МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра Системы информационной безопасности

«ДОПУЩЕН К ЗАЩИТЕ»
Заведующий кафедрой
_____
(ученая степень, звание, Ф.И.О.)
_____ «___» _____2018 г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Создание ПО в среде разработки RAD Studio для шифрования информации передаваемой по открытым каналам связи

Специальность 5В100200 Системы информационной безопасности

Выполнил(а) Бержнова Сабина Бериковна Группа СИБ-14-2
(Ф.И.О.)

Научный руководитель ст. преп. Якубов Б.М. MoS
(ученая степень, звание, Ф.И.О.)
_____ «29» 05 2018 г.
(подпись)

Рецензент: _____
(ученая степень, звание, Ф.И.О.)
_____ «____» _____2018 г.
(подпись)

Консультанты:
по экономической части:
к.э.н. доцент Салимбаева Р.О.
(ученая степень, звание, Ф.И.О.)
_____ «23» 05 2018 г.
(подпись)

по безопасности жизнедеятельности:
д.т.н. Бекбасаров Ш.Ш.
(ученая степень, звание, Ф.И.О.)
_____ «10» 05 2018 г.
(подпись)

по применению вычислительной техники:
MoS ст. преп. Якубов Б.М.
(ученая степень, звание, Ф.И.О.)
_____ «29» 05 2018 г.
(подпись)

Нормоконтролер: ст. преп. Шыртулова Е.А.
(ученая степень, звание, Ф.И.О.)
_____ «28» мая 2018 г.
(подпись)

Алматы 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра системы информационной безопасности

Специальность системы информационной безопасности

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Бериковой Сабине Бериковне
(Ф.И.О)

Тема проекта Создание ПО в среде разработки RAD Studio для шифрования информации передаваемой по открытым каналам связи

Утверждена приказом по университету № 155 от «23» 10 2017 г.

Срок сдачи законченного проекта «16» 05 2018 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):
Цель проекта: спроектировать систему защиты информации, передаваемой по открытым каналам связи, с применением алгоритма шифрования LEA-128 SEA и алгоритма копирования ключа Base64 в среде разработки RAD Studio Berlin 10.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:
- Выбор алгоритма шифрования
- Выбор алгоритма копирования ключа
- Выбор языка программирования
- Выбор среды разработки программного обеспечения
- Определить и рассчитать категории тяжести и оптимальное

условия труда через интегральную балльную оценку
- Произвести расчет материальных затрат и технико - экономическое обоснование
- Сделать вывод, написать заключение.

Перечень графического материала (с точным указанием обязательных чертежей): Рисунок 7 - Графическая оболочка среды разработки; Рисунок 8, 9 - Схема локальной сети и прохождения трафика по смоделированной в эмуляторе локальной сети; Рисунок 10 - Скорость прохождения трафика по сети; Рисунок 11 - Средняя нагрузка внутри сети; Рисунок 12 - Средняя нагрузка на коммутатор; Рисунок 13 - Нагрузка на Центральный процессор 50% при полной работоспособности ПК; Рисунок 14-24 - Среда разработки AAO Studio Berlin 10; Рисунок 25-35 - создание программного обеспечения; Рисунок 36-53 - тестирование работы программного обеспечения; Рисунок 54 - Нагрузка на центральный процессор при полной работоспособности ПК: 40%.

Основная рекомендуемая литература: 1) Ребко Б.Я., Фионов А.Н. Криптографические методы защиты информации. - Москва: Горячая линия - Телеком, 2005. 2) Сэмбо Мао. Современная криптография: теория и практика. - Вильямс, 2005. - 259-280 с. 3) Якубов Б.М. Защита информации в телекоммуникационных системах. Учебное пособие для ВУЗов. - Нур-Принт, Алматы, 2017. 4) Аманжолова К.Б., Алибаева С.А. Экономика предприятий телекоммуникации: Учебное пособие. - Алматы: АИЭС, 2003. 5) Мазалов И.Ф., Мустафин К.Г., Гонцетко Е.М., Сералиева М.А. Методические указания по выполнению РГР для студентов специальности 5В073100-БИ. - Алматы: АУЭС, 2015. - 38 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
БЖД	ст.т.н. Бейбасаров Ш.Ш	10.04.18 10.05.18	
Экономика	к.э.н. Самигбаева Р.О	14.04.18 23.05.18	
Научный руководитель	Доц Якубов Б.М.	29.05.18	

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Изучение криптографических методов защиты информации	15.01 - 30.01	
Программная реализация работы алгоритмов шифрования	05.02 - 20.02	
Разработка программного обеспечения ASPET Code в среде разработки Embarcadero RAD Studio Berlin 10	07.03 - 01.04	
Тестирование программного обеспечения	02.04 - 07.04	
Технико-экономическое обоснование	08.04 - 20.04	
Безопасность жизнедеятельности	21.04 - 30.04	
Заключение	10.05 - 15.05	

Дата выдачи задания «10» 09 2018 г.

Заведующий кафедрой _____ (Бердибаев Р.Ш.)
(подпись) (Ф.И.О)

Научный руководитель проекта _____ (Лебедев В.Н.)
(подпись) (Ф.И.О)

Задание принял к исполнению студент _____ (Беркинова С.Б.)
(подпись) (Ф.И.О)

Аңдатпа

Дипломдық жобада Delphi 7 бағдарламалау тілінде, Embarcadero RAD Studio Berlin 10-да ашық байланыс арнасы арқылы жіберілетін ақпаратты шифрлеуге арналған бағдарламалық жасақтама құрастырылды. Жобаланған бағдарламалық жасақтама файлдарды шифрлеуге жыне криптоберітілігін арттыру мақсатында кілттерді кодтауға мүмкіндік береді.

Экономика бөлімінде ұсынылатын модельдің мекемеге әкелетін пайдасын және таза табысын есептедім. Тіршілік қауіпсіздігі бөлімінде еңбек қауіпсіздігінің оңтайлы жағдайлары қарастырылды.

Аннотация

В дипломном проекте, спроектировано и скомпилировано на Embarcadero RAD Studio Berlin 10 на языке программирования Delphi 7 программное обеспечение для шифрования информации, передаваемой по открытым каналам связи. Скомпилированное программное обеспечение позволяет шифровать файлы и кодировать ключи для повышения их криптостойкости.

В экономической части были приведены расчеты чистой прибыли и прибыль предприятия в случае внедрения предлагаемой модели. В части безопасности жизнедеятельности были рассмотрены оптимальные условия труда.

Annotation

In the diploma project, the software for encrypting information transmitted over open communication channels was designed and compiled on the RAD Studio Berlin 10 in the Delphi 7 programming language. Compiled software allows you to encrypt files and encrypt keys to increase their cryptographic strength.

In the economic part, calculations were made of the net profit and the profit of the enterprise in the case of the introduction of the proposed model. In terms of life safety, optimal working conditions were considered.

Содержание

Введение

В век скорого развития IT технологий проблема о защите информации занимает основную позицию.

В связи с популяризацией использования различных интернет ресурсов, которые несут информацию о людях, о компаниях и т.д. остро стоит вопрос обеспечения безопасности при хранении, обработке и передаче данной информации. Обеспечение информационной безопасности является главной проблемой сети Интернет, которая представляет собой эффективную, но в то же время непредсказуемую среду, полную различных угроз и опасностей.

Актуальность проблемы сетевой безопасности не подвергается никаким сомнениям.

Имеется немало методов и способов защиты информации и как вариант ее шифрование, для чего применяются специальные программы.

Шифрование представляет собой преобразование данных в нечитабельную форму, используя ключи шифрования и дешифрования. Шифрование обеспечивает конфиденциальность, в тайне храня информацию от третьих лиц, кому она не предназначена.

Цель дипломного проекта – создание безопасной сессии при передаче зашифрованной информации по открытым каналам связи.

1 Криптографические методы защиты информации

Криптология – наука, занимающаяся проблемой обеспечения защищенности информации с помощью ее преобразования. Эта наука, в свою очередь, включает в себя два противоположных по принципу направления: криптографию и криптоанализ.

Принцип работы криптографии заключается в поиске и исследовании математических путей преобразования информации. Криптоанализ же, включает в сферу своих интересов возможности расшифровки информации без обладания знаний о ключах шифрования.

Криптографические методы имеют ряд основных функций, такие как обеспечение передачи конфиденциальной информации по каналам связи (например, электронная почта), установление подлинности передаваемых сообщений, защищенное хранение разного рода информации в зашифрованном виде (например, документы, базы данных) на физических носителях.

Таким образом, криптография предоставляет возможность преобразования информации так, чтобы ее восстановление было возможно только при знании ключа, использованного в процессе шифрования.

Существуют симметричные и ассиметричные криптосистемы.

В процессе симметричного шифрования и для шифрования, и для дешифрования используется один и тот же ключ.

В ассиметричных системах используются открытый и закрытый ключи, связанные друг с другом математически. Информация шифруется с помощью открытого ключа, который находится в общем доступе, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

На сегодняшний день криптография является одним из наиболее используемых способов обеспечения конфиденциальности и подлинности информации. Стандарты шифрования устроены таким способом, что информация о алгоритме шифрования доступна всем, а ключ является секретным. В данном случае как при шифровании, так и при дешифровании необходимо использовать один и тот же ключ, иначе восстановить и прочитать информацию в первоначальном виде будет невозможно.

Актуальность проекта заключается в применении кодирования открытого ключа, передаваемого по тому же каналу связи, и разработке проекта программного средства защиты для достижения поставленной цели. Откуда вытекают следующие задачи:

1) выбор алгоритма шифрования;
2) выбор алгоритма кодирования ключа;
3) выбор языка программирования;
4) выбор среды разработки программного обеспечения.

1.1 Алгоритм шифрования DES

DES (Data Encryption Standard) – один из видов симметричного алгоритма шифрования, который был разработан компанией IBM. В 1977 г. данный алгоритм был утвержден правительством Соединенных Штатов Америки в качестве официального стандарта (FIPS 46-3). В основе структуры DESлежит 16 цикловая структура сети Фейстеля. Длина блоков в DES–64 бита, ключ, используемый для шифрования, имеет длину в 56 бит.В работе алгоритма используются комбинации нелинейных (S-блоки) и линейных (E, IP, IP-1 перестановки) преобразований.

DES является алгоритмом блочного шифрования. Для того, чтобы понять принцип работы DES нужно проанализировать принцип работы сети Фейстеля. В блочном шифровании входными данными служат блок размером n-бит и ключ размером k-бит. После применения шифрующего преобразования получается зашифрованный блок длиной в n-бит. Необходимо учитывать, что незначительный различия входных данных приводят к существенным изменениям в полученных результатах. Применение некоторых основных преобразований над частями исходного кода является принципом блочного шифрования:

1) сложное преобразование на одной локальной части блока;
2) простое преобразование между частями блока.

В связи с тем, что преобразование выполняется в блоках, в качестве отдельного этапа требуется разделение исходных данных на блоки требуемого размера. Однако, независимо от формата исходных данных, таких как текстовые документы, изображения или другие файлы, они должны интерпретироваться в двоичной форме и только после этого шага могут быть разделены на блоки. Это может быть реализовано как программным, так и аппаратным обеспечением.

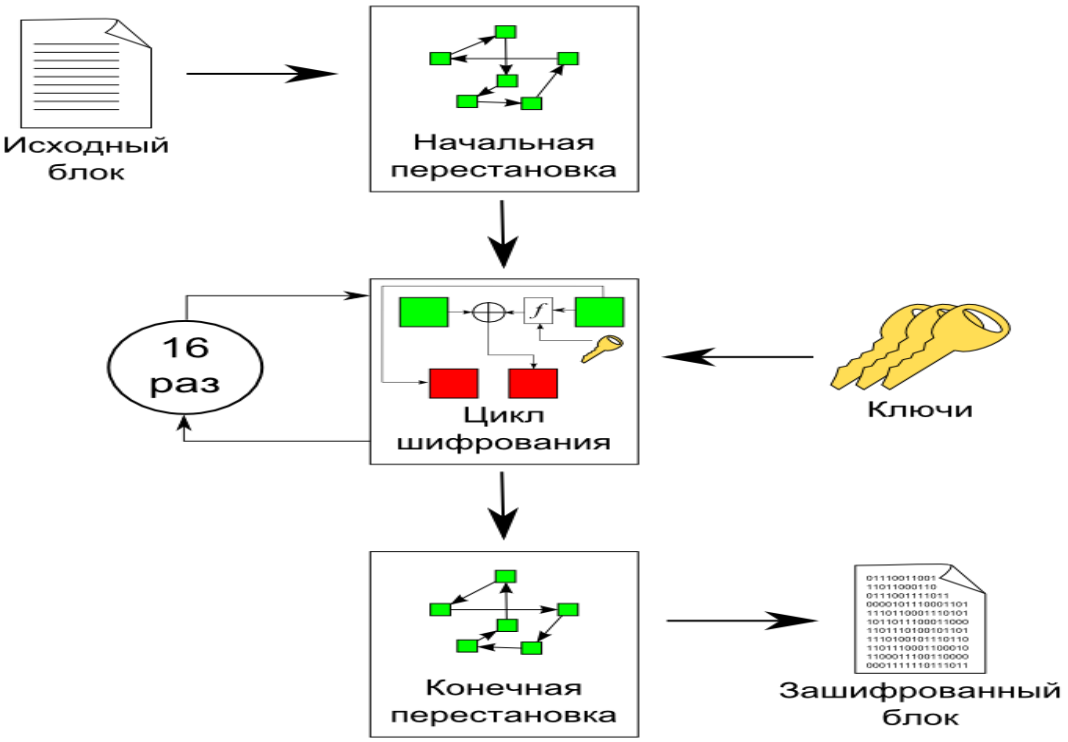

Рисунок 1 - Процесс шифрования в алгоритме DES

Алгоритм DES имеет несколько основных преимуществ:

1) алгоритм использует только один ключ, длиною в 56 бит;

2) при шифровании сообщения используется один пакет, при дешифровании используют любой другой пакет;

3) простота самого алгоритма шифрования дает высокую скорость обработки информации;

4) криптостойкость алгоритма реализована на высшем уровне.

Процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, обратной перестановки битов (рисунок 1).

Рисунок 2 - Обобщенная схема шифрования в алгоритме DES

Рассмотрим функцию шифрования f(R(i-1),K(i)). Схематически она показана на рисунке 3.

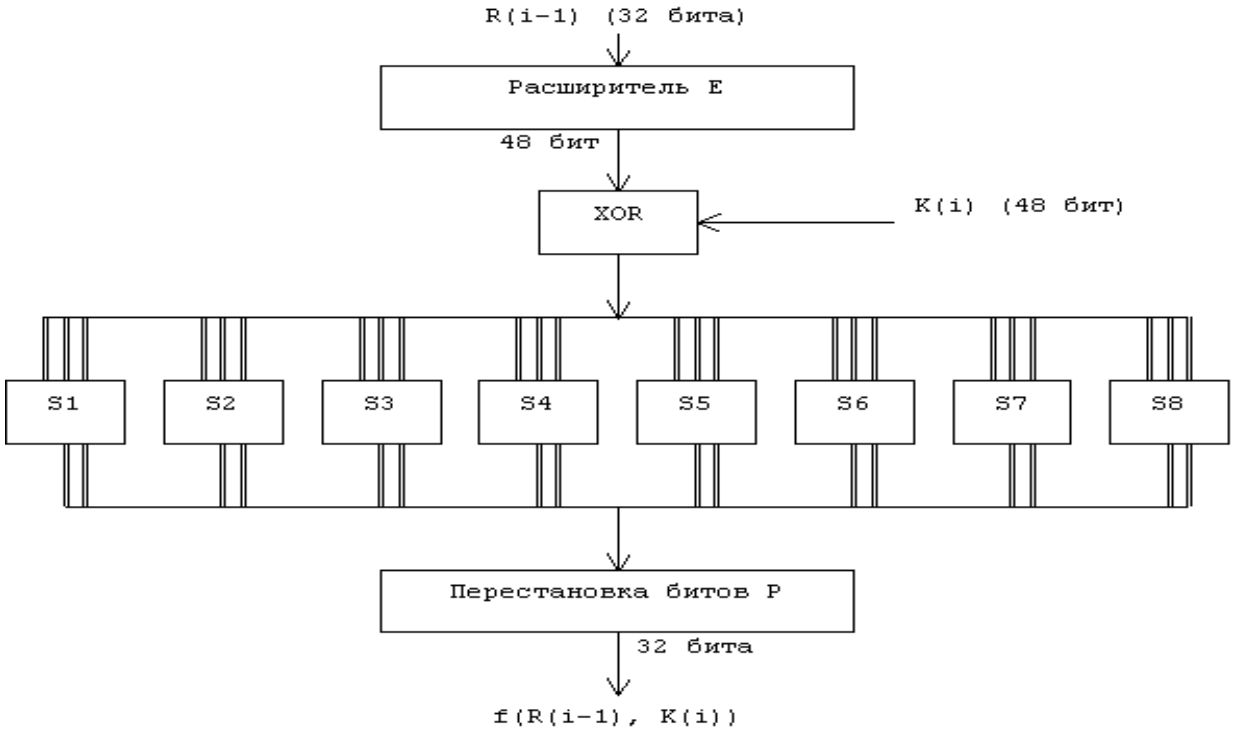

Рисунок 3 - Вычисление функции f(R(i-1), K(i))

Для вычисления значения функции f используются следующие функции-матрицы:

1) E - расширение 32-битовой последовательности до 48-битовой,
   S1, S2, ... , S8 - преобразование 6-битового блока в 4-битовый,
2) P - перестановка бит в 32-битовой последовательности. [1]

1.2 Алгоритм шифрования ГОСТ 28147-89

ГОСТ 28147-89 является российским алгоритмом блочного шифрования. В процессе шифрования данные разбиваются на блоки длиной 64-бит и шифруются 256-битным ключом.

Во время работы алгоритма выполняются 32 раунда криптографических преобразований. В каждом раунде предусмотрены следующие действия:

- Один из 32-битных подблоков данных складывается с 32-битным значением ключа раунда **Ki** по модулю 2.
- Результаты предыдущего действия разбиваются на 8 фрагментов по 4 бита, которые параллельно «прогоняются» через 8 таблиц замен S1…S8.
- После замены 4-битные фрагменты объединяются обратно в 32-битный подблок, значение которого циклически сдвигается влево на 11 бит.
- Обработанный предыдущими операциями подблок накладывается на необработанный с помощью побитовой логической операции XOR.
- Подблоки меняются местами.

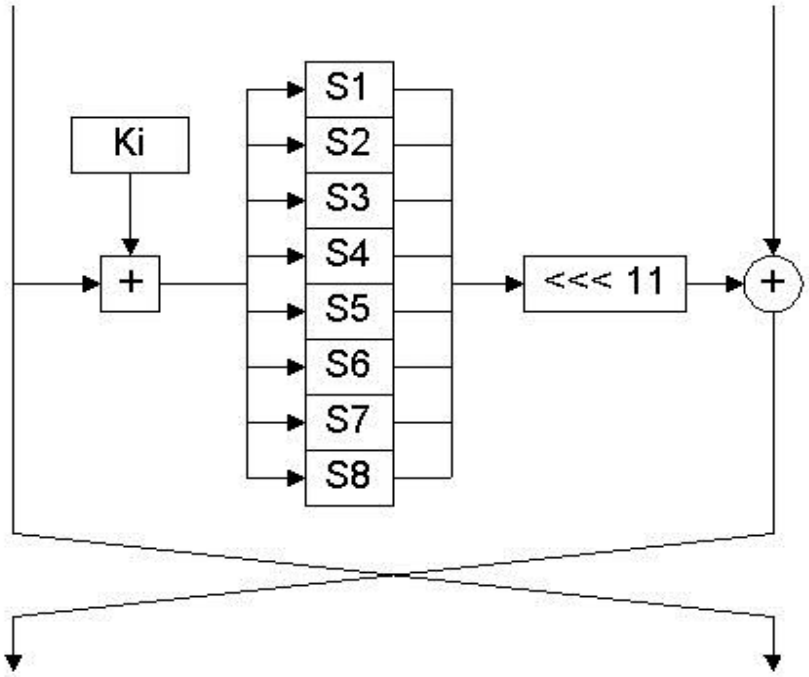

Рисунок 4 - Раунд шифрования в алгоритме ГОСТ 28147-89

В алгоритме фактически нет процедуры расширения ключа: в раундах шифрования 32-битные фрагменты K1 ... K8 исходного 256-битного ключа шифрования последовательно используются в следующем порядке: K1, K2,

K3, K4, K5, K6, K7, K8, за исключением последних 8 раундов - в раундах с 25 по 31 фрагменты используются в обратном порядке.

Процесс расшифровки схож с процессом зашифровки, за исключением использования другого порядка использования фрагментов ключа: прямой порядок – первые 8-раундов, обратный порядок – остальные раунды.

Режимы использования алгоритма:

Режимы гаммирования и гаммирования с обратной связью предусматривают в процессе работы алгоритма вычисление, которое использует описанные выше преобразования псевдослучайной последовательности – гаммы шифра и ее наложение на шифруемый текст.

Режим вычисления имитовставки. Имитовставка – это криптографическая контрольная сумма, используемая для подтверждения целостности данных. В данном режиме работы алгоритма выполняются 16 раундов преобразований, вместо 32-х.

Ряд основных различий между двумя блочными системами шифрования DES и ГОСТ представлены ниже:

- В DES по сравнению с ГОСТ используется более сложная процедура генерации подключей из ключей.

- Длина ключа в DES – 56 бит, в ГОСТ – 256 бит. Также необходимо не забывать о секретных перестановках S-блоков, что делает длину полного объема секретной информации в ГОСТ равной 610 бит.

- S-блоки в DESимеют входы и выходы длиной 6-бит и 4-бит соответственно. - S-блоки в ГОСТ и на входе, и на выходе имеют длину 4-бит. В работе обоих алгоритмов используются по 8 S-блоков, однако размер S-блока ГОСТ меньше размера S-блока DES в 4 раза.

- В DES используются P-блоки – нерегулярные перестановки, а в ГОСТе используется 11-битный циклический сдвиг влево.

- В DES 16 циклов, а в ГОСТе — 32.

Нет смысла использовать силовую атаку на ГОСТ. При шифровании ГОСТ использует 256-битовый ключ, а если учитывать секретные S-блоки, то длина ключа будет еще больше. По сравнению с DES, ГОСТ более устойчив к дифференциальному и линейному криптоанализу. Хотя случайные S-блоки ГОСТа при некотором выборе не гарантируют высокой криптостойкости по сравнению с фиксированными S-блоками DES, их секретность увеличивает устойчивость ГОСТа к дифференциальному и линейному криптоанализу. К тому же эффективность этих криптоаналитических методов зависит от количества циклов преобразования - чем больше циклов, тем труднее криптоанализ. ГОСТ использует в два раза больше циклов, чем DES, что, возможно, приводит к несостоятельности дифференциального и линейного криптоанализа. [1]

В ГОСТ не используется перестановка с расширением, которая существует в DES. Можно предположить, что отсутствие вышеуказанной операции имеет негативное влияние на криптографическую стойкость работы алгоритма ГОСТ. Однако с точки зрения криптоустойчивости операция арифметического сложения в ГОСТ ничем не уступает операции «исключающее или» в DES.

Основным различием представляется использование в ГОСТе циклического сдвига вместо перестановки. Перестановка DES увеличивает лавинный эффект. В ГОСТе изменение одного входного бита влияет на один S-блок одного цикла преобразования, который затем влияет на два S-блока следующего цикла, затем на три блока следующего цикла и т.д. Потребуется восемь циклов, прежде чем изменение одного входного бита повлияет на каждый бит результата; в DES для этого нужно только пять циклов. Однако ГОСТ состоит из 32 циклов, а DES только из 16. [1]

Команда разработчиков алгоритма ГОСТ предпринимала все попытки для достижения равновесия между криптографической стойкостью и эффективностью. Используя в качестве фундамента (основы) структуру Фейстеля, разработчикам удалось создать криптографический алгоритм, который больше подходит к программной реализации, чем DES. С целью повысить криптостойкость был введен ключ большего размера и удвоено количество циклов преобразований. Тем не менее, вопрос о криптоустойчивости ГОСТ перед DES все еще открыт.

1.3 Алгоритм шифрования Base64

При кодировке очередность знаков делится на группы по 4 знака. Дальше в согласовании с таблицей кодировки Base64, любому знаку сопоставляется десятичное значение порядкового номера. Десятичное значение отображается в 6-битовом виде, и вслед за тем в итоге выходит 24-битовая очередность. Приобретенная на предыдущем шаге очередность разбивается на 3 группы длиною 8 бит. Эти группы дальше приводятся к символьному виду, и в итоге чего мы получаем 3 ASCII символа.

Шесть бит первого символа Base64 представляют собой шесть бит первого байта данных. Они должны быть перемещены с позиций 1-6 на позиции 3-8. Для этого происходит сдвиг влево на две позиции. Затем выбирают биты из позиций 5-6 из второго символа, и выбранные 6 старших и 2 младших бита объединяются. Результатом является 1-й байт данных.

Чтобы получить второй байт данных, выполняются одни и те же операции, но вместо того, чтобы сдвигать влево на два и вправо на четыре бита, идет сдвиг влево на четыре и вправо на два бита.

Чтобы получить 3-й байт данных, мы берем 5-6 бит третьего символа Base64, соответствующего первым двум верхним битам данных и 4-му символу. В результате операции сложения мы получаем третий байт данных.

Base64 буквально означает систему позиционирования с базой 64. Здесь 64 - количество символов в алфавите кодирования, из которых последний буквенно-цифровой текст основан на латинском алфавите. Число соответствует наивысшей мощности двух (26), которые могут быть представлены с использованием печатных символов ASCII. Эта система широко используется в электронной почте для представления двоичных файлов в тексте сообщения (транспортное кодирование). Все известные варианты, известные как Base64, используют символы A-Z, a-z и 0-9, что составляет 62 символа; для отсутствующих двух символов в разных системах используются разные символы.

Процедура кодирования: процедура предназначена для преобразования двоичных файлов перед их передачей через службы, которые поддерживают только 7-битное кодирование ASCII (SMTP, NNTP и т. Д.). Сущность преобразования сводится к замене двоичных цифр в серии символов ASCII. Каждая последовательность из трех байтов (24 бит) сообщения преобразуется в четыре шестибитовых значения (рисунок 5). Затем каждому шестибитовому значению присваивается символ ASCII в соответствии с числом, представленным шестью битами. Количество символов ASCII ограничено 64. Ниже приведен пример соответствия кодов Base64 и ASCII. Если количество символов (байтов) не кратно трем, то используется дополнительный символ «=». [2]

Слово:		C	O	P	E	
Шестнадцатеричное:		0x43	0x4F	0x50	0x45	
Двоичное:		01000011	01001111	01010000	01000101	
6-битовое:	010000	110100	111101	010000	010001	010000
Десятичное:	16	52	61	16	17	16
Base64:	Q	0	9	Q	R	Q

Рисунок 5 – Преобразование сообщения Base64

Процедура декодирования: последовательность символов делится на группы по 4 символа. Далее, согласно таблице кодирования Base64, каждому символу присваивается десятичное значение порядкового номера. Десятичное значение представлено в 6-битной форме, а затем результат - 24-битная

последовательность. Последовательность, полученная на предыдущем этапе, делится на 3 группы по 8 бит. Эти группы далее сводятся к символической форме, и в результате мы получаем 3 символа ASCII.

Шесть бит первого символа Base64 представляют шесть бит первого байта данных. Они должны быть перемещены с позиций 1-6 на позиции 3-8. Для этого сдвиньте символы влево на две позиции. Затем выбирают биты из положений 5-6 второго символа, и выбранные 6 старших и 2 младших бита объединяются. Результатом является 1-й байт данных.

Чтобы получить второй байт данных, выполняются те же операции что и выше, но вместо сдвига влево и вправо на четыре бита происходит сдвиг влево на четыре и вправо на два бита.

Чтобы получить 3-й байт данных, мы берем 5 и 6 биты третьего символа Base64, соответствующего первым двум верхним битам данных и 4-му символу. В результате операции сложения мы получаем третий байт данных.

1.4 Алгоритм шифрования LEA-128-SEA

Принцип работы симметричного алгоритма шифрования (SEA) основан на применении информационной свертки – специальным образом организованной процедуры поглощения битов. Для ее демонстрации рассмотрим процедуру информационного свертывания на примере произвольной битовой строки. Свертка данных происходит следующим образом: берутся два смежных входных бита и на основе их значений устанавливается значение выходного бита. Далее процесс повторяется, т.е. рассматриваются следующие два смежных бита. Важной особенностью информационной свертки является возможность обратного преобразования при условии применения симметричного правила преобразования. [2]

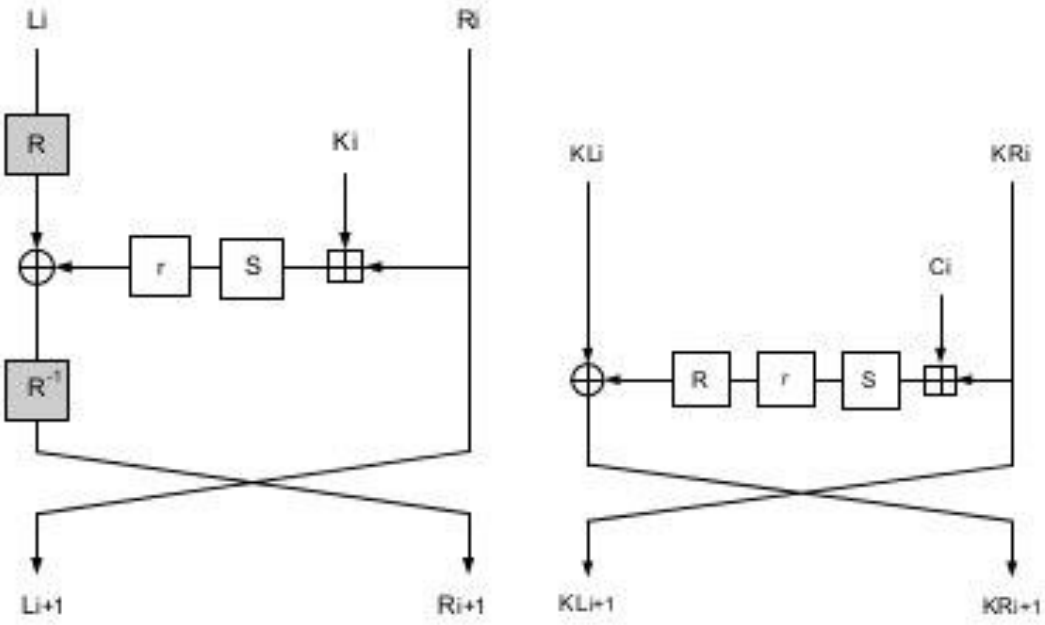

Рисунок 6 – 1 раунд алгоритма LEA-128 SEA

При обратном процессе получения информационной развертки размер данных увеличивается на 1 бит. Однако в результате развертки образуются две битовых строки, свертка которых даст исходную строку. Данное свойство информационной свертки можно использовать для шифрования данных.

Способ шифрования SEA (Symmetric Encryption Algorithm) является наиболее быстрым среди уже известных симметричных методов. При аппаратной реализации шифрование может происходить со скоростью N тактов на блок, где N длина ключа. Блок же может достигать больших размеров (мегабайт и более). Самым важным преимуществом SEA перед другими симметричными способами является возможность изменять длину ключа без изменения алгоритма шифрования. Она может быть любой. Естественно, что скорость шифрования зависит от длины ключа. Поскольку длина ключей не велика, то для их распространения можно использовать известные методы. [3]

Проведя сравнительный анализ рассмотренных выше криптосистем мной был сделан выбор в пользу алгоритмов LEA-128 SEA и Base64, что и будет рассмотрено в моем дипломном проекте.

2 Программная реализация работы алгоритмов шифрования

2.1 Язык программирования Delphi

Для программной реализации работы выбранных мною ранее алгоритмов шифрования был выбран язык программирования Delphi 7.

Остановимся на истории создания данного языка программирования. Когда упоминают Delphi возникает ассоциация со средой разработки приложений фирмы Borland, базированных на языке Object Pascal. Object Pascal – это наследник Turbo Pascal имеющий объектно-ориентированные расширения. Этот язык берет свое начало от «чистого» Pascal, который был создан Никлаусом Виртом в 1970 году. Далее, в 2002 году разработчики компании Borland официально приравняли языки Delphi и Object Pascal.

Delphi императивный, структурированный, объектно-ориентированный язык программирования со строгой статической типизацией переменных. В основном используется для написание прикладного программного обеспечения. [4]

2.1.1 Структура программы на Delphi

Написанная на языке Delphi программа имеет в своей структуре заголовок программы, Uses – поля используемых модулей, которые могут не входить в саму структуру, как Uses Windows, Messages, SysUtils и др., блоки описания и исполнения, начинающиеся составным оператором begin и заканчивающиеся оператором end. Ниже представлен пример написанного кода программы:

```
program Project1;          // Заголовок программы, с её именем «Project1»
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};       // модули, которые подключены к проекту и используются программой
{$R *.res}
begin
  Application.Initialize;             // Инициализация приложения
  Application.CreateForm(TForm1, Form1); // Создание формы/окна
  Application.Run;                    // Запуск и исполнение
end. [4]
```

2.1.2 Инструменты разработки

В качестве инструмента разработки программного обеспечения будет использоваться Delphi 7.

Delphi 7 является дальнейшим развитием Delphi 6, имеет улучшенную библиотеку CLX (Component Library for Cross Platform – кроссплатформенная библиотека для разработки программ), новую корреспондирующую версию среды разработки под операционную систему Linux – Kylix 3. Кроме этого, начиная с этой версии, компания Borland обеспечила совместимость языка с платформой Microsoft .NET.

2.1.3 Особенности языка

В начале, так как язык был предназначен для обучения программированию и его дисциплине, он ставил на первое место стройность и высокую читаемость кода. Изначально заложенная стройность значительно упростила расширение языка новыми конструкциями в процессе роста аппаратных мощностей и в результате появления новых парадигм. В итоге, сложность объектного C++, по сравнению с C, выросла весьма существенно и затруднила его изучение в качестве первого языка программирования, чего нельзя сказать об Object Pascal относительно того же самого Pascal. [4]

В Delphi формальное начало любой программы отличается от остальной части кода и должно быть расположено в одном исходном файле с расширением .dpr, в то время как другие файлы исходного кода имеют расширение .pas. В C-подобных языках программирования вход обычно представляет собой глобальную функцию или статический метод с основным именем и определенным списком параметров, и эта функция может быть расположена в любом из файлов исходного кода проекта. [4]

В Delphi чтение идентификаторов типов, переменных, ключевых слов не зависит от регистра, т.е. идентификатор SomeVar полностью эквивалентен somevar.

В исходных файлах pas (которые, как правило, содержат основную часть программы), на уровне языковых инструментов было введено строгое разделение на раздел интерфейса и раздел реализации. Часть интерфейса содержит только объявления типов и методов, тогда как код реализации в интерфейсной части не разрешен на уровне компиляции. Это разделение также характерно для языков C / C ++, где вводятся заголовки и фактические файлы реализации, но это разделение не предоставляется на уровне языка или компилятора. [4]

В Delphi метод или функция четко определяется процедурой (prodedure) или функцией (function), зарезервированной для этого ключевого слова, тогда как в языках C-типа различие связано с ключевым словом, которое определяет тип возврата.

Начало и конец программного блока определяются ключевыми словами начала и конца, тогда как на языках программирования C для этих целей используются фигурные скобки: {}. Таким образом, Delphi обеспечивает лучшую читаемость кода.

2.2 Среда разработки RAD Studio

В качестве среды разработки была выбрана RAD Studio 10.1 Berlin.

RAD Studio™ является самым быстрым вариантом при разработке кросс-платформенных приложений, используя облачные сервисы и широкое подключение Internet of things. Данная среда разработки предоставляет мощные компоненты Visual Component Library - библиотека визуальных компонентов для операционной системы Windows 10, также дает возможность разработки на FireMonkey для Windows, Mac и мобильных устройств. В сегменте корпоративно-ориентированного развития RAD Studio осуществляет поддержку языков программирования Delphi и C++. Среда разработки дает возможность обеспечения 5-кратно увеличенной скорости разработки и развертывания на нескольких настольных, мобильных, облачных платформах и системах, включая 32 и 64-разрядные операционные системы Windows 10.

RAD Studio™ 10.1 Berlin Architect - это самый быстрый способ создания и обновления приложений с интенсивным использованием данных, распределенных и высокоинтерактивных приложений с расширенным и визуально привлекательным интерфейсом для Windows 10, Mac, мобильных устройств, Internet of things и других платформ с использованием Object Pascal и C ++. Есть возможность легкого обновления Visual Component Library и FireMonkey-приложения для Windows 10, используя новые элементы управления и Visual Component Library-стили для Windows, а также компоненты служб универсальной платформы Windows. RAD Studio Architect поддерживает все функции Enterprise Edition, а также содержит мощную среду для моделирования и разработки баз данных ER / Studio Developer.

RAD Studio™ 10.1 Berlin Enterprise - это комплексное решение для разработки программного обеспечения для независимых и корпоративных разработчиков. Есть возможность создания клиент-серверных и многоуровневых совместимых приложений, которые подключаются к широкому спектру корпоративных баз данных и облачных платформ, включая Microsoft SQL Server, DB2, Oracle, Sybase, InterBase, Amazon и Microsoft Azure. RAD Studio Enterprise поддерживает все функции Professional edition, а также обеспечивает подключение к корпоративным данным с помощью DataSnap Software Development Kit и содержит лицензию разработчика для Enterprise Mobility Services.

RAD Studio™ 10.1 Berlin Professional - это самый быстрый способ создания и обновления приложений с интенсивным использованием данных,

распределенных и высокоинтерактивных приложений с расширенным и визуально привлекательным интерфейсом для Windows 10, Mac, мобильных устройств, Internet of things и других платформ с использованием Object Pascal и C ++ . Вы можете легко обновить Visual Component Library и FireMonkey-приложения для Windows 10, используя новые элементы управления и Visual Component Library-стили для Windows, а также компоненты служб универсальной платформы Windows. [5]

Рисунок 7 – Графическая оболочка среды разработки

2.3 Построение модели локальной сети предприятия на NetCracker 4.1 Professional.

NetCracker Professional 4.1–это программное обеспечение, используемое для проектирования, моделирования и анализа компьютерных сетей.

Программное обеспечение NetCracker позволяет собирать релевантные данные о существующей сети, не останавливая ее работу, создавать проект этой сети и проводить необходимые эксперименты для определения предельных характеристик, расширяемости, топологии изменения и модификации сетевого оборудования для его дальнейшего совершенствования и развития.[6]

С помощью NetCracker Professional 4.1 мною была спроектирована модель локальной сети «Центра анализа и расследования кибер атак», состоящая из 21 ноутбуков и персональных компьютеров.

2.3.1 Элементы сети:

- Рабочая станция: 21

- Сервер: 1

- Коммутатор: 1

- Маршрутизатор: 1

- Серверное программное обеспечение: 6

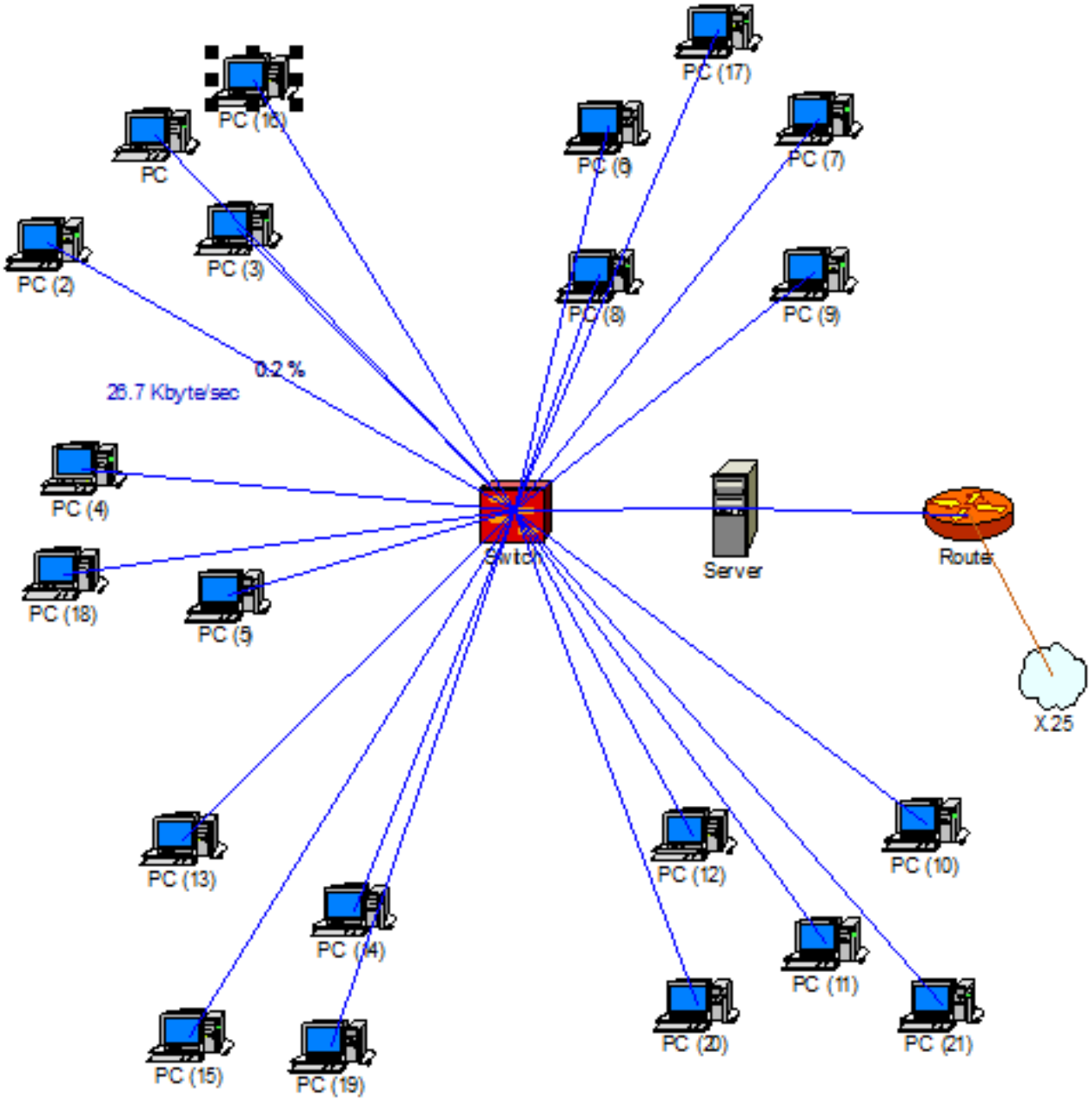

Рисунок 8 – Модель локальной сети «Центра анализа и расследования кибер атак»

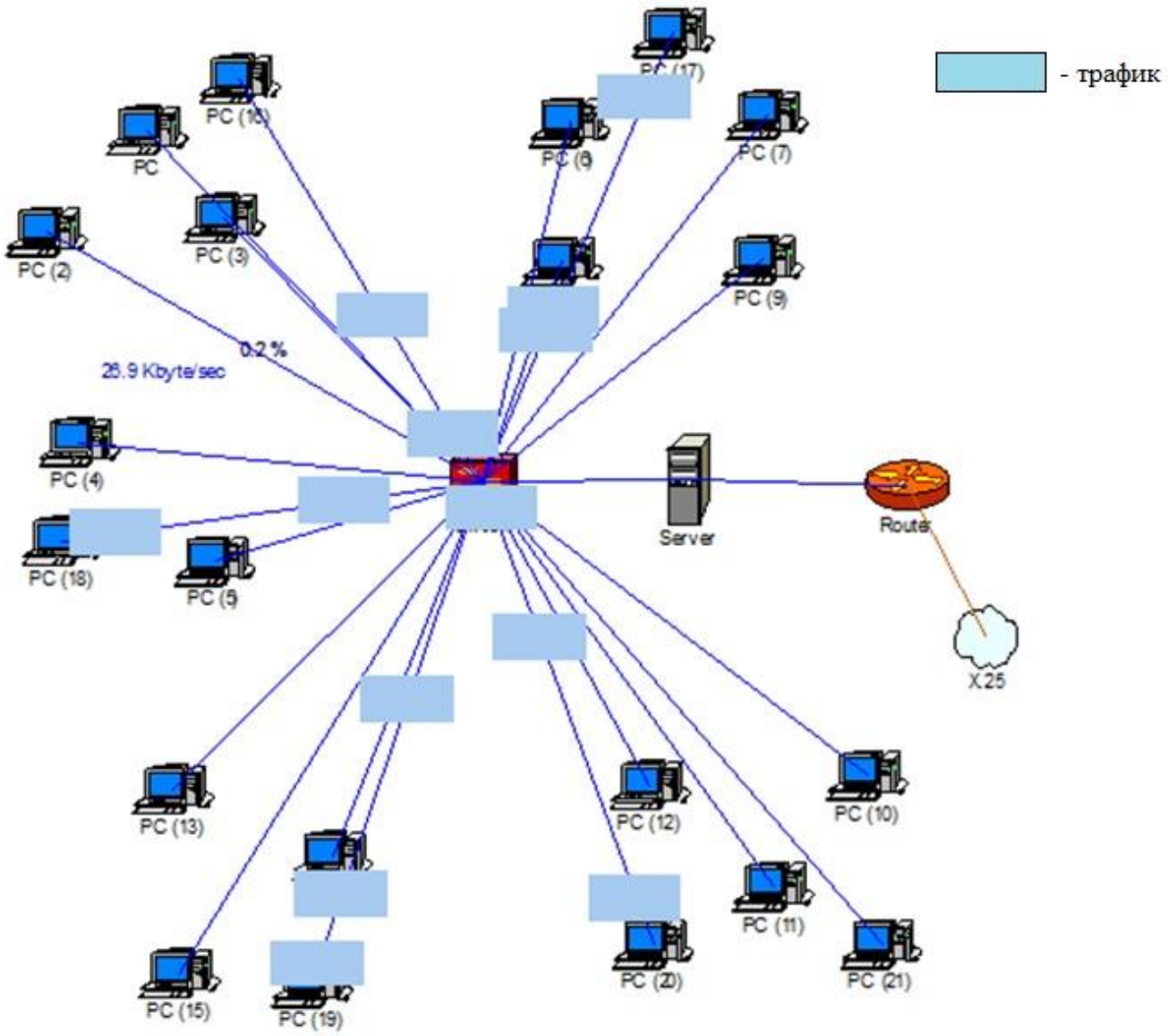

Рисунок 9 – Схема прохождения трафика по смоделированной в эмуляторе локальной сети

Были выполнены следующие действия:

- Switch, Workstations, Server, Server Software были выбраны из базы данных инструментов и перенесены в рабочее пространство;

- между коммутатором и хостом (рабочей станцией), а также между коммутатором, сервером и маршрутизатором были созданы соединения. Свойства канала указывали на тип кабеля (оптическое волокно или витую пару) его длину и максимальную скорость передачи (3 м, 10 Мбит / с и 100 Мбит / с);

- были созданы контурные профили рабочих нагрузок, затем добавлены свойства трафика между сервером и клиентом: 4 класса приложения с размером пакета 50 байтов и 2 класса с размером пакета 1500 байт, время подготовки и обработка составляет 2 мс. Показаны признаки применения каналов передачи и скорости обработки на хосте и на сервере.

Как видно из моего примера, построить модель внутренней сети организации не трудно, и также позволяет четко продемонстрировать ситуацию.

Теперь необходимо определить скорость прохождения трафика и общую нагрузку на сеть. Входными данными для имитационного моделирования являются:

а) топология сети;

б) рабочие нагрузки сети;

в) законы распределение трафиков.

Сетевая топология - это физическое расположение сетевых объектов относительно друг друга (имеет некоторую геометрическую форму).

На любом сайте установлено определенное количество портов. Например, компьютер - 1 или 2 порта, коммутатор - более 8 портов. Каждому порту назначается определенное число - номер подключения. Порты разных узлов, имеющих один и тот же номер соединения, объединяются. Порт узла, которому назначен номер подключения 0, является бесплатным.

Чтобы определить нагрузку на сеть, вам понадобятся следующие понятия:

- маршруты прохода;
- размеры пакетов в байтах N3;
- время генерации пакетов запросов в миллисекундах;
- размер ответов в байтах N0;
- время для подготовки ответов в байтах To;
- время транзитных циклов движения;

Запрос в этом случае означает запрос от клиента и необязательный ответ на запрос с сервера.

Основной узел, в котором обрабатываются запросы, является прямым клиентом. При обработке в этом случае он предназначен для отправки запросов абонентскими приложениями в течение времени T3. Один клиент может отправить определенное количество запросов на передачу на один или несколько серверов. Период подготовки запроса в клиенте определяется только T3 и не зависит от N3.

Объем пакета N3 зависит от периода его передачи. С увеличением Nm этот период увеличивается, однако эта зависимость не является линейной и зависит от большого числа условий.

Параметры «To» и «No» важны только в этом случае, если вы хотите отправить ответ сервера на запрос клиента. В этом случае их значения аналогичны значениям T3 и N3.

Обычно клиентское приложение обращается к серверу с запросами на чтение, загрузку WEB-страниц и т. Д. В этом случае клиент, как правило, не отправляет последующий запрос, пока не получит результат по ранее отправленному запросу. Чтобы показать это, необходимо указать период цикла Tc как 0.

Часто достаточно использовать датчики разных характеристик, которые формируют необходимую информацию с помощью некоторых идентичных временных интервалов. Эти интервалы времени должны быть указаны в свойстве времени цикла Tc.

2.3.2 Время подбора моделирования

Временные характеристики работы сетей обычно проявляются в периоде передачи 1-го бита. Аналогичным образом мы продемонстрируем это в нашем случае.

Для моделирования был выбран минимальный промежуток, или квант времени t_{min}, равный 1 нс = 10^{-9}с, умножением которого на целое число n_x рассчитываются все необходимые задержки. Число n_t определяется следующим образом:

$$n_x = T_x / t_{min}, \quad (2.2)$$

где T_x = требуемая задержка.

Например, передача одного байта при скорости передачи 1 Гбит/с займет: $n_x = (1 / 10^9 \text{ с}) * 8 / 10^{-9} \text{ с} = 8$ [квантов].

В моделируемой сети может быть несколько сайтов (домены конфликтов) с различными пропускными способностями. Каждому такому разделу присваивается собственный nx, в зависимости от его пропускной способности.

При уменьшении t_{min} увеличивается точность моделирования, однако значительно возрастает вычислительная нагрузка. Поэтому предусмотрена возможность использования адаптивного минимального промежутка $t_{min\ ад}$. Например, если все участки сети работают на скорости передачи 10 Мбит/с, нет необходимости ждать каждому участку для передачи одного байта $n_x = (1 / 10 * 10^6 \text{ с}) * 8 / 10^{-9}\text{с} = 800$ [квантов]. В этом случае целесообразней взять $t_{min\ ад} = t_{min} * n_x = 10^{-9} \text{ с} * 800 = 8 * 10^{-7}$с и адаптивное $n_{x\ ад} = 1$.

Значения $n_{x\ ад}$ и $t_{min\ ад}$ определяются участком сети с наибольшей пропускной способностью.

Время, необходимое для передачи одного байта информации в данном разделе сети, будет называться тактом.

Описание структуры данных:

Параметры устройств сети и заявок представляются в виде классов объектов на языке ObjectPascal. Основными классами модуля являются:

Параметры сетевых устройств и приложений представлены в виде классов объектов в ObjetCascal. Основными классами модуля являются:

1) Domain, представляющий домен коллизий сети Интернет. Он имеет в своем составе переменные:

- TicsReTact: integer - количество квантов, необходимых для передачи одного байта в этом домене;

- TicsLeft: integer - сколько осталось времени до конца текущей меры;

- TotalsTacts: int64 - общее количество мер, смоделированных в этой области;

- FrameMinLength: byte - время, в течение которого обнаружение столкновения возможно;

- JobWaitTics: array® int64 - общее время ожидания для каждого приложения в данном домене, включая время ожидания в очереди;

- JobServicingTics: array® int64 - общее время ожидания для каждого приложения в данном домене без учета времени ожидания в очереди;

2) NetNode, который представляет такие узлы сети, как клиент, сервер или коммутатор. Он содержит переменные:

- Cnns: array [0..31] ofinteeeee - это массив номеров доменов, к которым подключен этот узел. Как правило, для рабочей станции используется только одно соединение, для коммутатора - несколько;

- MaxTaskNumber: integer - максимальное количество запросов, которые могут обрабатываться одновременно в этом узле;

- TotalsTacts: int64 - общее количество циклов, имитируемых в этом узле;

- JobWaitTics: array® int64 - общее время ожидания для каждого запроса в данном узле, включая время ожидания в очереди;

- JobServicingTics: array® int64 - общее время ожидания для каждого запроса в этом узле, независимо от времени ожидания в очереди;

3) NetJob, в котором хранятся параметры каждого приложения. Большинство параметров уже были описаны выше:

- ReqSize: intger - размер запроса в байтах;

- AnsSize: intger - размер ответа в байтах;

- AnssNumbehr: integer - сколько ответов на запрос клиента необходимо. Может быть 0 или 1;

- CycleTyme: int64 - время цикла приложения;

Остальные параметры приложения:

- Horts: arrayofinteger - маршрут приложения, состоящий из номеров узлов. Первым номером в этом списке является номер клиента, последний - номер сервера. Между ними номера обозначаются переключателем;

- HorDelays: arrayif int64 - массив задержек приложений в каждом узле. Время подготовки к клиенту и обработка на сервере находятся в этом массиве в первом и последнем местах, соответственно.

4) NetWork, который объединяет три класса, перечисленные выше. Он содержит массивы объектов Domain, NetNode и Netgob.

Настройка и сохранение топологии сети, а также поиск пути между клиентом и сервером назначаются другим модулям NetCracker.

Средняя длина очереди

Средняя длина очереди L для каждого узла узла или домена коллизий Dom и Order Z рассчитывается следующим образом:

L: = Node.JobWaitTacts [Z] / Node.JobWaitTacts,

L: = Dom.JobWaitTacts [Z] / Dom.JobWaitTacts,

Значение JowWaitTacts для этого домена рассчитывается следующим образом:

Dom.JobWaitTacts [i]: = Dom.JobWaitTics [i] * Dom.TicsRerTacts,

Средняя общая длина очереди для каждого узла или домена вычисляется как сумма L для всех запросов из массива Job.

Средняя латентность

Среднее время ожидания W в миллисекундах для каждого узла узла или домена Dom и приложения Z рассчитывается следующим образом:

W: = Node.JobWaitTics [Z] / Jobs [Z] .StatSuccCount / nX,

W: = Dom.JobWaitTixs [Z] / Jobs [Z] .StatSuccConunt / nX,

Переменная StatSuccCount сохраняет количество успешных передач приложения типа Z. Значение nX равно 106 / nxmin.

Среднее общее время ожидания рассчитывается как сумма W для всех запросов из массива Job.

Средняя загрузка

Средняя загрузка U в процентах для каждого узла Node или домена Dom из приложения Z вычисляется следующим образом:

U: = Node.JobServicingTics [Z] * 100 / TatalTics,

U: = Dom.JobServationTics [Z] * 100 / TothTics,

где TotalTics - это число квантов моделирования сети.

Общая нагрузка узла или домена вычисляется как сумма U для всех запросов из массива Job.

Время отклика сети

Очевидно, что время, когда пользователь получит ответ от сервера, равно времени обработки приложения W [Z]. Он рассчитывается как сумма средних времен запросов, ожидающих в каждом узле сети.

Также имеет смысл, что среднее время отклика сети Wzs, которое рассчитывается следующим образом:

Wzs: = □Jobs [Z] .StatSuccCount / Lan.StatSuccCount * W [Z],

где Lan.StatSuccConunt является суммой Job [Z] .StatSuccCont для всех запросов из массива Job. [6]

В соответствии с текущей перегрузкой были показаны маршруты и характеристики заказов.

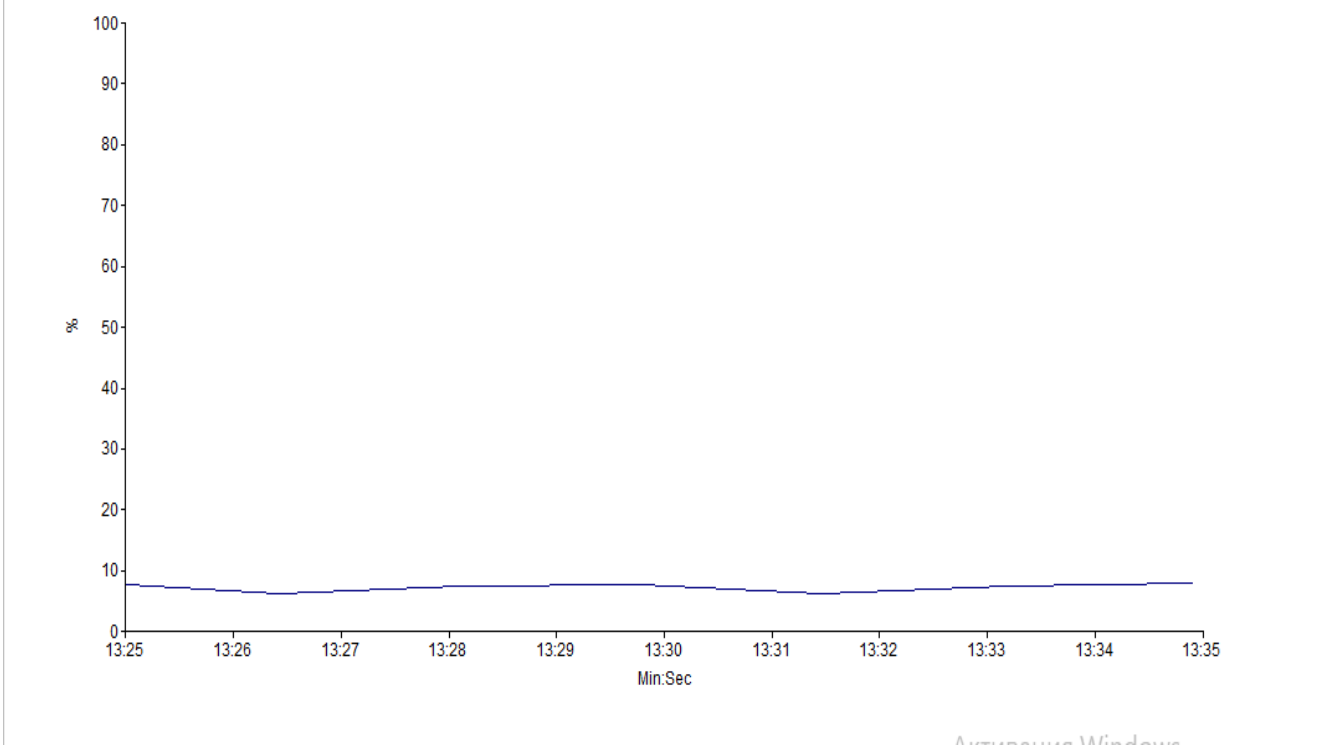

Рисунок 10 – Скорость прохождения трафика по сети

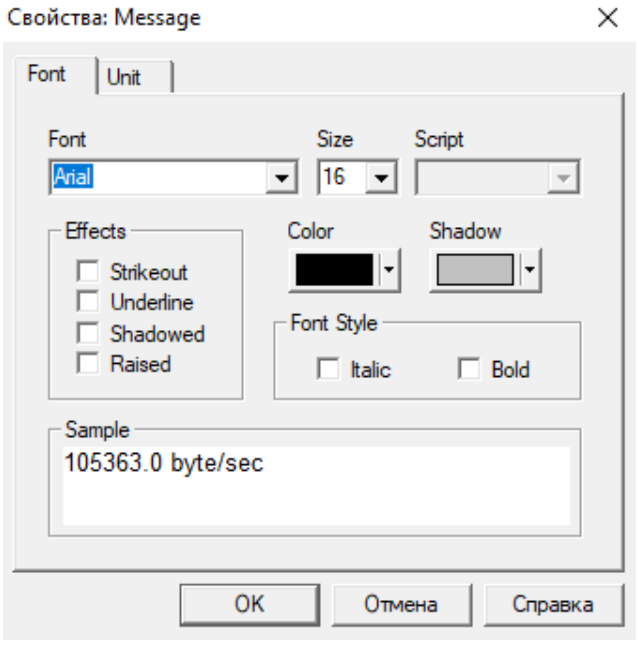

Рисунок 11 – Средняя нагрузка внутри сети

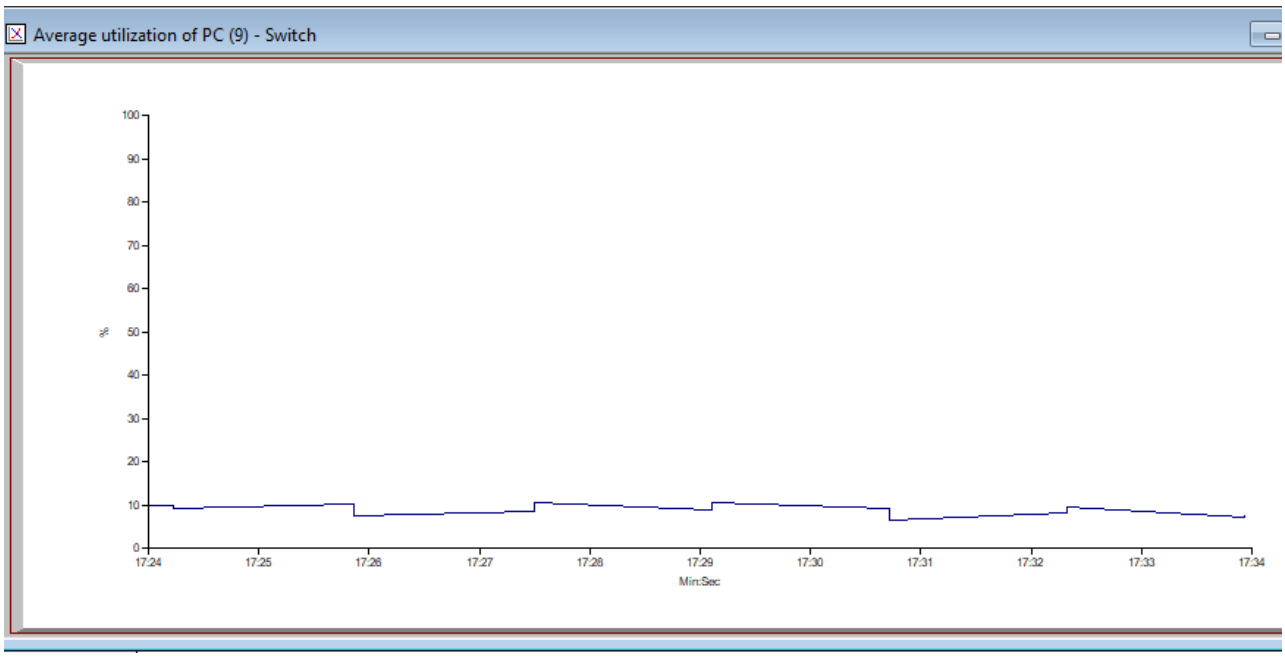

Рисунок 12 – Средняя нагрузка на коммутатор

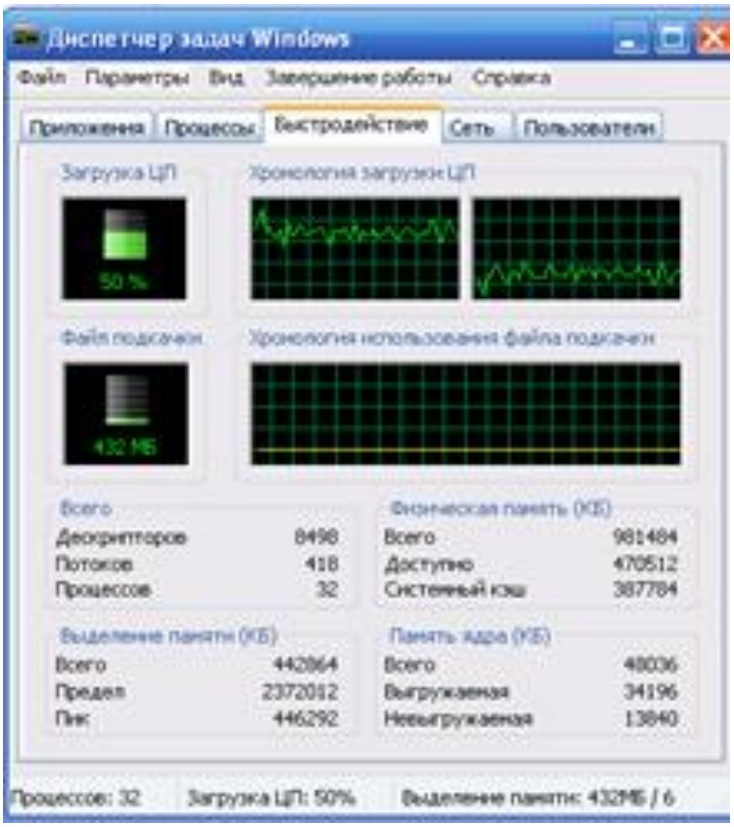

Рисунок 13 – Нагрузка на центральный процессор 50% при полной работоспособности ПК

3 Разработка программного обеспечения AIPET Code в среде разработки Embarcadero® RAD Studio Berlin 10

3.1 Начало работы с Embarcadero® RAD Studio Berlin 10

После успешной установки среды разработки на рабочий компьютер, можно начинать работу над созданием и компиляцией нашего программного обеспечения.

Для начала работы с RAD Studio, необходимо запустить саму среду разработки. Для этого в меню Пуск, в поисковой строке вводим запрос «RAD Studio», запускаем двойным кликом мышки RAD Studio Berlin 10.1 (рисунок 14). Если среда разработки была установлена корректно, то должно появится окно запуска (рисунок 15).

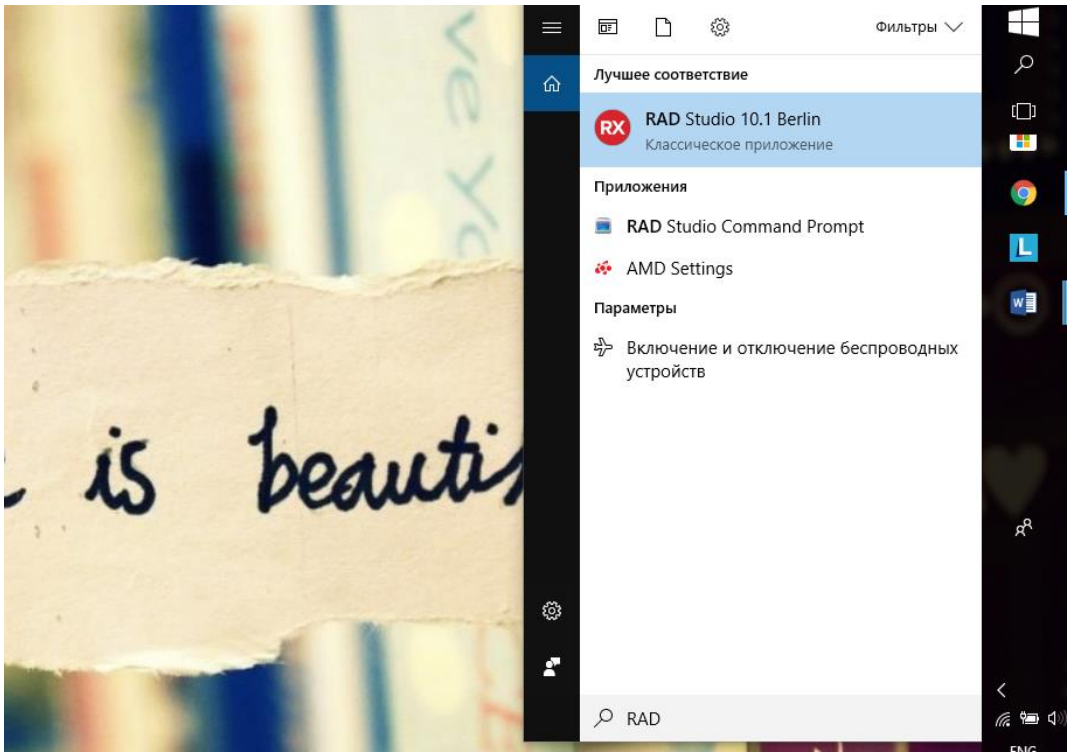

Рисунок 14 – Запуск среды разработки RAD Studio

Рисунок 15 – Успешный запуск среды разработки

Программное обеспечение AIPET Code я буду программировать на языке Delphi, в используемой мной среде разработки рабочая область, главное окно программы будет выглядеть как на рисунке 16.

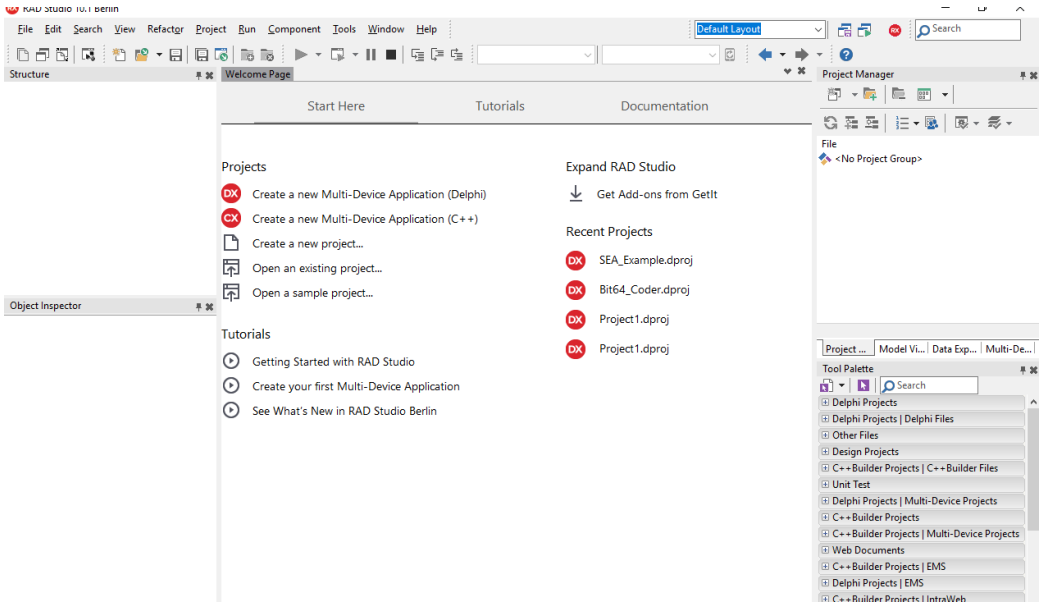

Рисунок 16 – Главное окно среды разработки RAD Studio

Для создания нового приложения (программы) необходимо в главном окне, в вкладке File выбрать New, затем VCL Forms Application – Delphi. По выполнению предыдущих действий создается новая, пустая форма, где размещая компоненты и программируя их, я создам свое программное обеспечение. Далее на рисунке 17 представлена пустая форма моего будущего программного обеспечения.

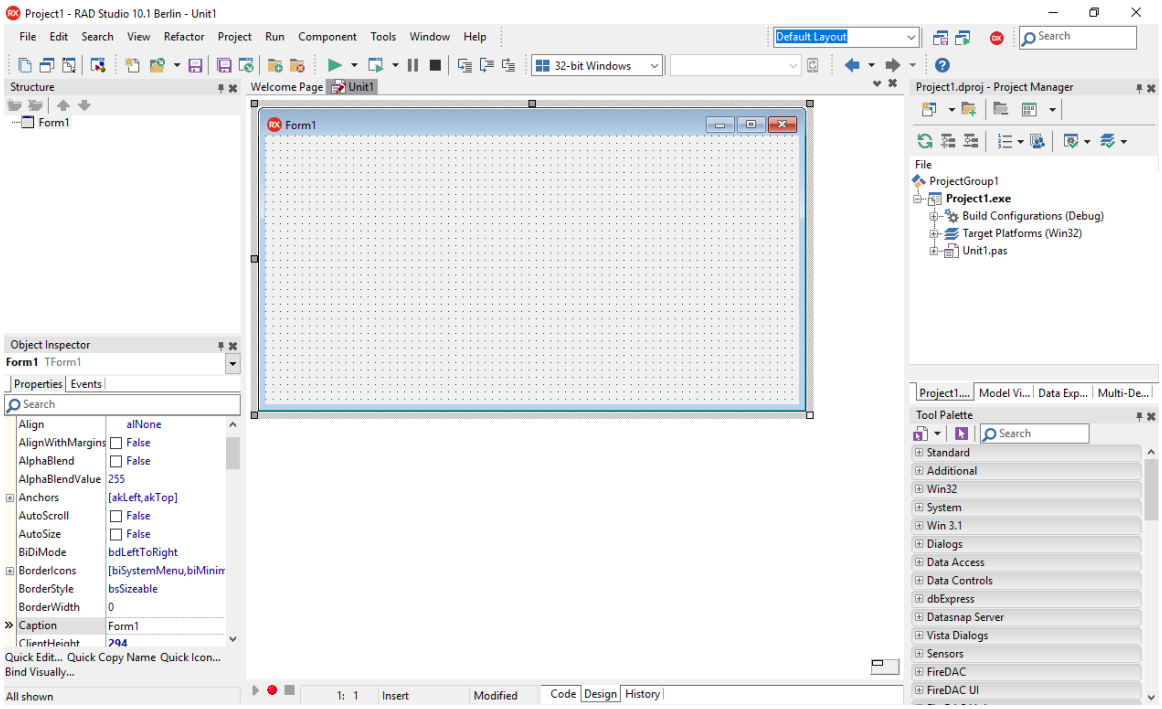

Рисунок 17 – Внешний вид рабочей области в среде разработки RAD Studio

Form 1 – это форма в которой будут размещены программные компоненты и дизайн программного обеспечения. Название можно поменять так как хочется. В моем случае это будет AIPET Code (рисунок 18).

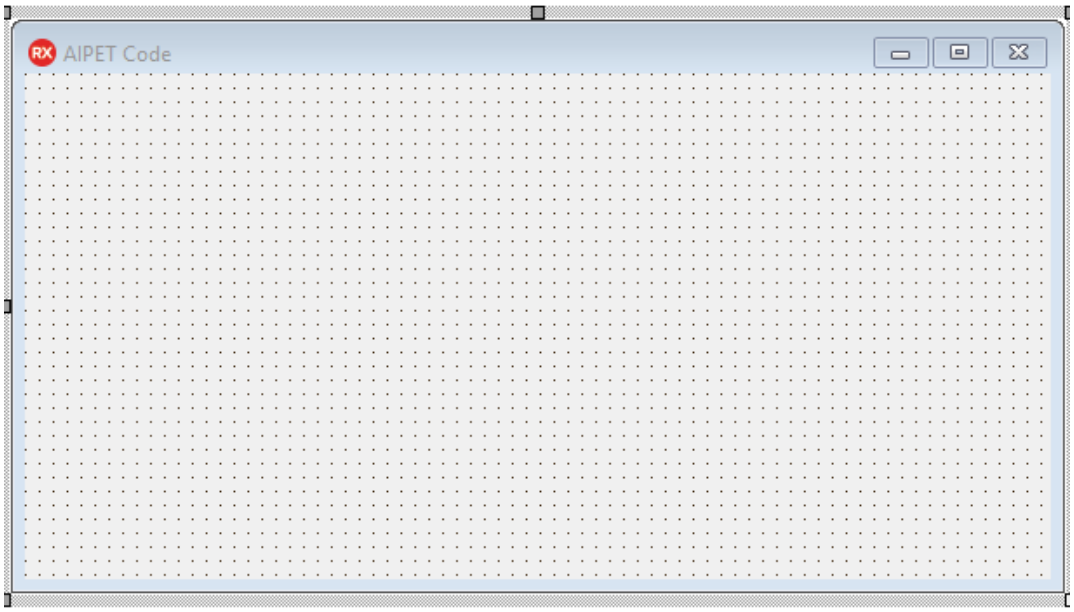

Рисунок 18 – Окно программы

В правом и левом нижнем углах рабочей области располагаются Панель инструментов (Tool Palette) (рисунок 19) и Инспектор объектов (Object Inspector) (рисунок 20) соответственно. Они отвечают за компоненты программы и их свойства.

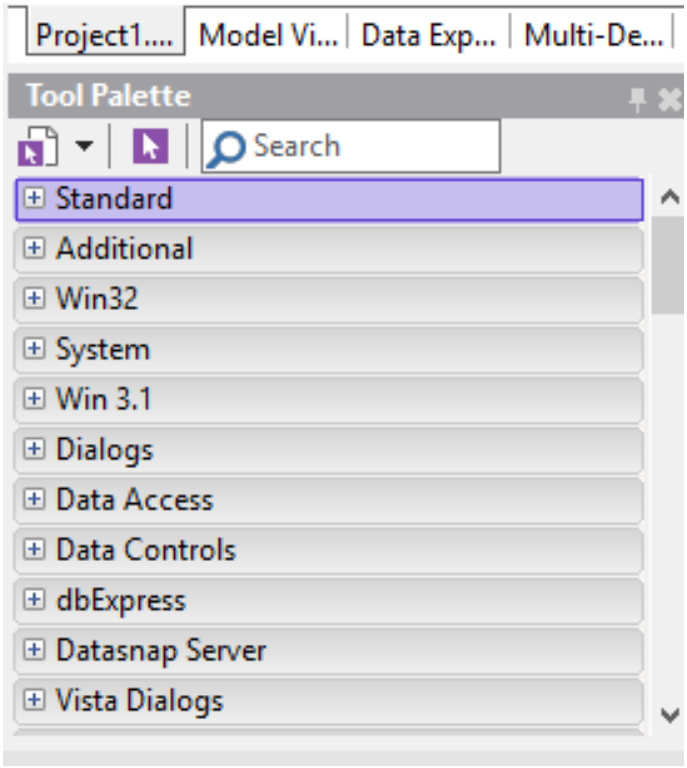

Рисунок 19 – Панель инструментов

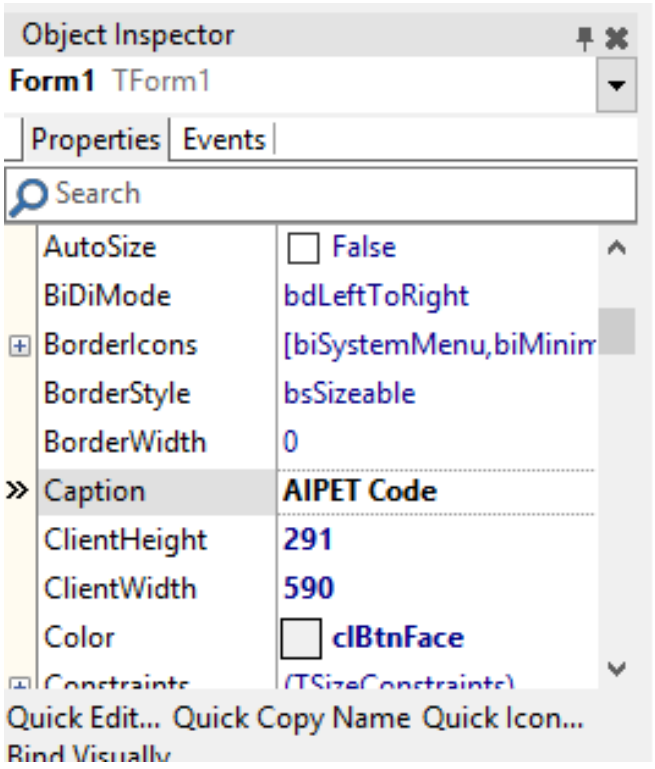

Рисунок 20 – Инспектор объектов

Панель инструментов содержит большое количество визуальных и не только компонентов, с помощью которых можно создать удобный и читаемый пользовательский интерфейс программного обеспечения. Большая часть

необходимых базовых компонентов находится в вкладках Standard, Additional и Win32 (рисунок 21).

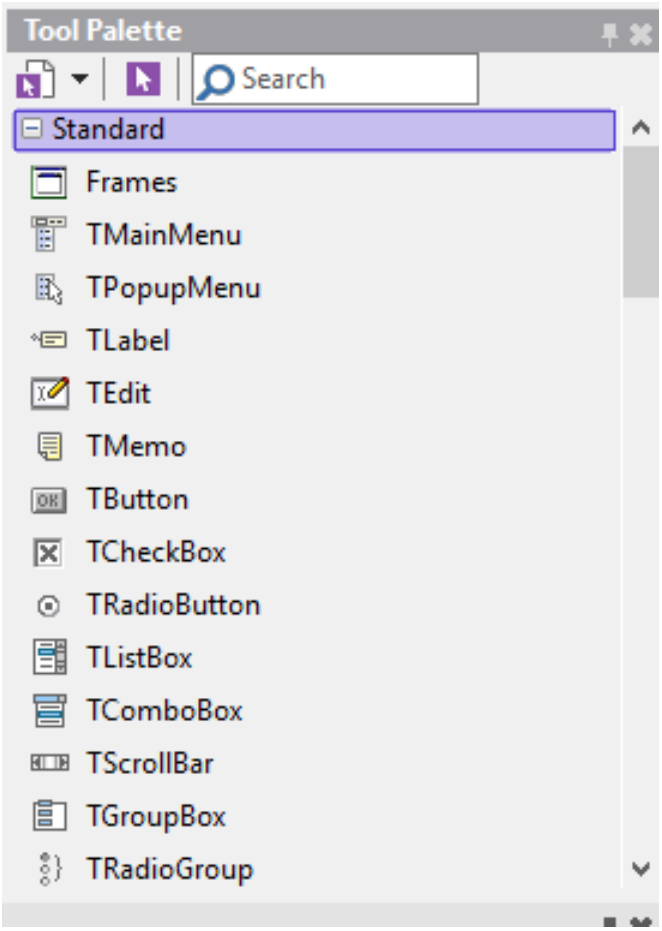

Рисунок 21 – Компоненты вкладки Standard

Из вкладки Standard в создании программного обеспечения используются компоненты: TLabel, TButton, TRadioButton, TEdit. Для создания данного программного обеспечения достаточно компонентов из вкладки Standard. Однако, если разработчик желает добавить дополнительные компоненты, необходимые для пользовательского интерфейса, среда разработки и панель инструментов предоставляют такую возможность.

Инспектор объектов, упомянутый выше содержит две вкладки: Свойства (Properties) и События (Events). Вкладка Properties помогает настроить свойства формы или какого-либо компонента формы. К примеру, название окна программы, шрифты, цвет фона и текста, ширину, высоту и т.д (рисунок 22). Events позволяет выбрать и настроить действия или события, допустимые для формы или ее компонентов (рисунок 23).

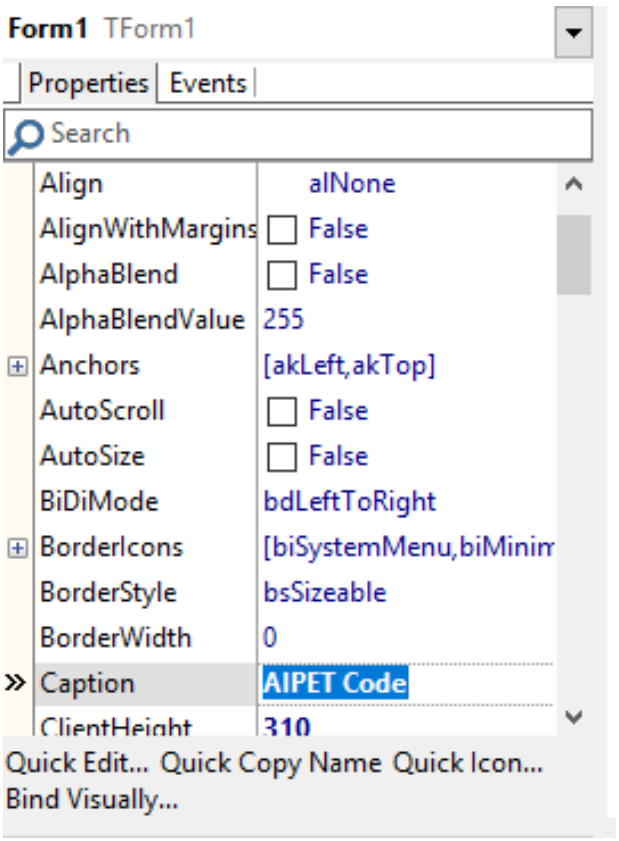

Рисунок 22 – Вкладка Properties

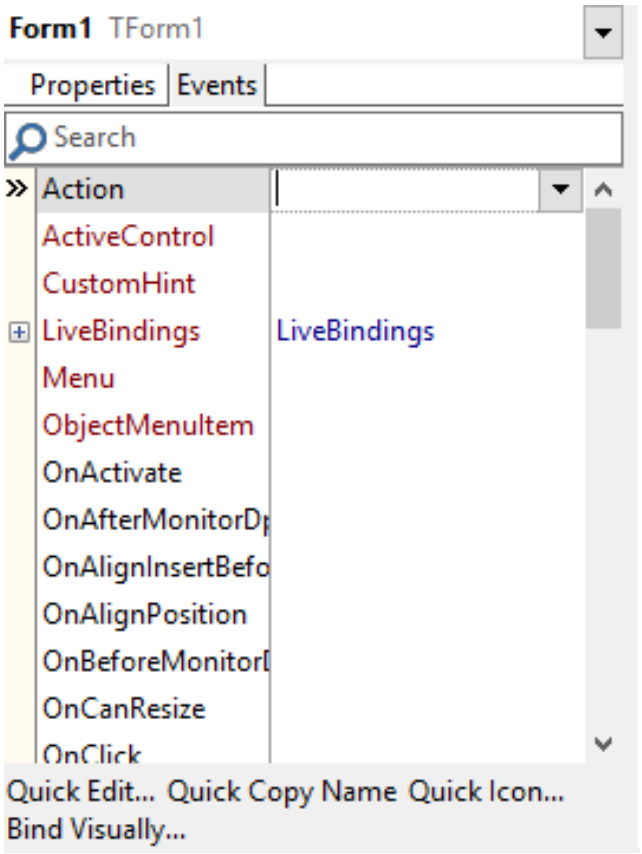

Рисунок 23 – Вкладка Events

Таким образом, используя компоненты из панели инструментов и настройки инспектора объектов можно создать форму, интерфейс программного обеспечения, и попробовать запустить ее. Запуск осуществляется нажатием кнопки Запуск (Run) или сочетанием клавиш Shift+Ctrl+F9. Кнопка запуска имеет следующий вид:  . Запущенная форма представлена на рисунке 24.

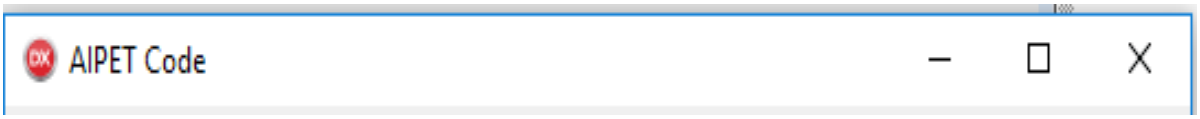

Рисунок 24 – Окно программы после ее запуска

В этой части моего дипломного проекта были показаны начальные и базовые этапы создания программного обеспечения в среде разработки RAD Studio.

3.2 Разработка программного обеспечения AIPET Code

При разработке программного обеспечения используются перечисленные ранее компоненты из Панели инструментов, вкладка Standard. Чтобы создать интерфейс программного обеспечения, кликаем на необходимые компоненты и перемещаем их на окно формы. Размещаем компоненты там, где они должны находиться и производим их настройку. Таким образом, в результате произведенных действий на выходе получается следующий интерфейс программного обеспечения (рисунок 25).

Рисунок 25 – Пользовательский интерфейс программного обеспечения

Я изменила название формы на AIPET Code в Инспекторе объектов, в вкладке Properties, поле Caption. Аналогичным образом были изменены и

настроены другие компоненты формы. Также, в Properties была изменена иконка приложения, в поле Icon было выбрано подходящее изображение в формате .ico (рисунок 26).

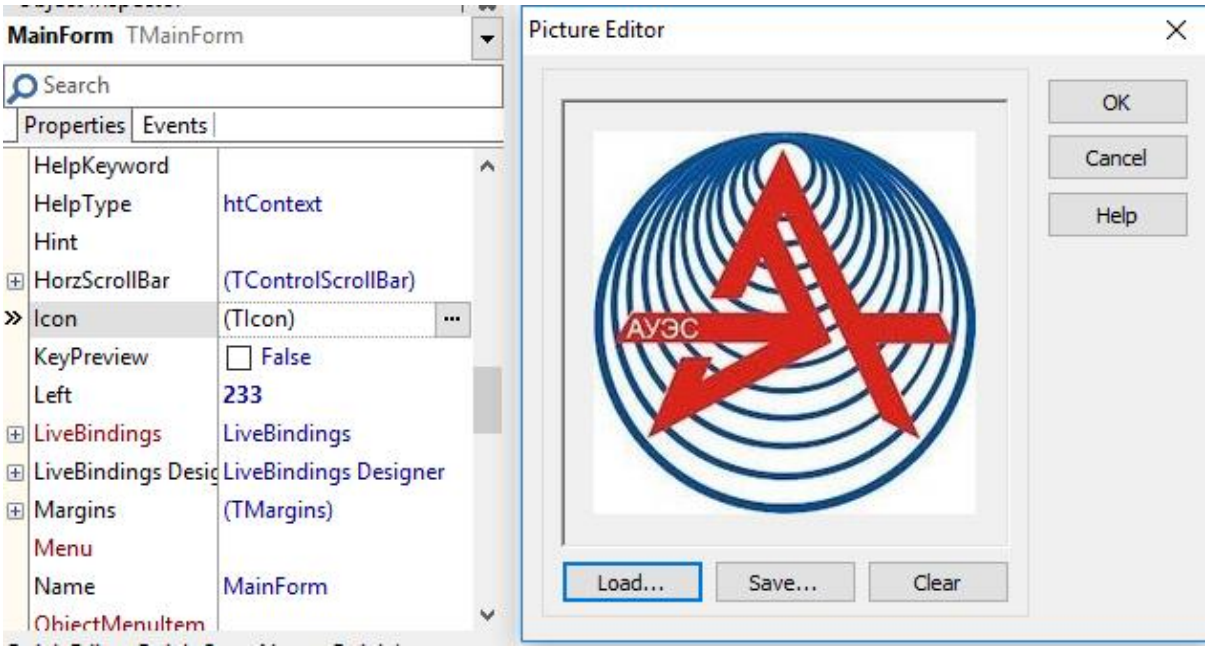

Рисунок 26 – Изменение иконки программного обеспечения

После сбора и настройки всех компонентов формы, на выходе получаем пользовательский интерфейс программного обеспечения или его внешнюю оболочку (рисунок 27). Как видно на рисунке 27 пользовательский интерфейс выполнен в выдержанном стиле, все нужные кнопки видны и доступны, надписи и подсказки будут понятны любому пользователю.

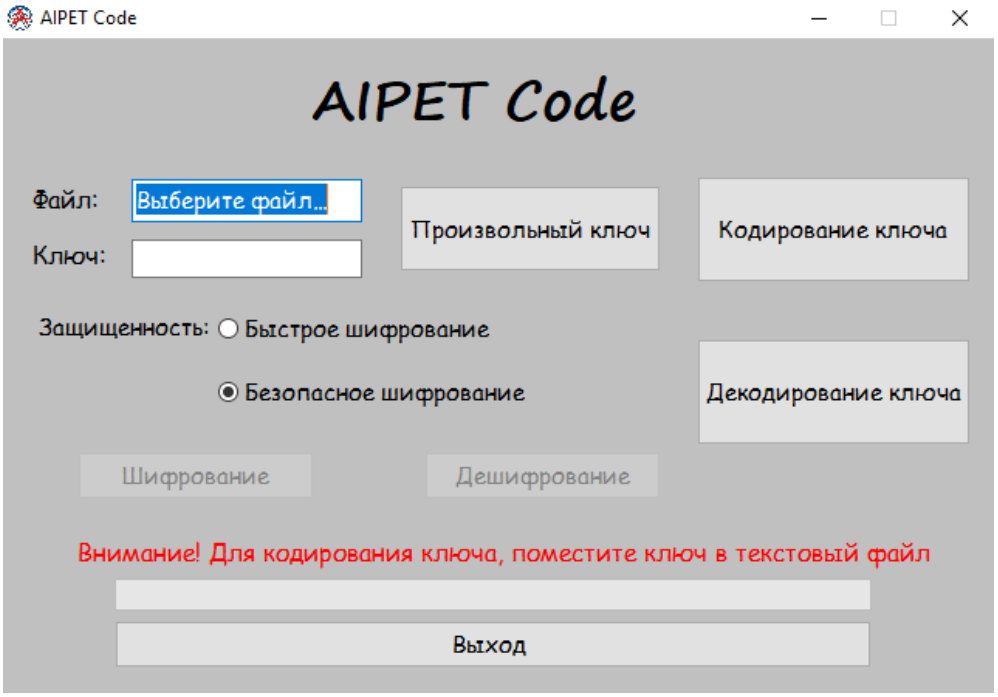

Рисунок 27 – Внешний вид программного обеспечения

Собранный интерфейс программного обеспечения имеет следующую структуру (рисунок 28). Структура формы показывает все ее компоненты, ее можно просмотреть в правом верхнем углу рабочего пространства среды разработки RAD Studio. Каждый компонент, отвечает за определенное действие. Окошко называется Structure.

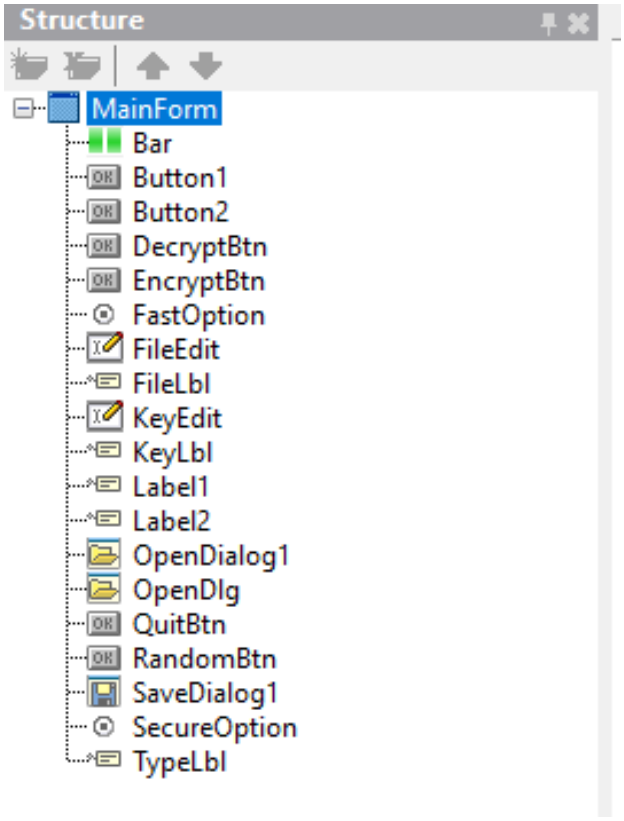

Рисунок 28 – Структура пользовательского интерфейса программного обеспечения

Как уже упоминалось ранее, каждый компонент формы отвечает за определенное действие, поэтому каждый компонент важен. Чтобы кнопки, расположенные на форме, выполняли действие, необходимо их запрограммировать. Нужно написать программный код для каждой кнопки, чтобы она могла выполнять свою функцию.

Компонент TButton как раз и является кнопкой. В моем программном обеспечении кнопки выполняют, как и специфичные, так и общий функции. Объясним это на примере нескольких кнопок.

К примеру, кнопка «Произвольный ключ» отвечает за генерацию ключей для шифрования информации. Чтобы данный элемент генерировал ключ, пишется программный код. Чтобы написать программный. код необходимо произвести настройки в вкладке Events инспектора объектов (рисунок 29).

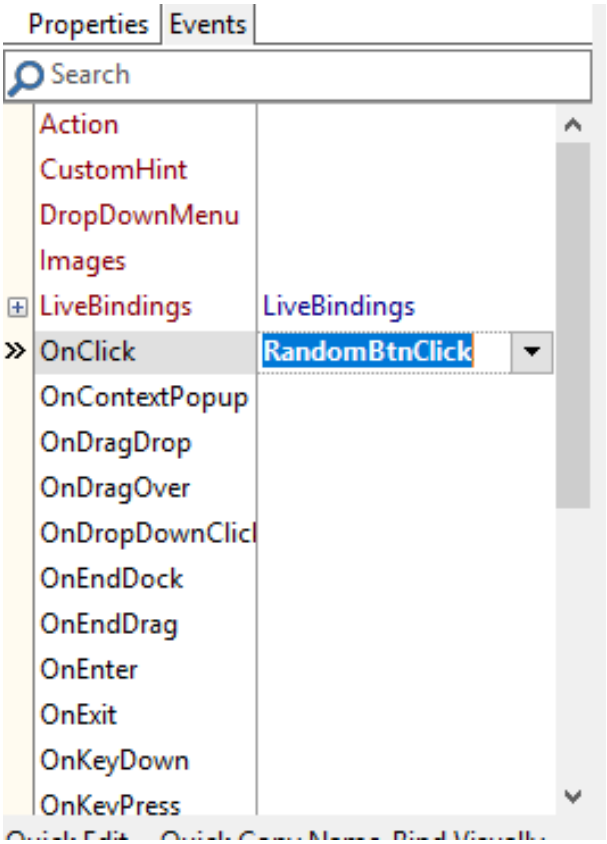

Рисунок 29 – Настройка действия и сущности кнопки

Чтобы написать программный код для этой кнопки, в режиме Design нужно два раза нажать на необходимый элемент и откроется окно с областью для кодирования. Например, кликая два раза по элементу кнопки «Произвольный ключ» можно наблюдать следующее (рисунок 30).

```
procedure
TMainForm.RandomBtnClick(Sender: TObject);
begin
```

Рисунок 30 – Программный код кнопки «Произвольный ключ»

Аналогичным образом программируются и другие компоненты формы, и в результате мы получаем цельное, полнофункциональное программное обеспечение. Скриншоты реализации кодирования элементов формы представлены ниже (рисунок 31-35).

```
begin
  OpenDialog1.Title:='Выбор файла-ключа для кодирования';
```

Рисунок 31 – Программный код кодирования ключа

```
begin
 OpenDialog1.Title:='Выбор файла-ключа для декодирования';
 SaveDialog1.Title:='Укажите имя декодированного файла';
```

Рисунок 32 – Программный код декодирования ключа

Важно отметить, что кодирование и декодирование ключа происходит по алгоритму Base64.

```
procedure TMainForm.EncryptBtnClick(Sender: TObject);
 const
```

Рисунок 33 – Программный код реализации шифрования передаваемого файла

```
begin
if Sender is TButton then with TButton(Sender) do
try
```

Рисунок 34 – программный код реализации дешифрования полученного файла

Для шифрования и дешифрования передаваемой по открытому каналу связи информации использовался алгоритм LEA-128 SEA.

```
procedure TMainForm.QuitBtnClick(Sender: TObject);
 begin
 Close;
 end;
```

Рисунок 35 – программный код кнопки выхода из приложения

На данном этапе необходимо детальное представление работы программного обеспечения. Разработчик должен четко и ясно представлять, как будет осуществлена программная реализация работы алгоритмов, так ка даже самая небольшая ошибка нарушит процесс шифрования и кодирования ключей.

При написании программного кода системы для каждой из кнопок объявлялась своя процедура, что значительно упрощает обращение к отдельным частям кода во время компиляции и запуска программного обеспечения.

Выше представлены фрагменты исходного программного кода элементов формы, кнопок, каждая из которых отвечает за определенное действие. Правильно реализовав запланированный функционал, на выходе можно получить готовое, работающее программное обеспечение.

3.3 Тестирование разрабатываемого программного обеспечения

Проведем тестирование программного обеспечения. Допустим, абонент А хочет передать по открытому каналу связи, в нашем случае через электронную почту, файл абоненту Б. Исходный файл – это файл с

расширением .txt, содержащий определенный текст (рисунок 36). Но абонент А хочет сделать передачу максимально защищенной, при минимальных затратах.

Добрый день! Это пример шифрование файла, используя алгоритм LEA-128 SEA.

Рисунок 36 – Исходный файл для передачи

Для этого необходимо выполнить несколько шагов:

1) На стороне абонента А запускаем программное обеспечение. После запуска абоненту открывается главное окно программы (рисунок 37)

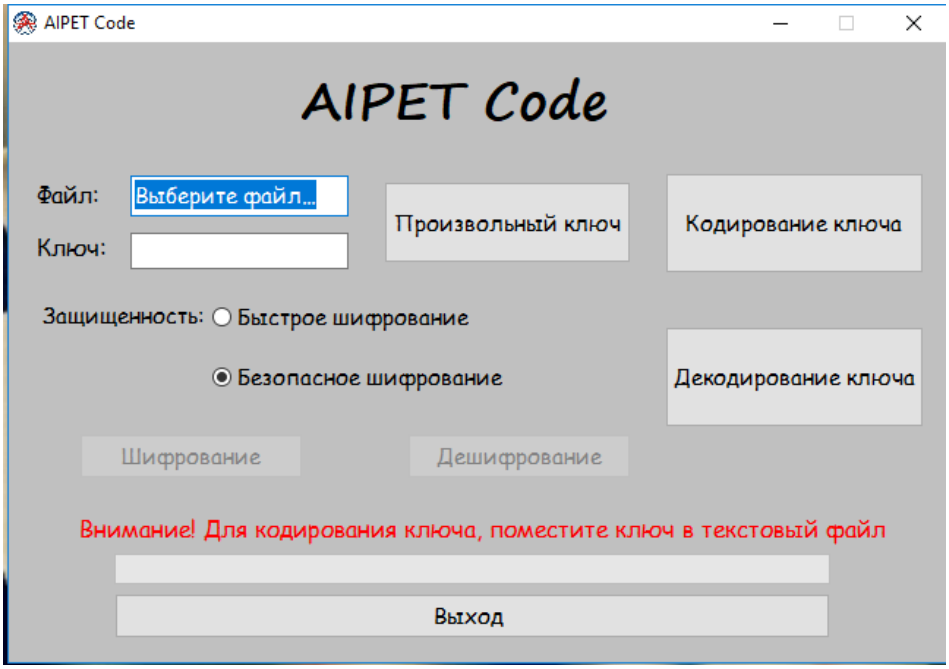

Рисунок 37 – Главное окно программы

2) Далее абонент А выбирает файл, который он хочет зашифровать. Шифрование файла происходит по алгоритму LEA-128 SEA (рисунок 38).

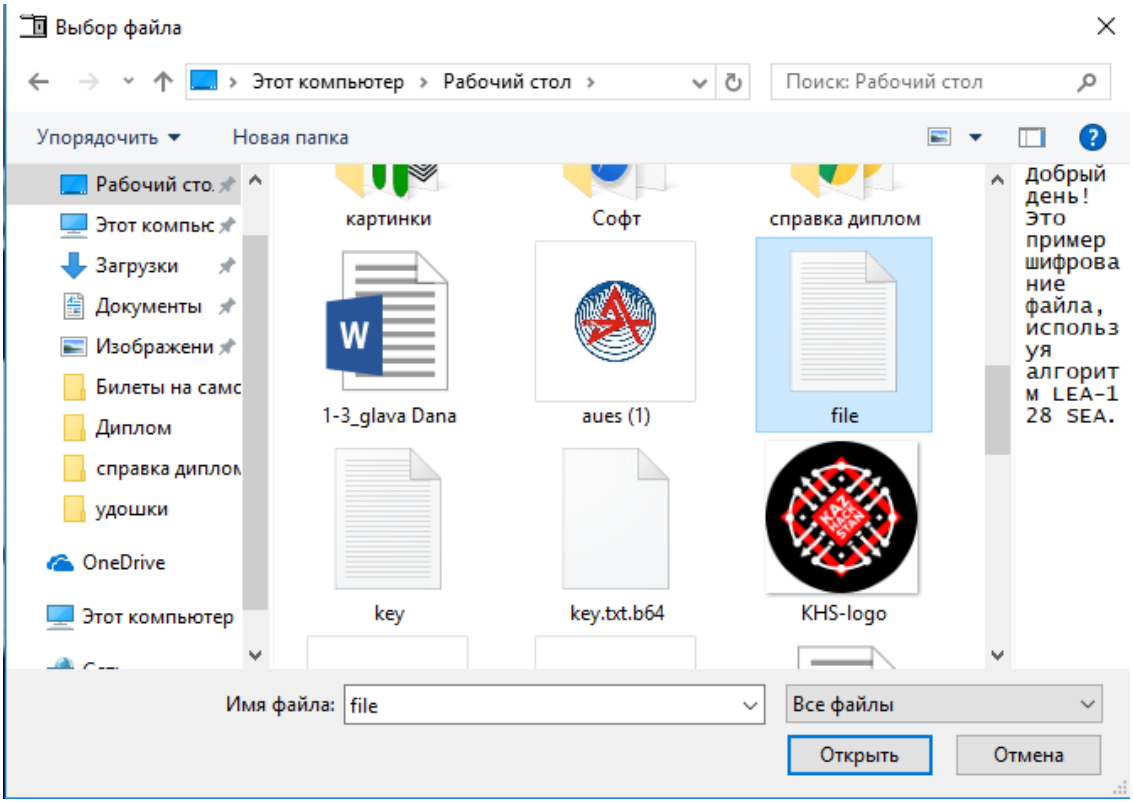

Рисунок 38 – Выбор файла для шифрования и передачи

3) Нажав на кнопку «Произвольный ключ», абонент А генерирует ключ, который будет использоваться в шифровании файла (рисунок 39). Важно на этом этапе скопировать и сохранить ключ в .txt – файл (рисунок 40).

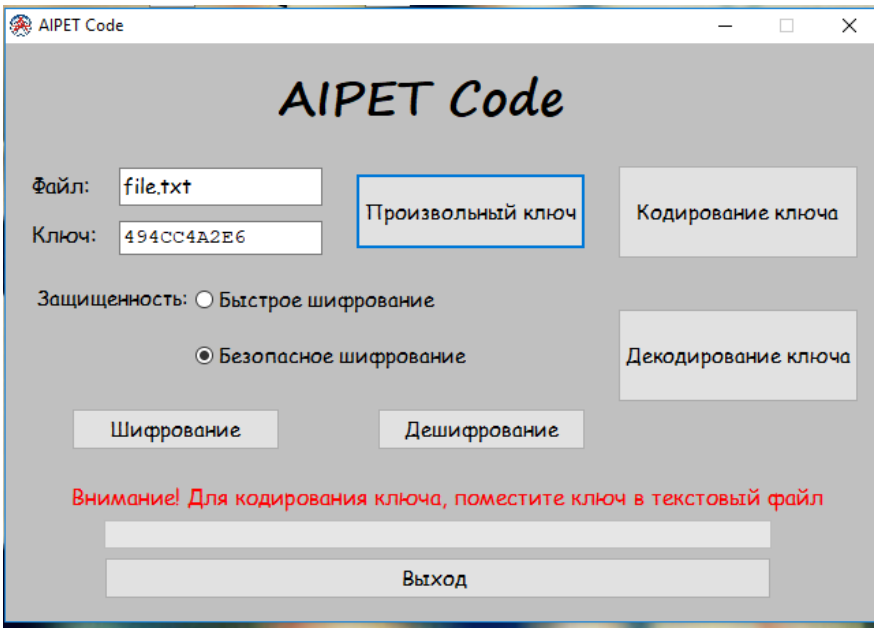

Рисунок 39 – Генерация ключа

Рисунок 40 – Файл, для сохранения ключа

4) Далее абонент А выбирает тип шифрования: быстрое или безопасное. Выбор зависит от критичности передаваемой информации (рисунок 41).

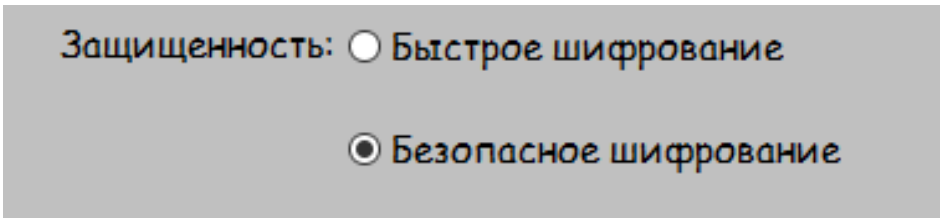

Рисунок 41 – Выбор типа шифрования информации

5) Абонент А нажимает на кнопку «Шифрование» и тем самым происходит шифрование передаваемого файла (рисунок 42). Результат шифрования сохранится в том же исходном файле (рисунок 43).

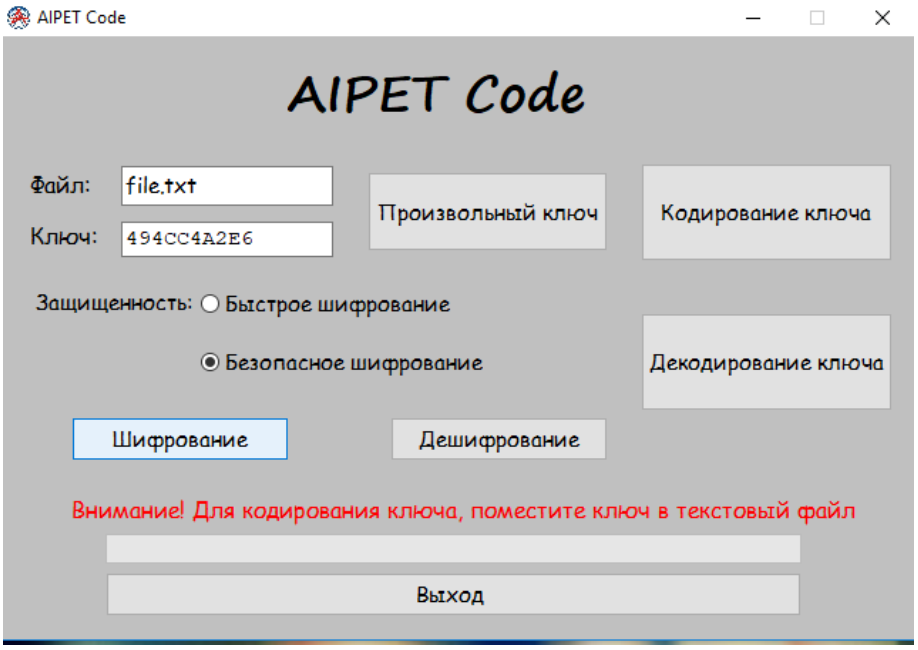

Рисунок 42 – Запуск шифрования исходного файла

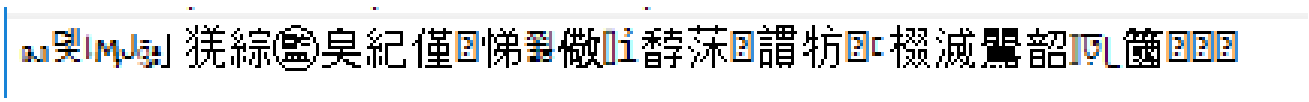

Рисунок 43 – Зашифрованный передаваемый файл

6) Теперь абонент А произведет кодирование ключа, который ранее использовался в шифровании передаваемого файла. Нажимает кнопку «Кодирование ключа». Где открывается диалоговое окно выбора файла с ключом (рисунок 44).

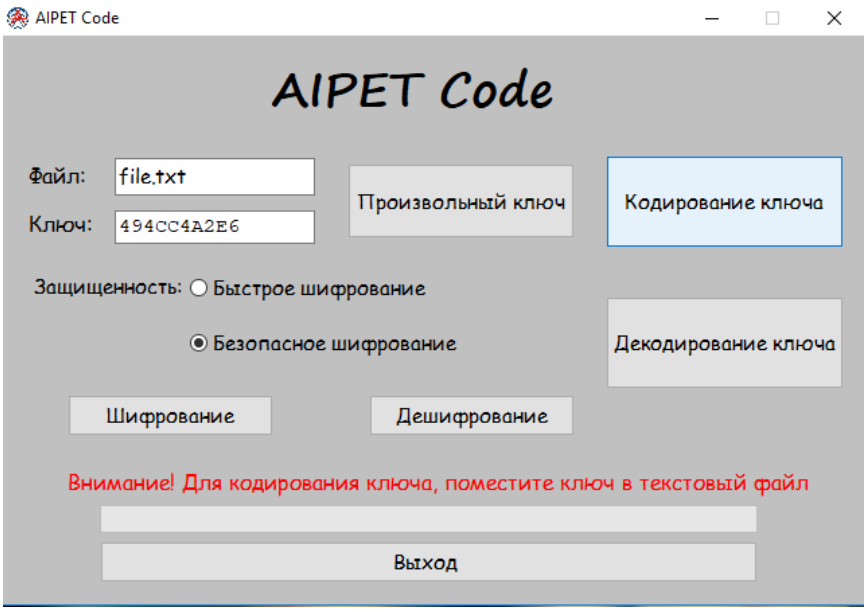

Рисунок 44 – Запуск кодирования ключа

7) Абонент А выбирает файл с ключом (рисунок 45).

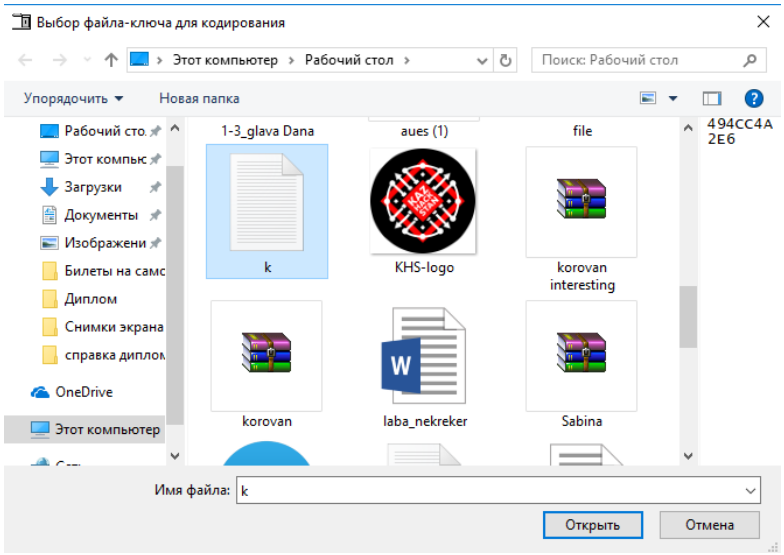

Рисунок 45 – Выбор файла с ключом

8) После кодирования ключ сохраняется с расширением .txt.b64 (рисунок 46)

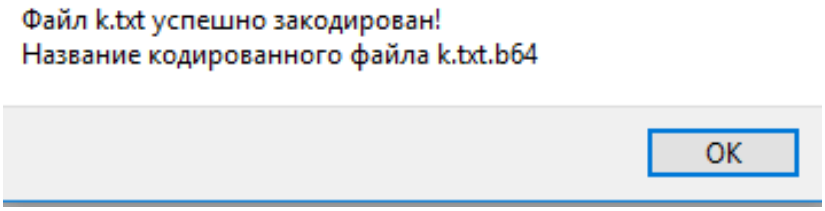

Рисунок 46 – Файл ключа с расширением .txt.b64

9) Теперь абонент А отправляет пару: зашифрованный файл и ключ абоненту Б через электронную почту (рисунок 47).

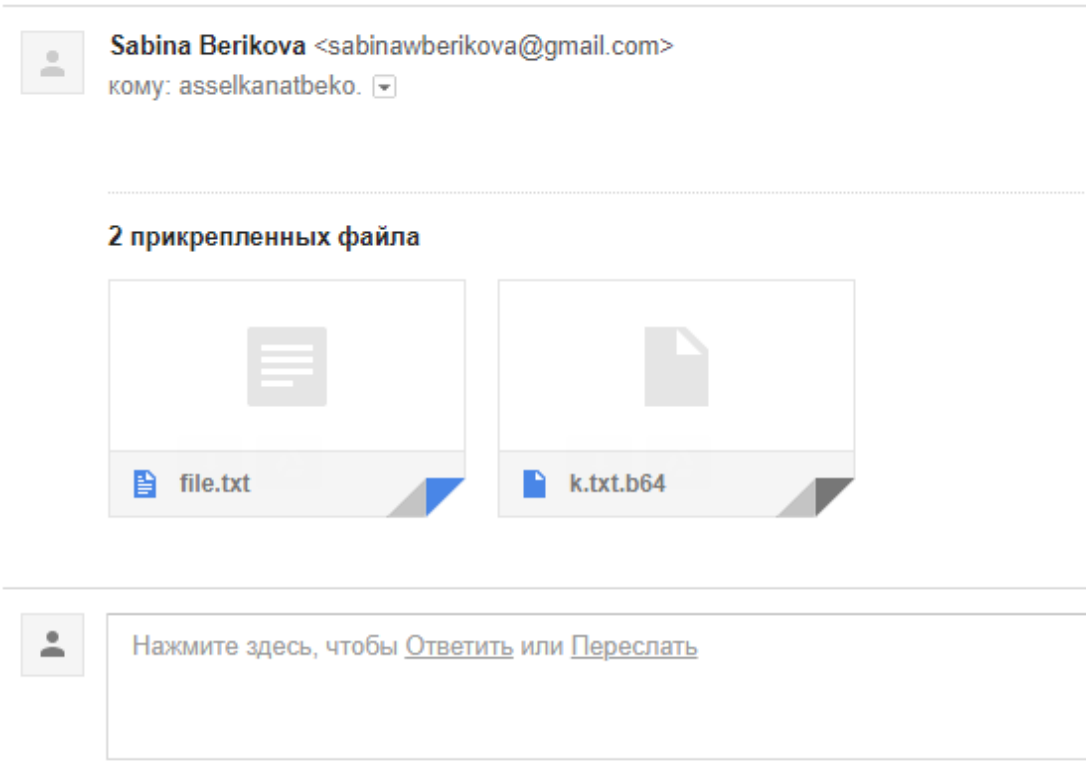

Рисунок 47 – Отправленные зашифрованный файлы от абонента А абоненту Б

10) Абонент Б получает в электронном письме пару: зашифрованный файл и ключ (рисунок 48).

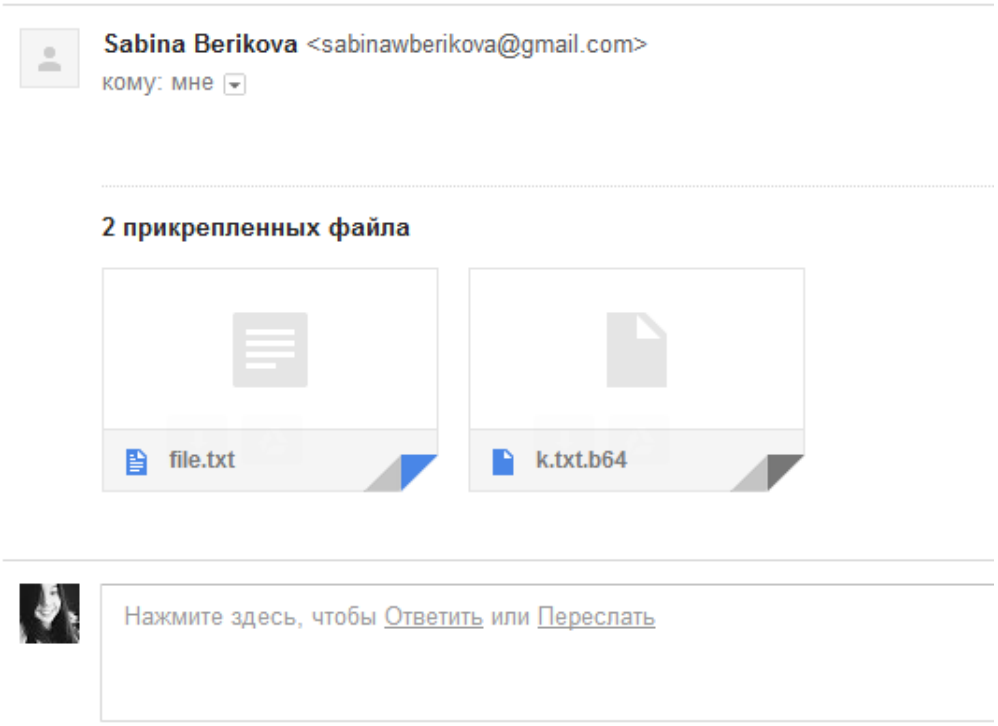

Рисунок 48 – Полученные зашифрованные файлы абонентом Б от абонента А

11) На своей стороне абонент Б запускает программное обеспечение AIPET Code.

12) Абонент Б выбирает зашифрованный файл (рисунок 49).

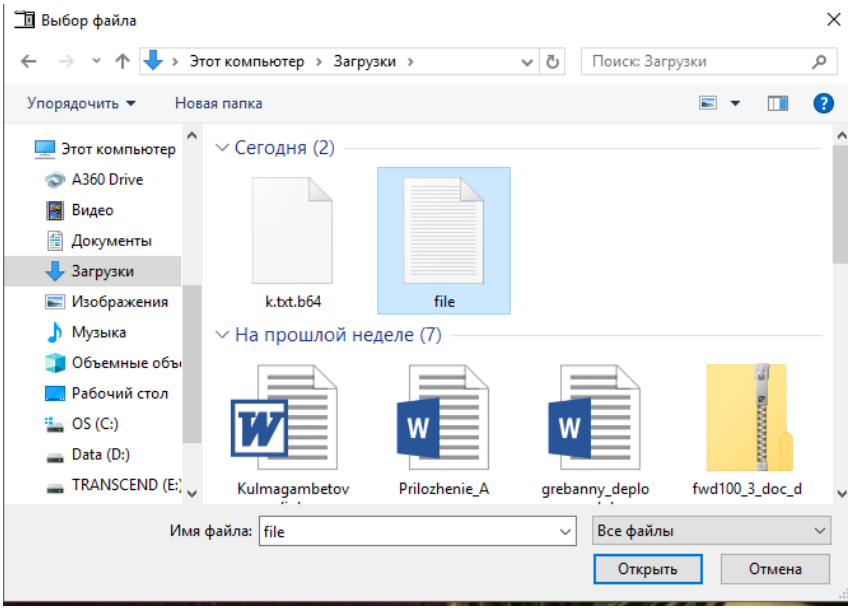

Рисунок 49 – Выбор зашифрованного файла абонентом Б

13) Вначале необходимо декодировать ключ. Абонент Б нажимает на кнопку «Декодирование ключа». Открывается диалоговое окно с выбором файла с ключом (рисунок 50).

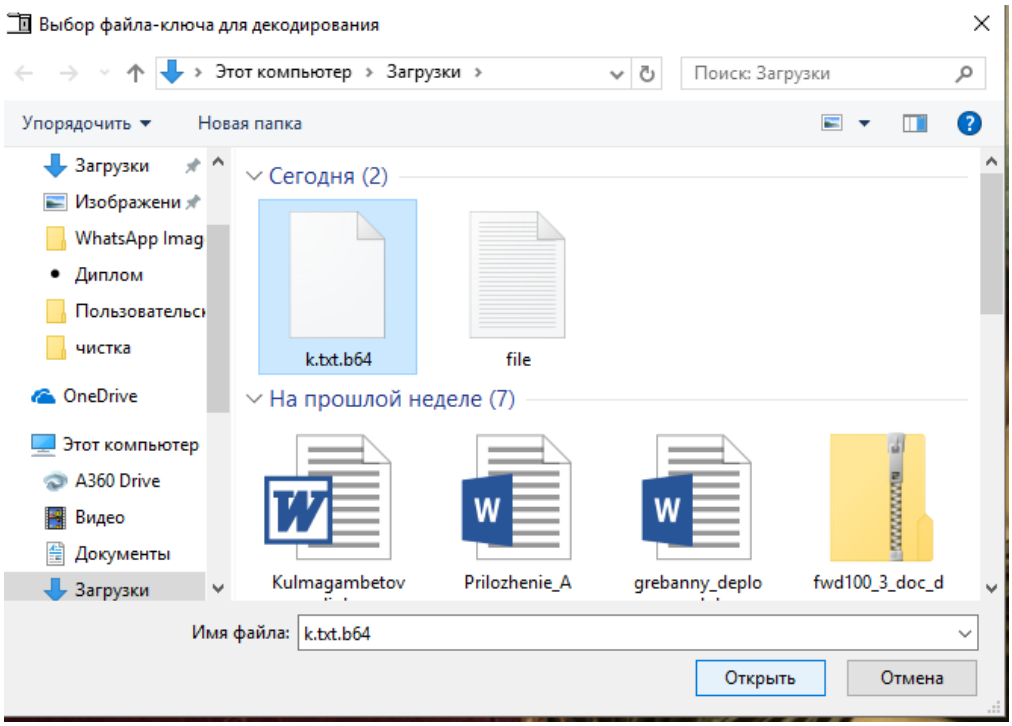

Рисунок 50 – Выбор закодированного файла с ключом, для его дальнейшей декодировки

14) После декодировки ключ сохраняется в файл с .txt расширением. Открыв который можно увидеть ключ (рисунок 51).

Рисунок 51 – Декодированный абонентом Б ключ

15) Абонент Б копирует ключ с декодированного файла и вставляет в поле ключа в окне программного обеспечения (рисунок 52).

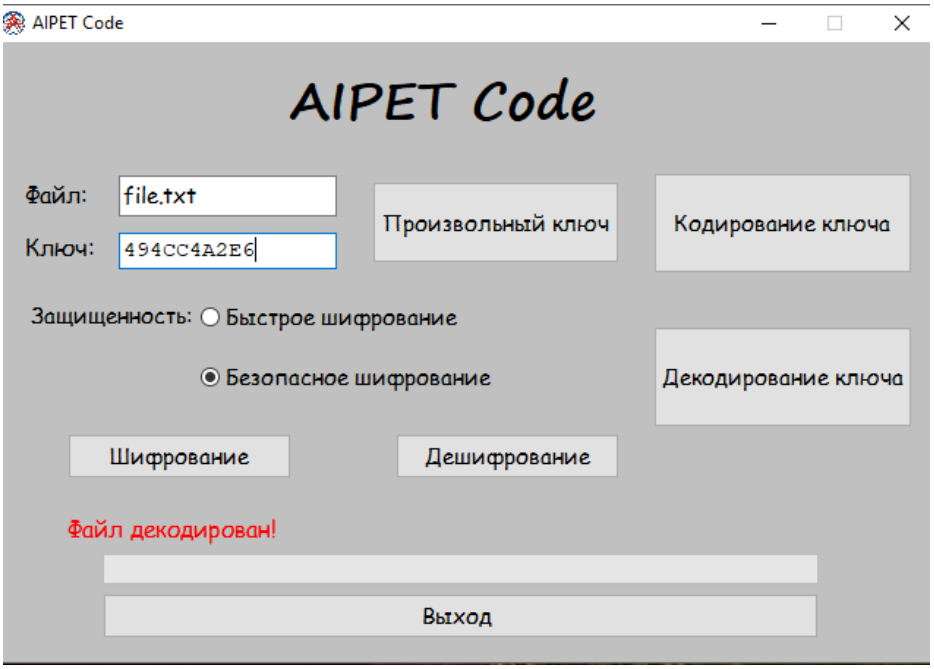

Рисунок 52 – Ключ декодирован и готов для дешифрования исходного файла

16) Теперь выбрав также режим безопасного шифрования, абонент Б нажимает на кнопку «Дешифрование» и получает дешифрованный файл (рисунок 53).

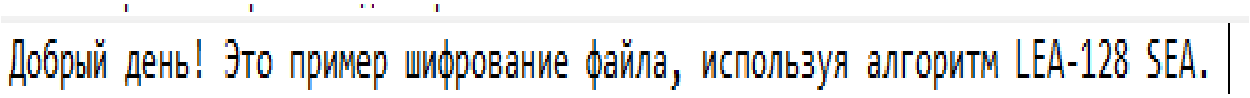

Рисунок 53 – Дешифрованный абонентом Б файл

Как видно из приведенных доказательств, передача файлов прошла успешно. На приемной стороне исходный файл был дешифрован и идентичен тому файлу, который шифровался на стороне абонента А.

Ранее, во второй части дипломного проекта был представлен скриншот, показывающий нагрузку на центральный процесс при использовании в работе существующие аналоги моего программного обеспечения. Нагрузка на центральный процесс составляла 50%. После написания программного обеспечения, во время использования его в работе нагрузка на центральный процессор составила 40%, что подтверждает легковесность моего программного обеспечения (рисунок 54).

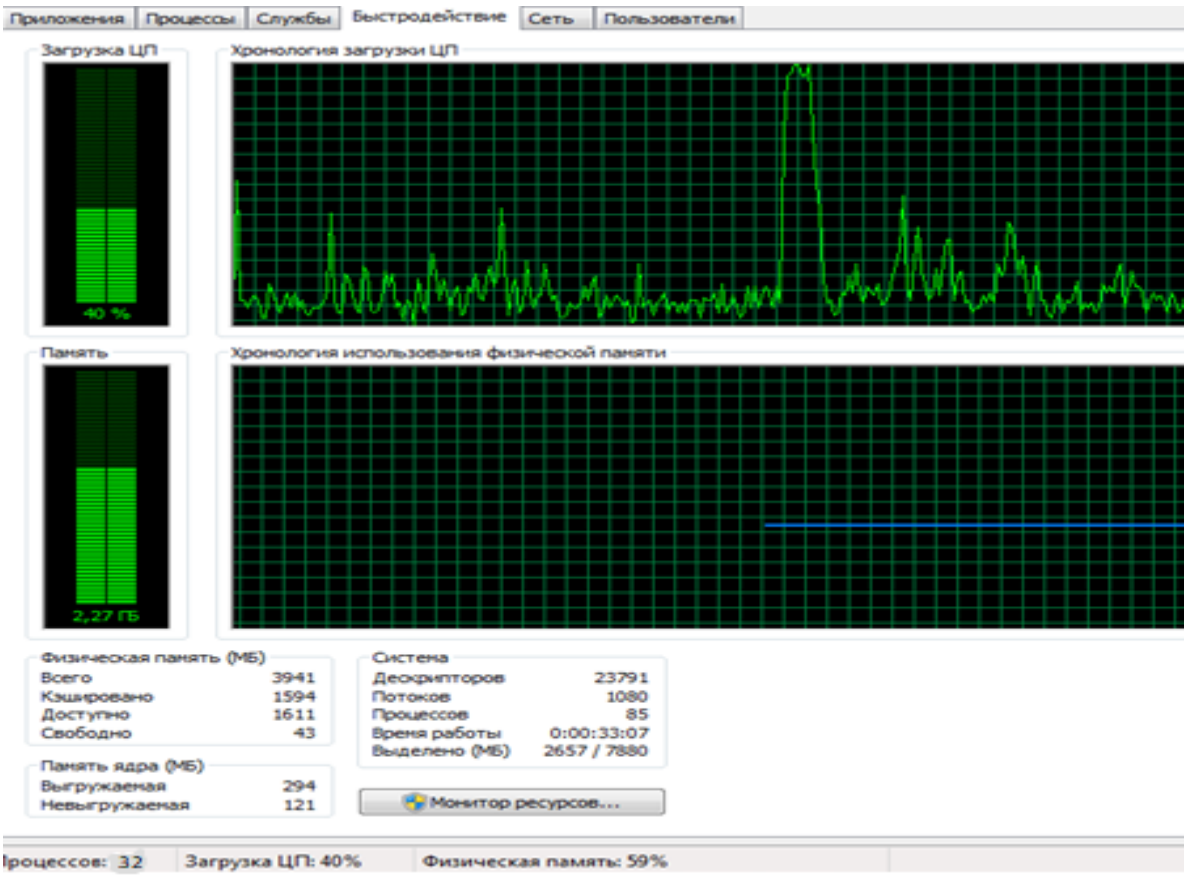

Рисунок 54 - Нагрузка на ЦП при полной работоспособности ПК: 40%

Вывод

В результате проведенной работы актуальность темы была доказана. В сегодняшних условиях важности информационной безопасности, важно использовать криптографические меры защиты для организации безопасной передачи данных.

Мною было скомпилировано программное обеспечение AIPET Code для шифрования передаваемой информации по открытым каналам связи в среде разработки RAD Studio Berlin 10. Программное обеспечение прошло этап тестирования и готово к эксплуатации.

4 Технико-экономическое обоснование

В данном дипломном проекте мною была спроектирована программа, обеспечивающая шифрование, передаваемой по открытому каналу связи, информации с применением кодирования ключа. В процессе работы использовались алгоритмы шифрования LEA 128 SEA и Base64. Шифрование необходимо для защиты информации от неправомерного доступа к ней третьих лиц и для защиты ее целостности. Так как сейчас происходит большой обмен информацией через интернет, в частности через электронную почту, организация надежной защиты передаваемой информации является одной из важных проблем пользователей.

Спроектированное мною программное обеспечение осуществляет генерацию ключей для шифрования информации и ключей для кодирования использованных на предыдущем шаге ключей. В данной части моего дипломного проекта будут рассчитаны все затраты на создание программного обеспечения, включающие в себя затраты на устройства и используемое программного обеспечение, затраты на человеческие ресурсы и электроэнергию.

4.1 Расчет трудоемкости разработки ПО

Трудоемкость разработки программного обеспечения может быть определена приведенным списком, содержащим все основные этапы работы, необходимые к выполнению. Разделенные этапы работы и требуемое для их выполнения время представлено в таблице 4.1.

Таблица 4.1 – Распределение этапов работы и оценка их трудоемкости

Этапы разработки программного обеспечения	Виды работ	Трудоемкость разработки, чел. х ч.
1 этап	Постановка задач к выполнению	8
2 этап	Разработка и утверждение технического задания на разработку программного обеспечения	10
3 этап	Ознакомление с существующими алгоритмами шифрования	22
4 этап	Поиск и ознакомление	10

Продолжение таблицы 4.1

Этапы разработки программного обеспечения	Виды работ	Трудоемкость разработки, чел. х ч.
	с технической литературой на тему шифрования	
5 этап	Оформление обзорной части по теме дипломного проекта	8
6 этап	Написание практической части дипломного проекта	24
7 этап	Анализ и выбор среды разработки программного обеспечения	5
8 этап	Реализация дипломного проекта	35
9 этап	Отладка программного обеспечения	4
10 этап	Оформление практической части дипломного проекта в виде отчета о выполненной работе	10
11 этап	Тестирование дипломного проекта на ошибки	10
12 этап	Внедрение программного обеспечения	6
Итого:		152

Количество часов в одном рабочем дне – 8 часов. Таким образом, количество рабочих дней, потраченных на создание дипломного проекта – 19 дней.

4.2 Расчет затрат на разработку программного обеспечения

Затраты на разработку программного обеспечения можно произвести, основываясь на смете, включающей в себя следующие пункты:

- материальные затраты;

- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;
- прочие затраты.

В свою очередь, материальные затраты состоят из основных и вспомогательных материальных ресурсов необходимых для разработки программного обеспечения.

Расчет затрат производится, используя данные, приведенные в таблице 4.2.

Таблица 4.2 – Затраты на материальные ресурсы

Наименование	Марка	Ед. измерения	Количество	Цена за ед. (тг)	Сумма (тг)
Офисная бумага А4 500 лист.	Белоснежка	пачка	1	1000	1000
Блокнот 96 лист.		штук	1	1500	1500
Ручка		штук	3	150	450
Карандаш простой		штук	3	100	300
Итого (тг):	3250				

Так как мною был куплен новый ноутбук Lenovo Ideapad 110, в нем уже заранее было предустановлено необходимое системное и прикладное программное обеспечение, поэтому при проектировании не предусмотрены затраты на приобретение лицензионной операционной системы Windows 10 и офисного пакета Microsoft Office. Также мною была найдена в свободном доступе версия среды разработки RAD Studio Berlino 10.

Таблица 4.3 – Затраты на системное и прикладное программное и аппаратное обеспечение

Наименование	Марка	Ед. измерения	Количество	Цена за ед. (тг)	Сумма (тг)
Ноутбук	Lenovo Ideapad 110 core i7	штук	1	210 990	210 990
Принтер	HP LaserJet 3055	штук	1	35 000	35 000

Продолжение таблицы 4.3

Наименование	Марка	Ед. измерения	Количество	Цена за ед. (тг)	Сумма (тг)
Операционная система	Windows 10	штук	1	-	-
Среда разработки	Embarcadero RAD Studio Berlino 10	штук	1	-	-
Офисный пакет	Microsoft Office 2013	штук	1	-	-
Модем	TP-Link TL-WR740N	штук	1	9850	9850
Итого (тг):	254 940				

Общая сумма затрат на материальные ресурсы, необходимые для создания программного обеспечения ($З_м$) определяется по формуле 4.1:

$$З_м = \sum P_i \times Ц_i , \quad (4.1)$$

где P_i – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;

n – количество видов материальных ресурсов

$$З_м = 1000+1500+450+300+210990+35000+9850 = 259\ 090 \text{ тг.}$$

Материальные затраты на реализацию дипломного проекта составят 259 090 тенге (тг.). Все материальные затраты лягут на основные средства.

4.3 Расчет затрат на электроэнергию

Поскольку в процессе производства используется электрооборудование, необходимо рассчитать затраты на электроэнергию. Затраты на электроэнергию для производственных нужд включают в себя расходы электроэнергии на оборудование и дополнительные нужды.

Время работы оборудования для разработки ПП берется равным 152 часов для ноутбуков и модема, данное количество часов было рассчитано в таблице 4.1. Для принтера время работы для разработки ПП берется равным 6 часов, так нет необходимости постоянного его использования.

$$Э = З_{эл.эн.обор} + З_{доп.нуж} , \quad (4.2)$$

где $З_{эл.эн.обор}$ – затраты на электроэнергию оборудования;

$З_{доп.нуж.}$ – затраты электроэнергии на дополнительные нужды.

Расходы электроэнергии на оборудование рассчитывается по формуле:

$$З_{эл.эн.обор} = \sum W \times К_{исп} \times S \times T, \quad (4.3)$$

где W – потребляемая мощность, Вт;

$К_{исп}$ – коэффициент использования ($К_{исп}$ = 0,7..0,9);

T – время работы;

S – тариф (1кВт/ч = 16,65тг).

Сводные результаты расчета затрат на электроэнергию представлены в таблице 4.4.

Таблица 4.4 – Затраты на электроэнергию

Наименование приборов	Паспортная мощность, кВт	Коэффициент мощности	Время работы оборудования, ч	Цена ЭЭ тг/ кВтч	Сумма, тг
Ноутбук	0,6	0,7	152	16,65	1062,94
Модем	0,08	0,9	60	16,65	71,93
Принтер	0,5	0,9	6	16,65	44,95
Кондиционер	0,8	0,9	120	16,65	1438,56
Освещение	0,3	0,7	100	16,65	349,65
Итого	2968,03				

$З_{эл.эн.обор}$ = 1062,94+ 71,93 + 44,95 + 1438,56 + 349,65 = 2968,03 (тенге)

Затраты на дополнительные потребности берутся по укрупненному показателю в размере 5% от затрат на оборудование:

$$З_{доп.нуж} = 5\% \times З_{эл.эн.обор}, \quad (4.4)$$

Затраты на дополнительные потребности рассчитаны по формуле (4.4):

$З_{доп.нуж}$ = 0,05 × 2968,03 = 148,40 (тенге)

Таким образом суммарные затраты на электроэнергию составляют:

Э = 2968,03 + 148,40 = 3116,43 (тенге)

4.4 Расчет затрат на оплату труда

Над разработкой проекта работают два сотрудника:

- руководитель проекта – изучение предметной области, анализ требований к системе, внедрение и поддержка;
- разработчик – создание модели, реализация модели, тестирование продукта.

Общая сумма затрат на оплату труда ($З_{тр}$) определяется по формуле:

$$З_{тр} = \sum ЧС_i \times T_i, \quad (4.5)$$

где $ЧС_i$ – часовая ставка i-го работника, тг;

Ti – трудоемкость разработки модели, чел.×ч;

i – категория работника;

n – количество работников, занятых разработкой ПП.

На этапах разработки, участники разработки задействованы неравноценно, для этого необходимо рассчитать часовую ставку работника, а затем общий размер заработной платы [7, с.77-86].

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.6)$$

где Зпi – месячная заработная плата i-го работника, тг;

ФРВi- месячный фонд рабочего времени i-го работника, час

Месячная заработная плата сотрудников:
Профессор – 160 000 тг;
Разработчик – 80 000 тг.

$ЧС_i = 160\,000 / 22 \times 8 = 909{,}09$ тг/ч
$ЧС_i = 80\,000 / 22 \times 8 = 454{,}54$ тг/ч

Часовая ставка научного руководителя составляет 909,09 (тг/ч), трудоемкость разработки – 50 ч. Часовая ставка разработчика составляет 454,54 (тг/ч), трудоемкость разработки – 152 ч.

Рассчитаем общую сумму затрат на оплату труда по формуле (4.5):

$З_{тр} = 909{,}09 \times 50 + 454{,}54 \times 152 = 45\,254{,}5 + 69\,090{,}08 = 114\,344{,}58$ тг.

Сводные результаты расчета затрат на оплату труда показаны в таблице 4.5.

Таблица 4.5 – Расчёт основной заработной платы разработчиков.

Категория работника	Квалификация	Трудоемкость разработки ПП, час.	Часовая ставка, тг/ч	Сумма, тг.
Научный руководитель	Профессор	50	909,09	45 254,5
Разработчик	Студент	152	454,54	69 090,08
Итого				114 344,58

4.5 Расчет затрат по социальному налогу

Социальный налог – согласно Налоговому кодексу Республики Казахстан составляет 9,5 % от ФОТ (фонда оплаты труда). Следует отметить, что пенсионные отчисления не облагаются социальным налогом.

$$С_н = (ФОТ - ПО) \times 0{,}095, \quad (4.7)$$

где ПО - отчисления в пенсионный фонд, 10% от ФОТ.

Социальный налог рассчитываем по формуле (4.7):

ПО = 114 344,58 × 0,1 = 11 434,46 тенге;
$С_н$ = (114 344,58 – 11 434,46) × 0,095 = 9776,46 тенге

Сводные результаты расчета затрат представлены в таблице 4.6.

Таблица 4.6 - Начисление социального налога

Категория работника	Количество человек	Заработная плата, тг	Пенсионные отчисления, тг	Социальный налог, тг
Научный руководитель	1	45 254,5	4525, 45	3869,26
Разработчик	1	69 090,08	6909,01	5907,20
Итого				9776,46

4.6 Амортизация основных фондов и прочие затраты

Годовые нормы амортизации ОФ принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования ОФ. Амортизация основных фондов определяется:

$$А_г = \frac{С_{об} \times Н_а}{100}, \quad (4.8)$$

где, $С_{об}$ – стоимость оборудования;
$Н_а$ – норма амортизации (норма амортизация = 20);

По формуле 4.8 рассчитаем сумму амортизационных отчислений за год для ноутбука:

$$А_г = \frac{210\,990 \times 20}{100} = 42\,198 \text{ тг}$$

Рассчитаем сумму амортизации за время разработки:

$$А_р = \frac{42\,198 \times 28}{365} = 3237{,}11 \text{ тг}$$

Аналогичным способом рассчитаем сумму амортизации для остального оборудования.

Результаты расчетов приведены в таблице 4.7

Таблица 4.7 - Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Сумма амортизации за год, тг	Сумма амортизации за время разработки, тг
Ноутбук	210 090	20	42 198	3237,11
Принтер	35 000	20	7000	536,99
Модем	9 850	15	1477,5	113,34
RAD Studio Berlin 10	0	10	0	0
Итого: амортизация основных средств			50 675,5	3887,44

На основании полученных данных по отдельным статьям составляется смета затрат на разработку ПП по форме, приведенной в таблице 4.8.

Таблица 4.8 – Смета затрат на разработку ПП

Статьи затрат	Сумма, тг
Затраты на оборудование	254 940
Затраты на программное обеспечение	0
Затраты на оплату труда	114 344,58
Социальные налоги	9776,46
Затраты на электроэнергию	2968,03
Амортизация основных фондов	3887,44
Прочие расходы (интернет)	22 500
Итого по смете	408 416,51

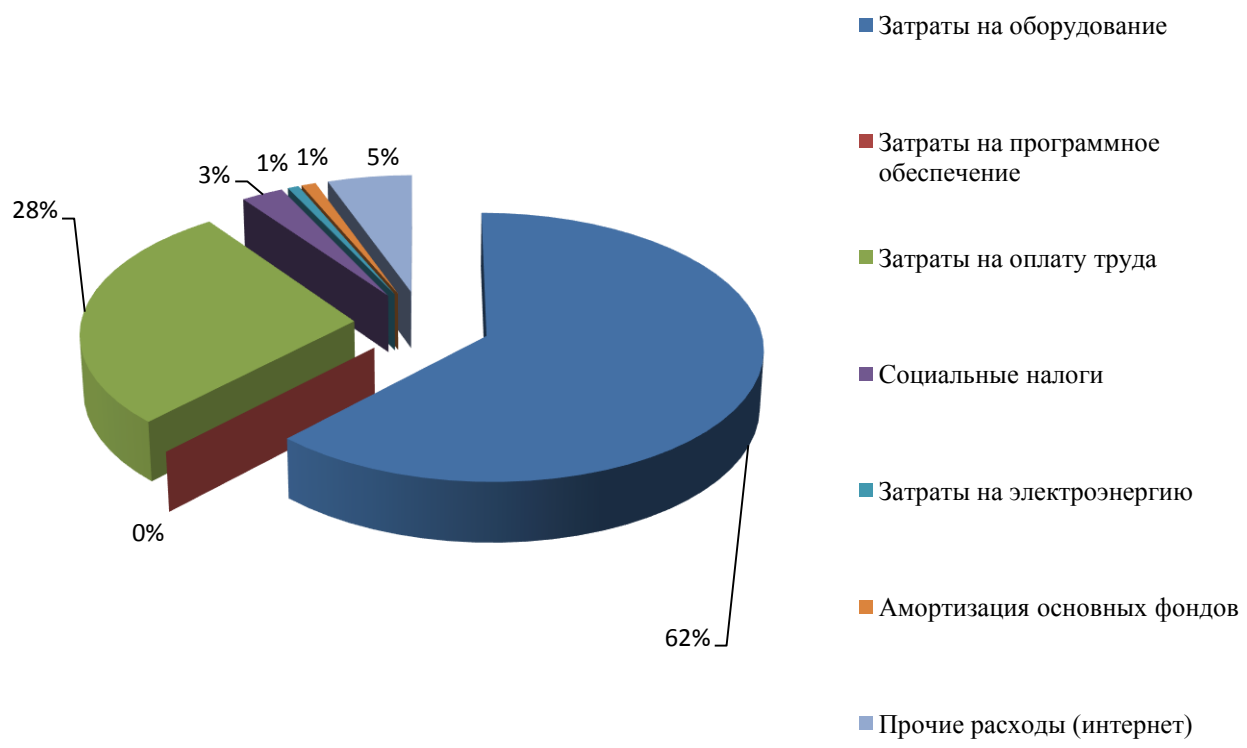

Рисунок 4.1 - Диаграмма структуры эксплуатационных затрат

4.7 Определение возможной (договорной) цены программного продукта

Величина возможной (договорной) цены программного продукта устанавливается на основе эффективности, качества и сроков её выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя [7, с.77-86].

Договорная цена ЦД для прикладных программных продуктов рассчитывается по формуле:

$$Ц_Д = З_{НИР}\left(1 + \frac{Р}{100}\right), \quad (4.9)$$

где $З_{НИР}$- затраты на разработку ПП, тг;

Р – средний уровень рентабельности ПП. % (принимается в размере 20-30% по согласованию с консультантом по экономической части).

$$Ц_Д = 408\,416{,}51 \times (1 + 20/100) = 490\,099{,}81 \text{ тенге}$$

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка (НДС) устанавливается законодательно. Налоговым Кодексом РК. На 2018 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_Д + Ц_Д \times НДС\ , \qquad (4.10)$$

490 099,81 + (490 099,81 × 0,12) = 490 099,81 + 58811,98 = 548 911,79 тенге

Рассчитанную возможную цену ПП можно округлить до 550 000 тенге.

4.8 Оценка социально – экономических результатов функционирования ЛВС

Экономическая потребность в этом плане будет развиваться из количественного и высококачественного компонента.

Экономический результат для разработчиков заключается в улучшении денежной позиции как отдельных разработчиков, так и предприятия, которое будет реализовывать проект.

Научный эффект характеризует получение новых научных знаний и отражает рост информации, предназначенной для внутриутробного потребления. Экономический эффект характеризуется выраженной экономичной экономией жизни и социального труда в общественном производстве, полученной с использованием результатов применения локальных сетей. Социальный эффект проявляется в улучшении условий труда, улучшении экологических показателей и т. [7, с.77-86].

Вывод

В этой главе были сделаны расчеты экономических затрат на приобретение необходимого оборудования и программного обеспечения для разработки программного обеспечения для шифрования информации, передаваемой по открытым каналам связи, включая расчет затрат на рабочую силу. Расходы на закуп оборудования были полностью рассчитаны; расчет сложности разработки программного продукта; расчет эксплуатационных расходов: социальный налог и пенсионные взносы, расходы на электроэнергию и амортизационные отчисления также были учтены.

Для конечного пользователя экономический результат станет исходить из: уменьшения затрат на использования оборудования, повышения экономической эффективности применения основных средств разработки. Качественный результат для покупателя заключается в том, что разрабатываемое программное обеспечение даст возможность обеспечения безопасной сессии при передаче данных по открытому каналу связи, что позволить уменьшить расходы в случае потери информации. Также был произведен расчет возможной договорной цены программного обеспечения, что составило 550 000 тенге, что является оптимальной ценой с точки зрения экономической эффективности.

5 Безопасность жизнедеятельности

5.1 Определение категории тяжести труда через интегральную бальную оценку

Условия труда - это набор факторов в производственной среде, которые влияют на здоровье и производительность человека во время трудового процесса. Факторы, составляющие условия труда можно разделить на 4 группы.

Первое, санитарно-гигиенические факторы - показатели, характеризующие рабочую среду рабочей зоны (влажность, температура воздуха, свет, шум и вибрация, электромагнитное излучение). Присутствие этих факторов определяется воздействием используемого оборудования и используемых процессов. Все показатели санитарно-гигиенических факторов подлежат количественной оценке и нормируются.

Второе, психофизиологические факторы (тяжесть труда) характеризуются физическим напряжением, нервным напряжением, скоростью работы, монотонностью труда и обусловлены самим трудом.

Третье, эстетические факторы (элементы) характеризуются цветовым дизайном рабочих мест и помещений, эстетизацией трудового процесса, продуктом труда, окружающей средой рабочей среды и определяющим восприятие рабочей среды и ее элементов трудящимися.

Четвертое, социально-психологические факторы характеризуются сплоченностью команды, межгрупповыми отношениями в команде и определяют психологический климат в этом рабочем коллективе. Что также является одним из основополагающих факторов из этих четырех.

Условия на рабочем месте оказывают значительное влияние на работу человека и его здоровье.

Чтобы предотвратить или уменьшить негативное воздействие вредных производственных факторов, увеличить эффективность, предотвратить появление профессиональных заболеваний, необходимо спланировать и осуществить меры по улучшению условий труда. Для этого необходимо проанализировать условия работы с определением категории тяжести работы.

Чтобы оценить влияние вредных факторов на здоровье и производительность, можно использовать категорию тяжести работы с учетом общего воздействия всех факторов производственной среды [8, с4].

5.1.1 Разделение работ по тяжести и напряженности

Из-за влияния вредных факторов производства во время трудового процесса могут сформироваться одно из трех функциональных состояний организма: нормальное, пограничное (балансирующее между нормой и патологией) и патологическое.

Согласно существующей классификации, условия труда можно разделить на 6 категорий тяжести работы.

Первая категория тяжести включает те виды работ, которые могут быть выполнены в оптимальных условиях внешней производственной среды и при

оптимальном значении физической, умственной и нейро-эмоциональной нагрузки. В этом случае рабочая нагрузка будет соответствовать физиологическим возможностям человеческого организма и его способностям.

Вторая категория тяжести включает те работы, в результате которых уровни вредных и опасных производственных факторов не превышают оптимальных или максимально допустимых значений. В то же время работоспособность не ухудшается, нет никаких отклонений в состоянии здоровья, которые могут быть связаны с профессиональной деятельностью. Возможные функциональные изменения исчезают во время отдыха.

Третью категорию тяжести следует отнести к работе, выполняемой в условиях, когда практически здоровые люди испытывают реакции, характерные для пограничного состояния организма. Наблюдается небольшое снижение показателей труда (производительность труда). Использование рациональных режимов работы и отдыха может быстро устранить эти негативные последствия.

Четвертая категория тяжести включает в себя работу, в результате которой организм может сформировать достаточно глубокое пограничное состояние даже у практически здоровых людей. Большинство физиологических показателей ухудшаются (реакции замедляются), особенно в конце смены или рабочей недели. Могут появиться характерные состояния патологий, обусловленные производством.

Пятая категория тяжести включает работу, которая в конце смены и или рабочей недели формирует реакции, характерные для патологического состояния организма у практически здоровых людей и которые, как правило, исчезают у большинства рабочих после полного отдыха. Но у некоторых людей изменения могут привести к производственным и профессиональным заболеваниям.

Шестая категория тяжести включает работу, в результате которой четко проявляются признаки патологического состояния в организме человека. Эти работы выполняются в особо опасных (критических) условиях работы. В то же время патологические реакции могут развиваться достаточно быстро, могут быть необратимыми и часто сопровождаются серьезным нарушением функций жизненно важных органов и систем [8, с.5].

5.1.2 Количественный анализ тяжести и напряженности труда

Условия работы оказывают прямое влияние на состояние организма, которое характеризуется определенными реакциями. Чтобы оценить негативное воздействие внешних условий на трудящихся, необходимо определить категорию тяжести работы.

При проведении количественного анализа тяжести работы необходимо учитывать санитарно-гигиенические и психофизиологические факторы рабочей среды, характеризующие условия труда.

К санитарно-гигиеническим факторам производственной среды в соответствии с ГОСТ следует отнести:

- микроклимат в рабочей зоне;
- наличие вредных веществ различных классов опасности;
- наличие и концентрацию производственной пыли;
- вибро-акустические факторы и ультразвук;
- интенсивность теплового излучения;
- электромагнитные излучения различных диапазонов частот;
- ионизирующие излучения (рентгеновское, гамма-, α-β-излучения);
- биологические факторы.

К психофизиологическим факторам соответствии с ГОСТ следует отнести:

- физическую, динамическую и статическую нагрузки;
- рабочую позу и перемещения в пространстве;
- сменность, продолжительность непрерывной работы в течение суток;
- разряд зрительных работ;
- число важных объектов наблюдения;
- темп работы, монотонность работы;
- объем получаемой и перерабатываемой информации;
- режим труда и отдыха;
- нервно-эмоциональная нагрузка;
- интеллектуальная нагрузка.

В анализе учитываются факторы рабочей среды, которые характерны для этого рабочего места и конкретной профессии. Как правило, условия труда определяются набором факторов рабочей среды, поэтому каждый показатель или фактор окружающей среды должен оцениваться в баллах от 1 до 6, в зависимости от их численного значения (Приложение А, таблицы 1.1-1.3) [8, с.6].

Категория тяжести и напряженности труда связана с интегральной бальной оценкой, которую можно определить по формуле:

$$U_r = \left[X_{max} + \frac{\sum_{i=1}^{n} X_i}{n-1} \times \frac{6 - X_{max}}{6}\right] \times 10, \qquad (5.1)$$

где X_{max} - самая большая из полученных частных балльных оценок;

X_i - балльная оценка по i-му из учитываемых факторов;

n - общее число факторов без учета одного фактора X_{max};

N - общее количество факторов.

Зависимость категории тяжести от интегральной балльной оценки приведена в таблице 5.1.

Таблица 5.1 - Категории тяжести труда

Категория тяжести труда	1	2	3	4	5	6
Интегральная оценка элементов условий труда, $U_Т$, баллы	до 18	18,1- 33	33,1- 45	45,1- 53	53,1- 59	59,1- 60

Если вредный фактор влияет не на всю рабочую смену, оценка факторов и показателей условий труда должна определяться в зависимости от времени их воздействия на работника:

$$X_{i\text{факт}} = X_i \cdot \frac{t}{t_{\text{см}}} \ , \qquad (5.2)$$

где X_i - оценка *i*-го элемента условий труда в баллах;

t- фактическая длительность действия фактора, мин.;

$t_{см}$- продолжительность смены, мин.

Повышение тяжести труда повлияет на работоспособность человека. Снижение работоспособности напрямую связано с состоянием усталости, которое может быть определено количественно с помощью индекса усталости, выраженного в условных единицах. Связь между интегральным индексом тяжести труда и степенью усталости может быть выражена уравнением:

$$\text{У} = \frac{U_T - 15{,}6}{0{,}64} \ , \qquad (5.3)$$

где *У* - показатель утомления в условных единицах;

15,6 и 0,64 - коэффициенты регрессии;

U_T - интегральный показатель категории тяжести труда в баллах.

По приведенной ниже формуле можно определить уровень работоспособности, при условии, что мы имеем значение уровня усталости

$$\text{R} = 100 - \text{У} \ , \qquad (5.4)$$

где R - уровень работоспособности в относительных единицах.

По значениям работоспособности, определенным до и после проведения мероприятий по улучшению условий труда можно рассчитать изменение производительности труда (прирост производительности) по формуле:

$$\text{П}_{nm} = \left[\frac{R_2}{R_1} - 1\right] \times 100 \times 0{,}2 \ , \qquad (5.5)$$

где $П_{пт}$ - прирост производительности труда;

R_2 и R_1 - работоспособность в условных единицах до и после проведения мероприятий по улучшению и оздоровлению условий труда;

0,2 - поправочный коэффициент, который отражает зависимость между увеличением работоспособности и ростом производительности труда.

Тяжесть и интенсивность труда напрямую влияют на рост производственных травм. Поскольку интегральная оценка позволяет определить категорию тяжести труда, величина профессионального травматизма может быть рассчитана по формуле:

$$K = \frac{1}{1{,}3 - 0{,}0185 \cdot U_T}\ , \qquad (5.6)$$

где K - рост производственного травматизма, количество раз;

U_T - интегральный показатель категории тяжести труда в баллах.

На рабочем месте должно быть предусмотрено создание благоприятной рабочей среды и формирование условий труда, которые будут относиться к первой категории тяжести работы (оптимальной). Если оборудование имеет низкий риск получения травмы и более высокую производительность, то количество травм можно принять равным единице, и в этом случае интегральный показатель тяжести труда будет равен:

$$U_T = (1{,}3 - 1{,}0) / 0{,}0185 = 16{,}2 \qquad (5.7)$$

что будет характеризовать наиболее приемлимую травмобезопасность данного рабочего места.

5.2 Определение категории тяжести и напряженности труда оператора персональных электронных вычислительных машин (ПЭВМ)

Таблица 5.2 - Исходные данные для выполнения расчета

Профессия	Фактор рабочей среды и условия труда	Значение показателя до модернизации	Значение показателя после модернизации	Продол. времени действия
Оператор Security Operation Center	Температура воздуха на РМ в теплый период года, C^0	30	22	480
	Превышение допустимого уровня звука, дБа	90	70	480/420
	РМ стационарное, поза свободная	-	-	480
	Масса перемещаемых грузов	до 5 кг	до 2 кг	480
	Работа в утреннюю смену	-	-	-
	Продолжительность непрерывной работы в течение суток, часов	7	5	480
	Длительность сосредоточенного наблюдения, % от продолжительности рабочей смены	80	60	480/240
	Обоснованный режим труда и отдыха с применением функциональной музыки и	-	-	-
	Нервно-эмоциональная нагрузка возникает в результате простых действий по индивидуальному плану	-	-	-

Решение:

В результате мероприятий по безопасности и охране труда была произведена модернизация факторов рабочей среды и условия труда: замена устаревшего оборудования, разделено рабочее место оператора, установлен кондиционер, сокращена продолжительность непрерывной работы в течение суток и длительность сосредоточенного наблюдения. Изменились показатели факторов рабочей среды и условий труда.

По исходным данным и таблицам приложения А проводим выставляем баллы каждому фактору рабочей среды и показателю до и после проведения мероприятий по оздоровлению условий труда. При оценке необходимо корректировать значение балла в зависимости от времени воздействия. Результаты оценки представляем в виде таблицы (таблица 5.3) [8, с.10].

Таблица 5.3 - Балльная оценка факторов рабочей среды и условий труда

Фактор рабочей среды и условия труда	Значение показателя	Оценка факторов в баллах	
		До проведения мероприятий	После проведения мероприятий
Температура воздуха на РМ в холодный период года, C^0	15/22	5	1
Превышение допустимого уровня звука, дБа	90/70	4	2
РМ стационарное, поза свободная, масса перемещаемых грузов	5/2	2	1
Работа в утреннюю смену. Продолжительность непрерывной работы в течение суток, часов	7/5	2	1
Длительность сосредоточенного наблюдения, % от продолжительности рабочей смены	80/60	2	2
Обоснованный режим труда и отдыха с применением функциональной музыки и гимнастики		2	1
Нервно-эмоциональная нагрузка возникает в результате простых действий по индивидуальному плану		1	1

*В графе "Значение показателя" в числителе указаны значения до проведения мероприятий, а в знаменателе - после.

После оценки в баллах факторов и показателей необходимо рассчитать интегральную оценку тяжести труда до и после проведения мероприятий по формуле (5.1):

а) до проведения мероприятий по улучшению условий труда:

$$U_1 = \left[5 + \frac{5+4+2+2+2+2+1}{6} \times \frac{6-5}{6}\right] \times 10 = 55$$

из таблицы 5.1 определяем, что данные условия труда относятся к пятой категории тяжести труда, значит у работника формируется достаточно устойчивое патологическое состояние, которое характеризуется замедлением реакций;

б) после проведения мероприятий по улучшению условий труда.

Так как после проведения мероприятий изменилось время воздействия факторов рабочей среды и условий труда, необходимо пересчитать оценку факторов.

Принимаем продолжительность смены равной 480 мин.

В нашем случае после проведения мероприятий изменилось время воздействия шумов (фиксировалось превышение ПДУ шума), поэтому бальную оценку необходимо провести с учетом данного изменения:

$$X_{\text{кор 1}} = 2 \cdot \frac{420}{480} = 1{,}75\ ,$$

и при изменении продолжительности нервно-эмоциональных нагрузок:

$$X_{\text{кор 2}} = 1 \cdot \frac{240}{480} = 0{,}5.$$

Интегральная балльная оценка после проведения мероприятий с учетом коррекции будет равна:

$$U_2 = \left[2 + \frac{1 + 1{,}75 + 1 + 1 + 2 + 1 + 0{,}5}{6} \times \frac{6 - 2}{6}\right] \times 10 = 29{,}2$$

из таблицы 1 определяем, что данные условия труда относятся к второй категории тяжести труда. В таких условиях работоспособность не ухудшается, все симптомы патологий могут отступить после отдыха.

Прогноз изменения травматизма после проведения мероприятий по улучшению условий труда выполняем следующим образом. Рост травматизма для пятой и третьей категории тяжести оцениваем по формуле (5.6).

Определим рост травматизма до проведения мероприятий по улучшению условий труда:

$$\text{У}_1 = \frac{1}{1{,}3 - 0{,}0185 \times 55} = 3{,}53\ ,$$

После проведения мероприятий (изменение температуры воздуха рабочей среды, уменьшение уровня шума и времени воздействия на оператора и т.д.) категория тяжести труда снизится до второй (U2=29,2), что будет соответствовать росту травматизма в 1,3 раза по сравнению с рациональными условиями труда:

$$\text{У}_2 = \frac{1}{1{,}3 - 0{,}0185 \times 29{,}2} = 1{,}32,$$

При проведении мероприятий по улучшению условий труда категория тяжести изменилась с пятой до второй. Как отмечалось выше тяжесть труда негативно влияет на степень утомления, а значит и работоспособность человека.

Для исследования динамики изменения работоспособности и производительности необходимо рассчитать значения показателей утомления и работоспособности [8, с.12]:

а) до проведения комплекса мероприятий:

- показатель утомления по формуле (5.3):

$$У_1 = \frac{55 - 15,6}{0,64} = 61,6 ,$$

- уровень работоспособности по формуле (5.4):

$$R_1 = 100\text{-}61,6 = 38,4 ,$$

б) после проведения комплекса мероприятий:

- показатель утомления:

$$У_2 = \frac{29,2 - 15,6}{0,64} = 21,2 ,$$

- уровень работоспособности:

$$R_2 = 100 – 21,2 = 78,8.$$

5. Изменение производительности труда (прирост производительности труда) за счет изменения работоспособности по формуле (5) составит:

$$П_{nm} = \left[\frac{R_2}{R_1} - 1\right] \times 100 \times 0,2 = \left[\frac{78,8}{38,4} - 1\right] \times 100 \times 0,2 = 21,04 .$$

Вывод

В этой главе был проведен анализ оптимальных условий труда для разработки программного обеспечения и рассчитаны необходимые меры по охране труда.

При анализе тяжести труда на рабочем месте оператора выполнен расчет интегральной балльной оценки. В результате расчета условия труда оператора относятся к пятой категории тяжести, что негативно будет влиять на его работоспособность и здоровье. Поэтому необходимо было разработать мероприятия по улучшению условий труда, такие как: снижение длительности воздействия шумов и нервно-эмоциональных нагрузок. После внедрения мероприятий категория тяжести труда повысилась до второй степени. Уровень работоспособности увеличился с 38,4 относительных единиц до 78,8, прирост производительности труда составил 21,04 %. Следует проводить дополнительные мероприятия по улучшению условий труда [8, с.13].

Заключение

В данном дипломном проекте были рассмотрены и был произведен сравнительный анализ симметричных систем шифрования, с целью выбора наиболее подходящих алгоритмов для шифрования информации и кодирования ключа. Изучены основы криптографической защиты и механизмы работы алгоритмов LEA-128 SEA и Base64. Скомпилировано и протестировано программное обеспечение для шифрования, передаваемой по открытым каналам связи, информации «AIPET Code».

Были проделаны расчеты касательно технико-экономического обоснования и безопасности жизнедеятельности. В результате расчетов, уровень работоспособности увеличился с 38,4 относительных единиц до 78,8, прирост производительности труда составил 21,04 %. Согласно экономическим расчетам стоимость разработки программного продукта составила 550 000 тг. Поставленные в начале работы цели и задачи были достигнуты. Сделаны выводы о проделанной работе.

Список литературы

1 Якубов Б.М. Защита информации в телекоммуникационных системах. Учебное пособие для ВУЗов. - Нур-Принт, Алматы, 2017.

2 Высшая школа Казахстана, ISSN 1560-1749. Статья: Передача данных с применением алгоритмов шифрования. – Алматы: Дом издательств, 2017. – 316-321 с.

3 Алгоритмы шифрования на Delphi. URL: http://www.delphisources.ru/pages/sources/raznoe/2009-year/lea-128-sea-cipher.html (дата обращения 18.01.2018)

4 Язык программирования Delphi. URL: https://ru.bmstu.wiki/Delphi_(язык_программирования) (дата обращения 21.01.2018)

5 Официальный сайт *RAD* Studio™ 10.1. URL: https://www.embarcadero.com/ru/products/rad-studio (дата обращения 25.01.2018)

6 NetCracker Professional 4.1. URL: http://soft-landia.ru/netcracker.html (дата обращения 30.01.2018)

7 Аманжолова К. Б., Алибаева С. А. Экономика предприятий телекоммуникации: Учебное пособие. – Алматы: АИЭС, 2003.

8 Мазалов И.Ф., Мустафин К.Г., Тыщенко Е.М., Сералиева М.А. Методические указания по выполнению РГР для студентов специальности 5В073100-БЖ. – Алматы: АУЭС, 2015. – 38 с.