МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1—98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ ЗАНЯТИЯМ**

по дисциплине «**Основы информационной безопасности**»

Вариант 11

**Исполнитель**

студент 2 курса 8 группы Кривенчук Максим Игоревич

**Руководитель** Ржеутская Н. В.

**Практическое задание № 1**

**Тема «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.

**Теоретическое введение**

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 задания**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | *E*0 | *E* | *K* | *C* |
| 5 | 11000 | 9500 | 3 | 1500 |

**Исполнительная часть**

Из формулы (1):

Найдем по формуле (2) относительную эффективность: **0,86**

Снижение эффективности функционирования объекта при наличии средств защиты**:**

Итоговая эффективность:

**0,818**

**Вывод:** В ходе выполнения практического задания я научился решать задачи разработки средств защиты для обеспечения максимальной эффективности объекта в условиях несанкционированного доступа и было выяснено, что с данными значениями этого варианта защита проекта эффективна.

**Практическое задание №2**

**Тема «Разработка политики информационной безопасности бизнес-компании»**

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

**Введение**

С развитием информационных технологий появилась необходимость прорабатывания требований в области защиты информации. Это подразумевает под собой, что каждая система должна поддерживать и проводить определенный ряд мер по работе с информационной безопасностью этой системы. В случае, если политика безопасности не будет проработана, появляются определенные риски, от кражи данных или нанесения вреда имуществу и вплоть до полного прекращения существования системы, в которой не уделялось внимание этому вопросу.

В наше время не только самые крупные корпорации, но и небольшие компании заинтересованы в информационной безопасности. Однако стоит помнить, что каждый случай уникален, и необходимо выбирать подход. Процесс построения концепции ИБ можно поделить на несколько уровней : административный (меры, предпринимаемые руководством организации – политика безопасности –  совокупность документированных управленческих решений, направленных на защиту информации и ассоциированных с ней ресурсов), к процедурному уровню относятся меры безопасности, реализуемые людьми, к программно-техническому уровню можно отнести такие механизмы безопасности, как идентификация и аутентификация пользователей, управление доступом, криптография и т. д. Важно управлять информационной системой в целом и механизмами безопасности в особенности. Упомянутые меры безопасности должны опираться на общепринятые стандарты, быть устойчивым к сетевым угрозам, учитывать специфику отдельных сервисов.

Источниками угроз могут стать сотрудники организации, программное обеспечение, аппаратные средства, несанкционированный доступ к информационным ресурсам, незаконное копирование данных в информационных системах, противозаконный сбор и использование информации и другие.

Со всеми этими факторами необходимо разобраться и разработать проект политики безопасности бизнес-компании.

1. **Описание структуры бизнес-компании**

Риэлтерская компания "Инвестиции в Недвижимость" является специализированной организацией, занимающейся операциями на рынке недвижимости. В рамках описания структуры компании, её участники могут быть подразделены на следующие группы:

– Пользователи услуг компании включают клиентов, заинтересованных в приобретении, продаже или аренде недвижимости, а также партнёров и контрагентов, таких как другие брокерские компании, заинтересованные стороны и юридические лица, с которыми компания сотрудничает.

– Работники компании состоят из риэлторов и брокеров, ответственных за поиск и предоставление клиентам информации о недвижимости, проведение сделок и консультации по инвестициям в недвижимость, а также административного персонала, включая секретарей и ассистентов, обеспечивающих бесперебойное функционирование офиса.

– Управляющие включают топ-менеджмент и руководителей, занимающихся стратегическим планированием, разработкой бизнес-стратегии, финансовым управлением и маркетингом компании, а также отдел кадров, отвечающий за найм и обучение персонала, управление трудовыми отношениями, и отдел информационных технологий, обеспечивающий функционирование информационных систем и техническую поддержку сотрудников.

– Отдел безопасности информации и конфиденциальности ответственен за обеспечение безопасности данных клиентов и соблюдение законодательства о защите данных.

– Обслуживающий персонал включает сотрудников отдела по работе с недвижимостью, обслуживающих объекты недвижимости, проводящих технические инспекции и обеспечивающих поддержку клиентов после заключения сделок, а также бухгалтерию и финансовый отдел, занимающиеся учетом финансовых операций компании и уплатой налогов, и отдел маркетинга и рекламы, осуществляющий продвижение услуг компании и взаимодействие с клиентами.

Каждое из этих звеньев важно для успешного функционирования риэлтерской компании. Сотрудники разных подразделений взаимодействуют друг с другом и с внешними стейк-холдерами (клиентами, партнёрами, регуляторами) для достижения общей цели - предоставления высококачественных услуг в сфере недвижимости.

Важной частью функционирования риэлтерской компании является обработка и хранение конфиденциальных данных клиентов. Эти данные включают в себя информацию о недвижимости, контактных данных клиентов, финансовых операциях и другую персональную информацию. Управляющие лица компании и сотрудники отдела безопасности информации обязаны обеспечивать защиту и конфиденциальность этих данных, соблюдая законы и нормы, регулирующие обработку и хранение персональных данных.

Функции компании "Инвестиции в Недвижимость" включают в себя предоставление клиентам качественных услуг в сфере недвижимости, обеспечение уплаты налогов и создание рабочих мест для сотрудников. Структура компании разработана таким образом, чтобы эффективно выполнять эти функции и удовлетворять потребности клиентов, соблюдая при этом требования законодательства и стандарты информационной безопасности.

1. **Оценка рисков**

**Несанкционированный доступ к данным:** это одна из наиболее серьезных угроз информационной безопасности. Несанкционированные лица могут попытаться получить доступ к вашим базам данных с личными данными клиентов и сотрудников. Это может произойти через взлом паролей, атаки на слабые точки в системе безопасности или даже внутренних угроз со стороны сотрудников. Потенциальные последствия включают утечку конфиденциальных данных, в том числе их использование в мошенничестве, и потерю доверия клиентов.

**Утечка конфиденциальной информации:** утечка информации о недвижимости, сделках и финансовых операциях клиентов может повлечь за собой серьезные последствия. Это может произойти через хакерские атаки, утечку данных сотрудниками или даже случайно. Важно разработать стратегии для предотвращения таких утечек и гарантировать безопасность хранения и передачи конфиденциальных данных.

**Взлом веб-сайта или приложения:** взлом вашего веб-сайта или приложения может быть катастрофическим для бизнеса. Это может привести к потере данных, перерывам в работе и нарушению обслуживания клиентов. Помимо технических мер безопасности, важно обеспечить мониторинг на предмет внутренних и внешних угроз, а также иметь план восстановления после инцидента.

**Риск внутренних угроз:** внутренние угрозы могут включать в себя действия сотрудников, которые имеют доступ к конфиденциальной информации. Это может быть продажа данных, незаконное использование или утечка по недосмотру. Необходимо регулярно обучать сотрудников в вопросах информационной безопасности, а также иметь системы мониторинга и аудита для выявления подозрительной активности внутри компании.

Оценка рисков информационной безопасности является важным шагом в разработке стратегии безопасности компании и позволяет идентифицировать уязвимости и потенциальные угрозы, чтобы затем разработать меры по их управлению и предотвращению.

1. **Разработка мер защиты**

Начнем с разработки мер **административного уровня**. В нем должны быть рассмотрены следующие аспекты:

• управление доступом к средствам вычислительной техники, программе и данным;

• антивирусная защита;

• вопросы резервного копирования;

• проведение ремонтных и восстановительных работ;

• информирование об инцидентах об области ИБ.

Разработать четкие правила доступа, определяющие, какие сотрудники имеют доступ к средствам вычислительной техники, программам и данным.

Регулярно обновлять список пользователей и их уровни доступа в соответствии с изменениями в организационной структуре компании.

Установить и регулярно обновлять антивирусное программное обеспечение на всех компьютерах и серверах компании.

Проводить ежедневное или еженедельное создание резервных копий всех важных данных компании, включая информацию о сделках с недвижимостью и финансовых операциях.

Хранить резервные копии в защищенных помещениях, удаленных от основных серверов компании, чтобы избежать потери данных в случае физических инцидентов.

Разработать планы восстановления после различных инцидентов, таких как сбои в работе информационных систем или кибератаки.

Проводить регулярные учебные учения с сотрудниками, чтобы обеспечить готовность к быстрому восстановлению работы.

Создать процедуры информирования сотрудников и клиентов о возможных инцидентах информационной безопасности и мероприятиях по их устранению.

Далее организация сводится к **процедурному уровню**. Можно выделить следующие группы процедурных мер:

• управление персоналом;

• физическая защита;

• поддержание работоспособности;

• реагирование на нарушения режима безопасности;

• планирование восстановительных работ.

Разработать политику информационной безопасности, которая устанавливает правила и требования для сотрудников.

Обеспечить обучение всех сотрудников по вопросам информационной безопасности, включая конфиденциальность данных и соблюдение политики безопасности.

Назначить ответственных сотрудников, отвечающих за соблюдение политики безопасности и контроль за ее соблюдением.

Установить системы контроля доступа к помещениям с серверами и критической инфраструктурой.

Установить системы видеонаблюдения для мониторинга физической безопасности.

Проводить регулярное техническое обслуживание и мониторинг всех информационных систем и оборудования, чтобы предотвратить сбои.

Вести инцидентные реестры для документирования и анализа всех инцидентов информационной безопасности.

Анализировать и расследовать инциденты для выявления уязвимостей и устранения потенциальных угроз.

Разработать подробные планы восстановления после инцидентов, включая процедуры перехода на резервные площадки и восстановления работы.

Далее переходим к **программно-техническому уровню**.

К нему относятся:

• идентификация и аутентификация пользователей;

• управление доступом;

• протоколирование и аудит;

• криптография;

• экранирование;

• обеспечение высокой доступности и т.д.

Внедрить систему идентификации пользователей, которая удостоверяет личность пользователя.

Внедрить механизмы аутентификации для проверки прав доступа пользователей.

Разработать ролевую модель, определяющую, какие роли имеют доступ к каким данным и ресурсам.

Применить шифрование данных и сетевого трафика для защиты от несанкционированного доступа и перехвата информации.

Вести журнал аудита для записи и анализа действий пользователей, а также выявления аномальных событий.

Применить методы криптографии для защиты конфиденциальных данных при их передаче и хранении.

Использовать методы экранирования и защиты информации на мобильных устройствах сотрудников.

Создать резервные системы и каналы связи для обеспечения непрерывной работы компании даже при сбоях или инцидентах.

**Вывод**

Информационная безопасность системы – очень важный аспект, про который не стоит забывать при планировании компании. Ведь под угрозу могут попасть не только персональные и конфиденциальные данные компании и других лиц, связанные с ней, но также могут быть похищены денежные ресурсы. В данной лабораторной работе на практике был получен опыт создания концепции информационной безопасности бизнес-компании. В ходе её выполнения была описана структура риэлтерской компании, которая состоит из нескольких взаимосвязанных компонентах, функционирующих как одно целое. Были описаны риски, угрозы, уязвимости, как-либо связанные с информационной безопасностью. Разработан ряд мер защиты, включающий в себя управление доступом к средствам вычислительной техники, программа и данным, антивирусную защиту, вопросы резервного копирования, проведение ремонтных и восстановительных работ, информирование об инцидентах об области ИБ, идентификация и аутентификация пользователей, управление доступом, протоколирование и аудит, криптография, экранирование, обеспечение высокой доступности и некоторые другие.

**Практическое задание № 3**

**Тема «Настройка Брандмауэра Windows»**

**Цель**: Овладение навыками настройки и использования Брандмауэра Windows.

Брандмауэр (Межсетевой экран) - это аппаратный или программный комплекс, позволяющий проверять (фильтровать) входные и выходные потоки данных, проходящие через интернет или сеть. В случаи нарушения политики безопасности компьютера, брандмауэр блокирует эти данные. Межсетевой экран является одним из основных компонентов защиты сетей. Политика реализации межсетевых экранов определяет правила доступа к ресурсам внутренней сети.

Первое задние практической работы – создать правила для входящих подключений. Для создания правила блокировки подключения была выбрана программа NetCracker, выбираем блокирование всех профилей.

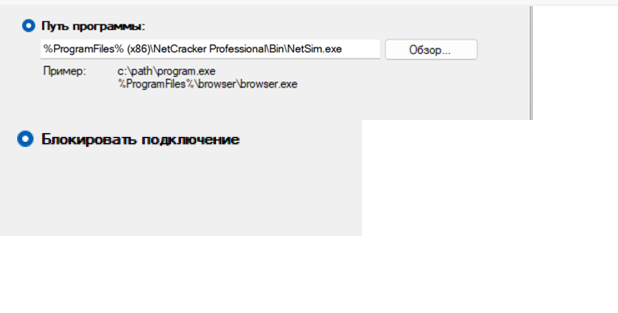


Рисунок 1 – Создание правила

После создания правила в общем списке оно выглядит следующим образом:

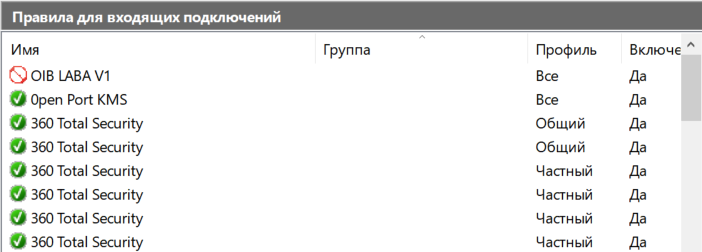


Рисунок 2 – Запрещающее правило для входящего подключения

Мы можем рассмотреть его детально при двойном клике мышью. Далее таким же образом создаём правило для входящих подключений на разрешение, также для всех профилей. В списке разрешающее правило обозначается как белая галка в зеленом кругу.

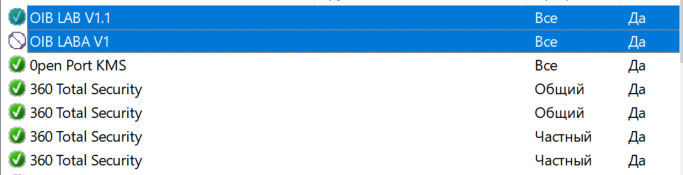


Рисунок 3 – Разрешающее правило для входящего подключения

Далее перейдем к созданию правил для исходящих подключений. Принцип их создания ничем не отличается от предыдущих. Для запрещающего внешнее соединение правила была выбрана программа InternetExplorer, для разрешающего – Unlocker.

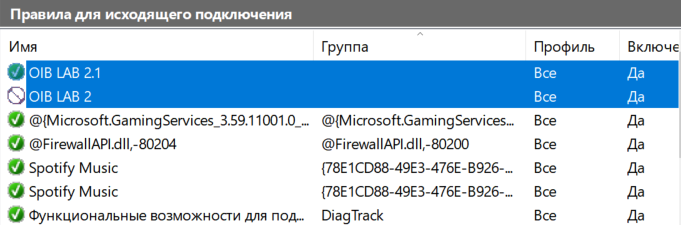


Рисунок 4 – Задание №2

В задании №3 предлагается отключить все созданные ранее правила, дабы вернуть состояние Брандмауэра в исходное состояние. Стоит отметить, что в правилах, сгенерированных системой и приложениями есть только активные на разрешение и отключенные.



Рисунок 5 – Отключение правил

Далее опробуем действие нескольких консольных команд. Первая будет посвящена вызову редактору реестра – команда regedit – рисунок 6. Команда verifier запускает диспетчер проверки драйверов, control admintools – панель админинстрирования, chkdsk – проверку дисков, calc – калькулятор. Можно сказать, что эти команды являются своеобразными «шорткатами» для запуска определенных процессов.

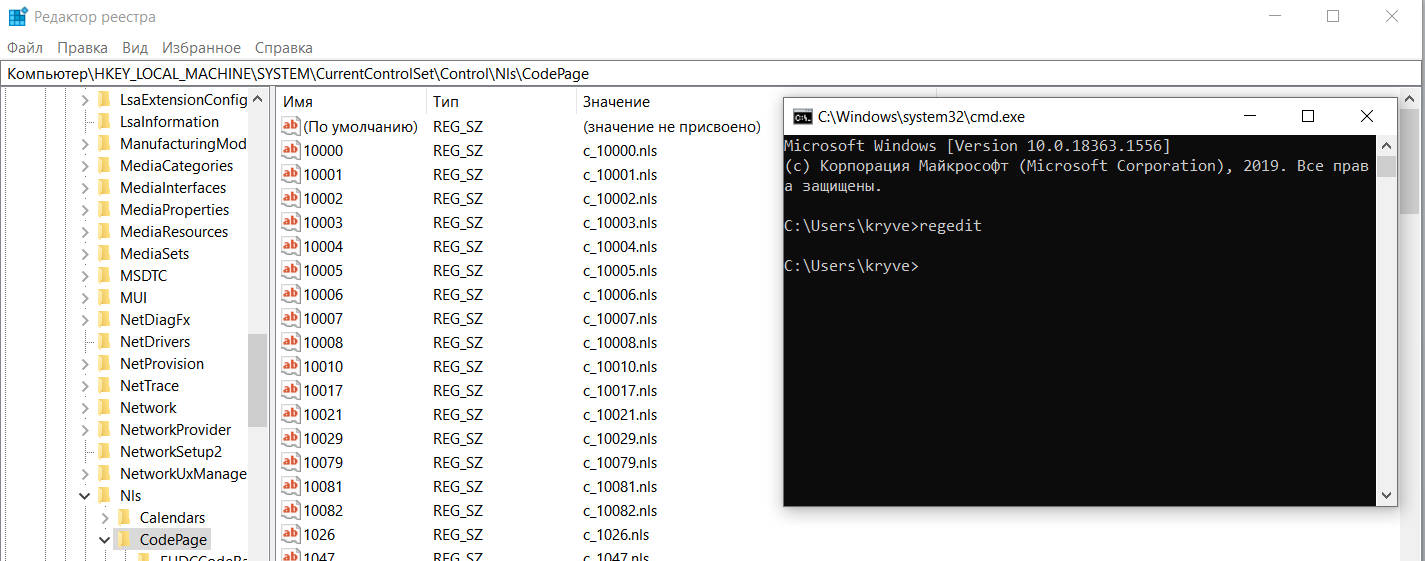


Рисунок 6 – Вызов редактора реестра через консольную команду.

**Вывод**

В ходе лабораторной работы была исследована такая технология, как Брандмауэр Windows, позволяющий проверять (фильтровать) входные и выходные потоки данных, проходящие через интернет или сеть. Были получены практические навыки создания правил фильтрования подключения для определенных программ, а также навыки использования консольных команд.

**Практическое задание №4.1**

**Тема «Криптографическая защита информации»**

Цель: овладение основными криптографическими алгоритмами симметричного шифрования

**Теоретическое введение**

Криптография - наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним) и аутентичности (целостности и подлинности авторства) информации.

Изначально криптография изучала методы шифрования информации – обратимого преобразования открытого (исходного) текста на основе секретного алгоритма и/или ключа в шифрованный текст (шифротекст). Традиционная криптография образует раздел симметричных криптосистем, в которых зашифрование и расшифрование проводится с использованием одного и того же секретного ключа.

Помимо этого современная криптография включает в себя асимметричные криптосистемы, системы электронной цифровой подписи, хеш-функции, управление ключами, получение скрытой информации, квантовую криптографию.

Шифрованием (encryption) называют процесс преобразования открытых данных (plaintext) в зашифрованные (шифртекст, ciphertext) или зашифрованных данных в открытые по определенным правилам с применением ключей.

В англоязычной литературе зашифрование / расшифрование – enciphering / deciphering.

Классификация алгоритмов шифрования

1. Симметричные (с секретным, единым ключом, одноключевые, single-key).

1.1. Потоковые:

· с одноразовым или бесконечным ключом (infinite-key cipher);

· с конечным ключом;

· на основе генератора псевдослучайных чисел.

1.2. Блочные:

1.2.1. Шифры перестановки (permutation, P-блоки);

1.2.2. Шифры замены (substitution, S-блоки):

· моноалфавитные;

· полиалфавитные;

2. Асимметричные (с открытым ключом, public-key):

· Диффи-Хеллман DH (Diffie, Hellman);

· Райвест-Шамир-Адлeман RSA (Rivest, Shamir, Adleman);

· Эль-Гамаль (ElGamal).

Симметричные алгоритмы шифрования (или криптография с секретными ключами) основаны на том, что отправитель и получатель информации используют один и тот же ключ. Этот ключ должен храниться в тайне и передаваться способом, исключающим его перехват.

Обмен информацией осуществляется в 3 этапа:

* отправитель передает получателю ключ (в случае сети с несколькими абонентами у каждой пары абонентов должен быть свой ключ, отличный от ключей других пар);
* отправитель, используя ключ, зашифровывает сообщение, которое пересылается получателю;
* получатель получает сообщение и расшифровывает его.

Если для каждого дня и для каждого сеанса связи будет использоваться уникальный ключ, это повысит защищенность системы.

При блочном шифровании информация разбивается на блоки фиксированной длины и шифруется поблочно. Блочные шифры бывают двух основных видов:

· шифры перестановки (transposition, permutation, P-блоки);

· шифры замены (подстановки, substitution, S-блоки).

Шифры перестановок переставляют элементы открытых данных (биты, буквы, символы) в некотором новом порядке. Различают шифры горизонтальной, вертикальной, двойной перестановки, решетки, лабиринты, лозунговые и др.

Шифры замены заменяют элементы открытых данных на другие элементы по определенному правилу. Paзличают шифры простой, сложной, парной замены, буквенно-слоговое шифрование и шифры колонной замены. Шифры замены делятся на две группы:

· моноалфавитные (код Цезаря);

· полиалфавитные (шифр Видженера, цилиндр Джефферсона, диск Уэтстоуна, Enigma).

В моноалфавитных шифрах замены буква исходного текста заменяется на другую, заранее определенную букву. Например в коде Цезаря буква заменяется на букву, отстоящую от нее в латинском алфавите на некоторое число позиций.



Очевидно, что такой шифр взламывается совсем просто. Нужно подсчитать, как часто встречаются буквы в зашифрованном тексте, и сопоставить результат с известной для каждого языка частотой встречаемости букв.

В полиалфавитных подстановках для замены некоторого символа исходного сообщения в каждом случае его появления последовательно используются различные символы из некоторого набора. Понятно, что этот набор не бесконечен, через какое-то количество символов его нужно использовать снова. В этом слабость чисто полиалфавитных шифров.

В современных криптографических системах, как правило, используют оба способа шифрования (замены и перестановки). Такой шифратор называют составным (product cipher). Oн более стойкий, чем шифратор, использующий только замены или перестановки.

В асимметричных алгоритмах шифрования (или криптографии с открытым ключом) для зашифровывания информации используют один ключ (открытый), а для расшифровывания - другой (секретный). Эти ключи различны и не могут быть получены один из другого.

Схема обмена информацией такова:

· получатель вычисляет открытый и секретный ключи, секретный ключ хранит в тайне, открытый же делает доступным (сообщает отправителю, группе пользователей сети, публикует);

· отправитель, используя открытый ключ получателя, зашифровывает сообщение, которое пересылается получателю;

· получатель получает сообщение и расшифровывает его, используя свой секретный ключ.

**Шифрование с помощью аффинной системы подстановок Цезаря**

В данном методе используется ключ шифрования в виде пары целых чисел (A, K). Число A задает переход при шифровании вперед на A\*J букв, а число K – дополнительное смещение по алфавиту на K букв. Следовательно, аффинную систему подстановок Цезаря можно описать следующей формулой:

(5.1)

где A>=0, K<M, НОД(A,M) = 1, I – значение кода символа в результирующей таблице, J – значение кода символа в исходной таблице.

Например, пусть M = 26, A = 3, K = 5. Тогда, очевидно, НОД (3,26) = 1, и мы получаем следующее соответствие между числовыми кодами букв:

Таблица 5.1– таблица кодов символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| J | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 3t+5 | 5 | 8 | 11 | 14 | 17 | 20 | 23 | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 1 | 4 | 7 | 10 | 13 | 16 | 19 | 22 | 25 | 2 |

Преобразуя числа в буквы английского языка, получаем следующее соответствие для букв открытого текста и шифр текста:

Таблица 5.2– таблица символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| F | I | L | O | R | U | Х | А | D | G | J | М | Р | S | V | Y | В | Е | Н | K | N | Q | Т | W | Z | C |

Функция дешифровки выглядит следующим образом:

(5.2)

**Шифр Трисемуса**

В 1508 г. аббат из Германии Иоганн Трисемус написал печатную работу по криптологии под названием «Полиграфия». В этой книге он впервые систематически описал применение шифрующих таблиц, заполненных алфавитом в случайном порядке. Для получения такого шифра замены обычно использовались таблица для записи букв алфавита и ключевое слово (или фраза). В таблицу сначала вписывалось по строкам ключевое слово, причем повторяющиеся буквы отбрасывались. Затем эта таблица дополнялась не вошедшими в нее буквами алфавита по порядку. На рис.4.7 изображена таблица с ключевым словом «ДЯДИНА».

Таблица 5.3 – Таблица шифрозамен для шифра Трисемуса



Каждая буква открытого сообщения заменяется буквой, расположенной под ней в том же столбце. Если буква находится в последней строке таблицы, то для ее шифрования берут самую верхнюю букву столбца. Например, исходное сообщение «АБРАМОВ», зашифрованное – «ЖЗЦЖУФЙ».

**Шифр Плейфейра**

Шифр Плейфера использует матрицу 5х5 (для латинского алфавита, для кириллического алфавита необходимо увеличить размер матрицы до 4х8 или 6х6), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки матрицы буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо, либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом, составляет матрицу 5х5 и является ключом шифра.

Для того чтобы зашифровать сообщение, необходимо разбить его на биграммы (группы из двух символов), например «Hello World» становится «HE LL OW OR LD», и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга. Затем, руководствуясь следующими 4 правилами, зашифровываем пары символов исходного текста:

1. Если два символа биграммы совпадают (или если остался один символ), добавляем после первого символа «Х», зашифровываем новую пару символов и продолжаем. В некоторых вариантах шифра Плейфера вместо «Х» используется «Q».

2. Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.

3. Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящиеся непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.

4. Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

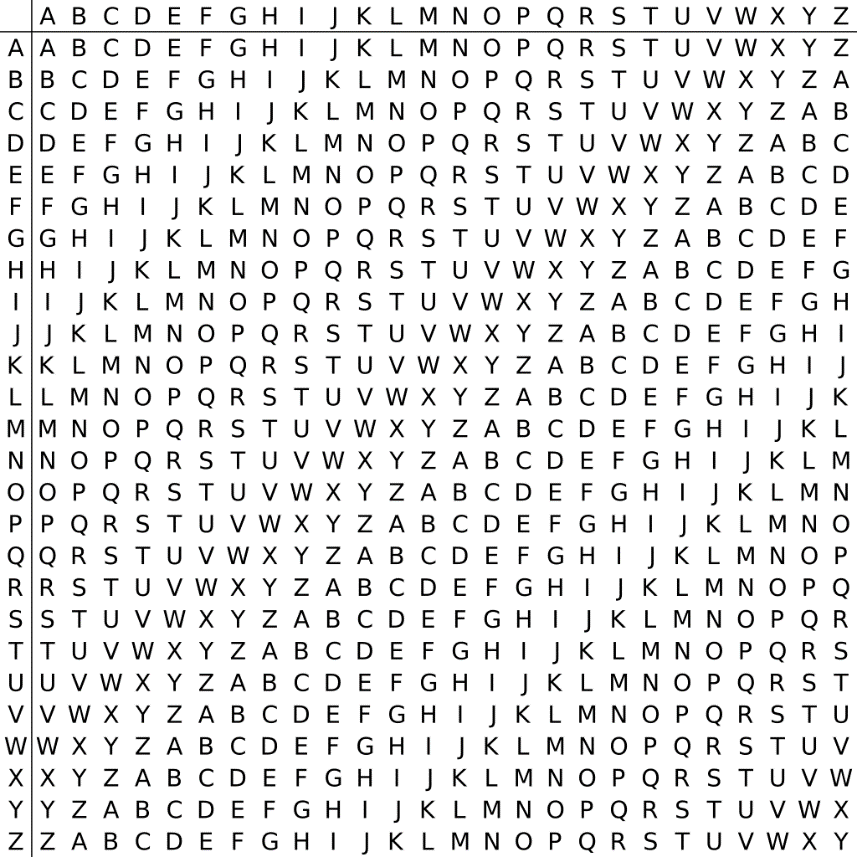
Для расшифровки необходимо использовать инверсию этих четырёх правил, откидывая символы «Х» (или «Q»), если они не несут смысла в исходном сообщении.

**Шифр Вижинера**

В [шифре Цезаря](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F) каждая буква алфавита сдвигается на несколько строк; например в шифре Цезаря при сдвиге +3, A стало бы D, B стало бы E и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет вид: ATTACKATDOWN

Человек, посылающий сообщение, записывает ключевое слово («LEMON») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста: LEMONLEMONLE

Таблица 5.4 – Таблица шифрозамен для шифра Вижинера



Первый символ исходного текста A зашифрован последовательностью L, которая является первым символом ключа. Первый символ L шифрованного текста находится на пересечении строки L и столбца A в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; то есть второй символ шифрованного текста X получается на пересечении строки E и столбца T. Остальная часть исходного текста шифруется подобным способом.

# **Задание для выполнения**

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.

**Шифр Цезаря**

Необходимо зашифровать сообщение «Кривенчук Максим Игоревич» с помощью шифра Цезаря и ключа k=11. Исходный алфавит русского языка представлен в таблице 7.2.

Таблица 7.2 – алфавит русского языка и его смещение на один символ

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

Следовательно, сообщение «Кривенчук Максим Игоревич», зашифрованное с использованием шифра Цезаря и заданного ключа k=11, будет иметь следующий вид: «Хыунршвюх Члхьуч Уощырнув».

Шифр Цезаря на языке C#:

using System;

class CaesarCipherExample

{

static void Main()

{

string message = "Кривенчук Максим Игоревич";

int key = 11;

string encryptedMessage = EncryptCaesar(message, key);

Console.WriteLine("Сообщение:" + message);

Console.WriteLine("Шифр Цезаря: " + encryptedMessage);

Console.ReadLine();

}

static string EncryptCaesar(string message, int key)

{

string encryptedMessage = "";

foreach (char character in message)

{

if (char.IsLetter(character))

{

char encryptedCharacter = EncryptCharacter(character, key);

encryptedMessage += encryptedCharacter;

}

else

{

encryptedMessage += character;

}

}

return encryptedMessage;

}

static char EncryptCharacter(char character, int key)

{

char baseCharacter = char.IsUpper(character) ? 'А' : 'а';

int alphabetSize = 32;

int offset = character - baseCharacter;

int encryptedOffset = (offset + key) % alphabetSize;

char encryptedCharacter = (char)(baseCharacter + encryptedOffset);

return encryptedCharacter;

}

}



**Шифр Трисемуса**

Необходимо зашифровать сообщение «Кривенчук Максим Игоревич» с помощью шифра Трисемуса и ключевого слова «Защита».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| з | а | щ | и | т | б |
| в | г | д | е | ё | ж |
| й | к | л | м | н | о |
| п | р | с | т | у | ф |
| х | ц | ч | ш | ъ | ы |
| ь | э | ю | я | - | - |

Таблица 7.3 – Таблица Трисумеса

Каждая буква открытого сообщения заменяется буквой, расположенной под ней в том же столбце. Если буква находится в последней строке таблицы, то для ее шифрования берут самую верхнюю букву столбца. Следовательно, сообщение «Кривенчук Максим Игоревич», зашифрованное с использованием шифра Трисемуса и заданного ключевого слова «Защита», будет иметь следующий вид: «Рцеймуюър Тгрчет Екфцмйею».

Шифр Трисемуса на языке C#:

using System;

using System.Text;

class TrisemusCipherExample

{

static void Main()

{

string message = "Кривенчук Максим Игоревич";

string encryptedMessage = EncryptTrisemus(message);

Console.WriteLine("Сообщение: " + message);

Console.WriteLine("Шифр Трисемуса: " + encryptedMessage);

Console.ReadLine();

}

static string EncryptTrisemus(string message)

{

char[,] table = new char[6, 6]

{

{ 'з', 'а', 'щ', 'и', 'т', 'б' },

{ 'в', 'г', 'д', 'е', 'ё', 'ж' },

{ 'й', 'к', 'л', 'м', 'н', 'о' },

{ 'п', 'р', 'с', 'т', 'у', 'ф' },

{ 'х', 'ц', 'ч', 'ш', 'ъ', 'ы' },

{ 'ь', 'э', 'ю', 'я', '-', '-' }

};

StringBuilder encryptedMessage = new StringBuilder();

string[] words = message.Split(' ');

foreach (string word in words)

{

StringBuilder encryptedWord = new StringBuilder();

foreach (char character in word.ToLower())

{

if (char.IsLetter(character))

{

int rowIndex = -1;

int columnIndex = -1;

// Находим индексы символа в таблице

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 6; j++)

{

if (table[i, j] == character)

{

rowIndex = i;

columnIndex = j;

break;

}

}

if (rowIndex != -1)

break;

}

// Проверяем, что символ найден в таблице

if (rowIndex != -1 && columnIndex != -1)

{

// Вычисляем индексы шифрованного символа

int encryptedRowIndex = (rowIndex + 1) % 6;

int encryptedColumnIndex = columnIndex;

char encryptedCharacter = table[encryptedRowIndex, encryptedColumnIndex];

encryptedWord.Append(encryptedCharacter);

}

}

else

{

encryptedWord.Append(character);

}

}

// Сохраняем первую букву слова в исходном регистре

if (encryptedWord.Length > 0)

{

char firstCharacter = word[0];

encryptedWord[0] = char.IsUpper(firstCharacter) ? char.ToUpper(encryptedWord[0]) : char.ToLower(encryptedWord[0]);

}

encryptedMessage.Append(encryptedWord.ToString() + " ");

}

return encryptedMessage.ToString().TrimEnd();

}

}



**Шифр Плейфейра**

Необходимо зашифровать сообщение «Кривенчук Максим Игоревич» с помощью шифра Плейфейра и ключевого слова «Защита». Промежуточная биграмма имеет «кр-ив-ен-чу-къ-ма-кс-им-иг-ор-ев-ич».

Таблица 7.4 -Таблица Плейфейра

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | - | , | . | ! |

Следовательно, сообщение «Кривенчук Максим Игоревич», зашифрованное с использованием шифра Плейфейра и заданного ключевого слова «Защита», будет иметь следующий вид: «рч-зе-нф-ър-мч-лщ-лр-щн-ае-кх-жг-зы».

Шифр Плейфейра на языке C#:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

char[] alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ ,.!".ToCharArray();

string text;

int i\_first = 0, j\_first = 0;

int i\_second = 0, j\_second = 0;

string s1 = "", s2 = "";

string encodetString;

int rows = 0, columns = 0;

int i, j;

text = "";

encodetString = "";

bool isValidTable;

do

{

Console.Write("Количество колонок в таблице: ");

isValidTable = int.TryParse(Console.ReadLine(), out columns) && columns > 1;

if (!isValidTable)

{

Console.WriteLine("Необходимо ввести число больше 1");

}

else

{

rows = alphabet.Length / columns;

isValidTable &= rows > 1 && rows \* columns == alphabet.Length;

if (!isValidTable)

{

Console.WriteLine("Необходимо ввести число колонок таким образом, чтобы число строк таблицы было больше 1 и таблица могла вмещать в себе все символы алфавита");

}

}

}

while (!isValidTable);

char[] keyWord;

bool isValidKeyWord;

do

{

Console.Write("Введите ключевое слово: ");

keyWord = Console.ReadLine().ToUpper().Distinct().ToArray();

isValidKeyWord = keyWord.Length > 0 && keyWord.Length <= alphabet.Length;

if (!isValidKeyWord)

{

Console.WriteLine("Ключевое слово не может быть пустой строкой или содержать число уникальных символов больше размера алфавита");

}

else

{

isValidKeyWord = !keyWord.Except(alphabet).Any();

if (!isValidKeyWord)

{

Console.WriteLine("Ключевое слово не может содержать символы, которых нет в алфавите");

}

}

}

while (!isValidKeyWord);

var table = new char[rows, columns];

for (i = 0; i < keyWord.Length; i++)

{

table[i / columns, i % columns] = keyWord[i];

}

alphabet = alphabet.Except(keyWord).ToArray();

for (i = 0; i < alphabet.Length; i++)

{

int position = i + keyWord.Length;

table[position / columns, position % columns] = alphabet[i];

}

for (i = 0; i < rows; i++)

{

for (j = 0; j < columns; j++)

{

Console.Write(table[i, j] + " ");

}

Console.WriteLine();

}

Console.WriteLine("Введите текст для зашифровки:");

text = Console.ReadLine().ToUpper();

int t = text.Length;

int temp = t % 2;

if (temp != 0)

{

text = text.PadRight((t + 1), 'Я');

}

int len = text.Length / 2;

string[] str = new string[len];

int l = -1;

for (i = 0; i < t; i += 2)

{

l++;

if (l < len)

{

str[l] = Convert.ToString(text[i]) + Convert.ToString(text[i + 1]);

}

}

foreach (string both in str)

{

for (i = 0; i < rows; i++)

{

for (j = 0; j < columns; j++)

{

if (both[0] == (table[i, j]))

{

i\_first = i;

j\_first = j;

}

if (both[1] == (table[i, j]))

{

i\_second = i;

j\_second = j;

}

}

}

if (i\_first == i\_second)

{

if (j\_first == columns - 1)

{

s1 = Convert.ToString(table[i\_first, 0]);

}

else

{

s1 = Convert.ToString(table[i\_first, j\_first + 1]);

}

if (j\_second == columns - 1)

{

s2 = Convert.ToString(table[i\_second, 0]);

}

else

{

s2 = Convert.ToString(table[i\_second, j\_second + 1]);

}

}

if (j\_first == j\_second)

{

if (i\_first == rows - 1)

{

s1 = Convert.ToString(table[0, j\_first]);

}

else

{

s1 = Convert.ToString(table[i\_first + 1, j\_first]);

}

if (i\_second == rows - 1)

{

s2 = Convert.ToString(table[0, j\_second]);

}

else

{

s2 = Convert.ToString(table[i\_second + 1, j\_second]);

}

}

if (i\_first != i\_second && j\_first != j\_second)

{

s1 = Convert.ToString(table[i\_first, j\_second]);

s2 = Convert.ToString(table[i\_second, j\_first]);

}

if (s1 == s2)

{

encodetString = encodetString + s1 + "я" + s2;

}

else

{

encodetString = encodetString + s1 + s2;

}

Console.WriteLine(encodetString.ToLower());

}

Console.ReadKey();

}

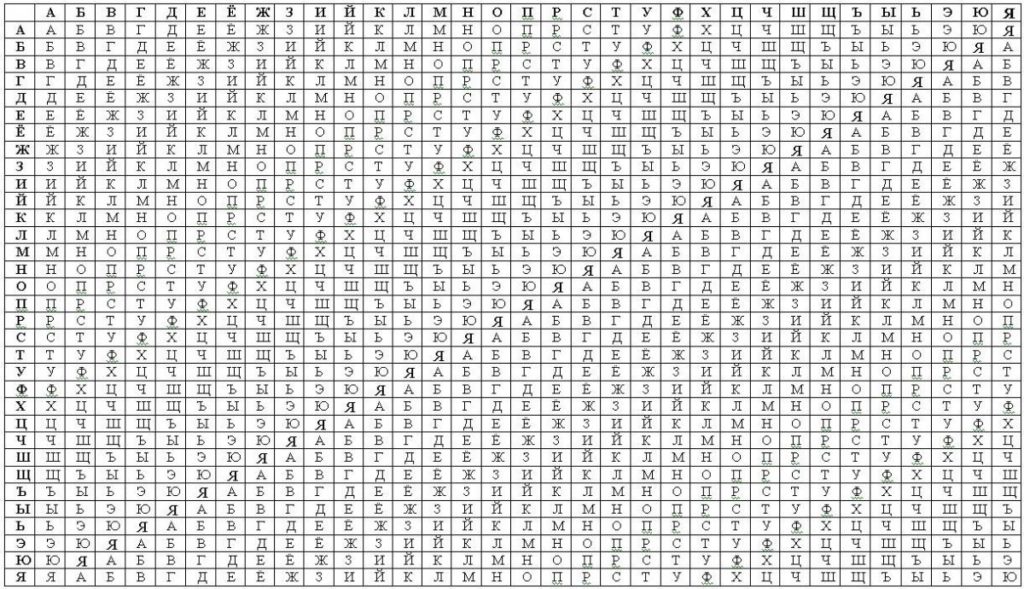
}}



**Шифр Вижинера**

Необходимо зашифровать сообщение «Кривенчук Максим Игоревич» с помощью шифра Вижинера и ключевого слова «Защита». Промежуточный ключ имеет вид «защита-защита-защита-защит» (человек, посылающий сообщение, записывает ключевое слово («защита») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста).

Таблица 7.5-Таблица Виженера



Следовательно, сообщение «Кривенчук Максим Игоревич» защита-защита-защита-защит, зашифрованное с использованием шифра Виженера и заданного ключевого слова «Защита», будет иметь следующий вид: «Трвкчняуд Яатсвх Икойнфия».

Шифр Виженера на языке C#:

using System;

public class VigenereCipher

{

const string defaultAlphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

readonly string letters;

public VigenereCipher(string alphabet = null)

{

letters = string.IsNullOrEmpty(alphabet) ? defaultAlphabet : alphabet;

}

private string GetRepeatKey(string s, int n)

{

var p = s;

while (p.Length < n)

{

p += p;

}

return p.Substring(0, n);

}

private string Vigenere(string text, string password, bool encrypting = true)

{

var gamma = GetRepeatKey(password, text.Length);

var retValue = "";

var q = letters.Length;

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

{

var letterIndex = letters.IndexOf(text[i]);

var codeIndex = letters.IndexOf(gamma[i]);

if (letterIndex < 0)

{

retValue += text[i].ToString();

}

else

{

retValue += letters[(q + letterIndex + ((encrypting ? 1 : -1) \* codeIndex)) % q].ToString();

}

}

return retValue;

}

public string Encrypt(string plainMessage, string password)

=> Vigenere(plainMessage, password);

public string Decrypt(string encryptedMessage, string password)

=> Vigenere(encryptedMessage, password, false);

}

class Program

{

static void Main(string[] args)

{

var cipher = new VigenereCipher("АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");

Console.Write("Введите текст: ");

var inputText = Console.ReadLine().ToUpper();

Console.Write("Введите ключ: ");

var password = Console.ReadLine().ToUpper();

var encryptedText = cipher.Encrypt(inputText, password);

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

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

Console.ReadLine();

}

}



# **Индивидуальное задание**

Расшифровать следующие сообщения.

|  |  |  |
| --- | --- | --- |
| № варианта | Сообщение | Способ |
| 11 | **гг\_ \_ ип-оомитрдр\_ос,яубогп\_**  **мбадираоитнла\_гноаавуоа** | Расшифровать с помощью **Метода простой перестановки Таблица7х7** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| г | г | \_ | \_ | и | п | - |
| о | о | м | и | т | р | д |
| р | \_ | о | с | , | я | у |
| б | о | г | п | \_ | м | б |
| а | д | и | р | а | о | и |
| т | н | л | а | \_ | г | н |
| о | а | а | в | у | о | а |

Метод простой перестановки Таблицы7х7на языке C#:

using System;

class SimpleSubstitutionDecryptor

{

static void Main()

{

string encryptedText = "гг\_\_ип-оомитрдр\_ос,яубогп\_мбадираоитнла\_гноаавуоа";

char[,] table = CreateTable(encryptedText);

PrintWords(table);

Console.ReadLine();

}

static char[,] CreateTable(string encryptedText)

{

int rowCount = 7;

int columnCount = (int)Math.Ceiling((double)encryptedText.Length / rowCount);

char[,] table = new char[rowCount, columnCount];

int index = 0;

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

{

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

{

if (index < encryptedText.Length)

{

if (encryptedText[index] != '\_')

{

table[i, j] = encryptedText[index];

}

else

{

table[i, j] = ' ';

}

index++;

}

else

{

table[i, j] = ' ';

}

}

}

return table;

}

static void PrintWords(char[,] table)

{

int rowCount = table.GetLength(0);

int columnCount = table.GetLength(1);

string currentWord = "";

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

{

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

{

char character = table[i, j];

if (character != ' ')

{

currentWord += character;

}

else if (!string.IsNullOrWhiteSpace(currentWord))

{

Console.Write(currentWord + " ");

currentWord = "";

}

}

}

if (!string.IsNullOrWhiteSpace(currentWord))

{

Console.Write(currentWord);

}

}

}



Результат: горбатого одна могила исправит, а упрямого-дубина.

**Вывод:** в ходе лабораторной работы были изучены основные криптографические алгоритмы симметричного шифрования.

**Практическое задание №4.2**

**Тема «Криптографическая защита информации»**

Цель: Овладение основными криптографическими алгоритмами асимметричного шифрования.

**Задание №1)** Процесс работы алгоритма RSA

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.  
 Работу алгоритма можно разделить на 3 этапа: генерация ключей, шифрование и расшифровка.

**Генерация ключей.** Выбираем 2 простых числа p и q 🡺 вычисляем произведение этих чисел n 🡺 вычисляем функцию Эйлера : f(n) = (p-1) \* (q-1) 🡺 выбираем открытую экспоненту е = 3 🡺 вычисляем секретную экспоненту d = e^(-1) mod f(n) 🡺 публикуем открытый ключ {e, n} 🡺 сохраняем закрытый ключ {d,n}

**Шифрование.** Выбираем текст для зашифровки (двоичный формат) m 🡺 Вычисляем шифротекст : m^e mod n

**Расшифровка.** Пусть с – сообщение 🡺 с^d mod n

Вышеперечисленное в табличной форме записи:

|  |  |  |
| --- | --- | --- |
| 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} |

Таблица 1 – Алгоритм RSA

**Задание №2)** процесс работы алгоритма Диффи-Хеллмана.

В 1976 Уитфилд Диффи (Whitfield Diffie) и Мартин Хеллман (Martin Hellman) опубликовали свой алгоритм обмена ключами. Это была первая публикация на тему криптографии с открытым ключом и, возможно, самый большой шаг вперед в области криптографии, сделанный когда‑либо. Он был ориентирован на передачу секретных ключей DES, ARS или других подобных алгоритмов через небезопасную среду. В большинстве случаев алгоритм Диффи‑Хеллмана не используется для шифрования сообщений, потому что он, в зависимости от реализации, от 10 до 1000 раз медленнее алгоритма DES.

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.

**Самое сложное в алгоритме** Диффи‑Хеллмана обмена ключами – это понять, что в нем фактически два различных независимых цикла шифрования. Алгоритм Диффи‑Хеллмана применяется для обработки небольших сообщений от отправителя получателю. Но в этом маленьком сообщении передается секретный ключ для расшифровки большого сообщения.

**Сильная сторона алгоритма** - никто не сможет скомпрометировать секретное сообщение, зная один или даже два открытых ключа получателя и отправителя. В качестве секретных и открытых ключей используются очень большие целые числа. Алг оритм Диффи‑Хеллмана основан на полезных для криптографии свойствах дискретных логарифмов.

**Задание №3)** процесс работы алгоритма Эль-Гамаля.

## **Генерация ключей**

1. Генерируется случайное простое число ~p длины ~n битов.
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**

Сообщение ~M шифруется следующим образом:

1. Выбирается сессионный ключ — случайное целое число ~k такое, что ~1 < k < p - 1
2. Вычисляются числа a = g^k\,\bmod\,p и b = y^k M\,\bmod\,p.
3. Пара чисел \left( a, b \right) является шифротекстом.

Нетрудно видеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

## **Расшифрование**

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

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

При этом нетрудно проверить, что

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

и поэтому

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

Для практических вычислений больше подходит следующая формула:

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

## **Пример**

**Шифрование**

Допустим, что нужно зашифровать сообщение ~M=5.

Произведем генерацию ключей :

пусть ~p=11, g=2. Выберем ~x=8 - случайное целое число ~x такое, что ~1 < x < p.

Вычислим ~y= g^x\bmod{p}=2^8\bmod{11}=3.

Итак, открытым является тройка ~(p,g,y)=(11,2,3), а закрытым ключом является число ~x=8.

Выбираем случайное целое число ~k такое, что 1 < k < (p − 1). Пусть ~k=9.

Вычисляем число ~a=g^k\bmod{p}=2^9 \bmod{11}=512 \bmod{11}=6.

Вычисляем число ~b=y^k M\bmod{p}=3^9 5 \bmod{11}=19683 \cdot 5 \bmod{11}=9.

Полученная пара ~(a,b)=(6,9) является шифротекстом.

**Расшифрование**

Необходимо получить сообщение ~M=5 по известному шифротексту ~(a,b)=(6,9) и закрытому ключу ~x=8.

Вычисляем M по формуле : ~M=b(a^x)^{-1}\bmod{p}=9(6^8)^{-1}\mod{11}=5

Получили исходное сообщение ~M=5.

**Задание №4\***

Используя существующие криптографические библиотеки, создать приложение и проанализировать работу вышеперечисленных алгоритмов

using System;

using System.Security.Cryptography;

class Program

{

static void Main()

{

string fullName = "Krivenchuk Maxim";

// Генерация ключей RSA

using (var rsa = new RSACryptoServiceProvider())

{

// Шифрование ФИО с использованием RSA

byte[] encryptedData = RSAEncrypt(fullName, rsa.ExportParameters(false));

// Дешифрование ФИО с использованием RSA

string decryptedData = RSADecrypt(encryptedData, rsa.ExportParameters(true));

Console.WriteLine("Шифрование и дешифрование ФИО с использованием RSA:");

Console.WriteLine("Исходное ФИО: " + fullName);

Console.WriteLine("Зашифрованное ФИО: " + Convert.ToBase64String(encryptedData));

Console.WriteLine("Расшифрованное ФИО: " + decryptedData);

Console.WriteLine();

}

// Генерация ключей Диффи-Хеллмана

using (var dh = new ECDiffieHellmanCng())

{

// Шифрование ФИО с использованием Диффи-Хеллмана

byte[] encryptedData = DHEncrypt(fullName, dh);

// Дешифрование ФИО с использованием Диффи-Хеллмана

string decryptedData = DHDecrypt(encryptedData, dh);

Console.WriteLine("Шифрование и дешифрование ФИО с использованием Диффи-Хеллмана:");

Console.WriteLine("Исходное ФИО: " + fullName);

Console.WriteLine("Зашифрованное ФИО: " + Convert.ToBase64String(encryptedData));

Console.WriteLine("Расшифрованное ФИО: " + decryptedData);

Console.WriteLine();

}

// Генерация ключей Эль-Гамаля

using (var elGamal = new ECDiffieHellmanCng())

{

// Шифрование ФИО с использованием Эль-Гамаля

byte[] encryptedData = ElGamalEncrypt(fullName, elGamal);

// Дешифрование ФИО с использованием Эль-Гамаля

string decryptedData = ElGamalDecrypt(encryptedData, elGamal);

Console.WriteLine("Шифрование и дешифрование ФИО с использованием Эль-Гамаля:");

Console.WriteLine("Исходное ФИО: " + fullName);

Console.WriteLine("Зашифрованное ФИО: " + Convert.ToBase64String(encryptedData));

Console.WriteLine("Расшифрованное ФИО: " + decryptedData);

}

Console.ReadLine();

}

// Метод для шифрования ФИО с использованием RSA

static byte[] RSAEncrypt(string data, RSAParameters rsaParameters)

{

using (var rsa = new RSACryptoServiceProvider())

{

rsa.ImportParameters(rsaParameters);

return rsa.Encrypt(System.Text.Encoding.UTF8.GetBytes(data), true);

}

}

// Метод для дешифрования ФИО с использованием RSA

static string RSADecrypt(byte[] encryptedData, RSAParameters rsaParameters)

{

using (var rsa = new RSACryptoServiceProvider())

{

rsa.ImportParameters(rsaParameters);

byte[] decryptedData = rsa.Decrypt(encryptedData, true);

return System.Text.Encoding.UTF8.GetString(decryptedData);

}

}

// Метод для шифрования ФИО с использованием Диффи-Хеллмана

static byte[] DHEncrypt(string data, ECDiffieHellmanCng dh)

{

byte[] publicKey = dh.PublicKey.ToByteArray();

byte[] sharedKey = dh.DeriveKeyMaterial(CngKey.Import(publicKey, CngKeyBlobFormat.EccPublicBlob));

byte[] encryptedData;

using (var aes = new AesCryptoServiceProvider())

{

aes.Key = sharedKey;

aes.GenerateIV();

using (var encryptor = aes.CreateEncryptor())

using (var memoryStream = new System.IO.MemoryStream())

{

// Записываем IV в начало потока

memoryStream.Write(aes.IV, 0, aes.IV.Length);

using (var cryptoStream = new System.Security.Cryptography.CryptoStream(memoryStream, encryptor, System.Security.Cryptography.CryptoStreamMode.Write))

using (var streamWriter = new System.IO.StreamWriter(cryptoStream))

{

streamWriter.Write(data);

}

encryptedData = memoryStream.ToArray();

}

}

return encryptedData;

}

// Метод для дешифрования ФИО с использованием Диффи-Хеллмана

static string DHDecrypt(byte[] encryptedData, ECDiffieHellmanCng dh)

{

byte[] publicKey = dh.PublicKey.ToByteArray();

byte[] sharedKey = dh.DeriveKeyMaterial(CngKey.Import(publicKey, CngKeyBlobFormat.EccPublicBlob));

byte[] iv = new byte[16];

using (var aes = new AesCryptoServiceProvider())

{

aes.Key = sharedKey;

// Читаем IV из начала зашифрованных данных

Array.Copy(encryptedData, iv, iv.Length);

aes.IV = iv;

using (var decryptor = aes.CreateDecryptor())

using (var memoryStream = new System.IO.MemoryStream(encryptedData, iv.Length, encryptedData.Length - iv.Length))

using (var cryptoStream = new System.Security.Cryptography.CryptoStream(memoryStream, decryptor, System.Security.Cryptography.CryptoStreamMode.Read))

using (var streamReader = new System.IO.StreamReader(cryptoStream))

{

return streamReader.ReadToEnd();

}

}

}

// Метод для шифрования ФИО с использованием Эль-Гамаля

static byte[] ElGamalEncrypt(string data, ECDiffieHellmanCng elGamal)

{

byte[] publicKey = elGamal.PublicKey.ToByteArray();

byte[] sharedKey = elGamal.DeriveKeyMaterial(CngKey.Import(publicKey, CngKeyBlobFormat.EccPublicBlob));

byte[] encryptedData;

using (var aes = new AesCryptoServiceProvider())

{

aes.Key = sharedKey;

aes.GenerateIV();

using (var encryptor = aes.CreateEncryptor())

using (var memoryStream = new System.IO.MemoryStream())

{

// Записываем IV в начало потока

memoryStream.Write(aes.IV, 0, aes.IV.Length);

using (var cryptoStream = new System.Security.Cryptography.CryptoStream(memoryStream, encryptor, System.Security.Cryptography.CryptoStreamMode.Write))

using (var streamWriter = new System.IO.StreamWriter(cryptoStream))

{

streamWriter.Write(data);

}

encryptedData = memoryStream.ToArray();

}

}

return encryptedData;

}

// Метод для дешифрования ФИО с использованием Эль-Гамаля

static string ElGamalDecrypt(byte[] encryptedData, ECDiffieHellmanCng elGamal)

{

byte[] publicKey = elGamal.PublicKey.ToByteArray();

byte[] sharedKey = elGamal.DeriveKeyMaterial(CngKey.Import(publicKey, CngKeyBlobFormat.EccPublicBlob));

byte[] iv = new byte[16];

using (var aes = new AesCryptoServiceProvider())

{

aes.Key = sharedKey;

// Читаем IV из начала зашифрованных данных

Array.Copy(encryptedData, iv, iv.Length);

aes.IV = iv;

using (var decryptor = aes.CreateDecryptor())

using (var memoryStream = new System.IO.MemoryStream(encryptedData, iv.Length, encryptedData.Length - iv.Length))

using (var cryptoStream = new System.Security.Cryptography.CryptoStream(memoryStream, decryptor, System.Security.Cryptography.CryptoStreamMode.Read))

using (var streamReader = new System.IO.StreamReader(cryptoStream))

{

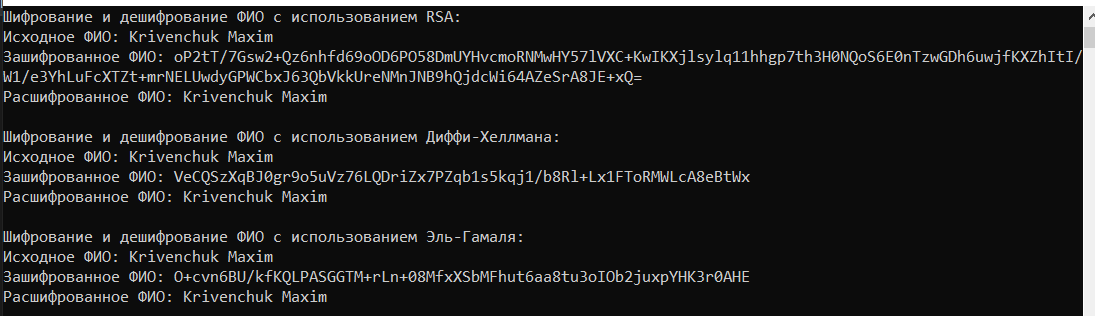
return streamReader.ReadToEnd();

}

}

}

}



**Вывод:**

В результате исследования второй части четвертой лабораторной работы были получены теоретические знания об алгоритмах RSA, Эль-Гамаля, Диффи‑Хеллмана.

**Практическое задание № 5**

**Тема «Криптографическая защита информации»**

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

**Необходимо ответить на следующие вопросы:**

1. Дайте определение понятию "электронная цифровая подпись".
2. Объясните какой порядок использования ключей (открытый; закрытый) при отправке и проверке ЭЦП.
3. Перечислите специальные схемы ЭЦП.
4. Перечислите недостатки цифровой подписи RSA

**1)** 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").

**2)** 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.**

**3)** 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;

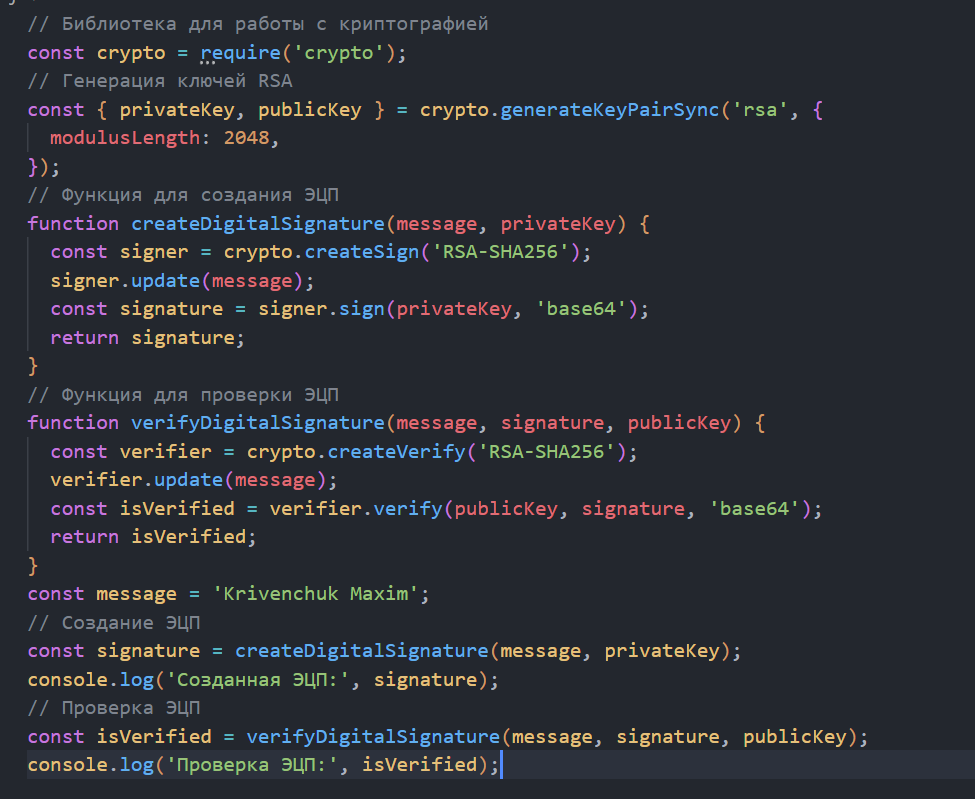
**4)** Недостатки цифровой подписи RSA:

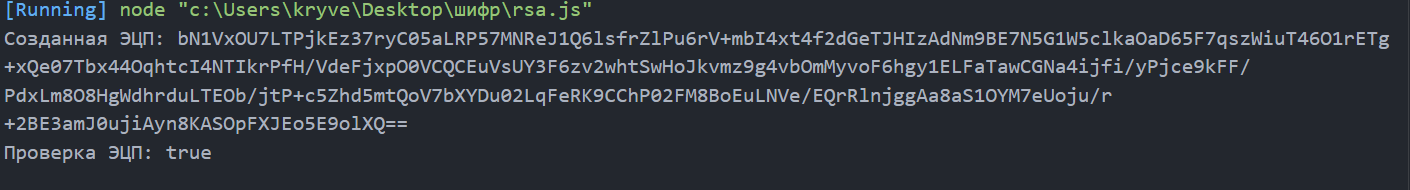
* 1. Можно повторно использовать подписанный документ, так как копирование файлов в компьютерных системах очень простая задача. Чтобы преодолеть этот недостаток, документ должен содержать, например, метку времени, а проверяющий подпись должен создать базу данных, содержащую метки времени получаемых от отправителя документов. Тогда при повторном предъявлении документа легко обнаружить попытку обмана.
* 2. Отправитель сообщения может подписать сообщение, а затем отказаться от подписи, заявив, что его секретный ключ был скомпрометирован.
* 3. Стойкость протоколов подписи RSA основана на сложности факторизации большого натурального числа *N.* В 1978 году, в момент опубликования RSA, считалось, что достаточно взять *N »* 10100. В настоящее время выбор *N* \* 10150 уже не обеспечивает защиту от подделки. Таким образом, лицо, подписавшее документ в 1978 г., может сейчас отказаться от своей подписи под этим документом. Или злоумышленник может подделывать подписи под документами, датировав их 1978 г. Такое свойство электронной подписи приводит к тому, что будущие историки не будут доверять электронным документам нашего времени.
* 4. При вычислении модуля *N,* ключей *Е* и *D* для системы цифровой подписи RSA необходимо проверять большое количество дополнительных условий, что сделать трудно практически. Невыполнение любого из этих условий делает возможным фальсификацию цифровой подписи со стороны того, кто обнаружит такое невыполнение. При подписании важных документов нельзя допускать такую возможность даже теоретически.
* 5. Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации необходимо использовать при вычислениях *N, D* и *Е* целые числа длиной не менее 512 бит, что требует больших вычислительных затрат, превышающих на 20—30% вычислительные затраты других алгоритмов цифровой подписи при сохранении того же уровня криптостойкости.
* 6. Цифровая подпись RSA уязвима к так называемой мультипликативной атаке. Иначе говоря, алгоритм цифровой подписи RSA позволяет злоумышленнику без знания секретного ключа *D* сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов.

Использованные источники:

1. Теория к практической работе №5.
2. Сайт www.bstudy.net – о недостатках цифровой подписи.

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





**Вывод:**

В ходе практической работы были получены теоретические знания об Электронной цифровой подписи: порядок использования ключей при отправке и проверке ЭЦП, определение, специальные схемы ЭЦП, недостатки алгоритма цифровой подписи ЭЦП.

|  |
| --- |
| **Практическое задание № 6** |
| **Тема «Теория чисел»** |
| Цель**:**  получение основных сведений из курса теории чисел, решение алгебраических задач.  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.  **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).*  **Задание для выполнения:**   1. Найти канонические разложения чисел а и b. 2. Найти НОД  пользуясь:   a) алгоритмом Евклида,  б) разложением чисел на простые множители.   1. С помощью расширенного алгоритма Евклида найти целые u, v, удовлетворяющие соотношению Безу: au + bv = НОД . 2. Найти остаток от деления данного числа на простое. |

Условия для варианта №11 следующие:



**Задание №1. Каноническое разложение a и b.**

**A B**

7957549 | 349 23118553 | 541

22801 | 151 42733 | 283

151 | 151 151 | 151

1 1

**Задание №2. Найти НОД(a,b)**

* 1. Алгоритмом Евклида

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(7957549, 23118553) = 151
  1. Разложением чисел на простые множители

Разложим числа 7957549 и 23118553 на простые множители

7957549 = 151 × 151 × 349

23118553 = 151 × 283 × 541

Подчеркнём общие множители

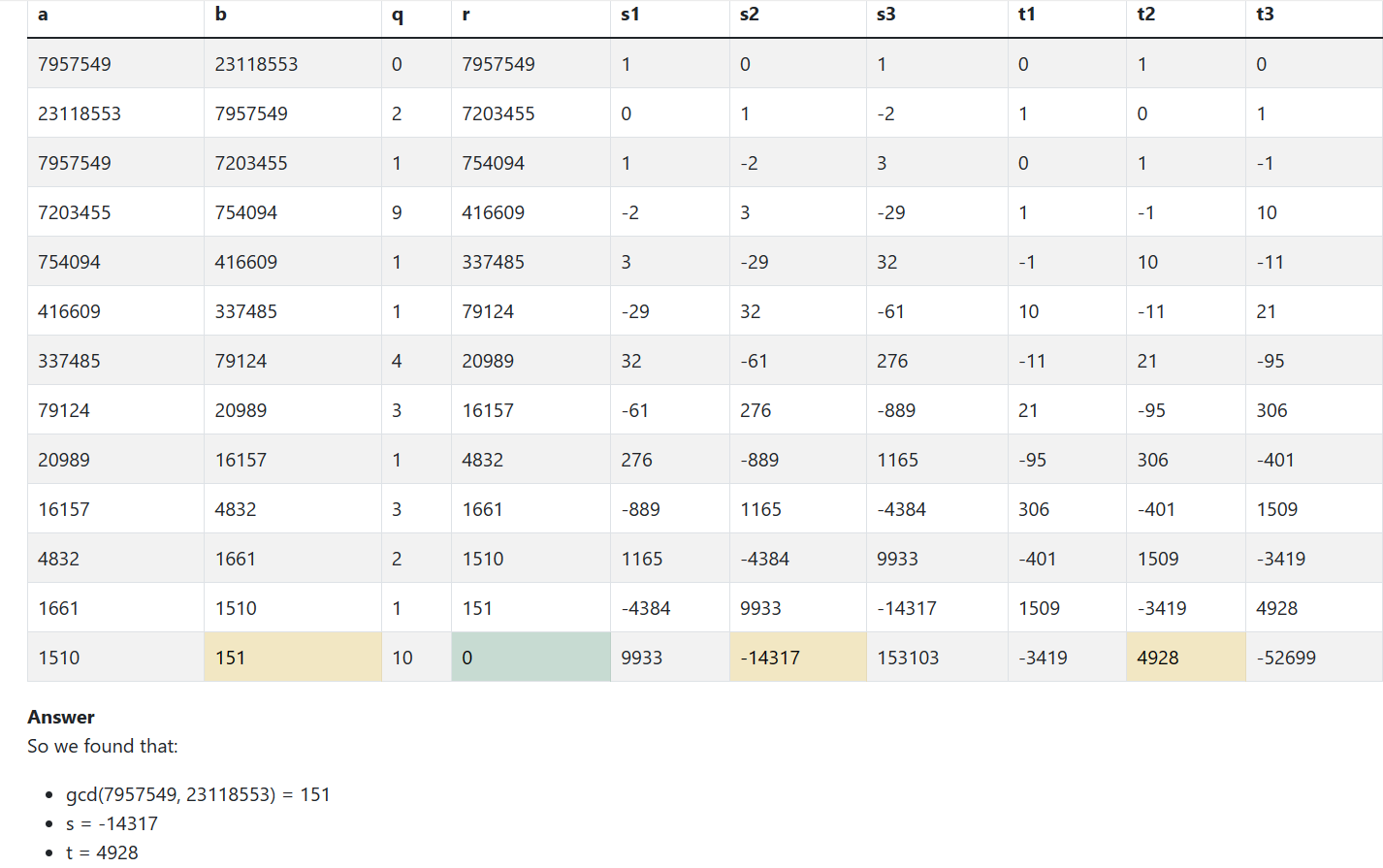
7957549 = 151 × 151 × 349

23118553 = 151 × 283 × 541

Наибольший общий делитель чисел 7957549 и 23118553

НОД(7957549, 23118553) = 151

**Задание №3. С помощью расширенного алгоритма Евклида найти целые u, v, удовлетворяющие соотношению Безу: au + bv = НОД .**



Таким образом, значения u и v, удовлетворяющие соотношению Безу, равны:

u = -14317  
v = 4928

**Задание №4. Найти остаток от деления данного числа на простое.**

Малая теорема Ферма утверждает что a^(p-1) ≡ 1(mod p)

При делении 2005 на 19 получаем остаток = 10

2005 = 105 \* 19 + 10; 2005 = 10 mod 19

f(19) = 19 \* (1 – 1/19) = 18; 1999 = 18 \* 111 + 1 => =

10 mod 19 = 10 Остаток от деления = 10;

**Вывод:**

В результате выполнения данной практической работы были повторены основные теоретические сведения математики: свойства чисел, теоремы, определения. Также были разложены 2 канонических разложения. Получены навыки высчитывания НОД(a,b) через алгоритм Евклида, а также найден способ нахождения остатка от деления одного большого числа на простое.

**Практическое задание № 7**

**Тема «**Настройка антивирусов**»**

Цель: Овладение навыками настройки и использования различных антивирусов.

# **Задание для выполнения**

1. Установить и настроить антивирусную программу.

2. Обновить базу данных сигнатур вирусов.

3. Выполнить сканирование дисков.

В рамках данной лабораторной работы был установлен антивирус Avast Free Antivirus.



Рис. 1. — Ярлык Avast Free Antivirus

Проведем проверку на вирусы, нажав соответствующую кнопку.

Описание: Изображение выглядит как текст, электроника, черный, другой

Автоматически созданное описание

Рис. 2. — Кнопка запуска проверки на вирусы

Этапы проведения проверки представлены на рисунках 3-15.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 3.

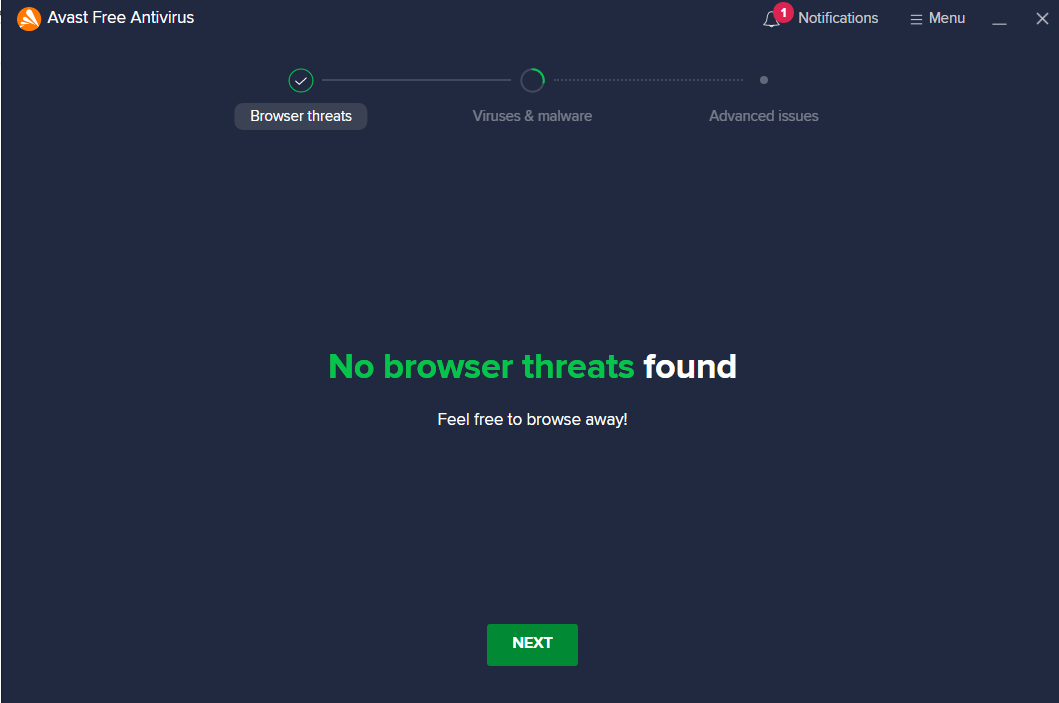


Рис. 4.

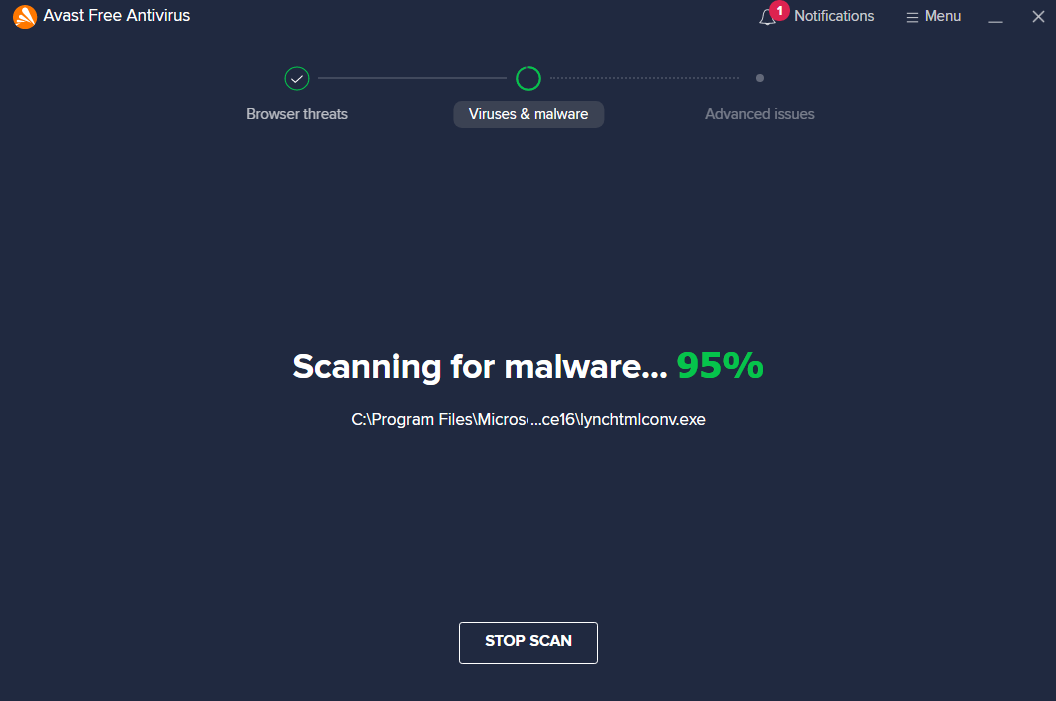


Рис. 5.

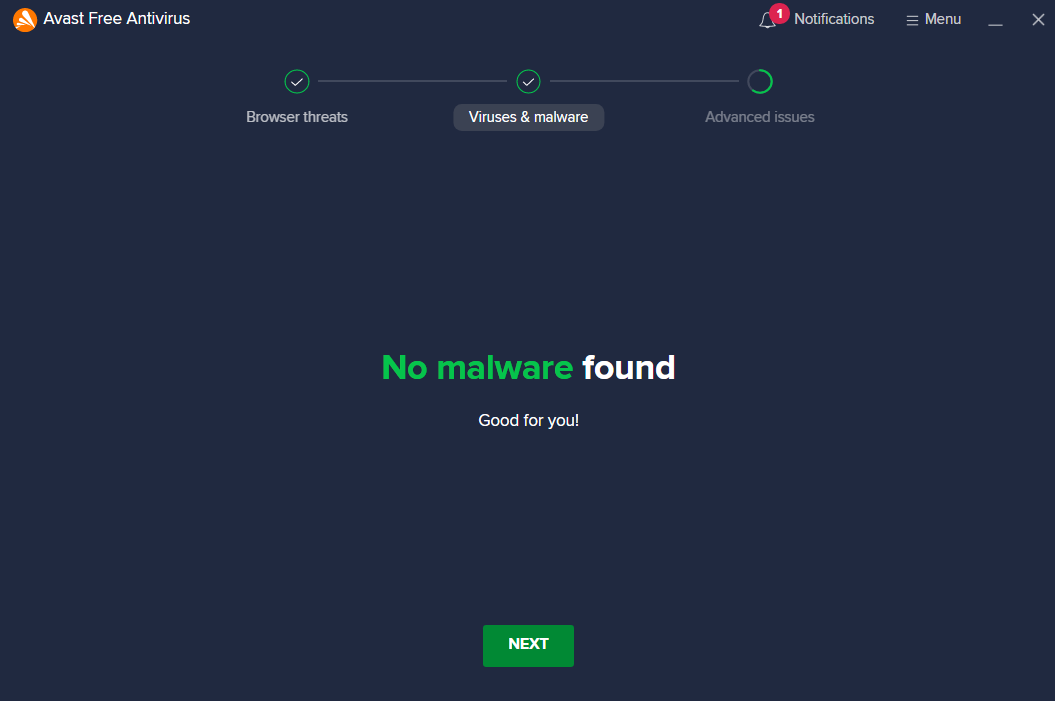


Рис. 6.

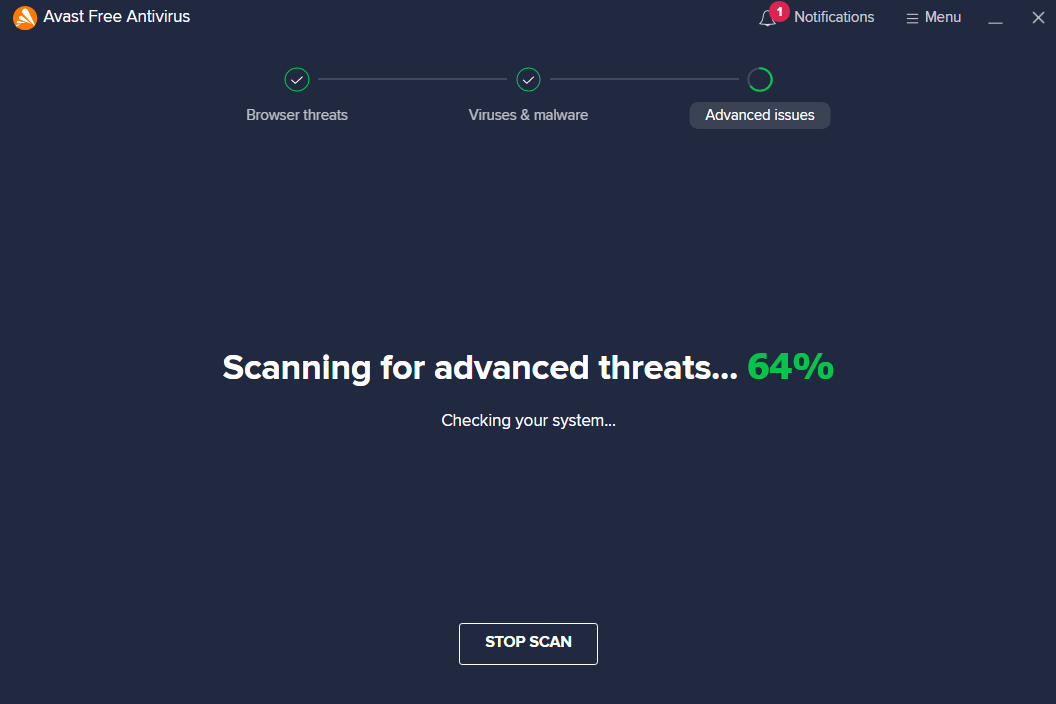


Рис. 7.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 8.

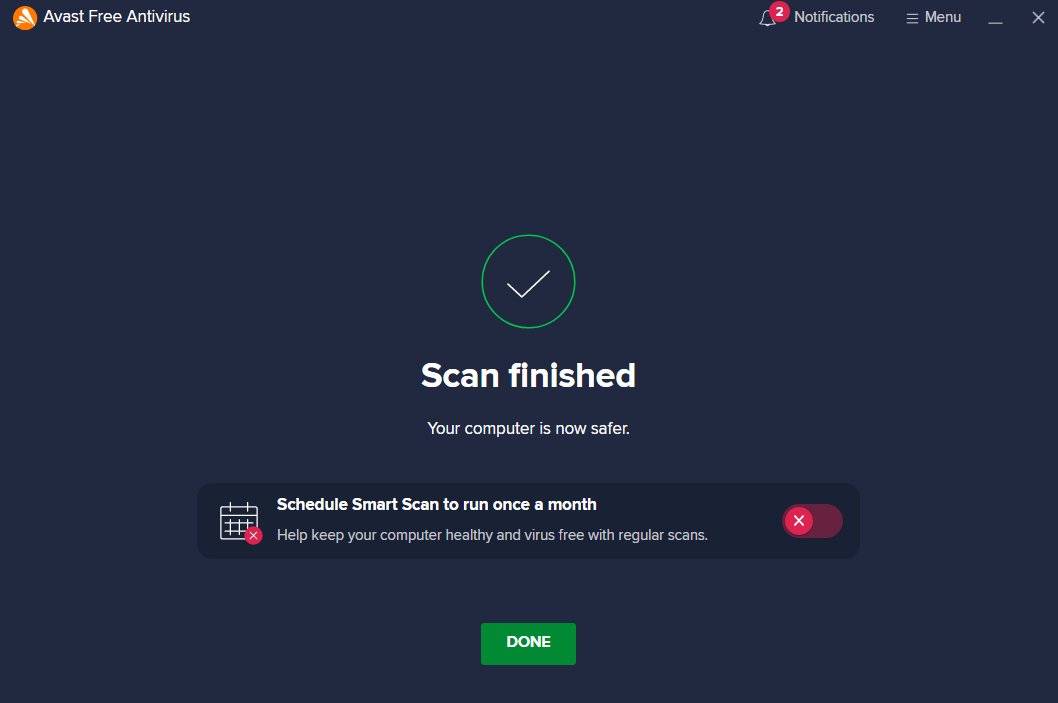


Рис. 9.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 10.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 11.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 12.

Описание: Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рис. 13.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 14.

Описание: Изображение выглядит как текст

Автоматически созданное описание

Рис. 15

Таким образом, проблемы на этом компьютере были исправлены.

**Вывод:** в ходе работы были получены навыки настройки и использования антивируса Avast.

|  |
| --- |
| **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**

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. Шифрование и дешифрование для асимметричных алгоритмов выполняется проще из за встроенных функций Encrypt и Decrypt.

Задание 11-го варианта выглядит следующим образом:



Необходимо выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам. Используя функции работы с файлами сохранить ключи шифрования, результаты шифрования и хеширования. Используя hex-редактор продемонстрировать ключи шифрования, зашифрованные и захешированные данные.

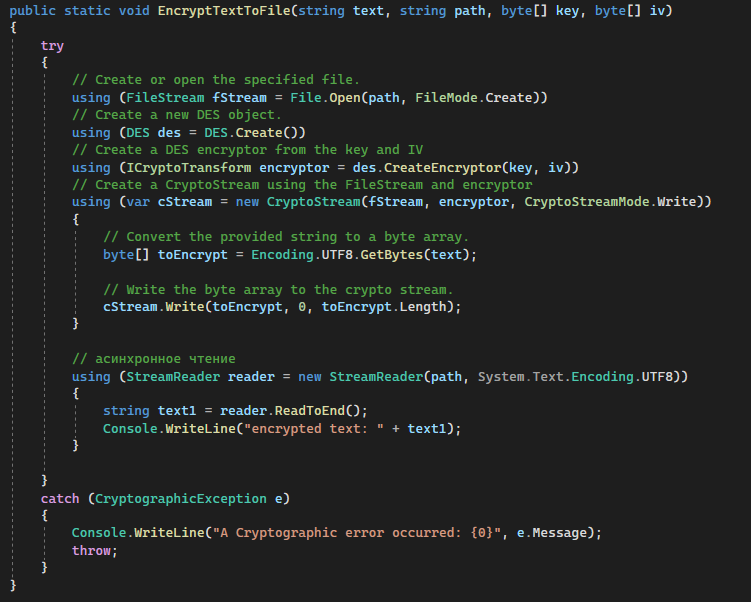
**Задание №1**

Выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам ( DES для шифрования и SHA1 ). Используя функции работы с файлами сохранить ключи шифрования, результаты шифрования и хеширования.

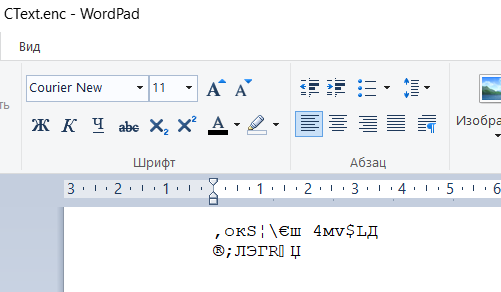
Исходный текст



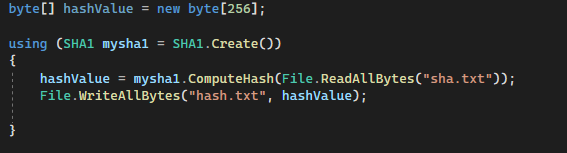
Шифрование и Дешифрование выполняется Симметричным DES Алгоритмом



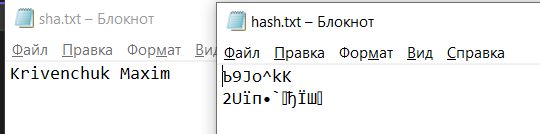
Зашифрованный текст записанный в файл



Алгоритм Хеширование SHA1

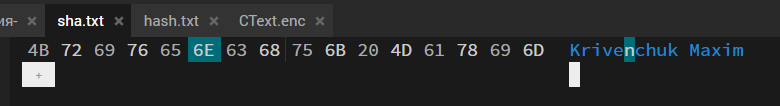
****

Исходный и захешированный текст

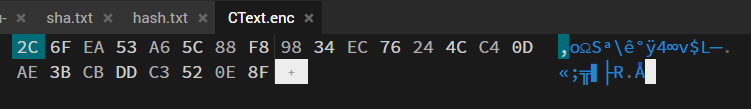


**Задание №2**

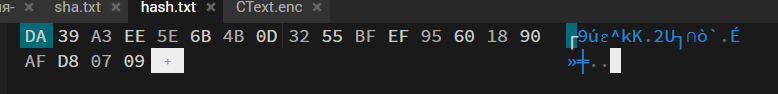
Для выше указанных алгоритмов используя Hex-редактор продемонстрировать ключи шифрования, зашифрованные и захешированные данные.



DES

****

SHA1



**Вывод:** используя встроенные в .NET методы шифровки разными алгоритмами можно написать программу, которая будет шифровать данные таким образом, каким мы хотим.

**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 интерфейс включают в себя возможность генерировать, использовать и применять пары криптографических ключей; шифровать и дешифровать сообщения; надежно генерировать случайные числа

Хеширование в SubtleCrypto представлено следующими алгоритмами.

* Sha1
* Sha256
* Sha385
* Sha512

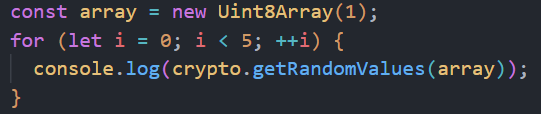
Метод SubtleCrypto.digest() используется для создания хеша сообщения.

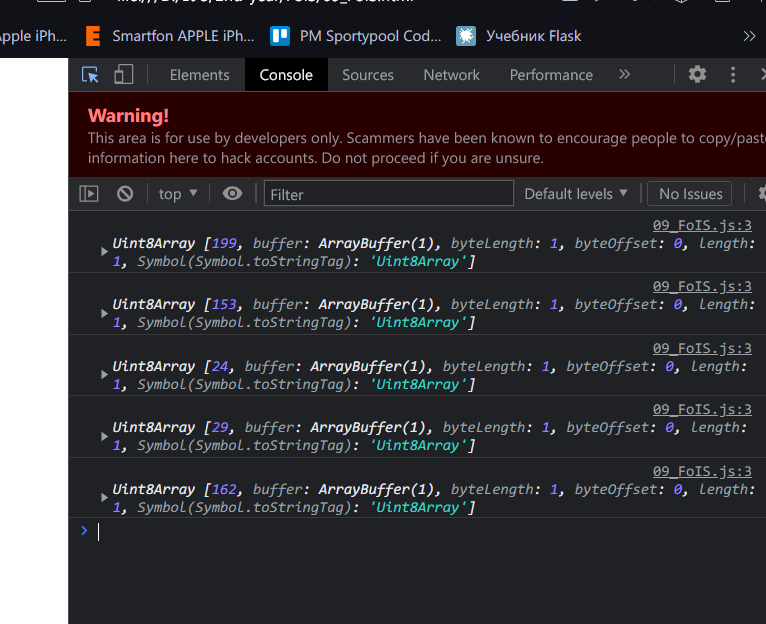
Генерация случайного CryptoKey выполняется с помощью метода SubtleCrypto.generateKey(algorithm, extractable, keyUsages). В этот метод передается объект params, указывающий целевой алгоритм, логическое значение,указывающее, должен ли ключ извлекаться из объекта CryptoKey, и массив строк — keyUsages, указывающий,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

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

**Задание №1**

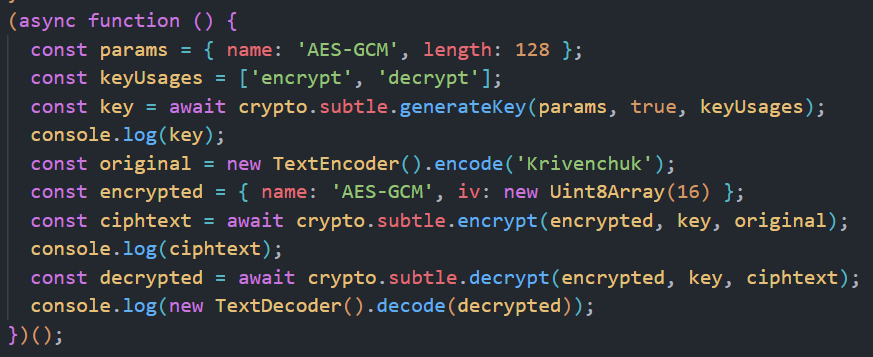
Выполнить генерацию и вывод в консоль случайный чисел.

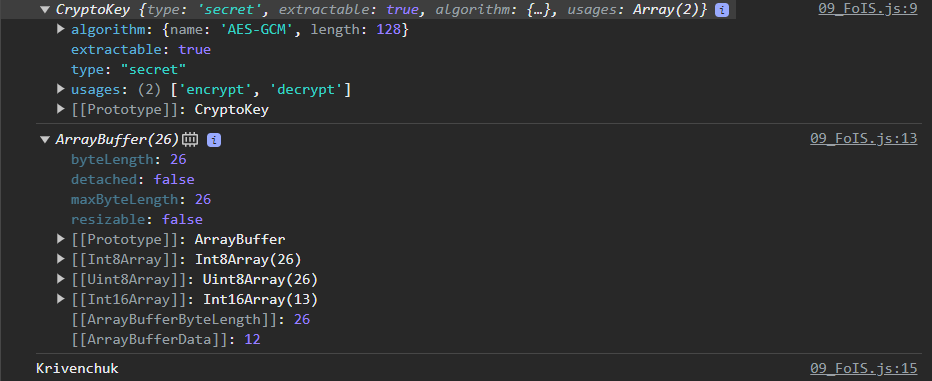


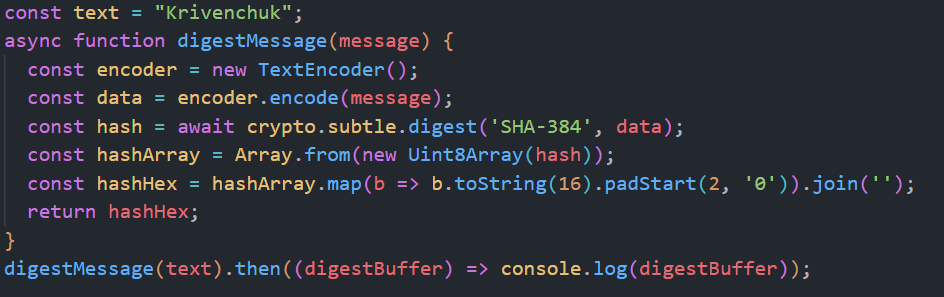
****

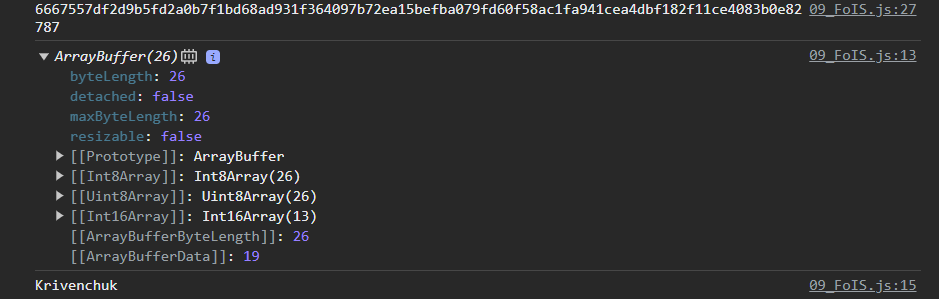
**Задание №2**

Выполнить шифрование, дешифрование и хеширование своей фамилии по указанным алгоритмам.

****

****

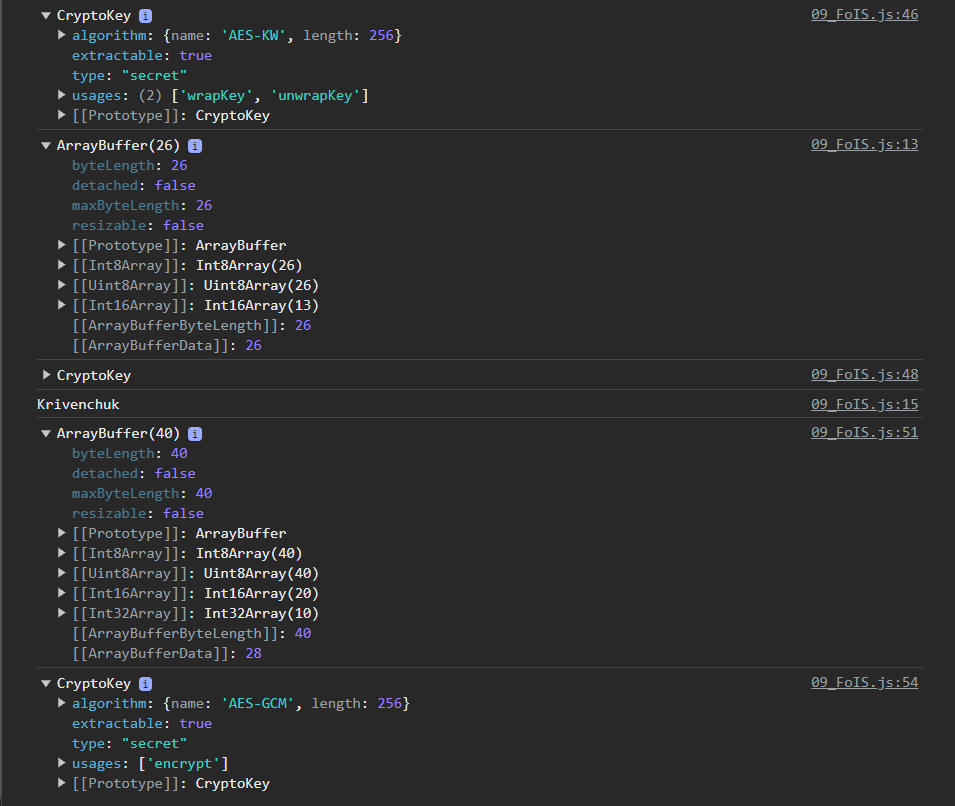
****

****

**Задание №3**

Продемонстрировать упаковку и распаковку ключа, полученного в предыдущем задании используя алгоритм AES-KW.



****

**Задание №4**

****

Рисунок 5 – Генерация подписи

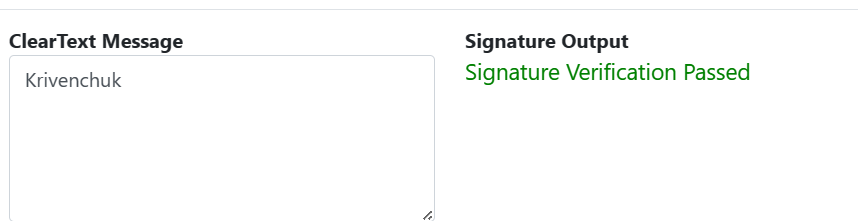
****

Рисунок 6 – Проверка подписи

**Вывод:**

По результатам практической работы были получены практические навыки хеширование, дешифрования и шифрования информации на определенных алгоритмах с помощью ЯП JavaScript, а также отработана упаковка-распаковка ключа, выполнена процедура подписи сообщения и проверку подлинности с использованием RSA-PSS. Web Crypto API – это интерфейс, позволяющий использовать криптографические примитивы для построения систем с использованием криптографии. Данный интерфейс включает в себя возможность генерировать, использовать и применять пары криптографических ключей; шифровать и дешифровать сообщения; надежно генерировать случайные числа. Существует множество алгоритмов шифрования, однако каждый из них отличается своей надёжностью. Если безопасность данных жизненно необходима, можно воспользоваться алгоритмом SHA-256. Упаковка и распаковка ключей используется для обеспечения безопасной и надёжной передачи данных по незащищённому каналу.