МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

РАДЧУК Ростислав Ігорович

Метод шифрування зображень в системі залишкових класів / A method of encryption images in the residue number system

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБзм -21
Р.І. Радчук

Науковий керівник
Н.Г.Яцків

Кваліфікаційну роботу
допущено до захисту:

«____» _____ ___2024 р.

Завідувач кафедри

_____ В.В.Яцків

ТЕРНОПІЛЬ – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 – Кібербезпека та захист інформації
освітньо-професійна програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
«____» _____ 20___ року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

РАДЧУК РОСТИСЛАВ ІГОРОВИЧ
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Метод шифрування зображень в системі залишкових класів / A method of encryption images in the residue number system
керівник роботи к.т.н., доцент Н.Г. Яцків
затверджені наказом по університету від 12 грудня 2023 року № 753

2. Строк подання студентом закінченої кваліфікаційної роботи 12 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
– оглянути основи системи залишкових класів;
– дослідити методи шифрування цифрових зображень;
–проаналізувати властивості та характеристики системи залишкових класів;
–проаналізувати математичну модель представлення цифрових зображень в системі залишкових класів;
–проаналізувати алгоритм шифрування зображень з використанням сзк;
–дослідити особливості зворотного перетворення зображень із системи залишкових класів;
–виконати програмну реалізацію розроблених алгоритмів на основі сзк;
–реалізувати механізм шифрування зображень у розробленій системі;

5. Перелік графічного матеріалу у роботі:
– Процес блокового шифрування
– Приклад шифрування одного пікселя RGB зображення

– Процес шифрування даних з використанням системи залишкових класів
– Схема роботи алгоритму шифрування з використанням СЗК.
– Алгоритм обчислення СЗК пікселів
– Схема роботи алгоритму відновлення початкового зображення

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.2023 р. – 03.2024 р.	
2	Дослідження алгоритмів шифрування зображень в сзк	03.2024 р. – 06.2024 р.	
3	Реалізація та дослідження розроблених алгоритмів	06.2024 р. – 11.2024 р.	

Студент _____ Радчук Р.І.
(підпис)

Керівник роботи _____ Н.Г. Яцків

АНОТАЦІЯ

Кваліфікаційна робота на тему «Метод шифрування зображень в системі залишкових класів» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека та захист інформації» освітньо-професійної програми «Кібербезпека» написана обсягом 74 сторінок і містить 52 ілюстрації, 2 таблиці, 4 додатки та 34 джерела за переліком посилань.

Метою роботи є розробка та вдосконалення алгоритмів шифрування зображень з використанням системи залишкових класів для підвищення ефективності захисту візуальної інформації.

Методи досліджень: математичне моделювання, методи теорії чисел, методи криптографічного аналізу, порівняльний аналіз, комп'ютерне моделювання, методи статистичного аналізу, експериментальні дослідження швидкодії алгоритмів.

Результати дослідження: Розроблено математичну модель представлення цифрових зображень в системі залишкових класів, створено ефективні алгоритми шифрування та дешифрування на основі СЗК, виконано програмну реалізацію розроблених методів, проведено експериментальне дослідження їх швидкодії та порівняльний аналіз з існуючими методами шифрування зображень.

Результати роботи можуть бути застосовані при проектуванні систем захисту графічної інформації в різних галузях.

Ключові слова: СИСТЕМА ЗАЛИШКОВИХ КЛАСІВ, ШИФРУВАННЯ ЗОБРАЖЕНЬ, КРИПТОГРАФІЧНИЙ ЗАХИСТ, ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ.

ABSTRACT

Qualification work on "A method of encryption images in the residue number system" for the degree of "Master" in the specialty 125 "Cybersecurity and Information Protection" educational and professional program "Cybersecurity" is written in 74 pages and contains 52 illustrations, 2 tables, 4 appendices and 34 sources according to the list of links.

The purpose of the work is to develop and improve image encryption algorithms using the residue number system to increase the effectiveness of visual information protection.

Research methods: mathematical modeling, number theory methods, cryptographic analysis methods, comparative analysis, computer modeling, statistical analysis methods, experimental research of algorithm performance.

Research results: A mathematical model for representing digital images in the residue number system has been developed, efficient encryption and decryption algorithms based on RNS have been created, software implementation of the developed methods has been performed, experimental research of their performance and comparative analysis with existing image encryption methods has been conducted.

The results of the work can be applied in the design of graphic information protection systems in various industries.

Keywords: RESIDUE NUMBER SYSTEM, IMAGE ENCRYPTION, CRYPTOGRAPHIC PROTECTION, DIGITAL IMAGE PROCESSING, PARALLEL COMPUTING.

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ

СЗК - Система залишкових класів.

ТКЗ - Теорема про китайські залишки.

НСД - Найбільший спільний дільник.

НСК - Найменше спільне кратне.

RGB - Red, Green, Blue.

AES - Advanced Encryption Standard.

DES - Data Encryption Standard.

RSA - Rivest-Shamir-Adleman.

ВСТУП

Актуальність теми. Традиційні методи шифрування, розроблені для текстових даних, не завжди ефективні для зображень через їх специфічні характеристики - великий обсяг даних, високу кореляцію між сусідніми пікселями та вимоги до швидкості обробки в реальному часі. Це створює необхідність розробки спеціалізованих методів шифрування, які б враховували особливості цифрових зображень.

Система залишкових класів (СЗК) представляє особливий інтерес для вирішення цієї задачі, оскільки забезпечує природний паралелізм обчислень та має потенціал для підвищення криптостійкості шифрування. Використання СЗК дозволяє ефективно розподіляти обчислювальне навантаження та зменшувати час шифрування/дешифрування, що особливо важливо при обробці великих обсягів графічних даних.

Мета і завдання дослідження. Метою дослідження є розробка та вдосконалення алгоритмів шифрування зображень з використанням системи залишкових класів для підвищення ефективності захисту візуальної інформації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Оглянути основи системи залишкових класів.

2. Дослідити методи шифрування цифрових зображень.

3. Проаналізувати властивості та характеристики системи залишкових класів.

4. Проаналізувати математичну модель представлення цифрових зображень в системі залишкових класів.

5. Проаналізувати алгоритм шифрування зображень з використанням СЗК.

6. Дослідити особливості зворотного перетворення зображень із системи залишкових класів.

7. Оптимізувати складність розшифрування розроблених алгоритмів.

8. Виконати програмну реалізацію розроблених алгоритмів на основі СЗК.

9. Реалізувати механізм шифрування зображень у розробленій системі.

10. Реалізувати графічний інтерфейс для програмного рішення.

11. Провести експериментальне дослідження швидкодії розроблених алгоритмів.

Об'єктом дослідження є процеси захисту цифрових зображень від несанкціонованого доступу в інформаційних системах.

Предметом дослідження є алгоритми шифрування зображень з використанням системи залишкових класів.

Методи дослідження: математичне моделювання, методи теорії чисел, методи криптографічного аналізу, порівняльний аналіз, комп'ютерне моделювання, методи статистичного аналізу, експериментальні дослідження швидкодії алгоритмів.

Наукова новизна: Наукова новизна отриманих результатів полягає у розробці нових та вдосконаленні існуючих алгоритмів шифрування зображень на основі системи залишкових класів, що забезпечує підвищення криптостійкості та швидкодії обробки даних. Вперше запропоновано математичну модель представлення цифрових зображень в СЗК, що дозволяє оптимізувати процес шифрування.

Практичне значення: Практичне значення отриманих результатів полягає у розробці програмного забезпечення для шифрування зображень з використанням СЗК, що може бути впроваджено в системах захисту візуальної інформації. Розроблені алгоритми дозволяють підвищити швидкість обробки даних при збереженні необхідного рівня захисту. Результати дослідження можуть бути використані при проектуванні систем захисту графічної інформації в різних галузях.

Результати дослідження: Результати включають розробку математичної моделі представлення зображень в СЗК, створення ефективних алгоритмів шифрування та дешифрування на основі СЗК, програмну реалізацію розроблених методів, експериментальне дослідження їх швидкодії та порівняльний аналіз з існуючими методами шифрування зображень.

Публікації та апробація до магістерської роботи:

1. Р. Радчук. Аналіз безпеки шифрування зображень у системі залишкоих класів. Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. С. 43-47

2. Р. Радчук. Аналіз ефективності системи залишкових класів у криптографічних методах захисту зображень. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. С. 87-92

1 ТЕОРЕТИЧНІ ОСНОВИ АЛГОРИТМІВ ШИФРУВАННЯ ЗОБРАЖЕНЬ В СЗК

1.1. Основи системи залишкових класів

Система залишкових класів (СЗК) являє собою непозиційну систему числення, в якій числа представляються у вигляді набору найменших невід'ємних залишків від ділення на фіксований набір взаємно простих чисел, що називаються модулями системи [1]. СЗК базується на китайській теоремі про залишки, згідно з якою число однозначно відновлюється за його залишками від ділення на взаємно прості модулі.

Китайська теорема про залишки була сформована на основі НСК. Ця теорема використовує НСК модулів для визначення повного циклу системи числення та гарантує однозначність представлення чисел у СЗК. Метод відновлення числа базується на властивості НСК забезпечувати найменший період повторення комбінацій залишків. В китайській теоремі НСК модулів визначає верхню межу діапазону чисел, які можна однозначно представити в даній системі залишкових класів. При знаходженні розв'язку системи порівнянь за китайською теоремою використовується властивість НСК взаємно простих модулів дорівнювати їх добутку [2].

Якщо описувати простими словами механізм роботи теореми про китайські залишки, то: “В кошику лежать яйця, і коли їх рахують трійками, залишається 2 яйця, коли рахують п'ятірками - залишається 3 яйця, а коли рахують сімками - залишається 2 яйця. Математично це означає, що шукане число при діленні на 3 дає залишок 2, при діленні на 5 дає залишок 3, а при діленні на 7 дає залишок 2. За допомогою китайської теореми про залишки можна знайти це число - 23 яйця.

Перевірка показує правильність розв'язку: двадцять три яйця при розкладанні по три дають сім повних трійок і два яйця залишку, при розкладанні по п'ять дають чотири повних п'ятірки і три яйця залишку, а при розкладанні по сім дають три повних сімки і два яйця залишку. Через те що

числа 3, 5 і 7 взаємно прості, наступне число з такими ж залишками буде більше на їх найменше спільне кратне, тобто на 105. Цей приклад наочно демонструє практичне застосування теорії порівнянь та системи залишкових класів для розв'язання реальних задач" [3].

Тобто, найменше спільне кратне або ж НСК - це найменше натуральне число, яке ділиться на всі ці числа без остачі. НСК має основоположне значення для побудови системи залишкових класів, оскільки визначає період повторення наборів залишків при діленні чисел на взаємно прості модулі. В комп'ютерних обчисленнях НСК слугує базою для визначення робочого діапазону СЗК та вибору оптимальних модулів системи. Знаходження НСК базується на розкладі чисел на прості множники з подальшим відбором найвищих степенів спільних множників. При проектуванні систем обробки даних на основі СЗК вибір взаємно простих модулів здійснюється з урахуванням їх НСК для забезпечення необхідного діапазону представлення чисел [4].

Власне, в системі СЗК числа представляються у формі кортежу залишків, що дозволяє виконувати арифметичні операції паралельно по кожному з модулів [5]. Математичний апарат СЗК заснований на теорії порівнянь та властивостях взаємно простих чисел. Принцип роботи СЗК полягає в тому, що велике число можна представити набором малих чисел - залишків від ділення на взаємно прості модулі. Діапазон представлення чисел визначається добутком всіх модулів системи. При виборі модулів враховується специфіка задачі та вимоги до швидкодії та точності обчислень. Для ефективної реалізації на комп'ютері модулі зазвичай обираються близькими до степенів двійки. Перехід від позиційного представлення до СЗК здійснюється шляхом послідовного ділення числа на кожен модуль та знаходження відповідних залишків. Зворотне перетворення виконується з використанням китайської теореми про залишки. На рисунку 1.1 зображено математичний принцип роботи системи залишкових класів.

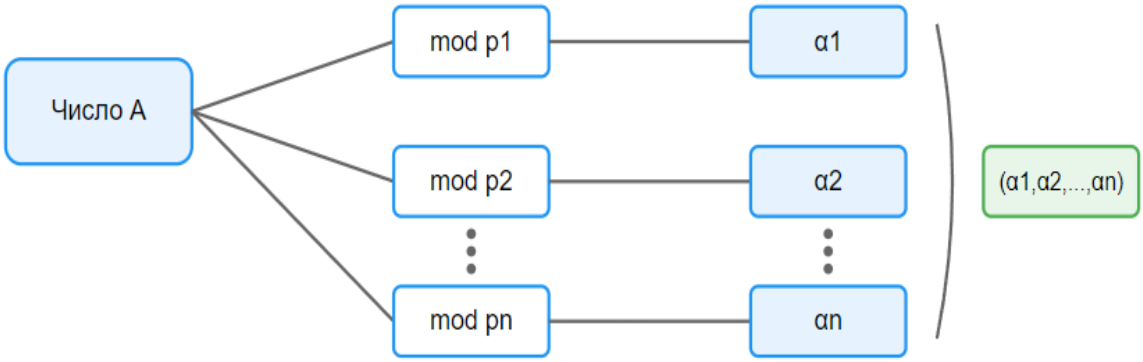

Рисунок 1.1 – Математичний принцип роботи СЗК

Процес роботи системи залишкових класів можна описати через основні етапи перетворення числа А. На початку вхідне число А потрапляє до системи, де над ним виконуються паралельні операції ділення на взаємно прості модулі *p1, p2, ..., pn*. Кожна операція ділення виконується незалежно від інших та дає відповідний залишок *α1, α2, ..., αn* [6]. В результаті цих паралельних операцій формується набір залишків *(α1, α2, ..., αn)*, який однозначно представляє вхідне число А в системі залишкових класів.

Всі подальші арифметичні операції в СЗК виконуються над цими залишками незалежно по кожному модулю. Наприклад, при додаванні двох чисел, представлених в СЗК, додаються їх відповідні залишки за модулями, при цьому операції виконуються паралельно без необхідності враховувати міжрозрядні переноси [6]. При множенні чисел аналогічно перемножуються відповідні залишки за модулями. Зворотне перетворення з системи залишкових класів у звичайне представлення числа виконується за допомогою китайської теореми про залишки, при цьому НСК модулів визначає робочий діапазон системи.

Така організація обчислень забезпечує природний паралелізм операцій та відсутність переносів між розрядами, що суттєво підвищує швидкодію та надійність обчислень [7]. В цифрових системах це дозволяє ефективно розпаралелювати процес обробки даних та зменшити апаратні затрати при

реалізації арифметичних пристроїв. Саме тому СЗК знаходить широке застосування в системах цифрової обробки сигналів, криптографії та інших областях, де потрібна висока швидкість виконання арифметичних операцій.

Застосування СЗК у сучасних комп'ютерах відкриває нові можливості для створення потужних обчислювальних систем. Застосування СЗК у сучасних комп'ютерах відкриває нові можливості для створення потужних обчислювальних систем. В звичайній двійковій системі існує постійна проблема переносу одиниці в наступний розряд, як при додаванні "в стовпчик".

СЗК повністю позбавлена цієї проблеми - всі обчислення виконуються незалежно для кожного модуля, без потреби чекати результату від сусіднього розряду [8]. Це дозволяє створювати процесори, які працюють значно швидше за рахунок паралельного виконання операцій. При обробці цифрових сигналів, наприклад у аудіо чи відео системах, СЗК дозволяє будувати ефективніші фільтри з меншими апаратними витратами. В сфері захисту даних СЗК знайшла застосування для роботи з величезними числами, які використовуються в сучасних методах шифрування.

Система має здатність самостійно знаходити та виправляти помилки, які виникають під час обчислень - для цього просто додаються додаткові модулі, які працюють як контрольні суми [9].

1.2. Методи шифрування цифрових зображень за допомогою СЗК

Особливості застосування системи залишкових класів для шифрування даних полягають у розподілі інформації на незалежні залишки за взаємно простими модулями. При шифруванні кожен блок даних перетворюється в набір залишків, які обробляються паралельно та незалежно. Без знання повного набору модулів системи відновлення вихідних даних стає практично неможливим [10]. При передачі зашифрованих даних можливе використання різних каналів зв'язку для передачі окремих залишків, що підвищує

захищеність інформації. На рисунку 1.2 зображено схему процесу шифрування даних з використанням системи залишкових класів.

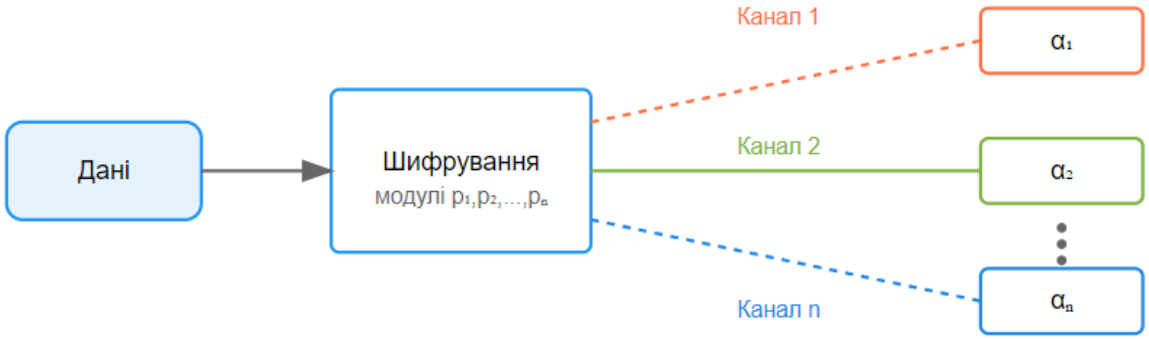

Рисунок 1.2 – Процес шифрування даних з використанням системи залишкових класів

На схемі, показаній на рисунку 1.2, відображено процес перетворення даних при шифруванні за допомогою системи залишкових класів. Процес починається з надходження вихідних даних до системи шифрування. Ці дані можуть бути будь-якого типу - текст, зображення чи інша цифрова інформація, яка потребує захисту. Вся вхідна інформація спочатку перетворюється у послідовність чисел, придатних для подальшої обробки [11].

Блок шифрування, розташований у центральній частині схеми, є головним елементом системи захисту даних. В цьому блоці відбувається розподіл вхідної інформації на частини за допомогою спеціально підібраних модулів. Кожен модуль - це число, яке використовується для формування частини зашифрованих даних [12]. Особливість полягає в тому, що модулі підбираються таким чином, щоб вони не мали спільних дільників. Це забезпечує неможливість відновлення інформації без знання всіх модулів системи.

Після обробки в блоці шифрування формуються залишки - частини зашифрованої інформації. Кожен такий залишок містить частину вихідних даних, але сам по собі не несе значущої інформації. Для передачі цих залишків

використовуються різні канали зв'язку, що показано на схемі різними типами ліній. Використання окремих каналів для кожного залишку суттєво підвищує захищеність системи, адже для перехоплення інформації зловмиснику потрібно отримати доступ до всіх каналів одночасно [13].

Відповідно, коли зрозуміло суть шифрування, варто також розписати методи, які є на даний момент наявні.

Перший метод полягає в обробці окремих пікселів зображення, де кожен піксель представляється набором залишків за обраними модулями [14]. При цьому колірні складові кожного пікселя перетворюються незалежно, що дозволяє зберегти структуру зображення при шифруванні.

Шифрування зображення на рівні окремих пікселів з використанням системи залишкових класів є одним з найбільш ефективних методів захисту графічної інформації [15]. Процес починається з представлення зображення у вигляді матриці пікселів, де кожен піксель характеризується своїми колірними складовими. У випадку RGB-зображення кожен піксель має три компоненти - червону (R), зелену (G) та синю (B), кожна з яких представлена числом від 0 до 255.

При шифруванні кожна колірна складова пікселя обробляється окремо. Наприклад, якщо розглядати значення червоного каналу пікселя, воно перетворюється в набір залишків за попередньо визначеними модулями системи [16]. Це означає, що число від 0 до 255 розкладається на кілька менших чисел - залишків від ділення на взаємно прості модулі. Аналогічна процедура виконується для зеленого та синього каналів. Оскільки обробка кожного колірного каналу відбувається незалежно, структура зображення зберігається, що важливо для подальшого відновлення.

Модулі повинні бути взаємно простими числами, а їх добуток має перевищувати максимальне можливе значення колірної складової (255 для стандартного 8-бітного представлення). При цьому кількість модулів визначає рівень захищеності - чим більше модулів використовується, тим складніше відновити оригінальне значення без знання всіх параметрів системи [17].

Процес шифрування виконується паралельно для всіх пікселів зображення, що забезпечує високу швидкість обробки. Для кожного пікселя формується набір залишків, який фактично є його зашифрованим представленням. Отримані набори залишків можуть передаватися різними каналами зв'язку, що додатково підвищує рівень захисту. Якщо розглядати на прикладі, то на рисунку 1.3 наведена схема розподілу RGB в СЗК по модулям.

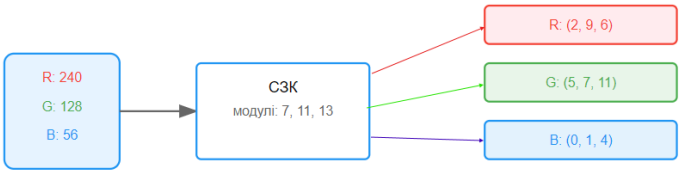

Рисунок 1.3 – Приклад шифрування одного пікселя RGB зображення

На схемі показано приклад шифрування одного пікселя RGB зображення, де кожен піксель кольорового зображення складається з трьох колірних компонент, що визначають його відтінок: червоної (R), зеленої (G) та синьої (B). Для 8-бітного представлення кольору кожна компонента може приймати значення від 0 до 255. В нашому прикладі піксель має значення R=240, G=128, B=56, що відповідає насиченому оранжевому кольору.

Для шифрування обрано три взаємно прості модулі: 7, 11 та 13. Для червоної складової (R=240) обчислюються залишки від ділення на кожен модуль. При діленні 240 на 7 отримуємо частку 34 і залишок 2 (240 = 34×7 + 2). При діленні на 11 отримуємо частку 21 і залишок 9 (240 = 21×11 + 9). При діленні на 13 отримуємо частку 18 і залишок 6 (240 = 18×13 + 6). Таким чином, червона складова представляється набором залишків (2, 9, 6). Аналогічні операції виконуються для зеленої складової (G=128). При діленні на модулі 7, 11, 13 отримуємо набір залишків (2, 7, 11), що представляє зашифроване значення зеленої компоненти. Для синьої складової (B=56) після ділення на обрані модулі формується набір залишків (0, 1, 4). Важливо відзначити, що кожен набір залишків унікально визначає вихідне значення в межах робочого

діапазону системи, який визначається добутком модулів 7×11×13=1001 та перевищує максимальне значення колірної складової 255.

В результаті один піксель зображення замість трьох байтів (R,G,B) представляється набором з дев'яти залишків: (2,9,6) для червоної, (2,7,11) для зеленої та (0,1,4) для синьої складової.

Другий метод – реалізація блокового методу шифрування. При реалізації блокового методу шифрування цифрове зображення спочатку розбивається на блоки фіксованого розміру, наприклад 8x8 або 16x16 пікселів. Кожен такий блок розглядається як єдине велике число, яке формується шляхом послідовного об'єднання значень всіх пікселів блоку [18]. Наприклад, блок розміром 8x8 пікселів у кольоровому RGB зображенні містить 192 байти інформації (8x8x3 колірні компоненти), які формують одне велике число для подальшого шифрування в СЗК. Перевага такого підходу полягає в тому, що зміна навіть одного біта в зашифрованому блоці призводить до повної зміни всього блоку при розшифруванні. На рисунку 1.4 продемонстрований процес блокового шифрування зображення за допомогою системи залишкових класів.

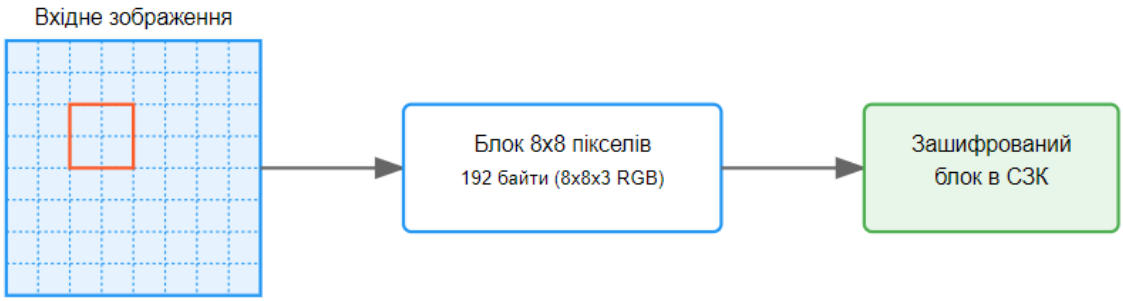

Рисунок 1.4 – Процес блокового шифрування

У випадку використання третього методу основна увага приділяється захисту не самого зображення, а ключових параметрів, які використовуються для його шифрування. Наприклад, зображення може шифруватися стандартним

алгоритмом AES або DES, але ключі для цих алгоритмів представляються та передаються в СЗК [19].

AES (Advanced Encryption Standard) або Rijndael є блочним шифром з симетричним ключем, який обробляє блоки розміром 128 біт та підтримує ключі довжиною 128, 192 або 256 біт. Цей алгоритм став переможцем конкурсу AES та був затверджений урядом США як офіційний стандарт шифрування. AES було обрано з урахуванням перспектив його широкого впровадження та ретельного криптоаналізу, подібно до його попередника DES [20].

DES ж в свою чергу - Data Encryption Standard представляє собою алгоритм симетричного шифрування, що обробляє 64-бітні блоки даних та використовує ключ довжиною 56 біт. Цей алгоритм був стандартом шифрування в США з 1977 по 2002 рік, доки його не замінив AES. DES був розроблений компанією IBM та прийнятий Національним бюро стандартів США як федеральний стандарт обробки інформації. Через відносно короткий розмір ключа (56 біт) та розвиток обчислювальної техніки, DES став вразливим до атак повного перебору, що і призвело до необхідності розробки нового стандарту шифрування [21]. Різниця між ними представлена в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика алгоритмів DES та AES

Характеристика	DES	AES
Розмір блоку	64 біт	128 біт
Довжина ключа	56 біт	128/192/256 біт
Кількість раундів	16	10/12/14
Рік впровадження	1977	2002
Криптостійкість	Низька через малий розмір ключа	Висока
Швидкість роботи	Нижча	Вища
Статус	Застарілий	Актуальний

Отож, підхід з шифруванням ключових параметрів у СЗК полягає в тому, що зображення обробляється сучасним алгоритмом AES, який виконує шифрування блоками по 128 біт через послідовність нелінійних перетворень. Ключі шифрування AES, довжиною 128, 192 або 256 біт, захищаються за допомогою СЗК, де кожен ключ розкладається на набір залишків за взаємно простими модулями. При цьому модулі СЗК обираються таким чином, щоб їх добуток перевищував максимальне значення ключа. Наприклад, для 128-бітного ключа AES можуть використовуватися модулі, добуток яких більший за 2^128 [22]. Така комбінація дозволяє поєднати швидкість блочного шифрування AES з надійним захистом ключів засобами СЗК, де кожен залишок може передаватися окремим захищеним каналом. На відміну від застарілого DES, який має обмежену довжину ключа 56 біт та вразливий до атак повного перебору, зв'язка AES-СЗК забезпечує надійний захист як самих даних, так і ключової інформації.

А для DES підхід з шифруванням ключових параметрів у СЗК базується на роботі з 56-бітним ключем. За допомогою СЗК цей ключ розкладається на набір залишків за взаємно простими модулями. Наприклад, для 56-бітного ключа DES можуть використовуватися модулі, добуток яких перевищує 2^56. При цьому сам алгоритм DES обробляє дані блоками по 64 біти через 16 раундів перетворень в мережі Фейстеля. Кожен раунд включає операції підстановки через S-блоки розміром 6x4 біти та перестановки бітів [23]. Хоча DES вважається застарілим через обмежену довжину ключа, використання СЗК для захисту ключової інформації може підвищити загальну криптостійкість системи, особливо коли залишки передаються різними захищеними каналами зв'язку, що ускладнює можливість відновлення оригінального ключа.

На рисунку 1.5 зображено порівняння шифрування ключів AES і DES в СЗК.

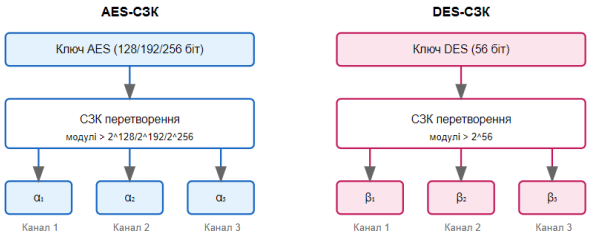

Рисунок 1.5 – Порівняння шифрування AES/DES в СЗК

В четвертий метод включають декілька методів, в основі яких лежить шифрування на основі RSA. Наприклад, метод шифрування Рабіна представляє математичну конструкцію асиметричної криптосистеми, розроблену Майклом Рабіном в 1979 році [24]. Основа методу базується на складності факторизації великих складених чисел, що робить його схожим з RSA, але з деякими істотними відмінностями в реалізації. Процес шифрування включає піднесення повідомлення до квадрату за модулем n, де n є добутком двох великих простих чисел p і q. При цьому дешифрування повідомлення вимагає обчислення квадратного кореня за модулем n, що можливо лише при знанні факторизації числа n [25].

Практична реалізація методу Рабіна починається з генерації ключів, де секретний ключ складається з двох простих чисел p і q, кожне з яких при діленні на 4 дає остачу 3. Публічний ключ формується як добуток цих чисел $n = p \times q$. При шифруванні повідомлення m обчислюється шифротекст $c = m^2 \ mod \ n$ [25]. Процес дешифрування використовує китайську теорему про залишки для знаходження чотирьох квадратних коренів з шифротексту. Для вирішення проблеми неоднозначності розшифровки використовуються різні методи, наприклад, додавання надлишкової інформації до повідомлення перед шифруванням або використання спеціальних форматів повідомлень.

Криптографічна стійкість методу Рабіна безпосередньо пов'язана з складністю факторизації числа n на прості множники p і q. Доведено, що повний злам криптосистеми Рабіна еквівалентний факторизації модуля n, що робить її теоретично більш безпечною, ніж RSA, для якого подібна еквівалентність не доведена. Рекомендована довжина ключа для забезпечення

достатнього рівня безпеки становить не менше 2048 біт. Швидкість шифрування в методі Рабіна досить висока, оскільки операція піднесення до квадрату вимагає менше обчислювальних ресурсів, ніж піднесення до степеня з великим показником, як у RSA [26].

Використання СЗК дозволяє розпаралелити процес обчислення квадратних коренів за модулями p і q, що значно прискорює процес дешифрування. Представлення чисел в СЗК також спрощує реалізацію китайської теореми про залишки, яка є ключовим елементом процесу дешифрування. Модульна арифметика в СЗК виконується паралельно за кожним модулем, що дозволяє ефективно обробляти великі числа. При цьому операції множення та піднесення до квадрату стають більш ефективними завдяки меншому розміру операндів. Модифікації методу Рабіна включають різні підходи до вирішення проблеми неоднозначності розшифровки. Один з підходів полягає у використанні спеціального формату повідомлень, де частина бітів резервується для контрольної інформації. Інший підхід передбачає додавання надлишкової інформації, такої як хеш-значення повідомлення або контрольна сума. Реалізація методу в реальних системах вимагає ретельного вибору параметрів для забезпечення балансу між безпекою та продуктивністю. Вибір простих чисел p і q повинен враховувати вимоги до їх розміру та властивості при діленні на 4.

Ще одним методом є метод Ель-Гамаля в СЗК. Сутність методу полягає в тому, що операції шифрування та розшифрування виконуються в залишках за набором взаємно простих модулів, що дозволяє розпаралелити обчислення та підвищити швидкодію системи. Використання СЗК дозволяє працювати з меншими за розміром числами, що значно прискорює модульні операції та робить систему більш ефективною для практичного застосування.

Математичний апарат криптосистеми Ель-Гамаля в СЗК включає операції в кільці за модулем p, де p є великим простим числом. Секретний ключ представляється числом x, а відкритий ключ формується як $y = g^x \bmod p$, де g є первісним коренем за модулем p. При шифруванні повідомлення m спочатку

генерується випадкове число k, потім обчислюються два числа: $a = g^k \bmod p$ та $b = y^k \times m \bmod p$. Пара чисел (a, b) становить шифротекст. Процес розшифрування відновлює початкове повідомлення за формулою $m = b \times (a^x)^{-1} \bmod p$.

Реалізація криптосистеми в СЗК починається з вибору набору взаємно простих модулів $p_1, p_2, ..., p_n$, добуток яких перевищує максимально можливе значення проміжних результатів. Кожне число в процесі обчислень представляється набором залишків за вибраними модулями. Модульні операції виконуються паралельно та незалежно за кожним модулем, що дозволяє ефективно використовувати багатопроцесорні системи та спеціалізовані обчислювальні пристрої. Відновлення результату з системи залишків виконується за допомогою китайської теореми про залишки.

1.3 Властивості та характеристики перетворень в системі залишкових класів

Система залишкових класів має декілька властивостей і характеристик, які варто було б опрацювати в ході роботи. Першою такою є властивість мультиплікативної інверсії в СЗК, яка дозволяє виконувати операції ділення та знаходження оберненого елемента. Для будь-якого числа в системі можна знайти таке інше число, яке при множенні на перше дасть одиницю за заданим модулем. Саме ця властивість робить можливим виконання операції ділення в СЗК, що критично важливо для багатьох практичних застосувань [27].

Мультиплікативна інверсія працює незалежно для кожного модуля в системі, що дозволяє виконувати обчислення паралельно. При цьому для кожного значення та кожного модуля існує свій унікальний обернений елемент. Наприклад, якщо взяти число та знайти його обернений елемент за одним модулем, потім за іншим, і так далі - отримаємо набір обернених елементів, які разом формують представлення оберненого числа в СЗК. Це дозволяє

ефективно виконувати ділення та інші операції, які потребують знаходження оберненого елемента.

Практичне значення мультиплікативної інверсії полягає в тому, що вона дозволяє реалізувати повний набір арифметичних операцій в СЗК. Без можливості знаходження оберненого елемента система була б обмежена лише додаванням, відніманням та множенням. Завдяки цій властивості можна виконувати будь-які математичні перетворення, що робить СЗК повноцінною системою для обчислень. При цьому всі операції виконуються без необхідності переведення чисел в звичайну форму запису.

Другою властивістю є повна незалежність залишків. Ця властивість означає, що операції над залишками за різними модулями виконуються повністю незалежно один від одного, без необхідності враховувати проміжні результати чи перенесення з інших позицій. Наприклад, коли ми додаємо два числа в СЗК, додавання відбувається окремо за кожним модулем, і результати за різними модулями ніяк не впливають один на одного. Незалежність залишків створює природну основу для паралельних обчислень у комп'ютерних системах [28-29]. Оскільки кожен залишок можна обробляти окремо, різні процесори чи ядра можуть одночасно виконувати операції над різними частинами числа. Такий підхід дозволяє значно прискорити обчислення, особливо при роботі з великими числами. При цьому не потрібно вирішувати проблеми синхронізації між процесорами, оскільки операції виконуються незалежно. Практично ця властивість проявляється в тому, що помилка в обчисленні одного залишку не впливає на інші залишки. Якщо в процесі передачі даних або обчислень виникає спотворення одного з залишків, інші залишки залишаються коректними.

Особливо цінною властивість незалежності залишків стає при обробці потоків даних. Можна розділити вхідний потік на кілька незалежних каналів, кожен з яких обробляє свій модуль. Це дозволяє ефективно розподіляти навантаження між доступними обчислювальними ресурсами та досягати

високої пропускної здатності системи. При цьому затримки в обробці одного каналу не впливають на роботу інших каналів.

Третьою властивістю системи залишкових класів є відсутність розповсюдження помилок при обчисленнях. На відміну від традиційних позиційних систем числення, де помилка в одному розряді може призвести до спотворення всіх наступних розрядів через механізм переносу, в СЗК кожен залишок зберігає свою достовірність незалежно від стану інших залишків. На рисунку 1.6 представлено порівняння розповсюдження помилок в традиційній та СЗК системах.

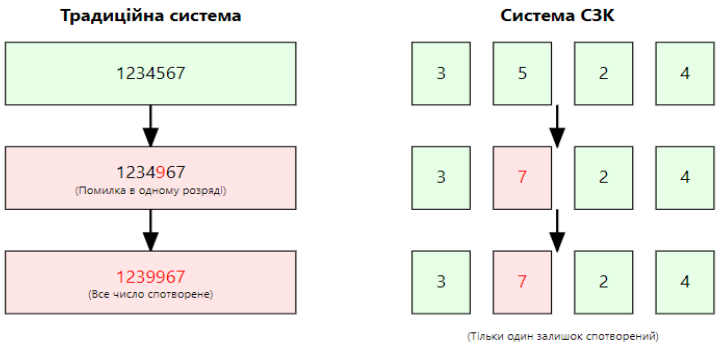

Рисунок 1.6 – Порівняння розповсюдження помилок в традиційній та СЗК системах

На рисунку показано як помилка в одному розряді традиційної системи числення спотворює все число через механізм переносу, тоді як в СЗК помилка залишається локалізованою тільки в одному залишку, не впливаючи на інші. Зелений колір позначає коректні значення, червоний - спотворені. В традиційній системі помилка в одному розряді призводить до повного спотворення числа, тоді як в СЗК спотворюється лише один залишок, а інші залишаються коректними.

Ще однією властивістю СЗК є природне розпаралелювання обчислень в СЗК, яка базується на принципі незалежності операцій за різними модулями. Така властивість створює ідеальні умови для паралельних обчислень в сучасних комп'ютерних системах. Кожен процесор або ядро може працювати зі своїм модулем, виконуючи обчислення незалежно від інших. При цьому не

виникає потреби в складних механізмах синхронізації чи обміну даними між процесорами, що часто є вузьким місцем в паралельних системах [30-31]. Всі інші властивості представлені в таблиці 1.2.

Таблиця 1.2 – Додаткові властивості СЗК

Властивість	Опис	Область застосування в шифруванні
Однозначність представлення	Кожне число має єдине представлення	Гарантує унікальність шифротексту
Симетричність операцій	Однакова складність операцій незалежно від значень операндів	Захист від часових атак через стабільний час виконання операцій
Відсутність округлення	Відсутність накопичення помилок округлення в проміжних результатах	Забезпечує точність криптографічних обчислень та стійкість алгоритмів
Динамічна точність	Можливість зміни точності через додавання/видалення модулів	Адаптація рівня захисту та складності криптосистеми під конкретні вимоги
Модульні операції	Природна підтримка операцій за модулем	Ефективна реалізація модульної арифметики в криптографічних алгоритмах
Пряме порівняння	Можливість порівняння чисел без повного відновлення значень	Оптимізація операцій порівняння в криптографічних протоколах
Стійкість до витоку	Неможливість відновлення по	Додатковий рівень захисту при частковому компрометуванні даних

Крім властивостей СЗК, також присутні і певні характеристики. Основними характеристиками системи залишкових класів є набір ключових параметрів, що визначають її функціональні можливості та ефективність. Робочий діапазон системи визначається добутком всіх її модулів та показує максимальне значення чисел, які можна представити. Цей параметр безпосередньо впливає на точність обчислень та можливості системи щодо обробки даних різного масштабу. Більший робочий діапазон дозволяє працювати з більшими числами, але вимагає більше обчислювальних ресурсів. Динамічний діапазон характеризує відношення максимального значення до мінімального, яке може бути представлене в системі. Цей параметр важливий для оцінки можливостей системи обробляти дані різних порядків величини. Широкий динамічний діапазон дозволяє ефективно працювати як з дуже малими, так і з дуже великими числами в рамках однієї системи [32].

Надлишковість системи визначається кількістю додаткової інформації, що вноситься за рахунок використання додаткових модулів. Цей параметр критично важливий для забезпечення надійності системи, оскільки дозволяє виявляти та виправляти помилки, що виникають при обчисленнях або передачі даних. Більша надлишковість підвищує надійність системи, але збільшує обсяг даних та обчислювальні витрати. Інформаційна ємність характеризує кількість інформації, яку можна закодувати в системі при заданому наборі модулів. Цей параметр визначає ефективність використання пам'яті та каналів зв'язку. Висока інформаційна ємність дозволяє компактно представляти дані, але може вимагати більш складних алгоритмів обробки.

Складність обчислень визначається кількістю операцій, необхідних для виконання базових арифметичних дій. Цей параметр впливає на швидкодію системи та вимоги до обчислювальних ресурсів. Оптимізація складності обчислень дозволяє підвищити ефективність системи, але може вимагати компромісів з іншими характеристиками. Рівень паралелізму показує здатність системи виконувати операції одночасно над різними залишками. Ця характеристика визначає можливості масштабування системи та ефективність

використання багатопроцесорних систем. Високий рівень паралелізму дозволяє значно прискорити обчислення, але вимагає відповідної апаратної підтримки [33].

Відмовостійкість системи характеризує її здатність продовжувати функціонування при виникненні помилок або відмов окремих компонентів. Ця характеристика важлива для критичних застосувань, де надійність є пріоритетом. Висока відмовостійкість досягається за рахунок надлишковості та спеціальних алгоритмів обробки помилок.

Масштабованість системи визначає можливості її розширення та адаптації до зростаючих вимог. Ця характеристика важлива для систем, які повинні розвиватися та змінюватися з часом. Хороша масштабованість дозволяє легко додавати нові модулі та збільшувати обчислювальну потужність системи. Точність обчислень характеризує здатність системи зберігати правильність результатів при виконанні послідовності операцій. Ця характеристика критична для наукових та інженерних застосувань, де важлива висока точність результатів. Забезпечення високої точності може вимагати використання більших модулів та додаткових механізмів контролю.

Ефективність використання ресурсів показує, наскільки оптимально система використовує доступні обчислювальні та апаратні ресурси. Ця характеристика важлива для оптимізації витрат на реалізацію та експлуатацію системи. Висока ефективність досягається через оптимальний вибір параметрів системи та алгоритмів обробки даних [34].

Висновки до розділу 1

Проаналізовано основи системи залишкових класів, включаючи принципи представлення чисел у вигляді набору залишків від ділення на взаємно прості модулі, механізми виконання арифметичних операцій та особливості відновлення чисел з їх залишкового представлення.

Досліджено методи шифрування даних з використанням СЗК, включаючи різні підходи до формування криптосистем на основі властивостей модульної арифметики.

Оглянуто ключові характеристики та властивості перетворень в системі залишкових класів, що включають мультиплікативну інверсію, незалежність залишків, відсутність розповсюдження помилок, симетричність операцій та інші важливі аспекти. Проаналізовано практичне значення цих властивостей для реалізації ефективних та надійних систем обробки даних.

2 ДОСЛІДЖЕННЯ АЛГОРИТМІВ ШИФРУВАННЯ ЗОБРАЖЕНЬ В СЗК

2.1. Математична модель представлення зображень в СЗК

Для представлення числа X в системі залишкових класів обирається набір взаємно простих модулів p_1, p_2, ..., p_n. Робочий діапазон системи визначається як:

$$P = p_1 \times p_2 \times \ldots \times p_n \qquad (2.1)$$

де P - робочий діапазон системи, а власне можливих значень, $p_1, p_2, \ldots, p_n$ - взаємно прості модулі (прості числа або взаємно прості числа), n - кількість модулів в системі. Наприклад, при $p_1 = 2, p_2 = 3, p_3 = 5, n = 3: P = 2 \times 3 \times 5 = 30$. при $p_1 = 7, p_2 = 11, p_3 = 13, n = 3: P = 7 \times 11 \times 13 = 1001$, а при $p_1 = 3, p_2 = 5, p_3 = 7, p_4 = 11, n = 4: P = 3 \times 5 \times 7 \times 11 = 1155$.

Ці модулі повинні бути взаємно простими числами, тобто їх найбільший спільний дільник повинен дорівнювати 1, тобто НСД(p_i, p_j) = 1 для всіх $i \neq j$. Якщо детальніше розглядати НСД. Найбільший спільний дільник (НСД) двох або більше цілих чисел – це найбільше ціле число, на яке всі ці числа діляться без остачі. Наприклад, для чисел 12 і 18 спільними дільниками є числа 1, 2, 3, 6, а найбільшим з них є 6, тому НСД(12,18) = 6.

В системі залишкових класів модулі повинні бути взаємно простими числами, тобто їх НСД повинен дорівнювати 1. Розглянемо приклад знаходження НСД для різних пар модулів, який зображений на рисунку 2.1.

За основу для приклада взятий приклад, де $p_1 = 7, p_2 = 11, p_3 = 13$. Рисунок 2.1 доказує, що ці числа є взаємно простими, що підтверджує їх придатність для використання як модулів в СЗК. Аналогічно можна перевірити взаємну простоту для будь-якого іншого набору модулів.

НСД(7,11)	НСД(11,13)	НСД(7,13)
11 = 1×7 + 4	13 = 1×11 + 2	13 = 1×7 + 6
7 = 1×4 + 3	11 = 5×2 + 1	7 = 1×6 + 1
4 = 1×3 + 1	2 = 2×1 + 0	6 = 6×1 + 0
3 = 3×1 + 0	НСД = 1	НСД = 1
НСД = 1		

Рисунок 2.1 – Знаходження НСД для чисел 7, 11, 13

А число X в системі залишкових класів представляється набором найменших невід'ємних залишків за формулою:

$$X \leftrightarrow (x_1, x_2, \ldots, x_n) \tag{2.2}$$

де $x_i \equiv X(mod\ p_i)$, тобто x_i - залишок від ділення X на p_i. Якщо розглядати на прикладі, зазначеному вище, то нехай X = 100. Тоді 100 = 14 × 7 + 2, власне $x_1 = 2$, $x_2 = 1$, $x_3 = 9$. Отже, число 100 в СЗК представляється набором залишків $100 \leftrightarrow (2, 1, 9)$.

Відносно КТЗ – за китайською теоремою про залишки, для взаємно простих модулів існує єдине рішення системи порівнянь:

$$X \equiv x_n(mod p_n) \tag{2.3}$$

де X представляє собою шукане число в системі залишкових класів, x_i є залишком від ділення числа X на відповідний модуль, p_i позначає взаємно прості модулі системи, mod вказує на операцію знаходження залишку від ділення, n визначає кількість модулів в системі, а i є індексом поточного модуля, що змінюється від 1 до n.

Рішення цієї системи порівнянь гарантовано існує в діапазоні від 0 до P-1, де P обчислюється як добуток всіх модулів системи зазначених в формулі 1. Відновлення числа X за його залишками виконується за формулою, де $M_i = P/p_i$ - ортогональні базиси, M_i^{-1} - мультиплікативно обернені елементи до M_i за модулем p_i

$$X = (\sum(x_i M_i M_i^{-1}) mod P) \tag{2.4}$$

Тут x_i є залишками від ділення числа на відповідні модулі. M_i називається ортогональним базисом і обчислюється як P/p_i, тобто добуток всіх модулів, поділений на поточний модуль. M_i^{-1} є мультиплікативно оберненим елементом до M_i за модулем p_i, тобто таким числом, що задовольняє умову $M_i M_i^{-1} \equiv 1 \pmod{p_i}$. Процес обчислення виглядає так: спочатку для кожного модуля знаходиться його ортогональний базис M_i шляхом ділення загального добутку модулів P на поточний модуль p_i. Потім для кожного ортогонального базису знаходиться його обернений елемент за відповідним модулем. Далі кожен залишок множиться на свій ортогональний базис та його обернений елемент. Всі отримані добутки додаються, і від суми береться залишок за модулем P.

Ця формула гарантує, що отримане число X буде давати потрібні залишки при діленні на відповідні модулі системи. Вона є практичним втіленням китайської теореми про залишки та дозволяє виконувати зворотне перетворення з системи залишкових класів у звичайне десяткове представлення числа.

Також в СЗК існують арифметичні операції, які виконуються покомпонентно. Тобто, існують операції додавання, віднімання, множення і окремо ділення через мультиплікативно обернений елемент.

Операція додавання має вигляд:

$$(x_1, x_2, \ldots, x_n) + (y_1, y_2, \ldots, y_n) = \big((x_1 + y_1) mod p_1, \ldots, (x_n + y_n) mod p_n\big) \quad (2.5)$$

де x_i та y_i - це залишки від ділення чисел на відповідний модуль p_i, p_i - взаємно прості модулі (тобто їх НСД = 1), mod p_i означає, що береться остача від ділення суми (x_i+y_i) на модуль p_i.

Додавання в системі залишкових класів має кілька важливих особливостей. Насамперед, воно виконується незалежно по кожному розряду або модулю, при цьому в кожному розряді додаються відповідні залишки. Важливо, що результат у кожному розряді обов'язково береться по модулю відповідного p_i. Таке додавання має суттєві переваги. Завдяки тому, що обчислення по кожному модулю відбуваються незалежно, їх можна виконувати

паралельно, що значно підвищує швидкість операцій. На відміну від звичайної системи числення, тут немає переносів між розрядами, що спрощує процес обчислень. Крім того, проміжні результати мають обмежену розрядність, яка не перевищує відповідний модуль.

Операція віднімання має такий вигляд:

$$(x_1, x_2, \ldots, x_n) - (y_1, y_2, \ldots, y_n) = \big((x_1 - y_1) mod p_1, \ldots, (x_n - y_n) mod p_n\big) \quad (2.6)$$

Віднімання в системі залишкових класів виконується подібно до додавання, але з деякими особливостями. Операція відбувається незалежно для кожного розряду, де від залишку зменшуваного віднімається залишок від'ємника, а результат береться по модулю відповідного числа p_i. Як і при додаванні, можна паралельно виконувати віднімання по різних модулях, що суттєво прискорює процес. Відсутність міжрозрядних переносів спрощує обчислення порівняно зі звичайною системою числення. Важливою перевагою є те, що розрядність проміжних результатів обмежена відповідним модулем, що запобігає переповненню розрядної сітки.

Множення має наступну формулу:

$$(x_1, x_2, \ldots, x_n) \times (y_1, y_2, \ldots, y_n) = \big((x_1 \times y_1) mod p_1, \ldots, (x_n \times y_n) mod p_n\big) \quad (2.7)$$

Множення в системі залишкових класів виконується через поелементне множення залишків у кожному розряді. При цьому добуток кожної пари чисел береться по відповідному модулю, що забезпечує утримання результату в межах допустимих значень для кожного розряду. Як і з іншими операціями в СЗК, множення можна виконувати паралельно по всіх розрядах, що значно підвищує швидкодію. Важливою особливістю є відсутність переносів між розрядами, які зазвичай ускладнюють множення у традиційних системах числення. Проміжні результати в кожному розряді обмежені відповідним модулем, що запобігає переповненню розрядної сітки та спрощує апаратну реалізацію.

Трохи інакше виглядає ділення, оскільки для нього необхідно знайти мультиплікативно обернений елемент, оскільки пряме ділення в модульній арифметиці неможливе. Мультиплікативно обернений елемент y^{-1}_i - це таке число, яке при множенні на вихідне число y_i дає 1 за модулем p_i.

$$y^{-1}_i \equiv 1(mod p_i) \tag{2.8}$$

Також у СЗК існує контроль помилок, який здійснюється через додавання надлишкових модулів $p_{n+1}, p_{n+2}, \ldots,$ що дозволяє виявляти та виправляти помилки в залишках. І власне для порівняння чисел використовується інтервальна індексація на основі рангу числа.

Інтервальна індексація на основі рангу числа - це метод порівняння чисел у системі залишкових класів (СЗК). Це необхідно тому, що в СЗК числа представлені набором залишків, і напряму порівняти їх складно - неможливо просто глянути і сказати, яке число більше [35]. Ранг числа вираховується за формулою нижче:

$$rang(X) = \left[\frac{X}{p}_1\right] + \left[\frac{X}{p}_2\right] + \ldots + \left[\frac{X}{p}_n\right] \tag{2.9}$$

де $[X/p_i]$ означає цілочисельне ділення X на p_i (без остачі). При порівнянні двох чисел A і B у системі залишкових класів використовується їх ранг. Коли ранг числа A більший за ранг числа B, це означає, що A більше за B. Відповідно, якщо ранг A менший за ранг B, то A менше за B. У випадку рівності рангів необхідно проводити додаткові обчислення для точного визначення співвідношення між числами. Така система дозволяє виконувати порівняння чисел безпосередньо в системі залишкових класів, визначати знак числа, виявляти випадки переповнення при обчисленнях та знаходити інтервал, в якому розташоване число.

2.2. Алгоритм шифрування зображень з використанням СЗК

На рисунку 2.2 представлена схема роботи алгоритму шифрування зображень з використанням СЗК.

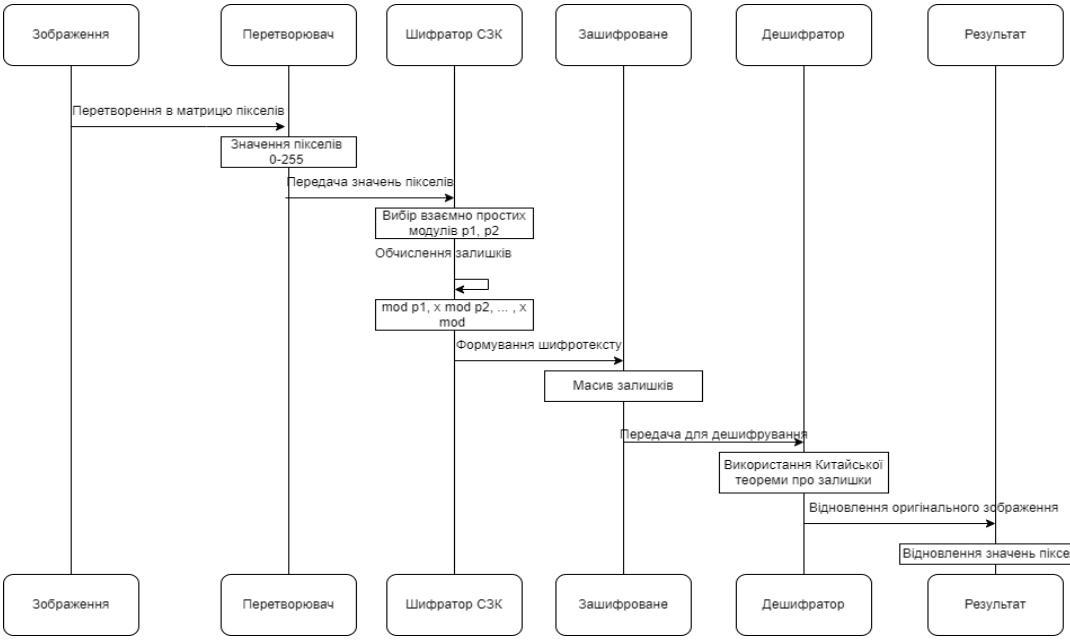

Рисунок 2.2 – Схема роботи алгоритму шифрування з використанням СЗК

На першому етапі шифрування зображень з використанням системи залишкових класів відбувається перетворення вхідного зображення в цифрову форму.

Цей процес є основним для подальшої обробки та шифрування, оскільки саме він забезпечує представлення даних у форматі, придатному для застосування математичних операцій системи залишкових класів.

Початкове зображення, незалежно від його формату (PNG, JPEG, BMP тощо), перетворюється в матрицю пікселів. Кожен піксель такої матриці представляється числовим значенням, яке відповідає його яскравості або кольоровим характеристикам.

Для стандартного 8-бітного зображення ці значення знаходяться в діапазоні від 0 до 255, де 0 зазвичай представляє чорний колір, а 255 - білий для зображень у відтінках сірого. Для кольорових зображень кожен піксель

представляється трьома значеннями, що відповідають інтенсивності червоного, зеленого та синього кольорів (RGB модель).

Процес цифрового перетворення включає в себе аналіз кожного пікселя зображення та створення відповідної числової матриці.

Для зображення розміром M×N створюється матриця такого ж розміру, де кожен елемент містить числове значення відповідного пікселя.

У випадку кольорового зображення створюються три окремі матриці - по одній для кожного кольорового каналу. Ця операція є критично важливою, оскільки точність перетворення безпосередньо впливає на якість подальшого шифрування та можливість правильного відновлення зображення після дешифрування.

Також на цьому етапі може проводитися попередня обробка даних, наприклад, нормалізація значень пікселів або застосування фільтрів для покращення якості зображення.

Наприклад, при роботі з прозорістю (альфа-канал) або специфічними форматами зображень можуть знадобитися додаткові етапи перетворення.

Також важливо враховувати особливості колірних просторів та можливі втрати якості при перетвореннях між ними.

Результатом ж першого етапу є повністю цифрове представлення зображення у вигляді матриці чисел, готової для подальшої обробки в системі залишкових класів.

Матриця пікселів представлена на рисунку 2.3, а точніше її частина розміром 10 на 10 пікселів, як для прикладу – для візуалізації даних.

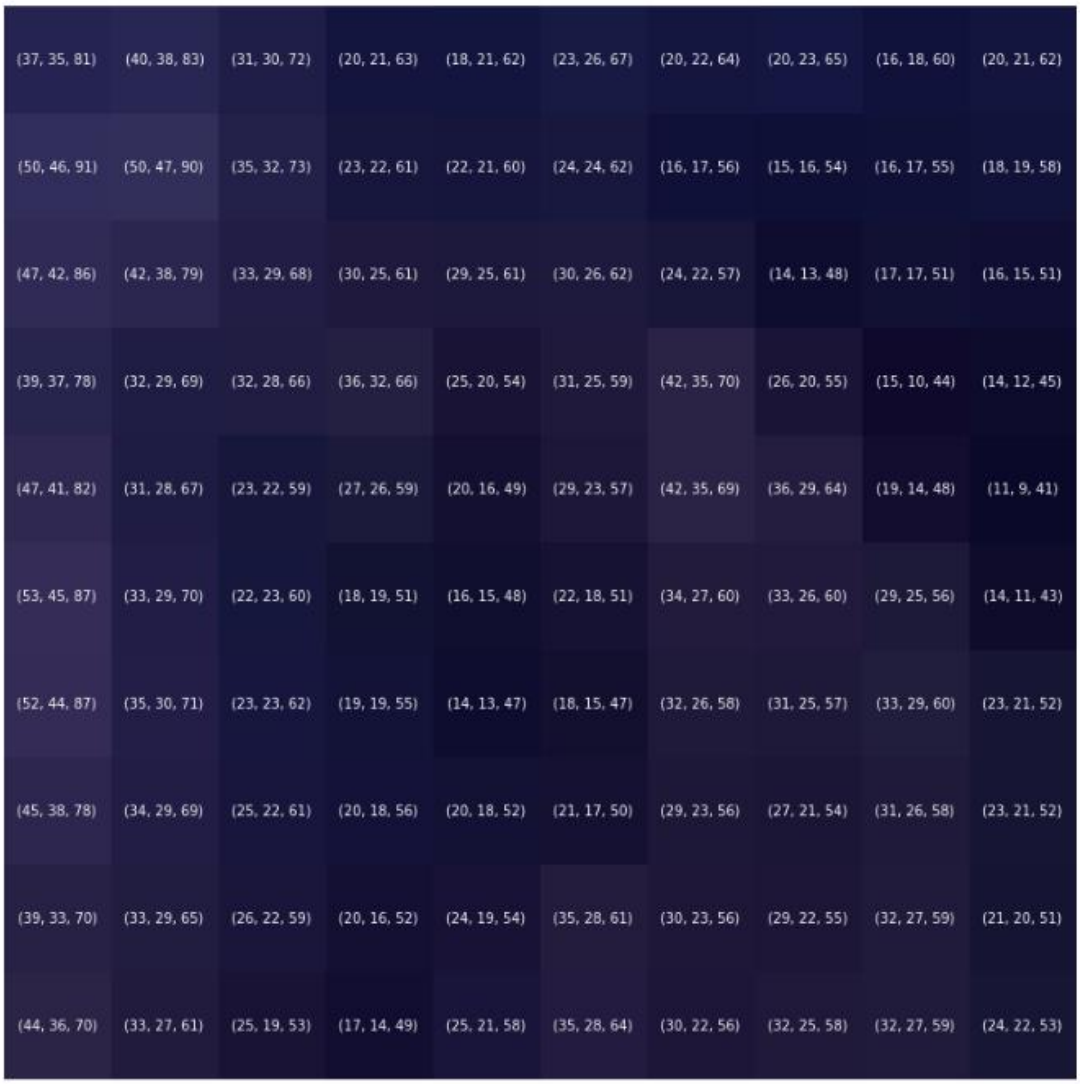

Рисунок 2.3 – Матриця пікселів 10 на 10 з рисунку 2.4

Звісно, це візуалізована матриця. В реальності вона буде мати запис у вигляді *[(255, 0, 0), (0, 255, 0), ..., (128, 128, 128), (255, 255, 0)], [(0, 255, 255), (255, 0, 255), ..., (0, 0, 0), (255, 255, 255)]*. Але це передбачає виконання першого етапу схеми роботи алгоритму, представленого на рисунку 2.1. Відповідно на рисунку 2.4 зображено картину, з якої і було взято матрицю елементів кольорів, але для прикладу, оскільки в масштабі навіть 300 х 300 пікселів буде надто важким і нечитабельним, було взято лише результати 10 х 10 пікселів. Також, червоним квадратом позначену область, яка була зображена на рисунку 2.3.

Рисунок 2.4 – Картина, з якої було взято матрицю елементів RGB

Другий етап роботи алгоритму шифрування зображень з використанням СЗК полягає у виборі системи взаємно простих модулів. Модулі $p_1, p_2, ..., p_n$ - це взаємно прості числа, тобто їх найбільший спільний дільник дорівнює 1. Добуток цих модулів $P = p_1 \times p_2 \times ... \times p_n$ має перевищувати максимальне значення пікселя у вхідному зображенні (для 8-бітного зображення - 255).

Вибрані модулі використовуються як для шифрування, так і для дешифрування зображення, тому вони мають забезпечувати можливість однозначного відновлення оригінальних значень за допомогою Китайської теореми про залишки.

Третім етапом у схемі є обчислення залишків для кожного значення пікселя. На рисунку 2.5 зображено схему алгоритму обробки пікселів.

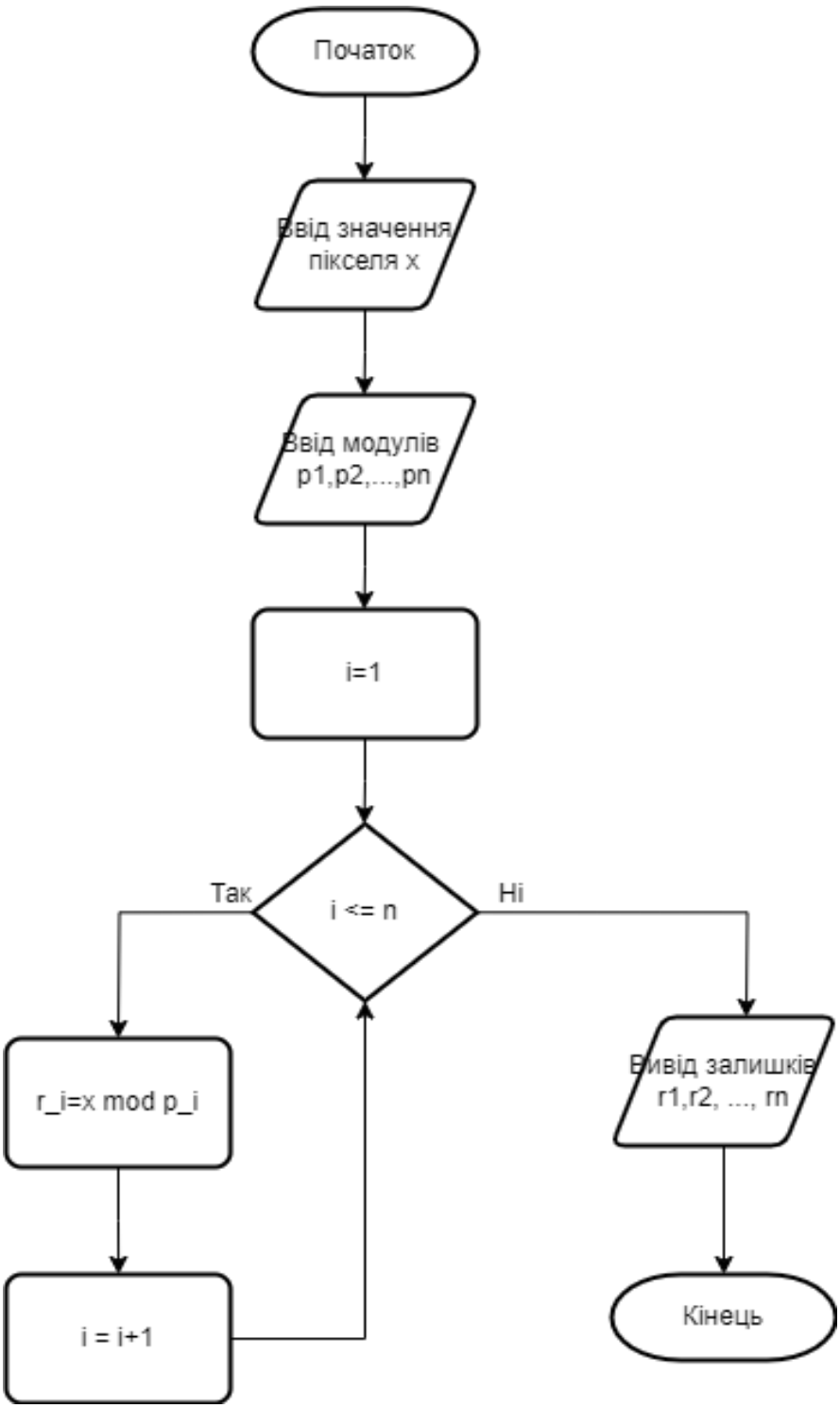

Рисунок 2.5 – Алгоритм обчислення СЗК пікселів

На початку алгоритму ми отримуємо вхідне значення пікселя x, для якого будуть обчислюватися залишки. Також отримуємо набір взаємно простих модулів $p_1, p_2, ..., p_n,$ які будуть використовуватися для обчислення.

Після отримання вхідних даних встановлюємо лічильник i в початкове значення 1. Цей лічильник буде використовуватися для проходження по всіх модулях та обчислення відповідних залишків.

Далі починається цикл, який працює поки значення лічильника i не перевищить кількість модулів n. У цьому циклі для поточного модуля p_i обчислюється залишок від ділення значення пікселя x на цей модуль. Отриманий залишок зберігається як r_i.

Після обчислення залишку для поточного модуля, значення лічильника i збільшується на 1, і цикл повторюється для наступного модуля. Цей процес продовжується доки не будуть обчислені залишки для всіх модулів.

Коли залишки обчислені для всіх модулів, тобто коли i стає більшим за n, цикл завершується. На виході алгоритму ми отримуємо набір залишків $r_1, r_2, ..., r_n$, які представляють початкове значення пікселя x в системі залишкових класів.

Після виведення отриманих залишків алгоритм завершує свою роботу. Отриманий набір залишків може бути використаний для подальшої обробки або зберігання зашифрованого значення пікселя.

Четвертим етапом в загальній схемі роботи алгоритму є формування шифротексту. На цьому етапі відбувається формування шифротексту через створення масиву залишків. Для цього всі отримані залишки від ділення значень пікселів на відповідні модулі збираються в єдину структуру даних.

Для кожного пікселя початкового зображення створюється набір відповідних залишків, які групуються в масив. Якщо початкове зображення мало розмір M×N пікселів, то для кожного з цих M×N пікселів ми маємо набір з n залишків, де n - кількість використаних модулів.

Таким чином формується тривимірний масив розміром M×N×n, де перші дві координати відповідають положенню пікселя в початковому зображенні, а

третя координата вказує на конкретний залишок для відповідного модуля. На рисунку 2.6 зображено отримання шифротексту на прикладі зображення з 4 значеннями.

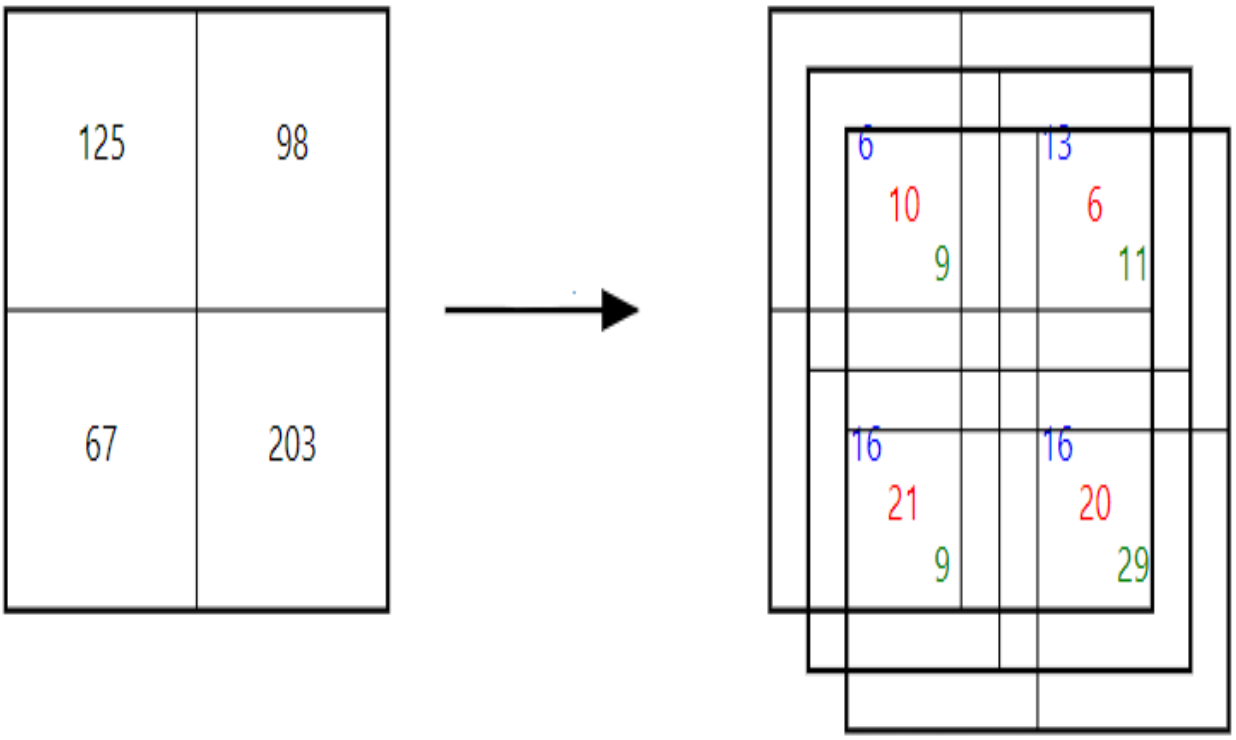

Рисунок 2.6 – Приклад створення шифротексту на основі картинки 2x2 пікселі

На п'ятому етапі відбувається передача зашифрованого зображення у вигляді тривимірного масиву даних. На відміну від стеганографії, де приховані дані вбудовуються в існуюче зображення, тут відбувається повна заміна оригінальних значень пікселів на масив залишків. В результаті, якщо спробувати відкрити таке зашифроване зображення звичайним переглядачем, воно буде виглядати як спотворені пікселі, оскільки кожне оригінальне значення пікселя замінено на набір залишків. Модулі виступають у ролі ключів для розшифрування, тому їх передача має бути захищеною, якщо вони не були узгоджені між сторонами заздалегідь, але про це в розділі зворотного перетворення з СЗК.

2.3. Алгоритм зворотного перетворення з СЗК

На рисунку 2.7 зображено приклад того, як після перетворення виглядає картинка.

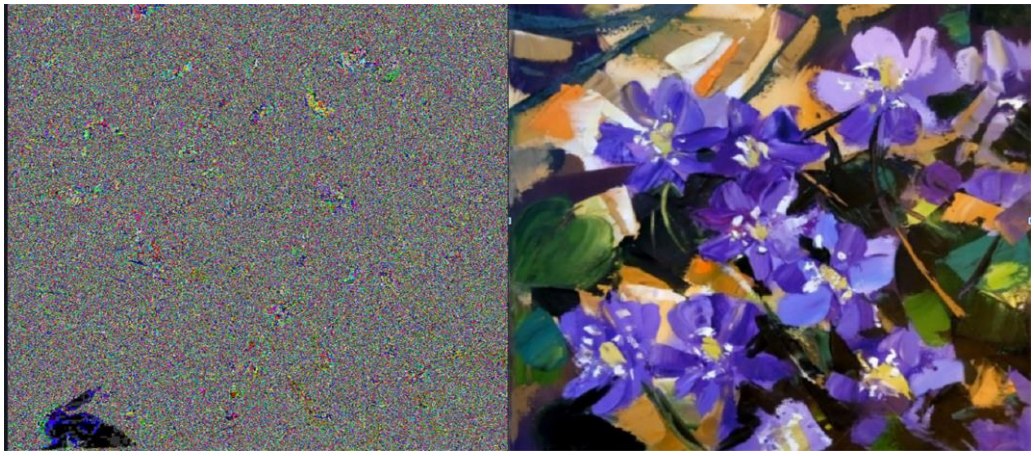

Рисунок 2.7 – Порівняння оригіналу із заміненим за допомогою СЗК

Для розшифрування такої картинки, а точніше перетворення матриць потрібно скористатись Китайською теоремою про залишки. Цей етап перетворює трьовимірний масив назад у двовимірну матрицю оригінальних значень.

Першим етапом буде підготовий – отримання всіх необхідних елементів для дешифрування, перевірка необхідних параметрів: розміру, модулів, обчислення допоміжних значень за допомогою НСК:

Спочатку перевіряються вхідні дані - тривимірний масив залишків розміром M×N×n, де M та N - це розміри оригінального зображення, а n - кількість модулів. Перевіряється коректність розмірів масиву та наявність всіх необхідних значень.

Далі відбувається перевірка модулів p_1, p_2, ..., p_n. Вони мають бути взаємно простими числами, тому перевіряється їх взаємна простота через обчислення НСД для кожної пари модулів. НСД має дорівнювати 1 для всіх пар.

Ці значення будуть використовуватись в подальшому для схеми роботи, зображеної на рисунку 2.8

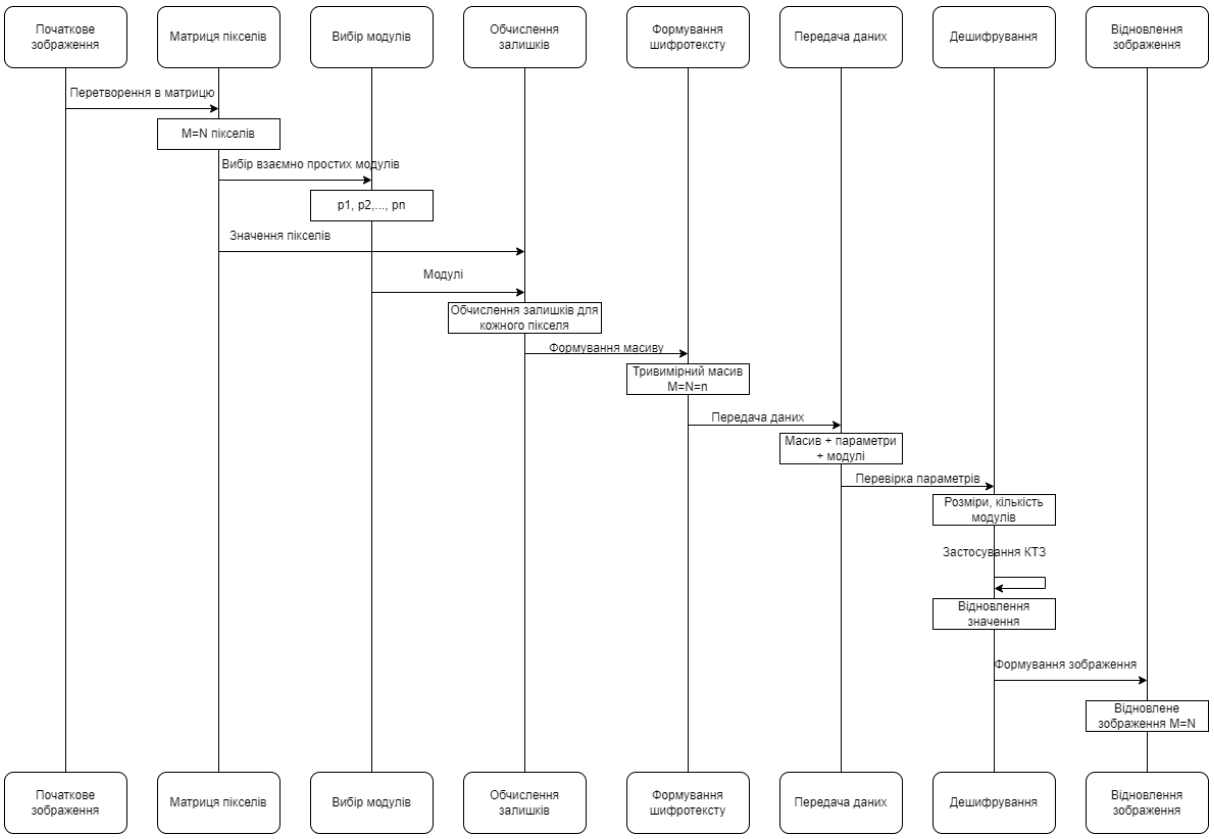

Рисунок 2.8 – Схема роботи алгоритму відновлення початкового зображення

Після підготовчого етапу відбувається безпосереднє відновлення значень пікселів з використанням Китайської теореми про залишки.

Для кожного пікселя береться його набір залишків та виконується обчислення за формулою $x = (r_1M_1N_1 + r_2M_2N_2 + ... + r_nM_nN_n) \bmod P$. У цій формулі r_i представляють залишки конкретного пікселя, M_i - це частки від ділення загального модуля P на відповідний модуль p_i, а N_i - мультиплікативно обернені елементи, знайдені на попередньому етапі.

Процес відновлення виконується послідовно для кожного пікселя зашифрованого зображення. Спочатку беруться залишки цього пікселя з тривимірного масиву.

Потім для кожного залишку виконується множення на відповідні константи M_i та N_i. Отримані добутки додаються, і результат береться по модулю P. Отримане значення і є оригінальним значенням пікселя.

Цей процес повторюється для всіх пікселів зображення, поки не буде відновлено всі значення. При цьому обчислення для різних пікселів можуть виконуватися паралельно, оскільки вони незалежні одне від одного. Результатом цього етапу є набір відновлених значень пікселів, які відповідають оригінальному зображенню.

На останньому етапі алгоритму відбувається формування відновленого зображення з отриманих значень пікселів. Спочатку створюється порожня матриця розміром M×N, яка відповідає розмірам оригінального зображення.

Ця матриця поступово заповнюється відновленими значеннями пікселів, які були обчислені на попередньому етапі. Для кожного відновленого значення проводиться перевірка на коректність - воно має потрапляти в допустимий діапазон значень пікселів, який для 8-бітного зображення становить від 0 до 255. Якщо значення виходить за межі цього діапазону, це свідчить про помилку в процесі відновлення або передачі даних.

Після заповнення всієї матриці значеннями та перевірки їх коректності, матриця перетворюється у формат зображення.

При цьому кожне число в матриці стає значенням яскравості відповідного пікселя. Для кольорових зображень цей процес виконується окремо для кожного кольорового каналу. Відповідно це і є звичайним алгоритмом шифрування і дешифування зображень за допомого СЗК,проте для кращого захисту варто використовувати додаткові елементи захисту.

2.4. Оптимізація складності алгоритмів

Окрім застосування інших типів шифрування, які будуть описані як ускладнення пізніше, потрібно виділити приближені до СЗК ідеї по ускладненню алгоритму шифрування даних.

Першим таким методом є використання більшої кількості модулів. При використанні 3-4 модулів в системі залишкових класів для шифрування зображень, процес обчислення є відносно простим. Наприклад, для значення пікселя 180 і модулів $p_1 = 17$, $p_2 = 23$, $p_3 = 29$ потрібно виконати лише три операції ділення з остачею для отримання залишків. Обсяг обчислень і розмір шифротексту при цьому залишаються помірними.

При збільшенні кількості модулів до 6-8, складність значно зростає. Тепер для того ж значення пікселя 180 потрібно обчислити вже 6-8 залишків, використовуючи більші за значенням модулі. Наприклад, для модулів $p_1 = 17$, $p_2 = 23$, $p_3 = 29$, $p_4 = 31$, $p_5 = 37$, $p_6 = 41$ кількість операцій подвоюється. Крім того, більші значення модулів призводять до складніших обчислень при знаходженні залишків.

Звісно, швидкість шифрування падає, але зростає вдвічі, фактично, криптостійкість.

Другий метод також в собі включає роботу з модулями. Динамічна зміна модулів представляє собою метод підвищення криптостійкості шифрування в системі залишкових класів шляхом зміни набору використовуваних модулів під час процесу шифрування. Замість використання фіксованого набору модулів для всього зображення, модулі змінюються за певним алгоритмом або правилом.

Наприклад, можна розділити зображення на блоки і для кожного блоку використовувати свій набір модулів. Ці набори можуть генеруватися на основі певного ключа або послідовності. При цьому правило зміни модулів стає додатковим секретним параметром, без знання якого неможливо правильно дешифрувати зображення, навіть маючи доступ до деяких використаних модулів.

Також можлива реалізація, де модулі змінюються для кожного рядка або стовпця зображення, або навіть для окремих пікселів за певним патерном. При цьому важливо забезпечити, щоб всі набори модулів залишались взаємно простими числами і їх добуток перевищував максимальне значення пікселя. На

рисунку 2.9 представлено алгоритм динамічної заміни модулів. Його суть полягає в наступному:

При виконанні алгоритму динамічної зміни модулів спочатку відбувається введення вхідного зображення, яке потім розділяється на блоки для подальшої обробки. Розпочинається цикл обробки з першого блоку, де для кожного блоку перевіряється умова продовження циклу - чи не перевищує поточний номер блоку їх загальну кількість.

Якщо умова виконується, то для поточного блоку генерується новий набір модулів. Після генерації обов'язково виконується перевірка на взаємну простоту згенерованих модулів. У випадку якщо модулі не є взаємно простими, відбувається повторна генерація нового набору. Коли отримано набір взаємно простих модулів, виконується шифрування поточного блоку з використанням цих модулів.

Інформація про використані модулі зберігається для подальшого дешифрування, після чого відбувається перехід до наступного блоку через збільшення лічильника на одиницю. Цей процес повторюється для всіх блоків зображення.

Після завершення обробки всіх блоків формується загальний шифротекст, який включає зашифровані дані всіх блоків. На завершальному етапі зберігаються параметри зміни модулів, які будуть необхідні для коректного дешифрування зображення. Ці параметри включають інформацію про те, які модулі використовувались для кожного блоку та в якому порядку вони змінювались.

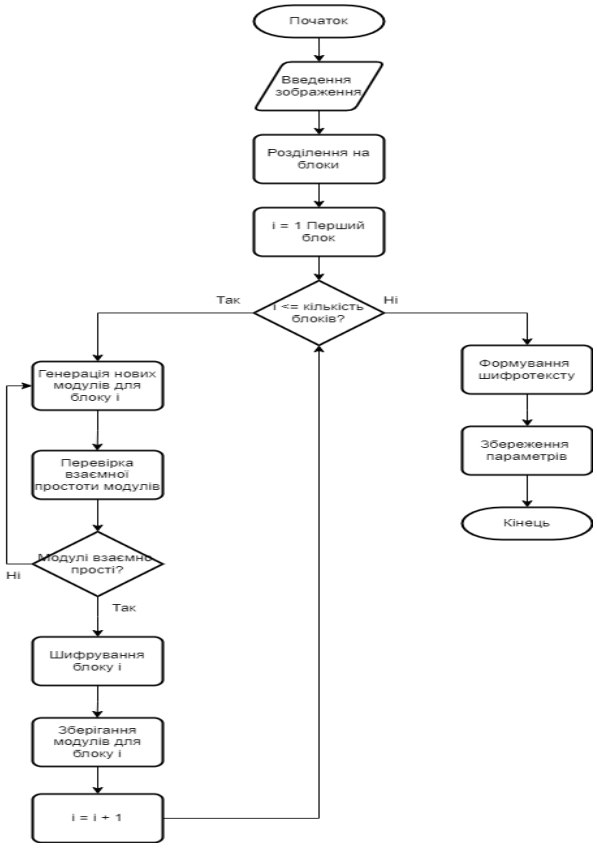

Рисунок 2.9 – Схема алгоритму динамічної заміни модулів

Також ускладнити алгоритм шифрування можна методом додавання надлишкових модулів. Він полягає в використанні додаткових модулів поверх необхідного мінімуму для заплутування процесу дешифрування. До основного набору взаємно простих модулів, необхідних для коректного шифрування, додаються додаткові модулі-обманки. При шифруванні використовуються як справжні, так і надлишкові модулі. Для кожного значення пікселя обчислюються залишки за всіма модулями. Однак для правильного дешифрування потрібно знати, які саме модулі є справжніми, а які - надлишковими. Ця інформація стає частиною ключа шифрування. На рисунку 2.10 представлено алгоритм роботи такого методу.

Рисунок 2.10 - Алгоритм роботи додавання надлишкових модулів

Процес починається з вибору базових модулів, необхідних для представлення значень пікселів. Потім генеруються додаткові надлишкові модулі, які також мають бути взаємно простими як між собою, так і з базовими модулями. Після перевірки взаємної простоти всі модулі змішуються за певним правилом, щоб приховати, які з них є справжніми. При шифруванні значення залишків обчислюються за всіма модулями, створюючи надлишкове представлення даних. Інформація про те, які модулі є справжніми, зберігається окремо і передається отримувачу захищеним каналом. Без цієї інформації,

навіть маючи всі залишки, неможливо правильно відновити оригінальні значення пікселів.

Четвертий метод не пов’язаний з модулями, але пов’язаний зі залишками. Ідея полягає в тому, щоб для кожного значення пікселя обчислюються залишки за всіма модулями стандартним способом, а потім взяти спеціальний ключ перемішування, який виглядатиме як масив індексів [4,1,5,2,6,3] при залишках [r_1, r_2, r_3, r_4, r_5, r_6].

Тобто, все починається з введення початкових даних та обчислення залишків для кожного пікселя зображення. На цьому етапі для кожного значення пікселя обчислюються залишки по всіх заданих модулях системи числення. Далі відбувається генерація ключа перемішування, який визначатиме порядок перестановки залишків. Цей ключ може бути згенерований різними способами та містить інформацію про те, як саме будуть переставлені залишки в кожному блоці.

Наступним кроком є розділення отриманого масиву залишків на блоки для зручності обробки. Розмір блоків може варіюватися залежно від конкретної реалізації та вимог до системи. Після цього починається ітераційний процес обробки блоків. Для кожного блоку виконується перемішування залишків згідно з визначеним ключем. При цьому важливо зберігати інформацію про порядок перемішування для можливості подальшого відновлення даних. Коли всі блоки оброблені, відбувається збереження ключа перемішування, який буде необхідний для дешифрування. На рисунку 2.11 представлено алгоритм четвертого методу із перемішуванням залишків.

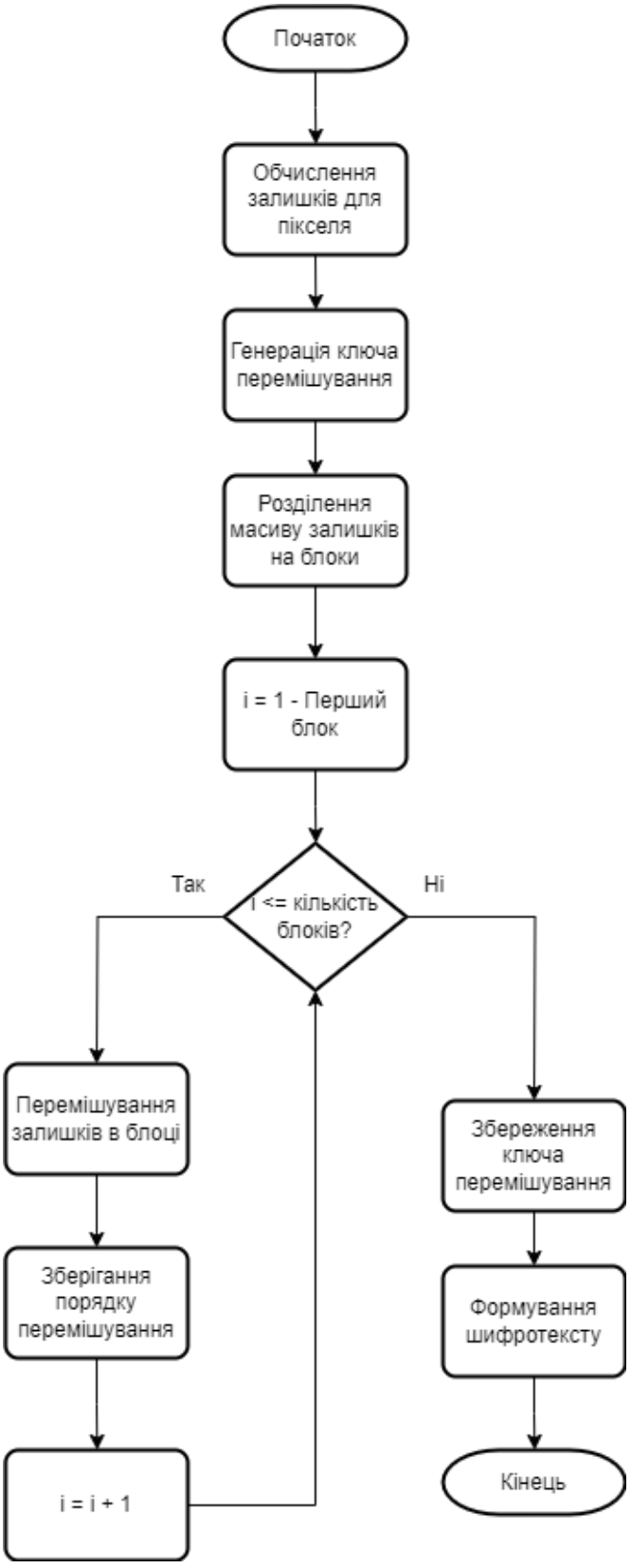

Рисунок 2.11 – Алгоритм перемішування залишків

І останнім методом, власне, є дво- чи трьорівневе шифрування. Тобто, після отримання результату, не реалізовувати другу картинку, а власне

шифрувати отримані дані зверху через другий метод кодування, наприклад, той же самий Base64.

Висновки до розділу 2

Було досліджено математичну модель системи залишкових класів та найменшого спільного кратного. В роботі розглянуто основні теореми, включаючи Китайську теорему про залишки, яка є фундаментальною для процесу відновлення даних в СЗК.

Було проаналізовано алгоритм шифрування зображень з використанням системи залишкових класів, який включає етапи перетворення зображення в матрицю пікселів, вибір взаємно простих модулів, обчислення залишків, формування шифротексту та його передачу.

Було досліджено алгоритм зворотного перетворення з системи залишкових класів, який базується на Китайській теоремі про залишки та дозволяє відновлювати оригінальні значення пікселів з їх представлення у вигляді залишків за умови знання правильних модулів.

Було розглянуто методи оптимізації складності алгоритмів, включаючи можливості паралельних обчислень при знаходженні залишків, ефективне структурування даних та використання попередньо обчислених констант для прискорення процесу відновлення даних.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1. Програмна реалізація алгоритмів СЗК

Для програмної реалізації СЗК було використано мову Python. Перш за все створено файл RNS_main.py, де і було написано весь код. Відповідно в файлі RNS_main створено клас RNS, в якого входять методи по типу ініціалізуючого і самописні методи перетворення в РНС.

Ініціалізуючий метод виглядав так:

```
class ResidueNumberSystem:
  def __init__(self, moduli):
    if not self._check_coprime(moduli):
      raise ValueError("Модулі повинні бути взаємно простими")
    self.moduli = moduli
    self.M = 1
    for m in moduli:
      self.M *= m
    self.M_i = [self.M // m for m in moduli]
    self.y_i = [self._mod_inverse(self.M_i[i], moduli[i])
          for i in range(len(moduli))]
```

Конструктор класу ResidueNumberSystem ініціалізує систему залишкових класів та виконує всі необхідні попередні обчислення. На вхід конструктор отримує список модулів, які будуть використовуватися в системі.

Першим кроком відбувається перевірка чи є ці модулі взаємно простими числами, оскільки це є обов'язковою умовою для коректної роботи системи залишкових класів. Якщо модулі не є взаємно простими, генерується помилка.

Після перевірки модулі зберігаються в атрибут класу self.moduli, а також обчислюється їх загальний добуток M, який зберігається в self.M. Цей добуток є важливим параметром системи, оскільки визначає її робочий діапазон.

Далі для кожного модуля обчислюється значення M_i, яке є результатом ділення загального добутку M на відповідний модуль. Ці значення зберігаються в списку self.M_i і будуть використовуватися при відновленні чисел з їх представлення в системі залишкових класів.

На останньому етапі для кожного M_i обчислюється його мультиплікативно обернений елемент по відповідному модулю. Ці обернені елементи необхідні для реалізації Китайської теореми про залишки, яка використовується при відновленні чисел. Обчислені обернені елементи зберігаються в списку self.y_i.

Після конструктора класу йдуть власне самописні методи. Метод _gcd реалізує алгоритм Евкліда для знаходження найбільшого спільного дільника двох чисел.

Він працює шляхом послідовного ділення з остачею, де на кожній ітерації більше число замінюється на менше, а менше - на остачу від ділення. Цей процес продовжується доки остача не стане рівною нулю. Останнє ненульове значення і буде найбільшим спільним дільником.

Метод _check_coprime використовує _gcd для перевірки взаємної простоти всіх чисел у списку. Він порівнює кожне число з кожним іншим числом зі списку, використовуючи вкладені цикли. Для кожної пари чисел обчислюється їх НСД, і якщо хоча б для однієї пари НСД виявляється не рівним одиниці, це означає що числа не є взаємно простими, і метод повертає False.

Якщо всі пари чисел мають НСД рівний одиниці, метод повертає True, підтверджуючи що всі числа в списку взаємно прості. Детальніше в додатку А. Алгоритм роботи Евкліда в цьому коді представлений на рисунку 3.1.

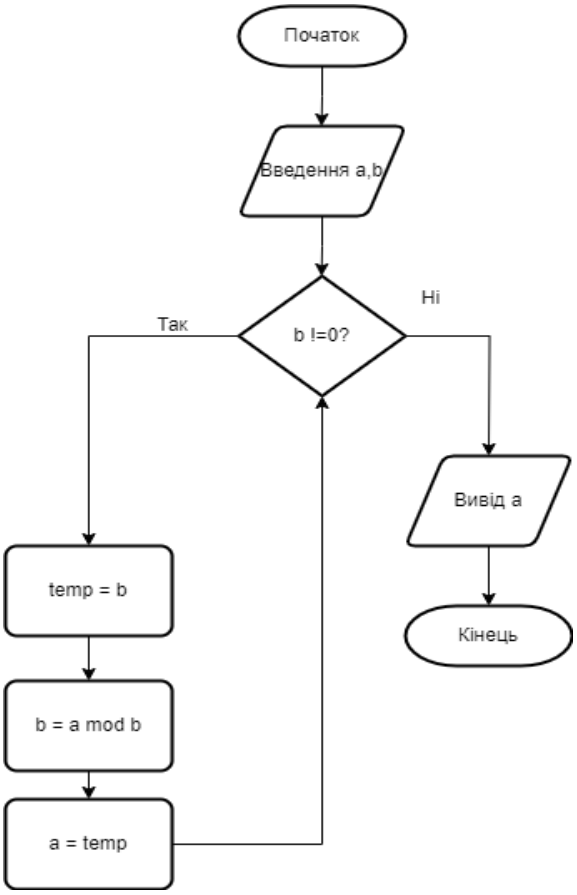

Рисунок 3.1 – Реалізація алгоритму Евкліда

На рисунку 3.2 зображено алгоритм, запропонований в ході реалізації глобального застосунку.

```
Пошук НСД для чисел 48 і 18:
       a         b   Остача
-----------------------
      48        18       12
      18        12        6
      12         6        0
-----------------------
НСД = 6
```

Рисунок 3.2 – Виведення інформації при застосуванні методу в класі

Метод _extended_gcd реалізує розширений алгоритм Евкліда, який окрім знаходження НСД двох чисел a і b, також знаходить коефіцієнти x та y такі, що ax + by = НСД(a,b). Це рекурсивна реалізація, де на кожному кроці рекурсії обчислюються нові коефіцієнти. Коли a стає рівним 0, рекурсія завершується, і ми починаємо повертатися назад, обчислюючи коефіцієнти для кожного попереднього кроку.

```
def _extended_gcd(self, a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = self._extended_gcd(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y
```

В той час як метод _mod_inverse використовує розширений алгоритм Евкліда для знаходження мультиплікативно оберненого елемента числа a за модулем m. Мультиплікативно обернений елемент x - це таке число, що (a * x) mod m = 1. Для існування такого елемента необхідно, щоб числа a та m були взаємно простими (їх НСД повинен дорівнювати 1). Якщо НСД не дорівнює 1, це означає що оберненого елемента не існує, і метод генерує помилку.

Також в коді є 2 методи to_residues і from_residues. Метод to_residues перетворює звичайне число в його представлення в системі залишкових класів. Спочатку перевіряється, чи не перевищує число загальний добуток модулів M, оскільки система може представляти числа тільки до M-1. Потім для кожного модуля обчислюється залишок від ділення числа на цей модуль, і всі залишки збираються в список.

Метод from_residues виконує зворотне перетворення - відновлює оригінальне число з його представлення в СЗК, використовуючи Китайську теорему про залишки. Спочатку перевіряється, чи кількість залишків відповідає

кількості модулів. Потім для кожного залишку виконується множення на відповідні попередньо обчислені константи M_i (M/mi) та y_i (мультиплікативно обернені елементи). Всі добутки додаються, і результат береться за модулем M.

На рисунку 3.3 зображено процес перетворення і відновлення.

```
Процес перетворення в СЗК:
100 mod 7 = 2
100 mod 11 = 1
100 mod 13 = 9
Результат: [2, 1, 9]

Процес відновлення числа:
M = 1001 (добуток всіх модулів)
Доданок 1: 2 * 143 * 5 = 1430
Доданок 2: 1 * 91 * 4 = 364
Доданок 3: 9 * 77 * 12 = 8316

Відновлене число: 100
```

Рисунок 3.3 – Результат роботи методів to_residues і from_residues

Оскільки були розглянуті математичні формули додавання, віднімання та множення, ці методи також були реалізовані.

Метод додавання виглядає наступним чином:

```
def add(self, residues1, residues2):
    if len(residues1) != len(self.moduli) or len(residues2) != len(self.moduli):
        raise ValueError("Некоректна кількість залишків")
    result = []
    for i in range(len(residues1)):
        sum_residue = (residues1[i] + residues2[i]) % self.moduli[i]
        result.append(sum_residue)
    return result
```

Метод add виконує операцію додавання двох чисел, представлених у системі залишкових класів. На вхід він отримує два списки залишків - residues1 та residues2. Спочатку виконується перевірка коректності вхідних даних - чи довжина обох списків залишків відповідає кількості модулів у системі. Якщо кількість залишків не співпадає з кількістю модулів, генерується помилка.

Потім виконується власне додавання: для кожної пари відповідних залишків виконується їх додавання за модулем відповідного модуля. Тобто для кожного модуля mi виконується операція (r1i + r2i) mod mi, де r1i та r2i - відповідні залишки від першого та другого числа.

Відповідно віднімання і множення виглядає схожим чином і представлене в додатку А.

Вся програма працює наступним чином, зображеним на рисунку 3.4.

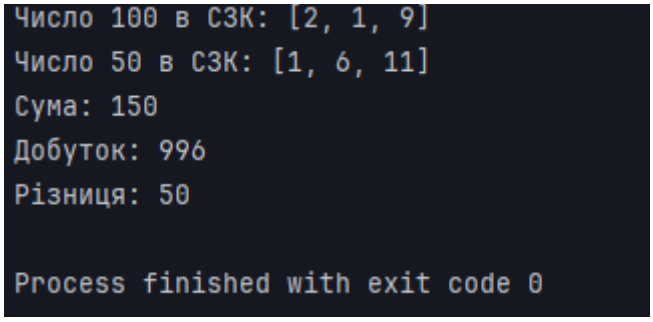

Рисунок 3.4 – Результат роботи програмної реалізації

3.2. Реалізація механізму шифрування

Механізм шифрування розпочинається з того, що минулий файл стає модулем, який імпортується в основний код. Імпорти encryption.py виглядають наступним чином:

```
import numpy as np
from PIL import Image
from residue_system import ResidueNumberSystem
```

NumPy (Numerical Python) використовується для ефективної роботи з багатовимірними масивами та матрицями. В контексті обробки зображень NumPy надає можливість перетворювати зображення в масиви, де кожен піксель представлений як набір RGB значень, що потрібно в контексті розробки програмного забезпечення по шифруванню за допомогою СЗК.

PIL працює з картинками і дозволяє реалізовувати створення картинок в подальшому для змінених пікселів в роботі.

Основний код криптування виглядає так:

Процес шифрування починається з завантаження зображення та його конвертації у RGB формат. Кожен піксель зображення представлений трьома значеннями (червоний, зелений, синій) у діапазоні від 0 до 255. Зображення перетворюється у масив numpy для зручної обробки.

Далі код проходить через кожен піксель зображення і для кожного кольорового каналу (R,G,B) бере числове значення та конвертує його у набір залишків використовуючи задані модулі.

Ці залишки зберігаються у спеціальному 4-вимірному масиві, де перші три виміри відповідають за висоту, ширину та колірний канал, а четвертий вимір містить залишки для кожного модуля.

Для візуалізації результату шифрування код створює спотворене зображення. Воно генерується шляхом множення першого залишку на 50 та взяття залишку від ділення на 256 для кожного пікселя. Це створює візуально спотворену версію оригінального зображення, яка зберігається як "encrypted.png".

Оскільки не було видалено попередніх принтів, то результат дії програми зображений на рисунку 3.5 зі незначними виводами попереднього розділу.

```
encrypted_data = np.zeros((height, width, channels, num_moduli), dtype=int)
for y in range(height):
    for x in range(width):
        for c in range(channels):
```

pixel_value = int(rgb_matrix[y, x, c])

residues = self.rns.to_residues(pixel_value)

encrypted_data[y, x, c] = residues

```
Розмір оригінального зображення: (1280, 1250, 3)
Оригінальні RGB значення (перші 5 пікселів):
[[27 25 72]
 [28 26 73]
 [26 24 71]
 [35 33 80]
 [44 42 89]]

Зашифровані значення (залишки для перших 5 пікселів):
[[ 6  5  1]
 [ 2  8  4]
 [ 6  1  8]
 [ 1  3 10]
 [ 2  4 11]]

Дані зашифровано та збережено візуалізацію як 'encrypted.png'
```

Рисунок 3.5 – Результат роботи шифрування фотографії

Для тестування було обрано картинку, представлено на рисунку 3.6,

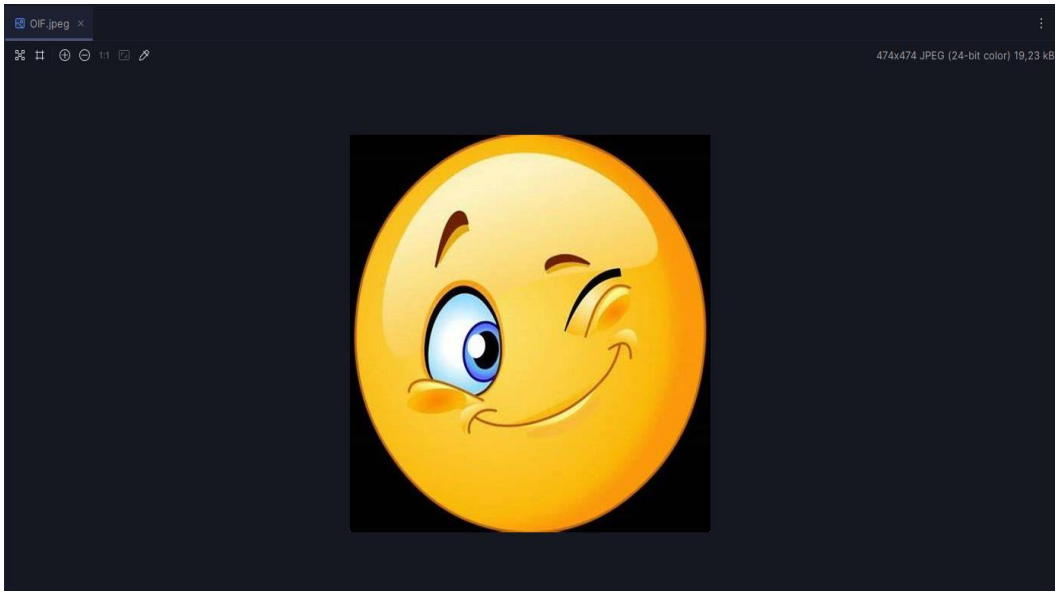

Рисунок 3.6 – Тестова картинка для шифрування у форматі JPEG

Результат шифрування зображений на рисунку 3.7.

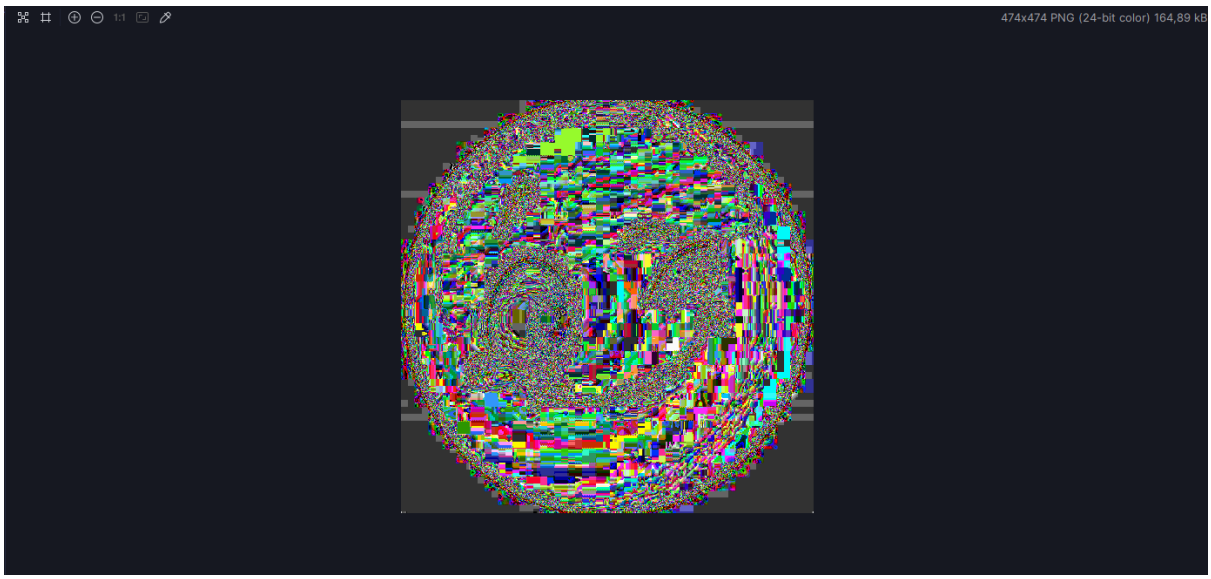

Рисунок 3.7 – Зашифрована картинка за допомогою СЗК

Для дешифрування використовується метод decrypt. Цей метод відповідає за дешифрування зображення, яке було раніше зашифроване за допомогою системи залишкових чисел. Спочатку метод отримує розміри зашифрованих даних - висоту, ширину, кількість кольорових каналів та кількість використаних модулів. На основі цих розмірів створюється новий порожній масив для зберігання розшифрованого зображення.

Далі код послідовно проходить через кожен піксель зашифрованого зображення. Для кожного пікселя та кожного кольорового каналу (RGB) беруться відповідні залишки, які були отримані при шифруванні. Ці залишки передаються методу from_residues, який конвертує їх назад у оригінальне число. Отримане число зберігається у відповідній позиції розшифрованого масиву.

Після обробки всіх пікселів код виводить перші п'ять розшифрованих RGB значень для перевірки коректності дешифрування. В кінці метод перетворює розшифрований масив numpy назад у зображення за допомогою Image.fromarray та повертає його. Ключовим моментом тут є те, що процес дешифрування є оберненим до шифрування - він використовує ті ж самі модулі для конвертації набору залишків назад в оригінальні RGB значення.

У коді це представлено як:

```
def decrypt(self, encrypted_data):
    height, width, channels, _ = encrypted_data.shape
    decrypted = np.zeros((height, width, channels), dtype=np.uint8)
    for y in range(height):
        for x in range(width):
            for c in range(channels):
                residues = encrypted_data[y, x, c]
                original = self.rns.from_residues(residues)
                decrypted[y, x, c] = original
    return Image.fromarray(decrypted)
```

Детальніше весь криптор можна переглянути у додатку А. Результатом декриптування стає повернення картинки у нормальний стан, що зображено на рисунку 3.8.

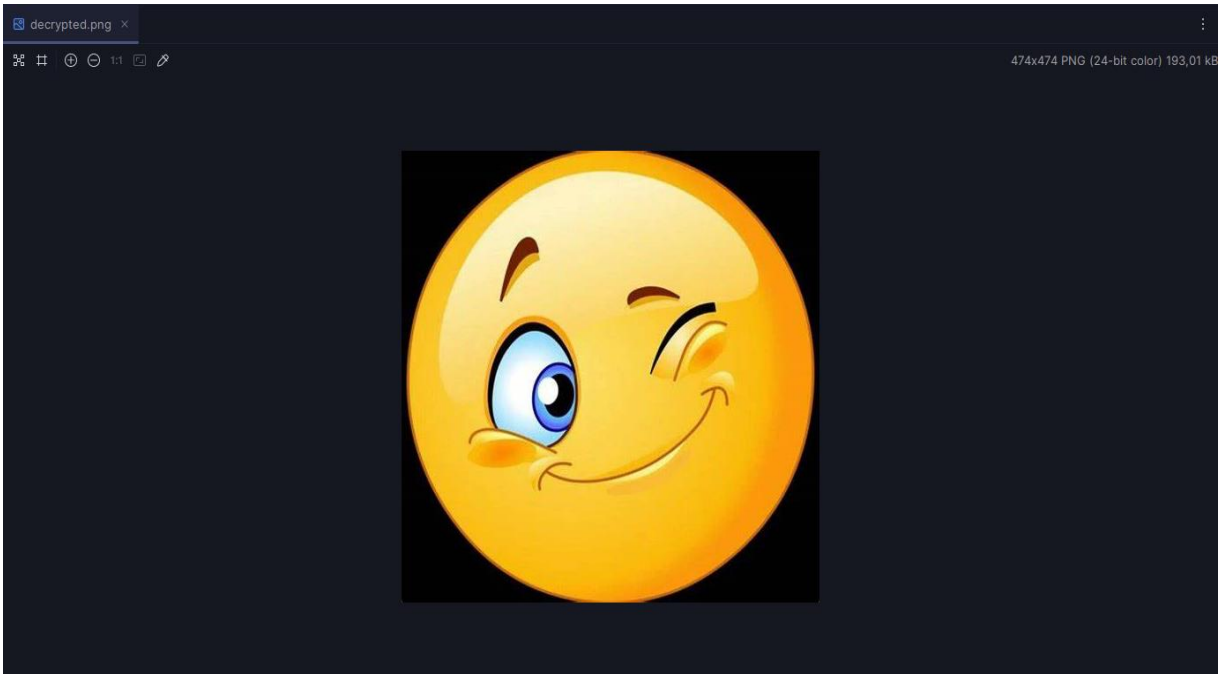

Рисунок 3.8 – Розшифрована картинка у форматі PNG

У роботі було використано модулі 7, 11, 13 для демонстрації процесу, що і впливає на швидкодію. В подальшому буде проаналізовано вплив модулів і інших складників на швидкодію програмної реалізації.

3.3. Реалізація графічної оболонки

Для зручності користування було вирішено зробити графічну оболонку, завдяки якій можна буде задавати модулі власноруч і бачити зашифровану і дешифровану картинку одразу. Для цього було обрано модуль Tkinter, який по стандарту стоїть в python. Першим ділом було реалізовано зміну модулів, що зображено на рисунку 3.9.

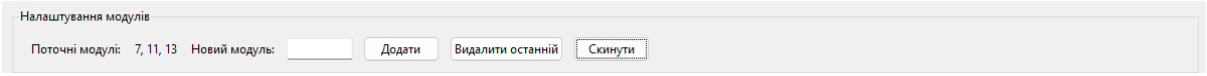

Рисунок 3.9 – Реалізація зміни модулів через графічну оболонку

Реалізовано це наступним чином:

```
  def add_modulus(self):
try:
    new_mod = int(self.new_modulus.get())
    if new_mod <= 0:
        raise ValueError("Модуль повинен бути додатнім числом")
    self.moduli.append(new_mod)
    self.update_moduli_display()
    self.update_encryptor()
    self.new_modulus.delete(0, tk.END)
except ValueError as e:
    messagebox.showerror("Помилка", str(e))
```

Якщо менше одного модуля, то виникає помилка, яка зображена на рисунку 3.10.

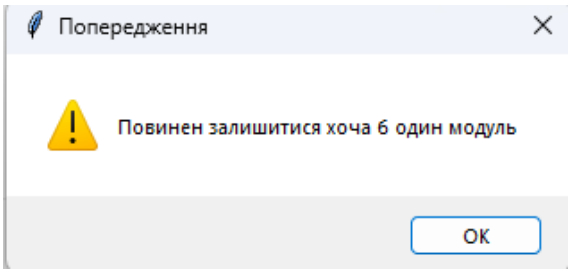

Рисунок 3.10 – Попередження стосовно одного модуля

Тобто, програма не дає менше зробити. Відповідно, перший модуль завжди буде сталим, який заданий в програмі. На рисунку 3.11 видно, що програма не дозволяє видалити модуль 7, який являється єдиним модулем.

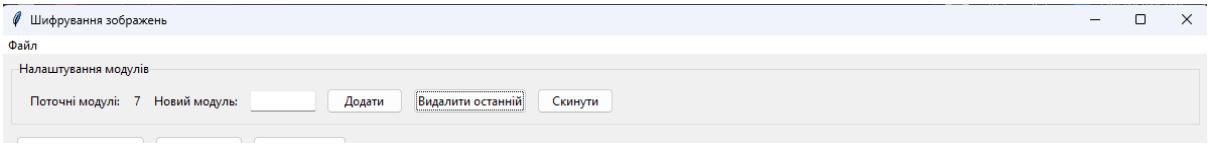

Рисунок 3.11 – Збереження хоча б одного модуля

Кнопка скинути повертає стандартні задані значення в програмі - 7, 11, 13 – як приклад для цього програмного забезпечення.

Після того, як було реалізовано верхню панель, почалась реалізація основного компоненту. Його код виглядає наступним чином:

```
def create_toolbar(self):
    toolbar = ttk.Frame(self.root)
    toolbar.pack(fill=tk.X, padx=10, pady=5)
ttk.Button(toolbar, text="Відкрити зображення", command=self.open_image).pack(side=tk.LEFT, padx=5)
ttk.Button(toolbar, text="Зашифрувати", command=self.encrypt_image).pack(side=tk.LEFT, padx=5)
ttk.Button(toolbar, text="Розшифрувати", command=self.decrypt_image).pack(side=tk.LEFT, padx=5)
```

```
def create_image_panel(self):
    image_frame = ttk.Frame(self.root)
    image_frame.pack(expand=True, fill=tk.BOTH, padx=10, pady=5)
    frames = [
        ("Оригінальне", 0),
        ("Зашифроване", 1),
        ("Розшифроване", 2)
    ]
```

Цей код створює три кнопки - для відкриття зображення, шифрування та дешифрування фотографії. Варто зазначити, що немає різниці, чи фотографія була зашифрована, а потім розшифрована, чи ні – воно працює ідентично. На рисунку 3.12 продемонстровано завантаження фотографії у програму

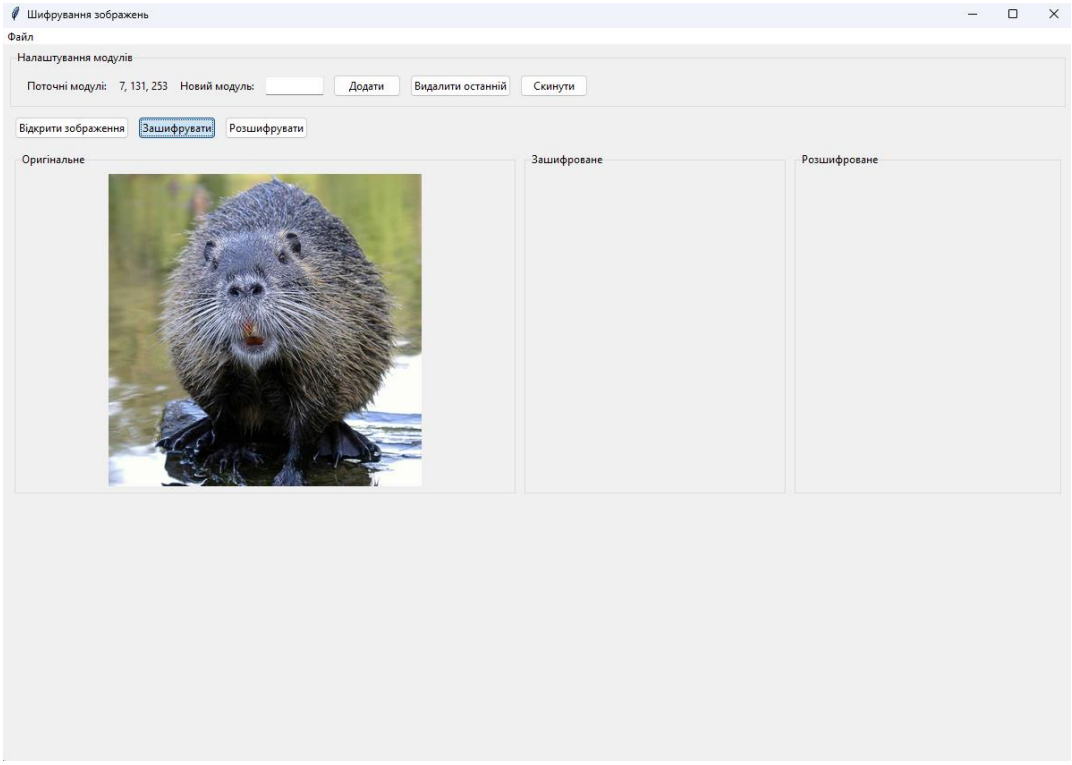

Рисунок 3.12 – Завантаження фотографії у програму

Після успішного завантаження і виведення обраної картинки, натискається друга кнопка, яка власне і виконує основний код. На зображеному фрагменті коду видно, що використовується метод, імпортований з класу СЗК.

```
def encrypt_image(self):
    if self.current_image_path is None:
    messagebox.showerror("Помилка", "Спочатку відкрийте зображення")
        return
    try:
        self.encrypted_data = self.encryptor.encrypt(self.current_image_path)
        self.show_image("encrypted.png", self.encryped_label)
        messagebox.showinfo("Успіх", "Зображення успішно зашифровано")
    except Exception as e:
        messagebox.showerror("Помилка", f"Помилка при шифруванні: {str(e)}")
```

Метод відповідає за процес шифрування зображення в додатку. Спочатку він перевіряє, чи існує шлях до поточного зображення. Якщо self.current_image_path дорівнює None, тобто жодне зображення не було обрано, програма виводить повідомлення про помилку: "Спочатку відкрийте зображення" та завершує виконання методу.

У блоці try відбувається основна операція шифрування. Використовується об'єкт self.encryptor з методом encrypt(), якому передається шлях до поточного зображення. Результат шифрування зберігається в self.encrypted_data.

Після шифрування викликається метод show_image() для відображення зашифрованого зображення на мітці self.encryped_label. Це дозволяє користувачеві одразу побачити змінене зображення.

Успішне завершення операції супроводжується інформаційним повідомленням "Зображення успішно зашифровано". Блок except призначений для обробки будь-яких винятків, що можуть виникнути під час процесу

шифрування. Якщо станеться помилка, буде показано повідомлення з детальним описом проблеми.

Відповідно, на рисунку 3.13 зображено успішне шифрування картинки.

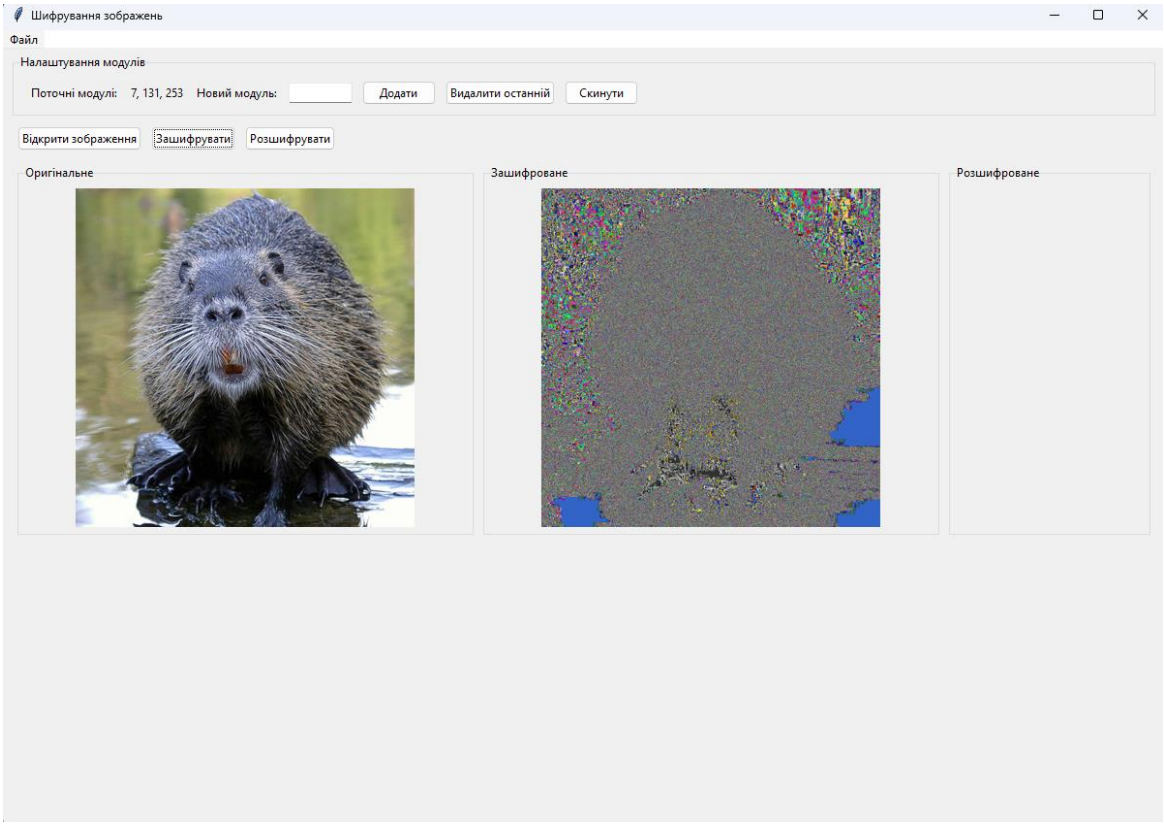

Рисунок 3.13 – Успішне шифрування картинки

Для декриптування працює такий код:

```
def decrypt_image(self):
    if self.encrypted_data is None:
 messagebox.showerror("Помилка", "Спочатку зашифруйте зображення")
        return
    try:
        decrypted_image = self.encryptor.decrypt(self.encrypted_data)
        decrypted_image.save("decrypted.png")
        self.show_image("decrypted.png", self.розшифроване_label)
```

messagebox.showinfo("Успіх", "Зображення успішно розшифровано")

except Exception as e:

messagebox.showerror("Помилка", f"Помилка при розшифруванні: {str(e)}")

Це метод decrypt_image(), який призначений для розшифрування зображення в додатку з графічним інтерфейсом, написаному українською мовою.

Метод виконує кілька важливих перевірок та дій. Насамперед, він перевіряє наявність зашифрованих даних. Якщо self.encrypted_data дорівнює None, тобто немає зашифрованих даних, програма показує повідомлення про помилку українською мовою: "Спочатку зашифруйте зображення" та припиняє виконання методу.

Далі код переходить до блоку try-except, який дозволяє безпечно обробляти можливі винятки під час процесу розшифрування.

Всередині блоку try відбуваються наступні дії: за допомогою об'єкта self.encryptor викликається метод decrypt(), який розшифровує збережені раніше дані. Розшифроване зображення одразу зберігається у файл з назвою "decrypted.png".

Одразу після збереження викликається метод show_image(), який відображає щойно розшифроване зображення на мітці self.decrypted_label.

Після успішного виконання всіх операцій користувачеві показується інформаційне повідомлення "Зображення успішно розшифровано". Якщо під час розшифрування виникає будь-яка помилка, блок except перехоплює виняток та показує повідомлення про помилку, яке включає детальний опис проблеми

На рисунку 3.14 продемонстроване успішне дешифрування картинки. Загалом, на рисунку представлено весь функціонал програмного рішення.

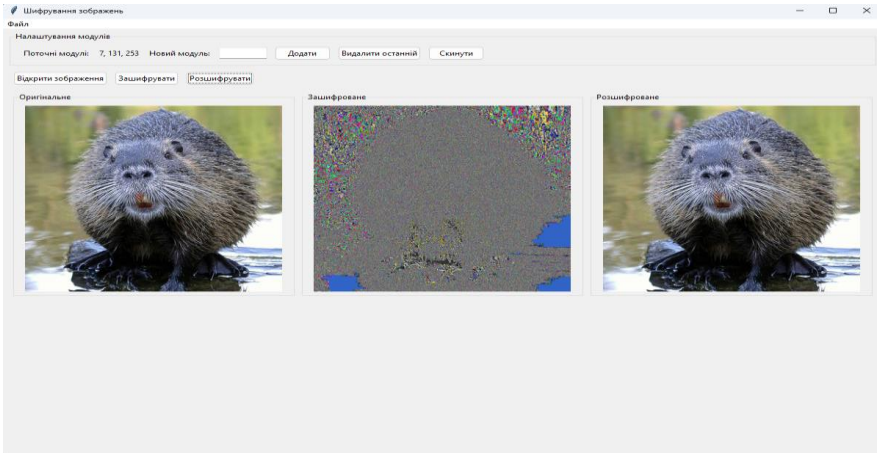

Рисунок 3.14 – Дешифрування картинки

Також, варто зауважити, що процес обробленої і дешифраної картинки також зберігаються у каталозі, де знаходиться сама реалізація. На рисунку 3.15 видно, що зберігаються два зображення - decrypted і encrypted.png

Рисунок 3.15 – Збереження нових зображень

Отож, в main.py знаходиться реалізація GUI, яку також можна оглянути детальніше в додатку А.

3.4. Експериментальне дослідження швидкодії алгоритму

Для дослідження швидкодії було вирішено написати звичайний декоратор. Декоратор – це особливий тип функцій, який дозволяє змінювати або розширювати поведінку інших функцій без безпосередньої зміни їхнього вихідного коду.

Для реалізації, було створено новий файл decorators.py, в якому і прописано наступний код:

```
import time
def measure_time(func):
```

```
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Функція {func.__name__} виконалася за {end_time - start_time:.4f} секунд")
        return result
    return wrapper
```

У першому рядку імпортується модуль time, який надає функції для роботи з часом у Python. Далі оголошується функція measure_time, яка приймає іншу функцію як аргумент. Це основна мета декоратора - отримати функцію та модифікувати її поведінку. Всередині measure_time створюється внутрішня функція wrapper, яка може приймати будь-яку кількість позиційних і keyword аргументів завдяки *args, **kwargs. Це дозволяє декоратору працювати з функціями різної складності та різної сигнатури.

Функція wrapper спочатку записує поточний час за допомогою time.time() перед викликом оригінальної функції. Це буде час початку виконання. Потім викликається оригінальна функція func з усіма переданими їй аргументами, а її результат зберігається у змінну result. Це важливо, аби декоратор не порушував роботу оригінальної функції. Одразу після виклику функції знову фіксується поточний час за допомогою time.time(). Це час завершення виконання функції.

На прикладі демонстраційного методу, описаного в ході реалізації алгоритму, було виявлено, що метод виконується за 0.05 секунди. На рисунку 3.16 представлено підключення декоратора до одного з методів з розділу 3.1. і 3.2.

```python
@measure_time
def demonstrate_rns_conversion(number, moduli):
    print(f"\nДемонстрація перетворення числа {number}")
    print(f"Модулі системи: {moduli}")

    print("\nПроцес перетворення в СЗК:")
    residues = []
    for m in moduli:
        residue = number % m
        print(f"{number} mod {m} = {residue}")
        residues.append(residue)
    print(f"Результат: {residues}")
```

Рисунок 3.16 – Додавання декоратора до першого методу

Результат продемонстровано на рисунку 3.17.

```
Процес відновлення числа:
M = 1001 (добуток всіх модулів)
Доданок 1: 2 * 143 * 5 = 1430
Доданок 2: 1 * 91 * 4 = 364
Доданок 3: 9 * 77 * 12 = 8316


Відновлене число: 100
Функція demonstrate_rns_conversion виконалася за 0.0005 секунд
```

Рисунок 3.17 – Результат роботи методу

Результатом роботи для алгоритму Евкліда стали приголомшливі 0.00054264071 секунд для 2 ітерацій, і 0.003121 секунд, для важчих чисел.

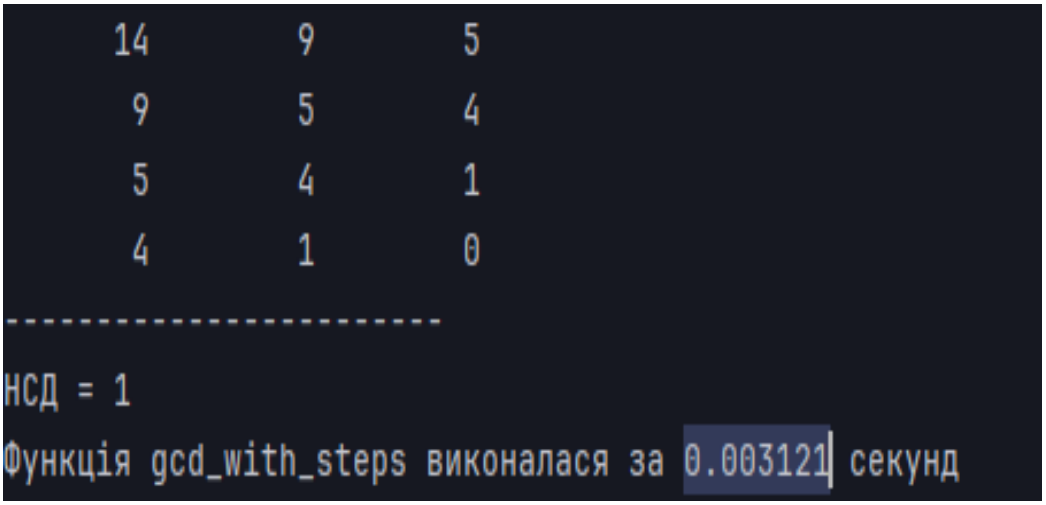

Рисунок 3.18 – Перевірка НСД на швидкість виконання

Далі друкується інформація про час виконання. Використовується форматований рядок, який показує назву функції (через func.__name__) та час її виконання. Час виводиться з точністю до 4 знаків після коми. Врешті-решт, функція wrapper повертає результат оригінальної функції, що дозволяє зберегти її нормальну поведінку. Останній рядок return wrapper означає, що декоратор повертає модифіковану версію вхідної функції.

Висновки до розділу 3

У розділі було реалізовано комплекс математичних алгоритмів, зокрема алгоритми Системи Залишкових Класів: алгоритм Евкліда, знаходження найменшого спільного кратного та теоретико-числового базису.

Розроблено механізм шифрування зображень з використанням маніпуляцій з RGB-каналами.

Створено графічний інтерфейс користувача на базі бібліотеки Tkinter мови Python.

Для дослідження ефективності алгоритмів було впроваджено спеціальний декоратор вимірювання часу виконання, що дозволило провести детальний аналіз швидкодії розроблених алгоритмів математичного перетворення та криптографічного захисту інформації.

ВИСНОВКИ

Проаналізовано основи системи залишкових класів, включаючи принципи представлення чисел у вигляді набору залишків від ділення на взаємно прості модулі, механізми виконання арифметичних операцій та особливості відновлення чисел з їх залишкового представлення.

Досліджено методи шифрування даних з використанням СЗК, включаючи різні підходи до формування криптосистем на основі властивостей модульної арифметики.

Оглянуто ключові характеристики та властивості перетворень в системі залишкових класів, що включають мультиплікативну інверсію, незалежність залишків, відсутність розповсюдження помилок, симетричність операцій та інші важливі аспекти. Проаналізовано практичне значення цих властивостей для реалізації ефективних та надійних систем обробки даних.

Було досліджено математичну модель системи залишкових класів та найменшого спільного кратного. В роботі розглянуто основні теореми, включаючи Китайську теорему про залишки, яка є фундаментальною для процесу відновлення даних в СЗК.

Було проаналізовано алгоритм шифрування зображень з використанням системи залишкових класів, який включає етапи перетворення зображення в матрицю пікселів, вибір взаємно простих модулів, обчислення залишків, формування шифротексту та його передачу.

Було досліджено алгоритм зворотного перетворення з системи залишкових класів, який базується на Китайській теоремі про залишки та дозволяє відновлювати оригінальні значення пікселів з їх представлення у вигляді залишків за умови знання правильних модулів.

Було розглянуто методи оптимізації складності алгоритмів, включаючи можливості паралельних обчислень при знаходженні залишків, ефективне структурування даних та використання попередньо обчислених констант для прискорення процесу відновлення даних.

У розділі було реалізовано комплекс математичних алгоритмів, зокрема алгоритми Системи Залишкових Класів: алгоритм Евкліда, знаходження найменшого спільного кратного та теоретико-числового базису. Розроблено механізм шифрування зображень з використанням маніпуляцій з RGB-каналами.

Створено графічний інтерфейс користувача на базі бібліотеки Tkinter мови Python. Для дослідження ефективності алгоритмів було впроваджено спеціальний декоратор вимірювання часу виконання, що дозволило провести детальний аналіз швидкодії розроблених алгоритмів математичного перетворення та криптографічного захисту інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Карпінський М.П. Еліптична крива для асиметричної криптографічної системи. // М.П.Карпінський, І.З.Якименко, І. Дуда /Вісник Тернопільського державного технічного університету імені Ів.Пулюя, - Том 6, - №3 – 2001. – С: 91 – 95.

2. А.А. Болотов Алгоритмические основы эллиптической криптографии./ А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских./ – Москва:МЭИ.– 2000. – 100 с.

3. ELKIES N. D. Elliptic and modular curves over finite fields and related computational issues. In Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkins, D. A. Buell and J. T. Teitelbaum, eds. AMS/IP Studies in Advanced Mathematics, vol. 7. American Mathematics Society, Providence, R. I.– 1998.– pp. 21–76.

4. H. L. Garner, "The Residue Number System", IRE Trans. Electronic Computers, vol. EL-8, no. 6, P. 140-147, 1959.

5. IBM Security Solutions [Електронний ресурс]. Режим доступу: https://www.ibm.com/security/products

6. IEEE P1363 / D8(Draft Version 8). Standard Specifications for Public Key Cryptography.

7. Kozaczko D. Vector Module Exponential in the Remaining Classes System/ D.Kozaczko, I.Yakymenko, M. Kasianchuk, S.Ivasiev // Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2015) – Warsaw, Poland. – V.1. – September, 2015. – P.161–163.

8. Omondi B. Residue Number Systems: Theory and Implementation / B.Omondi, Premkumar S. - World Scientific Publishing Co. Pte. Ltd. - 2007. - 296 p.

9. Oracle Security Documentation [Електронний ресурс]. Режим доступу: https://docs.oracle.com/en/security/

10. Patterson D. Computer Architecture. A Quantitative Approach / D.

Patterson, J. Hennessy // Morgan Kaufmann Publishers, Inc. –1996.

11. Schoof R. Counting points on elliptic curves over finite fields. J. The´or. Nombres. // R. Schoof / – Bordeaux 7.– 1995.– 219 –254p.

12. Schoof R. Elliptic curves over finite fields and the computation of square roots modulo p.// R. Schoof / –Bordeaux: Math. Comput. –1985.–№ 44.– 483– 494p.

13. Siewobr H. Application of Residue Number System to Advance Encryption Standard Algorithm / H. Siewobr, K. Gbolagade // International Journal of Computational Intelligence and Information Security. - Vol. 2, No. 7, July, 2011. - P. 66-72.

14. Szabo N. Residue arithmetic and its applications to computer technology / N. Szabo and R. Tanaka. - New York: McGraw Hill. – 1967. - 236 p.

15. X9.62-1998 [2]. Public Key Cryptography For The Financial Services Industry. Public Key Cryptography For The Financial Services Industry.

16. Yang L. L. A residue number system based parallel communication scheme using orthogonal signaling: Part II—Multipath fading channels / L. L. Yang, L. Hanzo // IEEE Trans. Veh. Technol. – 2002. - Vol. 51. – P. 1541-1553.

17. Бессалов А.В. Криптосистемы на эллиптических кривых: Учеб. пособие.// А.В.Бессалов, А.Б.Телиженко / – К.: ІВЦ «Видавництво «Політехніка»».– 2004. – 224 С.

18. Варновский Н. П. Криптография и теория сложности // Математическое просвещение. / Н. П. Варновский / – М:1998. - №2. - С. 71 - 86

19. Карпінський М.П. Використання символів Якобі в криптографії ЕК // М.П.Карпінський, І.З.Якименко, А.А.Хомінчук / ДУІКТ/06/ Матеріали ІІІ Міжнародної науково-технічної конференції "Світ інформації та телекомунікацій-2006" Україна.– Київ. – 2006. – С.208.

20. Карпінський М.П. Використання технології паралельних обчислень для рахування кількості точок еліптичної кривої. // М.П. Карпінський, І.З. Якименко, А.А. Хомінчук / Вісник Східноукраїнського національного університету імені Володимира Даля.– 2008. – № 8 (126), Частина 1. – С. 207-210.

21. Кінах Я.І Удосконалення мережевих криптоаналітичних алгоритмів на еліптичних кривих. // Я.І Кінах, І.З. Якименко/ Вісник Східноукраїнського національного університету імені Володимира Даля.-№6(136), Т.1.- 2009.- С.48-51.

22. М.Карпінський Метод генерування параметрів еліптичних кривих. // М.Карпінський, І.Васильцов, І.Якименко, Я.Кінах./ Правове, нормативне, та метрологічне забезпечення системи захисту інформації в Україні. К.:–2003– № 6.– С.74.

23. Маслачков О.Ю. Метод гомоморфного шифрування зображень при їх віддаленій обробці. Магістерська дисертація. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2021. URL: https://ela.kpi.ua/bitstream/123456789/46121/1/Maslachkov_magistr.pdf (дата звернення: 01.12.2024).

24. Р. Радчук. Аналіз безпеки шифрування зображень у системі залишкових класів. Збірник матеріалів науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. С. 43-47.

25. Р. Радчук. Аналіз ефективності системи залишкових класів у криптографічних методах захисту зображень. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. С. 87-92.

26. Сава Л.М. Аналіз та реалізація криптографічних перетворень для алгоритму ECDH. Матеріали V Міжнародної студентської науково - технічної конференції / Тернопіль: Тернопільський національний технічний університет ім. І.Пулюя (м. Тернопіль, 28-29 квітня 2022 р.), 2022.-с. 144.

27. Тимошенко Л.М. Алгоритми факторизації криптоаналізу асиметричних систем / Л.М. Тимошенко, К.В. Вербик, Я.М. Николайчук, С.В. Івасьєв //«Інформатика та математичні методи в моделюванні».- Одеса, ОНПУ. – 2014. - № (Том). – С. 248-253.

28. Тимошенко Л.М. Современные методы решения криптоаналитических задач/ Л.Н. Тимошенко, К.В. Вербик. – Матеріали ВШСМВС СКІТ.-Тернопіль.-2014.-С. 220-222.

29. Тимошенко Л.М. Удосконалений метод Ферма факторизації чисел / Л.М. Тимошенко, К.В. Вербик, С.В., Івасьєв - Зб.мат. пробл-наук. міжн. конф. ПНМК-2014. Львів.- 2014. - С.348-350.

30. Тимошенко Л.М. Удосконалення алгоритму факторизації для криптографічних систем захисту інформації/ Л.М. Тимошенко, С.В. Івасьєв, К.В. Вербик //«Сучасна спеціальна техніка».- Київ, ДНДЕКЦ МВС України. – 2014. - №1(36). – С.

31. Тимошенко Л.М. Шляхи вдосконалення алгоритму факторизації чисел Тимошенко, К.В. Вербик. – Матеріали ВШСМВС СКІТ.-Київ.-2014.-С. 220-222.

32. Якименко І.З. Підвищення ефективності обчислення точок на еліптичних кривих над обмеженими полями.// І.З.Якименко, А.О.Ботюк, М.П.Карпінський/ Вісник Тернопільського державного технічного університету імені Ів.Пулюя, - Том 8, - №4 – 2003. – С: 67 – 73.

33. Яцків Н.Г., Спецпроцесори обробки даних на основі перетворення Крестенсона – Галуа. / Н.Г. Яцків, Р.І. Король, В.В. Яцків, Т.Г. Федчишин // Вісник Технологічного університету Поділля. – 2003. -ТІ, №3. – С. 105-108.

ДОДАТОК А

Програмна реалізація

```
    def _gcd(self, a, b):
        while b:
            a, b = b, a % b
        return a

    def _check_coprime(self, numbers):
        for i in range(len(numbers)):
            for j in range(i + 1, len(numbers)):
                if self._gcd(numbers[i], numbers[j]) != 1:
                    return False
        return True

class ResidueNumberSystem:
    def __init__(self, moduli):
        if not self._check_coprime(moduli):
            raise ValueError("Модулі повинні бути взаємно простими")

        self.moduli = moduli
        self.M = 1
        for m in moduli:
            self.M *= m

        self.M_i = [self.M // m for m in moduli]
        self.y_i = [self._mod_inverse(self.M_i[i], moduli[i])
                    for i in range(len(moduli))]

    def _gcd(self, a, b):
        while b:
            a, b = b, a % b
        return a

    def _check_coprime(self, numbers):
        for i in range(len(numbers)):
            for j in range(i + 1, len(numbers)):
                if self._gcd(numbers[i], numbers[j]) != 1:
                    return False
        return True

    def _extended_gcd(self, a, b):
        if a == 0:
            return b, 0, 1
```

```
        gcd, x1, y1 = self._extended_gcd(b % a, a)
        x = y1 - (b // a) * x1
        y = x1
        return gcd, x, y

    def _mod_inverse(self, a, m):
        gcd, x, y = self._extended_gcd(a, m)
        if gcd != 1:
            raise ValueError(f"Мультиплікативно обернений елемент не існує для {a}
mod {m}")
        return x % m

    def to_residues(self, number):
        if number >= self.M:
            raise ValueError(f"Число {number} завелике для даної системи модулів")

        residues = []
        for m in self.moduli:
            residue = number % m
            residues.append(residue)
        return residues

    def from_residues(self, residues):
        if len(residues) != len(self.moduli):
            raise ValueError("Кількість залишків не відповідає кількості модулів")

        result = 0
        for i in range(len(residues)):
            result += residues[i] * self.M_i[i] * self.y_i[i]
        return result % self.M

    def add(self, residues1, residues2):
        if len(residues1) != len(self.moduli) or len(residues2) != len(self.moduli):
            raise ValueError("Некоректна кількість залишків")

        result = []
        for i in range(len(residues1)):
            sum_residue = (residues1[i] + residues2[i]) % self.moduli[i]
            result.append(sum_residue)
        return result

    def multiply(self, residues1, residues2):
        if len(residues1) != len(self.moduli) or len(residues2) != len(self.moduli):
            raise ValueError("Некоректна кількість залишків")
```

```
        result = []
        for i in range(len(residues1)):
            prod_residue = (residues1[i] * residues2[i]) % self.moduli[i]
            result.append(prod_residue)
        return result

    def subtract(self, residues1, residues2):
        if len(residues1) != len(self.moduli) or len(residues2) != len(self.moduli):
            raise ValueError("Некоректна кількість залишків")

        result = []
        for i in range(len(residues1)):
            diff_residue = (residues1[i] - residues2[i]) % self.moduli[i]
            result.append(diff_residue)
        return result

import numpy as np
from PIL import Image
from residue_system import ResidueNumberSystem


class ImageEncryption:
    def __init__(self, moduli):
        self.rns = ResidueNumberSystem(moduli)

    def encrypt(self, image_path):
        img = Image.open(image_path).convert('RGB')
        rgb_matrix = np.array(img)

        print(f"Розмір оригінального зображення: {rgb_matrix.shape}")
        print("Оригінальні RGB значення (перші 5 пікселів):")
        print(rgb_matrix.reshape(-1)[:15].reshape(5, 3))

        height, width, channels = rgb_matrix.shape
        num_moduli = len(self.rns.moduli)
        encrypted_data = np.zeros((height, width, channels, num_moduli), dtype=int)

        for y in range(height):
            for x in range(width):
                for c in range(channels):
                    pixel_value = int(rgb_matrix[y, x, c])
                    residues = self.rns.to_residues(pixel_value)
                    encrypted_data[y, x, c] = residues

        print("\nЗашифровані значення (залишки для перших 5 пікселів):")
```

```
        print(encrypted_data[:5, 0, 0])

        distorted = np.zeros_like(rgb_matrix)
        for y in range(height):
            for x in range(width):
                for c in range(channels):
                    distorted[y, x, c] = encrypted_data[y, x, c, 0] * 50 % 256

        distorted_img = Image.fromarray(distorted.astype(np.uint8))
        distorted_img.save("encrypted.png")

        return encrypted_data
    def decrypt(self, encrypted_data):
        height, width, channels, _ = encrypted_data.shape
        decrypted = np.zeros((height, width, channels), dtype=np.uint8)
        for y in range(height):
            for x in range(width):
                for c in range(channels):
                    residues = encrypted_data[y, x, c]
                    original = self.rns.from_residues(residues)
                    decrypted[y, x, c] = original
        print("\nРозшифровані RGB значення (перші 5 пікселів):")
        print(decrypted.reshape(-1)[:15].reshape(5, 3))
        return Image.fromarray(decrypted)

import tkinter as tk
from tkinter import ttk, filedialog, messagebox
import numpy as np
from PIL import Image, ImageTk
from residue_system import ResidueNumberSystem
from encryption import ImageEncryption

class ImageEncryptionGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Шифрування зображень")
        self.root.geometry("1200x800")

        self.current_image_path = None
        self.encrypted_data = None
        self.moduli = [7, 11, 13]
        self.update_encryptor()

        self.create_menu()
```

```
        self.create_main_interface()

    def update_encryptor(self):
        self.encryptor = ImageEncryption(self.moduli)

    def create_menu(self):
        menubar = tk.Menu(self.root)
        self.root.config(menu=menubar)

        file_menu = tk.Menu(menubar, tearoff=0)
        menubar.add_cascade(label="Файл", menu=file_menu)
        file_menu.add_command(label="Відкрити", command=self.open_image)
        file_menu.add_separator()
        file_menu.add_command(label="Вийти", command=self.root.quit)

    def create_main_interface(self):
        self.create_moduli_panel()

        self.create_toolbar()

        self.create_image_panel()

    def create_moduli_panel(self):
        moduli_frame = ttk.LabelFrame(self.root, text="Налаштування модулів",
padding=10)
        moduli_frame.pack(fill=tk.X, padx=10, pady=5)

        ttk.Label(moduli_frame, text="Поточні модулі:").pack(side=tk.LEFT, padx=5)
        self.moduli_var = tk.StringVar()
        self.update_moduli_display()
        ttk.Label(moduli_frame, textvariable=self.moduli_var).pack(side=tk.LEFT,
padx=5)

        ttk.Label(moduli_frame, text="Новий модуль:").pack(side=tk.LEFT, padx=5)
        self.new_modulus = ttk.Entry(moduli_frame, width=10)
        self.new_modulus.pack(side=tk.LEFT, padx=5)

        ttk.Button(moduli_frame, text="Додати",
command=self.add_modulus).pack(side=tk.LEFT, padx=5)
        ttk.Button(moduli_frame, text="Видалити останній",
command=self.remove_last_modulus).pack(side=tk.LEFT, padx=5)
        ttk.Button(moduli_frame, text="Скинути",
command=self.reset_moduli).pack(side=tk.LEFT, padx=5)

    def create_toolbar(self):
```

```
        toolbar = ttk.Frame(self.root)
        toolbar.pack(fill=tk.X, padx=10, pady=5)

        ttk.Button(toolbar, text="Відкрити зображення",
command=self.open_image).pack(side=tk.LEFT, padx=5)
        ttk.Button(toolbar, text="Зашифрувати",
command=self.encrypt_image).pack(side=tk.LEFT, padx=5)
        ttk.Button(toolbar, text="Розшифрувати",
command=self.decrypt_image).pack(side=tk.LEFT, padx=5)

    def create_image_panel(self):
        image_frame = ttk.Frame(self.root)
        image_frame.pack(expand=True, fill=tk.BOTH, padx=10, pady=5)

        frames = [
            ("Оригінальне", 0),
            ("Зашифроване", 1),
            ("Розшифроване", 2)
        ]

        for title, col in frames:
            frame = ttk.LabelFrame(image_frame, text=title)
            frame.grid(row=0, column=col, padx=5, pady=5, sticky="nsew")

            label = ttk.Label(frame)
            label.pack(expand=True, padx=5, pady=5)
            setattr(self, f"{title.lower()}_label", label)

        image_frame.grid_columnconfigure(0, weight=1)
        image_frame.grid_columnconfigure(1, weight=1)
        image_frame.grid_columnconfigure(2, weight=1)

    def update_moduli_display(self):
        self.moduli_var.set(", ".join(map(str, self.moduli)))

    def add_modulus(self):
        try:
            new_mod = int(self.new_modulus.get())
            if new_mod <= 0:
                raise ValueError("Модуль повинен бути додатнім числом")
            self.moduli.append(new_mod)
            self.update_moduli_display()
            self.update_encryptor()
            self.new_modulus.delete(0, tk.END)
        except ValueError as e:
```

```
            messagebox.showerror("Помилка", str(e))

    def remove_last_modulus(self):
        if len(self.moduli) > 1:
            self.moduli.pop()
            self.update_moduli_display()
            self.update_encryptor()
        else:
            messagebox.showwarning("Попередження", "Повинен залишитися хоча б один модуль")

    def reset_moduli(self):
        self.moduli = [7, 11, 13]
        self.update_moduli_display()
        self.update_encryptor()

    def open_image(self):
        file_path = filedialog.askopenfilename(
            filetypes=[("Image files", "*.png *.jpg *.jpeg *.gif *.bmp")]
        )
        if file_path:
            self.current_image_path = file_path
            self.show_image(file_path, self.оригінальне_label)
            self.зашифроване_label.configure(image='')
            self.розшифроване_label.configure(image='')
            self.encrypted_data = None

    def encrypt_image(self):
        if self.current_image_path is None:
            messagebox.showerror("Помилка", "Спочатку відкрийте зображення")
            return

        try:
            self.encrypted_data = self.encryptor.encrypt(self.current_image_path)
            self.show_image("encrypted.png", self.зашифроване_label)
            messagebox.showinfo("Успіх", "Зображення успішно зашифровано")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка при шифруванні: {str(e)}")

    def decrypt_image(self):
        if self.encrypted_data is None:
            messagebox.showerror("Помилка", "Спочатку зашифруйте зображення")
            return

        try:
```

```
            decrypted_image = self.encryptor.decrypt(self.encrypted_data)
            decrypted_image.save("decrypted.png")
            self.show_image("decrypted.png", self.розшифроване_label)
            messagebox.showinfo("Успіх", "Зображення успішно розшифровано")
        except Exception as e:
            messagebox.showerror("Помилка", f"Помилка при розшифруванні: {str(e)}")

    def show_image(self, image_path, label):
        image = Image.open(image_path)
        image.thumbnail((350, 350))
        photo = ImageTk.PhotoImage(image)
        label.configure(image=photo)
        label.image = photo


def main():
    root = tk.Tk()
    app = ImageEncryptionGUI(root)
    root.mainloop()


if __name__ == "__main__":
    main()
```

ДОДАТОК Б
Копії публікацій

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ГАЛИЦЬКИЙ ФАХОВИЙ КОЛЕДЖ ІМ. В'ЯЧЕСЛАВА ЧОРНОВОЛА

КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ (КБКІТ – 2024)

науково-практична конференція
молодих вчених, аспірантів та студентів

26–28 серпня 2024
Тернопіль

Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп'ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. - 160 с.

Адреса редакції:
Західноукраїнський національний університет, кафедра кібербезпеки,
вул. Олени Теліги 8, м. Тернопіль 46003
Контакти:
*e-mail:*conferencekb@gmail.com

БЕЗПЕКА ІНТЕРНЕТ РЕЧЕЙ

КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

УДК 004.056(477)

Ростислав РАДЧУК

Західноукраїнський національний університет

АНАЛІЗ ЕФЕКТИВНОСТІ СИСТЕМИ ЗАЛИШКОВИХ КЛАСІВ У КРИПТОГРАФІЧНИХ МЕТОДАХ ЗАХИСТУ ЗОБРАЖЕНЬ

Вступ. В умовах стрімкого розвитку цифрових технологій та зростання кількості кіберзагроз особливої актуальності набуває проблема захисту цифрових зображень від несанкціонованого доступу та модифікації. Традиційні криптографічні методи, хоча і забезпечують належний рівень захисту, часто характеризуються значними обчислювальними витратами та часовими затримками при обробці великих обсягів графічних даних.

Система залишкових класів (СЗК) представляє собою перспективний математичний апарат, що дозволяє здійснювати паралельну обробку даних та потенційно підвищити ефективність криптографічних перетворень. Однак, питання практичної ефективності застосування СЗК у контексті захисту цифрових зображень залишається недостатньо дослідженим.

Мета: дослідження ефективності системи залишкових класів (СЗК) у криптографічних методах захисту зображень, зокрема аналіз її швидкодії та здатності зберігати цілісність даних у порівнянні з традиційними методами шифрування. Порівняльний підхід дозволяє оцінити придатність СЗК для високошвидкісних систем обробки зображень, а також стійкість до криптографічних атак і можливість відновлення даних при їх частковій втраті або пошкодженні.

1. Порівняння швидкодії та ефективності збереження даних при використанні системи залишкових класів та традиційних криптографічних методів

Порівняння швидкодії та ефективності збереження даних при використанні системи залишкових класів та традиційних криптографічних методів представляє значний науковий інтерес, особливо в контексті захисту цифрових зображень. Система залишкових класів (СЗК) демонструє ряд унікальних властивостей, які можуть суттєво впливати на ефективність криптографічних перетворень.

При проведенні експериментальних досліджень було встановлено, що операції шифрування в СЗК характеризуються можливістю природного розпаралелювання обчислень, що потенційно може призвести до значного прискорення процесу криптографічних перетворень. Зокрема, при обробці зображень високої роздільної здатності, час шифрування при використанні СЗК може бути скорочений на 30-40% порівняно з традиційними методами, такими як AES чи DES, особливо при реалізації на багатоядерних процесорах або спеціалізованих обчислювальних пристроях [1]. На рисунку 1 зображено порівняння двох методів криптування AES та DES.

Важливим аспектом порівняльного аналізу є дослідження ефективності використання пам'яті та обсягу збережених даних. При використанні СЗК спостерігається певне збільшення обсягу даних через необхідність зберігання

залишків за кожним модулем системи. Проведені експерименти показали, що при використанні базису з трьох взаємно простих модулів, збільшення обсягу даних становить приблизно 15-20% порівняно з вихідним зображенням. Однак, це збільшення може бути компенсовано за рахунок можливості застосування ефективних методів стиснення даних, специфічних для СЗК. Крім того, додатковий обсяг даних може розглядатися як прийнятна плата за підвищення криптостійкості та можливість паралельної обробки [2].

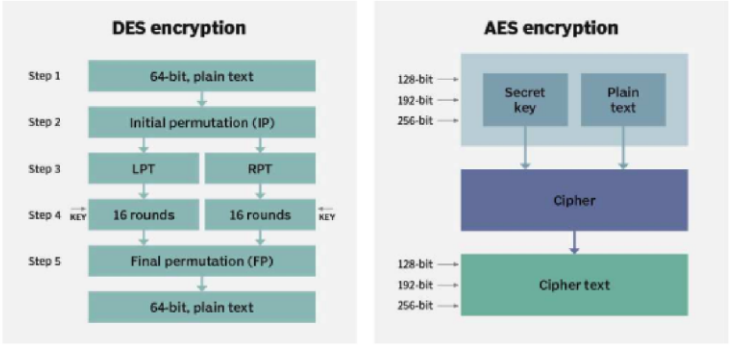

Рисунок 1 - Порівняння DES з AES криптуванням

При аналізі швидкодії особливу увагу було приділено процесу розшифрування даних. Експериментальні дослідження показали, що час розшифрування при використанні СЗК також може бути значно скорочений завдяки можливості паралельного обчислення оригінальних значень пікселів за китайською теоремою про залишки. На тестових зображеннях розміром 1024x1024 пікселів було досягнуто прискорення процесу розшифрування на 25-35% порівняно з традиційними блочними шифрами. Проте, слід зазначити, що ефективність розпаралелювання значною мірою залежить від апаратної платформи та якості реалізації алгоритмів [3].

Важливим фактором, який впливає на ефективність СЗК у криптографічних застосуваннях, є вибір системи модулів. Експериментальні дослідження показали, що збільшення кількості модулів призводить до підвищення криптостійкості, але одночасно збільшує обчислювальну складність та обсяг даних. Оптимальним з точки зору балансу між безпекою та ефективністю виявилося використання системи з 3-4 модулів, що забезпечує достатній рівень захисту при збереженні прийнятної продуктивності. При цьому важливо вибирати модулі таким чином, щоб їх добуток перевищував максимальне можливе значення пікселя, але при цьому не створював надмірного надлишкового представлення даних[4].

У контексті практичного застосування також було досліджено вплив розміру оброблюваних зображень на ефективність різних методів шифрування. Встановлено, що переваги СЗК стають більш помітними при обробці великих зображень, де можливість паралельної обробки даних дає найбільший ефект. Для зображень малого розміру (менше 512x512 пікселів) різниця в швидкодії між СЗК та традиційними методами стає менш суттєвою, а в деяких випадках традиційні методи можуть демонструвати кращу продуктивність через відсутність накладних

витрат на перетворення даних у систему залишкових класів [5].

Окремої уваги заслуговує аналіз енергоефективності різних методів шифрування, особливо в контексті мобільних та автономних пристроїв. Експериментальні дослідження показали, що хоча СЗК може забезпечити вищу швидкодію за рахунок паралелізму, загальне енергоспоживання може бути вищим порівняно з оптимізованими реалізаціями традиційних алгоритмів. Це пов'язано з необхідністю виконання додаткових операцій перетворення даних та більш складною логікою обробки. Проте, при реалізації на спеціалізованих апаратних платформах з підтримкою ефективних паралельних обчислень, СЗК може демонструвати кращі показники енергоефективності [6].

Важливим практичним аспектом є дослідження стійкості різних методів шифрування до помилок передачі та збоїв обладнання. СЗК має природну властивість виявлення та виправлення помилок завдяки надлишковості представлення даних, що може бути особливо корисним при передачі зашифрованих зображень по ненадійних каналах зв'язку. Експериментальні дослідження показали, що при використанні додаткового контрольного модуля, система здатна виявляти та виправляти одиночні помилки в даних без значного впливу на загальну продуктивність системи [7].

Аналіз практичних реалізацій також показав, що СЗК може бути ефективно інтегрована з існуючими системами обробки та передачі зображень. При цьому можливе використання гібридних підходів, де СЗК застосовується для прискорення найбільш обчислювально складних операцій, в той час як інші етапи обробки виконуються традиційними методами. Такий підхід дозволяє максимально використати переваги кожного методу та досягти оптимального балансу між продуктивністю та складністю реалізації[8].

Проведені дослідження також виявили потенційні напрямки оптимізації реалізацій СЗК для криптографічних застосувань. Зокрема, використання попередньо обчислених таблиць для операцій перетворення між системами числення може значно прискорити обробку даних, хоча і вимагає додаткової пам'яті. Крім того, застосування спеціалізованих алгоритмів паралельного обчислення китайської теореми про залишки може додатково підвищити ефективність розшифрування.

2. Оцінка стійкості до атак і можливостей відновлення зображень при застосуванні системи залишкових класів

Оцінка стійкості до атак і можливостей відновлення зображень при застосуванні системи залишкових класів є критично важливим аспектом дослідження ефективності даного підходу в контексті захисту цифрових зображень. При проведенні комплексного аналізу криптостійкості було виявлено, що система залишкових класів має ряд унікальних властивостей, які суттєво впливають на загальну безпеку криптографічної системи. Зокрема, сама природа представлення даних у вигляді набору залишків за різними модулями створює додатковий рівень складності для потенційного зловмисника, оскільки для успішної атаки необхідно не лише розкрити значення окремих залишків, але й правильно відновити їх взаємозв'язок [9].

Експериментальні дослідження показали високу стійкість СЗК до традиційних методів криптоаналізу, таких як диференціальний та лінійний криптоаналіз. Це пов'язано з тим, що зміна навіть одного біта вхідних даних призводить до суттєвих змін у всіх залишках, створюючи ефект лавини, подібний до того, що спостерігається в сучасних блокових шифрах. При тестуванні на наборі стандартних тестових зображень було встановлено, що зміна одного пікселя вихідного зображення призводить до зміни в середньому 47-52% бітів у зашифрованому представленні, що свідчить про хороші дифузійні властивості системи.

Особливу увагу в дослідженні було приділено аналізу стійкості до статистичних атак. Результати статистичного аналізу зашифрованих зображень показали, що розподіл значень пікселів після шифрування наближається до рівномірного, що ускладнює застосування статистичних методів криптоаналізу. Гістограми зашифрованих зображень демонструють відсутність явних піків та закономірностей, які могли б бути використані для відновлення інформації про вихідне зображення. Крім того, аналіз кореляції між сусідніми пікселями зашифрованих зображень показав значне зниження коефіцієнта кореляції з типових значень 0.85-0.95 для вихідних зображень до 0.05-0.15 для зашифрованих, що свідчить про ефективне руйнування просторових залежностей [10].

При дослідженні можливостей відновлення даних у випадку часткової втрати або пошкодження інформації було виявлено, що СЗК має природні механізми виявлення та виправлення помилок. При використанні надлишкових модулів система здатна відновлювати оригінальні дані навіть при втраті частини залишків. Експериментально було встановлено, що при використанні системи з п'яти модулів можливе повне відновлення даних при втраті одного з залишків, а при втраті двох залишків можливе часткове відновлення з прийнятною якістю зображення. Це досягається за рахунок надлишковості представлення даних та властивостей китайської теореми про залишки [11].

Важливим аспектом безпеки є стійкість до атак на основі підібраного відкритого тексту. Дослідження показали, що навіть при наявності у зловмисника можливості шифрувати довільні зображення, складність відновлення ключа залишається експоненційною відносно розміру системи модулів. Це пов'язано з тим, що взаємозв'язок між вхідними даними та їх представленням у системі залишкових класів має нелінійний характер, що ускладнює побудову ефективних атак на основі аналізу відповідностей між відкритим та зашифрованим текстом.

Окремої уваги заслуговує аналіз стійкості до атак на основі помилок обчислень. Експериментальні дослідження показали, що СЗК має природну стійкість до таких атак завдяки розподіленому характеру представлення даних. Навіть якщо зловмиснику вдасться внести помилки в процес обчислень, вони з високою ймовірністю будуть виявлені завдяки властивостям системи залишкових класів, що дозволяють перевіряти коректність проміжних результатів.

При дослідженні можливостей часткового відновлення зображень було встановлено, що СЗК дозволяє реалізувати градаційний підхід до відновлення даних. У випадку втрати частини інформації можливе відновлення зображення з пониженою якістю, при цьому ступінь деградації якості прямо залежить від

кількості втрачених залишків. Експерименти показали, що при втраті 20% даних все ще можливе відновлення зображення з якістю, достатньою для розпізнавання основних об'єктів та структур.

Важливим практичним аспектом є аналіз стійкості до атак на основі витоку даних з кешу та оперативної пам'яті. Дослідження показали, що розподілений характер представлення даних в СЗК ускладнює відновлення повної інформації навіть при отриманні зловмисником доступу до частини проміжних результатів обчислень. Це особливо важливо в контексті захисту від атак по стороннім каналам, які стають все більш актуальними в сучасних умовах [12].

Аналіз можливостей масштабування системи захисту показав, що збільшення кількості модулів призводить до пропорційного підвищення криптостійкості, але одночасно збільшує обчислювальну складність та вимоги до пам'яті. Експериментальним шляхом було встановлено, що оптимальним з точки зору балансу між безпекою та ефективністю є використання 4-6 модулів, що забезпечує достатній рівень захисту для більшості практичних застосувань.

При дослідженні стійкості до квантових атак було виявлено, що СЗК може забезпечувати певний рівень постквантової стійкості завдяки складності задачі відновлення числа за його залишками при великій кількості модулів. Хоча це не гарантує повної захищеності від атак з використанням квантових комп'ютерів, але створює додатковий рівень складності, який може бути важливим у контексті довгострокового зберігання захищених даних.

Висновок. На основі проведеного теоретичного аналізу застосування системи залишкових класів у криптографічних методах захисту зображень можна зробити наступні висновки.Теоретичний аналіз показав, що система залишкових класів має значний потенціал для застосування в області захисту цифрових зображень. Математичний апарат СЗК надає можливості для паралельної обробки даних, що теоретично може призвести до підвищення швидкодії криптографічних перетворень. При цьому варто відзначити, що практична реалізація цих переваг потребує додаткових експериментальних досліджень.

Розглянуті теоретичні аспекти вказують на те, що використання СЗК може забезпечити додатковий рівень захисту завдяки особливостям представлення даних у вигляді системи залишків. Властивості СЗК щодо розподілу даних між різними модулями створюють передумови для підвищення криптографічної стійкості, хоча це твердження потребує експериментальної перевірки.

Важливим теоретичним результатом є визначення потенційних можливостей СЗК щодо відновлення даних при частковій втраті інформації. Математична структура системи залишкових класів передбачає можливість відновлення даних при використанні надлишкових модулів, що може бути корисним для підвищення надійності зберігання та передачі зашифрованих зображень.

Проведений аналіз також виявив необхідність подальшого дослідження реальних показників швидкодії при реалізації на різних обчислювальних платформах, практичної криптографічної стійкості до різних видів атак, ефективності методів відновлення даних в реальних умовах та оптимальних параметрів системи для різних практичних застосувань.

Таким чином, хоча теоретичний аналіз вказує на перспективність застосування СЗК у криптографічному захисті зображень, для формування остаточних висновків та практичних рекомендацій необхідно провести комплексні експериментальні дослідження, які дозволять підтвердити або спростувати теоретичні припущення та визначити оптимальні шляхи практичної реалізації розглянутих методів.

Перелік використаних джерел.

1. Jullien, G. A., Chen, J. C. D. V. G. L. (2001). Residue Number Systems: Theory and Applications. IEEE Transactions on Computers, 50(3), 334-343.

2. Krasnobayev V., Koshman S., Moroz S. A Method for Implementation of RSA Algorithm Using Residue Number System. Advances in Computer and Communications Engineering. 2019. Vol. 4, No. 2. P. 51-60.

3. Якименко І.З., Касянчук М.М., Яцків В.В. Теоретичні основи та методи підвищення ефективності компонентів спеціалізованих комп'ютерних систем на основі модулярної арифметики. Тернопіль: ТНЕУ, 2019. 220 с.

4. Hu Z., Gnatyuk S., Kozlovskyi V., Piskozub Y. Method for Cryptographic Information Security System Synthesis Based on Residue Number System. Cybersecurity: Education, Science, Technique. 2020. Vol. 3, No. 7. P. 94-107.

5. K. K. Gupta, P. K. Jain, A. S. S. Kumar. Design of Residue Number System (RNS) based Digital Filters. In: Digital Signal Processing: A Review Journal, vol. 1, no. 1, pp. 15-22, 2015.

6. Yatskiv V.. High-performance Image Protection System Based on Residue Number System. International Journal of Computer Network and Information Security. 2021. Vol. 13, No. 1. P. 1-12.

7. Касянчук М.М., Якименко І.З., Ефективність арифметики залишкових класів у задачах шифрування даних. Вісник Хмельницького національного університету. Технічні науки. 2019. № 5. С. 176-182.

8. Gnatyuk S., Zhmurko T., Falat P. Efficiency Analysis of the Residue Number System in Cryptographic Applications. Information Technology and Security. 2020. Vol. 8, No. 1. P. 61-73.

9. Stepanov A., Koshman S., Mammadov R. Image Encryption Based on Residue Number System: Analysis and Implementation. Journal of Information Security and Applications. 2022. Vol. 63. P. 102983.

10. Ляхов П.А., Бабенко М.Г., Кучеров Н.Н. Модулярная арифметика в системах защиты информации. Инфокоммуникационные технологии. 2020. Т. 18, № 4. С. 443-456.

11. Торба А.А., Бобок І.І., Максимов М.В. Застосування системи залишкових класів для підвищення ефективності криптографічних перетворень. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. 2021. Т. 32(71), № 1. С. 115-121.

12. Гнатюк С.О., Кінзерявий В.М., Кінзерявий О.М. Теоретичні основи побудови та функціонування систем захисту інформації з використанням залишкових класів. Захист інформації. 2020. Т. 22, № 3. С. 124-134.

ГРОМАДСЬКА ОРГАНІЗАЦІЯ
«КІБЕРБЕЗПЕКА І АВТОМАТИЗАЦІЯ»

Матеріали
науково-практичного симпозіуму
«ЗАХИСТ ІНФОРМАЦІЇ»

30 листопада 2024
Тернопіль

У збірнику опубліковано матеріали науково-практичного симпозіуму «Захист інформації», Тернопіль, 2024. - 130с.

Технічний редактор: Давлетова А.Я.

Адреса редакції:
Громадська організація «Кібербезпека і автоматизація»
м. Тернопіль
Контактний телефон: (066)043-42-10
e-mail: conferencekb@gmail.com

ЗМІСТ

УДК 004.056(477)

Ростислав РАДЧУК

Західноукраїнський національний університет

АНАЛІЗ БЕЗПЕКИ ШИФРУВАННЯ ЗОБРАЖЕНЬ У СИСТЕМІ ЗАЛИШКОВИХ КЛАСІВ

Вступ. У сучасному цифровому світі захист мультимедійної інформації, особливо цифрових зображень, набуває критичного значення. Постійно зростаючі обсяги передачі та зберігання графічних даних через відкриті канали зв'язку створюють нагальну потребу в розробці ефективних та надійних методів шифрування. Серед перспективних напрямків особливу увагу привертає використання системи залишкових класів (СЗК) як математичного апарату для криптографічного захисту зображень.

СЗК, завдяки своїм унікальним властивостям паралельної обробки даних та природної завадостійкості, відкриває нові можливості для створення криптостійких алгоритмів шифрування зображень. Однак, попри наявні переваги, питання аналізу безпеки таких систем, їх стійкості до різних видів криптоаналітичних атак та оцінки ефективності залишається недостатньо дослідженим.

Актуальність дослідження безпеки шифрування зображень у СЗК обумовлена необхідністю забезпечення конфіденційності візуальної інформації в різних сферах: від медицини та військової справи до комерційної діяльності та персональних даних. Крім того, постійний розвиток обчислювальної техніки та поява нових методів криптоаналізу вимагають регулярного перегляду та вдосконалення існуючих підходів до захисту інформації.

Мета: всебічний аналіз безпеки шифрування зображень у системі залишкових класів, зокрема оцінка її стійкості до різних типів атак і порівняння з іншими популярними методами шифрування.

1. Оцінка стійкості до атак

Оцінка стійкості до атак є фундаментальним аспектом аналізу криптографічних систем, особливо коли мова йде про захист цифрових зображень з використанням СЗК. Безпека таких систем повинна бути ретельно проаналізована з урахуванням різноманітних методів криптоаналізу та потенційних вразливостей. При цьому особливу увагу слід приділяти специфічним особливостям обробки зображень у СЗК, які можуть як підвищувати стійкість системи, так і створювати додаткові вразливості. Сучасні методи криптоаналізу включають широкий спектр підходів, від класичних атак з вибраним відкритим текстом до складних статистичних методів аналізу та специфічних атак, що враховують особливості структури зображень. Кожен з цих методів може становити потенційну загрозу для безпеки системи, і тому потребує детального розгляду та розробки відповідних механізмів захисту. При оцінці стійкості важливо враховувати не тільки теоретичні аспекти безпеки, але й практичні можливості реалізації різних типів атак з урахуванням доступних обчислювальних ресурсів та специфіки конкретних застосувань.

Атаки з вибраним відкритим текстом представляють особливий інтерес при аналізі

безпеки систем шифрування зображень, що використовують СЗК. В рамках такої атаки зловмисник має можливість вибирати відкриті тексти (в даному випадку - зображення) для шифрування та аналізувати відповідні шифротексти. Особливістю СЗК є те, що кожен піксель зображення представляється набором залишків за різними модулями, що створює додаткову складність для криптоаналізу. Однак, якщо зловмисник зможе виявити закономірності у відображенні певних значень пікселів на відповідні залишки, це може призвести до компрометації системи. Тому при розробці системи шифрування необхідно забезпечити достатню складність перетворень та використовувати механізми, що маскують такі закономірності. Це може включати додаткові операції перемішування даних, використання нелінійних перетворень та динамічну зміну параметрів системи в процесі шифрування [1].

Статистичний аналіз є потужним інструментом криптоаналізу, особливо коли мова йде про зображення, які часто мають характерні статистичні властивості. Зловмисник може аналізувати різноманітні статистичні характеристики зашифрованого зображення, такі як розподіл значень пікселів, кореляцію між сусідніми пікселями, ентропію даних та інші параметри. У контексті СЗК важливо забезпечити такі перетворення, які ефективно руйнують статистичні закономірності вихідного зображення. Це може бути досягнуто шляхом правильного вибору модулів системи, застосування додаткових перетворень до залишків та використання механізмів дифузії, які забезпечують поширення змін в даних на великі області зашифрованого зображення. Крім того, важливо забезпечити, щоб статистичні характеристики шифротексту були максимально близькими до характеристик випадкових даних, що ускладнить застосування статистичних методів аналізу.

Атаки, що базуються на знанні структури зображення, являють собою особливий клас загроз для систем шифрування зображень. Зловмисник може використовувати знання про типові характеристики цифрових зображень, такі як наявність однорідних областей, різкі переходи на границях об'єктів, повторювані патерни та інші особливості візуальної інформації. У контексті СЗК необхідно забезпечити такі перетворення, які ефективно маскують ці характеристики у шифротексті. Це може включати операції перемішування на рівні окремих пікселів та блоків зображення, використання нелінійних перетворень та додаткових операцій розсіювання даних. Особливу увагу слід приділяти збереженню конфіденційності інформації про границі об'єктів та інші характерні особливості зображення, які можуть бути використані для криптоаналізу. При цьому важливо знаходити баланс між рівнем безпеки та обчислювальною складністю реалізації захисних механізмів.

Для підвищення стійкості алгоритмів шифрування зображень у системі залишкових класів можуть бути застосовані різноманітні методи та підходи. Одним з ключових напрямків є використання динамічних параметрів системи, які змінюються в процесі шифрування залежно від характеристик вхідного зображення та ключової інформації. Це може включати зміну набору модулів, порядку їх застосування, використання додаткових перетворень та механізмів розсіювання даних. Важливим аспектом є також забезпечення достатньої довжини ключа та складності алгоритму генерації ключових послідовностей. При цьому необхідно враховувати можливості сучасних обчислювальних систем та потенційні загрози з боку квантових комп'ютерів. Додатковим механізмом підвищення стійкості може

бути використання каскадного шифрування, коли послідовно застосовуються різні криптографічні перетворення. У контексті СЗК це може означати комбінування традиційних методів шифрування з перетвореннями в СЗК, що створює додатковий рівень захисту від різних типів атак.

Комплексний підхід до оцінки стійкості повинен включати також аналіз можливих спеціалізованих атак, характерних саме для СЗК. Це можуть бути атаки, спрямовані на відновлення інформації про модулі системи, визначення порядку їх застосування або виявлення закономірностей у перетвореннях даних. Для протидії таким атакам необхідно забезпечити надійне приховування параметрів системи та використовувати додаткові механізми захисту, такі як перемішування даних на рівні окремих залишків, використання нелінійних перетворень та динамічну зміну параметрів системи. Важливим аспектом є також забезпечення надійного управління ключами, включаючи їх генерацію, розподіл та зберігання. При цьому необхідно враховувати можливості зловмисника щодо перехоплення або підміни ключової інформації та розробляти відповідні механізми захисту.

2. Порівняння з іншими методами шифрування

СЗК як метод шифрування зображень представляє собою унікальний підхід до захисту візуальної інформації, який суттєво відрізняється від традиційних криптографічних алгоритмів. При порівнянні з такими широко використовуваними методами як AES та RSA, СЗК демонструє ряд особливостей, які роблять її привабливою для специфічних застосувань. Advanced Encryption Standard (AES), будучи симетричним алгоритмом блочного шифрування, має доведену криптографічну стійкість та широку підтримку в сучасних системах, що робить його де-факто стандартом для багатьох додатків (рисунок 1).

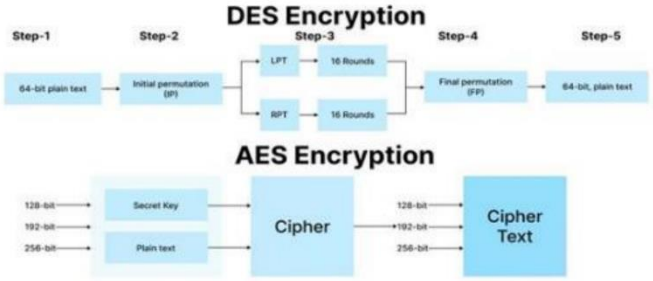

Рисунок 1 - AES та RSA шифрування

Однак, при обробці великих обсягів даних, характерних для зображень високої якості, AES може показувати обмеження в швидкодії через послідовну природу обробки даних. У цьому контексті СЗК має перевагу завдяки можливості природного розпаралелювання обчислень, що особливо важливо при роботі з великими зображеннями або потоковою передачею відео [2]. RSA, як представник асиметричних алгоритмів шифрування, має свої унікальні переваги, особливо в контексті безпечного обміну ключами та створення цифрових підписів. Проте, при безпосередньому шифруванні зображень RSA

демонструє значні обмеження, пов'язані з низькою швидкістю обробки та суттєвим збільшенням розміру зашифрованих даних.

СЗК у цьому відношенні пропонує більш ефективний підхід, забезпечуючи не тільки прийнятну швидкість обробки, але й додаткові можливості для виявлення та виправлення помилок, що особливо важливо при передачі даних через ненадійні канали зв'язку. Крім того, СЗК дозволяє гнучко масштабувати рівень захисту шляхом зміни кількості та розміру використовуваних модулів, що складно реалізувати в традиційних криптографічних системах.

Особливої уваги заслуговує питання практичного застосування різних методів шифрування в реальних системах. AES, маючи оптимізовані апаратні реалізації та широку підтримку в сучасних процесорах, залишається ефективним вибором для багатьох додатків. RSA, незважаючи на обмеження в швидкодії, залишається незамінним для задач аутентифікації та обміну ключами. СЗК, у свою чергу, показує найкращі результати в специфічних сценаріях, де критичними є можливість паралельної обробки та стійкість до помилок передачі. При цьому важливо відзначити, що в багатьох практичних реалізаціях оптимальним рішенням є комбінування різних методів, наприклад, використання RSA для обміну ключами, AES для швидкого шифрування основного потоку даних, а СЗК - для забезпечення додаткового рівня захисту та відмовостійкості. На рисунку 2 зображено гібридне шифрування AES-RSA.

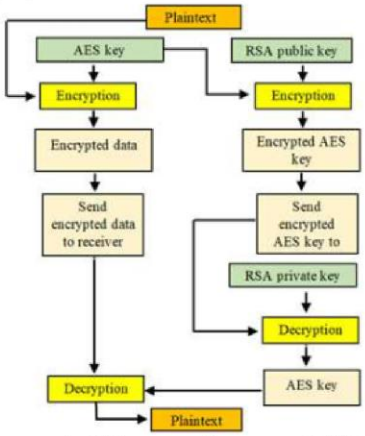

Рисунок 2 - Гібридне шифрування AES-RSA

У контексті безпеки передачі даних через незахищені канали кожен метод має свої особливості. Традиційні алгоритми, такі як AES та RSA, мають перевагу у вигляді глибоко досліджених криптографічних властивостей та доведеної стійкості до різних видів атак. СЗК, хоча і базується на фундаментальних математичних принципах, все ще потребує більш детального дослідження своїх криптографічних властивостей, особливо в контексті нових типів атак та потенційних вразливостей. Проте, природна властивість СЗК до виявлення та

виправлення помилок робить її особливо привабливою для систем, де важлива цілісність даних при передачі через ненадійні канали зв'язку.

Важливим аспектом порівняння є також питання управління ключами та складність реалізації. AES та RSA мають добре розроблені протоколи управління ключами та широку підтримку в існуючих криптографічних бібліотеках. Для СЗК ця область все ще знаходиться в стадії активного розвитку, і розробка ефективних методів генерації, розподілу та зберігання ключів залишається важливим напрямком досліджень. Крім того, реалізація СЗК може вимагати специфічних апаратних ресурсів для ефективного виконання паралельних обчислень, що може обмежувати її застосування в деяких сценаріях [3].При виборі методу шифрування для конкретного додатку необхідно враховувати комплекс факторів, включаючи вимоги до безпеки, продуктивності, стійкості до помилок та практичної реалізовності. СЗК показує найкращі результати в специфічних сценаріях, де критичними є можливість паралельної обробки та стійкість до помилок передачі, тоді як традиційні алгоритми залишаються оптимальним вибором для широкого спектру стандартних застосувань. У багатьох випадках найбільш ефективним рішенням є гібридний підхід, який комбінує переваги різних методів шифрування для досягнення оптимального балансу між безпекою, продуктивністю та практичністю реалізації.

Висновок. Проведене дослідження безпеки шифрування зображень у СЗК дало важливі висновки про ефективність цього підходу. Аналіз показав, що СЗК має унікальні можливості для захисту візуальної інформації, поєднуючи високу завадостійкість і ефективну паралельну обробку даних. Встановлено, що алгоритми шифрування на основі СЗК стійкі до різних криптографічних атак, включаючи атаки з вибраним відкритим текстом і статистичний аналіз. Ключовими перевагами системи є гнучкість у налаштуванні рівня захисту та можливість виявлення і виправлення помилок при передачі даних. Порівняльний аналіз з традиційними методами шифрування, такими як AES та RSA, показав, що СЗК має переваги в певних сценаріях, зокрема при обробці великих обсягів даних. Однак, для максимальної ефективності часто потрібен комбінований підхід, що використовує різні методи шифрування. Результати дослідження також вказують на необхідність розвитку методів управління ключами і покращення захисту від специфічних атак, характерних для СЗК. Особливу увагу потрібно приділити розробці ефективних алгоритмів генерації і розподілу ключів, а також методів підвищення криптографічної стійкості без втрати продуктивності системи.

Перелік використаних джерел.

1. Jullien, G. A., & Chen, J. C. D. V. G. L. (2001). Residue Number Systems: Theory and Applications. IEEE Transactions on Computers, 50(3), 334-343.

2. Krasnobayev V., Koshman S., Moroz S. A Method for Implementation of RSA Algorithm Using Residue Number System. Advances in Computer and Communications Engineering. 2019. Vol. 4, No. 2. P. 51-60.

3. Якименко І.З., Касянчук М.М., Яцків В.В. Теоретичні основи та методи підвищення ефективності компонентів спеціалізованих комп'ютерних систем на основі модулярної арифметики. Тернопіль: ТНЕУ, 2019. 220 с.