|  |  |  |
| --- | --- | --- |
| ***1)Оценка надежности пароля. Основные атаки, используемые при подборе пароля, и способы противодействия им. Рекомендации по формированию пароля.***  Основные атаки:  1.полный перебор (еще называется методом грубой силы brute force). Этот метод позволяет подобрать любой пароль, однако время, необходимое для этого, может оказаться колоссальным. Действенной защитой от этого вида атак является выбор достаточно длинных паролей (например, 10 и более символов) и расширение алфавита пароля (например, при добавлении больших и маленьких русских букв получаем , что в случае дает (22 разряда)). Следует обратить внимание, что в вышеприведенных рассуждениях, предполагается, что при взломе известна длина пароля. Однако, если известна только максимально возможная длина пароля , то время расчета увеличивается. Следует отметить, что с учетом роста вычислительных мощностей современного аппаратного обеспечения и применением многопоточных вычислений количество перебираемых в секунду паролей может быть существенно выше, чем 1000000. Это, в свою очередь, потребует увеличения длины пароля для сохранения устойчивости к атаке полным перебором;  2.атака по словарю. Метод предполагает, что пароль является обычным словом, т.к. его легче запомнить. Суть атаки заключается в подстановке слов из некоторого словаря, которые могут быть паролем (с учетом максимально возможной длины пароля, если она известна). Хорошей защитой от этой атаки является выбор пароля, который не является словом;  3.атака по слогам. Этот метод является разновидностью предыдущего и предполагает подбор пароля из слогов, из которых строятся слова на языке формирования пароля. Временные затраты для этого метода выше, чем для атаки по словарю. Защитой от такой атаки является отказ от использования слов (даже несколько видоизмененных) в качестве пароля;  4.атака на основе правил. Является самой мощной из атак, направленных на раскрытие пароля. Суть атаки является в комбинации полного перебора с некоторыми эвристическими знаниями о том, как формировался пароль (атака возможно ТОЛЬКО в случае наличия таких знаний). Например, если известно, что в качестве пароля используется слово в комбинации с цифрами, то перебор можно выполнять с использованием такого шаблона. Разновидностью атаки такого вида является также приоритетность в переборе паролей: сначала могут быть рассмотрены часто используемые пароли, сочетания символов, имеющих определенное значение для человека, формировавшего пароль (день рождения, имя любимого родственника, марка любимого автомобиля, кличка собаки и т.п. Такая атака еще называется атакой на основе дней рождения).  Рекомендации по формированию пароля:  1.не задавать общеизвестные пароли типа 111, 123, qwerty и т.п., т.к. их будут перебирать в первую очередь;  2.не задавать пароли, которые имеют непосредственное отношение к формирующему их человеку. Например, дату рождения (свою или родственника), имя родственника или домашнего животного и т.п. Несоблюдение этой рекомендации может привести к существенному сокращению времени подбора паролей, если использовать атаку на основе правил, и будут накоплены данные о личной жизни задавшего пароль;  3.не использовать в качестве пароля слова, т.к. это позволяет использовать атаку по словарю или атаку по слогам;  4.не использовать один и тот же шаблон при формировании паролей, т.к. это позволяет использовать атаку на основе правил;  5.не использовать слишком коротких паролей.  6.не использовать всегда пароли одной и той же длины, особенно если это может стать известным злоумышленнику, т.к. это несколько сокращает время подбора;  7.несмотря на все вышеперечисленные требования, пароль должен быть пригоден для запоминания, иначе его можно забыть.  Для борьбы с взломом паролей могут быть использованы различные методы: искусственное постепенное увеличение времени ответа об удачности авторизации при превышении неудачных вводов пароля некоторого заданного значения, блокировка логина, для которого выполняется большое количество неудачных попыток подключения и т.п. | ***2) Криптография. Основные понятия и определения.***  Основные направления использования криптографических методов − передача конфиденциальной информации по каналам связи (например, электронная почта), установление подлинности передаваемых сообщений, хранение информации (документов, баз данных) на носителях в зашифрованном виде.  Криптографические методы защиты информации в автоматизированных системах могут применяться как для защиты информации, обрабатываемой в ЭВМ или хранящейся на различного типа ЗУ, так и для закрытия информации, передаваемой между различными элементами системы по линиям связи. Криптографическое преобразование как метод предупреждения несанкционированного доступа к информации имеет многовековую историю. В настоящее время разработано большое количество различных методов шифрования, созданы теоретические и практические основы их применения.  Итак, криптография дает возможность преобразовать информацию таким образом, что ее прочтение (восстановление) возможно только при знании ключа.  В качестве информации, подлежащей шифрованию и дешифрованию, будут рассматриваться тексты, построенные на некотором алфавите. Под этими терминами понимается следующее:  **алфавит** − конечное множество используемых для кодирования информации знаков. Примерами алфавита могут служить большие и маленькие буквы латинского алфавита (алфавит Z52), символы, входящие в стандартные коды ASCII (алфавит Z256) и т.п.;  **текст** − набор из элементов алфавита;  **шифрование** − процесс преобразования **исходного текста (открытого текста)** путем замены его **шифрованным текстом**;  **дешифрование** − обратный шифрованию процесс. На основе ключа шифрованный текст преобразуется в исходный;  **ключ** − информация, необходимая для беспрепятственного шифрования и дешифрования текста;  **крип­то­гра­фи­че­ская сис­те­ма**пред­став­ля­ет со­бой се­мей­ст­вопре­об­ра­зо­ва­ний от­кры­то­го тек­ста. Криптосистемы разделяются на **симметричные** и **с открытым ключом**;  в **симметричных криптосистемах** и для шифрования, и для дешифрования используется **один и тот же ключ** (секретный ключ), который должен быть известен только узкому кругу лиц;  в **системах с открытым ключом** используются два ключа − **открытый** и **закрытый**, которые математически связаны друг с другом. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения;  **крип­то­стой­ко­стью**на­зы­ва­ет­ся ха­рак­те­ри­сти­ка шиф­ра, оп­ре­де­ляю­щая его стой­кость к де­шиф­ро­ва­нию без зна­ния клю­ча (т.е. **крип­тоа­на­ли­зу**). Имеется несколько показателей криптостойкости, среди которых:  количество всех возможных ключей;  среднее время, необходимое для криптоанализа;  стоимость вычисления ключевой информации или исходного текста.  Для со­вре­мен­ных крип­то­гра­фи­че­ских сис­тем за­щи­ты ин­фор­ма­ции сфор­му­ли­ро­ва­ны сле­дую­щие об­ще­при­ня­тые тре­бо­ва­ния:   1. за­шиф­ро­ван­ное сообщение дол­жно под­да­вать­ся чте­нию толь­ко при на­ли­чии клю­ча; 2. чис­ло опе­ра­ций, не­об­хо­ди­мых для оп­ре­де­ле­ния ис­поль­зо­ван­но­го клю­ча шиф­ро­ва­ния по фраг­мен­ту шиф­ро­ван­но­го сообщения и со­от­вет­ст­вую­ще­го ему от­кры­то­го тек­ста, долж­но быть не мень­ше об­ще­го чис­ла воз­мож­ных клю­чей; 3. чис­ло опе­ра­ций, не­об­хо­ди­мых для рас­шиф­ро­вы­ва­ния ин­фор­ма­ции пу­тем пе­ре­бо­ра все­воз­мож­ных ключей долж­но иметь стро­гую ниж­нюю оцен­ку и вы­хо­дить за пре­де­лы воз­мож­но­стей со­вре­мен­ных ком­пь­ю­те­ров (с учетом возможности использования сетевых вычислений); 4. зна­ние ал­го­рит­ма шиф­ро­ва­ния не долж­но вли­ять на на­деж­ность за­щи­ты; 5. не­зна­чи­тель­ное из­ме­не­ние клю­ча долж­но при­во­дить к су­ще­ст­вен­но­му из­ме­не­нию ви­да за­шиф­ро­ван­но­го сообщения да­же при ис­поль­зо­ва­нии од­но­го и то­го же сообщения; 6. струк­тур­ные эле­мен­ты ал­го­рит­ма шиф­ро­ва­ния долж­ны быть не­из­мен­ны­ми; 7. до­пол­ни­тель­ные би­ты, вво­ди­мые в сообщение в про­цес­се шиф­ро­ва­ния, должен быть пол­но­стью и на­деж­но скры­ты в шиф­ро­ван­ном тек­сте; 8. дли­на шиф­ро­ван­но­го тек­ста долж­на быть рав­ной дли­не ис­ход­но­го тек­ста; 9. не долж­но быть про­стых и лег­ко ус­та­нав­ли­вае­мых зависимостей ме­ж­ду клю­ча­ми, по­сле­до­ва­тель­но ис­поль­зуе­мы­ми в про­цес­се шиф­ро­ва­ния; 10. лю­бой ключ из мно­же­ст­ва возможных дол­жен обес­пе­чи­вать на­деж­ную за­щи­ту ин­фор­ма­ции; 11. ал­го­ритм должен до­пус­кать как про­грамм­ную, так и ап­па­рат­ную реа­ли­за­цию, при этом из­ме­не­ние длины к­лю­ча не долж­но вес­ти к ка­че­ст­вен­но­му ухуд­ше­нию алгоритма шифрования. | ***3) Криптоанализ. Основные понятия и определения.***  Главным действующим лицом при криптоанализе выступает нарушитель (или краптоаналитик). Под ним понимают лицо или группу лиц, целью которых является прочтение или подделка защищенных криптографическими методами сообщений.  В отношении нарушителя принимается ряд допущений, которые, как правило, являются основой математических или иных моделей:  1. нарушитель знает алгоритм шифрования и особенности его реализации в конкретном случае, но не знает секретного ключа;  2. нарушителю доступны все зашифрованные тексты, а также могут быть доступны некоторые исходные тексты, для которых известны соответствующие им зашифрованные тексты;  3. нарушитель имеет в своем распоряжении вычислительные, людские, временные и иные ресурсы, объем которых соответствует потенциальной ценности информации, которая может быть добыта в результате криптоанализа.  Попытку прочтения или подделки зашифрованного сообщения, вычисления ключа методами криптоанализа называют криптоатакой или атакой на шифр. Удачную криптоатаку называют взломом.  Все методы криптоанализа в целом укладываются в 4 направления:  1. статистический криптоанализ. Исследует возможности взлома криптосистем на основе изучения статистических закономерностей исходных и зашифрованных сообщений. Его применение осложнено тем, что в реальных криптосистемах информация перед шифрованием подвергается сжатию;  2. алгебраический криптоанализ. Нацелен на поиск математически слабых звеньев криптоалгоритмов;  3. дифференциальный (или разностный) криптоанализ. Основан на анализе зависимости изменения шифрованного текста от изменения исходного текста.  4. линейный криптоанализ. Метод, основанный на поиске линейной аппроксимации между исходным и шифрованным текстом.  Следует отметить, что главным методом, однако, остается «лобовая атака» (метод грубой силы – brute force)  подбор ключа.  В целом, можно выделить 3 уровня криптоатаки по нарастанию сложности:  1. атака по шифрованному тексту  нарушителю доступны все или некоторые зашифрованные сообщения;  2. атака по паре «исходный текст  зашифрованный текст». Нарушителю доступны все или некоторые зашифрованные сообщения и соответствующие им исходные сообщения;  3. атака по выбранной паре «исходный текст  шифрованный текст». Нарушитель имеет возможность выбирать исходный текст, получать для него шифрованный текст и на основе анализа зависимостей между ними вычислять ключ. |
| ***4)*** ***Симметричные алгоритмы шифрования. Типы симметричных алгоритмов и представители этих видов (с кратким описанием алгоритма).***  Все многообразие су­ще­ст­вую­щих симметричных крип­то­гра­фи­че­ских ме­то­дов мож­но све­сти к сле­дующим клас­сам пре­об­ра­зо­ва­ний:  **многоалфавитная подстановка (**или **подстановка)** − наи­бо­лее про­стой вид пре­об­ра­зо­ва­ний, за­клю­чаю­щий­ся в за­ме­не сим­во­лов ис­ход­но­го тек­ста на другие (того же алфавита) по бо­лее или ме­нее слож­но­му пра­ви­лу. Для обес­пе­че­ния вы­со­кой крип­то­стой­ко­сти тре­бу­ет­ся ис­поль­зо­ва­ние боль­ших клю­чей.  **пе­ре­ста­нов­ка** − не­слож­ный ме­тод крип­то­гра­фи­че­ско­го пре­об­ра­зо­ва­ния. Ис­поль­зу­ет­ся, как пра­ви­ло, в со­че­та­нии с дру­ги­ми ме­то­да­ми;  **гам­ми­ро­ва­ние** − этот ме­тод за­клю­ча­ет­ся в на­ло­же­нии на ис­ход­ный текст не­ко­то­рой псев­до­слу­чай­ной по­сле­до­ва­тель­но­сти, ге­не­ри­руе­мой на ос­но­ве клю­ча;  **блочные шифры** представляют со­бой по­сле­до­ва­тель­ность (с воз­мож­ным по­вто­ре­ни­ем и че­ре­до­ва­ни­ем) ос­нов­ных ме­то­дов пре­об­ра­зо­ва­ния, при­ме­няе­мую к блоку (части) шиф­руе­мого­ тек­ста. Блочные шифры на прак­ти­ке встре­ча­ют­ся ча­ще, чем «чис­тые» пре­об­ра­зо­ва­ния то­го или ино­го клас­са в си­лу их бо­лее вы­со­кой крип­то­стой­ко­сти. Рос­сий­ский и аме­ри­кан­ский стан­дар­ты шиф­ро­ва­ния ос­но­ва­ны имен­но на этом классе шифров.  Одним из таких алгоритмов шифрования являются моноалфавитные подстановки (секретный ключ фактически отсутствует, т.к. представляет собой единственную подстановку), примером чего служит шифр Цезаря.  При использовании шифра Цезаря над каждым символом исходного текста выполняется следующее преобразование:  .  Из этого преобразования следует, что в данном алгоритме отсутствует секретный ключ, что делает его крайне уязвимым, т.к. злоумышленнику достаточно знать суть алгоритма для удачного взлома. Дешифрование происходит с помощью обратного сдвига номера символа шифрованного текста в выбранном алфавите на 3 символа «влево» (вычитание числа 3). Если в результате этого получаем отрицательное значение, то к нему добавляется n .  К классу перестановок относится шифр маршрутная транспозиция и его вариант постолбцовая транспозиция. Суть этого метода заключается в том, что при шифровании в прямоугольник размерности  , где N - длина исходного текста, а lmax длина секретного ключа, построчно вписывается исходный текст. Зашифрованный текст формируется путем выбора (в прямом или обратном порядке) символов каждого столбца полученного прямоугольника. При этом столбцы берутся или по порядку (от 1-го до последнего), или в порядке следования букв пароля в алфавите.  Еще одним достаточно простым методом шифрования является гаммирование, сутью которого является выполнение следующего преобразования:    где kl очередной символ секретного ключа, длина которого равна lmax, а сложение по модулю 2 (эквивалентно исключающему «или»). Повышение криптостойкости данного алгоритма может осуществляться за счет предварительного сжатия исходного текста для усложнения статистического криптоанализа и за счет увеличения длины секретного ключа. Гаммирование является составной частью многих алгоритмов шифрования. | ***5) Симметричные алгоритмы, основанные на моно- или многоалфавитной подстановке. Примеры таких алгоритмов (с подробным описанием процессов шифрования и дешифрования). Способы повышения их криптостойкости.***  **многоалфавитная подстановка (**или **подстановка)** − наи­бо­лее про­стой вид пре­об­ра­зо­ва­ний, за­клю­чаю­щий­ся в за­ме­не сим­во­лов ис­ход­но­го тек­ста на другие (того же алфавита) по бо­лее или ме­нее слож­но­му пра­ви­лу. Для обес­пе­че­ния вы­со­кой крип­то­стой­ко­сти тре­бу­ет­ся ис­поль­зо­ва­ние боль­ших клю­чей.  Одним из таких алгоритмов шифрования являются моноалфавитные подстановки (секретный ключ фактически отсутствует, т.к. представляет собой единственную подстановку), примером чего служит шифр Цезаря.  При использовании шифра Цезаря над каждым символом исходного текста выполняется следующее преобразование:    Из этого преобразования следует, что в данном алгоритме отсутствует секретный ключ, что делает его крайне уязвимым, т.к. злоумышленнику достаточно знать суть алгоритма для удачного взлома. Дешифрование происходит с помощью обратного сдвига номера символа шифрованного текста в выбранном алфавите на 3 символа «влево» (вычитание числа 3). Если в результате этого получаем отрицательное значение, то к нему добавляется .  Существует модифицированный шифр Цезаря (предложен аббатом Тритемеусом), который описывается следующим преобразованием:    где ikl порядковый номер в алфавите Zn l-го символа ключа. В этом преобразовании уже присутствует секретный ключ, что несколько повышает криптостойкость данного алгоритма. Если ключ содержит 2 или более различных символов, то этот шифр будет представлять уже многоалфавитную подстановку, т.к. будет содержать не менее 2-ух различных подстановок. Дешифрование модифицированного шифра Цезаря происходит аналогично дешифрованию базового шифра, но сдвиги «влево» зависят от значения символа секретного ключа. Еще одним способом дешифрования многоалфавитной подстановки является следующее преобразование:    Этот способ дешифрования не содержит сравнений, но в нем присутствует деление.  Развитием шифра Цезаря является система шифрования Виженера, представляющая собой многоалфавитную перестановку. Основным отличием от шифра Цезаря является то, что ключ пользователя продлевается до бесконечности путем его повторения. Таким образом, в бесконечном ключе наблюдается период r, с которым его символы повторяются. | ***6)*** ***Симметричные алгоритмы, основанные на перестановке. Примеры таких алгоритмов (с подробным описанием процессов шифрования и дешифрования). Способы повышения их криптостойкости.***  **пе­ре­ста­нов­ка** − не­слож­ный ме­тод крип­то­гра­фи­че­ско­го пре­об­ра­зо­ва­ния. Ис­поль­зу­ет­ся, как пра­ви­ло, в со­че­та­нии с дру­ги­ми ме­то­да­ми;  К классу перестановок относится шифр маршрутная транспозиция и его вариант постолбцовая транспозиция. Суть этого метода заключается в том, что при шифровании в прямоугольник размерности  , где N - длина исходного текста, а lmax длина секретного ключа, построчно вписывается исходный текст. Зашифрованный текст формируется путем выбора (в прямом или обратном порядке) символов каждого столбца полученного прямоугольника. При этом столбцы берутся или по порядку (от 1-го до последнего), или в порядке следования букв пароля в алфавите.  Шифрование согласно порядка следования букв в пароле (именно порядка, а не самих букв) предпочтительнее, т.к. зашифрованный текст зависит не только от длины, но и от содержимого пароля. При дешифровании зашифрованный текст необходимо вписать в прямоугольник по соответствующим столбцам, а потом прочитать исходный текст по строкам, начиная с первой. |
| ***7)*** ***Симметричные алгоритмы, основанные на гаммировании. Примеры таких алгоритмов (с подробным описанием процессов шифрования и дешифрования). Способы повышения их криптостойкости.***  **гам­ми­ро­ва­ние** − этот ме­тод за­клю­ча­ет­ся в на­ло­же­нии на ис­ход­ный текст не­ко­то­рой псев­до­слу­чай­ной по­сле­до­ва­тель­но­сти, ге­не­ри­руе­мой на ос­но­ве клю­ча;  Еще одним достаточно простым методом шифрования является гаммирование, сутью которого является выполнение следующего преобразования:    где kl очередной символ секретного ключа, длина которого равна lmax, а сложение по модулю 2 (эквивалентно исключающему «или»). Повышение криптостойкости данного алгоритма может осуществляться за счет предварительного сжатия исходного текста для усложнения статистического криптоанализа и за счет увеличения длины секретного ключа. Гаммирование является составной частью многих алгоритмов шифрования.  Другим часто используемым подходом к генерации бесконечного ключа является использование генератора псевдослучайных чисел. | ***8)*** ***Статистический криптоанализ. Особенности его применения при взломе простых шифров.***  Исследует возможности взлома криптосистем на основе изучения статистических закономерностей исходных и зашифрованных сообщений. Его применение осложнено тем, что в реальных криптосистемах информация перед шифрованием подвергается сжатию;  Одной из достаточно эффективных атак при использовании простых алгоритмов шифрования является статистический криптоанализ (частным случаем которого является частотный криптоанализ). В частности, для моноалфавитных перестановок и простых подстановок имея таблицу частот символов для русского языка можно легко осуществить взлом. Однако, при использовании многоалфавитных подстановок и ключей большей, чем 1, длины такой подход будет неэффективен. В этом случае необходимо сначала определить длину пароля. Это можно сделать на основании того, что при гаммировании и многоалфавитных перестановках для повторяющихся последовательностей исходного текста и символов ключа получаем одинаковые последовательности на выходе. Соответственно, анализ зашифрованного текста достаточной длины на предмет поиска повторяющихся последовательностей позволяет (теоретически) определить длину ключа. Суть такого поиска заключается в анализе зашифрованного текста на предмет поиска интервала повторения символов. Т.е., начиная с длины 1 и до предполагаемой/известной максимально возможной длины пароля, производится анализ символов, отстоящих друг от друга на n , где n − текущая (предполагаемая) длина ключа. Далее происходит частотный анализ lmax фрагментов зашифрованного текста. Естественно, такой подход возможен только в случает относительно небольших паролей и достаточно длинных текстов.  Следует также отметить, что при частотном анализе текста можно получать близкие (практически одинаковые) значения частот для разных символов, что усложняет процесс взлома. Однако, для улучшения результатов взлома может быть реализован автоматический (или полуавтоматический/ручной) анализ текста с целью замены слов с неправильно дешифрованными символами на правильные аналоги (в случае шифрования текстовой информации). После такой замены есть возможность для относительно небольших длин пароля получить его символы, что позволит корректно дешифровать весь текст. | ***9)*** ***Блочные алгоритмы шифрования. Сеть Фейштеля. Особенности такого подхода при реализации алгоритма блочного шифрования.***  Сеть Фейштеля представляет собой блочный алгоритм, преобразующий n-битный блок незашифрованного текста в n-битный блок зашифрованного текста. Число блоков длины равно 2n. Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста. При маленькой длине блока такая подстановка плохо скрывает статистические особенности незашифрованного текста. Если блок имеет длину 64 бита, то он уже хорошо скрывает статистические особенности исходного текста. Но в данном случае преобразование текста не может быть произвольным в силу того, что ключом будет являться само преобразование, что исключает эффективную как программную, так и аппаратную реализации.  Сети Фейштеля получили широкое распространение, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны.  Сеть Фейштеля имеет следующую структуру: входной блок делится на несколько равной длины подблоков, называемых ветвями. Если блок имеет длину 64 бита, используются две ветви по 32 бита каждая. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево. Такое преобразование выполняется несколько циклов или раундов. В случае двух ветвей каждый раунд имеет структуру, показанную на рис    Функция F называется образующей. Каждый раунд состоит из вычисления функции F для одной ветви и побитового выполнения операции (исключающее «ИЛИ» или XOR) результата F с другой ветвью. После этого ветви меняются местами. Считается, что оптимальное число раундов от 8 до 32. Важно то, что увеличение количества раундов значительно увеличивает криптостойкость алгоритма. Возможно, эта особенность и повлияла на столь активное распространение сети Фейштеля. В последнее время количество раундов не фиксируется, а лишь указываются допустимые пределы.  Сеть Фейштеля является обратимой даже в том случае, если функция F не является таковой, так как для дешифрования не требуется вычислять F-1 . Для дешифрования используется тот же алгоритм, но на вход подается зашифрованный текст, и ключи используются в обратном порядке.  Часто используются различные разновидности сети Фейштеля для 128-битного блока с 4-мя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что процессоры с 32-разрядными словами до сих пор остаются достаточно часто используемыми, следовательно, оперировать 32-разрядными словами эффективнее, чем с 64-разрядными.  Основной характеристикой алгоритма, построенного на основе сети Фейштеля, является функция F . Различные варианты касаются также начального и конечного преобразований. Подобные преобразования, называемые забеливанием (whitening), осуществляются для того, чтобы выполнить начальную рандомизацию входного текста. |
| ***0) Алгоритм шифрования DES. Характеристики и подробное описание процесса генерации ключей раундов (табличные преобразования точно приводить не обязательно).***  DES является классической сетью Фейштеля с двумя ветвями.  Ключ K(i) формируется на каждой итерации из исходного 64-битного ключа K . 8 из этих 64 бит используются для контроля по четности (добавляются так, чтобы количество единиц в каждом байте было нечетным). Номера позиций контрольных бит ключа: 8, 16, 24, 32, 40, 48, 56, 64 (если номер 1-го бита 1). Рассмотрим процесс формирования 16-ти 48-битных ключей из исходного 64-битного:  а)к исходному ключу K применяется функция G, что позволяет избавиться от битов контроля по четности и выполнить перестановку оставшихся 56 битов;  б)результат преобразования предыдущего шага разбивается на два 28-битных блока C(0) и D(0) , которые формируются согласно таблице 12 (относительно исходного значения. Т.е. применяется либо функция G, либо функции C(0) и D(0);  в)при формировании каждого из 16-ти 48-битных ключей происходит следующее: вычисляются текущие значения C(i) и D(i) путем циклического сдвига значений C(i-1) и D(i-1) на 1 или 2 бита влево (количество битов сдвига определяется номером итерации и отображается в таблице 13). После этого полученное путем конкатенации и 56-битное значение обрабатывается с помощью функции H, в результате чего и формируется очередной 48-битный ключ. | ***11)*** ***Алгоритм шифрования DES. Характеристики и подробное описание процесса шифрования / дешифрования (табличные преобразования точно приводить не обязательно).***  DES является классической сетью Фейштеля с двумя ветвями.  Данные алгоритмом DES шифруются 64-битными блоками, используя 56-битный ключ (на самом деле ключ 64-битный, содержащий 8 бит контроля по четности, причем расширение ключа до 64 бит должно проводиться автоматически программным обеспечением). Алгоритм преобразует за 16 раундов 64-битный вход в 64-битный выход. Дешифрование в DES является операцией обратной шифрованию. Она выполняется путем повторения операций шифрования в обратной последовательности.    1.блок исходного текста (64-бита), который подвергается шифрованию и обозначается T, преобразуется с помощью матрицы начальной перестановки IP.Начальная перестановка выполняется следующим образом: 58-ой бит исходного текста становится 1-ым, 50-ый бит - 2-ым и т. д. Аналогично применяются все последующие табличные перестановки и расширения;  2.выполняются 16 раундов применения одной и той же функции F, которая использует операции сдвига и подстановки. Аргументами этой функции всегда являются 32-битная последовательность R(i-1) - правая часть зашифрованного на предыдущей итерации 64-битного блока исходного текста и 48-битный ключ K(i) - результат преобразования для текущей итерации исходного 64-битного ключа K.  Ключ формируется на каждой итерации из исходного 64-битного ключа K. Как уже говорилось выше, 8 из этих 64 бит используются для контроля по четности (добавляются так, чтобы количество единиц в каждом байте было нечетным). Номера позиций контрольных бит ключа: 8, 16, 24, 32, 40, 48, 56, 64 (если номер 1-го бита 1).  3. после выполнения 16-го раунда левая и правая половина текущего блока исходного текста обмениваются местами, и из них путем конкатенации составляется зашифрованный 64-битный блок;  4. выполняется формирование результата шифрования с помощью обратной перестановки IP-1 64-битного значения, полученного на предыдущем этапе.  Дешифрование текущего 64-битного блока происходит таким же образом, как и шифрование, но 48-битные ключи берутся в обратном порядке: на 1-ой итерации используется ключ K(16), а на последней – K(1) . | ***12)Алгоритм шифрования 3DES. Характеристики и подробное описание алгоритма. Отличия от DES.***  Алгоритм шифрования 3DES на основе алгоритма DES, с целью устранения главного недостатка последнего - малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа (с учетом развития современных технологий). Скорость работы алгоритма 3DES в 3 раза ниже, чем у DES, однако криптостойкость намного выше. Алгоритм 3DES является самым простым в реализации способом устранения недостаточной криптостойкости алгоритма DES.  Существует 3 типа алгоритма 3DES:  1. DES-EEE3. Шифрование каждого 64-битного блока данных происходит следующим образом:  (1)  где DES - применение алгоритма шифрования DES, а K1, K2, K3 - различные 56-битные ключи шифрования.  2. DES-EDE3. Шифрование каждого 64-битного блока данных происходит следующим образом:  (2)  где DES-1 применение алгоритма DES для дешифрования.  3. DES-EEE2 или DES-EDE2. Шифрование каждого 64-битного блока данных происходит по формулам (1) или (2), соответственно, но K1=K3 .  Возможен также вариант применения 3DES, когда шифрование происходит по формуле (1), а K1=K2=K3. Дешифрование во всех случаях происходит путем троекратного применения обратных преобразований с соответствующим ключом. |
| ***13)*** ***Режимы работы алгоритмов DES и 3DES. Подробное описание.***  Рассмотрим различные режимы работы алгоритма DES или 3DES при шифровании файла данных, который представляет собой набор фрагментов T(i), () где n - число блоков. После шифрования файл представляет собой набор зашифрованных фрагментов C(i), () . Если последний блок исходного файла менее 64 бит, то он дополняется до 64 бит путем добавления символов с кодом 0 или пробелов.  1.ECB - Electronic Codebook (электронный шифроблокнот). Каждый блок Ti исходного файла шифруется алгоритмом DES или 3DES с использованием одного и того же исходного 56-битного ключа K. Основное достоинство этого режима работы - простота, однако его недостатком является относительно слабая криптоустойчивость против квалифицированных криптоаналитиков. В частности, не рекомендуется использовать данный режим работы при шифровании EXE файлов, потому что 1-ый же блок - заголовок файла, является вполне удачным началом для взлома всего шифра.  2.CBC- Cipher Block Chaining (цепочка цифровых блоков). В этом режиме шифрование происходит следующим образом:  а)1-ый блок шифруется согласно следующей формуле:    где DES - применение алгоритма DES или 3DES для шифрования 64-битного блока, причем этот блок формируется путем сложения по модулю 2 T(1) и 64-битного начального вектора  , который должен меняться ежедневно и содержаться в секрете;  б)каждый i-ый блок шифруется согласно следующей формуле:    Дешифрование происходит следующим образом:  а)1-ый блок расшифровывается по следующей формуле:    где DES-1 - применение алгоритма DES или 3DES для дешифрования 64-битного блока;  б)каждый i-ый блок расшифровывается по следующей формуле:    Достоинствами данного режима работы являются более высокая степень криптостойкости и отсутствие накопления ошибки при передаче данных, т.к. дешифрование текущего блока зависит только от него самого и предшествующего ему блока.  3.CFB - Cipher Feedback (цифровая обратная связь). В этом режиме работы предполагается, что размер Tmax каждого блока T(i) должен удовлетворять неравенству Tmaz<=4 и быть одинаковым для всех шифруемых блоков. Шифрование в этом режиме происходит с использованием 64-битового сдвигового регистра P, в котором в начальный момент времени содержится 64-битный вектор  (должен меняться для каждого нового сеанса шифрования), следующим образом: каждый i-ый блок шифруется по формуле:    где P(i-1) – старшие Tmaz бит полученных в результате выполнения шифрования алгоритмом DES или 3DES содержимого регистра P на предыдущей итерации. После шифрования i-го блока происходит удаление Tmax старших бит регистра P путем сдвига их влево и справа дописывается значение C(i).  Дешифрование происходит следующим образом: каждый i-ый блок расшифровывается по формуле:    где P(i-1) – старшие Tmaz бит полученных в результате выполнения дешифрования алгоритмом DES или 3DES содержимого регистра P на предыдущей итерации. После дешифрования i-го блока происходит удаление Tmax старших бит регистра P путем сдвига их влево и справа дописывается значение C(i).  4.OFB - Output Feedback (внешняя обратная связь). Этот режим аналогичен предыдущему, однако отличается способом обновления сдвигового регистра P : после сдвига влево Tmax бит справа дописывается значение P(i-1) .  Режим ECB хорошо подходит для шифрования ключей. Режимы CBC и CFB пригодны для аутентификации данных. Режим CFB, кроме того, предназначен для шифрования отдельных символов. Режим OFB нередко используется в спутниковых системах связи. | ***14)*** ***Алгоритм шифрования DES. Способы повышения эффективности (быстродействия) при реализации процесса шифрования/дешифрования (описать подробно).***  Общие рекомендации по практической реализации алгоритма шифрования DES одного блока данных с целью повышения производительности  (с ориентацией на синтаксис языка С++).  1. Ключи раунда генерируются 1 раз для всех блоков  2. Все таблицы (кроме S-boxes) НЕ ХРАНИТЬ в виде массивов, а использовать константные значения напрямую в коде. Константные значения прописать через директиву #define.  3. При вычислении функции E реализовать групповые сдвиги бит.  4. Значения в S-boxes должны быть изменены таким образом, чтобы входное 6-ьитное значение являлось индексом одномерного массива, который будет представлять каждый S-box. При обращении к элементу такого массива по заданному индексу должно получаться искомое 4-битное значение.  5. Т.к. для Win32 приложений работать с 64-битными значениями неэффективно, но следует 64-битные значения обрабатывать как 2 32-битных. Для хранения же блока исходных данных может быть использовании тип данных union, позволяющий интерпретировать одно и то же значение по-разному (единым блоком или структурой из 2-ух 32-битных полей, например).  6. раскрытие циклов при небольшом количестве итераций.  7. Из-за инвертирования порядка следования байт при чтении 64-битного блока (единым целым, а не массивом байт) из файла следует этот факт учитывать при изменении таблицы начальной перестановки IP. Таблица конечной перестановки IP-1 должна основа осуществлять инвертирование порядка следования байт для последующей записи в файл. | ***15) Алгоритм шифрования AES. Математические основы алгоритма.***  AES (Advanced Encryption Standard) также известный под названием Rijndael является симметричным алгоритмом блочного шифрования. Размер блока шифруемого текста составляет 128 бит, а ключ - 128/192/256 бит.  Алгоритм AES не относится к реализациям сети Фейштеля. При разработке Rijndael во внимание принималось три критерия:  1) устойчивость по отношению ко всем известным атакам;  2) быстрота исполнения и компактность кода на разных платформах;  3) простота алгоритма.  Шифрование алгоритмом AES выполняется в несколько раундов, причем их количество зависти от длины ключа: 128 бит - 10 раундов, 192 бит - 12 раундов и 256 бит - 14 раундов. |
| 16. Алгоритм шифрования AES. Характеристики алгоритма, отличия от DES и 3DES (по скорости работы и надежности). Краткое описание алгоритма.  **AES** (**Advanced Encryption Standard**) также известный под названием **Rijndael** является симметричным алгоритмом блочного шифрования. Размер блока шифруемого текста составляет 128 бит, а ключ − 128/192/256 бит. Этот алгоритм принят в качестве стандарта шифрования правительством США в 2002 г. В зависимости от длины ключа эти алгоритмы шифрования получили также названия AES-128, AES-192 и AES-256, соответственно.  Алгоритм AES не относится к реализациям сети Фейштеля. При разработке Rijndael во внимание принималось три критерия:   1. **устойчивость по отношению ко всем известным атакам;** 2. **быстрота исполнения и компактность кода на разных платформах;** 3. **простота алгоритма.**   Шифрование алгоритмом AES выполняется в несколько раундов, причем их количество зависти от длины ключа: 128 бит − 10 раундов, 192 бит − 12 раундов и 256 бит − 14 раундов.  **Рассмотрим, собственно, сам алгоритм шифрования.** Входными данными алгоритма является текущий 128-битный блок исходного текста T и секретный ключ K (128 бит, 192 бит или 256 бит). Нумерация битов происходит от 0. Выходными данными является 128-битный зашифрованный фрагмент исходного текста C.  **Алгоритм шифрования AES** включает следующие действия:   1. расширение исходного ключа K. 2. инициализация массива состояний S исходными данными T согласно правилу; 3. начальное добавление ключа  к массиву состояний S (AddRoundKey); 4. раундов шифрования (от 1-го до -го), где  − общее число раундов, зависящее от , и  − число 32-битных слов секретного ключа (может быть 4, 6 или 8 для 128-битного, 192-битного и 256-битного ключей, соответственно). Каждый из  раундов может быть представлен следующим образом:    1. 4.1. замена байт (SubBytes) для массива S;    2. 4.2. сдвиг строк (ShiftRows) для массива S;    3. 4.3. смешивание столбцов (MixColumns) для массива s;    4. 4.4. добавление ключа  к массиву состояний s (AddRoundKey); 5. последний раунд отличается от остальных тем, что не содержит шага 4.3 − смешивания столбцов. При этом используется текущий ключ ; 6. формирование выходных (зашифрованных данных) C на основании массива состояния s по правилу:   . | 17.Алгоритм шифрования AES. Подробное описание процесса шифрования.   1. **замена байт (SubBytes)**. Это действие (**называется S-box**) представляет собой нелинейную замену байтов, которая действует на каждый элемент массива состояний S. Таблица замены является обратимой и строится на основе произведения 2-ух преобразований:    1. вычисляется мультипликативная инверсия в конечном поле  по формуле . Элемент {00} отображается сам в себя.    2. к полученному после выполнения предыдущего шага байту b применяется следующее аффинное преобразование:   ,  причем  и  − текущий бит входного и выходного байтов, а  − текущий бит значения {63} ({01100011}).   1. **сдвиг строк (ShiftRows)**. Это преобразование приводит к тому, что последние три строки матрицы состояния S циклически сдвигаются на указанное (различное для каждой строки) число байт влево. При этом количество байт сдвига влево соответствует номеру строки (если строки нумеровать от 0). 1-ая по счету строка матрицы S остается без изменений. 2. **смешивание столбцов (MixColumns)**. Это действие выполняется над каждым столбцом матрицы S, который интерпретируется как 4-ех элементный вектор с коэффициентами в поле . Каждый такой вектор умножается согласно формулам на вектор a(x), который определяется соотношением (43). Таким образом, каждый столбец, который обозначается вектором s(x), преобразуется в столбец  по следующей формуле:. Таким образом, коэффициенты вектора  вычисляются по следующим формулам:   ,  ,  ,  .   1. **добавление ключа (AddRoundKey)**. Это действие заключается в изменении каждого столбца (интерпретируется, как 4-ех элементный вектор с коэффициентами в поле ) матрицы состояний S для получения матрицы s’ следующим образом:   ,  где  − матрица ключа текущего раунда. При t=0 происходит первоначальное добавление ключа на шаге 3 алгоритма шифрования. Следует отметить, что матрица  представляет собой фрагмент матрицы  размерностью  байт (результат **расширения исходного ключа K**) из 4-ех строк и  столбцов. | 18.Алгоритм шифрования AES. Подробное описание процесса дешифрования. Различные подходы к решению этой задачи.  **Алгоритм дешифрования AES** включает следующие действия:   1. расширение исходного ключа K. 2. инициализация массива состояний S исходными зашифрованными данными C согласно правилу ; 3. начальное добавление ключа  к массиву состояний S (AddRoundKey); 4. раундов шифрования, начиная от -го и до 2-го. Каждый из  раундов может быть представлен следующим образом:    1. 4.1. обратный сдвиг строк (InvShiftRows) для массива S    2. 4.2. обратная замена байт (InvSubBytes) для массива S;    3. 4.3. добавление ключа  () к массиву состояний S (AddRoundKey);    4. 4.4. обратное смешивание столбцов (InvMixColumns) для массива S; 5. последний раунд отличается от остальных тем, что не содержит шага 4.4 − обратного смешивания столбцов. При этом используется текущий ключ ; 6. формирование выходных (расшифрованных данных) T на основании массива состояния S по правилу.   **Рассмотрим подробно действия, выполняемые на каждом шагу дешифрования (кроме тех, которые уже были рассмотрены):**  **1.обратный сдвиг строк (InvShiftRows)**. Это преобразование приводит к тому, что последние три строки матрицы состояния S циклически сдвигаются на указанное (различное для каждой строки) число байт вправо. При этом количество байт сдвига вправо соответствует номеру строки (если строки нумеровать от 0). 1-ая по счету строка матрицы S остается без изменений.  **2.обратная замена байт (InvSubBytes)**. Это действие (**называется обратный S-box**) представляет нелинейную замену байтов, которая действует на каждый элемент массива состояний S и состоит из обратного к соотношению (46) аффинного преобразования, за которым следует получение мультипликативной инверсии в поле .  **3.обратное смешивание столбцов (InvMixColumns)**. Это действие является обратным к **смешиванию столбцов (MixColumns)**  и выполняется над каждым столбцом матрицы S, который интерпретируется как 4-ех элементный вектор с коэффициентами в поле . Каждый такой вектор умножается согласно формулам на вектор , который определяется соотношением. Таким образом, каждый столбец, который обозначается вектором s(x), преобразуется в столбец s’(x) по следующей формуле:.  Таким образом, коэффициенты вектора s’(x) вычисляются по следующим формулам:  ,  ,  .  Следует отметить, что структура алгоритма дешифрования несколько отличается от структуры алгоритма шифрования. Однако существует возможность использовать алгоритм дешифрования, структура которого полностью совпадает (за исключением применения обратных действий) со структурой алгоритма шифрования: |
| 19.Алгоритм шифрования AES. Подробное описание расширения ключа при шифровании и дешифровании.  **расширения исходного ключа K**. Это действие выполняется в начале шифрования, в результате чего формируется матрица w (положим нумерацию ее строк и столбцов от 0) следующим образом:   * 1. 1-ые  столбцов (каждый столбец представляет собой последовательные 4 байта) матрицы w содержат байты исходного ключа k;   2. остальные столбцы матрицы w формируются по следующей формуле:     причем вектор  (его размерность − 4 байта) вычисляется на каждой итерации по следующему алгоритму:  сначала вектор  инициализируется значениями вектора : , если выполняется условие , то  происходит циклический сдвиг влево содержимого вектора , так что полученное значение вектора имеет вид:;   1. полученный после предыдущих шагов вектор  подвергается побайтно преобразованию, задаваемому таблице 2. выполняется сложение по модулю 2 вектора , полученного после предыдущего шага, с вектором  (где j − номер итерации):   ,  причем вектор  формируется на каждой итерации следующим образом в поле :,  где z={02};  если же условие не выполняется, то происходит проверка условий:, .  В случае, когда оба эти условия выполняются, то вектор  подвергается побайтно преобразованию.  **расширение исходного ключа K для выполнения дешифрования**.   1. инициализация массива состояний S исходными зашифрованными данными C согласно правилу ; 2. начальное добавление ключа  к массиву состояний S (AddRoundKey); 3. раундов шифрования, начиная от -го и до 2-го. Каждый из  раундов может быть представлен следующим образом:    1. 4.1. обратная замена байт (InvSubBytes) для массива s;    2. 4.2. обратный сдвиг строк (InvShiftRows) для массива s;    3. 4.3. обратное смешивание столбцов (InvMixColumns) для массива s;    4. 4.4. добавление ключа  () к массиву состояний s (AddRoundKey). 4. последний раунд отличается от остальных тем, что не содержит шага 4.3 − обратного смешивания столбцов. При этом используется текущий ключ ; 5. формирование выходных (расшифрованных данных) T на основании массива состояния s по правилу.   Однако при использовании такого алгоритма дешифрования на 1-ом шаге необходимо несколько изменить алгоритм расширения исходного ключа **K**. Таким образом, измененный алгоритм будет иметь следующий вид:   1. сначала полностью выполняется обычное расширение исходного ключа K; 2. затем каждый столбец матрицы w, начиная с -го до -го, подвергается обратному смешиванию столбцов (InvMixColumns). | 20.Алгоритм шифрования AES. Способы повышения эффективности (быстродействия) при реализации процесса шифрования/дешифрования (описать подробно).   1. Ключи раунда генерируются 1 раз для всех блоков. 2. Для хранения 128-битного блока может быть использована структура, объединяющая 2 64-битных значения. 3. Для хранения массива состояний можно использовать union, содержащий указатели на массив 32-битных беззнаковых значений, на массив беззнаковых 1-байтных значений и на 2-умерный массив байтовых значений. Для повышения эффективности работы массив состояний лучше хранить в развернутом (транспонированном по отношению к теоретическому описанию) виде в смысле индексации элементов. 4. Аналогичным образом может быть описан тип данных для хранения ключа раунда. Опять же матрицу ключей раундов лучше хранить в транспонированном по отношению к теоретическому описанию виде. 5. Для хранения данных, описанных в табл применяется одномерный массив, значения в котором располагаются построчно. Само же преобразование заключается в обращении к элементу массива по заданному индексу (индексом является байт массива состояний). При этом при шифровании и дешифровании отличия заключаются только в содержании элементов массива. Таким образом, может быть использован один и тот же метод, но с разными параметрами. 6. Сдвиги строк при шифровании и дешифровании отличаются только обработкой 2-ой и 4-ой строк (если нумеровать строки, начиная от 1). Обработка 2-ой строки при шифровании происходит также, как 4-ой строки при дешифровании, и наоборот. При хранении массива состояний в транспонированном виде это единственная операция, реализация которой будет несколько усложнена. 7. Для ускорения выполнения смешивания столбцов необходимо иметь предварительно просчитанные значения произведений всех числе от 0 до 255 на 2 и 3, соответственно, записанные в виде одномерного массива. Это можно делать каждый раз при запуске программы или это могут быть предварительно рассчитанные массивы до этапа компиляции. Аналогично следует поступить при реализации обратного смешивания столбцов, но таких массива уже будет 4. 8. При вычислении формулы наиболее простой способ формирования 1-го элемента вектора − это запись предварительно рассчитанных значений в одномерный массив. Эти значения могут быть взяты из описания стандарта. 9. Для компенсации влияния инвертирования порядка байт при чтении 128-битного блока из файла (когда чтение происходит единым блоком, а не массивом байт) проще всего инвертировать порядок байт в ключе шифрования.    1. Также следует отметить, что в большинстве современных центральных процессоров (х86 / х64) существует поддержка набора инструкций AES\_NI, предназначенных для ускорения работы алгоритма AES. Этот набор включает 6 инструкций: 2 инструкции для генерации ключей раунда, и по 2 инструкции для шифрования/дешифрования (не последний/последний раунд). | 21.Алгоритм шифрования AES. Режимы работы алгоритма (с детальным описанием).  **Режимы работы алгоритма AES** полностью совпадают с аналогичными режимами работы алгоритма DES. Отличия в работе касаются только максимальной длины блока исходного текста T − 128 бит и длины начального вектора  − 128 бит. Также введен один дополнительный режим работы CTR. В этом режиме для получения зашифрованного блока C(i)происходит шифрование некоторого счетчика CTR(i) и на основе результата его шифрования формируется значение C(I) по формуле:.  Дешифрование происходит по формуле:.  Очевидно, что в данном режиме алгоритм AES используется ТОЛЬКО для шифрования (что более производительно). Также предполагается, что значения CTR(i) формируются параллельно для каждого блока исходных данных (способ формирования зависит от конкретной программной реализаций и условий применения), что позволяет использовать параллельную обработку.  Слабые или частично слабые ключи для алгоритма AES неизвестны. |
| 22.Системы шифрования файлов. Особенности программной реализации.  Для со­вре­мен­ных крип­то­гра­фи­че­ских сис­тем за­щи­ты ин­фор­ма­ции сфор­му­ли­ро­ва­ны сле­дую­щие об­ще­при­ня­тые тре­бо­ва­ния:   1. за­шиф­ро­ван­ное сообщение дол­жно под­да­вать­ся чте­нию толь­ко при на­ли­чии клю­ча; 2. чис­ло опе­ра­ций, не­об­хо­ди­мых для оп­ре­де­ле­ния ис­поль­зо­ван­но­го клю­ча шиф­ро­ва­ния по фраг­мен­ту шиф­ро­ван­но­го сообщения и со­от­вет­ст­вую­ще­го ему от­кры­то­го тек­ста, долж­но быть не мень­ше об­ще­го чис­ла воз­мож­ных клю­чей; 3. чис­ло опе­ра­ций, не­об­хо­ди­мых для рас­шиф­ро­вы­ва­ния ин­фор­ма­ции пу­тем пе­ре­бо­ра все­воз­мож­ных ключей долж­но иметь стро­гую ниж­нюю оцен­ку и вы­хо­дить за пре­де­лы воз­мож­но­стей со­вре­мен­ных ком­пь­ю­те­ров (с учетом возможности использования сетевых вычислений); 4. зна­ние ал­го­рит­ма шиф­ро­ва­ния не долж­но вли­ять на на­деж­ность за­щи­ты; 5. не­зна­чи­тель­ное из­ме­не­ние клю­ча долж­но при­во­дить к су­ще­ст­вен­но­му из­ме­не­нию ви­да за­шиф­ро­ван­но­го сообщения да­же при ис­поль­зо­ва­нии од­но­го и то­го же сообщения; 6. струк­тур­ные эле­мен­ты ал­го­рит­ма шиф­ро­ва­ния долж­ны быть не­из­мен­ны­ми; 7. до­пол­ни­тель­ные би­ты, вво­ди­мые в сообщение в про­цес­се шиф­ро­ва­ния, должен быть пол­но­стью и на­деж­но скры­ты в шиф­ро­ван­ном тек­сте; 8. дли­на шиф­ро­ван­но­го тек­ста долж­на быть рав­ной дли­не ис­ход­но­го тек­ста; 9. не долж­но быть про­стых и лег­ко ус­та­нав­ли­вае­мых зависимостей ме­ж­ду клю­ча­ми, по­сле­до­ва­тель­но ис­поль­зуе­мы­ми в про­цес­се шиф­ро­ва­ния; 10. лю­бой ключ из мно­же­ст­ва возможных дол­жен обес­пе­чи­вать на­деж­ную за­щи­ту ин­фор­ма­ции; 11. ал­го­ритм должен до­пус­кать как про­грамм­ную, так и ап­па­рат­ную реа­ли­за­цию, при этом из­ме­не­ние длины к­лю­ча не долж­но вес­ти к ка­че­ст­вен­но­му ухуд­ше­нию алгоритма шифрования. | 23.Алгоритмы шифрования с открытым ключом. Отличия от симметричных алгоритмов шифрования. Примеры алгоритмов шифрования с открытым ключом (с кратким описанием математической основы алгоритма).  в **симметричных криптосистемах** и для шифрования, и для дешифрования используется **один и тот же ключ** (секретный ключ), который должен быть известен только узкому кругу лиц;  в **системах с открытым ключом** используются два ключа − **открытый** и **закрытый**, которые математически связаны друг с другом. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения;  Идея системы RSA, как, собственно, и для всех алгоритмов шифрования с открытым ключом, заключается в том, что функция шифрования F обладает следующим свойствами:   1. существует достаточно быстрый алгоритм вычисления значения F; 2. существует достаточно быстрый алгоритм вычисления значения обратной функции ; 3. функция F обладает некоторым «секретом», знание которого позволяет быстро вычислять значения ; в противном же случае вычисление  становится трудно разрешимой в вычислительном отношении задачей, требующей для своего решения столь много времени, что по его прошествии зашифрованная информация перестает представлять интерес для лиц, использующих функцию F для шифрования. | 24.Алгоритм шифрования RSA. Математические основы. Достоинства и недостатки  в основу асимметричных алгоритмов закладываются вычислительно трудные задачи факторизации, дискретного логарифмирования, проецирования точек на эллиптической кривой и т.д. Объединяет все эти задачи то, что они используют операцию получения остатка от целочисленного деления. Для любого положительного целого числа *n* и любого *a* при делении *a* на *n* мы получаем некоторое целое частное *q* и остаток *r*, удовлетворяющий соотношению *a* = *qn* + *r*, 0 ≤ *r* < *n*; *q* = int(*a/n*), где int(x) обозначает наибольшее целое число, не превышающее x. Если a является целым, а *n* - положительным, то *a* mod *n* определяется как остаток от деления *a* на *n*. Таким образом, для любого целого числа *a* можно записать *a* = int(*a/n*) \* *n* + (*a* mod *n*).  Говорят, что два целых числа *a* и *b* являются сравнимыми по модулю *n*, если (*a* mod *n*) = (*b* mod *n*). Это записывается в виде: *a* ≡ *b* mod *n*. Операции сравнения по модулю имеют следующие свойства: 1. *a* ≡ *b* mod *n*, *n* | (*a* - *b*) (*n* | *x* означает, что *n* делит *x* нацело). 2. Из (*a* mod *n*) = (*b* mod *n*) следует *a* ≡ *b* mod *n*.  3. Из *a* ≡ *b* mod *n* следует *b* ≡ *a* mod *n*. 4. Из *a* ≡ *b* mod *n* и *b* ≡ *c* mod *n* следует *a* ≡ *c* mod *n*. Операции арифметики в классах вычетов обладают следующими свойствами: 1. [(*a* mod *n*) + (*b* mod *n*)] mod *n* = (*a* + *b*) mod *n*. 2. [(*a* mod *n*) - (*b* mod *n*)] mod *n* = (*a* - *b*) mod *n*. 3.[(*a* mod *n*) \* (*b* mod *n*)] mod *n* = (*a* \* *b*) mod *n*. Пусть Zn обозначает множество всех не отрицательных целых чисел, которые меньше *n*: Zn = {0, 1, 2, ..., (n - 1)}. Это множество называется ещё множеством вычетов (остатков) по модулю *n*.  Для поиска мультипликативного обратного можно использовать расширенный алгоритм Евклида, который позволяет в целых числах найти решение уравнения *ax*+*by* = 1 при заданных *а* и *b*. Очевидно, что если решение существует, то *x* будет величиной, мультипликативно обратной *а* по модулю *b*. Алгоритм Евклида Определить матрицу *E*: http://www.volpi.ru/umkd/zki/images/ImgFormula-2.5.1.gif Вычислить *r* – остаток от деления числа *a* на *b*: *a* = *bq* + *r*,0 ≤ *r* < *b* Если *r* = 0, топервый столбец матрицы *E* является решением уравнения. Если *r* ≠ 0, заменить матрицу *Е*: http://www.volpi.ru/umkd/zki/images/ImgFormula-2.5.2.gif Поменять местами столбцы матрицы *Е*. Заменить пару чисел *a*, *b* на *b*, *r* и перейти к шагу 2. |
| 24.Алгоритм шифрования RSA. Подробное описание процесса шифрования / дешифрования.  Шифрование   1. Выбираются два различных простых числа p и q, **которые хранятся в тайне**, и вычисляется их произведениеN=p-q,   которое становится частью открытого и секретного ключей.   1. Шифруемый текст представляется в виде блоков сообщений (максимально возможное число разных блоков N). Для этого нужно выполнить следующие действия:    * + 1. сформировать список классов по модулю N, т.е. осуществить представление каждого символа блока в виде числа. Если сообщение содержит все возможные символы таблицы ASCII, то число − это код символа. Если же сообщение содержит только буквы, например, заглавные русские и пробел, то можно использовать следующую перекодировку: буквы от А до Я представляются числами от 10 до 41, а пробел − числом 99;        2. сформировать блоки таким образом, чтобы натуральное число, представляющее каждый блок, не превышало N. Следует отметить, что выбор блоков неоднозначен, но и не совсем произволен. Например, не следует выделять блоки, которые начинаются с 0. Натуральное число образуется путем последовательной записи чисел, представляющих каждый символ текущего блока слева направо. В качестве одного из способов разделения текста на блоки можно использовать следующий алгоритм (при использовании десятичной системы счисления): 2. вычислить число ASCII символов , необходимое для представления числа N по формуле:,   где  − количество десятичных цифр числа N;   1. каждый блок исходного текста должен содержать  ASCII символов, причем  определяется соотношением:.   Следует отметить, что при использовании вышеприведенного алгоритма разбиения на группы предпочтительнее применять шестнадцатеричную систему счисления. В этом случае число ASCII символов , необходимых для представления числа N, вычисляется по формуле:,  причем  − количество шестнадцатеричных цифр числа N. Значение  при этом вычисляется по формуле с использованием .  Еще одной проблемой при таком разбиении на группы является присутствие в начале группы символа с кодом 0. Если возникает такая ситуация, то одним из путей решения проблемы является уменьшения размера предыдущей группы на 1 символ (или более, если этот символ тоже 0), с которого будет начинаться новая группа. Однако применение такого подхода может быть существенно усложнено наличием длинных последовательностей символов с кодом 0. Другим простым решением данной проблемы является выделение каждого такого символа в отдельную группу. **Однако следует отметить тот факт, что число 0 не может быть зашифровано алгоритмом RSA.** Следовательно, это приведет к снижению криптостойкости.  Для борьбы с последовательностями символов с кодом 0 может быть применен какой-либо алгоритм сжатия. Простейшим вариантом является использование алгоритма RLE (Run-Length Encoding), суть которого заключается в представлении каждой последовательности одинаковых символов в виде двух символов: **количество повторяющихся символов**, **символ**.  Последовательности неповторяющихся символов представляются в виде:  **символ с кодом 0, количество неповторяющихся символов, последовательность символов**.  **Таким образом, максимально возможная длина последовательности символов с кодом 0 равна 2.**   1. Выбирается натуральное число E, обратимое по модулю  . Это число должно удовлетворять условиям: , ,   где НОД − наибольший общий делитель, а , т.к. p и q− простые, определяется соотношением:,  Следует отметить, что с точки зрения повышения производительности значение  лучше вычислять по формуле:.  Чтобы условие удовлетворялось, число E можно выбирать простым. Если же выбранное число E не является простым (соответствующий тест не был проведен), то для проверки удовлетворения условия используется алгоритм Евклида (для нахождения НОД  **Пара (N,E) называется открытым (публичным) или кодирующим ключом RSA.** Следует отметить, что на практике в качестве числа E часто выбирают простые числа Ферма, т.е. простые числа вида:.   1. Каждый блок исходной информации представляется в виде некоторого числа . 2. Происходит вычисление числа  по формуле:   ,  причем число  является текущим блоком зашифрованного текста. Для вычисления значения  используется алгоритм быстрого возведения в степень, суть которого заключается в следующем (одна из возможных реализаций):   * 1. число E представляется в двоичной системе счисления следующим образом (следует учитывать, что E всегда больше 0):,   где  − очередная цифра двоичного представления числа E и может принимать только значения 0 и 1. Если число хранится в стандартном числовом формате, то никаких специальных действий для такого представления выполнять не нужно;   * 1. устанавливается значение переменной ;   2. пока младший бит числа E равен 0, выполняются следующие действия:  1. вычисляется новое значение переменной :   , производится сдвиг числа E на один двоичный разряд вправо;   * 1. текущее значение результата ;   2. производится сдвиг числа E на один двоичный разряд вправо: E=E>>1;   3. если выполняется условие E=0, то производится выход, причем значение  и есть результат;   4. вычисляется новое значение переменной  по формуле   5. если младший бит числа E не равен 0, то вычисляется текущее значение результата по формуле:   . переход к пункту 5.5.  Из приведенного алгоритма видно, что чем меньше единиц в двоичном представлении числа E, тем быстрее будет получено итоговое значение . Поэтому и представляет интерес выбор простых чисел Ферма в качестве E.  Приведенный алгоритм не всегда оптимален, однако он требует ~ на 30% меньше времени (в худшем случае) по сравнению с простым перемножением.  **Дешифрование в системе RSA**   1. Дешифруемый текст представляется в виде блоков сообщений (максимально возможное число блоков N). Для этого нужно выполнить следующие действия:    * сформировать список классов по модулю N, что выполняется по тому же алгоритму, как и при шифровании;    * выделить блоки зашифрованного текста (алгоритмы выделения при дешифровании и шифровании должны соответствовать друг другу). Если выделение блоков при шифровании производилось согласно предложенному выше алгоритму, то при дешифровании выделение блоков происходит следующим образом:      1. вычисляется значение  по формуле или  по формуле;      2. каждый блок представляет собой  или  символов, причем отбрасываются все незначащие символы с кодом 0, расположенные слева, после чего оставшиеся интерпретируются в виде числа, образованного их ASCII кодами. 2. На основе известных значений p и q вычисляется значение 3. Вычисляется такое число D, которое удовлетворяет условию:,   **Пара (N,D) называется секретным или декодирующим ключом RSA.** Если при расчетах в расширенном алгоритме Евклида используются только неотрицательные числа, то можно знак числа хранить отдельно, имитируя таким образом работу со знаковыми числами.   1. Каждый блок исходной информации представляется в виде некоторого числа . 2. Происходит вычисление числа  по формуле:.   При вычислении числа  также применяется алгоритм быстрого возведения в степень;   1. Если при шифровании производилось сжатие последовательности символов, то на последнем шаге выполняется обратная процедура, в результате чего и получаем исходный текст. | 26.Алгоритм шифрования RSA. Особенности программной реализации (представление больших чисел и краткое описание алгоритмов, необходимых для реализации 4-ех основных арифметических действий с большими числами).  При необходимости оперировать большими числами без потери точности возникает необходимость самостоятельного описания как структур данных, хранящих цифры числа, так и подпрограмм, реализующих арифметические действия над числами в выбранном формате.  Прежде всего, нужно определиться с представлением больших чисел. Для описания большого числа на языке С++ будем использовать класс. Простейшим способом представления больших чисел является массив байт, каждый элемент которого хранит цифру числа (желательно в шестнадцатеричной системе, т.к. это позволяет легко реализовывать побитовые операции). Кроме того, хранятся максимально возможное и занятое число элементов. Такой способ позволяет также выполнять умножение двух больших чисел, без использования умножений и делений. Однако с точки зрения использования памяти данный способ крайне неэффективен, т.к. для хранения числа достаточно в 2 раза меньше байт (при использовании шестнадцатеричной системы счисления), т.к. 2 цифры занимают 1 байт.  При более рациональном использовании памяти − каждый байт хранит 2 шестнадцатеричных цифры − несколько замедляется операция умножения, но в при этом немного ускоряется сложения за счет автоматического переноса в следующий разряд.  Простейшие математические операции над числами включают сложение, вычитание умножение и деление (получение частного и остатка от деления).  Выполнение сложения производится также как и вручную на бумаге. При этом в зависимости от выбранного представления числа флаг переноса в следующий разряд формируется автоматически или вручную.  При выполнении вычитания следует учитывать, что можно хранить только неотрицательные числа (с целью повышения производительности). Следовательно, из числа большей длины нужно вычитать число меньшей длины. При равной длине чисел можно либо определить большее путем побайтового сравнения, либо вычитать «как есть», а потом, при необходимости, выполнить инверсию результата (если он окажется отрицательным).  Умножение выполняется аналогично тому, как оно делается на бумаге (хотя и существуют алгоритмы «быстрого» умножения, но их практическая реализация достаточно сложна и они не всегда окажутся эффективнее). При этом с целью повышения производительности можно использовать заранее рассчитанную таблицу результатов умножения каждой пары цифр. Также можно адекватно выполнять умножение на цифру 0.  Для выполнения деления может быть использован следующий достаточно простой в реализации алгоритм (требует возможности выполнения побитовых операций для больших чисел, т.к. выполняется деление в двоичной системе счисления), который определяет значения q и rem в выражении: n=m\*q+rem.  Этот алгоритм является расширенной версией алгоритма для определения остатка от деления, который был рассмотрен выше:   1. определяется значение  = количеству бит числа n; 2. устанавливаются начальные значения частного q=0 и остатка от деления rem=0; 3. ; 4. пока выполняется условие  (в противном случае значение rem содержит искомый остаток, а q − частное от деления, и происходит выход из цикла); 5. вычисляется текущее значение переменной rem по формуле 6. q=q<<1; 7. если выполняется условие rem>m, то    1. rem=rem-m;    2. младший бит числа q устанавливается в 1; 8. i=i-1; 9. переход к пункту 4.   С учетом особенностей системы RSA рассмотренный выше алгоритм не рассматривает возможность деления на 0. | 27.Алгоритм шифрования RSA. Особенности программной реализации (представление больших чисел и краткое описание алгоритмов, необходимых для реализации математических действий с большими числами, затребованных криптосистемой RSA).  При необходимости оперировать большими числами без потери точности возникает необходимость самостоятельного описания как структур данных, хранящих цифры числа, так и подпрограмм, реализующих арифметические действия над числами в выбранном формате.  При работе с большими числами использование метода проб для проверки правдивости гипотезы о том, что число − простое, является крайне неэффективным.  Для этих целей обычно применяются вероятностные методы (результат, возвращаемый ими, верен с определенной вероятностью), примером которых является **алгоритм Миллера-Рабина**  Суть алгоритма Миллера-Рабина заключается в следующем:   * 1. представляем число m-1 в виде, показанном в выражении (25). Для этих целей может быть использован последовательный сдвиг числа m-1 вправо до тех пор, пока младший бит не станет равным 1. При этом количество сдвигов будет соответствовать s, а результат после сдвигов − t;   2. установим значение переменной i=0;   3. выбираем целое число , являющееся основанием, по которому будет производиться тест Миллера. Этот выбор можно сделать случайным образом, однако более эффективно выбирать в качестве основания простые числа, начиная от 2 (список этих чисел для повышения производительности должен храниться в виде массива);   4. вычисляется значение p по формуле;   5. если выполняются условия:, ,   то производятся следующие действия:   * 1. установить значение k=0;   2. p=p\*p(modm);   3. если выполняется условие p=1, то выход, т.к. m − составное;   4. если выполняется условие p=m-1, то переход к пункту 6;   5. k=k+1;   6. если выполняется условие k<s-1, то переход к пункту 5.2;   7. выход, т.к. m − составное;   8. i=i+1;   9. если выполняется условие i<r, то переход к пункту 3;   10. m − вероятно простое.   Для проверки гипотезы о том, что число b является простым, может быть также использована теорема Ферма, из которой следует, что, если число b − составное, то выполняется условие:,  где значения a выбираются аналогично тесту Миллера-Рабина. Легко увидеть, что тест Миллера-Рабина основан на теореме Ферма, но его использование (с учетом работы с большими числами) более эффективно. |
| 28.Алгоритм шифрования RSA. Оценка надежности. Потенциальные уязвимости и способы взлома. Способы противодействия взлому.  Т.к. публичный ключ  содержит часть секретного ключа , то для удачного взлома нужно определить число D. Для этого необходимо знать значение , чтобы можно было применить расширенный алгоритм Евклида для вычисления значения D. Для получения значения  достаточно знать числа p и q, которые можно определить разложив известное значение N на множители. Поэтому для успешного противостояния атакам необходимо, чтобы число N было очень большим, что сделает процесс его разложения на множители практически невыполнимым.  Для обеспечения адекватного уровня криптостойкости размерность ключа должна быть как минимум 1024 бит, хотя надежной система RSA при этом считаться не может. Ключ такой длины может быть взломан, хотя для этого требуется привлечения существенных вычислительных ресурсов, большого объема памяти и некоторых временных ресурсов. Хотя с использованием сети биткоин-майнеров 1024-битный ключ может быть взломан где-то за полдня. На данный момент система RSA может считаться надежной при длине ключа не менее 2048 бит, а для особо важных задач рекомендуется использовать ключи большей длины, например, 4096 бит. **Под длиной ключа подразумевается длина числа N**.  С течением времени из-за совершенствования аппаратного обеспечения длина ключа может быть увеличена. При этом одной и той же парой открытого и секретного ключей рекомендуется пользоваться не более 1 года. | 29.Функции хеширования. Характеристики и области их применения. Примеры хеш-функций (назвать).  Хеш – функции некое математическое преобразование, которое для некой последовательности байт получает фиксированное по длине числовое значение. (При разных исходных данных разные значения).  Основные свойства хэш-функции: 1.на вход хэш-функции подается сообщение произвольной длины; 2. на выходе хэш-функции формируется блок данных фиксированной длины; 3.значения на выходе хэш-функции распределены по равномерному закону; 4. при изменении одного бита на входе хэш-функции существенно изменяется выход.  Кроме того, для обеспечения устойчивости хэш-функции к атакам она должна удовлетворять следующим требованиям: 1. если мы знаем значение хэш-функции *h*, то задача нахождения сообщения M такого, что Н(М) = *h*, должна быть вычислительно трудной; 2. при заданном сообщении M задача нахождения другого сообщения M’, такого, что Н(М) = H(M’), должна быть вычислительно трудной.  Если хэш-функция будет удовлетворять перечисленным свойствам, то формируемое ею значение будет уникально идентифицировать сообщения, и всякая попытка изменения сообщения при передаче будет обнаружена путем выполнения хэширования на принимающей стороне и сравнением с дайджестом, полученным на передающей стороне. Еще одной особенностью хэш-функций является то, что они не допускают обратного преобразования – получить исходное сообщения по его дайджесту невозможно. Поэтому их называют еще односторонними функциями шифрования. Хеш-функции широко используются в криптографии. Хеш используется как ключ во многих структурах данных — [хеш-таблицаx](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), [фильтрах Блума](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D1%82%D1%80_%D0%91%D0%BB%D1%83%D0%BC%D0%B0) и [декартовых деревьях](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D0%B2%D0%BE_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE). | 30.Электронная цифровая подпись (ЭЦП). Описание данного механизма и области применения. Способы реализации ЭЦП. Стандарты ЭЦП (назвать).  ЭЦП — это реквизит [электронного документа](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82), полученный в результате криптографического преобразования [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) с использованием [закрытого ключа](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи (целостность), принадлежность подписи владельцу [сертификата ключа подписи](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D1%8E%D1%87%D0%B0) (авторство), а в случае успешной проверки подтвердить факт подписания электронного документа (неотказуемость).  В настоящее время ЭЦП используется в основном  -      для аутентификации автора (создателя) информации и  -      для доказательства (проверки) того факта, что подписанное сообщение или данные не были модифицированы при передаче информации в компьютерных сетях. Электронная цифровая подпись строится на основе двух компонент:  1)     содержания информации, которая подписывается,  2)     личной информации (код, пароль, ключ) того, кто подписывает.  Очевидно, что изменение каждой компоненты приводит к изменению электронной цифровой подписи.   Битовая строка, присоединяемая к документу после подписания, называется **электронной цифровой подписью**.  Протокол, с помощью которого получатель убеждается в подлинности отправителя и целостности сообщения, называется *проверкой подлинности (аутентификацией).*  Первые варианты цифровой подписи были реализованы с помощью симметричных криптосистем. В качестве алгоритма криптографического преобразования использовалась любая симметричная криптосистема, обладающая специальными режимами функционирования.  Современные процедуры создания и проверки электронной цифровой подписи основаны на шифровании с открытым ключом.   Национальные стандарты  -         **Digital Signature Standard** (принят NIST в 1991 г. с последующими изменениями в 1993, 1996 г.),  -         российский стандарт цифровой подписи **ГОСТ Р 34.10-94**,  -         стандарт Республики Беларусь **СТБ 1176.2**, введенный в 1999 г.  реализовывают ЭЦП в простом поле. |