МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

Дипломна робота

Застосування криптографічних алгоритмів до шифрування та дешифрування зображень

Виконав: студент групи ПМП-42
спеціальності
113 - прикладна математика

_____ Стадник Р.І._____

Керівник _____ Стягар А.О._____

Рецензент_____
(прізвище та ініціали)

Львів - 2023

Зміст

Вступ

З появою сучасних технологій обробки та передачі зображень, проблема безпеки стала однією з найважливіших у галузі інформаційної безпеки. Зображення містять значну кількість важливої інформації, яка може бути конфіденційною або потребує забезпечення цілісності. Таким чином, захист конфіденційності та цілісності зображень стає необхідним завданням у різних сферах, включаючи медицину, фінанси, правоохоронні органи та багато інших.

Проблема безпеки зображень полягає в тому, що зображення можуть бути піддані несанкціонованому доступу, модифікації або перехопленню під час передачі по мережі. Це може призвести до розголошення конфіденційної інформації, порушення приватності або спотворення зображень, що може мати серйозні наслідки в різних сферах життя.

Тому, забезпечення безпеки зображень є актуальним завданням, яке потребує використання криптографічних алгоритмів та захисних механізмів. Криптографічні алгоритми дозволяють зашифрувати зображення з метою збереження конфіденційності та цілісності, а також дешифрувати їх для отримання початкового зображення. Важливою задачею є розробка ефективних і безпечних методів шифрування та дешифрування зображень, а також оцінка їх продуктивності та безпеки.

У даній дипломній роботі я буду досліджувати та реалізовувати систему шифрування та дешифрування зображень, використовуючи сучасні криптографічні алгоритми.

Розділ 1
Огляд криптографічних алгоритмів

1.1. Симетричне шифрування

Симетричне шифрування використовує один і той самий ключ як для шифрування, так і для дешифрування даних. Це означає, що той, хто має доступ до ключа, може виконувати обидві операції. Одним з найпопулярніших симетричних алгоритмів є AES (Advanced Encryption Standard). Він забезпечує високу стійкість і швидкість обробки даних, що робить його ефективним для шифрування зображень.

Рисунок 1.1 – Схема шифрування симетричним алгоритмом

Переваги симетричного шифрування:

- Висока швидкість шифрування та дешифрування.
- Простота реалізації алгоритмів.
- Ефективність для обробки великих обсягів даних.

Недоліки симетричного шифрування:

- Необхідність безпечного обміну ключами між відправником і одержувачем.
- Однаковий ключ використовується для всіх операцій шифрування, що може створювати питання щодо конфіденційності.

1.2. Асиметричне шифрування

Асиметричне шифрування використовує два різних ключі: публічний ключ для шифрування та приватний ключ для дешифрування. Публічний ключ розповсюджується, тоді як приватний ключ залишається секретним. Алгоритм RSA є одним з найпоширеніших алгоритмів асиметричного шифрування. Використання асиметричного шифрування для зображень може бути обмеженим через великі розміри даних.

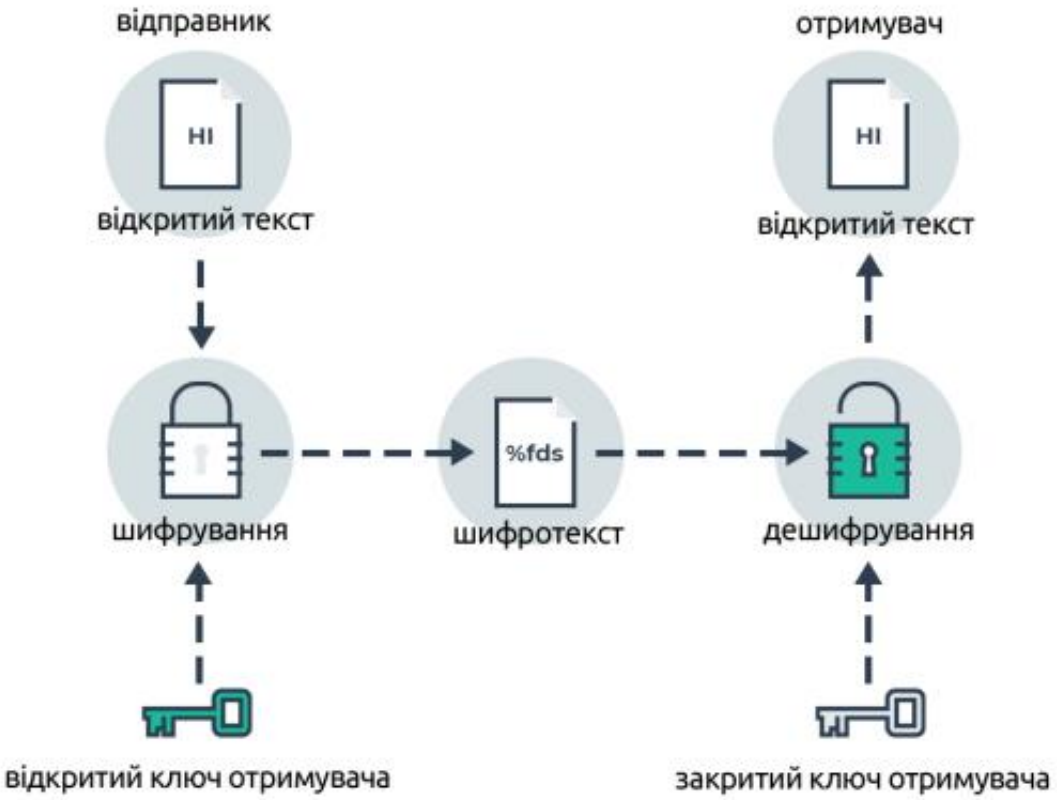

Рисунок 2.2 – Схема асиметричного алгоритму

Переваги асиметричного шифрування:

- Безпека: приватний ключ зберігається у власника і не передається.
- Можливість шифрування без необхідності обміну ключами.

Недоліки асиметричного шифрування:

- Повільна швидкість обробки даних порівняно з симетричним шифруванням.
- Обмежена продуктивність для великих обсягів даних.

1.3. Порівняння криптографічних алгоритмів щодо шифрування та дешифрування зображень

При виборі криптографічного алгоритму для шифрування та дешифрування зображень необхідно враховувати наступні фактори:

- Рівень безпеки: деякі алгоритми можуть мати більшу стійкість до криптоаналізу та злому.
- Швидкість обробки: ефективність алгоритму у відношенні до швидкості шифрування та дешифрування.
- Обсяг даних: деякі алгоритми можуть бути ефективнішими для обробки великих зображень, тоді як інші можуть бути обмежені.

Алгоритми, такі як AES, зазвичай є найпопулярнішими виборами для шифрування зображень, оскільки вони поєднують високу стійкість і швидкість обробки. Однак, вибір конкретного алгоритму залежить від конкретних вимог до безпеки, швидкості та обсягу даних.

Розділ 2
Методи шифрування зображень

2.1. Огляд методів шифрування зображень на основі криптографічних алгоритмів

Методи шифрування зображень на основі криптографічних алгоритмів використовують принципи симетричного або асиметричного шифрування для забезпечення конфіденційності та безпеки зображень. Ці методи можуть використовувати алгоритми, такі як AES, RSA, ECC та інші, для шифрування та дешифрування зображень.

Дослідження алгоритмів AES, RSA та ECC щодо їх застосування до шифрування зображень:

- AES (Advanced Encryption Standard): Цей симетричний алгоритм шифрування є одним з найбільш популярних і ефективних для застосування до шифрування зображень. Він забезпечує високий рівень безпеки та швидкості обробки даних.
- RSA (Rivest-Shamir-Adleman): Цей асиметричний алгоритм шифрування використовує публічний та приватний ключі для шифрування та дешифрування. RSA може бути використаний для шифрування ключів AES, які в свою чергу використовуються для шифрування зображень.
- ECC (Elliptic Curve Cryptography): Цей асиметричний алгоритм шифрування базується на використанні еліптичних кривих. Він відомий своєю ефективністю та надійністю, забезпечуючи високий рівень безпеки при меньших обчислювальних витратах.

2.2. Аналіз та порівняння ефективності різних методів шифрування зображень

Аналіз та порівняння ефективності різних методів шифрування зображень включає такі фактори:

- Швидкість обробки: Вимірювання часу, необхідного для шифрування та дешифрування зображень. Швидкість обробки є важливим фактором, особливо при шифруванні великих обсягів даних або в ситуаціях, де потрібно забезпечити миттєву відповідь.
- Збереження якості зображень: Оцінка здатності алгоритму шифрування зберегти якість зображень. Деякі методи шифрування можуть призводити до втрати якості зображення або артефактів на засобах зображення. Важливо вибрати метод, який забезпечує максимально можливу якість зображень після дешифрування.
- Рівень безпеки: Оцінка рівня безпеки, який забезпечується кожним методом шифрування. Це включає в себе опірність до атак злому шифру, здатність стійкого збереження ключів та можливість запобігання несанкціонованому доступу до зображень.
- Розмір ключа: Розмір ключа, який використовується для шифрування зображень, також важливий фактор. Методи шифрування з більшими розмірами ключів, як правило, забезпечують більшу стійкість, але можуть вимагати більшої обчислювальної потужності та ресурсів.

Аналіз та порівняння цих факторів допоможе визначити найбільш підходящий метод шифрування зображень залежно від вимог щодо швидкості, якості зображень та безпеки.

Розділ 3
Методи дешифрування зображень

Методи дешифрування зображень використовуються для відновлення оригінальних зображень з зашифрованих даних. Основні методи дешифрування зображень включають:

1. Симетричне дешифрування: Цей метод використовує один і той самий ключ для шифрування та дешифрування зображень. За допомогою симетричних алгоритмів, таких як AES (Advanced Encryption Standard), здійснюється розшифрування зашифрованого зображення шляхом застосування оберненої операції до шифрованого тексту.
2. Асиметричне дешифрування: Цей метод використовує пару ключів - публічний ключ для шифрування та приватний ключ для дешифрування. За допомогою асиметричних алгоритмів, таких як RSA (Rivest-Shamir-Adleman) або ECC (Elliptic Curve Cryptography), здійснюється дешифрування зашифрованого зображення шляхом використання приватного ключа.
3. Комбіновані методи: Іноді використовуються комбіновані методи, які поєднують симетричне та асиметричне шифрування для дешифрування зображень. Наприклад, може бути застосовано симетричне шифрування для шифрування самого зображення, а потім асиметричне шифрування для зашифрування ключа, який використовувався для шифрування зображення.

При дешифруванні зображень важливо враховувати безпеку та ефективність алгоритмів. Потрібно використовувати надійні ключі, забезпечувати відповідний рівень захисту та виконувати оптимальні алгоритми дешифрування для досягнення якісного та швидкого відновлення оригінальних зображень.

Крім основних методів дешифрування, існують деякі спеціалізовані техніки та алгоритми, які можуть бути використані для відновлення зашифрованих зображень. Ось кілька додаткових методів дешифрування зображень:

4. Атаки на шифротекст: Існують різні види атак на шифротекст, такі як статистичні атаки, диференціальні атаки, лінійні атаки тощо. Ці атаки базуються на аналізі шаблонів або властивостей шифротексту для виявлення слабкостей або знаходження ключа шифрування. Використання таких атак може сприяти розшифруванню зашифрованого зображення.
5. Криптоаналіз: Це процес аналізу криптографічного алгоритму з метою знаходження слабкостей або вразливостей, які можуть бути використані для дешифрування зображень без належного ключа. Криптоаналіз може включати методи, такі як аналіз частоти, аналіз кореляції, диференціальний криптоаналіз та інші.
6. Брутфорс: Цей метод полягає в спробі відновити ключ шифрування шляхом перебору всіх можливих комбінацій. Брутфорс може бути використаний в ситуаціях, коли ключ шифрування є досить коротким або має низьку ентропію. Проте, цей метод вимагає значних обчислювальних ресурсів та тривалого часу, особливо при використанні сильних шифрів.

Вибір методу дешифрування залежить від конкретного шифрувального алгоритму, використаного для зашифрування зображення, та доступності необхідних ключів чи слабкостей алгоритму. Розуміння та застосування різних методів дешифрування може допомогти відновити зашифровані зображення та забезпечити їхню цілісність та доступність.

Розділ 4

Реалізація та тестування системи шифрування та дешифрування зображень

Я використав мову програмування **Python** та бібліотеку **Tkinter** для створення графічного інтерфейсу користувача. Код реалізації системи містить наступні методи:

- **encrypt_image_ecb(image_path, key)**: Метод для шифрування зображення в режимі ECB. Він створює резервну копію оригінального зображення, відкриває файл зображення, застосовує шифрування AES у режимі ECB з заданим ключем та зберігає зашифровані дані у файлі.

- **decrypt_image_ecb(image_path, key)**: Метод для дешифрування зображення, зашифрованого в режимі ECB. Він створює резервну копію зашифрованого зображення, відкриває файл зображення, застосовує дешифрування AES у режимі ECB з заданим ключем та зберігає дешифровані дані у файлі.

- **encrypt_image_cbc(image_path, key)**: Метод для шифрування зображення в режимі CBC. Він створює резервну копію оригінального зображення, відкриває файл зображення, генерує випадковий вектор ініціалізації (IV), застосовує шифрування AES у режимі CBC з заданим ключем та зберігає IV та зашифровані дані у файлі.

- **decrypt_image_cbc(image_path, key)**: Метод для дешифрування зображення, зашифрованого в режимі CBC. Він створює резервну копію зашифрованого зображення, відкриває файл зображення, зчитує IV та зашифровані дані, застосовує дешифрування AES у режимі CBC з заданим ключем та зберігає дешифровані дані у файлі.

- **display_error_message(message)**: Метод для відображення повідомлення про помилку у вікні повідомлення.

- **open_file_dialog()**: Метод для відкриття діалогового вікна вибору файлу та повернення обраного шляху до файлу.

- Методи **encrypt_button_click_ecb()**, **decrypt_button_click_ecb()**, **encrypt_button_click_cbc()**, **decrypt_button_click_cbc()** відповідають за обробку подій натискання кнопок у графічному інтерфейсі.

Режими шифрування використані в програмі:

- ECB (Electronic Codebook) - це один з простих режимів шифрування блоками, який використовується для шифрування повідомлень в криптографії. У цьому режимі повідомлення розбивається на блоки фіксованого розміру, а кожен блок шифрується незалежно один від одного за допомогою одного і того ж ключа шифрування.

Процес шифрування ECB включає наступні кроки:

1. Розбивка повідомлення: Повідомлення, яке потрібно зашифрувати, розбивається на блоки фіксованого розміру. Якщо останній блок не заповнений повністю, його можна доповнити деякими значеннями (наприклад, нулями) до необхідного розміру.
2. Шифрування блоків: Кожен блок повідомлення шифрується окремо незалежно один від одного. Для шифрування використовується один і той же ключ шифрування. Шифрування блоку здійснюється за допомогою шифру блочного шифрування, такого як AES (Advanced Encryption Standard) або DES (Data Encryption Standard). Кожен блок обробляється індивідуально, без урахування інших блоків.
3. Конкатенація блоків: Після того, як всі блоки повідомлення зашифровані, шифротекст утворюється шляхом конкатенації зашифрованих блоків. Тобто блоки докладаються один до одного без будь-яких змін або перестановок.

Дешифрування процесу ECB включає зворотні кроки:

1. Розбивка шифротексту: Шифротекст розбивається на блоки фіксованого розміру, який відповідає розміру блоків, використаних під час шифрування.
2. Дешифрування блоків: Кожен блок шифротексту дешифрується незалежно від інших блоків за допомогою того самого ключа шифрування. Дешифрування блоку виконується з використанням розшифрування шифру блочного шифрування, який відповідає використовуваному шифруванню.

3. Конкатенація блоків: Після дешифрування кожного блоку отримуються розшифровані блоки. Розшифровані блоки конкатенуються, щоб отримати повідомлення, яке було зашифровано в оригінальному процесі шифрування.

Важливо зазначити, що ECB не є надійним режимом шифрування для деяких застосувань. Однак, він все ще використовується в деяких ситуаціях, наприклад, коли блоки повідомлення не мають залежностей між собою, і необхідне просте і швидке шифрування.

- CBC (Cipher Block Chaining) - це режим шифрування блоків, що використовується в криптографії. У режимі CBC кожен блок повідомлення комбінується з попереднім зашифрованим блоком перед шифруванням, створюючи ланцюжок блоків. Це забезпечує додатковий рівень безпеки та запобігає розпізнаванню шаблонів в шифротексті.

Процес шифрування CBC включає наступні кроки:

1. Ініціалізація вектора ініціалізації (IV): Випадкове значення, відоме як вектор ініціалізації (IV), обирається перед початком шифрування. IV використовується для перших блоків і вводиться в алгоритм шифрування разом з першим блоком повідомлення.
2. Об'єднання з IV: Перший блок повідомлення комбінується з IV за допомогою операції XOR (виключне або). Результат цієї операції стає вхідним для шифрування.
3. Шифрування блоків: Зашифрований результат першого блоку, разом з наступними блоками повідомлення, використовується для шифрування

наступних блоків. Кожен блок шифрується окремо за допомогою шифру блочного шифрування, такого як AES або DES, використовуючи один і той самий ключ шифрування.

4. Операція XOR: Зашифрований блок XOR-ується з наступним блоком повідомлення перед шифруванням наступного блоку. Це дозволяє створити залежність між блоками шифротексту.
5. Конкатенація блоків: Після шифрування всіх блоків повідомлення отримується шифротекст, який складається з зашифрованих блоків, утворених у ланцюжку.

Дешифрування процесу CBC включає зворотні кроки:

1. Розбиття шифротексту: Шифротекст розбивається на блоки фіксованого розміру, який відповідає розміру блоків, використованих під час шифрування.
2. Дешифрування блоків: Кожен блок шифротексту дешифрується окремо з використанням шифру блочного шифрування і того самого ключа шифрування.
3. Операція XOR: Результат дешифрування кожного блоку XOR-ується з попереднім зашифрованим блоком, створюючи вихідний блок повідомлення.
4. Конкатенація блоків: Розшифровані блоки повідомлення конкатенуються, щоб отримати вихідне повідомлення.

Важливо пам'ятати, що для безпеки використовуваного режиму CBC необхідно враховувати дотримання правильного управління вектором ініціалізації (IV) та ключа шифрування.

Для перевірки правильності роботи системи шифрування та дешифрування зображень ми виконуємо наступні кроки:

1. Запускаємо програму та вибираємо режим шифрування (ECB або CBC) через графічний інтерфейс. (рис. 4.1)

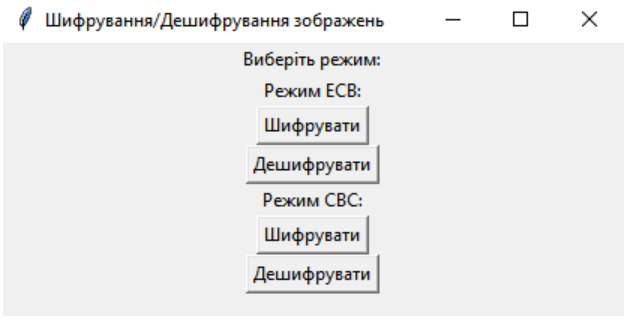

Рисунок 4.1 – інтерфейс програми

2. Обираємо зображення, яке потрібно зашифрувати, за допомогою вікна вибору файлу. (рис. 4.2)

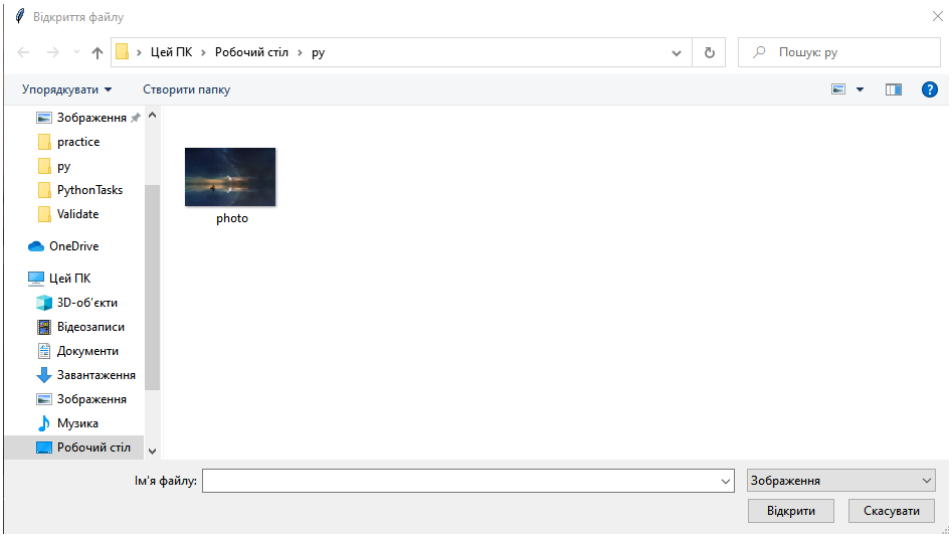

Рисунок 4.2 – вікно вибору файла

3. Виконуємо шифрування та чекаємо повідомлення про успішне шифрування яке відображається у вікні повідомлення. (рис. 4.3)

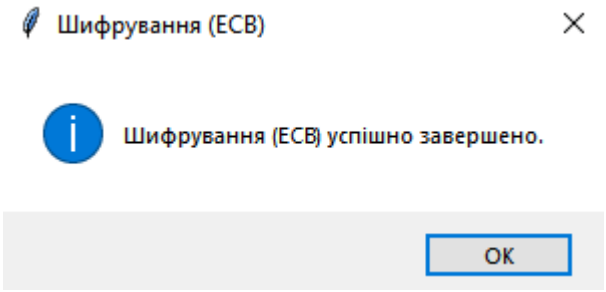

Рисунок 4.3 – повідомлення про успішне шифрування

4. Після шифрування перевіряємо фото (рис. 4.4). Щоб повернути фото потрібно виконати дешифрування.

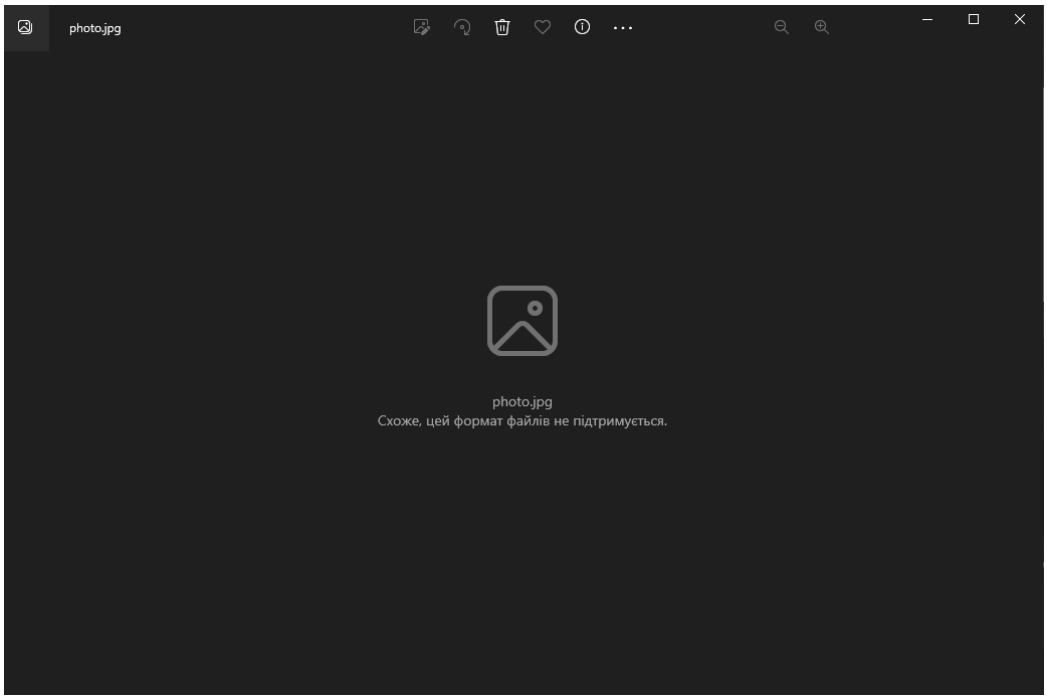

Рисунок 4.4 – фото

5. Повторюємо кроки 2-4 для дешифрування зображення.
6. Повідомлення про успішне дешифрування відображається у вікні повідомлення.

7. Перевіряємо, що оригінальне зображення і зображення після дешифрування співпадають. (рис. 4.5)

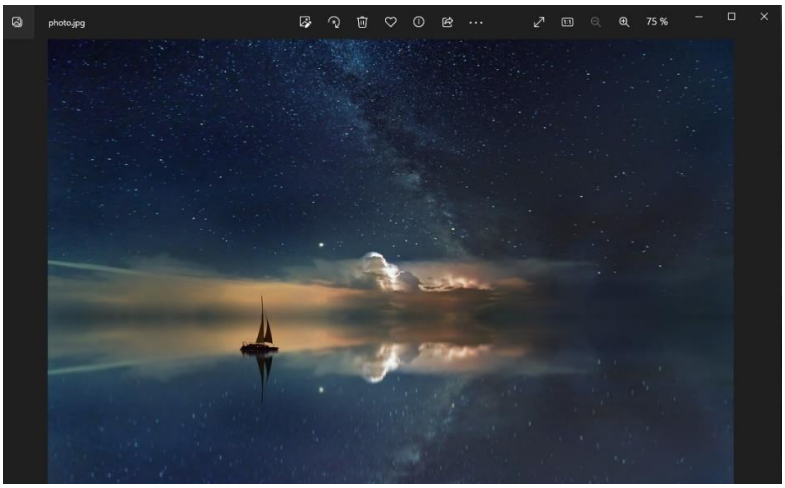

Рисунок 4.5 – фото

Висновок

В даній дипломній роботі я розглянув застосування криптографічних алгоритмів до шифрування та дешифрування зображень з метою забезпечення безпеки, конфіденційності та цілісності даних. Проблема безпеки при обробці та передачі зображень визначена як важлива, оскільки зображення можуть містити чутливу інформацію, яку необхідно захистити від несанкціонованого доступу.

У першому розділі я надав огляд криптографічних алгоритмів, зокрема симетричного та асиметричного шифрування. Також розглянув переваги та недоліки різних криптографічних алгоритмів і провів аналіз і порівняння цих алгоритмів щодо їх застосування до шифрування та дешифрування зображень.

У другому розділі дослідив методи шифрування зображень на основі криптографічних алгоритмів, зокрема алгоритми AES, RSA та ECC. Проаналізував ефективність різних методів шифрування зображень і провів порівняння їх.

У третьому розділі розглянув методи дешифрування зображень за допомогою криптографічних алгоритмів і провів експериментальне порівняння розроблених алгоритмів дешифрування.

У четвертому розділі я реалізував та протестував систему шифрування та дешифрування зображень з використанням режимів ECB та CBC. Розробив графічний інтерфейс, який дозволяє користувачам зручно шифрувати та дешифрувати зображення обраними алгоритмами.

Отже, на основі проведених досліджень та експериментів я зробив висновок, що застосування криптографічних алгоритмів до шифрування та дешифрування зображень є ефективним способом забезпечення безпеки та конфіденційності зображень. Розроблена система шифрування та дешифрування зображень може бути використана для захисту важливої інформації, що міститься в зображеннях, від несанкціонованого доступу.

Список використаних джерел

1. Boneh, D., Shoup, V. (2000). A Graduate Course in Applied Cryptography. https://crypto.stanford.edu/~dabo/cryptobook/
2. Information Security: Principles and Practice 2nd Edition by Mark Stamp. information-security-principles-and-practice-2nd-edition-stamp.pdf (wordpress.com)
3. Кулаковський, С.В., Лаврик, І.М. (2010). Криптографічні методи захисту інформації.
4. Python documentation. https://www.python.org/doc/
5. Asymmetric Encryption - Simply explained. https://www.youtube.com/watch?v=AQDCe585Lnc&ab_channel=SimplyExplained
6. What is Decryption | How Decryption works | what are the types of decryption? FORnSEC Solutions https://www.youtube.com/watch?v=FpJSsmQuzJw&ab_channel=FORnSECSolutions
7. Cryptography and its Types https://www.geeksforgeeks.org/cryptography-and-its-types/

Додаток

```
import shutil
from tqdm import tqdm
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto import Random
from tkinter import Tk, Label, Button, filedialog, messagebox

# Метод для ECB-шифрування зображення
def encrypt_image_ecb(image_path, key):
    # Створюємо резервну копію оригінального файлу зображення
    backup_path = image_path + '.bak'
    shutil.copyfile(image_path, backup_path)

    # Відкриваємо файл для зчитування
    with open(image_path, 'rb') as file:
        # Зчитуємо дані зображення
        image_data = file.read()

    # Створюємо об'єкт шифру AES з ключем та режимом ECB
    cipher = AES.new(key, AES.MODE_ECB)

    # Виконуємо шифрування шляхом додавання доповнення та шифрування даних
    encrypted_data = cipher.encrypt(pad(image_data, AES.block_size))

    # Відкриваємо файл для запису
    with open(image_path, 'wb') as file:
        # Записуємо зашифровані дані у файл
        file.write(encrypted_data)

    print('Шифрування (ECB) завершено...')
    print('Створено резервну копію у файлі:', backup_path)

# Метод для ECB-дешифрування зображення
def decrypt_image_ecb(image_path, key):
    # Створюємо резервну копію зашифрованого файлу зображення
    backup_path = image_path + '.bak'
    shutil.copyfile(image_path, backup_path)

    # Відкриваємо файл для зчитування
    with open(image_path, 'rb') as file:
        # Зчитуємо зашифровані дані
        encrypted_data = file.read()

    # Створюємо об'єкт шифру AES з ключем та режимом ECB
```

```
    cipher = AES.new(key, AES.MODE_ECB)

    # Виконуємо дешифрування шляхом дешифрування даних та видалення доповнення
    decrypted_data = unpad(cipher.decrypt(encrypted_data), AES.block_size)

    # Відкриваємо файл для запису
    with open(image_path, 'wb') as file:
        # Записуємо дешифровані дані у файл
        file.write(decrypted_data)

    print('Дешифрування (ECB) завершено...')
    print('Створено резервну копію у файлі:', backup_path)

# Метод для CBC-шифрування зображення
def encrypt_image_cbc(image_path, key):
    # Створюємо резервну копію оригінального файлу зображення
    backup_path = image_path + '.bak'
    shutil.copyfile(image_path, backup_path)

    # Відкриваємо файл для зчитування
    with open(image_path, 'rb') as file:
        # Зчитуємо дані зображення
        image_data = file.read()

    # Генеруємо випадковий вектор ініціалізації (IV)
    iv = Random.new().read(AES.block_size)

    # Створюємо об'єкт шифру AES з ключем, режимом CBC та IV
    cipher = AES.new(key, AES.MODE_CBC, iv)

    # Виконуємо шифрування шляхом додавання доповнення та шифрування даних
    encrypted_data = cipher.encrypt(pad(image_data, AES.block_size))

    # Відкриваємо файл для запису
    with open(image_path, 'wb') as file:
        # Записуємо IV та зашифровані дані у файл
        file.write(iv + encrypted_data)

    print('Шифрування (CBC) завершено...')
    print('Створено резервну копію у файлі:', backup_path)

# Метод для CBC-дешифрування зображення
def decrypt_image_cbc(image_path, key):
    # Створюємо резервну копію зашифрованого файлу зображення
    backup_path = image_path + '.bak'
    shutil.copyfile(image_path, backup_path)

    # Відкриваємо файл для зчитування
```

```
    with open(image_path, 'rb') as file:
        # Зчитуємо IV та зашифровані дані
        iv = file.read(AES.block_size)
        encrypted_data = file.read()

    # Створюємо об'єкт шифру AES з ключем, режимом CBC та IV
    cipher = AES.new(key, AES.MODE_CBC, iv)

    # Виконуємо дешифрування шляхом дешифрування даних та видалення доповнення
    decrypted_data = unpad(cipher.decrypt(encrypted_data), AES.block_size)

    # Відкриваємо файл для запису
    with open(image_path, 'wb') as file:
        # Записуємо дешифровані дані у файл
        file.write(decrypted_data)

    print('Дешифрування (CBC) завершено...')
    print('Створено резервну копію у файлі:', backup_path)

# Метод для відображення повідомлення про помилку у вікні повідомлення
def display_error_message(message):
    messagebox.showerror('Помилка', message)

# Метод для відкриття діалогового вікна файлу та повернення обраного шляху файлу
def open_file_dialog():
    root = Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename(filetypes=[('Зображення', '*.jpg;*.jpeg;*.png;*.bmp')])
    root.destroy()
    return file_path

# Метод для обробки події натискання кнопки шифрування ECB
def encrypt_button_click_ecb():
    file_path = open_file_dialog()
    if file_path:
        key = b'Sixteen byte key'
        try:
            encrypt_image_ecb(file_path, key)
            messagebox.showinfo('Шифрування (ECB)', 'Шифрування (ECB) успішно завершено.')
        except Exception as e:
            display_error_message(str(e))

# Метод для обробки події натискання кнопки дешифрування ECB
def decrypt_button_click_ecb():
    file_path = open_file_dialog()
    if file_path:
        key = b'Sixteen byte key'
        try:
```

```
            decrypt_image_ecb(file_path, key)
            messagebox.showinfo('Дешифрування (ECB)', 'Дешифрування (ECB) успішно завершено.')
        except Exception as e:
            display_error_message(str(e))

# Метод для обробки події натискання кнопки шифрування CBC
def encrypt_button_click_cbc():
    file_path = open_file_dialog()
    if file_path:
        key = b'Sixteen byte key'
        try:
            encrypt_image_cbc(file_path, key)
            messagebox.showinfo('Шифрування (CBC)', 'Шифрування (CBC) успішно завершено.')
        except Exception as e:
            display_error_message(str(e))

# Метод для обробки події натискання кнопки дешифрування CBC
def decrypt_button_click_cbc():
    file_path = open_file_dialog()
    if file_path:
        key = b'Sixteen byte key'
        try:
            decrypt_image_cbc(file_path, key)
            messagebox.showinfo('Дешифрування (CBC)', 'Дешифрування (CBC) успішно завершено.')
        except Exception as e:
            display_error_message(str(e))

# Створення вікна GUI
window = Tk()
window.title('Шифрування/Дешифрування зображень')
window.geometry('300x200')

# Створення міток
label = Label(window, text='Виберіть режим:')
label.pack()

# Створення кнопок для режиму ECB
label_ecb = Label(window, text='Режим ECB:')
label_ecb.pack()

encrypt_button_ecb = Button(window, text='Шифрувати', command=encrypt_button_click_ecb)
encrypt_button_ecb.pack()

decrypt_button_ecb = Button(window, text='Дешифрувати', command=decrypt_button_click_ecb)
decrypt_button_ecb.pack()

# Створення кнопок для режиму CBC
label_cbc = Label(window, text='Режим CBC:')
```

```
label_cbc.pack()

encrypt_button_cbc = Button(window, text='Шифрувати', command=encrypt_button_click_cbc)
encrypt_button_cbc.pack()

decrypt_button_cbc = Button(window, text='Дешифрувати', command=decrypt_button_click_cbc)
decrypt_button_cbc.pack()

window.mainloop()
```