

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Сибирский государственный университет телекоммуникаций и информатики»  
(СибГУТИ)

Институт информатики и вычислительной техники

Кафедра прикладной математики и кибернетики

**Лабораторная работа №7**  
**по дисциплине**  
**Прикладная стеганография**

Выполнил:

студент гр.МГ-411

«29» мая 2025 г.

Шевельков П.С.  
ФИО студента

Новосибирск 2025 г.

## Задание на лабораторную работу:

Модифицировать метод встраивания, реализованный в рамках работы № 4.

- Программа для встраивания информации должна брать текст из файла, преобразовывать его в двоичный вид, складывать с псевдослучайным ключом и записывать в изображение-контейнер в максимально возможном объеме. Также программа должна извлекать обратно вложенный текст **без ошибок**.
- Модифицировать метод встраивания, применив стеганографический **код, базирующийся на линейной хэш-функции**.
- Вывести для сравнения, сколько бит максимально получилось встроить в один и тот же контейнер до и после модификации.
- Применить **код добавления избыточности**, оценивая распределение вероятностей бит в областях 8x8 пикселей пустого контейнера перед интерполяцией. Метод оценки вероятностей выбрать самостоятельно.
- Провести сравнительный стегоанализ полученных вариантов метода встраивания с помощью реализованного в работе 5 приложения.
- Подготовить **доклад-выступление** о полученных результатах.
- **Отчет по работе** должен содержать описание применяемых стеганографических методов, описание программы и блок-схему ее работы. Также в отчет необходимо включить полученные результаты на 10 различных контейнерах и ссылку на исходный код программы.

# Результаты работы программы:

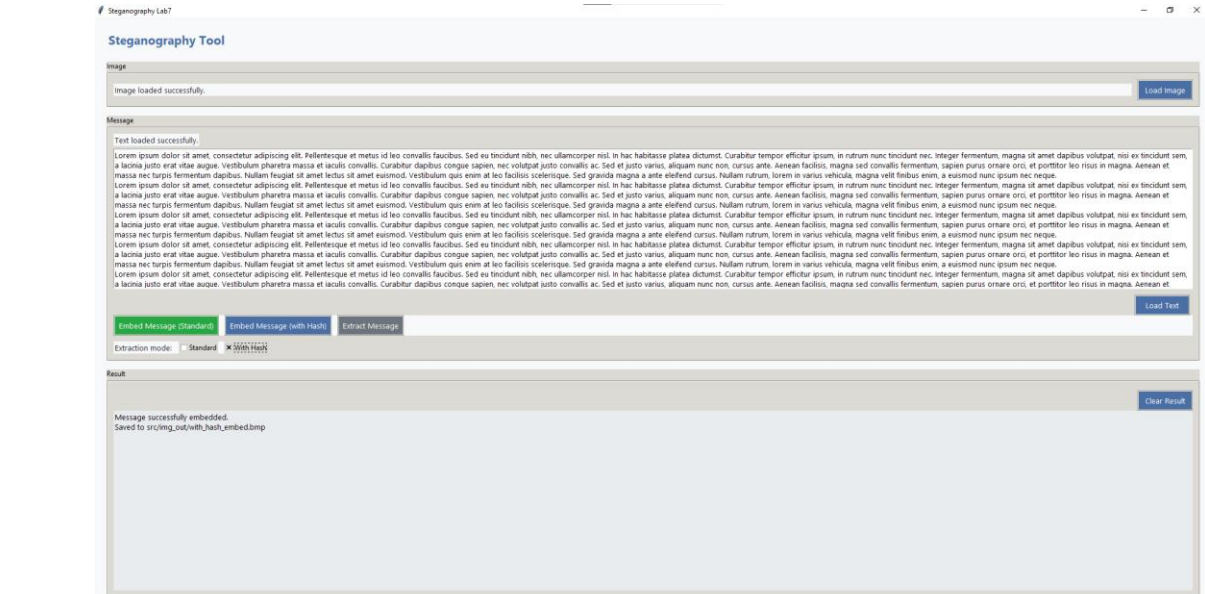


рисунок 1. Встраивание с помощью хэша.



рисунок 2. Извлечение сообщения.



рисунок 3. Попытка извлечь сообщение, зашифрованное с хэшем, обычной интерполяцией.

## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)Embed Message (with Hash)Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp


 Максимально получилось встроить: 87360  
Extrcted message: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non,

рисунок 4. Первый контейнер 512x512

Steganography Lab7

— □ ×

## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)Embed Message (with Hash)Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp


 Максимально получилось встроить: 12344  
Extrcted message: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non,

рисунок 4. Второй контейнер 212x175

## Steganography Tool

### Image

Image loaded successfully.

Load Image

### Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)

Embed Message (with Hash)

Extract Message

Extraction mode: ☐ Standard ☒ With Hash

### Result

Clear Result

Message successfully embedded.

Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 87152

Extrcted message: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non,

рисуюнок 5. Третий контейнер 604x433

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)

Embed Message (with Hash)

Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.

Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 121976

Extrcted message: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum

рисуюнок 6. Четвертый контейнер 489x750

## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)Embed Message (with Hash)Extract Message

Extraction mode: ☒ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 441168

рисунок 7. Пятый контейнер 1680x1120

## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)Embed Message (with Hash)Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 294112

рисунок 8. Шестой контейнер 1430x1424

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)

Embed Message (with Hash)

Extract Message

Extraction mode:
 ☐ Standard
 ☒ With Hash

Result

Clear Result

Message successfully embedded.  
 Saved to src/img\_out/with\_hash\_embed.bmp  
 Максимально получилось встроить: 119976

рисунок 9. Седьмой контейнер 600x600

Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)

Embed Message (with Hash)

Extract Message

Extraction mode:
 ☐ Standard
 ☒ With Hash

Result

Clear Result

Message successfully embedded.  
 Saved to src/img\_out/with\_hash\_embed.bmp  
 Максимально получилось встроить: 16664

рисунок 10. Восьмой контейнер 215x234



## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)
 Embed Message (with Hash)
 Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 20808

рисунок 11. Девятый контейнер 250x250

## Steganography Tool

Image

Image loaded successfully.

Load Image

Message

Text loaded successfully.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque et metus id leo convallis faucibus. Sed eu tincidunt nibh, nec ullamcorper nisl. In hac habitasse platea dictumst. Curabitur tempor efficitur ipsum, in rutrum nunc tincidunt nec. Integer fermentum, magna sit amet dapibus volutpat, nisi ex tincidunt sem, a lacinia justo erat vitae augue. Vestibulum pharetra massa et iaculis convallis. Curabitur dapibus congue sapien, nec volutpat justo convallis ac. Sed et justo varius, aliquam nunc non, cursus ante. Aenean facilisis, magna sed convallis fermentum, sapien purus ornare orci, et porttitor leo risus in magna. Aenean et massa nec turpis fermentum dapibus. Nullam feugiat sit amet lectus sit amet euismod. Vestibulum quis enim at leo facilisis scelerisque.

Load Text

Embed Message (Standard)
 Embed Message (with Hash)
 Extract Message

Extraction mode: ☐ Standard ☒ With Hash

Result

Clear Result

Message successfully embedded.  
Saved to src/img\_out/with\_hash\_embed.bmp

Максимально получилось встроить: 80176

рисунок 12. Десятый контейнер 600x401



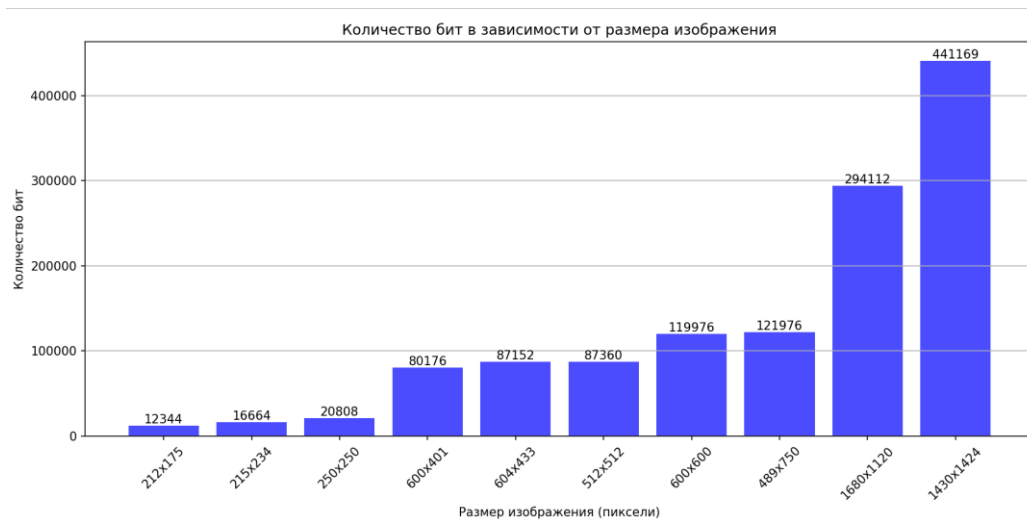


Рисунок 13. График результатов встраивания

## **Описание программы**

### **1. Загрузка изображений:**

- Программа позволяет загружать изображения в формате BMP для дальнейшей обработки.

### **2. Внедрение сообщений:**

- Реализованы два метода внедрения сообщений:
  - **Стандартный метод:** Внедрение битов сообщения в изображение без дополнительных изменений.
  - **Метод с хешированием:** Внедрение битов сообщения с использованием хеш-функции для повышения устойчивости к обнаружению.

### **3. Извлечение сообщений:**

- Программа поддерживает два метода извлечения:
  - **Стандартный метод:** Извлечение битов сообщения из изображения без дополнительных проверок.
  - **Метод с хешированием:** Извлечение битов сообщения с проверкой целостности с помощью хеш-функции.

### **4. Анализ внедрения:**

- Программа проводит анализ внедрения сообщений, включая проверку целостности и корректности извлечённых данных.

### **5. Отображение результатов:**

- Результаты внедрения и извлечения отображаются в виде текстового отчёта, а также графически для наглядности.

### **6. Сохранение результатов:**

- Возможность сохранить результаты анализа и отчёты в файл для дальнейшего использования.

## **Методы внедрения сообщений:**

### **1. Стандартный метод внедрения:**

- Простой и эффективный способ внедрения битов сообщения в изображение, без дополнительных изменений.

### **2. Метод с хешированием:**

- Внедрение битов сообщения с использованием хеш-функции для повышения устойчивости к обнаружению и проверки целостности.

## **Архитектура программы:**

### **1. MainWindow (главное окно):**

- Интерфейс на основе Tkinter.
- Содержит:
  - Виджет для отображения загруженных изображений.
  - Панель управления (выбор метода внедрения, параметры анализа).
  - Отображение результатов внедрения и извлечения.
- Поддерживает асинхронную обработку через использование потоков.

### **2. SteganographyApp:**

- Основной класс приложения, инициализирующий интерфейс и обрабатывающий события.

### 3. Методы обработки:

- `extract_with_hash`: Извлечение битов сообщения с проверкой хеш-функции.
- `embed_with_hash`: Внедрение битов сообщения с использованием хеширования.
- `extract_standard`: Стандартное извлечение битов сообщения.
- `embed_message_standard`: Стандартное внедрение битов сообщения.
- `generate_random_binary_matrix`: Генерация случайной двоичной матрицы для внедрения.
- `save_image`: Сохранение обработанного изображения.
- `load_image`: Загрузка изображения для обработки.
- `load_text`: Загрузка текстового сообщения для внедрения.

### 4. Дополнительные функции:

- Методы для преобразования битов в строку и обратно, а также для выполнения операций с битами (например, XOR).

## Выводы:

### 1. Эффективность внедрения:

Программа демонстрирует высокую эффективность внедрения сообщений в изображения, обеспечивая возможность использования как стандартного метода, так и метода с хешированием. Это позволяет пользователям выбирать подходящий метод в зависимости от требований к безопасности и устойчивости к обнаружению.

### 2. Устойчивость к обнаружению:

Метод с хешированием значительно повышает устойчивость к обнаружению внедрённых сообщений, что делает его предпочтительным для приложений, где важна защита информации.

### 3. Анализ и проверка целостности:

Внедрение механизма проверки целостности с помощью хеш-функции позволяет пользователям уверенно извлекать сообщения, зная, что они не были изменены. Это особенно важно в контексте передачи конфиденциальной информации.

### 4. Удобство использования:

Интуитивно понятный интерфейс на основе Tkinter делает программу доступной для пользователей с различным уровнем подготовки. Возможность загрузки изображений и текстов, а также простота в использовании методов внедрения и извлечения сообщений способствуют широкому применению программы.

### 5. Гибкость и расширяемость:

Архитектура программы позволяет легко добавлять новые методы внедрения и анализа, что открывает возможности для дальнейшего развития и улучшения функциональности. Пользователи могут адаптировать программу под свои нужды, добавляя новые алгоритмы или улучшая существующие.

## **6. Практическое применение:**

Программа может быть использована в различных областях, включая защиту авторских прав, безопасную передачу данных и скрытую коммуникацию. Это делает её полезным инструментом как для исследователей, так и для практиков в области информационной безопасности.

**Ссылка на программу:**

<https://github.com/bothyD/steganograf>

## Листинг:

```
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
import numpy as np
from matplotlib import pyplot as plt
from PIL import Image
import random
from bitarray import bitarray
import os
from collections import Counter

class SteganographyApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Steganography Lab7")
        self.container_image = None
        self.stego_image = None
        self.text_input = None
        self.message_bits = []
        self.key = [1, 0, 1, 1, 0, 1, 0, 0]
        self.A = self.generate_random_binary_matrix(m=4, N=8, seed=42)
        self.extract_mode_standard = tk.BooleanVar(value=True)
        self.extract_mode_hash = tk.BooleanVar(value=False)
        self.colors = {
            "primary": "#4a6fa5",
            "primary_dark": "#3d5d8a",
            "secondary": "#6c757d",
            "bg": "#f8f9fa",
            "text": "#212529",
            "light_accent": "#e9ecef",
            "success": "#28a745",
            "warning": "#ffc107"
        }
        self.style = ttk.Style()
        self.style.theme_use('clam')
        self.configure_style()
        self.setup_ui()

    # --- Интерфейс - визуальная составляющая
    def configure_style(self):
        self.style.configure('TButton', background=self.colors["primary"],
                               foreground='white', font=('Segoe UI', 10), padding=6)
        self.style.map('TButton', background=[('active',
                                                self.colors["primary_dark"])]])
        self.style.configure('TLabel', background=self.colors["bg"],
                               foreground=self.colors["text"], font=('Segoe UI', 10))
        self.style.configure('TEntry',
                               fieldbackground=self.colors["light_accent"], font=('Segoe UI', 10))
        self.style.configure('TFrame', background=self.colors["bg"]])
```

```

        self.style.configure('Primary.TButton',
background=self.colors["primary"], foreground='white')
        self.style.configure('Success.TButton',
background=self.colors["success"], foreground='white')
        self.style.map('Success.TButton', background=[('active', '#218838')])
        self.style.configure('Secondary.TButton',
background=self.colors["secondary"], foreground='white')
        self.style.map('Secondary.TButton', background=[('active', '#5a6268')])

# --- Интерфейс - настройка кнопок/полей
def setup_ui(self):
    self.root.geometry("800x600")
    self.root.configure(bg=self.colors["bg"])

    main_frame = ttk.Frame(self.root, padding="20 20 20 20", style='TFrame')
    main_frame.pack(fill=tk.BOTH, expand=True)

    header_frame = ttk.Frame(main_frame, style='TFrame')
    header_frame.pack(fill=tk.X, pady=(0, 20))

    title_label = tk.Label(header_frame, text="Steganography Tool",
font=('Segoe UI', 16, 'bold'), bg=self.colors["bg"], fg=self.colors["primary"])
    title_label.pack(side=tk.LEFT)
    # --- блок изображения
    file_frame = ttk.LabelFrame(main_frame, text="Image", padding="10 10 10
10")
    file_frame.pack(fill=tk.X, pady=(0, 15))

    self.file_status = tk.StringVar(value="Image not loaded")
    file_status_label = ttk.Label(file_frame, textvariable=self.file_status)
    file_status_label.pack(side=tk.LEFT, fill=tk.X, expand=True)

    load_btn = ttk.Button(file_frame, text="Load Image",
command=self.load_image, style='TButton')
    load_btn.pack(side=tk.LEFT, padx=(10, 0))

    # --- блок текста
    message_frame = ttk.LabelFrame(main_frame, text="Message", padding="10 10
10 10")
    message_frame.pack(fill=tk.BOTH, expand=True, pady=(0, 15))

    self.text_status = tk.StringVar(value="Text not loaded")
    text_status_label = ttk.Label(message_frame,
textvariable=self.text_status)
    text_status_label.pack(anchor=tk.W, pady=(0, 5))

    self.text_display = tk.Text(message_frame, height=6, wrap=tk.WORD,
font=('Segoe UI', 10))
    self.text_display.pack(fill=tk.BOTH, expand=True, pady=(0, 10))

```

```

        load_text_btn = ttk.Button(message_frame, text="Load Text",
command=self.load_text, style='TButton')
        load_text_btn.pack(anchor=tk.E, padx=(0, 5))

        btn_frame = ttk.Frame(message_frame, style='TFrame')
        btn_frame.pack(fill=tk.X)

        embed_standard_btn = ttk.Button(btn_frame, text="Embed Message
(Standard)", command=self.embed_message_standard, style='Success.TButton')
        embed_standard_btn.pack(side=tk.LEFT)

        embed_hash_btn = ttk.Button(btn_frame, text="Embed Message (with Hash)",
command=self.embed_with_hash, style='Primary.TButton')
        embed_hash_btn.pack(side=tk.LEFT, padx=(10, 0))

        extract_btn = ttk.Button(btn_frame, text="Extract Message",
command=self.extract_message, style='Secondary.TButton')
        extract_btn.pack(side=tk.LEFT, padx=(10, 0))

        result_frame = ttk.LabelFrame(main_frame, text="Result", padding="10 10
10 10")
        result_frame.pack(fill=tk.BOTH, expand=True)

        self.output_text = tk.Text(result_frame, wrap=tk.WORD,
bg=self.colors["light_accent"], relief=tk.FLAT, font=('Segoe UI', 10), height=10)
        # Кнопка очистки результата
        clear_result_btn = ttk.Button(result_frame, text="Clear Result",
command=lambda: self.output_text.delete(1.0, tk.END), style='Danger.TButton')
        clear_result_btn.pack(anchor=tk.E, pady=(5, 0))
        self.output_text.pack(fill=tk.BOTH, expand=True)
        self.output_text.config(state='normal')

        self.output_text.bind("<Control-c>", lambda e:
self.root.clipboard_append(self.output_text.selection_get()))

        # -----
        mode_frame = ttk.Frame(message_frame, style='TFrame')
        mode_frame.pack(anchor=tk.W, pady=(10, 0))

        ttk.Label(mode_frame, text="Extraction mode:").pack(side=tk.LEFT,
padx=(0, 10))

        ttk.Checkbutton(
            mode_frame, text="Standard", variable=self.extract_mode_standard
        ).pack(side=tk.LEFT, padx=(0, 10))

        ttk.Checkbutton(
            mode_frame, text="With Hash", variable=self.extract_mode_hash
        ).pack(side=tk.LEFT)

```



```

# --- изображение
def load_image(self):
    path = filedialog.askopenfilename(
        initialdir="src/img_in",
        filetypes=[("Image files", "*.png *.bmp")])
    if path:
        try:
            image = Image.open(path).convert('L')
            image_array = np.array(image)
            self.container_image = image_array.copy()
            self.stego_image = image_array.copy() # сохраняем исходное
изображение
            self.file_status.set("Image loaded successfully.")

        except Exception as e:
            messagebox.showerror("Error", f"Failed to load image: {e}")

# --- текст
def load_text(self):
    path = filedialog.askopenfilename(
        initialdir="src/texts",
        filetypes=[("Text files", "*.txt")])
    if path:
        try:
            with open(path, 'r', encoding='utf-8') as file:
                self.text_input = file.read()
                self.text_display.delete(1.0, tk.END) # очистить поле
                self.text_display.insert(tk.END, self.text_input) # вставить
текст
            self.text_status.set("Text loaded successfully.")

        except Exception as e:
            messagebox.showerror("Error", f"Failed to load text: {e}")

def save_image(self, array, nameOutImg):
    # 4. Сохраняем изображение
    output_dir = "src/img_out/"
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    output_dir = os.path.join(output_dir, nameOutImg+".bmp")
    image = Image.fromarray(array.astype(np.uint8))
    image.save(output_dir)
    self.output_text.insert(tk.END, f"Message successfully embedded.\nSaved to
{output_dir}"+'\n')

# --- Битовые утилиты ---
def str_to_bits(self, s):
    return [int(b) for c in s for b in format(ord(c), '08b')]

```

```

def bits_to_str(self, bits):
    chars = []
    for i in range(0, len(bits), 8):
        byte = bits[i:i+8]
        if len(byte) < 8:
            break
        chars.append(chr(int(''.join(map(str, byte)), 2)))
    return ''.join(chars)

def xor_bits(self, bits, key):
    return [b ^ key[i % len(key)] for i, b in enumerate(bits)]

def get_full_bits(self, message_bits):
    length_bits = [int(b) for b in format(len(message_bits), '032b')]
    return length_bits + message_bits

def extracted_bits_message(self, extracted_bits):
    extracted_len = int("".join(map(str, extracted_bits[:32])), 2)
    print(f"Extracted length: {extracted_len}")
    return extracted_bits[32:32 + extracted_len]

# --- Хэш-функция (линейная) ---
def linear_hash_matrix(self, block, A):
    x = np.array(block, dtype=int)
    result = (A @ x) % 2
    return result.tolist()

# --- Линейная хэш-функция: lambda(x) = A * x mod 2 ---
def generate_random_binary_matrix(self, m, N, seed=None):
    if seed is not None:
        np.random.seed(seed)
    return np.random.randint(0, 2, size=(m, N), dtype=int)

def add_matrix_hash_blocks(self, bits, block_size=8, A=None):
    assert A is not None, "Матрица A обязательна"
    m = A.shape[0]
    hashed = []

    for i in range(0, len(bits), block_size):
        block = bits[i:i+block_size]
        if len(block) < block_size:
            break
        h = self.linear_hash_matrix(block, A)
        hashed.extend(block + h)

    return hashed

def check_and_extract_matrix_hash_blocks(self, bits, block_size=8, A=None):
    assert A is not None, "Матрица A обязательна"
    m = A.shape[0]
    recovered = []

```

```

for i in range(0, len(bits), block_size + m):
    block = bits[i:i+block_size]
    h = bits[i+block_size:i+block_size+m]
    if len(block) < block_size or len(h) < m:
        break
    expected_h = self.linear_hash_matrix(block, A)
    if h == expected_h:
        recovered.extend(block)
return recovered

# --- Встраивание интерполяцией
def interpolation_method(self, img_array, full_message_bits):
    height, width = img_array.shape
    stego_array = img_array.copy()
    idx = 0
    total_bits = len(full_message_bits)

    for i in range(height):
        for j in range(0, width - 1, 2):
            if idx >= total_bits:
                return stego_array

            p1 = int(stego_array[i, j])
            p2 = int(stego_array[i, j + 1])
            mid = (p1 + p2) // 2
            desired_bit = full_message_bits[idx]

            current_bit = mid % 2
            if current_bit != desired_bit:
                if mid % 2 == 0:
                    mid += 1
                else:
                    mid -= 1

            candidates = []
            for delta1 in range(-4, 5):
                for delta2 in range(-4, 5):
                    np1 = p1 + delta1
                    np2 = p2 + delta2
                    if 0 <= np1 <= 255 and 0 <= np2 <= 255 and (np1 +
np2) // 2 == mid:

                        diff = abs(delta1) + abs(delta2)
                        candidates.append((diff, np1, np2))

            if candidates:
                _, new_p1, new_p2 = min(candidates)
                stego_array[i, j] = new_p1

```

```

        stego_array[i, j + 1] = new_p2

        idx += 1

    return stego_array

def embed_message_standard(self):
    if self.stego_image is None:
        messagebox.showerror("Error", "Please load an image first.")
        return
    message = self.text_input
    if not message:
        messagebox.showerror("Error", "Please enter a message.")
        return

    message_bits = self.str_to_bits(message)

    full_bits = self.get_full_bits(message_bits)

    self.stego_image = self.interpolation_method(self.container_image,
full_bits)
    # --- save and notify
    self.save_image(self.stego_image, "standard_embed")

def embed_with_hash(self):
    if self.stego_image is None:
        messagebox.showerror("Error", "Please load an image first.")
        return
    message = self.text_input
    if not message:
        messagebox.showerror("Error", "Please enter a message.")
        return

    message_bits = self.str_to_bits(message)
    encrypted_bits = self.xor_bits(message_bits, self.key)
    hashed_bits = self.add_matrix_hash_blocks(encrypted_bits, 8, self.A)
    full_bits_hashed = self.get_full_bits(hashed_bits)
    self.stego_image = self.interpolation_method(self.container_image,
full_bits_hashed)
    # --- save and notify
    self.save_image(self.stego_image, "with_hash_embed")

# --- Извлечение сообщения
def extract_message(self):
    if self.stego_image is None:
        messagebox.showerror("Error", "No stego image available.")
        return
    use_standard = self.extract_mode_standard.get()
    use_hash = self.extract_mode_hash.get()

    if not (use_standard or use_hash):

```

```

        messagebox.showerror("Error", "Please select at least one extraction
mode.")
    return
    extracted_bits = self.extract_bits_form_stego(self.stego_image)

    if use_standard:
        self.extract_standard(extracted_bits)
    else:
        self.extract_with_hash(extracted_bits, self.key)

def extract_bits_form_stego(self, stego_array):
    height, width = stego_array.shape
    bits = []
    for i in range(height):
        for j in range(0, width - 1, 2):
            p1 = int(stego_array[i, j])
            p2 = int(stego_array[i, j + 1])
            interp = (p1 + p2) // 2
            bits.append(interp % 2)

    return bits

def extract_standard(self, extracted_bits):
    extrcted_bit = self.extracted_bits_message(extracted_bits)
    self.output_text.insert(tk.END, '\n[ ] Максимально получилось встроить:
'+ str(len(extrcted_bit))+'\n')
    extracted_msg = self.bits_to_str(extrcted_bit)
    print("len extract message: ", len(extracted_msg))
    self.output_text.insert(tk.END, '\nExtrcted message:
'+extracted_msg+'\n')

def extract_with_hash(self, extracted_bits, key):
    raw_encrypted = self.extracted_bits_message(extracted_bits)
    recovered_encrypted =
self.check_and_extract_matrix_hash_blocks(raw_encrypted, 8, self.A)
    decrypted = self.xor_bits(recovered_encrypted, key)
    self.output_text.insert(tk.END, '\n[ ] Максимально получилось встроить:
'+ str(len(decrypted))+'\n')

    extracted_msg = self.bits_to_str(decrypted)
    print("len extract message: ", len(extracted_msg))
    self.output_text.insert(tk.END, 'Extrcted message: '+extracted_msg+'\n')

if __name__ == "__main__":
    root = tk.Tk()
    app = SteganographyApp(root)
    root.mainloop()

```

