увеличение количества файлов на диске;

 существенное уменьшение размера свободной оперативной памяти;

 вывод на экран непредусмотренных сообщений или изображений;

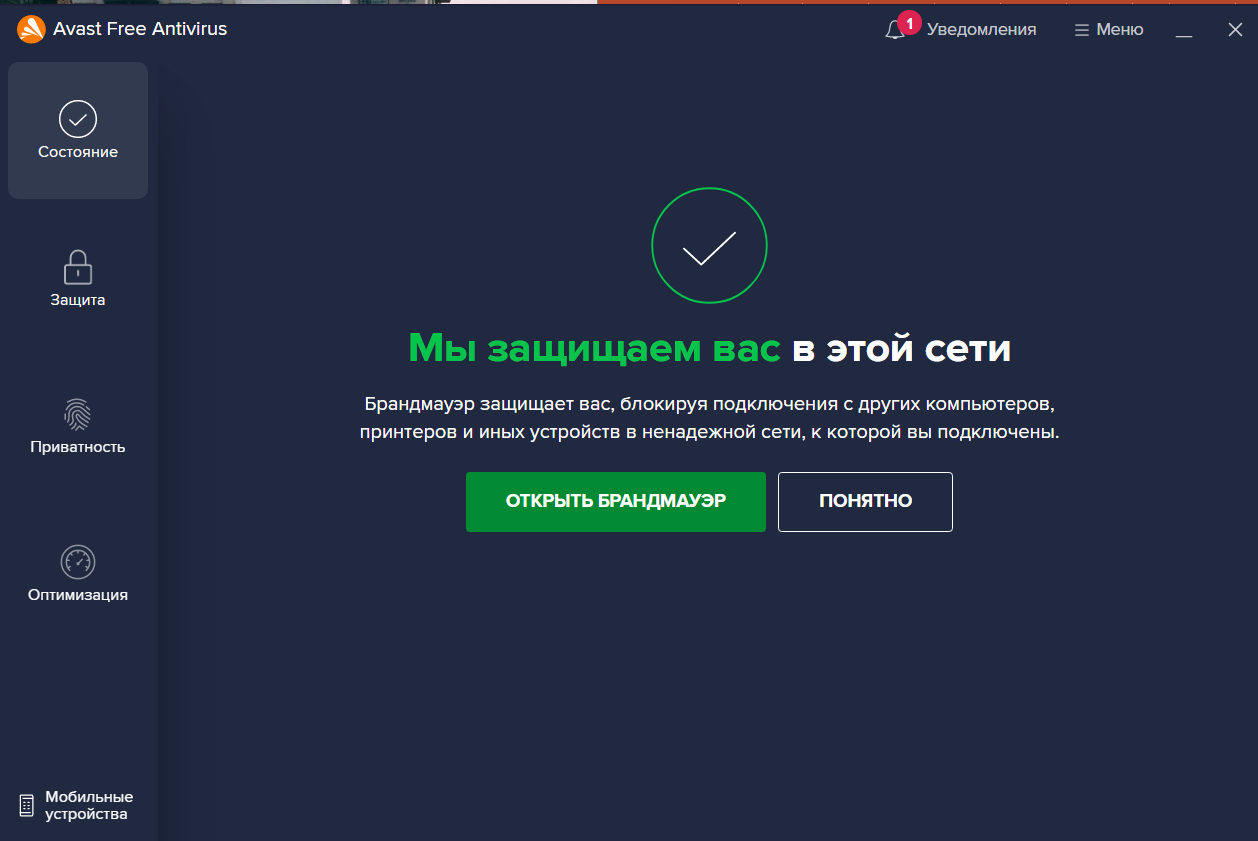
 подача непредусмотренных звуковых сигналов;

 частые зависания и сбои в работе компьютера.

**Задание**

1. Установить и настроить антивирусную программу по варианту.
2. Обновить базу данных сигнатур вирусов.
3. Выполнить сканирование дисков.

Рассмотрим антивирусное программное обеспечение Avast Free Antiviru



Рис*.1* – Главный экран

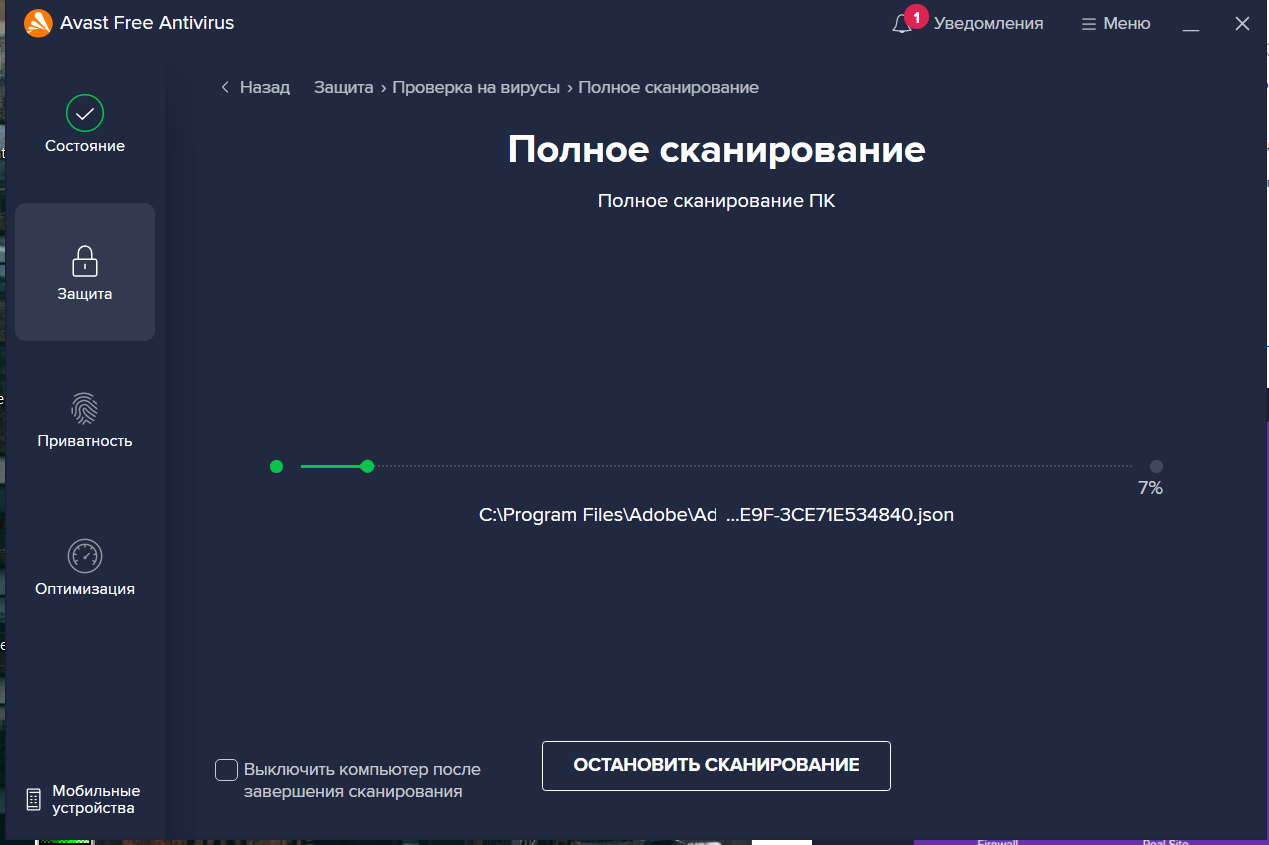


Рис.*2* – Полное сканирование ПК

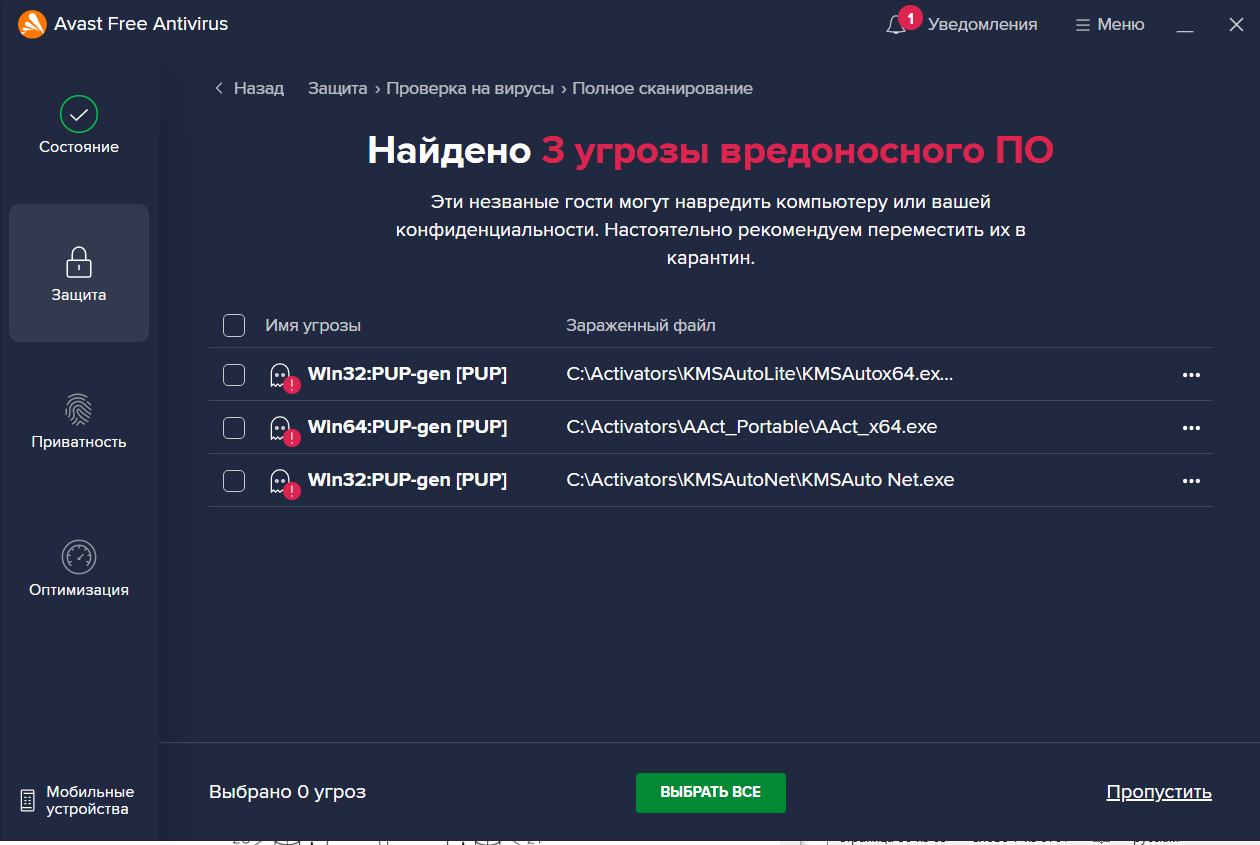


Рис.*3* – Результат полного сканирование ПК

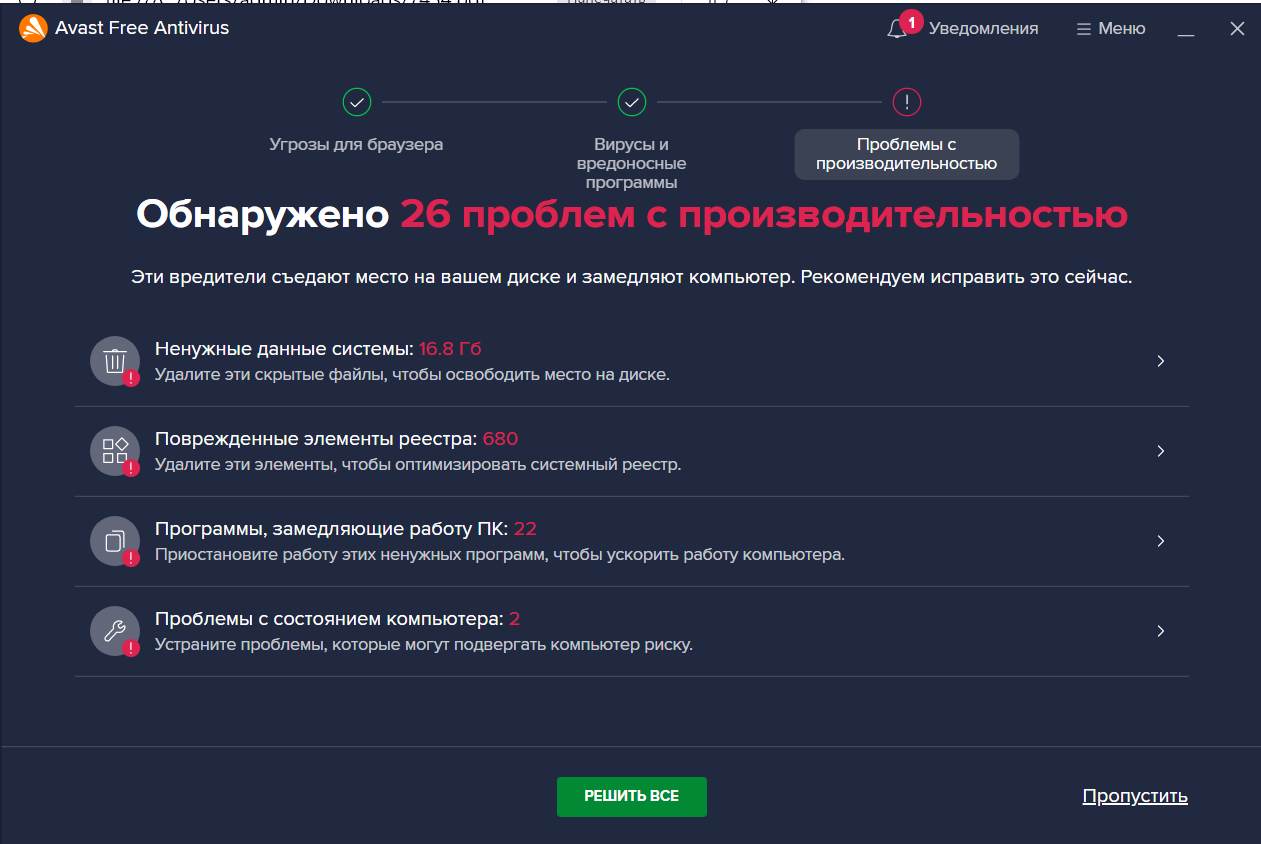


Рис.*4* – Интеллектуальное сканирование

|  |
| --- |
| **Практическое занятие №8** |
| **Тема «Изучение стандартных средств для реализации приложений, использующих симметричное и ассиметричное шифрование с использованием библиотеки** [**System.Security.Cryptography**](https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography)**»** |
| Цель: Изучить модель криптографии .NET Framework, основные классы и структуры данных, разработать приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования |
| **Теоретические сведения** |

**Криптография в .Net**

В .Net Framework присутствует пространство имён для выполнения криптографических операций под названием System.Security.Cryptography. Данное пространство имён предоставляет криптографические службы, включающие безопасное кодирование и декодирование данных, а также другие операции, такие как хеширование сообщений, генерация случайных чисел и проверка подлинности сообщений. Данная библиотека предоставляет доступ для использования различных реализаций алгоритмов в основном это программные интерфейсы CryptoApi (CAPI) и Cryptography Next Generation API (CNG API) помимо этого для некоторых алгоритмов возможно использование реализаций на основе OpenSsl.

CryptoAPI — интерфейс программирования приложений, который обеспечивает разработчиков Windows-приложений стандартным набором функций для работы с криптопровайдером. Входит в состав операционных систем Microsoft. Большинство функций CryptoAPI поддерживается, начиная с Windows 2000.

Cryptography Next Generation стала долгосрочной заменой CAPI. Данный набор интерфейсов поддерживает все алгоритмы предлагаемые CAPI а также другие алгоритмы перечисленные в своде правил Suite B Агентства национальной безопасности США [1]. Данный интерфейс поддерживает следующие длины ключей или размерность хеша.

* RSA 512 бит to 16384 бит, с шагом 64 бит
* DH 512 бит to 16384 бит, с шагом 64 бит
* DSA 512 бит to 1024 бит, с шагом 64 бит
* ECDSA P-256, P-384, P-521 (NIST Curves)
* ECDH P-256, P-384, P-521 (NIST Curves)
* MD2 128 бит
* MD4 128 бит
* MD5 128 бит
* SHA-1 160 бит
* SHA-256 256 бит
* SHA-384 384 бит
* SHA-512 512 бит

Рассматривая структуру наследования для симметричных алгоритмов в .Net стоит упомянуть что SymmetricAlgorithm является абстрактным классом, от который наследуют абстрактные классы для реализаций каждого из алгоритмов. В свою очередь каждая из реализации алгоритма является производной от абстрактного класса алгоритма. Ниже представлена структура наследования.

SymmetricAlgorithm

Aes

AesCng

AesManeged

AesCryptoServiceProvider

Des

DesCng

DesManeged

DesCryptoServiceProvider

TripleDes

TripleDesCng

TripleDesManeged

TripleDesCryptoServiceProvider

Данная структура наследования повторяется для каждого из трех типов поддерживаемых криптографических операций: SymmetricAlgorithm, AsymmetricAlgorithm, HashAlgorithm.

**Задание**

1. Ознакомиться с созданием криптографического приложения;
2. Выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам. Используя функции работы с файлами сохранить ключи шифрования, результаты шифрования и хеширования.
3. Для выше указанных алгоритмов используя Hex-редактор продемонстрировать ключи шифрования, зашифрованные и захешированные данные.

\*Реализовать проверку сообщения (фамилии) и хеша по примеру ЭЦП. Также продемонстрировать что будет если будет изменен хеш или сообщение. Пример того что нужно реализовать представлен ниже.



**Выполнение**

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

class Program

{

static byte[] EncryptString(string plainText, byte[] key)

{

using (var des3 = TripleDES.Create())

{

des3.Key = key;

des3.Mode = CipherMode.ECB;

des3.Padding = PaddingMode.PKCS7;

using (var encryptor = des3.CreateEncryptor())

{

byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);

byte[] encryptedBytes = encryptor.TransformFinalBlock(plainBytes, 0, plainBytes.Length);

return encryptedBytes;

}

}

}

static string DecryptString(byte[] encryptedBytes, byte[] key)

{

using (var des3 = TripleDES.Create())

{

des3.Key = key;

des3.Mode = CipherMode.ECB;

des3.Padding = PaddingMode.PKCS7;

using (var decryptor = des3.CreateDecryptor())

{

byte[] decryptedBytes = decryptor.TransformFinalBlock(encryptedBytes, 0, encryptedBytes.Length);

string decryptedText = Encoding.UTF8.GetString(decryptedBytes);

return decryptedText;

}

}

}

static string ComputeHash(string plainText)

{

using (var sha1 = SHA1.Create())

{

byte[] hashBytes = sha1.ComputeHash(Encoding.UTF8.GetBytes(plainText));

string hash = BitConverter.ToString(hashBytes).Replace("-", "").ToLower();

return hash;

}

}

static void Main()

{

// Генерация ключа шифрования

byte[] key = Encoding.UTF8.GetBytes("0123456789abcdef");

// Сохранение ключа шифрования в файл

using (FileStream fileStream = new FileStream("encryption\_key.bin", FileMode.Create))

{

fileStream.Write(key, 0, key.Length);

}

Console.WriteLine("Ключ шифрования сохранен в файле 'encryption\_key.bin'.");

// Шифрование фамилии

string surname = "Кореневский";

byte[] encryptedBytes = EncryptString(surname, key);

string encryptedSurname = Convert.ToBase64String(encryptedBytes);

Console.WriteLine("Зашифрованная фамилия: " + encryptedSurname);

// Дешифрование фамилии

byte[] decryptedBytes = Convert.FromBase64String(encryptedSurname);

string decryptedSurname = DecryptString(decryptedBytes, key);

Console.WriteLine("Расшифрованная фамилия: " + decryptedSurname);

// Хеширование фамилии

string hashedSurname = ComputeHash(surname);

Console.WriteLine("Хеш фамилии: " + hashedSurname);

// Демонстрация ключей шифрования с помощью Hex-редактора

using (FileStream fileStream = new FileStream("encryption\_key.bin", FileMode.Open))

{

byte[] keyBytes = new byte[fileStream.Length];

fileStream.Read(keyBytes, 0, keyBytes.Length);

string hexString = BitConverter.ToString(keyBytes).Replace("-", "");

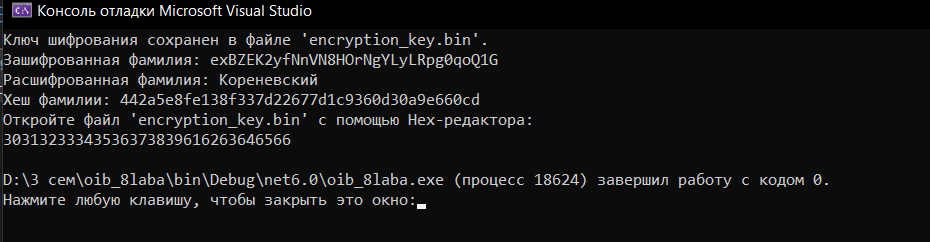
Console.WriteLine("Откройте файл 'encryption\_key.bin' с помощью Hex-редактора:");

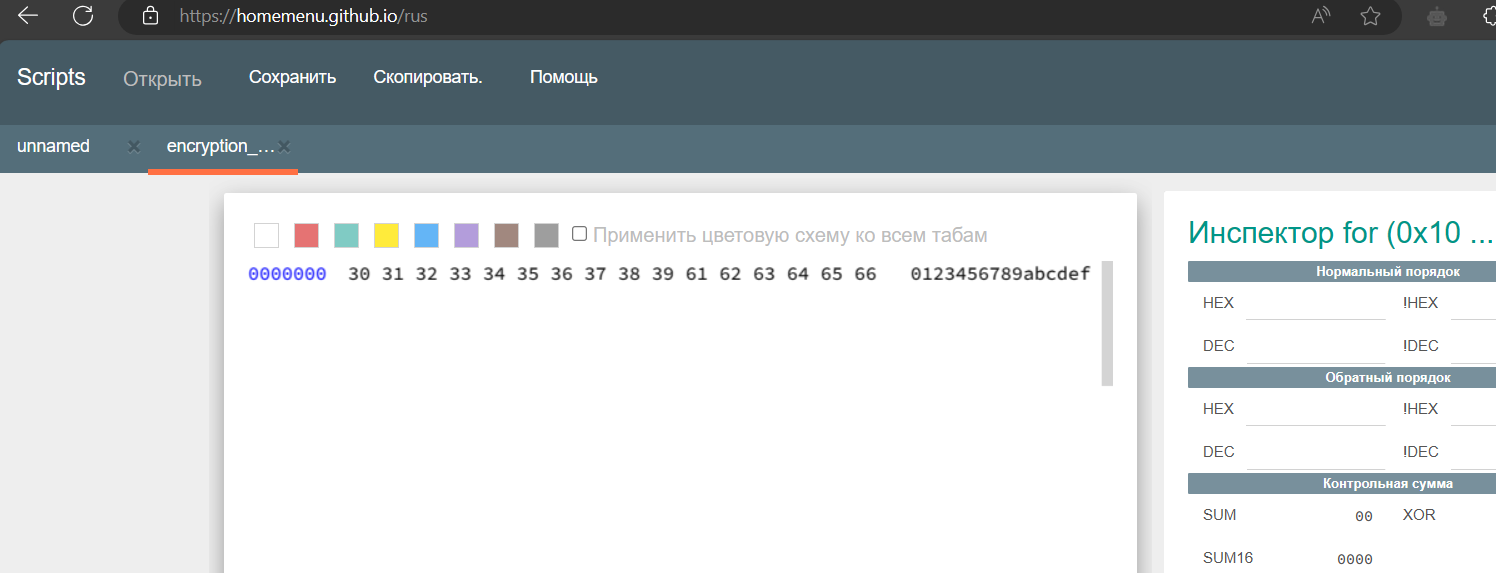
Console.WriteLine(hexString);

}

}

}





**Вывод:** Изучил модель криптографии .NET Framework, основные классы и структуры данных, разработал приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования.

**Практическое задание № 9**

**Тема «Изучение стандартных средств для реализации симметричного и ассиметричного шифрование с использованием SubtleCrypto в js»**

Цель: Изучить интерфейс **SubtleCrypto**, Основные классы и структуры данных, разработать приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования

**Криптография в Js**

Web Crypto API - это интерфейс, позволяющий использовать криптографические примитивы для построения систем с использованием криптографии. Данный интерфейсвключают в себя возможность генерировать, использовать и применять пары криптографических ключей; шифровать и дешифровать сообщения; надежно генерировать случайные числа

Некоторые браузеры реализовали интерфейс под названием Crypto, без точной структуры. Чтобы избежать путаницы, методы и свойства этого интерфейса были удалены из браузеров, реализующих Web Crypto API, и все методы Web Crypto API доступны в новом интерфейсе: SubtleCrypto.

Интерфейс SubtleCrypto Web Crypto API предоставляет ряд низкоуровневых криптографических функций. Доступ к функциям SubtleCrypto осуществляется через объёкт Crypto.subtle .

Этот объект содержит набор методов для выполнения общих криптографических функций, таких как шифрование, хеширование, подписывание и генерация ключей. Поскольку все криптографические операции выполняются с необработанными двоичными данными, каждый метод SubtleCrypto имеет дело с типами ArrayBuffer и ArrayBufferView. Из-за того, что строки так часто становятся предметом криптографических операций, классы TextEncoder и TextDecoder будут часто использоваться вместе с SubtleCrypto для преобразования в строки и обратно.

Одной из проблем криптографии является генерация случайных чисел. Если будет использоватся Math.random() то будет сгенерировано псевдослучайное число которое использует генератор PRNG (pseudorandom number generator). Поскольку сгенерированное число внутреннее состояние PRNG то использование данного алгоритма для криптографии неприемлемо. Для решения данной задачи решить эту проблему, криптографически безопасный генератор псевдослучайных чисел (cryptographically secure pseudorandom number generator, CSPRNG) дополнительно включает в себя источник энтропии качестве входных данных, такие системных свойств, которые проявляют непредсказуемое поведение. Это алгоритм медленнее PRNG, но значения, генерируемые CSPRNG, являются остаточно непредсказуемыми для криптографических целей.

Генерация случайного CryptoKey выполняется с помощью метода SubtleCrypto.generateKey(algorithm, extractable, keyUsages). В этот метод передается объект params, указывающий целевой алгоритм, логическое значение,указывающее, должен ли ключ извлекаться из объекта CryptoKey, и массив строк — keyUsages, указывающий, с какими методами

Поскольку разные алгоритмы используют разный набор данных для ключей то в первый параметр содержит сооствутсвующее название алгоритма.

* RSA (RSASSA-PKCS1-v1\_5, RSA-PSS, or RSA-OAEP )использует объект RsaHashedKeyGenParams.
* ECDSA и ECDHи спользует объект EcKeyGenParams.
* HMAC использует объект HmacKeyGenParams.
* AES (AES-CTR, AES-CBC, AES-GCM, AES-KW) использует объект AesKeyGenParams

Значение extractable является логическим значением и указывает на возможность экспорта ключа.

Третий параметр keyUsages описывает, с какими алгоритмами можно использовать ключ.

* encrypt: Ключ используется для шифрования сообщений.
* decrypt: Ключ используется для расшифровки сообщений.
* sign: Ключ используется для подписи сообщений.
* verify: Ключ используется для проверки подписанного сообщения.
* deriveKey: Ключ используется для получения ключа.
* deriveBits: Ключ используется для получения битов.
* wrapKey: Ключ используется для упаковки ключа
* unwrapKey: Ключ используется для распоковки ключа.

**Задание к выполнению:**

1. Ознакомиться с созданием криптографического приложения
2. Выполнить генерацию и вывод в консоль случайный чисел.
3. Выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам.

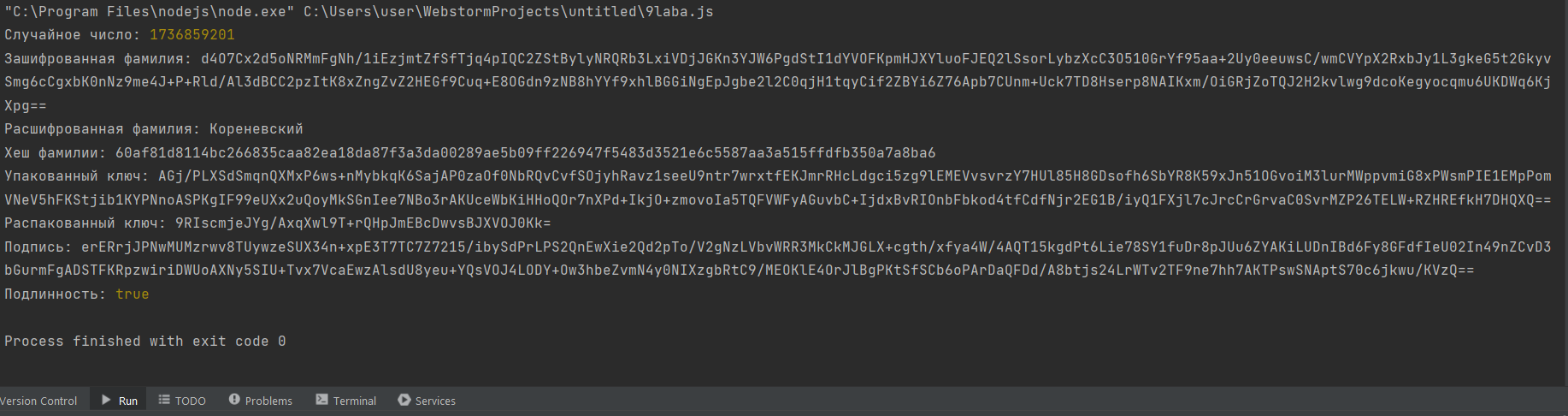
|  |  |  |
| --- | --- | --- |
| № | Алгоритм шифрования(размер ключа) | Алгоритм хеширования |
| 1 | RSA-OAEP | SHA-1 |
| 2 | AES-CTR | SHA-256 |
| 3 | AES-CBC | SHA-384 |
| 4 | AES-GCM | SHA-512 |
| 5 | AES-CTR | SHA-1 |
| 6 | AES-CBC | SHA-256 |
| 7 | AES-GCM | SHA-384 |
| 8 | RSA-OAEP | SHA-512 |
| 9 | AES-CBC | SHA-1 |
| 10 | AES-GCM | SHA-256 |
| 11 | RSA-OAEP | SHA-384 |
| 12 | AES-CTR | SHA-512 |
| 13 | AES-GCM | SHA-1 |
| 14 | RSA-OAEP | SHA-256 |
| 15 | AES-CTR | SHA-384 |

Если количество человек в группе больше 15 то номер варианта определяется как x=n mod 15 где n номер в списке. В случае номер варианта в ходе решения уравнения ровен 0 то номер варианта равен 15.

1. Продемонстрировать упаковку и распаковку ключа, полученного в предыдущем задании используя алгоритм AES-KW.
2. Выполнить процедуру подписи сообщения и проверку подлинности с использованием RSA-PSS или ECDSA на выбор.

**Выполнение**

const crypto = ***require***('crypto');  
const { generateKeyPairSync } = ***require***('crypto');  
const { privateKey, publicKey } = generateKeyPairSync('rsa', {  
 modulusLength: 2048,  
 publicKeyEncoding: {  
 type: 'spki',  
 format: 'pem'  
 },  
 privateKeyEncoding: {  
 type: 'pkcs8',  
 format: 'pem'  
 }  
});  
  
// Генерация случайного числа  
const randomNum = crypto.randomBytes(4).readUInt32BE(0);  
***console***.log('Случайное число:', randomNum);  
  
// Шифрование и дешифрование фамилии  
const surname = 'Кореневский';  
  
const encryptedSurname = crypto.publicEncrypt(publicKey, ***Buffer***.from(surname, 'utf8'));  
***console***.log('Зашифрованная фамилия:', encryptedSurname.toString('base64'));  
  
const decryptedSurname = crypto.privateDecrypt(privateKey, encryptedSurname);  
***console***.log('Расшифрованная фамилия:', decryptedSurname.toString('utf8'));  
  
// Хеширование фамилии  
const hashedSurname = crypto.createHash('sha384').update(surname).digest('hex');  
***console***.log('Хеш фамилии:', hashedSurname);  
  
// Упаковка и распаковка ключа с использованием AES-KW  
const aesKey = crypto.randomBytes(32); // 256 битный ключ  
const packedKey = crypto.publicEncrypt(publicKey, aesKey);  
***console***.log('Упакованный ключ:', packedKey.toString('base64'));  
  
const unpackedKey = crypto.privateDecrypt(privateKey, packedKey);  
***console***.log('Распакованный ключ:', unpackedKey.toString('base64'));  
  
// Подпись сообщения и проверка подлинности  
const message = 'Сообщение для подписи';  
  
const sign = crypto.sign('sha384', ***Buffer***.from(message, 'utf8'), {  
 key: privateKey,  
 padding: crypto.constants.***RSA\_PKCS1\_PSS\_PADDING***,  
 saltLength: crypto.constants.***RSA\_PSS\_SALTLEN\_DIGEST***});  
***console***.log('Подпись:', sign.toString('base64'));  
  
const isAuthentic = crypto.verify('sha384', ***Buffer***.from(message, 'utf8'), {  
 key: publicKey,  
 padding: crypto.constants.***RSA\_PKCS1\_PSS\_PADDING***,  
 saltLength: crypto.constants.***RSA\_PSS\_SALTLEN\_DIGEST***}, sign);  
***console***.log('Подлинность:', isAuthentic);



**Вывод**: Изучил интерфейс **SubtleCrypto**, Основные классы и структуры данных, разработал приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования.