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.

---

2. 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())
```