Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

Лабораторная работа 5

по дисциплине «Прикладная стеганография»

Выполнил: ст. гр. ЗМП-41 Лёвкин И. А.

Проверила: Мерзлякова Е.Ю.

Новосибирск 2025

**Описание метода**

Программа реализует метод стеганографии для скрытой передачи информации в текстовых файлах. Используется техника пробельных символов, где биты секретного сообщения кодируются с помощью добавления или отсутствия двойного пробела в конце строк текста-контейнера.

**Принцип работы метода:**

* Секретное сообщение преобразуется в бинарную последовательность (последовательность нулей и единиц)
* Каждый бит сообщения сопоставляется с одной строкой текста-контейнера
* Если бит равен '1', в конец соответствующей строки добавляется двойной пробел
* Если бит равен '0', строка остается без изменений
* Для извлечения сообщения анализируются окончания строк - наличие двойного пробела означает '1', его отсутствие - '0'

**Преимущества метода:**

* Простота реализации
* Сообщение визуально незаметно при чтении
* Не требует изменения значимого содержимого текста

**Ограничения:**

* Требует достаточно длинного текста-контейнера
* Добавляет незначительные изменения в форматирование текста
* Не обеспечивает криптографической защиты (только скрытие факта передачи)

**Обзор основных функций программы**

1. encode\_message()**– кодирование сообщения в текст**

**Назначение:**  
Встраивает скрытое сообщение в текст-контейнер с помощью метода стеганографии (пробельные символы в конце строк).

**Логика работы:**

1. **Проверка входных данных:**
   * Если сообщение пустое – выводится предупреждение.
   * Если текст-контейнер не загружен – выводится ошибка.
2. **Преобразование сообщения в бинарный вид:**
   * Каждый символ сообщения конвертируется в 8-битный ASCII-код.
   * Например, символ 'A' (ASCII 65) превращается в 01000001.
3. **Проверка длины текста:**
   * Если строк в тексте меньше, чем битов в сообщении – выводится ошибка (не хватает места для встраивания).
4. **Встраивание битов в текст:**
   * Для каждого бита сообщения:
     + Если бит '1' – добавляет два пробела в конец строки.
     + Если бит '0' – оставляет строку без изменений.
   * Строки, не участвующие в кодировании, остаются нетронутыми.
5. **Вывод результата:**
   * Полученный текст с внедрённым сообщением отображается в интерфейсе.

**Обработка ошибок:**

* Если возникает исключение (например, некорректный текст), выводится сообщение об ошибке.

1. decode\_message()**– извлечение скрытого сообщения**

**Назначение:**  
Извлекает закодированное сообщение из текста, анализируя пробелы в конце строк.

**Логика работы:**

1. **Проверка входных данных:**
   * Если текст не загружен – выводится предупреждение.
2. **Анализ строк текста:**
   * Каждая строка проверяется на наличие двойного пробела в конце:
     + " " (два пробела) → бит '1'.
     + Нет двойного пробела → бит '0'.
   * Формируется бинарная строка (последовательность 0 и 1).
3. **Преобразование бинарной строки в текст:**
   * Бинарная строка разбивается на 8-битные блоки (байты).
   * Каждый байт конвертируется в символ ASCII.
4. **Вывод результата:**
   * Извлечённое сообщение отображается в интерфейсе.

**Обработка ошибок:**

* Если возникает ошибка (например, некорректный формат текста), выводится сообщение об ошибке.

**Листинг**

import random

import sys

from PyQt6.QtWidgets import (

QApplication,

QMainWindow,

QTabWidget,

QWidget,

QVBoxLayout,

QHBoxLayout,

QLabel,

QTextEdit,

QLineEdit,

QPushButton,

QFileDialog,

QMessageBox,

)

from PyQt6.QtCore import Qt, QRegularExpression

from PyQt6.QtGui import QTextCursor, QRegularExpressionValidator

import urllib.request

class SecretMessageApp(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Встраивание секретного сообщения")

self.setGeometry(100, 100, 920, 780)

self.central\_widget = QWidget()

self.setCentralWidget(self.central\_widget)

self.main\_layout = QVBoxLayout(self.central\_widget)

self.main\_layout.setContentsMargins(10, 10, 10, 10)

self.create\_interface()

def create\_interface(self):

# Create tab widget

self.tabs = QTabWidget()

self.main\_layout.addWidget(self.tabs)

self.create\_encode\_tab()

self.create\_decode\_tab()

def \_load\_from\_gutenberg(self):

GUTENBERG\_BOOKS = [

"https://dev.gutenberg.org/files/1342/1342-0.txt", # Pride and Prejudice

"https://dev.gutenberg.org/files/11/11-0.txt", # Alice's Adventures in Wonderland

"https://dev.gutenberg.org/files/2701/2701-0.txt", # Moby Dick

"https://dev.gutenberg.org/files/84/84-0.txt", # Frankenstein

"https://dev.gutenberg.org/files/98/98-0.txt", # A Tale of Two Cities

]

try:

url = random.choice(GUTENBERG\_BOOKS)

self.source\_text.setPlainText("Идет загрузка текста...")

QApplication.processEvents()

with urllib.request.urlopen(url) as response:

text = response.read().decode("utf-8")

lines = text.split("\n")

source\_text = "\n".join(lines[:2000])

self.source\_text.setPlainText(source\_text)

except Exception as e:

QMessageBox.critical(

self, "Ошибка", f"Не удалось загрузить текст: {str(e)}"

)

def create\_encode\_tab(self):

self.tab1 = QWidget()

self.tabs.addTab(self.tab1, "Встраивание")

layout = QVBoxLayout(self.tab1)

layout.setContentsMargins(10, 10, 10, 10)

layout.setSpacing(10)

# Source text section

source\_label = QLabel("Текст (контейнер):")

layout.addWidget(source\_label)

self.source\_text = QTextEdit()

layout.addWidget(self.source\_text)

# Message section

message\_label = QLabel("Сообщение:")

layout.addWidget(message\_label)

self.message\_entry = QLineEdit()

layout.addWidget(self.message\_entry)

# Buttons

btn\_layout = QHBoxLayout()

btn\_layout.setSpacing(5)

self.process\_btn = QPushButton("Встроить")

self.process\_btn.clicked.connect(self.encode\_message)

btn\_layout.addWidget(self.process\_btn)

self.load\_source\_btn = QPushButton("Открыть файл")

self.load\_source\_btn.clicked.connect(self.load\_source\_file)

btn\_layout.addWidget(self.load\_source\_btn)

self.random\_btn = QPushButton("Случайный")

self.random\_btn.clicked.connect(self.\_load\_from\_gutenberg)

btn\_layout.addWidget(self.random\_btn)

layout.addLayout(btn\_layout)

# Result section

result\_label = QLabel("Зашифрованный текст:")

layout.addWidget(result\_label)

self.result\_text = QTextEdit()

layout.addWidget(self.result\_text)

# Result buttons

result\_btn\_layout = QHBoxLayout()

result\_btn\_layout.setSpacing(5)

self.copy\_btn = QPushButton("Копировать")

self.copy\_btn.clicked.connect(self.copy\_result)

result\_btn\_layout.addWidget(self.copy\_btn)

# self.reset\_btn = QPushButton("Reset")

# self.reset\_btn.clicked.connect(self.reset\_fields)

# result\_btn\_layout.addWidget(self.reset\_btn)

self.save\_btn = QPushButton("Сохранить")

self.save\_btn.clicked.connect(self.save\_result)

result\_btn\_layout.addWidget(self.save\_btn)

layout.addLayout(result\_btn\_layout)

def create\_decode\_tab(self):

self.tab2 = QWidget()

self.tabs.addTab(self.tab2, "Извлечение")

layout = QVBoxLayout(self.tab2)

layout.setContentsMargins(10, 10, 10, 10)

layout.setSpacing(10)

# Encoded text section

encoded\_label = QLabel("Зашифрованный текст:")

layout.addWidget(encoded\_label)

self.encoded\_text = QTextEdit()

layout.addWidget(self.encoded\_text)

# Buttons

btn\_layout1 = QHBoxLayout()

btn\_layout1.setSpacing(5)

self.open\_btn = QPushButton("Открыть")

self.open\_btn.clicked.connect(self.load\_encoded\_file)

btn\_layout1.addWidget(self.open\_btn)

self.paste\_btn = QPushButton("Вставить")

self.paste\_btn.clicked.connect(self.paste\_text)

btn\_layout1.addWidget(self.paste\_btn)

layout.addLayout(btn\_layout1)

# Extract buttons

btn\_layout2 = QHBoxLayout()

btn\_layout2.setSpacing(5)

self.extract\_btn = QPushButton("Извлечь")

self.extract\_btn.clicked.connect(self.decode\_message)

btn\_layout2.addWidget(self.extract\_btn)

self.clear\_btn = QPushButton("Очистить")

self.clear\_btn.clicked.connect(self.clear\_fields)

btn\_layout2.addWidget(self.clear\_btn)

layout.addLayout(btn\_layout2)

# Decoded message

decoded\_label = QLabel("Извлечённое сообщение:")

layout.addWidget(decoded\_label)

self.decoded\_msg = QLineEdit()

layout.addWidget(self.decoded\_msg)

def paste\_text(self):

clipboard = QApplication.clipboard()

text = clipboard.text()

if text:

self.encoded\_text.setPlainText(text)

else:

QMessageBox.warning(self, "Warning", "Clipboard is empty")

def load\_source\_file(self):

filename, \_ = QFileDialog.getOpenFileName(

self, "Open File", "", "Text Files (\*.txt)"

)

if filename:

try:

with open(filename, "r", encoding="utf-8") as f:

content = f.read()

self.source\_text.setPlainText(content)

except Exception as e:

QMessageBox.critical(self, "Error", f"Can't load file: {str(e)}")

def load\_encoded\_file(self):

filename, \_ = QFileDialog.getOpenFileName(

self, "Open File", "", "Text Files (\*.txt)"

)

if filename:

try:

with open(filename, "r", encoding="utf-8") as f:

content = f.read()

self.encoded\_text.setPlainText(content)

except Exception as e:

QMessageBox.critical(self, "Error", f"Can't load file: {str(e)}")

def encode\_message(self):

message = self.message\_entry.text()

if not message:

QMessageBox.warning(self, "Warning", "Enter message to hide")

return

source = self.source\_text.toPlainText()

if not source.strip():

QMessageBox.warning(self, "Warning", "Load source text first")

return

try:

binary = "".join(f"{ord(c):08b}" for c in message)

lines = source.split("\n")

if len(lines) < len(binary):

QMessageBox.critical(self, "Error", "Text too short for message")

return

result = []

for i, line in enumerate(lines):

if i < len(binary):

result.append(line.rstrip() + (" " if binary[i] == "1" else ""))

else:

result.append(line)

self.result\_text.setPlainText("\n".join(result))

except Exception as e:

QMessageBox.critical(self, "Error", f"Encoding error: {str(e)}")

def decode\_message(self):

text = self.encoded\_text.toPlainText()

if not text.strip():

QMessageBox.warning(self, "Warning", "Load encoded text first")

return

try:

lines = text.split("\n")

binary = "".join("1" if line.endswith(" ") else "0" for line in lines)

message = ""

for i in range(0, len(binary), 8):

byte = binary[i : i + 8]

if len(byte) == 8:

message += chr(int(byte, 2))

self.decoded\_msg.setText(message)

except Exception as e:

QMessageBox.critical(self, "Error", f"Decoding error: {str(e)}")

def copy\_result(self):

text = self.result\_text.toPlainText()

if text.strip():

clipboard = QApplication.clipboard()

clipboard.setText(text)

QMessageBox.information(self, "Info", "Copied to clipboard")

def save\_result(self):

text = self.result\_text.toPlainText()

if not text.strip():

QMessageBox.warning(self, "Warning", "Nothing to save")

return

filename, \_ = QFileDialog.getSaveFileName(

self, "Save File", "", "Text Files (\*.txt)"

)

if filename:

try:

with open(filename, "w", encoding="utf-8") as f:

f.write(text)

QMessageBox.information(self, "Info", "File saved")

except Exception as e:

QMessageBox.critical(self, "Error", f"Can't save file: {str(e)}")

def reset\_fields(self):

self.source\_text.clear()

self.message\_entry.clear()

self.result\_text.clear()

def clear\_fields(self):

self.encoded\_text.clear()

self.decoded\_msg.clear()

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

app = QApplication(sys.argv)

window = SecretMessageApp()

window.show()

sys.exit(app.exec())