МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Выполнил студент 2 курса 3 группы специальность ИСиТ Халалеенко А.Н.

(Ф.И.О.)

Преподаватель старший преподаватель Ржеутская Надежда Викентьевна

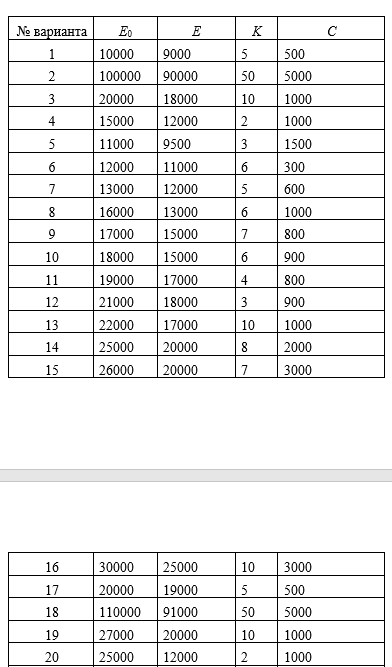
(Ф.И.О.)

**Практическая работа №1**

по дисциплине «Основы защиты информации»

на тему: 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.



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

– законодательные (правовые);

– организационные;

– технические;

– комплексные.

Для обеспечения защиты объектов информационной безопасности должны быть соответствующие правовые акты, устанавливающие порядок защиты и ответственность за его нарушение. Законы должны давать ответы на следующие вопросы: что такое информация, кому она принадлежит, как может с ней поступать собственник, что является посягательством на его права, как он имеет право защищаться, какую ответственность несет нарушитель прав собственника информации.

Установленные в законах нормы реализуются через комплекс организационных мер, проводимых прежде всего государством, ответственным за выполнение законов, и собственниками информации. К таким мерам относятся издание подзаконных актов, регулирующих конкретные вопросы по защите информации (положения, инструкции, стандарты и т. д.), и государственное регулирование сферы через систему лицензирования, сертификации, аттестации.

Поскольку в настоящее время основное количество информации генерируется, обрабатывается, передается и хранится с помощью технических средств, то для конкретной ее защиты в информационных объектах необходимы технические устройства. В силу многообразия технических средств нападения приходится использовать обширный арсенал технических средств защиты. Наибольший положительный эффект достигается в том случае, когда все перечисленные способы применяются совместно, т.е. комплексно.

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*.

Эффективность функционирования объекта с учетом воздействия несанкционированного доступа:

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

ΔЕ = 25000 – 12000 = 13000;

Относительная эффективность:

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

Относительная эффективность: 12000/25000 = 0,48;

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.

Снижение эффективности: 13000/2 = 6500;



Ез = 25000 – 6500 = 18500; qз = 18500/25000 = 0.74;

Стоимость средств защиты зависит от их эффективности, и в общем случае К – есть возрастающая функция от стоимости средств защиты (С):

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

K = f(C) = 2;

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

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

Eз = 25000 – (13000/2) – 1000 = 19500;

q = Eз/E0 = 0.78 (<1).

Так как Eз < E0 (19500<25000), а q = 0.78 и стремится к единице, то, делаем вывод, что наше средство защиты эффективно.

При выполнении данного практического задания была использована теория, прилагаемая в файле с заданием.

**Практическая работа №2**

по дисциплине «Основы защиты информации»

на тему: Разработка политики информационной безопасности рекламного агентства

Цель:Разработать проект политики информационной безопасности рекламного агентства.

# **Введение**

Что такое информационная безопасность?

Информационная безопасность — это различные меры по защите информации от посторонних лиц. В доцифровую эпоху для защиты информации люди запирали важные документы в сейфы, нанимали охранников и шифровали свои сообщения на бумаге.

Сейчас чаще защищают не бумажную, а цифровую информацию, но меры, по сути, остались теми же: специалисты по информационной безопасности создают защищенные пространства (виртуальные «сейфы»), устанавливают защитное ПО вроде антивирусов («нанимают охранников») и используют криптографические методы для шифрования цифровой информации.

Однако цифровую информацию тоже нужно защищать не только виртуально, но и физически. Антивирус не поможет, если посторонний похитит сам сервер с важными данными. Поэтому их ставят в охраняемые помещения.

За что отвечает информационная безопасность

Она отвечает за три вещи: конфиденциальность, целостность и доступность информации. В концепции информационной безопасности их называют принципами информационной безопасности.

Конфиденциальность означает, что доступ к информации есть только у того, кто имеет на это право. Например, ваш пароль от электронной почты знаете только вы, и только вы можете читать свои письма. Если кто-то узнает пароль или другим способом получит доступ в почтовый ящик, конфиденциальность будет нарушена.

Целостность означает, что информация сохраняется в полном объеме и не изменяется без ведома владельца. Например, на вашей электронной почте хранятся письма. Если злоумышленник удалит некоторые или изменит текст отдельных писем, то это нарушит целостность.

Доступность означает, что тот, кто имеет право на доступ к информации, может ее получить. Например, вы в любой момент можете войти в свою электронную почту. Если хакеры атакуют серверы, почта будет недоступна, это нарушит доступность.

Какая бывает информация и как ее защищают

Информация бывает общедоступная и конфиденциальная. К общедоступной имеет доступ любой человек, к конфиденциальной — только отдельные лица.

Может показаться, что защищать общедоступную информацию не надо. Но на общедоступную информацию не распространяется только принцип конфиденциальности — она должна оставаться целостностной и доступной. Поэтому информационная безопасность занимается и общедоступной информацией.

Например, возьмем интернет-магазин. Карточки товаров, статьи в блоге, контакты продавца — все это общедоступная информация, ее может просматривать любой. Но интернет-магазин все равно нужно защищать, чтобы никто не нарушил его работу, например, не изменил важную информацию в карточках товаров или не «уронил» его сайт.

Главная задача информационной безопасности в IT и не только — защита конфиденциальной информации. Если доступ к ней получит посторонний, это приведет к неприятным последствиям: краже денег, потере прибыли компании, нарушению конституционных прав человека и другим неприятностям.

Основные виды конфиденциальной информации

Персональные данные. Информация о конкретном человеке: ФИО, паспортные данные, номер телефона, физиологические особенности, семейное положение и другие данные.

Тот, кто работает с персональными данными, обязан защищать их и не передавать третьим лицам. Информация о клиентах и сотрудниках относится как раз к персональным данным.

Коммерческая тайна. Внутренняя информация о работе компании: технологиях, методах управления, клиентской базе. Если эти данные станут известны посторонним, компания может потерять прибыль.

Компания сама решает, что считать коммерческой тайной, а что выставлять на всеобщее обозрение. При этом не вся информация может быть коммерческой тайной — например, нельзя скрывать имена учредителей юрлица, условия труда или факты нарушения законов. Подробнее о коммерческой тайне рассказывает закон 98-ФЗ.

Профессиональная тайна. Сюда относятся врачебная, нотариальная, адвокатская и другие виды тайны, относящиеся к профессиональной деятельности. С ней связано сразу несколько законов.

Служебная тайна. Информация, которая известна отдельным службам, например, налоговой или ЗАГСу. Эти данные обычно хранят государственные органы, они отвечают за их защиту и предоставляют только по запросу.

Государственная тайна. Сюда относят военные сведения, данные разведки, информацию о состоянии экономики, науки и техники государства, его внешней политики. Эти данные самые конфиденциальные — к безопасности информационных систем, в которых хранится такая информация, предъявляют самые строгие требования.

Если ваша компания хранит персональные данные, коммерческую или профессиональную тайну, то эти данные нужно защищать особым образом. Для этого необходимо ограничить доступ к ней посторонним лицам — установить уровни доступа и пароли, поставить защитное ПО, настроить шифрование.

**Структура рекламного агентства**

Рекламное агентство с полным циклом услуг имеет следующие отделы:

* творческий (креативный) отдел;
* отдел по работе с клиентами;
* производственный отдел;
* отдел маркетинга;
* финансово – хозяйственный отдел.

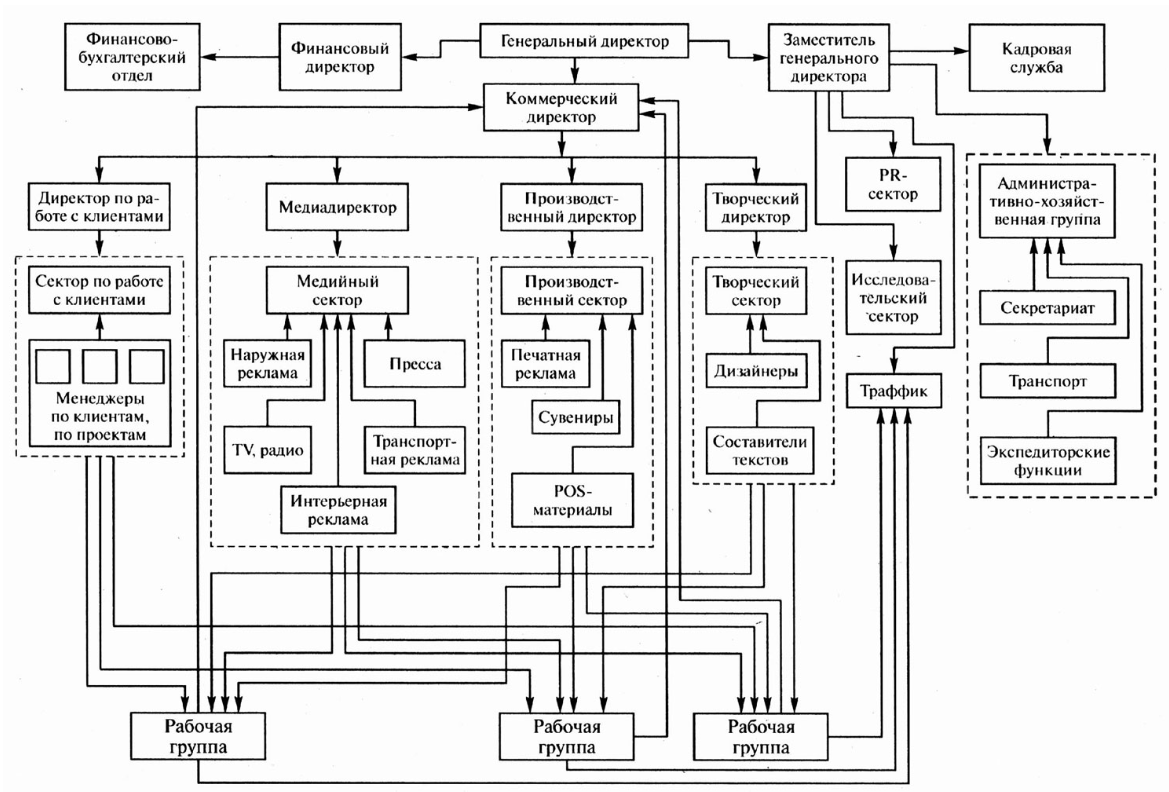
Творческий (креативный) отдел объединяет художников, создателей текстов, режиссеров, графиков, редакторов и других творческих работников. Сотрудники творческого отдела работают над идеей рекламного обращения, находят варианты ее реализации.

В состав отдела по работе с клиентами входят руководитель отдела и руководители рабочих групп, которых называют ответственными исполнителями проектов, или экаунт-менеджерами. Ответственный исполнитель проекта (accountexecutive) – сотрудник отдела работы с клиентами в рекламном агентстве, который «ведет» рекламный заказ конкретного клиента.

Производственный отдел рекламного агентства занимается изготовлением рекламных носителей и в некоторых случаях объединяет типографию, мастерские по производству рекламных щитов и другие производственные подразделения.

В отделе маркетинга рекламного агентства работают специалисты по распространению рекламы, проведению маркетинговых исследований и мероприятий по стимулированию сбыта. Чаще всего отдел маркетинга занимается планированием, закупками и исследованиями.

Помимо непосредственного осуществления своих функций, рекламное агентство должно эффективно вести свою финансово-хозяйственную деятельность и обеспечивать эффективное управление процессом. Для этого существует такая вспомогательная служба, как финансово-хозяйственный отдел, который может включать в себя библиотеку, отдел кадров и другие структурные подразделения.



# **Оценка рисков**

1. Определить ценность информационных активов в денежном выражении.

2. Оценить в количественном выражении потенциальный ущерб от реализации каждой угрозы в отношении каждого информационного актива.

Следует получить ответы на вопросы «Какую часть от стоимости актива составит ущерб от реализации каждой угрозы?», «Какова стоимость ущерба в денежном выражении от единичного инцидента при реализации данной угрозы к данному активу?».

3. Определить вероятность реализации каждой из угроз ИБ.

Для этого можно использовать статистические данные, опросы сотрудников и заинтересованных лиц. В процессе определения вероятности рассчитать частоту возникновения инцидентов, связанных с реализацией рассматриваемой угрозы ИБ за контрольный период (например, за один год).

4. Определить общий потенциальный ущерб от каждой угрозы в отношении каждого актива за контрольный период (за один год).

Значение рассчитывается путем умножения разового ущерба от реализации угрозы на частоту реализации угрозы.

5. Провести анализ полученных данных по ущербу для каждой угрозы.

По каждой угрозе необходимо принять решение: принять риск, снизить риск либо перенести риск.

Принять риск — значит осознать его, смириться с его возможностью и продолжить действовать как прежде. Применимо для угроз с малым ущербом и малой вероятностью возникновения.

Снизить риск — значит ввести дополнительные меры и средства защиты, провести обучение персонала и т д. То есть провести намеренную работу по снижению риска. При этом необходимо произвести количественную оценку эффективности дополнительных мер и средств защиты. Все затраты, которые несет организация, начиная от закупки средств защиты до ввода в эксплуатацию (включая установку, настройку, обучение, сопровождение и проч.), не должны превышать размера ущерба от реализации угрозы.

Перенести риск — значит переложить последствия от реализации риска на третье лицо, например с помощью страхования.

В результате количественной оценки рисков должны быть определены:

ценность активов в денежном выражении;

полный список всех угроз ИБ с ущербом от разового инцидента по каждой угрозе;

-частота реализации каждой угрозы;

-потенциальный ущерб от каждой угрозы;

-рекомендуемые меры безопасности, контрмеры и действия по каждой угрозе.

Система управления информационной безопасностью в Компании строится на основе международных стандартов ISO 27001 и ISO 27002.

Меры защиты информации внедряются по результатам проведения оценки рисков информационной безопасности.

Оценка рисков информационной безопасности проводится ежегодно, а также в случае значительных изменений в структуре Компании и ее бизнес-процессах.

При оценке рисков учитывается влияние реализации угроз информационной безопасности на финансовое положение Компании и ее репутацию на рынке.

Стоимость принимаемых мер не должна превышать возможный ущерб, возникающий при реализации угроз.

Успешное достижение целей настоящей политики возможно только при выполнении положений следующих детальных политик информационной безопасности:

Политика использования паролей;

Политика использования сети Интернет;

Политика использования электронной почты;

Несоблюдение политик информационной безопасности сотрудниками Компании может повлечь дисциплинарные меры взыскания вплоть до увольнения.

**Классификация угроз**

Подугрозой информационной безопасности объекта будем понимать возможные воздействия на него, приводящие к ущербу.

По виду:

— физической и логической целостности (уничтожение или искажение информации);

— конфиденциальности (несанкционированное получение);

— доступности (работоспособности);

— права собственности;

По происхождению:

— случайные (отказы, сбои, ошибки, стихийные явления);

— преднамеренные (злоумышленные действия людей);

По источникам:

— люди (персонал, посторонние);

— технические устройства;

— модели, алгоритмы, программы;

— внешняя среда (состояние атмосферы, побочные шумы, сигналы и наводки).

Информационные угрозы могут быть обусловлены:

естественными факторами (пожар, наводнение, прорыв батарей отопления и др.);

человеческими факторами.

Последние (человеческий фактор), в свою очередь, подразделяются на:

угрозы, носящие случайный, неумышленный характер. Это угрозы, связанные с ошибками процесса подготовки, обработки и передачи информации;

угрозы, обусловленные умышленными, преднамеренными действиями людей. Это угрозы, связанные с несанкционированным доступом к ресурсам АИС.

Умышленные угрозы преследуют цель нанесения ущерба пользователям АИС и, в свою очередь, подразделяются на активные и пассивные.

# **Разработка мер защиты**

Методами обеспечения защиты информации являются следующие: препятствие, управление до­ступом, маскировка, регламентация, принуждение и побуждение.  
  
Препятствие — метод физического преграж­дения пути злоумышленнику к защищаемой ин­формации (к аппаратуре, носителям информации и т. п.).  
  
Управление доступом — метод защиты инфор­мации регулированием использования всех ре­сурсов автоматизированной информационной системы организации (фирмы). Управление до­ступом включает следующие функции защиты:  
  
· идентификацию пользователей, персонала и ресурсов информационной системы (присвое­ние каждому объекту персонального идентифи­катора);  
  
· аутентификацию (установление подлиннос­ти) объекта или субъекта по предъявленному им идентификатору;  
  
· проверку полномочий (проверка соответ­ствия дня недели, времени суток, запрашивае­мых ресурсов и процедур установленному рег­ламенту);  
  
· разрешение и создание условий работы в пределах установленного регламента;  
  
· регистрацию (протоколирование) обраще­ний к защищаемым ресурсам;  
  
· реагирование (сигнализация, отключение, задержка работ, отказ в запросе) при попытках несанкционированных действий.

Для обеспечения защиты объектов информационной безопасности должны быть соответствующие правовые акты, устанавливающие порядок защиты и ответственность за его нарушение. Законы должны давать ответы на следующие вопросы: что такое информация, кому она принадлежит, как может с ней поступать собственник, что является посягательством на его права, как он имеет право защищаться, какую ответственность несет нарушитель прав собственника информации.

Установленные в законах нормы реализуются через комплекс организационных мер, проводимых прежде всего государством, ответственным за выполнение законов, и собственниками информации. К таким мерам относятся издание подзаконных актов, регулирующих конкретные вопросы по защите информации (положения, инструкции, стандарты и т. д.), и государственное регулирование сферы через систему лицензирования, сертификации, аттестации.

Поскольку в настоящее время основное количество информации генерируется, обрабатывается, передается и хранится с помощью технических средств, то для конкретной ее защиты в информационных объектах необходимы технические устройства. В силу многообразия технических средств нападения приходится использовать обширный арсенал технических средств защиты. Наибольший положительный эффект достигается в том случае, когда все перечисленные способы применяются совместно, т.е. комплексно.

Бывают формальные и неформальные средства защиты.

Формальные средства защиты:

Физические средства - механические, электрические, электромеханические, электронные, электронно-механические и тому подобные устройства и системы, которые функционируют автономно от информационных систем, создавая различного рода препятствия на пути дестабилизирующих факторов (замок на двери, жалюзи, забор, экраны, датчики пожаротушения и др.).

Аппаратные средства - механические, электрические, электромеханические, электронные, электронно-механические, оптические, лазерные, радиолокационные и тому подобные устройства, встраиваемые в информационных системах или сопрягаемые с ней специально для решения задач защиты информации.

Программные средства - пакеты программ, отдельные программы или их части, используемые для решения задач защиты информации. Программные средства не требуют специальной аппаратуры, однако они ведут к снижению производительности информационных систем, требуют выделения под их нужды определенного объема ресурсов и т.п.

К специфическим средствам защиты информации относятся криптографические методы. В информационных системах криптографические средства защиты информации могут использоваться как для защиты обрабатываемой информации в компонентах системы, так и для защиты информации, передаваемой по каналам связи. Само преобразование информации может осуществляться аппаратными или программными средствами, с помощью механических устройств, вручную и т.д.

Неформальные средства защиты:

Законодательные средства – законы и другие нормативно-правовые акты, с помощью которых регламентируются правила использования, обработки и передачи информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил. Распространяются на всех субъектов информационных отношений.

Организационные средства - организационно-технические и организационно-правовые мероприятия, осуществляемые в течение всего жизненного цикла защищаемой информационной системы (строительство помещений, проектирование информационных систем, монтаж и наладка оборудования, испытания и эксплуатация информационных систем). Другими словами – это средства уровня организации, регламентирующие перечень лиц, оборудования, материалов и т.д., имеющих отношение к информационным системам, а также режимов их работы и использования. К организационным мерам также относят сертификацию информационных систем или их элементов, аттестацию объектов и субъектов на выполнение требований обеспечения безопасности и т.д.

Морально-этические средства - сложившиеся в обществе или в данном коллективе моральные нормы или этические правила, соблюдение которых способствует защите информации, а нарушение приравнивается к несоблюдению правил поведения в обществе или коллективе, ведет к потере престижа и авторитета.

Для защиты от повышенного уровня статического электричества существуют: заземляющие устройства; нейтрализаторы; увлажняющие устройства; антиэлектростатические вещества; экранизирующие устройства.

Также стоит помнить о рисках проникновения на территорию организации посторонних лиц. Для того, чтобы не допускать таких ситуаций, на всех предприятиях установлены пункты пропуска – турникеты, пускающие сотрудников организации лишь по специальным пропускным документам. Для отслеживания посторонних лиц на территории здания следует устанавливать камеры видеонаблюдения.

Не стоит забывать о пожарной безопасности. Для этого устанавливаются датчики уровня дыма и системы пожаротушения.

Для того, чтобы в сеть не попал нежелательный пользователь, либо вредоносное программное обеспечение, стоит устанавливать антивирусы на рабочие компьютеры. Также можно хранить копии данных на внешних носителях либо серверах. В случае несанкционированного проникновения будет возможность отключить всю систему, не беспокоясь о безопасности данных. На каждом компьютере должен быть установлен брандмауэр.

# **Вывод**

Важно помнить, что прежде чем внедрять какие-либо решения по защите информации необходимо разработать политику безопасности, адекватную целям и задачам современного предприятия. В частности, политика безопасности должна описывать порядок предоставления и использования прав доступа пользователей, а также требования отчетности пользователей за свои действия в вопросах безопасности.

Система информационной безопасности (СИБ) окажется эффективной, если она будет надежно поддерживать выполнение правил политики безопасности, и наоборот.

Этапы построения политики безопасности – это внесение в описание объекта автоматизации структуры ценности и проведение анализа риска, и определение правил для любого процесса пользования данным видом доступа к ресурсам объекта автоматизации, имеющим данную степень ценности.

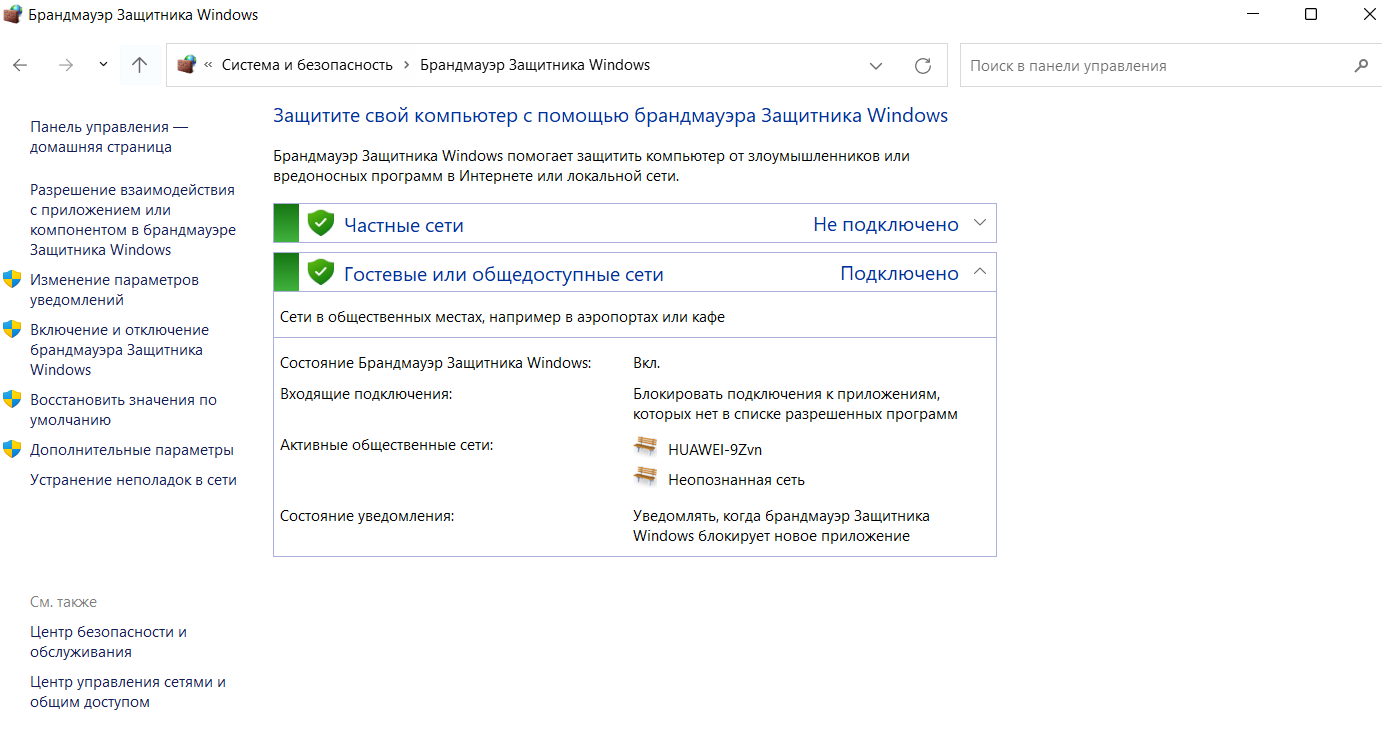
**Практическая работа №3**

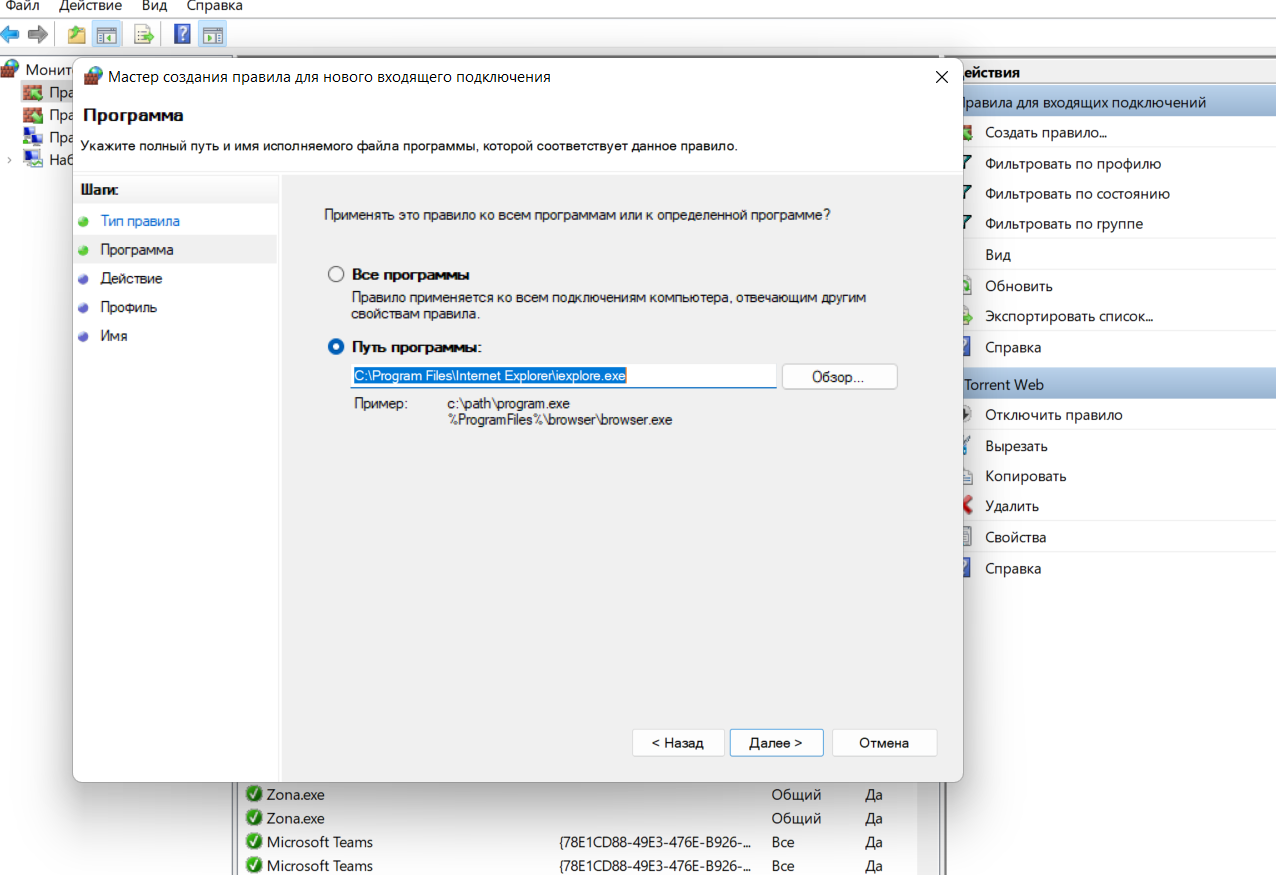
по дисциплине «Основы защиты информации»

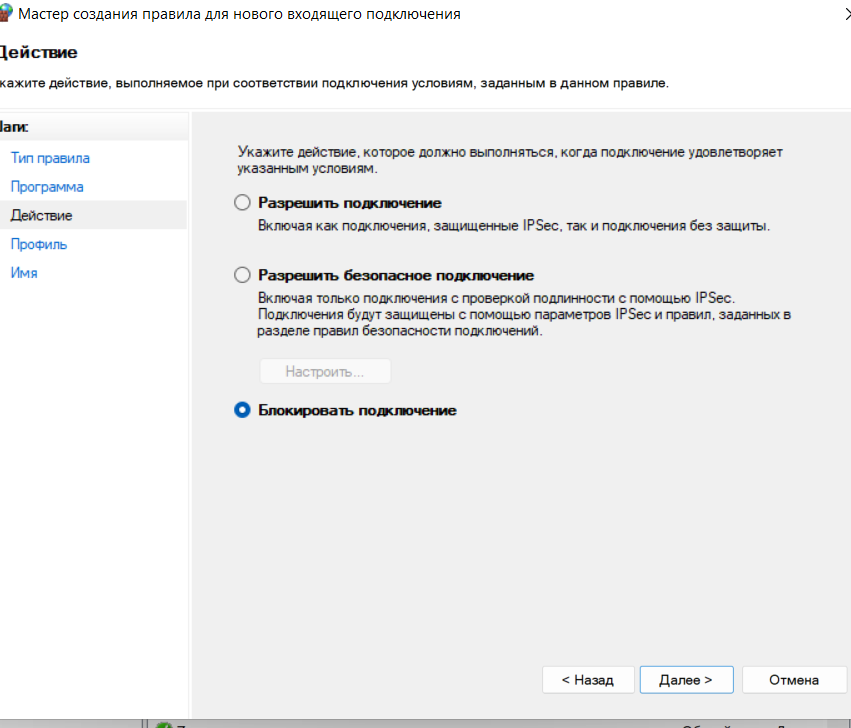
на тему: Настройка Брандмауэра Windows

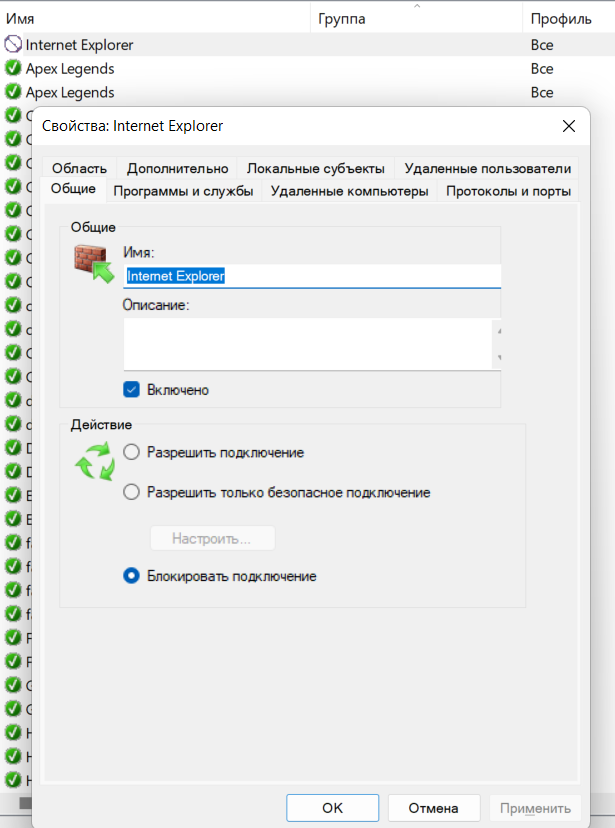
Цель: Овладение навыками настройки и использования Брандмауэра Windows.

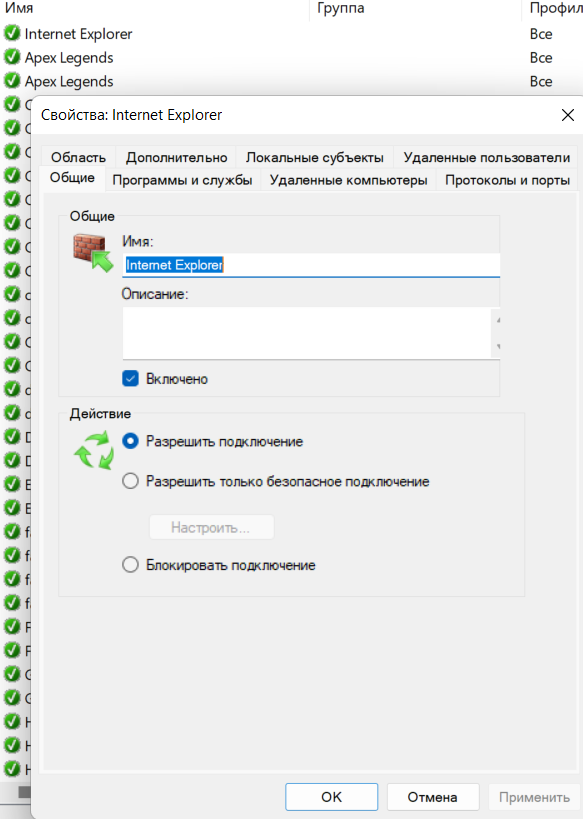
1. **Правила для входящих подключений**



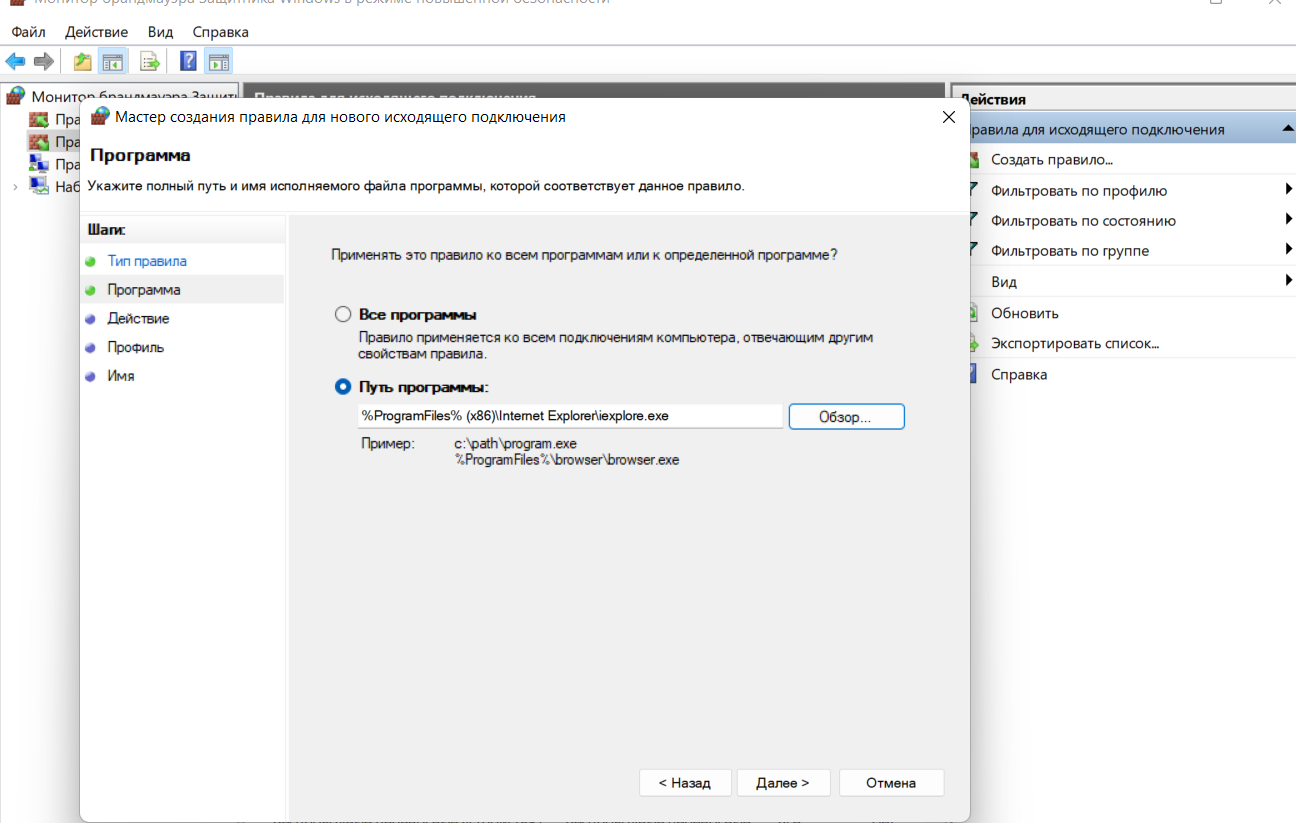


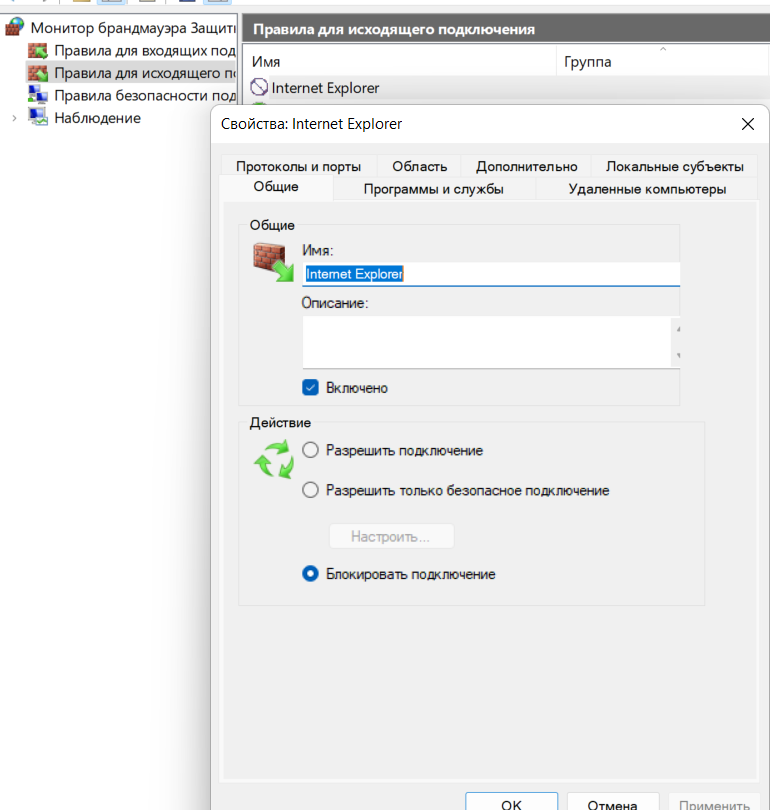


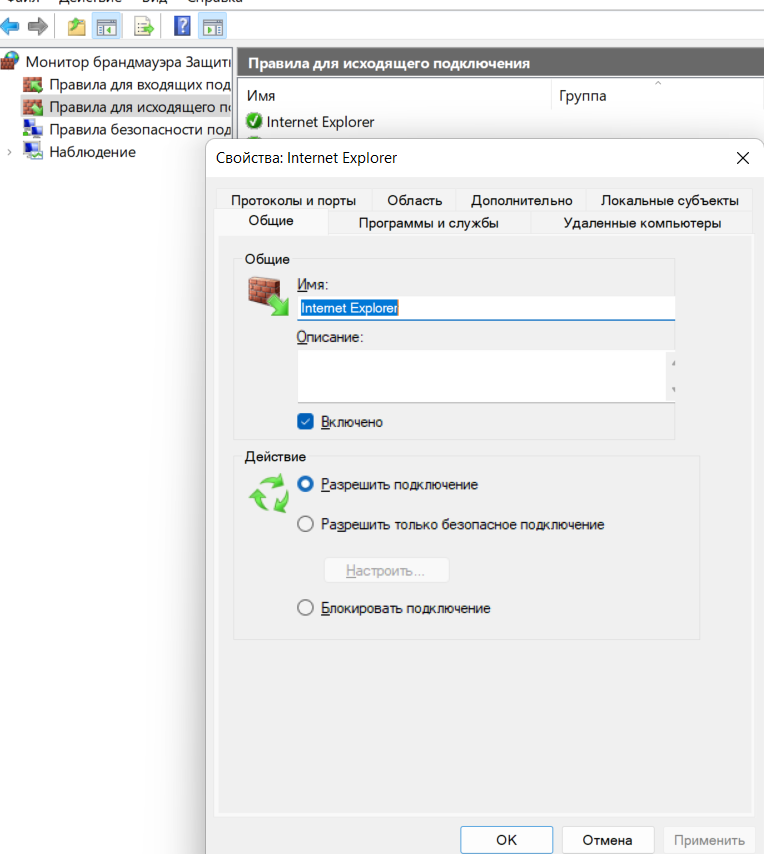
Блокировка: 

Разрешение: 

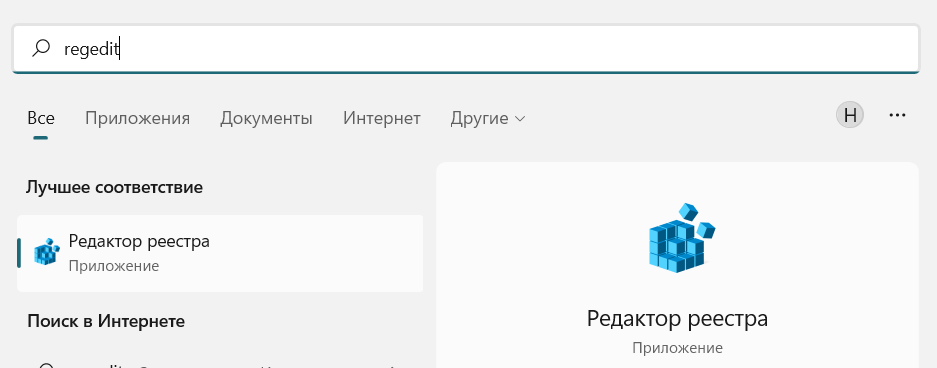
1. **Правила для исходящих подключений**

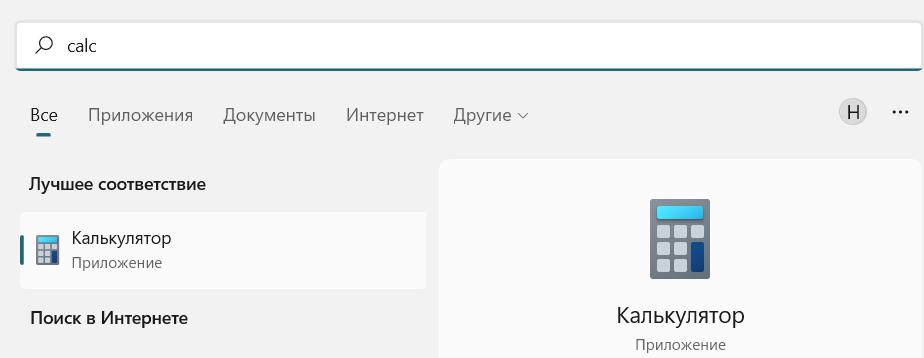


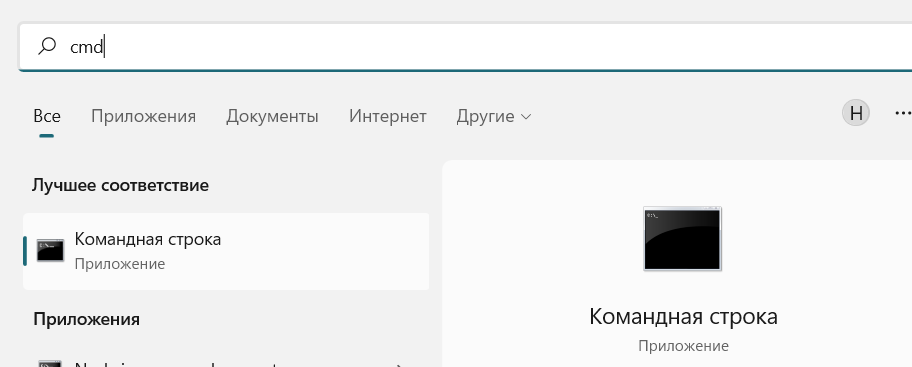
Блокировка: 

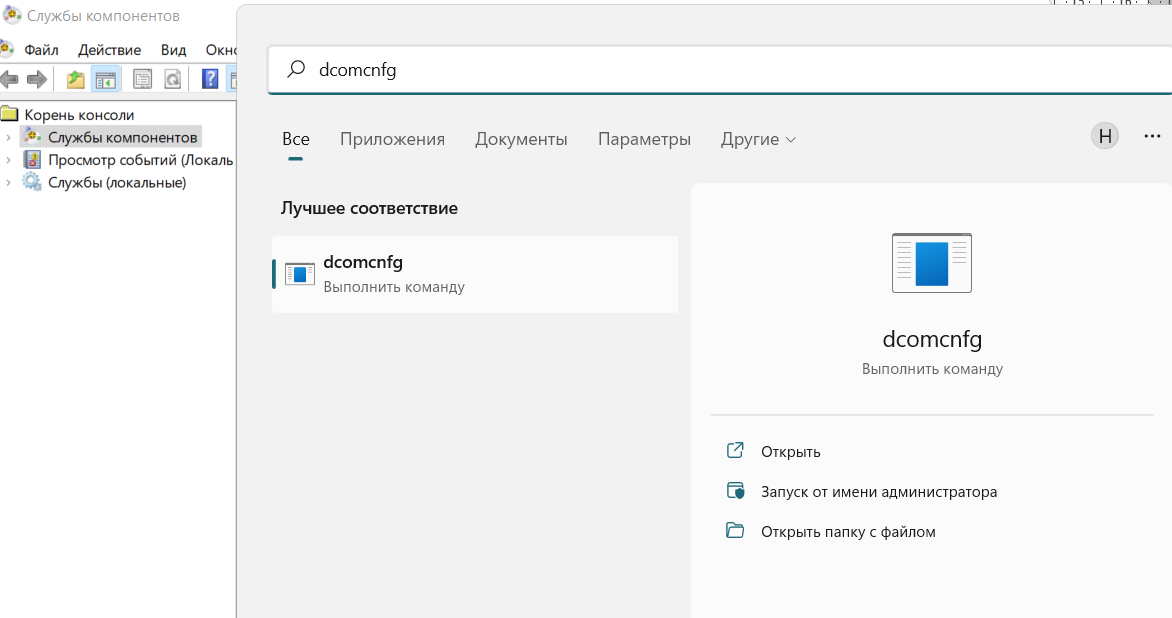
Разрешение: 

1. **Команды**









**Практическая работа №4.1**

по дисциплине «Основы защиты информации»

на тему: 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 по блокам. 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. Различают 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.



Рисунок 5.1 – Принцип замены

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 использованием системы Цезаря:**

Шифр Цезаря — один из древнейших шифров. При шифровании каждый символ заменяется другим, отстоящим от него в алфавите на фиксированное число позиций. Шифр Цезаря можно классифицировать как шифр подстановки, при более узкой классификации — шифр простой замены.

Шифр назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки.

Математическая модель

Если сопоставить каждому символу алфавита его порядковый номер (нумеруя с 0), то шифрование и дешифрование можно выразить формулами:

http://kriptografea.narod.ru/13.png

http://kriptografea.narod.ru/14.png

где x — символ открытого текста

y — символ шифрованного текста

n — мощность алфавита (кол-во символов)

k — ключ.



Рисунок 5.2 – Алфавит

Пример:

Таблица 5.1 Шифр Цезаря

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Сообщение** | **Х** | **А** | **Л** | **А** | **Л** | **Е** | **Е** | **Н** | **К** | **О** | **А** | **Н** | **Д** | **Р** | **Е** | **Й** |
| **Номер** | **23** | **1** | **13** | **1** | **13** | **6** | **6** | **15** | **12** | **16** | **1** | **15** | **5** | **18** | **6** | **11** |
| **Номер 1+4** | **27** | **5** | **17** | **5** | **17** | **11** | **11** | **19** | **16** | **20** | **5** | **19** | **9** | **22** | **10** | **15** |
| **Шифр** | **Щ** | **Д** | **П** | **Д** | **П** | **И** | **И** | **С** | **О** | **Т** | **Д** | **С** | **З** | **Ф** | **И** | **Н** |

Ответ: «Щдпдпиисот Дсзфин», сдвиг 4

**Шифрование с использованием системы 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.5.2 изображена таблица с ключевым словом «ЗАЩИТА».



Рисунок 5.2 – Таблица шифрозамен для шифра Трисемуса с ключевым словом «ЗАЩИТА»

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 «ХАЛАЛЕЕНКО АНДРЕЙ», зашифрованное – «ЪБПБПЛЛСОУ БСКХЛН»

**Шифрование с использованием системы Виженера:**

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 32 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 32 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет вид:

ХАЛАЛЕЕНКОАНДРЕЙ

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

ЗАЩИТАЗАЩИТАЗАЩИ

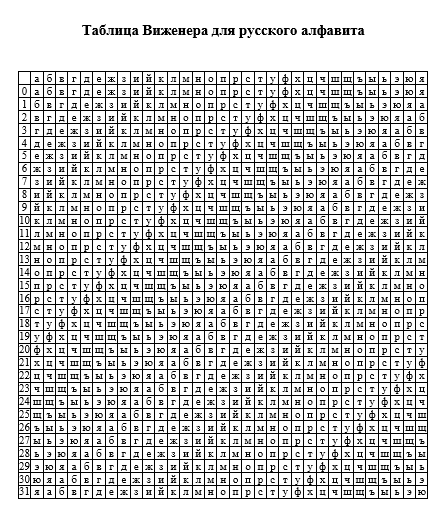


Рисунок 5.3 – Таблица Виженера для русского алфавита

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

*Исходный текст: ХАЛАЛЕЕНКОАНДРЕЙ*

*Ключ: ЗАЩИТАЗАЩИТАЗАЩИ*

*Зашифрованный текст: эаеиюемндч тнлрют*

**Шифрование с использованием системы Плейфера:**

Шифр Плейфера использует матрицу 5х5 (для латинского алфавита, для кириллического алфавита необходимо увеличить размер матрицы до 4х8), 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 русских текстах обычно «Е» и «Ё» 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 4х8 и является ключом шифра.

Для того чтобы зашифровать сообщение, необходимо разбить его на биграммы (группы из двух символов), например «Халалеенко Андрей» становится «ХА ЛА ЛЕ ЕН КО АН ДР ЕЙ», 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.

Пример: Используем ключ «ЗАЩИТА», тогда матрица примет вид:

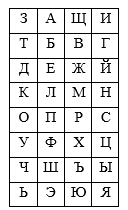


Рисунок 5.4 – Таблица шифрозамен для шифра Плейфера с ключевым словом «ЗАЩИТА»

Зашифруем сообщение «Халалеенко Андрей»

ХА ЛА ЛЕ ЕН КО АН ДР ЕЙ

1. Биграмма ХА формирует прямоугольник, заменяем её на ЩФ.  
   2. Биграмма ЛА расположена в одном столбце, заменяем её на ПБ.  
   3. Биграмма ЛЕ расположена в одном столбце, заменяем её на ПЕ.  
   4. Биграмма ЕН формирует прямоугольник, заменяем её на ЛЙ.  
   5. Биграмма КО расположена в одном столбце, заменяем её на ОУ.  
   6. Биграмма АН формирует прямоугольник, заменяем её на ЛИ.  
   7. Биграмма ДР формирует прямоугольник, заменяем её на ОЖ.

8. Биграмма ЕЙ расположена в одной строке, заменяем её на ЖД.

Получаем зашифрованный текст «ЩФ ПБ ПЕ ЛЙ ОУ ЛИ ОЖ ЖД»

Таким образом сообщение «ХАЛАЛЕЕНКО АНДРЕЙ» преобразуется в «ЩФПБПЕЛЙОУ ЛИОЖЖД»

Расщифровка:

Зашифрованный текст «**иецеж пецен, пй ыд зещеъез**»

Расшифровать с помощью шифра Трисемуса

**Ключ МОСКВА**

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

Получаем расшифрованный текст «волос долог, да ум короток»

Таким образом сообщение «иецеж пецен, пй ыд зещеъез» преобразуется в «волос долог, да ум короток»

Вывод: в ходе этой практической работы, мы получили навыки работы с методами шифрования для криптографической защиты информации при передаче и хранении данных.

**Практическая работа №4.2**

по дисциплине «Основы защиты информации»

на тему: Криптографическая защита информации

Цель: Овладение основными криптографическими алгоритмами асимметричного шифрования .

# **Теоретическое введение**

# **Реализация элементов криптосистемы RSA**

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE и других.

Весь алгоритм расписан 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} |

# **Преимущество** — при помощи открытого ключа алгоритма шифрования невозможно прочитать видоизмененное сообщение. Для этого требуется закрытый ключ, который есть только у адресата.

**Недостаток** — медленное шифрование из-за громоздкости вычислительных операций. Невозможность получить доступ ко всем сообщениям, которые шифруются одним ключом.

# **Реализация элементов схемы шифрования Дифи-Хеллмана**

## **Генерация ключей**

В 1976 году после публичной критики алгоритма DES и указания на сложность обработки секретных ключей 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.

Из‑за невысокого быстродействия, свойственного асимметричным алгоритмам, алгоритм Диффи‑Хеллмана не предназначен для шифрования данных. 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.

До алгоритма Диффи‑Хеллмана было сложно совместно использовать зашифрованные данные из‑за проблем хранения ключей и передачи информации. В большинстве случаев передача информации по каналам связи небезопасна, потому что сообщение может пройти десятки систем, прежде чем оно достигнет потенциального адресата, и нет никаких гарантий, что по пути никто не сможет взломать секретный ключ. Уитфилд Диффи и Мартин Хеллман предложили зашифровывать секретный ключ DES по алгоритму Диффи‑Хеллмана на передающей стороне и пересылать его вместе с сообщением, зашифрованным с использованием DES. Тогда на другом конце его сможет расшифровать только получатель сообщения.

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. Алгоритм Диффи‑Хеллмана основан на полезных для криптографии свойствах дискретных логарифмов.

**Преимущество** — никто не сможет скомпрометировать секретное сообщение. Полученный ключ может быть использован для шифрования по любым доступным сторонам алгоритма.

**Недостаток** — два различных независимых цикла шифрования. Отсутствие взаимной аутентификации сторон.

# **Реализация элементов схемы шифрования Эль-Гамаля**

## **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**

Сообщение ~M шифруется следующим образом:

1. 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
2. Вычисляются числа a = g^k\,\bmod\,p и b = y^k M\,\bmod\,p.
3. 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).

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

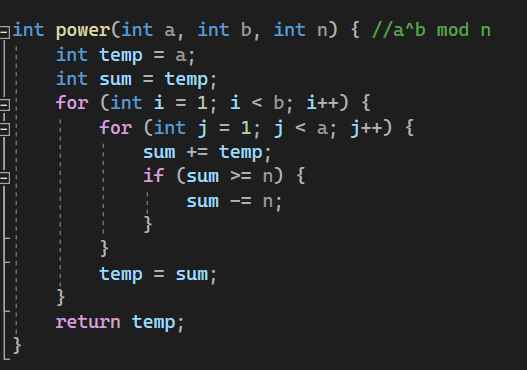
~b(a^x)^{-1}\equiv (y^kM)g^{-xk}\equiv (g^{xk}M) g^{-xk}\equiv M \pmod{p}.

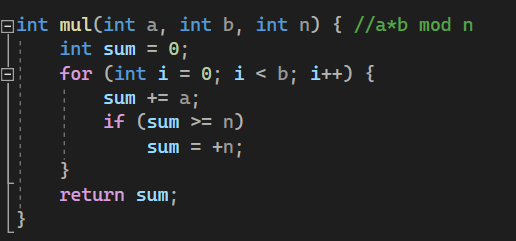
Для практических вычислений больше подходит следующая формула:

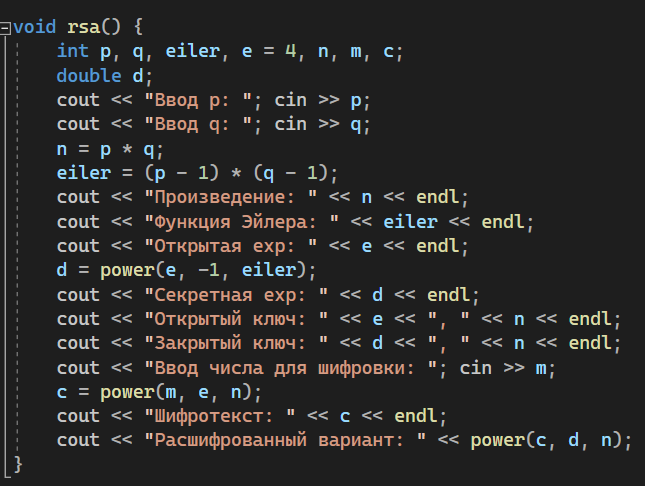
M = b(a^x)^{-1}\,\bmod\,p = b \cdot a^{(p-1-x)}\,\bmod\,p 

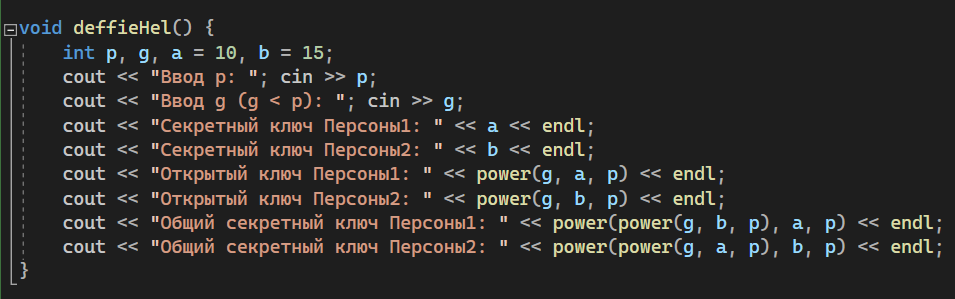
**Преимущество** — вероятностный характер шифрования — большая стойкость.

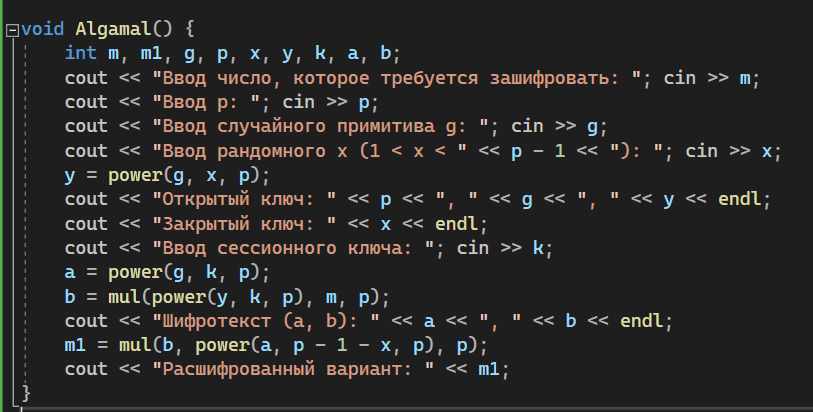
**Недостаток** — удвоение длины зашифрованного текста по сравнению с начальным текстом. Само сообщение и ключ не определяют шифртекст однозначно.











**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.

**Этап 1.** Выработка ключей (выполняет отправитель **А**) - см. практическую работу 6  "Шифрование методом RSA".

**Этап 2.** Отправка сообщения и электронной подписи (выполняет отправитель **А**).

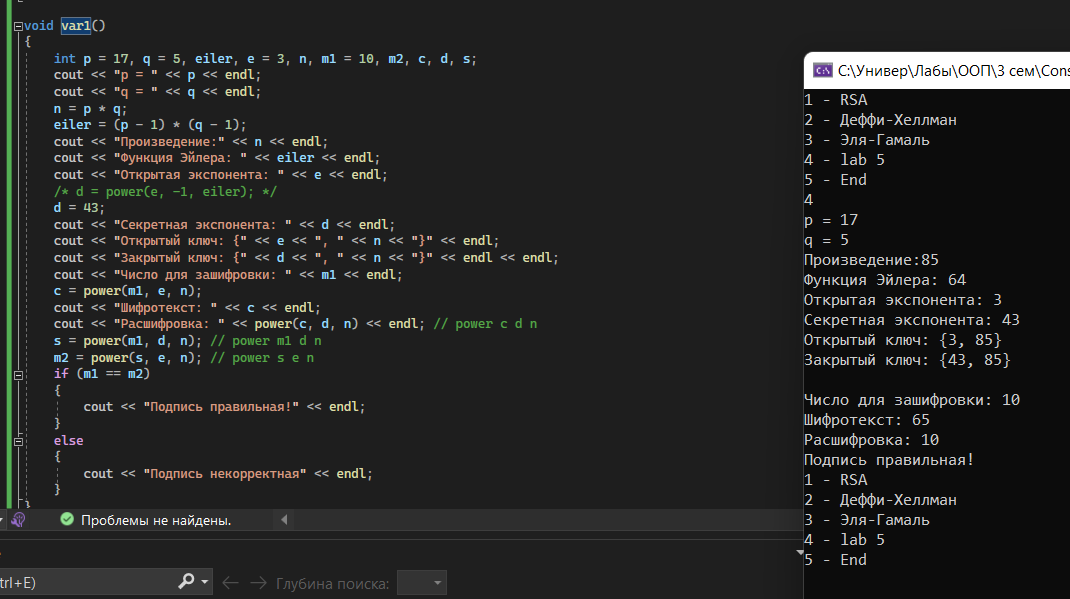
Отправка сообщения и ЭЦП на базе алгоритма RSA



**Этап 3.** Получение сообщения и проверка электронной подписи (выполняет получатель **B**).

Получение сообщения и проверка ЭЦП на базе алгоритма RSA

* 

****

|  |
| --- |
| **Практическое задание № 6** |
| **Тема «Теория чисел»** |
| Цель**:**  получение основных сведений из курса теории чисел |
| **Теоретические сведения** |

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.

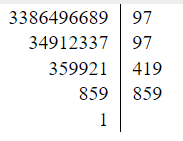
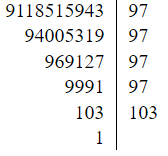
**Решение.** **. Поэтому в силу свойства 4 сравнений . Двоичная запись: 29=11101. Следовательно, для любого натурального *a* величина . Далее, . Поэтому . Тогда . Следовательно, . Таким образом, остаток от деления  на 31 равен 4.

**Варианты индивидуальных заданий**

|  |  |
| --- | --- |
| № варианта | Задание |
|  | 1-3. *а*=101398751, *b*=326147777.  4. Найти остаток от деления  на 29. |
|  | 1-3. *а* = 5999801, *b* = 48685811  4. Найти остаток от деления  на 17. |
|  | 1-3. *а* = 660422941, *b* = 36481301. +  4. Найти остаток от деления  на 17. |
|  | 1-3. *а* = 9002242397, *b* = 433817903.  4. Найти остаток от деления  на 19. |
|  | 1-3. а = 9118515943, b = 3386496689.  4. Найти остаток от деления  на 23. |
|  | 1-3. *а* = 5336161097, *b* = 196210799. .  4. Найти остаток от деления  на 19. |
|  | 1-3. *а* = 7049964661, *b* = 168687989. .  4. Найти остаток от деления  на 17. |
|  | 1-3. *а* = 83748733, *b* = 73435591. .  4. Найти остаток от деления  на 11. |
|  | 1-3. *а* = 16254559, *b* = 1029073. .  4. Найти остаток от деления  на 19. |
|  | 1-3. *а* = 6099377, *b* = 9568217.  4. Найти остаток от деления  на 17 числа. |
|  | 1-3. а = 7957549, b = 23118553.  4. Найти остаток от деления  на 19. |
|  | 1-3. *а* = 16088437, *b* = 18216949.  4. Найти остаток от деления  на 16. |
|  | 1-3. *а* = 244604911, *b* = 61875907.  4. Найти остаток от деления  на 17. |
|  | 1-3. *а* = 356216713, *b* = 31238065.  4. Найти остаток от деления  на 19. |
|  | 1-3. *а* = 7409621, *b* = 6793883.  4. Найти остаток от деления  на 29. |

**Вариант: 5**

а = 9118515943, b = 3386496689

****

1. 9118515943 = 97 ∙ 97 ∙ 97 ∙ 97 ∙ 103;

3386496689 = 97 ∙ 97 ∙ 419 ∙ 859;

1. НОД(9118515943, 3386496689) = 3 (вообще, должен быть 1, но, учитывая погрешности при вычислении…)

9118515943 / 3386496689 = 2 (6926103228)

6926103228 / 3386496689 = 2 (452118705)

3386496689 / 452118705 = 7 (4902822)

452118705 / 4902822 = 92 (216014)

4902822 / 216014 = 22 (69677)

216014 / 69677 = 3 (1002)

69677 / 1002 = 69 (537)

1002 / 537 = 1 (86)

537 / 86 = 6 (24)

86 / 24 = 3 (6)

24 / 6 = 4 (0)

6926103228 = 9118515943 + 3386496689(-2)

452118705 = 6926103228 + 3386496689(-2)

4902822 = 3386496689 + 452118705(-7)

216014 = 452118705 + 4902822(-92)

69677 = 4902822 + 216014(-22)

1002 = 216014 + 69677(-3)

537 = 69677 + 1002(-69)

86 = 1002 + 537(-1)

24 = 537 + 86(-6)

1. = 86 + 24(-3)
2. 6 = 86 + 24(-3) = 86 + (537 + 86(-2))(-3) = 86\*7 + 537(-3) = (1002 + 537(-1))\*7 + (69677 + 1002(-69))(-3) = ((216014 + 69677(-3)) + (69677 + 1002(-69)))\*7 + ((4902822 + 216014(-22)) + (216014 + 69677(-3))(-3) = ((452118705 + 4902822(-92)) + (4902822 + 216014(-22))(-3) + (4902822 + 216014(-22)) + (216014 + 69677(-3)(-69)))\*7 + ((3386496689 + 452118705(-7)) + 216014(-22) + (216014 + 69677(-3))\*(-3))) =

#include <iostream>

#include <tuple> // std::tuple, std::make\_tuple, std::tie

using namespace std;

// Рекурсивная функция для демонстрации расширенного алгоритма Евклида.

// Он возвращает несколько значений, используя кортеж в C++.

tuple<int, int, int> extended\_gcd(int a, int b)

{

if (a == 0) {

return make\_tuple(b, 0, 1);

}

int gcd, x, y;

// распаковываем возвращаемый функцией кортеж в переменные

tie(gcd, x, y) = extended\_gcd(b % a, a);

return make\_tuple(gcd, (y - (b / a) \* x), x);

}

int main()

{

int a = 9118515943;

int b = 3386496689;

tuple<int, int, int> t = extended\_gcd(a, b);

int gcd = get<0>(t);

int x = get<1>(t);

int y = get<2>(t);

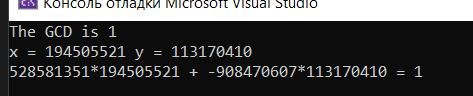
cout << "The GCD is " << gcd << endl;

cout << "x = " << x << " y = " << y << endl;

cout << a << "\*" << x << " + " << b << "\*" << y << " = " << gcd << endl;

return 0;

}



9118515943\*194505521 + 3386496689\*113170410 = 1.

1. Найти остаток от деления  на 23:

1999 делится на 23 с остатком 9, 1999^2 делится на 23 с остатком 1. Следовательно, при дальнейшем возведении двойки в степень остатки от деления будут чередоваться 9, 1, 9, 1…

Значит, в силу нечетности степени 2005, остаток от деления требуемого числа на 23 будет равен 9.

**Практическое задание № 7**

**Тема «**Настройка антивирусов**»**

Цель: Овладение навыками настройки и использования различных антивирусов.

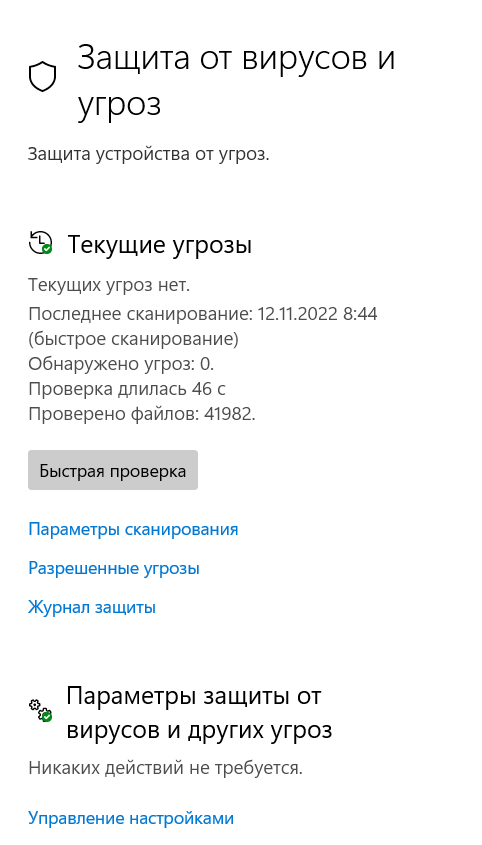


Рисунок 1 – проверка базы на наличие вирусных угроз

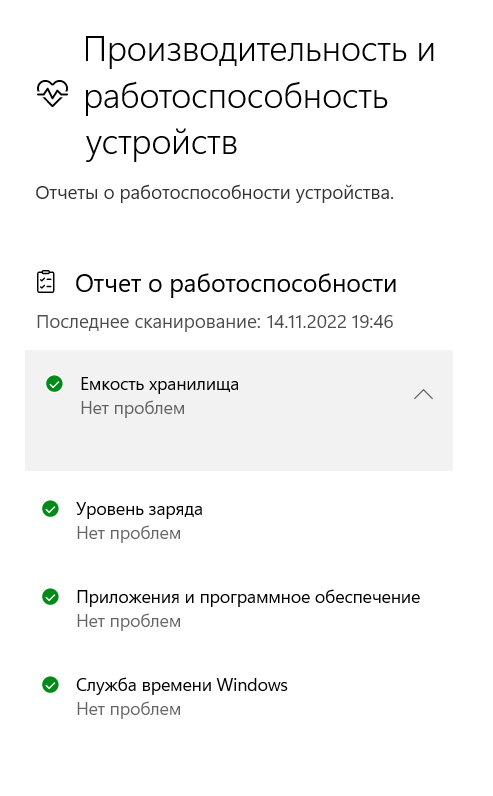


Рисунок 2 – проверка устройства (а, соответственно, и дисков) на наличие угроз

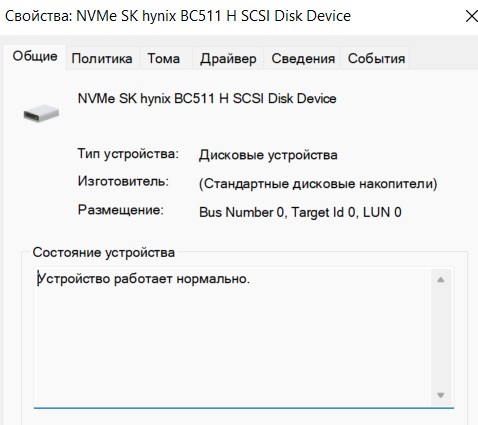


Рисунок 3 – проверка совместимости диска с драйверами

|  |
| --- |
| **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.  Рассмотрим часть кода для шифрования сообщения используя алгоритм Aes. Как представлено ниже мы создаём объект шифратор на основе созданного экземпляра криптографического объекта aesAlg. Далее мы создаём потоки для шифрования. Стоит отметить CryptoStream который определяет поток, связывающий потоки данный с криптографическим преобразованием.  ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);  // создание потоков используемого для шифрования  using (MemoryStream msEncrypt = new MemoryStream())  {  using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))  //1-используемый поток, 2 - криптографические преобразование, 3 тип доступа  {  using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))  {  swEncrypt.Write(plainText);  }  encrypted = msEncrypt.ToArray();  }  }  }  Дешифрование выполняется аналогично за исключением изменения типа доступа CryptoStream и изменения потока с записи на чтение.  ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);  using (MemoryStream msDecrypt = new MemoryStream(cipherText))  {  using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))  {  using (StreamReader srDecrypt = new StreamReader(csDecrypt))  {  plaintext = srDecrypt.ReadToEnd();  }  }  }  Также стоит упомянуть что сложность большинства алгоритмов шифрование имеет сильную зависимость от длины ключа. Изменение длины ключа возможно в большинстве алгоритмов как симметричного, так и ассиметричного шифрования. Для того что бы изменить размер ключа после инициализации алгоритма необходимо изменить свойство KeySize, что представлено в листинге ниже.  using (TripleDES myDes = TripleDES.Create())  {  myDes.KeySize = 128;  EncryptStringToFile\_DES(route, "2 keys usedE.txt", myDes.Key, myDes.IV);  DecryptStringToFile\_DES("2 keys usedE.txt", "2 key deceypted.txt", myDes.Key, myDes.IV);  }  Что касается ассиметричных алгоритмов то они представлены в данной структуре наследования.  AsymmetricAlgorithm  Rsa  RsaCng  RsaOpenSsl  RsaCryptoServiceProvider  Dsa  DsaCng  DsaOpenSsl  DsaCryptoServiceProvider  ECDiffieHellman  ECDiffieHellmanCng  ECDiffieHellmanOpenSsl  Шифрование и дешифрование для асимметричных алгоритмов выполняется проще из за встроенных функций Encrypt и Decrypt. Так же стоит помнить, что Rsa является блочным алгоритмов и если длина данных не совпадает с длиной блока, то данные нужно дополнить до длины блока. Для этого в данном пространстве имён можно использовать разные режимы заполнения.  using (RSA myRsa = RSA.Create())  {  string publickey = myRsa.ToXmlString(false); //получим открытый ключ  string privatekey = myRsa.ToXmlString(true);//получим закрытый ключ  byte[] encrypted = myRsa.Encrypt(data, RSAEncryptionPadding.Pkcs1);  byte[] decrypted = myRsa.Decrypt(encrypted, RSAEncryptionPadding.Pkcs1);  Console.WriteLine("До шифрования: {0}", original);  Console.WriteLine("Зашифрованное: {0}", System.Text.Encoding.UTF8.GetString(encrypted));  Console.WriteLine("После дешифровки: {0}", System.Text.Encoding.UTF8.GetString(decrypted));  }  Как представлено в коде выше после создания экземпляра Rsa возможна сразу же зашифровать и дешифровать данные использую автоматически сгенерированные ключи. Также код представленный выше содержит пример для получения открытого и закрытого ключа в виде строки в формате Xml.  <RSAKeyValue>  <Modulus>6yEjtrItcUq1hoA01xc63EW5/P99kstIobXsxPCUfUODRn2dazzcyhJ5Quhw1oHodlOMvtDN3xJdOTWDbH3xdQ==  </Modulus>  <Exponent>AQAB  </Exponent>  </RSAKeyValue>  Пример сохранённого в файл открытого ключа  <RSAKeyValue>  <Modulus>6yEjtrItcUq1hoA01xc63EW5/P99kstIobXsxPCUfUODRn2dazzcyhJ5Quhw1oHodlOMvtDN3xJdOTWDbH3xdQ==  </Modulus>  <Exponent>AQAB  </Exponent>  <P>9TsbWvgvA2OqPZxUZ96PomUG8rJk2T0SiH6chz65zkc=  </P>  <Q>9XR41YP8/CUoajovRPKWQZou3J23n3usp1acC3v9dGM=  </Q>  <DP>lnjHJ0GD72t3KUjETdu1fKrK4Z5u3RFpTtGJkD7/2b8=  </DP>  <DQ>9Qv2ppNCuigOlR7JmjflslDPgAklDN9XmyhoWT7L5qk=  </DQ>  <InverseQ>gSJ6G275fGFrEMqwSdgJYvmUQhnpCTcX0T3imIVQwOE=  </InverseQ>  <D>z0gMwu+6zehNtP/rFT9eXXd+qgHWAwYAxsapr0hjrzSXv1qS9QJJ+062YbdHc24WZagrKqABfOLQ3hfLXP3JdQ==  </D></RSAKeyValue>  Пример сохранённого в файл открытого ключа  Как видно на представленных выше файлах открытого и закрытого ключа объём ключей отличается, но также совпадают modulus (n) и Exponent (e). Остальные параметры в закрытом ключе совпадают с общепринятыми обозначениями кроме DP=d mod (p-1), DQ=d mode (q-1) и D обозначающего r или закрытый показатель степени.  Xеширования в .Net выполняется проще остальных операций из за отсутствия необходимости в обратном преобразовании. В данном пространстве имён поддерживаются следующие алгоритмы:   * MD5 * SHA256 * SHA384 * SHA512   Пример кода представлен ниже.  byte[] hashValue = new byte[256];    using (SHA256 mysha256 = SHA256.Create())  {  hashValue = mysha256.ComputeHash(File.ReadAllBytes("sha.txt"));  File.WriteAllBytes("hash.txt", hashValue);    }  Как видно из представленного кода после инициализации экземпляра для шифрования нужно вызвать только один метод ComputeHash который выполнит хеширование данных.  Пример захешированных данных представлен ниже    Рисунок 1  **Вариант 20**  using System.Security.Cryptography;  using System.Text;  class RSACSPSample  {  static void Main()  {  try  {  //Create a UnicodeEncoder to convert between byte array and string.  UnicodeEncoding ByteConverter = new UnicodeEncoding();  //Create byte arrays to hold original, encrypted, and decrypted data.  byte[] dataToEncrypt = ByteConverter.GetBytes("Andry Halaleenko");  byte[] encryptedData;  byte[] decryptedData;  //Create a new instance of RSACryptoServiceProvider to generate  //public and private key data.  using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())  {  //Pass the data to ENCRYPT, the public key information  //(using RSACryptoServiceProvider.ExportParameters(false),  //and a boolean flag specifying no OAEP padding.  encryptedData = RSAEncrypt(dataToEncrypt, RSA.ExportParameters(false), false);  //Pass the data to DECRYPT, the private key information  //(using RSACryptoServiceProvider.ExportParameters(true),  //and a boolean flag specifying no OAEP padding.  decryptedData = RSADecrypt(encryptedData, RSA.ExportParameters(true), false);  using (var writter = new StreamWriter("file.txt"))  {  writter.WriteLine("RSA");  writter.WriteLine("encrypted text: {0}", ByteConverter.GetString(encryptedData));  writter.WriteLine("Decrypted text: {0}", ByteConverter.GetString(decryptedData));  writter.WriteLine(RSA.ExportParameters(false));  }  Console.WriteLine("encrypted text: {0}", ByteConverter.GetString(encryptedData));  Console.WriteLine("Decrypted text: {0}", ByteConverter.GetString(decryptedData));  }  byte[] hashValue = new byte[256];  using (MD5 md = MD5.Create())  {  hashValue = md.ComputeHash(File.ReadAllBytes("C:\\Загрузки\\ConsoleApp2\\ConsoleApp2\\bin\\Debug\\net6.0\\md5.txt"));  File.WriteAllBytes("C:\\Загрузки\\ConsoleApp2\\ConsoleApp2\\bin\\Debug\\net6.0\\md5.txt", hashValue);  }  }  catch (ArgumentNullException)  {  //Catch this exception in case the encryption did  //not succeed.  Console.WriteLine("Encryption failed.");  }  }  public static byte[] RSAEncrypt(byte[] DataToEncrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)  {  try  {  byte[] encryptedData;  //Create a new instance of RSACryptoServiceProvider.  using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())  {  //Import the RSA Key information. This only needs  //toinclude the public key information.  RSA.ImportParameters(RSAKeyInfo);  //Encrypt the passed byte array and specify OAEP padding.  //OAEP padding is only available on Microsoft Windows XP or  //later.  encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding);  }  return encryptedData;  }  //Catch and display a CryptographicException  //to the console.  catch (CryptographicException e)  {  Console.WriteLine(e.Message);  return null;  }  }  public static byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)  {  try  {  byte[] decryptedData;  //Create a new instance of RSACryptoServiceProvider.  using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())  {  //Import the RSA Key information. This needs  //to include the private key information.  RSA.ImportParameters(RSAKeyInfo);  //Decrypt the passed byte array and specify OAEP padding.  //OAEP padding is only available on Microsoft Windows XP or  //later.  decryptedData = RSA.Decrypt(DataToDecrypt, DoOAEPPadding);  }  return decryptedData;  }  //Catch and display a CryptographicException  //to the console.  catch (CryptographicException e)  {  Console.WriteLine(e.ToString());  return null;  }  }  }  Шифрование:  Hash:    Окно отладки в памяти:    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. Код для генерации несколких случайных чисел можно увидеть в листинге 1.  const array = new Uint8Array(1);  for (let i=0; i<5; ++i) {  console.log(crypto.getRandomValues(array));  }  Листинг 1 – Код генерации случайных чисел  **Хеширование**  Хеширование в SubtleCrypto представлено следующими алгоритмами.   * Sha1 * Sha256 * Sha385 * Sha512   Метод SubtleCrypto.digest() используется для создания хеша сообщения. Ниже представлен пример для хеширования сообщения представлен в листинге 2.  const text = 'Hash using sha256';  async function digestMessage(message) {  const encoder = new TextEncoder();  const data = encoder.encode(message);  const hash = await crypto.subtle.digest(**'SHA-256'**, data);  return hash;  }  digestMessage(text)  .then((digestBuffer) => console.log(digestBuffer.byteLength));  Листинг 2 – Пример хеширования данных  Как видно из примера выше то для указания алгоритма хеширования используется первый параметр. Также стоит упомянуть что результат хеширования обычно используется в 16-ой форме. Для преобразования стоит добавить код представленный в листинге 3.  const hashArray = Array.from(new Uint8Array(hash));  const hashHex = hashArray.map((b) => b.toString(16).padStart(2, '0')).join('');  return hashHex;  Листинг 3 – Пример преобразование хеша в 16-ричную форму.  **Генерация ключей**  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.   Пример генерация ключа представлен в листинге 4.  (async function() {  const params = {  name: 'AES-CTR',  length: 128  };  const keyUsages = ['encrypt', 'decrypt'];  const key = await crypto.subtle.generateKey(params, false, keyUsages);  console.log(key);  })();  Листинг 4 – Пример генерации ключей.  **Шифрование и дешифрование**  Объект SubtleCrypto позволяет использовать как открытый ключ, так и симметричные алгоритмы для шифрования и дешифрования сообщений. Это может быть выполнено с использованием методов SubtleCrypto.encrypt() и SubtleCrypto.decrypt() соответственно. Ниже представлена часть кода для шифрования и дешифрования данных где algoIdentifier это название алгоритма и представлена в листинге 5.  const originalPlaintext = (new TextEncoder()).encode('Crypto');  const encryptDecryptParams = {  name: algoIdentifier,  iv: crypto.getRandomValues(new Uint8Array(16))  };  const ciphertext = await crypto.subtle.encrypt(encryptDecryptParams, key,  originalPlaintext);  console.log(ciphertext);  // ArrayBuffer(32) {}  const decryptedPlaintext = await crypto.subtle.decrypt(encryptDecryptParams,  key, ciphertext);  console.log((new TextDecoder()).decode(decryptedPlaintext));  Листинг 5 – Часть кода для шифрования и дешифрования.  **Создание цифровой подписи и проверка сообщений**  Объект SubtleCrypto позволяет использовать алгоритмы с открытым ключом для генерации подписей с использованием закрытого ключа или для проверки подписей с использованием открытого ключа. Они выполняются с использованием методов SubtleCrypto.sign() и SubtleCrypto.verify() соответственно. Для подписания сообщения требуется объект params, чтобы указать алгоритм и любые необходимые значения, частный CryptoKey и ArrayBuffer или ArrayBufferView для подписи. В примере, представленном в листинге 6 можно увидеть процесс создания цифровой подписи и проверки сообщения с цифровой подписью.  async function() {  const keyParams = {  name: 'ECDSA',  namedCurve: 'P-256'  };  const keyUsages = ['sign', 'verify'];  const {publicKey, privateKey} = await crypto.subtle.generateKey(keyParams,  true, keyUsages);  const message = (new TextEncoder()).encode('Mes to sign');  const signParams = {  name: 'ECDSA',  hash: 'SHA-256'  };  const signature = await crypto.subtle.sign(signParams, privateKey, message);  const verified = await crypto.subtle.verify(signParams, publicKey, signature,  message);  console.log(verified); // true  })();  Листинг 6 – Пример создания и проверки сообщений с цифровой подписью.  **Упаковка и распаковка ключа**  Объект SubtleCrypto позволяет упаковывать и распаковывать ключи, чтобы обеспечить передачу по ненадежному каналу. Это выполняется с использованием методов SubtleCrypto.wrapKey() и SubtleCrypto.unwrapKey() соответственно. Для переноса ключа требуется строка форматирования, экземпляр CryptoKey для переноса, CryptoKey для выполнения переноса и объект params для указания алгоритма и любых необходимых значений. В примере представленном в листинге 7 симметричный ключ AES-GCM, упаковывается с помощью AES-KW и, распаковывается обратно.  (async function() {  const keyFormat = 'raw';  const extractable = true;  const wrappingKeyAlgoIdentifier = 'AES-KW';  const wrappingKeyUsages = ['wrapKey', 'unwrapKey'];  const wrappingKeyParams = {  name: wrappingKeyAlgoIdentifier,  length: 256};  const keyAlgoIdentifier = 'AES-GCM';  const keyUsages = ['encrypt'];  const keyParams = {  name: keyAlgoIdentifier,  length: 256};  const wrappingKey = await crypto.subtle.generateKey(wrappingKeyParams,  extractable, wrappingKeyUsages);  console.log(wrappingKey);  const key = await crypto.subtle.generateKey(keyParams, extractable, keyUsages);  console.log(key);  const wrappedKey = await crypto.subtle.wrapKey(keyFormat, key, wrappingKey,  wrappingKeyAlgoIdentifier);  console.log(wrappedKey);  const unwrappedKey = await crypto.subtle.unwrapKey(keyFormat, wrappedKey,  wrappingKey, wrappingKeyParams, keyParams, extractable, keyUsages);  console.log(unwrappedKey);  })()  Листинг 7 – Пример упаковки и распаковки ключа.  //AES-CTR  //SHA-1  async function importKey() {      return window.crypto.subtle.importKey(          "jwk", //может быть "jwk" или "raw"          {   //пример jwk ключа, "raw" должен быть ArrayBuffer              kty: "oct",              k: "Y0zt37HgOx-BY7SQjYVmrqhPkO44Ii2Jcb9yydUDPfE",              alg: "A256GCM",//AES-CTR              ext: true,          },          {   //это варианты алгоритма, которые вы планируете использовать              name: "AES-GCM",          },          false, //является ли ключ извлекаемым (т.е. может быть использован в exportKey)          ["encrypt", "decrypt"] //может быть "encrypt", "decrypt", "wrapKey", or "unwrapKey"      )  }  async function generateKey() {      return window.crypto.subtle.generateKey(          {              name: "AES-GCM",              length: 256, //может быть 128, 192, or 256              hash: "SHA-1", //может быть "SHA-1", "SHA-256", "SHA-384", or "SHA-512"          },          true, //является ли ключ извлекаемым (т.е. может быть использован в exportKey)          ["encrypt", "decrypt"] //может быть "encrypt", "decrypt", "wrapKey", or "unwrapKey"      )  }    async function encrypt(data, key, iv) {      return window.crypto.subtle.encrypt(          {              name: "AES-GCM",              //Не используйте повторно векторы инициализации!              //Всегда генерируйте новый iv при каждом шифровании!              //Рекомендуется использовать длину 12 байт              iv: iv,              //Длина тега (необязательно)              tagLength: 128, //can be 32, 64, 96, 104, 112, 120 or 128 (default)          },          key, //из generateKey или importKey выше          data //ArrayBuffer данных, которые вы хотите зашифровать      )  }  async function decrypt(data, key, iv) {      return window.crypto.subtle.decrypt(          {              name: "AES-GCM",              iv: iv, //Вектор инициализации, который вы использовали для шифрования.              //additionalData: ArrayBuffer, //Дополнительные данные, которые вы использовали для шифрования (если есть)              tagLength: 128, //Длина метки, которую вы использовали для шифрования (если есть)          },          key, //из generateKey или importKey выше          data //ArrayBuffer данных      )  }    async function fun() {      var keys = await importKey()      var iv = new Uint8Array([188, 185, 57, 146, 246, 194, 114, 34, 12, 80, 198, 77])      var enc = new TextEncoder();      var data = enc.encode("Халалеенко")      var encryptedData = await encrypt(data, keys, iv)      //document.write(encryptedData + '<br>');      var decryptedData = await decrypt(encryptedData, keys, iv)      //document.write(decryptedData + '<br>');      var enc = new TextDecoder("utf-8");      document.write(enc.decode(encryptedData) + '<br>');      document.write(enc.decode(decryptedData) + '<br>');  }  fun() |