Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

**Бакалаврська дипломна робота на тему:**

«Засіб для шифрування зображень»

Виконав: студент 2 курсу групи 1БС-21мс
спеціальність 125 – Кібербезпека

_____ Кирилюк О.П.

Керівник зав. кафедри ЗІ д.т.н., проф.

_____ Лужецький В. А.

Рецензент к. т. н., доц. каф. ПЗ

_____ Бабюк Н.П.

**Допущено до захисту**
Завідувач кафедри ЗІ
д. т. н., професор
_____ Лужецький В.А.
«19» червня 2023 р.

Вінниця - 2023 року

Вінницький національний технічний університет
Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 125 – Кібербезпека

**ЗАТВЕРДЖУЮ**
**Зав. кафедри ЗІ д.т.н., проф.**
**В. А. Лужецький**
**20 березня 2023 р.**

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ**

Кирилюку Олександру Павловичу

1. Тема роботи: «Засіб для шифрування зображень»
   Керівник роботи: Лужецький Володимир Андрійович, д.т.н., проф,
   затверджені наказом ректора ВНТУ від 20 березня 2023 року № 67
2. Строк подання студентом роботи 19.06.2023 р.
3. Вихідні дані до роботи:
   - формат зображення – bmp;
   - тип зображення – кольорове;
   - методи захисту – шифрування на основі перестановок і псевдовипадковий розподіл секрету на три частини;
   - розрядність секретного ключа для шифрування – 32.
   - розрядність секретного ключа для розподілу секрету – 32.
4. Зміст розрахунково-пояснювальної записки: Вступ. Огляд підходів до захисту зображень. Розроблення методу захисту зображення. Розроблення програмного засобу. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Алгоритм зашифрування зображень(плакат А4). Алгоритм розшифрування зображень(плакат А4). Структура bmp файлу (плакат А4). Алгоритм перестановки байтів (плакат А4). Алгоритм розподілу секрету (плакат А4). Алгоритм формування псевдовипадвох чисел (плакат А4). Алгоритм роботи програми (плакат А4).

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада Консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1 | Лужецький В. А., д.т.н., проф., зав кафедри ЗІ | 20.03.23 | 16.06.23 |
| 2 | Лужецький В. А., д.т.н., проф., зав кафедри ЗІ | 20.03.23 | 16.06.23 |
| 3 | Лужецький В. А., д.т.н., проф., зав кафедри ЗІ | 20.03.23 | 16.06.23 |

7. Дата видачі завдання 20 березня 2023 року

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів Бакалаврської дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз завдання. Вступ | 20.03.23 –26.03.23 | |
| 2 | Аналіз літературних джерел за напрямком бакалаврської кваліфікаційної роботи роботи | 27.03.23 – 09.04.23 | |
| 3 | Розробка рішень, моделей, алгоритмів | 10.04.23 – 23.04.23 | |
| 4 | Практична реалізація, моделювання, експериментування, результати | 24.04.23 – 21.05.23 | |
| 5 | Оформлення пояснювальної записки, підготовка ілюстративного матеріалу | 22.05.23 – 24.05.23 | |
| 6 | Попередній захист БДР | 25.05.23 – 31.05.23 | |
| 7 | Виправлення зауважень, перевірка БДР на плагіат | 01.06.23 – 15.06.23 | |
| 8 | Представлення БДР до захисту, рецензування | 16.06.23 – 19.06.23 | |
| 9 | Захист БДР | 20.06.23 – 23.06.23 | |

Студент _____ (підпис) Кирилюк О.П.

Керівник роботи _____ (підпис) Лужецький В. А

ЗМІСТ

# АНОТАЦІЯ

Кирилюк О.П. Засіб для шифрування зображень. Бакалаврська дипломна робота / Кирилюк Олександр Павлович – Вінниця ВНТУ, 2023 – 63с. Українською мовою. Рисунків – 19, бібліографі – 11.

Бакалаврська кваліфікаційна робота присвячена розробці методу та програмному засобу для шифрування зображення. У роботі було здійснено аналіз відоміх підходів до захисту зображення, вказано на їхні переваги та недоліки, та доцільність розроблення власного методу. Розроблено власний метод шифрування зображення, та програмний засіб, який реалізує цей метод.

# ABSTRACT

Kirilyuk O.P. Image encryption tool. Bachelor thesis / Kyrylyuk Oleksandr Pavlovich – Vinnytsia VNTU, 2023 – 63p. In ukrainian. There are 19 figures, 11 bibliographies.

The bachelor qualification work is devoted to the development of a method and a software tool for image encryption. The paper analyzed known approaches to image protection, indicated their advantages and disadvantages, and indicated the feasibility of developing one's own method. A proprietary image encryption method and a software tool that implements this method have been developed.

# ВСТУП

Інформація має важливе значення в життєдіяльності людства. При цьому вона стає все більш вразливою через зростаючі обсяги збережених і переданих даних. Тому все більшу важливість набуває проблема захисту інформації від несанкціонованого доступу під час передачі і зберіганні.

Візуальна криптографія це спеціальний метод шифрування, суть якого полягає у прихованні інформації в зображеннях таким чином, що воно може бути розшифровано тільки якщо використовуються правильні ключ-зображення. Перевагою даного методу є те, що він є простим як у реалізації, так як не вимагає спеціального обладнання і може бути розшифрована людським оком.

Існуючі схеми мають дві складові: розподіл і відновлення секрету. До поділу відноситься формування частин секрету і розподіл їх між членами групи, що дозволяє розділити відповідальність за секрет між її учасниками. Зворотна схема повинна забезпечити його відновлення за умови доступності його зберігачів у деякій необхідній кількості.

Об’єктом дослідження є процес захисту зображення.

Предметом дослідження є метод та засіб для шифрування зображення.

Метою бакалаврської кваліфікаційної роботи є підвищення захищеності секретного вмісту зображень за рахунок розробки методу шифрування та програмного засобу для шифрування.

Для досягнення мети необхідно:

- проаналізувати підходи до захисту зображень;
- розробити метод захисту зображення;
- розробити програмний засіб для шифрування зображення.

Практичну цінність роботи складає програмний засіб, для захисту зображень з секретним змістом.

# 1 ОГЛЯД ПІДХОДІВ ДО ЗАХИСТУ ЗОБРАЖЕНЬ

## 1.1 Візуальна криптографія

Візуальна криптографія визначається як процес абсолютного шифрування цифрових даних, які можуть бути розкодовані тільки з використанням зорової системи людини. Ця ідея дозволить виводити дані, в нашому випадку зображення, які передаються або зберігаються в цифровій формі, не хвилюючись про те, що дані можуть бути перехоплені і випадково виявлені неуповноваженими особами. Первинне повідомлення кодується в два або більше шарів. Коли дивився на окремі шари, то вони не розкривають ніякої інформації про повідомленні, що міститься в них і нагадують випадковий шум [1].

Наор і Шамір продемонстрували (k, n) візуальну схему секретного обміну, де зображення було розбито на n частин, таким чином, що будь-хто, що володів будь-якими k частинами міг розшифрувати його, в той час як будь-які k-1 частин не давали ніякої інформації про зміст вихідного зображення. Коли всі k частини будуть накладені один на одного, ми побачимо вихідне зображення [2].

Алгоритм візуального шифрування має ряд наступних властивостей:

– незалежність шифрування кожного пікселя;

– простота вибору матриць;

– однакові дії для кожного вихідного пікселя;

Легко побачити, що для схеми візуальної криптографії (k, n), алгоритм шифрування зображення має такі властивості:

– Якщо візуальна криптографія використовується для безпечного спілкування, то відправник передасть одну або більше копії випадкового шару 1 завчасно одержувачу [2].

– Якщо у відправника є повідомлення, він створює шар 2 для конкретного відправленого шару 1 і передає його одержувачу. Одержувач з'єднує два шари і отримує секретну інформацію. При цьому всьому, йому не потрібно використовувати пристрої розшифрування, робити складні математичні розрахунки, і навіть не обов'язково застосовувати комп'ютер (якщо зображення

знаходяться в друкованому вигляді). Система є стійкою до тих пір, поки обидві частини зображення не потраплятьв чужі руки. Якщо ж перехоплений тільки один шар, то розшифрування вихідного зображення неможливе [2].

Як додатковий засіб криптостійкості можливе використання візуальної криптографії на основі стеганографічних методів. Одним із таких методів є приховання шарів у зображенні. Зашифроване зображення виходить накладенням випадкового шуму на картинку з текстом із застосуванням операції XOR або аналогічних, але при цьому використовується робота з матрицями пікселів. Стійкість методу до злому - безумовна, але при цьому потрібні і одноразові ключі. Крім того, відправка шумових графічних повідомлень малоцікава в сучасному світі: навіть якщо у одержувача немає комп'ютера і криптопрограм, сам факт отримання підозрілого шумового графічного повідомлення він приховати не може [1].

Таким чином, використовуючи спеціально підготовлені на комп'ютері зображення з впровадженим текстом, одержувач, також має ключове зображення у вигляді роздрукованої картинки, може отримати повідомлення від відправника та прочитати його поєднанням аркушів паперу на просвіт, без використання криптографічного або стеганографічного програмного забезпечення [2].

Даний метод може використовуватися для приховування фактів використання стеганографії, для таємного зберігання або передавання ключів шифрування при загрозі обшуків або оглядів, надсилання коротких повідомлень в умовах листування для осіб, які не мають доступу до комп'ютерів [2].

Розділення секрету – термін в криптографії, під яким розуміють будь-який з способів розділення секрету серед групи учасників, кожному з яких дістається якась своя частина секрету. Секрет може відтворити тільки група учасників з первісної групи, причому входити в групу має не менше деякого відомого початкового числа [3].

Схеми поділу секрету застосовуються у випадках, коли існує значна ймовірність компрометації одного або декількох учасників розподілу секрету, але

ймовірність недобросовісної змови значної частини учасників вважається досить малою [3].

## 1.2 Схеми розподілу секрету

### 1.2.1 Найпростіша схема

Нехай $S_n$ – розділяємий секрет, $n$ – кількість учасників групи. Розділення секрету між учасниками відбувається в декілька етапів [4].

Із скінченного поля $Z_P$ ($p$ - просте число) вибираються випадково $n$-1 елементів:$S_1, S_2, \dots, S_{n-1}$ [4].

Формується відкрите значення S: $S_1, S_2, \dots, S_n(mod\ p)$.

Кожному учаснику відправляється його частина - $S_j$.

Таким чином, кожен з учасників групи отримує свою частину загального секрету і знає відкрите значення $S$. Для відновлення секрету учасники повинні об'єднати свої частини і відновити секрет $S_1, S_2, \dots, S_n(mod\ p)$.

Наведена схема поділу секрету є ідеальною - в даному прикладі розмір секрету дорівнює розміру частини секрету [4].

### 1.2.2 Схема Шаміра

Схема Шаміра заснована на тому факті що для однозначного відновлення багаточлена степені $t$-1 потрібно не менше $t$ його значений в будь-яких $t$ попарно різних точках (трьох точок достатньо для відновлення параболи, двох – для відновлення прямої) [4].

Розподіл секрету відбувається в декілька етапів.

На етапі ініціалізації вибирається $n$ різних елементів $x_j$ скінченного поля $Z_p$ ($p$ - просте число).

Значення $x_j$ є відкритими значеннями (наприклад, $x_j = i$).

Секретний ключ $K$ розділяється між учасниками групи. Для цього вибирається багаточлен $f(x) = a_0 + a_1 x + a_{a-1} x^{t-1}$ у скінченому полі $Z_p$, $a_0 = k$ і визначається значення багаточлена в вибраних точках $y_i = f(x_i)$.

Кожне з обчислених значень відправляється по захищеному каналу учасникам групи - це і є частини секрету. Значення $k$ потім знищується.

Секрет можна відновити двома способами:

– розв’язавши систему лінійних рівнянь.

– обчислюючи значення інтерполяційного багаточлена Лагранжа [5].

Схема лінійних рівнянь має вигляд $y_i = f(x_j)$ і розв’язується відносно коефіцієнтів $a_0, a_1, a_2, a_{t-1}$. В результаті вираховується секрет $a_0 = k$. Секрет відновлюється будь-якою групою з $t$ учасників, так як визначник матриці А відповідної с истеми лінійних рівнянь, завжди не дорівнює 0.

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \cdots & x_t^{t-1} \end{bmatrix}$$

Визначник матриці є визначником Вандермонда і обчислюється за формулою [5].

$$\det A = k \prod 1 \le j \le k \le t (x_k - x_j) \bmod p .$$

Визначник завжди не дорівнює 0, за умови що всі $x_j$ відмінні, так як твірна ненульових елементів в полі завжди відмінна від 0 [3].

Секрет також можна відновити за допомогою інтерполяційної формули Лагранжа. Формула має вигляд [3]:

$$a(x) = \sum_{j=1}^{t} y_j \prod_{\substack{1 \le k \le t \\ k \ne j}} t \frac{x - x_k}{x_j - x_k}$$

Відновлюється секрет за формулою [5]:

$$a_0 = a(0) = \sum_{j=1}^{t} y_i \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{x_k}{x_j - x_k}$$

Дана схема забезпечує повну таємність (теоретико-інформаційну стійкість) проти спроби обчислення секрету будь-якою групою з $j$ учасників, де $j < t$, що володіють необмеженою обчислювальною потужністю. Досконала стійкість обґрунтована тим, що при кількості учасників $j$ $(j < t)$ система має безліч рішень що утворюють лінійний простір рішень системи лінійних алгебраїчних рівнянь [4].

Велику їх кількість зумовлено тим фактом, що в цьому випадку ранг матриці А буде менше кількості невідомих.

Досконала стійкість обґрунтована тим, що не існує критерію, який дозволив би виділити найбільш пріоритетний розв'язок серед відповідних [4].

1.2.3 Схема Асмута – Блума

Асмут і Блум запропонували таку схему [3].

Нехай М – розділяємий секрет, $p$ – просте число, більше любого із секретів, який буде розділятись $P > (\max(M)$.

Вираховуються $N$ взаємно простих числа $d_1, d_2, \dots, d_N$ таких, що:

$\forall i: d_j > P;$

$\forall i: d_j > d_{j+1};$

$\prod_{i=1}^{K} d_j > p * \prod_{i=N-K+2}^{N} d_j.$

Вибирається випадкове число $r$ таке що:

$$\sqrt[K]{M`} < d_j \ll \sqrt[K-1]{M`}, i = 1, \dots, n,$$

де $M' = M + rP$

Генерація частин секрету.

У частині секрету, що видається $i$ - му учаснику схеми, є три числа $\{P, d_j, k_j\}$, де $k_i$ обчислюється як $k_i \equiv M`(mod\ d_i)$ [3].

Відновлення секрету.

Зібравши $K$ частин секрету разом, та використовуючи китайську теорему про залишки, вирішується система рівнянь відносно невідомого М':

$$\begin{cases} M` \equiv k_1(mod\ d_1) \\ M` \equiv k_2(mod\ d_2) \\ \dots \\ M` \equiv k_K(mod\ d_K) \end{cases}$$

Учасники отримують загальний секрет М, розв'язавши систему рівнянь [3].

1.2.4 Схема заснована на еліптичних кривих

Підготовча фаза. Нехай $E$ - еліптична крива, визначена в кінцевому полі $F(p)$. Властивості точок еліптичної кривої визначаються як властивості адитивних абелевих груп [5].

Для них визначаються операції додавання:

$$P(x,y) = P_1(x_1,y_1) + P_2(x_2,y_2)$$

Важливою операцією для точок еліптичної кривої є помноження точки на число [5]:

$$P(x,y) = k \cdot P_1(x_1,y_1)$$

Генерація частин секрету.

Підставляючи у вираз різні *t* можна отримати приватні секрети.

Нехай $j$ - номер абонента, якому відсилається секрет [5]. Тоді приватні секрети абонентів рівні:

$$S_j = P_0 + iP_1 + i^2P_2 + \cdots + i^{K-1}P_{K-1}$$

Відновлення секрету.

Розглянемо порогову (*N*, *K*) схему [5]. Приватні секрети абонентів в загальному випадку виходять відповідно до виразу:

$$S_j \left| \begin{matrix} N-1 \\ 0 \end{matrix} \right. = \sum_{j=0}^{k-1} P_i t_j^i$$

Де $N$ – загальна кількість секторів на першому кроці, $j$ – номер секрету.

Для структури доступу вираз приводится до виду [5].

$$S_j \left| \begin{matrix} N-1 \\ 0 \end{matrix} \right. = \sum_{j=0}^{k-1} P_i t_j^i = \sum_{j=0}^{k-1} P_i (x_j)_j$$

Далі обраховується спільний вираз

$$S_{1,j} \left| \begin{matrix} K-2 \\ 0 \end{matrix} \right. = S_j(x_1)_{j+1} - S_{j+1}(x_1)_j = \sum_{i=0}^{K-1} P_i \left( (x_i)(x_1)_{j+1} - \left(x_j\right)_{j+1}(x_1)_j \right) =$$

$$= \sum_{i=0}^{K-1} P_i (x_2)_j$$

Шляхом аналогічних перетворень можна отримати

$$R = S_{K-1}, j = S_{K-2}, (x_{K-1})_{j+1} - S_{K-2,j+1}(x_{K-1})_j = P_0 x_K$$

Значення R та $x_K$ і є отриманим секретом [5].

## 1.3 Методи шифрування зображень

### 1.3.1 Метод чорно-білих зображень

Для деякої заданої схеми візуальної криптографії (*k, n*), алгоритм шифрування зображення має такі властивості як:

– регулярність ( властивість при якій для кожного вихідного пікселля виробляються однакові дії);

– незалежність (властивість при якій шифрування кожного пікселя на виході відбувається не залежно від інших пікселів);

– простота ( властивіть при якій маттриці вибираються випадковим чином відповідно до сукупності, яка задана схемою поділу секрету) [6].

Відбувається поділення пікселів шифрованого зображення на деякі менші блоки. Відбувається певна умова, а саме: кількість протилежних блоків (білих, чорних) завжди буде однаковою. Після поділення пікселя на дві частини отримуються один білий, та один чорний блок. І відповідно коли при розділенні кількох пікселів, з кожного із них буде отримуватися один білий, та один чорний блок відповідно [6].

На рис 1.2 можна бачити, що при розділенні пікселю на чотири різні частини він може мати шість різніз станів. Відповідно до цього положення пікселю на першому шарі і на другому будуть мати зовсім різні положення, і на другому шарі положення будуть мати два різні положення: однакове, або різне в порівнянні до піксел. першого шару. При такій умові коли пікселі обох шарів будуть ідентичними, тоді результатом перекриття буде піксель, який наполовину білий і наполовирну чорний. Значення такого пікселю важається сірим, або порожнім. При умові коли пікселі протилежні, тоді в результаті буде отримано повністю чорний піксель, який важається інформаційним. [6].

У тому випадку коли пікселі першого шару зображення (прозорі пікселі), будуть мати певний випадковий стан з шести можливих. Шари 1 та 2 будуть ідентичними, якщо не рахувати інформаційні пікселі, а саме чорні. Вони у свою чергу будуть мати протилежний стан відповідно до того ж пікселю першого шару.

При накладенні одного шару на другий, ті області де стани однакові, будуть мати сірий колір, а ті області які мають зворотні стани, будуть мати блоки чорного кольору [6].

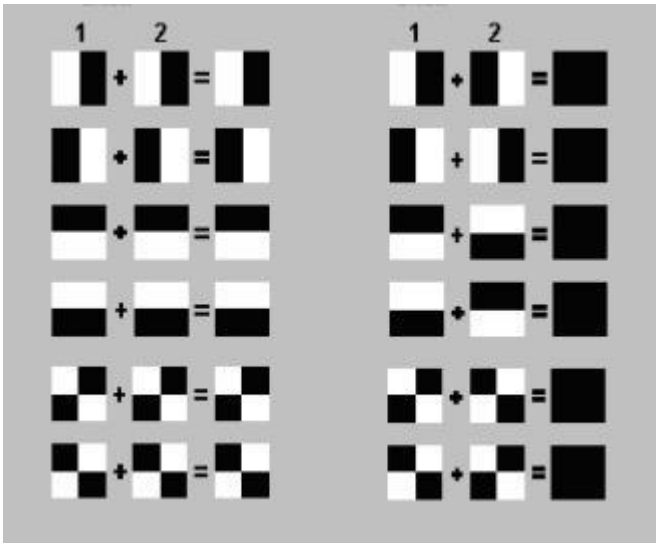



Рисунок 1.1 – Модель процесу накладання шарів

Один із прикладів використання візуальної криптографії є безпечне спілкування, при які відправнику необхідно передвати копії випадкових шару 1 завчасно для отримувача. У тому випадку коли отримувач отримав повідомлення, йому необхідно створити шар 2 для шару 1 який був отриманий раніше і передати його далі для одержувача. Відповідно після чого отрисувачу необхідно об’єднати отримані шари і після цього він зможе отримати зашифровану інформацію. Основною із переваг даного методу є те, що отримувачу не довилося використовувати будь які засоби для розшифрування, не потрібно робити ніяких математичних операціій, які зазвичай є досить важкими, а якщо зображення знаходяться у друкованому вигляді тоді не обов’язковою умовою є навіть використання комп’ютера. Негативної частиною такого методу є його досить проста можливість зламу, якщо частини потраплять в чужі руки то вони зможуть без проблем дізнатися зашифровану інофрмацію. Тому потрібно відправляти шари завжди по різних каналах і слідкувати за конфіденційністю цих операцій. Якщо ж зловмисник отримає тільки один шар, то ніякої інформації не дізнається [6].

При використані даного підходу однією з умов є те, що немає можливості дізнатися використання пікселя другого шару, який використовується для створення сірого або чорного пікселя, до того часу поки невідомо, який стан цього пікселя на першому шару, для того щоб можна було дізнатися яким буде результат перекриття. У тому випадку коли при використанні даного методу повністю дотримується випадковий підхід до розбиття пікселів на блоки, тоді можна важати що підхід пропонує абсолютну надійність і секретність [6].

1.3.2 Метод кольорових зображень

Початковими зображенням є зображення BMP-формату розрядністю в 24 біти, де по 8 біт інформації припадає кожен колір. Суть алгоритму полягає в тому, що секретне зображення розподіляється на складові кольору: синій, зелений червоний. потім кожна складова записується в молодші біти одного з зображень-контейнерів. Тобто після розподілення зображень, кожен контейнер (*R, G, B*) буде містити в собі одну складову кольору секретного зображення [7].

Перед запуском алгоритму визначається кількісне значення кожного з трьох відтінків в кожному зображенні-контейнері для того, щоб визначити, в яке з зображень ховати *B*- складову, в яке - *R*, а в яке - *G,* Контейнери вибираються таким чином, щоб різниця в кольорах кольорового примітиву секретного зображення і контейнера була мінімальна. Потім з кожного кольорового примітиву два старших біти записується в молодші біти в відповідний контейнер та колір. Два молодших біти в двох кольорах, що залишились, зануляються. Така операція проводиться для всіх пікселів [7].

Для відновленні зображення беремо почергово, кожен піксель з кожного зображення-контейнера. Два молодших біта кожного кольору в цих пікселях ст ають старшими бітами дві відповідні колірні складові об’єднуються, так як під час розподілу молодші біти не розподіленого кольору обнулялися, то нульове значення матиме тільки одна колірна складова в кожному контейнері. Таким чином відновлюється колір відповідного пікселя секретного зображення (з деякою погрішністю). Далі повторимо цю операцію для всіх пікселів і отримаємо відновлене секретне зображення [7].

Для відновлення зображення, розподіленого даним методом, необхідно мати всі три зображення-контейнера, отриманих в результаті розподілення. В іншому випадку замість відновленого зображення відновиться лише колірний шум [7].

В даного методу існує недолік. У ході розподілу секретного зображення спотворюється, так як, використовуються лише два старших біти з восьми [7].

1.3.3 Метод розподілу зображень за допомогою Фур’є образу

Нехай зображення представлено RGB матрицею. Для отримання Фур'є - образу зображення можна застосувати дискретне перетворення Фур'є. Відновлення полягає в застосуванні зворотного дискретного перетворення Фур'є [7].

Нехай дано цифрове представлення кольорового зображення за допомогою трьох матриць (*R, G, B*), що утворюють трьох мірний масив *C = cat (3, R, G, B )*, де cat - конкатенація. Якщо матриці мають розмір $m \times n$, то $C$ містить $m \times n \times 3$ елементів. Кожен елемент визначається трьома індексами $C[m, n, z]$, де $z = 1, 2, 3$ відповідає матрицями *R, G, B*:

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

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

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

Тут використовуються три матриці, які називають червоним *R*, зеленим *G* і синім *G* колірним каналом або кольоровою компонентою. З огляду на не комутативность конкатенації, порядок матриць є важливим. Тобто кольорове зображення можна представити у вигляді трьох матриць розміру $m$x$n$ [7].

Покажемо для прикладу для компоненти *R*.

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

Де кожна компонента має вигляд $f(x, y)$, в якій амплітуда $f$ в будь-якій точці інтенсивності чи значення кольору, в даному випадку R – червоного, а $x$ і $y$

просторові координати. Нехай $f$ ($x$, $y$) при $x$ = 0, 1, $m$-1, y = 0, 1, $n$-1 позначає зображення розміру $m$x$n$. Тоді двохвимірне дискретне перетворення Фур'є [7]:

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

Зворотне двохвимірне дискретне перетворення Фур'є:

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

В даному розділі ми розглянули основні методи, які використовуються для зашифрування зображень.

## 1.4 Робота з зображеннями

### 1.4.1 Формати зображень

Розглянемо найбільш поширені графічні формати, що використовуються для створення зображень, фотографій, анімацій і т.д.

BMP (Windows Device Independent Bitmap). Основний формат Windows. Він підтримується всіма графічними редакторами, що працюють під управлінням цієї операційної системи. Застосовується для зберігання растрових зображень, призначених для використання в Windows і, на цьому сфера його застосування закінчується. Використання BMP не для потреб Windows є достатньо поширеною помилкою [8].

GIF (CompuServe Graphics Interchange Format). Незалежний від апаратного забезпечення формат GIF був розроблений в 1987 році (GlF87a) фірмою CompuServe для передачі растрових зображень по мережах. У 1989-му формат був модифікований (GIF89a), були додані підтримка прозорості і анімації. GIF використовує LZW-компресію, що дозволяє непогано стискувати файли, в яких багато однорідних заливок (логотипи, написи, схеми) [8].

JPEG (Joint Photographic Experts Group). Строго кажучи JPEG не формат, а алгоритм компресії, заснований не на пошуку однакових елементів, а на різниці між пікселями. Чим вище рівень компресії, тим більше даних відкидається, і тим

нижче якість. Використовуючи JPEG можна отримати файл в 1-500 разів менше, ніж BMP! Спочатку в специфікаціях формату не було CMYK, Adobe додала підтримку кольороділення, проте CMYK JPEG в багатьох програмах створює проблеми. JPEG'ом краще стискуються растрові картинки фотографічної якості, ніж логотипи або схеми [8].

TIFF, TIF (Target Image File Format). Апаратний незалежний формат TIFF, один з найпоширеніших і надійніших на сьогоднішній день, його підтримують практично всі програми на PC і Macintosh так чи інакше пов'язані з графікою. Йому доступний весь діапазон колірних моделей від монохромної до RGB, CMYK і інших. TIFF може містити обтравочні контури, альфа-канали, шари, інші додаткові дані. У форматі TIFF є можливість збереження із застосуванням декількох видів компресії: JPEG, ZIP, але, як правило використовується тільки LZW-компресія [8].

EPS (Encapsulated PostScript). Формат використовує спрощену версію PostScript: не може містити в одному файлі більш за одну сторінку, не зберігає ряд установок для принтера. EPS призначений для передачі векторів і растру у видавничі системи, створюється майже всіма програмами, що працюють з графікою. Використання його має сенс тільки тоді, коли вивід здійснюється на PostScript - приладі. EPS підтримує всі необхідні для друку колірні моделі. EPS має багато різновидів, що залежить від програми-творця. Найнадійніші EPS створюють програми виробництва Adobe Systems: Photoshop, Illustrator, InDesign [8].

PM (Page Maker). Формат програми верстки Adobe Systems. Надзвичайно простий в плані можливостей пакет. Призначався насамперед для переходу з ручного вигляду верстки на комп'ютерний з мінімальними витратами на навчання персоналу. Поширення у нас набув завдяки своєчасній русифікації і знову таки легкості освоєння для новачків. В даний час розвиток пакету зупинений [8].

ID (InDesign). Кодова назва «Quark Killer» Послідовник PM, покликаний потіснити конкурентів на видавничому ринку, насамперед Quark. ID - виявився невдалим, і є достатньо складним і незручним пакетом. До переваг можна віднести лише вбудований інтерпретатор PostScript і сумісність з іншими продуктами Adobe [8].

PDF (Portable Document Format) - запропонований фірмою Adobe як незалежний від платформи формат для створення електронної документації, презентацій, передачі верстки і графіки по мережах. PDF-файли створюються шляхом конвертації з PostScript-файлів або функцією експорту ряду програм. Формат спочатку проектувався як засіб зберігання електронної документації. Тому всі дані в нім можуть стискуватися, причому по-різному: JPEG, RLE, CCITT, ZIP. PDF може також зберігати всю інформацію для вивідного пристрою, яка була в початковому PostScript-файле [8].

Adobe PostScript - мова опису сторінок. Був створений в 80-х роках для реалізації принципу WYSIWYG (What You See is What You Get). Файли цього формату фактично є програмою з командами на виконання для вивідного пристрою. Такі файли містять в собі сам документ, зв'язані файли, використані шрифти, і іншу інформацію: плати кольороділення, додаткові плати, лініатуру растру і форму растрової точки для кожної плати і інші дані для вивідного пристрою. Дані в PostScript-файлі, як правило, записуються в двійковому кодуванні (Binary). Бінарний код займає удвічі менше місця, чим ASCII [8].

CDR - формат популярного векторного редактора CorelDraw. Свою популярність і розповсюдження пакет отримав завдяки простоті використання, інтерактивним спецефектам (лінзам, нестандартним градієнтам і так далі). Широкі можливості цієї програми, в плані ефектів, пояснюються багатшою внутрішньою мовою опису сторінок ніж у продуктів Adobe, що використовують PostScript. Саме це і є основним мінусом CorelDraw. PostScript з корелівськими спецефектами часто є головним болем друкарень і бюро до друкарської підготовки [8].

CCX - формат векторної графіки від компанії Corel. Окрім CorelDraw нічим не підтримується. Для поліграфії і Інтернету непридатний. До переваг можна віднести лише невеликий об'єм файлів, збережених в цьому форматі і наявність безлічі відмінних кліпартів [8].

Векторна графіка є математичним описом об'єктів відносно початку координат. Так, для відображення прямої потрібні координати всього двох точок. Для кола - координати центру і радіус і так далі [8].

Графічні формати можуть містити в собі масу додаткової інформації: альфа-канали, шляхи, колірну модель, лініатуру растру і навіть анімацію. Вибір формату для поліграфічної продукції насамперед залежить від вивідного пристрою. Фотонабірні автомати працюють під управлінням мови PostScript. Тому для поліграфії основними форматами зберігання даних є TIFF і EPS. Відповідно формат растрової і векторної графіки. Останнім часом набирає силу PDF (Portable Document Format) [8].

1.4.2 Бінарний код зображення

Щоб зберегти в двійковому коді фотографію, її спочатку віртуально розділяю на велику кількість дрібних кольорових крапок, які називаються пікселями (щось схоже до мозаїки) [9].

Після розбиття на крапки колір кожного пікселя кодується у бінарний код і записується на запам'ятовуючому пристрої [9].

Якщо говорять, що розмір зображення складає, наприклад, 512 x 512 крапок, це означає, що воно є матрицею, сформованою з 262144 пікселів (кількість пікселів по вертикалі, перемножена на кількість пікселів по горизонталі) [9].

Приладом, який "розбиває" зображення на пікселі, є будь-яка сучасна фотокамера (у тому числі веб-камера, камера телефону) або сканер.

І якщо в характеристиках камери значиться, наприклад, "10 MegaPixels", значить кількість пікселів, на які ця камера розбиває зображення для запису в двійковому коді, - 10 мільйонів [9].

Чим на більшу кількість пікселів розділено зображення, тим реалістичніше виглядає фотографія в декодованому вигляді (на моніторі або після роздрукування) [9].

Проте якість кодування фотографій залежить не лише від кількості пікселів, але й від їх колірної різноманітності [9].

Алгоритмів запису кольору в двійковому коді існує декілька. Найпоширенішим з них є RGB. Ця абревіатура - перші літери назв трьох основних кольорів: червоного - англ. Red, зеленого, - англ. Green, синього, - англ. Blue [9].

Зі шкільних уроків малювання, Вам, напевно, відомо, що змішуючи ці три кольори у різних пропорціях, можна отримати будь-який інший колір або відтінок [9].

На цьому і побудований алгоритм RGB. Кожен піксель записується в двійковому коді шляхом вказання кількості червоного, зеленого і синього кольору, що беруть участь в його формуванні [9].

Чим більше бітів виділяється для кодування пікселя, тим більше варіантів змішування цих трьох каналів можна використати і тим значнішою буде колірна насиченість зображення [9].

Колірна різноманітність пікселів, з яких складається зображення, називається глибиною кольору [9].

Якщо для кодування кожного пікселя якогось зображення виділяється 8 бітів двійкового коду, колірна різноманітність складе 256 кольорів [9].

Глибина кольору 12-бітів дасть 4096 кольорів, 16-бітів - 65536 кольорів, 18-бітів - 262144 кольорів [9].

Максимальна глибина кольору, що використовується в комп'ютерній техніці, - 24 біти. Таку глибину часто називають True Color ("Справжній колір"). Вона дозволяє відобразити близько 16,7 млн. кольорів. Око людини не здатне сприймати більшу їх кількість [9].

Проте, часто зустрічається й так звана 32-бітна глибина кольору. Вона не передбачає збільшення кількості відтінків. Додаткові біти, які виділяються для кодування кожного пікселя, призначені для регулювання ступеня його прозорості або ж не використовуються [9].

Описана вище техніка формування зображень з дрібних крапок є найпоширенішою і називається растровою. Але окрім растрової графіки, в комп'ютерах використовується ще й так звана векторна графіка [9].

Векторні зображення створюються тільки за допомогою комп'ютера (фотокамери цього робити "не вміють") і формуються не з пікселів, а з графічних примітивів (ліній, багатокутників, кіл та ін.) [9].

Навіщо потрібна векторна графіка? У відомій дитячій пісеньці співається, що для зображення "чоловічка" достатньо намалювати всього дві "палки" й "огірочок". А уявіть, наскільки складно вручну скласти чоловічка з великої кількості крапок [9].

Векторна графіка - це креслярська графіка. Вона дуже зручна для комп'ютерного "малювання" і широко використовуються дизайнерами при графічному оформленні друкарської продукції, у тому числі створенні величезних рекламних плакатів, а також в інших подібних ситуаціях [9].

Векторне зображення в двійковому коді записується як сукупність примітивів із вказанням їх розмірів, кольору заливки, місця розташування на полотні і деяких інших властивостей [9].

Наприклад, щоб записати на запам'ятовуючому пристрої векторне зображення кола, комп'ютеру достатньо в двійковий код закодувати тип об'єкту (коло), координати його центру на полотні, довжину радіусу, товщину і колір лінії, колір заливки [9].

У растровій системі довелося б кодувати колір кожного пікселя. І якщо розмір зображення великий, для його зберігання знадобилося б значно більше місця на запам'ятовуючому пристрої [9].

Однак, векторний спосіб кодування не дозволяє записувати в двійковому коді реалістичні фото. Тому усі фотокамери працюють тільки за принципом растрової графіки. Звичайному користувачу мати справу з векторною графікою в повсякденному житті доводиться не часто [9].

# 2 РОЗРОБЛЕННЯ МЕТОДУ ЗАХИСТУ ЗОБРАЖЕННЯ

## 2.1 Метод шифрування зображення

Розглянувши основні підходи захисту зображення їхні переваги та недоліки, було розроблено власний метод, який дозволить позбутися недоліків попередніх, та дозволить покращити швидкодію процесу зашифрування та розшифрування. Граф – схема алгоритму зашифрування зображено на рис. 2.1.

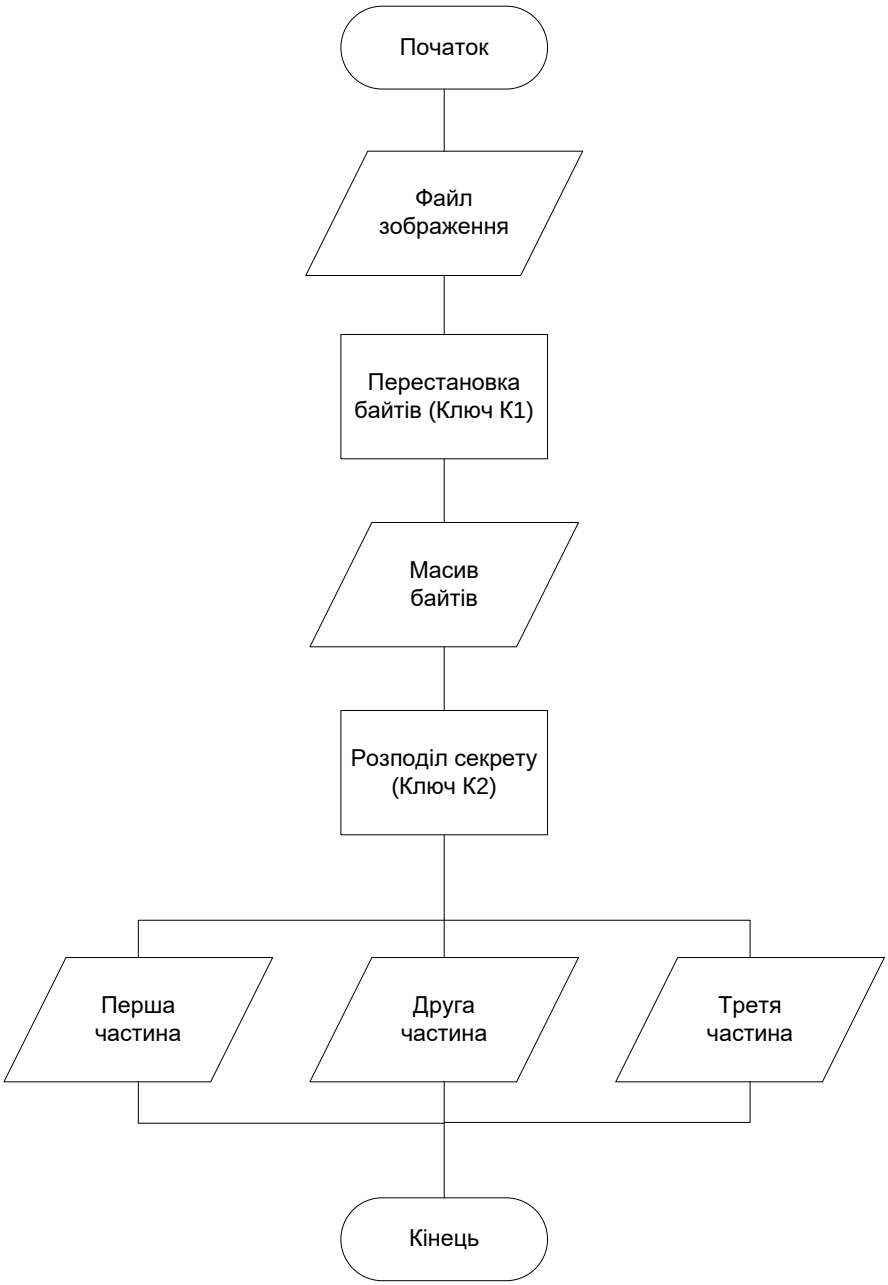



Рисунок 2.1 – Алгоритм зашифрування зображення

Розроблений метод включає в себе такі основні етапи як:

– Перестановка байтів. На цьому етапі необхідно вести значення ключа К1, відповідно до якого розроблений генератор ПВЧ перемістить значення з файлу зображення в масив байтів в хаотичному порядку. Також необхідно додати, що перед перестановкою байтів необхідно відділити заголовок файлу, це необхіднoo щоб не змінилися розміри та розширення. На данному етапі на виході ми отримуємо зашифрований файл, який без правильного ключа розшифрувати ніяк не вийде, але для кращої криптостійкості є ще наступний етап.

– Розділення секрету. На цьому етапі масив байтів отриманий після перестановки, розділяється на три рівні частини, для цього необхідно використати ключ К2. Розділення відбувається в хаотичному порядку в три блоки на рівну кількість значень, за це відповідає регістр зсуву з лінійним зворотнім зв’язком. Після виконання цього етапу на виході отримуються три зображення, які самі по собі не містять ніякої корисної інформації, але в той же час потрібні люди при зборі усіх частин завжди зможуть відновити зображення.

Розроблений метод включає роботу з зображенями такого формату як bmp, з якого необхідно вирахувати інформацію про кількість пікселів, та розмірність. Зображення данного формату складається із двох частин: узаголовку файлу міститься загальна інформація про файл, основні поля, сигнатура формату, та його розмір, також його ширина ти висота у пікселях, а в масиві пікселів міститься вся інформація про кольори зображення. Схема зображення наведена на рис.2.2.


Рисунок 2.2 – Структура зображення

Метод розшифрування зображення, буде обернений до методу зашифрування та зображений на рис.2.3.

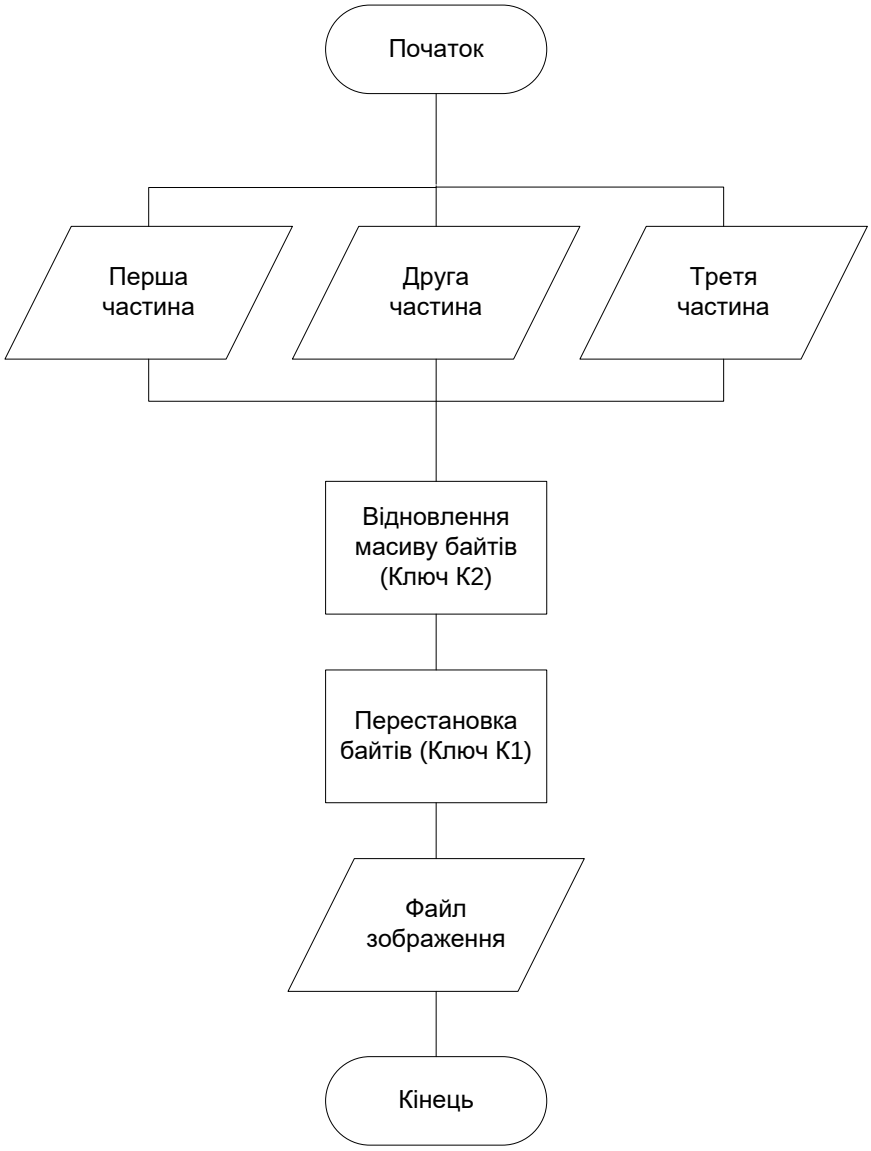



Рисунок 2.3 – Алгоритм розшифрування зображення

Як можна бачити на рис.2.1., на першому етапі по розшифруванню зображення необхідно зібрати усі частини, завдяки чому в розробленого методу досить висока криптостійкість і через те, що розроблені підходи а саме, перестановки байтів та розділення секрету, працюють на базі генератора та регістру зсуву з лінійним зворотнім зв’язком, а це є досить не складні операції, то порівнючи розроблений метод з відомими в розроблений дозволив в значній мірі покращити швидкодія пристою.

## 2.2 Алгоритм перестановки байтів

Алгоритм перестановки байтів для шифрування зображення, зображено на рис.2.4.


Рисунок 2.4 – Алгоритму перестановки байтів

На першому етапі  методу необхідно вести значення даних користувачів, а саме необхідно встановити значення приватного ключа К1, який буде використано для шифрування.  Значення ключа буде використано для того, щоб встановити параметри генератора псевдовипадкових чисел(ПВЧ), який буде описаний у майбутньому розділі.

Прикладу перестановки байтів наведено на рис.2.5.

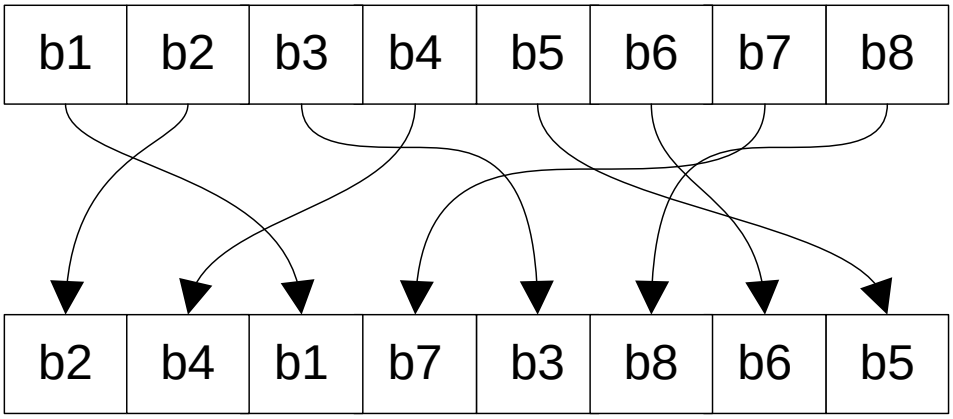



Рисунок 2.5 – Приклад перестановки байтів

Значення з файлу початкового зображення переносяться в масив байтів, в певному порядку, який задає розроблений генератор псевдовипадкових чисел відповідно до заданого ключа К1, операції над яким виконуються з допомогою регістру зсуву з лінійним зворотнім зв’язком.

Крім шифрування зображення представленим раніше методом також необхідно відповідно до методу розділити зображення на 3 рівні частини, для цього необхідно створити опис методу розподілу секрету, про нього в наступному розділі.

## 2.3 Метод розподілу секрету

Сутність данного методу полягає в тому, щоб зображення було поділено між учасниками на 3 рівні частини. Це необхідно для того, щоб можливість

розшифрування було тільки після того, як кожен з учасників надасть свій шматок зображення. Завдяки регістру зсуву з лінійним зворотнім зв’язком, був згенерований параметр, який відповідає за розподіл байтів.

Алгоритм роботи методу зображено на рис. 2.5.

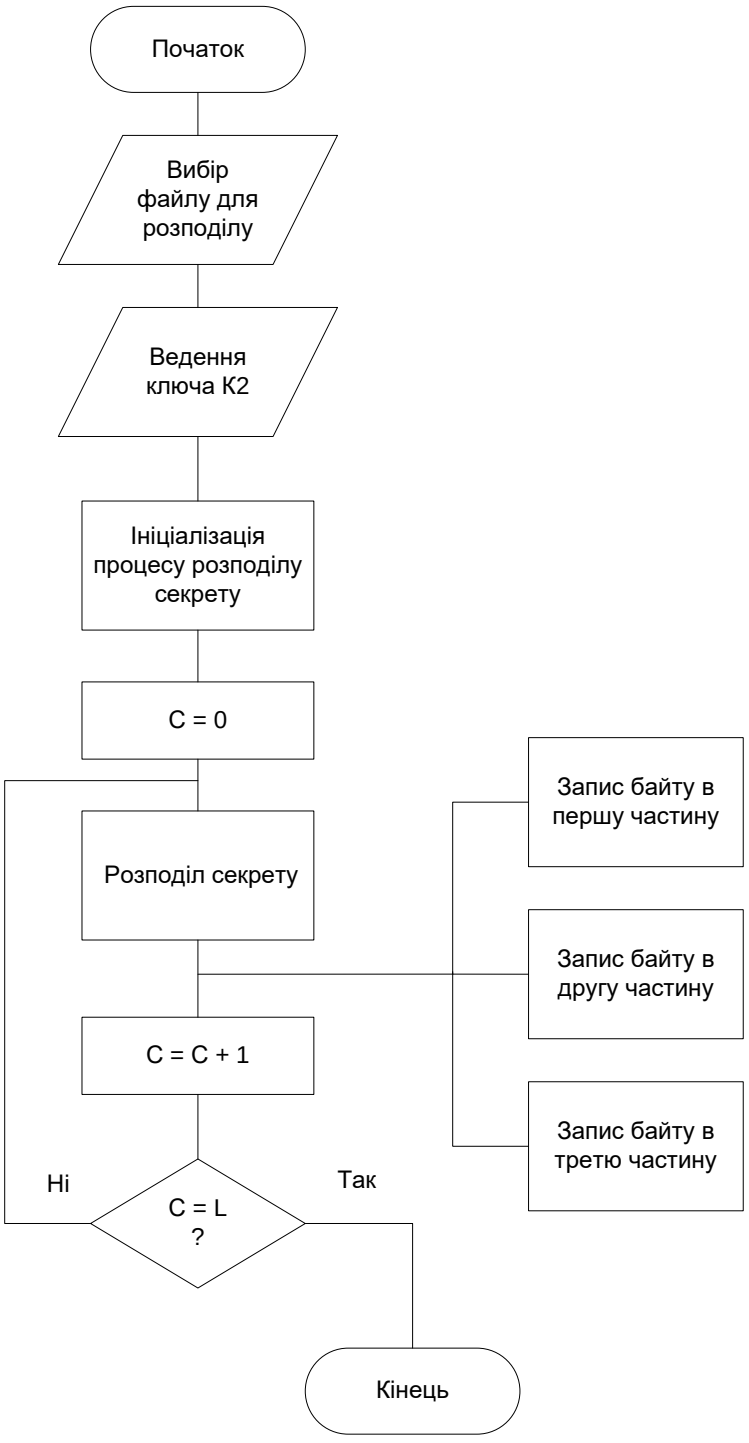



Рисунок 2.6 – Алгоритм розподілу секрету

Для початку необхідно вибрати файл, який необхідно розподілити, в нашому випадку це буде масив байтів отриманий в результаті шифрування. Після того коли отримується параметр, який генерує регістр зсуву з лінійним зворотнім зв'язком, відбувається розділення значень файлу на три частини відповідно певного заданого параметру, а саме: коли два останні значення на кінці регістру будуть рівні 00, тоді значення з файлу запишется в третю частину, коли значення будуть рівні 01, тоді запишеться в першу частину частину, 10 в другу частину, 11 в третю частину відповідно. Саме відповідно до таких параметрів буде відбуватися розподіл байтів на кілька частин.

## 2.4 Метод генерування ПВЧ

Для того, щоб виконувати операції перестановок пікселів в зображенні необхідно розробити генератор, який буде генерувати псевдовипадкові числа без повторень в певному заданому діапазоні.

Є певна кількість значення, які необхідно переставити, довжина рівна $N$ і пронумеровані вони від 0 до $N-1$ та розташовані в природному порядку. Необхідно порядок розташування цих значень і це працює наступним чином. [10]

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

G – множина функцій формування псевдовипадкових послідовностей:

$$G = \{G_1, G_2, G_3, G_4\},$$

де:

$G_1$ – функція формування підпослідовностей;

$G_2$ – функція формування кількості чисел $q_i$ у підпослідовностях;

$G_3$ – функція формування номеру підпослідовності, для випробування;

$G_4$ – функція вибору числа (індексу) з підпослідовності;

$F$ – множина правил перестановок;

$p$ – множина перестановок у підпослідовностях.

$I$ – індикатор перестановки;

Послідовність з $N$ чисел розбивається на $k$ підпослідовностей, тобто:

$$N = \sum_{i=0}^{k-1} N_i$$

Кількість учасників у підпослідовностях може бути як однаковою, так і різною. Якщо у підпослідовностяз кількість чисел буде однаковою тоді для їх формування буде використовуватися функція $G_1$. Також можливо два варіанти, які формують підпослідовності. Перший із них передбачає, ща завдання кількості підпослідовностей $k$. Виходячи з його значення, обчислюється кількість чисел $q_i$ у підпослідовностях. Якщо число $N$ ділиться на число $k$ без остачі, тоді $k$ підпослідовностей з певною кількістю чисел $q_i = \frac{N}{k}$. У випадку коли результат ділення має остачу, відмінну від нуля, то підпослідовності з 0-ї до $(k - 2)$-ї складатимуться з $q_i = \left]\frac{N}{k}\right[$ чисел, а $(k\text{-}1)$-$a$ підпослідовність – з $N - \left]\frac{N}{k}\right[ \cdot (k-1)$ чисел, де ] [ означає округлення до більшого цілого числа [10].

Другий варіант передбачає необіхність завдання однакової кількості чисел у підпослідовностях $q_i$, яку необхіно вибрати з діапазону значень:

$$q_i = 2 \div \left]\frac{N}{2}\right[$$

Отримані підпослідовності необхідно обчислити за формулою:

$$k = \left]\frac{N}{q_i}\right[.$$

Таким чином, підпослідовності з 0-ї до (k – 2)-ї складатимуться з $q_i$ чисел, а (k – 1)-а підпослідовність – *з* $(N - q \cdot (k - 1))$ чисел. У випадку різної кількості чисел у підпослідовностях реалізація функції $G_2$ передбачає виконання таких дій. Кількість чисел у кожній окремій підпослідовності $q_i$ формується за допомогою деякого генератора ПВЧ. При цьому використовується операція підрахунку кількості сформованих підпослідовностей за таким алгоритмом. На початку кількість послідовностей $k := 0$ [10].

Деякому параметру $R$ присвоюється (як початкове) значення кількості чисел $N$: $R_0 := N$ .

З кожним із наступних кроків формування підпослідовностей (отримання чергового значення кількості чисел $q_i$) значення параметру $R$ зменшується на цю кількість:

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

Для того, щоб процес послідовнестей тривав, необхідно щоб виконувалася умова:

$$R > 0.$$

Кожен наступний крок завершується тим, що кількість підпослідовностей збільшується на одиницю:

$$k := k + 1.$$

У тому випадку, коли $R < 0$ , тоді кількість чисел в останній із підпослідовностей буде рівна $R_k$. Недивлячись на те, яким із способів здійснюється розбиття, щоб обчислите перше число підпослідовності необхідно скористанись формулою:

$$n_{0_i} = \sum_{j=0}^{i} q_i - 1.$$

У першому випадку вибір здійснюється із постійним значенням кроку. Вхідними параметрами при цьому є величина кроку $a$ та зміщення $b$. Номер наступної підпослідовності обчислюється за формулою:

$$n_i = (a \cdot i + b) mod\ k, i = 0,1, \dots, (k-1).$$

У разі псевдовипадкового порядку вибору підпослідовностей значення їх номерів формує деякий генератор ПВЧ. При детермінованому порядку вибору чисел із підпослідовностей їх номери обчислюються за формулою:

$$n_{ij} = (a \cdot i + b) mod\ q_i, \qquad i = 0,1, \dots, (k-1); j = 0,1, \dots, (q_i - 1)$$

При застосуванні псевдовипадкового порядку вибору підпослідовностей необхідно використовувати допоміжний параметр – індикатор перестановки [12]. Індикатор перестановки – це ціле число, яке обчислюється для кожної окремої підпослідовності. Його початкове значення дорівнює кількості чисел даної підпослідовності:

$$I_i := q_i.$$

При кожному виборі числа з підпослідовності $N_i$ її індикатор перестановки зменшується на 1. Коли значення індикатора досягає нуля, дана підпослідовність не використовується для вибору чисел. На основі окремих значень індикаторів перестановки формується значення загального індикатора, який визначається за формулою [10]:

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

Коли значення індикаторів перестановки усіх підпослідовностей досягнуть нуля, значення загального індикатора також буде дорівнювати нулю і це свідчитиме про завершення процесу формування перестановок. [10].

Отже, процес формування перестановок складається з трьох основних етапів1:

– розбиття вхідної послідовності на підпослідовності;
– вибір підпослідовностей;
– вибір чисел із підпослідовності.

Для кращого розуміння методів, розберемо приклад роботи власного методу для генерування псевдовипадкових чисел.

В таблиці 2.1 зображено, яким чином лічильника присвоюється максимальне, або мінамальне значення відносно ключа зашифрування К3.

Таблиця 2.1 – Присвоєння значенів лічильників

| Ключ К3 = 0110 | | | |
|---|---|---|---|
| Лічильник 0 | 0 | min | + |
| Лічильник 1 | 1 | max | - |
| Лічильник 2 | 1 | max | - |
| Лічильник 3 | 0 | min | + |

В таблиці 2.2 зображено, яким чином відбувається заміна значень в лічильниках відповідно до ключа зашифрування К3.

Таблиця 2.2 – Заміна значень відносно ключа К3

| К3 | Ліч.0 | Ліч.1 | Ліч.2 | Ліч.3 |
|---|---|---|---|---|
| 00 | x + 1 | x | x | x |
| 01 | x | x – 1 | x | x |
| 10 | x | x | x – 1 | x |
| 11 | x | x | x | x + 1 |

Схему алгоритму формування псевдовипадкових чисел наведено на рис 2.4

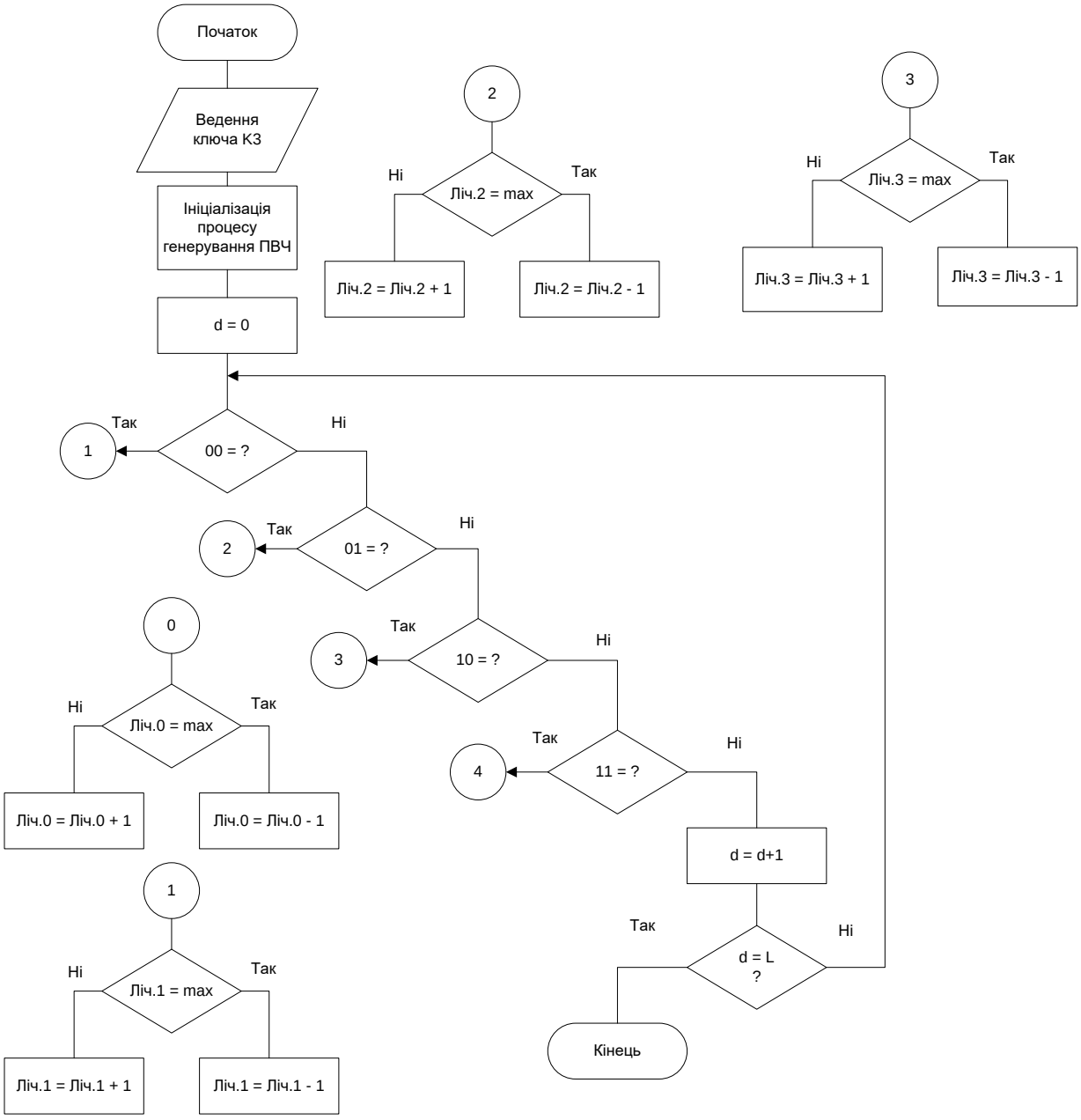



Рисунок 2.7 – Схема алгоритму формування псевдовипадкових чисел

Відповідно до двох останніх значень значень на виході регістру зсуву з зворотнім зв’язком, який використовується для формування значень на вхід генератора ПВЧ, відповідно до ключа, який використовується при зашифруванні, вибирається те який лічильник буде використовуватися:

– якщо 00, то Лічильник 0;
– якщо 01, то Лічильник 1;
– якщо 10, то Лічильник 2;
– якщо 11, то Лічильник 3.

Після вибору лічильника, наступною дією буде заміна в ньому значення з кроком + 1, при умові що значення в лічильнику було мінімальним, або – 1 коли значення в лічильнику було максимальним.

Нехай ми маємо псевдовипдкову послідовність: 01100010110100100111100010101, яка має 31 значення, відповідно до методу розбиваємо ці значення на 4 лічильника.

Початкову послідовність ділимо на рівні 4 частини (рис. 2.6).

| Лічильник 0 | | | Лічильник 1 | | | Лічильник 2 | | | Лічильник 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ……. | 7 | 8 | …… | 15 | 16 | ……. | 23 | 24 | ……... | 31 |

Рисунок 2.6 – Вхідна послідовність генератора з чотирма лічильниками

Відповідно до розрядів секретного ключа, визначаються початкові стани лічильника та початкові виконуванні операції.

Нехай значення ключа буде рівним «0110» відповідно до цього початкові стани лічильників будуть мати такі значення:

– Лічильник 0 = 0;
– Лічильник 1 =15.
– Лічильник 2 =23;
– Лічильник 3 =24.

Послідовність дій, поточні значення станів лічильників і псевдовипадкова послідовність чисел показано в таблиці 2.3

Таблиця 2.3 – Послідовність дій генератора ПВЧ

| Крок | Вхід | Ліч. 0 | Ліч.1 | Ліч. 2 | Ліч. 3 | Вихіл |
|---|---|---|---|---|---|---|
| 1 | 01 | 0 | 15<br>15-1=14 | 23 | 24 | 15 |
| 2 | 11 | 0 | 14 | 23 | 24<br>24+1 | 24 |
| 3 | 00 | 0<br>0+1 | 14 | 23 | 25 | 0 |
| 4 | 10 | 1 | 14 | 23-1 | 25 | 23 |
| 5 | 01 | 1 | 14<br>14-1 | 22 | 25 | 14 |
| 6 | 11 | 1 | 13 | 22 | 25<br>25+1 | 25 |
| 7 | 00 | 1<br>1+1 | 13 | 22 | 26 | 1 |
| 8 | 10 | 2 | 13 | 22<br>22-1 | 26 | 22 |
| 9 | 01 | 2 | 13<br>13-1 | 21 | 26 | 13 |
| 10 | 10 | 2 | 12 | 21<br>21-1 | 26 | 21 |
| 11 | 11 | 2 | 12 | 20 | 26<br>26+1 | 26 |
| 12 | 00 | 2<br>2+1 | 12 | 20 | 27 | 2 |
| 13 | 01 | 3 | 12<br>12-1 | 20 | 27 | 12 |
| 14 | 00 | 3<br>3+1 | 11 | 20 | 27 | 3 |
| 15 | 11 | 4 | 11 | 20 | 27<br>27+1 | 27 |
| 16 | 10 | 4 | 11 | 20<br>20-1 | 28 | 20 |
| 17 | 11 | 4 | 11 | 19 | 28<br>28+1 | 28 |
| 18 | 00 | 4<br>4+1 | 11 | 19 | 29 | 4 |
| 19 | 01 | 5 | 11<br>11-1 | 19 | 29 | 11 |
| 20 | 10 | 5 | 10 | 19<br>19-1 | 29 | 19 |
| 21 | 00 | 5<br>5+1 | 10 | 18 | 29 | 5 |
| 22 | 10 | 6 | 10 | 18<br>18-1 | 29 | 18 |
| 23 | 00 | 6<br>6+1 | 10 | 17 | 29 | 6 |
| 24 | 01 | 7 | 10<br>10-1 | 17 | 29 | 10 |

Продовження таблиці 2.2.

| | | | | | | |
|---|---|---|---|---|---|---|
| 25 | 11 | 7 | 8 | 17 | 29<br>29+1 | 29 |
| 26 | 00 | 7<br>stop | 9 | 17 | 30 | 7 |
| 27 | 10 | | 9 | 17<br>17-1 | 30 | 17 |
| 28 | 11 | | 9 | 16 | 30<br>30+1 | 30 |
| 29 | 01 | | 9<br>9-1 | 16 | 31 | 9 |
| 30 | 11 | | 8 | 16 | 31<br>stop | 31 |
| 31 | 10 | | 8 | 16<br>stop | | 16 |
| 32 | 01 | | 8<br>stop | | | 8 |

В даному розділі було розглянено розроблений метод для генерування псевдовипадкових чисел, який працює на базі регістру звусу з лінійним зворотнім зв’язком.

# 3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАСОБУ

## 3.1 Вибір середовища програмування для розроблення засобу

Метою дипломної кваліфікаційної роботи є створення програмного засобу, для захисту секретного вмісті зображення. Відповідно, щоб це реалізувати потрібно серйозно віднестись до вибору мови програмування. Також не менш важливим для розробки є вибір середовища, яке буде використано для написання ті налагодження програмного засобу.

Розроблений програмний засіб було реалізовано у середовищі розробки Microsoft Visual Studio 2022. Також були використані API та декілька власних функцій.

Visual Studio це потужний засіб розробника, який можна використовувати для єдиного завершення всього циклу розробки. Це комплексне інтегроване середовище розробки (IDE), яке можна використовувати для написання, редагування, налагодження та складання коду, а потім для розгортання програми. Крім редагування та налагодження коду, Visual Studio включає компілятори, засоби завершення коду, систему керування версіями, розширення та багато інших функцій для покращення кожного етапу процесу розробки програмного забезпечення. [11]

Інтегроване середовище розробки Visual Studio надає безліч функцій , що спрощують написання коду та управління ним. Наприклад, є можливість швидко і точно використовувати пропозиції щодо коду IntelliSense, швидко покращити код за допомогою лампочок, що пропонують дії, або розгорнути або згорнути блоки коду за допомогою структурування. Організовувати та вивчати код за допомогою браузера рішень, який показує код, впорядкований за файлами, або представлення класів, в якому відображається код, впорядкований за класами. [11]

Можна компілювати та створювати програми, щоб негайно створювати збірки та тестувати їх у налагоджувачі. Для проектів C++ та C# можна запускати багатопроцесорні зборки. Visual Studio також надає кілька параметрів, які можна

налаштувати під час створення програм. Можна створити конфігурацію збірки, що настроюється, на додаток до вбудованих конфігурацій, приховати певні попереджувальні повідомлення або збільшити вихідні дані збірки. [11]

Вбудована налагодження в Visual Studio дозволяє легко виконувати налагодження, профіль та діагностику. Розробник виконує покрокове виконання коду і переглядаєте значення, що зберігаються в змінних, задає контрольні значення для змінних, щоб побачити, коли значення змінюються, перевіряється шлях виконання коду та інші способи налагодження коду під час його виконання. [11]

Наступна можливість писати високоякісний код за допомогою комплексних засобів тестування Visual Studio. Модульні тести дозволяють розробникам та тестувальникам швидко знаходити логічні помилки у коді. Також можна проаналізувати, скільки коду тестується, і побачити миттєві результати в наборі тестів або дізнатися про вплив кожної зміни, яка вноситься за допомогою розширених функцій, які тестують код під час введення. [11]

За допомогою вбудованих функцій Git Visual Studio можна клонувати, створювати або відкривати власні репозиторії. Вікно інструментів Git містить все необхідне для фіксації та відправки змін до коду, керування гілками та вирішення конфліктів злиття. Якщо у вас є обліковий запис GitHub, ви можете керувати цими репозиторіями безпосередньо у Visual Studio. [11]

При написанні програмного засобу, використовувалися функції Windows API. Дані функції містяться в бібліотеках динамічного завантаження (Dynamic Link Libraries, або DLL), оскільки вони складаються з великого числа функцій, які завантажуються в пам’яті тільки в момент, коли до них відбулося звернення. Відповідно завдяки цьому вони полегшують роботи розробнику і дозволяють заощадити вільну пам'ять на диску. [11]

Вони є найпрямішим із способів для взаємодії додатків з операційної системи Windows. При використанні вище згаданих функцій, можна швидше і простіше запрограмувати інтерфейс програмного засобу, адже це уже готові функції, які необхідно просто викликати.

3.2 Розроблення схеми функціонування програми

Засіб для шифрування зображення буде працювати згідно алгоритму, що зображений на рис. 2.1.

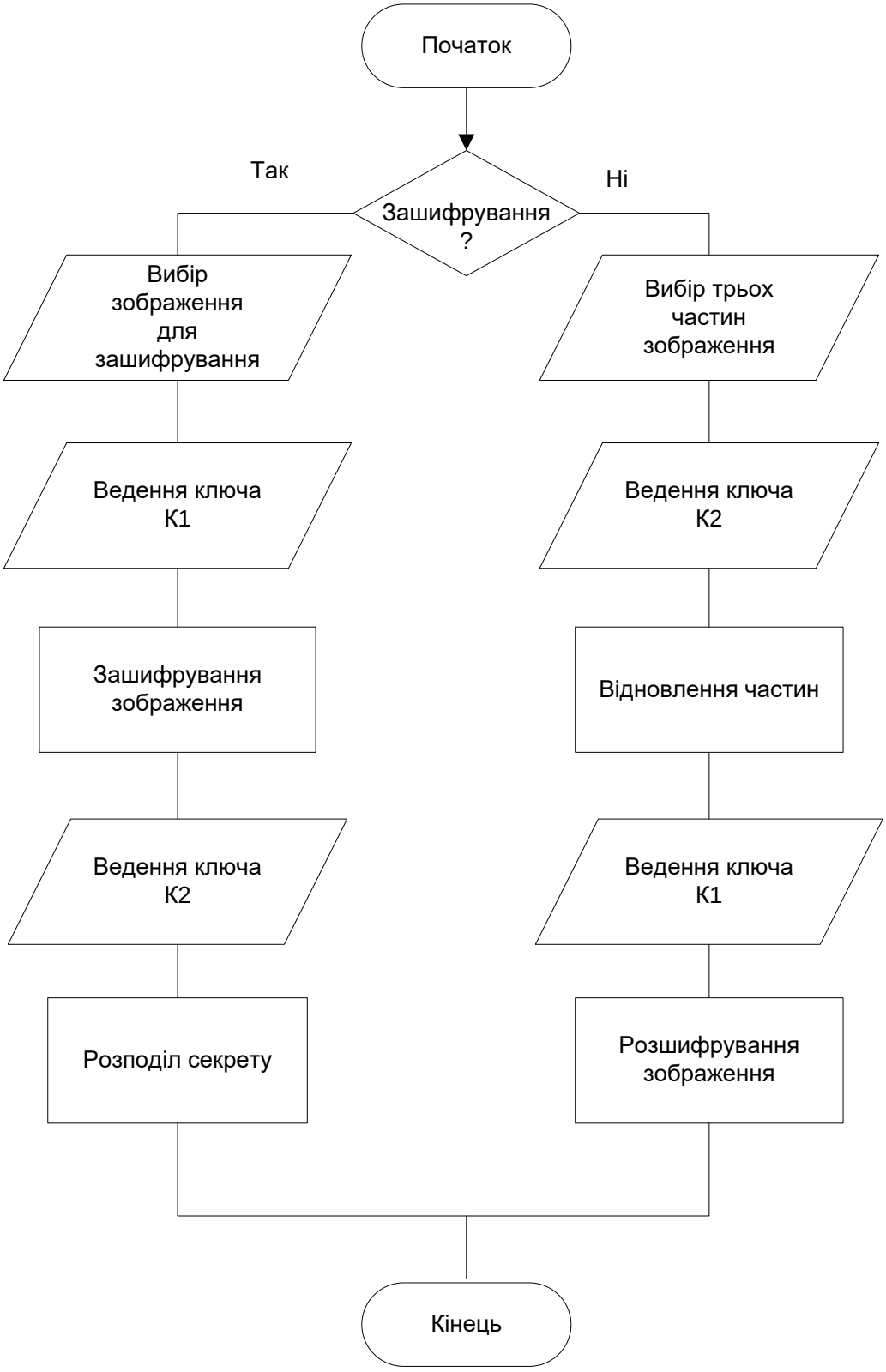



Рисунок 3.1 – Алгоритм роботи програмного засобу

Розглянувши основні підходи для захисту зображення, було розроблено власний метод, який працює у двох режимах, а саме:

– Режим зашифрування, який включає такі основні етапи: вибір зображення, ведення секретного ключа К1, зашифрування, ведення ключа К2, та етап розподілу секрету. Завдяки використаню одразу двох підходів захисту зображення, вдалося в значній мірі покращити криптографічну стійкість методу;

– Режим розшифрування включає в себе збір інформації від учасників, а саме трьох зашифрованих зображень та значення приватного ключа К2, щоб відновити частини, та значення приватного ключа К1, щоб розшифрувати відновлену частину.

Відповідно розробленого методу для того, щоб розшифрувати зображення усі учасники повинні бути присутніми, якщо хоча б хтось один буде відсутній, відновлення частин не відбудеться і відповідно подальше розшифрування буде неможливим, також відповідно без вірних значень обох ключів, або невідповідність ключів діям для яких вони використовується, зображення не розшифрується.

## 3.3 Програмний засіб зашифрування зображення

### 3.3.1 Реалізація відкриття BMP файлу

Найпершим кроком необхідно отримати шлях до зображення, яке необхідно шифрувати і відкрити до цього зображення текстовий потік

```
FILE *pFile = fopen(szFileOne, "rb");
```

Тут szFileOne це шлях до зображення, який отримується з стандартної панелі відкриття файлів.

Після того, як відбулося відкриття потрібного файлу, відбувається його читаття, та створюється структура

```
BITMAPFILEHEADER header;
```

в яку запишеться інформаційний заголовок файлу:

```
header.bfType      = read_u16(pFile);
header.bfSize      = read_u32(pFile);
header.bfReserved1 = read_u16(pFile);
header.bfReserved2 = read_u16(pFile);
header.bfOffBits   = read_u32(pFile);
```

де буде записано основну інформацію про файл, а саме його тип та розмір. Потім в створену структуру

```
BITMAPINFOHEADER bmiHeader;
```

запишеться заголовок зображення з інформацією про нього

```
bmiHeader.biSize          = read_u32(pFile);
bmiHeader.biWidth         = read_s32(pFile);
bmiHeader.biHeight        = read_s32(pFile);
bmiHeader.biPlanes        = read_u16(pFile);
bmiHeader.biBitCount      = read_u16(pFile);
bmiHeader.biCompression   = read_u32(pFile);
bmiHeader.biSizeImage     = read_u32(pFile);
bmiHeader.biXPelsPerMeter = read_s32(pFile);
bmiHeader.biYPelsPerMeter = read_s32(pFile);
bmiHeader.biClrUsed       = read_u32(pFile);
bmiHeader.biClrImportant  = read_u32(pFile);
```

де буде міститись уся основна інформація про зображення, яка буде необхідна для подальшої роботи з ним.

Після того, як було отримано всю необхідну інформацію про зображення, розпочинається зчитування його основної інформації, для чого необхідно створити масив структур

```
RGBQUAD *pixels = new RGBQUAD[bmiHeader.biHeight * bmiHeader.bi-Width];
```

розмірність якого міє бути, висота зображення помножена на ширину, це дозволить визначити які кількість пікселів у всьому зображенні.

Наступним кроком в циклі, який має довжину в кількість пікселів всього зображення, зчитуються усі кольори та записуються в схему, яка було описана раніше.

```
for (int i = 0; i < bmiHeader.biHeight * bmiHeader.biWidth; i++)
{
     pixels[i].rgbBlue = getc(pFile);
     pixels[i].rgbGreen = getc(pFile);
     pixels[i].rgbRed = getc(pFile);
 }
```

Після чого відбувається закривання потоку до зображення

```
fclose(pFile);
```

З допомогою усіх вищеописаних команд, вдалося відкрити та прочитати увесь зміст зображення, дізнатися про нього усю інформацію та отримати значення кольорів його пікселів.

### 3.3.2 Реалізація перестановок пікселів

Найпершим кроком необхідно для генератора ПВЧ, згенерувати псевдовипадкову послідовність, для цього потрібно програмно реалізувати генератор Галуа.

Для його реалізації необхідно згенерувати 8 розрядний початковий стан та отримати потрібний поліном.

```
int Sj[8]={1,1,1,1,0,0,0,1}, P[8]={0,0,0,1,1,1,0,0};
```

Наступним кроком необхідно створити двовимірний масив, який має 255 рядків довжиною у 8 розрядів для того, щоб зберегти отримані стани.

```
const int N=254;
int Sj1[N][8];
```

Далі в циклі, який має довжину в кількість з генерованих елементів, окремо запам’ятовується старший розряд, а далі циклічно зсувається весь масив.

```
K=Sj[0];//запамятувомо старший розряд
buf=Sj[0];//зсув
Sj[0]=Sj[1];
Sj[1]=Sj[2];
Sj[2]=Sj[3];
Sj[3]=Sj[4];
Sj[4]=Sj[5];
Sj[5]=Sj[6];
Sj[6]=Sj[7];
Sj[7]=buf;
```

Наступним кроком перевіряється чи рівний старший розряд  одиниці, якщо так, тоді виконується операція XOR з поліномом і станом регістру.

```
if(K==1)
{
for (int counter = 0; counter < 8; counter++)
{
   Sj[counter]=Sj[counter]^P[counter];///XOR
}

}
```

У тому випадку коли не дорівню одині, тоді ця частина виконуватися не буде. Кожен із станів регістру записується в двовимірний масив, який було створено, ще раніше.

```
for (int i=0; i<8; i++)
Sj1[count][i] = Sj[i];
```

Отже після усіх ітерацій, було отримано масив станів, який не повторюється.

Для генератора псевдовипадкових чисел, необхідно виконати перетворення над раніше створеним двовимірним масивом для перетворення його в одновимірний шляхом виконання наступних кроків.

```
int a[2040];
for(int i=0;i<=255;i++)
{
	for(int j=0;j<=8;j++)
	{
		a[i*8+j] = Sj1[i][j];
	}
}
```

Для того, щоб генератора міг працювати необхідно створити усі необхідні лічильники, масив позицій розмірністю в кількість пікселів, та отримати значення про кількість пікселів в зображення, яке отримується з його заголовку.

```
unsigned long int count = bmiHeader.biHeight * bmiHeader.biWidth;
int * aray = new int[bmiHeader.biHeight * bmiHeader.biWidth];
unsigned long int x = count/2;
unsigned long int L1 = 0;
unsigned long int L2 = x+1;
unsigned long int L3 = x;
unsigned long int L4 = count
```

Реалізований генератора використовує два лічильника в генеруванні позицій. Лічильний 0 який працює від $N_0{}^{\min}$ до $N_0{}^{\max}$, та Лічильник 1 який працює від $N_1{}^{\min}$, до $N_1{}^{\max}$..

До того часу поки усі лічильники не досягнуть всього кінцевого стану , генерація позицій буде відбуватися в циклі. Код розробленого генератора наведено нище..

```
for (; ;)
	{
		if(x1==true && x2==true)
		break;

		if(j==sizeof(a)/sizeof(int))
				j=0;
				if(x2==true)goto m2;
				if (a[j]==1 && x1==false)
			{
m2:				aray[i]=L3;
				i++;
				aray[i]=L1;
				i++;
				j++;
				L1=L1+1;
				L3=L3-1;
				if(L1>=L3)
				{
					if(L1==L3)
					{
```

```
                                    aray[i]=L1;
                                    i++;
                              }
                              x1=true;
                        }


                  }

                        if(x1==true)goto m1;
                        if (a[j]==0 && x2==false)
                  {

m1:                     aray[i]=L4;
                        i++;
                        L4=L4-1;
                        aray[i]=L2;
                        L2=L2+1;
                        i++;
                        j++;
                        if(L2>=L4)
                        {
                              if(L2==L4)
                              {

                                    L2=L4;
                                    aray[i]=L2;
                                    i++;
                              }
                                    x2=true;
                        }
                  }
      }
```

Розроблений генератор псевдовипадкових чисел дозволив згенерувати нову позицію для кожного піклселя.

3.3.3 Розділення секрету

Для початку необхідно створити три файли з розширенням .bmp, для можливості подальшого відновлення, в заголивки яких записується вся інформація з початкового зображення.

```
FILE *oFile = fopen("output10.bmp", "wb");
FILE *oFile2 = fopen("output20.bmp", "wb");
FILE *oFile3 = fopen("output30.bmp", "wb");
// записываем заголовок файла
write_u16(header.bfType, oFile);
write_u32(header.bfSize, oFile);
write_u16(header.bfReserved1, oFile);
write_u16(header.bfReserved2, oFile);
write_u32(header.bfOffBits, oFile);
 // записываем заголовок файла 2
 write_u16(header.bfType, oFile2);
write_u32(header.bfSize, oFile2);
write_u16(header.bfReserved1, oFile2);
write_u16(header.bfReserved2, oFile2);
```

```
write_u32(header.bfOffBits, oFile2);
 // записываем заголовок файла 3
write_u16(header.bfType, oFile3);
write_u32(header.bfSize, oFile3);
 write_u16(header.bfReserved1, oFile3);
write_u16(header.bfReserved2, oFile3);
write_u32(header.bfOffBits, oFile3);
```

Завдяки цьому на виході буде створено три однакові зображення по розширеню, але які мають різний вміст пікселів.

Необхідно в циклі який має довжину в кількість пікселів зображення, в кожний файл записати, якусь складову його оригінального кольору.

```
for(int i=0;i<bmiHeader.biHeight * bmiHeader.biWidth;i++)
{
      modul = mod(i);
      putc(pixels[aray[i]].rgbBlue, oFile);
      putc(modul, oFile);
      putc(modul, oFile);

      modul = mod(i);
      putc(modul, oFile2);
      putc(pixels[aray[i]].rgbGreen, oFile2);
      putc(modul, oFile2);

      modul = mod(i);
      putc(modul, oFile3);
      putc(modul, oFile3);
      putc(pixels[aray[i]].rgbRed, oFile3);
 }
```

Так як в створеному зображенні є уже тільки одна складова пікселя, інші дві необхідно теж заповнити, для йього була створена функція mod() вхідним параметром якої є номер позиції пікселя. Від значення яке було отримано необхідно відняти 256, до тих пір поки це значення не стане меншип 256. Результатом буде те, що функція поверне значення що потрапляє в діапазон можливих значень кольору, а саме від 0 – 255. В подальшому це значення буде записуватися в інші складові пікселя.

```
unsigned long int mod (unsigned long int pos)
{
      for(;;)
      {
            if(pos>255)
            {
                  pos=pos-256;
            }
            else break;
      }
      return pos;
}
```

```
fclose(oFile);
      fclose(oFile2);
fclose(oFile3);
```

Після завершення усіх операцій, усі файли закриваються, і будуть готові для передачі усім із учасників розподілу секрету.

### 3.4 Програмний засіб розшифрування зображення

Процедура розшифрування буде виконуватися обернено до процедури зашифрування, спочатку з панелі вибору елементів отримуються, шляхи, до усіх файлів та їхні назви, файлів що були зашифровані

```
nFile = fopen(str1, "rb");
nFile2 = fopen(str2, "rb");
nFile3 = fopen(str3, "rb");
```

де str1,str2,str3 – це імена та шляхи до зображень.

Відкриття та читання файлів відбувається аналогічно до етапу зашифрувнання, з винятком, що туж процедуру необхідно провести з тьома файлами.

На наступному кроці створюються, три масиви структур, що будуть містити данні з кольорами трьох зображень. Так як кількість ппікселів у всіх зображенях однакова, то відповідно і розмір для них заданий однаковий.

```
pixels1 = new RGBQUAD[bmiHeader.biHeight * bmiHeader.biWidth];
pixels2 = new RGBQUAD[bmiHeader.biHeight * bmiHeader.biWidth];
pixels3 = new RGBQUAD[bmiHeader.biHeight * bmiHeader.biWidth];
```

Генерацію нових позицій здійснено аналогічно до раніше створеного алгоритму.

Процес зчитування файлів здійснюється в порядку, який визначений генератором. В кожен піксель, номер якого визначено генератором записується значення із файлу.

```
for(int i=0;i < bmiHeader.biHeight * bmiHeader.biWidth;i++)
      {
            pixels1[aray[i]].rgbBlue = getc(nFile);
         pixels1[aray[i]].rgbGreen = getc(nFile);
         pixels1[aray[i]].rgbRed = getc(nFile);

            pixels2[aray[i]].rgbBlue = getc(nFile2);
         pixels2[aray[i]].rgbGreen = getc(nFile2);
         pixels2[aray[i]].rgbRed = getc(nFile2);

            pixels3[aray[i]].rgbBlue = getc(nFile3);
```

```
            pixels3[aray[i]].rgbGreen = getc(nFile3);
            pixels3[aray[i]].rgbRed = getc(nFile3);
        }
```

Відповідно на даному етапі уже створено три проміжні зображення, в яких правильне розташування пікселів, але поки що значення кольорів розподілено по них трьох.

Наступним кроком є створення нового файлу із розширенням .bmp в який буде записано уже розшифроване зображення.

```
xFile = fopen("output1_1.bmp", "wb");

write_u16(header.bfType, xFile);
write_u32(header.bfSize, xFile);
write_u16(header.bfReserved1, xFile);
write_u16(header.bfReserved2, xFile);
write_u32(header.bfOffBits, xFile);

write_u32(bmiHeader.biSize, xFile);
write_s32(bmiHeader.biWidth, xFile);
write_s32(bmiHeader.biHeight, xFile);
write_u16(bmiHeader.biPlanes, xFile);
write_u16(bmiHeader.biBitCount, xFile);
write_u32(bmiHeader.biCompression, xFile);
write_u32(bmiHeader.biSizeImage, xFile);
write_s32(bmiHeader.biXPelsPerMeter, xFile);
write_s32(bmiHeader.biYPelsPerMeter, xFile);
write_u32(bmiHeader.biClrUsed, xFile);
write_u32(bmiHeader.biClrImportant, xFile);
```

Заголовок одного із зображень, що надійшли для об’єднання повністю копіюються в розшифроване зображення

Вміст пікселів створюється з допомогою наступних команд.

```
for (int i = 0; i < bmiHeader.biHeight * bmiHeader.biWidth; i++) {

            putc(pixels1[i].rgbBlue, xFile);
            putc(pixels2[i].rgbGreen, xFile);
            putc(pixels3[i].rgbRed, xFile);
    }
```

У наступному кроці отримується оригінальна складова із структури кожного із файлів і при запису об’єнується в один файл. Після завершенню циклу файл закриється, вміст файлу в повній мірі відтворює зображення до зашифрування.

### 3.5 Інтерфейс програмного засобу

Однією із досить важливих частин програмного засобу при його створенні, є його інтерфейс, який грає величезну роль для зручності особливо звичайному кори-

стувачеві, інтерфейс передбачає:

– головне вікно;
– інформаційні вікна;
– меню для керування режимами роботи програми;
– додаткові можливості(робота з файлами).

Вигляд головного вікна представлено на рисунку 3.2


Рисунок 3.2 – Головне вікна програми

Технології Visual Stydio дозволили створити усі елементи керування прямо в середовіщі. Для зручності було використано такі додаткові елементи керування як меню (рис.3.3).

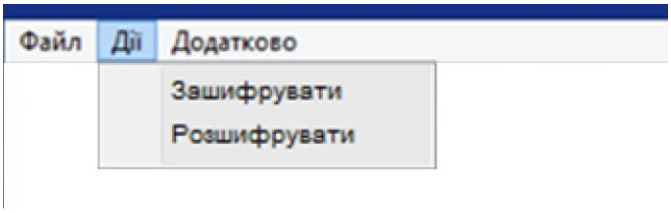



Рисунок 3.3 – Меню головного вікна

Так як при розшифруванні необхідно отримати файл трьох зображень, для цього було реалізовано діалогове вікно ( рис.3.4).

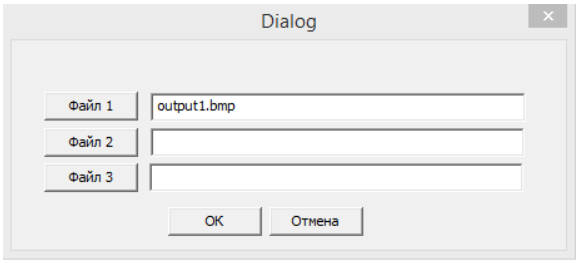



Рисунок 3.4 – Меню вибору файлів для розшифрування

Для виконання пошуку файлу в комп'ютері використовується стандартна панель відкриття файлів Windows (рис.3.5).

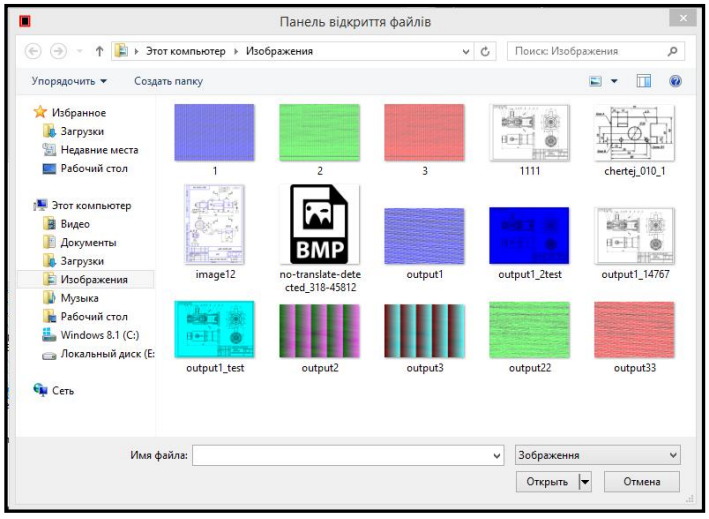



Рисунок 3.5 – Вікно панелі відкриття файлів

Відповідно в кінці було отримано зручний та інтуїтивно зрозумілий інтерфейс для будь якого користувача, задачою якого є реалізація зашифрування зображення, а також його розшифрування.

## 3.6 Результати тестування програмного засобу

Для перевірки роботи засобу, необхідно відкрити зображення, для цього потрібно, в головному меню натиснути на кнопку «Відкрити», у панелі відкриття файлів потрібно вибрати потрібне зображення ( рис.3.6).

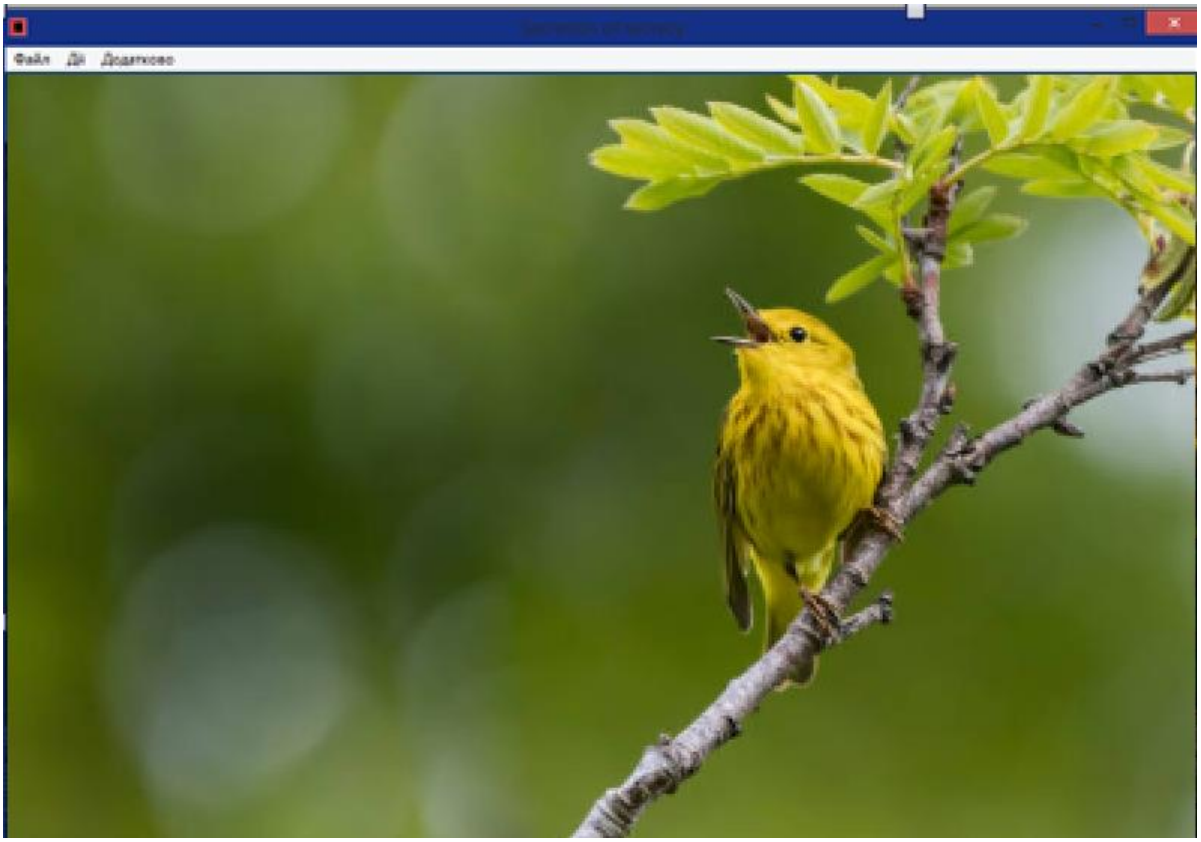



Рисунок 3.6 – Вигляд відкритого зображення

Наступним кроком необхідно в головному меню, у вкладці «Параметри» вибрати пункт «Зашифрувати»

Після виконання зашифрування в папці разом із початковим зображення, мають з’явитися три файли із .bmp розширенням, із вмістом подімним до початкового зображенння (рис.3.8).

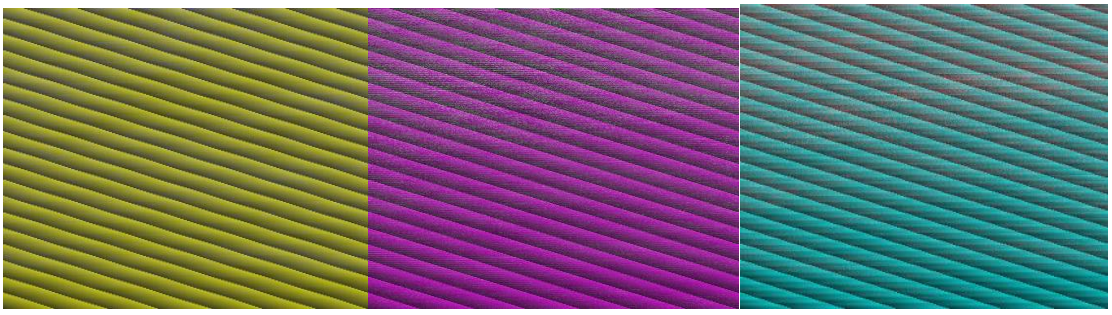


Рисунок 3.8 – Вигляд зашифрованих зображень

Для кращого розуміння можливостей програмного засобу, перевіримо його роботу на другому типі зображення, а саме на чорно-білому ( рис.3.9).

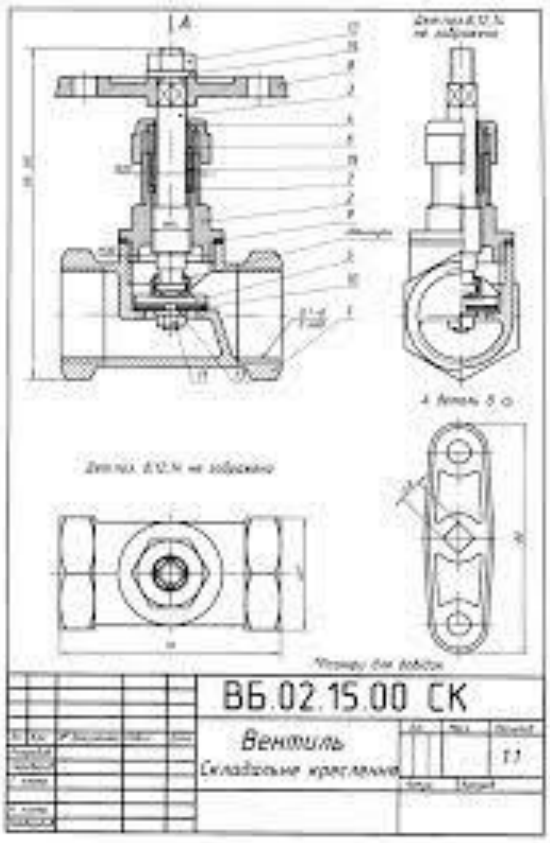



Рисунок 3.9 – Вигляд чорно-білого зображення

Після зашифрування були отримані наступні зображення (рис.3.10).

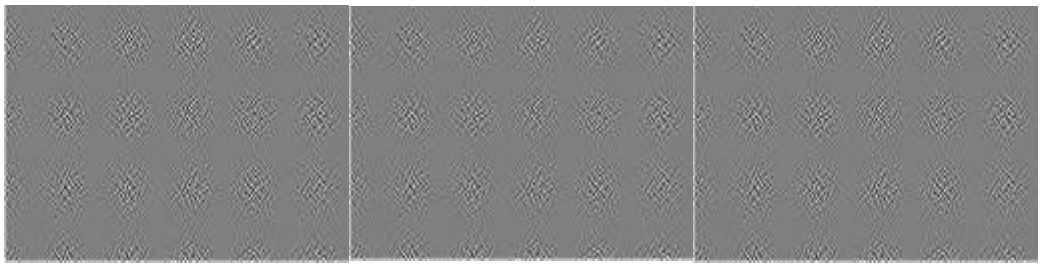


Рисунок 3.10 – Вигляд зашифрованих чорно-білих зображень

Підсумком експерементів є отримані в обох випадках зображення, зміст який далекий від початкового.

Для того, щоб виконати процедуру обернену до попередньої, необхідно у вкладці «Параметри» натиснути на пункт меню «Розшифрувати» та у діалоговому вікні вибрати всі файли необхідні для розшифрування ( рис.3.11).

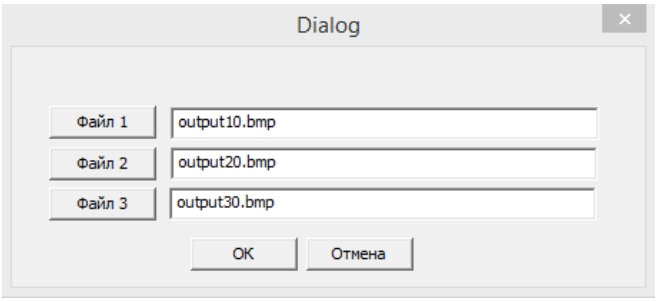



Рисунок 3.11 – Діалогове вікно з вибором файлів для відновлення

Після натиснення «Ок» в папці разом із початковими зображеннями з’явитися відновлене зображення ( рис.3.12).

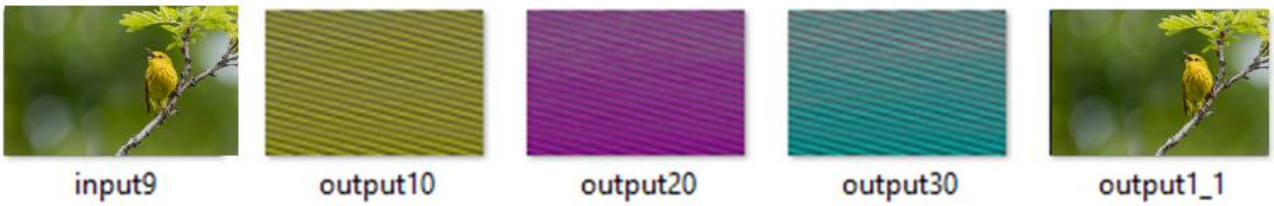



Рисунок 3.12 – Вигляд  каталогу з усіма зображення

Підсумком тестування програмного продукту було, що усі знайдені помилки було усунено. Доведено що програма працює безпомилок і процедури зашифрування та розшифрування відбуваються правильно.

# ВИСНОВОК

Проведений аналіз підходів для захисту зображень показав, що цей захист може бути забезпечений, або шифруванням зображень з використанням відомих блокових шифрів, або шляхом розподілу секрету на кілька частин з використанням певних математичних перетворень. Однак недоліком цих підходів є відносно невелика швидкість перетворень, яка пов’язана з тим що зображення, як правило мають великий обсяг.

В роботі розроблено метод шифрування зображень, який є комбінацією двох процедур: перестановка байтів, яка еквівалентна шифруванню і розподіл секрету шляхом псевдовипадкового порядку формування частин зображення. Оскільки ці процедури є достатньо простими в реалізації, то забезпечене пришвидшення процесу зашифрування зображення, крім того комбінування перестановок та розбиття секрету забезпечує високий рівень криптографічної стійкості.

Тестування розробленого програмного засобу показало коректність розробленого методу. Таким чином усі сформульовані задачі дослідження розв’язано і досягнута мета дослідження.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. D. Jin. Progressive color visual cryptography / D. Jin, W.Q. Yan, M.S. Kankanhalli // Journal Of Electronic Imaging, 2005.— Vol. 14, Issue 3.

2. Naor Moni. Visual cryptography / Naor Moni, Adi Shamir //. Workshop on the Theory and Application of of Cryptographic Techniques. – Springer Berlin Heidelberg, 1994. –

3. Schneier B. Applied Cryptography. Protocols, Algorithms and Source Code in C. – 1996 John Wiley , 1996. – 784 p.

4. Shamir A. How to share a secret // Communications of the ACM — New York City: Association for Computing Machinery, 1979. — Vol. 22, Iss. 11. — P. 612—613.

5. Darrel Hankerson. Guide to Elliptic Curve Cryptography / Darrel Hankerson, Alfred Menezes, Scott Vanstone // 2004, Springer – Verlag New York Inc. – 205 p

6. Young-Chang Hou. Visual cryptography for color images // Department of Information Management, National Central University, Jung Li, Taiwan 320, ROC Received 6 June 2002; accepted 26 August 2002

7. Mehta M. L. Eigenvalues and eigenvectors of the finite Fourier transform (англ.) // Journal of Mathematical Physics — AIP, 1986. — Vol. 28, Iss. 4. — P. 781—785.

8. Огляд основних графічних форматів [Електроний ресурс].— https://studfile.net/preview/8850603/page:2/

9. Бінарне кодування інформації [Електроний ресурс] – https://www.chaynikam.info/ukr/dvoichnoe-kodirovanie.html#photo

10. Лужецький В. А. Метод формування перестановок довільної кількості елементів / В.А. Лужецький, І.С. Горбенко // Захист інформації, том 15, липень-вересень 2013. – №3. – С. 262-265.

11. Що таке Visual Studio [Електроний ресурс] – https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022

55

**Додаток А**

# ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб для шифрування зображень
Автор роботи: Кирилюк Олександр Павлович
Тип роботи: бакалаврська дипломна робота.
Підрозділ кафедра захисту інформації ФІТКІ.

**Показники звіту подібності Unicheck**

Оригінальність – 71,1%. Схожість – 28,9%.
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 _____ Каплун В. А.
(підпис) (прізвище, ініціали)

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

Автор роботи _____ Кирилюк О.П.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Лужецький В.А.
(підпис) (прізвище, ініціали)

56

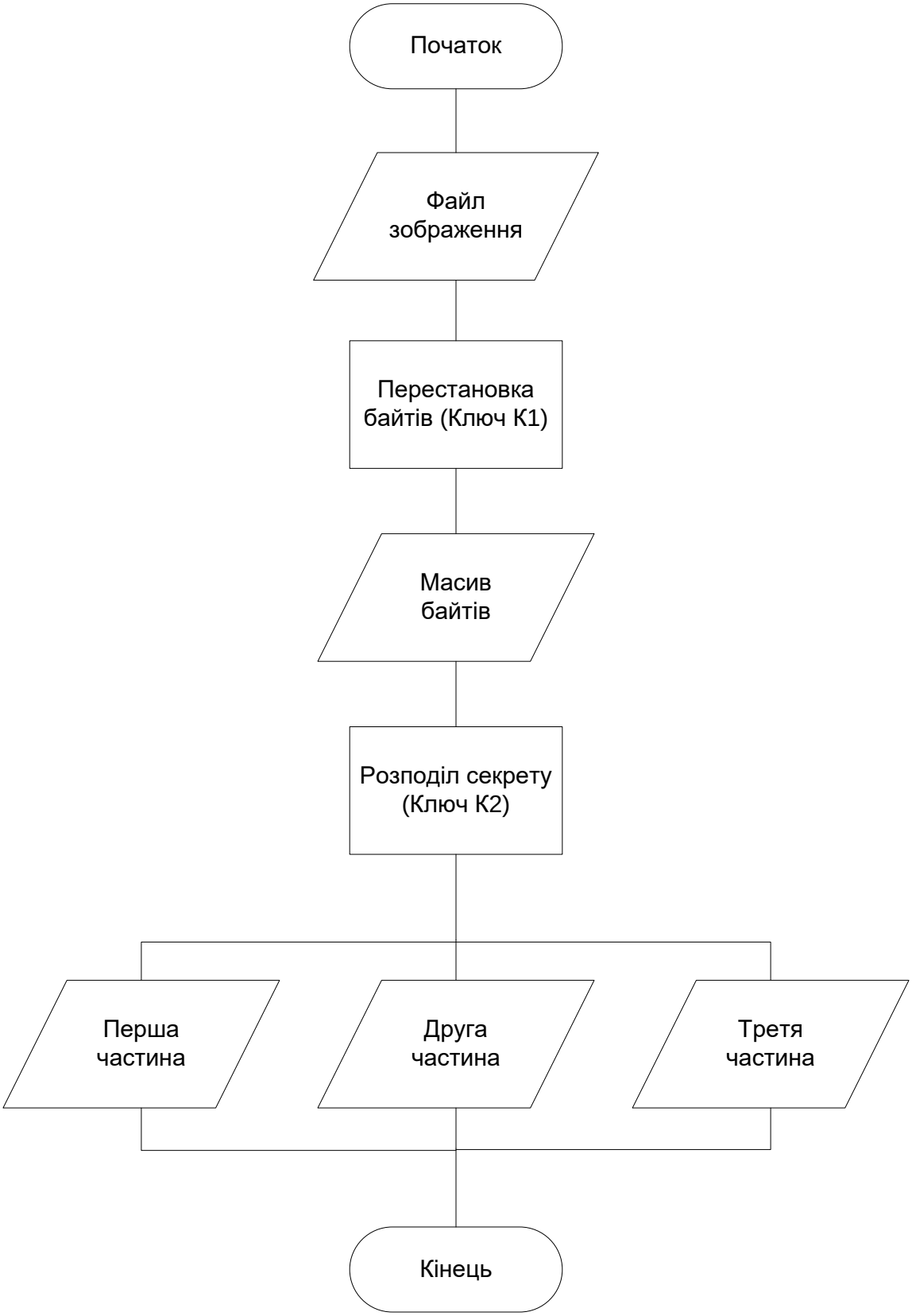
**ІЛЮСТРАТИВНА ЧАСТИНА**

Засіб для шифрування зображень

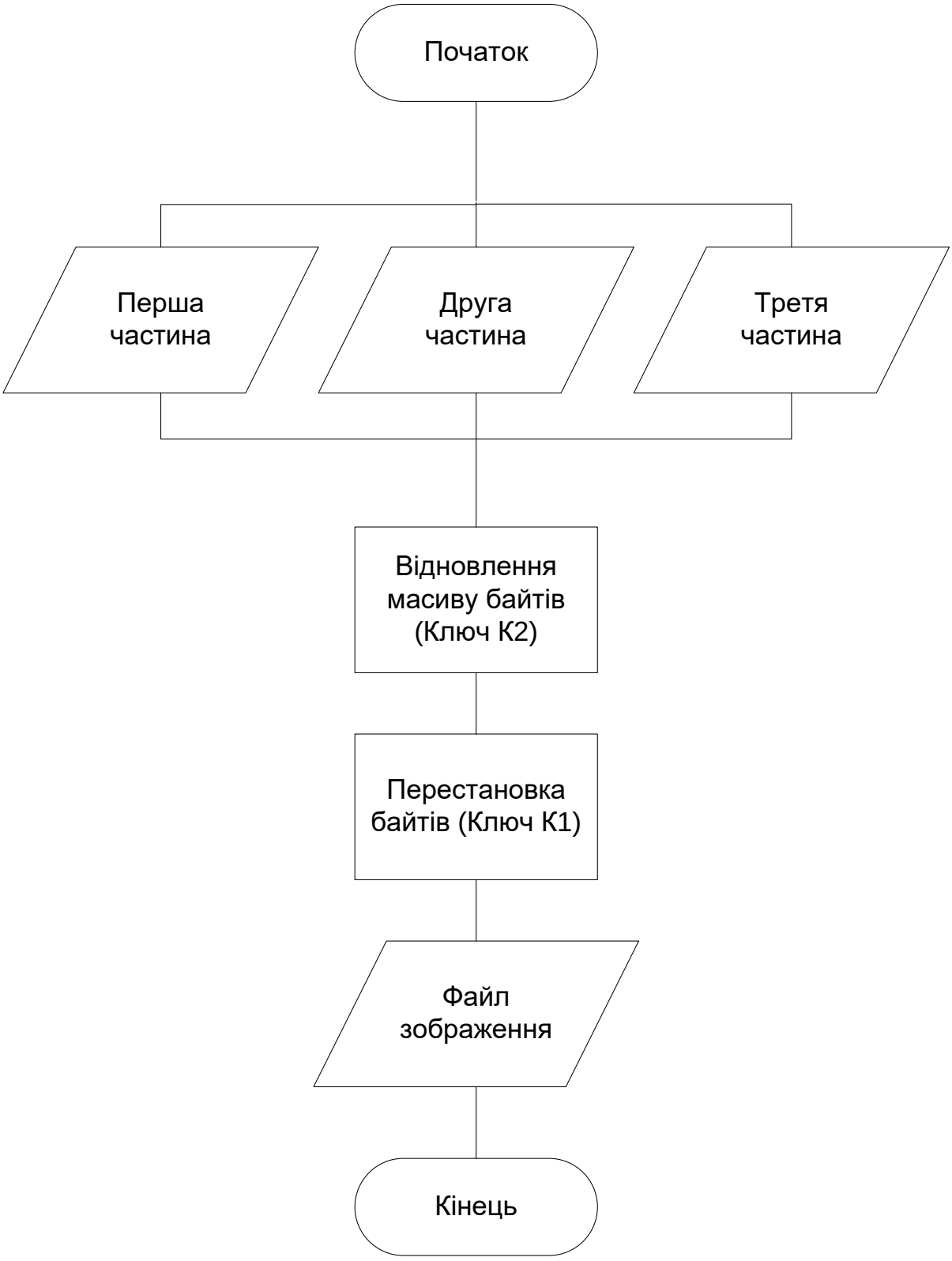
Виконав: студент 2 курсу групи 1БС-21мс
спеціальності 125 Кібербезпека
_____ Кирилюк О.П.
19 червня 2023 р.

Керівник: зав. кафедри ЗІ д.т.н., проф
_____ Лужецький В.А.
19 червня 2023 р.

# АЛГОРИТМ ЗАШИФРУВАННЯ ЗОБРАЖЕННЯ

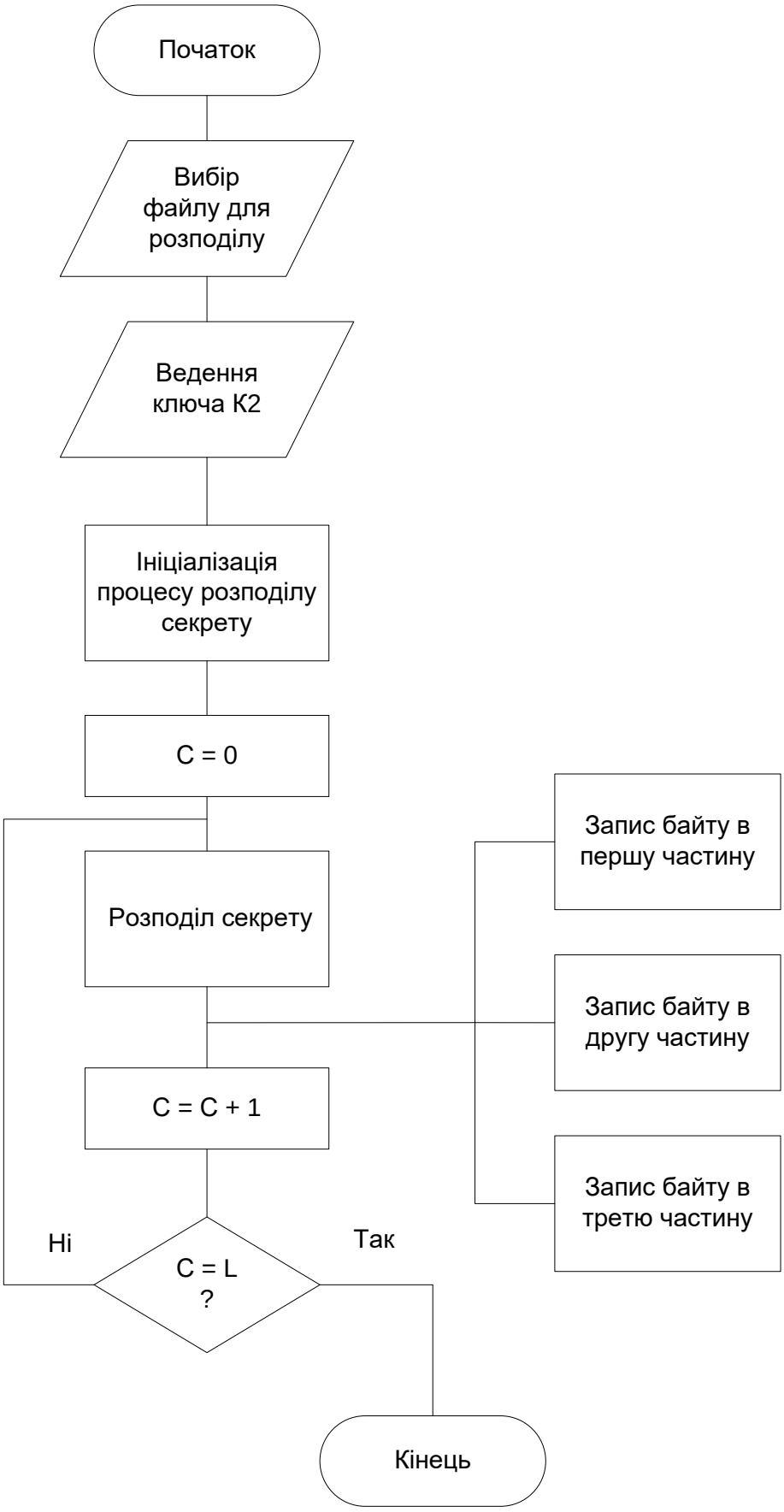


АЛГОРИТМ РОЗШИФРУВАННЯ ЗОБРАЖЕННЯ

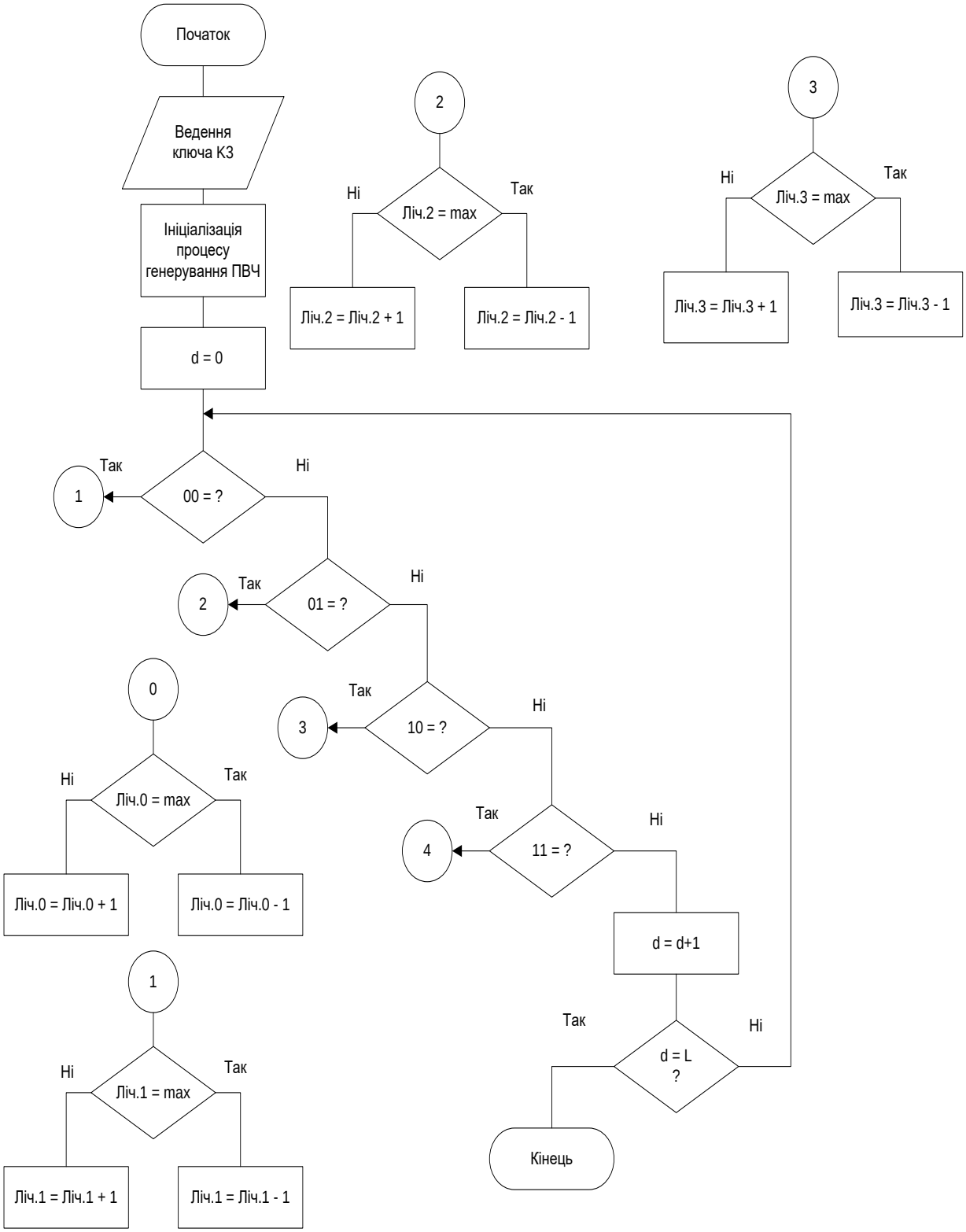


# АЛГОРИТМ ПЕРЕСТАНОВКИ БАЙТІВ

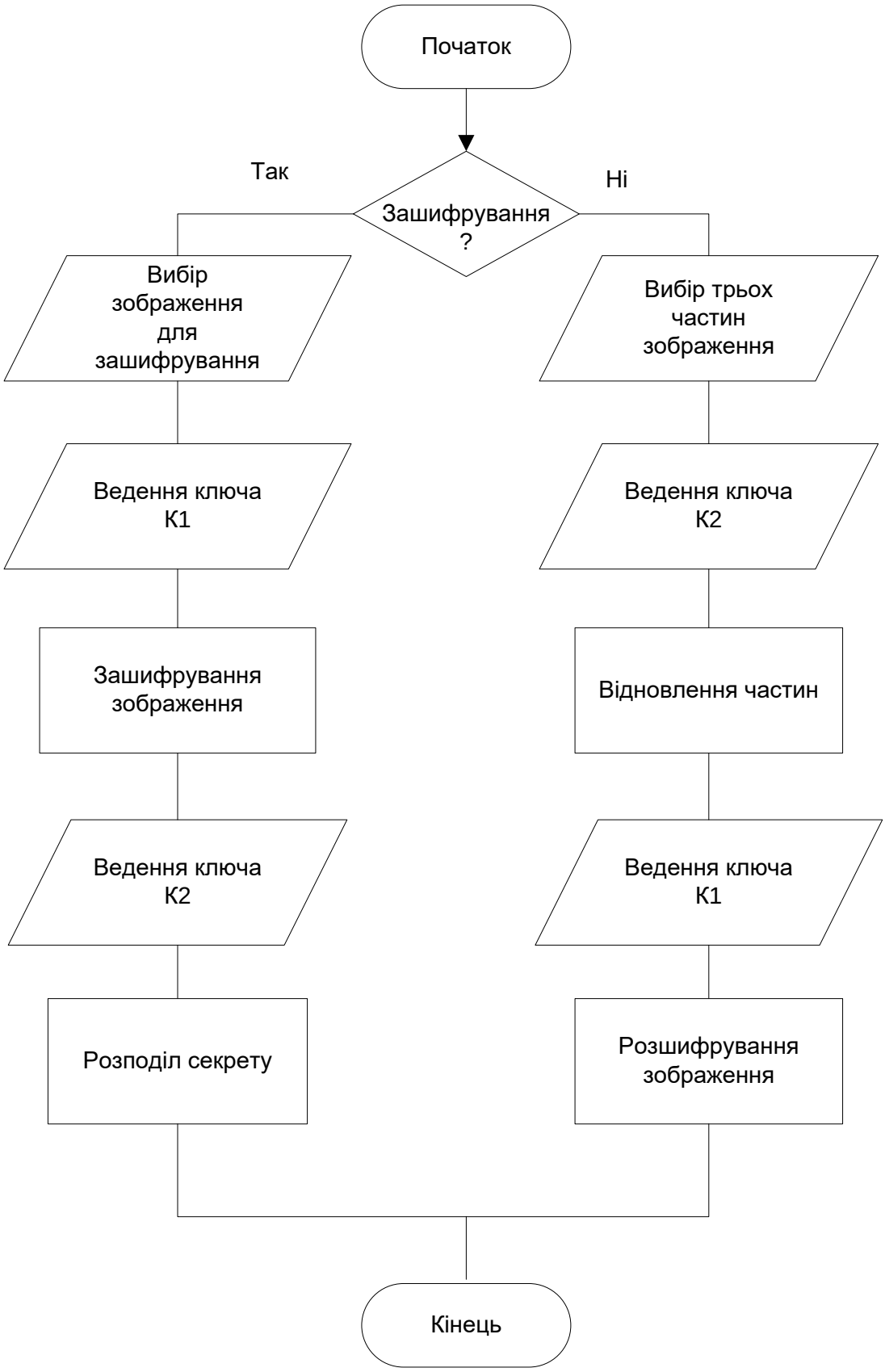
# АЛГОРИТМ РОЗПОДІЛУ СЕКРЕТУ



# АЛГОРИТМ ФОРМУВАННЯ ПСЕВДОВИПАДВОХ ЧИСЕЛ



# АЛГОРИТМ РОБОТИ ПРОГРАМИ



СТРУКТУРА BMP ЗОБРАЖЕННЯ