МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

**по дисциплине: «Программирование»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-424», «АВТФ»,  *Амплеев Егор Сергеевич*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#__RefHeading___Toc1861572_3922301009)

[1. Теоретические аспекты реализации методов шифрования 4](#__RefHeading___Toc1861574_3922301009)

[1.1 Алгоритм работы шифра Бэкон. 4](#__RefHeading___Toc1919123_3922301009)

[1.2 Алгоритм работы шифра Трисемуса. 5](#__RefHeading___Toc1931212_3922301009)

[1.3 Алгоритм работы шифра Нигилистов. 6](#__RefHeading___Toc1942893_3922301009)

[2. Практическая часть 8](#__RefHeading___Toc1958340_3922301009)

[2.1 Техническое задание 8](#__RefHeading___Toc1958342_3922301009)

[2.2 Характеристика задачи 8](#__RefHeading___Toc1958344_3922301009)

[2.3 Алгоритм решения 9](#__RefHeading___Toc1958346_3922301009)

[2.4 Руководство пользователя 11](#__RefHeading___Toc1958348_3922301009)

[2.5 Руководство системного программиста 15](#__RefHeading___Toc1958350_3922301009)

[2.6 Контрольный пример 17](#__RefHeading___Toc1958352_3922301009)

[ЗАКЛЮЧЕНИЕ 38](#__RefHeading___Toc1963514_3922301009)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#__RefHeading___Toc1861578_3922301009)

[ПРИЛОЖЕНИЕ A 40](#__RefHeading___Toc1861580_3922301009)

[ПРИЛОЖЕНИЕ B 41](#__RefHeading___Toc1861582_3922301009)

# **ВВЕДЕНИЕ**

В современном мире защита данных становится критически важной. В данной работе исследуются три классических алгоритма шифрования, которые демонстрируют фундаментальные принципы криптографии.

Актуальность исследования обусловлена необходимостью понимания базовых механизмов защиты информации, которые лежат в основе современных криптографических систем.

Цель работы — разработка программы, реализующей три алгоритма шифрования с удобным интерфейсом для шифрования и дешифрования данных.

Основные задачи:

Изучение принципов работы алгоритмов.

Программная реализация методов на C++.

Создание консольного интерфейса для выбора алгоритма, операции и источника данных.

Тестирование и анализ эффективности каждого метода.

Ожидаемые результаты:

Универсальная программа для работы с разными алгоритмами шифрования.

Практическое применение в учебных и исследовательских целях.

Сравнение методов по скорости работы и уровню защиты.

Эффективность работы подтверждается успешной реализацией всех алгоритмов, удобством использования и возможностью дальнейшего расширения функционала.

# 1. Теоретические аспекты реализации методов шифрования

## **1.1 Алгоритм работы шифра Бэкон.**

Френсис Бэкон был знаком с типографией не понаслышке, в начале своей карьеры он выполнял дипломатические поручения в английском посольстве при французском дворе. По его словам, свой шифр изобрел, когда ему было семнадцать лет, однако впервые его упоминает лишь через двадцать семь лет в работе «О преумножении наук». Хотя нет оснований сомневаться в словах Бэкона, но в 1586 г., опередив англичанина, французский ученый Блез де Виженер издает «Трактат о Цифрах и Тайнописи», где описывает тот же самый шифр. Возможно, они оба заимствовали у более ранних криптологов свои шифры. Несмотря на это именно Бэкон первым обратил внимание на то, что благодаря этому изобретению люди смогут передавать свои мысли на любое расстояние с помощью объектов, способных выражать хотя бы два различия. В своей книге «О достоинстве и приумножении наук» он выделяет несколько качеств, которыми обладают лучшие шифры. Одно из них – скрытность. Однако как оказалось этого можно добиться, с помощью противоположной стратегии, нарочно представляя информацию в явном секретном виде, чтобы с помощью этого утаить основной метод шифрования. Именно на этом и основывается шифр Бэкона.

Чтобы реализовать данный шифр для начала нужно составить два алфавита, в которые входят набор букв и знаков. Один алфавит используется для записи секретного сообщения, а другой для текста несекретного характера. Далее оба сообщения вкладываются друг в друга так, что их буквы перемешиваются. В итоге полученное сообщение может быть расшифровано тем, кто знает оба алфавита. Если же информация будет перехвачена, а отправитель или получатель задержаны, он может назвать алфавит, который шифрует не значимый текст, а символы, которыми написано настоящее сообщение объявить незначащими, т. е. введенными для усложнения расшифровки письма. Также следует отметить, что, используя этот метод, следует использовать почти не отличимые шрифты, чтобы разницу смог заметить только заранее осведомленный человек.

Одним из преимуществом данного метода является возможность представлять маскирующую информацию на одном языке, а секретную на другом. Однако этот шифр имеет низкую скорость передачи данных, т. к. для кодировки одной буквы требуется 5 символов.

К сожалению данный шифр был разработан давно и поэтому чтобы реализовать его в современном мире нужно работать не с символами, а с байтами. Таким образом секретное сообщение принимает вид S={s1,s2,…,sn}, где n — длина сообщения в байтах, si-й байт сообщения (например, символ ASCII); отвлекающий текст — D={d1,d2,…,dm}, где m — длина отвлекающего текста в байтах, dk — k-й байт отвлекающего текста. Далее начинается процесс шифрования. Каждый байт si преобразуется в 8 бит: si =b7 b6…b0 Для каждого бита bj выбирается байт dk из D: dk′ {dk∣0x80, если bj=1, dk & 0x7F, если bj=0}. И получаем результат: R={d1′,d2′,…,d8n′}. Дешифровка:  
из каждого dk′ извлекается бит bj=(dk′≫7)&1, затем биты группируются в байты si.

## **1.2 Алгоритм работы шифра Трисемуса.**

Иоганн Тритемий – немецкий монах, который приобрел данный шифр. В 1508 г. он написал печатную работу «Полиграфия», где впервые описал алгоритм применения шифрующих таблиц, заполненных алфавитом в случайном порядке. Тритемий разрабатывал этот метод не только для защиты информации, но и для своих мистических исследований, из-за чего его обвиняли в колдовстве. По сравнению с другими многоалфавитными шифрами, данный способ стал более надежным в результате применения разных алфавитов для мест в тексте. Подход Третимия впервые предложил механический способ реализации сложной замены, а также стал переходным звеном между простыми и сложными криптосистемами.

Описанный шифр использует таблицу для записи букв алфавита и ключевое слово. В начало таблицы записываются уникальные символы предоставленной фразы. Затем добавляются остальные символы, исключая дубликаты. Далее начинается процесс шифрования. Ищется символ в таблице и заменяется на тот, который расположен под ним в том же столбце. Если же это последняя строка, то берется самый верхний столбец. Процесс дешифрования аналогичен только сдвиг происходит не вниз, а вверх.

Благодаря ключевой фразе появляется возможность использования разных таблиц для шифрования, которые заполняются символами в случайном порядке, также упрощается процесс шифрования и дешифрования, что делает алгоритм простым и понятным. Ключом может быть любое слово, что позволяет его изменять. Данный метод может использоваться без компьютера, а это важно, потому что он был создан до появления ЭВМ. Однако по современным меркам шифр ненадежный, его легко преодолеть даже без знания ключа. Если злоумышленник знает часть исходного текста и соответствующий шифртекст, он может восстановить таблицу или ключ, т. к. сдвиги в таблице предсказуемы. Для больших объемов требуется и огромная таблица.

Чтобы зашифровать бинарный файл строится T — таблица 16×16, содержащая все 256 уникальных байт (от 0 до 255). Ключ K задает начальный порядок строк. И затем начинается процесс шифрования. Для байта c с позицией (i,j) в таблице: с′=T[(i+1)mod16][j], где i = c/16 (строка), j=c mod 16 (столбец). Для дешифрование c=T[(i−1) mod 16][j].

## **1.3 Алгоритм работы шифра Нигилистов.**

Зародился данный метод в подпольной России 1870-х годов, революционеры под натиском царской власти были вынуждены изобретать новые конспиративные методы. Этот цифровой шифр стал не просто инструментом, а символом этой эпохи. В кружке чайковцев была предложена идея заменить буквы цифрами. С одной стороны, что может быть проще? Но именно из-за этой простоты шифр стал стремительно распространяться по всем революционным кружкам России. Сначала жандармы считали, что это просто бухгалтерские записи, пока не догадались сопоставить частоту повторяющихся комбинаций цифр, с частотой букв в русском алфавите. Именно с помощью этого шифра происходила координация подготовки покушения на Александра II. Когда «Народная воля» была разгромлена этот метод не исчез, а переродился. Эссеры усложнили его, цифры стали обозначать не буквы, а слова из книг.

Нигилисты для нумерации клеток квадрата использовали секретную фразу, а каждой букве присваивался двузначный код, в соответствии с ее номером в алфавите. Текст, который требуется зашифровать разбивается на буквы, каждая буква заменяется на соответствующий ей двухзначный код. Цифры записываются слитно или группируются. Для большей безопасности можно добавлять случайные цифры между группами или же переставлять по определенному правилу.

Главным плюсом этого метода является проста, он не требует сложных вычислений, любой участник подполья мог быстро научится. Шифр не вызывает подозрений, выглядит как бухгалтерские записи или случайные числа. Данный способ может быть модифицирован, добавляя незначащие символы или используя перестановки, поэтому разные группы революционеров использовали свой собственный способ. Во времена нигилистов взломать его было трудно, не зная ключа. Однако ключ следует часто менять чтобы шифр было труднее вскрыть. Со временем развития криптоанализа данный метод становился все более ненадёжным. Длинные тексты требовали много времени для шифрования, увеличивался риск ошибок, а также размер сообщения увеличивался в два раза.

Чтобы зашифровать бинарный файл этим способом, приходится кодировать символы. Каждый байт c представляется парой чисел: Code(c)=(q,r),где q=⌊ с/16⌋+1, r=(cmod16)+1. Затем начинается процесс шифрования. Для сообщения M={m1,m2,…,mn}и ключа *K*={*k1*,*k2*,…,*kp*}: ∀i∈[1,n]: (qi,ri) = Code(mi), (qk,rk) = Code(k(i mod p)+1), (qi′,ri′)=(qi+qk, ri+rk).

# **2. Практическая часть**

## **2.1 Техническое задание**

2.1.1 Постановка задачи

Необходимо разработать программу, которая должна шифровать и

расшифровать исходные данные одним из представленных алгоритмов. Программа должна выполнять следующие основные действия:

- обеспечить процедуру проверки пользовательских данных путем

авторизации в системе;

- предусмотреть ввод исходного текста с клавиатуры или загрузку данных из файла;

- реализовать шифрование данных и демонстрацию полученных

результатов;

- реализовать дешифрование (расшифрование) данных и демонстрацию

полученных результатов (консоль, файл).

Методы шифрования реализовать в виде отдельных функций/модулей.

## **2.2 Характеристика задачи**

2.2.1 Программа «» предназначена для автоматизации шифрования и расшифрования исходных данных авторизованного в системе пользователя.

2.2.2 Программа используется пользователем для защиты персональной информации.

2.2.3 Периодичность решения задачи по запросу пользователя.

2.2.4 Прекращение автоматизированного решения задачи происходит при отключении источника электропитания ЭВМ.

2.2.5 Связь с другими задачами отсутствует.

2.2.6 Специальных ограничений на временные характеристики решения задачи не налагается.

2.2.7 Специальных требований на уровень подготовки пользователя не налагается. Но лицо, работающее с программой, должно иметь минимальное представление о компьютере (знание необходимых операций).

## **2.3 Алгоритм решения**

1. Запустить «Encryption Algorithm RGR»;

2. Вывод меню:

«Выберите шифр:»

«Нажмите 1 для выбора шифра Бэкона»;

«Нажмите 2 для выбора шифр Нигилиста»;

«Нажмите 3 для выбора шифра Трисемуса»;

«Нажмите 4 для выхода»;

2.1 Если выбран пункт - «Нажмите 1 для выбора шифра Бэкона»;

2.1.1 Вывод меню:

«Выберите действие:»

«Нажмите 1 для шифрования»;

«Нажмите 2 для дешифрования»;

2.1.1.1 Если выбран пункт - «Нажмите 1 для шифрования»;

2.1.1.2 Вывод: «Введите секретные данные»;

2.1.1.3 Ввод секретных данный;

2.1.1.4 Вывод: «Введите отвлекающий текст»;

2.1.1.5 Ввод отвлекающего текста;

2.1.1.6 Шифрование текста шифром Бэкона;

2.1.1.7 Вывод зашифрованного текста в bacon\_encrypted.txt;

2.1.1.8 Переход на п.2

2.1.2.1 Если выбран пункт - «Нажмите 2 для дешифрования»;

2.1.2.2 Вывод: «Введите зашифрованный текст»;

2.1.2.3 Ввод зашифрованного текста;

2.1.2.4 Расшифрование текста шифром Бэкона;

2.1.2.5 Вывод расшифрованного текста в bacon\_decrypted.txt;

2.1.1.8 Переход на п.2;

2.2 Если выбран пункт - «Нажмите 2 для выбора шифра Нигилиста»;

2.2.1 Вывод меню:

«Выберите действие:»

«Нажмите 1 для шифрования»;

«Нажмите 2 для дешифрования»;

2.2.1.1 Если выбран пункт - «Нажмите 1 для шифрования»;

2.2.1.2 Вывод: «Введите ключ»;

2.2.1.3 Ввод ключа;

2.2.1.4 Вывод: «Введите сообщение»;

2.2.1.5 Ввод сообщения;

2.2.1.6 Шифрование текста шифром Нигилиста;

2.2.1.7 Вывод зашифрованного текста в nigilist\_encrypted.txt;

2.2.1.8 Переход на п.2

2.2.2.1 Если выбран пункт - «Нажмите 2 для дешифрования»;

2.2.2.2 Вывод: «Введите зашифрованный текст»;

2.2.2.3 Ввод зашифрованного текста;

2.2.2.4 Расшифрование текста шифром Нигилиста;

2.2.2.5 Вывод расшифрованного текста в nigilist \_decrypted.txt;

2.2.1.8 Переход на п.2

2.3 Если выбран пункт - «Нажмите 3 для выбора шифра Трисемуса»;

2.3.1 Вывод меню:

«Выберите действие:»

«Нажмите 1 для шифрования»;

«Нажмите 2 для дешифрования»;

2.3.1.1 Если выбран пункт - «Нажмите 1 для шифрования»;

2.3.1.2 Вывод: «Введите ключ»;

2.3.1.3 Ввод ключа;

2.3.1.4 Вывод: «Введите сообщение»;

2.3.1.5 Ввод сообщения;

2.3.1.6 Шифрование текста шифром Трисемуса;

2.3.1.7 Вывод зашифрованного текста в trisemus\_encrypted.txt;

2.3.1.8 Переход на п.2

2.3.2.1 Если выбран пункт - «Нажмите 2 для дешифрования»;

2.3.2.2 Вывод: «Введите зашифрованный текст»;

2.3.2.3 Ввод зашифрованного текста;

2.3.2.4 Расшифрование текста шифром Трисемуса;

2.3.2.5 Вывод расшифрованного текста в trisemus \_decrypted.txt;

2.3.1.8 Переход на п.2

## **2.4 Руководство пользователя**

**2.4.1 Введение**

2.4.1.1 Программа «Encryption Algorithm RGR» предназначена для шифрования и дешифрования данных с использованием трех классических алгоритмов:

* Шифра Бэкона (двоичное скрытие в тексте)
* Шифра Нигилиста (числовое преобразование)
* Шифра Трисемуса (табличный сдвиговый метод)

Применяется для:

* Защиты конфиденциальных текстовых сообщений
* Учебных целей в криптографии

2.4.1.2 Программа предоставляет пользователю следующие возможности:

* Полный доступ к системе шифрования/дешифрования
* Ввод исходных данных для обработки
* Изменение параметров шифрования (выбор алгоритма, ввод ключа)
* Вывод результатов в формате .txt

2.4.1.3 Программа реализована на алгоритмическом языке C++. Работает в любой среде, совместимой с операционными системами Windows (7 и новее), Linux и macOS.

Требования к системе:

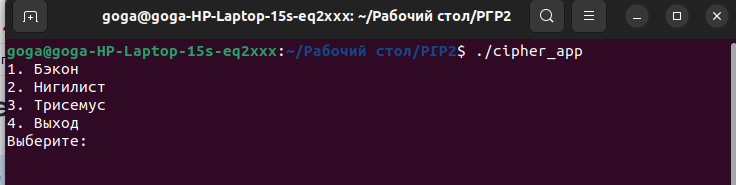
* Дисковой памяти для запуска программы требуется не менее 5 Мб
* Оперативной памяти для нормальной работы программы требуется не менее 50 Мб
* Процессор с архитектурой x86 или x64

**2.3.2 Описание операций**

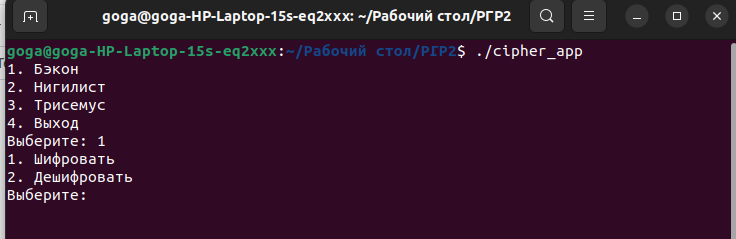
Основные функции программы «Encryption Algorithm RGR» соответствуют созданию

системы для шифрования и дешифрования данных с использованием различных алгоритмов Для удобства пользователя и более легкого изучения системы большинство форм и диалогов имеют идентичный интерфейс. Далее описаны все функции системы, а также формы и диалоги для ввода и вывода информации. Для каждой формы приведены основные компоненты и их назначение.

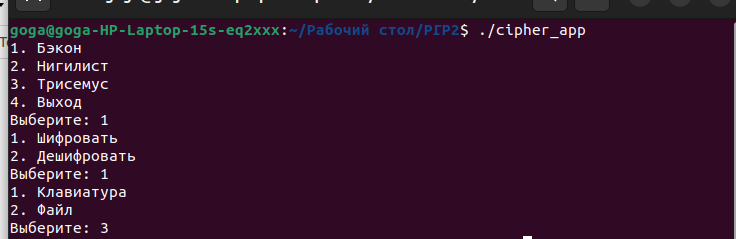
После входа в систему на экран выводится главное окно программы, которое предлагает пользователю выбрать один из трех алгоритмов шифрования (Рисунок 1).

Рисунок 1 — главное окно программы

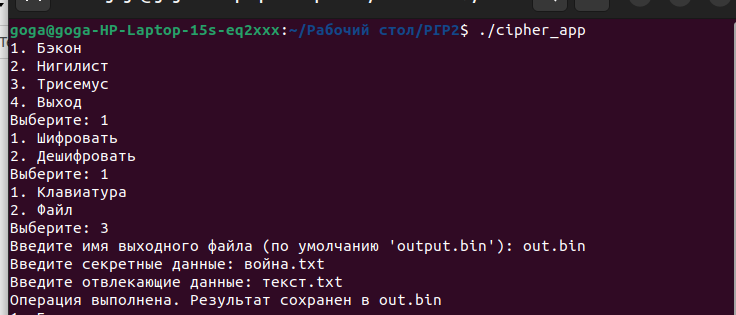
Далее про программа предоставляет выбор шифровать или дешифрвать (Рисунок 2).

Рисунок 2 — выбор процесса шифрования или дешифрования

Затем пользователю нужно решить с чем он будет работать. Он может либо вводить текст с клавиатура либо загружать файлы (Рисунок 3).

Рисунок 3 — выбор вида шифруемых данных

После этого программа предложит ввести данные необходимые для данного типа шифрования или дешифрования (Рисунок 3).

Рисунок 3 — ввод необходимых данных необходимых для шифрования или дешифрования

**2.3.3 Сообщения пользователю**

При работе с программой могут появиться следующие сообщения,

представленные в таблице 2.1.

Таблица 2.1 – Сообщения пользователю

|  |  |
| --- | --- |
| Текст сообщения программы | Ответ пользователя |
| Файл не найден! | Проверить наличие указанного файла в каталоге с программой. |
| Неверный формат данных! | Убедиться, что вводимые данные соответствуют требованиям выбранного алгоритма шифрования. |
| Ключ не может быть пустым! | Ввести корректный ключ шифрования. |
| Ошибка при записи в файл | Проверить доступность файла для записи и наличие свободного места на диске. |

**2.3.4 Аварийные ситуации**

К аварийным ситуациям относятся: нехватка оперативной памяти для создания окна программы. Если исполняемый модуль программы не запускается, либо не выполнены требования условия работы программы, либо один из файлов поврежден, необходимо обратиться к разработчику программы.

## **2.5 Руководство системного программиста**

Программа " Encryption Algorithm RGR " реализована на языке C++. Архитектура программы модульная и состоит из следующих компонентов:

Модули программы

1. cipher.h - главный заголовочный файл, содержит: объявления классов шифров, константы для типов шифров, прототип основной функции ProcessCipher().
2. bacon.h/nihilist.h/trisemus.h - заголовочные файлы для каждого алгоритма шифрования, содержат классы с методами encryptData() и decryptData(), вспомогательные функции преобразования данных.
3. main.cpp содержит функцию main() с логикой работы меню, обработку ввода пользователя.
4. cipher.cpp реализация функций необходимых для корректной работы программы, обработки ошибок.
5. bacon.срр/nihilist.срр/trisemus.срр – реализация алгоритмов шифрования.

Программа содержит ряд сообщений, предназначенных для сигнализации ошибок:

1. Недопустимый выбор – ошибка при вводе несуществующего пункта меню, решение - выбрать значение от 1 до 4
2. Ошибка записи - проблемы с сохранением результатов, решение - проверить доступное место на диске.

Связь между модулями программы

1. Главный управляющий модуль (main.cpp) взаимодействует с cipher.h (получает объявления функций и констант для работы с шифрами).

2. Модуль обработки шифров (cipher.cpp) взаимодействует с main.cpp (получает тип шифра, данные и ключ); bacon.h, nihilist.h, trisemus.h (вызывает соответствующие методы шифрования/дешифрования.

3. Модули алгоритмов (bacon.cpp, nihilist.cpp, trisemus.cpp) взаимодействуют с cipher.cpp (получают данные для обработки и возвращают результат).

4. Заголовочные файлы (\*.h) взаимодействуют со всеми модулями, содержат объявления классов, констант и функций, обеспечивают согласованность типов данных между модулями.

Сообщения системному программисту приведены в таблице 2.2.

Таблица 2.2 – Сообщения системному программисту

|  |  |  |
| --- | --- | --- |
| Текст сообщения | Расшифровка | Решение |
| Failed to open file | Ошибка открытия файла | Проверить путь и права доступа |
| Invalid cipher type | Неверный тип шифра | Убедиться в корректности входных данных |
| Memory allocation failed | Ошибка выделения памяти | Увеличить доступную память |
| Key is too short | Ключ недостаточной длины | Использовать ключ от 8 символов |
| Corrupted data | Поврежденные входные данные | Проверить целостность файла |

## **2.6 Контрольный пример**

2.6.1 Шифр Бэкона

Секретное сообщение: "Амплеев Егор АБс-424"

Секретное сообщение по UTF-8: D0 90 D0 BC D0 BF D0 BB D0 B5 D0 B5 D0 B2 20 D0 95 D0 B3 D0 BE D0 A0 20 D0 90 D0 91 D1 81 2D 34 32 34

Отвлекающий текст: "abcde"

Отвлекающий текст по UTF-8: (61 62 63 64 65)

В итоге получилось 160 бит, значит отвлекающий текст расширяется в 32 раза.

Встраивание битов: если бит 1 → OR 0x80 (установить старший бит),

если бит 0 → AND 0x7F (сбросить старший бит).

Для первых 2 байт секрета (D0 90):

D0 = 11010000 → биты: 1,1,0,1,0,0,0,0

Бит 1 (1): 61 | 0x80 = E1

Бит 2 (1): 62 | 0x80 = E2

Бит 3 (0): 63 & 0x7F = 63

Бит 4 (1): 64 | 0x80 = E4

Бит 5 (0): 65 & 0x7F = 65

Бит 6 (0): 61 & 0x7F = 61

Бит 7 (0): 62 & 0x7F = 62

Бит 8 (0): 63 & 0x7F = 63

90 = 10010000 → биты: 1,0,0,1,0,0,0,0

Бит 1 (1): 64 | 0x80 = E4

Бит 2 (0): 65 & 0x7F = 65

Бит 3 (0): 61 & 0x7F = 61

Бит 4 (1): 62 | 0x80 = E2

Бит 5 (0): 63 & 0x7F = 63

Бит 6 (0): 64 & 0x7F = 64

Бит 7 (0): 65 & 0x7F = 65

Бит 8 (0): 61 & 0x7F = 61

В итоге результат для первых 16 байт:

E1 E2 63 E4 65 61 62 63 E4 65 61 E2 63 64 65 61

Аналогично для всех оставшихся байтов. И окончательное сообщение в зашифрованном виде:

E1 E2 63 E4 65 61 62 63 E4 65 61 E2 63 64 65 61

E1 E2 63 E4 65 61 62 63 E4 65 E2 E4 E4 E4 65 65

E1 E2 63 E4 65 61 62 63 E4 65 E2 E4 E4 E4 E4 E4

E1 E2 63 E4 65 61 62 63 E4 65 E2 E4 E4 65 61 62

E1 E2 63 E4 65 61 62 63 E4 65 E2 E4 E4 65 61 62

E1 E2 63 E4 65 61 62 63 63 64 65 61 62 63 64 65

E1 E2 63 E4 65 61 62 63 63 64 65 61 62 63 64 65

E1 E2 63 E4 65 61 62 63 63 64 65 61 62 63 64 65

E1 E2 63 E4 65 61 62 63 63 64 65 61 62 63 64 65

E1 E2 63 E4 65 61 62 63 63 64 65 61 62 63 64 65

Дешифровка

Для каждого байта проверяем старший бит:

Если 1 → бит 1.

Если 0 → бит 0.

Первые 8 байт (E1 E2 63 E4 65 61 62 63):

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Собираем биты: 11010000 → 0xD0 (первый байт 'А').

Второй байт (90):

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

E2 (11100010) → 1

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 0

61 (01100001) → 0

Биты: 1 0 0 1 0 0 0 0 → 90 (второй байт символа 'А')

Итог: D0 90 → символ 'А'

E1 E2 63 E4 65 61 62 63 | E4 65 E2 E4 E4 E4 65 65

Расшифровка:

Первый байт (D0):

E1 (1), E2 (1), 63 (0), E4 (1), 65 (0), 61 (0), 62 (0), 63 (0)

Биты: 1 1 0 1 0 0 0 0 → D0

Второй байт (BC):

E4 (1), 65 (0), E2 (1), E4 (1), E4 (1), E4 (1), 65 (0), 65 (0)

Биты: 1 0 1 1 1 1 0 0 → BC

Итог: D0 BC → символ 'м'

E1 E2 63 E4 65 61 62 63 | E4 65 E2 E4 E4 E4 E4 E4

Расшифровка:

Первый байт (D0):

Аналогично предыдущим → D0

Второй байт (BF):

E4 (1), 65 (0), E2 (1), E4 (1), E4 (1), E4 (1), E4 (1), E4 (1)

Биты: 1 0 1 1 1 1 1 1 → BF

Итог: D0 BF → символ 'п'

E1 E2 63 E4 65 61 62 63 | E4 65 E2 E4 E4 65 61 62

Расшифровка:

Первый байт (D0) → D0

Второй байт (BB):

E4 (1), 65 (0), E2 (1), E4 (1), E4 (1), 65 (0), 61 (0), 62 (0)

Биты: 1 0 1 1 1 0 0 0 → BB

Итог: D0 BB → символ 'л'

E1 E2 63 E4 65 61 62 63 | E4 65 E2 E4 E4 65 61 62

Расшифровка:

Первый байт (D0) → D0

Второй байт (B5):

E4 (1), 65 (0), E2 (1), E4 (1), E4 (1), 65 (0), 61 (0), 62 (0)

Биты: 1 0 1 1 0 1 0 1 → B5

Итог: D0 B5 → символ 'е'

'в' (D0 B2)

Байт 1: D0 (11010000)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Байт 2: B2 (10110010)

E4 (11100100) → 1

65 (01100101) → 0

E2 (11100010) → 1

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 1

63 (01100011) → 0

Пробел (20)

Байт: 20 (00100000)

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 1

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 0

'Е' (D0 95)

Байт 1: D0 (11010000)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Байт 2: 95 (10010101)

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 1

'г' (D0 B3)

Байт 1: D0 (11010000)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Байт 2: B3 (10110011)

E4 (11100100) → 1

65 (01100101) → 0

E2 (11100010) → 1

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 1

E2 (11100010) → 1

'о' (D0 BE)

Байт 1: D0 (11010000)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Байт 2: BE (10111110)

E4 (11100100) → 1

65 (01100101) → 0

E2 (11100010) → 1

E4 (11100100) → 1

E4 (11100100) → 1

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

'р' (D0 A0)

Байт 1: D0 (11010000)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

Байт 2: A0 (10100000)

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 1

62 (01100010) → 0

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 0

61 (01100001) → 0

Пробел (20)

Аналогично предыдущему пробелу

'А' (D0 90)

Аналогично первому символу

'Б' (D0 91)

Байт 1: D0 (11010000)

Байт 2: 91 (10010001)

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

E2 (11100010) → 1

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 0

E2 (11100010) → 1

'с' (D1 81)

Байт 1: D1 (11010001)

E1 (11100001) → 1

E2 (11100010) → 1

63 (01100011) → 0

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

E2 (11100010) → 1

Байт 2: 81 (10000001)

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 0

63 (01100011) → 0

64 (01100100) → 0

65 (01100101) → 0

E2 (11100010) → 1

-' (2D)

Байт: 2D (00101101)

63 (01100011) → 0

64 (01100100) → 0

E4 (11100100) → 1

65 (01100101) → 0

E2 (11100010) → 1

63 (01100011) → 1

64 (01100100) → 0

E4 (11100100) → 1

'4' (34)

Байт: 34 (00110100)

63 (01100011) → 0

64 (01100100) → 0

E2 (11100010) → 1

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 1

62 (01100010) → 0

63 (01100011) → 0

'2' (32)

Байт: 32 (00110010)

63 (01100011) → 0

64 (01100100) → 0

E2 (11100010) → 1

E4 (11100100) → 1

65 (01100101) → 0

61 (01100001) → 0

62 (01100010) → 1

63 (01100011) → 0

15. '4' (34)

Аналогично предыдущей цифре 4

2.6.2 Шифр Нигилистов

Исходные данные:

Ключ: "ключ" (4 байта в UTF-8: D0 BA D0 BB D1 8E D1 87)

Сообщение: "Амплеев Егор АБс-424" (20 байт в UTF-8)

Каждый байт сообщения разбивается на два числа:

(байт / 16 + 1, байт % 16 + 1)

Ключ разбивается аналогично и повторяется циклически. Числа сообщения и ключа складываются.

Сообщение:

'А' (D0 90):

D0 → (208 / 16 + 1, 208 % 16 + 1) = (14, 1)

90 → (144 / 16 + 1, 144 % 16 + 1) = (10, 1)

'м' (D0 BC):

D0 → (14, 1)

BC → (188 / 16 + 1, 188 % 16 + 1) = (12, 13)

Ключ "ключ" (D0 BA D0 BB D1 8E D1 87):

'к' (D0 BA):

D0 → (14, 1)

BA → (186 / 16 + 1, 186 % 16 + 1) = (12, 11)

'л' (D0 BB):

D0 → (14, 1)

BB → (187 / 16 + 1, 187 % 16 + 1) = (12, 12)

'ю' (D1 8E):

D1 → (209 / 16 + 1, 209 % 16 + 1) = (14, 2)

8E → (142 / 16 + 1, 142 % 16 + 1) = (9, 15)

'ч' (D1 87):

D1 → (14, 2)

87 → (135 / 16 + 1, 135 % 16 + 1) = (9, 8)

Шифрование (сложение пар чисел)

Циклически повторяем ключ для всего сообщения:

‘А’ (D0 90) + ‘к’ (D0 BA):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

90: (10, 1) + л (D0): (12, 11) = (10 + 12, 1 + 11) = (22, 12)

‘м’ (D0 BC) + ‘л’ (D0 BB):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

BC: (12, 13) + л (D0): (12, 12) = (12 + 12, 13 + 12) = (24, 25)

‘п’ (D0 BF) + ‘ю’ (D1 8E):

D0: (14, 1) + к (D1): (14, 2) = (14 + 14, 1 + 2) = (28, 3)

BF: (12, 16) + л (8E): (9, 15) = (12 + 9, 16 + 15) = (21, 31)

‘л’ (D0 BB) + ‘ч’ (D1 87):

D0: (14, 1) + к (D1): (14, 2) = (14 + 14, 1 + 2) = (28, 3)

BB: (12, 12) + л (87): (9, 8) = (12 + 9, 12 + 8) = (21, 20)

‘е’ (D0 B5) + ‘к’ (D0 BA):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

B5: (12, 6) + л (D0): (12, 11) = (12 + 12, 6 + 11) = (24, 17)

‘е’ (D0 B5) + ‘л’ (D0 BB):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

B5: (12, 6) + л (D0): (12, 12) = (12 + 12, 6 + 12) = (24, 18)

‘в’ (D0 B2) + ‘ю’ (D1 8E):

D0: (14, 1) + к (D1): (14, 2) = (14 + 14, 1 + 2) = (28, 3)

B2: (12, 3) + л (8E): (9, 15) = (12 + 9, 3 + 15) = (21, 18)

’ ’ (20) + ‘ч’ (D1 87):

20: (3, 1) + к (D1): (14, 2) = (3 + 14, 1 + 2) = (17, 3)

‘Е’ (45) + ‘к’ (D0 BA):

45: (5, 6) + к (D0): (14, 1) = (5 + 14, 6 + 1) = (19, 7)

‘г’ (D0 B3) + ‘л’ (D0 BB):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

B3: (12, 4) + л (D0): (12, 12) = (12 + 12, 4 + 12) = (24, 16)

‘о’ (D0 BE) + ‘ю’ (D1 8E):

D0: (14, 1) + к (D1): (14, 2) = (14 + 14, 1 + 2) = (28, 3)

BE: (12, 15) + л (8E): (9, 15) = (12 + 9, 15 + 15) = (21, 30)

‘р’ (D1 80) + ‘ч’ (D1 87):

D1: (15, 2) + к (D1): (14, 2) = (15 + 14, 2 + 2) = (29, 4)

80: (9, 1) + л (87): (9, 8) = (9 + 9, 1 + 8) = (18, 9)

’ ’ (20) + ‘к’ (D0 BA):

20: (3, 1) + к (D0): (14, 1) = (3 + 14, 1 + 1) = (17, 2)

‘А’ (D0 90) + ‘л’ (D0 BB):

D0: (14, 1) + к (D0): (14, 1) = (14 + 14, 1 + 1) = (28, 2)

90: (10, 1) + л (D0): (12, 12) = (10 + 12, 1 + 12) = (22, 13)

‘Б’ (D0 91) + ‘ю’ (D1 8E):

D0: (14, 1) + к (D1): (14, 2) = (14 + 14, 1 + 2) = (28, 3)

91: (10, 2) + л (8E): (9, 15) = (10 + 9, 2 + 15) = (19, 17)

‘с’ (D1 81) + ‘ч’ (D1 87):

D1: (15, 2) + к (D1): (14, 2) = (15 + 14, 2 + 2) = (29, 4)

81: (9, 2) + л (87): (9, 8) = (9 + 9, 2 + 8) = (18, 10)

’-’ (2D) + ‘к’ (D0 BA):

2D: (3, 14) + к (D0): (14, 1) = (3 + 14, 14 + 1) = (17, 15)

‘4’ (34) + ‘л’ (D0 BB):

34: (4, 5) + к (D0): (14, 1) = (4 + 14, 5 + 1) = (18, 6)

‘2’ (32) + ‘ю’ (D1 8E):

32: (4, 3) + к (D1): (14, 2) = (4 + 14, 3 + 2) = (18, 5)

‘4’ (34) + ‘ч’ (D1 87):

34: (4, 5) + к (D1): (14, 2) = (4 + 14, 5 + 2) = (18, 7)

Итог

(28, 2) (22, 12) (28, 2) (24, 25) (28, 3) (21, 31) (28, 3) (21, 20) (28, 2) (24, 17) (28, 2) (24, 18) (28, 3) (21, 18) (17, 3) (19, 7) (28, 2) (24, 16) (28, 3) (21, 30) (29, 4) (18, 9) (17, 2) (28, 2) (22, 13) (28, 3) (19, 17) (29, 4) (18, 10) (17, 15) (18, 6) (18, 5) (18, 7)

Дешифрование

Берем очередную пару зашифрованных чисел. Вычитаем из нее соответствующую пару ключа (ключ повторяется циклически). Преобразуем полученные числа обратно в байт по формуле: байт = (первое\_число - 1) \* 16 + (второе\_число — 1). Собираем байты в исходное сообщение.

Дешифровка:

‘А’ (D0 90):

(28, 2) - ‘к’ (D0): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(22, 12) - ‘л’ (BA): (12, 11) = (10, 1)

→ (10 - 1) \* 16 + (1 - 1) = 144 = 90

→ ‘А’ (D0 90)

‘м’ (D0 BC):

(28, 2) - ‘ю’ (BA): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(24, 25) - ‘ч’ (BB): (12, 13) = (12, 12)

→ (12 - 1) \* 16 + (12 - 1) = 188 = BC

→ ‘м’ (D0 BC)

‘п’ (D0 BF):

(28, 3) - ‘к’ (8E): (14, 2) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(21, 31) - ‘л’ (87): (9, 15) = (12, 16)

→ (12 - 1) \* 16 + (16 - 1) = 191 = BF

→ ‘п’ (D0 BF)

‘л’ (D0 BB):

(28, 3) - ‘ю’ (8E): (14, 2) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(21, 20) - ‘ч’ (87): (9, 8) = (12, 12)

→ (12 - 1) \* 16 + (12 - 1) = 187 = BB

→ ‘л’ (D0 BB)

‘е’ (D0 B5):

(28, 2) - ‘к’ (8E): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(24, 17) - ‘л’ (87): (12, 11) = (12, 6)

→ (12 - 1) \* 16 + (6 - 1) = 181 = B5

→ ‘е’ (D0 B5)

‘е’ (D0 B5):

(28, 2) - ‘ю’ (8E): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(24, 18) - ‘ч’ (87): (12, 8) = (12, 6)

→ (12 - 1) \* 16 + (6 - 1) = 181 = B5

→ ‘е’ (D0 B5)

‘в’ (D0 B2):

(28, 3) - ‘к’ (8E): (14, 2) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(21, 18) - ‘л’ (87): (9, 8) = (12, 4)

→ (12 - 1) \* 16 + (4 - 1) = 178 = B2

→ ‘в’ (D0 B2)

’ ’ (20):

(17, 3) - ‘ю’ (8E): (14, 2) = (3, 1)

→ (3 - 1) \* 16 + (1 - 1) = 32 = 20

→ ’ ’ (20)

‘Е’ (45):

(19, 7) - ‘ч’ (87): (14, 2) = (5, 6)

→ (5 - 1) \* 16 + (6 - 1) = 69 = 45

→ ‘E’ (45)

‘г’ (D0 B3):

(28, 2) - ‘к’ (8E): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(24, 16) - ‘л’ (87): (12, 11) = (12, 4)

→ (12 - 1) \* 16 + (4 - 1) = 179 = B3

→ ‘г’ (D0 B3)

‘о’ (D0 BE):

(28, 3) - ‘ю’ (8E): (14, 2) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(21, 30) - ‘ч’ (87): (9, 8) = (12, 15)

→ (12 - 1) \* 16 + (15 - 1) = 190 = BE

→ ‘о’ (D0 BE)

‘р’ (D1 80):

(29, 4) - ‘к’ (8E): (14, 2) = (15, 2)

→ (15 - 1) \* 16 + (2 - 1) = 241 = D1

(18, 9) - ‘л’ (87): (12, 11) = (6, 8)

→ (12 - 1) \* 16 + (12 - 1) = 128 = 80

→ ‘р’ (D1 80)

’ ’ (20):

(17, 2) - ‘ю’ (8E): (14, 2) = (3, 0)

→ (3 - 1) \* 16 + (0 - 1) = 32

→ ’ ‘(20)

‘А’ (D0 90):

(28, 2) - ‘ч’ (87): (14, 1) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(22, 13) - ‘к’ (8E): (12, 11) = (10, 2)

→ (10 - 1) \* 16 + (2 - 1) = 145 = 91

→ ‘A’ (D0 90)

‘Б’ (D0 91):

(28, 3) - ‘л’ (87): (14, 2) = (14, 1)

→ (14 - 1) \* 16 + (1 - 1) = 208 = D0

(19, 17) - ‘ю’ (8E): (9, 15) = (10, 2)

→ (10 - 1) \* 16 + (2 - 1) = 145 = 91

→ ‘Б’ (D0 91)

‘с’ (D1 81):

(29, 4) - ‘ч’ (87): (14, 2) = (15, 2)

→ (15 - 1) \* 16 + (2 - 1) = 241 = D1

(18, 10) - ‘к’ (8E): (9, 8) = (9, 2)

→ (9 - 1) \* 16 + (2 - 1) = 129 = 81

→ ‘с’ (D1 81)

’-’ (2D): \* (17, 15) - ‘ч’ (87): (14, 2) = (3, 14)

→ (3 - 1) \* 16 + (14 - 1) = 45 = 2D

→ ’45’(20)

‘4’ (34): \* (18, 6) - ‘к’ (8E): (14, 2) = (4, 5)

→ (4 - 1) \* 16 + (5 - 1) = 52 = 34

→ ’34’(20)

‘2’ (32): \* (18, 5) - ‘ю’ (8E): (14, 2) = (4, 3)

→ (4 - 1) \* 16 + (3 - 1) = 50 = 32

→ ’32’(20)

‘4’ (34): \* (18, 7) - ‘ч’ (87): (14, 2) = (4, 5)

→ (4 - 1) \* 16 + (5 - 1) = 52 = 34

→ ’34’(20)

Итог: Амплеев Егор Абс-424 — верно

2.6.2 Шифр Трисемуса

Подготовка таблицы:

Ключ "ключ" добавляется первым (к, л, ю, ч). Затем все остальные символы (0-255) в порядке возрастания. Таблица строится динамически, размер зависит от количества уникальных символов.

Для примера возьмем уменьшенную копию таблицы. (В реальном коде таблица будет 16×16 для всех 256 байтов.)

к л ю ч А Б В Г

Д Е Ё Ж З И Й М

Н О П Р С Т У Ф

Х Ц Ш Щ Ъ Ы Ь Э

Я - 0 1 2 3 4 5

6 7 8 9 (пробел)

Шифрование

А (0,4) Д (1,4)

м (1,7) Н (2,7)

п (2,2) Х (3,2)

л (0,1) Д (1,1)

е (1,1) Н (2,1)

е (1,1) Н (2,1)

в (0,6) Д (1,6)

пробел (5,8) к (0,0)

Е (1,1) Н (2,1)

г (0,7) Д (1,7)

о (2,1) Х (3,1)

р (2,3) Х (3,3)

пробел (5,8) к (0,0)

А (0,4) Д (1,4)

Б (0,5) Д (1,5)

с (2,4) Х (3,4)

- (4,1) 6 (5,1)

4 (4,6) (пробел)

2 (4,4) 6 (5,4)

4 (4,6) (пробел)

Дешифрование

Д (1,4) А (0,4)

Н (2,7) М (1,7)

Х (3,2) П (2,2)

Д (1,1) Л (0,1)

Н (2,1) Е (1,1)

Н (2,1) Е (1,1)

Д (1,6) В (0,6)

к (0,0) (пробел)

Н (2,1) Е (1,1)

Д (1,7) Г (0,7)

Х (3,1) О (2,1)

Х (3,3) Р (2,3)

к (0,0) (пробел)

Д (1,4) А (0,4)

Д (1,5) Б (0,5)

Х (3,4) С (2,4)

6 (5,1) - (4,1)

(пробел) 4

6 (5,4) 2 (4,4)

(пробел) 4

Итог: АМПЛЕЕВ ЕГОР АБС-424

К сожалению в данном примере потерян регистр, это происходит из-за того что мы взяли упрощенную таблицу для примера, но программа не изменит регистр, потому что использует таблицу 16х16.

Все три кода программы вывели Амплеев Егор Абс-424. Таким образом, результат работы программы совпадает с контрольным примером. Программа работает корректно.

# ЗАКЛЮЧЕНИЕ

В результате выполнения расчетно-графической работы были достигнуты основные цели:

Были успешно реализованы и протестированы три алгоритма шифрования , что позволило изучить их принципы работы, преимущества и недостатки.

Разработана программа с интерактивным меню, позволяющая пользователям выбирать алгоритм, операцию (шифрование/дешифрование) и источник данных (клавиатура или файл).

Программа поддерживает работу с бинарными данными, что расширяет её применение для различных типов файлов.

Для достижения поставленных целей были выполнены следующие задачи:

Исследованы принципы стеганографии (Бэкон), полиалфавитного шифрования (Нигилист) и табличного шифрования (Трисемус).

Каждый алгоритм был реализован в отдельном модуле, что обеспечило чистоту кода и простоту расширения функционала.

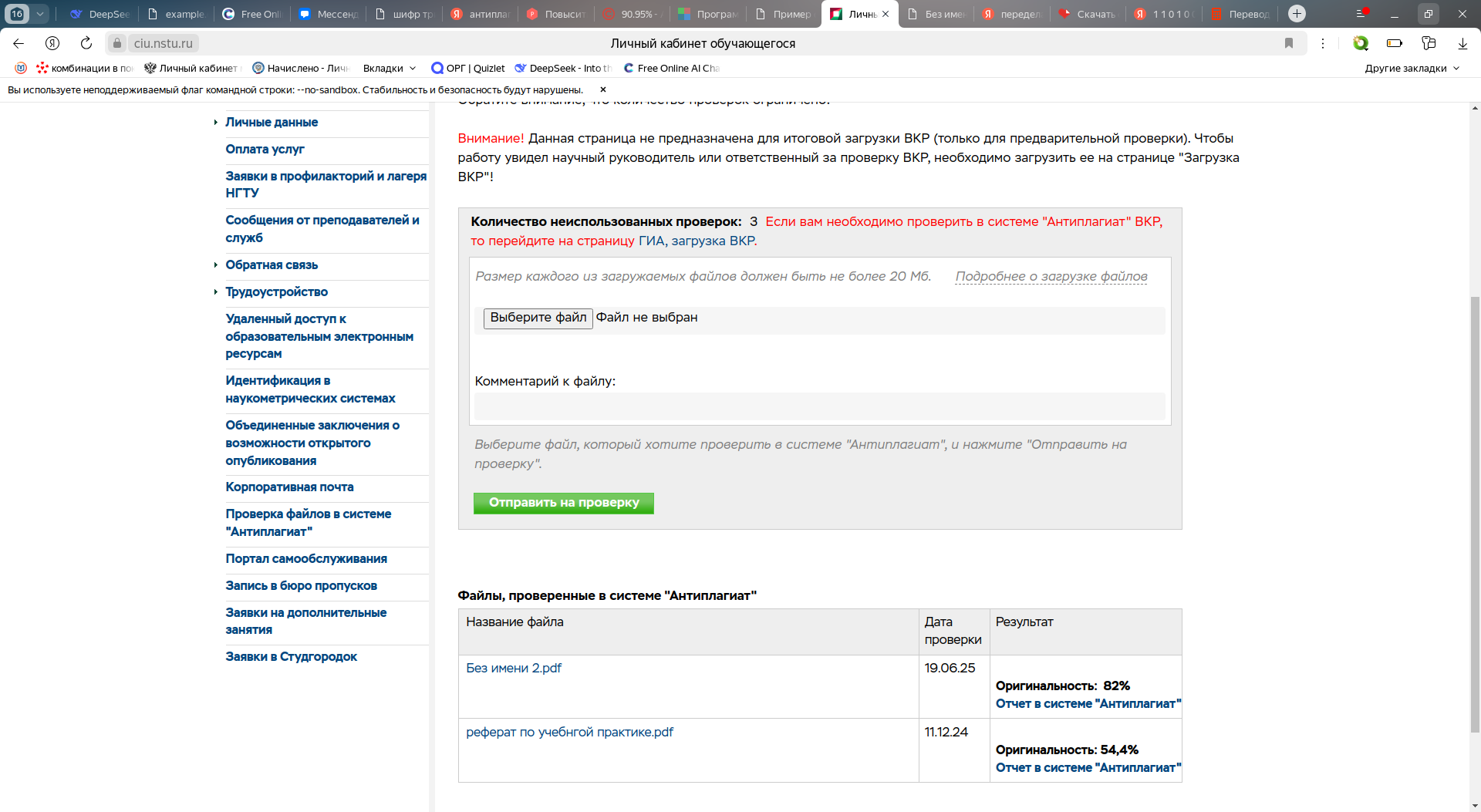
Проведено тестирование на различных типах данных для проверки корректности работы алгоритмов и обработки ошибок.

На сегодняшний день …. Данные об объекте внесены в журнал учета аттестованных объектов организации, проводившей аттестацию.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ A

**Отчет антиплагиат**



# ПРИЛОЖЕНИЕ B

Текст программы