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

Магістерська кваліфікаційна робота на тему:
«Система шифрування зображень. Частина 1. Підсистема зашифрування зображень»

Виконав: студент 2 курсу групи 2БС-22м
спеціальності 125 Кібербезпека
_____ Олександр ГОНЧАР

Керівник: д. т. н., професор каф. ЗІ
_____ Володимир ЛУЖЕЦЬКИЙ
«14» 12 2023 р.

Опонент: к. т. н., доцент 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
«15» 09 2023 року

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

Гончару Олександру Ігоровичу

1. Тема роботи: «Система шифрування зображень. Частина 1. Підсистема зашифрування зображень»
керівник роботи: Лужецький Володимир Андрійович, д. т. н., професор кафедри ЗІ, затверджені наказом ректора ВНТУ від 18 вересня 2023 №247.
2. Строк подання студентом роботи: 13 грудня 2023р.
3. Вхідні дані до роботи:
   - Метод шифрування – на основі розподілу секрету
   - Функції розподілу секрету: (n,n), n=3, 4, 5, 6
   - Мова програмування – Phyton
   - Формат зображення: .jpg, .png.
   - Обсяг файлу зображень – не більше 15 МБ
4. Зміст текстової частини: Вступ. 1. Аналіз літературних джерел. 2. Розроблення методу розділення секрету. 3. Розроблення підсистеми шифрування зображенн. 4. Економічна частина. Висновки. Перелік використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Ініціалізація генератора (плакат А4), Схема роботи регістру зсуву з зворотнім зв'язком (плакат А4). Схема роботи алгоритму розділення секрету (плакат А4). Загальна схема роботи програми (плакат А4). Результати тестування (плакат А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф. каф. ЗІ	19.09	25.09
2	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф. каф. ЗІ	19.09	10.10
3	Володимир ЛУЖЕЦЬКИЙ, д. т. н., проф. каф. ЗІ	19.09	17.10
4	*Ольга РАТУШНЯК, к. т. н., доц. каф. ЕПВМ*	19.09	20.11

7. Дата видачі завдання 1 вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної 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	Науково-технічне обгрунтування	16.09.2023 – 22.09.2023	
4	Розробка рішень	30.09.2023 – 12.10.2023	
5	Практична реалізація, моделювання, експериментування, результати	14.10.2023 – 10.11.2023	
6	Розробка розділу економічного обгрунтування доцільного розробки	11.11.2023 – 17.11.2023	
7	Оформлення пояснювальної записки	17.11.2023 – 30.11.2023	
8	Попередній захист та доопрацювання МКР	28.11.2023 – 10.12.2023	
9	Перевірка магістерської роботи на наявність плагіату	02.12.2023 – 10.12.2023	
10	Представлення МКР до захисту	11.12.2023 – 14.12.2023	
11	Захист МКР	14.12.2023 – 21.12.2023	

Студент _____ Олександр ГОНЧАР

Керівник роботи _____ Володимир ЛУЖЕЦЬКИЙ

АНОТАЦІЯ

УДК 681.325.5

Гончар О. Система шифрування зображень. Частина 1. Підсистема зашифрування зображень. Магістерська кваліфіка-ційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2021. 105 с.

Укр. мовою. Бібліогр.: 26 назв; рис.: 20; табл.: 13; форм.: 31.

Магістерська робота присвячена вивченню та розвитку методів розподілу секрету в області візуальної криптографії. Акцент роботи робиться на використанні генераторів перестановок та псевдовипадкових послідовностей для створення ефективної системи розподілу секрету, придатної для захисту конфіденційної інформації у вимогливих умовах обробки та збереження зображень. Аналіз літературних джерел у роботі дозволяє отримати глибше розуміння сучасного стану досліджень у галузі розподілу секрету та візуальної криптографії. В роботі обґрунтовується вибір генераторів перестановок та псевдовипадкових послідовностей як основного елемента методу, що дозволяє створювати надійну систему розподілу секрету. Розроблено метод забезпечення ефективного та безпечного розподілу конфіденційної інформації. Програмний засіб структурований та високою ефективний в умовах обробки та збереження зображень.

В економічній частині оцінено затрати на розробку.

Ключові слова: зображення, шифрування, генератори перестановок та псевдовипадкових послідовностей, розподілення секрету.

ABSTRACT

Honchar O. Image encryption system. Part 1. Encrypt image system. Bachelor's thesis in specialty 125 – cybersecurity. Vinnitsa: VNTU, 2021. – 105 p.

In Ukrainian language. Bibliographer: 26 titles; fig.: 20; tabl.: 13, form.:31.

The master's thesis is devoted to the study and development of secret distribution methods in the field of visual cryptography. The emphasis of the work is on the use of permutation generators and pseudo-random sequences to create an efficient secret distribution system suitable for protecting confidential information under demanding image processing and storage conditions. The analysis of literary sources in the work allows you to get a deeper understanding of the current state of research in the field of secret sharing and visual cryptography. The work justifies the choice of generators of permutations and pseudo-random sequences as the main element of the method, which allows creating a reliable secret distribution system. A method of ensuring effective and safe distribution of confidential information has been developed. The software tool is structured and highly efficient in image processing and storage.

Development costs are estimated in the economic part.

Keywords: image, encryption, generators of permutations and pseudo-random sequences, secret distribution.

ЗМІСТ

ВСТУП

В епоху стрімкого технологічного розвитку, де обмін інформацією та обробка даних є неодмінною частиною нашого повсякденного життя, питання забезпечення безпеки конфіденційної інформації стає важливішим за будь-коли. Зокрема, коли йдеться про обробку та передачу зображень, використання методів криптографії набуває особливої актуальності.

У першій частині магістерської роботи виконано докладний аналіз сучасних літературних джерел з області розподілу секрету. Зосереджуючись на основних ідеях цього напрямку, робота розглядає різноманітні схеми та методи візуальної криптографії, вивчаючи їхню застосовність та вагомі переваги.

Особлива увага приділяється використанню випадкових чисел та генераторів, що є ключовим елементом в створенні безпечних систем розподілу секрету. Досліджуючи ці аспекти, робота спрямована на розробку ефективного та надійного методу захисту конфіденційної інформації у контексті обробки зображень.

Після вивчення сучасних методів розподілу секрету та визначення їхнього теоретичного фундаменту, в роботі пропонується новий метод, який ґрунтується на узагальненому описі та використанні генераторів перестановок та псевдовипадкових послідовностей. Цей метод визначає алгоритм розподілення секрету, який не лише забезпечує високий рівень надійності, але і виявляється ефективним у захисті конфіденційної інформації.

Детальний огляд розробленого програмного засобу допомагає визначити його ключові аспекти. Обґрунтування вибору середовища програмування розкриває причини вибору конкретного технічного підходу, забезпечуючи високу ефективність та зручність розробки. Вимоги до програмного засобу ретельно визначають функціональність та технічні характеристики, гарантуючи відповідність поставленим завданням.

Загальна схема функціонування програми надає чіткий огляд процесів, які відбуваються в системі. Вибір формату зображень визначається з огляду на вимоги ефективності та збереження конфіденційності даних. Реалізація розподілу секрету та інтерфейсу є ключовим етапом у виведенні теоретичних концепцій у практичний план, а отримані результати використовуються для підтвердження ефективності розробленого програмного засобу.

Об’єкт дослідження – процес пришвидшення процесу захисту зображеннь.

Предмет дослідження – метод та засіб для зашифрування зображень.

Мета – пришвидшення процесу зашифрування зображень шляхом розробки методу зашифрування зображень на основі розподілення секрету та засобу що його реалізує.

Наукова новизна роботи полягає в тому, що розроблено метод зашифрування зображень, який відрізняється від відомих тим, що передбачає розділення конфіденційної інформації на певну кількість частин з використанням операцій перестановок і який забезпечує пришвидшення процесу зашифрування.

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

1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Основна ідея розподілу секрету

Шифрування походить від грецького слова "Кріпто", що означає прихований, і "Графо", що означає написаний. Це вивчення та застосування технік для приховування інформації або просто для захисту повідомлення чи тексту від читання. Інформація, яку захищають, може бути писаним текстом, електронними сигналами, електронними листами або передачею даних. Процес зроблення інформації незрозумілою - це шифрування, а результатом шифрування є шифротекст або криптограма. Відновлення оригінальної зрозумілої інформації шляхом розшифрування називається дешифруванням або розшифруванням. Для шифрування чи дешифрування інформації використовується алгоритм або так званий шифр [1].

З часів існування людства люди мали секрети, і інші люди хотіли знати ці секрети. Перші форми шифрування виконувалися з допомогою олівця та паперу і були доступні лише тим, хто мав доступ до належної освіти. Сьогодні наше життя повністю цифроване, і шифрування стало невід'ємною частиною майже щоденного життя кожного з нас, і воно використовується для захисту конфіденційної інформації від хакерів. Майже всі наші особисті дані зберігаються в одній з численних баз даних від уряду, банків, служб охорони здоров'я і т. д. Шифрування захищає право на приватність та право на конфіденційне спілкування [1].

Безпечне спілкування може захищати особисте приватне життя, бізнесові відносини та соціальні чи політичні діяльності.

Шифрування має довгу та захопливу історію , і це одна з найважливіших галузей в сфері безпеки. Візуальне шифрування використовує характеристики людського зору для розшифрування зашифрованих зображень. У ньому секретне зображення розбивається на два або більше окремих випадкових зображень, які називаються "частками". Для розшифрування зашифрованої інформації частки розташовують одна на одній, і секретне зображення з'являється. Завдяки своїй

простоті, кожен може фізично маніпулювати елементами системи і візуально спостерігати за процесом розшифрування без знання криптографії і без проведення криптографічних обчислень [2].

Зі всеосяжним використанням Інтернету в усіх галузях, необхідність обміну важливими документами з одного офісу до іншого через цей засіб стає все більш необхідною. З наближенням ери електронної комерції існує негайна потреба в розв'язанні проблеми забезпечення безпеки інформації в сьогоднішньому все більш відкритому мережевому середовищі. Для захисту безпеки інформації зазвичай використовуються різні технології шифрування традиційної криптографії. За допомогою таких технологій дані можуть бути зрозумілими після шифрування та пізніше відновлені за допомогою вірного ключа. Без вірного ключа зашифрований вміст джерела може бути важко виявити, навіть якщо несанкціоновані особи вкрадуть дані [2].

1.2 Схеми розподілу секрету

Коли важлива секретна інформація керується фізичними особами, секрети можуть витікати. Припустимо, є сховище, яке повинно відкриватися щодня в банку. Незважаючи на те, що банк працює з трьома старшими касирами, управління може не довіряти жодному окремому касиру. Тому необхідно знайти можливе рішення для створення системи, за допомогою якої будь-які два з трьох старших касирів зможуть отримати доступ до сховища, але жоден окремий касир не зможе цього зробити. Цю проблему можна легко вирішити за допомогою схеми розподілу секретів [3].

У більш загальній ситуації може знадобитися вказати, які саме підмножини учасників повинні мати можливість визначити ключ, а які - ні. Схеми розподілу секретів корисні в багатьох ситуаціях, де потрібна згода кількох вибраних осіб, як в запуску ракети чи входженні в зону обмеженого доступу (наприклад, в банківське сховище) [3].

Схема розподілу секрету розподіляє (ділить) секретний ключ K серед обмеженої множини n учасників таким чином, що лише певні визначені

підмножини (кваліфіковані підмножини) учасників можуть обчислити секретний ключ K, збираючи свою інформацію. Цю схему було відкрито незалежно Г.Р. Блейклі та Аді Шаміром. Схема секретного розподілу Шаміра є інтерполяційною схемою, заснованою на інтерполяції полінома, тоді як схема секретного розподілу Блейклі має геометричний характер. Обидві схеми є схемами k-з-n, але вони представляють два різних способи створення таких схем, на основі яких можна розробити більш вдосконалені схеми секретного розподілу. Найбільша мотивація для секретного розподілу - це безпечне керування ключами. Зокрема в певних ситуаціях існує лише один секретний ключ, який надає доступ до багатьох важливих файлів. Якщо такий ключ втрачений, то всі важливі файли стають недоступними [3, 13].

Основна модель для секретного розподілу називається схемою k-з-n (іноді відомою як (k, n) порігова схема). У цій схемі є відправник і n учасників. Секретна інформація ділиться на n частин відправником, і він дає кожному учаснику одну частину так, що будь-які k частин можуть бути об'єднані для відновлення секрету, але будь-які k - 1 частини не розкривають жодної інформації про секрет. Ці частини зазвичай називаються частками або тінями. Різний вибір значень k та n відображає компроміс між безпекою та надійністю. Схема секретного розподілу є ідеальною, якщо будь-яка група не більше k - 1 учасника (внутрішніх осіб) не має переваги в відгадуванні секрету в порівнянні зі сторонніми особами [3].

У пороговій схемі секрет може бути будь-якого типу даних. Наприклад, це може бути зображення I, яке складається з чорних і білих пікселів. Секретне зображення I може бути закодовано як двійковий рядок K = K(I), де 1 представляє чорний піксель, а 0 - білий піксель. За допомогою будь-якої зручної схеми розподілу секрету можуть бути побудовані частки для K. Пізніше K буде відновлено за допомогою відповідного алгоритму для схеми розподілу секрету. Зображення I перетворюється назад за допомогою отриманого двійкового рядка. Однак у цій базовій схемі розподілу секрету для обміну секретом та

розкодування секрету з відкритими даними потрібні криптографічні обчислення за допомогою комп'ютера. У всіх схемах розподілу секрету необхідна значна складність для шифрування та розшифрування секрету, тому комп'ютери є необхідними [4].

Питання, яке було запитано Кафрі і Кереном, полягало в тому, чи можливо створити схему розподілу секретів, в якій секретне зображення I може бути візуально відновлено шляхом накладання випадкових сіток? Кожна сітка складається з прозорості, створеної з чорних і білих пікселів. Пізніше Наор і Шамір представили конкретну реалізацію, яку назвали візуальним розподілом секрету (VSS). Цей метод може безпечно розподіляти інформацію зображення (надрукований текст, рукописні нотатки, зображення тощо), і можливо розкодувати поділені секрети за допомогою візуальної системи людини. На основі секретного повідомлення (початкового зображення) схема VSS генерує n зображень (відомих як частки), які можуть бути надруковані на n прозоростях. У схемі k-з-n буде n прозоростей, і якщо на них накладаються будь-які k або більше, ніж k прозоростей, початкове секретне зображення I повинно з'явитися, але жодної інформації про початкове зображення не можна отримати, якщо накладено менше, ніж порігова кількість з k - 1 частки) [3].

Основна відмінність між традиційною поріговою схемою та візуальною поріговою схемою полягає в способі відновлення секрету. Традиційна порігова схема зазвичай передбачає обчислення в скінченному полі; у візуальній поріговій схемі обчислення виконуються візуальною системою людини. У обох типах схем умови безпеки однакові [5].

Рекурсивне приховування секретів вперше було представлено М. Гнанагурупараном та Субхашем Каком з застосуванням до зображень та надрукованого тексту з метою підвищення ефективності візуальної криптографії та можливості включити додаткову секретну інформацію, яка служить стеганографічним каналом. Ідея, що має відношення до рекурсивного приховування секретів, полягає в тому, що декілька повідомлень може бути

приховано в одній з часток початкового секретного зображення. Секретні зображення, які мають бути приховані, вибираються відповідно до їхніх розмірів від найменшого до найбільшого (тобто розмір секрету подвоюється на кожному кроці). Найменше секретне зображення ділиться на n частин за основною ідеєю візуальної криптографії. Ці n часток розміщуються одна під одною, і тепер вони представляють першу частку секретного зображення. Друга частина створюється таким чином, що якщо накладено n часток, то секретне зображення розкривається під час розгляду. Цей процес повторюється рекурсивно. Важливо зазначити, що частина початкового секретного зображення, яка містить рекурсивно приховану інформацію, також повинна містити обидві частки останнього секретного зображення, яке приховується. Щодо початкового секретного зображення це накладає обмеження на розмір зображень [5, 6].

1.3 Методи візуальної криптографії

Візуальна криптографія - це потужний метод шифрування, який дозволяє приховати інформацію в зображеннях таким чином, що її можна розшифрувати за допомогою людського зору, якщо використовується правильне ключове зображення. Вона використовує два або більше тимчасових зображень (відомих як частки). Одне зображення містить випадкові пікселі, а інше - секретну інформацію. З одного зображення неможливо відновити секретну інформацію. Для розкриття секретної інформації потрібні прозорі зображення або шари.

Найпростіший спосіб реалізації візуальної криптографії - це друк двох шарів на прозорому аркуші. Коли випадкове зображення містить дійсно випадкові пікселі, його можна розглядати як систему одноразового блокноту і вона пропонує недоламний метод шифрування. Під час накладання анімації шарів можна спостерігати, проковзуючи два шари один над одним до тих пір, поки вони правильно вирівняються, і прихована інформація з'являється.

У візуальному розподілі секрету біт повідомлення складається з колекції чорних і білих пікселів, тобто вважається, що він є бінарним зображенням, і кожен піксель обробляється окремо. Кожен початковий піксель з'являється в n

модифікованих версіях (частках) зображення, по одній для кожної прозорості. Кожна частина складається з m чорних і білих підпікселів. Кожна частка підпікселів друкується на прозорості в непосередній близькості одна від одної (щоб найкраще підтримувати сприйняття людей, зазвичай розташовуються разом, щоб сформувати квадрат із m обраним як числом). Кінцева структура може бути описана булевою матрицею [n × m] S = (Sij)n × m, де sij = 1 тоді і тільки тоді, коли j-й підпіксель в і-й частці (прозорості) чорний, і sij = 0 тоді і тільки тоді, коли j-й підпіксель в і-й частці (прозорості) білий. Коли прозорості i1, i2, . . . ir накладаються одна на одну так, щоб правильно вирівняти підпікселі, ми бачимо об'єднану частку, чорні підпікселі якої представлені булевим "або" рядків i1, i2, . ir y S. Сірий рівень цієї об'єднаної частки пропорційний вага Хеммінга H(V) "або" ed m-вектора V. Цей сірий рівень інтерпретується візуальною системою користувачів як білий, якщо H(V) < d- αm, і як чорний, якщо H(V) ≥ d для фіксованого порога 1 ≤ d ≤ m і відносної різниці α > 0 [7].

Початкове зображення розбивається на дві частки, які є його частками. Окремо ці частки виглядають як випадковий шум, але при їх об'єднанні відкривається зображення (див. рис. 1). Кожен окремий піксель розбивається на підпікселі, і людське зорове сприйняття все ще сприймає їх як один піксель.

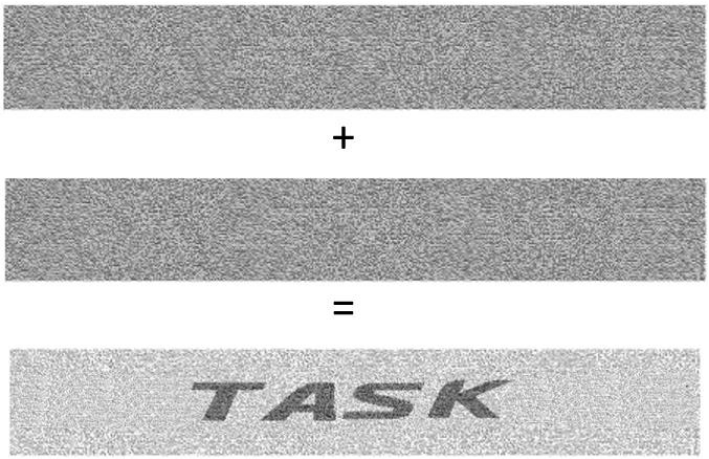


Рисунок 1 – Розбиття зображення на сегменти

Щоб спробувати це, можна скопіювати частку 1 і 2 та надрукувати їх на прозорому аркуші або тонкому папері. Все одно використовуйте програму, яка

відображає як чорні, так і білі пікселі правильно вирівняні, і встановіть принтер так, щоб всі пікселі були надруковані точно [7].

Схема кодування Two-out-of-Two полягає в тому, щоб розділити бінарне зображення на дві різні частки - Частина 1 і Частина 2. Кожен піксель розбивається на чорний і білий підпіксель, які розташовані поруч один з одним.. Коли ці частки розміщуються одна на одній, пікселі візуально об'єднуються за допомогою операції "OR", і тому білий піксель виглядає сірим (половина чорний і половина білий) для людського ока. Пікселі обираються аналогічно і для чорного пікселя. Але коли підпікселі візуально об'єднуються, два чорних підпікселя, розташовані поруч, виглядають як один чорний піксель [8].

Проблему візуального розподілу секрету "2 з 2" можна вирішити за допомогою такої колекції матриць розміром n × n:

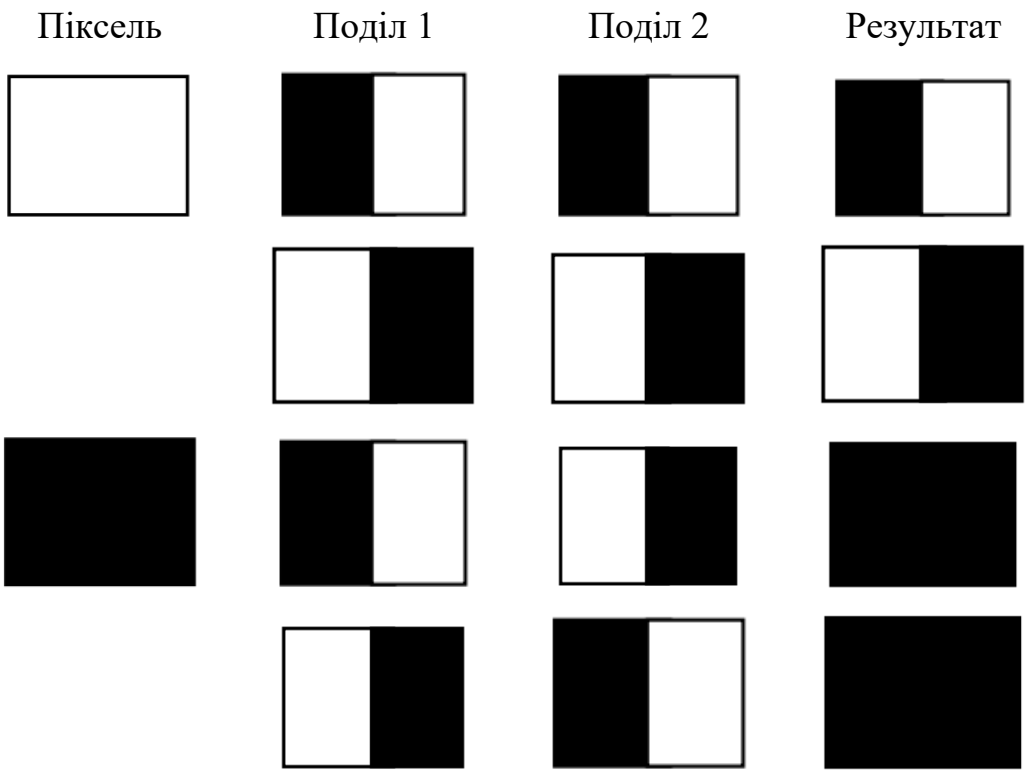


Рисунок 2 – Розподіл секрету методом «2 з 2»

Початкова проблема візуальної криптографії є особливим випадком проблеми розподілу секрету "Два з двох". Її можна вирішити з використанням 2

підпікселів на піксель, але на практиці це може спотворити співвідношення сторін початкового зображення. Тому рекомендується використовувати 4 підпікселя, розташованих у масиві 2 × 2, де кожна частина має одну з візуальних форм(рис.3). Білий піксель розподіляється на два ідентичних масиви зі списку, а чорний піксель розподіляється на два доповнюючих масиви зі списку. Будь-яка окрема частина - це випадковий вибір двох чорних і двох білих підпікселів, що виглядає середньо-сірим. Коли дві частки розміщуються одна на одній, результат - це або середньо-сірий (який представляє білий), або повністю чорний [8].

Горизонтальний поділ Вертикальний поділ Діагональний поділ

Рисунок 3 – Схема поділу 2 з 2 з чотирьма підпікселями

Горизонтальні, вертикальні та діагональні частки, описані на рисунку-2, використовуються для вирішення наступної схеми "3 з 3". Шість часток, описаних нижче рядками C0 і C1, є точними шістьма 2 × 2 масивами підпікселів. Розміщуючи всі три прозорості з C0 і C1, ми можемо побачити, що C0 містить лише ¾ чорного кольору, тоді як C1 повністю чорний [8].

Можна побачити, що піксель, розділений на чотири частини, може мати шість різних станів. Якщо піксель на шарі 1 має певний стан, то піксель на шарі 2 може мати один з двох станів: ідентичний або обернений щодо пікселя на шарі 1. Якщо піксель на шарі 2 ідентичний до шару 1, то на накладеному пікселі буде половина чорного і половина білого (рис.4). Такий накладений піксель називається сірим або порожнім. Якщо стани пікселів на шарі 1 і 2 обернені або протилежні, то на накладеній версії буде повністю чорний піксель. Це інформаційний піксель. Якщо стани пікселів на шарі 1 є дійсно випадковими (крипто захищеними), то і порожні і інформаційні пікселі на шарі 2 також

матимуть абсолютно випадкові стани. Неможливо знати, чи піксель на шарі 2 використовується для створення сірого чи чорного пікселя, оскільки нам потрібен стан цього пікселя на шарі 1 (який є випадковим), щоб знати результат накладення [9].

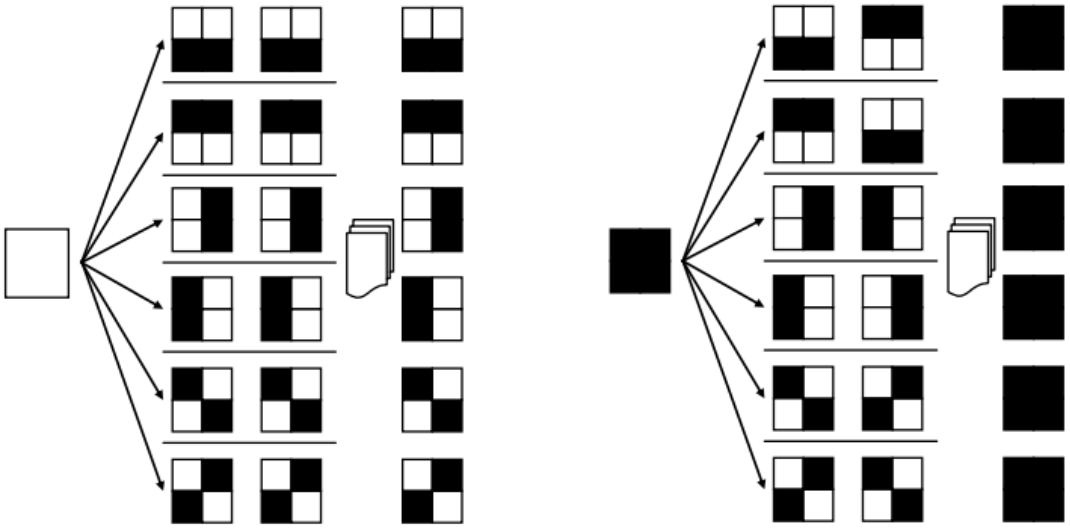


Рисунок 4 – Стани пікселя розділеного на 4

Якщо всі вимоги до справжньої випадковості виконані, візуальна криптографія надає абсолютну конфіденційність відповідно до інформації. Відправник розподілить один або кілька випадкових шарів 1 заздалегідь одержувачеві. Якщо відправник має секретне повідомлення, він створює шар 2 для певного розподіленого шару 1 і надсилає його одержувачеві. Одержувач вирівнює два шари, і секретна інформація розкривається, і це все без необхідності використовувати пристрій шифрування, комп'ютер або виконувати будь-які математичні обчислення вручну. Вся система недоламана, поки обидва шари не потраплять в невірні руки. Коли перехоплюється один з обох шарів, неможливо отримати зашифровану інформацію [9].

Загалом існує чотири критерії, які використовуються для оцінки продуктивності схеми (k, n) візуального розподілу секрету. Перший критерій - це безпека: менше k тіней не надають жодної інформації про секретне зображення, де $k \leq n$. Другий критерій - це точність: схожість між відновленим зображенням і оригінальним. Наступний критерій - це обчислювальна

складність: кількість операцій, необхідних для створення тіней та генерації відновленого зображення. Останній критерій - це розмір тіні, який також називається проблемою розширення пікселів [10].

1.4 Випадкові числа та генератори

Для розділення секрету та перестановки пікселів часто використовуються випадкові числа та псевдовипадкові генератори.

Випадкові числа є природною частиною інформатики. Вони мають багато варіантів використання, для наприклад у математичному моделюванні, безпеці чи індустрії розваг. Особливо у сфері безпеки процес створення випадкових чисел повинен відповідати різним вимогам.

Однією з них є неможливість передбачити наступне число в послідовності заданого або більше попередніх значень. Інша критична необхідність — рівномірний розподіл виробленого числа по всьому спектру можливих значень.

Генератори псевдовипадкових чисел (PRNG) здебільшого використовуються комп'ютерами для генерації випадкових послідовностей. Вони містять алгоритм, який отримує деяку вихідну інформацію (seed); на основі отриманої інформації алгоритм детерміновано виробляє а послідовність випадкових чисел. Коли вихідне число використовується двічі, алгоритм працює еквівалентні послідовності, однак, коли її змінено, результуюча послідовність повинна бути іншою.

Природним способом перевірити, чи відповідають генератори вимогам, є спроба атакувати їх з усіма відомими атаками та визначити, чи можуть вони протистояти. Добре відомі атаки лінійні і диференціальний криптоаналіз, який використовує знання алгоритму для визначення деяких секретна інформація. Інший тип атак заснований на аналізі рівномірного розподілу

вихідної послідовності. Це можна вважати атаками чорної скриньки, оскільки знання алгоритму не є обов'язковим. Основна ідея полягає в тому, що повинна бути довга послідовність випадкових чисел відповідати деяким статистичним властивостям. Існує багато тестів, кожен з яких аналізує певну

статистичну характеристику. Результат кожного тесту має форму ймовірності того, що а точний випадковий генератор створить послідовність з гіршими або такими ж статистичними властивостями. Ці тести згруповані в так звані батареї тестів, наприклад, NIST STS [11].

Коли послідовність із PRNG досягає розумної ймовірності, ми говоримо що він виглядає випадковим або також що його неможливо відрізнити від випадкових даних. Тому це процес також називають тестуванням на випадковість.

Випадковий генератор - це все, що дає випадковий результат. У реальному світі це може бути монетою, кубиками тощо. У інформатиці ми розрізняємо два типи генераторів випадковості: дійсно генератори випадкових чисел (TRNG) і генератори псевдовипадкових чисел (PRNG). Детермінований процес не може створити істинну випадковість – деякі необхідний недетермінізм, також відомий як ентропія. Взаємодія комп'ютера з деякими фізичні компоненти, такі як пристрої введення (клавіатура, миша, мікрофон тощо), можуть бути використовується як джерело недетермінізму. Однак, це може бути так, ці компоненти виробляють меншу кількість ентропії, ніж вимагає комп'ютер; наприклад, якщо зловмисник здатний контролювати ці пристрої. Тому ми не можемо покладатися тільки на це джерело випадкових даних [12].

З цієї причини були введені PRNG. Ідея PRNG детерміновано (використовуючи алгоритм) виробляють, здавалося б, випадкову (псевдовипадкову) послідовність на основі невелика кількість ентропії (насіння). Зверніть увагу, що послідовність може генерувати будь-хто хто знає оригінальний алгоритм і насіння [12].

Алгоритм із PRNG викликається ітеративно, і кожен виклик створює вихідні дані фіксованого розміру.

PRNG зберігає внутрішній стан S, який повністю визначає значення наступного виходу. I-та ітерація складається з двох кроків: генерація наступного

виходу Xi+1 на основі поточного стану Si отримання Si+1 модифікацією Si. Насіння зазвичай представляє початковий стан S0.

У деяких випадках внутрішній стан Si збігається з попереднім виходом Xi−1. У цьому випадку генератор може бути описаний детермінованою функцією f , яка викликається для генерації і-го виходу за допомогою формули

$$Xi = f(Xi-1) \quad (1.1)$$

Оскільки розмір стану фіксований, існує кінцева кількість варіантів входів і відповідних виходів, що в кінцевому підсумку призводить до повторень. Кількість ітерацій перед першою подвійність називається періодом. Тривалість періоду також залежить від насіння. Генератори створені так, щоб термін був максимальним для будь-якого насіння. При виборі PRNG слід враховувати термін застосування; його треба вибирати так, щоб період ніколи не досягався. Однак цього можна досягти регулярним пересівом із дійсно випадковим посівом даних [12].

Насправді сучасні PRNG періодично змінюють внутрішній стан, використовуючи справді випадкові дані – вони називаються гібридними PRNG. Безпеку покращує можливість відновлення скомпрометований стан, шляхом зміни внутрішнього стану таким чином, що зловмисник не може передбачити. Однак гібридні PRNG детерміновані, лише між двома споживачами більш істинні випадкові дані [13].

Оскільки всю послідовність, породжену PRNG, можна визначити з насіння, важливо, з точки зору безпеки, тримати насіння в таємниці і генерувати його так, щоб він був непередбачуваним і дійсно випадковим.

Послідовність, вироблена PRNG, повинна виглядати випадковою. Це означає, що він також повинен пройти всі емпіричні тести випадковості з високою ймовірністю [13].

Для використання PRNG в рамках чутливих додатків (таких як криптографічна програма), йому потрібно виконати ще одну вимогу. Це повинно бути неможливо обчислити будь-яку інформацію про попередній/наступний

вихід, враховуючи, що атакуючий знає частину згенерованої послідовності, включаючи алгоритм і апаратну специфікацію. Такі генератори називаються криптографічно безпечними псевдовипадковими генераторами (CSPRNG). [13]

Лінійно конгруентні генератори псевдовипадкових чисел (LCG) - це один із видів генераторів псевдовипадкових чисел (ПГЧ), які використовуються для створення послідовностей чисел, що мають властивості, схожі на випадкові числа. LCG базуються на простих математичних операціях, таких як додавання, множення та модульна арифметика [14].

Основні компоненти лінійно конгруентного генератора:

1. Початковий стан (сід): LCG починає свою роботу з початкового значення, відомого як "сід" (seed). Сід визначає початковий стан генератора, і від нього починається створення послідовності псевдовипадкових чисел.

2. Параметри генератора: LCG має декілька параметрів, таких як:

- `a` (множник): це число, яке використовується для множення поточного числа в послідовності.

- `c` (прибавка): це число, яке додається до поточного числа.

- `m` (модуль): модуль - це число, до якого взято по модулю, щоб забезпечити, що числа завжди перебувають у певному діапазоні.

Починаючи зі сіда `X0`, наступне число в послідовності обчислюється наступним чином:

$$X1 = (a * X0 + c) \% m \quad (1.2)$$

Де `%` вказує на операцію взяття по модулю.

Таким чином, LCG генерує послідовні числа X0, X1, X2 і так далі. Важливо вибирати параметри `a`, `c` та `m` так, щоб генератор був стійким та генерував числа з більшою властивістю випадковості. Поганий вибір параметрів може призвести до непередбачуваних властивостей генерованих чисел.

PRNG в криптографії використовуються для декількох цілей, наприклад, для генерації ключі, вектори ініціалізації, але також для шифрування та

дешифрування всередині симетричної криптографії. У цьому розділі детально розглянуто останнє використання [17].

Шеннон теоретизував, що досягти досконалої секретності можливо лише у випадку, якщо кількість можливих ключів більша або дорівнює низці можливих повідомлень. Ідеальна секретність означає, що в шифротексті немає інформації про відкритий текст. Одноразова криптосистема pad досягає цієї властивості, маючи ключ довше, ніж повідомлення. Для шифрування, це використовує простий біт за бітом XOR відкритого тексту з великою по-справжньому випадковою послідовністю, що не повторюється як ключ. Однак розподіл такого ключа ускладняється, і в разі, якщо буде схема обміну таким ключем безпечним способом, вона може бути використана для відправки повідомлення сам [15].

Потокові шифри були введені з таким же принципом на увазі, як одноразова накладка. Однак замість того, щоб використовувати довгу по-справжньому випадкову послідовність як ключ, вихід з CSPRNG використовується. Шифротекст C створюється з відкритого тексту P і keystream (генерується з CSPRNG) K наступним чином: Ci = Pi $\oplus$ Ki, для кожного біта у відкритому тексті. Для зашифрованого спілкування обидві сторони повинні знати ключ, який є нічим іншим, як насінням для CSPRNG, щоб виробляють один і той же ключ). Потокові шифри корисні для довгі потоки (такі як аудіо або відео потоки), особливо для їх нульового поширення помилок [16].

Це означає, що якщо один біт буде перевернуто під час передачі даних по мережі, лише один біт перевертається у вигляді розшифрованого відкритого тексту. Наприклад, якщо така ж ситуація виникає під час використання а блоковий шифр у режимі CBC весь блок розбивається після дешифрування [16].

Однак блочні шифри можуть діяти подібно до потокових шифрів. Наприклад, у режимі вихідного зворотного зв’язку (OFB) або режимі лічильника (CTR) роботи для блокових шифрів відкритий текст ніколи не проходьте через сам шифр. Криптографічний примітив, в даному випадку, служить лише як а

генератор потоку ключів, який виконує XOR з відкритим текстом, подібно до потокових шифрів [18].

Генератори псевдовипадкових чисел (ПГЧ) мають різні переваги в криптографії та системах захисту зображень. Ось декілька з них:

1. Стійкість до атак: Генератори ПГЧ можуть створювати послідовності чисел, які дуже важко відрізнити від випадкових чисел без знання сіда та параметрів генератора. Це робить їх важливими для створення ключів та векторів ініціалізації в криптографічних протоколах.
2. Випадковість: Генератори ПГЧ можуть забезпечити псевдовипадкові числа, які корисні в різних криптографічних задачах, включаючи шифрування та підписи. Вони можуть допомогти створити ключі та дані, які надійно приховують інформацію.
3. Відтворюваність: Генератори ПГЧ можуть бути відтворюваними, тобто якщо ви знаєте сід, ви завжди можете відновити ту саму послідовність чисел. Це важливо для ситуацій, де потрібно відтворити той самий ключ чи вектор ініціалізації.
4. Швидкість: Багато генераторів ПГЧ працюють дуже швидко, що робить їх ефективними для великих обчислень, таких як криптографічні операції або обробка зображень.
5. Застосування в стеганографії: Генератори ПГЧ можуть використовуватися для генерації ключів та інших даних, які приховують інформацію в стеганографічних застосуваннях, де важливо зберігати приховані дані.
6. Випробувальні методи: Важливою перевагою генераторів ПГЧ є те, що їх якість може бути випробувана за допомогою різних статистичних тестів. Це дозволяє визначити, наскільки вони генерують випадкові числа та наскільки стійкі вони до криптоаналізу.

Зазначимо, що вибір правильного генератора ПГЧ та правильних параметрів дуже важливий, особливо в криптографічних застосуваннях.

Неправильний вибір може призвести до низької стійкості та небезпеки для системи. Тому важливо ретельно вивчати та вибирати генератори для конкретних завдань [19].

Хоча генератори псевдовипадкових чисел (ПГЧ) мають свої переваги, вони також мають деякі недоліки та обмеження. Ось деякі з них:

1. Відсутність справжньої випадковості: Генератори ПГЧ базуються на детермінованих математичних алгоритмах, і, отже, їхні числа не є дійсно випадковими. Це означає, що при великому обсязі даних або недостатньому складності алгоритму можуть виникнути відмінності від реальних випадкових подій.
2. Непередбачуваність сіда: Якщо сід (початковий стан) генератора ПГЧ не вибирається випадково або не захищається від доступу, атакуючий може вгадати сід і відтворити всю послідовність чисел.
3. Періодичність: Багато генераторів ПГЧ мають обмежений період, після якого послідовність чисел повторюється. Це може призвести до непередбачуваних проблем у деяких випадках.
4. Властивості статистики: Не всі генератори ПГЧ мають властивості статистичної рівномірності та незалежності, які важливі в криптографії. Деякі генератори можуть відзначатися деякими недоліками у розподілі згенерованих чисел.
5. Криптоаналіз: Деякі генератори ПГЧ можуть бути вразливі до криптоаналізу, особливо якщо їх параметри не вибрані правильно. Атаки можуть включати в себе визначення сіда, відновлення параметрів та інші методи.
6. Не відповідність криптографічним стандартам: Більшість генераторів ПГЧ не відповідають криптографічним стандартам, і вони не підходять для використання в справжніх криптографічних застосуваннях без додаткових заходів захисту.

7. Залежність від параметрів: Вибір правильних параметрів (таких як множник, прибавка і модуль) є важливим завданням, і неправильний вибір може призвести до поганої якості генерованих чисел.

У криптографії та системах захисту інформації, де потрібна велика ступінь випадковості та стійкості, часто використовують більш складні генератори ПГЧ або інші методи створення ключів та випадкових чисел, такі як генератори на основі хеш-функцій або квантові генератори випадкових чисел [19].

Лінійно конгруентні генератори псевдовипадкових чисел (LCG) можуть мати певні переваги в контексті візуальної криптографії, де важлива обробка та захист зображень. Переваги конгурентних генераторів:

1. LCG - це дуже прості в реалізації генератори, які можуть працювати дуже швидко, що важливо для операцій над зображеннями. Вони не вимагають великих обчислювальних ресурсів.
2. Як показує назва, LCG є лінійними функціями. Це означає, що вони мають математичну простоту та передбачуваність, що може бути важливими у візуальній криптографії, де певні ефекти мають бути добре контрольованими.
3. LCG дозволяють легко відтворити послідовності чисел, що може бути важливим у візуальних криптосистемах, де користувачам можуть бути надані ключі для відтворення захищених зображень.
4. Оскільки LCG мають просту математичну структуру, їх параметри можуть бути легко контрольовані та настроєні для відповідності вимогам стійкості в конкретних візуальних криптосистемах.
5. LCG можуть бути використані для генерації шуму, який важливий для різних візуальних криптосистем, включаючи системи для приховування інформації та обробки зображень.
6. Деякі LCG можуть бути налаштовані так, щоб відповідати особливостям людського зору, що важливо в візуальній криптографії.

Зазначимо, що використання LCG у візуальній криптографії має свої обмеження, особливо в тих випадках, коли вимагається висока стійкість до криптоаналізу та велика ступінь випадковості. Для більш вимогливих застосувань візуальної криптографії, можуть бути використані більш сучасні та складні генератори ПГЧ.

Лінійно конгруентні генератори псевдовипадкових чисел (LCG) мають декілька недоліків у візуальній криптографії та інших областях:

1. LCG є відносно ненадійними для криптографічних цілей, оскільки їхні числа мають властивості, які можуть бути передбачуваними атаками, які використовують знання кількох послідовних чисел. Це зазвичай не підходить для сучасних вимог безпеки.
2. LCG генерують числа, які відповідають обмеженому діапазону та мають певні математичні залежності, що роблять їх менш випадковими, ніж може знадобитися в візуальній криптографії. Це може призвести до небажаних структур у візуальних шифрах.
3. Багато LCG мають обмежений період, після якого послідовність чисел повторюється. Це може призвести до непередбачуваних ефектів в візуальній криптографії, коли зображення або шифри повторюються.
4. Використання LCG для генерації шуму чи інших візуальних ефектів може призвести до відомих артефактів або структур, які можуть бути помічені в зображеннях, що використовують такий шум.
5. LCG, які використовуються у візуальній криптографії, можуть мати обмежену варіативність та можливість генерувати лише обмежені типи візуальних ефектів.

Враховуючи ці недоліки, важливо вибирати генератори псевдовипадкових чисел у візуальній криптографії з належною обачністю та розумінням обмежень LCG. Для більш вимогливих застосувань важливо розглядати більш сучасні та стійкі генератори ПГЧ або інші методи генерації випадкових даних.

2. РОЗРОБЛЕННЯ МЕТОДУ РОЗДІЛЕННЯ СЕКРТЕТУ

2.1 Узагальнений опис методу

Метод розподілу секрету передбачає розділення конфіденційної інформації (секрету) на декілька частин, відомих як “шматки", і розподіл цих шматків між різними особами або пристроями. Цей метод забезпечує безпеку, оскільки для відновлення секретної інформації потребується об'єднання або “збирання” достатньої кількості сегментів.

Розподіл секрету проходить в декілька етапів:

- створення секрету – потрібно обрати інформацію, картинку чи ключ який в подальшому буде використовуватись для розподілу;
- розподіл секрету – конфіденційна інформація розбивається на декілька частин;
- зберігання секрету – сформовані частини секрету роздаються окремо одна від одної та кожен учасник отримує лише одну з них;
- відновлення секрету – секрет може бути відновлений тільки тоді, коли всі частини об’єднуються.

Робота присвячена розподіленню секрету, що міститься зображеннях, тому для створення секрету буде використовуватись файл з розширенням .jpeg, .jpg. Метод захисту секрету передбачає виконання таких дій:

- вибір зображення – обирається файл який буде використаний як секрет;
- кодування секрету – зображення перетворюється в потік байтів, де кожен піксель буде у вигляді числа;
- вибір пікселів для перестановки – за допомогою генератора ПВЧ обираються порядок перестановки пікселів в частини секрету;
- перестановка – після створення інструкції перестановки формуються частини секрету;
- збереження – після формування частин зберігаються окремі зображення з закодованим вмістом.

Для того щоб зрозуміти як буде відбуватись перестановка потрібно ознайомитись з структурою файлу .jpeg (таб. 2.1). Базова структура складається з п’яти частин, однак вона може змінюватись в залежності від файлу та його стиснення.

Таблиця 2.1 – Структура файлу .jpeg

SOI (Start of Image)	Позначає початок зображення і завжди міститься на початку файлу.
APP (Application Marker)	Це секції, які містять додатковий інформаційний заголовок , який вказує на стандартний формат обміну файлами JPEG.
SOF (Start of Frame)	Містить інформацію про параметри зображення, такі як розмір, кількість компонент кольору
SOS (Start of Scan)	Вказує початок області сканування, необхідні для декодування зображення.
Data: K/M	– Коефіцієнти ДКП: Це числові значення, які представляють інтенсивність пікселів у зображенні. – Маркери кінця блоків: Вказують кінець блоків даних, таких як кінець коефіцієнтів ДКП чи кінець маркера.
EOI (End of Image)	Позначає кінець зображення і завжди міститься в кінці файлу.

Для перестановки зображення JPEG-формату використовується поле Data: K/M – інформація про його розмір, кількість пікселів та компонентів кольору.

Після того, як обрано зображення здійснюється його розподіл наприклад на 3 частини. За допомогою генератора ПВЧ обирається піксель та записується

в першу частину секрету, потім інший піксель та записується в частину секрету під номером – два. Детальна схема розподілу представлена на рисунку 2.1.

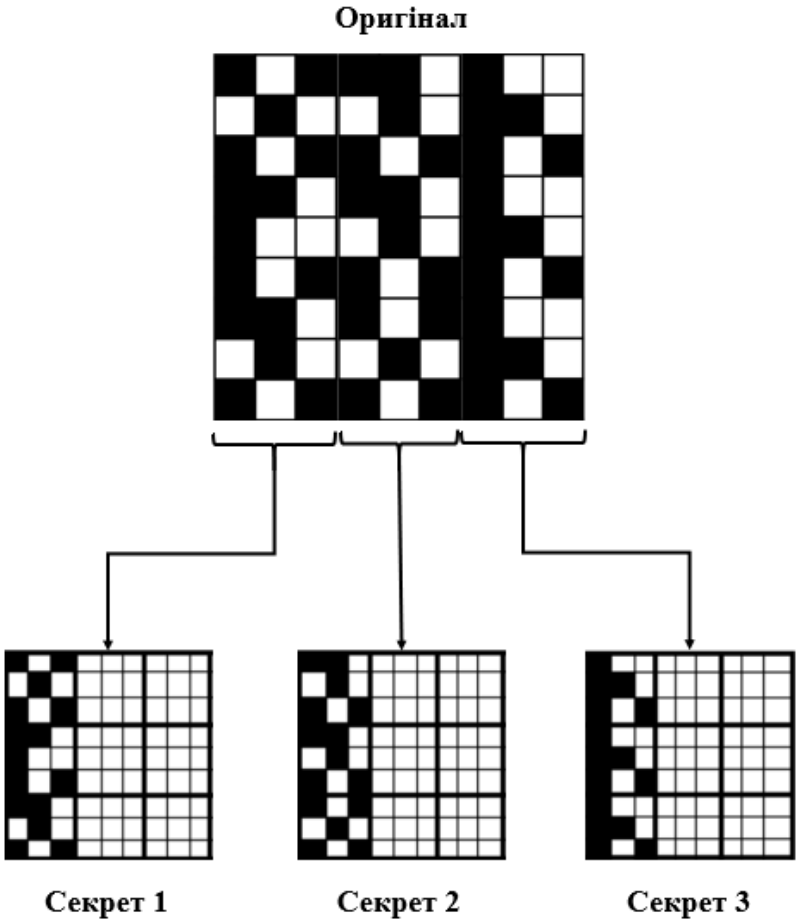


Рисунок 2.1 – Схема розподілу секрету

Виходячи з схеми після перестановок формується три файли секрети які можна розділити між трьома учасниками. Кожен секретний файл має таке ж розширення - .jpeg. Для того щоб не втратилась можливість відновити початкове зображення кожен файл секрет має таку ж роздільну здатність як оригінал, тобто зберігається кількість пікселів по висоті та ширині.

Так як повинна зберегтись можливість відновлення, розділення початкового зображення на три частини із збереженням роздільної здатності – неможливе. Це зумовлене тим, що кількість пікселів які відбираються не використовуються повторно, тому потрібно створювати так звані хибні пікселі, які будуть міститись в зображенні але не нестимуть правдивої інформації.

2.2 Генератори перестановок та псевдовипадкових послідовностей

Як сказано вище у структурі файлу містяться поля з інформацією про сигнатуру формату, розмір та позицію пікселів.

У полі DAT K/M міститься про його ширину та висоту в пікселях.

В масиві пікселів міститься вся кольорова інформація зображення у вигляді двовимірного масиву.

Кожен піксель має свою координату по висоті (x) і ширині (y).

Для цього випадковим чином вибирається піксель з початкового масиву *M* . Така процедура відповідає перестановці пікселів.

Кожен піксель *P* має три складові:

1. *R* - значення червоного кольору;
2. *G* - значення синього кольору;
3. *B* - значення зеленого кольору.

З даних про нові позиції пікселів та їх кольори, створюється три його копії.

$$S \to \{S_1, S_2, S_3\} \tag{1.3}$$

Для коректного відображення зображень, заголовки копіюються із початкового зображення і кожен піксель, кожної копії записуються одна складова кольору оригінального кольору, інші дві заповнюються значеннями, обчисленими певним чином.

З урахуванням цього, маємо такий вміст S1:

$$S_1 = \{P_{xy}\}, P_{xy} = \{R_{xy}, G_{xy}, B_{xy}\}, \tag{1.4}$$

де: $R_{xy} = R_{xy}$;

$G_{xy} = B_{xy} = L \; mod \; 256$;

$L = (i + 1)(j + 1)$.

Вибір модуля 256 здійснюється таким чином, щоб отримане число потрапляло в діапазон значень певного кольору. Для інших елементів кольору виконуються аналогічні дії з іншими компонентами.

$$S_2 = \{P_{xy}\}, \quad P_{xy} = \{R_{xy}, G_{xy}, B_{xy}\}\,, \tag{1.5}$$
$$\vdots$$
$$S_3 = \{P_{xy}\},\ P_{xy} = \{R_{xy}, G_{xy}, B_{xy}\},$$

В результаті отримуємо три окремі зображення, які можна передати учасникам секрету.

Щоб реалізувати перестановку пікселів в зображені потрібно мати генератор який буде працювати без повторень в діапазоні заданих чисел.

Для простоти представлення, візьмемо деякий порядок чисел яким потрібно змінити їх порядок. Задача стоїть так, щоб змінити їх послідовність. Числа пронумеровані та відсортовані за зростанням від 0 до N-1. Ці числа це і є представлення пікселів, тому він буде включати наступну інформацію:

$$P = \{N, F, I, X, t\} \tag{1.6}$$

F – множина функцій формування псевдовипадкових послідовностей:

$$F = \{F_1, F_2, F_3, F_4\}, \tag{1.7}$$

F_1 – функція формування підпослідовностей;

F_2 – функція формування кількості чисел q_i у підпослідовностях;

F_3 – функція формування номеру підпослідовності, для випробування;

F_4 – функція вибору числа (індексу) з підпослідовності;

I – індикатор перестановки;

X – множина правил перестановок;

t – множина перестановок у підпослідовностях.

Послідовність з чисел розбивається на x підпослідовностей, тобто:

$$N = \sum_{i=0}^{x-1} N_i \tag{1.8}$$

Послідовність – це просто набір чисел, тому вона може бути як парною так і не парною. Всі послідовності створюються функцією F1. Існує два варіанти поділу на підпослідовності: коли числа діляться без остачі тобто парна кількість, та непарна.

Перший варіант це коли виходячи із значення *x* обчислюється кількість чисел *t* у підпослідовностях.

Якщо *N* ділиться на *x* без остачі, то буде *x* підпослідовностей з кількістю чисел $t = \frac{N}{x}$. Якщо ж результат ділення має остачу то кількість буде округлюватись до більшого цілого числа.

Другий варіант передбачає завдання однакової кількості чисел у підпослідовностях *t*, яка вибирається з діапазону значень:

$$t = 2 \div \frac{N}{x} \tag{1.9}$$

Ця послідовність теж округлюється до більшого числа.

Таким чином, підпослідовності з 0-ї до (x – N)-ї складатимуться з t чисел (рис 2.2).

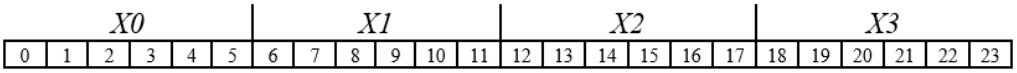


Рисунок 2.2 – Підпослідовності однакової кількості чисел *t*.

Якщо ж присутня різна кількість чисел у підпослідовностях то функція F_2 працює наступним чином:

- за допомогою ПВЧ формується кількість чисел *t*;
- початкова кількість підпослідовностей $x := 0$;
- проміжному параметру *V* присвоюється початкова кількість чисел яка пропорційна кількості *N*. При кожному формуванні *x* з *V* віднімається кількість використаних чисел доки *V* = 0. При проходженні кількість підпослідовностей збільшується на одиницю.

В результаті остання підпослідовність буде з непарною (рис 2.3)

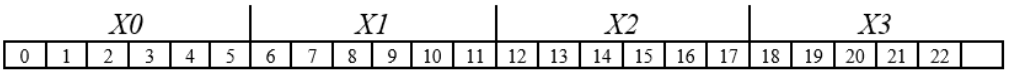


Рисунок 2.3 – Підпослідовності з різною кількістю чисел

Для реалізації перестановки потрібно вирішити проблему вибору підпослідовностей та вибору чисел з неї.

Існує декілька способів вибору підпослідовностей:

- детермінований;
- псевдовипадковий.

При детермінованому підході вибір здійснюється фіксованим заданим кроком. В ньому задається величина кроку та його заміщення. Кожна підпослідовність проходиться одна за одною.

Псевдовипадковий метод використовує генератор ПВЧ, який генерує підпослідовностям номера проходження. Для застосування вибору за допомогою ПВЧ потрібно враховувати додатковий параметр такий як індикатор перестановки. Цей параметр підраховує кількість перестановок та запобігає повторенню вибору чисел з підпослідовностей.

На початку обрахунків індикатор пропорційний кількості чисел в підпослідовностях і дорівнює:

$$I \coloneqq t \tag{1.10}$$

Після кожного кроку обрахунку в підпослідовності індикатор зменшується на одиницю. Коли індикатор зменшується до 0, то підпослідовність перестає використовуватись. За допомогою значень індикаторів формується загальний індикатор який дорівнює кількості чисел підпослідовності і обчислюється за формулою:

$$I = I_0 + I_1 + \ldots + I_{x-1}. \tag{1.11}$$

У випадку, коли індикатори всіх підпослідовностей дійшли до 0, це означає що процес перестановки завершився.

Отож, процес перестановки включає три етапи:

1) розділення послідовності на підпослідовності;

2) вибір підпослідовностей;

3) вибір чисел із підпослідовності.

Вибір підпослідовності буде здійснюватися на основі псевдовипадкої послідовності нулів і одиниць. Якщо символ «0», то вибирається підпослідовність N_0, інакше – підпослідовність N_1.

Вибір чисел з підпослідовності буде здійснюватися детермінованим способом, а, саме, збільшенням або зменшенням поточного числа на 1. Така процедура реалізується за допомогою лічильника.

Розглянемо процес формування 4 підпослідовностей псевдовипадкових чисел для перестановок.

Наприклад існує послідовність N. Для того щоб розпочати з нею роботу, послідовність потрібно розбити на підпослідовності. Тобто, з послідовності N утворюється декілька частин відповідно до поділу, в цьому випадку на 4 – N_0, N_1, N_2, N_3:

$$N_0 = \{N_0{}^{\min} \div N_0{}^{\max}\} \tag{1.12}$$
$$\vdots$$
$$N_3 = \{N_3{}^{\min} \div N_3{}^{\max}\}$$

Існує дві послідовності – 0 та 1. В залежності від послідовності буде обрано який лічильник буде використано: 00 – це Лічильник 0, 01 – це Лічильник 1, 10 – це Лічильник 2, 11 – це Лічильник 3.

При виборі лічильника буде вибиратись відповідно крок + чи – 1.

Нехай N = 24, та псевдовипадкова послідовність 010010011110001100101100001111011010001101101100 0.

Початкову послідовність ділимо на рівні 4 частини (рис. 2.4).

Лічильник 0						Лічильник 1						Лічильник 2						Лічильник 3					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Рисунок 2.4 – Вхідна послідовність генератора

Для такого розбиття маємо:

$N_0{}^{\min}$ =0, $N_0{}^{\max}$ =5, $N_1{}^{\min}$ =6, $N_1{}^{\max}$ =11,

$N_2{}^{\min}$ =12, $N_2{}^{\max}$ =17, $N_3{}^{\min}$ =18, $N_3{}^{\max}$ =23.

Через розряд секретного ключа визначаються виконувані операції відповідно до початкових станів лічильників що показано в табл. 2.2 і табл. 2.3.

Лічильники мають такі початкові значення:

Лічильник 0 = $N_0{}^{\min}$ =0; Лічильник 1 = $N_1{}^{\max}$ =11.

Лічильник 2 = N_0^{min} =12; Лічильник 3 = N_1^{max} =23.

В таблиці 2.4 наведені поточні дані лічильників, послідовність дій і псевдовипадкова послідовність чисел.

Таблиця 2.2 - Початкові стани лічильників

K1	K2	K3	K4	Ліч0	Ліч1	Ліч2	Ліч3
0	0	0	0	N_0^{min}	N_1^{min}	N_2^{min}	N_3^{min}
0	0	0	1	N_0^{min}	N_1^{min}	N_2^{min}	N_3^{max}
0	0	1	0	N_0^{min}	N_1^{min}	N_2^{max}	N_3^{min}
0	0	1	1	N_0^{min}	N_1^{min}	N_2^{max}	N_3^{max}
0	1	0	0	N_0^{min}	N_1^{max}	N_2^{min}	N_3^{min}
0	1	0	1	N_0^{min}	N_1^{max}	N_2^{min}	N_3^{max}
0	1	1	0	N_0^{min}	N_1^{max}	N_2^{max}	N_3^{min}
0	1	1	1	N_0^{min}	N_1^{max}	N_2^{max}	N_3^{max}
1	0	0	0	N_0^{max}	N_1^{min}	N_2^{min}	N_3^{min}
1	0	0	1	N_0^{max}	N_1^{min}	N_2^{min}	N_3^{max}
1	0	1	0	N_0^{max}	N_1^{min}	N_2^{max}	N_3^{min}
1	0	1	1	N_0^{max}	N_1^{min}	N_2^{max}	N_3^{max}
1	1	0	0	N_0^{max}	N_1^{max}	N_2^{min}	N_3^{min}
1	1	0	1	N_0^{max}	N_1^{max}	N_2^{min}	N_3^{max}
1	1	1	0	N_0^{max}	N_1^{max}	N_2^{max}	N_3^{min}
1	1	1	1	N_0^{max}	N_1^{max}	N_2^{max}	N_3^{max}

Таблиця 2.3 - Виконувані операції

K1	K2	K3	K4	Ліч0	Ліч1	Ліч2	Ліч3
0	0	0	0	+1	+1	+1	+1
0	0	0	1	+1	+1	+1	-1
0	0	1	0	+1	+1	-1	+1
0	0	1	1	+1	+1	-1	-1
0	1	0	0	+1	-1	+1	+1
0	1	0	1	+1	-1	+1	-1
0	1	1	0	+1	-1	-1	+1
0	1	1	1	+1	-1	-1	-1
1	0	0	0	-1	+1	+1	+1
1	0	0	1	-1	+1	+1	-1
1	0	1	0	-1	+1	-1	+1
1	0	1	1	-1	+1	-1	-1
1	1	0	0	-1	-1	+1	+1
1	1	0	1	-1	-1	+1	-1
1	1	1	0	-1	-1	-1	+1
1	1	1	1	-1	-1	-1	-1

Таблиця 2.4 – Послідовність дій

Крок	Послідовність	Ліч. 0	Ліч.1	Ліч. 2	Ліч. 3	Послідовність Р
1	01	0	11 11-1=10	12	23	11
2	00	0+1=1	10	12	23	0
3	10	1		12+1=13	23	12
4	01	1	10 10-1=9	13	23	10
5	11	1	9	13	23 23-1=22	23
6	10	1	9	13 13+1=14	22	13
7	00	1 1+1=2	9	14	22	1
8	11	2	9	14	22 22-4=21	22
9	00	2 2+1=3	9	14	21	2
10	10	3	9	14 14+1=15	21	14
11	11	3	9	15	21 21-1=20	21
12	00	3 3+1=4	9	15	20	3
13	01	4	9 9-1=8	15	20	9
14	11	4	8	15	20 20-1=19	20
15	10	4	8	15 15+1=16	19	15
16	11	4	8	16	19 19-1=18	19
17	01	4	8 8-1=7	16	18	8
18	00	4 4+1=5	7	16	18	4
19	01	5	7 7-1=6	16	18	7
20	10	5	6	16 16+1=17	18	16
21	11	5	6	17	18 stop	18
22	01	5	6 stop	17		6
23	10	5		17 stop		17
24	00	5stop				5

Необхідно мати генератор перестановок, який має вхід 0 і 1, оскільки цього вимагає задача.

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.

Кожному стану регістру, незалежно від конфігурації, можна поставити у відповідність поліном

$$a_{r-1} x^0 \oplus_{ar-2} x^1 \oplus \ldots \oplus a_0 x^{r-1} \quad (1.13)$$

Поліном, який відповідає наступному стану регістру, утворюється множенням поліному, що відповідає поточному стану, на x (так відображається зсув) та виконанням деякої додаткової операції. У конфігурації Галуа, ця додаткова операція полягає у тому, що коли у поточному стані a0 = 1, до результату добутку додається поліном.

$$x^r \oplus c_1 x^{r-1} \oplus c_2 x^{r-2} \oplus \ldots \oplus c_{r-1} x \oplus 1. \quad (1.14)$$

На рис. 2.5 графічно зображено схему роботи чотирьохрозрядного регістру.

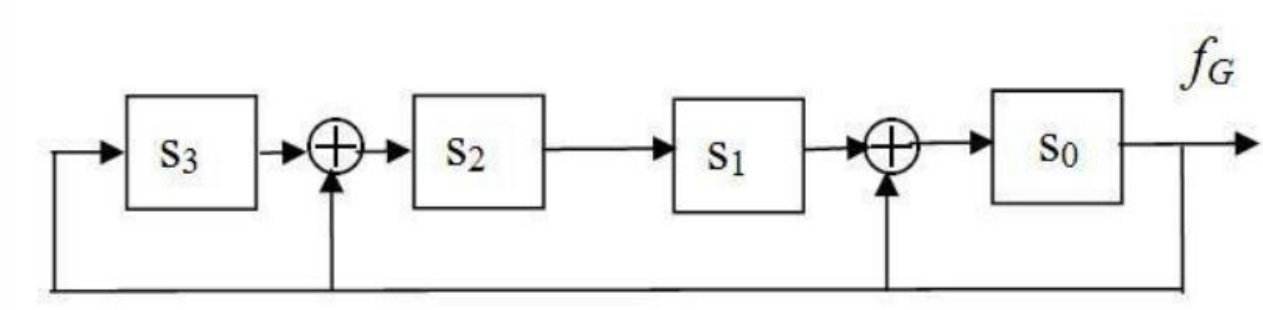


Рисунок 2.5 – Чотирьохрозрядний регістр Галуа що відповідає поліному

Запропонована методика перестановки чисел у межах усієї еталонної множини забезпечує їх псевдовипадковий порядок. Це особливість алгоритму. Використання цієї техніки підвищує криптографічну стабільність шифрування за рахунок збільшення кількості потенційних перестановок.

2.3 Метод розподілення секрету

Поставлена задача реалізується на основі розподілення секрету.

Метод розподілення секрету базується на використанні зображення для створення декількох частин секрету та їхнього подальшого розподілу між учасниками. Цей процес містить декілька пунктів:

- вхідний секрет, який потрібно розподілити, визначається і приймається для обробки;
- за допомогою генератора псевдовипадкових чисел формується послідовність, яка буде використовуватися для детерміністичного поділу зображення та створення трьох окремих частин секрету;
- зображення розбивається на пікселі за допомогою псевдовипадкової послідовності, яка визначає розташування кожного пікселя в одній з трьох окремих частин;
- для кожної частини секрету формується окреме зображення, яке містить відповідні пікселі, розташовані за допомогою псевдовипадкової послідовності.

Як описано вище зображення розбивається на пікселі за допомогою послідовності. Ця псевдовипадкова послідовність працює на основі лічильників блок-схема яких наведена на рисунку . Перед початком роботи з послідовністю зображення розбивається на потік байтів та розбивається на декілька частин в залежності від кількості частин на які потрібно поділити зображення. Частини діляться на рівні відрізки та до кожної назначається лічильник (рис 2.5).

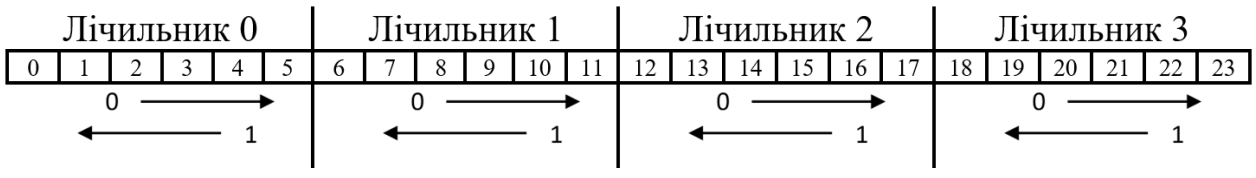


Рисунок 2.5 – Рух по лічильнику

До кожного лічильника генерується випадкове число 0 чи 1, яке буде позначати напрямок руху по лічильнику на спад чи на підйом. Ініціалізація генератора представлена на рисунку 2.6.

Після визначення напрямку лічильника ініціалізується лічильник та крок за кроком проходить по всій послідовності генеруючи результат перестановки.

Рисунок 2.6 – Ініціалізація генератора

Для коректної реалізації перестановок нам потрібен генератор. З представленої вище схеми видно як формується початковий стан регістру

зсуву. Він нам потрібен для перестановки байтів. Використаний регістр зсуву з зворотнім зв’язком використовує поліном для забезпечення зсуву. Схема роботи регістру та його кроків з поліномом представлена на рис. 2.7.

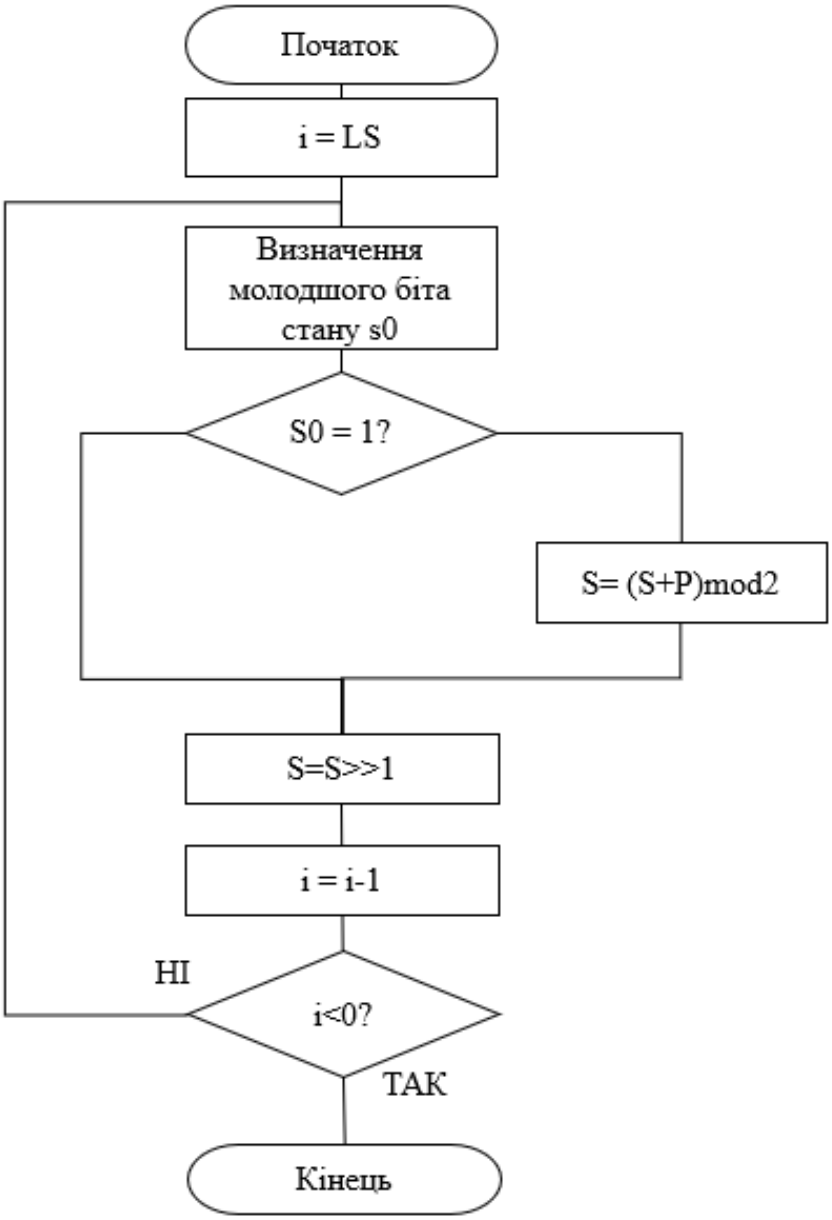


Рисунок 2.7 – Схема роботи регістру зсуву з зворотнім зв’язком

В результаті роботи всіх кроків виходить послідовність перестановки пікселів за допомогою якої можна створити частини зображення з якими надалі можна маніпулювати.

Цей метод дозволяє безпечно розділити конфіденційну інформацію та гарантує її відновлення лише при наявності всіх частин секрету.

Ввесь алгоритм розподілення секрету на основі зображення проілюстровано на рис. 2.8.

Рисунок 2.8 – Схема роботи алгоритму розділення секрету

Розглянувши концепцію розподілу секрету можна підбити підсумки:

- алгоритм забезпечує високий рівень безпеки, оскільки розподіл секрету залежить від унікальних властивостей файлу зображення;
- детерміністичний підхід до розподілу та відновлення секрету робить алгоритм надійним та передбачуваним для учасників;
- використання зображення дозволяє ефективно та швидко розділити секрет на частини, зменшуючи час та ресурси, необхідні для проведення процедури.

Запропонований метод має масштабованість, адже в залежності від кількості частин варіюється стійкість конфіденційної інформації та відновлення секрету.

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.

$$T=T_{зч}+T_G+T_{зап} \tag{2.15}$$

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):

$$T=L+L+5L=7L(у.о.) \tag{2.16}$$

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. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Обґрунтування вибору середовища програмування

Магістерська робота ставить перед собою завдання розробити програмний продукт, тому важливо обрати мову програмування та середовище розробки. У даному випадку програма буде реалізована у середовищі PyCharm Pyton 3.11.

Python, як мова програмування, також може бути ефективним вибором для розробки програмного забезпечення. Використання різноманітних бібліотек та інструментів у Python сприяє швидкій і зручній реалізації завдань. У порівнянні з Microsoft Visual C++, Python володіє великою кількістю вбудованих інструментів для візуального програмування та має зручний синтаксис.

Система розробки на мові Python, зокрема у середовищі Jupyter Notebook чи PyCharm, може бути відмінним вибором для створення як невеликих персональних програм чи утиліт, так і корпоративних систем, що взаємодіють з базами даних на різних платформах.

Під час написання програми важливим елементом може бути використання функцій, які надає Python для взаємодії з операційною системою. Це може включати використання модулів, таких як os та sys, для роботи з функціями операційної системи. Вони дозволяють спростити роботу програміста та оптимізувати використання ресурсів.

Одним з значущих плюсів Python є бібліотека Pillow, яка дозволяє полегшити роботу з зображеннями. Дана бібліотека спрощує обробку зображень та дозволяє значно скоротити вихідний код програми.

Також Python дозволяє створити ефективний та простий графічний інтерфейс для взаємодії з програмою. Для цього можна використати бібліотеку «ttk» та графічний модуль розробки інтерфейсу «Formation».

Таким чином, Python може слугувати ефективним інструментом для реалізації програми, а використання вбудованих функцій та бібліотек дозволить спростити процес програмування та роботи з операційною системою.

3.2 Вимоги до програмного засобу

У магістерській кваліфікаційній роботі передбачено розробку програми, яка реалізує розподіл секрету на основі зображень у формі Windows додатку. Програма повинна мати привабливий вигляд та використовувати стандартні елементи керування для забезпечення зручного взаємодії користувача. Для досягнення цієї мети планується використання таких ресурсів:

- Основне вікно: Воно буде використовуватися для роботи з програмою, забезпечуючи основний функціонал.
- Кнопки: Елементи керування у вигляді кнопок будуть використовуватися як основний засіб управління програмою, спрощуючи взаємодію з нею.
- Вікна повідомлень: Для виведення інформації про помилки в програмі та неправильне використання передбачено використання вікон повідомлень.
- Додаткові елементи керування: Додаткові елементи будуть додані для покращення зручності користування програмою.

У процесі розробки планується виконати такі кроки:

- дослідження генератора псевдовипадкових послідовностей;
- підбір програмних засобів;
- побудова алгоритму та схеми даних;
- програмна реалізація методу;
- розробка інтерфейсу;
- тестування та виявлення помилок;
- усунення помилок.

Процес розробки буде систематично веденням крок за кроком для досягнення визначених цілей та забезпечення успішної реалізації програмного продукту.

3.3 Розробка загальної схеми функціонування програми

Коли запускається програма з`являється головне вікно, яке дає змогу обрати та вивести на екран зображення для шифрування. На рис 3.1 представлена загальна схема функціонування програми.

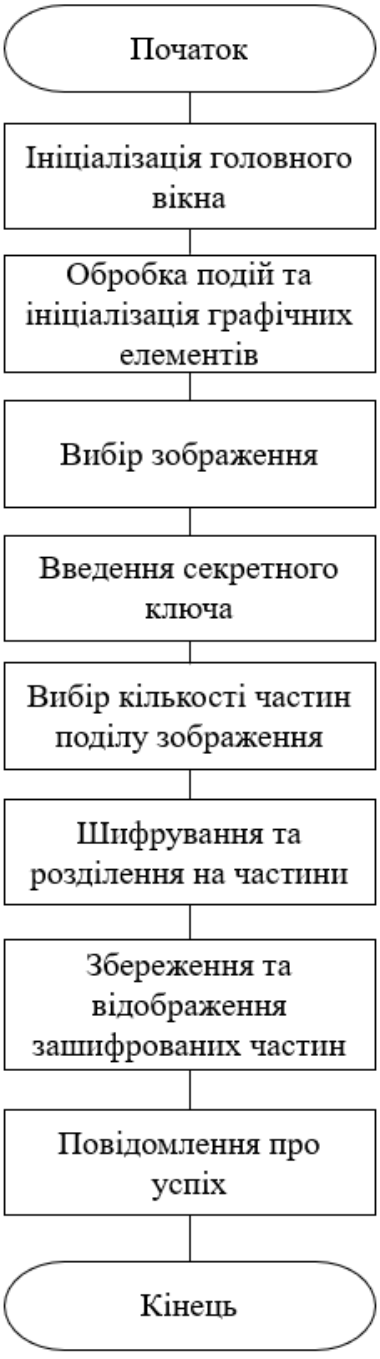


Рисунок 3.1 – Загальна схема роботи програми

Крім того, у вікні є додаткові елементи керування які дають змогу обрати на яку кількість частин буде розподілений секрет.

3.4 Вибір формату зображень

Дивлячись на те, що реалізація програмного застосунку здійснюється в операційній системі сімейства Windows, потрібно обрати такий формат який підтримує система

JPEG — це тип формату зображення, який зберігається за допомогою методу стиснення з втратами. Вихідне зображення, як результат стиснення, є компромісом між розміром пам’яті та якістю зображення. Користувачі можуть регулювати рівень стиснення, щоб досягти бажаного рівня якості, одночасно зменшуючи розмір пам’яті. Якщо до зображення застосувати стиснення 10:1, на якість зображення вплине незначно. Чим вище значення стиснення, тим більше погіршується якість зображення [20].

Формат файлу зображення JPEG був стандартизований Joint Photographic Experts Group і, отже, отримав назву JPEG. Формат був обраний для зберігання та передачі фотографічних зображень в Інтернеті. Майже всі операційні системи тепер мають засоби перегляду, які підтримують візуалізацію зображень JPEG, які також часто зберігаються з розширенням JPG. Навіть веб-браузери підтримують візуалізацію зображень JPEG. Перш ніж переходити до специфікацій формату файлу JPEG, необхідно згадати загальний процес створення JPEG [20].

Для того щоб картинка була максимально оптимізована та не займала багато місця існує сім етапів стиснення JPEG: трансформація, зменшення вибірки, організація в групи, дискретне косинусне перетворення, квантування, кодування, додавання заголовка [21].

Трансформація: кольорові зображення перетворюються з RGB на зображення яскравості/колірності (око чутливе до яскравості, а не до кольоровості, тому частина кольоровості може втратити багато даних і, таким чином, може бути сильно стиснута) [21].

Зменшення вибірки: Зменшення вибірки виконується для кольорового компонента, а не для компонента яскравості. Зменшення вибірки виконується у співвідношенні 2:1 по горизонталі та 1:1 по вертикалі (2 год. 1 В). Таким чином

зображення зменшується в розмірі, оскільки компонент «у» не торкається, немає помітної втрати якості зображення [21].

Організація в групи: пікселі кожного компонента кольору організовані в групи 8×2 пікселів, які називаються «одиницями даних», якщо кількість рядків або стовпців не кратна 8, нижній рядок і крайні праві стовпці дублюються [21].

Дискретне косинусне перетворення: Дискретне косинусне перетворення (DCT) потім застосовується до кожного блоку даних для створення карти 8×8 трансформованих компонентів. DCT передбачає певну втрату інформації через обмежену точність комп'ютерної арифметики. Це означає, що навіть без карти буде деяка втрата якості зображення, але зазвичай вона невелика [21].

Квантування: кожен із 64 перетворених компонентів у блоці даних ділиться на окреме число, яке називається його «коефіцієнтом квантування (QC)», а потім округлюється до цілого числа. Це місце, де інформація втрачається безповоротно, великий контроль якості спричиняє ще більше втрат. Загалом, більшість програм JPEG дозволяють використовувати таблиці контролю якості, рекомендовані стандартом JPEG [21].

Кодування: 64 квантованих перетворених коефіцієнта (які тепер є цілими числами) кожного блоку даних кодуються за допомогою комбінації кодування RLE і Хаффмана [21].

Додавання заголовка: останній крок додає заголовок і всі використовувані параметри JPEG і виводить результат [21].

Декодер JPEG використовує дії у зворотному порядку, щоб створити оригінальне зображення зі стисненого.

Зображення JPEG представляється як послідовність сегментів, де кожен сегмент починається з маркера. Кожен маркер починається з байта 0xFF, за яким слідує прапор маркера, що представляє тип маркера. Корисне навантаження, за яким слідує маркер, залежить від типу маркера. Поширені типи маркерів JPEG перераховані нижче в таблиці 3.1:

Таблиця 3.1 – Маркери формату JPEG

Коротке ім'я	Байти	Корисне навантаження	Ім'я
SOI	0xFF, 0xD8	none	Start of Image
S0F0	0xFF, 0xC0	variable size	Start of Frame
S0F2	0xFF, 0xC2	variable size	Start fo Frame
DHT	0xFF, 0xC4	variable size	Define Huffman Tables
DQT	0xFF, 0xDB	variable size	Define Quantization Table(s)
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval
SOS	0xFF, 0xDA	variable size	Start Of Scan
RSTn	0xFF, 0xD//n//(//n//#0..7)	none	Restart
APPn	0xFF, 0xE//n//	variable size	Application specific
COM	0xFF, 0xFE	variable size	Comment
EOI	0xFF, 0xD9	none	End Of Image

У ентропійно-кодованих даних після будь-якого байта 0xFF кодер вставляє байт 0x00 перед наступним байтом, щоб не було маркера там, де він не призначений, запобігаючи помилкам кадрування. Декодери повинні пропускати цей байт 0x00. Ця техніка, що називається заповненням байтами (див. розділ F.1.2.3 специфікації JPEG), застосовується лише до ентропійно-кодованих даних, а не до даних корисного навантаження маркера. Зауважте, однак, що ентропійно-кодовані дані мають кілька власних маркерів; зокрема маркери скидання (від 0xD0 до 0xD7), які використовуються для ізоляції незалежних фрагментів ентропійно-кодованих даних для можливості паралельного декодування, і кодери можуть вільно вставляти ці маркери скидання через регулярні проміжки часу (хоча не всі кодери роблять це) [22].

Файл **PNG** (Portable Network Graphics) — це формат растрового зображення, який використовує стиснення без втрат. Цей формат файлу було створено як заміну формату Graphics Interchange Format (GIF) і не має обмежень щодо авторських прав. Однак формат файлу PNG не підтримує

анімацію. Формат файлу PNG підтримує стиснення зображень без втрат, що робить його популярним серед користувачів. З плином часу PNG перетворився на один із широко використовуваних форматів файлів зображень [23].

Основною причиною створення формату файлу PNG був запатентований алгоритм стиснення Лемпеля-Зіва-Велча, який використовувався у форматі файлу GIF. Це разом з іншими обмеженнями GIF створило необхідність заміни формату файлу GIF [23].

Формат файлу PNG був розроблений як простий і портативний, юридично необтяжений, взаємозамінний, гнучкий і надійний. У наведеній нижче таблиці наведено функції GIF, успадковані форматом файлу PNG (таб. 3.2).

Таблиця 3.2 – Успадковані функції PNG.

Особливість	GIF	PNG
Індексні кольорові зображення до 256 кольорів	Так	Так
Підтримка потокової передачі	Так	Так
Прозорість	Так	Так
Допоміжні відомості	Так	Так
Незалежність від апаратного забезпечення та платформи	Так	Так
Ефективний	Так	Так
Зображення Truecolor до 48 біт на піксель	Немає	Так
Зображення у відтінках сірого до 16 біт на піксель	Немає	Так
Повний альфа-канал (загальні маски прозорості)	Немає	Так
Інформація про гамму зображення	Немає	Так
Надійність	Немає	Так
Швидша початкова презентація	Немає	Так

Майже всі операційні системи підтримують відкриття файлів PNG. Наприклад, засіб перегляду Microsoft Windows має можливість відкривати файли PNG, оскільки ОС за замовчуванням підтримує підтримку, доступну як частину встановлення [23].

Перші вісім байтів файлу PNG завжди містять такі (десяткові) значення:

{{{137 80 78 71 13 10 26 10 }}}

Цей підпис вказує на те, що залишок файлу містить єдине зображення PNG, яке складається з послідовності фрагментів, починаючи з фрагмента IHDR і закінчуючи фрагментом IEND [24].

Кожен чанк складається з чотирьох частин:

Довжина: 4-байтове ціле число без знаку, що вказує кількість байтів у полі даних блоку. Довжина враховує лише поле даних, а не його самого, код типу послідовності або CRC. Нуль є дійсною довжиною [24].

Тип блоку: 4-байтовий код типу блоку. Для зручності в описі та дослідженні файлів PNG коди типів обмежені складатися з великих і малих літер ASCII (AZ і az, або 65-90 і 97-122 десяткові). Однак кодери та декодери повинні розглядати коди як фіксовані двійкові значення, а не рядки символів. Наприклад, було б неправильно представляти код типу IDAT за допомогою еквівалентів цих букв EBCDIC. Додаткові угоди про іменування типів блоків обговорюються в наступному розділі [24].

Дані блоку: Байти даних, що відповідають типу блоку, якщо такі є. Це поле може мати нульову довжину [24].

CRC: 4-байтовий CRC (перевірка циклічної надлишковості), обчислений на основі попередніх байтів у послідовності, включаючи код типу послідовності та поля даних послідовності, але не включаючи поле довжини. CRC присутній завжди, навіть для блоків, які не містять даних [24].

Довжина даних блоку може становити будь-яку кількість байтів до максимальної; отже, розробники не можуть припускати, що фрагменти вирівняні за будь-якими межами, більшими за байти [25].

Блоки можуть з’являтися в будь-якому порядку, відповідно до обмежень, накладених на кожен тип фрагментів. (Одне важливе обмеження полягає в тому, що IHDR має з’являтися першим, а IEND — останнім; таким чином фрагмент

IEND служить маркером кінця файлу.) Можуть з’являтися кілька фрагментів одного типу, але лише якщо це спеціально дозволено для цього типу [25].

Типи фрагментів класифікуються на критичні та допоміжні блоки на основі 4-байтового чутливого до регістру значення ASCII, призначеного типу блоку. Усі реалізації повинні розуміти та успішно відтворювати стандартні критичні блоки. Дійсне зображення PNG має містити блок IHDR, один або кілька блоків IDAT і фрагмент IEND [25].

Метод стиснення PNG 0 (єдиний метод стиснення, визначений на даний момент для PNG) визначає стиснення за допомогою ковзного вікна розміром не більше 32768 байт. Стиснення Deflate — це похідна версія LZ77, яка використовується в zip, gzip, pkzip і пов’язаних програмах. Стиснуті дані в потокі даних zlib зберігаються у вигляді серії блоків, кожен з яких може представляти необроблені (нестиснені) дані, стиснуті за допомогою LZ77 дані, закодовані фіксованими кодами Хаффмана, або стиснуті за допомогою LZ77 дані, закодовані за допомогою спеціальних кодів Хаффмана. Біт маркера в останньому блоці ідентифікує його як останній блок, що дозволяє декодеру розпізнати кінець стисненого потоку даних [26].

Фільтри попереднього стиснення застосовуються для підготовки даних зображення до оптимального стиснення. Метод фільтрації PNG (таб. 3.3) визначає п'ять основних типів фільтрів:

- none;
- sub;
- up;
- average;
- paeth.

Алгоритми фільтрації застосовуються до байтів, а не до пікселів, незалежно від бітової глибини кольору зображення. Алгоритми фільтрації працюють з послідовністю байтів, утвореною скан-рядком. Якщо зображення містить альфа-канал, альфа-дані фільтруються, як і дані зображення [26].

Таблиця 3.3 – Методи фільтрації PNG.

Тип фільтра	Ім'я	Прогнозована вартість
0	None	Рядок сканування передається без змін
1	Sub	Передає різницю між кожним байтом і значенням відповідного байта попереднього пікселя.
2	Up	Фільтр Up() схожий на фільтр Sub(), за винятком того, що піксель безпосередньо над поточним пікселем, а не ліворуч від нього, використовується як предиктор.
3	Average	Фільтр Average() використовує середнє значення двох сусідніх пікселів (ліворуч і вище), щоб передбачити значення пікселя.
4	Paeth	Фільтр Paeth() обчислює просту лінійну функцію трьох сусідніх пікселів, а потім вибирає як предиктор сусідній піксель, найближчий до обчисленого значення.

Алгоритми фільтрації застосовуються до байтів, а не до пікселів, незалежно від бітової глибини кольору зображення. Алгоритми фільтрації працюють з послідовністю байтів, утвореною скан-рядком. Якщо зображення містить альфа-канал, альфа-дані фільтруються, як і дані зображення [26].

Коли зображення є черезрядковим, кожен прохід шаблону черезрядкової розгортки розглядається як незалежне зображення з метою фільтрації. Фільтри працюють на послідовностях байтів, утворених пікселями, фактично переданими під час проходу, а «попередній рядок сканування» — це той, який раніше був переданий у тому самому проході, а не сусідній у повному зображенні. Зауважте, що допоміжне зображення, яке передається за один прохід, завжди прямокутне, але має меншу ширину та/або висоту, ніж повне зображення. Фільтрування не застосовується, якщо це допоміжне зображення порожнє [26].

3.5 Програмна реалізація розподілу секрету

Для зручності розробки програмного застосунку прийнято рішення розділити код на дві частини. Перша частина це реалізація модуля шифрування, а друга графічний інтерфейс який викликає засоби модуля шифрування та розширює його функціонал.

Модуль шифрування виглядає наступним чином.

Для роботи з зображеннями та файлами необхідно встановити додаткову бібліотек Pillow. Ця бібліотека дозволяє нам завантажувати, обробляти та зберігати зображення.

Щоб розпочати роботу необхідно завантажити зображення з вказаного шляху та описати функцію збереження. Для цього використовуємо функції:

```
def load_image(file_path):
    return Image.open(file_path)
def save_image(image, file_path):
    image.save(file_path)
```

Далі необхідно реалізувати лічильник, який використовується для генерації псевдовипадкової послідовності чисел. Ця послідовність використовується для генерації перестановок та для операції XOR при шифруванні. Конструктор, приймає параметр seed для ініціалізації початкового стану лічильника. Також потрібно ініціалізувати функцію яка буде генерувати наступне число в послідовності та оновлювати стан лічильника.

```
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
```

Так як за алгоритмом зображення розбивається на потік пікселів потрібно реалізувати перестановку заданої довжини потоку та параметр для визначення руху перестановки:

```
def generate_permutation_sequence(length, counter, forward):
    sequence = list(range(length))
    if not forward:
        sequence.reverse()
```

```
        return sequence
```

Наступним кроком є шифрування заданої послідовності за допомогою операції XOR та перестановки:

```
    def encrypt_subsequence(subsequence, counter, permutation_sequence):
        encrypted_subsequence = []
        for i, pixel in enumerate(subsequence):
            encrypted_pixel = pixel ^ counter.next()
            encrypted_subsequence.append(encrypted_pixel)
        encrypted_subsequence = [encrypted_subsequence[i] for i in
permutation_sequence]
        return encrypted_subsequence
```

На основі описаних функцій формується головна функція для шифрування зображення. Вона розділяє зображення на кілька частин та застосовує шифрування до кожної частини незалежно.

Спочатку для обробки потрібно конвертувати зображення у масив. Це можна зробити задопомогою пареметру:

```
    pixels = np.array(image)
    height, width, channels = pixels.shape
```

Далі необхідно обчислити кількість підпослідовностей на які розбита послідовність:

```
    subsequences_per_part = width
```

Потрібно створити лічильник для генерації унікальних чисел для кожної частини та визначити напрямок перестановки на основі генерованого числа. Для кожної частини зображення викликається encrypt_subsequence, яка зашифровує кожну підпослідовність.

```
    def encrypt_image(image, key, num_counters):
        pixels = np.array(image)
        height, width, channels = pixels.shape
        pixels_per_counter = width // num_counters
        encrypted_image_parts = []
        for i in range(num_counters):
            start_col = i * pixels_per_counter
            end_col = (i + 1) * pixels_per_counter
            direction_forward = int(key[i*8:(i+1)*8], 2) % 2 == 0
```

```
            counter=SimpleCounter(seed=int(key[i*8:(i+1)*8],2), key=key)
            forward = direction_forward        permutation_sequence =
generate_permutation_sequence(pixels_per_counter, counter, forward)
            for j in range(height):
                subsequence = pixels[j, start_col:end_col, :]
                encrypted_subsequence = encrypt_subsequence
                pixels[j,start_col:end_col,:]= np.array
            encrypted_image_parts.append(pixels[:,start_col:end_col, :])
        return encrypted_image_parts
```

Результати шифрування додаються до списку encrypted_image_parts та повертається список шифрованих частин зображення.

```
encrypted_image_parts.append(pixels[:,start_col:end_col,:])
```

3.6 Реалізація інтерфейсу

Щоб зручно та інтуєтивно користуватись програмою потрібно розробити простий та інформативний інтерфейс. Для того, щоб досягти цього потрібно дотримуватись таких елементів:

- головне вікно;
- кнопки керування програмою;
- вікна попередження та успішності;
- можливість роботи з файлами.

Вигляд головного вікна представлено на рисунку 3.2

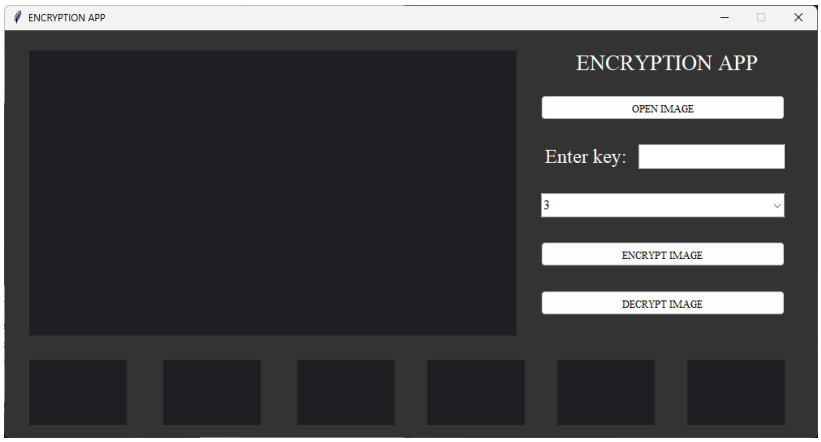


Рисунок 3.2 – Вигляд головного вікна програми

Ввесь функціонал керування програми були створені за допомогою бібліотеки «TK» в середовищі розробки PyCharm. Для реалізації керування

створено три кнопки та випадаюче для вибору кількості частин розподілення секрету (рис.3.3).

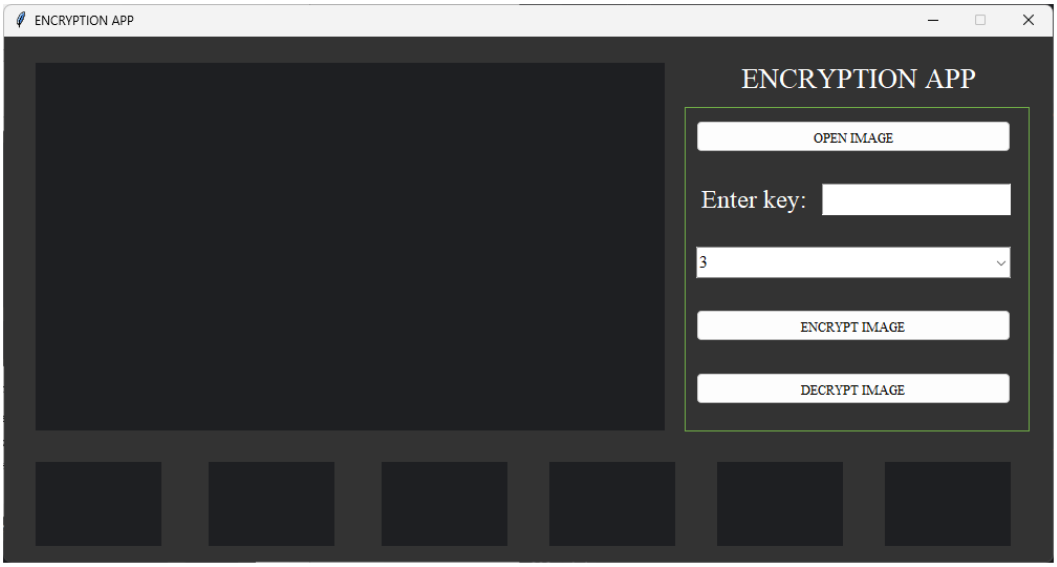


Рисунок 3.3 – Вигляд меню головного вікна

Для вибору зображення яке потрібно зашифрувати натисненням кнопки викликається стандартна панель провідника з заздалегідь налаштованими розширеннями (рис.3.4).

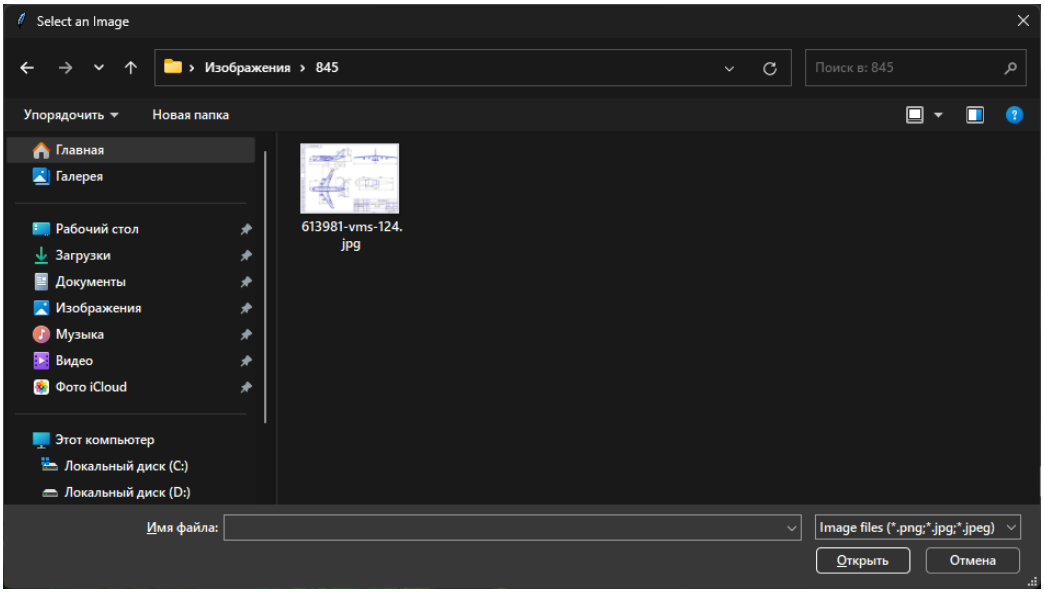


Рисунок 3.4 – Вигляд панелі відкриття файлів

Таким чином створено зручний та інтуїтивно зрозумілий інтерфейс, який реалізовує розподіл секретних зображень а також їх об’єднання.

3.7 Результати роботи програми

Розпочати перевірку слід з відкриття файлу зображення, для цього потрібно натиснути кнопку «OPEN IMAGE» та обрати файл (рис.3.5).

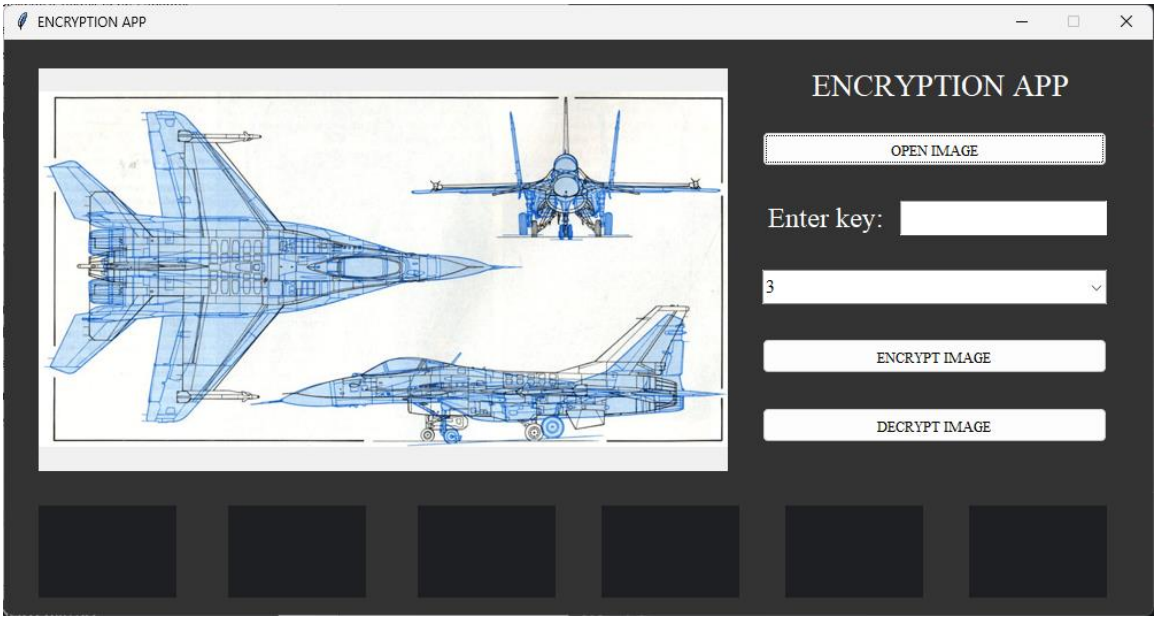


Рисунок 3.5 – Вигляд обраного зображення

Наступним кроком буде розділення та шифрування зображення, для цього потрібно натиснути «ENCRYPT IMAGE» (рис.3.7).

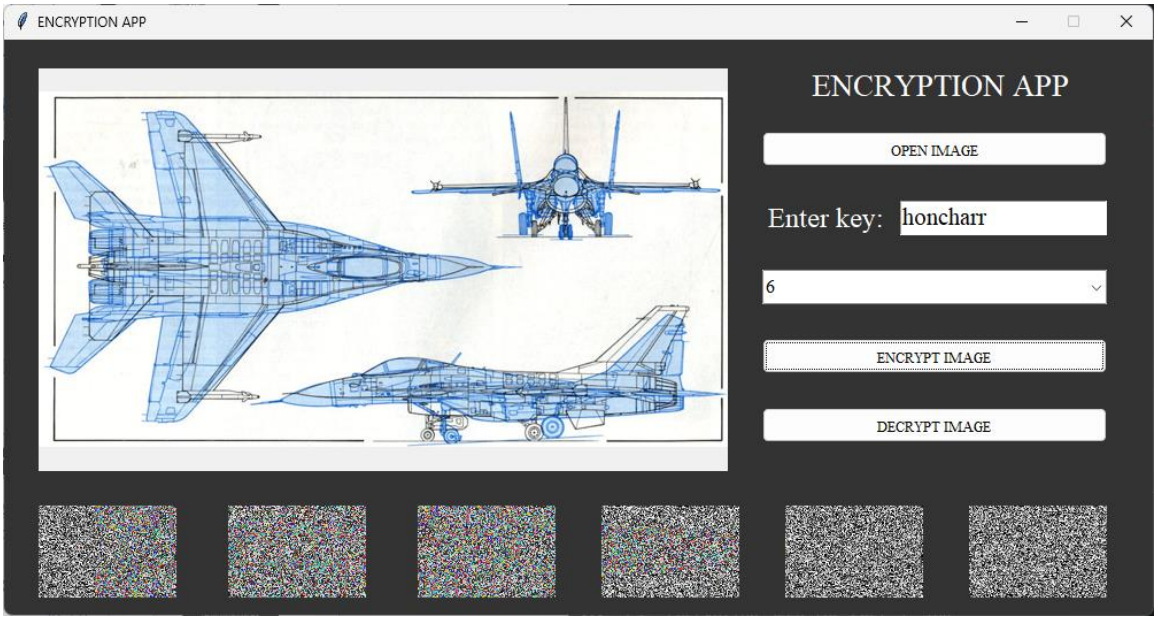


Рисунок 3.6 – Вигляд обраного зображення та зашифрованих частин

Після завершення роботи програми у вікні програми з’являються зашифровані частини а в директорії зберігаються зашифровані файли (рис.3.7).

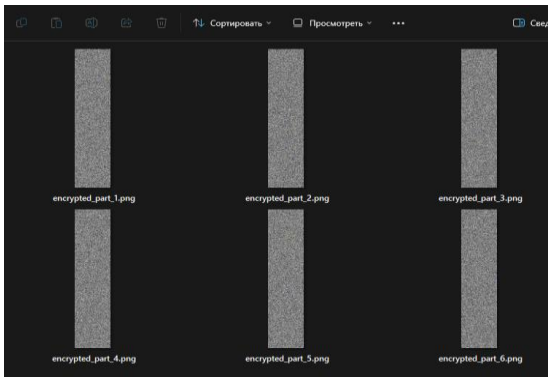


Рисунок 3.7 – Вигляд розподілених зображень

Для перевірки коректності роботи поділу на іншу кількість частин протестуємо на іншому зображенні (рис.3.8).

Рисунок 3.8 – Початковий вигляд зображення

Після розподілення було отримано наступний вміст (рис.3.10).

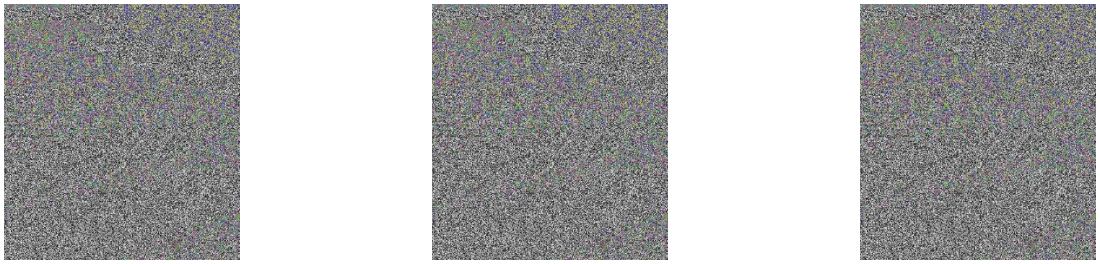


Рисунок 3.9 – Вигляд розподілених частин зображення

В результаті вийшло три зашифрованих чистини, вміст яких розпізнати неможливо без розшифрування.

Результатом тестування доведено, що додаток працює без помилок, ввесь функціонал працює коректно та шифрування здійснюється вдало.

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 1. Підсистема зашифрування 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

Метою проведення комерційного і технологічного аудиту дослідження за темою "Система шифрування зображень Частина 1. Підсистема зашифрування зображень" є пришвидшення процесу зашифрування зображення

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.

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 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 динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	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 наявних фахівців	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 використовуються 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, Дудатьєв Андрій Веніамінович, к. т. н., доцент.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	Кондратенко Н. Р.	Дудатьєв А. В.	Лужецький В. А.
	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
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..

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 1. Підсистема зашифрування 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 1. Підсистема зашифрування 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:

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

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – 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:

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

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

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

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

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-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.

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

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

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}} \ , \qquad (4.4)$$

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

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, 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 1. Підсистема зашифрування зображень", під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.5)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 15000 \cdot 5 / 21 = 3409 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
1. Керівник проекту	15000	681,8	5	3409
2. Розробник №1	12000	545,5	35	19091
3. Розробник №2	12000	545,5	40	21818
4. Тестувальник	14000	636,4	40	25455
Всього				69773

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система шифрування зображень Частина 1. Підсистема зашифрування зображень» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (4.6)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (4.7)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6500 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б)

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1 \cdot 1{,}65 / (21 \cdot 8) = 65{,}8$ грн.
$З_{р1} = 65{,}8 \cdot 1 = 65{,}8$ грн.

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
1.Підготовчі	5	1	65,8	329,0
2.Монтажні	4	3	88,8	355,3

3.Складальні	15	5	111,9	1678,0
4.Налагоджувальні	6	2	72,4	434,3
5.Випробувальні	4	4	59,8	239,3
Всього				3035,9

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.8)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (69773+3035,9) · 11 / 100% = 8008,95 грн.

4.4 Розрахунок економічної ефективності

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 1. Підсистема зашифрування зображень» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – 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 $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою :

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.9)$$

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

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

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

Збільшення чистого прибутку 1-го року:

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

Збільшення чистого прибутку 2-го року:

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

Збільшення чистого прибутку 3-го року:

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

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.10)$$

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, τ =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 PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=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 $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.12)$$

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 $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.13)$$

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.

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

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =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 $E_в$, 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 $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.15}$$

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.

ВИСНОВКИ

Результати цієї магістерської роботи свідчать про значущий внесок у сферу захисту інформації, зокрема в контексті розподілу секрету при обробці та збереженні зображень. Використання методів криптографії, особливо в контексті візуальної криптографії, виявляється ефективним та сучасним механізмом захисту конфіденційної інформації в умовах швидкого розвитку сучасного інформаційного середовища.

Аналіз літературних джерел, проведений у першій частині роботи, пройняв усі аспекти галузі розподілу секрету та візуальної криптографії. Цей глибокий огляд літератури не лише дозволив зрозуміти теоретичні засади, але й виявив широкий спектр практичних реалізацій. Це важливо, оскільки реальна застосовність та ефективність методів криптографії визначаються не лише абстрактними концепціями, але і їхніми практичними можливостями.

Розроблений метод розподілу секрету, що ґрунтується на генераторах перестановок та псевдовипадкових послідовностях, представляє собою перспективний напрямок для створення ефективної та надійної системи захисту. Використання цих елементів дозволяє забезпечити не тільки надійний розподіл конфіденційної інформації, але й ефективний захист від потенційних атак чи спроб несанкціонованого доступу. Такий підхід відкриває нові можливості для розширення застосування криптографії в контексті обробки та збереження зображень.

Вибір генераторів перестановок та псевдовипадкових послідовностей в якості основи для розподілу секрету виявився доречним, оскільки ці елементи забезпечують необхідний ступінь випадковості та варіабельності, що є критичним для стійкості криптографічних методів.

Програмний засіб, розроблений на основі цього методу, вражає своєю структурованістю та ефективністю. Обґрунтування вибору середовища програмування та визначення вимог до програмного засобу відображають

високий науковий рівень підходу до розробки, що гарантує не лише якість реалізації, але й високу швидкість та зручність користування.

Загальна схема функціонування програми ретельно визначає процеси, які відбуваються в системі, що є ключовим для розуміння її принципів роботи. Вибір формату зображень та реалізація інтерфейсу демонструють практичний підхід до вирішення завдань збереження конфіденційності та зручного використання програми.

Завершення роботи переповнене висновками, що наголошують на важливості внеску цієї роботи в сферу криптографії та кібербезпеки. Розроблений метод та програмний засіб можуть знайти застосування в різних галузях, де конфіденційність та безпека інформації визначають подальший успіх. Усе це свідчить про неабияке значення та актуальність проведених досліджень у вирішенні сучасних проблем інформаційної безпеки.

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. Droste, S. 1996. New Results on Visual Cryptography. Advances in Cryptology-CRYPTO `96, Lecture Notes in Computer Science. – 401-415 с.
2. Prisco, R. and Santis, A. 2006. Cheating Immune Threshold Visual Secret Sharing. – 216-228 с.
3. Blakely, G. R. 1979. Safeguarding Cryptographic Keys. Proceedings of the National Computer Conference, American Federation of Information Processing Societies Proceedings. – 313-317 с.
4. Kak, S. 1982. On Asymmetric Secret Sharing. LSU ECE Technical Report. May.
5. Alon, N. and Spencer, J. 1992.
6. Parakh, A. and Kak, S. 2010. A Tree Based Recursive Information Hiding Scheme. To appear in proceedings of IEEE ICC 2010.
7. Parakh, A. and Kak, S. 2009. Recursive Secret Sharing for Distributed Storage and Information Hiding. – 1-3 с.
8. Parakh, A. and Kak, S. 2009. Space Efficient Secret Sharing: A recursive approach. Cryptologye Print Archive, Report 2009/365.
9. Innes Muecke. 1999. Greyscale and Colour Visual Cryptography, Thesis of degree of Master of Computer Science, Dalhouse University – Daltech.
10. Menezes, A. J., Oorschot, P. C., & Vanstone, S. A. (1996). "Handbook of Applied Cryptography." CRC Press.
11. L'Ecuyer, P. (1999). "Good Parameter Sets for Combined Multiple Recursive Random Number Generators." Operations Research. – 199-211 с.
12. Knuth, D. E. (1997). "The Art of Computer Programming, Volume 2: Seminumerical Algorithms." Addison-Wesley.
13. L'Ecuyer, P. (1999). "Recursive Random Number Generators." Operations Research. – 159-164 с.
14. Schindler, W., Keller, N., & Barwood, M. (2007). "Random Numbers in Cryptography." – 128-141 с.

15. Schindler, W., Keller, N., & Barwood, M. (2007). "Random Numbers in Cryptography." Cryptographic Hardware and Embedded Systems. – 128-141 с.
16. Fumy, W. (2000). "Cryptographically Secure Pseudo-Random Number Generators." – 97-113 с.
17. Aumasson, J. P., & Vaudenay, S. (2003). "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." Selected Areas in Cryptography. – 31-48 с.
18. Shaltiel, R. (2002). "Recent Developments in Explicit Constructions of Extractors." – 719-728 с.
19. Rukhin, A., et al. (2010). "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications." National Institute of Standards and Technology (NIST).
20. 0xwan. PNG structure for beginner. Medium. URL: https://medium.com/@0xwan/png-structure-for-beginner-8363ce2a9f73 (date of access: 12.12.2023).
21. JPEG - Image File Format. File Format Docs. URL: https://docs.fileformat.com/image/jpeg/ (date of access: 12.12.2023).
22. W3C. URL: https://www.w3.org/Graphics/JPEG/jfif3.pdf (дата звернення: 12.12.2023).
23. JPG Signature Format: Documentation & Recovery Example. Find and Restore Lost Files: Undelete deleted files and recover damaged disks. URL: https://www.file-recovery.com/jpg-signature-format.htm (date of access: 12.12.2023).
24. PNG File Format - Raster Image File. File Format Docs. URL: https://docs.fileformat.com/image/png/#png-file-structure (date of access: 12.12.2023).
25. PNG files structure – iTecNote. iTecNote – Provides Technology Solution. URL: https://itecnote.com/tecnote/png-files-structure/ (date of access: 12.12.2023).

26. PNG: Summary from the Encyclopedia of Graphics File Formats. FileFormat.Info · The Digital Rosetta Stone. URL: https://www.fileformat.info/format/png/egff.htm (date of access: 12.12.2023).

ДОДАТКИ

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ
МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Система шифрування зображень. Частина 1. Підсистема зашифрування зображень

Автор роботи: Гончар Олександр Ігорович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 93,78 %. Схожість – 6,22 %.

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 _____ (підпис) Валентина КАПЛУН

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

Автор роботи _____ (підпис) Олександр ГОНЧАР

Керівник роботи _____ (підпис) Володимир ЛУЖЕЦЬКИЙ

Додаток Б

Лістинг 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 encrypt_subsequence(subsequence, counter, permutation_sequence):
   # print(f"Original Pixels (Encryption): {subsequence}")
    encrypted_subsequence = []
    for i, pixel in enumerate(subsequence):
```

```
            encrypted_pixel = pixel ^ counter.next()
            encrypted_subsequence.append(encrypted_pixel)

        encrypted_subsequence = [encrypted_subsequence[i] for i in
permutation_sequence]

        return encrypted_subsequence

    def encrypt_image(image, key, num_counters):
        pixels = np.array(image)
        height, width, channels = pixels.shape
        pixels_per_counter = width // num_counters
        encrypted_image_parts = []
        for i in range(num_counters):
            start_col = i * pixels_per_counter
            end_col = (i + 1) * pixels_per_counter
            direction_forward = int(key[i*8:(i+1)*8], 2) % 2 == 0
            counter = SimpleCounter(seed=int(key[i*8:(i+1)*8], 2),
key=key)
            print(f"Encryption Seed for Part {i + 1}: {counter.state}")
            forward = direction_forward
            permutation_sequence =
generate_permutation_sequence(pixels_per_counter, counter, forward)
            for j in range(height):
                subsequence = pixels[j, start_col:end_col, :]
                encrypted_subsequence = encrypt_subsequence(subsequence,
counter, permutation_sequence)
                pixels[j, start_col:end_col, :] =
np.array(encrypted_subsequence)
            encrypted_image_parts.append(pixels[:, start_col:end_col,
:])
        return encrypted_image_parts


    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("ENCRYPTION APP")
            root.configure(bg='#333333')
            width = 1000
            height = 500
            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.jpg",
```

```
                "C:\\Encrypt\\encrypted_part_2.jpg",
                "C:\\Encrypt\\encrypted_part_3.jpg",
                "C:\\Encrypt\\encrypted_part_4.jpg",
                "C:\\Encrypt\\encrypted_part_5.jpg",
                "C:\\Encrypt\\encrypted_part_6.jpg"
            ]
            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)

            # Змінено порядок використання ключа при дешифрації
            decrypted_subsequence = [decrypted_subsequence[i] for i in
permutation_sequence]

            return decrypted_subsequence

        def create_widgets(self):
            # Завантаження зображення для кнопок
            button_image = Image.open("img_1.png")
            button_image = ImageTk.PhotoImage(button_image)

            # Створення стилю ttk
            style = ttk.Style()
            style.configure("TButton", font=('Times', 10), padding=5,
relief="flat", background="#333333",
                            foreground="black")
            style.configure("TCheckbutton", background="#333333",
foreground="white")

            # Кнопка для вибору зображення
```

```
            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")

            # Встановлення фонового зображення на кнопці
            select_button.config(image=button_image, compound="center")

            # Кнопка для шифрування зображення
            encrypt_button    =    ttk.Button(command=self.encrypt_image,
style="TButton")
            encrypt_button.place(x=660, y=260, width=300, height=30)
            encrypt_button.config(text="ENCRYPT IMAGE")

            # Встановлення фонового зображення на кнопці
            encrypt_button.config(image=button_image, compound="center")
            # Кнопка для дешифрації зображення
            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")

            # Встановлення фонового зображення на кнопці
            decrypt_button.config(image=button_image, compound="center")

            # Label для назви додатку
            app_label = tk.Label(text="ENCRYPTION 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)

            # Поле для введення тексту (ключа)
            self.key_entry = tk.Entry(root, font=('Times', 16))
            self.key_entry.place(x=780, y=140, width=180, height=30)
```

```
            # Випадаюче меню
            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)

            # Центральний фрейм з вибраним зображенням
            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)

            # Створення фреймів для шифрованих зображень
            self.create_encrypted_image_frames()

            initial_image_path = "img.png"
            self.load_and_display_initial_image(initial_image_path)
            self.image_path = None

        def create_encrypted_image_frames(self):
            encrypted_image_frame_positions = [
                (90, 445),
                (255, 445),
                (420, 445),
                (580, 445),
                (740, 445),
                (900, 445)
            ]

            self.encrypted_image_frames  =  [ttk.Frame(root,  width=120,
height=80) for _ in range(6)]
            self.encrypted_image_labels  =  [tk.Label(frame,  image=None)
for frame in self.encrypted_image_frames]
```

```
            for frame, (x, y), label in zip(self.encrypted_image_frames,
encrypted_image_frame_positions,

self.encrypted_image_labels):
                frame.place(x=x, y=y, anchor='center')
                label.place(relwidth=1, relheight=1)

        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

            # Вставити початкове зображення в усі зашифровані фрейми
            for i, frame in enumerate(self.encrypted_image_frames):
                label_encrypted_image = frame.winfo_children()[0]
                label_encrypted_image.configure(image=tk_image)
                label_encrypted_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)

            # Встановлюємо обране зображення в центральний фрейм
            self.label_image.configure(image=tk_image)
            self.label_image.image = tk_image

            # Оновлюємо зображення на всіх зашифрованих фреймах
            for i, frame in enumerate(self.encrypted_image_frames):
                encrypted_tk_image = self.encrypted_tk_images[i]
                if encrypted_tk_image:
                    label_encrypted_image = frame.winfo_children()[0]

label_encrypted_image.configure(image=encrypted_tk_image)
                    label_encrypted_image.image = encrypted_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()  # Додано виклик нової
функції

    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')

            # Якщо бінарний код непарний, здійснюємо зсув вліво та
додаємо 0 в крайню позицію
            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 encrypt_image(self):
            try:
                image = load_image(self.image_path)
            except AttributeError:
                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

            # Збереження оригінального зображення у байтовому форматі
            original_image_bytes = image.tobytes()
```

```
            self.encrypted_image_parts = encrypt_image(image, key,
num_counters)

            output_folder = "C:/Encrypt/"
            os.makedirs(output_folder, exist_ok=True)

            # Збереження зашифрованих частин
            for i, (part, label) in
enumerate(zip(self.encrypted_image_parts, self.encrypted_image_labels)):
                output_path = os.path.join(output_folder,
f"encrypted_part_{i + 1}.png")
                save_image(Image.fromarray(part.astype(np.uint8)),
output_path)

                # Оновлення зашифрованих зображень у фреймі
                encrypted_tk_image =
ImageTk.PhotoImage(Image.open(output_path))
                label.configure(image=encrypted_tk_image)
                label.image = encrypted_tk_image

            # Збереження оригінального зображення
            original_image_path = os.path.join(output_folder,
"original_image.png")
            save_image(Image.frombytes(image.mode, image.size,
original_image_bytes), original_image_path)

            messagebox.showinfo("Success", f"Encrypted image saved to
{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)

            messagebox.showinfo("Success", f"Combined decrypted image
saved to {output_folder}")

        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)

                # Додайте зворотнє відображення перестановки, якщо
direction_forward не відповідає дійсності
                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)

            self.save_decrypted_image(decrypted_image_parts)

        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

            encrypted_image_parts = [np.array(Image.open(file_path)) for
file_path in file_paths]

            #      Зміна      порядку      частини      зображення      в
self.encrypted_image_parts
            self.encrypted_image_parts = encrypted_image_parts
```

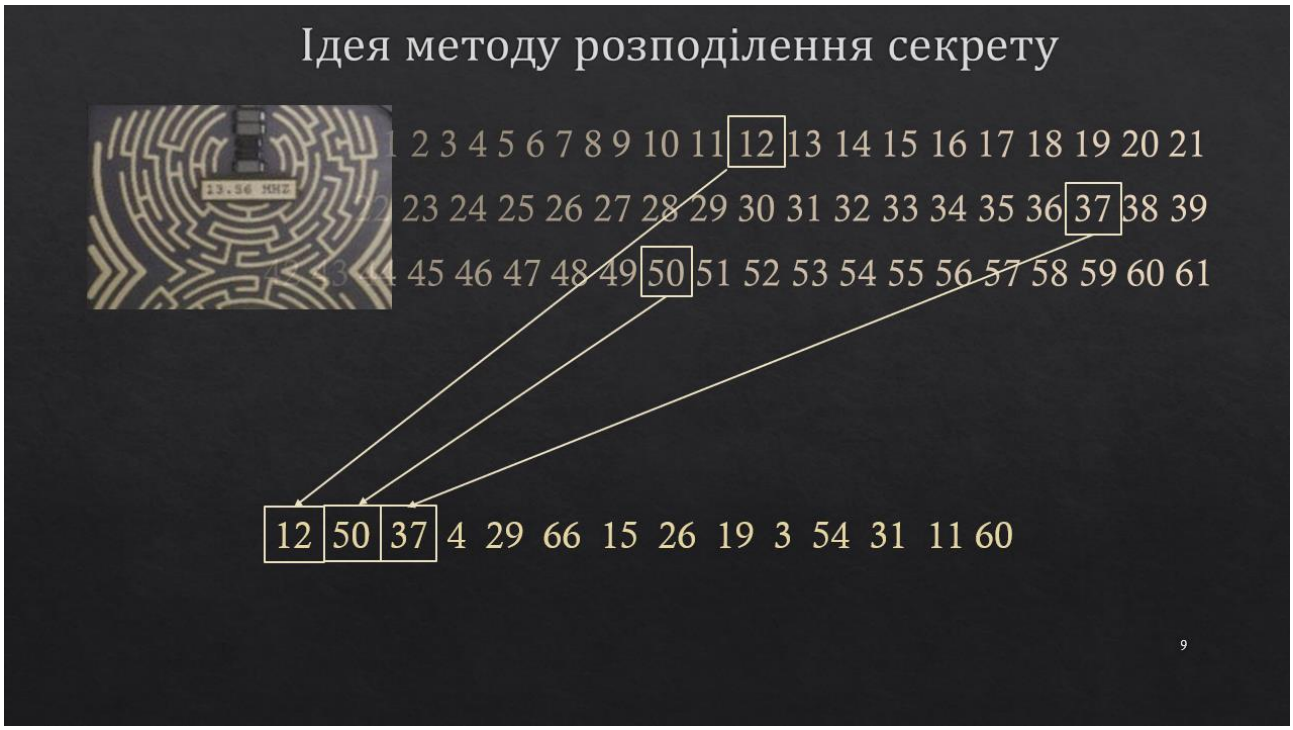
```
            decrypted_image_parts                                    =
self.decrypt_image_parts(encrypted_image_parts, key, num_counters)

            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 1. ПІДСИСТЕМА ЗАШИФРУВАННЯ ЗОБРАЖЕНЬ

Ідея методу розподілення



Реалізація регістру зсуву

Ініціалізація генератора

Алгоритм зашифрування зображення

Ініціалізація генератора

Схема роботи регістру зсуву з зворотнім зв’язком

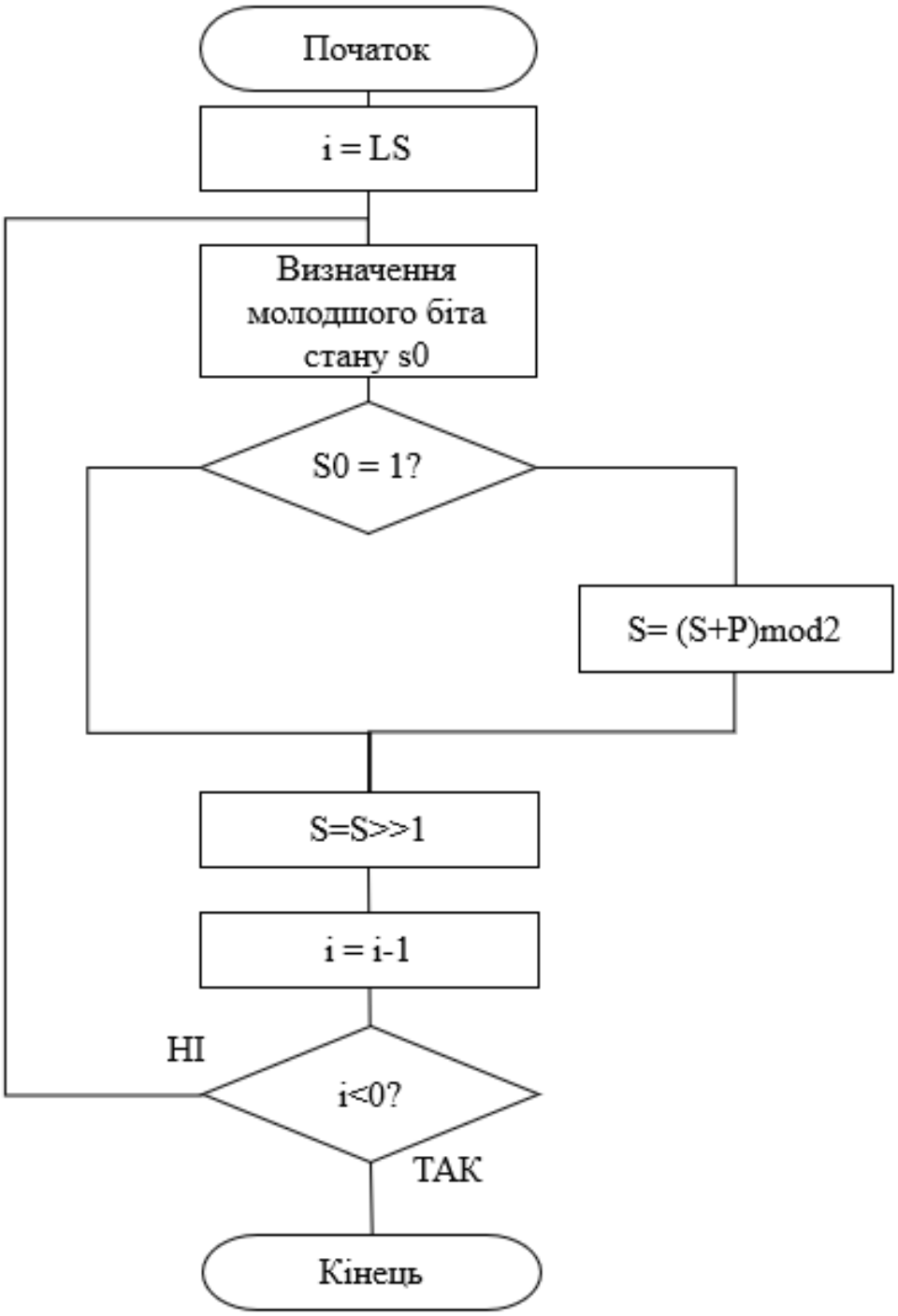
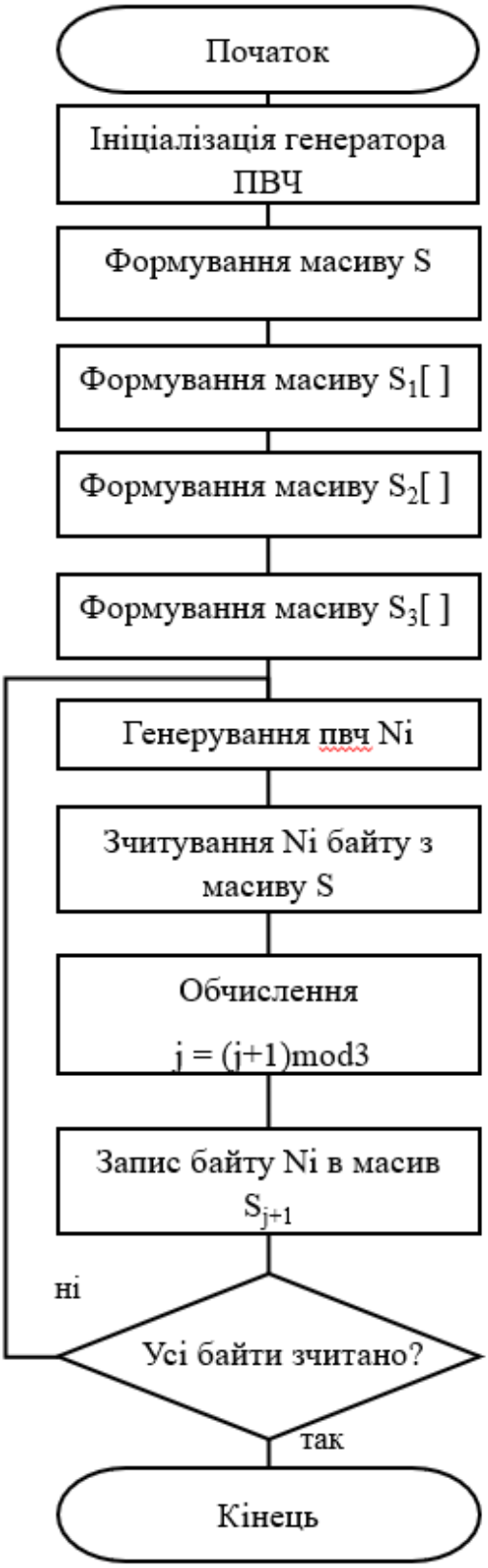


Схема роботи алгоритму розділення секрету



Загальна схема роботи програми

Результати тестування