МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Отчет по курсовой работе

по дисциплине «Защита данных»

по теме №21:

Программная реализация криптоалгоритма RC5

Работу выполнил
студент группы А-13б-20
Бегунов Никита
Преподаватель:
Хорев Павел Борисович

Москва 2023

Оглавление

Введение

Цель. Цель курсовой работы заключается в программной реализации криптоалгоритма RC5.

Задачи. Для выполнения работы были поставлены следующие задачи:

- Проектирование пользовательского интерфейса, выбор языка и среды программирования;
- Реализация криптоалгоритма RC5 средствами языка программирования без использования встроенных библиотечных средств;
- Реализация проекта программы в среде программирования для реализации разработанного пользовательского интерфейса;
- Тестирование и отладка программы;
- Подготовка отчета по курсовой работе.

Требования к работе. Программа должна поддерживать следующие функции:

- Возможность шифрования/расшифрования как выбираемых файлов любого типа, так и вводимых текстовых сообщений на ключе, выводимом из парольной фразы с регулируемой пользователем длинной и сложностью;
- Возможность выбора пользователем режима шифрования/расшифрования;
- Возможность при расшифровании проверять правильность парольной фразы (например, путем добавления к данным перед их шифрованием сигнатуры – специальной строки символов – с проверкой ее наличия в расшифрованных данных и удалением из них в случае успешной проверки);
- Возможность удаления шифруемого (расшифровываемого) файла после выполнения криптографической операции;
- Возможность сохранения зашифрованного введенного пользователем текстового сообщения в файле с выбираемым именем.

1. Описание алгоритма

1.1 Блочный шифр

Блочный шифр - разновидность симметричного шифра, оперирующего группами бит фиксированной длины — блоками, характерный размер которых меняется в пределах 64–256 бит. Если исходный текст (или его остаток) меньше размера блока, перед шифрованием его дополняют. Фактически, блочный шифр представляет собой подстановку на алфавите блоков, которая, как следствие, может быть моно- или полиалфавитной. Блочный шифр является важной компонентой многих криптографических протоколов и широко используется для защиты данных, передаваемых по сети.

К достоинствам блочных шифров относят сходство процедур шифрования и расшифрования, которые, как правило, отличаются лишь порядком действий. Это упрощает создание устройств шифрования, так как позволяет использовать одни и те же блоки в цепях шифрования и расшифрования. Гибкость блочных шифров позволяет использовать их для построения других криптографических примитивов: генератора псевдослучайной последовательности, поточного шифра, имитовставки и криптографических хешей.

1.2 Описание криптоалгоритма RC5

1.2.1 Общее описание алгоритма

RC5 — это блочный шифр, разработанный Роном Ривестом из компании RSA Security с переменным количеством раундов, длиной блока и длиной ключа. В классическом алгоритме используются три примитивные операции и их инверсии:

- Сложение по модулю 2^{ω}
- Побитовое исключающее ИЛИ (XOR)
- Операции циклического сдвига на переменное число бит ($x \ll y$)

Шифрование по алгоритму RC5 состоит из двух этапов. Процедура расширения ключа и непосредственно шифрование. Для расшифрования выполняется сначала процедура расширения ключа, а затем операции, обратные процедуре шифрования. Все операции сложения и вычитания выполняются по модулю 2^{ω}.

Так как алгоритм RC5 имеет переменные параметры, то для спецификации алгоритма с конкретными параметрами принято обозначение RC5-W/R/b, где

- W — половина длины блока в битах, возможные значения 16, 32 и 64. Для эффективной реализации величину W рекомендуют брать равным

машинному слову. Например, для 32-битных платформ оптимальным будет выбор W=32, что соответствует размеру блока 64 бита.

- R — число раундов, возможные значения от 0 до 255. Увеличение числа раундов обеспечивает увеличение уровня безопасности шифра. Так, при R=0 информация шифроваться не будет. Также алгоритм RC5 использует таблицу расширенных ключей размера слов, которая получается из ключа, заданного пользователем.
- b – длинна ключа в байтах, возможные значения от 0 до 255.

Перед непосредственно шифрованием или расшифрованием данных выполняется процедура расширения ключа. Процедура генерации ключа состоит из четырёх этапов:

- генерация констант
- разбиение ключа на слова
- построение таблицы расширенных ключей
- перемешивание

1.2.2 Генерация констант

Для заданного параметра W генерируются две псевдослучайные величины используя две математические константы: e (экспонента) и f (золотое сечение).

$$Q_\omega \leftarrow Odd\big((f-1)\cdot 2^\omega\big)$$

$$P_\omega \leftarrow Odd\big((e-1)\cdot 2^\omega\big)$$

Где Odd() – это округление до ближайшего нечетного целого.

Для $\omega = 16, 32, 64$ получатся следующие константы:

$$P_{16} = B7E1_{16}$$

$$Q_{16} = 9E37_{16}$$

$$P_{32} = B7E15163_{16}$$

$$Q_{32} = 9E3779B9_{16}$$

$$P_{64} = B7E151628AED2A6B_{16}$$

$$Q_{64} = 9E3779B97F4A7C15_{16}$$

1.2.3 Разбиение ключа на слова

На этом этапе происходит копирование ключа $K_0 \dots K_{b-1}$ в массив слов $L_0 \dots L_{c-1}$, где $c = b/u$, где $u = W/8$, то есть, количество байт в слове.

Если b не кратен $W/8$, то L дополняется нулевыми битами до ближайшего размера c, кратного $W/8$.

В случае, если $b = c = 0$, то мы устанавливаем значение $c = 1$, а $L_0 = 0$.

1.2.4 Построение таблицы расширенных ключей

На этом этапе происходит построение таблицы расширенных ключей $S_0 \dots S_{2\cdot(R+1)-1}$, которое выполняется следующим образом:

$$S_0 = P_\omega$$

$$S_{i+1} = S_i + Q_\omega$$

1.2.5 Перемешивание

Циклические N раз выполняются следующие действия:

$$G = S_i = (S_i + G + H) \lll 3$$

$$H = L_j = \left(L_j + G + H\right) \lll (G + H)$$

$$i = (i + 1) mod \left(2(R + 1)\right)$$

$$j = (j + 1) mod\ c$$

Причем G, H, i, j – временные переменные, начальные значения которых равны 0. Количество итераций цикла N – это максимальное из двух значений $3 * c$ и $(3 \cdot 2 \cdot (R + 1))$.

1.2.6 Шифрование

Перед первым раундом выполняются операции наложения расширенного ключа на шифруемые данные:

$$A = (A + S_0)\ mod\ 2^\omega$$

$$B = (B + S_1)\ mod\ 2^\omega$$

В каждом раунде выполняются следующие действия:

$$A = \left((A \oplus B) \lll B\right) + S_{2i}$$

$$B = \left((B \oplus A) \lll A\right) + S_{2i+1}$$

1.2.7 Расшифрование

Для расшифрования данных используются обратные операции, то есть для $i = R, R - 1, \dots, 1$ выполняются следующие раунды:

$$B = \left((B - S_{2i+1}) \ggg A\right) \oplus A$$

$$A = \left((A - S_{2i}) \ggg B\right) \oplus B$$

После выполнения всех раундов, исходное сообщение находится из выражения:

$$B = (B - S_1)\ mod\ 2^{\omega}$$

$$A = (A - S_0)\ mod\ 2^{\omega}$$

1.3 Режимы шифрования

1.3.1 Описание режимов шифрования

Режим шифрования — метод применения блочного шифра (алгоритма), позволяющий преобразовать последовательность блоков открытых данных в последовательность блоков зашифрованных данных. При этом для шифрования одного блока могут использоваться данные другого блока.

Обычно режимы шифрования используются для изменения процесса шифрования так, чтобы результат шифрования каждого блока был уникальным вне зависимости от шифруемых данных и не позволял сделать какие-либо выводы об их структуре. Это обусловлено, прежде всего, тем, что блочные шифры шифруют данные блоками фиксированного размера, и поэтому существует потенциальная возможность утечки информации о повторяющихся частях данных, шифруемых на одном и том же ключе.

1.3.2 Режим электронной кодовой книги (режим простой замены) – ECB

Шифрование: пусть дано сообщение P (открытый текст). Во время шифрования выполняются следующие действия (Рис. 1.1):

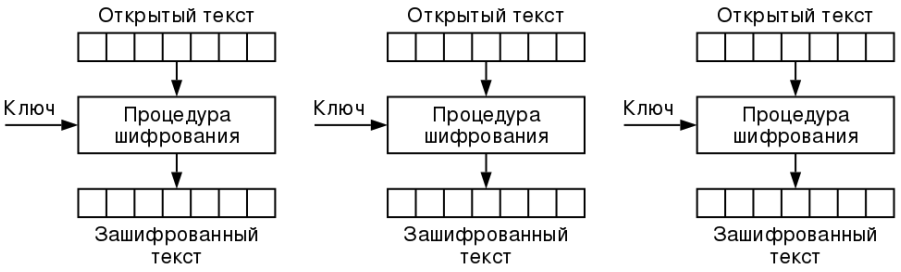

Рис. 1.3.1. Шифрование в режиме ECB (режим электронной кодовой книги)

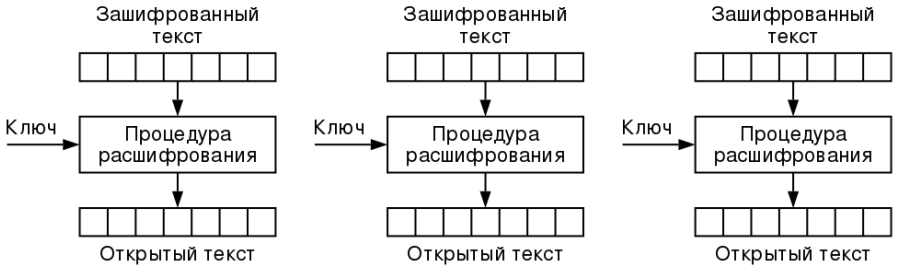

Рис. 1.3.2. Расшифрование в режиме ECB (режим электронной кодовой книги)

1. Сообщение делится на блоки одинакового размера. Размер (длинна) блока равен n и измеряется в битах. В результате получается последовательно блоков $P_1, P_2, \dots, P_q$. Последний блок при необходимости дополняется до длинны n.
2. Каждый блок P_i шифруется алгоритмом шифрование E_k с использованием ключа k:

$$C_i = E_k(P_i, k),$$

Где:

- i – номер блока;
- k – ключ;
- P_i – блок сообщения (открытый текст);
- C_i – зашифрованный блок (шифротекст);
- E_k – функция, выполняющая блочное шифрование.

В результате получаются зашифрованные блоки $C_1, C_2, \dots, C_q$.

Расшифровка: выполняется функцией D_k с использованием того же ключа k:

$$\mathrm{P}_i = D_k(C_i, k)$$

1.3.3 Режим сцепления блоков шифротекста – CBC

Для шифрования некоторого сообщения P выполняются следующие действия:

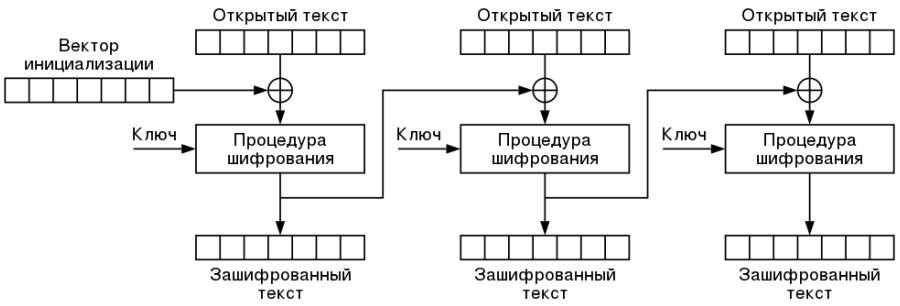

Рис. 1.3.3. Шифрование в режиме CBC (режим сцепления блоков шифротекста)

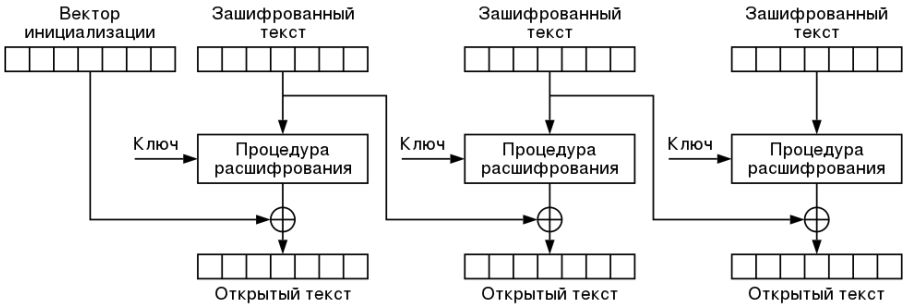

Рис. 1.3.4. Расшифрование в режиме CBC (режим сцепления блоков шифротекста)

- Сообщение разбивается на блоки одинакового размера. Размер (длинна) блока равен n и измеряется в битах. При необходимости последний блок дополняется до длинны n.
- Шифрование очередного (i-го) блока сообщения (P_i) выполняется с использованием предыдущего зашифрованного (($i-1$)-го) блока (C_{i-1}). Для первого блока (P_1) зашифрованного блока (C_0) не существует, поэтому первый блок шифруют с использованием «вектора инициализации»: $C_0 = IV$ (вектор инициализации – случайное число). Размер (длинна) IV равна размеру блока n.
- В функцию шифрования E_k передается сумма по модулю 2 («$\oplus$») текущего блока сообщения P_i и предыдущего зашифрованного блока C_{i-1}: $\mathrm{C}_i = E_k(P_i \oplus C_{i-1}, k)$, где:
  - i – номер блока;
  - k – ключ;
  - IV – вектор инициализации;
  - P_i – блок сообщения (открытый текст);
  - C_i – зашифрованный блок (шифротекст);
  - C_{i-1} – шифрованный текст (шифротекст), полученный на предыдущем шаге шифрования;
  - E_k – функция, выполняющая блочное шифрование.

Расшифровка выполняется функцией D_k с использованием тех же ключа k и вектора инициализации IV:

$$C_0 = IV$$

$$P_i = C_{i-1} \oplus D_k(C_i, k)$$

1.3.4 Режим обратной связи по шифротексту – CFB

Во время шифрования каждый блок открытого текста складывается по модулю 2 с блоком, зашифрованным на предыдущем шаге.

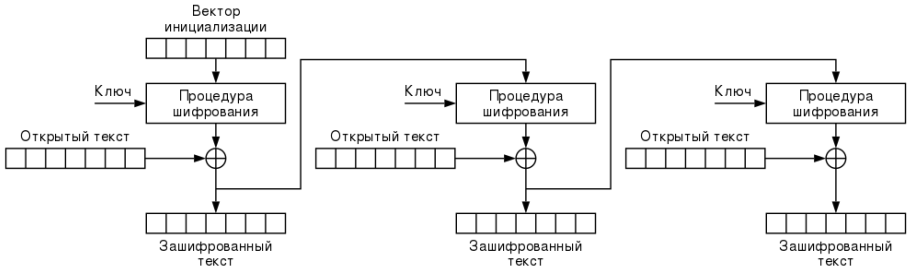

Рис. 1.3.5. Шифрования в режиме CFB (режим обратной связи по шифротексту)

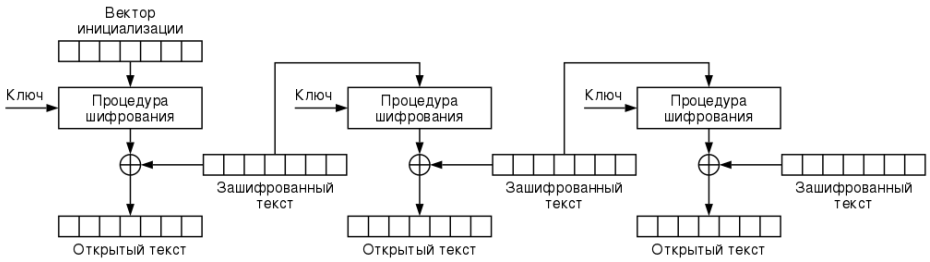

Рис. 1.3.6. Расшифрование в режиме CFB (режим обратной связи по шифротексту)

Шифрования может быть описано следующим образом:

$$C_0 = IV$$

$$C_i = E_k(C_{i-1}, k) \oplus P_i$$

$$P_i = E_k(C_{i-1}, k) \oplus C_i$$

Где

- i – номер блока;
- k – ключ;
- IV – вектор инициализации;
- P_i – блок сообщения (открытый текст);
- C_i – зашифрованный блок (шифротекст);
- C_{i-1} – шифрованный текст (шифротекст), полученный на предыдущем шаге шифрования;
- E_k – функция, выполняющая блочное шифрование.

1.3.5 Режим обратной связи по выходу – OFB

Режим обратной связи вывода превращает блочный шифр в синхронный шифр потока: он генерирует ключевые блоки, которые являются результатом сложения с блоками открытого текста, чтобы получить зашифрованный текст. Так же, как с другими шифрами потока, зеркальное отражение в

зашифрованном тексте производит зеркально отражённый бит в открытом тексте в том же самом местоположении.

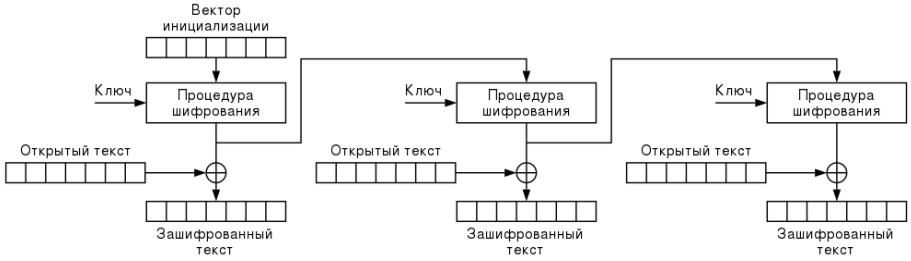

Рис. 1.3.7. Шифрование в режиме OFB (режим обратной связи по выходу)

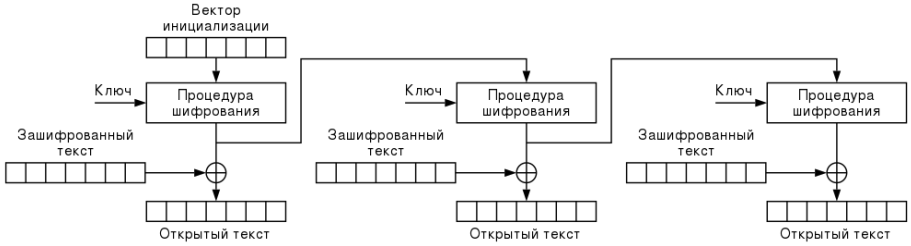

Рис. 1.3.8. Расшифрование в режиме OFB (режим обратной связи по выходу)

Схема шифрования в режиме OFB определяется следующим образом:

$$K_0 = IV$$

$$K_i = E(K_{i-1}, k) \text{ для } i = 1, \dots, n$$

$$C_i = K_i \oplus P_i$$

Где

- i – номер блока;
- k – ключ;
- IV – вектор инициализации;
- K_i – ключевой поток;
- P_i – блок сообщения (открытый текст);
- C_i – зашифрованный блок (шифротекст);
- E – функция, выполняющая блочное шифрование.

2. Проектирование программной реализации

2.1 Пользовательский интерфейс

Пользовательский интерфейс реализуется на языке программирования Python с использованием среды разработки PyCharm. Для реализации окон используется библиотека PyQt5. Ниже показаны примеры окон пользовательского интерфейса.

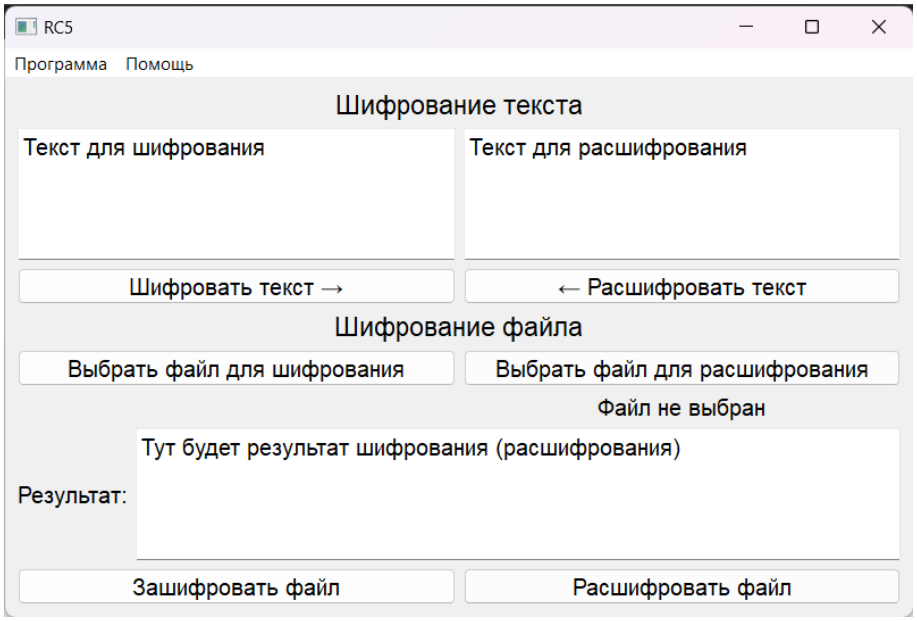

Рис. 2.1.1. Главная форма оконного приложения

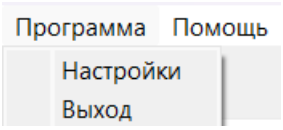

Рис. 2.1.2. Пункты выбора в пункте меню «Программа»

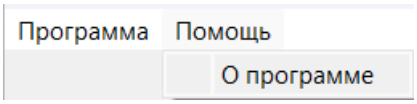

Рис. 2.1.3. Пункты выбора в пункте меню «Помощь»

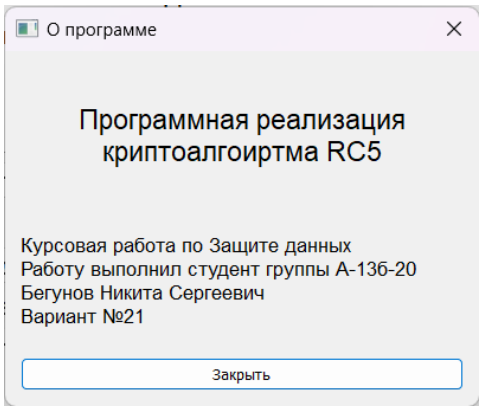

Рис. 2.1.4. Окно «О программе»

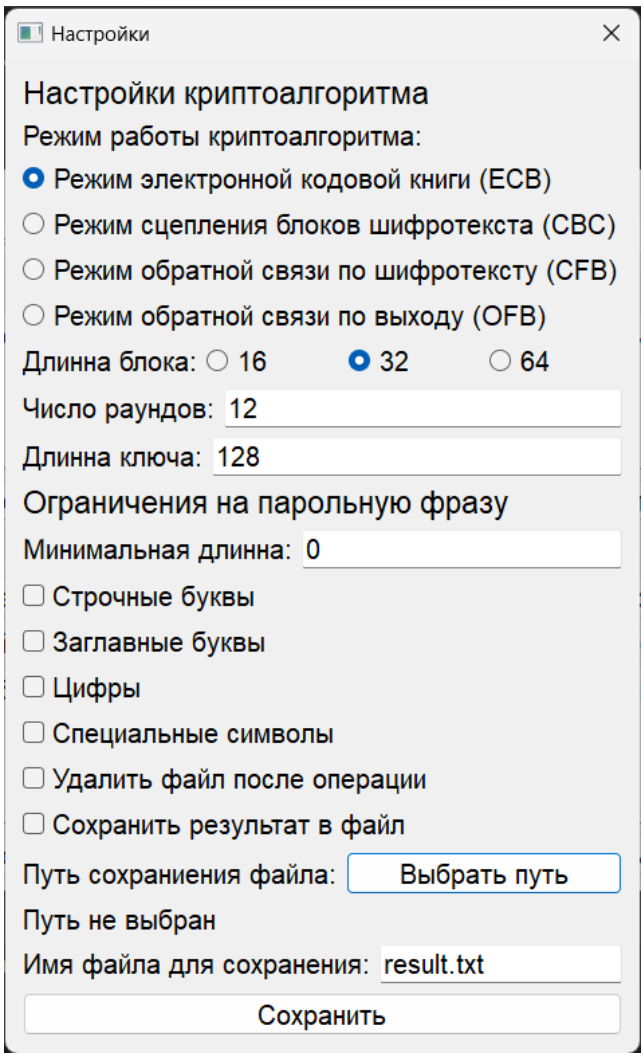

Рис. 2.1.5. Окно «Настройки»

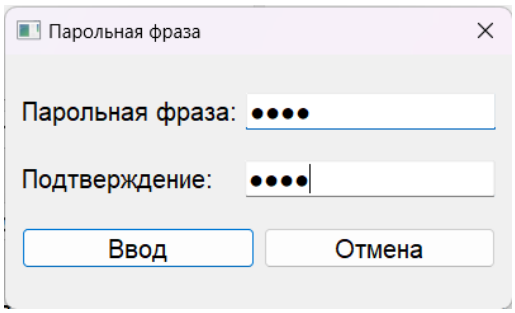

Рис. 2.1.6. Окно ввода парольной фразы для шифрования

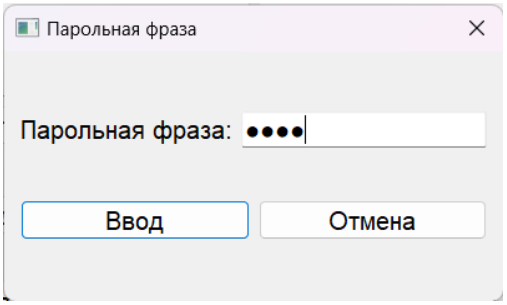

Рис. 2.1.7. Окно ввода парольной фразы для расшифрования

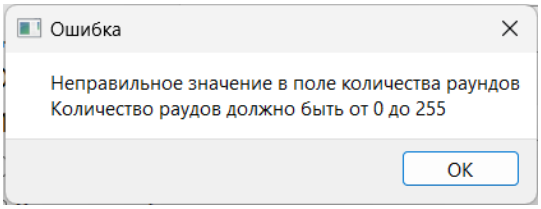

Рис. 2.1.8. Ошибка при вводе количества раундов

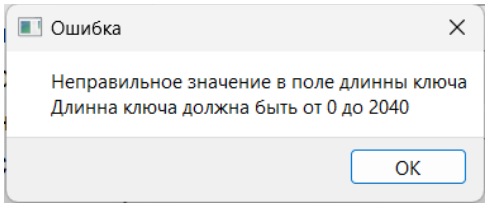

Рис. 2.1.9. Ошибка при вводе длинны ключа

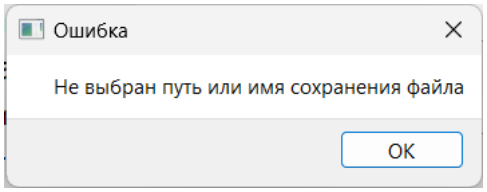

Рис. 2.1.10. Ошибки при выборе пункта сохранения в файл

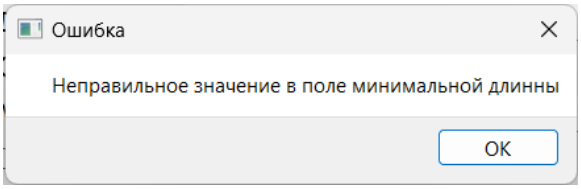

Рис. 2.1.11. Ошибка при вводе неверного значения в поле минимальной длинны

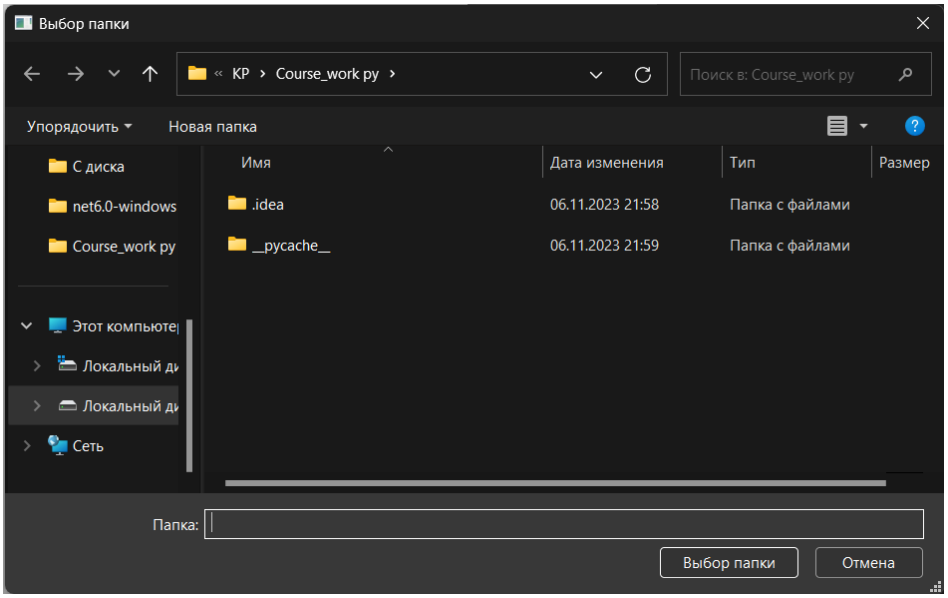

Рис 2.1.12. Окно выбора папки для сохранения файла

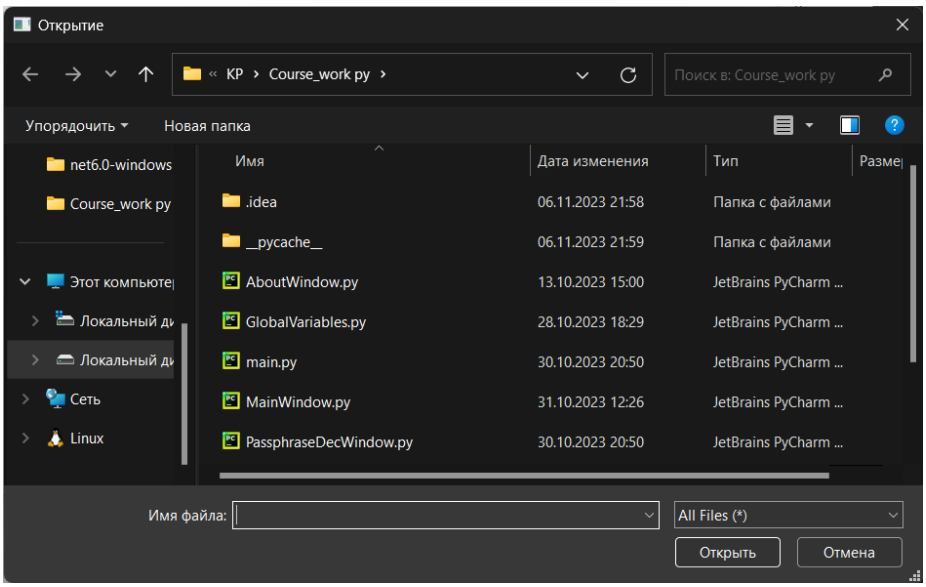

Рис 2.1.13. Выбор файла для шифрования или расшифрования

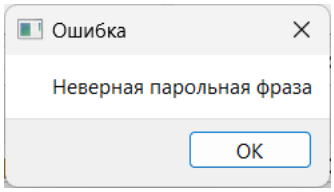

Рис. 2.1.14. Ошибка при вводе неверной парольной фразы

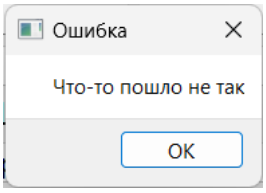

Рис. 2.1.15. Другие ошибки при шифровании или расшифровании

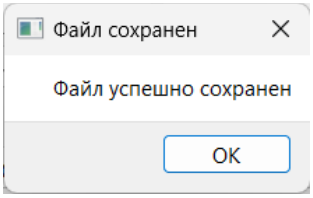

Рис. 2.1.16. Успешное сохранение результата в файл

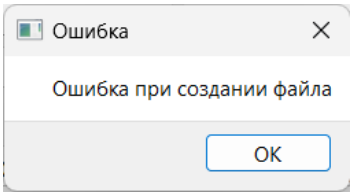

Рис. 2.1.17. Ошибка при сохранении результата в файл

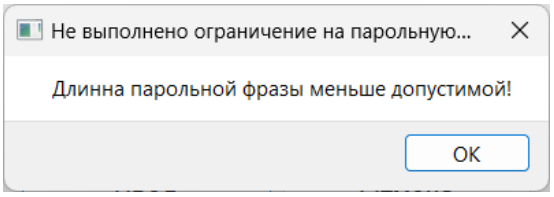

Рис. 2.1.18. Ошибка при вводе парольной фразы (длинна меньше допустимой)

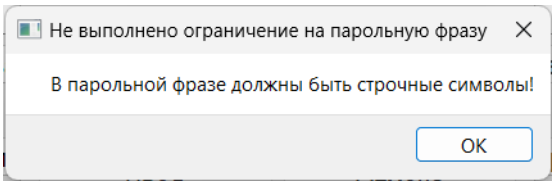

Рис. 2.1.19. Ошибка при вводе парольной фразы (отсутствуют строчные символы)

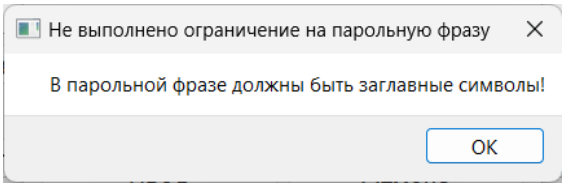

Рис. 2.1.20. Ошибка при вводе парольной фразы (отсутствуют заглавные символы)

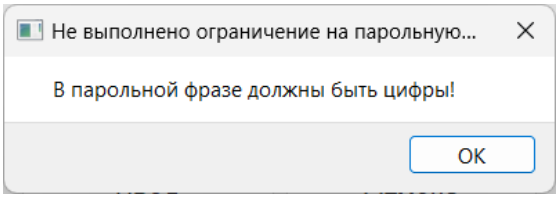

Рис. 2.1.21. Ошибка при вводе парольной фразы (отсутствуют цифры)

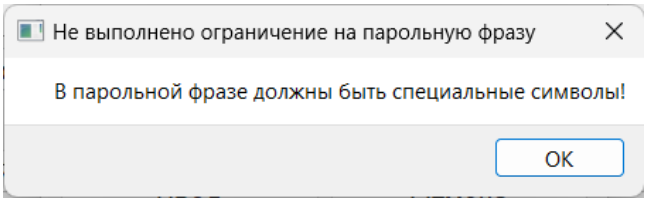

Рис. 2.1.18. Ошибка при вводе парольной фразы (отсутствуют специальные символы)

2.2 Программная реализация

Для программной реализации приложения были разработаны следующие классы (с их свойствами и методами).

Класс MainWindow

Класс MainWindow содержит информацию о главном окне программы: разметке расположения элементов на окне, а также действиях в главном окне.

Свойства и методы класса:

1. Метод __init__ инициализации класса содержит информацию о расположении элементов на главном окне, присваивает переменным начальные значения, запускает создание меню;
2. Метод create_menu_bar создает меню в главном окне программы;
3. Метод close_action закрывает программу при нажатии на пункт выхода в меню;
4. Метод about_action открывает окно «О программе» при выборе соответствующего пункта в меню;
5. Метод settings_action открывает окно «Настройки» при выборе соответствующего пункта в меню;
6. Метод click_enc_file шифрует файл при нажатии на соответствующую кнопку в главном окне программы;
7. Метод click_dec_file расшифровывает файл при нажатии на соответствующую кнопку в главном окне программы;
8. Метод click_encrypt_text шифрует введенный текст при нажатии на соответствующую кнопку в главном окне программы;
9. Метод click_decrypt_text расшифровывает введенный текст при нажатии на соответствующую кнопку в главном окне программы;
10. Метод click_select_enc_file открывает окно выбора файла для шифрования;
11. Метод click_select_def_file открывает окно выбора файла для расшифрования;
12. Свойство file_encrypt содержит информацию о файле для шифрования;
13. Свойство file_decrypt содержит информацию о файле для расшифрования.

Класс AboutDialog

Класс AboutDialog содержит информацию об окне «О программе»: разметке расположения элементов на окне и действиях при нажатии на кнопки в окне.

Свойства и методы класса:

1. Метод __init__ инициализации класса содержит информацию о расположении элементов на окне, а также обработку при нажатии на кнопку «Закрыть».

Класс SettingsDialog

Класс SettingsDialog содержит настройки криптоалгоритма, установку ограничений на парольную фразу и выбор варианта сохранения результата выполнения программы.

Свойства и методы класса:

1. Метод __init__ инициализации класса содержит информацию о расположении элементов на окне, присваивает элементам на окне необходимые значения;
2. Метод radiobutton_blocks_clicked изменяет значение количества блоков при выборе соответствующей радиокнопки;
3. Метод radiobutton_mods_clicked изменяет значение режима шифрования при выборе соответствующей радиокнопки;
4. Метод btn_path_file_save_click открывает окно выбора папки для сохранения файла;
5. Свойство path_file_save содержит информацию о выбранном пользователем пути сохранения файла;
6. Метод save_click проверяет введенные пользователем значения и сохраняет настройки в класс GlobalSettings.

Класс GlobalSettings

Класс GlobalSettings хранит значения глобальных переменных программы.

Свойства и методы класса:

1. Свойство blocks содержит информацию о длине блока шифрования (16, 32 или 64);
2. Свойство rounds содержит информацию о количестве раундов шифрования (до 0 до 255);
3. Свойство key_len содержит информацию о длине ключа (от 0 до 2040 бит);
4. Свойство min_len содержит информацию о минимальной длине парольной фразы;
5. Свойство lowercase_letter содержит информацию, должны ли быть строчные буквы в парольной фразе;
6. Свойство uppercase_letter содержит информацию, должны ли быть заглавные буквы в парольной фразе;
7. Свойство digits содержит информацию, должны ли быть цифры в парольной фразе;
8. Свойство special_symbols содержит информацию, должны ли быть специальные символы (!@#$%№^&?./*\()-_=+[]{}"`~:;<>| или пробел) в парольной фразе;
9. Свойство delete_after содержит информацию, нужно ли удалить файл (при шифровании или расшифровании файла) после выполнения криптографической операции;

10. Свойство save_in_file содержит информацию, нужно ли сохранить результат выполнения криптографической операции в файл;
11. Свойство path_file_save содержит путь сохранения файла;
12. Свойство name_file_save содержит название файла при сохранении результата в файл;
13. Свойство passphrase содержит введенную пользователем в окне ввода парольной фразы для шифрования или расшифрования;
14. Свойство enc_state содержит информацию, ввел ли пользователь парольную фразу для шифрования;
15. Свойство dec_state содержит информацию, ввел ли пользователь парольную фразу для расшифрования.

Класс PassphraseEncDialog

Класс PassphraseEncDialog содержит информацию о окне ввода парольной фразы для шифрования: разметке расположения элементов на окне и действиях при нажатиях на кнопки.

Свойство и методы класса:

1. Методы __init__ инициализации класса содержит информацию о расположении элементов на окне и присваивает действия при нажатии на кнопки;
2. Метод save_click обрабатывает нажатие на кнопку «Ввод» в окне: сравнивает введенные пароли и проверяет пароль на соответствие ограничениям на парольную фразу;
3. Метод cancel_click обрабатывает нажатие на кнопку «Отмена» в окне: закрывает окно ввода парольной фразы для шифрования.

Класс PassphraseDecDialog

Класс PassphraseDecDialog содержит информацию о окне ввода парольной фразы для расшифрования: разметке расположения элементов на окне и действиях при нажатиях на кнопки.

Свойство и методы класса:

1. Методы __init__ инициализации класса содержит информацию о расположении элементов на окне и присваивает действия при нажатии на кнопки;
2. Метод save_click обрабатывает нажатие на кнопку «Ввод» в окне: передает полученную парольную фразу в глобальные переменные и закрывает окно;
3. Метод cancel_click обрабатывает нажатие на кнопку «Отмена» в окне: закрывает окно ввода парольной фразы для расшифрования.

Класс RC5

Класс RC5 содержит реализацию криптоалгоритма RC5: выполнения криптографических операций и шифрования и расшифрования текста и файлов.

Свойства и методы класса:

1. Метод __init__ инициализации класса присваивает свойствам значения и подготавливает криптоалгоритм для работы;
2. Свойство w содержит информацию о размере блока шифрования (16, 32 или 64);
3. Свойство R содержит информацию о количестве раундов шифрования (от 0 до 255);
4. Свойство key содержит ключ для шифрования или расшифрования;
5. Свойство mode содержит информацию о режиме шифрования (ecb, cbc, cfb или ofb);
6. Свойство salt содержит примесь для парольной фразы, необходимая для проверки правильности введенной парольной фразы при расшифровании;
7. Свойство T содержит вспомогательную переменную, равную $2 \cdot (R + 1)$;
8. Свойство w4 содержит вспомогательную переменную, равную $w \,//\, 4$;
9. Свойство w8 содержит вспомогательную переменную, равную количеству байт в размере блока шифрования $w \,//\, 8$;
10. Свойство mod содержит вспомогательную переменную, равную 2^w;
11. Свойство mask содержит маску для шифрования, равную mod – 1;
12. Свойство b содержит информацию о длине ключа;
13. Свойство S содержит таблицы расширенных ключей;
14. Метод get_iv генерирует вектор инициализации для режимов шифрования (cbc, cfb и ofb);
15. Метод xor применяет операцию XOR к полученным байтам;
16. Метод generate_key добавляет примесь к парольной фразе, после чего применяет функцию хеширования md5 к полученной строке и возвращает ключ необходимой длинны;
17. Метод __lshift применяет сдвиг влево на n бит;
18. Метод __rshift применяет сдвиг вправо на n бит;
19. Метод __const генерирует необходимые константы;
20. Метод __key_align разбивает ключ на слова;
21. Метод __key_extend строит таблицы расширенных ключей;
22. Метод __shuffle выполняет перемешивание;
23. Метод encrypt_block шифрует полученный блок;

24. Метод decrypt_block расшифровывает полученный блок;
25. Метод encrypt_bytes шифрует полученные байты;
26. Метод decrypt_bytes расшифровывает полученные байты;
27. Метод encrypt_file шифрует полученный файл;
28. Метод decrypt_file расшифровывает полученный файл.

3. Тестирование разработанной программы

Для проверки корректности работы программы проведем несколько тестов, в которых проверим работу криптоалгоритма на различных исходных данных с различными параметрами: шифрование и расшифрование текста и файлов, работа с разными режимами работы, длинной блока, числом раундов и длинной ключа для криптоалгоритма, а также с различными ограничениями на парольную фразу и сохранением результата работы криптоалгоритма в файл.

Тест №1

Проверим работу криптоалгоритма в режиме электронной кодовой книги (ECB), длинной блока 32 бита, 12 раундами и длинной ключа 128 бит. Ограничений на парольную фразу в данном тесте не будет, результат будет выводиться на экран.

Текст для проверки: «Текст для проверки работоспособности криптоалгоритма». Парольная фраза: «RC5». В результате на экране должен появиться зашифрованный текст, который при расшифровке даст исходный текст.

1. Выполним указанные настройки, введем текст для проверки и парольную фразу.

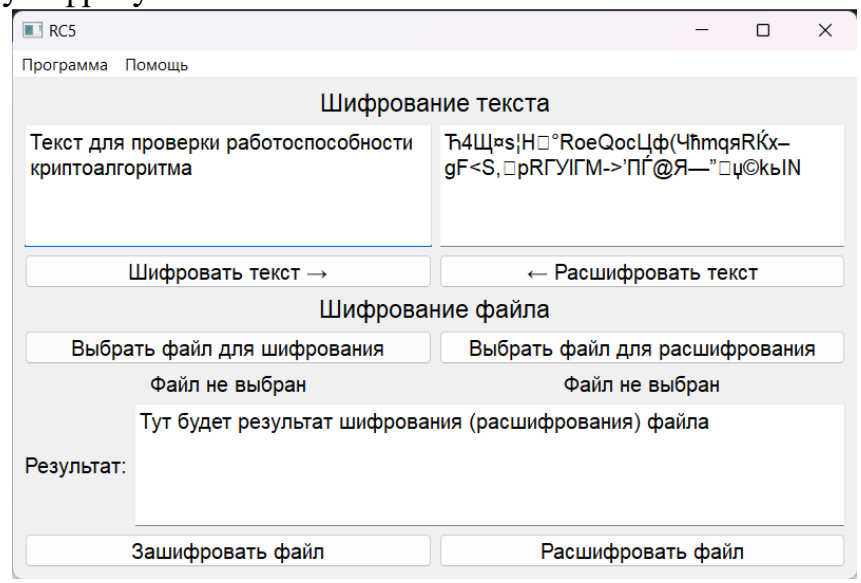

Рис. 3.1.1. Результат шифрования текста для проверки

2. Удалим текст для шифрования и попробуем расшифровать текст, полученный на предыдущем шаге.

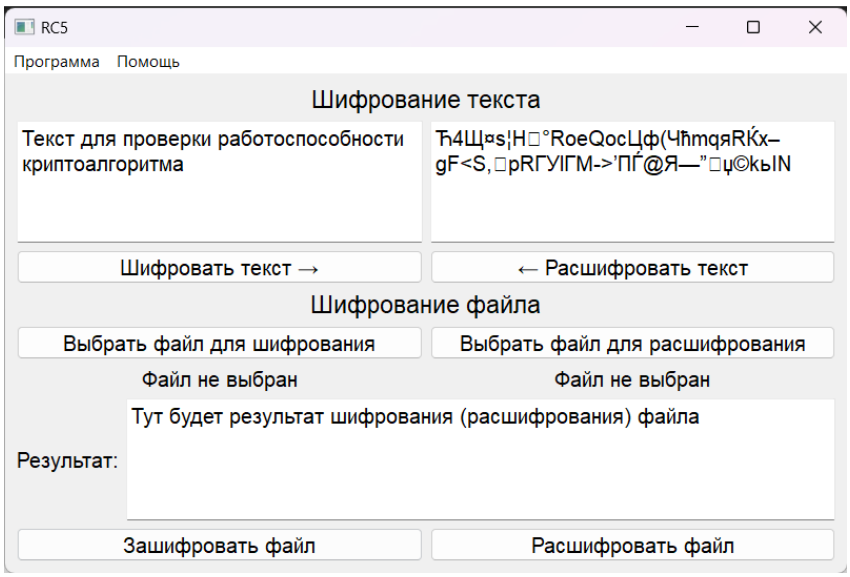

Рис. 3.1.2. Результат расшифрования текста для проверки

Как видно из приведенного примера, исходный текст был успешно восстановлен после расшифрования. Тест №1 пройден.

Тест №2

Проверим работу криптоалгоритма в режиме обратной связи по шифротексту (CFB), длинной блока 64 бита, 64 раундами и длинной ключа 256 бит. Ограничения на парольную фразу – наличие строчных букв и цифр, результат сохраняется в файл result.txt.

Текст для проверки: «Текст для второго теста». Парольная фраза: «rc5». В результате должен быть сохранен файл с шифром, который при расшифровке данного файла даст исходный текст.

1. Выполним указанные настройки, введем текст для проверки и парольную фразу.

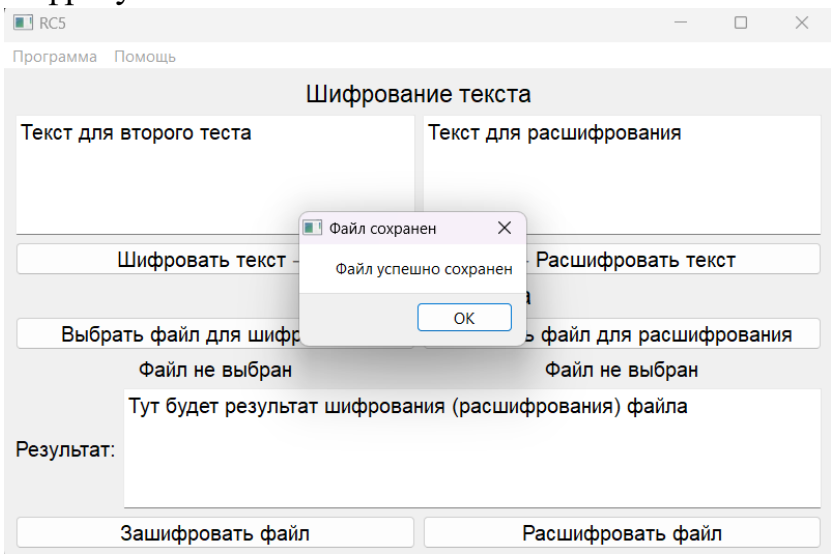

Рис. 3.2.1. Окно успешного сохранения файла

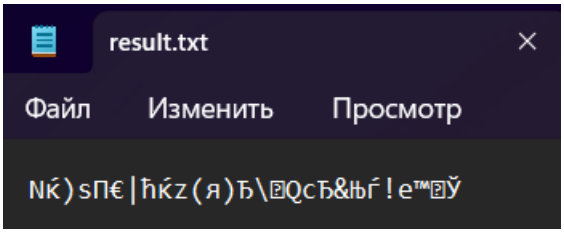

Рис. 3.2.2. Содержание сохраненного файла

2. Укажем как файл для расшифрования result.txt и выполним его расшифровку, результат выведем на экран.

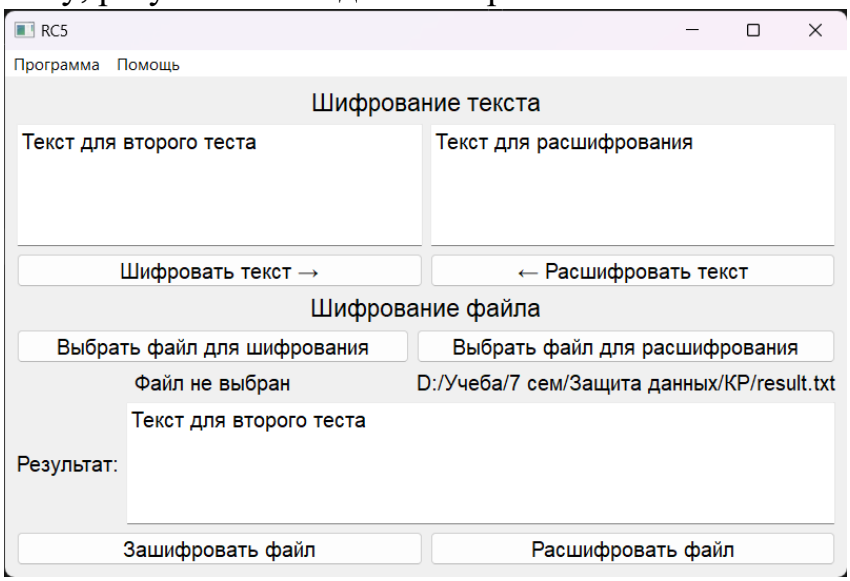

Рис. 3.2.3. Результат расшифрования файла

Как видно, результат расшифрования текста совпал с исходным текстом, поэтому тест №2 пройден.

Тест №3

Проверим работу криптоалгоритма в режиме обратной связи по выходу (OFB), длинной блока 16 бит, 255 раундами и длинной ключа 1000 бит. Ограничения на парольную фразу отсутствуют, текст изначально находится в файле test_text.txt, результат шифрования выводится на экран, после чего зашифрованный текст копируется в поле расшифровки текста и результат расшифровки выводится на экран.

Текст для проверки: «Текст для третьего теста». Парольная фраза: «testRC5». В результате должен быть получен зашифрованный текст, который при расшифровке даст исходный текст из файла.

1. Запишем текст для проверки в файл test_text.txt и выполним шифрование файла с выводом результата на экран.

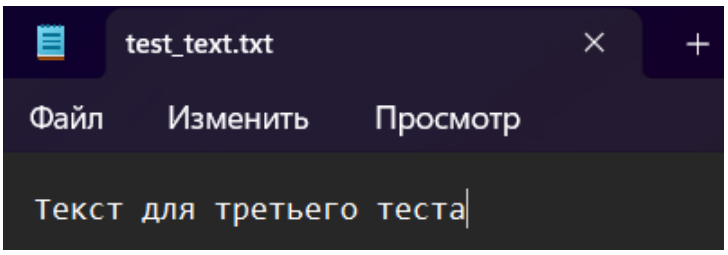

Рис. 3.3.1. Содержимое файла test_text.txt

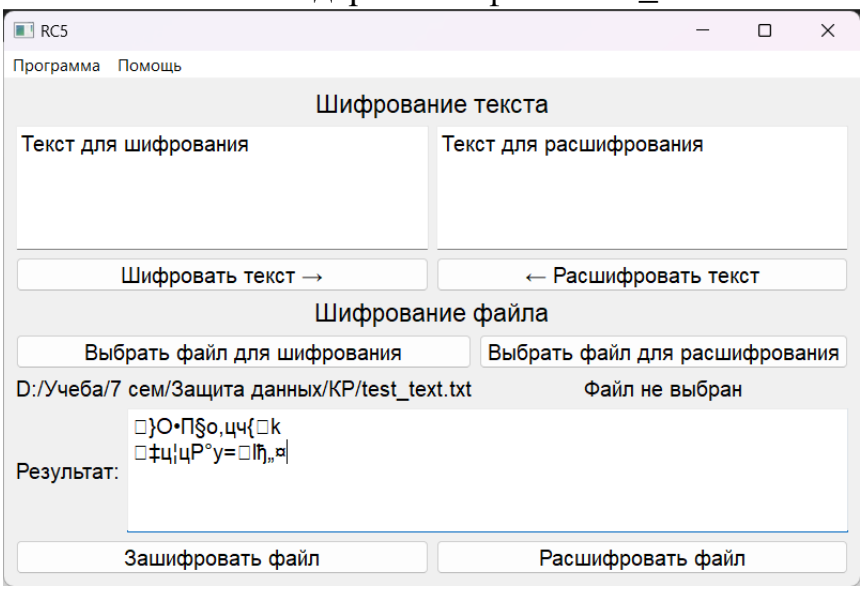

Рис. 3.3.2. Результат шифрования файла

2. Скопируем полученный на предыдущем шаге зашифрованный текст и скопируем в поле для расшифровки текста, после чего получим результат его расшифровки

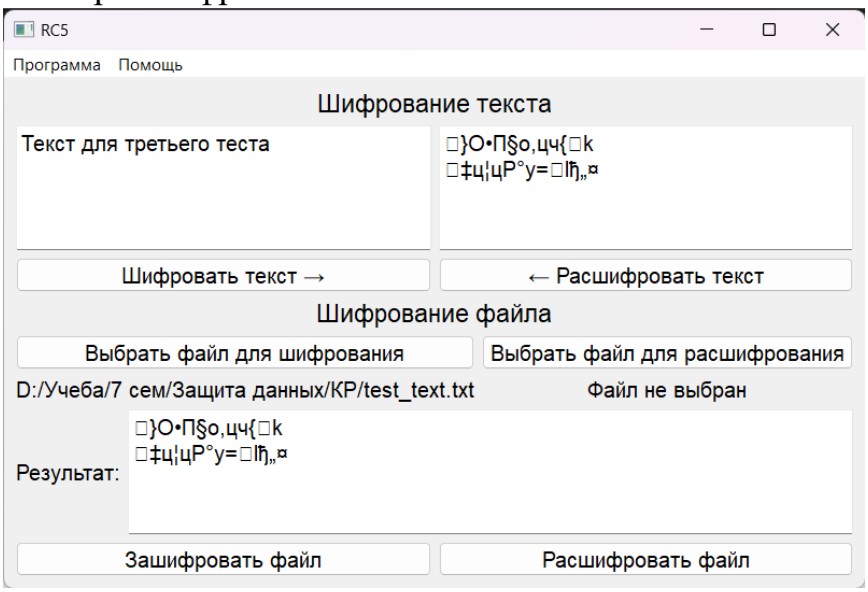

Рис. 3.3.3. Результат расшифровки текста

Из приведенного выше примера видно, что текст для шифрования совпал с результатом его расшифровки, из чего можно сделать вывод, что тест №3 пройден.

Тест №4

В этом тесте будет проведена проверка, правильно ли программа проверяет парольную фразу при расшифровке текста. В качестве режима шифрования будет выбран режим сцепления блоков шифротекста (CBC), длинна блока – 32 бита, число раундов – 12, длинна ключа – 128 бит. Ограничений на парольную фразу нет, результат выводится на экран.

Текст для проверки: «Текст для проверки парольной фразы», парольная фраза при шифровании – «rc5», парольная фраза при расшифровании – «RC5». В результате при расшифровке должно появиться окно, сообщающее, что была введена неверная парольная фраза при расшифровке.

1. Установим указанные настройки, введем текст для проверки и зашифруем текст с парольной фразой для шифрования.

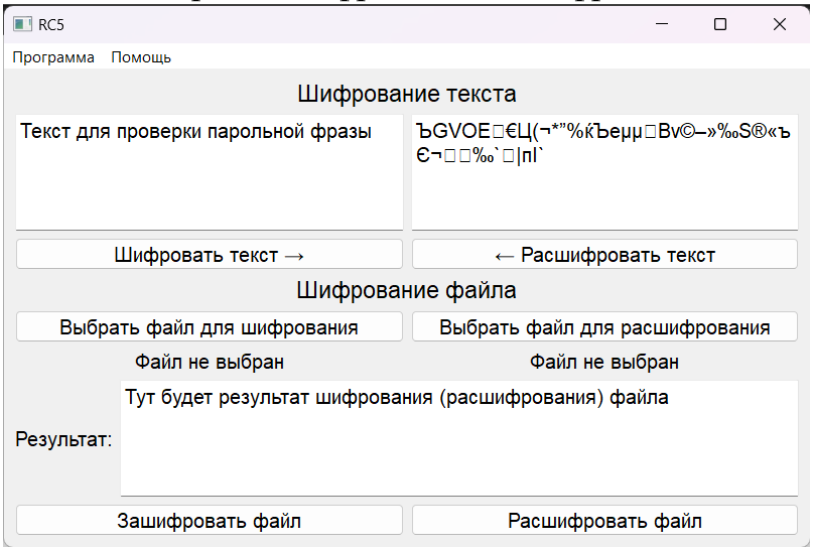

Рис. 3.4.1. Результат шифрования текста

2. Удалим текст для шифрования и расшифруем текст из предыдущего пункта с неверной парольной фразой.

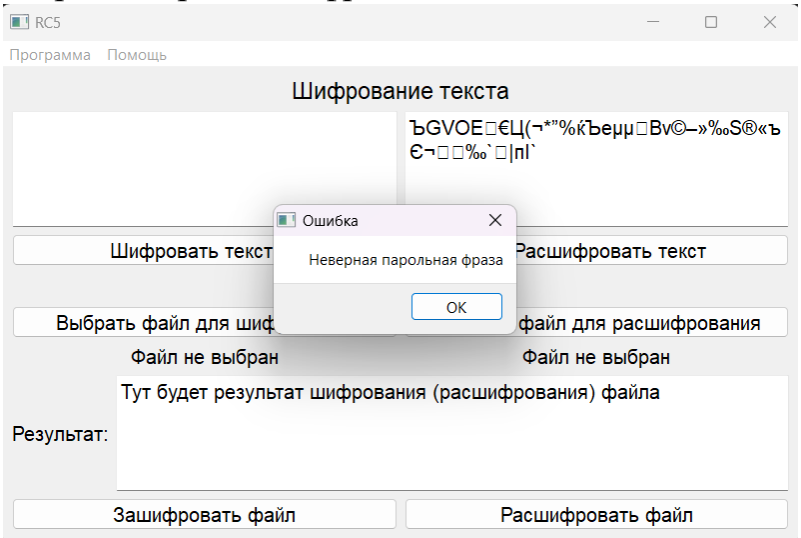

Рис. 3.4.2. Результат расшифровки с неверной парольной фразой

Из приведенного выше примера видно, что программа вывела результат, который можно ожидать при неверной парольной фразе. Тест №4 пройден.

Тест №5

Проведем тест на проверку ограничений на парольную фразу. Установим ограничение на парольную фразу – наличие заглавных букв, цифр, специальных символов и минимальной длинной 6 символов. Режим шифрования - режим электронной кодовой книги (ECB), длинна блока 32 бита, 12 раундов и длинна ключа 128 бит.

Текст для шифрования: «Текст для проверки ограничений на парольную фразу», парольная фраза, вводимая при шифровании – «Qwe#Rty». В результате ожидается, что программа выведет ошибку при вводе парольной фразы, сообщающую об отсутствии цифр.

1. Установим указанные настройки, введем текст для шифрования и нажмем на кнопку «Шифровать текст»

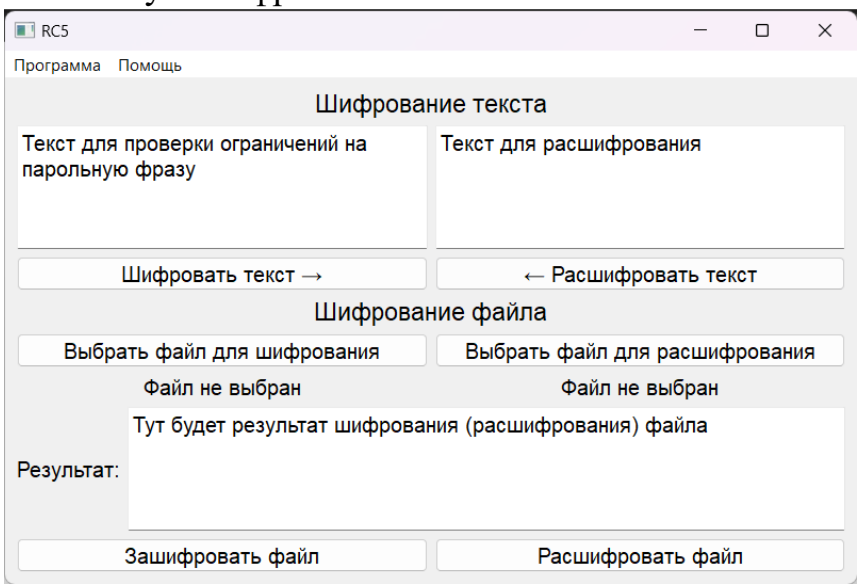

Рис. 3.5.1. Шифрование текста

2. Введем парольную фразу и посмотрим на результат

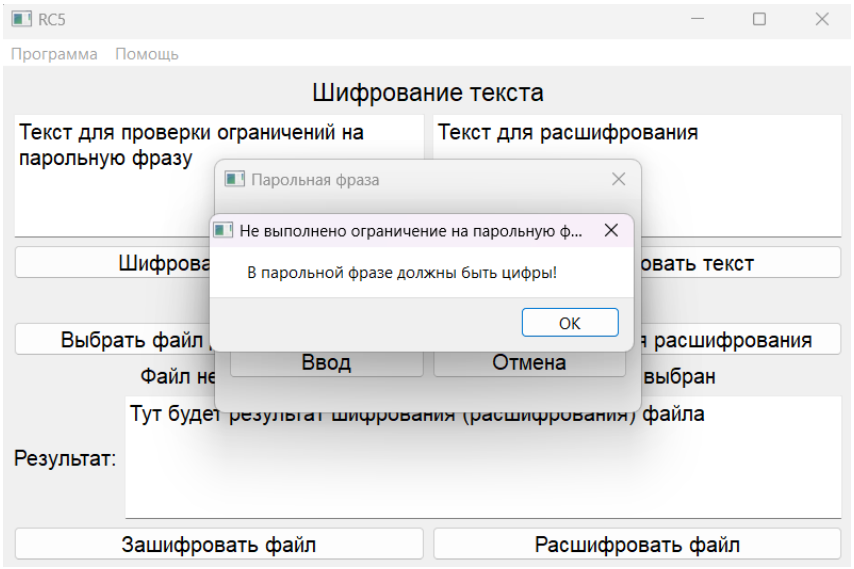

Рис. 3.5.2. Результат ввода парольной фразы

Из данного примера видно, что программа корректно обрабатывает ограничения на парольную фразу. Тест №5 пройден.

Из приведенных выше примеров видно, что программа корректно обрабатывает различные введенные данные и фактический результат совпадает с ожидаемым, из чего можно сделать вывод о том, что программа работает корректно.

Заключение

В результате выполнения курсовой работы в соответствии с целями и задачами были проделаны следующие действия:

- Изучен алгоритм работы RC5 и режимов работы криптоалгоритмов, проведен его анализ и описана пошаговая его реализация;
- Спроектирован и разработан интерфейс, реализующий поставленные требования к работе криптоалгоритма;
- Разработана программа на языке программирования Python с использованием библиотеки PyQt5, реализующая спроектированный интерфейс и криптоалгоритм RC5;
- Проведена отладка и комплексное тестирование программы, на основании чего сделан вывод о том, что программа работает корректно;
- Составлен отчет о проделанной работе.

Криптоалгоритм RC5 обладает сравнительно хорошей криптостойкостью после 15 раундов, что говорит о ценности данной работы. Таким образом, выполнение данной работы позволило мне расширить познания в области криптографии и получить навыки в реализации криптоалгоритмов на языке программирования Python.

Список литературы

1. RC5 // Википедия сайт. – URL: https://ru.wikipedia.org/wiki/RC5
2. Режим шифрования // Википедия сайт. – URL: https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F
3. Блочный шифр // Википедия сайт. – URL: https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80
4. Алгоритм шифрования RC5 и его реализация на Python // Хабр сайт. – URL: https://habr.com/ru/articles/267295/
5. PyQt5 Reference Guide // Riverbank Computing сайт. – URL:
6. https://www.riverbankcomputing.com/static/Docs/PyQt5/
7. Auto PY to EXE // pypi сайт. – URL: https://pypi.org/project/auto-py-to-exe/

Приложение 1. Код программы

Файл main.py

```
import sys
from MainWindow import MainWindow
from PyQt5.QtWidgets import QApplication


def main():
    app = QApplication(sys.argv)
    win = MainWindow()
    win.show()
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```

Файл MainWindow.py

```
from PyQt5.QtWidgets import (QTextEdit, QWidget, QPushButton, QFileDialog,
                             QMainWindow, QAction, QVBoxLayout,
                             QHBoxLayout, QMessageBox, QLabel)
from SettingsWindow import SettingsDialog
from PassphraseEncWindow import PassphraseEncDialog
from PassphraseDecWindow import PassphraseDecDialog
from AboutWindow import AboutDialog
from PyQt5.QtGui import QFont
from GlobalVariables import GlobalSettings
from PyQt5.QtCore import Qt
from RC5 import RC5
import os


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('RC5')
        self.resize(800, 500)

        self.file_encrypt = 0
        self.file_decrypt = 0

        GlobalSettings.blocks = 32
        GlobalSettings.mode = 'ecb'
        GlobalSettings.passphrase = ''
        GlobalSettings.key_len = 128
        GlobalSettings.path_file_save = ''
        GlobalSettings.name_file_save = 'result.txt'
        GlobalSettings.enc_state = False
        GlobalSettings.dec_state = False

        label_text = QLabel('Шифрование текста')
        label_text.setFont(QFont('Arial', 14))
        label_text.setAlignment(Qt.AlignHCenter)
        label_file = QLabel('Шифрование файла')
        label_file.setFont(QFont('Arial', 14))
        label_file.setAlignment(Qt.AlignHCenter)

        self.edit_encrypt_text = QTextEdit('Текст для шифрования')
        self.edit_encrypt_text.setFont(QFont('Arial', 12))
        self.edit_decrypt_text = QTextEdit('Текст для расшифрования')
        self.edit_decrypt_text.setFont(QFont('Arial', 12))
```

```
        button_encrypt_text = QPushButton('Шифровать текст →')
        button_encrypt_text.setFont(QFont('Arial', 12))
        button_encrypt_text.clicked.connect(self.click_encrypt_text)
        button_decrypt_text = QPushButton('← Расшифровать текст')
        button_decrypt_text.setFont(QFont('Arial', 12))
        button_decrypt_text.clicked.connect(self.click_decrypt_text)

        button_select_encrypt_file = QPushButton('Выбрать файл для
шифрования')
        button_select_encrypt_file.setFont(QFont('Arial', 12))

button_select_encrypt_file.clicked.connect(self.click_select_enc_file)
        button_select_decrypt_file = QPushButton('Выбрать файл для
расшифрования')
        button_select_decrypt_file.setFont(QFont('Arial', 12))

button_select_decrypt_file.clicked.connect(self.click_select_dec_file)

        self.label_select_encrypt_file = QLabel('Файл не выбран')
        self.label_select_encrypt_file.setFont(QFont('Arial', 12))
        self.label_select_encrypt_file.setAlignment(Qt.AlignHCenter)
        self.label_select_decrypt_file = QLabel('Файл не выбран')
        self.label_select_decrypt_file.setFont(QFont('Arial', 12))
        self.label_select_decrypt_file.setAlignment(Qt.AlignHCenter)

        label_result_file = QLabel('Результат:')
        label_result_file.setFont(QFont('Arial', 12))
        self.edit_result_file = QTextEdit('Тут будет результат шифрования
(расшифрования) файла')
        self.edit_result_file.setFont(QFont('Arial', 12))

        button_encrypt_file = QPushButton('Зашифровать файл')
        button_encrypt_file.setFont(QFont('Arial', 12))
        button_encrypt_file.clicked.connect(self.click_enc_file)
        button_decrypt_file = QPushButton('Расшифровать файл')
        button_decrypt_file.setFont(QFont('Arial', 12))
        button_decrypt_file.clicked.connect(self.click_dec_file)

        layout = QVBoxLayout()
        layout_text = QHBoxLayout()
        layout_text.addWidget(self.edit_encrypt_text)
        layout_text.addWidget(self.edit_decrypt_text)

        layout_buttons_text = QHBoxLayout()
        layout_buttons_text.addWidget(button_encrypt_text)
        layout_buttons_text.addWidget(button_decrypt_text)

        layout_encrypt_file = QVBoxLayout()
        layout_encrypt_file.addWidget(button_select_encrypt_file)
        layout_encrypt_file.addWidget(self.label_select_encrypt_file)
        layout_decrypt_file = QVBoxLayout()
        layout_decrypt_file.addWidget(button_select_decrypt_file)
        layout_decrypt_file.addWidget(self.label_select_decrypt_file)
        layout_file = QHBoxLayout()
        layout_file.addLayout(layout_encrypt_file)
        layout_file.addLayout(layout_decrypt_file)

        layout.addWidget(label_text)
        layout.addLayout(layout_text)

        layout.addLayout(layout_buttons_text)
        layout.addWidget(label_file)
```

```
        layout.addLayout(layout_file)
        layout_result_file = QHBoxLayout()
        layout_result_file.addWidget(label_result_file)
        layout_result_file.addWidget(self.edit_result_file)
        layout.addLayout(layout_result_file)
        layout_buttons_file = QHBoxLayout()
        layout_buttons_file.addWidget(button_encrypt_file)
        layout_buttons_file.addWidget(button_decrypt_file)
        layout.addLayout(layout_buttons_file)

        self.setLayout(layout)
        central_widget = QWidget()
        central_widget.setLayout(layout)
        self.setCentralWidget(central_widget)

        self.create_menu_bar()

    def click_enc_file(self):
        pew = PassphraseEncDialog()
        pew.exec_()
        rc5 = RC5(GlobalSettings.blocks, GlobalSettings.rounds,
GlobalSettings.passphrase,
                  GlobalSettings.key_len, GlobalSettings.mode)
        if not GlobalSettings.enc_state:
            return
        try:
            if GlobalSettings.save_in_file:
                try:
                    with open(GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save, 'w') as file:

file.write(rc5.encrypt_file(self.file_encrypt).decode(encoding='cp1251'))
                    QMessageBox.about(self, 'Файл сохранен', 'Файл успешно
сохранен')
                except FileNotFoundError:
                    QMessageBox.about(self, 'Ошибка', 'Ошибка при создании
файла')
                    return
            else:

self.edit_result_file.setText(rc5.encrypt_file(self.file_encrypt).decode(enco
ding='cp1251'))
            if GlobalSettings.delete_after \
                    and (GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save) != self.file_encrypt:
                os.remove(self.file_encrypt)
        except Exception:
            QMessageBox.about(self, 'Ошибка', 'Что-то пошло не так')

    def click_dec_file(self):
        pdw = PassphraseDecDialog()
        pdw.exec_()
        rc5 = RC5(GlobalSettings.blocks, GlobalSettings.rounds,
GlobalSettings.passphrase,
                  GlobalSettings.key_len, GlobalSettings.mode)
        if not GlobalSettings.dec_state:
            return
        try:
            if GlobalSettings.save_in_file:
                try:
                    with open(GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save, 'w') as file:
```

```
file.write(rc5.decrypt_file(self.file_decrypt).decode(encoding='cp1251'))
                    QMessageBox.about(self, 'Файл сохранен', 'Файл успешно
сохранен')
                except FileNotFoundError:
                    QMessageBox.about(self, 'Ошибка', 'Ошибка при создании
файла')
                    return
            else:

self.edit_result_file.setText(rc5.decrypt_file(self.file_decrypt).decode(enco
ding='cp1251'))
            if GlobalSettings.delete_after \
                    and (GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save) != self.file_decrypt:
                os.remove(self.file_decrypt)
        except Exception as exc:
            if exc.args[0] == '':
                QMessageBox.about(self, 'Ошибка', 'Что-то пошло не так')
            else:
                QMessageBox.about(self, 'Ошибка', exc.args[0])

    def click_encrypt_text(self):
        pew = PassphraseEncDialog()
        pew.exec_()
        rc5 = RC5(GlobalSettings.blocks, GlobalSettings.rounds,
GlobalSettings.passphrase,
                  GlobalSettings.key_len, GlobalSettings.mode)
        if not GlobalSettings.enc_state:
            return
        try:
            if GlobalSettings.save_in_file:
                try:
                    with open(GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save, 'w') as file:

file.write(rc5.encrypt_bytes(self.edit_encrypt_text.toPlainText().encode(enco
ding='cp1251'))
                                        .decode(encoding='cp1251'))
                    QMessageBox.about(self, 'Файл сохранен', 'Файл успешно
сохранен')
                except FileNotFoundError:
                    QMessageBox.about(self, 'Ошибка', 'Ошибка при создании
файла')
                    return
            else:
                self.edit_decrypt_text.setText(rc5.encrypt_bytes(

self.edit_encrypt_text.toPlainText().encode(encoding='cp1251')).decode(encodi
ng='cp1251'))
        except Exception:
            QMessageBox.about(self, 'Ошибка', 'Что-то пошло не так')

    def click_decrypt_text(self):
        pdw = PassphraseDecDialog()
        pdw.exec_()
        rc5 = RC5(GlobalSettings.blocks, GlobalSettings.rounds,
GlobalSettings.passphrase,
                  GlobalSettings.key_len, GlobalSettings.mode)
        if not GlobalSettings.dec_state:
            return
        try:
            if GlobalSettings.save_in_file:
                try:
```

```
                    with open(GlobalSettings.path_file_save + '/' +
GlobalSettings.name_file_save, 'w') as file:

file.write(rc5.decrypt_bytes(self.edit_decrypt_text.toPlainText().encode(enco
ding='cp1251'))
                                        .decode(encoding='cp1251'))
                    QMessageBox.about(self, 'Файл сохранен', 'Файл успешно
сохранен')
                except FileNotFoundError:
                    QMessageBox.about(self, 'Ошибка', 'Ошибка при создании
файла')
                    return
            else:
                self.edit_encrypt_text.setText(rc5.decrypt_bytes(

self.edit_decrypt_text.toPlainText().encode(encoding='cp1251')).decode(encodi
ng='cp1251'))
        except Exception as exc:
            if exc.args[0] == '':
                QMessageBox.about(self, 'Ошибка', 'Что-то пошло не так')
            else:
                QMessageBox.about(self, 'Ошибка', exc.args[0])

    def click_select_enc_file(self):
        try:
            self.file_encrypt = QFileDialog.getOpenFileName()[0]
            self.label_select_encrypt_file.setText(self.file_encrypt)
        except Exception:
            return

    def click_select_dec_file(self):
        try:
            self.file_decrypt = QFileDialog.getOpenFileName()[0]
            self.label_select_decrypt_file.setText(self.file_decrypt)
        except Exception:
            return

    def create_menu_bar(self):
        menu_bar = self.menuBar()

        file_menu = menu_bar.addMenu("Программа")
        settings_action = QAction('Настройки', self)
        settings_action.triggered.connect(self.settings_action)
        close_action = QAction('Выход', self)
        close_action.triggered.connect(self.close_action)

        file_menu.addAction(settings_action)
        file_menu.addAction(close_action)

        help_menu = menu_bar.addMenu("Помощь")
        about = QAction('О программе', self)
        about.triggered.connect(self.about_action)
        help_menu.addAction(about)

    @staticmethod
    def close_action():
        exit()

    @staticmethod
    def about_action():
        about = AboutDialog()
        about.exec_()
```

```
    @staticmethod
    def settings_action():
        settings = SettingsDialog()
        settings.exec_()
```

Файл AboutWindow.py

```
from PyQt5.QtWidgets import QDialog, QLabel, QVBoxLayout, QPushButton
from PyQt5.QtGui import QFont
from PyQt5.QtCore import Qt


class AboutDialog(QDialog):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(self.windowFlags() &
(~Qt.WindowContextHelpButtonHint))
        self.setWindowTitle("О программе")
        self.resize(400, 300)

        layout = QVBoxLayout()

        label1 = QLabel("Программная реализация\nкриптоалгоиртма RC5")
        label1.setFont(QFont('Arial', 14))
        label1.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
        label2 = QLabel("Курсовая работа по Защите данных\n"
                        "Работу выполнил студент группы А-13б-20\n"
                        "Бегунов Никита Сергеевич\n"
                        "Вариант №21")
        label2.setFont(QFont('Arial', 10))
        layout.addWidget(label1)
        layout.addWidget(label2)

        close_button = QPushButton("Закрыть")
        close_button.clicked.connect(self.close)
        layout.addWidget(close_button)

        self.setLayout(layout)
```

Файл PassphraseDecWindow.py

```
from PyQt5.QtWidgets import (QDialog, QLabel, QVBoxLayout, QPushButton,
                             QHBoxLayout, QLineEdit)
from PyQt5.QtGui import QFont
from GlobalVariables import GlobalSettings
from PyQt5.QtCore import Qt


class PassphraseDecDialog(QDialog):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(self.windowFlags() &
(~Qt.WindowContextHelpButtonHint))
        self.setWindowTitle("Парольная фраза")
        self.resize(400, 200)

        layout = QVBoxLayout()

        label_passphrase = QLabel('Парольная фраза:')
        label_passphrase.setFont(QFont('Arial', 12))
        self.edit_passphrase = QLineEdit('')
        self.edit_passphrase.setFont(QFont('Arial', 12))
        self.edit_passphrase.setEchoMode(QLineEdit.Password)
```

```
        layout_passphrase = QHBoxLayout()
        layout_passphrase.addWidget(label_passphrase)
        layout_passphrase.addWidget(self.edit_passphrase)

        save_button = QPushButton("Ввод")
        save_button.setFont(QFont('Arial', 12))
        save_button.clicked.connect(self.save_click)
        cancel_button = QPushButton('Отмена')
        cancel_button.setFont(QFont('Arial', 12))
        cancel_button.clicked.connect(self.cancel_click)
        layout_buttons = QHBoxLayout()
        layout_buttons.addWidget(save_button)
        layout_buttons.addWidget(cancel_button)

        layout.addLayout(layout_passphrase)
        layout.addLayout(layout_buttons)

        self.setLayout(layout)

    def save_click(self):
        GlobalSettings.passphrase = self.edit_passphrase.text()
        GlobalSettings.dec_state = True
        self.close()

    def cancel_click(self):
        GlobalSettings.dec_state = False
        self.close()
```

Файл PassphraseEncWindow.py

```
from PyQt5.QtWidgets import (QDialog, QLabel, QVBoxLayout, QPushButton,
                             QHBoxLayout, QLineEdit, QMessageBox)
from PyQt5.QtGui import QFont
from GlobalVariables import GlobalSettings
from PyQt5.QtCore import Qt


class PassphraseEncDialog(QDialog):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(self.windowFlags() &
(~Qt.WindowContextHelpButtonHint))
        self.setWindowTitle("Парольная фраза")
        self.resize(400, 200)

        layout = QVBoxLayout()

        label_passphrase = QLabel('Парольная фраза:')
        label_passphrase.setFont(QFont('Arial', 12))
        self.edit_passphrase = QLineEdit('')
        self.edit_passphrase.setFont(QFont('Arial', 12))
        self.edit_passphrase.setEchoMode(QLineEdit.Password)
        layout_passphrase = QHBoxLayout()
        layout_passphrase.addWidget(label_passphrase)
        layout_passphrase.addWidget(self.edit_passphrase)

        label_comfirm = QLabel('Подтверждение:    ')
        label_comfirm.setFont(QFont('Arial', 12))
        self.edit_comfirm = QLineEdit('')
        self.edit_comfirm.setFont(QFont('Arial', 12))
        self.edit_comfirm.setEchoMode(QLineEdit.Password)
        layout_comfirm = QHBoxLayout()
        layout_comfirm.addWidget(label_comfirm)
```

```
        layout_comfirm.addWidget(self.edit_comfirm)

        save_button = QPushButton("Ввод")
        save_button.setFont(QFont('Arial', 12))
        save_button.clicked.connect(self.save_click)
        cancel_button = QPushButton('Отмена')
        cancel_button.setFont(QFont('Arial', 12))
        cancel_button.clicked.connect(self.cancel_click)
        layout_buttons = QHBoxLayout()
        layout_buttons.addWidget(save_button)
        layout_buttons.addWidget(cancel_button)

        layout.addLayout(layout_passphrase)
        layout.addLayout(layout_comfirm)
        layout.addLayout(layout_buttons)

        self.setLayout(layout)

    def save_click(self):
        if (pass_phrase := self.edit_passphrase.text()) !=
self.edit_comfirm.text():
            QMessageBox.about(self, 'Ошибка', 'Парольные фразы должны
совпадать!')
            return
        if GlobalSettings.lowercase_letter and pass_phrase.upper() ==
pass_phrase:
            QMessageBox.about(self, 'Не выполнено ограничение на парольную
фразу',
                              'В парольной фразе должны быть строчные
символы!')
            return
        if GlobalSettings.uppercase_letter and pass_phrase.lower() ==
pass_phrase:
            QMessageBox.about(self, 'Не выполнено ограничение на парольную
фразу',
                              'В парольной фразе должны быть заглавные
символы!')
            return
        if GlobalSettings.digits and not any(ch.isdigit() for ch in
pass_phrase):
            QMessageBox.about(self, 'Не выполнено ограничение на парольную
фразу',
                              'В парольной фразе должны быть цифры!')
            return
        if GlobalSettings.special_symbols and not any(ch in pass_phrase for
ch in '!@#$%№^&?./*\()-_=+[]{}"`~:;<>| '):
            QMessageBox.about(self, 'Не выполнено ограничение на парольную
фразу',
                              'В парольной фразе должны быть специальные
символы!')
            return
        if len(pass_phrase) < GlobalSettings.min_len:
            QMessageBox.about(self, 'Не выполнено ограничение на парольную
фразу',
                              'Длинна парольной фразы меньше допустимой!')
            return
        GlobalSettings.passphrase = pass_phrase
        GlobalSettings.enc_state = True
        self.close()

    def cancel_click(self):
        GlobalSettings.enc_state = False
        self.close()
```

Файл GlobalVariables.py

```
class GlobalSettings:
    blocks = 32
    rounds = 12
    min_len = 0
    mode = ''
    key_len = 0
    lowercase_letter = False
    uppercase_letter = False
    digits = False
    special_symbols = False
    delete_after = False
    save_in_file = False
    path_file_save = ''
    name_file_save = 'result.txt'
    passphrase = ''
    enc_state = False
    dec_state = False
```

Файл SettingsWindow.py

```
from PyQt5.QtWidgets import (QDialog, QLabel, QVBoxLayout, QPushButton,
                             QRadioButton, QHBoxLayout, QCheckBox,
                             QLineEdit, QMessageBox, QFileDialog,
QButtonGroup)
from PyQt5.QtGui import QFont
from GlobalVariables import GlobalSettings
from PyQt5.QtCore import Qt


class SettingsDialog(QDialog):
    def __init__(self):
        super().__init__()

        self.setWindowFlags(self.windowFlags() &
(~Qt.WindowContextHelpButtonHint))
        self.setWindowTitle("Настройки")
        self.resize(400, 400)

        self.path_file_save = GlobalSettings.path_file_save

        layout = QVBoxLayout()

        label_mode = QLabel("Режим работы криптоалгоритма:")
        label_mode.setFont(QFont('Arial', 12))

        layout_radiobuttons_mods = QVBoxLayout()

        self.group_radiobutton_mods = QButtonGroup(exclusive=True)
        radiobutton_mods = QRadioButton('Режим электронной кодовой книги
(ECB)')
        radiobutton_mods.setFont(QFont('Arial', 12))
        radiobutton_mods.setChecked(True if GlobalSettings.mode == 'ecb' else
False)
        radiobutton_mods.mode = 'ecb'
        layout_radiobuttons_mods.addWidget(radiobutton_mods)
        self.group_radiobutton_mods.addButton(radiobutton_mods)

        radiobutton_mods = QRadioButton('Режим сцепления блоков шифротекста
(CBC)')
        radiobutton_mods.setFont(QFont('Arial', 12))
        radiobutton_mods.setChecked(True if GlobalSettings.mode == 'cbc' else
False)
```

```
        radiobutton_mods.mode = 'cbc'
        layout_radiobuttons_mods.addWidget(radiobutton_mods)
        self.group_radiobutton_mods.addButton(radiobutton_mods)

        radiobutton_mods = QRadioButton('Режим обратной связи по шифротексту
(CFB)')
        radiobutton_mods.setFont(QFont('Arial', 12))
        radiobutton_mods.setChecked(True if GlobalSettings.mode == 'cfb' else
False)
        radiobutton_mods.mode = 'cfb'
        layout_radiobuttons_mods.addWidget(radiobutton_mods)
        self.group_radiobutton_mods.addButton(radiobutton_mods)

        radiobutton_mods = QRadioButton('Режим обратной связи по выходу
(OFB)')
        radiobutton_mods.setFont(QFont('Arial', 12))
        radiobutton_mods.setChecked(True if GlobalSettings.mode == 'ofb' else
False)
        radiobutton_mods.mode = 'ofb'
        layout_radiobuttons_mods.addWidget(radiobutton_mods)
        self.group_radiobutton_mods.addButton(radiobutton_mods)

self.group_radiobutton_mods.buttonClicked.connect(self.radiobutton_mods_click
ed)

        label_settings_crypto = QLabel('Настройки криптоалгоритма')
        label_settings_crypto.setFont(QFont('Arial', 14))

        layout_radiobuttons_blocks = QHBoxLayout()

        label_blocks = QLabel('Длинна блока:')
        label_blocks.setFont(QFont('Arial', 12))
        layout_radiobuttons_blocks.addWidget(label_blocks)

        self.group_radiobutton_blocks = QButtonGroup(exclusive=True)
        radiobutton_blocks = QRadioButton('16')
        radiobutton_blocks.setFont(QFont('Arial', 12))
        radiobutton_blocks.setChecked(True if GlobalSettings.blocks == 16
else False)
        radiobutton_blocks.blocks = 16
        layout_radiobuttons_blocks.addWidget(radiobutton_blocks)
        self.group_radiobutton_blocks.addButton(radiobutton_blocks)

        radiobutton_blocks = QRadioButton('32')
        radiobutton_blocks.setFont(QFont('Arial', 12))
        radiobutton_blocks.setChecked(True if GlobalSettings.blocks == 32
else False)
        radiobutton_blocks.blocks = 32
        layout_radiobuttons_blocks.addWidget(radiobutton_blocks)
        self.group_radiobutton_blocks.addButton(radiobutton_blocks)

        radiobutton_blocks = QRadioButton('64')
        radiobutton_blocks.setFont(QFont('Arial', 12))
        radiobutton_blocks.setChecked(True if GlobalSettings.blocks == 64
else False)
        radiobutton_blocks.blocks = 64
        layout_radiobuttons_blocks.addWidget(radiobutton_blocks)
        self.group_radiobutton_blocks.addButton(radiobutton_blocks)

self.group_radiobutton_blocks.buttonClicked.connect(self.radiobutton_blocks_c
licked)

        layout_num_rounds = QHBoxLayout()
```

```
        label_num_rounds = QLabel('Число раундов:')
        label_num_rounds.setFont(QFont('Arial', 12))
        layout_num_rounds.addWidget(label_num_rounds)
        self.edit_num_rounds = QLineEdit(str(GlobalSettings.rounds))
        self.edit_num_rounds.setFont(QFont('Arial', 12))
        layout_num_rounds.addWidget(self.edit_num_rounds)

        layout_key_len = QHBoxLayout()
        label_key_len = QLabel('Длинна ключа:')
        label_key_len.setFont(QFont('Arial', 12))
        layout_key_len.addWidget(label_key_len)
        self.edit_key_len = QLineEdit(str(GlobalSettings.key_len))
        self.edit_key_len.setFont(QFont('Arial', 12))
        layout_key_len.addWidget(self.edit_key_len)

        label_restrictions = QLabel('Ограничения на парольную фразу')
        label_restrictions.setFont(QFont('Arial', 14))

        label_min_len = QLabel('Минимальная длинна:')
        label_min_len.setFont(QFont('Arial', 12))
        self.edit_min_len = QLineEdit(str(GlobalSettings.min_len))
        self.edit_min_len.setFont(QFont('Arial', 12))
        self.checkbox_lowercase = QCheckBox('Строчные буквы')
        self.checkbox_lowercase.setFont(QFont('Arial', 12))

self.checkbox_lowercase.setCheckState(GlobalSettings.lowercase_letter)
        self.checkbox_uppercase = QCheckBox('Заглавные буквы')
        self.checkbox_uppercase.setFont(QFont('Arial', 12))

self.checkbox_uppercase.setCheckState(GlobalSettings.uppercase_letter)
        self.checkbox_digits = QCheckBox('Цифры')
        self.checkbox_digits.setFont(QFont('Arial', 12))
        self.checkbox_digits.setCheckState(GlobalSettings.digits)
        self.checkbox_special_symbols = QCheckBox('Специальные символы')
        self.checkbox_special_symbols.setFont(QFont('Arial', 12))

self.checkbox_special_symbols.setCheckState(GlobalSettings.special_symbols)
        self.checkbox_delete_after = QCheckBox('Удалить файл после операции')
        self.checkbox_delete_after.setFont(QFont('Arial', 12))
        self.checkbox_delete_after.setCheckState(GlobalSettings.delete_after)
        self.checkbox_save_result_in_file = QCheckBox('Сохранить результат в
файл')
        self.checkbox_save_result_in_file.setFont(QFont('Arial', 12))

self.checkbox_save_result_in_file.setCheckState(GlobalSettings.save_in_file)

        layout_path_file_save = QHBoxLayout()
        label_path_file_save = QLabel('Путь сохраниения файла:')
        label_path_file_save.setFont(QFont('Arial', 12))
        layout_path_file_save.addWidget(label_path_file_save)
        btn_path_file_save = QPushButton('Выбрать путь')
        btn_path_file_save.setFont(QFont('Arial', 12))
        btn_path_file_save.clicked.connect(self.btn_path_file_save_click)
        layout_path_file_save.addWidget(btn_path_file_save)
        self.label_path_file_save_global = QLabel(f'Выбран путь:
{GlobalSettings.path_file_save}'
                                                  if
GlobalSettings.path_file_save != '' else 'Путь не выбран')
        self.label_path_file_save_global.setFont(QFont('Arial', 12))

        layout_name_file_save = QHBoxLayout()
        label_name_file_save = QLabel('Имя файла для сохранения:')
        label_name_file_save.setFont(QFont('Arial', 12))
```

```
        layout_name_file_save.addWidget(label_name_file_save)
        self.edit_name_file_save = QLineEdit(GlobalSettings.name_file_save)
        self.edit_name_file_save.setFont(QFont('Arial', 12))
        layout_name_file_save.addWidget(self.edit_name_file_save)

        save_button = QPushButton("Сохранить")
        save_button.setFont(QFont('Arial', 12))
        save_button.clicked.connect(self.save_click)

        layout.addWidget(label_settings_crypto)
        layout.addWidget(label_mode)
        layout.addLayout(layout_radiobuttons_mods)
        layout.addLayout(layout_radiobuttons_blocks)
        layout.addLayout(layout_num_rounds)
        layout.addLayout(layout_key_len)
        layout.addWidget(label_restrictions)
        layout_min_len = QHBoxLayout()
        layout_min_len.addWidget(label_min_len)
        layout_min_len.addWidget(self.edit_min_len)
        layout.addLayout(layout_min_len)
        layout.addWidget(self.checkbox_lowercase)
        layout.addWidget(self.checkbox_uppercase)
        layout.addWidget(self.checkbox_digits)
        layout.addWidget(self.checkbox_special_symbols)
        layout.addWidget(self.checkbox_delete_after)
        layout.addWidget(self.checkbox_save_result_in_file)
        layout.addLayout(layout_path_file_save)
        layout.addWidget(self.label_path_file_save_global)
        layout.addLayout(layout_name_file_save)
        layout.addWidget(save_button)

        self.setLayout(layout)

    def save_click(self):
        try:
            rounds = int(self.edit_num_rounds.text())
            if 0 <= rounds <= 255:
                GlobalSettings.rounds = rounds
            else:
                raise Exception('Неверное значение')
        except Exception:
            QMessageBox.about(self, 'Ошибка', 'Неправильное значение в поле
количества раундов\n'
                                              'Количество раудов должно быть
от 0 до 255')
            return
        try:
            GlobalSettings.min_len = int(self.edit_min_len.text())
        except Exception:
            QMessageBox.about(self, 'Ошибка', 'Неправильное значение в поле
минимальной длинны')
            return
        try:
            key_len = int(self.edit_key_len.text())
            if 0 <= key_len <= 2040:
                GlobalSettings.key_len = key_len
            else:
                raise Exception('Неверное значение')
        except Exception:
            QMessageBox.about(self, 'Ошибка', 'Неправильное значение в поле
длинны ключа\n'
                                              'Длинна ключа должна быть от 0
до 2040')
```

```
            return
        GlobalSettings.lowercase_letter =
self.checkbox_lowercase.checkState()
        GlobalSettings.uppercase_letter =
self.checkbox_uppercase.checkState()
        GlobalSettings.digits = self.checkbox_digits.checkState()
        GlobalSettings.special_symbols =
self.checkbox_special_symbols.checkState()
        GlobalSettings.delete_after = self.checkbox_delete_after.checkState()
        GlobalSettings.save_in_file =
self.checkbox_save_result_in_file.checkState()
        GlobalSettings.path_file_save = self.path_file_save
        GlobalSettings.name_file_save = self.edit_name_file_save.text()
        if GlobalSettings.save_in_file and (GlobalSettings.path_file_save ==
'' or GlobalSettings.name_file_save == ''):
            QMessageBox.about(self, 'Ошибка', 'Не выбран путь или имя
сохранения файла')
            return
        if GlobalSettings.save_in_file and
GlobalSettings.name_file_save.find('.') == -1:
            QMessageBox.about(self, 'Ошибка', 'Неправильно имя сохранения
файла')
            return
        self.close()

    @staticmethod
    def radiobutton_blocks_clicked(btn):
        GlobalSettings.blocks = btn.blocks

    @staticmethod
    def radiobutton_mods_clicked(btn):
        GlobalSettings.mode = btn.mode

    def btn_path_file_save_click(self):
        self.path_file_save = QFileDialog.getExistingDirectory()
        self.label_path_file_save_global.setText(f'Выбран путь:
{self.path_file_save}'
                                                 if self.path_file_save != ''
else 'Путь не выбран')
```

Файл RC5.py

```
from hashlib import md5


class RC5:
    def __init__(self, w, R, passphrase, key_len=128, mode='ecb'):
        self.w = w  # block size (32, 64 or 128 bits)
        self.R = R  # number of rounds (0 to 255)
        self.key = self.generate_key(passphrase=passphrase, key_len=key_len)
# key (0 to 2040 bits)
        self.mode = mode

        self.salt = "1234".encode(encoding='cp1251')  # salt for passphrase
verification
        self.T = 2 * (R + 1)
        self.w4 = w // 4
        self.w8 = w // 8
        self.mod = 2 ** self.w
        self.mask = self.mod - 1
        self.b = len(self.key)

        self.__key_align()
        self.__key_extend()
```

```
        self.__shuffle()

    def get_iv(self):
        if self.w == 16:
            return 'абвг'.encode(encoding='cp1251')
        elif self.w == 32:
            return 'абвгдежз'.encode(encoding='cp1251')
        elif self.w == 64:
            return 'абвгдежзеклмопрс'.encode(encoding='cp1251')

    @staticmethod
    def xor(x, y):
        return bytes(a ^ b for a, b in zip(x, y))

    @staticmethod
    def generate_key(passphrase, key_len):
        passphrase += 'qWeR123АбВ'
        hash = md5(passphrase.encode(encoding='cp1251')).hexdigest()
        key = ''
        for elem in hash:
            x = bin(elem.encode(encoding='cp1251')[0])[2:]
            if len(x) < 8:
                temp = '0' * (8 - len(x)) + x
            else:
                temp = x
            key += temp
        if len(key) < key_len:
            key = '0' * (key_len - len(key)) + key
        key = key[:key_len]
        if len(key) % 8 != 0:
            key = '0' * (8 - (len(key) % 8)) + key
        return bytes(int(key[i:i + 8], 2) for i in range(0, len(key), 8))

    def __lshift(self, val, n):
        n %= self.w
        return ((val << n) & self.mask) | ((val & self.mask) >> (self.w - n))

    def __rshift(self, val, n):
        n %= self.w
        return ((val & self.mask) >> n) | (val << (self.w - n) & self.mask)

    def __const(self):  # constants generation
        if self.w == 16:
            return 0xB7E1, 0x9E37  # return P, Q values
        elif self.w == 32:
            return 0xB7E15163, 0x9E3779B9
        elif self.w == 64:
            return 0xB7E151628AED2A6B, 0x9E3779B97F4A7C15

    def __key_align(self):
        if self.b == 0:  # key is empty
            self.c = 1
        elif self.b % self.w8:
            self.key += b'\x00' * (self.w8 - self.b % self.w8)  # fill key
with \x00 bytes
            self.b = len(self.key)
            self.c = self.b // self.w8
        else:
            self.c = self.b // self.w8
        L = [0] * self.c
        for i in range(self.b - 1, -1, -1):
            L[i // self.w8] = (L[i // self.w8] << 8) + self.key[i]
        self.L = L
```

```
    def __key_extend(self):
        P, Q = self.__const()
        self.S = [(P + i * Q) % self.mod for i in range(self.T)]

    def __shuffle(self):
        i, j, A, B = 0, 0, 0, 0
        for k in range(3 * max(self.c, self.T)):
            A = self.S[i] = self.__lshift((self.S[i] + A + B), 3)
            B = self.L[j] = self.__lshift((self.L[j] + A + B), A + B)
            i = (i + 1) % self.T
            j = (j + 1) % self.c

    def encrypt_block(self, data):
        A = int.from_bytes(data[:self.w8], byteorder='little')
        B = int.from_bytes(data[self.w8:], byteorder='little')
        A = (A + self.S[0]) % self.mod
        B = (B + self.S[1]) % self.mod
        for i in range(1, self.R + 1):
            A = (self.__lshift((A ^ B), B) + self.S[2 * i]) % self.mod
            B = (self.__lshift((A ^ B), A) + self.S[2 * i + 1]) % self.mod
        return (A.to_bytes(self.w8, byteorder='little')
                + B.to_bytes(self.w8, byteorder='little'))

    def decrypt_block(self, data):
        A = int.from_bytes(data[:self.w8], byteorder='little')
        B = int.from_bytes(data[self.w8:], byteorder='little')
        for i in range(self.R, 0, -1):
            B = self.__rshift(B - self.S[2 * i + 1], A) ^ A
            A = self.__rshift(A - self.S[2 * i], B) ^ B
        B = (B - self.S[1]) % self.mod
        A = (A - self.S[0]) % self.mod
        return (A.to_bytes(self.w8, byteorder='little')
                + B.to_bytes(self.w8, byteorder='little'))

    def encrypt_bytes(self, data):
        res, run = b'', True
        data = self.salt + data
        if self.mode == 'ecb':
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    data = data.ljust(self.w4, b'\x00')
                    run = False
                res += self.encrypt_block(temp)
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'cbc':
            pred = self.get_iv()
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    data = data.ljust(self.w4, b'\x00')
                    run = False
                pred = self.encrypt_block(self.xor(temp, pred))
                res += pred
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'cfb':
            pred = self.get_iv()
            while run:
```

```
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    data = data.ljust(self.w4, b'\x00')
                    run = False
                pred = self.xor(temp, self.encrypt_block(pred))
                res += pred
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'ofb':
            pred = self.get_iv()
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    data = data.ljust(self.w4, b'\x00')
                    run = False
                pred = self.encrypt_block(pred)
                res += self.xor(temp, pred)
                data = data[self.w4:]
                if not data:
                    break
        return res

    def decrypt_bytes(self, data):
        res, run = b'', True
        if self.mode == 'ecb':
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    run = False
                res += self.decrypt_block(temp)
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'cbc':
            pred = self.get_iv()
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    run = False
                res += self.xor(self.decrypt_block(temp), pred)
                pred = temp
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'cfb':
            pred = self.get_iv()
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    run = False
                pred = self.encrypt_block(pred)
                res += self.xor(temp, pred)
                pred = temp
                data = data[self.w4:]
                if not data:
                    break
        elif self.mode == 'ofb':
            pred = self.get_iv()
            while run:
                temp = data[:self.w4]
                if len(temp) != self.w4:
                    run = False
```

```
                pred = self.encrypt_block(pred)
                res += self.xor(temp, pred)
                data = data[self.w4:]
                if not data:
                    break
        if res[:len(self.salt)] == self.salt:
            res = res[len(self.salt):]
        else:
            raise Exception('Неверная парольная фраза')
        return res

    def encrypt_file(self, inp_file_name):
        with open(inp_file_name, 'r', encoding='utf-8') as file:
            return self.encrypt_bytes(file.read().encode(encoding='cp1251'))

    def decrypt_file(self, inp_file_name):
        with open(inp_file_name, 'r') as file:
            return self.decrypt_bytes(file.read().encode(encoding='cp1251'))
```

Приложение 2. Файл setup

Файл setup.ps1

```
python3 -m pip install pyqt5
```

Приложение 3. README с инструкцией по установке

Файл README.md

```
    Курсовая работа по защите данных по теме "Программная реализация криптоалгоритма RC5"

    Работу выполнил студент 4 курса "НИУ МЭИ" группы А-13б-20 Бегунов Никита Сергеевич


    ###### Инструкция по установке ######

    1. Установить Python актуальной версии (3.11);

    1. Запустить скрипт setup.ps1 (ПКМ -> выполнить с помощью PowerShell);

    2. Запустить файл RC5.exe.


    ###### Информация о файлах в папке

    1. Папка _internal содержит файлы, необходимые для запуска exe-файла RC5.exe;

    2. Папка code содержит исходный код проекта;

    3. Приложение RC5.exe для запуска программы;

    4. Этот файл README.md с информацией о программе;

    5. Скрипт PowerShell для установки библиотеки, необходимой для работы программы.
```