МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

18 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна технологія забезпечення конфіденційності та безпеки даних в цифрових зображеннях»

здобувачки групи ІН.м - 22 Лавренко Анни Олександрівни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Анна ЛАВРЕНКО

(підпис)

Керівник, Наталія БАРЧЕНКО _____

к.т.н.,доцент (підпис)

Консультант, Оксана ГЛАДЧЕНКО _____

старший викладач кафедри (підпис)

іноземних мов та лінгводидактики

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-22 Лавренко Анна Олександрівна

1. Тема роботи: «Інформаційна технологія забезпечення конфіденційності та безпеки даних в цифрових зображеннях»
затверджую наказом по СумДУ від *«06» грудня 2023 р. № 1412-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 18 грудня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1)Аналіз проблеми предметної області, постановка й фомулювання завдань дослідження. 2).Огляд та аналіз теоретичних питань 3). Проектування та розробка інформаційної технології 4). Аналіз результатів*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Визначення предмету дослідження, актуальності та постановка задачі*		
2	*Виконання огляду та аналізу теоретичних питань*		
3	*Проектування та розобка інформаційної технології*		
4	*Аналіз результатів розробки інформаційної технології*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 59 стр., 26 рис., 1 додаток, 21 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі забезпечення конфіденційності та безпеки в цифрових зображеннях за допомогою шифрування.

Об'єкт дослідження — Процес шифрування текстової інформації в цифровому зображені.

Мета роботи — Розробка інформаційної технологія забезпечення конфіденційності та безпеки даних в цифрових зображеннях шляхом реалізації алгоритму шифрування.

Методи дослідження — Алгоритми стеганографії та криптографії в шифруванні текстових даних в цифрових зображеннях.

Результати — Розроблено інформаційну технологію, яка реалізує алгоритм шифрування та дешифрування текстової інформації в цифровому зображенні, що надає можливість користувачу засекречувати дані в фото за допомогою ключа або, навпаки, отримувати авторизований доступ до зашифрованих в цифрових файлах повідомлень, забезпечуючи конфіденційність та безпеку даних. Проведено тестування та аналіз ефективності розробленої інформаційної технології.

ШИФРУВАННЯ, АЛГОРИТМ, КЛЮЧ, ДЕШИФРУВАННЯ, PYTHON, СТЕГАНОГРАФІЯ, PHOTOSHOP, КРИПТОГРАФІЯ, BMP, JPEG

ЗМІСТ

ВСТУП 5
1 АКТУАЛЬНІСТЬ ТА ТЕОРЕТИЧНИЙ ОГЛЯД 6
1.1 Актуальність обраної теми та постановка задачі 6
1.2 Формати цифрових зображень та їх особливості 8
1.3 Принципи шифрування зображення 13
1.4 Аналіз методів стеганографії та криптографії 16
2 АЛГОРИТМИ ШИФРУВАННЯ JPEG ТА BMP ЗОБРАЖЕНЬ 18
2.1 JPEG зображення та методи їх шифрування 18
2.2 BMP зображення та методи їх шифрування 23
2.3 Порівняння алгоритмів шифрування для JPEG та BMP 28
3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 31
3.1 Принципи розробленого алгоритму шифрування 31
3.2 Опис функціоналу реалізованої інформаційної технології 33
3.3 Процес шифрування текстової інформації в фото 36
3.4 Аналіз ефективності інформацйної технології 40
ВИСНОВКИ 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49
ДОДАТОК А 52

ВСТУП

Актуальність. У сучасному діджиталізованому світі, обмін інформацією через Інтернет став звичною справою. Проте, забезпечення конфіденційності та цілісності даних, що передаються таким способом, є важливим завданням. Це особливо актуально в контексті зростаючих загроз кібербезпеці та необхідності захисту інтелектуальної власності. У цьому світлі, шифрування текстової інформації в цифрових зображеннях може бути гарною ідеєю для забезпечення безпечної передачі та зберігання даних.

Об'єкт дослідження. Процес шифрування текстової інформації в цифровому зображенні.

Предмет дослідження. Стеганографічні та криптографічні методи шифрування текстової інформації в цифровому зображенні.

Гіпотеза. Шифрування текстової інформації в цифровому зображенні здатне забезпечити конфіденційність та безпеку інформації при її зберіганні або передачі.

Новизна. Описана в даній роботі інформаційна технологія дозволить засекречувати текстові повідомлення в цифрових файлах за допомогою алгоритму, що поєднує стеганографічні та криптографічні техніки, не впливаючи на візуальну складову зображень.

Структура. Дана робота складається зі вступу, визначення актуальності шифрування та сфери його застосування, постановки задачі, огляду форматів зображень та їх особливостей, дослідження методів стеганографії та криптографії, порівняльного аналізу JPEG та BMP зображень в контексті шифрування, математичної моделі розробленого алгоритму, детального опису компонентів реалізованої інформаційної технології, аналізу її ефективності, висновків, списку використаних джерел та додатків.

1 АКТУАЛЬНІСТЬ ТА ТЕОРЕТИЧНИЙ ОГЛЯД

1.1 Актуальність обраної теми та постановка задачі

Збереження конфеденційності, безпеки даних, захист від атак, перешкоджання отримання несанкціонованого доступу та ряд інших викликів існують не лише на рівні кібербезпеки звичайних користувачів або компаній, а й на рівні держави також. Тому актуальність застосування шифрування зростає щоденно. Шифрування – це технічний процес перетворення інформації за допомогою спеціальних алгоритмів [1]. Метою шифрування є перетворення інформації на неприйнятні, замасковані дані, які користувач отримує або зберігає.

Шифрування здатне забезпечити:

- Конфіденційність інформації. Варто вказати, що основна функція шифрування – це приховати вміст, тобто забезпечити конфіденційність. Коли лише конкретна особа чи система, що володіє ключем, здатна отримати доступ до оригіналу даних[2]. Це здатне зменшити ризики отримання неавторизованого доступу, крадіжки або зловживання інформацією. Забезпечення конфіденційності вкрай важливе при зберіганні та передачі даних по електронній пошті або мережі, наприклад банківські транзакції, особисті повідомлення.
- Інтегрітет даних. Саме шифрування здатне захистит інформацію від модифікацій, оскільки зміна даних в процесі їх передачі, може спричинити втрату цілісності. Також в деяких підходах шифрування існує механізм детекції. Тобто в разі зміни або пошкодження даних, механізм може повідомити про це та дати зелене світло на прийняття певних заходів.

- Аутентифікація. Деякі системи використовують шифрування для аутентифікації пристроїв або користувачів. Найбільш розмовсюдженої практикою це є в області мережевої безпеки, де шифрування використовують для перевірки ідентичності сторін комунікації.
- Попередження несанкціонованого доступу та захист від атак. Шифрування є ефективним інструментом для захисту від різноманітних загроз: перехоплення пакетів, атаки на середовище та ключі, фішинг та втручання в додатки.
- Виконання нормативно-правових актів. Багато галузевих нормативних документів вимагають від компаній шифрування даних користувача, наприклад, HIPAA, PCI-DSS, FIPS, GDPR [2].

Вцілому шифрування сприяє створенню безпечного середовища для передачі та обробки інформації, застосування якого поширується на різноманітні сфери: захист персональної інформації, забезпечення безпеки бізнесу, банківські транзакції, медичні записи, телекомунікація, військовий звязок.

Враховуючи, що більшість інформації поширюється через цифрові медіа та обмін інформацією через Інтернет зростає щоденно, цифрова епоха вимагає забезпечення конфіденційності та безпеки особистих або комерційних даних, які є вбудованими в зображення. Імплементація шифрування зображень має різні виклики, але не дивлячись на це, існує ряд методів та алгоритмів шифрування, дослідження та застосування яких здатно ефективно вплинути на забезпечення конфеденційності та безпеки даних [3].

З оглядом на широкий спектр можливостей шифрування та актуальності його застосування в сучасному світі, в рамках даної магістерської роботи було сформовано наступні задачі:

1. Виконати огляд форматів зображень та їх особливості.
2. Розглянути поняття стеганографії та криптографії, проаналізувати основні методи цих технологій.
3. Виконати порівняльний аналіз JPEG та BMP зображень в контексті шифрування.
4. Створити математичну модель алгоритму, що буде покладено в основну інформаційної системи.
5. Реалізувати програмний продукт, описати основні комопненти та функції.
6. Провести аналіз отриманих результатів розробки інформаційної технології. Визначити її ефективність.

1.2 Формати цифрових зображень та їх особливості

Основні формати зображень, які широко використовуються у цифровій обробці, включають JPEG, PNG, GIF, BMP та TIFF[3]. Кожен з цих форматів має унікальні характеристики, що роблять його більш або менш підходящим для певних застосувань:

JPEG (Joint Photographic Experts Group) — формат зі стисненням із втратами, який найкраще підходить для зображень з багатими деталями та гладкими переходами кольорів, наприклад, для фотографій [3]. Він підтримує високу кольорову глибину та ефективно зменшує розміри файлів, але стиснення може ускладнити застосування деяких методів стеганографії через

потенційну втрату прихованої інформації. JPEG також підтримує Exif-метадані, які можна використовувати для додаткового шифрування.

- Стиснення: Зі стисненням із втратами.
- Кольорові Простори: Підтримує кольорові простори, такі як sRGB і Adobe RGB.
- Особливості: Оптимізований для фотографій з гладкими переходами кольорів.
- Використання: Широко використовується для зображень в Інтернеті, завдяки гарному співвідношенню якості та розміру файлу.

PNG (Portable Network Graphics) — формат зображення без втрат, що підтримує альфа-канал для прозорості, роблячи його ідеальним для зображень, які вимагають прозорого фону, наприклад, веб-графіки та логотипів [4]. PNG зберігає високу якість зображення після стиснення і дозволяє більш ефективне приховування даних у порівнянні з JPEG, завдяки відсутності втрат при стисненні.

- Стиснення: Без втрат.
- Прозорість: Підтримує альфа-канал для прозорості.
- Особливості: Підтримує гамму-корекцію та двовимірне інтерлейсування.
- Використання: Чудово підходить для Інтернету, збереження зображень з прозорістю та зображень з різкими краями, як-от логотипи.

GIF (Graphics Interchange Format) — це формат зі стисненням без втрат, який підтримує анімацію та прозорість (один прозорий колір). Його обмежена палітра (до 256 кольорів) робить його менш підходящим для фотографічних зображень, але він є вибором для простих анімаційних ефектів та зображень із невеликою кількістю кольорів.

- Стиснення: Без втрат, але з обмеженою палітрою в 256 кольорів.

- Анімація: Підтримує просту анімацію.
- Прозорість: Обмежена підтримка прозорості (тільки один колір може бути прозорим).
- Використання: Часто використовується для анімацій та простих зображень на веб-сайтах.

BMP (Bitmap Image File) — це формат зображення без стиснення, що забезпечує високу кольорову глибину та точність. Через його простоту і швидкість обробки він широко використовується в операційних системах Windows [4]. BMP ідеально підходить для методів стеганографії, які вимагають безвтратності, але великі розміри файлів можуть бути незручними для передачі через Інтернет.

- Стиснення: Зазвичай без стиснення, хоча підтримує стиснення RLE.
- Кольорова Глибина: Може містити зображення з високою кольоровою глибиною.
- Особливості: Простий формат, легко обробляється більшістю програмного забезпечення.
- Використання: Часто використовується в ОС Windows, ідеальний для стеганографії.

TIFF (Tagged Image File Format) TIFF — це гнучкий формат зображення, який підтримує стиснення як із втратами, так і без втрат. Це робить його підходящим для збереження високоякісних зображень та сканування документів. TIFF може містити множину зображень або сторінок у одному файлі та підтримує розширені опції метаданих, що може бути корисним для шифрування.

- Стиснення: Підтримує як стиснення без втрат (наприклад, LZW), так і зі стисненням із втратами.
- Гнучкість: Підтримує різні кольорові простори та кольорові моделі.

- Особливості: Може містити багатосторінкові зображення та багато додаткової інформації (метаданих).
- Використання: Широко використовується у видавництві, поліграфії, фотографії, де потрібна висока якість зображення.

Кожен із цих форматів має свої переваги та недоліки, що робить його підходящим для різних цілей використання. Наприклад, формат JPEG є вибором номер один для фотографій через його ефективність збереження файлів, в той час як PNG краще підходить для зображень, де важлива прозорість та чіткість країв. TIFF використовується для зберігання великих зображень у високій якості без втрати даних, що ідеально підходить для друку та архівації. GIF використовується для простих анімацій, а BMP залишається популярним вибором для робочих середовищ Windows і задач, що потребують простоти обробки зображення.

Таблиця 1.1 – Порівння плюсів та мінусів та всіх форматів

Формат	Стиснення	Прозорість	Глибина кольору	Анімація	Плюси для Шифрування	Мінуси для Шифрування
JPEG	З втратами	Ні	Висока (до 24-біт)	Ні	Широко використовуєтьс я, добре для фото	Стиснення з втратами може пошкоджувати вбудовані дані

PNG	Без втрат	Так	(альфа-канал)Висока (до 48-біт)	Ні	Підтримка прозорості та якості без втрат	Більший розмір файлу, ніж у JPEG
GIF	Без втрат (обмежені кольори)	Так	Обмежена (8-біт)	Так	Проста анімація та малий розмір файлу	Обмежена палітра кольорів може обмежувати дані
BMP	Немає або RLE	Ні	Дуже висока (до 32-біт)	Ні	Простота, відсутність стиснення (ідеально для стеганографії)	Великий розмір файлу без стиснення Нестиснений формат точно зберігає дані TIFF
TIFF	Залежно від стиснення	Ні	Дуже висока (до 48-біт)	Ні	Гнучкість, підтримка багатосторінкових зображень та метаданих	Може бути складним у використанні через багато опцій

1.3 Принципи шифрування зображення

Принципи шифрування інформації у цифрових зображеннях базуються на ідеї забезпечення конфіденційності та цілісності даних, які вбудовуються або передаються через візуальні медіа. Основною ціллю такого шифрування є захист інформації від несанкціонованого доступу або виявлення [5]. Для приховування повідомлень в зображеннях, використовуються стеганографічні методи. Двома основними методами є метод заміни пікселів та метод скремблювання пікселів [6].

Метод заміни пікселів включає в себе зміну значень певних пікселів для вбудовування інформації в зображення. Це може бути виконано різними способами:

1. **Найменш значущі біти (LSB)**: принуип полягає в тому, що найменш значущі біти мало впливають на загальний колір пікселя, зміна їх не вносить помітних візуальних змін [7] (Рисунок 1.1).
2. **Колірні канали**: замість зміни всіх каналів пікселя, можна змінювати значення лише в одному або двох каналах (наприклад, тільки в синьому або зеленому), щоб зменшити візуальний ефект.
3. **Палітра Колірного Кодування**: у форматах з палітрою, як-от GIF, можна замінювати кольори в палітрі, що відповідають певним пікселям, для вбудовування інформації.

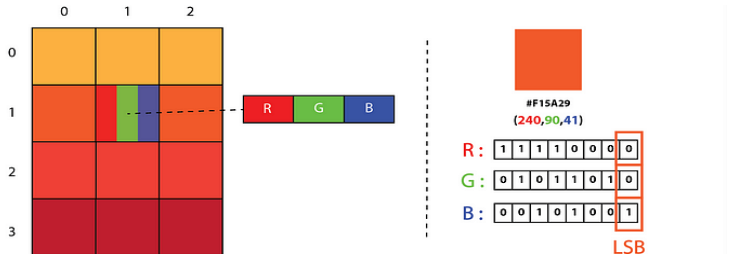

Рисунок 1.1 – Принцип роботи стеганографія зображення lsb

Метод скремблювання пікселів зосереджується на зміні положення пікселів в зображенні, тим самим змішуючи вміст інформації [8]. Цей метод може бути виконаний кількома способами:

1. **Перестановка**: пікселі або блоки пікселів зображення переставляються відповідно до певного алгоритму. Така перестановка забезпечує зміну візуальної інформації без зміни фактичних кольорових значень.
2. **Заміна та віддзеркалення**: пікселі можуть бути замінені або віддзеркалені для створення складного візерунка, який ускладнює розшифровку без знання оригінального алгоритму перетасовки.
3. **Псевдовипадкове розташування**: використання псевдовипадкових чисел для визначення нового положення пікселів забезпечує високий рівень випадковості в розміщенні, роблячи шаблон скремблювання менш передбачуваним [8].

Окрім стеганографії, можливо використовувати криптографічне шифрування для захисту інформації в зображеннях [9].

Спеціальний алгоритм шифрування ще називають криптографічним ключем. Криптографічний ключ – це параметри, що є відомими математичними величинами, та узгоджені між відправником та отримувачем. Саме ключ виступає важливою частиною в криптосистемах, оскільки від його вибору та обробки залежить безпека шифрування[10]. Криптографія включає в себе:

1. **Симетричне Шифрування**. Дані шифруються та розшифровуються з використанням одного і того ж ключа (Рисунок 1.2). Алгоритми, як AES (Advanced Encryption Standard), часто використовуються для шифрування цілих зображень або вбудованих даних [11].

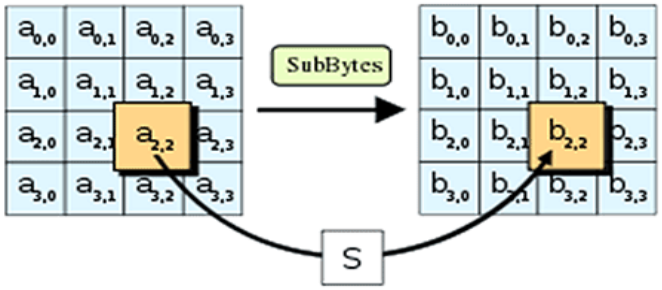

Рисунок 1.2 – Принцип роботи алгоритм шифрування AES

2. **Асиметричне Шифрування**: Включає в себе використання пари ключів — відкритого та приватного (Рисунок 1.3). RSA є популярним алгоритмом асиметричного шифрування, який може бути використаний для захисту інформації, переданої через зображення [12].

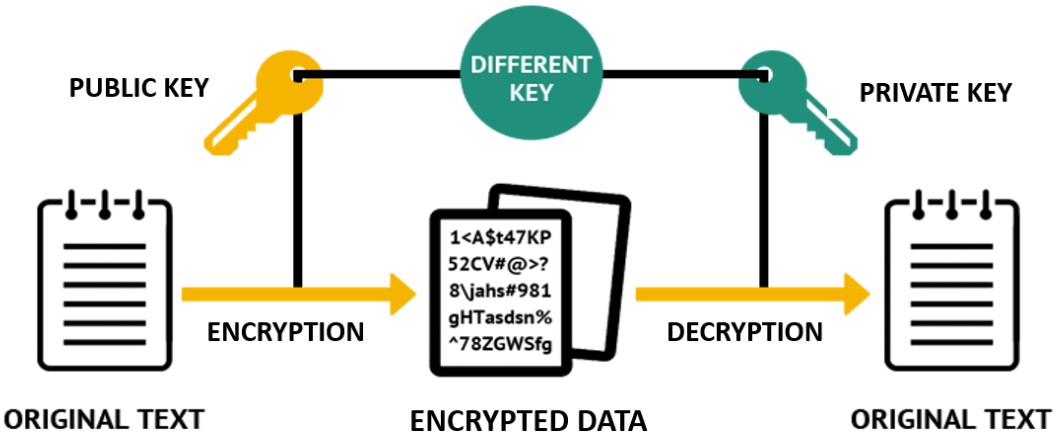

Рисунок 1.3 – Принцип роботи алгоритм шифрування RSA

Важливо зазначити, що кожен метод має свої переваги та недоліки, і вибір методу залежить від конкретних вимог до безпеки, видимості змін та обсягу даних, які потрібно передати. Крім того, ефективність шифрування залежить від якості використовуваної криптографічної системи та її стійкості до криптоаналізу.

1.4 Аналіз методів стеганографії та криптографії

Стеганографія – це метод приховування інформації всередині іншого носія, такого як цифрове зображення, аудіофайл або відео, таким чином, що сам факт наявності прихованої інформації залишається невідомим. У контексті цифрових зображень, стеганографія зазвичай включає вбудовування даних у пікселі зображення. Проаналізуємо детальніше два, найбільш розповсюджених метода: метод найменш значуших бітів (LSB) та метод маскування та фільтрації.

Метод LSB є одним з найпростіших, але ефективних підходів у стеганографії. Цей метод полягає у заміні одного або декількох найменш значущих бітів пікселів на біти прихованої інформації [13] . Наприклад, якщо маємо байт прихованої інформації b, то він може бути вбудований у трьох пікселях зображення p1,p2,p3 наступним чином:

$$p_1' = (p_1 \wedge 0x_{FE}) \vee (b_1 \wedge 0x_{01})$$
$$p_2' = (p_2 \wedge 0x_{FE}) \vee (b_2 \wedge 0x_{01})$$
$$p_3' = (p_3 \wedge 0x_{FE}) \vee (b_3 \wedge 0x_{01}) \quad (1.1)$$

де p1′,p2′,p3′ — нові значення пікселів після вбудовування бітів b1,b2,b3, які є частинами байта b.

Методи Маскування та Фільтрації – полягає у модифікації пікселів зображення таким чином, щоб вбудувати інформацію у вигляді візуального шуму або малозначних змін у текстурах [14]. Це може бути виконано шляхом накладання маски на певні області зображення або застосуванням фільтрів, які змінюють характеристики певних пікселів.

Основним викликом у стеганографії є збалансування між приховуванням достатньої кількості даних і збереженням оригінального вигляду зображення. Занадто великі зміни можуть зробити приховані дані помітними, що знижує ефективність стеганографії.

Стеганографію часто використовують разом з криптографічними методами, щоб забезпечити двошаровий захист інформації. Спочатку дані шифруються за допомогою криптографічного алгоритму, а потім вбудовуються у зображення за допомогою стеганографії. Навіть якщо стеганографічний шар буде виявлений, самі дані залишатимуться захищеними завдяки криптографії.

2 АЛГОРИТМИ ШИФРУВАННЯ JPEG ТА BMP ЗОБРАЖЕНЬ

2.1 JPEG зображення та методи їх шифрування

JPEG розшифровується як Joint Photographic Experts Group, особливістю якого є стиснення, яке виконується щоб зменшити розмір файлу без шкоди для його якості. Зменшивши розмір, можливо зберігати його у величезній кількості. Зменшення розміру зображень також покращує ефективність системи, оскільки зменшується навантаження на неї (Рисунок 2.1).

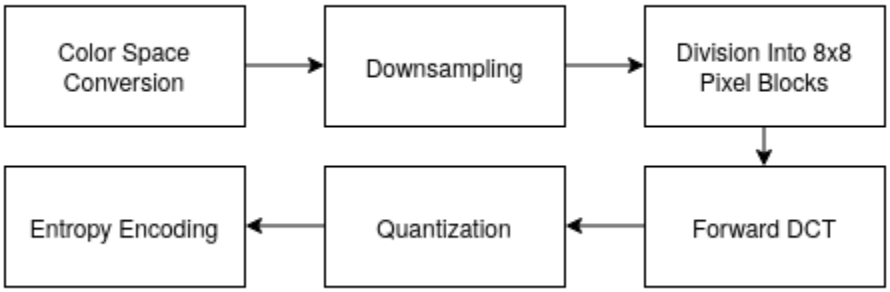

Рисунок 2.1 – Принцип створення файлу зображення *.jpg* .

Процес стиснення JPEG: спочатку перетворюється колірний формат R, G, B у формат Y, Cb, Cr. Деякі кольори більш чутливі для людського ока, тому є високочастотними. Деякі кольори сполук хрому, як-от Cb і Cr, менш чутливі для очей людини, тому їх можна ігнорувати [15].

Колірний простір — це певна організація кольорів. Прикладами колірних просторів є RGB (червоний, зелений і синій) і CMYK (блакитний, пурпуровий, жовтий і чорний). Крім того, кожен піксель на зображенні має свої значення колірного простору [16].

Людське око більш чутливе до яскравості, ніж до кольору. Таким чином, алгоритм може використовувати це, вносячи значні зміни в інформацію про

колір зображення, не впливаючи на сприйману якість зображення . Для цього JPEG має спочатку змінити колірний простір зображення з RGB на YCbCr.

YCbCr складається з 3 різних каналів: Y — це канал яскравості, що містить інформацію про яскравість, а Cb і Cr — канали кольоровості синього та кольоровості червоного, що містять інформацію про колір [16]. У цьому колірному просторі JPEG може вносити зміни в канал кольоровості, щоб змінити інформацію про колір зображення, не впливаючи на інформацію про яскравість, яка міститься в каналі яскравості.

Після перетворення кожен піксель має три нові значення, які представляють інформацію про яскравість, кольоровість синього та кольоровість червоного.

Після того, як відокремлюємо інформацію про колір від інформації про яскравість, JPEG зменшує дискретизацію каналів кольоровості до чверті їх початкового розміру. Кожен блок із 4 пікселів усереднюється в одне значення кольору для всіх 4 пікселів. У результаті частина інформації втрачається, а розмір зображення зменшується вдвічі, але оскільки людське око не дуже чутливе до кольору, зміни нелегко розрізнити.

Важливо зазначити, що зменшення дискретизації застосовується лише до каналів кольоровості, а канал яскравості зберігає свій початковий розмір .

Після зменшення дискретизації піксельні дані кожного каналу поділяються на блоки 8×8 по 64 пікселя. Відтепер алгоритм обробляє кожен блок пікселів незалежно.

Спочатку значення кожного пікселя для кожного каналу віднімається на 128, щоб отримати діапазон значень від -128 до +127. За допомогою DCT для кожного каналу кожен блок із 64 пікселів може бути реконструйований шляхом множення постійного набору базових зображень на відповідні

значення ваги, а потім їх підсумовування (Рисунок 2.2).

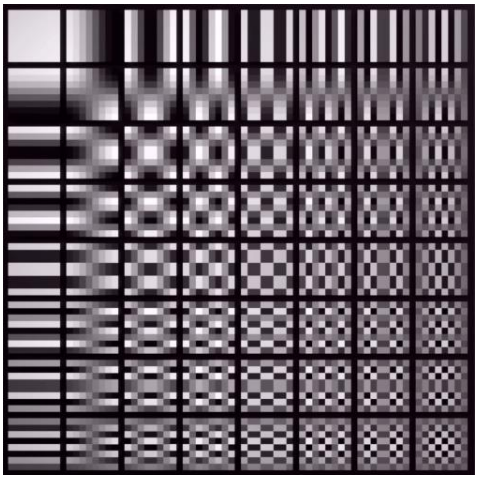

Рисунок 2.2 – Формат базового зображення:

Застосування прямого DCT до кожного блоку з 64 пікселів дає нам матрицю, що містить відповідні значення ваги для кожного базового зображення . Ці значення показують, скільки кожного базового зображення використовується для реконструкції оригінального блоку 8×8 пікселів, і це виглядає приблизно так:

$$\begin{bmatrix} 463 & -170 & -15 & 33 & -33 & 8 & 4 & -5 \\ -197 & 1 & 40 & 0 & 10 & 5 & 1 & -1 \\ 36 & 41 & 9 & -17 & 12 & -11 & 0 & 0 \\ -5 & -25 & 0 & -4 & 0 & 8 & 0 & 0 \\ 4 & 9 & 0 & 5 & 2 & -5 & 4 & -5 \\ -2 & -1 & 3 & 0 & 2 & -1 & -2 & 4 \\ -8 & 5 & 4 & -6 & 6 & -1 & -4 & 5 \\ 7 & -1 & -2 & 0 & 3 & 4 & -2 & 0 \end{bmatrix}$$

Для кожного блоку з 64 пікселів ми матимемо три вагові матриці, одну для яскравості та дві для кольоровості. Крім того, на цьому етапі жодна інформація не втрачається.

Людське око не дуже добре сприймає високочастотні елементи зображення. На цьому кроці JPEG видаляє частину цієї високочастотної інформації, не впливаючи на сприйману якість зображення .

Для цього вагова матриця ділиться на попередньо розраховану таблицю квантування, а результати округлюються до найближчого цілого числа. Ось як виглядає таблиця квантування для каналів кольоровості:

$$\begin{bmatrix} 10 & 8 & 9 & 9 & 9 & 8 & 10 & 9 \\ 9 & 9 & 10 & 10 & 10 & 11 & 12 & 17 \\ 13 & 12 & 12 & 12 & 12 & 20 & 16 & 16 \\ 14 & 17 & 18 & 20 & 23 & 23 & 22 & 20 \\ 25 & 25 & 25 & 25 & 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 & 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 & 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 & 25 & 25 & 25 & 25 \end{bmatrix}$$

І для каналу яскравості:

$$\begin{bmatrix} 6 & 4 & 4 & 6 & 10 & 16 & 20 & 24 \\ 5 & 5 & 6 & 8 & 10 & 23 & 24 & 22 \\ 6 & 5 & 6 & 10 & 16 & 23 & 28 & 22 \\ 6 & 7 & 9 & 12 & 20 & 35 & 32 & 25 \\ 7 & 9 & 15 & 22 & 27 & 44 & 41 & 31 \\ 10 & 14 & 22 & 26 & 32 & 42 & 45 & 37 \\ 20 & 26 & 31 & 35 & 41 & 48 & 48 & 40 \\ 29 & 37 & 38 & 39 & 45 & 40 & 41 & 40 \end{bmatrix}$$

Ми бачимо, що таблиці квантування мають більші числа в нижньому правому куті, де знаходяться високочастотні дані, і нижчі числа у верхньому лівому куті, де розташовані низькочастотні дані. У результаті після ділення кожного значення ваги високочастотні дані округляються до 0:

$$\begin{bmatrix} -78 & -61 & -36 & -13 & -16 & -10 & -2 & 1 \\ -45 & -33 & -25 & -13 & -15 & -7 & -3 & -2 \\ 9 & 7 & 2 & -2 & -1 & 0 & 0 & 0 \\ -10 & -7 & -5 & -4 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Тепер у є три матриці, що містять цілі значення для кожного блоку з 64 пікселів, одна для яскравості та дві для кольоровості.

Тепер все, що залишилося зробити, це перерахувати значення, які знаходяться всередині кожної матриці, запустити RLE (Run Length Encoding), оскільки багато нулів, а потім запустити алгоритм кодування Хаффмана перед збереженням даних.

RLE і кодування Хаффмана — це алгоритми стиснення без втрат, які зменшують простір, необхідний для зберігання інформації, не видаляючи жодних даних [17].

Як бачимо, результат квантування має багато нулів у нижньому лівому куті. Щоб зберегти надлишковість 0, алгоритм JPEG виконує зигзагоподібне сканування, яке зберігає всі нулі разом (Рисунок 2.3).

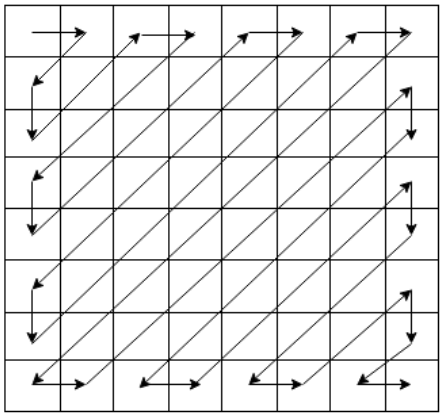

Рисунок 2.3 – Алгоритм зигзагоподібного сканування для JPEG

Після зигзагоподібного сканування отримуємо список цілих значень, а після запуску RLE у списку отримуємо кожне число та кількість повторів.

Нарешті, JPEG використовує кодування Хаффмана для результату, і значення, які мають більшу частоту (наприклад, нулі), представлені меншою

кількістю бітів, ніж менш часті значення. Під час цієї фази дані не втрачаються.

Щоразу, при відкриті файлу .jpg за допомогою програми перегляду зображень, файл потрібно декодувати, перш ніж його можна буде відобразити. Щоб декодувати зображення *.jpg*, програма перегляду зображень виконує всі описані вище дії у зворотному порядку :

1. Запустити Huffman Decoding для даних файлу
2. Деконструювати RLE
3. Помістити список чисел у матрицю 8×8
4. Помножити цілі значення на відповідну таблицю квантування
5. Помножити значення ваги на відповідні базові зображення, а потім підсумувати їх, щоб отримати значення пікселів
6. Розширити канали кольоровості
7. Змінити колірний простір з YCbCr на RGB
8. Відобразити зображення

2.2 BMP зображення та методи їх шифрування

Формат BMP існує з перших днів персональних комп'ютерів. Вперше його було представлено в операційній системі Windows 1.0 у 1985 році. Ініціали «BMP» означають Bitmap Picture. Формат був розроблений як метод зберігання піксельних зображень таким чином, щоб їх можна було легко відобразити на екрані комп'ютера [18].

Microsoft розробила та підтримує формат BMP. Протягом багатьох років цей формат пережив кілька переглядів із додаванням нових колірних режимів і можливістю зберігати більше інформації про зображення. Оригінальна версія, відома як BMP версія 1, підтримувала лише 1-бітний колір,

еквівалентний чорно-білому. Проте з випуском Windows 3.0 була представлена версія BMP 2, яка підтримувала 8-бітний колір, дозволяючи відображати ширший діапазон кольорів на зображенні.

Версія BMP 3, випущена з Windows NT 3.1, побачила додавання реальних кольорів (24-біт) і кодування довжини для растрових зображень. Це дозволило включити в зображення широкий діапазон кольорів і запропонувало форму стиснення для зменшення розміру файлу. У версії BMP 4, представленій у Windows 95, додано підтримку альфа-каналів і типів простору кольорів. Наразі BMP версії 5 є останньою версією, доданою разом із Windows 98 для підтримки профілів ICC [18].

Характеристики файлів BMP: Файл BMP, як і будь-яка растрова графіка, складається з сітки крихітних пікселів, кожен з яких містить певну інформацію про колір. Ця інформація про колір зберігається як набір значень, який може бути прочитаний комп'ютером для відтворення зображення. Формат BMP підтримує різні глибини кольору, починаючи від 1-бітного (чорно-білий) до 32-бітного (понад 16,7 мільйонів кольорів).

Однією з унікальних характеристик файлів BMP є те, що вони зберігають дані про колір без будь-якого стиснення. Це означає, що інформація про колір для кожного пікселя зберігається точно такою, якою вона є, без втрати якості. Однак це також означає, що файли BMP можуть бути досить великими, особливо для зображень із високою роздільною здатністю.

Файли BMP також містять заголовок растрового файлу, який зберігає інформацію про файл, наприклад його розмір, розміри в пікселях, глибину кольору тощо. Цей заголовок дозволяє комп'ютеру зрозуміти, як читати дані кольору у файлі та відтворювати зображення.

Незважаючи на великі розміри файлів, BMP-файли все ще знаходять деяке застосування до цього дня, хоча це рідко. Одним із таких випадків

використання є розробка програмного забезпечення. Створюючи програмне забезпечення для Windows, розробники іноді використовують файли BMP для піктограм для його рідної підтримки, хоча багато хто перейшов на файли .ico або PNG [18].

Формат BMP також використовується в певних галузях промисловості, як-от медична візуалізація. Зокрема, у старих системах файли BMP досі використовуються для якості зображення без втрат, хоча багато сучасних машин перейшло до спеціалізованого формату під назвою Digital Imaging and Communications in Medicine (DICOM).

Переваги формату BMP:

1. Висока якість і деталізація. Оскільки він зберігає кольорові дані без стиснення, зображення, збережені у форматі BMP, зберігають повну якість і деталізацію.
2. Проста структура. Файли BMP складаються з прямого масиву пікселів, причому колірні дані кожного пікселя зберігаються безпосередньо у файлі. Це робить формат BMP легким для читання та запису, що може бути значною перевагою для певних програм.
3. Нестиснутий характер формату. Оскільки він зберігає кольорові дані без будь-якого стиснення, зображення, збережені у форматі BMP, можна редагувати та повторно зберігати кілька разів без втрати якості.

Недоліки формату BMP:

1. Великий розмір файлу. Оскільки файли BMP зберігають кольорові дані без стиснення, вони можуть бути досить великими, особливо для зображень із високою роздільною здатністю. Це може ускладнити керування файлами BMP.

2. Має обмежені можливості порівняно з іншими форматами зображень. Наприклад, він не підтримує шари або прозорість, які є загальними функціями в більш просунутих форматах зображень, таких як PNG або PSD. Це може обмежити універсальність і функціональність формату BMP для певних програм.
3. BMP формат не підтримується повсюдно. Деякі платформи та пристрої можуть мати проблеми з читанням або відображенням файлів BMP, особливо старі або менш поширені системи. Це може обмежити зручність використання та доступність зображень, збережених у форматі BMP.

Два методи шифрування, які можна застосовувати для зображень у форматі BMP: метод найменш значущих бітів (LSB) та шифрування блоками. Ці методи відрізняються підходами до вбудовування інформації та рівнем безпеки.

Метод Найменш Значущих Бітів (LSB) полягає у заміні найменш значущих бітів у пікселях зображення на біти прихованої інформації. У 24-бітному кольоровому BMP-зображенні, кожен піксель складається з трьох кольорових компонентів (червоний, зелений, синій), кожен з яких представлений 8 бітами.

Якщо P - піксель зображення з компонентами (R,G,B), а D - байт прихованої інформації, тоді кожен біт D вбудовується у LSB відповідних компонентів P. Це можна описати формулами:

$$R' = (R \wedge 0x_{FE}) \vee (D_0)$$
$$G' = (G \wedge 0x_{FE}) \vee (D_1)$$

$$B' = (B \wedge 0x_{FE}) \vee (D_2) \quad (2.1)$$

де R',G',B' - нові значення кольорових компонентів після вбудовування бітів ,$D1$,$D2$ (найменш значущі біти з D).

Шифрування блоками, таке як AES (Advanced Encryption Standard), застосовується до блоків даних, які потім можуть бути вбудовані у зображення. У цьому випадку, блоки даних, які містять текст або іншу інформацію, спочатку шифруються за допомогою вибраного алгоритму, а потім результат шифрування вбудовується у зображення.

Якщо M - блок даних для шифрування, K - ключ шифрування, тоді зашифрований блок C може бути представлений як: $C=EK(M)$, де E - функція шифрування (наприклад, AES), K - ключ шифрування, M - блок відкритого тексту, і C - зашифрований текст.

Зашифрований текст C потім вбудовується у зображення за допомогою методу, аналогічного LSB або іншого підходу стеганографії.

Обидва ці методи демонструють, як можна ефективно використовувати шифрування для забезпечення конфіденційності даних у зображеннях BMP. Вибір конкретного методу залежить від вимог до безпеки, непомітності, та обсягу даних, які потрібно зашифрувати.

2.3 Порівняння алгоритмів шифрування для JPEG та BMP

Шифрування для форматів JPEG та BMP вимагає розгляду особливостей кожного формату, що впливають на підходи до шифрування та стеганографії [19]. Ось основні відмінності та порівняльний аналіз:

1. Особливості Форматів
    - JPEG: Це формат із стисненням із втратами, який використовує дискретне косинусове перетворення (DCT) та квантування для зменшення розміру файлу. Стиснення із втратами ускладнює використання деяких методів стеганографії.
    - BMP: Формат без стиснення, що зберігає всю інформацію про пікселі зображення. Це робить його більш підходящим для простих методів стеганографії, таких як зміна найменш значущих бітів (LSB).
2. Метод Найменш Значущих Бітів (LSB)
    - JPEG: Застосування методу LSB до JPEG може бути проблематичним через стиснення із втратами. Шифрування даних перед стисненням може вплинути на якість зображення та спричинити візуальні артефакти.
    - BMP: Метод LSB ідеально підходить для BMP, оскільки відсутність стиснення забезпечує, що вбудована інформація залишиться незмінною.
3. Шифрування Блоками (наприклад, AES)
    - JPEG: При шифруванні блоками необхідно враховувати, що після стиснення JPEG кінцеве зображення може мати змінені

значення пікселів, що ускладнює точне відновлення шифротексту.

- BMP: Шифрування блоками ефективне для BMP, оскільки кожен зашифрований блок даних може бути точно вбудований у зображення без ризику втрати інформації під час стиснення.

4. Шифрування Хаотичними Ключами
    - JPEG: Хаотичне шифрування може бути використане, але необхідно враховувати потенційні втрати даних після стиснення зображення.
    - BMP: Хаотичні ключі можуть бути ефективно використані для шифрування, забезпечуючи високий рівень безпеки без ризику втрати даних.
5. Загальне Порівняння
    - Стійкість до Втрат: BMP забезпечує більшу стійкість до втрат інформації при шифруванні, в той час як JPEG вимагає більш обережного підходу та попередніх розрахунків [20].
    - Візуальний Вплив: Шифрування в JPEG може впливати на візуальну якість зображення, особливо при високих рівнях стиснення.
    - Гнучкість і Безпека: BMP пропонує більшу гнучкість для різних методів шифрування і стеганографії, але JPEG може вимагати більш складних алгоритмів для досягнення аналогічного рівня безпеки.

Виконавши порівняльний аналіз, можна прийти висновку, що зображення в форматі BMP більш підходить для шифрування, оскільки не використовує стиснення та має високу якість. Тому розроблена інформаційна система буде

включати механізм конвертації JPEG зображення в BMP, і лише потім подальше шифрування.

Таблиця 2.1 – Порівняння особливостей форматів JPEG і BMP

Особливість	JPEG	BMP
Стиснення	Зі стисненням із втратами (DCT-базоване)	Без стиснення (незжатий)
Придатність Методу LSB	Проблематична через стиснення із втратами	Ідеальна, оскільки немає стиснення
Придатність Блокового Шифрування	Ефективна, але необхідно враховувати зміни після стиснення	Висока ефективність, точне вбудовування даних
Шифрування Хаотичними Ключами	Можливе, але з урахуванням можливої втрати даних після стиснення	Висока ефективність, забезпечує високий рівень безпеки
Ризик Втрати Даних	Вищий ризик через стиснення	Низький ризик, оскільки немає втрати даних від стиснення
Візуальний Вплив	Може впливати на якість зображення, особливо при високих рівнях стиснення	Мінімальний вплив, оскільки дані вбудовуються без зміни візуальної якості
Гнучкість та Безпека	Вимагає складніших алгоритмів	Більша гнучкість для різних методів шифрування та стеганографії

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Принципи розробленого алгоритму шифрування

В основу розробленого алгоритму покладено технологію LSB з розбиванням кольору. Розроблено математичну модель процесу шифрування тексту в пікселях зображення, з використанням трьох кольорових каналів (червоного, зеленого, синього) та розбиттям байту символу на частини, які вставлені в кожен канал.

Процес шифрування текстових даних в зображенні:

Нехай T - це текстова стрічка, яка містить n символів $T = \{t_1, t_2, \dots, t_n\}$, кожний символ t_i перетворюємо у відповідний байт. S - сид (ключ шифрування), P - множина пікселів зображення, кожен піксель p_i містить три кольорові компоненти (R_i, G_i, B_i).

1. Кожен символ t_i кодується як один байт b_i, і далі цей байт розділяється на три частини (b_{i1}, b_{i2}, b_{i3}), де:
   - b_{i1} - старші 3 біти b_i,
   - b_{i2} - середні 2 біта b_i,
   - b_{i3} - молодші 3 біти b_i.
2. Генеруємо послідовність псевдовипадкових індексів пікселів $I = \{i_1, i_2, \dots, i_N\}$ за допомогою псевдовипадкового генератора чисел з ключем S.
3. Для кожного індексу i_K, в I, ми модифікуємо відповідний піксель p_{IK}, таким чином:

$$R'_{ik} = (R_{ik} \wedge 0x_{F8}) \vee b_{i1}$$
$$G'_{ik} = (G_{ik} \wedge 0x_{FC}) \vee b_{i2}$$
$$B'_{ik} = (B_{ik} \wedge 0x_{F8}) \vee b_{i3} \quad (2.2)$$

де *Rik′*,*Gik′*,*Bik′* - це нові значення кольорових компонентів після шифрування.

4. Зберігаємо модифіковане зображення.

Процес дешифрування текстової інформації з цифрового зображення:

Для розшифрування, ми виконуємо зворотні операції.

1. Знову генеруємо послідовність псевдовипадкових індексів пікселів *I* з використанням того ж ключа *S*.

2. Для кожного індексу i_k в *I*, ми витягуємо компоненти з зашифрованих пікселів:

$$b_{i1} = R'_{ik} \wedge 0x_{07}$$
$$b_{i2} = G'_{ik} \wedge 0x_{03}$$
$$b_{i3} = B'_{ik} \wedge 0x_{07} \quad (2.2)$$

3. Відновлюємо байт символу, об'єднавши три частини: $b_i = (b_{i1} << 5) \vee (b_{i2} << 3) \vee b_{i3}$

4. Конвертуємо кожен байт b_i назад у символ тексту, щоб відновити оригінальний текст *T*.

3.2 Опис функціоналу реалізованої інформаційної технології

В розділі представлено практичну реалізацію інформаційної технології, яка забезпечує конфіденційність та безпеку інформації, вбудованої в цифрові зображення. Програма, розроблена в рамках цього дослідження, дозволяє користувачу конвертувати JPEG зображення в BMP та зашифрувати текст у за допомогою ключа . Розроблений програмний засіб також надає можливість розшифровувати засекречену попередньо інформацію в цифровому зображені, повертаючи її до початкового стану. Програмна реалізація виконана за допомогою інтерпретованої об’єктно-орієнтованої мови програмування Python [21] .

Розроблене програмне забезпечення включає в себе декілька ключових компонентів:

1. Інтерфейс Користувача: зручний і інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам вибирати зображення, вводити текст для шифрування та обирати ключ шифрування (Рисунок 3.1).

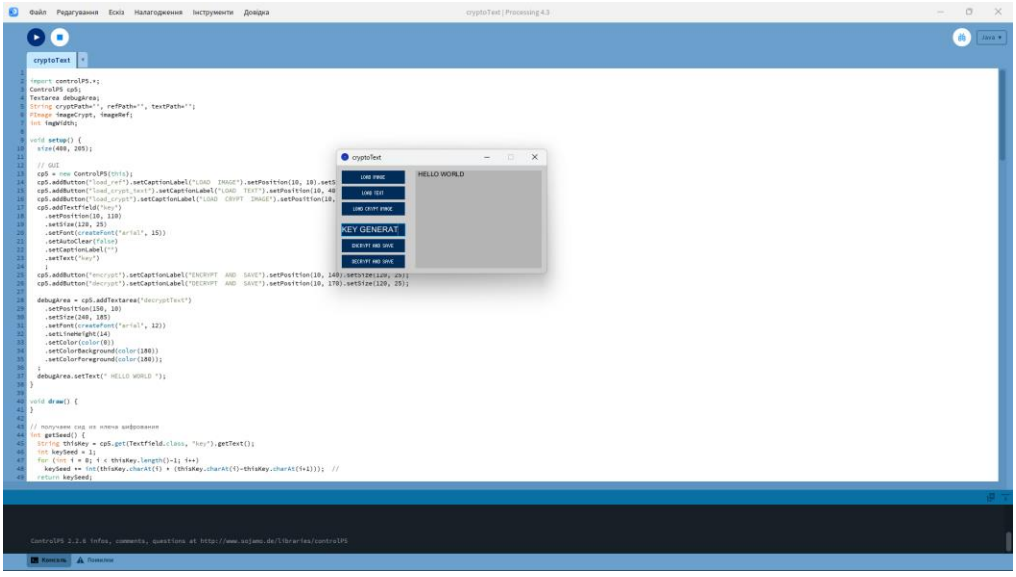

Рисунок 3.1 – Інтерфейс Користувача

2. Механізм Шифрування/Розшифрування: включає алгоритми для шифрування тексту, його вбудовування у зображення та розшифрування зворотно до первинного тексту (Рисунок 3.2).

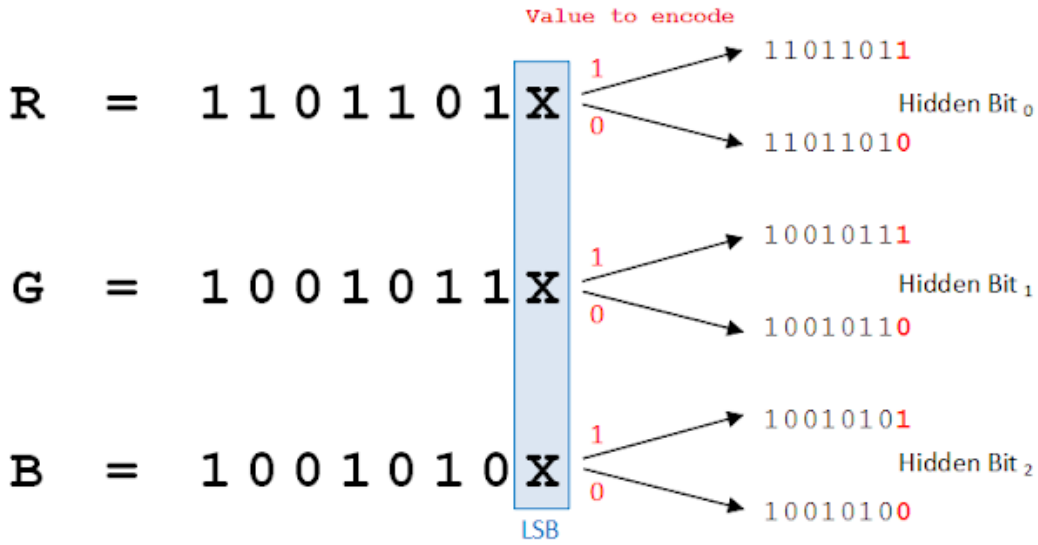

Рисунок 3.2 – LSB-стеганографія

3. Конвертація Зображень: функціонал, що дозволяє конвертувати зображення з JPEG у BMP (Рисунок 3.3). Це є необхідним для забезпечення сумісності між різними форматами цифрових файлів.

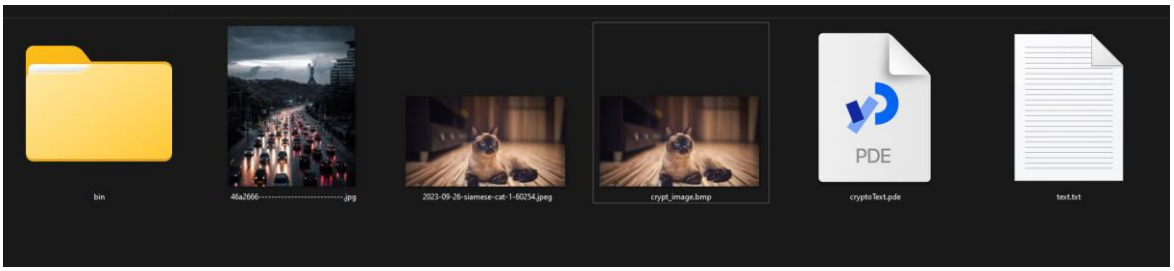

Рисунок 3.3 – Конвертація зображення

4. Генерація Ключа: У коді ключ генерується випадковим чином (Рисунок 3.4). Це забезпечує, що ключ буде унікальним і важко передбачуваним для потенційних нападників. У прикладі використовується os.urandom(16), що генерує 16-байтний (або 128-бітний) випадковий ключ. Розмір ключа визначає рівень безпеки: чим довший ключ, тим складніше його взламати.

Рисунок 3.4 – Key generation

5. Експорт та кодування тексту: цей пункт є ключовим, оскільки користувач має змогу експортувати текстову інформацію після її дешифрування (Рисунок 3.5).

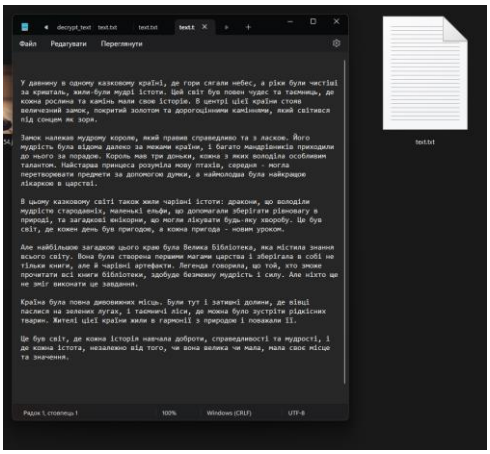

Рисунок 3.5 – txt файл з дешифрованою інформацією

3.3 Процес шифрування текстової інформації в фото

Перед початком роботи, потрібно підготувати всі потрібні матеріали для кодування, перевірки та порівняння даних. Для початку обирається фото та створюється або генерується текст, в даному випадку це звичайний Lorem ipsum, який збережений в txt файл.

За допомогою Adobe Photoshop можна переглянути вміст файлу та знайти артефакти, якщо вони з’являться при кодуванні файлу, та детального візуального порівняння файлу до і після кодування (Рисунок 3.6).

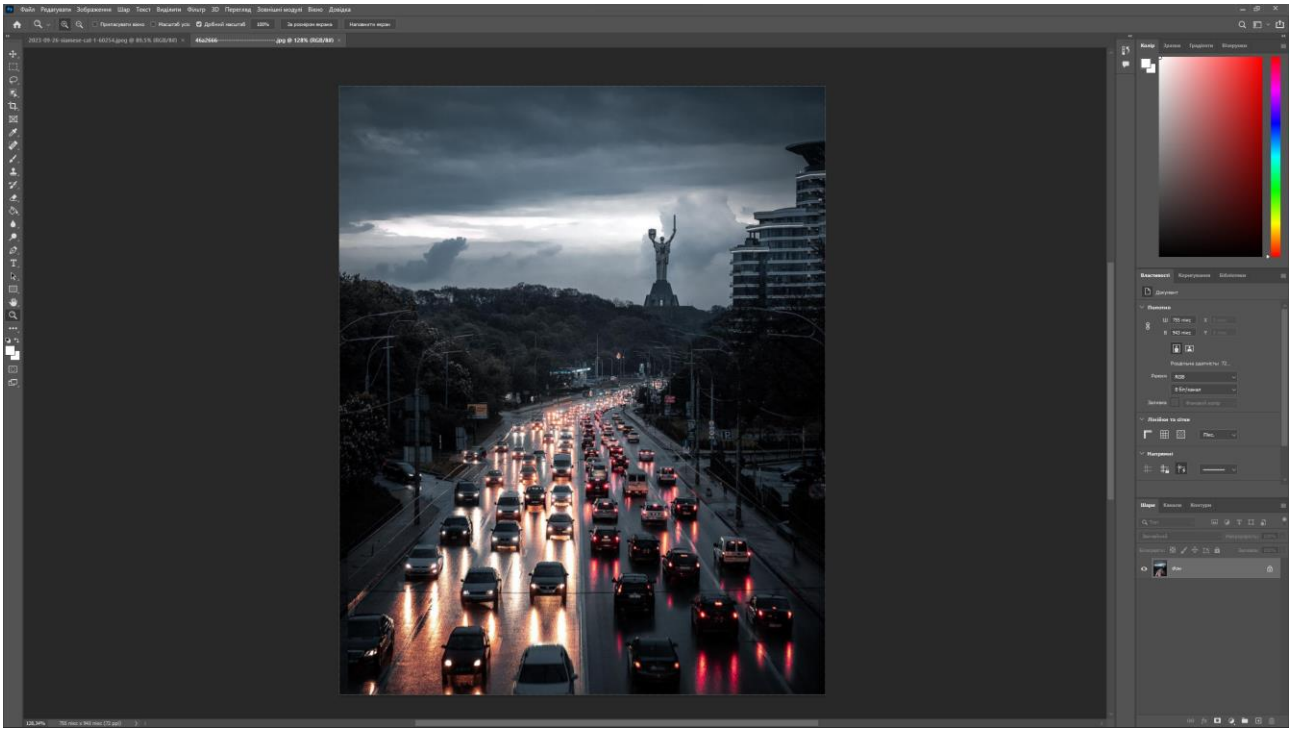

Рисунок 3.6 – вміст JPEG файлу

Для подальшої аналізу потрібно буде скористуватися XVI32, яка відображає вміст файлу у шістнадцятковому (гексадецимальному) форматі. Основні елементи того, що ми бачимо - це шістнадцяткові дані які містить

файл у шістнадцятковому форматі. Кожна пара символів представляє один байт даних. Наприклад, "FE", "D8", "E1" і так далі. В праві частині вікна відображає те саме бінарне представлення даних, але у текстовому вигляді, де це можливо. Недруковані або спеціальні символи показуються як точки або спеціальні іконки. Також ще існує адреси пам'яті. Зліва є стовпець з адресами пам'яті, які вказують на місце кожного байта у файлі. Вони також відображаються у шістнадцятковому форматі (Рисунок 3.7).

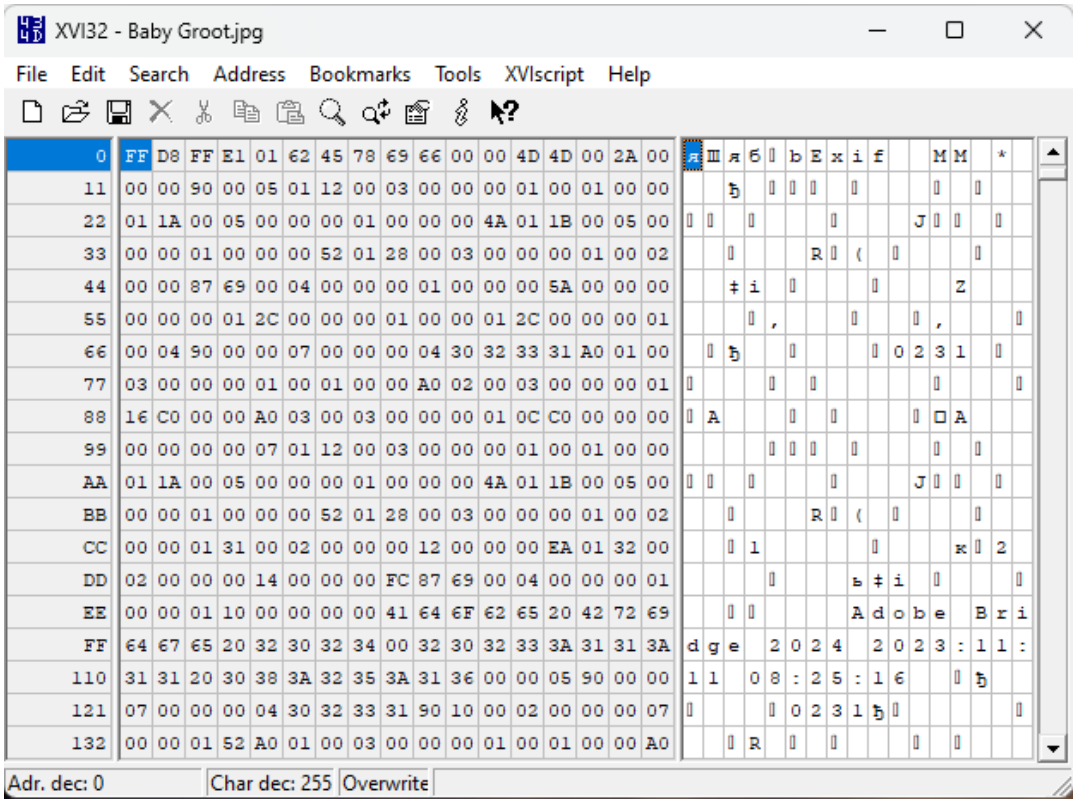

Рисунок 3.7 – Вміст JPEG файлу в програмі XVI32

На цьому зображенні з назвою "Baby Groot.jpg", яке, відкрите для аналізу у шістнадцятковому редакторі. Видимо, що файл починається з шістнадцяткових значень, які типові для JPEG файлів (наприклад, "FF D8" вказує на початок JPEG файлу). Використання шістнадцяткового редактора, як

XVI32, дозволяє користувачам редагувати або аналізувати вміст файлів на дуже низькому рівні, що може бути корисним для розробників, інженерів-програмістів, аналітиків безпеки або ентузіастів, зацікавлених у реверс-інжинірингу.

Після аналізу та підготови матеріалу потрібно запустити сам проект з кодом.

Користувач обирає JPEG-зображення для завантаження і завантажує текстовий файл з повідомленням, яке потрібно зашифрувати. Зображення конвертується в BMP формат. Після чого програма використовує алгоритми шифрування для перетворення тексту в шифр (Рисунок 3.8 – 3.9).

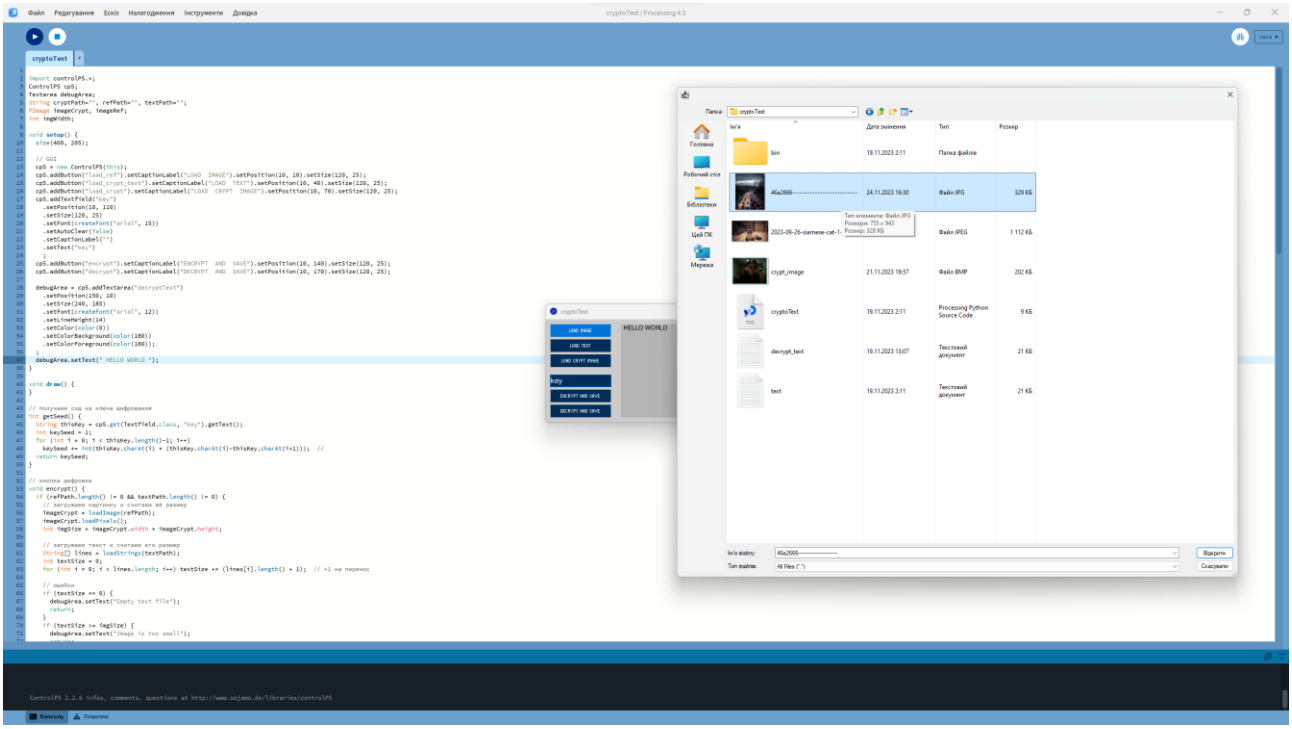

Рисунок 3.8 – Функціонал. Вибір файлу для шифрування

Шифр вбудовується в BMP-зображення. І вже вбудований у BMP-зображення шифр шифрується додатково за допомогою ключа, забезпечуючи

додатковий рівень безпеки.

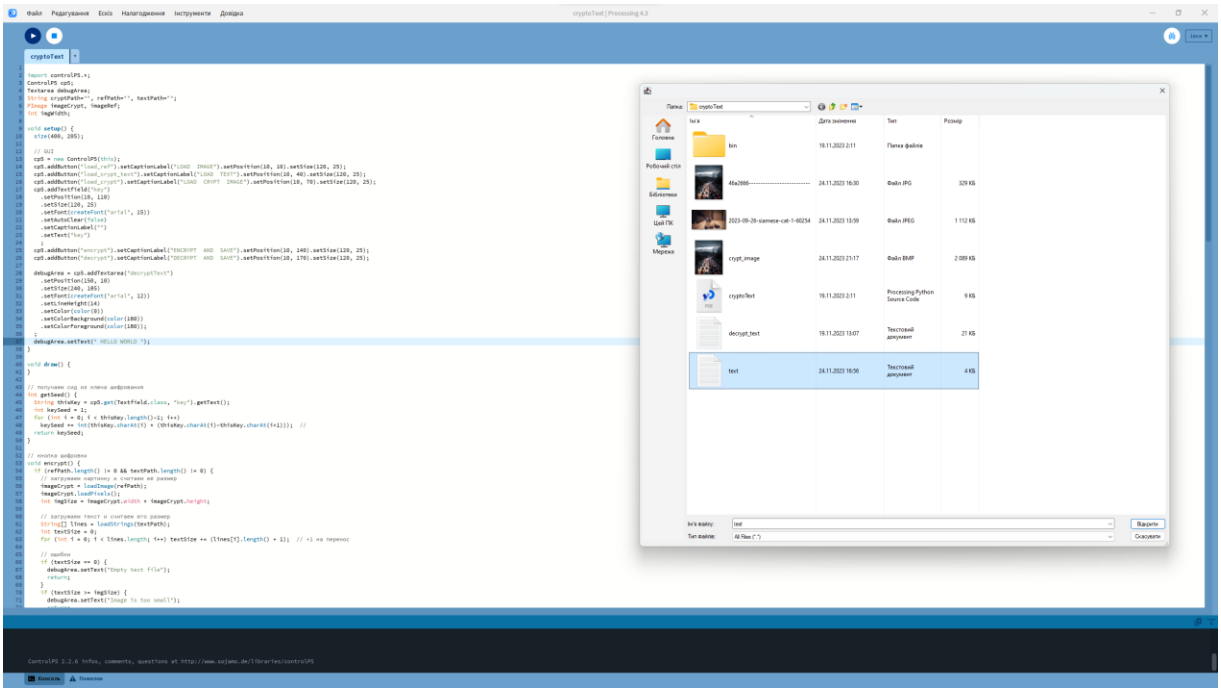

Рисунок 3.9 – Функціонал. Вибір тексту для шифрування

Програма дозволяє розшифрувати інформацію, повертаючи її до початкового стану так, що після всіх проведених дій не буде помітно що з файлом проводились якісь маніпуляції, для подальшого аналізу та порівняння (Рисунок 3.10).

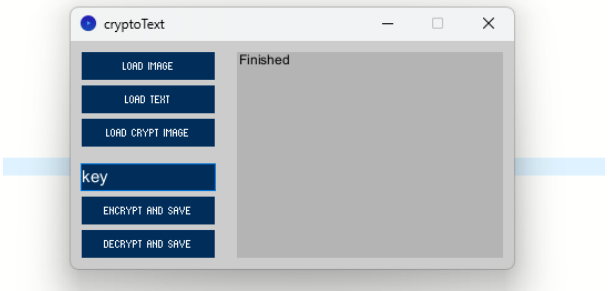

Рисунок 3.10 – Успішне шифрування файлу

3.4 Аналіз ефективності інформацйної технології

Останнім етапом роботи є порівняння оригінального зобрвження та зображеня отриманого після шифрування тексту в ньому. Порівняльний аналіз включає:

- Оцінку візуальних відмінностей між оригінальним та зашифрованим зображеннями.
- Аналіз ефективності шифрування з точки зору конфіденційності та безпеки вбудованих даних.
- Оцінку впливу шифрування на якість та розмір файлу зображення.

Ця практична реалізація демонструє ефективний підхід до захисту конфіденційної інформації, вбудовуючи її в цифрові зображення, та забезпечує комплексний аналіз ефективності використаних методів шифрування. Після всіх маніпуляцій ми маємо BMP файл з зашифрованим вмістом який візуально не відрізняється від початкового, але містить дані нові які можно побачити за допомогою XVI32 (Рисунок 3.11-3.12).

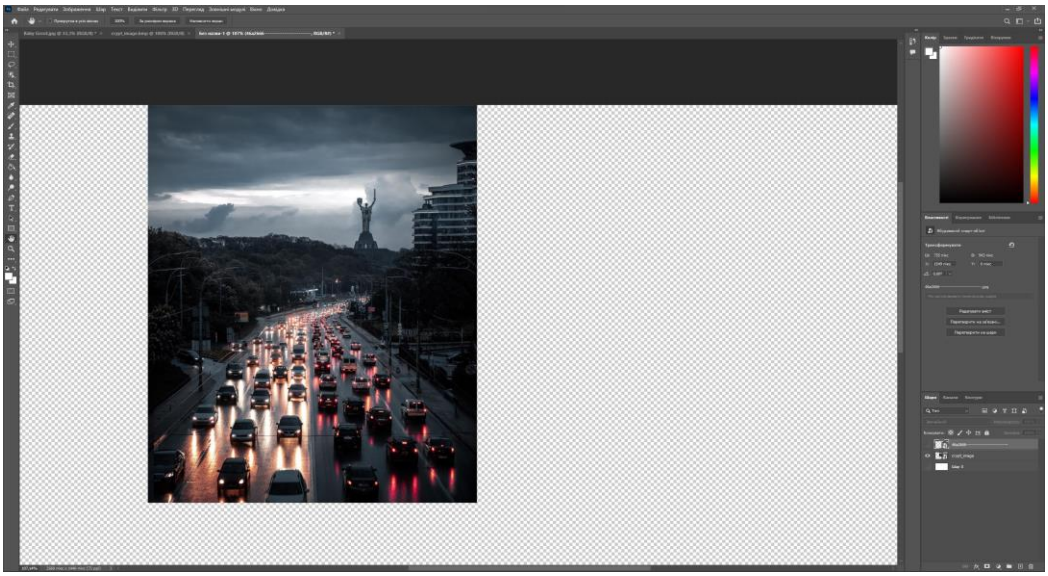

Рисунок 3.11 – Візуальний вигляд BMP файлу

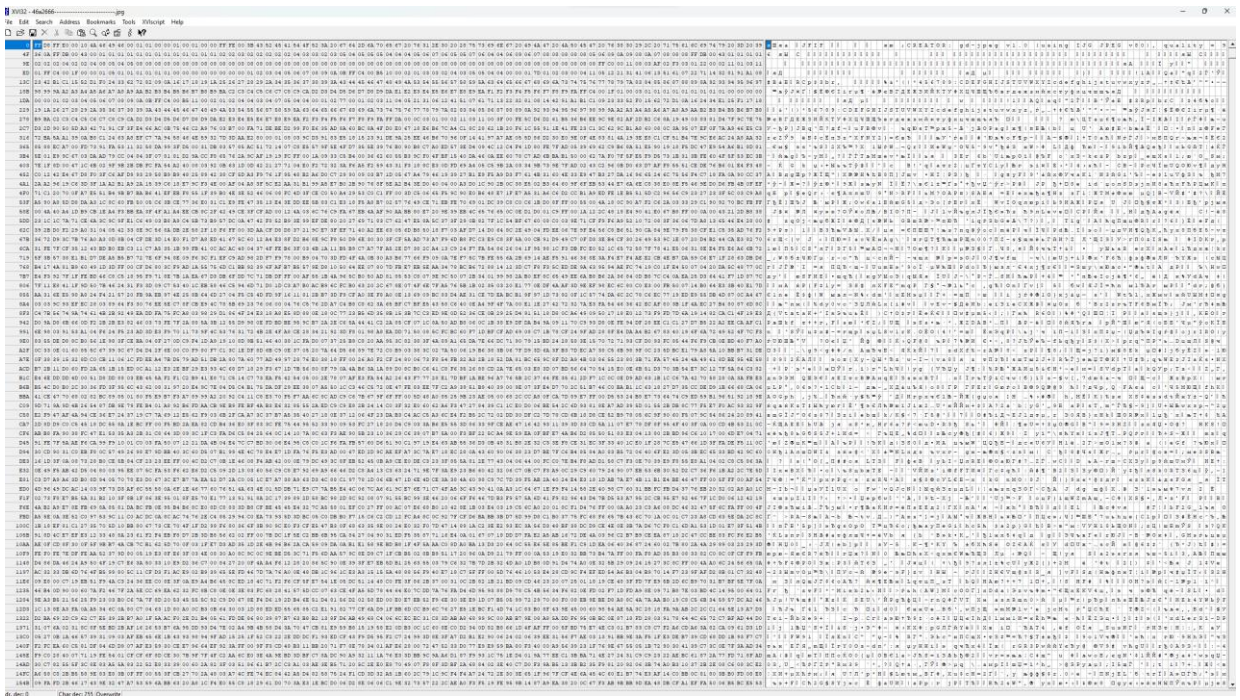

Рисунок 3.12 – Візуальний вигляд BMP в XVI32

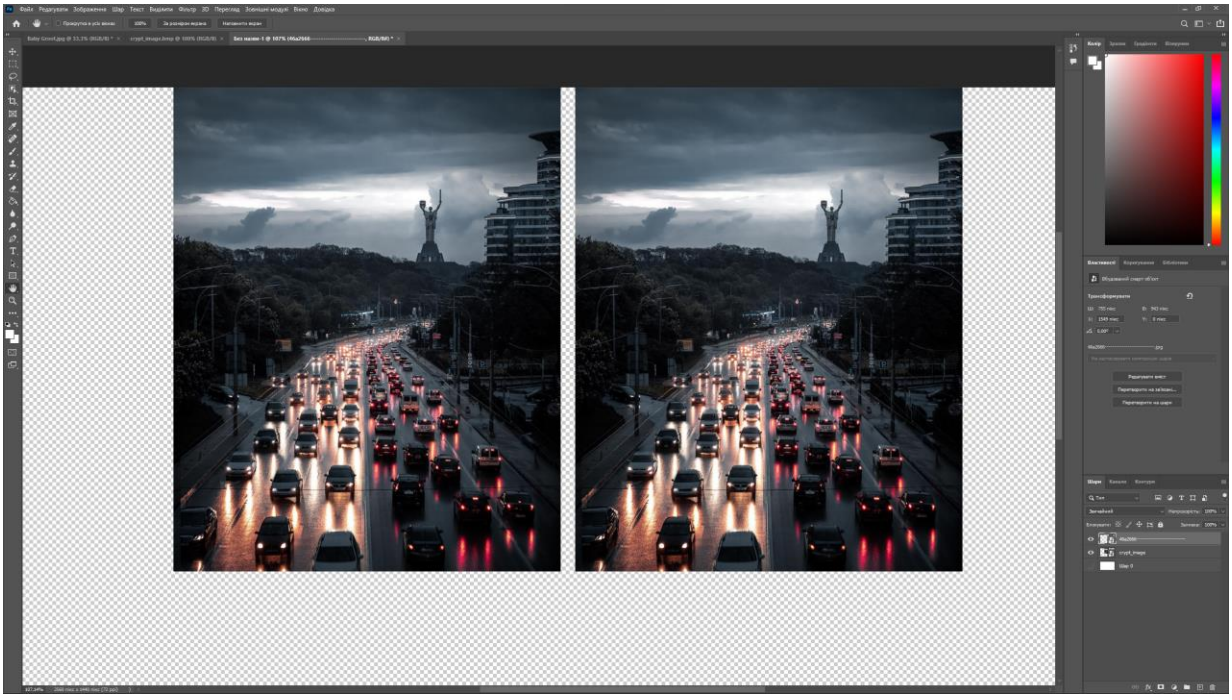

Рисунок 3.13 – Візуальне порівняння двох фото BMP та JPEG

Щоб перевірити чи відрізняються фото більш детальніше, потрібно знову завантажити дані, щоб отримати візуальний аналіз фото та поступового аналізу шуму градієнту та повторів. В кінцевому випадку маємо такий результат (Рисунок 3.13-3.14).

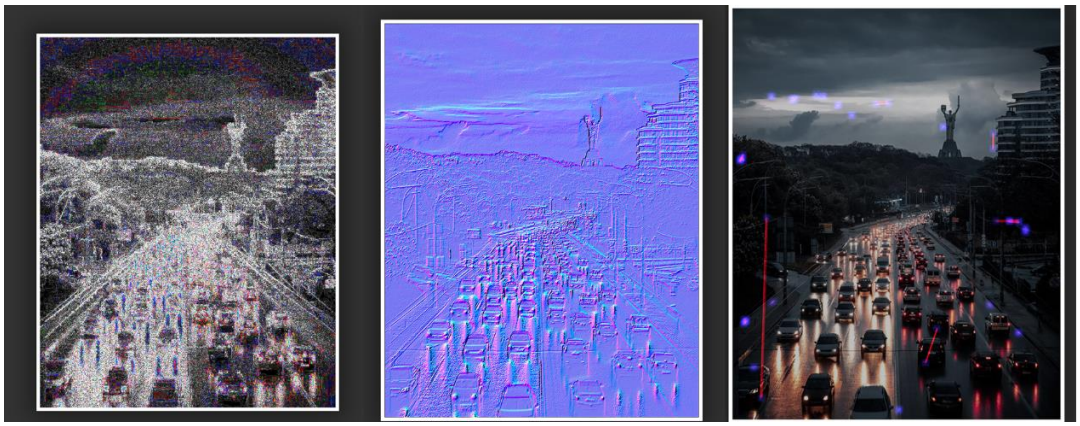

Рисунок 3.14 – Візуальне порівняння шуму градієнту та повторів BMP

Також можна власноруч перевірити на одному і тому самому участку фото чи є візуальні зміни, для цього знову потрібен Adobe Photoshop, та приблизити 1 об’єкт на 2 фото, візуально оглянути участок та порівняти (Рисунок 3.15 – 3.17) .

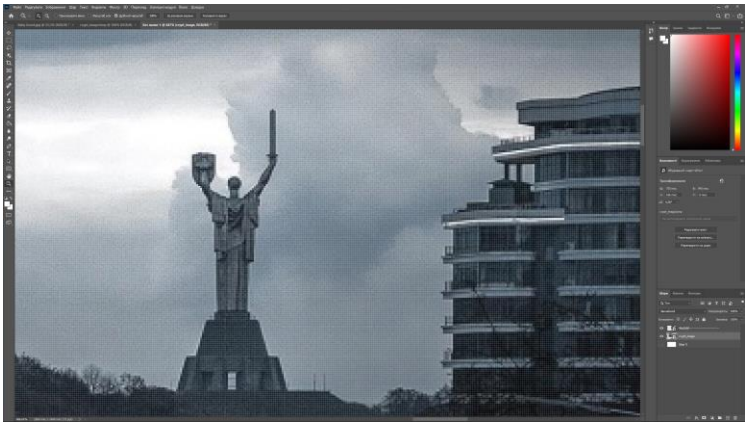

Рисунок 3.15 – BMP файл з збільшенням

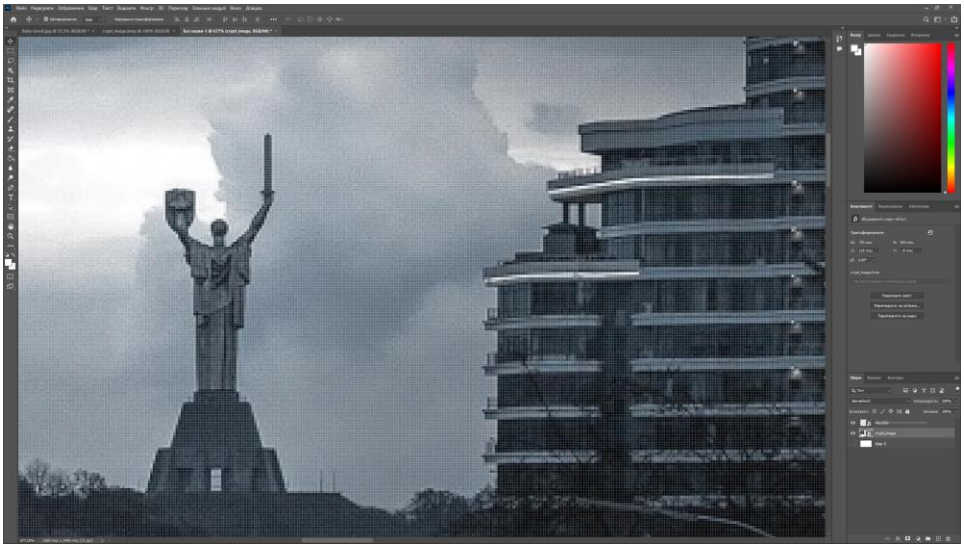

Рисунок 3.16 – JPEG файл з збільшенням

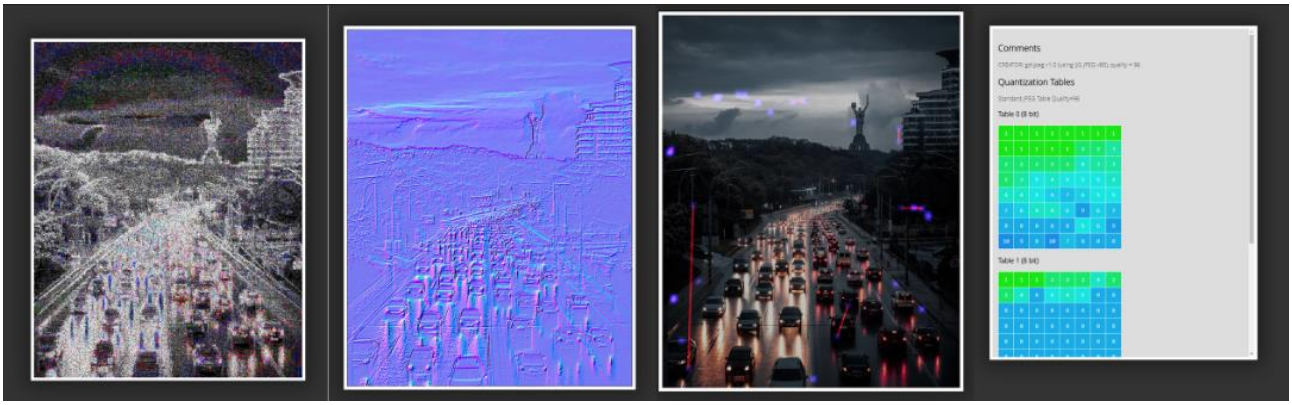

Рисунок 3.17 – Візуальне порівняння шуму градієнту та повторів JPEG

Аналіз базується на порівнянні двох зображень: одне в форматі BMP, що містить зашифровані дані, та інше в оригінальному JPEG форматі без шифрування. Шум, градієнти та повторення в зображеннях аналізуються за допомогою спеціалізованого програмного забезпечення, яке може виявити і кількісно оцінити будь-які зміни, внесені шифруванням .

Шум на зображенні може бути результатом шифрування, особливо якщо прихована інформація вбудована на низькому рівні в індивідуальні пікселі.

Висока частота шуму на BMP зображенні порівняно з JPEG може свідчити про успішне вбудовування шифрованих даних. Але візуально вони подібні і майже не відрізняються, та яскравих артефактів не знайдено.

Градієнтні карти дозволяють виявити варіації в яскравості та кольорі, які можуть вказувати на модифікації, зроблені при шифруванні. Наявність неприродних градієнтів у BMP зображенні порівняно з JPEG може вказувати на зміни в піксельних значеннях, які не очікуються в не зашифрованому зображенні. І як з артефактами різниця мінімальна.

І повторення наявність або відсутність повторюваних патернів у зашифрованому зображенні може свідчити про методи шифрування. Зашифровані дані можуть внести повторювані патерни, які не присутні в оригінальному JPEG зображенні. І все те саме візуальних дефектів не має.

Порівняльний аналіз виявив наступне:

- Зашифроване BMP зображення містить візуально мінімальні виявлені зміни, включаючи шум і змінені градієнти, що вказує на успішне шифрування.
- Не зашифроване JPEG зображення зберігає свої первинні властивості, без виявлених аномалій, що свідчить про його оригінальний стан.

Цей аналіз підкреслює ефективність розробленної інформаційної технології для шифрування та підтверджує, що розроблений алгоритм шифрування має незначний вплив на візуальні характеристики зображення, який є непомітним для неозброєного ока. Зашифроване зображення в BMP форматі може бути використане для надійного зберігання або передачі конфіденційних даних з мінімальним візуальним впливом на якість зображення.

Для декодування потрібно зробити ті самі кроки що і для кодування але в зворотному порядку, алгоритм діє в обидві сторони. Потрібно просто обрати

BMP файл та завантажити і отримаємо текст в програмі та decrypt_text.txt з аналогічним текстом (Рисунок 3.17-3.19).

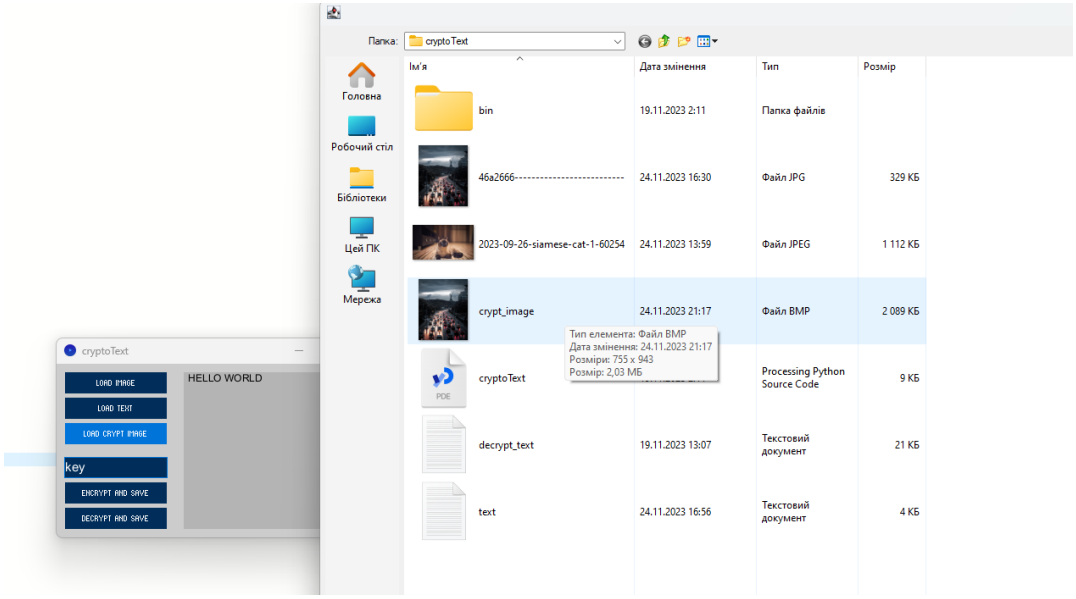

Рисунок 3.18 – Завантаження BMP для декодування

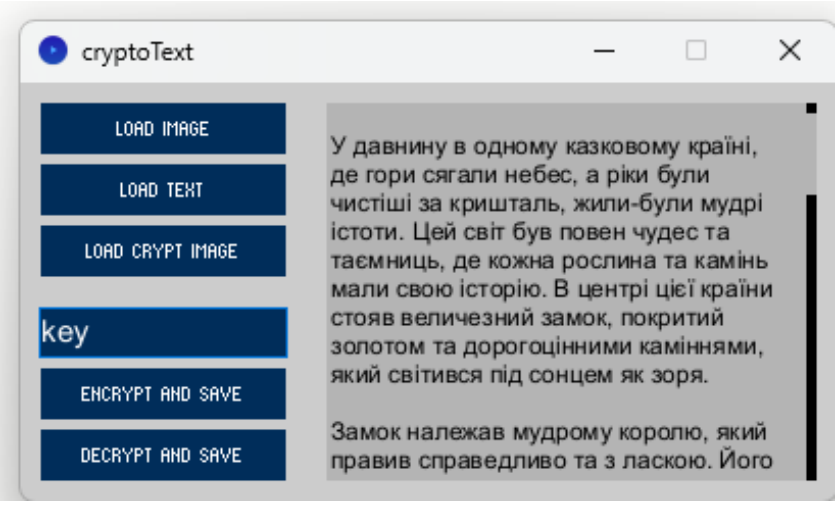

Рисунок 3.19 – Успішне декодування для BMP

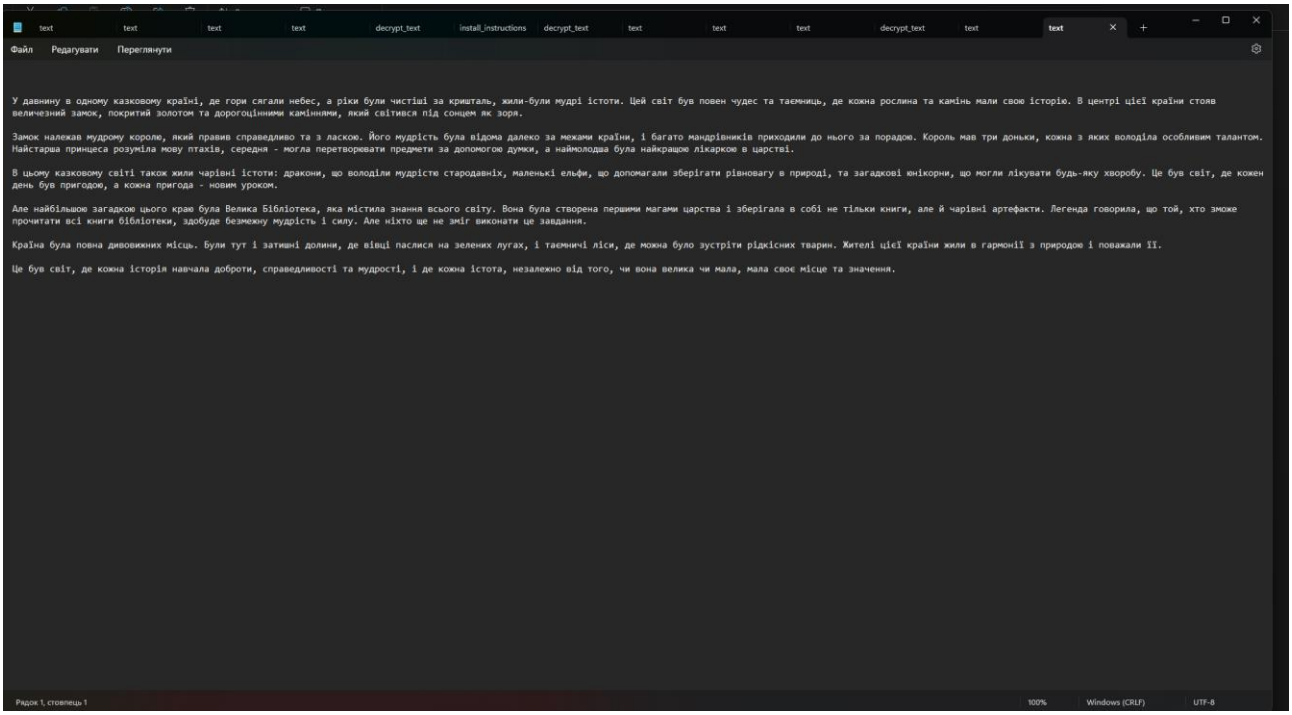

Рисунок 3.20 – Вміст decrypt_text.txt

ВИСНОВКИ

Ця магістерська робота була присвячена створеню інформаційної технології забезпечення безпеки та конфіденційності даних в цифрових зображеннях за допомогою ширування. Основною метою дослідження було створення програмного рішення, здатного ефективно вбудовувати та приховувати шифрований текст в зображеннях форматів JPEG та BMP, з подальшою можливістю його вилучення та розшифрування.

В ході роботи було досягнуто наступних ключових результатів:

1. Розробка алгоритму: було успішно розроблено алгоритм, що інтегрує сучасні криптографічні методики з методами стеганографії для вбудовування шифрованого тексту в зображення.
2. Програмна реалізація: реалізовано інформаційну технологію, яка демонструє практичне використання розробленого алгоритму. Програма забезпечує інтерфейс для введення та шифрування тексту, вбудовування його в зображення, а також для розшифрування і аналізу кінцевих результатів.
3. Аналіз ефективності: проведено аналіз ефективності алгоритму через порівняння шуму, градієнтів та повторів у зашифрованих та не зашифрованих зображеннях. Було встановлено, що алгоритм здатний виконувати шифрування з мінімальним впливом на візуальні характеристики зображення.
4. Безпека та конфіденційність: підтверджено, що розроблене рішення забезпечує високий рівень безпеки та конфіденційності, що робить його

придатним для застосування в областях, де ці аспекти є критично важливими.

5. Можливості подальшого розвитку: визначено шляхи подальшого розвитку та удосконалення алгоритму, зокрема, підвищення стійкості до стеганалітичних атак та розширення функціональності програми.

У висновку, ця магістерська робота забезпечила значний вклад у розвиток методів шифрування в цифрових зображеннях, демонструючи важливість інтеграції криптографічних та стеганографічних підходів для забезпечення безпеки даних. Результати дослідження підкреслюють потенціал застосування розробленого рішення в різних сферах, де необхідний захист інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is encryption? Data encryption defined | IBM [Electronic resource]. URL: https://www.ibm.com/topics/encryption (accessed: 28.11.2023).
2. Global Encyclopedia of Public Administration, Public Policy, and Governance // Global Encyclopedia of Public Administration, Public Policy, and Governance. Springer International Publishing, 2022.
3. Sundararajan D. Digital image processing: A signal processing and algorithmic approach // Digital Image Processing: A Signal Processing and Algorithmic Approach. Springer Singapore, 2017. P. 1–468.
4. Растрові файли| Adobe [Electronic resource]. URL: https://www.adobe.com/ua/creativecloud/file-types/image/raster.html (accessed: 28.11.2023).
5. Matted S., Shankar G., Jain B.B. Enhanced Image Security Using Stenography and Cryptography // Lecture Notes on Data Engineering and Communications Technologies. Springer Science and Business Media Deutschland GmbH, 2021. Vol. 58. P. 1171–1182.
6. Daemen J., Rijmen V. The advanced encryption standard process // Information Security and Cryptography. Springer Science and Business Media Deutschland GmbH, 2020. P. 1–8.
7. Gaur S. et al. Image Distortion Analysis in Stego Images Using LSB // Lecture Notes in Networks and Systems. Springer Science and Business Media Deutschland GmbH, 2023. Vol. 421. P. 667–679.
8. Marvel L.M. Information Hiding: Steganography and Watermarking // Optical and Digital Techniques for Information Security. Springer, New York, NY , 2005. P. 113–133.

9. A Brief Introduction to Modern Cryptography // Cryptographic Algorithms on Reconfigurable Hardware. Boston, MA: Springer, Boston, MA, 2006. P. 7–33.
10. Encyclopedia of Cryptography and Security // Encyclopedia of Cryptography and Security. Springer US, 2011.
11. Priyanka Brahmaiah V. et al. Implementation of AES Algorithm // Lecture Notes in Networks and Systems. Springer Science and Business Media Deutschland GmbH, 2023. Vol. 720 LNNS. P. 161–171.
12. Vasupongayya S. et al. Initial ASEAN digital security and interoperability standard proposal. 2023. P. 020003.
13. (PDF) Steganography in Images Using LSB Technique [Electronic resource]. URL: https://www.researchgate.net/publication/371671984_Steganography_in_Images_Using_LSB_Technique (accessed: 28.11.2023).
14. Vyas A., Yu S., Paik J. Fundamentals of digital image processing // Signals and Communication Technology. Springer Science and Business Media Deutschland GmbH, 2018. P. 3–11.
15. Taubman D.S., Marcellin M.W. JPEG2000 Image Compression Fundamentals, Standards and Practice // JPEG2000 Image Compression Fundamentals, Standards and Practice. Springer US, 2002.
16. Reinhard E., Pouli T. Colour spaces for colour transfer // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin, Heidelberg, 2011. Vol. 6626 LNCS. P. 1–15.
17. Advances in Artificial Intelligence and Data Engineering / ed. Chiplunkar N.N., Fukao T. Singapore: Springer Nature Singapore, 2021. Vol. 1133.

18. Qiu R.S., Gao Y.Q. A Windows bitmap-based numerically controlled sculpture method and its application // J Shanghai Univ. Springer, 2008. Vol. 12, № 1. P. 71–75.
19. Saeed Seyed Agha Banihashemi. Some comparison on application of cryptology algorithms // International Journal of Science and Research Archive. GSC Online Press, 2022. Vol. 5, № 1. P. 067–072.
20. Chillali S., Oughdir L. Encryption Algorithms on BMP and JPEG Images // WSEAS TRANSACTIONS ON SIGNAL PROCESSING. World Scientific and Engineering Academy and Society (WSEAS), 2022. Vol. 18. P. 21–24.
21. Stewart J.M., Mommert M. Basic Python // Python for Scientists. Cambridge University Press, 2023. P. 22–71.

ДОДАТОК А

```
import controlP5.*;
ControlP5 cp5;
Textarea debugArea;
String cryptPath="", refPath="", textPath="";
PImage imageCrypt, imageRef;
int imgWidth;

void setup() {
  size(400, 205);

  cp5 = new ControlP5(this);
  cp5.addButton("load_ref").setCaptionLabel("LOAD  IMAGE").setPosition(10,
10).setSize(120, 25);
  cp5.addButton("load_crypt_text").setCaptionLabel("LOAD
TEXT").setPosition(10, 40).setSize(120, 25);
  cp5.addButton("load_crypt").setCaptionLabel("LOAD  CRYPT
IMAGE").setPosition(10, 70).setSize(120, 25);
  cp5.addTextfield("key")
    .setPosition(10, 110)
    .setSize(120, 25)
    .setFont(createFont("arial", 15))
    .setAutoClear(false)
    .setCaptionLabel("")
    .setText("key")
    ;
```

```
  cp5.addButton("encrypt").setCaptionLabel("ENCRYPT  AND
SAVE").setPosition(10, 140).setSize(120, 25);
  cp5.addButton("decrypt").setCaptionLabel("DECRYPT  AND
SAVE").setPosition(10, 170).setSize(120, 25);

  debugArea = cp5.addTextarea("decryptText")
    .setPosition(150, 10)
    .setSize(240, 185)
    .setFont(createFont("arial", 12))
    .setLineHeight(14)
    .setColor(color(0))
    .setColorBackground(color(180))
    .setColorForeground(color(180));
  ;
  debugArea.setText("CryptoText v1.");
}

void draw() {
}

int getSeed() {
  String thisKey = cp5.get(Textfield.class, "key").getText();
  int keySeed = 1;
  for (int i = 0; i < thisKey.length()-1; i++)
    keySeed *= int(thisKey.charAt(i) * (thisKey.charAt(i)-thisKey.charAt(i+1)));  //
  return keySeed;
}
```

```
//
void encrypt() {
  if (refPath.length() != 0 && textPath.length() != 0) {
    imageCrypt = loadImage(refPath);
    imageCrypt.loadPixels();
    int imgSize = imageCrypt.width * imageCrypt.height;

    //
    String[] lines = loadStrings(textPath);
    int textSize = 0;
    for (int i = 0; i < lines.length; i++) textSize += (lines[i].length() + 1);  // +1

    //
    if (textSize == 0) {
      debugArea.setText("Empty text file");
      return;
    }
    if (textSize >= imgSize) {
      debugArea.setText("Image is too small");
      return;
    }

    lines[lines.length-1] += '\0';
    textSize += 1;

    randomSeed(getSeed());
```

```
int[] pixs = new int[textSize];  //
int counter = 0;

for (int i = 0; i < lines.length; i++) {         //
  for (int j = 0; j < lines[i].length() + 1; j++) {  //

    //
    int thisPix;
    while (true) {
      thisPix = (int)random(0, imgSize);         //
      boolean check = true;                      //
      for (int k = 0; k < counter; k++) {        //
        if (thisPix == pixs[k]) check = false;   //
      }
      if (check) {                               //
        pixs[counter] = thisPix;                 //
        counter++;                               // ++
        break;                                   //
      }
    }

    int thisChar;
    if (j == lines[i].length()) thisChar = int('\n');  //
    else thisChar = lines[i].charAt(j);        //

    if (thisChar > 1000) thisChar -= 890;    //
```

```
      int thisColor = imageCrypt.pixels[thisPix];  //
      int newColor = (thisColor & 0xF80000);   // 11111000 00000000 00000000
      newColor |= (thisChar & 0xE0) << 11;     // 00000111 00000000 00000000
      newColor |= (thisColor & (0x3F << 10));  // 00000000 11111100 00000000
      newColor |= (thisChar & 0x18) << 5;      // 00000000 00000011 00000000
      newColor |= (thisColor & (0x1F << 3));   // 00000000 00000000 11111000
      newColor |= (thisChar & 0x7);            // 00000000 00000000 00000111

      imageCrypt.pixels[thisPix] = newColor;   //
    }
   }
   imageCrypt.updatePixels();                 //
   imageCrypt.save("crypt_image.bmp");        //
   debugArea.setText("Finished");
  } else debugArea.setText("Image is not selected");
}

void decrypt() {
  if (cryptPath.length() != 0) {
   imageCrypt = loadImage(cryptPath);
   imageCrypt.loadPixels();
   int imgSize = imageCrypt.width * imageCrypt.height;

   randomSeed(getSeed());

   int[] pixs = new int[imgSize];  //
```

```
String decryptText = "";        //
int counter = 0;

//
while (true) {

 //
 int thisPix;
 while (true) {
  thisPix = (int)random(0, imgSize);
  boolean check = true;
  for (int k = 0; k < counter; k++) {
   if (thisPix == pixs[k]) check = false;
  }
  if (check) {
   pixs[counter] = thisPix;
   counter++;
   break;
  }
 }

 int thisColor = imageCrypt.pixels[thisPix];

 int thisChar = 0;
 thisChar |= (thisColor & 0x70000) >> 11;  // 00000111 00000000 00000000 ->
00000000 00000000 11100000
```

```
      thisChar |= (thisColor & 0x300) >> 5;     // 00000000 00000011 00000000 ->
00000000 00000000 00011000
      thisChar |= (thisColor & 0x7);           // 00000000 00000000 00000111

      if (thisChar > 130) thisChar += 890;
      if (thisChar == 0) break;
      decryptText += char(thisChar);
    }
    debugArea.setText(decryptText);

    String[] lines = new String[1];
    lines[0] = decryptText;
    saveStrings("decrypt_text.txt", lines);
  } else debugArea.setText("Crypted image is not selected");
}

void load_ref() {
  selectInput("", "selectRef");
}

void selectRef(File selection) {
  if (selection != null) {
    refPath = selection.getAbsolutePath();
    debugArea.setText(refPath);
  } else debugArea.setText("Image is not selected");
}
```

```
void load_crypt() {
  selectInput("", "selectCrypt");
}

void selectCrypt(File selection) {
  if (selection != null) {
    cryptPath = selection.getAbsolutePath();
    debugArea.setText(cryptPath);
  } else debugArea.setText("Crypted image is not selected");
}

void load_crypt_text() {
  selectInput("", "selectCryptText");
}

void selectCryptText(File selection) {
  if (selection != null) {
    textPath = selection.getAbsolutePath();
    debugArea.setText(textPath);
  } else debugArea.setText("Text file is not selected");
}
```