Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дзюбенко Д.О.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Дзюбенку Дмитру Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 47-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Марченко К.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Дзюбенко Д.О.
(прізвище та ініціали)

АНОТАЦІЯ

Дзюбенко Д.О. Програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Метою розробки є програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Результат роботи – програмна реалізація системи забезпечення авторських прав за допомогою цифрових водяних знаків.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, цифрові водяні знаки

ABSTRACT

Dzyubenko D.O. Software for the copyright enforcement system using digital watermarks. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the copyright enforcement system using digital watermarks.

The purpose of the development is software for the copyright enforcement system using digital watermarks.

The result of the work is the software implementation of the copyright enforcement system using digital watermarks.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, digital watermarks

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 12
2.3 Розгорнута постановка завдання 17
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 19
3.1 Опис функціонування системи 19
3.2 Розробка структурної схеми..... 27
3.3 Розробка функціональної схеми 41
3.4 Розробка діаграми процесів..... 46
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 48
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 48
4.2 Захист розробленого програмного забезпечення..... 59
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 64
6 ОСНОВНІ ВИСНОВКИ..... 68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВДТ	–	відео-дисплейні термінали
ЕОМ	–	електронно-обчислювальна машина
ЕЦП	–	електронний цифровий підпис
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
СБ	–	служба безпеки
ТЗ	–	технічне завдання
ЦВЗ	–	цифрові водяні знаки
DES	–	стандарт шифрування США
DSA	–	Digital Signature Algorithm
ECDSA	–	Elliptic Curve Digital Signature Algorithm
EGSA	–	El Gamal Signature Algorithm
IDEA	–	International Date Encryption Algorithm – алгоритм шифрування
IP	–	Internet Protocol
LSB	–	Least Significant Bits – метод стеганографії
PGP	–	Pretty Good Privacy – міжнародний криптографічний стандарт
RSA	–	алгоритм асиметричного шифрування
SHA-1	–	Secure Hash Algorithm 1 – алгоритм криптографічного хешування
TDES	–	Triple DES –модифікація DES з трьома незалежними підключами
JPEG	–	Joint Photographic Experts Group – растровий формат зображення

ВСТУП

Актуальність теми. Все частіше, особливо на сайтах в Інтернеті, можна бачити цифрові зображення із вкладеними цифровими водяними знаками (ЦВЗ). Вони являють собою напівпрозорі логотипи фірм, назви сайтів, прізвища авторів і т.д. Використання ЦВЗ дозволяє власникам і авторам цифрових рисунків і фото захищати свої авторські права на дану інтелектуальну власність і гарантувати собі гонорари за копії. Треба сказати, що ЦВЗ не завжди видимо людському оку. Залежно від області застосування, вкладення намагаються в більшому або меншому ступені сховати в копію, щоб не давати зайвої інформації про способи захисту авторських прав.

Однак, на жаль, деякі користувачі намагаються незаконно поширити продукти й залишитися при цьому непоміченими. Подібних людей називають «піратами». Завдання таких зловмисників складається у видаленні ЦВЗ із копії й подальший продаж нелегального екземпляра або використанні його у своїх цілях. Такі дії піратів класифікуються як атака на систему ЦВЗ.

Відомо велику кількість типів атак і методів видалення ЦВЗ. Їхнє детальне вивчення дозволяє успішно боротися з деякими з них. Одним з найбільш ефективних способів одержання нелегальної копії зображення без ЦВЗ (або точніше з неможливістю його виділення власником продукту) є коаліційна атака, коли кілька несумлінних одержувачів копій авторського продукту з тими, що знаходяться в них ЦВЗ, намагаються об'єднати свої ресурси й обробити їх так, щоб у результаті одержати продукт, виділення ЦВЗ із якого його законним автором виявиться неможливим.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем забезпечення авторських прав за допомогою цифрових водяних знаків.

– Дослідження системи забезпечення авторських прав за допомогою цифрових водяних знаків.

– Програмна реалізація системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі забезпечення авторських прав за допомогою цифрових водяних знаків.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Метою роботи є розробка ефективного способу захисту від коаліційних атак на системи ЦВЗ, вкладених у цифрові зображення. Для досягнення заданої мети в бакалаврській роботі пропонуються рішення наступних приватних завдань:

1. Дослідження ефективності використання широкополосних сигналів (ШПС), використовуваних як цифрові відбитки пальців (ЦВП) в умовах коаліційних атак.

2. Дослідження можливості підвищення стійкості ЦВП до коаліційних атак з використанням антикоаліційних кодів (АКК) на основі послідовностей, формованих за правилами неповних збалансованих блок-схем (НЗБС).

3. Використання нового класу диз'юнктних кодів в евклідовому просторі (ДКЕ), замість регулярних послідовностей, таких, як ортогональні сигнали, а також застосування випадкових ДКЕ для захисту від антикоаліційних атак.

1.2 Область застосування

Проблема захисту авторських прав придбала особливу актуальність із розвитком цифрових технологій і Інтернету. Незважаючи на постійне вдосконалювання законодавства, у тому числі міжнародного, у сфері авторського права, а також безперервне вдосконалювання й ускладнення технічних і програмних засобів, спрямованих на запобігання порушень у цій області, піратське використання інтелектуальної власності стало масовим явищем, збиток від якого в глобальному масштабі перевищує багато мільярдів доларів. Украй актуальним завдання захисту авторських прав є для тих, чия робота зв'язана зі

ЗМІ, як друкованими, так і електронними, а також рекламною діяльністю. Як тільки текст, фотографія або відеозапис представлені привселюдно, автор або правовласник практично втрачає контроль над можливим несанкціонованим використанням матеріалу. Для всіх очевидна гострота цієї проблеми, коли мова йде про розміщення тих або інших робіт в Інтернеті, однак на практиці дуже часто грубі порушення авторського права мають місце й тоді, коли представлений у вигляді цифрового файлу матеріал надається, наприклад, у видавництво або прямо в друкарню. У випадку, коли крім законної публікації з'являються й несанкціоновані, природно, без укладання договору з автором або правовласником, без виплати відповідної винагороди за використання роботи, а найчастіше, навіть без вказівки авторства, у законного власника прав на піратськи використаний матеріал залишається, практично, тільки один шлях – шлях судового розгляду. Судовий дозвіл такої ситуації, як правило, вимагає чималих зусиль позивача й займає багато часу, у зв'язку із чим, нерідко законні власники авторських прав воліють ігнорувати ті, які мають місце порушення, незважаючи на матеріальні, а можливо, і репутаційні втрати. Крім того, для того, щоб подати позов у суд, насамперед, необхідно виявити сам факт порушення авторського права, що далеко не завжди можливо, тому що встежити за всіма публікуємими матеріалами, включаючи Інтернет-публікації, зовсім нереально. У цьому зв'язку найбільш надійним шляхом захисту авторських прав варто визнати використання спеціальних технічних і програмних засобів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Оцифровані кольорові зображення займають багато місця на диску, тому для їх звичайно стискають. Існують алгоритми стиску без втрат, до них відносяться BMP, PNG і GIF. Ця методика дозволяє точно відновити вихідне зображення. Крім цього існують алгоритми стиску із втратами – до них ставиться самий популярний на сьогоднішній день формат JPEG. Він сильно зменшує кількість місця, займана фотографією, але вносить у неї непомітні людському оку зміни. Як використовувати стеганографію в JPG, які програми й методи для цього використовуються розповідає ця стаття.

Особливість методів стеганографії в JPEG у тім, що класичні способи приховання інформації в зображеннях, такі наприклад, як метод LSB у чистому виді для формату стиску із втратами не спрацює – всі мікроколивання кольорів будуть вирізані шумоподавленням. Тому в JPEG-файлах можуть використовуватися наступна стеганографія:

– дописування даних приховуваної інформації в кінець файлу – ну, це взагалі класика жанру, 90 % всіх програм для стеганографії використовують цей метод.

– приховання інформації в непрямих даних файлу для приховання зашифрованої інформації як правило використовується поле коментарю в exif заголовку;

– приховання інформації з використанням таблиць квантування – властиво саме цей метод ближче всього до справжньої стеганографії;

– приховання інформації між блоками даних файлу.

Існує не так багато стеганографічних програм, які дійсно працюють із JPG, хоча в цей час це найбільше широко використовуваний формат для зберігання зображень. Незважаючи на те, що багато програм декларують реалізацію JPG стеганографії, насправді вони використовують, як правило, додавання даних у кінець файлу, або приховують дані в поле «коментар» заголовка файлу. Реальна JPG стеганографія має на увазі змішування схованої інформації повідомлення з фактичною інформацією пікселів. Для роботи вони звичайно використовують бібліотеку Independent JPEG Group S '(IJG), що є найбільш потужною на сьогоднішній день.

Відзначимо, що багато хто з методів уважаються нестійкими, тобто стегоаналітик з використанням статистичних критеріїв може виявити факт приховання інформації.

Незважаючи на це, програми для стеганографії в JPEG-файлах забезпечують досить високу ступенем захищеності від стегоанализу, тому що можливість варіювання якості стислого зображення в широкому діапазоні не дозволяє легко встановити, чи є виникаючі в результаті стиску погрішності наслідком приховання даних або наслідком використання високих коефіцієнтів квантування.

Програми для стеганографії в JPEG

Зараз найпоширеніші чотири програми стеганографії, що працюють під керування Windows:

– F5 , що був розроблений у рамках досліджень, що спеціалізуються на стеганографії. Імовірно, одна із самих передових програм, що перебувають у публічному доступі. Програма використовує різні методи, щоб компенсувати зміни статистичних характеристик контейнера при внесенні змін, що утрудняє статистичний аналіз. Доступні вихідні коди мовою програмування Java.

– JPHide / JPSeek / JPHSWin – існує починаючи з 1999 року, вихідний код доступний. Програма використовує Blowfish, як генератор псевдо-випадкових чисел.

– JSteg – одна з перших програм для приховання даних у форматі JPEG. Програма не підтримує шифрування приховуваних даних. Графічний інтерфейс, що називається JStegShell, був написаний пізніше й додає кілька додаткових можливостей, такі як можливість шифрування. Вихідні коди доступні.

– StegHide нова відносно свіжа програма з відкритим вихідним кодом, підтримувана розроблювачами

Принцип роботи стеганографії (на прикладі програми JSteg)

Для демонстрації роботи стеганографічного алгоритму розглянемо докладніше програму JSteg. Вона була створена, на базі IJG бібліотеки, шляхом додавання до неї нової опції ("-steg ім'я файлу"), що виконується файлу. Так, забув сказати, програма ця консольна й зараз уважається застарілої, оскільки вона не використовує шифрування. Якщо використовувати консоль по який те причині для Вас важко, використовуєте JStegShell. Це GUI для JSteg зручний і простій у використанні. Однак, щоб зрозуміти принципи стеганографії в JPEG вона відмінно підійде. Давайте вже подивимося, як можна витягати сховані дані з формату JPG.

Спочатку програма JSteg перетворить приховуваний файл у байтову послідовність дуже простим способом. Він додає ім'я приховуваного файлу (якщо ви вибрали цю опцію) у початок. Потім, після ім'я схованого файлу, він додає байт із довжиною назви файлу. І, нарешті, наприкінці як підпис, додається рядок korejwa (ім'я автора).

Хоча JSteg використовує весь комплекс угод формату JPG, він виконує стеганографію дуже простим способом, використовуючи для приховання даних DCT коефіцієнти. За замовчуванням, DCT коефіцієнти, які дорівнюють нулю або одиниці, не змінюються. Інші можуть бути використані для вбудовування в них одного біта схованої інформації, шляхом перезапису з використанням алгоритму заміни найменш значущого біта (LSB). Приховувана

інформації має наступний формат (узятий з файлу README, і небагато спрощено):

ABBB ... BCCCCCCCCCCCCCCCCCC. . .

де область:

– A – довжиною в 5 біт. Він виражає довжину (у бітах) поля B;

– B – деяка кількість біт, певна значенням, що втримується в області A. Кількість бітів може варіюватися в діапазоні від 0 до 31. Він виражає довжину (у байтах) приховуваного повідомлення;

– C – містить саме приховуване повідомлення.

Таким чином, для роботи зі схованому повідомленням необхідно витягти з перших п'яти коефіцієнтів DCT молодші біти, щоб одержати розмір розмір (поле "A"), потім одержати з наступного біта мають розмір схованих даних (поле "B"). Після цього, можливо добування схованих даних (поля "C"), і реконструкції байт із витягнутих біт.

Для аналізу роботи програми JSteg сховаємо інформацію в зображенні обкладинки "Планета історій", збірника науково фантастичних розповідей. Це невелике JPEG зображення, розміром 15 Kb. На рисунку наведені зображення до й після приховання інформації. Зображення ліворуч не містить ніяких схованих повідомлень. Усередині правого зображення , схований вірш "If ", Редьярда Кіплінга, англійською мовою, розміром 1,5 Kb.

Вивчимо розходження між зображеннями до й після приховання інформації. У таблиці показані відмінності у квантованих коефіцієнтах DCT першого лівого блоку зображення JPEG розміром 8x8 пікселів (64 байта). У лівому стовпці таблиці показаний уміст блоку до приховання тексту, у середньому показаний блок після приховання тексту, у правому стовпці показані молодші біти зміненого блоку. У таблиці виділені жирним змінені програмою JSteg байти. Зміна здійснена за допомогою алгоритму LSB.

Кольором позначені області формату даних JSteg, описані вище: червоний колір для поля "A" – першого 5 бітного блоку, що містить розмір

наступного поля розміру; зелений для блоку "B", що містить розмір даних, і синій для поля "З", у якому перебувають сховані дані. Важливо пам'ятати, що якщо байт дорівнює 0 або 1, він не приймаються в увагу.

Вихідне зображення 16778 байт Зображення зі схованим текстом 17344 байт

Рисунок 2.1 – Зображенні обкладинки "Планета історій"

Далі декодируем наведений у таблиці перший блок розміром 8x8 вручну, щоб зрозуміти, як працює алгоритм приховання даних. У результаті:

Поле A: завжди містить 5 біт для декодування. З таблиці видно що це число рівне 00 01 00 01 01 або 01011 у двійковій системі числення або 11 у десятковій системі числення. Це розмір наступного поля, що містить розмір схованих даних.

Поле B: 11 біт для декодування: 01 01 00 00 00 00 01 00 00 00 01 або 11000010001 у двійковій або 1553 у десятковій системі. Це точний розмір приховуваного вірша.

Поле C: 1553 x 8 біт для декодування, починаючи з: 00 01 00 00 01 00 00 01 або 01001001 у двійковому або 73 у десятковій системі, що відповідно до таблиці ASCII є буквою «I» у верхньому регістрі, і відповідає першій букві схованого вірша.

Таблиця 2.1 – Відмінності у квантованих коефіцієнтах DCT першого лівого блоку зображення JPEG

Вихідне зображення	Зображення зі схованим текстом	
Квантовані коефіцієнти DCT	Квантовані коефіцієнти DCT	Молодші біти квантованих коефіцієнтів DCT
D6 69 13 05 03 15 F2 EB	D6 69 12 05 03 15 F3 EA	00 01 00 01 01 01 01 00
FF 04 01 00 FA FB F9 FF	FE 04 01 00 FA FB F8 FE	00 00 01 00 00 01 00 00
06 02 FE FF 00 00 00 FF	06 03 FE FF 00 00 00 FE	00 01 01 00 00 00 00 00
01 03 02 01 01 FF 00 00	01 02 03 01 01 FE 00 00	01 00 01 01 01 00 00 01
01 00 00 00 00 00 00 00	01 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00 FF FF 00 00 00 00 00	00 FE FF 00 00 00 00 00	00 00 01 00 00 00 00 00
00 00 00 00 00 01 00 00	00 00 00 00 00 01 00 00	00 00 00 00 00 01 00 00
00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python https://www.python.org/. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних,

для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітних проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.

2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.

3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.

4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.

5. World of Tanks.

6. Вільна енциклопедія Вікіпедія.

7. Пошукова система Google.

8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.

9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після

чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл fetch – decode – execute (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом.

З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди.

Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.

2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою https://www.python.org/downloads.

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

– IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.

– Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.

– Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.

– PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.

– Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.

– Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.

– PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.

– Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

– Mu – редактор коду Python для програмістів-початківців. Для користувачів Linux, Windows і macOS.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи забезпечення авторських прав за допомогою цифрових водяних знаків.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і

негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Технічні й програмні методи захисту авторських прав можна розділити на три групи:

– методи, що передбачають наявність явного підпису, що вказує на автора або власника прав на матеріал;

– методи, що передбачають наявність схованої авторської «підпису», впроваджуваної в цифровий файл, як правило, з використанням стеганографічних технологій;

– методи захисту від несанкціонованого копіювання, поширення й використання авторського матеріалу, наприклад, шляхом захисту від копіювання CD або DVD, що містять цифрові файли, які захищаються.

Розглянемо першу групу методів – методів «підписування» авторських матеріалів. Явна наявність підпису під текстом, фотографією, рисунком або іншим зображенням, а також вказівка авторства в титрах відеозапису жодним чином не захищає матеріал від незаконного використання. Деяким виключенням можна вважати ситуацію, коли прямо по фотографії наноситься напис, що вказує на автора роботи, причому такий напис, що її непомітне видалення практично неможливо. Очевидно, що в цьому випадку ми маємо справу із прийомом, що псує фотографію настільки, що її піратське відтворення стає безглуздим. Такий метод часто застосовується при розміщенні фотозображень в Інтернеті, поряд з іншим методом, що припускає розміщення зображення настільки малого розміру й поганої якості, що його використання інакше, крім як у якості «прев'юшки» для переднього перегляду неможливо. Зрозуміло, що методи «підписування» авторських матеріалів не здатні забезпечити захист прав авторів.

Друга група методів передбачає потайливе впровадження в графічний аудіо- або відеофайл спеціальної інформації, включаючи інформацію про правовласника, його поштовій і електронній адресі, телефоні й інше даних (потайливе впровадження даних у текстові файли також можливо, але з метою захисту авторських прав не використовується, тому що присутність «явної» підпису не псує текст). Наявність інформації про автора матеріалу дає можливість особам, зацікавленим у його використанні зв'язатися з автором з метою висновку з ним угоди про законне використання його роботи. З іншого боку, наявність стеганографічних впроваджених даних, у принципі, дозволяє «відстежити» розміщення матеріалу в Інтернеті, а виходить, виявити факти можливого його несанкціонованого використання. Розглянемо цю групу методів більш докладно.

Стеганографія (у перекладі із грецького – тайнопис) – техніка схованої передачі або схованого зберігання інформації. Основний принцип комп'ютерної стеганографії припускає використання двох типів файлів – файл-повідомлення, що повинне бути приховано, і файл-контейнер, у якому повинне бути сховане повідомлення. Контейнер – будь-який файл або потік даних, у який може бути приховано убудована інформація. Якщо контейнер не містить убудоване повідомлення, то він називається порожнім. Контейнер, що містить убудовану інформацію, називається заповненим або стегоконтейнером. У нашім випадку в ролі файлу-контейнера виступає цифровий файл, у який необхідно впровадити інформацію про власника авторських прав. Найважливішою вимогою до стегоконтейнеру є зовнішня нерозрізненість порожнього й заповненого контейнерів, щоб наявність схованої інформації для стороннього спостерігача ніяк зовні не проявлялося.

Одним з основних і найбільш перспективних напрямків використання стеганографії є вбудовування цифрових водяних знаків (ЦВЗ) (watermarking), що сприяють захисту авторських прав на графічні й аудіо- або відеофайли від піратства й забезпечують можливість контролю за поширенням захищеної

інформації. Цифровий водяний знак (ЦВЗ) – спеціальна мітка, непомітно впроваджувана в графічний, аудіо-, відео- або інший файл із метою контролю його використання. Для впровадження й розпізнавання цифрових водяних знаків розроблене спеціальне програмне забезпечення. Програми розпізнавання ЦВЗ дозволяють витягати інформацію про власника авторських прав і про те, як вступити з ним у контакт. Програми виявлення ЦВЗ дозволяють контролювати поширення захищеної інформації. Таким чином, основною областю застосування ЦВЗ є захист інтелектуальної власності від копіювання й несанкціонованого використання.

Основним недоліком методів, що передбачають впровадження в цифрових водяних знаків, є їхня нездатність запобігти незаконному використанню авторських матеріалів.

Третя група методів захисту авторських прав спрямована на блокування несанкціонованого копіювання й використання авторського матеріалу. Слід зазначити, що таке завдання є типовим завданням захисту інформації, рішення якого часто є обов'язковою для забезпечення інформаційної безпеки. Очевидно, що якщо тим або іншому методу забезпечити захист від несанкціонованого копіювання графічних, аудіо- або відеофайлів, те їхнє піратське використання стає неможливим і завдання захисту авторських прав вирішується найбільше успішно. Таким чином, можна вважати, що найкращим способом захисту прав авторів на матеріали, представлені в цифровому виді, є забезпечення інформаційної безпеки носіїв цих даних або захист відповідних файлів від копіювання.

На сьогоднішній день найпоширенішими є методи, що забезпечують захист від копіювання інформації, записаної на CD або DVD. Незважаючи на те, що такі методи широко використовуються, майже всі вони мають істотний недолік – неприступністю для «простого» користувача, що не є професіоналом-комп’ютерщиком і не володіє певними апаратними засобами. Крім того, не всі існуючі методи досить надійні й найчастіше легко обходяться за допомогою

спеціальних «хакерських» програм. У роботі [1; 2] запропонована вільна від зазначених недоліків система захисту інформації, записаної на компакт-диск, від несанкціонованого копіювання. Алгоритм і його програма, що реалізує, передбачають кілька ступенів захисту даних з використанням криптографії, фізичних характеристик диска й контролю реєстру. Програма складається із двох підпрограм, перша з яких зберігається в комп'ютері, на якому здійснюється запис інформації на компакт-диск, і забезпечує постановку захисту на диск, а друга підпрограма записується на компакт-диск разом із інформацією, яка захищається, і призначена для перевірки диска на оригінальність. При успішному завершенні перевірки диска на оригінальність друга підпрограма забезпечує роботу із захищеним диском. Запропонований алгоритм захисту від копіювання даних, записаних на CD, включає кілька рівнів захисту.

Перший рівень захисту припускає шифрування даних. Для шифрування використовується швидко працюючий криптографічний алгоритм симетричного шифрування. Ключ шифрування складається шляхом множення фіксованого коефіцієнта, запрограмованого в програмі захисту диска, і випадкового числа, автоматично генеруємого й записуваного в спеціальний файл, записуваний на жорсткий диск у папку із проектом запису. Використання як ключ тільки випадкового числа не забезпечує необхідного захисту, тому що файл, у якому це число зберігається, може бути легко зламаний, а визначення додаткового множника, що зберігається в програмі захисту, ускладнює в цьому випадку завдання визначення ключа тільки по випадковому числу.

Другий рівень захисту – це «плаваючий» серійний номер використовуваної для запису «болванки» компакт-диску. «Плаваючої» він є тому, що на чистих компакт-дисках серійний номер не є постійним і генерується програмами запису при кожному записі або дозапису диска, тобто тільки при закінченні запису диска. Використання цього номера програмою захисту здійснюється в такий спосіб. Після запису проекту на компакт-диск програма захисту зчитує з компакт-диску згенерирований «плаваючий» номер і кількість

зайнятого проектом на диску місця в байтах. За допомогою цих даних, що є унікальними для кожного диска, генерується спеціальний серійний номер захищеного проекту. Цей номер повідомляється користувачеві по закінченні запису компакт-диску, для чого на екран монітора виводиться вікно із пропозицією записати номер захищеного проекту на поверхню компакт-диска або його впакування. Необхідність запису номера проекту пов'язана з тим, що надалі цей номер буде використовуватися для перевірки диска на оригінальність, тобто, для перевірки чи є диск копією або оригіналом.

Третій рівень захисту – запис до реєстру в спеціально створювану галузь HKEY_LOCAL_MACHINE\SOFTWARE\SYSDBA набору даних, використовуваних надалі для перевірки диска на оригінальність. Цими даними є буква, що позначає диск, що відповідає приводу CD-ROM, у який вставлений захищений диск, і номер захищеного проекту. Запис цієї інформації до реєстру відбувається при першому запуску захищеного диска, а при послу- запусках, що дмуть, автоматично здійснюється перевірка захищеного диска на оригінальність. Такий запис забезпечує захист від емуляторів диска, тому що емулятору відповідає інша буква диска, а запис до реєстру номера захищеного проекту при першому запуску CD уможливлює не вводити цей номер при наступних запусках. Це досягається завдяки тому, що записана на захищений компакт-диск підпрограма перевіряє записи в реєстрі й при виявленні зазначених вище даних звіряє номер, записаний у реєстрі захищеного проекту, з аналогічним номером, записаним на CD, також звіряється буква привода CD-ROM, розпізнана програмою, з буквою, записаної в реєстрі. Більш докладно порядок роботи із програмою описаний в [1; 2].

Методи протидії несанкціонованому копіюванню компакт-дисків дозволяють захистити від піратського використання будь-які цифрові файли: графічні, відео-, аудіо- і ін. Але якщо авторські матеріали представлені на іншому носії, наприклад, USB-флеш накопичувачі або флеш-карті (USB Flash Drive, Flash-Card), те захист даних повинна організовуватися інакше. Для того, щоб не

було залежності від того, на якому носії записані авторські матеріали, необхідна розробка спеціальних методів. Більш прості й надійними такі методи захисту виявляються тоді, коли розробляються під конкретний вид авторського матеріалу.

Розглянемо, як може бути вирішена проблема захисту авторських прав на представлені в цифровому виді фотографії й інші зображення. З погляду інформаційної безпеки таке завдання може трактуватися як завдання захисту файлів графічних форматів від несанкціонованого копіювання.

Як вказувалося вище, найпростіший шлях запобігання незаконного використання графічних матеріалів полягає в тому, що зображення представляються в сильно зменшеному розмірі й/або навмисно погіршеній якості. Такий підхід далеко не завжди допустимо. Так, у сфері реклами досить розповсюдженої є ситуація, коли замовник просить фотографа або дизайнера надати добірку фотографій або графічних робіт для можливого відбору з метою використання в рекламних виданнях або зовнішній рекламі. Найважливішою умовою, пропонованою до зображень, що відбираються, є їх високий технічний і художній рівень. Зрозуміло, ніякі прийоми захисту авторських прав, засновані на погіршенні якості зображення в такому випадку незастосовні. У той же час, пред'явивши потенційному замовникові крупноформатні високоякісні зображення, автор ризикує, що його робота може бути піратськи використана, тобто його права будуть порушені. Вихід із цієї ситуації можливий, якщо зробити високоякісні цифрові зображення захищеними від копіювання. У роботах [3; 4; 5] запропонований новий підхід до рішення такого завдання. Складність протидії піратському використанню зображень полягає в тому, що необхідно не тільки забезпечити захист від несанкціонованого копіювання файлу, але й захист від збереження зображення, виведеного на екран, для чого звичайно використовується функція Print Screen.

На перший погляд, розмір зображення на екрані в пікселях, а саме таке зображення може бути скопійоване за допомогою Print Screen, не настільки

великий, щоб авторові випливало турбуватися із приводу його можливого, наприклад, поліграфічного відтворення. Насправді це не так. Наприклад, при розмірі цифрового файлу, рівного розміру екрана 2560 х 1600 пікселів (такий дозвіл мають багато хто монітори, наприклад, Dell 3007 WFP-HC, Apple Cinema Display 30 (Alu) і ін.) можлива якісна типографська печатка зображення формату 21,7 х 13,5 см, що близько до формату А5 (розрахунки проведені для друку з линиатурой типографського растра 150 lpi і коефіцієнтом якості 2, чому відповідає дозвіл цифрового файлу 300 dpi). Якщо прийняти коефіцієнт якості рівним 1,5 (дозвіл цифрового файлу 225 dpi), то розмір друку складе 28,9 х 18 см. Зрозуміло, що зображення такого розміру може використовуватися не тільки для друку листівок, але й календарів, буклетів і іншої рекламної продукції, а також у зовнішній рекламі. У тому випадку, коли розмір цифрового файлу становить 3840 х 2400 пікселів (такий дозвіл – більше 9 мегапікселів – має, наприклад, монітор IBM T221 з розміром діагоналі 22,2 дюйми) розмір якісного поліграфічного відбитка може досягати 43,3 х 27,1 см, що можна віднести до крупноформатного друку. Таким чином, як показують наведені розрахунки, вимога подання розміру графічного файлу всього лише рівного дозволу гарного монітора, створює передумови для найрізноманітнішого піратського використання зображення, включаючи не тільки розміщення в Інтернет-галереях, але і якісне поліграфічне відтворення. Програмне забезпечення, призначене для захисту авторських прав на зображення шляхом блокування їхнього несанкціонованого копіювання, у тому числі на стадії перегляду, повинне вирішувати наступні завдання.

По-перше, необхідно здійснити протидію копіюванню зображення з використанням функції Print Screen, що представляє найпростіший спосіб копіювання інформації, виведеної на екран. Для захисту від такого способу копіювання необхідно забезпечити перехоплення натискань клавіші <Print Screen> і після цього підмінити зображення, розміщене в буфері обміну на деяке зображення, підготовлене заздалегідь.

По-друге, необхідно вирішити завдання захисту цифрового зображення від копіювання, здійснюваного за допомогою сторонніх додатків, що роблять скриншоти в автоматичному режимі. Найбільш ефективним захистом від програм, що роблять скриншоти за допомогою API-функцій, є контроль викликів WinApi функцій, тобто при виклику функцій, що копіюють зображення, забороняти ця дія системними засобами. Для рішення такого завдання необхідно забезпечити одержання всього списку запущених у системі процесів або потоків з метою наступного контролю викликів зі сторони цих процесів WinApi функцій, що копіюють зображення, виведене на екран.

У запропонованому в роботах [1; 2; 3] методі рішення завдання захисту цифрових зображень від копіювання при запуску спеціально розробленої програми формується список всіх процесів і до кожного з них прикріплюється dll бібліотека, що заміняє що перехоплюється WinApi функцію в таблиці імпорту на якусь заздалегідь написану функцію, код якої реалізований у даній бібліотеці. Далі, при виявленні нового процесу до нього також прикріплюється dll бібліотека, що забезпечує контроль над всіма потоками, що виконуються в системі. Основною складністю при цьому є проблема обміну даними з dll бібліотекою, тому що наявність всіх вхідних даних для одержання контролю над викликуваними процесом WinApi-функціями потрібно на момент запуску бібліотеки, що обумовлює необхідність одержання даних в dll Entry Point. Для цього викорикористується технологія File Mapping.

Описані принципи захисту цифрових зображень від несанкціонованого копіювання були реалізовані в спеціальному програмному забезпеченні, тестування якого показало його коректну й стабільну роботу. Зокрема, як це й мабуть, виявилося заблоковане копіювання з використанням функції Print Screen, а також копіювання за допомогою стороннього програмного забезпечення.

Таким чином, найбільш діючими методами захисту авторських прав на матеріали, представлені в цифровому виді, у тому числі матеріали, призначені для розміщення в засобах масової комунікації, варто визнати програмно-апаратні методи, що базуються на принципах захисту інформації від несанкціонованого

копіювання й використання. У цьому змісті захист прав автора може розглядатися як спеціальний випадок забезпечення інформаційної безпеки його робіт.

3.2 Розробка структурної схеми

У роботі дамо визначення цифрових відбитків пальців (ЦВП), що занурюються в цифрові зображення. Зловмисники прагнуть одержати нелегальну копію, що не містить вкладення, що вони потім можуть продати, як свій власний продукт. Група таких нечесних користувачів називається коаліцією, а спроба видалити вкладення класифікується як коаліційна атака.

Розглянута в роботі основна атака усередненням виробляється відповідно до вираження:

$$C'_w(n) = \frac{1}{K}\sum_{j \in S_c} C_{wj}(n) + \varepsilon(n), n = 1,2...N, \qquad (3.1)$$

де:

– S_c – безліч користувачів, що об'єдналися в коаліцію;

– K – розмір коаліції «піратів», $\varepsilon(n)$ – додатковий адитивний шум;

– $w_j(n) \in \{-1,1\}$ – унікальний ідентифікатор $j^{го}$ користувача (ЦВЗ), псевдовипадкова послідовність, що приблизно можна вважати рівноймовірностною і взаємонезалежної (3.визначається стегоключом);

– $C'_w(n)$ – копія, що буде далі нелегально поширюватися від імені всіх учасників коаліції, N – кількість елементів ЦВЗ;

– $C_{wj}(n)$ – сигнал із вкладеним ЦВЗ для $j^{го}$ користувача.

У бакалаврській роботі розглянуті особливості детектування ЦВЗ в умовах атак змовою, наведені критерії ефективності коаліційних атак. Для визначення ступеня успішності коаліційних атак по видаленню ЦВП використовуються наступні параметри:

– P_e – імовірність помилкового виявлення учасників змови;

– P_{fai} – імовірність помилкового спрацьовування детектора й занесення безневинного $i^{го}$ користувача до складу піратів;

– P_{mi} – імовірність пропуску $i^{го}$ учасника змови;

– $S`_c$ – состав коаліції, виявлений детектором, що може відрізнятися від щирого списку зловмисників S_c, додатковим параметром ефективності може також бути візуальна оцінка нелегальної копії.

Далі зробимо оцінку ефективності методів захисту від коаліційних атак при використанні широкополосних сигналів. У цьому випадку, як і у всій роботі, мається на увазі, що інформація, вкладена в цифрові зображення, містить ідентифікаційні дані продавця й/або легального покупця, що дозволяє виявити згодом порушників, які копіюють і нелегально поширюють куплені продукти.

Розглядається коаліційна атака на систему цифрових водяних знаків 1-го типу, при використанні того самого ЦВЗ для різних ПС. Вкладення описується в такий спосіб:

$$C_{wi}(n) = C_i(n) + \alpha w(n), \tag{3.2}$$

де $i=1,2...L, n=1,2...N, \alpha \geq 1$, $w(n) \in \{-1,1\}$ –псевдовипадкова послідовність, що приблизно можна вважати рівноймовірностною і взаємонезалежною.

Коаліційна атака по видаленню ЦВЗ 1-го типу представлена в такий спосіб:

$$C'_{wi}(n) = C_{wi}(n) - \tilde{w}(n) + \varepsilon(n), \tag{3.3}$$

де:

$$\tilde{w}(n) = \frac{1}{K}\sum_{j \in S_c} C_{wj}(n) = \frac{1}{K}\sum_{j \in S_c} C_j(n) + w(n),$$

– S_c – безліч порядкових номерів користувачів, що входять у коаліцію розміром K;

– $\varepsilon(n)$ – аддітивний шум.

Далі відзначено, що звичайно використовується кореляційний інформований декодер виявлення ЦВЗ:

$$\Lambda_i = \sum_n \left(C'_{wi}(n) - C_i(n) \right) w(n) = \sum_n \left(C_{wi}(n) - \tilde{w}(n) - C_i(n) \right) w(n) =$$
$$= \sum_n \left(\frac{1}{K} \sum_{j \in S_c} C_j(n) \right) w(n). \qquad (3.4)$$

У бакалаврській роботі позначені мети атаки, які переслідують зловмисники:

1. Унеможливити виділення ЦВЗ легальним користувачем.

2. Не дозволити виявити учасників коаліції.

Оскільки $\frac{1}{K} \sum_{j \in S_c} C_j(n)$, *n=1,2,...N*не пов'язане з *w(n)*, те виявлення ЦВЗ виявляється неможливим і першою метою атаки тривіально досягається.

Якість зображення критично погіршується (що видно на прикладах, наведених у бакалаврській роботі).

Доводиться що відношення сигнал/шум *(S/N=η_a)* після такої атаки:

$$\eta_a = \frac{\sigma_c^2}{Var\{\frac{1}{K} \sum_{j \in S_c} C_j(n)\}} = K, \qquad (3.5)$$

Де $\sigma_c^2 = Var\{C_i(n)\}$ – середня дисперсія оригіналу.

Далі оцінюється ефективність досягнення 2$^{\text{й}}$ мети. Для цього перебуває кореляція між оригіналами користувачів без вкладення й усередненою копією. На основі отриманих результатів робиться вивід про те, що виявлення $i^{го}$ користувача в складі коаліції можливо, але власникові ПС юридично неможливо довести, що в $C_i(n)$ вироблялося вкладення його ЦВЗ.

Через сильне погіршення якості нелегальної копії детальне вивчення системи ЦВЗ 1го типу не є основним завданням бакалаврській роботі.

Опишемо коаліційну атаку усередненням на системи ЦВП 2-го типи при використанні псевдовипадкових послідовностей. У системі другого типу крім інформації про продавця в копію можуть вкладатися деякі дані про покупців, наприклад, порядковий номер по кількості проданих екземплярів. Наступні

формули описують додавання в те саме зображення унікальних ЦВП, що відповідають різним користувачам:

$$C_{wi}(n) = C(n) + w_i(n), \tag{3.6}$$

де:

– $C(n)$ – нерухливе зображення (ПС), передане одночасно L легальним покупцям;

– $w_i(n)$ – псевдовипадкова послідовність, що виробляється по секретному *стегоключу* індивідуально для кожного $i^{го}$ користувача (Покупця), $w_i(n) \in \{-\alpha, \alpha\}$,

– $C_{wi}(n)$ – сигнал із вкладеним ЦВЗ для і-г*о* користувача;

– α – амплітудний коефіцієнт, що визначає «глибину» вкладення ЦВЗ і впливає на якість ПС після вкладення ЦВЗ,

– N – кількість елементів ЦВП, $n=1,2...N$;

– L – загальна кількість легальних покупців зображення $C(n)$;

– K – кількість учасників коаліції.

Атака виробляється відповідно до вираження (3.1). Інформований декодер для визначення учасників коаліції працює по формулі:

$$\begin{aligned} &\Lambda_i = \sum_{n=1}^{N} (C'_w(n) - C(n)) w_i(n), \\ &i = 1, 2...L, \end{aligned} \tag{3.7}$$

де входження в коаліцію визначається умовами $\Lambda_i > \lambda -> i \in S_c$, $\Lambda_i \le \lambda -> i \notin S_c$.

У роботі виведені формули для розрахунку ймовірності пропуску P_{mi} і фіктивної тривоги P_{fiai} для учасників коаліції при досить більших довжинах послідовностей ЦВЗ N, що використовують широкополосні сигнали:

$$P_{fai} = 1 - Q\left(\frac{-\lambda}{\sqrt{N(\sigma_\varepsilon^2 \alpha^2 + \frac{\alpha^4}{K})}} \right), \tag{3.8}$$

$$P_{mi} = Q\left(\frac{\frac{\alpha^2 N}{K} - \lambda}{\sqrt{N(\sigma_\varepsilon^2 \alpha^2 + \frac{\alpha^4 (K-1)}{K^2})}}\right). \quad (3.9)$$

Оцінюється якість ПС відразу після занурення ЦВЗ і після коаліційної атаки, у термінах *відносини сигнал/шум* (η_w і η_a, відповідно):

$$\eta_w = \frac{\sigma_c^2}{\alpha^2}, \quad (3.10)$$

$$\eta_a = \frac{\sigma_c^2}{Var\{C(n) - C'_w(n)\}} = \frac{\sigma_c^2}{\sigma_\varepsilon^2 + \frac{\alpha^2}{K}}. \quad (3.11)$$

При виборі оптимального порога, що забезпечує виконання умов $P_e = P_{mi} = P_{fai}$, при $K>>1$, отримане вираження для P_e:

$$P_e = Q\left(\frac{1}{2}\sqrt{\frac{N}{K^2 \eta}}\right), \quad (3.12)$$

де $\eta = \frac{\eta_w}{\eta_a}$.

З (3.12) робиться важливий вивід**:** компенсація коаліційної атаки виконуваною групою з K «піратів» (3.тобто збереження такої ж імовірності помилки P_e, як і при відсутності коаліційної атаки) вимагає збільшення N (А, отже, і зниження обсягу вкладення) в K^2 разів. Ця властивість названа квадратичною компенсацією коаліційної атаки.

Як видно із прикладів, представлених у бакалаврській роботі, після коаліційної атаки якість нелегальної копії поліпшується.

При більших величинах K, перехід до ортогональних ЦВЗ не дає істотного захисту від коаліційної атаки.

Для того, щоб забезпечити ЦВЗ великій кількості користувачів і зменшити довжину послідовності N, перспективної є ідея (відома раніше з літератури)

штучно ввести кореляцію в послідовності ЦВЗ. Такі послідовності називаються *коаліційними кодами.*

Дамо оцінку ефективності використання антикоаліційних кодів, побудованих на основі неповних збалансованих блок-схем (НЗБС), для захисту від коаліційних атак. Дається загальне визначення: НЗБС с параметрами *(ν,k,λ)* називається безліч *A* послідовностей (або блоків) довжини k, кожна позиція яких належить безлічі *X* чисел (1,2 …ν) за умови, що пари чисел з *X* зустрічається точно в λ блоках.

На основі безлічі *A* формується матриця инциденцій **C**, з кожного стовпця якої виходить послідовність ЦВЗ. Кількість стовпців дорівнює числу користувачів системи. Перетворивши кожний стовпець c_j матриці **C** у сигнал ЦВЗ w_j, одержуємо систему ЦВЗ.

Описується алгоритм виявлення учасників коаліції: коли сигнали ЦВЗ із (ν,k, λ)-НЗБС усередняються, позиції, у яких вони збігаються й мають значення "1", визначають учасників коаліції.

Приведемо методи вкладення АКК у зображення в умовах коаліційних атак:

– вкладення в пікселі, обумовлені по стегоключу;

– вкладення в послідовні блоки l_n по M пікселів;

– вкладення в послідовні блоки l_n по M пікселів за допомогою псевдовипадкової послідовності (ПСП).

Однак, кількість комбінацій, отриманих на основі НЗБС, різко обмежено. Крім того, ЦВЗ (ЦВП) повинні бути секретними, щоб уникнути тривіальної атаки за допомогою їхнього вирахування. У той же час відомо, що коди типу НЗБС (і інші з малою взаємною кореляцією) обмежені по обсязі при використанні їх як ключі. Тому в наступній главі пропонується використання принципове нових кодів для визначення состава учасників коаліції.

Розглянемо застосування диз'юнктних кодів в евклідовому просторі (ДКЕ) для захисту від коаліційних атак. Відзначається, що існують такі послідовності,

які мають малу взаємну кореляцію, але охоплюють велика кількість користувачів. Набір кодів буде великий, якщо задовольняються умови $\mathbf{S}^{\mathrm{T}}\mathbf{S} = \mathbf{I}_{\mathbf{M}}$, якщо $M \leq N$, і $\mathbf{S}\mathbf{S}^{\mathrm{T}} = (M / N)\mathbf{I}_N$, коли $M > N$, де $\mathbf{S}$ – матриця елементів кодів ЦВЗ для всіх користувачів стегосистеми, M – кількість користувачів, а N – довжина послідовності.

Такі послідовності, називаються кодами, що задовольняють граничним умовам Уелча. Однак для них існують лише оцінки середнього квадрата кореляції, але не точні границі мінімального евклідова відстані, які потрібні для оцінки ефективності ЦВП. Крім того, кількість таких послідовностей недостатньо для виключення атаки за допомогою вирахування ЦВЗ. Тому були обрані коди (Запропоновані раніше для рішення зовсім інших завдань), визначення яких приводиться нижче.

Визначення: Нехай **C** буде повним набором ортонормованих векторів у просторі R^N. Для будь-яких піднаборів A існує число елементів $|A|$ і $f(3.A)$ визначає суму векторів **x** в A $f(A) = \sum_{x \in A} x$.

Також для $m=0,1,...T$ існує:

$$\mathcal{A}(m) = \{A \subseteq C : |A| \leq m\},$$
$$\varphi^{(m)} = \{f(A) : A \in \mathcal{A}(m)\},$$
$$d_E(\varphi^{(m)}) = \min\|f(A) - f(B)\|;$$
$$A \neq B, A, B \in \mathcal{A}(m).$$

Норма $||x||$ є звичайна евклідова норма $\left\|x^2\right\| = \sum_{i=1}^{N} x^2$.

Набір **З** буде називатися ДКЕ з параметрами (n, *m, T, d*), якщо $|C|=T$ і $d(\varphi^{(3.m)}) \geq d$.

Якщо АКК обрані випадковим образом, то мінімальне евклідова відстань $d(\varphi^{(3.m)})$ позначене як $\tilde{d}$,(3.як показано в роботі T.Ericson, L.Gyorfi), буде пов'язане з кількістю користувачів T і розміром коаліції L наступним вираженням:

$$T = e^{NE(L,\tilde{d})}, \qquad (3.14)$$

де

$$E(L,\tilde{d}) \geq A(\tilde{d}) \min_{1 \leq l \leq L} \frac{log\sqrt{\pi l}}{2l}, \text{ де } A(\tilde{d}) = \max_{0 \leq x \leq 1} \frac{1-x}{1+x} x^{\tilde{d}/4}.$$

Співвідношення між $\tilde{d} = d_E(\phi^{(L)})$ для послідовностей $\left\{\pm\frac{1}{\sqrt{N}}\right\}$ і d для АКК (значення $\{\pm\alpha\}$), буде мати вигляд:

$$d = \frac{\sqrt{N}}{L}\tilde{d}\alpha\,. \tag{3.15}$$

Відзначається, що фактично ДКЕ це випадково обрані ШПС коди, але, як відзначено раніше, для них можна розрахувати мінімальна евклідова відстань між результатами усереднення коаліції з «m» піратів, а завдяки існуванню АСД, вони мають ефективний алгоритм декодування.

Таким чином, у справжній роботі був здійснений перехід від регулярних структур до випадкових структур кодів, щоб одержати гарантоване найбільше мінімальне евклідова відстань між різними коаліціями. Випадкові коди давно відомі в теорії кодування, але там перешкодою був алгоритм декодування й неперебірний обсяг кодів. У нашім випадку кількість кодових слів дорівнює кількості користувачів системи ЦВЗ, а для визначення состава учасників коаліції використовується декодування по мінімальній евклідовій відстані за допомогою так званого алгоритму сферичного декодування. Відомо, що декодер за критерієм відносини правдоподібності буде декодером мінімального евклідова відстані в просторі R^N:

$$\tilde{S}_c = \arg\min_{S_c} \left\| C'_w - C - \frac{1}{L}\sum_{i \in S_c} w_i \right\|, \tag{3.16}$$

де ||…||евклідова норма в R^N, L – кількість учасників коаліції.

Виведено формулу для ймовірності помилки при визначенні помилкової коаліції $S`_c$ замість щирої коаліції S_c:

$$\Pr\{S'_c / S_c\} \approx Q\left(\frac{1}{2}\sqrt{\frac{d(S_c,S'_c)}{\sigma_\varepsilon^2}}\right) = Q\left(\frac{\sqrt{d(S_c,S'_c)}}{2\sigma_\varepsilon}\right), \tag{3.17}$$

де

$$d(S_c, S_c') = \left\| \frac{1}{L} \sum_{i \in S_c} w_i - \frac{1}{L} \sum_{i \in S_c'} w_i \right\|,$$

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{t^2}{2}} dt .$$

Нехай мінімальна евклідова відстань d у наборі слів ЦВЗ буде:

$$d = \min_{S_c \neq S'} d(S_c, S_c') . \tag{3.18}$$

Тоді ймовірність помилки буде мати верхню границю:

$$Pe = \Pr\{S_c' / S_c\} \leq Q\left(\frac{d}{2\sigma_\varepsilon}\right) = Q\left(\frac{\sqrt{N}\tilde{d}}{2L}\right). \tag{3.19}$$

У бакалаврській роботі наведені розрахунки ймовірності помилки по (3.19), а також при вкладенні в блоки пікселів довжини M, які показали високу ефективність ДКЕ при декодуванні за правилом (3.16).

Однак, оскільки рішення рівняння (3.16) має неполіноміальну складністю обчислення щодо розмірів коаліції, необхідно було перейти до алгоритму сферичного декодування.

Завдання (3.16) відоме як завдання цілочисельного квадратичного програмування (ЦКП). Її можна компактно записати як:

$$\tilde{S} = \arg\min_{S \in Z^m} \|\mathbf{x} - \mathbf{Hs}\|, \tag{3.20}$$

де:

$$\mathbf{x} = C_w' = (C_w'(1), ...C_w'(N)). \tag{3.21}$$

Рішення (3.20) далі заміняється знаходженням всіх можливих векторів коаліцій S, що задовольняють умові:

$$\|x' - \mathbf{R}S\|^2 \leq r^2, \tag{3.22}$$

де r – деяка задана величина. Рішення нерівності (3.22), як показано у відомих роботах володіє (За певних умов) поліноміальною складністю.

При виборі порога r варто керуватися тим, що якщо вибрати r занадто більшим, то складність рішення (3.22) може виявитися занадто великою. Якщо

визначити *r* дуже малим, то може виявитися, що жоден варіант **S** не задовольняє (3.22). Якщо ж у сферу потрапив хоча б один варіант, то рішення завжди буде відповідати оптимальному.

Відзначається, що при відомому розмірі коаліції, буде справедливо нерівність:

$$\left\| \overline{S_C} + \overline{\varepsilon} - \overline{S_C'} \right\| \le \left\| \overline{S_C} - \overline{S_C'} \right\| + \left\| \overline{\varepsilon} \right\|,$$

де $\overline{S_C'}$ – щира коаліція атакуючих, а $\overline{S_C'}$ – передбачувана коаліція атакуючих. Тоді для правильного рішення $\overline{S_C} = \overline{S_C'}$:

$$r \le \left\| \overline{\varepsilon} \right\|. \qquad (3.23)$$

Оскільки дисперсія гауссовського шуму σ_ε^2 передбачається відомої, то радіус гіперсфери можна знайти як $r = \sqrt{N\sigma_\varepsilon^2}$.

Розраховані теоретично й знайдені моделюванням величини мінімальної евклідової відстані близькі між собою. Були також знайдені значення кількості користувачів у системі ЦВЗ для заданого *d*, еквівалентні кількості кодових слів *T*.

Як показано в бакалаврській роботі для випадку використання ДКЕ, імовірність правильного виявлення коаліції піратів виявляється вище, ніж при використанні послідовностей на основі НЗБС.

Випадкові ДКЕ коди дозволяють також постачити ЦВП більша кількість користувачів. З іншого боку, запропонований код забезпечує ефективність (P_d) навіть кращу, чим у звичайних ЦВП, і одночасно стійкість до атаки по видаленню ЦВП їхнім простим вирахуванням. Такий гарний результат досягається за рахунок застосування АСД, що має поліноміальну складність.

Структурна схема системи у вигляді сукупності схеми пристроїв стеганографічного приховування та вилучення даних в просторовій області зображень із використанням прямого розширення спектру, які побудовані за пропонованим способом зображено на рис. 3.1.

Пристрій стеганографічного приховування даних (див. рис. 3.1) працює наступним чином. Джерело інформаційних повідомлень формує послідовність інформаційних даних, які подаються пристрій шифрування, ініційований таємним ключем K_3, що формується джерелом ключів 3. Зашифровані інформаційні повідомлення подаються на пристрій перешкодостійкого кодування, в якому виконується внесення спеціально формованої надмірності для підвищення достовірності інформаційних зашифрованих даних. Джерело ключів 2 формує таємний ключ K_2, який ініціює генератор псевдовипадкових послідовностей. Результатом роботи генератору псевдовипадкових послідовностей є дискретні сигнали, тобто дискретні послідовності, елементи яких сформовано псевдовипадковим чином. Сформовані псевдовипадкові послідовності Φ_j поступають на додатково (в порівнянні із відомим способом) введений пристрій відбору послідовностей за правилом (3.14), в якому для всіх $i = 0,..,N-1$ розраховується значення коефіцієнту кореляції

$$\rho\left(C_i,\Phi_j\right)=\frac{1}{n}\sum_{z=0}^{n-1}C_{i_z}\varphi_{j_z}$$

та порівнюється із наперед визначеним значенням $\rho_{\max}$.

У випадку, коли хоча б для одного $i\in\{0,..,N-1\}$ розраховане значення $\rho\left(C_i,\Phi_j\right)$ перевищить значення порогу $\rho_{\max}$ сформована псевдовипадкова послідовність бракується, тобто дискретні сигнали Φ_j із $\rho\left(C_i,\Phi_j\right)>\rho_{\max}$ хоча б для одного $i\in\{0,..,N-1\}$ для стеганографічного приховування інформаційних даних не застосовуються.

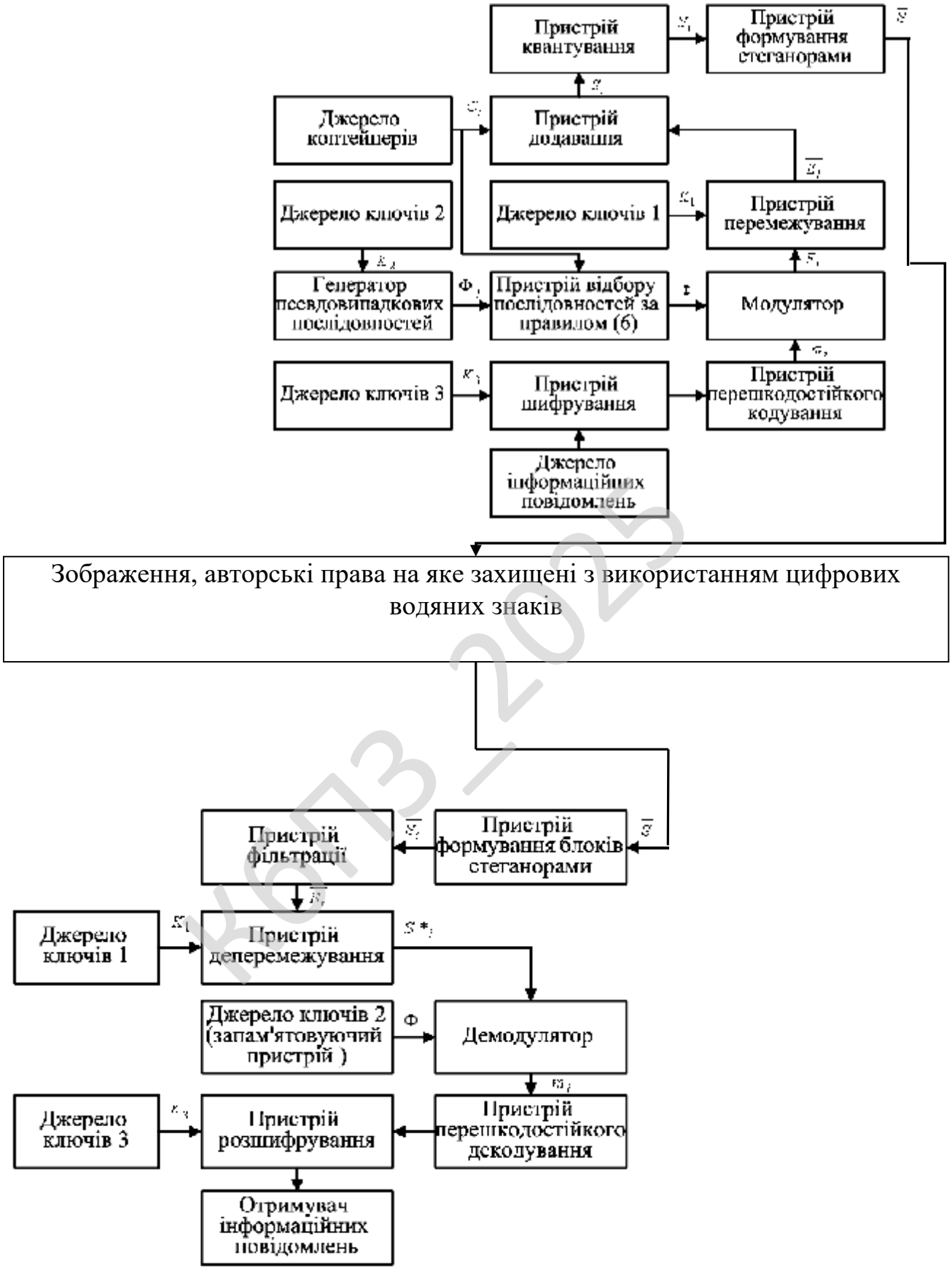

Рисунок 3.1 – Структурна схема системи

Якщо для сформованого дискретного сигналу Φ_j та для всіх $i = 0,..,N-1$ розраховані значення коефіцієнту кореляції $\rho(C_i, \Phi_j)$ менші або дорівнюють встановленому порогу $\rho_{\max}$, тобто, якщо виконується умова (3.14) для всіх блоків даних контейнеру, відповідне значення Φ_j приймається до подальшого стеганографічного приховування інформаційних даних. Сформовані таким чином псевдовипадкові послідовності складають ансамбль дискретних сигналів $\Phi = \{\Phi_0, \Phi_1, ..., \Phi_{M-1}\}$, вони враховують статистичні властивості контейнера та подаються до модулятора. На модулятор подається також блок інформаційних даних $m_i = (m_{i_0}, m_{i_1}, ..., m_{i_{k-1}})$, $k \le M$, який модулюється псевдовипадковими послідовностями за правилом (3.9). Сформоване таким чином модульоване повідомлення E_i подається на пристрій перемежування, ініційованого таємним ключем K_1, який сформовано джерелом ключів 1. Пристрій перемежування обробляє модульоване повідомлення E_i, тобто за правилом f, яке задає таємний ключ K_1, псевдовипадковим чином переставляє місцями елементи E_i. Отримані дані $\overline{E_i} = f(E_i, K_1)$ подаються на пристрій додавання, у якому виконується поелементне додавання з даними контейнеру C_i (з даними цифрового зображення в просторовій області): $S_i = C_i + \overline{E_i} \cdot G$, де $G > 0$ – коефіцієнт підсилення розширювального сигналу, який задає «енергію» вбудованих блоків інформаційного повідомлення. Контейнер формується джерелом контейнерів. Отримані дані S_i подаються на пристрій квантування, який виконує певне перетворення для зберігання початкового динамічного діапазону зображення-контейнеру, в результаті чого формуються окремі блоки стеганограми $\overline{S_i}$ та заповнений контейнер $\overline{S} = \overline{S_0} \cup \overline{S_1} \cup ... \cup \overline{S_{N-1}}$, який передається приймальній стороні.

Пристрій стеганографічного вилучення даних (див. рис. 3.1) працює аналогічно відповідному пристрою як і у відомому способі [1-3], за винятком генератора псевдовипадкових послідовностей, що формує відповідні дискретні сигнали. Пристрій функціонує наступним чином. Отримана стеганограмма $\overline{S}$ на

приймальній стороні подається на пристрій формування блоків стеганограми, в якому формуються блоки $\overline{S_i}$ та подаються на пристрій фільтрації. Після фільтрації отримані дані $\overline{\overline{S_i}}$ подаються на пристрій зворотного перемежування, на якому виконується дія, інверсна перемежуванню на передавальній стороні. Пристрій деперемежування ініційовано секретним ключем K_1, який сформовано джерелом ключів 1. Отримані після деперемежування дані S^*_i подаються на демодулятор, який виконує функцію кореляційного приймача дискретних сигналів за розглянутим вище правилом. Тобто в демодуляторі обчислюється значення коефіцієнту кореляції даних S^*_i та псевдовипадкових послідовностей Φ_i (дискретних сигналів). Ансамбль дискретних сигналів не формується (як у відомому способі) відповідним генератором псевдовипадкових чисел, що ініційований секретним ключем K_2, а зберігається у запам'ятовуючому пристрою, тобто весь ансамбль псевдовипадкових послідовностей $\Phi = \{\Phi_0, \Phi_1, ..., \Phi_{M-1}\}$ виступає у ролі секретного ключа K_2 і зберігається цілком в такому вигляді. Таким чином, запам'ятовуючий пристрій можна розглядати як аналог джерела ключів 2 у відомому способі [3], а послідовності, які надходять з нього є тотожними тим, які застосовуються на передавальній стороні при вбудовуванні інформаційних повідомлень. Таким чином, в демодуляторі обчислюється значення коефіцієнту кореляції між отриманими даними S^*_i та послідовностями, які застосовувалися при вбудовуванні інформації. Рішення, стосовно значення вбудованих даних, приймається відповідно до значення обрахованого коефіцієнту кореляції за правилом (5). Вилучені дані m_i подаються на пристрій перешкодостійкого декодування, в якому за визначеним правилом із використанням внесеної надмірності виправляються деякі помилки, відповідно до корегуючої здатності коду. Це призводить до деякого підвищення достовірності переданих даних, які після декодування подаються на пристрій розшифрування, ініційованого таємним ключем K_3, що формується джерелом ключів 3. Розшифровані повідомлення подаються отримувачу інформаційних повідомлень.

3.3 Розробка функціональної схеми

Технологія застосування системи припускає наявність мережі абонентів, що посилають один одному підписані фотографії з мікропейментового фотобанку. Для кожного абонента генерується пара ключів: закритий і відкритий.

Закритий ключ зберігається абонентом в таємниці і використовується їм для формування ЕЦП.

Відкритий ключ відомий всім іншим користувачам мікропейментового фотобанку і призначений для перевірки ЕЦП одержувачем підписаної фотографії з мікропейментового фотобанку. Відкритий ключ не дозволяє обчислити секретний ключ. Кожен легальний користувач може за допомогою відкритого ключа перевірити достовірність і незмінність електронного документу.

Цифровий водяний знак, прихований у файлі, служить гарантом того, що навіть якщо зловмисник підбере закритий ключ і підпише файл, результати перевірки підпису і ЦВЗ не співпадуть і можна буде встановити порушення. ЦВЗ виступає як додатковий рівень захисту, який іноді важко навіть виявити, а тим більше обійти.

Розроблена система складається з наступних модулів:

– генерація ключів (датчик псевдо-випадкових чисел);

– хеш-функція SHA-1;

– криптографічний модуль (алгоритми RSA та DES);

– стеганографічний модуль (для вбудовування ЦВЗ).

Модуль генерації ключів створює для користувача закритий та відкритий RSA ключі, та надсилає запит на сертифікацію відкритого ключа. У разі успішного пройдешня сертифікації, відкритий ключ, термін його придатності та інформація про власника зберігаються в базі даних (репозиторії), для того, щоб одержувачі підписаних документів могли перевірити його автентичність.

Хеш-функція та криптографічний модуль використовуються при створенні електронного цифрового підпису та при його перевірці. Також криптографічний

модуль задіяно при шифруванні та дешифруванні документу. Асиметричний алгоритм RSA використовується для створення підпису, а блочний алгоритм DES для шифрування.

Стеганографічний модуль необхідний для вбудовування цифрового водяного знаку в графічне зображення відсканованого документу та для його вилучення з документу і перевірки.

Центр сертифікації ключів (англ. Certification authority, CA) – це організація, яка надає послуги електронного цифрового підпису, зокрема:

– Надання у користування засобів електронного цифрового підпису.

– Допомога при генерації відкритих та особистих ключів.

– Обслуговування сертифікатів ключів:

– формування;

– розповсюдження ;

– скасування;

– зберігання;

– блокування;

– Надання інформації щодо чинних, скасованих і блокованих сертифікатів відкритих ключів.

– Послуги фіксування часу.

– Консультації та інші послуги.

Головним призначенням центру сертифікації є засвідчення автентичності відкритих ключів користувачів.

Центр сертифікації ключів, акредитований в установленому порядку, є акредитованим центром сертифікації ключів. Відмінністю акредитованого центру є те, що він має право обслуговувати виключно посилені сертифікати ключів.

Акредитований центр сертифікації ключів має виконувати усі зобов'язання та вимоги, встановлені законодавством для центру сертифікації ключів, та

додатково зобов'язаний використовувати для надання послуг електронного цифрового підпису надійні засоби електронного цифрового підпису.

Порядок акредитації та вимоги, яким повинен відповідати акредитований центр сертифікації ключів, встановлюються Кабінетом Міністрів України.

Відкриті ключі й інша інформація про користувачів зберігається центрами сертифікації, у вигляді цифрових сертифікатів, що мають наступну структуру:

– серійний номер сертифіката;

– об'єктний ідентифікатор алгоритму електронного підпису;

– ім'я центру, що засвідчує;

– строк придатності;

– ім'я власника сертифіката (ім'я користувача, якому належить сертифікат);

– відкритий ключ власника сертифіката;

– об'єктні ідентифікатори алгоритмів, асоційованих з відкритими ключами власника сертифіката.

Рисунок 3.2 – Функціональна схема роботи системи

Система включає дві процедури:

– процедуру захисту документу цифровими водяними знаками, електронним підписом та шифруванням;

– процедуру розшифрування та перевірки легітимності документу, шляхом перевірки ЕЦП та ЦВЗ.

У процедурі створення підпису використовується закритий ключ RSA відправника, в процедурі перевірки підпису – відкритий ключ RSA відправника. При вбудовуванні ЦВЗ в документ застосовується стегоключ, а при шифруванні секретний DES ключ, щоб забезпечити секретність цих ключів, вони шифруються відкритим ключем RSA одержувача, після чого лише він може розшифрувати їх своїм закритим RSA ключем.

Принциповим моментом в системі ЕЦП є неможливість підробки електронного підпису користувача без знання його секретного ключа.

Кожен підпис містить наступну інформацію:

– дату підпису;

– термін закінчення дії ключа даного підпису;

– інформацію про особу, що підписала файл (П.І.Б., посада, коротке найменування фірми);

– ідентифікатор того, хто підписав (ім'я відкритого ключа);

– власне цифровий підпис.

Перед відправкою файлу по мережі, в нього вбудовується ЦВЗ. При формуванні ЕЦП відправник перш за все обчислює хеш-функцію h(M) документу M, що слід підписати.

Обчислене значення хеш-функції h(M) являє собою один короткий блок інформації m, що характеризує весь документ M в цілому. Потім число m

шифрується секретним ключем відправника. Отримувана при цьому пара чисел є ЕЦП для даного документу М.

Далі документ шифрується і формується електронний конверт, що містить: зашифрований файл з попередньо вбудованими водяними знаками, підпис та зашифровані стего– та DES ключі. Електронний конверт передається по мережі, де може відбутися атака зловмисника на нього з метою підробки чи модифікації його змісту.

Прийнятий по каналу зв'язку документ М розшифровується, одержувач повідомлення, знову обчислює його хеш-функцію m=h(M), після чого за допомогою відкритого ключа відправника перевіряє, чи відповідає отриманий підпис обчисленому значенню m хеш-функції. Потім дешифрує своїм закритим ключем стегоключ, виймає з файлу стеговставку та порівнює її з шаблоном. Якщо перевірка підпису та цифрових водяних знаків пройшла успішно, то документ легітимний.

Хеш-функція призначена для стиснення підписуваного документа М до декількох десятків або сотень біт. Хеш-функція h() приймає як аргумент документ М довільного розміру і повертає хеш-значення h(M)=H фіксованої довжини. Зазвичай хеш-значення є стислим двійковим представленням основного повідомлення довільної довжини. Слід зазначити, що значення хеш-функції h(M) складним чином залежить від документа М і не дозволяє відновити сам документ М.

Хеш-функція повинна задовольняти цілому ряду умов:

– хеш-функція повинна бути чутлива до всіляких змін в тексті М, таких як вставки, викиди, перестановки і т.п.;

– хеш-функція повинна мати властивість безповоротності, тобто підбір документу М', який би мав необхідне значення хеш-функції, повинний бути неможливим;

– вірогідність того, що значення хеш-функцій двох різних документів (незалежно від їх довжин) співпадуть, повинна бути нікчемно малою.

Тобто хеш-функцією є таке математичне або алгоритмічне перетворення заданого блоку даних, яке володіє наступними властивостями:

– нескінченна область визначення;

– кінцева область значень;

– необоротність;

– зміна вхідного потоку інформації на 1 біт змінює близько половини всіх біт вихідного потоку.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

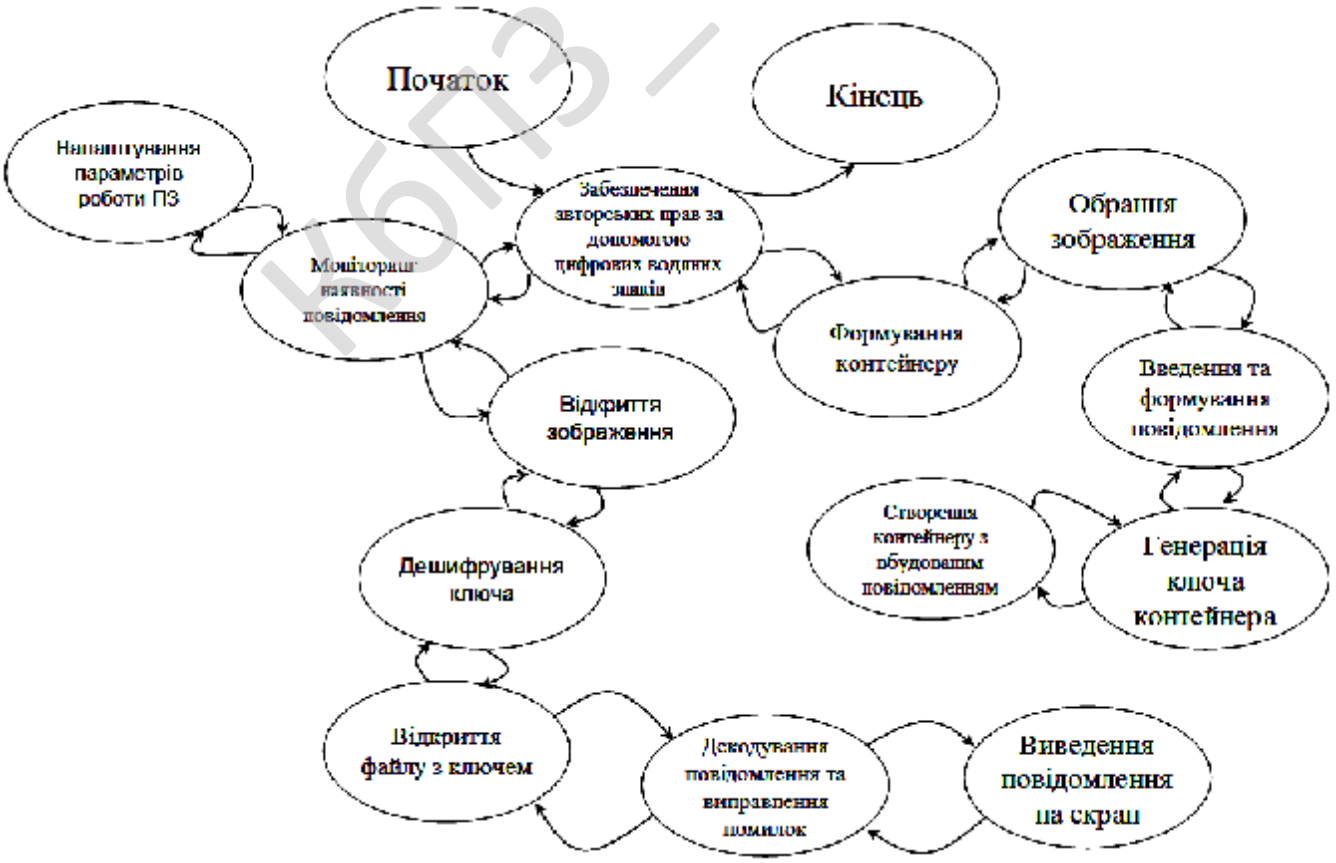

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи забезпечення авторських прав за допомогою цифрових водяних знаків.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

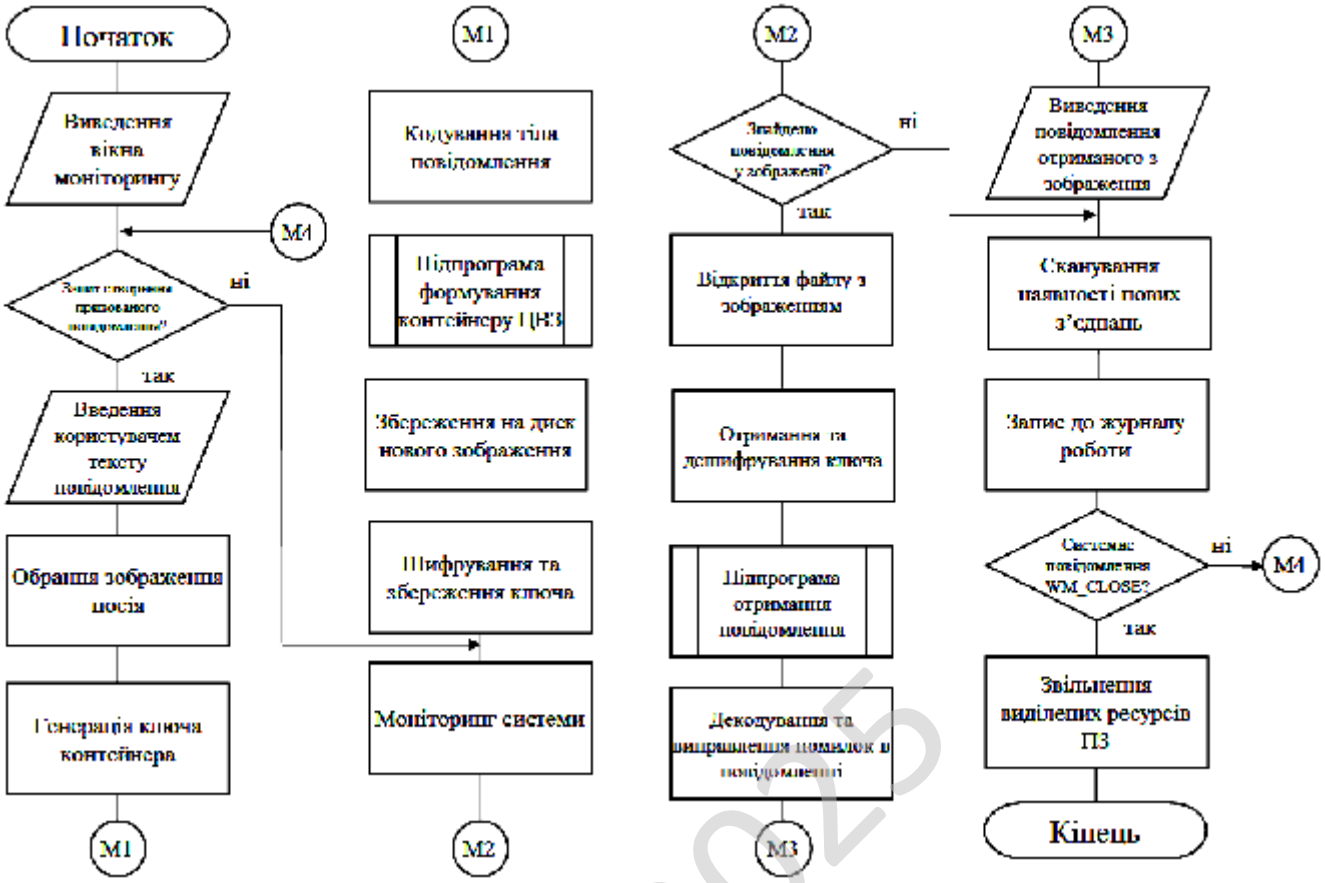

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм.

Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

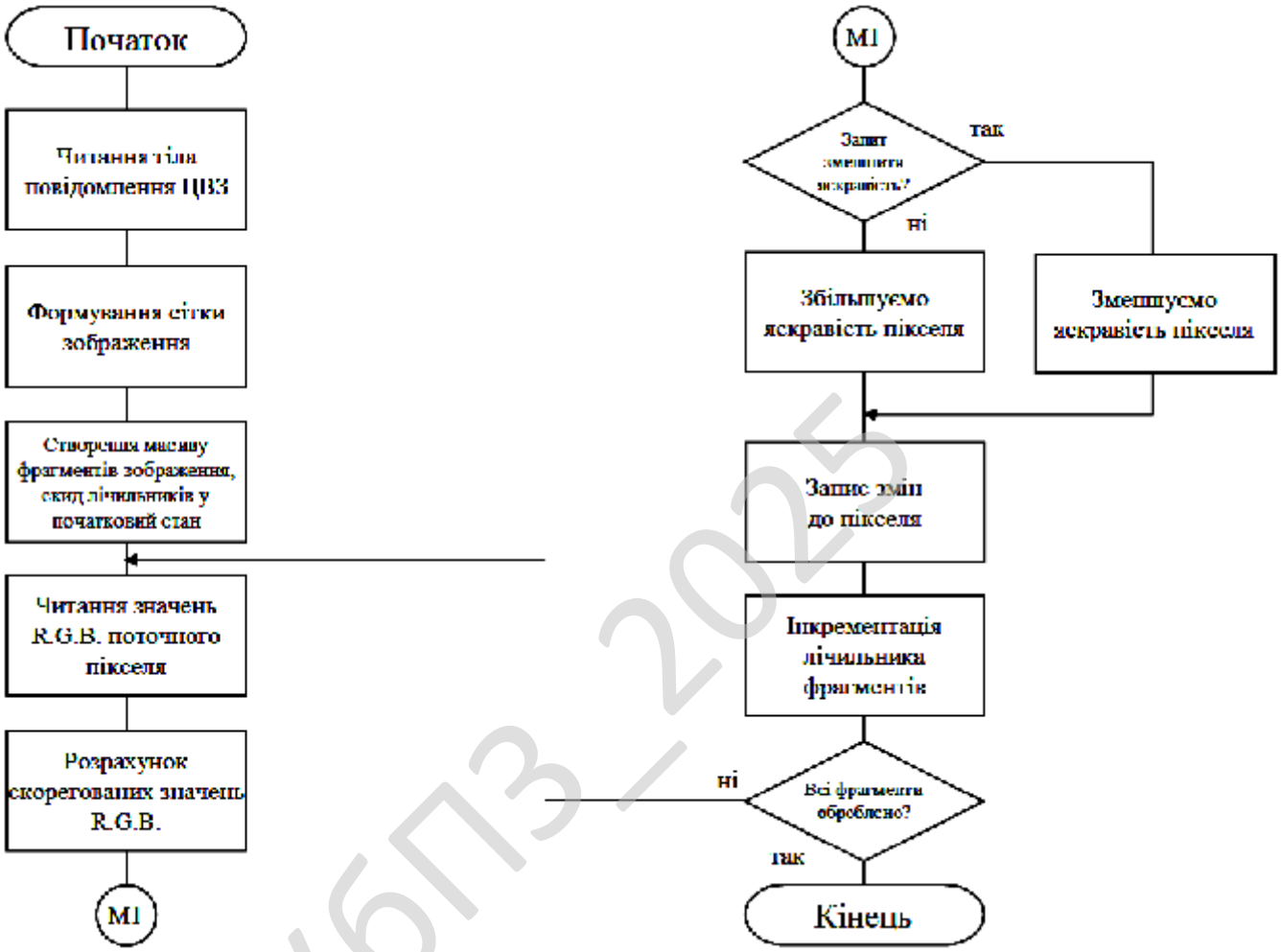

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною

одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого

програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку

можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Опис ситеми

Система забезпечення авторських прав за допомогою цифрових водяних знаків реалізує захист мультимедійного контенту від несанкціонованого використання.

Ця система використовує цифрові водяні знаки, які вбудовуються у зображення, аудіофайли або відеофайли, забезпечуючи ідентифікацію власника авторських прав. Основним середовищем розробки є мова програмування Python.

Архітектура системи складається з основних компонентів.

Перший компонент – модуль вбудовування водяного знака. Він забезпечує модифікацію вихідного зображення шляхом додавання у нього спеціального маркера.

Другий компонент – модуль витягування водяного знака, який аналізує зображення та ідентифікує вбудований маркер.

Третій компонент – модуль перевірки автентичності, що визначає, чи було внесено зміни у зображення після його розповсюдження.

Основні модулі системи. OpenCV використовується для обробки зображень. NumPy застосовується для маніпуляцій з матрицями та векторами. PyWavelets використовується для роботи з дискретним вейвлет-перетворенням. Cryptography забезпечує шифрування водяного знака для захисту від підробки.

Функціональність основних модулів

Модуль вбудовування водяного знака застосовує метод дискретного косинусного або вейвлет-перетворення для приховання інформації у високочастотних або середньочастотних ділянках зображення. Це робить водяний знак невидимим для людського ока, але стійким до базових атак, таких як стиснення JPEG або зміна яскравості. Модуль витягування водяного знака застосовує зворотне перетворення для отримання прихованої інформації. Модуль перевірки порівнює отриманий водяний знак із оригіналом та оцінює рівень його спотворення.

Алгоритм роботи системи

Зображення завантажується та перетворюється у числову матрицю. Виконується дискретне косинусне або вейвлет-перетворення. До вибраних коефіцієнтів додається закодований водяний знак.

Змінені коефіцієнти повертаються у початковий формат, після чого формується новий файл із вбудованим маркером.

Розрахунок ефективності

Для оцінки коректності роботи системи використовуються метрики PSNR та SSIM. Для тестового зображення 512x512 результати показують, що середнє значення PSNR перевищує 40 дБ, що означає низький рівень помітності водяного знака.

Метрика SSIM у середньому перевищує 0.98, що вказує на високу якість збереження візуальних характеристик зображення. Для оцінки витривалості водяного знака тестуються різні атаки, такі як стиснення JPEG з коефіцієнтом 50, фільтрація та поворот. У більшості випадків водяний знак залишається розпізнаваним.

Як результат система забезпечує захист авторських прав шляхом додавання стійкого водяного знака у зображення. Вона дозволяє ідентифікувати власника контенту та перевіряти автентичність матеріалів. Використання методів вейвлет-перетворення та шифрування підвищує безпеку та ефективність алгоритму.

Частина коду

```
import cv2
import numpy as np
import pywt
from cryptography.fernet import Fernet

# Генерація ключа шифрування
key = Fernet.generate_key()
cipher = Fernet(key)

def embed_watermark(image_path, watermark_text, output_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    coeffs2 = pywt.dwt2(image, 'haar')
    LL, (LH, HL, HH) = coeffs2

binary_wm = ''.join(format(ord(char), '08b') for char in watermark_text)
    index = 0

    for i in range(HH.shape[0]):
        for j in range(HH.shape[1]):
```

```
            if index < len(binary_wm):
HH[i, j] = HH[i, j] - (HH[i, j] % 2) + int(binary_wm[index])
                index += 1

    image_with_wm = pywt.idwt2((LL, (LH, HL, HH)), 'haar')
    cv2.imwrite(output_path, np.uint8(image_with_wm))

def extract_watermark(image_path, watermark_length):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    coeffs2 = pywt.dwt2(image, 'haar')
    LL, (LH, HL, HH) = coeffs2

    binary_wm = ''
    for i in range(HH.shape[0]):
        for j in range(HH.shape[1]):
            if len(binary_wm) < watermark_length * 8:
                binary_wm += str(int(HH[i, j] % 2))
            else:
                break

watermark_text = ''.join(chr(int(binary_wm[i:i+8], 2)) for i
                 in range(0, len(binary_wm), 8))
    return watermark_text

# Вбудовування водяного знака
embed_watermark('input.jpg', 'Protected', 'watermarked.jpg')

# Витягування водяного знака
extracted_wm = extract_watermark('watermarked.jpg', len('Protected'))
print('Extracted Watermark:', extracted_wm)
```

Це приклад коду що реалізує:

1. Вбудовування водяного знака – перетворює зображення за допомогою дискретного вейвлет-перетворення, змінює високочастотні коефіцієнти для кодування текстового водяного знака.

2. Витягування водяного знака – аналізує зображення, відновлює бітовий рядок та декодує текстовий водяний знак.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою MARS, який є блочно-симетричним шифром з відкритим ключем. Розмір блоку при шифруванні 128 біта, розмір ключа може варіюватися від 128 до 448 біт включно (кратні 32бітам). Творці прагнули поєднати в своєму алгоритмі швидкість кодування і стійкість шифру. В результаті вийшов один з самих криптостійкий алгоритм з алгоритмів, які брали участь в конкурсі AES.

Алгоритм унікальний тим, що використовував практично всі існуючі технології, застосовувані в криптоалгоритмах, а саме:

– Найпростіші операції (додавання, віднімання, виключаюче або).
– Підстановки з використанням таблиці замін.
– Фіксований циклічний зсув.
– Залежний від даних циклічний зсув.
– Множення за модулем 2^{32}.
– Ключове забілювання.

Використання подвійного перемішування представляє складність для криптоаналіз а, що деякі відносять до недоліків алгоритму. У той же час на даний момент не існує будь-яких ефективних атак на алгоритм, хоча деякі ключі можуть генерувати слабкі підключі.

Структура алгоритму

Автори шифру виходили з наступних припущень:

1. Вибір операцій. MARS був спроектований для використання на найсучасніших комп'ютерах того часу. Для досягнення найкращих захисних характеристик в нього були включені самі «сильні операції» підтримувані в них. Це дозволило добитися більшого відносини securityper-instruction для різних реалізації шифру.

2. Структура шифру. Двадцятирічний досвід роботи в області криптографії підштовхнув творців алгоритму до думки, що кожен раунд

шифрування грає свою роль в забезпеченні безпеки шифру. Зокрема, ми можемо бачити, що перший і останній раунди зазвичай сильно відрізняються від проміжних («центральних») раундів алгоритму в плані захисту від криптоаналітичних атак. Таким чином, при створенні MARSa використовувалася змішана структура, де перший і останній раунди шифрування істотно відрізняються від проміжних.

3. Аналіз. Швидше за все, алгоритм з гетерогенною структурою буде краще протистояти криптоаналітичних методам майбутнього, ніж алгоритм, всі раунди якого ідентичні. Розробники алгоритму MARS надали йому сильно гетерогенну структуру – раунди алгоритму дуже різняться між собою.

У шифрі MARS використовувалися такі методи шифрування:

1. Робота з 32-х бітними словами. Всі операції застосовуються до 32-бітовим словами. тобто вся початкова інформація розбивається на блоки по 32біта. (Якщо ж блок опинявся меншої довжини, то він доповнювався до 32біт)

2. Мережа Фейстеля. Творці шифру вважали, що це найкращий варіант поєднання швидкості шифрування і криптостійкості. В MARS використана мережа Фейстеля 3-го типу.

3. Симетричність алгоритму. Для стійкості шифру до різних атакам всі його раунди були зроблені повністю симетричними, тобто друга частина раунду є дзеркальне повторення першої його частини.

Структуру алгоритму MARS можна описати таким чином:

1. Попереднє накладення ключа: на 32-бітові субблоки A, B, C, D накладаються 4 фрагмента розширеного ключа k_0 … k_3 операцією складання за модулем 2^{32}.

2. Виконуються 8 раундів прямого перемішування (без участі ключа шифрування).

3. Виконуються 8 раундів прямого криптоперетворення.

4. Виконуються 8 раундів зворотного криптоперетворення. [2]

5. Виконуються 8 раундів зворотного перемішування, також без участі ключа шифрування.

6. Фінальне накладення фрагментів розширеного ключа k_{36} … k_{39} операцією віднімання за модулем 2^{32}.

Пряме перемішування

У першій фазі на кожне слово даних накладається слово ключа, а потім відбувається вісім раундів змішування згідно з мережею Фейстеля третього типу спільно з деякими додатковими змішування. У кожному раунді ми використовуємо одне слово даних (зване, вихідним словом) для модифікації трьох інших слів (звані, цільовими словами). Ми розглядаємо чотири байта вихідного слова як індексів на двох S-блоків, S_0 і S_1 , кожен, що складається з 256 32-розрядних слів, а далі проводимо операції XOR або додавання даних відповідного S-блоку в три інших слова.

Якщо чотири байти вихідного слова b_0 , b_1 , b_2, b_3 (де b_0 є першим байтом, а b_3 є старшим байтом), то ми використовуємо b_0 , b_2, як індекси в блоку S_0 і байти b_1 , b_3 , як індекси в S-блоці S_1 . Спочатку зробимо XOR S_0 до першого цільовим речі, а потім додамо S_1 до того ж слова. Ми також додаємо S_0 до другого цільовим слову і XOR блоку-S_1 до третього цільовим слову. У висновку, ми обертаємо вихідне слово на 24 біта вправо.

У наступному раунді ми обертаємо наявні у нас чотири слова: таким чином, нинішні перші цільове слово стає наступним вихідним словом, поточним другий цільове слово стає новим першим цільовим словом, третє цільове слово стає наступний другий цільовим словом, і поточне вихідне слово стає третім цільовим словом.

Більш того, після кожного з чотирьох раундів ми додаємо одне з цільових слів назад у вихідне слово. Зокрема, після першого і п'ятого раундів ми додав третю цільове слово назад у вихідне слово, а після другого і шостого раунду ми додаємо першої цільової слово назад у вихідне слово. Причиною цих додаткових операцій змішування, є ліквідація декількох простих диференціальних

криптоатаки в фазі перемішування, щоб порушити симетрію у фазі змішування та отримати швидкий потік.

Криптографічне ядро

Криптографічне ядро MARS – мережа Фейстеля 3-го типу, що містить в собі 16 раундів. У кожному раунді ми використовуємо ключову E-функцію, яка є комбінацією множень, обертань, а також звернень до S-блоків. Функція приймає на вхід слово даних, а повертає три слова, з якими згодом буде здійснена операція додавання або XOR до інших трьох слів даними. У доповненні вихідне слово обертається на 13 біт вліво.

Для забезпечення, серйозного опору до криптоатаки, три вихідних значення E-функції (O_1 , O_2, O_3) використовуються в перших восьми раундах і в останніх восьми раундах в різних порядках. У перші вісім раундів ми додаємо O_1 і O_2до першого і другого цільовим речі, відповідно, і XOR O_3 у третьому цільовим слову. За останні вісім раундів, ми додаємо O_1 і O_2 до третього і другого цільовим речі, відповідно, і XOR O_3 до першого цільовим слову.

E-функція

E-функція приймає як вхідні дані одне слово даних і використовує ще два ключових слова, виробляючи на виході три слова. У цій функції ми використовуємо три тимчасові змінні, що позначаються L, M і R (для лівої, середньої та правої).

Спочатку ми встановлюємо в R значення вихідного слова зміщеного на 13 біт вліво, а в M – сума вихідних слів і першого ключового слова. Потім ми використовуємо перші дев'ять бітів M як індекс до однієї з 512S-блоків (яке виходить суміщенням S_0 і S_1 змішуванням фази), і зберігаємо в L значення відповідного S-блоку.

Потім помножимо друге ключове слово на R, зберігши значення в R. Потім обертаємо R на 5 позицій вліво (так, 5 старших бітів стають 5 нижніми бітами R після обертання). Тоді ми робимо XOR R в L, а також переглядаємо п'ять нижніх біт R для визначення величини зсуву (від 0 до 31), і обертаємо M

вліво на цю величину. Далі ми обертаємо R ще на 5 позицій вліво і робимо XOR в L. У висновку, ми знову дивимося на 5 молодших бітів R, як на величину обертання і обертаємо L на цю величину вліво. Таким чином результат роботи E-функції – 3 слова (по порядку): L, M, R.

Зворотне перемішування

Зворотне перемішування практично збігається з прямим перемішуванням, за винятком того факту, що дані обробляються в зворотному порядку. Тобто, якби ми поєднали пряме і зворотне перемішування так, щоб їх виходи і входи були б з'єднані в зворотному порядку (D [0] прямого і D [3] зворотного, D [1] прямого і D [2] зворотного), то не побачили б результату перемішування. Як і в прямому змішування, тут ми теж використовуємо одне вихідне слово і три цільових. Розглянемо чотири перших байта вихідного слова: b_0 , b_1 , b_2, b_3 . Будемо використовувати b_0 , b_2як індекс до S-блоку – S_1 , а b_1 b_3 для S_0 . Зробимо XOR S_1 [b_0] в перше цільове слово, віднімемо S_0 [b_3] з другого слова, віднімемо S_1 [b_2] з третього цільового слів і потім проробимо XOR S_0 [b_1] також до третього цільового слова. Нарешті, ми повертаємо початкове слово на 24 позицій вліво. Для наступного раунду ми обертаємо наявні слова так, щоб нинішнє перше цільове слово стало наступним вихідним словом, поточне друге цільове слово стало першим цільовим словом, поточне третє цільове слово стало другим цільовим словом, і поточне вихідне слово стало третім цільовим словом. Крім того, перед одним з чотирьох «особливих» раундів ми віднімаємо одне з цільових слів з вихідного слова: перед четвертим і восьмим раундами ми віднімаємо першої цільової слово, перед третьому і сьомим раундами ми віднімемо третя цільова слово з вихідного.

Дешифрування

Процес декодування обернений процесу кодування. Код дешифрування схожий (але не ідентичний) на код шифрування.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків складається з наступних функціональних блоків:

– Навігаційне меню: Дані; Шаблони; Налаштування; Довідка.

– Підвікно виведення отриманого зображення, з можливістю виділення блоків маркування.

– Вікна обрання групи .

– Вікно виведення результату роботи системи.

– Функціональних кнопок ПЗ.

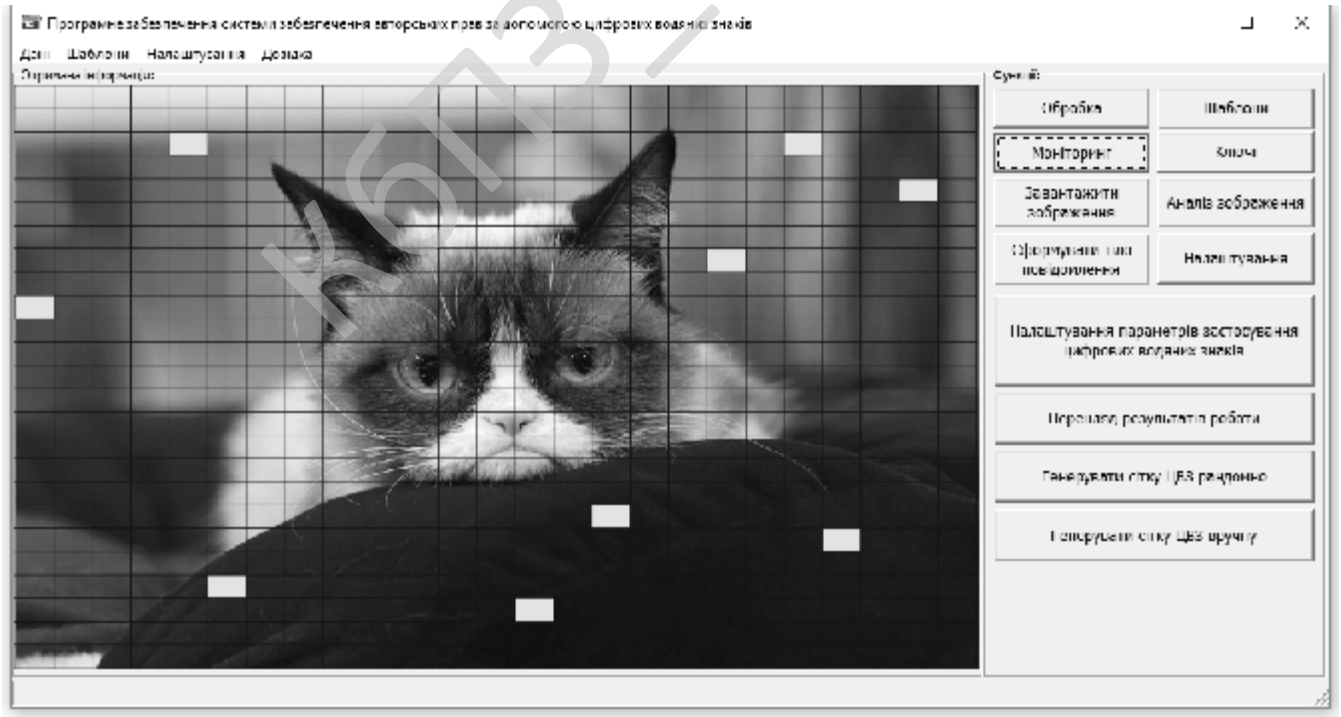

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

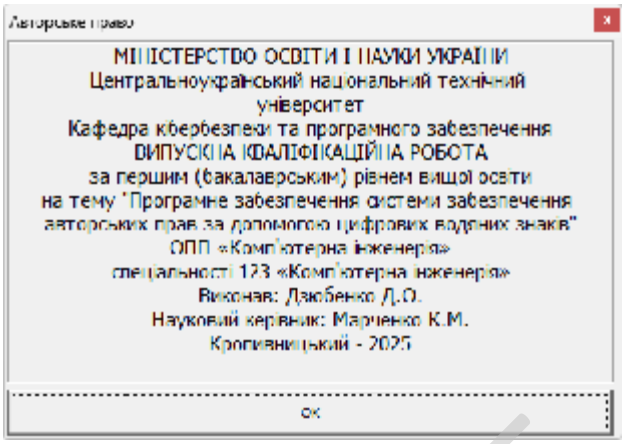

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи забезпечення авторських прав за допомогою цифрових водяних знаків.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем забезпечення авторських прав за допомогою цифрових водяних знаків.

– Досліджена система забезпечення авторських прав за допомогою цифрових водяних знаків.

– На основі отриманих результатів досліджень створена програмна реалізація системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання забезпечення авторських прав за допомогою цифрових водяних знаків.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи забезпечення авторських прав за допомогою цифрових водяних знаків.

Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм MARS.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

2. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

3. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

4. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

5. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

6. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

7. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

8. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

9. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

10. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

11. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

12. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

13. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with

Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

14. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

15. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

16. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

17. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

18. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

19. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

20. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

21. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

22. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

23. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

24. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

25. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

26. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

27. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

28. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

29. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

30. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

31. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

32. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

33. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

34. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

35. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

36. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

37. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

38. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

39. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

40. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

41. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

42. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

43. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of

Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

44. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

45. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

46. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

47. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

48. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

49. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

50. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

51. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

52. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

53. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи забезпечення авторських прав за допомогою цифрових водяних знаків.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи забезпечення авторських прав за допомогою цифрових водяних знаків;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 76 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Марченко К.М.

Програмне забезпечення системи забезпечення авторських прав за допомогою цифрових водяних знаків

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 17

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import cv2
import numpy as np
import os
import random
import string
import sqlite3
import rsa
import qrcode
import boto3
from flask import Flask, request
from pydub import AudioSegment
from PyPDF2 import PdfReader, PdfWriter
from reportlab.pdfgen import canvas
from cryptography.fernet import Fernet

def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)

def load_key():
    return open("secret.key", "rb").read()

def encrypt_message(message, key):
    cipher = Fernet(key)
    encrypted_message = cipher.encrypt(message.encode())
    return encrypted_message

def decrypt_message(encrypted_message, key):
    cipher = Fernet(key)
    decrypted_message = cipher.decrypt(encrypted_message).decode()
    return decrypted_message

def generate_random_watermark(length=10):
    return ''.join(random.choices(string.ascii_uppercase + string.digits,
k=length))

def embed_text_watermark(image_path, text, output_path):
    image = cv2.imread(image_path)
    height, width, _ = image.shape
    font_scale = min(width, height) / 500
    thickness = max(1, int(font_scale * 2))
    position = (width - 200, height - 50)
    cv2.putText(image, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
(255, 255, 255), thickness, cv2.LINE_AA)
    cv2.imwrite(output_path, image)

def embed_audio_watermark(audio_path, watermark_text, output_path):
    audio = AudioSegment.from_file(audio_path)
    print(f"Вбудовування водяного знака '{watermark_text}' у {audio_path}")
    audio.export(output_path, format="wav")

def create_watermark_database():
    conn = sqlite3.connect("watermarks.db")
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS watermarks (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        image_path TEXT,
                        watermark TEXT)''')
    conn.commit()
    conn.close()
```

```
def save_watermark(image_path, watermark_text):
    conn = sqlite3.connect("watermarks.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO watermarks (image_path, watermark) VALUES (?,
?)", (image_path, watermark_text))
    conn.commit()
    conn.close()

def upload_to_cloud(file_path, bucket_name):
    s3 = boto3.client("s3")
    s3.upload_file(file_path, bucket_name, file_path.split("/")[-1])

def register_watermark_in_blockchain(image_path, watermark_text):
    print(f"Реєстрація водяного знака '{watermark_text}' у блокчейні.")

def detect_duplicate_watermark(image1_path, image2_path):
    img1 = cv2.imread(image1_path, cv2.IMREAD_GRAYSCALE)
    img2 = cv2.imread(image2_path, cv2.IMREAD_GRAYSCALE)
    difference = np.sum(img1 != img2)
    return difference == 0

def generate_qr_code(text, output_path):
    qr = qrcode.QRCode(version=1, box_size=10, border=5)
    qr.add_data(text)
    qr.make(fit=True)
    img = qr.make_image(fill="black", back_color="white")
    img.save(output_path)

def main():
    generate_key()
    key = load_key()
    original_image = "input.jpg"
    watermarked_image = "output_text.jpg"
    watermark_text = generate_random_watermark()
    encrypted_watermark = encrypt_message(watermark_text, key)
    embed_text_watermark(original_image, watermark_text, watermarked_image)
    save_watermark(original_image, watermark_text)
    generate_qr_code(watermark_text, "watermark_qr.png")
    upload_to_cloud(watermarked_image, "your-bucket-name")
    register_watermark_in_blockchain(original_image, watermark_text)
    decrypted_watermark = decrypt_message(encrypted_watermark, key)
    print("Дешифрований водяний знак:", decrypted_watermark)

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

Файл get_optimal_font_size.py

```
import cv2
import numpy as np
from PIL import Image, ImageFont

def get_optimal_font_size(image_path, text):
    # Завантаження зображення
    image = Image.open(image_path)
    width, height = image.size

    # Початковий розмір шрифту
    font_size = min(width, height) // 10
    font = ImageFont.truetype("arial.ttf", font_size)

    text_width, text_height = font.getsize(text)

    # Зменшення шрифту, якщо він не поміщається в зображення
    while text_width > width * 0.9 or text_height > height * 0.9:
        font_size -= 2
        font = ImageFont.truetype("arial.ttf", font_size)
        text_width, text_height = font.getsize(text)

    return font_size

def embed_adaptive_text_watermark(image_path, text, output_path):
    # Визначення оптимального розміру шрифту
    font_size = get_optimal_font_size(image_path, text)

    image = cv2.imread(image_path)
    width, height = image.shape[1], image.shape[0]

    font_scale = font_size / 20
    thickness = max(1, int(font_size / 15))

    position = (width - int(font_size * 5), height - int(font_size * 2))

    cv2.putText(image, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
(255, 255, 255), thickness, cv2.LINE_AA)
    cv2.imwrite(output_path, image)
```

Файл embed_text_with_position.py

```
def embed_text_with_position(image_path, text, output_path, position="bottom-
right"):
    image = cv2.imread(image_path)
    height, width, _ = image.shape

    font_scale = min(width, height) / 500
    thickness = max(1, int(font_scale * 2))

    text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
thickness)[0]

    positions = {
        "top-left": (10, 30),
        "top-right": (width - text_size[0] - 10, 30),
        "bottom-left": (10, height - 10),
        "bottom-right": (width - text_size[0] - 10, height - 10),
        "center": ((width - text_size[0]) // 2, (height - text_size[1]) // 2)
    }

    cv2.putText(image, text, positions.get(position, (10, 10)),
cv2.FONT_HERSHEY_SIMPLEX, font_scale, (255, 255, 255), thickness, cv2.LINE_AA)
    cv2.imwrite(output_path, image)
```

Файл embed_colored_transparent_watermark.py

```
def embed_colored_transparent_watermark(image_path, text, output_path,
color=(255, 255, 255), opacity=128):
    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
    overlay = image.copy()

    height, width, _ = image.shape
    font_scale = min(width, height) / 500
    thickness = max(1, int(font_scale * 2))

    position = (width - 200, height - 50)

    cv2.putText(overlay, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
color, thickness, cv2.LINE_AA)

    cv2.addWeighted(overlay, opacity / 255, image, 1 - opacity / 255, 0, image)
    cv2.imwrite(output_path, image)

def embed_text_with_font(image_path, text, output_path, font_path="arial.ttf",
font_size=20):
    image = cv2.imread(image_path)
    height, width, _ = image.shape

    font_scale = font_size / 20
    thickness = max(1, int(font_size / 15))

    position = (width - int(font_size * 5), height - int(font_size * 2))

    cv2.putText(image, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
(255, 255, 255), thickness, cv2.LINE_AA)
    cv2.imwrite(output_path, image)
```

Файл embed_random_watermark.py

```
import random
import string

def generate_random_watermark(length=10):
    return ''.join(random.choices(string.ascii_uppercase + string.digits,
k=length))

def embed_random_watermark(image_path, output_path):
    text = generate_random_watermark()
    embed_text_with_position(image_path, text, output_path, position="center")

def embed_layered_watermark(image_path, text, output_path):
    image = cv2.imread(image_path)

    height, width, _ = image.shape
    font_scale = min(width, height) / 500
    thickness = max(1, int(font_scale * 2))

    position = (width // 4, height // 4)

    cv2.putText(image, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
(255, 255, 255, 128), thickness, cv2.LINE_AA)

    cv2.imwrite(output_path, image)

def embed_lsb_watermark(image_path, text, output_path):
    image = cv2.imread(image_path)
    binary_text = ''.join(format(ord(i), '08b') for i in text)

    data_index = 0
    for row in image:
        for pixel in row:
            for channel in range(3):
                if data_index < len(binary_text):
                    pixel[channel] = pixel[channel] & ~1 |
int(binary_text[data_index])
                    data_index += 1

    cv2.imwrite(output_path, image)

def embed_watermark_video(video_path, text, output_path):
    cap = cv2.VideoCapture(video_path)
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(output_path, fourcc, 30.0, (int(cap.get(3)),
int(cap.get(4))))

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        cv2.putText(frame, text, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,
255, 255), 2)
        out.write(frame)

    cap.release()
    out.release()

def detect_watermark(image_path):
    print(f"Аналізуємо {image_path} на наявність водяного знака...")
```

```
def embed_complex_texture_watermark(image_path, text, output_path):
    print(f"Вставка водяного знака '{text}' у текстури зображення {image_path}")
```

Файл dct_watermark.py

```
import cv2
import numpy as np

def embed_dct_watermark(image_path, text, output_path):
    # Завантаження зображення у відтінках сірого
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Перетворення в плаваючий формат
    image = np.float32(image)

    # Застосування DCT
    dct_transformed = cv2.dct(image)

    # Вбудовування тексту у верхні частоти DCT
    for i, char in enumerate(text):
        dct_transformed[i][i] += ord(char)

    # Зворотне перетворення DCT
    image_with_watermark = cv2.idct(dct_transformed)

    # Конвертація у формат uint8
    image_with_watermark = np.uint8(image_with_watermark)

    # Збереження вихідного файлу
    cv2.imwrite(output_path, image_with_watermark)
```

Файл noise_protection.py

```
import cv2
import numpy as np

def embed_noise_protected_watermark(image_path, text, output_path):
    # Завантаження зображення
    image = cv2.imread(image_path)

    # Додавання тексту водяного знака
    height, width, _ = image.shape
    font_scale = min(width, height) / 400
    thickness = 2
    position = (width // 10, height // 10)

    cv2.putText(image, text, position, cv2.FONT_HERSHEY_SIMPLEX, font_scale,
(255, 255, 255), thickness, cv2.LINE_AA)

    # Генерація випадкового шуму
    noise = np.random.randint(0, 50, (height, width, 3), dtype=np.uint8)

    # Додавання шуму
    noisy_image = cv2.add(image, noise)

    # Збереження зображення
    cv2.imwrite(output_path, noisy_image)
```

Файл rsa_signature.py

```
import rsa

def generate_keys():
    # Генерація відкритого та закритого ключів
    public_key, private_key = rsa.newkeys(512)

    # Збереження ключів у файли
    with open("public.pem", "wb") as pub_file, open("private.pem", "wb") as
priv_file:
        pub_file.write(public_key.save_pkcs1())
        priv_file.write(private_key.save_pkcs1())

def sign_watermark(text):
    # Завантаження закритого ключа
    with open("private.pem", "rb") as priv_file:
        private_key = rsa.PrivateKey.load_pkcs1(priv_file.read())

    # Підписування тексту
    signature = rsa.sign(text.encode(), private_key, "SHA-256")

    return signature

def verify_signature(text, signature):
    # Завантаження відкритого ключа
    with open("public.pem", "rb") as pub_file:
        public_key = rsa.PublicKey.load_pkcs1(pub_file.read())

    # Перевірка підпису
    try:
        rsa.verify(text.encode(), signature, public_key)
        return True
    except rsa.VerificationError:
        return False
```

Файл database_watermark.py

```
import cv2
import os
import sqlite3
import numpy as np
import boto3

def create_watermark_database():
    # Підключення до бази даних
    conn = sqlite3.connect("watermarks.db")
    cursor = conn.cursor()

    # Створення таблиці
    cursor.execute('''CREATE TABLE IF NOT EXISTS watermarks (
                        id INTEGER PRIMARY KEY AUTOINCREMENT,
                        image_path TEXT,
                        watermark TEXT)''')

    conn.commit()
    conn.close()

def save_watermark(image_path, watermark_text):
    # Збереження водяного знака у базу
    conn = sqlite3.connect("watermarks.db")
    cursor = conn.cursor()

    cursor.execute("INSERT INTO watermarks (image_path, watermark) VALUES (?,
?)", (image_path, watermark_text))

    conn.commit()
    conn.close()

import qrcode

def generate_qr_code(text, output_path):
    # Генерація QR-коду
    qr = qrcode.QRCode(version=1, box_size=10, border=5)
    qr.add_data(text)
    qr.make(fit=True)

    # Збереження QR-коду
    img = qr.make_image(fill="black", back_color="white")
    img.save(output_path)

from PyPDF2 import PdfReader, PdfWriter
from reportlab.pdfgen import canvas

def add_pdf_watermark(input_pdf, output_pdf, watermark_text):
    # Створення PDF з водяним знаком
    watermark_pdf = "watermark.pdf"
    c = canvas.Canvas(watermark_pdf)
    c.setFont("Helvetica", 20)
    c.drawString(100, 500, watermark_text)
    c.save()

    # Додавання водяного знака у PDF
    reader = PdfReader(input_pdf)
    writer = PdfWriter()

    for page in reader.pages:
        page.merge_page(PdfReader(watermark_pdf).pages[0])
        writer.add_page(page)
```

```
    with open(output_pdf, "wb") as output_file:
        writer.write(output_file)

def apply_watermark_to_folder(folder_path, text):
    # Отримання списку файлів
    for filename in os.listdir(folder_path):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            output_path = os.path.join(folder_path, "watermarked_" + filename)
            embed_text_with_position(os.path.join(folder_path, filename), text,
output_path)

def enhance_watermark(image_path, output_path):
    image = cv2.imread(image_path)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
    enhanced = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
    cv2.imwrite(output_path, enhanced)

def detect_duplicate_watermark(image1_path, image2_path):
    img1 = cv2.imread(image1_path, cv2.IMREAD_GRAYSCALE)
    img2 = cv2.imread(image2_path, cv2.IMREAD_GRAYSCALE)
    difference = np.sum(img1 != img2)
    return difference == 0

def upload_to_cloud(file_path, bucket_name):
    s3 = boto3.client("s3")
    s3.upload_file(file_path, bucket_name, file_path.split("/")[-1])
```

Файл LSBSteg.py

```
# Freeware module

import cv2
import docopt
import numpy as np


class SteganographyException(Exception):
    pass


class LSBSteg():
    def __init__(self, im):
        self.image = im
        self.height, self.width, self.nbchannels = im.shape
        self.size = self.width * self.height

        self.maskONEValues = [1,2,4,8,16,32,64,128]

#Mask used to put one ex:1->00000001, 2->00000010 .. associated with OR bitwise
self.maskONE = self.maskONEValues.pop(0)
#Will be used to do bitwise operations

        self.maskZEROValues = [254,253,251,247,239,223,191,127]
#Mak used to put zero ex:254->11111110, 253->11111101 ..
# associated with AND bitwise
        self.maskZERO = self.maskZEROValues.pop(0)

        self.curwidth = 0
# Current width position
        self.curheight = 0
# Current height position
        self.curchan = 0
# Current channel position

    def put_binary_value(self, bits):
#Put the bits in the image
        for c in bits:
            val = list(self.image[self.curheight,self.curwidth])
#Get the pixel value as a list
            if int(c) == 1:
                val[self.curchan] = int(val[self.curchan]) | self.maskONE
#OR with maskONE
            else:
                val[self.curchan] = int(val[self.curchan]) & self.maskZERO #AND
with maskZERO

            self.image[self.curheight,self.curwidth] = tuple(val)
            self.next_slot()
#Move "cursor" to the next space

    def next_slot(self):
#Move to the next slot were information can be taken or put
        if self.curchan == self.nbchannels-1:
#Next Space is the following channel
            self.curchan = 0
            if self.curwidth == self.width-1:
#Or the first channel of the next pixel of the same line
                self.curwidth = 0
                if self.curheight == self.height-1:
#Or the first channel of the first pixel of the next line
```

```
                    self.curheight = 0
                    if self.maskONE == 128:
#Mask 1000000, so the last mask
raise SteganographyException("No available slot remaining (image filled)")
                    else:
#Or instead of using the first bit start using the second and so on..
                        self.maskONE = self.maskONEValues.pop(0)
                        self.maskZERO = self.maskZEROValues.pop(0)
                else:
                    self.curheight +=1
            else:
                self.curwidth +=1
        else:
            self.curchan +=1

    def read_bit(self):
#Read a single bit int the image
        val = self.image[self.curheight,self.curwidth][self.curchan]
        val = int(val) & self.maskONE
        self.next_slot()
        if val > 0:
            return "1"
        else:
            return "0"

    def read_byte(self):
        return self.read_bits(8)

    def read_bits(self, nb):
#Read the given number of bits
        bits = ""
        for i in range(nb):
            bits += self.read_bit()
        return bits

    def byteValue(self, val):
        return self.binary_value(val, 8)

    def binary_value(self, val, bitsize):
#Return the binary value of an int as a byte
        binval = bin(val)[2:]
        if len(binval) > bitsize:
            raise SteganographyException("binary value larger than the expected
size")
        while len(binval) < bitsize:
            binval = "0"+binval
        return binval

    def encode_text(self, txt):
        l = len(txt)
        binl = self.binary_value(l, 16)
#Length coded on 2 bytes so the text size can be up to 65536 bytes long
        self.put_binary_value(binl)
#Put text length coded on 4 bytes
        for char in txt:
#And put all the chars
            c = ord(char)
            self.put_binary_value(self.byteValue(c))
        return self.image

    def decode_text(self):
        ls = self.read_bits(16)
#Read the text size in bytes
```

```
        l = int(ls,2)
        i = 0
        unhideTxt = ""
        while i < l:
#Read all bytes of the text
            tmp = self.read_byte()
#So one byte
            i += 1
            unhideTxt += chr(int(tmp,2))
#Every chars concatenated to str
        return unhideTxt

    def encode_image(self, imtohide):
        w = imtohide.width
        h = imtohide.height
        if self.width*self.height*self.nbchannels < w*h*imtohide.channels:
            raise SteganographyException("Carrier image not big enough to hold
all the datas to steganography")
        binw = self.binary_value(w, 16)
#Width coded on to byte so width up to 65536
        binh = self.binary_value(h, 16)
        self.put_binary_value(binw)
#Put width
        self.put_binary_value(binh)
#Put height
        for h in range(imtohide.height):
#Iterate the hole image to put every pixel values
            for w in range(imtohide.width):
                for chan in range(imtohide.channels):
                    val = imtohide[h,w][chan]


                    self.put_binary_value(self.byteValue(int(val)))
        return self.image


    def decode_image(self):
        width = int(self.read_bits(16),2)
#Read 16bits and convert it in int
        height = int(self.read_bits(16),2)
        unhideimg = np.zeros((width,height, 3), np.uint8)
#Create an image in which we will put all the pixels read
        for h in range(height):

            for w in range(width):

                for chan in range(unhideimg.channels):

                    val = list(unhideimg[h,w])
                    val[chan] = int(self.read_byte(),2)
#Read the value
                    unhideimg[h,w] = tuple(val)
        return unhideimg

    def encode_binary(self, data):
        l = len(data)
        if self.width*self.height*self.nbchannels < l+64:
raise SteganographyException("Carrier image not big enough to hold all the datas
to steganography")

        self.put_binary_value(self.binary_value(l, 64))
        for byte in data:
            byte = byte if isinstance(byte, int) else ord(byte)
```

```
# Compat py2/py3
            self.put_binary_value(self.byteValue(byte))
        return self.image

    def decode_binary(self):
        l = int(self.read_bits(64), 2)
        output = b""
        for i in range(l):
            output += bytearray([int(self.read_byte(),2)])
        return output


def main():
    args = docopt.docopt(__doc__, version="0.2")
    in_f = args["--in"]
    out_f = args["--out"]
    in_img = cv2.imread(in_f)
    steg = LSBSteg(in_img)
    lossy_formats = ["jpeg", "jpg"]

    if args['encode']:

#Handling lossy format
        out_f, out_ext = out_f.split(".")
        if out_ext in lossy_formats:
            out_f = out_f + ".png"

            print("Output file changed to ", out_f)
        data = open(args["--file"], "rb").read()
        res = steg.encode_binary(data)

        cv2.imwrite(out_f, res)
    elif args["decode"]:
        raw = steg.decode_binary()
        with open(out_f, "wb") as f:
            f.write(raw)


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