

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Информационная безопасность
автоматизированных систем

Специализация -

Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ КРИПТОАНАЛИЗА ПРОСТЫХ
ШИФРОВ

Студент _____ Каравайкин Александр Витальевич

Руководитель _____ Липилин О. В.

Нормоконтролёр _____ Фатеев А.Г.

Рецензент начальник НИС АО «ПНИЭИ» _____ Акмаев А.Ж.

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ С.Л. Зефирова

Работа защищена с отметкой _____ (протокол заседания ГЭК
от _____ № _____)

Секретарь ГЭК _____ А.П. Иванов

Пенза 2017

РЕФЕРАТ

Отчет содержит 112 с., 45 рис., 22 таблицы, 18 источников.

ПРОСТОЙ ШИФР, КРИПТОАНАЛИЗ, ШИФРОВАНИЕ,
ШИФРОТЕКСТ, ПРОГРАММНОЕ СРЕДСТВО,

Объектом исследований является криптоанализ простых шифров.

Цель дипломного проекта – разработка программного средства для криптоанализа простых шифров.

В процессе работы был проведен обзор существующих методов криптоанализа простых шифров и проведен анализ возможных способов реализации программного средства с графическим интерфейсом, а также разработана методика эксплуатации программного средства.

В результате было разработано программное средство для криптоанализа простых шифров, которое будет положено в основу методических указаний по дисциплине «Криптографические методы защиты информации» для студентов третьего курса специальности 10.05.03 «Информационная безопасность автоматизированных систем».

СОДЕРЖАНИЕ

Обозначения и сокращения.....	5
Введение.....	6
1 Описание простых шифров.....	7
1.1 Шифр простой замены	7
1.2 Сдвиговый шифр замены	8
1.3 Шифр гаммирования	9
1.4 Шифры одиночной перестановки	13
1.5 Шифр двойной перестановки	14
1.6 Шифр маршрутной перестановки	15
1.7 Шифр полиалфавитной замены.....	17
1.8 Алгоритм RSA.....	18
2 Методы криптоанализа простых шифров	21
2.1 Метод полного перебора.....	21
2.2 Метод статистического криптоанализа – подсчет частоты встречи символов в шифротексте.....	22
2.3 Метод статистического криптоанализа – подсчет частоты встречи биграмм в шифротексте	24
2.4 Метод разложения числа на простые множители	31
3 Параметры используемых шифров	32
3.1 Параметры шифра простой замены	32
3.2 Параметры шифра сдвиговой замены.....	34
3.3 Параметры шифра гаммирования	35
3.4 Параметры шифра одиночной перестановки.....	35
3.5 Параметры шифра двойной перестановки	36
3.6 Параметры шифра RSA.....	36
3.7 Параметры шифра маршрутной перестановки.....	37

3.8	Параметры шифра полиалфавитной замены	38
4	Архитектура программного средства	39
4.1	Средства реализации программного средства	39
4.2	Описание классов программного средства	40
5	Руководство пользователя программного средства	46
5.1	Краткое описание возможностей	46
5.2	Уровень подготовки пользователя	46
5.3	Выполняемые функции и задачи	47
5.4	Описание операций технологического процесса обработки исходных данных, необходимых для выполнения учебных задач криптоанализа	48
5.4.1	Шифрование	48
5.4.2	Расшифрование	67
5.4.3	Криптоанализ	68
	Заключение	72
	Список использованных источников	73
	Приложение А Исходный код программного средства	75

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ОТ – открытый текст;

ШТ – шифротекст;

ПС – программное средство;

ГИ – графический интерфейс;

ОС – операционная система.

ВВЕДЕНИЕ

В сфере информационной безопасности ключевую роль играет такая наука, как криптография. Для обеспечения защиты информации необходимо правильно подбирать методы криптографической защиты.

Студенты, обучающиеся по специальности 10.03.05 «Информационная безопасность автоматизированных систем» должны знать методы анализа стойкости шифров, для получения умений по выбору методов криптографической защиты информации, поэтому актуальной является проблема разработки программного средства для автоматизации криптоанализа простых шифров. Таким образом, тема выпускной квалификационной работы является актуальной.

1 Описание простых шифров

1.1 Шифр простой замены

В криптографии шифром принято называть совокупность обратимых преобразований множества открытых данных на множество зашифрованных данных, задаваемых ключом и алгоритмом преобразования.

Криптографические преобразования имеют цель обеспечить недоступность информации для лиц, не имеющих ключа, и поддержание с требуемой надежностью обнаружения несанкционированных искажений.

В криптографии используются следующие базовые алгоритмы шифрования:

- алгоритм замены (подстановки) – символы шифруемого текста заменяются символами того же или другого алфавита в соответствии с заранее обусловленной схемой замены;
- алгоритм перестановки – символы шифруемого текста переставляются по определенному правилу в пределах некоторого блока этого текста;
- гаммирование – символы шифруемого текста складываются с символами некоторой случайной последовательности;
- аналитическое преобразование – преобразование шифруемого текста по некоторому аналитическому правилу (формуле). [1]

Простые шифры включают в себя один из вышеперечисленных базовых алгоритмов шифрования.

Простейшая подстановка – прямая замена символов шифруемого сообщения другими буквами того же самого или другого алфавита.

Стойкость метода простой замены низкая. Наиболее эффективный подход к криптоанализу шифра простой замены, помимо метода грубой силы, является частотный анализ частоты встречи символов в шифротексте. Текст, зашифрованный методом простой замены, одинаковые статистические характеристики, что и исходный текст, так зная стандартные частоты

появления символов в том языке, на котором написано сообщение, и осуществляя подбор по частотам появления символы в зашифрованном сообщении, можно достаточно легко восстановить таблицу замены. Для этого необходим лишь достаточно длинный шифротекст, для того, чтобы получить достоверные оценки частот появления символов. Поэтому шифр простой замены используют лишь тогда, когда шифруемое сообщение не превышает 50 символов.

При использовании полиалфавитной замены повышается стойкость шифра. Пример таблицы замены представлен в таблице 1.[2]

Таблица 1 – Таблица замены

Исходный алфавит	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ш	ъ	ы	ь	э	ю	я
Алфавит замены	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ш	ъ	ы	ь	э	ю	я	а	б	в

1.2 Сдвиговой шифр замены

Шифр простой замены, простой подстановочный шифр, моноалфавитный шифр – класс методов шифрования, который сводится к созданию по определённому алгоритму таблицы шифрования, в которой каждой букве открытого текста сопоставляется единственная буква шифротекста. Само шифрование заключается в замене букв согласно таблице. Для расшифровки достаточно иметь ту же таблицу, или же знать алгоритм, по которому она генерируется. Шифр простой замены не всегда представляет собой замену буквы на какую-нибудь другую букву. Также допускается использовать замену буквы на цифру. [3]

К шифрам простой замены относятся способы шифрования, возникшие в древности или средневековье, такие как Шифр Цезаря.

Шифр Цезаря – это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, который находится на некотором постоянном числе позиций левее или правее него в алфавите.

Например, в шифре со сдвигом 3 А была бы заменена на Г, Б станет Д, и так далее.

Шаг шифрования, который выполняется шифром Цезаря, часто включается как часть более сложных схем, таких как шифр Виженера, и все ещё имеет современное приложение в системе ROT13. Как и все моноалфавитные шифры, шифр Цезаря легко вскрывается и не имеет практически никакого применения на практике.

Шифрование с использованием ключа k равно 3. Буква «Е» «сдвигается» на три буквы вперёд и становится буквой «З». Твёрдый знак, перемещённый на три буквы вперёд, становится буквой «Э», и так далее. Пример сдвига с ключом k равно 3 представлен в таблице 2.

Таблица 2 – Пример сдвига алфавита с использованием ключа k равно 3

Исходный алфавит	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ъ	ы	ь	э	ю	я
Шифрованный алфавит	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ъ	ы	ь	э	ю	я	а	б	в

Открытый текст: «Съешь же ещё этих мягких французских булок, да выпей чаю.» после выполнения шифрования шифром Цезаря с ключом k равно 3 будет выглядеть следующим образом: «Фэзыя йз зьи ахлш пвёнлш чугрщцкфнлш дцосн, жг еютзм тгб.» [4]

1.3 Шифр гаммирования

Гаммированием (gamma xoring) называется процесс «наложения» гамма-последовательности на открытые данные. Обычно это суммирование по какому-либо модулю, например, по модулю два, такое суммирование принимает вид обычного «исключающего ИЛИ» суммирования.[5]

В алфавите любого естественного языка буквы следуют друг за другом в определенном порядке. Эта особенность позволяет присвоить каждой букве алфавита ее естественный порядковый номер. Например, в английском алфавите букве А присваивается порядковый номер 1, букве Q -

порядковый номер 17, а букве Z - порядковый номер 26. Похожую операцию можно применить и для русского алфавита, например для алфавита RUS30 (где Ё равно Е, Й равно И, Ъ равно Ь). Буква А будет иметь порядковый номер 1, О - номер 14, Я - 30. Если в открытом сообщении произвести замену каждой буквы ее порядковым номером в предложенном алфавите, то преобразование числового сообщения в буквенное позволит однозначно восстановить исходное открытое сообщение. К примеру, числовое сообщение: «1 11 20 1 3 9 18» в алфавите RUS30 преобразуется в буквенное сообщение: АЛФАВИТ. Алфавит RUS30 представлен в таблице 3.

Таблица 3 – Алфавит RUS30

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Ъ	Э	Ю	Я
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30			

Далее задаются преобразования зашифрования f и преобразования расшифрования g для произвольного шифра гаммирования. Пусть:

- требуется зашифровать сообщение, представленное выражением (1) в алфавите, который представлен выражением (2);

$$X = x_1, \dots, x_t \quad (1)$$

$$\Omega = \{a_1, \dots, a_n\} \quad (2)$$

- n - мощность алфавита;
- каждая буква ставится в соответствие со своим порядковым номером в алфавите;
- необходимо выбрать некоторую последовательность, представленную в выражении (3), составленную из букв.

$$\Omega = \{\gamma_1, \dots, \gamma_T\} \quad (3)$$

Данная последовательность называется гаммой шифра, или ключевой последовательностью.

Тогда преобразование зашифрования f_{ki} будет являться преобразованием, представленное выражением (4).

$$y_i = f_{ki}(x_i) = r_n(x_i + \gamma_i) \quad (4)$$

Где k_i равно γ_i - используемый знак гаммы последовательности для шифрования i -той буквы сообщения x_i ; $r_n(b)$ - остаток от деления числа b на n (полагаем, что $r_n(n)$ равно n).

Итак, процесс зашифрование шифром гаммирования означает «сложение» или «наложение» некоторой последовательности (гаммы) на знаки (буквы) открытого текста. Очевидно, что в таком случае для расшифрования нужно вычесть из букв шифртекста знаки гаммы, Соответственно, весь отрезок гаммы (то есть вся последовательность) является ключом данного шифра, именно поэтому ее называют ключевой последовательностью.

Аналогичные формулы для шифрования и расшифрования есть в сдвиговых шифрах. Дело в том, что сдвиговой шифр является частным случаем шифра гаммирования, когда вся гамма представляется одним и тем же значением k (то есть, другими словами, когда все элементы гаммы равны k). На рисунке 1 изображен процесс зашифрования сообщения шифром гаммирования.

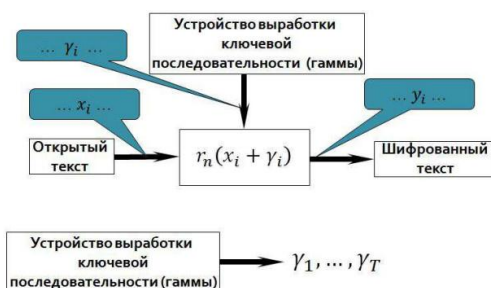


Рисунок 1 – Зашифрование сообщения шифром гаммирования

В каждый момент времени в устройство шифрования (шифратор) подается определенная буква открытого текста x_i и подается знак гаммы γ_i , сгенерированный по некоторому правилу устройством, которое вырабатывает ключевую последовательность согласно формуле (5).

$$y_i = f_{ki}(x_i) = r_n(x_i + \gamma_i) \quad (5)$$

Затем шифратор вырабатывает очередную букву шифрованного текста. Данный процесс длится до тех пор, пока в через шифратор не «пройдут» все буквы открытого текста.

Необходимо зашифровать слово АЛФАВИТ на следующей гамме: ИДФНТВХ. Результат представлен в выражениях (6), (7), (8), (9), (10), (11), (12).

$$y_1 = r_{30}(A + И) = r_{30}(1 + 9) = 10 = K, \quad (6)$$

$$y_1 = r_{30}(Л + Д) = r_{30}(11 + 5) = 16 = P, \quad (7)$$

$$y_1 = r_{30}(\Phi + \Phi) = r_{30}(20 + 20) = 10 = K, \quad (8)$$

$$y_1 = r_{30}(А + Н) = r_{30}(1 + 13) = 14 = O, \quad (9)$$

$$y_1 = r_{30}(В + Т) = r_{30}(3 + 18) = 21 = X, \quad (10)$$

$$y_1 = r_{30}(И + В) = r_{30}(9 + 3) = 12 = M, \quad (11)$$

$$y_1 = r_{30}(Т + Х) = r_{30}(18 + 21) = 9 = И. \quad (12)$$

В результате получается шифротекст: КРКОХМИ. [6]

1.4 Шифр одиночной перестановки

Каждое преобразование шифра перестановки, предназначенное для зашифрования сообщения длиной n символов, можно задать с помощью таблицы, которая представлена выражением (13).

$$\begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & n \end{pmatrix} \quad (13)$$

Где i_1 – номер места шифртекста, на которое перемещается первая буква исходного сообщения при выбранном преобразовании, i_2 – номер места для второй буквы и т. д. В верхней строке таблицы выписаны по порядку числа от 1 до n , а в нижней – те же числа, но в произвольном порядке. Такая таблица называется подстановкой степени n . Зная подстановку, которая задает преобразование, можно осуществить как зашифрование, так и расшифрование текста.

Из этого следует, что подстановка является ключом шифра перестановки, при этом легко заметить, что число всех возможных шифров перестановок для текста заданной длины равно числу ключей, или таблиц указанного вида.

Например, если для преобразования используется подстановка, представленная выражением (14).

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 3 & 1 & 4 & 6 \end{pmatrix} \quad (14)$$

В соответствии с данной подстановкой зашифровывается слово МОСКВА, то получится шифротекст КОСВМА. Пример зашифрования представлен в таблице 4.

Таблица 4 – Зашифрование методом перестановки

М	О	С	К	В	А
1	2	3	4	5	6
5	2	3	1	4	6

Для расширения шифра перестановки используется также таблица подстановок. На том же ключе, что и в предыдущем примере, зашифровано некоторое слово и получен шифртекст: НЧЕИУК.

Для расшифрования необходимо составить подстановку (таблицу) «обратную» к данному ключу, для этого необходимо поменять местами первую и вторую строки исходной таблицы и затем упорядочим столбцы по возрастанию номеров в первой строке.

Полученная таким образом таблица (подстановка) является ключом расшифрования. Если исходная таблица показывает, что k – ую букву необходимо поставить на место i_k , то полученная таблица показывает обратное, то есть i_k – ую букву нужно подставить на место с номером k . Пример расшифрования представлен в таблице 5. [6]

Таблица 5 – Расшифрование для метода перестановки

1	2	3	4	5	6
Н	Ч	Е	И	У	К
У	Ч	Е	Н	И	К

1.5 Шифр двойной перестановки

В таблицу по определенному маршруту записывается текст сообщения, затем выполняется перестановка столбцов, а потом переставляются строки.

Пример шифрования сообщения «АБРАМОВ+ДЯДИНА» показан на рисунке 2



Рисунок 2 – Шифрование методом двойной перестановки

В результате получается шифротекст – «ОАБЯ+_АИВ_РДМНАД».

Взломать шифр двойной перестановки сложнее, чем взломать шифр одиночной перестановки. Это объясняется тем, что анаграммы не появятся при попытке применить различные размеры матрицы к шифротексту.

Криптоаналитик должен опробовать множество различных комбинаций ключевых слов, чтобы найти похожий в зашифрованном тексте. Шифр будет достаточно просто взломать, если будут перехвачены несколько сообщений одной длины и зашифрованные одними и теми же ключами.

Можно предположить, что наличие нескольких сообщений одинаковой длины, которые зашифрованы одинаковым ключами, позволят криптоаналитику найти и секретные ключи и открытый текст.

Эта техника использовалась французами для перехвата немецких сообщений в начале Первой мировой войны, до тех пор, пока немецкие криптографы не улучшили свою систему шифрования.

Шифр двойной перестановки остается одним из самых сильных шифров, которые могут использоваться для шифрования сообщений без применения электронного оборудования.

1.6 Шифр маршрутной перестановки

Изначально решетка Кардано представляла собой трафарет с прорезанными в нем отверстиями. В этих отверстиях на листе бумаги, который клали под решетку, записывались буквы, слоги и слова сообщения. Затем трафарет убирали, и свободные места заполнялись более или менее осмысленным текстом для сокрытия секретного сообщения. Данный метод сокрытия информации относится к такой науке как стеганография.

Позднее был разработан шифр «поворотная решетка» или, как его еще называют, «решетка для выющихся растений», поскольку решетка напоминала отверстия в деревянных решетках садовых строений.

Помимо того, что между изначальным предложением Кардано и шифром «поворотная решетка» существенная разница, методы сокрытия

информации, которые основаны на использовании трафаретов, принято называть «решетками Кардано».

Для шифрования и дешифрования с помощью данного шифра изготавливается прямоугольный трафарет с четным количеством строк и столбцов. В трафарете вырезаются клетки таким образом, чтобы при наложении его на таблицу того же размера четырьмя возможными способами, его вырезы полностью покрывали все ячейки таблицы ровно по одному разу.

При шифровании трафарет накладывается на таблицу. В видимые ячейки таблицы выписываются буквы исходного текста слева-направо сверху-вниз. Затем происходит поворот трафарета и вписывается следующая часть букв. Эта операция повторяется еще два раза. Шифротекст выписывают из итоговой таблицы по определенному маршруту.

Таким образом, ключом при шифровании является трафарет, порядок его поворотов и маршрут выписывания.

Пример шифрования сообщения «АБРАМОВ+ДЯДИНА» представлен на рисунке 3. [7]

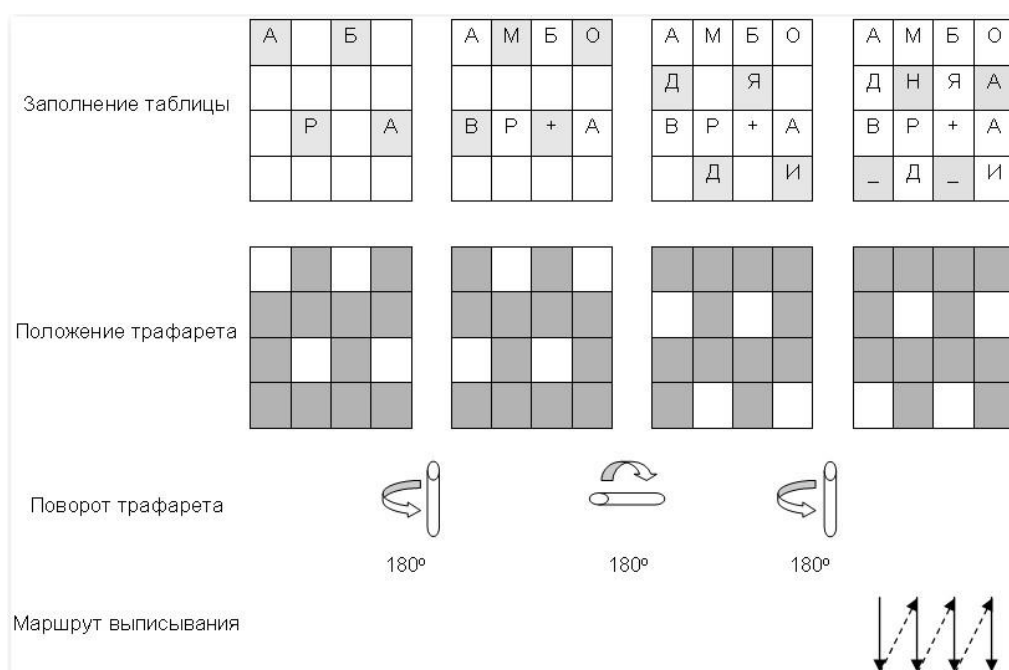


Рисунок 3 – Шифрование с помощью шифра Кардано

1.7 Шифр полиалфавитной замены

Шифры сложной замены называют полиалфавитными, так как для шифрования каждого символа исходного сообщения применяют свой шифр простой замены. Полиалфавитная подстановка последовательно и циклически меняет используемые алфавиты.

При r -алфавитной подстановке символ x_0 исходного сообщения заменяется символом y_0 из алфавита B_0 , символ x_1 - символом y_1 из алфавита B_1 , и так далее, символ x_{r-1} заменяется символом y_{r-1} из алфавита B_{r-1} , символ x_r заменяется символом y_r снова из алфавита B_0 , и т.д. Общая схема полиалфавитной подстановки для случая r равно 4 показана в таблице 6.

Таблица 6 – Схема r -алфавитной подстановки для случая r равно 4

Входной символ	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Алфавит подстановки	B_0	B_1	B_2	B_3	B_0	B_1	B_2	B_3	B_0	B_1

Эффект использования полиалфавитной подстановки заключается в том, что обеспечивается маскировка естественной статистики исходного языка, так как конкретный символ из исходного алфавита A может быть преобразован в несколько разных символов шифровальных алфавитов B_j . Степень обеспечиваемой защиты теоретически пропорциональна длине периода r в последовательности используемых алфавитов B_j .

Шифр Гронсфельда является модификацией шифра Цезаря с помощью добавления числового ключа. Для этого под буквами исходного сообщения выписывают цифры числового ключа. В том случае, если ключ оказывается короче сообщения, то его запись циклически повторяют. Шифротекст получают примерно таким же способом, как в шифре Цезаря, но отсчитывают по алфавиту не третью букву, как это выполняется в шифре Цезаря, а выбирают ту букву, которая смещена по алфавиту на соответствующую цифру ключа. Например, применяя в качестве ключа группу из четырех начальных цифр числа e (основания натуральных

логарифмов), а именно 2,718, для исходного сообщения получается ВОСТОЧНЫЙ ЭКСПРЕСС следующий шифртекст: ДХТЬРЮО ГЛ ДЛЩСЧЖЩУ. Пример шифрования с помощью шифра Гронсфельда представлен в таблице 7.

Таблица 7 – Зашифрование шифром Гронсфельда

Сообщение	В	О	С	Т	О	Ч	Н	Ы	Й		Э	К	С	П	Р	Е	С	С
Ключ	2	7	1	8	2	7	1	8	2		7	1	8	2	7	1	8	2
Шифротекст	Д	Х	Т	Ь	Р	Ю	О	Г	Л		Д	Л	Щ	С	Ч	Ж	Щ	У

Чтобы осуществить зашифрование первой буквы сообщения *В*, используя первую цифру ключа 2, нужно отсчитать вторую по порядку букву от *В* в алфавите. Таким образом получается первая буква шифртекста *Д*.

Правильно реализованный шифр полиалфавитной замены довольно сложно взломать. Его стойкость основана на множестве возможных комбинаций смены букв алфавита. Некоторые эффективные методы криптоанализа на такие шифры были обнаружены еще в девятнадцатом веке. Они позволяют узнать длину секретного ключа. После этого остается лишь изучить зашифрованный текст, используя методы частотного анализа. [8]

1.8 Алгоритм RSA

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, который основывается на вычислительной сложности задачи факторизации больших целых чисел. [9]

В основе алгоритма RSA лежит задача факторизации произведения двух простых больших чисел. Для шифрования используется простая операция возведения в степень по модулю N . Для расшифрования же необходимо вычислить функцию Эйлера от числа N , для этого необходимо знать разложение числа n на простые множители (в этом и состоит задача факторизации).

В RSA открытый и закрытый ключ состоит из пары целых чисел. Закрытый ключ хранится в секрете, а открытый ключ сообщается другому участнику, либо где-то публикуется.

Всё начинается с генерации ключевой пары (открытый, закрытый ключ). Генерация ключей в RSA осуществляется следующим образом:

- выбираются два простых числа p и q (такие что p не равно q);
- вычисляется модуль, согласно выражению (15);

$$N = p \cdot q \quad (15)$$

- вычисляется значение функции Эйлера от модуля N ;
- выбирается число e , взаимно простое со значением функции $\phi(N)$, называемое открытой экспонентой, число e должно лежать в интервале, представленном в выражении (16);

$$1 < e < \phi(N) \quad (16)$$

- вычисляется число d , называемое секретной экспонентой, в соответствии с выражением (17).

$$d \cdot e = 1(mod\phi(N)) \quad (17)$$

То есть является мультипликативно обратное к числу e по модулю $\phi(N)$.

В результате выполнения действий описанных выше получаются 2 пары ключей: (e, N) – открытый ключ и (d, N) – закрытый ключ.

К примеру, есть следующий сценарий: Боб и Алиса ведут переписку в интернете, но хотят чтобы их переписка осталась в тайне. Алиса заранее сгенерировала пару: закрытый и открытый ключ, а затем отправила

открытый ключ Бобу. Боб, в свою очередь, хочет послать зашифрованное сообщение Алисе.

Боб шифрует сообщение m , используя открытый ключ Алисы. В соответствии с выражением (18). И отправляет c Алисе.

$$(e, N)(e, N) : C = E(M) = M^e \bmod(N) \quad (18)$$

Алиса принимает зашифрованное сообщение c . Используя закрытый ключ $(d, N)(d, N)$, расшифровывает сообщение. В соответствии с выражением (19).

$$M = D(C) = C^d \bmod(N) \quad (19)$$

Принцип работы алгоритма RSA представлен на рисунке 4. [10]

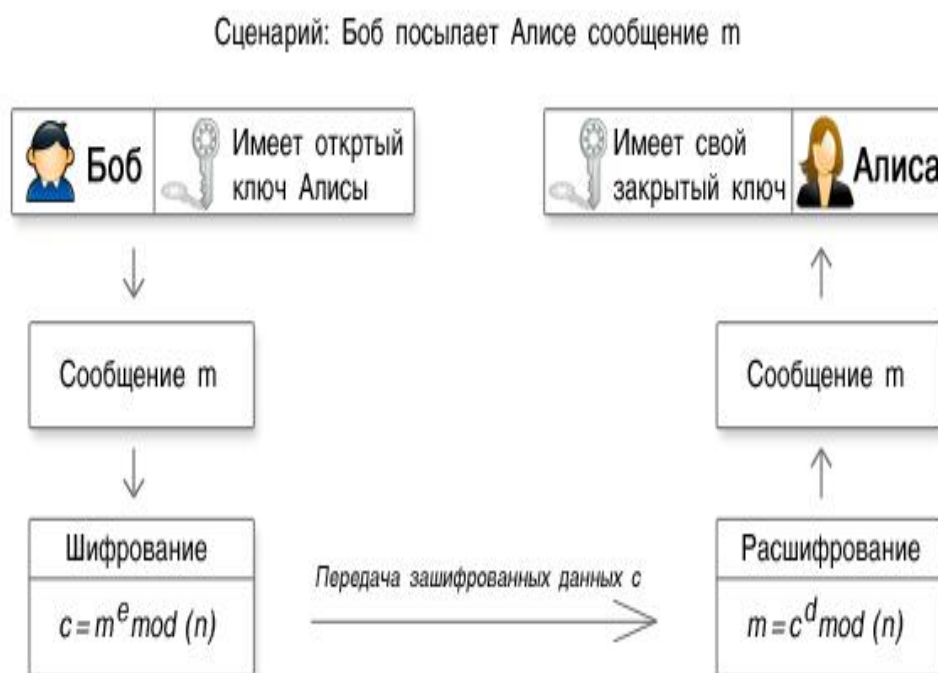


Рисунок 4 – Принцип работы алгоритма RSA

2 Методы криптоанализа простых шифров

2.1 Метод полного перебора

Криптоанализ — наука о способах получения исходного значения зашифрованных данных, не имея доступа к секретному ключу, необходимого для этого. Чаще всего под этим понимается нахождение ключа. Иными словами — криптоанализ есть взлом шифра. Термин был введён американским криптографом Уильямом Ф. Фридманом в 1920 году. [11]

В криптографии на методе полного перебора базируется криптографическая атака методом «грубой силы». Её основной особенностью является возможность применения против практически любого шифра. Но такая возможность существует лишь в теории так как зачастую требует огромные затраты ресурсов и времени. Использование атаки методом «грубой силы» актуально в тех случаях, когда не представляется возможным найти слабые места в криптосистеме, которая подвергается атаке. При обнаружении таких недостатков криптоаналитиками разрабатываются методики проведения криптоанализа, которые базируются на их особенностях, что способствует упрощению вскрытия шифра.

Устойчивость к атаке методом грубой силы определяется используемым в криптосистеме ключом шифрования. С увеличением длины ключевой последовательности сложность взлома методом грубой силы возрастает экспоненциально. В простейшем случае шифротекст длиной в N битов взламывается, в худшем случае, за время, пропорциональное 2^N . Среднее время вскрытия шифра в этом случае в два раза меньше и составляет 2^{N-1} . [12]

Наибольшая эффективность метода полного перебора достигается при применении его к простым шифрам, например к шифру Цезаря. Для взлома данного шифра необходимо произвести замену каждой буквы шифртекста на букву, которая стоит на один знак левее в алфавите. Если при этом не удалось получить исходный открытый текст, то необходимо повторить

действие, но уже сместив буквы на два знака левее. И так далее, до тех пор пока в результате не получится читаемый текст. [13]

2.2 Метод статистического криптоанализа – подсчет частоты встречи символов

Статистический криптоанализ — метод криптографического анализа, с помощью которого могут быть вскрыты многие типы шифров.

В общем случае статистический анализ выполняется следующим образом:

- по перехваченному шифротексту E производится вычисление некоторой статистики. Эта статистика такая, что для всех осмысленных сообщений M она принимает значения, которое мало отличается от S_k , величины, которая зависит только от частного используемого ключа;
- полученная таким образом величина служит для выявления таких возможных ключей, для которых значение S_k лежит окрестности наблюдаемого значения.

Статистика, которая не зависит от K или меняется в зависимости от M так же сильно, как и в зависимости от K , не может быть существенна для выделения некоторого подмножества ключей. Так, в шифрах перестановки подсчёт частот встречи букв не даёт абсолютно никакой информации о K – для любого K эта статистика остается одинаковой. Поэтому нет пользы от подсчёта частот для вскрытия шифров перестановки. Более точно данной статистике S можно приписать некоторую «разрешающую мощность». Для каждой величины S имеется условная ненадёжность ключа $H_s(k)$ (ненадёжность при фиксированном значении S) и это всё, что известно о ключе. Среднее арифметическое взвешенное этих величин даёт среднюю ненадёжность ключа при известном S , где S является априорной вероятностью конкретного значения S . Разность объёма ключа $H(K)$ и этой средней неопределённости измеряет «разрешающую мощность» статистики

S. В идеальном шифре все статистики данной криптограммы не зависят от частного используемого ключа *M*. [14]

Буквы любого открытого текста, которые написанны на любом языке, использующим алфавит, встречаются в тексте разное количество раз. К примеру, если взять произвольный открытый текст большой длины, который написан на русском языке и подсчитать в нем число букв от А до Я, то получится результат, который представлен на рисунке 5.

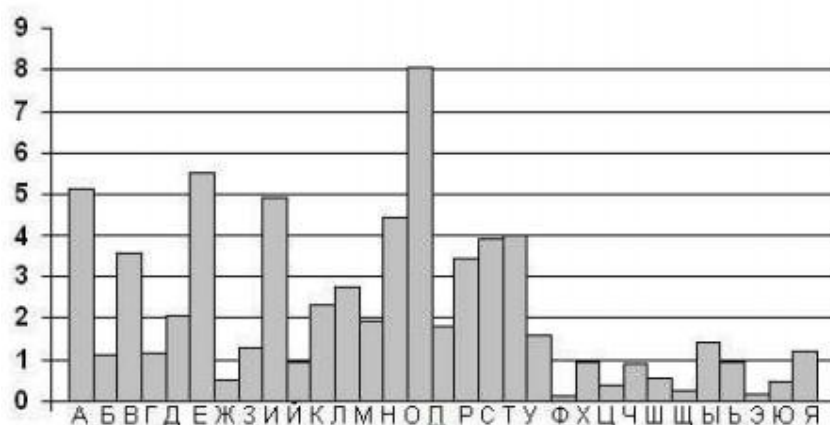


Рисунок 5 – Частоты букв русского языка

Высота столбца каждой буквы говорит о процентном содержании в тексте этой буквы. Из этого рисунка (гистограммы) получается, что самой частой буквой в русском открытом тексте является буква О, а самыми редкими буквами - Ф, Щ, Э.

Если в открытом сообщении часто встречается какая-нибудь буква, то в сообщении, которое зашифровано с помощью такого шифра простой замены, часто будет встречаться аналогичный ей символ или буква. Следовательно, в случае шифра простой замены, имеет смысл, выделить из шифртекста самую часто встречающуюся букву (предварительно подсчитав частоту встречи всех букв в шифртексте) и найти сопоставление этой буквы букве языка в соответствии с рисунком 2.1. И продолжить эту процедуру сопоставления частот букв шифртекста частотам букв языка и таким образом, вскрыть данный шифр. Из-за особенности некоторых текстов или небольшой их длины самому часто встречающемуся символу шифртекста не

всегда соответствует самый частый символ языка, а возможно символ с меньшей частотой встречи.[15]

2.3 Метод статистического криптоанализа – подсчет частоты встречи биграмм в шифротексте

На протяжении многих сотен лет криптоанализу шифротекстов помогает частотный анализ появления отдельных символов и их сочетаний. Вероятности встречи отдельных букв в тексте сильно различаются. Для русского языка, к примеру, буква "о" появляется в 45 раз чаще буквы "ф" и в 30 раз чаще буквы "э". Производя анализ достаточно длинного текста, который зашифрован методом простой замены, можно по частотам появления символов произвести обратную замену и восстановить исходный текст. Относительные частоты появления русских букв представлены в таблице 8.

Таблица 8 – Относительные частоты появления русских букв

Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота
о	0,09	в	0,038	з	0,016	ж	0,007
её	0,072	л	0,035	ы	0,016	ш	0,006
а	0,062	к	0,028	б	0,014	ю	0,006
и	0,062	м	0,026	ь, ъ	0,014	ц	0,004
н	0,053	д	0,025	г	0,013	щ	0,003
т	0,053	п	0,023	ч	0,012	э	0,003
с	0,045	у	0,021	й	0,01	ф	0,002
р	0,04	я	0,018	х	0,009		

Относительная частота встречи пробела или знака препинания в русском языке составляет 0,174. Помимо этого, порядок букв в словах и фразах естественного языка подчиняется определенным статистическим закономерностям. Частотный анализ также учитывает частоту появления различных сочетаний букв: к примеру, пара стоящих рядом букв «ся» в русском языке более вероятна, чем «цы», а «оь», которые не встречаются никогда. Для большинства естественных языков такая статистика уже просчитана и описана. [16]

Пусть имеется шифротекст «АЗЮЖЕ СШГТООИПЕР», который представлен в таблице 9.

Таблица 9 – Шифротекст

	1	2	3	4
1	А	З	Ю	Ж
2	Е		С	Ш
3	Г	Т	О	О
4	И	П	Е	Р

Рассматривая маловероятные сочетания букв, легко найти истинную последовательность столбцов. Так, сочетание ГТ в третьей строке шифротекста указывает на то, что после первого столбца вряд ли следует второй столбец. Рассчитаем, какой столбец скорее всего следует за первым. Для этого можно воспользоваться таблицей логарифмов частот встречи биграмм в русском тексте. Частота следования одного столбца за другим равна произведению частоты встречи биграмм в строках этих столбцов. Достаточно суммировать логарифмы частоты биграмм, а потом выбрать сочетание столбцов с максимальной частотой. Для частоты следования за первым столбцом два, три и четыре в соответствии с выражениями (20), (21), (22).

$$p(1-2)=p(AЗ)p(E) p(ГТ)p(ИП)=7+9+0+5=21 \quad (20)$$

$$p(1-3)=p(AЮ)p(ЕС)p(ГО)p(ИЕ)=6+8+8+8=30 \quad (21)$$

$$p(1-4)=p(AЖ)p(ЕШ)p(ГО)p(ИП)=7+5+8+7=27 \quad (22)$$

В данном случае, скорее всего, после столбца под номером один следует столбец под номером три. Для такой небольшой таблицы с шифротекстом, можно перебрать все возможные варианты перестановок – их

всего лишь 24. В случае большего числа столбцов будет разумным оценить частоты пар сочетаний разных столбцов и решить оптимизационную задачу, которая укажет перестановку столбцов, дающую фрагменты естественного текста с большей частотой. В данном случае лучший результат достигается при расстановке столбцов (2413), что практически вдвое по оценке частотей достовернее ближайшей к ней по частоте расстановки (4132). После того, как столбцы шифротекста расставлены, необходимо правильно расставить строчки таблицы по смыслу фрагментов текста. Результат представлен в таблице 10.

Таблица 10 – Результат расстановки строк

	2	4	1	3
1	3	Ж	А	Ю
2		Ш	Е	С
3	Т	О	Г	О
4	П	Р	И	Е

Текст в ней уже читается и, расставив строки в порядке (4123), получается открытый текст: ПРИЕЗЖАЮ ШЕСТОГО.

Пример дешифровки шифра решетки Кардано. Шифротекст представлен в таблице 11.

Таблица 11 – Шифротекст

З	Т	П	
О	Ж	Ш	Р
Е	И	Г	А
Е	С	Ю	О

В этом шифротексте содержится 2 слова, при условии, что он длиной 16 букв, а одно слово русского языка содержит в среднем 7 букв, а также имеется пробел. Дешифрование проще всего начать с биграммы СТ.

Допустим, что самая частая биграмма в русском языке СТ входит в одно из слов текста. Так как буква Т располагается в решетке выше буквы С,

из этого следует, что между ними произошел поворот решетки. Поворот решетки представлен на в таблице 12.

Таблица 12 – Поворот решетки на 90°

	Т		

С учетом того, что эти буквы принадлежат различным прорезам решетки, получается вид таблиц представленных в таблице 13.

Таблица 13 – Поворот решетки

		?	?		Т					П			?	?		
		?	?		О						Р			?	?	
Е						?	?									А
	С					?	?								Ю	
0°				90°				180°				270°				

Проверка правильности этого предположения осуществляется чтением, еще не ясные буквы заменяются точками: ..ЕСТО..ПР....АЮ. В решетке помечена клетка буквы С цифрой 1, а клетка буквы Т обозначена цифрой 2.

При составлении решеток отверстия в решетке размещают таким образом, чтобы в каждой колонке и каждом столбце их было по одной, это гарантирует случайное и равномерное заполнение квадрата буквами. В этом случае имеется всего два варианта: 1234 и 123'4', которые представлены в таблице 14.

Таблица 14 – Макет решетки

		3'	4
		3	4'
2			
	1		

Второй вариант 123'4' не подходит, так как не покрывает весь квадрат при поворотах, и остается 1234, что дает ШЕСТОГОПРИЕЗЖАЮ.

Сообщение уже ясно, хотя расшифровка начата с неправильного поворота решетки. С учетом этого получается сообщение: ПРИЕЗЖАЮ ШЕСТОГО.

Шифр Гронсфельда вскрывается достаточно легко, при учете следующего обстоятельства: так как цифр всего 10, то имеется лишь 10 вариантов прочтения каждой буквы. Необходимо выписать их столбцами таким образом, что одной строке соответствуют буквы одного значения ключа. Номер варианта будет определяться цифрой ключа от 0 до 9. При этом получится таблица, при правильном выборе по одной букве из каждой колонки которой получится исходный текст. В таблице 15 представлены варианты записи букв.

Таблица 15 – Варианты записи букв

Ключ	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
Вариант 0	ф	п	ж	и	с	ь	и	о	с	с	а	х	и	л	ф	и	у	с
Вариант 1	у	о	е	з	р	ы	з	н	р	р		ф	э	к	у	з	т	р
Вариант 2	т	н	д	ж	п	ь	ж	м	п	п	я	у	ж	й	т	ж	с	п
Вариант 3	с	м	г	е	о	щ	е	л	о	о	ю	т	е	и	с	е	р	о
Вариант 4	р	л	в	д	н	ш	д	к	н	н	э	с	д	з	р	д	п	н
Вариант 5	п	к	б	г	м	ч	г	й	м	м	ь	р	г	ж	п	г	о	м
Вариант 6	о	й	а	в	л	ц	в	и	л	л	ы	п	в	к	о	в	н	л
Вариант 7	н	и		б	к	х	б	з	к	к	ь	о	б	д	н	б	м	к
Вариант 8	м	з	я	а	й	ф	а	ж	й	й	щ	н	а	г	м	а	л	й
Вариант 9	л	ж	ю		и	у		е	и	и	ш	м		в	л		к	и
Сообщение	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Если прочтение текста напрямую осуществить не удалось, то необходимо рассуждать. Самым часто встречающимся символом текста является — пробел, а разбиение фразы на слова может оказать большую помощь в расшифровке, как при вскрытии шифра решетки Кардано. Так как длина шифротекста равна 19 символам, то, скорее всего, она состоит из двух или трех слов, которые разделены пробелами. Хорошее положение для пробела дает лишь вариант 1, а другие варианты, 7 и 9 или их сочетания маловероятны. Из этого следует, что, скорее всего, шифротекст разбивается на два слова: ФПЖИСЬИОСС ХИФИУСС. Это значит, что в 11 позиции текста стоит пробел и в той же позиции ключа находится цифра 1.

У выделенных слов шифротекста одинаковое окончание СС, и, вероятно, что период ключа делит 9 — длину второго слова вместе с пробелом. Предположим, что в этом случае одинаковые окончания слов 1 текста попали на одинаковые участки ключа и дали одинаковые символы шифротекста. Может показаться, что это маловероятно, чтобы встречаться на практике. Но таких «находок», которые помогают расшифровке, всегда достаточно много в сообщениях большой длины. Длину ключа можно подобрать вручную. Есть два выбора для периода ключа: 3 и 9. Период ключа 3 представлен в таблице 16.

Таблица 16 – Период ключа 3

Ключ	?	1	?	?	1	?	?	1	?	?	1	?	?	1	?	?	1	?	?
Вариант 0	ф		ж	и		ь	и		с	с		х	и		ф	и		с	с
Вариант 1	у	о	е	з	р	ы	з	н	р	р		ф	э	к	у	з	т	р	Р
Вариант 2	т		д	ж		ь	ж		п	п		у	ж		т	ж		п	П
Вариант 3	с		г	е		щ	е		о	о		т	е		с	е		о	О
Вариант 4	р		в	д		ш	д		н	н		с	д		р	д		н	Н
Вариант 5	п		б	г		ч	г		м	м		р	г		п	г		м	М
Вариант 6	о		а	в		ц	в		л	л		п	в		о	в		л	Л
Вариант 7	н			б		х	б		к	к		о	б		н	б		к	К
Вариант 8	м		я	а		ф	а		й	й		н	а		м	а		й	Й
Вариант 9	л		ю			у			и	и		м			л			и	И
Сообщение	?	о	?	?	р	?	?	н	?	?		?	?	к	?	?	т	?	?

Таблица ощутимо поредела, но по-прежнему остается достаточно сложной для непосредственного прочтения. Поэтому необходимо подобрать символ ключа, который стоит в первой позиции, перебором десяти вариантов. Так как ключ длиной 3 циклически повторяется, то этот же символ стоит в 4, 7, 10, 13, 16 и 19 позициях ключа. Вероятность варианта для первой цифры ключа равна произведению вероятностей биграмм, состоящих из символа по этому варианту расшифровки и следующего за ним, уже известного символа. Если l_i — буква текста, стоящая на месте i , то вероятность одного из 10 вариантов, которые представлены выражением (23).

$$(1_1 1_2;) P(1_4 1_5) P(1_7 1_8) P(1_{10} 1_{11}) P(1_{13} 1_{14}) P(1_{16} 1_{17}) \quad (23)$$

Для вычисления вероятности биграмм необходимо пользоваться логарифмами вероятностей биграмм, которые достаточно суммировать. В результате для вариантов от 0 до 3 получаются численные значения вероятностей, представленные в выражениях (24), (25), (26), (27).

$$p(0)=p(\Phi O)P(\text{ИР})P(\text{ИН})p(C)p(\text{ИТ})=43; \quad (24)$$

$$p(1)=p(YO)P(\text{ЭР})p(3H)p(P)p(3K)p(3T)=23; \quad (25)$$

$$p(2)=p(TO)P(\text{ЖР})p(\text{ЖН})p(\Pi)p(\text{ЖК})p(\text{ЖТ})=27; \quad (26)$$

$$p(3) =p(CO)P(EP)P(EH)p(O)p(EK)p(ET)=50. \quad (27)$$

В результате для первой цифры ключа получается наиболее вероятным вариант расшифровки. Вариант расшифровки представлен в таблице 17.

Таблица 17 – Вариант расшифровки

Ключ	?	1	?	?	1	?	?	1	?	?	1	?	?	1	?	?	1	?	?
Вариант 0			ж			ь			с			х			ф				с
Вариант 1		о	е		р	ы		н	р			ф		к	у		т		Р
Вариант 2			д			ь			п			у			т				П
Вариант 3	с		г	е		щ	е		о	о		т	е		с	е		о	О
Вариант 4			в			ш			н			с			р				Н
Вариант 5			б			ч			м			р			п				М
Вариант 6			а			ц			л			п			о				Л
Вариант 7						х			к			о			н				К
Вариант 8			я			ф			й			н			м				Й
Вариант 9			ю			у			и			м			л				И
Сообщение	с	о	?	е	р	?	е	н	?	о		?	е	к	?	е	т	о	?

После проделанных действий сообщение читается достаточно просто. Необходимо выбрать одну из 10 строк с вариантом для 3 цифры ключа. В

результате получается ключ и текст сообщения. Текст сообщения представлен в таблице 18. [17]

Таблица 18 – Текст сообщения

Ключ	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3
Сообщение	с	о	в	е	р	ш	е	н	н	о		с	е	к	р	е	т	н	о

2.4 Метод разложения числа на простые множители

Поиском эффективных способов разложения целых чисел на множители математики занимаются очень давно. Самым простым метод разложения числа n на множители – перебор простых делителей которые не превышают $\sqrt{n}$.

Существует также другой метод (метод Ферма), основанный на представлении числа n в виде разности квадратов. Разность квадратов представлена выражением (28).

$$n = a^2 - b^2 = (a + b)(a - b) \quad (28)$$

Ферма предложил, вычисляя $\text{НОД}(n, a - b)$, попытаться найти нетривиальный делитель n . Он предложил способ, позволяющий найти требуемое представление. Если разлагаемое число имеет два не очень различающиеся по величине множителя, этот способ позволяет определить их быстрее, чем простой перебор делителей. Лежандр обратил внимание на то, что при таком подходе достаточно получить сравнение (29).

$$a^2 \equiv b^2 \pmod{n} \quad (29)$$

Не каждая пара чисел, удовлетворяющая ему, позволяет разложить n на множители. Для нахождения чисел, связанных соотношением (28), Лежандр использовал непрерывные дроби. В его работе описано некоторое усовершенствование метода Ферма.

По сложности как простой переборный метод, так и метод Ферма оцениваются величиной $O(\sqrt{n})$, однако последний метод может оказаться эффективнее, если делители числа n близки друг к другу.

Для достаточно больших чисел n время работы таких алгоритмов становится неприемлемым. Прежде чем приступить к факторизации таких чисел, следует убедиться в том, что они действительно составные. Для этого лучше всего использовать один из вероятностных тестов на простоту, например, алгоритм Миллера—Рабина.

Методов факторизации с экспоненциальной сложностью для больших чисел используются в сочетании с субэкспоненциальными методами факторизации, которые будут описаны далее, и применяются, как правило, для предварительного отделения небольших простых делителей у факторизуемого числа. [18]

3 Параметры используемых шифров

3.1 Параметры шифра простой замены

В данном разделе приводится описание параметров простых шифров, которые были заданы для проверки правильности криптоанализа, шифрования и расшифрования.

Для шифрования текста был выбран алфавит замены представленный в таблице 19.

Таблица 19 – Алфавит замены

Алфавит	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	щ	ъ	ы	ь	э	ю	я
Алфавит замены	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	щ	ъ	ы	ь	э	ю	я	а	б	в	г

Выбран следующий открытый текст:

«Квантовая криптография изучает способы генерации криптографических ключей, секретность которых гарантируется фундаментальными законами квантовой механики.».

После произведения шифрования получен следующий шифротекст:

«Е У W И О К У W Ы Ь Е М Д Л О К Z М W P Д Ы Ь Д Г П У W Б О
Б Н Л К Н К Х Ч Ь Z Б И Б М W Т Д Д Ь Е М Д Л О К Z М W P Д У Б Н Е Д
С Ь Е Ж Ь У Б Ь Н Б Е М Б О И К Н О Ш Ь Е К О К М Ч С Ь Z W M W И О Д
М П Б О Н Ы Ь Р П И А».

При расшифровании данного шифротекста ошибок не выявлено.

При Выполнении криптоанализа методом полного перебора получился результат, представленный на рисунке 6.

1	2
DEFGHIJKLMN...	ОЖДСЦТЖДСРОФНУЦТЗФДШНСДНМЧЫДКЦХУТХТЯРДЗКСКФДЬННДОФНУЦТЗФДШНЫКХОНЩРОПВЫКДХКОФКЦСТХЦ...
EFGHIJKLMNO...	НЕГРХСЕГВСНУМТХСЖУГЧМВСМЛЦГХИХСФТСФСДЮСЖИРИУГЦММСНУМТХСЖУГЧМЫИФНМШСНОАЫСИФИНУИХРСФ...
FGHIJKLMNOR...	МДВГФРДВАВМТЛСФРЕТВЦЛАВЛКХЩБЗФВУСРУРГЭВЕЗПЗТВШЛЛВМТЛСФРЕТВЦЛЩЗУМЛЧВМНЩЗВУЗМТЗФПРУФЮВ...
GHIJKLMNORQ...	ЛГБОУПГБ АЛСКРУПДСБХК АКИФШБЖУАТРПТПВБДЖОЖСБЧКАЛСКРУПДСБХКШЖТЛКЦАЛМЯШЖАТЖЛСЖУОПТУЗАЛ...
HIJKLMNORQR...	КВАНТОВАЯ КРИПТОГРАФИЯ ИЗУЧАЕТ СПОСОБЫ ГЕНЕРАЦИИ КРИПТОГРАФИЧЕСКИХ КЛЮЧЕ СЕКРЕТНОСТЬ КОТОРЫХ Г...
IJKLMNORQRS...	ИБЗМСНБЗЮЯИПЗОСНВПЗУЗЮЯЗТЦЗДСЯРОНРНАЪЯВДМДГПЗХЗЯИПЗОСНВПЗУЗДРИЗФЯИКЗДЯРДИГДСМНРСЫ...
JKLMNORQRST...	ЗАУЛРМАУЗЮЗОЖНРМБОУТЖЮЖЕСХУГРЮПНМГМЗЩЮБГЛГОУФЖЮЗОЖНРМБОУТЖХГПЗЖУЮЗИЪХГЮПЗГОРЛ...
KLMNORQRST...	ЖЗХКПЛЗХЪЭЖНЕМПЛАНХСЬЭДРФХВГЭОМЛОЛУШЭАВКВНХУЕЕЭЖНЕМПЛАНХСФВОЖЕТЗЖЫФВЗОВЖНВПКЛОЩ...
LMNORQRSTU...	ЕУWИОКУWЫЬЕМДЛОКZМWPДЫЬДГПУWБОНЛКНХЧЬЗБИБMWТДДЬЕМДЛОКZМWPРДУБНЕДСЬЕЖЬУБЬНБЕМБОИКНОШ...

Рисунок 6 – Результат приптоанализа простой замены

3.2 Параметры шифра сдвиговой замены

Для шифрования текста был выбран алфавит замены представленный в таблице 20.

Таблица 20 – Алфавит замены для сдвиговой замены

Алфавит	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ъ	ы	ь	э	ю	я
Алфавит замены	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	щ	ъ	ы	ь	э	ю	я	а	б	в	г	д

Выбран следующий открытый текст:

«в связи с необходимостью повышения безопасности облачных вычислений становится важным создание эффективных алгоритмов полностью гомоморфного шифрования, позволяющего производить произвольные вычисления над данными без предварительной расшифровки».

После произведения шифрования получен следующий шифротекст:

«А Ю П А Э Е Ж Ю П Ю Л Г М Z У М В Ж К М П Р Ъ Ъ Ю Н М А Щ Ц Г Л Ж Э Ю Z Г Е М Н У П Л М П Р Ж Ю М Z И У Х Л Щ У Ю А Щ Х Ж П И Г Л Ж Ю П Р У Л М А Ж Р П Э Ю А У Д Л Щ К Ю П М Е В У Л Ж Г Ю Ы Т Т Г З Р Ж А Л Щ У Ю У И Б М О Ж Р К М А Ю Н М И Л М П Р Ъ Ъ Ю Б М К М К М О Т Л М Б М Ю Ц Ж Т О М А У Л Ж Э Ю Н М Е А М И Э Ъ Ч Г Б М Ю Н О М Ж Е А М В Ж Р Ъ Ю Н О М Ж Е А М И Ъ Л Щ Г Ю А Щ Х Ж П И Г Л Ж Э Ю Л У В Ю В У Л Л Щ К Ж Ю Z Г Е Ю Н О Г В А У О Ж Р Г И Ъ Л М Ю О У П Ц Ж Т О М А З Ж».

При осуществлении криптоанализа методом полного перебора получился результат, представленный на рисунке 7.

БЭЮА ABCDEF...	ДВУДАКЛВУВПЗРГЧРЖЛОРУФЮ ВСРДЭЪЗПЛАВГЗКРСВУПРУФЛВРГНВЩПЭЧВДЭЩЛНУЗПЛВУФВПРДЛУАВДВИПЭОВУРК...
ЭЮА ABCDEFG...	ГАТГ ИКАТАОЖПВЦПЕКНПТУЭЯРПГЫЩЖОК АВЖИПРБТОПТУКАПВМБШОБЦАГШКТМЖОКАТУБОПГУТ АГБЗОБНАТП...
ЮА ABCDEFGH...	В СВЯЗИ С НЕОБХОДИМОСТЬЮ ПОВЫШЕНИЯ БЕЗОПАСНОСТИ ОБЛАЧНЫХ ВЫЧИСЛЕНИ СТАНОВИТСЯ ВАЖНЫМ СОЗДА...
Я ABCDEFGHI...	БЯРБЮЖЭЯРЯМДНАФНГЗЛНРСЭЯОНБЪЧДМЗЮАДЖНОЗРМНРСЭЯНАКЗЦМЪФЯБЪЦЗРКДМЗЯРСЗМНБЗСРЮБЗЕМЪЛ...

Рисунок 7 – Криптоанализ шифра сдвиговой замены

3.3 Параметры шифра гаммирования

Для шифрования текста выбраны следующая гамма-последовательность: «С Ч Л Э У Ы Р Ю Я Д Х К Р Ц Щ З Б Т К КТ Ж Д Э Ю Ф Ш П К».

Также был выбран открытый текст: «основным достоинством криптографии на эллиптических кривых является то, что для задачи дискретного логарифмирования в группе точек эллиптической кривой не существует субэкспоненциальных алгоритмов решения, это позволяет сократить длину ключа и увеличить производительность».

При выполнении шифрования был получен следующий шифротекст:

«Я Е Е У В Ю Ъ Ю Ъ Л Г Ш Ю Я Ф Щ Р С Г Е Т Н Р М Т М О Р З О
Ю Ш Ю Л Р А У Е Ы Х Ц К Ж В Ж Р З Б Л Ж К Б О В Ю З Ю Я Ч У Р М В
Е Ф Э Т Э Ы Ж М С Д Р А П Ц С И Ж У Э В Т Ь И Ф Ч О Е Е Х А О Я Ч У
Г Т Л Щ К У М Д Г У Ч Ф А Э Т И К Ц Ч У К Т Ш Е Ц Ш М Ы Ч Ц Ц Л Ъ
Ф Б Ч З В Г Ю Ц У Ш Л С Г К Я А Ф О Ж Е Д К М Щ Ч Д Л М У С Е Ъ П
Р Л Ш М Э Р И И О Ж С Ж Ж А О Щ Р Р Э К В Г Т Ч Ы Ы Ш Х Ц Щ А Я Ш
Ж Г Р Щ Ч А Ъ А Х Ф Х Ф У Ф Р Х К Е Ш Н Ч О Ц И Ю З Р Ф Ф Ы Б Л Р
Я Щ Ы А Ф А В Д О Т Щ Э С Р Ъ С Ж И Я Т Ь П Е Ш Ж Э Р О Ч И».

3.4 Параметры шифра одиночной перестановки

Для выполнения шифрования был выбран следующий ключ: «КРИПТОГРФИЯ».

Также был выбран открытый текст:

«Однако, в настоящее время они существуют лишь в теории, единственное успешное вычисление которое было успешно проведено квантовым компьютером это факторизация числа пятнадцать "микромаскетом" квантового вычислителя».

При выполнении шифрования был получен следующий шифротекст:

«Р Е И С С О Ш О К Э А М В У Ш Е Ч О П Е М М И О О Я Н Я
Н Т Р Н Е О О Т Т К И А О И А М Т С Е Е О К М О И К В Ч О Т У Е В
Н И Ы П А Ъ Ф О Н С О Е С И Е Ы Т С Д Ы О Р А Т Г Л А Щ И И Н К

ВОЕТСКВТДООЮОЕОЕЛРНЮАЧМТЛВВЩЬДУИРЕН
ЗМКЕЛИОВОУЕВРОЛЕОЕНЕСВНПЛЕНОТЦИКЫ
СЯВТТШНБВПЯРАИ».

3.5 Параметры шифра двойной перестановки

Для выполнения шифрования был выбран следующие ключи:
«ИСПОЛНИТЕЛЬСТВО» и «СООТВЕТСТВУЮЩИЙ».

Также был выбран открытый текст:

«Эксперты отмечают, что протокол, который основан на эллиптических кривых, имеет небольшой размер ключа без ущерба для криптографической стойкости, благодаря чему эллиптическую криптографию можно использовать в тех областях, где существуют проблемы из-за ограничения памяти и вычислительных мощностей».

При выполнении шифрования был получен следующий шифротекст:

«ЫЕЛМСЛДКТЕБЕКСФЗЙИАОЙНТИАОСЕОСЕ
ЛЮЕСОБАЙХЕЯЛТЕЖЕЕРЬТИЬПЬСЫЕТДГАЧР
МБЯОСТЧЧЬАВРОЛЛСЛБРТЫООПТЩХЕИЧЧЕИО
ХТКЛЛВООИНЛЗКТДТБОИИТЛГРЮХИОУИОКЧГ
ЩОПЧКЛХЕЕИАКАРБОЛЭССССЕСНКЫОНОАУЧН
ЙАЮИОКРРППИЕТКНМИТОЮТУРМИР».

3.6 Параметры шифра RSA

Для шифрования с помощью алгоритма RSA были выбраны параметры, представленные в таблице 21.

Таблица 21 – Параметры шифра RSA

p	q	Пара открытых ключей		Пара закрытых ключей	
		e	n	d	n
67	47	5	3149	2429	3149

Также был выбран открытый текст: «стойкость криптографическая система rsa основывается на сложности задачи факторизации больших чисел».

При выполнении шифрования был получен следующий шифротекст:

«2807 168 2494 2381 2494 2807 168 1083 0 2381 3108 2367 466 168 2494 1024

3108 1 616 2367 2936 1478 2807 2381 1 1592 0 2807 2367 2807 168 1478 61 1 0
 2494 2807 2860 2494 243 2063 243 1 1478 168 2807 1592 0 2860 1 0 2807 452
 2494 1062 2860 2494 2807 168 2367 0 1278 1 3125 1 2936 2367 0 616 1 2381
 168 2494 3108 2367 1278 1 1868 2367 2367 0 32 2494 452 1083 1952 2367 2997
 0 2936 2367 2807 1478 452».

3.7 Параметры шифра маршрутной перестановки

Для выполнения шифрования была выбрана решетка, представленная на рисунке 8.



Рисунок 8 – Трафарет

Для выполнения шифрования был выбран следующий текст: «традиционные криптографические алгоритмы строятся с помощью сочетания большого числа относительно несложных математических преобразований таким образом, чтобы обеспечить лучшие характеристики итогового алгоритма».

В результате выполнения шифрования был получен следующий шифротекст: «Ф Ф Б Ж Н Я П У Р П Э К А М Ф Т Х Ч С Е Т Д Х Л Ы П Ч П
 М З Б Д М Е Т Щ П Ч П Э Б Х У Т Т З Ш Ц Б Б У Г Р Р Р Ч Я А А Б У Т Ш З
 Ц К У О Б Б Г Т М Ю Ъ Ч К У В Щ Л Х М В Г Ч Ш Т С У Л Ц Ж Н Ц Ф Д Р
 З У П П И С Г Ы Д В О В Ц Ж О Д Ы П Ь И У М Н Ц Б У Щ М У Д Т В М П
 Д Д Ц П Г В Ф В О К О Г Ч З Х Г К Р Р А Щ Ц Ч З В Р Г К Т С К П Ч Я Б Н
 Ч Ш Ъ Н П Е Ъ Г Т В О У З Ф Т Ч Ч М М Л Г К Ф Т Н Ф З С Е Р Г Б Н З Ч Ц
 О Х О В».

3.8 Параметры шифра полиалфавитной замены

Для выполнения шифрования с помощью данного метода был выбран ключ: «5618».

Также был выбран открытый текст: «в теории уже имеются доказательства связи множества используемых вычислительно сложных задач с другими аналогичными задачами».

В результате выполнения шифрования был получен следующий шифротекст: «Д Г У З Т Щ П О В Х И К А Л Р П Г Ч Ф А Б И П М Д С Ж Ч И Н Ю Х У Д Д И Ч З Б К Л Г Н П Т Р М Ц Х Д В Г К У У Ч С А Л Х З Р Ь Ч Г М А Ъ М У Н Н У З П Д У У В У Н Т З П Я Ю Е Н Г Ж В Ы А У Г О Ц Ш Ж Л О Н А В С К С У Ж Л Щ С Ъ О Н И О Е З В Щ Д Н Л».

4 Архитектура программного средства

4.1 Средства реализации программного средства

Данное программное средство реализовано с помощью объектно-ориентированного языка высокого уровня C++ и кроссплатформенного инструмента разработки программного обеспечения QT 5.8, а также среды разработки QT Creator 4.2.1.

Кроссплатформенный инструмента разработки QT 5.8 позволяет создавать графические интерфейсы для различных операционных систем с помощью простой компиляции программ для каждой операционной системы без изменения исходного кода программы.

Qt Creator – кроссплатформенная интегрированная среда, предназначенная для разработки Qt-приложений на таких языках программирования, как C, C++ и QML. В нее включен удобный отладчик, а также различные визуальные средства разработки интерфейса, использующие QtWidgets и QML. Среда разработки поддерживает проекты таких средств, как Qmake, Cmake, Autotools. А для тех проектов, которые были созданы при помощи других программ, была предусмотрена возможность использования среды в качестве редактора исходных программных кодов.

Архитектура программного средства – это его строение, а именно представление программного средства как системы, которая состоит из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают либо отдельные программы, либо модули программного средства.

Данное программное средство относится к такому классу архитектуры как цельная программа. Эта архитектура представляет собой случай, когда в состав программного средства входит только одна программа. Такой класс архитектуры был выбран из-за того, что программное средство должно выполнять несколько простых функций таких как: шифрование, дешифрование и криптоанализ. Такая архитектура требует описания только

класса архитектуры, так как отображение внешних функций на эту программу достаточно просто, а описывать способ взаимодействия с другими системами не нужно из-за того, что программное средство взаимодействует только с пользователем. Взаимодействие с пользователем описывается в руководстве по применению к программному средству. Схема архитектуры программного средства представлена на рисунке 9.

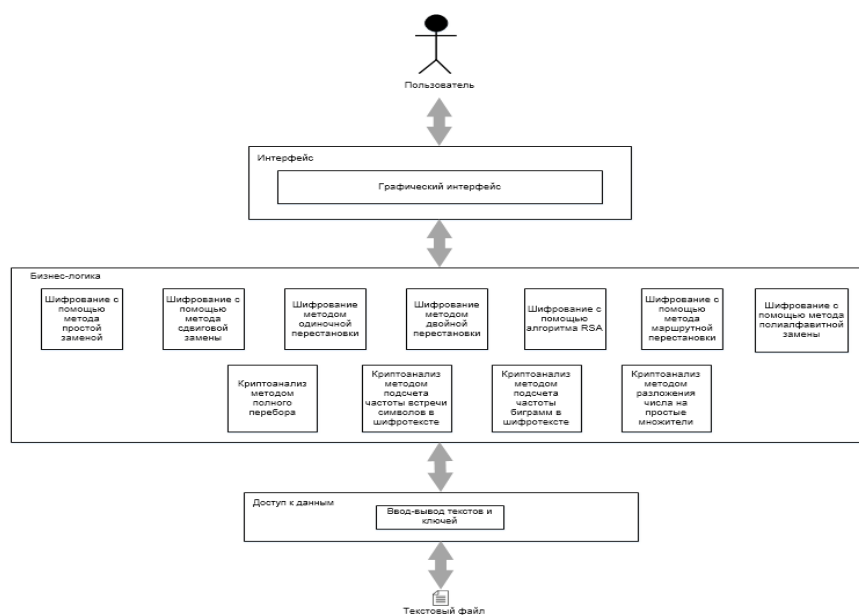


Рисунок 9 – Схема архитектуры программного средства

4.2 Описание классов программного средства

Класс «SingleShift» реализует шифр одиночной перестановки. Включает в себя следующие методы:

- «encrypt» – метод, реализующий одиночное шифрование. На вход метода поступает строка открытого текста типа QString. Возвращаемым значением является строка шифротекста типа QString;
- «decrypt» – метод, реализующий расшифрование шифротеста методом одиночной перестановки. На вход метода поступает строка шифротекста типа QString. Возвращаемым значением является строка открытого текста типа QString;
- «setkey» – метод, с помощью которого для объекта класса SingleShift устанавливается ключ шифрования.

Класс «Factorization» реализует разложение числа на простые множители. Содержит статическую функцию «factorize», с помощью которой выполняется разложение. Результатом выполнения является вектор простых чисел типа BigInteger.

Класс «KeyFileDialog» – класс диалогового окна, предназначенного для формирования ключа для шифров простой замены и сдвигового шифра замены (класс «SimpleReplacement»). Данный класс включает в себя следующие методы:

- «readDict» – метод класса, реализующий чтение словаря, полученного с помощью данного окна. Результатом выполнения функции является словарь типа QMap<QString, QString>;

- «setCustomDict» – метод класса, реализующий ввод словаря из файла. На вход функции подается словарь типа QMap<QString, QString>.

Класс «SimpleReplacement» реализует шифры простой замены и сдвигового шифра замены. Данный класс включает в себя следующие методы:

- «simpleReplacement» – конструктор класса, с помощью которого для создаваемого объекта устанавливается словарь для шифрования и расшифрования;

- «encrypt» – метод, реализующий одиночное шифрование. На вход метода поступает строка открытого текста типа QString. Возвращаемым значением является строка шифротекста типа QString;

- «decrypt» – метод, реализующий расшифрование шифротеста методом одиночной перестановки. На вход метода поступает строка шифротекста типа QString;

- «setkey» – метод, с помощью которого для объекта класса SimpleReplacement устанавливается ключ шифрования. На вход подается словарь типа QMap<QString, QString>.

«RSAKeyDialog» – класс диалогового окна, предназначенного для формирования ключа для шифра RSA (класс «RSACrypt»). Данный класс включает в себя следующие методы:

- «RSAKeyDialog» – конструктор класса для создания объекта класса;
- «getPubKeyParams» – метод, предназначенный для ввода параметров открытого ключа;
- «getPrivKeyParams» – метод, предназначенный для ввода параметров секретного ключа;
- «isKeySet» – метод, предназначенный для статуса установки параметров ключей;
- «getPrimesList» – метод, возвращающий список простых чисел;
- «setPrimes» – метод, предназначенный для установки параметров P и Q алгоритма RSA.

Класс «RSACrypt» реализует алгоритм шифрования RSA. Данный класс включает в себя следующие методы:

- «RSACrypt» – конструктор, с помощью которого для создаваемого объекта устанавливаются алфавит;
- «encrypt» – метод, с помощью которого осуществляется зашифрование открытого текста, передаваемого в качестве параметра;
- «decrypt» – метод, с помощью которого осуществляется расшифрование закрытого текста, передаваемого в качестве параметра;
- «setKeyParams» – метод, предназначенный для ввода параметров ключей E, D, N алгоритма RSA.

Класс «GammaCypher» реализует метод шифрования путем гаммирования. Данный класс включает в себя следующие методы:

- «encrypt» – метод, с помощью которого осуществляется зашифрование открытого текста, передаваемого в качестве параметра;

- «decrypt» – метод, с помощью которого осуществляется расшифрование закрытого текста, передаваемого в качестве параметра;
- «setGamma» – метод, с помощью которого устанавливается гамма;
- «getGamma» – метод, с помощью которого возвращает установленную гамму;
- «GenNewGamma» – метод, с помощью которого генерируется новая гамма.

Класс «DoubleTransposition» реализует шифр двойной перестановки. Включает в себя следующие методы:

- «encrypt» – метод, реализующий одиночное шифрование. На вход метода поступает строка открытого текста типа Qstring. Возвращаемым значением является строка шифротекста типа Qstring;
- «decrypt» – метод, реализующий расшифрование шифротекста методом одиночной перестановки. На вход метода поступает строка шифротекста типа Qstring. Возвращаемым значением является строка открытого текста типа Qstring;
- «setkey» – метод, с помощью которого для объекта класса «SingleShift» устанавливаются ключи шифрования.

Класс «Gronspheld» реализует шифр Гронсфельда. Данный класс включает в себя следующие методы:

- «Gronspheld» – конструктор класса, устанавливающий для создаваемого объекта класса базовый алфавит;
- «encrypt» – метод, с помощью которого осуществляется зашифрование открытого текста, передаваемого в качестве параметра;
- «decrypt» – метод, с помощью которого осуществляется расшифрование закрытого текста, передаваемого в качестве параметра;
- «setKey» – метод, с помощью которого устанавливается ключ;

- «genNewKey» – метод, с помощью которого генерируется новая гамма.

- «SetAlpha» – метод, с помощью которого устанавливается базовый алфавит.

Класс «Cardano» реализует метод шифрования «решетка Кардано». Данный класс включает в себя следующие методы:

- «Cardano» – конструктор класса, устанавливающий для создаваемого объекта класса базовый алфавит;

- «setKey» – метод, с помощью которого устанавливается ключ шифрования;

- «encrypt» – метод, с помощью которого осуществляется зашифрование открытого текста, передаваемого в качестве параметра;

- «decrypt» – метод, с помощью которого осуществляется расшифрование закрытого текста, передаваемого в качестве параметра.

«CardanoKeyDialog» – класс диалогового окна, предназначенного для формирования ключа для шифра «решетки Кардано». Включает в себя метод getMatrix, предназначенный для получения ключа шифрования.

«mainwindow» – класс, который реализует интерфейс главного окна программного средства.

«bruteforce» – класс, который реализует метод полного перебора. Включает в себя методы:

- «bruteRSA» – метод, который подбирает простое число до 100, для алгоритма RSA;

- «bruteSimpleReplacement» – метод, который реализует полный перебор ключа для метода простой замены;

- «bruteDoubleTransposition» – метод, который реализует полный перебор ключа для метода двойной перестановки;

- «bruteGronsfeld» – метод, который реализует полный перебор ключа для метода полиалфавитной замены.

Взаимосвязь классов отображает диаграмма классов, которая представлена на рисунке 10.

Исходный код программного средства представлен в приложении А.

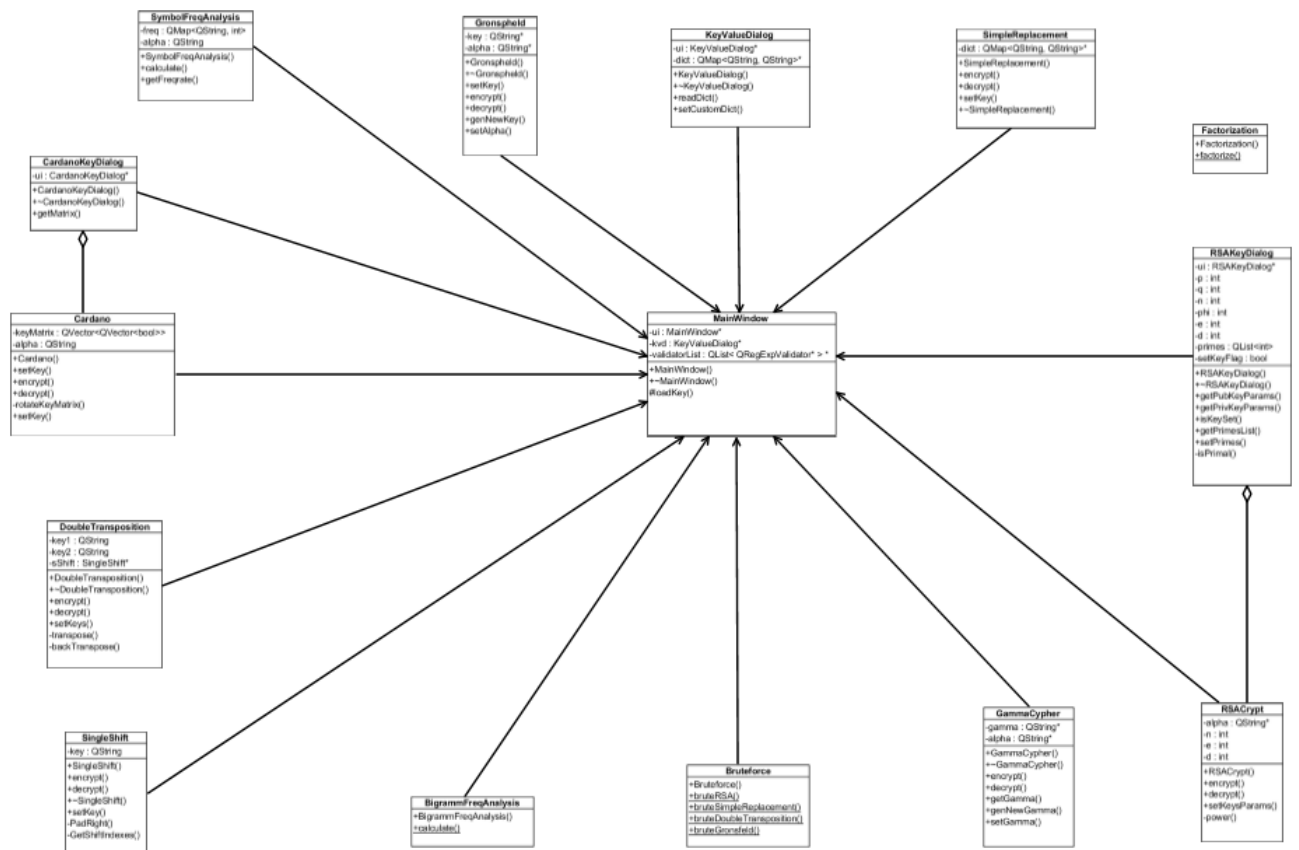


Рисунок 10 – Диаграмма классов

5. Руководство пользователя программного средства

5.1 Краткое описание возможностей

Программное средство предназначено для автоматизации криптоанализа простых шифров, а также программное средство предоставляет возможность для осуществления шифрования с помощью алгоритмов простых шифров.

Программное средство предоставляет возможность криптоанализа, шифрования и расшифрования следующих простых шифров:

- простой замены;
- сдвигового шифра замены;
- гаммирования;
- одиночной перестановки;
- двойной перестановки;
- шифра RSA;
- маршрутной перестановки;
- полиалфавитной замены.

В программном средстве реализованы следующие методы криптоанализа:

- метод полного перебора;
- метод частотного криптоанализа (подсчет частоты встречи букв и биграмм) в шифротексте;
- метод разложения числа на простые множители.

5. 2 Уровень подготовки пользователя

Пользователь программного средства должен иметь опыт работы с ОС Windows (XP/7/8/10), а также следующими навыками:

- знать принцип работы соответствующих простых шифров;
- знать и иметь навыки шифрования/расшифрования/криптоанализа простых шифров вручную.

5.3 Выполняемые функции и задачи

Данное программное средство выполняет функции и задачи, приведенные в таблице 22.

Таблица 22 – Функции и задачи программного средства

Функции	Задачи	Описание
Шифрование	Шифрование текста содержащего буквы русского и английского алфавита	В ходе выполнения данной задачи пользователю программного средства предоставляется возможность осуществить шифрование текста, содержащего буквы русского и английского алфавита длиной до 5000 символов, с помощью доступных алгоритмов шифрования и сохранить результат в текстовой файл
Расшифрование	Расшифрование данных для всех представленных простых шифров	В ходе выполнения данной задачи пользователю ПС предоставляется возможность осуществить расшифрование ранее зашифрованных текстов

Продолжение таблицы 22

Функции	Задачи	Описание
Криптоанализ	Криптоанализ текстов, зашифрованных с помощью доступных в программном алгоритмов	В ходе выполнения данной задачи пользователю программного средства предоставляется возможность осуществить криптоанализ доступных алгоритмов

5.4 Описание операций технологического процесса обработки исходных данных, необходимых для выполнения учебных задач криптоанализа.

5.4.1 Шифрование

5.4.1.1 Шифрование с помощью шифра простой замены и сдвигового шифра замены.

Для выполнения шифрования с помощью шифра простой замены и сдвигового шифра замены в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать» в соответствии с рисунком 11;
- в поле «Шифр» в выпадающем списке выбрать пункт «Простая замена» либо «Сдвиговый шифр замены» в соответствии с рисунком 12;
- выбрать необходимый язык текста для шифрования в выпадающем меню соответствии с рисунком 13;

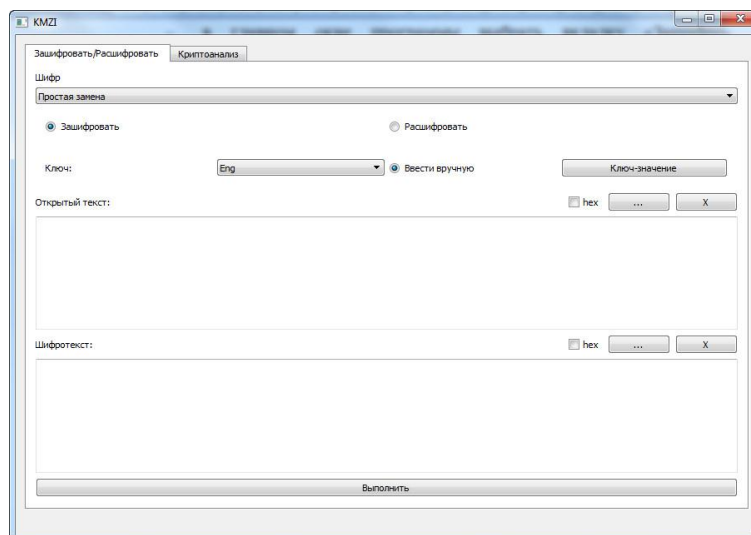


Рисунок 11 – Главное окно программы

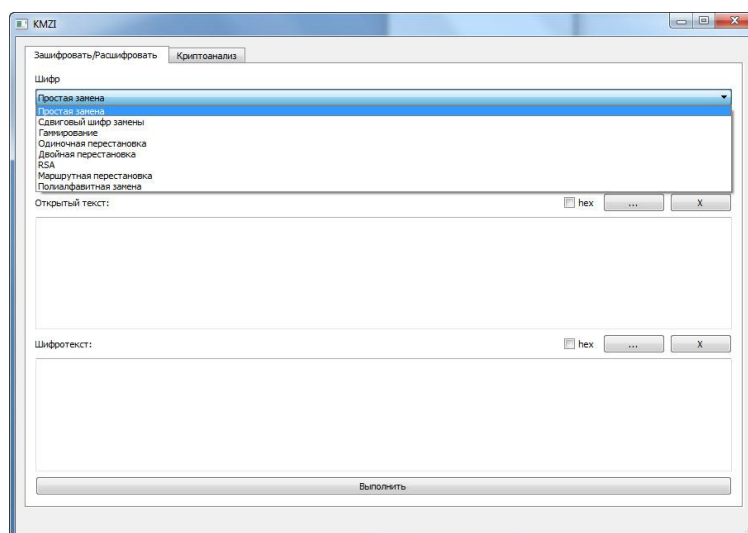


Рисунок 12 – Выбор пункта меню «Простая замена»

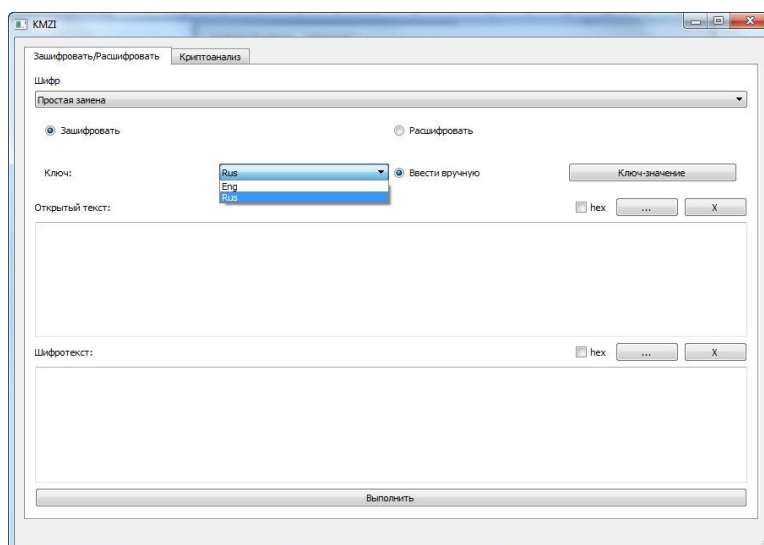


Рисунок 13 – Выбор языка текста

– открыть окно «Dialog» для задания параметра ключевого значения с помощью кнопки «Ключ-значение» в соответствии с рисунком 14;

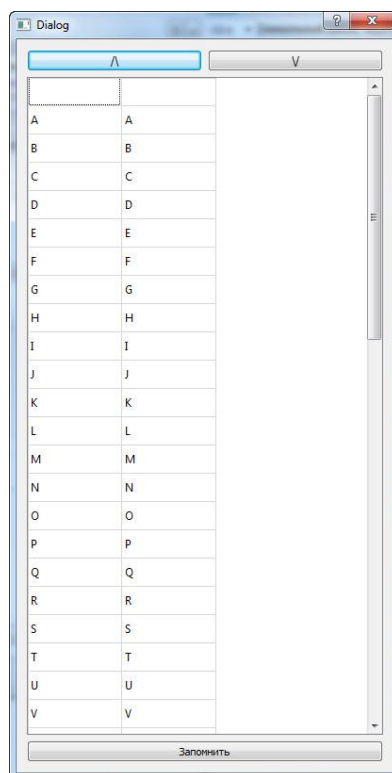


Рисунок 14 – Окно «Dialog»

– с помощью клавиш «Λ» и «V» задать необходимое смещение одного алфавита относительно другого и нажать кнопку «Запомнить» в соответствии с рисунком 15;

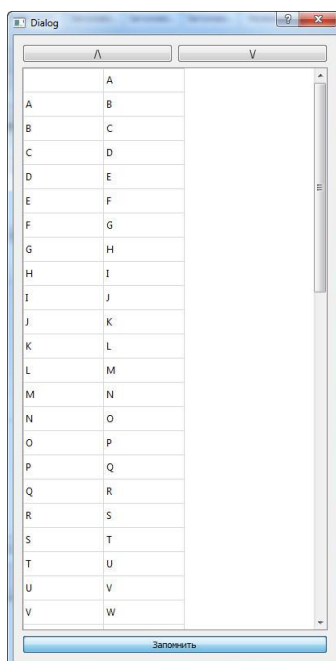


Рисунок 15 – Выбор смещения алфавита

- ввести текст, необходимый для шифрования, в поле «Открытый текст» заглавными буквами в соответствии с рисунком 16;

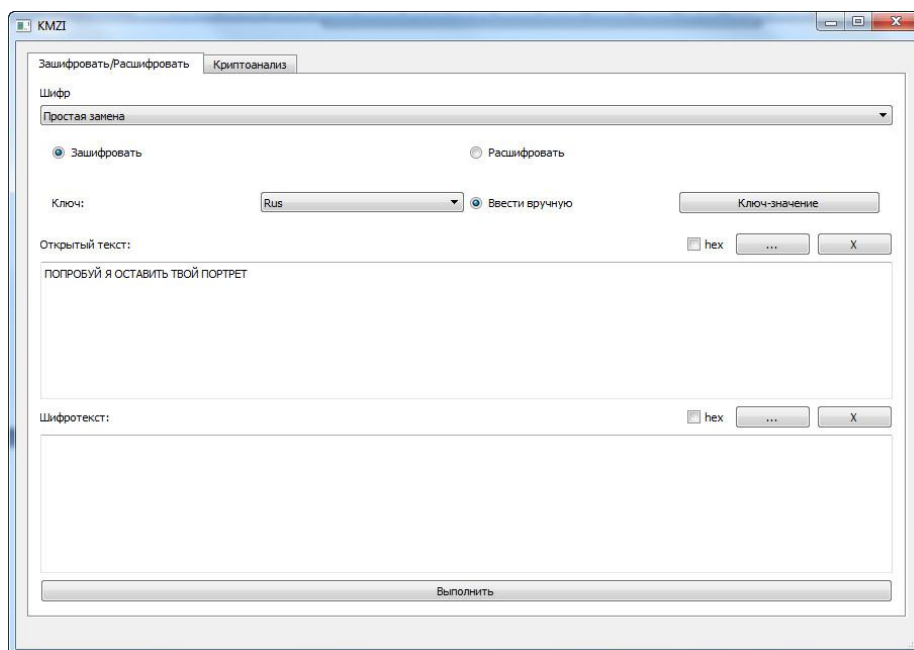


Рисунок 16 – Ввод текста

- для ввода текста из файла необходимо нажать клавишу «...» над полем «Открытый текст» в соответствии с рисунком 17;

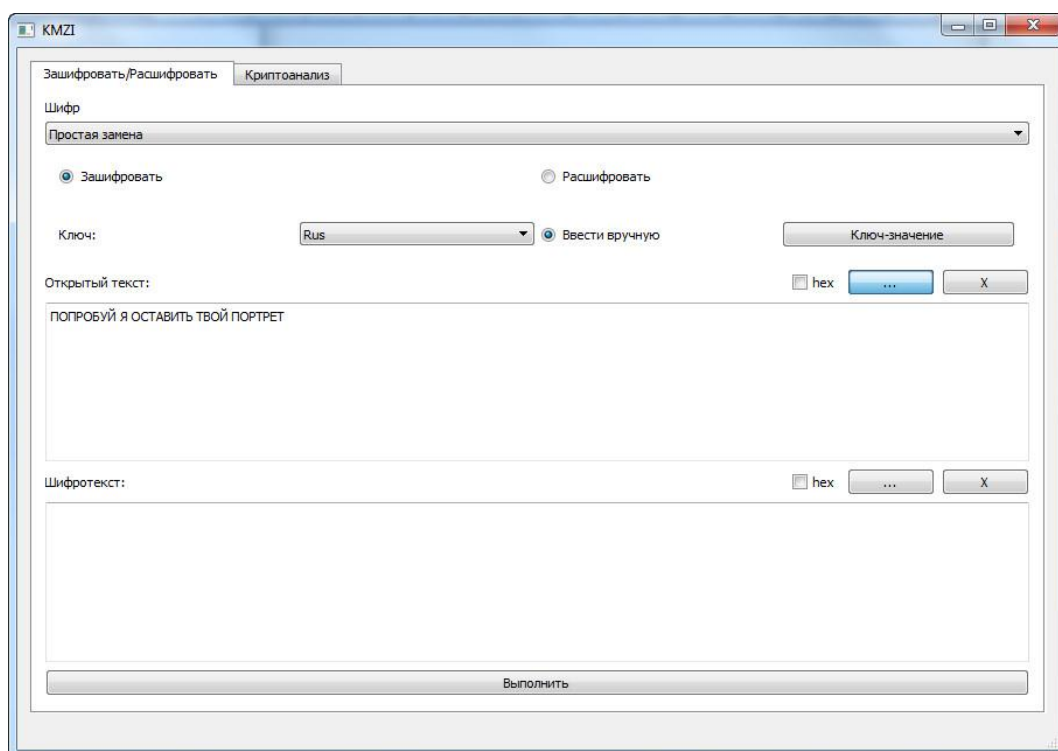


Рисунок 17 – Ввод текста из файла

- для очистки поля «Открытый текст» необходимо нажать кнопку «X» в соответствии с рисунком 18;

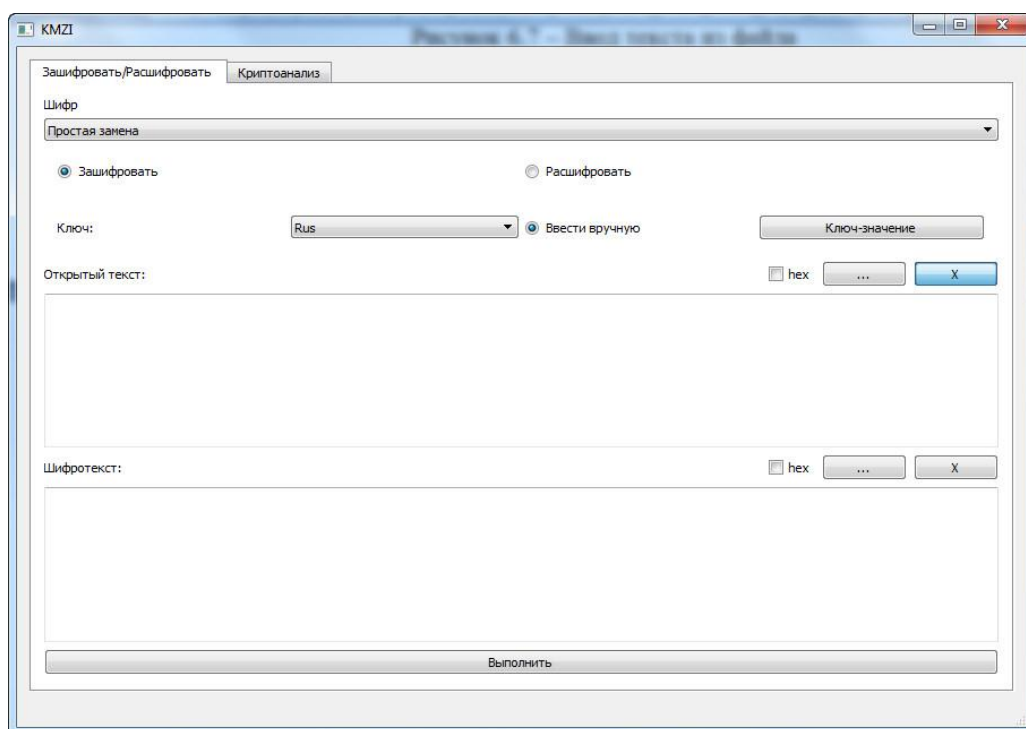


Рисунок 18 – Очистка поля «Открытый текст»

- для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex» в соответствии с рисунком 19;

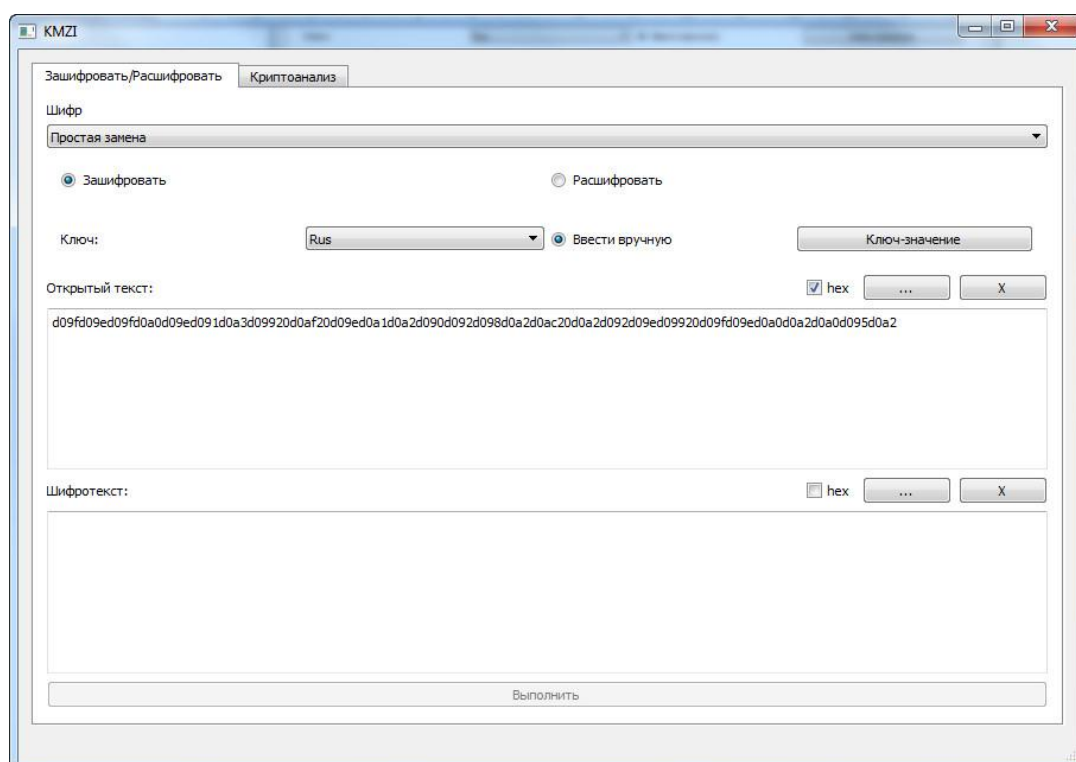


Рисунок 19 – Представление текста в hex-коде

- для выполнения шифрования необходимо нажать кнопку «Выполнить» в соответствии с рисунком 20;

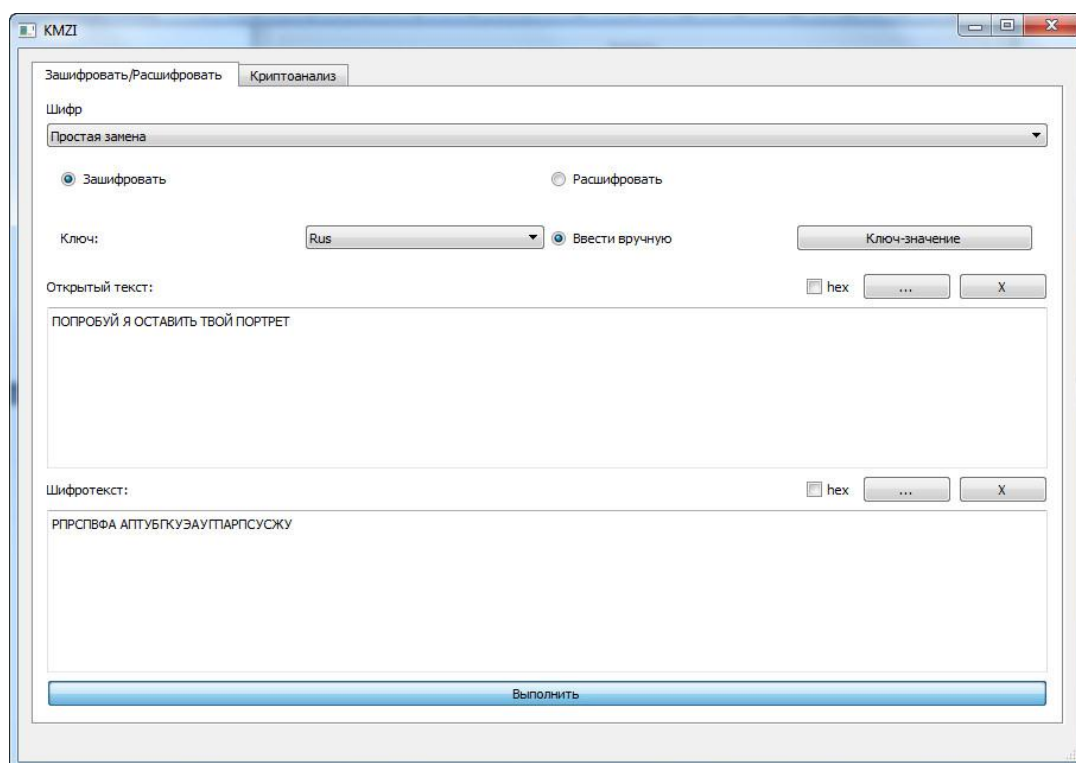


Рисунок 20 – Выполнение шифрования

– для сохранения получившегося шифротекста в текстовый файл необходимо нажать кнопку «...» в соответствии с рисунком 21;

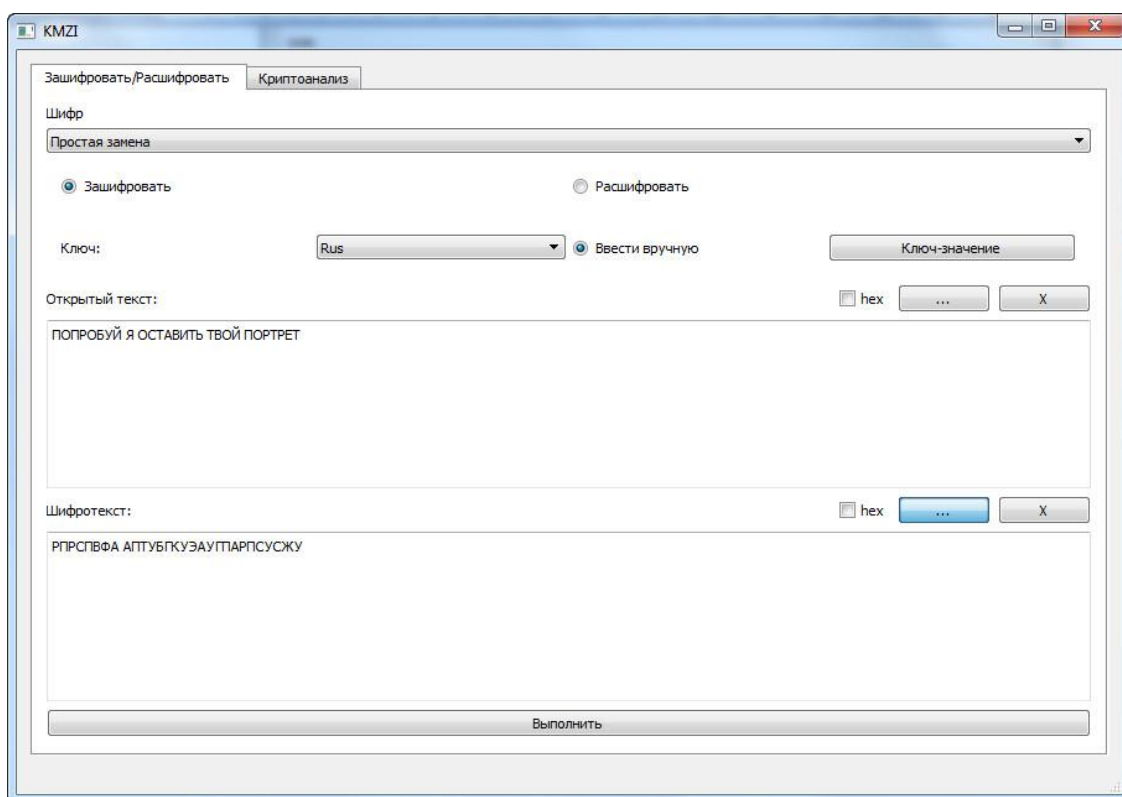


Рисунок 21 – Сохранение шифротекста в файл

- для очистки поля «Шифротекст» необходимо нажать кнопку «X» в соответствии с рисунком 22;

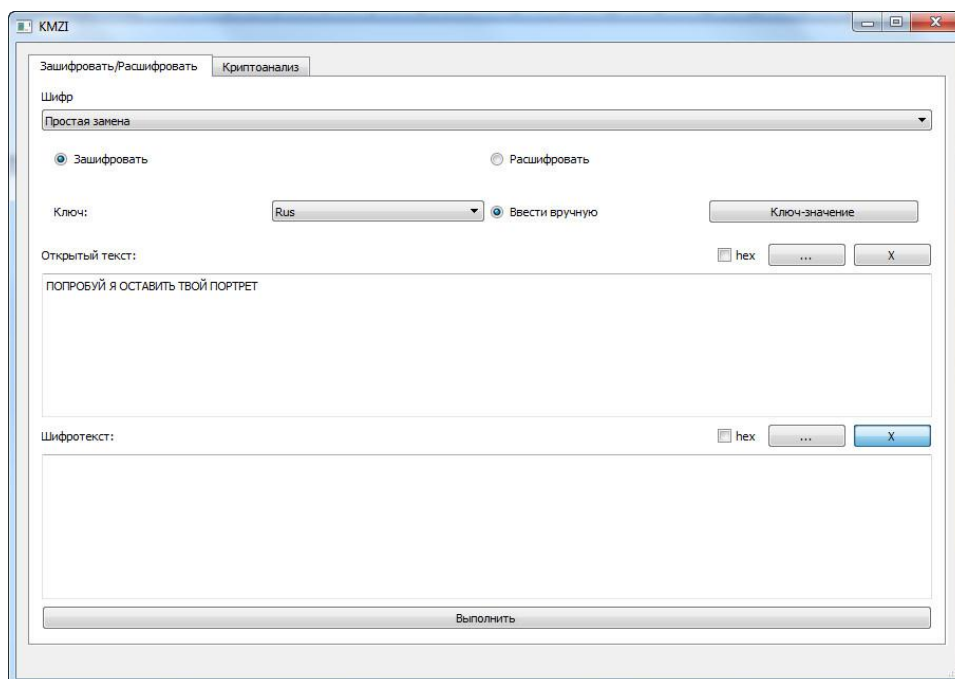


Рисунок 22 – Очистка поля «Шифротекст»

5.4.1.2 Шифрование с помощью шифра гаммирования.

Для осуществления шифрования с помощью шифра гаммирования в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать» в соответствии с рисунком 23;

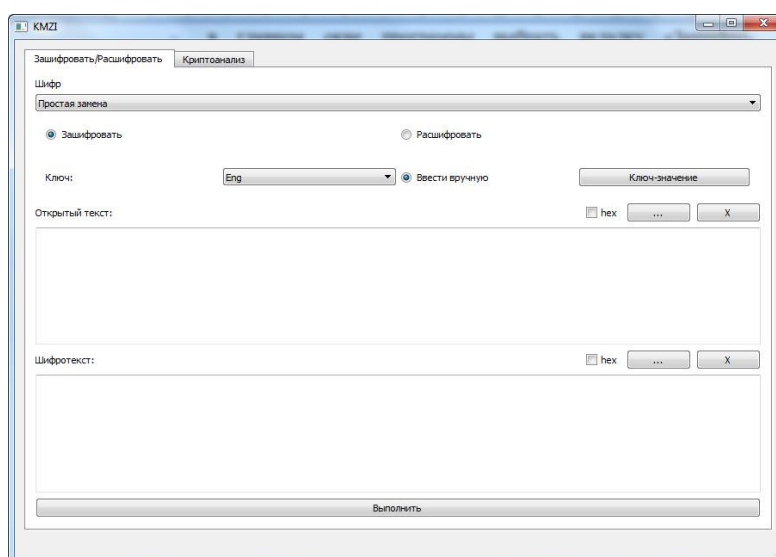


Рисунок 23 – Вкладка «Зашифровать/Расшифровать»

- в поле «Шифр» в выпадающем списке выбрать пункт «Гаммирование» в соответствии с рисунком 24;

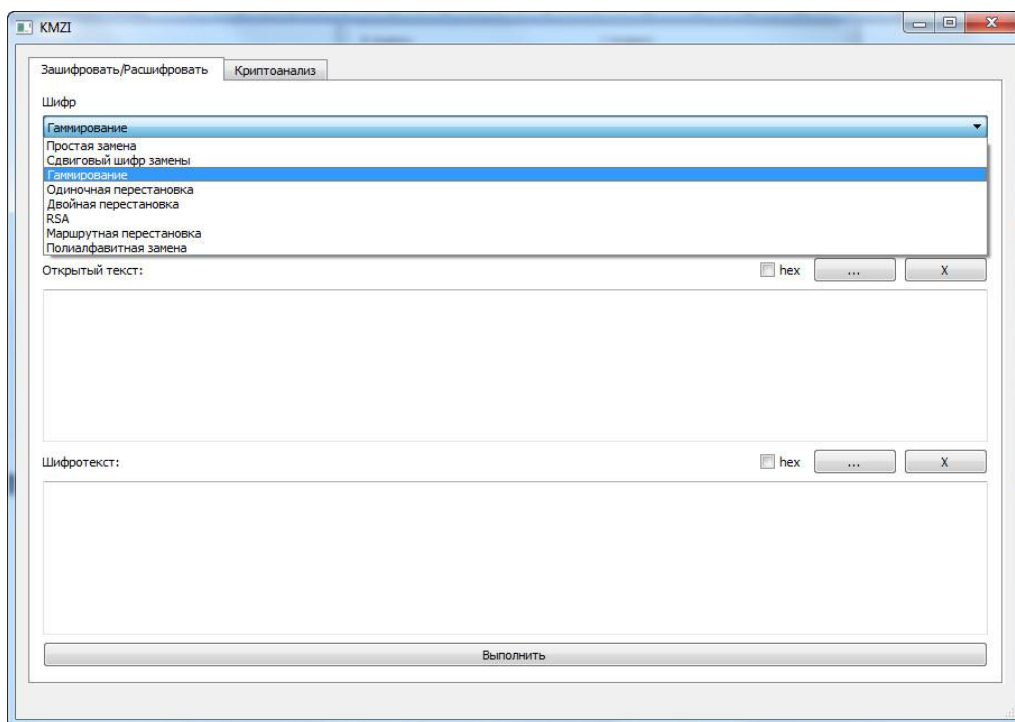


Рисунок 24 – Выбор шифра гаммирования

- для автоматической генерации ключевой последовательности необходимо выбрать радиокнопку «Сгенерировать» в соответствии с рисунком 25;

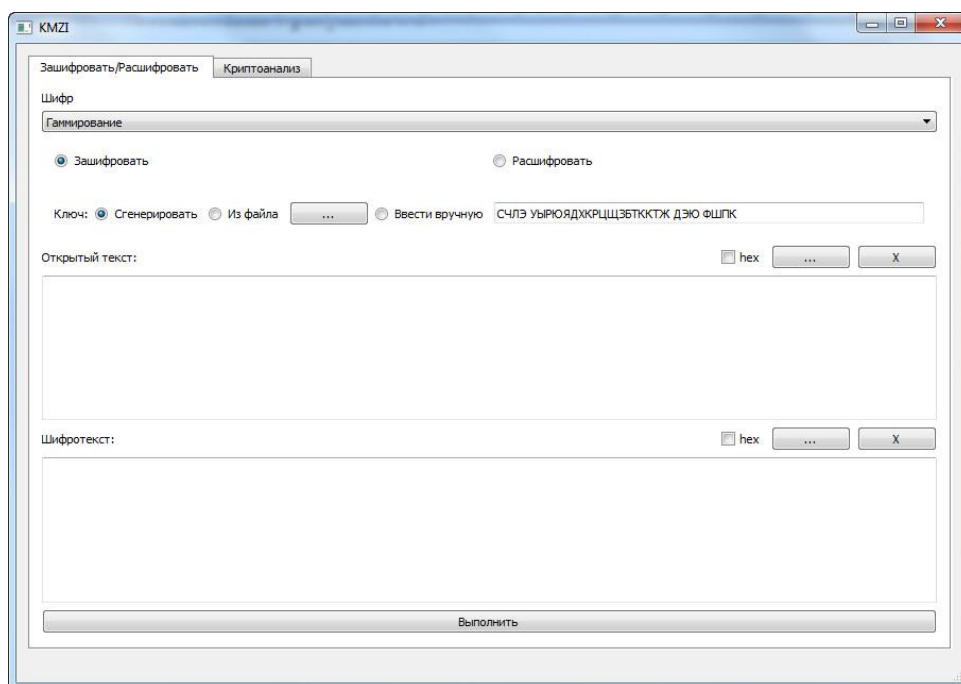


Рисунок 25 – Радиокнопка «Сгенерировать»

— для ввода ключевой последовательности из текстового файла необходимо выбрать радиокнопку «Из файла» и нажать кнопку «...» в соответствии с рисунком 26;

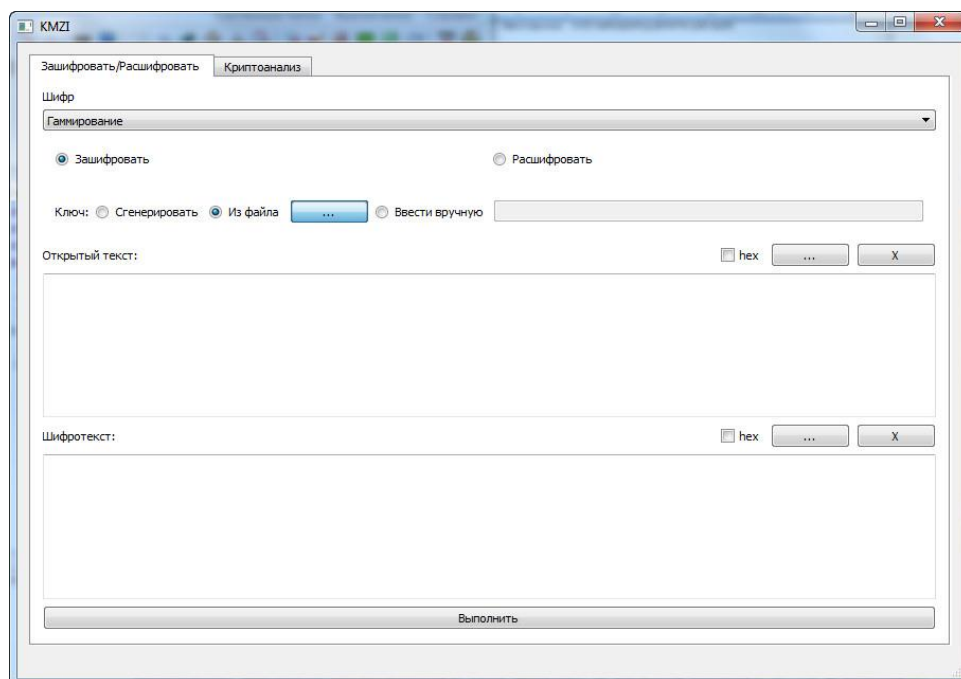


Рисунок 26 – Ввод ключа из файла

— для осуществления ввода ключевой последовательности вручную необходимо выбрать радиокнопку «Ввести вручную» и ввести заглавными только русскими буквами текст ключевой последовательности в соответствии с рисунком 27;

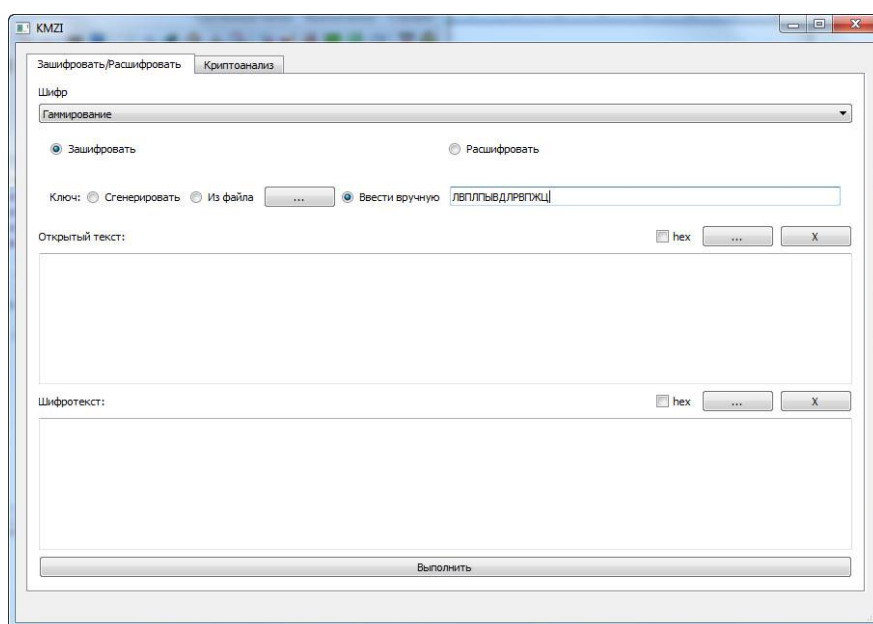


Рисунок 27 – Ввод ключа вручную

— ввод текста для шифрования необходимо осуществлять в поле «Открытый текст» только заглавными буквами русского алфавита в соответствии с рисунком 28;

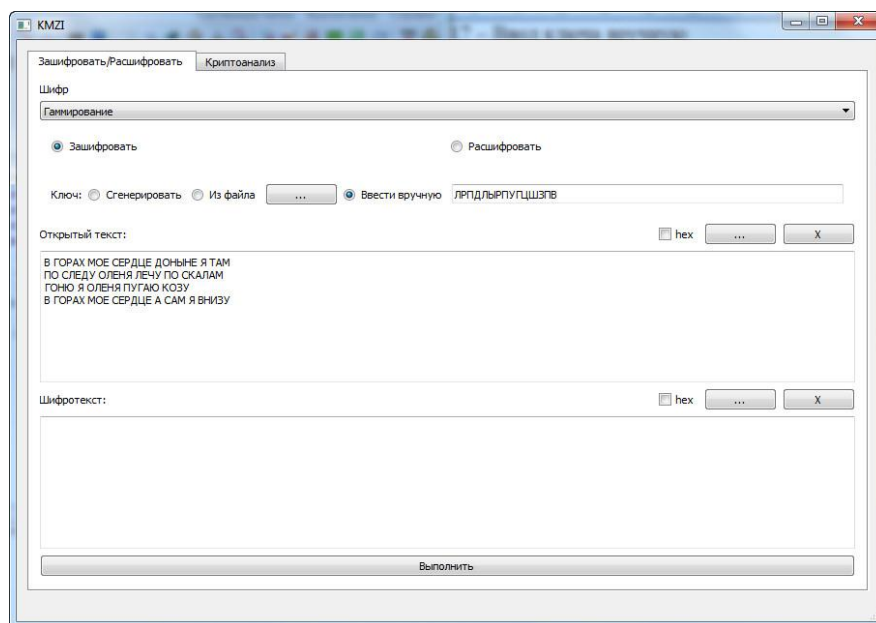


Рисунок 28 – Ввод открытого текста

— для осуществления ввода открытого текста из файла необходимо нажать кнопку «...» в соответствии с рисунком 29;

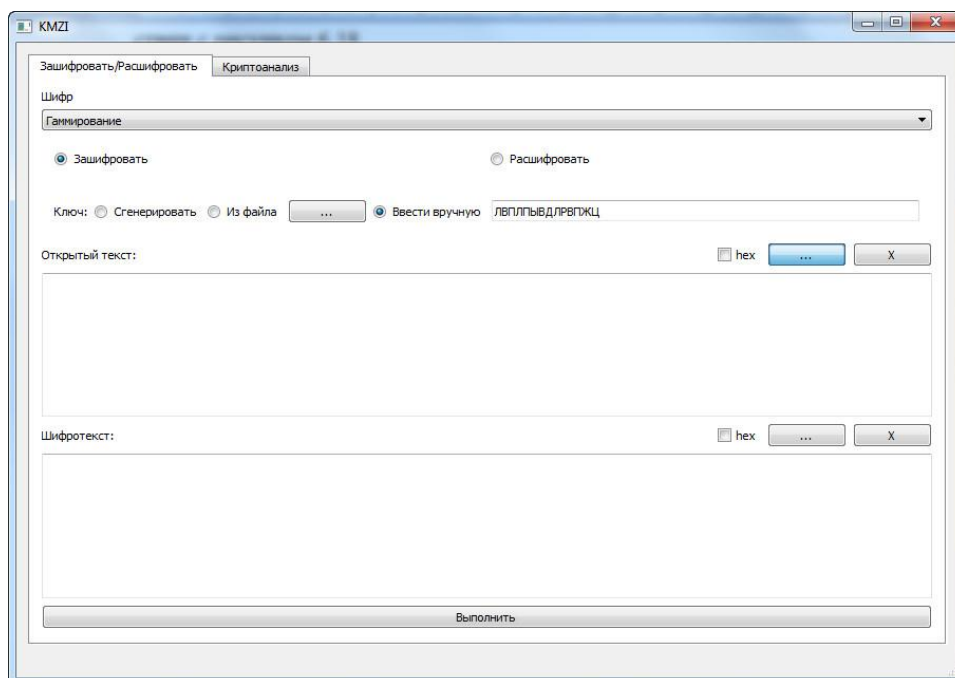


Рисунок 29 – Ввод текста из файла

— для очистки поля «Открытый текст» необходимо нажать кнопку «X» в соответствии с рисунком 30;

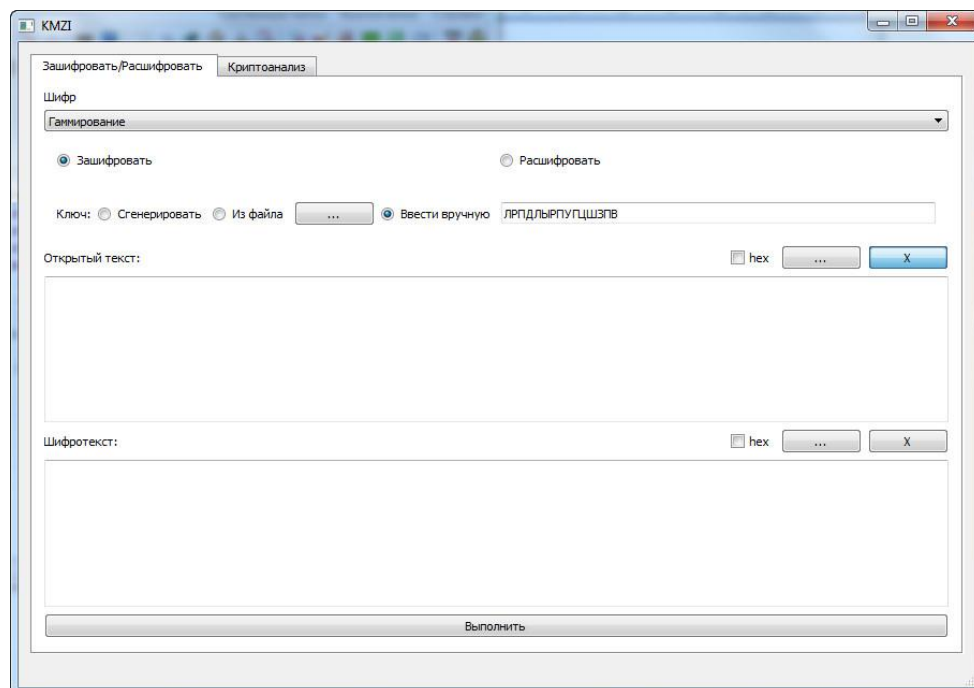


Рисунок 30 – Очистка поля «Открытый текст»

— для выполнения шифрования необходимо нажать кнопку «Выполнить» в соответствии с рисунком 31;

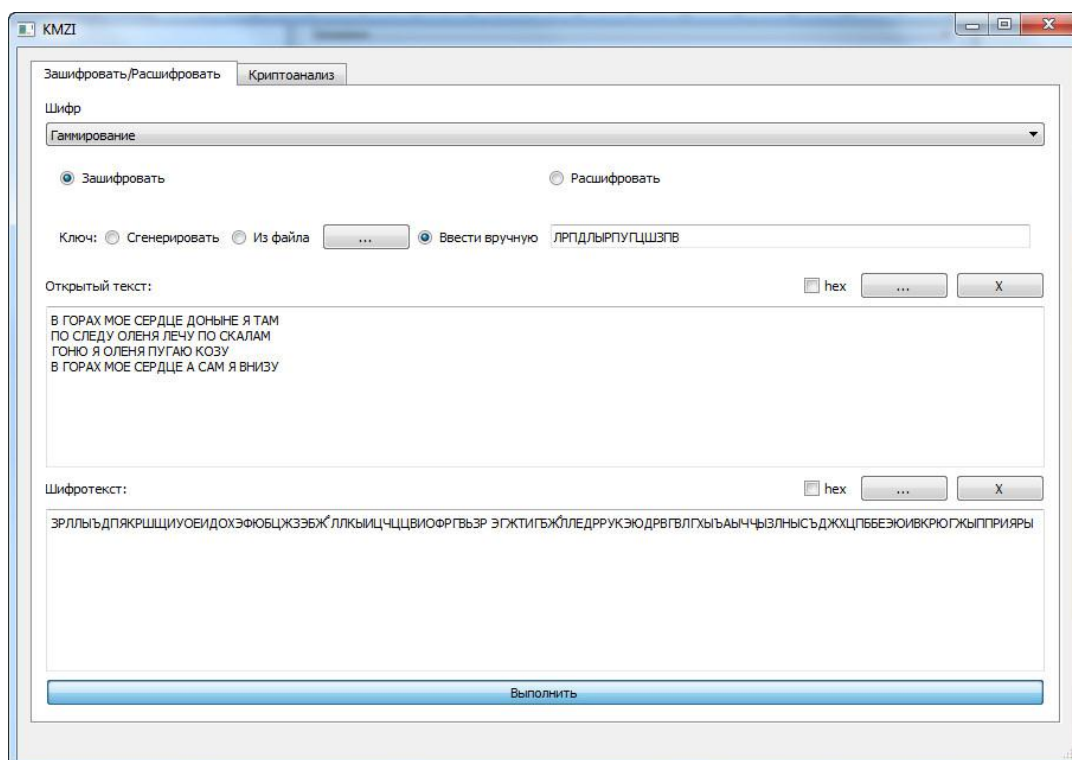


Рисунок 31 – Выполнение шифрования

— для того, чтобы сохранить шифротекст в текстовый файл, необходимо нажать кнопку «...» в соответствии с рисунком 32;

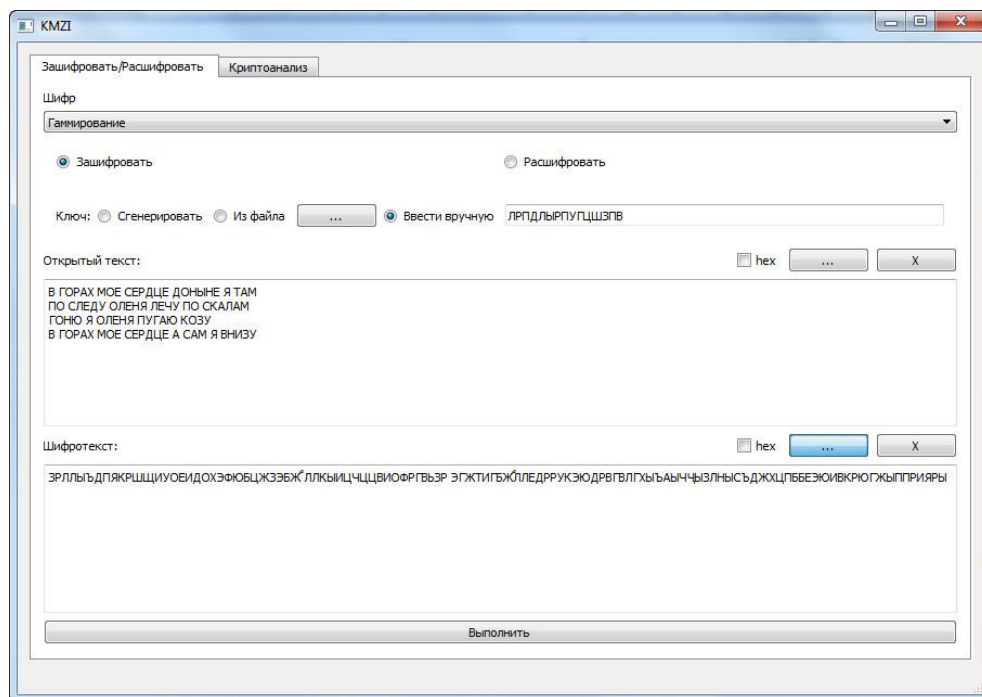


Рисунок 32 – Сохранение шифротекста в файл

– для очистки поля «Шифротекст» необходимо нажать кнопку «X» в соответствии с рисунком 33;

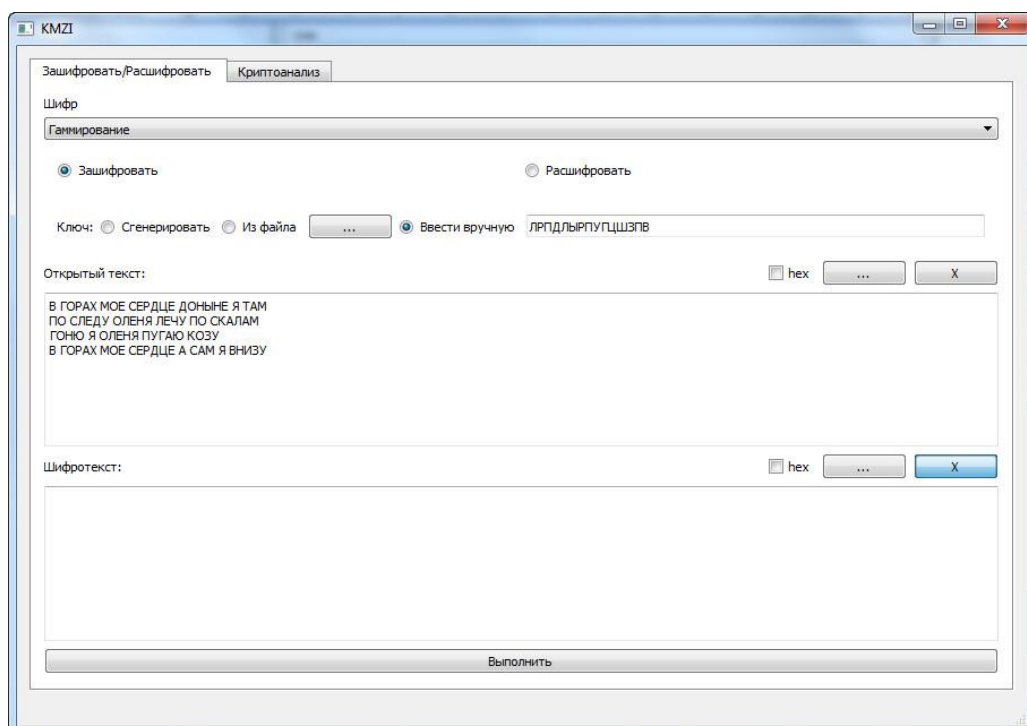


Рисунок 33 – Очистка поля «Шифротекст»

5.4.1.3 Шифрование с помощью шифра одиночной перестановки.

Для осуществления шифрования с помощью шифра одиночной перестановки в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать». Данное действие выполняется аналогично предыдущему подразделу;

- в поле «Шифр» в выпадающем списке выбрать пункт «Одиночная перестановка». Данное действие выполняется аналогично предыдущему подразделу;

- для задания ключевой последовательности в поле «Ключ» необходимо ввести заглавные буквы русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

- открытый текст необходимо вводить в поле «Открытый текст» заглавными буквами русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления ввода открытого текста, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для очистки поля «Открытый текст» необходимо нажать кнопку «X». Данное действие выполняется аналогично предыдущему подразделу;

- для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;

- для выполнения шифрования необходимо нажать кнопку «Выполнить». Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления сохранения шифротекста в текстовый файл, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для представления шифротекста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;

- для очистки поля «Шифротекст» необходимо нажать кнопку «X» данное действие выполняется аналогично предыдущему подразделу.

5.4.1.4 Шифрование с помощью шифра двойной перестановки.

Для осуществления шифрования с помощью шифра двойной перестановки в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать». Данное действие выполняется аналогично предыдущему подразделу;

- в поле «Шифр» в выпадающем списке выбрать пункт «Двойная перестановка». Данное действие выполняется аналогично предыдущему подразделу;

- для задания ключевой последовательности в поле «Ключ» необходимо ввести заглавные буквы русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

- открытый текст необходимо вводить в поле «Открытый текст» заглавными буквами русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления ввода открытого текста, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для очистки поля «Открытый текст» необходимо нажать кнопку «X». Данное действие выполняется аналогично предыдущему подразделу;

- для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;

- для выполнения шифрования необходимо нажать кнопку «Выполнить». Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления сохранения шифротекста в текстовый файл, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для представления шифротекста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;

- для очистки поля «Шифротекст» необходимо нажать кнопку «X» данное действие выполняется аналогично предыдущему подразделу.

5.4.1.5 Шифрование с помощью шифра RSA.

Для осуществления шифрования с помощью шифра RSA в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать». Данное действие выполняется аналогично предыдущему подразделу;

- в поле «Шифр» в выпадающем списке выбрать пункт «RSA». Данное действие выполняется аналогично предыдущему подразделу;

- для автоматической генерации ключевой последовательности необходимо выбрать радиокнопку «Сгенерировать» в соответствии с рисунком 34;

- для ввода ключа вручную необходимо нажать кнопку «RSA-ключ» согласно рисунку 35;

- для расчета пары ключей необходимо в полях «p» и «q» диалогового окна «Dialog» ввести целые числа до 100 b и нажать кнопку «Рассчитать» в соответствии с рисунком 36;

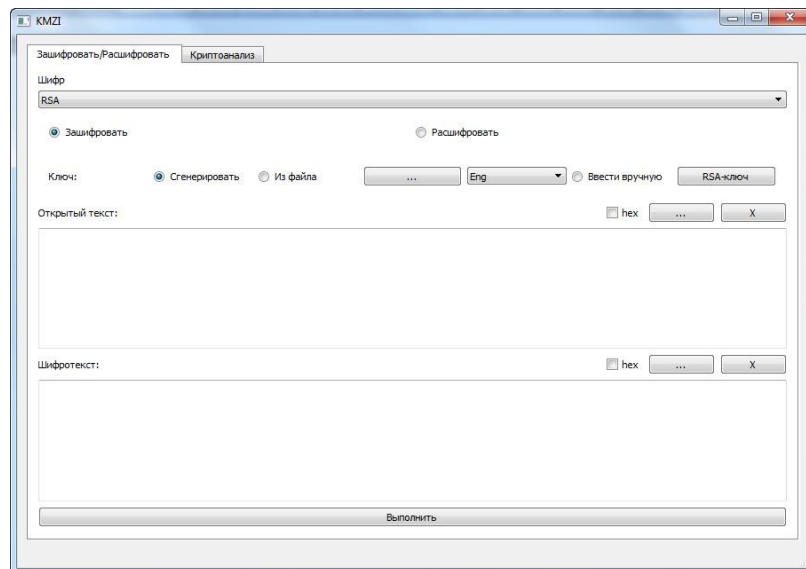


Рисунок 34 – Генерация ключа RSA

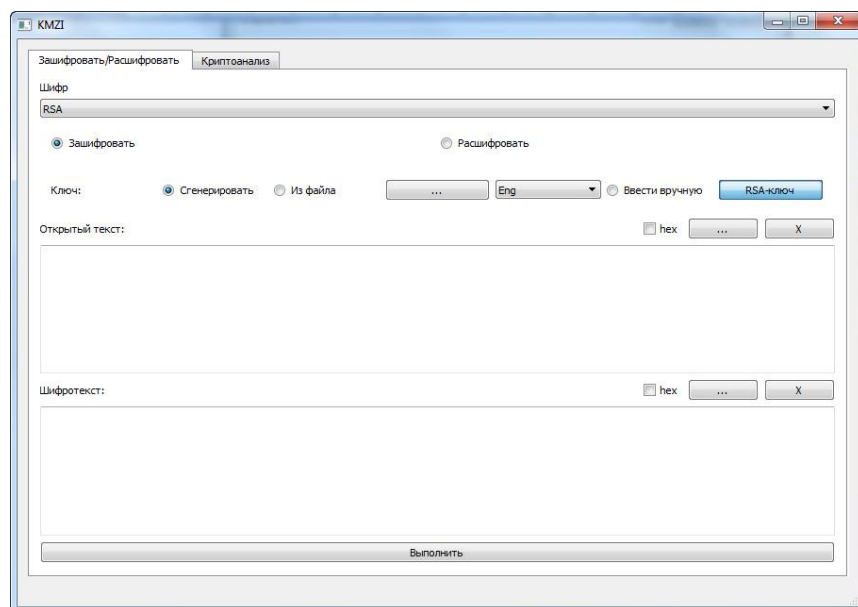


Рисунок 35 – Ручной ввод ключа RSA

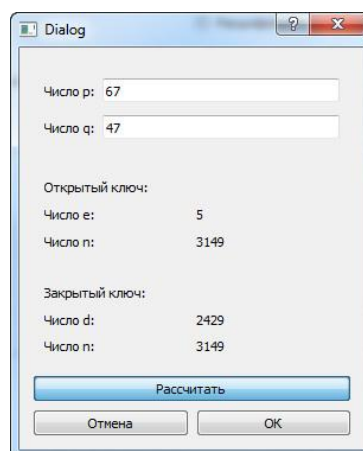


Рисунок 36 – Расчет пары ключей

- открытый текст необходимо вводить в поле «Открытый текст» заглавными буквами русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления ввода открытого текста, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для очистки поля «Открытый текст» необходимо нажать кнопку «X». Данное действие выполняется аналогично предыдущему подразделу;

- для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему подразделу;

- для выполнения шифрования необходимо нажать кнопку «Выполнить». Данное действие выполняется аналогично предыдущему подразделу;

- для осуществления сохранения шифротекста в текстовый файл, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

- для представления шифротекста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему подразделу;

- для очистки поля «Шифротекст» необходимо нажать кнопку «X» данное действие выполняется аналогично предыдущему подразделу.

5.4.1.6 Шифрование с помощью шифра маршрутной перестановки.

Для осуществления шифрования с помощью шифра маршрутной перестановки в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать». Данное действие выполняется аналогично предыдущему подразделу;

– в поле «Шифр» в выпадающем списке выбрать пункт «Маршрутная перестановка». Данное действие выполняется аналогично предыдущему подразделу;

– для задания ключевой последовательности необходимо нажать кнопку «Кардано-ключ» для открытия диалогового окна «Dialog». Затем в появившейся матрице отметить флагами прорези в трафаретах согласно рисунку 37.

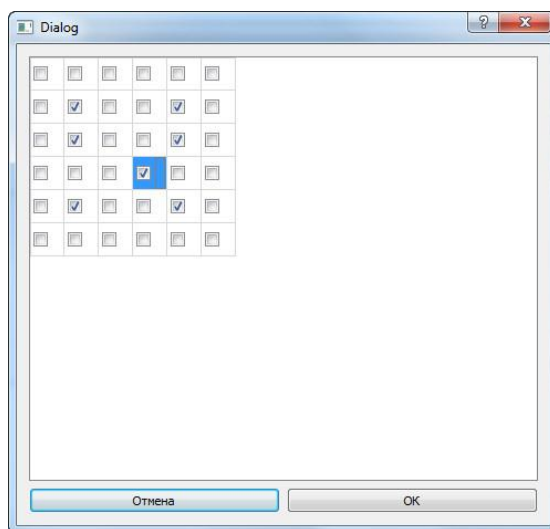


Рисунок 37 – Прорези в трафарете

– открытый текст необходимо вводить в поле «Открытый текст» заглавными буквами русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;

– для осуществления ввода открытого текста, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;

– для очистки поля «Открытый текст» необходимо нажать кнопку «X». Данное действие выполняется аналогично предыдущему подразделу;

– для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;

- для выполнения шифрования необходимо нажать кнопку «Выполнить». Данное действие выполняется аналогично предыдущему подразделу;
- для осуществления сохранения шифротекста в текстовый файл, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;
- для представления шифротекста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;
- для очистки поля «Шифротекст» необходимо нажать кнопку «X» данное действие выполняется аналогично предыдущему подразделу.

5.4.1.7 Шифрование с помощью шифра полиалфавитной замены.

Для осуществления шифрования с помощью шифра полиалфавитной замены в программном средстве необходимо выполнить следующие действия:

- в главном окне программы выбрать вкладку «Зашифровать/Расшифровать». Данное действие выполняется аналогично предыдущему подразделу;
- в поле «Шифр» в выпадающем списке выбрать пункт «Полиалфавитная замена». Данное действие выполняется аналогично предыдущему подразделу;
- открытый текст необходимо вводить в поле «Открытый текст» заглавными буквами русского алфавита. Данное действие выполняется аналогично предыдущему подразделу;
- для осуществления ввода открытого текста, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;
- для очистки поля «Открытый текст» необходимо нажать кнопку «X». Данное действие выполняется аналогично предыдущему подразделу;

- для представления текста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему подразделу;
- для выполнения шифрования необходимо нажать кнопку «Выполнить». Данное действие выполняется аналогично предыдущему подразделу;
- для осуществления сохранения шифротекста в текстовый файл, необходимо нажать кнопку «...». Данное действие выполняется аналогично предыдущему подразделу;
- для представления шифротекста как последовательности байтов в hex-коде необходимо установить флажок «hex». Данное действие выполняется аналогично предыдущему разделу;
- для очистки поля «Шифротекст» необходимо нажать кнопку «X» данное действие выполняется аналогично предыдущему подразделу.

5.4.2 Расшифрование

Для осуществления операции расшифрования в программном средстве необходимо выполнить следующие действия:

- во вкладке «Шифрование/Расшифрование» для любого шифра выбрать радиокнопку «Расшифровать» согласно рисунку 38;

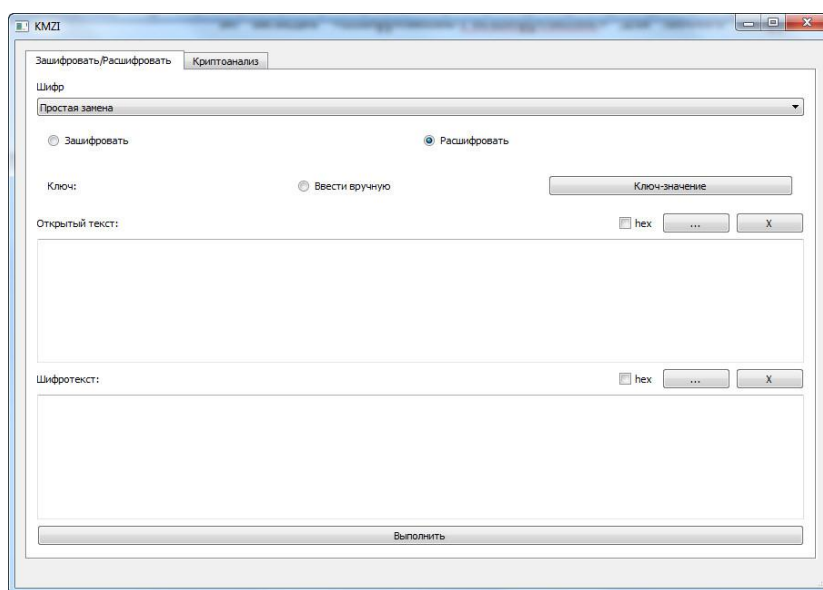


Рисунок 38 – Выбор процедуры расшифрования

– ввести шифротекст в поле «Шифротекст» и нажать кнопку «Выполнить».

5.4.3 Криптоанализ

5.4.3.1 Метод полного перебора.

Для осуществления операции криптоанализа в программном средстве необходимо выполнить следующие действия:

– в основном окне программы выбрать вкладку «Криптоанализ» согласно рисунку 39;

– затем нужно выбрать один из доступных шифров для криптоанализа согласно рисунку 40;

– после чего необходимо ввести с клавиатуры текст в поле «Шифротекст» и правой кнопкой мыши нажать кнопку «Перебор» согласно рисунку 41;

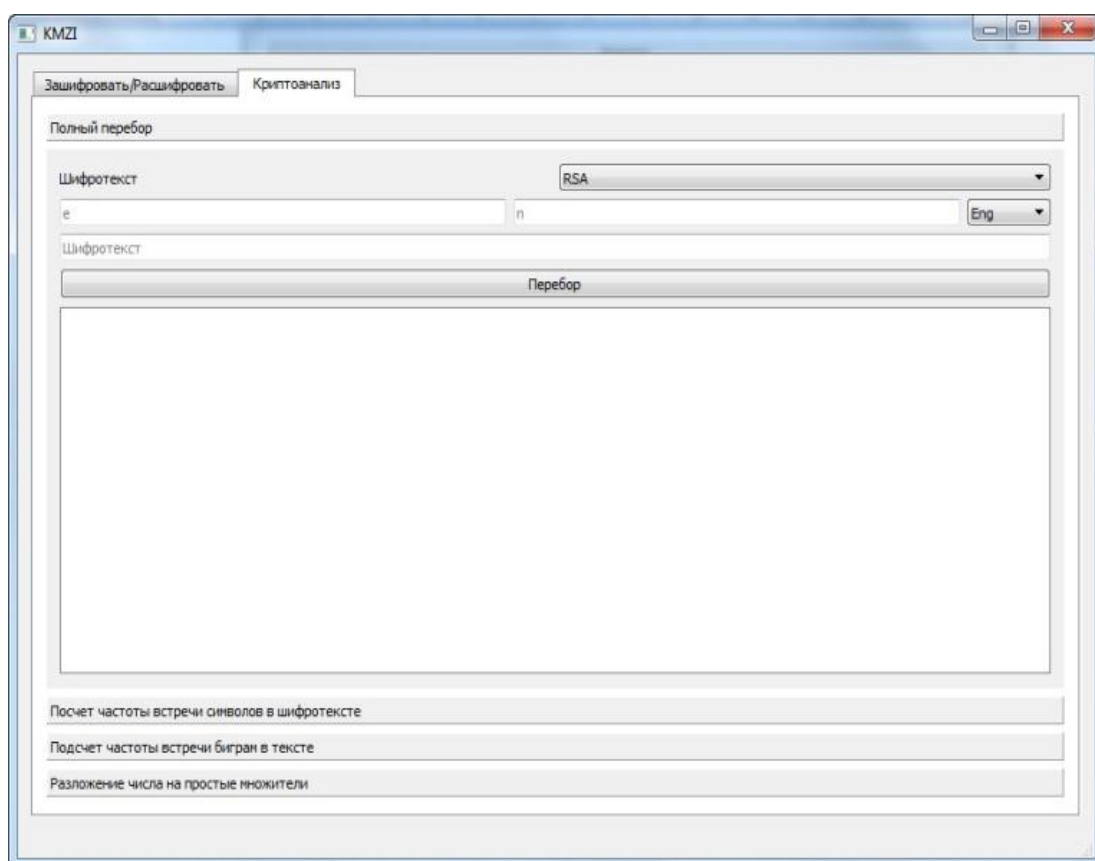


Рисунок 39 – Вкладка «Криптоанализ»

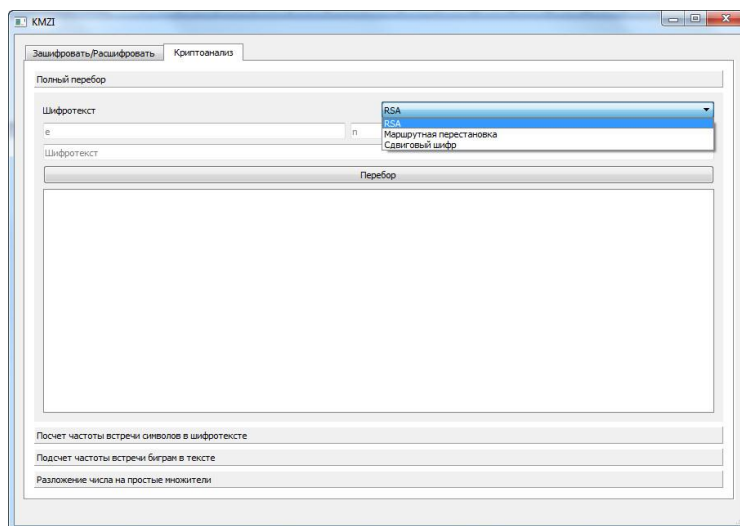


Рисунок 40 – Выбор шифра для криптоанализа

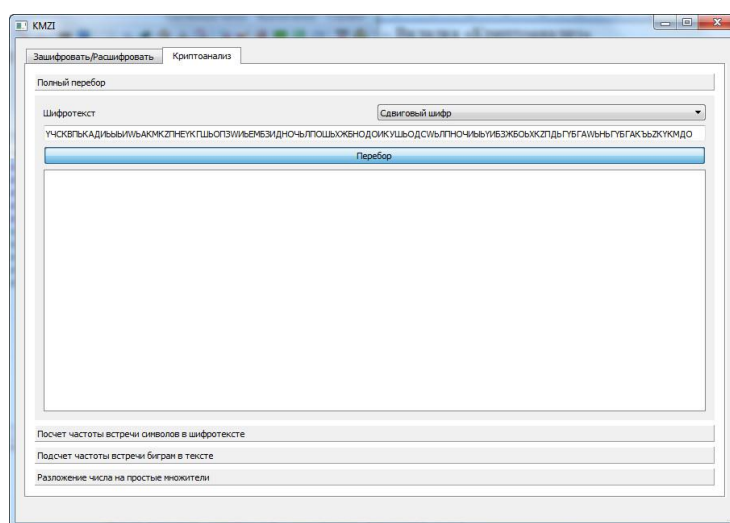


Рисунок 41 – Кнопка «Переворот»

– после того, как закончится перебор, среди всех вариантов следует выбрать подходящий, осмысленный вариант согласно рисунку 42.

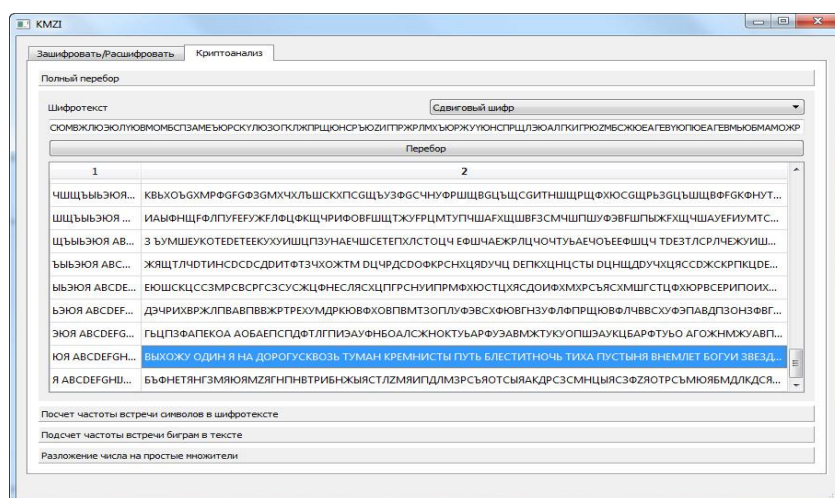


Рисунок 42 – Выбор подходящего варианта текста

5.4.3.2 Подсчет частоты встречи символов в шифротексте.

Для осуществления данной операции в программном средстве необходимо выполнить следующее действие – для осуществления подсчета частоты встречи символов в тексте необходимо в поле «Шифротекст» ввести шифротекст, который был получен при шифровании методом простой замены и нажать кнопку «Подсчитать» согласно рисунку 43;

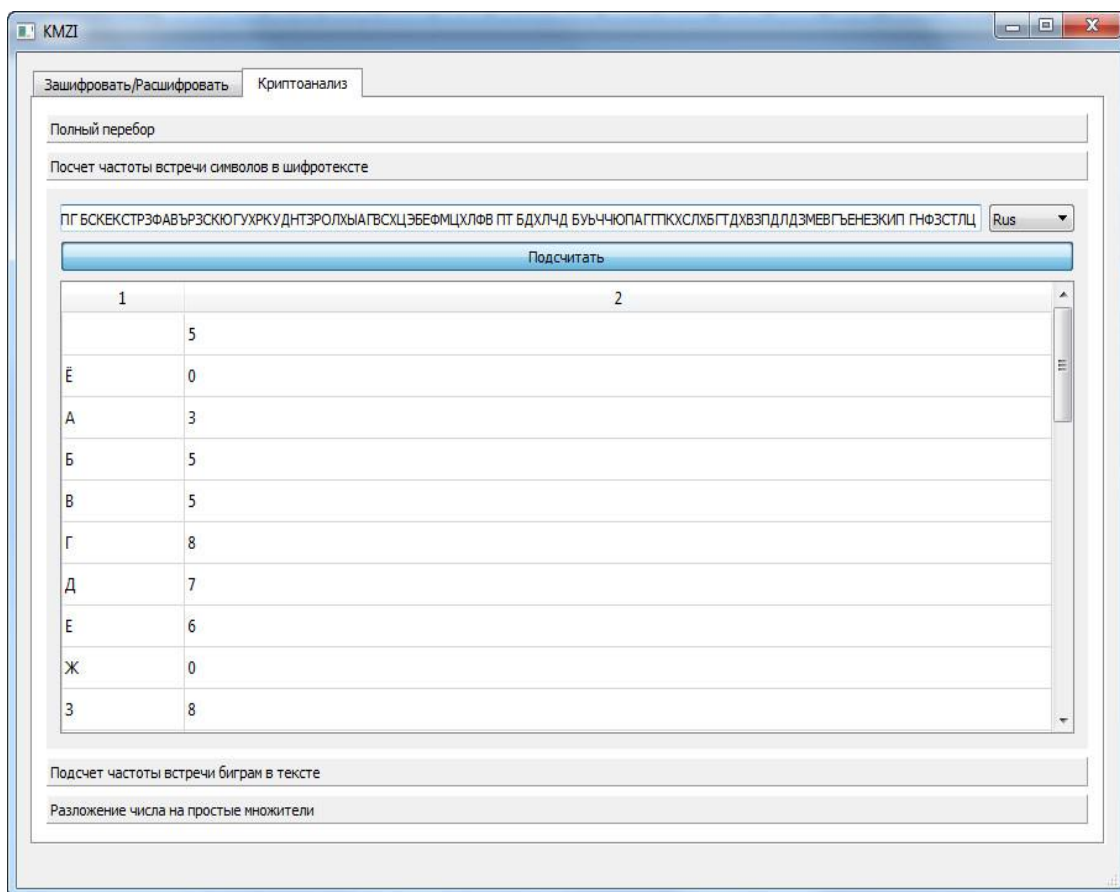


Рисунок 43 – Подсчет частоты встречи символов

5.4.3.3 Подсчет частоты встречи биграмм в шифротексте.

Для осуществления данной операции в программном средстве необходимо выполнить следующее действие – для осуществления подсчета частоты встречи биграмм в тексте необходимо в поле «Шифротекст» ввести необходимый шифротекст, который был получен при шифровании методом простой замены и правой кнопкой мыши нажать кнопку «Подсчитать» согласно рисунку 44;

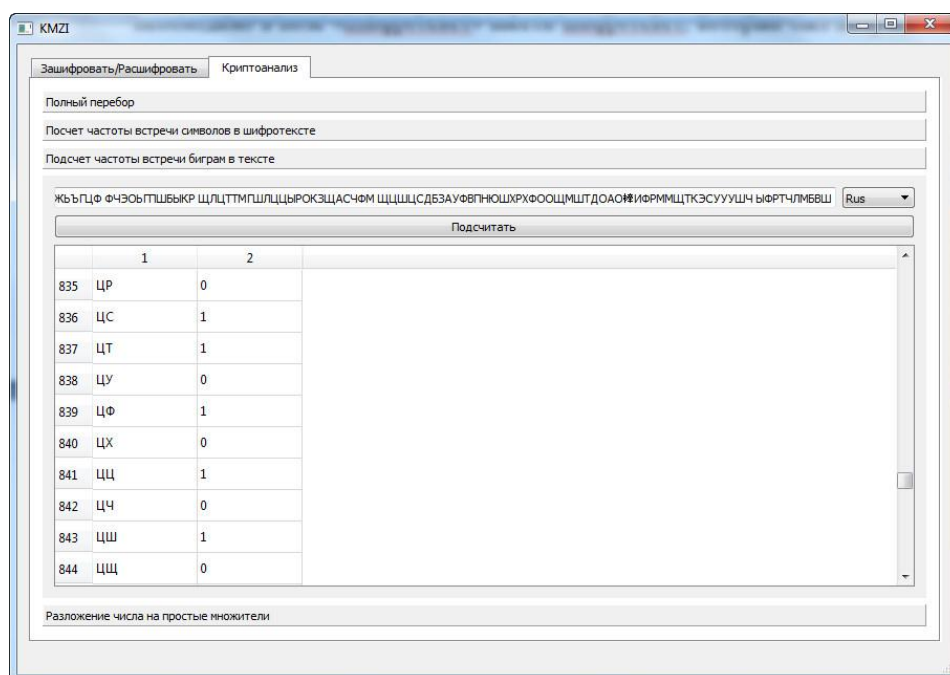


Рисунок 44 – Подсчет частоты встречи биграмм

5.4.3.4 Разложение числа на простые множители.

Для осуществления данной операции в программном средстве необходимо выполнить следующие действия: ввести число в поле «Число» и нажать кнопку «Вычислить» в соответствии с рисунком 45.

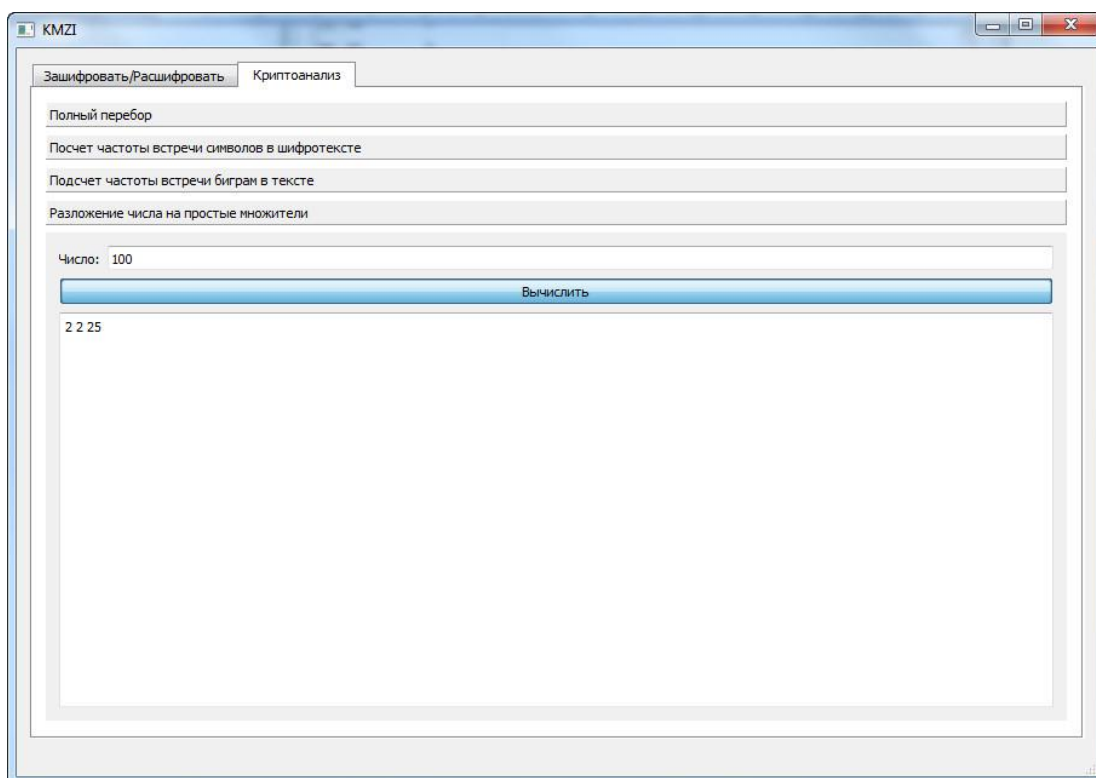


Рисунок 45 – Разложение числа на простые множители

ЗАКЛЮЧЕНИЕ

В процессе выполнения дипломного проекта были решены следующие задачи:

- произведен обзор существующих простых шифров;
- произведен анализ способов криптоанализа простых шифров;
- разработана архитектура программного средства;
- разработано руководство пользователя программного средства.

В результате разработано программное средство для криптоанализа простых шифров.

В разработанной программе имеются следующие возможности:

- графический интерфейс;
- ввод/вывод открытого текста и шифротекста в текстовый файл;
- шифрование и расшифрование текста русского и английского алфавитов длиной до 5000 символов с помощью простых шифров;
- выполнение подсчета частоты встречи символов и биграмм в шифротексте;
- разложение числа на простые множители.

Полученное, в результате выполнения дипломного проекта, программное средство будет положено в основу методических указаний для студентов 3 курса специальности 10.05.03 «Информационная безопасность автоматизированных систем».

Также, данное программное средство рекомендовано для ознакомления абитуриентам данной специальности.

Таким образом, цель ВКР достигнута, требования технического задания на выпускную квалификационную работу выполнены в полном объеме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Простейшие криптографические шифры. [Электронный ресурс]. – Режим доступа: http://life-prog.ru/2_86186_prosteyshie-kriptograficheskie-shifri.html, свободный. – Загл. с экрана.
- 2 Простые криптосистемы. [Электронный ресурс]. – Режим доступа: <http://protect.htmlweb.ru/p11.htm#OP>, свободный. – Загл. с экрана.
- 3 Шифр простой замены. [Электронный ресурс]. – Режим доступа: http://kmb.ufoctf.ru/crypto/frequency_analysis_substitution/main.html, свободный. – Загл. с экрана.
- 4 Шифр Цезаря. [Электронный ресурс]. – Режим доступа: <https://sites.google.com/site/kriptografics/sifr-cezara>, свободный. – Загл. с экрана.
- 5 Симметричное шифрование (гаммирование). [Электронный ресурс]. – Режим доступа: <http://technomag.bmstu.ru/doc/187185.html>, свободный. – Загл. с экрана.
- 6 Шифр перестановки. Шифр гаммирования. [Электронный ресурс]. – Режим доступа: <http://www.unn.ru/bibn/files/materials/kripto/kr2.pdf>, свободный. – Загл. с экрана.
- 7 Шифры перестановки. [Электронный ресурс]. – Режим доступа: <https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema5>, свободный. – Загл. с экрана.
- 8 Шифрование текстов. Основные принципы. Метод Гронсфельда. [Электронный ресурс]. – Режим доступа: <http://www.akademout.ru/courseworks/information1/3.php>, свободный. – Загл. с экрана.
- 9 Тейс Г. Н. Методы и средства защиты информации. Асимметричный алгоритм шифрования RSA. [Электронный ресурс]. – Режим доступа: <http://www.novsu.ru/file/1036040>, свободный. – Загл. с экрана.

- 10 Алгоритм шифрования RSA на пальцах. [Электронный ресурс]. – Режим доступа: <http://teh-box.ru/informationsecurity/algorithm-shifrovaniya-rsa-na-palцах.html>, свободный. – Загл. с экрана.
- 11 Введение в криптоанализ. Криптоанализ симметричных криптосистем: блочные шифры. [Электронный ресурс]. – Режим доступа: http://www.nrjetix.com/fileadmin/doc/publications/Lectures_security/Lecture4-1.pdf, свободный. – Загл. с экрана.
- 12 Метод полного перебора. [Электронный ресурс]. – Режим доступа: <http://ru.likefm.org/artist/Bruteforce>, свободный. – Загл. с экрана.
- 13 Классический криптоанализ. [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/271257/>, свободный. – Загл. с экрана.
- 14 Статистический криптоанализ. [Электронный ресурс]. – Режим доступа: http://dictionary.sensagent.com/Статистический_криптоанализ/ru-ru/, свободный. – Загл. с экрана.
- 15 Введение в криптографию. Шифры замены. [Электронный ресурс]. – Режим доступа: <http://www.unn.ru/bibn/files/materials/kript/kr1.pdf>, свободный. – Загл. с экрана.
- 16 Введение в Криптоанализ. Криптоанализ симметричных криптосистем: блочные шифры. [Электронный ресурс]. – Режим доступа: http://www.nrjetix.com/fileadmin/doc/publications/Lectures_security/Lecture4-1.pdf, свободный. – Загл. с экрана.
- 17 Ященко В. В. Историческое введение в основы криптологии. – Москва: МЦНМО, 2012. – 335 с.
- 18 Авдошин С. М. Криптоанализ: современное состояние и перспективы развития. – Москва: Новые Технологии, 2007. – 32 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код программного средства

```

#include "bigrammfreqanalysis.h"
#include <QMap>

BigrammFreqAnalysis::BigrammFreqAnalysis(QObject *parent) : QObject(parent)
{
}

QMap<QString, int> BigrammFreqAnalysis::calculate(const QString &ctext, int size)
{
    QVector<QVector<int> > bigrammTable = {
        {2, 7, 8, 6, 7, 7, 7, 7, 4, 7, 7, 7, 8, 8, 3, 7, 6, 7, 8, 2, 6, 6, 7, 7, 5, 5, 0, 0, 0, 0, 6, 7, 9},
        {7, 1, 1, 0, 1, 6, 2, 2, 6, 0, 5, 6, 3, 5, 7, 2, 7, 5, 0, 7, 0, 5, 4, 1, 0, 5, 5, 7, 2, 2, 0, 3, 5},
        {8, 0, 5, 0, 4, 8, 0, 3, 7, 1, 6, 7, 5, 6, 8, 4, 6, 6, 6, 6, 0, 3, 0, 1, 3, 0, 0, 8, 2, 0, 0, 4, 8},
        {6, 0, 1, 1, 6, 5, 0, 0, 6, 0, 4, 5, 4, 4, 8, 0, 7, 0, 0, 6, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 4},
        {8, 1, 6, 3, 4, 8, 1, 0, 7, 0, 4, 7, 1, 7, 8, 4, 6, 5, 2, 7, 1, 3, 3, 3, 4, 0, 0, 6, 4, 0, 4, 5, 7},
        {5, 5, 6, 7, 8, 6, 6, 6, 4, 7, 7, 8, 8, 9, 6, 5, 8, 8, 9, 3, 3, 6, 5, 6, 5, 6, 0, 0, 1, 1, 5, 5, 9},
        {6, 0, 0, 0, 6, 7, 2, 1, 7, 0, 5, 0, 2, 7, 1, 0, 1, 2, 1, 3, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2},
        {8, 4, 6, 2, 6, 4, 1, 1, 6, 1, 5, 5, 6, 6, 7, 1, 5, 0, 0, 6, 0, 0, 2, 1, 0, 0, 2, 6, 2, 0, 0, 4, 6},
        {6, 6, 7, 6, 6, 8, 5, 7, 7, 7, 7, 6, 8, 8, 5, 5, 7, 8, 8, 1, 5, 7, 7, 7, 6, 3, 0, 1, 0, 0, 6, 7, 9},
        {0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 3, 6, 5, 4, 0, 0, 0, 6, 6, 0, 0, 0, 1, 2, 3, 0, 0, 0, 0, 0, 0, 8},
        {8, 1, 5, 1, 1, 6, 5, 2, 7, 1, 2, 7, 0, 5, 8, 0, 7, 6, 6, 7, 0, 0, 6, 0, 1, 0, 0, 0, 0, 0, 0, 7},
        {8, 4, 1, 2, 1, 8, 6, 1, 8, 0, 4, 4, 1, 6, 7, 0, 0, 3, 3, 6, 3, 0, 0, 3, 1, 1, 0, 6, 8, 0, 7, 8, 6},
        {7, 5, 7, 2, 2, 8, 0, 1, 7, 0, 4, 4, 7, 6, 8, 5, 1, 3, 1, 6, 1, 0, 0, 0, 0, 0, 0, 7, 3, 0, 0, 6, 8},
        {9, 0, 3, 3, 6, 8, 1, 1, 9, 0, 6, 0, 1, 7, 8, 0, 0, 5, 7, 6, 5, 2, 5, 3, 0, 0, 0, 8, 5, 0, 4, 6, 7},
        {2, 8, 8, 8, 8, 6, 7, 7, 6, 8, 7, 8, 8, 7, 6, 7, 8, 8, 8, 3, 2, 5, 6, 7, 6, 5, 0, 0, 1, 5, 2, 5, 9},
        {7, 0, 0, 0, 0, 8, 0, 4, 7, 0, 3, 6, 1, 4, 8, 4, 9, 4, 5, 6, 2, 0, 1, 0, 0, 0, 0, 4, 5, 3, 0, 4, 4},
        {9, 1, 6, 4, 4, 8, 6, 0, 8, 0, 5, 2, 6, 6, 8, 4, 2, 6, 6, 7, 3, 5, 4, 2, 4, 2, 0, 7, 4, 0, 1, 6, 7},
        {6, 4, 6, 2, 5, 7, 2, 0, 7, 0, 7, 8, 6, 6, 8, 7, 5, 6, 9, 6, 3, 5, 1, 5, 5, 0, 0, 5, 6, 1, 3, 8, 7},
        {8, 2, 7, 1, 4, 8, 0, 0, 8, 0, 6, 4, 5, 6, 9, 3, 8, 8, 4, 6, 0, 0, 0, 4, 0, 2, 1, 7, 8, 0, 1, 5, 8},
        {3, 4, 4, 6, 6, 7, 6, 5, 3, 3, 6, 5, 5, 6, 0, 6, 7, 7, 7, 1, 5, 5, 0, 6, 3, 6, 0, 0, 0, 0, 7, 4, 8},
        {6, 0, 0, 0, 0, 5, 0, 0, 6, 0, 0, 2, 2, 0, 6, 0, 4, 0, 3, 5, 4, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2},
        {4, 3, 3, 0, 0, 4, 0, 0, 3, 0, 1, 1, 0, 5, 6, 0, 5, 3, 1, 3, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 8},
        {5, 0, 6, 0, 0, 6, 0, 0, 7, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 5},
        {7, 0, 1, 0, 0, 8, 0, 0, 7, 0, 6, 1, 0, 6, 2, 0, 1, 0, 7, 3, 0, 0, 0, 1, 3, 0, 0, 1, 3, 0, 0, 4},
        {5, 0, 0, 0, 0, 6, 0, 0, 7, 0, 3, 3, 0, 3, 4, 0, 3, 0, 3, 4, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 5},
        {6, 0, 0, 0, 0, 7, 0, 0, 6, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 4, 0, 1, 0, 1},
        {0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 2},
        {1, 4, 7, 3, 5, 7, 1, 5, 1, 7, 5, 5, 6, 2, 1, 5, 5, 5, 6, 0, 0, 7, 0, 5, 4, 1, 0, 1, 0, 0, 0, 1, 8},
        {0, 1, 0, 0, 0, 3, 0, 7, 1, 0, 6, 0, 4, 7, 1, 0, 0, 6, 4, 0, 0, 0, 0, 1, 6, 1, 0, 0, 0, 0, 6, 2, 8},
        {0, 0, 4, 0, 0, 1, 0, 0, 0, 2, 6, 5, 2, 1, 0, 2, 0, 1, 7, 0, 4, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
        {0, 5, 0, 0, 2, 0, 1, 2, 0, 4, 1, 0, 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 0, 6, 1, 7, 0, 0, 1, 0, 3, 0, 7},
        {0, 1, 5, 2, 5, 6, 2, 5, 0, 2, 2, 3, 6, 5, 0, 1, 4, 4, 7, 0, 0, 4, 4, 3, 0, 4, 0, 0, 0, 0, 6, 4, 9},
        {7, 8, 9, 7, 8, 7, 5, 8, 8, 3, 8, 6, 8, 9, 9, 9, 8, 9, 8, 7, 7, 6, 7, 8, 5, 1, 1, 2, 1, 8, 2, 6, 0}
    };
};

QString alpha;
alpha = QString("АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ ");
QMap<QString, int> freq;
auto splittedString = ctext.split(";");

QString tmp("");

```

```

    for (auto k = 0; k < size; k++)
        for (auto i = k; i < size; i++)
            for (auto j = 0; j < size - 1; j++) {
                tmp = QString(splittedString[j][k]) + QString(splittedString[j][i]);
                freq[tmp] = bigrammTable.at(alpha.indexOf(tmp[1])).at(alpha.indexOf(tmp[0]));
            }
    return freq;
}
#include "bruteforce.h"
#include "rsacrypt.h"
#include "singleshift.h"
#include "simplereplacement.h"
#include "gronspheld.h"
#include <QMap>
#include <QDebug>

Bruteforce::Bruteforce(QObject *parent) : QObject(parent)
{

}

QList<std::tuple<int, int, QString> > Bruteforce::bruteRSA(int eParam, int nParam, int lang,
const QString &ctxt)
{
    QList<std::tuple<int, int, QString> > result;
    RSACrypt rsaCrypt(lang);
    QList<int> primes = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79,
                        83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173,
                        179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263,
269, 271,
                        277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373,
379, 383,
                        389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479,
487, 491,
                        499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601,
607, 613,
                        617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719,
727, 733,
                        739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839,
853, 857,
                        859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971,
977, 983,
                        991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063,
1069, 1087,
                        1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181,
1187, 1193,
                        1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289,
1291, 1297,
                        1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423,
1427, 1429,

```

1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499,
 1511, 1523,
 1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609,
 1613, 1619,
 1621, 1627, 1637, 1657, 1663, 1667, 1669, 1693, 1697, 1699, 1709, 1721, 1723,
 1733, 1741,
 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823, 1831, 1847, 1861,
 1867, 1871,
 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979,
 1987, 1993,
 1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083,
 2087, 2089,
 2099, 2111, 2113, 2129, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207,
 2213, 2221,
 2237, 2239, 2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293, 2297, 2309, 2311,
 2333, 2339,
 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389, 2393, 2399, 2411, 2417,
 2423, 2437,
 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539, 2543, 2549, 2551,
 2557, 2579,
 2591, 2593, 2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683,
 2687, 2689,
 2693, 2699, 2707, 2711, 2713, 2719, 2729, 2731, 2741, 2749, 2753, 2767, 2777,
 2789, 2791,
 2797, 2801, 2803, 2819, 2833, 2837, 2843, 2851, 2857, 2861, 2879, 2887, 2897,
 2903, 2909,
 2917, 2927, 2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001, 3011, 3019, 3023,
 3037, 3041,
 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169,
 3181, 3187,
 3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301,
 3307, 3313,
 3319, 3323, 3329, 3331, 3343, 3347, 3359, 3361, 3371, 3373, 3389, 3391, 3407,
 3413, 3433,
 3449, 3457, 3461, 3463, 3467, 3469, 3491, 3499, 3511, 3517, 3527, 3529, 3533,
 3539, 3541,
 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607, 3613, 3617, 3623, 3631, 3637,
 3643, 3659,
 3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761, 3767,
 3769, 3779,
 3793, 3797, 3803, 3821, 3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889,
 3907, 3911,
 3917, 3919, 3923, 3929, 3931, 3943, 3947, 3967, 3989, 4001, 4003, 4007, 4013,
 4019, 4021,
 4027, 4049, 4051, 4057, 4073, 4079, 4091, 4093, 4099, 4111, 4127, 4129, 4133,
 4139, 4153,
 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231, 4241, 4243, 4253, 4259,
 4261, 4271,
 4273, 4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397,
 4409, 4421,
 4423, 4441, 4447, 4451, 4457, 4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519,
 4523, 4547,

4549, 4561, 4567, 4583, 4591, 4597, 4603, 4621, 4637, 4639, 4643, 4649, 4651,
4657, 4663,
4673, 4679, 4691, 4703, 4721, 4723, 4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799,
4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909, 4919, 4931, 4933,
4937, 4943,
4951, 4957, 4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023,
5039, 5051,
5059, 5077, 5081, 5087, 5099, 5101, 5107, 5113, 5119, 5147, 5153, 5167, 5171,
5179, 5189,
5197, 5209, 5227, 5231, 5233, 5237, 5261, 5273, 5279, 5281, 5297, 5303, 5309,
5323, 5333,
5347, 5351, 5381, 5387, 5393, 5399, 5407, 5413, 5417, 5419, 5431, 5437, 5441,
5443, 5449,
5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531, 5557, 5563,
5569, 5573,
5581, 5591, 5623, 5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689,
5693, 5701,
5711, 5717, 5737, 5741, 5743, 5749, 5779, 5783, 5791, 5801, 5807, 5813, 5821,
5827, 5839,
5843, 5849, 5851, 5857, 5861, 5867, 5869, 5879, 5881, 5897, 5903, 5923, 5927,
5939, 5953,
5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047, 6053, 6067, 6073, 6079, 6089,
6091, 6101,
6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211, 6217,
6221, 6229,
6247, 6257, 6263, 6269, 6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329,
6337, 6343,
6353, 6359, 6361, 6367, 6373, 6379, 6389, 6397, 6421, 6427, 6449, 6451, 6469,
6473, 6481,
6491, 6521, 6529, 6547, 6551, 6553, 6563, 6569, 6571, 6577, 6581, 6599, 6607,
6619, 6637,
6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703, 6709, 6719, 6733, 6737,
6761, 6763,
6779, 6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869,
6871, 6883,
6899, 6907, 6911, 6917, 6947, 6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991,
6997, 7001,
7013, 7019, 7027, 7039, 7043, 7057, 7069, 7079, 7103, 7109, 7121, 7127, 7129,
7151, 7159,
7177, 7187, 7193, 7207, 7211, 7213, 7219, 7229, 7237, 7243, 7247, 7253, 7283,
7297, 7307,
7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417, 7433, 7451, 7457,
7459, 7477,
7481, 7487, 7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559,
7561, 7573,
7577, 7583, 7589, 7591, 7603, 7607, 7621, 7639, 7643, 7649, 7669, 7673, 7681,
7687, 7691,
7699, 7703, 7717, 7723, 7727, 7741, 7753, 7757, 7759, 7789, 7793, 7817, 7823,
7829, 7841,
7853, 7867, 7873, 7877, 7879, 7883, 7901, 7907, 7919, 7927, 7933, 7937, 7949,
7951, 7963,

```

7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089, 8093, 8101,
8111, 8117,
8123, 8147, 8161, 8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237,
8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311, 8317, 8329, 8353, 8363, 8369, 8377,
8387, 8389,
8419, 8423, 8429, 8431, 8443, 8447, 8461, 8467, 8501, 8513, 8521, 8527, 8537,
8539, 8543,
8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627, 8629, 8641, 8647, 8663, 8669,
8677, 8681,
8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761, 8779,
8783, 8803,
8807, 8819, 8821, 8831, 8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923,
8929, 8933,
8941, 8951, 8963, 8969, 8971, 8999, 9001, 9007, 9011, 9013, 9029, 9041, 9043,
9049, 9059,
9067, 9091, 9103, 9109, 9127, 9133, 9137, 9151, 9157, 9161, 9173, 9181, 9187,
9199, 9203,
9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283, 9293, 9311, 9319, 9323,
9337, 9341,
9343, 9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437,
9439, 9461,
9463, 9467, 9473, 9479, 9491, 9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587,
9601, 9613,
9619, 9623, 9629, 9631, 9643, 9649, 9661, 9677, 9679, 9689, 9697, 9719, 9721,
9733, 9739,
9743, 9749, 9767, 9769, 9781, 9787, 9791, 9803, 9811, 9817, 9829, 9833, 9839,
9851, 9857,
9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941, 9949, 9967,
9973};
for (auto i : primes) {
    rsaCrypt.setKeysParams(eParam, i, nParam);
    result.append(std::tuple<int, int, QString>(i, nParam, rsaCrypt.decrypt(ctext)));
}
return result;
}

QList<std::tuple<QString, QString> > BruteForce::bruteSimpleReplacement(const QString
&ctext)
{
    QString alpha = QString("
ABCDEFGHIJKLMNOPQRSTUVWXYZАБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮ
Я");
    QMap<QString, QString> tmpMap;
    SimpleReplacement srepl(tmpMap);
    QString tmpString;
    QList<std::tuple<QString, QString> > result;
    for (auto i = 0; i < alpha.size(); i++) {
        for (auto j = 0; j < alpha.size(); j++) {
            tmpMap[alpha.mid(j, 1)] = alpha.mid((i+j) % alpha.size(), 1);
            tmpString += alpha.mid((i+j) % alpha.size(), 1);
        }
    }
}

```

```

        srepl.setKey(tmpMap);
        result.append(std::tuple<QString, QString>(tmpString, srepl.decrypt(ctext)));
        tmpString.clear();
    }
    return result;
}

```

```

QList<std::tuple<int, QString> > Bruteforce::bruteGronsfeld(int lang, int range, const QString
&ctext)
{
    Gronspheld grons(lang);
    QList<std::tuple<int, QString> > result;
    for (auto i = 0; i < range; i++) {
        grons.setKey(QString::number(i));
        result.append(std::tuple<int, QString>(i, grons.decrypt(ctext)));
    }
    return result;
}
#include "cardano.h"
#include <QDebug>

```

```

Cardano::Cardano(int lang, QObject *parent) : QObject(parent)
{
    switch(lang) {
    case 0:
        alpha = QString(" ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        break;
    case 1:
        alpha = QString(" АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
    }
}

```

```

void Cardano::setKey(QVector<QVector<bool> > key)
{
    keyMatrix = key;
}

```

```

QString Cardano::encrypt(const QString &ptext)
{
    QVector<QVector<int> > text = {
        QVector<int>(6), QVector<int>(6), QVector<int>(6),
        QVector<int>(6), QVector<int>(6), QVector<int>(6)
    };
    int count = 0;
    for (auto round = 0; round < 4; round++) {
        for (auto i = 0; i < 6; i++)
            for (auto j = 0; j < 6; j++){
                if (keyMatrix[i][j])
                    text[i][j] = alpha.indexOf(ptext.mid(count++, 1));
            }
        rotateKeyMatrix();
    }
}

```

```

    }
    qDebug() << count;
    QString result;
    for (auto i = 0; i < 6; i++)
        for (auto j = 0; j < 6; j++)
            result += alpha.mid(text[i][j], 1);
    return result;
}

QString Cardano::decrypt(const QString &ctext)
{
    QVector<QVector<int> > text = {
        QVector<int>(6), QVector<int>(6), QVector<int>(6),
        QVector<int>(6), QVector<int>(6), QVector<int>(6)
    };
    int count = 0;
    for (auto i = 0; i < 6; i++)
        for (auto j = 0; j < 6; j++)
            text[i][j] = alpha.indexOf(ctext.mid(count++, 1));

    QString result;
    for (auto round = 0; round < 4; round++) {
        for (auto i = 0; i < 6; i++)
            for (auto j = 0; j < 6; j++) {
                if (keyMatrix[i][j])
                    result += alpha.mid(text[i][j], 1);
            }
        rotateKeyMatrix();
    }
    return result;
}

void Cardano::rotateKeyMatrix() //TODO: ПОМЕНЯТЬ НАПРАВЛЕНИЕ ПОВОРОТА.
    СЕЙЧАС ПОВОРАЧИВАЕТ ПО ЧАСОВОЙ СТРЕЛКЕ)
{
    QChar tmp;
    QVector<QVector<bool> > tmpMatrix = {
        QVector<bool>(6), QVector<bool>(6), QVector<bool>(6),
        QVector<bool>(6), QVector<bool>(6), QVector<bool>(6)
    };
    for (auto i = 0; i < 6; i++)
        for (auto j = 0; j < 6; j++)
            tmpMatrix[i][j] = keyMatrix[6-j-1][i];
    keyMatrix = tmpMatrix;
}

#include "cardanokeydialog.h"
#include "ui_cardanokeydialog.h"
#include <QTableWidget>
#include <QCheckBox>
#include <QDebug>

```

```

CardanoKeyDialog::CardanoKeyDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::CardanoKeyDialog)
{
    ui->setupUi(this);

    ui->keyTable->setColumnCount(6);
    ui->keyTable->setRowCount(6);
    ui->keyTable->horizontalHeader()->hide();
    ui->keyTable->verticalHeader()->hide();
    ui->keyTable->resizeColumnsToContents();

    for (auto i = 0; i < ui->keyTable->rowCount(); i++)
        for (auto j = 0; j < ui->keyTable->columnCount(); j++) {
            ui->keyTable->setItem(i, j, new QTableWidgetItem());
            ui->keyTable->item(i, j)->setCheckState(Qt::Unchecked);
        }
    QObject::connect(ui->cancelButton,                &QPushButton::clicked,                this,
&CardanoKeyDialog::clearClose);
    QObject::connect(ui->okButton,                    &QPushButton::clicked,                    this,
&CardanoKeyDialog::saveMatrix);
    //QObject::connect(ui->keyTable,    SIGNAL(itemChanged(QTableWidgetItem*)),    this,
    SLOT(toggleCells(QTableWidgetItem*)));
}

CardanoKeyDialog::~CardanoKeyDialog()
{
    delete ui;
}

QVector<QVector<bool> > CardanoKeyDialog::getMatrix()
{
    return keyMatrix;
}

void CardanoKeyDialog::clearClose()
{
    for (auto i = 0; i < ui->keyTable->rowCount(); i++)
        for (auto j = 0; j < ui->keyTable->columnCount(); j++) {
            ui->keyTable->item(i,j)->setCheckState(Qt::Unchecked);
            keyMatrix[i][j] = false;
        }
    close();
}

void CardanoKeyDialog::saveMatrix()
{
    for (auto i = 0; i < ui->keyTable->rowCount(); i++)
        for (auto j = 0; j < ui->keyTable->columnCount(); j++) {
            if (ui->keyTable->item(i, j)->checkState() == Qt::Checked) {
                keyMatrix[i][j] = true;
            }
        }
}

```

```

    }
    }
    close();
}

//void CardanoKeyDialog::toggleCells(QTableWidgetItem *item)
//{
//    for (auto i = 0; i < ui->keyTable->rowCount(); i++) {
//        if (i == item->row())
//            continue;
//        if (ui->keyTable->item(i, item->column())->checkState() == Qt::Checked)
//            ui->keyTable->item(i, item->column())->setFlags(ui->keyTable->item(i, item-
//>column())->flags() ^ Qt::ItemIsEnabled);
//        else
//            ui->keyTable->cellWidget(i, item->column())->setEnabled(true);
//    }
//}

#include "doubletransposition.h"
#include <list>
#include <cmath>
#include <QDebug>
using namespace std;

DoubleTransposition::DoubleTransposition(QObject *parent) : QObject(parent)
{
    sShift = new SingleShift(this);
}

DoubleTransposition::~DoubleTransposition()
{
}

QString DoubleTransposition::encrypt(QString input, QChar padChar)
{
    input += QString(" ").repeated(key1.size()*key2.size() - input.size());
    sShift->setKey(key1);
    input = sShift->encrypt(input, padChar);
    sShift->setKey(key2);
    input = sShift->encrypt(input, padChar);
    input = transpose(input);
    return input;
}

QString DoubleTransposition::decrypt(QString input)
{
    sShift->setKey(key2);
    input = backTranspose(input);
    input = sShift->decrypt(input);
    sShift->setKey(key1);
    input = sShift->decrypt(input);
}

```

```

    return input;
}

void DoubleTransposition::setKeys(QString k1, QString k2)
{
    key1 = k1;
    key2 = k2;
}

QString DoubleTransposition::transpose(QString input)
{
    QString result;
    for (auto i = 0; i < key1.size(); i++)
        for (auto j = 0; j < key2.size(); j++)
            result += input[key1.size()*j+i];
    return result;
}

QString DoubleTransposition::backTranspose(QString input)
{
    QString result;
    result.resize(input.size());
    int cnt = 0;
    for (auto j = 0; j < key1.size(); j++)
        for (auto i = 0; i < key2.size(); i++)
            result[key1.size()*i+j] = input[cnt++]; //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    return result;
}
#include "factorization.h"
#include "bigint/BigIntegerLibrary.hh"
#include <QDebug>

Factorization::Factorization(QObject *parent) : QObject(parent)
{

}

 QVector<BigInteger> Factorization::factorize(std::string number)
{
    QVector<BigInteger> result;
    BigInteger n = stringToBigInteger(number);
    BigInteger i = 2;
    while (n > i * i)
        if ((n % i).isZero()) {
            result.append(i);
            n /= i;
        }
        else
            i++;
    if (n > 1)
        result.append(n);
    return result;}

```

```

#include "gammacypher.h"
#include <random>
#include <QDebug>
#include <QTextCodec>

GammaCypher::GammaCypher(QObject *parent) : QObject(parent)
{
    gamma = new QString();
    alpha = new QString(" АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
}

GammaCypher::~GammaCypher()
{
    delete gamma;
    delete alpha;
}

QString GammaCypher::encrypt(const QString &ptext) const
{
    QString arr;
    int tmpAlp = 0, tmpGam = 0;
    for (auto i = 0; i < ptext.size(); i++) {
        tmpAlp = alpha->indexOf(ptext[i]);
        tmpGam = alpha->indexOf(gamma->at(i % gamma->size()));
        arr.append(alpha->at(tmpAlp ^ tmpGam)); //Ссанный английский алфавит не кратен
        степени 2, так что липила, иди нахуй
    }
    return arr;
}

QString GammaCypher::decrypt(const QString &ctext) const
{
    QString arr;
    int Ci = 0, Gi = 0;
    for (auto i = 0; i < ctext.size(); i++) {
        Ci = alpha->indexOf(ctext[i]);
        Gi = alpha->indexOf(gamma->at(i % gamma->size()));
        arr.append(alpha->at(Ci ^ Gi)); //Ссанный английский алфавит не кратен степени 2, так
        что липила, иди нахуй
    }
    return arr;
}

int GammaCypher::genNewGamma()
{
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<int> dist(0, alpha->size()-1);
    gamma->clear();

    for (int i=0; i<32; ++i)
        this->gamma->append(alpha->at(dist(mt)));
}

```

```

    return 0;
}

QString GammaCypher::getGamma() const {
    return QString(this->gamma->data());
}

void GammaCypher::setGamma(QString gammatext) const{
    gamma->clear();
    gamma->append(gammatext);
}
#include "gronspheld.h"
#include <random>
#include <QDebug>

Gronspheld::Gronspheld(int lang, QObject *parent) : QObject(parent)
{
    key = new QString();
    switch (lang) {
    case 0:
        alpha = new QString(" ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        break;
    case 1:
        alpha = new QString(" АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
    }
}

Gronspheld::~Gronspheld()
{
}

void Gronspheld::setKey(QString k)
{
    key->clear();
    key->append(k);
}

QString Gronspheld::encrypt(QString ptext)
{
    int offset = 0, indexOf = 0;
    QString ctext;
    QString newKey = key->repeated((ptext.size() / key->size() + 1);
    if (newKey.size() > ptext.size())
        newKey.truncate(ptext.size());
    for (auto i = 0; i < ptext.size(); i++) {
        indexOf = alpha->indexOf(ptext.at(i));
        if ((indexOf + newKey.at(i).digitValue()) >= alpha->size()) {
            offset = (indexOf + newKey.at(i).digitValue()) % alpha->size();
        }
        else

```

```

        offset = indexOf + newKey.at(i).digitValue();
        qDebug() << offset;
        ctext.append(alpha->at(offset));
    }
    return ctext;
}

QString Gronspheld::decrypt(QString ctext)
{
    int offset = 0, indexOf = 0;
    QString ptext;
    QString newKey = key->repeated((ctext.size() / key->size() + 1);
    if (newKey.size() > ctext.size())
        newKey.truncate(ctext.size());
    for (auto i = 0; i < ctext.size(); i++) {
        indexOf = alpha->indexOf(ctext.at(i));
        if ((indexOf - newKey.at(i).digitValue()) < 0)
            offset = indexOf - newKey.at(i).digitValue() + alpha->size();
        else
            offset = indexOf - newKey.at(i).digitValue();
        ptext.append(alpha->at(offset));
    }
    return ptext;
}

QString Gronspheld::genNewKey()
{
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<unsigned int> dist(0, (unsigned int) -2);
    key->clear();
    key->append(QString::number(dist(mt)));
    return *key;
}

void Gronspheld::setAlpha(int lang)
{
    switch (lang) {
    case 0:
        alpha = new QString(" ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        break;
    case 1:
        alpha = new QString(" АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
    }
}

#include "keyvaluedialog.h"
#include "ui_keyvaluedialog.h"
#include <QTableWidget>

KeyValueDialog::KeyValueDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::KeyValueDialog)

```

```

{
    ui->setupUi(this);
    this->setModal(true);
    ui->tableWidget->setColumnCount(2);
    ui->tableWidget->setRowCount(58);
    ui->tableWidget->horizontalHeader()->hide();
    ui->tableWidget->verticalHeader()->hide();
    dict = new QMap<QString, QString>({
        {QString(" "), QString(" ")}, {QString("A"), QString("A")},
        {QString("B"), QString("B")}, {QString("C"), QString("C")},
        {QString("D"), QString("D")}, {QString("E"), QString("E")},
        {QString("F"), QString("F")}, {QString("G"), QString("G")},
        {QString("H"), QString("H")}, {QString("I"), QString("I")},
        {QString("J"), QString("J")}, {QString("K"), QString("K")},
        {QString("L"), QString("L")}, {QString("M"), QString("M")},
        {QString("N"), QString("N")}, {QString("O"), QString("O")},
        {QString("P"), QString("P")}, {QString("Q"), QString("Q")},
        {QString("R"), QString("R")}, {QString("S"), QString("S")},
        {QString("T"), QString("T")}, {QString("U"), QString("U")},
        {QString("V"), QString("V")}, {QString("W"), QString("W")},
        {QString("X"), QString("X")}, {QString("Y"), QString("Y")},
        {QString("Z"), QString("Z")}, {QString("A"), QString("A")},
        {QString("B"), QString("B")}, {QString("B"), QString("B")},
        {QString("I"), QString("I")}, {QString("J"), QString("J")},
        {QString("E"), QString("E")}, {QString("K"), QString("K")},
        {QString("3"), QString("3")}, {QString("H"), QString("H")},
        {QString("K"), QString("K")},
        {QString("J"), QString("J")}, {QString("M"), QString("M")},
        {QString("H"), QString("H")}, {QString("O"), QString("O")},
        {QString("I"), QString("I")}, {QString("P"), QString("P")},
        {QString("C"), QString("C")}, {QString("T"), QString("T")},
        {QString("Y"), QString("Y")}, {QString("Φ"), QString("Φ")},
        {QString("X"), QString("X")}, {QString("I"), QString("I")},
        {QString("C"), QString("C")}, {QString("III"), QString("III")},
        {QString("III"), QString("III")}, {QString("b"), QString("b")},
        {QString("bI"), QString("bI")}, {QString("b"), QString("b")},
        {QString("ᄃ"), QString("ᄃ")}, {QString("IO"), QString("IO")},
        {QString("Я"), QString("Я")});

    for (auto i = 0; i < dict->keys().length(); i++) {
        ui->tableWidget->setItem(i, 0, new QTableWidgetItem(dict->keys()[i]));
        ui->tableWidget->item(i,0)->setFlags(ui->tableWidget->item(i,0)->flags() ^
Qt::ItemIsEditable
        ^ Qt::ItemIsSelectable);
        ui->tableWidget->setItem(i, 1, new QTableWidgetItem(dict->keys()[i]));
    }

    connect(ui->upButton, &QPushButton::clicked, this, &KeyValueDialog::moveUp);
    connect(ui->downButton, &QPushButton::clicked, this, &KeyValueDialog::moveDown);
    connect(ui->rememberDictButton, &QPushButton::clicked, this,
    &KeyValueDialog::slotRememberDict);
}

```

```

KeyValueDialog::~KeyValueDialog()
{
    delete ui;
}

 QMap<QString, QString> KeyValueDialog::readDict() const
{
    return *dict;
}

void KeyValueDialog::setCustomDict(QMap<QString, QString> *customDict)
{
    dict = customDict;
    for (auto i = 0; i < customDict->keys().length(); i++)
        ui->tableWidget->item(i, 0)->setText(customDict->keys()[i]);
}

void KeyValueDialog::moveUp() {
    QTableWidgetItem *tmpItem = new QTableWidgetItem(ui->tableWidget->item(0,1)->text());
    delete ui->tableWidget->item(0,1);
    QTableWidgetItem *tmp;
    for (auto i = 0; i < ui->tableWidget->rowCount() - 1; i++) {
        tmp = new QTableWidgetItem(ui->tableWidget->item((i+1) % ui->tableWidget-
>rowCount(), 1)->text());
        ui->tableWidget->setItem(i, 1, tmp);
        delete ui->tableWidget->item((i+1) % ui->tableWidget->rowCount(), 1);
    }
    ui->tableWidget->setItem(ui->tableWidget->rowCount()-1, 1, tmpItem);
}

void KeyValueDialog::moveDown() {
    QTableWidgetItem *tmpItem = new QTableWidgetItem(ui->tableWidget->item(ui-
>tableWidget->rowCount()-1,1)->text());
    delete ui->tableWidget->item(ui->tableWidget->rowCount()-1, 1);
    QTableWidgetItem *tmp;
    for (auto i = ui->tableWidget->rowCount() - 1; i > 0; i--) {
        tmp = new QTableWidgetItem(ui->tableWidget->item((i-1) % ui->tableWidget-
>rowCount(), 1)->text());
        ui->tableWidget->setItem(i, 1, tmp);
        delete ui->tableWidget->item((i-1) % ui->tableWidget->rowCount(), 1);
    }
    ui->tableWidget->setItem(0, 1, tmpItem);
}

void KeyValueDialog::slotRememberDict()
{
    for (auto i = 0; i < ui->tableWidget->rowCount(); i++) {
        dict->insert(ui->tableWidget->item(i,0)->text(), ui->tableWidget->item(i,1)->text());
    }
}

```

```

#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QAbstractButton>
#include <QPushButton>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <QTextStream>
#include "simplereplacement.h"
#include "singleshift.h"
#include "doubletransposition.h"
#include "factorization.h"
#include "gronspheld.h"
#include "cardano.h"
#include "rsacrypt.h"
#include "bruteforce.h"
#include "symbolfreqanalysis.h"
#include "bigrammfreqanalysis.h"
#include <gammacypher.h>
#include <QCheckBox>
#include <QRadioButton>
#include <QLineEdit>
#include <QDebug>
#include <QComboBox>
#include <random>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setWindowTitle("KMZI");
    ui->handmadeKey->setChecked(true);
    ui->keyLine->setVisible(false);
    ui->encrypt->setChecked(true);
    ui->genKey->setVisible(false);
    ui->key2Line->setVisible(false);
    ui->rsaKeyButton->setVisible(false);
    ui->cardanButton->setVisible(false);
    ui->bruteTable->verticalHeader()->hide();
    ui->bruteTable->horizontalHeader()->setStretchLastSection(true);

```

```

ui->cFreqTable->verticalHeader()->hide();
ui->cFreqTable->horizontalHeader()->setStretchLastSection(true);
ui->bigrammSize->setValue(5);
validatorList = new QList<QRegExpValidator*>();
validatorList->append(new QRegExpValidator(QRegExp(QString("[A-EЖ-ИК-Я\\s]+"))));
//gamma
validatorList->append(new QRegExpValidator(QRegExp(QString("[A-ZA-EЖ-ИК-Я]+"))));
//transpos
validatorList->append(new QRegExpValidator(QRegExp(QString("[\\d]+")))); //grons
kvd = new KeyValueDialog();
rsaKeyDialog = new RSAKeyDialog(this);
cardKeyDialog = new CardanoKeyDialog(this);
QObject::connect(ui->genKey, &QAbstractButton::clicked, [=] () {
    ui->keyLine->setEnabled(true);
});
QObject::connect(ui->fromFile, &QAbstractButton::clicked, [=] () {
    ui->keyLine->setEnabled(false);
});
QObject::connect(ui->handmadeKey, &QAbstractButton::clicked, [=] () {
    ui->keyLine->setEnabled(true);
    switch (ui->cyphersBox->currentIndex()) {
        case 0:
            ui->keyLine->setVisible(false);
            ui->keyPair->setVisible(true);
            break;
        case 1:
            ui->keyLine->setVisible(false);
            ui->keyPair->setVisible(true);
            break;
        case 2:
            ui->keyLine->setVisible(true);
            ui->keyPair->setVisible(false);
            break;
    }
});
QObject::connect(ui->clearPlainTextButton, &QPushButton::clicked, ui->plainText,
&QPlainTextEdit::clear);
QObject::connect(ui->clearCyphTextButton, &QPushButton::clicked, ui->cypherText,
&QPlainTextEdit::clear);
QObject::connect(ui->openPlainText, &QPushButton::clicked, [=] () {this->loadFile(ui-
>plainText);});
QObject::connect(ui->openCyphText, &QPushButton::clicked, [=] () {this->loadFile(ui-
>cypherText);});
QObject::connect(ui->keyPair, &QPushButton::clicked, kvd, &KeyValueDialog::show);
QObject::connect(ui->doButton, &QPushButton::clicked, this,
&MainWindow::performOperation);
QObject::connect(ui->cyphersBox, SIGNAL(currentIndexChanged(int)), this,
SLOT(toggleKeyInputWay(int)));
QObject::connect(ui->plainHex, &QCheckBox::stateChanged, [=] () {this->convertText(ui-
>plainHex, ui->plainText);});
QObject::connect(ui->cryptHex, &QCheckBox::stateChanged, [=] () {this->convertText(ui-
>cryptHex, ui->cypherText);});

```

```

    QObject::connect(ui->rsaKeyButton, &QPushButton::clicked, rsakeyDialog,
&RSAKeyDialog::show);
    QObject::connect(ui->cardanButton, &QPushButton::clicked, cardKeyDialog,
&CardanoKeyDialog::show);
    QObject::connect(ui->factorizeButton, &QPushButton::clicked, this,
&MainWindow::factorize);
    QObject::connect(ui->bruteCyphersBox, SIGNAL(currentIndexChanged(int)), this,
SLOT(toggleBruteFields(int)));
    QObject::connect(ui->bruteButton, &QPushButton::clicked, this, &MainWindow::brute);
    QObject::connect(ui->calcFreqButton, &QPushButton::clicked, this,
&MainWindow::calculateFrequence);
    QObject::connect(ui->bigramCalcButton, &QPushButton::clicked, this,
&MainWindow::calculateBigramm);
    QObject::connect(ui->genKey, &QRadioButton::clicked, this, &MainWindow::generateKey);
    QObject::connect(ui->keyFromFileButton, &QPushButton::clicked, this,
&MainWindow::loadKeyFromFile);
}

```

```

MainWindow::~MainWindow()

```

```

{
    delete ui;
    delete kvd;
    delete rsakeyDialog;
}

```

```

void MainWindow::loadFile(QPlainTextEdit *pte) {
    QString fileName = QFileDialog::getOpenFileName(this, tr("Выберите файл"),
"/home",
tr("Any files (*)"));

    QFile file(fileName);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
        QMessageBox::warning(this, tr("Ошибка"), tr("Не удастся открыть файл"),
QMessageBox::Ok);
    else {
        QString *text = new QString();
        QTextStream in(&file);
        while (!in.atEnd())
            text->append(in.readLine());
        file.close();

        if (text->length() > 5000)
            text->truncate(5000);
        *text = text->toUpper();
        text->remove(QRegExp(QString::fromUtf8("[^A-ZА-ЕЖ-ИК-Я\\s]")));
        pte->clear();
        pte->document()->setPlainText(*text);
        delete text;
    }
}

```

```

void MainWindow::performOperation()
{

```

```

if (ui->encrypt->isChecked()) {
    switch (ui->cyphersBox->currentIndex()) {
        case 0: {
            if (ui->genKey->isChecked()) {

                }
            else if (ui->fromFile->isChecked()) {

                }
            else if (ui->handmadeKey->isChecked()) {
                SimpleReplacement *srepl = new SimpleReplacement(kvd->readDict());
                if (ui->plainText->document()->toPlainText().isEmpty()) {
                    QMessageBox::warning(this, tr("Ошибка"), tr("Не введен открытый текст"));
                    break;
                }
                ui->cypherText->setPlainText(srepl->encrypt(ui->plainText->document()-
>toPlainText()));
                delete srepl;
                break;
            }
        }
        case 1: {
            if (ui->genKey->isChecked()) {

                }
            else if (ui->fromFile->isChecked()) {

                }
            else if (ui->handmadeKey->isChecked()) {
                SimpleReplacement *srepl = new SimpleReplacement(kvd->readDict());
                if (ui->plainText->document()->toPlainText().isEmpty()) {
                    QMessageBox::warning(this, tr("Ошибка"), tr("Не введен открытый текст"));
                    break;
                }
                ui->cypherText->setPlainText(srepl->encrypt(ui->plainText->document()-
>toPlainText()));
                delete srepl;
                break;
            }
        }
        case 2: {
            GammaCypher gcyph(this);
            ui->plainText->setPlainText(ui->plainText-
>toPlainText().remove(QRegExp(QString::fromUtf8("[^А-ЕЖ-ИК-Я\\s]"))));
            if (ui->genKey->isChecked()) {
                gcyph.genNewGamma();
                ui->keyLine->setText(gcyph.getGamma());
                ui->cypherText->clear();
                ui->cypherText->setPlainText(gcyph.encrypt(ui->plainText->toPlainText()));
            }
        }
    }
}

```

```

else if (ui->fromFile->isChecked()) {
    //gcyph.setGamma((*this->loadKey());
    ui->keyLine->clear();
    ui->keyLine->setText(gcyph.getGamma());
    QString cryptText = gcyph.encrypt(ui->plainText->toPlainText());
    ui->cypherText->setPlainText(cryptText);
}
else if (ui->handmadeKey->isChecked()) {
    gcyph.setGamma(ui->keyLine->text());
    QString cryptText = gcyph.encrypt(ui->plainText->toPlainText());
    ui->cypherText->document()->setPlainText(cryptText);
}
break;
}
case 3: {
    SingleShift sShift(this);
    ui->plainText->setPlainText(ui->plainText-
>toPlainText().remove(QRegExp(QString::fromUtf8("[^A-ZA-EЖ-ИК-Я\\s"])))));
    if (ui->genKey->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        sShift.setKey(ui->keyLine->text());
        ui->cypherText->clear();
        ui->cypherText->setPlainText(sShift.encrypt(ui->plainText->toPlainText()));
    }
    break;
}
case 4: {
    DoubleTransposition doubleTrans(this);
    if (ui->genKey->isChecked()){

    }
    else if (ui->handmadeKey->isChecked()) {
        doubleTrans.setKeys(ui->keyLine->text(), ui->key2Line->text());
        ui->cypherText->clear();
        ui->cypherText->setPlainText(doubleTrans.encrypt(ui->plainText->toPlainText()));
    }
    break;
}
case 5: {
    RSACrypt rsacrypt(ui->alphBox->currentIndex(), this);
    ui->plainText->setPlainText(ui->plainText-
>toPlainText().remove(QRegExp(QString::fromUtf8("[^A-ZA-EЖ-ИК-Я\\s"])))));
    if (ui->genKey->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        int e, d, n;
        rsaKeyDialog->getPubKeyParams(e, n);
        rsaKeyDialog->getPrivKeyParams(d, n);
        rsacrypt.setKeysParams(e, d, n);

```

```

        if (rsaKeyDialog->isKeySet()) {
            ui->cypherText->clear();
            ui->cypherText->setPlainText(rsacrypt.encrypt(ui->plainText->toPlainText()));
        }
    }
    break;
}
case 6: {
    Cardano cardano(ui->alphBox->currentIndex(), this);
    ui->plainText->setPlainText(ui->plainText->
    toPlainText().remove(QRegExp(QString::fromUtf8("[^A-ZА-Я\\s]"))));
    if (ui->fromFile->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        cardano.setKey(cardKeyDialog->getMatrix());
        ui->cypherText->clear();
        ui->cypherText->setPlainText(cardano.encrypt(ui->plainText->toPlainText()));
    }
    break;
}
case 7: {
    GronspheId grons(ui->alphBox->currentIndex(), this);
    if (ui->genKey->isChecked()) {
        QString newKey = grons.genNewKey();
        ui->keyLine->setText(newKey);
        ui->cypherText->clear();
        ui->cypherText->setPlainText(grons.encrypt(ui->plainText->toPlainText()));
    }
    else if (ui->handmadeKey->isChecked()) {
        grons.setKey(ui->keyLine->text());
        ui->cypherText->clear();
        ui->cypherText->setPlainText(grons.encrypt(ui->plainText->toPlainText()));
    }
}
}
}
else if (ui->decrypt->isChecked()) {
    switch (ui->cyphersBox->currentIndex()) {
        case 0: {
            if (ui->genKey->isChecked()) {

            }
            else if (ui->fromFile->isChecked()) {

            }
            else if (ui->handmadeKey->isChecked()) {
                SimpleReplacement *srepl = new SimpleReplacement(kvd->readDict());
                if (ui->cypherText->document()->toPlainText().isEmpty()) {
                    QMessageBox::warning(this, tr("Ошибка"), tr("Не введен закрытый текст"));
                    break;
                }
            }
        }
    }
}

```

```

        }
        ui->plainText->setPlainText(srepl->decrypt(ui->cypherText->document()-
>toPlainText()));
        delete srepl;
        break;
    }

}

case 1: {
    if (ui->genKey->isChecked()) {

    }
    else if (ui->fromFile->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        SimpleReplacement *srepl = new SimpleReplacement(kvd->readDict());
        if (ui->cypherText->document()->toPlainText().isEmpty()) {
            QMessageBox::warning(this, tr("Ошибка"), tr("Не введен закрытый текст"));
            break;
        }
        ui->plainText->setPlainText(srepl->decrypt(ui->cypherText->document()-
>toPlainText()));
        delete srepl;
        break;
    }
}

case 2: {
    GammaCypher gcyph(this);
    if (ui->genKey->isChecked()) {
        ui->keyLine->setText(gcyph.getGamma());
        ui->plainText->clear();
        ui->plainText->setPlainText(gcyph.decrypt(ui->cypherText->toPlainText()));
    }
    else if (ui->fromFile->isChecked()) {
        gcyph.setGamma(this->loadKey());
        ui->keyLine->clear();
        ui->keyLine->setText(gcyph.getGamma());
        QString cryptText = gcyph.encrypt(ui->plainText->document()->toPlainText());
        ui->cypherText->setPlainText(cryptText);
    }
    else if (ui->handmadeKey->isChecked()) {
        gcyph.setGamma(ui->keyLine->text());
        qDebug() << "HERE";
        QString plainText = gcyph.decrypt(ui->cypherText->document()->toPlainText());
        ui->plainText->document()->setPlainText(plainText);
    }
    break;
}

case 3:{
    SingleShift sShift(this);
    if (ui->genKey->isChecked()) {

```

```

    }
    else if (ui->handmadeKey->isChecked()) {
        sShift.setKey(ui->keyLine->text());
        ui->plainText->clear();
        ui->plainText->setPlainText(sShift.decrypt(ui->cypherText->toPlainText()));
    }
    break;
}
case 4: {
    DoubleTransposition doubleTrans(this);
    if (ui->genKey->isChecked()){

    }
    else if (ui->handmadeKey->isChecked()) {
        doubleTrans.setKeys(ui->keyLine->text(), ui->key2Line->text());
        ui->plainText->clear();
        ui->plainText->setPlainText(doubleTrans.decrypt(ui->cypherText->toPlainText()));
    }
    break;
}
case 5: {
    RSACrypt rsacrypt(ui->alphBox->currentIndex(), this);
    if (ui->genKey->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        int e, d, n;
        rsaKeyDialog->getPubKeyParams(e, n);
        rsaKeyDialog->getPrivKeyParams(d, n);
        rsacrypt.setKeysParams(e, d, n);

        if (rsaKeyDialog->isKeySet()) {
            ui->plainText->clear();
            ui->plainText->setPlainText(rsacrypt.decrypt(ui->cypherText->toPlainText()));
        }
    }
    break;
}
case 6: {
    Cardano cardano(ui->alphBox->currentIndex(), this);
    ui->plainText->setPlainText(ui->plainText->
    toPlainText().remove(QRegExp(QString::fromUtf8("[^A-ZA-Я\\s]"))));
    if (ui->fromFile->isChecked()) {

    }
    else if (ui->handmadeKey->isChecked()) {
        qDebug() << cardKeyDialog->getMatrix();
        cardano.setKey(cardKeyDialog->getMatrix());
        ui->plainText->clear();
        ui->plainText->setPlainText(cardano.decrypt(ui->cypherText->toPlainText()));
    }
    break;
}

```

```

    }
    case 7:{
        Gronspheld grons(ui->alphBox->currentIndex(), this);
        if (ui->genKey->isChecked()) {
            QString newKey = grons.genNewKey();
            ui->keyLine->setText(newKey);
            ui->plainText->clear();
            ui->plainText->setPlainText(grons.decrypt(ui->cypherText->toPlainText()));
        }
        else if (ui->handmadeKey->isChecked()) {
            grons.setKey(ui->keyLine->text());
            ui->plainText->clear();
            ui->plainText->setPlainText(grons.decrypt(ui->cypherText->toPlainText()));
        }
        break;
    }
}
}
}

```

```

void MainWindow::toggleKeyInputWay(int i) {
    switch (i) {
        case 0:
            ui->genKey->setVisible(false);
            ui->keyLine->setVisible(false);
            ui->keyPair->setVisible(true);
            ui->fromFile->setVisible(true);
            ui->keyFromFileButton->setVisible(true);
            ui->key2Line->setVisible(false);
            ui->alphBox->setVisible(false);
            ui->rsaKeyButton->setVisible(false);
            ui->cardanButton->setVisible(false);
            break;
        case 1:
            ui->genKey->setVisible(false);
            ui->keyLine->setVisible(false);
            ui->keyPair->setVisible(true);
            ui->fromFile->setVisible(false);
            ui->keyFromFileButton->setVisible(false);
            ui->key2Line->setVisible(false);
            ui->alphBox->setVisible(false);
            ui->rsaKeyButton->setVisible(false);
            ui->cardanButton->setVisible(false);
            break;
        case 2:
            ui->genKey->setVisible(true);
            ui->keyLine->setVisible(true);
            ui->keyPair->setVisible(false);
            ui->fromFile->setVisible(true);
            ui->keyFromFileButton->setVisible(true);
            ui->key2Line->setVisible(false);
            ui->alphBox->setVisible(false);

```

```

    ui->keyLine->setValidator(validatorList->at(0));
    ui->rsaKeyButton->setVisible(false);
    ui->cardanButton->setVisible(false);
    break;
case 3:
    ui->genKey->setVisible(false);
    ui->keyPair->setVisible(false);
    ui->keyLine->setVisible(true);
    ui->fromFile->setVisible(false);
    ui->keyFromFileButton->setVisible(false);
    ui->key2Line->setVisible(false);
    ui->alphBox->setVisible(false);
    ui->keyLine->setValidator(validatorList->at(1));
    ui->rsaKeyButton->setVisible(false);
    ui->cardanButton->setVisible(false);
    break;
case 4:
    ui->genKey->setVisible(false);
    ui->keyLine->setVisible(true);
    ui->key2Line->setVisible(true);
    ui->fromFile->setVisible(false);
    ui->keyFromFileButton->setVisible(false);
    ui->alphBox->setVisible(false);
    ui->rsaKeyButton->setVisible(false);
    ui->cardanButton->setVisible(false);
    break;
case 5:
    ui->genKey->setVisible(true);
    ui->fromFile->setVisible(true);
    ui->keyFromFileButton->setVisible(true);
    ui->rsaKeyButton->setVisible(true);
    ui->keyLine->setVisible(false);
    ui->key2Line->setVisible(false);
    ui->keyPair->setVisible(false);
    ui->cardanButton->setVisible(false);
    break;
case 6:
    ui->fromFile->setVisible(false);
    ui->keyFromFileButton->setVisible(false);
    ui->keyLine->setVisible(false);
    ui->alphBox->setVisible(true);
    ui->genKey->setVisible(false);
    ui->rsaKeyButton->setVisible(false);
    ui->cardanButton->setVisible(true);
    ui->keyPair->setVisible(false);
    break;
case 7:
    ui->alphBox->setVisible(true);
    ui->keyLine->setValidator(validatorList->at(2));
    ui->key2Line->setVisible(false);
    ui->fromFile->setVisible(true);
    ui->keyFromFileButton->setVisible(true);

```

```

        ui->keyPair->setVisible(false);
        ui->keyLine->setVisible(true);
        ui->genKey->setVisible(true);
        ui->rsaKeyButton->setVisible(false);
        ui->cardanButton->setVisible(false);
        break;
    }
}

void MainWindow::convertText(QCheckBox *cbox, QPlainTextEdit *pte)
{
    if (cbox->isChecked()) {
        QByteArray array;
        array.append(pte->document()->toPlainText());
        pte->clear();
        pte->setPlainText(QString(array.toHex()));
        ui->doButton->setEnabled(false);
    }
    else {
        QByteArray array;
        array.append(pte->document()->toPlainText());
        pte->clear();
        pte->setPlainText(QString(QByteArray::fromHex(array)));
        ui->doButton->setEnabled(true);
    }
}

void MainWindow::factorize()
{
    if (ui->factorizeNumberLine->text().isEmpty()) {
        QMessageBox::warning(this, QString("Ошибка"), QString("Не введено число"));
        return;
    }
    QString tmp;
    for (auto i : Factorization::factorize(ui->factorizeNumberLine->text().toStdString()))
        tmp += QString::fromStdString(bigIntegerToString(i)) + QString(" ");
    ui->factorizeResultText->setPlainText(tmp);
}

void MainWindow::brute()
{
    switch (ui->bruteCyphersBox->currentIndex()) {
        case 0: {
            ui->bruteTable->clear();
            ui->bruteTable->setColumnCount(3);
            QList<std::tuple<int, int, QString> > data = Bruteforce::bruteRSA(ui->eParamLine-
>text().toInt(),
                                                    ui->nParamLine->text().toInt(),
                                                    ui->bruteLangComboBox->currentIndex(),
                                                    ui->ctextLine->text());
            ui->bruteTable->setRowCount(data.size());
            for (auto i = 0; i < data.size(); i++) {

```

```

        ui->bruteTable->setItem(i, 0, new
QTableWidgetItem(QString::number(std::get<0>(data[i]))));
        ui->bruteTable->setItem(i, 1, new
QTableWidgetItem(QString::number(std::get<1>(data[i]))));
        ui->bruteTable->setItem(i, 2, new QTableWidgetItem(std::get<2>(data[i]));
    }
    break;
}
case 1: {
    ui->bruteTable->clear();
    ui->bruteTable->setColumnCount(2);
    QList<std::tuple<int, QString> > data = Bruteforce::bruteGronsfeld(ui-
>bruteLangComboBox->currentIndex(), 10000, ui->ctextLine->text());
    ui->bruteTable->setRowCount(data.size());
    for (auto i = 0; i < data.size(); i++) {
        ui->bruteTable->setItem(i, 0, new
QTableWidgetItem(QString::number(std::get<0>(data[i]))));
        ui->bruteTable->setItem(i, 1, new QTableWidgetItem(std::get<1>(data[i]));
    }
    break;
}
case 2: {
    ui->bruteTable->clear();
    ui->bruteTable->setColumnCount(2);
    QList<std::tuple<QString, QString> > data = Bruteforce::bruteSimpleReplacement(ui-
>ctextLine->text());
    ui->bruteTable->setRowCount(data.size());
    for (auto i = 0; i < data.size(); i++) {
        ui->bruteTable->setItem(i, 0, new QTableWidgetItem(std::get<0>(data[i]));
        ui->bruteTable->setItem(i, 1, new QTableWidgetItem(std::get<1>(data[i]));
    }
    break;
}
}
}

void MainWindow::toggleBruteFields(int i)
{
    switch (i) {
    case 0:
        ui->eParamLine->setVisible(true);
        ui->nParamLine->setVisible(true);
        ui->bruteLangComboBox->setVisible(true);
        break;
    case 1:
        ui->eParamLine->setVisible(false);
        ui->nParamLine->setVisible(false);
        ui->bruteLangComboBox->setVisible(true);
        break;
    case 2:
        ui->eParamLine->setVisible(false);
        ui->nParamLine->setVisible(false);

```

```

        ui->bruteLangComboBox->setVisible(false);
        break;
    case 3:
        ui->eParamLine->setVisible(false);
        ui->nParamLine->setVisible(false);
        ui->bruteLangComboBox->setVisible(false);
        break;
    }
}

void MainWindow::calculateFrequency()
{
    SymbolFreqAnalysis sfreq(ui->langFreqBox->currentIndex(), this);
    ui->cFreqTable->clear();
    ui->cFreqTable->setColumnCount(2);
    QMap<QString, int> data = sfreq.calculate(ui->ctexForFreq->text());
    ui->cFreqTable->setRowCount(data.keys().size());
    QList<QString> keys = data.keys();
    for (auto i = 0; i < keys.size(); i++) {
        ui->cFreqTable->setItem(i, 0, new QTableWidgetItem(keys[i]));
        ui->cFreqTable->setItem(i, 1, new QTableWidgetItem(QString::number(data[keys[i]])));
    }
}

void MainWindow::calculateBigramm()
{
    ui->probListWidget->clear();
    ui->bigramTableWidget->clear();
    ui->bigramTableWidget->setColumnCount(2);
    ui->bigramTextLine->setText(ui->bigramTextLine->text().toUpper());
    QMap<QString, int> data = BigrammFreqAnalysis::calculate(ui->bigramTextLine->text(), ui->bigrammSize->value());
    ui->bigramTableWidget->setRowCount(data.size());
    QList<QString> keys = data.keys();
    for (auto i = 0; i < keys.size(); i++) {
        ui->bigramTableWidget->setItem(i, 0, new QTableWidgetItem(keys[i]));
        ui->bigramTableWidget->setItem(i, 1, new
QTableWidgetItem(QString::number(data[keys[i]])));
    }
    ui->bigramTableWidget->sortItems(1, Qt::DescendingOrder);
    auto splittedString = ui->bigramTextLine->text().split(";");
    QString propstring("");
    QString tmp("");
    int count = 0;
    for (auto k = 0; k < ui->bigrammSize->value(); k++){
        for (auto i = k; i < ui->bigrammSize->value(); i++) {
            propstring = QString("p(") + QString::number(k) + QString(":") + QString::number(i) +
QString(")");
            for (auto j = 0; j < ui->bigrammSize->value(); j++) {
                tmp = QString(splittedString[j][k]) + QString(splittedString[j][i]);
                count += data[tmp];
            }

```

```

        ui->probListWidget->addItem(propstring + QString::number(count));
        count = 0;
    }
}

void MainWindow::generateKey()
{
    std::random_device rd;
    std::mt19937 mt(rd());

    if(ui->cyphersBox->currentIndex() == 2) {
        QString alpha(" АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
        QString tmp("");
        std::uniform_int_distribution<int> dist(0, alpha.size()-1);

        for (int i=0; i<32; ++i)
            tmp.append(alpha.at(dist(mt)));
        ui->keyLine->clear();
        ui->keyLine->setText(tmp);
    }
    else if (ui->cyphersBox->currentIndex() == 5) {
        std::uniform_int_distribution<int> dist(0, rsaKeyDialog->getPrimesList().size()-1);
        rsaKeyDialog->setPrimes(rsaKeyDialog->getPrimesList().at(dist(mt)),
                                rsaKeyDialog->getPrimesList().at(dist(mt)));
    }
    else if (ui->cyphersBox->currentIndex() == 7) {
        std::uniform_int_distribution<int> dist(1, 10000);
        ui->keyLine->clear();
        ui->keyLine->setText(QString::number(dist(mt)));
    }
}

void MainWindow::loadKeyFromFile()
{
    QString filename = QFileDialog::getOpenFileName(this, tr("Выберите файл"),
                                                    "/home",
                                                    tr("Text (*.txt);; Any files (*)"));

    QFile file(filename);
    QStringList key;
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        QMessageBox::warning(this, tr("Ошибка"), tr("Не удастся открыть файл"),
                              QMessageBox::Ok);
        return;
    }
    else {
        QTextStream in(&file);
        while (!in.atEnd())
            key.append(in.readLine());
        file.close();
    }
    if (ui->cyphersBox->currentIndex() == 1) {

```

```

    QMap<QString, QString> map;
    foreach (auto i, key) {
        auto tmp = i.split("\t");
        map[tmp[0]] = tmp[1];
    }

    kvd->setCustomDict(&map);
}
else if (ui->cyphersBox->currentIndex() == 2) {
    key[0].toUpper();
    key[0].remove(QRegExp(QString::fromUtf8("[^A-Я\\s]")));
    key[0].truncate(32);
    ui->keyLine->setText(key[0]);
}
else if (ui->cyphersBox->currentIndex() == 5) {
    key[0].remove(QRegExp(QString::fromUtf8("[^\\d\\s]")));
    auto primes = key[0].split(" ");
    rsaKeyDialog->setPrimes(primes[0].toInt(), primes[1].toInt());
}
else if (ui->cyphersBox->currentIndex() == 7) {
    key[0].remove(QRegExp(QString::fromUtf8("[^\\d]")));
    key[0].truncate(4);
    ui->keyLine->setText(key[0]);
}
}

QString MainWindow::loadKey() {
    QString fileName = QFileDialog::getOpenFileName(this, tr("Выберите файл"),
                                                    "/home",
                                                    tr("Text (*.txt);; Any files (*)"));

    QFile file(fileName);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        QMessageBox::warning(this, tr("Ошибка"), tr("Не удастся открыть файл"),
        QMessageBox::Ok);
        return QString("");
    }
    else {
        QString text;
        QTextStream in(&file);
        while (!in.atEnd())
            text.append(in.readLine());
        file.close();

        if (text.length() > 5000)
            text.truncate(5000);
        text = text.toUpper();
        text.remove(QRegExp(QString::fromUtf8("[^A-ZА-Я\\s]")));
        return text;
    }
}

```

```

#include "rsacrypt.h"
#include <QDebug>

RSACrypt::RSACrypt(int alph, QObject *parent) : QObject(parent)
{
    switch (alph) {
    case 0:
        alpha = new QString(" ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        break;
    case 1:
        alpha = new QString(" АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
        break;
    }
}

QString RSACrypt::encrypt(QString ptext)
{
    QString ctext;
    for (auto i : ptext){
        ctext += QString::number((power(alpha->indexOf(i), e) % n).toUnsignedInt());
        ctext += " ";
    }
    ctext.truncate(ctext.size()-1);
    return ctext;
}

QString RSACrypt::decrypt(QString ctext)
{
    QString ptext;
    QStringList ctextList = ctext.split(' ', QString::SkipEmptyParts);
    for (auto i : ctextList) {
        ptext += alpha->at((power(i.toInt(), d) % n).toUnsignedInt() % alpha->size());
    }
    return ptext;
}

void RSACrypt::setKeysParams(int eParam, int dParam, int nParam)
{
    e = eParam;
    d = dParam;
    n = nParam;
}

BigUnsigned RSACrypt::power(BigUnsigned x, BigUnsigned y)
{
    BigUnsigned temp;
    if (y == 0)
        return 1;

    temp = power(x, y / 2);
    if ((y % 2) == 0)
        return temp * temp;

```

```

        else
            return x * temp * temp;}
#include "rsakeydialog.h"
#include "ui_rsakeydialog.h"
#include <algorithm>
#include <QMessageBox>
#include <QDebug>

RSAKeyDialog::RSAKeyDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::RSAKeyDialog)
{
    ui->setupUi(this);
    ui->pLine->setValidator(new QRegExpValidator(QRegExp("[\\d]{1,2}"), this));
    ui->qLine->setValidator(new QRegExpValidator(QRegExp("[\\d]{1,2}"), this));
    QObject::connect(ui->calcButton, SIGNAL(clicked()), this, SLOT(calculate()));
    QObject::connect(ui->cancelButton, &QPushButton::clicked, [=] () {
        p = q = n = phi = e = d = 0;
        this->close();
    });
    QObject::connect(ui->okButton, SIGNAL(clicked()), this, SLOT(close()));
}

RSAKeyDialog::~RSAKeyDialog()
{
    delete ui;
}

void RSAKeyDialog::getPubKeyParams(int &ee, int &nn)
{
    ee = e;
    nn = n;
}

void RSAKeyDialog::getPrivKeyParams(int &dd, int &nn)
{
    dd = d;
    nn = n;
}

bool RSAKeyDialog::isKeySet()
{
    return setKeyFlag;
}

QList<int> RSAKeyDialog::getPrimesList() const
{
    return primes;
}

void RSAKeyDialog::setPrimes(int pParam, int qParam)
{

```

```

    ui->pLine->setText(QString::number(pParam));
    ui->qLine->setText(QString::number(qParam));
    while (ui->qLine->text() == ui->pLine->text())
        ui->qLine->setText(QString::number(qParam));
    this->calculate();
}

void RSAKeyDialog::calculate()
{
    if (ui->pLine->text().isEmpty() || ui->qLine->text().isEmpty()) {
        QMessageBox::warning(this, tr("Ошибка"), tr("Не указано значение p или q"),
        QMessageBox::Ok);
        return;
    }
    if (!isPrimal(ui->pLine->text().toInt()) || !isPrimal(ui->qLine->text().toInt())) {
        QMessageBox::warning(this, tr("Ошибка"), tr("p или q не является простым числом"),
        QMessageBox::Ok);
        return;
    }
    n = ui->pLine->text().toInt() * ui->qLine->text().toInt();
    phi = (ui->pLine->text().toInt() - 1) * (ui->qLine->text().toInt() - 1);
    for (auto i = 2; i < 100; i++) {
        if (std::__gcd(phi, i) == 1) {
            e = i;
            break;
        }
    }
    for (auto i = phi; i > 0; i--) {
        if ((i * e) % phi == 1) {
            d = i;
            break;
        }
    }
    ui->nLabel->setText(QString::number(n));
    ui->eLabel->setText(QString::number(e));
    ui->nnLabel->setText(QString::number(n));
    ui->dLabel->setText(QString::number(d));
    setKeyFlag = true;
}

bool RSAKeyDialog::isPrimal(int n)
{
    if (primes.indexOf(n) == -1)
        return false;
    return true;}

#include "simplereplacement.h"

SimpleReplacement::SimpleReplacement(QMap<QString, QString> dictionary, QObject
*parent) : QObject(parent)
{
    dict = new QMap<QString, QString>(dictionary);

```

```

}

QString SimpleReplacement::encrypt(const QString plainText)
{
    QString cryptText("");
    for (auto i : plainText) {
        cryptText.append(dict->value(i));
    }
    return cryptText;
}

QString SimpleReplacement::decrypt(const QString &cryptText) const
{
    QString plainText("");
    for (auto i : cryptText) {
        plainText.append(dict->key(i));
    }
    return plainText;
}

void SimpleReplacement::setKey(QMap<QString, QString> &dictMap)
{
    dict = &dictMap;
}

SimpleReplacement::~SimpleReplacement()
{}
#include "singleshift.h"
#include <list>
#include <cmath>
using namespace std;

SingleShift::SingleShift(QObject *parent) : QObject(parent)
{

}

bool compare(const SingleShift::KeyValuePair &first, const SingleShift::KeyValuePair
&second) {
    return first.Value < second.Value;
}

QChar** Create2DArray(int rowCount, int colCount) {
    QChar** arr = new QChar*[rowCount];

    for (int i = 0; i < rowCount; ++i)
        arr[i] = new QChar[colCount];

    return arr;
}

```

```

QString SingleShift::PadRight(QString str, int max, QChar padChar) {
    int strLen = str.length();

    if (strLen < max) {
        int padLen = max - strLen;
        for (int i = 0; i < padLen; ++i) str += padChar;
    }

    return str;
}

int* SingleShift::GetShiftIndexes(QString key)
{
    int keyLength = key.length();
    int* indexes = new int[keyLength];
    list<KeyValuePair> sortedKey;
    int i;

    for (i = 0; i < keyLength; ++i)
        sortedKey.push_back(KeyValuePair{ i, key[i] });

    sortedKey.sort(compare);
    i = 0;

    for (auto pair : sortedKey) {
        indexes[pair.Key] = i;
        ++i;
    }

    return indexes;
}

QString SingleShift::encrypt(QString input, QChar padChar)
{
    QString output = "";
    int totalChars = input.size();
    int keyLength = key.size();
    input = (totalChars % keyLength == 0) ? input : PadRight(input, totalChars -
(totalChars % keyLength) + keyLength, padChar);
    totalChars = input.length();
    int totalColumns = keyLength;
    int totalRows = (int)ceil((double)totalChars / totalColumns);
    QChar** rowChars = Create2DArray(totalRows, totalColumns);
    QChar** colChars = Create2DArray(totalColumns, totalRows);
    QChar** sortedColChars = Create2DArray(totalColumns, totalRows);
    int currentRow, currentColumn, i, j;
    int* shiftIndexes = GetShiftIndexes(key);

    for (i = 0; i < totalChars; ++i)
    {
        currentRow = i / totalColumns;
        currentColumn = i % totalColumns;

```

```

        rowChars[currentRow][currentColumn] = input[i];
    }

    for (i = 0; i < totalRows; ++i)
        for (j = 0; j < totalColumns; ++j)
            colChars[j][i] = rowChars[i][j];

    for (i = 0; i < totalColumns; ++i)
        for (j = 0; j < totalRows; ++j)
            sortedColChars[shiftIndexes[i]][j] = colChars[i][j];

    for (i = 0; i < totalChars; ++i)
    {
        currentRow = i / totalRows;
        currentColumn = i % totalRows;
        output += sortedColChars[currentRow][currentColumn];
    }

    return output;
}

QString SingleShift::decrypt(QString input)
{
    QString output;
    int keyLength = key.length();
    int totalChars = input.length();
    int totalColumns = (int)ceil((double)totalChars / keyLength);
    int totalRows = keyLength;
    QChar** rowChars = Create2DArray(totalRows, totalColumns);
    QChar** colChars = Create2DArray(totalColumns, totalRows);
    QChar** unsortedColChars = Create2DArray(totalColumns, totalRows);
    int currentRow, currentColumn, i, j;
    int* shiftIndexes = GetShiftIndexes(key);
    for (i = 0; i < totalChars; ++i)
    {
        currentRow = i / totalColumns;
        currentColumn = i % totalColumns;
        rowChars[currentRow][currentColumn] = input[i];
    }

    for (i = 0; i < totalRows; ++i)
        for (j = 0; j < totalColumns; ++j)
            colChars[j][i] = rowChars[i][j];

    for (i = 0; i < totalColumns; ++i)
        for (j = 0; j < totalRows; ++j)
            unsortedColChars[i][j] = colChars[i][shiftIndexes[j]];

    for (i = 0; i < totalChars; ++i)
    {
        currentRow = i / totalRows;
        currentColumn = i % totalRows;

```

```

        output += unsortedColChars[currentRow][currentColumn];
    }

    return output;
}

SingleShift::~~SingleShift()
{

}

void SingleShift::setKey(QString k)
{
    key = k;
}

#include "symbolfreqanalysis.h"

SymbolFreqAnalysis::SymbolFreqAnalysis(int alp, QObject *parent) : QObject(parent)
{
    switch (alp) {
    case 0:
        alpha = QString(" ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        break;
    case 1:
        alpha = QString(" АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ");
        break;
    }
    for (auto i = 0; i < alpha.size(); i++)
        freq.insert(alpha.mid(i, 1), 0);
}

 QMap<QString, int> SymbolFreqAnalysis::calculate(const QString &ctext)
{
    for (auto i = 0; i < ctext.size(); i++)
        if (alpha.contains(ctext.mid(i, 1)))
            freq[ctext.mid(i, 1)]++;
    return freq;
}

 QMap<QString, int> SymbolFreqAnalysis::getFreqrate()
{
    return freq;
}

```