

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра защиты информации



Новосибирский
государственный
технический университет

НЭТИ

КУРСОВАЯ РАБОТА
по дисциплине: «Программирование»
на тему: «*Разработка*
криптографического программного
обеспечения "Шифровка и дешифровка
***текста"*»**

Выполнил:

Студент гр. «АБ-121», «АВТФ»

Втюрин Александр Романович

«16» декабря 2022г.

(подпись)

Проверил:

доцент кафедры ЗИ

Архипова А. Б.

«__» _____ 20__ г.

(подпись)

Новосибирск 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра _____ Защита информации
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой Иванов А. В.

(подпись, дата)

**ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ**

студенту _____ Втюрину Александру Романовичу
(фамилия, имя, отчество)

Направление подготовки _____ 10.03.01 – Информационная безопасность
(код и наименование направления подготовки бакалавра)

_____ Факультет автоматики и вычислительной техники
(полное название факультета)

Тема Разработка криптографического программного обеспечения "Шифровка и дешифровка текста"

Календарный план

Наименование задач (мероприятий), составляющих задание	Дата выполнения задачи (мероприятия)	Подпись руководителя
1	2	3
Развернутая постановка задачи, изучение необходимой научно- технической литературы	17.09.2022 г.	
Разработка структуры данных и алгоритма решения задачи	04.10.2022 г.	

1	2	3
Написание текста задачи	21.10.2022 г.	
Тестирование и отладка программного продукта	23.11.2022 г.	
Оформление отчета о проделанной работе	13.12.2022 г.	
Сдача работы руководителю и ее защита	15.12.2022 г.	

Задание согласовано и принято к исполнению.

Руководитель
от НГТУ
Архипова Анастасия
Борисовна

(фамилия, имя, отчество)

К.Т.Н.

(ученая степень, ученое звание)

(подпись, дата)

Студент

(фамилия, имя, отчество)

(факультет, группа)

(подпись, дат

СОДЕРЖАНИЕ

Введение	5
1 Современные блочные алгоритмы шифрования.....	7
1.1 Общие сведения о блочных шифрах	7
1.2 Математические основы	7
1.3 Сеть Фейстеля.....	8
2. Алгоритм AES.....	10
2.1 Описание алгоритма.....	10
2.2 Математический аппарат.....	11
2.3 Методы взлома	17
3 Описание программного обеспечения	19
3.1 Состав и структура программного обеспечения	19
3.2 Программная реализация функций	20
3.3. Руководство пользователя.....	20
3.3.1 Введение.....	20
3.3.2 Назначение и условия применения	21
3.3.3 Подготовка к работе.....	21
3.3.4 Описание операций	22
3.3.5 Сообщение пользователю.....	33
3.3.6 Аварийные ситуации.....	33
4 Руководство системного программиста.....	34
4.1 Общие сведения.....	34
4.2 Структура программы.....	34
4.3 Сообщение системному программисту.....	36
5 Контрольный пример	37
5.1 Запуск программы	37
Заключение.....	52
Список используемых источников	53
Приложение. Листинг программы.....	55

ВВЕДЕНИЕ

Сегодня криптография является неотъемлемой частью нашей жизни, и применяется во многих её сферах, например передача сообщений или каких-либо данных. Так же сегодня криптография активно применяется для защиты персональных данных от посторонних. Современные методы шифрования являются крайне надёжными, я решил воспользоваться возможностью для создания программного обеспечения для шифровки\дешифровки текста.

Теоретическую основу моей работы составляют следующие книги: Введение в криптографию. Курс лекций (В.А. Романьков)[1], Курс криптографии (Земор Жиль)[2], Основы криптографии : учебное пособие(Г. В. Басалова)[3], Криптография и безопасность сетей : учебное пособие(Б. А. Фороузан)[4], Алгоритмы блочной криптографии учебно-методическое пособие (Н.Р. Спиричева)[5].

Практическую основу моей работы составляют следующие книги: Дополнительные приемы программирования на языке с# (Керов Л.А.)[6], Основы программирования на С# (Биллиг, В. А.)[7], Основы объектного программирования на С# (Биллиг, В. А.)[8], User interface development based on Windows Forms class library (Абрамян, М. Э.)[9], Разработка Windows-приложений на основе Visual C# (Кариев, Ч. А.)[10].

Целью курсовой работы является Разработка криптографического программного обеспечения "Шифровка и дешифровка текста".

Для реализации поставленной цели необходимо решить ряд задач:

- проанализировать программные средства, выполняющие аналогичную задачу;
- проработать структуру программы;
- изучить работу с файлами и данными;
- разработать функционал для работы с данными, а именно:
 - ввод исходного текста или зашифрованного текста;
 - ввод или автоматическую генерацию ключей шифрования;

- разработать алгоритмы шифровки дешифровки текста согласно выбранным криптографическим алгоритмам;
- вывод шифрованного и дешифрованного текста;
- изучить и проанализировать необходимую информацию о разработке графического приложения;
- разработать и реализовать внешнюю оболочку программы, понятную и удобную для рядового пользователя;
- реализовать функционал и связать его с внешней оболочкой программы.

Ожидаемый результат: полностью функционирующая программа, позволяющая пользователю шифровать и дешифровать текст.

1 Современные блочные алгоритмы шифрования

1.1 Общие сведения о блочных шифрах

Характерной особенностью блочных криптоалгоритмов является тот факт, что в ходе своей работы они осуществляют преобразование блока входной информации фиксированной длины и получают результирующий блок той же длины, но недоступный для прочтения сторонним лицам, не владеющим ключом. Таким образом, схему работы блочного шифра можно описать функциями:

$$Z = \text{EnCrypt}(X, \text{Key}) \text{ и } X = \text{DeCrypt}(Z, \text{Key}),$$

Где Z – зашифрованный текст, $\text{EnCrypt}(X, \text{Key})$ – криптографическая функция шифрующая исходный текст X с помощью ключа Key , X – исходный текст, $\text{DeCrypt}(Z, \text{Key})$ – криптографическая функция дешифрующая зашифрованный текст Z с помощью ключа Key .

Ключ Key является параметром блочного криптоалгоритма и представляет собой некоторый блок двоичной информации фиксированного размера. Исходный X и зашифрованный Z блоки данных также имеют фиксированную разрядность, равную между собой, но необязательно равную длине ключа.[5]

1.2 Математические основы

Все действия, производимые над данным блочным криптоалгоритмом, основаны на том факте, что преобразуемый блок может быть представлен в виде целого неотрицательного числа из диапазона, соответствующего его разрядности. Так, например, 32-битный блок данных можно интерпретировать как число из диапазона 0-4'294'967'295. Кроме того, блок, разрядность которого обычно является "степенью двойки", можно трактовать как несколько независимых неотрицательных чисел из меньшего диапазона (рассмотренный выше 32-битный блок можно также представить в виде 2 независимых чисел из диапазона 0-65535 или в виде 4 независимых чисел из диапазона 0-255). Над этими числами блочным

криптоалгоритмом и производятся по определенной схеме действия, представленные в таблице 1.1 (первый столбец - условные обозначения данных действий на графических схемах алгоритмов). [5]

Таблица 1.1 – Основные операции в блочных криптоалгоритмах

Операция	Уравнение
Биективные математические функции	
Сложение	$X' = X + V$
Исключающее ИЛИ	$X' = X \text{ XOR } V$
Умножение по модулю $2^N + 1$	$X' = (XV) \bmod 2^N + 1$
Умножение по модулю 2^N	$X' = (XV) \bmod 2^N$
Битовый сдвиг	
Арифметический сдвиг влево	$X' = X \text{ SHL } V$
Арифметический сдвиг вправо	$X' = X \text{ SHR } V$
Циклический сдвиг влево	$X' = X \text{ ROL } V$
Циклический сдвиг вправо	$X' = X \text{ ROR } V$
Табличные подстановки	
S-box	$X' = \text{Table}[X, V]$

В качестве параметра V для любого из этих преобразований может использоваться число:

- 1) фиксированное (например, $X' = X + 125$);
- 2) получаемое из ключа (например, $X' = X + F(\text{Key})$);
- 3) получаемое из независимой части блока (например, $X_2' = X_2 + F(X_1)$).

1.3 Сеть Фейстеля

Сеть Фейстеля является дальнейшей модификацией описанного выше метода смешивания текущей части шифруемого блока с результатом некоторой функции,

вычисленной от другой независимой части того же блока. Эта методика получила широкое распространение, поскольку обеспечивает выполнение требования о многократном использовании материала ключа и исходного блока информации.

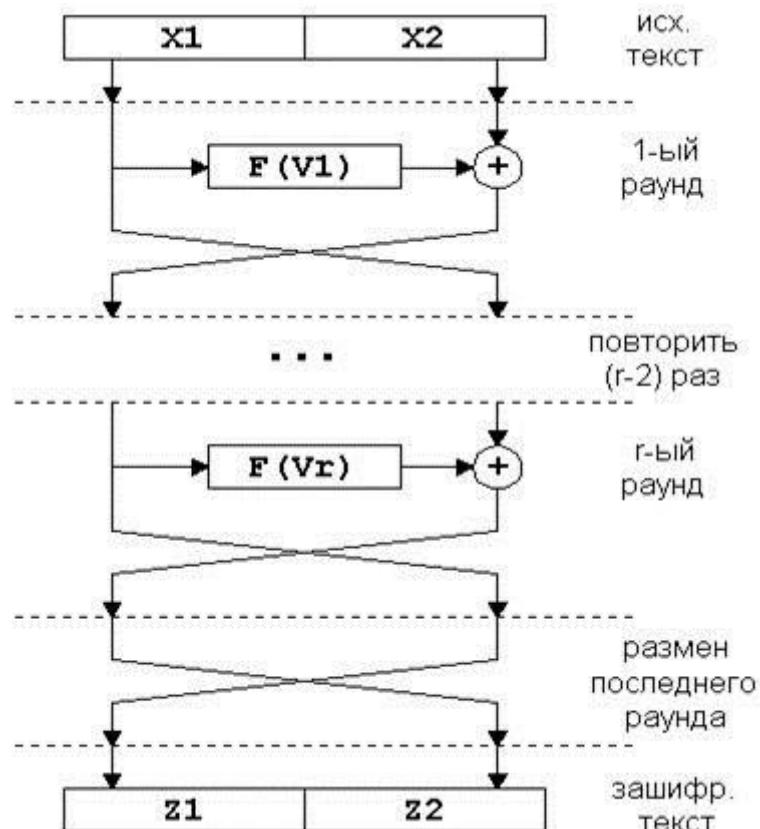


Рисунок 1.1 - Классическая сеть Фейстеля

Независимые потоки информации, появляющиеся из исходного блока, называются ветвями сети. В классической схеме их две. Величины V_i именуются параметрами сети, обычно это функции от материала ключа. Функция F является образующей. Действие, состоящее из однократного вычисления образующей функции и последующего наложения ее результата на другую ветвь с обменом их местами, называется циклом или раундом (англ. round) сети Фейстеля. Оптимальное число раундов $K - 8-32$. Важно то, что увеличение количества раундов значительно увеличивает криптостойкость любого блочного шифра к криптоанализу. [5]

2. Алгоритм AES

2.1 Описание алгоритма

Алгоритм Rijndael (читается «Рейндал») разработан бельгийскими специалистами Joan Daemen (Proton World International) и Vincent Rijmen (Katholieke Universiteit Leuven). Этот шифр победил в проведенном Национальным институтом стандартов и техники (NIST) США конкурсе на звание AES (Advanced Encryption Standard) и в 2001 году был принят в качестве нового американского стандарта. Алгоритм Rijndael достаточно сложен для описания, поэтому рассмотрим только основные аспекты построения и особенности использования шифра.

Шифр Rijndael/AES (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа. В принципе, структуру Rijndael можно приспособить к любым размерам блока и ключа, кратным 32, а также изменить число раундов.[4]

В отличие от шифров, предлагаемых DES и ГОСТ 28147-89, в основе Rijndael не лежит сеть Фейштеля. Основу Rijndael составляют так называемые линейно-подстановочные преобразования. Блок данных, обрабатываемый с использованием Rijndael, делится на массивы байтов, и каждая операция шифрования является байт-ориентированной. Каждый раунд состоит из трех различных обратимых преобразований, называемых слоями. Эти слои следующие.

1. Нелинейный слой. На этом слое выполняется замена байтов. Слой реализован с помощью S-блоков, имеющих оптимальную нелинейность, и предотвращает возможность использования дифференциального, линейного и других современных методов криптоанализа.

2. Линейный перемешивающий слой гарантирует высокую степень взаимопроникновения символов блока для маскировки статистических связей. На этом слое в прямоугольном массиве байтов выполняется сдвиг строк массива и

перестановка столбцов.

3. Слой сложения по модулю 2 с подключом выполняет непосредственно шифрование.

Шифр начинается и заканчивается сложением с ключом. Это позволяет закрыть вход первого раунда при атаке по известному тексту и сделать криптографически значимым результат последнего раунда.[4]

2.2 Математический аппарат

Блочные шифры могут использоваться для выполнения различных задач. Поэтому для любого симметричного блочного алгоритма шифрования определено несколько режимов его применения. Каждый из режимов имеет свои особенности и сферы применения.

Пусть имеется некоторый блочный шифр, который выполняет преобразование f исходного блока данных X с помощью ключа K в зашифрованный блок Y :

$$Y=f(X,K).$$

Рассмотрим некоторые возможные режимы выполнения преобразования f . Простейшим режимом является режим простой поблочной замены. Специалистами этот режим называется ECB - Electronic Code Book, что переводится как "электронная кодовая книга". В этом режиме каждый блок исходных данных шифруется независимо от остальных блоков, с применением одного и того же ключа шифрования. Если сообщение длиннее, чем длина блока соответствующего алгоритма, то оно разбивается на блоки X_1, X_2, \dots, X_n соответствующей длины, причём последний блок дополняется в случае необходимости фиксированными значениями. Каждый блок шифруется блочным шифром:

$$Y=f(X_i,K) \text{ для всех } i \text{ от } 1 \text{ до } n.$$

В результате шифрования всех блоков исходных данных X_i получается зашифрованное сообщение $Y=Y_1, Y_2, \dots, Y_n$. Расшифрование выполняется по правилу:

$$X=f^{-1}(Y_i,K) \text{ для всех } i \text{ от } 1 \text{ до } n.$$

Из определения режима ECB следует, что расшифрование сообщения можно производить, выбирая блоки шифротекста в произвольном порядке. Такой режим удобен во многих реальных ситуациях, частности для обработки файлов с произвольным доступом. Например, в режиме ECB можно работать с зашифрованной базой данных при условии, что каждая запись представляет собой отдельный блок данных и зашифрована отдельно от остальных.

Недостатком данного режима является то, что одинаковые блоки исходного текста преобразуются в одинаковый шифротекст. В большинстве реальных шифруемых наборах данных есть повторяющиеся элементы. Сообщения могут иметь высокую избыточность, повторяющиеся заголовки и окончания или содержать длинные серии нулей или пробелов. Таким образом, противник может получить в свое распоряжение данные для частотного криптоанализа. Более серьезной проблемой рассматриваемого режима является то, что противник может изменить или подменить шифрованные сообщения с целью обмана получателя.

В целом, режим ECB рекомендуется использовать для передачи одиночных коротких сообщений (например, криптографического ключа).

Для устранения недостатков ECB при передаче нескольких блоков данных может использоваться режим CBC (Cipher Block Chaining) – режим сцепления блоков шифра.

Преобразование в режиме CBC выполняется следующим образом: каждый блок открытого текста складывается по модулю 2 с результатом шифрования предыдущего блока. Таким образом, результаты шифрования предыдущих блоков влияют на шифрование следующих блоков. Математически операция шифрования в режиме CBC описывается следующей формулой:

$$Y = f((X_i \oplus Y_{i-1}),K) \text{ для всех } i \text{ от } 1 \text{ до } n.$$

То есть перед шифрованием очередного блока над открытым текстом и результатом шифрования предыдущего блока выполняется операция "сумма по модулю 2". Когда блок открытого текста зашифрован, он сохраняется в памяти шифрующего устройства, например, в регистре обратной связи. Перед

шифрованием следующего блока данных, он подвергается операции "сумма по модулю 2" вместе с регистром обратной связи и только после этого шифруется. Полученный зашифрованный блок снова сохраняется в регистре обратной связи и используется при шифровании следующего блока входных данных и так далее до конца сообщения. Блок Y_0 должен быть сформирован перед шифрованием первого блока исходных данных. Он называется вектором инициализации и используется для сложения по модулю 2 с блоком входных данных. В результате использования обратной связи шифрование каждого блока зависит от всех предыдущих блоков.

Зашифрованное сообщение можно расшифровать следующим образом:

$$X_i = Y_{i-1} \oplus f^{-1}(Y_i, K) \text{ для всех } i \text{ от } 1 \text{ до } n.$$

Блок шифротекста сначала сохраняется в регистре обратной связи, а затем расшифровывается как обычно. Далее расшифровывается следующий блок и подвергается операции "сумма по модулю 2" с регистром обратной связи. И так выполняется до конца сообщения.[4]

Шифрование происходит по следующей схеме:

Предварительно входные данные разбиваются на блоки по 16 байт, если полный размер не кратен 16 байтам, то данные дополняется до размера, кратного 16 байтам. Блоки представляются в виде матрицы 4x4 — state. Далее происходит процедура расширения ключа и к каждому блоку state применяются операции 2-4. Итак, алгоритм состоит из следующих шагов:

1. Расширение ключа - KeyExpansion;
2. Начальный раунд - сложение state с основным ключом;
3. 9 раундов шифрования, каждый из которых состоит из преобразований:
 - SubBytes;
 - ShiftRows;
 - MixColumns;
 - AddRoundKey;
4. Финальный раунд, состоящий из преобразований:
 - SubBytes;
 - ShiftRows;

- AddRoundKey.

Рассмотрим подробнее каждое из представленных выше преобразований:

SubBytes - замена байтов state по таблице S-box. Каждый байт представляется в виде двух шестнадцатеричных чисел $b = (x, y)$, где x определяется 4 старшими разрядами b , а y — 4 младшими. В таблице S-box размера 16×16 находятся значения для замены исходного байта: значение b' на пересечении строки x и столбца y S-box используется в качестве замены исходному байту b .

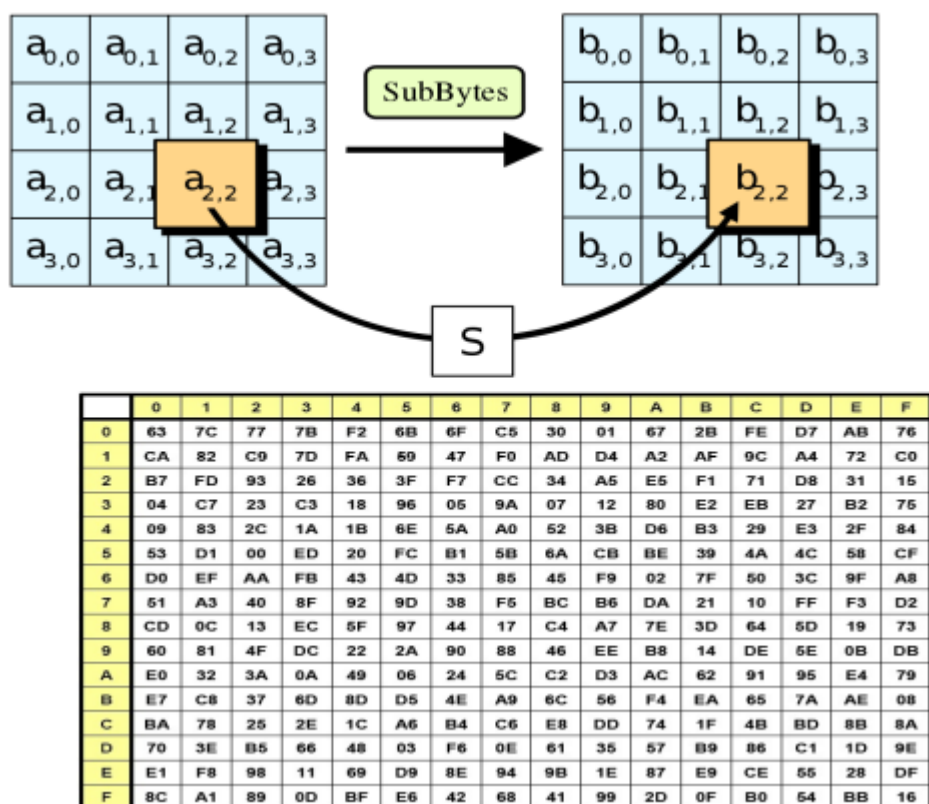


Рисунок 2.3 – Схема операции SubBytes

ShiftRows — циклический сдвиг строк state. Нулевая строка остается на месте, первая смещается влево на 1 байт, вторая на 2 байта и третья на 3 соответственно.

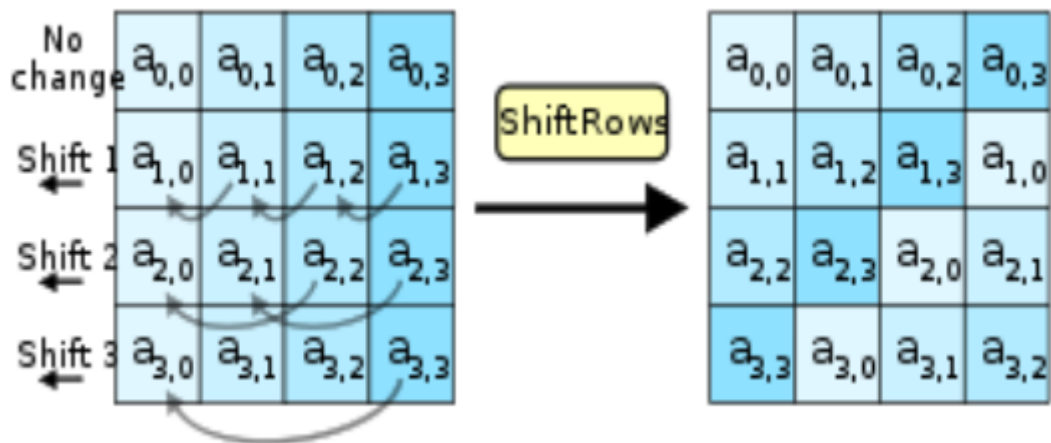


Рисунок 2.4 – Схема операции ShiftRows

MixColumns — умножения каждого столбца state на фиксированную матрицу. Таким образом осуществляется линейное преобразование над столбцами state. Причем умножение и сложение производится по правилам, описанным выше.

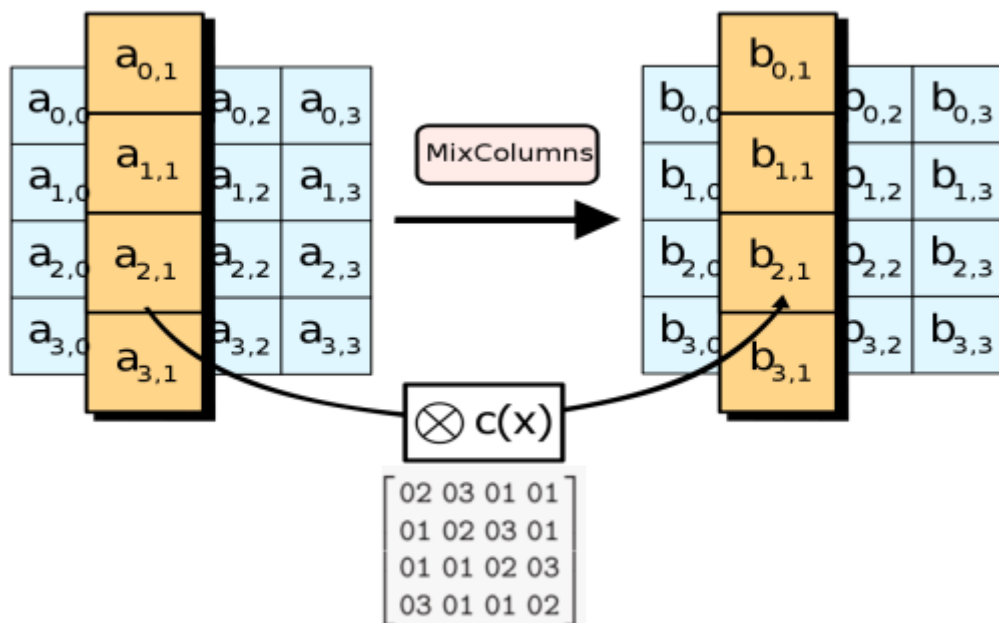


Рисунок 2.5 – Схема операции MixColumns

AddRoundKey — раундовый ключ поэлементно добавляется к state с помощью поразрядного XOR.

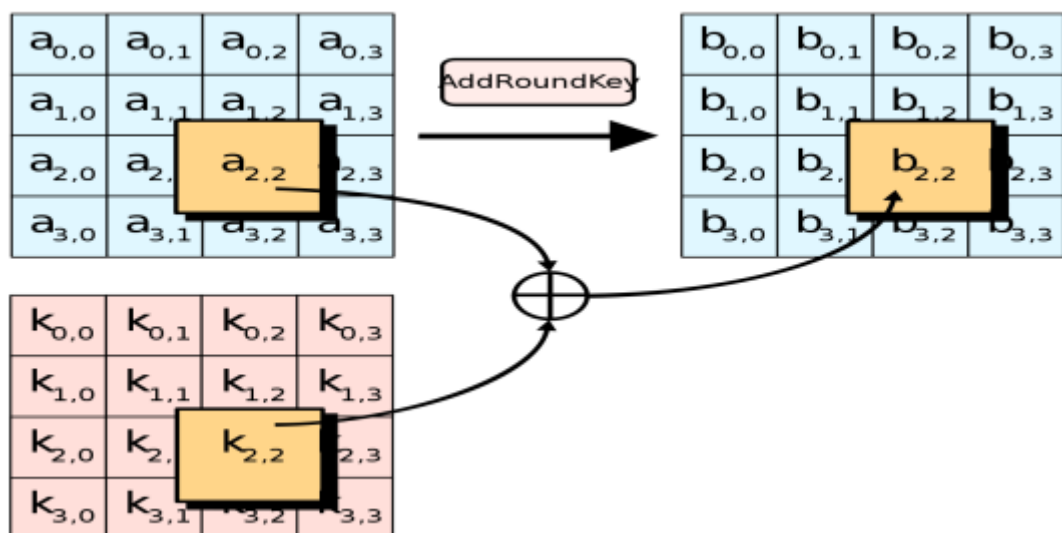


Рисунок 2.6 Схема операции AddRoundKey

KeyExpansion — процедура расширения основного ключа для создания раундовых ключей, которые затем используются в раундах шифрования. Расширенный ключ состоит из 44 четырехбайтовых слов (w_i): 4 слова на основной ключ и по 4 слова на 10 раундовых ключей. Таким образом, полная длина расширенного ключа составляет 1408 бит. [4]

Операция расширения ключа использует массив Rcon и состоит из следующих действий:

- Четыре слова основного ключа переносятся в первые четыре слова расширенного ключа;
- Если число i без остатка делится на 4, то $w_i = \text{SubBytes}(\text{RotByte}(w_{i-1})) \oplus \text{Rcon}_{i/4}$;
- Иначе: $w_i = w_{i-4} \oplus w_{i-1}$;

Операция RotByte производит циклическую перестановку байта исходного слова: $\{x_0, x_1, x_2, x_3\} \rightarrow \{x_3, x_0, x_1, x_2\}$.

Дешифровка происходит по следующей схеме:

Так как все преобразования шифрования выполняются однозначно, то существует обратное преобразование, с помощью которого шифротекст

переводится в открытый текст. Обратное преобразование представляет собой последовательность инвертированных операций шифрования, выполняемых в обратном порядке:

1. Расширение ключа - KeyExpansion;
2. 9 раундов дешифрования, каждый из которых состоит из преобразований:
 - AddRoundKey — суммирование state с раундовым ключом;
 - InverseMixColumns — обратная перестановка столбцов state;
 - InverseShiftRows — обратный циклический сдвиг столбцов state;
 - SubBytes — замена байтов state по обратной таблице замен InverseS-box;
3. Финальный раунд:
 - AddRoundKey;
 - InverseShiftRows;
 - InverseSubBytes.

2.3 Методы взлома

Необходимо выполнить следующие шаги $2^{25,5}$ раз:

1. Подготовить структуру открытых текстов, как указано ниже.
2. Зашифровать его на ключах K_A и K_B и сохранить полученные множества S_A и S_B .
3. Необходимо осуществить операцию xor Δc для всех шифротекстов в S_A и расшифровать изменённые шифротексты ключом K_C . S_C — новое множество открытых текстов.
4. Аналогично для S_B и ключа K_D . S_D — новое множество открытых текстов.
5. Сравнение всех пар открытых текстов из S_C и S_D с длиной 56 бит.
6. Для всех остальных выполнить проверку на различие значения вероятности $p_{i,0}$ при $i > 0$. Если оно равно с обеих сторон 16-битного фильтра, то для

пар ключей или ещё их называют квартетом (K_A, K_B) и (K_C, K_D) , при $\nabla k_{i,7}^0 = 0$, Δk_i^0 тоже будут равны с обеих сторон.

7. Необходимо отобрать квартеты, различия которых не могут быть затронуты активными S-блоками в первом раунде и активными S-блоками в алгоритме выработки ключа.

8. Отфильтровывая неправильные квартеты, частично восстановим значение ключа.

Каждая структура имеет всевозможные варианты значений нулевого столбца, нулевой строки и константное значение в других байтах. Из 2^{72} текстов в каждой структуре можно сравнить 2^{144} пар. Из этих пар $2^{(144-8*9)} = 2^{72}$ пройдут первый раунд. Таким образом, получаем 1 нужный квартет из $2^{(96-72)} = 2^{24}$ структур и 3 нужных квартета из $2^{25,5}$ структур. Вычислим количество квартетов прошедших 6 шагов, их будет около $2^{(144-56-16)} = 2^{72}$ пар. Следующим шагом будет применение 16-битного фильтра, так получим общее количество возможных квартетов $2^{(72+25,5-6)} = 2^{91,5}$.

3 Описание программного обеспечения

3.1 Состав и структура программного обеспечения

В программу «Cryptography.exe» включены 3 класса.

Класс «AES.cs» отвечает за математическую реализацию процессов шифрования/дешифрования, а также за преобразование ключей и сообщения для корректной работы программы, и содержит поля «AES Key», «AES IV», «encrypted» и «plaintext» для работы с преобразованными в байтовые массивы входными данными. Методы «Encrypt_Aes» и «Decrypt_Aes» содержат вызовы необходимых функций шифровки и дешифровки текста.

Класс «LogInForm» отвечает за графический интерфейс окна авторизации и соответствующее взаимодействие с неавторизованным пользователем.

Класс «CryptoProtocolForm» отвечает за графический интерфейс, взаимодействие с пользователем и первичное преобразование входных данных в виде ключа и сообщения. Методы отвечают за файловый ввод и вывод ключей и сообщений, генерацию ключей, сохранение текстов из поля вывода и т.д.

3.2 Программная реализация функций

Алгоритм шифрования/дешифрования текста реализован в классе AES при помощи криптографической библиотеки System.Security.Cryptography внутри класса объявлен метод «Encrypt_Aes» принимающий на вход строку и 2 ключа в формате byte[] первая Key длиной 32 байта записанных через “ “ и вторая IV длиной 16 байт записанных через “ “.

Далее генерируется ключ шифрования при помощи выражения «ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);».

После чего в крипто поток передаётся поток памяти и сгенерированный ключ, после чего крипто поток передаётся в StreamWriter и внутри StreamWriter шифруется строка через выражение «swEncrypt.Write(soursetext);», после чего считывается поток памяти объявленный ранее.

Дешифровка происходит внутри метода «Decrypt_Aes» в который передаётся строка байт, а также ключи шифровки, при помощи которых производилась шифровка.

Аналогично методу шифровки создаётся ключ шифровки при помощи выражения «ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);».

После чего объявляется поток памяти, в который предаётся зашифрованный массив байт. Далее внутри потока памяти объявляется крипто поток, в который передаётся экземпляр потока памяти и ключ шифровки. Внутри крипто потока объявляется StreamReader в который предаётся экземпляр крипто потока.

Внутри StreamReader при помощи выражения «plaintext = srDecrypt.ReadToEnd();» считывается дешифрованный текст.

3.3. Руководство пользователя

3.3.1 Введение

3.3.1.1 Программа «Cryptography.exe» предназначена для шифрования \ дешифрования текста, вывода зашифрованного \ дешифрованного текста в файл.

3.3.1.2 Программа предоставляет пользователю следующие возможности:

- добавление исходного текста;
- самостоятельный выбор ключей шифрования;
- случайную генерацию ключей шифрования;
- шифровка\дешифровка текста с помощью выбранного крипто протокола;
- вывод шифрованного\дешифрованного текста в файл.

3.3.2 Назначение и условия применения

3.3.2.1 Программа «Cryptography.exe» для оптимизации процесса шифровки\дешифровки текста.

Решение данной задачи призвано обеспечить:

- быструю и удобную работу с шифровкой\дешифровкой;
- исключение ошибок при вычислении шифротекста;
- удобный интерфейс, позволяющий работать с данными.

3.3.2.2 Программа реализована на языке C#. Работает в любой среде совместимой с Windows. Дискковой памяти для запуска программы требуется не менее 20 Мб. Оперативной памяти для нормальной работы программы требуется не менее 40 Мб.

3.3.3 Подготовка к работе

3.3.3.1 Система состоит из исполнительного файла «Cryptography.exe»

3.3.3.2 Запуск программы

Для установки программы нужно скачать файл «Cryptography.exe». После установки файла его можно запустить и начать работу.

3.3.4 Описание операций

3.3.4.1 Введение

Основные функции программы соответствуют генерации криптографических ключей для шифровки и дешифровки сообщений. Далее описаны все функции системы, а также формы и диалоги для ввода информации. Для каждой формы приведены основные компоненты и их назначение.

3.3.4.2 Окно входа в систему

После запуска программы на экран выводится окно, где необходимо ввести правильные данные и выполнить вход. Вид окна входа представлен на рисунке 3.2.

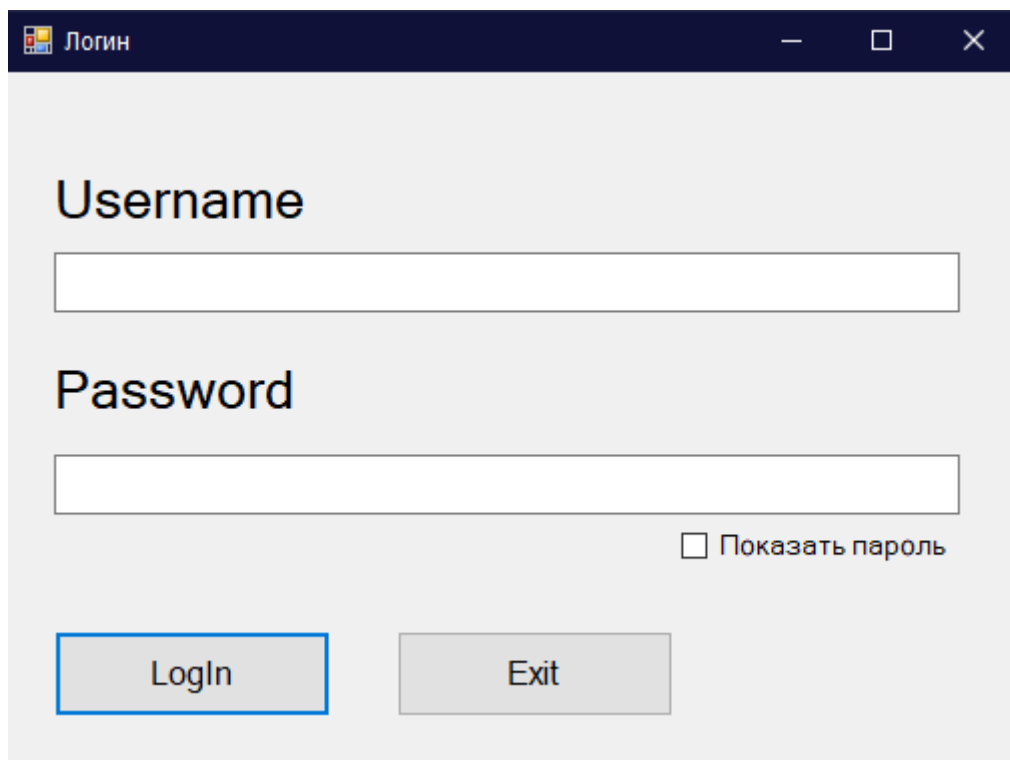


Рисунок 3.2 – Окно входа

В данном окне видим два поля для ввода:

«Username» и «Password».

Вводим данные, установленные по умолчанию:

Логин: admin;

Пароль: 1111.

После ввода данных для входа нажимаем на кнопку «Login».

3.3.4.3 Главное окно приложения

После успешного входа вы попадаете на основную «рабочую область» приложения, с которым вам предстоит работать. Вид окна приведён на рисунке 3.3.

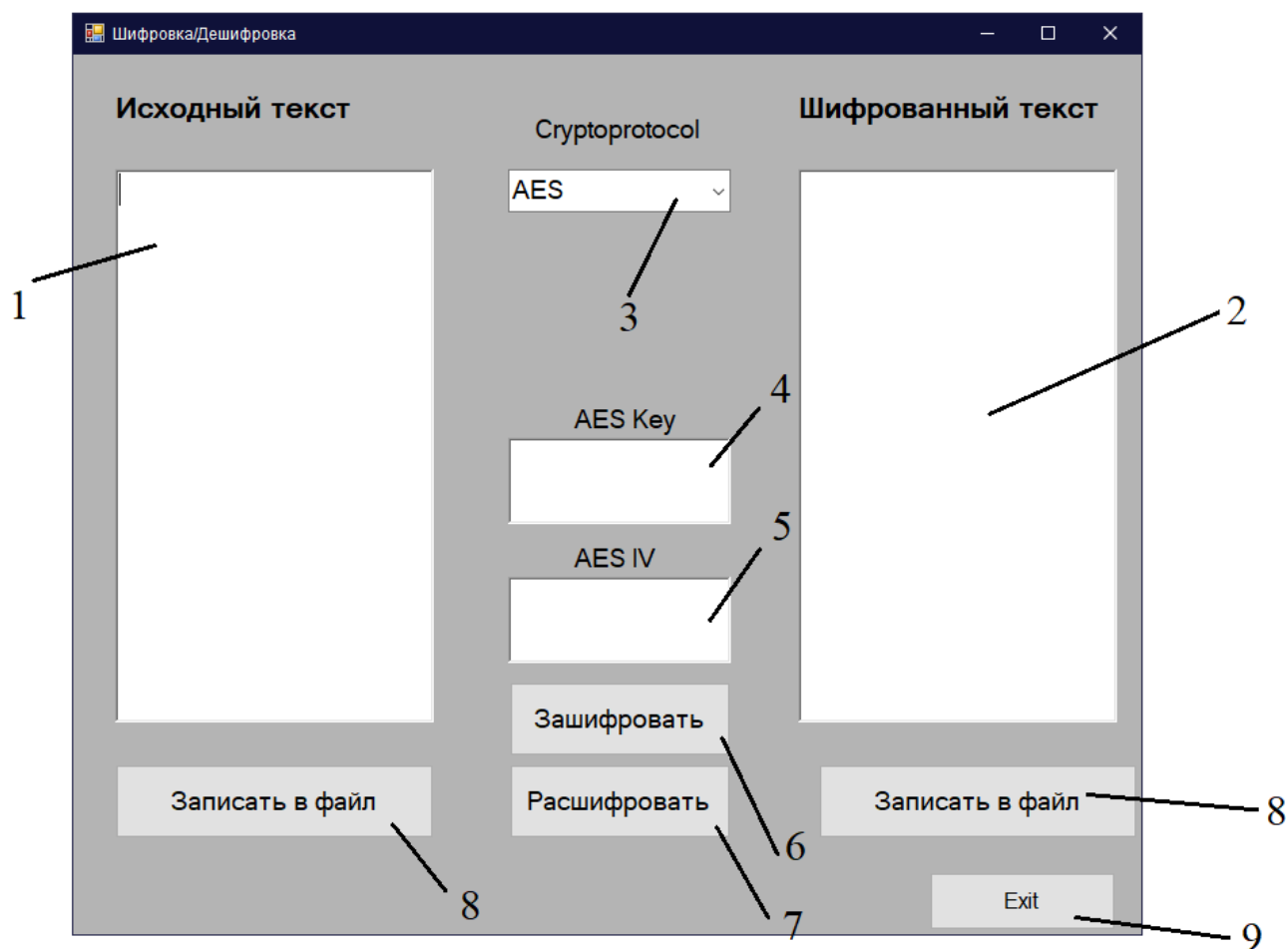


Рисунок 3.3 – Главное меню

Элементы основного окна:

- 1) Модуль ввода исходного текста и вывода расшифрованного текста.
- 2) Модуль ввода\вывода шифротекста.
- 3) Блок выбора крипто протокола.
- 4) Модуль ввода\вывода первого ключа шифровки.
- 5) Модуль ввода\вывода второго ключа шифровки.
- 6) Кнопка шифровки текста.
- 7) Кнопка дешифровки шифротекста.
- 8) Кнопка записи текста в файл.

Таблица 3.1 - Функциональные кнопки

Название	Действие системы
Зашифровать	Выполняет шифровку текста из поля 1 со случайными ключами из поля 4 и 5, если они не были заданы заранее
Расшифровать	Выполняет дешифровку текста из поля 2 с помощью ключей из поля 4 и 5
Записать в файл	Выполняет запись крипто ключей из поля 4 и 5, и шифрованного\дешифрованного из поля 2\1 в .txt файл
Exit	Выполняет выход из программы

При нажатии на кнопку зашифровать мы увидим окно, представленное на рисунке 3.4.

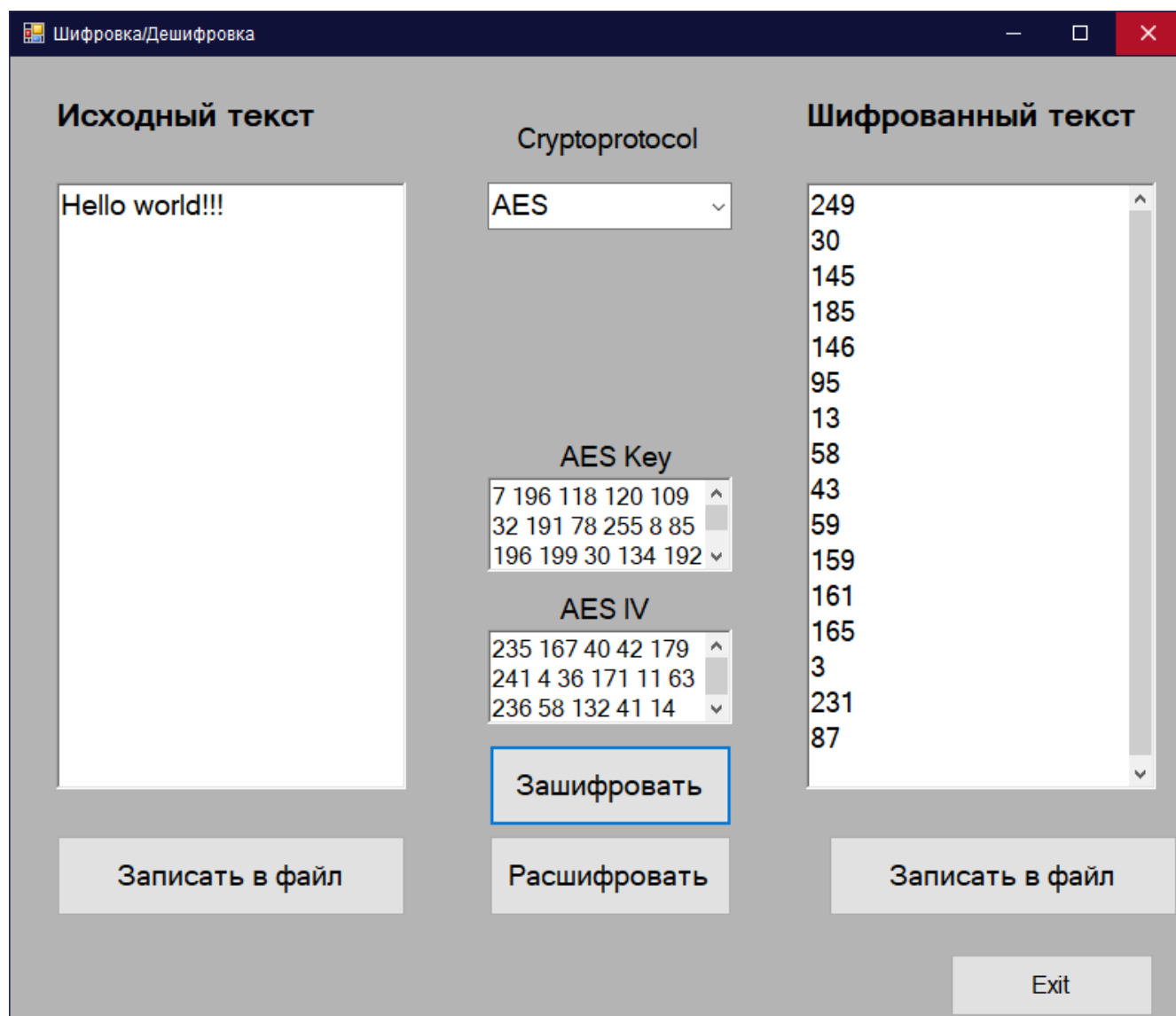


Рисунок 3.4 – Окно с шифрованным текстом

При нажатии на кнопку расшифровать мы увидим окно, представленное на рисунке 3.5.

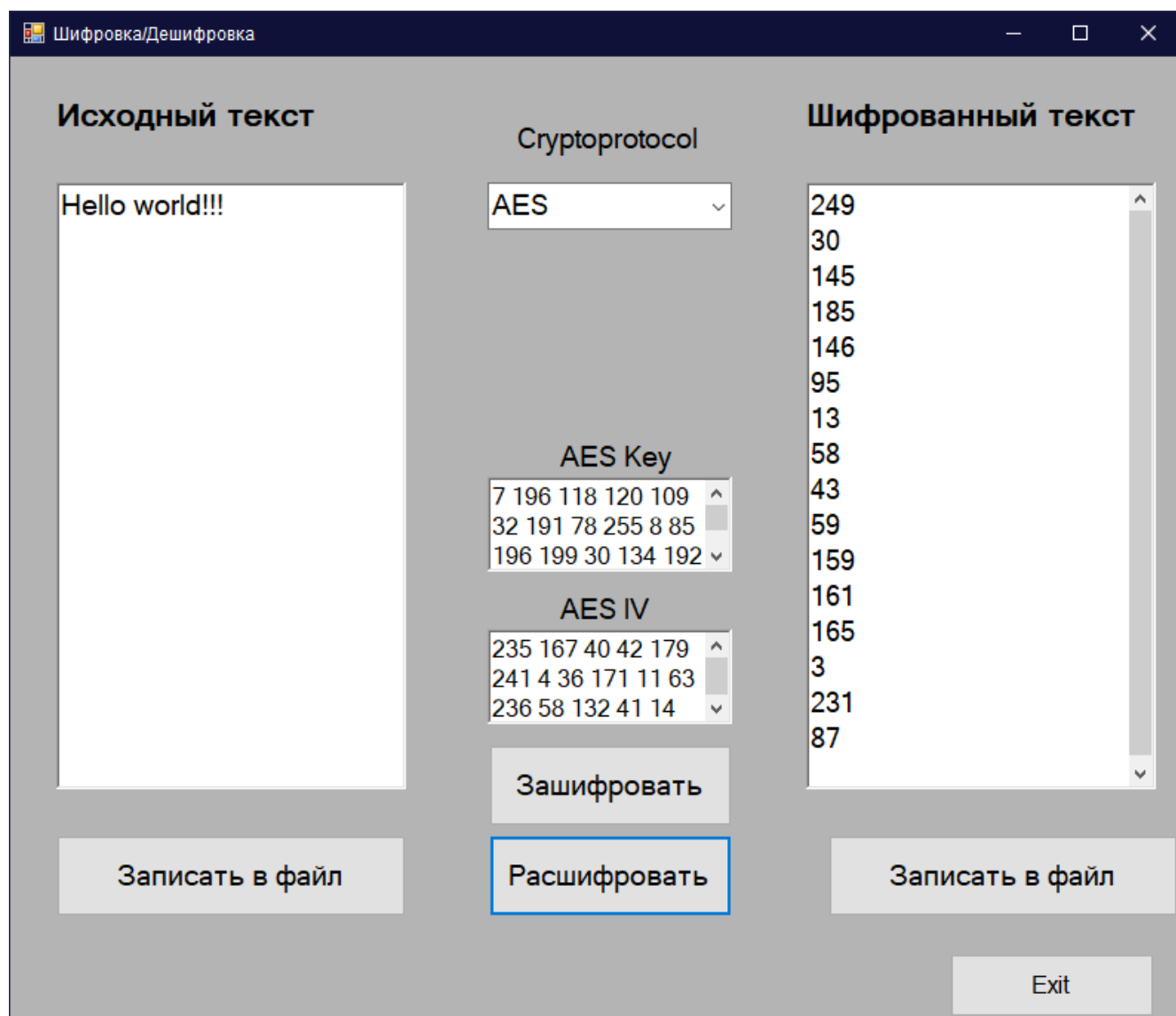


Рисунок 3.5 – Окно с дешифрованным текстом

При нажатии на кнопку записать в файл мы увидим окно, представленное на рисунке 3.6.

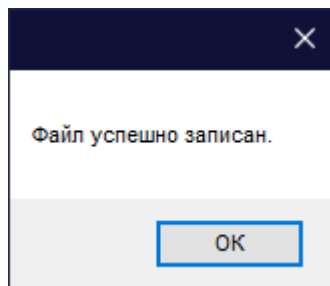


Рисунок 3.6 – Сообщение о записи файла

3.3.4.3.1 Первый модуль программы



The image shows a graphical user interface for a program. At the top, there is a header bar with the text "Исходный текст" (Original text). Below this header is a large, empty rectangular text input field. The number "1111" is entered at the top left of this field. At the bottom of the interface, there is a button with the text "Записать в файл" (Save to file).

Рисунок 3.7 – Модуль ввода\вывода исходного\дешифрованного текста

Первый модуль- модуль ввода\вывода исходного\дешифрованного текста (см. рисунок 3.7).

На рисунке 3.7 представлен образец заполнения поля исходный текст.

3.3.4.3.2 Второй модуль программы

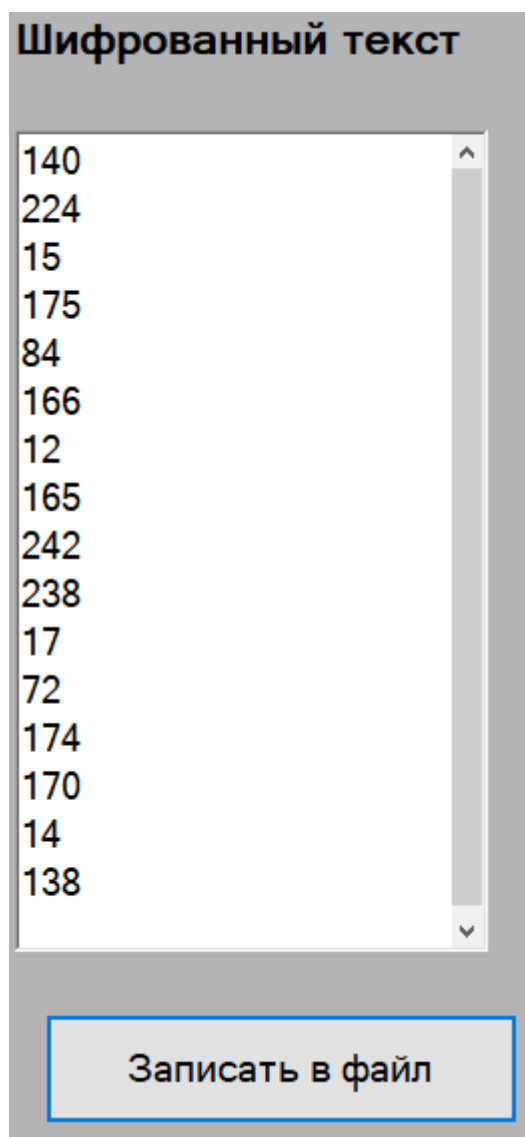


Рисунок 3.8 – Модуль ввода\вывода шифрованного текста

Второй модуль – модуль ввода\вывода шифрованного текста (см. рисунок 3.8).

На рисунке 3.8 представлен образец заполнения поля шифрованный текст.

3.3.4.3.3 Третий модуль программы

The screenshot shows a software interface for cryptographic operations. At the top, the title 'Cryptoprotocol' is displayed. Below it is a dropdown menu currently set to 'AES'. Further down, there are two input sections. The first is labeled 'AES Key' and contains a text box with the hexadecimal value '77 218 133 72 91 86 63 121 216 182 228 7 147 134 59'. The second section is labeled 'AES IV' and contains a text box with the hexadecimal value '43 137 253 102 125 206 132 228 74 72 247 30 133 65 170'. Both text boxes have small up and down arrow icons on their right sides, indicating they are scrollable or have a dropdown component.

Рисунок 3.9 – Модуль выбора крипто протокола и ввода ключей шифрования

Третий модуль - Модуль выбора крипто протокола и ввода ключей шифрования. Ключи могут генерироваться как самостоятельно, так и могут быть введены вручную (см. рисунок 3.9).

На рисунке 3.9 представлен образец заполнения полей «AES Key», «AES IV».

3.3.4.3.4 Четвёртый модуль программы

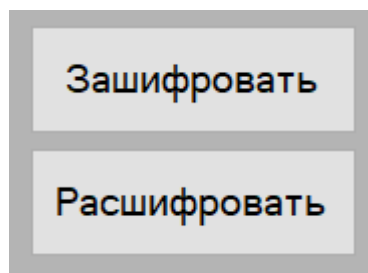


Рисунок 3.10 – Модуль вызова функций шифровки\дешифровки

Четвёртый модуль – Модуль выбора зашифровать или расшифровать текст (см. рисунок 3.10).

3.3.5 Сообщение пользователю

При работе с программой могут появиться следующие сообщения, представленные в таблице 3.2.

Таблица 3.2 – Сообщение пользователю

Текст сообщения программы	Ответ пользователя
Неверный логин или пароль!!!	Ввести верный логин или пароль
Не введён текст для шифровки	Ввести текст в поле «Исходный текст»
Не введён текст для дешифровки	Ввести текст в поле «Шифрованный текст»
Файл успешно записан.	Текст успешно записан в файл
Введён неверный ключ шифровки	Введен неверный ключ в поле «AES Key» или «AES IV»
Ошибка шифровки \ дешифровки.	Введён неверный текст или ключ шифровки

3.3.6 Аварийные ситуации

-

4 Руководство системного программиста

4.1 Общие сведения

Программа «Cryptography.exe» предназначена для шифрования дешифрования текста, вывода зашифрованного \ дешифрованного текста в файл.

Программа реализована на языке C#. Работает в любой среде совместимой с Windows. Дискетной памяти для запуска программы требуется не менее 20 Мб. Оперативной памяти для нормальной работы программы требуется не менее 40 Мб.

4.2 Структура программы

Программа реализована на языке C# в среде Visual Studio, основанном на визуальном построении приложений (помещение компонентов на формы и изменение их свойств и методов).

Модули программы:

Program – главный модуль программы, начало программы;

LogIn Form - модуль, предназначенный для входа в Crypto Protocol Form;

Crypto Protocol Form – модуль предназначенный для вызова криптографических функций программы;

AES - модуль отвечающий за шифровку\дешифровку текста;

Процедуры и функции программы:

Модуль LogInForm содержит следующие функции и процедуры:

public LogInForm() – конструктор формы;

private void button2_Click(object sender, EventArgs e) – метод для выхода из программы;

private void Login_Click(object sender, EventArgs e) метод входа в основную часть программы;

private void checkBox1_CheckedChanged(object sender, EventArgs e) метод

показывающий или скрывающий пароль.

Модуль CryptoProtocolForm содержит следующие функции и процедуры:

`public CryptoProtocolForm()` – конструктор формы;

`private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)` – метод проверяющий выбранный крипто протокол и отображающий соответственные поля;

`private void Encrypt_Click(object sender, EventArgs e)` - метод вызывающий класс крипто протокола, и методы шифровки текста;

`private void Exit_Click(object sender, EventArgs e)` – метод для выхода из программы;

`private void Decrypt_Click(object sender, EventArgs e)` - метод вызывающий класс крипто протокола, и методы дешифровки текста;

`private void Encrypt_to_file_Click(object sender, EventArgs e)` – метод записи зашифрованного текста и ключей в файл;

`private void button1_Click(object sender, EventArgs e)` - метод записи дешифрованного текста и ключей в файл.

Модуль AES содержит следующие функции и процедуры:

`public string[] encryptKeyToString(byte[] Key)` – метод преобразующий `byte[]` в `string[]` и возвращающий `string[]`;

`public string[] encryptIVToString(byte[] IV)` - метод преобразующий `byte[]` в `string[]` и возвращающий `string[]`;

`public string[] encrypttextToString(byte[] encrypt)` - метод преобразующий `byte[]` в `string[]` и возвращающий `string[]`;

`public byte[] StringToByte (string s)` - метод преобразующий `string` в `byte[]` и возвращающий `string[]`;

`public byte[] Encrypt_Aes(string soursetext, byte[] Key, byte[] IV)` – метод производящий шифровку исходного текста и возвращающий `byte[]`;

`public string Decrypt_Aes(byte[] cipherText, byte[] Key, byte[] IV)` - метод производящий шифровку исходного текста и возвращающий `string`.

4.3 Сообщение системному программисту

Сообщение системному программисту приведено на рисунке 4.1.

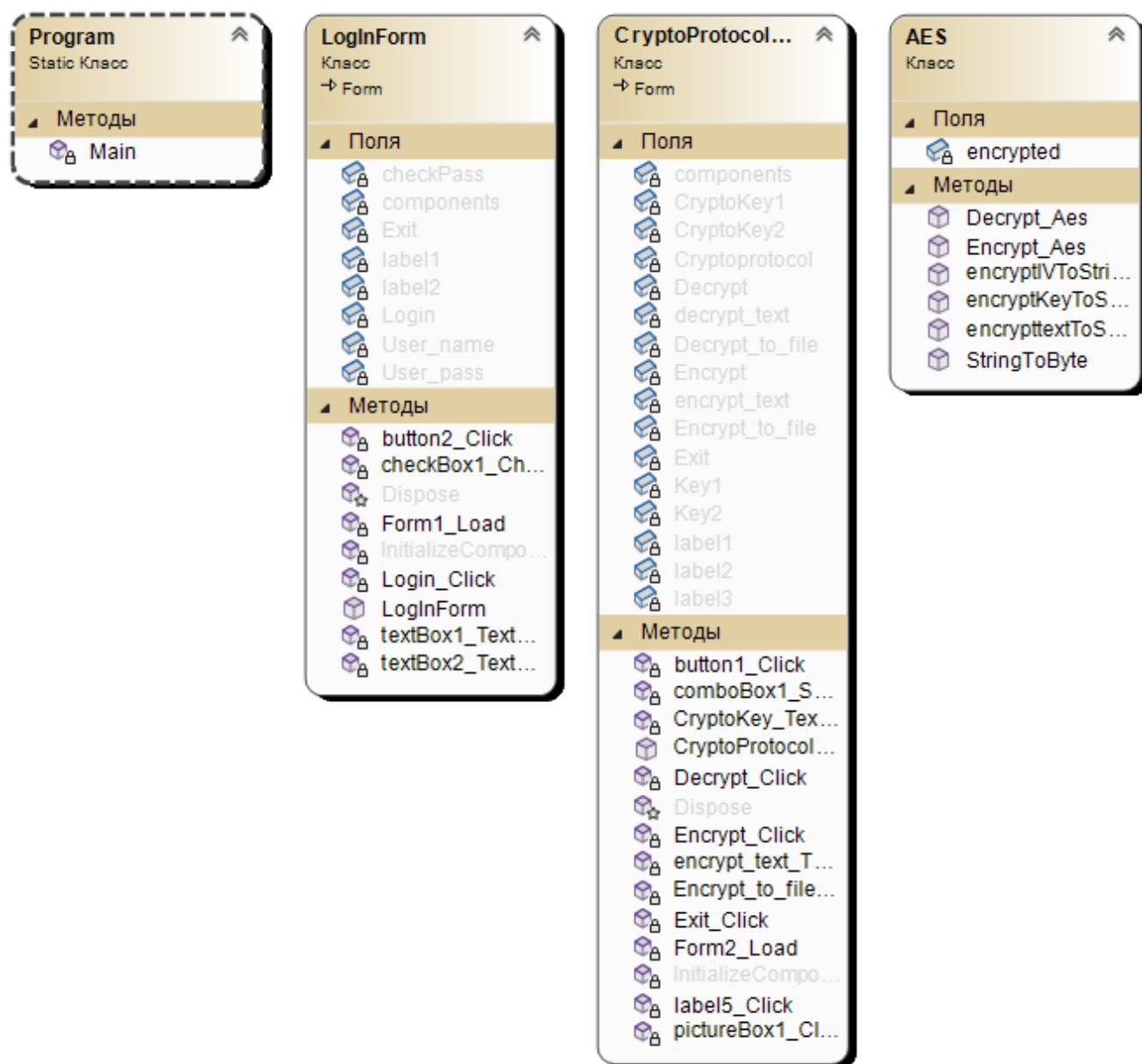
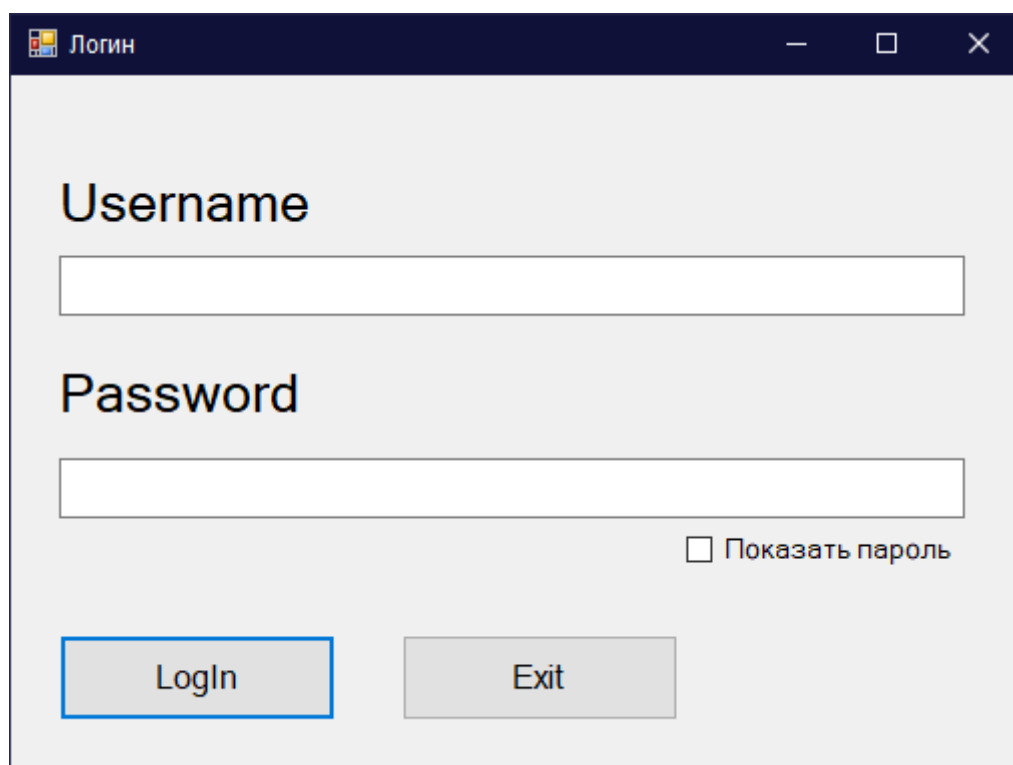


Рисунок 4.1 – UML диаграмма программы

5 Контрольный пример

5.1 Запуск программы

5.1.1 После запуска на экран выводится окно входа, вид которого представлен на рисунке 5.1.



The image shows a Windows-style login window. The title bar is dark blue and contains the text 'Логин' (Login) on the left and standard minimize, maximize, and close buttons on the right. The main content area has a light gray background. It features two text input fields: the first is labeled 'Username' and the second is labeled 'Password'. Below the password field, there is a checkbox with the label 'Показать пароль' (Show password). At the bottom of the window, there are two buttons: 'Login' and 'Exit'.

Рисунок 5.1 – Окно входа

5.1.2 Для входа в программу необходимо ввести в поле Username «admin» и в поле Password «1111», после нажать кнопку «LogIn», как указано на рисунке 5.2.

Рисунок 6.2 – Пример заполнения окна входа

5.1.3 После нажатия кнопки «LogIn» мы попадаем в основное окно программы вид которого представлен на рисунке 5.3.

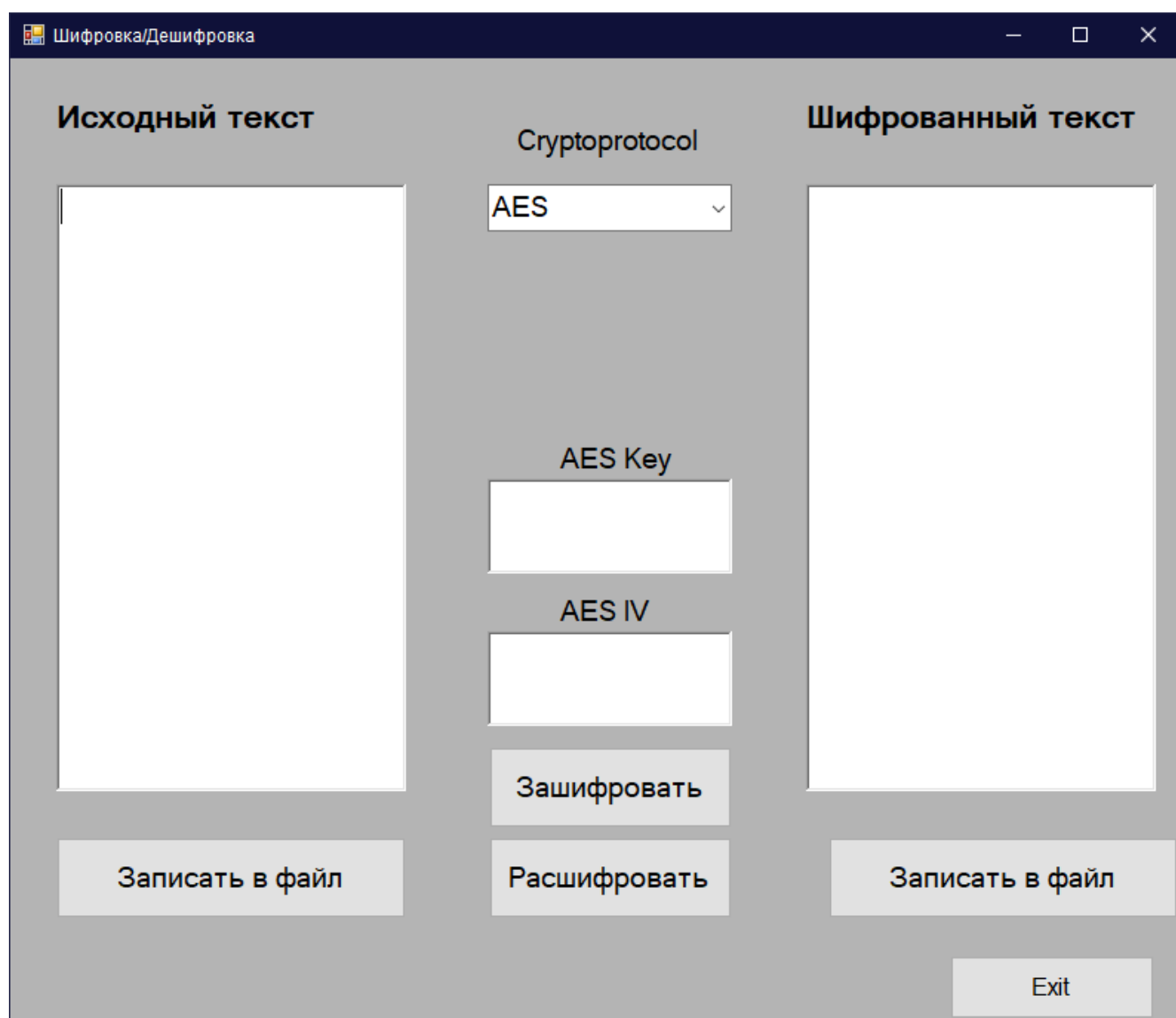


Рисунок 5.3 – Главное окно программы

5.1.4 Для начала шифровки необходимо заполнить поле «Исходный текст», как представлено на рисунке 5.4.

The image shows a Windows application window titled "Шифровка/Дешифровка". The window has a dark blue title bar with standard minimize, maximize, and close buttons. The main interface is divided into three vertical sections. The left section, titled "Исходный текст", contains a large text area with the text "Hello word!!!". Below this text area is a button labeled "Записать в файл". The middle section, titled "Cryptoprotocol", contains a dropdown menu currently showing "AES", followed by two empty input fields labeled "AES Key" and "AES IV". Below these are two buttons: "Зашифровать" and "Расшифровать". The right section, titled "Шифрованный текст", contains a large empty text area. Below this area is a button labeled "Записать в файл". At the bottom right of the window is an "Exit" button.

Рисунок 5.4 – Пример заполнения поля «Исходный текст»

5.1.5 Далее при необходимости можно заполнить поля «AES Key» (строка из 32-х байт записанная через пробел) и «AES IV» (строка из 16-ти байт через пробел), как показано на рисунке 5.5.

Шифровка/Дешифровка

Исходный текст

Hello word!!!

Cryptoprotocol

AES

AES Key

20 59 234 92 219
65 16 153 18 53 96
0 2 238 199 50 15

AES IV

247 58 169 230 232
116 149 118 67 78
245 217 132 113

Шифрованный текст

Зашифровать

Расшифровать

Записать в файл

Записать в файл

Exit

Рисунок 5.5 – Пример заполнения полей «AES Key» и «AES IV» вручную

Либо оставить поля «AES Key» и «AES IV» пустыми, в таком случае значения будут сгенерированы автоматически (генерация происходит после нажатия кнопки зашифровать), как на рисунке 5.6.

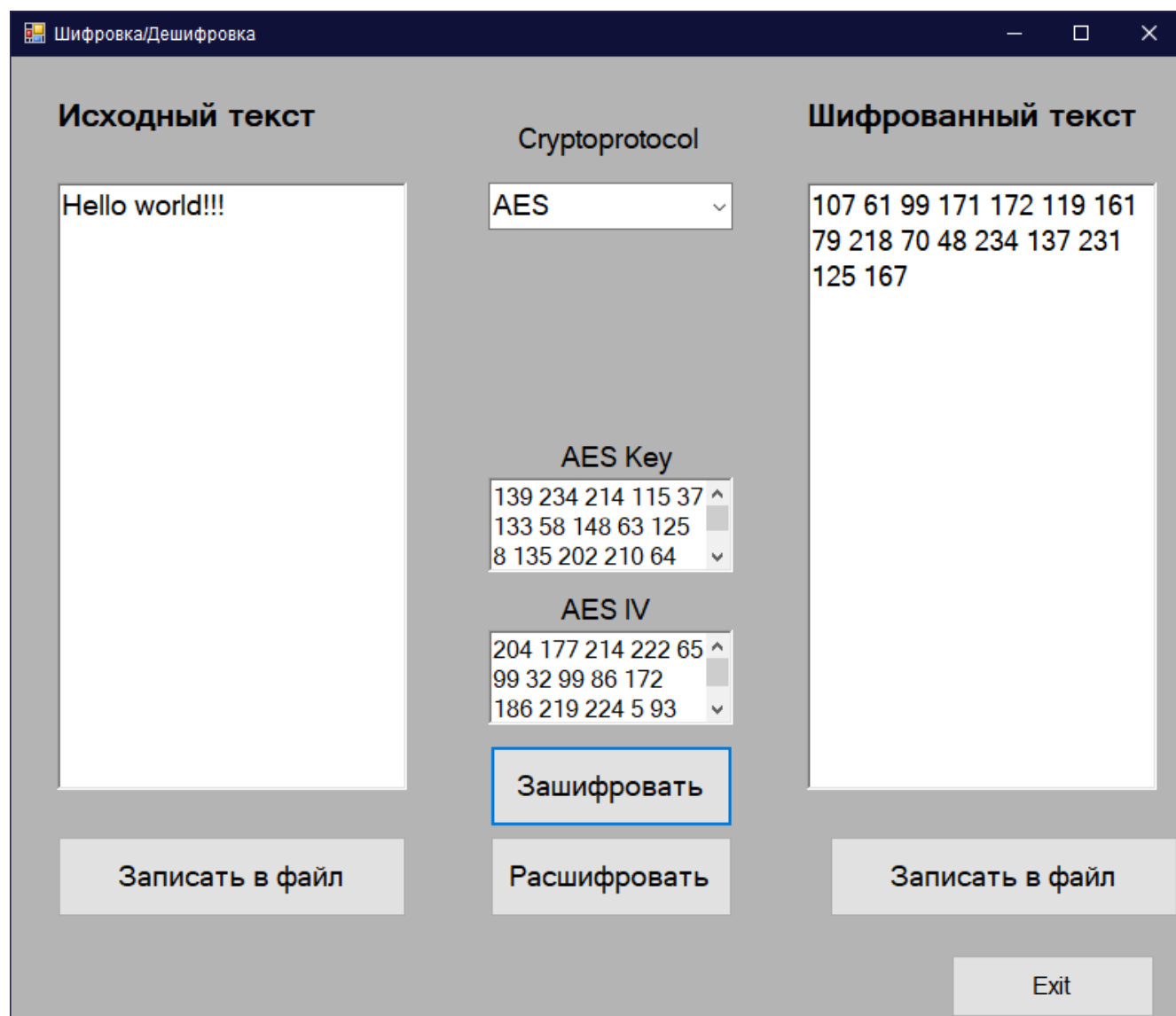


Рисунок 5.6 – Пример автоматической генерации ключа

5.1.6 После введения или автоматической генерации ключей, для шифровки текста необходимо нажать кнопку «Зашифровать», как показано на рисунке 5.7.

The screenshot shows a software window titled "Шифровка/Дешифровка" (Encryption/Decryption). The interface is divided into three main sections: "Исходный текст" (Original text), "Cryptoprotocol", and "Шифрованный текст" (Encrypted text).
- In the "Исходный текст" section, the text "Hello world!!!" is entered.
- In the "Cryptoprotocol" section, "AES" is selected from a dropdown menu.
- Below the protocol, there are two input fields for "AES Key" and "AES IV", each containing three lines of hexadecimal values.
- In the "Шифрованный текст" section, the encrypted text is displayed as:
109 55 57 112 107 138 221
187 238 160 71 199 130 74
226 57
- At the bottom, there are four buttons: "Записать в файл" (Save to file) under the original text, "Зашифровать" (Encrypt) in the center, "Расшифровать" (Decrypt) under the encrypted text, and "Exit" at the bottom right.

Рисунок 5.7 – Шифровка текста

5.1.7 После шифровки сообщения можно сохранить полученный шифр в текстовый файл, нажав на кнопку «Записать в файл», как показано на рисунке 5.8.

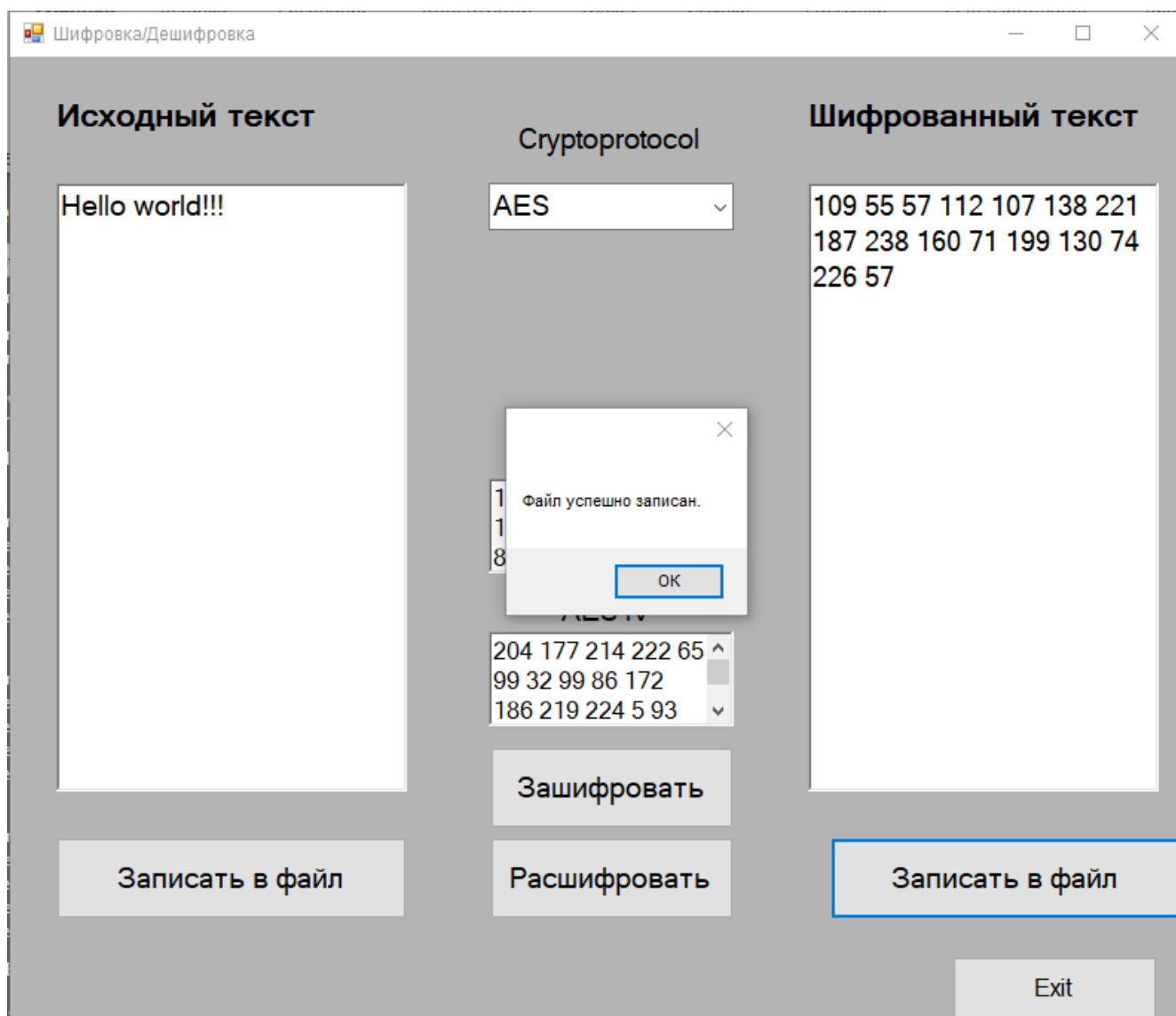


Рисунок 5.8 – Успешная запись в файл

В случае успешной записи шифротекста в файл, появиться сообщение, «Файл успешно записан.», как показано на рисунке 5.8.

Файл с записанным текстом появляется в директории программы, записанный текст показан на рисунке 5.9.

Шифрованный текст записывается в файл «Encrypt text.txt».

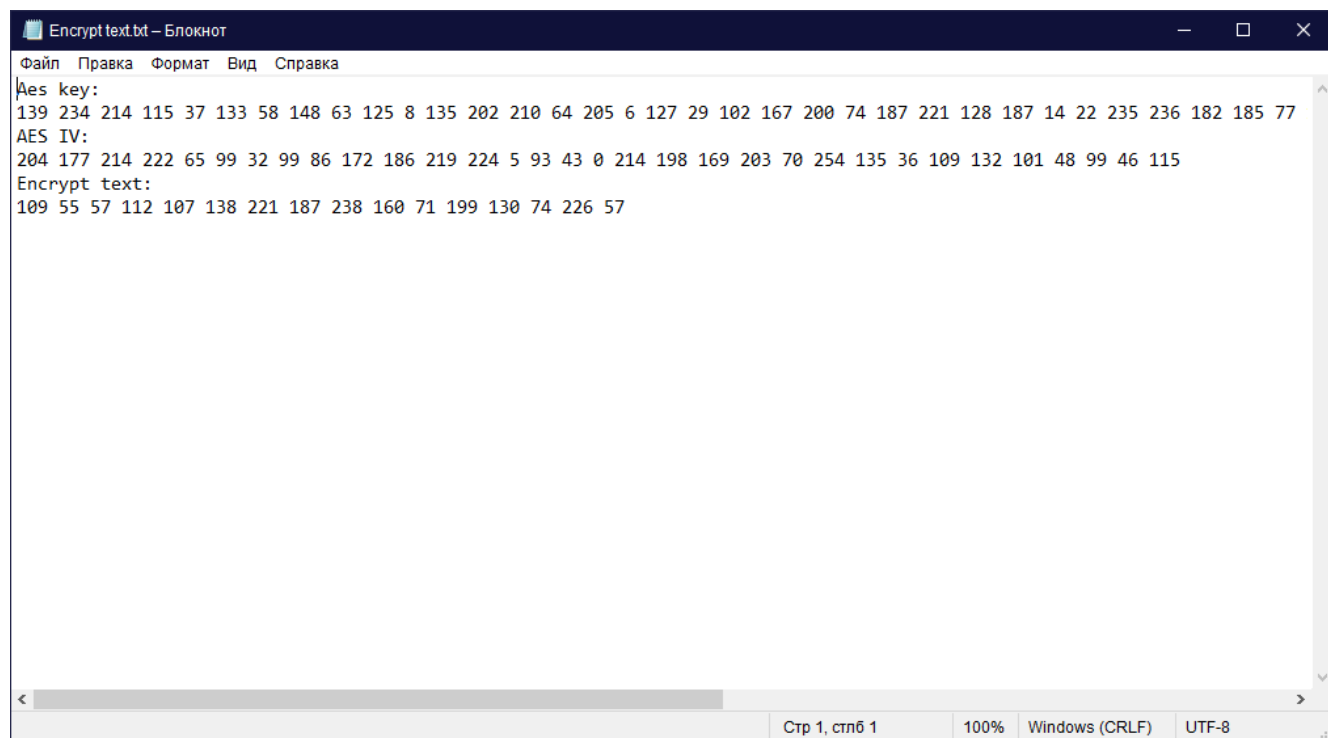


Рисунок 5.9 – Записанный текст в файле «Encrypt text.txt»

5.1.8 Для дешифровки необходимо ввести зашифрованный текст в формате строка байтов, записанных через пробел, как показано на рисунке 5.10.

Шифровка/Дешифровка

Исходный текст

Cryptoprotocol

AES

AES Key

AES IV

Зашифровать

Расшифровать

Записать в файл

Шифрованный текст

109 55 57 112 107 138 221
187 238 160 71 199 130 74
226 57

Записать в файл

Exit

Рисунок 5.10 – Пример ввода зашифрованного текста

5.1.9 Далее необходимо ввести ключи шифровки, полученные при шифровании текста в поля «AES Key» (строка из 32-х байт записанная через пробел) и «AES IV» (строка из 16-ти байт через пробел), как показано на рисунке 5.11.

Шифровка/Дешифровка

Исходный текст

Шифрованный текст

Cryptoprotocol

AES

AES Key

139 234 214 115 37
133 58 148 63 125
8 135 202 210 64

AES IV

109 132 101 48 99
46 115

Зашифровать

Расшифровать

Записать в файл

Записать в файл

Exit

109 55 57 112 107 138 221
187 238 160 71 199 130 74
226 57

Рисунок 5.11 – Пример заполнения полей «AES Key» и «AES IV»

5.1.10 Далее для дешифровки сообщения необходимо нажать кнопку «Расшифровать», как показано на рисунке 5.12.

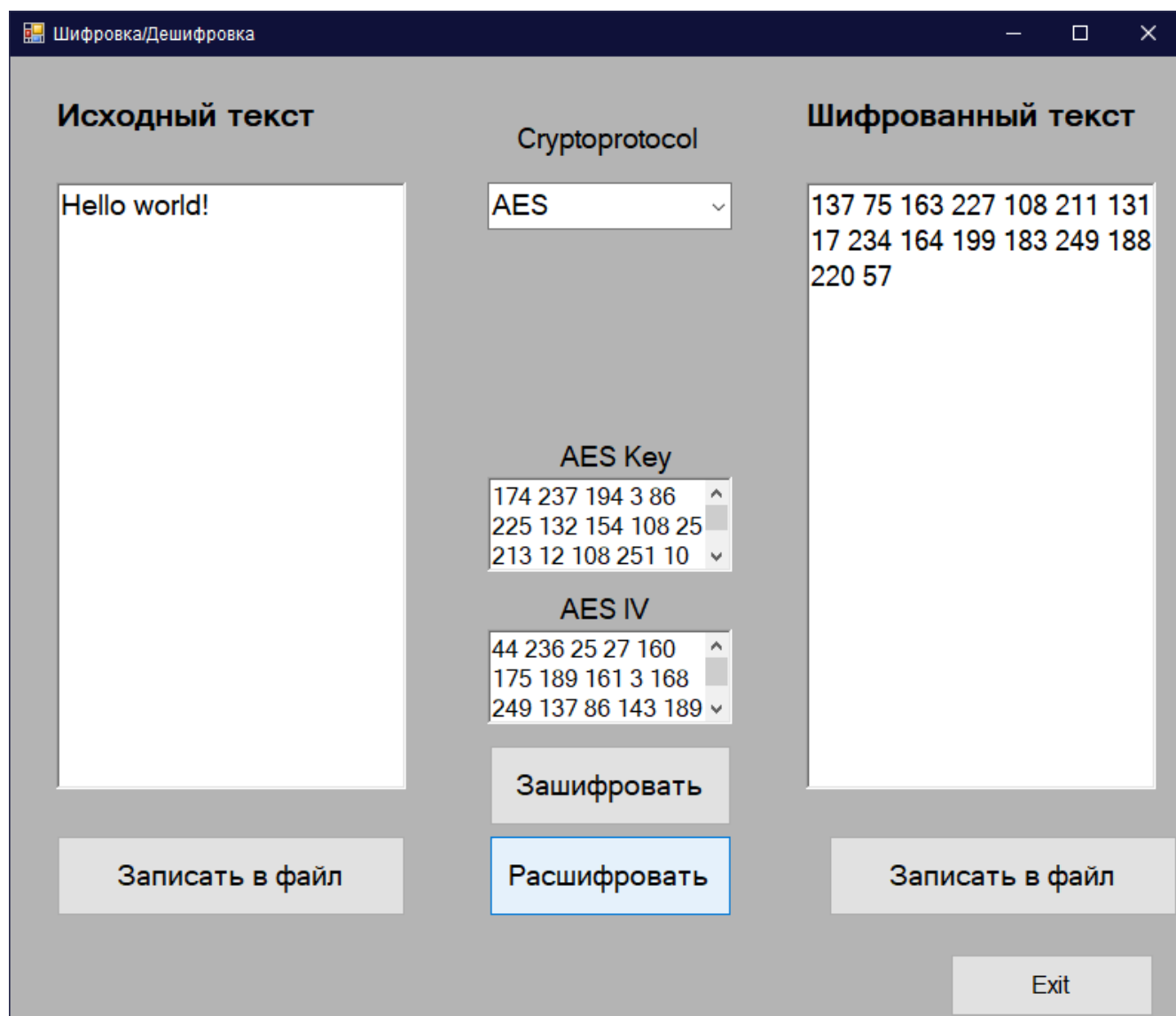


Рисунок 5.12 – Пример дешифровки

5.1.11 После шифровки сообщения можно сохранить полученный шифр в текстовый файл, нажав на кнопку «Записать в файл», как показано на рисунке 5.13.

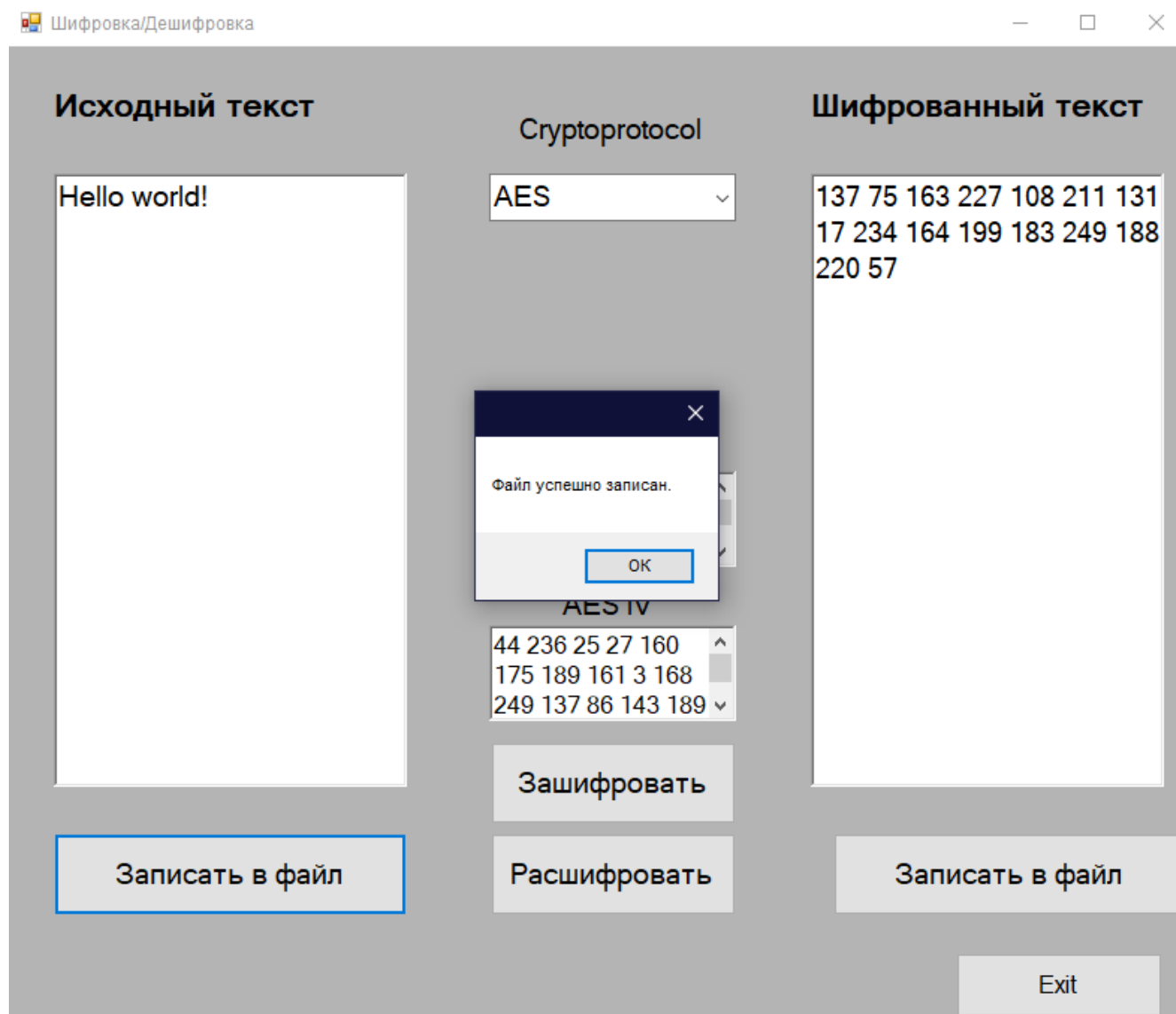


Рисунок 5.13 – Успешная запись в файл

В случае успешной записи шифротекста в файл, появиться сообщение, «Файл успешно записан.», как показано на рисунке 5.13.

Файл с записанным текстом появляется в директории программы, записанный текст показан на рисунке 5.14.

Шифрованный текст записывается в файл «Decrypt text.txt».

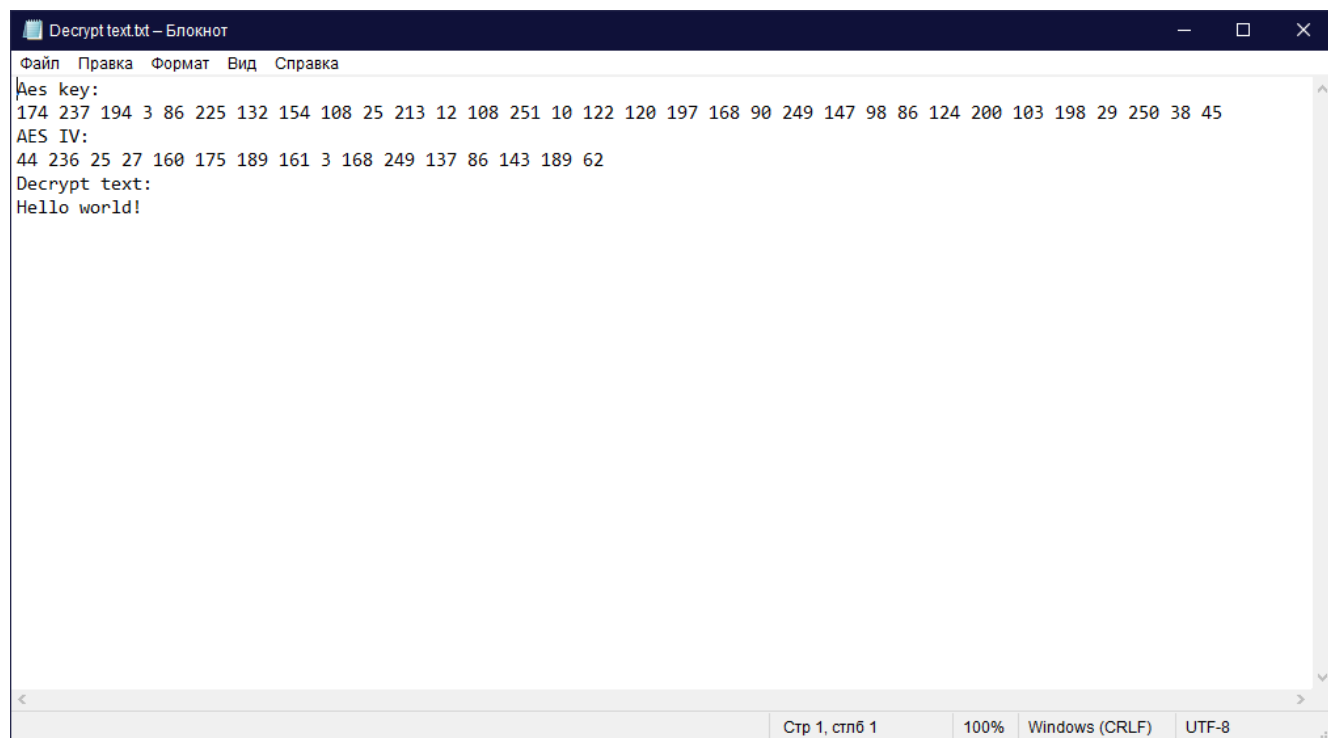


Рисунок 6.14 – Записанный текст в файле «Decrypt text.txt»

5.1.12 Для выхода из программы необходимо нажать кнопку «Exit», как показано на рисунке 5.15.

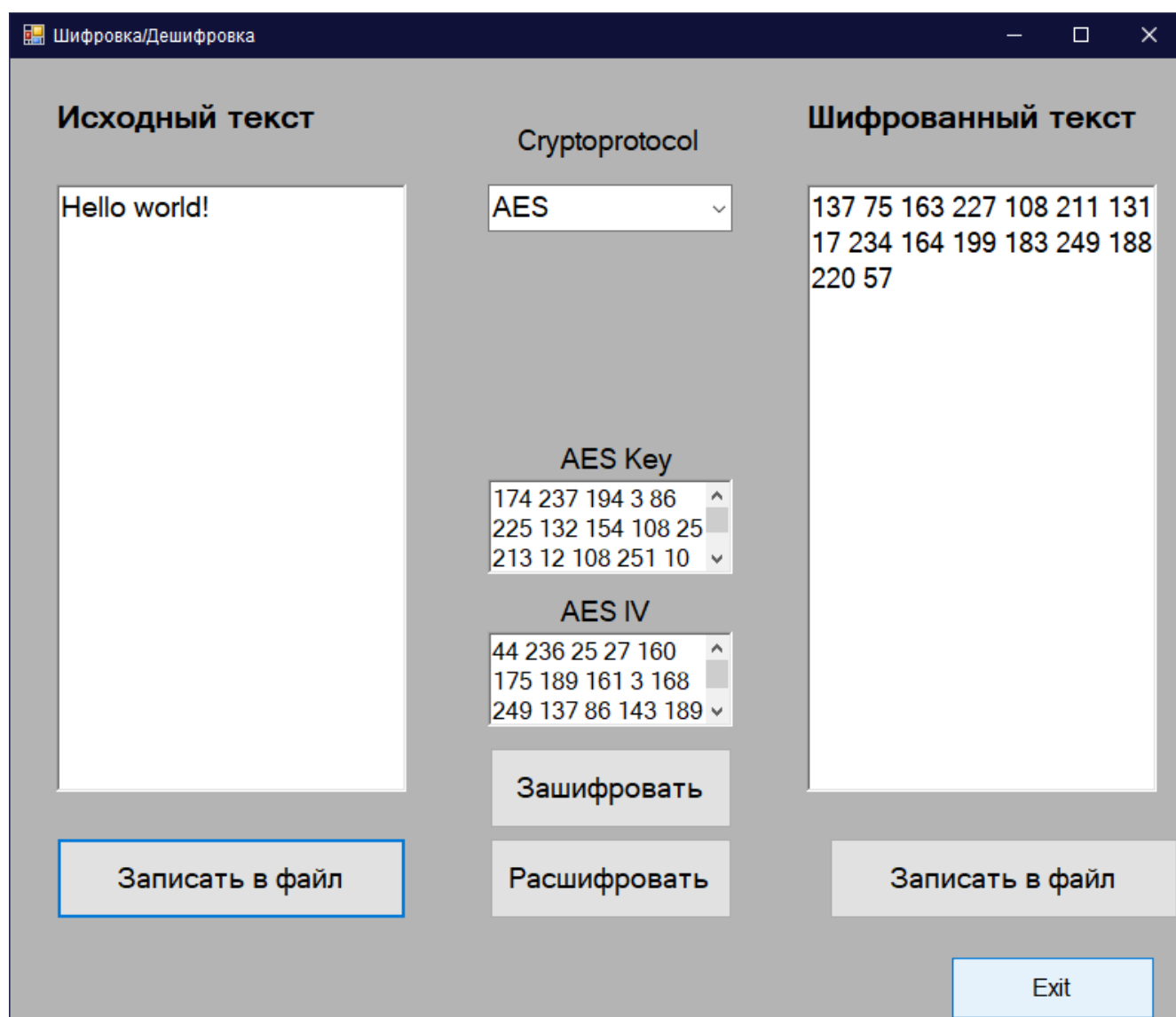


Рисунок 5.15 – Выход из программы

ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы были изучены: работа с Windows Forms на С# (работа с элементами формы, их свойствами, создание дополнительных форм, добавление к элементам формы какого-либо функционала и т.п.), а также крипто протоколы библиотеки System.Security.Cryptography для С#.

В результате выполнения курсовой работы было разработано приложение с простым интерфейсом, позволяющее шифровать и дешифровать текст. К достоинствам разработанного ПО можно отнести простой и понятный, минималистический интерфейс программы и удобную работу с данными с возможность их записи в файл. К недостаткам же возможно отнести единственный верный логин и пароль, отсутствие базы данных.

В дальнейшем приложение может быть дополнено базой данных, благодаря которой приложение станет ещё удобнее и избавится от ряда недостатков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Введение в криптографию. Курс лекций / В.А. Романьков. М. : ФОРУМ, 2012. - 240 с. - (Высшее образование). ISBN 978-5-91134-573-0(дата обращения 22.10.22).
2. Жиль, Земор Курс криптографии / Земор Жиль ; перевод В. В. Шуликовская. — Москва, Ижевск : Регулярная и хаотическая динамика, Институт компьютерных исследований, 2019. — 256 с. — ISBN 978-5-4344-0770-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/91941.html> (дата обращения: 17.10.22). — Режим доступа: для авторизир. пользователей.
3. Басалова, Г. В. Основы криптографии : учебное пособие / Г. В. Басалова. — 3-е изд. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 282 с. — ISBN 978-5-4497-0340-8. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/89455.html> (дата обращения: 5.11.22). — Режим доступа: для авторизир. пользователей.
4. Фороузан, Б. А. Криптография и безопасность сетей : учебное пособие / Б. А. Фороузан ; под редакцией А. Н. Берлина. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 776 с. — ISBN 978-5-4497-0946-2. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102017.html> (дата обращения: 15.11.22). — Режим доступа: для авторизир. пользователей.
5. Керов Леонид Александрович Дополнительные приемы программирования на языке с# [Электронный ресурс] // КИО. 2009. №6. URL: <https://cyberleninka.ru/article/n/dopolnitelnye-priemy-programmirovaniya-na-yazyke-s> (дата обращения: 21.10.2022).
6. Алгоритмы блочной криптографии учебно-методическое пособие / Н. Р. Спиричева. - Екатеринбург : Изд-во Урал, ун-та, 2 013.-78, [2] с. ISBN 978-5-

7996-0934-4 Текст : электронный [сайт]. — URL: <https://elar.urfu.ru/bitstream/10995/28062/1/978-5-7996-0934-4.pdf> (дата обращения: 16.12.2022). — Режим доступа: общий доступ

7. Биллиг, В. А. Основы программирования на C# : учебное пособие / В. А. Биллиг. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 573 с. — ISBN 978-5-4497-0893-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102033.html> (дата обращения: 21.10.2022). — Режим доступа: для авторизир. пользователей.

8. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) : учебник / В. А. Биллиг. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 409 с. — ISBN 978-5-4497-0880-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102029.html> (дата обращения: 29.10.2022). — Режим доступа: для авторизир. пользователей.

9. Абрамян, М. Э. User interface development based on Windows Forms class library : textbook / М. Э. Абрамян. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2021. — 278 с. — ISBN 978-5-9275-3830-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/117147.html> (дата обращения: 04.10.2022). — Режим доступа: для авторизир. пользователей.

10. Кариев, Ч. А. Разработка Windows-приложений на основе Visual C# : учебное пособие / Ч. А. Кариев. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 978 с. — ISBN 978-5-4497-0909-7. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102057.html> (дата обращения: 04.10.2022). — Режим доступа: для авторизир. пользователей.

ПРИЛОЖЕНИЕ

Текст программы

Program – главный модуль программы, начало программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Курсач
{
    class Users
    {

    }

    internal static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LogInForm());
        }
    }
}
```

LogIn Form - модуль, предназначенный для входа в Crypto Protocol

Form:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```

using System.Threading.Tasks;
using System.Windows.Forms;

namespace Кrypta
{
    public partial class LogInForm : Form
    {
        public LogInForm()
        {
            InitializeComponent();
            User_pass.UseSystemPasswordChar = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
        }

        private void Login_Click(object sender, EventArgs e)
        {
            if (User_name.Text == "admin" && User_pass.Text == "1111")
            {
                CryptoProtocolForm newForm = new CryptoProtocolForm();
                newForm.Show();
                this.Hide();
            }
            else
            {
                User_name.Text = "";
                User_pass.Text = "";

                MessageBox.Show("Неверный логин или пароль!!!");
            }
        }
    }
}

```



```

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkPass.Checked)
    {
        User_pass.UseSystemPasswordChar = false;
    }else User_pass.UseSystemPasswordChar = true;
}

private void Form1_Load(object sender, EventArgs e)
{
}
}
}

```

Crypto Protocol Form – модуль предназначенный для вызова криптографических функций программы:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Security.Cryptography;
using System.Net.NetworkInformation;

namespace Купсау
{
    public partial class CryptoProtocolForm : Form
    {
        public CryptoProtocolForm()
        {
            InitializeComponent();
            Cryptoprotocol.Items.AddRange(new string[] { "AES" });
            Cryptoprotocol.SelectedItem = "AES";
            Cryptoprotocol.SelectedIndexChanged += comboBox1_SelectedIndexChanged;
            Size = new Size(837, 699);
        }

        private void Form2_Load(object sender, EventArgs e)
        {
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (Cryptoprotocol.SelectedItem.ToString() == "AES")
            {

```

```

        Key1.Show();
        Key1.Text = "AES Key";

        Key2.Show();
        Key2.Text = "AES IV";

        CryptoKey1.Show();
        CryptoKey2.Show();
    }

}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void Encrypt_Click(object sender, EventArgs e)
{
    encrypt_text.Clear();
    if (decrypt_text.Text != null && Cryptoprotocol.SelectedItem.ToString() ==
"AES")
    {
        AES aes = new AES();
        using (Aes Aes = Aes.Create())
        {

            string[] enckey = aes.encryptKeyToString(Aes.Key);
            CryptoKey1.Text = null;
            for (int i = 0; i < enckey.Length; i++)
            {

                CryptoKey1.Text += enckey[i];

                CryptoKey1.Text += " ";
            }

            string[] enciv = aes.encryptIVToString(Aes.IV);
            CryptoKey2.Text = null;
            for (int i = 0; i < enciv.Length; i++)
            {

                CryptoKey2.Text += enciv[i];

                CryptoKey2.Text += " ";
            }
            if (CryptoKey1.Text == "" || CryptoKey2.Text == "")
            {
                string[] encntext =
aes.encrypttextToString(aes.Encrypt_Aes(decrypt_text.Text, Aes.Key, Aes.IV));
                for (int i = 0; i < encntext.Length; i++)
                {

                    encrypt_text.Text += encntext[i];

                    encrypt_text.Text += " ";
                }
            }
            else if (CryptoKey1.Text != "" && CryptoKey2.Text != "" &&
decrypt_text.Text != null)
            {
                if (aes.StringToByte(CryptoKey1.Text, 32, true).Length == 32 &&
aes.StringToByte(CryptoKey2.Text, 16, true).Length == 16)
                {
                    string[] encntext =
aes.encrypttextToString(aes.Encrypt_Aes(decrypt_text.Text,

```

```

aes.StringToByte(CryptoKey1.Text, 32, true), aes.StringToByte(CryptoKey2.Text, 16,
true)));

        if (enctext != null && enctext.Length > 0)
        {
            for (int i = 0; i < enctext.Length; i++)
            {
                encrypt_text.Text += enctext[i];

                encrypt_text.Text += " ";
            }
        }
    }
}

else
{
    MessageBox.Show("Выберите метод шифровки/дешифровки!!!");
}

private void encrypt_text_TextChanged(object sender, EventArgs e)
{
}

private void CryptoKey_TextChanged(object sender, EventArgs e)
{
}

private void Exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void Decrypt_Click(object sender, EventArgs e)
{
    decrypt_text.Text = null;
    try
    {
        if (encrypt_text.Text != null && Cryptoprotocol.SelectedItem.ToString()
== "AES")
        {
            AES aes = new AES();
            using (Aes Aes = Aes.Create())
            {
                if ((CryptoKey1.Text != "" && CryptoKey2.Text != "") &&
(aes.StringToByte(CryptoKey1.Text, 0, true) != null &&
aes.StringToByte(CryptoKey2.Text, 0, true) != null) &&
(aes.StringToByte(CryptoKey1.Text, 0, true).Length == 32 &&
aes.StringToByte(CryptoKey2.Text, 0, true).Length == 16))
                {

                    string dectext =
aes.Decrypt_Aes(aes.StringToByte(encrypt_text.Text, 0, true),
aes.StringToByte(CryptoKey1.Text, 32, true), aes.StringToByte(CryptoKey2.Text, 16,
true));

                    decrypt_text.Text += dectext;

                    decrypt_text.Text += "\n";
                }
            }
        }
    }
}

```

```

        else
        {
            MessageBox.Show("Введён неверный ключ шифровки");
            CryptoKey1.Text = null;
            CryptoKey2.Text = null;
        }
    }
} catch
{
    MessageBox.Show("Ошибка шифровки / дешифровки.");
}
}

private void Encrypt_to_file_Click(object sender, EventArgs e)
{
    FileStream file = new FileStream("Encrypt text.txt", FileMode.Create);
    StreamWriter stream = new StreamWriter(file);
    stream.WriteLine("Aes key:");
    stream.WriteLine(CryptoKey1.Text);
    stream.WriteLine("AES IV:");
    stream.WriteLine(CryptoKey2.Text);
    stream.WriteLine("Encrypt text:");
    stream.Write(encrypt_text.Text);
    stream.Close();
    file.Close();
    MessageBox.Show("Файл успешно записан.");
}

private void label5_Click(object sender, EventArgs e)
{
}

private void button1_Click(object sender, EventArgs e)
{
    FileStream file = new FileStream("Decrypt text.txt", FileMode.Create);
    StreamWriter stream = new StreamWriter(file);
    stream.WriteLine("Aes key:");
    stream.WriteLine(CryptoKey1.Text);
    stream.WriteLine("AES IV:");
    stream.WriteLine(CryptoKey2.Text);
    stream.WriteLine("Decrypt text:");
    stream.Write(decrypt_text.Text);
    stream.Close();
    file.Close();
    MessageBox.Show("Файл успешно записан.");
}

private void encrypt_text_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;

    if (!Char.IsDigit(number) && e.KeyChar != 32)
    {
        e.Handled = true;
    }
}

private void CryptoKey1_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;

```

```

        if (!Char.IsDigit(number) && e.KeyChar != 32)
        {
            e.Handled = true;
        }
    }

    private void CryptoKey2_KeyPress(object sender, KeyPressEventArgs e)
    {
        char number = e.KeyChar;

        if (!Char.IsDigit(number) && e.KeyChar != 32)
        {
            e.Handled = true;
        }
    }
}
}}

```

AES - модуль отвечающий за шифровку\дешифровку текста:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Security.Cryptography;
using System.Windows.Forms;
using System.Linq.Expressions;

namespace Купсач
{
    class AES
    {
        private byte[] encrypted;
        private string plaintext;

        public string[] encryptKeyToString(byte[] Key)
        {
            string[] encryptkey = new string[Key.Length];

            for (int i = 0; i < Key.Length; i++)
            {
                encryptkey[i] = Key[i].ToString();
            }

            return encryptkey;
        }

        public string[] encryptIVToString(byte[] IV)
        {
            string[] encryptiv = new string[IV.Length];

            for (int i = 0; i < IV.Length; i++)
            {
                encryptiv[i] = IV[i].ToString();
            }

            return encryptiv;
        }

        public string[] encrypttextToString(byte[] encrypt)
        {
            if (encrypt != null)
            {

```

```

        string[] encText = new string[encrypt.Length];
        for (int i = 0; i < encrypt.Length; i++)
        {
            encText[i] = encrypt[i].ToString();
        }
        return encText;
    }
    else
    {
        MessageBox.Show("Ошибка шифровки / дешифровки.");
        return null;
    }
}

public byte[] StringToByte (string s, int keylength, bool enc)
{
    if (keylength > 0)
    {
        string[] encText = new string[keylength];
        byte[] result = new byte[keylength];
        for (int i = 0, j = 0; i < s.Length - 1 && j < keylength; i++)
        {
            encText[j] += s[i];
            if (enc == true && encText[j].Length > 3)
            {
                MessageBox.Show("Введён неверный ключ шифровки");
                return null;
            }
            if (i <= s.Length - 3)
            {
                if ((s[i + 1] == ' ' || s[i + 1] == '\n') && (s[i + 2] != ' ' || s[i + 2] != '\n'))
                {
                    j++;
                    i++;
                }
                else if ((s[i + 1] == ' ' || s[i + 1] == '\n') && (s[i + 2] == ' ' || s[i + 2] == '\n')) continue;
            }
        }
        for (int i = 0; i < encText.Length; i++)
        {
            result[i] = Convert.ToByte(encText[i]);
        }
        return result;
    }
    else
    {
        int n = 0;
        for (int i = 0; i < s.Length; i++)
        {
            if (s[i] == ' ')
                n++;
        }
        string[] encText = new string[n];
        byte[] result = new byte[n];
        for (int i = 0, j = 0; i < s.Length - 1 && j < s.Length - 1; i++)
        {
            encText[j] += s[i];
            if (enc == true && encText[j].Length > 3)
            {
                MessageBox.Show("Введён неверный ключ шифровки");
                return null;
            }
        }
    }
}

```

```

        }
        if (i <= s.Length - 3)
        {
            if ((s[i + 1] == ' ' || s[i + 1] == '\n') && (s[i + 2] != ' ' || s[i + 2] != '\n') && (i <= s.Length - 3))
            {
                j++;
                i++;
            }
            else if ((s[i + 1] == ' ' || s[i + 1] == '\n') && (s[i + 2] == ' ' || s[i + 2] == '\n') && (i <= s.Length - 3)) continue;
        }
    }
    for (int i = 0; i < encrptext.Length; i++)
    {
        result[i] = Convert.ToByte(encrptext[i]);
    }
    return result;
}

}

public byte[] Encrypt_Aes(string soursetext, byte[] Key, byte[] IV)
{
    try
    {
        // Check arguments.
        if (soursetext == null || soursetext.Length <= 0)
        {
            MessageBox.Show("Не введён текст для шифровки");
            return null;
        }
        if (Key == null || Key.Length <= 0)
        {
            MessageBox.Show("Введён неверный ключ шифровки");
            return null;
        }
        if (IV == null || IV.Length <= 0)
        {
            MessageBox.Show("Введён неверный ключ шифровки");
            return null;
        }

        // Create an Aes object
        // with the specified key and IV.
        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
            aesAlg.IV = IV;

            // Create an encryptor to perform the stream transform.
            ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key,
aesAlg.IV);

            // Create the streams used for encryption.
            using (MemoryStream msEncrypt = new MemoryStream())
            {
                using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
encryptor, CryptoStreamMode.Write))
                {
                    using (StreamWriter swEncrypt = new
StreamWriter(csEncrypt))
                    {

```

```

        //Write all data to the stream.
        swEncrypt.Write(soursetext);
    }
    encrypted = msEncrypt.ToArray();
}
}

// Return the encrypted bytes from the memory stream.
return encrypted;
}
catch
{
    MessageBox.Show("Ошибка шифровки / дешифровки.");
    return null;
}
}

public string Decrypt_Aes(byte[] cipherText, byte[] Key, byte[] IV)
{
    try
    {
        // Проверка аргументов.
        if (cipherText == null || cipherText.Length <= 0)
        {
            MessageBox.Show("Не введён текст для дешифровки");
            return null;
        }
        if (Key == null || Key.Length <= 0)
        {
            MessageBox.Show("Введён неверный ключ шифровки");
            return null;
        }
        if (IV == null || IV.Length <= 0)
        {
            MessageBox.Show("Введён неверный ключ шифровки");
            return null;
        }

        plaintext = null;

        // Создание объекта класса AES библиотеки System.Security.Cryptography
        using (Aes aesAlg = Aes.Create())
        {
            // Указание в качестве ключей дешифровки, пользовательских ключей
            aesAlg.Key = Key;
            aesAlg.IV = IV;

            // Создание дешифратора для выполнения преобразования потока.
            ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key,
aesAlg.IV);

            // Создание потока для расшифровки
            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new
StreamReader(csDecrypt))
                    {
                        // Чтение расшифрованных байтов из потока и помещение их
в строку для вывода
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }
    }
}

```



```

        }
    }
}

return plaintext;
}
catch
{
    MessageBox.Show("Ошибка шифровки / дешифровки.");
    return null;
}
}
}

```