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

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Метод та засіб розподілення секрету»
08-20.МКР.005.00.000 ПЗ

Виконав: студент 2 курсу, групи 1 БС-18 м
спеціальність 125 – Кібербезпека

_____ Дехтяренко М. С.

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

_____ Лужецький В. А.

Рецензент к. т. н., доц., доц. каф. ОТ

_____ Крупельницький Л. В.

Вінниця - 2019 року

Вінницький національний технічний університет

Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації
Освітньо-кваліфікаційний рівень магістр
Спеціальність 125 – Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри ЗІ, д.т.н., проф.

_____ В. А. Лужецький

_____ _____ 2019 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІНУ РОБОТУ СТУДЕНТУ

Дехтяренко Миколі Сергійовичу

1. Тема роботи: «Метод та засіб розподілення секрету» керівник роботи: Лужецький Володимир Андрійович, д. т. н., проф., зав. кафедри ЗІ, затверджена наказом ректора ВНТУ від 02.10.2019 року №254.
2. Строк подання студентом роботи 17.12.2019 р.
3. Вихідні дані до роботи:
   - тип даних – кольорове зображення;
   - формат даних зображення – BMP;
   - схема розподілу секрету – порогова;
   - мінімальний розмір зображення – 1 кб.
4. Зміст розрахунково-пояснювальної: Вступ. Аналіз відомих методів. Розробка методу розділення секрету. Розробка програмного додатку. Економічна частина. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: Етапи розподілу секрету. Етапи відновлення секрету. Схема перестановок вмісту секрету. Масиви елементів секрету. Приклад для порогової схеми (3, 7). Алгоритм роботи програми. Алгоритм генерування ключа. Алгоритм розподілу секрету. Алгоритм відновлення секрету. Приклад розділення зображення за схемою (6, 3) з використанням 8 лічильників. Приклад розділення зображення за схемою (6,

3) з використанням 64 лічильників. Приклад відновлення зображення за схемою (6, 3) з використанням лише двох коректних частин.

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 каф.ЗІ		
2	Лужецький В. А., д.т.н., проф., Зав каф.ЗІ		
3	Лужецький В. А., д.т.н., проф., Зав каф.ЗІ		
4	Мацкевічус С. С., ст. викл. каф. ЕПВМ		

7. Дата видачі завдання 01 вересня 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	1.1.1 Примітка
1	Аналіз завдання. Вступ	01.09.2019 – 04.09.2019	
2	Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2019 – 15.09.2019	
3	Науково-технічне обґрунтування	16.09.2019 – 22.09.2019	
4	Розробка технічного завдання	23.09.2019 – 29.09.2019	
5	Розробка рішень	30.09.2019 – 12.10.2019	
6	Практична реалізація, моделювання, експериментування, результати	14.10.2019 – 10.11.2019	
7	Розробка розділу економічного обґрунтування доцільності розробки	11.11.2019 – 17.11.2019	
8	Аналіз виконання ТЗ, висновки	18.11.2019 – 24.11.2019	
9	Оформлення пояснювальної записки	25.11.2019 – 30.11.2019	
10	Попередній захист та доопрацювання МКР	28.11.2019 – 01.12.2019	
11	Перевірка магістерської роботи на наявність плагіату	02.12.2019 – 10.12.2019	
12	Представлення МКР до захисту	11.12.2019 – 14.12.2019	
13	Захист МКР	16.12.2019 – 20.12.2019	

Студент_____Дехтяренко М. С.
(підпис)

Керівник роботи _____Лужецький В. А.
(підпис)

АНОТАЦІЯ

Магістерська кваліфікаційна робота присвячена розробці методу та засобу розподілу секрету. В роботі проаналізовано існуючі методи розподілу секрету. Розроблено схему розподілу секрету для зображення. Проаналізовано структуру BMP зображення. Розроблено алгоритм роботи регістру зсуву з лінійним зворотним зв'язком та блок перестановки та заміни. Реалізовано програмний засіб на основі наведених алгоритмів. Після розробки було проведене тестування програмного засобу. В економічному розділі оцінено витрати на розробку.

ABSTRACT

Master's thesis is devoted to the development of the method and means of sharing the secret. The paper analyzes the existing methods of secret distribution. The scheme of distribution of a secret for the image is developed. The structure of the BMP image is analyzed. A linear feedback shift algorithm is developed and a permutation and replacement unit. Implemented a software tool based on the above algorithms. The software was tested after development. The economic section estimates development costs.

ЗМІСТ

ВСТУП

Інформація має важливе значення в життєдіяльності людства. При цьому вона стає все більш вразливою через зростаючі обсяги збережених даних.

Тому все більшу важливість набуває проблема захисту інформації від несанкціонованого доступу під час передачі та зберіганні.

Для захисту секретної інформації від втрати і компрометації необхідно підвищити надійність зберігання секретної інформації. Є кілька варіантів підвищення надійності зберігання секретної інформації, наприклад [1, 2]:

- створення декількох копій секретної інформації та зберігання їх в різних місцях. Даний метод підвищує надійність при втраті, але підвищує ймовірність компрометації ключа;
- поділ секретних даних між учасниками певної групи. Даний метод зменшує ймовірність компрометації ключа.

У магістерській кваліфікаційній роботі буде детально розглянутий другий метод, а саме розподіл секрету між учасниками певної групи, та розроблено метод розподілу секрету, що зберігається у вигляді зображення на основі порогових схем.

Об'єктом дослідження є процес захисту інформації за допомогою розподілення секрету.

Предметом дослідження є метод та засіб для розподілу секрету, що зберігається у вигляді зображень.

Метою магістерської кваліфікаційної роботи є підвищення захищеності секретного вмісту зображень за рахунок розробки методу розподілу секрету.

Для досягнення мети необхідно:

– проаналізувати відомі методи розподілу секрету;
– проаналізувати методи візуальної криптографії;
– розробити власний метод розподілу секрету;
– розробити програмний засіб для виконання даного методу.

Наукова новизна полягає в тому, що вперше запропоновано метод розподілу секрету, що міститься в зображеннях, який, на відміну від відомих, дозволяє використати порогову схему відновлення.

Практичну цінність роботи складає програмний засіб для розподілу секрету, що міститься в зображенні.

2 АНАЛІЗ ВІДОМИХ МЕТОДІВ

2.1 Узагальнена схема розподілу секрету

Схема розподілу секрету – криптографічна схема, що дозволяє розділити секрет між учасниками групи, при цьому кожен учасник отримує частку секрету, а вихідний секрет стирається. Відтворити секрет може певна коаліція учасників [3].

Узагальнена схема розподілу секрету визначається набором [4]:

$$D=(P,\Gamma,Z,S,\{F_z : z\in Z\},G)$$

де P – скінченна множина учасників,

Z – скінченна множина секретів,

S – скінченна множина частин секретів,

Γ – структура доступу. Непуста множина деяких непустих підмножин множини P, замкнута співвідношенням включення $(A\in\Gamma \& A\subseteq B\subseteq P)\Rightarrow B\in\Gamma$.

$F_z : z\in Z$ – скінченна множина правил розподілу секрету. Кожне правило це – функція $f: P\to S$ поліноміальної складності.

G – функція(правило) відновлення секрету з областю визначення $F^A=\{f^A : f\in F_z, z\in Z, A\in\Gamma\}$ та областю значень Z, з такою властивістю $G(f^A)=z$ для всіх $z\in Z$, $f\in F_z$ та $A\in\Gamma$.

При цьому для будь-якої функції $f: X\to Y$ та будь-якого $A\subseteq X$ через f^A позначають обмеження f на A. Тобто функція $f^A: A\to Y$ визначається, як $f^A(a)=f(a)$ для всіх $a\in A$.

Також для $P_i\in P$ та $f\in F_z$ значення $f(P_i)$ є частиною секрету z, яку отримує учасник P_i при розподілі z за правилом f, а f^A для $A\subseteq P$ – це розподіл частин z учасникам в A при такому розподілі. Відносно D та Γ кажуть, що схема розподілу секрету D реалізує структуру доступу Γ.

Підмножини множини P, що є елементами структури доступу Γ називаються авторизованими підмножинами, а ті що не є елементами структури

доступу Γ – неавторизованими підмножинами. Внаслідок замкнутості Γ будь-яка множина, що включає в себе деяку авторизовану підмножину є авторизованою. Необхідно зауважити, що відповідно до властивості правила відновлення секрету, для будь-якої авторизованої підмножини учасників A та для будь-яких правил розподілу секрету $f \in F_z$ та $g \in F_{z'}$ коли $z \neq z'$ випливає $f^A \neq g^A$. Це передбачає, зокрема, імплікацію $z \neq z' \Rightarrow F_z \cap F_{z'} = \varnothing$.

Нехай усюди далі $F = \bigcup_{z \in Z} F_z$. Схема розподілу секрету D називається досконалою, якщо для будь-якої неавторизованої підмножини $A \subseteq P$ та для будь-яких $f \in F$ та $z' \in Z$ існує $g \in F_{z'}$ таке, що $f^A = g^A$. Якщо $f \in F_z$, то існування для будь-якого $z' \in Z$ функції $g \in F_{z'}$, із властивістю $f^A = g^A$ означає, що по f^A жоден з елементів z' в Z не відрізняється від секрету z. Таким чином у випадку досконалої системи розподілу секрету учасники неавторизованої підмножини A, об'єднавши свої частини секрету (обчисливши f^A), не зможуть заперечити ніякий із секретів в Z як неможливий.

Для кожного учасника $P_i \in P$ визначимо $S(P_i) = \{ f(P_i) : f \in F \}$ та $\rho(P_i) = \dfrac{\log|Z|}{\log|S(P_i)|}$. Нехай $\rho = \min_{P_i \in P} p(P_i)$. Величина ρ називається швидкістю інформації схеми D . Вона слугує показником ефективності схеми розподілу секрету – того, як співвідносяться довжини в бітах (двійкове представлення) секрету та його частин в розподілі по схемі D . Схема розподілу секрету називається ідеальною, якщо $\rho = 1$. В досконалій схемі розподілу секрету $\rho \leq 1$.

2.2 Схеми розподілу секртету

2.2.1 Схема Брікелла

Нехай p – просте число, n – натуральне число, $n \geq 2$, Z_p^n – лінійний n-вимірний векторний простір над полем Z_p та $\phi : P \to Z_p^n$ – деяка функція, що має таку властивість: $10^{n-1} \in \langle \phi(A) \rangle \Leftrightarrow A \in \Gamma$, де $\phi(A) = \{ \phi(P_i) : P_i \in A \}$ та $\langle V \rangle$ для $V \subseteq Z_p^n$

означає множину всіх лінійних комбінацій елементів в V, як вектор простору Z_p^n. Тоді в схемі Брікелла $D=(P,\Gamma,Z,S,\{F_z : z\in Z\},G)$, за визначенням, $Z=S=Z_p$, для кожного секрету $z\in Z$ будь-яке правило $f\in F_z$ задається деяким набором чисел $(a_1, ..., a_{n-1})\in Z_p^{n-1}$ та визначається для будь-якого учасника $P_i\in P$ як $f(P_i)=(a,\phi(P_i))$ – скалярне множення векторів $a=(z,a_1, ..., a_{n-1})$ та $\phi(P_i)$, а правило G відновлення невідомого секрету за відомими частинами f^A для $f\in F$ та $A\in\Gamma$ визначається як $G(f^A)=\sum_{P_i\in A} c_i f(P_i)$, де коефіцієнти $c_i\in Z_p$ для $P_i\in A$ обраховуються з умови $10^{n-1}=\sum_{P_i\in A} c_i\phi(P_i)$ [4].

Будь-яка схема Брікелла є досконалою та ідеальною схемою розподілу секрету [4].

2.2.2 Структури доступу Брікелла

Структура доступу Γ на P називається структурою доступу в лінійно векторному просторі або структурою Брікелла, якщо існують такі прості p, натуральне $n\geq 2$ та функція $\phi: P\to Z_p^n$, що $10^{n-1}\in\langle\phi(A)\rangle\Leftrightarrow A\in\Gamma$. З встановленого вище випливає, що будь яка така структура доступу реалізується досконалою та ідеальною схемою розподілу секрету – схемою Брікелла. Нажаль, докладний опис всіх структур доступу в лінійно векторному просторі відсутній, невідомі необхідні та достатні умови для довільної структури доступу бути структурою Брікелла. З цієї причини опишемо два класи структур Брікелла – структури доступу на основі повного багато реберного графа та порогові структури доступу [4].

Нехай далі для довільної структури доступу Γ на P множина $\Gamma_0=\{A_1,...,A_k\}$ є нижня границя множини Γ, упорядкованого відношення включення, тобто $\Gamma_0=\{A\in\Gamma:\neg\exists B\in\Gamma(B\neq A\& B\subseteq A)\}$. Множина Γ_0 однозначно визначає Γ, а саме $\Gamma=\{A\subseteq P:\exists B\in\Gamma_0(B\subseteq A)\}$.

Розіб’ємо множину учасників P довільним чином на класи $P^{(1)},...,P^{(l)}$ та опишемо нижню границю наступним чином

$$\Gamma_0 = \left\{\{u, v\} \subseteq P : \exists i, j \in \{1,..., l\} (i \neq j \,\&\, u \in P^{(i)} \,\&\, v \in P^{(j)})\right\},$$

тобто Γ_0 складається з усіх пар учасників з різних класів розбиття. Описану структуру доступу Γ на P назвемо структурою повного l-дольного графа, тому що таким є граф (P, Γ_0) з множиною вершин P та множиною ребер Γ_0. Структура доступу Γ повного l-дольного графа є структурою Бріккела.

Будь-яка (m, n)-порогова структура доступу є структурою Брікелла.

Схему розподілу секрету Брікелла, що реалізовує (m, n)-порогову структуру доступу, прийнято називати на ім'я її автора (m, n)-порогова схема Шаміра.

2.2.3 Схема Шаміра

Нехай $p \geq |P|$ – просте число та кожному учаснику $P_i \in P$ поставлено у відповідність деяке число $x_i \in Z_p^*$, так що різним учасникам відповідають різні числа. Тоді в (m, t)-пороговій схемі Шаміра D, за визначенням, $Z = S = Z_p$ для кожного $z \in Z$ будь-яке правило $f \in F_Z$ задається набором чисел $(h_1,..., h_{t-1}) \in Z_p^{t-1}$ та визначається для будь-якого учасника $P_i \in P$ як $f(P_i) = h(x_i)$, де $h(x) = \sum_{j=0}^{t-1} h_j x^j \in Z_p[x]$ для $h_0 = z$, а правило G відновлення невідомого секрету по відомому розподіленню частин f^A для $f \in F$ та $A \in \Gamma$ визначається як $G(f^A) = \sum_{j=1}^{t} c_j y_j$, де $c_j = \prod_{1 \leq k \leq t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}$ для $j = 1,..., t$ та довільного $\{P_{i_1},..., P_{i_t}\} \subseteq A$ [4].

Схема Шаміра є досконалою та ідеальною схемою розподілення секрету.

2.2.4 Загальна схема розподілу секрету на основі порогової

Побудуємо схему розподілу секрету $D_t = (P, \Gamma, Z, S, \{F_z : z \in Z\}, G)$ з заданою множиною учасників $P = \{P_1,.., P_m\}$ для $m \geq 2$ та довільною структурою доступу Γ на P, в якій $Z = Z_n$ для $n \geq 2$ [4].

Нехай $\Gamma_0 = \{A_1,..., A_k\}$ та $A_j = \{P_{j_1}, P_{j_2},..., P_{j_{r(j)}}\}$ для $j = 1,..., k$. Розглянемо довільний секрет $z \in Z$ та для кожного $j = 1,..., k$ оберемо з Z випадкові числа

$z_{j_1}, z_{j_2}, ..., z_{j_{r(j)}}$, таким чином щоб $\left(z_{j_1} + z_{j_2} + ... + z_{j_{r(j)}}\right) \bmod n = z$, тобто частини секрету z від його розподілу по простій (t,t) -пороговій схемі при $t = r(j)$. Для всіх $i = 1,...,m$ та $j = 1,...,k$ ведемо

$$v_{ij}(z) = \begin{cases} -, \text{якщо } P_i \notin A_j; \\ z_{j_u}, \text{якщо } \exists u \in \{1,...,r(j)\} (i = j_u) \end{cases}$$

та ведемо $v_i(z) = \left(v_{i1}(z), v_{i2}(z), ..., v_{ik}(z)\right)$ та $v_i(z) = \left(v_1(z), v_2(z), ..., v_m(z)\right)$.

Тут символ «-» означає «не визначено». За побудовою в наборі $v_i(z)$ компонента $v_{ij}(z)$ визначена (відмінна від «-»), тоді і тільки тоді $P_i \in A_j$; в цьому випадку $v_{ij}(z)$ є деяким числом в Z_n. Нехай k_i - кількість всіх A_j для $j = 1,....,k$, що містять P_i та S_i - множини всіх наборів в $(Z_n \cup \{-\})^k$ з $k - k_i$ невизначеними компонентами. Тоді $v_i(z) \in S_i$. Визначимо $S = S_1 \cup S_2 \cup ... \cup S_m$. При фіксованому z набір $v(z)$ визначається однозначно вибраними для кожного $j = 1,....,k$ числами $z_{j_1}, z_{j_2}, ..., z_{j_{r(j)}}$. Позначимо $V(z)$ множиною всіх таких наборів $v(z)$, можливих при різному виборі цих чисел та визначимо множину F_z, віднісши до нього будь-яке таке і тільки таке правило $f: P \to S$, що для деякого набору $v(z) = \left(v_1(z), v_2(z), ..., v_m(z)\right) \in V(z)$ виконується рівність $f\left(P_i\right) = v_i(z)$ для кожного $i = 1,...,m$.

Для визначення правила відновлення G розглянемо $A = \left\{P_{i_1}, ..., P_{i_r}\right\} \in \Gamma$ та $f \in F = \bigcup_{z \in Z} F_z$. За визначенням Γ_0 знайдеться таке $A_j = \left\{P_{j_1}, P_{j_2}, ..., P_{j_{r(j)}}\right\} \in \Gamma_0$, що задовольняє умову $A_j \subseteq A$. Нехай $f\left(P_{j_u}\right) = \left(v_{j_u 1}, ..., v_{j_u k}\right)$ для $u = 1,...,r(j)$. Припустимо $G\left(f^A\right) = \left(v_{j_1 j} + ... + v_{j_{r(j)} j}\right) \bmod n$ та переконаємося, що визначена таким чином функція G дійсно відновлює розділений секрет. Справді, нехай $f \in F_z$. Тоді за визначенням f для будь-якого $u = 1,...,r(j)$ справедлива рівність $f\left(P_{j_u}\right) = \left(v_{j_u 1}(z), ..., v_{j_u k}(z)\right)$, де для $P_{j_u} \in A_j$ справедливо $v_{j_u j}(z) = z_{j_u}$ та $\left(z_{j_1} + z_{j_2} + ... + z_{j_{r(j)}}\right) \bmod n = z$. Отже

$$G\left(f^A\right) = \left(v_{j_1 j}(z) + ... + v_{j_{r(j)} j}(z)\right) \bmod n = \left(z_{j_1} + z_{j_2} + ... + z_{j_{r(j)}}\right) \bmod n = z.$$

Побудова схеми D_t закінчено.

2.3 Візуальна криптографія

Візуальна криптографія – один з криптографічних методів, який дозволяє зашифрувати візуальну інформацію (картинку, текст і т. д.) таким чином, що розшифрування стає механічною операцією, яка не потребує обов’язкового використання комп'ютера [5, 6].

Основна ідея візуальної криптографії полягає в розбитті вихідного зображення на кілька шифрованих («тіньових» зображень, shadow images), кожне з яких не дає ніякої інформації про оригінальний документ крім, його розміру. Наприклад для зображень це може бути штатна та довжина. При накладенні шифрованих зображень один на одного, можна отримати вихідне зображення. Таким чином, для декодування не потрібно спеціальних знань, високопродуктивних обчислень і навіть комп'ютера (в разі, якщо роздрукувати «тіньові» зображення на прозорих плівках).

У разі використання цього алгоритму в комп'ютерних системах, накласти всі частини зображення один на одного можна використовуючи логічні операції AND, OR, XOR, NOT XOR (або встановивши більш високу ступінь прозорості в графічному редакторі). Дана технологія має високу криптостійкість за рахунок того, що при розподілі вихідного зображення на множину шифрованих зображень відбувається випадковим чином [5].

Алгоритми візуального шифрування мають такі властивості [5]:

– регулярність (виконуються однакові дії для кожного вихідного пікселя);
– незалежність (кожен вихідний піксель шифрується незалежно від інших);
– простота (можливо візуальне розшифрування за допомогою фізичного процесу накладення шумоподібних зображень без обчислень) .

Візуальна криптографія, як і звичайна криптографія, застосовується з метою приховування інформації. Основною особливістю візуальної криптографії спочатку була можливість розшифрування даних без використання комп'ютера, але зараз цей напрямок практично не використовується.

У наш час можна виділити такі основні області застосування візуальної криптографії:

– Шифрування даних. Для шифрування необхідна прямокутна підкладка-зображення, що складається випадковим чином сформованих чорних і білих пікселів. Підкладка генерується один раз, і передається співрозмовнику. Потім, якщо один зі співрозмовників має намір передати повідомлення, то генерує другу таку підкладку таким чином, щоб при накладенні на вихідну виходив осмислений текст або зображення. Але, так як шифрування переданих даних за допомогою методів візуальної криптографії вимагає великих обчислювальних ресурсів, розумно використовувати ці методи тільки лише для шифрування обмеженого обсягу інформації.
– Шифрування ключової інформації (управління ключами доступу, наприклад, для обміну ключами симетричною криптографії, і спільне використання паролів). Цей варіант є більш переважним, оскільки малий обсяг інформації, що приховується нівелює основний недолік візуальної криптографії – ресурсо місткість і низька швидкість роботи.
– Захист від несанкціонованого поширення інформації та перевірка справжності (авторських прав). Для цієї мети використовується алгоритм маркування зображень для захисту від копіювання та несанкціонованого поширення інформації. Суть цього алгоритму полягає в наступному: необхідно цифровий водяний знак автора впровадити на основі наявної у автора підкладки в зображення; щоб автор міг переконатися в достовірності, йому потрібно буде

розшифрувати зображення, поєднавши підкладку з зображенням, і отримати водяний знак, який і підтвердить авторство.

– Відстеження електронних бланків при віддаленому голосуванні. Фактично, це окремий випадок попереднього варіанту. Візуальна криптографія може бути використана при проведенні будь-яких виборів, опитувань, референдумів з використанням електронних бюлетенів як спосіб контролю кожним виборцем правильності обліку його голосу при здійсненні підрахунку голосів без порушення принципу анонімності.

2.3.1 Метод Наора та Шаміра для чорно-білих зображень

Автори розробили даний метод в 1994 році. Вони продемонстрували схему розподілу секретного зображення, згідно якої зображення було розділене на n частин так, що тільки людина, яка має всі n частин, могла розшифрувати зображення, в той час як інші $n-1$ частини не давали жодної інформації про оригінальне зображення. Кожна частина була надрукована на окремих плівках і розшифрування виконувалося шляхом накладання цих частин. Тобто при накладанні всіх n частин з'являється вихідне зображення. Таким чином, для розшифрування не потрібно високопродуктивних обчислень, спеціальних знань і навіть комп'ютера [7].

Для того, щоб розбити чорно-біле зображення на n частин, необхідно кожний піксель представити у вигляді блоків розміром два на два, піксель розділений на чотири частини, може мати шість різних станів. Якщо піксель на першому шарі має одне положення, піксель на іншому шарі в свою чергу може мати два положення: ідентичне або інвертоване порівняно з пікселем першого шару. Якщо піксель шару 2 ідентичний пікселя шару 1, то піксель, отриманий в результаті накладання буде наполовину білий і наполовину чорний. Такий піксель називають сірим або порожнім. Якщо пікселі шару 1 і шару 2 протилежні, то піксель, отриманий в результаті перекриття буде повністю чорним. Він буде інформаційним [7].

Всі пікселі першого прозорого зображення (першого шару) мають випадкові стани (з шести можливих, наведених на рис. Рисунок 1.1). Шар 2 ідентичний шару 1, за винятком тих пікселів, які повинні бути чорними (інформаційні). Ці пікселі мають стан, протилежний тому ж пікселю першого шару. Якщо перекрити один шар другим, області з однаковими станами будуть виглядати сірими, а області з зворотними станами будуть виглядати чорними [8].

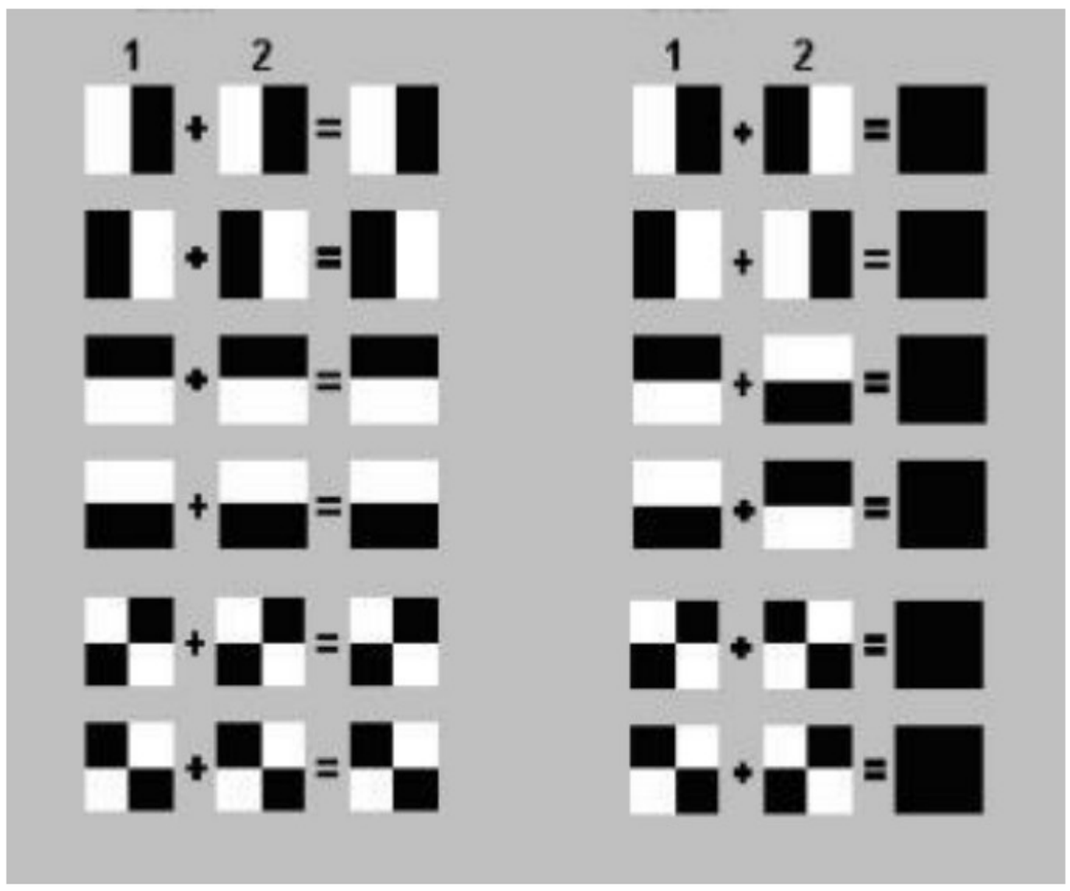


Рисунок 1.1 – Модель процесу накладання шарів

Тому, коли два зображення накладаються, з'являється оригінальне зображення. Однак розглянуті окремо зображення не дають ніякої інформації про оригінальне зображення; воно не відрізняється від випадкового набору пар виду чорно-білого / біло-чорного. Крім того, якщо є одне із зображень, можна використовувати правила, наведені вище для створення підробки другої частини зображення, яка в поєднанні з першою може дати взагалі будь-яке зображення.

Розглянемо приклад розбиття чорно-білого зображення з текстовою інформацією на дві частини за методом Наора та Шаміра. Початкове зображення показано на рисунку Рисунок 1.2.

Рисунок 1.2 – Тестове чорно-біле зображення

Після розбиття отримано два «тіньових» зображення (рис. Рисунок 1.3 та рис. Рисунок 1.4).

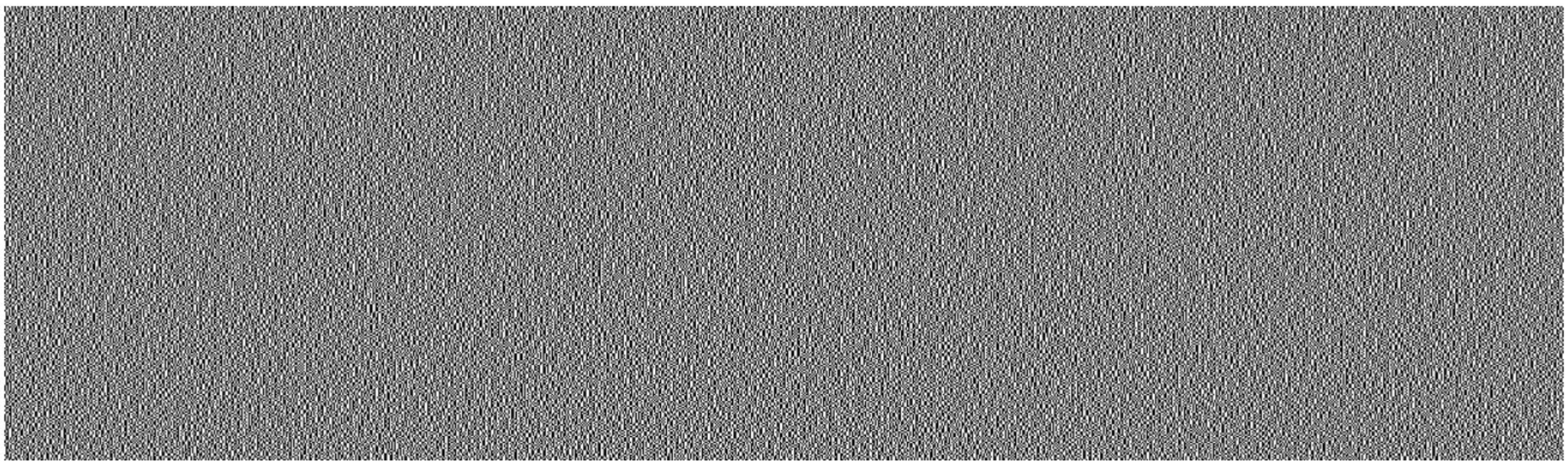


Рисунок 1.3 – Тіньове зображення №1

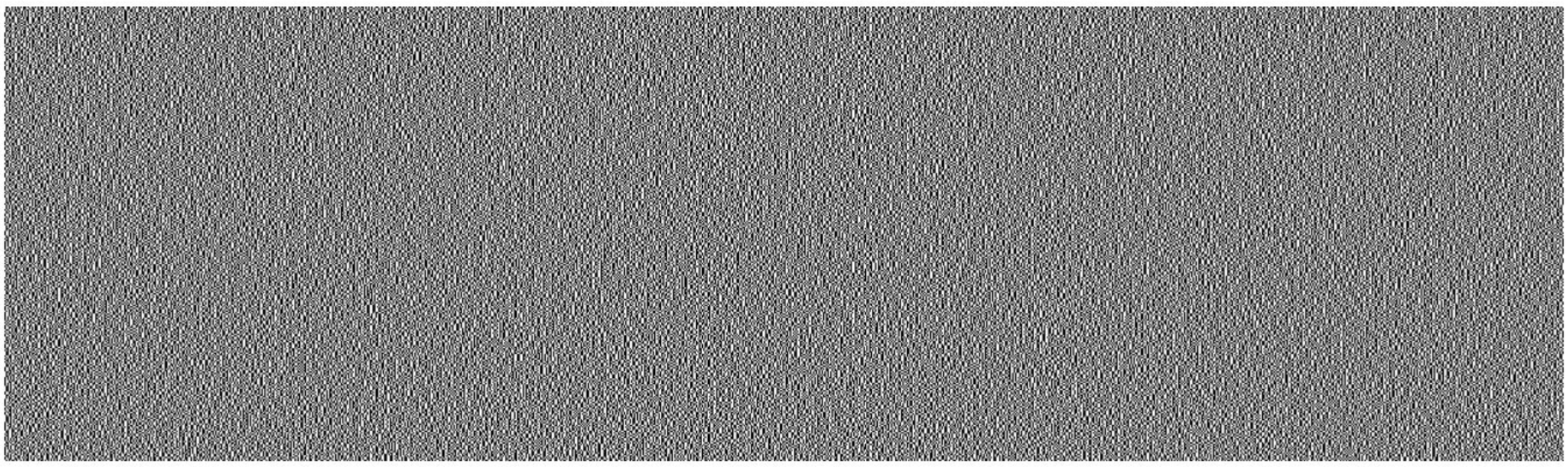


Рисунок 1.4 – Тіньове зображення №2

Початкове зображення, що вийшло в наслідок операції розшифрування за допомогою накладання кольорів з використанням AND (рис Рисунок 1.5).

Рисунок 1.5 – Відновлене зображення з використанням операції AND

Початкове зображення, що вийшло в наслідок операції розшифрування за допомогою накладання кольорів з використанням OR (рис. Рисунок 1.6).

Рисунок 1.6 – Відновлене зображення з використанням операції OR

Початкове зображення, що вийшло в наслідок операції розшифрування за допомогою накладання кольорів з використанням XOR (рис. Рисунок 1.7).

Рисунок 1.7 – Відновлене зображення з використанням операції XOR

Початкове зображення, що вийшло в наслідок операції розшифрування за допомогою накладання кольорів з використанням NOT XOR (рис. Рисунок 1.8).

Рисунок 1.8 – Відновлене зображення з використанням операції NOT XOR

З отриманих відновлених зображень очевидно, що використання логічних операцій XOR та NOT XOR найкраще підходять для відновлення зображень за розглянутим методом.

Недолік даного методу полягає в тому, що розмір зашифрованих частин в чотири рази більший оригіналу.

2.3.2 Схема передачі зашифрованих зображень

Схема передачі зашифрованих зображень $(n, n+1)$ (multi secret image sharing Scheme) – схема візуальної криптографії, яка дозволяє відправити n зображень у вигляді $n+1$ зображень, що говізуально представляють шум. Для відновлення n зашифрованих зображень, необхідною умовою є наявність всіх $n+1$ «зашумлених» зображень. Дана схема передачі має високу ступінь захисту за рахунок використання схеми розподілу секрету та стійкості «зашумлених» зображень до змін. Будь-яке «зашумлене» зображення не дає часткової інформації про початкові зображення [9, 10].

Унікальність методу полягає в тому, що замість звичної для подібних алгоритмів з використанням операції XOR - використовується модульна арифметика.

Позначимо два числа протилежними, якщо $a + b = 0 (\mathrm{mod}\, n)$. Для чорно-білих або кольорових зображень, RGB значення пікселя приймемо за скалярну величину в діапазоні [0, 255]. Кожне число з даного діапазону має протилежне, зі значенням модуля 256.

Алгоритм шифрування:

Вхідні дані: n - секретних зображення $SI_1, SI_2, ..., SI_n$ розміром $h \times w$.

Вихідні дані: $n+1$ «зашумлених» зображення $NI_1, NI_2, ..., NI_n, NI_{n+1}$.

1) Генерується n тимчасових змінних $C_1, C_2, ..., C_n$. $C_i = \left\lfloor \frac{SI_i}{n+1} \right\rfloor$, $i = 1, ..., n$.

2) Генерується випадкова матриця R розміром $h \times w$.

3) На віддаленому сервері генерується ключ

$SK = (C_1) \bmod 256;$

$SK = (C_i + SK) \bmod 256;\ i = 2, ..., n.$

4) Генеруються $n+1$ «зашумлених» зображення $NI_1, NI_2, ..., NI_n, NI_{n+1}$.

$NI_i = (C_i + SK + R) \bmod 256$, $i = 1, ..., n$

$NI_{i+1} = ((n+1) * (SK + R)) \bmod 256$. NI_{i+1} - необхідне для знаходження R на приймаючій стороні.

Процес відновлення відрізняється від алгоритму шифрування і забезпечує додаткову безпеку. Навіть при отриманні зловмисником доступу до алгоритму шифрування, він не зможе отримати секретну інформацію з «зашумлених» зображень.

Алгоритм відновлення зашифрованих зображень:

Вхідні дані: $n+1$ «зашумлених» зображення $N_1, N_2, ..., N_n, N_{n+1}$

Вихідні дані: n - відновлених зображення $RI_1, RI_2, ..., RI_n$.

1) На стороні клієнта генерується ключ

$CK = (N_1) \bmod 256;$

$CK = (N_i + CK) \bmod 256;\ i = 2, ..., n.$

2) Генерується тимчасові зображення

$P_i = ((n+1) * N_i) \bmod 256$, $i = 1, ..., n$.

3) Генерується матриця R яка була згенерована на сервері при шифруванні для забезпечення випадковості в шаблоні зображень з

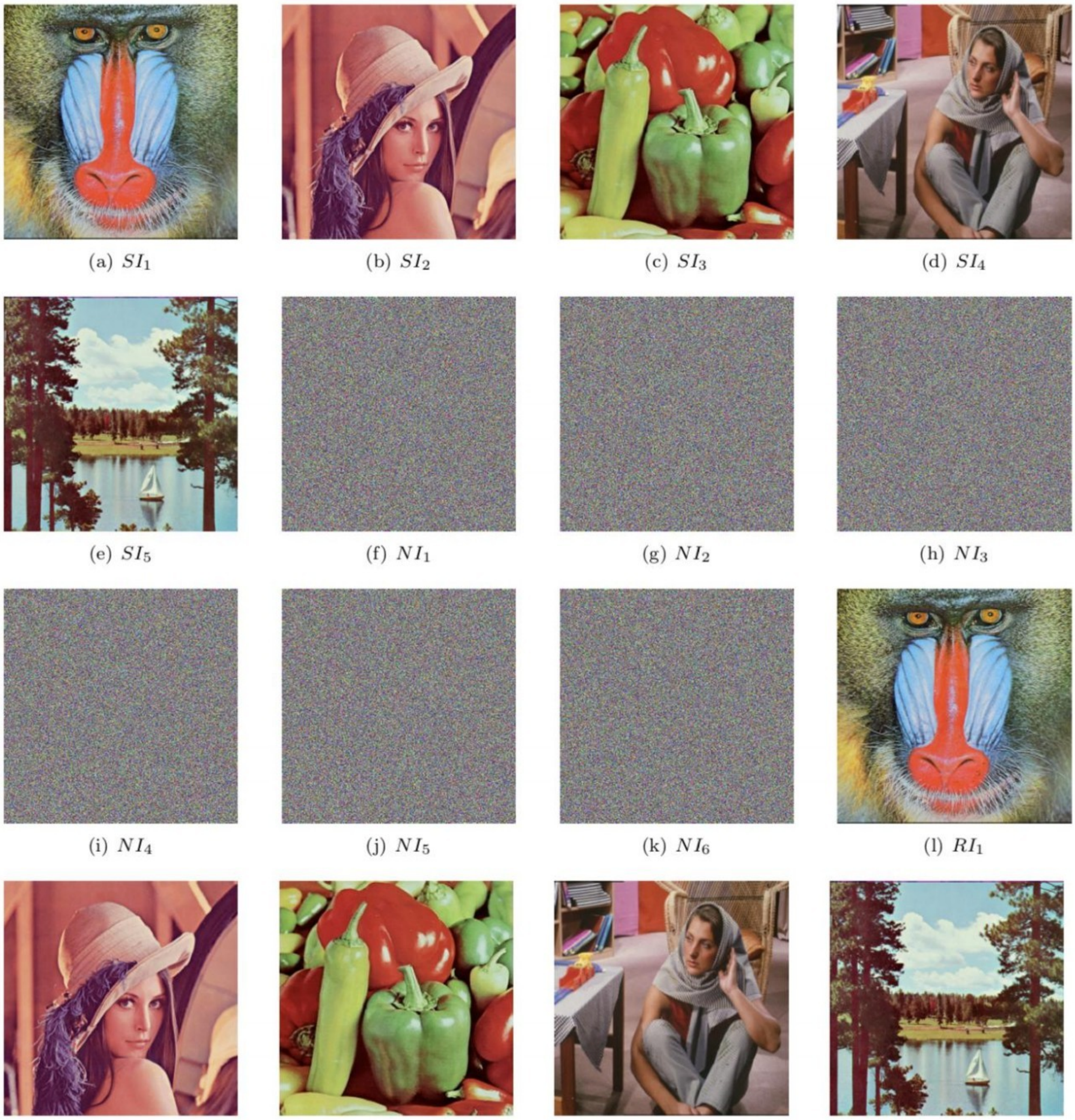
шумом.

$R = (N_{n+1} - CK) \bmod 256$.

4) Відновлення початкових зображень

$RI_i = (P_i - (CK + R)) \bmod 256$, $i = 1,..., n$.

На рисунку Рисунок 1.9 показано приклад роботи схеми.



(a) SI_1 (b) SI_2 (c) SI_3 (d) SI_4

(e) SI_5 (f) NI_1 (g) NI_2 (h) NI_3

(i) NI_4 (j) NI_5 (k) NI_6 (l) RI_1

(m) RI_2 (n) RI_3 (o) RI_4 (p) RI_5

Рисунок 1.9 – Приклад роботи схеми передачі зашифрованих картинок.

Оцінимо відповідність відновлених зображень. Для оцінки відповідності відновлених і вихідних зображень зручно користуватися: коефіцієнтом

кореляції, середньоквадратичної помилкою і піковим відношенням сигналу до шуму.

Коефіцієнт кореляції r приймає значення від -1 до +1. Значення +1 означає, що два порівнюваних зображення рівні, значення -1 – зображення протилежні, а значення $r = 0$ - означає, що зображення не корелюють. В даному випадку коефіцієнт кореляції задається формулою:

$$r = \frac{n^* \sum pq - \sum p \sum q}{\sqrt{\left(n^* \sum p^2 - \left(\sum p\right)^2\right) * \left(n^* \sum q^2 - \left(\sum p\right)^2\right)}},$$

де n - кількість парних значень пікселя у початкового p та пікселя відновленого q зображення.

Середньоквадратична помилка σ дає тим більше значення, чим менша відповідність порівнюваних зображень. Задається такою формулою:

$$\sigma = \sqrt{\frac{1}{M^* N} \sum_{x=1}^{M} \sum_{y=1}^{N} (SI_{xy} - RI_{xy})^2}$$

де M та N - розміри зображення;

SI_{xy} - значення пікселя у секретному зображені;

RI_{xy} - значення пікселя у відновленому зображені.

Пікове відношення сигналу до шуму має тим вище значення, чим краще співпадає початкове та відтворене зображення. Задається такою формулою:

$$PSNR(db) = 20 * \log_{10} 255/\sigma$$

Розглянута схема дозволяє розшифровувати отримані зображення з помилкою $\sigma \approx 3$, що не поступається схемам на основі логічних функцій. Приклад оцінювання відповідності тестових зображень показано в таблиці 1.1.

Таблиця 1.1 – Оцінка якості роботи схеми

Зображення	Кореляція	σ	PSNR, dB
SI_1, RI_1	0,9992	3,0288	38,54
SI_2, RI_2	0,9994	3,1793	38,12
SI_3, RI_3	0,9995	3,0270	38,54
SI_4, RI_4	0,9993	3,0306	38,53
SI_5, RI_5	0,9997	3,0293	38,54
SI_6, RI_6	0,9997	3,0293	38,54

Ключова проблема багатьох схем візуальної криптографії передачі зашифрованих зображень в тому, що вони розкривають часткову інформацію з менш ніж $n+1$ «зашумлених» зображень 1через недосконаліть методів шифрування на основі операції XOR, що ставить під загрозу безпеку. Вищеописаний алгоритм з використанням модульної арифметики позбавлений подібної вразливості через використання випадково-згенерованої матриці R, що вносить випадковий характер в зашифровані зображення, хоча залишає можливість злому при наявності у зловмисника повного набору зашифрованих зображень.

За умови наявності у зловмисника алгоритму шифрування і всіх переданих зображень - останнім етапом захисту є порядок переданих зображень. Коректний порядок необхідний для відновлення клієнтського ключа СК та матриці R. Кількість варіантів підбору оцінюється наступним числом комбінацій:

$$\binom{n+1}{2} = C_2^{n+1} = \frac{(n+1)!}{2!(n-1)!}.$$

що вносить додаткові обчислювальні витрати в атаку грубою силою при відправці великої кількості зображень.

3 РОЗРОБКА МЕТОДУ РОЗДІЛЕННЯ СЕКРЕТУ

3.1 Узагальнений опис методу

Метод розподілу секрету, що пропонується передбачає, що секрет можна представити у вигляді одновимірного масиву байтів:

$$M = \{m_1, m_2, m_3, m_4, m_5, ..., m_s\},$$

де s – кількість елементів масиву.

Розподіл секрету відбуватиметься між N учасниками, таким чином, щоб довільні K учасників могли відтворити оригінальний масив об'єднавши свої унікальні частини. Таким чином буде реалізована порогова схема розподілу секрету *(n, k)*, де n – кількість часток (рівна кількості учасників), на які буде поділений секрет, а k – кількість часток, які потрібні для відновлення секрету.

Для того, щоб після поділу секрет могли відновити тільки K учасників, секрет буде поділено на рівні частини за алгоритмом описаним нижче таким чином, щоб у кожного учасника був певний набір частин від загального секрету.

Для того, щоб M можна було поділити на n рівних частин в кінці масиву буде дописано довільні дані та розмір реальних даних в по-байтовому представлені.

$$ME = \{m_1, m_2, m_3, m_4, m_5, ..., m_s, d_1, d_2, ..., d_L, S_0, S_1, S_2, S_3\},$$

де $d_1, d_2, ..., d_L$ – набір випадкових байтів,

L – кількість випадкових байтів,

S_0, S_1, S_2, S_3 – кількість елементів масиву s в по байтовому представлені.

Передбачається, що s без знакове ціле число довжиною чотири байти.

Кількість випадкових байтів обчислюється за формулою:

$$L = n - \left\lfloor \frac{(s+4)}{n} \right\rfloor.$$

Метод розподілу секрету складається з таких етапів:

1) зчитування секрету;
2) формування ME на основі секрету;
3) перемішування елементів масиву ME;
4) поділ масиву ME на n однакових частин $Q=(Q_1, Q_2, Q_3..., Q_n)$,

   $Q_i = \lfloor q_1 \quad q_2 \quad \cdots \quad q_p \rfloor$, p - довжина вектора Q_i;
5) формування зашифрованих («тіньових») частин ;
6) запис зашифрованих частин.

Етапи розподілу представлено на рис. 2.1.

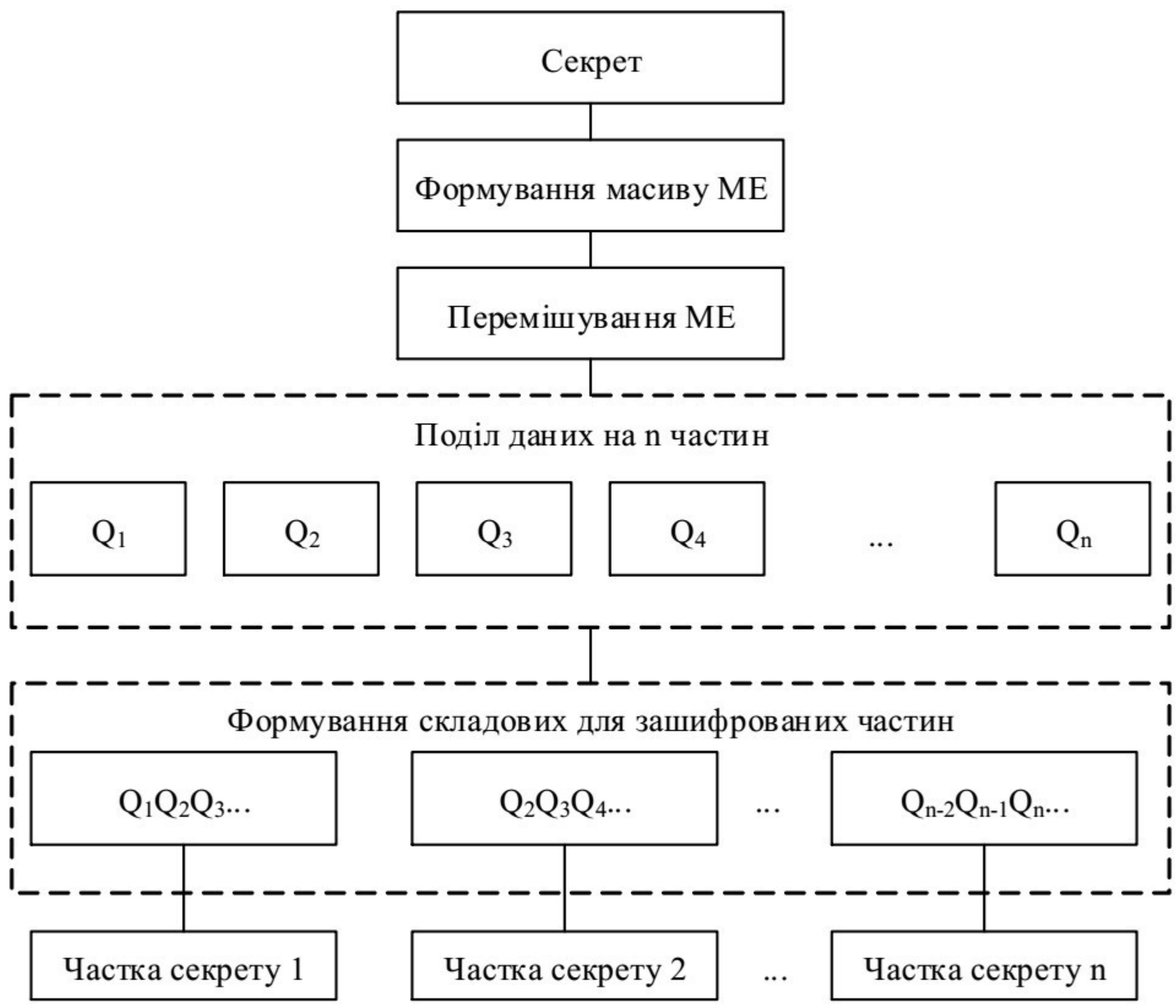


Рисунок 2.1 – Етапи розподілу секрету

Метод відновлення секрету зворотний до методу розподілу та складається з таких етапів:

1) зчитування частин k учасників;
2) відновлення $Q=(Q_1, Q_2, Q_3..., Q_n)$;
3) відновлення ME;

4) відновлення порядку елементів масиву *ME*;

5) формування секрету з масиву *ME*;

6) запис секрету.

Етапи відновлення представлено на рис. 2.2.

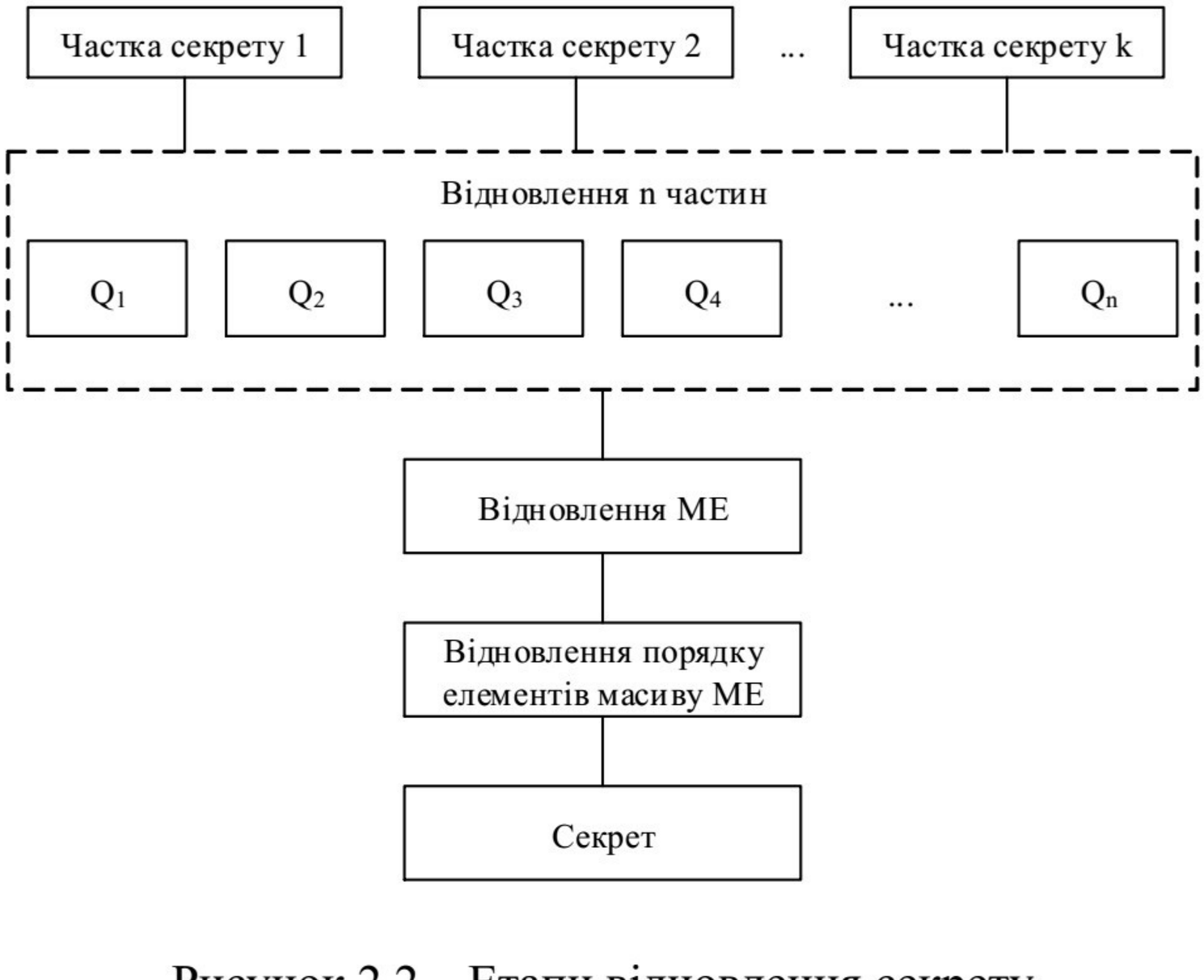


Рисунок 2.2 – Етапи відновлення секрету

Розглянемо кожний з етапів детальніше.

3.2 Перемішування вмісту секрету

Першим кроком при розподілі секрету є перемішування вмісту *ME* за допомогою перестановок.

Для реалізації перестановки за певним правилом генеруються адреси. Після чого вибирається байт з масиву *ME* за вказаною адресою та послідовно записуються в проміжний масив.

Перед записом значення у проміжний масив до нього за модулем 256 додається згорнуте значення адреси. Згортання адреси відбувається додаванням за модулем 256 усіх значень адреси в по-байтному представлені (рис. 2.3).

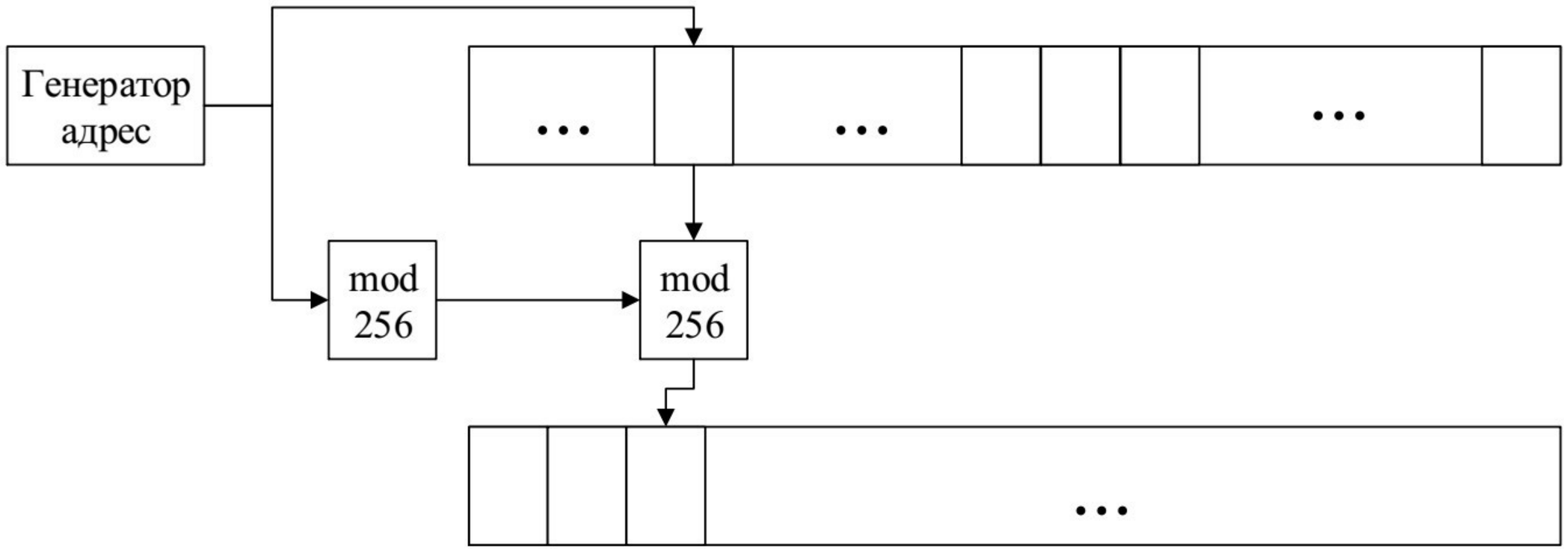


Рисунок 2.3 – Схема перестановок вмісту секрету

Для формування адрес використовується генератор псевдовипадкових чисел (ГПВЧ) без повторень в заданому діапазоні з використанням лічильників, кількість лічильників може бути зміною і задається користувачем.

Ідея побудови цього генератора полягає у такому. Задаються два масиви U_1 та U_2 однакової довжини. Довжина масивів вказує на кількість лічильників [11].

Масив U_1 – містить порядок функціонування для кожного лічильника, 0 – пряма лічба або 1 – зворотня.

Масив U_2 – містить початкові стани для кожного із лічильників. Початкові стани формуються з довжини масиву елементи якого необхідно переставити.

Наприклад при необхідності переставити елементи у масиві довжиною 32 з використанням чотирьох лічильників, значення кожного з лічильників будуть в такому діапазоні (табл.. 2.1):

Таблиця 2.1 – Мінімальні та максимальні значення лічильника

	Мінімальне значення	Максимальне значення
Лічильник 0	0	7
Лічильник 1	8	15
Лічильник 2	16	23
Лічильник 3	24	31

Вибір чисел з лічильників відбувається псевдовипадковим чином на основі послідовності, що формуються за допомогою генератора псевдовипадкових послідовностей (ГПВП).

Наприклад при використані 4 лічильників комбінація псевдовипадкових бітів 00 буде запускати Лічильник 0, 01 – Лічильник 1, 10 – Лічильник 2, 11 – Лічильник 3. Псевдовипадковим числом є число, що відповідає стану задіяного лічильника.

У табл. 2.2 наведено приклад формування ПВЧ з діапазону від 0 до 31. Тут $U_1 = \{0,1,1,0\}$, $U_2 = \{0,15,23,24\}$.

Таблиця 2.2 – Приклад формування псевдовипадкових чисел

Вихід РЗЛЗЗ	Ліч 0, 00	Ліч 1, 01	Ліч 2, 10	Ліч 3, 11	Псевдовипадкові числа
1	2	3	4	5	6
-	0	15	23	24	-
10	0	15	23 – 1= 22	24	23
01	0	15 – 1 =14	22	24	15
00	0 + 1 = 1	14	22	24	0
00	1 + 1 = 2	14	22	24	1
10	2	14	22 – 1 = 20	24	22
11	2	14	20	24 + 1 = 25	24
10	2	14	20 – 1 = 19	25	20
01	2	14 – 1 = 13	19	25	14
00	2 + 1 = 3	13	19	25	2
11	3	13	19	25 + 1 = 26	25

Продовження таблиці 2.2

1	2	3	4	5	6
01	3	13 – 1 = 12	19	26	13
00	3 + 1 = 4	12	19	26	3
10	4	12	19 – 1 = 18	26	19
11	4	12	18	26 + 1 = 27	27
01	4	12 – 1 = 11	18	27	12
01	4	11	18	27 + 1 = 28	27
00	4 + 1 = 5	11	18	27	4
00	5 + 1 = 6	11	18	27	5
10	6	11	18 – 1 =17	27	18
01	6	11 – 1 = 10	17	27	11
11	6	10	17	27 + 1 = 28	27
00	6 + 1 = 7	10	17	28	6
11	7	10	17	28 + 1 = 29	28
10	7	10	17 – 1 = 16	28	17
01	7	10 – 1 = 9	16	28	10
00	7 + 1 = 8	9	16	28	8
11	8	9	16	28 +1 = 29	28
11	8	9	16	29 + 1 = 30	29
10	8	9	16 – 1 = 15	30	16
01	8	9 – 1 = 8	15	30	9
11	8	8	15	30 + 1 = 31	30
01	8	8 – 1 = 7	15	31	8
11	8	7	15	31 + 1 = 32	31

Кожен лічильник працює поки не досягне свого максимума при прямій лічбі або мінімума при зворотній. Якщо один з лічильників досяг свого мінімуму або максимуму ітерація генерування бітів запускається повторно поки не буде задіяний інший лічильник. Коли всі лічильники досягнуть граничних значень процес перестановки завершується.

Оскільки для роботи лічильників необхідно генерувати псевдовипадкову послідовність обрано генератор на основі регістрів зсуву з лінійно зворотним зв’язком.

3.3 Генератор псевдовипадкових послідовностей

Найважливішим класом псевдовипадкових послідовностей (ПВП) є послідовності, що формуються генераторами на основі регістрів зсуву з лінійно зворотним зв’язком. Даний клас генераторів псевдовипадкових послідовностей заснований на ідеї перетворення двійкового представлення деякого числа [12, 13].

Основними перевагами цих генераторів є [12, 14, 15]:

- використання найпростіших операцій, апаратно реалізованих практично на всіх обчислювальних пристроях;
- простота апаратної та програмної реалізації;
- можливість на їх основі побудови складніших генераторів;
- висока швидкодія створених на їх основі криптографічних алгоритмів;
- рівномірний закон розподілу, ймовірності появи символів;
- велика кількість теоретичних досліджень властивостей лінійно рекурентних послідовностей, що свідчать про задовільні криптографічні властивості.

Регістр зсуву зі зворотним лінійним зв’язком – мікросхема з комірками пам’яті, в які записаний один біт інформації. Комірки пам’яті мають по одному входу і виходу. Кількість комірок p називають довжиною регістра. Друга складова – функція зворотного зв’язку $f(x_1, x_2, \ldots, x_p)$. Для лінійного регістра – це операція XOR над деякими його бітами (рис. 2.4).

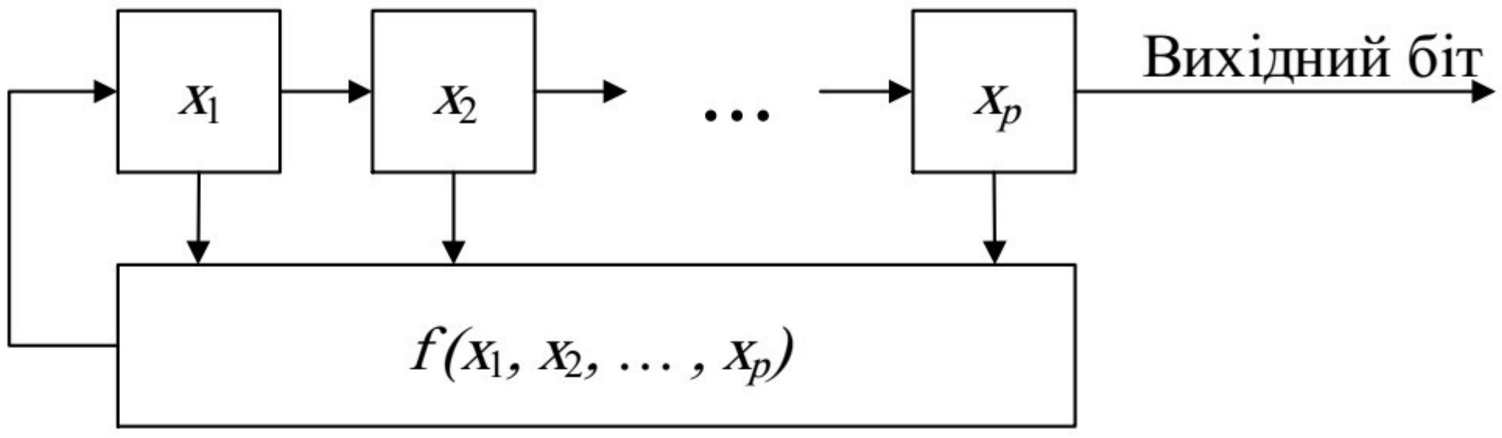


Рисунок 2.4 – Загальна схема регістру зсуву з лінійно зворотним зв’язком

Для побудови двійкового РЗЛЗЗ використовують твірні поліноми. Степінь поліному визначає розрядність регістру зсуву, а не нульові коефіцієнти – характер зворотних зв’язків. Поліном із степенем p має наступний вигляд:

$$\Phi(x) = a_p x^p + a_{p-1} x^{p-1} + \cdots + a_2 x^2 + a_1 x + a_0, a_i \in \{0,1\}.$$

При $a_i = 1$ множення на a_i вказує на наявність зв’язку, при $a_i = 0$ множення на a_i вказує на відсутність зв’язку.

Існують два варіанти реалізації РЗЛЗЗ: конфігурація Галуа (рис 2.5) та Фібоначчі (рис 2.6).

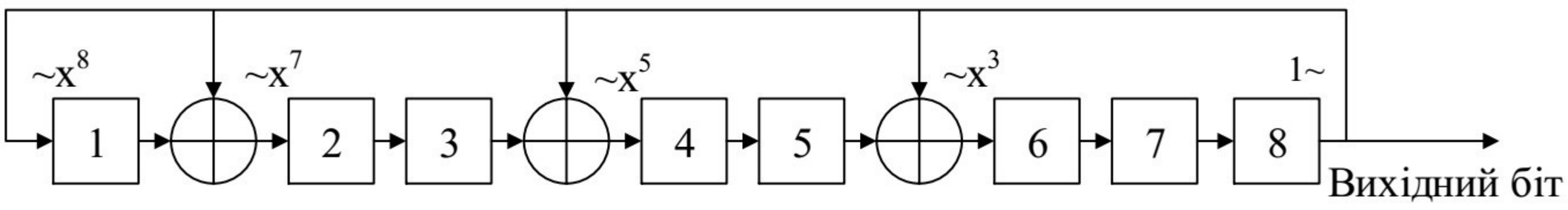


Рисунок 2.5 – Конфігурація Галуа для $\Phi(x) = x^8 + x^7 + x^5 + x^3 + 1$

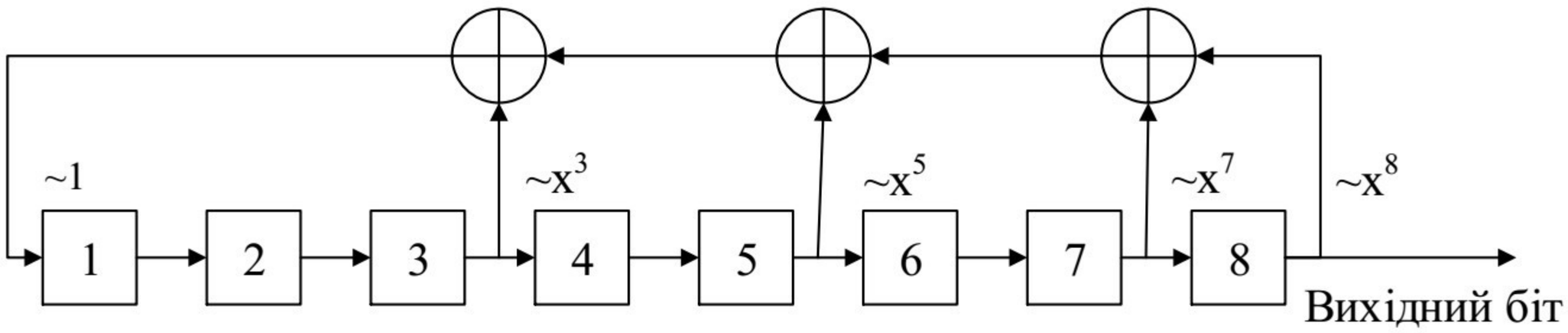


Рисунок 2.6 – – Конфігурація Фібоначчі для $\Phi(x) = x^8 + x^7 + x^5 + x^3 + 1$

Обидві конфігурації не відрізняються крипостійкістю. Конфігурація Галуа легша в програмній реалізації та має більшу швидкодію: всі операції XOR можна виконати за один такт за рахунок розпаралелення, а не послідовно, як в конфігурації Фібоначчі.

3.4 Формування складових учасників

Для формування складових учасників використовуються частини отримані в результаті поділу масиву ME на n рівних частин $Q = (Q_1, Q_2, Q_3..., Q_n)$.

Процес формування складових частин складається з таких етапів:

1) Формується двовимірний масив $QD_{i\times j}$ де $i = n-(k-1)$, $j = n$.

$$QD = \begin{bmatrix} Q_1 & Q_2 & Q_3 & \cdots & Q_{i-1} & Q_i \\ Q_2 & Q_3 & Q_4 & \cdots & Q_i & Q_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_i & Q_1 & Q_2 & \cdots & Q_{i-2} & Q_{i-1} \end{bmatrix}$$

Кожний рядок містить частини записані в унікальному не повторюваному порядку.

2) На наступному кроці над кожними двома сусідніми елементами в рядку крім першого, якщо i непарне, та крім першого та другого, якщо i парне здійснюється додавання. Також перший доданок множиться на коефіцієнт $K_{i,m} = 1,2,3,..., k$. Всі обчислення виконуються за модулем 256.

Вигляд масиву, коли i непарне

$$QE = \begin{bmatrix} Q_1 & K_{2,1}\cdot Q_2 + Q_3 & \cdots & K_{i-1,1}\cdot Q_{i-1} + Q_i \\ Q_2 & K_{3,1}\cdot Q_3 + Q_4 & \cdots & K_{i,1}\cdot Q_i + Q_1 \\ \cdots & \cdots & & \cdots \\ Q_i & K_{1,m}\cdot Q_1 + Q_2 & \cdots & K_{i-2,m}\cdot Q_{i-2} + Q_{i-1} \end{bmatrix}$$

Вигляд масиву, коли i парне

$$QE = \begin{bmatrix} Q_1 & Q_2 & K_{3,1}\cdot Q_3 + Q_4 & \cdots & K_{i-1,1}\cdot Q_{i-1} + Q_i \\ Q_2 & Q_3 & K_{4,1}\cdot Q_4 + Q_5 & \cdots & K_{i,1}\cdot Q_i + Q_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_i & Q_1 & K_{2,m}\cdot Q_2 + Q_3 & \cdots & K_{i-2,m}\cdot Q_{i-2} + Q_{i-1} \end{bmatrix}$$

3) На основі рядків матриці QE формуються складові учасників. В результаті буде отримано n унікальних частин.

Наприклад при $n=7$ та $k=3$ складові учасників матимуть такий вигляд

$$QE = \begin{bmatrix} Q_1 & 1\cdot Q_2 + Q_3 & 1\cdot Q_4 + Q_5 \\ Q_2 & 1\cdot Q_3 + Q_4 & 1\cdot Q_5 + Q_6 \\ Q_3 & 2\cdot Q_4 + Q_5 & 1\cdot Q_6 + Q_7 \\ Q_4 & 2\cdot Q_5 + Q_6 & 1\cdot Q_7 + Q_1 \\ Q_5 & 2\cdot Q_6 + Q_7 & 1\cdot Q_1 + Q_2 \\ Q_6 & 2\cdot Q_7 + Q_1 & 2\cdot Q_2 + Q_3 \\ Q_7 & 2\cdot Q_1 + Q_2 & 2\cdot Q_3 + Q_4 \end{bmatrix}.$$

Для відновлення початкових частин $Q=(Q_1, Q_2, Q_3 ..., Q_n)$ коли зберуться k учасників, необхідно розв’язати рівняння маючи відомі частини.

Наприклад коли зберуться учасники 2, 4 та 7 будуть відомі такі частини:

$$\begin{bmatrix} Q_2 & 1\cdot Q_3 + Q_4 & 1\cdot Q_5 + Q_6 \\ Q_4 & 2\cdot Q_5 + Q_6 & 1\cdot Q_7 + Q_1 \\ Q_7 & 2\cdot Q_1 + Q_2 & 2\cdot Q_3 + Q_4 \end{bmatrix},$$

відомо три частини Q_2, Q_4 та Q_5, необхідно розв’язати такі рівняння:

$$Q_1 = (1 \cdot Q_7 + Q_1) - Q_7,$$

$$Q_3 = (1 \cdot Q_3 + Q_4) - Q_4,$$

$$Q_5 = (2 \cdot Q_5 + Q_6) - (1 \cdot Q_5 + Q_6),$$

$$Q_6 = (1 \cdot Q_5 + Q_6) - Q_5.$$

Розв’язавши рівняння отримаємо усі частини.

3.5 Особливості використання методу для зображень

Метод розподілу буде розглянуто для зображень у форматі BMP. Це пов’язано з простою внутрішньою будовою BMP, а саме відсутність складних алгоритмів кодування та ущільнення, пікселів що спрощую обробку та аналіз зображень цього типу, також є велика кількість готових бібліотек по обробці BMP, що спростить програмну реалізацію запропонованого методу.

При роботі з BMP секретом буде масив кольорів:

$$M = \{r_1, g_1, b_1, r_2, g_2, b_2, ..., r_s, g_s, b_s\}$$

де r – червона складова кольору,

g – зелена складова кольору,

b – синя складова кольору,

s – кількість пікселів зображення.

Після розподілу отримаємо матрицю QD де на основі кожного з рядків буде сформовано «тіньові» зображення, маючи які можна відновити початкове зображення.

При збережені зображень необхідно доповнити сформовані частини таким чином щоб їхній розмір обраховувався за формулою

$$size = width * height * 3$$

де $size$ – розмір масиву кольорів,;

$width$ – ширина зображення;

height – висота зображення.

Для цього в кінець кожного масиву допишуться кольори з початку масиву, а також в кінці буде вставлено реальний розмір сформованої частки.

Висота для зображень буде братися така сама як в оригінальному зображені, а ширина буде обраховуватися за формулою:

$$width = \left\lfloor \frac{size + 4}{height * 3} \right\rfloor .$$

Розмір даних необхідний, щоб дописати у кінець масиву обраховується за формулою $sizeData = height * width * 3 - (size + 4)$.

Розглянемо детальніше структуру BMP файлу.

3.6 Структура BMP файлу

BMP (англ. BitMap Picture) – апаратно-незалежне побітове зображення Windows. Підтримується будь-якими Windows-сумісними програмами. Структура файлу BMP використовується Windows для зберігання растрових зображень. Наприклад, в цьому форматі зберігаються малюнки фону, піктограми та інші растрові зображення Windows. Формат зводить до мінімуму ймовірність помилок або неправильної інтерпретації растрових даних [16, 17].

У даному форматі можна зберігати тільки одношарові растри. На кожен піксель в різних файлах може приходити різна кількість біт (глибина кольору). Формат підтримує бітності 1, 2, 4, 8, 16, 24, 32, 48 і 64 [18].

Основним недоліком даного формату є те, що тільки версії формату з 4 та 8 бітовим кольором піддаються ущільненню. Отже, 24 бітові файли BMP будуть дуже великими.

Крім того, застосування файлів BMP обмежене платформами Windows і OS / 2. Все це унеможливлює застосування даного формату в мережі інтернет.

Кожний файл BMP містить заголовок файлу, заголовок зображення, растрові дані, а також може містити таблицю кольорів для зображень де на збереження кольору припадає менше 24 біт [19].

Заголовок файлу BMP містить інформацію про тип і розмір, а також про розташування в ньому даних. Після заголовка файлу знаходиться інформація про розмір, кольори та тип ущільнення зображення. Якщо зображення складається з 1, 4, 8 - бітових пікселів, то повинна використовуватися таблиця кольорів.

За таблицею з кольорами знаходяться растрові дані. Вони зберігаються у вигляді 1, 4, або 8-бітових індексів або у вигляді буквених 24-бітових даних системи RGB. Пікселі в растрі записуються, починаючи з нижнього лівого кута зображення, і читаються зліва направо і знизу вгору.

Структуру BMP файлу зображено в таблиці 2.3.

Таблиця 2.3 – Структура BMP файлу

Ім'я	Довжина	Зміщення	Опис
Заголовок файлу (BitMapFileHeader)			
Type	2	0	Сигнатура файлу «BM»
Size	4	2	Розмір файлу
Reserved 1	2	6	Зарезервоване поле
Reserved 2	2	8	Зарезервоване поле
OffsetBits	4	10	Зміщення зображення від початку файлу
Інформаційний заголовок (BitMapInfoHeader)			
Size	4	14	Довжина заголовку
Width	4	18	Ширина зображення
Height	4	22	Висота зображення
Planes	2	26	Кількість шарів
BitCount	2	28	Глибина кольору, біт на піксель
Compression	4	30	Тип ущільнення
SizeImage	4	34	Розмір зображення, байт
XpelsPerMeter	4	38	Кількість пікселів на метр по горизонталі
YpelsPerMeter	4	42	Кількість пікселів на метр по вертикалі
ColorsUsed	4	46	Розмір таблиці кольорів
ColorsImportant	4	50	Кількість основних кольорів

Таблиця кольорів (палітра) (ColorTable)			
ColorTable	1024	54	256 елементів по 4 байта
Дані зображення (BitMap Array)			
Image	Size	1078	Зображення, записане по рядках зліва на правота знизу верх

В програмному додатку метод розділення секрету буде реалізований для 24 бітних зображень тому, що вони зберігаються в не ущільненому форматі та такі зображення найлегше обробляти.

4 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

4.1 Вимоги до програмного засобу

У магістерській кваліфікаційній роботі передбачено створення програмного додатку, що дозволить розділяти та відновлювати секрет збережений у вигляді BMP – зображення з використанням запропонованого методу.

Передбачається, що програма працюватиме з терміналу, тобто графічний інтерфейс буде відсутній. Тому для того, щоб програма повністю виконувала поставлену задачу, і була зручною у використанні, вона має відображати коротку змістовну інформацію у вигляді текстових повідомлень.

Враховуючи всі вимоги, для повного і нормально функціонування програми необхідно:

1) розробити загальну структуру функціонування програми;
2) розробити алгоритм розділення та відновлення;
3) розробити алгоритми генерування псевдовипадкових послідовностей;
4) розробити алгоритм перестановок початкових даних;
5) розробити алгоритм формування складових учасників;
6) протестувати програму на правильність роботи.

4.2 Структура програми

Перед написанням програмного додатку необхідно розробити загальну схему роботи програмного додатку. Передбачається, що програма матиме такі функції:

- генерування ключа на основі вказаних параметрів;
- розподілення на частини вказаного зображення;
- відновлення оригінального зображення.

Відповідно до описаних функцій схема роботи програми має вигляд (рис. Рисунок 3.1 – Алгоритм роботи програми):

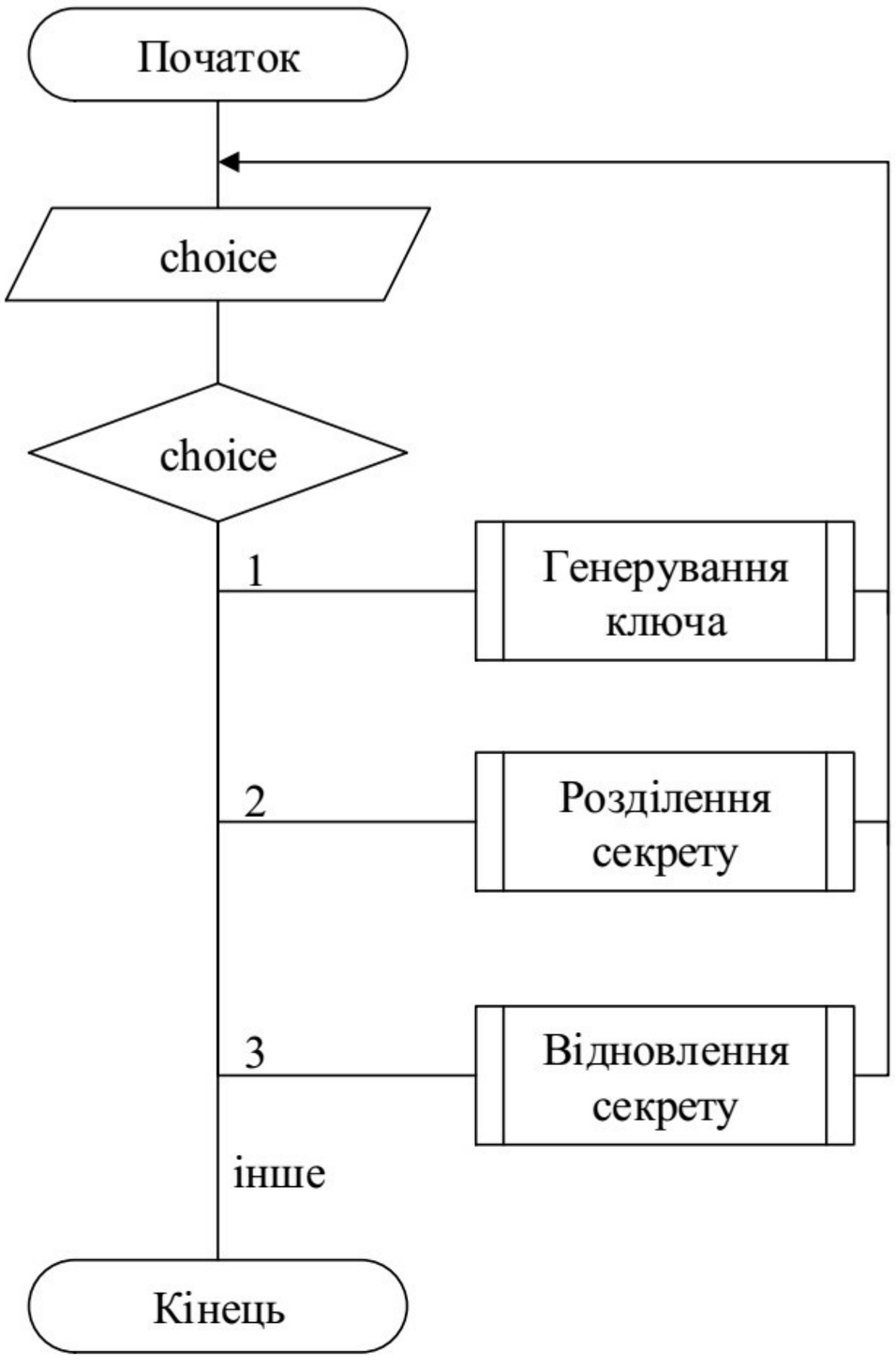


Рисунок 3.1 – Алгоритм роботи програми

При запуску програми користувач матиме можливість обрати одну із перерахованих функцій або завершити роботу з програмою.

При виборі «Генерування ключа» користувачу буде запропоновано вести дані необхідні для генерування ключа, а саме N – кількість, на яку потрібно розділити секрет, K – мінімальна кількість частин для відновлення, кількість лічильників. Також користувачу буде необхідно вести шлях для збереження ключа.

При виборі «Розділення секрету» користувачу буде запропоновано вести шлях до зображення, шлях для збереження розподілених частин та шлях до ключа або згенерувати новий. Після ведення необхідних даних програма має розподілити секрет та зберегти отриманий результат до вказаної дерикторії.

При виборі «Відновлення секрету» користувачу буде необхідно вести шлях до файлу з ключем та мінімальної кількості розділених частин після чого програма відновить оригінальне зображення та збереже у вказану дерикторію.

4.3 Опис алгоритму генерування псевдовипадкових послідовностей

Згідно розробленого методу для генерування псевдовипадкових послідовностей буде використовуватися регістри зсуву з лінійно зворотним зв’язком конфігурації Галуа. Така конфігурація легша в програмній реалізації та має більшу швидкодію: всі операції XOR можна виконати за один такт за рахунок розпаралелення, а не послідовно, як в конфігурації Фібоначчі (рис 3.2).

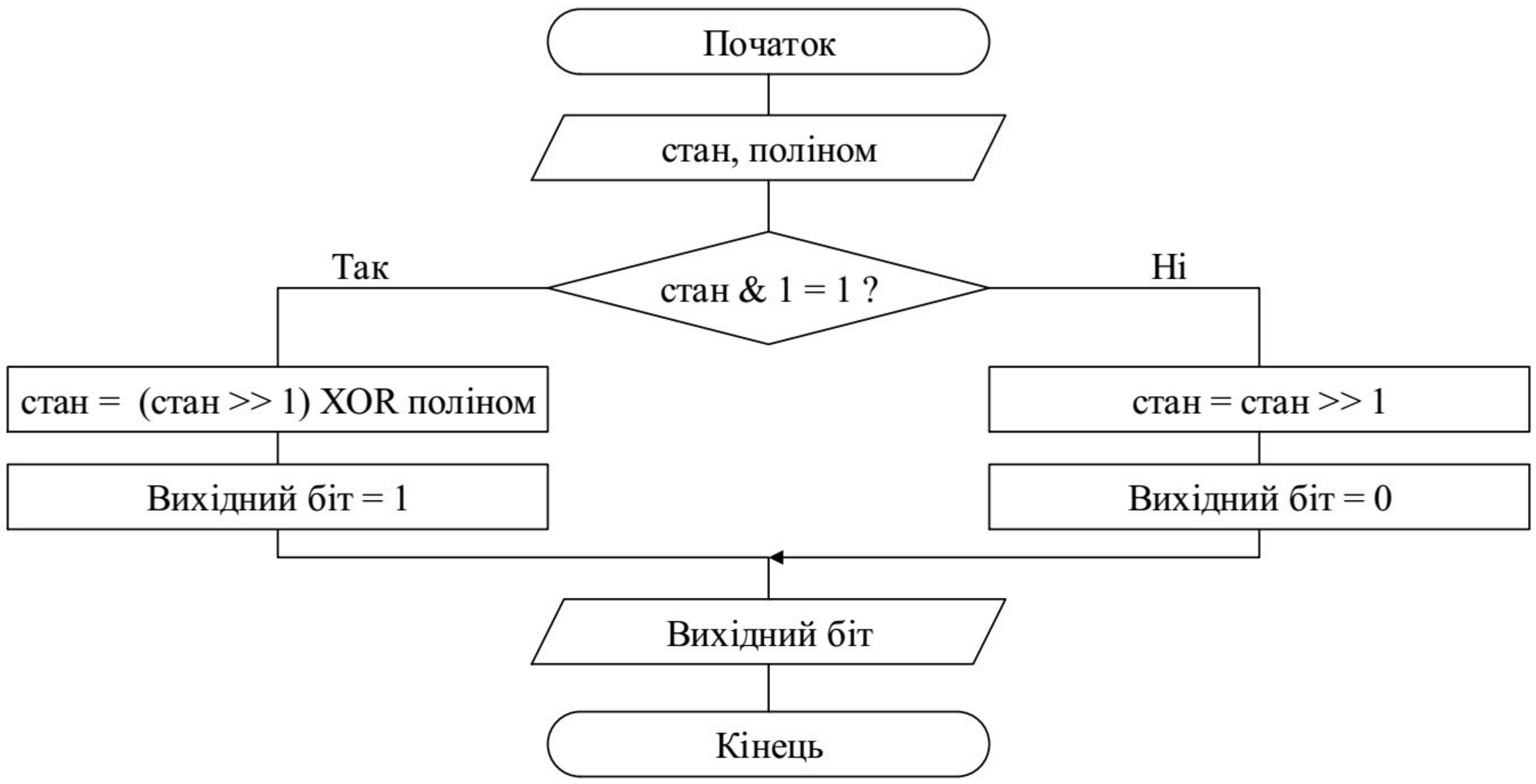


Рисунок 3.2 – Схема генерування одного біту

Для генерування послідовностей буде використано генератор з використанням трьох регістрів зсуву (рис 3.3).

Рисунок 3.3 – Схема генератора

В якості функції об’єднання буде використано операцію XOR. Для трьох РЗЛЗЗ вихідний біт можна представити наступним чином

$$b(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$$

де x_1 - вихідний біт РЗЛЗЗ-1, x_2 - вихідний біт РЗЛЗЗ-2, x_3 - вихідний біт РЗЛЗЗ-3.

Теоретично-системний аналіз показує, що загальний період рівний

$$T = T_1 \cdot T_2 \cdot T_3,$$

де T_1, T_2, T_3 - період першого, другого та третього РЗЛЗЗ.

Лінійна складність для трьох РЗЛЗЗ розраховується за формулою

$$L = L_1 \cdot L_2 + L_2 \cdot L_3 + L_1 \cdot L_3,$$

де L_1, L_2, L_3 - довжина першого, другого та третього РЗЛЗЗ.

В програмі використовуватимуться регістри розрядністю 21, 23 та 32. Відповідно загальний період рівний

$$T = T_1 \cdot T_2 \cdot T_3 = (2^{32} - 1) \cdot (2^{23} - 1) \cdot (2^{21} - 1) = 75\ \ 557\ \ 818\ \ 672\ \ 330\ \ 169\ \ 122\ \ 815$$

Лінійна складність рівна

$$L = L_1 \cdot L_2 + L_2 \cdot L_3 + L_1 \cdot L_3 = 32 * 23 - 23 * 21 + 21 * 32 = 1891.$$

4.4 Опис алгоритму генерування ключа

Для операцій розділення/відновлення секрету необхідний ключ. До складу ключа входять такі параметри:

- N – кількість учасників;
- K – мінімальна кількість учасників для відновлення;
- масив з порядком функціонування для кожного з лічильників, 0 – пряма лічба або 1 – зворотня;
- початкові стани та поліноми для 21, 23 та 32 розрядного РЗЛЗЗ.

Ключ зберігатиметься у файл і повинен бути доступним лише для учасників розподілу секрету. Передбачається збереження даних в форматі

JSON. Це дозволить переглядати вміст та корегувати його при тестуванні програми. Алгоритм генерування ключа показано на рисунку 3.4.

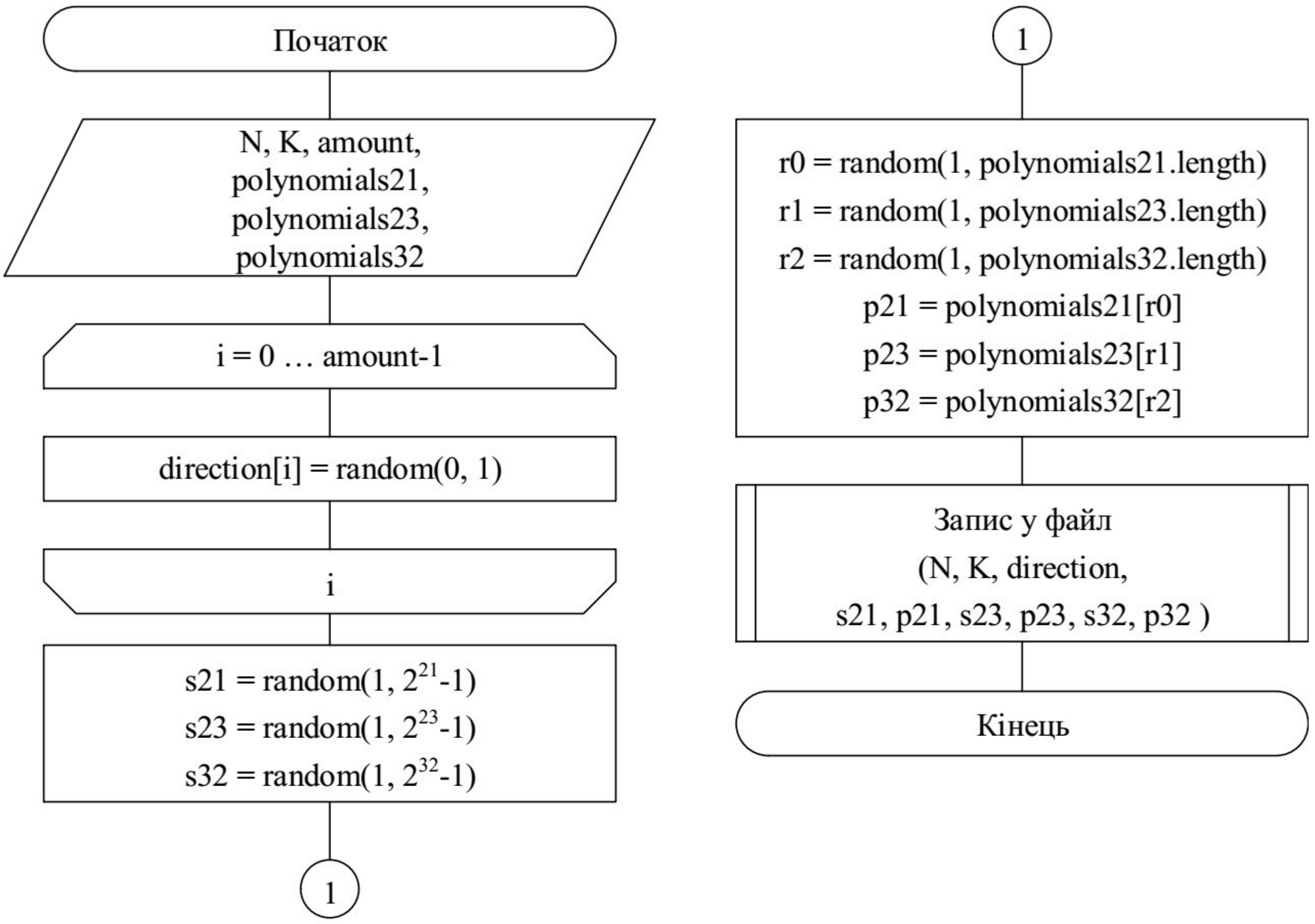


Рисунок 3.4 – Алгоритм генерування ключа

На вході від користувача отримуємо N, K, amount – кількість лічильників та масиви поліномів для 21, 23 та 32 розрядних РЗЛЗЗ. Для зручності передбачається, що поліноми будуть зберігатися у файлі та читатися з фалу при необхідності.

Після чого створюється direction – масив для збереження порядку функціонування кожного з лічильників та заповнюється випадковими значеннями 0 або 1. Далі випадковим чином генеруються початкові стани для РЗЛЗЗ s21, s23 та s32 та поліноми p21, p23, p32. Згенеровані дані записуються у файл.

4.5 Опис алгоритму розділення секрету

Алгоритм розподілу секрету показано на рисунку 3.5.

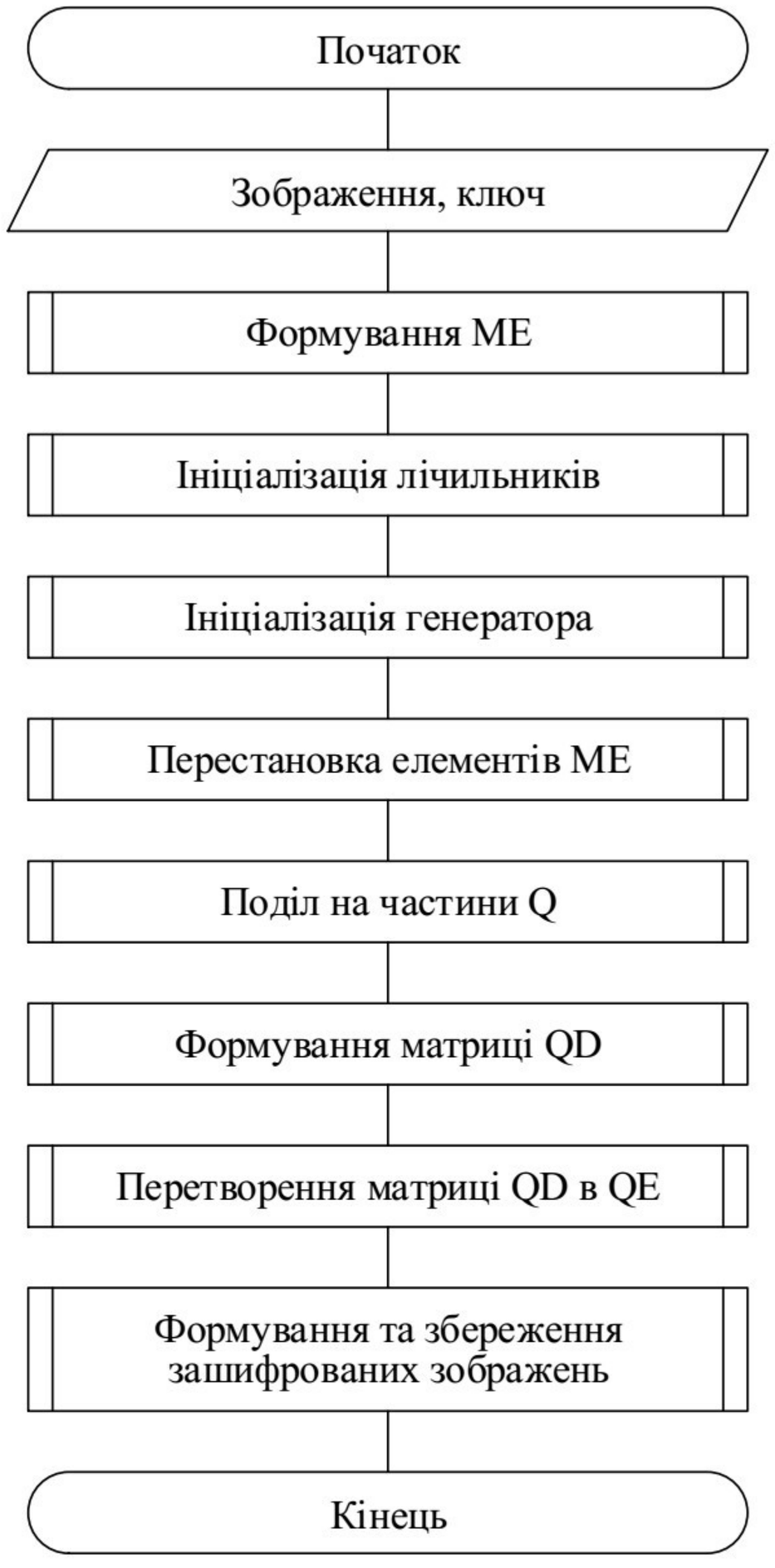


Рисунок 3.5 – Алгоритм розподілу секрету

На вхід подаються зображення для розподілу та ключ. На основі зображення з кольорів формується масив *ME*.

Далі ініціалізуються лічильники та генератор на основі ключа. Після ініціалізації елементи масиву *ME* переставляються місцями.

По завершенню перестановок *ME* ділиться на частини та формується матриця *QD* котра перетворюється в матрицю *QE*.

На основі матриці *QE* формуються та зберігаються зашифровані зображення.

4.6 Опис алгоритму відновлення секрету

Алгоритм розподілу секрету показано на рисунку 3.6.

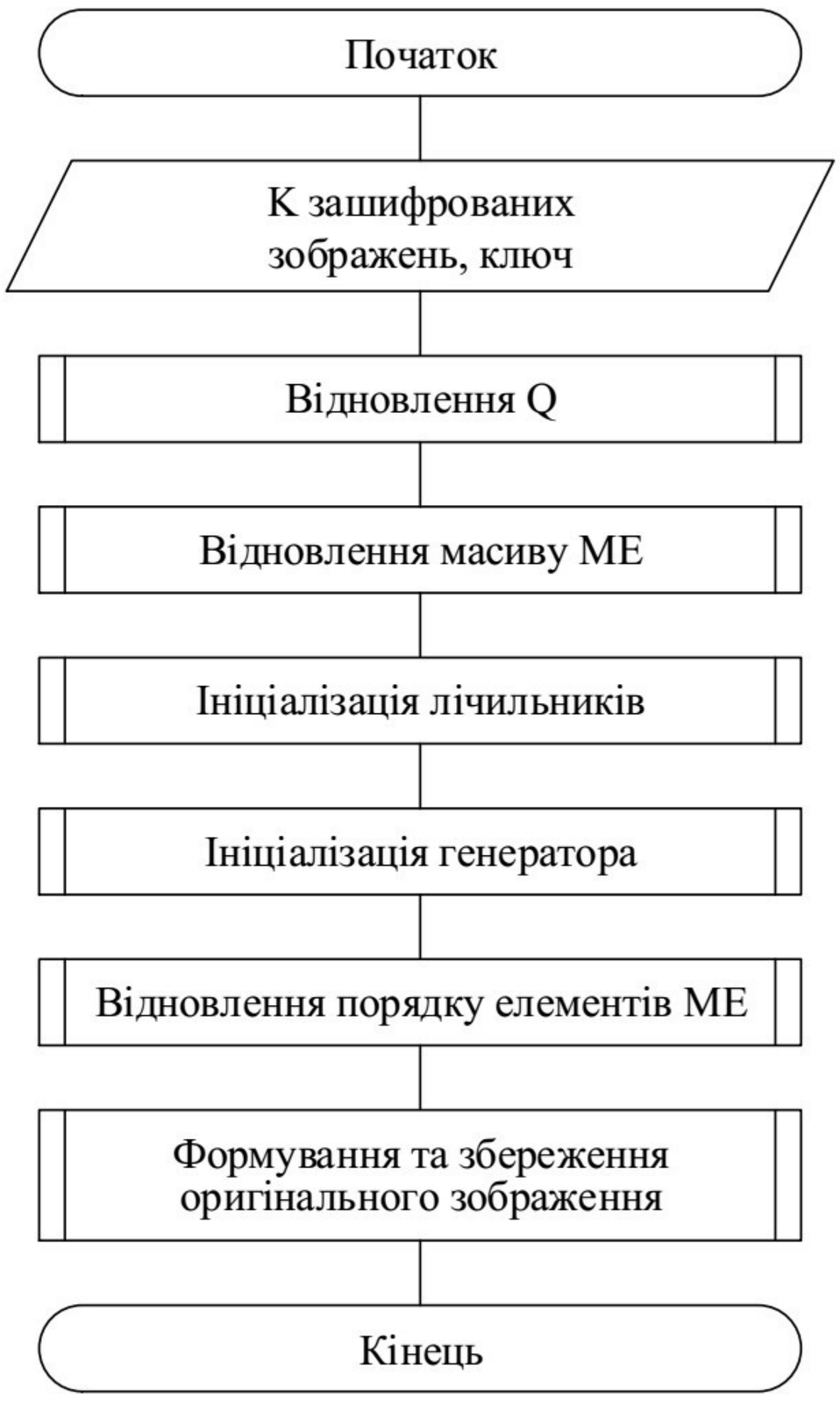


Рисунок 3.6 – Алгоритм відновлення секрету

На вхід подаються *K* зашифрованих зображень та ключ. На основі *K* зображень відновлюються частини $Q=(Q_1, Q_2, Q_3..., Q_n)$. З частин *Q* відновлюється масив *ME* та відновлюється порядок елементів в ньому.

На основі *ME* формується та зберігається оригінальне зображення.

4.7 Результати роботи програми

Після розробки програмного додатку необхідно перевірити коректність його роботи.

Головне меню додатку показано на рисунку 3.7.

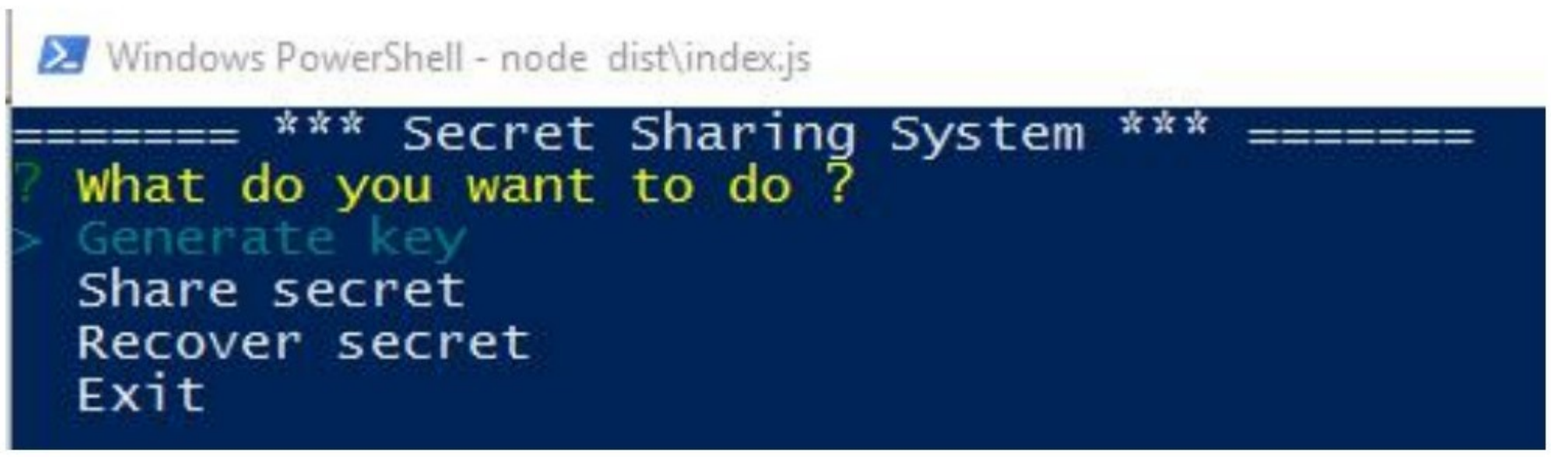


Рисунок 3.7 – Вигляд головного меню додатку

Вибір пунктів меню здійснюється за допомогою стрілок верх/вниз та підтвердження вибору натисненням клавіші «Enter».

При виборі пункту «Generate key» (рис. 3. 8) буде згенеровано ключ на основі ведених даних та збережено у файл.

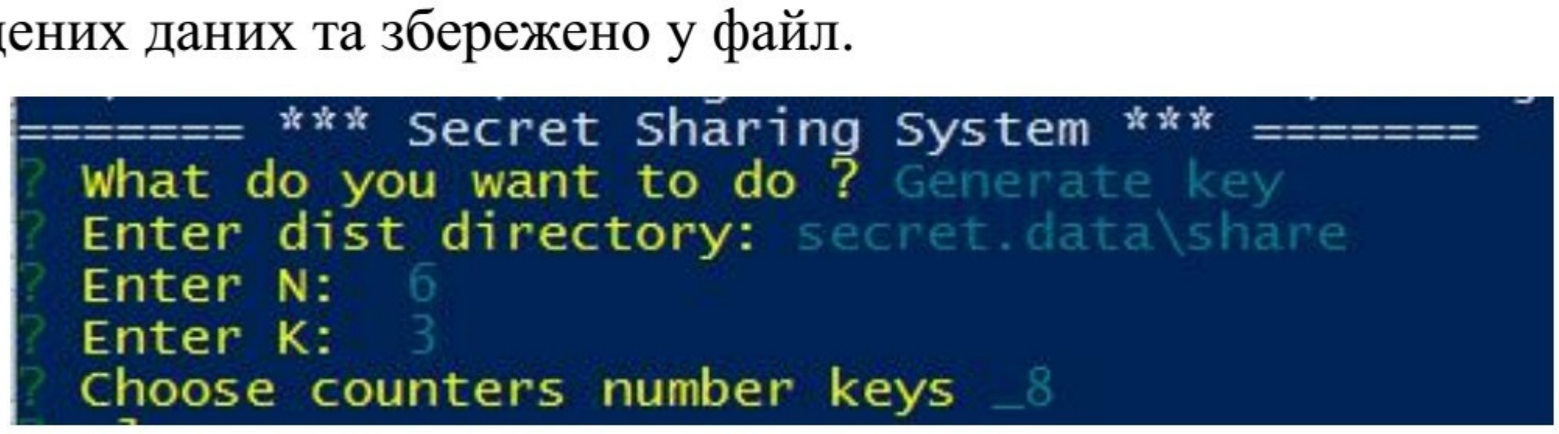


Рисунок 3.8 – Ведення даних для генерування ключа

Згенерований ключ зображено на рисунку 3.9.

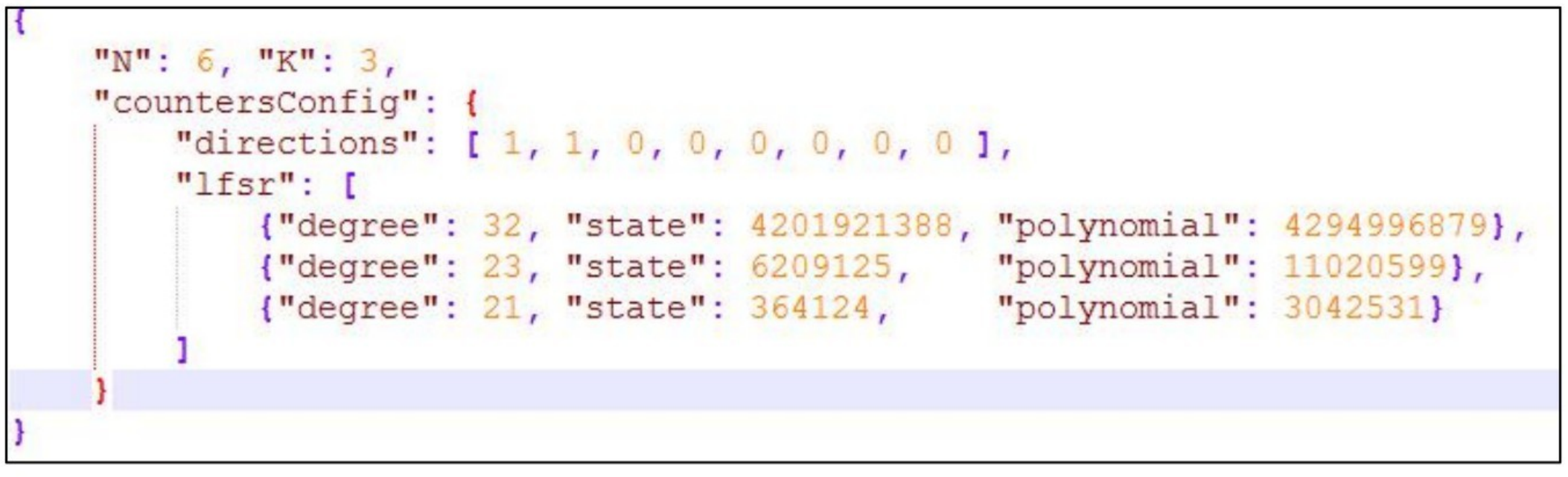


Рисунок 3.9 – Вигляд ключа

При виборі пункту «Share secret» (рис. 3. 10) буде запропоновано вести шлях до зображення, ключа та теки для зберігання зашифрованих зображень.

Розділимо картинку на основі згенерованого ключа.

```
======= *** Secret Sharing System *** =======
? What do you want to do ? Share secret
? Enter path to source file: secret.data\boy.bmp
? Enter path to key file: secret.data\share\key_N_6_K_3__8.json
? Enter dist directory: secret.data\share
```

Рисунок 3.10 – Ведення даних для розділення секрету

Оригінальне зображення показано на рисунку 3.11.

Рисунок 3.11 – Вигляд оригінального зображення

Після розподілу отримаємо шість зашифрованих зображень (3.12).

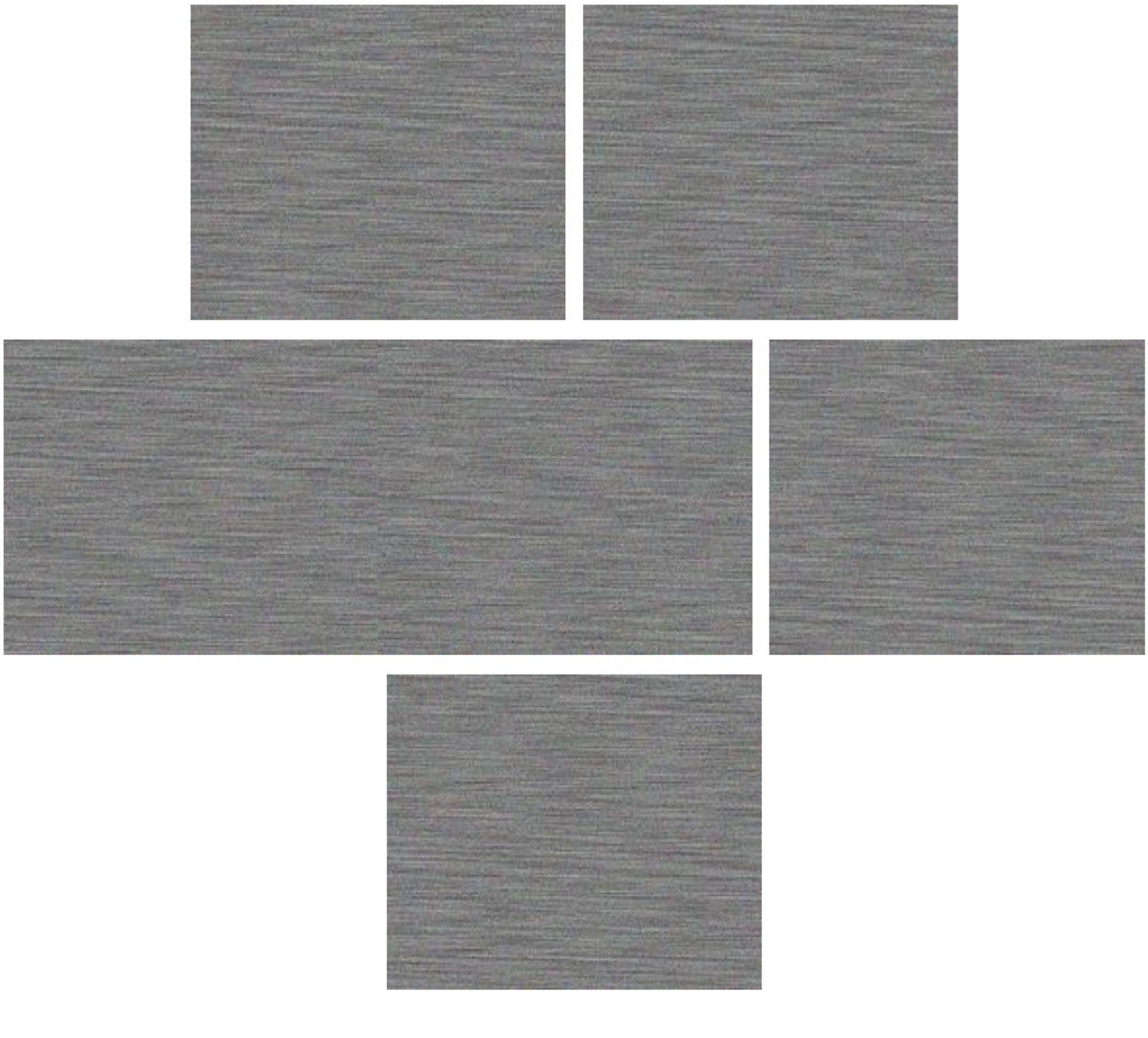


Рисунок 3.12 – Вигляд зашифрованих зображень з використанням 8 лічильників

Згенеруємо ще один ключ на 64 лічильники та ще раз розділимо оригінальне зображення.

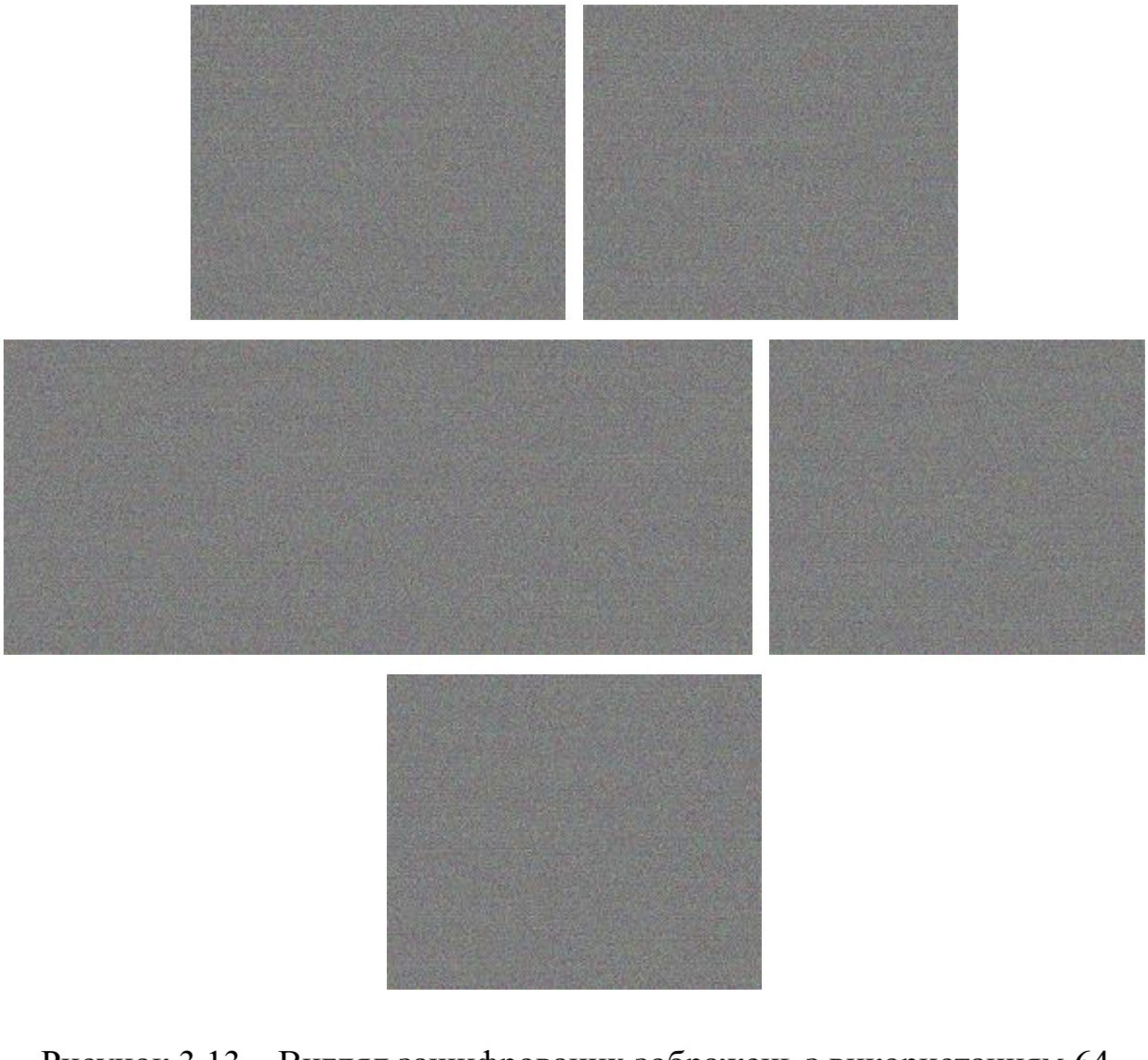


Рисунок 3.13 – Вигляд зашифрованих зображень з використанням 64 лічильників

При порівнянні на зображеннях зашифрованих з використанням 8 лічильників видно повторювані лінії, а на зображеннях зашифрованих з використанням 64 лічильників лінії відсутні. Це свідчить про те, що збільшення лічильників покращує перемішування елементів секрету та потребує подальшого дослідження та аналізу.

При виборі пункту «Recover secret» (рис. 3. 14) буде запропоновано вести шлях до ключа та зашифрованих частин та теки для зберігання відновленого зображення. Після відновлення отримаємо початкове зображення.

```
======= *** Secret Sharing System *** =======
? What do you want to do ? Recover secret
? Enter path to key file: secret.data\share\64\key_N_6_K_3__64.json
? Enter path to part #1: secret.data\share\64\1.share.boy.bmp
? Enter path to part #2: secret.data\share\64\5.share.boy.bmp
? Enter path to part #3: secret.data\share\64\4.share.boy.bmp
? Enter dist directory: secret.data\share\64
```

Рисунок 3.14 – Ведення даних для відновлення секрету

Змінемо в ключі з 8 лічильниками напрямок всіх лічильників крім останього на протилежний та спробуємо відновити зображення (рис. 3.15).

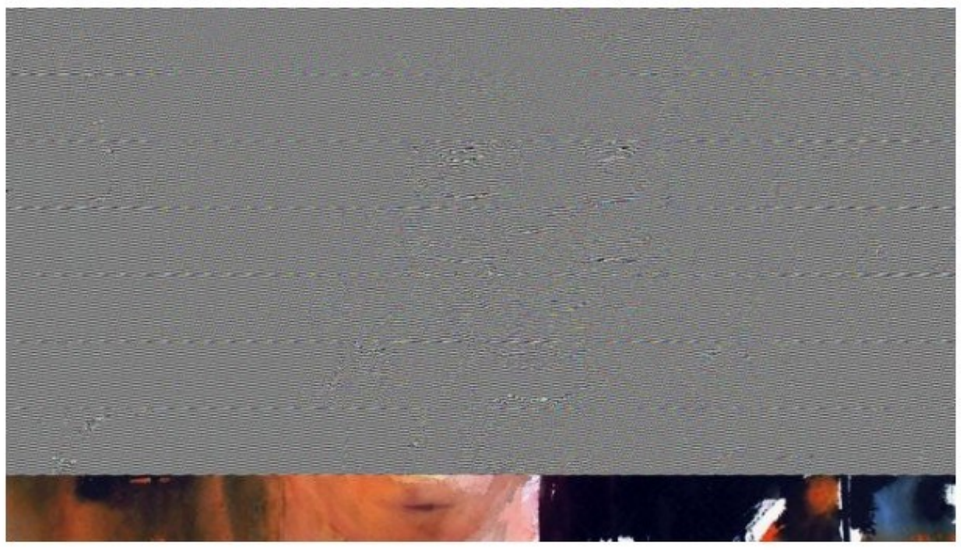


Рисунок 3.15 – Вигляд відновленого зображення за допомогою хибного ключа

При спробі відновити зображення маючи лише дві частини, а третя частина замінена чорним квадратом отримаємо зображення на якому видні лише деякі фрагменти оригінального зображення.

Рисунок 3.16 – Вигляд відновленого зображення

В результаті отримали зображення частина якого буде містини оригінальне зображення, а інша частина міститиме кольорові шуми.

Отже програмний додаток був протестований на коректність роботи. Доведено що програма розподіляє та відновлює вміст зображень правильно.

5 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини магістерської кваліфікаційної роботи є обґрунтування економічної доцільності розробки методу та засобу розподілу секрету за допомогою зображень, для цього необхідно виконати такі етапи робіт:

- оцінити комерційний потенціал розробки;
- спрогнозувати витрати на виконання наукової роботи та впровадження її результатів;
- спрогнозувати комерційній ефект від реалізації результатів розробки;
- розрахувати ефективність вкладених інвестицій та період їх окупності.

5.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Об’єктом дослідження магістерської кваліфікаційної роботи є метод та засіб розподілу секрету за допомогою зображень.

Для проведення технологічного аудиту було залучено трьох незалежних експертів: Баришев Юрій Володимирович, Каплун Валентина Аполінаріївна, Куперштейн Леонід Михайлович. Баришев Ю.В.- к.т.н. ст. вик. кафедри ЗІ, Каплун В.А. – ст. вик. Кафедри ЗІ, Куперштейн Л.М. – к.т.н., доц. кафедри ЗІ. Кожен з експертів повинен ознайомитися з запропонованою розробкою, та заповнити таблицю, яка визначає рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можливу оцінку в балах. Після виконання цього, підраховується середньоарифметична сума балів та визначається який рівень комерційного потенціалу має нова розробка.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ю критеріями, наведеними в таблиці 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	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 роботоздатність 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 аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
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 ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
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 фінансуванн я є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуютьс я у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Баришев Ю.В	Каплун В.А.	Куперштейн Л.М.
	Бали, виставлені експертами:		
1	3	4	4
2	2	2	2
3	3	4	3
4	2	3	3
5	3	3	2
6	1	1	2
7	3	2	3
8	4	4	4
9	4	4	4
10	4	4	3
11	4	4	4
12	4	4	4
Сума балів	$СБ_1$=37	$СБ_2$=39	$СБ_3$=38
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^i СБ_i}{i} = \frac{37+39+38}{3} = 38$		

Отже, з отриманих даних таблиці 4.2 видно, що середньоарифметична сума балів дорівнює 38, тобто нова розробка має рівень комерційного потенціалу вище середнього.

На ринку майже не існує засобів для розподілу секретної візуальної інформації. Тому є гостра необхідність в розробці нових методів та засобів для розподілу секретного вмісту зображень. Також існуючі методи мають недоліки пов’язані з неповним відновленням інформації і складними математичними обрахунками в ході роботи. А за результатами проведеного дослідження розроблений метод і засіб позбавлений цих недоліків.

5.2 Прогнозування витрат на виконання науково – дослідної роботи та конструкторсько – технологічної роботи.

Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи може складатися з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.

2-й етап: розрахунок загальних витрат на виконання даної роботи.

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Розрахунок витрат на розробку методики дослідження здійснюється по статтях калькуляції, а саме:

- основна заробітна плата розробників;
- додаткова заробітна плата розробників;
- нарахування на заробітну плату розробників;
- амортизація персонального комп'ютера;
- витрати на матеріали, що були використані в процесі розробки програмного продукту;
- витрати на електроенергію;

- інші витрати.

У розробці даного програмного продукту брав участь один спеціаліст та один науковий керівник. Основна заробітна плата для спеціаліста визначається за формулою:

$$З_о = \frac{М}{Т_р} t$$

де М- місячний посадовий оклад конкретного розробника, наукового керівника;

Тр - кількість робочих днів у місяці, Тр = 22 дня;

t - число днів роботи.

Оплата за робочий день розробника проекту становить 454,55 грн, а наукового керівника – 545,45 грн.

Розрахунки заробітної плати для наведені в таблиці 4.3.

Заробітна плата наукового керівника:

$$З_{О_{НК}} = \frac{12000}{22} \times 25 = 545{,}45 \times 25 = 13636{,}25 (грн)$$

Заробітна плата розробника:

$$З_{О_Р} = \frac{10000}{22} \times 74 = 454.55 \times 74 = 33636{,}7 \ (грн)$$

Витрати на оплату праці, основна заробітна плата:

$$З_О = З_{О_{НК}} + З_{О_Р} = 13636{,}25 + 33636{,}7 = 47272{,}95 \ (грн)$$

Таблиця 4.3 – Розрахунки основної заробітної плати спеціаліста

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник проекту	12000	545,45	25	13636,25
Розробник	10000	454,55	74	33636,7
Всього:				47272,95

Розрахуємо додаткову заробітну плату. Додаткова заробітна плата $З_д$ розраховується як (10…12)% від суми основної заробітної плати, тобто:

$$З_Д = (0{,}1 \ldots 0{,}12) \times З_о$$

Додаткова заробітна плата:

$$З_Д = 0{,}1 \times (З_{О_{НК}} + З_{О_Р}) = 0.1 \times (13636{,}25 + 33636{,}7) = 4727{,}3 \ (грн)$$

Нарахування на заробітну плату $Н_{зп}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_д) \times \frac{\beta}{100}$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата розробників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування.

Єдиний соціальний внесок на загальнообов'язкове державне соціальне страхування (скор. *ЄСВ*) — консолідований страховий внесок в Україні, збір якого здійснюється в системі загальнообов'язкового державного страхування в обов'язковому порядку та на регулярній основі [2]. Відповідно до даних офіційного сайту міністерства фінансів в. 2019 році ЄСВ становить 22%.

$$Н_{зп} = (47272{,}95 + 4727{,}3\ ) \times \frac{22}{100} = 52000{,}25 \times 0{,}22$$
$$= 11440{,}06 \ (грн)$$

Розрахунок амортизаційних витрат для комп'ютера вартістю 10200 грн. виконується за такою формулою:

$$А = \frac{Ц \times Н_а}{100} \times \frac{Т}{12}$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = 25 %;

Т – термін використання (Т=4 міс.).

Тв – корисний час використання (Тв для комп'ютера становить 4 роки).

Отже, розрахуємо амортизаційні відрахування:

$$A = \frac{10200 \times 25}{100} \times \frac{4}{12} = \frac{255000}{100} \times 0{,}33 = 2550 \times 0{,}33 = 850\ (грн)$$

Витрати на матеріали розраховуються за формулою:

$$M = \sum_{1}^{n} Н_i \bullet Ц_i \bullet К_i - \sum_{1}^{n} В_i \bullet Ц_в$$

де $Н_i$ – витрати матеріалу *i*-го найменування;

$Ц_i$ – вартість матеріалу *i*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *i*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *i*-го найменування, грн/кг;

n – кількість видів матеріалів.

Вартість матеріалів, що були використані на розробку системи зведені до таблиці 4.4.

М=355,06×1,1=390,61 (грн)

Таблиця 4.4 – Вартість матеріалів, що були використані для розробки алгоритму тестування.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	Шт.	150	1	150
Пачка паперу	Уп	95	2	190
Ручка	Шт	7,55	2	15,1
Всього				355,1

Таким чином витрати на матеріали, що були використані в процесі розробки алгоритму тестування складуть М = 390,61 грн.

Розрахуємо витрати на комплектуючі.

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B \times П \times Ф \times K_П$$

де В – вартість 1кВт-години електроенергії (В=2,44 грн/кВт);

П – установлена потужність ноубука (П=0,03кВт);

Ф – фактична кількість годин роботи ноутбука (Ф=8 год.×74 дня);

Кп – коефіцієнт використання потужності (Кп< 1, Кп=0,8).

Потужність ноутбука складає 30Вт/год = 0,03кВт/год + потужність на освітлення, тоді П = 0,04кВт/год

$$B_E = 2,44 \times 0,04 \times (8 \times 74) \times 0,8 = 46,22\ (грн)$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $В_{ін}$ можна прийняти як (100-300)% від суми основної заробітної плати розробників, які виконували дану роботу, тобто:

$В_{ін} = (1..3)\times(З_о+З_р)$.

Отже, розрахуємо інші витрати:

Він = 1×(47272,95)= 47272,95 (грн).

Усі витрати складають:

$$В= 47272,95 + 4727,3 + 11440,06 + 850 + 390,61 + 46,22 + 47272,95 = 112000,09(грн)$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$B_{заг} = \frac{B}{\alpha}$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{112000,09}{1} = 112000,09(грн)$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо прогнозовані загальні витрати:

$$ЗВ = \frac{112000,09}{0.7} = 160000,13\ (грн)$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В економічному розділі магістерської кваліфікаційної роботи обґрунтовується економічна доцільність розробки методу та засобу розподілу секрету на основі зображень. На виконання усіх необхідних робіт необхідно 74 робочих дні.

Зростання чистого продукту для даного засобу можна оцінити у теперішній вартості грошей. Зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_{1}^{n} (\Delta П_я \times N + П_я \Delta N)_i$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати матеріалів на розробку алгоритму зменшаться на 200 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 200 грн), а кількість користувачів збільшиться: протягом першого року – на 90 користувачів, протягом другого року – на 100 користувачів, протягом третього року – на 80 користувачів.

Реалізація продукції до впровадження результатів наукової розробки складала 400 шт., а прибуток, що його отримувало підприємство (організація) на одиницю продукції до впровадження результатів наукової розробки – 350 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 200 \times 400 + (350 + 200) \times 90 = 80000 + 550 \times 90 = 129500\ (грн)$$

Протягом другого року:

$$\Delta П_2 = 200 \times 400 + (350 + 200) \times (90 + 100) = 80000 + 550 \times 190 = 184500\ (грн)$$

Протягом третього року:

$$\Delta П_3 = 200 \times 400 + (350 + 200) \times (90 + 100 + 80) = 80000 + 550 \times 270 = 228500\ (грн)$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1-й крок. Розрахуємо теперішню вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану раніше, тобто будемо вважати, що ЗВ = PV= 160000,13 *грн*.

2-й крок. Розрахуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами раніше та становить: $\Delta П_1 = 129500$ *грн*, $\Delta П_2 = 184500$ *грн*, $\Delta П_3 = 228500$ *грн*.

3-й крок. Для спрощення подальших розрахунків необхідно побудувати вісь часу, на яку наносять всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Якщо загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнюють 160000,13 грн., а результати вкладених у наукову розробку інвестицій почнуть виявлятися вже вкінці другого року впровадження. То ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 129500 грн. відносно базового року, у другому році – збільшення чистого прибутку на 1845 00 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 228500 грн (відносно базового року).

Тоді рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 4.1.

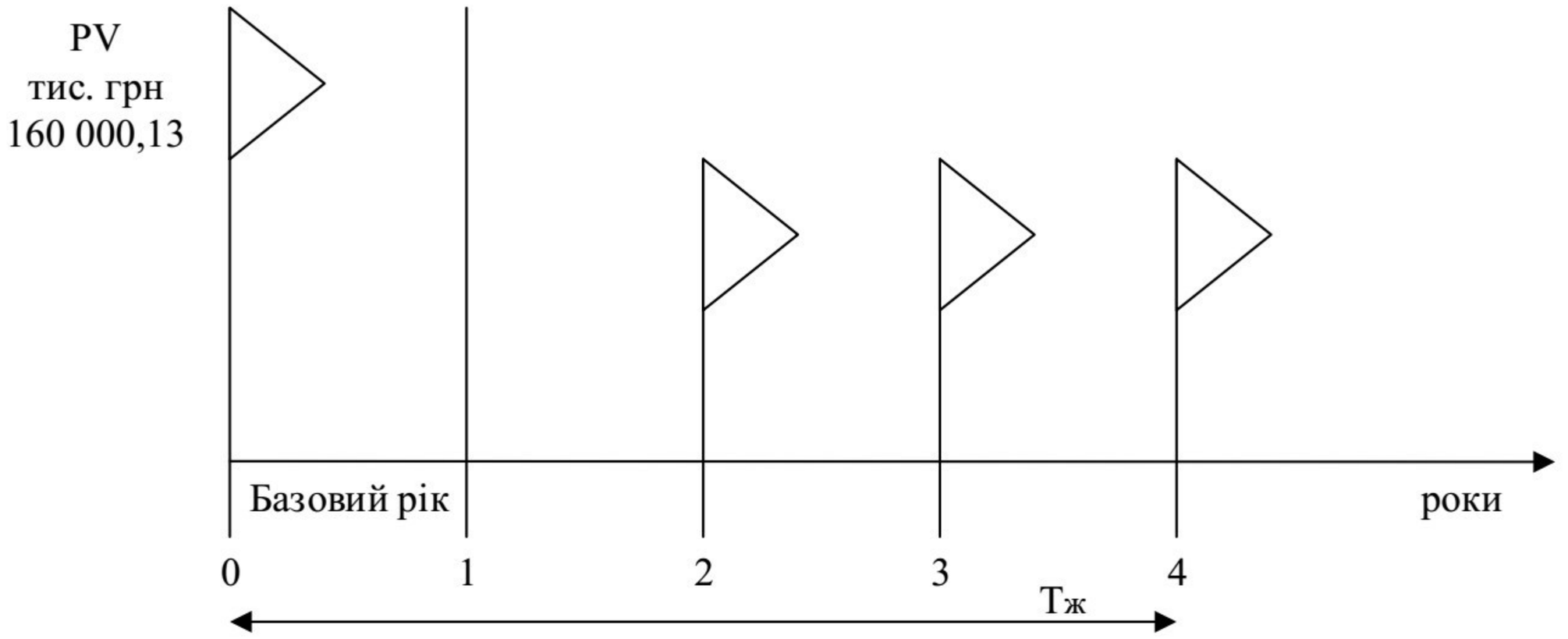


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4-й крок. Розрахуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$. Для цього скористаємося формулою:

$$Е_{абс} = (ПП – PV),$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.; PV – теперішня вартість інвестицій PV = ЗВ= 160000,13 грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП= \frac{129500}{(1+0,1)^2}+\frac{184500}{(1+0,1)^3}+\frac{228500}{(1+0,1)^4}=\frac{129500}{1,21}+\frac{184500}{1,331}+\frac{228500}{1,4641}=$$

$$107024,79+138617,58+156068,57=401710,94(грн)$$

$E_{абс}$ =401710,94 – 160000,13 = 241710,81 (грн)

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

5-й крок. Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[Т_ж]{1+\frac{E_{абс}}{PV}}-1,$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV – теперішня вартість інвестицій PV = ЗВ, грн; $Т_ж$ – життєвий цикл наукової розробки, роки.

Далі, розрахована величина $E_в$ порівнюється з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = (0,14...0,2); f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але може бути і значно більше.

Якщо величина $E_в > \tau_{мін}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде.

Спочатку спрогнозуємо величину $\tau_{мін}$. Припустимо, що за даних умов

$\tau_{мін} = 0{,}15 + 0{,}05 = 0{,}2$.

Тоді відносна (щорічна) ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_в = \sqrt[4]{1 + \frac{241710{,}81}{160000{,}13}} - 1 = \sqrt[4]{1 + 1{,}51} - 1 = \sqrt[4]{2{,}51} - 1 = 1{,}26 - 1 = 0{,}26$$ або 26%.

Оскільки $E_в = 26\% > \tau_{мін} = 0{,}2 = 20\%$, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

6-й крок. Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ок} = \frac{1}{E_в}.$$

Якщо $T_{ок}$ < 3…5-ти років, то фінансування даної наукової розробки в принципі є доцільним. В інших випадках потрібні додаткові розрахунки та обґрунтування.

$$T_{ок} = \frac{1}{0{,}26} = 3{,}8 \text{ року}$$

$T_{ок}$ < 5 років, що свідчить про доцільність фінансування даної наукової розробки.

Висновки

Рівень комерційного потенціалу засобу для розподілу секрету методом зображень є вище середнього. Загальні витрати на створення нового програмного продукту склали 160000,13 грн. Абсолютна ефективність капіталовкладень для даної розробки за 3 роки 241710,81 грн. Показники ефективності показують, що даний метод є доцільним і буде цікавий для інвестора. Термін окупності розробленого проекту менше 5-ти років, що підтверджує доцільність вкладання коштів в дану розробку.

ВИСНОВОК

За результат атами проведених досліджень та техніко-економічного обґрунтування доцільності досліджень вирішено продовжити роботу над удосконаленням методу та засобу розподілу секрету відповідно до теми магістерської роботи з виконанням поставлених задач кваліфікаційної роботи.

Проаналізовано відомі схеми розподілу секрету та досліджено методи їх побудови та аналізу

Проаналізовано відомі методи візуальної криптографії для розподілу секрету. Їх основними недоліками є неможливість використання порогової схеми повною мірою, надлишковість розмірів у вихідних зображеннях та можливі втрати якості при відновленні початкового зображення.

Розроблено власний метод розподілу секретного вмісту зображень, відмінний від інших тим, що відновлює початкове зображення без втрат, і може застосовуватись для будь-якого зображення BMP-формату.

Створений в результаті робити програмний засіб показав високу швидкість роботи, при незначному навантажені на процесор, цей програмний засіб має мінімальний набір інструментів для використання його без сторонніх засобів.

Проведено експериментальне дослідження на предмет розподілу секретного вмісту зображень. Після проведення розподілу програмним засобом отримано вихідні зображення з високою якістю приховування вмісту, а саме максимально нечитабельні зображення без будь-яких схожостей з оригіналом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. – М.: ДМК, 2000. – 448 с.
2. Червяков Н. И. Алгебраические и практические аспекты реализации нейросетево й порогово й схемы раделения секрета / Н. И. Червяков [та ін..] // Наука. Инновации. Технологии. – 2014. - № 2(6) – С. 14-26.
3. Введение в криптографию / Под общ. ред. В. В. Ященко. 4-е изд., доп. М.: МЦНМО, 2012. 348 с.
4. Агибалов Г. П. Избранные теоремы начального курса криптографии: Учебное пособие. – Томск: Изд-во НТЛ, 2005. – 116 с.
5. Визуальная криптографія. // CryptoWiki [Електронний ресурс] – Режим доступу: URL http://cryptowiki.net/index.php?title=Визуальная_криптография – Назва з титул. екрана.
6. Романова Е.А., Мелешко Е.А. Методы защиты информации с использованием визуальной криптографии. [Електроний ресурс] – URL: http://www.rusnauka.com/31_ONBG_2011/Informatica/4_96500.doc.htm – Назва з титул. екрана.
7. Naor, Moni, and Adi Shamir. «Visual cryptography» Workshop on the Theory and Application of of Cryptographic Techniques. Springer Berlin Heidelberg, 1994.
8. От визуальной криптографии к визуальной стеганографии и разделению секрета [Електроний ресурс]. – URL: https://clck.ru/EMYpJ/ – Назва з титул. екрана.
9. Verheul, E.R. and H.C.A.van Tilborg. Constructions and properties of k out of n visual secret sharing schemes. — Design Codes and Cryptography, 1997. — С. 11(2):179–196.
10. Схема передачи зашифрованных зображений [Електроний ресурс] – URL: https://ru.wikipedia.org/wiki/Схема_передачи_зашифрованных_изображений_(n,_n+1) – Назва з титул. екрана.

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

12. Иванов М. А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М. А. Иванов, И. В. Чугунков. – Москва : КУДИЦ-ОБРАЗ, 2003. – 240 с.

13. Lloris Ruiz A. Algebraic circuits / A. Lloris Ruiz, E. Castillo Morales, L. Parrilla Roure, A. García Ríos. - Berlin, Heidelberg : Springer Berlin Heidelberg, 2014.

14. Алферов А. П. Основи криптографии учебное пособие / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмін, А. В. Черемушкин. - Москва : Гелиос АРВ, 2002. - 480 с.

15. Кузнецов Г. В. Математичні основи криптографії / Г. В. Кузнецов, В. В. Фомичов, С. О. Сушко, Л. Я. Фомичова. - Дніпропетровський Національний гірничий університет, 2004. - 391 с.

16. BMP (Bitmap Picture) - Национальная библиотека им. Н. Э. Баумана Bauman National Library [Електроний ресурс] – URL: https://ru.bmstu.wiki/BMP_(Bitmap_Picture) – Назва з титул. екрана.

17. BMP file format [Електроний ресурс] – URL: https://en.wikipedia.org/wiki/BMP_file_format – Назва з титул. екрана.

18. Microsoft Windows Bitmap File Format Summary [Електроний ресурс] – URL: https://www.fileformat.info/format/bmp/egff.htm – Назва з титул. екрана.

19. Описание формата BMP [Електроний ресурс] – URL: https://jenyay.net/Programming/Bmp – Назва з титул. екрана.

20. Методичні вказівки до виконання студентами-магістрантами наукового напрямку економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський - Вінниця: ВНТУ, 2012. - 22 с.

ДОДАТКИ

Додаток А
Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д.т.н., проф.
_____ В. А. Лужецький
_____ _____ 2019 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання науково-дослідної роботи

на тему: «МЕТОД ТА ЗАСІБ РОЗПОДІЛЕННЯ СЕКРЕТУ»
08-20.МКР.005.00.000 ТЗ

Керівник магістерської кваліфікаційної роботи

д.т.н., проф. зав. каф. ЗІ В. А. Лужецький

Вінниця 2019

1 Підстави для проведення робіт

Робота проводиться на підставі наказу ВНТУ від 2 жовтня 2019 року № 254.

Дата початку роботи 1.09.19 р.

Дата закінчення роботи 14.12.19 р.

2 Мета та призначення НДР

Метою магістерської кваліфікаційної роботи є підвищення захищеності секретного вмісту зображень за рахунок розробки методу розподілу секрету.

Об'єктом дослідження є процес захисту інформації за допомогою розподілення секрету.

Актуальність теми. Інформація має важливе значення в життєдіяльності людства. При цьому вона стає все більш вразливою через зростаючі обсяги збережених даних. Тому все більшу важливість набуває проблема захисту інформації від несанкціонованого доступу під час передачі та зберіганні. Для захисту секретної інформації від втрати і компрометації необхідно підвищити надійність зберігання секретної інформації. Є кілька варіантів підвищення надійності зберігання секретної інформації, наприклад: створення декількох копій секретної інформації та зберігання їх в різних місцях. Даний метод підвищує надійність при втраті, але підвищує ймовірність компрометації ключа; поділ секретних даних між учасниками певної групи. Даний метод зменшує ймовірність компрометації ключа.

3 Вихідні дані для проведення НДР

НДР проводиться вперше і вихідними даними для проведення НДР є:

1. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. – М.: ДМК, 2000. – 448 с.

2. Червяков Н. И. Алгебраические и практические аспекты реализации нейросетево й порогово й схемы раделения секрета / Н. И. Червяков [та ін..] // Наука. Инновации. Технологии. – 2014. - № 2(6) – С. 14-26.

3. Введение в криптографию / Под общ. ред. В. В. Ященко. 4-е изд., доп. М.: МЦНМО, 2012. 348 с.

4. Агибалов Г. П. Избранные теоремы начального курса криптографии: Учебное пособие. – Томск: Изд-во НТЛ, 2005. – 116 с.

5. Визуальная криптографія. // CryptoWiki [Електронний ресурс] – Режим доступу: URL http://cryptowiki.net/index.php?title=Визуальная_криптография – Назва з титул. екрана.

6. Романова Е.А., Мелешко Е.А. Методы защиты информации с использованием визуальной криптографии. [Електроний ресурс] – URL: http://www.rusnauka.com/31_ONBG_2011/Informatica/4_96500.doc.htm – Назва з титул. екрана.

4 Виконавці НДР

Студент групи БС-18м Дехтяренко Микола Сергійович

5 Вимоги до виконання НДР

Для захищеності секретного вмісту зображень за рахунок розподілу секрету необхідно:

- проаналізувати відомі методи розподілу секрету;
- проаналізувати методи візуальної криптографії;
- розробити власний метод розподілу секрету;
- розробити програмний засіб для виконання даного методу.

6 Вимоги до супровідної документації

6.1 Графічна і текстова документація повинна відповідати діючим стандартам України.

7 Етапи НДР

Робота з теми виконується у 8 етапів.

Зміст етапу	Початок	Закінчення	Очікувані результати	Звітна
Аналіз завдання. Вступ	01.09.2019	04.09.2019	Вступ	Чернетка вступу
Розробка технічного завдання	16.09.2019	22.09.2019	Технічне завдання	Проект технічного завдання
Розробка техніко-економічного та науково-технічного обґрунтування доцільності досліджень	05.09.2019	15.09.2019	Аналіз існуючих аналогів. Вибір напрямку дослідження	Чернетка першого розділу
Аналіз літературних джерел за напрямком магістерської кваліфікаційної роботи	23.09.2019	29.09.2019	Аналіз відомих методів. Постановка завдання	Чернетка другого розділу
Розробка методу розподілу секрету	30.09.2019	12.10.2019	Запропоновано метод розподілу секрету.	Чернетка третього розділу
Експериментальні дослідження	13.10.2019	09.11.2019	Програмний засіб, який реалізує розроблювані методи	Чернетка четвертого розділу
Розробка економічного розділу	11.11.2019	17.11.2019	Економічні показники дослідження	Чернетка з економічного розділу
Оформлення пояснювальної записки	25.11.2019	30.11.2019	Пояснювальна записка	Пояснювальна записка

8 Очікувані результати та порядок реалізації НДР

Передбачається розробка методу розподілення секрету, що міститься в зображеннях. Заплановане створення програмного засобу, який може бути використаний у навчальному процесі.

9 Матеріали які подаються після закінчення НДР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

10 Порядок приймання НДР та її етапів

Апробація на науково-технічних конференціях та семінарах. Результати роботи будуть розглядатися на засіданні ДЕК із захисту магістерських кваліфікаційних робіт.

Попередній захист та доопрацювання МКР листопад 2019 р.

Представлення МКР до захисту грудень 2019.

Захист МКР 17.12.2019

11 Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп'ютерного набору у відповідності вимог ДСТУ 3008-95. «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення»

12 Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв'язку з тим що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

Розробив студент групи БС-18м _____ Дехтяренко М. С.

Додаток Б
Лістинг програми

Counter.ts

```
import { CounterDirection } from './CounterDirection';
export class Counter {
  private readonly direction: CounterDirection;
  private readonly start: number;
  private readonly finish: number;
  private current: number;

  constructor(start: number, finish: number) {
    this.direction = start <= finish ? CounterDirection.UP : CounterDirection.DOWN;
    this.start = start;
    this.finish = finish;
    this.current = start - this.direction;
  }

  public getStart(): number {
    return this.start;
  }

  public getFinish(): number {
    return this.finish;
  }

  public getCurrent(): number {
    if (this.current + this.direction === this.start) {
      return this.start;
    } else {
      return this.current;
    }
  }

  public step(): number {
    if (this.current !== this.finish) {
      this.current += this.direction;
    }

    return this.current;
  }

  public hasNextStep(): boolean {
    return this.current !== this.finish;
  }
}
```

CounterDirection.ts

```
export enum CounterDirection {UP = +1, DOWN = -1}
export type CounterDirectionKeys = keyof typeof CounterDirection;
```

CountersNumber.ts

```
export enum CountersNumber { _2 = 2, _4 = 4, _8 = 8, _16 = 16, _32 = 32, _64 = 64, _128 =
128, _256 = 256, _512 = 512, _1024 = 1024, _2048 = 2048, _8192 = 8192, _16384 = 16384 }
export type CountersNumberKeys = keyof typeof CountersNumber;
```

KeyConfigGenerator.ts

```
import { CountersNumber, CountersNumberKeys } from '../counters/CountersNumber';
import { POLYNOMIALS } from '../lfsr/GaloisLeftLFSR';
import { Random } from '../../utils/Random';

export interface LFSRConfig {
  degree: number;
  state: number;
  polynomial: number;
}
export interface CountersConfig {
  directions: Array<number>;
  lfsr: Array<LFSRConfig>;
  numberCounters: CountersNumber;
}
export class ConfigGenerator {
  public static generateLFSRConfig(degree: number): LFSRConfig {
    return {
      degree: degree,
      state: Random.generateSequence(1, 1, 2 ** degree - 1)[0],
      polynomial: ConfigGenerator.randomPolynomial(degree)
    };
  }

  public static generateCountersConfig(numberCountersKeys: CountersNumberKeys):
CountersConfig {
    const numberCounters: CountersNumber = Reflect.get(CountersNumber,
numberCountersKeys);
    const directions: Array<number> = Random.generateSequence(numberCounters, 0,
Random.MAX_INTEGER).map(
      (v: number) => v % 2
    );

    return {
      directions: directions,
      lfsr: [
        ConfigGenerator.generateLFSRConfig(32),
        ConfigGenerator.generateLFSRConfig(23),
        ConfigGenerator.generateLFSRConfig(21)
      ],
      numberCounters: numberCounters
    };
  }

  private static randomPolynomial(degree: number): number {
    const generator: Random = new Random(1, 2 ** degree - 1);
    const degreeBitsPolynomials = POLYNOMIALS[+degree];
    const randomIndex: number = generator.nextInt() % degreeBitsPolynomials.length;

    return degreeBitsPolynomials[+randomIndex];
  }
}
```

ShareSecretKey.ts

```
import { outputFileSync, readFileSync } from 'fs-extra';
import { resolve } from 'path';
```

```
import { IllegalArgumentException } from '../../exceptions/IllegalArgumentException';
import { CountersNumberKeys } from '../counters/CountersNumber';
import { ConfigGenerator, CountersConfig } from './KeyConfigGenerator';
import { IOException } from '../../exceptions/IOException';

export class ShareSecretKey {
  public N: number;
  public K: number;
  public countersConfig: CountersConfig;

  private constructor() {}

  public static generateKey(N: number, K: number, numberCountersKeys:
CountersNumberKeys): ShareSecretKey {
    if (N <= 0 || K <= 0) {
      throw new IllegalArgumentException('N and K must be more than zero.');
    }

    if (K > N) {
      throw new IllegalArgumentException('Must be N >= K > 0.');
    }

    const ssk: ShareSecretKey = new ShareSecretKey();
    ssk.N = N;
    ssk.K = K;
    ssk.countersConfig = ConfigGenerator.generateCountersConfig(numberCountersKeys);

    return ssk;
  }

public static write(key: ShareSecretKey, filePath: string): void {
  try {
    const jsonKey: string = JSON.stringify(key);
    outputFileSync(resolve(filePath), jsonKey, { encoding: 'utf8' });
  } catch (error) {
    throw new IOException(`Fail write key to file: ${resolve(filePath)}`);
  }
}

public static read(filePath: string): ShareSecretKey {
  const jsonKey: string = readFileSync(resolve(filePath), { encoding: 'utf8' });
  const keyObject: ShareSecretKey = JSON.parse(jsonKey);

  return Object.assign(new ShareSecretKey(), keyObject);
}

toString(): string {
  return JSON.stringify(this);
  }
}
```

GaloisLeftLFSR.ts

```
import { GaloisLFSR } from './GaloisLFSR';

export class GaloisLeftLFSR extends GaloisLFSR {
  private readonly clearMask: number;
  private readonly mask: number;

  constructor(degree: number, state: number, polynomial: number) {
    super(degree, state, polynomial);
```

```
  this.clearMask = GaloisLFSR.generateClearMask(degree);

  this.state = state & this.clearMask;
  this.polynomial = polynomial & this.clearMask;

  this.mask = 1 << (degree - 1);
}

nextBit(): number {
  this.iteration++;

  if ((this.state & this.mask) === this.mask) {
    this.state = ((this.state << 1) ^ this.polynomial) & this.clearMask;
    return 1;
   } else {
    this.state = (this.state << 1) & this.clearMask;
    return 0;
   }
  }
}

export const POLYNOMIALS: { [index: number]: Array<number> } = {
  21: require('./polynomials/21.json'),
  23: require('./polynomials/23.json'),
  32: require('./polynomials/32.json')
};
```

GaloisLFSR.ts

```
import { LFSR } from './LFSR';

export abstract class GaloisLFSR implements LFSR {
  protected degree: number;
  protected state: number;
  protected polynomial: number;
  protected iteration: number;

  protected static generateClearMask(degree: number): number {
    let clearMask = 0;

    for (let index = 0; index < degree; ++index) {
      clearMask = (clearMask << 1) | 1;
    }

    return clearMask;
  }

  protected constructor(degree: number, state: number, polynomial: number) {
    this.degree = degree;
    this.state = state;
    this.polynomial = polynomial;
    this.iteration = 0;
  }

  maxPeriod(): number {
    return Math.pow(2, this.degree) - 1;
  }

  abstract nextBit(): number;

  nextBits(amount: number): number {
```

```
  let bits = 0;

  for (let index = 0; index < amount; ++index) {
    bits = (bits << 1) | this.nextBit();
  }

  return bits;
}

nextState(): number {
  this.nextBit();
  return this.state;
  }

  currentState(): number {
    return this.state;
  }

  getIteration(): number {
    return this.iteration;
  }
}
```

LFSR.ts

```
export interface LFSR {
  maxPeriod(): number;
  nextBit(): number;
  nextBits(amount: number): number;
  nextState(): number;
  currentState(): number;
  getIteration(): number;
}
```

BinaryData.ts

```
import { Data } from '../Data';

export class BinaryData extends Data {
  private data: Buffer;

  constructor(data: Buffer) {
    super();
    this.data = data;
  }
  public setData(data: Buffer): void {
    this.data = data;
  }
  public getData(): Buffer {
    return this.data;
  }
}
```

BinaryDataReader.ts

```
import { readFileSync } from 'fs-extra';
import { resolve } from 'path';
import { DataReader } from '../DataReader';
import { BinaryData } from './BinaryData';
export class BinaryDataReader implements DataReader<BinaryData> {
```

```
  public read(fileName: string): BinaryData {
    const fileData: Buffer = readFileSync(resolve(fileName));
    return new BinaryData(fileData);
  }
}
```

BinaryDataTransformer.ts

```
import { DataTransformer } from '../DataTransformer';
import { BinaryData } from './BinaryData';
import { SharedPart } from '../../ShareSecret';
export class BinaryDataTransformer implements DataTransformer<BinaryData> {
  preEncode(srcData: BinaryData): BinaryData {
    return srcData;
  }
  postEncode(srcData: BinaryData, parts: Array<Buffer>): Array<BinaryData> {
    const transformerData: Array<BinaryData> = new Array<BinaryData>(parts.length);
    parts.forEach((value: Buffer, index: number): void => {
      transformerData[+index] = new BinaryData(value);
    });
    return transformerData;
  }
  preDecode(parts: Array<SharedPart>): Array<SharedPart> {
    return parts;
  }
  postDecode(parts: Array<SharedPart>, decodeData: Buffer): BinaryData {
    return new BinaryData(decodeData);
  }
}
```

BinaryDataWriter.ts

```
import { outputFileSync } from 'fs-extra';
import { resolve } from 'path';
import { DataWriter } from '../DataWriter';
import { BinaryData } from './BinaryData';
export class BinaryDataWriter implements DataWriter<BinaryData> {
  public write(data: BinaryData, fileName: string): void {
    outputFileSync(resolve(fileName), data.getData());
  }
}
```

bmp-js.d.ts

```
declare module 'bmp-js' {
  interface BmpDecoderData {
    pos?: number;
    buffer?: Buffer;
    is_with_alpha?: boolean;
    bottom_up?: boolean;
    flag?: string;
    fileSize?: number;
    reserved?: number;
    offset?: number;
    headerSize?: number;
    width: number;
    height: number;
    planes?: number;
    bitPP?: number;
    compress?: number;
    rawSize?: number;
```

```
  hr?: number;
  vr?: number;
  colors?: number;
  importantColors?: number;
  data: Buffer;
}

interface BmpEncodeData {
  data: Buffer;
  width: number;
  height: number;
}
  interface BmpEncoderData {
    data: Buffer;
    width: number;
    height: number;
  }

  function decode(bmpBuffer: Buffer): BmpDecoderData;
  function encode(bmpData: BmpEncodeData): BmpEncoderData;
}
```

BMPData.ts

```
import { BmpDecoderData } from 'bmp-js';
import { Data } from '../Data';
export class BMPData extends Data {
  private bmpDecoderData: BmpDecoderData;
  constructor(bmpDecoderData: BmpDecoderData) {
    super();
    this.bmpDecoderData = bmpDecoderData;
  }
  getData(): Buffer {
    return this.bmpDecoderData.data;
  }
  setData(data: Buffer): void {
    this.bmpDecoderData.data = data;
  }
  get width(): number {
    return this.bmpDecoderData.width;
  }
  set width(value: number) {
    this.bmpDecoderData.width = value;
  }
  get height(): number {
    return this.bmpDecoderData.height;
  }
  set height(value: number) {
    this.bmpDecoderData.height = value;
  }
}
```

BMPDataReader.ts

```
import { decode } from 'bmp-js';
import { readFileSync } from 'fs-extra';
import { resolve } from 'path';

import { DataReader } from '../DataReader';
import { BMPData } from './BMPData';
```

```
export class BMPDataReader implements DataReader<BMPData> {
  read(fileName: string): BMPData {
    const fileBuffer: Buffer = readFileSync(resolve(fileName));
    return new BMPData(decode(fileBuffer));
  }
}
```

BMPDataTransformer.ts

```
import { BufferTools } from '../../../utils/BufferTools';
import { SharedPart } from '../../ShareSecret';
import { DataTransformer } from '../DataTransformer';
import { BMPData } from './BMPData';

export class BMPDataTransformer implements DataTransformer<BMPData> {
  private static increaseBMPData(srcData: Buffer, height: number): Buffer {
    const size: number = srcData.length;
    const width = Math.ceil((size + 4) / (height * 3));
    const unnecessaryDataSize: number = height * width * 3 - (size + 4);
    const unnecessaryData: Buffer = BufferTools.copyBuffer(srcData, 0, unnecessaryDataSize);
    const sizeToBuffer: Buffer = BufferTools.numberToBuffer(size);
    return Buffer.concat([srcData, unnecessaryData, sizeToBuffer]);
  }

  private static decreaseBMPData(srcData: Buffer): Buffer {
    const sizeInBuffer: Buffer = BufferTools.copyBuffer(srcData, srcData.length - 4, 4);
    const size: number = BufferTools.bufferToNumber(sizeInBuffer);
    return BufferTools.copyBuffer(srcData, 0, size);
  }

private static rasterToRGB(rasterARGB: Buffer): Buffer {
  const rasterRGB: Array<number> = [];
  for (let index = 0; index < rasterARGB.length; index += 4) {
    rasterRGB.push(rasterARGB[index + 1]); // r
    rasterRGB.push(rasterARGB[index + 2]); // g
    rasterRGB.push(rasterARGB[index + 3]); // b
  }

  return Buffer.from(rasterRGB);
}

private static rasterToARGB(rasterRGB: Buffer): Buffer {
  const rasterARGB: Array<number> = [];
  for (let index = 0; index < rasterRGB.length; index += 3) {
    rasterARGB.push(0); // a
    rasterARGB.push(rasterRGB[+index]); // r
    rasterARGB.push(rasterRGB[index + 1]); // g
    rasterARGB.push(rasterRGB[index + 2]); // b
  }
  return Buffer.from(rasterARGB);
}
preEncode(srcData: BMPData): BMPData {
  srcData.setData(BMPDataTransformer.rasterToRGB(srcData.getData()));
  return srcData;
}
postEncode(srcData: BMPData, parts: Array<Buffer>): Array<BMPData> {
  const transformerData: Array<BMPData> = new Array<BMPData>(parts.length);
  parts.forEach((value: Buffer, index: number) => {
    const height = srcData.height;
    const imgData: Buffer = BMPDataTransformer.increaseBMPData(value, height);
    const width = imgData.length / (height * 3);
```

```
    transformerData[+index] = new BMPData({
      data: BMPDataTransformer.rasterToARGB(imgData),
      width: width,
      height: height
    });
  });

  return transformerData;
}

preDecode(parts: Array<SharedPart>): Array<SharedPart> {
  parts.forEach((sharedPart: SharedPart) => {
    const buffer: Buffer = sharedPart.partData.getData();

sharedPart.partData.setData(BMPDataTransformer.decreaseBMPData(BMPDataTransformer.ras
terToRGB(buffer)));
    });

    return parts;
  }
  postDecode(parts: Array<SharedPart>, decodeData: Buffer): BMPData {
    const partBMPData: BMPData = parts[0].partData as BMPData;
    const data: Buffer = BMPDataTransformer.rasterToARGB(decodeData);
    const height = partBMPData.height;
    const width = data.length / (height * 4);
    return new BMPData({
      data: data,
      height: height,
      width: width
    });
  }
}
```

BMPDataWriter.ts

```
import { encode } from 'bmp-js';
import { outputFileSync } from 'fs-extra';
import { resolve } from 'path';
import { DataWriter } from '../DataWriter';
import { BMPData } from './BMPData';

export class BMPDataWriter implements DataWriter<BMPData> {
  write(data: BMPData, fileName: string): void {
    const rawImageData = encode({
      data: data.getData(),
      width: data.width,
      height: data.height
    });
    outputFileSync(resolve(fileName), rawImageData.data);
  }
}
```

Data.ts

```
import { BufferTools } from '../../utils/BufferTools';

export abstract class Data {
  abstract getData(): Buffer;
  abstract setData(data: Buffer): void;
  public equals(other: Data): boolean {
    if (this === other) {
```

```
      return true;
    }
    if (other === null) {
      return false;
    }
    return BufferTools.equals(this.getData(), other.getData());
  }
}
```

DataReader.ts

```
import { Data } from './Data';

export interface DataReader<T extends Data> {
  read(fileName: string): T;
}
```

DataTransformer.ts

```
import { SharedPart } from '../ShareSecret';
import { Data } from './Data';

export interface DataTransformer<T extends Data> {
  preEncode(srcData: T): T;
  postEncode(srcData: T, parts: Array<Buffer>): Array<T>;
  preDecode(parts: Array<SharedPart>): Array<SharedPart>;
  postDecode(parts: Array<SharedPart>, decodeData: Buffer): T;
}
```

DataWriter.ts

```
import { Data } from './Data';

export interface DataWriter<T extends Data> {
  write(data: T, fileName: string): void;
}
```

PermuteData.ts

```
import { BufferTools } from '../utils/BufferTools';
import { ShareSecretKey } from './key/ShareSecretKey';
import { PermuteGenerator } from './PermuteGenerator';

export class PermuteData {
  private static increaseData(srcData: Buffer, N: number): Buffer {
    const size: number = srcData.length;
    const unnecessaryDataSize: number = N - ((size + 4) % N);
    const unnecessaryData: Buffer = BufferTools.copyBuffer(srcData, 0, unnecessaryDataSize);
    const sizeToBuffer: Buffer = BufferTools.numberToBuffer(size);
    return Buffer.concat([srcData, unnecessaryData, sizeToBuffer]);
  }

  private static decreaseData(srcData: Buffer): Buffer {
    const sizeInBuffer: Buffer = BufferTools.copyBuffer(srcData, srcData.length - 4, 4);
    const size: number = BufferTools.bufferToNumber(sizeInBuffer);
    return BufferTools.copyBuffer(srcData, 0, size);
  }

  public static encode(srcData: Buffer, key: ShareSecretKey): Buffer {
    const increaseData: Buffer = PermuteData.increaseData(srcData, key.N);
```

```
    const permuteGenerator: PermuteGenerator = new PermuteGenerator(key.countersConfig,
increaseData.length);

    const encryptedData = Buffer.alloc(increaseData.length);
    for (let index = 0; index < increaseData.length; ++index) {
      const gn: number = permuteGenerator.generateNumber();
      encryptedData[+index] = (increaseData[+gn] + BufferTools.compressNumber(gn)) & 0xff;
    }

    return encryptedData;
  }

  public static decode(srcData: Buffer, key: ShareSecretKey): Buffer {
    const permuteGenerator: PermuteGenerator = new PermuteGenerator(key.countersConfig,
srcData.length);

    const decryptedData: Buffer = Buffer.alloc(srcData.length);
    for (let index = 0; index < srcData.length; ++index) {
      const gn: number = permuteGenerator.generateNumber();
      decryptedData[+gn] = (srcData[+index] - BufferTools.compressNumber(gn)) & 0xff;
    }
    return PermuteData.decreaseData(decryptedData);
  }
}
```

PermuteGenerator.ts

```
import { Counter } from './counters/Counter';
import { CounterDirection } from './counters/CounterDirection';
import { CountersConfig, LFSRConfig } from './key/KeyConfigGenerator';
import { GaloisLeftLFSR } from './lfsr/GaloisLeftLFSR';
export class PermuteGenerator {
  private readonly counters: Array<Counter>;
  private registers: Array<GaloisLeftLFSR>;
  private readonly bitsAmount: number;

  constructor(countersConfig: CountersConfig, srcDataSize: number) {
    this.counters = PermuteGenerator.initCounters(countersConfig.directions, srcDataSize);
    this.registers = PermuteGenerator.initRegisters(countersConfig.lfsr);
    this.bitsAmount = PermuteGenerator.getBitsAmounts(this.counters.length);
  }

  private static initCountersDirection(directions: Array<number>): Array<CounterDirection> {
    return directions.map((value: number) => (value <= 0 ? CounterDirection.DOWN :
CounterDirection.UP));
  }

  private static initCounters(directions: Array<number>, srcDataSize: number):
Array<Counter> {
    const countersDirection: Array<CounterDirection> =
PermuteGenerator.initCountersDirection(directions);

    const amountCounters: number = directions.length;
    const countersArray: Array<Counter> = new Array<Counter>(amountCounters);
    const partSize: number = (srcDataSize / amountCounters) | 0; // parseInt((srcDataSize /
amountCounters).toString());
    const remainder: number = srcDataSize % amountCounters;
    let counter = 0;
    for (let index = 0; index < amountCounters; ++index) {
      const direction = countersDirection[+index];
      if (direction === CounterDirection.UP) {
```

```
      countersArray[+index] = new Counter(counter, counter + partSize - 1);
     } else {
      countersArray[+index] = new Counter(counter + partSize - 1, counter);
     }
     counter += partSize;
    }
    if (remainder !== 0) {
     const lastCounters = countersArray[amountCounters - 1];
     if (lastCounters.getCurrent() === srcDataSize - 1 - remainder) {
      countersArray[amountCounters - 1] = new Counter(srcDataSize - 1,
lastCounters.getFinish());
     } else {
      countersArray[amountCounters - 1] = new Counter(lastCounters.getStart(), srcDataSize -
1);
     }
    }
    return countersArray;
  }

  private static initRegisters(lfsrConfigs: Array<LFSRConfig>): Array<GaloisLeftLFSR> {
    const lfsrs: Array<GaloisLeftLFSR> = new Array<GaloisLeftLFSR>(lfsrConfigs.length);
    lfsrConfigs.forEach((config: LFSRConfig, index: number) => {
     const { degree, state, polynomial } = config;
     lfsrs[+index] = new GaloisLeftLFSR(degree, state, polynomial);
    });
    return lfsrs;
  }

  private static getBitsAmounts(value: number): number {
    return Math.ceil(Math.log(value) / Math.log(2));
  }

  private generateBits(): number {
    let bits = 0;
    this.registers.forEach((register: GaloisLeftLFSR) => {
     bits = bits ^ register.nextBits(this.bitsAmount);
    });
    return bits;
  }

  public generateNumber(): number {
    let newNumber = -1;
    while (newNumber === -1) {
     const bits = this.generateBits() % this.counters.length;
     const currentCounter = this.counters[+bits];
     if (currentCounter.hasNextStep()) {
      newNumber = currentCounter.step();
     }
     return newNumber;
    }
  }
```

ShareSecret.ts

```
import { BufferTools } from '../utils/BufferTools';
import { Matrix } from '../utils/Matrix';
import { ShareSecretKey } from './key/ShareSecretKey';
import { PermuteData } from './PermuteData';
import { Data } from './plaindata/Data';
```

```
export interface SharedPart {
  index: number;
  partData: Data;
}

interface PermutedDataSlice {
  index: number;
  parts: Array<Buffer>;
}

export class ShareSecret {
  private static generateDivideMatrix(N: number, K: number): Array<Array<number>> {
  const rowsLength: number = N;
  const columnsLength: number = N - K + 1;

  const matrix = Matrix.generateMatrix(rowsLength, columnsLength);

  for (let rowIndex = 0; rowIndex < matrix.length; ++rowIndex) {
    const row: Array<number> = matrix[+rowIndex];
    let counter: number = rowIndex + 1;
    for (let columnIndex = 0; columnIndex < row.length; ++columnIndex) {
      row[+columnIndex] = counter;
      counter = (counter + 1) % (N + 1);
      counter = counter === 0 ? 1 : counter;
    }
  }

  return matrix;
}

  public static encode(srcData: Buffer, key: ShareSecretKey): Array<Buffer> {
    const permuteData: Buffer = PermuteData.encode(srcData, key);
    const divideMatrix: Array<Array<number>> = ShareSecret.generateDivideMatrix(key.N,
key.K);

    const parts = BufferTools.split(permuteData, key.N);
    const encodeData: Array<Buffer> = new Array<Buffer>(key.N);

    for (let index = 0; index < encodeData.length; ++index) {
      const p = divideMatrix[+index];
      encodeData[+index] = Buffer.concat(p.map((v: number) => parts[v - 1]));
    }

    return encodeData;
  }

  private static split(sharedParts: Array<SharedPart>, count: number):
Array<PermutedDataSlice> {
    return sharedParts.map(
      (part: SharedPart): PermutedDataSlice => {
        return {
          index: part.index,
          parts: BufferTools.split(part.partData.getData(), count)
        };
      }
    );
  }

  private static calculateRealParts(data: Array<PermutedDataSlice>, divideMatrix: number[][]):
Array<Array<Buffer>> {
    const realParts: Array<Array<Buffer>> = [];
```

```
data.forEach(({ index, parts }: PermutedDataSlice) => {
  const vector = divideMatrix[+index];

  vector.forEach((num: number, index: number) => {
    const buff: Buffer = parts[+index];

    if (!Array.isArray(realParts[num - 1])) {
      realParts[num - 1] = [];
    }

    realParts[num - 1].push(buff);
    });
   });

   return realParts;
  }

  public static decode(sharedParts: Array<SharedPart>, key: ShareSecretKey): Buffer {
    const   divideMatrix:   Array<Array<number>>   =   ShareSecret.generateDivideMatrix(key.N,
key.K);
    const data = ShareSecret.split(sharedParts, key.N - key.K + 1);

    const temp1 = ShareSecret.calculateRealParts(data, divideMatrix);
    const temp2 = temp1.map((arr: Buffer[]) => arr[0]);
    return PermuteData.decode(Buffer.concat(temp2), key);
  }
}
```

IllegalArgumentException.ts

```
export class IllegalArgumentException extends Error {}
```

IOException.ts

```
export class IOException extends Error {}
```

BufferTools.ts

```
import { IllegalArgumentException } from '../exceptions/IllegalArgumentException';

export class BufferTools {
  public static numberToBuffer(value: number): Buffer {
    const size = 4;
    const buffer: Buffer = Buffer.alloc(size);

    let mask = 0x00;
    for (let index = 0; index < size; ++index) {
      buffer[+index] = (value >> mask) & 0xff;
      mask += 0x08;
    }

    return buffer;
  }

  public static bufferToNumber(buffer: Buffer): number {
    let number = 0x00;
    let mask = 0x00;

    buffer.forEach((value: number) => {
      number |= value << mask;
```

```
    mask += 0x08;
  });

  return number;
}

public static compressNumber(value: number): number {
  const toBuffer = BufferTools.numberToBuffer(value);
  let compressValue = 0;

  const size = toBuffer.length;
  for (let index = 0; index < size; ++index) {
    compressValue = (compressValue + toBuffer[+index]) & 0xff;
  }
  return compressValue;
}

public static copyBuffer(srcBuffer: Buffer, offset: number, size: number): Buffer {
  const dstBuffer: Buffer = Buffer.alloc(size);

  for (let index = 0; index < size; ++index) {
    dstBuffer[+index] = srcBuffer[index + offset];
  }

  return dstBuffer;
}

public static split(srcBuffer: Buffer, amount: number): Array<Buffer> {
  const srcLength = srcBuffer.length;

  if (srcLength % amount !== 0) {
    throw new IllegalArgumentException('Must be srcBuffer.length % amount == 0');
  }

  const partSize = srcLength / amount;
  const parts: Array<Buffer> = new Array<Buffer>(amount);

  for (let index = 0; index < amount; ++index) {
    parts[+index] = BufferTools.copyBuffer(srcBuffer, index * partSize, partSize);
  }

  return parts;
}

private static _equals(x: Buffer, y: Buffer): boolean {
  // compare with yourself (compare pointers to object)
  if (x === y) {
    return true;
  }

const lengthX: number = x.length;
const lengthY: number = y.length;

if (lengthX !== lengthY) {
  return false;
}

for (let index = 0; index < lengthX; ++index) {
  const elementX = x[+index];
  const elementY = y[+index];
```

```
    if (elementX !== elementY) {
      return false;
    }
  }

  return true;
}

public static equals(arg0: Buffer, ...args: Array<Buffer>): boolean {
  for (let index = 0, length: number = args.length; index < length; ++index) {
    if (!BufferTools._equals(arg0, args[+index])) {
      return false;
    }
  }

  return true;
 }
}
```

Matrix.ts

```
export class Matrix {
  public static generateMatrix(rowsLength: number, columnsLength: number):
Array<Array<number>> {
    const matrix: Array<Array<number>> = new Array<Array<number>>(rowsLength);

    for (let rowIndex = 0; rowIndex < rowsLength; ++rowIndex) {
      matrix[+rowIndex] = new Array<number>(columnsLength).fill(0);
    }

    return matrix;
  }
}
```

Random.ts

```
import { randomFillSync } from 'crypto';

import { IllegalArgumentException } from '../exceptions/IllegalArgumentException';

export class Random {
  public static readonly MAX_INTEGER: number = 2 ** 32 - 1;

  private readonly min: number;
  private readonly max: number;

  constructor(min: number, max: number) {
    if (min >= max) {
      throw new IllegalArgumentException(`Max value must be more than min value.`);
    }

    this.min = min;
    this.max = max;
  }

  public nextInt(): number {
    const randomNumber: number = randomFillSync(new Uint32Array(1))[0];
    return this.min + (randomNumber % (this.max + 1 - this.min));
  }

  public static generateSequence(size: number, min: number, max: number): Array<number> {
```

```
    const generator: Random = new Random(min, max);

    const sequence: Array<number> = new Array<number>(size);
    for (let index = 0; index < size; ++index) {
      sequence[+index] = generator.nextInt();
    }

    return sequence;
  }
}
```

index.ts

```
import * as inquirer from 'inquirer';
import { parse, resolve } from 'path';
import { stdout } from 'process';
import { CountersNumberKeys } from './crypto/counters/CountersNumber';
import { ShareSecretKey } from './crypto/key/ShareSecretKey';
import { BinaryDataReader } from './crypto/plaindata/binarydata/BinaryDataReader';
import { BinaryDataTransformer } from './crypto/plaindata/binarydata/BinaryDataTransformer';
import { BinaryDataWriter } from './crypto/plaindata/binarydata/BinaryDataWriter';
import { BMPDataReader } from './crypto/plaindata/bmp/BMPDataReader';
import { BMPDataTransformer } from './crypto/plaindata/bmp/BMPDataTransformer';
import { BMPDataWriter } from './crypto/plaindata/bmp/BMPDataWriter';
import { Data } from './crypto/plaindata/Data';
import { DataReader } from './crypto/plaindata/DataReader';
import { DataTransformer } from './crypto/plaindata/DataTransformer';
import { DataWriter } from './crypto/plaindata/DataWriter';
import { SharedPart, ShareSecret } from './crypto/ShareSecret';
import { Questions } from './Questions';
inquirer.registerPrompt('fuzzypath', require('inquirer-fuzzy-path'));

interface DataActions<T extends Data> {
  reader: DataReader<T>;
  writer: DataWriter<T>;
  transformer: DataTransformer<T>;
}

class Main {
  private static getDataActions(ext: string): DataActions<Data> {
    switch (ext) {
      case '.bmp': {
        return {
          reader: new BMPDataReader(),
          writer: new BMPDataWriter(),
          transformer: new BMPDataTransformer()
        };
      }
      default: {
        return {
          reader: new BinaryDataReader(),
          writer: new BinaryDataWriter(),
          transformer: new BinaryDataTransformer()
        };
      }
    }
  }

  private static async generateKey(): Promise<void> {
    const { distPath } = await inquirer.prompt([Questions.DIST_PATH]);
    const { N, K } = await inquirer.prompt([Questions.N, Questions.K]);
```

```
    const { countersNumberKeys } = await inquirer.prompt([Questions.COUNTERS_NUMBER_KEYS]);

    const shareKey: ShareSecretKey = ShareSecretKey.generateKey(N, K, countersNumberKeys as CountersNumberKeys);
    ShareSecretKey.write(shareKey, resolve(distPath, `key_N_${N}_K_${K}_${countersNumberKeys}.json`));
  }

  private static async shareSecret(): Promise<void> {
    const { srcFilePath } = await inquirer.prompt([Questions.SRC_FILE_PATH]);
    const { keyPath } = await inquirer.prompt([Questions.KEY_PATH]);
    const { distPath } = await inquirer.prompt([Questions.DIST_PATH]);

    const { name, ext } = parse(srcFilePath);
    const { reader, writer, transformer } = Main.getDataActions(ext);

    const data = transformer.preEncode(reader.read(srcFilePath));
    const shareKey: ShareSecretKey = ShareSecretKey.read(keyPath);

    const shareParts = ShareSecret.encode(data.getData(), shareKey);

    transformer.postEncode(data, shareParts).forEach((part: Data, index: number) => {
      writer.write(part, resolve(distPath, `${index + 1}.share.${name}${ext}`));
    });
  }

  private static async recoverSecret(): Promise<void> {
    const { keyPath } = await inquirer.prompt([Questions.KEY_PATH]);
    const recoverKey: ShareSecretKey = ShareSecretKey.read(keyPath);
    const partPaths: { [key: string]: string } = await inquirer.prompt(Questions.PART_PATH(recoverKey.K));
    const { distPath } = await inquirer.prompt([Questions.DIST_PATH]);

    const { name, ext } = parse(partPaths.partPath0);

    const { reader, writer, transformer } = Main.getDataActions(ext);

    const sharedParts: Array<SharedPart> = Object.values(partPaths).map(
      (value: string): SharedPart => {
        const partData = reader.read(resolve(value));
        const { name } = parse(value);
        return {
          index: parseInt(name) - 1,
          partData: partData
        };
      }
    );

    const decodeData: Buffer = ShareSecret.decode(transformer.preDecode(sharedParts), recoverKey);

    const namePart = name.split('.');
    const fileName: string =
      namePart[namePart.length - 1] === '' ? `restore.secret${ext}` : `${namePart[namePart.length - 1]}${ext}`;

    writer.write(transformer.postDecode(sharedParts, decodeData), resolve(distPath, fileName));
  }

  public static async main(): Promise<void> {
```

```
let isRun = true;
while (isRun) {
  stdout.write('\x1Bc');
  stdout.write('\x1Bc');
  console.log('======= *** Secret Sharing System *** =======');

  const { action } = await inquirer.prompt([Questions.CHOOSE_ACTION]);

  switch (action) {
    case 'Generate key': {
      await Main.generateKey();
      await inquirer.prompt([Questions.WAIT_ENTER]);
      break;
    }
    case 'Share secret': {
      await Main.shareSecret();
      await inquirer.prompt([Questions.WAIT_ENTER]);
      break;
    }
    case 'Recover secret': {
      await Main.recoverSecret();
      await inquirer.prompt([Questions.WAIT_ENTER]);
      break;
    }
    default: {
      isRun = false;
    }
  }
 }
}

Main.main();
```

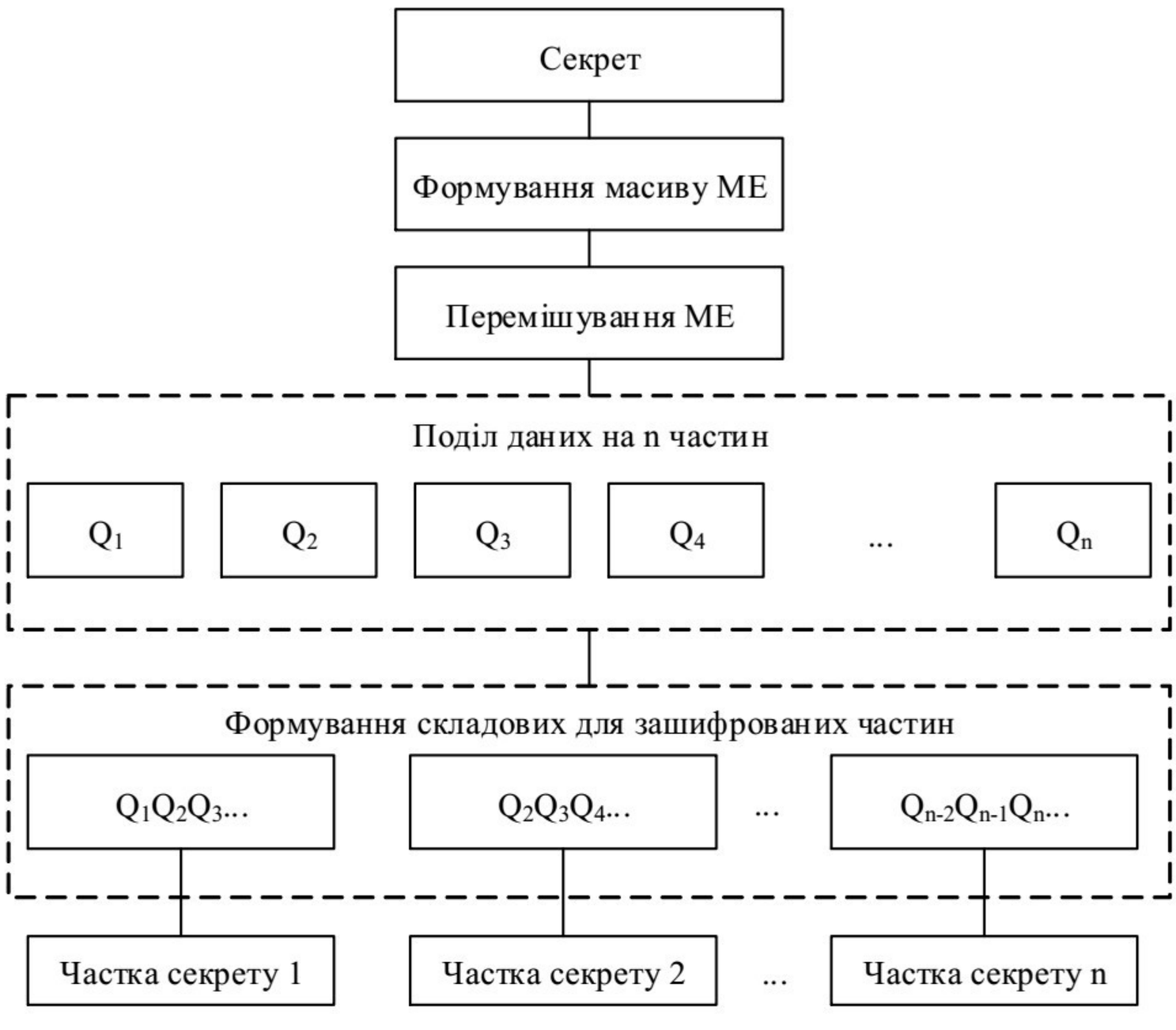
Questions.ts

```
interface InquirerPromptConfig {
  type: string;
  name: string;
  message: string;
  choices?: Array<string | number>;
  excludePath?: (nodePath: string) => boolean;
  itemType?: string;
  depthLimit?: number;
}

const excludePathCallback = (nodePath: string): boolean =>
  nodePath.startsWith('node_modules') || nodePath.startsWith('.git') ||
  nodePath.startsWith('.idea') || nodePath.startsWith('.vscode');

export const Questions = {
  WAIT_ENTER: {
    type: 'string',
    name: 'enter',
    message: 'Please press enter ...'
  },
  CHOOSE_ACTION: {
    type: 'list',
    name: 'action',
    message: 'What do you want to do ?',
    choices: ['Generate key', 'Share secret', 'Recover secret', 'Exit']
```

```
},
DIST_PATH: {
  type: 'fuzzypath',
  name: 'distPath',
  excludePath: excludePathCallback,
  itemType: 'directory',
  message: 'Enter dist directory:',
  depthLimit: 5
},
KEY_PATH: {
  type: 'fuzzypath',
  name: 'keyPath',
  excludePath: excludePathCallback,
  itemType: 'file',
  message: 'Enter path to key file:',
  depthLimit: 5
},
SRC_FILE_PATH: {
  type: 'fuzzypath',
  name: 'srcFilePath',
  excludePath: excludePathCallback,
  itemType: 'file',
  message: 'Enter path to source file:',
  depthLimit: 5
},
PART_PATH: (amount: number): Array<InquirerPromptConfig> => {
  const questions: Array<InquirerPromptConfig> = [];

  for (let index = 0; index < amount; ++index) {
    questions.push({
      type: 'fuzzypath',
      name: `partPath${index}`,
      excludePath: excludePathCallback,
      itemType: 'file',
      message: `Enter path to part #${index + 1}:`,
      depthLimit: 5
    });
  }
  return questions;
},
N: {
  type: 'number',
  name: 'N',
  message: 'Enter N: '
},
K: {
  type: 'number',
  name: 'K',
  message: 'Enter K: '
},
COUNTERS_NUMBER_KEYS: {
  type: 'list',
  name: 'countersNumberKeys',
  message: 'Choose counters number keys',
  choices: ['_2', '_4', '_8', '_16', '_32', '_64', '_128', '_256', '_512',
    '_1024', '_2048', '_4096', '_8192', '_16384']
}
};
```

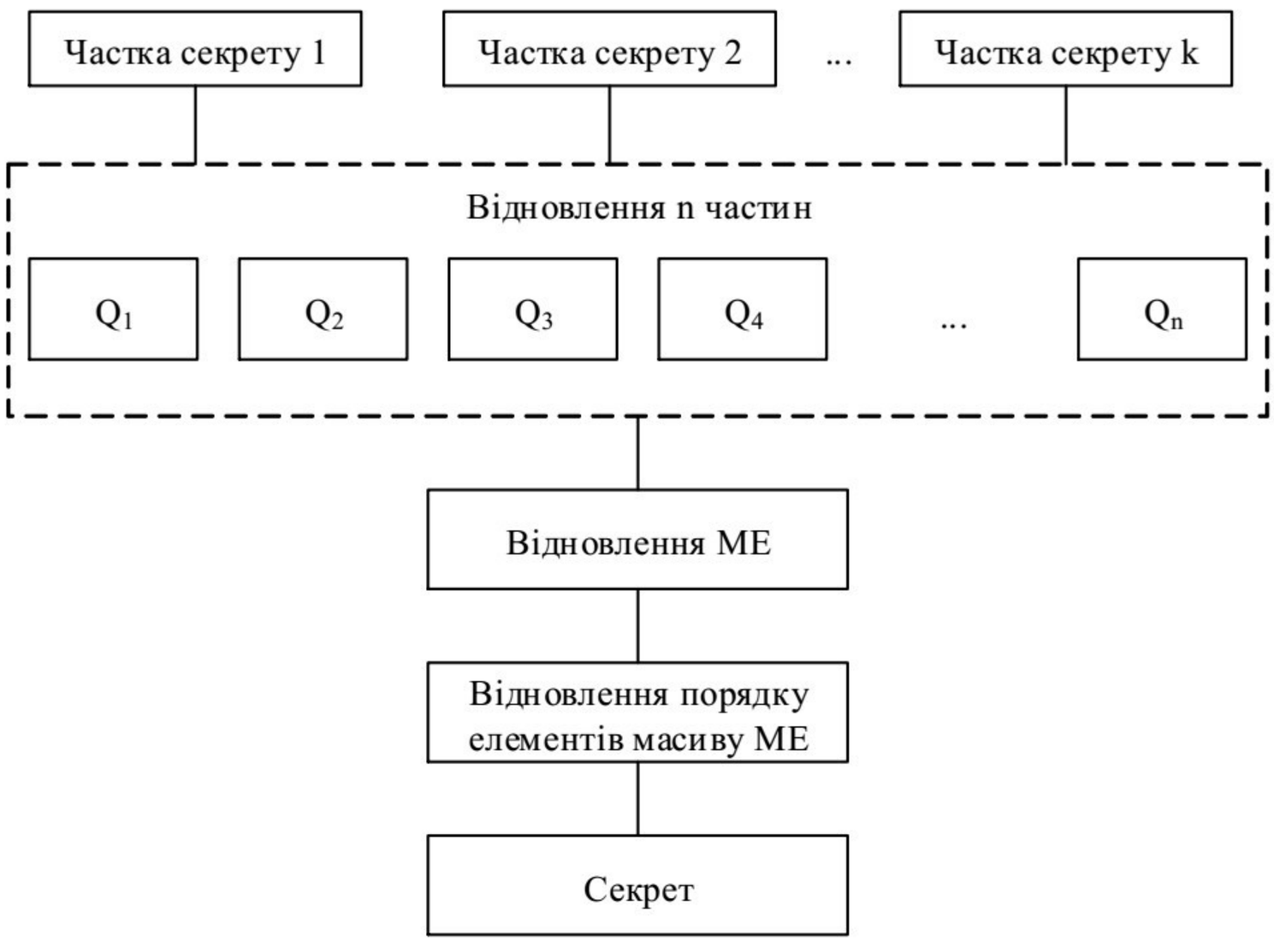
ІЛЮСТРАТИВНА ЧАСТИНА

Етапи розподілу секрету



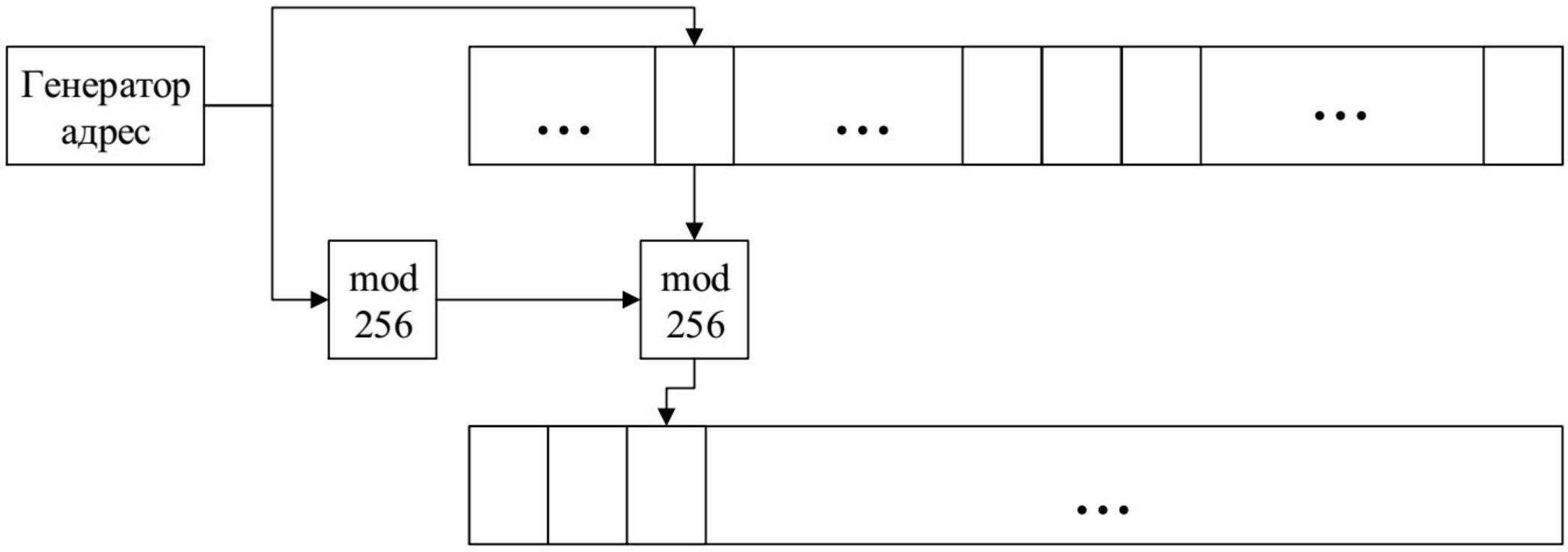
					08-20.МКР.005.00.000 ІЧ1		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Етапи розподілу секрету.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				1 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Етапи відновлення секрету



					08-20.МКР.005.00.000 ІЧ2		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Етапи відновлення секрету.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				2 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Схема перестановок вмісту секрету



					08-20.МКР.005.00.000 ІЧЗ		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Схема перестановок вмісту секрету	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				3 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Масиви елементів секрету

Масив QD

$$QD=\begin{bmatrix} Q_1 & Q_2 & Q_3 & \cdots & Q_{i-1} & Q_i \\ Q_2 & Q_3 & Q_4 & \cdots & Q_i & Q_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_i & Q_1 & Q_2 & \cdots & Q_{i-2} & Q_{i-1} \end{bmatrix}$$

Масив QE для непарного i

$$QE=\begin{bmatrix} Q_1 & K_{2,1}\cdot Q_2+Q_3 & \cdots & K_{i-1,1}\cdot Q_{i-1}+Q_i \\ Q_2 & K_{3,1}\cdot Q_3+Q_4 & \cdots & K_{i,1}\cdot Q_i+Q_1 \\ \cdots & \cdots & & \cdots \\ Q_i & K_{1,m}\cdot Q_1+Q_2 & \cdots & K_{i-2,m}\cdot Q_{i-2}+Q_{i-1} \end{bmatrix}$$

Масив QE для парного i

$$QE=\begin{bmatrix} Q_1 & Q_2 & K_{3,1}\cdot Q_3+Q_4 & \cdots & K_{i-1,1}\cdot Q_{i-1}+Q_i \\ Q_2 & Q_3 & K_{4,1}\cdot Q_4+Q_5 & \cdots & K_{i,1}\cdot Q_i+Q_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Q_i & Q_1 & K_{2,m}\cdot Q_2+Q_3 & \cdots & K_{i-2,m}\cdot Q_{i-2}+Q_{i-1} \end{bmatrix}$$

					08-20.МКР.005.00.000 ІЧ4		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Масиви елементів секрету .	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				4 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Приклад для порогової схеми (3, 7)

Масив QE з розподіленими частинами

$$QE = \begin{bmatrix} Q_1 & 1\cdot Q_2 + Q_3 & 1\cdot Q_4 + Q_5 \\ Q_2 & 1\cdot Q_3 + Q_4 & 1\cdot Q_5 + Q_6 \\ Q_3 & 2\cdot Q_4 + Q_5 & 1\cdot Q_6 + Q_7 \\ Q_4 & 2\cdot Q_5 + Q_6 & 1\cdot Q_7 + Q_1 \\ Q_5 & 2\cdot Q_6 + Q_7 & 1\cdot Q_1 + Q_2 \\ Q_6 & 2\cdot Q_7 + Q_1 & 2\cdot Q_2 + Q_3 \\ Q_7 & 2\cdot Q_1 + Q_2 & 2\cdot Q_3 + Q_4 \end{bmatrix}$$

Відновлення за допомогою 2, 4 та 7 частин

$$\begin{bmatrix} Q_2 & 1\cdot Q_3 + Q_4 & 1\cdot Q_5 + Q_6 \\ Q_4 & 2\cdot Q_5 + Q_6 & 1\cdot Q_7 + Q_1 \\ Q_7 & 2\cdot Q_1 + Q_2 & 2\cdot Q_3 + Q_4 \end{bmatrix}$$

$$Q_1 = (1\cdot Q_7 + Q_1) - Q_7,$$

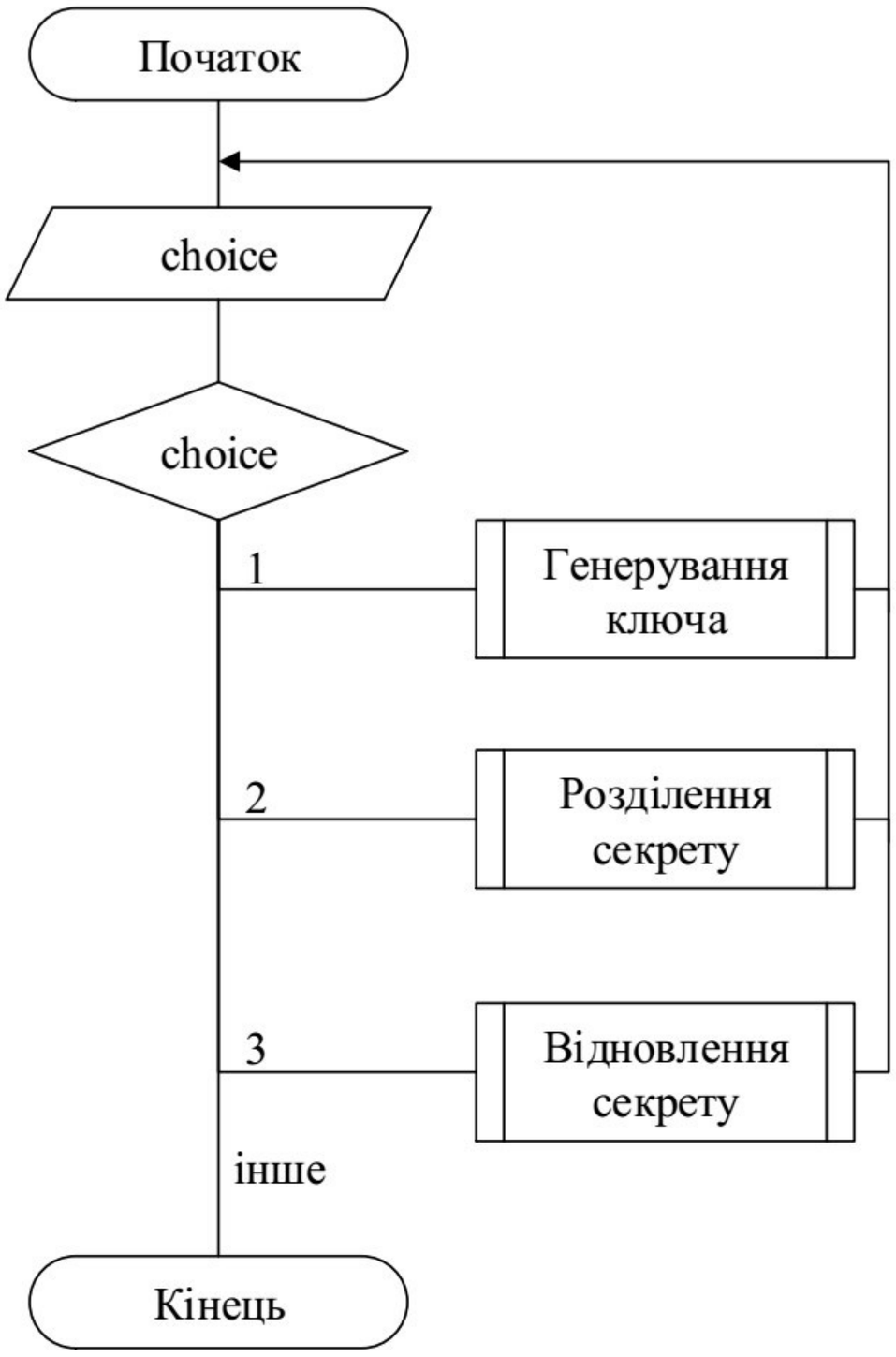
$$Q_3 = (1\cdot Q_3 + Q_4) - Q_4,$$

$$Q_5 = (2\cdot Q_5 + Q_6) - (1\cdot Q_5 + Q_6),$$

$$Q_6 = (1\cdot Q_5 + Q_6) - Q_5.$$

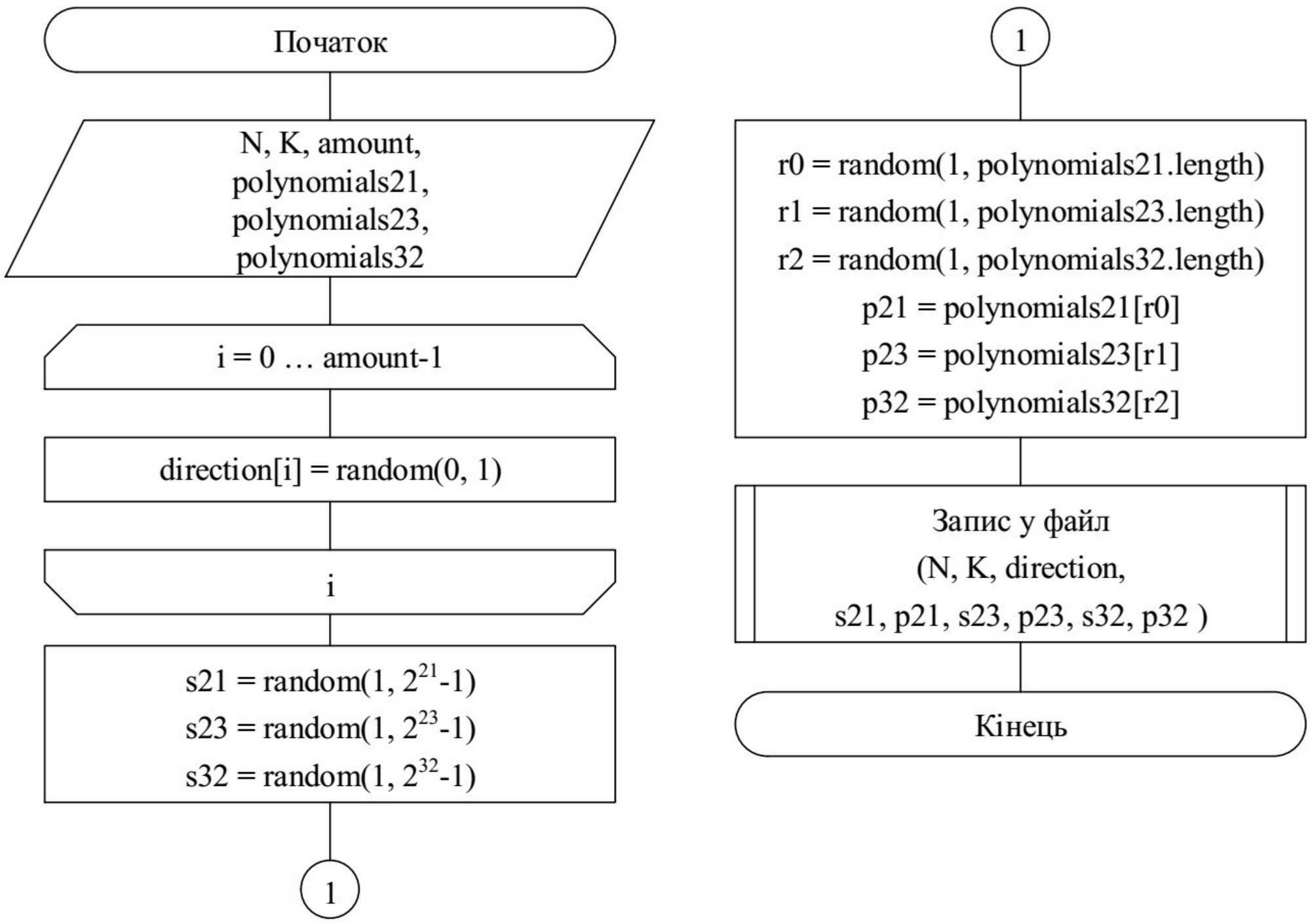
					08-20.МКР.005.00.000 ІЧ5	
Змн	Арк.	№ докум.	Підпис	Дата		
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Приклад для порогової схеми (3, 7)	Літ. / Маса / Масштаб
Перевір.		Луже				
Реценз.		Крупельницький				5 ВНТУ, гр. БС-18м
Н. Контр.		Луже				
Затверд.		Лужец				

Алгоритм роботи програми



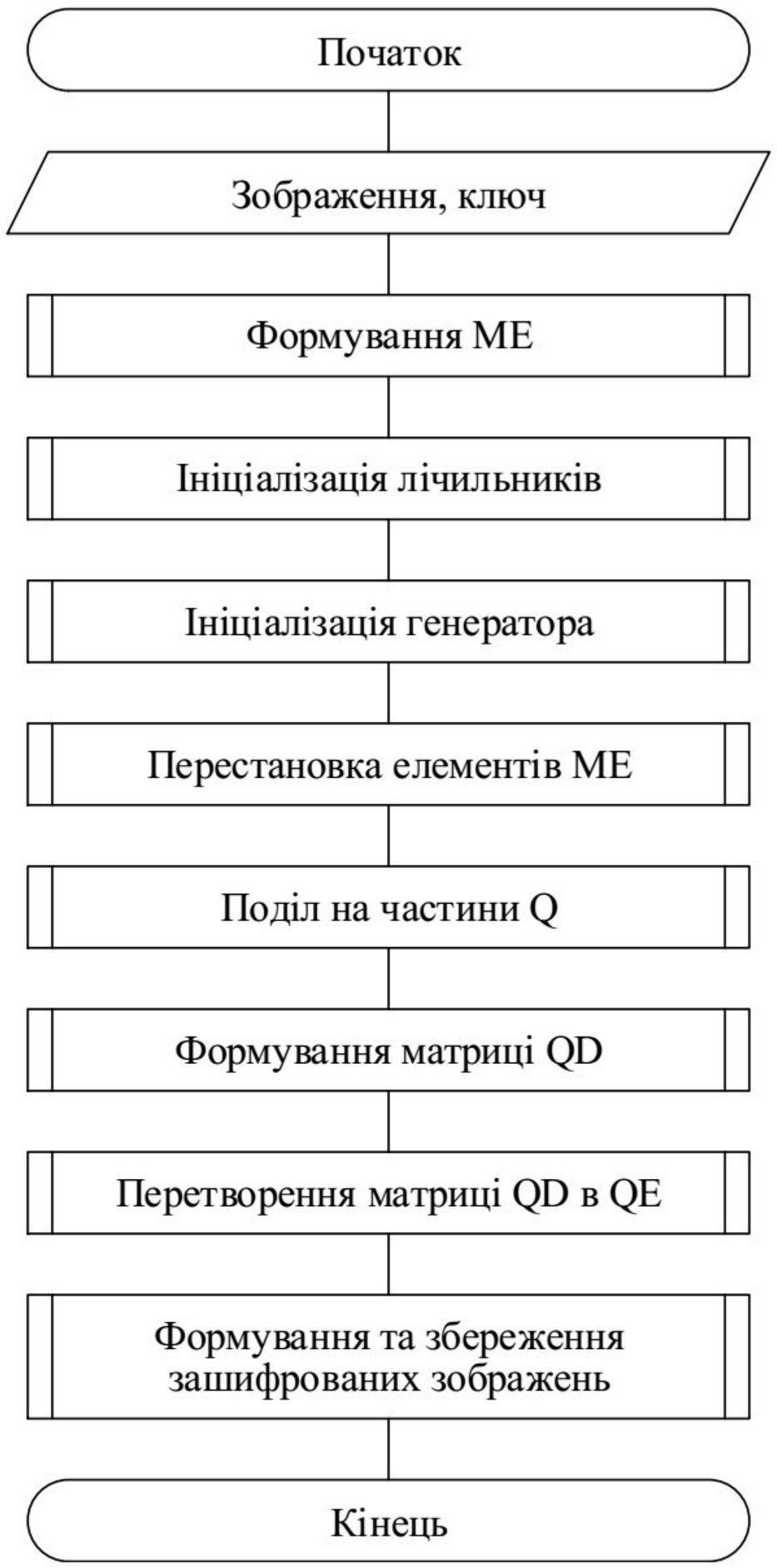
					08-20.МКР.005.00.000 ІЧ6		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Алгоритм роботи програми.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				6 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Алгоритм генерування ключа



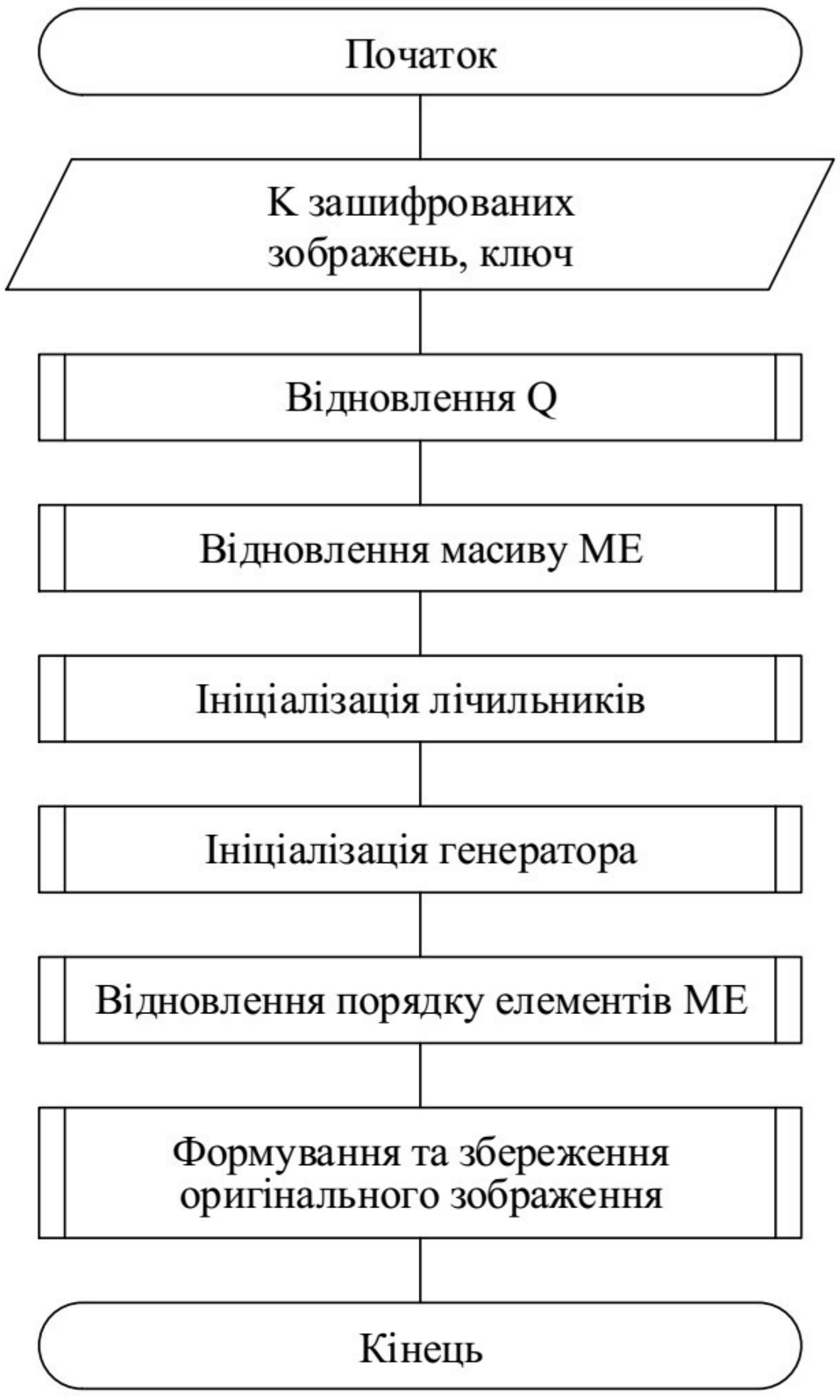
					08-20.МКР.005.00.000 ІЧ7		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Алгоритм генерування ключа.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				7 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Алгоритм розподілу секрету



					08-20.МКР.005.00.000 ІЧ8		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Алгоритм розподілу секрету.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				8 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Алгоритм відновлення секрету



					08-20.МКР.005.00.000 ІЧ9		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Алгоритм відновлення секрету.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				9 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Приклад розділення зображення за схемою (6, 3) з використанням 8 лічильників

Вигляд секретного зображення

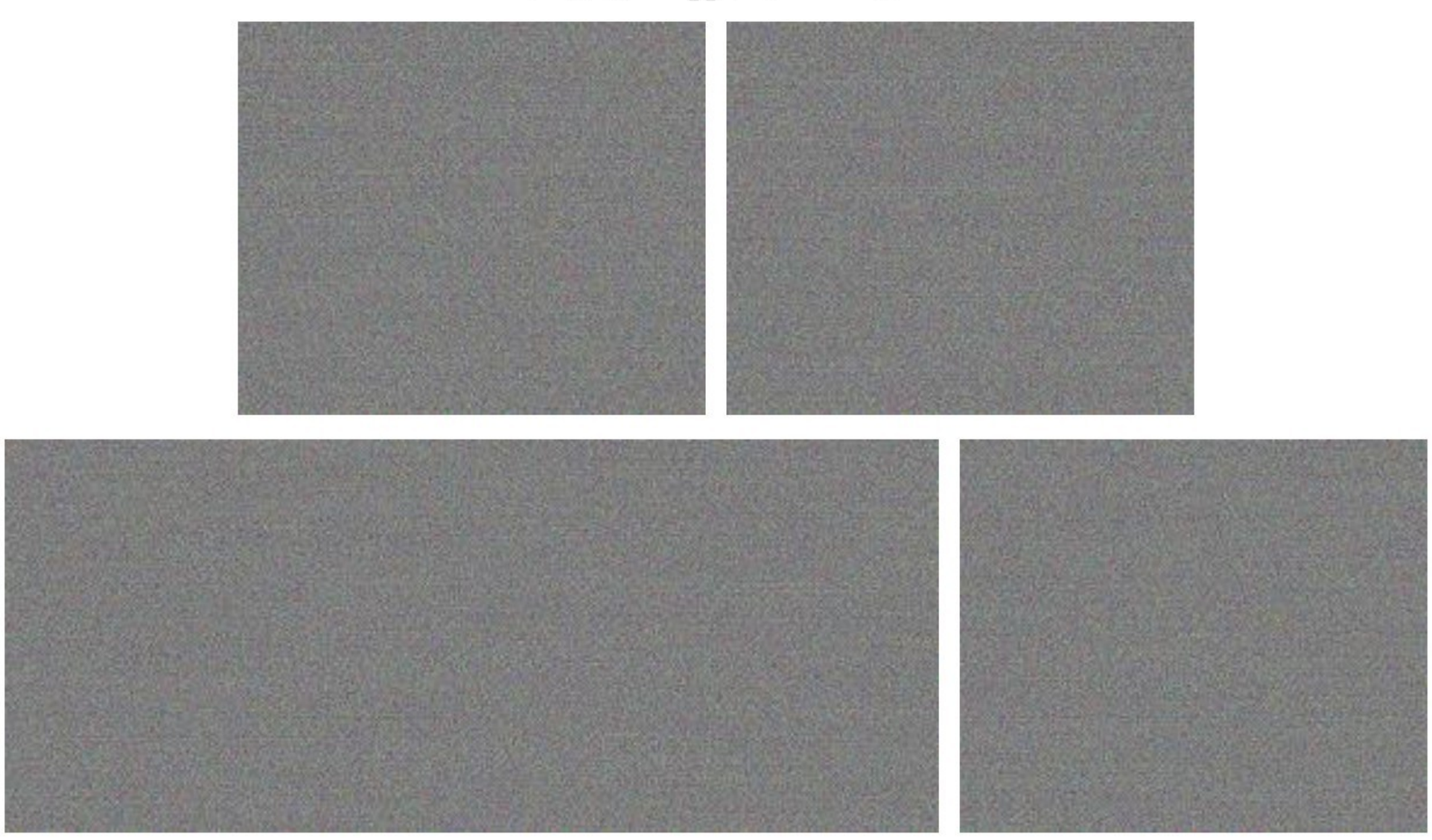
Вигляд зашифрованих частин

					08-20.МКР.005.00.000 ІЧ10		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Приклад розділення зображення за схемою (6, 3) з використанням 8 лічильників	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				10 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Приклад розділення зображення за схемою (6, 3) з використанням 64 лічильників

Вигляд секретного зображення

Вигляд зашифрованих частин



					08-20.МКР.005.00.000 ІЧ11		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Вигляд зашифрованих зображень з використанням 64 лічильників.	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				11 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					

Приклад відновлення зображення за схемою (6, 3) з використанням лише двох коректних частин

Частини 1, 2 та 3

Відновлене зображення

					08-20.МКР.005.00.000 ІЧ12		
Змн	Арк.	№ докум.	Підпис	Дата			
Розроб.		Дехтяренко М			Метод та засіб розподілення секрету. Приклад відновлення зображення за схемою (6, 3) з використанням лише двох коректних частин	Літ.	Маса / Масштаб
Перевір.		Луже					
Реценз.		Крупельницький				12 ВНТУ, гр. БС-18м	
Н. Контр.		Луже					
Затверд.		Лужец					