

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИСКУССТВЕННОГО
ИНТЕЛЛЕКТА

КУРСОВАЯ РАБОТА

По программе «Защита данных»

Тема: Программная реализация криптоалгоритма DES.

Вариант 11

Выполнила:

Зайцева И.А.

А-13а-20

Преподаватель:

Хорев П. Б.

Содержание

Введение	3
Глава 1	4
1.1 Схема шифрования алгоритма DES.....	4
1.2 Схема расшифрования алгоритма DES.	10
1.3 Режим ECB – «электронная кодовая книга».....	11
1.4 Режим CBC – «сцепление блоков шифра».....	12
1.5 Режим CFB – «обратная связь по шифротексту».....	12
1.6 Режим OFB – «обратная связь по выходу».....	13
Глава 2.	14
2.1 Результаты проектирования структуры программы.	14
2.2 Реализация пользовательского интерфейса.	15
Глава 3.	23
3.1 Результаты тестирования созданной программы.	23
Заключение	25
Список использованных источников.....	26

Введение

DES (англ. Data Encryption Standard) – алгоритм для симметричного блочного шифрования данных, разработанный фирмой IBM. На вход алгоритма поступают 64-битные блоки данных, которые преобразуются в зашифрованные блоки идентичной длины. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) шифрования и 64-битным ключом (содержащим 8 контрольных битов). Алгоритм DES до 2001 г. являлся федеральным стандартом США на защиту информации, не относящейся к государственной тайне.

Целью выполнения данной курсовой работы ставится реализация программы с пользовательским интерфейсом, позволяющей производить шифрование и расшифрование данных любого типа с помощью криптоалгоритма DES.

Для достижения цели были поставлены следующие задачи:

- 1) детально изучить схемы шифрования и расшифрования DES;
- 2) ознакомиться с основными режимами использования данного алгоритма;
- 3) выбрать функцию хэширования для вывода ключа шифрования из парольной фразы;
- 4) реализовать алгоритмы шифрования и расшифрования данных, а также предоставить возможность работы программы в 4 режимах;
- 5) для проверки корректности расшифрования перед шифрованием добавить сигнатуру и проверять ее наличие в расшифрованном файле;
- 6) создать пользовательский интерфейс, позволяющий шифровать и расшифровывать как введенный пользователем текст, так и загруженные файлы любого типа, сохранять зашифрованные и расшифрованные данные на компьютер, а также при необходимости, генерировать парольную фразу, заданной сложности.

Далее в данном отчете подробно описаны: алгоритм DES, реализованные режимы его работы – ECB, CBC, CFB и OFB, реализация пользовательского интерфейса. Также представлены результаты тестирования созданной программы.

Глава 1

1.1 Схема шифрования алгоритма DES.

На вход программы поступает информация (читается из файла либо вводится пользователем), которая преобразуется в последовательность бит. Далее эта последовательность разбивается на блоки, каждый из которых имеет длину 64 бита.

Над каждым таким блоком производятся преобразования, согласно схеме, изображенной на рисунке 1.1.



Рисунок 1.1 Обобщённая схема шифрования DES

Матрицы начальной IP и конечной IP^{-1} перестановок являются стандартными, в них указано в каком порядке должны быть переставлены биты исходного блока. То есть, если в таблице на первом месте указано число 58, а на втором месте число 50, значит в блоке первым должен идти 58-ой бит, за ним 50-ый, и так далее.

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Таблица 1.1 Матрица начальной перестановки

После применения матрицы начальной перестановки, полученная последовательность разбивается на 2 блока по 32 бита (L – левая

подпоследовательность и R – правая подпоследовательность). Затем применяется шифрование, состоящее из 16 итераций по схеме, изображенной на рисунке 1.2.

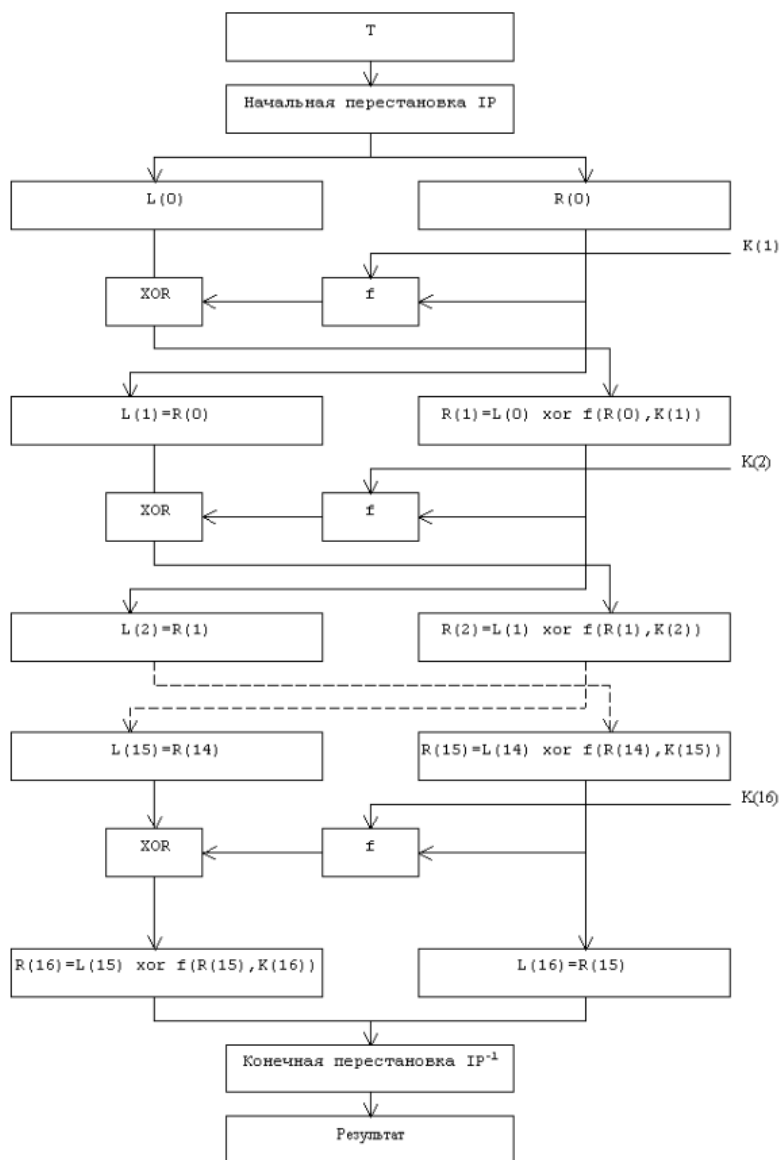


Рисунок 1.2 Схема реализации алгоритма DES

Результат i -й итерации описывается следующими формулами (прямое преобразование сетью Фейстеля):

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i),$$

где XOR - операция «исключающее или», f – функция шифрования, аргументы которой 32-битовая последовательность R_{i-1} , полученная на $(i - 1)$ итерации, к которой применяется «расширитель» до 48 бит, и 48-битный ключ K_i , который является результатом преобразования 64-битового ключа K. Подробное описание функции шифрования и алгоритма генерации ключей будет представлено ниже.

На 16-й итерации получаем последовательности R_{16} и L_{16} , которые конкатенируются в 64-битовую последовательность $R_{16}L_{16}$. Затем позиции битов этой последовательности переставляются в соответствии с матрицей конечной перестановки IP^{-1} – таблица 1.2.

40	08	48	16	56	24	64	32
39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30
37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28
35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26
33	01	41	09	49	17	57	25

Таблица 1.2 Матрица конечной перестановки

Вычисление функции шифрования $f(R_{i-1}, K_i)$ схематично представлено на рисунке 1.3.

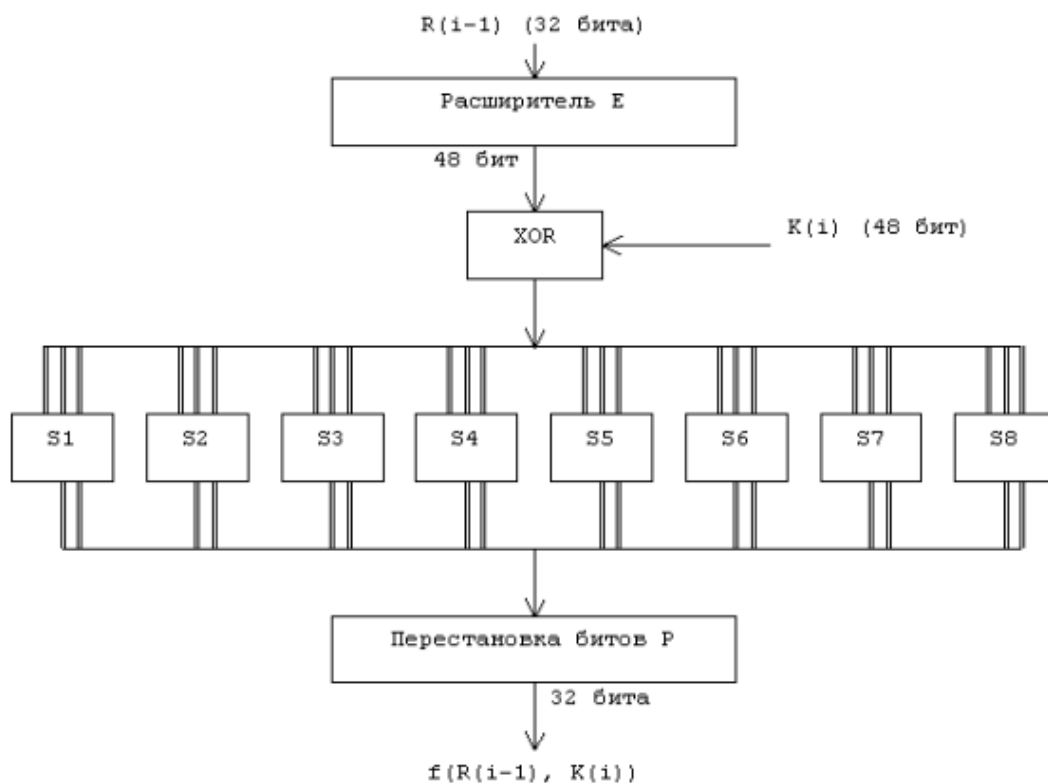


Рисунок 1.3 Вычисление функции шифрования $f(R(i-1), K(i))$

Для вычисления значения функции f используются следующие функции-матрицы:

- E – расширение 32-битовой последовательности до 48-битовой,
- $S1, S2, \dots, S8$ – преобразование 6-битового блока в 4-битовый,
- P – перестановка бит в 32-битовой последовательности.

Функция расширения E определяется таблицей 1.3.

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	01

Таблица 1.3 Матрица расширения E

Результат функции $E(R_{i-1})$ есть 48-битовая последовательность, которая складывается по модулю 2 (операция XOR) с 48-битовым ключом K_i . Получается новая 48-битовая последовательность, которая разбивается на восемь 6-битовых блоков $B_1B_2B_3B_4B_5B_6B_7B_8$. То есть:

$$E(R_{i-1}) \text{ xor } K_i = B_1B_2B_3B_4B_5B_6B_7B_8.$$

Функции $S1, S2, \dots, S8$ определяются в таблице 1.4.

		Номер столбца																		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
Н о м е р с т р о к и	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S1		
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8			
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0			
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13			
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S2		
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5			
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15			
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9			
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S3		
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1			
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7			
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12			
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S4		
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9			
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4			
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14			
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S5		
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6			
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14			
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3			
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S6		
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8			
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6			
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13			
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S7		
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6			
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2			
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12			
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S8		
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2			
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8			
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11			

Таблица 1.4 Функции S1, S2 ,..., S8

Пояснение к таблице 1.4. Пусть на вход функции-матрицы S_j поступает 6-битовый блок $B_j = b_1b_2b_3b_4b_5b_6$, тогда двухбитовое число b_1b_6 указывает номер строки таблицы 1.4, а $b_2b_3b_4b_5$ - номер столбца. Результатом $S_j(B_j)$ будет 4-битовый элемент, расположенный на пересечении указанных строки и столбца.

Применив операцию выбора к каждому из 6-битовых блоков $B_1, B_2 \dots B_8$, получаем 32-битовую последовательность $S1(B_1)S2(B_2) \dots S8(B_8)$. Для получения результата функции шифрования надо переставить биты этой последовательности. Для этого применяется функция перестановки P – таблица 1.5.

16	07	20	21
29	12	28	17
01	15	23	26
05	18	31	10
02	08	24	14
32	27	03	09
19	13	30	06
22	11	04	25

Таблица 1.5 Матрица перестановки P

Таким образом, $f(R_{i-1}, K_i) = P(S1(B_1), \dots, S8(B_8))$.

Далее рассмотрим процедуру вычисления 48-битовых ключей K_i , $i = 1 \dots 16$. На каждой итерации используется новое значение ключа K_i , которое вычисляется из начального ключа K . K представляет собой 64-битовый блок с восемью битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных битов и перестановки остальных используется функция G первоначальной подготовки ключа – таблица 1.6.

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

Таблица 1.6 Матрица G первоначальной подготовки ключа

Результат преобразования $G(K)$ разбивается на два 28-битовых блока C_0 и D_0 , причем C_0 будет состоять из битов 57, 49, ..., 44, 36 ключа K , а D_0 будет состоять из битов 63, 55, ..., 12, 4 ключа K . После определения C_0 и D_0

рекурсивно определяются C_i и $D_i, i = 1 \dots 16$. Для этого применяют циклический сдвиг влево на один или два бита в зависимости от номера итерации, как показано в таблице 1.7.

Номер итерации	Сдвиг (бит)
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Таблица 1.7 Сдвиг для вычисления ключа

К полученной в результате конкатенации C_i и D_i последовательности бит применяется перестановка в соответствии с матрицей H - таблица 1.8.

14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Таблица 1.8 Матрица H завершающей обработки ключа

Таким образом: $K(i) = H(C(i)D(i))$

Схематично процедура вычисления ключа представлена на рисунке 1.4.

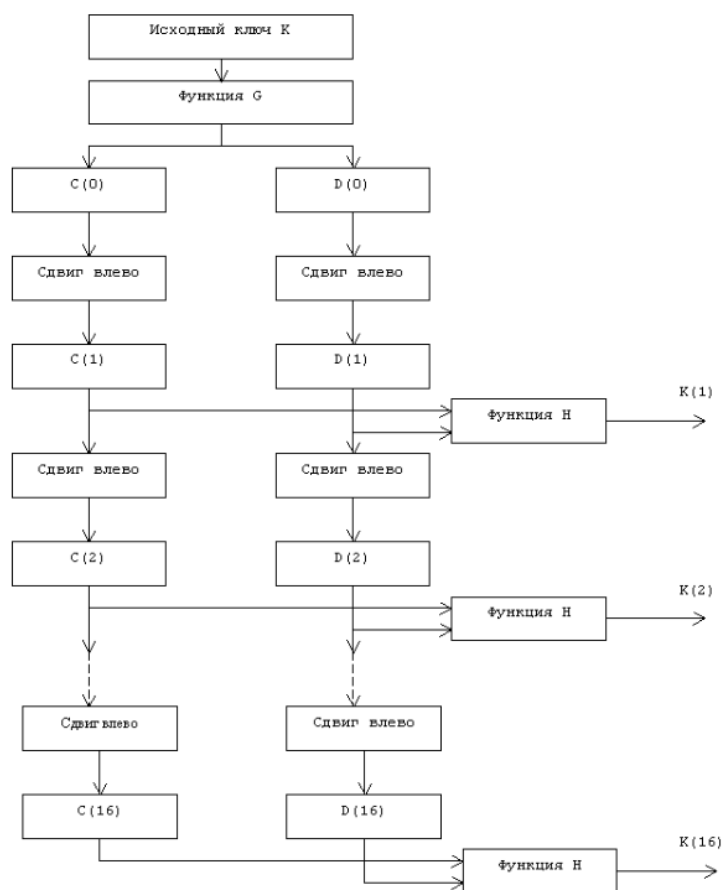


Рисунок 1.4 Схема вычисления i -ого ключа

1.2 Схема расшифрования алгоритма DES.

Процесс расшифрования данных в алгоритме DES является инверсным по отношению к процессу шифрования. Действия, выполняемые в ходе 16 итераций, должны быть выполнены в обратном порядке. Ключи также применяются в обратной последовательности – сначала K_{16} , затем K_{15}, \dots, K_1 .

На вход алгоритма расшифрования подается информация в битовом представлении, которая затем делится на блоки из 64-бит. Затем к каждому из этих блоков применяется матрица начальной перестановки IP. Полученная последовательность бит будет представлять собой последовательность $R_{16}L_{16}$. Затем над этой последовательностью выполняются те же действия, что и в процессе шифрования, но в обратном порядке. Итеративный процесс расшифрования (обратное преобразование сетью Фейстеля) запишем в виде:

$$R_{i-1} = L_i, \quad i = 1, 2, \dots, 16$$

$$L_{i-1} = R_i \text{ XOR } f(L_i, K_i), \quad i = 1, 2, \dots, 16$$

На 16-итерации расшифрования получают последовательности L_0 и R_0 , которые конкатенируют в 64-битовую последовательность L_0R_0 . Затем позиции бит этой последовательности переставляют в соответствии с матрицей конечной перестановки IP^{-1} . Результатом этой перестановки является исходная 64-

битовая последовательность. Схематично алгоритм расшифрования представлен на рисунке 1.5.

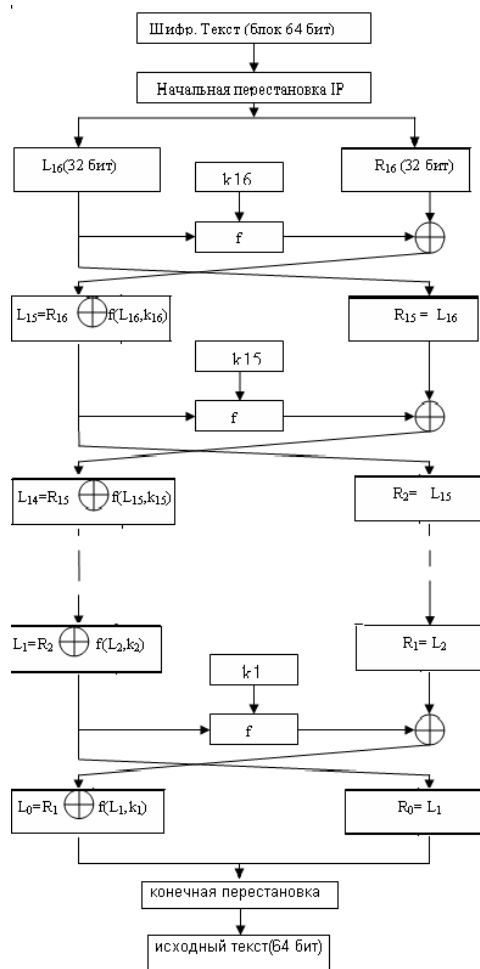


Рисунок 1.5 Схема расшифрования алгоритма DES

Далее будут рассмотрены режимы шифрования DES.

1.3 Режим ЕСВ – «электронная кодовая книга».

Шифруемый текст разбивается на блоки по 64 бита, при этом каждый блок шифруется отдельно, не взаимодействуя с другими блоками. Схема шифрования и расшифрования режима ЕСВ представлена на рисунке 1.6.

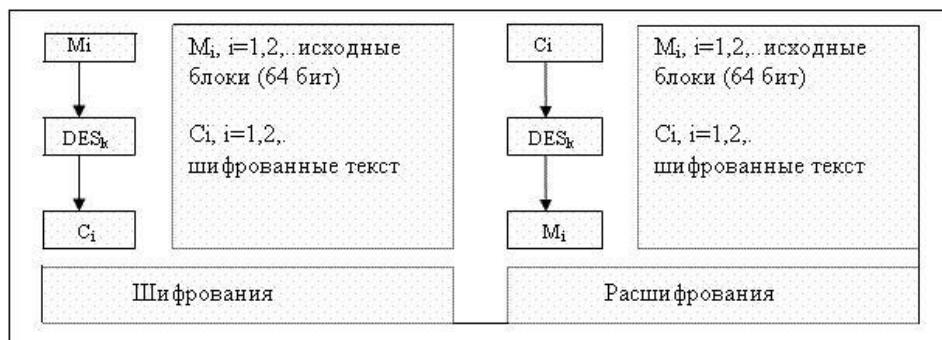


Рисунок 1.6 Режим работы ЕСВ

1.4 Режим CBC – «сцепление блоков шифра».

Каждый очередной блок M_i $i \geq 1$ складывается по модулю 2 с предыдущим блоком зашифрованного текста C_{i-1} . Вектор C_0 – начальный вектор, который меняется ежедневно и хранится в секрете. Схема шифрования и расшифрования режима CBC представлена на рисунке 1.7.

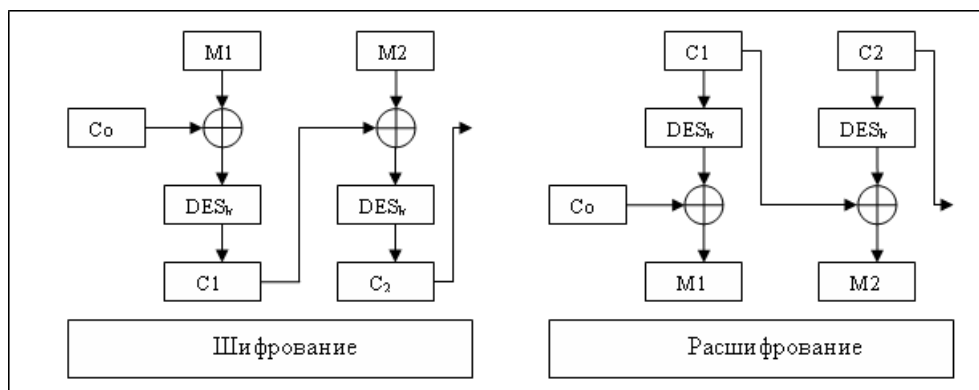


Рисунок 1.7 Режим работы CBC

1.5 Режим CFB – «обратная связь по шифротексту».

В этом режиме размер блока может отличаться от 64 бит. Исходный файл M считывается последовательными t -битовыми блоками ($t \leq 64$): $M = M_1 M_2 \dots M_n$ (остаток дописывается нулями).

64-битовый сдвиговый регистр (входной блок) вначале содержит вектор инициализации IV , выравненный по правому краю. Для каждого сеанса шифрования используется новый IV . Для всех $i = 1 \dots n$ блок шифротекста C_i определяется следующим образом:

$$C_i = M_i \text{ XOR } P_{i-1},$$

где P_{i-1} , – старшие t бит операции $DES(C_{i-1})$, причем $C_0 = IV$.

Обновление сдвигового регистра осуществляется путем удаления его старших t битов и дописывания справа C_i . При восстановлении зашифрованных данных: P_{i-1} и C_i вычисляются аналогичным образом и

$$M_i = C_i \text{ XOR } P_{i-1}.$$

Схема шифрования и расшифрования режима CFB представлена на рисунке 1.8.

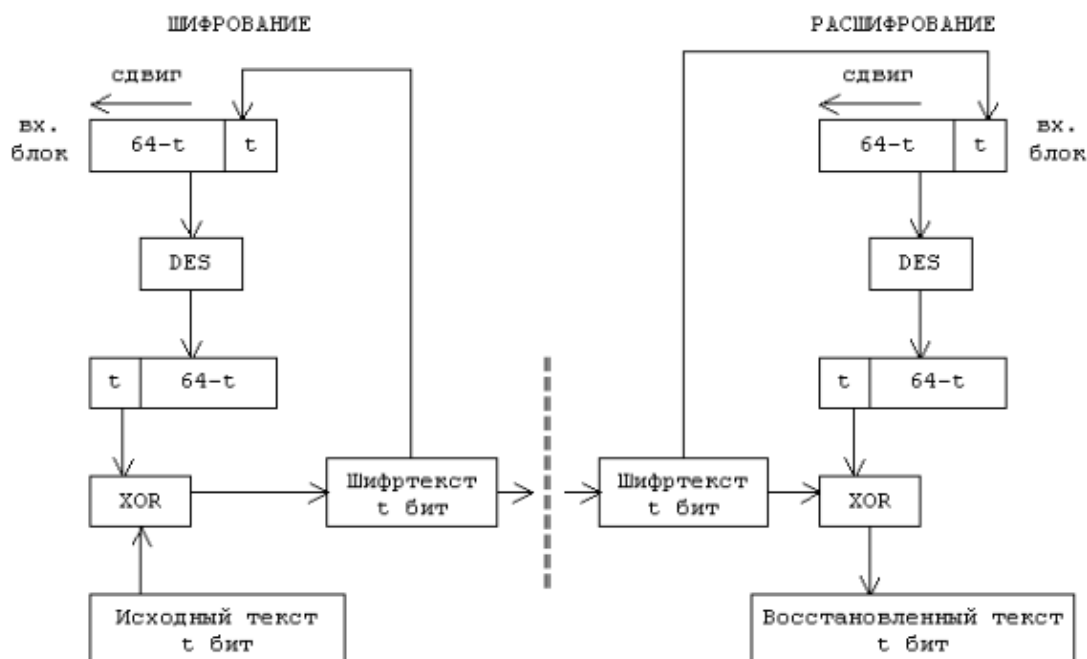


Рисунок 0.8 Режим работы CFB и OFB

1.6 Режим OFB – «обратная связь по выходу».

Режим OFB схож по своему действию с алгоритмом CFB. Отличие состоит только в методе обновления сдвигового регистра. В данном случае это осуществляется путем удаления его старших t битов и дописывания справа P_{i-1} . Схема шифрования и расшифрования режима OFB представлена на рисунке 1.8.

Глава 2.

2.1 Результаты проектирования структуры программы.

Программный код написан на языке C# в среде Visual Studio 2019г. Для реализации криптоалгоритма DES был создан класс DES_CRYPT. При создании экземпляра данного класса необходимо указать 3 параметра – (string)content, (int) mode, (string)passphrase.

Первый параметр может иметь 2 назначения. Это может быть либо путь к шифруемому (расшифровываемому) файлу, либо введенный пользователем текст. Во втором случае программа, в которой реализован интерфейс, автоматически добавляет в начало строки введенного текста сигнатуру «ТЕХТ», по которой и происходит различение этих двух случаев, затем сигнатура удаляется и производится последующая обработка.

Второй параметр – выбор одного из 4 режимов шифрования, ожидает значения 1, 2, 3 или 4 для режимов ECB, CBC, CFB и OFB соответственно.

Третий параметр – парольная фраза для шифрования или расшифрования.

Для того, чтобы зашифровать файл с помощью DES_CRYPT, необходимо вызвать функцию Des_Encode(outfile), в параметры которой нужно передать путь для сохранения готового зашифрованного файла. Аналогично реализовано расшифрование, для него потребуется функция Des_Decode(inpfile, outfile), для вызова которой нужно передать в её параметры путь к файлу, который требуется расшифровать, и путь для сохранения готового расшифрованного файла.

Внутреннее устройство функций Des_Encode и Des_Decode содержит применение алгоритма хэширования MD5 из модуля System.Security.Cryptography для вывода начального ключа из парольной фразы. Для правильной работы DES берутся первые 64 бита хэша, возвращаемого функцией MD5.

Функция Des_Encode добавляет в исходный файл перед шифрованием сигнатуру «IrinaZaitseva», а после шифрования удаляет ее из исходного файла. Функция Des_Decode проверяет с помощью внутреннего вызова функции CheckSignature наличие этой сигнатуры в расшифрованном файле. Если наличие подтверждается, функция возвращает значение «истина», что сигнализирует о том, что расшифрование прошло успешно, в противном случае будет возвращено значение «ложь».

В остальном реализация функций шифрования и расшифрования файлов выполнена в соответствии с алгоритмами, указанными в главе 1 данного отчета.

2.2 Реализация пользовательского интерфейса.

В данном разделе отчета можно увидеть снимки экранных форм, демонстрирующих интерфейсную часть программы.

Приветственное окно программы. При наведении курсора мыши на текст «Начало работы», «О программе» и «Выход» эти поля выделяются цветом.

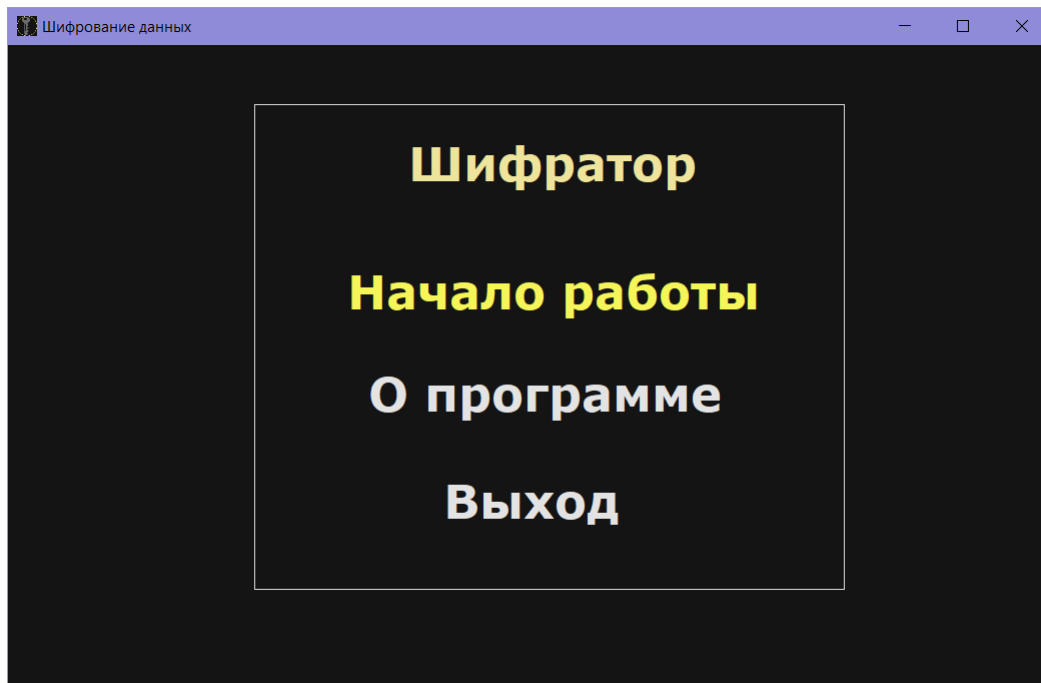


Рисунок 2.1 Начальное окно программы

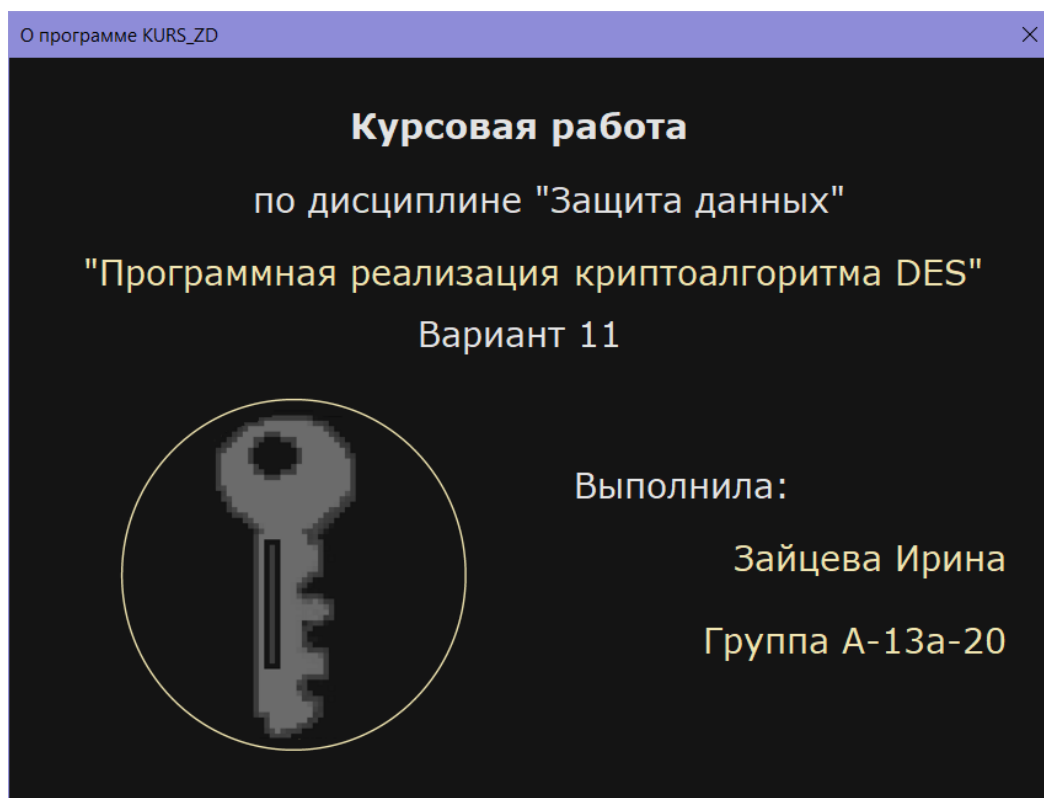


Рисунок 2.2 Окно "О программе"

Переход в пункт меню «Начало работы».

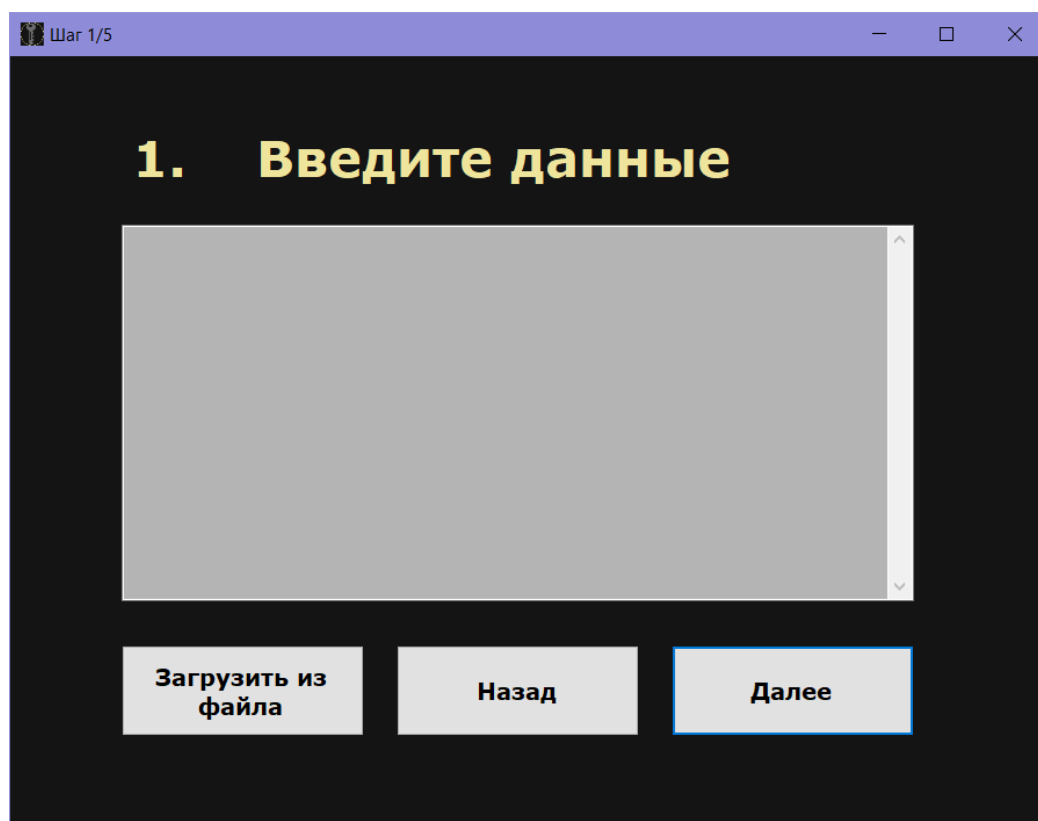


Рисунок 2.3 Шаг 1 - ввод данных для шифрования

При нажатии на кнопку «Загрузить из файла» будет открыт стандартный диалог выбора файлов в проводнике.

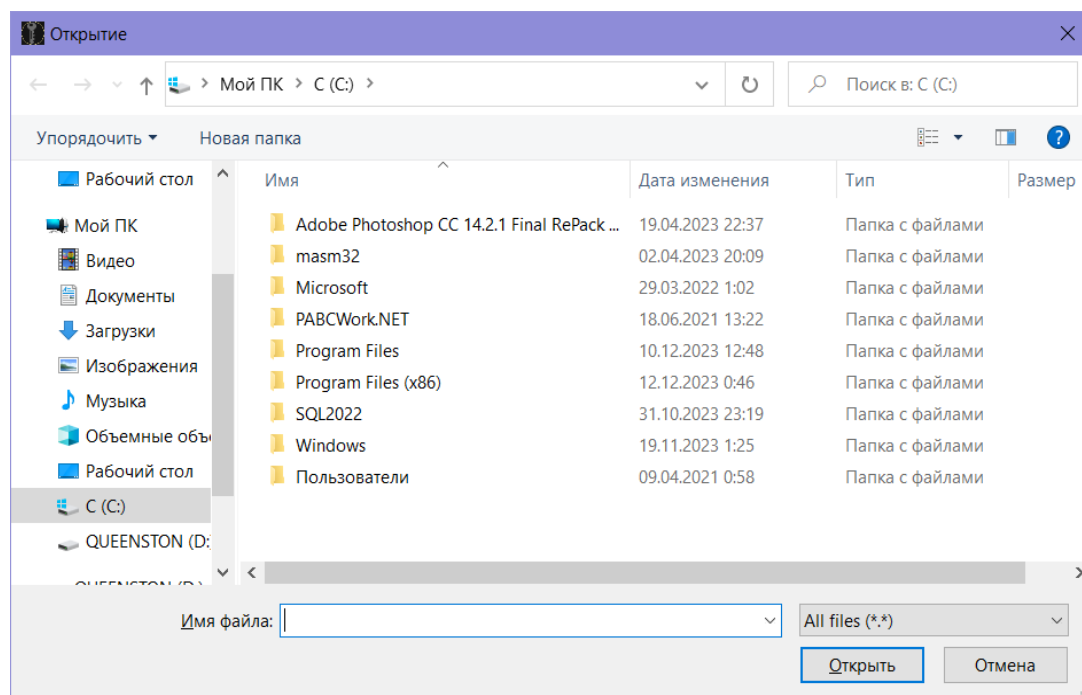


Рисунок 2. 4 Окно проводника выбора файла

После выбора файла и нажатия на кнопку «Открыть» на экране увидим путь к выбранному файлу, а кнопка загрузки файлов изменит свой текст на «Отменить загрузку файла». В этом же поле можно ввести текст для шифрования вручную.

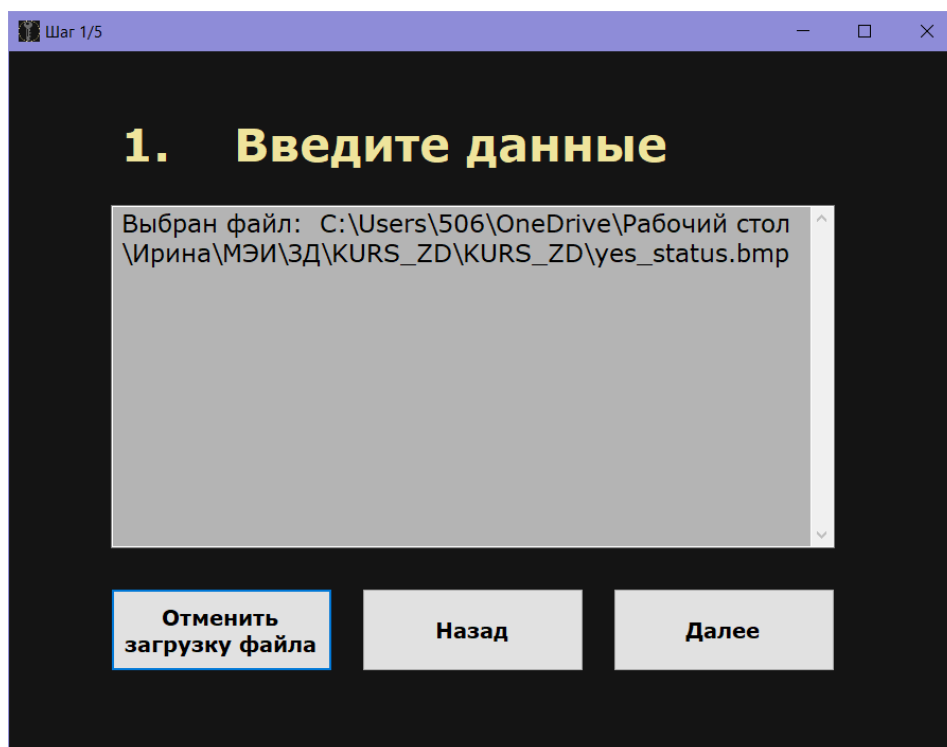


Рисунок 2.5 Шаг 1 Выбран файл

Переход по кнопке «Далее».

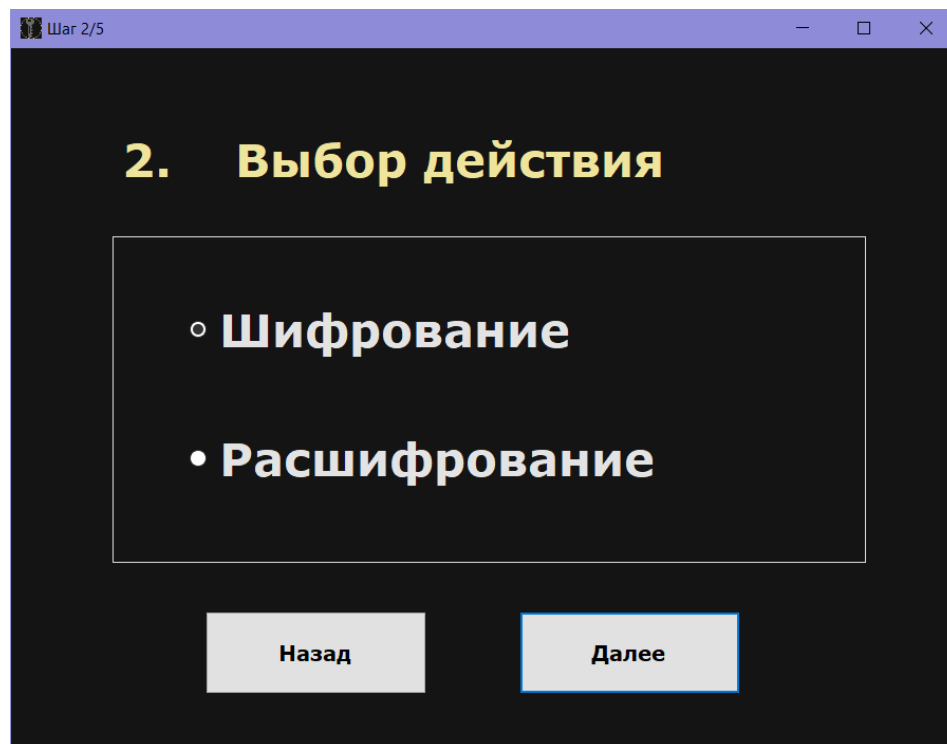
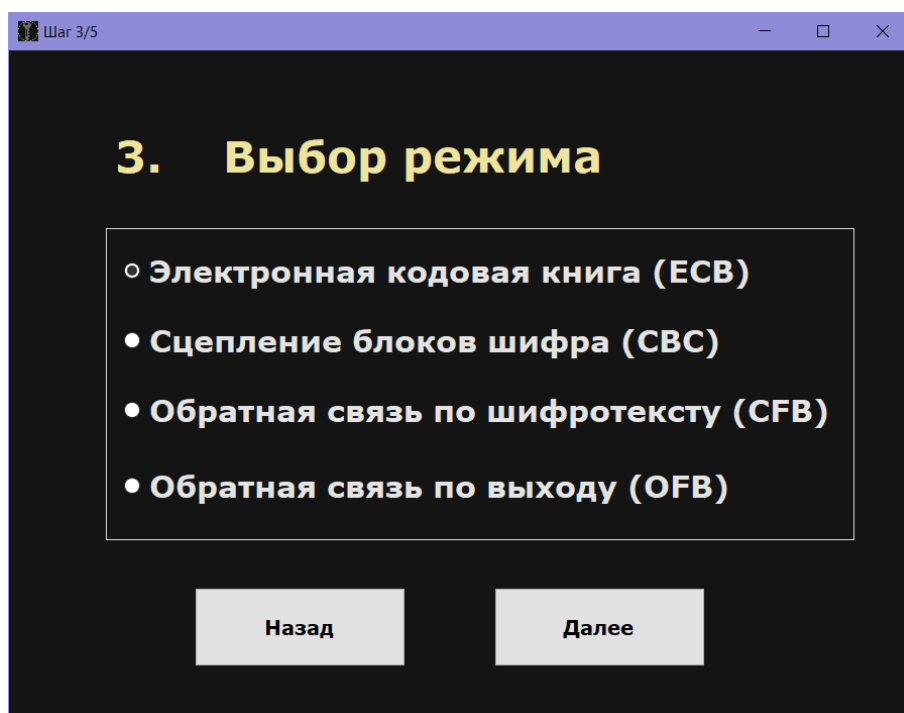


Рисунок 2.6 Шаг 2 Выбор действия для обработки файла

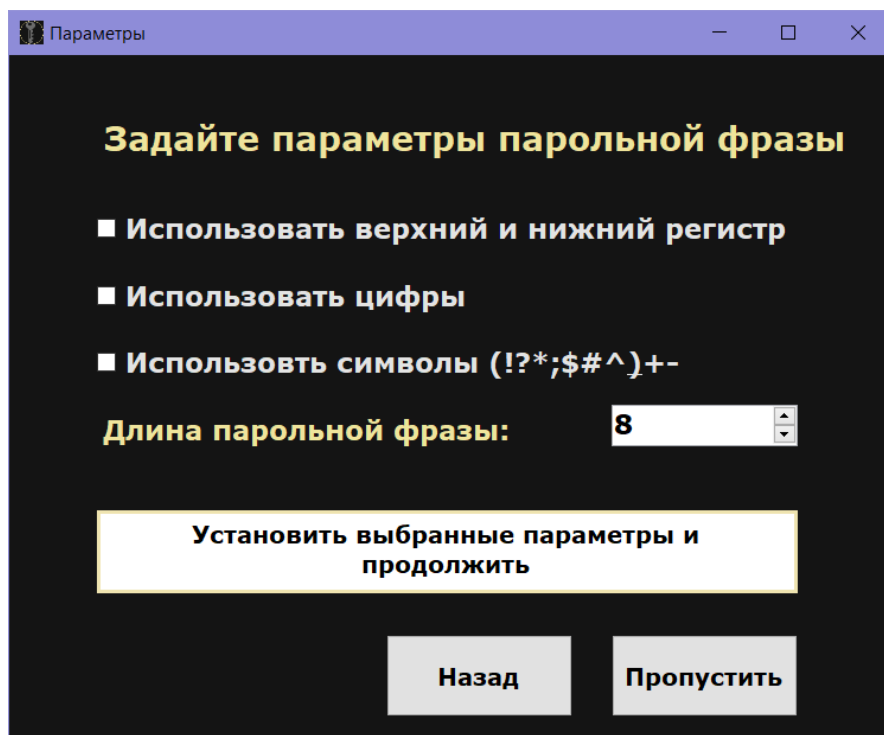
Переход по кнопке «Далее».



The screenshot shows a window titled 'Шаг 3/5' with a dark background. At the top, the text '3. Выбор режима' is displayed in yellow. Below it, a list of four encryption modes is shown, each preceded by a radio button: 'Электронная кодовая книга (ECB)', 'Сцепление блоков шифра (CBC)', 'Обратная связь по шифротексту (CFB)', and 'Обратная связь по выходу (OFB)'. At the bottom, there are two buttons: 'Назад' and 'Далее'.

Рисунок 2.7 Шаг 3 Выбор режима шифрования

Переход по кнопке «Далее», если ранее был выбран режим «Шифрование».



The screenshot shows a window titled 'Параметры' with a dark background. The main heading is 'Задайте параметры парольной фразы' in yellow. Below it, there are three checkboxes: 'Использовать верхний и нижний регистр', 'Использовать цифры', and 'Использовать символы (!?*; \$#^)+-'. Below these is a label 'Длина парольной фразы:' followed by a text input field containing the number '8'. At the bottom, there is a large button labeled 'Установить выбранные параметры и продолжить'. At the very bottom, there are two buttons: 'Назад' and 'Пропустить'.

Рисунок 2.8 Установка ограничений на парольную фразу

При нажатии на кнопку «Пропустить» будут установлены ограничения по умолчанию и будет выполнен переход к следующей форме.

Переход по кнопке «Далее». Поля можно заполнять вручную или с помощью кнопки «Сгенерировать».

Рисунок 2.9 Шаг 4 Ввод парольной фразы для режима шифрование

На парольную фразу установлены ограничения по умолчанию: минимальное число знаков = 8, обязательно использование хотя бы одной заглавной буквы и хотя бы одной цифры (или ограничения, выбранные пользователем).

При вводе парольной фразы, не удовлетворяющей ограничениям, могут выводиться следующие сообщения:

Рисунок 2.10 Возможные сообщения об ошибках

Переход по кнопке «Сгенерировать». При генерации можно установить более жесткие ограничения, чем те, что были заданы ранее (рисунок 2.8), но нельзя установить менее жесткие. Галочка на кнопке показана нажата для наглядности демонстрации работы. В обычном режиме сгенерированная парольная фраза затемняется. При нажатии на кнопку продолжить, сгенерированный пароль автоматически переносится в форму установления пароля.

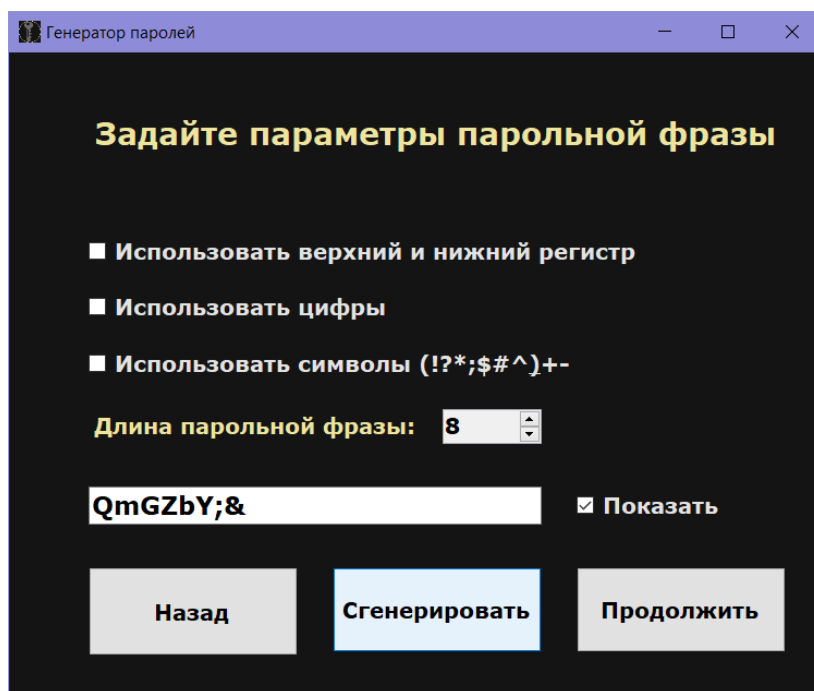


Рисунок 2.11 Генератор паролей

Если на шаге 2 выбран режим «Расшифрование», то переход по кнопке «Далее».

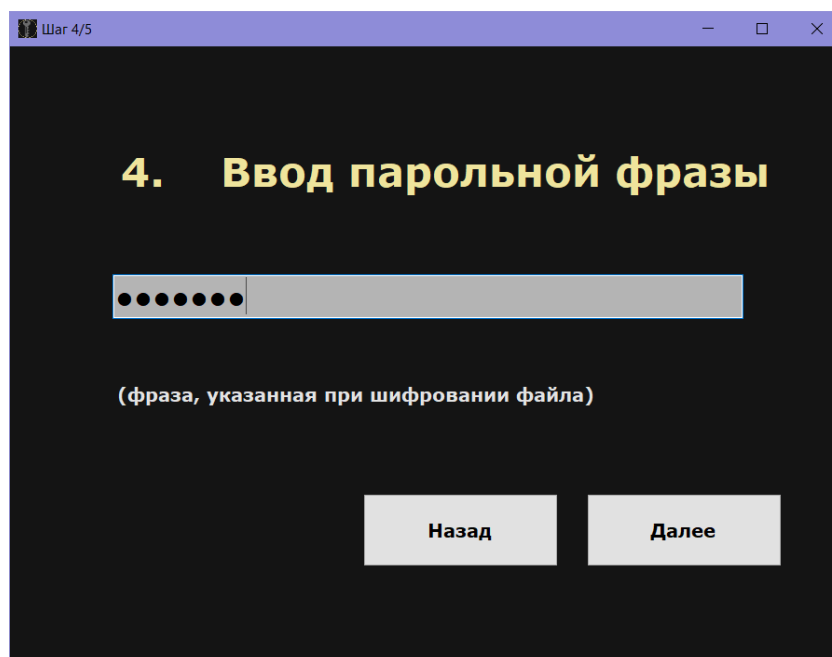


Рисунок 2.12 Шаг 4 Ввод парольной фразы при расшифровании

Переход по кнопке «Далее» на шаг 5 по любой из веток «Шифрование» или «Расшифрование».

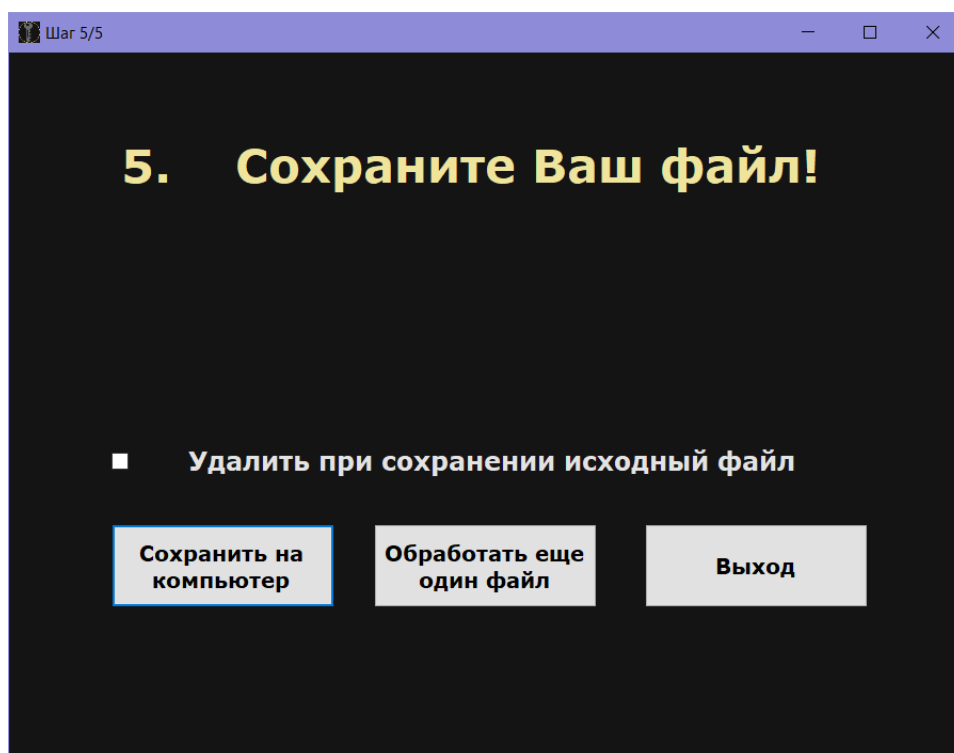


Рисунок 2.13 Шаг 5 Сохранение файла

При нажатии на кнопку «Сохранить на компьютер» откроется стандартное диалоговое окно сохранения файла.

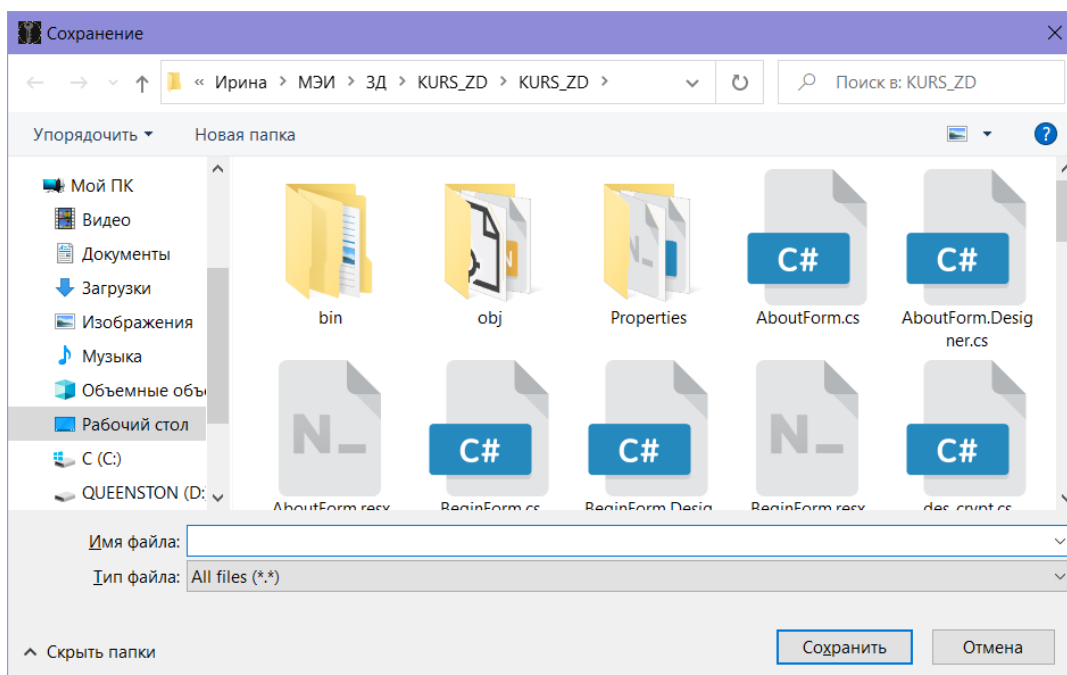


Рисунок 2.14 Сохранение файла на компьютер

Если процедура шифрования/расшифрования прошла успешно, то пользователь увидит следующее окно:

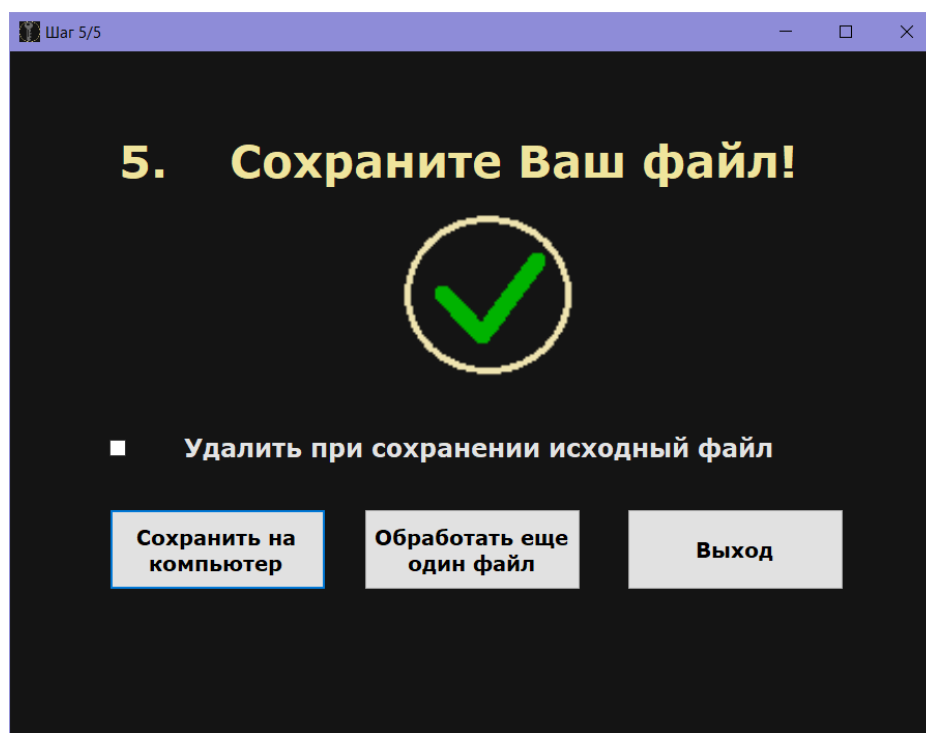


Рисунок 2.15 Успешное сохранение обработанного файла

Если процедура расшифрования не удалась, то пользователь увидит соответствующее сообщение.

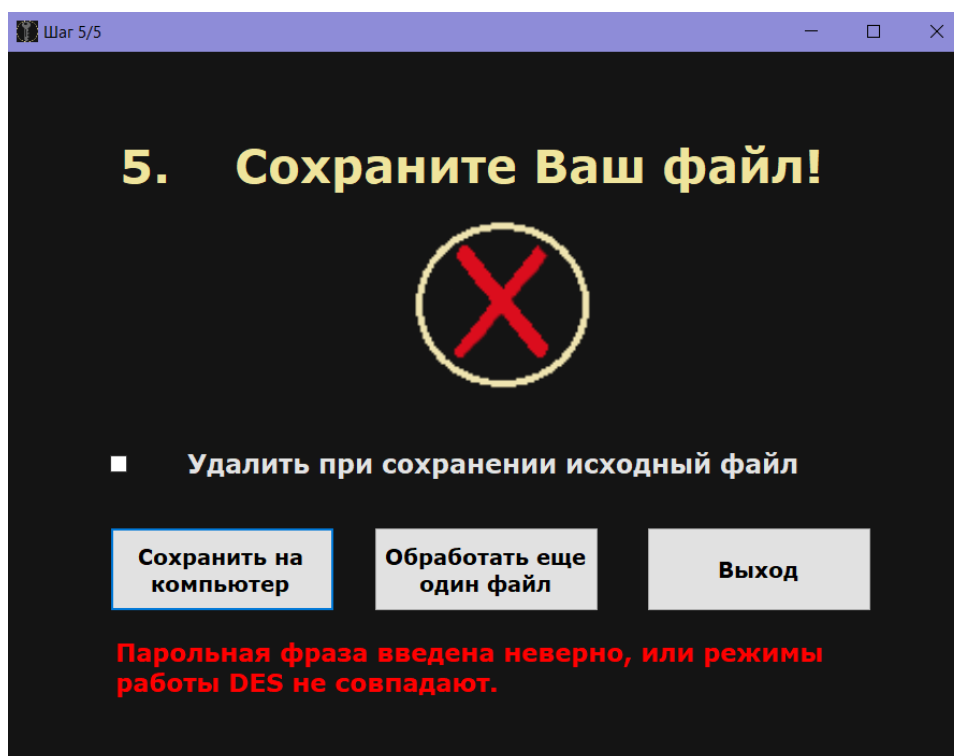


Рисунок 2.16 Неудачная попытка расшифрования

Глава 3.

3.1 Результаты тестирования созданной программы.

Содержимое исходного файла.

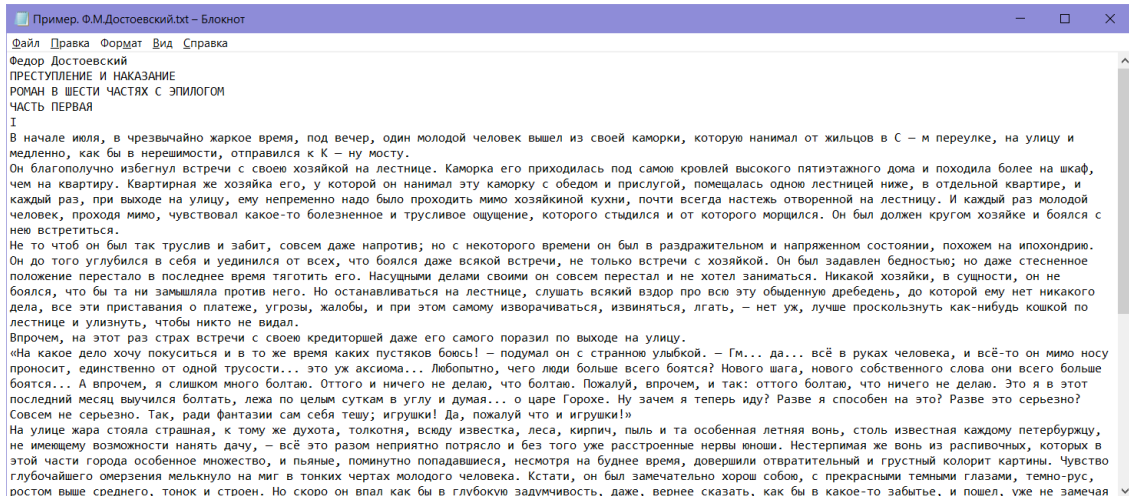


Рисунок 3.1 Содержимое исходного файла с текстом на русском языке

Содержимое текстового файла после шифрования.

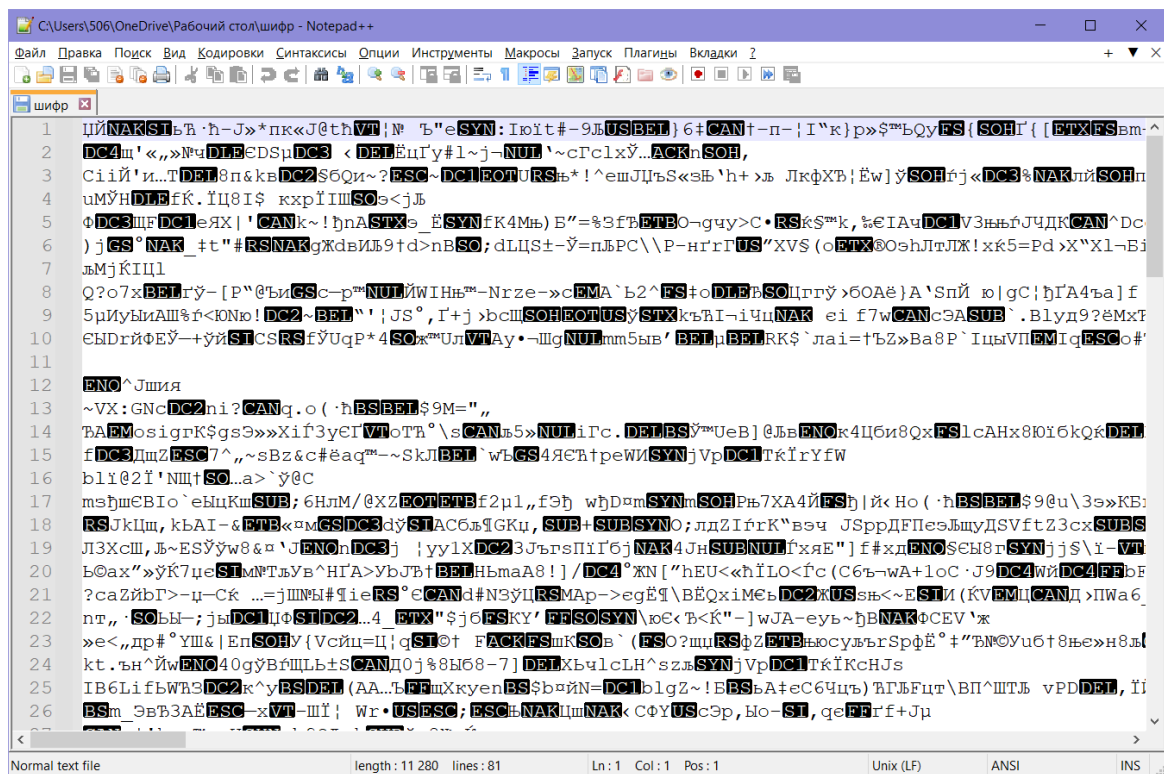


Рисунок 3.2 Шифротекст

Можно видеть, что исходный текст невозможно распознать по данному файлу.

Содержимое расшифрованного файла в точности соответствует содержимому исходного файла.

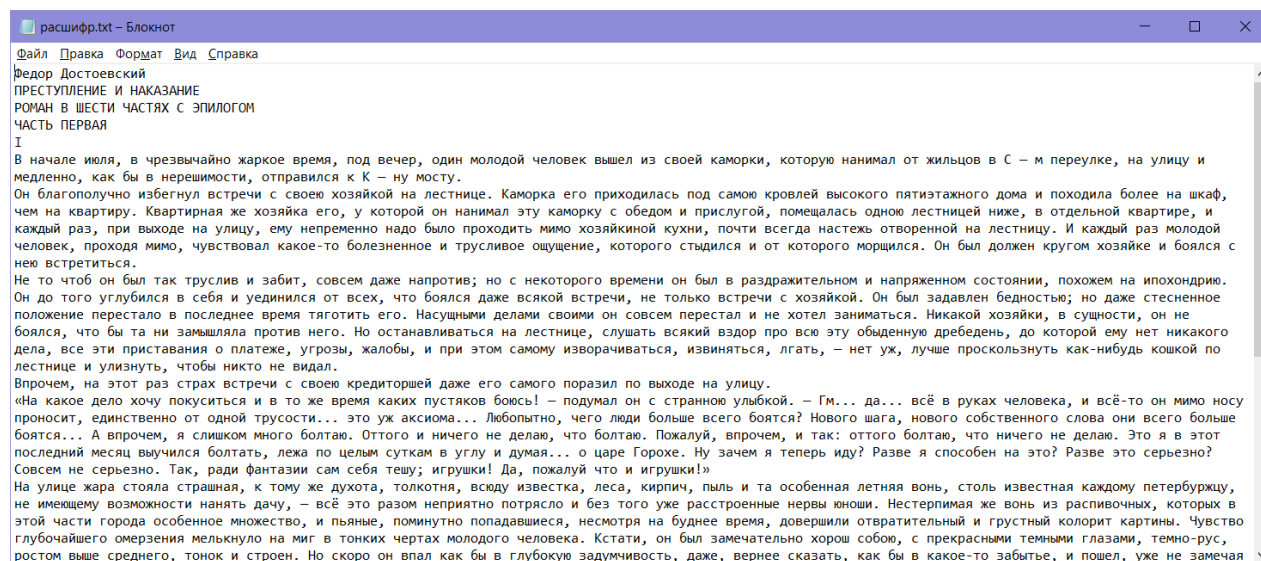


Рисунок 3. 3 Расшифрованный текст

Программа работает с любым форматом исходного файла, так как при реализации алгоритмов чтение информации из исходных файлов происходит по байтам, а не по символам.

К недостаткам программы можно отнести относительно низкую скорость работы при попытке шифрования слишком больших файлов (например, художественных фильмов, длительностью более 1.5 часов). Это связано с низкой скоростью реализации перестановок, а также способом хранения битов исходного файла в массивах. Это может быть оптимизировано, если хранить все биты блоков как одно число, однако данная реализация существенно ухудшает читаемость и простоту понимания кода, поэтому в данной версии программы не была реализована.

Заключение

В результате выполнения курсовой работы был изучен криптоалгоритм симметричного блочного шифрования DES и возможные режимы его работы. Этот алгоритм лёг в основу программы с пользовательским интерфейсом, позволяющей шифровать и расшифровывать файлы любого типа и сохранять результаты на компьютер пользователя. В программе поддерживаются все 4 возможных режима шифрования – ECB, CBC, CFB и OFB.

В качестве дополнительного инструмента в программе имеется генератор паролей, который создает последовательность случайных символов по заданным ограничениям.

Список использованных источников

1. Криптографические и стеганографические средства защиты данных: учеб. пособие / А.В. Сергеев, П.Б. Хорев. – М.: Издательство МЭИ, 2023. – 72 с.
2. DES [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/DES>.
3. Стандарт шифрования данных Data Encryption Standard [Электронный ресурс]: Московский авиационный институт. Кафедра № 401. – URL: <https://kaf401.rloc.ru/Criptfiles/DES.htm>