Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Магістерська дипломна робота на тему:
«Система шифрування зображень. Частина 2. Підсистема розшифрування зображень.»

Виконав: студент 2 курсу групи 2БС-22м
спеціальності 125 Кібербезпека
_____ Максим ЧІЧАНОВСЬКИЙ

Керівник: зав. каф. ЗІ , д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
« 13 » 12 2023 р.

Опонент : к.т.н., доц. каф. ПЗ
_____ Вікторія ВОЙТКО
« 13 » 12 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«14» 12 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., проф.
Володимир ЛУЖЕЦЬКИЙ
«19» 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чічановському Максиму Юрійовичу

1. Тема роботи: «Система шифрування зображень. Частина 2. Підсистема розшифрування зображень.»
Керівник роботи: Лужецький Володимир Андрійович , зав. каф. ЗІ , д. т. н., проф.,
затверджені наказом ректора. ВНТУ від 18 вересня 2023 №244
2. Строк подання студентом роботи: 13 грудня 2023р.
3. Вихідні дані до роботи:
   - Метод шифрування – на основі розподілу секрету.
   - Функції розподілу секрету: (n,n) n= 3,4,5,6.
   - Мова програмування – Phyton.
   - Метод що реалізується – шифрування зображення.
   - Формат зображення: .png, .jpg, .HEIC Обсяг файлу зображень – не більше 15 МБ

   Зміст текстової частини: Вступ. 1. Аналіз літературних джерел. 2. Розробка алгоритму відновлення файлів . 3. Розроблення програмного засобу. 4. Економічна частина. Висновки. Перелік використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Об'єкт та предмет дослідження (плакат А4). Мета та задачі дослідження (плакат А4). Метод розшифрування зображення (плакат А4). Схема перестановки байтів на основі ПВЧ (плакат А4). Алгоритм розшифрування зображення (плакат А4). Алгоритм ініціалізації генератора (плакат А4). Результати тестування засобу при відсутності завад (плакат А4). Результати тестування засобу при наявності завад(плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата: завдання видав	Підпис, дата: завдання прийн
1	Володимир ЛУЖЕЦЬКИЙ, зав. каф. ЗІ, д. т. н., проф.	1909	
2	Володимир ЛУЖЕЦЬКИЙ, зав. каф. З , д. т. н., проф.	1909	
3	Володимир ЛУЖЕЦЬКИЙ, зав. каф. ЗІ, д. т. н., проф.	1909	
4	Ольга РАТУШНЯК, к. т. н., доц. каф. ЕПВМ	1909	

7. Дата видачі завдання 1 вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2023 – 10.09.2023	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	10.09.2023 – 15.09.2023	
3	Розробка рішень	16.09.2023 – 22.09.2023	
4	Розробка модуля програмного засобу	23.09.2023-12.10.2023	
5	Практична реалізація, моделювання, експериментування, результати	14.10.23-16.11.2023	
6	Розробка розділу економічного обґрунтування доцільного розробки	17.11.23-19.11.23	
7	Аналіз виконання ІЗ, висновки	20.11.2023 – 24.11.2023	
8	Оформлення пояснювальної записки	25.11.2023 – 30.11.2023	
9	Попередній захист та доопрацювання МКР	28.11.2023 – 10.12.2023	
10	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
11	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Максим ЧІЧАНОВСЬКИЙ

Керівник роботи _____ Володимир ЛУЖЕЦЬКИЙ

АНОТАЦІЯ

Магістерська кваліфікаційна робота складається з 93 сторінок формату А4, на якій є 18 рисунків, 7 таблиць, 60 формули, список використаних джерел містить 42 найменування

Магістерська кваліфікаційна робота присвячена розробці методу та засобу захисту зображень. Акцент роботи робиться на використанні генераторів перестановок та псевдовипадкових послідовностей для створення ефективної системи розподілу та відновлення секрету, придатної для захисту конфіденційної інформації у вимогливих умовах обробки та збереження зображень. У роботі проаналізовано сучасні методи захисту зображень а також основні загрози кібербезпеки для таких застосунків. У межах роботи виконано опис методу відновлення зображення. Програмний засіб структурований та ефективний. В економічній частині оцінено витрати на розробку.

Ключові слова: захист зображень, відновлення зображень, генератор перестановок, псевдовипадкові послідовності, загрози кібербезпеки.

ABSTRACT

The master's thesis consists of 93 pages of A4 format, on which there are 18 figures, 7 tables, 60 formulas, the list of used sources contains 41 names.

The master's thesis is devoted to the development of the method and the whole picture of protection. Emphasis is placed on the use of permutation generators and pseudorandom components to create an efficient secret distribution and recovery system suitable for protecting confidential information under demanding image processing and storage conditions. The paper analyzes modern methods of protection, as well as the main cyber security threats for such applications. The description of the method of image restoration is included in the work. The software tool is structured and highly efficient in image processing and storage. Development costs are estimated in the economic part.

Keywords: image protection, image restoration, permutation generator, pseudorandom effects, cyber security threats

ЗМІСТ

ВСТУП

У наш час інформаційні технології стають необхідною складовою практично кожної сфери життя, і в зв'язку з цим зростає актуальність питань забезпечення безпеки обробки та зберігання конфіденційної інформації. Зокрема, важливим завданням є захист зображень від несанкціонованого доступу та використання.

Сучасний світ переживає бурхливий ріст обсягів цифрової інформації, що включає в себе не лише фотографії та зображення, а й велику кількість відео та графічних матеріалів. Забезпечення надійного захисту та можливості відновлення стає важливим завданням в умовах постійної загрози втрати чи пошкодження цих цінних даних.

У сферах, де освітні та дослідницькі матеріали містять важливі зображення, забезпечення їхньої конфіденційності є важливою вимогою. Медичні дані, наукові дослідження та технічні проекти вимагають ефективних методів відновлення зображень під час захисту конфіденційності.

Зростання кількості кібератак та витоків інформації наводить на необхідність вдосконалення методів захисту зображень. Відновлення зображень стає важливим елементом відновлення цілісності даних внаслідок кібератак та непередбачених ситуацій. Завдяки швидкому розвитку алгоритмів та обчислювальних можливостей, сучасні методи відновлення зображень можуть забезпечувати високий рівень точності та ефективності. Це стимулює подальше дослідження та розвиток в даній галузі.

Отже, враховуючи сучасні виклики та можливості, подальші дослідження та розробка систем захисту та відновлення зображень є настільки важливими, що їхнє вдосконалення може суттєво покращити безпеку та збереження цифрових зображень у різних областях використання.

Метою даної роботи є пришвидшення процесу розшифрування зображення.

Завдання на магістерську кваліфікаційну роботу:

1. Проаналізувати відомі методи візуальні криптографії.
2. Розробити метод розшифрування зображень.
3. Розробити програмний засіб для розшифрування зображень.
4. Дослідити вплив завад на результат розшифрування зображень.

Об’єктом дослідження є процес захисту зображення.

Предмет дослідження є метод та засіб для розшифрування зображень.

Наукова новизна роботи полягає в тому, що розроблено метод розшифрування зображень, який відрізняється від відомих тим, що передбачає відновлення конфіденційної інформації з певної кількості частин з використанням операцій перестановок і який забезпечує пришвидшення процесу розшифрування.

Практичною цінністю роботи є програмний засіб, що є складовою системи шифрування зображень.

1 АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Основна ідея візуальної криптографії

Візуальна криптографія є потужною технікою, яка поєднує поняття ідеальних шифрів і обміну секретами в криптографії з поняттями растрової графіки. Двійкове зображення можна розділити на частки, які можна скласти разом, щоб приблизно відновити вихідне зображення. Схема обміну секретами дозволяє розподіляти секрет між сторонами, так що лише попередньо визначені авторизовані набори зможуть реконструювати секрет. Секрет, з точки зору візуальної криптографії, може бути реконструйований візуально шляхом накладання часток [1].

Візуальна криптографія дозволяє передавати візуальну інформацію, і багато аспектів цієї галузі охоплено, включаючи її початок до сучасних методів, які використовуються та активно досліджуються сьогодні [1].

Дуже бажано мати можливість приховати таку інформацію, як особисті дані. Коли дані приховані в одному зображенні (так званому спільному ресурсі), його неможливо ідентифікувати. Хоча частини зображення розділені, дані абсолютно не послідовні. Кожне зображення містить різні дані, і, об’єднавши їх, секрет можна легко відновити. Кожен з них покладається на одного для розшифрованої інформації. Ніхто не може розшифрувати інформацію, що міститься в будь-якому загальному ресурсі. Коли частинки згруповані разом, декодування можливе, коли частинки знаходяться одна на одній. У цей момент інформація доступна негайно. Розшифрування повідомлення не вимагає жодної обчислювальної потужності. Все декодування виконується зоровою системою людини (ЗС) [1].

Обмін секретами за допомогою візуальної криптографії відрізняється від типового обміну криптографічними секретами. Останнє дозволяє кожній стороні зберігати частину таємниці та надає можливість дізнатися принаймні частину таємниці, тоді як перше це суворо забороняє. Шифрування з використанням кількох ключів є можливим рішенням. Однак це рішення вимагає великої

кількості ключів, тому управління такою схемою стає клопітким, як продемонстрував Шамір [1].

У 1979 році Аді Шамір опублікував статтю під назвою «Як поділитися секретом» [1].

Обмін зображеннями є підмножиною обміну секретами, оскільки він діє як особливий підхід до загальної проблеми обміну секретами. Таємниці в даному випадку - це приховані образи. Кожен секрет обробляється як число, що дозволяє використовувати конкретну схему кодування для кожного джерела секретів. Без проблеми зворотного перетворення цифри можуть бути неправильно інтерпретовані для відображення справжнього значення секрету. Спільне використання зображень визначає схему, ідентичну до загального обміну секретами. В (k , n) обмін зображеннями, зображення, яке містить таємницю, поділяється на n шматків (відомих як частки), і дешифрування є абсолютно невдалим, якщо принаймні k частини збираються та накладаються. Візуальна криптографія була спочатку винайдена та запроваджена Моні Наор та Аді Шаміром у 1994 році на конференції Eurocrypt. Візуальна криптографія – це «новий тип криптографічної схеми, яка може декодувати приховані зображення без будь-яких криптографічних обчислень». Як випливає з назви, візуальна криптографія пов’язана із зоровою системою людини. Коли k сегментів складені разом, розшифровкою займаються людські очі. Це дозволяє будь-кому використовувати систему без знання криптографії та без виконання будь-яких обчислень. Це ще одна перевага візуальної криптографії перед іншими популярними схемами умовно безпечної криптографії. Механізм дуже безпечний і дуже простий у впровадженні. Електронним секретом можна поділитися безпосередньо, або ж секрети можна роздрукувати на прозорих плівках і накласти, відкриваючи секрет [2].

Початкова реалізація Наора та Шаміра передбачає, що зображення або повідомлення є набором чорних і білих пікселів, кожен піксель обробляється окремо, і слід зазначити, що білий піксель представляє прозорий колір. Одним із недоліків цього є те, що процес дешифрування є втратним, область, яка страждає

через це, – це контраст. Контраст дуже важливий у візуальній криптографії, оскільки він визначає чіткість відновленого секрету зоровою системою людини. Відносна різниця у вазі Хеммінга між представленням білих і чорних пікселів означає втрату контрасту відновленого секрету. Вага Хеммінга пояснюється далі. Новіші схеми, які обговорюються пізніше, стосуються сірих і кольорових зображень, які намагаються мінімізувати втрату контрасту за допомогою цифрового пів тонування [3].

Напівтонування дозволяє зображення безперервних тонів, яке може складатися з нескінченного діапазону кольорів або відтінків сірого, бути представленим у вигляді бінарного зображення. Різні розміри точок і відстань між ними ці точки створюють оптичну ілюзію. Саме ця ілюзія дозволяє людському оку змішувати ці точки, завдяки чому на півтонове зображення виглядає як суцільне тональне зображення. Через те, що цифрове напівтонування само по собі є процесом із втратами, неможливо повністю відновити оригінальне секретне зображення [4].

Завдання шифрування формулюється як задача обміну секретами k з n. З огляду на зображення чи повідомлення, прозорі плівки генеруються так, що вихідне зображення (повідомлення) є видимим, якщо воно є k з них складені разом. Зображення залишається прихованим, якщо менше ніж k прозорі плівки складені разом. Кожен піксель відображається всередині модифіковані версії (відомі як спільні ресурси) на прозорість. Сегменти є сукупністю чорних та білих субпікселів, розташованих близько одне до одного. Структуру можна описати булевою матрицею C розміру $m \times n$. Структуру C можна описати так: $\mathrm{C} = (\mathrm{C}_{ij})_{m \times n}$, де C_{ij} дорівнює 1 або 0 відповідно сегмент чорний або білий.

Важливими параметрами схеми є [4]:

1. m-кількість пікселів у сегменті. Це означає втрату роздільної здатності від вихідного зображення до відновленого.
2. a- відносна різниця ваги між об’єднаними сегментами, які проходять від білого та чорного пікселів у вихідному зображенні, тобто втрата

контрасту.

3. y - розмір матриці C_0 і C_1 . C_0 відноситься до субпікселів візерунків у сегментах для білого пікселя та C_1 , відноситься до шаблонів субпікселів у спільних ресурсах для чорного пікселя.

Вага Хемінга *H(V) ORed m-вектор V* інтерпретується зоровою системою таким чином [4].

Чорний піксель інтерпретується, якщо $X(V) \leq d$ і білий, якщо $X(V) < d$ – для деякого фіксованого порогу $1 \leq d \leq m$ і відносна різниця dm $d > 0$.

Конструкцію спільних ресурсів можна чітко проілюструвати схемою візуальної криптографії 2 із 2 (широко відомою як (2,2-VCS) [5]. Наступні матриці 2×2 визначаються як 2 матриці:

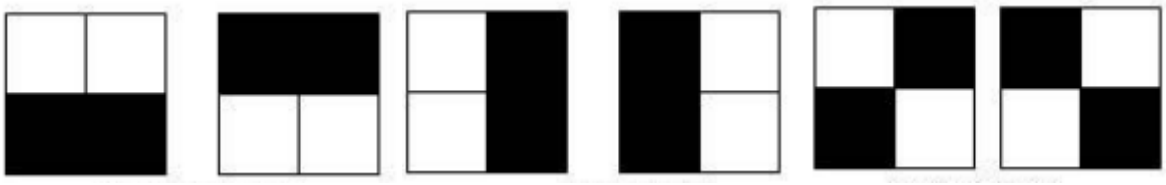
C_0 = {всі матриці отримані шляхом перестановки стовпців{ $\begin{vmatrix} 1 & 1\,0 & 0 \\ 1 & 1\,0 & 0 \end{vmatrix}$}

C_1 = {всі матриці отримані шляхом перестановки стовпців{ $\begin{vmatrix} 1 & 1\,0 & 0 \\ 0 & 0\,1 & 1 \end{vmatrix}$}

Завдяки цьому піксельному розширенню один піксель вихідного зображення розширюється на чотири пікселі. Сегменти можуть бути створені в такий спосіб [5]:

1. Якщо піксель вихідного зображення білий, навмання вибрати його візерунок із чотирьох пікселів для обох сегментів.
2. Якщо піксель вихідного зображення чорний, вибрати додаткову пару візерунків, тобто візерунки того самого стовпця. Візуальне представлення різних типів шаблонів сегментів представлено на рис.1.

Якщо прозорі плівки накладені та субпікселі правильно вирівняні, то чорні пікселі в об’єднаних сегментах представленні логічним значенням АБО рядків у матриці. Пікселі можна розташувати різними способами в середині матриці.



Горизонтальний сегмент Вертикальний сегмент Діагональний сегмент

Рисунок 1.1 – Різні типи шаблонів пікселів, які використовуються під час створення спільних ресурсів VC

Оскільки окремі спільні ресурси не дають підказки про те, чи є певний піксель чорним або білим, то стає неможливим розшифрувати сегмент, незалежно від того, скільки доступно обчислювальних потужностей.

На рис.1.2 показано реалізацію та результати базового візуального зображення (2,2)-VCS [5]. Тут наведено секретне зображення, два згенерованих спільних ресурси та відновлення секрету після накладання першого та другого спільних ресурсів.

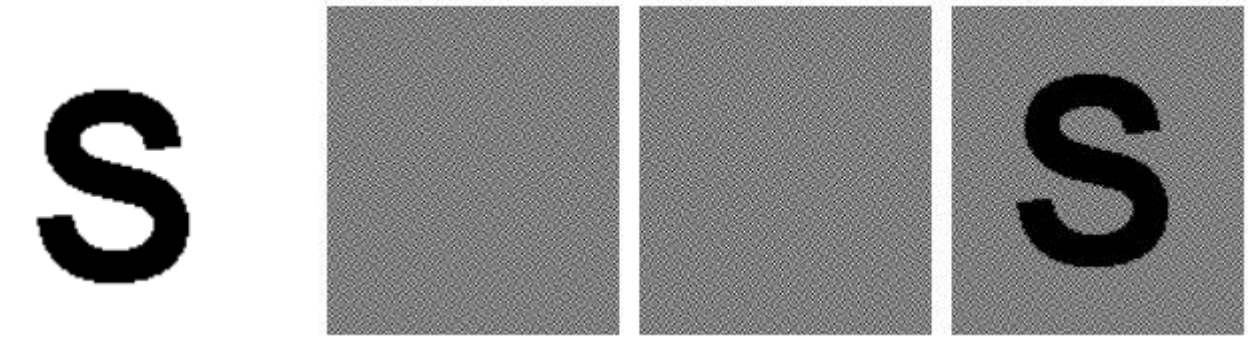


Рисунок 1.2 – Результати традиційної схеми візуальної криптографії

1.2 Розширена схема візуальної криптографії

Схема розширеної візуальної криптографії (EVCS), запропонована Ateniese та ін. [6] базується на структурі доступу, яка містить два типи наборів: структура кваліфікованого доступу Γ_{Qual} і структура забороненого доступу Γ_{Forb} в наборі n учасників. Однак для будь-якого набору, який є членом структури

забороненого доступу і не має інформації про спільний секрет, це означає, що жодна корисна інформація не може бути отримана зі стеку учасників.

Основна відмінність між базовою візуальною криптографією та розширеною візуальною криптографією полягає в тому, що на кожному спільному ресурсі можна переглядати розпізнаване зображення; після накладання спільних ресурсів (за умови, що вони є частиною структури кваліфікованого доступу), зображення на спільних ресурсах зникне, а секретне повідомлення стане видимим [6].

Розширені схеми візуальної криптографії дозволяють створювати схеми візуального секретного обміну, в яких спільні ресурси є значущими на відміну від випадкового шуму на спільних ресурсах. Після накладання наборів спільних ресурсів ця інформація зникає, а секрет відновлюється. Це основа для розширеної форми візуальної криптографії (EVCS) [6].

З EVCS перші n сегментів мають бути зображеннями чогось на зразок автомобіля, човна чи собаки, певної форми значущої інформації. Секретне повідомлення або зображення зазвичай обробляються останніми $n+1$. Для цього потрібна техніка, яка має враховувати колір пікселя в секретному зображенні, яке ми хочемо отримати, тому, коли n сегменти накладаються, їхні окремі зображення зникають, а секретне зображення видно. У загальному вигляді це має таке позначення $C_C^{C_1 \dots C_n}$, з $C, C_1, C_n \in \{b,w\}$, де сукупність матриць, з яких ми можемо вибрати матрицю для визначення сегментів, задано C_i буде кольором і невинним зображенням, a,c — колір таємного зображення. Щоб реалізувати цю схему, потрібно мати 2^n пар таких колекцій, по одній для кожної можливої комбінації необхідно згенерувати білі та чорні пікселі на n оригінальних зображеннях [6].

Передбачається, що невідома інформація про значення пікселів вихідного зображення, яке приховується. Єдине, що відомо, це те, що пікселі можуть бути чорними або білими. Розподіл імовірностей пікселів невідомий. Немає способу визначити, чи чорний піксель більш імовірний, ніж білий піксель. Для

шифрування зображень необхідно виконати три умови [6].

По-перше, зображення, які належать до кваліфікованої структури доступу до набору, повинні під час накладання розкривати секретне зображення.

По-друге, при огляді сегментів не повинно бути жодних натяків на те, який секрет приховано в сегментах . Нарешті, зображення в сегментах не повинно бути змінено в жодному разі, тобто після *n* оригінальні зображення були закодовані, вони все ще повинні бути розпізнаними користувачем.

Найпростішим прикладом є (2,2)-проблема EVCS. Матриці $C_C^{C_1\ C_2}$ отримані шляхом перестановки стовпців наступних матриць [6]:

$$C_w^{ww}\ \begin{vmatrix}1 & 0\,0 & 1\\ 1 & 0\,0 & 0\end{vmatrix}\ \text{i}\ C_b^{ww}\ \begin{vmatrix}1 & 0\,0 & 1\\ 0 & 1\,1 & 0\end{vmatrix}$$

$$C_w^{wb}\ \begin{vmatrix}1 & 0\,0 & 1\\ 1 & 0\,1 & 1\end{vmatrix}\ \text{i}\ C_b^{ww}\ \begin{vmatrix}1 & 0\,0 & 1\\ 0 & 1\,1 & 1\end{vmatrix}$$

$$C_w^{bw}\ \begin{vmatrix}1 & 0\,1 & 1\\ 1 & 0\,1 & 0\end{vmatrix}\ \text{i}\ C_b^{bw}\ \begin{vmatrix}1 & 0\,1 & 1\\ 0 & 1\,1 & 1\end{vmatrix}$$

$$C_w^{bb}\ \begin{vmatrix}1 & 0\,1 & 1\\ 1 & 0\,1 & 1\end{vmatrix}\ \text{i}\ C_b^{bb}\ \begin{vmatrix}1 & 0\,1 & 1\\ 0 & 1\,1 & 1\end{vmatrix}$$

Також можна перевірити, що для (2, 2)-EVCS значення контрасту, досягнуті для обох спільних ресурсів та відновленого секретного образу мають значення ¼.

На рис.1.3 наведено приклад (2, 2)-EVCS. Бачимо, що дві значущих сегмента генеруються з базових зображень. Під час створення сегменту, секрет кодується між кожним із спільних ресурсів [6].

Після накладання кожного сегменту секрет повністю відновлюється, а кожен сегмент, який має значення у формуванні, зникає.

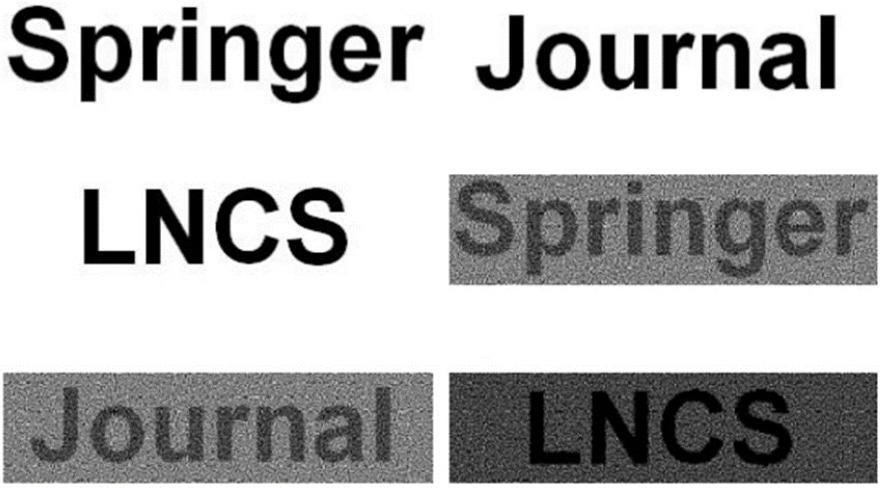


Рисунок 1.3- Результати застосування розширеної схеми візуальної криптографії

Щоб використовувати цю розширену схему візуальної криптографії, необхідно визначити загальну конструкцію. Ateniese та ін. [5] розробили механізм, за допомогою якого можна генерувати сегменти для схеми.

Сильнішою моделлю безпеки для EVCS є модель, у якій користувач може перевіряти спільні ресурси, пов'язані із забороненою підмножиною, тобто секретне зображення все одно залишатиметься повністю прихованим, навіть якщо всі *n* сегменти раніше відомі користувачеві. Системний підхід для повного вирішення загального (*k,n*) полягає в такому [6].

Для кожного набору структур доступу нехай P = {1, ⋯ , n} представляє набір елементів і нехай 2P позначає множину всіх підмножин P. Нехай $\Gamma_{Qual}/\Gamma_{Forb}$ бути набором кваліфікованих/заборонених наборів. Пара називається структурою доступу схеми. Будь-який кваліфікований набір може відновити спільне зображення шляхом стекування прозорих плівок його учасників, тоді як будь-який заборонений набір не має інформації про спільне зображення. Це розширення узагальнює вихідну проблему спільного доступу до секрету [6].

У [7] автори пропонують новий метод реалізації *(k, n)*-VCS, який є кращим по відношенню до розширення пікселів, ніж запропонований Наором і Шаміром.

1.3 Візуальна криптографія з незмінним розміром

Одну з перших статей, в якій розглядається незмінний розмір зображення VC, запропонували Ito та ін. [8]. Як описано раніше, традиційні схеми візуальної криптографії використовували піксельне розширення, хоча багато хто працював над тим, як це покращити [9].

Схема Ito [10] усуває потребу в цьому піксельному розширенні. У схемі використовується традиційна *(k, n)* схема де *n* (кількість субпікселів у спільному пікселі) дорівнює одиниці. Структура цієї схеми описується булевим n-вектором V = [v1, ⋯ , vn]T , де vi представляє колір пікселя в і-му спільному доступі зображення. Якщо vi = 1, то піксель чорний, інакше, якщо vi = 0, то піксель є білий. Щоб відновити секрет, до пікселів у V застосовується традиційне АБО. Відновлений секрет можна розглядати як різницю ймовірностей, з якою чорний піксель у реконструйованому зображенні генерується з білого та чорного пікселів на секретному зображенні. Як і у традиційній візуальній криптографії, n × m наборів для схеми необхідно визначити матриці:

C_0 = {всі матриці отримані шляхом перестановки стовпців{ $\begin{vmatrix} 1 & 1\,0 & 0 \\ 1 & 1\,0 & 0 \end{vmatrix}$}

C_1 = {всі матриці отримані шляхом перестановки стовпців{ $\begin{vmatrix} 1 & 1\,0 & 0 \\ 0 & 0\,1 & 1 \end{vmatrix}$}

Оскільки ця схема не використовує піксельне розширення, m завжди дорівнює одиниці, а значення *n* залежить від типу схеми, що використовується, наприклад, схема *(2, 3)*, $n = 3$. Найважливішою частиною будь-якої схеми обміну візуальними секретами є контраст. Чим менший контраст, тим важче візуально відновити секрет. Контраст для цієї схеми визначається наступним чином: $\beta = |p_0 - p_1|$, де p_0 і p_1 — ймовірності, з якими генерується чорний піксель на реконструйованому зображенні з білого та чорного пікселя на секретному зображенні. Використовуючи визначені набори матриць C_0 і C_1 і контраст $\beta = 13$, булевих матриць $n \times m$ S0 і S1 вибираються випадковим чином із C0 і C1 відповідно [10]:

$$S_0 = \begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 0, \\ 1 & 0 & 0 \end{matrix} \quad S_0 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Щоб поділитися білим пікселем, вибирається один зі стовпців S0, для спільного використання чорний піксель вибирається як один зі стовпців S1. Це обраний вектор-стовпець $V = [v1, \cdots, v_n]^T$ визначає колір кожного пікселя у відповідному спільному файлі зображення. Кожен vi інтерпретується як чорний, якщо $vi = 1$, і як білий, якщо $vi = 0$. Спільне використання, наприклад, чорний піксель, один стовпець вибирається навмання в $S1$, в результаті чого наступний вектор:

$$V = \begin{matrix} 0 \\ 1 \\ 0 \end{matrix}$$

Отже, i-й елемент визначає колір пікселів у i-му спільному в доступі. Таким чином, у цьому (2, 3) прикладі v_1 є білим на першому спільному зображенні, v_2 – білим чорний на другому спільному зображенні, а на третьому спільному зображенні v_3 білий.

Цей процес повторюється для всіх пікселів секретного зображення, що призводить до остаточного набору спільних ресурсів. На рис.1.4 наведено приклад на основі схеми (2,2).

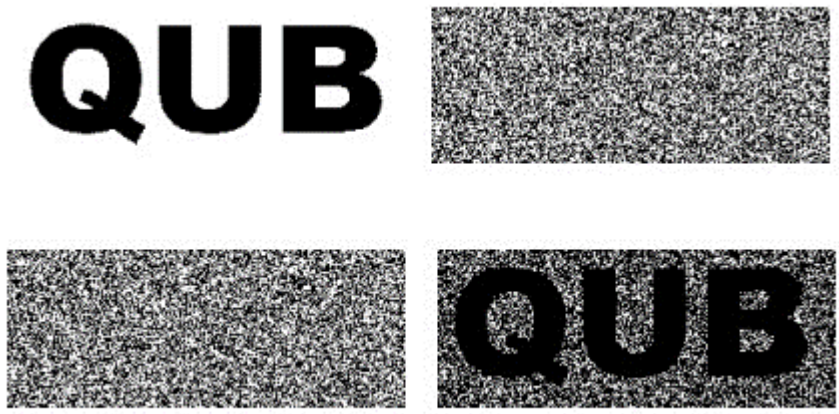


Рисунок 1.4 - Результат схеми візуальної криптографії з незмінним розміром

Імовірнісний метод роботи з сегментами, незмінними за розміром, запропоновано в [10], у якому частота білих пікселів використовується для

показу контрасту відновленого зображення. Схема не розширюється і може бути легко реалізована на основі звичайних схем обміну візуальними секретами (VSS). Термін «нерозширюваний» означає, що розміри оригінального зображення та тіней однакові. Як обговорювалося раніше, багато представлених схем передбачають розширення пікселів. Дослідники досліджували цю сферу та виявили, що вона є вартою дослідницької теми [11,12]. Це веде до пов'язаної теми в рамках схем з інваріантом розміру, а саме співвідношення сторін.

Спільний доступ до секрету з інваріантним співвідношенням сторін представлено Янгом і Ченом [13]. Ця схема обміну секретом із незмінним співвідношенням сторін різко зменшує кількість додаткових субпікселів, необхідних для створення секрету. Це приводить до менших сегментів, ближчих до розміру оригінального секрету, а також зберігає співвідношення сторін, таким чином уникаючи спотворень під час реконструкції секрету. Крім того, цю проблему можна розглянути з протилежного боку, регулюючи загальний розмір сегменту та контрасту. Представлена схема регулювання розміру [14], яка дозволяє користувачеві вибрати відповідний розмір спільного ресурсу, практичний для поточного використання спільних ресурсів. Якщо якість і контраст мають значення, розмір спільних ресурсів збільшиться, тоді як може статися навпаки, якщо ці речі не є надто важливими для конкретної програми користувача.

1.4 Водяні знаки

Практичне застосування візуальної криптографії – це водяні знаки. Мемон і Вонг [15] пропонують різні методи, за допомогою яких ці водяні знаки можуть бути застосовані до зображень. Проста схема вставки водяного знака полягає в тому що він вбудовано в найменш значущий біт зображення [16]. Однак вона не є надійною, оскільки водяний знак можна легко знищити. Надійніша схема повинна впоратися з ущільненням, фільтрацією та скануванням зображень із втратами. Ідея випадкового шуму [17] використовується на кольорових зображеннях, щоб ускладнити видалення водяного знака. Криптографічні

функції, такі як геш MD5, також використовувалися для покращення функцій безпеки, коли мова йде про вбудовування даних в межах зображень [18].

Схема авторського права на цифрове зображення, заснована на візуальній криптографії, представлена в [20]. Метод простий і ефективний як для вбудовування, так і для пошуку водяних знаків, а також є достатньо надійним, коли зображення з водяним знаком ущільняється. Після ущільнення водяний знак все ще можна відновити та перевірити. Однак схема не є надійною з точки зору незначних модифікацій зображення з водяним знаком. Точне відновлення неможливо. Інша проблема полягає в тому, що водяний знак можна успішно відновити із зображення, яке демонструє певну схожість з оригіналом, навіть якщо зображення не є оригіналом.

Замість схеми випадкового вибору пікселів, запропонованої в [20], [21] пропонує схему, за якою вибираються певні пікселі з вихідного зображення. Одна проблема з цією невипадковою схемою полягає в тому, що будь-які зміни, внесені в оригінал, такі як пошкодження зображення, будуть відображені у відновленому водяному знаку. Водяний знак усе ще можна розпізнати, але спотворення помітні. Однак важливою частиною цієї схеми є те, що сам водяний знак невидимий. Це означає, що вихідне зображення виглядає точно так само, як зображення з водяним знаком. Схема стійка до незначних змін у зображенні, але ці зміни присутні у відновленому водяному знаку. Ключ, який використовується для відновлення водяного знака, залежить від безпеки схеми. Якщо використовується маленький ключ (8 біт), схема не буде настільки безпечною, як ключ довжиною 128 біт.

Схема водяних знаків на основі VC [22]. Ця вдосконалена схема підтримує чорні та білі зображення, а також кольорові зображення та стійкі до масштабування та обертання зображення з водяним знаком. Надійне відновлення водяного знака також можливе після зображення було зіпсовано. Як і в інших розглянутих раніше схемах, ця схема також є залежною від ключа. Без ключа неможливо відновити водяний знак. Одним із найнадійніших способів приховати секрет у природних зображеннях є типове використання візуальної

криптографії, заснованої на напівтонових техніках. Ідеальна схема надзвичайно практична і може розкривати секрети без участі комп'ютера. Останні сучасні водяні знаки [23] можуть приховати водяний знак в документи, які не потребують спеціального ключа для його отримання. Видалення ключа є досить важливою, оскільки це ще більше підвищує безпеку та надійність процесу нанесення водяних знаків.

Хо і Чен [24] реалізували асиметричну схему водяних знаків на основі візуальної криптографії. Для зберігання водяного знака створюються два спільні ресурси. Один вбудовано в зображення обкладинки, а інше зберігається як секретний ключ для вилучення водяного знака. Водяний знак витягується за допомогою традиційного укладання властивості візуальної криптографії. Водяний знак надійний тим, що це важко змінити або видалити і може витримати низку атак.

1.5 Використання візерунків Муар

Потенційним застосуванням візуальної криптографії є її використання в поєднанні з візерунками Муар. Муарові візерунки [25] (або смуги) індукуються під час виявлення такого шару, як точкова растра або лінійна решітка, накладається на форму, що періодично повторюється. На отриманий візерунок муару впливає зміна будь-якого з наступних геометричних параметрів, що характеризують окремі структури сітки, а саме період, орієнтація та форма [26,27,28]. Незалежно від того, чи використовується точковий екран чи лінійна гратка, обидва створюють муарові смуги з однаковими геометричними властивостями [29].

Розкриваючий шар містить горизонтальні чорні лінії (лінійну решітку), між цими лініями прозорий білий простір. Коли розкриваючий шар накладається, фігури, які з'являються, є збільшеними версіями повторюваного візерунка. Ця властивість збільшення [30,31] може бути використана як метод пошуку прихованих спільних ресурсів VC у візерунку Муар.

Десмедт і Лі [32] надають схему візуальної криптографії з використанням

візерунків Муар, яка задовольняє як секретність, так і анонімність. Муарові візерунки виникають, коли високочастотні решітки поєднуються разом для створення низькочастотних решіток. Це різниця в цих високих частотах які створюють візерунки муару. На рис 1.5 показаний приклад муарових візерунків [33].

Криптографічна модель Муара виглядає наступним чином: вбудоване (секретне) зображення, на якому є рандомізовані два сегменти, відомі як попередні сегменти. Кожен з них незалежний від вихідного зображення. Виконання спільних операцій XOR відновить оригінал.

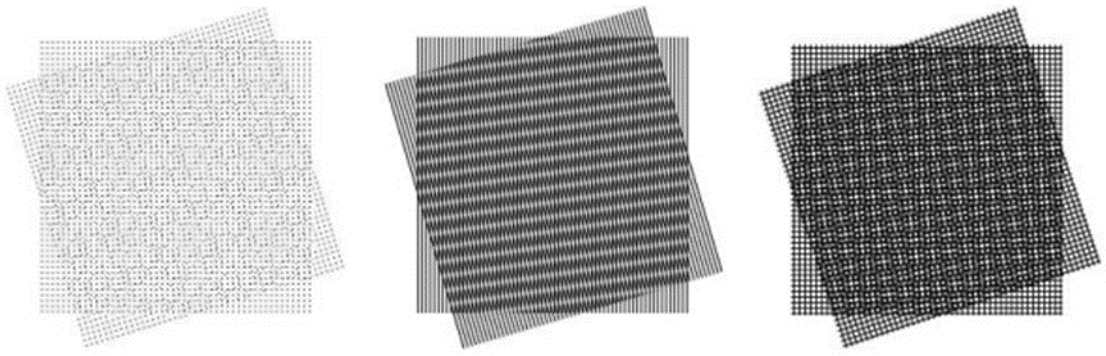


Рисунок 1.5- Муарові візерунки, створені різними стилями

Далі, алгоритм приховування бере зображення обкладинки та поєднує його з кожним із попередні сегментів окремо. Його результатом є останні два сегменти, які використовуються для розкриття вихідного вбудованого зображення. Ці отримані сегменти виглядають так само, як і вхідні дані зображення обкладинки, яке використовується.

Існують три різні схеми Муара, запропоновані Десмедтом і Лі [32]: обертання решітки, плавне обертання решітки та орієнтація точки. Проблема з обертанням решітки полягає в тому, що межа між різними обертовими областями в сегментах стають видимими. Однак ця схема дала дуже чітке розшифрування зашифрованого тексту. Інша проблема, а саме, артефакти, внесені в сегменти,

також виділяються багато і стають видимими. Вони зупинилися на остаточній схемі, точкової орієнтації, як обрану ними реалізацію. Крапки з сегментів перетворюються на «крапки» у формі ромба, це робить межі менш помітними, ніж круглі або еліптичні точки. Схема кодує білий піксель шляхом накладання двох квадратів на долі, крапки яких орієнтовані під різними кутами. Щоб закодувати чорний піксель, точкові візерунки використовуються під однаковим кутом. Це створює два різних візерунки муару для білих і чорних точок. Це означає, що ця схема використовує візерунки муару для відновлення таємного вбудованого зображення, а не традиційної схеми візуальної криптографії, які використовують рівень сірих квадратів для відновлення секрету.

Ці муарові візерунки можна використовувати в поєднанні з технологією голограм [33]. Це може забезпечити безпечні рішення для перевірки згенерованих голограм.

1.6 Спільне використання кількох секретів кольорів

Використання напівтонових і кольорових зображень як основи або обкладинки для спільного використання кількох секретів це цікава тема. Методи, запропоновані в [34], дозволяють використовувати менший набір спільних ресурсів (які можуть бути унікальними), які потрібно приховати за допомогою цих значущих кольорових зображень. Використання ідеї головного ключа здатне відновити всі секрети, які сформовано за окресленою схемою, вона використовується для покриття півтону або кольорове зображення, щоб розкрити таємниці. Секретні частки в цій справі вбудовані в зображення обкладинки, це допомагає усунути підозри, що будь-які шифрування відбулося або зображення навіть було змінено конкретний помітний спосіб. Рис.1.6 ілюструє застосування цієї схеми.

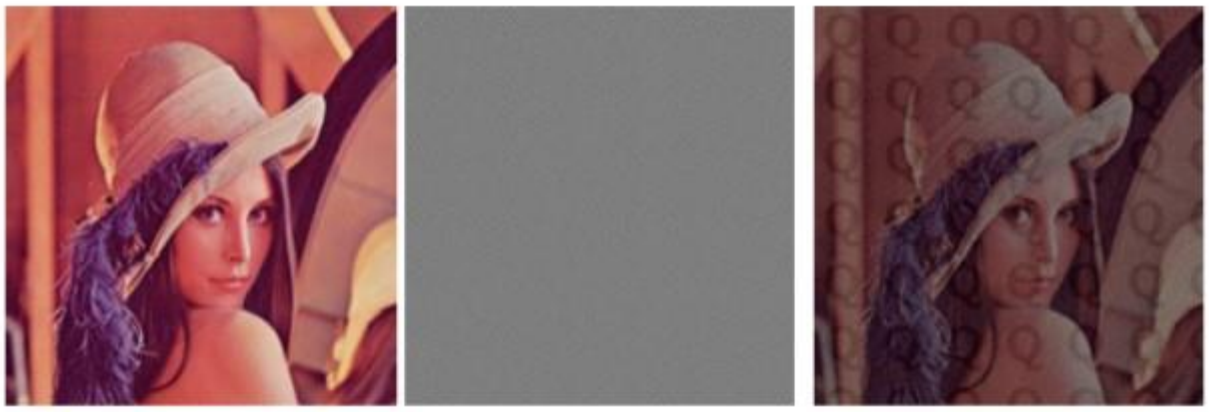


Рисунок1.6 - Об’єднання частини візуальної криптографії з кольоровим зображенням

1.6.1 Метод кольорових зображень

Початкове зображення представлено у форматі PNG і має роздільність 24 біти, де кожен колір (синій, зелений, червоний) складається з 8 біт інформації. Алгоритм передбачає розподіл секретного зображення на компоненти кольору (синій, зелений, червоний), які потім вбудовуються в менш значущі біти одного з контейнерних зображень. Іншими словами, після розподілення, кожен контейнер (R, G, B) міститиме в собі одну компоненту кольору секретного зображення [35].

Перед запуском алгоритму визначається кількість кожного з трьох відтінків в кожному зображенні-контейнері, щоб визначити, в якому з них слід приховати складову B, в якому - R, а в якому - G. Вибір контейнерів здійснюється так, щоб різниця у кольорах між кольоровим примітивом секретного зображення і контейнером була мінімальною. Далі два старші біти з кожного кольорового примітиву записуються в менш значущі біти відповідного контейнера та кольору. Два молодших біти в двох залишених кольорах занулюються. Ця операція повторюється для всіх пікселів [35].

У процесі відновлення зображення взято послідовно кожен піксель із кожного зображення-контейнера. Два менш значущі біти кожного кольору у цих пікселях стають більш значущими для об'єднання, оскільки під час розподілу менш значущі біти нерозподіленого кольору анулювалися. Таким чином, у

кожному контейнері залишається тільки одна колірна складова з нульовим значенням. Це дозволяє відновити колір відповідного пікселя секретного зображення з певною ступенем погрішності. Цю операцію повторюємо для всіх пікселів, отримуючи відновлене секретне зображення [35].

Для ефективного відновлення зображення, що було розподілено за вказаним методом, обов'язково потрібно наявність усіх трьох зображень-контейнерів, отриманих під час розподілу. У противному випадку, відновлення буде обмежено лише відтворенням колірного шуму, а не бажаного зображення.

У цьому методі присутній певний недолік. Під час розподілу секретного зображення відбувається його спотворення, оскільки використовуються лише два старші біти з восьми [35].

1.6.2 Метод розподілу зображень за допомогою Фур’є образу

Нехай у нас є матриця зображення у форматі RGB. Для отримання частотного представлення цього зображення можна використовувати дискретне перетворення Фур'є. Процес відновлення включає застосування оберненого дискретного перетворення Фур'є [36].

Припустимо, що маємо цифрове представлення кольорового зображення, виражене через три матриці (R, G, B), які утворюють тривимірний масив C = cat(3, R, G, B), де "cat" позначає конкатенацію. Якщо розмір матриць становить m × n, то масив C включає в себе m × n × 3 елементи. Кожен елемент ідентифікується трьома індексами C[m, n, z], де z = 1, 2, 3 відповідає матрицям R, G, B відповідно.

$$C\,[m, n, 1] = R\,[m, n],$$

$$C\,[m, n, 2] = G\,[m, n],$$

$$C\,[m, n, 3] = B\,[m, n].$$

У даному випадку використовуються три матриці, які отримали назви за відповідними кольорами: червоний (R), зелений (G), і синій (B) - це кольорові канали або компоненти. Зважаючи на властивість не комутативності конкатенації, порядок розташування матриць має важливе значення. Іншими словами, кольорове зображення можна визначити через трійку матриць розміром

mxn [36].

Покажемо для прикладу для компоненти *R*.

$$\boldsymbol{R} = \begin{vmatrix} f_{00}\, f_{01} & \cdots & f_{0n} \\ f_{10}\, f_{11} & \ddots & f_{1m} \\ \vdots & & \vdots \\ f_{n0}\, f_{n2} & \cdots & f_{nm} \end{vmatrix}$$

Кожна компонента цього представлення має форму f(x, y), де амплітуда f в будь-якій точці визначає інтенсивність або значення кольору, у даному випадку R - червоного кольору, а x і y представляють просторові координати. Задано, що f(x, y) при x = 0, 1, m-1 і y = 0, 1, n-1 представляє зображення розміру mxn. Таким чином, двовимірне дискретне перетворення Фур'є [36] визначається для цього контексту.

$$F(u,v) = \sum_{x=0}^{m+1} \sum_{y=0}^{m+1} f\,(x,y) e^{i}(\frac{ux}{m} + \frac{vy}{n})$$

Зворотне двохвимірне дискретне перетворення Фур’є:

$$f(x,y) = \frac{1}{n\,m} \sum_{x=0}^{m+1} \sum_{y=0}^{m+1} F\,(u,v) e^{i2\pi(\frac{ux}{m}+\frac{vy}{n})}$$

1.7 Шифрування зображень

Шифрування зображень є важливим інструментом для захисту візуальної інформації в сучасному цифровому світі. З розвитком технологій розуміння та впровадження методів шифрування зображень стає все більш важливим.

Шифрування зображень — це процес перетворення зображення у нечитабельний формат за допомогою алгоритмів шифрування та ключів. Цей захисний захід гарантує, що лише авторизовані особи зможуть отримати доступ до зображення та переглянути його [37].

Існує два основних методи шифрування зображень: симетричне та асиметричне [37].

Симетричне шифрування використовує один ключ як для зашифрування,

так і для розшифрування. Цей метод є ефективним, але вимагає безпечного методу обміну ключами.

Асиметричне шифрування використовує пару ключів, відкритий та закритий. Відкритий ключ використовується для зашифрування, а закритий — для розшифрування. Цей метод забезпечує більш високий рівень безпеки, але може бути менш ефективним, ніж симетричне шифрування.

Шифрування зображень використовується в різних сферах, включаючи [37]:

- безпечний зв'язок: шифрування зображень використовується для захисту конфіденційності зображень, що передаються через Інтернет або інші мережі;

- безпечне зберігання: шифрування зображень використовується для захисту зображень, що зберігаються на локальних комп'ютерах, серверах або хмарних платформах.

- медична візуалізація: шифрування зображень використовується для захисту медичних зображень, таких як рентгенівські знімки та МРТ.

- криміналістичний аналіз: шифрування зображень використовується для захисту доказів, таких як фотографії злочинних дій.

Методи шифрування зображень мають забезпечити:

- керування ключами: захист ключів шифрування має вирішальне значення для безпеки зашифрованих зображень.

Існує багато різних методів шифрування зображень, але деякі з найпоширеніших включають:

- шифрування блочного режиму;
- шифрування потокового режиму;
- шифрування полів.

Шифрування блочного режиму є одним з найпоширеніших методів шифрування зображень. Він працює, поділяючи зображення на блоки, а потім шифруючи ці блоки за допомогою шифру блочного режиму, такого як AES, DES

або 3DES [38].

Шифр блочного режиму - це алгоритм, який приймає на вхід блок даних і генерує на виході зашифрований блок даних. Шифр блочного режиму має ключ, який використовується для шифрування та розшифровування даних.

Існує багато різних режимів шифрування блочного режиму, які можна використовувати для шифрування зображень. Деякі з режимів включають:

- режим CBC (cipher block chaining);
- режим CFB (cipher feedback);
- режим OFB (output feedback);
- режим CTR (counter).

Режим CBC є одним з найпоширеніших режимів шифрування блочного режиму для шифрування зображень. Він працює, XOR-уючи попередній зашифрований блок з поточним блоком відкритого тексту перед шифруванням. Це забезпечує високу безпеку, оскільки кожний блок шифротексту залежить від всіх попередніх блоків.

Режим CFB працює, XOR-уючи поточний блок шифротексту з попереднім блоком відкритого тексту перед шифруванням. Це забезпечує меншу безпеку, ніж режим CBC, але він може бути більш ефективним [38].

Режим OFB працює, XOR-уючи поточний блок шифротексту з інкрементним лічильником перед шифруванням. Це забезпечує меншу безпеку, ніж режим CBC або режим CFB, але він може бути дуже ефективним [38].

Режим CTR працює, XOR-уючи поточний блок шифротексту з інкрементним лічильником перед шифруванням. Це забезпечує таку ж безпеку, як і режим CBC, але він може бути більш ефективним [38].

Шифрування блочного режиму має такі переваги: висока безпека, доступність.

Шифрування блочного режиму є ефективним методом шифрування зображень, оскільки він забезпечує високу безпеку. Однак він може бути менш ефективним, ніж шифрування потокового режиму, оскільки він вимагає додаткових обчислень для поділу зображення на блоки [39].

Шифрування потокового режиму працює, генеруючи потік ключів, який потім використовується для XOR-ування з відкритим текстом. Потік ключів генерується за допомогою шифру блочного режиму, такого як AES, DES або 3DES.

Існує багато різних режимів шифрування потокового режиму, які можна використовувати для шифрування зображень. Деякі з найпоширеніших режимів включають: режим ECB (electronic codebook), режим OFB (output feedback), режим CTR (counter).

Режим ECB є найпростішим режимом шифрування потокового режиму. Він працює, XOR-уючи відкритий текст з блоком ключа. Це забезпечує низьку безпеку, оскільки кожний блок шифротексту однаковий [39].

Режим OFB працює, XOR-уючи відкритий текст з послідовним блоком ключа. Це забезпечує більшу безпеку, ніж режим ECB, але він може бути менш ефективним [39].

Режим CTR працює, XOR-уючи відкритий текст з інкрементним лічильником. Це забезпечує таку ж безпеку, як і режим OFB, але він може бути дуже ефективним [39].

Шифрування потокового режиму є іншим методом шифрування зображень. Він працює, шифруючи зображення послідовно. Цей метод шифрування може бути швидшим за шифрування блочного режиму, але він може бути менш безпечним.

Шифрування потокового режиму може бути більш ефективним, ніж шифрування блочного режиму, доступний в багатьох реалізаціях.

Шифрування потокового режиму є ефективним методом шифрування зображень, оскільки він не вимагає додаткових обчислень для поділу зображення на блоки. Однак він може бути менш безпечним, ніж шифрування блочного режиму, оскільки він може бути більш сприйнятливим до атак [40].

Шифрування полів є менш поширеним методом шифрування зображень. Він працює, шифруючи зображення полем. Поле може бути розміром від 4 до 16 біт. Інший спосіб реалізації шифрування полів полягає в тому, щоб

використовувати шифр блочного режиму для шифрування кожного пікселя зображення. Цей спосіб може бути менш ефективним, ніж перший спосіб, але він може бути більш безпечним [40].

Шифрування полів є ефективним методом шифрування зображень, оскільки він може забезпечити високу безпеку. Однак він може бути менш ефективним, ніж шифрування блочного режиму або шифрування потокового режиму, оскільки він вимагає додаткових обчислень для шифрування кожного поля [40].

Вибір методу шифрування зображень залежить від таких факторів:

- безпека;
- ефективність;
- доступність.

Якщо для виконання завдань важлива безпека, то слід використовувати шифрування блочного режиму або шифрування полів. Якщо ж важлива ефективність, то слід використовувати шифрування потокового режиму.

2 РОЗРОБКА МЕТОДУ ВІДНОВЛЕННЯ ФАЙЛІВ ЗОБРАЖЕНЬ

2.1 Узагальнений опис методу відновлення

Основна ідея алгоритму відновлення секрету базується на здійсненні операцій, зворотних до розбиття секрету, з метою отримання вихідного зображення. Для оцінювання коректності відновлення використовується накладання шуму на сегменти, що були отримані внаслідок алгоритму розподілення секрету з використанням генератора псевдовипадкових чисел.

Процес відновлення секрету має такі етапи:

1. Вибір сегментів: обираються сегменти, які були отримані в результаті розподілення секрету.

2. Зчитування байтів за допомогою генератора псевдовипадкових чисел: для кожного сегмента використовується генератор псевдовипадкових чисел для зчитування байту та запису його в масив вихідного зображення.

3. Зчитування розмірності та пікселів із кожного сегмента: з сегментів витягуються дані щодо розмірності та всіх пікселів з компонентами кольору (червоний, зелений, синій).

4. Створення масиву зі складових кольору: створюється масив, з якого обирається оригінальна складова кольору з кожного сегмента. Кожен піксель на новому зображенні складається з компонент кольору інших зображень.

5. Збірка вихідного зображення: з сегментів обираються пікселі, які послідовно записуються в масив, щоб сформувати вихідне зображення.

Кожен піксель включає три складові: інтенсивність червоного (R), синього (B) та зеленого (G) кольорів. Описані кроки забезпечують відновлення оригінального зображення шляхом об'єднання інформації з різних сегментів, отриманих під час розподілення секрету. Узагальнена схема роботи алгоритму відновлення зображена на рис 2.1 . S1 це сегмент з потоку переміщених пікселів

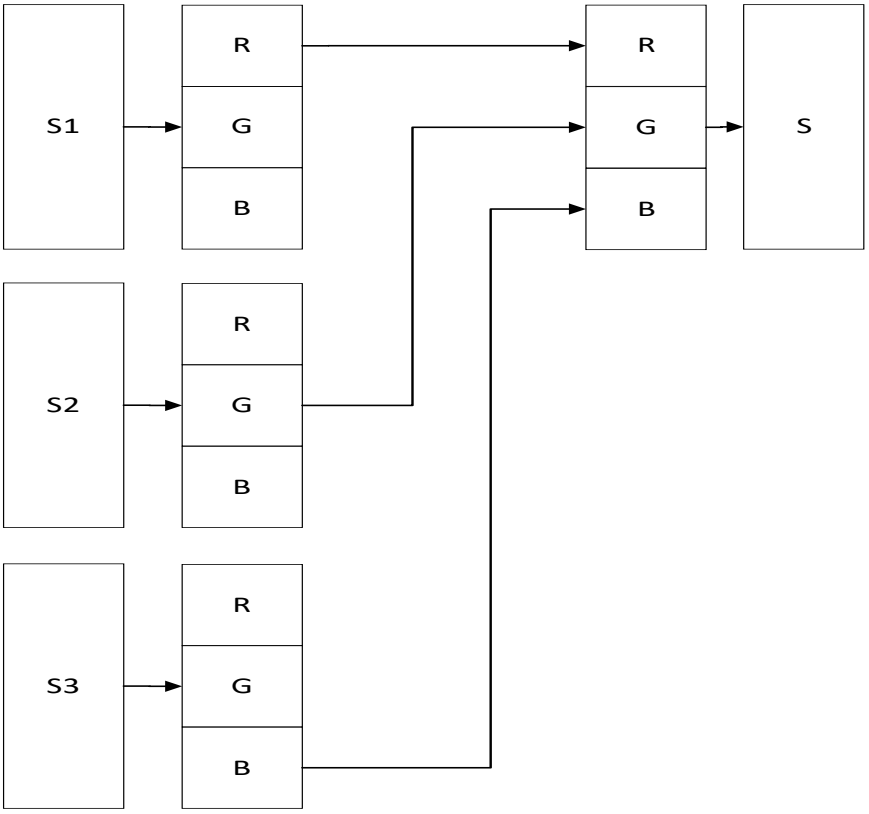


Рисунок 2.1 – Схема формування масиву вихідного зображення

У даному підрозділі було описано узагальнений алгоритм відновлення секрету.

2.2 Генератор псевдовипадкових чисел

Для виконання операції перестановки пікселів у зображенні необхідно мати генератор, який здатний створювати псевдовипадкові числа без повторень у заданому діапазоні. В сучасних шифрах перестановки застосовуються тільки в межах окремих блоків, а не для всього повідомлення, що обмежує можливість підвищення стійкості криптопротоколів. Нинішні методи формування перестановок не надають можливості створення окремої псевдовипадкової перестановки. Таким чином, у дослідженні [41] був представлений метод, який дозволяє створювати псевдовипадковий набір будь-якої довжини, відповідної різноманітності [41].

Кількість перестановок n елементів складає:

$$P_n = n!$$

Збільшення кількості елементів у процесі перестановки призводить до збільшення кількості можливих варіантів перестановок. Якщо перестановка залежить від секретного ключа, це може підвищити стійкість шифрування, оскільки операції перестановки виконуються на рівні всього повідомлення, а не в межах окремого блоку [41].

Порядкові номери байтів у файлі - це числа від 0 до n-1. Задача перестановки цих чисел полягає в тому щоб змінити їх порядок розташування. Цю задачу згідно з методом перестановки подається у вигляді формули [41]:

$$P = \{N, G, I, F, p\},$$

де N це набір послідовностей, на які розділяється послідовність чисел;

G - це множина функцій формування випадкових послідовностей[41]:

$$G = \{G_1, \dots, G_4\}$$

G_1 – функція формування підпослідовностей.

G_2 – функція формування кількості чисел q_i у підпослідовностях.

G_3 – функція формування номеру підпослідовності, для випробування.

G_4 – функція вибору числа з підпослідовності.

I – індикатор перестановки;

F – множина правил перестановки;

P – множина перестановок у підпослідовностях.

Послідовність з чисел розбивається на k підпослідовностей[41]:

$$N = \{Ni\}, \ i = 0,1,\dots,(k-1).$$

Кількість чисел у підпослідовностях може бути або однаковою, або відмінною. У випадку однакової кількості чисел для формування підпослідовностей використовується функція G1. Також існують два можливих варіанти створення цих підпослідовностей [41]:

- Перший варіант передбачає завдання кількості підпослідовностей k, судячи зі значення k, обчислюється кількість чисел qi у послідовностях. Якщо n ділиться на k без залишку, то буде k підпослідовність з кількістю чисел qi= n/k .
- Другий варіант передбачає отримання однакової кількості чисел у підпослідовностях q_i яка визначається з проміжку значень:

$$q_i = 2 \div]\frac{n}{2}[$$

Отримана множина підпослідовностей обчислюється за формулою:

$$k =]\frac{n}{q_i}[$$

Оскільки розглянуті варіанти, описані вище, не проявляють особливих різниць, можна визначити, що ці варіації є взаємозамінними [41].

У випадку різної кількості чисел у підпослідовностях, необхідно впровадити функцію G2, яка включає в себе наступні кроки. Кількість чисел у кожній окремій множині формується за допомогою генератора псевдовипадкових чисел (ПВЧ). При цьому використовується операція підрахунку кількості створених множин за певним алгоритмом. Спочатку визначається кількість послідовностей [41].

$$k := 0$$

Деякому параметру R надається значення кількості чисел *n* [41]:

$$R_0 := n$$

На кожному етапі формування множини значення параметра R зменшується на таку кількість[41]:

$$R_{k+1} := R_{k-q_i}$$

Процес виконання множин триває, поки виконується умова:

$$R > 0$$

Кожен крок закінчується збільшенням кількості множин на одиницю:

$$k := k + 1$$

Незалежно яким чином здійснюється розбиття, перше число кожної послідовності вираховується за формулою [41]:

$$n_{\Pi_i} = \sum_{j}^{i-1} q_j$$

а останнє – за формулою:

$$n_{O_i} = \sum_{j=0}^{i} q_j - 1$$

З використанням псевдовипадкового вибору множини необхідно використовувати додатковий параметр – індикатор перестановки. Індикатор перестановки представляє собою ціле число, яке розраховується для кожної окремої множини. Початкове значення індикатора дорівнює кількості чисел у множині [41]:

$$I_i := q_i$$

При кожному виборі числа з множини Ni, індикатор множини зменшується на одиницю. Коли значення індикатора стає рівним нулю, дана множина чисел більше не враховується. Зі значень окремих індикаторів формується загальне значення індикатора, яке розраховується відповідно до формули [41]:

$$I = I_0 + I_1 + \cdots + I_{k-1}$$

Використання індикатора перестановки застосовано тільки для детермінованого вибору чисел з множини [41].

Унікальною особливістю запропонованого методу є переміщення чисел в межах всієї опорної множини з гарантією їх псевдовипадкового розташування.

Всі елементи підпослідовності мають номера від 0 до L-1. Суть

перестановок полягає в тому, щоб витягувати елементи за номерами, які є псевдовипадковими числами від 0 до L-1, та впорядковувати їх в природному порядку [41]. Схема процесу перестановки наведено на рис 2.2

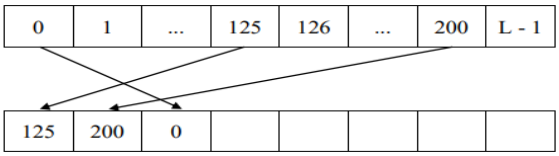


Рисунок 2.2 – Схема процесу перестановки

Для створення псевдовипадкових чисел необхідно розділити діапазон від 0 до L-1 на чотири під діапазони, кожен з яких визначений своїм мінімальним та максимальним значенням. Вибір піддіапазону відбувається природним чином шляхом збільшення або зменшення значень. Визначення піддіапазону здійснюється на основі псевдовипадкової послідовності з нулів та одиниць [41].

Для здійснення перестановок необхідно розподілити L на чотири рівні частини, кожна з яких має свій унікальний індекс. Цей індекс визначатиме, з якої конкретної частини файлу потрібно взяти елемент (рис. 2.3).

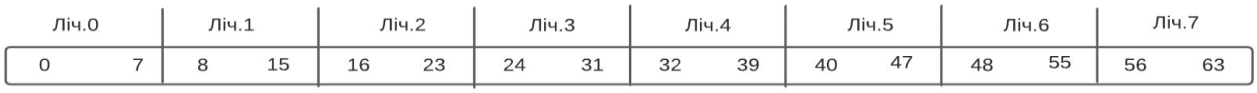


Рисунок 2.3 – Вигляд розподілених частин

Для визначення піддіапазону використовується регістр зсуву з лінійним зворотнім зв'язком, і для вибору чисел використовується лічильник [41].

Для встановлення початкового значення лічильника потрібно мати ключ, який складається з бітів. Якщо перший біт дорівнює нулю, то обирається мінімальне значення піддіапазону. У випадку, коли біт дорівнює одиниці, для лічильника вибирається максимальне значення в межах піддіапазону [41].

Розмір ключа становить чотири біти і визначається першими бітами початкового стану регістра. Ключ необхідний для визначення правил зміни лічильника. Якщо значення біта ключа рівне нулю, то лічильник збільшується, у випадку ж одиниці – зменшується (табл. 2.1).

Таблиця 2.1 – Початкові стани лічильників

Розряди кодів ключа								Початкові значення лічильників							
К0	К1	К2	К3	К4	К5	К6	К7	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч. 6	Ліч. 7
0	0	0	0	0	0	0	0	N_{min}^{0}	N_{min}^{1}	N_{min}^{0}	N_{min}^{1}	N_{min}^{4}	N_{min}^{5}	N_{min}^{6}	N_{min}^{7}
0	0	0	1	0	0	0	1	N_{min}^{0}	N_{min}^{1}	N_{min}^{2}	N_{max}^{3}	N_{min}^{4}	N_{min}^{5}	N_{min}^{6}	N_{max}^{7}
0	0	1	0	0	0	1	0	N_{min}^{0}	N_{min}^{1}	N_{max}^{2}	N_{min}^{3}	N_{min}^{4}	N_{min}^{5}	N_{max}^{6}	N_{min}^{7}
0	0	1	1	0	0	1	1	N_{min}^{0}	N_{min}^{1}	N_{max}^{2}	N_{max}^{3}	N_{min}^{4}	N_{min}^{5}	N_{max}^{6}	N_{max}^{7}
0	1	0	0	0	1	0	0	N_{min}^{0}	N_{max}^{1}	N_{min}^{2}	N_{min}^{3}	N_{min}^{4}	N_{max}^{5}	N_{min}^{6}	N_{min}^{7}
0	1	0	1	0	1	0	1	N_{min}^{0}	N_{max}^{1}	N_{min}^{2}	N_{max}^{3}	N_{min}^{4}	N_{max}^{5}	N_{min}^{6}	N_{max}^{7}
0	1	1	0	0	1	1	0	N_{min}^{0}	N_{max}^{1}	N_{max}^{2}	N_{min}^{3}	N_{min}^{4}	N_{max}^{5}	N_{max}^{6}	N_{min}^{7}
0	1	1	1	0	1	1	1	N_{min}^{0}	N_{max}^{1}	N_{max}^{2}	N_{max}^{3}	N_{min}^{4}	N_{max}^{5}	N_{max}^{6}	N_{max}^{7}
1	0	0	0	1	0	0	0	N_{max}^{0}	N_{min}^{1}	N_{min}^{2}	N_{min}^{3}	N_{max}^{4}	N_{min}^{5}	N_{min}^{6}	N_{min}^{7}
1	0	0	1	1	0	0	1	N_{max}^{0}	N_{min}^{1}	N_{min}^{2}	N_{max}^{3}	N_{max}^{4}	N_{min}^{5}	N_{min}^{6}	N_{max}^{7}
1	0	1	0	1	0	1	0	N_{max}^{0}	N_{min}^{1}	N_{max}^{2}	N_{min}^{3}	N_{max}^{4}	N_{min}^{5}	N_{max}^{6}	N_{min}^{7}
1	0	1	1	1	0	1	1	N_{max}^{0}	N_{min}^{1}	N_{max}^{2}	N_{max}^{3}	N_{max}^{4}	N_{min}^{5}	N_{max}^{6}	N_{max}^{7}
1	1	0	0	1	1	0	0	N_{max}^{0}	N_{max}^{1}	N_{min}^{2}	N_{min}^{3}	N_{max}^{4}	N_{max}^{5}	N_{min}^{6}	N_{min}^{7}
1	1	0	1	1	1	0	1	N_{max}^{0}	N_{max}^{1}	N_{min}^{2}	N_{max}^{3}	N_{max}^{4}	N_{max}^{5}	N_{min}^{6}	N_{max}^{7}
1	1	1	0	1	1	1	0	N_{max}^{0}	N_{max}^{1}	N_{max}^{2}	N_{min}^{3}	N_{max}^{4}	N_{max}^{5}	N_{max}^{6}	N_{min}^{7}
1	1	1	1	1	1	1	1	N_{max}^{0}	N_{max}^{1}	N_{max}^{2}	N_{max}^{3}	N_{max}^{4}	N_{max}^{5}	N_{max}^{6}	N_{max}^{7}

Лічильники можуть або збільшувати або зменшувати свій стан. Секретний ключ визначає правила лічильника. Відповідно для кожного лічильника буде встановлено початкове значення (табл. 2.2)

Таблиця 2.2 – Правила функціонування лічильників

Розряди кодів ключа								Початкові значення лічильника							
К0	К1	К2	К3	К4	К5	К6	К7	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч. 6	Ліч. 7
0	0	0	0	0	0	0	0	+1	+1	+1	+1	+1	+1	+1	+1
0	0	0	1	0	0	0	1	+1	+1	+1	-1	+1	+1	+1	-1
0	0	1	0	0	0	1	0	+1	+1	-1	+1	+1	+1	-1	+1
0	0	1	1	0	0	1	1	+1	+1	-1	-1	+1	+1	-1	-1
0	1	0	0	0	1	0	0	+1	-1	+1	+1	+1	-1	+1	+1
0	1	0	1	0	1	0	1	+1	-1	+1	-1	+1	-1	+1	-1
0	1	1	0	0	1	1	0	+1	-1	-1	+1	+1	-1	-1	+1
0	1	1	1	0	1	1	1	+1	-1	-1	-1	+1	-1	-1	-1
1	0	0	0	1	0	0	0	-1	+1	+1	+1	-1	+1	+1	+1
1	0	0	1	1	0	0	1	-1	+1	+1	-1	-1	+1	+1	-1
1	0	1	0	1	0	1	0	-1	+1	-1	+1	-1	+1	-1	+1
1	0	1	1	1	0	1	1	-1	+1	-1	-1	-1	+1	-1	-1
1	1	0	0	1	1	0	0	-1	-1	+1	+1	-1	-1	+1	+1
1	1	0	1	1	1	0	1	+1	+1	+1	+1	+1	+1	+1	+1
1	1	1	0	1	1	1	0	+1	+1	+1	-1	+1	+1	+1	-1
1	1	1	1	1	1	1	1	+1	+1	-1	+1	+1	+1	-1	+1

З кожного регістру зсуву з лінійним зв’язком береться по два біти для

визначення з якої частини файлу брати елементи.

Приклад генерування псевдовипадкової послідовності чисел розглянемо для інтервалу від 0 до 63. Інтервал розбиваєм на 8 частини від 0 до 7, від 8 до 15, від 16 до 23, від 24 до 31, від 32 до 39, від 40 до 47, від 48 до 55, від 56 до 63 . Нехай ключ К {1,0,0,1,1,0,0,1}, згенерований код: 11 На основі вхідних параметрів формується початкове значення лічильників: Ліч.0 =7, Ліч.1 = 8, Ліч.2 =16, Ліч.3 = 31, Ліч.4 = 39, Ліч.5= 40, Ліч.6= 48 , Ліч.8=63.

Таблиця 2.3 – Приклад формування послідовності псевдовипадкових чисел

Код	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч.6	Ліч.7	Результат
001	7	8 8+1	16	31	39	40	48	63	8
011	7	9	16	31 31-1	39	40	48	63	31
000	7 7-1	9	16	30	39	40	48	63	7
101	6	9	16	30	39	40	48 48+1	63	48
111	6	9	16	30	39	40	49	63 63-1	63
000	6 6-1	9	16	30	39	40	49	62	6
011	5	9	16	30 30-1	39	40	49	62	30
110	5	9	16	29	39	40 40+1	49	62	40
010	5	9	16 16+1	29	39	41	49	62	16
011	5	9	17	29 29-1	39	41	49	62	29
100	5	9	17	28	39 39-1	41	49	62	39
100	5	9	17	28	38 38-1	41	49	62	38
110	5	9	17	28	37	41 41+1	49	62	41
110	5	9	17	28	37	42 42+1	49	62	42
000	5 5-1	9	17	28	37	43	49	62	5
010	4	9	17 17+1	28	37	43	49	62	17
011	4	9	18	28 28-1	37	43	49	62	28

Продовження таблиці 2.3

Код	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч.6	Ліч.7	Результат
101	4	9	18	27	37	43	49 49+1	62	49
101	4	9	18	27	37	43	50 50+1	62	50
111	4	9	18	27	37	43	51	62 62-1	62
100	4	9	18	27	37 37-1	43	51	61	37
000	4 4-1	9	18	27	36	43	51	61	4
101	3	9	18	27	36	43	51 51+1	61	51
100	3	9	18	27	36 36-1	43	52	61	36
010	3	9	18 18+1	27	35	43	52	61	18
111	3	9	19	27	35	43	52	61 61-1	61
100	3	9	19	27	35 35-1	43	52	60	35
001	3	9 9+1	19	27	34	43	52	60	9
111	3	10	19	27	34	43	52	60 60-1	60
001	3	10 10+1	19	27	34	43	52	59	10
001	3	11 11+1	19	27	34	43	52	59	11
110	3	12	19	27	34	43 43+1	52	59	43
010	3	12	19 19+1	27	34	44	52	59	19
011	3	12	20	27 27-1	34	44	52	59	27
011	3	12	20	26 26-1	34	44	52	59	26
000	3 3-1	12	20	25	34	44	52	59	3
001	2	12 12+1	20	25	34	44	52	59	12
001	2	13 13+1	20	25	34	44	52	59	13
110	2	14	20	25	34	44 44+1	52	59	44
110	2	14	20	25	34	45 45+1	52	59	45

Продовження таблиці 2.3

Код	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч.6	Ліч.7	Результат
010	3	9	18 18+1	27	35	43	52	61	18
111	3	9	19	27	35	43	52	61 61-1	61
100	3	9	19	27	35 35-1	43	52	60	35
001	3	9 9+1	19	27	34	43	52	60	9
111	3	10	19	27	34	43	52	60 60-1	60
001	3	10 10+1	19	27	34	43	52	59	10
001	3	11 11+1	19	27	34	43	52	59	11
110	3	12	19	27	34	43 43+1	52	59	43
010	3	12	19 19+1	27	34	44	52	59	19
011	3	12	20	27 27-1	34	44	52	59	27
011	3	12	20	26 26-1	34	44	52	59	26
000	3 3-1	12	20	25	34	44	52	59	3
001	2	12 12+1	20	25	34	44	52	59	12
001	2	13 13+1	20	25	34	44	52	59	13
110	2	14	20	25	34	44 44+1	52	59	44
110	2	14	20	25	34	45 45+1	52	59	45
111	2	14	20	25	34	46	52	59 59-1	59
110	2	14	20	25	34	46 46+1	52	58	46
000	2 2-1	14	20	25	34	47	52	58	2
010	1	14	20 20+1	25	34	47	52	58	20
110	1	14	21	25	34	47 stop	52	58	47
001	1	14 14+1	21	25	34		52	58	14
011	1	15	21	25 25-1	34		52	58	25
010	1	15	21 21+1	24	34		52	58	21

Продовження таблиці 2.3

Код	Ліч.0	Ліч.1	Ліч.2	Ліч.3	Ліч.4	Ліч.5	Ліч.6	Ліч.7	Результат
000	1 1-1	15		24	34		52	58	1
111	0 stop	15		24	34		52	58 58-1	58
100	0	15		24	34 34-1		52	57	34
100	0	15		24	33 33-1		52	57	33
111	0	15		24			52	57 57-1	57
001	0	15 stop		24			52	56	15
010	0		22 22+1	24			52	56	22
101	0			24			52 52+1	56	52
100	0			24	32 stop		53	56	32
000	0			24			53	56	0
101				24			53 53+1	56	53
111				24				56 stop	56
011				24 stop					24
010			23 stop						23
111							54 54+1		54
000							55 stop		55

Так як генератор перестановок приймає на вхід послідовність, що складається з нулів і одиниць, необхідний генератор для створення таких послідовностей [41].

Більшість використовуваних до даних пір генераторів базуються на регістрах зсуву з лінійним зворотним зв'язком (РЗЛЗЗ). Це обумовлено трема основними факторами:

- їхні структури легко втілюються в апаратному та програмному забезпеченні.
- послідовності, що створює РЗЛЗЗ, мають високі статистичні

характеристики;

- поведінка цих генераторів легко піддається аналізу за допомогою алгебри;

Регістр зсуву із зворотним зв'язком має дві основних складових: функції зворотного зв'язку та регістра зсуву [41].

Регістр зсуву представляє собою послідовність бітових комірок, і його розмір виражається в бітах: якщо регістр складається з r комірок, то його називають r-бітовим регістром зсуву. Кожен раз, коли генерується новий біт, всі біти регістра зсуваються на одну позицію вправо. Новий лівий біт обчислюється як функція від інших бітів регістра; конкретний вигляд функції залежить від використовуваного зворотного зв'язку. Вихід регістра зсуву в кожному такті представляє собою один біт, а саме - наймолодший біт регістра. Період регістра зсуву визначається як довжина вихідної послідовності до того моменту, коли вона починає повторюватися [41].

Для створення двійкового регістра зсуву використовується вихідна інформація у вигляді твірного полінома (багаточлена). Ступінь цього полінома визначає розрядність регістра зсуву, а ненульові коефіцієнти вказують на характер зворотних зв'язків. Наприклад, цей поліном служить основою для конфігурації регістра зсуву. Ф(x)=x8+x7 +x5 +x3+1, відповідає РЗЛЗЗ, схему якого наведено на рис 2.4 [41].

Найпростішим типом регістрів зсуву є регістр зсуву з лінійним зворотним зв'язком або РЗЛЗЗ. Двійкові псевдовипадкові періодичні послідовності, що генеруються з використанням регістрів зсуву з лінійним зворотним зв'язком, називаються РЗЛЗЗ - послідовностями або лінійними рекурентними послідовностями [41].

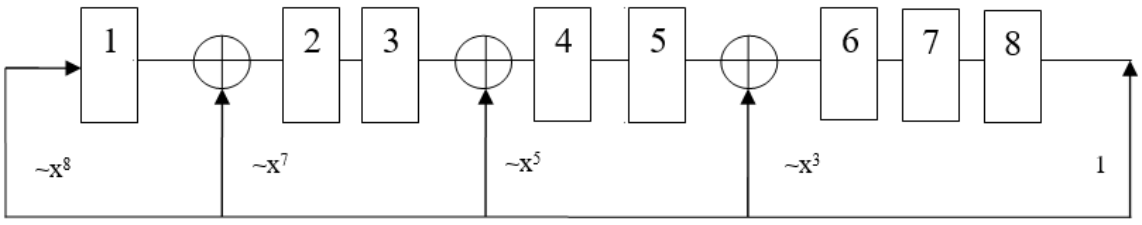


Рисунок 2.4 – Генератор Галуа, що відповідає поліному

$$\Phi(x) = x^8+x^7+x^5+x^3+1$$

Метод, який пропонується, відзначається тим, що числа переставляються в межах всієї опорної множини так, щоб забезпечити псевдовипадковий порядок. Застосування такого підходу призводить до підвищення криптографічної стійкості шифрування, оскільки збільшується кількість можливих перестановок [12].

2.3 Алгоритм відновлення секрету

Алгоритм відновлення секрету — це спосіб розподілу таємної інформації між кількома учасниками таким чином, що тільки певна комбінація їх частин дозволяє відновити початковий секрет. Цей метод може бути використаний для забезпечення безпеки та надійності даних, наприклад, для резервного копіювання критичної інформації або для розподілу ключів шифрування.

Схема відновлення зображення показана на (рис.2.5)

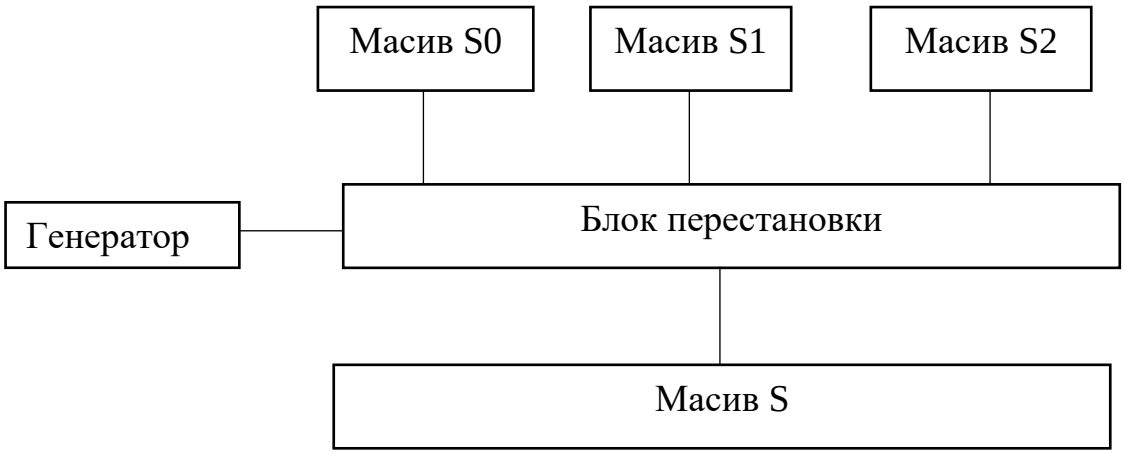


Рисунок 2.5 – Схема відновлення зображення

Алгоритм відновлення зображення працює таким чином: спочатку відбувається ініціалізація генератора, наступним кроком відбувається формування масиву довжиною Ls. Наступним кроком відбувається перевірка чи масив зчитано , далі відбувається формування псевдовипадкового числа R, після цього відбувається зчитування байту з масиву Si та запис байту у масив S за адресою R. Далі відбувається перевірка чи повністю зчитався масив довжиною Ls, якщо ж Ls менше 0 то алгоритм завершується, а якщо масив більше 0 то

відбувається повернення на етап формування псевдовипадковго числа. На (рис.2.6) зображень алгоритм відновлення зображення.

Рисунок 2.4 – Алгоритм відновлення зображень

Для реалізації цього алгоритму потрібно розробити алгоритм ініціалізація

генератора.

Алгоритм ініціалізація генератора псевдовипадкових чисел працює наступним чином: спочатку відбувається формування масиву довжиною Ls, далі відбувається визначення молодшого біта стану S0. Далі відбувається процес перестановки, наступним кроком відбувається зсув на одиницю, наступним кроком відбувається перевірка i<0 якшо так алгоритм закінчує свою роботу, якщо ж ні алгоритм повертається на етап визначення молодшого біту (рис.2.7).

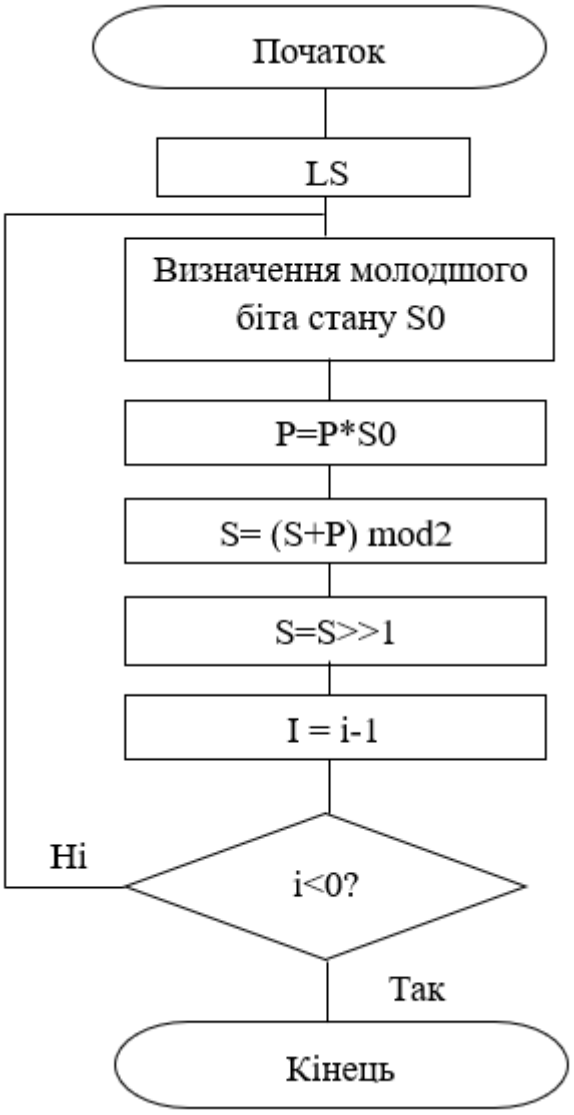


Рисунок 2.7 – Алгоритм ініціалізації генератора

Отже в даному розділі було розроблено алгоритм відновлення зображення та алгоритм ініціалізації генератора псевдовипадкових чисел. Було наведено

приклад функціонування генератора псевдовипадкових чисел на основі регістру зсуву зі зворотнім зв’язком.

2.4 Оцінка складності реалізації запропонованого методу.

Кількість виконуваних операцій T визначається:

- розміром файлу зображення в байтах L;
- кількістю операцій зчитування байтів файлу L;
- кількістю операцій для генерування псевдовипадкового числа G;
- кількістю операцій запису в частини розподіленого секрету L.

$$T=T_{зч}+T_G+T_{зап}$$

Нехай операції зчитування і запису оцінюються однією умовною одиницею.

Генерування псевдовипадкового числа передбачає зміну стану регістру зсуву з лінійним зворотнім зв’язком (4 операції) і зміну стану лічильників (1 операція).

Таким чином генерування одного псевдовипадкового числа вимагає виконання п’яти операцій. З урахуванням цього маємо таку складність розподілення секрету (зашифрування):

$$T=L+L+5L=7L(у.о.)$$

Порівняємо складність зашифрування зображення з використанням блокового шифру AES і на основі запропонованого методу. Оскільки зчитування і запис байтів файлів зображень буде виконуватись в обох випадках, то ці операції для порівняння не будемо враховувати.

Відомі оцінки складності шифрування за допомогою шифру AES показують, що для шифрування одного біта потрібно виконати 2,7 операцій або 21,6 операцій на байт.

Запропонований метод зашифрування передбачає виконання 5 операцій на байт, а отже в 4,1 рази менше операцій.

Таким чином, мету дослідження досягнуто, оскільки забезпечено пришвидшення процесу зашифрування в 4,1 рази.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

3.1 Вибір засобів реалізації програмного застосунку

Магістерська кваліфікаційна робота має на меті створення програмного засобу. Тому вибір мови для реалізації відіграє важливу роль. Також є важливим середовище розробки, в якому буде розроблятись програмний застосунок.

Дана програма буде реалізована у середовищі pyCharm.

Мова програмування Python:

1. Простота та Читабельність коду: Python володіє простим синтаксисом, що дозволяє швидко вивчити основи мови. Це сприяє зручності написання та розумінню коду.

2. Широкий спектр бібліотек: Python має величезну кількість бібліотек для різних завдань, включаючи роботу з мережами, аналіз даних та веб-розробку. Це полегшує реалізацію різноманітних функцій у програмному забезпеченні.

3. Активна спільнота розробників: Python має велику та активну спільноту розробників, що дозволяє отримувати підтримку, розвивати проекти та використовувати готові рішення.

4. Підтримка об'єктно-орієнтованого програмування (ООП): Python повністю підтримує об'єктно-орієнтоване програмування, що дозволяє створювати структуровані та модульні програми.

Середовище розробки PyCharm:

1. Зручний інтерфейс: PyCharm надає зручний та інтуїтивно зрозумілий інтерфейс, що полегшує написання, відлагодження та тестування коду.

2. Підтримка віртуальних середовищ: PyCharm інтегрується з віртуальними середовищами, що дозволяє ізолювати залежності проекту та забезпечує його стабільність.

3. Автоматичне визначення помилок: PyCharm виявляє та відображає помилки під час написання коду, що полегшує їх виправлення та забезпечує високу якість програмного забезпечення.

4. Підтримка відладки: PyCharm має вбудовані інструменти для ефективного відлагодження коду, що сприяє швидкому виявленню та усуненню помилок.

Отже, обрані мова програмування Python та середовище розробки PyCharm є оптимальними для досягнення мети магістерської кваліфікаційної роботи, забезпечуючи ефективний розвиток програмного засобу та його якісну реалізацію.

3.2 Вимоги до програмного засобу

В магістерській кваліфікаційній роботі потрібно розробити програму, яка виконує розподіл секрету на основі зображень. Програма повинна бути розроблена з використанням дизайну та має мати вигляд Windows додатку. Програма повинна мати елементи керування, працювати з зображеннями і виконувати поставлені задачі.

Доцільно передбачити використання таких ресурсів як:

– кнопки як основний вид керування програмою;
– діалогове вікно як основне вікно для роботи з програмою;
– вікна повідомлень;

Інформація про помилки та неправильне використання програми буде виводитись в вікні повідомлення.

Під час розробки програми потрібно виконувати і дотримуватись таких дій:

– дослідження структур зображень;
– підібрати програмні засоби для реалізації програмного забезпечення;
– побудувати алгоритми виконання, схеми даних;
– програмна реалізація методу відновлення зображень
– розробка інтерфейсу
–тестування програмного засобу
–виявлення помилок та їх усунення.

3.3 Розробка загальної схеми функціонування програми

Під час запуску програми відкривається вікно, у якому є можливість відкрити папку з зашифрованими частинами, а також буде показано кінцеве зображення після об’єднання зашифрованих частин. На рис 3.1 представлена загальна схема роботи програми

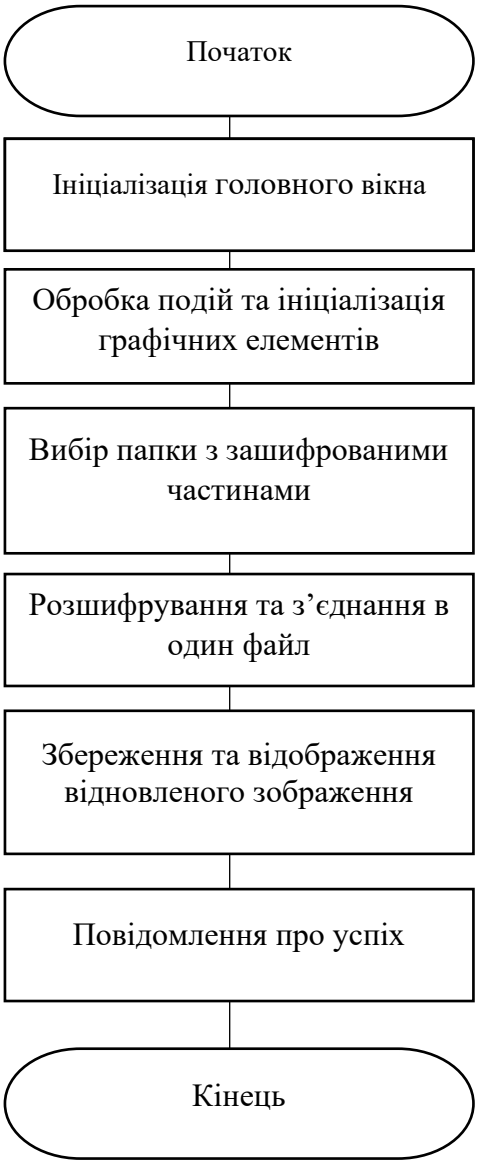


Рисунок 3.1 Загальна схема роботи програми

Також у програмі є додаткові елементи керування такі як кнопки відкриття папки з зашифрованими частинами та кнопка розшифрування.

3.4 Вибір формату зображень

HEIC (High-Efficiency Image Format) - це формат зображення, розроблений консорціумом MPEG та ITU-T для ефективного ущільнення та збереження зображень. Основними перевагами .HEIC є висока якість зображення при ефективному використанні простору зберігання.

Можливості формату HEIC:

– Ущільнення: формат .HEIC використовує сучасні алгоритми стиснення, такі як HEVC (High Efficiency Video Coding), для збереження зображень з високою якістю при низькому розмірі файлу.

– Підтримка альфа-каналу: .HEIC підтримує прозорість та альфа-канал, що дозволяє зберігати зображення з прозорістю для кращої інтеграції з різними фонами.

– Множинні зображення та анімації: формат дозволяє зберігати не тільки одиночні зображення, але й групи зображень та навіть анімації, усе в одному файлі .HEIC.

– Кольоровий простір і HDR:. HEIC підтримує розширені кольорові простори та високий динамічний діапазон (HDR), дозволяючи зображенням виглядати більш реалістично та живо.

– Ефективне використання простору зберігання: Завдяки потужному стисненню та продуманому використанню бітів, .HEIC дозволяє ефективно використовувати простір зберігання на пристроях.

Розширення файлів у форматі .HEIC вказує на використання даного формату. Наприклад, "image1.heic" вказує на файл .HEIC, який містить зображення.

Сумісність та Використання: формат .HEIC є популярним серед пристроїв, що працюють на iOS, а також у сучасних Android-пристроях. Його використання забезпечує високу якість та ефективне використання ресурсів пристроїв для збереження та обміну зображеннями.

3.5 Програмна реалізація відновлення зображень

Насамперед потрібно інціалізувати генератор псевдовипадкових чисел на основі регістру зсуву зі зворотнім зв’язком

```
Class SimpleCounter:
def init(self, seed=0, key="None"):
    self.state = seed
    self.feedback_mask = int(key, 2)
def next(self):
    feedback_bit = self.state & 1
    self.state >>= 1
    if feedback_bit:
        self.state ^= self.feedback_mask
    return self.state
```

Відновлення зображення починається з функції розшифрування підпослідовностей:

```
 def decrypt_subsequence(self, subsequence,
permutation_sequence):
    decrypted_subsequence = []
    for i, pixel in enumerate(subsequence):
        decrypted_pixel = pixel ^ self.next()
        decrypted_subsequence.append(decrypted_pixel)
    decrypted_subsequence = [decrypted_subsequence[i] for i in
permutation_sequence]
    return decrypted_subsequence;
```

Також для відновлення зображення потрібна функція розшифрування частин зображення:

```
 def decrypt_image_parts(self, encrypted_image_parts, key,
num_counters):
    decrypted_image_parts = []
    for i in range(num_counters):
        pixels_per_counter = encrypted_image_parts[i].shape[1]
        direction_forward = int(key[i*8:(i+1)*8], 2) % 2 == 0
        decryption_counter =
```

```
SimpleCounter(seed=int(key[i*8:(i+1)*8], 2), key=key)

        print(f"Decryption Seed for Part {i + 1}:
{decryption_counter.state}")

        forward = direction_forward

        permutation_sequence =
generate_permutation_sequence(pixels_per_counter,
decryption_counter, forward)

        decrypted_part = np.zeros_like(encrypted_image_parts[i],
dtype=np.uint8)

        for j in range(encrypted_image_parts[i].shape[0]):

            subsequence = encrypted_image_parts[i][j, :, :]

            decrypted_subsequence =
decryption_counter.decrypt_subsequence(subsequence,
permutation_sequence)

            decrypted_part[j, :, :] =
np.array(decrypted_subsequence)

        decrypted_image_parts.append(decrypted_part)

    return decrypted_image_parts
```

Для успішної роботи програми також потрібно використовувати функцію для розшифрування зображень:

```
def decrypt_image(self):

        if self.image_path is None:

            messagebox.showerror("Error", "Please select an image
first.")

            return

        key = self.get_user_key()

        if key is None:

            return

        num_counters = self.get_num_counters()

        if num_counters is None:

            return

        file_paths = filedialog.askopenfilenames(

            title="Обрати зашифровані частини",

            filetypes=[("Файли зображень", "*.png")])

        if len(file_paths) != num_counters:

            messagebox.showerror("Error", f"Please select exactly
{num_counters} encrypted parts.")
```

```
            return
```

Функція для збереження розшифрованого зображення

```
def save_decrypted_image(self, decrypted_image_parts):
        combined_decrypted_image = np.concatenate(decrypted_image_parts,
axis=1)
        output_folder = "C:/Decrypt/"
        os.makedirs(output_folder, exist_ok=True)
        output_path = os.path.join(output_folder,
"combined_decrypted_image.png")
```

3.6 Реалізація інтерфейсу

Інтерфейс відіграє важливу роль для зручності використання програми тому було реалізовано інтерфейс в якому є такі елементи:

- головне вікно;
- кнопки для відкриття папки та розшифрування;
- інформаційні вікна

Вигляд головно вікна програми представлено на (рис 3.2)

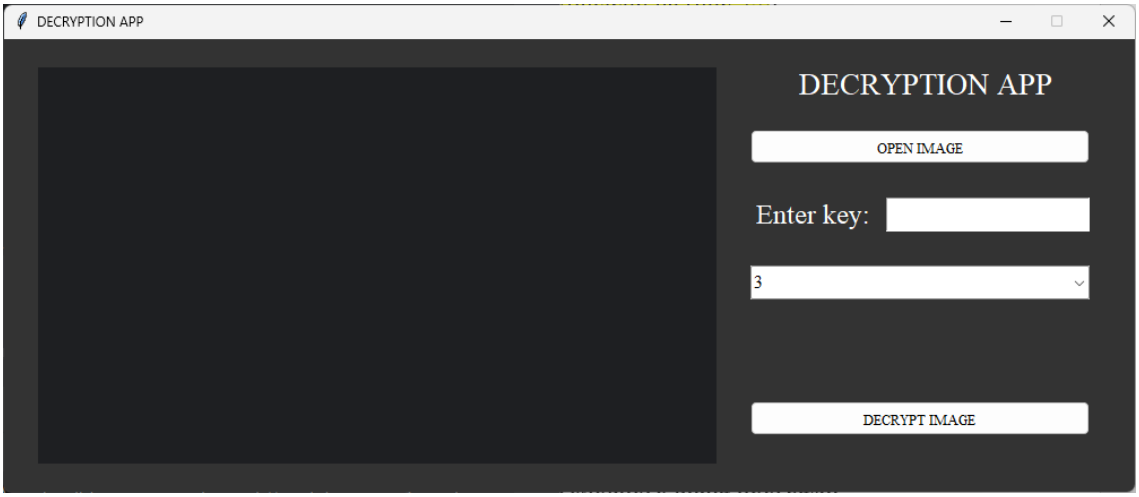


Рисунок 3.2 Вигляд головного вікна програми

Усі елементи програми були розроблені в середовищі розробки PyCharm з використанням бібліотеки Tk.

Пошук файлів в комп’ютері виконується викликом стандартної панелі відкриття файлів(рис. 3.3)

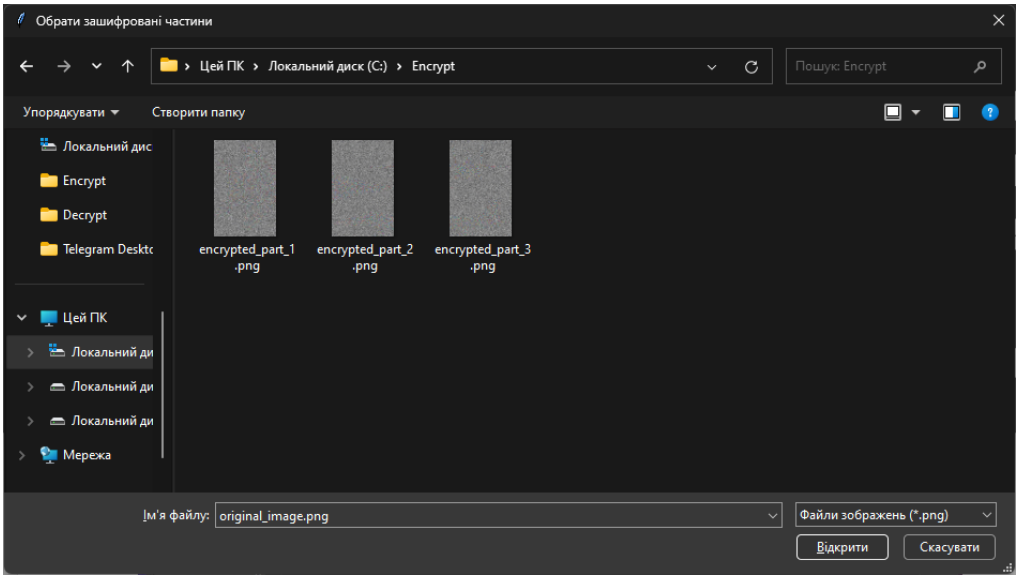


Рисунок 3.3 – Вигляд панелі відкриття файлів

Отже створено зручний та інтуїтивно зрозумілий інтерфейс для використання та реалізовує процес відновлення зображення.

3.7 Результати роботи програми

Для перевірки працездатності програми потрібно запустити програму і в головному вікні натиснути на кнопку «DECRYPT IMAGE ». У відкритій панелі файлів вибрати потрібну необхідну папку де зберігаються зашифровані частини(рис 3.4).

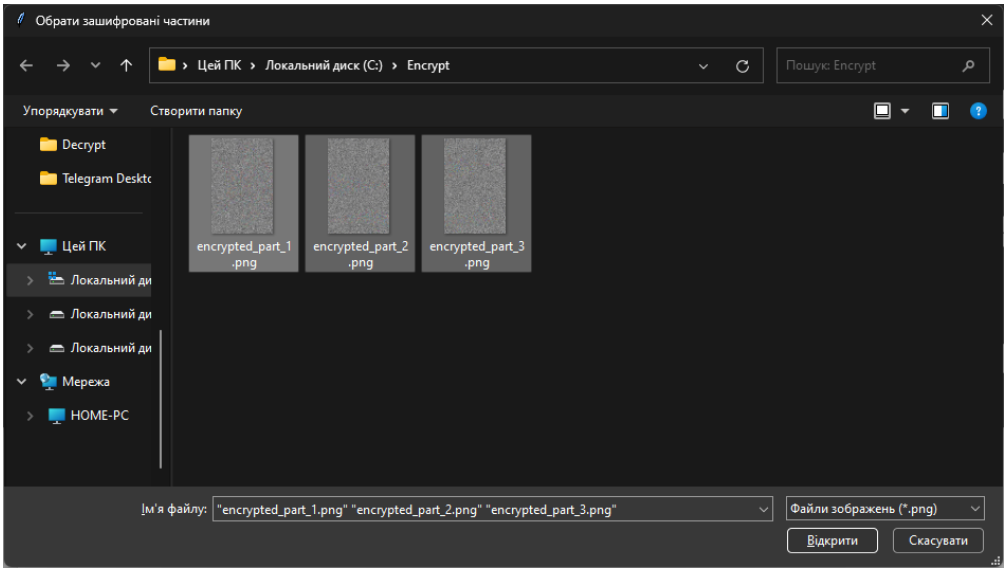


Рисунок 3.4 – Вигляд вікна вибору папки

Після того як вибрали потрібні частини потрібно натиснути на кнопку «Відкрити». Після того виводить повідомлення про успіх (рис3.5)

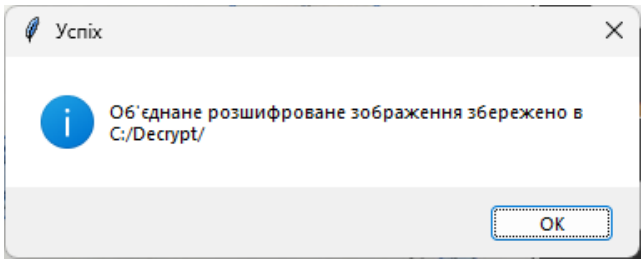


Рисунок 3.5 – Повідомлення про успіх

Після натискання кнопки «ОК» в вікно програми виводить розшифроване зображення(рис 3.6).

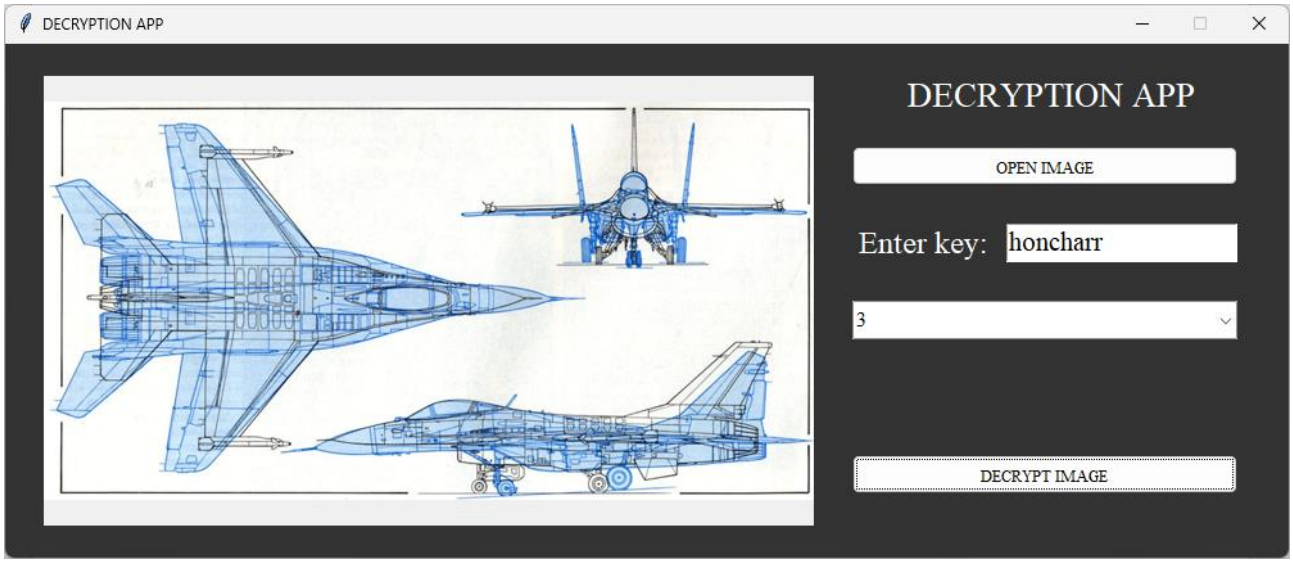


Рисунок 3.6 – Вигляд головного вікна з розшифрованим зображенням

В цьому підрозділі було протестовано програмний застосунок для розшифрування зображень. Далі буде проведено дослідження для того щоб продемонструвати процес розшифрування зображення коли буде накладено завади на зашифровані частини зображення.

Першим етапом дослідження нам потрібно отримати зашифровані частини зображення з першої частини роботи та потрібно накласти на них завади(рис 3.7)

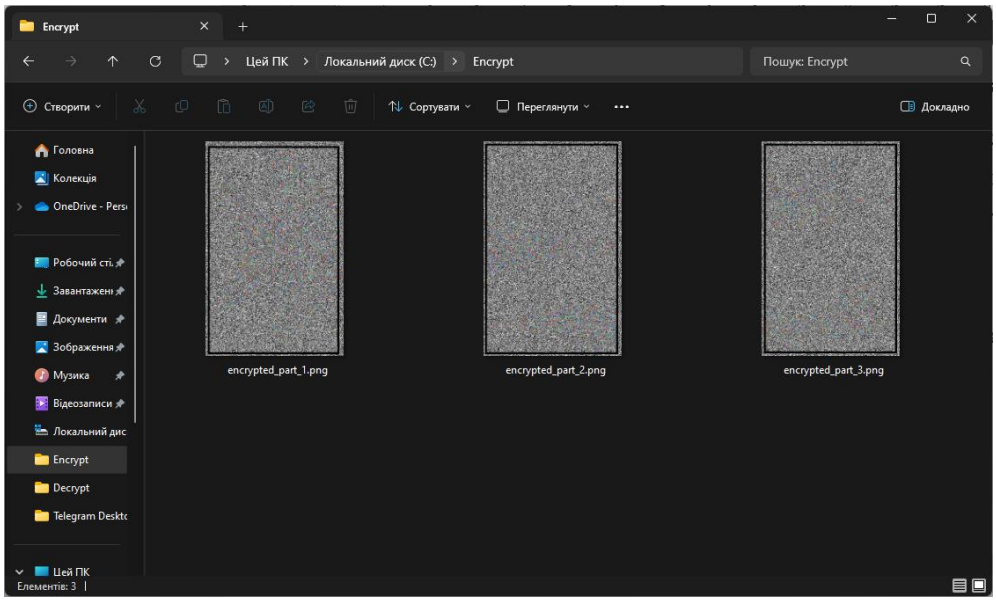


Рисунок 3.7 – Вигляд зображень з накладеними завадами

Після того як завади на кладені та відбувся процес розшифрування можна відстежити те що якість зображення погіршилась та на ньому видно накладені завади(рис 3.8)

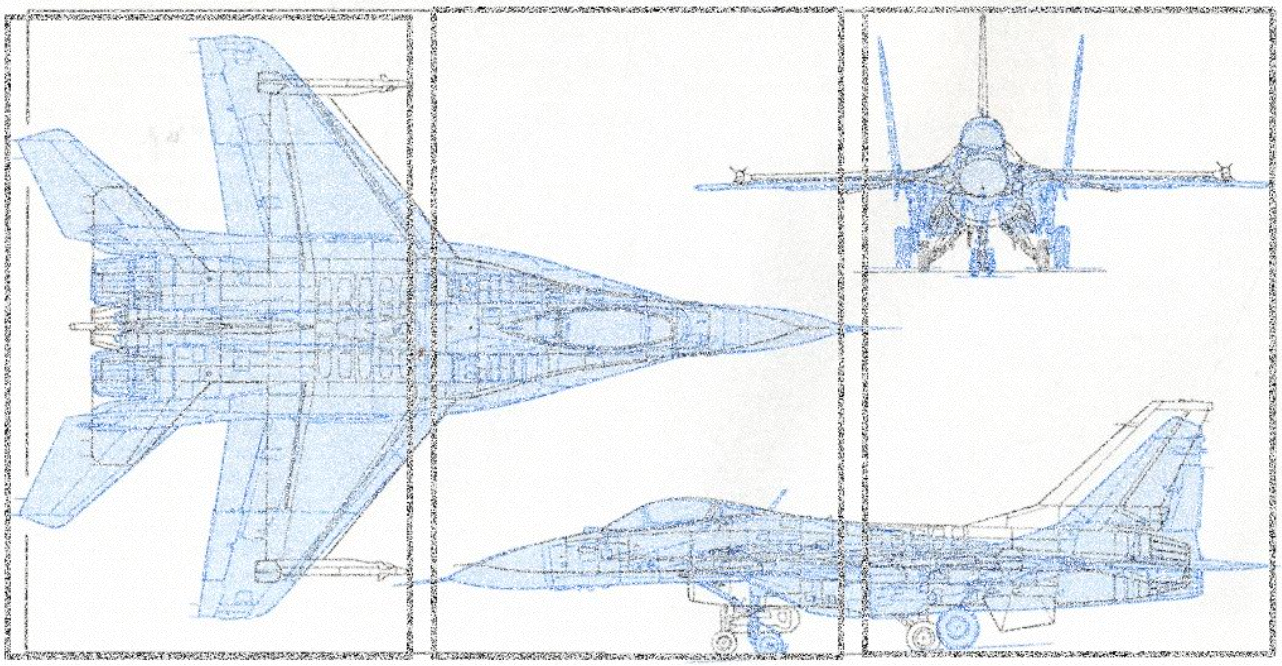


Рисунок 3.8 – Вигляд зображення після накладання завад

Отже в даному розділі було розроблено програмний застосунок для відновлення зображень, та проведено експеримент стосовно впливу завад на відновлення зображення.

4 ЕКОНОМІЧНА ЧАСТИНА

Для успішного впровадження науково-технічної розробки критично важливо, щоб вона відповідала сучасним вимогам науково-технічного прогресу та враховувала економічні аспекти. Надання оцінки економічної ефективності результатів науково-дослідної роботи є важливою частиною цього процесу. Дослідження, яке представлене у магістерській роботі і присвячене розробці та вивченню "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень", віднесено до науково-технічних робіт, спрямованих на виведення на ринок. Рішення про комерціалізацію розробки може бути прийняте протягом проведення самої роботи, відкриваючи можливості для подальшого виведення на ринок. Цей напрямок визначається як пріоритетний, оскільки розроблені результати можуть бути корисними для різних зацікавлених сторін і приносити економічні вигоди. Однак для успішної реалізації цього процесу ключовим є залучення зацікавленого інвестора, який виявить інтерес до втілення даного проекту, і переконання його у доцільності інвестування у цю розробку. З метою досягнення цього завдання були визначені такі етапи виконання робіт:

1. Проведення комерційного аудиту науково-технічної розробки, включаючи визначення науково-технічного рівня та комерційного потенціалу.
2. Розрахунок витрат на реалізацію науково-технічної розробки.
3. Проведення розрахунку економічної ефективності впровадження та комерціалізації науково-технічної розробки для потенційного інвестора, а також обґрунтування економічної доцільності комерціалізації з точки зору інвестора.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень". Оцінювання науково-технічного рівня розробки та її комерційного

потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [42].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція не підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці. Для оцінки науково-технічного рівня і комерційного потенціалу розробки експертами було запрошено трьох незалежних експертів Вінницького національного технічного університету кафедри «Захисту інформації»: Лужецький Володимир Андрійович, д. т. н., професор, Кондратенко Наталія Романівна, к. т. н., професор, Каплун Валентина Аполлінаріївна ст. викладач

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного

Критерії	Експерт (ПІБ, посада)		
	Лужецький В.А.	Каплун В.А.	Кондратенко Н.Р.
	Бали, виставлені експертами:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги (ціна продукту)	4	5	5
4. Ринкові переваги (технічні властивості)	3	4	5
5. Ринкові переваги (експлуатаційні витрати)	4	5	4
6. Ринкові перспективи (розмір ринку)	4	4	5
7. Ринкові перспективи (конкуренція)	2	3	3
8. Практична здійсненність (наявність фахівців)	4	4	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	2	2	3
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	$СБ_1$=39	$СБ_2$=44	$СБ_3$=45
Середньоарифметична сума балів $СБ_c$	43		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [42].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень" становить 43 бали, що, відповідно до таблиці 4.3 рівень комерційного потенціалу розробки високий, що свідчить про комерційну важливість проведення даних досліджень.

Магістерська кваліфікаційна робота "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень" відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто при цьому відбувається комерціалізація науково-технічної розробки. Цей напрямок є для нас пріоритетним, оскільки результатами розробки можуть користуватися не тільки самі розробники, а й інші споживачі, отримуючи при цьому суттєвий економічний ефект.

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

В якості аналога для розробки було обрано EOS-1D X Mark II. Основними недоліками аналога є слабкість шифрованих сегментів. Також до недоліків можна віднести незалежність сегментів одне від одного. У розробці дана проблема вирішується розбиттям файлу на частини який міститься в кожному

сегменті. Також система випереджає аналог за такими параметрами як швидкодія, надійність та простота використання.

Одиничний параметричний індекс розраховуємо за формулою [42]:

$$q_i = \frac{P_i}{P_{базі}} \quad (4.1)$$

де qi – одиничний параметричний індекс, розрахований за і-м параметром;

Pi – значення і-го параметра виробу;

Рбазі – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Стійкість шифрування	60	100	1,7	30%
Функціонал	5	12	2,4	30%
Похибка, %	2	1	2	10%
Розмір програми, МБ	400	130	3,1	30%

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [42]:

$$I_{НП} = \prod_{i=1}^{n} q_i \quad ,(4.2)$$

де Інп – загальний показник конкурентоспроможності за нормативними параметрами;

qi – одиничний (частинний) показник за і-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому Інп = 1.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [42]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \quad (4.3)$$

де Ітп – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

qi – одиничний параметричний показник і-го параметра;

αi – вагомість і-го параметричного показника $\sum_{i=1}^{n} \alpha_i = 1$, ;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

$$Imn = 1{,}7 \cdot 0{,}3 + 2{,}4 \cdot 0{,}3 + 2 \cdot 0{,}1 + 3{,}1 \cdot 0{,}3 = 2{,}36.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [42]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \quad ,(4.4)$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

qi – економічний параметр і-го виду;

βi – частка і-го економічного параметра, ;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП} = 0{,}75 \cdot 0{,}5 + 0{,}86 \cdot 0{,}5 = 0{,}80.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [42]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \quad (4.5)$$

$$K_{ІНТ} = 1 \cdot 2{,}36 / 0{,}80 = 2{,}95.$$

Інтегральний показник конкурентоспроможності КІНТ > 1, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень", під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень" передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

– збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 1 особа

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 1000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на 300,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [42]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad (4.6)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році =18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (1·300+1000·1900)·0,83·0,4·(1-0,18/100%)=421992,7 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ = (1·300+1000·(1900+1000))·0,83·0,4·(1-0,18/100%)=644315,91 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ = (1·300+1000·(1900+1000+900))·0,83·0,4·(1-0,18/100%)=844182,91 грн.

Приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.7)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, =18%;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =421992,7 /(1+0,18)1+644315,91 /(1+0,18)2+844182,91/(1+0,18)3=
= 1289906,2 грн.

Величина початкових інвестицій , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ \quad (4.8)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 325706,17 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 325706,17 = 651412,33 грн.

Абсолютний економічний ефект для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.9)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1289906,2 грн;

PV – теперішня вартість початкових інвестицій, 651412,33 грн.

$E_{абс} = ПП – PV$ = 1289906,2 - 651412,33 = 638493,87 грн.

Внутрішня економічна дохідність інвестицій , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.10)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+638493,87 /651412,33)1/3 – 1 = 0,44, \quad (4.11)$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій :

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,1+0,25 = 0,35 < 0,44 свідчить про те, що внутрішня економічна дохідність інвестицій , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою

«Інформаційна технологія онтологічного моделювання бази знань з організації бібліотеки» доцільно.

Період окупності інвестицій які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в} \ , \quad (4.12)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок} = 1 / 0{,}44 = 2{,}3$ р.

$T_{ок} < 3$-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень" становить 43 бали, що, свідчить про комерційну важливість проведення даних досліджень оскільки рівень комерційного потенціалу розробки високий.

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,95 рази.

Також термін окупності становить 2,3 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою "Система шифрування зображень. Частина 2. Підсистема розшифрування зображень"

ВИСНОВКИ

Результатами виконання даної магістерської кваліфікаційної роботи є метод та програмний засіб для відновлення зображень.

Були проаналізовані існуючі методи розподілу секрету, і виявлено, що головним недоліком є використання складних математичних обрахунків при формуванні частин секрету для їхнього подальшого розподілу.

Було проведено аналіз відомих методів візуальної криптографії, що застосовуються для розподілу секретів. Основними недоліками цих методів є складні математичні обрахунки при створенні зображень, складнощі у відновленні секрету після втрат та обмежене застосування методу чорно-білих зображень, який може використовуватися тільки для певного типу зображень.

Аналіз сучасних засобів захисту інформації дозволив визначити актуальність розробки програмного забезпечення, яке покращує захист конфіденційної інформації. Завдяки результатам аналізу було визначено відмінності між засобами захисту зображень та структурувати етапи розробки програмного засобу.

Розроблений власний метод відновлення зображень що ґрунтується на використанні перестановок, що забезпечуються генератором псевдовипадкових чисел. Головною відмінністю цього методу є те що він не використовує складних математичних обрахунків, і може застосовуватись для сучасних форматів зображення.

Було проведено експериментальне дослідження, спрямоване на відновлення секретного вмісту на зображеннях. Для цього було використано зображення. За допомогою програмного засобу, який використовували для дослідження, вдалося отримати вихідні зображення високої якості відновлення вмісту. Ці відновлені зображення відзначались високою якістю та були максимально подібні до оригіналів, зберігаючи чіткість та деталі без будь-яких відмінностей.

Таким чином усі задачі дослідження розв’язано і досягнуто мету дослідження, а саме, забезпечено пришвидшення процесу зашифрування в 4,1 рази.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шамір А.: Як поділитися секретом. Повідомлення ACM 22(11), 612–613
2. Наор, М., Шамір, А.: Візуальна криптографія. In: De Santis, A. EUROCRYPT. LNCS, том. 950. 1–12с. Шпрінгер, Гейдельберг
3. Блундо, К., Д'Арко, П., Де Сантіс, А., Стінсон, Д.Р.: Контрастні оптимальні порогові схеми візуальної криптографії. SIAM Journal on Discrete Mathematics 16(2), 224–261 с.
4. Lau, DL, Arce, GR: Сучасне цифрове напівтонування. Марсель Деккер, Нью-Йорк
5. С. М. Нікітіна, І. М. Нікітіна та А. А. Сафін "IEEE Transactions on Information Theory"
6. Ateniese, G., Blundo, C., De Santis, A., Stinson, DR: Розширені схеми для візуальної криптографії. Теоретична інформатика 250, 1–16с.
7. Ateniese, G., Blundo, C., De Santis, A., Stinson, R: Візуальна криптографія для структур загального доступу. Інформація та обчислення 129 (2), 86–106 с.
8. Yang, CN, Chen, TS: Розширені схеми обміну візуальними секретами з високоякісними тіньовими зображеннями з використанням сірих субпікселів. У: Kamel, MS, Campilho, AC (eds.) ICIAR 2005. LNCS, vol. 3656, 1184–1191с. Шпрінгер, Гейдельберг
9. Іто, Р., Кувакадо, Х., Танака, Х.: Візуальна криптографія з інваріантним розміром зображення. Транзакції IEICE E82-A(10), 2172–2177
10. Tzeng, WG, Hu, CM: Новий підхід до візуальної криптографії. Проекти, коди та криптографія 27(3), 207–227с.
11. Yang, CN: Нові візуальні схеми обміну секретами з використанням імовірнісного методу. Листи розпізнавання образів 25(4), 481–494 с.
12. Yang, CN, Chen, TS: Нові схеми обміну візуальними секретами зі зменшеним розміром із удвічі зменшеним розміром тіні. Транзакції IEICE 89-A(2), 620–625с.

13. Yang, CN, Chen, TS: Візуальна схема обміну секретами: покращення контрастності відновленого зображення за допомогою різних піксельних розширень. У: Campilho, A., Kamel, MS (eds.) ICIAR 2006. LNCS, vol. 4141, 468–479c. . Шпрінгер, Гейдельберг
14. Ян К. Н., Чен Т. С.: Схеми обміну візуальними секретами з можливістю регулювання розміру. Транзакції IEICE 88-A(9), 2471–2474
15. Мемон, Н., Вонг, П. В.: Захист цифрового медіаконтенту. Повідомлення ACM 41(7), 35–43c.
16. ван Шиндел Р.Г., Тіркель А.З., Осборн К.Ф.: цифровий водяний знак. В: ICIP(2), 86–90c.
17. Бродуей Г.В., Магерлайн К.А., Мінцер Ф.: Захист загальнодоступних зображень за допомогою видимого водяного знака. У: van Renesse, RL (ed.) Серія конференцій Товариства інженерів фотооптичних приладів (SPIE), Серія конференцій Товариства інженерів фотооптичних приладів (SPIE), том. 2659, 126–133c.
18. Wong, PW: Водяний знак для підтвердження цілісності зображення та права власності. У: PICS, IS&T – Товариство науки та технології обробки зображень,. 374–379c.
19. Луо Х., Пан Дж. С. Лу З. М. Приховування кількох водяних знаків у прозорих плівках візуальної криптографії. Інтелектуальне приховування інформації та обробка мультимедійних сигналів 1, 303–306c.
20. Hwang, RJ: Схема захисту авторських прав на цифрові зображення на основі візуальної криптографії. Tamkang Journal of Science and Engineering 3(2), 97–106c.
21. Хассан, Массачусетс, Халілі, Массачусетс: Самонанесення водяних знаків на основі візуальної криптографії. Праці Всесвітньої академії науки, техніки та технологій 8, 159– 162c.
22. Слейт, А., Абусітта, А.: Технологія водяних знаків на основі візуальної криптографії для індивідуальних і групових зображень. Система, кібернетика та інформатика 5(2), 24–32c.

23.Чуанг С.К., Хуан Ч.Х., Ву Дж.Л.: Невидимі видимі водяні знаки. В: ICIP(3), 261–264с. IEEE, Лос-Аламітос

24.. Hou, YC, Chen, PM: Асиметрична схема водяних знаків на основі візуальної криптографії. In: WCCC-ICSP 5th International Conference on Signal Processing Proceedings, 992–995c.

25.Hersch, RD, Chosson, S.: Смугові муарові зображення. У: ACM SIGGRAPH, 239–247 с. ACM, Нью-Йорк

26.Ноттс, ME, Hemphill, RG: Вибрані статті про оптичний муар і застосування. Новини оптики та фотоніки, 53–55с.

27.Кафрі О., Глатт І.: Фізика метрології Муару. Вайлі, Нью-Йорк

28. Indebetouw, G., Czarnek, R.: Вибрані статті про оптичний муар і застосування. SPIE Milestones Series, том. MS64

29.Амідрор, І.: Теорія феномена Муару. Клювер, Дордрехт

30.Хатлі, М., Стівенс, Р.: Оптична перевірка масивів і періодичних структур за допомогою муарового збільшення. У: Пошук інформації: підходи до штучного інтелекту та пошуку інформації дводенний семінар IEE, стор.

31.Камал, Х., Фелькель, Р., Альда, Дж.: Властивості муарових луп. Оптична інженерія 37 (11), 3007–3014

32.Desmedt, Y., Le, TV: Муарова криптографія. У: Конференція ACM з комп’ютерної та комунікаційної безпеки, 116–124с.

33.Лю, С., Чжан, Х., Лай, Х.: Художній ефект і застосування візерунків муаре в захисних голограмах. Прикладна оптика 34(22), 4700–4702

34.Вейр, Дж., Ян, В., Крукс, Д.: Безпечна маска для приховування кольорових зображень. У: Комунікації та мережі в Китаї, ChinaCom ,., стор. 1304–1307

35.Визуальная криптография [Електронний ресурс]. – Режим доступа : URL www.cryptowiki.net/index.php?title =Схема_ Визуальная_криптография - Назва з екрану.

36.Шифрування RGB зображення за допомогою Фурьє образів [Електронний ресурс]. – Режим доступа : URL https://github.com/lanit-tercom-school/grouplock/wiki - Назва з екрану.

37. Alfred J. Menezes; Paul C. van Oorschot; Scott A. Vanstone. Handbook of Applied Cryptography (вид. Fifth printing). CRC Press. ISBN 0-8493-8523.
38. NIST Special Publication 800-56A: "Шифрування блочного режиму є найбільш поширеним методом шифрування зображень."
39. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone . Handbook of Applied Cryptography. CRC Press. ISBN 0-8493-8523-7.
40. Моргун О.М. Криптографічні методи захисту інформації. [Електронний ресурс] – Режим доступу: http://a-morgun.narod.ru/a10-01/LK02.pdf.
41. Метод формування перестановок довільної кількості елементів В.А Лужецький, І.С. Горбенко.
42. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

73

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ
МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система шифрування зображень. Частина 2. Підсистема розшифрування зображень.

Автор роботи: Чічановський Максим Юрійович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 84,4 %. Схожість – 15,6 %.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Валентина КАПЛУН
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Максим ЧІЧАНОВСЬКИЙ
(підпис)

Керівник роботи _____ Володимир ЛУЖЕЦЬКИЙ
(підпис)

Додаток Б

Текст програми

```
import tkinter as tk
import subprocess
from tkinter import ttk, filedialog, messagebox
from PIL import Image, ImageTk
import numpy as np
import os
class SimpleCounter:
    def __init__(self, seed=0, key="0101010101010101"):
        self.state = seed
        self.feedback_mask = int(key, 2)
    def next(self):
        feedback_bit = self.state & 1
        self.state >>= 1
        if feedback_bit:
            self.state ^= self.feedback_mask
        return self.state
def load_image(file_path):
    return Image.open(file_path)
def save_image(image, file_path):
    image.save(file_path)
def generate_permutation_sequence(length, counter, forward):
    sequence = list(range(length))
    if not forward:
        sequence.reverse()
    return sequence
def get_user_key():
    key_length = 8
    user_input = input(f"Введіть ключ з {key_length} символів: ")
    while len(user_input) != key_length:
        print(f"Будь ласка, введіть рівно {key_length} символів.")
        user_input = input(f"Введіть ключ з {key_length} символів: ")
    binary_key = ''.join(format(ord(char), '08b') for char in user_input)
    return binary_key
def decrypt_subsequence(self, subsequence, permutation_sequence):
    decrypted_subsequence = []
```

```
    for i, pixel in enumerate(subsequence):
        decrypted_pixel = pixel ^ self.next()
        decrypted_subsequence.append(decrypted_pixel)
    decrypted_subsequence = [decrypted_subsequence[i] for i in permutation_sequence]
    return decrypted_subsequence
def decrypt_image_parts(self, encrypted_image_parts, key, num_counters):
    decrypted_image_parts = []
    for i in range(num_counters):
        pixels_per_counter = encrypted_image_parts[i].shape[1]
        direction_forward = int(key[i*8:(i+1)*8], 2) % 2 == 0
        decryption_counter = SimpleCounter(seed=int(key[i*8:(i+1)*8], 2), key=key)
        print(f"Decryption Seed for Part {i + 1}: {decryption_counter.state}")
        forward = direction_forward
        permutation_sequence = generate_permutation_sequence(pixels_per_counter,
decryption_counter, forward)
        decrypted_part = np.zeros_like(encrypted_image_parts[i], dtype=np.uint8)
        for j in range(encrypted_image_parts[i].shape[0]):
            subsequence = encrypted_image_parts[i][j, :, :]
            decrypted_subsequence = decryption_counter.decrypt_subsequence(subsequence,
permutation_sequence)
            decrypted_part[j, :, :] = np.array(decrypted_subsequence)
        decrypted_image_parts.append(decrypted_part)
    return decrypted_image_parts
class App:
    def __init__(self, root):
        root.title("DECRYPTION APP")
        root.configure(bg='#333333')
        width = 1000
        height = 400
        screenwidth = root.winfo_screenwidth()
        screenheight = root.winfo_screenheight()
        alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth - width) / 2,
(screenheight - height) / 2)
        root.geometry(alignstr)
        root.resizable(width=False, height=False)
        self.counter = None
        self.counter = SimpleCounter()
        self.create_widgets()
        self.image_path = None
```

```
        self.encrypted_image_paths = [
            "C:\\Encrypt\\encrypted_part_1.png",
            "C:\\Encrypt\\encrypted_part_2.png",
            "C:\\Encrypt\\encrypted_part_3.png",
            "C:\\Encrypt\\encrypted_part_4.png",
            "C:\\Encrypt\\encrypted_part_5.png",
            "C:\\Encrypt\\encrypted_part_6.png"
        ]
        self.encrypted_tk_images = [None] * 6
        self.image_path = None
    def decrypt_subsequence(self, subsequence, counter, permutation_sequence):
        decrypted_subsequence = []
        for i, pixel in enumerate(subsequence):
            decrypted_pixel = pixel ^ counter.next()
            decrypted_subsequence.append(decrypted_pixel)
        decrypted_subsequence = [decrypted_subsequence[i] for i in permutation_sequence]
        return decrypted_subsequence
    def create_widgets(self)
        button_image = Image.open("img_1.png")
        button_image = ImageTk.PhotoImage(button_image)
        style = ttk.Style()
        style.configure("TButton", font=('Times', 10), padding=5, relief="flat",
background="#333333",
                        foreground="black")
        style.configure("TCheckbutton", background="#333333", foreground="white")
        select_button = ttk.Button(command=self.choose_image, style="TButton")
        select_button.place(x=660, y=80, width=300, height=30)
        select_button.config(text="OPEN IMAGE")
        decrypt_button = ttk.Button(command=self.decrypt_image, style="TButton")
        decrypt_button.place(x=660, y=320, width=300, height=30)
        decrypt_button.config(text="DECRYPT IMAGE")
        decrypt_button.config(image=button_image, compound="center")
        app_label = tk.Label(text="DECRYPTION APP", font=('Times', 20), fg='#FFFFFF',
bg='#333333', justify="center")
        app_label.place(x=410, y=25, width=810, height=30)
        app_label = tk.Label(text="Enter key:", font=('Times', 18), fg='#FFFFFF',
bg='#333333', justify="left")
        app_label.place(x=660, y=140, width=110, height=30)
        self.key_entry = tk.Entry(root, font=('Times', 16))
```

```
        self.key_entry.place(x=780, y=140, width=180, height=30)
        options = ["1", "2", "3", "4", "5", "6"]
        self.dropdown_var = tk.StringVar(root)
        self.dropdown_var.set(options[2])  # Значення за замовчуванням (4)
        self.dropdown_menu = ttk.Combobox(root, textvariable=self.dropdown_var,
values=options, font=('Times', 12))
        self.dropdown_menu.place(x=660, y=200, width=300, height=30)
        self.image_frame = ttk.Frame(root, width=600, height=350)
        self.image_frame.place(x=330, y=200, anchor='center')
        self.label_image = tk.Label(self.image_frame, image=None)
        self.label_image.place(relwidth=1, relheight=1)
        initial_image_path = "img.png"
        self.load_and_display_initial_image(initial_image_path)
        self.image_path = None
    def load_and_display_initial_image(self, initial_image_path):
        pil_image = Image.open(initial_image_path)
        pil_image.thumbnail((600, 350))
        tk_image = ImageTk.PhotoImage(pil_image)
        self.label_image.configure(image=tk_image)
        self.label_image.image = tk_image
    def load_and_display_image(self):
        pil_image = Image.open(self.image_path)
        pil_image.thumbnail((600, 350))
        tk_image = ImageTk.PhotoImage(pil_image)
        self.label_image.configure(image=tk_image)
        self.label_image.image = tk_image
    def choose_image(self):
        file_path = filedialog.askopenfilename(
            title="Обрати зображення",
            filetypes=[("Файли зображень", "*.png;*.jpg;*.jpeg;*.bmp;*.gif")]
        )
        if file_path:
            self.image_path = file_path
            self.label_image.config(text=f"Обране зображення:
{os.path.basename(file_path)}")
            self.load_and_display_image(file_path)
    def get_user_key(self):
        key = self.key_entry.get()
        key_length = 8
```

```
        if len(key) != key_length:
            messagebox.showerror("Помилка", f"Будь ласка, введіть рівно {key_length}
символів.")
            return None
        binary_key = ''
        for char in key:
            ascii_code = ord(char)
            binary_representation = format(ascii_code, '08b')
            if ascii_code % 2 != 0:
                binary_representation = binary_representation[1:] + '0'
            binary_key += binary_representation
        return binary_key
    def get_num_counters(self):
        try:
            num_counters = int(self.dropdown_var.get())
            if 1 <= num_counters <= 6:
                return num_counters
            else:
                messagebox.showerror("Помилка", "Будь ласка, введіть число від 3 до 6.")
                return None
        except ValueError:
            messagebox.showerror("Помилка", "Неправильний ввід. Будь ласка, введіть
правильне число.")
            return None
    def open_encrypted_folder(self):
        output_folder = "C:/Encrypt/"
        os.makedirs(output_folder, exist_ok=True)
        subprocess.Popen(['explorer', output_folder])
    def save_decrypted_image(self, decrypted_image_parts):
        combined_decrypted_image = np.concatenate(decrypted_image_parts, axis=1)
        output_folder = "C:/Decrypt/"
        os.makedirs(output_folder, exist_ok=True)
        output_path = os.path.join(output_folder, "combined_decrypted_image.png")
        save_image(Image.fromarray(combined_decrypted_image.astype(np.uint8)),
output_path)
        self.load_and_display_image(output_path)

        messagebox.showinfo("Успіх", f"Об'єднане розшифроване зображення збережено в
{output_folder}")
```

```
    def load_and_display_image(self, image_path):
        pil_image = Image.open(image_path)
        pil_image.thumbnail((600, 350))
        tk_image = ImageTk.PhotoImage(pil_image)
        self.label_image.configure(image=tk_image)
        self.label_image.image = tk_image
    def decrypt_image_parts(self, encrypted_image_parts, key, num_counters):
        decrypted_image_parts = []
        for i in range(num_counters):
            pixels_per_counter = encrypted_image_parts[i].shape[1]
            direction_forward = int(key[i * 8:(i + 1) * 8], 2) % 2 == 0
            decryption_counter = SimpleCounter(seed=int(key[i * 8:(i + 1) * 8], 2),
key=key)
            print(f"Decryption Seed for Part {i + 1}: {decryption_counter.state}")
            forward = direction_forward
            permutation_sequence = generate_permutation_sequence(pixels_per_counter,
decryption_counter, forward)
            if not direction_forward:
                permutation_sequence.reverse()
            decrypted_part = np.zeros_like(encrypted_image_parts[i], dtype=np.uint8)
            for j in range(encrypted_image_parts[i].shape[0]):
                subsequence = encrypted_image_parts[i][j, :, :]
                decrypted_subsequence = self.decrypt_subsequence(subsequence,
decryption_counter, permutation_sequence)
                decrypted_part[j, :, :] = np.array(decrypted_subsequence)
            decrypted_image_parts.append(decrypted_part)
        for i in range(1, num_counters):
            if decrypted_image_parts[i].shape[1] != decrypted_image_parts[0].shape[1]:
                raise ValueError("Rounded decrypted parts do not have consistent sizes
along axis 1.")
        self.save_decrypted_image(decrypted_image_parts)
    def decrypt_image(self):
        if self.image_path is None:
            return
        key = self.get_user_key()
        if key is None:
            return
        num_counters = self.get_num_counters()
        if num_counters is None:
```

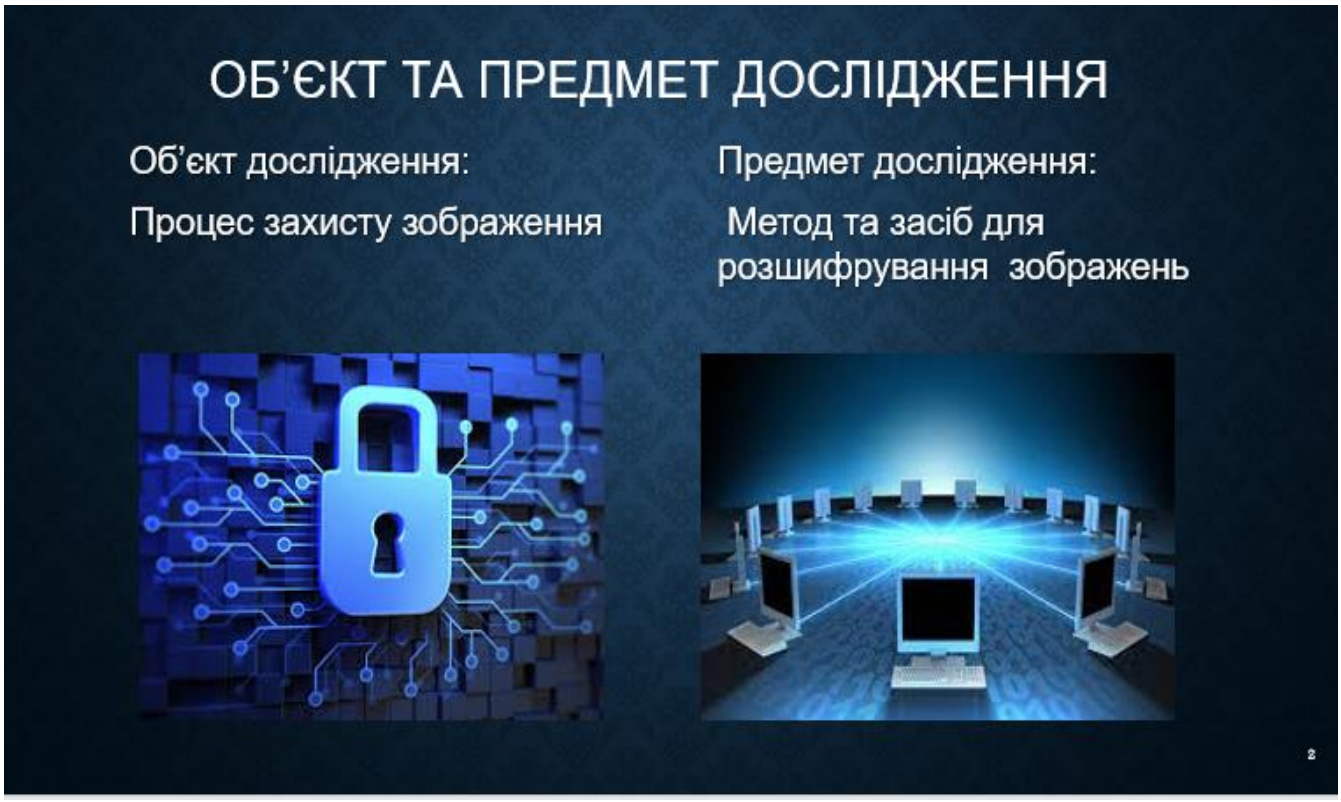
```
            return
        file_paths = filedialog.askopenfilenames(
            title="Обрати зашифровані частини",
            filetypes=[("Файли зображень", "*.png")]
        )
        if len(file_paths) != num_counters:
            messagebox.showerror("Error", f"Please select exactly {num_counters}
encrypted parts.")
            return
        encrypted_image_parts = [np.array(Image.open(file_path)) for file_path in
file_paths]
        self.encrypted_image_parts = encrypted_image_parts
        decrypted_image_parts = self.decrypt_image_parts(encrypted_image_parts, key,
num_counters)
        output_folder = "C:/Decrypt/"
        os.makedirs(output_folder, exist_ok=True)
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

Додаток В

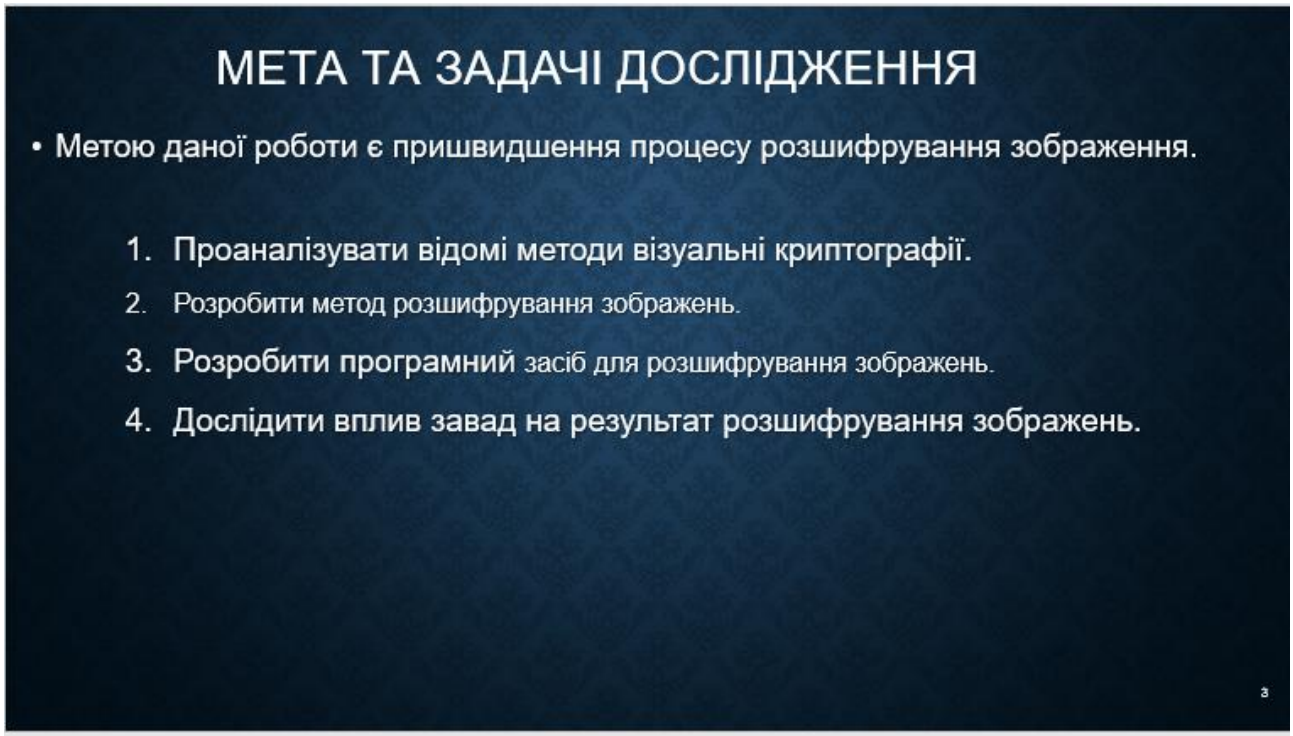
ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА ШИФРУВАННЯ ЗОБРАЖЕНЬ. ЧАСТИНА 2. ПІДСИСТЕМА РОЗШИФРУВАННЯ ЗОБРАЖЕННЯ

Об’єкт та предмет дослідження



Мета та задачі дослідження



Метод розшифрування зображення

МЕТОД РОЗШИФРУВАННЯ ЗОБРАЖЕННЯ

- Основна ідея алгоритму відновлення секрету базується на здійсненні операцій, зворотних до розбиття секрету, з метою отримання вихідного зображення. Для оцінювання коректності відновлення використовується накладання шуму на сегменти, що були отримані внаслідок алгоритму розподілення секрету з використанням генератора псевдовипадкових чисел.
- Секрет S розподілений на 3 складові S1,S2,S3
- R –інтенсивність червоного, B – синього, G-зеленого

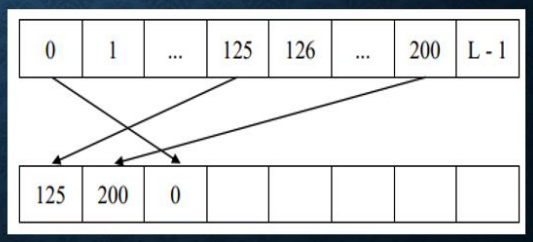
Схема перестановки байтів на основі ПВЧ

СХЕМА ПЕРЕСТАНОВКИ БАЙТІВ НА ОСНОВІ ПВЧ

Для виконання операції перестановок пікселів в зображенні потрібен генератор, який буде генерувати псевдовипадкові числа без повторень в заданому діапазоні. У сучасних шифрах перестановки виконуються лише в межах окремих блоків, а не у межах всього повідомлення, тому це не дозволяє підвищити стійкість криптопротоколів. Також сучасні методи формування перестановок не дають можливості створити окрему псевдовипадкову перестановку. Тому в роботі було запропоновано метод, який дозволяє створювати псевдовипадкову множину довільної довжини, що відповідає довжині різноманітних . Чим більше елементів в операції перестановки, тим більше варіантів перестановок. Порядкові номери байтів у файлі - це числа від 0 до n-1. Задача перестановки цих чисел полягає в тому щоб змінити їх порядок розташування. Цю задачу згідно з методом перестановки подається у вигляді формули:

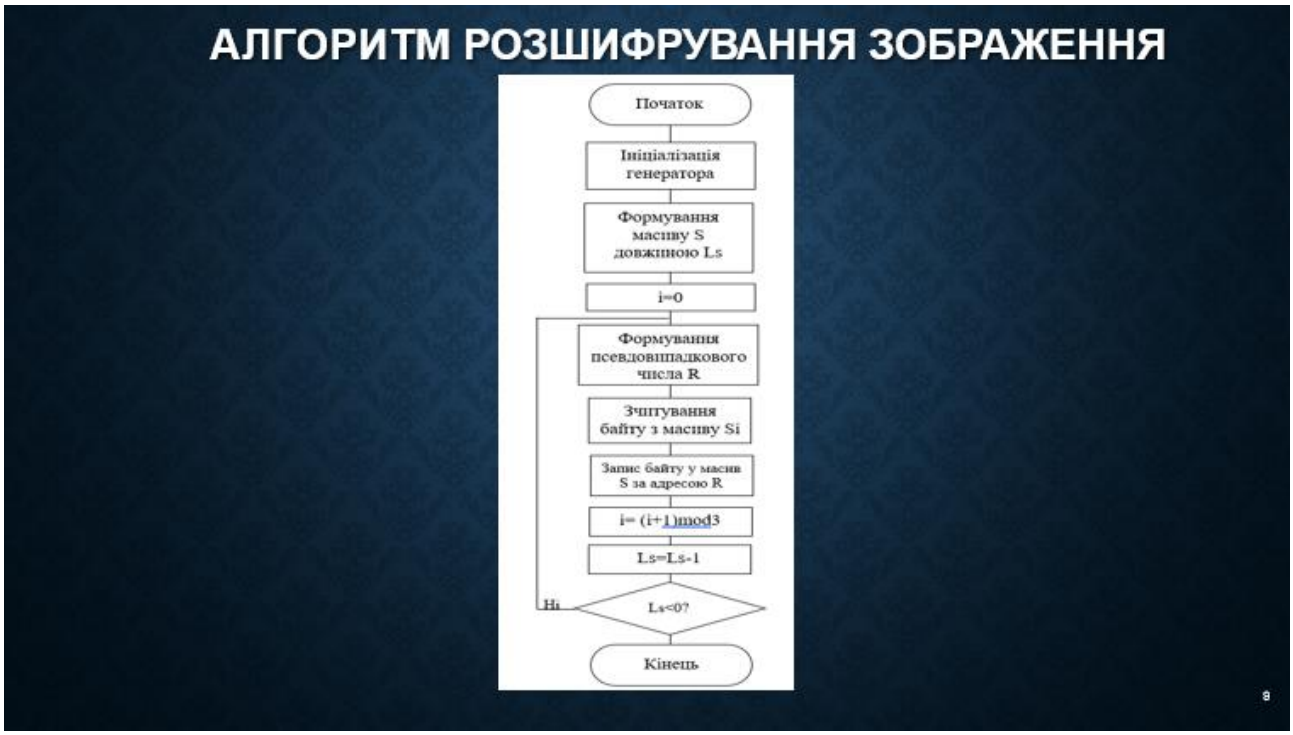
- $P = \{N, G, I, F, p\},$

- Де N – це набір послідовностей, на які розподіляється послідовність чисел.
- G – це множина функцій формування випадкових послідовностей
- I – індикатор перестановок
- F – множина правил перестановки
- P – множина перестановок у послідовностях



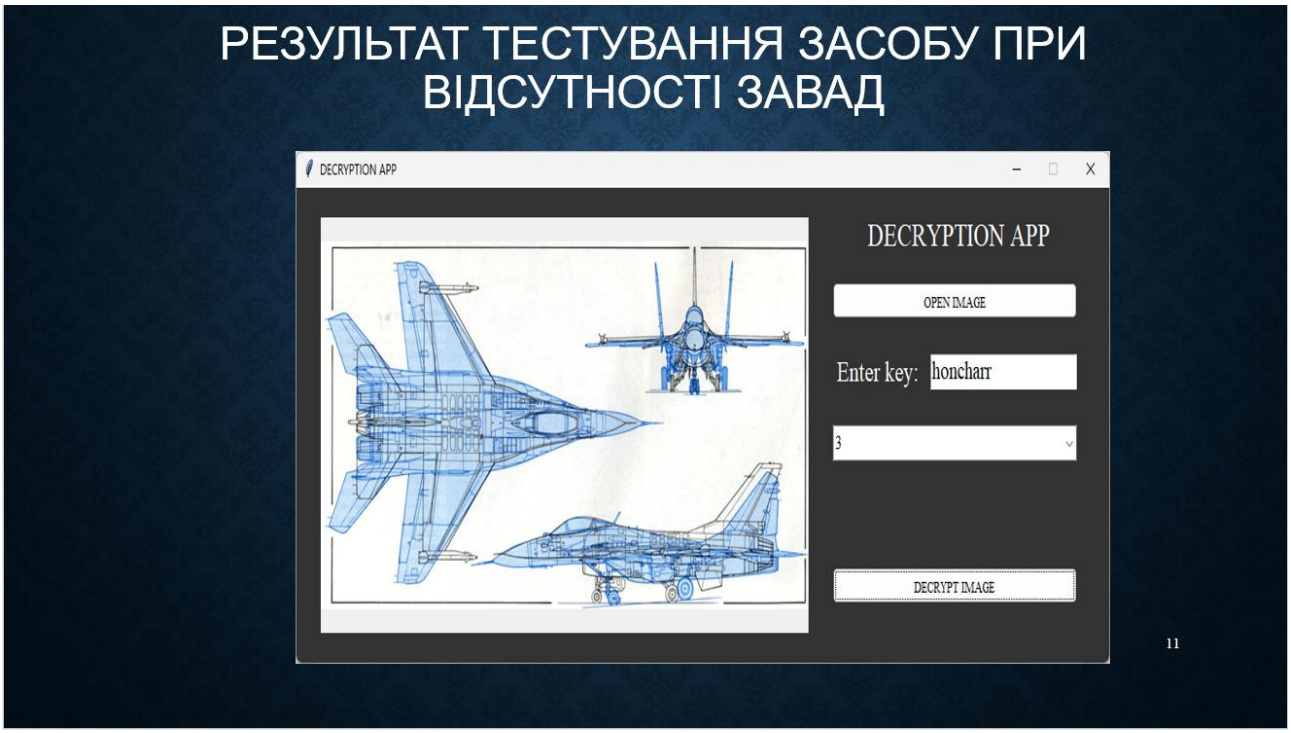
8

Алгоритм розшифрування зображення



Алгоритм ініціалізації генератора

Результати тестування засобу при відсутності завад



Результати тестування засобу при наявності завад