МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

СТЕГАНОГРАФІЧНА СИСТЕМА НА ОСНОВІ ОБРОБЛЕННЯ КОМП'ЮТЕРНИХ ШРИФТІВ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 401.2010316

Виконав студент 4-го курсу, групи 401
_____О. В. Скворцов
«20» червня 2024 р.

Керівник: д-р техн. наук, проф.
_____І. М. Журавська
«20» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Скворцову Олександру В'ячеславовичу.

1. Тема кваліфікаційної роботи «Стеганографічна система на основі оброблення комп'ютерних шрифтів».

Керівник роботи Журавська Ірина Миколаївна, д-р техн. наук, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «20» червня 2024 р.

3. Вхідні (початкові) дані до роботи: відкриті джерела інформації, включаючи наукові статті, технічну документацію та інтернет-ресурси.

Очікуваний результат: система для приховування та відновлення інформації на основі комп'ютерних шрифтів.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- визначення цілей та завдань дослідження, опис об'єкта та предмету;
- огляд історії розвитку стеганографії та аналіз основних напрямів;
- класифікація стеганографічних методів та їх огляд;
- аналіз взаємозв'язку між стеганографією, криптографією та обфускацією.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Правила та вимоги охорони праці при роботі з комп'ютерним обладнанням»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __д-р техн. наук, проф. Журавська І. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Скворцов О. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_14_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Стеганографічна система на основі оброблення комп'ютерних шрифтів_____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	
7	Виконання КРБ: аналіз сучасного стану стеганографії, огляд існуючих технологій, розробка ПЗ для приховування даних з використанням шрифтів.	13.05.2024	22.06.2024	
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	
12	Захист КРБ перед екзаменаційною комісією (ЕК)	27.06.2024	27.06.2024	

Розробив студент __Скворцов О. В._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ д-р техн. наук, проф. Журавська І. М. _____
(*посада, прізвище, ім'я, по батькові)*
(підпис)

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра студента групи 401 ЧНУ ім. Петра Могили

Скворцова Олександра В'ячеславовича

Тема: **«СТЕГАНОГРАФІЧНА СИСТЕМА НА ОСНОВІ ОБРОБЛЕННЯ КОМП'ЮТЕРНИХ ШРИФТІВ»**

Актуальність теми обумовлена зростаючою потребою у захисті інформації в умовах постійної загрози кібератак та несанкціонованого доступу.

Об'єкт роботи – процес приховування інформації за допомогою стеганографії.

Предмет роботи – стеганографічна система, яка використовує комп'ютерні шрифти для приховування даних.

Метою роботи є забезпечення захисту текстових даних шляхом приховування повідомлень у текстових документах за допомогою стеганографічної системи, заснованої на обробленні комп'ютерних шрифтів. Це передбачає створення комплексного підходу, який інтегрує криптографічні методи з технологіями цифрового дизайну шрифтів.

Пояснювальна записка складається зі вступу, трьох розділів, висновків, переліку джерел посилання та шести додатків.

У першому розділі аналізується предметна область, визначається актуальність поставленої задачі. У другому розділі представлено теоретичну частину стосовно стеганографії, порівняння стеганографії з іншими схожими методами, розглянуто реальні атаки з використанням стеганографії та методи їх протидії. У третьому розділі детально розглядаються та описуються використані для створення стеганографічної системи методи та результати роботи застосунку.

Кваліфікаційна робота бакалавра містить 65 сторінок, 21 рисунок, 5 таблиць, 21 літературне джерело та 6 додатків.

***Ключові слова**: приховування даних, стеганографічна система, текстовий стегоконтейнер, оброблення комп'ютерних шрифтів, згорткова нейронна мережа, Python.*

ABSTRACT

to the Bachelor’s thesis by the student of the group 401 of Petro Mohyla Black Sea National University

Skvortsov Oleksandr

«STEGANOGRAPHIC SYSTEM BASED ON COMPUTER FONT PROCESSING»

Supervisor: Doctor of Technical Sciences, Professor Zhuravska Iryna Mykolaivna

The relevance of the topic is driven by the growing need for information protection in the face of constant threats from cyberattacks and unauthorized access.

The object of the study is the process of concealing information using steganography.

The subject of the study is a steganographic system that uses computer fonts to hide data.

The aim of the research is to ensure the protection of text data by hiding messages in text documents due to development the steganographic system based on the computer fonts processing. This involves creating a comprehensive approach that integrates cryptographic methods with digital font design technologies.

The explanatory note consists of an introduction, three chapters, conclusions, references and six appendices.

In the first chapter, the subject area is analyzed, and the relevance of the posed problem is determined. The second chapter presents the theoretical part concerning steganography, compares steganography with other similar methods, examines real attacks using steganography, and methods to counter them. The third chapter details and describes the methods used to create the steganographic system, and demonstration of the application works.

The bachelor's qualification work contains 65 pages, 21 figures, 5 tables, 21 references and 6 appendices.

Keywords: *data hiding, steganographic system, text stegocontainer, computer font processing, Convolutional Neural Network, Python.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІзОД	– інформація з обмеженим доступом
НСД	– несанкціонований доступ
ПЗ	– програмне забезпечення
AR	– Augmented Reality
IDE	– Integrated Development Environment
HD	– High Definition
NFT	– Non-Fungible Token
PenTest	– Penetration Test
VR	– Virtual Reality
CNN	– Convolutional Neural Network

ВСТУП

Стеганографічні системи для приховування текстової інформації відображають актуальний напрям у сфері інформаційної безпеки. Стеганографія, що означає буквально «прихований запис», є методом приховування повідомлень в межах іншого об'єкта або медіа, забезпечуючи тим самим конфіденційність передачі інформації. Розвиток цифрових технологій значно розширив можливості стеганографії, зокрема через використання цифрових шрифтів як носіїв прихованих повідомлень. Використання комп'ютерних шрифтів як медіа для стеганографії є інноваційним підходом, який дозволяє вбудовувати секретні дані безпосередньо в текстові документи, роблячи ці дані непомітними для стороннього спостерігача без спеціальних знань або інструментів.

Актуальність цієї теми обумовлена зростаючою потребою у захисті інформації в умовах постійної загрози кібератак та несанкціонованого доступу (НСД). Методи стеганографії на основі шрифтів можуть бути застосовані в різних областях: від корпоративної безпеки до захисту особистої інформації в інтернеті. Вони забезпечують додатковий рівень захисту, оскільки інформація, зашифрована за допомогою таких методів, залишається прихованою навіть при потенційному перехопленні текстових повідомлень або документів. Крім того, розвиток цього напрямку сприяє пошуку нових методів криптографічного захисту, відкриваючи шлях для інновацій у сфері безпеки даних.

Метою роботи є забезпечення захисту текстових даних шляхом приховування повідомлень у текстових документах за допомогою стеганографічної системи, заснованої на обробленні комп'ютерних шрифтів. Це передбачає створення комплексного підходу, який інтегрує криптографічні методи з технологіями цифрового дизайну шрифтів.

Дослідження прагне не лише розширити теоретичні знання у сфері стеганографії, але й запропонувати практичні рішення для забезпечення конфіденційності інформації у цифровому світі, зокрема в контексті поширення

електронного документообігу та потреби у захисті інтелектуальної власності.

Завдання, які потрібно виконати для досягнення мети роботи:

– проаналізувати аналоги – існуючі системи для приховування тексту;

– обґрунтувати вибір середовища програмування для розробки стегосистеми;

– створити програмний застосунок описаної стегосистеми;

– розробити графічний інтерфейс для розробленого застосунку;

– провести тестування та проаналізувати властивості створеної стегосистеми.

Об'єктом дослідження є процес приховування текстової інформації в стеганографічних системах, які використовують комп'ютерні шрифти. Цей напрям включає в себе технології та методи, що дозволяють інкорпорувати повідомлення в структуру цифрових текстових документів, зокрема через модифікацію візуальних аспектів шрифтів, таких як форма, розмір символів та інтервали, без зміни візуальної сприйнятності тексту для звичайного користувача.

Предметом дослідження є методи та процеси, за допомогою яких відбувається оброблення таких шрифтів та їхнє використання для стеганографічного приховування інформації. Це охоплює аналіз алгоритмів вбудовування та вилучення даних, оцінку криптостійкості прихованих повідомлень до атак на лінгвістичну стегосистему, а також розробку методик для забезпечення максимальної ефективності та безпеки передачі інформації з обмеженим доступом (ІзОД). Важливим аспектом є також дослідження потенційних застосувань цих технологій у різноманітних сферах – від корпоративної безпеки до захисту особистих даних.

Оскільки сучасний світ все більше залежить від цифрових комунікацій, значення та вплив таких досліджень лише зростатимуть, роблячи розробку безпечних і надійних методів приховування інформації критично важливою для забезпечення конфіденційності в цифровому столітті.

1 АНАЛІЗ ВИМОГ ДО ЛІНГВІСТИЧНИХ СТЕГОСИСТЕМ ТА ІНСТРУМЕНТІВ ДЛЯ ЇХ РОЗРОБКИ

1.1 Опис предметної області та актуальність

Стеганографія, як наука приховування інформації, має давню історію, яка налічує тисячоліття. Її головна мета полягає в тому, щоб передати інформацію таким чином, щоб сам факт передачі залишився непомітним для сторонніх осіб. З розвитком інформаційних технологій стеганографічні методи також еволюціонували, інтегруючись з різноманітними цифровими носіями, такими як зображення, аудіо, відео та текстові файли.

Одним з напрямів стеганографії є використання комп'ютерних шрифтів для приховання даних. Цей підхід має кілька переваг, серед яких можна виділити високу ступінь непомітності, оскільки модифікації шрифтів можуть бути практично невидимими для людського ока. Крім того, шрифти є невід'ємною частиною текстових документів, що робить їх ідеальним середовищем для стеганографічних методів.

Актуальність дослідження стеганографічних систем на основі комп'ютерних шрифтів (надалі – лінгвістичних стегосистем) зумовлена кількома факторами [1]. По-перше, в умовах сучасного інформаційного суспільства безпека передачі даних стає критично важливою. Зростання кіберзлочинності та різноманітні загрози конфіденційності стимулюють розвиток нових методів захисту інформації. Стеганографія, у цьому контексті, надає додатковий рівень безпеки, що може бути використаний разом з криптографічними методами для досягнення максимальної захищеності даних [2].

По-друге, стрімкий розвиток цифрових технологій і постійне зростання обсягів переданої інформації вимагають ефективних та економічних методів її захисту. Використання комп'ютерних шрифтів для приховування інформації є перспективним напрямком, оскільки дозволяє приховувати значний обсяг даних

у стандартних текстових документах без значного збільшення їх розміру чи зміни структури.

Нарешті, стеганографічні системи на основі комп'ютерних шрифтів мають широкий спектр застосувань – від забезпечення конфіденційності у приватній та корпоративній комунікації до використання у спеціальних службах для прихованого обміну даними. Вони також можуть використовуватися для захисту авторських прав і запобігання несанкціонованому копіюванню документів.

1.2 Аналіз аналогічних систем та публікацій

При виконанні кваліфікаційної роботи було проаналізовано схожі системи, які вирішують питання приховування інформації, використовуючи метод стеганографії.

Перша система з назвою «FontCode: Embedding Information in Text Documents using Glyph Perturbation» пропонує метод стеганографії, який використовує мінімальні зміни у формах гліфів для приховування інформації в текстових документах за допомогою згорткової нейронної мережі (англ. Convolutional Neural Network, CNN). Спочатку ініціалізується набір кандидатів-гліфів і створюється повний граф, де вузли представляють ці гліфи. У кожній ітерації випадково вибирається набір пар гліфів, для яких обчислюється точність навчання CNN. Якщо точність нижче 95 %, відповідні ребра видаляються з графа. Потім обчислюється максимальна кліка графа, і набір гліфів оновлюється. Якщо набір не змінюється, цикл завершується. Після цього CNN тренується на всіх елементах нового набору, і видаляються елементи з точністю тестування нижче 90 %. Основний недолік цього алгоритму – високі обчислювальні витрати [3].

У другій статті під назвою «Text Steganographic Approaches: A Comparison» представляється три нові підходи до текстової стеганографії. Перший підхід використовує ідею головоломки з пропущеними літерами, де кожен символ повідомлення приховується шляхом пропуску однієї або більше літер у слові-

контейнері. Середній показник схожості за метрикою Яро–Вінклера становив 0,95, що свідчить про високу схожість між контейнером і стегофайлом. Другий підхід приховує повідомлення у списку слів, де ASCII-значення вбудованого символу визначає довжину та початкову літеру слова. Третій підхід приховує повідомлення, не погіршуючи контейнер, використовуючи початкову та кінцеву літери слів контейнера. Для підвищення безпеки секретного повідомлення воно спочатку шифрується за допомогою одноразового блокнота, а потім зашифрований текст приховується у контейнері [4].

Третя система «New Text Steganography Technique by using Mixed-Case Font» використовує змішування верхнього і нижнього регістрів символів для приховування інформації. Метод полягає у зміні верхнього і нижнього регістрів окремих літер для кодування бітів інформації. Наприклад, зміна букви «а» на «А» може кодувати один біт інформації. Процес кодування включає наступні кроки: вибір текстового файлу, розбиття його на літери, конвертація секретного повідомлення у бітовий потік, та заміна регістру літер у вихідному тексті відповідно до бітів секретного повідомлення. Декодування включає аналіз тексту для виявлення змін у регістрах символів. Серед недоліків можна виділити легку помітність змін при аналізі тексту та обмежену кількість інформації, що може бути прихована [5].

У четвертій системі «Improvement of «Text Steganography Based on Unicode of Characters in Multilingual» by Custom Font with Special Properties» використовуються спеціальні властивості шрифтів для приховування даних у текстових документах. Метод базується на використанні Unicode символів з різними кодами, але схожими гліфами. Спеціальний шрифт створюється таким чином, що деякі англійські символи замінюються символами з іншими кодами, але тими ж гліфами, що дозволяє приховувати двійкові дані в тексті.

Серед недоліків цього підходу є потреба в спеціально створеному шрифті для кодування та декодування та складність у реалізації через необхідність точного підбору гліфів і кодів символів [6].

1.3 Вимоги до програмного забезпечення, постановка задачі

Стеганографічна система, що використовує комп'ютерні шрифти для приховування тексту, повинна відповідати ряду важливих вимог для забезпечення її ефективного функціонування. До основних вимог відносяться: забезпечення високого рівня непомітності прихованої інформації, надійність роботи програми, зручність у використанні для кінцевого користувача, а також оптимізація роботи з обчислювальними ресурсами.

Основна задача розробки полягає у створенні програмного забезпечення, здатного приховувати та відновлювати текстові повідомлення у документах за допомогою маніпуляції шрифтами. Це завдання включає дві основні частини:

1) модуль приховування тексту: програма повинна інтегрувати стеганографічний алгоритм у процес редагування текстового документа, змінюючи шрифт символі, відповідно даним, які треба приховати. Зміни повинні бути непомітними для користувача і не впливати на читабельність документа;

2) модуль відновлення прихованого тексту: програма повинна бути здатною швидко та точно відновлювати приховані повідомлення, використовуючи модифіковані параметри шрифтів.

Програму для приховування тексту доцільно реалізувати на мові програмування Python, яка відома своєю гнучкістю та багатим набором бібліотек для обробки текстових даних. Середовище розробки PyCharm може бути використане для написання, відлагодження та тестування коду, оскільки надає зручний інтерфейс і потужні інструменти для розробників.

Для реалізації алгоритмів стеганографії найчастіше використовується бібліотека *python-docx*, яка забезпечує роботу з документами формату DOCX [7].

Ця бібліотека дозволяє програмно створювати та модифікувати документи Word, зокрема змінювати параметри шрифтів та інші атрибути тексту, що є ключовим аспектом для реалізації стеганографічних методів у текстових файлах.

Висновки до розділу 1

У цьому розділі було проведено аналіз вимог до лінгвістичних стегосистем та інструментів для їх розробки, а також розглянуто основні аспекти предметної області та обґрунтовано актуальність теми. Стеганографія, як наука приховування інформації, є критично важливою в умовах сучасного інформаційного суспільства. Вона надає додатковий рівень безпеки, який може бути використаний разом із криптографічними методами для забезпечення максимальної захищеності даних. Серед різних методів стеганографії особливу увагу було приділено використанню комп'ютерних шрифтів для приховання даних, що має кілька переваг, включаючи непомітність та інтеграцію у стандартні текстові документи без значного збільшення їх розміру чи зміни структури.

Було проведено аналіз існуючих стегосистем для приховування тексту, що дозволило визначити їх сильні та слабкі сторони, а також оцінити їх ефективність та надійність. Це дало можливість сформулювати основні вимоги до програмного забезпечення для стеганографії, яке використовує комп'ютерні шрифти. Важливими аспектами є забезпечення високого рівня непомітності прихованої інформації, надійність роботи програми, зручність у використанні для кінцевого користувача, а також оптимізація роботи з обчислювальними ресурсами.

Постановка задачі включає розробку програмного забезпечення, здатного приховувати та відновлювати текстові повідомлення у документах за допомогою маніпуляції шрифтами.

Python був обраний через свою зручність, простоту синтаксису та широкі можливості для обробки тексту. PyCharm, як інтегроване середовище розробки (англ. Integrated Development Environment, IDE), надає інструменти для

написання, тестування та відлагодження коду, що значно спрощує процес розробки [8].

Таким чином, розроблюване програмне забезпечення відповідатиме вимогам до стеганографічних систем, забезпечуючи ефективне приховування та відновлення тексту за допомогою зміни шрифтів у документах.

2 МІСЦЕ ТА АСПЕКТИ КОМП'ЮТЕРНОЇ СТЕГАНОГРАФІЇ У ЗАГАЛЬНІЙ СИСТЕМІ ЗАХИСТУ ІНФОРМАЦІЇ ПРИХОВУВАННЯМ

2.1 Галузі застосування стеганографії

Історія стеганографії нараховує тисячоріччя [9]. Приховування факту існування таємного повідомлення завжди було доцільним для його захисту, а існування різних технічних, хімічних, фізичних та психологічних методів такого приховування забезпечувало можливість його реалізації (рис. 2.1).

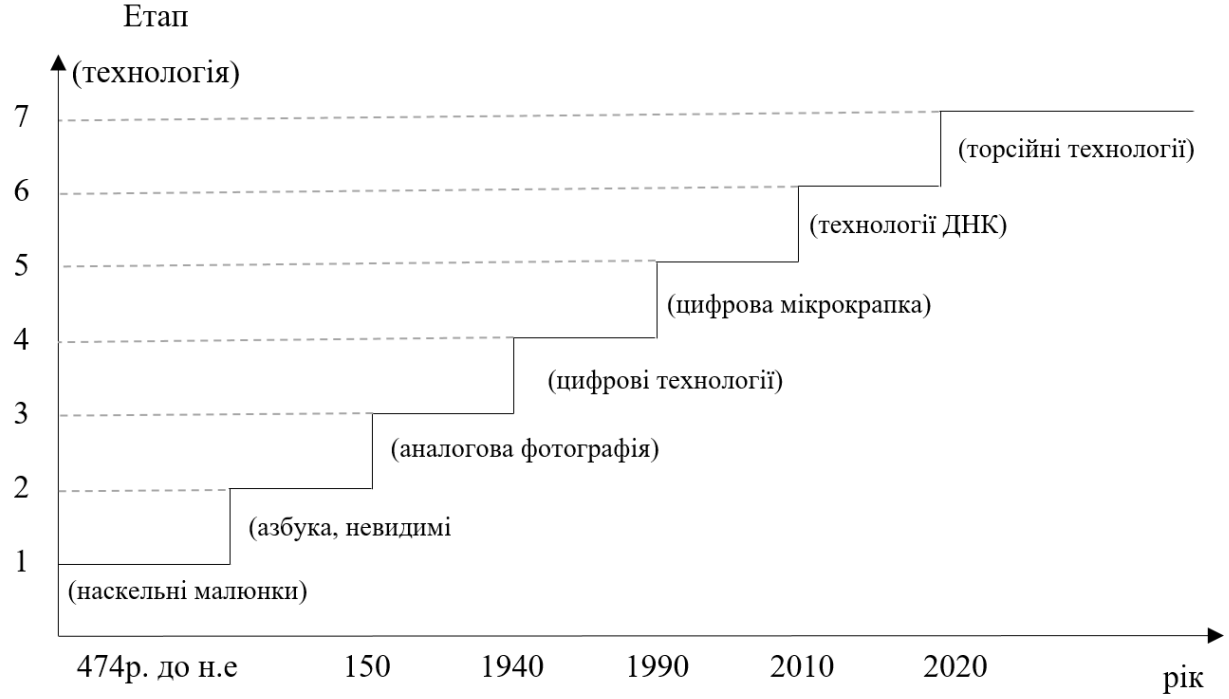


Рисунок 2.1 – Становлення стеганографії

Місцем зародження стеганографії вважається Єгипет, хоча «стеганографічними повідомленнями» можна назвати і наскельні малюнки прадавніх людей.

В літературі вперше згадано про стеганографію ще у давній Греції, а схема була доволі простою: текст наносили на дерев'яні дощечки, покриті воском. З розповідей Геродота відомо, що для передавання повідомлення про захоплення Греції царем персів використовувалася звичайна дощечка із секретним

повідомленням під шаром воску. Варті дощечка з посланням була пред'явлена як звичайна заготовка для листа, тому і не викликала підозри.

Іншим вдалим методом була передача послання на голові раба, якого голили наголо, а саме повідомлення наносили у вигляді татуювання. Після того як волосся відростало, виявити повідомлення не знаючи про нього було доволі складно, а щоб прочитати, досить було знову поголитися.

У Китаї листи писали на смужках шовку. Тому для приховування повідомлень смужки з текстом листа згорталися в кульки, покривалися воском і потім ковтались посильними.

Темне середньовіччя породило не тільки інквізицію. Посилення стеження привело до розвитку як криптографічних, так і стеганографічних методів. Саме в середні віки вперше було застосовано спільне використання шифрів і стеганографічних методів.

До кінця XIX-го століття криптографія і стеганографія розвивалися паралельно в рамках однієї науки – тайнопису. Криптографія відокремилася від стеганографії після формулювання правила Кірхгофа про принципи побудови криптографічних систем. Воно полягає в тому, що секретним є не сам алгоритм, а лише набір конкретних параметрів без зниження стійкості алгоритму нижче допустимої величини. Криптографія перетворилася на повноцінну науку із сукупності особливих методів. Вона ґрунтується на теорії ймовірності, математичній статистиці, теорії числових полів.

Стеганографія, як техніка приховування інформації, відзначається своєю універсальністю та широким спектром застосувань. Від таємних повідомлень та прихованої інформації до захисту від несанкціонованого доступу, вона знаходить використання в різних сферах життя, де конфіденційність та безпека є пріоритетом (рис. 2.2).

Рисунок 2.2 – Множини застосування стеганографії

Нині стеганографію використовують для вирішення таких основних завдань:

- захист авторських прав;
- відстеження каналів витоку інформації;
- захист конфіденційної інформації від несанкціонованого доступу;
- створення секретних каналів передачі інформації;
- подолання систем моніторингу;
- камуфляж програмного забезпечення (ПЗ).

2.2 Напрями стеганографії

У світі стеганографії виокремлюють декілька основних напрямків: класичний, комп’ютерний, цифровий. Окрім того, існують лінгвістична та квантова стеганографія. Кожен з цих напрямків розвивається від класичної стеганографії і має свої коріння ще з давніх часів.

2.2.1 Класична стеганографія

Це давня та традиційна форма стеганографії, яка використовувалася задовго до появи комп'ютерів. Вона включає в себе всі методи, не пов'язані з комп'ютерними системами. Цей напрям включає в себе всі методи, які не пов'язані з комп'ютерними системами, але використовуються для приховування інформації в різних формах комунікації, таких як письмові повідомлення, книги, малюнки тощо (рис. 2.3).

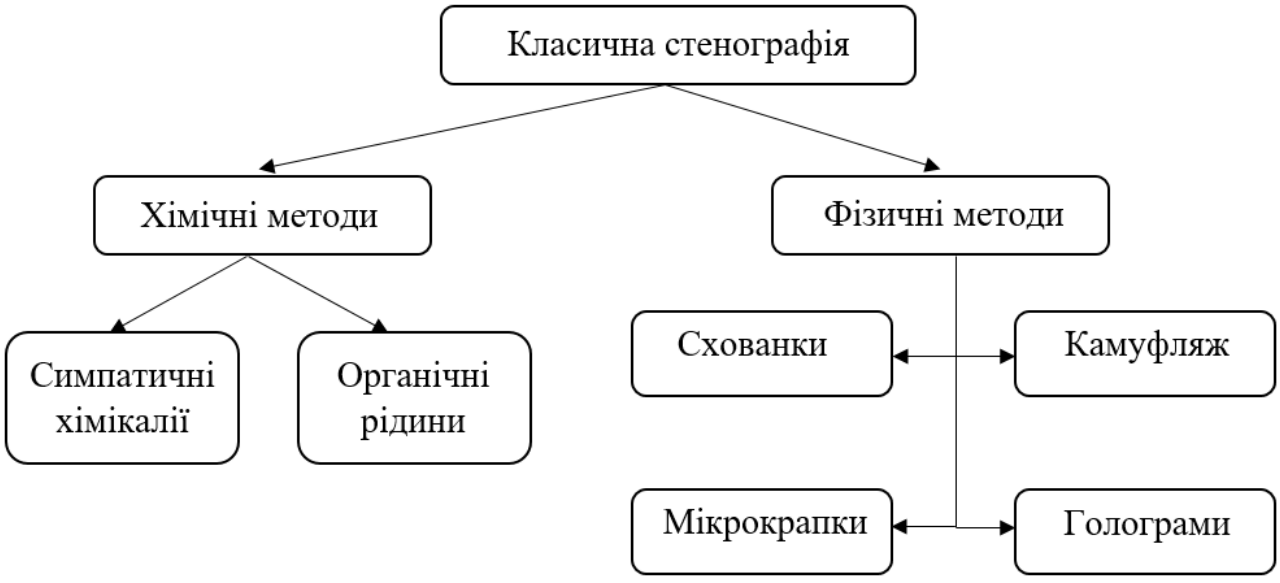


Рисунок 2.3 – Методи класичної (традиційної) стеганографії

Сьогоднішня класична стеганографія насичена різноманітними прийомами, як хімічними, так і фізичними. Хімічні методи цієї техніки зазвичай базуються на

використанні невидимих чорнил, які наносяться на носії інформації. Вони можуть включати симпатичні хімікалії або органічні розчинники. Симпатичні хімікалії є одним із найпоширеніших методів, де важлива інформація записується невидимим чорнилом, що активується під певними умовами, тоді як на поверхню накладається банальний напис. Органічні розчинники мають подібні властивості, темнішаючи під впливом тепла через вміст вуглецю.

Фізичні методи стеганографії охоплюють схованки, камуфляж та мікрокрапки. У сучасний період особливий інтерес викликають дослідження різних носіїв інформації для запису даних, які не виявляються стандартними методами. Зокрема, нові технології, основані на класичній стеганографії, використовують останні досягнення мікроелектроніки, наприклад, голограми.

Одним із найпоширеніших методів класичної стеганографії є використання прихованих повідомлень у текстах. Наприклад, застосування спеціальних шифрів, вбудованих у текстові документи, дозволяє відправнику і отримувачу спілкуватися без можливості розшифрування повідомлення третім особам.

Крім того, класична стеганографія використовує різні фізичні об'єкти для приховування інформації, такі як зображення, малюнки, музика тощо. Це може включати в себе використання спеціальних методів запису інформації на носіях, таких як папір, стіни, обкладинки книг тощо.

У класичній стеганографії важливою є якість приховування інформації та стійкість до виявлення. Це означає, що методи, які використовуються для приховування інформації, повинні бути надійними та ефективними, а також не привертати зайву увагу до самого факту прихованої комунікації.

Загалом, класична стеганографія залишається важливим інструментом для збереження конфіденційності та безпеки інформації. Вона використовується у різних сферах, від військової комунікації до культурного обміну, і продовжує відігравати важливу роль у сучасному світі. На сьогоднішній день аналізуючи методи класичної стеганографії, можна зробити певні висновки. Серед переваг -

доступність та широкі можливості застосування. Але слід враховувати і недоліки, такі як складність реалізації та ризик виявлення таємної інформації.

2.2.2 Комп'ютерна стеганографія

Це сучасний напрям стеганографії, що базується на особливостях комп'ютерних систем та використанні спеціальних властивостей форматів даних. Цей підхід включає в себе розробку алгоритмів та ПЗ для приховування інформації в електронних об'єктах.

Одним із ключових аспектів комп'ютерної стеганографії є можливість вбудовування інформації в різноманітні типи файлів, такі як тексти, зображення, відео, аудіо тощо [10]. Цей процес може бути здійснений шляхом модифікації деяких бітів у межах файлу або використання спеціальних методів кодування для приховування інформації.

Під час розробки комп'ютерних методів стеганографії велика увага приділяється не лише ефективності вбудовування інформації, але й стійкості до виявлення. Існують різноманітні техніки, спрямовані на ускладнення процесу виявлення схованої інформації, такі як використання криптографічних методів, стеганографічних ключів та інших захистів.

Крім того, у комп'ютерній стеганографії велика увага приділяється розробці спеціального ПЗ, яке дозволяє здійснювати процес приховування та виявлення схованої інформації. Ці програми можуть надавати різні функції, від базових інструментів для приховування текстової інформації до складних алгоритмів для вбудовування даних у мультимедійні файли.

Комп'ютерна стеганографія залишається важливим інструментом для збереження конфіденційності та безпеки даних у цифровому середовищі. Вона забезпечує можливість таємного обміну інформацією, що є ключовим аспектом в сучасному світі технологій та комунікацій.

2.2.3 Цифрова стеганографія

Це напрям класичної стеганографії, заснований на приховуванні або впровадженні додаткової інформації в цифрові об'єкти. Цифрова стеганографія, безсумнівно, є одним з найбільш цікавих та актуальних напрямків у світі секретного збереження інформації. Цей підхід заснований на використанні особливостей цифрових об'єктів, таких як зображення, відео, аудіо та інші формати файлів, для приховування або вбудовування додаткової інформації, непомітної при звичайному спостереженні (рис. 2.4).

Рисунок 2.4 – Методи цифрової стеганографії

Одним з ключових аспектів цифрової стеганографії є методика вбудовування інформації в контейнерні файли. Це може включати в себе вбудовування текстових повідомлень, файлів, чи навіть інших медіа-об'єктів у вже існуючі цифрові об'єкти. Зазвичай, цей процес здійснюється шляхом зміни певних бітів у межах контейнера так, щоб надана інформація не була помітна при перегляді або прослуховуванні.

Технології цифрової стеганографії постійно розвиваються, із з'явленням нових методів та алгоритмів, спрямованих на поліпшення ефективності та стійкості приховування інформації. Наприклад, алгоритми стеганографії можуть використовувати різні методи кодування та захисту, щоб зробити вбудовану інформацію більш стійкою до виявлення [11].

Однак, варто відзначити, що цифрова стеганографія також має свої обмеження та виклики. Одним із найбільш важливих аспектів є збереження якості контейнерних файлів. Під час вбудовування інформації може виникати спотворення, яке може вплинути на якість вихідного об'єкту. Додатково, існують методи атак, спрямованих на виявлення та вилучення схованої інформації, що робить важливим пошук балансу між стійкістю та прихованістю.

У цілому, цифрова стеганографія залишається важливим інструментом для збереження конфіденційності та безпеки інформації в цифровому світі. Вона забезпечує можливість передачі секретної інформації, не привертаючи уваги до самого факту комунікації, та продовжує розвиватися разом із зростанням вимог до кібербезпеки та конфіденційності.

Аналіз різновидів стеганографії дозволяє отримати уявлення про різноманітність і складність цього методу приховування інформації. Класична, комп'ютерна та цифрова стеганографія кожна має свої особливості та застосування, але вони всі спрямовані на одну мету – збереження конфіденційності та безпеки даних. Використовуючи різні підходи та методи, стеганографія стає важливим інструментом у світі секретного збереження

інформації, забезпечуючи можливість таємного обміну даними у цифрову епоху. Щодо класичної, комп'ютерної та цифрової стеганографії, вони відіграють важливу роль у забезпеченні безпеки, конфіденційності та ефективності комунікацій, і продовжують розвиватися разом з технологічними змінами та викликами у сучасному світі.

2.2.4 Лінгвістична стеганографія

Лінгвістична стеганографія – це не лише поле досліджень для криптографів та лінгвістів, але й потужний інструмент для забезпечення безпеки і конфіденційності інформації в сучасному цифровому світі. Вона використовує мовні ресурси та техніки для того, щоб приховувати повідомлення всередині іншого тексту так, щоб це було максимально непомітно для непроінформованих осіб.

У напрямі лінгвістичної стеганографії існують різні підходи до таємного кодування інформації, які можна розділити на дві основні категорії: умовне письмо і семаграми, як наведено на рис. 2.5 [12].

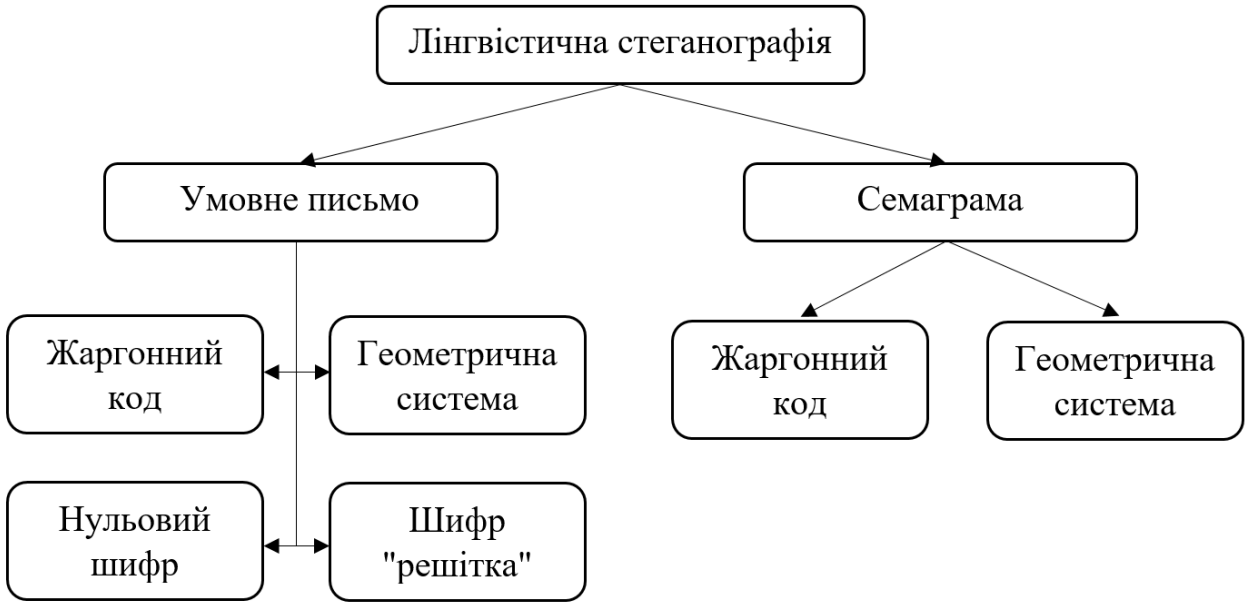


Рисунок 2.5 – Методи лінгвістичної стеганографії

Умовне письмо включає такі методи, як жаргонний код, геометрична система, нульовий шифр і шифр «решітка» [13]. Жаргонний код оперує термінами, що мають альтернативне значення, щоб зберегти непримітність тексту перед непроінформованими особами.

Геометрична система використовує розташування слів у певних точках або на перетині геометричних фігур для передачі інформації.

Нульовий шифр застосовує певні правила для приховування повідомлення, наприклад, вказівки читати кожне п'яте слово.

Шифр «решітка» використовується для приховування повідомлення за допомогою шаблону, де слова виступають як ключові елементи.

Семаграми представляють іншу категорію, в якій символи (крім літер і цифр) використовуються як значення для шифрування. Ці повідомлення можуть бути вбудовані в різноманітні об'єкти або образи, які на перший погляд здаються непідозрілими. Також існують текстові семаграми, які змінюють зовнішній вигляд тексту для приховування інформації.

Одним з найцікавіших аспектів лінгвістичної стеганографії є її історичний контекст. Методи приховування інформації за допомогою мови були використані ще в давні часи. Сучасні технології дозволяють вдосконалити ці методи до неймовірного рівня. Від використання спеціальних алгоритмів для вбудовування таємних повідомлень у цифрові файли до розвитку штучного інтелекту, який може автоматично генерувати стеганографічні засоби, лінгвістична стеганографія залишається одним з найбільш динамічних напрямків у сфері кібербезпеки.

Більш того, лінгвістична стеганографія знаходить застосування не лише в області кібербезпеки, але й у літературі, мистецтві та навіть політиці. Від закодованих повідомлень у літературних творах до використання мови та символіки для ведення таємних переговорів між політичними діячами – лінгвістична стеганографія може бути дієвим інструментом в різних сферах життя [14].

Методи лінгвістичної стеганографії мають перевагу в передачі великих обсягів інформації. Однак, серед їх недоліків можна відзначити ризик випадкового розкриття кодування та складність процесу шифрування повідомлення.

Тим не менш, разом з розвитком цих технологій постають і нові виклики. Сучасні аналітичні методи та програми для виявлення стеганографічних повідомлень стають все більш ефективними, що вимагає постійного вдосконалення та розвитку методів лінгвістичної стеганографії для забезпечення надійності та конфіденційності інформації.

2.2.5 Квантова стеганографія

У сучасному дослідженні інформаційних технологій активно обговорюється напрям розвитку, який використовує квантові принципи для створення систем забезпечення інформаційної безпеки (рис. 2.6). Хоча квантова стеганографія ще не отримала широкого розповсюдження, проте вже існують деякі моделі, які поєднують у собі класичні та квантові аспекти інформаційних технологій. Вони базуються на поєднанні квантової фізики з класичною теорією інформації. Однією з таких моделей є концепція використання квантових коригуючих кодів для приховування таємних повідомлень шляхом створення помилок в квантовому каналі.

Сучасною є ідея приховування інформації у формі синдрому помилки за допомогою квантових коригуючих кодів [15]. Проте, розроблений протокол не гарантує непомітного приховування конфіденційної інформації у квантовому середовищі. У подальших дослідженнях пропонується використовувати квантові характеристики для приховування інформації шляхом інтеграції квантового розподілу ключів для підвищення безпеки зв’язку, медичних записів, фінансових операцій, військової оборони, захисту інтелектуальної власності тощо [16].

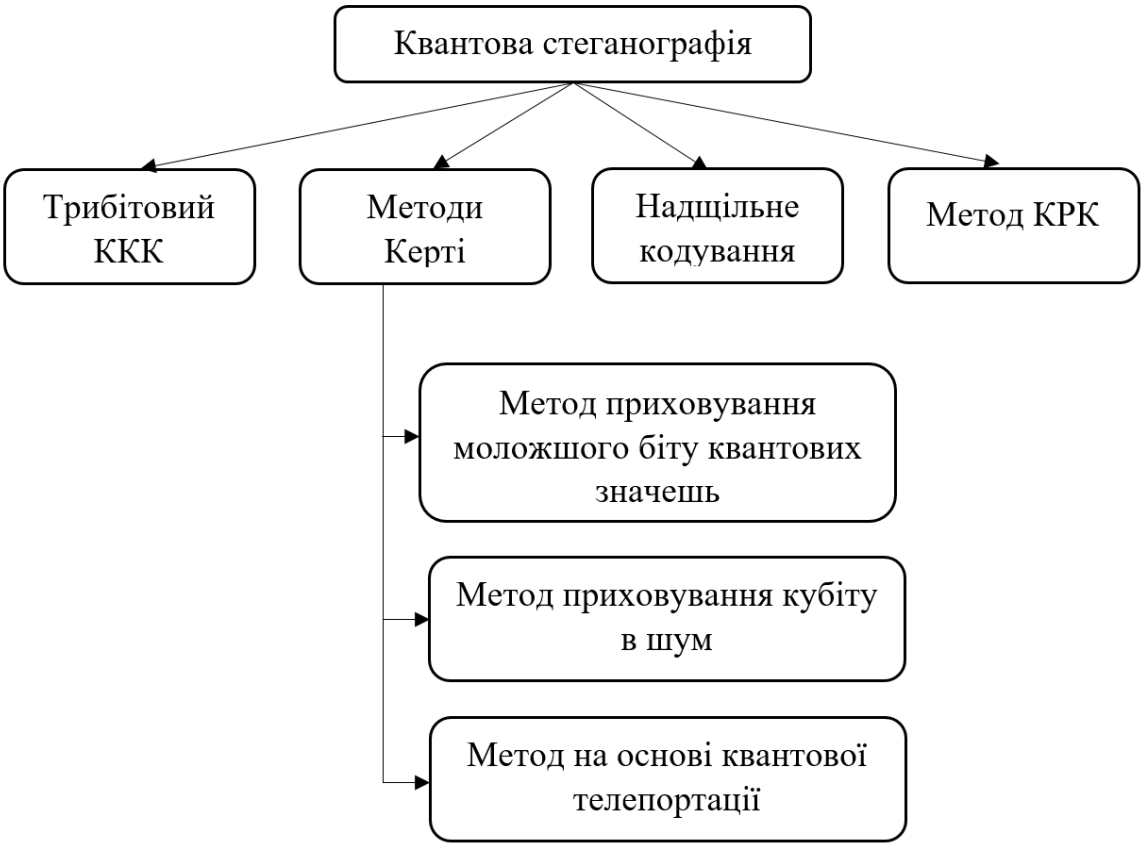


Рисунок 2.6 – Методи квантової стеганографії

Перша з цих систем передбачає приховування одного класичного біта у шумовому кубіті шляхом заміни кубіта. Друга система приховує два класичних біти у кубіті зі щільним кодуванням. Безпека цих систем залежить від ідентичності квантового шуму та справжнього шуму. У третій системі кубіт передається через класичний стеганографічний канал за допомогою квантової телепортації.

Незважаючи на те, що ці протоколи вирішують певні аспекти стеганографії, вони не вирішують питання непомітності при передачі повідомлень через відкриті або загальні канали з урахуванням секретності. Дослідники також розглядають інші аспекти квантової стеганографії, такі як модифікація надщільного кодування та розробка протоколів квантового розподілу ключів. Ці напрямки дозволяють зробити квантову стеганографію більш стійкою порівняно з традиційними

методами, оскільки перехоплення і декодування конфіденційної інформації, закодованої у квантових станах, теоретично ускладнюється.

Незважаючи на переваги, квантова стеганографія також має свої обмеження. Наприклад, вона може бути обмежена швидкістю передачі через обмежену доступність квантових каналів зв'язку та складність процесів кодування та декодування квантових повідомлень.

У будь-якому випадку, квантова стеганографія представляє собою цікаву та перспективну галузь, яка може знайти застосування в сучасних системах забезпечення інформаційної безпеки.

2.3 Класифікація стеганографічних методів

Аналізуючи поточні методи таємної передачі інформації, можна висунути новий підхід до класифікації комп'ютерної стеганографії. Усі існуючі методи можна розподілити на декілька категорій згідно з такими критеріями: вибір контейнера, наявність ключа, призначення, метод приховування та рівень стійкості.

Перший критерій – вибір контейнера – можна поділити на п'ять основних напрямків:

1) **за форматом:**
   - спеціальне форматування текстових файлів;
   - використання зарезервованих для розширення полів у комп'ютерних форматах даних;
   - експлуатація надлишковості аудіо- та візуальної інформації, а також комбінованого мультимедіаконтенту;
2) **за способом вилучення інформації:**
   - з оригіналом;
   - без оригіналу;
   - використання фрагменту оригіналу;

3) за розміром:

– потокові контейнери

– фіксовані контейнери;

4) за способом вибору контейнера:

– сурогатні методи, де вибір контейнера відсутній;

– селективні методи, що враховують статистичні характеристики шуму;

– конструктивні методи, що генерують контейнер відповідно до потреб стегосистеми;

5) за організацією контейнера, подібно до завадостійких кодів:

– систематичні, де місця розташування інформаційних та шумових бітів вказані;

– несистематичні, де ці місця не визначені наперед.

Наявність ключа розділяє стегосистеми на ключові, безключові та змішані. Використання безключових систем не потребує додаткових даних окрім алгоритму стеганографічного перетворення. Ключові системи можуть бути з секретним або відкритим ключем, що впливає на потребу в захищеному каналі обміну стегоключами.

За призначенням, методи стеганографії можуть бути застосовані у таких галузях:

– захист від копіювання;

– прихована анотація документів;

– аутентифікація;

– прихований зв'язок.

Використання стеганографічних систем є ефективним способом захисту обмеженого доступу до інформації та має потенціал у сферах аутентифікації та маркування авторської продукції для захисту авторських прав. Також, методи стеганографії можуть використовуватись для камуфляжу ПЗ та створення

прихованих каналів передачі інформації.

Щодо принципу приховування, методи стеганографії можна розділити на дві основні категорії: безпосередня заміна та спектральні методи. Перші використовують надлишок інформаційного середовища для заміни бітів контейнера секретними даними, в той час як другі базуються на спектральному представленні середовища.

З погляду стійкості, стегосистеми можуть бути класифіковані як стійкі, вразливі та напіввразливі, залежно від їхньої здатності зберігати приховану інформацію в умовах різного роду атак, як цифрових, так і фізичних.

2.4 Криптографія, стеганографія, обфускація

Стеганографія та криптографія є двома важливими галузями кібербезпеки, які спрямовані на захист конфіденційності та цілісності інформації. Вони використовують різні підходи до досягнення цієї мети, і кожен з них має свої унікальні переваги та обмеження (табл. 2.1).

Криптографія, з одного боку, оперує математичними методами для забезпечення безпеки інформації. Вона використовує алгоритми шифрування для перетворення звичайного тексту у нечитабельний шифрований текст, який може бути розшифрований лише за допомогою спеціального ключа. Цей підхід є досить прозорим: якщо атакувач перехопить зашифроване повідомлення, він може визначити, що дані були зашифровані, і спробувати зламати шифр. Однак надійні криптографічні алгоритми здатні стійко захищати інформацію від несанкціонованого доступу.

Стеганографія, з іншого боку, приховує факт наявності секретного повідомлення. Вона не перетворює саме повідомлення, але розташовує його в незрозумілому для сторонніх місці, такому як у різниці у пікселях зображення, пробілах в тексті, або в аудіо-сигналах. Таким чином, навіть якщо атакувач отримає доступ до захованого повідомлення, він може навіть не підозрювати, що

ця інформація існує. Однак стеганографія часто вимагає більш складних та тонких методів для вбудовування інформації і може бути менш ефективною для захисту великих обсягів даних.

Таблиця 2.1 – Порівняння стеганографії та криптографії

Фактори	**Стеганографія**	**Криптографія**
Пояснення	Це метод приховування факту комунікації, що відбувається	Це спосіб зробити інформацію нерозбірливою
Мета	Підтримувати безпеку комунікації	Забезпечити захист даних
Ключ	Необов'язково, але підвищує безпеку при використанні	Необхідна умова
Видимість даних	Ні	Так
Недоліки	Після декодування прихованої інформації дані можуть бути використані будь-ким	Можна відновити оригінальне повідомлення із зашифрованого тексту, якщо є доступ до ключа розшифрування
Структура даних	Не змінює загальну структуру даних	Змінює загальну структуру даних

Варто зазначити, що в багатьох випадках криптографія та стеганографія можуть бути використані разом для створення більш потужних систем захисту інформації. Наприклад, дані можуть бути спочатку зашифровані з використанням криптографії, а потім цей зашифрований текст може бути вбудований в приховані канали за допомогою стеганографії. Це дозволяє поєднати переваги обох методів

і забезпечити високий рівень захисту для конфіденційної інформації.

Обфускація та стеганографія представляють схожі за загальним принципом методи захисту інформації, але вони відрізняються за способами виконання та цілями, які вони переслідують.

Обфускація в основному зосереджується на ускладненні зрозуміння або аналізу даних, зазвичай шляхом зміни їх структури чи кодування таким чином, щоб вони залишалися функціональними, але були менш зрозумілими для сторонніх користувачів. Це може включати зміну імен змінних, розміщення логіки коду в неочевидних місцях, або використання технік, які роблять код важким для розуміння без додаткового контексту. В контексті ПЗ обфускація широко використовується для утруднення зворотного інжинірингу та уникнення копіювання.

Фахівці з кібербезпеки використовують обфускацію для захисту конфіденційної інформації, наприклад, програмних кодів. Цей процес ускладнює читання кодів хакерами, що, в свою чергу, заважає їм використовувати дані.

Отже, хоча і обфускація, і стеганографія займаються заплутуванням даних, їх підходи та цілі відрізняються. Обфускація спрямована на ускладнення аналізу, тоді як стеганографія ставить своєю метою приховування наявності інформації.

Стеганографія, криптографія та обфускація є важливими інструментами в галузі кібербезпеки, кожен з яких має свої унікальні переваги та застосування. Успішне застосування цих методів залежить від конкретних вимог та сценаріїв застосування, а також від вміння правильно вибирати і комбінувати їх для досягнення максимального рівня захисту даних. Комплексне використання стеганографії, криптографії та обфускації може створити додаткові бар'єри для несанкціонованого доступу та забезпечити надійний захист конфіденційної інформації.

2.5 Стеганографія і NFT

Є деяка схожість між стеганографією і невзаємозамінними токенами (англ. Non-Fungible Token, NFT). Хоча NFT самі по собі не є стеганографічними інструментами, вони можуть містити стеганографічно приховану інформацію. При створенні NFT в нього зазвичай можна додати додатковий контент, який буде видно тільки власнику цього NFT. Це може бути що завгодно: мультимедійний контент високої чіткості (HD-контент), повідомлення, відео, доступ до секретних спільнот, промокоди і навіть смарт-контракти або активи в онлайн-іграх.

Наприклад, художник може створити цифрове зображення як NFT і використовувати стеганографію для приховування спеціального повідомлення або іншого контенту всередині зображення. Тільки власник NFT може знати про це приховане повідомлення і мати засоби для його розшифрування.

2.5.1 Потенціал NFT зі стеганографією

Використання стеганографії у NFT може відкрити нові можливості для цифрового мистецтва та колекціонування. Художники можуть вбудовувати приватні повідомлення або бонуси, які стають доступними тільки після покупки NFT. Це може включати:

- вища якість або розширені версії творів мистецтва;
- секретні аудіофайли або відео, які можна відтворити тільки після покупки;
- смарт-контракти, що активуються за певних умов, наприклад, надання доступу до закритих заходів;
- промокоди або купони для майбутніх покупок чи знижок;
- віртуальні активи, які можуть бути використані в онлайн-іграх або інших цифрових середовищах.

2.5.2 Безпека та приватність

Використання стеганографії в NFT також може підвищити рівень безпеки та приватності. Оскільки інформація прихована, вона менш схильна до несанкціонованого доступу або копіювання. Це створює додатковий шар захисту для власників NFT та може служити засобом для забезпечення автентичності та виключності цифрового контенту.

2.5.3 Майбутнє NFT та стеганографії

Оскільки технології розвиваються, можливості застосування стеганографії в NFT можуть розширюватися. Це може включати інтеграцію з доповненою реальністю (англ. Augmented Reality, AR), віртуальною реальністю (англ. Virtual Reality, VR), іграми на блокчейні та іншими цифровими платформами. Завдяки цьому власники NFT можуть отримувати ще більш персоналізований і унікальний досвід.

У міру розвитку світової арт-спільноти змінюються і технології NFT. У майбутньому створення NFT, що містять приватні метадані, буде дедалі популярнішим, а сфера їхнього застосування - дедалі ширшою, включно з геймінгом, платними підписками, розповсюдженням квитків на заходи тощо.

Ще одною перевагою використання стеганографії в NFT є те, що це може збільшити цінність та унікальність цифрових активів. Колекціонери та інвестори можуть бути зацікавлені в придбанні NFT не тільки через їх візуальну привабливість або історичну цінність, але і через можливість отримати додаткові приховані або ексклюзивні вміст або можливості, які можуть бути розкриті лише власникові.

Однак важливо враховувати, що збереження безпеки та приватності є ключовими аспектами у використанні стеганографії в NFT. Надійні методи шифрування та захисту даних повинні бути використані для забезпечення того, що прихована інформація залишається доступною лише авторизованим особам.

Додатковою можливістю розвитку є інтеграція стеганографії в блокчейн-платформи, що дозволить створювати стійкі до злому та маніпуляцій механізми для забезпечення цілісності та конфіденційності цифрових активів. Такі інновації можуть революціонізувати спосіб, яким ми сприймаємо та обмінюємося цифровим мистецтвом та іншими цифровими активами.

2.6 Виявлення стеганографії

У своїй роботі аналітики безпеки шукають індикатори стандартних стратегій тестування атак і проникнень (англ. Penetration Test, PenTest). З часом були виявлені типові підписи, що використовуються стеганографічним ПЗ. Завдяки цьому, наприклад, антивірусне ПЗ може легко виявити типові моделі поведінки стеганографічних програм.

Як наслідок, тестувальники проникнення та зловмисники постійно коригують свої методи, щоб залишатися непоміченими. Так само дослідники безпеки постійно шукають нові сигнатури і тактики атак, а кіберзлочинці постійно адаптують свої інструменти і підходи.

2.7 Реальні атаки з використанням стеганографії

У 2020 році компанії у Великій Британії, Німеччині, Італії та Японії постраждали від кампанії з використанням стеганографічних документів.

Хакери могли уникнути виявлення, використовуючи стеганографічне зображення, завантажене на хорошу платформу, наприклад, Imgur, для зараження документа Excel. Mimikatz, шкідливе ПЗ, яке викрадає паролі Windows, було завантажено за допомогою секретного скрипту, що містився в зображенні.

2.8 Пом'якшення наслідків атак на основі стеганографії

Стеганографію легко застосувати під час кібератаки. Однак набагато важче запобігти їй, оскільки люди, які становлять загрозу, стають все більш винахідливими та винахідливими, що ускладнює розробку контрзаходів.

Код, замаскований у зображеннях та інших видах маскування, з більшою ймовірністю буде виявлений динамічно за допомогою поведінкового механізму. Тому компаніям слід використовувати сучасні рішення для захисту кінцевих точок, які виходять за рамки статичних перевірок, елементарних підписів та інших застарілих компонентів.

Співробітники повинні усвідомлювати ризик відкриття графічних файлів, оскільки вони можуть містити віруси. Крім того, слід встановлювати найновіші патчі безпеки, коли вони стають доступними, а також використовувати вебфільтри, щоб забезпечити безпечний перегляд вебсторінок співробітниками.

Висновки до розділу 2

У розділі підсумовуються ключові аспекти стеганографії, її історичні корені, сучасні методи та галузі застосування. Від прадавніх методів приховування повідомлень на дерев'яних дощечках і в татуюваннях до сучасних комп'ютерних і цифрових технологій, стеганографія демонструє постійний розвиток та адаптацію до нових викликів безпеки та конфіденційності. Класична стеганографія використовує фізичні та хімічні методи, тоді як комп'ютерна і цифрова стеганографія інтегрують приховування інформації в електронні та мультимедійні файли. Лінгвістична стеганографія, що використовує мовні ресурси, та квантова стеганографія, яка спирається на квантові принципи, відзначаються своєю складністю і потенціалом для майбутніх розробок.

Стеганографія залишається важливим інструментом для захисту інформації, що використовують в різних сферах: від авторських прав до безпеки цифрових комунікацій. Впровадження нових методів і алгоритмів для приховування інформації та розвиток програмного забезпечення для її виявлення і захисту свідчить про значний прогрес і актуальність цієї науки в сучасному світі технологій. Кожен напрямок, будь то класична, комп'ютерна чи цифрова стеганографія, має свої переваги та обмеження, що робить їх застосування залежним від конкретних вимог безпеки та конфіденційності. Стеганографічні методи продовжують розвиватися разом з технологічними змінами, забезпечуючи надійний захист інформації в умовах зростаючих загроз і викликів.

3 СТВОРЕННЯ СТЕГАНОГРАФІЧНОЇ СИСТЕМИ

У цьому розділі буде детально пояснено запропонований метод з процедурами кодування, вбудовування та вилучення. Також буде приділено увагу розгляду схожих шрифтів для кодування та таблиці кодів для використаного алфавіту.

3.1 Огляд шрифтів

Microsoft Word – популярний текстовий редактор, який входить до складу пакету Microsoft Office. Однією з причин його популярності є величезна кількість можливостей форматування тексту. Однією з таких можливостей є форматування шрифту, перевагами якого є велика ємність, непомітність і широкий діапазон застосування.

Шрифт – це графічний дизайн, який застосовується до набору цифр, символів і знаків. Шрифт описує певну типографіку разом з іншими характеристиками, такими як розмір, інтервал і нахил. Шрифти використовуються для відображення тексту на екрані та для друку тексту. Шрифти мають такі стилі накреслення, як курсив, напівжирний і напівжирний курсив.

MS-Word надає ряд стандартних шрифтів, включаючи Times New Roman, Courier New, Arial та багато інших, які зроблять вміст документа більш виразним. Деякі шрифти можуть мати однаковий розмір, але виглядати більшими завдяки висоті *x*. Висота *x* – це буквально висота маленької літери *x* у сімействі шрифтів. Різні шрифти мають різну висоту *x*, і в результаті деякі шрифти виглядають більшими за інші, навіть якщо вони мають однаковий кегль [17]. На рис. 3.1 показано, як вимірюється розмір шрифту (кегль) і висота *x*.

Деякі новіші сімейства шрифтів, такі як Tahoma і Verdana, розроблені з великою висотою *x*. Це означає, що різні сімейства шрифтів, які мають однаковий розмір кегля, виглядають більшими через більшу висоту *x*.

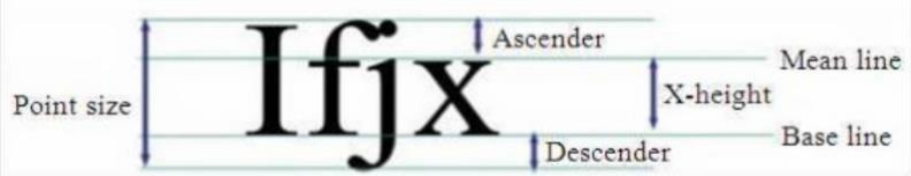


Рисунок 3.1 – Складові елементи шрифтів

3.2 Запропонований метод на основі оброблення комп’ютерних шрифтів

Це дослідження базується на текстових документах, які є найбільш поширеною і незамінною формою інформації в наш час і завжди використовуються в якості носія для приховання даних. Більшість текстових стеганографічних програм базуються на форматах TXT, MS Word, PDF, PPT.

Запропонований метод представляє новий прийом створення прихованих повідомлень в тексті формату документа, який використовує найбільш схожі типи англійських шрифтів для приховування повідомлення шляхом зміни шрифту на інший. Взагалі, будь-який тип шрифту має багато схожих варіантів і саме ця властивість є основою для стеганографічної системи.

Для створення стеганографічної системи потрібно реалізувати нижче описані компоненти та процеси.

1. Створити масив схожих шрифтів. Це один з найважливіших компонентів методу, від нього залежить непомітність прихованого повідомлення. Потрібно почату з визначення типу шрифту документа, а потім знайти найбільш схожі типи. У цьому дослідженні взято (8) типів шрифтів, які є які є найбільш вживаними та поширеними в текстових документах (TXT, MS Word, PDF, PPT). У табл. 3.1 наведено шрифти та їхні аналоги; для кожного типу буде використано по три аналоги.

Таблиця 3.1 – Шрифт документа та схожі на нього шрифти

№ з/п	Шрифт документа	Схожі шрифти		
		Шрифт 1	**Шрифт 2**	**Шрифт 3**
1	Arial	Arial Unicode MS	Microsoft Sans Serif	Arial GEO
2	Book Antiqua	Palatino Linotype	Caudex	Mergenthaler Antiqua
3	Candara	Ebrima	Microsoft New Tai Lue	Khmer UI
4	Century	Century Old Style Std	CenturyExpd BT	Century751 BT
5	Calibri	Gisha	Leelawadee	Liberation Serif
6	Times New Roman	Tinos	Georgia	Thorndale
7	Courier New	Courier New CE	Courant	SmallTypeWriting
8	Franklin Gothic Book	Ebrima	Corbel	Trebuchet MS

2. Створити таблицю кодування. Кодування кожного символу в секретному повідомленні представлено трьома типами шрифтів, таким чином, 27 символів (англійські літери з пробілами) можна приховати в 3 буквах, використовуючи 3 різні шрифти, наприклад: подібний масив шрифтів для документу зі шрифтом Times New Roman:

Times New Roman = {Tinos, Georgia, Thorndale}.

Як можна помітити, якщо код поточного символу (1, 1, 1), то буде використано (перший схожий, перший схожий, перший схожий) шрифти з масиву схожих шрифтів. Також, (1, 2, 2) означає (перший схожий, другий схожий, другий

схожий). Комбінація (3, 1, 1) означає (третій схожий, перший схожий, перший схожий) і так далі. Початок повідомлення починається з першої великої літери у документі. У табл. 3.2 наведено таблицю кодів, яка використовується для наповнення стегоконтейнера.

Таблиця 3.2 – Таблиця кодів для алфавіту

№ з/п	Буква	Шрифт 1	Шрифт 2	Шрифт 3
1	a	1	1	1
2	b	1	1	2
3	c	1	1	3
4	d	1	2	1
5	e	1	2	2
6	f	1	2	3
7	g	1	3	1
8	h	1	3	2
9	i	1	3	3
10	g	2	1	1
11	k	2	1	2
12	l	2	1	3
13	m	2	2	1
14	n	2	2	2
15	o	2	2	3
16	p	2	3	1
17	q	2	3	2
18	r	2	3	3
19	s	3	1	1
20	t	3	1	2
21	u	3	1	3
22	v	3	2	1
23	w	3	2	2
24	x	3	2	3
25	y	3	3	1
26	z	3	3	2
27	space	3	3	3

3.3 Процес приховування (вбудовування) інформації

3.3.1 Перший метод

Секретне повідомлення буде вбудовуватися лише у великі літери документа, оскільки великі літери відрізняються за візерунком від малих літер англійського алфавіту. Процес вбудовування складається з трьох кроків. На першому кроці за допомогою функції *def get_fonts_for_document* визначається шрифт документа для отримання масиву схожих шрифтів (рис. 3.2).

```
def get_fonts_for_document(doc_font):
    conn = sqlite3.connect('fonts.db')
    cursor = conn.cursor()
    cursor.execute('SELECT font1, font2, font3 FROM fonts WHERE doc_font = ?',
(doc_font,))
    fonts = cursor.fetchone()
    conn.close()
    return fonts if fonts else (None, None, None)
```

Рисунок 3.2 – Функція *def get_fonts_for_document*

Ця функція отримує відповідні шрифти для кодування символів, відповідно до шрифту документа. Шрифти зберігаються в базі даних і використовуються для кодування символів у приховане повідомлення. Функція підключається до бази даних SQLite, виконує запит для отримання трьох шрифтів, які відповідають заданому шрифту документа. Якщо відповідність знайдена, функція повертає ці шрифти у вигляді кортежу з трьох рядків. Якщо відповідність не знайдена, функція повертає три значення *None*, що сигналізує про відсутність відповідних шрифтів у базі даних.

На другому кроці сканується документ, щоб знайти англійські великі літери, як уже було сказано, потрібно три великі літери, щоб приховати один символ. Нарешті, на третьому кроці, змінюється шрифт перших трьох великих літер на

схожі шрифти в залежності від коду. Ця логіка виконується у методі *encrypt_capitals* (рис. 3.3).

```
    def encrypt_capitals(file_path, message):
        doc = Document(file_path)
        message_index = 0
        encrypted_chars = 0
        doc_font = doc.paragraphs[0].runs[0].font.name
        fonts = get_fonts_for_document(doc_font)
        if not all(fonts):
            raise ValueError("Font mapping not found for the document font.")
        for paragraph in doc.paragraphs:
            for run in paragraph.runs:
                original_run_text = run.text
                original_font_size = run.font.size
                run.text = ''
                for char in original_run_text:
                    if char.isupper() and message_index < len(message) and encrypted_chars < 3:
                        code = encode_symbol(message[message_index])[encrypted_chars]
                        font_name = get_font_name(code, fonts)
                        new_run = paragraph.add_run(char)
                        new_run.font.name = font_name
                        new_run.font.size = original_font_size
                        encrypted_chars += 1
                        if encrypted_chars == 3:
                            message_index += 1
                            encrypted_chars = 0
                    else:
                        new_run = paragraph.add_run(char)
                        new_run.font.name = run.font.name
                        new_run.font.size = original_font_size
                    new_run.font.color.rgb = run.font.color.rgb if run.font.color else None
        doc.save(file_path)
```

Рисунок 3.3 – Метод *encrypt_capitals*

Метод відкриває заданий користувачем документ, проходить через усі параграфи та сегменти (runs) документа, перевіряючи кожен символ. Якщо символ є великою літерою, функція змінює його шрифт відповідно до коду повідомлення. Кожна літера повідомлення кодується в три шрифти, які замінюють шрифти заголовних букв у тексті документа. Функція також зберігає вихідний розмір шрифту, щоб уникнути зміни форматування тексту та щоб документ майже не відрізнявся від оригіналу. У кінці зміни зберігаються.

В процесі кодування спочатку отримуються шрифти для кодування з бази даних. за допомогою *get_fonts_for_document*. Метод *encrypt_capitals* виконує кодування кожної великої літери у відповідний шрифт:

– *encode_symbol*: кожна велика літера в повідомленні кодується у тризначний код;

– get_font_name: для кожного з цих тризначних кодів отримується відповідний шрифт. Це забезпечує зміну шрифту для кожної великої літери у документі (рис. 3.4).

```
def encode_symbol(letter):
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "
    index = alphabet.index(letter)
    return (index // 9 + 1, index % 9 // 3 + 1, index % 3 + 1)
```

Рисунок 3.4 – Функція *encode_symbol*

Ця функція кодує символ (літеру) у тризначний код. Для цього використовується спеціальний алфавіт, який містить усі великі літери англійського алфавіту і пробіл. Кожен символ має свій унікальний індекс, який ділиться на три частини для створення тризначного коду. Функція повертає цей код у вигляді кортежу з трьох чисел.

Метод *def get_font_name* використовується для отримання назви шрифту за заданим кодом. Він приймає код (число від 1 до 3) і список шрифтів, які були отримані з бази даних. Метод повертає шрифт, який відповідає заданому коду.

Якщо код виходить за межі допустимих значень, повертається *None* (рис. 3.5).

```
def get_font_name(code, fonts):
    return fonts[code - 1] if 0 < code <= len(fonts) else None
```

Рисунок 3.5 – Метод *def get_font_name*

3.3.2 Другий метод

Оскільки перший метод дозволяє приховувати повідомлення тільки у великих літерах документа, то це може стати проблемою коли потрібно зашифрувати достатньо велике повідомлення у маленькому тексті з невеликою кількістю великих літер. Саме тому було додано другий метод, який мав би перевагу у приховуванні повідомлення будь якої довжини методом додавання у кінець документа потрібної кількості букв для приховування заданого повідомлення (рис. 3.6).

```
def encrypt_message(file_path, message):
    doc = Document(file_path)
    paragraph = doc.add_paragraph()

    doc_font = doc.paragraphs[0].runs[0].font.name
    fonts = get_fonts_for_document(doc_font)
    if not all(fonts):
        raise ValueError("Font mapping not found for the document font.")

    for letter in message:
        codes = encode_symbol(letter)
        for code in codes:
            font_name = get_font_name(code, fonts)
            run = paragraph.add_run("X ")
            run.font.name = font_name
            run.font.color.rgb = RGBColor(255, 255, 255)
            run.font.size = Pt(1)
```

Рисунок 3.6 – Метод *def encrypt_message*

Як уже було сказано, цей метод кодує повідомлення у вигляді прихованих символів, що додаються до документа. Створюється новий параграф у документі і додаються до нього приховані символи білого кольору. Кожен символ кодується трьома шрифтами, які відповідають кодам повідомлення. Цей метод дозволяє приховувати повідомлення без зміни основного тексту документа та незважаючи на довжину повідомлення. У кінці зміни зберігаються.

3.4 Відтворення прихованої інформації

Для відтворення інформації виконаємо зворотній до приховування процес – для кожної з трьох великих літер визначимо код одного прихованого символу. Логіку цього процесу представлено у методі *extract_message* (рис. 3.7).

```
def extract_message(file_path):
    doc = Document(file_path)
    codes = []
    for paragraph in doc.paragraphs:
        for run in paragraph.runs:
            if run.text.isupper():
                font_code = get_font_code(run.font.name)
                if font_code != 0:
                    codes.append(font_code)
                    if len(codes) == 3:
                        yield decode_symbol(codes).lower()
                        codes = []
```

Рисунок 3.7 – Метод *def extract_message*

Нижче представлено процес вилучення повідомлення:

- відкрити документ з прихованим повідомленням;
- прочитати кожний фрагмент тексту (*run*) у документі;
- перевірити кожну велику літеру у фрагменті тексту;
- визначити шрифтовий код великої літери;
- зібрати коди для кожних трьох великих літер;

– перетворити тризначний код у символ;
– додати символ до секретного повідомлення.

Для витягу та декодування кодів шрифтів у символи використовуються функції *get_font_code* та *decode_symbol* (рис. 3.8).

```
def get_font_code(font_name):
    conn = sqlite3.connect('fonts.db')
    cursor = conn.cursor()

    cursor.execute('SELECT font1, font2, font3 FROM fonts')
    rows = cursor.fetchall()

    conn.close()

    for row in rows:
        if font_name in row:
            return row.index(font_name) + 1
    return 0
```

Рисунок 3.8 – Функція *get_font_code*

Функція отримує код шрифту за його назвою. Для цього створюється підключення до бази даних SQLite, виконується запит для отримання всіх шрифтів, а потім перевіряється, чи міститься заданий шрифт у результатах запиту. Якщо шрифт знайдено, функція повертає його позицію у списку як код (число від 1 до 3). Якщо шрифт не знайдено повертається 0.

Функція *def decode_symbol* декодує тризначний код у символ. Вона приймає кортеж з трьох чисел, який представляє код символу, і обчислює індекс символу в алфавіті. Індекс використовується для отримання символу з алфавіту і повертає його. Якщо індекс виходить за межі допустимих значень, функція повертає порожній рядок (рис. 3.9).

```
def decode_symbol(codes):
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "
    index = (codes[0] - 1) * 9 + (codes[1] - 1) * 3 + (codes[2] - 1)
    if 0 <= index < len(alphabet):
        return alphabet[index]
    else:
        return ""
```

Рисунок 3.9 – Функція *def decode_symbol*

3.5 Експериментальні результати запропонованих методів

Обидва методи стеганографії тестуватимуться шляхом взяття одного документу з різними типами шрифтів і приховування в них одного і того ж секретного повідомлення. Щоб приховати один символ секретного повідомлення (один символ у трьох великих буквах) потрібно три великих літери. Це означає, що якщо супровідний файл містить шість символів, то можливо заховати в ньому два символи. Результати, отримані у першому експерименті, можна узагальнити до табл. 3.3–3.4. Порівняння розмірів оригінального та стегодокумента показало, що розмір наповненого стегоконтейнера (стегодокумента після приховування) зменшився у середньому на 9,16 % від початкового розміру.

Таблиця 3.3 – Експериментальні результати для першого методу

№ з/п	**Шрифт документа**	**Кількість великих літер**	**Максимальна довжина прихованого повідомлення**	**Зміна розміру документа, %**
1	Arial	51	17	–9,40
2	Book Antiqua			–9,46
3	Candara			–9,64
4	Century			–9,63
5	Calibri			–9,64
6	Times New Roman			–7,24

№ з/п	Шрифт документа	Кількість великих літер	Максимальна довжина прихованого повідомлення	Зміна розміру документа, %
7	Courier New			–9,55
8	Franklin Gothic Book			–8,79
Середнє:				**–9,16745**

Таблиця 3.4 – Експериментальні результати для другого методу

№ з/п	Шрифт документу	Кількість великих літер	Зміна розміру документа, %
1	Arial	51	–11,91
2	Book Antiqua		–11,81
3	Candara		–11,77
4	Century		–11,77
5	Calibri		–11,74
6	Times New Roman		–11,73
7	Courier New		–11,69
8	Franklin Gothic Book		–11,65
Середнє:			**–11,76**

Для другого результату ситуація виявилася схожою – у середньому розмір документа зменшився на 11,76 %.

3.6 Розробка графічного інтерфейсу

У цьому розділі розглядається процес розробки графічного інтерфейсу для системи стеганографії. Для створення користувацького інтерфейсу було обрано бібліотеку Flet. Далі буде детальніше розглянуто цю бібліотеку, її переваги та недоліки, а також причини вибору саме цього інструменту [21].

Бібліотека Flet

Flet – це бібліотека для створення багатоплатформних користувацьких інтерфейсів з використанням мови програмування Python. Вона дозволяє розробникам швидко створювати привабливі та інтерактивні програми для різних платформ, включаючи настільні комп'ютери, мобільні пристрої та веб-браузери. Flet забезпечує простий і зрозумілий інтерфейс, що полегшує розробку графічних інтерфейсів без необхідності використовувати складні фреймворки або платформи.

Переваги бібліотеки Flet:

– простота використання: Flet має інтуїтивно зрозумілий API, що робить його доступним для розробників різного рівня підготовки. Всі основні елементи інтерфейсу, такі як кнопки, текстові поля, діалогові вікна, можуть бути легко створені і налаштовані;

– багатоплатформність: програми, створені за допомогою Flet, можуть працювати на різних платформах, включаючи веб, мобільні пристрої та ПК. Це дозволяє розробникам створювати універсальні рішення, які можуть бути використані на будь-якому пристрої;

– гнучкість: Flet дозволяє розробникам налаштовувати вигляд і поведінку елементів інтерфейсу за допомогою простих налаштувань і властивостей, що забезпечує високу гнучкість і можливість створення унікальних користувацьких інтерфейсів;

– активна спільнота: бібліотека Flet має активну спільноту розробників, що забезпечує швидку підтримку та наявність прикладів і документації.

Під час розробки цієї системи було прийнято рішення використовувати Flet з декількох причин. По-перше, простота використання та інтуїтивно зрозумілий синтаксис дозволили швидко створити необхідний інтерфейс без значних витрат часу на навчання. По-друге, багатоплатформність бібліотеки дозволяє забезпечити доступність програми на різних пристроях і операційних системах.

По-третє, активна спільнота та доступність підтримки допомогли швидко вирішувати виникаючі питання під час розробки.

Графічний інтерфейс складається з двох основних вкладок:

1) вкладка «Hide Text» (рис. 3.10). У цій вкладці користувач може вибрати документ Word, ввести повідомлення для шифрування, вибрати метод шифрування та натиснути кнопку «Hide Message» для приховування повідомлення у вибраному документі;

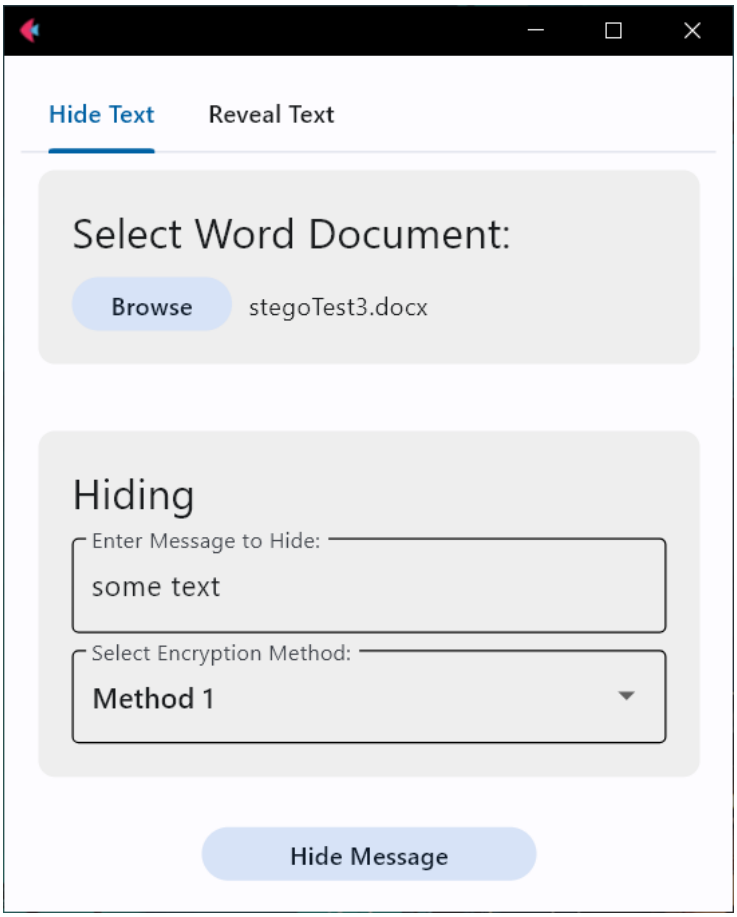


Рисунок 3.10 – Вкладка Hide Text

2) вкладка «Reveal Text» (рис. 3.11). У цій вкладці користувач може вибрати документ Word та натиснути кнопку «Reveal Message» для розшифрування прихованого повідомлення. Розшифроване повідомлення відображається у спеціальному полі.

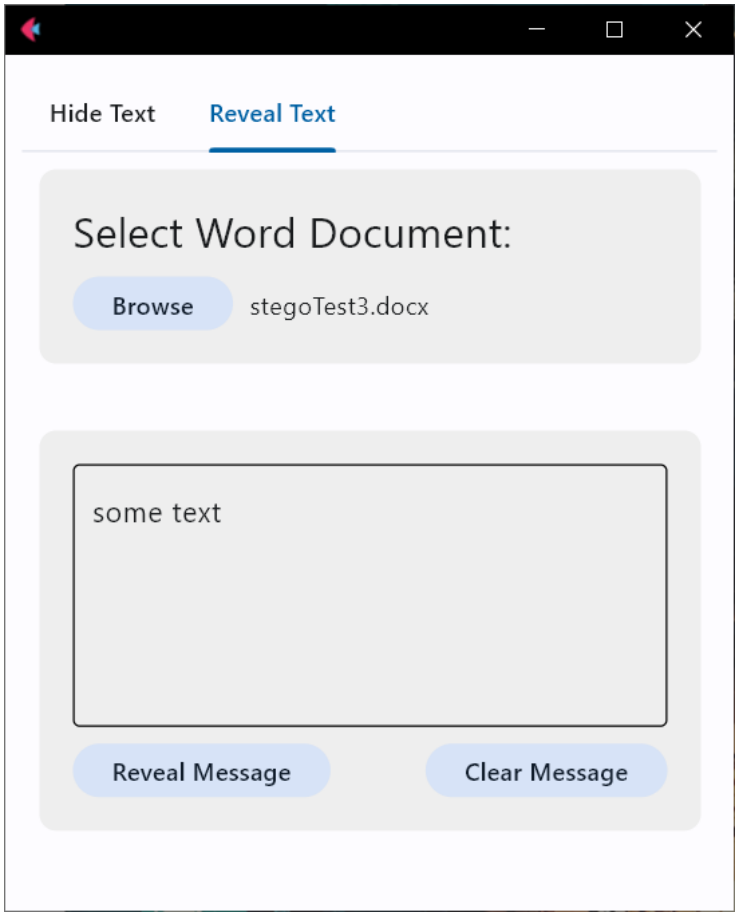
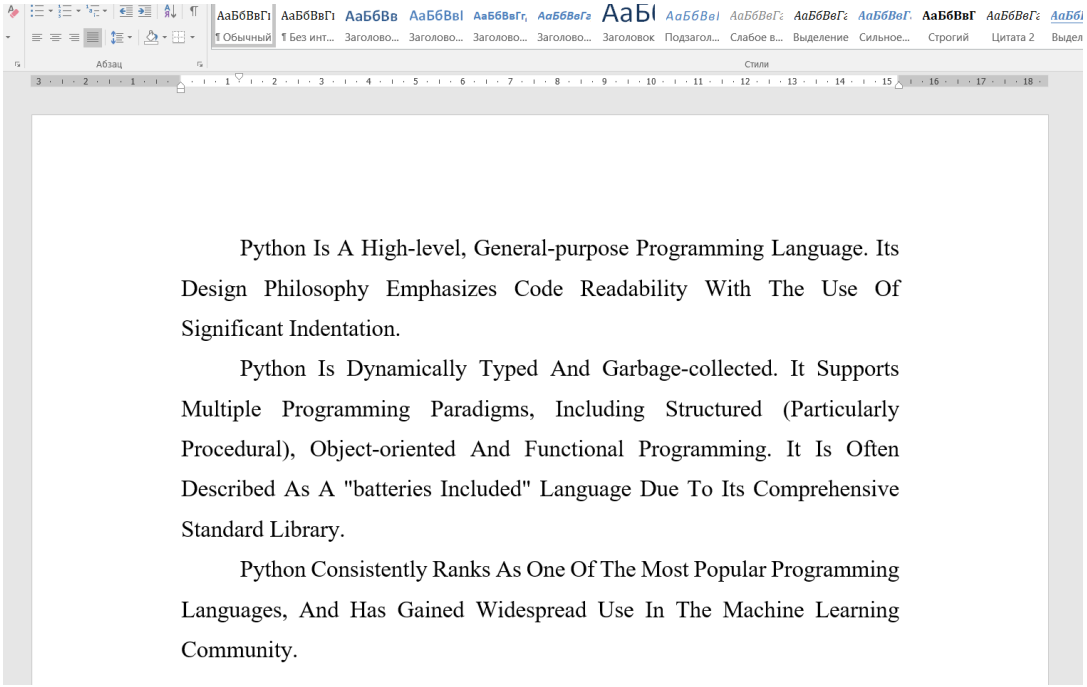


Рисунок 3.11– Вкладка Reveal Text

Кожна вкладка реалізована як окреме вікно з відповідними компонентами, такими як текстові поля, кнопки та випадаючі списки.

3.7 Демонстрація роботи системи

Приклад роботи першого (рис. 3.12–3.13) та другого (рис. 3.14–3.15) методів стегосистеми можна побачити на скріншотах нижче.

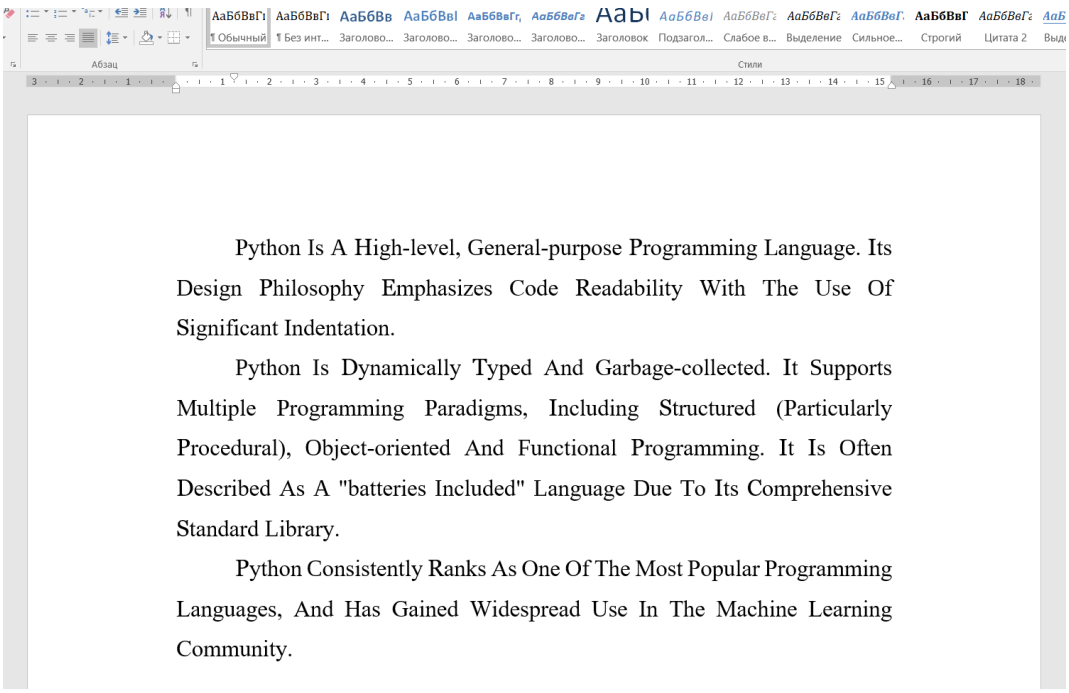


Python Is A High-level, General-purpose Programming Language. Its Design Philosophy Emphasizes Code Readability With The Use Of Significant Indentation.

Python Is Dynamically Typed And Garbage-collected. It Supports Multiple Programming Paradigms, Including Structured (Particularly Procedural), Object-oriented And Functional Programming. It Is Often Described As A "batteries Included" Language Due To Its Comprehensive Standard Library.

Python Consistently Ranks As One Of The Most Popular Programming Languages, And Has Gained Widespread Use In The Machine Learning Community.

Рисунок 3.12 – Оригінальний документ



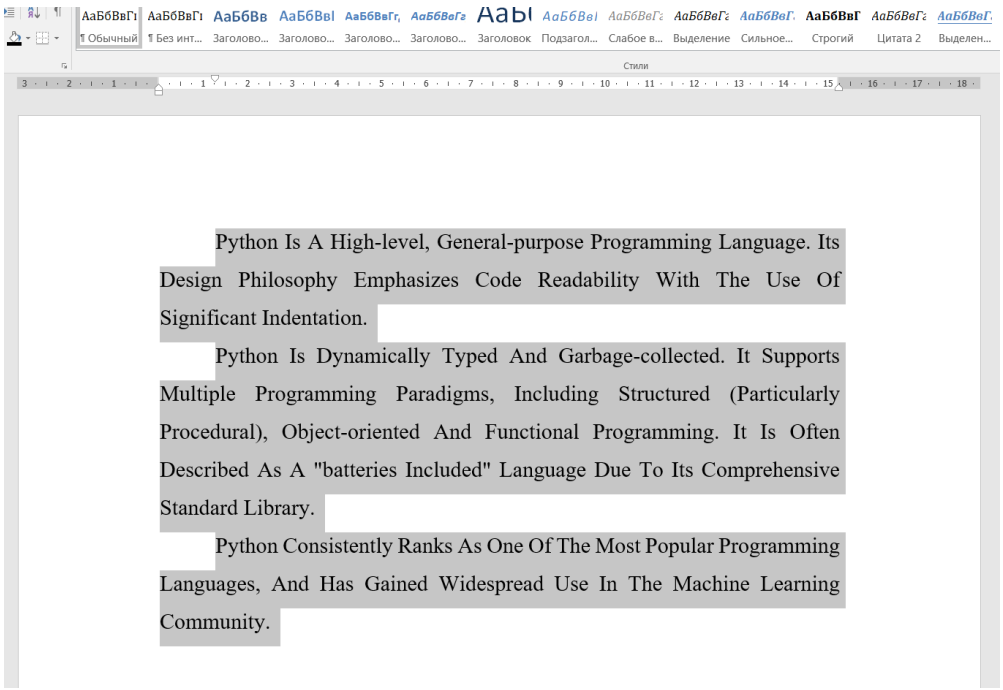
Python Is A High-level, General-purpose Programming Language. Its Design Philosophy Emphasizes Code Readability With The Use Of Significant Indentation.

Python Is Dynamically Typed And Garbage-collected. It Supports Multiple Programming Paradigms, Including Structured (Particularly Procedural), Object-oriented And Functional Programming. It Is Often Described As A "batteries Included" Language Due To Its Comprehensive Standard Library.

Python Consistently Ranks As One Of The Most Popular Programming Languages, And Has Gained Widespread Use In The Machine Learning Community.

Рисунок 3.13 – Стегодокумент

На рис. 3.12 наведено скріншот оригінального документу, на рис. 3.13 – скріншот стегодокумента, у якому зашифровано повідомлення довжиною 19 символів.

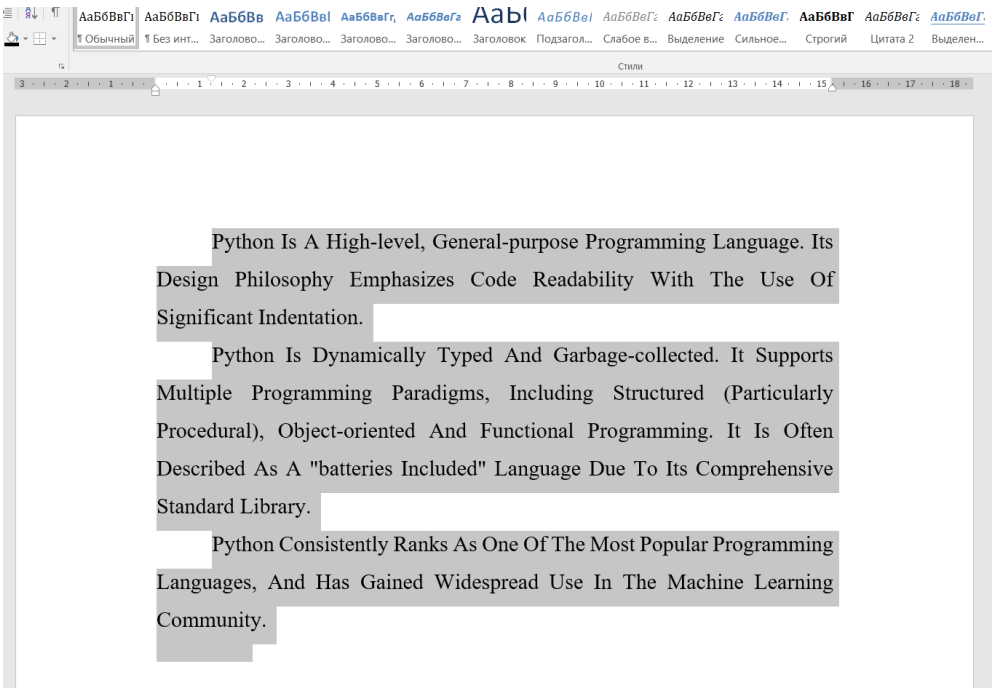


Python Is A High-level, General-purpose Programming Language. Its Design Philosophy Emphasizes Code Readability With The Use Of Significant Indentation.

Python Is Dynamically Typed And Garbage-collected. It Supports Multiple Programming Paradigms, Including Structured (Particularly Procedural), Object-oriented And Functional Programming. It Is Often Described As A "batteries Included" Language Due To Its Comprehensive Standard Library.

Python Consistently Ranks As One Of The Most Popular Programming Languages, And Has Gained Widespread Use In The Machine Learning Community.

Рисунок 3.14 - Оригінальний документ



Python Is A High-level, General-purpose Programming Language. Its Design Philosophy Emphasizes Code Readability With The Use Of Significant Indentation.

Python Is Dynamically Typed And Garbage-collected. It Supports Multiple Programming Paradigms, Including Structured (Particularly Procedural), Object-oriented And Functional Programming. It Is Often Described As A "batteries Included" Language Due To Its Comprehensive Standard Library.

Python Consistently Ranks As One Of The Most Popular Programming Languages, And Has Gained Widespread Use In The Machine Learning Community.

Рисунок 3.15 - Стего-документ

Перший рисунок (рис. 3.14) є скріншотом оригінального документу, а другий (рис. 3.13) - скріншот стего-документа, у якому зашифровано повідомлення довжиною 19 символів. Так як повідомлення дописано білим шрифтом, то помітити його можна лише виконавши деякі маніпуляції з документом.

Висновки до розділу 3

Було створено стеганографічну систему на основі обробки комп'ютерних шрифтів, що включає методи кодування, вбудовування та вилучення прихованих повідомлень. Особлива увага була приділена вибору схожих шрифтів, що є ключовим елементом забезпечення непомітності прихованої інформації. Детально було описано створення масиву схожих шрифтів та таблиці кодування, що дозволяють ефективно приховувати та відтворювати дані.

Було запропоновано два методи вбудовування інформації: перший метод використовує лише великі літери документа, що може бути обмеженням при невеликій кількості великих літер у тексті; другий метод додає приховані символи у кінець документа, що дозволяє приховувати довші повідомлення без зміни основного тексту. Експериментальні результати показали, що обидва методи зменшили розмір документа після приховування на 9,16 % та 11,76 % відповідно.

Також було розглянуто розробку графічного інтерфейсу для стеганографічної системи за допомогою бібліотеки Flet. Використання бібліотеки Flet дозволило швидко і ефективно створити кросплатформний графічний інтерфейс для системи стеганографії на основі оброблення комп’ютерних шрифтів, забезпечуючи зручний та інтуїтивно зрозумілий користувацький досвід. Інтерфейс дозволить користувачам зручно приховувати та розшифровувати повідомлення в текстових документах. Було підкреслено переваги Flet, такі як простота використання, багатоплатформність, гнучкість та активна спільнота розробників, що зробило цю бібліотеку оптимальним вибором для реалізації користувацького інтерфейсу системи.

Таким чином, створена стеганографічна система демонструє ефективність у приховуванні інформації в текстових документах за допомогою маніпуляцій зі схожими шрифтами, забезпечуючи при цьому зручність використання через інтуїтивно зрозумілий графічний інтерфейс.

ВИСНОВКИ

У даній дипломній роботі була розроблена стеганографічна система, яка використовує методи конструювання комп'ютерних шрифтів для приховування інформації в текстових документах. Основою цієї системи стала ідея вбудовування секретних повідомлень у великі літери тексту за допомогою схожих шрифтів, що дозволяє забезпечити доволі високий рівень непомітності, а значить і конфіденційності.

У ході дослідження було здійснено глибокий аналіз існуючих методів стеганографії, визначено їхні переваги та недоліки. Особлива увага приділялася методам приховування інформації за допомогою шрифтів, що є відносно новим напрямом. Було розглянуто різні алгоритми вбудовування та вилучення даних, оцінено їхню стійкість до стеганалітичних атак та ефективність у реальних умовах.

Створена система використовує два основні методи приховування інформації. Перший метод передбачає зміну шрифтів для окремих літер у документі, які відповідають певним кодам прихованого повідомлення. Другий метод полягає у вставленні «невидимих» символів у кінець документа, що приховує повідомлення у вигляді білого тексту на білому фоні. Обидва методи дозволяють зберігати візуальну непомітність прихованих даних для звичайного користувача.

У процесі роботи над системою було розроблено програмний продукт, який включає інтерфейс для користувача, що дозволяє зручно здійснювати приховування та відтворення інформації. Використання бібліотек для роботи з текстовими документами та шрифтами забезпечило інтеграцію системи з існуючими інструментами обробки тексту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Тарасенко Я. В., Півень О. Б., Федотова-Півень І. М. Метод семантичного стиснення текстової інформації для протидії комп’ютерній лінгвістичній стеганографії. *Наука і техніка Повітряних Сил Збройних Сил України*. 2018. № 3 (32). С. 68–78. DOI: 10.30748/nitps.2018.32.10.

2. The New NFT Standard: when cryptography meets steganography. URL: https://medium.com/@zenon.network/the-new-nft-standard-when-cryptography-meets-steganography-9e356007dcaa (Last accessed: 07.04. 2024).

3. Xiao C., Zhang C., Zheng C. FontCode: Embedding Information in Text Documents using Glyph Perturbation. ACM Transactions on Graphics. 2017. Vol. 1, No. 1, Article 1. Publication date: December 2017.

4. Agarwal M. Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA). 2013. Vol. 5, No. 1. DOI: 10.5121/ijnsa.2013.5107.

5. Ali A. A., Seddik A. H. New Text Steganography Technique by Using Mixed-Case Font. International Journal of Computer Applications. 2013. Vol. 62, No. 3. DOI: 10.5120/10058-4650.

6. Baawi S. S., Nasrawi D. A., Abdulameer L. T. Improvement of "Text Steganography Based on Unicode of Characters in Multilingual" by Custom Font with Special Properties. IOP Conf. Series: Materials Science and Engineering. 2020. Vol. 870. Article 012125. DOI: 10.1088/1757-899X/870/1/012125.

7. Ortega J. Mastering Python for networking and security. Packt Publishing, 2018. 426 p.

8. Fincher J. Python IDEs and Code Editors (Guide). *Real Python : Newsletter. Podcast*. URL: http://surl.li/uilkk (Last accessed: 07.04. 2024).

9. What is Steganography? Types, Techniques, Examples & Applications. URL: https://www.simplilearn.com/what-is-steganography-article (Last accessed: 11.04.2024).

10. Weldu T. A. A comparison of different approaches in text steganography. *International Journal of Science and Research (IJSR)*. March 2018. Vol. 7, Is. 3. P. 179–182. DOI: 10.21275/ART2018487. ISSN (Online) 2319-7064.

11. Хорошко В. О., Яремчук Ю. Є., Карпінець В. В. Комп'ютерна стеганографія. Вінниця : ВНТУ, 2017. 155 с.

12. Bhaya W., Rahma A. M., AL-Nasrawi D. Text steganography based on font type in MS-Word documents. *Journal of Computer Science*. 2013. Vol. 9 (7). P. 898–904. ISSN 1549-3636 (Last accessed: 21.04.2024).

13. Nirmatha T. M, Amaresan S. Text stenography using computer fonts. *International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)*. 2017. Vol. 4, Is. 9. P. 9–12 (Last accessed: 18.04.2024).

14. Кузнецов О. О., Євсеєв С. П., Король О. Г. Стеганографія : навчальний посібник. ХНЕУ, 2011. 232 с.

15. Самборський І., Толстова А. Сучасний стан та перспективи розвитку стеганографії у телекомунікаційних системах. *Information Technology and Security*. January–June 2022. Vol. 10. Is. 1 (18). P. 27–38. DOI 10.20535/2411-1031.2022.10.1.261079.

16. Стасюк О. І ., Гнатюк С. О., Довгич Н. І., Літош М. С. Сучасні стеганографічні методи захисту інформації. *Захист інформації*. 2011. № 1. С. 1–7. URL: https://jrnl.nau.edu.ua/index.php/ZI/article/view/1994/1985 (дата звернення: 21.04.2024).

17. Будова символів у шрифті. URL: https://cases.media/en/article/anatomiia-shryftu (дата звернення: 21.04.2024).

18. Wu H., Yang T., Zheng X., Fang Yu. Linguistic steganography and linguistic steganalysis. *In book: Adversarial Multimedia Forensics*. Springer Nature, 2024. P. 163–190. DOI: 10.1007/978-3-031-49803-9_7.

19. Keller J., Langsdorf S. Error codes in and for network steganography. *In book: Architecture of Computing Systems.* Springer Nature, 2023. P. 81–93. DOI: 10.1007/978-3-031-42785-5_6.

20. Agrawal A., Soni R., Tomar A. Perspective chapter: Quantum steganography – encoding secrets in the quantum domain. *In book: Steganography – The Art of Hiding Information.* IntechOpen, 2024. 15 p. DOI: 10.5772/intechopen.1004597.

21. Introduction. What is Flet? URL: https://flet.dev/docs/ (Last accessed: 21.04.2024).

22. Rani, N., Chaudhary, J. Text Steganography Techniques: A Review. International Journal of Engineering Trends and Technology (IJETT). July 2013. Vol. 4, Is. 7. ISSN 2231-5381 (Last accessed: 22.04.2024).

23. Askari, M., Mahmood, A., Iqbal, Z. A novel font color and compression text steganography technique, 2023 International Conference on Communication, Computing and Digital Systems (C-CODE). DOI: 10.1109/C-CODE58145.2023.10139867.

24. El Rahman, S. A. Text Steganography Approaches Using Similarity of English Font Styles. International Journal of Swarm Intelligence. July 2019. DOI: 10.4018/IJSI.2019070102.

ДОДАТОК А

Файл db_create.py

```
import sqlite3

conn = sqlite3.connect('fonts.db')
cursor = conn.cursor()

cursor.execute('''
CREATE TABLE IF NOT EXISTS fonts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    doc_font TEXT NOT NULL,
    font1 TEXT NOT NULL,
    font2 TEXT NOT NULL,
    font3 TEXT NOT NULL
)
''')

cursor.execute('''
INSERT INTO fonts (doc_font, font1, font2, font3) VALUES
('Arial', 'Arial Unicode MS', 'Microsoft Sans Serif', 'Arial GEO'),
('Book Antiqua', 'Palatino Linotype', 'Caudex', 'Mergenthaler Antiqua'),
('Candara', 'Ebrima', 'Microsoft New Tai Lue', 'Khmer UI'),
('Century', 'Century Old Style Std', 'CenturyExpd BT', 'Century751 BT'),
('Calibri', 'Gisha', 'Leelawadee', 'Liberation Serif'),
('Times New Roman', 'Tinos', 'Georgia', 'Thorndale'),
('Courier New', 'Courier New CE', 'Courant', 'SmallTypeWriting'),
('Franklin Gothic Book', 'Ebrima', 'Corbel', 'Trebuchet MS')

''')

conn.commit()
conn.close()
```

ДОДАТОК Б

Файл db_operations.py

```
import sqlite3


def get_fonts_for_document(doc_font):
    conn = sqlite3.connect('fonts.db')
    cursor = conn.cursor()

    cursor.execute('SELECT font1, font2, font3 FROM fonts WHERE doc_font = ?',
(doc_font,))
    fonts = cursor.fetchone()

    conn.close()
    return fonts if fonts else (None, None, None)


def get_font_code(font_name):
    conn = sqlite3.connect('fonts.db')
    cursor = conn.cursor()

    cursor.execute('SELECT font1, font2, font3 FROM fonts')
    rows = cursor.fetchall()

    conn.close()

    for row in rows:
        if font_name in row:
            return row.index(font_name) + 1
    return 0
```

ДОДАТОК В

Файл encoding.py

```
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "


def encode_symbol(letter):
    index = alphabet.index(letter)
    return (index // 9 + 1, index % 9 // 3 + 1, index % 3 + 1)


def decode_symbol(codes):
    index = (codes[0] - 1) * 9 + (codes[1] - 1) * 3 + (codes[2] - 1)
    if 0 <= index < len(alphabet):
        return alphabet[index]
    else:
        return ""
```

ДОДАТОК Г

Файл file_operations.py

```
from docx import Document
from docx.shared import RGBColor, Pt
from db_operations import get_fonts_for_document, get_font_code
from encoding import encode_symbol, decode_symbol

def get_font_name(code, fonts):
    return fonts[code - 1] if 0 < code <= len(fonts) else None

def encrypt_capitals(file_path, message):
    doc = Document(file_path)
    message_index = 0
    encrypted_chars = 0

    doc_font = doc.paragraphs[0].runs[0].font.name
    fonts = get_fonts_for_document(doc_font)
    if not all(fonts):
        raise ValueError("Font mapping not found for the document font.")

    for paragraph in doc.paragraphs:
        for run in paragraph.runs:
            original_run_text = run.text
            original_font_size = run.font.size
            run.text = ''
            for char in original_run_text:
                if char.isupper() and message_index < len(message) and
encrypted_chars < 3:
                    code = encode_symbol(message[message_index])[encrypted_chars]
                    font_name = get_font_name(code, fonts)
                    new_run = paragraph.add_run(char)
                    new_run.font.name = font_name
                    new_run.font.size = original_font_size
                    encrypted_chars += 1
                    if encrypted_chars == 3:
                        message_index += 1
                        encrypted_chars = 0
                else:
                    new_run = paragraph.add_run(char)
                    new_run.font.name = run.font.name
                    new_run.font.size = original_font_size
                    new_run.font.color.rgb = run.font.color.rgb if run.font.color
else None

    doc.save(file_path)

def encrypt_message(file_path, message):
    doc = Document(file_path)
    paragraph = doc.add_paragraph()

    doc_font = doc.paragraphs[0].runs[0].font.name
    fonts = get_fonts_for_document(doc_font)
    if not all(fonts):
        raise ValueError("Font mapping not found for the document font.")

    for letter in message:
        codes = encode_symbol(letter)
```

```
        for code in codes:
            font_name = get_font_name(code, fonts)
            run = paragraph.add_run("X ")
            run.font.name = font_name
            run.font.color.rgb = RGBColor(255, 255, 255)
            run.font.size = Pt(1)

    doc.save(file_path)

def extract_message(file_path):
    doc = Document(file_path)
    codes = []
    for paragraph in doc.paragraphs:
        for run in paragraph.runs:
            if run.text.isupper():
                font_code = get_font_code(run.font.name)
                if font_code != 0:
                    codes.append(font_code)
                    if len(codes) == 3:
                        yield decode_symbol(codes).lower()
                        codes = []
```

ДОДАТОК Д

Файл gui.py

```
import flet as ft
from file_operations import encrypt_capitals, encrypt_message, extract_message

def select_file(dialog, file_path_field, file_name_text, page):
    def pick_file(e):
        if e.files:
            file_path_field.value = e.files[0].path
            file_name_text.value = e.files[0].name
            page.update()

    dialog.on_result = pick_file
    dialog.pick_files(allow_multiple=False, file_type="custom",
allowed_extensions=["docx", "doc"])

def hide_text(e, file_path_field, message_field, method_field, page):
    file_path = file_path_field.value
    message = message_field.value.upper()
    method = method_field.value
    if file_path and message:
        try:
            if method == "Method 1":
                encrypt_capitals(file_path, message)
            elif method == "Method 2":
                encrypt_message(file_path, message)
            page.snack_bar = ft.SnackBar(ft.Text(f"Message encrypted and saved
to: {file_path}"))
        except ValueError as ve:
            page.snack_bar = ft.SnackBar(ft.Text(str(ve)), bgcolor=ft.colors.RED)
        page.snack_bar.open = True
    else:
        page.snack_bar = ft.SnackBar(ft.Text("Please select a file and enter a
message."), bgcolor=ft.colors.RED)
        page.snack_bar.open = True
    page.update()

def reveal_text(e, file_path_extract_field, output_text_field, page):
    file_path = file_path_extract_field.value
    if file_path:
        try:
            message = ''.join(extract_message(file_path))
            output_text_field.value = message
        except ValueError as ve:
            page.snack_bar = ft.SnackBar(ft.Text(str(ve)), bgcolor=ft.colors.RED)
            page.snack_bar.open = True
        page.update()
    else:
        page.snack_bar = ft.SnackBar(ft.Text("Please select a file."),
bgcolor=ft.colors.RED)
        page.snack_bar.open = True
        page.update()

def clear_text(e, output_text_field, page):
    output_text_field.value = ""
    page.update()
```

```
def main(page: ft.Page):
    file_dialog = ft.FilePicker()
    page.overlay.append(file_dialog)

    file_path_field = ft.TextField(visible=False)
    file_name_text = ft.Text()
    file_container = ft.Container(
        content=ft.Column(
            [
                ft.Text("Select Word Document:", style="headlineSmall"),
                ft.Row(
                    [
                        ft.FilledTonalButton(text="Browse", on_click=lambda _:
select_file(file_dialog, file_path_field, file_name_text, page)),
                        file_name_text,
                    ]
                ),
            ],
            spacing=10
        ),
        padding=20,
        border_radius=10,
        bgcolor=ft.colors.GREY_200,
        margin=10
    )

    message_field = ft.TextField(label="Enter Message to Hide:", width=400)
    method_field = ft.Dropdown(
        label="Select Encryption Method:",
        options=[
            ft.dropdown.Option("Method 1"),
            ft.dropdown.Option("Method 2"),
        ],
        value="Method 1",
        width=400,
    )

    message_container = ft.Container(
        content=ft.Column(
            [
                ft.Text("Hiding", style="headlineSmall"),
                message_field,
                method_field,
            ],
            spacing=10
        ),
        padding=20,
        border_radius=10,
        bgcolor=ft.colors.GREY_200,
        margin=10
    )

    hide_button = ft.FilledTonalButton(text="Hide Message", on_click=lambda e:
hide_text(e, file_path_field, message_field, method_field, page), width=200)

    file_path_extract_field = ft.TextField(visible=False)
    file_name_extract_text = ft.Text()
    extract_file_container = ft.Container(
        content=ft.Column(
            [
```

```
                ft.Text("Select Word Document:", style="headlineSmall"),
                ft.Row(
                    [
                        ft.FilledTonalButton(text="Browse", on_click=lambda _:
select_file(file_dialog, file_path_extract_field, file_name_extract_text, page)),
                        file_name_extract_text,
                    ]
                ),
            ],
            spacing=10
        ),
        padding=20,
        border_radius=10,
        bgcolor=ft.colors.GREY_200,
        margin=10
    )

    output_text_field = ft.TextField(value="", read_only=True,
border=ft.InputBorder.OUTLINE, multiline=True, min_lines=5)
    output_container = ft.Container(
        content=ft.Column(
            [
                output_text_field,
                ft.Row(
                    [
                        ft.FilledTonalButton(text="Reveal Message",
on_click=lambda e: reveal_text(e, file_path_extract_field, output_text_field,
page)),
                        ft.FilledTonalButton(text="Clear Message",
on_click=lambda e: clear_text(e, output_text_field, page)),
                    ],
                    alignment=ft.MainAxisAlignment.SPACE_BETWEEN
                )
            ],
            spacing=10
        ),
        padding=20,
        border_radius=10,
        bgcolor=ft.colors.GREY_200,
        margin=10
    )

    page.add(
        ft.Tabs(
            tabs=[
                ft.Tab(
                    text="Hide Text",
                    content=ft.Column(
                        [
                            file_container,
                            message_container,
                            ft.Row([hide_button],
alignment=ft.MainAxisAlignment.CENTER),
                        ],
                        spacing=20,
                    ),
                ),
                ft.Tab(
                    text="Reveal Text",
                    content=ft.Column(
```

```
                    [
                        extract_file_container,
                        output_container,
                    ],
                    spacing=20,
                ),
            ),
        ],
    )
)

page.theme_mode = ft.ThemeMode.LIGHT
page.window_width = 450
page.window_height = 550
page.update()
```

ДОДАТОК Е

Файл main.py

```
import os
import flet as ft
from gui import main


def initialize_database():
    db_path = 'fonts.db'
    if not os.path.exists(db_path):
        print("Database not found. Initializing database...")
        with open('db_create.py') as f:
            code = f.read()
            exec(code)
    else:
        print("Database already initialized.")


initialize_database()

ft.app(target=main)
```