Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Колледж информатики и программирования

ДОПУСКАЮ К ЗАЩИТЕ
заместитель директора колледжа
по учебно-производственной
работе
_____ Л.В. Фокина
«____» июня 2019г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему: Разработка программы шифрования файлов

Студент группы 4ПКС-
Черников Алексей Владимирович «____» июня 2019г.

Основная профессиональная образовательная программа по специальности

09.02.03 Программирование в компьютерных системах

Форма обучения очная

Руководитель ВКР _____Аксёнова Т.Г.

Председатель предметно-цикловой комиссии _____Пестов А.И.

Москва
2019

Содержание

Введение

Проблема защиты информации путем её преобразования волновала человечество с давних времен. С распространением письменности появилась потребность в обмене письмами и сообщениями, что вызвало необходимость сокрытия их содержимого от посторонних. Так начала формироваться наука под названием криптография и именно поэтому её можно считать ровесницей истории человеческого языка.

Первые криптосистемы встречаются уже в начале нашей эры. В основном развитию криптографии способствовали войны. Письменные приказы и донесения обязательно шифровались, чтобы пленение курьеров не позволило противнику получить важную информацию. Так, древнеримский политический деятель и полководец Гай Юлий Цезарь в своих переписках активно использовал уже более-менее систематический шифр, получивший его имя.

Бурное развитие криптографические системы получили в годы первой и второй мировых войн. Начиная с послевоенного времени и по нынешний день появление вычислительных средств ускорило разработку и совершенствование криптографических методов.

Почему проблема использования криптографических методов стала в настоящий момент особенно актуальна?

На сегодняшний день криптография является одним из наиболее мощных средств обеспечения конфиденциальности и контроля целостности информации. Во многих отношениях она занимает центральное место среди программно-технических регуляторов безопасности. Например, для портативных компьютеров, физически защитить которые крайне трудно, только криптография позволяет гарантировать конфиденциальность информации даже в случае кражи. Что уж говорить про важность криптографии в глобальной сети «Интернет», по которой ежедневно передаются колоссальные объемы информации государственного, военного,

коммерческого и частного характера, которая нуждается в защите от доступа к ней посторонних лиц.

Переоценить ее возможности для человечества сложно. С момента появления она прошла множество модификаций и сейчас представляет собой систему безопасности, которая практически не может быть взломана. Современные методы криптографии применяются практически во всех отраслях, в которых присутствует необходимость безопасной передачи или хранения данных.

Целью данной выпускной квалификационной работы является разработка программы шифрования файлов (и их дешифровки соответственно), которая позволила бы пользователю защитить желаемую информацию наиболее эффективными методами.

Для достижения поставленной цели необходимо выполнить ряд следующих задач:

- детально изучить предметную область;
- провести сравнительный анализ и на его основе выбрать наиболее эффективные алгоритмы шифрования;
- составить необходимые модели и диаграммы;
- разработать функциональные и нефункциональные требования к программе;
- выбрать подходящие для разработки инструментальные средства и обосновать свой выбор;
- реализовать функциональные и нефункциональные требования;
- провести тестирование и отладку ошибок в программе;
- составить руководство по использованию программы.

Глава 1 Теоретическая часть

1.1 Предпроектное обследование

Криптография – это наука, изучающая способы сокрытия данных и обеспечения их конфиденциальности. Это одна из старейших наук и ее история насчитывает четыре тысячелетия. Сам термин «криптография» образовался от двух древнегреческих слов «крипто» – скрытый, «графо» – пишу.

До 1975 года криптография представляла собой шифровальный метод с секретным ключом, который предоставлял доступ к расшифровке данных. Позже начался период ее современного развития и были разработаны методы криптографии с открытым ключом, которые может передаваться по открытым каналам связи и использоваться для проверки данных.

Современная прикладная криптография представляет собой науку, образованную на стыке математики и информатики. Смежной наукой криптографии считается криптоанализ. Криптография и криптоанализ тесно взаимосвязаны между собой, только в последнем случае изучаются способы расшифровки сокрытой информации.

С модификацией до открытого ключа криптография получила более широкое распространение и стала применяться частными лицами и коммерческими организациями, а в 2009 году на ее основе была выпущена первая криптовалюта Биткоин. До этого времени она считалась прерогативой государственных органов правления.

В основе криптографических систем лежат различные виды криптографии. Всего различаю четыре основных криптографических примитива:

- симметричное шифрование – данный метод предотвращает перехват данных третьими лицами и базируется на том, что отправитель и получатель данных имеет одинаковые ключи для разгадки шифра;

- асимметричное шифрование – в этом методе задействованы открытый и секретный ключ. Ключи взаимосвязаны – информация, зашифрованная открытым ключом, может быть раскрыта только связанным с ним секретным ключом. Применять для разгадки ключи из разных пар невозможно, поскольку они связаны между собой математической зависимостью;
- хэширование – метод основывается на преобразовании исходной информации в байты заданного образца. Преобразование информации называется хэш-функцией, а полученный результат хэш-кодом. Все хэш-коды имеют уникальную последовательность символов;
- электронная подпись – это преобразование информации с использованием закрытого ключа, позволяющее подтвердить подлинность документа и отсутствие искажений данных.

Изначально криптография использовалась правительством для безопасного хранения или передачи документов. Современные же асимметричные алгоритмы шифрования получили более широкое применение в сфере IT-безопасности, а симметричные методы сейчас применяются преимущественно для предотвращения несанкционированного доступа к информации во время хранения.

В частности, криптографические методы применяются для:

- безопасного хранения информации коммерческими и частными лицами;
- реализации систем цифровой электронной подписи;
- подтверждения подлинности сертификатов;
- защищенной передачи данных онлайн по открытым каналам связи.

В связи с описанным выше передо мной была поставлена задача разработать программу для шифрования файлов и их дешифровке. Для этого были выбраны следующие алгоритмы:

- моноалфавитный шифр;
- полиалфавитный шифр;

- двухлитерный шифр;
- омофонический шифр;
- исключающее ИЛИ (XOR);
- Rivest, Shamir, Adleman (RSA);
- Digital Signature Algorithm (DSA);
- Advanced Encryption Standard (AES).

1.2 Характеристика инструментальных средств разработки

Для написания программы использовалась среда программирования «Microsoft Visual Studio 2017», представляющая собой полный набор средств разработки для создания веб-приложений ASP.NET, XML (веб-службы), настольных приложений и мобильных приложений. «Visual Studio» использует единую интегрированную среду разработки (IDE), которая позволяет совместно использовать средства и упрощает создание решений на базе нескольких языков. Кроме того, в этих языках используются функциональные возможности платформы .NET Framework, которая позволяет получить доступ к ключевым технологиям, упрощающим разработку веб-приложений ASP и XML.

«Microsoft Visual Studio» объединяет в себе огромное количество функций, позволяющих осуществлять разработки для Windows всех версий, Интернета, SharePoint, различных мобильных устройств и облачных технологий. В Visual Studio реализуется новая среда разработчика, благодаря которой создавать приложения стало проще.

В качестве системы для построения клиентских приложений была выбрана Windows Presentation Foundation (WPF).

WPF – это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF входит в состав .NET Framework, поэтому вы можете создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

При разработке поведения приложения главной задачей является обеспечение реакции на действия пользователя, включая обработку событий (таких как выбор пункта меню или нажатие на кнопку), и вызов в ответ бизнес-логики и логики доступа к данным. В WPF такое поведение реализуется в коде, связанном с разметкой. Этот код называется кодом программной части.

Возможности взаимодействия с пользователем, обеспечиваемые моделью приложения, реализуются с помощью сконструированных элементов управления. В WPF элемент управления – это общий термин, который относится к категории классов WPF, размещаемых в окне или на странице, имеющих пользовательский интерфейс и реализующих некоторое поведение.

В качестве языка программирования был выбран язык C# версии 7.0. NET Framework 4.7.1.

Язык C# появился на свет в июне 2000 г., в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом.

Создание инструментария для разработчиков с их полноценной поддержкой является одной из главных задач нового языка C#.

Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков с богатством возможностей и мощью C++.

Особенности C#:

- полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализаций, виртуальных функций и перегрузки операторов;
- полный и хорошо определенный набор основных типов;
- встроенная поддержка автоматической генерации XML-документации;
- возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции;
- автоматическое освобождение динамически распределенной памяти;
- полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API;
- указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого;
- поддержка свойств и событий в стиле VB;
- простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX;
- поддержка как Framework Class Library (FCL), так и Common Language Runtime (CLR);
- возможность использования C# для написания динамических web-страниц ASP.NET.

Для создания контекстной справки использовалась программа «Htm2chm». Она представляет собой проприетарный формат файлов контекстной справки, разработанный корпорацией Microsoft и выпущенный в 1997 году в качестве замены формата WinHelp. Содержит в себе набор HTML-страниц, может также включать в себя содержание со

ссылками на страницы, предметный указатель, а также базу для полнотекстового поиска по содержимому страниц.

Преимуществами программы можно считать:

- размер файла меньше, чем у обычного HTML;
- используются все возможности форматирования, имеющиеся в HTML и CSS;
- возможность полнотекстового поиска;
- возможность просмотра множества .chm-файлов как один, с общим содержанием и предметным указателем.

Для построения схем и диаграмм при составлении технического задания и сопровождающей документации было решено использовать Draw.io. Draw.io – это сервис, предназначенный для формирования диаграмм и схем. Сервис разделён на три части – меню, панель объектов и сам документ.

С помощью веб-сервиса Draw.io можно создавать:

- диаграммы;
- UML-модели;
- вставка в диаграмму изображений;
- графики;
- блок-схемы;
- формы;
- другое.

Также доступен экспорт готовых схем в изображение (PNG, GIF, JPG, PDF) и синхронизация полученных документов с Google Диском.

Глава 2 Практическая часть

2.1 Постановка задачи

2.1.1 Описание входных данных

Описание входной информации включает в себя описание входных документов и/или входных данных задачи. Формы входных документов рекомендуется оформлять в виде приложения к пояснительной записке. Входных документов в задаче может не быть, например, при разработке компьютерных игр. В этом случае следует ограничиться описанием входных данных задачи. Описание входных данных рекомендуется оформлять в виде таблицы с полями: Наименование, Идентификатор, Тип данных, Размер.

2.1.2 Описание выходных данных

Описание выходной информации включает в себя описание выходных документов и/или выходных данных задачи. Формы выходных документов рекомендуется оформлять в виде приложения к пояснительной записке. Выходных документов в задаче также может не быть, в этом случае следует ограничиться описанием выходных данных задачи. Описание выходных данных рекомендуется оформлять в виде таблицы с полями: Наименование, Идентификатор, Тип данных, Размер.

2.1.3 Математическая модель задачи

Математическая модель задачи включается в пояснительную записку только для задач вычислительного типа и содержит все формулы и уравнения, используемые при написании программного кода, с подробным описанием коэффициентов, входящих в их состав.

2.1.4 Требования к программному обеспечению

2.1.4.1 Функциональные требования

Что должна делать программа

2.1.4.2 Нефункциональные требования

Требования к интерфейсу, требования к реализации и требования к надежности. В требованиях к надежности необходимо указать способы защиты информации в программе

2.2 Анализ требований и определение спецификаций программного обеспечения

содержит определенный набор моделей и диаграмм (в зависимости от используемого подхода к разработке ПО):

- функциональную диаграмму;
- диаграмму потоков данных;
- диаграмму «сущность-связь» (при наличии подключаемой к программе базы данных);
- модели данных;
- диаграмму вариантов использования;
- диаграмму классов и другие.

Перечисленные выше диаграммы и модели оформляются в виде рисунков и могут выноситься в приложения пояснительной записки.

2.2.1 Диаграмма вариантов использования

2.2.2 Диаграмма потоков данных

2.2.3 Функциональная диаграмма

2.2.4 Диаграмма классов

2.3 Проектирование программного обеспечения

2.3.1 Структурная схема

2.3.2 Функциональная схема

2.4 Разработка пользовательских интерфейсов программного обеспечения

представляет собой скриншоты интерфейсов всех составных частей программы (подсистем) с отображением диалоговых окон, управляющих элементов и полей ввода информации.

2.5 Тестирование и отладка программного обеспечения

содержит примеры ввода в программу как верных, так и ошибочных входных данных с указанием реакции программы. Реакцию программы необходимо оформлять в виде скриншотов. Тестовые данные рекомендуется оформлять в виде таблицы с полями: № операции, Входные данные, Вводимое значение, Реакция программы. Также данный подраздел должен содержать краткий анализ приведенных в таблице тестовых данных, а также выводы о соответствии работы программного средства функциональным и нефункциональным требованиям, заявленным в предпроектном обследовании.

2.6 Руководство по использованию программы

2.6.1 Руководство системного программиста

2.6.1.1 Общие сведения о программе

В пункте «Общие сведения о программе» должны быть указаны назначение и функции программы и сведения о технических и программных средствах, обеспечивающих выполнение данной программы.

2.6.1.2 Структура программы

В пункте «Структура программы» должны быть приведены сведения о структуре программы, ее составных частях, о связях между составными частями и связях с другими программами.

2.6.1.3 Настройка программы

В пункте «Настройка программы» должно быть приведено описание действий по настройке программы на условия конкретного применения.

2.6.1.4 Проверка программы

В пункте «Проверка программы» должны быть приведено описание способов проверки, позволяющих дать общее заключение о работоспособности программы (контрольные примеры, методы прогона, результаты).

2.6.1.5 Дополнительные возможности

В пункте «Дополнительные возможности» должно быть приведено описание дополнительных разделов функциональных возможностей программы и способов их выбора.

2.6.1.6 Сообщения системному программисту

В пункте «Сообщения системному программисту» должны быть указаны тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, а также в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.6.2 Руководство программиста

2.6.2.1 Назначение и условия применения программы

В пункте «Назначение и условия применения программы» должны быть указаны назначение и функции, выполняемые программой, условия, необходимые для выполнения программы (системные требования).

2.6.2.2 Характеристики программы

В пункте «Характеристики программы» должно быть приведено описание основных характеристик и особенностей программы.

2.6.2.3 Обращение к программе

В пункте «Обращение к программе» должно быть приведено описание процедур вызова программы.

2.6.2.4 Входные и выходные данные

В пункте «Входные и выходные данные» должно быть приведено описание организации, используемой входной и выходной информации.

2.6.2.5 Сообщения

В пункте «Сообщения» должны быть указаны тексты сообщений, выдаваемых программисту или пользователю в ходе выполнения программы, описание их содержания и действия, которые необходимо предпринять по этим сообщениям.

2.6.3 Руководство пользователя

2.6.3.1 Назначение программы

В пункте «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

2.6.3.2 Условия выполнения программы

В пункте «Условия выполнения программ» должны быть указаны условия, необходимые для выполнения программы (системные требования).

2.6.3.3 Выполнение программы

В пункте «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, должно быть приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также описание реакции программы.

2.6.3.4 Сообщения пользователю

В пункте «Сообщения пользователю» должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия пользователя.

Заключение

//заключение

Список литературы

//список литературы в алфавитном порядке

Приложение А

Приложение Б

Приложение В