МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных технологий и систем безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(дипломная работа)

На тему «Разработка мессенджера со встроенным симметричным шифрованием для безопасной передачи сообщений в сети»

Исполнитель Назыров Вадим Арсенович
(фамилия, имя, отчество)

Руководитель кандидат технических наук
(ученая степень, ученое звание)

Хейстонен Дмитрий Павлович
(фамилия, имя, отчество)

«К защите допускаю»

Заведующий кафедрой _____
(подпись)

доктор технических наук, профессор
(ученая степень, ученое звание)

Бурлов Вячеслав Георгиевич
(фамилия, имя, отчество)

«17» февраля 2017 г.

Санкт–Петербург
2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных технологий и систем безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(дипломная работа)

На тему «Разработка мессенджера со встроенным симметричным шифрованием для безопасной передачи сообщений в сети»

Исполнитель Назыров Вадим Арсенович
(фамилия, имя, отчество)

Руководитель кандидат технических наук
(ученая степень, ученое звание)

Хейстонен Дмитрий Павлович
(фамилия, имя, отчество)

«К защите допускаю»

Заведующий кафедрой _____
(подпись)

доктор технических наук, профессор
(ученая степень, ученое звание)

Бурлов Вячеслав Георгиевич
(фамилия, имя, отчество)

«__ »_____20 г.

Санкт–Петербург
2017

Содержание

Введение

Мы живём в эпоху одного из самых значительных скачков в развитии человечества. Наше общество почти уже достигло той стадии, когда его уже с полной уверенностью можно назвать информационным. Большую роль в развитии коммуникаций играет появление Интернета.

Интернет – это глобальная сеть, насчитывающая тысячи компьютеров, разбросанных по всему миру. Когда два компьютера взаимодействуют между собой, весь трафик от источника до пункта назначения проходит через множество других устройств.

В современном мире люди активно пользуются сетью Интернет в целях получения информации, общения и передачи каких-либо файлов друг другу. Это очень эффективный и доступный метод, однако, существует большая вероятность перехвата личных сообщений и других данных в сети третьими лицами. При этом участники сеанса связи могут даже не подозревать об этом.

Вопросами защиты информации занимаются специалисты по информационной безопасности. «Информационная безопасность представляет собой состояние сохранности информационных ресурсов и защищенности законных прав личности и общества в информационной сфере.»[1]

Любой пользователь Интернета заинтересован в конфиденциальности передаваемых и получаемых им сообщений. Конфиденциальная информация, которая передается по сети Интернет, проходит через определенное количество маршрутизаторов и серверов, прежде чем достигнет пункта назначения. Обычно маршрутизаторы не отслеживают проходящие сквозь них потоки информации, но возможность того, что информация может быть перехвачена, существует.

Более того, информация может быть изменена и передана адресату в измененном виде. К сожалению, сама архитектура сети Интернет всегда оставляет возможность для недобросовестного пользователя осуществить подобные действия.

Проблемы, возникающие с безопасностью передачи информации при работе в компьютерных сетях, можно разделить на четыре основных типа:

- перехват информации – целостность информации сохраняется, но ее конфиденциальность нарушена;
- модификация информации – исходное сообщение изменяется либо полностью подменяется другим и отсылается адресату;
- подмена авторства информации;
- перехват сообщения с его изъятием.

В соответствии с перечисленными проблемами при обсуждении вопросов безопасности можно выделить три основных признака, характеризующих систему, содержащую информацию, как безопасную:

«Конфиденциальность – состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на неё право;» [2]

«Целостность – избежание несанкционированной модификации информации;» [2]

«Доступность – избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.»[2]

Обеспечение безопасности информационной системы предусматривает одновременное выполнение трех процедур: идентификация, аутентификация и авторизация.

«Идентификация – присвоение пользователю (объекту или субъекту ресурсов) уникальных имен и кодов – идентификаторов.»[2]

«Аутентификация – установление подлинности пользователя, представившего идентификатор или проверка того, что лицо или устройство, сообщившее идентификатор является действительно тем, за кого оно себя выдает. Наиболее распространенным способом аутентификации является присвоение пользователю пароля и хранение его в компьютере.»[2]

«Авторизация – проверка полномочий или проверка права пользователя на доступ к конкретным ресурсам и выполнение определенных операций над ними. Авторизация проводится с целью разграничения прав доступа к сетевым и компьютерным ресурсам.»[2]

В некоторых случаях пользователями или потребителями меры по обеспечению безопасности могут быть расценены как меры по ограничению доступа и эффективности. Однако такие средства, как, например, криптография, позволяют значительно усилить степень защиты, не ограничивая доступ пользователей к данным.

Криптографическое преобразование – это преобразование информации, основанное на некотором алгоритме, зависящем от изменяемого параметра (криптографического ключа), и обладающее свойством невозможности восстановления исходной информации по преобразованной, без знания действующего ключа, с трудоемкостью меньше заранее заданной.[3]

Основным достоинством криптографических методов является то, что они обеспечивают высокую степень защиты, которую можно рассчитать и выразить в числовой форме (средним числом операций или временем, необходимым для раскрытия зашифрованной информации или вычисления ключей).

Проблема шифрования и дешифрования информации, а в особенности текстовой информации, всегда была и до сих пор остается очень актуальной. В связи с развитием компьютерных технологий и применением электронно-вычислительных машин в этой области, криптография шагнула в новую эпоху своего развития и находится на том уровне, когда зашифрованные сообщения невозможно расшифровать подручными средствами. Зачастую для этого требуются огромные вычислительные и временные ресурсы, которые могут позволить себе далеко не все.

В данной выпускной квалификационной работе описываются методы построения современных телекоммуникационных систем, их уязвимости и методы борьбы с ними.

Целью проекта является создание защищенного соединения между двумя удаленными компьютерами. Защита должна быть обеспечена таким образом, чтобы злоумышленник, в случае перехвата сообщения, не смог понять его содержимое и узнать информацию об источнике или адресате.

Для достижения этой цели были поставлены следующие задачи:

- Исследование основных принципов построения ТКС, существующих угроз и методов защиты от них;
- Разработка архитектуры защищенного канала связи;
- Разработка программного обеспечения для обмена сообщениями в сети – мессенджера;
- Разработка и внедрение симметричного шифрования в программу для защиты сообщений от несанкционированного доступа.

ГЛАВА I Теоретические основы построения защищённых телекоммуникационных вычислительных сетей

1.1 Описание телекоммуникационных вычислительных сетей.

«Телекоммуникационная вычислительная сеть – это вычислительный комплекс, включающий территориально распределенную систему компьютеров и их терминалов, объединенных в единую систему.»[4]

Вычислительные сети создаются для того, чтобы группа пользователей могла совместно задействовать одни и те же ресурсы: файлы, принтеры, модемы, процессоры и т.п. Каждый компьютер в сети оснащается сетевым адаптером, адаптеры соединяются с помощью сетевых кабелей и тем самым связывают компьютеры в единую сеть. Компьютер, подключенный к вычислительной сети, называется рабочей станцией или сервером, в зависимости от выполняемых им функций.

Пользователи вычислительной сети получают возможность совместно использовать ее программные, технические, информационные и организационные ресурсы. В вычислительной сети выделяют совокупность узлов и соединяющих их ветвей.

По степени географического распространения вычислительные сети подразделяются на локальные, городские, корпоративные, глобальные и другие.

«Локальная вычислительная сеть (LAN – Local Area Network) — это совокупность компьютеров и других средств вычислительной техники, объединенных с помощью кабелей и сетевых адаптеров и работающих под управлением сетевой операционной системы. Локальная вычислительная сеть объединяет абонентов, находящихся на небольшом расстоянии друг от друга в пределах 10-15 км.»[5]

Информационные системы, построенные на базе локальных вычислительных сетей, обеспечивают решение следующих задач:

- хранение данных;
- обработка данных;
- организация доступа пользователей к данным;
- передача данных и результатов их обработки пользователям.

Наиболее распространенными областями применения локальных сетей являются:

- обмен информацией между абонентами сети;
- реализация электронного документооборота на предприятиях и в организациях;
- обеспечение распределенной обработки информации, организация информационных систем, содержащих общие или индивидуальные, локальные или распределенные базы данных;
- поддержка принятия управленческих решений на основе оперативной информации о производственно-хозяйственной, коммерческой и торговой деятельности подразделений.

«Городская сеть (MAN – Metropolitan Area Networks) — это сеть большего размера, чем ЛС. Она обычно покрывает область одного города (от нескольких десятков до сотни километров). В городских сетях часто используется различное оборудование и среды передачи с целью эффективной их работы на таких расстояниях.»[6]

«Корпоративные сети (масштаба предприятия) — совокупность связанных между собой ЛВС, охватывающих территорию, на которой размещено одно предприятие или учреждение в одном или нескольких близко расположенных зданиях.»[6]

«Глобальная сеть (WAN – Wide Area Network) — компьютерная сеть, охватывающая большие территории и включающая большое число узлов. Глобальные вычислительные сети связывают компьютеры, рассредоточенные на

расстоянии сотен и тысяч километров. Примером одной из глобальных сетей является Интернет. Глобальные сети дают возможность пользоваться обменом неограниченному числу абонентов, но при этом каналы связи сети имеют сравнительно низкую скорость передачи и не слишком качественную связь, чем и отличаются от локальных сетей.»[6]

В глобальных сетях для передачи информации применяются следующие виды коммутации:

- коммутация каналов (используется при передаче аудиоинформации по обычным телефонным линиям связи);
- коммутация сообщений применяется в основном для передачи электронной почты, в телеконференциях, электронных новостях);
- коммутация пакетов (для передачи данных, в последнее время используется также для передачи аудио - и видеоинформации).

1.2 Принципы построения ТКС

При построении локальных компьютерных сетей необходимо учитывать множество различных факторов, например, количество объединяемых в сеть компьютеров, удаленность их друг от друга, обеспечение конфиденциальности передаваемых по сети данных и т.д.

В состав типичной коммуникационной системы входят серверы, пользовательские компьютеры, каналы связи, а также активное оборудование – модемы, концентраторы и прочее.

Основные компоненты телекоммуникационной системы:

1. Серверы, хранящие и обрабатывающие информацию.
2. Рабочие станции и пользовательские ПК, служащие для ввода запросов к базам данных, получения и обработки результатов запросов и выполнения других задач конечных пользователей информационных систем.
3. Коммуникационные каналы – линии связи, по которым данные передаются между отправителем и получателем информации. Коммуникационные

каналы используют различные типы среды передачи данных: телефонные линии, волоконно-оптический кабель, коаксиальный кабель, беспроводные и другие каналы связи.

4. Сетевое оборудование – модемы, сетевые адаптеры, концентраторы, коммутаторы, маршрутизаторы и прочее. Эти устройства необходимы для передачи и приема данных.

5. Сетевое программное обеспечение, управляющее процессом передачи и приема данных и контролирующее работу отдельных частей коммуникационной системы.

1.2.1 Сетевая модель OSI

Для единого представления данных в сетях с неоднородными устройствами и программным обеспечением была разработана базовая модель взаимодействия открытых систем OSI (Open System Interconnection). Эта модель описывает правила и процедуры передачи данных в различных сетевых средах при организации сеанса связи. Основными элементами модели являются уровни, прикладные процессы и физические средства соединения.

Каждый уровень модели OSI выполняет определенную задачу в процессе передачи данных по сети. Базовая модель является основой для разработки сетевых протоколов. OSI разделяет коммуникационные функции в сети на семь уровней, каждый из которых обслуживает различные части процесса области взаимодействия открытых систем.

Модель OSI описывает только системные средства взаимодействия, не касаясь приложений конечных пользователей. Приложения реализуют свои собственные протоколы взаимодействия, обращаясь к системным средствам.[7]

Сетевая модель OSI состоит из 7 уровней (отсчёт принято начинать с нижнего):

7. Прикладной уровень (application layer)
6. Уровень представления (presentation layer)
5. Сеансовый уровень (session layer)
4. Транспортный уровень (transport layer)
3. Сетевой уровень (network layer)
2. Канальный уровень (data link layer)
1. Физический уровень (physical layer)

Появление именно такой структуры было обусловлено следующими соображениями:

1. Уровень должен создаваться по мере необходимости отдельного уровня абстракции.
2. Каждый уровень должен выполнять строго определенную функцию.
3. Выбор функций для каждого уровня должен осуществляться с учетом создания стандартизированных международных протоколов.
4. Границы между уровнями должны выбираться так, чтобы поток данных между интерфейсами был минимальным.
5. Количество уровней должно быть достаточно большим, чтобы различные функции не объединялись в одном уровне без необходимости, но не слишком высоким, чтобы архитектура не становилась громоздкой.

Прикладной уровень.

Прикладной уровень содержит набор популярных протоколов, необходимых пользователям. Одним из наиболее распространенных является протокол передачи гипертекста HTTP (HyperText Transfer Protocol), который составляет основу технологии Всемирной Паутины. Когда браузер запрашивает веб-страницу, он передает ее имя (адрес) и рассчитывает на то, что сервер будет использовать HTTP. Сервер в ответ отсылает страницу. Другие прикладные

протоколы используются для передачи файлов, электронной почты, сетевых рассылок.

Уровень представления.

Задача уровня представления данных состоит в том, чтобы информация уровня приложений, которую посылает одна система (отправитель), могла быть прочитана уровнем приложений другой системы (получателя). При необходимости уровень представления преобразует данные в один из многочисленных существующих форматов, который поддерживается обеими системами.

На этом уровне может осуществляться сжатие/распаковка или кодирование/декодирование, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.

Другой важной задачей этого уровня является шифрование и расшифровка данных. Типовыми графическими стандартами шестого уровня являются стандарты PICT, TIFF и JPEG. Примерами стандартов шестого уровня эталонной модели, описывающих формат представления звука и видео, являются стандарты MIDI и MPEG.

Сеансовый уровень.

Сеансовый уровень позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом. При этом предоставляются различные типы сервисов, среди которых управление диалогом (отслеживание очередности передачи данных), управление маркерами (предотвращение одновременного выполнения критичной операции несколькими системами) и синхронизация (установка служебных меток внутри длинных сообщений, позволяющих после устранения ошибки продолжить передачу с того места, на котором она оборвалась).

Примерами протоколов пятого уровня могут служить сетевая файловая система (Network File System — NFS), система X-Window и протокол сеанса AppleTalk (AppleTalk Session Protocol — ASP).

Транспортный уровень.

Транспортный уровень сегментирует данные передающей станции и вновь собирает их в одно целое на принимающей стороне. Границу между транспортным уровнем и уровнем сеанса связи можно рассматривать как границу между протоколами приложений и протоколами передачи данных.

В то время как уровни приложений, представления данных и сеанса связи занимаются аспектами коммуникаций, которые связаны с работой приложений, нижние четыре уровня решают вопросы транспортировки данных по сети. Транспортный уровень старается обеспечить службу передачи данных таким образом, чтобы скрыть от верхних уровней детали процесса передачи данных. В частности, задачей транспортного уровня является обеспечение надежности передачи данных между двумя рабочими станциями.

При обеспечении службы связи транспортный уровень устанавливает, поддерживает и соответствующим образом ликвидирует виртуальные каналы. Для обеспечения надежности транспортной службы используются выявление ошибок при передаче и управление информационными потоками. Примерами протоколов четвертого уровня могут служить протокол управления передачей (Transmission Control Protocol — TCP), протокол пользовательских дейтаграмм (User Datagram Protocol — UDP) и протокол последовательного обмена пакетами (Sequenced Packet Exchange — SPX).

Сетевой уровень.

Сетевой уровень является комплексным уровнем, обеспечивающим выбор маршрута и соединение между собой двух рабочих станций, которые могут быть расположены в географически удаленных друг от друга сетях. Кроме того, сетевой уровень решает вопросы логической адресации.

Если в подсети одновременно присутствует слишком большое количество пакетов, то они могут закрыть дорогу друг другу, образуя заторы в узких местах. Недопущение подобной закупорки также является задачей сетевого уровня. В более широком смысле сетевой уровень занимается предоставлением определенного уровня обслуживания (задержки, время передачи, вопросы синхронизации).

Примерами протоколов третьего уровня могут служить Internet протокол (IP), протокол межсетевого пакетного обмена (Internetwork Packet Exchange — IPX) и протокол AppleTalk.

Канальный уровень.

Канальный уровень обеспечивает надежную передачу данных по физическому каналу. Основная задача данного уровня — организация передачи «сырых» данных физического уровня по надежной линии связи, свободной от необнаруженных ошибок с точки зрения вышестоящего сетевого уровня. Уровень выполняет эту задачу при помощи разбиения входных данных на кадры, обычный размер которых колеблется от нескольких сотен до нескольких тысяч байт. Кадры данных передаются последовательно с обработкой кадров подтверждения, отсылаемых обратно получателем.

Спецификация IEEE 802 разделяет канальный уровень на 2 подуровня. MAC (Media Access Control) регулирует доступ к разделяемой физической среде, LLC (Logical Link Control) обеспечивает обслуживание сетевого уровня.

Примерами протоколов, работающих на канальном уровне, являются: Ethernet для локальных сетей (многоузловой), Point-to-Point Protocol (PPP), HDLC и ADCCP для подключений точка-точка (двухузловой).

Физический уровень.

Физический уровень занимается реальной передачей необработанных битов по каналу связи. При разработке сети необходимо убедиться, что когда одна сторона передает единицу, то принимающая сторона получает также единицу, а не ноль. Принципиальными вопросами здесь являются следующие: какое напряжение должно использоваться для отображения единицы, а какое — для нуля; сколько микросекунд длится бит; может ли передача производиться одновременно в двух направлениях; как устанавливается начальная связь и как она прекращается, когда обе стороны закончили свои задачи; из какого количества проводов должен состоять кабель и какова функция каждого провода. Вопросы разработки в основном связаны с механическими, электрическими и процедурными интерфейсами, а также с физическим носителем, лежащим ниже физического уровня.[8],[9]

1.2.2 Архитектура сети

Для организации любой сети необходимо обозначить архитектуру взаимодействия компонентов этой сети. Существует две основных архитектуры сетей – одноранговая («peer-to-peer», «P2P») и иерархическая («клиент-сервер»).

В одноранговой сети все компьютеры равноправны — имеют один ранг. Поэтому любой компьютер может выступать как в роли сервера, то есть предоставлять свои ресурсы (файлы, принтеры) другому компьютеру, так и в роли клиента — использовать предоставленные ему ресурсы.

Одноранговые сети являются наиболее простым для монтажа и настройки, а также дешевым типом сетей. Для построения одноранговой сети требуется всего лишь несколько компьютеров с установленными клиентскими ОС, и снабженных

сетевыми картами. Все параметры безопасности определяются исключительно настройками каждого из компьютеров.

К основным достоинствам одноранговых сетей можно отнести:

- простоту работы в них;
- низкую стоимость, поскольку все компьютеры являются рабочими станциями;
- относительную простоту администрирования.

Недостатки одноранговой архитектуры:

- эффективность работы зависит от количества компьютеров в сети;
- защита информации и безопасность зависит от настроек каждого компьютера.

Серьезной проблемой одноранговой сетевой архитектуры является ситуация, когда компьютеры отключаются от сети. В этих случаях из сети исчезают все общесетевые сервисы, которые они предоставляли (например, общая папка на диске отключенного компьютера, или общий принтер, подключенный к нему).

Администрировать такую сеть достаточно просто лишь при небольшом количестве компьютеров. Если же число рабочих станций, допустим, превышает 25-30 – то это будет вызывать определенные сложности.

В отличие от одноранговой сети, в иерархической существует один или несколько главных компьютеров — серверов. Все остальные компьютеры сети называются клиентами или рабочими станциями (workstations). Сервер — это специальный компьютер, который предоставляет определенные услуги другим компьютерам. Серверы обычно представляют собой высокопроизводительные ПК с серверной операционной системой (например, Windows Server 2003 или Windows Server 2008), отказоустойчивыми дисковыми массивами и системой защиты от сбоев. Как правило, на этих компьютерах локальные пользователи не работают, поэтому сервера принято называть выделенными. Серверы управляют

сетью и хранят информацию, которую совместно используют остальные компьютеры сети. Существуют различные виды серверов: серверы баз данных, файловые серверы, серверы печати, почтовые серверы, Web-серверы, и т.д.

Иерархические сети обладают рядом преимуществ по сравнению с одноранговыми:

1. выход из строя рабочих станций никак не сказывается на работоспособности сети в целом;
2. проще организовать локальные сети с большим количеством рабочих станций;
3. администрирование сети осуществляется централизованно — с сервера;
4. обеспечивается высокий уровень безопасности данных.

Тем не менее, клиент-серверной архитектуре присущ ряд недостатков:

- неисправность или сбой единственного сервера может парализовать всю сеть;
- наличие выделенных серверов повышает общую стоимость сети;
- it-персонал должен обладать достаточными знаниями и навыками администрирования домена.

Выбор архитектуры сети зависит от специфики организации, назначения сети и количества рабочих станций. От выбора типа сети зависит также и ее дальнейшее будущее: расширяемость, возможность использования того или иного ПО и оборудования, надежность сети и многое другое.[6]

1.2.3 Сетевые топологии

Для организации взаимодействия сетевого оборудования при построении сетей используются различные сетевые топологии.

«Топология сети – геометрическая форма и физическое расположение компьютеров по отношению к друг другу. Топология сети позволяет сравнивать и классифицировать различные сети. Различают три основных вида топологий:» [6]

1) Шина;

2) Кольцо;

3) Звезда.

Топология «Шина».

При построении сети по шинной схеме каждый компьютер присоединяется к общему кабелю, на концах которого устанавливаются терминаторы. Сигнал проходит по сети через все компьютеры, отражаясь от конечных терминаторов.

Шина проводит сигнал из одного конца сети к другому, при этом каждая рабочая станция проверяет адрес послания, и, если он совпадает с адресом рабочей станции, она его принимает. Если же адрес не совпадает, то сигнал уходит по линии дальше.

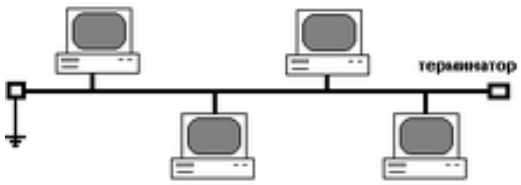

Рисунок 1.1 – Топология «Шина»

У данной топологии есть свои достоинства и недостатки. К достоинствам можно отнести простоту и гибкость соединений, отсутствие необходимости в большом количестве кабеля. Также, при использовании данной топологии, отказ любой из рабочих станций не влияет на работу всей сети.

Среди недостатков же стоит отметить трудность обнаружения дефектов соединений, малую пропускную способность сети, ограничение на длину кабеля и

количество рабочих станций. Помимо этого, разрыв кабеля или другие неполадки в соединении могут остановить работу всей сети.

Топология «Кольцо».

«Кольцо» – это топология, в которой рабочие станции подключены последовательно друг к другу, образуя замкнутое кольцо. Данные передаются от одной рабочей станции к другой в одном направлении (по кругу). Каждый компьютер работает как повторитель, ретранслируя сообщения к следующему участнику сети. Если компьютер получает данные, предназначенные для другого компьютера – он передает их дальше по кольцу.

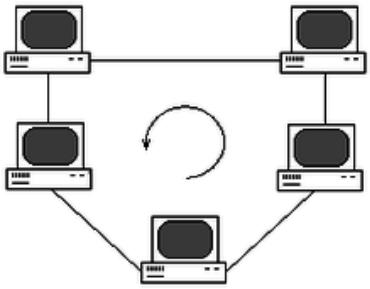

Рисунок 1.2 – Топология «Кольцо»

У данной топологии, как и у предыдущей, также имеются свои плюсы и минусы. Сеть с такой топологией легко организовать, в ней не требуется дополнительного оборудования, скорость передачи стабильна, вне зависимости от нагрузки на сеть.

Однако, поскольку сигнал проходит через каждый компьютер, сбой одного из них приводит к нарушению работы всей сети. Помимо этого, на каждом компьютере должно быть установлено 2 сетевых интерфейса. Также весьма проблематична конфигурация и настройка сети, поиск неисправностей.

Топология «Звезда».

Топология «Звезда» - схема соединения, при которой каждый компьютер подсоединяется к сети при помощи отдельного соединительного кабеля. Один конец кабеля соединяется с гнездом сетевого адаптера компьютера, другой подсоединяется к центральному устройству, называемому концентратором (hub). Концентратор управляет движением пакетов в сети.

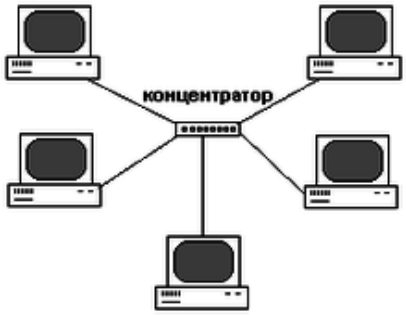

Рисунок 1.3 – Топология «Звезда»

Среди достоинств данной топологии следует отметить возможность мониторинга сети и централизованного управления сетью, хорошую расширяемость и возможность модернизации без остановки работы сети, а также возможность быстрой локализации дефектов соединений.

Недостатками топологии является дороговизна реализации, связанная с необходимостью большого количества кабеля. Помимо этого, отказ концентратора приводит к остановке работы всей сети.

Несмотря на недостатки, топология «звезда» наиболее распространена, поскольку является наиболее надёжной среди остальных.

Топология «Token Ring».

Помимо физических топологий также существуют логические. Логические топологии определяют направления потоков данных между узлами сети и способы передачи данных вне зависимости от их физического расположения.

В крупных компаниях обычно используют комбинированные топологии. Одной из самых известных является топология «Token Ring». Она представляет собой сочетание физической «звезды» и логического «кольца».

В данной топологии все рабочие станции подключаются к центральному концентратору как в топологии «звезда». Но при этом каждая станция соединяется только с предыдущей и последующей станциями. Таким образом, рабочие станции связаны петлей кабеля, по которой пакеты данных передаются от одной станции к другой и каждая ретранслирует эти посланные пакеты. В каждой рабочей станции для этого имеется приемо-передающее устройство, которое позволяет управлять прохождением данных в сети.

Рисунок 1.4 – Топология «Token Ring»

Концентратор создаёт первичное (основное) и резервное кольца. Если в основном кольце произойдёт обрыв, то его можно обойти, воспользовавшись резервным кольцом, так как используется четырёхжильный кабель. Отказ станции или обрыв линии связи рабочей станции не влечет за собой отказ сети как в топологии «кольцо», потому что концентратор отключит неисправную станцию и замкнет кольцо передачи данных.

В архитектуре Token Ring маркер передаётся от узла к узлу по логическому кольцу, созданному центральным концентратором. Такая маркерная передача осуществляется в фиксированном направлении.

Для передачи данных рабочие станции должны сначала дождаться прихода свободного маркера. В маркере содержится адрес станции, пославшей этот маркер, а также адрес той станции, которой он предназначается. После этого отправитель передает маркер следующей в сети станции для того, чтобы и та могла отправить свои данные.

Один из узлов сети создаёт маркер, который отправляется в кольцо сети. Такой узел выступает в качестве активного монитора, который следит за тем, чтобы маркер не был утерян или разрушен.

Топология Token Ring является довольно дорогой в плане реализации, так как требует наличия концентратора и большого расхода кабеля. Несмотря на это, топология является весьма распространенной, поскольку позволяет обеспечить равный доступ ко всем рабочим станциям, а также высокую надежность, так как сеть устойчива к неисправностям отдельных станций и к разрывам соединения отдельных станций.

1.2.4 Коммуникационное оборудование

К коммуникационному оборудованию относятся всевозможные аппаратные средства, необходимые для объединения узлов компьютерной сети, ее расширения и выполнения других функций.

Хаб (hub, концентратор).

Хаб – многопортовое устройство для соединения нескольких сегментов сети, количество которых равно количеству портов на самом концентраторе. Классический концентратор (пассивный), в своей архитектуре не имеет функционального элемента, предназначенного, для определения узла сети и локализации трафика. По этой причине хаб может распознать только с какого порта поступил сигнал, а отправляет пакеты на все сегменты сети, что приводит к бесполезной нагрузке на сеть. При большом количестве абонентов сети, в комбинированных топологиях, эффективность работы сети понижается и есть большая вероятность возникновения коллизии (сталкивания пакетов файлов). В настоящее время концентраторы вытеснены коммутаторами.

Свитч (switch, коммутатор).

Свитч – устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного сегмента. В отличие от концентратора, распространяет трафик только к нужному порту, а не ко всем. Это не загружает сеть, и препятствует образованию коллизии.

Свитч работает на канальном уровне модели OSI и может соединять сегменты сети по MAC-адресам. Принцип работы коммутатора таков: Коммутатор хранит в памяти таблицу MAC, в которой прописано соответствие MAC-адреса узла и порта компьютера. При первом включении свитч, таблица пуста, и он работает в режиме обучения. Свитч отправляет пакеты на все порты, а потом анализирует их, определяя MAC-адрес отправителя, и записывает его в таблицу. И теперь, если на один из портов коммутатора поступит пакет, предназначенный для этого компьютера, этот пакет будет отправлен только на соответствующий порт. В результате трафик локализируется.

Свитчи можно разделить на управляемые и неуправляемые. Управляемые сложные свитчи позволяют контролировать коммутацию на канальном и сетевом уровне модели OSI. Управление свитчем может осуществляться посредством протокола Web-интерфейса, SNMP и RMON.

Репитер (repeater, повторитель).

Репитер – устройство для соединения двух сегментов сети, как правило шинных. Основное назначения повторителя, это восстановление исходного уровня информационного сигнала, за счет чего возможно расширение сети. Репитер применяется либо внутри одного сегмента шины либо соединяет два. На сегодняшний день репитеры мало распространены, так как топология шина, применяется редко.

Мост (bridge).

Мост – сетевое оборудование для объединения сегментов локальной сети. Сетевой мост работает на канальном уровне модели OSI, обеспечивая ограничение домена коллизий. Мосты направляют фреймы данных в соответствии с MAC-адресами фреймов. В общем случае коммутатор (свитч) и мост аналогичны по функциональности, разница заключается во внутреннем устройстве: мосты обрабатывают трафик, используя центральный процессор, коммутатор же использует коммутационную матрицу (аппаратную схему для коммутации пакетов). В настоящее время мосты практически не используются (так как для работы требуют производительный процессор), за исключением ситуаций, когда связываются сегменты сети с разной организацией первого уровня, например, между xDSL соединениями, оптоволокном, Ethernet'ом.

В некоторых видах высокоуровневого сетевого оборудования и операционных систем имеется режим моста (bridging), где используется для «логического объединения» нескольких портов в единое целое (с точки зрения вышестоящих протоколов), превращая указанные порты в виртуальный коммутатор.

Роутер (router, маршрутизатор).

Роутер – сетевое устройство, на основании информации о топологии сети и определённых правил, принимающее решения о пересылке пакетов сетевого уровня (уровень 3 модели OSI) между различными сегментами сети. Работает на более высоком уровне, нежели коммутатор и сетевой мост. Обычно роутер использует адрес получателя, указанный в пакетах данных, и определяет по таблице маршрутизации путь, по которому следует передать данные. Если в таблице маршрутизации для адреса нет описанного маршрута, пакет отбрасывается. Существуют и другие способы определения маршрута пересылки пакетов, когда, например, используется адрес отправителя, используемые протоколы верхних уровней и другая информация, содержащаяся в

заголовках пакетов сетевого уровня. Нередко маршрутизаторы могут осуществлять трансляцию адресов отправителя и получателя, фильтрацию транзитного потока данных на основе определённых правил с целью ограничения доступа, шифрование/дешифрование передаваемых данных.

Таблица маршрутизации содержит информацию, на основе которой маршрутизатор принимает решение о дальнейшей пересылке пакетов. Таблица состоит из некоторого числа записей - маршрутов, в каждой из которых содержится адрес сети получателя, адрес следующего узла, которому следует передавать пакеты и некоторый вес записи - метрика. Метрики записей в таблице играют роль в вычислении кратчайших маршрутов к различным получателям.

Шлюз (gateway).

Шлюз – аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной). Сетевые шлюзы работают почти на всех известных операционных системах. Основная задача сетевого шлюза -конвертировать протокол между сетями.

Роутер сам по себе принимает, проводит и отправляет пакеты только среди сетей, использующих одинаковые протоколы. Сетевой шлюз же может с одной стороны принять пакет, сформатированный под один протокол (например, Apple Talk) и конвертировать в пакет другого протокола (например, TCP/IP) перед отправкой в другой сегмент сети. Сетевые шлюзы могут быть аппаратным решением, программным обеспечением или тем и другим вместе, но обычно это программное обеспечение, установленное на роутер или компьютер. Сетевой шлюз должен понимать все протоколы, используемые роутером.

В качестве среды передачи сигналов между узлами сети используются различные типы кабелей. Рассмотрим наиболее распространенные из них.

Коаксильный кабель.

В недалеком прошлом – самый распространенный тип кабеля. Коаксиальный кабель состоит из медной жилы, окружающей ее изоляции, экрана в виде металлической оплетки и внешней оболочки. Если кабель, кроме металлической оплетки, имеет и слой фольги, он называется кабелем с двойной экранизацией. Коаксиальный кабель более помехоустойчив, чем кабель типа «витая пара», однако на данный момент почти полностью вытеснен последним, поскольку является достаточно дорогим и сложным в плане монтажа. Тем не менее, данный тип кабеля всё еще используется для передачи на большие расстояния.

Витая пара

Самым распространенным на данный момент является кабель типа «витая пара» (twisted pair). Представляет собой два перевитых вокруг друг друга изолированных медных провода. Существует два типа тонкого кабеля: неэкранированная (unshielded) витая пара (UTP) и экранированная (shielded) витая пара (STP). Несколько витых пар часто помещают в одну защитную оболочку. Их количество в таком кабеле может быть разным. Завивка проводов позволяет избавиться от электрических помех, наводимых соседними парами и другими источниками, например двигателями, реле и трансформаторами. Скорость передачи в данном типе кабеля достигает 100 Мбит в секунду. Витая пара является простым и недорогим решением при построении сетей, поэтому используется почти повсеместно.

Оптоволоконный кабель

В оптоволоконном кабеле цифровые данные распространяются по оптическим волокнам в виде модулированных световых импульсов. Данный кабель представляет собой прозрачную нить, выполненную из стекла или пластика. Нить покрыта стеклянной оболочкой, коэффициент преломления света в которой ниже чем у нити. Это обеспечивает полное отражение света внутри

кабеля. Оптоволоконные линии предназначены для перемещения больших объемов данных на очень высоких скоростях (до 200 000 Мбит/с), так как сигнал в них практически не затухает и не искажается.

Помимо этого, отсутствие электрических импульсов в данном кабеле исключает возможность несанкционированного перехвата трафика. Оптоволокно также отличается хорошим уровнем защиты от помех.

На сегодняшний день оптоволоконные линии связи распространены не так сильно, как например, витая пара, поскольку их установка является довольно дорогостоящей. Чаще всего оптоволокно используется при передаче информации на большие расстояния.

1.3 Угрозы и методы защиты ТКС

Одной из наиболее важных задач при построении вычислительных сетей является обеспечение их защиты от несанкционированного доступа. Существует множество потенциальных угроз безопасности данных в сети.

«Под угрозой безопасности информации понимаются события или действия, которые могут привести к искажению, несанкционированному использованию или даже к разрушению информационных ресурсов управляемой системы, ее программных и аппаратных средств.»[2]

К угрозам безопасности относятся как внешние источники угроз – объекты или субъекты, находящиеся за пределами вашей локальной сети, действия которых вы не можете контролировать, так и внутренние – это могут быть как некачественные технические или программные средства обработки информации, так и сотрудники организации, которые случайно или намеренно передают конфиденциальную информацию в руки злоумышленника. К внешним угрозам относятся компьютерные вирусы, DoS-атаки и стихийные бедствия.

Компьютерный вирус – вид вредоносного программного обеспечения, способного создавать копии самого себя и внедряться в код других программ,

системные области памяти, загрузочные секторы, а также распространять свои копии по разнообразным каналам связи.

Целью вируса является: удаление или хищение файлов, приведение в негодность структур размещения данных, блокирование работы пользователей или же приведение в негодность аппаратных комплексов компьютера и т. п.

DoS-атака (с англ. Denial of Service — «отказ в обслуживании») – это атака на сервер вычислительной системы, основной целью которой является выведение системы из строя путём подачи большого количества ложных запросов. В результате такой атаки сервера, обслуживающие систему, вынуждены обрабатывать чрезмерный объём ложных запросов, и система становится недоступной для простого пользователя.

В настоящее время эффективных методов защиты от DoS-атак не существует.

Для защиты локальной сети от внутренних угроз безопасности необходимо использовать только качественное и проверенное оборудование и программное обеспечение. Если локальная сеть состоит из множества рабочих станций и вы не можете поручиться за честность или компетентность их пользователей, то возникает необходимость внедрения DLP-систем.

DLP-системы позволяют отслеживать и пресекать нежелательные действия участников локальной сети. Таким образом, DLP-системы предотвращают утечки конфиденциальной информации. Помимо этого, с помощью DLP-систем можно контролировать присутствие сотрудников на рабочем месте и их благонадежность.

Для защиты рабочих станций конечных пользователей от внешних источников угроз применяются следующие средства:

- Использование межсетевого экрана;
- Использование антивирусного ПО;

Межсетевой экран (firewall) – программный или аппаратный элемент компьютерной сети, который позволяет фильтровать приходящий из внешней сети трафик в соответствии с заданными правилами.

Среди задач, которые решают межсетевые экраны, основной является защита сегментов сети или отдельных хостов от несанкционированного доступа с использованием уязвимых мест в протоколах сетевой модели OSI или в программном обеспечении, установленном на компьютерах сети. Межсетевые экраны пропускают или запрещают трафик, сравнивая его характеристики с заданными шаблонами.

Антивирусное программное обеспечение позволяет обнаруживать компьютерные вирусы и другие потенциально опасные программы и блокировать их работу еще до нанесения ими какого либо ущерба. Помимо этого, в ряде случаев, антивирусы могут восстанавливать уже зараженные файлы.

Помимо описанных источников угроз безопасности необходимо также отметить возможность перехвата информации с помощью программ-снифферов, которые позволяют просматривать и анализировать весь трафик, проходящий внутри сети. Для этого злоумышленнику зачастую достаточно стать участником данной сети. Риск перехвата может быть снижен при использовании коммутаторов в сети вместо концентраторов, поскольку данные в первых передаются адресату по прямому маршруту, минуя остальные сегменты сети.

Самым верным способом защиты информации от несанкционированного доступа является использование криптографических методов. Обычно они включают в себя непосредственно шифрование конфиденциальной информации и создание электронно-цифровой подписи (ЭЦП), гарантирующей подлинность источника. Таким образом, данные методы позволяют выполнить сразу два основных критерия безопасности информации – конфиденциальность и целостность данных, при этом сохраняя ее доступность.

«Криптография (с греческого — «тайнопись») – наука о методах обеспечения конфиденциальности, целостности данных и аутентификации. В основе криптографии лежат методы шифрования информации.»[12]

«Шифрование – это обратимый процесс преобразования информации, позволяющий скрыть ценные сведения, содержащиеся в ней. Шифрование производится при помощи секретного ключа, о котором известно только отправителю и получателю. Таким образом, злоумышленнику, чтобы получить доступ к информации, необходимо завладеть этим ключом.»[10]

В большинстве случаев шифрование не является абсолютно надёжным средством защиты, поскольку ключ можно подобрать случайным образом, насколько сложным бы он ни был. Всё упирается во время и объем вычислительных ресурсов, необходимых на подбор. Чем сложнее ключ, тем больше потребуется времени, чтобы подобрать его. Однако, на это могут уйти месяцы и, возможно, даже годы. За это время информация обычно перестает быть актуальной, поэтому криптография, как метод защиты информации, остается весьма надежным. Ниже будут рассмотрены основные алгоритмы шифрования, их классификация и критерии их надёжности.

1.3.1 Классификация алгоритмов шифрования

На сегодняшний день существует множество различных алгоритмов шифрования. Но все они делятся на два основных класса – симметричные и ассиметричные.

В симметричных алгоритмах для шифрования и дешифровки сообщений используется один и тот же криптографический ключ. Этот ключ должен быть известен всем участникам сеанса связи и храниться в секрете. Основным недостатком такого подхода является необходимость незаметной передачи ключа собеседнику. [13]

В ассиметричных же алгоритмах шифрования используются два ключа – открытый и закрытый. Первый – для шифровки сообщений, второй – для дешифровки. Открытый ключ можно передавать по незащищенному каналу связи, поскольку, в случае перехвата, он не даст злоумышленнику никакой полезной информации. Закрытый ключ хранится в тайне и не передается вообще. Схема работает следующим образом: для передачи сообщения участник сеанса связи запрашивает открытый ключ у собеседника, затем шифрует свое сообщение с помощью этого ключа и отправляет его. Собеседник же, в свою очередь, дешифрует это сообщение с помощью своего закрытого ключа, известного только ему. Передача сообщения в обратную сторону происходит аналогично. Таким образом, каждый участник сеанса связи имеет свой открытый и закрытый ключ. Эта схема гарантирует отсутствие риска перехвата секретного ключа и, следовательно, является более надежной.

Однако при таком подходе имеются свои недостатки, по сравнению с симметричным шифрованием, а именно:

- В алгоритм сложнее внести изменения
- Большой размер ключа
- Шифрование/дешифровка с использованием пары ключей проходит на два-три порядка медленнее

- Требуются существенно бóльшие вычислительные ресурсы

Симметричные алгоритмы шифрования, в свою очередь, подразделяются на потоковые и блочные. Потоковые алгоритмы позволяют шифровать информацию побитово/посимвольно, в то время как блочные разделяют сообщение на блоки, характерный размер которых меняется в пределах 64–256 бит, и шифруют их как единое целое. Если число символов в сообщении не кратно размеру блока, то последний блок дополняется нулями.

Потоковые алгоритмы, в отличие от блочных, являются более простыми и дешевыми в реализации, а также обладают более высокой скоростью шифрования. Однако они являются менее надежными, поскольку преобразование с ключом происходит всего один раз и, при недостаточной длине ключа, могут быть обнаружены закономерности в шифре, что в конечном итоге позволит криптоаналитику расшифровать сообщение.

Основным критерием при выборе алгоритма является его криптостойкость – способность алгоритма противостоять криптоанализу. Стойким считается алгоритм, успешная атака на который требует от атакующего обладания недостижимым на практике объёмом вычислительных ресурсов либо настолько значительных затрат времени на раскрытие, что к его моменту защищённая информация утратит свою актуальность.[10]

1.3.2 Основные алгоритмы шифрования

Несмотря на то, что криптографические методы существуют уже достаточно давно, многие из старых методов до сих пор используются в основе современных алгоритмов шифрования. Некоторые из них рассмотрены ниже.

Шифр Вернама.

Шифр Вернама – это шифр с симметричным поточным алгоритмом шифрования, изобретённый в 1917 году сотрудником компании AT&T – Гилбертом Вернамом.

Данный шифр работает следующим образом: каждый символ исходного текста объединяется с ключом при помощи булевой функции «Исключающее ИЛИ» (XOR). Ключом является случайная последовательность бит. Так как «Исключающее ИЛИ» является обратимой операцией, то для расшифровки сообщения на приёмной стороне достаточно просто объединить зашифрованное сообщение с тем же ключом:

Зашифрованный текст = открытый текст $\oplus$ ключ;

Открытый текст = зашифрованный текст $\oplus$ ключ;

Основное достоинство шифра Вернама заключается в том, что, не смотря на свою простоту, он может обладать абсолютной криптографической стойкостью, то есть быть полностью устойчивым к взлому. Для этого должны выполняться следующие условия:

- Длина ключ должна быть не меньше длины сообщения;
- Ключ должен быть сгенерирован полностью случайным образом, без использования каких-либо алгоритмов;
- Ключ должен использоваться только один раз.

Выполнение всех трех условий является весьма проблематичным. Поэтому, в большинстве случаев, на практике об абсолютно стойких шифрах говорить не приходится. Тем не менее, существуют алгоритмы генерации псевдослучайных последовательностей, использование которых позволяет достаточно надёжно зашифровывать сообщения.

Гаммирование.

Гамми́рование — метод симметричного шифрования, заключающийся в «наложении» последовательности, состоящей из случайных чисел, на открытый текст. Обычно, как и в шифре Вернама, открытый текст складывается с последовательностью с помощью операции «Исключающее ИЛИ». Последовательность случайных чисел называется гамма-последовательностью и образуется при помощи генератора псевдослучайных последовательностей.

Требования к гамме:

- Для шифрования каждого нового сообщения нужно использовать новую гамму, поскольку при сложении двух шифрограмм, образованных от одной гаммы, используя операцию XOR, мы получим сумму двух исходных сообщений:

$Z_1 \oplus Z_2 = (X_1 \oplus Y) \oplus (X_2 \oplus Y) = X_1 \oplus X_2$,

где Z_1 и Z_2 - зашифрованные сообщения;

X_1 и X_2 - исходные сообщения;

Y - гамма;

- Для формирования гаммы желательно использовать аппаратные генераторы случайных чисел, основанные на физических процессах, поскольку программные генераторы используют алгоритмы, что дает злоумышленнику возможность проведения успешного криптоанализа.
- Длина гаммы должна быть не меньше длины защищаемого сообщения. В противном случае, для получения открытого текста потребуется подобрать длину гаммы, проанализировать блоки шифротекста угаданной длины, подобрать биты гаммы.

К современным методам шифрования можно отнести следующие:

Сеть Фейстеля.

Сеть Фейстеля – один из методов построения блочных шифров. Представляет собой разбиение обрабатываемого блока данных на несколько подблоков (чаще всего - на два), один из которых, при помощи раундового ключа, обрабатывается некоей функцией f и накладывается на один или несколько остальных подблоков при помощи XOR. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево. Такое преобразование выполняется несколько циклов или *раундов*. Ключ раунда вычисляется из исходного по какому-либо математическому правилу.

$M_{i+1} = M_{i-1} \oplus f(K_i M_i)$, где M_{i+1} – шифруемый блок, K*i* – ключ раунда.

Те же операции происходят и при расшифровке, с тем только отличием, что ключи применяются в обратном порядке. Криптостойкость данного метода напрямую зависит от количества раундов – чем больше раундов, тем выше криптостойкость. В роли функции f может выступать любая операция, однако, как правило, эта функция представляет собой последовательность зависящих от ключа нелинейных замен, перемешивающих перестановок и сдвигов. Схемы, реализующие эти преобразования, называются SP-сетями.

Ввиду простоты операций сеть Фейстеля легко реализовать как программно, так и аппаратно. Большинство современных блочных шифров (DES, RC2, RC5, RC6, Blowfish, FEAL, CAST-128, TEA, XTEA, XXTEA и др.) используют сеть Фейстеля в качестве основы. [10][13]

RSA.

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

RSA-ключи генерируются следующим образом:

- Выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое);
- Вычисляется их произведение n=p*q, которое называется модулем;
- Вычисляется значение функции Эйлера от числа n: $\varphi(n) = (p-1) * (q-1)$;
- Выбирается целое число e ($1 < e < \varphi(n)$) взаимно простое со значением функции $\varphi(n)$ (желательно, чтобы e было не меньше 5);
- Вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$.
- Пара {e,n} публикуется в качестве открытого ключа RSA.
- Пара {d,n} играет роль закрытого ключа RSA и держится в секрете.

Сообщения шифруются по следующей формуле:

$c = E(m) = m^e \bmod n$, где m – открытый текст;

Формула для расшифровки:

$m = D(c) = c^d \bmod n$

Стойкость алгоритма основывается на сложности вычисления обратной функции к функции шифрования. Вычисление обратного элемента по модулю не является сложной задачей, однако злоумышленнику неизвестно значение $\varphi(n)$. Для вычисления функции Эйлера от известного числа n необходимо знать разложение этого числа на простые множители. Нахождение таких множителей и является сложной задачей.

Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи, и является на данный момент одной из самых актуальных. [14]

1.3.3 ЭЦП

Для того чтобы убедиться, что полученное сообщение в процессе передачи не было модифицировано третьими лицами, используется электронная цифровая подпись. Такая подпись является цифровым аналогом собственноручной подписи.

Обычно ЭЦП ставится не на передаваемый документ, а на его хэш. Для вычисления хэша используются криптографические хэш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функцией называется алгоритм, конвертирующий строку произвольной длины (сообщение) в битовую строку фиксированной длины, называемой хэш-кодом, проверочной суммой или цифровым отпечатком. Хэш-функции не являются частью алгоритма ЭП, поэтому в схеме может быть использована любая надёжная хэш-функция.

Известные алгоритмы хэширования:

MD4 – алгоритм хэширования, разработанный Рональдом Л. Ривестом из RSA Data Security, Inc. Это быстрый алгоритм (на 32-битных процессорах) и его используют при вычислении хэшей в peer-to-peer сети EDonkey 2000. Хэш-код представляет шестнадцатеричное число из 32 символов. В настоящее время считается ненадёжным.

MD5 – ещё один алгоритм хэширования, разработанный Рональдом Л. Ривестом. Представляет улучшенную версию MD4. В течении многих лет MD5 был стандартом интернет, но сейчас считается сломанным. Хэш-код представляет шестнадцатеричное число из 32 символов.

SHA1 (Secure Hash Algorithm 1) – алгоритм хэширования, разработанный NSA в 1993. Он примерно в 2-3 раза медленнее алгоритма MD5. Хэш-код представляет шестнадцатеричное число длины 40.

Tiger – современная хэш-функция изобретённая Россом Андерсом и Эли Бихамом. Была специально придумана такой, чтобы быстро вычисляться на 64-битных процессорах. Хэш-код представляет шестнадцатеричное число длины 48.

После того, как хэш вычислен, на него применяется один из алгоритмов шифрования. В основном, используются асимметричные алгоритмы шифрования (например, RSA). Хэш-код, в таком случае, шифруется при помощи закрытого ключа, а расшифровывается открытым. Таким образом, подпись может просмотреть любой человек, знающий открытый ключ, но при этом не может подделать ее. [11][12]

Выводы по главе.

Рассмотрев основные принципы построения телекоммуникационных систем, возможные угрозы и методы защиты от них, можно сделать вывод, что для гарантированного обеспечения безопасности необходимо применять различные методы защиты, начиная от правильного выбора аппаратных и программных средств, обрабатывающих информацию, и заканчивая внедрением дополнительной защиты в виде использования криптографических методов.

Надежная криптографическая система должна удовлетворять следующим требованиям:

- Процедуры зашифровывания и расшифровывания должны быть "прозрачны" для пользователя;
- Дешифрование закрытой информации должно быть максимально затруднено;
- Содержание передаваемой информации не должно сказываться на эффективности криптографического алгоритма.

ГЛАВА II. Проектирование защищённого канала связи

2.1 Выбор сетевых компонентов

Для достижения цели, поставленной в данной работе, в первую очередь необходимо разработать архитектуру защищённого канала связи.

Поскольку в данной работе не подразумевается проектирование какой-то конкретной сети, а сети в самом общем ее случае, то канал может быть построен при помощи любых из описанных в первой главе сетевых компонентов. Однако если сеть предполагается не домашней, а, например, корпоративной, то среди коммуникационного оборудования рекомендуется выбирать коммутатор либо маршрутизатор, поскольку данные устройства, как уже было отмечено ранее, направляют приходящий трафик в конкретный сегмент сети, что значительно снижает риск перехвата трафика внутри сети. Помимо этого, для защиты от внешних атак на рабочие станции необходимо установить антивирусное ПО и файрволы (межсетевые экраны). Это позволит фильтровать приходящий трафик и защитить информацию, хранящуюся на рабочей станции, от хищения, модификации и/или уничтожения. На рабочие станции, находящиеся в одной локальной сети, достаточно одного межсетевого экрана, поскольку чаще всего необходимо фильтровать трафик, приходящий из глобальной сети. Такой межсетевой экран обычно представляет собой отдельное устройство, устанавливаемое на границе локальной сети. Если же существует угроза внутри локальной сети (например, среди сотрудников или посетителей организации), то файрвол ставится на каждую рабочую станцию (чаще всего в виде программного обеспечения).

2.2 Проектирование программы-мессенджера

Для того, чтобы организовать связь между двумя удаленными компьютерами, помимо аппаратных компонентов сети, необходимо программное обеспечение, с помощью которого участники сети будут вводить и получать необходимую информацию. В данной работе поставлена задача разработать

программу-массенджер для передачи мгновенных сообщений, иными словами – обыкновенный сетевой чат.

Программа должна организовывать соединение на логическом уровне, то есть обращаться к конкретному IP-адресу и порту собеседника. Программа работает на архитектуре «клиент-сервер». Таким образом, один из участников сеанса связи выступает в роли сервера, а другие подключаются к нему в роли клиентов. Клиент в проектируемой программе является так называемым «толстым» клиентом, поскольку все криптографические операции при отправке и получении сообщений выполняются непосредственно на нем.

Клиентом указывается IP-адрес сервера, к которому необходимо подключиться. Затем указывается порт, по которому будет происходить соединение. Необходимо, чтобы порт сервера и порт клиента совпадали. Также, каждому участнику сеанса связи нужно добавить указанный порт в исключения брандмауэра (файрвола), чтобы он не заблокировал соединение.

Для передачи сообщений используется протокол TCP. В отличие от протокола UDP, используемого для передачи потоковой информации (голосовая и видео связь, многопользовательские игры и др.), данный протокол позволяет обеспечить надёжную доставку пакетов.

2.3 Проектирование алгоритма шифрования

Основным методом защиты канала связи является использование криптографических методов, а именно – применение симметричного шифрования на уровне отправки сообщения. Это позволит обеспечить конфиденциальность информации при ее возможном перехвате.

При разработке алгоритма шифрования учитывались следующие требования к алгоритму:[15]

1) Зашифрованное сообщение должно читаться только при использовании ключа.

2) Ошибки в шифровании не должны приводить к потере информации.

3) Число операций, необходимых, для расшифровки путем перебора всевозможных ключей должно иметь строгую нижнюю границу и выходить за допустимые пределы вычислительных и временных ресурсов.

4) Знание алгоритма шифрования не должно влиять на надежность защиты (за исключением образующих функций).

5) Структурные элементы алгоритма шифрования должны оставаться неизменными.

6) Дополнительные биты, вводимые в сообщение, в процессе шифрования должны быть полностью и надежно скрыты в шифрованном тексте.

7) Любой ключ из множества возможных должен обеспечивать надежную защиту информации.

8) Шифрование и дешифрование должно происходить за относительно небольшой промежуток времени, поскольку общение в чате происходит в онлайн-режиме.

Для разработки алгоритма шифрования, описываемого в данной работе, в качестве его основы была использована классическая сеть Фейстеля. Алгоритм данной сети изображен на рисунке 2.1.

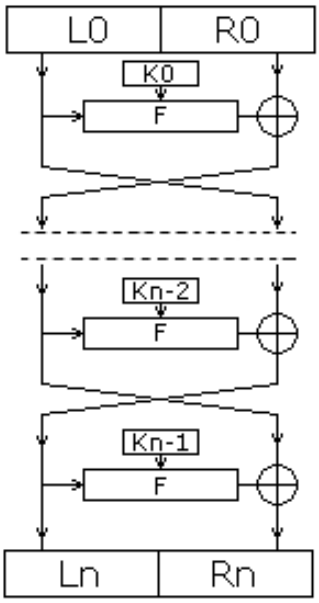

Рисунок 2.1 – Классическая сеть Фейстеля

Данный выбор обоснован тем, что сеть Фейстеля является довольно простым методом построения шифра в плане программной реализации, но в то же время достаточно криптостойким.

Разрабатываемый алгоритм состоит из шестидесяти раундов (итераций). Длина блока составляет восемь символов. Алгоритм имеет следующие уникальные свойства:

- Раундовые ключи меняются с каждым новым обрабатываемым блоком (количество смен зависит от длины основного ключа);
- Ключ нового раунда не зависит от ключа предыдущего;
- Пользователь может ввести ключ любой длины, и от этого будет зависеть уровень криптостойкости алгоритма – чем длиннее ключ, тем надежнее алгоритм.

Перечисленные свойства позволяют увеличить криптостойкость алгоритма. Помимо этого, криптостойкость также увеличивается за счет большого количества раундов.

Выводы по главе.

Таким образом, для построения защищенного канала связи между удаленными рабочими станциями, необходимо выполнить следующие процедуры:

- Организовать физическое соединение этих рабочих станций при помощи коммуникационного оборудования с учетом указанных рекомендаций;
- Разработать программное обеспечение для отправки и приема сообщений (мессенджер);
- Разработать алгоритм шифрования и внедрить его в мессенджер.

Модель защищенного канала связи изображена на рисунке 2.2.

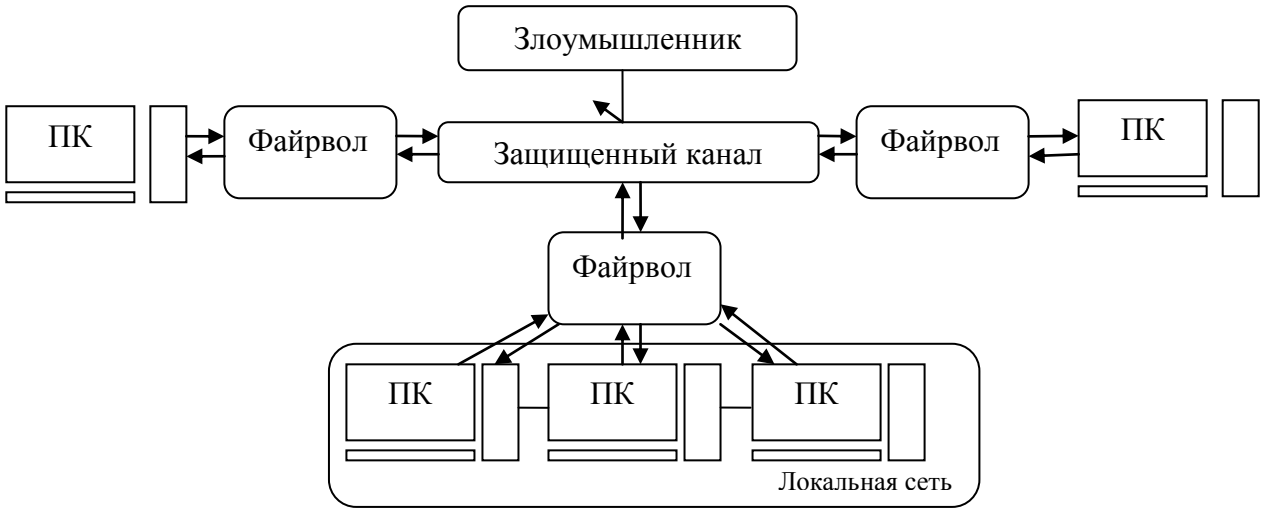

Рисунок 2.2 – Модель защищенного канала связи.

Под аббревиатурой «ПК» в данной схеме понимается персональный компьютер, с установленными криптомессенджером и антивирусом. Компьютер может выполнять роль как сервера, так и клиента.

ГЛАВА III. Реализация мессенджера и алгоритма шифрования

3.1 Общие сведения о программе

Мессенджер представляет собой программу с расширением .exe, работающую в операционной системе Windows. Для установки соединения мессенджер должен быть запущен у всех участников сеанса связи. Количество участников неограниченно.

В отличие от других подобных программ, разрабатываемый мессенджер не имеет удаленного промежуточного сервера для хранения информации, что снижает вероятность попадания сообщения не в те руки. Помимо этого программа не имеет никаких дополнительных библиотек, не требует установки и начинает работать сразу после ее запуска. Это снижает требования к мощности оборудования, на котором запускается программа. Размер программы на жестком диске (при использовании файловой системы NTFS) составляет всего 422 КБайта, что обеспечивает легкую возможность ее передачи, как на съемном носителе, так и по сети.

3.2 Выбор среды разработки

Для написания программы была выбрана среда разработки Borland Delphi 7. Delphi 7 является языком высокого уровня и позволяет быстро и эффективно создавать приложения. Он включает в себя компилятор кода и предоставляет средства визуального программирования. В основе Delphi лежит язык Object Pascal, который является расширением объектно-ориентированного языка Pascal. В Delphi также входят локальный SQL-сервер, генераторы отчетов, библиотеки визуальных компонентов, и прочее хозяйство, необходимое для того, чтобы чувствовать себя совершенно уверенным при профессиональной разработке информационных систем или просто программ для Windows-среды.[16]

Прежде всего, Delphi предназначен для профессиональных разработчиков, желающих очень быстро разрабатывать приложения в архитектуре клиент-сервер. Delphi производит небольшие по размерам (до 15-30 Кбайт) высокоэффективные исполняемые модули (.exe и .dll).

Преимущества Delphi по сравнению с аналогичными программными продуктами:

– быстрота разработки приложения;

– высокая производительность разработанного приложения;

– низкие требования разработанного приложения к ресурсам компьютера;

– наращиваемость за счет встраивания новых компонент и инструментов в среду Delphi;

– возможность разработки новых компонент и инструментов собственными средствами Delphi;

– удачная проработка иерархии объектов.

Система программирования Delphi рассчитана на программирование различных приложений и предоставляет большое количество компонентов для этого.[17]

3.3 Функциональные требования

При разработке программы были учтены следующие требования к ее функционалу:

- Выбор режима работы программы – режим клиента или режим сервера;
- Возможность ввода имени пользователя;
- Возможность указания порта для установки соединения;
- Возможность ввода IP-адреса для подключения, если программа работает в режиме клиента;
- Включение/Отключение шифрования и дешифрования отправляемого и получаемого сообщения соответственно;
- Возможность ввода криптографического ключа;

- Возможность ввода сообщения;
- История сообщений и имен их отправителей;
- Запись времени полученных и отправленных сообщений.

3.4 Описание графического интерфейса

Разработанная программа-мессенджер обладает довольно примитивным графическим интерфейсом и состоит из одного окна, в котором расположены все необходимые элементы для организации защищенного соединения, а именно:

- Кнопки «Создать сервер»/«Закрыть сервер» в зависимости от того, запущен ли сервер;
- Кнопки «Подключиться»/«Отключиться» в зависимости от того, подключен ли пользователь к серверу;
- Поле для ввода имени пользователя;
- Поле для ввода IP-адреса сервера для подключения к нему;
- Поле для ввода порта;
- Кнопка включения/отключения шифрования;
- Поле для ввода криптографического ключа (по умолчанию установлен ключ «SuperKey»);
- Поле для ввода сообщения;
- Кнопка «Отправить» для отправки сообщения;
- Окно истории отправленных и полученных сообщений.

После создания сервера или подключения к серверу, поля для ввода IP-адреса и порта становятся недоступными для ввода.

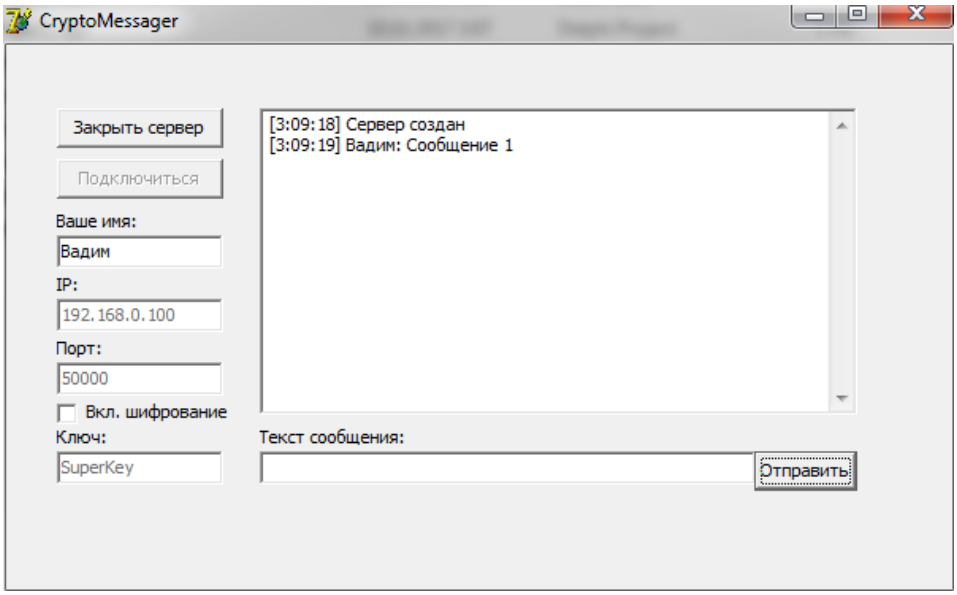

Рисунок 3.1 – Интерфейс программы

3.5 Принцип работы программы

Разработанная программа-мессенджер работает по следующему принципу. При подключении к серверу, программа отправляет запрос на адрес, указанный в поле ввода IP-адреса. Порт для подключения берется из соответствующего поля. Если сервер найден, то устанавливается соединение, а в истории сообщений добавляется соответствующая запись.

При нажатии кнопки «Отправить» программа считывает содержимое полей ввода сообщения и имени пользователя и, если включена опция шифрования, выполняет криптографическое преобразование этих данных с ключом, введенным в соответствующее поле. Далее, если программа работает в режиме клиента, то полученное зашифрованное сообщение отправляется на сервер, который, в свою очередь, пересылает его остальным участникам сеанса связи (клиентам). Если же программа запущена в режиме сервера, то сообщение сразу рассылается всем клиентам. Незашифрованное сообщение добавляется в историю сообщений вместе со временем отправки и именем отправителя.

Мессенджер постоянно находится в режиме «прослушки» порта, когда связь установлена, и, при получении сообщения, незамедлительно выполняется обратное криптографическое преобразование этого сообщения (если включена опция шифрования), после чего расшифрованное сообщение появляется в окне истории сообщений вместе со временем получения и именем отправителя. Для того чтобы сообщение расшифровывалось правильно, необходимо, чтобы ключи в полях отправителя и получателя совпадали.

3.6 Описание работы алгоритма шифрования

Перед передачей введенного пользователем сообщения, оно обрабатывается функцией шифрования, если включена соответствующая опция. Функция представляет собой ряд арифметических и логических операций над исходным текстом.

Сначала поступающий текст делится на блоки. Размер блока равен восьми символам. Количество блоков зависит от длины сообщения. Если длина сообщения не кратна размеру блока, то в конце сообщения добавляются случайные символы, чтобы шифрование прошло успешно. По окончанию операции лишние символы отсекаются.

Процесс шифрования начинается с первого блока. Для этого он разбивается на два подблока – «левый» и «правый». Левый подблок обрабатывается функцией F, а затем складывается логической операцией «Исключающее ИЛИ» (XOR) с правым подблоком. Далее получившийся результат возвращается в левую часть, а левый исходный подблок в правую. После этого операция повторяется. Количество таких операций (раундов) равняется шестидесяти.

Поскольку логические операции могут производиться только над числами, необходимо преобразовать исходный текст в числовой формат. Для этого функция преобразует каждый символ исходного текста в соответствующий ему код по таблице кодировки Unicode. Таким образом, логические операции производятся не над текстом, а над соответствующими ему кодами. После

выполнения всех операций над кодами, получившиеся значения преобразовываются в текст по этой же таблице. Поскольку значения изменились, то и символы на выходе получаются другими.

Для каждого раунда вычисляется отдельный (раундовый) ключ из основного. Для этого символы основного ключа заменяются на символы, стоящие по порядку дальше в таблице кодировки. Уровень смещения прямо пропорционально зависит от номера обрабатываемого блока.

Функция F складывает поступающий левый подблок с ключом раунда при помощи XOR. При этом ключ раунда делится на длину подблока и складываемая часть зависит от того, какой по счету блок исходного текста обрабатывается на данный момент. С каждым новым блоком складывается новая часть раундового ключа. Это происходит до тех пор, пока весь раундовый ключ не будет использован в функции, после чего он берется повторно.

При приеме зашифрованного сообщения выполняется процедура его дешифрования. Данная процедура полностью идентична процедуре шифрования, поскольку функции F и XOR являются обратимыми. В итоге мы имеем исходный открытый текст без каких либо искажений.

Блок-схема алгоритма шифрования изображена на рисунке 3.2.

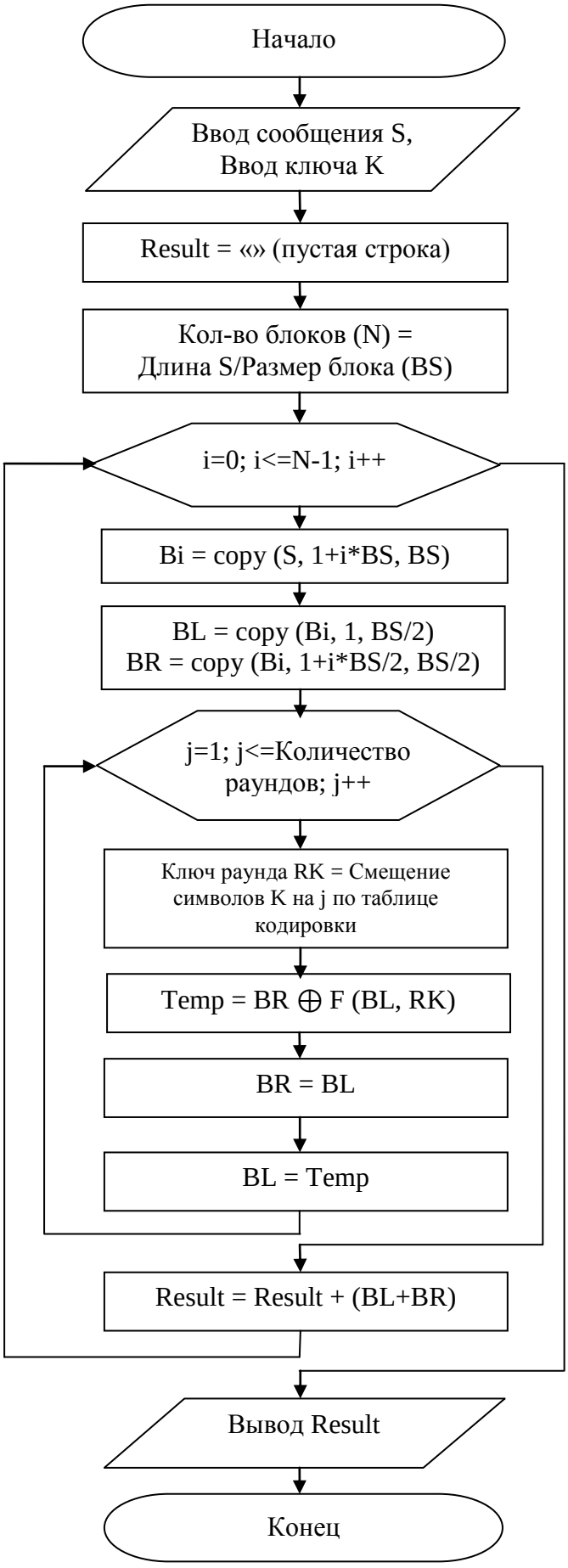

Рисунок 3.2 - Блок-схема алгоритма шифрования.

3.7 Рекомендации по использованию программы

Для организации надежного и быстрого соединения, при большом количестве участников сеанса связи, рекомендуется запускать сервер на достаточно мощной машине, имеющей стабильное подключение к сети Интернет. Это связано с тем, что сервер обрабатывает запросы каждого подключенного клиента и перенаправляет приходящие от них сообщения остальным участникам связи, а эти операции являются довольно требовательными к системным и сетевым ресурсам.

Также, при использовании функции шифрования сообщений, рекомендуется использовать длинные ключи (не менее восьми символов) и, по возможности, менять их при каждом новом сеансе связи. Желательно, чтобы ключ состоял из логически не связанных между собой символов, символов верхнего и нижнего регистров, а также, чтобы в качестве символов использовались цифры. Это позволит усложнить криптоанализ потенциальному злоумышленнику, а, соответственно, повысить уровень защиты соединения.

Выполнение указанных рекомендаций позволит организовать стабильное и защищенное соединение.

3.8 Экономическая целесообразность проекта

В проекте разрабатывается программное обеспечение для обмена сообщениями в сети Интернет. Поскольку на разработку было потрачено определенное количество временных, а также вычислительных ресурсов, имеет смысл провести экономический расчет затрат и возможную прибыль в случае продажи продукта.

3.8.1 Расчет себестоимости программы

Прежде всего, нужно рассчитать себестоимость разработанного продукта.

Для этого необходимо учесть:

- Амортизационные отчисления на полное восстановление технических средств и программного обеспечения;
- Оплату труда программиста;
- Доплаты и надбавки к заработной плате;
- Затраты электроэнергии, расходуемой техническими средствами;
- Накладные расходы;
- Единый социальный налог.

Расчет амортизационных отчислений.

При разработке мессенджера используются следующие технические средства:

- IBM совместимый компьютер - 20000 руб;

- Монитор LCD ViewSonic 24” - 7000 руб.

Примем норму амортизации на технические средства равной 20%.

Общая стоимость технических средств:

Цтс = 20000 + 7000 = 27000 р.

При разработке мессенджера используется следующее программное обеспечение:

- ОС Windows 7 64 Максимальная – 6000 р.;

- Borland Delphi 7 – 5000 р.

Общая стоимость программного обеспечения:

Цпо = 6000 + 5000 = 11000 р.

Общая стоимость технических средств и программного обеспечения:

Цо = Цтс + Цпо

Цо = 27000 + 11000=38000 р.

Годовые амортизационные отчисления на полное восстановление технических средств и программного обеспечения рассчитываются по формуле:

Ао = Цо · На, где На – норма амортизации.

Ао = 38000 · 0,2 = 7200 р.

Амортизационные отчисления за период создания программного продукта:

Ап = Ао · Кдн / Крг,

где Кдн = 20 дн. – количество отработанных дней;

Крг = 280 дн. – количество рабочих дней в году.

Ап = 7200 · 20 / 280 ≈ 514 р.

Расчет расходов на энергопотребление.

ПЭВМ, на которой была разработана программа, является потребителем электрической энергии сети переменного тока, напряжением 220 В. Согласно технической документации, суммарная мощность, потребляемая компьютером и монитором, составляет:

Мс= 300 Вт·ч.

Расход денежных средств, связанный с энергопотреблением технических средств можно найти по формуле:

Рэ = Кдн·Враб·Мс·Цэн,

где Кдн – период написания программы, дн.;

Враб – длительность рабочей смены, ч.;

Мс – мощность, потребляемая техническими средствами, кВт·ч;

Цэн – цена на электроэнергию по действующим тарифам, р/кВт·ч;

Отсюда:

Рэ ≈ 10·6·0,3·4 ≈ 72 р.

Расчет заработной платы программиста.

Исходя из фактически отработанного времени программиста, которое составило 10 рабочих шестичасовых дней, найдем количество фактически отработанного времени:

Тф = Кдн·Враб

Тф = 10·6 = 60 ч.

Принимая часовую заработную плату программиста в расчете 250 руб., получим основную заработную плату:

Зосн = Тф·Тч, где Тч – часовая тарифная ставка программиста.

Зосн = 60·250 = 15000 р.

Для определения общей суммы расходов на оплату труда необходимо учесть доплаты и надбавки. Принимаем удельный вид доплат и надбавок в размере 15 % от основной заработной платы:

Удоп = Зосн · 0,15

Удоп = 15000 · 0,15 = 2250 р.

Отсюда находим общие расходы на оплату труда:

Робщ = Зосн + Удоп

Робщ = 15000 + 2250 = 17250 р.

Далее определим единый социальный налог; в соответствии со ставкой он составляет 26% от расходов на оплату труда, что составит:

Осоц = Робщ · 0,26

Осоц = 17250 · 0,26 ≈ 4485 р.

Итак, для расчета себестоимости продукта необходимо суммировать все затраты на его создание:

С = Ап + Рэ + Робщ + Осоц

С = 514 + 72 + 17250 + 4485 = 22321 р.

3.8.2 Определение отпускной цены программы

Для нахождения отпускной цены необходимо учесть:

- прибыль от реализации 15 %;
- налог на добавленную стоимость 13 %.

Отсюда следует, что отпускная цена вычисляется по формуле:

Цп = С + П + НДС, где П – прибыль от реализации продукта
НДС – налог на добавленную стоимость

П = С · 0,15,

НДС = С· 0,13,

Цп = 22321 + 22321 · 0,15 + 22321· 0,13 ≈ 28571 р.

Таким образом, отпускная цена программы составляет примерно 28571 рублей.

Выводы по главе**.**

На основе приведенной блок схемы, в среде разработки Borland Delphi 7, была написана функция шифрования. Данная функция была внедрена в программу в той же среде. Таким образом был реализован мессенджер с возможностью шифрования сообщений. Полный листинг программы приведен в приложении 1.

Глава IV. Безопасность жизнедеятельности

В данной работе разрабатывается программа для обмена сообщениями между пользователями. Деятельность пользователей, работающих с подобного рода программами, неразрывно связана с персональным компьютером, и зачастую время работы является продолжительным. Поэтому следует учесть, что от правильной организации рабочего места, зависит здоровье человека, использующего данную программу. Также необходимо рассмотреть возможные угрозы, возникающие при работе в помещении с ПЭВМ.

4.1 Характеристика вредных факторов при работе с ПК

На организм человека – пользователя ПК – может комплексно воздействовать ряд опасных и вредных факторов. Наиболее значительными являются:

а) Возможность появления напряжения на металлических частях ПК, которое может привести к электротравме.

б) Несоответствие норм параметров микроклимата нормативным требованиям, повышенная температура из-за постоянного нагрева деталей ПК, пониженная влажность.

в) Различные излучения:

- электромагнитное излучение в низкочастотном, высокочастотном и сверхвысокочастотном диапазоне;
- рентгеновское излечение от ЭЛТ;
- ультрафиолетовое излучение;
- инфракрасное излучение;
- электростатическое поле.

г) Пониженный или повышенный уровень освещенности в помещениях, не соответствующие санитарным нормам визуальные параметры дисплея.

д) Повышенный уровень содержания в воздухе патогенной микрофлоры и химических веществ (окись углерода, озона, аммиак, окислы серы, азота, соли тяжелых металлов и органических соединений).

е) Психофизиологическая напряженность труда:

- монотонность работы;
- повышенное умственное напряжение из-за большого объема перерабатываемой и усваиваемой информации;
- повышенное нервно-эмоциональное напряжение;
- длительные статические нагрузки.

4.2 Организация рабочего места с ПК. Требования к помещениям с ПК

Помещения для эксплуатации ПК должны иметь естественное и искусственное освещение. Эксплуатация ПК в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке.

Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации.

Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток, и обеспечивать коэффициент естественной освещенности (КЕО) не ниже 1.2 в зонах с устойчивым снежным покровом и не ниже 1.5 на остальной территории.

Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Не допускается размещение мест пользователей ПЭВМ во всех образовательных и культурно-развлекательных учреждениях для детей и подростков в цокольных и подвальных помещениях.

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка – 0.7 – 0.8; для стен – 0.5 – 0.6; для пола – 0.3 – 0.5.

Полимерные материалы используемые для внутренней отделки интерьера помещений с ПЭВМ должны иметь наличие положительного санитарно-эпидемиологического заключения.

Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

4.2.1 Требования к размещению и оборудованию рабочего места с ПК

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами должно быть не менее 2 м, а расстояние между боковыми поверхностями мониторов - не менее 1.2 м.

В помещениях с источниками вредных производственных факторов рабочие места должны размещаться в изолированных кабинах с организованным воздухообменом. При выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рабочие места рекомендуется изолировать друг от друга перегородками высотой 1.5 – 2.0 м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600 - 700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей.

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

Конструкция рабочего стула (кресла) должна позволять менять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для снижения утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья.

Клавиатуру следует располагать на поверхности стола на расстоянии 100 - 300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

4.2.2 Требования к санитарно-гигиеническим параметрам рабочего места

В зависимости от категории трудовой деятельности и уровня нагрузки за рабочую смену при работе с ПЭВМ устанавливается суммарное время регламентированных перерывов, нормативные данные приведены в таблице 4.1.

Категория работы с ПЭВМ	Уровень нагрузки за рабочую смену при видах работ с ПЭВМ			Суммарное время регламентированных перерывов, мин.	
	группа А, количество знаков	группа Б, количество знаков	группа В, ч	при 8-часовой смене	при 12-часо вой смене
I	до 20 000	до 15 000	до 2	50	80
II	до 40 000	до 30 000	до 4	70	110
III	до 60 000	до 40 000	до 6	90	140

Таблица 4.1 - Суммарное время регламентированных перерывов в зависимости от продолжительности работы, вида и категории трудовой деятельности с ПЭВМ.

В помещениях, где работа на ПЭВМ является основной, должны обеспечиваться оптимальные параметры микроклимата, приведенные в таблице 4.2.

Период года	Категория работ	Температура воздуха С º, не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая – 1а	22-24	40-60	0,1
	Легкая – 1б	21-23	40-60	0,1
Теплый	Легкая – 1а	23-25	40-60	0,1
	Легкая – 1б	22-24	40-60	0,2

Таблица 4.2 - Оптимальные нормы микроклимата для помещений с ПЭВМ.

К категории 1а относятся работы, производимые сидя и не требующие физического напряжения, при которых расход энергии составляет до 120ккал/ч. К категории 1б относятся работы, производимые сидя, стоя или связанные с ходьбой и сопровождающиеся некоторым физическим напряжением, при которых расход энергии составляет от 120 до 150 ккал/ч.

Уровни электромагнитных излучений, считающихся безопасными для здоровья, приведены в таблице 4.3.

Наименование параметров		ВДУ
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

Таблица 4.3 - Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах.

В результате работы ПЭВМ генерируется весьма широкий спектр звуков (включая ультразвук), причем каждый компьютер отличается в этом смысле своей индивидуальностью. Корпус компьютера при этом является резонатором и привносит в общую картину шума низкочастотные составляющие. Нормативные параметры звука приведены в таблице 4.4.

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									Уровни звука в дБА
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
86 дБ	71 дБ	61 дБ	54 дБ	49 дБ	45 дБ	42 дБ	40 дБ	38 дБ	50

Таблица 4.4 - Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого ПЭВМ.

Искусственное освещение в помещениях с ПЭВМ должно осуществляться системой общего равномерного освещения. Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 - 500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

4.3 Причины и профилактика статического и зрительного утомления

Психофизиологические исследования показали, что при напряженной работе с компьютером утомление возникает в среднем через два часа после начала работы. Чтобы не допускать этого, после двух часов непрерывной работы

с дисплеем нужно делать перерывы. Во время перерывов необходимо проводить гимнастику для глаз.

Профилактика зрительного утомления

Перед началом работы установите регуляторы яркости и контрастности изображения в наименьшее положение, при котором можно комфортно считывать информацию. В дальнейшем (по мере наступления зрительного утомления) увеличивайте яркость (контрастность) для достижения оптимальных условий зрительного восприятия. Не смотрите подолгу пристально на экран. Глаза должны постоянно перемещаться по экрану, а не фиксироваться на какой-либо его части. Старайтесь периодически (через каждые 3-5 мин) переводить взгляд (на 3-5 с) с экрана монитора на самый дальний предмет в комнате или (что существенно лучше) на отдаленный объект за окном. При чтении с экрана старайтесь после каждой строки моргнуть, а после каждого большого абзаца - поднять глаза и посмотреть на 2-3 секунды вдаль. При вводе информации старайтесь не читать только что написанные слова. Во всех случаях, даже в момент интенсивной мыслительной работы, введите привычку моргать каждые 3-5 секунд. Каждые 2 часа выполняйте комплексы расслабляющих упражнений.

4.4 Пожарная безопасность

Пожарная безопасность подразумевает такое состояние объекта, при котором вероятность возникновения пожара минимальная, а при его возникновении обеспечивается защита людей и материальных ценностей. Пожарная безопасность обеспечивается системой организационных мер и технических средств по предотвращению пожара, т.е. пожарной профилактикой, а также системой мер, позволяющих быстро ликвидировать начавшийся пожар с наименьшими потерями, т.е. организацией пожаротушения.

Во всех производственных, административных, складских и вспомогательных помещениях на видных местах должны быть вывешены таблички с указанием номера телефона вызова пожарной охраны.

В каждой организации распорядительным документом должен быть установлен соответствующий пожарной опасности противопожарный режим, в том числе:

- определены и оборудованы места для курения;

- определен порядок обесточивания электрооборудования в случае пожара и по окончании рабочего дня;

Бытовые электроприборы в помещениях, в которых по окончании рабочего времени отсутствует дежурный персонал, должны быть обесточены, за исключением дежурного освещения, установок пожаротушения и противопожарного водоснабжения, пожарной и охранно-пожарной сигнализации. Другие электроустановки и электротехнические изделия могут оставаться под напряжением, если это обусловлено их функциональным назначением и/или предусмотрено требованиями инструкции по эксплуатации.

Необходимые действия при возникновении пожара:

- Сообщить в пожарную охрану.
- Оповестить всех окружающих коллег о пожаре.
- Если горение только началось, попробовать, используя пожарные краны, огнетушители, подручные средства, потушить огонь.
- Если потушить пожар не удаётся, покинуть опасную зону.
- По прибытии пожарных объяснить, что и где горит.

При невозможности покинуть здание (высокая температура, задымление) необходимо плотно закрыть дверь помещения, уплотнить тканью щели, вентиляционные отверстия, открыть окно и ждать пожарных. Важно помнить, что при задымлении воздух более чист над полом. Это может пригодиться при эвакуации и ожидании помощи.

Выводы к главе.

Знание санитарно-гигиенических параметров и их соблюдение в пределах норм, правильный режим труда и отдыха, выполнение требований эргономики, пожарной и электробезопасности позволяют снизить или совсем исключить

влияние вредных и опасных факторов на работников, и повысить эффективность труда.

Заключение

Исследовав основы построения телекоммуникационных систем, возможные угрозы и методы защиты от них, было принято решение разработки защищенного канала связи, обеспечивающего безопасную передачу сообщений между удаленными пользователями. Для достижения этой цели была разработана программа-мессенджер, позволяющая передавать сообщения по сети Интернет в зашифрованном виде.

В первую очередь, при помощи среды разработки Borland Delphi 7, была написана программа для отправки и принятия сообщений между удаленными пользователями. После этого был разработан и внедрен алгоритм шифрования. Основой алгоритма послужила классическая сеть Фейстеля, поскольку данный метод построения шифров является довольно простым в плане программной реализации, но в то же время достаточно криптостойским. Возможность шифрования сообщений позволяет обеспечить их конфиденциальность.

Таким образом, был разработан защищенный канал связи, а значит, поставленная в работе цель была достигнута.

Список использованных источников

1. Нестеров С. Информационная безопасность: СПб, 2009.
2. Аскеров Т.М. Защита информации и информационная безопасность: Учебное пособие: 2001.
3. Шнайер Б. Прикладная криптография: 1994.
4. Таненбаум Э.С. Компьютерные сети: 2005.
5. Новиков Ю.В. Основы локальных сетей - курс лекций
6. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: 2016.
7. Филимонов А. Построение мультисервисных сетей Ethernet: 2007.
8. Статья «Сетевая модель OSI». [Электронный ресурс]

   (url: www.itandlife.ru/technology/computer-networks/setevaya-model-osi-open-system-interconnection/)
9. Шринивас В. Качество обслуживания в сетях IP, 2003.
10. Петров А.А. Компьютерная безопасность. Криптографические методы защиты: СПб, 2000.
11. Щербаков Л.Ю. Домашен А.В. Прикладная криптография.
12. Нечаев В. И. Элементы криптографии.
13. Н.Н.Токарева. Симметричная криптография. Краткий курс: 2012.
14. Баричев С.Г., Гончаров В.В. Основы современной криптографии: 2011.
15. Столингс В. Криптография и защита сетей: 2001.
16. Культин Н.Б. Основы программирования Delphi: 2015.
17. Фленов М. Библия Delphi: 2008.

18. Статья «Архитектура сетей». [Электронный ресурс] (url: www.blogsisadmina.ru/seti/odnorangovye-i-ierarxicheskie-seti-v-chem-otlichie.html)
19. Wikipedia, свободная интернет-энциклопедия. [Электронный ресурс] (url: www.ru.wikipedia.org)
20. СНиП 2.09.04 – 87 Административные и бытовые здания и сооружения
21. СНиП 2.01.02 – 85 Противопожарные нормы
22. СНиП 2.05.02 – 89 Общие требования к освещению
23. ГОСТ 12.1.005 – 88 ССБТ. Общие и санитарно – гигиенические требования к воздуху рабочей зоны.
24. СНиП 2.04.05 – 91 Отопление, вентиляция и кондиционирование.
25. ГОСТ 12.1.003 – 83 ССБТ. Шум. Общие требования безопасности.
26. ГР 2411 – 81
27. ГОСТ 12.1.029 – 80 ССБТ. Средства и методы защиты от шума.
28. ГОСТ 12.2.007 – 75 ССБТ.
29. ГОСТ 25.851 – 85 ССБТ.
30. ГНАТО 0.00. – 1.21.98
31. СНиП II-4-79. Естественное и искусственное освещение.

Приложение 1. Листинг программы

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ScktComp;

type
  TForm1 = class(TForm)
    ClientSocket: TClientSocket;
    ServerSocket: TServerSocket;
    portedit: TEdit;
    lblport: TLabel;
    IP: TLabel;
    Hostedit: TEdit;
    Nik: TLabel;
    NikEdit: TEdit;
    txtlbl: TLabel;
    TextEdit: TEdit;
    ChatMemo: TMemo;
    SendBtn: TButton;
    ServerBtn: TButton;
    ClientBtn: TButton;
    KeyEdit: TEdit;
    Keylbl: TLabel;
    cryptchk: TCheckBox;
    cryptlbl: TLabel;

    procedure FormCreate(Sender: TObject);
    procedure ServerBtnClick(Sender: TObject);
    procedure ServerSocketClientDisconnect(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ServerSocketClientRead(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure SendBtnClick(Sender: TObject);
    procedure ClientBtnClick(Sender: TObject);
    procedure ClientSocketRead(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocketConnect(Sender: TObject;
    Socket: TCustomWinSocket);
    procedure ClientSocketDisconnect(Sender: TObject;
```

```
Socket: TCustomWinSocket);
procedure ServerSocketAccept(Sender: TObject;
Socket: TCustomWinSocket);
procedure ClientSocketConnecting(Sender: TObject;
Socket: TCustomWinSocket);
procedure ServerSocketClientError(Sender: TObject;
Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
var ErrorCode: Integer);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure cryptchkClick(Sender: TObject);
procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
ErrorEvent: TErrorEvent; var ErrorCode: Integer);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Users: TStringList;

implementation

{$R *.dfm}


//ЗАПОЛНЕНИЕ ПОЛЕЙ
procedure TForm1.FormCreate(Sender: TObject);
begin
PortEdit.Text:='50000';
HostEdit.Text:='192.168.0.100';

NikEdit.Clear;
TextEdit.Clear;
ChatMemo.Lines.Clear;
Users:=TStringList.Create;
KeyEdit.Enabled := False;
end;

procedure TForm1.cryptchkClick(Sender: TObject);
begin
```

```
If cryptchk.Checked=enabled then KeyEdit.Enabled:=True else
KeyEdit.Enabled:=False ;
end;


//ФУНКЦИЯ ШИФРОВАНИЯ И ДЕШИФРОВАНИЯ
function Crypt(Key, Text: string):string;
var BlockSize, Rounds, LengthOfText, check, b, b1, i, j, n, test : Integer;
var RoundKey, block, blockL, blockR, newblock, F, temp : string;

begin
Rounds := 60;  //Количество раундов
BlockSize := 8; //Размер блока
newblock := '';
LengthOfText := Length(Text);
n := Length(Key) div (BlockSize div 2);
b1:=0;


 if (LengthOfText <> BlockSize) then    //Проверка на заполнение блока
 begin
 for check:=1 to (BlockSize-(LengthOfText mod BlockSize)) do
 begin
 Text := Text + chr(Random(500));       //Добавление случайных символов
 end;
 end;

 for b:=0 to (Length(Text) div BlockSize)-1 do
 begin
 block:= Copy (Text, 1+b*BlockSize, BlockSize);
 blockL:= Copy (block, 1, (BlockSize div 2));
 blockR:= Copy (block, 1+(BlockSize div 2), (BlockSize div 2));

  for i:=1 to Rounds do
  begin
     temp := '';
     F := '';
     RoundKey := '';

     for j:=1 to Length(Key) do            //Вычисление раундового ключа
     begin
     RoundKey := RoundKey + chr(ord(Key[j])+ 2*i);
     end;
```

```
        if b1 < n then                      //Функция F
        begin
        for j:=1 to Length(blockL) do
        begin
         //test := ord(blockL[j]);
         //ShowMessage(IntToStr(test));
        F := F + chr((ord(blockL[j]) xor ord(RoundKey[4*b1+j])));
        end;
        if i = Rounds then b1:=b1+1;
        end
        else
        begin
        for j:=1 to Length(blockL) do
        begin
        F := F + chr((ord(blockL[j]) xor ord(RoundKey[j])));
        end;
        if i = Rounds then b1:=0;
        end;

        for j:=1 to Length(blockL) do
        begin
          temp := temp + chr((ord (F[j]) xor ord(blockR[j])));    //XOR с правым
подблоком
        end;

        blockR:=blockL; //Перемешивание подблоков
        blockL:=temp;

    end;

  newblock := newblock + (blockL+blockR);  //Построение защифрованной строки
  end;
Result := Copy (newblock, 0, LengthOfText);  //Зашифрованная строка без лишних
символов
end;


//СЕРВЕР
procedure TForm1.ServerBtnClick(Sender: TObject);
begin
If ServerBtn.Tag=0 then
Begin
ClientBtn.Enabled:=False;
HostEdit.Enabled:=False;
```

```
PortEdit.Enabled:=False;

ServerSocket.Port:=StrToInt(PortEdit.Text);
ServerSocket.Active:=True;
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сервер создан');
ServerBtn.Tag:=1;
ServerBtn.Caption:='Закрыть сервер';
end
else
Begin
ClientBtn.Enabled:=True;
HostEdit.Enabled:=True;
PortEdit.Enabled:=True;

ServerSocket.Active:=False;
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сервер закрыт.');
ServerBtn.Tag:=0;
ServerBtn.Caption:='Создать сервер';
end;
end;

procedure TForm1.ServerSocketClientRead(Sender: TObject;
Socket: TCustomWinSocket);
var
X: Integer;
Msg1: string;
begin
Msg1:=Socket.ReceiveText;

 for X := 0 to ServerSocket.Socket.ActiveConnections-1 do
   ServerSocket.Socket.Connections[X].SendText(Msg1);

If cryptchk.Checked=enabled then Msg1 := Crypt(KeyEdit.Text, Msg1);
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] '+Msg1);
end;

procedure TForm1.ServerSocketClientDisconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Клиент отключился.');
end;

procedure TForm1.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
```

```
begin
 ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Подключился клиент.');
end;

procedure TForm1.ServerSocketClientError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Ошибка.');
end;


//КЛИЕНТ
procedure TForm1.ClientBtnClick(Sender: TObject);
begin
If ClientBtn.Tag=0 then
Begin

ServerBtn.Enabled:=False;
HostEdit.Enabled:=False;
PortEdit.Enabled:=False;

ClientSocket.Port:=StrToInt(PortEdit.Text);
ClientSocket.Host:=HostEdit.Text;
ClientSocket.Address:=HostEdit.Text;
ClientSocket.Active:=True;

ClientBtn.Tag:=1;
ClientBtn.Caption:='Отключиться';
end
else
Begin
ServerBtn.Enabled:=True;
HostEdit.Enabled:=True;
PortEdit.Enabled:=True;

ClientSocket.Active:=False;
ClientBtn.Tag:=0;
ClientBtn.Caption:='Подключиться';
end;
end;

procedure TForm1.ClientSocketRead(Sender: TObject;
Socket: TCustomWinSocket);
```

```
var Msg: string;
begin
If cryptchk.Checked=enabled then Msg := Crypt(KeyEdit.Text, Socket.ReceiveText)
else Msg := Socket.ReceiveText;

ChatMemo.Lines.Add('['+TimeToStr(Time)+'] '+Msg);
end;

procedure TForm1.ClientSocketConnecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Подключение к серверу.');
end;

procedure TForm1.ClientSocketConnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Успешно.');
end;

procedure TForm1.ClientSocketDisconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
ChatMemo.Lines.Add('['+TimeToStr(Time)+'] Сеанс завершен.');
end;

procedure TForm1.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
ShowMessage('Сервер не найден!');
end;


//ОТПРАВКА СООБЩЕНИЯ
procedure TForm1.SendBtnClick(Sender: TObject);
var
X: Integer;
EncryptedText: string;
begin
  if Length(KeyEdit.Text) < 4 then
  begin
    ShowMessage('Ключ должен быть не менее 4х символов');
    Exit;
```

```
  end;
  if Length(TextEdit.Text) = 0 then
  begin
    ShowMessage('Введите сообщение');
    Exit;
  end;

  If cryptchk.Checked=enabled then
  begin
    EncryptedText := Crypt(KeyEdit.Text, (NikEdit.Text+': '+TextEdit.Text));
    end
    else
    begin
    EncryptedText := NikEdit.Text+': '+TextEdit.Text;
    end;

  If ServerSocket.Active=True then
  begin
  for X := 0 to ServerSocket.Socket.ActiveConnections-1 do
      ServerSocket.Socket.Connections[X].SendText(EncryptedText);
  ChatMemo.Lines.Add('['+TimeToStr(Time)+'] '+NikEdit.Text+': '+TextEdit.Text+'');
  end
  else
  begin
  ClientSocket.Socket.SendText(EncryptedText);
  end;

TextEdit.Text:='';
end;


procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
ClientSocket.Active:=False;
ServerSocket.Active:=False;
end;

end.
```