ГУАП

КАФЕДРА № 82

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, кандидат техн. наук |  |  |  | Е.Л. Турнецкая |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| РАЗРАБОТКА ПРОГРАММЫ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ |
| по курсу: МЕТОДОЛОГИЯ И ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 8025М |  |  |  | Н.С. Калимов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Цель работы:**

Разработать программу криптографической защиты, которая выполняет функции шифрования и дешифрования текстовых данных.

**Ход работы:**

Для выполнения этой работы необходимо выполнить 4 шага:

1. Определиться с алгоритмом шифрования
2. Спроектировать интерфейс
3. Реализовать алгоритм на выбранном языке программирования
4. Объединить реализацию и интерфейс.

**Алгоритм шифрования**

В качестве алгоритма шифрования я выбрал шифр Виженера.

Шифр Виженера – это метод полиалфавитного шифрования буквенного текста с использованием ключевого слова. Это простой для понимания и реализации метод, но в то же время он является шифром повышенной надежности, т.к. размывает частоту букв. Это помогает противостоять частотному анализу и прочим статистическим методам дешифровки текста.

Реализация этого шифра следующая:

Шифрование производится по формуле

*,*

*где cj – зашифрованный символ,*

*mj – буква открытого текста,*

*kj – буква ключа,*

*n – модуль для деления.*

Для дешифрования используется следующая формула

В своей работе я использовал шифрование и дешифрование для русского и английского текста. Для шифрования английского текста используется модуль 26 + количество спец. символов, для шифрования русского текста – модуль 33 + количество спец. символов. В качестве ключа используется любое слово соответствующего языка, и если искомый текст длиннее ключевого слова, то последнее накладывается на искомый текст ровно столько раз, сколько нужно его покрыть.

Например, чтобы зашифровать слово «солнце» ключом «юля», нужно записать «юляюля». Чтобы зашифровать слово «солнц» - записать «юляюл», и так далее.

Таким образом, мы получаем набор слов и ключ, «накрываем» ключом набор слов нужное количество раз. Затем нумеруем каждую букву в соответствии с индексом алфавита – эта операция выполняется и для исходного текста, и для ключа. Затем эти два набора каждый, поэлементно, складывается / вычитается по модулю N + количество спец. символов, где N – количество букв в используемом алфавите.

Определившись с алгоритмом, я приступил к макету интерфейса программы.

**Интерфейс приложения**

Прежде чем начать проектирование интерфейса, я зафиксировал требуемые функции от приложения:

* Ввод данных вручную / считывание текстовых файлов;
* Возможность зашифровать и тут же расшифровать обратно текст для его проверки;
* Ввод или генерация ключей для шифрования.

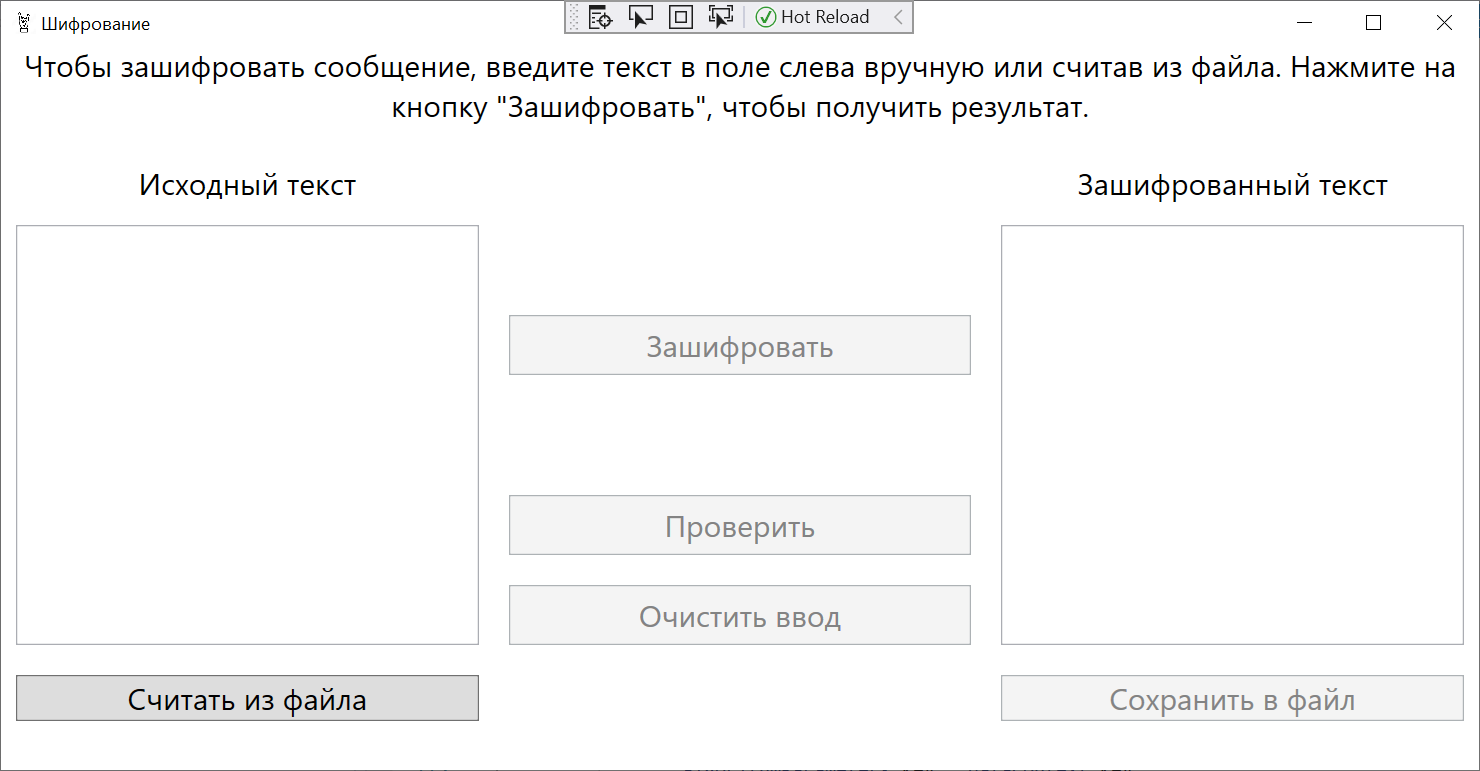
В результате я получил следующий интерфейс:

Рисунок 1 – интерфейс приложения

Определившись с интерфейсом, я перешел к реализации алгоритма на выбранном языке программирования. Опишу теперь и используемый стек технологий в придачу:

Для разработки использовалась IDE Visual Studio Community 2019, хотя для разработки .NET-приложений можно использовать и обычный блокнот. Тем не менее, среда разработки позволяет мне легче отыскивать баги и ошибки в коде. В качестве фреймворка я выбрал WPF – Windows Presentation Foundation, и таким образом разработал приложение для рабочего стола. Код – на языке C#.

Для построения архитектуры приложения воспользовался паттерном MVVM – Model – View – ViewModel, позволяющий максимально отделить исполняемый код от графического интерфейса. При всем желании, это же приложение потом можно будет перевести в форму веб-приложения или даже в мобильный клиент.

Весь код фиксировался и хранится по сей день в моем репозитории на GitHub, и в качестве контроля версий использовался Git.

Итак, продемонстрирую свою реализацию алгоритма шифра Виженера.

Для того, чтобы не привязываться к конкретному алгоритму шифрования, я использую интерфейс ICodec, где Codec – гибрид английских слов CodeDecode. Этот интерфейс предоставляет всего 4 публичных метода – зашифровать, расшифровать и получить результаты.

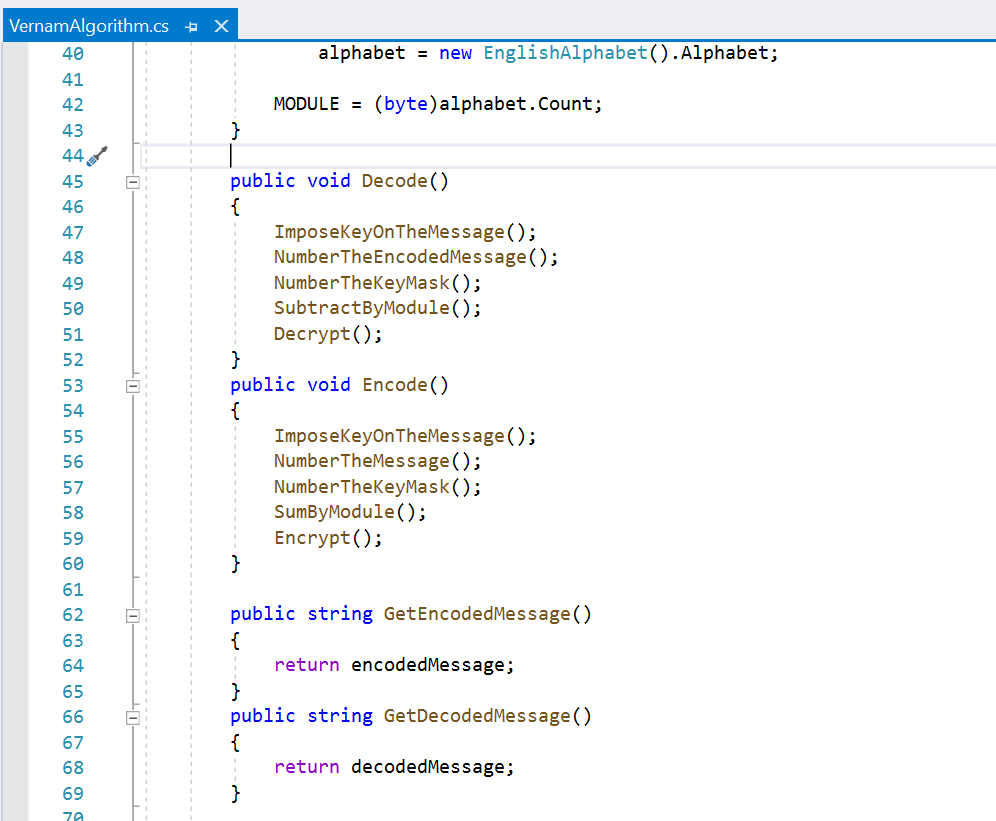


Рисунок 2 – публичные методы шифратора / дешифратора.

Вместо документирования своего кода комментариями, я руководствуюсь принципами написания чистого кода Роберта Мартина и пишу самодокументируемый код. Принцип его работы можно понять, глядя на названия методов и переменных. Документацию же использую только для методов API, которые будут публиковаться в Сети и пользоваться другими пользователями, или для публичных методов / свойств, где не удалось с помощью кода передать свои намерения.

Итак, процесс шифрования и дешифрования я разбил на 5 этапов:

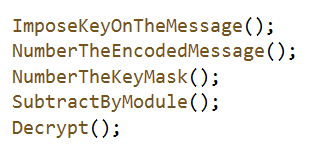


Рисунок 3 – последовательность действий для шифрования / дешифрования.

Наложить ключ на сообщение, пронумеровать искомое сообщение (EncodedMessage для зашифрованного, просто Message для исходного), пронумеровать маску ключа – ключ, наложенный на исходный текст; сложить по модулю алфавита для шифрования / вычесть по модулю алфавита для дешифрования, зашифровать / дешифровать.

Теперь покажу каждый метод в отдельности. ImposeKeyOnTheMessage:



Рисунок 4 – накладывание ключа на текст сообщения

NumberThe(Encoded)Message:

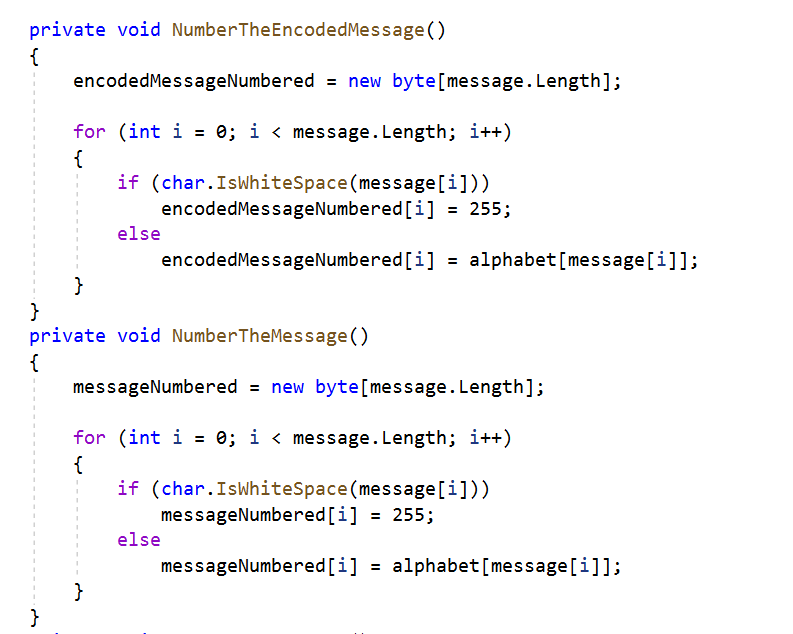


Рисунок 5 – нумерация букв сообщения по порядку в алфавите.

Аналогичный процесс выполняется для нумерации ключа, покрывающего шифруемый текст.

Subtract(Sum)ByModule – Сложение(Вычитание) по модулю:

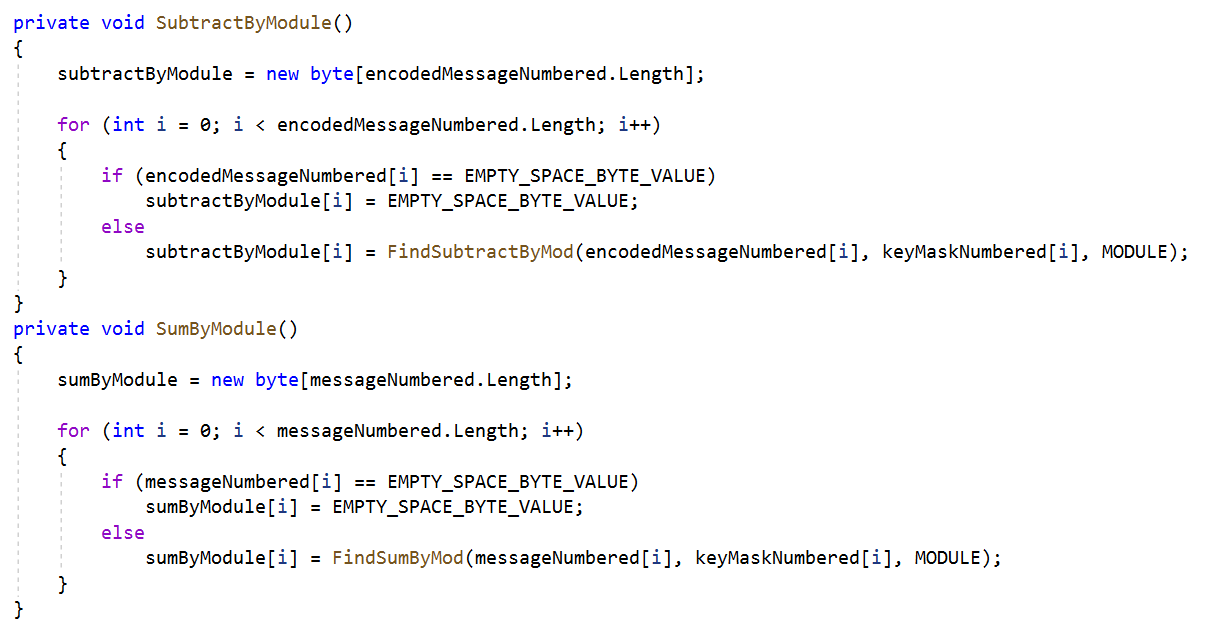


Рисунок 6 – методы сложения и вычитания по модулю.

Сам процесс шифрования (дешифрование проходит аналогично):

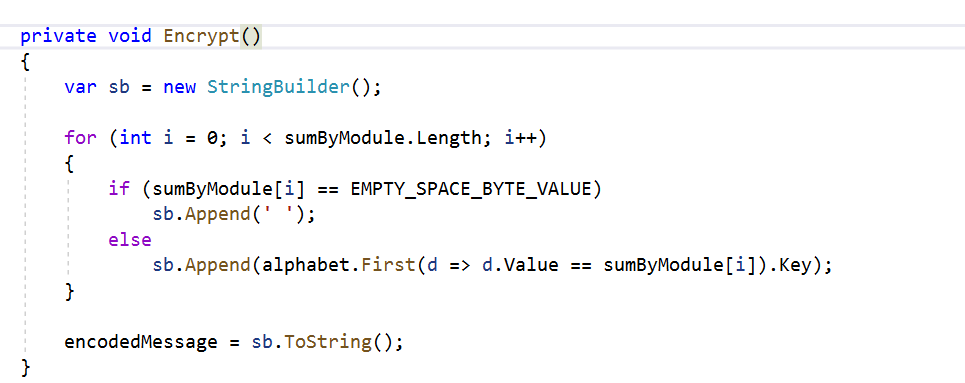
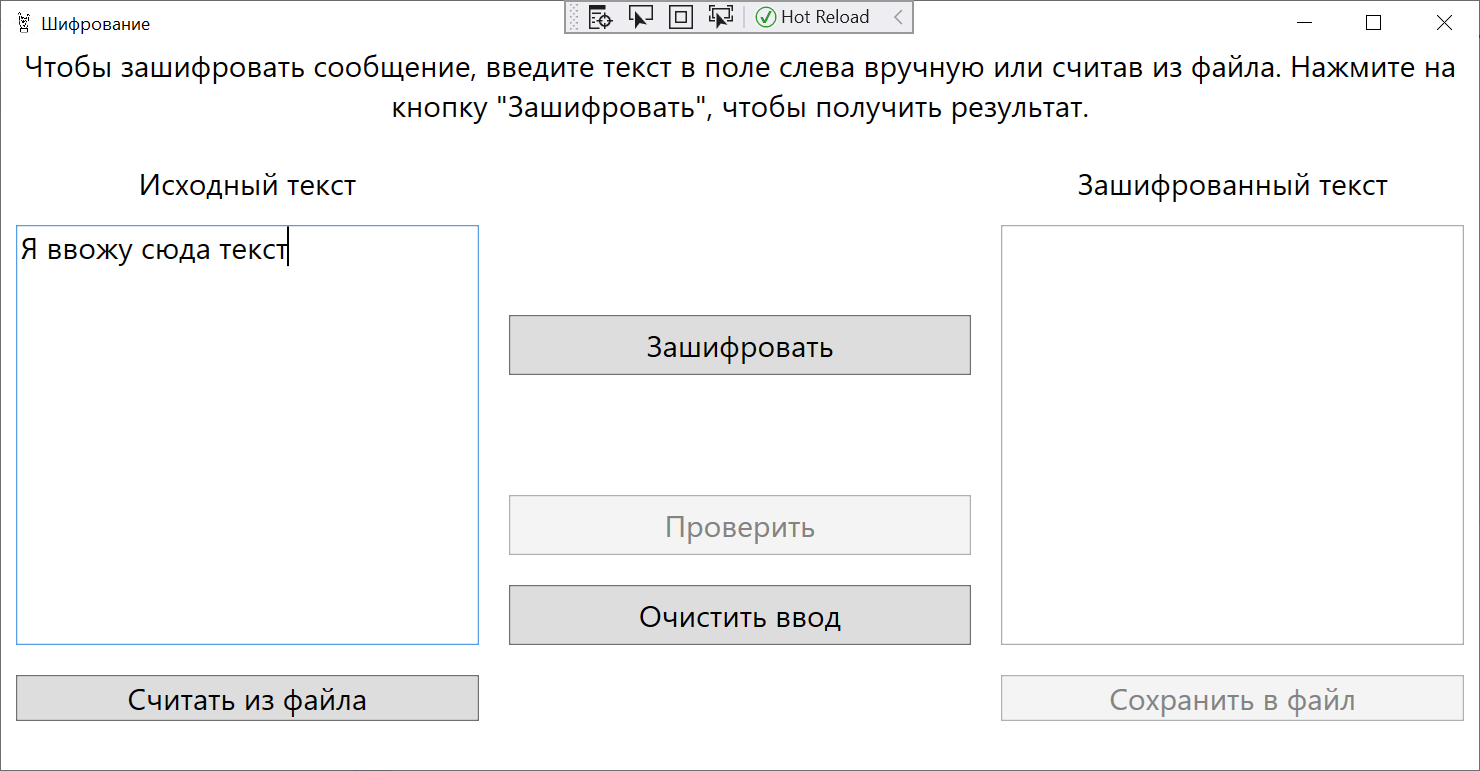


Рисунок 7 – процесс шифрования.

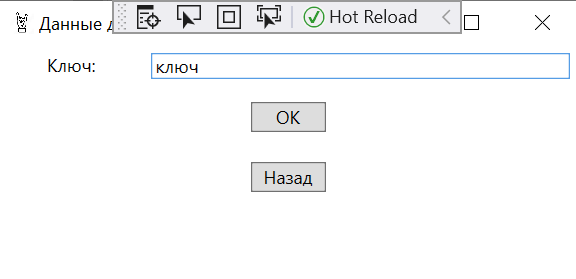
Весь исходный код хранится в репозитории GitHub по ссылке <https://github.com/mekk1t/IS_Design_Methodology>

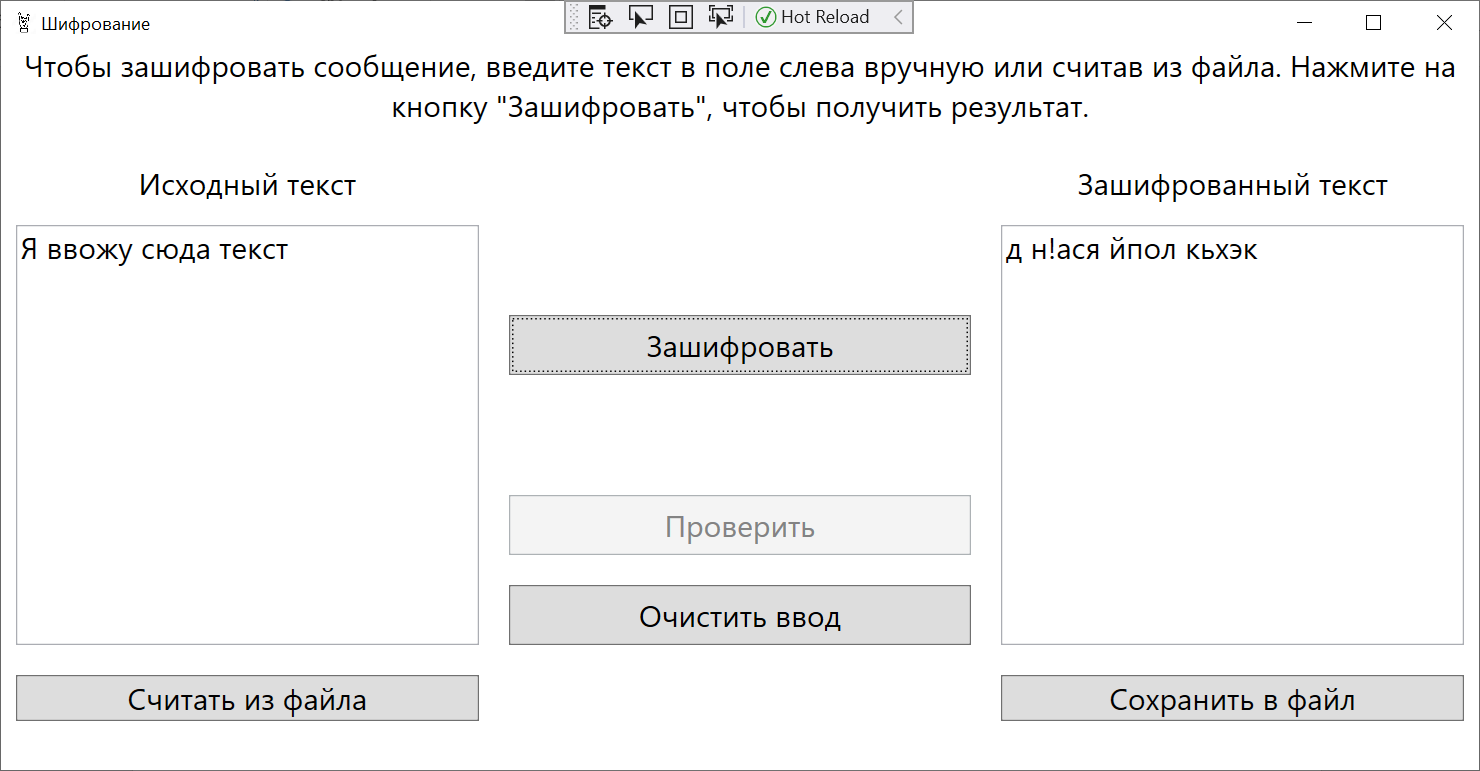
Там можно посмотреть на мой подход к архитектуре приложения, реализацию интерфейса и прочие тонкости.

Само приложение вручную было протестировано, результаты следующие:

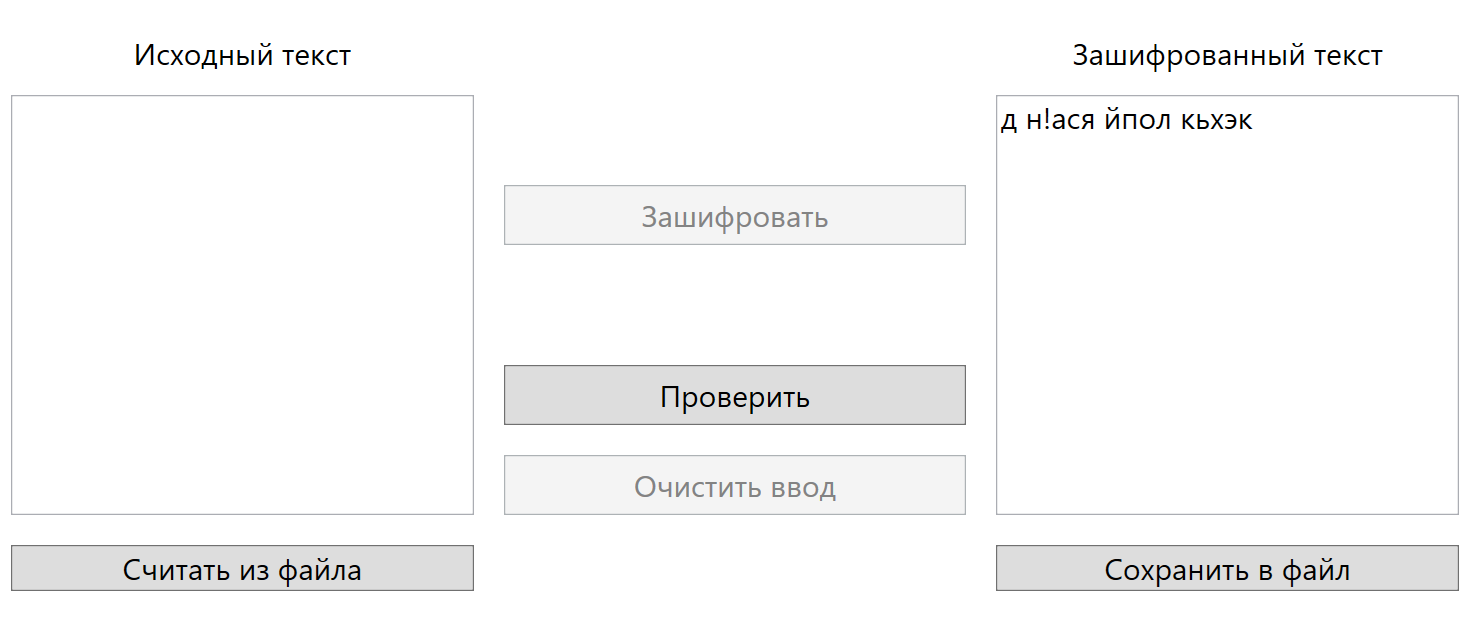


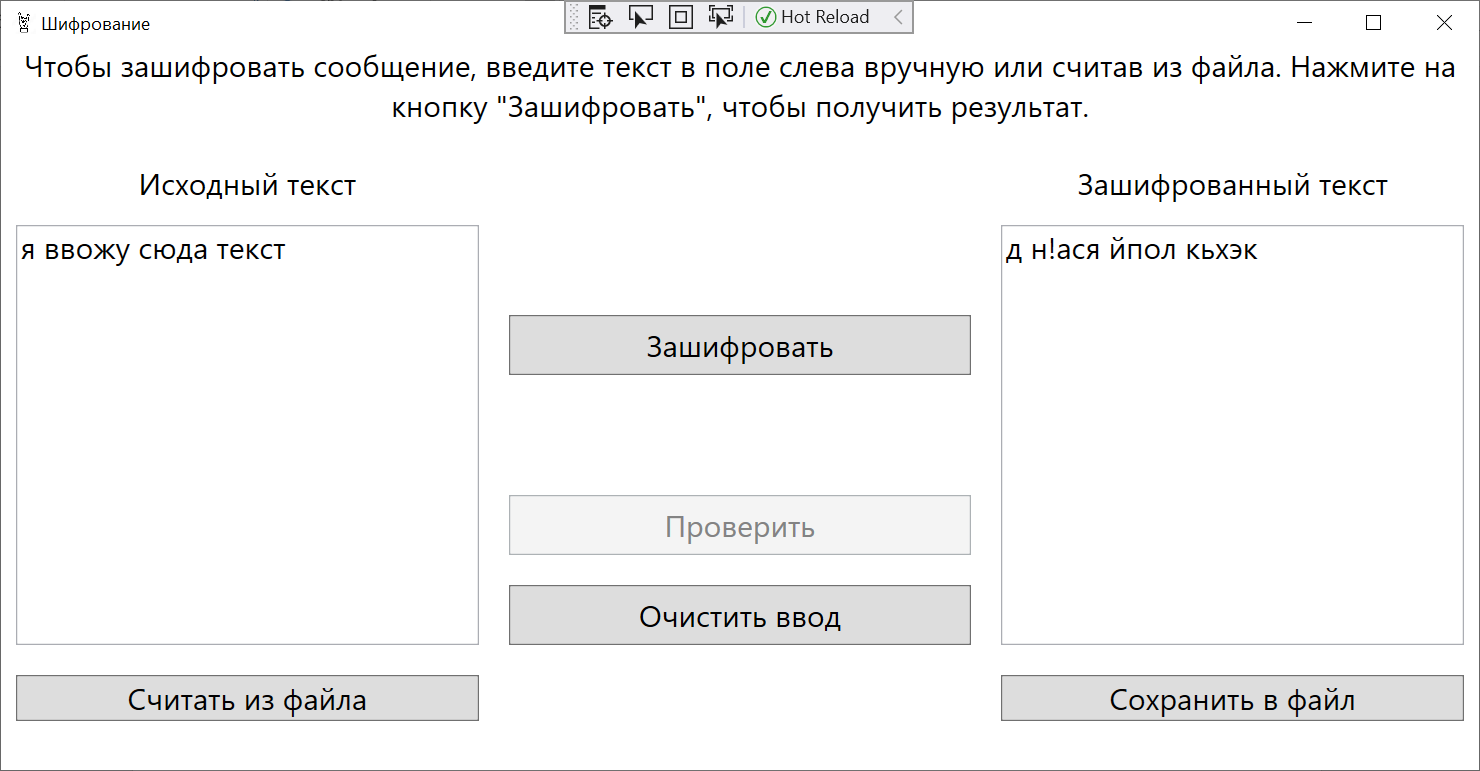
Обратите внимание, что я не позволяю пользователю сохранить результат в файл (т.к. его еще нет), а также не позволяю проверить по той же причине. Нажав на кнопку «Зашифровать», вводим ключ и получаем результат:



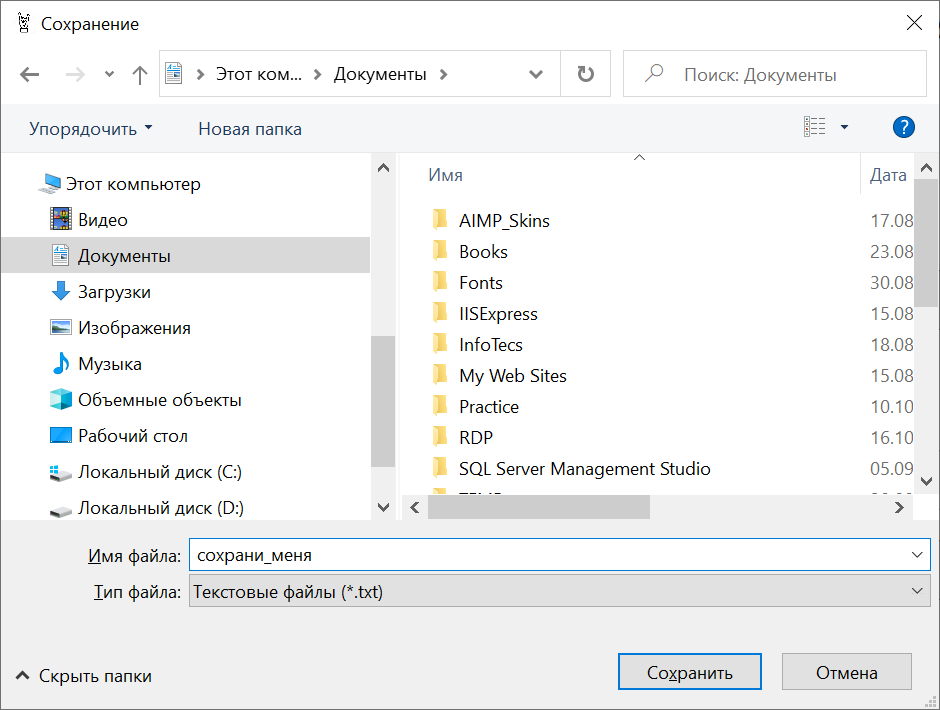


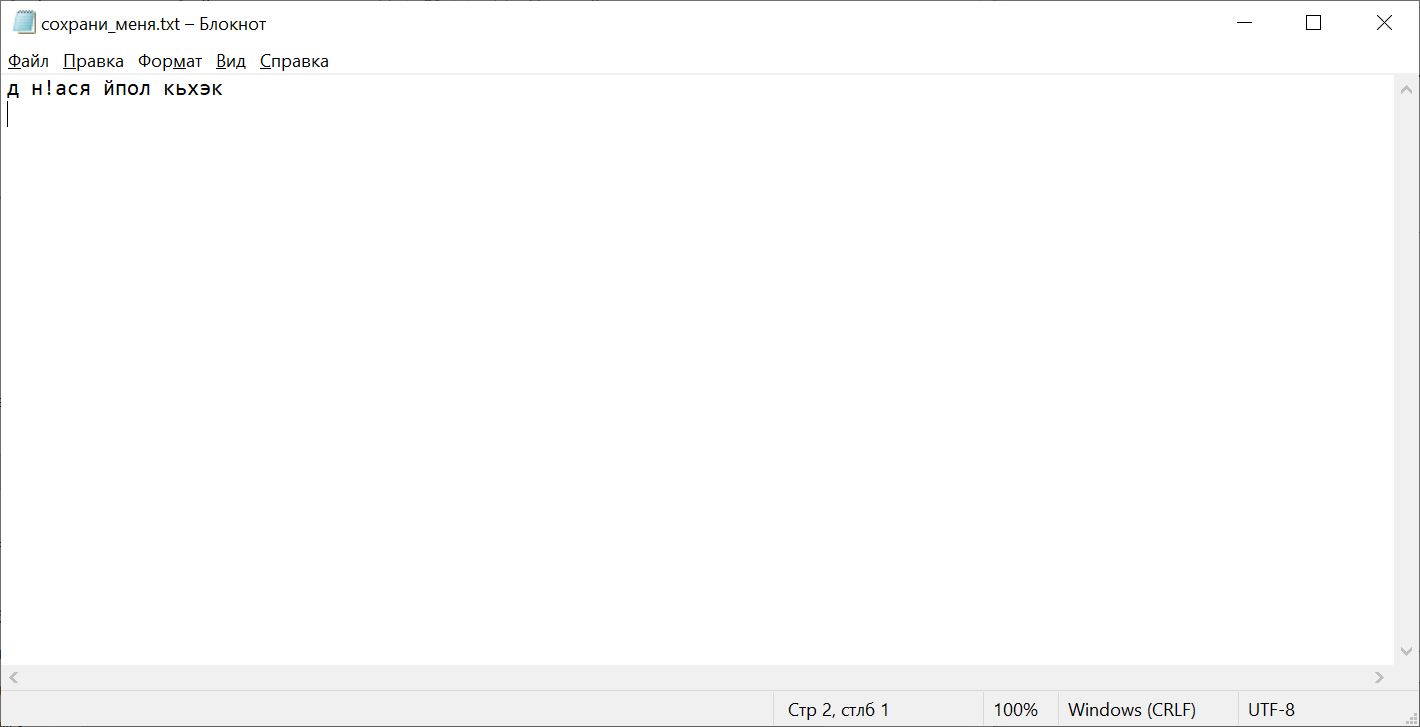
Чтобы «проверить», нужно сначала очистить поле ввода для исходного текста.





Сохраняем результат в файл:





**Вывод:**

Я вспомнил принципы криптографической защиты и проанализировал имеющиеся шифры и их алгоритмы. Закрепил навыки разработки приложений для рабочего стола. Реализовал программу для шифрования и дешифрования текста.

В планах – развить приложение до уровня шифрования файлов, сделав это путем работы и кодирования base64-строк.