МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проект

здобувача освіти денної форми навчання
КБ.02.22.000.ДП

ЧЕЛАК
ДМИТРІЯ ВІКТОРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка моделі захисту зображень за допомогою вбудованого шифрування

Проектний матеріал складається з пояснювальної записки на 81 сторінках та графічного (презентаційного) матеріалу на 15 аркушах (слайдах)

Дипломник _____ (Челак Д.В.)

Керівник _____ (Кривченко Ю.В.)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «23» червня 2025 р. Протокол ЕК № 3

Оцінка ЕК 5 (відмінно) / 90 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Челак Дмитрію Вікторовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка моделі захисту зображень за допомогою вбудованого шифрування*

затверджена наказом по коледжу від "14" 11 2024 р. № 246

2. Термін здачі закінченого проекту 16.06.25

3. Вихідні данні до проекту *1. Реалізувати модель шифрування/дешифрування зображень на базі гомоморфного шифрування з середньоарифметичною фільтрацією та симетричного алгоритму AES; 2. Передбачити шифрування і дешифрування растрових файлів PNG; 3. Реалізувати зберігання ключів шифрування у текстових файлах; 4. Реалізувати візуальний інтерфейс користувача для створюваного додатку; 5. Розробку застосунку виконати мовою C# або C++*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз сучасних методів і засобів захисту зображень; Аналіз основних форматів растрових зображень; Реалізація середньоарифметичної фільтрації для захисту зображень; Розробка програмної моделі та блок-схем алгоритмів застосунку для захисту зображень; Аналіз роботи застосунку та моделі захисту зображень у форматі PNG; Економічний розділ; Розділ охорони праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Структура файлу PNG; Обробка квадратної апертури за допомогою середньоарифметичного фільтра; Процес шифрування зображень за алгоритмом AES; Діаграма роботи алгоритму AES; Схема шифрування зображень з захищеною середньоарифметичною фільтрацією; Схема послідовності операцій перемішування стовпців при захищеній фільтрації; БСА середньо-арифметичної фільтрації зображень; Діаграма класів застосунку для захисту зображень; Приклади роботи застосунку при шифруванні і дешифруванні зображення

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Кривченко Ю.В.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Кривченко Ю.В.* _____
(підпис)

Завдання прийняв до виконання *Челак Д.В.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка задачі проектування*	15.05.25	Виконано
2	*Аналіз технічного завдання та пошук літератури*	16.05.25	Виконано
3	*Аналіз сучасних методів і засобів захисту зображень*	17.05.25	Виконано
4	*Аналіз основних форматів растрових зображень*	19.05.25	Виконано
5	*Реалізація середньоарифметичної фільтрації для*	22.05.25	Виконано
	захисту зображень		
6	*Моделювання роботи алгоритму модифікованого AES*	26.05.25	Виконано
7	*Розробка БСА та діаграми класів застосунку*	01.06.25	Виконано
8	*Реалізація візуального інтерфейсу застосунку*	06.06.25	Виконано
9	*Написання коду програми мовою C#*	10.06.25	Виконано
10	*Аналіз роботи застосунку та моделі захисту зображень*	11.06.25	Виконано
	у форматі PNG		
11	*Випробування застосунку та аналіз результатів*	12.06.25	Виконано
12	*Виконання економічних розрахунків*	13.06.25	Виконано
13	*Розробка питань з охорони праці та техніки безпеки*	14.06.25	Виконано
14	*Підготовка мультимедійної презентації проекту*	16.06.25	Виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 22 000. 00 ДП	Дипломний проект		
А4			КБ 02. 22 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 22 000. 00 ДП		
Розробив		Челак Д.В.		16.06	Розробка моделі захисту зображень за допомогою вбудованого шифрування	Літ.	Аркуш 4 / Аркушів 81
Перевірив		Кривченко Ю.В.		16.06		Н Д П	
Н. контр.		Петрашова В.І.		16.06		ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.		16.06			

ЗМІСТ

ВСТУП

Сучасний розвиток інформаційних технологій спричинив масове поширення цифрових зображень у різних сферах діяльності – від соціальних мереж та комунікацій до медичних досліджень і систем безпеки. З огляду на зростання обсягів передачі конфіденційних даних, зображень та мультимедійного контенту, забезпечення їх захисту від несанкціонованого доступу набуває особливої актуальності. Нетрадиційні підходи до шифрування, які дозволяють здійснювати обчислення без попереднього розшифрування даних, є перспективним напрямком досліджень у галузі безпеки інформації.

Одним із таких підходів є інтегроване шифрування, яке безпосередньо вбудовується в процес обробки зображень. Традиційні алгоритми шифрування зазвичай призначені для забезпечення конфіденційності даних під час їх зберігання або передачі, але не дозволяють виконувати подальшу аналіз і обробку закодованої інформації без розшифрування. Використання методів адитивно гомоморфного шифрування, зокрема алгоритму Paillier, дозволяє проводити арифметичні операції над зашифрованими даними, що відкриває можливості для безпечної обробки зображень без втрати їх вмісту.

Метою цього дипломного проекту є реалізація моделі захисту зображень із застосуванням вбудованого шифрування, яка об'єднує алгоритми цифрової обробки зображень, зокрема метод гомоморфного шифрування (на основі перестановки стовпців із середньоарифметичною фільтрацією) з симетричним шифруванням за стандартом AES. Цей підхід забезпечить не лише високий рівень захисту конфіденційних даних, але й дозволить виконувати подальшу обробку зображення (наприклад, фільтрацію) без необхідності розкривати його вміст. Також метою роботи є створення простого засобу захисту зображень, який може бути застосований у системах з високими вимогами до безпеки передачі та зберігання зображень. Теоретична основа роботи буде ґрунтуватися на аналізі сучасних методів цифрової обробки та криптографії, а практична – на розробці, реалізації та експериментальному тестуванні ефективної моделі вбудованого шифрування.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз сучасних методів і засобів захисту зображень

У сучасних інформаційних системах захист зображень виступає як одна з найактуальніших задач завдяки збільшенню обсягів передаваного та зберіганого візуального контенту. Ключовою вимогою є не лише конфіденційне зберігання зображень, але й можливість їх обробки без розкриття вихідних даних. Таким чином, інтегроване шифрування, яке забезпечує обчислення над зашифрованими даними, стає основою сучасних підходів до забезпечення безпеки цифрових зображень.

1.1.1 Спеціалізовані криптографічні алгоритми для захисту зображень

Стандартні методи, такі як блокове шифрування (AES, RSA), забезпечують високий рівень захисту зображень, проте не підтримують операції над зашифрованими даними без попереднього розшифрування. Альтернативним рішенням є використання гомоморфного шифрування, що дозволяє проводити арифметичні операції без розкриття вихідної інформації. Наприклад, алгоритм Paillier є адитивно гомоморфним, що виражається у формулі шифрування:

$$c = gm \cdot rn \backslash mod n^2, \qquad (1.1)$$

де m — повідомлення (значення пікселя), g, n — публічні параметри, а r — випадково вибране число, взаємно просте з n. Дешифрування виконується за допомогою операції:

$$m = L\left(c^{\lambda} \backslash mod n^2\right) \cdot \mu \backslash mod n, \quad \text{де} \quad L(u) = \frac{u-1}{n} \qquad (1.2)$$

Іншим підходом є інтервальне шифрування, яке базується на статистичному аналізі розподілу пікселів зображення. Значення пікселя відображаються у відповідні інтервали, і зашифроване число вибирається випадково з заданого діапазону. Такий метод дозволяє враховувати специфіку розподілу яскравості зображення та забезпечувати ефективне дешифрування на підставі збережених статистичних показників.

Одним із ключових факторів є можливість виконання обчислювальних

операцій без розшифрування даних. Завдяки адитивній гомоморфії алгоритму Paillier, математична операція додавання зашифрованих значень здійснюється за наступною схемою:

$$E(m_1) \cdot E(m_2) \backslash mod n^2 = E(m_1 + m_2) \qquad (1.3)$$

де *E(m)* позначає зашифроване значення повідомлення *m*. Це дозволяє, наприклад, реалізувати згорткові фільтри без необхідності виснажливого дешифрування кожного пікселя, що значно підвищує рівень конфіденційності даних під час їх обробки. Запропонована схема забезпечує збереження інформації, адже операції виконуються над шифротекстом навіть при застосуванні стандартних операцій фільтрації, таких як медіана або згортка з використанням ядра.

З математичної точки зору, ефективність даних підходів оцінюється за кількома критеріями, серед яких можна виділити точність відновлення вихідного зображення, обчислювальну складність та витрати пам'яті. Розглянемо узагальнену формулу для зашифрованої згортки зображення:

$$f(i,j) = \sum_{k=-a}^{a} \sum_{l=-b}^{b} w(k,l) \cdot E\big(I(i+k, j+l)\big) \qquad (1.4)$$

де *w(k, l)* — коефіцієнти ядра, *I(i, j)* — значення яскравості пікселя, а *E* позначає операцію шифрування. При цьому обчислення здійснюється над зашифрованими даними, що дозволяє отримати зашифроване результатне зображення, яке здатне пройти процедуру дешифрування для відновлення оригінальної інформації.

1.1.2 Захист зображень за допомогою алгоритму AES

Алгоритм шифрування AES (Advanced Encryption Standard), що базується на шифрі Rijndael, є симетричним алгоритмом, який широко застосовується для захисту даних, зокрема зображень, завдяки високій швидкості та ефективності реалізації апаратним і програмним шляхом. Для шифрування зображень даний алгоритм застосовується наступним чином: вихідне зображення, після попередньої обробки (наприклад, перетворення у двійкове представлення або отримання

матриці пікселів), розбивається на блоки довжиною 128 біт. Кожен блок розглядається як послідовність байтів, які упаковуються у матрицю розміром 4×4. На рис. 1.1 показано, як 128-бітний блок інформації формується з пікселів зображення та записується у матрицю 4×4, над якою виконуються подальші операції шифрування.

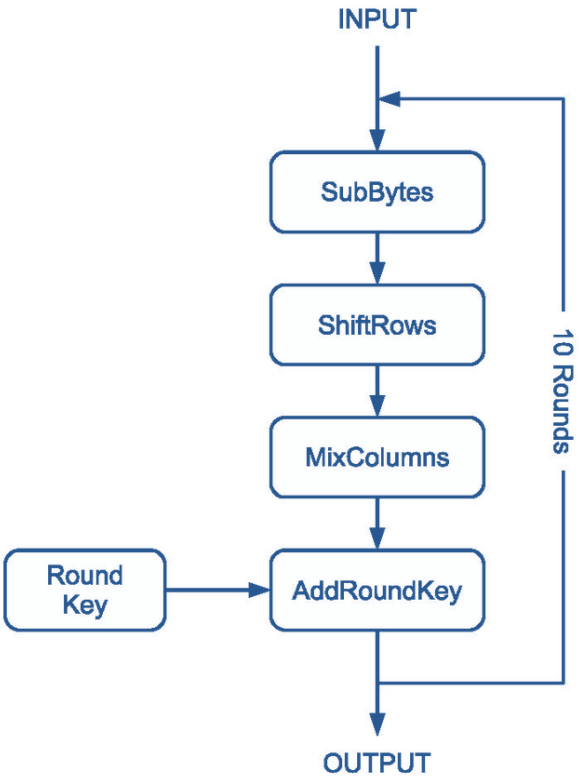


Рисунок 1.1. Процес шифрування зображень за алгоритмом AES

Основні операції шифрування в AES включають чотири базові процедури:

1. SubBytes. На цьому етапі кожен байт матриці замінюється відповідно до фіксованої таблиці S-боксів. Таблиця заміщення у AES не змінюється від раунду до раунду, що контрастує з деякими іншими алгоритмами;

2. ShiftRows. Ця процедура виконує циклічний зсув байтів у кожному рядку матриці: перший рядок залишається незмінним, другий зсувається на один байт вліво, третій – на два, а четвертий – на три;

3. MixColumns. Для кожного стовпця матриці здійснюється операція багаточленового множення у полі Галуа GF(2^8). У цій фазі кожен стовпець обробляється за допомогою полінома $c(x) = 3x^3 + x^2 + x + 2$, тоді як при дешифруванні використовується зворотна операція з фіксованим поліномом c(x) = $11x^3 + 13x^2 + 9x + 14$;

4. AddRoundKey. На цьому етапі поточний стан блоку піддається побітовій операції XOR з відповідним раундовим ключем, що отримується із головного ключа.

Криптографічна структура AES будується за принципом мережі заміщення-перестановки (Substitution-Permutation Network), що забезпечує достатній рівень дифузії та плутанини. Початкове шифрування розпочинається з операції AddRoundKey, після чого повторюється низка ітерацій, кожна з яких включає процедури SubBytes, ShiftRows, MixColumns та AddRoundKey. Загальна кількість раундів залежить від довжини ключа (10 раундів для 128-бітного, 12 для 192-бітного, 14 для 256-бітного ключів).

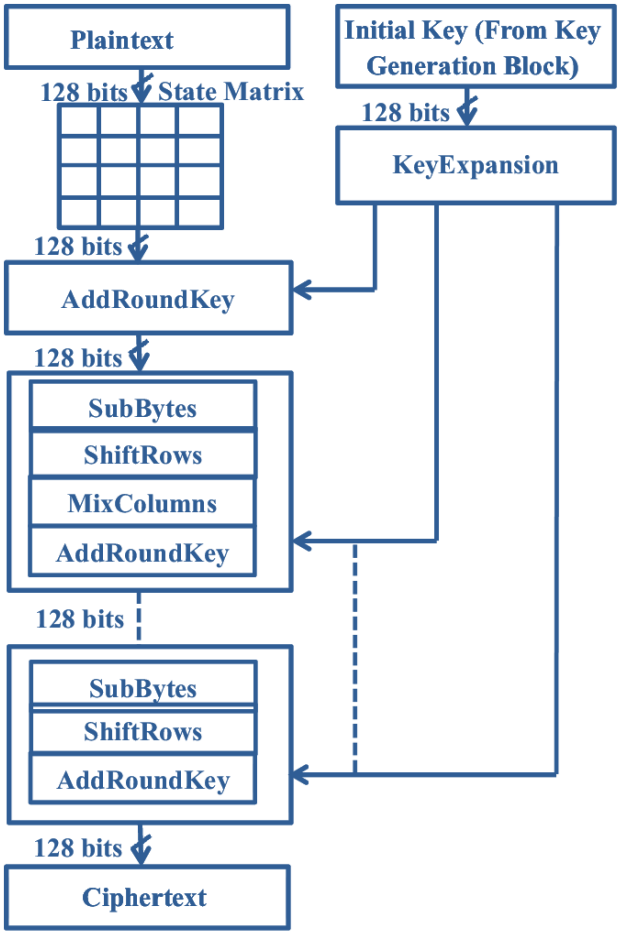


Рисунок 1.2. Діаграма роботи алгоритму AES

На рис.1.2 представлено схему послідовних етапів шифрування: початкова трансформація, раунди зокрема з процедурами SubBytes, ShiftRows, MixColumns, AddRoundKey, а також заключний раунд, де операція MixColumns не застосовується.

Процес розширення ключа є невід'ємною частиною алгоритму. Головний ключ (наприклад, 128-бітний, що представлений матрицею 4×4) розгортається в серію раундових ключів. Для отримання першого раундового ключа застосовується операція RotWord, що циклічно зсуває байти стовпця, після чого проводиться підстановка за допомогою S-боксу (SubBytes). Далі отриманий блок комбінується з константним масивом Rcon через операцію XOR, що дозволяє отримати новий раундовий ключ. Решта ключових блоків утворюється за схемою:

$$W_5 = W_1 \oplus \text{SubBytes}\big(\text{RotWord}(W_1)\big) \oplus R_1, \qquad (1.5)$$

а наступні блоки обчислюються як:

$$W_6 = W_2 \oplus W_5, \quad W_7 = W_3 \oplus W_6, \quad W_8 = W_4 \oplus W_7 \qquad (1.6)$$

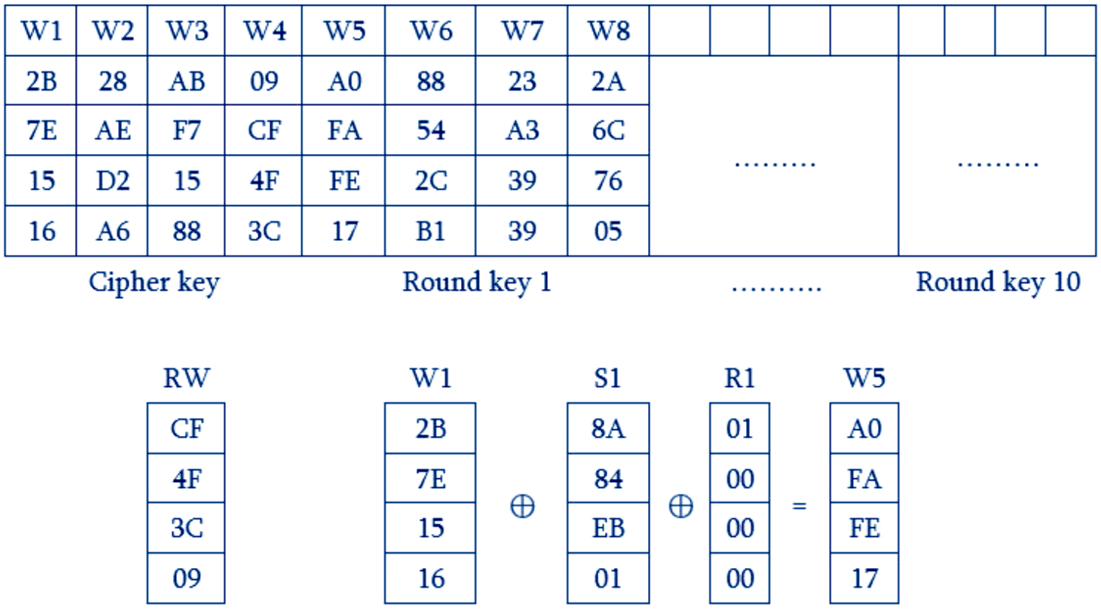


Рисунок 1.3. Етап отримання раундових ключів у AES

На рис.1.3 показано, як з початкового ключа формуються раундові ключі за допомогою операцій RotWord, SubBytes та XOR з Rcon.

При шифруванні зображення алгоритм AES дозволяє обробляти дані на рівні окремих блоків. Кожен блок, що містить 128 біт (16 байтів), піддається вищезазначеним фазам. Оскільки зображення зазвичай містить багато таких блоків, результатом шифрування є набір зашифрованих блоків, які разом формують захищене зображення.

Переваги застосування AES для шифрування зображень полягають у високій швидкості операцій завдяки роботі на рівні байтів та апаратній підтримці

алгоритму у багатьох процесорах. Однак метод має і недоліки:

- Одинакове шифрування ідентичних даних. Якщо два блоки вхідних даних однакові, їх зашифровані версії також будуть однаковими, що може створювати потенційні вразливості;
- Проста структурна архітектура. Хоча це робить алгоритм швидким, це також дає можливість певних типів атак, зокрема атак із використанням аналізу сторонніх каналів;
- Складність коректної імплементації. Правильне і безпечне впровадження алгоритму на програмному рівні вимагає дотримання численних стандартів і правил роботи з пам'яттю.

Для дешифрування зашифрованих блоків зображення застосовується зворотний процес. Послідовність операцій змінюється: спочатку виконується зворотна операція AddRoundKey, потім InvMixColumns (за винятком останнього раунду), InvShiftRows та InvSubBytes, що дозволяє відновити початковий блок даних.

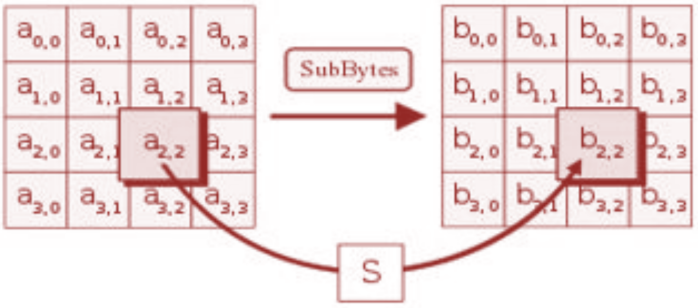


Рисунок 1.4. Етап SubBytes у AES

На рис.1.4 показано, як кожен байт блоку замінюється згідно з таблицею S-боксів, що застосовується як для шифрування, так і для дешифрування (у вигляді InvSubBytes).

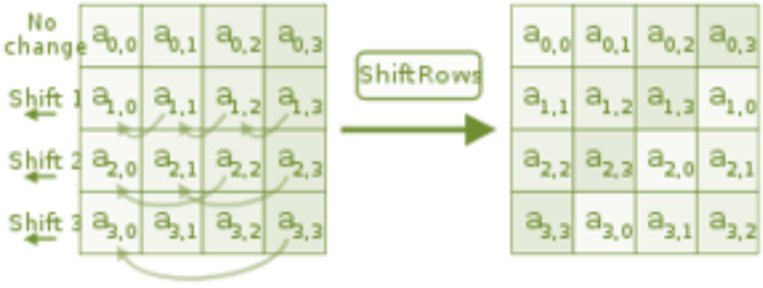


Рисунок 1.5. Етап ShiftRows у AES

На рис.1.5 ілюструється, як здійснюється циклічний зсув рядків у 4×4 матриці стану блоку: другий ряд зміщується на 1 байт, третій – на 2, четвертий – на 3 байти.

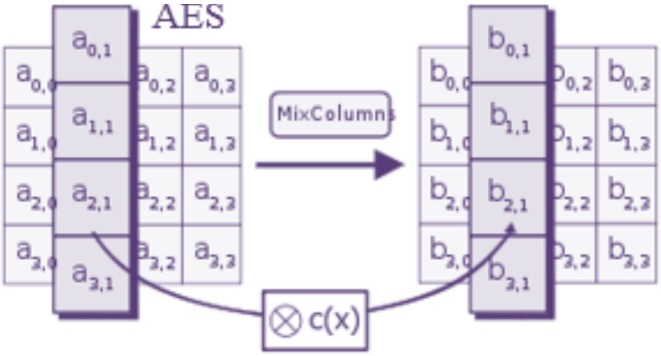


Рисунок 1.6. Етап MixColumns у AES

На рис.1.6 демонструється операція MixColumns, де кожен стовпець множиться на фіксований поліном у полі Галуа GF(2^8), що забезпечує змішання даних у стовпцях.

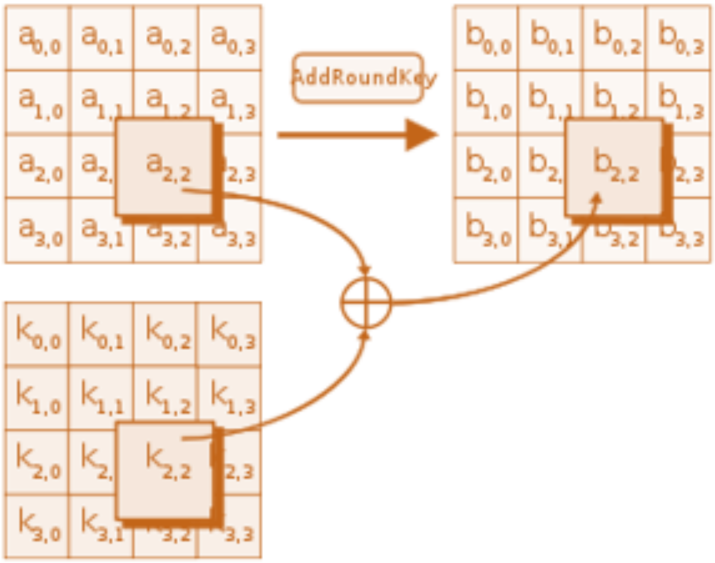


Рисунок 1.7. Етап AddRoundKey у AES

На рис.1.7 показано, як здійснюється побітова операція XOR кожного байта блока із відповідним байтом раундового ключа, що є фінальним етапом кожного раунду шифрування.

Використання алгоритму AES для шифрування зображень забезпечує високий рівень захисту завдяки надійним математичним властивостям, швидкості виконання та широкій апаратній підтримці. Незважаючи на деякі недоліки, такі як

однаковість шифрування однакових блоків і певна вразливість до атак за допомогою аналізу сторонніх каналів, AES залишається стандартом для сучасних систем захисту даних. Широке застосування AES в апаратних і програмних засобах гарантує довгострокову безпеку даного алгоритму, що особливо важливо для захисту зображень у критично важливих додатках.

1.1.3 Захист зображень за допомогою алгоритму RSA

Алгоритм RSA (Rivest–Shamir–Adleman) є асиметричним криптографічним алгоритмом, що базується на математичній складності факторизації великих чисел. На відміну від симетричних алгоритмів (наприклад, AES), для шифрування даних використовується пара ключів: відкритий (публічний) та закритий (приватний). У задачах захисту зображень RSA може застосовуватися для шифрування окремих блоків даних (наприклад, числового представлення пікселів) або для захисту симетричних ключів, що використовуються для подальшого шифрування зображень.

Нехай p та q – два великі прості числа. Модуль n розраховується як: $n = p \times q$. Функція Ейлера для n дорівнює: $\phi(n) = (p-1)(q-1)$. Вибирається число e, яке є взаємно простим з $\phi(n)$. Відкритий ключ формується як пара (n, e). Приватний ключ отримують як число d, для якого виконується співвідношення: $e \cdot d \equiv 1 \left(\mathrm{mod}\ \phi(n)\right)$. Процес шифрування кожного блоку даних m, який задовольняє умові $m < n$, виконується за допомогою формули $c = m^e \bmod n$, де c – зашифрований блок. Дешифрування відбувається за допомогою $m = c^d \bmod n$.

Щоб застосувати RSA для захисту зображень, зображення спочатку перетворюють у числове представлення. Це може бути, наприклад, послідовність пікселів або блокове представлення (де кожен блок має розмір, що не перевищує значення n. Далі кожен блок підлягає шифруванню за допомогою публічного ключа RSA. Загальний процес можна подати наступною схемою.

На рис.1.8 зображено генерацію ключів, шлях перетворення блоку даних m (представлення частини зображення) у зашифрований блок c, а також процес дешифрування m за допомогою приватного ключа.

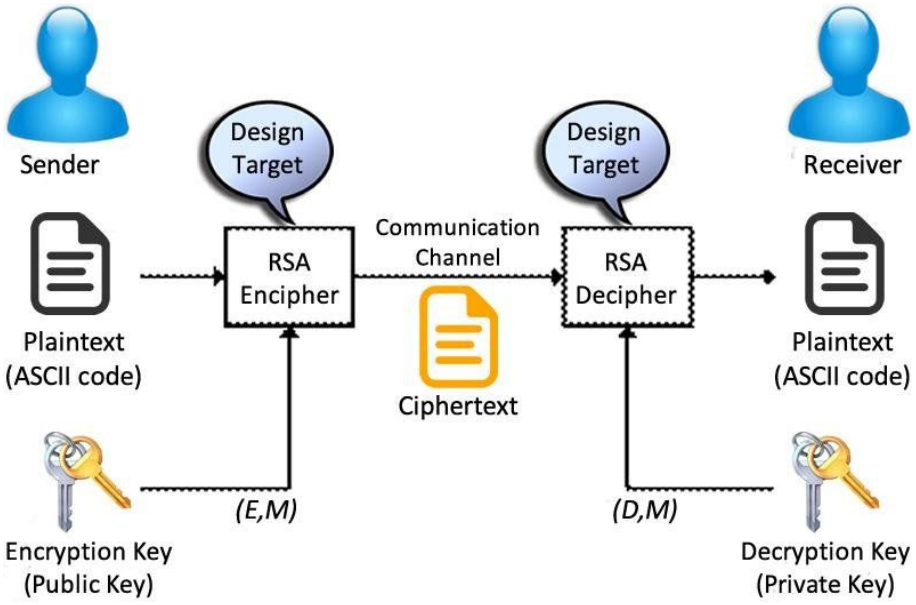


Рисунок 1.8. Схема криптографії RSA

Процес шифрування зображення RSA належним чином включає:

1. Попередню обробку зображення: Зображення розбивається на блоки заданого розміру, кожен з яких перетворюється у числове представлення m так, щоб $m < n$;
2. Шифрування: Кожен блок даних шифрується за допомогою формули $c = m^e \, mod \, n$. Отримані зашифровані блоки формують захищене зображення, яке може бути збережене або передане;
3. Дешифрування: За допомогою приватного ключа виконується зворотний процес: $m = c^d \, mod \, n$, що дозволяє відновити вихідні блоки та зібрати з них первинне зображення.

На рис.1.9 представлено, як вихідне зображення кодується в числове представлення, розбивається на блоки, після чого кожен блок шифрується формулою RSA. Після передачі або зберігання зашифрованих даних відбувається дешифрування окремих блоків за допомогою приватного ключа для відновлення зображення.

Переваги RSA для шифрування зображень:

- Асиметричність: Використання різних ключів для шифрування та дешифрування дозволяє підвищити безпеку передачі даних;
- Математична надійність: Складність факторизації великих чисел гарантує стійкість до криптоаналізу, за умови правильного вибору параметрів.

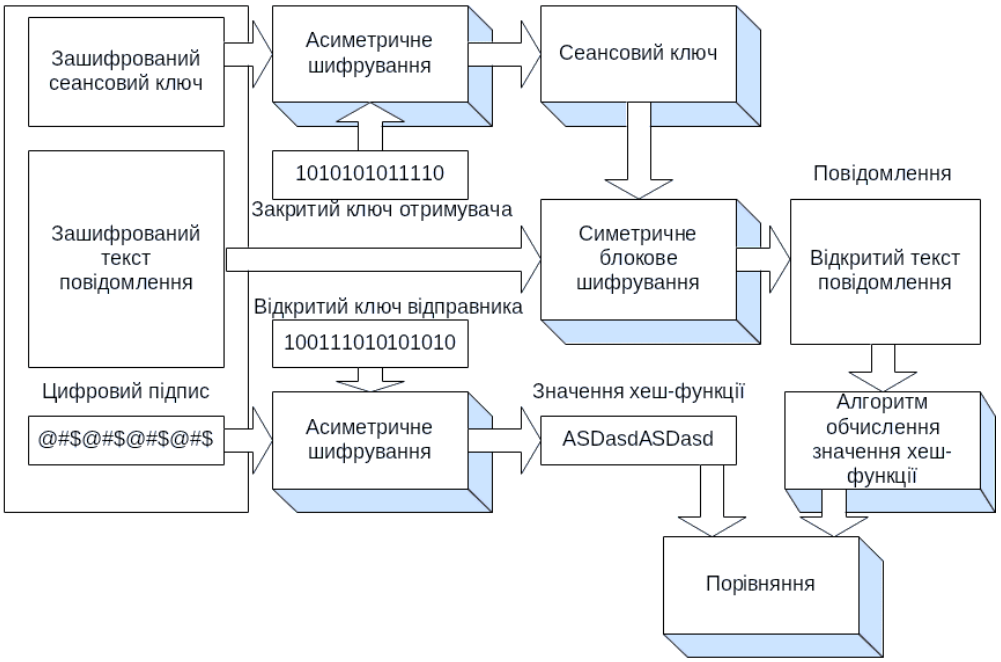


Рисунок 1.9. Процес шифрування зображення за допомогою RSA

Недоліки RSA для шифрування зображень:

- Обчислювальна складність: RSA є значно повільнішим у порівнянні з симетричними алгоритмами, такими як AES. Це обмежує його застосування для шифрування великих обсягів даних (як зображень високої роздільності);
- Обмеження розміру блоку: Оскільки m має бути меншим за n, зображення необхідно попередньо розбивати на менші блоки, що може збільшувати складність реалізації;
- Застосування для невеликих даних: Зазвичай RSA використовується для шифрування симетричних ключів або невеликих повідомлень, а не для прямих масивів великих об’ємів даних. Для захисту зображень RSA може бути застосований як частина гібридної системи, де симетричний ключ шифрує зображення, а RSA – цей ключ.

Алгоритм RSA забезпечує високий рівень захисту завдяки своїй асиметричній природі та використанню складних математичних операцій. При застосуванні до зображень RSA можливо використовувати для шифрування окремих блоків зображення або для захисту симетричного ключа, що використовується для прямого шифрування за допомогою більш ефективних алгоритмів. Незважаючи на певні обмеження, RSA залишається важливим інструментом у забезпеченні криптографічного захисту даних.

1.1.4 Інтегроване шифрування та обробка зображень

Для ілюстрації відмінностей між застосуванням стандартних та гомоморфних методів захисту зображень наведено табл.1.1, що порівнює основні показники. Ця таблиця демонструє, що кожен з підходів має свої сильні та слабкі сторони. Стандартні методи гарантують строгу криптографічну безпеку, але позбавлені можливості виконання операцій над зашифрованими даними. Адитивно гомоморфне шифрування, хоча і відкриває нові функціональні можливості, вимагає значних обчислювальних ресурсів через роботу з великими числами. Інтервальне шифрування є компромісним рішенням, де статистичні властивості зображення використовуються для збереження інформації, що дозволяє заощадити частину обчислювальної потужності.

Таблиця 1.1. Стандартні та гомоморфні методи захисту зображень

Показник	**Стандартне шифрування (AES, RSA)**	**Адитивно-гомоморфне шифрування (Paillier)**	**Інтервальне шифрування**
Конфіденційність	Висока	Висока	Висока (залежно від статистичного аналізу)
Можливість обчислення над даними	Немає (небезпечно)	Так (додавання, масштабування)	Обмежено (залежить від методики вибору інтервалів)
Обчислювальна складність	Помірна	Висока (через великі цілі числа)	Помірна-підвищена
Витрати пам'яті	Невеликі	Високі	Помірні
Гнучкість застосування	Обмежена	Висока (адаптована для спеціальних обчислень)	Середня

Ще одним важливим аспектом є забезпечення відновлення даних з мінімальними помилками. Для цього використовують додаткові схеми перевірки цілісності зображення, такі як гешування модулів зображення перед і після обробки. Наприклад, обчислення контрольного значення *H* може здійснюватися за формулою: $\mathrm{H(I)} = \mathrm{SHA\text{-}256(I)}$, де (I) — множина значень пікселів зображення, що дозволяє виявити можливі помилки при передачі або обробці.

На рис.1.10 подано схематичну ілюстрацію процесу інтегрованого шифрування та обробки зображень з використанням гомоморфних методів.

Аналіз сучасних методів захисту зображень показує, що інтегровані підходи,

де шифрування безпосередньо комбінується із обробкою даних, дозволяють значно підвищити рівень безпеки конфіденційної інформації. Якісне поєднання криптографічних алгоритмів із стандартними методами обробки зображень сприяє розробці ефективної моделі захисту, здатної забезпечити як збереження, так і подальшу обробку даних без компрометації їх цілісності.

Рисунок 1.10. Схема процесу інтегрованого шифрування зображень

1.2 Аналіз основних форматів растрових зображень

Для розробки моделі захисту зображень за допомогою вбудованого шифрування важливо розуміти, як представлені ці зображення на рівні даних. Основні формати растрових зображень можуть суттєво відрізнятися як за способом збереження пікселів, так і за використаними алгоритмами стиснення.

Некомпресовані формати (BMP) безпосередньо представляють кожен піксель у окремій байтовій послідовності, що полегшує поділ каналу на блоки для симетричних алгоритмів шифрування (AES) або для попереднього шифрування блоками за методом RSA. У той же час формати з втратами, такі як JPEG, можуть створювати додаткові труднощі через внутрішню оптимізацію та стиснення, що негативно впливає на точність відновлення даних після дешифрування.

Аналізуючи ключові характеристики основних форматів (роздільну здатність, тип стиснення, бітову глибину), можна обрати найбільш оптимальний формат для експериментів із захисту зображень. Це дозволить забезпечити якості шифрування і, одночасно, мінімізувати втрати даних під час обробки.

Різні застосунки та системи можуть вимагати використання певного формату через сумісність з іншими технологіями. Детальний аналіз форматів rastрових зображень допоможе обґрунтувати вибір конкретного стандарту для застосування у моделі захисту, адаптуючи алгоритми шифрування під його особливості.

1.2.1 Аналіз формату PNG

PNG (Portable Network Graphics) – це відкритий формат растрових зображень, орієнтований на забезпечення високої якості графічних даних за рахунок використання безвтратного стиснення. Формат PNG був створений як альтернатива GIF із усуненням обмежень ліцензування та підтримки прозорості. Його структура базується на концепції «chunk»-ів (блоків даних), що дозволяє гнучко організовувати зображення та забезпечує модульність при обробці.

Файл PNG починається з фіксованого 8-байтового сигнатурного поля, яке дозволяє системам розпізнавати формат і гарантувати, що дані не були пошкоджені при передачі: PNG Signature = {89 50 4E 47 0D 0A 1A 0A}.

Після сигнатури файл складається з послідовності блоків (chunk), де кожен блоку має таку структуру:

- Довжина блоку (4 байти): Визначає розмір даних блоку;
- Тип блоку (4 байти): Наприклад, IHDR, PLTE, IDAT, IEND;
- Дані блоку: Конкретна інформація, що залежить від типу блоку;
- CRC (4 байти): Контрольна сума для перевірки цілісності даних блоку.

Найважливіші блоки у файлі PNG:

- IHDR: Початковий блок, що містить основну інформацію: ширину, висоту (по 4 байти кожна), бітову глибину, тип кольору, метод стиснення, метод фільтрації та інтерлейсінг. Ця інформація є критичною для правильної обробки зображення;
- PLTE: (опційний) Блок палітри, який використовується для індексованих зображень;
- IDAT: Один або декілька блоків, що містять основні дані зображення, стиснуті за алгоритмом DEFLATE (без втрат);
- IEND: Заключний блок, що сигналізує про кінець файлу PNG.

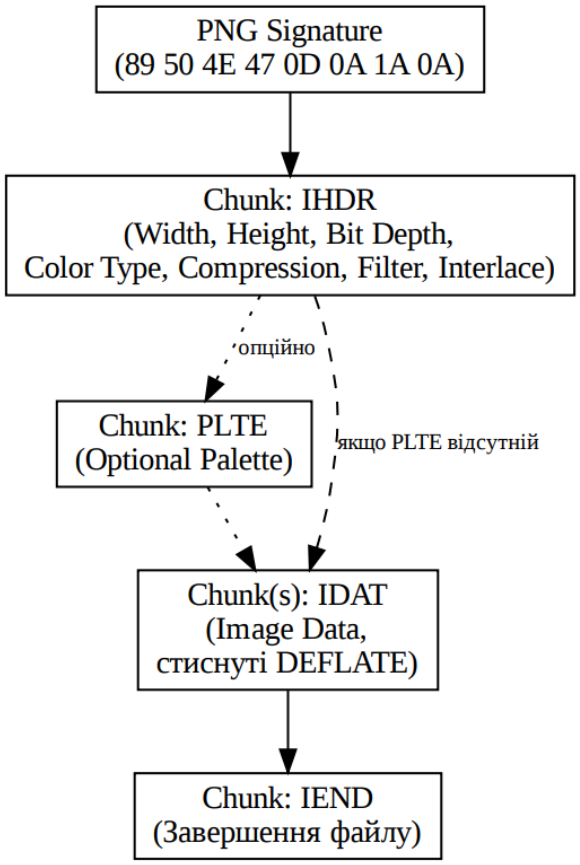


Рисунок 1.11. Структура файла PNG

Особливості та переваги PNG для шифрування зображень:

1. Безвтратне стиснення: PNG використовує алгоритм DEFLATE для стиснення даних без втрат, що гарантує абсолютну цілісність оригінального зображення після дешифрування. Це є надзвичайно важливим для криптографічних застосувань, де відновлення точного вихідного зображення є

критичною вимогою;

2. Структурована організація даних: Наявність окремих блоків дозволяє легко виділяти критичну інформацію (наприклад, заголовок IHDR) від основних даних зображення. Під час шифрування можна застосувати криптографічні алгоритми до блоків IDAT, зберігаючи метадані в незмінному вигляді. Це дозволяє зберігати коректність формату після розшифрування;

3. Підтримка прозорості: Формат PNG підтримує альфа-канал, що дозволяє зберігати інформацію про прозорість. Це важливо для зображень, де прозорі області мають бути збережені без змін після шифрування та подальшого дешифрування;

4. Стандартизація та сумісність: PNG є міжнародним стандартом, широко підтримуваним у програмних та апаратних засобах. Це забезпечує гарантію того, що методи захисту зображень, розроблені для цього формату, можуть бути легко інтегровані у різні системи обробки.

У контексті криптографії зображень, зображення у форматі PNG зазвичай перетворюється в послідовність байт, що утворюють блоки даних (IDAT). Далі ці блоки обробляються симетричними або асиметричними алгоритмами шифрування. Особливість безвтратного стиснення PNG гарантує, що шифрування не призведе до втрати інформації після розшифрування, а структурована організація допоможе зберегти метадані, такі як розмір зображення та інформацію про колірність.

1.2.2 Аналіз формату BMP

BMP (Bitmap) — це класичний формат зберігання растрових зображень, який був розроблений компанією Microsoft і широко використовується в операційних системах Windows. Завдяки своїй простоті та прямолінійній структурі BMP є зручним форматом для дослідження і впровадження методів криптографічного захисту зображень. Типовий BMP-файл складається з кількох основних компонентів:

- Заголовок файлу (File Header): Перші 14 байтів файлу містять інформацію про його тип та розмір. Заголовок починається з ідентифікатора, зазвичай це

два символи «BM», що позначають, що файл є bitmap-зображенням. Також міститься розмір файлу, зарезервовані поля і зсув до початку піксельних даних;

- Інформаційний заголовок (DIB Header або Bitmap Info Header): Цей розділ, розміру якого може варіюватися (найчастіше використовується BITMAPINFOHEADER розміром 40 байт), містить основні параметри зображення: ширину, висоту, кількість біт на піксель, тип стиснення (якщо використовується), розмір зображення тощо;
- Таблиця палітри (Color Palette): Для зображень з невеликою кількістю кольорів (наприклад, 1-, 4- або 8-бітних) використовується таблиця кольорів, що визначає відповідність між індексом і конкретним значенням кольору. У випадку зображень високої роздільної здатності (16-, 24- або 32-бітних) палітра може бути відсутньою, оскільки кольори задаються безпосередньо;
- Піксельні дані (Pixel Data): Це основна частина файлу, що містить інформацію про кожен піксель зображення. Дані зазвичай розташовані почергово знизу вгору (тобто перший ряд даних відповідає нижньому рядку зображення). Іноді дані можуть бути стиснутими з використанням RLE (Run-Length Encoding), проте найчастіше BMP-файли зберігаються без стиснення, що забезпечує максимум точності відтворення.

Особливості формату BMP:

1. Простота реалізації: Завдяки однозначній і зрозумілій структурі BMP-файлу програмою легко здійснювати доступ до головних блоків інформації (заголовки, палітра, піксельні дані). Це спрощує процес розробки алгоритмів для криптографічного захисту, оскільки доступ до піксельних даних здійснюється безпосередньо.

2. Відсутність (або мінімальне використання) стиснення: Більшість BMP-файлів використовують безвтратне зберігання даних без стиснення, що забезпечує повну цілісність інформації після шифрування та подальшого дешифрування. Однак відсутність стиснення призводить до високого розміру файлів, що може бути недоліком у системах з обмеженими ресурсами.

3. Гнучкість для криптографії: Проста структура BMP дозволяє окремо опрацьовувати метадані (заголовки і таблиця палітри) та піксельні дані. У процесі шифрування можна, наприклад, залишити заголовки без змін для збереження інформації про розмір зображення, а зашифрувати лише масив пікселів. Це забезпечує сумісність із стандартами відображення зображень навіть після криптографічної обробки.

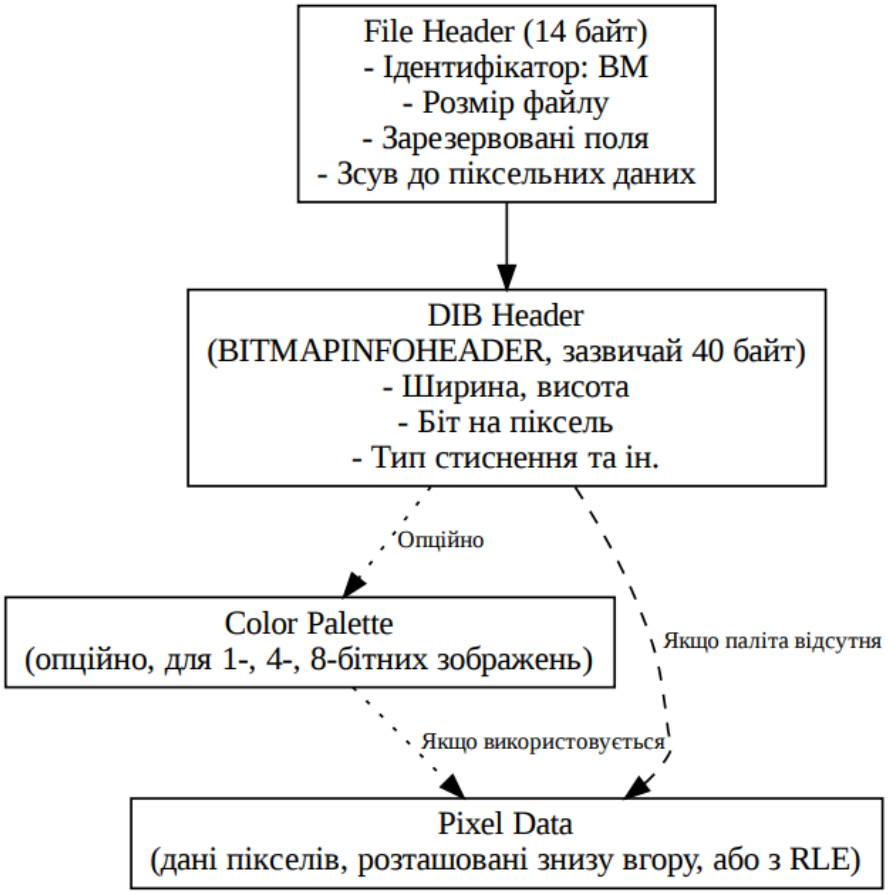


Рисунок 1.12. Структура файла BMP

При розробці систем захисту зображень за допомогою вбудованого шифрування, формат BMP здатен забезпечити:

- Прямий доступ до піксельних даних: Це дозволяє ефективно розбивати файл на блоки для застосування симетричних (наприклад, AES) або асиметричних (наприклад, RSA) методів шифрування;
- Збереження метаданих: Оскільки заголовки BMP-файлів містять інформацію про зображення, їх можна зберігати незмінними, що дозволяє після дешифрування правильно відновити початковий формат і параметри зображення;

- Надійність: Безвтратне зберігання даних гарантує, що жодна інформація не буде втрачена під час процесу шифрування і дешифрування, що особливо важливо для високоякісних зображень.

1.2.3 Аналіз формату JPEG

Формат JPEG (Joint Photographic Experts Group) є одним із найпопулярніших графічних форматів растрової графіки, що використовується для зберігання зображень із стисненням з втратою (lossy compression). Файли JPEG зазвичай мають розширення .jpg, .jfif, .jpe або .jpeg, при цьому .jpg – найпоширеніший варіант на всіх платформах.

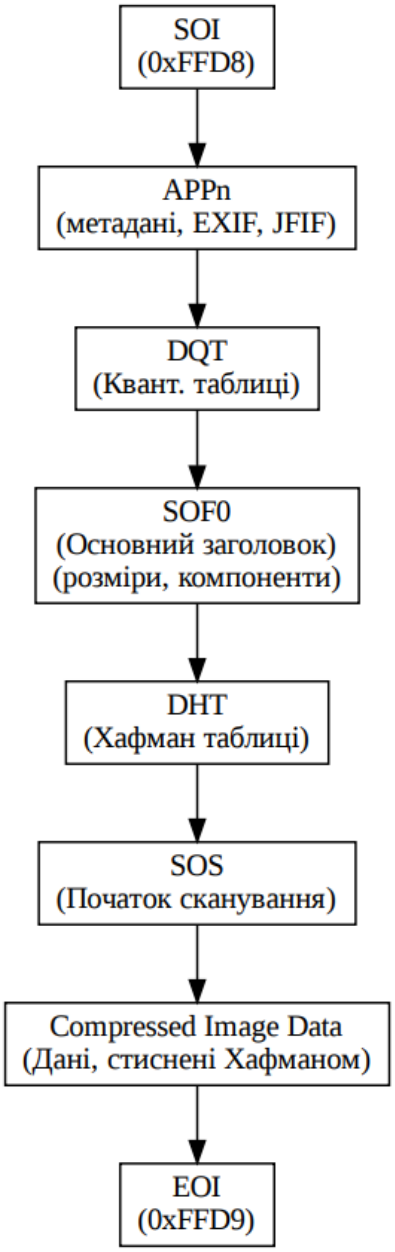


Рисунок 1.13. Структура файла JPEG

Завдяки своїй структурі (рис.1.13), що базується на послідовності спеціально визначених маркерів (вказівників), формат дозволяє швидко знаходити потрібну інформацію, таку як розміри зображення, кількість рядків та колірних компонентів стиснутого зображення.

Файл JPEG складається з послідовності маркерів, кожен з яких починається з байту 0xFF. Деякі маркери складаються лише з пари байтів, тоді як інші містять додаткове інформаційне поле, яке характеризується двома наступними байтами, що вказують довжину маркера (включаючи байти, що задають цю довжину). Основні маркери, що використовуються у JPEG, включають:

- 0xFF 0xD8 SOI (Start of Image): Позначає початок закодованої частини зображення;
- 0xFF 0xC0 SOF0 (Start of Frame 0): Вказує на базовий режим кодування з використанням дискретного косинусного перетворення (DCT) і Хаффманового кодування. Цей маркер містить інформацію про довжину та ширину зображення, кількість компонент (для кожного компонента – 8 біт) та співвідношення сторін компонентів (наприклад, 4:2:0);
- 0xFF 0xC1 SOF1 і 0xFF 0xC2 SOF2: Вказують на розширений та прогресивний режими кодування відповідно, при цьому можуть використовуватись 8- або 12-бітові компоненти. Прогресивний режим дозволяє здійснювати декілька проходів сканування зображення;
- 0xFF 0xC4 DHT (Define Huffman Table): Містить одну або декілька таблиць Хаффмана, які використовуються для оптимального кодування коефіцієнтів DCT;
- 0xFF 0xDB DQT (Define Quantization Table): У цьому маркері містяться таблиці квантування, які, множачи коефіцієнти DCT, визначають ступінь стиснення зображення;
- 0xFF 0xDA SOS (Start of Scan): Позначає початок сканування зображення. У базовому або розширеному режимі використовується одне сканування, у прогресивному – кілька. Маркер може розділяти інформаційну (заголовок) та стиснуту частини зображення;

- 0xFF 0xFE COM (Comment): Містить текстові коментарі або додаткову інформацію;
- 0xFF 0xD9 EOI (End of Image): Позначає кінець закодованої частини зображення.

Ця структурована розбивка дозволяє програмно швидко знайти і витягти необхідну інформацію, наприклад, розміри зображення чи структуру кольорових компонентів, що є особливо важливим при реалізації методів криптографічного захисту.

Процес стиснення зображення у форматі JPEG складається з декількох основних етапів:

1. Перетворення з RGB у YCbCr: Спочатку зображення конвертується з колірного простору RGB до YCbCr, де компонент Y відповідає за яскравість, а Cb і Cr – за колірну інформацію;

2. Підсумовування каналів Cb та Cr: Зазвичай канали Cb і Cr стискаються в 2 рази, що дозволяє зменшити обсяг даних, оскільки людське око менш чутливе до змін у колірних компонентах (рис. 1.14);

3. Блокування зображення: Значення кожного з каналів розбиваються на блоки розміром 8×8 пікселів;

4. Дискретне косинусне перетворення (DCT): До кожного блоку застосовується DCT, який перетворює блок пікселів у масив коефіцієнтів 8×8. Коефіцієнт, що знаходиться у верхньому лівому куті, є DC (середнє значення блоку), а решта 63 – AC коефіцієнтами;

5. Квантування: Отримані коефіцієнти множаться на відповідні елементи квантувальних таблиць (DQT), що дозволяє зменшити кількість значущих цифр для високочастотних компонентів, на яких меню не так добре працює людське око;

6. Кодування Хаффмана: Квантовані коефіцієнти кодуються за допомогою таблиць Хаффмана (DHT). Отримані коди, що представляють блоки коефіцієнтів, розташовуються в порядку каналів (спочатку Y, потім Cb, Cr) (рис. 1.15)

Файл JPEG являє собою послідовність вказівників, кожен з яких містить у собі інформацію про різні аспекти зображення. Наприклад:

- SOI (0xFF 0xD8) сигналізує про початок зображення;
- SOF0 (0xFF 0xC0) містить інформацію про розміри зображення, кількість і деталі компонентів при базовому кодуванні за DCT і Хаффманом;
- DHT (0xFF 0xC4) та DQT (0xFF 0xDB) задають таблиці для ефективного кодування;
- SOS (0xFF 0xDA) вказує на початок основного потоку стиснутих даних;
- EOI (0xFF 0xD9) позначає завершення закодованої частини зображення.

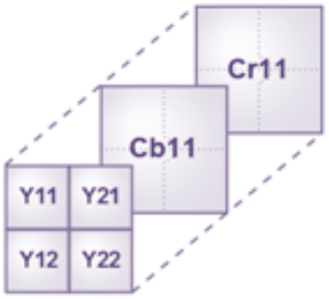


Рисунок 1.14. Стиснення каналів Cb і Cr у 2 рази

Ця структура дозволяє не лише зменшити розмір зображення, але й зберегти достатню якість при оптимальному балансу між розміром і втратою деталей. Завдяки контролю довжини маркерів (які після 0xFF містять два байти довжини) зображення може бути утворене таким чином, щоб інформаційні блоки (наприклад, розміри зображення, кількість компонент) були легко доступні для дешифрування та обробки.

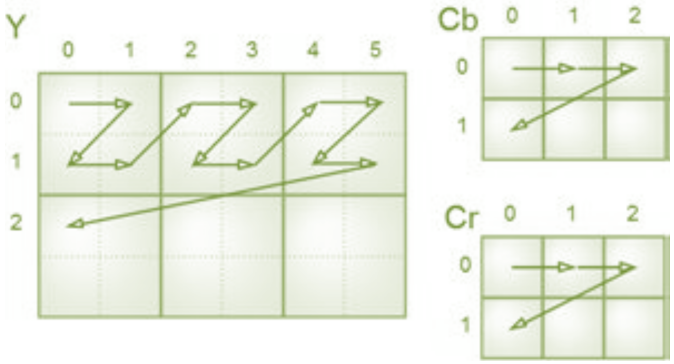


Рисунок 1.15. Розташування каналів Y, Cb, Cr у JPEG-форматі

Існують три типи формату JPEG: базовий, розширений та прогресивний. У даному проекті для реалізації стеганографії використовується базовий режим. У цьому випадку:

- Зображення кодується одним скануванням;
- Маркер SOF0 бере участь у визначенні параметрів зображення (розміри, компоненти, кількість біт на компонент);
- Інші ключові маркери (DQT, DHT, SOS) використовуються для побудови кодування та визначення порядку розташування даних при стисненні. Наприклад, значення пікселів розташовуються за допомогою зигзагоподібного порядку, що дозволяє компактно представити DC- та AC-коефіцієнти (рис. 1.16).

Зигзагоподібний порядок							
0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Рисунок 1.16. Застосування зигзагоподібного порядку байтів

Процес побудови Huffman-дерева на основі таблиць DHT (рис. 1.17) дозволяє декодувати стиснені дані, після чого отримані коефіцієнти можуть бути використані для подальшого відновлення зображення. Зокрема, DC-коефіцієнти кодуються як різниці між послідовними блоками, а AC-коефіцієнти заповнюють решту матриці у визначеному порядку. Після цього здійснюється зворотне дискретне косинусне перетворення (IDCT) для відновлення піксельних значень.

При переході від YCbCr до RGB застосовуються наступні формули:

1. $R = Y + 1.402 \times Cr$;
2. $G = Y - 0.34414 \times Cb - 0.71414 \times Cr$;
3. $B = Y + 1.772 \times Cb$.

Якщо результати виходять за межі діапазону [0, 255], використовуються граничні значення.

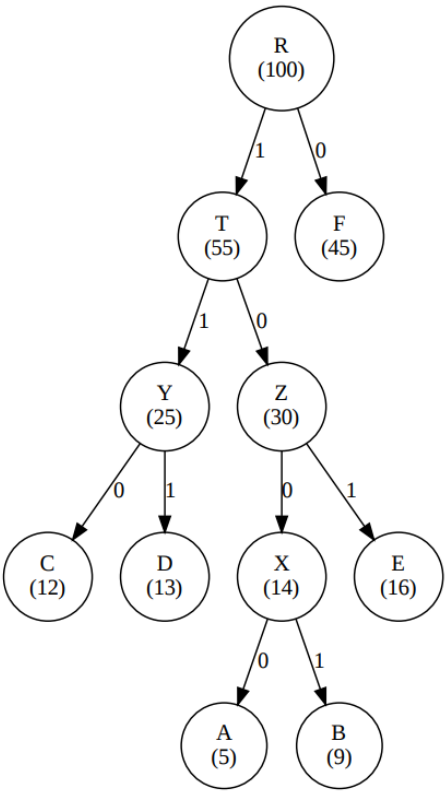


Рисунок 1.17. Застосування дерева Хаффмана

При розробці систем криптографічного захисту зображень важливо враховувати наступне:

- Обробка критичних сегментів: Для правильного шифрування зображення можна окремо обробляти дані заголовків (наприклад, SOF0) і стиснені дані (сегмент SOS). Збереження важливих метаданих дозволяє після дешифрування правильно відновити параметри зображення;
- Врахування втрат: Оскільки JPEG використовує стиснення з втратою, будь-які криптографічні операції повинні орієнтуватися на збереження якості відновленого зображення. Наприклад, застосування шифрування до окремих блоків коефіцієнтів DCT може бути оптимальним, якщо забезпечено мінімальне впливання на відновлення яскравості та кольорових компонентів;
- Розподіл даних за компонентами: Перетворення з RGB у YCbCr розділяє зображення на компонент яскравості і кольору. Це дозволяє реалізувати селективне шифрування – наприклад, зашифрувати лише AC-коефіцієнти або окремі блоки, що містять дані, де незначна зміна не вплине сильно на візуальне сприйняття.

1.3 Реалізація середньоарифметичної фільтрації для захисту зображень

1.3.1 Виконання середньоарифметичної фільтрації

Середньоарифметична фільтрація — один із найпоширеніших методів обробки зображень, що застосовується для усунення імпульсних шумів і покращення візуальної якості. Класична процедура полягає у наступному:

- Сканування матриці зображення: Зображення представлено у вигляді матриці Z розміром $k \times h$, де кожен елемент $p_{i,j}$ відповідає значенню яскравості пікселя;
- Використання квадратної апертури: Для кожного пікселя досліджується квадратна область (апертура) розміру $n \times n$ де n — непарне число, наприклад, 3, 5 або 7), центральним елементом якої є поточний піксель;
- Обчислення середнього значення: Центр апертури замінюється значенням середнього арифметичного всіх пікселів усередині блоку:

$$V[i,j] = \frac{1}{n2}\sum_{u=i-\frac{n-1}{2}} i + \frac{n-1}{2}\sum_{v=j-\frac{n-1}{2}}^{j+\frac{n-1}{2}} p_{u,v} \qquad (1.7)$$

Цей метод дозволяє ефективно згладжувати зображення, видаляючи раптові коливання яскравості, що часто є результатом імпульсного шуму.

1.3.2 Впровадження гомоморфного шифрування через перестановку

Для захисту зображень у межах запропонованої системи, до процесу середньоарифметичної фільтрації додається етап гомоморфного шифрування, що ґрунтується на перестановці стовпців матриці. Основна ідея полягає у тому, що порядок стовпців оригінальної матриці $Z = z_1, z_2, \dots, z_h$ шифрується за допомогою секретних таблиць перестановки.

Пряма та зворотна перестановка:

- Пряма перестановка: За допомогою таблиці T_1 кожному стовпцю z_x оригінального зображення призначається нова позиція: $y = T_1(x), \quad \forall, x \in 1,2, \dots, h$. В результаті отримуємо зашифроване зображення Q, в якому стовпці розташовані у новому порядку;

– Зворотна перестановка: Таблиця T_2 забезпечує відновлення первинного порядку, тобто:

$$T_2(T_1(x)) = x \quad \text{або} \quad T_2 \circ T_1 = I \qquad (1.8)$$

де I — одинична функція.

Таблиця прямої перестановки T_1 має вигляд, показаний у табл.1.2.

Таблиця 1.2. Таблиця прямої перестановки

Номер стовпця (x) (Z)	**Нова позиція ($\mathbf{T_1}$(x)) (Q)**
1	3
2	5
3	1
4	4
5	2

Відповідна таблиця зворотної перестановки T_2 матиме таке ж відображення, що гарантує відновлення початкового порядку.

1.3.3 Загальна схема робочого процесу

Запропонована методика реалізації захищеної середньоарифметичної фільтрації зображень містить наступні етапи:

1. Генерація ключа: Користувач випадково генерує таблицю прямої перестановки T_1 (наприклад, за допомогою генератора псевдовипадкових чисел). Відповідно формується таблиця зворотної перестановки T_2 згідно з рівнянням (1.8);

2. Шифрування зображення: Оригінальне зображення Z перетворюється у зашифроване зображення Q шляхом перестановки стовпців згідно з T_1;

3. Передача та віддалена обробка: Зашифроване зображення Q надсилається через хмарний сервіс на віддалену обчислювальну систему, де виконується часткова середньоарифметична фільтрація;

4. Часткова середньоарифметична фільтрація: На віддаленій системі здійснюється фільтрація кожного елемента матриці Q за наступною формулою:

$$D[i,j] = \frac{1}{n2}\sum_{p=i-\frac{n-1}{2}} i + \frac{n-1}{2} Q[p,j], \qquad (1.8)$$

де *D[i,j]* — відфільтроване значення, а *n* — розмір апертури (число елементів

у вертикальному напрямі). Для підвищення точності розрахунків значення D[i,j] обчислюються з використанням чисел з плаваючою точкою.

5. Дешифрування: Після завершення обробки матриця D повертається користувачу. Використовуючи таблицю зворотної перестановки T_2, здійснюється відновлення початкового порядку стовпців, що дозволяє отримати кінцеве відфільтроване зображення V: $V[i,j] = D[i,T_2(j)]$. На краях зображення, де з огляду на достатню кількість сусідніх елементів операцію фільтрації виконати складно, значення копіюються без змін із зображення Q.

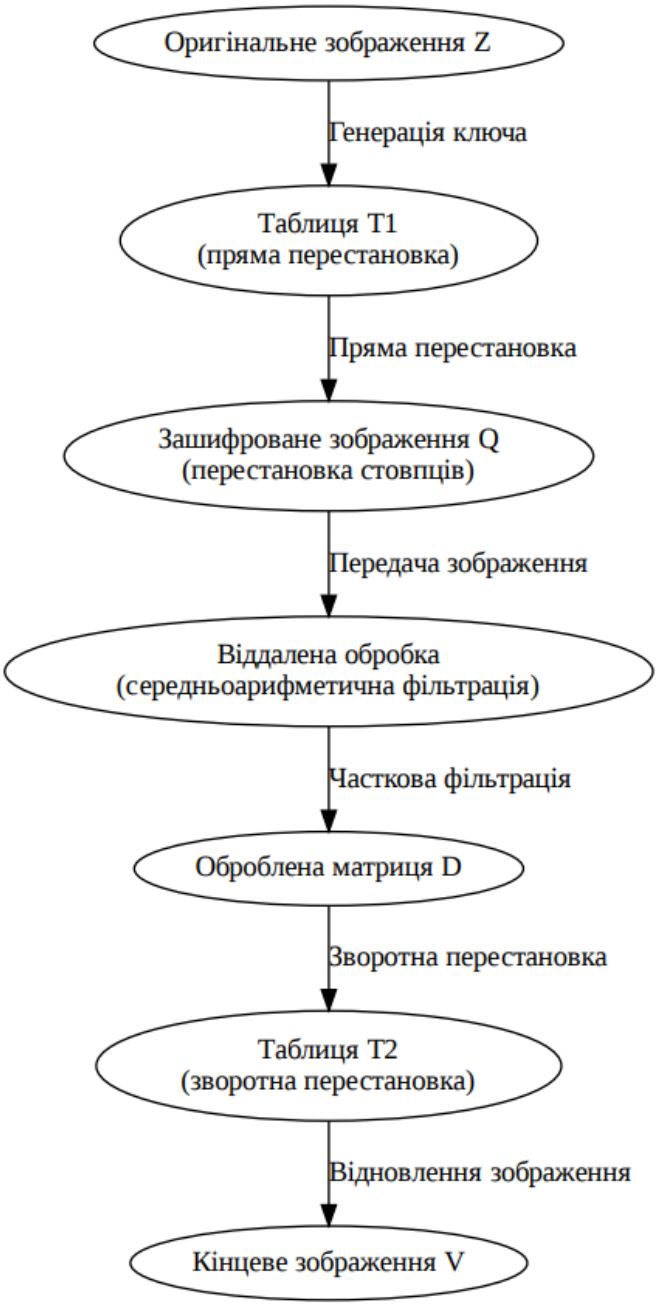


Рисунок 1.18. Загальна організація гомоморфного шифрування зображень при їх захищеній середньоарифметичній фільтрації

На рис.1.18 подано схему, що ілюструє послідовність дій від оригінального зображення *Z* до кінцевого відфільтрованого зображення *V*.

Первинне зображення *Z* перетворюється у зашифровану форму *Q* шляхом перестановки його стовпців відповідно до певного алгоритмічного порядку. Сам порядок перестановки фактично виконує роль ключа гомоморфного шифрування. Після цього зашифроване зображення *Q* надсилається до хмарної інфраструктури, яка розподіляє завдання обробки на віддалених комп'ютерних системах. На віддаленій системі здійснюється часткова середньоарифметична фільтрація *Q*, а результати обробки повертаються користувачу у вигляді матриці *T*. Далі користувач виконує гомоморфне дешифрування, застосовуючи зворотну перестановку стовпців та остаточне обчислення, що завершує етап середньоарифметичної фільтрації. Узагальнено, схема організації розробленого методу представлена на рис. 1.19.

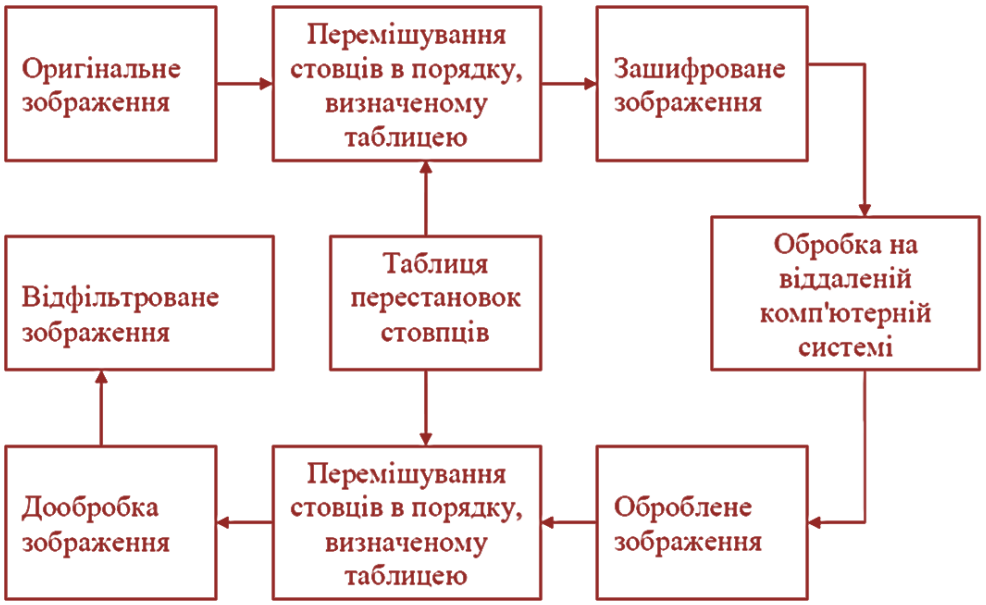


Рисунок 1.19. Реалізація шифрування зображень
з захищеною середньоарифметичною фільтрацією

1.3.4 Реалізація перемішування стовпців при захищеній фільтрації зображень

Особливістю організації захищеної середньоарифметичної фільтрації зображень є можливість паралельної обробки великої кількості точок зображення як в рамках одного зображення, так і одночасно для потоку зображень. При

вирішенні прикладної задачі замість обробки окремого зображення розглядається послідовність зображень: $\Omega = B_1, B_2, B_3, \dots, B_n$. Ця функціональна властивість дозволяє ефективно підвищувати захищеність зображень, оскільки перемішування стовпців може здійснюватися не лише в межах окремого зображення, а й для групи зображень.

Відповідно до розробленого методу гомоморфного шифрування, перемішування стовпців реалізується за допомогою секретної таблиці перестановок. Для підвищення ефективності захищеної фільтрації пропонується виконувати ці перестановки на рівні групи зображень $B_1, B_2, \dots, B_n$. Результатом цього є утворення групи зашифрованих зображень: $U_1, U_2, U_3, \dots, U_n$, які відправляються на віддалені обчислювальні системи для проведення часткової середньоарифметичної фільтрації.

Відмінність даного підходу полягає в тому, що перемішування стовпців здійснюється по широкому спектру зображень. Наприклад, перший стовпець першого зображення може бути розміщений у позицію 233-го стовпця п’ятого зображення. Для відновлення початкового порядку необхідно використати розширену таблицю перестановок T_2', що забезпечує зворотну перестановку стовпців для всієї групи.

Нехай кожне зображення B_i представлено матрицею розміру $k \times h$, де h – кількість стовпців. При гомоморфному шифруванні групи зображень виконується наступне:

1. Створення ключової таблиці T_1 для групи: Таблиця T_1 визначає нове розташування стовпців для всіх зображень одразу: $Q = q_1, q_2, \dots, q_{h \cdot n}$, де кожен елемент q_j відповідає певній позиції з початкової матриці певного зображення;

2. Гомоморфне шифрування групи зображень: Поміщаючи стовпці з усіх зображень за новим порядком, формується група зашифрованих зображень $U_1, U_2, \dots, U_n$. Цей процес секретного перемішування не вимагає додаткових обчислювальних ресурсів і може бути виконаний паралельно із передачею даних;

3. Віддалена фільтрація: На віддалених комп’ютерних системах здійснюється часткова середньоарифметична фільтрація кожного зображення U_i

за викладеними раніше алгоритмічними процедурами. В результаті отримуються відфільтровані зображення $Z_1, Z_2, \dots, Z_n$;

4. Гомоморфне дешифрування: Після отримання відфільтрованих зображень користувач виконує зворотну перестановку стовпців із використанням розширеної таблиці T_2', що відновлює початковий порядок. Результатом є група остаточно оброблених зображень: $R_1, R_2, \dots, R_n$.

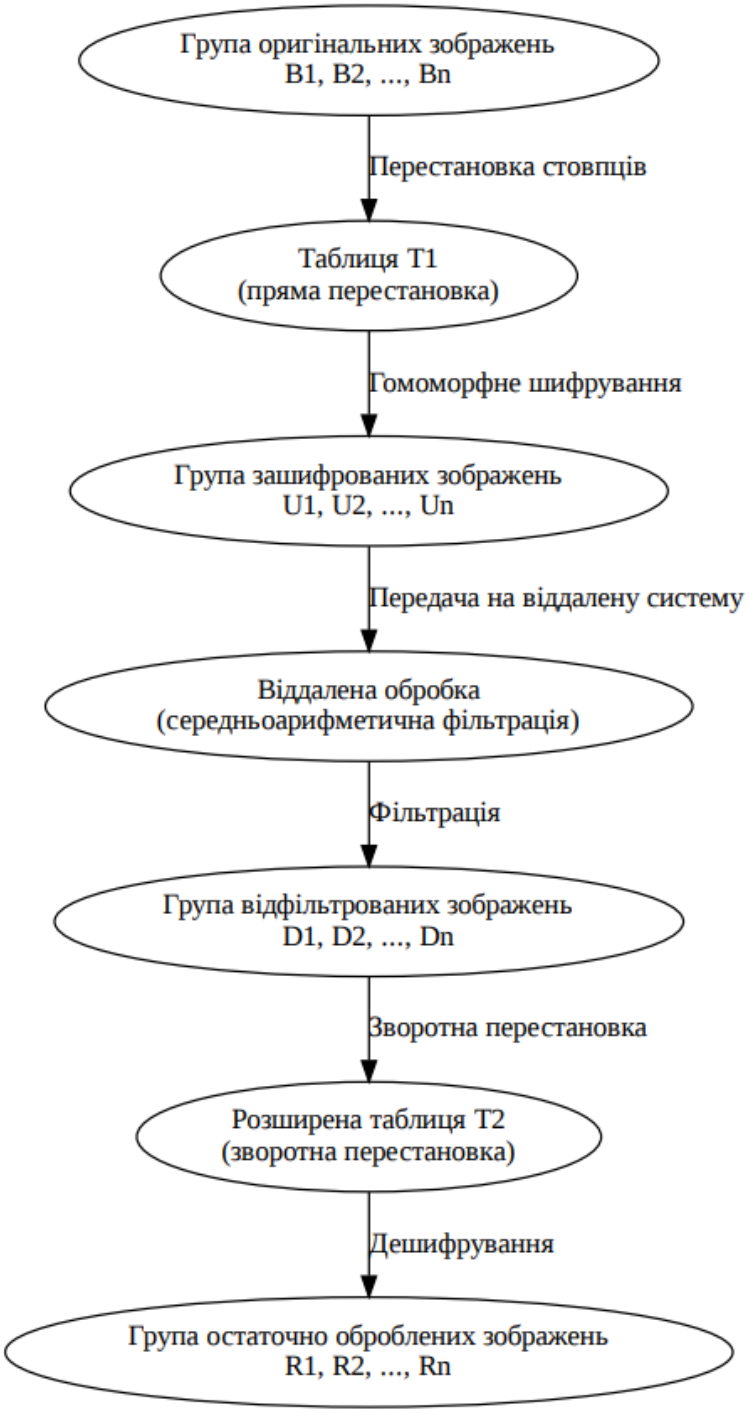


Рисунок 1.19. Схема послідовності операцій перемішування стовпців при захищеній фільтрації групи зображень

Нехай матриця оригінального зображення B має вигляд: $B = z_1, z_2, \dots, z_h$, де $z_j = p_{1,j}, p_{2,j}, \dots, p_{k,j}$. При груповому перемішуванні формуються зашифровані матриці U згідно з таблицею T_1: $U = \mathcal{P}\mathrm{T}_1(\mathrm{B})$. Після віддаленої обробки отримуємо матрицю D (або групу матриць $\mathrm{D}_1, \mathrm{D}_2, \dots, \mathrm{D}_\mathrm{n}$), для якої проводиться зворотна перестановка: $\mathrm{R} = \mathcal{P}T_2'(D)$, де $\mathcal{P}$ позначає операцію перестановки, а T_2' задовольняє умові: $T_2' \circ T_1 = I$. На рис.1.19 наведено схему, що ілюструє послідовність операцій перемішування стовпців при захищеній фільтрації групи зображень. Переваги групового перемішування:

- Покращення ефективності: Обробка групи зображень дозволяє використовувати паралельне розподілення завдань із середньоарифметичної фільтрації, що зменшує час загальної обробки;
- Підвищення безпеки: Перестановки здійснюються над групою зображень, що розширює простір можливих варіантів (при n) зображеннях простір перестановок збільшується до $(u \cdot n)!$;
- Гнучкість реалізації: Таблиці T_1 і T_2' можуть генеруватися псевдовипадково, що дозволяє адаптувати метод до різних вимог безпеки без зміни базового алгоритму.

1.3.5 Аналіз ефективності використаного алгоритму

Ефективність запропонованого алгоритму оцінюється за двома основними критеріями:

1. Продуктивність (прискорення обчислень). Оскільки етап гомоморфного шифрування в даному методі реалізується шляхом перестановки стовпців матриці зображення, обчислювальне навантаження на обчислювальну платформу користувача практично не зростає. Фактично, процес шифрування (та відповідне дешифрування за допомогою зворотного перемішування) є класичним приближенням, що не потребує спеціальних обчислювальних ресурсів і може бути суміщеним із процесом передачі даних.

Під час завершального етапу фільтрації, після повернення частково відфільтрованого зображення з віддаленої системи, користувач виконує наступні

операції:

- Для більшості точок (за винятком невеликої кількості, що становлять менше 0.05% елементів) виконується по одній операції додавання та одній операції віднімання;
- Для елементів першого (не крайнього) стовпця виконується лише операція додавання.

Таким чином, якщо позначити:

- k – кількість рядків матриці,
- u – кількість стовпців,
- t_a – час виконання однієї операції додавання (або віднімання) з плаваючою точкою,

то час обробки користувачем для завершальної фази фільтрації можна наближено записати як:

$$T = k \cdot u \cdot 2 \cdot t_a \qquad (1.9)$$

За довідковими даними, операція ділення з плаваючою точкою в середньому виконується приблизно у 40 разів повільніше за операцію додавання, що дозволяє врахувати домінуючу частину обчислювальних витрат на віддаленій системі. Аналіз показує, що в рамках запропонованого методу питомий обсяг операцій, який виконується на обчислювальній платформі користувача, становить близько 4% від загального обчислювального навантаження, а решта 96% операцій реалізується за допомогою віддалених комп'ютерних систем. Завдяки можливостям розпаралелювання у хмарних обчислювальних системах (навіть при залученні сотень процесорів) час виконання віддаленої частини практично мінімізується;

2. Рівень захищеності. Для аналізу безпеки методу розглядається загальна кількість можливих варіантів перестановок стовпців. Якщо вихідне зображення має u стовпців, то кількість можливих перестановок визначається як:

$$N = u! \qquad (1.10)$$

Припустимо, що для зображень мінімальна кількість стовпців становить u = 1024. Використовуючи наближення за допомогою формули Стірлінга, кількість

можливих перестановок оцінюється дуже великою, наприклад: $N \approx 10^{2639106}$. Такий величезний розмір простору перестановок практично унеможливлює перебір усіх варіантів для відновлення оригінального зображення методом “brute-force”, навіть якщо врахувати перспективи використання квантових комп’ютерів протягом наступних 20–40 років. Крім того, якщо обробляти групи з n зображень, то простір перестановок збільшується до:

$$N_{\text{group}} = (u \cdot n)! \qquad (1.11)$$

що додатково підсилює рівень захищеності даного методу.

Порівняно з іншими методами спеціалізованого гомоморфного шифрування, такими як маскове шифрування або адитивне шифрування, запропонований метод має значну перевагу з точки зору як продуктивності, так і витрат на обчислення користувацькою платформою. Основні переваги можна підсумувати наступним чином:

- Низькі обчислювальні витрати на стороні користувача. Процес гомоморфного шифрування (перестановка стовпців) здійснюється практично миттєво та може бути суміщений із передачею даних;
- Високий рівень захищеності. Величезний простір можливих перестановок (наприклад, $10^{2639106}$ варіантів для зображення з 1024 стовпцями) забезпечує майже абсолютну стійкість до атак перебором;
- Оптимізація за рахунок розподілених обчислень. Основна частина обчислень виконується на віддалених високопотужних комп’ютерних системах із можливістю розпаралелювання, що значно прискорює процес середньоарифметичної фільтрації.

Аналіз ефективності показує, що запропонований метод гомоморфного шифрування зображень із захищеною середньоарифметичною фільтрацією має наступні ключові переваги:

- Продуктивність: Завдяки розподіленій обробці лише 4% операцій виконується на користувацькій платформі, що забезпечує значне прискорення загального процесу;
- Захищеність: Величезний простір перестановок, що забезпечується

використанням секретних таблиць T_1 та T_2, гарантує високий рівень захисту від несанкціонованого відновлення оригінального зображення.

Для оцінки ефективності запропонованої модифікації методу гомоморфного шифрування груп зображень при їх середньоарифметичній фільтрації на віддалених обчислювальних потужностях доцільно враховувати два основні критерії:

1. Рівень захищеності зображень. Оскільки зашифровані зображення передаються через потенційно відкриті канали передачі (Інтернет), а обробка виконується на віддалених комп'ютерних системах, які можуть бути недоступними для користувача, ключовим показником є складність відновлення вихідного порядку стовпців, що використовується як секретний ключ;

У запропонованому методі захищеність забезпечується за рахунок комбінування перестановки (перемішування) та зсувів, що ускладнюють структуру таблиці прямої перестановки ключових елементів. При цьому, якщо розглядати обробку окремого зображення, кількість варіантів відновлення порядку стовпців дорівнює *u!* де *u* – кількість стовпців.

При груповій обробці, коли перемішування здійснюється над n зображеннями, отримуємо додаткове ускладнення – простір можливих перестановок зростає експоненціально, приблизно в *n!* разів, що значно ускладнює задачу перебору для потенційного зловмисника.

Таблиця 1.3. Оцінка рівня захищеності

Кількість зображень n	**Кількість можливих варіантів (приблизно)**
1	$u!$
5	$u! \times 5!$
10	$u! \times 10!$

Навіть за умов мінімального значення *u* та при відносно невеликій кількості зображень, простір варіантів перестановок стає вражаючим і практично унеможливлює перебір для відновлення оригінального порядку.

2. Виграш у швидкодії виконання фільтрації. Основною перевагою запропонованої модифікації є те, що обчислювальні витрати користувача залишаються мінімальними:

- Процес гомоморфного шифрування за рахунок перестановки стовпців у групі зображень не вимагає спеціальних обчислень і може виконуватися одночасно з передачею даних;
- Віддалена обробка (середньоарифметична фільтрація) здійснюється на багатопроцесорних серверах або кластерах, де можливе масштабування та розпаралелювання обчислень. При цьому часові характеристики обробки окремих точок зображення, що здійснюються за допомогою додавання, віднімання та операції ділення для обчислення середнього арифметичного, практично не змінюються незалежно від використання групового режиму.

Таким чином, загальний час виконання фільтрації для групи зображень практично не відрізняється від часового циклу для окремого зображення, що визначається сумарним часом обробки кожної точки. Отже, залучення віддалених обчислювальних потужностей не впливає на затримки в процесі шифрування та дешифрування, а навпаки, дозволяє суттєво скоротити загальний час обчислень за рахунок паралельного виконання.

1.4 Розробка програмної моделі та блок-схем алгоритмів застосунку для захисту зображень

У даному проекті розроблено програмний застосунок для захисту зображень, який реалізовано мовою C# у середовищі Visual Studio. Програма поєднує метод гомоморфного шифрування на основі перестановки стовпців із середньоарифметичною фільтрацією, що дозволяє як підвищувати якість зображення, так і забезпечувати його захищеність. При цьому всі операції виконуються на локальній комп'ютерній платформі, без залучення віддалених обчислювальних ресурсів. Основні компоненти застосунку будуть такими:

1. Модуль введення та попередньої обробки:
   - Вхідні дані: Користувач завантажує оригінальне зображення Z та задає параметри фільтрації (наприклад, розмір апертури $n \times n$ і конфігурації шифрування;
   - Попередня обробка: Підготовка зображення для подальшої обробки, розбиття на піксельні блоки, нормалізація значень;

2. Модуль генерації ключових таблиць та гомоморфного шифрування:

- Генерація таблиці перестановок T_1: На основі генератора псевдовипадкових чисел формується секретна таблиця, яка визначає новий порядок перестановки стовпців оригінального зображення Z. Таким чином, шифрування виконується як перетворення: $Q = \mathcal{P}_{\mathcal{T}_1}(Z)$, де Q – зашифроване зображення;
- Ключова роль: Порядок перестановки стовпців T_1 виконує роль секретного ключа для шифрування. Для відновлення вихідного порядку формується зворотна таблиця T_2, що задовольняє співвідношенню: $T_2 \circ T_1 = I$, де I – одиничне відображення;

3. Модуль локальної середньоарифметичної фільтрації:

- Фільтрація: Зашифроване зображення Q піддається операції середньоарифметичної фільтрації, що здійснюється за формулою: $V[i,j] = \frac{1}{n2}\sum_{u=i-\frac{n-1}{2}} i + \frac{n-1}{2}\sum_{v=j-\frac{n-1}{2}}^{j+\frac{n-1}{2}} Q[u,v]$, де n — розмір квадратної апертури, а V – проміжна матриця з результатами фільтрації;
- Оптимізація: Для прискорення обчислень застосовується рекурсивний алгоритм, який дозволяє обчислювати значення зменшеними обсягами операцій (адитивних операцій додавання та віднімання);

4. Модуль гомоморфного дешифрування та фінальної обробки:

- Дешифрування: Після фільтрації користувач застосовує зворотну перестановку стовпців із використанням таблиці T_2 для відновлення первинного порядку: $V_{\text{final}} = \mathcal{P}_{\mathcal{T}_2}(V)$;
- Остаточна обробка: Для крайніх точок зображення, де повна апертура не доступна, значення копіюються з зашифрованого зображення Q.

На рис.1.20 наведено БСА, що ілюструє послідовність операцій фільтрації зображень у застосунку. Використання методів гомоморфного шифрування через перестановку стовпців з середньоарифметичною фільтрацією забезпечує високий рівень захищеності без значних обчислювальних витрат.

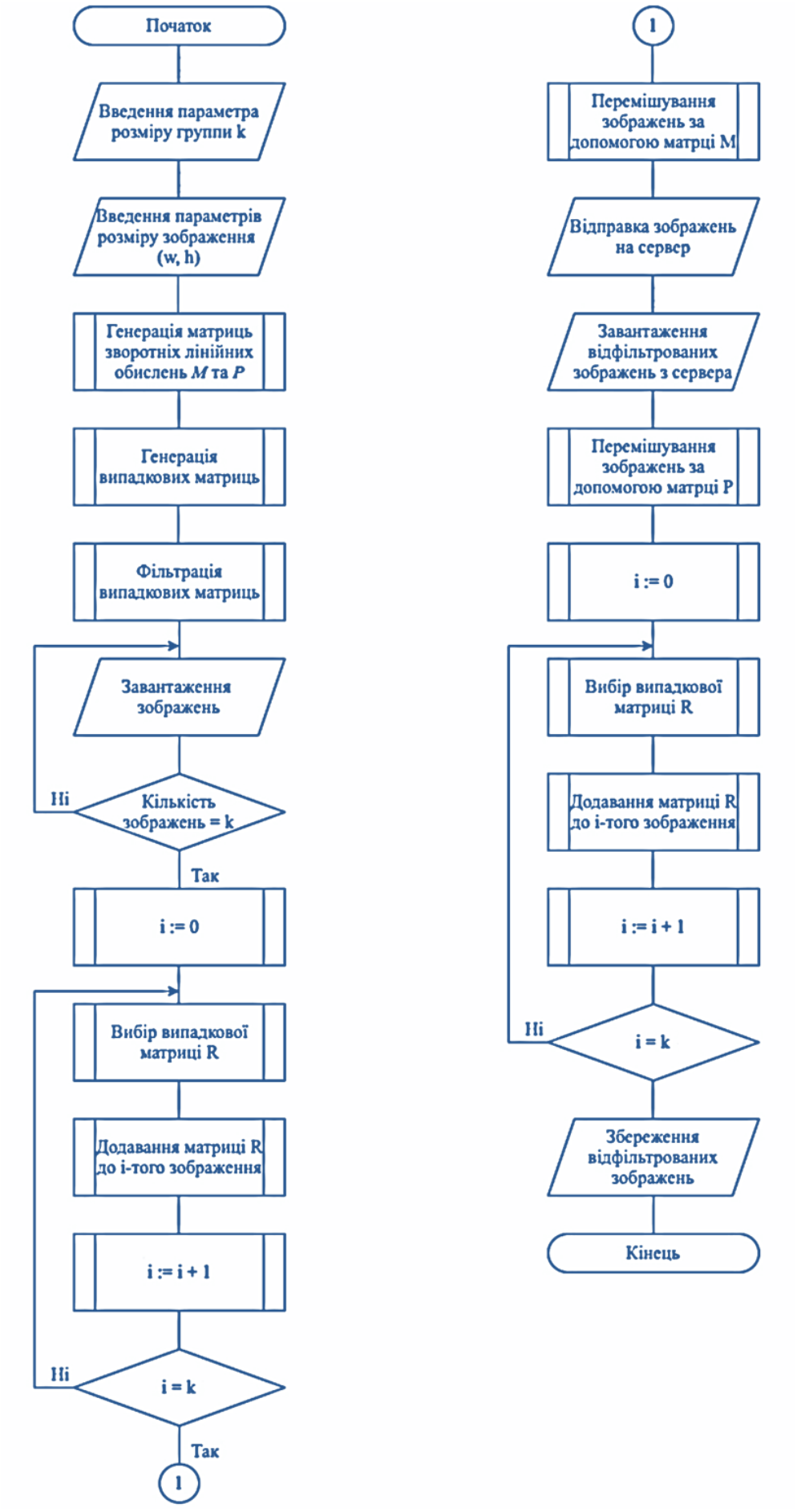


Рисунок 1.20. Блок-схема алгоритму середньоарифметичної фільтрації зображень

Розробка застосунку здійснюється на мові C# у Visual Studio. Основні технології, що використовуються:

- .NET Framework: Для обробки зображень використовується клас System.Drawing та інші бібліотеки для роботи з даними;
- Математичні алгоритми: Реалізовано алгоритми перестановки стовпців (гомоморфне шифрування), середньоарифметичної фільтрації, а також алгоритми зворотної перестановки для дешифрування;
- Інтерфейс користувача: Забезпечується інтуїтивно зрозумілий графічний інтерфейс Windows Forms, що дозволяє користувачу легко завантажувати зображення, налаштовувати параметри шифрування та запускати процес захисту.

Розроблюваний застосунок дозволить користувачу завантажувати зображення (формату PNG), здійснювати вбудоване гомоморфне шифрування та дешифрування на основі перестановки стовпців матриці пікселів, а також застосовувати середньоарифметичну або медіанну фільтрацію для покращення якості зображення. Весь процес відбувається на одній комп'ютерній системі з використанням платформи .NET і мови C#.

Для реалізації ключових елементів алгоритму захисту зображень було виділено наступні основні класи:

1. Matrix. Представляє матричне подання даних із зображення. Основні методи та оператори:
    - fillRandomValues() – заповнення матриці випадковими значеннями (корисно для генерації допоміжних матриць);
    - clone() – клонування матриці;
    - toString() – серіалізація матриці у рядковий вигляд;
    - Оператори додавання, віднімання та множення матриці на число для підтримки арифметичних операцій, необхідних при фільтрації;
2. HSB. Забезпечує конвертацію колірних даних між форматами RGB і HSB. Основні методи:
    - ColorToHSB(Color rgbColor) – конвертує значення пікселя з формату

RGB у формат HSB;

- ColorFromHSB(double[] hsb) – виконує обернену конвертацію з HSB у RGB;

3. Image. Інкапсулює зображення у форматі HSB. Зображення розкладається на три матриці типу Matrix:

- hue
- saturation
- brightness

Основні методи:

- Конструктор, що зчитує PNG-файл із файлової системи та формує відповідні матриці компонент кольорової моделі;
- save() – збереження зображення у форматі Bitmap;
- medianFilterImage() і meanFilterImage() – методи для фільтрації зображення з використанням відповідних класів фільтрів;

4. MedianFilter. Реалізує фільтрацію типу медіанної над об’єктами типу Matrix. Метод process() застосовується для кожного блоку значень матриці;

5. MeanFilter. Виконує середньоарифметичну фільтрацію над матрицями. Також має метод process(), який обчислює середнє значення елементів блоків;

6. MedianMatrixEncryptor. Займається шифруванням та дешифруванням матриць типу Matrix для алгоритмів, що базуються на медіанні фільтрації. Основні методи:

- encryptMatrix(Matrix matrix, out Dictionary<double, double> encryptedKey);
- decryptMatrix(Matrix encryptedMatrix, Dictionary<double, double> encryptedKey);

7. MedianImageEncryptor. Клас, що дозволяє шифрувати та дешифрувати зображення типу Image з використанням алгоритмів медіанного шифрування. Основні методи:

- encryptImage(Image image, out Dictionary<double, double[]> encryptedKey);

– decryptImage(Image image, Dictionary<double, double[]> encryptedKey);

8. MeanGroupMatrixEncryptor. Реалізує шифрування та дешифрування групи матриць типу Matrix для середньоарифметичної фільтрації. Клас включає генерацію випадкових допоміжних матриць, їх обробку (за допомогою MeanFilter) та формування ключа для дешифрування. Основні методи:

– encryptMatrices(Matrix[] matrices, out int[] key);

– decryptMatrices(Matrix[] encryptedMatrices, int[] key);

9. MeanImageEncryptor. Адаптований для обробки зображень типу Image, інтегруючи функціонал групового шифрування. Основні методи:

– encryptImage(Image image, out Dictionary<double, double[]> keys);

– decryptImage(Image image, Dictionary<double, double[]> keys).

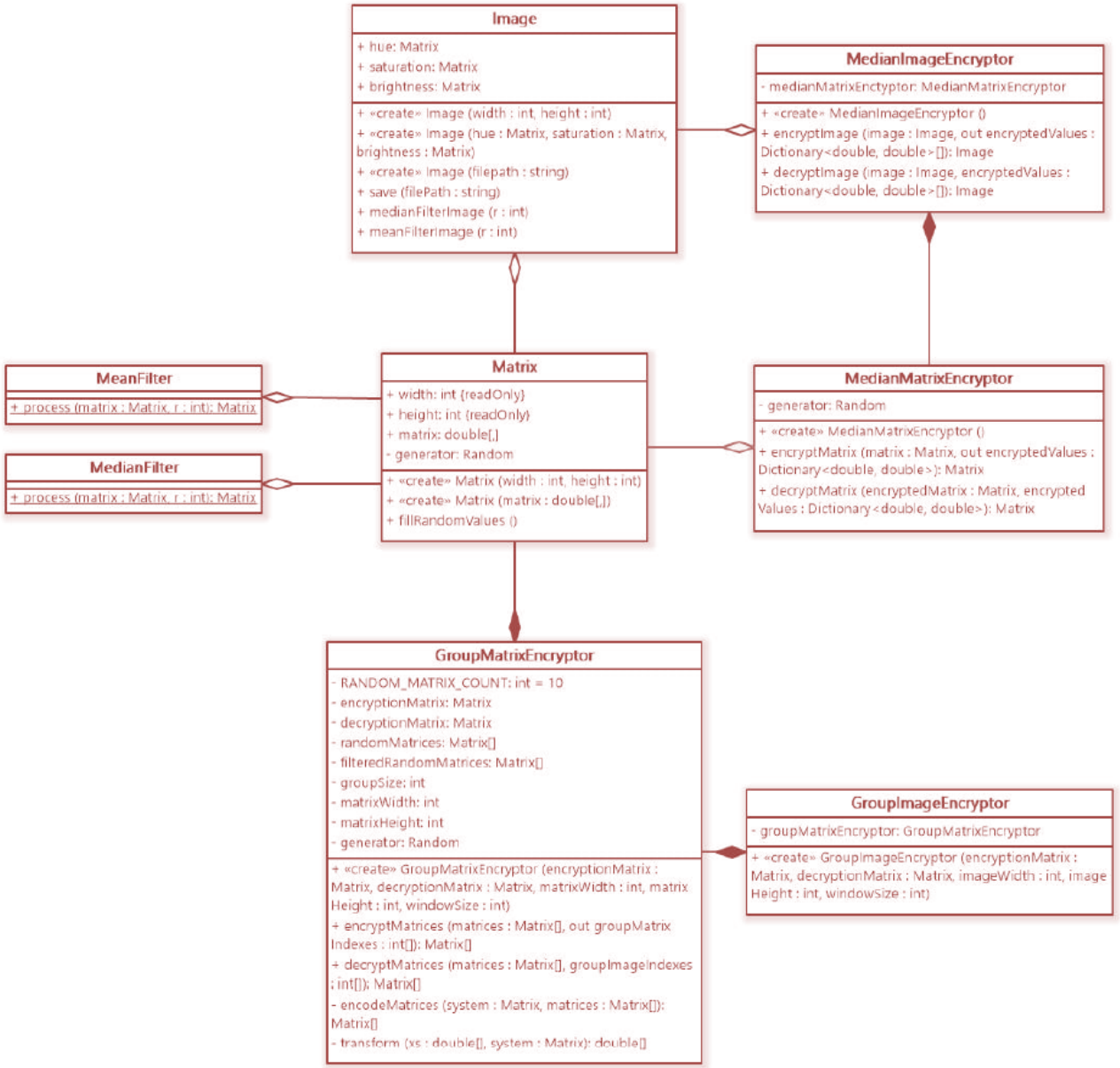


Рисунок 1.21. Діаграма класів застосунку для захисту зображень

Зв’язки класів є такими:

- Клас Image використовує об’єкти типу Matrix для зберігання кожної з компонент кольорової моделі (hue, saturation, brightness);
- HSB забезпечує конвертацію значень між форматами, що використовуються у класі Image;
- Фільтри (MedianFilter та MeanFilter) застосовуються як класами, так і методами класу Image для покращення якості зображення;
- Шифрування та дешифрування реалізуються спеціалізованими класами-енкрипторами (MedianMatrixEncryptor, MedianImageEncryptor, MeanGroupMatrixEncryptor, MeanImageEncryptor), які використовують операції над об’єктами Matrix або Image та відповідні таблиці перестановок (ключовий аспект гомоморфного шифрування, де порядок перестановки стовпців є секретом).

На рис.1.21 подано діаграму класів у вигляді схематичного представлення. Цей варіант використано для подальшої розробки застосунку:

1. Завантаження зображення: Користувач завантажує PNG-зображення через інтерфейс застосунку. Клас Image зчитує файл, перетворює піксельні дані з формату RGB у формат HSB (за допомогою класу HSB) та розкладає його на три матриці (hue, saturation, brightness);

2. Шифрування: Метод шифрування, реалізований у класах-енкрипторах (наприклад, MedianImageEncryptor або MeanImageEncryptor), здійснює перестановку стовпців кожної матриці відповідно до секретної таблиці ключ T_1. Ця операція забезпечує гомоморфне шифрування, оскільки порядок стовпців є секретним і може бути відновлений лише за допомогою відповідного дешифрувального ключа T_2;

3. Фільтрація: Після шифрування зображення опрацьовується методом середньоарифметичної або медіанної фільтрації. Для цього застосовуються класи MeanFilter або MedianFilter, які виконують операції згладжування над матрицями;

4. Дешифрування та відновлення зображення: Після фільтрації, клас-енкриптор виконує зворотну перестановку (за таблицею T_2), відновлюючи

початковий порядок стовпців, що дозволяє отримати кінцеве захищене та відфільтроване зображення.

Діаграма класів (рис.1.21) відображає модульну архітектуру застосунку, де окремі класи відповідають за різні функції: обчислення над матрицями, конвертацію кольорів, обробку зображень, фільтрацію, а також гомоморфне шифрування та дешифрування. Такий підхід дозволяє легко модифікувати окремі компоненти та забезпечити високий рівень захищеності за рахунок використання секретних таблиць перестановок.

У окремому модулі захисту AES реалізовано функціонал для шифрування та дешифрування зображень після виконання гомоморфного шифрування. Ключова ідея – забезпечити безпечне перетворення зображень у форматі PNG з використанням симетричного шифрування, де як ключ, так і вектор ініціалізації (IV) зберігаються у текстових файлах у форматі Base64. Це дозволяє розділити процес створення ключів від їх використання, забезпечуючи гнучкість та безпеку їх зберігання при розробці застосунку.

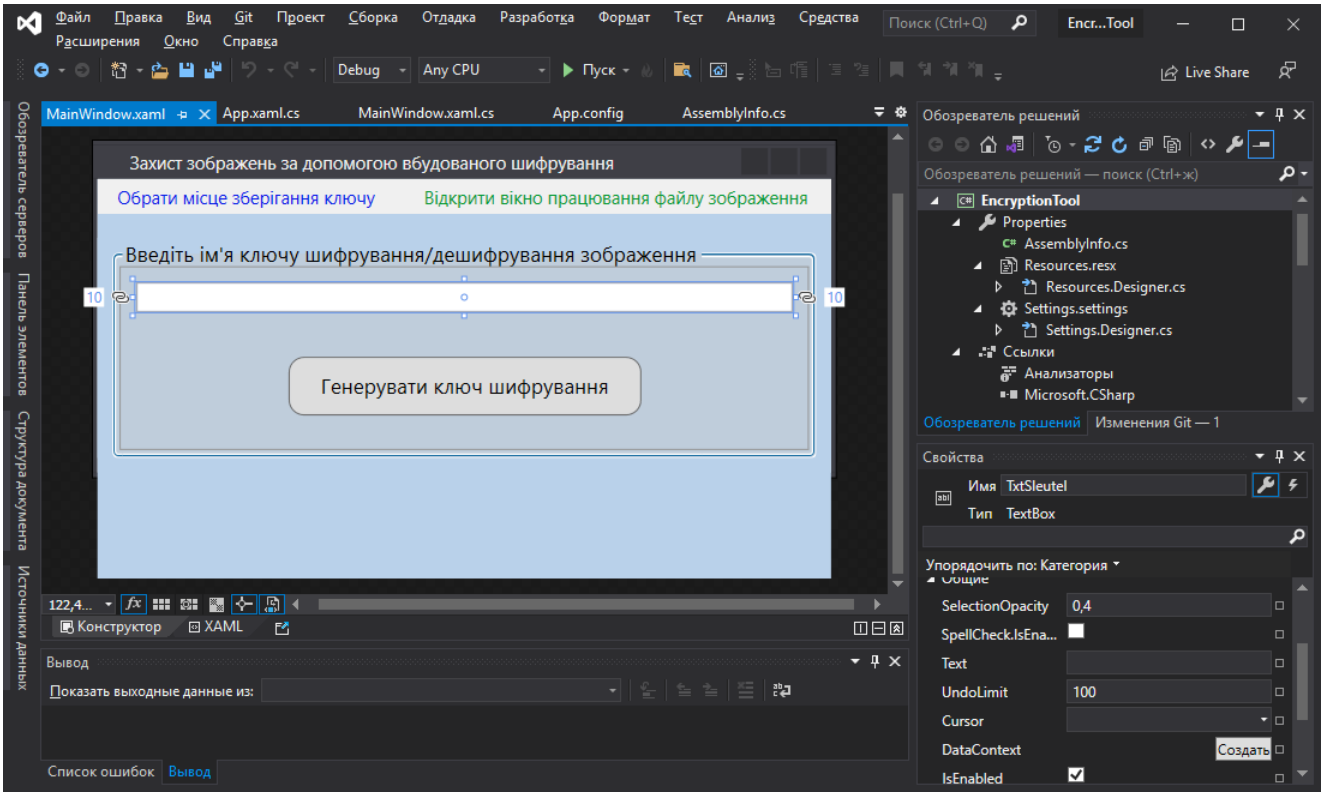


Рисунок 1.22. Етап розробки головної форми застосунку у Visual Studio

Код застосунку написано мовою C# із застосуванням Windows Presentation

Foundation (WPF) для створення користувацького інтерфейсу. Інтерфейс організовано у вигляді окремого вікна, яке є частковою реалізацією (partial class) та інтегрує елементи WPF (для відображення зображень, кнопок, текстових полів) з елементами Windows Forms (наприклад, MessageBox та OpenFileDialog). Такий підхід забезпечує можливість гнучкого вибору папок для зберігання зашифрованих, дешифрованих та вихідних зображень, що значно підвищує зручність застосунку для кінцевого користувача.

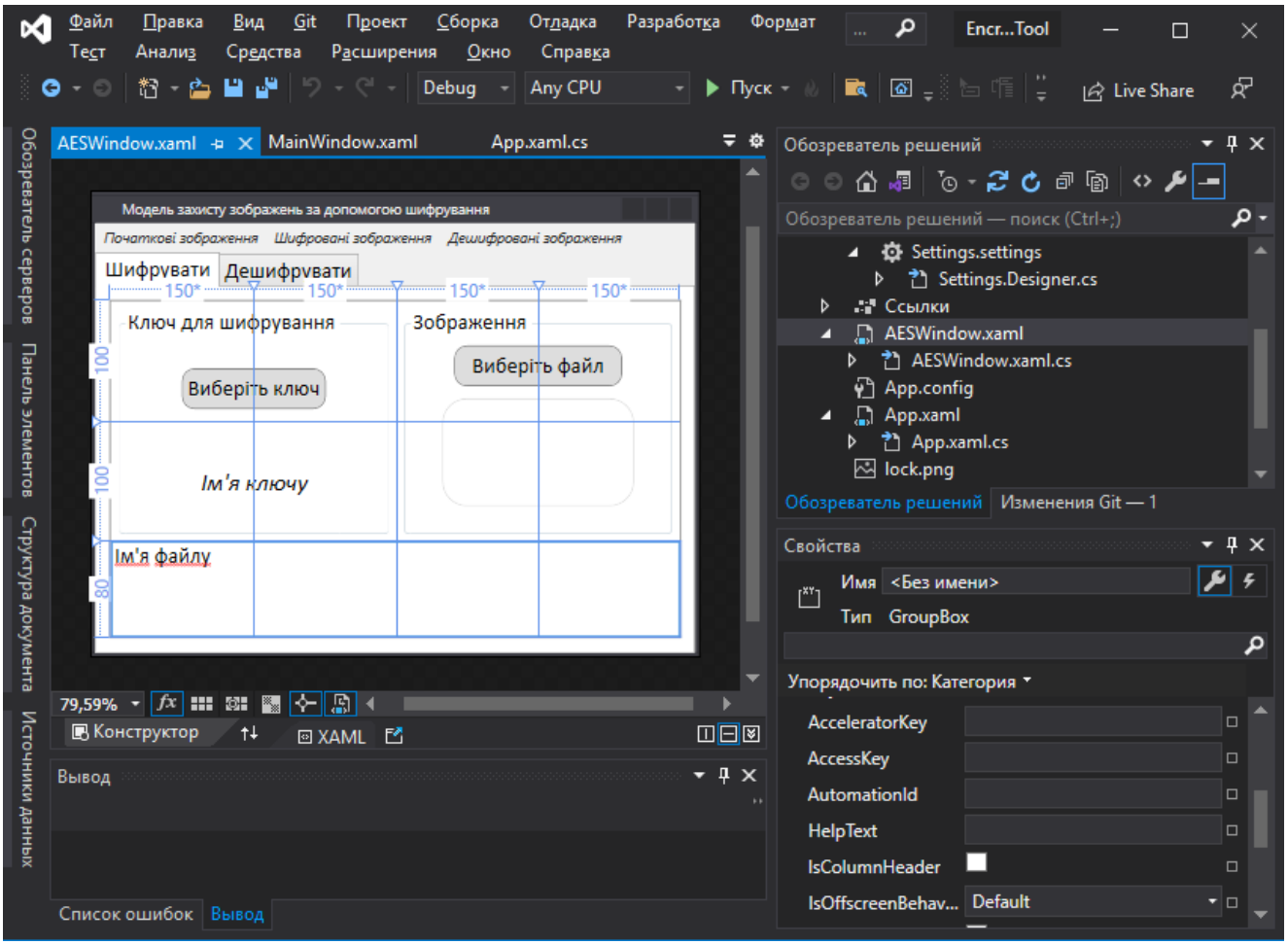


Рисунок 1.23. Етап розробки дочірнього вікна та вкладки “Шифрувати”

Функції BtnKeyInlezenEncrypt_Click та BtnKeyInlezenDecrypt_Click відповідають за завантаження відповідних ключів і векторів ініціалізації. Результуючі тексти з файлів спочатку читаються як рядки, після чого виконуються перетворення в масиви байтів за допомогою Convert.FromBase64String. Використання формату Base64 дозволяє зберігати двійкові значення ключа та IV у зручному для зберігання та передачі текстовому форматі, що значно спрощує їхній подальший обмін та архівування.

При зашифруванні зображення відбувається кілька критичних операцій:

- Зображення після попереднього гомоморфного шифрування на основі перестановки стовпців із середньоарифметичною фільтрацією зчитується як послідовність байтів;
- Створюється екземпляр класу AES із заданими параметрами (KeySize та BlockSize), після чого встановлюється ключ і вектор ініціалізації;
- За допомогою CryptoStream байти зображення шифруються "на льоту" та записуються у MemoryStream. Це дозволяє оперативно обробляти дані без необхідності записувати їх у тимчасові файли;
- Отримані зашифровані байти записуються у новий файл із розширенням .encrypted, що гарантує відокремлення зашифрованих даних від вихідних.

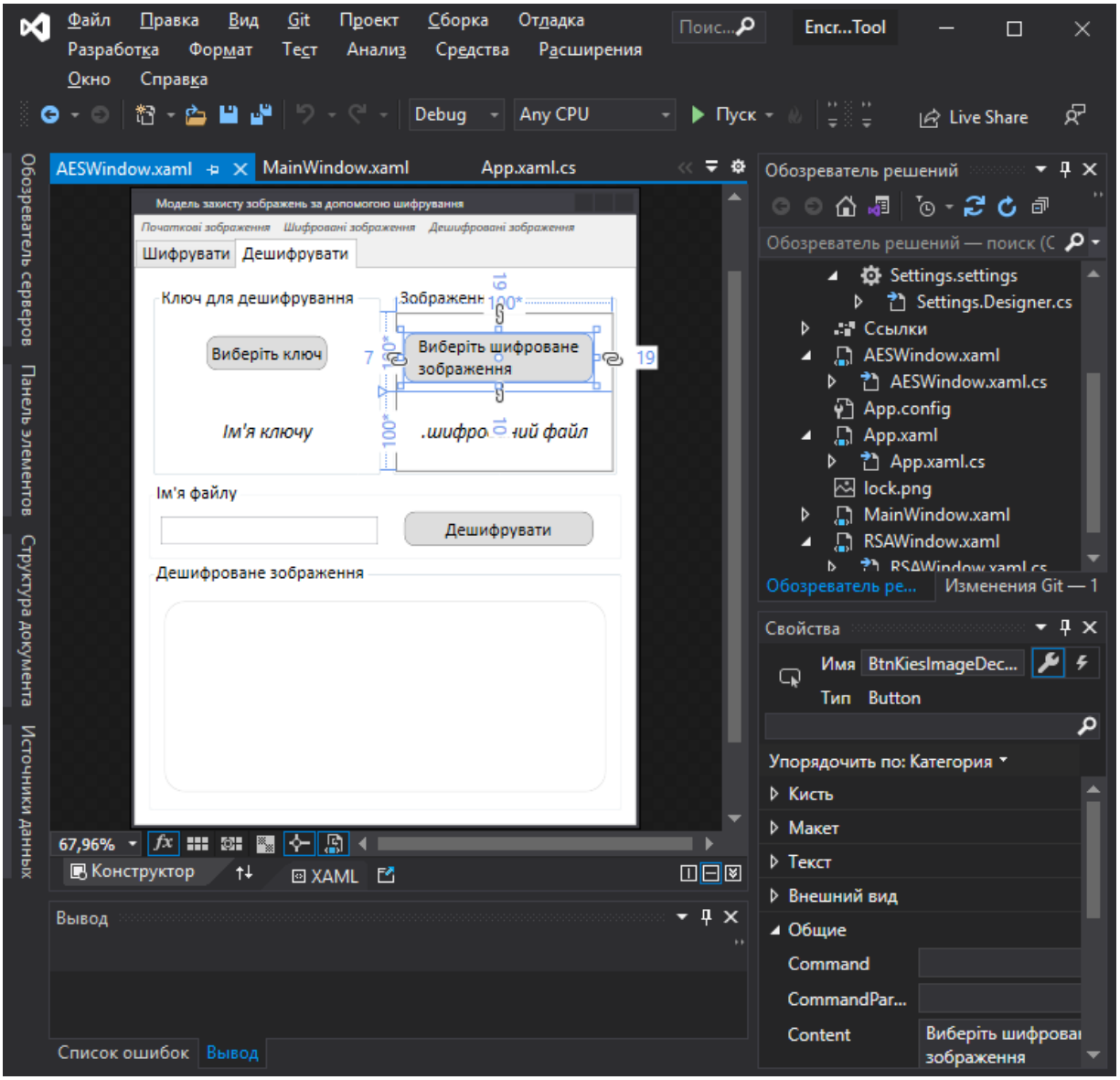


Рисунок 1.24. Етап розробки дочірнього вікна та вкладки “Дешифрувати”

Код активно використовує подієво-орієнтовану модель, притаманну WPF. Це дозволяє реагувати на дії користувача (натискання кнопок, вибір файлів) через відповідні обробники подій. Наприклад, функції для вибору шляху до папок (методи MenuPlainImagesStorage_Click, MenuEncryptedImagesStorage_Click та MenuDecryptedImagesStorage_Click) полегшують користувачу встановити необхідні директорії для роботи з файлами. На рис.1.22-1.24 наведено скріншоти етапів розробки застосунку, показано створення форм головного та дочірнього вікна з відповідними елементами, що забезпечують керування процесом створення ключа шифрування, приховування та відновлення зображення.

1.5 Аналіз роботи застосунку та моделі захисту зображень у форматі PNG

При старті програми користувачу одразу відображається модальне вікно повідомлення, що має простий та інтуїтивно зрозумілий дизайн, відповідний стандартним засобам оформлення Windows. Центральним елементом вікна є текстове повідомлення, яке інформує користувача про базову конфігурацію системи за замовчуванням (рис.1.25).

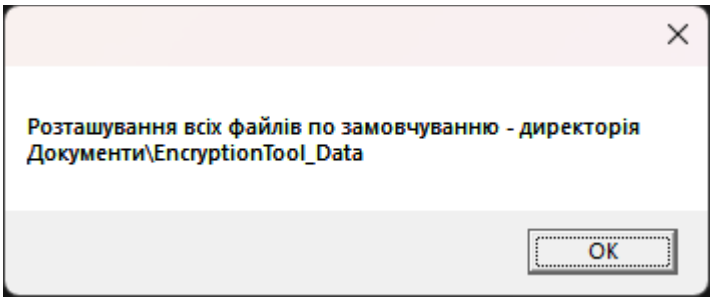


Рисунок 1.25. Вікно повідомлення про розташування файлів по замовчуванню

Таке повідомлення дає зрозуміти, що усі файли, з якими працюватиме програма (ключі шифрування, зображення для шифрування чи дешифрування, а також результати операцій), зберігатимуться у визначеній директорії. Це спрощує подальшу взаємодію користувача із файловою системою, дозволяючи одразу орієнтуватися, де шукати необхідні дані чи зберігати результати роботи програми.

Головне вікно є центральним пунктом управління у програмі. Саме з нього користувач отримує доступ до всіх основних функцій, що стосуються обробки та захисту зображень. Інтерфейс спроектовано так, щоб кожна операція виконувалася

максимально інтуїтивно: від запуску дочірнього вікна для шифрування/дешифрування до вибору директорії для зберігання ключів і генерації самої ключової інформації. Основні елементи інтерфейсу:

1. Кнопка «Відкрити вікно опрацювання файлу зображення»: Натискання цієї кнопки відкриває дочірнє вікно (рис.1.28), де користувач може завантажити зображення, провести його обробку та виконати операції шифрування/дешифрування. У цьому вікні застосовується комбінована технологія – спочатку зображення піддається попередній обробці за допомогою гомоморфного шифрування (перестановка стовпців із середньоарифметичною фільтрацією), а далі результуючі дані шифруються алгоритмом AES. Це дозволяє забезпечити високий рівень конфіденційності та складність для потенційних атак;

2. Кнопка «Обрати місце зберігання ключу»: При натисканні відкривається діалогове вікно (за допомогою стандартного компонента FolderBrowserDialog), де користувач може вказати бажану директорію для зберігання текстового файлу з згенерованим криптографічним ключем. Завдяки такому підходу, користувачу надається можливість самостійно контролювати місце зберігання ключових даних, що є важливим аспектом безпеки в системах криптографічного захисту;

3. Кнопка «Генерувати ключ шифрування»: Після натискання кнопки застосунок генерує криптографічний ключ (з можливим використанням алгоритмів, що забезпечують достатню стійкість) і записує його у текстовий файл. При цьому, збереження файлу відбувається у раніше обраній користувачем директорії, що гарантує централізоване управління ключами. Даний механізм дозволяє оперативно створювати нові ключі для шифрування, що може бути корисним при регулярній зміні криптографічних параметрів або при оновленні системи захисту даних. При успішному створенні ключу з’являється відповідне повідомлення (рис.1.27);

4. Текстове поле для введення імені ключа: Це поле дозволяє користувачу задати унікальне ім’я для згенерованого ключа. Введення імені ключа допомагає організувати файли з ключами, роблячи їх відновлення та подальше використання зручним та структурованим.

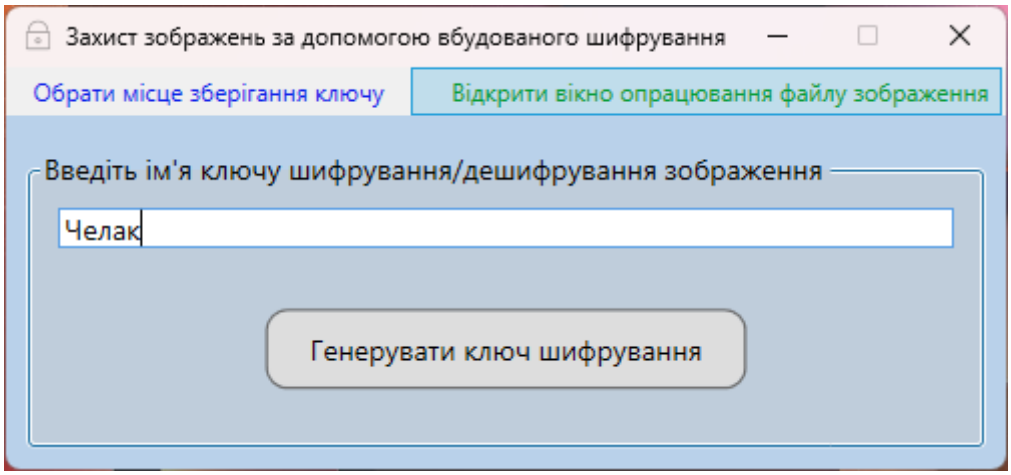


Рисунок 1.26. Головне вікно з доступом до основних функцій застосунку

При запуску застосунку головне вікно виконує роль стартової платформи. Звідти користувач може негайно перейти до конкретних задач, таких як обробка файлів зображень або управління криптографічними ключами. Натискання кнопки «Відкрити вікно опрацювання файлу зображення» спрямовує користувача до дочірнього вікна, де вже реалізовано комплексну обробку зображень із застосуванням комбінованого шифрування. Опція «Обрати місце зберігання ключу» забезпечує можливість зміни директорії за замовчуванням, що використовуються для збереження текстового файлу з ключем, що підвищує гнучкість роботи програми. Натискання кнопки «Генерувати ключ шифрування» запускає процес створення нового ключа, який одразу зберігається у вказаному місці, що надалі використовується для шифрування зображень.

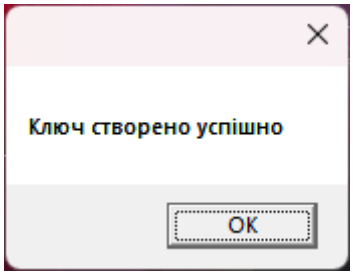


Рисунок 1.27. Вікно повідомлення про успішне створення ключу

В верхній частині дочірнього вікна (рис.1.28) розташовано кілька вкладок, що забезпечують розподіл функціональності за етапами обробки:

- Початкові зображення: Тут відображаються оригінальні файли до застосування будь-яких алгоритмів шифрування;
- Зашифровані зображення: Ця вкладка призначена для перегляду результатів

шифрування;

- Дешифровані зображення: Дозволяє бачити відновлені (дешифровані) зображення після процесу дешифрування.

У вікні передбачено спеціальний блок для роботи з ключем:

- Блок "Ключ для шифрування" містить кнопку для завантаження або вибору існуючого криптографічного ключа.
- Поруч розміщене текстове поле, яке відображає назву поточного ключа. Це дозволяє переконатися, що використовується правильний ключ для зашифрування чи дешифрування.

Для роботи з зображенням є окрема секція:

- За допомогою кнопки «Виберіть файл» користувач може відкрити діалогове вікно вибору файлу, яке дозволяє завантажити потрібне зображення;
- Після вибору файлу поруч або в окремій панелі відображається попередній перегляд зображення, що гарантує правильність вибору.

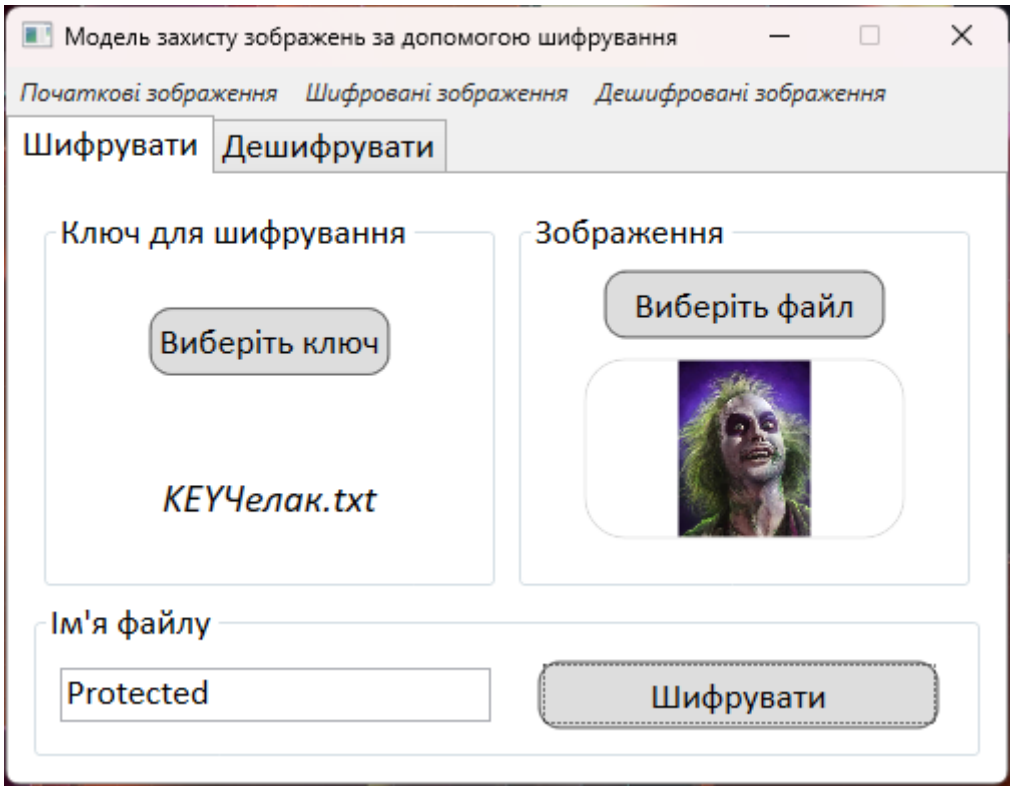


Рисунок 1.28. Дочірнє вікно з вкладками шифрування/дешифрування зображення

У вікні передбачено текстове поле із заголовком «Ім'я файлу», де користувач може задати бажану назву для вихідного файлу після обробки. Це важливо для

збереження порядку та ідентифікації результатів роботи програми. Ключовими елементами управління є дві основні кнопки:

- «Шифрувати»: При натисканні на цю кнопку запускається процес шифрування обраного зображення. На першому етапі може виконуватися попередня обробка (за допомогою гомоморфного шифрування з перестановкою стовпців та середньоарифметичною фільтрацією), після чого оброблені дані піддаються симетричному шифруванню AES. При відсутності текстового файлу ключу у заданій директорії буде видане відповідне повідомлення (рис.1.29). У випадку наявності файлу ключу, початкового зображення у форматі PNG та заданого імені вихідного зашифрованого файлу буде видане повідомлення про успішне шифрування (рис.1.30);
- «Дешифрувати»: Ця кнопка активує зворотний процес, що дозволяє відновити оригінальне зображення із зашифрованого файлу відповідно до отриманого шифр-ключа та параметрів дешифрування.

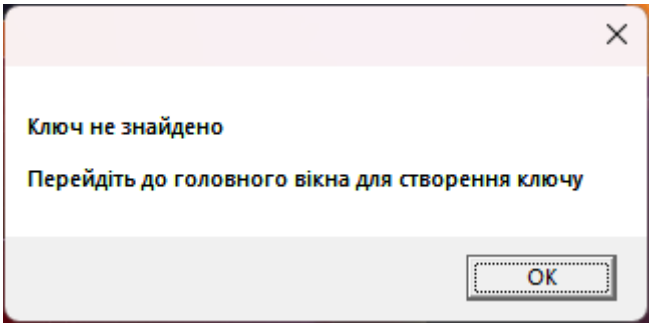


Рисунок 1.29. Вікно повідомлення про відсутність ключу

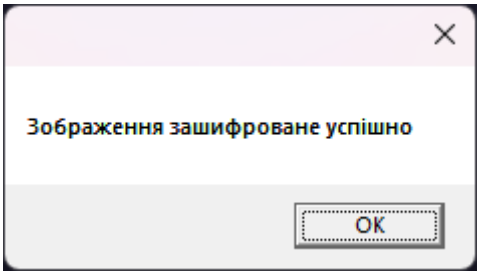


Рисунок 1.30. Вікно повідомлення про успішне шифрування

Вкладка "Дешифрувати" (рис.1.31) забезпечує покроковий процес відновлення оригінальних зображень із зашифрованих файлів. Вона побудована

так, щоб дозволити користувачу легко здійснювати операції дешифрування, мінімізуючи можливість помилок при виборі файлів та ключів. У полі для вибору ключа дешифрування користувач має завантажити необхідний криптографічний ключ. Розташовано кнопку "Вибрати ключ", що відкриває діалог для перегляду файлів.

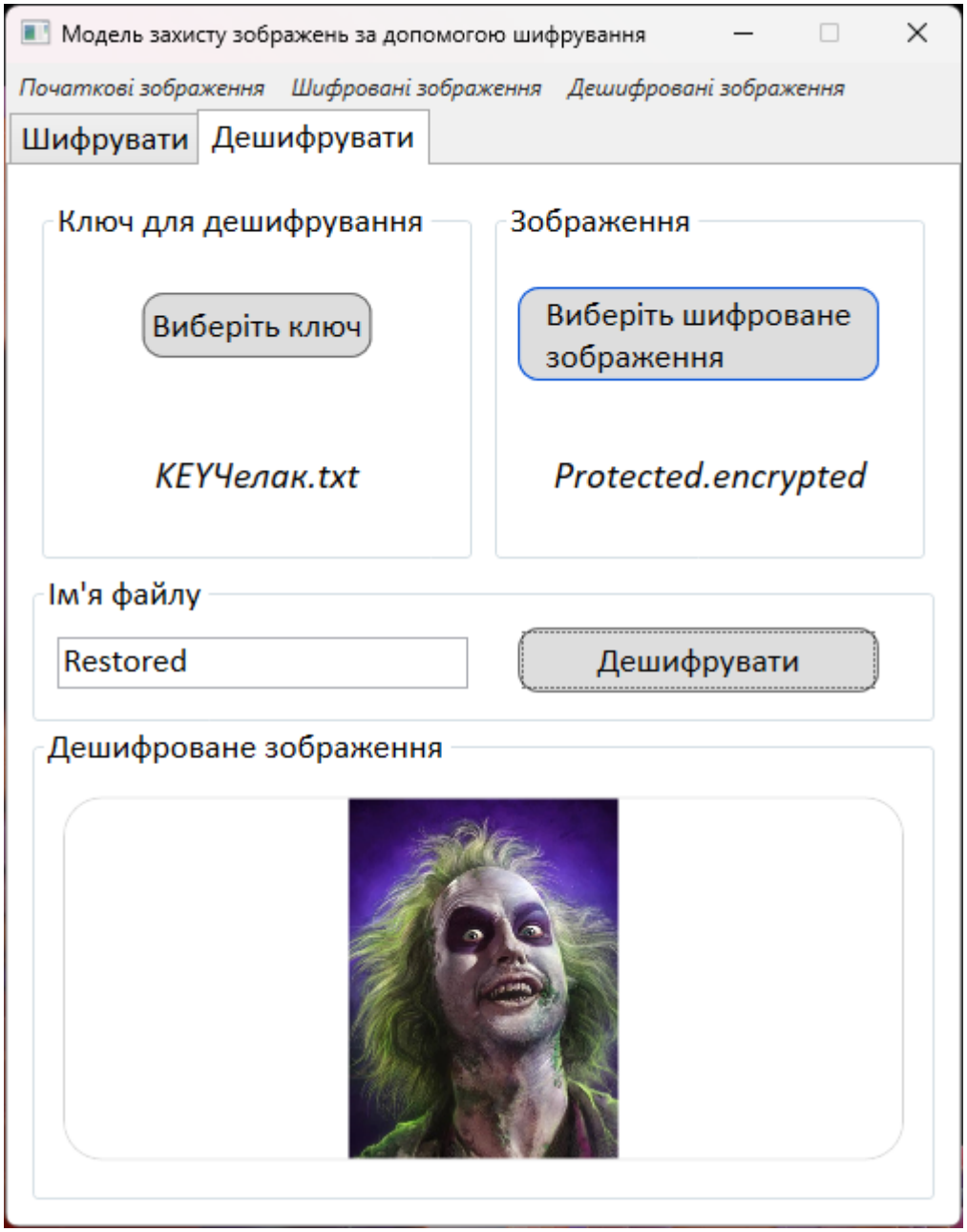


Рисунок 1.31. Дочірнє вікно з відкритою вкладкою “Дешифрувати“

Для виконання процесу відновлення даних користувачу потрібно вказати файл із зашифрованим зображенням. Наявна кнопка "Вибрати шифроване зображення", після натискання якої відкривається стандартний файловий діалог. Поле, що відображає назву файлу, дозволяє впевнитися, що обрано правильний

файл. Поле для введення імені відновлюваного файлу дозволяє задати бажане ім’я для файлу після дешифрування. Введене ім’я допомагає зберегти порядок у сховищі файлів і гарантує, що відновлений файл буде легко ідентифікований. Після вказання ключа та вибору зашифрованого зображення користувач натискає кнопку "Дешифрувати". З цього моменту запускається процес, який у двох етапах – спочатку симетричне дешифрування AES, а потім зворотня обробка, спрямована на відновлення первинної структури зображення – повертає оригінальний вміст. Результат операції відображається у вигляді попереднього перегляду. Це візуальне поле дозволяє користувачу переконатися в успішності дешифрування та якості відновленої інформації.

Перед виконанням операції система перевіряє, чи обрано криптографічний ключ для дешифрування та чи вказано коректне зашифроване зображення. Після натискання кнопки "Дешифрувати" застосунок ініціює процес відновлення даних, після чого виводить повідомлення про успішне відновлення зображення (рис.1.32). Відновлене зображення негайно візуалізується у спеціальному полі, що дозволяє виконати попередню оцінку результату. Користувач бачить підписаний та ідентифікований файл, що містить дешифроване зображення. Це дозволяє оперативно зберегти результат за вказаним іменем, щоб надалі використовувати його за потребою.

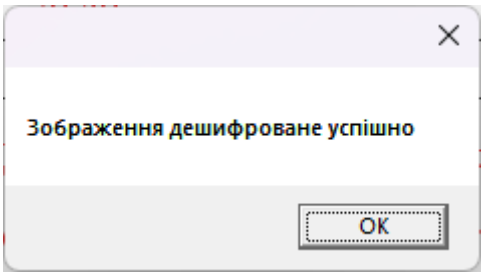


Рисунок 1.32. Вікно повідомлення про успішне дешифрування

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

В даному дипломному проекті здійсненна розробка моделі захисту зображень за допомогою вбудованого шифрування. Це забезпечить не лише високий рівень захисту конфіденційних даних, але й дозволить виконувати подальшу обробку зображення (наприклад, фільтрацію) без необхідності розкривати його вміст.

Рівень ефективності програмного забезпечення залежить не тільки від його якості, а й від результативності самого процесу його створення. Оцінювання якості ПЗ здійснюється за різними критеріями: з урахуванням користувацького досвіду, раціонального використання ресурсів і відповідності встановленим вимогам. Крім того, з точки зору користувача важливу роль відіграють витрати на розробку, зокрема фінансові та трудові ресурси. У цьому розділі наведено розрахунок вартості розробленого програмного продукту.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість створення програмного продукту визначається його масштабом, рівнем трудомісткості, кваліфікацією виконавців та встановленими ринковими термінами. Використовуючи метод структурної аналогії та аналіз відповідних каталогів аналогічного ПЗ, можна встановити обсяг програмного засобу, що виражається у тисячах умовних машинних команд для аналога

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, умовних машинних команд**
1. ПП автоматизації засобів по каталогу	680 – 7000
2. ПП автоматизованих розрахунків	1300 – 8600
3. ПП оптимізації розрахунків	1300 – 4200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Вибравши аналог ПЗ, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Визначення трудомісткості

Обсяг ПЗ, тис. умов. машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин).

Трудомісткість програмного продукту визначається окремо для кожного етапу розробки, виходячи з показників трудомісткості відповідного аналога. При цьому враховують рівень складності розробки, ступінь інноваційності та частку використання стандартних модулів. Розрахунки проводяться згідно з наступними формулами:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	**Ступінь новизни**		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	**Ступінь новизни**	**Значення $К_н$**
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП, що має аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	**Значення $К_т$**
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а * L_1 * К_н = 183,2 * 0,12 * 0,7 = 15,38 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а * L_2 * К_н = 183,2 * 0,11 * 0,7 = 14,11 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а * L_3 * К_н * К_т = 183,2 * 0,61 * 0,7 * 0,6 = 46,94 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 3 (стр), розробка ТП $N_{тп}$=29 (стр), розробка робочого проекту $N_{рп}$=9 (стр), пояснювальна записка відповідно $N_{пз}$=18 (стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=15,38	$T_{кк}$=0,7∗$N_{тз}$= 0,7*3=2,1	$T_{нк}$=0,15∗$N_{тз}$=0,15*3=0,45
2.Розробка ТП	$T_{Ртп}$=14,11	$T_{кк}$=0,7∗$N_{тп}$=0,7*29=20,3	$T_{нк}$=0,15∗$N_{тп}$=0,15*29=4,35
3.Розробка РП	$T_{Ррп}$= 46,94	$T_{кк}$=0,7∗$N_{рп}$=0,7*9=6,3	$T_{нк}$=0,15∗$N_{рп}$=0,15*9=1,35
4.Розробка ПЗ	$T_{пз}$=1,5∗$N_{пз}$= 1,5*18 =27	$T_{кк}$=0,7∗$N_{тз}$=0,7*18=12,6	$T_{нк}$=0,15∗$N_{пз}$=0,15*18=2,7
Усього, в т.ч.:	153,58		
- на розробку	$\Sigma T_р$=103,43		
- контроль керівника		$\Sigma T_{кк}$=41,3	
- нормоконтроль			$\Sigma T_{нк}$=8,85

2.3 Розрахунок ціни програмного продукту

Для оцінки вартості програмного продукту розглядаються основна заробітна плата виконавців, матеріальні витрати та загальні витрати на розробку ПЗ. Детальний розрахунок основної заробітної плати наведено у таблиці 2.7. Згідно зі статтею 8 «Закону про Державний бюджет України на 2025» з 1 січня 2025 року встановлено мінімальну місячну заробітну плату у розмірі 8000 гривень, а також мінімальну погодинну тарифну ставку – 48,00 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	103,43	48,00	4964,64
2.Контроль керівника	41,3	110,00	4543,00
3.Нормоконтроль	8,85	100,00	885,00
Усього	-	-	ΣЗо= 10392,64

Зробимо розрахунок матеріальних витрат на розробку ПЗ. Розрахунок зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	59	5.0	295,00
Разом	-	-	-	$В_{м\,і}$=295,00
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0{,}1 \times Вм1$= 0,1*280=28,0
Усього				$В_м$=$В_{мі}$+$В_{тр\ з=}$ 308,00

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	324,50	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	10392,64	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	1039,26	$Зд = 0{,}1 \times Зо$ = 10392,64*0,1
4.Відрахування до єдиного фонду соціального внеску	2515,02	$Вє.с.в. = 0{,}22 \times (Зо + Зд)$= 0,22* (10392,64+1039,26)
5. Накладні витрати	4157,06	$Внак. = 0{,}4 \times Зо$ =0,4 * 10392,64
6. Повна собівартість	18428,48	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 324,50+10392,64+1039,26+2515,02 +4157,06

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$П=(С_{пов}*Р)/100=(18428,48*15)/100=2764,27$ грн (2.7)

де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о = С_{пов}+П$ = 18428,48+2764,27 = 21192,75 грн; (2.8)

Виходячи з отриманих даних, ціна реалізації розробленого програмного забезпечення становитиме:

$Ц_р=Ц_о$+ПДВ = 21192,75 +21192,75 *0.2 = 25431,30 грн; (2.9)

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Безпека праці, спрямована на створення небезпечних і нешкідливих умов праці. На сучасному етапі розвитку виробництва вона набуває все більше важливого значення.

Вирішення завдань охорони праці базується на досягненнях ергономіки, наукової організації праці, технічної естетики, гігієни та фізіології праці, психофізіології. Крім того, успіх охорони праці визначається темпами впровадження передової техніки, підвищення рівня механізації і автоматизації виробничих процесів, удосконаленням технології та організації виробництв

Безпека праці на підприємстві може бути на належному рівні тільки тоді, коли всебічно відповідає вимогам трудового законодавства, державним стандартам України, норм і правил, розроблених для збереження здоров'я працюючих. Важливе місце при цьому належить виконанню організаційних вимог з охорони праці, а також трудовій та виробничій дисципліні працюючих.

Дипломний проектом передбачена розробка моделі захисту зображень за допомогою вбудованого шифрування. Виконання даної роботи проводилося за допомогою персонального комп'ютера. У зв'язку з цим необхідно проаналізувати фактори ризику при роботі з сучасним персональним комп'ютером.

3.1 Аналіз небезпечних і шкідливих факторів

У процесі розробки пристрою для калібрування комп'ютерних моніторів важливо враховувати фактори, що можуть впливати на безпеку працівника. До таких належать електромагнітне випромінювання, можливі перепади електричної напруги, інтенсивність освітлення робочого місця, а також тепловий вплив при проведенні пайки електронних компонентів. Робоче середовище має бути організоване таким чином, щоб мінімізувати вплив шкідливих чинників та забезпечити комфортні умови праці.

3.2 Гігієнічні вимоги до виробничого середовища

Для забезпечення ефективної роботи над калібрувальним пристроєм необхідно враховувати умови виробничого середовища. Освітлення має

відповідати нормам (300–400 лк згідно з ДБН В.2.5-28:2018), а робоче приміщення повинно бути обладнане вентиляцією для регулювання температури та рівня вологості. Важливо забезпечити зручне розташування робочого місця, захист від шуму та оптимальні санітарні умови.

Створення сприятливого робочого середовища є важливим аспектом продуктивної діяльності персоналу. Гігієнічні вимоги передбачають низку умов, які мають бути забезпечені у приміщенні, де здійснюється розробка та тестування пристрою.

Освітлення робочого місця повинно відповідати встановленим нормативам. Згідно з ДБН В.2.5-28:2018, необхідно забезпечити рівень освітленості 300–400 лк, що дозволить зменшити навантаження на зір та покращити точність роботи з дрібними компонентами пристрою.

Вентиляція та якість повітря Робоче приміщення повинно мати ефективну вентиляцію, щоб усувати шкідливі пари та забезпечувати доступ свіжого повітря. У холодну пору року рекомендована температура 18–20°C, у теплу – 22–25°C. Оптимальний рівень вологості складає 40–60%, що сприяє комфортному перебуванню у приміщенні.

Захист від шуму та вібрацій У місцях, де проводяться пайкові роботи та тестування пристрою, слід мінімізувати рівень шуму та вібрацій, які можуть негативно впливати на продуктивність працівників. Для цього застосовують шумоізолюючі матеріали та спеціальні амортизаційні конструкції.

3.3 Вимоги безпеки праці працівника

Безпека працівника під час роботи з паяльними інструментами та електронними пристроями є першочерговим завданням. Необхідно дотримуватися таких заходів:

Використання індивідуальних засобів захисту – працівник повинен працювати у захисних рукавичках, спеціальному одязі та з використанням ізоляційного покриття на робочій поверхні.

Дотримання правильного розташування робочого місця – важливо розташувати технічне обладнання таким чином, щоб уникнути ризику

перекидання чи випадкового контакту з нагрітими частинами.

Контроль електробезпеки – всі пристрої, які використовуються в роботі, повинні мати заземлення та відповідати технічним стандартам безпеки.

Рисунок 3.1. Організація робочого місця оператора ПК

Дотримання правил експлуатації обладнання – працівник повинен перевіряти справність інструментів перед початком роботи та уникати використання пошкодженого обладнання.

3.4 Правила безпеки праці

При виконанні паяльних робіт слід дотримуватися таких правил:

- Забороняється використання несправних інструментів.
- Не можна торкатися до нагрітих частин паяльника, щоб уникнути опіків.
- Обов’язкове використання витяжки для видалення шкідливих парів припою.
- Деталі утримувати плоскогубцями або спеціальними інструментами, щоб уникнути прямого контакту з гарячими компонентами.
- Регулярно провітрювати приміщення та дотримуватися санітарних норм після завершення роботи.

3.5 Пожежна безпека

Пожежна безпека є одним із критичних аспектів організації робочого місця, особливо при роботі з електронними пристроями, такими як система калібрування моніторів.

Основними причинами виникнення пожеж у виробничому приміщенні можуть бути:

- Несправність електрообладнання – коротке замикання, перевантаження електромережі та механічні пошкодження електрокабелів.
- Неправильне зберігання легкозаймистих матеріалів – відкриті ємності з хімічними речовинами, займисті припої та ізоляційні матеріали.
- Порушення техніки безпеки при пайці – попадання розплавленого припою на горючі матеріали, перегрів електропаяльників та залишення нагрітого обладнання без нагляду.
- Недотримання правил експлуатації електромереж – використання несправних розеток, відсутність захисного заземлення та неправильне підключення обладнання.
- Необережне поводження з вогнем – використання відкритого полум’я у робочому приміщенні, паління та неправильне поводження з нагрітими предметами.

Для запобігання пожежам необхідно дотримуватися ряду заходів безпеки:

- Систематичний контроль електромережі – перед початком роботи слід перевірити справність розеток, проводів та електроприладів.
- Забезпечення робочого приміщення засобами пожежогасіння – кожне місце роботи повинно бути оснащене необхідними протипожежними засобами, такими як:
- Вогнегасник (порошковий або вуглекислотний, залежно від специфіки приміщення).
- Ящик з піском об’ємом не менше 0,5 м³ для ліквідації розливів рідких займистих речовин.
- Лопати та відра для ефективного використання піску.

Рисунок 3.2. Засоби пожежогасіння

Пожежні щити мають бути розміщені на видимих місцях та містити необхідний набір засобів для оперативної ліквідації займання.

Щоб мінімізувати ризик виникнення пожеж, робоче місце має відповідати наступним вимогам:

- Запасні виходи повинні бути позначені світловими покажчиками із написом «Запасний вихід», видимими навіть при недостатньому освітленні.
- Пожежні крани повинні бути доступними на кожному поверсі, у коридорах та біля сходових клітин.
- Вогнегасники слід розміщувати на видимих місцях, на висоті не більше 1,5 м від підлоги для швидкого доступу.
- Евакуаційний план повинен бути розміщений у головному вході приміщення та містити детальний маршрут виходу при пожежі.
- Будівлі та приміщення повинні бути оснащені пожежними щитами з необхідним інструментом для ліквідації загоряння.
- Електромережа повинна відповідати нормам захисту – дроти та розетки повинні бути ізольованими та не перевантаженими.
- Забезпечення контрольованого доступу до виробничих приміщень – стороннім особам забороняється перебувати в робочій зоні без відповідного дозволу.

ВИСНОВКИ

У дипломному проекті було реалізовано комбінований підхід захисту зображень, який інтегрує метод гомоморфного шифрування (на основі перестановки стовпців із середньоарифметичною фільтрацією) з симетричним шифруванням за стандартом AES. Такий багаторівневий метод забезпечує не лише попередню обфускацію візуальних даних, а й високий рівень конфіденційності завдяки стійкості AES до криптоаналітичних атак.

Розроблене програмне забезпечення побудоване на основі .NET-технологій із використанням мови C# та WPF для створення зручного графічного інтерфейсу. Головне вікно забезпечує інтуїтивну навігацію між основними функціями, такими як керування ключами, відкриття дочірніх вікон для шифрування/дешифрування, а також налаштування параметрів зберігання даних. Дочірні вікна дозволяють ефективно організувати роботу із зображеннями, надаючи користувачу можливість легко виконувати операції шифрування та дешифрування.

Завдяки попередній обробці зображення методом гомоморфного шифрування, що включає перестановку стовпців і середньоарифметичну фільтрацію, зображення втрачає свою первинну структурну цілісність ще до застосування AES. Це значно ускладнює спроби злому системи, оскільки зловмиснику необхідно подолати два незалежних рівня захисту.

Ретельно продуманий інтерфейс, що включає головне вікно, діалогові вікна для вибору директорій та дочірні інтерфейси для шифрування/дешифрування, сприяє зниженню кількості людських помилок та полегшує роботу користувача із системою. Інтуїтивно зрозумілі елементи управління дозволяють як досвідченим фахівцям, так і новачкам швидко освоїти роботу з програмою.

Результати інтеграційного тестування підтвердили ефективність реалізованої моделі захисту зображень. Система демонструє високу стійкість до несанкціонованого доступу, а її модульна структура дозволяє легко модифікувати або розширювати функціонал при подальшій розробці.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Григоренко А. В. Застосування криптографії для захисту зображень – Дніпро: «Дніпропетровський національний університет», 2020. – 210 с.
2. Литвин О. П. Гомоморфне шифрування в інформаційних системах – Київ: «Інфра-М», 2021. – 180 с.
3. Антоненко С. А. Основи криптографії: сучасні методи захисту інформації – Київ: «Наукова думка», 2021. – 350 с.
4. Дорошенко О. О. Сучасні методи криптографічного захисту: навчальний посібник – Львів: «Літопис», 2020. – 280 с.
5. Шевченко І. І., Петренко М. К. Інформаційна безпека: теорія та практика – Харків: «Фенікс», 2019. – 310 с.
6. Гуменюк П. І. Методи шифрування даних: аналіз та перспективи – Одеса: Одеський національний університет, 2022. – 220 с.
7. Курбатов А. І., Мороз Т. Ф. Криптографічні алгоритми та їх застосування – Київ: «Київський університет», 2020. – 330 с.
8. Левченко В. Ф. Сучасні тенденції в криптографії та цифровій безпеці – Львів: Львівський національний університет ім. І. Франка, 2021. – 250 с.
9. Іваненко Р. Л. Аналіз криптографічних протоколів: теорія і практика – Одеса: «Астропринт», 2023. – 200 с.
10. Соколова О. Є., Бойко В. О. Квантова криптографія та інформаційна безпека – Київ: «Наукова думка», 2023. – 270 с.
11. Байрак І. М. Основи цифрової безпеки: алгоритми шифрування та підпис – Харків: ХНЕУ, 2022. – 310 с.
12. Коваленко М. О. Програмні засоби захисту інформації: методологія та практика – Львів: «Видавничий дім "Світ знань"», 2021. – 325 с.
13. Мельник С. І. Комплексний підхід до захисту цифрової інформації – Київ: «Університетська видавнича спілка», 2022. – 240 с.
14. E-learning платформа "КриптоЛаб". [Електронний ресурс]. – Режим доступу: https://cryptolab.edu.ua (05.05.2025).

ДОДАТОК А. Фрагмент коду мовою C# моделі захисту зображень за допомогою вбудованого шифрування

```
using System;
using System.Drawing;
using System.IO;
using System.Security.Cryptography;

namespace Encryptie_Tools
{
    public class ImageEncryption
    {
        /// <summary>
        /// Гомоморфне шифрування: перестановка стовпців та
         ///середньоарифметична фільтрація зображення.
        /// </summary>
        /// <param name="original">Оригінальне зображення (Bitmap).</param>
        /// <param name="permutationKey">Ключ, за яким генерується
перестановка стовпців.</param>
        /// <returns>Оброблене зображення, після застосування перестановки та
фільтрації.</returns>
        public static Bitmap HomomorphicEncrypt(Bitmap original, string
permutationKey)
        {
            int width = original.Width;
            int height = original.Height;

            // Генеруємо перестановку стовпців із використанням ключа
            int[] permutation = GeneratePermutation(width, permutationKey);

            // Виконуємо перестановку стовпців
            Bitmap colPermuted = new Bitmap(width, height);
            for (int row = 0; row < height; row++)
            {
                for (int col = 0; col < width; col++)
                {
                    int sourceCol = permutation[col];
                    Color pixelColor = original.GetPixel(sourceCol, row);
                    colPermuted.SetPixel(col, row, pixelColor);
                }
            }

            // Застосовуємо середньоарифметичну фільтрацію (mean filter)
            // для розмивання зображення
            Bitmap filtered = MeanFilter(colPermuted, filterSize: 3);

            return filtered;
        }

        /// <summary>
        /// Генерує псевдовипадкову перестановку чисел від 0 до n-1 за
заданим ключем.
        /// </summary>
        private static int[] GeneratePermutation(int n, string key)
        {
            int[] perm = new int[n];
            for (int i = 0; i < n; i++)
            {
                perm[i] = i;
            }
            int seed = key.GetHashCode();
            Random rnd = new Random(seed);
            for (int i = n - 1; i > 0; i--)
            {
                int j = rnd.Next(i + 1);
```

```
                int temp = perm[i];
                perm[i] = perm[j];
                perm[j] = temp;
            }
            return perm;
        }

        /// <summary>
        /// Застосовує простий середньоарифметичний фільтр (mean filter) до
зображення.
        /// </summary>
        /// <param name="image">Вхідне зображення.</param>
        /// <param name="filterSize">Розмір фільтра </param>
        /// <returns>Відфільтроване зображення.</returns>
        private static Bitmap MeanFilter(Bitmap image, int filterSize)
        {
            int width = image.Width;
            int height = image.Height;
            Bitmap result = new Bitmap(width, height);
            int offset = filterSize / 2;

            for (int y = 0; y < height; y++)
            {
                for (int x = 0; x < width; x++)
                {
                    int rSum = 0, gSum = 0, bSum = 0;
                    int count = 0;

                    for (int j = -offset; j <= offset; j++)
                    {
                        for (int i = -offset; i <= offset; i++)
                        {
                            int nx = x + i;
                            int ny = y + j;
                       if (nx >= 0 && nx < width && ny >= 0 && ny < height)
                            {
                                Color pixel = image.GetPixel(nx, ny);
                                rSum += pixel.R;
                                gSum += pixel.G;
                                bSum += pixel.B;
                                count++;
                            }
                        }
                    }

                    int rMean = rSum / count;
                    int gMean = gSum / count;
                    int bMean = bSum / count;
                    Color meanColor = Color.FromArgb(rMean, gMean, bMean);

                    result.SetPixel(x, y, meanColor);
                }
            }

            return result;
        }

        /// <summary>
        /// Перетворює об’єкт Bitmap у масив байтів (PNG-формат).
        /// </summary>
        public static byte[] BitmapToBytes(Bitmap bmp)
        {
            using (MemoryStream ms = new MemoryStream())
            {
                bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
                return ms.ToArray();
            }
        }
```

```
        /// <summary>
        /// Відтворює об’єкт Bitmap з масиву байтів.
        /// </summary>
        public static Bitmap BytesToBitmap(byte[] data)
        {
            using (MemoryStream ms = new MemoryStream(data))
            {
                return new Bitmap(ms);
            }
        }

        /// <summary>
        /// AES-шифрування масиву байтів із заданими ключем та вектором.
        /// </summary>
        public static byte[] AESEncrypt(byte[] plainBytes,
byte[] key, byte[] iv)
        {
            using (Aes aes = Aes.Create())
            {
                aes.KeySize = 128;
                aes.BlockSize = 128;
                aes.Key = key;
                aes.IV = iv;

                using (MemoryStream ms = new MemoryStream())
                {
                    using (CryptoStream cs = new CryptoStream(ms,
aes.CreateEncryptor(), CryptoStreamMode.Write))
                    {
                        cs.Write(plainBytes, 0, plainBytes.Length);
                    }
                    return ms.ToArray();
                }
            }
        }

        /// <summary>
        /// AES-дешифрування масиву байтів із заданими ключем та вектором
ініціалізації.
        /// </summary>
        public static byte[] AESDecrypt(byte[] cipherBytes, byte[] key,
byte[] iv)
        {
            using (Aes aes = Aes.Create())
            {
                aes.KeySize = 128;
                aes.BlockSize = 128;
                aes.Key = key;
                aes.IV = iv;

                using (MemoryStream ms = new MemoryStream())
                {
                    using (CryptoStream cs = new CryptoStream(ms,
aes.CreateDecryptor(), CryptoStreamMode.Write))
                    {
                        cs.Write(cipherBytes, 0, cipherBytes.Length);
                    }
                    return ms.ToArray();
                }
            }
        }

        /// <summary>
        /// Комбіноване шифрування: спочатку застосовується гомоморфне
        /// шифрування (перестановка стовпців + фільтрація),
        /// після чого виконують AES-шифрування отриманих даних.
        /// </summary>
```

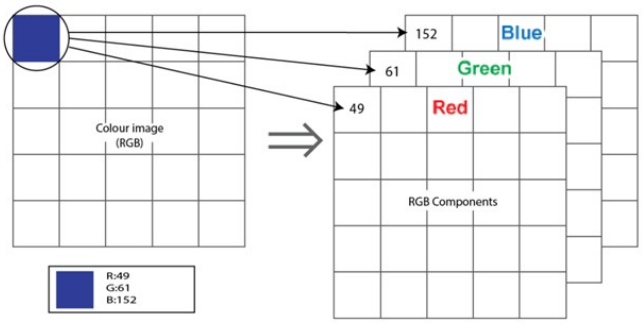
```
        /// <param name="originalImage">Оригінальне зображення.</param>
        /// <param name="permutationKey">Ключ для генерації перестановки
стовпців.</param>
        /// <param name="aesKey">Криптографічний ключ для AES.</param>
        /// <param name="aesIV">Вектор ініціалізації для AES.</param>
        /// <returns>Зашифровані байти зображення.</returns>
        public static byte[] CombinedEncrypt(Bitmap originalImage, string
permutationKey, byte[] aesKey, byte[] aesIV)
        {
            // 1. Гомоморфне шифрування: перестановка стовпців і фільтрація
            Bitmap homomorphicImage = HomomorphicEncrypt(originalImage,
permutationKey);
            // 2. Перетворення зображення на масив байтів (PNG)
            byte[] imageBytes = BitmapToBytes(homomorphicImage);
            // 3. AES-шифрування отриманого масиву байтів
            byte[] encryptedBytes = AESEncrypt(imageBytes, aesKey, aesIV);
            return encryptedBytes;
        }

        /// <summary>
        /// Комбіноване дешифрування: спочатку виконується AES-дешифрування,
        /// після чого здійснюється інверсія перестановки стовпців для
відновлення порядку.
        /// Примітка: середньоарифметичну фільтрацію неможливо звернути, тому
відновлений образ буде приблизним.
        /// </summary>
        public static Bitmap CombinedDecrypt(byte[] encryptedBytes, string
permutationKey, byte[] aesKey, byte[] aesIV)
        {
            // 1. AES-дешифрування
            byte[] decryptedBytes = AESDecrypt(encryptedBytes, aesKey,
aesIV);
            // 2. Перетворення байтів назад у Bitmap
            Bitmap decryptedImage = BytesToBitmap(decryptedBytes);

            int width = decryptedImage.Width;
            int height = decryptedImage.Height;
            // 3. Генеруємо ту саму перестановку за ключем
            int[] permutation = GeneratePermutation(width, permutationKey);
            // Формуємо зворотну перестановку:
            // для кожного індексу визначаємо початкову позицію
            int[] reversePermutation = new int[width];
            for (int i = 0; i < width; i++)
            {
                reversePermutation[permutation[i]] = i;
            }
            // 4. Відновлюємо порядок стовпців
            Bitmap restored = new Bitmap(width, height);
            for (int row = 0; row < height; row++)
            {
                for (int col = 0; col < width; col++)
                {
                    int sourceCol = reversePermutation[col];
                    Color pixelColor = decryptedImage.GetPixel(sourceCol,
row);
                    restored.SetPixel(col, row, pixelColor);
                }
            }
            return restored;
        }
    }
}
```

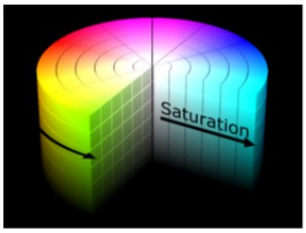
ДОДАТОК Б. Слайди мультимедійної презентації

Розробка моделі захисту зображень
за допомогою вбудованого шифрування

Дипломний проект Челак Дмитрія, гр.4КБ-02

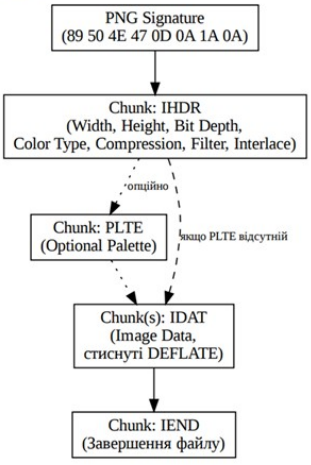
Представлення растрового кольорового RGB-зображення у вигляді матриць інтенсивності кольору





Компоненти моделі HVB (відтінок, насиченість, яскравість)

Структура файлу PNG



Обробка квадратної апертури зображення за допомогою медіанного фільтра

$$s_{x,y} = a_{t,p} : N(a_{i,j} \le a_{t,p}) = N(a_{i,j} \ge a_{t,p}),$$

де $x - \frac{r-1}{2} \le i \le x + \frac{r-1}{2}$, $y - \frac{r-1}{2} \le i \le x + \frac{r-1}{2}$, $t \subseteq i$, $p \subseteq j$,

$$S = \begin{bmatrix} s_{1,1}\, s_{1,2} \ldots s_{1,h} \\ s_{2,1}\, s_{2,2} \ldots s_{2,h} \\ \ldots \\ s_{k,1}\, s_{k,2} \ldots s_{k,h} \end{bmatrix}$$

2	4	6
1	9	9
3	5	6

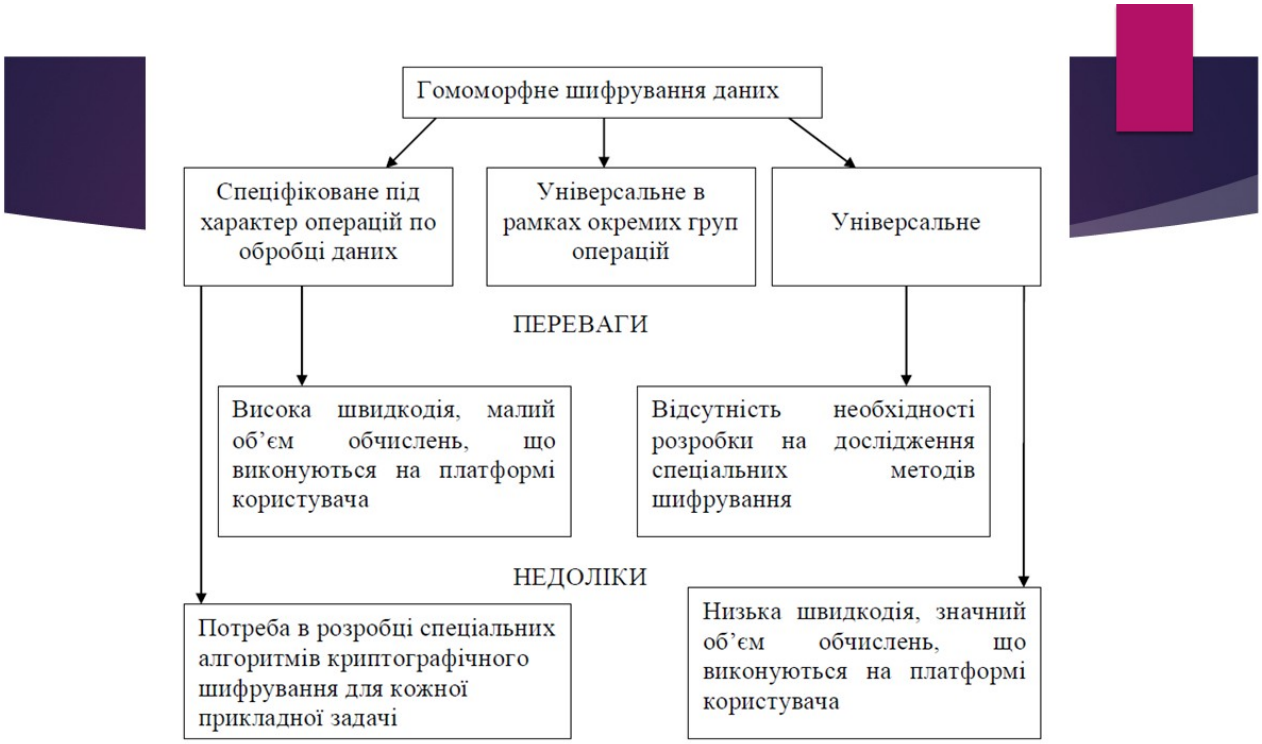
*	*	*
*	5	*
*	*	*

Обробка квадратної апертури за допомогою середньоарифметичного фільтра

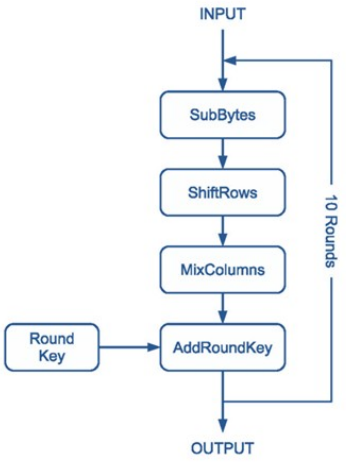
2	1	6
8	10	9
5	4	9

$$s_{x,y} = \frac{1}{r^2} \sum_{i=x-\frac{r-1}{2}}^{x+\frac{r-1}{2}} \sum_{j=y-\frac{r-1}{2}}^{y+\frac{r-1}{2}} a_{i,j}$$

*	*	*
*	6	*
*	*	*



Процес шифрування зображень за алгоритмом AES. Діаграма роботи алгоритму AES

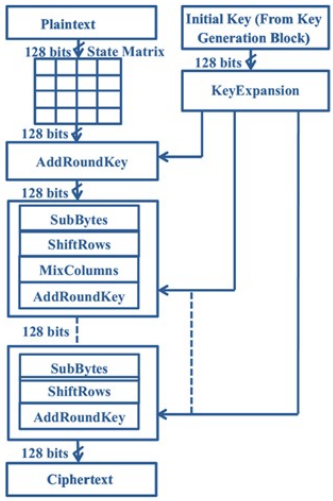


W1	W2	W3	W4	W5	W6	W7	W8								
2B	28	AB	09	A0	88	23	2A								
7E	AE	F7	CF	FA	54	A3	6C	.....				.....			
15	D2	15	4F	FE	2C	39	76								
16	A6	88	3C	17	B1	39	05								

Cipher key — Round key 1 — ..... — Round key 10

RW		W1		S1		R1		W5
CF		2B		8A		01		A0
4F		7E	⊕	84	⊕	00	=	FA
3C		15		EB		00		FE
09		16		01		00		17

Етап отримання раундових ключів у AES



Реалізація шифрування зображень з захищеною середньоарифметичною фільтрацією

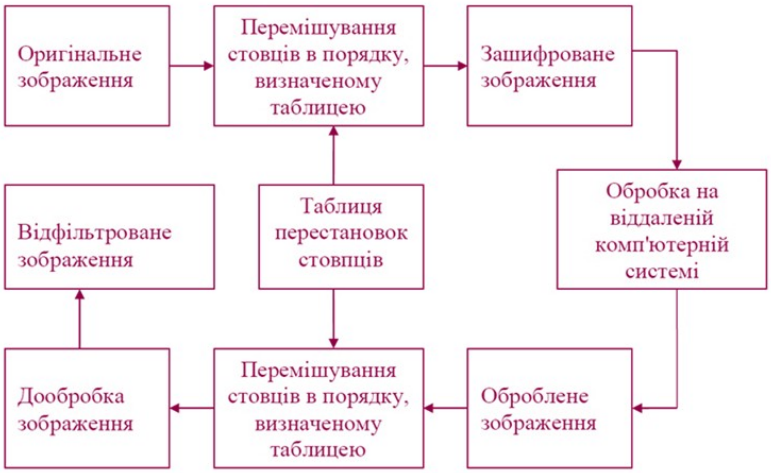
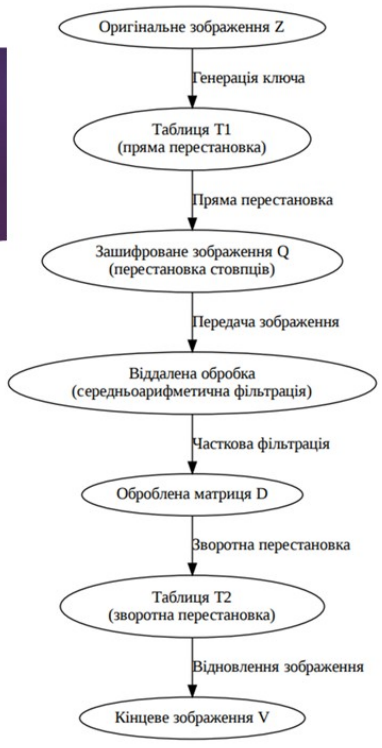


Схема процесу інтегрованого шифрування зображень

Організація гомоморфного шифрування зображень при їх захищеній середньоарифметичній фільтрації

$$V[i,j] = \frac{1}{n2}\sum_{u=i-\frac{n-1}{2}} i + \frac{n-1}{2}\sum_{v=j-\frac{n-1}{2}}^{j+\frac{n-1}{2}} p_{u,v}$$

Часткова середньоарифметична фільтрація



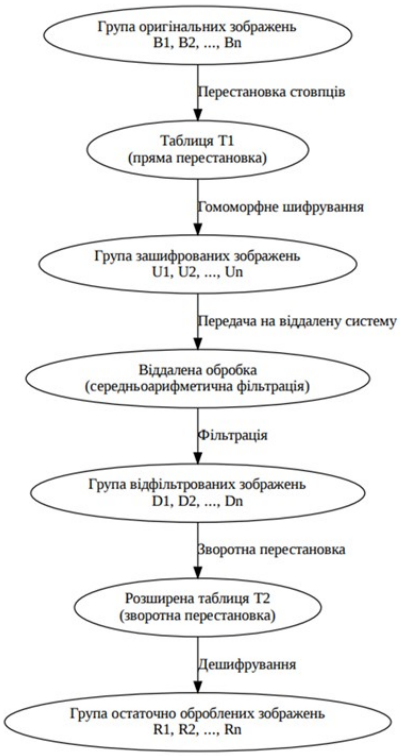


Схема послідовності операцій перемішування стовпців при захищеній фільтрації групи зображень

32	54	43	2	53
17	6	84	45	3
31	37	74	26	78
16	22	58	53	29
84	24	63	43	43

38	72	75	91	55
69	100	83	88	9
70	85	47	35	46
9	96	38	3	30
75	38	51	47	65

58	33	57	59	72
68	100	17	25	5
17	94	3	67	64
37	13	78	97	80
62	12	46	94	94

51	37	32	19	73
89	60	29	100	36
6	33	64	22	87
28	54	11	48	95
1	8	20	78	39

Оригінальні зображення

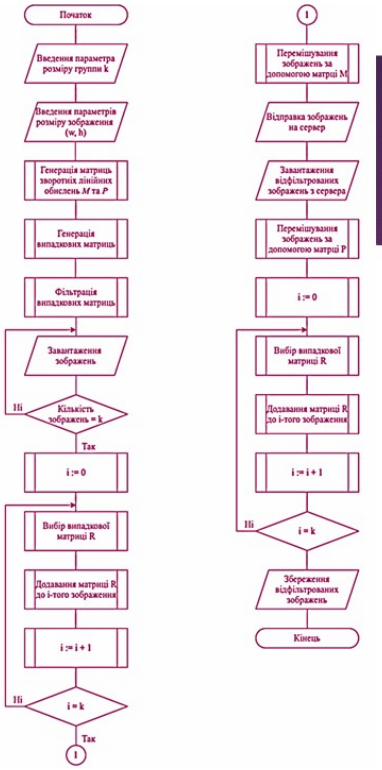
437	340	451	316	570
513	468	411	521	330
253	441	376	265	468
296	309	337	503	467
454	385	329	580	447

331	271	335	247	407
354	401	333	336	251
216	315	244	200	357
197	181	294	368	286
374	255	282	448	335

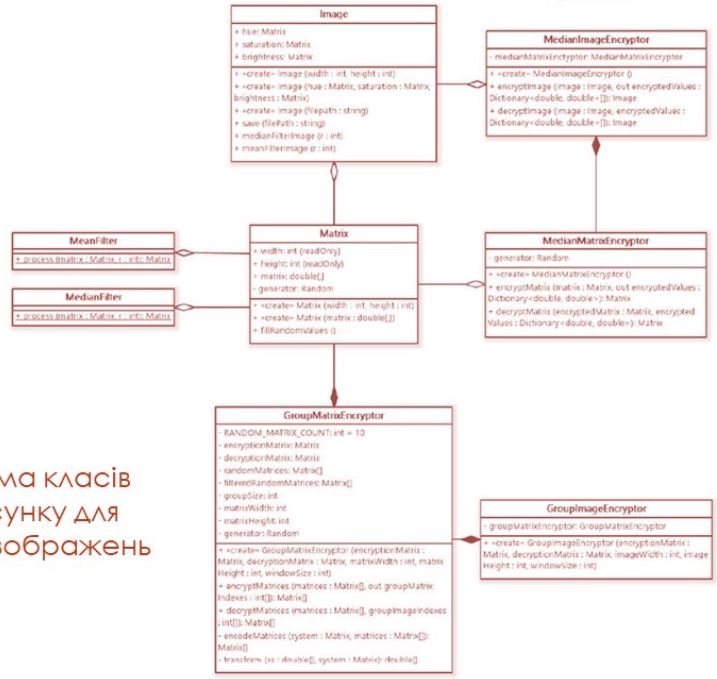
200	139	248	72	304
248	116	204	280	150
98	175	289	102	245
156	188	189	275	282
177	215	157	267	181

313	248	341	199	439
360	316	277	348	235
173	309	299	183	334
239	212	276	465	437
323	314	231	461	290

Зашифровані зображення

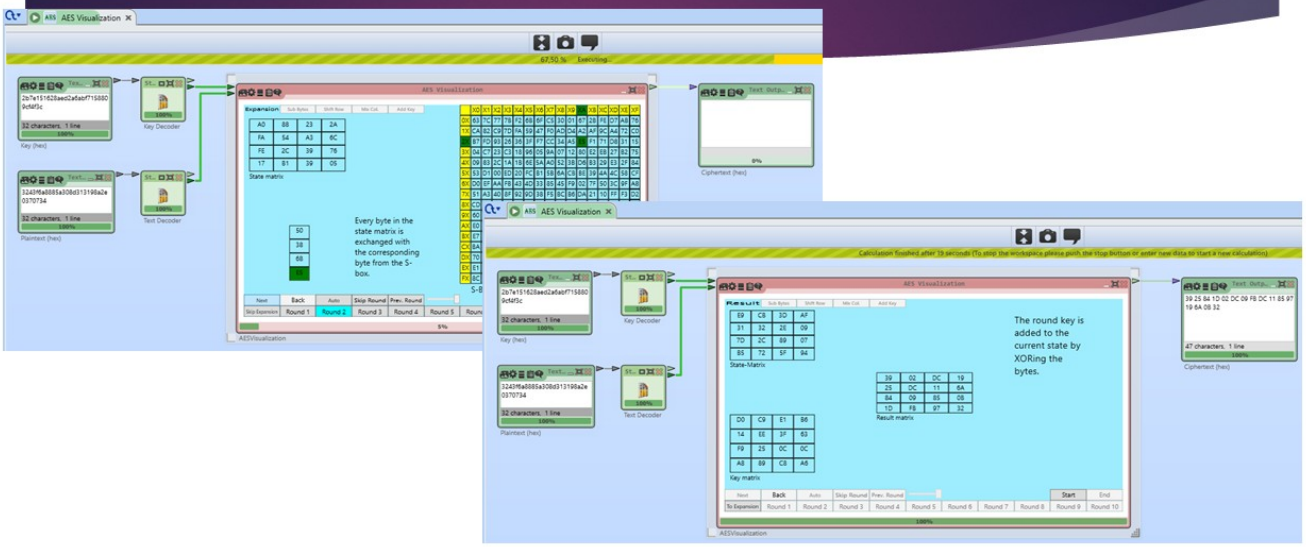


Блок-схема алгоритму середньо-арифметичної фільтрації зображень

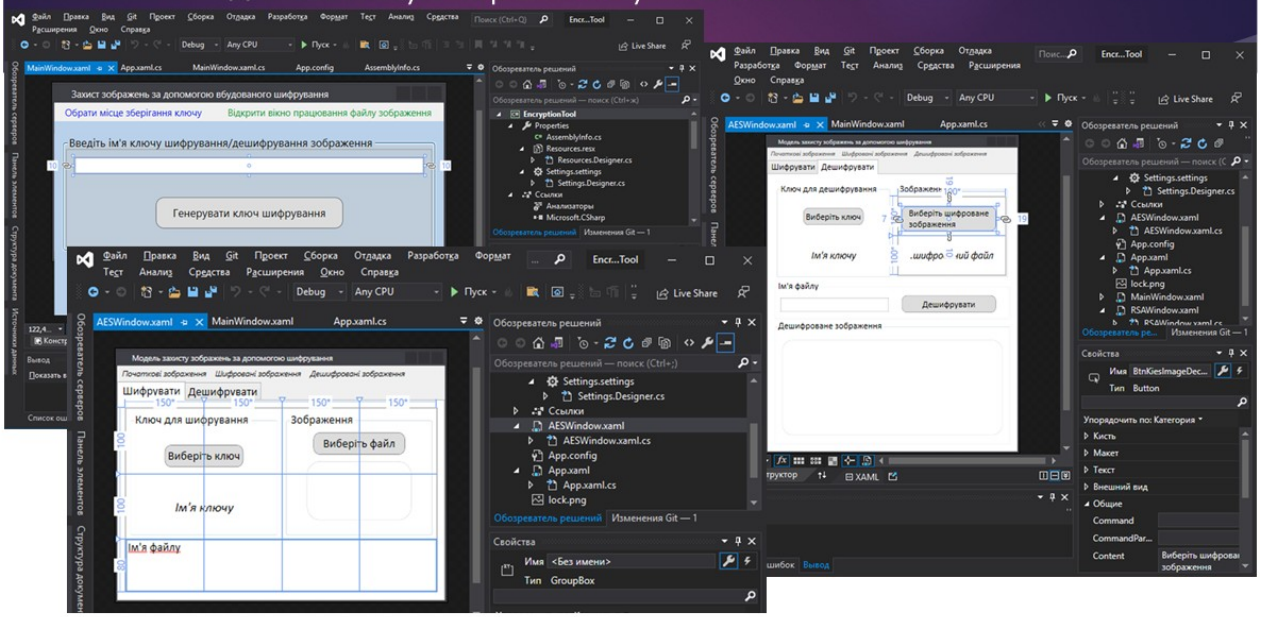


Діаграма класів застосунку для захисту зображень

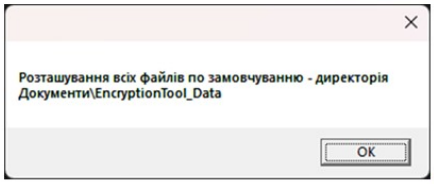
Моделювання роботи алгоритму AES у CrypTool2

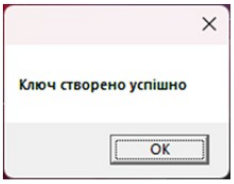


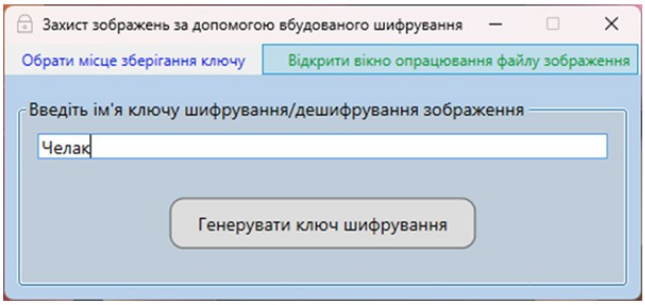
Етапи розробки застосунку для захисту зображень у Visual Studio



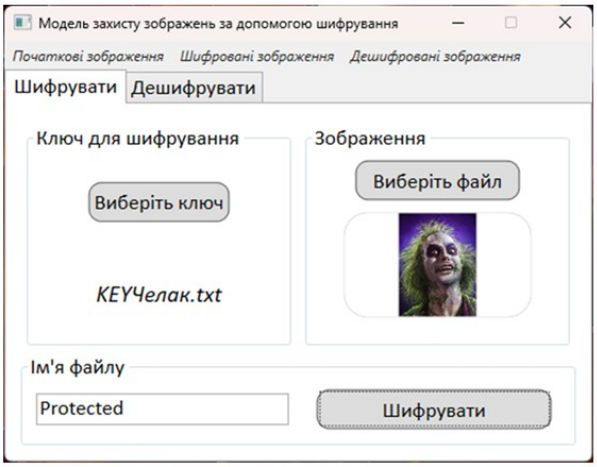
Головне вікно розробленого застосунку. Створення ключу

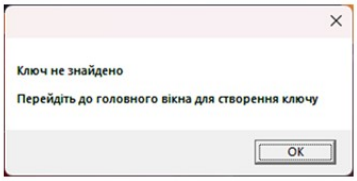


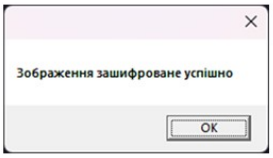




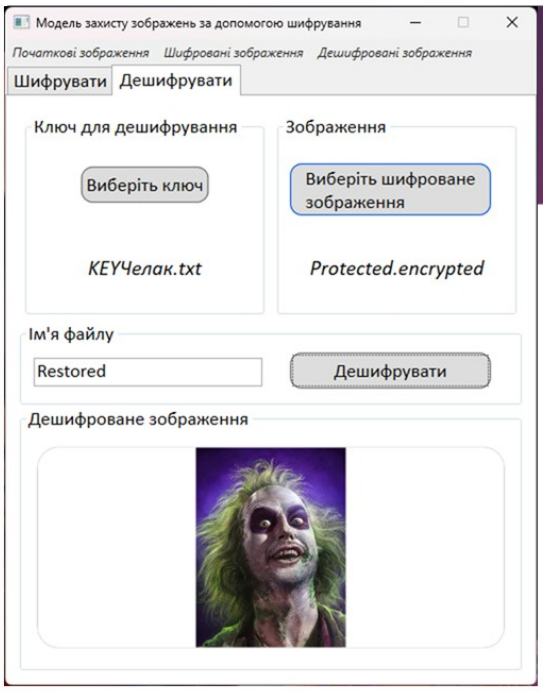
Шифрування зображення у розробленому застосунку

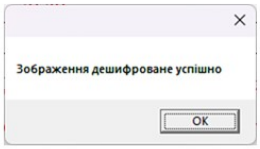






Дешифрування зображення у розробленому застосунку





ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Челак Дмитрія Вікторовича

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Кривченко Юрій Вікторович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка моделі захисту зображень за допомогою вбудованого шифрування*

Обсяг розрахунково-пояснювальної записки *81* сторінок

Обсяг графічної (презентаційної) частини *15* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений створенню моделі захисту зображень за допомогою вбудованого шифрування та її реалізації мовою програмування C# і складається з пояснювальної записки та мультимедійної презентації з відповідними схемами.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (Аналіз сучасних методів і засобів захисту зображень; Аналіз основних форматів растрових зображень; Реалізація середньоарифметичної фільтрації для захисту зображень; Розробка програмної моделі та блок-схем алгоритмів; Аналіз роботи застосунку та моделі захисту зображень у форматі PNG), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 15 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять структурні, та функціональні схеми, діаграми, блок-схеми алгоритмів, скріншоти передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізована модель шифрування зображень представляє практичний інтерес; Зберігання ключів шифрування, шифрованих та дешифрованих зображень реалізовано у структурованому вигляді у окремих каталогах*

д) основні недоліки дипломного проекту *Варто було б передбачити роботу застосунку з різними графічними форматами файлів, а не тільки з PNG; У економічному розділі присутні дрібні відрізнення форматування відносно основного розділу*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Рудніченко Микола Дмитрович*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис: _____

« *20* » *червня* *2025 р.*

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Челак Дмитрія Вікторовича

(прізвище, ім'я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка моделі захисту зображень за допомогою вбудованого шифрування*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 81 сторінок. У пояснювальній записці описано створення моделі захисту зображень за допомогою вбудованого шифрування та її реалізації мовою програмування C#. Графічна частина складається з 15 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Челак Дмитрій поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Челак Дмитрій під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Челак Дмитрій показав вміння організовано працювати над поставленим завданням, застосовувати знання у сфері безпеки комп'ютерних систем і мереж, програмування, використовуючи сучасні комп'ютерні програмні засоби розробки, такі як Microsoft Visual Studio 2022 (WPF)*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Кривченко Юрій Вікторович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерних технологій та програмної інженерії, голова ЦК*

Підпис

«14» червня 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Челак Д.В.,
здобувач освіти гр. 4КБ-02, та

Кривченко Ю.В.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка моделі захисту зображень за допомогою вбудованого шифрування» (автор роботи – Челак Д.В., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Челак Д.В. /

Керівник _____ / Кривченко Ю.В. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Челака Дмитрія Вікторовича

на тему *Розробка моделі захисту зображень за допомогою вбудованого шифрування*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 08.06.2025 р. значення коефіцієнту подібності в роботі становить 15,81%, коефіцієнт цитування – 1,06%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Челака Д.В.* (П.І.Б.)

проведена « *16* » *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/8/2025
Дата редагування 6/8/2025

Документ прийнятий

Звіт подібності

метадані

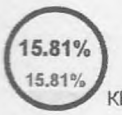
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка моделі захисту зображень за допомогою вбудованого шифрування

Автор
Челак Дмитрій Володимирович
Науковий керівник / Експерт
Кривченко Юрій Вікторович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1

КЦ

25	13843	117294
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	23
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	68

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content	88 0.64 %
2	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	70 0.51 %
3	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	59 0.43 %
4	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	57 0.41 %
5	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	47 0.34 %

6	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	47 0.34 %
7	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	43 0.31 %
8	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	39 0.28 %
9	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	38 0.27 %
10	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	34 0.25 %

з домашньої бази даних (0.09 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	12 (1) 0.09 %

з програми обміну базами даних (1.16 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Метод гомоморфного шифрування для захищеної обробки даних в хмарах [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	87 (12) 0.63 %
2	Метод гомоморфного шифрування зображень при їх віддаленій обробці [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	60 (6) 0.43 %
3	ФКПІ_2023_125_КлименкоА.В [redacted] 7/11/2024 Ukrainian national aviation university (Ukrainian national aviation university)	14 (1) 0.10 %

з Інтернету (14.56 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**897 (70) 6.48 %**
2	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**204 (6) 1.47 %**
3	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/995bdcec-4e4d-4321-8070-4d6badcb8e49/content**	**157 (8) 1.13 %**
4	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**125 (6) 0.90 %**
5	**https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download**	**111 (5) 0.80 %**
6	**https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download**	**94 (3) 0.68 %**
7	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**69 (4) 0.50 %**
8	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	59 (4) 0.43 %
9	**https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download**	**39 (2) 0.28 %**
10	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	35 (2) 0.25 %

11	https://card-file.ontu.edu.ua/server/api/core/bitstreams/44c16132-5f53-48e2-b6c0-61e9a2f0fd75/content	34 (1) 0.25 %
12	https://card-file.ontu.edu.ua/bitstreams/e4afae26-0a7e-4a4d-afc2-94341838de2a/download	31 (2) 0.22 %
13	https://palospublishing.com/writing-c-code-for-high-efficiency-image-and-signal-processing/	20 (2) 0.14 %
14	https://card-file.ontu.edu.ua/bitstreams/f9e1bea4-9c6d-4957-b037-04c4fbac9b21/download	19 (1) 0.14 %
15	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	18 (2) 0.13 %
16	https://utkusen.com/blog/dealing-with-script-kiddies-cryptear-b-incident.html	17 (2) 0.12 %
17	https://ami.lnu.edu.ua/wp-content/uploads/2020/11/Malik.pdf	14 (1) 0.10 %
18	https://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3502/lec/RecursionPermutations.pdf	11 (1) 0.08 %
19	https://www.mygreatlearning.com/blog/heap-sort/	10 (1) 0.07 %
20	https://gist.github.com/kavyaLokuge/bb1a64c82d58c53cb5a47913b31eee1c	10 (1) 0.07 %
21	https://knowledge.allbest.ru/programming/3c0a65625a2bc68b5d43b88421216d36_0.html	10 (1) 0.07 %
22	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	9 (1) 0.07 %
23	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	9 (1) 0.07 %
24	http://ukrefs.com.ua/print:page,1,171772-Razrabotka-i-issledovanie-imitacionnoiy-modeli-lokal-nyh-vychislitel-nyh-seteiy.html	7 (1) 0.05 %
25	https://cpsm.kpi.ua/stud/bak/Gulaevich_LP_71.pdf	6 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»_
Освітньо-професійна програма: **«Безпека комп'ютерних систем і мереж» Група:** 4КБ- 02

.Дипломний **проект здобувача освіти денної форми навчання КБ.** 02.22.000.ДП

ЧЕЛАК
ДМИТРІЯ ВІКТОРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»_
Освітньо-професійна програма: **«Безпека** комп'ютерних **систем і мереж» Група:** 4 КБ-02

ПОЯСНЮВАЛЬНА **ЗАПИСКА до дипломного** проекту_ **на тему:**
_____
_____ **Проектний матеріал складається з пояснювальної записки на** _____ **сторінках та графічного (презентаційного) матеріалу на** _____ **аркушах (слайдах) Дипломник** _____ (Челак Д.В.)
Керівник _____ (Кривченко Ю.В.)