МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет информатики
Кафедра геоинформатики и информационной безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«РАЗРАБОТКА СИСТЕМЫ ВЫБОРОЧНОГО ШИФРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ДАННЫХ»

по направлению подготовки 10.05.03 Информационная безопасность
автоматизированных систем
(уровень академического специалитета)
направленность (профиль) «Обеспечение информационной безопасности
распределенных информационных систем»

Студент_____А.Э. Коган
(подпись, дата)

Руководитель ВКР,
профессор, д.ф.-м.н._____В.В. Мясников
(подпись, дата)

Нормоконтролер_____Д.Б. Жмуров
(подпись, дата)

Самара 2019

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Кафедра _геоинформатики и информационной безопасности_____

УТВЕРЖДАЮ
Заведующий кафедрой
_____ /_____/
(подпись) *И.О.Фамилия*
«____»_____20___ г.

Задание на выпускную квалификационную работу (ВКР)

Обучающемуся__Когану Артуру Эдуардовичу_____
(ФИО, полностью)
группы__6511 – 100503 D_____
1.Тема ВКР: _Разработка системы выборочного шифрования пользовательских__ данных_____
_____
утверждена приказом по университету от «____»_____20___ г. №____
2.Перечень вопросов, подлежащих разработке в ВКР:_____
__1) Анализ существующих сторонних решений по шифрованию_____ передаваемых данных_____
__2) Разработка архитектуры и пользовательского интерфейса приложения__
__3) Реализация приложения выборочного шифрования пользовательских___ данных_____
__4) Проверка работоспособности и корректности работы разработанного___ приложения_____

3.Консультанты по разделам ВКР*(при наличии):*
раздел ВКР:_____
разрабатываемые вопросы: _____
_____
_____ /_____/
должность, степень *подпись* *И.О.Фамилия*

4.Дата выдачи задания: «____»_____20___ г.
5.Срок представления на кафедру законченной ВКР: «____»_____20__г.

Руководитель ВКР
_Профессор, д.ф.-м.н._____ _____ /_Мясников В.В.____/
должность, степень *подпись* *И.О.Фамилия*

Задание принял к исполнению _____ /_Коган А.Э._____/
подпись обучающегося *И.О.Фамилия обучающегося*

РЕФЕРАТ

Выпускная квалификационная работа специалиста: 43 с., 19 рисунков, 13 источников.

ВЫБОРОЧНОЕ ШИФРОВАНИЕ, AES-128, ARGON2, WINAPI, BASE64, ШИФРОВАНИЕ ДАННЫХ, WINDOWS, БУФЕР ОБМЕНА.

Цель работы – разработка удобного инструмента выборочного шифрования, использование которого возможно с любыми программами, установленными у пользователя.

В процессе работы были изучены алгоритмы шифрования и кодирования данных, средства взаимодействия с операционной системой, а так же получены навыки разработки системных приложений.

В результате работы была получена программа, позволяющая шифровать любые отправляемые пользователем сообщения. С возможностью работы сразу с несколькими собеседниками.

Эффективность работы заключается в возможности использование с любыми позволяющими обмениваться сообщениями программами пользователя.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире передавать сообщения можно различными путями. Как с помощью мобильного телефона по СМС, так и по сети Internet. Если перехватывать данные мобильной сети без специальных устройств является трудной задачей, то для перехвата по сети Internet требуется компьютер или ноутбук с подключением к сети.

Для связи можно использовать как веб-приложения, запускаемые через браузер, так и специальные программы клиенты, например почтовый клиент или обычный мессенджер. По мере развития информационных технологий совершенствуются и методы перехвата информации. Многие современные мессенджеры придерживаются стандартов защиты, но остаются пользователи, которые используют старые приложения, которые были разработаны во времена, когда не требовалось шифровать передаваемые данные и в них не реализована защита информации и работы по улучшению больше не ведутся. В таких случаях требуется дополнительная программа, которая обеспечит шифрование данных перед передачей по каналу связи.

Существует множество средств шифрования данных, но для их работы необходимо участие пользователя, а также установка нескольких программ и точная настройки используемых средств. Самым оптимальным решением для узкоспециализированных задач является написание своего инструмента, который будет удовлетворять требуемым условиям и режимам работы.

Так же о подобной проблеме задумаются и разработчики современных мессенджеров. В таких приложениях реализовано «прозрачное» шифрование. Пользователь не принимает никакого участия в процессе шифрования информации.

Разрабатываемое приложение должно быть универсальным, способным работать со всеми средствами обмена сообщениями. Это может быть использовано ради дополнительного шифрования

В соответствии с вышеизложенными причинами были сформулированы и решены следующие задачи:

– Анализ существующих сторонних решений по шифрованию передаваемых данных.

– Разработка архитектуры и пользовательского интерфейса приложения.

– Реализация приложения выборочного шифрования пользовательских данных.

– Проверка работоспособности и корректности работы разработанного приложения.

В 1 части данной работы будет произведён обзор существующих программ и решений в области шифрования передаваемых данных. Как доступных для скачивания и установки, так и тех, которым для корректной работы необходима предварительная настройка.

Во 2 части работы описывается процесс выбора архитектуры и реализации готового программного обеспечения. Приводится описание каждого из модулей будущего приложения, а также алгоритмы работы приложения для каждой из задач.

В 3 части описываются алгоритмы и функции, которые были использованы при реализации будущего приложения.

4 часть показывает работоспособность и корректность дешифрования сообщений с помощью реализованного приложения.

1 Программные средства шифрования передаваемых данных

В современном мире для передачи данных в зашифрованном виде могут использоваться как готовые программные решения, так и разработанные пользователем для себя с помощью одного из языков программирования. Достоинства уже реализованных продуктов в том, что над их качеством трудилась команда разработчиков, но так же там могут быть реализованы те функции, которые не нужны конкретному пользователю. В таком случае можно написать своё приложение, которое будет выполнять узкий круг требований или же реализовывать специфичный алгоритм обмена данными, но за качество и стойкость алгоритмов шифрования уже отвечает пользователь. Для обмена сообщениями по сети интернет могут быть использованы разные средства:

– E-Mail.

– Социальные сети.

– Мессенджеры.

– Клиенты и веб-версии различных чатов.

Для шифрования передаваемых данных можно использовать следующие программы и инструменты:

– Messenger 400.

– ProtonMail.

– Delta Chat.

– PGP.

– VPN.

– ZeroTier One.

Рассмотрим каждый из перечисленных вариантов подробнее.

1.1 Доступные программные решения

1.1.1 Messenger 400

Защищенная электронная почта X.400 [1] на базе электронного почтамта Messenger 400 [1] фирмы Infonet Software Solutions предназначена для предоставления абонентам почтовых услуг по обмену защищенными сообщениями с использованием механизма двусторонней аутентификации абонента на почтамте и почтамта на абонентском пункте, реализованного с помощью электронной цифровой подписи.

Из возможностей стоит отметить, что защищенная электронная почта X.400 способна устойчиво функционировать в различных сетях, в том числе на недорогих низкоскоростных линиях, и использовать различные протоколы связи, включая X.25, X.28, TCP/IP, IPX/SPX и другие. Система хранения и передачи сообщений поддерживает развитые средства маршрутизации, обеспечивающие возможность оптимальной производительности и настройки с целью уменьшения стоимости коммуникационных услуг. Использование сервисов, предусмотренных стандартом X.400, а именно квитанций о доставке и прочтении, гарантированной доставки и маршрутизации, является большим преимуществом данной почтовой системы по сравнению с другими системами.

В состав защищенной электронной почты X.400 входят программные и аппаратно-программные средства:

1) Защищенный почтовый сервер.

2) Защищенные средства криптографической защиты информации абонентские пункты.

3) Центр управления ключевой системой.

Абонентские пункты предназначены для обмена зашифрованными и подписанными сообщениями и поддержки механизма двухсторонней аутентификации при обмене через электронный почтамт. Абонентские пункты позволяют в соответствии с рекомендациями X.400 обмениваться зашифрованными и подписанными почтовыми сообщениями между пользователями, зарегистрированными в системе защищенной электронной почты X.400 с использованием механизма двухсторонней аутентификации. Абонентские пункты позволяют оператору создавать, рассылать и обрабатывать электронные сообщения в ручном и автоматическом режимах, редактировать конверты, работать с папками, работать по заданному сценарию без участия пользователя, сканировать сетевой разделяемый диск и обрабатывать сценарии. Абонентские пункты также позволяют выполнять ряд специфических задач, связанных с обеспечением безопасности и конфиденциальности данных. На абонентских пунктах может быть зарегистрировано неограниченное число пользователей. Разграничение доступа к ресурсам абонентских пунктов обеспечивается применением паролей. Часть пользователей может быть зарегистрирована в качестве администраторов и иметь доступ ко всем ресурсам абонентских пунктов. Для абонентских пунктов разработан специализированный помехоустойчивый протокол LAPS, позволяющий работать на каналах связи плохого качества и обеспечивающий механизм двухсторонней аутентификации, а также протокол ELINK, обеспечивающий работу абонентских пунктов в локальных сетях.

Но такая система отлично подходит для предприятий. Её невозможно использовать в повседневной жизни на компьютере или ноутбуке, так как для этого требуются дополнительные сервера и специализированное программное обеспечение, которое требует тонкой настройки.

1.1.2 ProtonMail

Особенностями безопасности ProtonMail [2] являются:

1) Сквозное шифрование.

Смысл сквозного шифрования состоит в том, что прочитать сообщение не сможет никто, кроме нужного получателя Сообщения хранятся на серверах ProtonMail в зашифрованном виде. Также в зашифрованном виде они передаются между серверами и пользовательскими устройствами. Сообщения между пользователями ProtonMail также передаются в зашифрованном виде внутри защищённой сети серверов компании. Proton Technologies. И поскольку данные в любом случае зашифрованы, риск перехвата сообщений в значительной мере устранён.

2) Нулевой доступ к данным пользователя.

Архитектура нулевого доступа ProtonMail означает, что данные зашифрованы таким способом, который делает их недоступными для владельцев сервера, на котором хранится информация. Данные шифруются на стороне клиента с применением ключа, доступа к которому ни у кого нет. Это значит, что владелец сервера не имеем технической возможности расшифровать сообщения, и как результат, не можем передать ваши данные третьим лицам. С ProtonMail конфиденциальность гарантирована математически.

3) Шифрование, основанное на открытом исходном коде.

ProtonMail применяет только безопасные реализации AES, RSA, а также OpenPGP. Кроме того, все применяемые шифровальные библиотеки являются открытым исходным кодом. Применяя библиотеки с открытым исходным кодом, гарантируется, что в используемых алгоритмах шифрования нет тайно встроенных «закладок». ПО с открытым исходным кодом ProtonMail было тщательно проверено экспертами в области безопасности со всего мира, чтобы гарантировать наивысшие степени защиты.

ProtonMail может использоваться на любых устройствах, но для работы с ним необходимо зарегистрироваться и получить адрес электронный почты. Соответственно, ProtonMail используется только для общения по средством электронной почты.

1.1.3 Delta chat

Delta Chat – децентрализованный мессенджер для Android, iOS, Linux, Mac, функционирующий поверх стека протоколов E-mail.

Ключевые особенности DeltaChat:

– Приложение подключается к любому серверу электронной почты, выбранному вами.

– Распределённая связь осуществляется через федерации почтовых серверов.

– Адресатом может быть любой владелец электронной почты, даже если он не установил себе Delta Chat.

– При наличии возможности для сквозного шифрования переписки применяется механизм Autocrypt.

– Отображение отметок о прочтении и статусе доставки и быстрые уведомления функционируют по протоколу Push-IMAP.

– Полностью открытый исходный код приложения под лицензией GPLv3 и протоколы, основанные на стандартах.

Для Delta Chat не нужно заводить новый адрес электронной почты, но он работает поверх E-mail. Для его работы необходим хотя бы один адрес электронной почты.

1.2 Другие средства защиты передаваемых данных

1.2.1 PGP

Шифрование PGP [4] осуществляется последовательно хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом. Симметричное шифрование производится с использованием сеансового ключа. Сеансовый ключ генерируется с использованием криптографически стойкого генератора псевдослучайных чисел. Сеансовый ключ зашифровывается открытым ключом получателя. Каждый открытый ключ соответствует имени пользователя или адресу электронной почты. Первая версия системы называлась Сеть Доверия и противопоставлялась системе X.509, использовавшей иерархический подход, основанной на удостоверяющих центрах, добавленный в PGP позже. Современные версии PGP включают оба способа.

Для корректной работы создаётся ключевая пара: открытый и закрытый ключ. При генерации ключей задаются их владелец, тип ключа, длина ключа и срок его действия. Открытый ключ используется для шифрования и проверки цифровой подписи. Закрытый ключ — для декодирования и создания цифровой подписи. PGP поддерживает аутентификацию и проверку целостности посредством цифровой подписи. В целях уменьшения объёма сообщений и файлов перед шифрованием производится сжатие данных.

Но PGP неудобно дешифровать и остается проблема хранения долговременных ключей.

1.2.2 VPN

Самым простым способом является использование VPN-туннеля между двумя пользователями. Никто, кроме участников сети, не сможет прочитать информацию, передаваемую по VPN. Одним из примеров является OpenVPN [5], он имеет возможность создания виртуального сетевого TAP-

интерфейса способного инкапсулировать Ethernet-фреймы, а не только более высокоуровневые IP-пакеты.

Но использование VPN служб не гарантирует безопасность, так как шифруются только данные передаваемые по VPN-туннелю между клиентами. В то время как многие приложения работают по клиент-серверной архитектуре и для корректной работы отправляют данные на сторонний сервер.

1.2.3 ZeroTier One

ZeroTier One [6] – это программный продукт с открытым исходным кодом, который устанавливает одноранговые VPN-соединения (P2PVPN) между ноутбуками, настольными компьютерами, телефонами, встроенными устройствами, облачными ресурсами и приложениями. ZeroTier может быть определен как Social VPN или F2F (Friend to Friend), но может быть использован для виртуальной частной сети компаний. ZeroTier One может установить прямое соединение между участниками, даже если они находятся за NAT, в то время как традиционные VPN-соединения устанавливаются от клиентов к серверу, где сервер имеет публичный IP-адрес. Любые компьютеры и устройства, подключенные к локальной сети, обычно подключаются к интернету через NAT и маршрутизатор. В домашней сети NAT и маршрутизатор, как правило, одно и то же устройство. ZeroTier One использует STUN и «hole punching», чтобы установить прямое VPN-соединение между участниками за NAT. Компании также имеют централизованный веб-портал с графическим интерфейсом для управления всеми ZeroTier One, сетевыми интерфейсами и корневыми узлами, установленными в интернете, используемыми для установления соединения аналогично ICE в WebRTC.

Но также, как и VPN, ZeroTier One использует VPN-туннель и для защищенного общения необходимы программы работающие по локальной сети.

1.2.4 Разработка средства шифрования с помощью языка программирования

Так же возможна реализация шифрования трафика канального уровня модели OSI с помощью языка программирования. Преобладающая часть программно-аппаратных средств по шифрованию трафика работает на третьем и более высоких сетевых уровнях. Логические структуры и организация сетей третьего уровня, зачастую, отдалённо коррелирует со структурой низлежащей сети второго. Из-за этого, создание виртуальных частных криптографически защищённых сетей является не тривиальной задачей, затрагивающей всю систему маршрутизации.

Прозрачное шифрование трафика именно на втором уровне избавит от изменений в маршрутизации, уменьшит накладные расходы от туннелированния сетевых пакетов. Проще организовать обмен криптографическими ключами между подразделениями одной организации и сократить сложные протоколы и процедуры установления общего ключа. Задержки связи из-за использования симметричной криптографии со статическими ключами отсутствуют. Кроме того, объёмы трафика останутся неизменны.

Для реализации собственных инструментов необходимы глубокие познания в области разработки программного обеспечения. Также, для реализации криптографически защищенных приложений нужны познания и в области криптографии.

1.3 Выводы и результаты

Средства шифрования требуют установки нескольких программ и точной настройки используемых средств. Также, часто программные решения

нацелены на использование только одного канала связи, например, электронной почты, что делает невозможным использование их в связке с мессенджерами. Самым оптимальным решением для узкоспециализированных задач является написание своего инструмента, который будет удовлетворять требуемым условиям и режимам работы.

2 Проектирование архитектуры и пользовательского интерфейса приложения

Для реализации шифрования передаваемых данных нам необходимо средство перехвата этих самых данных на уровне операционной системы. А также возможность получать зашифрованные данные от других участников общения и производить их дешифрование.

2.1 Требования к разрабатываемой программе

Программа должна реализовывать все требования к системе выборочного шифрования и не требовать вмешательства и долгой и трудной настройки от конечного пользователя. Должна быть возможность шифровать любые сообщения, отправляемые через программы, установленные на рабочей машине конечного пользователя, и корректно дешифровать все полученные сообщения собеседника.

2.2 Структура программы

Программа должна состоять из 3 основных модулей:

1) Шифратор/дешифратор и модуль по работе с ключами шифрования.

2) Модуль для преобразования данных.

3) Модуль по работе с операционной системой.

Шифратор/дешифратор отвечает за корректное шифрование отправляемых и дешифрование полученных пользовательских данных.

Модуль по работе с ключами шифрования отвечает за генерацию корректных ключей шифрования и поддержания актуального их состояния между двумя конкретными пользователями.

Модуль для преобразования данных необходимо для обработки бинарных данных для корректной работы с операционной системой и отображению их пользователю.

Модуль по работе с операционной системой необходимо, чтобы получать данные, введенные пользователем для дальнейшей их пересылки, а также отображения полученных от других пользователей сообщений.

2.2.1 Основные модули

Программа состоит из следующих модулей:

1) Шифратор – предназначен для шифрования данных заданным алгоритмом шифрования с использованием сгенерированного ключа шифрования.

2) Дешифратор – предназначен для дешифрования данных исходя из заданного алгоритма шифрования с использованием сгенерированного ключа дешифрования.

3) Модуль захвата отправляемых данных – необходимо для захвата введённых пользовательских данных, обработки их шифратором и возвращения программе для дальнейшей его работы по их пересылке.

4) Модуль отображения полученных данных – необходимо для перехвата полученных данных, их дешифрования и дальнейшего их отображения пользователю.

5) Модуль генерации ключей шифрования/дешифрования – необходимо для генерации корректных ключей, с помощью которых возможно произвести процедуру шифрования или дешифрования.

6) Модуль синхронизации ключей шифрования/дешифрования – необходимо для правильной работы программы и возвращения корректных результатов процедур шифрования/дешифрования даже после появления и устранения неисправностей в работе приложения или в режиме работы, когда свои данные шифрует только один из собеседников.

2.3 Выбор средств разработки и системных программных средств

Так как Windows является одной из популярных используемых операционных систем, то программа должна поддерживать данную операционную систему. Для реализации перехвата отправляемых и полученных данных будем использовать библиотеку WinAPI [7] и соответственно язык программирования C для работы с ней. Разработка программы будет происходить в среде разработки Visual Studio 2019 community edition, как в удобной и современной IDE для работы с языком программирования C.

2.4 Описание программы

Работу программы кратко можно описать следующим образом:

1) Обработка нажатия горячей клавиши.

2) Предобработка данных.

3) Шифрование/дешифрование.

4) Постобработка данных.

Также программа предусматривает сохранение «контактной книги» пользователя, чтобы не было необходимости вводить ключи собеседников при каждом запуске программы. Добавление и смена собеседника возможна как с помощью горячих клавиш, так и через меню главного окна программы.

На главном окне располагается информация о текущем собеседнике и место под текст, получаемый дешифровкой полученных от собеседника сообщений.

Рисунок 2.1 – Главное окно программы

Во время реализации программы были выделены следующие модули:

– Модуль взаимодействия с операционной системой.

– Модуль преобразования данных.

– Криптографический модуль.

– Модуль работы с собеседниками.

– Модуль графической подсистемы.

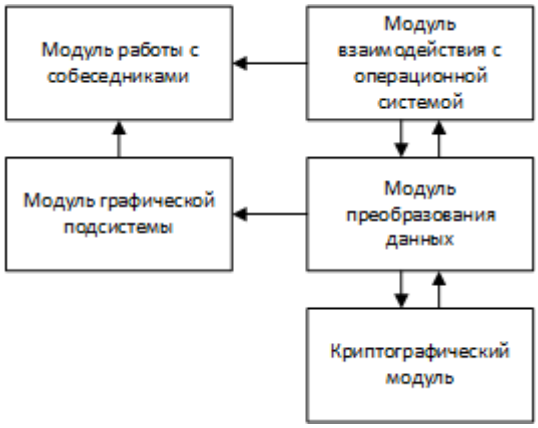

Рисунок 2.2 – Схема взаимодействия модулей программы

2.4.1 Описание модулей

Модуль взаимодействия с операционной системой предназначен для передачи и получения сообщений от пользователя посредством буфера обмена. Также, в этом модуле заложена обработка горячих клавиш. Всё взаимодействие реализовано с использованием функций WinAPI.

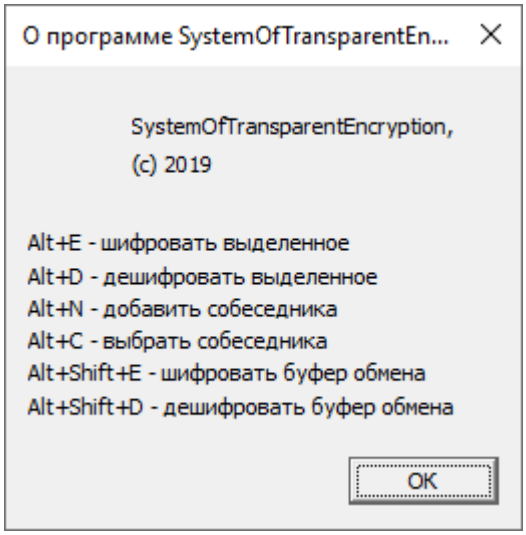

Рисунок 2.3 – Окно «О программе» со списком всех горячих клавиш

Модуль преобразования данных отвечает за Base64 [8] кодирование и декодирование сообщений. В этом модуле дополнительно реализованы функции работы со строками, такие как конкатенация, дополнение до требуемой длины и приведение к требуемому типу для отображения полученных сообщений на главном окне программы.

Криптографический модуль использует как для шифрования и дешифрования сообщений, так и для дополнительных операций, в частности генерация дополнительных данных требуемой длины и генерацию сессионных ключей на основе общего секрета.

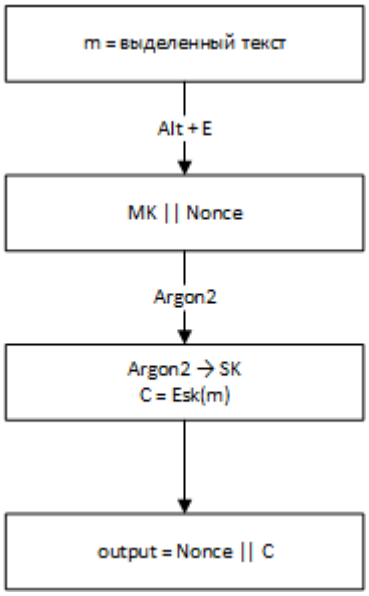

Рисунок 2.4 – Упрощенная схема процесса шифрования

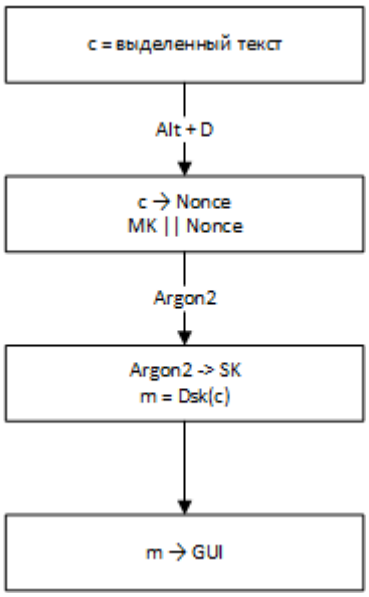

Рисунок 2.5 – Упрощенная схема процесса дешифрования

Модуль работы с собеседниками нужен для возможности пользователя общаться одновременно с несколькими собеседниками. Он содержит в себе также функции сохранения и загрузки файла ключей. Файл хранится в зашифрованном виде. Сохранение всех доступных собеседников в файл происходит в момент закрытия программы. Загрузка собеседников из файла происходит в момент запуска программы.

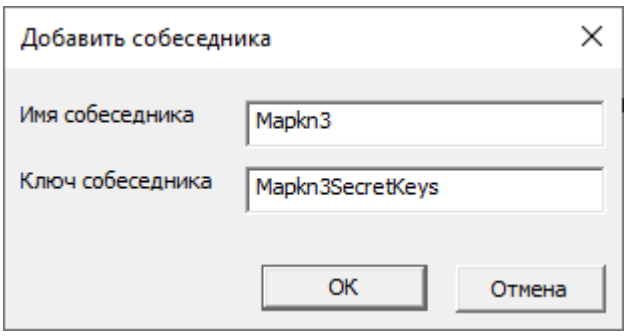

Рисунок 2.6 – Окно добавления нового собеседника

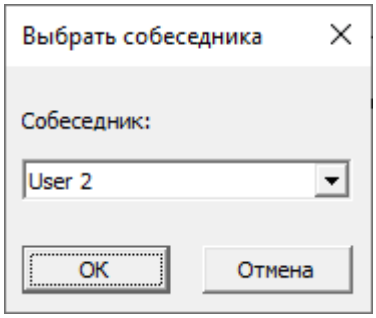

Рисунок 2.7 – Окно выбора текущего собеседника

Модуль графической подсистемы был добавлен для более удобного взаимодействия пользователя с программой. Он отвечает главное окно программы, где отображается вся текущая информация. Взаимодействие можно производить как через предоставленное меню, а и с помощью клавиш быстрого доступа вида Alt + <символ>.

Рисунок 2.8 – Пункт меню «Файл»

2.4.2 Описание алгоритмов

Для упрощения работы пользователя с реализуемым приложением, мы используем ключи, называемые общим секретом, являющиеся долговременными ключами шифрования.

На основе общих секретов собеседников генерируются сеансовые ключи шифрования [9]. Для гарантии стойкости и возможности дешифровывать любые сообщения в любой момент времени также используется случайное «одноразовое» число, называемое Nonce, полученное с помощью генератора псевдослучайных чисел.

Описанное ранее число Nonce будет использовано в связке с общим секретом собеседников. Для корректности обработки Nonce преобразуется в строку, состоящую из символов алфавита Base64.

Рисунок 2.9 – Описание генерации шифротекста

Для генерации отправляемого шифротекста нам потребуется исходное сообщение и ключ, называемый общим секретом, собеседника. Сначала мы генерируем случайное «одноразовое» число, которое будет использовано для генерации сеансового ключа шифрования с помощью алгоритма Argon2 [10]. Новым сгенерированным сеансовым ключом с помощью алгоритма AES [11] шифруем исходное сообщение. Для гарантии того, что сообщение отобразится в любой программе и не будет явных потерь при передаче, кодируем полученный шифротекст с помощью Base64. Чтобы собеседник смог расшифровать сообщение, мы передаем Nonce вместе с кодированным шифротекстом. Так как мы не знаем, какой длины будет сообщение, мы гарантируем, что Nonce будет фиксированной, заранее известной длины, и дописываем его в начало передаваемого сообщения. Полученное сообщение возвращаем в программу, которую в данный момент использует пользователь.

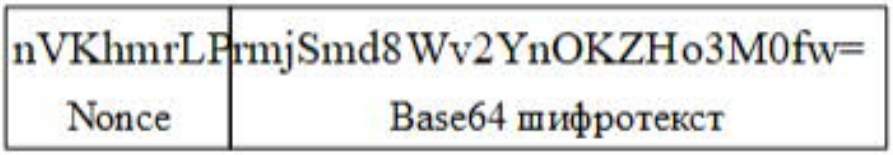

Рисунок 2.10 – Пример сгенерированного передаваемого сообщения

Рисунок 2.11 – Описание дешифрования полученного шифротекста

При получении от собеседника зашифрованного сообщения мы первым делом извлекаем из него случайное «одноразовое» число. После, алгоритмом Argon2 с помощью ключа собеседника, от которого получили сообщение, и извлечённого случайного «одноразового» числа мы генерируем сеансовый ключ шифрования. Далее, оставшуюся часть сообщения мы декодируем из Base64 в бинарные данные. Полученные бинарные данные с использованием

алгоритма шифрования AES с помощью полученного сеансового ключа мы дешифруем сообщение собеседника. Так у нас нет возможности вмешиваться в графический интерфейс других программ, установленных на устройстве пользователя, то мы отображаем полученное сообщение в специальном окне реализуемого приложения.

2.5 Выводы и результаты

Программа реализована в стиле системы выборочного шифрования и обработки пользовательских данных на уровнях получения сообщения из поля ввода и отображения полученного сообщения в специальном окне или передачи текста на отправку по каналу связи и приёма из канала связи ответа собеседника, в зависимости от реализации сетевого взаимодействия конкретной программы.

Для наименьшего вмешательства пользователя в работу программы было выбрано взаимодействие через горячие клавиши. Из-за отсутствия влияния на окна других программ, установленных на устройстве пользователя, в приложении реализован графический интерфейс, упрощающий не только выбор собеседника, но и работу с полученными сообщениями.

Продумано использование данного приложения с различными программами пользователя, что доказывает универсальность. Модульная архитектура приложения позволяет перенести программу на другую операционную систему с минимальными изменениями в программном коде. Переносимость достигается за счёт того, что основные модули реализованы платформонезависимыми и всё взаимодействие с операционной системой заложено в специально предназначенном для этого модуле.

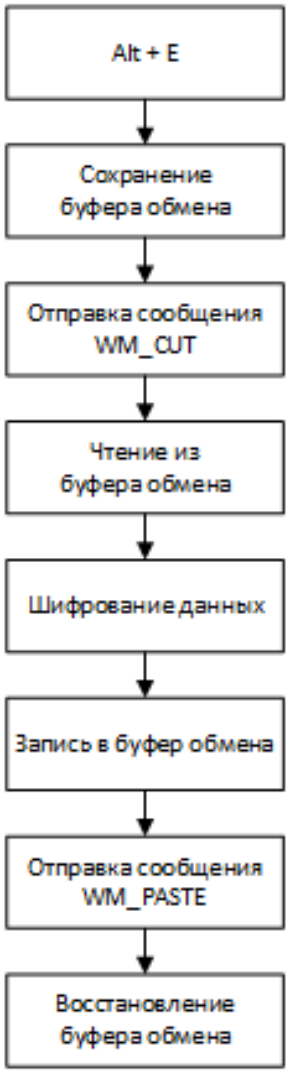

Рисунок 2.12 – Алгоритм работы программы при шифровании

Как видно по схеме, сначала происходит обработка нажатия горячей клавиши. После мы сохраняем текущее состояние буфера обмена, чтобы не вмешиваться в работу других процессов и работа программы была как можно менее заметной. Далее нам нужно получить сообщение для шифрования. Мы будем это делать с помощью буфера обмена и поэтому имитируем нажатие горячей клавиши «вырезать». Вырезаем мы для того, чтобы записать зашифрованные данные на место выделенного сообщения. После этого, буфер обмена содержит необходимое нам сообщение. Получаем сообщение из буфера обмена и готовимся его шифровать. Для корректной работы AES-128 размер текста должен быть кратен 128 битам или 16 символам. Дополняем текст нулевыми битами и отправляем на шифрование. Для

шифрования генерируем некое случайное число, которое будет служить для некой синхронизации сеансовых ключей шифрования. Для генерации сеансового ключа мы используем общий секрет с собеседником и на основе его и некоего случайного числа с помощью Argon2 генерируем сеансовый ключ, которым и будет производить шифрование. После шифрования для корректного отображения в любой программе мы должны преобразовать полученные бинарные данные во что-то читаемое и гарантированно корректно отображаемое в любой программе. Для этого мы используем кодирование в Base64, что гарантирует нам отображение только символов английского алфавита, цифр и корректных знаков. После кодирования к полученной строке добавляем то самое синхронизирующее случайное число, сгенерированное ранее. Для возвращения строки пользователю мы так же используем буфер обмена и записываем в него полученную Base64 строку. Для возвращения пользователю имитируем нажатие горячей клавиши «Вставить». И, как говорилось ранее, возвращаем буфер обмена в исходное состояние.

Рисунок 2.13 – Алгоритм работы программы при дешифровании

Для дешифрования поступаем аналогичным образом и обрабатываем нажатие горячей клавиши. Затем также сохраняем текущее состояние буфера обмена для дальнейшего его восстановления. Получаем сообщение собеседника, но уже с помощью имитации горячей клавиши «копировать», так как вмешиваться в состояние окон «чужой» программы мы не можем, а соответственно не сможем заменить полученный текст сообщения. Теперь буер обмена содержит Base64 строку с зашифрованными данными собеседника. Получаем из строки синхронизирующее случайное число, с помощью которого вместе с общим секретом мы получим сеансовый ключ шифрования. После удаления синхронизирующего значения мы производим обработку оставшегося текста. Так как для дешифрования нам необходимы бинарные данные, а имеем мы только Base64, то производим декодирование Base64 для получения бинарного сообщения. Далее с помощью Argon2 из полученного случайного числа и общего секрета мы получаем сеансовый

ключ. Дешифруем полученным ключом сообщение собеседника и отправляем его в главное окно программы. Для корректного отображения в главном окне, нам необходимо привести текст к требуемой кодировке. После корректного отображения мы восстанавливаем буфер обмена к исходному состоянию.

Работа с сессионными ключами шифрования будет происходить следующим образом:

– Получаем текст шифруемого сообщения.

– Генерируем из мастер-ключа и «одноразового» случайного числа значения на вход Argon2.

– Получаем из Argon2 сессионный ключ.

– Шифруем полученным сессионным ключом сообщение.

– Отправляем пользователю строку, состоящую из «одноразового» случайного числа и шифротекста.

Работа с сессионными ключами дешифрования будет происходить следующим образом:

– Получаем текст зашифрованного сообщения.

– Получаем из строки «одноразовое» случайное число.

– Генерируем из мастер-ключа и «одноразового» случайного числа значения на вход Argon2.

– Получаем из Argon2 сессионный ключ.

– Дешифруем полученным сессионным ключом сообщение.

– Отображаем пользователю полученное сообщение.

Для гарантии того, что зашифрованные данные отобразятся у любого пользователя, после шифрования и перед дешифрованием происходит кодирование/декодирование Base64.

3 Используемые алгоритмы и функции

3.1 AES-128

Rijndael – симметричный блочный шифр, который может обрабатывать данные блоками по 128 бит, используя ключи шифрования длиной 128, 192 и 256 бит. Rijndael спроектирован так, что позволяет использовать и другие длины блоков и ключей. Но данный шифр многие знают под названием AES.

Входом и выходом алгоритма AES являются последовательности из 128 бит. Эти последовательности рассматриваются как блоки. Количество бит в блоке будет называться длиной блока. Ключ шифрования для алгоритма AES – это последовательность из 128, 192 или 256 бит.

Базовым элементом, которым оперирует алгоритм AES, является байт – последовательность из восьми бит, обрабатываемых как единое целое. Описанные ранее последовательности бит, обрабатываются как массивы байт.

Внутри алгоритма AES выполняются операции над двумерным массивом байт, называемым матрицей состояния. Матрица состояния образована четырьмя строками, каждая из которых содержит Nb байт, где Nb – длина блока, делённая на 32. В матрице состояния обозначается символом s.

В начале процедур шифрования и дешифрования, вход копируется в матрицу состояния. Затем процедуры шифрования и дешифрования управляют матрицей состояния, после чего её финальное значение копируется в выход.

Четыре байта в каждом столбце матрицы состояния образуют 32-битные слова. Номер строки является для этих четырёх байт индексом в пределах каждого слова. Таким образом, матрица состояния может быть интерпретирована как одномерный массив 32-битных слов (столбцов).

В алгоритме AES все байты рассматриваются в качестве элементов конечного поля. Элементы конечного поля можно складывать и умножать. Но эти операции отличаются от принятых для обычных чисел.

3.1.1 Сложение

Сложение двух элементов в конечном поле производится путём «сложения» коэффициентов при соответствующих степенях многочленов, представляющих эти два элемента. Сложение выполняется с помощью операции XOR, то есть по модулю 2. Следовательно, вычитание многочленов равнозначно их сложению. Альтернативно сложение элементов конечного поля может быть описано, как сложение по модулю 2 соответствующих бит в байте.

3.1.2 Умножение

При представлении байт в виде многочленов операция умножения в поле $GF(2^8)$ соответствует умножению многочленов по модулю, представляющему собой неприводимый многочлен степени 8. Многочлен является неприводимым, если его делителями являются только единица и он сам. Для алгоритма AES этим неприводимым многочленом является:

$$m(x) = x^8 + x^4 + x^3 + x + 1. \quad (3.1)$$

Приведение к модулю m(x) гарантирует, что результат будет двоичным многочленом со степенью меньше 8 и, следовательно, может быть представлен в виде байта. В отличие от сложения, на уровне байт нет простой операции, которая соответствовала бы умножению.

Произведение двоичного многочлена на многочлен x вычисляется путём приведения полученного простым произведением многочлена к модулю m(x), определённому выражением (3.1). Умножение на x может быть выполнено на уровне байт как сдвиг влево и последующий условный побитовый XOR с числом 00011011. Эта операция над байтами обозначается xtime(). Умножение на более высокие степени x может быть выполнено путём повторения операции xtime(). Умножение на любую константу может быть выполнено путём сложения промежуточных результатов.

3.1.3 Описание алгоритма

В алгоритме AES входной блок, выходной блок и матрица состояния содержат 128 бит. Это показано равенством Nb = 4, где Nb отражает количество 32-битных слов в матрице состояния. В алгоритме AES длина ключа шифрования K равна 128, 192 или 256 бит. Длина ключа показана переменной Nk, равной 4, 6 или 8 и отражающей количество 32-битных слов в ключе шифрования. В алгоритме AES количество раундов, выполняющихся в процессе работы алгоритма, зависит от длины ключа. Количество раундов показано переменной Nr, где Nr = 10, когда Nk = 4; Nr = 12, когда Nk = 6; и Nr = 14, когда Nk = 8.

В алгоритме AES в процедуре шифрования и процедуре дешифрования используется функция раунда, состоящая из четырёх различных байт-ориентированных преобразований. Этими преобразованиями являются:

– Замена байт с помощью таблицы (S-блока).

– Сдвиг строк матрицы состояния на различную величину.

– Перемешивание данных в пределах каждого столбца матрицы состояния.

– Сложение ключа раунда с матрицей состояния.

В начале процедуры шифрования вход копируется в матрицу состояния. Затем производится первое сложение матрицы состояния и ключа раунда. После этого матрица состояния преобразуется с помощью функции раунда 10, 12 или 14 раз в зависимости от длины ключа. Причём последний раунд немного отличается от предыдущих Nr - 1 раундов. Конечное значение матрицы состояния копируется в выход.

Функция раунда использует в качестве параметра массив подключей, который представляет собой одномерный массив 4-байтных слов. Эти слова вычисляются с помощью процедуры расширения ключа. Отдельные преобразования – SubBytes(), ShiftRows(), MixColumns() и AddRoundKey() – обрабатывают матрицу состояния.

Преобразование SubBytes() выполняет нелинейную замену байт матрицы состояния с помощью таблицы (S-блока). При этом каждый байт обрабатывается независимо от других.

Преобразование ShiftRows() выполняет циклический сдвиг байт в трёх последних строках матрицы состояния на различное число байт (смещений). Первая строка не сдвигается. В итоге байты сдвигаются на «младшие» позиции в строке, в то время как «самые младшие» байты перемещаются вокруг строки в её «вершину».

Преобразование MixColumns() обрабатывает матрицу состояния столбец за столбцом. Каждый столбец используется в качестве четырёхчленного многочлена. Столбцы рассматриваются как многочлены в поле $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на фиксированный многочлен a(x):

$$a(x) = 3x^3 + x^2 + x + 2. \quad (3.2)$$

Преобразование AddRoundKey() выполняет сложение ключа раунда и матрицы состояния с помощью простой побитовой операции XOR. Каждый ключ раунда состоит из Nb слов массива подключей. Каждое из этих Nb слов складывается со столбцами матрицы состояния.

Процедура расширения ключа используется в алгоритме AES для создания массива подключей из ключа шифрования K. Всего генерируется Nb(Nr + 1) слов. В начале алгоритма требуется Nb слов и затем в каждом из Nr раундов требуется Nb слов ключевых данных.

Функция SubWord() принимает на вход 4-байтное слово. Выходное слово формируется путём замены каждого из этих четырёх байт с помощью S-блока.

Функция RotWord() принимает на вход слово и выполняет циклическую перестановку.

Первые Nk слов расширенного ключа заполняются ключом шифрования. Каждое последующее слово, вычисляется путём выполнения

операции XOR между предыдущим словом, и словом, находящимся на Nk позиций раньше.

При вычислении слов, находящихся на позициях кратных Nk, над предыдущим словом производятся дополнительные операции. Сначала в слове производится циклический сдвиг байт функцией RotWord(). Затем функция SubWord() изменяет все четыре байта слова с помощью таблицы. После чего выполняется операция XOR между выходом функции SubWord() и словом из массива констант.

Важно отметить, что процедура расширения ключа для 256-битных ключей шифрования (Nk = 8) немного отличается от процедуры для 128- и 192-битных ключей. Если Nk = 8 и i – 4 кратно Nk, то перед операцией XOR предыдущее слово обрабатывается функцией SubWord().

Преобразования, составляющие процедуру шифрования, могут быть инвертированы и применены в обратном порядке для получения прямой процедуры дешифрования алгоритма AES. Отдельные преобразования, использующиеся в процедуре дешифрования – InvShiftRows(), InvSubBytes(), InvMixColumns() и AddRoundKey() – обрабатывают матрицу состояния.

Преобразование InvShiftRows() является обратным к ShiftRows(). Байты в трёх последних строках матрицы состояния циклически сдвигаются на различное число байт (смещений). Первая строка (с номером r = 0) не сдвигается. Остальные строки сдвигаются аналогично ShiftRows().

Преобразование InvSubBytes() является обратным к преобразованию замены байт. Преобразование InvSubBytes() изменяет каждый байт матрицы состояния с помощью инвертированного S-блока. Для этого выполняется преобразование, обратное аффинному преобразованию, и затем находится обратный по умножению элемент в поле $GF(2^8)$.

Преобразование InvMixColumns() является обратным к MixColumns(). Преобразование InvMixColumns() обрабатывает матрицу состояния столбец за столбцом. Каждый столбец используется в качестве четырёхчленного

многочлена. Столбцы рассматриваются как многочлены в поле $GF(2^8)$ и умножаются по модулю $x^4 + 1$x на фиксированный многочлен $a^{-1}(x)$:

$$a^{-1}(x) = 11x^3 + 13x^2 + 9x + 14. \quad (3.3)$$

Преобразование AddRoundKey(), описанное ранее, является обратным само к себе, так как включает в себя только операцию XOR.

В прямой процедуре дешифрования порядок следования преобразований отличается от порядка, применённого в процедуре шифрования. При этом для шифрования и дешифрования используется одинаковый массив подключей. Однако некоторые свойства алгоритма AES позволяют использовать эквивалентную процедуру дешифрования, в которой последовательность преобразований совпадает с последовательностью в процедуре шифрования (при этом преобразования заменяются на обратные). Это достигается путём изменения массива подключей.

Двумя свойствами, благодаря которым возможна эквивалентная процедура дешифрования, являются:

– Преобразования SubBytes() и ShiftRows() коммутативны. То есть преобразование SubBytes(), следующее сразу после преобразования ShiftRows(), будет эквивалентно преобразованию ShiftRows(), следующему сразу после преобразования SubBytes(). То же верно и для обратных к ним преобразований – InvSubBytes() и InvShiftRows();

– Операции перемешивания столбцов – MixColumns() и InvMixColumns() – являются линейными по отношению к входному столбцу.

Эти свойства позволяют поменять местами преобразования InvSubBytes() и InvShiftRows(). Преобразования AddRoundKey() и InvMixColumns() тоже можно поменять местами, если при этом столбцы (слова) массива подключей дешифрования будут изменены преобразованием InvMixColumns().

Эквивалентная процедура дешифрования образуется путём обмена местами преобразований InvSubBytes() и InvShiftRows(), а также обмена

местами в “цикле раунда” преобразований AddRoundKey() и InvMixColumns(). При этом подключи дешифрования, используемые с 1 по Nr -1 раунд, должны быть изменены преобразованием InvMixColumns(). Но первые и последние Nb слов массива подключей дешифрования не должны быть им изменены.

С учётом этих изменений структура эквивалентной процедуры дешифрования является более эффективной.

3.2 Argon2

В современном мире остро стоит вопрос замедления хэширования. Это было еще до введения моды на быстрые алгоритмы по нахождению исходного значения для конкретного хэша. Существуют разные способы замедления, такие как применение хэширования несколько раз подряд или использование соли, но GPU и специальные устройства ускоряют перебор, с которым сложно бороться, даже с помощью таких вещей, как bcrypt [12].

Для решения возникшей проблемы было решено провести соревнование, результатом которого должен был стать алгоритм, который сложно ускорить на специальных устройствах и GPU, при этом он должен быть настраиваемым в зависимости от пожеланий разработчика. Таким алгоритмом стал Argon2.

Argon2 позволяет настраивать следующие параметры хэширования:

– Количество итераций.

– Желаемый объем занимаемой памяти.

– Степень параллелизма.

– Размер результата, в байтах.

– Секретный ключ.

– Дополнительные данные.

Так же, KDF [13] Argon2 реализован в двух вариантах: Argon2i и Argon2d. Argon2i делает больше проходов по памяти и более медленный, Argon2d быстрее, но у него отсутствует защита от timing атак, а также его сложнее подбирать на GPU. Чаще всего выбирают Argon2i.

Argon2i рекомендуется для хэширования паролей, Argon2d — для криптовалют, там timing атаки не страшны.

Алгоритм оптимизирован именно для архитектуры x86/x64, поэтому его крайне сложно ускорять на ASIC/GPU и прочих устройствах. Используется многократный проход по памяти, внутри формируется матрица хэшей большого объёма, которые зависят друг от друга и сложным образом обрабатываются.

3.3 Base64

Позволяет кодировать информацию, представленную набором байтов, используя всего 64 символа: A-Z, a-z, 0-9, /, +. В конце кодированной последовательности может содержаться несколько спецсимволов (обычно =).

Преимущества:

– Позволяет представить последовательность любых байтов в печатных символах.

– В сравнении с другими Base-кодировками дает результат, который составляет только 133.(3)% от длины исходных данных.

Недостатки:

– Регистрозависимая кодировка.

Помимо прочего, так как мы получаем данные в кодировке Unicode, то каждый символ занимает 2 байта, но после кодирования в Base64 каждый символ занимает 1 байт, что в 2 раза сокращает объём требуемой памяти.

4 Демонстрация работоспособности

Если приложение не может вырезать текст из сторонней программы, то можно использовать функцию шифрования буфера. По окончанию шифрования будет выведено справочное окно.

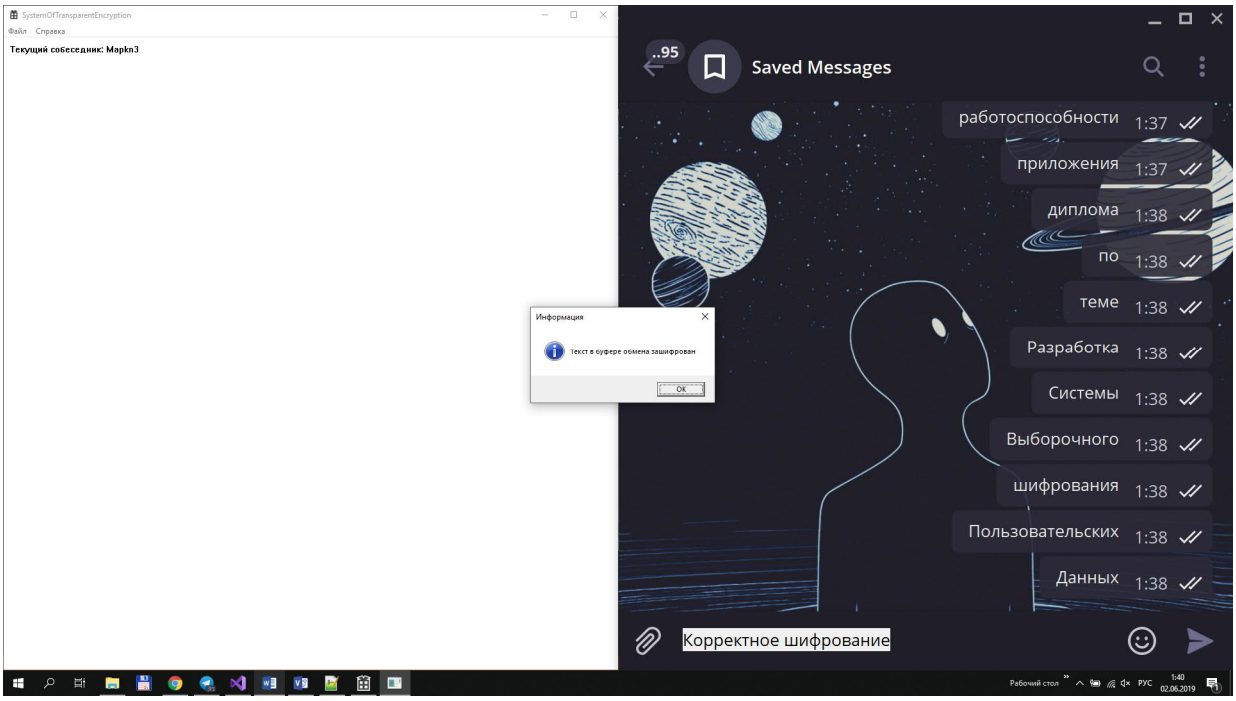

Рисунок 2.14 – Демонстрация шифрования в мессенджере Telegram

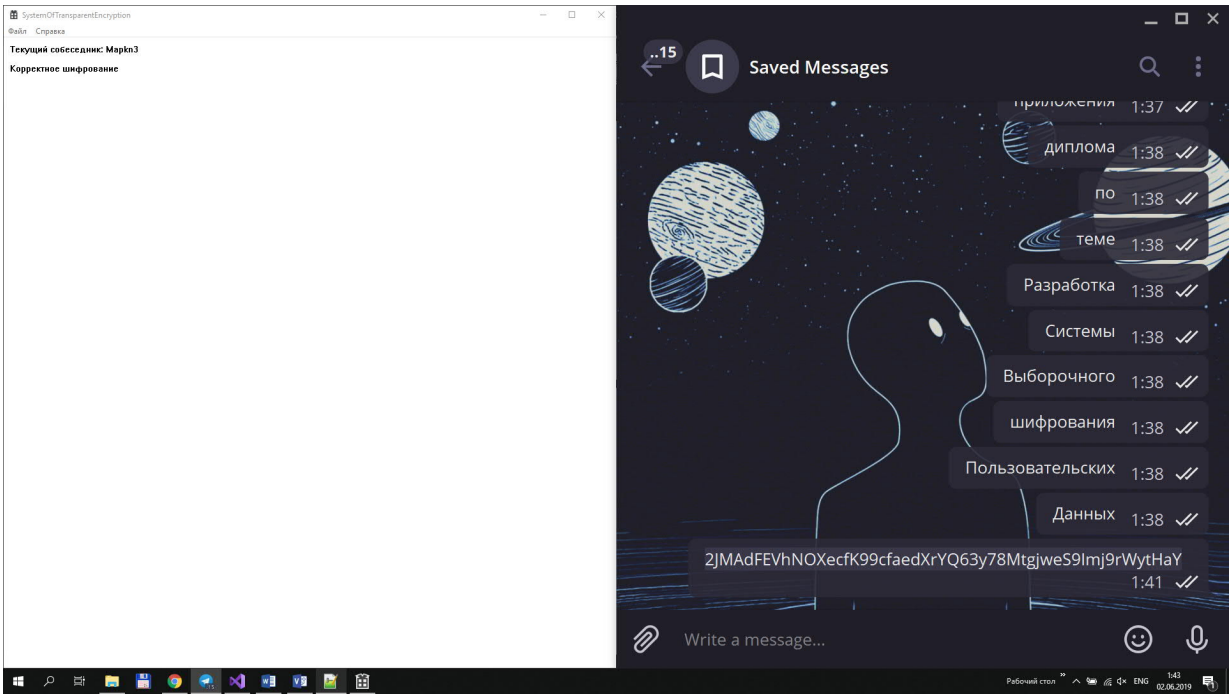

Рисунок 2.15 – Демонстрация дешифрования в мессенджере Telegram

При использовании чужого ключа другого собеседника, текст полученного сообщения не будет корректно дешифрован, и пользователь получит «нечитаемые» символы.

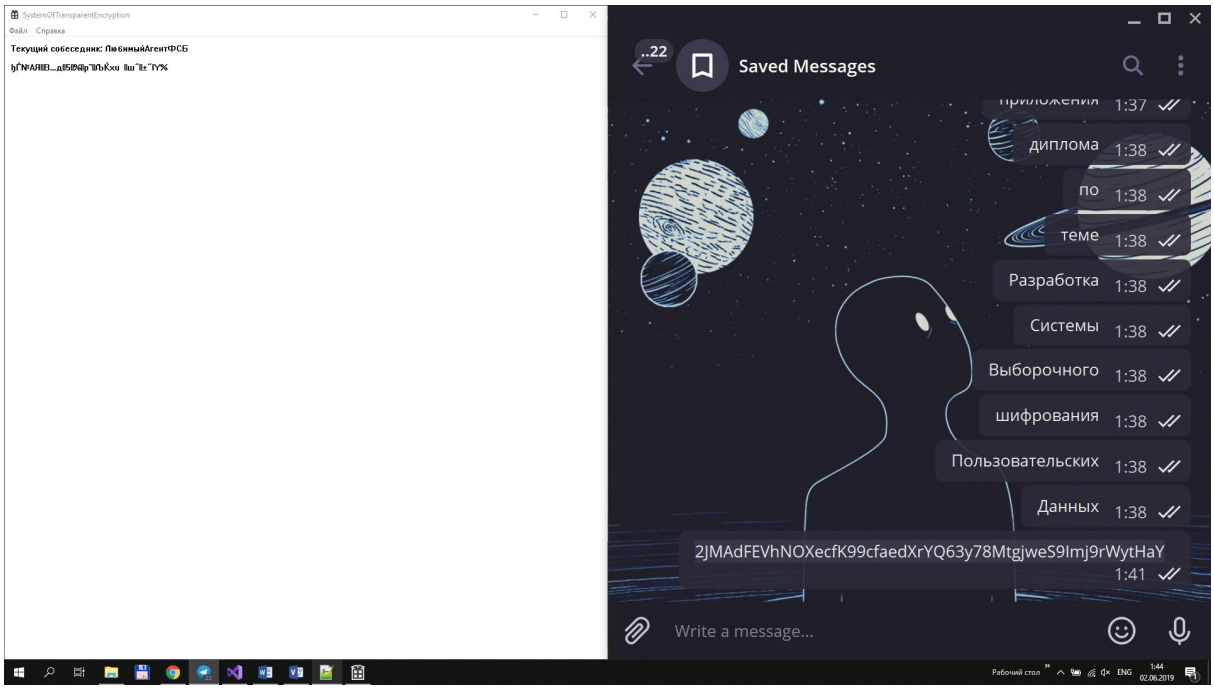

Рисунок 2.16 – Демонстрация дешифрования в мессенджере Telegram чужим ключом другого собеседника

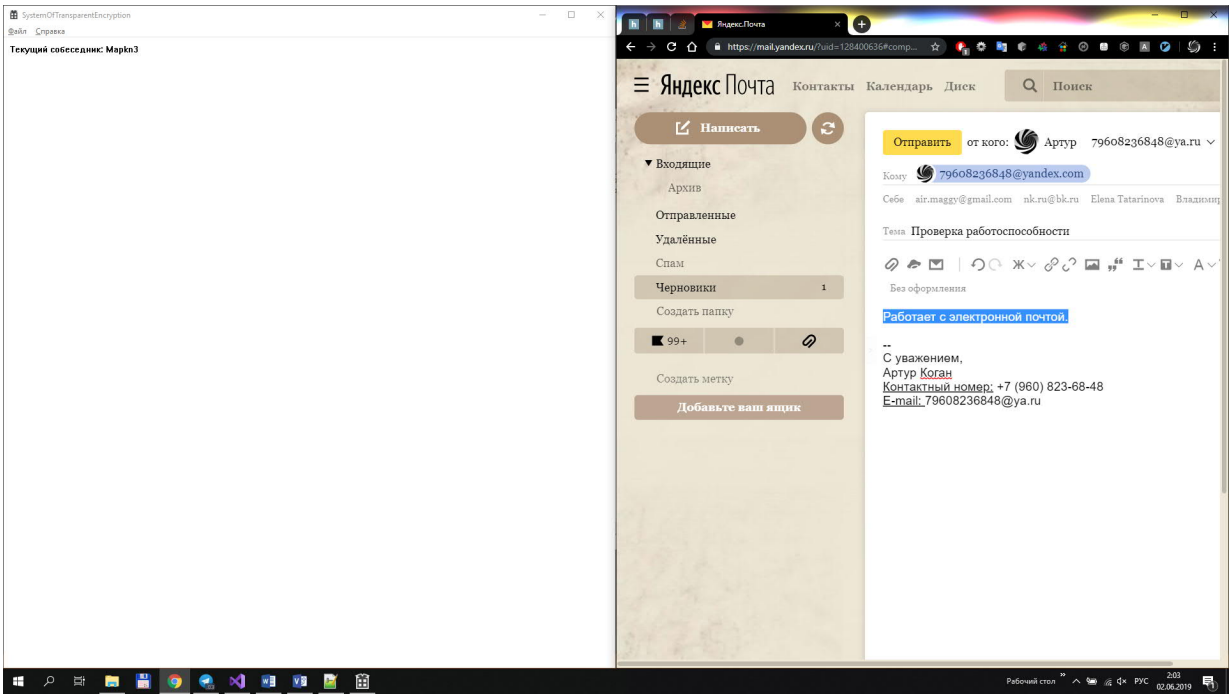

Рисунок 2.17 – Демонстрация шифрования электронного письма в браузере

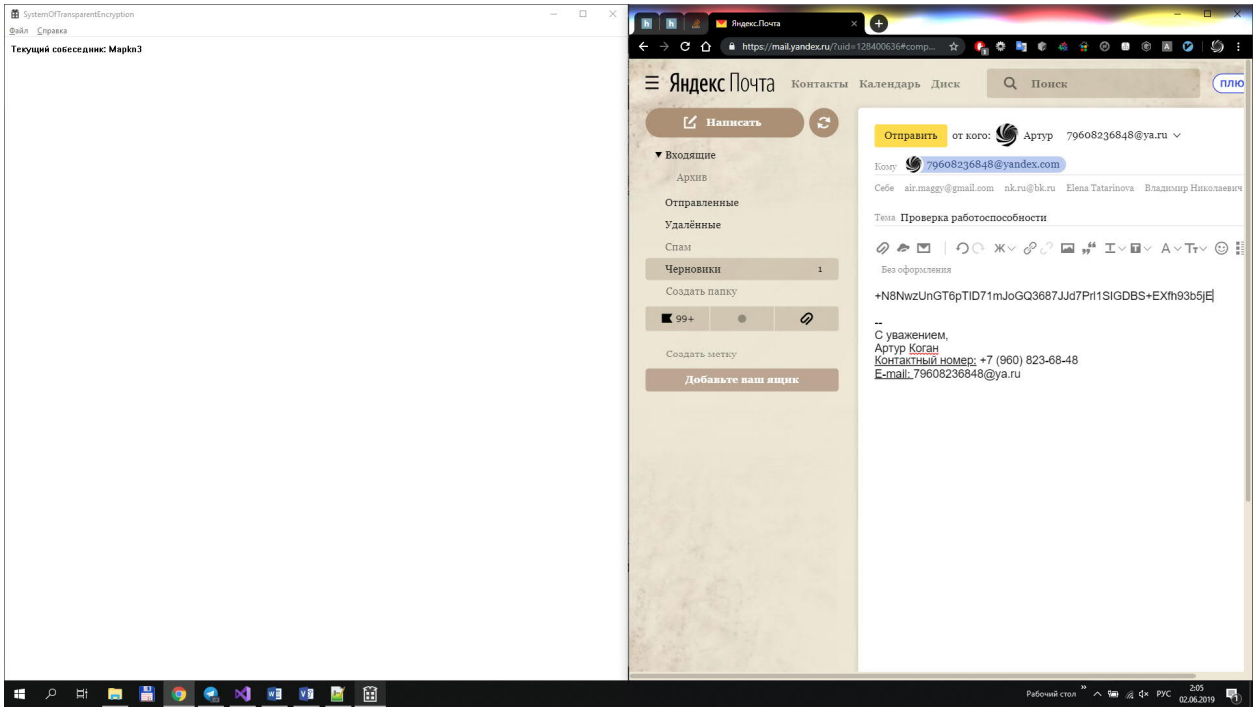

Рисунок 2.18 – Отправка зашифрованного электронного письма

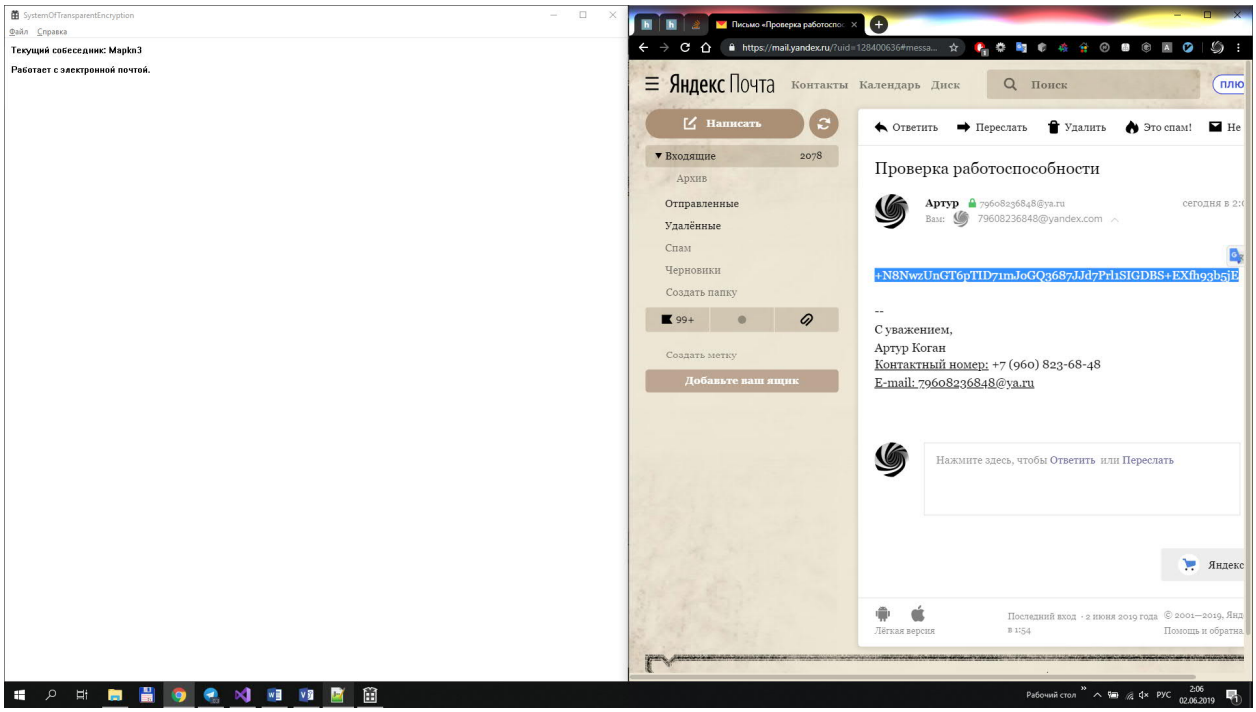

Рисунок 2.19 – Демонстрация дешифрования электронного письма в браузере

ЗАКЛЮЧЕНИЕ

После анализа существующих решений были выяснены основные достоинства и недостатки таких систем, а также требуемые режимы работы и уровень вмешательства конечного пользователя.

Данная программа направлена на работу со средствами обмена сообщениями, которые имею программы-клиенты для связи по сети Internet. За счёт перехвата сообщений во время отправки и получения между программой-клиентом и каналом связи, а также отсутствия действий и вмешательств в работу конечного пользователя, разрабатываемое приложение удовлетворяет требованиям системы выборочного шифрования.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Защищенная электронная почта X.400: [Электронный ресурс] // МОПНИЭИ Средства Криптографической Защиты Информации – URL: https://security.ru/default.php?target=x400&style=products (дата обращения 16.03.2019).

2. Главная страница ProtonMail: [Электронный ресурс] // ProtonMail – URL: https://protonmail.com/ru/ (дата обращения 16.03.2019).

3. Главная страница Delta Chat: [Электронный ресурс] // Delta Chat – URL: https://delta.chat/ru/ (дата обращения 16.03.2019).

4. Главная страница OpenPGP: [Электронный ресурс] // OpenPGP – URL: https://www.openpgp.org/ (дата обращения 16.03.2019).

5. Главная страница OpenVPN: [Электронный ресурс] // OpenVPN – URL: https://openvpn.net/ (дата обращения 16.03.2019).

6. Главная страница ZeroTier: [Электронный ресурс] // ZeroTier – URL: https://www.zerotier.com/ (дата обращения 16.03.2019).

7. Создание классических приложений для компьютеров с Windows: [Электронный ресурс] // Центр разработки для Windows – URL: https://docs.microsoft.com/ru-ru/windows/apps/desktop/ (дата обращения 06.04.2019).

8. The Base16, Base32, and Base64 Data Encodings: [Электронный ресурс] // IETF Tools – URL: https://tools.ietf.org/html/rfc4648/ (дата обращения 07.04.2019).

9. Брюс Шнайер: Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си [Текст] / Брюс Шнайер // Триумф — 2012.

10. Argon2: the memory-hard function for password hashing and other applications: [Электронный ресурс] // GitHub – URL: https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf (дата обращения 04.05.2019).

11. Advances Encryption Standart (AES): [Электронный ресурс] // National Institute of Standards and Technology – URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (дата обращения 04.05.2019).

12. A Future-Adaptable Password Scheme: [Электронный ресурс] // The OpenBSD Project – URL: http://www.openbsd.org/papers/bcrypt-paper.pdf (дата обращения 04.05.2019).

13. Boneh, D. A Graduate Course in Applied Cryptography [Текст] / Dan Boneh, Victor Shoup // Stanford University — 2017.