

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 10.05.02 –

Специализация -

Информационная безопасность
телекоммуникационных систем
Разработка защищенных
телекоммуникационных систем

ДИПЛОМНЫЙ ПРОЕКТ

на тему

МОДЕЛЬ ПОДСИСТЕМЫ ДИСТАНЦИОННОГО МОБИЛЬНОГО ДОСТУПА
АБОНЕНТОВ К ЗАЩИЩЕННОМУ WEB-РЕСУРСУ

Студент

Ручкин Дмитрий Владимирович

Руководитель начальник НИС АО «ПНИЭИ»

Клейменов А.А.

Нормоконтролёр

Фатеев А.Г.

Рецензент ведущий инженер АО «ПНИЭИ»

Печилейский С.В.

Работа допущена к защите (протокол заседания кафедры от 23.01.18 № 9)

Заведующий кафедрой

С.Л. Зефилов

Работа защищена с отметкой отлично (протокол заседания ГЭК
от 24.01.18 № 9)

Секретарь ГЭК

А.Г. Фатеев

Проверка ВКР - Ю.Г. Галов

Подпись С.В. Печилейского
Инженер 1 кат. ОК



ОГЛАВЛЕНИЕ

Введение.....	4
1 Технология VPN.....	6
1.1 Назначение VPN-сетей	6
1.2 Способы построения VPN-сетей	6
1.3 Обзор существующих программных средств, реализующих технологию VPN для мобильных устройств	9
2 Разработка модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу	12
2.1 Общие сведения о разработанной модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу	12
2.2 Разработка WEB-ресурса.....	17
2.3 Разработка специального программного обеспечения серверной части модели	18
2.4 Разработка специального программного обеспечения клиентской части модели	39
2.5 Тестирование разработанного специального программного обеспечения серверной и клиентской части модели	50
3 Руководство пользователя разработанной модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу	57
4 Анализ и обобщение результатов сравнительных исследований использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу с использованием отечественного криптоалгоритма.....	61
5 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	64
Заключение	68
Список используемых источников.....	69

Приложение А Текст программы WEB-ресурса.....	70
Приложение Б Формат сетевого пакета с зашифрованными данными.....	74
Приложение В Текст программы СПО-С	76
Приложение Г Текст программы СПО-К	116
Приложение Д Протокол проверки ВКР в системе «Антиплагиат»	141

ВВЕДЕНИЕ

В настоящее время широко распространен сетевой обмен данными. Оперативный обмен информацией – один из основных инструментов работы успешной организации. От быстроты и своевременности поступления информации зачастую зависит успех. Однако вопрос защищенности такого обмена при сохранении удобства информационного обмена между участниками, расположенными на значительном удалении друг от друга, открыт.

Одним из вариантов обеспечения такой передачи данных является технология построения VPN-сетей. Эта технология, позволяет выполнить сетевое соединение поверх другой сети, объединить удаленных пользователей в единую сеть и осуществляет криптографическую обработку передаваемых данных.

Распространенным вариантом применения технологии VPN является организация дистанционного мобильного доступа к WEB-ресурсам, но большинство программных обеспечений, реализующих данный вариант применения технологии VPN, разработано компаниями иностранных государств и работают с использованием иностранного алгоритма шифрования. В связи с чем, актуален вопрос создания собственной реализации технологии VPN, обеспечивающей дистанционный мобильный доступ к защищенному WEB-ресурсу.

Целью дипломного проекта является разработка модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу.

Разрабатываемая модель основана на VPN-сети, состоящей из удаленных мобильных устройств с установленным клиентским приложением и специального сервера, выполняющего функции приема, передачи и криптографической обработки сетевых данных.

Модель может использоваться как в образовательных целях, так и для проведения дальнейших исследований и разработок средств, методов организации дистанционного мобильного доступа к защищенным

WEB-ресурсам с использованием открытой незащищенной сети связи Интернет.

Первая глава дипломного проекта посвящена обзору технологии построения VPN-сетей.

Вторая глава посвящена разработке модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу. Подробно показаны алгоритм работы, реализация и тестирование специального программного обеспечения клиентской и серверной частей модели.

В третьей главе приведено руководство пользователя разработанной модели с примерами осуществления дистанционного мобильного доступа к защищенному WEB-ресурсу.

Четвертая глава посвящена проведению сравнительных исследований использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу с использованием отечественного криптоалгоритма.

1 Технология VPN

1.1 Назначение VPN-сетей

VPN (Virtual Private Network) - обобщённое название технологий, позволяющих обеспечить одно или несколько сетевых соединений поверх другой сети [1].

Данная технология может использоваться для:

- построения защищенной частной (корпоративной) сети, путем объединения нескольких разрозненных локальных сетей;
- подключения клиентов с мобильными устройствами (смартфоны и ноутбуки), а также клиентов с персональными компьютерами к корпоративным сетям;
- обеспечения анонимности пользователя в сети Интернет.

VPN-сети обеспечивают защищенный обмен данными между клиентами, вне зависимости от того, где каждый из них находится. Данное свойство VPN-сетей обеспечивается за счет шифрования передаваемых данных и передачи их по общедоступным каналам сети Интернет.

1.2 Способы построения VPN-сетей

На рисунке 1 показана типовая структура VPN-сети, обеспечивающая дистанционный мобильный доступ к ресурсам частной сети.

VPN-сети могут состоять из нескольких защищенных локальных сетей и мобильных устройств, находящихся за пределами данных сетей. Защищенные локальные сети могут иметь в своем составе персональные компьютеры с установленным клиентским приложением VPN, WEB-сервер, SMTP-сервер (почтовый сервер) и т.д.

Показанная типовая структура VPN-сети обеспечивает реализацию несколько видов подключения. На рисунках со 2 по 4 показана структура VPN-сети для подключений типа «клиент - клиент», «клиент – сеть», «сеть – сеть».

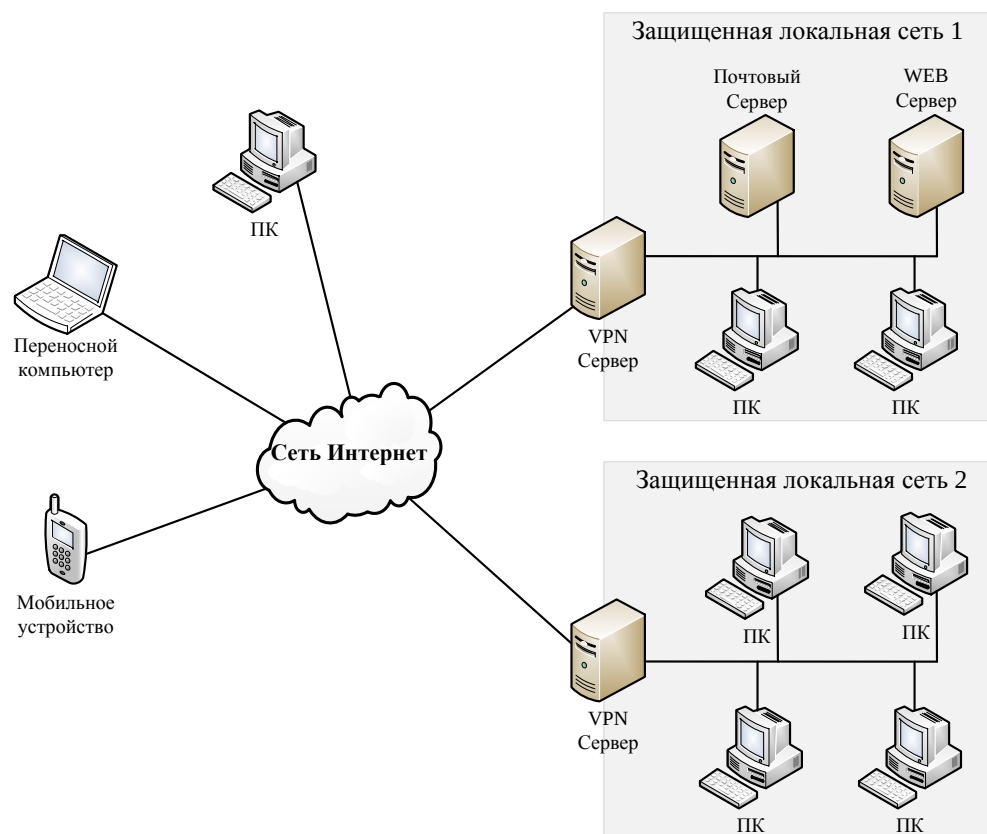


Рисунок 1 - Типовая структура VPN-сети для дистанционного мобильного доступа к ресурсам частной сети



Рисунок 2 – Подключение типа «клиент - клиент»

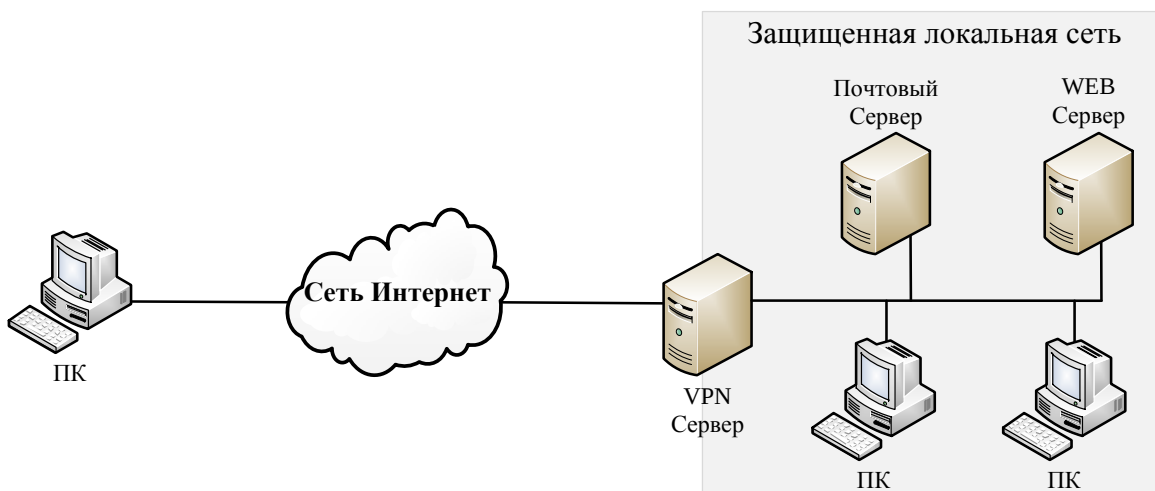


Рисунок 3 – Подключение типа «клиент - сеть»

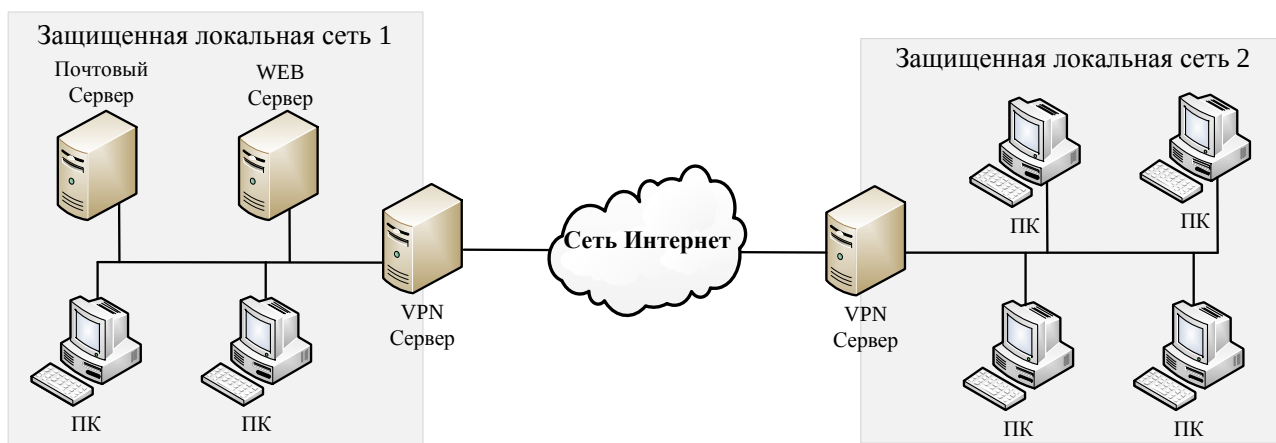


Рисунок 4 – Подключение типа «сеть - сеть»

Подключение типа «клиент – клиент» позволяет организовать защищенный обмен данными между двумя клиентами. К достоинствам такого подключения можно отнести простоту реализации. К недостаткам можно отнести отсутствие сервера распределения ключей (данный недостаток можно устранить используя специальные протоколы поддерживающие обмен ключами) и ограниченность функционала (у клиентов существует защищенный обмен данными только между собой).

Подключение типа «клиент – сеть» в настоящее время широко распространено. Данное подключение можно использовать для подключения клиентов находящихся за пределами корпоративной сети организации. Это очень удобно, в случае если сотрудники организации не размещены в одной рабочей области (здание или помещение) и существует необходимость обмена коммерческой информации. Также данное подключение широко используется для сохранения анонимности своих действий в сети интернет.

Подключение типа «сеть – сеть» подходит для обеспечения защищенного обмена данными между филиалами организаций.

Для построения VPN используются следующие протоколы:

- PPTP - туннельный протокол типа точка-точка, позволяющий компьютеру устанавливать защищённое соединение с сервером за счёт создания специального туннеля в стандартной, незащищённой сети. Протокол

PPTP позволяет инкапсулировать пакеты PPP в пакеты протокола Internet Protocol (IP) и передавать их по сетям IP;

- L2TP - туннельный протокол канального уровня. Объединяет протокол L2F (Layer 2 Forwarding), разработанный компанией Cisco, и протокол PPTP корпорации Microsoft. Позволяет организовывать VPN с заданными приоритетами доступа, однако не содержит в себе средств для защиты данных и механизмов аутентификации;

- IPSec - набор протоколов, касающихся вопросов обеспечения защиты данных при транспортировке IP-пакетов. IPSec также включает в себя протоколы для защищённого обмена ключами в сети Интернет. Протоколы IPSec работают на сетевом уровне;

- SSL (Secure Socket Layer) - протокол защищенных сокетов, обеспечивающий безопасную передачу данных по сети Интернет. При его использовании создается защищенное соединение между клиентом и сервером. Протокол использует защиту данных с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и безопасных хэш-функций. Использует RC4, MD5, RSA и другие алгоритмы защиты данных [2].

1.3 Обзор существующих программных средств, реализующих технологию VPN для мобильных устройств

Перечень программных средств, реализующих технологию VPN для мобильных устройств, с их характеристиками приведен в таблице 1.

Большинство программных средств, реализующих технологию VPN для мобильных устройств, разработаны иностранными компаниями и используют иностранные алгоритмы шифрования. Сетевые параметры, такие как скорость передачи данных и предоставляемый объем трафика зависят от типа лицензии. Так же некоторые VPN приложения сохраняют информацию о подключениях пользователей. Компании разработчики могут сотрудничать с правительством иностранных государств и передавать сведения о пользователях.

Данный обзор существующих программных средств, реализующих технологию VPN для мобильных устройств, показал актуальность создания отечественного программного средства работающего на отечественном алгоритме шифрования с открытым исходным кодом и реализующего дистанционный мобильный доступ.

Таблица 1 - Перечень программных средств, реализующих технологию VPN для мобильных устройств

Наименование	ОС	Стандарт шифрования	Скорость передачи данных	Лицензия	Лимит на передачу данных в месяц	Разработчик
Tunnelbear VPN	MacOS, Windows, Android, iOS	OpenVPN IPSec/IKEVv2	пинг – 232 мс вх – 3,52 Мбит/с исх – 13,25 Мбит/с	Бесплатная Платная	1,5 ГБ Неограничено	TunnelBear Inc.
ZenMate Security & Privacy VPN	MacOS, Windows, Android, iOS	IPSec	пинг – 280 мс вх – 14,41 Мбит/с исх – 14,56 Мбит/с	Бесплатная (ограничение по скорости) Платная	Неограничено	ZenMate
Hideman VPN	MacOS, Windows, Android, iOS	OpenVPN PPTP	пинг – 76 мс вх – 10,94 Мбит/с исх – 10,08 Мбит/с	Бесплатная (ограничение по времени) Платная	Неограничено	HIDEMAN LTD.
Rocket VPN	MacOS, Windows, Android, iOS	-	пинг – 245 мс вх – 2,95 Мбит/с исх – 4,56 Мбит/с	Бесплатная Платная	0,5 ГБ Неограничено	Liquidum Limited
Hide Me	MacOS, Windows, Android, iOS	OpenVPN L2TP	пинг – 97 мс вх – 3,86 Мбит/с исх – 5,57 Мбит/с	Бесплатная Платная	2 ГБ Неограничено	inCloak Network Ltd.
PrivateTunnel	MacOS, Windows, Android, iOS	OpenVPN	пинг – 126 мс вх – 5,86 Мбит/с исх – 7,57 Мбит/с	Бесплатная Платная	200 МБ Неограничено	OpenVPN Inc.
Hotspot Shield	MacOS, Windows, Android, iOS	IPSec	пинг – 159 мс вх – 1,94 Мбит/с исх – 3,57 Мбит/с	Бесплатная (ограниченное число серверов) Платная	Неограничено	AnchorFree

2 Разработка модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу

2.1 Общие сведения о разработанной модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу

В соответствии с техническими требованиями, разработанная модель подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу (далее – Модель) состоит из клиентской и серверной частей, обеспечивает криптографическую обработку сетевого трафика с использованием отечественного алгоритма и дистанционный мобильный доступ абонентов к защищенному WEB-ресурсу с использованием технологий Wi-Fi, 3G (UMTS), 4G (LTE). Разработанная модель позволяет провести сравнительные исследования использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) при осуществлении дистанционного мобильного доступа абонентов к защищенным WEB-ресурсам.

Модель может применяться для проведения исследований и разработок средств, методов организации дистанционного мобильного доступа к защищенным WEB-ресурсам с использованием открытой незащищенной сети связи Интернет.

Клиентская часть разработанной Модели реализована на базе смартфона общего пользования под управлением операционной системы (ОС) Android с разработанным специальным программным обеспечением (далее – СПО-К). СПО-К было установлено на два смартфона под управлением ОС Android 7.0 и Android 8.0, что удовлетворяет техническим требованиям к клиентской части. Разработка и реализация алгоритма работы СПО-К приведены в параграфе 2.4.

Серверная часть разработанной Модели реализована на базе ПЭВМ под управлением ОС Windows 7 с разработанным специальным программным обеспечением (далее – СПО-С), что удовлетворяет техническим требованиям к серверной части. Разработка и реализация алгоритма работы СПО-С приведены в параграфе 2.3.

На рисунке 5 показана структурная схема разработанной Модели. Данная модель основана на отключении типа «клиент - сеть».

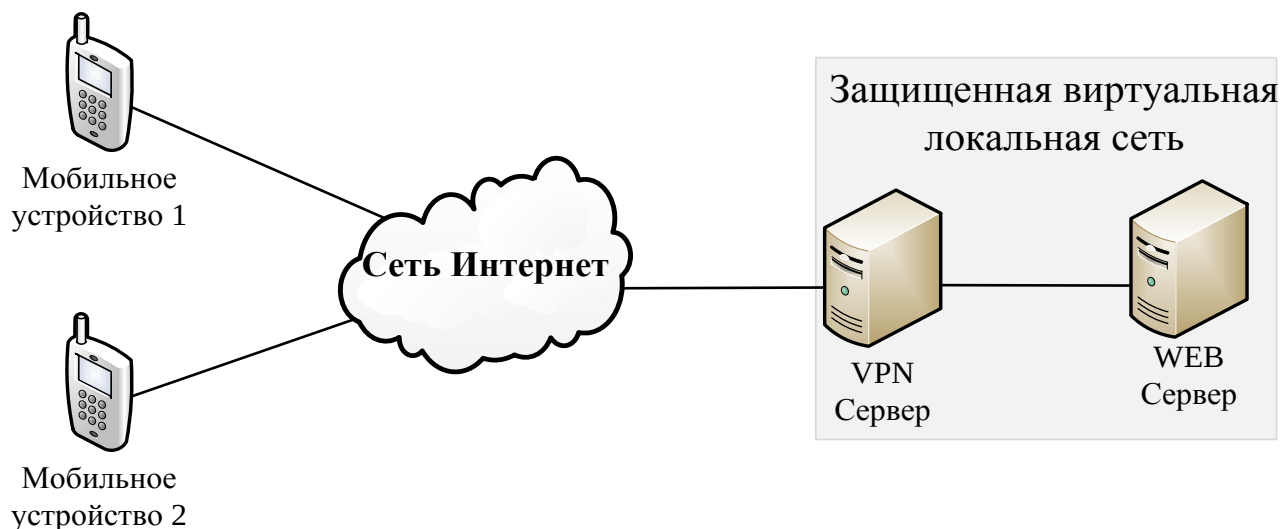


Рисунок 5 - Модель подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу

Данная структурная схема Модели состоит из следующих компонентов:

- два мобильных устройства, с которых осуществляется дистанционный доступ к защищенному WEB-ресурсу, с установленным СПО-К (далее – Клиент(ы));
- специальный сервер с установленным СПО-С (далее– VPN-сервер);
- WEB-сервер с разработанным WEB-ресурсом (далее – WEB-сервер);
- сеть интернет, используемая в качестве канала передачи данных.

В качестве WEB-сервера используется виртуальная машина под управлением ОС Windows 7. На WEB-сервере работает WEB-ресурс на базе ApacheTomcat 8.5, что удовлетворяет техническим требованиям к WEB-ресурсу. Разработка WEB-ресурса показана в параграфе 2.2.

WEB-сервер является гостевой виртуальной машиной и запускается на физической машине VPN-сервер.

Для разграничения локальной сети, в которой расположен WEB-сервер, от сети интернет (внешней сети), на VPN-сервере были использованы два сетевых адаптера:

– первый сетевой адаптер является настоящим и имеет белый IP-адрес для нахождения его в сети Интернет. Данный сетевой адаптер получает запросы от Клиентов и передает их на виртуальный адаптер;

– второй сетевой адаптер является виртуальным и имеет IP-адрес внутренней локальной сети для взаимодействия с WEB-сервером.

На рисунке 6 показаны настройки сетевых адаптеров соответственно для настоящего и виртуального адаптера VPN-сервера и для WEB-сервера.

Виртуальный сетевой адаптер VPN-сервера

IP-адрес: 192.168.56.1

Маска подсети: 255.255.255.0

Настоящий сетевой адаптер VPN-сервера

IP-адрес: 33.175.182.104

Маска подсети: 255.255.255.224

Сетевой адаптер WEB-сервера

IP-адрес: 192.168.56.2

Маска подсети: 255.255.255.0

Основной шлюз: 192.168.56.1

Рисунок 6 – Настройки сетевых адаптеров

Взаимодействие Клиента с WEB-ресурсом осуществляется в несколько этапов:

а) аутентификация Клиента. Клиент отправляет на VPN-сервер аутентифицирующие данные (пароль), VPN-сервер обрабатывает их и отправляет Клиенту результат. При успешном прохождении процедуры аутентификации начинается этап обмена данными;

б) обмен данными. Клиент посылает на VPN-сервер зашифрованный запрос. VPN-сервер расшифровывает его и пересылает на WEB-сервер. WEB-сервер, обработав полученный запрос, отправляет ответ на VPN-сервер. VPN-сервер зашифровывает ответ WEB-сервера и отправляет Клиенту;

в) завершение обмена данными. Завершение обмена данными осуществляется по истечении временного интервала, в течение которого клиент перестал отправлять данные.

На рисунке 7 показана диаграмма последовательности информационного взаимодействия клиентской и серверной частей (Клиент и VPN-сервер) Модели.

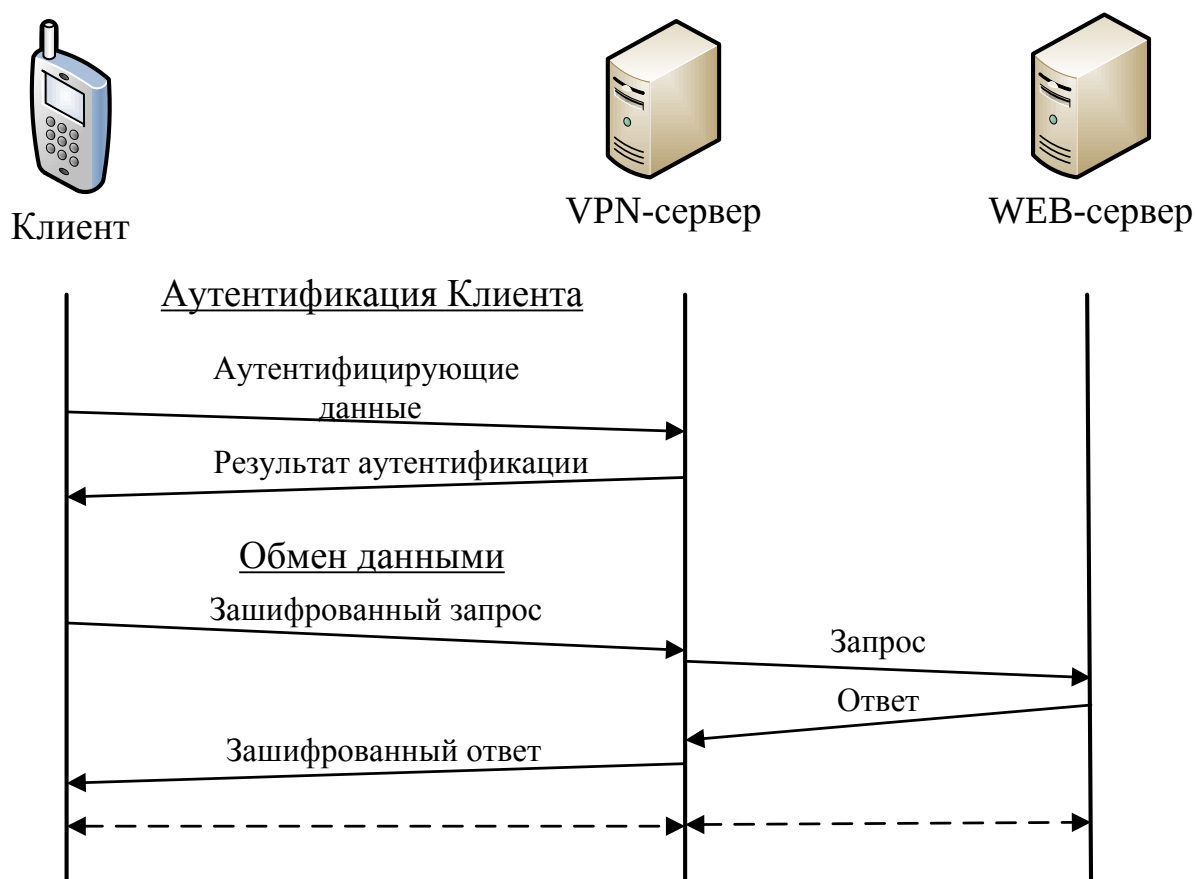


Рисунок 7 - Диаграмма последовательности информационного взаимодействия клиентской и серверной частей модели

На рисунке 8 показана структура Ethernet-пакета, используемая в информационном обмене Клиента и VPN-сервера.



Рисунок 8 - Ethernet-пакет

Открытые данные содержат служебную информацию. Зашифрованные данные содержат IMEI код Клиента и IP-пакет, получаемый Клиентом от WEB-браузера при обращении к WEB-ресурсу.

На рисунке 9 показана структура зашифрованных данных.

Зашифрованные данные



Рисунок 9 - Структура зашифрованных данных

VPN-сервер после расшифрования IP-пакета заменяет в нем IP-адрес источника на свой IP-адрес. В заголовке TCP-пакета порт отправителя заменяет на свой порт. Измененный IP-пакет отправляет на WEB-сервер.

При отправке ответа WEB-ресурса Клиенту, VPN-сервер заменяет в IP-пакете, полученном от WEB-сервера, IP-адрес назначения на IP-адрес Клиента. В заголовке TCP-пакета порт назначения заменяет на соответствующий порт Клиента. Измененный IP-пакет зашифровывает и отправляет Клиенту.

2.2 Разработка WEB-ресурса

WEB-Ресурс был написан на языке программирования Java. На данном языке было реализовано четыре сервлета:

- а) сервлет «Small» - реализует WEB-страницу, отображающую строку «Hello, world!» и счетчик обращения к ресурсу;
- б) сервлет «Big» - реализует WEB-страницу, отображающую строку «Hello, world!», счетчик обращения к ресурсу и 5 Кб символов (20 блоков символов с кодами от 0 до 255);
- в) сервлет «DownloadFile1Mb» - реализует WEB-страницу, при переходе на которую, начинается загрузка файла «File1Mb.txt» размером 1 Мб;
- г) сервлет «DownloadFile5Mb» - реализует WEB-страницу, при переходе на которую, начинается загрузка файла «File5Mb.txt» размером 5 Мб.

Текст программ, реализующих сервлеты, приведен в приложении А.

Для запуска WEB-ресурса на WEB-сервере был использован контейнер сервлетов «ApacheTomcat 8.5» (далее – Tomcat). Tomcat был настроен на обработку запросов приходящих на порт 8081.

На примере сервлета «Small» на рисунке 10 показана структура и содержимое каталога «webapps». Данный каталог расположен в корневом каталоге Tomcat и используется Tomcat для запуска пользовательских сервлетов.

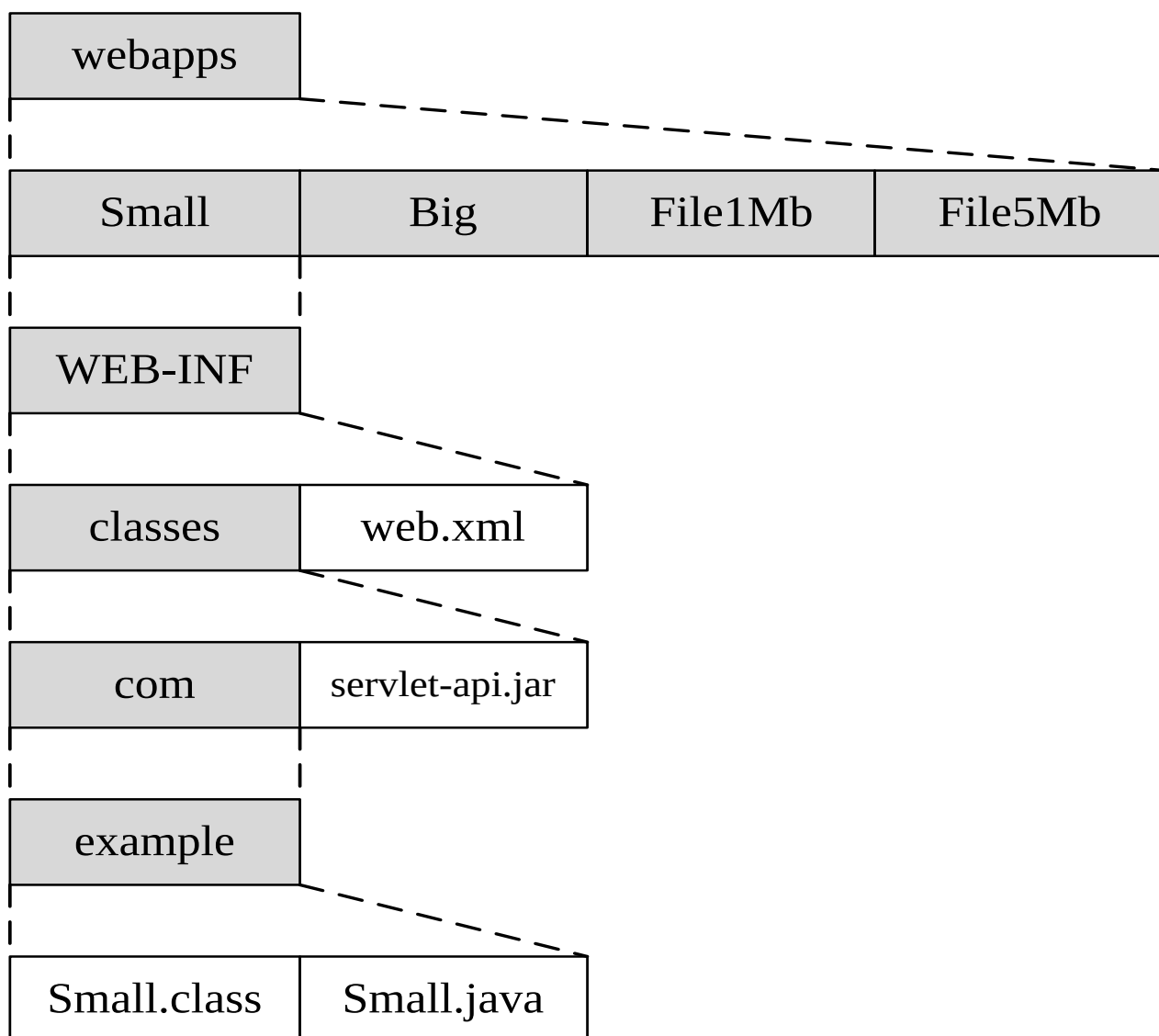


Рисунок 10 – Структура каталога «webapps»

2.3 Разработка специального программного обеспечения серверной части модели

2.3.1 Требования предъявляемые к СПО-С.

Требования, предъявляемые к СПО-С, аналогичны требованиям приведенным в ТЗ на выпускную квалификационную работу (далее – ВКР).

Шифрование (при передаче) и расшифрование (при приеме) данных осуществляется с использованием симметричного криптоалгоритма с малой длиной ключа (не более 56 бит), аналогичному ГОСТ 28147-89[3] (далее – Криптоалгоритм).

2.3.2 Разработка алгоритма работы СПО-С.

На рисунке 11 показан алгоритм работы СПО-С. Вначале выполняется процедура конфигурации СПО-С в рамках, которой осуществляется настройка СПО-С для работы с сетевыми адаптерами и загрузка базы данных (БД) Клиентов.

Процедура «Обновление БД» осуществляет проверку аутентификации Клиентов, если с последнего момента приема данных от клиента прошло 60 секунд, аутентификация клиента сбрасывается. Выполнение данной процедуры осуществляется раз в 30 секунд.

Далее выполняется процедура приема данных, СПО-С ожидает поступления данных из сети. При поступлении данных из сети выполняется процедура анализа принятых данных. Принятыми данными является Ethernet-пакет.

На рисунке 12 показан алгоритм процедуры анализа принятых данных. Проверка от кого пришли данные осуществляется по протоколу транспортного уровня. Если это UDP протокол, то данные пришли от Клиента, если TCP протокол, то данные от WEB-сервера. IP-пакеты, содержащие другие протоколы транспортного уровня, не обрабатываются. Запросы от клиента обрабатываются в случае если:

- клиент отправил данные на 8080 порт VPN-сервера;
- данные от клиента не содержат запрос на аутентификацию;
- синхропосылка корректна;
- имитовставка совпала;
- клиент зарегистрирован в БД СПО-С и аутентифицирован.

Ответы от WEB-сервера обрабатываются в случае, если данные отправлены на порт VPN-сервера, который привязан к зарегистрированному Клиенту.

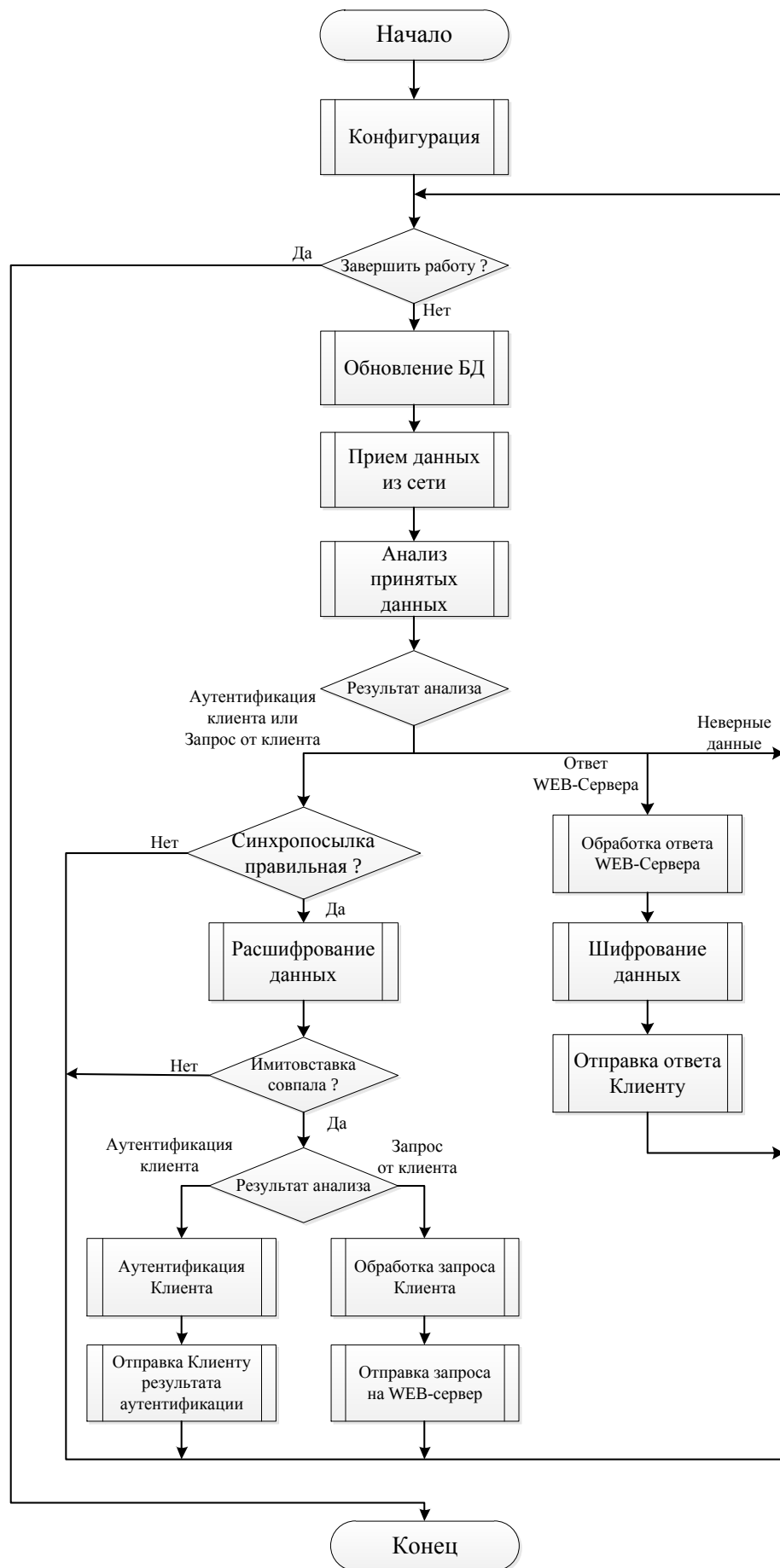


Рисунок 11 - Алгоритм работы СПО-С

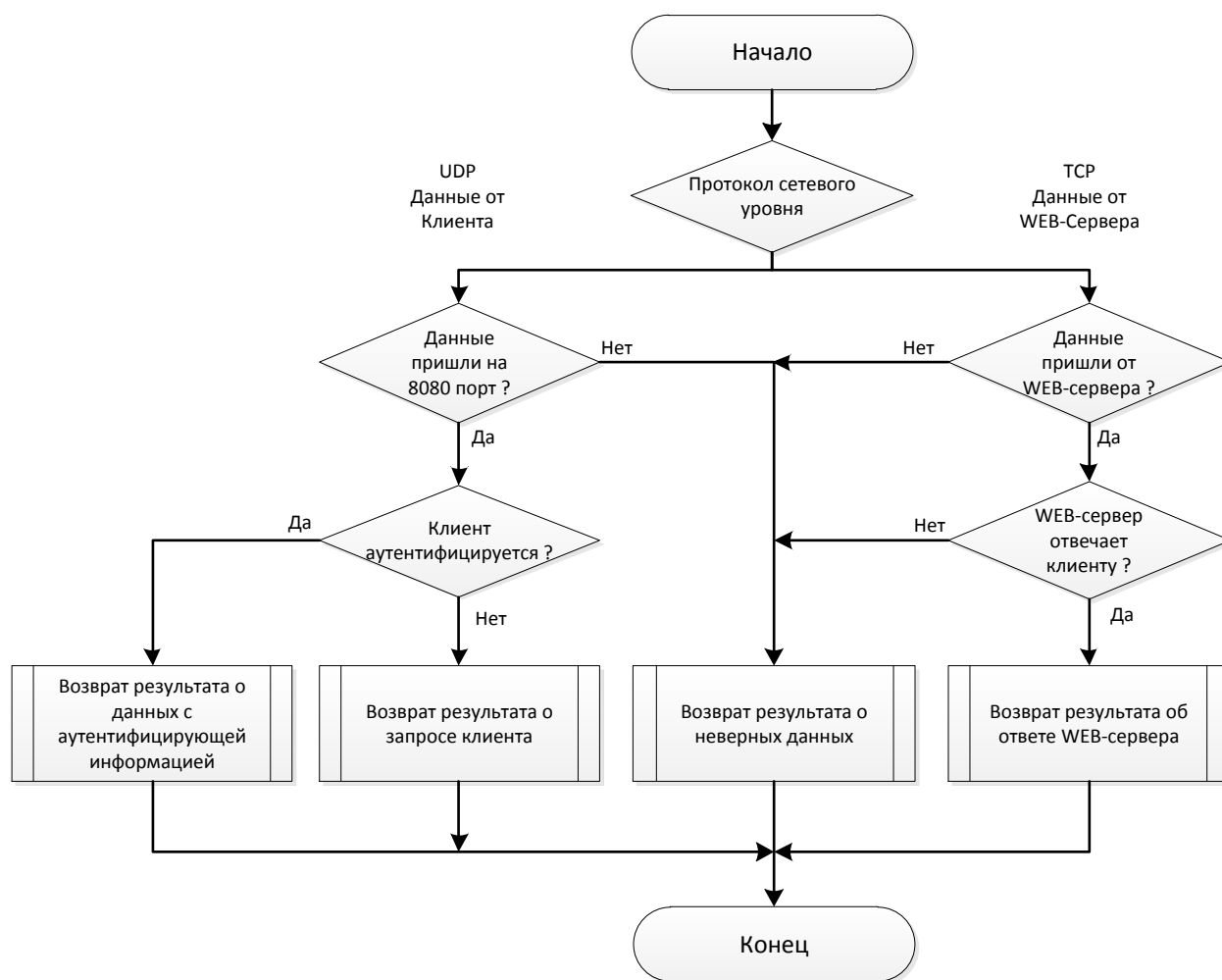


Рисунок 12 - Алгоритм процедуры анализа принятых данных

Дальнейшие действия СПО-С зависят от результатов выполнения процедуры анализа принятых данных. Возможные действия СПО-С:

- аутентификация клиента - проверка аутентифицирующей информации;
- обработка запроса Клиента;
- обработка ответа WEB-сервера;
- отбрасывание принятых данных.

Большинство действий СПО-С используют процедуры шифрования или расшифрования, данные процедуры основаны на Криптоалгоритме.

Перед расшифрованием проверяется синхропосылка. Синхропосылкой является локальное время устройства в момент начала шифрования. Если время Клиента отличается от времени Криптосервера больше чем на 86400 с. (24 часа) синхропосылка считается неправильной.

На рисунке 13 показан алгоритм шифрования и расшифрования данных.

Шифрование



Расшифрование

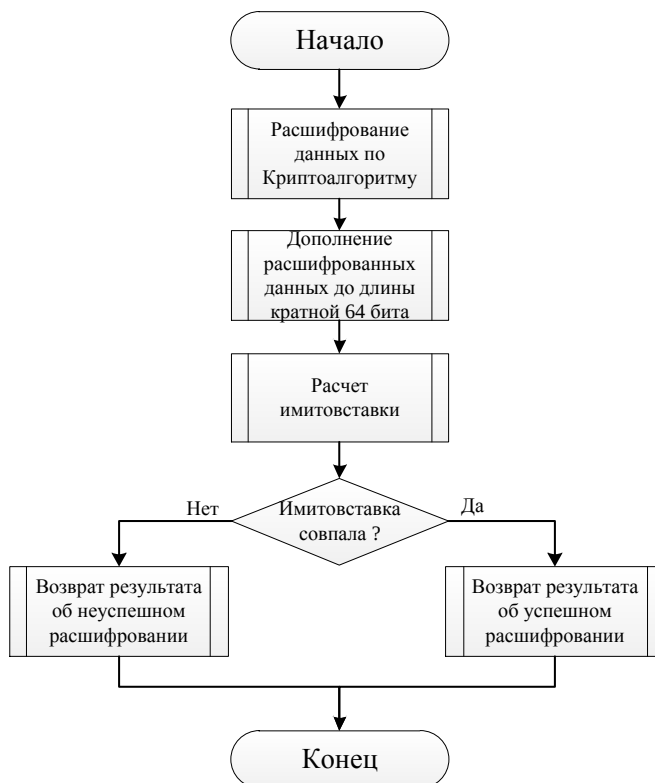


Рисунок 13 – Алгоритм шифрования и расшифрования данных

На рисунке 14 показан алгоритм работы процедуры «Аутентификация Клиента».

При аутентификации от Клиента приходят данные, структура которых показана на рисунке 15, серым цветом выделены зашифрованные данные.

Строка «Connect» является обязательной для осуществления процедуры аутентификации и свидетельствует о том, что клиент передал пароль.

Синхропосылка используется в СПО-С для расшифрования пароля.

Пароль и IMEI является аутентифицирующей информацией.

Если Клиент передал правильный пароль, аутентификация считается успешной. Далее осуществляется проверка регистрации Клиента в БД СПО-С. Если клиент ранее не был зарегистрирован, для него в БД создается новая запись, иначе его запись обновляется. Нахождение записи Клиента осуществляется по полю «IMEI».

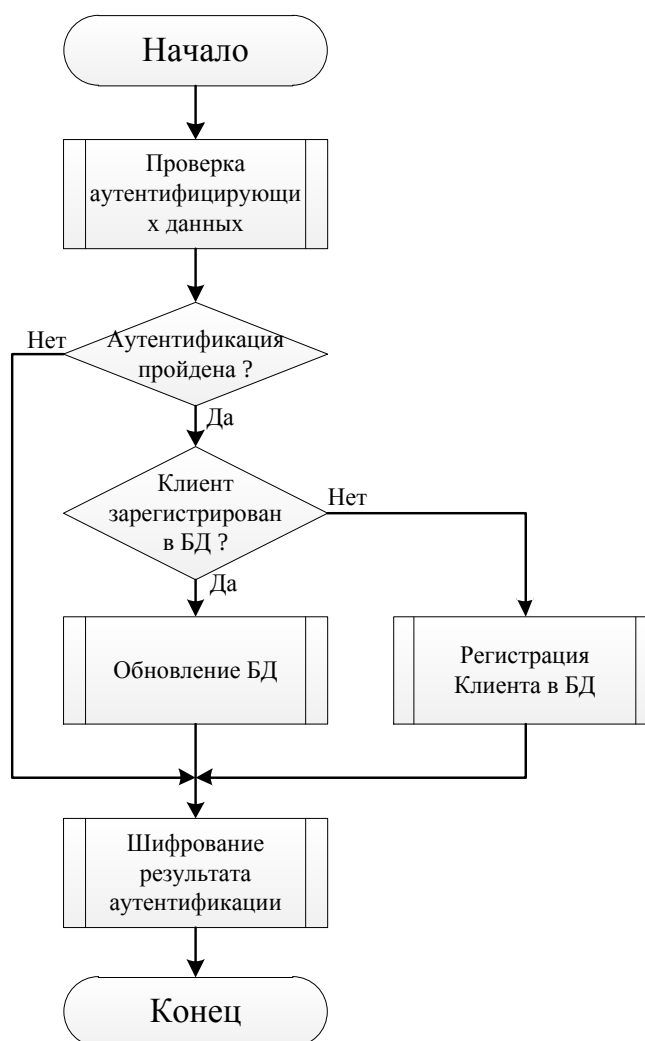


Рисунок 14 – Алгоритм процедуры «Аутентификация Клиента»

"Connect"	Синхропосылка	IMEI	Пароль	Имитовставка
7 байт	8 байт	8 байт	N байт	4 байта

Рисунок 15 – Структура данных, посылаемых Клиентом при аутентификации

В БД Криптосервера для каждого пользователя хранится запись, структура которой показана на рисунке 16, цифрами обозначен размер поля в байтах. При регистрации Клиента заполняются поля «IP-Адрес», «Порт Клиента», «IMEI», «MAC-Адрес», «Время последнего сообщения», «Метка аутентификации», «Номер последнего установленного ТСРсоединения». Поля «ТСР Соединение» используются для обеспечения обмена сообщениями между Клиентом и WEB-сервером.

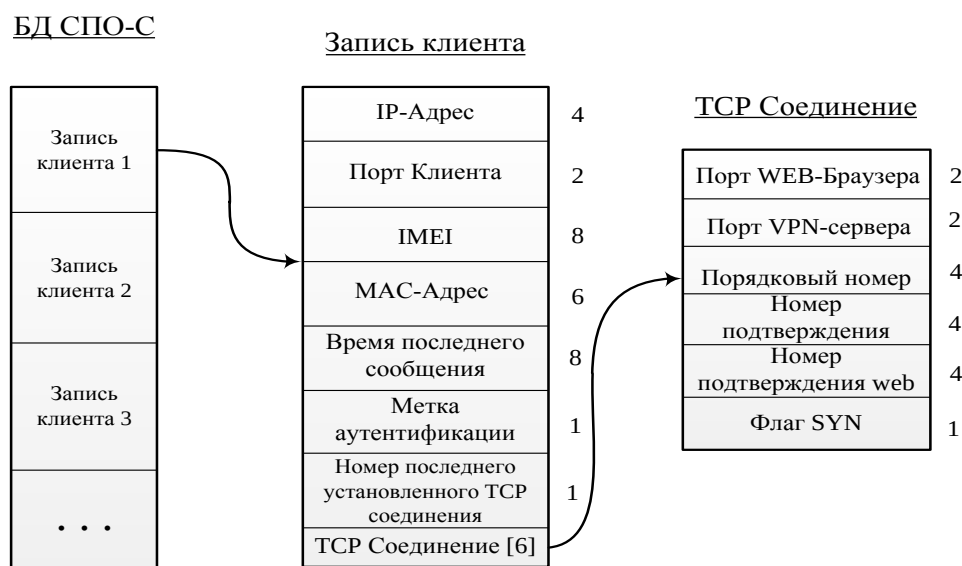


Рисунок 16 – Запись в БД о Клиенте

БД СПО-С реализована программно, в виде массива структур «Запись клиента». При загрузке, обновлении и добавлении новых записей используется файл «bdClients.txt», структура данных файла приведена на рисунке 17.

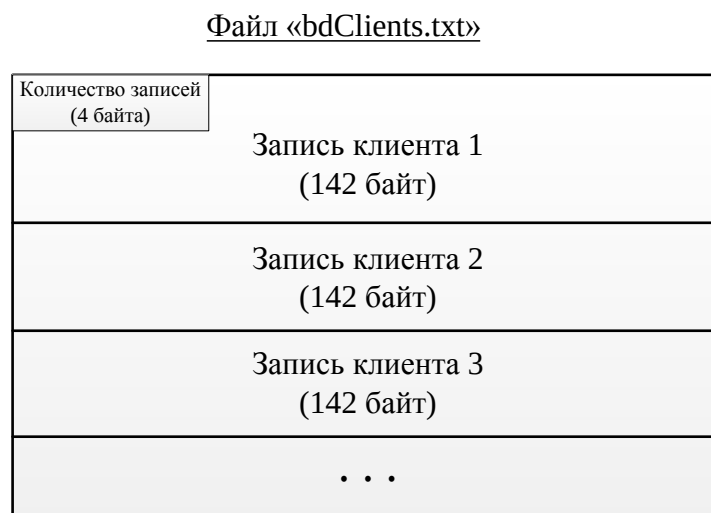


Рисунок 17 – Структура данных файла «bdClients.txt»

Далее СПО-С формирует ответ с результатами аутентификации, шифрует его и отправляет Клиенту. На рисунке 18 показана структура ответного пакета при успешной или неуспешной аутентификации, серым цветом выделены зашифрованные данные.

Успешная аутентификация

Синхропосылка	"a,"	"IP-Адрес Клиента"	",24 г,0.0.0.0,0"	Имитовставка
8 байт	2 байта	N байт	15 байт	4 байта

Неуспешная аутентификация

Синхропосылка	"p,errorPassword"	Имитовставка
8 байт	15 байт	4 байта

Рисунок 18 – Структура ответного пакета VPN-сервера при успешной или неуспешной аутентификации

На рисунке 19 показан алгоритм работы процедуры «Обработка клиентских запросов». Расшифрованный IP-пакет, непредназначенный для передачи на WEB-сервер, не обрабатывается. Исключение составляет «Служебный UDP-Пакет». Данный пакет предназначен для поддержания соединения. При принятии данного пакета, СПО-С обновляет в БД в записи Клиента значение поля «Время последнего сообщения», заменяя его на текущее время VPN-сервера.

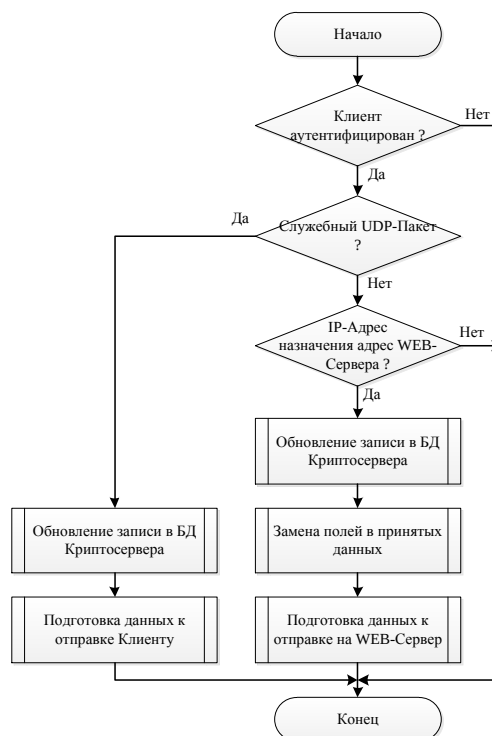


Рисунок 19 – Алгоритм работы процедуры «Обработка клиентских запросов»

Структура «Служебного UDP-Пакета» и ответного пакета VPN-сервера показана на рисунке 20.

Пакет поддержания соединения

Синхропосылка	IMEI	"01234567"	Имитовставка
8 байт	8 байт	8 байт	4 байта

Ответный пакет Криптосервера

Синхропосылка	"45670123"	Имитовставка
8 байт	8 байт	4 байта

Рисунок 20 – Структура «Служебного UDP-Пакета»

На рисунке 21 показана схема обработки СПО-С принятых запросов от Клиента к WEB-ресурсу. После расшифрования IP-пакета, СПО-С создает новую или обновляет существующую запись «TCP-соединение». Данная запись используется для корректного установления TCP соединения между WEB-браузером Клиента и WEB-сервером. Каждому порту WEB-браузера ставится в соответствие порт VPN-сервера. Создание новой записи «TCP-соединение» осуществляется при наличии установленного флага SYN.

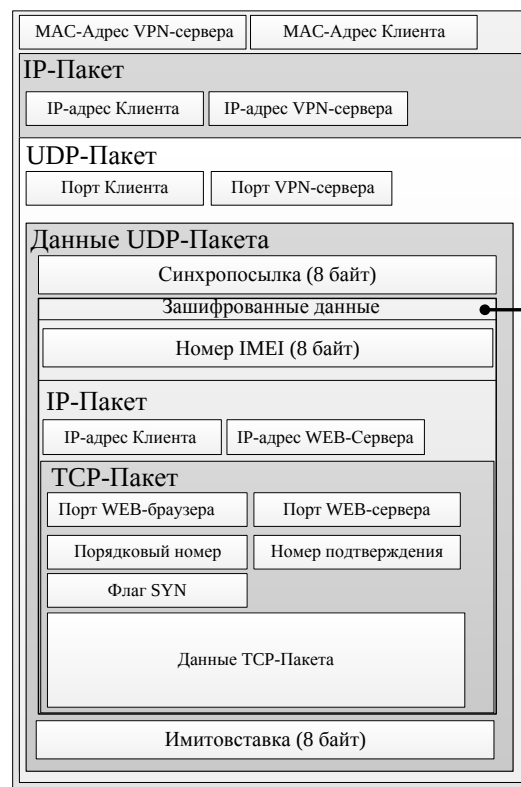
Далее СПО-С заменяет в IP-заголовке адрес Клиента на адрес VPN-сервера, в TCP-заголовке порт WEB-браузера заменяет в соответствии с записью «TCP-соединение», рассчитывает контрольные суммы, упаковывает полученный IP-пакет в Ethernet-пакет и отправляет на WEB-сервер.

Формат сетевого пакета с зашифрованными данными, отправляемый Клиентом, приведен в приложении Б.

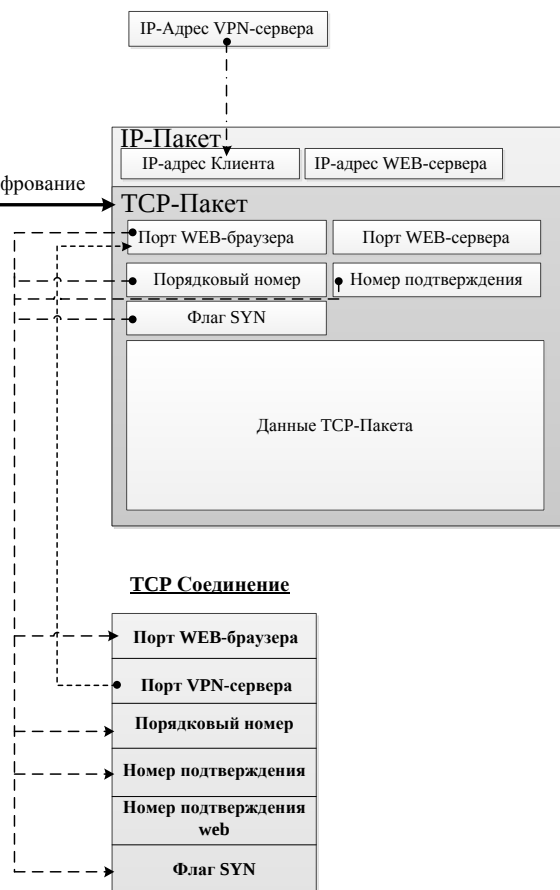
На рисунке 22 показан алгоритм процедуры «Обработка ответов WEB-ресурса». Если в БД СПО-С не нашлся Клиент, для которого предназначен ответ WEB-сервера, данный ответ считается устаревшим, в ответ на него СПО-С шлет команду на разрыв TCP соединения.

На рисунке 23 показана схема обработки СПО-С принятых от WEB-сервера данных. По значению поля «Порт VPN-сервера» СПО-С определяет, какому Клиенту предназначен ответ WEB-сервера, заменяет данное значение на значение поля «Порт WEB-браузера», зашифровывает полученный IP-пакет, упаковывает зашифрованные данные в UDP-пакет и отправляет Клиенту.

Принятый Ethernet-Пакет



Расшифрование



Ethernet-Пакет отправляемый на WEB-сервер

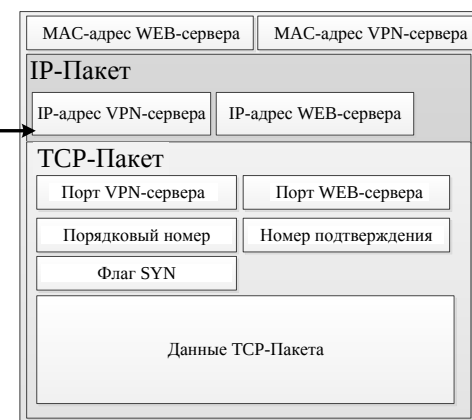


Рисунок 21 – Схема обработки принятых от Клиента запросов

Также СПО-С анализирует объем данных для отправки Клиенту, если он превышает максимально допустимый, СПО-С разбивает данные на две части и последовательно передает их Клиенту.

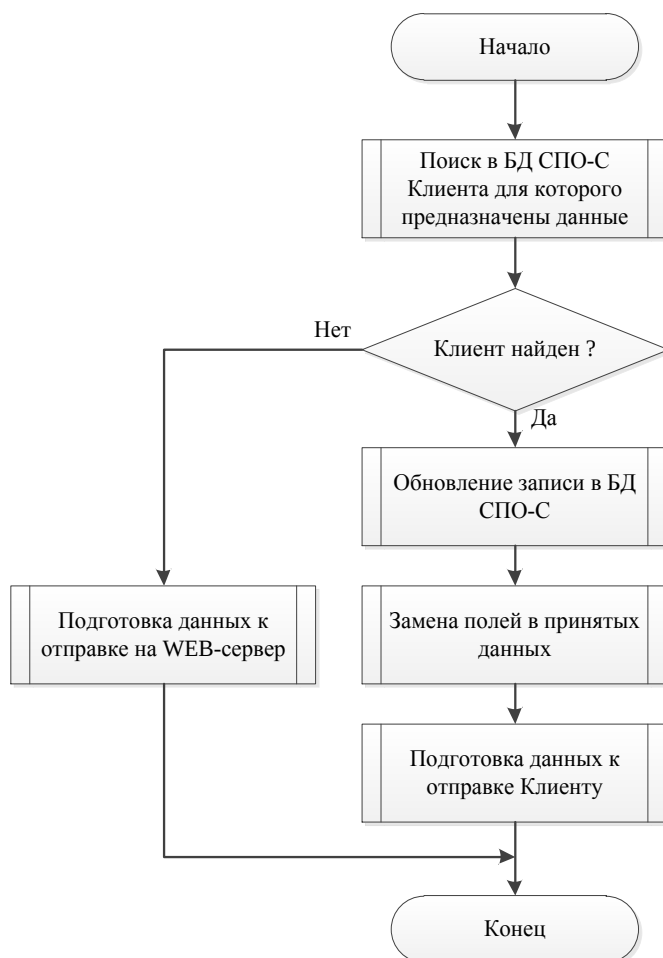


Рисунок 22 – Алгоритм работы процедуры «Обработка ответов WEB-ресурса»

Для пометки каждой части данных, к ним добавляется поле «№», значения данного поля:

- «0» - данные не разбивались на части;
- «1» - первая часть данных;
- «2» - вторая часть данных.

Во второй части данных синхропосылка не передается.

Поля «Порядковый номер» и «Номер подтверждения web» используются для корректной работы TCP соединения между Клиентом и WEB-сервисом.

Формат сетевого пакета с зашифрованными данными, отправляемый VPN-сервером, приведен в приложении Б.

Ethernet-Пакет отправляемый Клиенту

Принятый Ethernet-Пакет

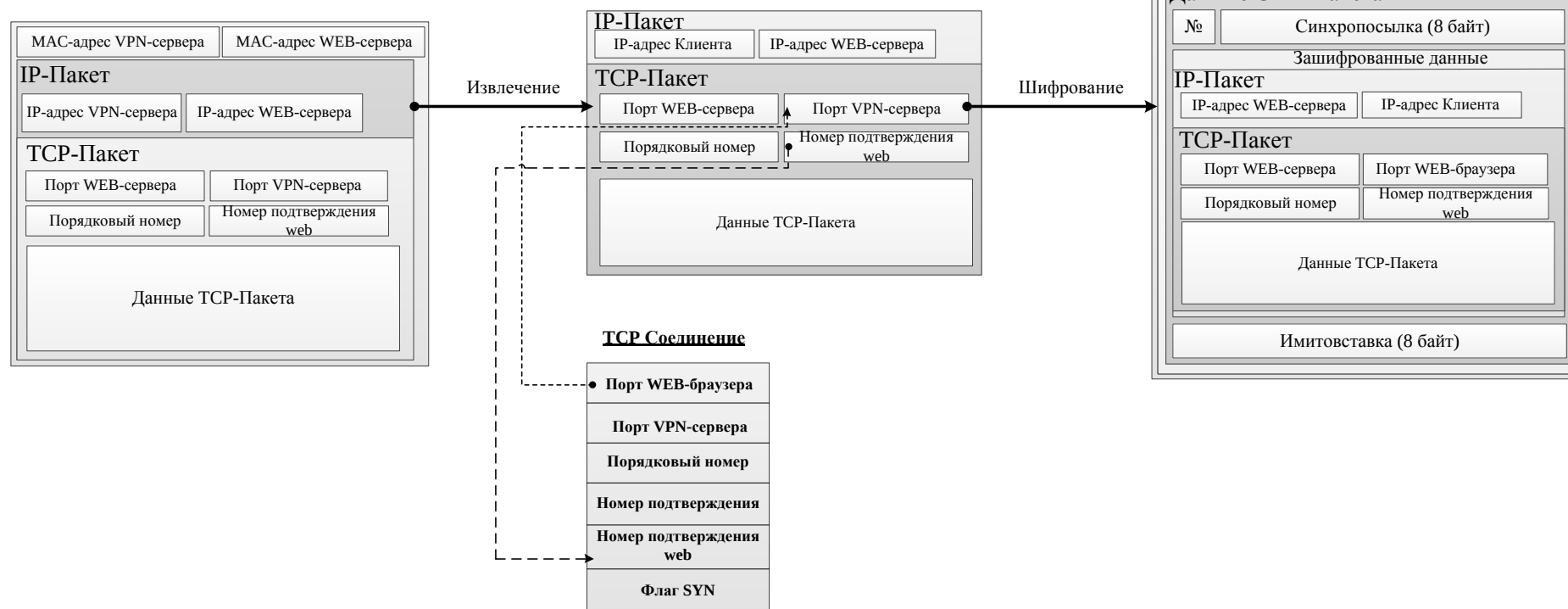


Рисунок 23 – Схема обработки принятых от WEB-сервера данных

2.4.3 Реализация алгоритма работы СПО-С.

СПО-С написан на языке C++ ANSI (American National Standards Institute). Компиляция СПО-С осуществляется с помощью линковщика. При разработке СПО-С использовалась среда разработки «Microsoft Visual Studio 2010 Professional» версии 10.0.30319.1.

СПО-С удовлетворяет техническим требованиям.

СПО-С обеспечивает реализацию следующих задач:

- прием данных с сети Ethernet;
- криптографическая обработка данных сетевого протокола IPv4;
- имитозащита передаваемых данных;
- обработка запросов клиентов к WEB-ресурсу;
- обработка ответов WEB-ресурса;
- аутентификация Клиентов.

В таблице 2 приведен состав и назначение модулей СПО-С. Подробное описание модулей приведено далее.

Таблица 2 - Состав и назначение модулей СПО-С

Наименование модуля	Состав модуля	Назначение модуля
Gost28147	Данные и методы класса «Gost28147»	Обеспечивает криптографическую обработку входящего и исходящего сетевого трафика по Криптоалгоритму
VPNServer	Данные и методы класса «VPNServer», библиотека «WinPcap»	Основной модуль СПО-С реализующий работу VPN-сервера
PacketMaster	Данные и методы класса «PacketMaster»	Модуль осуществляет работу с сетевыми пакетами

Для реализации алгоритмов криптографической обработки данных был написан класс «Gost28147» осуществляющий шифрование и расшифрование

данных по Кryptoалгоритму. В таблице 3 показаны данные класса. В таблице 4 показаны методы класса.

Текст программы, реализующий класс «Gost28147», приведен в приложении В.

Таблица 3 – Данные класса «Gost28147»

Имя переменной	Тип переменной	Описание
SP	unsigned long *	Синхрoпосылка
Key	unsigned long *	Ключ шифрования
Tz	unsigned long *	Таблица подстановок
Din	unsigned long *	Исходные данные
Dout	unsigned long *	Результат
LenBytes	unsigned long	Длина данных в байтах
D	unsigned long	Вспомогательная переменная
E	unsigned long	Вспомогательная переменная
N	unsigned long [2]	Вспомогательная переменная
M	unsigned long [4]	Вспомогательная переменная

Таблица 4 – Методы класса «Gost28147»

Имя метода	Модификатор доступа	Описание метода
cripted	Приватный	Метод, осуществляющий шифрование по Кryptoалгоритму
imvstavka	Приватный	Метод, осуществляющий выработку имитовставки по Кryptoалгоритму
gost_zam_izm	Приватный	Вспомогательная функция
gost_zam_enc	Приватный	Вспомогательная функция
gost_elem	Приватный	Вспомогательная функция
setLenBytes	Приватный	Метод установки
setDin	Приватный	Метод установки исходных данных

Продолжение таблицы 4

Имя метода	Модификатор доступа	Описание метода
getDout	Приватный	Метод установки результата
Gost28147	Публичный	Конструктор класса по умолчанию
~Gost28147	Публичный	Деструктор класса по умолчанию
criptedData	Публичный	Метод, осуществляющий расшифрование входных данных и проверку имитовставки
decriptedData	Публичный	Метод, осуществляющий шифрование входных данных и выработку имитовставки

Для реализации приема данных из сети Ethernet был написан класс «VPNServer». Данный класс использует методы библиотеки «WinPCap». В таблице 5 показаны данные класса. В таблице 6 показаны методы класса.

Текст программы, реализующий класс «VPNServer», приведен в приложении В.

Таблица 5 – Данные класса «VPNServer»

Имя переменной	Тип переменной	Описание
alldevs	Указатель на pcap_if_t	Объект библиотеки «WinPCap», хранит список доступных сетевых устройств
errbuf	char[]	Массив для сохранения сообщения об ошибке
fp	Указатель на pcap_t	Объект библиотеки «WinPCap», хранит объект выбранного сетевого устройства
header	Указатель на pcap_pkthdr	Объект библиотеки «WinPCap», содержит размер принятых из сети данных
SIZE_DB_CLIENTS	int	Количество записей в БД
bdClients	Указатель на	БД клиентов

Продолжение таблицы 5

Имя переменной	Тип переменной	Описание
	BDClients	
timeUpdateBD	long long	Время последней проверки БД
timeUpdate	long long	Время между проверками БД
timeChekClient	long long	Время, которое дается клиентам на ответ
packet_filter	Указатель на char	Строка содержащая параметры для фильтрации принимаемых данных

Таблица 6 – Методы класса «VPNServer»

Имя метода	Модификатор доступа	Описание метода
ipToString	Приватный	Метод преобразования IP-адреса в строку
updateClientAut	Приватный	
VPNServer	Публичный	Конструктор класса по умолчанию
~ VPNServer	Публичный	Деструктор класса по умолчанию
configure	Публичный	Метод, реализующий выбор сетевого устройства и настройки фильтров принимаемых сетевых пакетов.
receiveData	Публичный	Метод, реализующий прием данных из сети Ethernet
sendData	Публичный	Метод, реализующий передачи данных в сеть Ethernet
wtiteLog	Публичный	Метод, ведущий запись событий Криптосервера
getClient	Публичный	Метод получения записи клиента по индексу
updateTCPConnect	Публичный	Метод обновления поля

Продолжение таблицы 6

Имя метода	Модификатор доступа	Описание метода
		«TCPConnect»
registrationClient	Публичный	Метод регистрации клиента в БД
searchClientForPortWEB	Публичный	Метод поиска записи клиента по порту Криптосервера
searchPortHiden	Публичный	Метод возвращает поле «Порт WEB-Браузера»
searchClient	Публичный	Метод поиска зарегистрированного и аутентифицированного клиента
updateTimeClient	Публичный	Метод обновления времени последнего сообщения клиента

В таблице 7 приведены используемые функции библиотеки «WinPCap».

Таблица 7 – Функции библиотеки «WinPCap»

Имя метода	Описание метода
pcap_findalldevs	Функция получения списка сетевых устройств с локального компьютера
pcap_open_live	Функция настройки сетевого устройства для захвата пакетов
pcap_compile	Функция проверки строки-фильтра
pcap_setfilter	Функция настройки фильтрации пакетов сетевого устройства
pcap_next_ex	Функция приема сетевых данных
pcap_sendpacket	Функция отправки данных в сеть

Для работы с принятыми из сети данными (Ethernet-пакетами) был написан класс «PacketMaster». Данный класс позволяет изменять поля заголовков сетевых протоколов (Ethernet, IPv4, TCO, UDP) и создавать новые пакеты сетевых протоколов. В таблице 8 показаны методы класса «PacketMaster».

Таблица 8 – Методы класса «PacketMaster»

Имя метода	Модификатор доступа	Описание метода
Вспомогательные методы		
PacketMaster	Публичный	Конструктор класса по умолчанию
~PacketMaster	Публичный	Деструктор класса по умолчанию
create	Публичный	Метод заполнения данных класса
setDataToPoint	Публичный	Метод копирования данных по указанному адресу
getDataToPoint	Публичный	Метод получения данных по указанному адресу
setByteToPoint	Публичный	Метод установки байта по указанному адресу
Методы для работы с заголовком Ethernet-пакета		
getData	Публичный	Метод получения указателя на Ethernet-пакет
getDstMAC	Публичный	Метод получения указателя на MAC-адрес назначения
getSrcMAC	Публичный	Метод получения указателя на MAC-адрес источника
setDstMAC	Публичный	Метод установки MAC-адреса назначения
setSrcMAC	Публичный	Метод установки MAC-адреса источника
Методы для работы с заголовком IP-пакета		
getIpPacket	Публичный	Метод получения указателя на IP-пакет
setIpPacket	Публичный	Метод установки IP-пакета
getLenIP	Публичный	Метод получения размера пакета
setLenIP	Публичный	Метод установки размера пакета
getProtocol	Публичный	Метод получения номера сетевого протокола
setProtocol	Публичный	Метод установки номера сетевого протокола
getSrcIp	Публичный	Метод получения адреса источника
getDstIp	Публичный	Метод получения адреса назначения

Продолжение таблицы 8

Имя метода	Модификатор доступа	Описание метода
Методы для работы с заголовком IP-пакета		
setSrcIp	Публичный	Метод установки адреса источника
setDstIp	Публичный	Метод установки адреса назначения
calcChekSumIP	Публичный	Метод подсчета контрольной суммы IP-пакета
Методы для работы с заголовком UDP-пакета		
getSizeDataUDP	Публичный	Метод получения размера данных UDP-пакета
getDataUDP	Публичный	Метод получения указателя на данные UDP-пакета
setDataUDP	Публичный	Метод установки данных UDP-пакета
setLenUDP	Публичный	Метод установки размера UDP-пакета
getDstPort	Публичный	Метод получения порта назначения
getSrcPort	Публичный	Метод получения порта источника
setDstPort	Публичный	Метод установки порта назначения
setSrcPort	Публичный	Метод установки порта источника
calcChekSumUDP	Публичный	Метод подсчета контрольной суммы UDP-пакета
Методы для работы с заголовком TCP-Пакета		
getFlagsTCP	Публичный	Метод получения флагов TCP-Пакета
setFlagsTCP	Публичный	Метод установки флагов TCP-Пакета
getSeqNumber	Публичный	Метод получения порядкового номера
getAckNumber	Публичный	Метод получения номера подтверждения
setSeqNumber	Публичный	Метод установки порядкового номера
setAckNumber	Публичный	Метод установки номера подтверждения
setDataTCP	Публичный	Метод установки данных TCP-Пакета
calcChekSumTCP	Публичный	Метод подсчета контрольной суммы TCP-Пакета

Текст программы, реализующий класс «PacketMaster», приведен в приложении В.

Для реализации алгоритма анализа принятых данных была написана функция «checkPacket». Данная функция возвращает следующие значения:

- 'a' – принятые данные содержат аутентифицирующие данные;
- 'c' – принятые данные содержат запрос Клиента или «Служебный UDP-Пакет»;
- 'w' – принятые данные содержат ответ WEB-сервиса;
- 'n' – принятые данные не обрабатываются.

В данной функции были использованы два метода из класса «VPNServer»:

- «searchClient» – функция осуществляет поиск Клиента по IP-адресу и порту в БД Криптосерва. Если запись Клиента не найдена, функция возвращает «-1». При нахождении пользователя, проверяется аутентифицирован ли он, если аутентифицирован, функция возвращает номер (индекс) записи клиента, иначе «-1»;
- «searchClientForPortWEB» – функция осуществляет поиск Клиента по порту Криптосервера. Если запись Клиента не найдена, функция возвращает «-1». При нахождении пользователя, проверяется аутентифицирован ли он, если аутентифицирован, функция возвращает номер (индекс) записи клиента, иначе «-1».

Текст функции «checkPacket» приведен в приложении В.

Аутентификация клиентов была реализована со следующими упрощениями:

- аутентификация клиентов производится только по присланному паролю;
- пароль для всех пользователей одинаковый;
- пароль «12345678» записан в оперативную память VPN-сервера.

Присланные аутентифицирующие данные (пароль) проверяются по размеру (если размер больше 8 символов, аутентификация не пройдена) и

посимвольно сравниваются с эталонным паролем. Если присланные данные и эталонный пароль совпадают, аутентификация считается успешной.

Далее выполняется метод «registrationClient» класса «VPNServer», осуществляющий регистрацию клиента.

По результатам аутентификации формируются данные структура, которых, показана на рисунке 14.

Код программы реализующей алгоритм аутентификации клиентов приведен в приложении В.

Текст программы, осуществляющий обработку запросов клиентов, представлен в приложение В.

Для определения типа данных пришедших от клиента («Служебный UDP-Пакет» или запрос) проверяется размер данных UDP-пакета. Если размер равен 20 байтам, предполагается, что пришел «Служебный UDP-Пакет», иначе предполагается, что пришел запрос (TCP-пакет).

«Служебный UDP-Пакет» проверяется на содержимое согласно рисунку 16.

Запрос клиента обрабатывается, если он отправлен WEB-ресурсу. Обработка запроса заключается в обновление записи клиента в БД и обновление полей TCP-пакета. Обновление записи клиента в БД осуществляет метод «updateTCPConnect» класса «VPNServer».

Текст программы, осуществляющей обработку ответов WEB-ресурса, представлен в приложение В.

Поиск в БД Клиента, для которого предназначен ответ WEB-ресурса, выполняется методом «searchPortHidden» класса «VPNServer». Данный метод по номеру записи находит в БД запись Клиента. В данной записи сопоставляются поля «Порядковый номер» и «Номер подтверждения» с соответствующими значениями в TCP-пакете, присланным WEB-сервером. Если данные значения удалось соотнести по правилам TCPсоединения, метод возвращает значение поля «Порт WEB-Браузера», иначе возвращает 0.

Если метод возвратил 0, на WEB-сервер отправляется TCP-пакет с установленным флагом RST. Если было возвращено значение порта, принятый IP-пакет обновляется (в соответствии со схемой показанной на рисунке 19) и отправляется Клиенту.

2.4 Разработка специального программного обеспечения клиентской части модели

2.4.1 Требования предъявляемые к СПО-К.

Требования, предъявляемые к СПО-К, аналогичны требованиям, приведенным в ТЗ на ВКР.

Шифрование (при передаче) и расшифрование (при приеме) данных осуществляется с использованием Криптоалгоритма.

2.4.2 Разработка алгоритма работы СПО-К.

На рисунке 24 показан алгоритм работы СПО-К.

При запуске СПО-К на смартфоне пользователя, СПО-К запускает «Игровую» активность. «Игровая» активность является замаскированным способом ввода цифрового пароля, что соответствует техническим требованиям.

После успешного ввода цифрового пароля СПО-К запускает «Основную» активность. Данная активность обеспечивает возможность просмотра и редактирования сетевых настроек, включающих IP-адреса номер сетевого порта VPN-сервера, и аутентифицирующего пароля.

Также «Основная» активность предоставляет две кнопки «Connect» и «Disconnect» при нажатии, на которые, осуществляется запуск или остановка сервиса VPN-соединения.

На рисунке 25 показан алгоритм работы сервиса VPN-соединения.

В процедуру «Конфигурация» входит создание UDP-сокета, с параметрами подключения, полученными от пользователя и другие необходимые настройки.

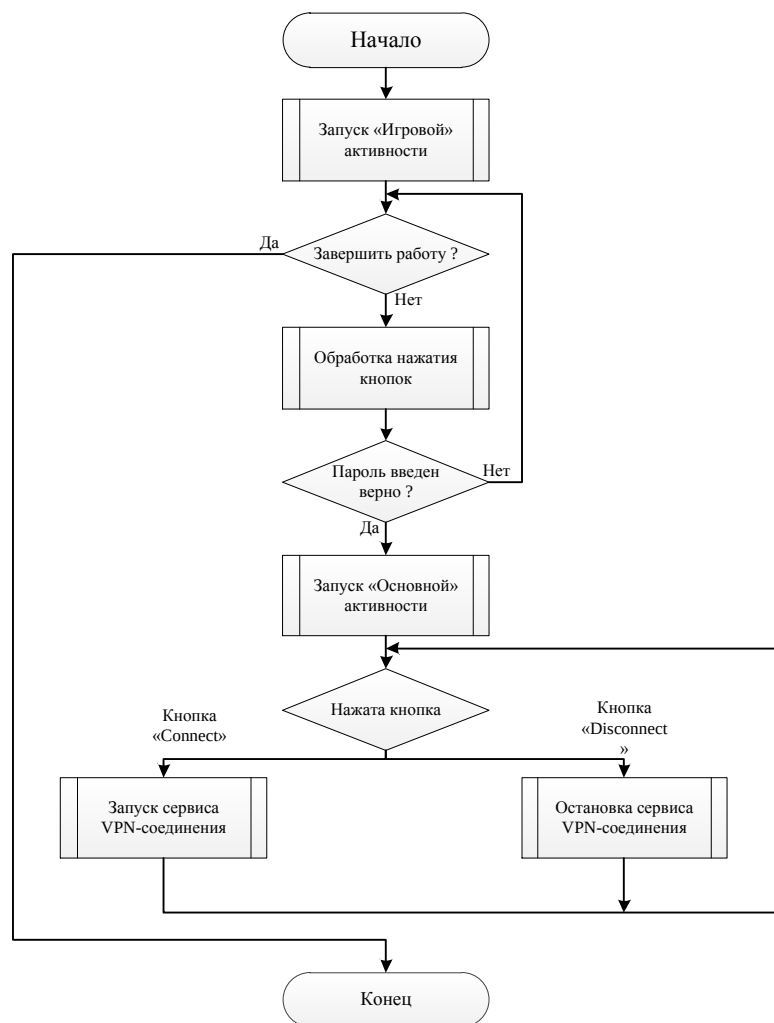


Рисунок 24 – Алгоритм работы СПО-К

Далее СПО-С, используя настроенный UDP-сокеты, проверяет доступность VPN-сервера с соответствующими сетевыми параметрами (IP-адрес и порт). Если VPN-сервер оказался недоступен, работа сервиса VPN-соединения прекращает свою работу.

Далее выполняется процедура установления соединения (Аутентификация). СПО-С формирует запрос на аутентификацию, приведенный на рисунке 15. И ожидает ответа от VPN-сервера. Варианты ответов VPN-сервера показаны на рисунке 18. Полученный IP-адрес от VPN-сервера устанавливается во всех IP-пакетах перехваченных сервисом VPN-соединения.

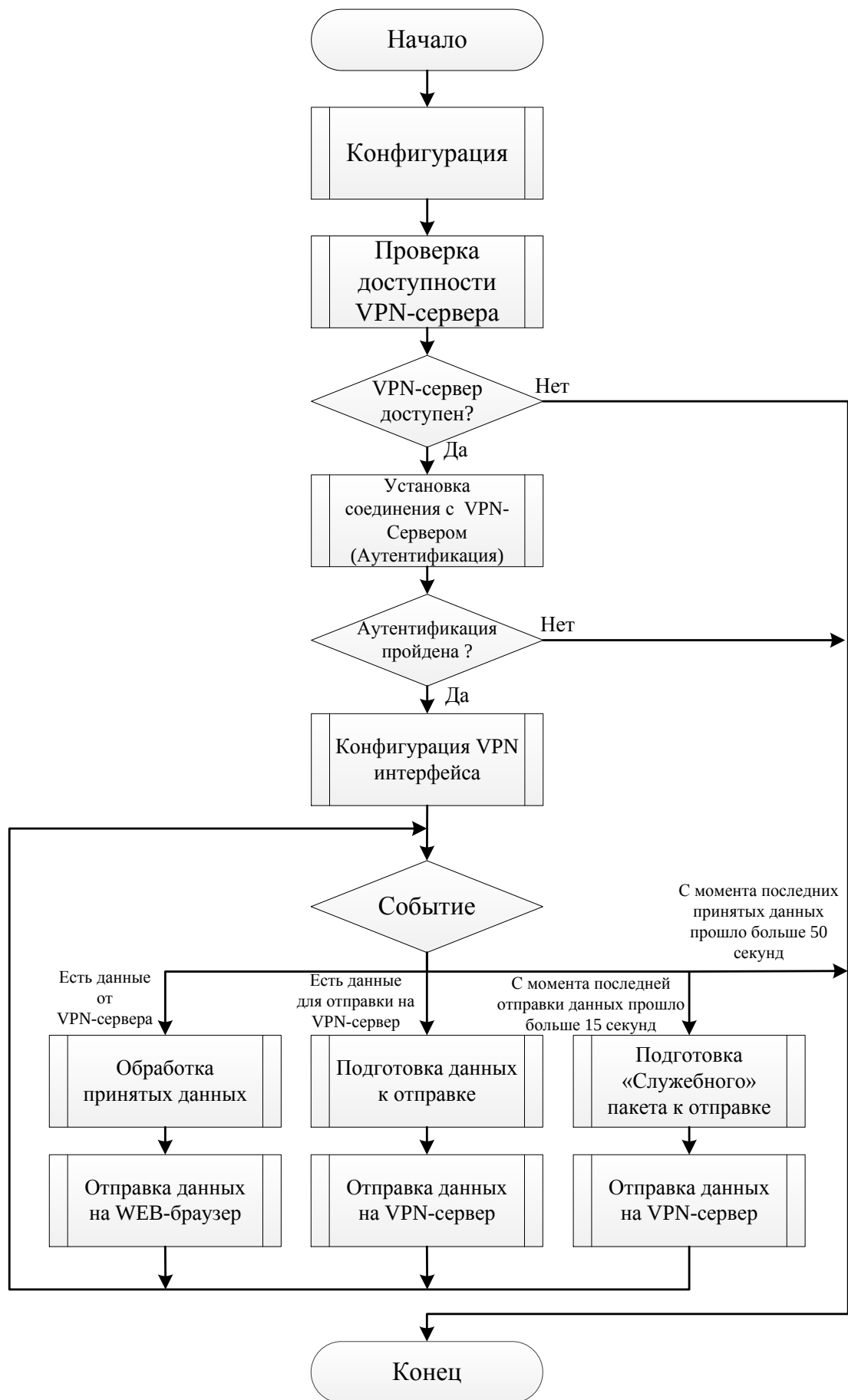


Рисунок 25 – Алгоритм работы сервиса VPN-соединения

Дальнейшие действия сервиса VPN-соединения зависят от наступления одного из событий:

- сервис перехватил данные от WEB-браузера. Полученные данные будут обработаны и отправлены на VPN-сервер. Формат исходящего пакета приведен в приложении Б;
- сервис получил данные от VPN-сервера. Полученные данные будут обработаны и отправлены на WEB-браузер. Формат входящего пакета приведен в приложении Б;
- время с последнего момента времени передачи данных на VPN-сервер превысило 15 секунд. Сервис VPN-соединения формирует «Служебный UDP-Пакет» и отправляет его на VPN-сервер. Структура данного пакета приведена на рисунке 20;
- время с последнего момента времени приема данных от VPN-сервера превысило 50 секунд. Сервис VPN-соединения прекращает свою работу.

2.4.3 Реализация алгоритма работы СПО-К.

СПО-К написан на языке программирования Java. При разработке СПО-К использовалась среда разработки «AndroidStudio» версии 2.3.3.

СПО-К удовлетворяет техническим требованиям.

СПО-К обеспечивает реализацию следующих задач:

- прием данных с сети Ethernet;
- криптографическая обработка данных сетевого протокола IPv4;
- имитозащита передаваемых данных;
- обработка запросов клиентов от WEB-браузера к WEB-ресурсу;
- обработка ответов WEB-ресурса;
- обеспечение дистанционного мобильного доступа пользователей к защищенному WEB-ресурсу с использованием технологий Wi-Fi, 3G (UMTS), 4G (LTE).

В таблице 9 приведен состав и назначение модулей СПО-К. Подробное описание модулей приведено далее.

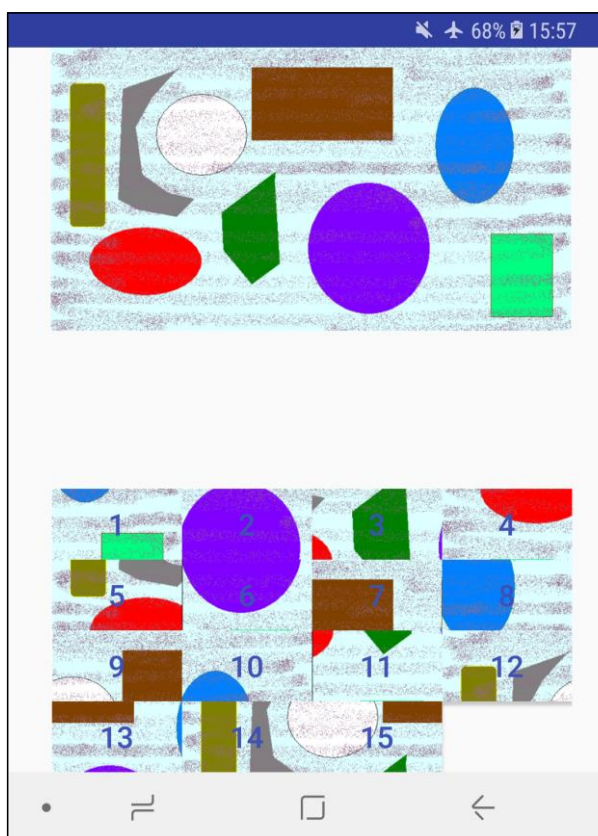
Таблица 9- Состав и назначение модулей СПО-К

Наименование модуля	Состав модуля	Назначение модуля
Gost28147	Данные и методы класса «Gost28147»	Обеспечивает криптографическую обработку входящего и исходящего сетевого трафика по Кryptoалгоритму
MainActivity	Данные и методы класса «MainActivity»	Начальное представление приложения, выполняет функции «Игрового» представления
VPNActivity	Данные и методы класса «VPNActivity»	Модуль выполняет функции «Основного» представления
FistVpnService	Данные и методы класса «FistVpnService»	Модуль выполняет функцию передачи данных между модулем VPNActivity и FistVpnConnection
FistVpnConnection	Данные и методы класса «FistVpnConnection»	Модуль выполняет функции сервиса VPN-соединения

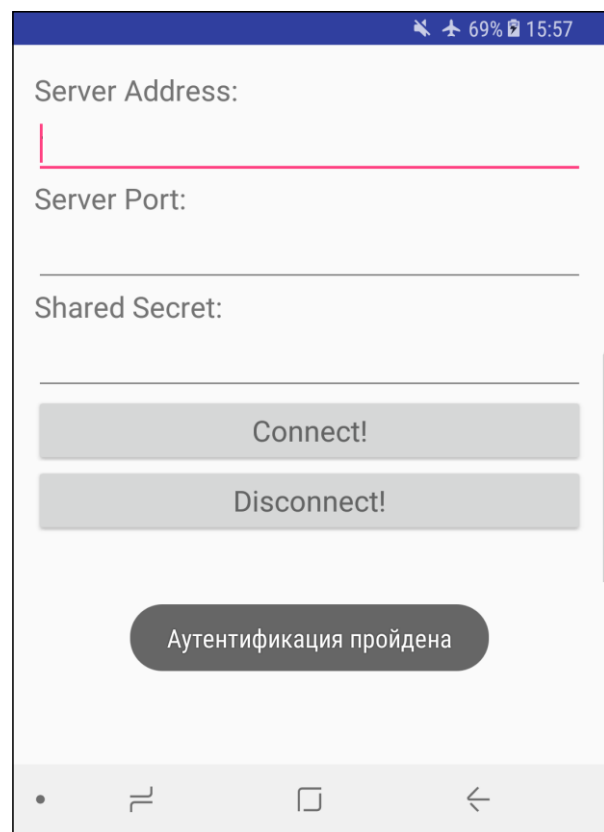
Для реализации алгоритмов криптографической обработки данных был написан класс «Gost28147» осуществляющий шифрование и расшифрование данных по Кryptoалгоритму. Данные и методы данного класса соответствуют тем, что приведены в таблицах 3 и 4 соответственно.

Для реализации «Игрового» представления был написан класс «MainActivity». Данное представление маскирует ввод цифрового пароля под игру «Фисташки». На рисунке 26 буквой «А» показан снимок экрана мобильного устройства с запущенным «Игровым» представлением.

Цифровой пароль вводится нажатием на кнопки с цифрами. Аутентификация считается успешной, если пользователь наберет текущее время на устройстве. В примере успешной аутентификации, рисунок 26 буква «Б», пользователь нажал на кнопки с цифрами «1», «5», «5», «7». Для набора символа «0» необходимо нажать на букву с числом «10».



А



Б

Рисунок 26 - Интерфейс пользователя СПО-К

В таблице 10 приведено описание данных класса. В таблице 11 приведено описание методов класса.

Текст программы, реализующий класс «MainActivity», приведен в приложении Г.

Таблица 10 – Описание данных класса «MainActivity»

Имя переменной	Тип переменной	Описание
button1 – button15	Button	Кнопки с цифрами
mas	int[4][4]	Массив, хранящий расположение кнопок
fullImage	ImageView	Картинка оригинал
clickNumber	int[4]	Массив, хранящий последние четыре номера нажатых кнопок

Таблица 11 – Описание методов класса «MainActivity»

Имя метода	Модификатор доступа	Описание метода
onCreate	Публичный	Метод создания активности
onClick	Публичный	Метод обработки нажатий кнопок
move	Приватный	Метод перемещения кнопок
pathSearching	Приватный	Метод определяет возможность перемещения нажатой кнопки
authentication	Приватный	Метод сверяет текущее время устройства с набранной комбинацией кнопок
setClickNumber	Приватный	Метод обновляет массив clickNumber при нажатии на кнопки

Для реализации «Основного» представления был написан класс «VpnActivity». Выполняемые функции данной активности приведены в пункте 2.4.2. На рисунке 26 буквой «Б» показан снимок экрана мобильного устройства с запущенным «Основным» представлением.

В таблице 12 приведено описание данных класса. В таблице 13 приведено описание методов класса.

Текст программы, реализующий класс «VpnActivity», приведен в приложении Г.

Таблица 12 – Описание данных класса «VpnActivity»

Имя переменной	Тип переменной	Описание
connect	Button	Кнопка запуска сервиса VPN-соединения
disconnect	Button	Кнопка остановки сервиса VPN-соединения
serverAddress	TextView	Текстовое поле для ввода IP-адреса VPN-сервера
serverPort	TextView	Текстовое поле для ввода сетевого порта VPN-сервера
sharedSecret	TextView	Текстовое поле для ввода пароля аутентификации

Таблица 13 – Описание методов класса «VpnActivity»

Имя метода	Модификатор доступа	Описание метода
onCreate	Публичный	Метод создания активности
onActivityResult	Приватный	Метод запуска сервиса VPN-соединения
getServiceIntent	Приватный	Вспомогательный метод запуска сервиса VPN-соединения
connect.setOnClickListener(...)	Приватный	Метод обработки нажатия на кнопку «Connect»
disconnect.setOnClickListener(...)	Приватный	Метод обработки нажатия на кнопку «Disconnect»

Для осуществления информационного обмена между активностью «VpnActivity» и сервисом VPN-соединения был написан класс «FistVpnService». Данный класс наследует класс «VPNService» APIAndroid, что позволяет использовать его методы. Также класс «FistVpnService» реализует интерфейс «Handler.Callback» для выдачи сообщений на экран устройства. Данный класс получает от активности «VpnActivity» команды на запуск или остановку сервиса VPN-соединения и выполняет необходимые действия.

В таблице 14 приведено описание данных класса. В таблице 15 приведено описание методов класса.

Текст программы, реализующий класс «FistVpnService», приведен в приложении Г.

Таблица 14 – Описание данных класса «FistVpnService»

Имя переменной	Тип переменной	Описание
fistVpnConnection	FistVpnConnection	Экземпляр класса «FistVpnConnection»
mHandler	Handler	Экземпляр класса

Продолжение таблицы 14

Имя переменной	Тип переменной	Описание
		«Handler»
mConnectingThread	AtomicReference<Thread>	Коллекция, хранящая запущенные сервисы VPN-соединения
mConnection	AtomicReference<Connection>	Коллекция, хранящая установленные соединения
mNextConnectionId	AtomicInteger	Коллекция, хранящая id подключений

Таблица 15 – Описание методов класса «FistVpnService»

Имя метода	Модификатор доступа	Описание метода
onCreate	Публичный	Метод создания сервиса FistVpnService
onStartCommand	Публичный	Метод обработки команд запуска или остановки сервиса VPN-соединения
onDestroy	Публичный	Метод завершения работы сервиса FistVpnService
handleMessage	Публичный	Метод вывода сообщений
connect	Приватный	Метод настраивает сервис VPN-соединения и запускает его
startConnection	Приватный	Метод запуска сервиса VPN-соединения
setConnectingThread	Приватный	Метод запускает сервис VPN-соединения в качестве нового потока
setConnection	Приватный	Метод закрывает последний созданный поток сервиса VPN-соединения
disconnect	Приватный	Метод остановки сервиса

Продолжение таблицы 15

Имя метода	Модификатор доступа	Описание метода
		VPN-соединения
updateForegroundNotification	Приватный	Метод настройки выдаваемых сообщений

Для реализации сервиса VPN-соединения был написан класс «FistVpnConnection». Данный класс реализует абстрактный метод «run» интерфейса APIAndroid«Runnable», что позволяет выполнять методы класса «FistVpnConnection» в отдельном потоке. Так же данный класс использует модуль «Gost28147» для криптографической обработки данных. В итоге класс «FistVpnConnection» выполняет основную роль в обеспечении информационного обмена с VPN-сервером с поддержкой криптографической обработки данных и работы в скрытом режиме, что удовлетворяет техническим требованиям на ВКР.

В таблице 16 приведено описание данных класса. В таблице 17 приведено описание методов класса.

Текст программы, реализующий класс «FistVpnConnection», приведен в приложении Г.

Таблица 16 – Описание данных класса «FistVpnConnection»

Имя переменной	Тип переменной	Описание
MAX_PACKET_SIZE	int	Хранит максимальный размер сетевых пакетов
KEEPALIVE_INTERVAL_MS	long	Хранит интервал времени по истечении, которого, если Клиент не отправлял данные VPN-серверу, СПО-С отправляет «Служебный UDP-пакет»
RECEIVE_TIMEOUT_	long	Хранит интервал времени по

Продолжение таблицы 16

Имя переменной	Тип переменной	Описание
MS		истечении, которого, если VPN-сервер не отправлял данные Клиенту, СПО-С прекращает работу
IDLE_INTERVAL_MS	long	Хранит время в течении которого сервис VPN-соединения бездействует
MAX_HANDSHAKE_ATTEMPTS	int	Хранит максимальное число исходящих пакетов с аутентифицирующими данными
mService	VpnService	Экземпляр класса VpnService
mConnectionId	int	id устанавливаемого соединения
mServerName	String	Ip-адрес VPN-сервера
mServerPort	int	Сетевой порт VPN-сервера
mSharedSecret	byte[]	Пароль
IMEI	long	Номер IMEI
mConfigureIntent	PendingIntent	Экземпляр класса PendingIntent

Таблица 17 – Описание методов класса «FistVpnConnection»

Имя метода	Модификатор доступа	Описание метода
ToyVpnConnection	Публичный	Конструктор класса «ToyVpnConnection»
setConfigureIntent	Публичный	Метод получения экземпляра класса PendingIntent
run()	Публичный	Реализованный метод класса Runnable
run(SocketAddress server)	Приватный	Метод реализует основную функцию СПО-С

Продолжение таблицы 17

Имя метода	Модификатор доступа	Описание метода
intArrayToByteArray	Приватный	Метод конвертирования массива типа int в массив типа byte
BufToStr	Приватный	Метод конвертирования массива типа byte в строку
handshake	Приватный	Метод реализует процедуру аутентификации Клиента
configure	Приватный	Метод реализует конфигурирование сервиса VPN-соединения, после успешной аутентификации

2.5 Тестирование разработанного специального программного обеспечения серверной и клиентской части модели

При тестировании модели была проверена работоспособность следующих компонентов:

- WEB-сервер:
 - проверка правильности работы на поступившие запросы;
 - проверка поддержки различных WEB-браузеров;
- СПО-С:
 - конфигурирование;
 - проверка синхропосылки;
 - проверка имитовставки;
 - аутентификация Клиента;
 - обработка запроса Клиента;
 - обработка ответов WEB-сервера;

- СПО-К:
 - проверка работы «Игрового» представления;
 - проверка работы «Основного» представления.

Для тестирования работоспособности функции конфигурирования СПО-С были смодулированы следующие события:

- отключен сетевой адаптер;
- отсутствует файл БД «bdClients.txt»;
- файл БД «bdClients.txt» содержит записи о пользователях.

На рисунке 27 показаны снимки экрана с результатами работы СПО-С.

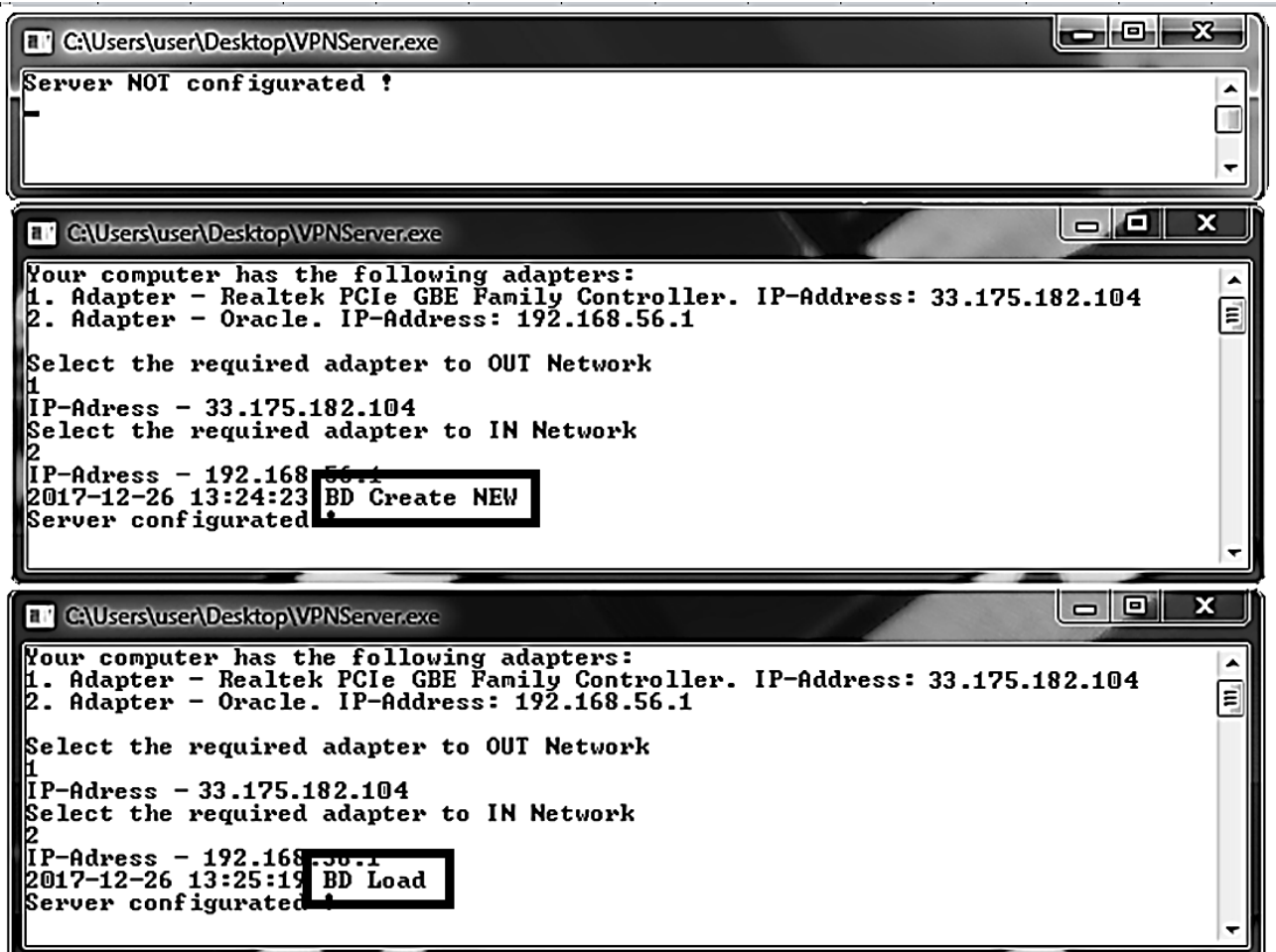


Рисунок 27 – Результаты тестирования конфигурирования СПО-С

По рисунку видно, что если отключены оба сетевых адаптера, приложение VPNServer (СПО-С) выдает сообщение «Server NOT configured!», дальнейшая работа невозможна.

Так же видно, что если СПО-С не обнаружил в корневом каталоге файл с БД Клиентов, он его создает, оповещая об этом сообщением «BDCreateNEW», иначе, если данный файл был найден, выводит сообщение «BDLoad».

На рисунке 28 показан результат тестирования синхропосылки. Клиент совершил попытку аутентификации, но локальное время клиента отличалось от времени VPN-сервера более чем на 1 сутки.

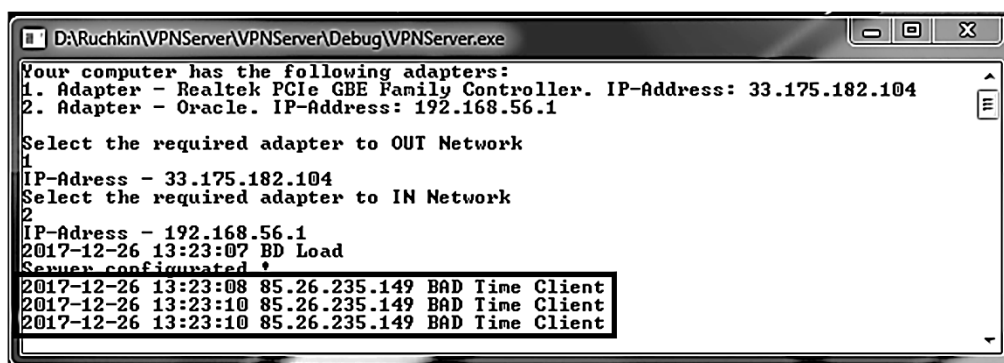


Рисунок 28 – Результат тестирования правильности синхропосылки

На рисунке 29 показан результат тестирования имитовставки. Клиент совершил попытку аутентификации, но в зашифрованные данные было внесено изменение, что привело к ошибке проверки имитовставки. Об ошибке проверки имитовставки СПО-С оповещает сообщением «wrongIMU».

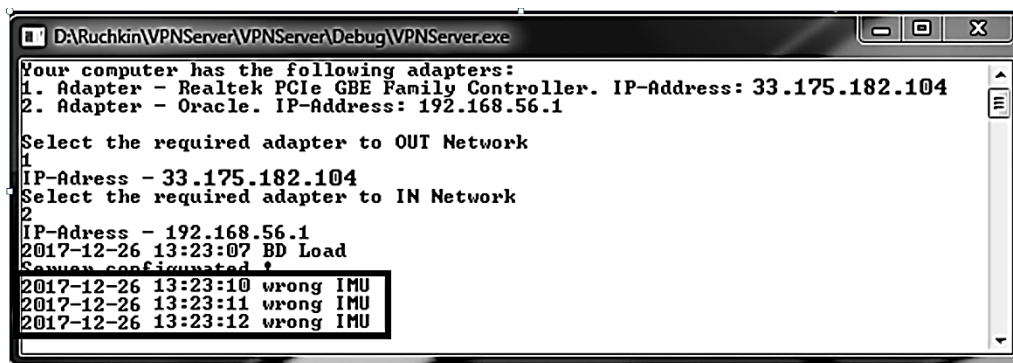


Рисунок 29 – Результат тестирования проверки имитовставки

На рисунке 30 показан результат тестирования аутентификации Клиента. Клиент совершил две попытки аутентификации. В первом случае Клиент аутентифицировался впервые и для него была создана запись в БД, во втором случае запись уже существовала и Криптосервер её обновил. Спустя время Клиент перестает слать «Служебный UDP-Пакет», Криптосервер сбрасывает аутентификацию Клиента.

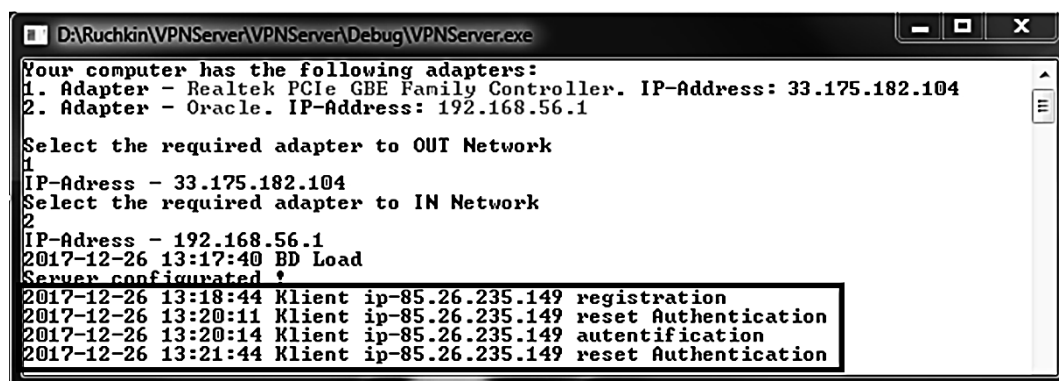


Рисунок 30 – Результат тестирования аутентификации Клиента

На клиентской стороне было запущено приложение с СПО-К, первое, что видит пользователь это «Игровую» активность, интерфейс которой показан на рисунке 23 буквой «А». Далее кнопками было набрано текущее время устройства, после чего запустилась «Основная» активность, интерфейс которой показан на рисунке 23 буквой «Б». После ввода сетевых настроек VPN-сервера и пароля «12345678», была нажата кнопка «Connect», результат данного действия является сообщение СПО-К «VPNisconnecting». Данное сообщение сообщает пользователю, что СПО-К пытается подключиться к VPN-серверу, полученное сообщение показано на рисунке 31.

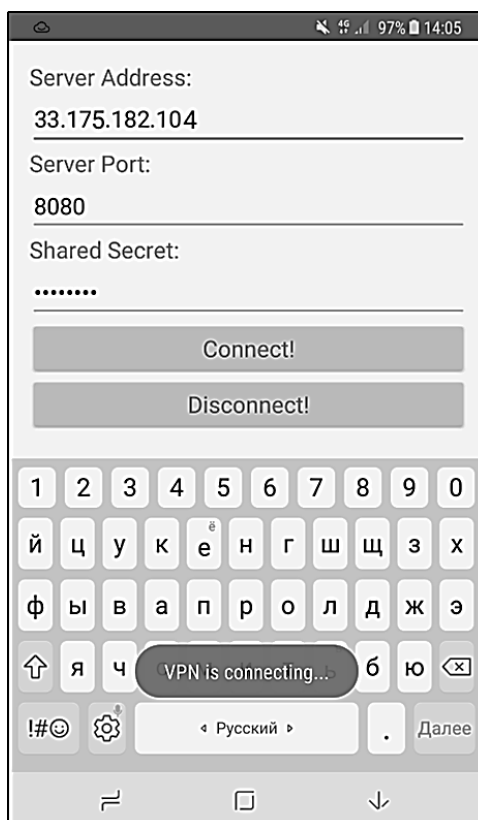


Рисунок 31 – Сообщение СПО-К попытке подключения к VPN-серверу

На рисунке 32 показано сообщение СПО-К «VPNisconnected!». Данное сообщение уведомляет пользователя об успешном прохождении аутентификации на VPN-сервере и готовности СПО-К к обмену данными.

Также в приложении с СПО-С реализованы уведомления в строке состояния, где при активном состоянии VPN-соединения отображается значок ключа.

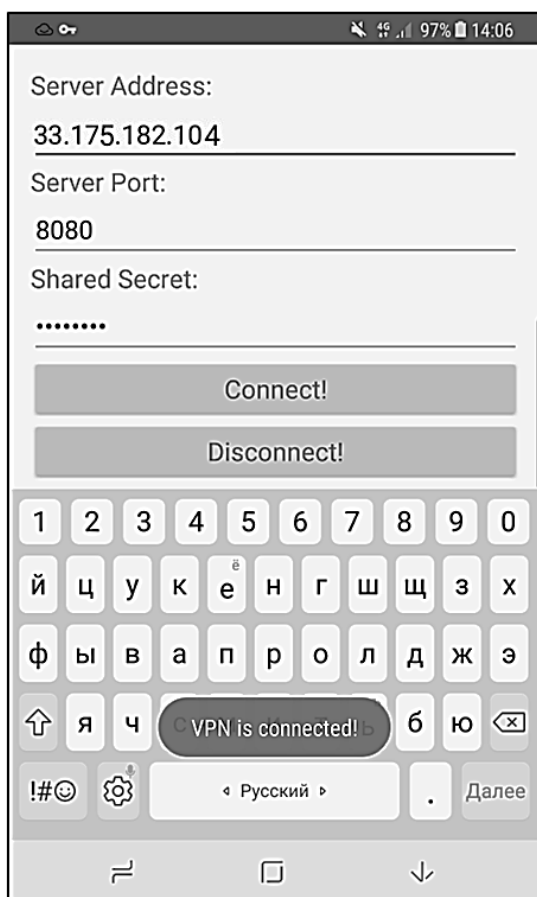


Рисунок 32 - Сообщение СПО-К об успешном подключении к VPN-серверу

На рисунке 33 показано сообщение СПО-К «VPNisdisconnected!». Данное сообщение появляется после нажатия на кнопку «Disconnected» уведомляет пользователя о закрытии VPN-соединения.

На рисунке 34 показано сообщение СПО-К «Passwordincorrect». Данное сообщение уведомляет пользователя, что введенный пароль неверный и аутентификация на VPN-сервере не пройдена.

На рисунке 35 показана загруженная страница по запросу «192.168.56.2:8081/Small/test». Данная страница сформированна сервлетом «Small» и отображает строку «Hello, world» и счетчик обработанных запросов.

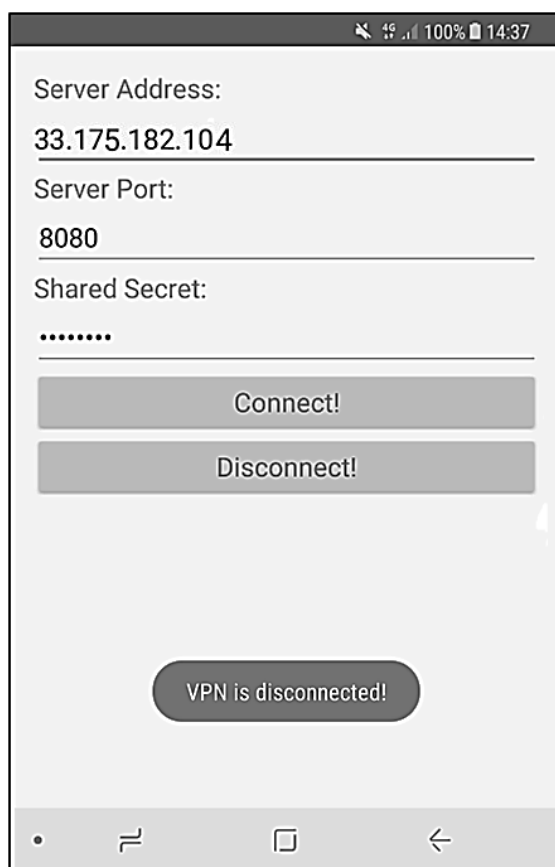


Рисунок 33 - Сообщение СПО-К о закрытии VPN-соединения

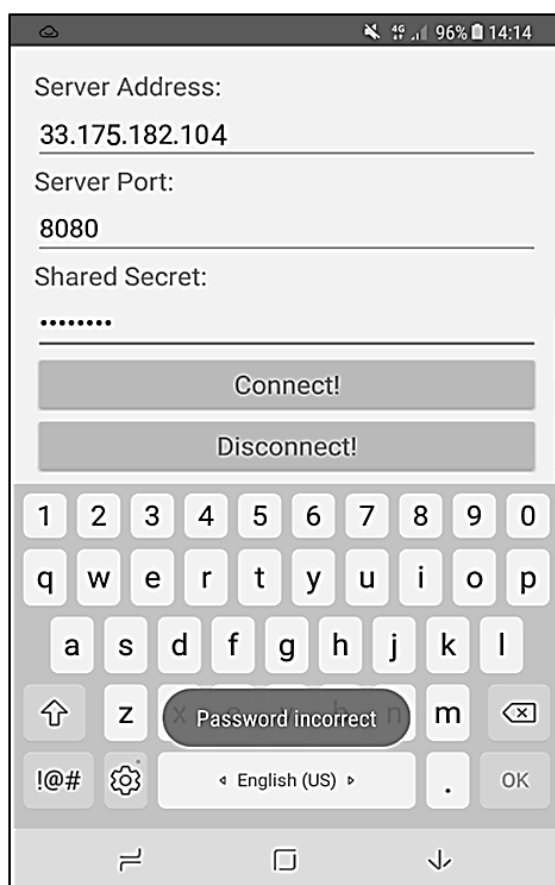


Рисунок 34 - Сообщение СПО-К о неправильном пароле

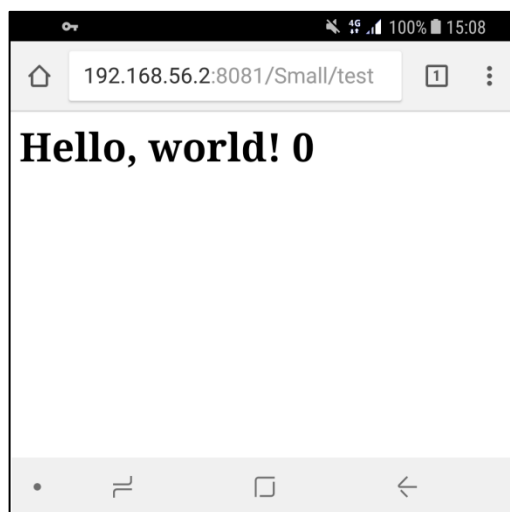


Рисунок 35 - Снимок экрана устройства Клиента с загруженной страницей сервлета «Small»

На рисунке 36 показана загруженная страница по запросу «192.168.56.2:8081/Big/test». Данная страница сформирована сервлетом «Big» и отображает строку «Hello, world», счетчик обработанных запросов и 5120 символов (20 блоков символов с кодами от 0 до 255).

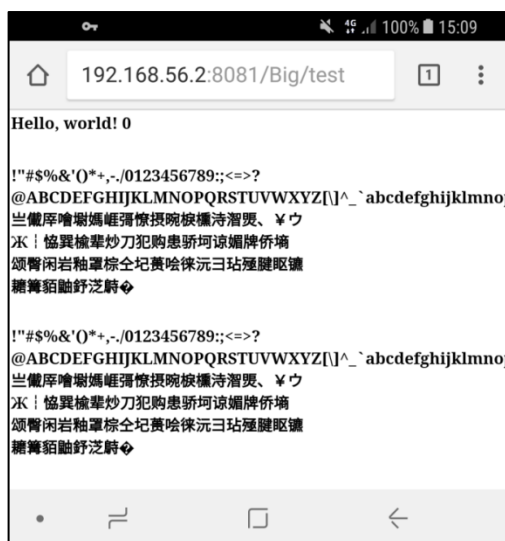


Рисунок 36 - Снимок экрана устройства Клиента с загруженной страницей сервлета «Big»

Также была проверена загрузка файлов «File1Mb.txt» и «File5Mb.txt», осуществив запросы «192.168.56.2:8081/File1Mb/test», «192.168.56.2:8081/File5Mb/test». Данные запросы были обработаны сервлетами «DownloadFile1Mb» и «DownloadFile5Mb» соответственно.

3 Руководство пользователя разработанной модели подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу

Исходные компоненты Модели:

- WEB-ресурс. Данный компонент установлен и настроен на виртуальной машине «WEB-ресурс.vdi»;
- СПО-С. Данный компонент представлен в виде исполняемого файла «VPNServer.exe»;
- СПО-К. Данный компонент представлен в виде установочного файла приложения Android «Fistashki.apk»;
- персональный компьютер с установленным приложением «Oracle VM VirtualBox» версии не ниже 5.2.4, доступ в сеть Интернет, белый IP-адрес;
- мобильное устройство под управлением ОС Android версии не ниже 5.0 с установленной SIM-картой с доступом в сеть Интернет.

Перед началом использования Модели необходимо выполнить следующие действия:

- запустить виртуальную машину «WEB-ресурс.vdi» (далее – Гостевая машина), запустить в ней «Apache Tomcat 8.5» и настроить параметры сетевого адаптера как показано на рисунке 6;
- в приложении «Oracle VM VirtualBox» зайти в настройки Гостевой машины и настроить во вкладке «Сеть» параметр «Тип подключения», выбрав «Виртуальный адаптер хоста»;
- на основной машине настроить сетевые адаптеры в соответствии с рисунком 6 (параметры связанные с белым IP-адресом индивидуальны);
- проверить сетевое соединение Гостевой машины с основной, выполнив команду «ping 192.168.56.2» в командной строке основной машины;
- установить приложение «Fistashki.apk» на мобильное устройство, предоставив все необходимые разрешения;
- на мобильном устройстве проверить доступ в сеть Интернет.

Для осуществления дистанционного мобильного доступа пользователь должен осуществить следующие действия:

- запустить СПО-С (исполняемый файл «VPNServer.exe»);
- запустить СПО-К (приложение «Fistashki»);
- ввести цифровой пароль, набрав кнопками с цифрами текущее время устройства;
- ввести сетевые параметры VPN-сервера;
- ввести пароль аутентификации;
- нажать на кнопку «Connect». В случае успешного подключения к VPN-серверу, будет выведено соответствующее сообщение;
- запустить WEB-браузер;
- в адресной строке ввести «192.168.56.2:8081/Big/test».

В адресной строке вводится не белый IP-адрес VPN-сервера, а локальный (серый) IP-адрес WEB-сервера. На рисунке 37 показан результат запроса, выполненный без запущенного СПО-К.

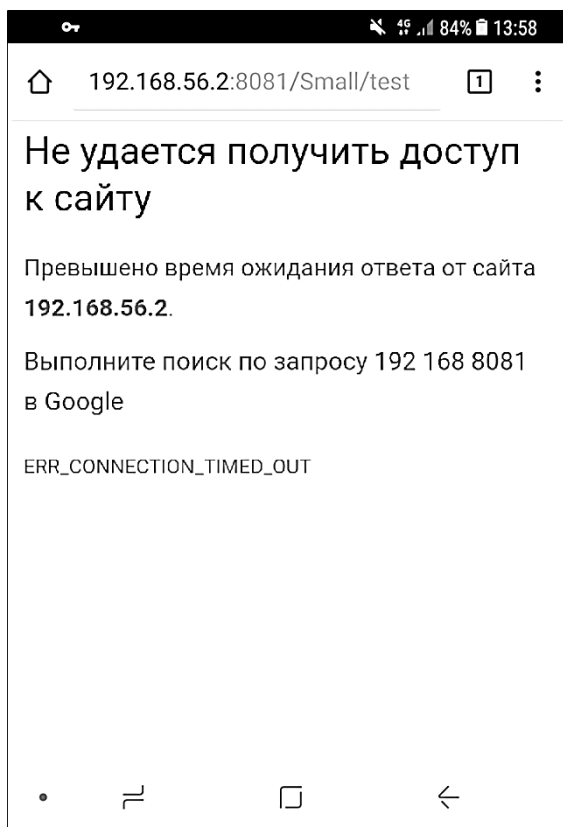


Рисунок 37 – Результат запроса, выполненный без запущенного СПО-К

В случае успеха будет загружена WEB-страница, как показано на рисунке 36. Доступ к защищенному WEB-ресурсу осуществлен.

На рисунке 38 показан пример получения дистанционного мобильного доступа к защищенному WEB-ресурсу.

Для изменения алгоритма шифрования необходимо внести изменения в модуль «Gost28147».

Для изменения протокола обмена данными между Клиентом и VPN-сервером необходимо внести изменения в модуль «FistVpnConnection» СПО-К и в модуль «VPNServer» СПО-С.

Пример получения дистанционного мобильного доступа к защищенному WEB-ресурсу

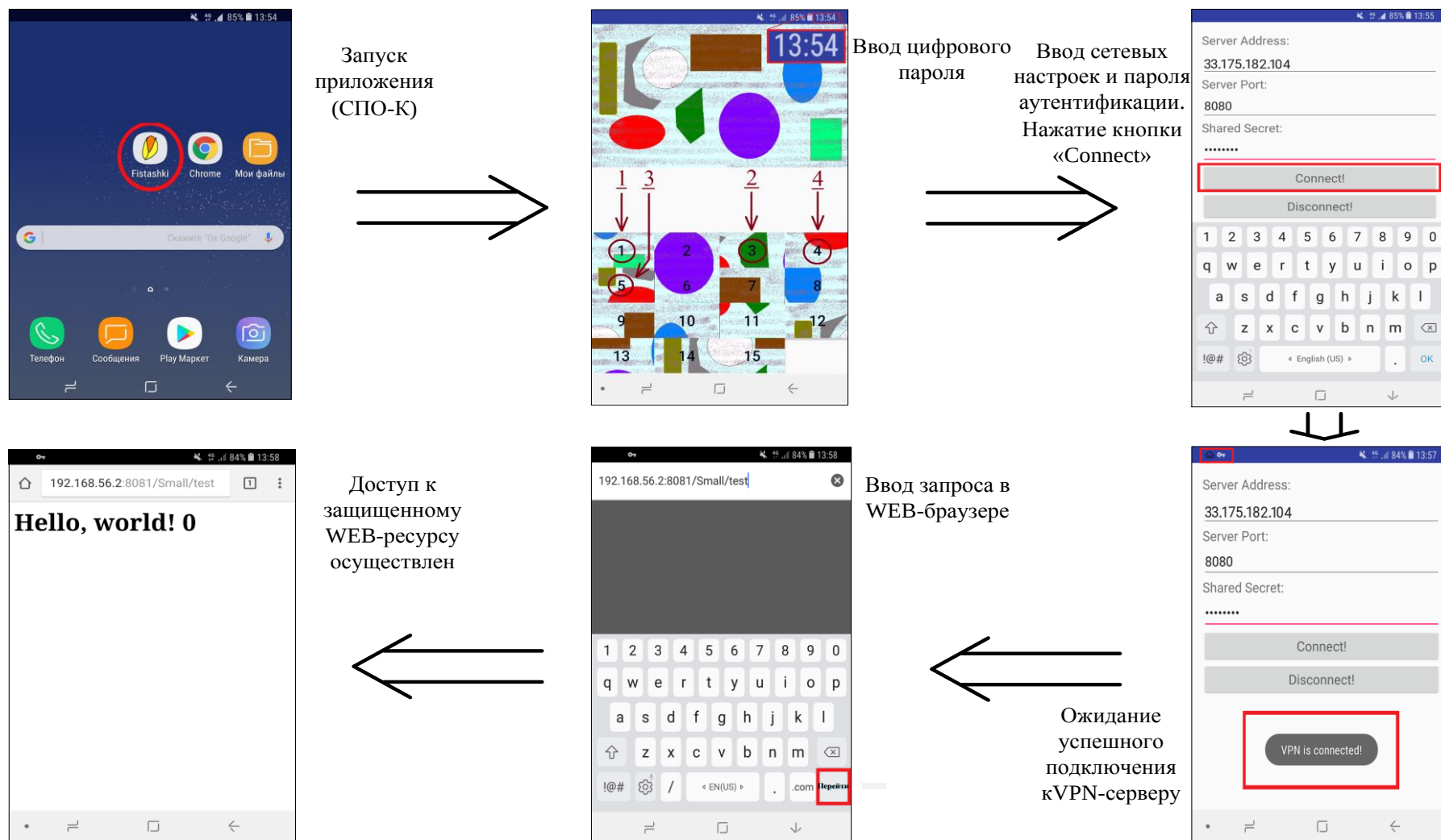


Рисунок 38 - Пример получения дистанционного мобильного доступа к защищенному WEB-ресурсу

4 Анализ и обобщение результатов сравнительных исследований использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу с использованием отечественного криптоалгоритма

Для выполнения сравнительного анализа использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу использовалась разработанная Модель.

Было проведено два вида исследований.

Первый вид исследований заключался в проведении замеров времени загрузки файла с WEB-ресурса размером в 10 Мбайт. Замеры осуществлялись в течение дня с 9:00 до 18:00 с интервалом в 1 час.

Второй вид исследований заключался в проведении замеров времени загрузки с WEB-ресурса файлов объемом в 1, 5, 10 и 50 Мбайт.

Числовые значения полученных результатов приведены в таблице 18 и 19 соответственно для первого и второго исследования. Также результаты исследований представлены в графическом виде на рисунке 39.

Таблица 18 – Результаты первого исследования

Время (чч:мм)	Сеть связи		
	Wi-Fi	4G(LTE)	3G(UMTS)
9:00	26 с	49 с	56 с
10:00	28 с	52 с	56 с
11:00	28 с	54 с	57 с
12:00	29 с	55 с	58 с
13:00	30 с	55 с	58 с
14:00	29 с	56 с	58 с
15:00	30 с	55 с	57 с
16:00	29 с	55 с	58 с
17:00	28 с	54 с	57 с
18:00	27 с	53 с	56 с

Таблица 19 – Результаты второго исследования

Размер файла	Сеть связи		
	Wi-Fi	4G(LTE)	3G(UMTS)
1 Мбайт	4 с	5 с	5 с
5 Мбайт	14 с	25 с	29 с
10 Мбайт	28 с	54 с	57 с
50 Мбайт	132 с	272 с	290 с

По полученным результатам первого исследования видно, что время загрузки файла в начале и конце временного периода меньше, чем в середине. Данный факт свидетельствует о повышении загруженности сетей связи к полудню и снижению к ночному времени суток. Наименьшее время загрузки файла было при подключении мобильного устройства к сети связи Wi-Fi, а наибольшее при подключении к сети связи 3G(UMTS), что соответствует пропускным способностям данных сетей связи.

Пропускная способность сети 3G(UMTS) до 3,6 Мбит/с, 4G(LTE) до 100 Мбит/с, Wi-Fi до 150 Мбит/с.

По полученным результатам второго исследования видно, что наименьшее время загрузки файлов было при подключении мобильного устройства к сети связи Wi-Fi, а наибольшее при подключении к сети связи 3G(UMTS), что соответствует пропускным способностям данных сетей связи.

Обобщая полученные результаты исследований можно сделать вывод, что предпочтительным вариантом для организации дистанционного мобильного доступа являются сети типа Wi-Fi и 4G(LTE). Данные сети можно применять в случаях, когда главным требованием к передаче данных является скорость обмена данными (загрузка файлов, поточная загрузка видео и поточная передача речевой информации). В случаях же когда необходимо осуществить доступ к WEB-ресурсу (загрузить страницу в WEB-браузере) достаточно сети связи типа 3G(UMTS).

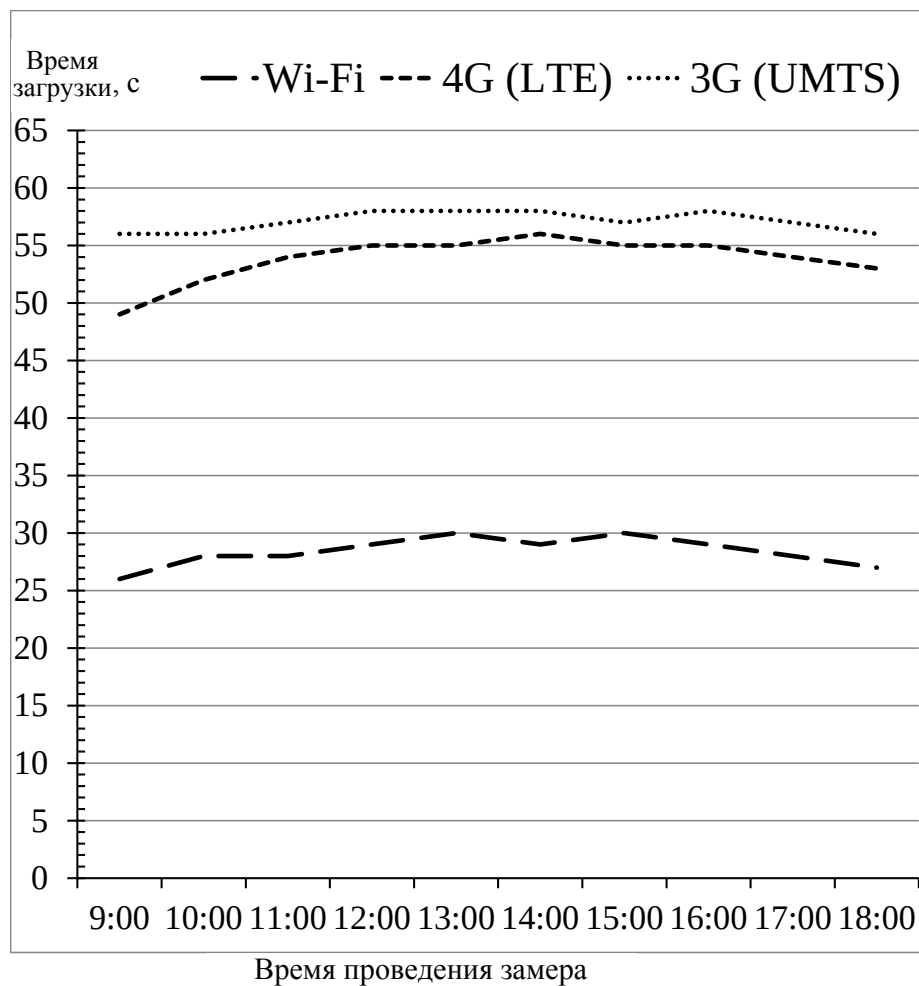
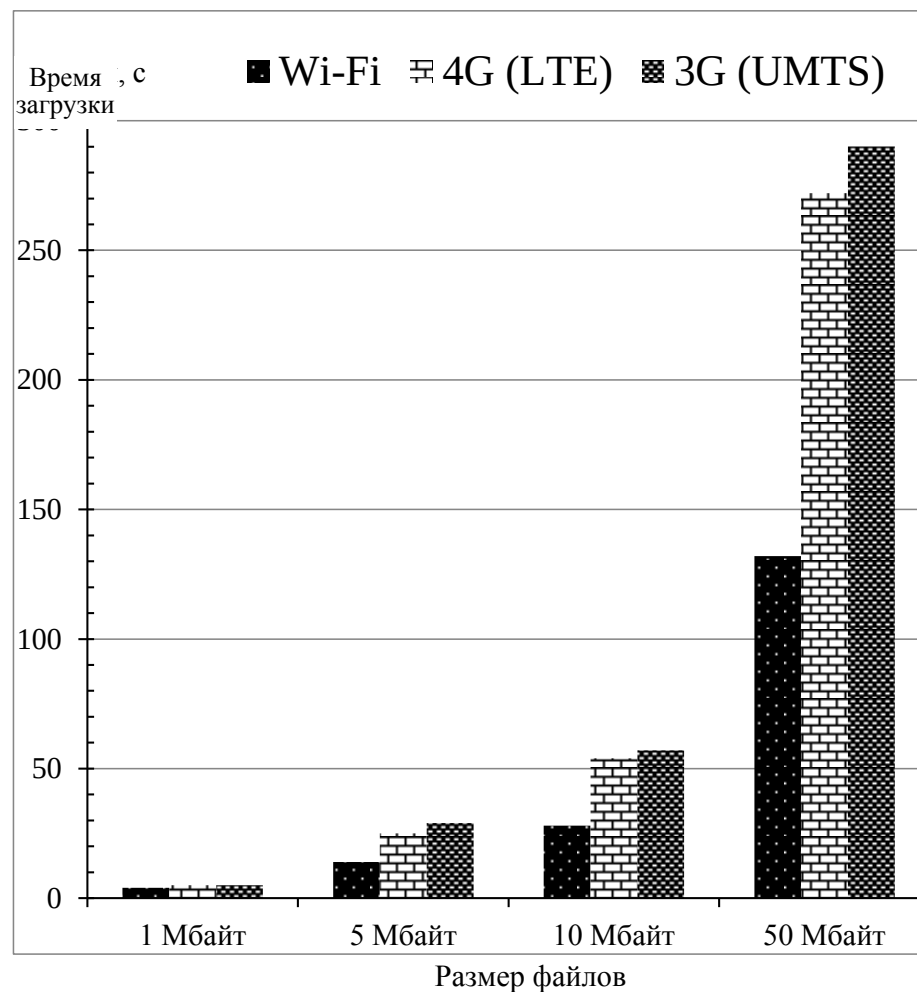


Рисунок 39 - Результаты сравнительных исследований использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу

5 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Эволюционно тело человека предназначено для физических нагрузок. Среди врачей есть образное выражение: «человек приспособлен для того, чтобы мало есть и много бегать». Но многие современные люди работают в офисах и ведут малоподвижный образ жизни. С появлением компьютеров проблема только усугубилась, что негативно сказывается на состоянии здоровья. Страдает опорно-двигательный аппарат, подвергаются значительным изменениям позвоночник и окружающие его мышцы. К этому приводят следующие факторы:

- неподвижная поза. Приводит к ослаблению мышечного корсета позвоночника. Одни мышцы выключены из работы, а другие постоянно напряжены, в итоге может возникать миофасциальный болевой синдром. Длительные статические нагрузки приводят к преждевременному изнашиванию позвонков и межпозвоночных дисков;

- согнутая спина. Если монитор расположен слишком низко, во время работы на компьютере человек вынужден постоянно наклоняться вперед, выгибать спину «колесом». Это усиливает нагрузки на передние края межпозвоночных дисков и может приводить к их протрузиям (выпячиваниям за пределы позвонка), грыжам;

- малоподвижный образ жизни и неполноценное питание. Многие люди, которым приходится подолгу работать за компьютером, перестают следить за своей физической формой и рационом. Это негативно сказывается на состоянии позвоночника и всех суставов.

Так же под влиянием вышеперечисленных факторов развиваются следующие патологические состояния:

- нарушение осанки и сколиоз. Данное патологическое состояние вызвано длительной работой на компьютере. У людей с данной патологией

раньше развивается остеохондроз, повышены риски развития межпозвонковых грыж;

- боль в шее, спине, пояснице, плечах, головная боль. Это может привести к развитию миофасциального синдрома.

Меры профилактики проблем с опорно-двигательным аппаратом, вызванных длительной работой на компьютере:

- использовать специальное офисное кресло. Оно поможет уменьшить нагрузки на позвоночник и плечевой пояс;

- ноги должны стоять на полу, быть согнуты под углом 90 градусов в коленных и тазобедренных суставах. Не класть ногу на ногу;

- держите спину ровно. Отрегулируйте офисное кресло так, чтобы вы могли сидеть с прямой спиной;

- делать перерывы. После каждого часа работы нужно встать и 10 минут походить;

- выполнять упражнения. Малоподвижный образ жизни вреден не только для опорно-двигательного аппарата, но и для сердечно-сосудистой системы и внутренних органов.

Так же к нарушениям опорно-двигательного аппарата относится кистевой туннельный синдром. Это заболевание, которое возникает из-за сдавления одного из нервов, иннервирующих кисть, между мышцами и сухожилиями. Разные виды монотонной работы, во время которых человек вынужден постоянно сгибать и разгибать руку в лучезапястном суставе, в том числе работа на клавиатуре, повышают риск развития синдрома лучезапястного канала. В основном данный синдром затрагивает большой, указательный, средний, в меньшей степени – безымянный палец кисти. Это соответствует зоне иннервации срединного нерва.

Срединный нерв – один из нервов, обеспечивающих чувствительность и движения кисти – проходит в ограниченном пространстве, в так называемом запястном канале, образованном соединительной тканью и костями запястья.

Во время работы на клавиатуре кисти рук постоянно испытывают следующие воздействия:

- постоянные, монотонные сгибания и разгибания в лучезапястных суставах;
- постоянное пребывание в напряжении;
- вибрация во время ударов пальцами по клавиатуре.

В результате этого происходит постепенная травматизация структур запястного канала. В них развивается воспаление и отек. Это приводит к сдавлению срединного нерва и возникновению характерных симптомов:

- в начале заболевания беспокоят неприятные ощущения в кисти: покалывание, жжение, онемение;
- боль – один из основных симптомов. Она тупая, ноющая, может захватывать пальцы, кисть, часть предплечья;
- чувство онемения кистей по утрам;
- постепенно развиваются нарушения со стороны мышц. Человеку становится сложно удерживать пальцами мелкие предметы, они выпадают из рук. Возникают сложности с выполнением точных движений;
- так как нервы регулируют функции сосудов, могут развиваться и сосудистые нарушения. Кисти могут стать бледными или приобрести синюшный оттенок, часто холодеют;
- нарушение иннервации потовых желез приводит к повышенной потливости или сухости кожи;
- нарушение иннервации и кровоснабжения кожи и ногтей приводит к ухудшению их внешнего вида.

Чтобы сохранить здоровье кистей и лучезапястных суставов, работая на компьютере, нужно:

- правильно использовать компьютерную мышь:
- ладонь должна располагаться таким образом, чтобы нижняя часть мышки находилась возле запястья;

- безымянный и большой палец должны находиться немного по бокам и слегка касаться боковых граней мышки;
- указательный палец должен легко лежать на левой кнопке мышки, средний – на правой;
- во время перемещения мышки не совершать сильные движения в лучезапястном суставе – вся рука должна свободно двигаться за мышкой;
- правильное использование клавиатуры:
 - руки должны быть расслаблены, согнуты в локтях под углом 90 градусов и свободно лежать на столе;
 - пальцы должны быть слегка согнуты;
 - кнопок клавиатуры нужно касаться только подушечками, нанося короткие мягкие удары;
 - во время набора текста совершать движения лучше не в лучезапястных суставах, а пальцами;
 - не класть ладони на стол и на клавиатуру;
- периодически делать перерывы, разминать кисти;
- выполнять простые комплексы упражнений. Например, круговые вращательные движения кистей в одну и в другую сторону.

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта была разработана модель подсистемы дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу. Разработанная модель состоит из двух компонентов СПО-К и СПО-С. Для каждого компонента разработан модуль криптографической обработки сетевого трафика.

В процессе выполнения дипломного проекта была изучена технология построения VPN-сетей и проведен обзор существующих программных средств, реализующих технологию VPN для мобильных устройств.

Разработаны и реализованы алгоритмы работы серверной и клиентской части модели.

Также для разработанной модели было написано руководство пользователя и проведен анализ результатов сравнительных исследований использования беспроводных сетей связи типа Wi-Fi, 3G (UMTS), 4G (LTE) для организации дистанционного мобильного доступа абонентов к защищенному WEB-ресурсу с использованием отечественного криптоалгоритма.

Полученная модель может использоваться как в образовательных целях, так и для проведения дальнейших исследований и разработок средств, методов организации дистанционного мобильного доступа к защищенным WEB-ресурсам с использованием открытой незащищенной сети связи Интернет.

Разработанная модель удовлетворяет тактико-техническим требованиям приведенным в ТЗ на ВКР.

Таким образом, задание на дипломную практику выполнено в полном объеме.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

- 1 Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях. — М.: КУДИЦ-ОБРАЗ, 2001. — 368 с.
- 2 Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. — СПб.: Питер, 2001. — 672 с.
- 3 ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Введ. 01.07.90. Изд-во стандартов – 28л.
- 4 Кульгин М. Технологии корпоративных сетей. Энциклопедия. — СПб.: Питер, 2000. — 704 с.
- 5 Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. 2-е изд. — М: Радио и связь, 2002. — 328 с.
- 6 Столлингс В. Основы защиты сетей. Приложения и стандарты = Network Security Essentials. Applications and Standards. — М.: «Вильямс», 2002. — С. 432. — ISBN 0-13-016093-8.

Приложение А
(обязательное)
Текст программы WEB-ресурса

```
//Small.java
//
package com.example;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Small extends HttpServlet {
    long count = 0;
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter pw = resp.getWriter();
        pw.println("<H1>Hello, world! " + count + " </H1>");
        count++;
    }
}
```

```
//Big.java
//
package com.example;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Big extends HttpServlet {
    long count = 0;
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        StringBuilder strB = new StringBuilder();
        for(int i = 0; i < 256; i++) {
            strB.append((char)i);
        }
        String str = strB.toString();
        PrintWriter pw = resp.getWriter();
        pw.println("<H1>Hello, world! " + count + " </H1>");

        for(int i = 0; i < 20; i++) {
            pw.println("<H1>" + str + "</H1>");
        }
        count++;
    }
}
```

```

    }
}

//DownloadFile1Mb.java
//
package com.example;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DownloadFile1Mb extends HttpServlet {

    private static final int BYTES_DOWNLOAD = 1024;
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException{
        response.setContentType("text/plain");
        response.setHeader("Content-Disposition",
            "attachment;filename=File1Mb.txt");
        ServletContext ctx = getServletContext();
        InputStream is = ctx.getResourceAsStream("/File1Mb.txt");

        int read=0;
        byte[] bytes = new byte[BYTES_DOWNLOAD];
        OutputStream os = response.getOutputStream();

        while((read = is.read(bytes))!= -1){
            os.write(bytes, 0, read);
        }
        os.flush();
        os.close();
    }
}

```

```

//DownloadFile5Mb.java
//
package com.example;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```

```

import javax.servlet.http.HttpServletResponse;

public class DownloadFile1Mb extends HttpServlet {
    private static final int BYTES_DOWNLOAD = 1024;
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws IOException{
        response.setContentType("text/plain");
        response.setHeader("Content-Disposition",
            "attachment;filename=File5Mb.txt");
        ServletContext ctx = getServletContext();
        InputStream is = ctx.getResourceAsStream("/File5Mb.txt");

        int read=0;
        byte[] bytes = new byte[BYTES_DOWNLOAD];
        OutputStream os = response.getOutputStream();

        while((read = is.read(bytes))!= -1){
            os.write(bytes, 0, read);
        }
        os.flush();
        os.close();
    }
}

```

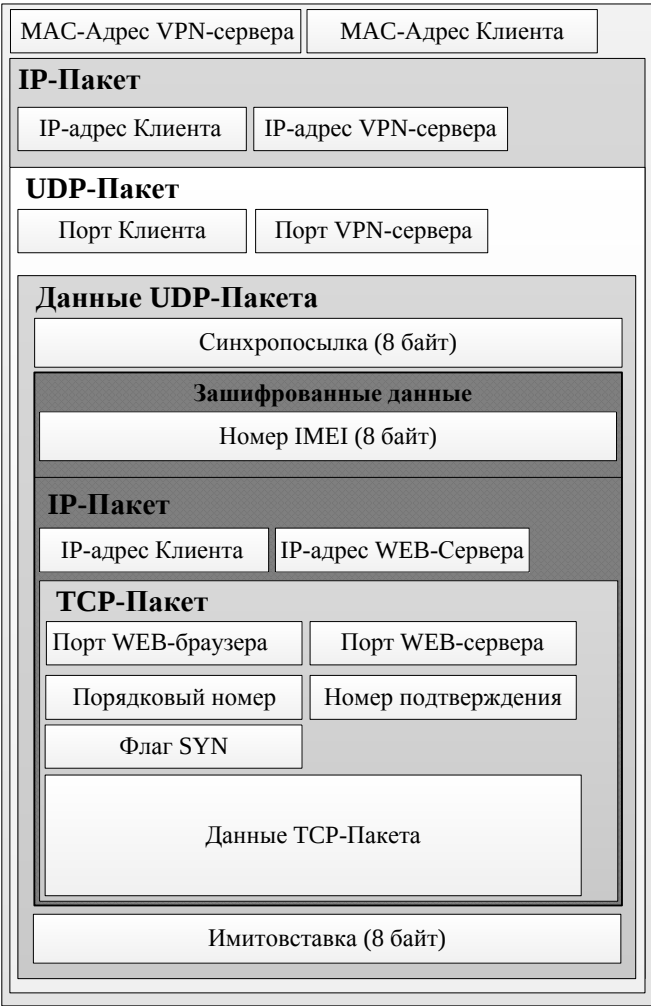
Приложение Б

(обязательное)

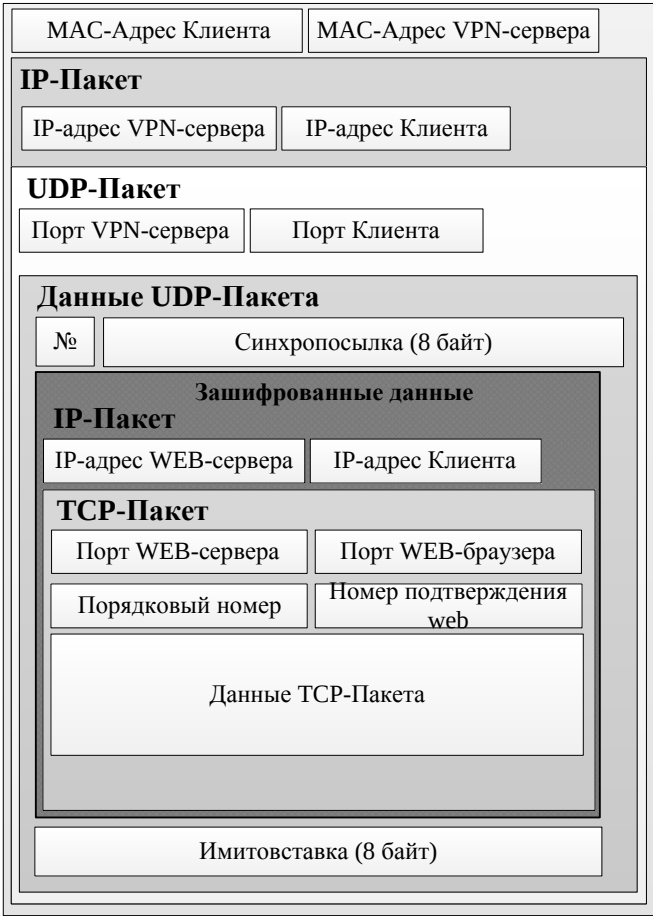
Формат сетевого пакета с зашифрованными данными

Формат сетевого пакета с зашифрованными данными

от Клиента



от VPN-сервера



ПриложениеВ
(обязательное)
Текст программы СПО-С

Текст программы модуля «Gost28147»:

```
//Gost28147.h
#ifndef _GOST28147_H_
#define _GOST28147_H_
classGost28147 {
private:
    unsignedlong * Sp;    //указатель на начальное заполнение (синхропосылка)
    unsignedlong * Key; //указатель на ключ (256 бит)
    unsignedchar * Tz;    //указатель на таблицу подстановок
    unsignedchar * Din; //указатель на исходные данные
    unsignedchar * Dout; //указатель на результат
    unsigned long LenBytes;
    unsigned long D, E, N[2];
    unsigned long M[4];
    short cripted();
    short invstavka();
    void gost_zam_imz();
    void gost_zam_enc();
    void gost_elem(unsigned int n1, unsigned int n2, unsigned long ki);
    void setLenBytes(unsigned long lenBytes);
    void setDin(unsigned char * din);
    void setDout(unsigned char * dout);
public:
    Gost28147();
    ~Gost28147();
    short decriptedData(unsigned char * criptedData, unsigned char * decriptedData,
                        unsigned long lenCriptData, unsigned long * SP);
    short criptedData(unsigned char * openData, unsigned char * criptedData,
                    unsigned long lenOpenData, unsigned long * SP);
};
#endif
//Gost28147.cpp
```

```

#include "Gost28147.h"
#include <cstring>
#include <Winsock2.h>
unsigned long Key1[8] = {0x0,0x0,0x0,0x0,0x0,0x0,0x00D9B739,0x714AB34A};
unsigned char Table[8][16] = {
    0xC,0x4,0x6,0x2,0xA,0x5,0xB,0x9,0xE,0x8,0xD,0x7,0x0,0x3,0xF,0x1,
    0x6,0x8,0x2,0x3,0x9,0xA,0x5,0xC,0x1,0xE,0x4,0x7,0xB,0xD,0x0,0xF,
    0xB,0x3,0x5,0x8,0x2,0xF,0xA,0xD,0xE,0x1,0x7,0x4,0xC,0x9,0x6,0x0,
    0xC,0x8,0x2,0x1,0xD,0x4,0xF,0x6,0x7,0x0,0xA,0x5,0x3,0xE,0x9,0xB,
    0x7,0xF,0x5,0xA,0x8,0x1,0x6,0xD,0x0,0x9,0x3,0xE,0xB,0x4,0x2,0xC,
    0x5,0xD,0xF,0x6,0x9,0x2,0xC,0xA,0xB,0x7,0x8,0x1,0x4,0x3,0xE,0x0,
    0x8,0xE,0x2,0x5,0x6,0x9,0x1,0xC,0xF,0x4,0xB,0x0,0xD,0xA,0x3,0x7,
    0x1,0x7,0xE,0xD,0x0,0x5,0x8,0x3,0x4,0xF,0xA,0x6,0x9,0xC,0xB,0x2,
};
Gost28147::Gost28147() { Key = Key1; }
Gost28147::~~Gost28147() {}
void Gost28147::setLenBytes(unsigned long lenBytes) { LenBytes = lenBytes; }
void Gost28147::setDin(unsigned char * din) { Din = din; }
void Gost28147::setDout(unsigned char * dout) { Dout = dout; }
short Gost28147::decriptedData(unsigned char * criptedData,
                                unsigned char * decriptedData,
                                unsigned long lenCriptData, unsigned long * SP){
    Din = criptedData; Dout = decriptedData;
    Sp = SP; LenBytes = lenCriptData-4;
    if(!cripted()) { nreturn 0; }
    LenBytes = lenCriptData-4;
    int num0 = 0;
    if(LenBytes % 8 != 0) { num0 = ((LenBytes / 8 + 1) * 8) - LenBytes; }
    unsigned char * buffer = new unsigned char[LenBytes + num0];
    memcpy(buffer, criptedData, LenBytes);
    for(int i = LenBytes; i < LenBytes + num0; i++) { buffer[i] = 0; }
    unsigned char imvst [4];

```

```

    Din = buffer; Dout = imvst;
    LenBytes = LenBytes + num0;
    if(!imvstavka()) { return 0; }
    delete [] buffer;
    LenBytes = lenCriptData-4;
    if((imvst[0] != criptedData[LenBytes]) &&
    (imvst[1] != criptedData[LenBytes + 1]) &&
    (imvst[2] != criptedData[LenBytes + 2]) &&
    (imvst[3] != criptedData[LenBytes + 3])) { return 2; }
    return 1;
}

short Gost28147::criptedData(unsigned char * openData, unsigned char * criptedData,
unsigned long lenOpenData, unsigned long * SP) {
    LenBytes = lenOpenData;
    Sp = SP;
    int num0 = 0;
    if(LenBytes % 8 != 0) { num0 = ((LenBytes / 8 + 1) * 8) - LenBytes; }
    unsigned char * buffer = new unsigned char[LenBytes + num0];
    memcpy(buffer, openData, LenBytes);
    for(int i = LenBytes; i < LenBytes + num0; i++) { buffer[i] = 0; }
    unsigned char imvst [4];
    Din = buffer; Dout = imvst;
    LenBytes = LenBytes + num0;
    if(!imvstavka()) { return 0;}
    delete [] buffer;
    Din = openData; Dout = criptedData;
    LenBytes = lenOpenData;
    if(!cripted()) {return 0; }
    for(int i = lenOpenData, j = 0; i < lenOpenData + 4; i++, j++) {
        criptedData[i] = imvst[j];
    }
    return 1;
}

```

```

}
short Gost28147::cripted() {
    unsigned int k;
    unsigned int count = 0;
    if(LenBytes == 0){ return 0;}
    N[0]=0; N[1]=0;k=0; N[0]=Sp[0]; N[1]=Sp[1];
    gost_zam_enc();
    D=N[0]; E=N[1];
    while(true){
        N[0]=D+=0x01010101;
        N[1]=E=E+0x01010104+
        (((((E>>16)&0xffff)+0x0101)+(((E&0xffff)+0x0104)>>16))>>16);
        gost_zam_enc();
        Dout[count*8 + 0]=(unsigned char)((N[0])^Din[count*8 + 0]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 1]=(unsigned char)((N[0]>>8)^Din[count*8 + 1]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 2]=(unsigned char)((N[0]>>16)^Din[count*8 + 2]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 3]=(unsigned char)((N[0]>>24)^Din[count*8 + 3]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 4]=(unsigned char)((N[1])^Din[count*8 + 4]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 5]=(unsigned char)((N[1]>>8)^Din[count*8 + 5]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 6]=(unsigned char)((N[1]>>16)^Din[count*8 + 6]);
    }
}

```

```

        LenBytes--;
        if(LenBytes == 0){ break; }
        Dout[count*8 + 7]=(unsigned char)((N[1]>>24)^Din[count*8 + 7]);
        LenBytes--;
        if(LenBytes == 0){ break; }
        count ++;
    }
    M[0] = N[0]; M[1] = N[1]; M[2] = D; M[3] = E;
    return 1;
}

short Gost28147::imvstavka() {
    unsigned int k;
    unsigned int count = 0;
    if(LenBytes == 0){ return 0;}
    N[0]=0; N[1]=0;k=0;
    N[0]=0; N[1]=0;
    while(LenBytes > 0) {
        N[0]^=(unsigned long *)(Din + count*8);
        N[1]^=(unsigned long *)(Din + (count*8 + 4));
        gost_zam_imz(); LenBytes -= 8; count ++;
    }
    *(unsigned long *)Dout=N[0];
    M[0] = N[0]; M[1] = N[1];
    return 1;
}

void Gost28147::gost_zam_imz() {
    gost_elem(0,1,Key[0]); gost_elem(1,0,Key[1]); gost_elem(0,1,Key[2]);
    gost_elem(1,0,Key[3]); gost_elem(0,1,Key[4]); gost_elem(1,0,Key[5]);
    gost_elem(0,1,Key[6]); gost_elem(1,0,Key[7]); gost_elem(0,1,Key[0]);
    gost_elem(1,0,Key[1]); gost_elem(0,1,Key[2]); gost_elem(1,0,Key[3]);
    gost_elem(0,1,Key[4]); gost_elem(1,0,Key[5]); gost_elem(0,1,Key[6]);
    gost_elem(1,0,Key[7]);

```

```

}

void Gost28147::gost_zam_enc() {
    long cur;
    gost_elem(0,1,Key[0]); gost_elem(1,0,Key[1]); gost_elem(0,1,Key[2]);
    gost_elem(1,0,Key[3]); gost_elem(0,1,Key[4]); gost_elem(1,0,Key[5]);
    gost_elem(0,1,Key[6]); gost_elem(1,0,Key[7]); gost_elem(0,1,Key[0]);
    gost_elem(1,0,Key[1]); gost_elem(0,1,Key[2]); gost_elem(1,0,Key[3]);
    gost_elem(0,1,Key[4]); gost_elem(1,0,Key[5]); gost_elem(0,1,Key[6]);
    gost_elem(1,0,Key[7]); gost_elem(0,1,Key[0]); gost_elem(1,0,Key[1]);
    gost_elem(0,1,Key[2]);gost_elem(1,0,Key[3]);gost_elem(0,1,Key[4]);
    gost_elem(1,0,Key[5]); gost_elem(0,1,Key[6]); gost_elem(1,0,Key[7]);
    gost_elem(0,1,Key[7]); gost_elem(1,0,Key[6]); gost_elem(0,1,Key[5]);
    gost_elem(1,0,Key[4]); gost_elem(0,1,Key[3]); gost_elem(1,0,Key[2]);
    gost_elem(0,1,Key[1]); gost_elem(1,0,Key[0]);
    cur = N[0]; N[0] = N[1]; N[1] = cur;
}

void Gost28147::gost_elem(unsigned int n1, unsigned int n2,unsigned long ki) {
    unsigned char r[4];
    *(unsigned long *)r=N[n1]+(ki);
    r[0]=(Table[1][(r[0]>>4)&0xf]<<4)|(Table[0][r[0]&0xf]);
    r[1]=(Table[3][(r[1]>>4)&0xf]<<4)|(Table[2][r[1]&0xf]);
    r[2]=(Table[5][(r[2]>>4)&0xf]<<4)|(Table[4][r[2]&0xf]);
    r[3]=(Table[7][(r[3]>>4)&0xf]<<4)|(Table[6][r[3]&0xf]);
    N[n2]^=(* (unsigned long *)r<<11)|((* (unsigned long*)r>>21)&0x7FF);}

Текстпрограммымодуля «VPNServer»:

//VPNServer.h
#ifndef H_VPNSEVER
#define H_VPNSEVER
#include<iostream>
#include<string>
#include<iostream>
#include <conio.h>

```

```

#include "PacketMaster.h"

#include "pcap.h"

using namespace std;

#define IP_WEB_SERVER          0xc0a83802 //192.168.56.2
#define IP_HIDDEN_CLIENT       0xc0a80101//192.168.1.1
#define PORT_WEB_SERVER       0x1f91//8081
#define PORT_VPN_SERVER       0x1f90//8080
#define START_PORT_VPN_CLIENT 0x1f91//8081
#define NUM_TCPCONNECT        6
#define LEN_PASSWORD           8
#define SIZE_BDClient          132

extern int IP_VPN_SERVER_OUT;
extern int IP_VPN_SERVER_IN;

// TCP Соединение
typedef struct TCPconnect {
    unsigned short PortHidden;
    unsigned short PortForWEB;
    unsigned long SN;
    unsigned long AN;
    unsigned long ANweb;
    bool SYN;
} TCPConnect;

//База данных подключенных клиентов
typedef struct Clients {
    unsigned long Ip;
    unsigned short Port;
    unsigned long long IMEI;
    unsigned char  MAC[6];
    unsigned long long time;
    bool authentication;
    unsigned char numOldTCPConnect;
    TCPConnect tcpConnect[NUM_TCPCONNECT];
}

```

```

} BDclients;

class VPNServer {
private:
    pcap_if_t * alldevs;
    pcap_if_t * outNet;
    pcap_if_t * inNet;
    char      errbuf[PCAP_ERRBUF_SIZE];
    unsigned char temp[142];
    pcap_t *   outp;
    pcap_t *   inp;
    char *     nameFileDBclients;
    struct pcap_pkthdr *header;
    int        SIZE_DB_CLIENTS;
    int        SIZE_DB_CLIENT;
    BDclients * bdclients;
    longlong   timeUpdateBD; // Время последней проверки БД
    longlong   timeUpdate;   //Время между проверками БД
    longlong   timeChekClient; //Время которое дается клиентам на ответ
    const char * packet_filterOut;
    const char * packet_filterIn;
    struct bpf_program fcodeOut;
    struct bpf_program fcodeIn;
    u_int      netmaskOut;
    u_int      netmaskIn;
    string ipToString(unsigned long ip);
    void updateClientAut(long long time);
    unsigned char * BDCtoBuff(BDclients * bdclients);
    void BufftoBDC(unsigned char * temp, int j);
public:
    VPNServer();
    ~VPNServer();
    bool configurate();

```

```

int receiveData(const unsigned char ** data, unsigned * lenData);
bool sendDataOut(unsigned char * data, unsigned lenData);
bool sendDataIn(unsigned char * data, unsigned lenData);
void writeLog(int numClient, string action);
BDClients * getClient(int numClient);
unsigned short updateTCPConnect(PacketMaster * packetMaster, int numClient);
unsigned short updateUDPConnect(PacketMaster * packetMaster, int numClient);
void registrationClient(PacketMaster * packetMaster, long long time);
void changeSIPMessage(PacketMaster * packetMaster, int numClient);
int searchClientForPortWEB(unsigned long port);
unsigned short searchPortHiden(int numClient, PacketMaster * packetMaster);
int searchClient(unsigned long long IMEI);
void updateTimeClient(int numClient, long long time);
void disconnectClient(int numClient);
};
#endif
//VPNServer.cpp
#include "VPNServer.h"
#include <ctime>
#include <fstream>
int IP_VPN_SERVER_OUT;
int IP_VPN_SERVER_IN;
VPNServer::VPNServer() {
    nameFileDBClients = "bdClients.txt"; //"bdClients"
    SIZE_DB_CLIENTS = 0; SIZE_DB_CLIENT = 142;
    timeUpdate = 30; timeChekClient = 60;
    timeUpdateBD = time(0);}
VPNServer::~~VPNServer() {}
bool VPNServer::configurate() {
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    system("color F0");
    if(pcap_findalldevs(&alldevs, errbuf) == -1) {

```

```

        cout << "Error in pcap_findalldevs: " << errbuf << endl;
        return false;}
if(alldevs == NULL) { return false; }
cout << "Your computer has the following adapters:\n";
int count = 1;
for(pcap_if_t * a = alldevs; a != NULL; a = a->next) {
    cout << count << ". Adapter - ";
    SetConsoleTextAttribute(hConsole, (WORD)((0xF<<4)|6)); //цвет текста
    cout << a->description ;
    SetConsoleTextAttribute(hConsole, (WORD)((0xF<<4)|0)); //цвет текста
    cout << ". IP-Address: ";
    for(pcap_addr_t * b = a->addresses; b != NULL; b = b->next) {
        if(b->addr->sa_family == AF_INET) {
            SetConsoleTextAttribute(hConsole, (WORD)((0xF<<4)|6));
            cout << inet_ntoa(((struct sockaddr_in *)b->addr)->sin_addr)
            << endl;
            SetConsoleTextAttribute(hConsole, (WORD)((0xF<<4)|0));}
        }
        count++;
    }
    cout << endl;
    cout << "Select the required adapter to OUT Network\n";
    char selectAdapter;
    int i;
    while(1) {
        cin.get(selectAdapter);
        for(i = 1; i < count; i++) {
            if((selectAdapter - '0') == i) { break; }
            if(selectAdapter == 'q' || selectAdapter == 'Q') { return 0; }
        }
        if(i != count) { break;}
    }
}

```

```

i--;
outNet = alldevs;
while(i) { outNet = outNet->next; i--;}
if ((outp = pcap_open_live(outNet->name, 65536, 1, 1, errbuf)) == NULL) {
    cout << "\nUnable to open the adapter. " << outNet->name
    << " is not supported by WinPcap\n";
    return false;
}
char * ipStr;
for(pcap_addr_t * a = outNet->addresses; a != NULL; a = a->next) {
    if(a->addr->sa_family == AF_INET) {
        IP_VPN_SERVER_OUT = htonl(((struct sockaddr_in *)a->addr)-
>sin_addr.S_un.S_addr);
        ipStr = inet_ntoa(((struct sockaddr_in *)a->addr)->sin_addr);
        cout << "IP-Adress - " << ipStr << endl;
        break;
    }
}

string s1 = "dst host "; s1 = s1 + ipStr; s1 = s1 + " and (udp or tcp)";
packet_filterOut = s1.c_str();
if(outNet->addresses != NULL) {
    netmaskOut = ((struct sockaddr_in *)(outNet->addresses->netmask))-
>sin_addr.S_un.S_addr;
}
else { netmaskOut = 0xffffffff; }
if(pcap_compile(outp, &fcodeOut, packet_filterOut, 1, netmaskOut) < 0) {
    cout << "Error1";
    return false;
}
if(pcap_setfilter(outp, &fcodeOut) < 0) {
    cout << "Error2";
}

```

```

        return false;
    }
    cout << "Select the required adapter to IN Network\n";
    while(1) {
        cin.get(selectAdapter);
        for(i = 1; i < count; i++) {
            if((selectAdapter - '0') == i){ break;}
            if(selectAdapter == 'q' || selectAdapter == 'Q') { return 0;}
        }
        if(i != count) { break; }
    }
    i--;
    inNet = alldevs;
    while(i) { inNet = inNet->next; i--; }
    if ((inp = pcap_open_live(inNet->name, 65536, 1, 1, errbuf)) == NULL) {
        cout << "\nUnable to open the adapter. " << inNet->name
        << " is not supported by WinPcap\n";
        return false;
    }
    for(pcap_addr_t * a = inNet->addresses; a != NULL; a = a->next) {
        if(a->addr->sa_family == AF_INET) {
            IP_VPN_SERVER_IN = htonl(((struct sockaddr_in *)a->addr)-
>sin_addr.S_un.S_addr);
            ipStr = inet_ntoa(((struct sockaddr_in *)a->addr)->sin_addr);
            cout << "IP-Adress - " << ipStr << endl;
            break;
        }
    }
    s1 = "dst host "; s1 = s1 + ipStr; s1 = s1 + " and (udp or tcp)";
    packet_filterIn = s1.c_str();
    if(inNet->addresses != NULL) {

```

```

        netmaskIn = ((struct sockaddr_in *)(inNet->addresses->netmask))-
>sin_addr.S_un.S_addr;
    }
    else { netmaskIn = 0xffffffff; }
    if(pcap_compile(inp, &fcodeIn, packet_filterIn, 1, netmaskIn) < 0) {
        cout << "Error1";
        return false;
    }
    if(pcap_setfilter(inp, &fcodeIn) < 0) {
        cout << "Error2";
        return false;
    }
    //Загрузка БД Клиентов
    FILE *f = fopen(nameFileDBClients, "rb");
    if(f != NULL) {
        fread((unsigned char *)&SIZE_DB_CLIENTS,
            sizeof(SIZE_DB_CLIENTS), 1, f);
        bdClients = new BDClients[SIZE_DB_CLIENTS];
        unsigned char * temp2 = (unsigned char *)malloc(SIZE_DB_CLIENT);
        for(int i = 0; i < SIZE_DB_CLIENTS; i++) {
            fread((unsigned char *)temp2, 1, SIZE_DB_CLIENT, f);
            BufftoBDC(temp2, i);
        }
        fclose(f);
        //Сброс аутентификации клиентов
        for(int i = 0; i < SIZE_DB_CLIENTS; i++) {
            bdClients[i].authentication = false;
        }
        writeLog(-1, " BD Load");
    }
    else {
        f = fopen(nameFileDBClients, "wb"); fclose(f);
    }

```

```

        writeLog(-1, " BD Create NEW");
    }
    return true;
}

int VPNServer::receiveData(const unsigned char ** data, unsigned int * lenData) {
    int res;
    char com;
    while(true) {
        if (kbhit()) {
            cin.get(com);
            if(com == 'q') {
                FILE * f = fopen(nameFileDBClients, "wb");
                fwrite((unsigned char *)&SIZE_DB_CLIENTS,
                    sizeof(SIZE_DB_CLIENTS), 1, f);
                for(int i = 0; i < SIZE_DB_CLIENTS; i++) {
                    fwrite((unsigned char *)&BDCtoBuff(&bdClients[i]),
                        SIZE_DB_CLIENT, 1, f);
                }
                fclose(f);
                return -1;}}
        //Обновление БД клиентов, сброс аутентификации
        if((time(0) - timeUpdateBD) > timeUpdate) {
            updateClientAut(time(0)); timeUpdateBD = time(0);
        }
        if((res = pcap_next_ex(outp, &header, data)) >= 0) {
            if(res == 0 || header->caplen <= 14)
                if((res = pcap_next_ex(inp, &header, data)) >= 0) {
                    if(res == 0 || header->caplen <= 14) continue;
                    if(res == -1) { cout << "Error3"; return 0;}
                }
            if(res == -1) { cout << "Error3"; return 0; }
        }
    }
}

```

```

        break;
    }
    header->caplen = 0;
    *lenData = header->len;
    return 1;
}

bool VPNServer::sendDataOut(unsigned char * data, unsigned int lenData) {
    if (pcap_sendpacket(outp, data, lenData) != 0) {
        string str = " \nError sending the packet: "; str += pcap_geterr(outp);
        str += "\n"; writeLog(-1, str);
        return false;
    }
    return true;
}

bool VPNServer::sendDataIn(unsigned char * data, unsigned int lenData) {
    if (pcap_sendpacket(inp, data, lenData) != 0) {
        string str = " \nError sending the packet: "; str += pcap_geterr(inp);
        str += "\n"; writeLog(-1, str);
        return false;
    }
    return true;
}

unsigned short VPNServer::updateTCPConnect(PacketMaster * packetMaster, int
numClient) {
    inti;
    //Если утановлен только флаг SYN
    if( packetMaster->getFlagsTCP() == 0x2) {
        //Поиск 0 в PortHiden
        for(i = 0; i < NUM_TCPCONNECT; i++) {
            if(bdClients[numClient].tcpConnect[i].PortHiden == 0) {
                bdClients[numClient].tcpConnect[i].PortHiden = packetMaster->getSrcPort();
                bdClients[numClient].tcpConnect[i].SN = packetMaster->getSeqNumber();
            }
        }
    }
}

```

```

        bdClients[numClient].tcpConnect[i].AN = packetMaster->getAckNumber();
        bdClients[numClient].tcpConnect[i].PortForWEB =
START_PORT_VPN_CLIENT + (numClient * NUM_TCPCONNECT) + i;
        bdClients[numClient].tcpConnect[i].SYN = true;
        bdClients[numClient].numOldTCPConnect = i;
        return bdClients->tcpConnect[i].PortForWEB;
    }
    else {
        //Если подключение с этого порта уже осуществлялось
        if(bdClients[numClient].tcpConnect[i].PortHiden ==
packetMaster->getSrcPort()) {
            bdClients[numClient].tcpConnect[i].SN = packetMaster->getSeqNumber();
            bdClients[numClient].tcpConnect[i].AN = packetMaster->getAckNumber();
            bdClients[numClient].tcpConnect[i].SYN = true;
            bdClients[numClient].numOldTCPConnect = i;
            return bdClients[numClient].tcpConnect[i].PortForWEB;
        }
    }
    numOldTCPConnect + 1
    i = bdClients[numClient].numOldTCPConnect =
(bdClients[numClient].numOldTCPConnect + 1) % NUM_TCPCONNECT;
    bdClients[numClient].tcpConnect[i].PortHiden = packetMaster->getSrcPort();
    bdClients[numClient].tcpConnect[i].SN = packetMaster->getSeqNumber();
    bdClients[numClient].tcpConnect[i].AN = packetMaster->getAckNumber();
    bdClients[numClient].tcpConnect[i].PortForWEB = 8081 + (numClient * 6) +
i;

    bdClients[numClient].tcpConnect[i].SYN = true;
    bdClients[numClient].numOldTCPConnect = i;
    return bdClients->tcpConnect[i].PortForWEB;
}
//Если установлены другие флаги

```

```

else {
    //ищем порт и обновляем данные
    for(i = 0; i < NUM_TCPCONNECT; i++) {
        if(bdClients[numClient].tcpConnect[i].PortHiden ==
            packetMaster->getSrcPort()) {
            bdClients[numClient].tcpConnect[i].SN = packetMaster->getSeqNumber();
            bdClients[numClient].tcpConnect[i].AN = packetMaster->getAckNumber();
            bdClients[numClient].tcpConnect[i].SYN = false;
            return bdClients[numClient].tcpConnect[i].PortForWEB;
        }
    }
    //Если не найдено, то для этого порта соединение устарело
    return 0;
}
return 0;
}

void VPNServer::registrationClient(PacketMaster * packetMaster, long long time) {
    int lenSP = 8;
    int i;
    FILE *f;
    unsigned long long IMEI = *((unsigned long long *)(packetMaster->getDataUDP() +
        7 + lenSP));
    for(i = 0; i < SIZE_DB_CLIENTS; i++) {
        //Если запись найдена обновляем данные
        if(bdClients[i].IMEI == IMEI) {
            bdClients[i].Ip                = packetMaster->getSrcIp();
            bdClients[i].Port              = packetMaster->getSrcPort();
            bdClients[i].time              = time;
            bdClients[i].authentication    = true;
            bdClients[i].numOldTCPConnect = 0;
            memcpy(bdClients[i].MAC, packetMaster->getSrcMAC(), 6);
            f = fopen(nameFileDBClients, "r+b");

```

```

        fseek(f, (4 + SIZE_DB_CLIENT * i), SEEK_SET);
        fwrite((unsigned char *)BDCtoBuff(&bdClients[i]),
                SIZE_DB_CLIENT, 1, f);
        fclose(f);
        writeLog(i, " authentication");
        break;
    }
}

if(i == SIZE_DB_CLIENTS) {
    SIZE_DB_CLIENTS = SIZE_DB_CLIENTS + 1;
    BDClient * newClient = new BDClient[1];
    memset((unsigned char *)newClient, 0, sizeof(BDClient));
    newClient->Ip = packetMaster->getSrcIp();
    newClient->Port = packetMaster->getSrcPort();
    newClient->IMEI = IMEI;
    newClient->time = time;
    newClient->authentication = true;
    newClient->numOldTCPConnect = 0;
    memcpy(newClient->MAC, packetMaster->getSrcMAC(), 6);

    f = fopen(nameFileDBClients, "r+");
    fwrite((unsigned char *)&SIZE_DB_CLIENTS, sizeof(SIZE_DB_CLIENTS),
           1, f);
    fclose(f);
    f = fopen(nameFileDBClients, "ab");
    fwrite((unsigned char *)BDCtoBuff(newClient), 1, SIZE_DB_CLIENT, f);
    fclose(f); delete [] newClient;
    delete [] bdClients;
    bdClients = new BDClient[SIZE_DB_CLIENTS];
    unsigned char * temp = (unsigned char *)malloc(SIZE_DB_CLIENT);
    f = fopen(nameFileDBClients, "rb");
    fseek(f, 4, SEEK_SET);

```

```

        for(int j = 0; j < SIZE_DB_CLIENTS; j++) {
            fread((unsigned char *)temp, 1, SIZE_DB_CLIENT, f);
            BufftoBDC(temp, j);
        }
        fclose(f);
        writeLog(i, " registration");
    }
}

unsigned char * VPNServer::BDCtoBuff(BDClients * bdClients) {
    *(unsigned long *)temp = bdClients->Ip;
    *(unsigned short *)(temp+4) = bdClients->Port;
    *(unsigned long long *)(temp+6) = bdClients->IMEI;
    *(temp + 14) = bdClients->MAC[0];
    *(temp + 15) = bdClients->MAC[1];
    *(temp + 16) = bdClients->MAC[2];
    *(temp + 17) = bdClients->MAC[3];
    *(temp + 18) = bdClients->MAC[4];
    *(temp + 19) = bdClients->MAC[5];
    *(unsigned long long *)(temp + 20) = bdClients->time;
    *(temp + 28) = bdClients->authentication;
    *(temp + 29) = bdClients->numOldTCPConnect;
    for(int i = 0; i < NUM_TCPCONNECT; i++) {
        *(unsigned short *)(temp + 30 + (i*19)) = bdClients->tcpConnect[i].PortHidden;
        *(unsigned short *)(temp + 32 + (i*19)) = bdClients->tcpConnect[i].PortForWEB;
        *(unsigned long *)(temp + 34 + (i*19)) = bdClients->tcpConnect[i].SN;
        *(unsigned long *)(temp + 38 + (i*19)) = bdClients->tcpConnect[i].AN;
        *(unsigned long *)(temp + 42 + (i*19)) = bdClients->tcpConnect[i].ANweb;
        *(temp + 46 + (i*19)) = bdClients->tcpConnect[i].SYN;
    }
    return temp;
}

void VPNServer::BufftoBDC(unsigned char * temp, int j) {

```

```

    bdClients[j].Ip = *(unsigned long *)temp;
    bdClients[j].Port = *(unsigned short *)(temp+4);
    bdClients[j].IMEI = *(unsigned long long *)(temp+6);
    bdClients[j].MAC[0] = *(temp + 14);
    bdClients[j].MAC[1] = *(temp + 15);
    bdClients[j].MAC[2] = *(temp + 16);
    bdClients[j].MAC[3] = *(temp + 17);
    bdClients[j].MAC[4] = *(temp + 18);
    bdClients[j].MAC[5] = *(temp + 19);
    bdClients[j].time = *(unsigned long long *)(temp + 20);
    bdClients[j].authentication = *(temp + 28);
    bdClients[j].numOldTCPConnect = *(temp + 29);
    for(int i = 0; i < NUM_TCPCONNECT; i++) {
        bdClients[j].tcpConnect[i].PortHiden = *(unsigned short *)(temp + 30 + (i*19));
        bdClients[j].tcpConnect[i].PortForWEB = *(unsigned short *)(temp + 32 + (i*19));
        bdClients[j].tcpConnect[i].SN = *(unsigned long *)(temp + 34 + (i*19));
        bdClients[j].tcpConnect[i].AN = *(unsigned long *)(temp + 38 + (i*19));
        bdClients[j].tcpConnect[i].ANweb = *(unsigned long *)(temp + 42 + (i*19));
        bdClients[j].tcpConnect[i].SYN = *(temp + 46 + (i*19));
    }
}

int VPNServer::searchClientForPortWEB(unsigned long port) {
    unsigned int NumClient;
    if((port >= START_PORT_VPN_CLIENT) &&
        (port < (START_PORT_VPN_CLIENT +
            (SIZE_DB_CLIENTS * NUM_TCPCONNECT)))) {
        NumClient = (port - START_PORT_VPN_CLIENT) / NUM_TCPCONNECT;
        for(int j = 0; j < NUM_TCPCONNECT; j++) {
            if(bdClients[NumClient].tcpConnect[j].PortForWEB == port) {
                if(bdClients[NumClient].authentication) {return NumClient; }
                else {return -1;}
            }
        }
    }
}

```

```

        }
    }
    else {return -1;}
    return -1;
}

unsigned short VPNServer::searchPortHiden(int numClient, PacketMaster * packetMaster)
{
    if(bdClients[numClient].authentication) {
        for(int i = 0; i < NUM_TCPCONNECT; i++) {
            //Если установлен флаг SYN, то AN = SN
            if(bdClients[numClient].tcpConnect[i].SYN) {
                if(packetMaster->getAckNumber() ==
                    (bdClients[numClient].tcpConnect[i].SN + 1)) {
                    return bdClients[numClient].tcpConnect[i].PortHiden;
                }
            }
        }
        else {
            if(packetMaster->getSeqNumber() ==
                bdClients[numClient].tcpConnect[i].AN) {
                bdClients[numClient].tcpConnect[i].ANweb =
                    packetMaster->getAckNumber();
                return bdClients[numClient].tcpConnect[i].PortHiden;
            }
            else if(packetMaster->getAckNumber() ==
                bdClients[numClient].tcpConnect[i].SN) {
                return bdClients[numClient].tcpConnect[i].PortHiden;
            }
        }
        if((packetMaster->getFlagsTCP() & 0x10) == 0x10) {
            if(bdClients[numClient].tcpConnect[i].ANweb ==
                packetMaster->getAckNumber()) {
                return bdClients[numClient].tcpConnect[i].PortHiden;
            }
        }
    }
}

```

```

        return 0;
    }
    int VPNServer::searchClient(unsigned long long IMEI) {
        for(int i = 0; i < SIZE_DB_CLIENTS; i++) {
            if(bdClients[i].IMEI == IMEI) {
                if(bdClients[i].authentication) {return i;}
                else {break; /}
            }
        }
        return -1;
    }
    void VPNServer::updateTimeClient(int numClient, long long time) {
        bdClients[numClient].time = time;
    }
    void VPNServer::disconnectClient(int numClient) {
        bdClients[numClient].authentication = false;
    }
    void VPNServer::updateClientAut(long long time) {
        for(int i = 0; i < SIZE_DB_CLIENTS; i++) {
            if((time - bdClients[i].time) > timeChekClient) {
                if(bdClients[i].authentication) writeLog(i, " reset Authentication");
                bdClients[i].authentication = false;
            }
        }
    }
}

```

Текст программы модуля «PacketMaster»:

```

//PacketMaster.h
#ifndef H_PACKETMASTER
#define H_PACKETMASTER
#include<iostream>
#include<string>

```

```

using namespace std;
class PacketMaster {
private:
    unsigned char * packetData;
    unsigned int lenPacketData;
public:
    PacketMaster(); ~PacketMaster();
    void create(const unsigned char * buffer, unsigned int lenBuffer);
    void setDataToPoint(unsigned char * buffer, int lenBuffer, int point);
    unsigned char * getDataToPoint(int point);
    void setByteToPoint(unsigned char v, int point);
    unsigned long long getIMEI();
    /* Методы для работы с заголовком Ethernet-Пакета */
    unsigned char * getData();
    unsigned int getLen();
    unsigned char * getDstMAC();
    unsigned char * getSrcMAC();
    void setDstMAC(unsigned char * mac);
    void setSrcMAC(unsigned char * mac);

    /* Методы для работы с заголовком IP-Пакета */
    unsigned char * getIpPacket();
    void setIpPacket(unsigned char * ipPacket, unsigned int lenIpPacket);
    unsigned short getLenIP();
    void setLenIP(unsigned short len);
    unsigned char getProtocol();
    void setProtocol(unsigned char protocol);
    unsigned long getSrcIp();
    unsigned long getDstIp();
    void setSrcIp(unsigned long ip);
    void setDstIp(unsigned long ip);
    void calcChekSumIP();

```

```

string scrIpStr();
/* Методы для работы с заголовком UDP-Пакета */
unsigned short getSizeDataUDP();
unsigned char * getDataUDP();
void setDataUDP(unsigned char * buffer, int len);
void setLenUDP(int len);
unsigned short getDstPort();
unsigned short getSrcPort();
void setDstPort(unsigned short port);
void setSrcPort(unsigned short port);
void calcChekSumUDP();
/* Методы для работы с заголовком TCP-Пакета */
unsigned char getFlagsTCP();
void setFlagsTCP(unsigned char flags);
unsigned long getSeqNumber();
unsigned long getAckNumber();
void setSeqNumber(unsigned long number);
void setAckNumber(unsigned long number);
void setDataTCP(unsigned char * buffer, int len);
void calcChekSumTCP();
};
#endif

//PacketMaster.cpp
#include "PacketMaster.h"
#include <Winsock2.h>
#define MAX_SIZE_PACKET 1600
PacketMaster::PacketMaster(){
    packetData = (unsigned char *)malloc(MAX_SIZE_PACKET); lenPacketData = 0;
}
PacketMaster::~~PacketMaster(){ free(packetData);}
void PacketMaster::create(const unsigned char * buffer, unsigned int lenBuffer) {

```

```

        memcpy(packetData, buffer, lenBuffer); lenPacketData = lenBuffer;
    }
    void PacketMaster::setDataToPoint(unsigned char * buffer, int lenBuffer, int point) {
        memcpy(packetData + point, buffer, lenBuffer);
    }
    unsigned char * PacketMaster::getDataToPoint(int point) { return (packetData + point);}
    void PacketMaster::setByteToPoint(unsigned char v, int point) { packetData[point] = v;}
    unsigned long long PacketMaster::getIMEI() {
        return *((unsigned long long *)(packetData + 50));
    }
    /* Методы для работы с заголовком Ethernet-Пакета */
    unsigned char * PacketMaster::getData() { return packetData;}
    unsigned int PacketMaster::getLen() { return lenPacketData;}
    unsigned char * PacketMaster::getDstMAC() { return packetData;}
    unsigned char * PacketMaster::getSrcMAC() { return (packetData + 6);}
    void PacketMaster::setDstMAC(unsigned char * mac) { memcpy(packetData, mac, 6);}
    void PacketMaster::setSrcMAC(unsigned char * mac) { memcpy(packetData + 6, mac, 6);}
    /* Методы для работы с заголовком IP-Пакета */
    unsigned char * PacketMaster::getIpPacket() { return (packetData + 14);}

    void PacketMaster::setIpPacket(unsigned char * ipPacket, unsigned int lenIpPacket) {
        memcpy(packetData + 14, ipPacket, lenIpPacket);
    }
    unsigned short PacketMaster::getLenIP() {
        return htons(*((unsigned short *)(packetData + 16)));
    }
    void PacketMaster::setLenIP(unsigned short len) { len = htons(len + 20);
        memcpy(packetData + 16, &len, 2);
    }
    unsigned char PacketMaster::getProtocol() { return *((unsigned char *)(packetData + 23)); }
    void PacketMaster::setProtocol(unsigned char protocol) {
        memcpy(packetData + 23, &protocol, 1);
    }

```

```

}
unsigned long PacketMaster::getSrcIp() {
    return htonl(*((unsigned long *)(packetData + 26)));
}
unsigned long PacketMaster::getDstIp() {
    return htonl(*((unsigned long *)(packetData + 30)));
}
void PacketMaster::setSrcIp(unsigned long ip) { ip = htonl(ip);
    memcpy(packetData + 26, (unsigned char *)&ip, 4);
}
void PacketMaster::setDstIp(unsigned long ip) { ip = htonl(ip);
    memcpy(packetData + 30, (unsigned char *)&ip, 4);
}
void PacketMaster::calcChekSumIP() {
    packetData[24] = packetData[25] = 0;
    unsigned short * bufferS = (unsigned short *)(packetData + 14);
    unsigned long cksum = 0; int size = (packetData[14] & 0xF) * 4;
    while (size > 1){
        cksum += *bufferS++;
        size -= sizeof(unsigned short);
    }
    if (size){ cksum += *(unsigned char*)bufferS; }
    cksum = (cksum >> 16 ) + (cksum & 0xffff);
    cksum += (cksum >>16); cksum = ~cksum;
    memcpy((packetData + 24), (unsigned char *)&cksum, 2);
}
/* Методы для работы с заголовком UDP-Пакета */
unsigned short PacketMaster::getSizeDataUDP() {
    return ntohs(*((unsigned short *)(packetData + 38)) - 8; //lenDataGramm - 8
}
unsigned char * PacketMaster::getDataUDP() {return (packetData + 42);}
void PacketMaster::setDataUDP(unsigned char * buffer, int len) {

```

```

        memcpy(packetData + 42, buffer, len);
        len = htons(len + 8); memcpy(packetData + 38, (unsigned char *)&len, 2);
    }
    void PacketMaster::setLenUDP(int len) {
        *(unsigned short *)(packetData + 38) = ntohs(len + 8);
    }
    unsigned short PacketMaster::getDstPort() {
        return htons(*(unsigned short *)(packetData + 36));
    }
    unsigned short PacketMaster::getSrcPort() {
        return htons(*(unsigned short *)(packetData + 34));
    }
    void PacketMaster::setDstPort(unsigned short port) {
        port = htons(port); memcpy(packetData + 36, &port, 2);
    }
    void PacketMaster::setSrcPort(unsigned short port) {
        port = htons(port); memcpy(packetData + 34, &port, 2);
    }
    void PacketMaster::calcChekSumUDP() {
        unsigned long cksum = 0;
        unsigned int lenUDP = ntohs(*(unsigned short *)(packetData + 38));
        unsigned int lenDataUDP = lenUDP - 8;
        unsigned char * buffer = (unsigned char*)malloc(12 + lenUDP);
        *(unsigned long *)(buffer) = *(unsigned long *)(packetData + 26);
        *(unsigned long *)(buffer+4) = *(unsigned long *)(packetData + 30);
        *(unsigned char *)(buffer+8) = 0;
        *(unsigned char *)(buffer+9) = 17;
        *(unsigned short *)(buffer+10) = *((unsigned short *)(packetData + 38));
        *(unsigned short *)(buffer+12) = *((unsigned short *)(packetData + 34));
        *(unsigned short *)(buffer+14) = *((unsigned short *)(packetData + 36));
        *(unsigned short *)(buffer+16) = *((unsigned short *)(packetData + 38));
        *(unsigned short *)(buffer+18) = 0;
    }

```

```

memcpy(buffer+20, (packetData + 42), lenDataUDP);
unsigned short * bufferS = (unsigned short *)buffer;
int size = 12 + lenUDP;
while (size > 1){
    cksum += *bufferS++; size -= sizeof(unsigned short);
}
if (size){ cksum += *(unsigned char*)bufferS; }
cksum = (cksum >> 16 ) + (cksum & 0xffff);
cksum += (cksum >>16); cksum = ~cksum;
memcpy((packetData + 40), (unsigned char *)&cksum, 2);
free(buffer); buffer = NULL; bufferS = NULL;
}

/* Методы для работы с заголовком TCP-Пакета */
unsigned char PacketMaster::getFlagsTCP() { return *(packetData + 47) & 0x3F;}
void PacketMaster::setFlagsTCP(unsigned char flags) { *(packetData + 47) = flags;}
unsigned long PacketMaster::getSeqNumber() {
    return htonl(*((unsigned long *) (packetData + 38)));
}
unsigned long PacketMaster::getAckNumber() {
    return htonl(*((unsigned long *) (packetData + 42)));}
void PacketMaster::setSeqNumber(unsigned long number) {
    *((unsigned long *) (packetData + 38)) = htonl(number);
}
void PacketMaster::setAckNumber(unsigned long number) {
    *((unsigned long *) (packetData + 42)) = htonl(number);
}
void PacketMaster::setDataTCP(unsigned char * buffer, int len) {
    memcpy(packetData + 42, buffer, len);
}
void PacketMaster::calcChekSumTCP() {
    unsigned long cksum = 0;

```

```

    unsigned int lenTCP = ntohs(*((unsigned short *)(packetData + 16))) -
    ((* (packetData + 14) & 0xf)*4);

    unsigned int lengtHeader = ((* (packetData + 46) & 0xf0) >> 4) * 4;
    unsigned int lenDataTCP = lenTCP - lengtHeader;
    unsigned char * buffer = (unsigned char*)malloc(12 + lenTCP);

    *((unsigned long *) (buffer))      = *((unsigned long *) (packetData + 26));
    *((unsigned long *) (buffer+4))    = *((unsigned long *) (packetData + 30));
    *((unsigned char *) (buffer+8))    = 0;
    *((unsigned char *) (buffer+9))    = 6;
    *((unsigned short *) (buffer+10))   = htons(lenTCP);

    //заголовок TCP
    *((unsigned short *) (buffer+12))  = *((unsigned short *) (packetData + 34));
    *((unsigned short *) (buffer+14))  = *((unsigned short *) (packetData + 36));
    *((unsigned long *) (buffer+16))   = *((unsigned long *) (packetData + 38));
    *((unsigned long *) (buffer+20))   = *((unsigned long *) (packetData + 42));
    *((unsigned short *) (buffer+24))  = *((unsigned short *) (packetData + 46));
    *((unsigned short *) (buffer+26))  = *((unsigned short *) (packetData + 48));
    *((unsigned short *) (buffer+28))  = 0;
    *((unsigned short *) (buffer+30))  = *((unsigned short *) (packetData + 52));
    memcpy(buffer+32, packetData + 54, lengtHeader - 20);
    memcpy(buffer + 12 + lengtHeader, (packetData + 54) + (lengtHeader - 20),
    lenDataTCP);
    unsigned short * bufferS = (unsigned short *)buffer;
    int size = 12 + lenTCP;
    while (size > 1){
        cksum += *bufferS++; size -= sizeof(unsigned short);
    }
    if (size){ cksum += *(unsigned char*)bufferS;}
    cksum = (cksum >> 16 ) + (cksum & 0xffff); cksum += (cksum >>16);
    cksum = ~cksum;
    memcpy((packetData + 50), (unsigned char *)(&cksum), 2);
    free(buffer); buffer = NULL; bufferS = NULL;

```

```
}
```

Текст программы модуля «Main»:

```
//main.h
#ifndef H_MAIN
#define H_MAIN
#pragma comment(lib, "Ws2_32.lib")
#include "Gost28147.h"
#include "VPNServer.h"
#include "PacketMaster.h"
#include "pcap.h"
#include <string>
#define MAX_LEN_DATA 1472
const char * PASSWORD = "12345678";
const unsigned char autCod[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
const unsigned char response[8] = {0x04, 0x05, 0x06, 0x07, 0x00, 0x01, 0x02, 0x03};
//Заголовки функций
char checkPacket(PacketMaster * packetMaster);
long long sinxroToLong(unsigned char * buffer, int len);
unsigned char MAC_VPN_SERVER_OUT[6] = {0x88, 0xd7, 0xf6, 0x7f, 0x3c, 0x6b};
//Определять программно !!!!
unsigned char MAC_VPN_SERVER_IN[6] = {0x0a, 0x00, 0x27, 0x00, 0x00, 0x0d};
//unsigned char MAC_WEB_SERVER[6] = {0x00, 0x16, 0x3f, 0x60, 0x72, 0x30};
//Наверно просто вводить
unsigned char MAC_WEB_SERVER[6] = {0x08, 0x00, 0x27, 0xb6, 0x04, 0x04};
//Наверно просто вводить
unsigned char MAC_ATS_SERVER[6] = {0x00, 0x0c, 0x29, 0x03, 0x65, 0xd1};
unsigned char verIP[2] = {0x08, 0x00};
#endif

//main.cpp
#include<iostream>
#include"main.h"
```

```

#include <ctime>
using namespace std;
Gost28147 gost;
VPNServer vpnServer;
PacketMaster packetMaster;
int numClient = -1;
int main() {
    if(vpnServer.configurate()) {cout << "Server configured !\n";}
    else {cout << "Server NOT configured !\n";while(1);}
    srand(time(0));
    unsigned short PortHiden = 0; unsigned short PortForWEB = 0;
    unsigned short PortForATS = 0; const unsigned char * receivedData;
    unsigned int lenReceivedData; const int lenSP = 8;
    const int lenIMEI = 8; const int lenIMZ = 4;
    unsigned long * SP; short result;
    unsigned char * buffer; long timeServer, timeClient;
    long secInDay = 86400; int lenData;
    while (true) {
        //Прием данных
        result = vpnServer.receiveData(&receivedData, &lenReceivedData);
        if(result == -1) { cout << "Stop server" << endl; cin.get(); cin.get();
            return 0;
        }
        else if(result == 0) { cout << "Error receive data\n"; cin.get();
            return 0;
        }
        packetMaster.create(receivedData, lenReceivedData);
        //Анализ принятого IP-пакета
        switch(checkPacket(&packetMaster)) {
            case 'n' : { break; }
            //Аутентификация
            case 'a' : {

```

```

string configs;
bool pas;
SP = (unsigned long *)(packetMaster.getDataUDP() + 7);
//Проверка синхропосылки
buffer = packetMaster.getDataUDP() + 7; //buffer - синхропосылка
timeServer = time(0);
timeClient = sinxroToLong(buffer, lenSP);
if(abs(timeServer - timeClient) > secInDay) {
    //Время клиента отличается на сутки
    configs = "t,errorSinxro"; lenData = configs.length(); pas = false;
    vpnServer.writeLog(-1, " " + packetMaster.scrIpStr()
        + " BAD Time Client");
}
else {
    buffer = packetMaster.getDataUDP() + 7 + lenSP;
    lenData = packetMaster.getSizeDataUDP() - 7 - lenSP;
    // Расшифрование
    result = gost.decriptedData(buffer, buffer, lenData, SP);
    if(result == 0) {
        vpnServer.writeLog(-1, " Error decripted"); break;
    }
    else if(result == 2) { vpnServer.writeLog(-1, " Wrong IMV");
        break;
    }
    // Проверка пароля
    pas = true;
    if((lenData - 4 - lenIMEI) == LEN_PASSWORD ) {
        for(int i = lenIMEI, j = 0; j < LEN_PASSWORD; i++, j++) {
            if(buffer[i] != PASSWORD[j]) {pas = false;}
        }
    }
    else {pas = false;}
}

```

```

        if(pas) {
            vpnServer.registrationClient(&packetMaster, timeServer);
            configs = "g,good";
            lenData = configs.length();
        }
        else {
            configs = "p,errorPassword";
            lenData = configs.length();
            vpnServer.writeLog(-1, " " + packetMaster.scrIpStr()
                                + " BAD Password");
        }
    }

    unsigned char * bufConf =
        (unsigned char *)malloc(lenData+lenIMZ+lenSP);
    SP[0] = timeServer & 0xffffffffL;
    SP[1] = (timeServer & 0xffffffff00000000L) >> 32;
    memcpy(bufConf, SP, lenSP);
    memcpy(bufConf + lenSP, configs.c_str(), lenData);
    //Шифруем строку с параметрами соединения
    gost.crippedData(bufConf+lenSP, bufConf+lenSP, lenData, SP);

    packetMaster.setDataUDP(bufConf, lenData + lenIMZ + lenSP);
    packetMaster.setDstPort(packetMaster.getSrcPort());
    packetMaster.setSrcPort(PORT_VPN_SERVER);
    packetMaster.setDstIp(packetMaster.getSrcIp());
    packetMaster.setSrcIp(IP_VPN_SERVER_OUT);
    packetMaster.calcChekSumUDP();
    packetMaster.setProtocol(17);
    packetMaster.setLenIP(lenData + lenIMZ + lenSP + 8);
    packetMaster.calcChekSumIP();
    //Отправка данных в сеть
    packetMaster.setDstMAC(packetMaster.getSrcMAC());

```

```

packetMaster.setSrcMAC(MAC_VPN_SERVER_OUT);
vpnServer.sendDataOut(packetMaster.getData(),
                        packetMaster.getLenIP()+14);

free(bufConf);
break;
    }

//Клиент
case 'c' : {
    SP = (unsigned long *)packetMaster.getDataUDP();
    buffer = packetMaster.getDataUDP(); //buffer - синхропосылка
    timeServer = time(0);
    timeClient = sinxroToLong(buffer, lenSP);
    if(abs(timeServer - timeClient) > secInDay) {break;}
    else {
        buffer = packetMaster.getDataUDP() + lenSP;
        lenData = packetMaster.getSizeDataUDP() - lenSP;
        short result = gost.decriptedData(buffer, buffer, lenData, SP);
        if(result == 0) {vpnServer.writeLog(-1, " Error decripted");
            break;
        }
        else if(result == 2) {vpnServer.writeLog(-1, " Wrong IMV");
            break;
        }
        numClient = vpnServer.searchClient(packetMaster.getIMEI());
        if(numClient != -1) {
            if(packetMaster.getSizeDataUDP() == 28) {
                if(!memcmp(buffer+8, autCod, 8)) {
                    vpnServer.updateTimeClient(numClient, timeServer);
                    //Шифрование
                    SP[0] = timeServer & 0xffffffffL;
                    SP[1] = (timeServer & 0xffffffff00000000L) >> 32;
                    lenData = 8;

```

```

        gost.criptedData(packetMaster.getDataToPoint(42 + lenSP),
            packetMaster.getDataToPoint(42 + lenSP), lenData, SP);
        lenData = lenSP + lenData + lenIMZ;

packetMaster.setDstIp(packetMaster.getSrcIp());
packetMaster.setSrcIp(IP_VPN_SERVER_OUT);
packetMaster.setDstPort(packetMaster.getSrcPort());
packetMaster.setSrcPort(PORT_VPN_SERVER);
packetMaster.calcChekSumUDP();
packetMaster.calcChekSumIP();
packetMaster.setDstMAC(packetMaster.getSrcMAC());
packetMaster.setSrcMAC(MAC_VPN_SERVER_OUT);
vpnServer.sendDataOut(packetMaster.getData(), packetMaster.getLenIP()+14);
    }
    break;
}
else {
    packetMaster.setIpPacket(buffer+8, lenData);
    unsigned long dstPort = packetMaster.getDstIp();
    if((packetMaster.getDstIp() == IP_WEB_SERVER)
        && (packetMaster.getProtocol() == 6)) {
        if((PortForWEB = vpnServer.updateTCPConnect(&packetMaster,
            numClient)) == 0) {break;}
        packetMaster.setSrcIp(IP_VPN_SERVER_IN);
        packetMaster.setSrcPort(PortForWEB);
        packetMaster.calcChekSumTCP();
        packetMaster.calcChekSumIP();

        packetMaster.setDstMAC(MAC_WEB_SERVER);
        packetMaster.setSrcMAC(MAC_VPN_SERVER_IN);
        vpnServer.sendDataIn(packetMaster.getData(), packetMaster.getLenIP()+14);
    }
    break;
}
}

```

```

        }
    }
    break;
}

//WEB-Сервер
case 'w': {
    PortHiden = vpnServer.searchPortHiden(numClient, &packetMaster);
    if(PortHiden == 0) {
        vpnServer.writeLog(-1, " Soedinenie ystarelo\n");
        packetMaster.setDstIp(IP_WEB_SERVER);
        packetMaster.setSrcIp(IP_VPN_SERVER_IN);
        packetMaster.setSrcPort(packetMaster.getDstPort());
        packetMaster.setDstPort(PORT_WEB_SERVER);
        packetMaster.setFlagsTCP(4); //RST
        packetMaster.setSeqNumber(packetMaster.getAckNumber());
        packetMaster.setAckNumber(0x12345678);
        packetMaster.calcChekSumTCP();
        packetMaster.calcChekSumIP();
        packetMaster.setDstMAC(MAC_WEB_SERVER);
        packetMaster.setSrcMAC(MAC_VPN_SERVER_IN);
        vpnServer.sendDataIn(packetMaster.getData(),
                             packetMaster.getLenIP()+14);
        break;
    }
    packetMaster.setDstIp(IP_HIDEN_CLIENT);
    packetMaster.setDstPort(PortHiden);
    packetMaster.calcChekSumTCP();
    packetMaster.calcChekSumIP();
    int lenData = packetMaster.getLenIP();
    packetMaster.setDataToPoint(packetMaster.getIpPacket(), lenData,
                                42 + 1 + lenSP);

    //Шифрование

```

```

        SP = (unsigned long *)(packetMaster.getDataUDP() + 1);
        SP[0] = (((rand() + rand()) % 65536) << 16) | ((rand() + rand()) % 65536);
        SP[1] = (((rand() + rand()) % 65536) << 16) | ((rand() + rand()) % 65536);
        gost.cryptedData(packetMaster.getDataToPoint(42 + 1 + lenSP),
            packetMaster.getDataToPoint(42 + 1 + lenSP), lenData, SP);
        lenData = 1 + lenSP + lenData + lenIMZ;
        if(lenData > MAX_LEN_DATA) {
            packetMaster.setByteToPoint(1, 42);
            packetMaster.setSrcPort(PORT_VPN_SERVER);
            packetMaster.setDstPort(vpnServer.getClient(numClient)-
>Port);

            packetMaster.setDstIp(vpnServer.getClient(numClient)->Ip);
            packetMaster.setSrcIp(IP_VPN_SERVER_OUT);
            packetMaster.setProtocol(17);
            packetMaster.setLenUDP(MAX_LEN_DATA);
            packetMaster.calcChekSumUDP();
            packetMaster.setLenIP(MAX_LEN_DATA + 8);
            packetMaster.calcChekSumIP();
            packetMaster.setDstMAC(vpnServer.getClient(numClient)-
>MAC);

            packetMaster.setSrcMAC(MAC_VPN_SERVER_OUT);
            vpnServer.sendDataOut(packetMaster.getData(),
                packetMaster.getLenIP()+14);
            packetMaster.setByteToPoint(2, 42);
            packetMaster.setDataToPoint(packetMaster.getDataUDP() +
                MAX_LEN_DATA, lenData - MAX_LEN_DATA, 42 + 1);
            packetMaster.setLenUDP(lenData - MAX_LEN_DATA + 1);
            packetMaster.calcChekSumUDP();
            packetMaster.setLenIP(lenData - MAX_LEN_DATA + 1 + 8);
            packetMaster.calcChekSumIP();
            vpnServer.sendDataOut(packetMaster.getData(),
                packetMaster.getLenIP()+14);

```

```

    }

    else {
        packetMaster.setByteToPoint(0, 42);
        packetMaster.setSrcPort(PORT_VPN_SERVER);
        packetMaster.setDstPort(vpnServer.getClient(numClient)-
>Port);

        packetMaster.setSrcIp(IP_VPN_SERVER_OUT);
        packetMaster.setDstIp(vpnServer.getClient(numClient)->Ip);
        packetMaster.setLenUDP(lenData);
        packetMaster.setProtocol(17);
        packetMaster.setLenIP(lenData + 8);
        packetMaster.calcChekSumUDP();
        packetMaster.calcChekSumIP();
        packetMaster.setDstMAC(vpnServer.getClient(numClient)-
>MAC);

        packetMaster.setSrcMAC(MAC_VPN_SERVER_OUT);
        vpnServer.sendDataOut(packetMaster.getData(),
                                packetMaster.getLenIP()+14);
    }
    break;
}}}
return 0;
}

long long sinxroToLong(unsigned char * buffer, int len) {
    long long result = 0;
    for(int i = len - 1; i >= 0; i--) {
        result = result << 8; result |= buffer[i];
    }
    return result;
}

char checkPacket(PacketMaster * packetMaster) {
    //UDP

```

```

if(packetMaster->getProtocol() == 17) {
    //Если клиент обращается на 8080 порт VPN-сервера
    if(packetMaster->getDstPort() == PORT_VPN_SERVER) {
        char * buf = (char *)packetMaster->getDataUDP();
        if((buf[0] == 'C' && (buf[1] == 'o')
            && (buf[2] == 'n') && (buf[3] == 'n')
            && (buf[4] == 'e') && (buf[5] == 'c')
            && (buf[6] == 't')) {
                return 'a';
            }
        return 'c';
    }
}
//TCP
else if(packetMaster->getProtocol() == 6) {
    //Проверка от кого пришел пакет
    //WEB-Сервер
    if(packetMaster->getSrcIp() == IP_WEB_SERVER) {
        numClient = vpnServer.searchClientForPortWEB(packetMaster-
>getDstPort());
        if(numClient != -1) { return 'w';}
        else {return 'n';}
    }
    //TCP пакеты от других отправителей отбрасываются
    else {return 'n';}
}
return 'n';
}

```

ПриложениеГ
(обязательное)
Текст программы СПО-К

Текст программы модуля «Gost28147» соответствует тексту программы для СПО-К.

Текст программы модуля «MainActivity»:

```
packageru.dritmiy.ruchkin.fistashki;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.View;
import android.view.animation.AnimationSet;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;
import java.text.SimpleDateFormat;
import static java.lang.Thread.*;
public class MainActivity extends Activity implements View.OnClickListener,
CheckSim.CheckSimCallback{
    int [][] mas = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,0}};
    Button button1, button2, button3, button4;
    Button button5, button6, button7, button8;
    Button button9, button10, button11, button12;
    Button button13, button14, button15;
    ImageView fullImage; boolean flag = true;
    private static int[] clickNumber = {9, 9, 9, 9};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i("INFO", "onCreate() MainActivity");
```

```
button1 = (Button) findViewById(R.id.button1);
button2 = (Button) findViewById(R.id.button2);
button3 = (Button) findViewById(R.id.button3);
button4 = (Button) findViewById(R.id.button4);
button5 = (Button) findViewById(R.id.button5);
button6 = (Button) findViewById(R.id.button6);
button7 = (Button) findViewById(R.id.button7);
button8 = (Button) findViewById(R.id.button8);
button9 = (Button) findViewById(R.id.button9);
button10 = (Button) findViewById(R.id.button10);
button11 = (Button) findViewById(R.id.button11);
button12 = (Button) findViewById(R.id.button12);
button13 = (Button) findViewById(R.id.button13);
button14 = (Button) findViewById(R.id.button14);
button15 = (Button) findViewById(R.id.button15);
fullImage = (ImageView) findViewById(R.id.imageView2);
fullImage.setMinimumWidth(button1.getMinimumWidth()*4);
fullImage.setMinimumHeight(button1.getMinHeight()*4);
button1.setOnClickListener(this);
button2.setOnClickListener(this);
button3.setOnClickListener(this);
button4.setOnClickListener(this);
button5.setOnClickListener(this);
button6.setOnClickListener(this);
button7.setOnClickListener(this);
button8.setOnClickListener(this);
button9.setOnClickListener(this);
button10.setOnClickListener(this);
button11.setOnClickListener(this);
button12.setOnClickListener(this);
button13.setOnClickListener(this);
```

```

        button14.setOnClickListener(this);
        button15.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        int id = Integer.decode(((Button)v).getText().toString());
        move((Button)v,pathSearching(id));
        if(id >= 1 && id <= 10) {
            if(id == 10)id = 0;
            setClickNumber(id);
            authentication((Button)v);
        }
    }
    void move(Button button, int turn) {
        float Xold = button.getX();
        float Yold = button.getY();
        float X = 0;
        float Y = 0;
        switch (turn) {
            case 0: {
                X = button.getX(); Y = button.getY();
                break;
            }
            case 1: {
                Y = button.getY() - button.getHeight(); X = button.getX();
                break;
            }
            case 2: {
                Y = button.getY(); X = button.getX() + button.getWidth();
                break;
            }
            case 3: {

```

```

        Y = button.getY() + button.getHeight(); X = button.getX();
        break;
    }
    case 4: {
        Y = button.getY(); X = button.getX() - button.getWidth();
        break;
    }
    default: {
        X = button.getX(); Y = button.getY();
    }
}

float sdvigX, sdvigY;
sdvigX = (X - Xold)/100; sdvigY = (Y - Yold)/100;
float X1 = Xold, Y1 = Yold;
    button.setY(Y);
    button.setX(X);
}

int pathSearching(int number) {
    int i = 0, j = 0;
    int i1, j1;
    boolean flag = false;
    for(i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            if (mas[i][j] == number) {flag = true; break;}
        }
        if(flag) break;
    }
    i1 = i - 1; j1 = j;
    if(i1 < 0) i1 = 0;
    if(mas[i1][j1] == 0) {mas[i][j] = 0; mas[i1][j1] = number; return 1;}
}

```

```

i1 = i;j1 = j + 1;
if(j1 > 3) j1 = 3;
if(mas[i1][j1] == 0) {mas[i][j] = 0;mas[i1][j1] = number;return 2;}
i1 = i + 1;j1 = j;
if(i1 > 3) i1 = 3;
if(mas[i1][j1] == 0) {
    mas[i][j] = 0;mas[i1][j1] = number; return 3;
}
i1 = i;j1 = j - 1;
if(j1 < 0) j1 = 0;
if(mas[i1][j1] == 0) {mas[i][j] = 0;mas[i1][j1] = number;return 4;}
return 0;
}

private void authentication(Button button) {
    long date = System.currentTimeMillis();
    SimpleDateFormat sdf = new SimpleDateFormat("HHmm");
    char[] dateChar = new char[4];
    sdf.format(date).getChars(0,4,dateChar,0);
    if((clickNumber[0] == (int)(dateChar[0])-'0') &&
        (clickNumber[1] == (int)(dateChar[1])-'0') &&
(clickNumber[2] == (int)(dateChar[2])-'0') &&
        (clickNumber[3] == (int)(dateChar[3])-'0')){
        Toast toast = Toast.makeText(getApplicationContext(),
"Аутентификация пройдена", Toast.LENGTH_SHORT);
        toast.show();
        Intent i = new Intent(MainActivity.this, VPNActivity.class);
        startActivity(i);
    }
}

private void setClickNumber(int number) {
    clickNumber[0] = clickNumber[1];
    clickNumber[1] = clickNumber[2];

```

```

        clickNumber[2] = clickNumber[3];
        clickNumber[3] = number;
    }
}

```

Текст программы модуля «VPNActivity»:

```

//VPNActivity.java
package ru.dritmiy.ruchkin.fistashki;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.VpnService;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
public class VPNActivity extends Activity implements LoadSim.LoadSimCallback{
    public interface Prefs {
        String NAME = "connection";
        String SERVER_ADDRESS = "server.address";
        String SERVER_PORT = "server.port";
        String SHARED_SECRET = "shared.secret";
        String IMEI = "shared.imei";
    }
    Button connect, disconnect;
    private TelephonyManager tManager;
    TextView serverAddress;
    TextView serverPort;
    TextView sharedSecret;
    LoadSim loadSim;

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.vpn_activity);
    Log.i("INFO", "onCreate() VPNActivity");
    serverAddress = (TextView) findViewById(R.id.address);
    serverPort = (TextView) findViewById(R.id.port);
    sharedSecret = (TextView) findViewById(R.id.secret);
    tManager =
(TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);

    final SharedPreferences prefs = getSharedPreferences(Prefs.NAME,
                                                         MODE_PRIVATE);

    serverAddress.setText("");
    serverPort.setText("");
    sharedSecret.setText("");
    connect = (Button) findViewById(R.id.connect);
    disconnect = (Button) findViewById(R.id.disconnect);
    connect.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            prefs.edit()
                .putString(Prefs.SERVER_ADDRESS, serverAddress.getText().toString())
                .putString(Prefs.SERVER_PORT, serverPort.getText().toString())
                .putString(Prefs.SHARED_SECRET, sharedSecret.getText().toString())
                .putString(Prefs.IMEI, tManager.getDeviceId())
                .commit();

            Intent intent = VpnService.prepare(VPNActivity.this);
            if (intent != null) { startActivityForResult(intent, 0);}
            else {onActivityResult(0, RESULT_OK, null);}
        }
    });
}

```

```

        disconnect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
startService(getServiceIntent().setAction(FistVpnService.ACTION_DISCONNECT));
                });
        }
        @Override
        protected void onActivityResult(int request, int result, Intent data) {
            if (result == RESULT_OK) {
                startService(getServiceIntent().setAction(FistVpnService.ACTION_CONNECT));
            }
        }
        private Intent getServiceIntent() {return new Intent(this, FistVpnService.class);}
        @Override
        public void loadParams(final int[] dataVPN) {
            StringBuilder sb = new StringBuilder();
            sb.append(Integer.toString(dataVPN[0]));
sb.append(".");
            sb.append(Integer.toString(dataVPN[1]));
sb.append(".");
            sb.append(Integer.toString(dataVPN[2]));
sb.append(".");
            sb.append(Integer.toString(dataVPN[3]));
            final String addr = sb.toString();
            this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    serverAddress.setText(addr);
                    serverPort.setText(Integer.toString(dataVPN[4]));
                }
            });
        }
    }

```

Текст программы модуля «FistVpnService»:

```
//FistVpnService.java
package ru.dritmiy.ruchkin.fistashki;
import android.app.Notification;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.VpnService;
import android.os.Handler;
import android.os.Message;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import android.util.Pair;
import android.widget.Toast;
import java.io.IOException;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.atomic.AtomicReference;
import ru.dritmiy.ruchkin.fistashki.MainActivity;
import ru.dritmiy.ruchkin.fistashki.ToyVpnConnection;
public class FistVpnService extends VpnService implements Handler.Callback {
    private static final String TAG = FistVpnService.class.getSimpleName();
    public static final String ACTION_CONNECT = "ru.dritmiy.ruchkin.fistashki.START";
    public static final String ACTION_DISCONNECT = "ru.dritmiy.ruchkin.fistashki.STOP";
    private Handler mHandler;
    private ToyVpnConnection toyVpnConnection;
    private static class Connection extends Pair<Thread, ParcelFileDescriptor> {
        public Connection(Thread thread, ParcelFileDescriptor pfd) {super(thread, pfd);}
    }
    private final AtomicReference<Thread> mConnectingThread =
new AtomicReference<>();
    private final AtomicReference<Connection> mConnection = new AtomicReference<>();
```

```

private AtomicInteger mNextConnectionId = new AtomicInteger(1);
private PendingIntent mConfigureIntent;
@Override
public void onCreate() {
    if (mHandler == null) {mHandler = new Handler(this);}
    Log.i("INFO", "onCreate() FistVpnService");
    mConfigureIntent = PendingIntent.getActivity(this, 0, new Intent(this, MainActivity.class),
        PendingIntent.FLAG_UPDATE_CURRENT);
}
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i("INFO", "onStartCommand() FistVpnService");
    if (intent != null && ACTION_DISCONNECT.equals(intent.getAction())) {
        disconnect(); return START_NOT_STICKY;
    } else {
        connect();return START_STICKY;
    }
}
@Override
public void onDestroy() {disconnect();}
@Override
public boolean handleMessage(Message message) {
    Toast.makeText(this, message.what, Toast.LENGTH_SHORT).show();
    if (message.what != R.string.disconnected) {
        updateForegroundNotification(message.what);
    }
    return true;
}
private void connect() {
    updateForegroundNotification(R.string.connecting);
    mHandler.sendMessage(R.string.connecting);
}

```

```

        final SharedPreferences prefs = getSharedPreferences(VPNActivity.Prefs.NAME,
MODE_PRIVATE);

        final String server = prefs.getString(VPNActivity.Prefs.SERVER_ADDRESS, "");
        final byte[] secret = prefs.getString(VPNActivity.Prefs.SHARED_SECRET,
        "").getBytes();

        final long imei = Long.parseLong(prefs.getString(VPNActivity.Prefs.IMEI, ""));
        final int port;
        try {
            port = Integer.parseInt(prefs.getString(VPNActivity.Prefs.SERVER_PORT, ""));
        } catch (NumberFormatException e) {
            Log.e(TAG, "Bad port: " + prefs.getString(VPNActivity.Prefs.SERVER_PORT,
null), e);
            return;
        }
        toyVpnConnection = new ToyVpnConnection(this,
mNextConnectionId.getAndIncrement(), server, port, secret, imei);
        startConnection(toyVpnConnection);
    }

    private void startConnection(final ToyVpnConnection connection) {
        final Thread thread = new Thread(connection, "ToyVpnThread");
        setConnectingThread(thread);
        connection.setConfigureIntent(mConfigureIntent);
        connection.setOnEstablishListener(new ToyVpnConnection.OnEstablishListener() {
            public void onEstablish(ParcelFileDescriptor tunInterface) {
                mHandler.sendMessage(R.string.connected);
                mConnectingThread.compareAndSet(thread, null);
                setConnection(new Connection(thread, tunInterface));
            }
            public void errorPassword() {
                mHandler.sendMessage(R.string.errorPassword);
            }
        });
    }

```

```

    public void errorSinxro() {mHandler.sendMessage(R.string.errorSinxro);}
    public void errorTimeOut(){mHandler.sendMessage(R.string.errorTimeOut);}
    });
    thread.start();
}
private void setConnectingThread(final Thread thread) {
    final Thread oldThread = mConnectingThread.getAndSet(thread);
    if (oldThread != null) {oldThread.interrupt();}
}
private void setConnection(final Connection connection) {
    final Connection oldConnection = mConnection.getAndSet(connection);
    if (oldConnection != null) {
        try {
            oldConnection.first.interrupt();oldConnection.second.close();
        } catch (IOException e) {Log.e(TAG, "Closing VPN interface", e);}
    }
}
private void disconnect() {
    mHandler.sendMessage(R.string.disconnected);
    setConnectingThread(null);
    setConnection(null);
    stopForeground(true);
}
private void updateForegroundNotification(final int message) {
    startForeground(1, new Notification.Builder(this)
        .setSmallIcon(R.drawable.ic_vpn)
        .setContentText(getString(message))
        .setContentIntent(mConfigureIntent)
        .build());
}
}

```

Текст программы модуля «FistVpnConnection»:

```
//FistVpnConnection.java
package ru.dritmiy.ruchkin.fistashki;
import static java.nio.charset.StandardCharsets.US_ASCII;
import android.app.PendingIntent;
import android.net.VpnService;
import android.os.ParcelFileDescriptor;
import android.util.Log;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.SocketAddress;
import java.net.SocketException;
import java.nio.ByteBuffer;
import java.nio.channels.DatagramChannel;
import java.util.concurrent.TimeUnit;
public class ToyVpnConnection implements Runnable {
    public interface OnEstablishListener {
        void onEstablish(ParcelFileDescriptor tunInterface);
        void errorPassword();
        void errorSinxro();
        void errorTimeOut();
    }
    private static final int MAX_PACKET_SIZE = Short.MAX_VALUE;
    private static final long KEEPALIVE_INTERVAL_MS =
TimeUnit.SECONDS.toMillis(15);
    private static final long RECEIVE_TIMEOUT_MS = TimeUnit.SECONDS.toMillis(50);
    private static final long IDLE_INTERVAL_MS =
TimeUnit.MILLISECONDS.toMillis(1);
    private static final int MAX_HANDSHAKE_ATTEMPTS = 50;
```

```

private final VpnService mService;
private final int mConnectionId;
private final String mServerName;
private final int mServerPort;
private final byte[] mSharedSecret;
private final long IMEI;
private PendingIntent mConfigureIntent;
private OnEstablishListener mOnEstablishListener;
public ToyVpnConnection(final VpnService service, final int connectionId,
final String serverName, final int serverPort, final byte[] sharedSecret, final long imei) {
    mService = service;
mConnectionId = connectionId;
    mServerName = serverName;
    mServerPort= serverPort;
    mSharedSecret = sharedSecret;
    IMEI = imei;
}
public void setConfigureIntent(PendingIntent intent) {mConfigureIntent = intent;}
public void setOnEstablishListener(OnEstablishListener listener) {
    mOnEstablishListener = listener;
}
@Override
public void run() {
    try {
        Log.i(getTag(), "Starting");
        final SocketAddress serverAddress = new InetSocketAddress(mServerName,
mServerPort);
        for (int attempt = 0; attempt < 10; ++attempt) {
            if (run(serverAddress)) {attempt = 0;}
            Thread.sleep(3000);
        }
        Log.i(getTag(), "Giving up");
    }
}

```

```

        } catch (IOException | InterruptedException | IllegalStateException |
IllegalArgumentExcepTion e) {
            Log.e(getTag(), "Connection failed, exiting", e);
        }
    }

private boolean run(SocketAddress server)
    throws IOException, InterruptedException, IllegalArgumentExcepTion {
    ParcelFileDescriptor iface = null;
    boolean connected = false;
    try (DatagramChannel tunnel = DatagramChannel.open()) {
        if (!mService.protect(tunnel.socket())) {
            throw new IllegalStateException("Cannot protect the tunnel");
        }
        tunnel.connect(server);
        tunnel.configureBlocking(false);
        iface = handshake(tunnel);
        if (iface == null) {return connected;}
        connected = true;
        FileInputStream in = new FileInputStream(iface.getFileDescriptor());
        FileOutputStream out = new FileOutputStream(iface.getFileDescriptor());
        ByteBuffer packet = ByteBuffer.allocate(MAX_PACKET_SIZE);
        long lastSendTime = System.currentTimeMillis();
        long lastReceiveTime = System.currentTimeMillis();
        byte [] heshBuf = new byte[1600];
        byte imvst [] = new byte [4];
        byte imvstCopy [] = new byte [4];
        byte data [];
        int length;
        int num0;
        int lenSP = 8;
        int lenIMZ = 4;

```

```

int lenIMEI = 8;
long [] SP = new long[2];
long [] SPbig = new long[2];
int maxLenData = 1472;
while (true) {
    boolean idle = true;
    length = in.read(packet.array());
    if (length > 0) {
        Gost28147 gost = new Gost28147();
        num0 = 0;
        length = length + lenIMEI;
        if(length % 8 != 0) {num0 = ((length / 8 + 1) * 8) - length;}
        byte [] buffer = new byte[length + lenIMZ + num0];
        System.arraycopy(packet.array(), 0, buffer, lenIMEI, length-lenIMEI);
        long maska = 0xff;
        for(int i = 0; i < lenIMEI; i++) {
            buffer[i] = (byte)((IMEI & maska) >> (i * 8));
            maska = maska << 8;
        }
        gost.setDin(buffer); gost.setDout(imvst);
        gost.setLenBytes(length + num0); gost.imvstavka();
        long time = System.currentTimeMillis()/1000;
        SP[0] = time & 0xffffffffL; SP[1] = (time & 0xffffffff00000000L) >> 32;
        gost.setDin(buffer); gost.setDout(buffer);
        gost.setLenBytes(length); gost.setSp(SP); gost.cripped();
        for(int i = length, j = 0; i < length + lenIMZ; i++, j++) { buffer[i] = imvst[j];}
        data = new byte[length + lenSP + lenIMZ + num0]; // с запасом
        System.arraycopy(intArrayToByteArray(SP, SP.length), 0, data, 0, lenSP);
        System.arraycopy(buffer, 0, data, lenSP, length + lenIMZ);
        packet.clear();
        packet.put(data)
            .flip();
    }
}

```

```

        packet.limit(lenSP + length + lenIMZ);
        packet.position(0);
        tunnel.write(packet);
        packet.clear();
        idle = false;
    }
    length = tunnel.read(packet);
    if (length > 0) {
        switch(packet.get(0)) {
            case 0 : {
                Gost28147 gost = new Gost28147();
                data = new byte[length - 1];
                System.arraycopy(packet.array(), 1 + lenSP, data, 0, length - (1 + lenSP));
                SP[0] = ((packet.array()[4]&0xff) << 24) | ((packet.array()[3]&0xff) <<
16) | ((packet.array()[2]&0xff) << 8) | (packet.array()[1]&0xff);
                SP[1] = ((packet.array()[8]&0xff) << 24) | ((packet.array()[7]&0xff) <<
16) | ((packet.array()[6]&0xff) << 8) | (packet.array()[5]&0xff);
                length = length - (1 + lenSP + lenIMZ);
                gost.setDin(data);
                gost.setDout(data);
                gost.setLenBytes(length);
                gost.setSp(SP);
                gost.cripted();
            }
        }
        //Проверка имитовставки
        num0 = 0;
        if(length % 8 != 0) {num0 = ((length / 8 + 1)* 8) - length;}
        imvstCopy[0] = data[length];
        imvstCopy[1] = data[length + 1];
        imvstCopy[2] = data[length + 2];
        imvstCopy[3] = data[length + 3];
        for(int i = length; i < length + num0; i++) {
            if(i == data.length) {int a = 0;a = 1;}
        }
    }
}

```

```

        data[i] = 0;
    }
    gost.setDin(data);gost.setDout(imvst);
    gost.setLenBytes(length + num0);gost.imvstavka();
    if((imvst[0] != imvstCopy[0]) && (imvst[1] != imvstCopy[1]) &&
        (imvst[2] != imvstCopy[2]) && (imvst[3] != imvstCopy[3])) {
        Log.i("INFO", "IMvstavka ne sovpala !!!");
    }
    else {out.write(data, 0, length);}
    break;
}
case 1 : {
    System.arraycopy(packet.array(), 1 + lenSP, heshBuf, 0, length - (1 +
lenSP));

    SPbig[0] = ((packet.array()[4]&0xff) << 24) | ((packet.array()[3]&0xff) <<
16) | ((packet.array()[2]&0xff) << 8) | (packet.array()[1]&0xff);
    SPbig[1] = ((packet.array()[8]&0xff) << 24) | ((packet.array()[7]&0xff) <<
16) | ((packet.array()[6]&0xff) << 8) | (packet.array()[5]&0xff);
    break;
}
case 2 : {
    System.arraycopy(packet.array(), 1, heshBuf, maxLenData - (1+lenSP),
length - 1);

    Gost28147 gost = new Gost28147();
    length = length - lenIMZ - 1 + (maxLenData - (1 + lenSP));
    gost.setDin(heshBuf);
    gost.setDout(heshBuf);
    gost.setLenBytes(length);
    gost.setSp(SPbig);
    gost.cripted();
    num0 = 0;
    if(length % 8 != 0) {num0 = ((length / 8 + 1)* 8) - length;}

```

```

        imvstCopy[0] = heshBuf[length];
        imvstCopy[1] = heshBuf[length + 1];
        imvstCopy[2] = heshBuf[length + 2];
        imvstCopy[3] = heshBuf[length + 3];
        for(int i = length; i < length + num0; i++) {heshBuf[i] = 0;}
        gost.setDin(heshBuf);
        gost.setDout(imvst);
        gost.setLenBytes(length + num0);
    gost.imvstavka();
        if((imvst[0] != imvstCopy[0]) && (imvst[1] != imvstCopy[1]) &&
            (imvst[2] != imvstCopy[2]) && (imvst[3] != imvstCopy[3])) {
            Log.i("INFO", "IMvstavka ne sovpala !!!");
        }
    else {out.write(heshBuf, 0, length);}
        break;
    }
    default : {break;}
}
packet.clear();
idle = false;
lastReceiveTime = System.currentTimeMillis();
}
if (idle) {
    Thread.sleep(IDLE_INTERVAL_MS);
    final long timeNow = System.currentTimeMillis();
    if (lastSendTime + KEEPALIVE_INTERVAL_MS <= timeNow) {
        Gost28147 gost = new Gost28147();
        length = 16;num0 = 0;
    if(length % 8 != 0) {num0 = ((length / 8 + 1) * 8) - length;}
        data = new byte[length + num0];

```

```

        long maska = 0xff;
        for(int i = 0; i < lenIMEI; i++) {
            data[i] = (byte)((IMEI & maska) >> (i * 8));maska = maska << 8;
        }
    for(int i = 0; i < 8; i++) {data[i+8] = (byte)i;}

        for(int i = length; i < length + num0; i++) {data[i] = 0;}
    gost.setDin(data);gost.setDout(imvst);
    gost.setLenBytes(length + num0);gost.imvstavka();
    data = new byte[length + 4];
    maska = 0xff;
    for(int i = 0; i < lenIMEI; i++) {data[i] = (byte)((IMEI & maska) >> (i * 8));
        maska = maska << 8;
    }

    for(int i = 0; i < 8; i++) {data[i+8] = (byte)i;}

        long time = System.currentTimeMillis()/1000;
    SP[0] = time & 0xffffffffL;SP[1] = (time & 0xffffffff00000000L) >> 32;
    gost.setDin(data);gost.setDout(data);
    gost.setLenBytes(length);gost.setSp(SP);gost.cripted();
    for(int i = length, j = 0; i < length + 4; i++, j++) {data[i] = imvst[j];}

        packet.put(intArrayToByteArray(SP,SP.length)).put(data).flip();
    packet.position(0);tunnel.write(packet);
    packet.clear();lastSendTime = timeNow;
        } else if (lastReceiveTime + RECEIVE_TIMEOUT_MS <= timeNow) {
            mOnEstablishListener.errorTimeOut();
            throw new IllegalStateException("Timed out");
        }
    }

    }

    } catch (SocketException e) {
        Log.e(getTag(), "Cannot use socket", e);
    } finally {
        if (iface != null) {

```

```

        try {iface.close();}
        catch (IOException e) {Log.e(getTag(), "Unable to close interface", e);}
    }
}
return connected;
}

private byte[] intArrayToByteArray(long [] array, int len) {
    byte [] byteArray = new byte[4 * len];
    for(int i = 0; i < len; i++) {
        byteArray[i*4 + 3] = (byte)((array[i] & 0xFF000000) >> 24);
        byteArray[i*4 + 2] = (byte)((array[i] & 0x00FF0000) >> 16);
        byteArray[i*4 + 1] = (byte)((array[i] & 0x0000FF00) >> 8);
        byteArray[i*4] = (byte)(array[i] & 0x000000FF);
    }
    return byteArray;
}

private String BufToStr(byte[] array, int len) {
    StringBuilder sb = new StringBuilder();
    for(int i = 0; i < len; i++) {sb.append(Integer.toString(array[i]&0xff,16));
        sb.append(" ");
    }
    return sb.toString();
}

private ParcelFileDescriptor handshake(DatagramChannel tunnel)
    throws IOException, InterruptedException {
    ByteBuffer packet = ByteBuffer.allocate(1024);
    Gost28147 gost = new Gost28147();
    byte imvst [] = new byte [4];
    int num0 = 0;int lenIMEI = 8;
    int length = mSharedSecret.length + lenIMEI;
    if(length % 8 != 0) {num0 = ((length / 8 + 1) * 8) - length;}
    byte data [] = new byte[length + num0];

```

```

long maska = 0xff;
for(int i = 0; i < lenIMEI; i++) {data[i] = (byte)((IMEI & maska) >> (i * 8));
    maska = maska << 8;
}
for(int i = lenIMEI; i < length; i++) {data[i] = mSharedSecret[i-lenIMEI];}
for(int i = length; i < length + num0; i++) {data[i] = 0;}
//Расчет имитовставки
gost.setDin(data);gost.setDout(imvst);
gost.setLenBytes(length + num0);gost.imvstavka();
data = new byte[length + 4];maska = 0xff;
for(int i = 0; i < lenIMEI; i++) {data[i] = (byte)((IMEI & maska) >> (i * 8));
    maska = maska << 8;
}
for(int i = lenIMEI; i < length; i++) {data[i] = mSharedSecret[i-lenIMEI];}
long [] SP = new long[2]; long time = System.currentTimeMillis()/1000;
SP[0] = time & 0xffffffffL; SP[1] = (time & 0xffffffff00000000L) >> 32;
gost.setDin(data); gost.setDout(data); gost.setLenBytes(length);
gost.setSp(SP); gost.cripted();
for(int i = length, j = 0; i < length + 4; i++, j++) { data[i] = imvst[j];}
packet.put((byte)'C').put((byte)'o').put((byte)'n').put((byte)'n').put((byte)'e')
    .put((byte)'c').put((byte)'t').put(intArrayToByteArray(SP,SP.length))
    .put(data).flip();
packet.clear();
for (int i = 0; i < MAX_HANDSHAKE_ATTEMPTS; ++i) {
    Thread.sleep(100);
    length = tunnel.read(packet);
    if (length > 0) {
gost = newGost28147();byte imvstCopy [] = new byte [4];
        length = length - 8;data = new byte[length];
        System.arraycopy(packet.array(), 8, data, 0, length);
    }
}

```

```

        SP[0] = ((packet.array()[3]&0xff) << 24) | ((packet.array()[2]&0xff) << 16) |
        ((packet.array()[1]&0xff) << 8) | (packet.array()[0]&0xff);
        SP[1] = ((packet.array()[7]&0xff) << 24) | ((packet.array()[6]&0xff) << 16) |
        ((packet.array()[5]&0xff) << 8) | (packet.array()[4]&0xff);
        length = length - 4; gost.setDin(data); gost.setDout(data); gost.setLenBytes(length);
        gost.setSp(SP); gost.cripted(); num0 = 0;
        if(length % 8 != 0) { num0 = ((length / 8 + 1)* 8) - length; }
        imvstCopy[0] = data[length]; imvstCopy[1] = data[length + 1];
        imvstCopy[2] = data[length + 2]; imvstCopy[3] = data[length + 3];
        for(i = length; i < length + num0; i++) { data[i] = 0; }
        gost.setDin(data); gost.setDout(imvst);
        gost.setLenBytes(length + num0); gost.imvstavka();
        if((imvst[0] != imvstCopy[0]) && (imvst[1] != imvstCopy[1]) &&
            (imvst[2] != imvstCopy[2]) && (imvst[3] != imvstCopy[3])) {
            Log.i("INFO", "IMvstavka ne sovpala !!!");
            return null;
        }
        return configure(new String(data, 0, length, US_ASCII).trim());
    }
}
throw new IOException("Timed out");
}

private ParcelFileDescriptor configure(String parameters) throws
IllegalArgumentException {
    boolean password = true;
    boolean sinxro = true;
    VpnService.Builder builder = mService.new Builder();
    for (String parameter : parameters.split(" ")) {
        String[] fields = parameter.split(",");
        try {
            switch (fields[0].charAt(0)) {
                case 'g':

```

```

        break;
    case 'p':
        password = false;
        break;
    case 't':
        sinxro = false;
        break;
    }
    builder.addAddress("192.168.1.1",24);
    builder.addRoute("0.0.0.0", 0);
} catch (NumberFormatException e) {
    throw new IllegalArgumentException("Bad parameter: " + parameter);
}
}
if(!password) {mOnEstablishListener.errorPassword();return null;}
else if(!sinxro) {mOnEstablishListener.errorSinxro();return null;}
else {
    final ParcelFileDescriptor vpnInterface;
    synchronized (mService) {
        vpnInterface = builder.setSession(mServerName)
            .setConfigureIntent(mConfigureIntent).establish();
        if (mOnEstablishListener != null) {
            mOnEstablishListener.onEstablish(vpnInterface);
        }
    }
    Log.i(getTag(), "New interface: " + vpnInterface + " (" + parameters + ")");
    return vpnInterface;
}
}

private final String getTag() {
    return FistVpnConnection.class.getSimpleName() + "[" + mConnectionId + "];"}

```

Приложение Д
(обязательное)
Протокол проверки ВКР в системе «Антиплагиат»

Ректору Пензенского
государственного университета
А.Д. Гулякову
студента 6 курса ФПИТЭ
очной формы обучения
Ручкина Дмитрия Владимировича

ЗАЯВЛЕНИЕ

Я, Ручкин Дмитрий Владимирович, студент 6 курса очной формы обучения специальности 10.05.02 – «Информационная безопасность телекоммуникационных систем», прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Модель подсистемы дистанционного мобильного доступа абонентов к защищенному web-ресурсу», выполненной мной самостоятельно, на содержание элементов плагиата.

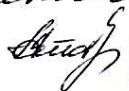
Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

 Д.В. Ручкин

ВКР представлена на проверку 12.01.18

Руководитель ВКР  А.А. Клейменов

Подпись А.А. Клейменова
Инженер 1 кат. СК 



Протокол

проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной квалификационной работы на тему «Модель подсистемы дистанционного мобильного доступа абонентов к защищенному web-ресурсу» студента 6 курса Д.В. Ручкина специальности 10.05.02 «Информационная безопасность телекоммуникационных систем».

Руководитель ВКР начальник НИС АО «ПНИЭИ» Клейменов Алексей Александрович

Отчет о проверке на заимствования

Автор: Ручкин Дмитрий Владимирович dritm1250194@yandex.ru / ID: 644

Проверяющий: Фатеев Алексей Григорьевич afat1@pnzgu.ru / ID: 393

Организация: Пензенский гос. Университет

Отчет предоставлен сервисом «Антиплагиат» - <http://pnzgu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 35
Начало загрузки: 17.01.2018 08:49:07
Длительность загрузки: 00:00:03
Имя исходного файла: Ручкин Дмитрий Владимирович ВКР_12ПТ1_Ручкин.pdf
Размер текста: 2637 кБ
Символов в тексте: 150557
Слов в тексте: 17857
Число предложений: 1190

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 17.01.2018 10:52:32
Длительность проверки: 00:00:16
Комментарии: не указано
Модули поиска: Модуль поиска ЗБС "БиблиоРоссика", Модуль поиска ЗБС "BOOK.ru", Коллекция РГБ, Цитирование, Модуль поиска ЗБС "Университетская библиотека онлайн", Коллекция eLIBRARY.RU, Модуль поиска ЗБС "Айбукс", Модуль поиска Интернет, Модуль поиска ЗБС "Лань", Модуль поиска "ПГУ", Кольцо вузов

ЗАИМСТВОВАНИЯ
7,67% ■

ЦИТИРОВАНИЯ
0%

ОРИГИНАЛЬНОСТЬ
92,33%

17.01.2018

 А.А. Клейменов

Подпись А.А. Клейменова
Инженер 1 кат. ЕВ

