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-115

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 5](#_Toc9607034)

[Глава 1 Теоретическая часть 7](#_Toc9607035)

[1.1 Предпроектное обследование 7](#_Toc9607036)

[1.2 Характеристика инструментальных средств разработки 17](#_Toc9607037)

[Глава 2 Практическая часть 22](#_Toc9607038)

[2.1 Постановка задачи 22](#_Toc9607039)

[2.2 Анализ требований и определение спецификаций программного обеспечения 23](#_Toc9607040)

[2.3 Проектирование программного обеспечения 24](#_Toc9607041)

[2.4 Разработка пользовательских интерфейсов 24](#_Toc9607042)

[2.5 Тестирование и отладка программного обеспечения 24](#_Toc9607043)

[2.6 Руководство по использованию программы 25](#_Toc9607044)

[Заключение 29](#_Toc9607045)

[Список литературы 30](#_Toc9607046)

[Приложение А 31](#_Toc9607047)

[Приложение Б 32](#_Toc9607048)

[Приложение В 33](#_Toc9607049)

# 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 возможности криптографии 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 задач:

* детально изучить предметную область;
* провести сравнительный анализ и на его основе выбрать наиболее эффективные алгоритмы шифрования;
* разработать функциональные и нефункциональные требования к программе;
* составить необходимые диаграммы и схемы;
* выбрать подходящие для разработки инструментальные средства;
* реализовать функциональные и нефункциональные требования;
* провести тестирование и отладку программы;
* составить руководства по использованию программы.

Исходя из написанного выше можно сказать что объектом исследования будет являться – криптография, а предметом исследования – математические методы кодирования информации в современной криптографии.

# Теоретическая часть

## Предпроектное обследование

### Описание предметной области

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.

### Изучение аналогов

На мировом рынке существует множество программ по шифрованию файлов самыми различными способами. При поиске информации о них мне удалось выделить такие программы как:

* Pretty Good Privacy (PGP) Desktop;
* Folder Lock.

Начнём с PGP Desktop. Она представляет собой комплекс программ для шифрования, обеспечивающий гибкое многоуровневое шифрование. От Folder Lock, которая будет описана позднее, она отличается тем, что имеет тесную интеграцию в системную оболочку, а доступ к её функциям осуществляется через контекстное меню проводника. Программа позволяет выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, которая предоставляется в электронном виде, а также шифровать информацию на съёмных носителях.

Шифрование PGP осуществляется последовательно: хешированием, сжатием данных, шифрованием с симметричным ключом, и, наконец, шифрованием с открытым ключом, причём каждый этап может осуществляться одним из нескольких поддерживаемых алгоритмов. Такой подход позволяет достичь сильной защищённости информации.

К сожалению, PGP Desktop имеет низкую производительность и отсутствие поддержки русского языка.

Программа Folder Lock работает почти по тем же принципам. Заходя в программу сразу же хочется отметить простой и удобный интерфейс, который позволяет быстро начать работу. Используя эту программу можно:

* скрыть конфиденциальные данные в папках, на съёмных носителях и электронике;
* зашифровать конфиденциальные данные в папках, на съёмных носителях и электронике;
* устанавливать пароли;
* хранить важную информацию в «облаке» на сайте производителя;
* создавать виртуальные зашифрованные диски;
* шифрование сообщений электронной почты;
* полное удаление файлов без какой-либо возможности восстановления;
* другое.

Как видно, возможностей у программы предостаточно (особенно для персонального использования). Однако также, как и в PGP Desktop отсутствует поддержка русского языка, поэтому для пользователей не знакомых с английским, могут возникнуть некоторые сложности при работе.

Подводя итоги изучения аналогов можно сказать что в разрабатываемой программе должно в обязательном порядке присутствовать русский язык и разные алгоритмы шифрования файлов, при работе которых программа будет иметь высокую производительность.

### Описание методов шифрования и принципов их работы

#### Транспозиция

Первым методом шифрования рассмотрим транспозицию или как его по-другому называют – шифр перестановки. Он относится к симметричным криптосистемам перестановочного типа. Принцип работы этого метода заключается в том, что элементы открытого текста меняются местами. Элементами текста могут быть как отдельные символы, так и их пары, тройки и так далее, а также комбинирование этих случаев.

Классическая криптография делит шифры перестановки на два класса:

* шифры простой перестановки – когда при шифровании символы открытого текста перемещаются с исходных позиций на новые один раз;
* шифры сложной перестановки – когда при шифровании символы открытого текста перемещаются с исходных позиций на новые несколько раз.

В разрабатываемой программе применяется первый класс подобных шифров – простой.

Как правило, при шифровании и дешифровании текста используется таблица перестановок, для примера, возьмём такую (рисунок 1.1).



Рисунок 1.1 – Таблица перестановок

Первая строка – номера символов в открытом тексте, вторая строка – номера позиций, которые должны занимать символы в шифрограмме (или проще говоря – это наш ключ, который задаёт порядок перестановки символов открытого текста).

Соответственно, для того, чтобы зашифровать исходное сообщение нужно разбить его на блоки, равные длине ключа. Как говорилось выше, кодирование осуществляется перестановкой букв (символов). Таким образом первый символ должен быть переставлен на второе место, второй на четвёртое, третий на пятое, четвёртый на первое, пятый на третье.

Если с помощью таблицы на рисунке 1.1 зашифровать текст «пример маршрутной перестановки» получиться таблица, продемонстрированная на рисунке 1.2.

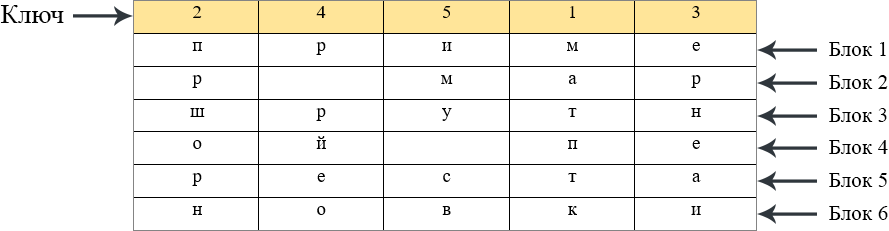


Рисунок 1.2 – Текст, зашифрованный методом транспозиции в табличном виде

В процессе шифрования символы текста переставляются в порядке, соответствующем заданному ключу по блокам (в данном примере по возрастанию) в результате чего получается следующая шифрограмма: «мпериарр мтшнрупоей траескниов».

Дешифрование производится в обратном порядке. На примере указанного ранее ключа: второй символ на первое, четвёртый на второе, третий на пятое и так далее.

Следует отметить, что при использовании любого блочного шифра могут возникать ситуации, когда текст не делиться на равные блоки. В таких случаях длину исходного текста увеличивают до тех пор, пока он не будет делиться на равные блоки длины заданного ключа.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования так как символы всего лишь переставляются на другие позиции.

Минусами этого метода можно считать сохранение частотных характеристик текста и малое количество возможных ключей шифрования, что делает его уязвимым к криптоатакам. Главным недостатком этого и других симметричных алгоритмов шифрования можно считать передачу ключа. Ведь для того, чтобы он не попал в чужие руки для его передачи требуется обеспечить дополнительную безопасность, его также требуется регулярно обновлять, а после его смены опять же возникает нужда в его безопасной передаче.

#### Моноалфавитный шифр

Следующий метод шифрования – моноалфавитный шифр или по-другому – шифр простой замены. Он относится к симметричным криптосистемам подстановочного типа. К этому типу относятся, наверное, самый известный шифр – шифр Цезаря, в котором каждый символ алфавита сдвигается на три позиции правее. Принцип работы подстановочных шифров сводится к созданию таблицы шифрования (по определённому алгоритму), в которой каждой букве открытого текста соответствует единственная сопоставимая ей буква шифротекста. Само же шифрование заключается в замене букв согласно созданной таблице. Как можно заметить, всё очень просто.

Отмечу также, что в шифрах замены не всегда подразумевается замена буквы на какую-то другую букву. Допускается использовать замену на число. Соответственно в создаваемой таблице каждой букве используемого алфавита приравнивается любое число.

В разрабатываемой программе используется вариант замены буквы на букву, поэтому рассмотрим принцип работы на основе шифра Цезаря.

Для упрощения воспользуемся русским алфавитом, состоящим только из заглавных букв. Следуя принципу работы описанному выше создадим таблицу шифрования, где каждая буква алфавита сдвигается на три позиции вправо. Из этого следует, что выбранный нами ключ равен числу три (рисунок 1.3).

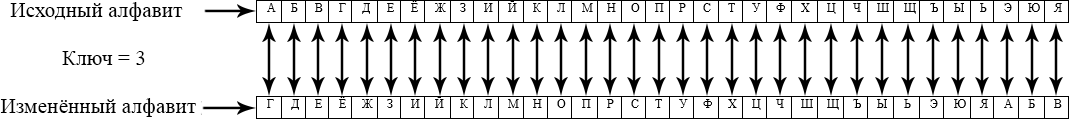


Рисунок 1.3 – Исходный алфавит и алфавит, сдвинутый на три позиции

Теперь, глядя на рисунок мы чётко видим, как будут взаимозаменяться буквы при шифровании и дешифровании текста. При шифровании буква «А» будет заменена на «Г», «Б» на «Д» и так далее, а при дешифровании обратно. Таким образом, зашифровав сообщение «ШИФР ЦЕЗАРЯ» мы получим «ЫЛЧУ ЩЗКГУВ».

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования.

Минусами этого метода можно считать то, что в шифротексте не скрывается частота появления символов открытого текста, что делает его уязвимым к криптоатакам и то, что максимальный количество ключей равно количеству букв в используемом алфавите. Также здесь имеется проблема передачи больших объёмов текста (чем больше текст, тем легче его взломать). Так как алгоритм относится к симметричным опять же требуется дополнительная безопасность при передаче ключа и его регулярное обновление.

#### Полиалфавитный шифр

Рассмотрим метод шифрования – полиалфавитный шифр или по-другому – многоалфавитный шифр.

Как и моноалфавитный шифр он относится к симметричным криптосистемам подстановочного типа принцип работы которых был описан ранее.

Суть работы полиалфавитного шифра заключается в циклическом применении нескольких моноалфавитных шифров к некоторому количеству букв открытого текста.

Рассмотрим, как это работает на примере. Для упрощения воспользуемся русским алфавитом, состоящим только из заглавных букв. Пусть в качестве ключа будет использоваться слово «КЛЮЧ». Это слово мы можем поделить на четыре отдельных моноалфавита в каждом из которых будет произведён сдвиг на количество символов равное номеру отдельной буквы ключевого слова в алфавите. Сделав это, мы получим следующие моноалфавиты (рисунок 1.4).

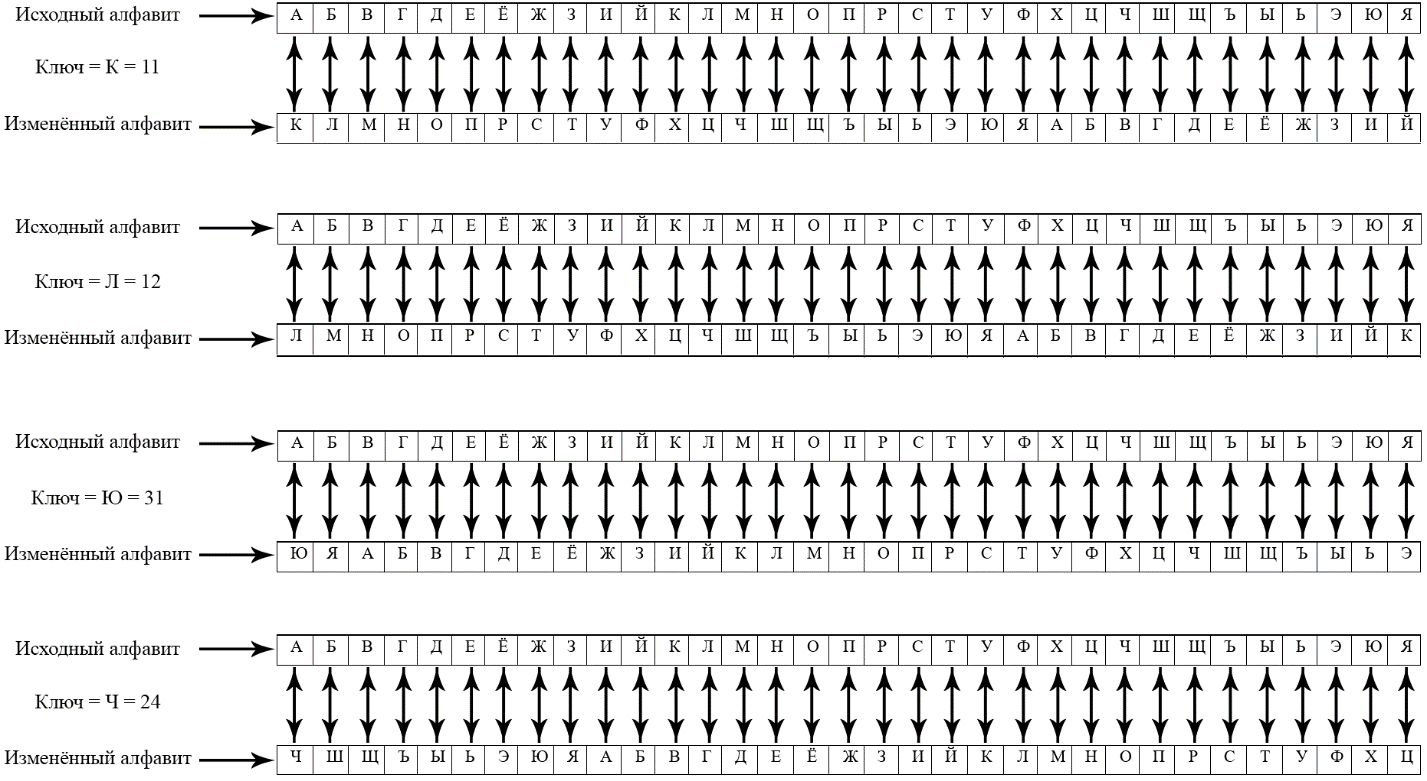


Рисунок 1.4 – Моноалфавиты полученные при использовании ключевого слова «КЛЮЧ»

Теперь, когда у нас алфавиты, изменённые под выбранный ключ можно приступить к шифрованию. Зашифруем сообщение «ПОЛИАЛФАВИТНЫЙ ШИФР». Для этого первую буква открытого текста будем шифровать через моноалфавит буквы «К», вторую через моноалфавит буквы «Л», третью через моноалфавит буквы «Ю», четвёртую через моноалфавит буквы «Ч». Для последующих букв требуется повторять цикл до тех пор, пока весь открытый текст не будет зашифрован. В результате шифрования мы получим «ЪЪЙАКЧТЧМФРЕЁХ ПУАО» (рисунок 1.5).

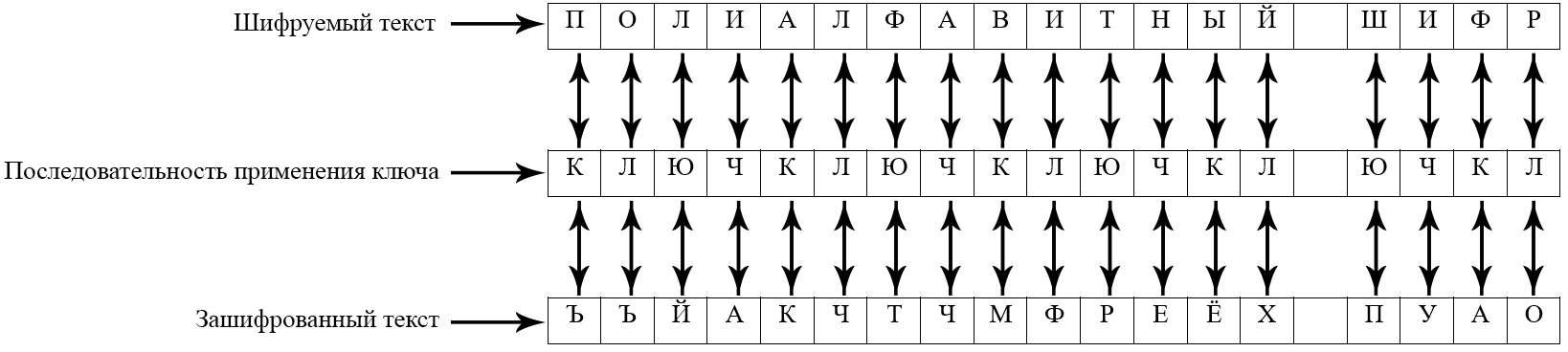


Рисунок 1.5 – Развёртка, как шифруется каждая буква открытого текста

Дешифрование соответственно происходит в обратном порядке.

Обычно, если текст хотят зашифровать или расшифровать этим методом используют таблицу под названием «квадрат Виженера» (рисунок 1.6).

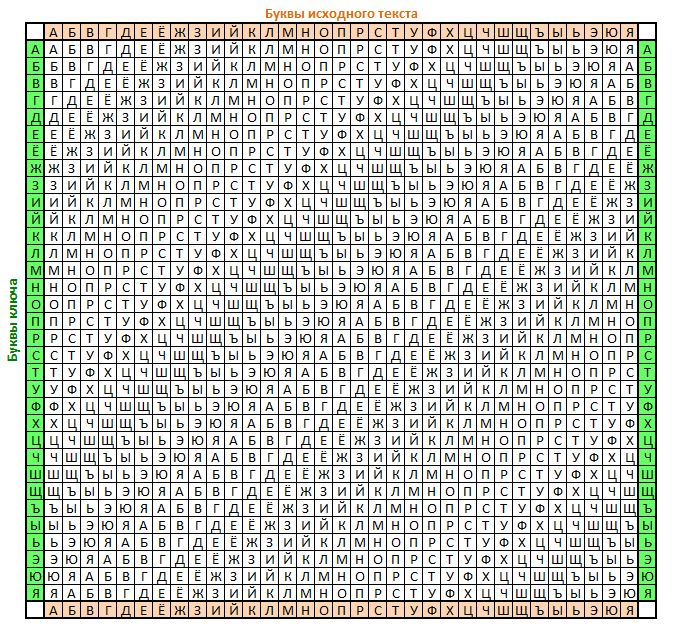


Рисунок 1.6 – Квадрат Виженера для русского алфавита

Пользоваться ей довольно просто, по вертикали выбирается буква ключа, по горизонтали буква текста, на их пересечении и будет шифруемая буква.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также маскировку частот появления тех или иных букв в тексте.

Минусами этого метода можно считать передачу больших объёмов текста, а также распространение ключей и их обновление.

#### Исключающее ИЛИ (XOR)

Метод исключающего ИЛИ (XOR) относится к симметричным криптосистемам типа гаммирование. Принцип работы этого типа шифров заключается в «наложении» последовательности, состоящей из случайных чисел на открытый текст. То есть генератор случайных чисел выдаёт последовательность битов (гамму), которая накладывается на открытый текст с помощью побитовой операции исключающего ИЛИ, в результате чего получается шифротекст.

Метод является достаточно простым. Давайте представим русский алфавит (для упрощения без буквы «Ё») в двоичном виде (таблица 1.1).

Таблица 1.1 – Русский алфавит в двоичном представлении

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | Код | Символ | Код | Символ | Код | Символ | Код |
| А | 00000 | И | 01000 | Р | 10000 | Ш | 11000 |
| Б | 00001 | Й | 01001 | С | 10001 | Щ | 11001 |
| В | 00010 | К | 01010 | Т | 10010 | Ъ | 11010 |
| Г | 00011 | Л | 01011 | У | 10011 | Ы | 11011 |
| Д | 00100 | М | 01100 | Ф | 10100 | Ь | 11100 |
| Е | 00101 | Н | 01101 | Х | 10101 | Э | 11101 |
| Ж | 00110 | О | 01110 | Ц | 10110 | Ю | 11110 |
| З | 00111 | П | 01111 | Ч | 10111 | Я | 11111 |

Как мы знаем из булевой алгебры, операция логического сложения имеет следующую семантику (рисунок 1.7).

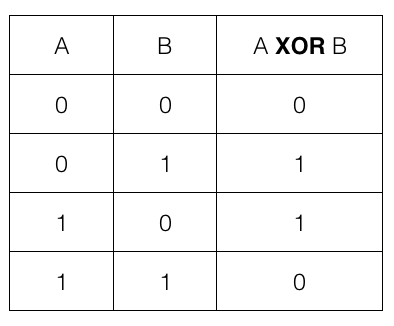


Рисунок 1.7 – Таблица истинности для операции исключающего ИЛИ (XOR)

Соответственно, если мы к примеру, захотим зашифровать букву «Х» (10101) используя в качестве ключа букву «У» (01011) результатом операции логического сложения будет буква «Ю» (11110). Для дешифровки нужно проделать те же самые действия, с тем же самым ключом, но уже над буквой «Ю». Отсюда вытекает свойство обратимости результата.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также его стойкость, определяющаяся гаммой (длительностью периода и равномерностью статических характеристик).

Минусами являются: распространение ключей и их обновление.

#### Одноразовый блокнот

Шифр Вернама или одноразовый блокнот – представляет собой систему симметричного шифрования типа гаммирование, так как использует булеву функцию «исключающее ИЛИ». Этот метод был изобретён в 1917 году Гилбертом Вернамом. При правильном использовании этого метода, текст, который был им зашифрован невозможно взломать. Этот метод является примером системы с абсолютной криптографической стойкостью при этом считаясь одной из простейших криптосистем.

Так как работает этот метод основываясь на XOR, алгоритм шифрования такой же, как был описан выше. Единственная разница в том, что длина ключа обязательно должна быть равна длине открытого текста.

Плюсами этого метода можно считать высокую скорость шифрования и дешифрования, а также его абсолютную криптографическую стойкость при правильном использовании.

Минусами являются: существенный размер ключа, а также распространение ключей и их регулярное обновление (настолько регулярное, что ни один ключ не должен использоваться более одного раза).

#### Rivest, Shamir, Adleman (RSA)

Последний метод, который мы рассмотрим называется RSA (аббревиатура от фамилий его создателей Rivest, Shamir, Adleman). Он относится к асимметричным криптосистемам.

Эта криптосистема стала первой системой, пригодной как для шифрования, так и для цифровой подписи. Принцип её работы можно разделить на три шага: первый – создание открытого (публичного) и закрытого (секретного) ключей на основе взаимно простых чисел (тех, которые делятся только на единицу или сами на себя), второй шаг – шифрование сообщения и третий шаг – расшифровка.

Как же создать эти самые ключи? Для создания открытого ключа нужно: выбрать два простых числа, вычислить модуль их произведения, вычислить функцию Эйлера, выбрать открытую экспоненту, которая также будет являться простым числом, при этом будет меньше числа, полученного при вычислении функции Эйлера и наконец будет взаимно простым для этой функции. В результате этих манипуляций с формулами мы получим пару чисел, это и есть наш публичный ключ. Закрытый ключ создаётся вычислением обратной открытой экспоненты по модулю функции Эйлера, при этом модуль должен быть равен единице.

Теперь нужно отдать полученный открытый ключ человеку, который с помощью него планирует отправлять вам зашифрованные сообщения. Допустим, что, выполнив все предыдущие операции мы получили числа 5 и 21 – это публичный ключ и числа 17 и 21 – это секретный ключ. Теперь допустим, что вы хотите зашифровать букву «С», в русском алфавите она располагается на 19 позиции. Для того чтобы зашифровать эту букву нужно возвести позицию нашей буквы в 5 степень и от полученного числа взять остаток от деления на 21. В результате получиться число 10 или буква «И» это и будут наши закодированные данные.

Шифрованные данные передаются владельцу секретного ключа. Тут следует обратить внимание на то, что открытый ключ не может расшифровать сообщение, а закрытый находится только у владельца (если он никому его не говорил), так что передача может осуществляться по открытому каналу. Итак, чтобы дешифровать сообщение нужно провести те же действия, что и при шифровании сообщения только используя закрытый ключ. Возьмём нашу зашифрованную букву «И» расположенную на позиции 10, возведём её в 17 степень и от полученного числа возьмём остаток от деления на 21. Результатом вычислений будет число 19 или буква «С».

Подводя итоги, если необходимо передать зашифрованное сообщение владельцу ключей, то отправитель должен получить у него открытый ключ, зашифровать своё сообщение и передать его владельцу. При этом расшифровать сообщение не может никто, кроме владельца секретного ключа. Весь этот процесс представлен на рисунке 1.8.

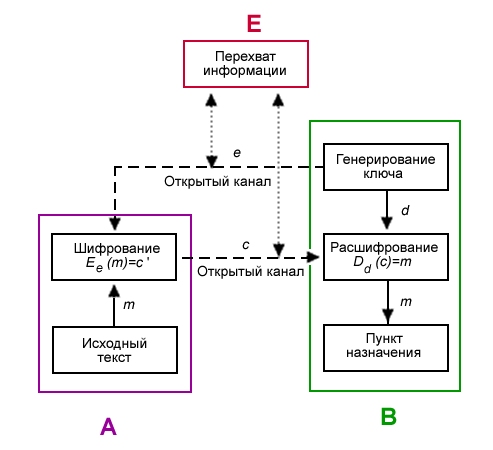


Рисунок 1.8 – Процесс шифрования при помощи алгоритмы RSA

Надёжность такого шифрования обеспечивается тем, что третьему лицу очень трудно вычислить закрытый ключ по открытому. Хотя ключи вычисляются из одной пары чисел и по сути связаны между собой, установить их связь очень сложно, а чем большие числа были взяты, тем длиннее становится процесс вскрытия.

Плюсами этого метода можно считать удобство распространения ключей и высокую безопасность при передаче небольших сообщений.

Минусами являются: существенный размер ключа, низкая скорость шифрования, а также то, что шифрование происходит по буквам, то есть одна и та же буква будет шифроваться одним и тем же числом, если злоумышленник перехватит достаточно большое сообщение расшифровать его не составит никакого труда.

## Характеристика инструментальных средств разработки

### Характеристика среды программирования

Для написания программы использовалась среда программирования Microsoft Visual Studio 2017, представляющая собой полный набор средств разработки для создания веб-приложений Active Server Pages (ASP), eXtensible Markup Language (XML), настольных приложений и мобильных приложений. Visual Studio использует единую интегрированную среду разработки Integrated Development Environment (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.

### Описание системы Windows Presentation Foundation (WPF)

В качестве системы для построения клиентских приложений была выбрана 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 eXtensible Application Markup Language (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.

### Описание языка программирования

В качестве языка программирования был выбран язык С# 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 c богатством возможностей и мощью C++.

Особенности С#:

* 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 Application Programming Interface;
* 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 Visual Basic;
* 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 С# для написания динамических web-страниц ASP.NET.

В настоящее время C# является одним из самых популярных языков программирования. Он применяется в разработке компьютерных, мобильных и веб-приложений. Также язык регулярно обновляется и имеет огромное количество документации, которая сильно упрощает и ускоряет процесс разработки программ на нём.

### Другие средства разработки

Для создания контекстной справки использовалась программа «Htm2chm». Она представляет собой [проприетарный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [формат файлов](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2) контекстной справки, разработанный корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) и выпущенный в [1997 году](https://ru.wikipedia.org/wiki/1997_%D0%B3%D0%BE%D0%B4) в качестве замены формата [WinHelp](https://ru.wikipedia.org/wiki/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](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA) по содержимому страниц.

Преимуществами программы можно считать:

* размер файла меньше, чем у обычного [HTML](https://ru.wikipedia.org/wiki/HTML);
* используются все возможности форматирования, имеющиеся в HTML и [CSS](https://ru.wikipedia.org/wiki/Cascading_Style_Sheets);
* возможность [полнотекстового поиска](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA);
* 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 и синхронизация полученных документов с Google Диском.

# Практическая часть

## Постановка задачи

### Описание входных данных

Входные данные – данные, вводимые или выбранные пользователем в ходе работы программы. К ним относится: выбор операции, выбор способа шифрования или дешифрования, ввод исходного текста, ввод ключей, ввод значений для генерации ключей и файлы, загружаемые в программу.

Подробнее входные данные описаны в таблице 2.1.

Таблица 2.1 – Входные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Идентификатор | Тип данных | Размер |
| Выбор операции | IsChecked | Bool | 1 байт |
| Выбор способа шифрования или дешифрования | SelectedIndex | Int32 | 4 байта |
| Исходный текст введённый пользователем вручную или считанный из файла | soursetext  openFileDlg | String  String | 255 байт  255 байт |
| Ключ введённый пользователем вручную или считанный из файла | shift  key  \_key | String  Int32  Int32 | 255 байт  4 байта  4 байта |
| Размер ключа | sizesourcetext | Int32 | 4 байта |
| Значение P | p | Int32 | 4 байта |
| Значение Q | q | Int32 | 4 байта |

### Описание выходных данных

Выходные данные – данные, полученные в ходе работы пользователя с программой. К ним относится: сгенерированные пользователем ключи, зашифрованный или дешифрованный текст.

Подробнее выходные данные описаны в таблице 2.2.

Таблица 2.2 – Выходные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Идентификатор | Тип данных | Размер |
| Сгенерированные пользователем ключи | shift  code  e  d  n | Int32  String  Long (Int64)  Long (Int64)  Long (Int64) | 4 байта  255 байт  8 байт  8 байт  8 байт |
| Зашифрованный или дешифрованный текст | EncryptedData | String | 255 байт |
| Сохраняемые изменения в тексте, результаты шифрования или дешифрования и ключи | saveFileDlg | String | 255 байт |

### Математическая модель задачи

В программе реализовано шесть алгоритмов шифрования и их дешифровки, которые работают по некоторым формулам.

Начнём с шифра транспозиции. Перед шифрованием текста нужно разбить его на блоки равные длине ключа. Для этого используется следующая формула (2.1):

(2.1) []

где i – индекс суммирования, n – верхняя граница суммирования равная остатку от деления текста на размер ключа, – переменная, обозначающая каждый член в серии.

Далее требуется выполнить перестановку символов, для этого используется следующая формула (2.2):

(2.2) []

где i – позиция символа в исходном тексте, j – позиция символа в блоке, m – ключ, – новая позиция символа, – позиция символа.

При дешифровании этим способом используется похожий алгоритм за исключением того, что текст не нужно увеличивать до длины, которая будет ровно делиться на длину ключа, так как он уже подходящего размера. Следовательно, остаётся только переставить символы по следующей формуле (2.3):

(2.3) []

где i – позиция символа в исходном тексте, j – позиция символа в блоке, m – ключ, – новая позиция символа, – позиция символа.

Следующими алгоритмами рассмотрим моноалфавитный и полиалфавитный шифры. Они работают по одинаковым формулам. Если сопоставить каждому символу алфавита его порядковый номер, то шифрование и дешифрование можно выразить формулами 2.4:

(2.4) []

где x – символ открытого текста, y – символ шифрованного текста, n – мощность алфавита (длина), k – ключ.

В шифрах исключающего ИЛИ и Вернама для шифрования и дешифрования текстов используют следующую формулу (2.5):

(2.5) []

где x – символ открытого или закрытого текста, j – номер символа в алфавите, k – номер символа ключа, l – мощность алфавита (длина).

Для шифрования алгоритмом RSA нужно создать открытый и закрытый ключи, для этого требуется найти модуль произведения простых чисел по формуле (2.6):

(2.6) []

где n – модуль произведения простых чисел, p, q – простые, отличные друг от друга числа.

Далее следует вычислить функцию Эйлера (2.7):

(2.7) []

где – значение функции Эйлера, n – модуль произведения простых чисел, p, q – простые, отличные друг от друга числа.

Далее ищем значение открытой экспоненты по следующим критериям (2.8):

(2.8) []

где – значение функции Эйлера, e – открытая экспонента.

Значения {e, n} – открытый ключ.

Чтобы вычислить значение для закрытого ключа, нужно подобрать такое число, которое было бы мультипликативно обратно к открытой экспоненте по модулю функции Эйлера, то есть число, удовлетворяющее сравнению. Оно находится по следующей формуле (2.9):

(2.9) []

где d – закрытая экспонента, e – открытая экспонента.

С ключами разобрались, теперь разберёмся как шифровать текст. Для этого нужно взять открытый ключ и воспользоваться следующей формулой (2.10):

(2.10) []

где c – закрытый текст, e – открытая экспонента, m – открытый текст, n – модуль произведения простых чисел.

Дешифруется текст с использованием закрытого ключа по следующей формуле (2.11):

(2.11) []

где c – закрытый текст, d – закрытая экспонента, m – открытый текст, n – модуль произведения простых чисел.

### Требования к программному обеспечению

#### Функциональные требования

Программа должна выполнять следующие функции:

* шифровать информацию с помощью выбранного алгоритма;
* дешифровать информацию с помощью выбранного алгоритма;
* загружать в программу уже существующие файлы;
* сохранять информацию, полученную в ходе работы программы;
* генерировать ключи для шифрования и дешифрования;
* возможность увеличивать размер шрифта;
* возможность уменьшать размер шрифта;
* возможность редактировать текст прямо из программы;
* возможность очищать поля от текста в одно нажатие;
* возможность выполнять необходимые операции посредством нажатия определённых клавиш на клавиатуре;
* выводить подсказки в ходе работы;
* сообщать об ошибках, возникающих в ходе неправильной работы;
* возможность посмотреть подробную информацию о программе посредством вызова справки;
* запрашивать у пользователя подтверждение для осуществления некоторых действий;
* работать в соответствии с заданным алгоритмом;
* производить бесперебойную работу по преобразованию информации.

Помимо этого, должно быть обеспечено чёткое и правильное выполнение шифрования и дешифрования текста в соответствии со всеми формулами и уравнениями, используемыми в программе.

#### Нефункциональные требования

Разрабатываемый интерфейс должен:

* быть интуитивно понятным пользователю;
* быть выполнен в нейтральных цветах, а также в минималистическом стиле и при этом привлекательно смотреться;
* обеспечивать видимость всех элементов управления, которые необходимы для выполнения конкретной задачи;
* предоставлять пользователю помощь в освоении программы посредством подсказок;
* обладать ненавязчивостью в своём стремлении помочь, то есть не заставлять пользователя совершать лишних действий;
* минимизировать возможность ошибок пользователя.

В программе должна быть предусмотрена защита от таких действий пользователя как:

* ввод неверных ключей;
* ввод неверных параметров при генерации ключа;
* попытки выполнить операцию шифрования или дешифрования без выбранного алгоритма;
* попытка сохранить ключ при его генерации до того, как он будет сгенерирован;
* попытка перезаписать исходный текст в файл, для которого не указан путь;
* попытка открытия справки «О программе», если файл был повреждён или удалён.

Также программа должна быть реализована на языке С# версии 7.0. NET Framework 4.7.1. в системе для построения клиентских приложений WPF.

## Анализ требований и определение спецификаций программного обеспечения

### Диаграмма вариантов использования

Диаграмма вариантов использования представлена на рисунке 2.1. Она описывает взаимоотношения и зависимости между группами вариантов использования и пользователем.

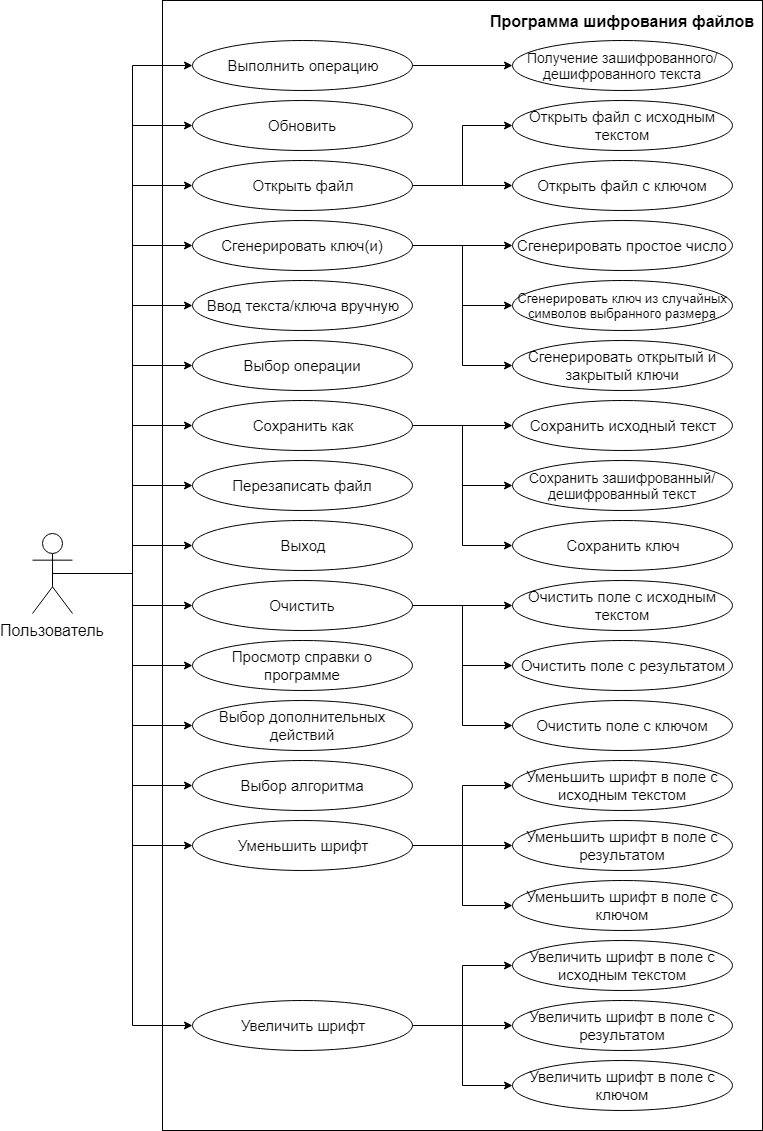


Рисунок 2.1 – Диаграмма вариантов использования

### Диаграмма потоков данных

Контекстная диаграмма потоков данных представлена на рисунке 2.2. Она представляет собой взаимодействие пользователя и программы, связанных потоками данных.

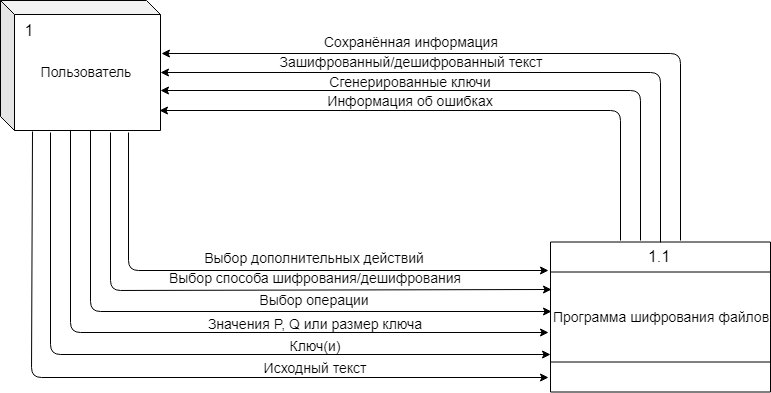


Рисунок 2.2 – Контекстная диаграмма потоков данных

### Функциональная диаграмма

Функциональная диаграмма – диаграмма, отражающая взаимосвязи функций разрабатываемого программного обеспечения (ПО). Контекстная функциональная диаграмма представлена на рисунке 2.3.

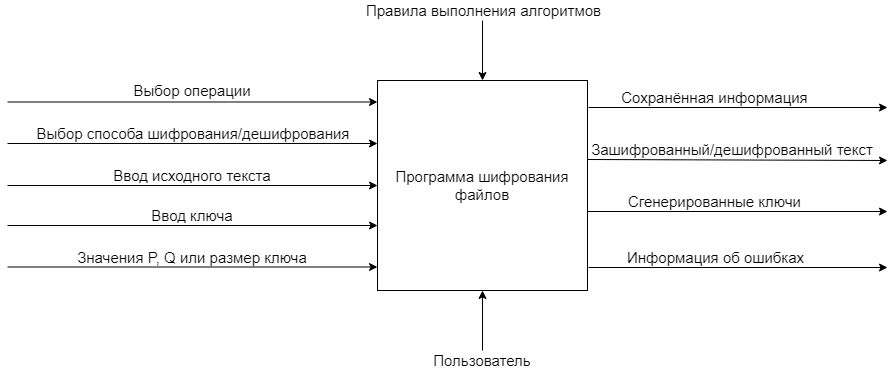


Рисунок 2.3 – Контекстная функциональная диаграмма

## Проектирование программного обеспечения

### Структурная схема

Структурная схема представляет собой совокупность элементарных звеньев объекта и связей между ними. Под элементарным звеном следует понимать часть объекта, системы управления, которая реализует элементарную функцию. Структурная схема программы представлена на рисунке 2.4.

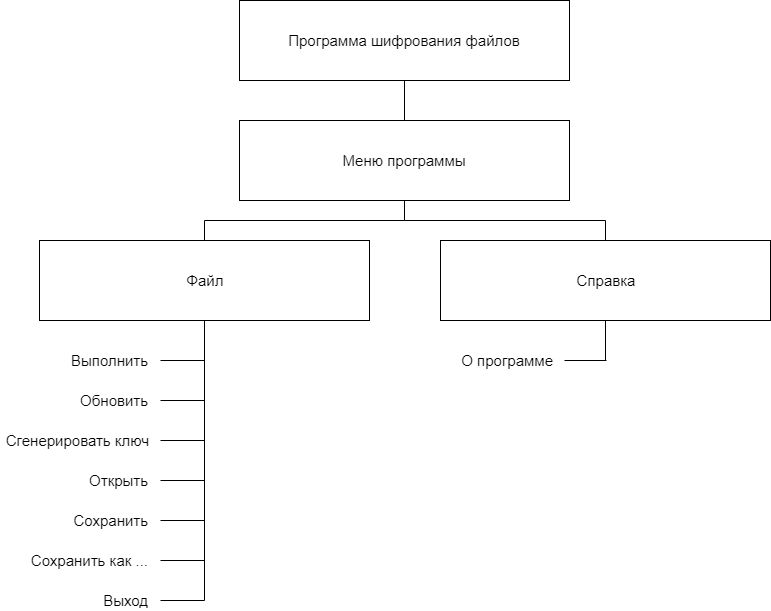


Рисунок 2.4 – Структурная схема

### Функциональная схема

Функциональная схема приведена на рисунке 2.5. Она разъясняет процессы, протекающие в отдельных функциях программы и программы в целом.

Рисунок 2.5 – Функциональная схема

## Разработка пользовательских интерфейсов

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

## 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.

## Руководство по использованию программы

### Руководство системного программиста

#### 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.

#### 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.

#### Настройка программы

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

#### Проверка программы

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

#### Дополнительные возможности

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

#### 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.

### Руководство программиста

#### 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).

#### Характеристики программы

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

#### Обращение к программе

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

#### Входные и выходные данные

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

#### 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.

### Руководство пользователя

#### Назначение программы

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

#### Условия выполнения программы

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

#### 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.

#### 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

Схемы и диаграммы

# Приложение Б

Исходный код программы

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Diagnostics;

namespace CryptographicApplication

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

#region Переменные

Functional func\_obj;

RandomKeyGeneration rndkey\_obj;

Transposition trans\_obj;

Monoalphabetic mono\_obj;

Polyalphabetic poly\_obj;

XOR xor\_obj;

Vernam vernam\_obj;

RSA rsa\_obj;

public bool transition = false;

#endregion

public MainWindow()

{

InitializeComponent();

rb\_Encryption.IsChecked = true;

func\_obj = new Functional();

trans\_obj = new Transposition();

mono\_obj = new Monoalphabetic();

poly\_obj = new Polyalphabetic();

xor\_obj = new XOR();

vernam\_obj = new Vernam();

rsa\_obj = new RSA();

rndkey\_obj = new RandomKeyGeneration();

}

#region Алгоритмы

public void Transposition\_Cipher()

{

trans\_obj.SetKey(tb\_Key.Text);

if (rb\_Encryption.IsChecked == true)

{

tb\_EncryptedData.Text = trans\_obj.Encrypt(tb\_SourceData.Text);

}

else

{

tb\_EncryptedData.Text = trans\_obj.Decrypt(tb\_SourceData.Text);

}

}

public void Monoalphabetic\_Cipher()

{

if (rb\_Encryption.IsChecked == true)

{

tb\_EncryptedData.Text = mono\_obj.Encrypt(tb\_SourceData.Text, Convert.ToInt32(tb\_Key.Text));

}

else

{

tb\_EncryptedData.Text = mono\_obj.Decrypt(tb\_SourceData.Text, Convert.ToInt32(tb\_Key.Text));

}

}

public void Polyalphabetic\_Cipher()

{

if (rb\_Encryption.IsChecked == true)

{

tb\_EncryptedData.Text = poly\_obj.Encrypt(tb\_SourceData.Text, tb\_Key.Text);

}

else

{

tb\_EncryptedData.Text = poly\_obj.Decrypt(tb\_SourceData.Text, tb\_Key.Text);

}

}

public void Vernam\_Cipher()

{

tb\_EncryptedData.Text = vernam\_obj.Encrypt\_and\_Decrypt(tb\_SourceData.Text, tb\_Key.Text);

}

public void XOR\_Cipher()

{

tb\_EncryptedData.Text = xor\_obj.Encrypt\_and\_Decrypt(tb\_SourceData.Text, tb\_Key.Text);

}

public void RSA\_Cipher()

{

rsa\_obj.SetKey(tb\_Key.Text);

if (rb\_Encryption.IsChecked == true)

{

tb\_EncryptedData.Text = rsa\_obj.Encrypt(tb\_SourceData.Text);

}

else

{

String[] s = tb\_SourceData.Text.Split(new String[] { "\n" }, StringSplitOptions.RemoveEmptyEntries);

List<string> sourcetext = new List<string>(s);

tb\_EncryptedData.Text = rsa\_obj.Decrypt(sourcetext);

}

}

#endregion

#region Элементы формы

#region Меню

private void Menu\_btn\_ExecuteOperation\_Click(object sender, RoutedEventArgs e)

{

List\_of\_Operations\_to\_Perform();

}

private void Menu\_btn\_Refresh\_Click(object sender, RoutedEventArgs e)

{

Refresh();

}

private void Menu\_btn\_Key\_Generation\_Click(object sender, RoutedEventArgs e)

{

List\_of\_Key\_Generation\_Methods();

}

private void Menu\_btn\_FileSelection\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.File\_Selection(tb\_FileName\_Source, tb\_SourceData, tb\_EncryptedData);

}

private void Menu\_btn\_FileSelection\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.File\_Selection(tb\_Key);

}

private void Menu\_btn\_SaveFile\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File(tb\_FileName\_Source, tb\_SourceData);

}

private void Menu\_btn\_SaveFileAs\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(tb\_FileName\_Source, tb\_SourceData);

}

private void Menu\_btn\_SaveFileAs\_Encrypted\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_EncryptedData, rb\_Encryption, true);

}

private void Menu\_btn\_SaveFileAs\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Key, rb\_Encryption, false);

}

private void Menu\_btn\_Exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Menu\_btn\_About\_the\_Program\_Click(object sender, RoutedEventArgs e)

{

Calling\_Help();

}

#endregion

#region Исходные данные

private void Btn\_FileSelection\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.File\_Selection(tb\_FileName\_Source, tb\_SourceData, tb\_EncryptedData);

}

private void Btn\_SaveFile\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File(tb\_FileName\_Source, tb\_SourceData);

}

private void Btn\_SaveFileAs\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(tb\_FileName\_Source, tb\_SourceData);

}

private void Tb\_FileName\_Source\_TextChanged(object sender, TextChangedEventArgs e)

{

func\_obj.Changed\_File\_Name\_TB(tb\_FileName\_Source, btn\_SaveFile\_Source, menu\_btn\_SaveFile);

}

private void Btn\_Clear\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Clear\_tb(tb\_SourceData);

}

private void Btn\_Increase\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_SourceData, true);

}

private void Btn\_Reduce\_Source\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_SourceData, false);

}

private void Tb\_SourceData\_TextChanged(object sender, TextChangedEventArgs e)

{

tb\_EncryptedData.Text = "";

if (tb\_SourceData.Text == "")

{

tb\_SourceData.ToolTip = "Введите текст";

}

else

{

tb\_SourceData.ToolTip = "Исходный текст";

}

}

#endregion

#region Результат

private void Btn\_SaveFileAs\_Encrypted\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_EncryptedData, rb\_Encryption, true);

}

private void Btn\_Clear\_Encrypted\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Clear\_tb(tb\_EncryptedData);

}

private void Btn\_Increase\_Encrypted\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_EncryptedData, true);

}

private void Btn\_Reduce\_Encrypted\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_EncryptedData, false);

}

private void Tb\_EncryptedData\_TextChanged(object sender, TextChangedEventArgs e)

{

if (tb\_EncryptedData.Text == "")

{

tb\_EncryptedData.ToolTip = "Выполните операцию";

}

else

{

tb\_EncryptedData.ToolTip = "Результат выполнения операции";

}

}

#endregion

#region Ключ

#region Основное меню

private void Btn\_FileSelection\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.File\_Selection(tb\_Key);

}

private void Btn\_SaveFileAs\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Key, rb\_Encryption, false);

}

private void Btn\_Clear\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Clear\_tb(tb\_Key);

}

private void Btn\_Increase\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_Key, true);

}

private void Btn\_Reduce\_Key\_Click(object sender, RoutedEventArgs e)

{

func\_obj.Font\_Size(tb\_Key, false);

}

private void Tb\_Key\_TextChanged(object sender, TextChangedEventArgs e)

{

if (tb\_Key.Text == "")

{

tb\_Key.ToolTip = "Введите ключ";

}

else

{

tb\_Key.ToolTip = "Ваш ключ";

}

tb\_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

#endregion

#region Меню для генерации ключей с выбором размера

private void Btn\_Key\_Generation\_1\_Click(object sender, RoutedEventArgs e)

{

Key\_Generation\_Selection();

}

private void Btn\_Backward\_Click(object sender, RoutedEventArgs e) //назад

{

transition = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

menu\_btn\_SaveFileAs\_Key.IsEnabled = true;

tb\_Key\_Size.Text = "";

tb\_Key\_1.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Size, 0);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_1, 1);

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Collapsed;

}

private void Btn\_OK\_Click(object sender, RoutedEventArgs e) //ок

{

if (func\_obj.Confirm\_Action(0))

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Key\_1, rb\_Encryption, false);

}

transition = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

menu\_btn\_SaveFileAs\_Key.IsEnabled = true;

tb\_Key.Text = tb\_Key\_1.Text;

tb\_Key\_Size.Text = "";

tb\_Key\_1.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Size, 0);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_1, 1);

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Collapsed;

}

private void Tb\_Key\_Size\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Key\_Size);

}

private void Tb\_Key\_Size\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Size, 0);

}

private void Tb\_Key\_Size\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Key\_Size, e);

}

private void Tb\_Key\_Size\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

func\_obj.Only\_Number(e);

}

private void Tb\_Key\_Size\_TextChanged(object sender, TextChangedEventArgs e)

{

tb\_Key\_Size.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

private void Tb\_Key\_1\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Key\_1);

}

private void Tb\_Key\_1\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Key\_1, 1);

}

private void Tb\_Key\_1\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Key\_1, e);

}

private void Tb\_Key\_1\_TextChanged(object sender, TextChangedEventArgs e)

{

try

{

if (tb\_Key\_1.Text == "Ваш ключ" || tb\_Key\_1.Text == "")

{

btn\_OK.IsEnabled = false;

}

else

{

btn\_OK.IsEnabled = true;

}

}

catch (Exception)

{

}

}

#endregion

#region Меню для генерации ключей RSA

private void Btn\_Key\_Generation\_2\_Click(object sender, RoutedEventArgs e)

{

Key\_Generation\_Selection();

}

private void Btn\_Backward\_1\_Click(object sender, RoutedEventArgs e)

{

transition = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

menu\_btn\_SaveFileAs\_Key.IsEnabled = true;

tb\_Key\_P.Text = "";

tb\_Key\_Q.Text = "";

tb\_Public\_Key.Text = "";

tb\_Privat\_Key.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_P, 2);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Q, 3);

func\_obj.Сheck\_the\_Сursor(tb\_Public\_Key, 4);

func\_obj.Сheck\_the\_Сursor(tb\_Privat\_Key, 5);

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Collapsed;

}

private void Btn\_OK\_1\_Click(object sender, RoutedEventArgs e)

{

if (func\_obj.Confirm\_Action(1))

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Public\_Key, true);

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Privat\_Key, false);

}

transition = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

menu\_btn\_SaveFileAs\_Key.IsEnabled = true;

if (rb\_Encryption.IsChecked == true)

{

tb\_Key.Text = tb\_Public\_Key.Text;

}

else

{

tb\_Key.Text = tb\_Privat\_Key.Text;

}

tb\_Key\_P.Text = "";

tb\_Key\_Q.Text = "";

tb\_Public\_Key.Text = "";

tb\_Privat\_Key.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_P, 2);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Q, 3);

func\_obj.Сheck\_the\_Сursor(tb\_Public\_Key, 4);

func\_obj.Сheck\_the\_Сursor(tb\_Privat\_Key, 5);

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Collapsed;

}

private void Tb\_Key\_P\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

func\_obj.Only\_Number(e);

}

private void Tb\_Key\_P\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Key\_P);

}

private void Tb\_Key\_P\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Key\_P, e);

}

private void Tb\_Key\_P\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Key\_P, 2);

}

private void Tb\_Key\_P\_TextChanged(object sender, TextChangedEventArgs e)

{

tb\_Key\_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

private void Tb\_Key\_Q\_TextChanged(object sender, TextChangedEventArgs e)

{

tb\_Key\_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

}

private void Tb\_Key\_Q\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Q, 3);

}

private void Tb\_Key\_Q\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Key\_Q, e);

}

private void Tb\_Key\_Q\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Key\_Q);

}

private void Tb\_Key\_Q\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

func\_obj.Only\_Number(e);

}

//открытый

private void Tb\_Public\_Key\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Public\_Key, 4);

}

private void Tb\_Public\_Key\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Public\_Key, e);

}

private void Tb\_Public\_Key\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Public\_Key);

}

private void Tb\_Public\_Key\_TextChanged(object sender, TextChangedEventArgs e)

{

Unlock\_Btn();

}

//закрытый

private void Tb\_Privat\_Key\_LostFocus(object sender, RoutedEventArgs e)

{

func\_obj.Сheck\_the\_Сursor(tb\_Privat\_Key, 5);

}

private void Tb\_Privat\_Key\_PreviewKeyDown(object sender, KeyEventArgs e)

{

func\_obj.Without\_a\_Space(tb\_Privat\_Key, e);

}

private void Tb\_Privat\_Key\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

func\_obj.Сheck\_for\_Text(tb\_Privat\_Key);

}

private void Tb\_Privat\_Key\_TextChanged(object sender, TextChangedEventArgs e)

{

Unlock\_Btn();

}

#endregion

#endregion

#region Прочее

public void List\_of\_Operations\_to\_Perform() //меню, определяющее какой метод выполнять

{

try

{

tb\_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

switch (cb\_Algorithms.SelectedIndex)

{

case -1: cb\_Algorithms\_Border.Background = Brushes.Red; break;

case 0: Transposition\_Cipher(); break;

case 1: Monoalphabetic\_Cipher(); break;

case 2: Polyalphabetic\_Cipher(); break;

case 3: XOR\_Cipher(); break;

case 4: Vernam\_Cipher(); break;

case 5: RSA\_Cipher(); break;

}

Additional\_Func();

//tb\_Key.ToolTip = "Ваш ключ";

}

catch (Exception)

{

tb\_Key.ToolTip = "Неверный ключ";

tb\_Key.BorderBrush = Brushes.Red;

}

}

public void Displaying\_the\_Key\_Generation\_Menu(bool a) //переходы

{

if (a == true)

{

transition = true;

menu\_btn\_SaveFileAs\_Key.IsEnabled = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Collapsed;

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Visible;

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Collapsed;

}

else

{

transition = true;

menu\_btn\_SaveFileAs\_Key.IsEnabled = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Collapsed;

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Collapsed;

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Visible;

}

}

public void List\_of\_Key\_Generation\_Methods() //меню, определяющее для какого способа генерировать ключ

{

switch (cb\_Algorithms.SelectedIndex)

{

case 0:

if (transition == false)

{

Displaying\_the\_Key\_Generation\_Menu(true);

}

else

{

Key\_Generation\_Selection();

}

break;

case 1: Key\_Generation\_Selection(); break;

case 2:

if (transition == false)

{

Displaying\_the\_Key\_Generation\_Menu(true);

}

else

{

Key\_Generation\_Selection();

}

break;

case 3:

if (transition == false)

{

Displaying\_the\_Key\_Generation\_Menu(true);

}

else

{

Key\_Generation\_Selection();

}

break;

case 4: Key\_Generation\_Selection(); break;

case 5:

if (transition == false)

{

Displaying\_the\_Key\_Generation\_Menu(false);

}

else

{

Key\_Generation\_Selection();

}

break;

}

}

public void Key\_Generation\_Selection() //меню, для выбора генерации ключей

{

try

{

switch (cb\_Algorithms.SelectedIndex)

{

case 0:

tb\_Key\_1.Text = rndkey\_obj.Rand\_Key\_Generation\_Transposition(Convert.ToInt32(tb\_Key\_Size.Text));

tb\_Key\_1.Foreground = Brushes.Black;

break;

case 1: tb\_Key.Text = rndkey\_obj.Rand\_Key\_Generation(); break;

case 2:

tb\_Key\_1.Text = rndkey\_obj.Rand\_Key\_Generation(Convert.ToInt32(tb\_Key\_Size.Text));

tb\_Key\_1.Foreground = Brushes.Black;

break;

case 3:

tb\_Key\_1.Text = rndkey\_obj.Rand\_Key\_Generation(Convert.ToInt32(tb\_Key\_Size.Text));

tb\_Key\_1.Foreground = Brushes.Black;

break;

case 4: tb\_Key.Text = rndkey\_obj.Rand\_Key\_Generation(tb\_SourceData.Text.Length); break;

case 5:

int p = 0, q = 0, i = 0;

p = func\_obj.Check(p, tb\_Key\_P);

q = func\_obj.Check(q, tb\_Key\_Q);

if (p > 10 && p != q && rsa\_obj.IsTheNumberSimple(p))

{

i++;

}

else

{

tb\_Key\_P.BorderBrush = Brushes.Red;

}

if (q > 10 && q != p && rsa\_obj.IsTheNumberSimple(q))

{

i++;

}

else

{

tb\_Key\_Q.BorderBrush = Brushes.Red;

}

if (i == 2)

{

tb\_Key\_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

tb\_Key\_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

long n = p \* q;

long fi = (p - 1) \* (q - 1);

long e = rsa\_obj.Calculate\_e(fi);

long d = rsa\_obj.Calculate\_d(e, fi);

tb\_Public\_Key.Foreground = Brushes.Black;

tb\_Privat\_Key.Foreground = Brushes.Black;

tb\_Public\_Key.Text = Convert.ToString(e) + " " + Convert.ToString(n);

tb\_Privat\_Key.Text = Convert.ToString(d) + " " + Convert.ToString(n);

}

break;

}

}

catch (Exception)

{

if (cb\_Algorithms.SelectedIndex == 0 || cb\_Algorithms.SelectedIndex == 2 || cb\_Algorithms.SelectedIndex == 3)

{

tb\_Key\_Size.BorderBrush = Brushes.Red;

}

}

}

public void Calling\_Help()

{

try

{

string commandText = "Help.chm";

var proc = new Process();

proc.StartInfo.FileName = commandText;

proc.StartInfo.UseShellExecute = true;

proc.Start();

}

catch

{

MessageBox.Show("Файл справки не найден!", "Не найден файл");

}

}

public void Refresh() //обновить всё

{

transition = false;

//операция

rb\_Encryption.IsChecked = true;

//способ шифрования

cb\_Algorithms.SelectedIndex = -1;

cb\_Algorithms\_Border.Background = this.Background;

//исходные

tb\_FileName\_Source.Text = "";

tb\_SourceData.Text = "";

tb\_SourceData.FontSize = 14;

//результат

tb\_EncryptedData.Text = "";

tb\_EncryptedData.FontSize = 14;

//ключ

menu\_btn\_SaveFileAs\_Key.IsEnabled = true;

tb\_Key.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

tb\_Key.Text = "";

tb\_Key.FontSize = 14;

//Дополнительные действия

chb\_SaveKey.IsChecked = false;

chb\_SaveEncryptText.IsChecked = false;

chb\_SaveSourceText.IsChecked = false;

chb\_Refresh.IsChecked = false;

//генарация ключа с вводом размера

tb\_Key\_Size.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

tb\_Key\_Size.Text = "";

tb\_Key\_1.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Size, 0);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_1, 1);

//генарация ключа RSA

tb\_Key\_P.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

tb\_Key\_Q.BorderBrush = new SolidColorBrush(Color.FromRgb(171, 173, 179));

tb\_Key\_P.Text = "";

tb\_Key\_Q.Text = "";

tb\_Public\_Key.Text = "";

tb\_Privat\_Key.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_P, 2);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Q, 3);

func\_obj.Сheck\_the\_Сursor(tb\_Public\_Key, 4);

func\_obj.Сheck\_the\_Сursor(tb\_Privat\_Key, 5);

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Collapsed;

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Collapsed;

}

public void Additional\_Func()

{

if (chb\_SaveSourceText.IsChecked == true)

{

func\_obj.Save\_File\_As(tb\_FileName\_Source, tb\_SourceData);

}

if (chb\_SaveEncryptText.IsChecked == true)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_EncryptedData, rb\_Encryption, true);

}

if (chb\_SaveKey.IsChecked == true)

{

func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Key, rb\_Encryption, false);

}

if (chb\_Refresh.IsChecked == true)

{

Refresh();

}

}

public void Unlock\_Btn()

{

try

{

if (tb\_Public\_Key.Text == "Ваш открытый ключ" || tb\_Public\_Key.Text == ""

|| tb\_Privat\_Key.Text == "Ваш закрытый ключ" || tb\_Privat\_Key.Text == "")

{

btn\_OK\_1.IsEnabled = false;

}

else

{

btn\_OK\_1.IsEnabled = true;

}

}

catch (Exception)

{

}

}

private void Window\_KeyDown(object sender, KeyEventArgs e) //горячие клавиши

{

switch (e.Key)

{

case Key.F1: List\_of\_Operations\_to\_Perform(); break; //выполнить

case Key.F2: Refresh(); break; //обновить

case Key.F3: if (cb\_Algorithms.SelectedIndex != -1) List\_of\_Key\_Generation\_Methods(); break; //сгенерировать ключ

case Key.F4: func\_obj.File\_Selection(tb\_FileName\_Source, tb\_SourceData, tb\_EncryptedData); break; //открыть файл с исходным текстом

case Key.F5: func\_obj.File\_Selection(tb\_Key); break; //открыть файл с ключом

case Key.F6: if (tb\_FileName\_Source.Text != "") func\_obj.Save\_File(tb\_FileName\_Source, tb\_SourceData); break; //сохранить

case Key.F7: func\_obj.Save\_File\_As(tb\_FileName\_Source, tb\_SourceData); break; //сохранить исходный текст

case Key.F8: func\_obj.Save\_File\_As(cb\_Algorithms, tb\_EncryptedData, rb\_Encryption, true); break; //сохранить зашифрованный/дешифрованный текст

case Key.F9: if (Grid\_Main\_Key\_Menu.Visibility == Visibility.Visible) func\_obj.Save\_File\_As(cb\_Algorithms, tb\_Key, rb\_Encryption, false); break; //сохранить ключ

case Key.F11: Calling\_Help(); break; //о программе

case Key.F12: this.Close(); break; //выход

}

}

private void Cb\_Algorithms\_SelectionChanged(object sender, SelectionChangedEventArgs e) //изменения комбобокса

{

transition = false;

Grid\_Main\_Key\_Menu.Visibility = Visibility.Visible;

tb\_Key\_Size.Text = "";

tb\_Key\_1.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Size, 0);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_1, 1);

Grid\_Generation\_Key\_Menu\_1.Visibility = Visibility.Collapsed;

tb\_Key\_P.Text = "";

tb\_Key\_Q.Text = "";

tb\_Public\_Key.Text = "";

tb\_Privat\_Key.Text = "";

func\_obj.Сheck\_the\_Сursor(tb\_Key\_P, 2);

func\_obj.Сheck\_the\_Сursor(tb\_Key\_Q, 3);

func\_obj.Сheck\_the\_Сursor(tb\_Public\_Key, 4);

func\_obj.Сheck\_the\_Сursor(tb\_Privat\_Key, 5);

Grid\_Generation\_Key\_Menu\_2.Visibility = Visibility.Collapsed;

cb\_Algorithms\_Border.Background = this.Background;

menu\_btn\_SaveFileAs\_Key.IsEnabled = true; //разблокировать кнопку сохранение ключа в меню

func\_obj.Changed\_CB(cb\_Algorithms, btn\_Key\_Generation, menu\_btn\_Key\_Generation); //разблокирует или блокирует кнопку "Сгенерировать ключ"

}

private void Rb\_Encryption\_Checked(object sender, RoutedEventArgs e)

{

gb\_algs.Header = "Способ шифрования";

chb\_SaveEncryptText.Content = "Сохранить зашифрованный текст";

menu\_btn\_SaveFileAs\_Encrypted.Header = "Сохранить зашифрованный текст";

}

private void Rb\_Decryption\_Checked(object sender, RoutedEventArgs e)

{

gb\_algs.Header = "Способ дешифрования";

chb\_SaveEncryptText.Content = "Сохранить дешифрованный текст";

menu\_btn\_SaveFileAs\_Encrypted.Header = "Сохранить дешифрованный текст";

}

private void Btn\_ExecuteOperation\_Click(object sender, RoutedEventArgs e)

{

List\_of\_Operations\_to\_Perform();

}

private void Btn\_Key\_Generation\_Click(object sender, RoutedEventArgs e)

{

List\_of\_Key\_Generation\_Methods();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (MessageBox.Show("Вы действительно хотите выйти?", "Выйти?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

e.Cancel = false;

}

else

{

e.Cancel = true;

}

}

#endregion

#endregion

}

}

Alphabet.cs

using System;

namespace CryptographicApplication

{

class Alphabet

{

public char[] lang = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!\"#$%^&\*()+=-\_'?.,|/`~№:;@[]{}\\".ToCharArray();

}

}

Functional.cs

using System;

using System.Text;

using System.Windows.Controls;

using System.IO;

using Microsoft.Win32;

using System.Windows.Media;

using System.Windows.Input;

using System.Windows;

namespace CryptographicApplication

{

class Functional

{

public void Clear\_tb(TextBox a)

{

a.Text = "";

}

public void Font\_Size(TextBox a, bool b)

{

if (b == true)

{

if (a.FontSize < 20)

a.FontSize++;

}

else

{

if (a.FontSize > 1)

a.FontSize--;

}

}

public void File\_Selection(TextBox localFileText) // для ключа

{

OpenFileDialog openFileDlg = new OpenFileDialog();

openFileDlg.DefaultExt = ".txt";

openFileDlg.Filter = "Текстовый документ (.txt) | \* .txt";

openFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

Nullable<bool> result = openFileDlg.ShowDialog();

if (result == true)

{

localFileText.Text = File.ReadAllText(openFileDlg.FileName, Encoding.Default);

}

}

public void File\_Selection(TextBox localFileName, TextBox localFileText, TextBox clearFileText) // для исходного

{

OpenFileDialog openFileDlg = new OpenFileDialog();

openFileDlg.DefaultExt = ".txt";

openFileDlg.Filter = "Текстовый документ (.txt) | \* .txt";

openFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

Nullable<bool> result = openFileDlg.ShowDialog();

if (result == true)

{

localFileName.Text = openFileDlg.FileName;

localFileText.Text = File.ReadAllText(openFileDlg.FileName, Encoding.Default);

clearFileText.Text = "";

}

}

public void Save\_File(TextBox localFileName, TextBox TextToSave)

{

File.WriteAllText(localFileName.Text, TextToSave.Text, Encoding.Default);

}

public void Save\_File\_As(TextBox NewFileName, TextBox TextToSave) // для исходного

{

SaveFileDialog saveFileDlg = new SaveFileDialog();

saveFileDlg.DefaultExt = ".txt";

saveFileDlg.Filter = "Текстовый документ (.txt) | \* .txt";

saveFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

saveFileDlg.FileName = "Новый текстовый документ";

Nullable<bool> result = saveFileDlg.ShowDialog();

if (result == true)

{

NewFileName.Text = saveFileDlg.FileName;

File.WriteAllText(saveFileDlg.FileName, TextToSave.Text, Encoding.Default);

}

}

public void Save\_File\_As(ComboBox cb, TextBox TextToSave, RadioButton operation, bool a) // для ключа и зашифрованного

{

SaveFileDialog saveFileDlg = new SaveFileDialog();

saveFileDlg.DefaultExt = ".txt";

saveFileDlg.Filter = "Текстовый документ (.txt) | \* .txt";

saveFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

if (a == true)

{

if (operation.IsChecked == true)

{

saveFileDlg.FileName = "Зашифрованный текст (способ - " + cb.Text + ")";

}

else

{

saveFileDlg.FileName = "Дешифрованный текст (способ - " + cb.Text + ")";

}

}

else

{

saveFileDlg.FileName = "Ключ (способ - " + cb.Text + ")";

}

Nullable<bool> result = saveFileDlg.ShowDialog();

if (result == true)

{

File.WriteAllText(saveFileDlg.FileName, TextToSave.Text, Encoding.Default);

}

}

public void Save\_File\_As(ComboBox cb, TextBox TextToSave, bool a) // для открытого и закрытого ключей

{

SaveFileDialog saveFileDlg = new SaveFileDialog();

saveFileDlg.DefaultExt = ".txt";

saveFileDlg.Filter = "Текстовый документ (.txt) | \* .txt";

saveFileDlg.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

if (a == true)

{

saveFileDlg.FileName = "Открытый ключ (способ - " + cb.Text + ")";

}

else

{

saveFileDlg.FileName = "Закрытый ключ (способ - " + cb.Text + ")";

}

Nullable<bool> result = saveFileDlg.ShowDialog();

if (result == true)

{

File.WriteAllText(saveFileDlg.FileName, TextToSave.Text, Encoding.Default);

}

}

public void Changed\_File\_Name\_TB(TextBox ChangedTB, Button BtnUsed, MenuItem MIUsed)

{

if (ChangedTB.Text == "")

{

BtnUsed.IsEnabled = false;

MIUsed.IsEnabled = false;

}

else

{

BtnUsed.IsEnabled = true;

MIUsed.IsEnabled = true;

}

}

public void Changed\_CB(ComboBox ChangedCB, Button BtnUsed, MenuItem MIUsed)

{

if (ChangedCB.SelectedIndex == -1)

{

BtnUsed.IsEnabled = false;

MIUsed.IsEnabled = false;

}

else

{

BtnUsed.IsEnabled = true;

MIUsed.IsEnabled = true;

}

}

public void Сheck\_for\_Text(TextBox a)

{

if (a.Foreground != Brushes.Black)

{

a.Text = "";

a.Foreground = Brushes.Black;

}

}

public void Сheck\_the\_Сursor(TextBox a, int b)

{

if (a.Text == "")

{

a.Foreground = new SolidColorBrush(Color.FromRgb(171, 173, 179));

switch (b)

{

case 0: a.Text = "Введите размер ключа"; break;

case 1: a.Text = "Ваш ключ"; break;

case 2: a.Text = "Введите P"; break;

case 3: a.Text = "Введите Q"; break;

case 4: a.Text = "Ваш открытый ключ"; break;

case 5: a.Text = "Ваш закрытый ключ"; break;

}

}

}

public void Only\_Number(TextCompositionEventArgs e)

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

public void Without\_a\_Space(TextBox a, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

public int Check(int value, TextBox a)

{

try

{

value = Convert.ToInt32(a.Text);

return value;

}

catch (Exception)

{

a.BorderBrush = Brushes.Red;

return 0;

}

}

public bool Confirm\_Action(int a) //подтверждение выхода или сохранения файла

{

bool choice = false;

switch (a)

{

case 0:

if (MessageBox.Show("Вы хотите сохранить ключ?", "Сохранить ключ?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

choice = true;

}

else

{

choice = false;

}

break;

case 1:

if (MessageBox.Show("Вы хотите сохранить ключи?", "Сохранить ключи?", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

choice = true;

}

else

{

choice = false;

}

break;

}

return choice;

}

}

}

RandomKeyGeneration.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Controls;

namespace CryptographicApplication

{

class RandomKeyGeneration

{

Alphabet alph = new Alphabet();

RSA RSA\_obj = new RSA();

Random rnd = new Random();

public string Rand\_Key\_Generation()

{

int value = rnd.Next(1, alph.lang.Length);

return Convert.ToString(value);

}

public string Rand\_Key\_Generation(int sizesourcetext)

{

StringBuilder code = new StringBuilder();

int value;

for (int i = 0; i < sizesourcetext; i++)

{

value = rnd.Next(0, alph.lang.Length);

code.Append(alph.lang[value]);

}

return code.ToString();

}

public string Rand\_Key\_Generation\_Transposition(int sizesourcetext)

{

StringBuilder code = new StringBuilder();

int[] arr = new int[sizesourcetext];

int value1, value2, temp;

for (int i = 0; i < sizesourcetext; i++)

{

arr[i] = i + 1;

}

for (int i = 0; i < sizesourcetext; i++) //перемешиваем элементы

{

value1 = rnd.Next(0, sizesourcetext);

value2 = rnd.Next(0, sizesourcetext);

temp = arr[value1];

arr[value1] = arr[value2];

arr[value2] = temp;

}

for (int i = 0; i < sizesourcetext; i++)

{

if (i == sizesourcetext - 1)

{

code.Append(arr[i]);

}

else

{

code.Append(arr[i] + " ");

}

}

return code.ToString();

}

}

}

Transposition.cs

using System;

namespace CryptographicApplication

{

class Transposition

{

private int[] key = null;

public void SetKey(int[] \_key)

{

key = new int[\_key.Length];

for (int i = 0; i < \_key.Length; i++)

key[i] = \_key[i];

}

public void SetKey(string[] \_key)

{

key = new int[\_key.Length];

for (int i = 0; i < \_key.Length; i++)

key[i] = Convert.ToInt32(\_key[i]);

}

public void SetKey(string \_key)

{

SetKey(\_key.Split(' '));

}

public string Encrypt(string input)

{

for (int i = 0; i < input.Length % key.Length; i++)

input += input[i];

string result = "";

for (int i = 0; i < input.Length; i += key.Length)

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[key[j] - 1] = input[i + j];

for (int j = 0; j < key.Length; j++)

result += transposition[j];

}

return result;

}

public string Decrypt(string input)

{

string result = "";

for (int i = 0; i < input.Length; i += key.Length)

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[j] = input[i + key[j] - 1];

for (int j = 0; j < key.Length; j++)

result += transposition[j];

}

return result;

}

}

}

Monoalphabetic.cs

using System;

using System.Text;

namespace CryptographicApplication

{

class Monoalphabetic

{

Alphabet alph = new Alphabet();

public string Encrypt(string sourcetext, int shift)

{

StringBuilder code = new StringBuilder();

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

code.Append(alph.lang[(j + shift) % alph.lang.Length]);

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

}

}

}

return code.ToString();

}

public string Decrypt(string sourcetext, int shift)

{

StringBuilder code = new StringBuilder();

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

code.Append(alph.lang[(j - shift + alph.lang.Length) % alph.lang.Length]);

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

}

}

}

return code.ToString();

}

}

}

Polyalphabetic.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CryptographicApplication

{

class Polyalphabetic

{

Alphabet alph = new Alphabet();

public string Encrypt(string sourcetext, string key)

{

StringBuilder code = new StringBuilder();

int[] key\_id = new int[key.Length];

int t = 0;

//поиск индексов букв ключа

for (int i = 0; i < key.Length; i++)

{

for (int j = 0; j < alph.lang.Length; j++)

{

if (key[i] == alph.lang[j])

{

key\_id[i] = j;

break;

}

}

}

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

if (t > key.Length - 1)

{

t = 0;

}

code.Append(alph.lang[(j + key\_id[t]) % alph.lang.Length]);

t++;

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

t++;

}

}

}

return code.ToString();

}

public string Decrypt(string sourcetext, string key)

{

StringBuilder code = new StringBuilder();

int[] key\_id = new int[key.Length];

int t = 0;

//поиск индексов букв ключа

for (int i = 0; i < key.Length; i++)

{

for (int j = 0; j < alph.lang.Length; j++)

{

if (key[i] == alph.lang[j])

{

key\_id[i] = j;

break;

}

}

}

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

if (t > key.Length - 1)

{

t = 0;

}

code.Append(alph.lang[(j + alph.lang.Length - key\_id[t]) % alph.lang.Length]);

t++;

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

t++;

}

}

}

return code.ToString();

}

}

}

XOR.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CryptographicApplication

{

class XOR

{

Alphabet alph = new Alphabet();

public string Encrypt\_and\_Decrypt(string sourcetext, string key)

{

StringBuilder code = new StringBuilder();

int[] key\_id = new int[key.Length];

int t = 0;

//поиск индексов букв ключа

for (int i = 0; i < key.Length; i++)

{

for (int j = 0; j < alph.lang.Length; j++)

{

if (key[i] == alph.lang[j])

{

key\_id[i] = j;

break;

}

}

}

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

if (t > key.Length - 1)

{

t = 0;

}

code.Append(alph.lang[(j ^ key\_id[t] % 32) % alph.lang.Length]);

t++;

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

t++;

}

}

}

return code.ToString();

}

}

}

Vernam.cs

using System;

using System.Linq;

using System.Text;

using System.Windows.Controls;

namespace CryptographicApplication

{

class Vernam

{

Alphabet alph = new Alphabet();

public string Encrypt\_and\_Decrypt(string sourcetext, string key)

{

StringBuilder code = new StringBuilder();

int[] key\_id = new int[key.Length];

//поиск индексов букв ключа

for (int i = 0; i < key.Length; i++)

{

for (int j = 0; j < alph.lang.Length; j++)

{

if (key[i] == alph.lang[j])

{

key\_id[i] = j;

break;

}

}

}

for (int i = 0; i < sourcetext.Length; i++)

{

//поиск символа в алфавите

for (int j = 0; j < alph.lang.Length; j++)

{

//если символ найден

if (sourcetext[i] == alph.lang[j])

{

code.Append(alph.lang[(j ^ key\_id[i] % 32) % alph.lang.Length]);

break;

}

//если символ не найден

else if (j == alph.lang.Length - 1)

{

code.Append(sourcetext[i]);

}

}

}

return code.ToString();

}

}

}

RSA.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Numerics;

namespace CryptographicApplication

{

class RSA

{

Alphabet alph = new Alphabet();

private int[] key = null;

public void SetKey(int[] \_key)

{

key = new int[\_key.Length];

for (int i = 0; i < \_key.Length; i++)

key[i] = \_key[i];

}

public void SetKey(string[] \_key)

{

key = new int[\_key.Length];

for (int i = 0; i < \_key.Length; i++)

key[i] = Convert.ToInt32(\_key[i]);

}

public void SetKey(string \_key)

{

SetKey(\_key.Split(' '));

}

public string Encrypt(string sourcetext)

{

StringBuilder code = new StringBuilder();

long e = Convert.ToInt64(key[0]);

long n = Convert.ToInt64(key[1]);

List<string> result = RSA\_Endoce(sourcetext, e, n);

foreach (string item in result)

code.Append(item + "\n");

return code.ToString();

}

public string Decrypt(List<string> sourcetext)

{

StringBuilder code = new StringBuilder();

long d = Convert.ToInt64(key[0]);

long n = Convert.ToInt64(key[1]);

code.Append(RSA\_Dedoce(sourcetext, d, n));

return code.ToString();

}

private List<string> RSA\_Endoce(string sourcetext, long e, long n) //шифрование

{

List<string> result = new List<string>();

BigInteger bi;

for (int i = 0; i < sourcetext.Length; i++)

{

int index = Array.IndexOf(alph.lang, sourcetext[i]);

bi = new BigInteger(index);

bi = BigInteger.Pow(bi, (int)e);

BigInteger bn = new BigInteger((int)n);

bi = bi % bn;

result.Add(bi.ToString());

}

return result;

}

private string RSA\_Dedoce(List<string> sourcetext, long d, long n) //дешифрование

{

string result = "";

int i = 0;

BigInteger bi;

foreach (string item in sourcetext)

{

bi = new BigInteger(Convert.ToDouble(item));

bi = BigInteger.Pow(bi, (int)d);

BigInteger bn = new BigInteger((int)n);

bi = bi % bn;

int index = Convert.ToInt32(bi.ToString());

if (index == -1)

{

if (i < 1)

{

result += " ";

i++;

}

else

{

result += "\n";

i = 0;

}

}

else

{

result += alph.lang[index].ToString();

i = 0;

}

}

return result;

}

public bool IsTheNumberSimple(int n) //является ли число простым

{

if (n < 2)

return false;

if (n == 2)

return true;

for (int i = 2; i < n; i++)

if (n % i == 0)

return false;

return true;

}

public long Calculate\_e(long fi) //вычисление открытой экспоненты

{

long e = 2;

for (int i = 2; i <= fi; i++)

if ((fi % i == 0) && (e % i == 0) && e < fi) //если имеют общие делители

{

e++;

i = 1;

}

return e;

}

public long Calculate\_d(long e, long fi) //вычисление закрытой экспоненты

{

long d = 2;

while (true)

{

if ((d \* e % fi == 1) && d != e)

break;

else

d++;

}

return d;

}

}

}

# Приложение В

Титульный лист презентации