МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Защита информации и надежность информационных систем»

Тема: «Сравнительный анализ производительности и безопасности симметричных алгоритмов шифрования XXTEA и CAST»

Исполнитель

Студент 4 курса группы 7 Боярчука С. И.

(Ф.И.О.)

Руководитель работы асс. Сазонова Д. В. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2024

Оглавление

[Введение 4](#_Toc184721050)

[1 Постановка задачи 5](#_Toc184721051)

[1.1 Вывод 6](#_Toc184721052)

[2 Описание метода 7](#_Toc184721053)

[2.1 Вывод 10](#_Toc184721054)

[3 Описание программного средства 11](#_Toc184721055)

[3.1 Вывод 16](#_Toc184721056)

[4 Тестирование программного средства 18](#_Toc184721057)

[4.1 Вывод 22](#_Toc184721058)

[5 Руководство пользователя 23](#_Toc184721059)

[5.1 Вывод 25](#_Toc184721060)

[Заключение 26](#_Toc184721061)

[Список используемых источников 27](#_Toc184721062)

[Приложение 28](#_Toc184721063)

# Введение

Аудио-стеганография представляет собой один из наиболее интересных и актуальных методов скрытия информации, который находит широкое применение в современных цифровых технологиях. Этот метод позволяет скрывать сообщения в аудиофайлах, обеспечивая тем самым защиту данных от несанкционированного доступа. В частности, реализация стеганографического метода LSB (Least Significant Bit) в аудиофайлах формата WAV стала популярной благодаря своей простоте и эффективности. Используя этот метод, можно встраивать информацию в наименьшие значащие биты аудиосигнала, что позволяет минимизировать вероятность обнаружения скрытого сообщения.

Формат WAV представляет собой несжатый формат аудиоданных, который обеспечивает высокое качество звука и подходит для применения методов стеганографии. В отличие от сжатых форматов, таких как MP3, в которых изменения в аудиоданных могут привести к потере качества, WAV-файлы сохраняют все детали звукового сигнала. Это делает их идеальными для внедрения скрытой информации без заметного влияния на восприятие аудиотрека. Метод LSB позволяет изменять только последние биты каждого семпла, что делает изменения практически незаметными для слушателя, но в то же время дает возможность встраивать значительные объемы данных.

Несмотря на свою простоту, реализация метода LSB требует внимательного подхода к выбору мест для встраивания данных и анализу возможности их извлечения. Важно учитывать, что не все аудиофайлы одинаково подходят для стеганографии, и качество скрытого сообщения напрямую зависит от характеристик исходного аудиосигнала. В данной работе будет рассмотрен процесс реализации метода LSB в аудиофайлах WAV, включая алгоритмы для встраивания и извлечения информации, а также потенциальные риски и ограничения, связанные с использованием данного подхода.

# Постановка задачи

В рамках данного курсового проекта необходимо разработать и реализовать стеганографический метод LSB (Least Significant Bit) для скрытия информации в аудиофайлах формата WAV. Основное внимание будет уделено внедрению алгоритма, оценке его эффективности и анализу безопасности, что позволит понять возможности и ограничения данного метода в современных условиях.

Для достижения поставленной цели следует рассмотреть следующие задачи:

Изучение теоретических основ стеганографии и метода LSB: Необходимо рассмотреть принципы стеганографии, ее применение в различных областях, а также глубже понять, как работает метод LSB, включая его преимущества и недостатки.

Разработка программного обеспечения: Приложение должно быть реализовано на языке программирования Python с использованием библиотеки Tkinter для создания графического интерфейса. Программа должна обеспечивать возможность загрузки аудиофайлов WAV, ввода сообщения для скрытия и выполнения операции стеганографии.

Реализация функций шифрования и расшифрования: Важно, чтобы приложение могло не только скрывать информацию в аудиофайле, но и извлекать ее, обеспечивая корректность и целостность данных. Это включает обработку ошибок и проверку успешности операций.

Оценка производительности: Приложение должно выводить время, затраченное на скрытие и извлечение сообщения, что позволит оценить производительность метода LSB в зависимости от размеров аудиофайлов и объема скрываемой информации.

Анализ безопасности: Провести анализ устойчивости метода LSB к различным атакам, таким как изменение аудиофайла, сжатие и другие виды манипуляций. Это важно для понимания, насколько надежно скрытая информация защищена.

Создание удобного интерфейса: Разработка интуитивно понятного пользовательского интерфейса, который позволит пользователям с различным уровнем подготовки легко взаимодействовать с приложением. Это включает возможность выбора аудиофайла, ввода сообщения и отображения результатов работы.

Документация и выводы: Необходимо составить итоговую документацию, в которой будут представлены результаты тестирования приложения, анализ производительности и безопасности, а также рекомендации по использованию метода LSB в различных сценариях.

Проект также подразумевает использование дополнительных библиотек для работы с аудиофайлами, таких как wave и numpy, что позволит упростить обработку данных и повысить эффективность работы приложения. Важно, чтобы в конечном итоге был представлен не только рабочий инструмент, но и глубокий анализ его возможностей, что поможет пользователям понять, как и когда применять данный метод стеганографии.

* 1. Вывод

В ходе реализации проекта по стеганографическому методу LSB в аудиофайлах формата WAV будет достигнута основная цель — разработка эффективного инструмента для встраивания и извлечения скрытых текстовых данных. Проект позволит глубже понять механизмы стеганографии и их применение в аудиофайлах, а также выявить сильные и слабые стороны метода LSB.

Создание программного обеспечения, которое обеспечивает высокое качество звука после встраивания данных, станет важным шагом в практической реализации стеганографических методов. Оценка производительности и качества аудиофайлов до и после обработки даст возможность пользователям увидеть реальные результаты работы программы и оценить ее эффективность.

Результаты тестирования не только подтвердят работоспособность алгоритма, но и позволят выявить оптимальные условия для его применения. Проект окажется полезным для специалистов в области информационной безопасности, разработчиков и исследователей, интересующихся стеганографией и защитой данных.

Данная работа также создаст основу для будущих исследований в области стеганографии, открывая возможности для дальнейших улучшений и оптимизаций методов скрытия информации. В конечном итоге, проект будет способствовать повышению уровня безопасности и конфиденциальности данных в условиях современного цифрового мира, где защиту информации невозможно переоценить.

# Описание метода

Компьютерная стеганография базируется на двух основных принципах.

Первый принцип заключается в том, что файлы, содержащие оцифрованное изображение или звук, могут быть до некоторой степени видоизменены без потери их функциональности в отличие от других типов данных, требующих абсолютной точности.

Второй принцип заключается в неспособности органов чувств человека различать незначительные изменения в цвете изображения или качестве звука. Этот принцип особенно легко применять к изображению или звуку, несущему избыточную информацию.

Так как компьютерная стеганография является молодым направлением в области защиты информации и до недавнего времени не имела своей терминологии, то в 1996 году было предложено использовать единую терминологию.

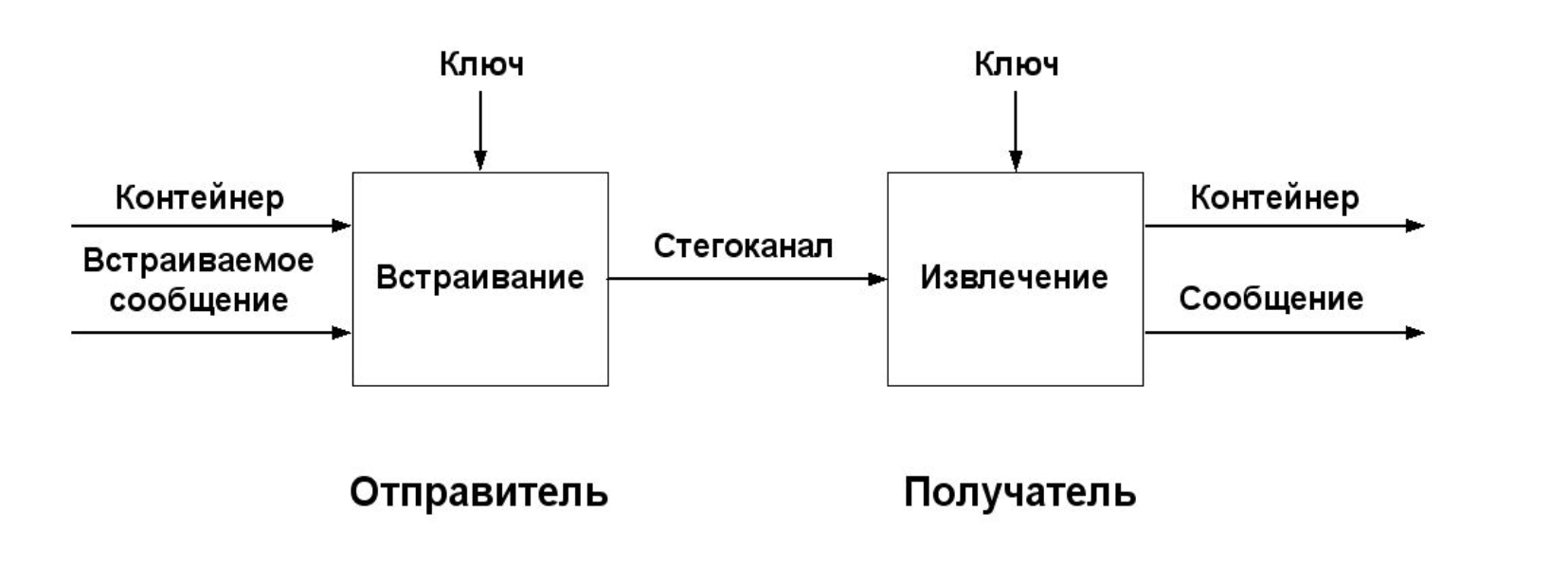


Рисунок 1.1 – Структурная схема стегосистемы.

Встроенное (скрытое) сообщение – это сообщение, встроенное в контейнер.

Контейнер – это любая информация, предназначенная для скрытия сообщения. Выбор вида контейнера оказывает существенное влияние на надёжность стегосистемы и возможность обнаружения факта передачи скрытого сообщения. По размеру (протяжённости) контейнеры можно разделить на два типа: непрерывные (потоковые) и ограниченной (фиксированной) длины.

Особенностью потокового контейнера является то, что невозможно определить его начало и конец. В таком контейнере биты информации, используемые для скрытия сообщения, включаются в общий поток в реальном масштабе времени и выбираются с помощью специального генератора, задающего расстояния между ними. В непрерывном потоке данных самая большая трудность для получателя – определить, когда начинается скрытое сообщение. При наличии в потоковом контейнере сигналов синхронизации или границ пакета, скрытое сообщение начинается сразу после одного из них. В свою очередь для отправителя сообщения возможны проблемы, если он не уверен в том, что поток контейнера будет достаточно длинным для размещения всего сообщения.

В зависимости от вида информации, используемой для встраивания сообщений, контейнеры могут быть визуальные, звуковые и текстовые. Звуковой контейнер представляет собой речевой или музыкальный сигнал, в котором для встраивания сообщений используются младшие биты аудиосигнала, что практически не отражается на качестве звука. Классификация методов компьютерной стеганографии. Методы компьютерной стеганографии можно разделить в целом на два вида: методы, основанные на избыточности визуальной и аудиоинформации; методы, основанные на использовании специальных свойств компьютерных форматов.

Методы, основанные на избыточности визуальной и аудиоинформации, для скрытия информации используют младшие разряды цифровых отсчётов цифрового изображения и звука, которые содержат очень мало полезной информации. Их заполнение дополнительной информацией практически не влияет на качество восприятия, что и даёт возможность скрытия конфиденциальной информации.

Преимуществом этих методов является возможность скрытой передачи большого объёма информации и возможность защиты авторского права путём создания скрытого изображения товарной марки, регистрационного номера и т.п.

Недостаток метода состоит в том, что за счёт введения дополнительной информации искажаются статистические характеристики  цифровых потоков. Для снижения компрометирующих признаков требуется коррекция статистических характеристик.

Одним из наиболее распространённых методов стеганографии, использующих психофизические особенности человека, является метод замены младших бит информации, или LSB-метод (Least Significant Bits). Техника LSB была впервые описана в 1996 году в работе, посвященной цифровой стеганографии. С тех пор этот метод стал популярным благодаря своей простоте и эффективности. Он активно используется в различных областях, включая защиту авторских прав, скрытую передачу информации и обеспечение конфиденциальности данных.

Суть метода состоит в замене нескольких младших бит в байтах данных. Он применяется в графических файлах, использующих для формирования цвета каждого элемента изображения (пиксела) значения некоторых составляющих (например, значения составляющих основных цветов – красного, зелёного и синего), или в звуковых файлах, использующих для формирования звука значения дискретизированных амплитуд сигнала.

При оцифровке изображения или звука всегда существует погрешность дискретизации, которая обычно находится на уровне младшего значащего бита. Это значит, что фактически неизвестно, что будет стоять в младшем значащем разряде цифрового представления цвета или звука. Поэтому при замене только самого младшего значащего бита говорить о каком-либо искажении изображения или звука  не имеет смысла. Однако при замене только одного младшего бита такой метод имеет достаточно малую ёмкость, порядка 10% от объёма файла-контейнера, поэтому на практике используют замену более одного бита.

Например, мы можем представить десятичное число 170 в двоичной записи как 10101010 (мы предполагаем, что машина имеет прямой порядок байтов, адрес начинается справа и увеличивается влево). Как показано на Рисунке 1.2, младший бит в данном случае равен 0.

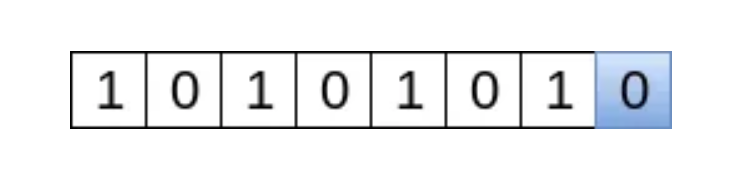


Рисунок 1.2 – Младший Бит.

В упрощенной форме алгоритм LSB заменяет младший бит каждого байта в «несущих» данных одним битом из «секретного» сообщения. Эта концепция визуализирована на Рисунке 1.3.

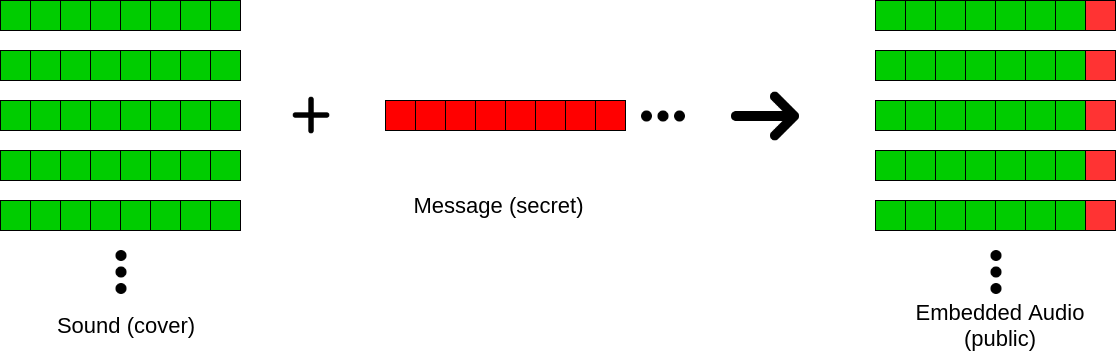


Рисунок 1.3 – Диаграмма работы LSB.

Отправитель выполняет «встраивание» битов секретного сообщения в данные носителя побайтно. В то время как получатель выполняет процедуру «извлечения», считывая младшие биты каждого байта полученных данных, таким образом получатель восстанавливает секретное сообщение. Как показано на рисунке 1.4

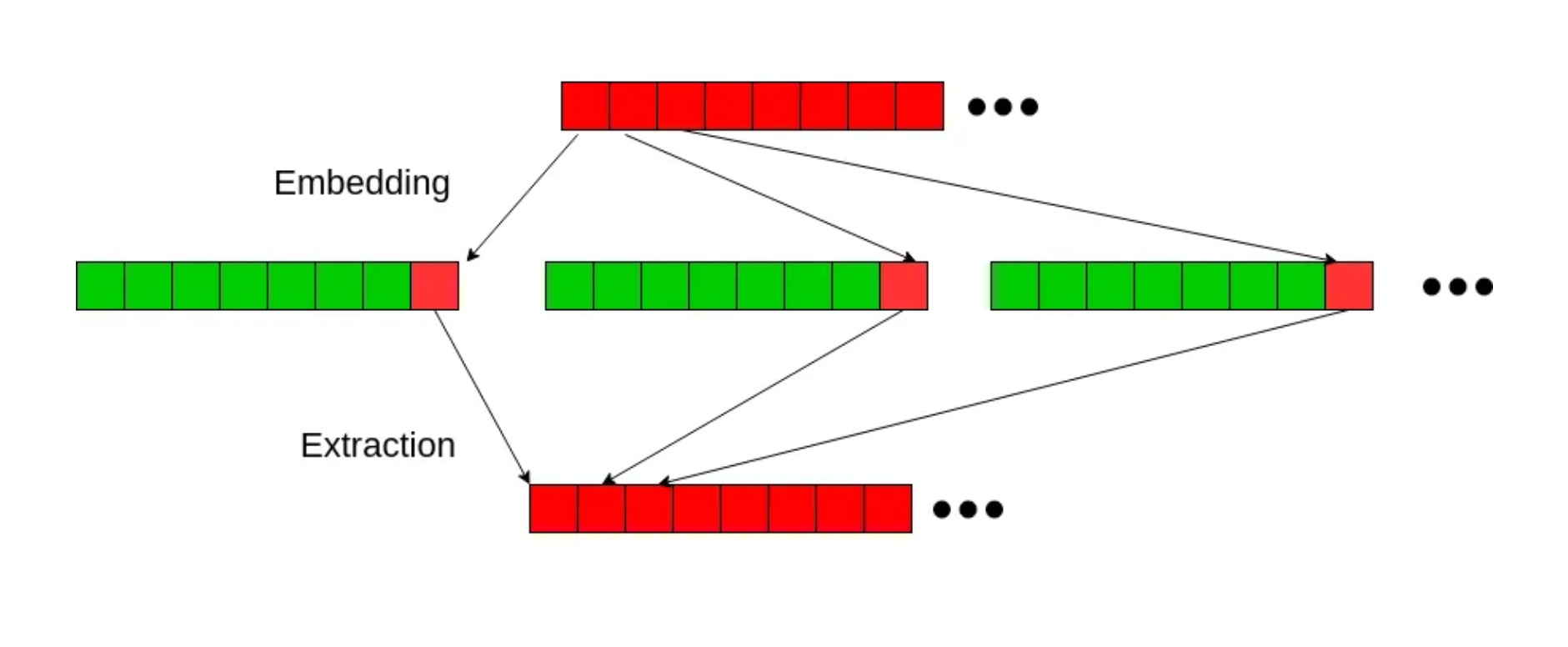


Рисунок 1.4 –Восстановление секретного сообщения.

Основная идея здесь в том, что мы пытаемся использовать человеческое восприятие целостности несущего сигнала. Стеганография LSB очень популярна для стеганографии изображений, то есть сокрытия секретов в изображениях. А изменение LSB влияет на цвет настолько незначительно, что изменение цвета обычно незаметно для человеческого глаза. Однако человеческое ухо более чувствительно к незначительным изменениям звука, и, следовательно, добавляемый нами «шум» имеет больше шансов быть замеченным.

* 1. Вывод

Метод компьютерной стеганографии, основанный на замене младших значащих битов (LSB), представляет собой эффективный и широко применяемый подход к скрытому обмену информацией в цифровых медиафайлах. Основные принципы, на которых базируется этот метод, заключаются в том, что цифровые изображения и аудиофайлы могут быть изменены без потери их функциональности, а человеческие органы чувств не способны различать незначительные изменения в цвете и звуке. Эти особенности делают LSB-метод особенно подходящим для встраивания данных, поскольку позволяет скрывать информацию с минимальными визуальными и акустическими искажениями.

Контейнеры, используемые для стеганографии, играют ключевую роль в надежности и обнаружении скрытых сообщений. Различие между потоковыми и фиксированными контейнерами определяет, как информация встраивается и извлекается. Визуальные и звуковые контейнеры обеспечивают возможность использования избыточной информации, что позволяет скрывать значительные объемы данных, сохраняя при этом качество оригинального файла.

Хотя LSB-метод имеет свои преимущества, такие как простота реализации и высокая степень скрытности, он также подвержен некоторым недостаткам, включая искажение статистических характеристик цифровых потоков. Эти недостатки требуют дополнительных мер для коррекции и защиты от обнаружения. Тем не менее, благодаря своей эффективности и простоте, LSB-метод остается одним из самых популярных инструментов в области стеганографии, применяясь в различных сферах, включая защиту авторских прав и конфиденциальность данных.

Таким образом, LSB-метод не только углубляет понимание стеганографии как дисциплины, но и предоставляет практические решения для защиты информации в современных условиях. С развитием технологий и увеличением цифровых данных, этот метод продолжает эволюционировать, открывая новые возможности для безопасной передачи информации.

# Описание программного средства

В данной курсовой работе реализовано приложение для стеганографии, использующее метод наименьших значащих битов (LSB) для скрытия информации в аудиофайлах формата WAV. Для разработки этого приложения использовались различные языки программирования и технологии, которые обеспечивают его функциональность и удобство использования.

Используются следующие языки и технологии:

Python: Python — это высокоуровневый язык программирования, который был выбран за свою простоту и читаемость. Эти характеристики позволяют быстро разрабатывать и тестировать алгоритмы стеганографии, что критически важно для эффективной работы приложения.

Библиотеки Python: В приложении используются несколько библиотек Python. Библиотека wave — это стандартная библиотека для работы с WAV-файлами, которая позволяет считывать и записывать аудиоданные в этом формате. Библиотека struct используется для работы с байтовыми данными и преобразования между строками и их представлением в виде структур, что необходимо для манипуляции аудиосигналами.

Библиотека math: Это библиотека для выполнения математических операций, которая может пригодиться при обработке данных. Например, она может использоваться для вычислений, связанных с количеством битов и слотов для хранения данных.

Tkinter: Tkinter — это библиотека для создания графического пользовательского интерфейса (GUI). Она позволяет разработать удобный интерфейс для ввода данных пользователем и отображения результатов, что делает приложение более доступным и понятным.

Filedialog и Messagebox: Эти модули из Tkinter предоставляют функции для открытия диалоговых окон выбора файлов и вывода сообщений пользователю. Это улучшает взаимодействие пользователя с приложением, позволяя легко загружать аудиофайлы и получать уведомления об ошибках.

TTK из Tkinter: Это модуль для создания современных виджетов в GUI, что улучшает внешний вид интерфейса. Использование TTK позволяет сделать приложение более привлекательным и удобным для пользователя.

Git: Git — это система контроля версий, используемая для управления версиями кода и совместной работы над проектом. Git позволяет отслеживать изменения, возвращаться к предыдущим версиям и эффективно работать в команде, что особенно полезно при разработке сложных приложений.

Visual Studio Code: Это интуитивно понятная и мощная среда для написания и отладки кода. Она поддерживает множество расширений, что делает её идеальной для работы с Python, позволяя разработчикам улучшать производительность и качество кода.

Jupyter Notebook: Jupyter Notebook может быть использован для тестирования отдельных частей кода и визуализации данных, что полезно для анализа результатов. Jupyter позволяет интерактивно выполнять код и визуализировать данные, что упрощает процесс отладки и разработки.

Структура разработанного приложения представлена на рисунке 3.1.

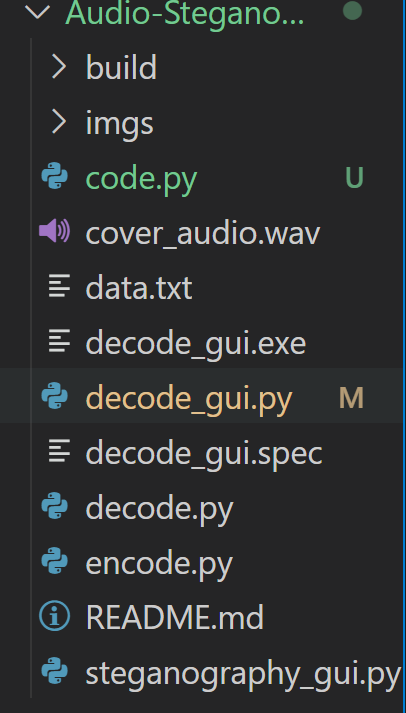


Рисунок 3.1 – Структура приложения

Ниже представлено описание структуры:

build: Директория build содержит скомпилированные файлы и артефакты, созданные во время сборки приложения. Эти файлы могут включать зависимости и другие ресурсы, необходимые для корректной работы программы.

imgs: Папка imgs предназначена для хранения изображений, используемых в приложении. Это могут быть иконки, фоны или другие графические элементы, которые улучшают визуальное восприятие интерфейса.

code.py: Этот файл содержит основной код приложения, реализующий алгоритмы стеганографии. Он отвечает за обработку аудиофайлов и внедрение скрытых данных, а также за взаимодействие с другими модулями.

cover\_audio.wav: Этот файл представляет собой аудиофайл в формате WAV, который используется в качестве основы для внедрения скрытой информации. Он служит тестовым примером для демонстрации работы алгоритма стеганографии.

data.txt: Файл data.txt используется для хранения текстовых данных, которые необходимо зашифровать и внедрить в аудиофайл. Этот файл может быть изменяемым в процессе тестирования и разработки.

decode\_gui.exe: Это исполняемый файл графического интерфейса для декодирования скрытой информации из аудиофайла. Он предоставляет пользователю удобный интерфейс для извлечения данных.

decode\_gui.py: Этот файл содержит код, реализующий графический интерфейс для декодирования. Он отвечает за взаимодействие с пользователем и отображение результатов декодирования.

decode\_gui.spec: Файл спецификации, используемый для настройки сборки исполняемого файла decode\_gui.exe. Он содержит информацию о зависимостях и конфигурации, необходимых для создания исполняемого файла.

README.md: Файл README.md содержит документацию по проекту, включая инструкции по установке, использованию и настройке приложения. Он служит руководством для разработчиков и пользователей.

steganography\_gui.py: Этот файл отвечает за создание графического интерфейса приложения для стеганографии. Он включает элементы управления для ввода данных и отображения результатов, обеспечивая удобство взаимодействия пользователя с приложением.

Базовый алгоритм изображен на рисунке 3.2

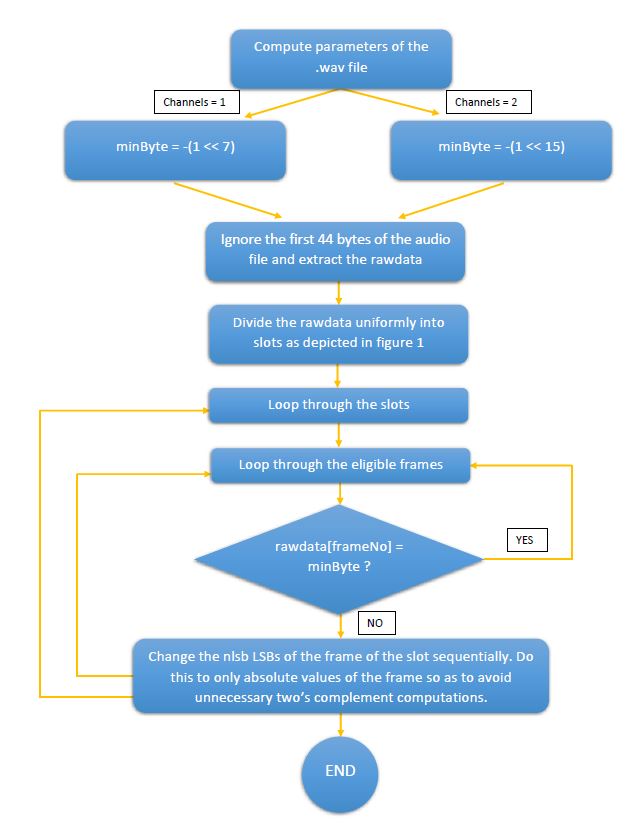


Рисунок 3.2 – Результат обработки ошибок

Мы вычисляем значение minByte – наименьшее возможное значение в raw-данных файла, которое зависит от количества каналов в файле. Мы также вводим количество младших битов – nlsb, которые будут использоваться. До 3 младших битов дают хорошие результаты. Мы последовательно меняем n младших битов кадров. Мы пропускаем кадры, значение которых равно minByte. Мы делаем это, потому что мы меняем значение, сначала взяв абсолют, данные кадра. Если эти данные равны минимально возможному (для этой длины), при взятии их абсолюта мы столкнемся с переполнением.

Было замечено, что вместо того, чтобы хранить все данные последовательно, только в 1 разделе аудиофайла, лучше равномерно распределить данные по всему файлу. Таким образом, человек, анализирующий аудио, не сможет так легко обнаружить шум. Поскольку шум будет существовать в течение очень короткого времени.

Если сообщение слишком большое, то распространение данных будет ощущаться как постоянный шум. Было создано новое значение, называемое continuous\_duration=0.2 secs. Это означает, что сообщение будет храниться в течение 0.2 секунд данных, затем некоторые кадры будут пропущены, затем некоторые данные будут храниться в течение 0.2 секунд кадров и так далее. Если звук длится всего 0.2 секунды, человеческое ухо вряд ли его уловит. Таким образом, создаются слоты изображены на рисунке 3.3

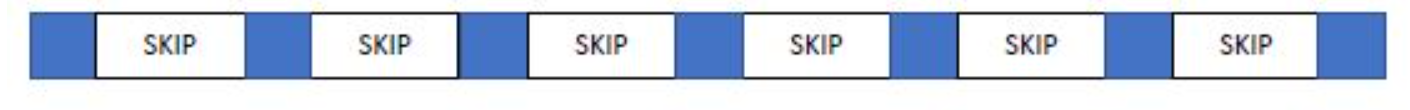


Рисунок 3.3 – Слоты

Синие области отображают 0,2 секунды данных, где сообщение сохранено. Белые области отображают неизмененные данные обложки.

Интерфейс приложения "Audio Steganography" разработан с использованием библиотеки Tkinter и включает два основных режима: шифрования и расшифровки. Оба режима позволяют пользователю легко взаимодействовать с функционалом приложения, обеспечивая интуитивно понятный доступ ко всем необходимым элементам управления.Как показано на рисунке 3.4.

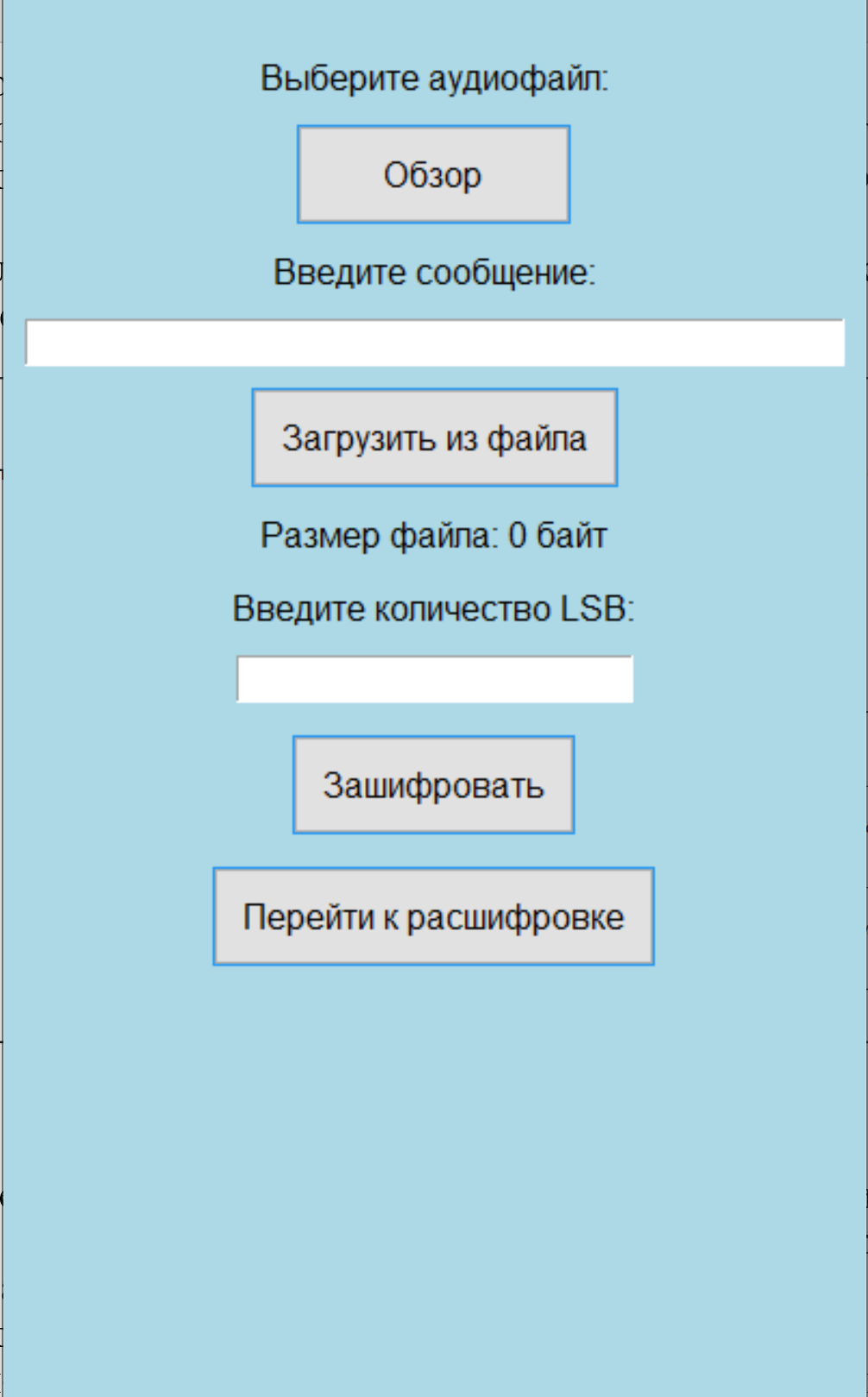


Рисунок 3.4 – Интерфейс приложения при шифровании

Главное окно

Главное окно приложения имеет заголовок "Audio Steganography" и размер 400x650 пикселей. Фон окна выполнен в светло-голубом цвете (#ADD8E6), что создает приятную атмосферу для работы с приложением.

Режим шифрования

В режиме шифрования пользователю представляется форма для выбора аудиофайла. Сначала отображается метка "Выберите аудиофайл:", после чего следует кнопка "Обзор", которая открывает диалоговое окно для выбора аудиофайла формата WAV. Далее находится поле ввода сообщения с меткой "Введите сообщение:", куда пользователь может ввести текст, который будет зашифрован в аудиофайл.

Также имеется кнопка "Загрузить из файла", которая позволяет загружать текстовое сообщение из текстового файла. Размер загруженного сообщения отображается в метке "Размер файла: 0 байт", что помогает пользователю оценить, сколько места потребуется для шифрования.

Пользователь также вводит количество наименьших значащих битов (LSB), которые будут использованы для шифрования, в соответствующее текстовое поле с меткой "Введите количество LSB:". Кнопка "Зашифровать" запускает процесс шифрования, проверяя введенные данные на корректность. В нижней части интерфейса находится кнопка "Перейти к расшифровке", которая переключает режим на расшифровку.

Режим расшифровки

В режиме расшифровки интерфейс предлагает аналогичные элементы управления. Сначала отображается метка "Выберите аудиофайл:", за которой следует кнопка "Обзор" для выбора аудиофайла с зашифрованным сообщением. Затем присутствует поле ввода для размера данных с меткой "Введите размер данных в байтах для расшифровки:", где пользователь вводит размер сообщения, которое будет извлечено,как продемонстрированное на рисунке 3.5

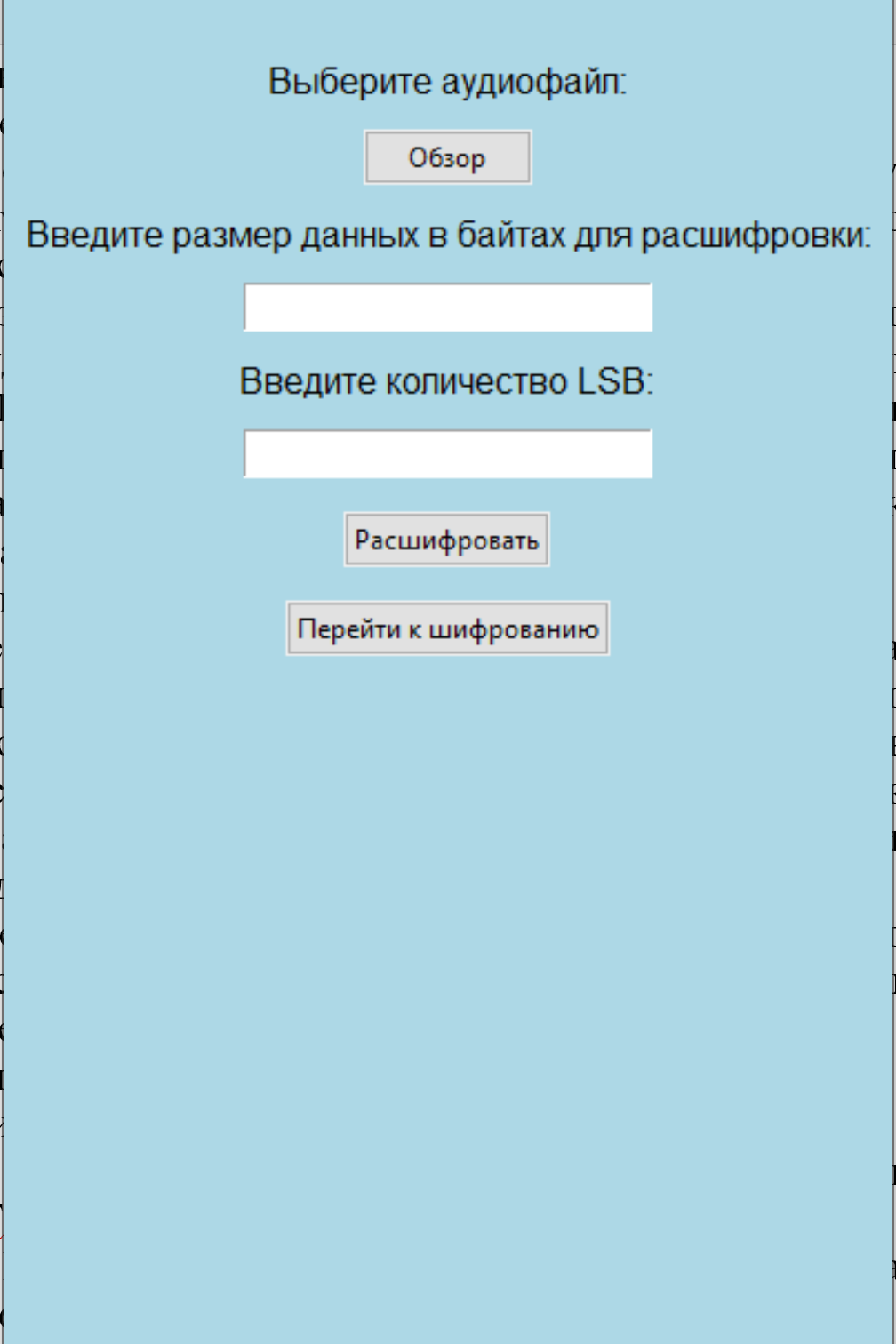


Рисунок 3.5 – Интерфейс приложения при расшифровании

Также есть поле для ввода количества LSB с соответствующей меткой, а кнопка "Расшифровать" активирует процесс извлечения сообщения из аудиофайла. Кнопка "Перейти к шифрованию" позволяет пользователю вернуться к режиму шифрования.

Общие элементы интерфейса

Все кнопки и элементы управления оформлены в едином стиле "Rounded.TButton", который предполагает округлые края и приятный цвет фона (#2196F3). При наведении кнопки меняют цвет, что улучшает пользовательский опыт. Шрифты всех текстовых элементов — Arial размером 12, что обеспечивает хорошую читабельность и комфорт при взаимодействии с приложением.

Таким образом, интерфейс приложения организован так, чтобы обеспечить простоту и интуитивность использования, позволяя пользователю без труда выполнять операции по шифрованию и расшифровке скрытой информации в аудиофайлах.

В итоге, разработанное веб-приложение позволяет выполнять следующие действия:

* + выбирать аудиофайлы в формате WAV для шифрования или расшифровки сообщений;
  + шифровать текстовые сообщения, встраивая их в аудиофайлы с использованием метода стеганографии, основанного на наименьших значащих битах (LSB);
  + загружать текстовые сообщения из текстовых файлов, что упрощает процесс ввода данных;
  + отображать размер загруженного сообщения в байтах, что помогает пользователю оценить, достаточно ли места в аудиофайле для встраивания данных;
  + извлекать скрытые сообщения из аудиофайлов, вводя размер данных для расшифровки и количество используемых LSB;
  + сохранять измененные аудиофайлы с зашифрованными сообщениями для дальнейшего использования;
  + обеспечивать удобный и интуитивно понятный интерфейс, позволяя пользователю легко переключаться между режимами шифрования и расшифровки.
  1. Вывод

Разработанное приложение представляет собой инструмент для стеганографии, позволяющий пользователям скрывать текстовые сообщения в аудиофайлах формата WAV. Оно обеспечивает возможность как шифрования, так и расшифровки данных, используя метод наименьших значащих битов (LSB).

Приложение предлагает интуитивно понятный интерфейс, который включает функции выбора аудиофайлов, ввода текстов и загрузки данных из файлов. Пользователи могут легко переключаться между режимами шифрования и расшифровки, что делает приложение доступным для широкой аудитории, даже для тех, кто не имеет технического опыта.

Кроме того, приложение демонстрирует высокую степень надежности и безопасности, обеспечивая конфиденциальность передаваемых сообщений. В итоге, разработанное веб-приложение является эффективным инструментом для работы со стеганографией, позволяя пользователям сохранять свои данные в секрете, используя аудиофайлы в качестве носителей информации.

# Тестирование программного средства

В ходе тестирования приложения были изучены различные сценарии использования, включая шифрование и расшифровку сообщений.

Первым этапом было тестирование функции шифрования текстового сообщения. После выбора аудиофайла и ввода сообщения приложение успешно встроило текст в аудиофайл с использованием алгоритма наименьших значащих битов (LSB). Результаты, продемонстрированные на рисунке 4.1, показали, что все функции шифрования работают корректно, обеспечивая целостность и безопасность данных.

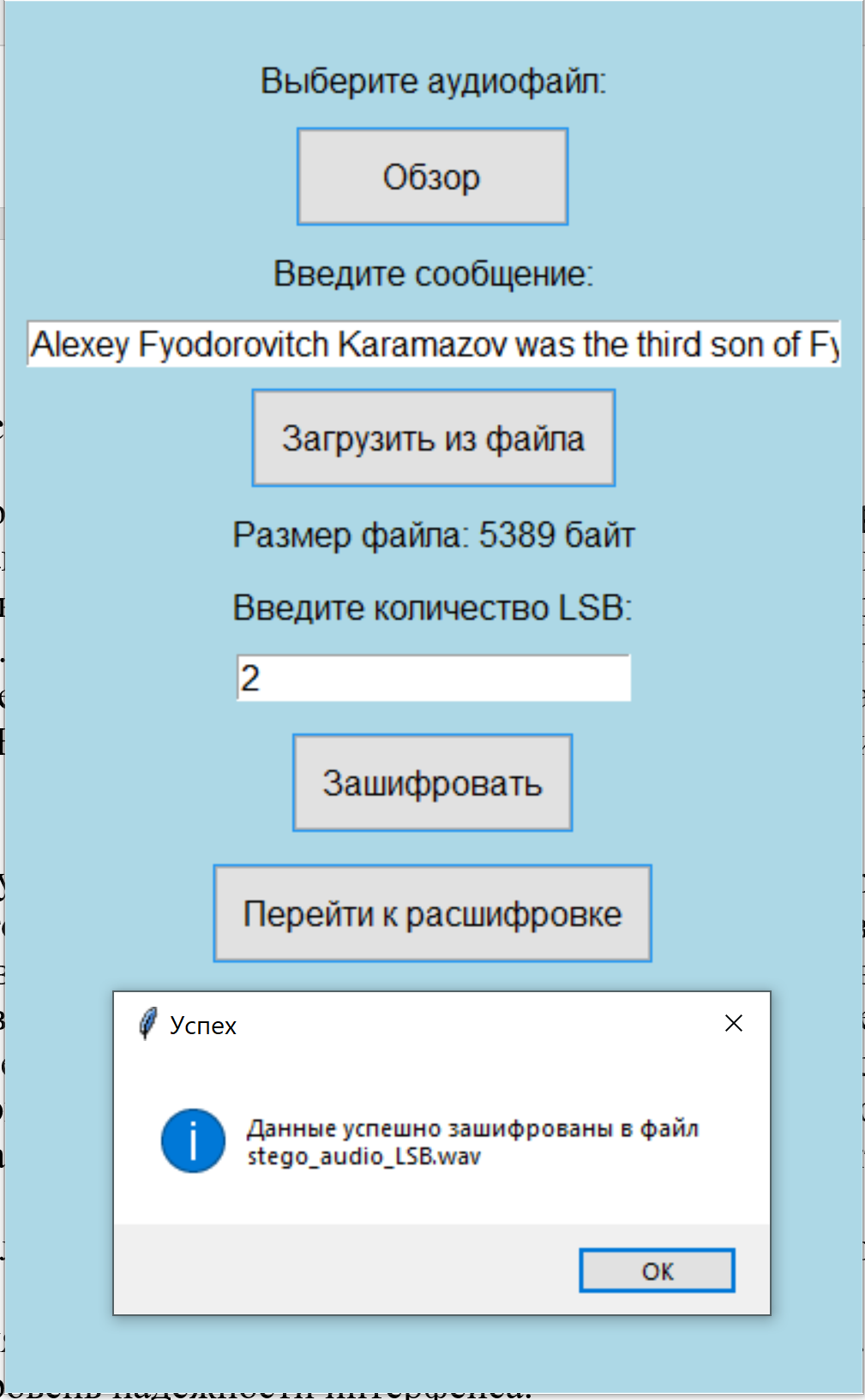


Рисунок 4.1 – Результат работы шифрования

Следующим этапом стало тестирование функции расшифровки. После выбора того же аудиофайла и ввода размера данных для извлечения приложение успешно восстановило исходное сообщение. Это подтвердило надежность алгоритмов и их способность корректно обрабатывать данные.

Результат работы показан на рисунке 4.2.

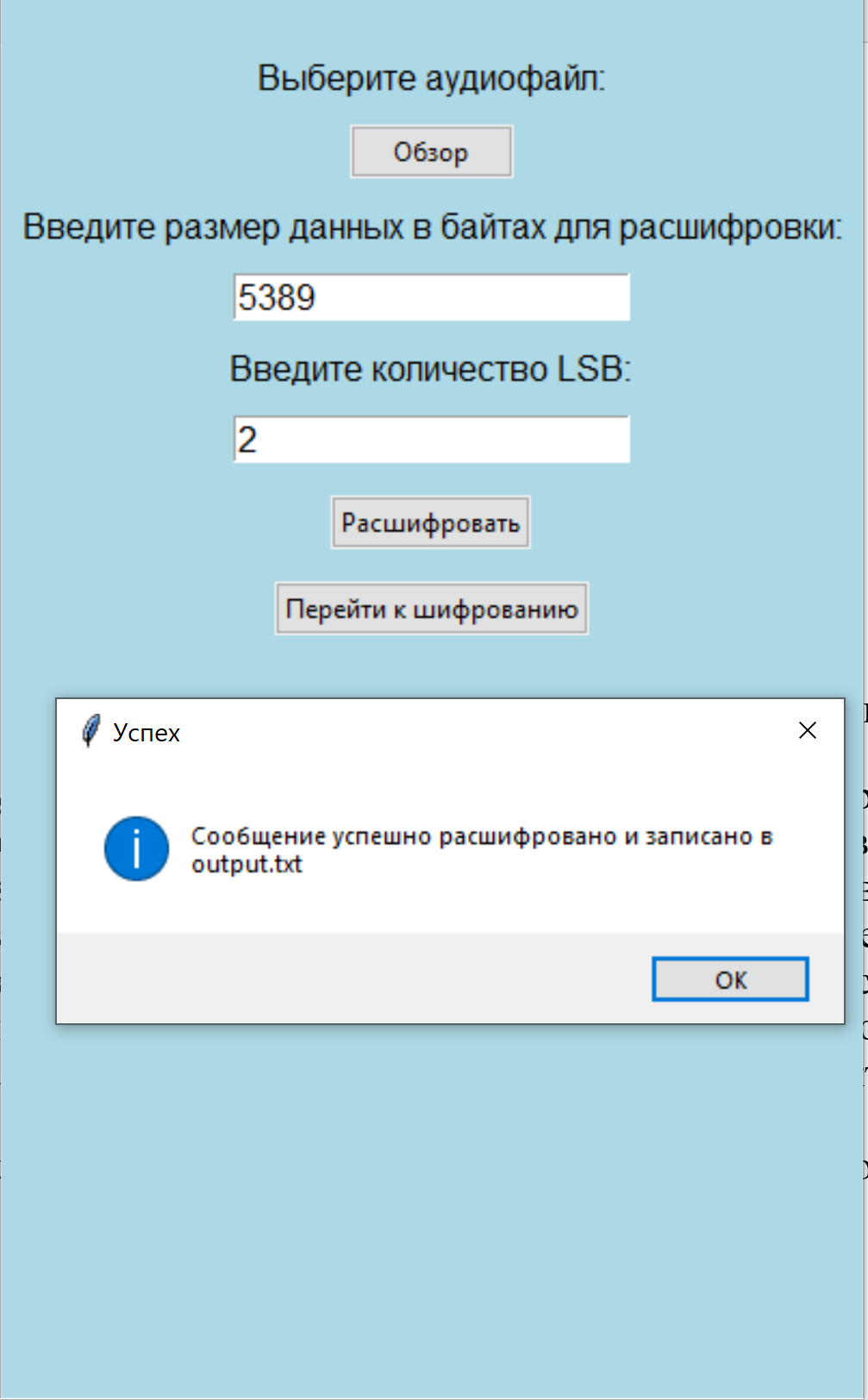


Рисунок 4.2 – Результат работы расшифрования

Для проверки корректности шифрования и насколько заметно зашифрованное сообщение была использована программа Audacity, позволяющая визуализировать аудиоданные. В процессе тестирования были созданы две звуковые дорожки: оригинальный аудиофайл и файл с зашифрованным сообщением. С помощью графического спектра Audacity было проведено сравнение этих дорожек. Оригинальная дорожка демонстрировала характерный звуковой спектр, в то время как зашифрованный файл, несмотря на наличие встраиваемых данных, сохранял аналогичное звучание и структуру, что указывает на успешное внедрение сообщения без заметных искажений. Этот анализ подтвердил, что алгоритм шифрования работает эффективно, обеспечивая сохранение качества аудиофайла и целостности передаваемых данных.

Оригинальная дорожка показана на рисунке 4.3,



Рисунок 4.3 – Оригинальная дорожка

Зашифрованная показана на рисунке 4.4

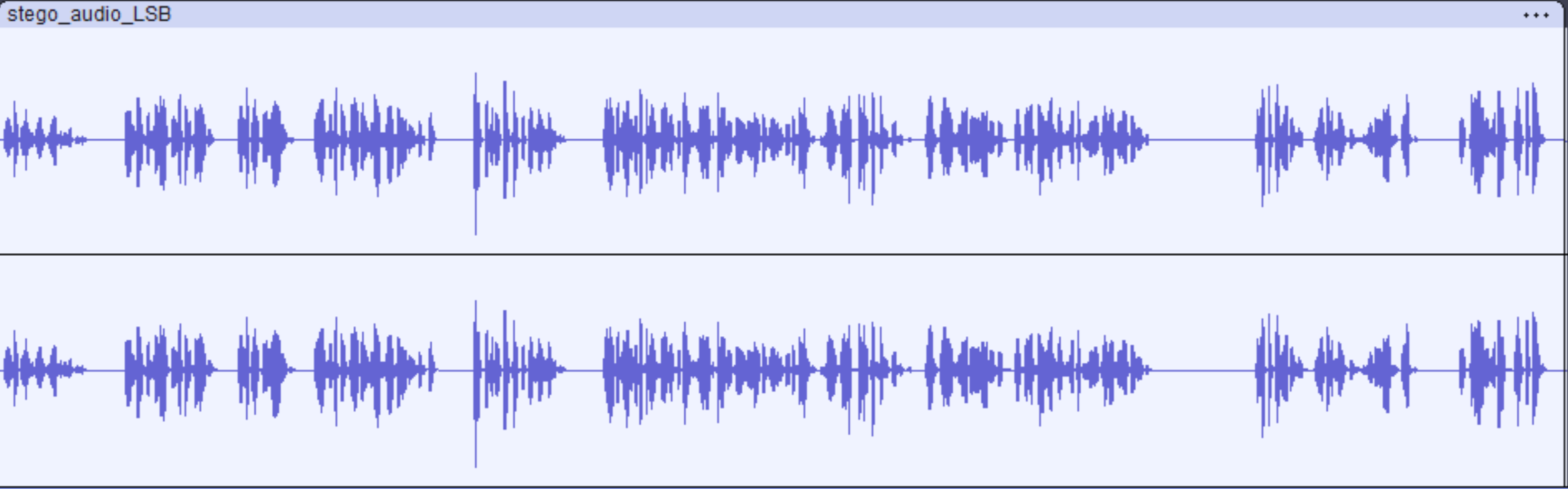


Рисунок 4.4 – Дорожка с зашифрованным сообщением

Графический спектр оригинальной дорожки показан на рисунке 4.5

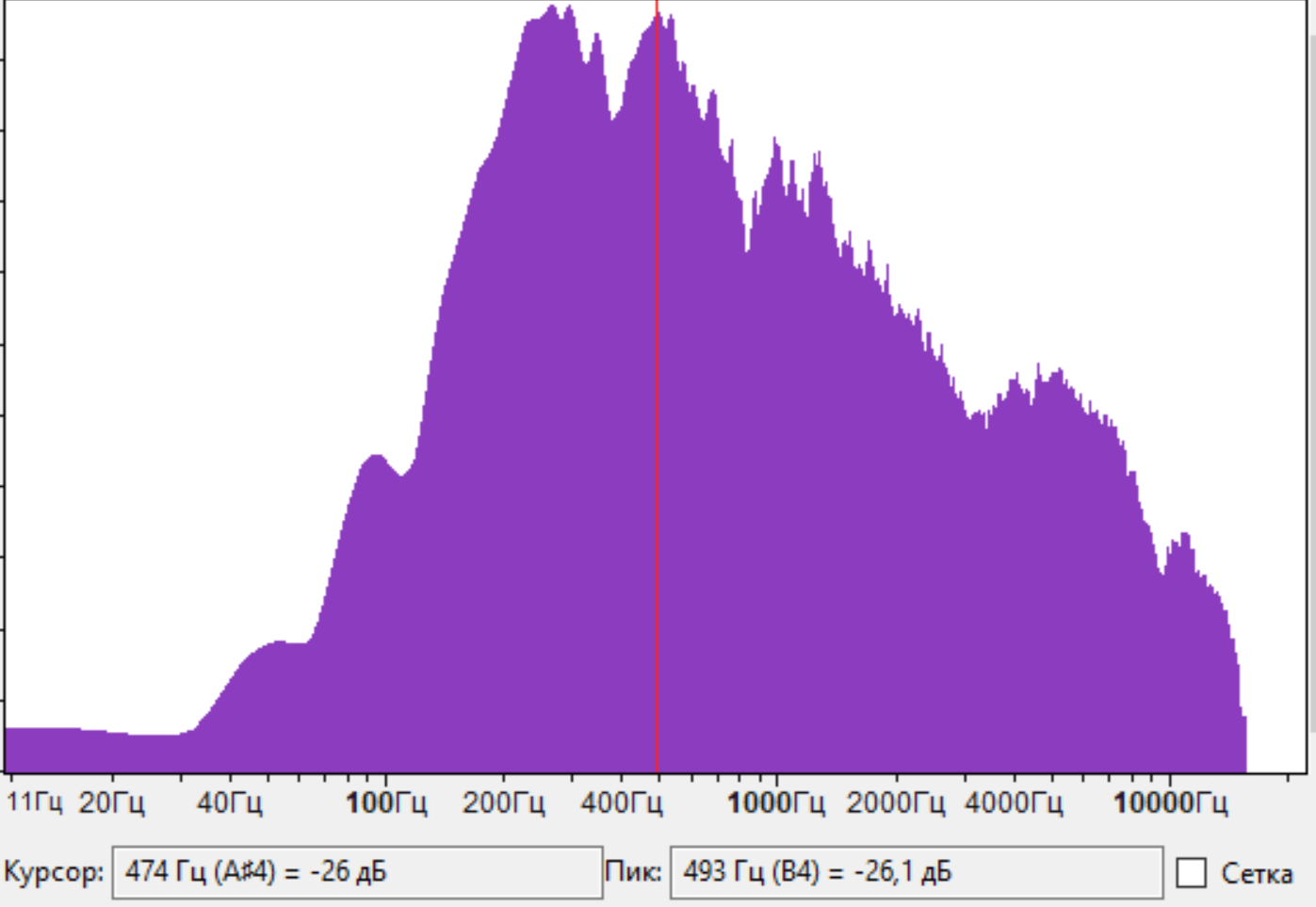


Рисунок 4.5 – Графический спектр оригинальной дорожки

Графический спектр зашифрованной дорожки показан на рисунке 4.6

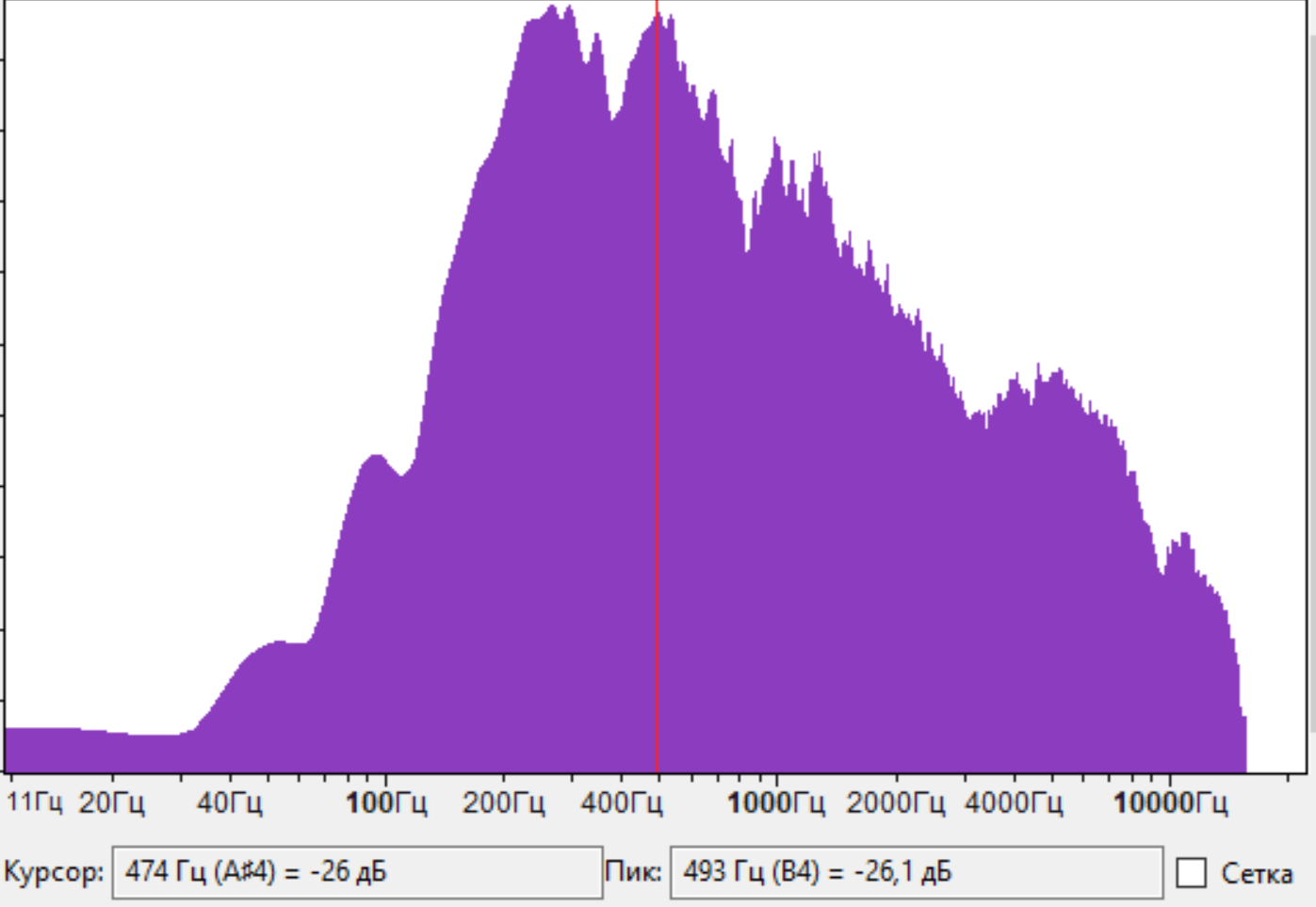


Рисунок 4.6 – Графический спектр зашифрованной дорожки

Также была протестирована возможность загрузки текстовых сообщений из файлов. Приложение правильно отображало загруженные сообщения и корректно рассчитывало их размер, что свидетельствует о его эффективности в работе с текстами. Показана на рисунке 4.7.

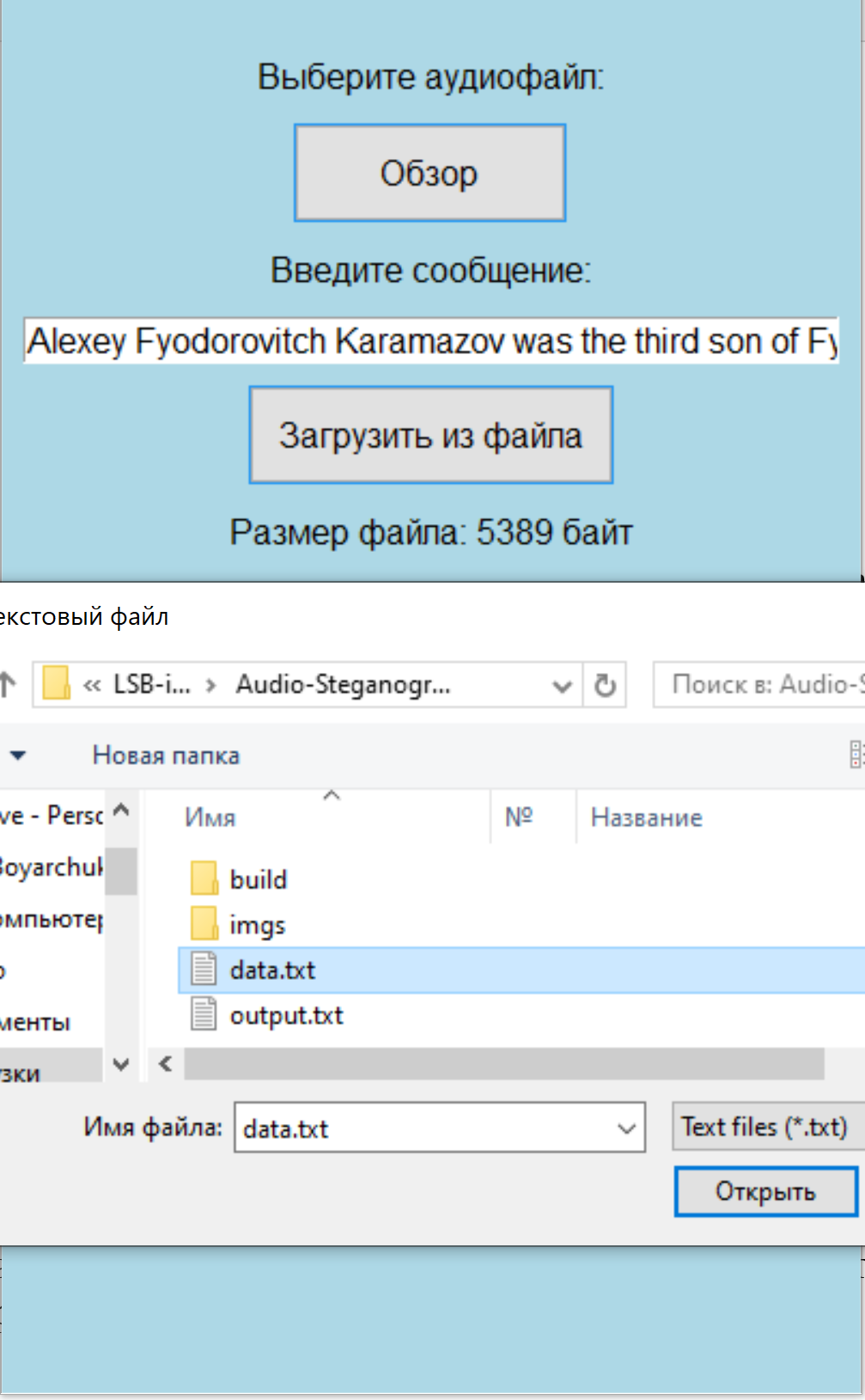


Рисунок 4.7 – Результат загрузки сообщения из файла

Дополнительно была проведена проверка обработки ошибок. Приложение реагировало на неверные форматы файлов и пустые сообщения, предоставляя пользователю ясные сообщения об ошибках. Это демонстрирует высокий уровень надежности интерфейса. Обработка ошибок показана на рисунке 4.8.

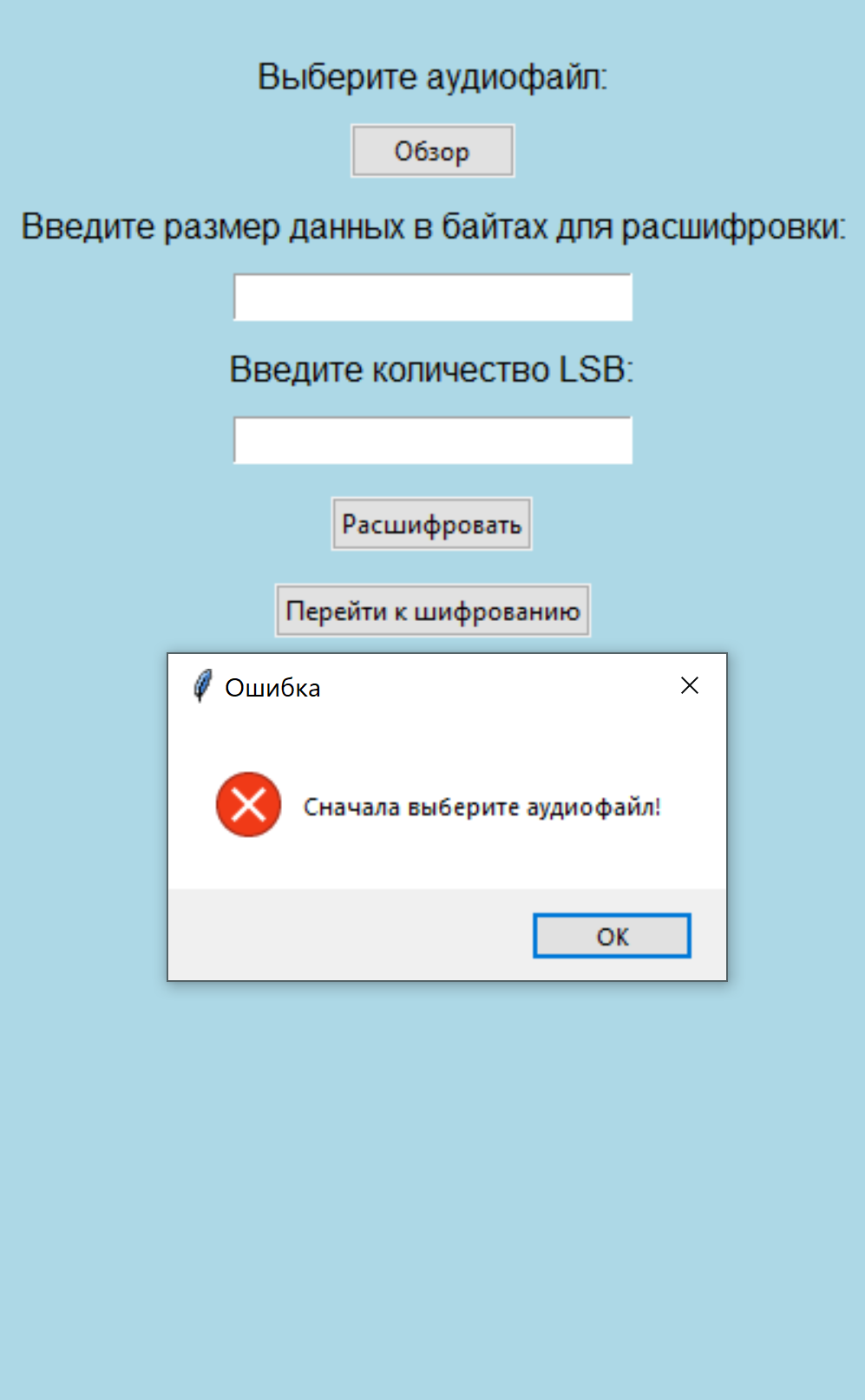


Рисунок 4.8 – Результат обработки ошибок

Пример ошибки при не выбранном айдиофайле

* 1. Вывод

Таким образом, в ходе тестирования были проверены различные аспекты функционала приложения. После шифрования для секретного сообщения размером 5389 байт и сокрытия, выполненного с использованием 3 младших битов, никаких заметных изменений в звуке не произошло. Все тестовые сценарии, включая шифрование и расшифровку, прошли успешно, что подтверждает универсальность и надежность разработанного решения. Результаты тестирования свидетельствуют о высоком уровне качества приложения, что позволяет использовать его в различных сценариях работы с аудиоданными.

# Руководство пользователя

В ходе курсового проекта было создано приложение которое позволяет пользователям скрывать текстовые сообщения в аудиофайлах формата WAV с использованием метода стеганографии.

Начальное окно приложения после запуска изображено на рисунке 5.1

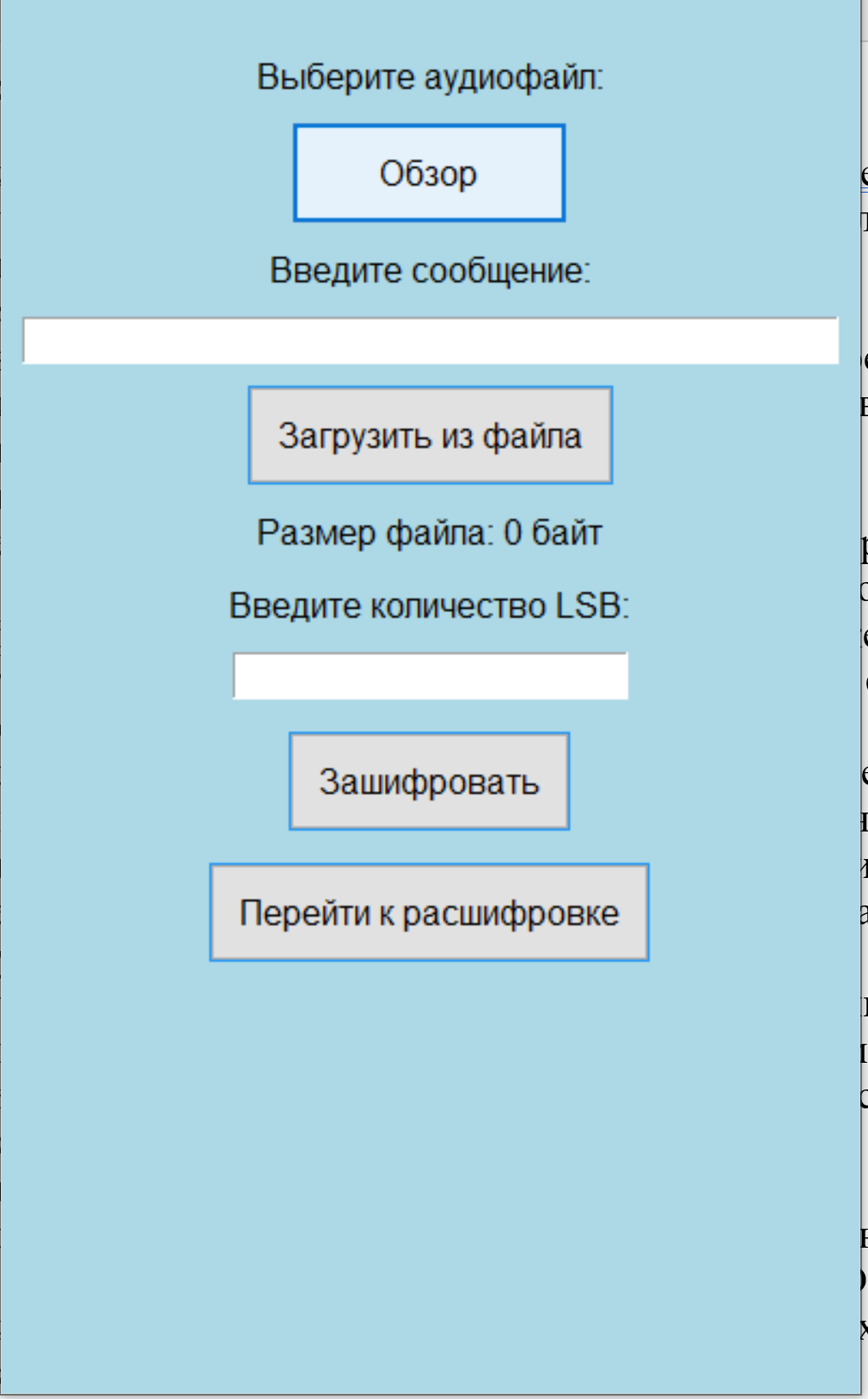


Рисунок 5.1 – Начальное окно

При запуске приложения на экране появляется главное окно с интуитивно понятным интерфейсом. В нижней части окна находится основная кнопка: для выбора режима работы.

Режим шифрования изображен на рисунке 5.2

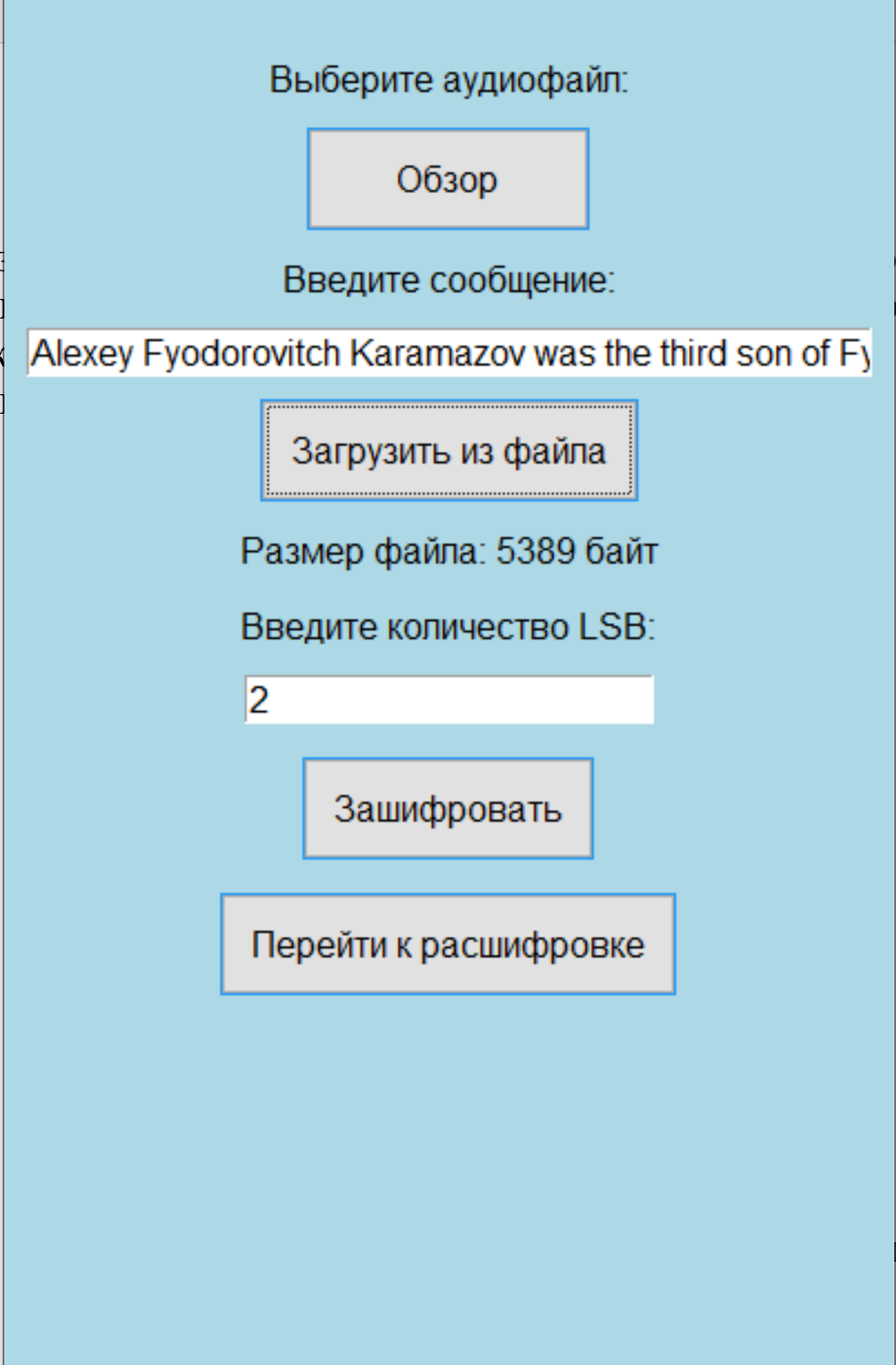


Рисунок 5.2 – Окно режима шифрования

Изначально задан выбора режима шифрования пользователю предлагается выбрать аудиофайл. Нажимая кнопку "Обзор", можно открыть диалоговое окно для выбора файла в формате WAV. После выбора файла приложение автоматически отображает размер загруженного сообщения в метке "Размер файла: X байт".

В следующем шаге пользователь вводит текстовое сообщение в поле "Введите сообщение:". Если необходимо, можно загрузить текст из файла, нажав кнопку "Загрузить из файла". После этого размер сообщения обновляется, что позволяет пользователю оценить, достаточно ли места для шифрования.

Далее необходимо ввести количество наименьших значащих битов (LSB), которые будут использоваться для шифрования. После ввода всех данных пользователь нажимает кнопку "Зашифровать", и приложение начинает процесс шифрования. По завершении операции появляется уведомление об успешном завершении, а зашифрованный файл сохраняется под именем "stego\_audio\_LSB.wav".

Режим расшифровки изображен на рисунке 5.3

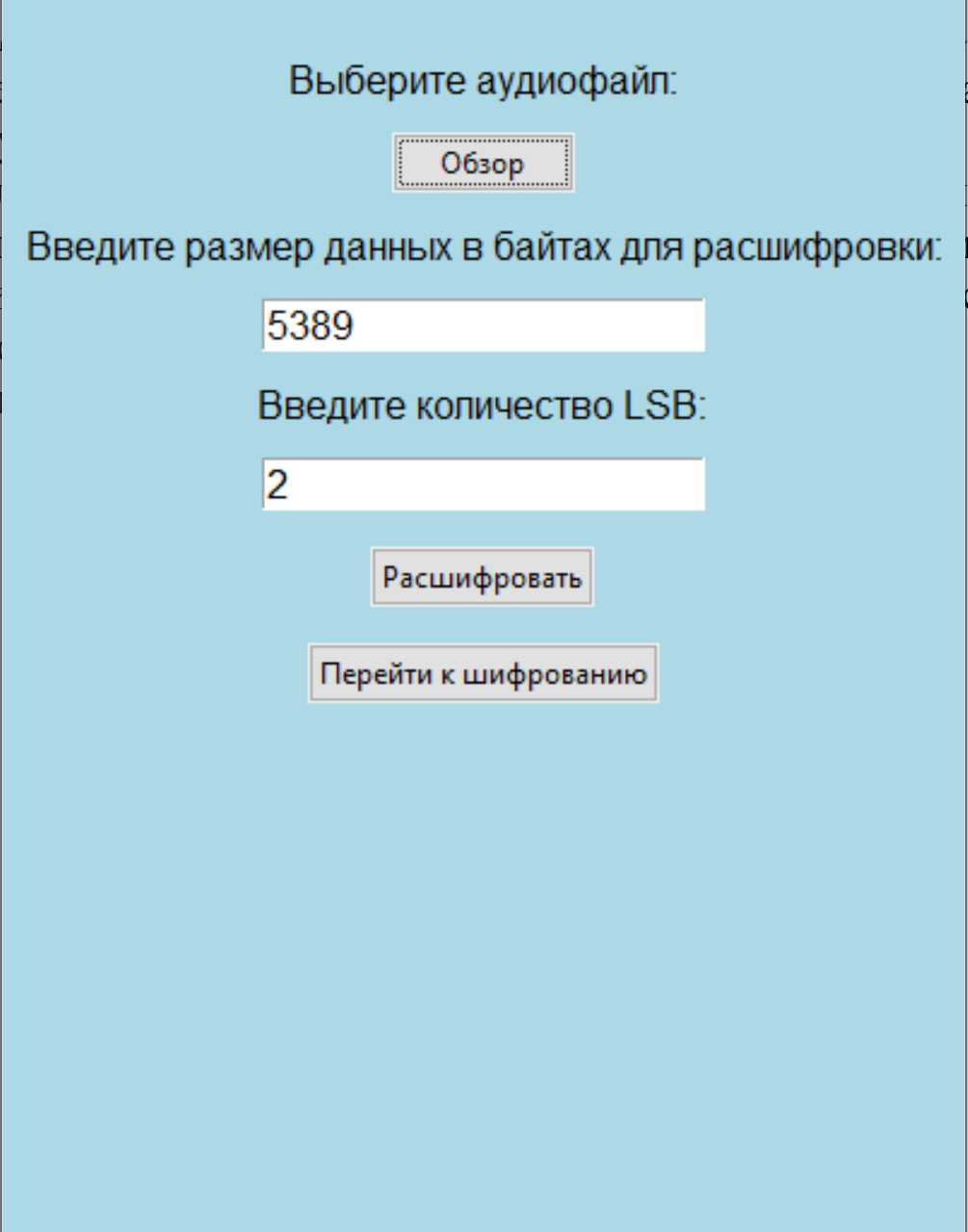


Рисунок 5.3 – Окно режима расшифрования

При переходе в режим расшифровки пользователю необходимо выбрать аудиофайл с зашифрованным сообщением, нажав кнопку "Обзор". После выбора файла приложение предлагает ввести размер данных, которые нужно расшифровать, в соответствующее поле.

Также требуется указать количество LSB, использованных при шифровании. После ввода всех необходимых данных пользователь нажимает кнопку "Расшифровать". Приложение извлекает скрытое сообщение, и по завершении процесса появляется уведомление об успешном извлечении.

Извлеченное сообщение сохраняется в текстовом файле с именем "output.txt", что позволяет пользователю легко получить доступ к расшифрованному тексту.

Обработка ошибок изображена на рисунке 5.4

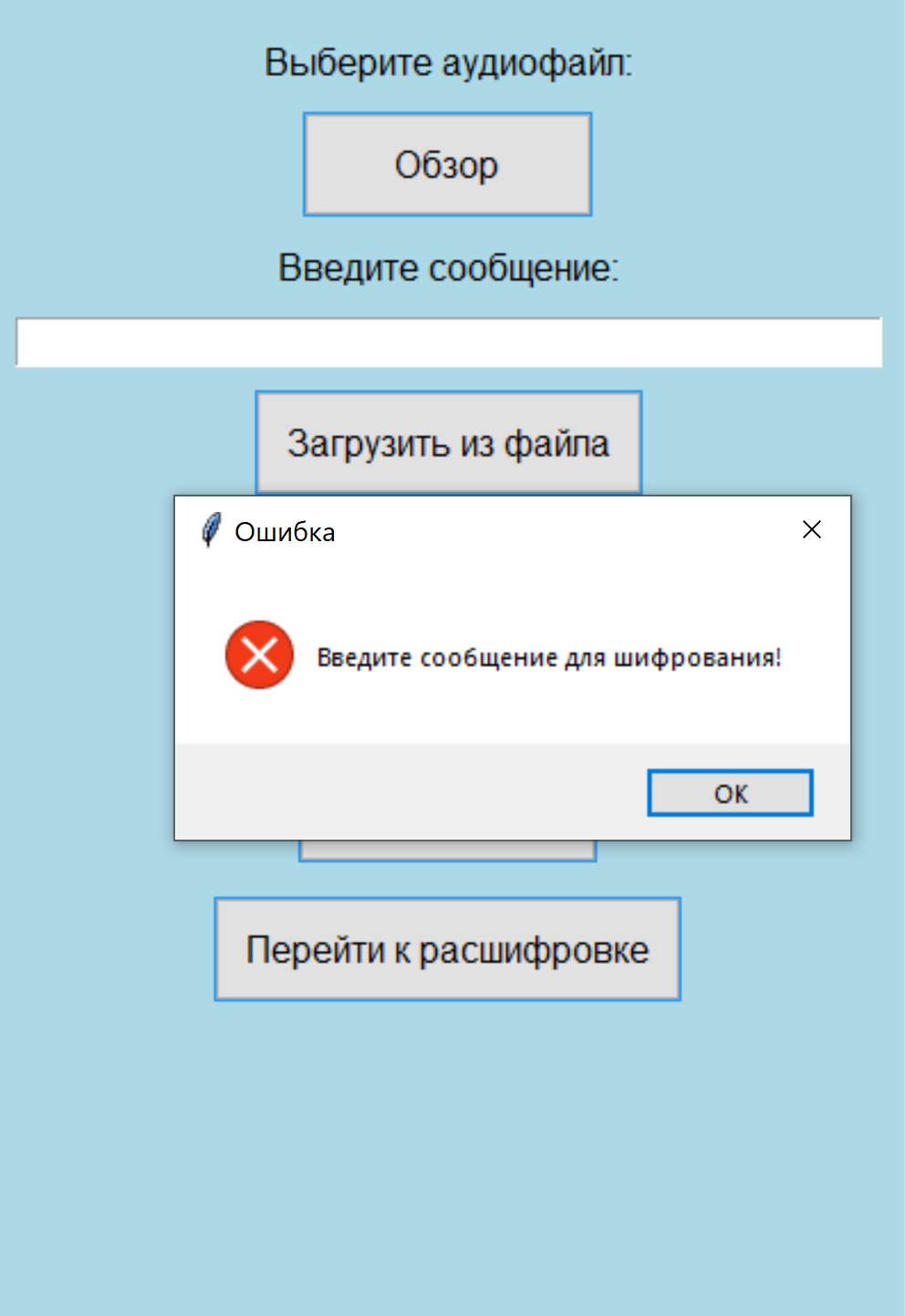


Рисунок 5.4 – Обработка ошибок

В случае возникновения ошибок, таких как неверный формат файла или пустые поля, приложение отображает соответствующее сообщение об ошибке. Это позволяет пользователю быстро понять, что необходимо исправить для успешного выполнения операции.

* 1. Вывод

Данное руководство пользователя содержит полную и структурированную информацию, необходимую для успешного использования приложения. В нем описаны все ключевые функции, включая процесс шифрования и расшифровки сообщений, а также навигацию по интерфейсу. Пользователь сможет легко найти инструкции по выбору аудиофайлов, вводу сообщений, отображению размеров данных и обработке ошибок. Следуя представленным шагам, пользователи смогут эффективно использовать приложение и реализовывать стеганографические операции с аудиофайлами без затруднений.

# Заключение

В рамках данного курсового проекта было создано приложение которое представляет собой эффективное решение для скрытия текстовых сообщений в аудиофайлах формата WAV с использованием метода стеганографии на основе наименьших значащих битов (LSB). Оно позволяет пользователям надежно встраивать информацию в аудиоданные, обеспечивая при этом высокую степень конфиденциальности и безопасности передаваемых сообщений.

Разработка приложения включала тщательную постановку задачи, выбор подходящего метода и создание интуитивно понятного интерфейса. Благодаря продуманной архитектуре, пользователи могут легко взаимодействовать с программой, шифруя и расшифровывая сообщения всего за несколько шагов. Тестирование подтвердило надежность и эффективность всех реализованных функций, что делает приложение готовым к использованию в различных сценариях.

В заключение, метод LSB является мощным инструментом для тех, кто ищет способы защиты своих данных. Его доступность и простота в использовании делают его подходящим как для опытных пользователей, так и для новичков в области стеганографии.

# Список используемых источников

1. Урбанович, П. П. Защита информации методами криптографии, стеганографии и обфускации: учеб.-метод. пособие. – Минск: БГТУ, 2016. – 220 с.
2. Урбанович, П. П. Лабораторный практикум по дисциплинам «Защита информации и надежность информационных систем» и «Криптографические методы защиты информации». В 2 ч. Ч. 1. Кодирование информации: учеб.-метод. пособие для студентов учреждений высшего образования / П. П. Урбанович, Д. В. Шиман, Н. П. Шутько. – Минск: БГТУ, 2019. – 116 с
3. Audio Steganography : The art of hiding secrets within earshot [электронный ресурс]. Режим доступа: https://sumit-arora.medium.com/audio-steganography-the-art-of-hiding-secrets-within-earshot-part-1-of-2 Дата доступа: 19.11.2024.
4. Борисевич М.Н. Основы информационных технологий для специалистов АПК. – Витебск:2017.

# Приложение

# import wave

# import struct

# import math

# import tkinter as tk

# from tkinter import filedialog, messagebox

# from tkinter import ttk

# cover\_path = "cover\_audio.wav"

# stego\_path = "stego\_audio\_LSB.wav"

# msg\_path = "data.txt"

# output\_path = "output.txt"

# continuous\_duration = 0.2

# class SteganographyApp:

# def \_\_init\_\_(self, root):

# self.root = root

# self.root.title("Audio Steganography")

# self.root.geometry("400x650")

# self.root.configure(bg="#ADD8E6")

# self.current\_frame = None

# self.create\_encode\_frame()

# style = ttk.Style()

# style.configure("Rounded.TButton",

# padding=10,

# 

# self.label\_file\_size.pack(pady=5)

# relief="flat",

# background="#2196F3",

# foreground="black",

# font=('Arial', 12),

# borderwidth=5)

# style.map("Rounded.TButton",

# background=[("active", "#1976D2")],

# foreground=[("active", "black")])

# def create\_encode\_frame(self):

# self.clear\_frame()

# self.current\_frame = tk.Frame(self.root, bg="#ADD8E6")

# self.current\_frame.pack(padx=10, pady=(20, 10))

# tk.Label(self.current\_frame, text="Выберите аудиофайл:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.button\_browse\_cover = ttk.Button(self.current\_frame, text="Обзор", command=self.browse\_cover\_file)

# self.button\_browse\_cover.pack(pady=5)

# tk.Label(self.current\_frame, text="Выберите файл с сообщением:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.button\_browse\_msg = ttk.Button(self.current\_frame, text="Обзор", command=self.browse\_msg\_file)

# self.button\_browse\_msg.pack(pady=5)

# tk.Label(self.current\_frame, text="Размер сообщения (в байтах):", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.label\_file\_size = tk.Label(self.current\_frame, text="", font=('Arial', 12), bg="#ADD8E6")

# tk.Label(self.current\_frame, text="Введите количество LSB:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.entry\_lsb = tk.Entry(self.current\_frame, font=('Arial', 12))

# self.entry\_lsb.pack(pady=5)

# self.button\_encode = ttk.Button(self.current\_frame, text="Зашифровать", style="Rounded.TButton", command=self.encode\_message)

# self.button\_encode.pack(pady=10)

# self.switch\_button = ttk.Button(self.current\_frame, text="Перейти к расшифровке", style="Rounded.TButton", command=self.create\_decode\_frame)

# self.switch\_button.pack(pady=5)

# def create\_decode\_frame(self):

# self.clear\_frame()

# self.current\_frame = tk.Frame(self.root, bg="#ADD8E6")

# self.current\_frame.pack(padx=10, pady=(20, 10))

# tk.Label(self.current\_frame, text="Выберите аудиофайл:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.button\_browse\_stego = ttk.Button(self.current\_frame, text="Обзор", command=self.browse\_stego\_file)

# self.button\_browse\_stego.pack(pady=5)

# tk.Label(self.current\_frame, text="Введите количество LSB:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.entry\_lsb = tk.Entry(self.current\_frame, font=('Arial', 12))

# 

# self.entry\_lsb.pack(pady=5)

# tk.Label(self.current\_frame, text="Введите размер данных в байтах:", font=('Arial', 12), bg="#ADD8E6").pack(pady=5)

# self.entry\_size = tk.Entry(self.current\_frame, font=('Arial', 12))

# self.entry\_size.pack(pady=5)

# self.button\_decode = ttk.Button(self.current\_frame, text="Расшифровать", style="Rounded.TButton", command=self.decode\_message)

# self.button\_decode.pack(pady=10)

# self.switch\_button = ttk.Button(self.current\_frame, text="Перейти к шифрованию", style="Rounded.TButton", command=self.create\_encode\_frame)

# self.switch\_button.pack(pady=5)

# def browse\_cover\_file(self):

# global cover\_path

# cover\_path = filedialog.askopenfilename(

# title="Выберите аудиофайл",

# filetypes=(("WAV files", "\*.wav"), ("All files", "\*.\*"))

# )

# def browse\_msg\_file(self):

# global msg\_path

# msg\_path = filedialog.askopenfilename(

# title="Выберите текстовый файл",

# return

# filetypes=(("Text files", "\*.txt"), ("All files", "\*.\*"))

# )

# if msg\_path:

# with open(msg\_path, 'r', encoding='utf-8') as file:

# message = file.read()

# file\_size\_bytes = len(message.encode('utf-8'))

# self.label\_file\_size.config(text=f"{file\_size\_bytes} байт")

# def browse\_stego\_file(self):

# global stego\_path

# stego\_path = filedialog.askopenfilename(

# title="Выберите аудиофайл",

# filetypes=(("WAV files", "\*.wav"), ("All files", "\*.\*"))

# )

# def clear\_frame(self):

# if self.current\_frame is not None:

# self.current\_frame.pack\_forget()

# def encode\_message(self):

# if not cover\_path:

# messagebox.showerror("Ошибка", "Сначала выберите аудиофайл!")

# return

# if not msg\_path:

# messagebox.showerror("Ошибка", "Сначала выберите файл с сообщением!")

# res = ''

# 

# nlsb = int(self.entry\_lsb.get())

# with wave.open(cover\_path, "r") as cover:

# with open(msg\_path, 'r', encoding='utf-8') as file:

# msg = file.read()

# print("Size of message in bytes: ", len(msg))

# msg = self.convertMsgToBin(msg)

# print("Length of message in bits: ", len(msg))

# self.stego(cover, msg, nlsb)

# except ValueError:

# messagebox.showerror("Ошибка", "Введите корректное значение для LSB!")

# def decode\_message(self):

# if not stego\_path:

# messagebox.showerror("Ошибка", "Сначала выберите аудиофайл!")

# return

# try:

# nlsb = int(self.entry\_lsb.get())

# size = int(self.entry\_size.get())

# with wave.open(stego\_path, "r") as stego:

# self.extract(stego, nlsb, size)

# except ValueError:

# messagebox.showerror("Ошибка", "Введите корректные значения!")

# def convertMsgToBin(self, m):

# print("frames", frames)

# for i in m:

# x = str(format(ord(i), 'b'))

# x = ('0' \* (8 - len(x))) + x

# res = res + x

# return res

# def frames\_continuous(self, time):

# global rate

# return int(rate \* time)

# def pre(self, cover, nlsb):

# global para, channels, sample\_width, frames, samples, fmt, mask, minByte, rate

# para = cover.getparams()

# channels = cover.getnchannels()

# sample\_width = cover.getsampwidth()

# frames = cover.getnframes()

# rate = cover.getframerate()

# print("\nrate", rate)

# duration = frames / rate

# samples = frames \* channels

# print("parameters", para)

# print("Total rawdata size expected:", frames \* channels \* sample\_width)

# print("channels", channels)

# print("sample\_width", sample\_width)

# slot\_len = curr = int(curr, 2)

# print("samples", samples)

# if channels == 1:

# fmt = str(samples) + "B"

# mask = (1 << 8) - (1 << nlsb)

# minByte = -(1 << 8)

# elif channels == 2:

# fmt = str(samples) + "h"

# mask = (1 << 15) - (1 << nlsb)

# minByte = -(1 << 15)

# else:

# raise ValueError("Number of channels is too large")

# def count\_available\_slots(self, rawdata):

# global minByte

# cnt = 0

# for i in range(len(rawdata)):

# if rawdata[i] != minByte:

# cnt += 1

# return cnt

# def stego(self, cover, msg, nlsb):

# self.pre(cover, nlsb)

# rawdata = list(struct.unpack(fmt, cover.readframes(frames)))

# print(f"Read {len(rawdata)} bytes from the audio file")

# available = self.count\_available\_slots(rawdata)

# self.frames\_continuous(continuous\_duration)

# nslots = math.ceil(len(msg) / (slot\_len \* nlsb))

# print("\nnslots", nslots, "\nslot\_len", slot\_len, "\navailable", available)

# skip = (available - (nslots \* slot\_len)) // (nslots - 1)

# print("skip", skip)

# cover\_ind = 0

# msg\_ind = 0

# res = []

# slot\_ind = 0

# while msg\_ind < len(msg) and cover\_ind < len(rawdata):

# if rawdata[cover\_ind] == minByte:

# res.append(struct.pack(fmt[-1], rawdata[cover\_ind]))

# cover\_ind += 1

# continue

# curr = ""

# while len(curr) < nlsb:

# if msg\_ind < len(msg):

# curr += msg[msg\_ind]

# else:

# curr += "0"

# msg\_ind += 1

# sign = 1

# if rawdata[cover\_ind] < 0:

# rawdata[cover\_ind] \*= -1

# sign = -1

# to\_append = ((rawdata[cover\_ind] & mask) | curr) \* sign

# res.append(struct.pack(fmt[-1], to\_append))

# cover\_ind += 1

# slot\_ind += 1

# if slot\_ind < slot\_len:

# continue

# i = 0

# while i < skip and cover\_ind < len(rawdata):

# if rawdata[cover\_ind] != minByte:

# i += 1

# res.append(struct.pack(fmt[-1], rawdata[cover\_ind]))

# cover\_ind += 1

# slot\_ind = 0

# if msg\_ind < len(msg):

# print("\n\nMessage length too long. Terminating process")

# messagebox.showerror("Ошибка", "Сообщение слишком длинное для выбранного аудиофайла!")

# return 0

# while cover\_ind < len(rawdata):

# res.append(struct.pack(fmt[-1], rawdata[cover\_ind]))

# cover\_ind += 1

# print("\nnslots", nslots, "\nslot\_len", slot\_len,

# steg.setparams(para)

# steg.writeframes(b"".join(res))

# print("\n\nSteganography complete. Data hidden in file", stego\_path)

# messagebox.showinfo("Успех", f"Данные успешно зашифрованы в файл {stego\_path}")

# return 1

# def decimalToBinary(self, n):

# return bin(n).replace("0b", "")

# def extract(self, stego, nlsb, size\_in\_bytes):

# global frames, samples, fmt, minByte

# self.pre(stego, nlsb)

# msg = ""

# stego\_index = 0

# rawdata = list(struct.unpack(fmt, stego.readframes(frames)))

# size = size\_in\_bytes \* 8

# msg\_index = 0

# available = self.count\_available\_slots(rawdata)

# slot\_len = self.frames\_continuous(continuous\_duration)

# nslots = math.ceil(size / (slot\_len \* nlsb))

# 

# "\navailable frames", available)

# skip = (available - (nslots \* slot\_len)) // (nslots - 1)

# print("skip", skip)

# mask = (1 << nlsb) - 1

# slot\_ind = 0

# while msg\_index < size:

# if rawdata[stego\_index] != minByte:

# curr = self.decimalToBinary(abs(rawdata[stego\_index]) & mask)

# msg += ('0' \* (nlsb - len(curr))) + curr

# msg\_index += nlsb

# stego\_index += 1

# slot\_ind += 1

# if slot\_ind < slot\_len:

# continue

# i = 0

# while i < skip:

# if rawdata[stego\_index] != minByte:

# i += 1

# stego\_index += 1

# slot\_ind = 0

# msg = msg[:size]

# val = len(msg) // 8

# dec\_msg = ''.join(chr(int(i, 2)) for i in chunks)

# with open(output\_path, 'w', encoding='utf-8') as file:

# file.write(dec\_msg)

# print("\nThe extracted message is written in", output\_path)

# messagebox.showinfo("Успех", "Сообщение успешно расшифровано и записано в output.txt")

# if \_\_name\_\_ == "\_\_main\_\_":

# root = tk.Tk()

# app = SteganographyApp(root)

# root.mainloop()