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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Мороз А.С.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 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 *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

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

Морозу Антону Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії*

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 кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2023 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	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Мороз А.С.
(прізвище та ініціали)

АНОТАЦІЯ

Мороз А.С. Програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для приховування інформації на основі методів стеганографії.

Метою розробки є програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії.

Результат роботи – програмна реалізація системи кібербезпеки для приховування інформації на основі методів стеганографії.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: кібербезпека, стеганографія

ABSTRACT

Moroz A.S. Cybersecurity system software for hiding information based on steganography techniques. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system to hide information based on steganography methods.

The purpose of the development is the software of the cyber security system for hiding information based on steganography methods.

The result of the work is the software implementation of a cyber security system for hiding information based on steganography methods.

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 tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10.4.1 **environment.**

Keywords: cyber security, steganography

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВДТ	–	відео-дисплейні термінали
ЕОМ	–	електронно-обчислювальна машина
ЕПТ	–	електроннопроменева трубка
ЕЦП	–	електронний цифровий підпис
ЗІ	–	захист інформації
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. Дослідження проблем розробки, удосконалювання й застосування методів захисту інформації в процесі її зберігання й передачі привертає увагу безлічі дослідників, тому що розробка нових і вдосконалювання наявних методів захисту має велике значення для розвитку інфокомунікаційних систем.

У сучасних системах захисту інформації величезну роль грають не тільки методи криптографії, але й методи стеганографії. Якщо класичне завдання криптографії полягає в тому, щоб сховати від третіх осіб зміст повідомлення, то класичне завдання стеганографії – сховати сам факт передачі повідомлення. Зазначене завдання стеганографії вирішується за допомогою впровадження повідомлень у необразливі на вид об'єкти даних, називані контейнерами, передача яких є звичайною справою й не викликає підозр. Ключовим поняттям стеганографії є стегосистема, тобто сукупність засобів і методів, використовуваних для організації схованого каналу передачі даних. Існує й ряд інших актуальних завдань, які належать стеганографії, наприклад, захист авторських прав, що також базується на впровадженні в авторські цифрові документи схованих повідомлень, що ідентифікують автора або законних одержувачів.

Зворотне завдання стеганографії називається стегоаналізом. На відміну від криптоаналізу, основною метою якого є розкриття змісту повідомлення, стегоаналіз, у першу чергу, спрямований на розкриття факту наявності зв'язку, тобто на виявлення наявності схованих повідомлень. Стеганографія й стегоаналіз нерозривно зв'язані між собою. Їхні методи постійно конкурують один з одним і успіхи в одній області, як правило, приводять до появи нових результатів в іншій. Так, неможливо якісно вирішити завдання стегоаналізу, не розглядаючи новітніх методів впровадження схованої інформації. Хоча розглянуті завдання відомі

давно, на сучасному етапі проблемами стеганографії й стегоаналізу займаються багато українських, російських і закордонних учених. Перші дослідження в цій області включають роботи М. Куттера (Kutter, M.), Ф. Джордана (Jordan, F.), Ф. Боссена (Bossen, F.), Г. Лангелара (Langelaar, G.), Л. Марвела (Marvel, L.) і багатьох інших. Серед нині діючих учених великий внесок у розвиток стеганографії внесли роботи Р. Андерсона (Anderson, R.), К. Кашена (Cachin, C.), Н. Провоса (Provos N.), К. Саллівана (Sullivan, K.), Х. Фарида (Farid, H.), Дж. Фридрич (Fridrich, J,), А. Кера (Ker, A.) і інших дослідників. В останні роки значні успіхи були також досягнуті представниками школи теорії інформації, очолюваною проф. Б. Я. Рябко.

Якщо говорити про теоретично доведені співвідношення, які можуть бути досягнуті між стеганографією і стегоаналізом, то варто розглянути так звані зроблені стегосистеми, тобто стегосистеми, у яких наявність схованих повідомлень виявити в принципі неможливо. Концепція зробленої стегосистеми вперше була уведена в роботі К. Кашена (Cachin, C.). Потім, у спільних роботах Б. Я. Рябко, Д. Б. Рябко й А. Н. Фіонова були запропоновані ефективні конструкції побудови таких систем, що базуються на ідеях і методах теорії інформації. Особливість даних конструкцій у тому, що вони застосовні до імовірнісних джерел інформації й не завжди можуть прямо використовуватися в конкретних практичних ситуаціях.

Один з найбільше активно використовуваних і досліджуваних видів контейнерів – це цифрові зображення. Такі контейнери володіють рядом переваг, таких як заздалегідь відомий щодо великий розмір цифрового подання зображення, наявність у більшості зображень областей із шумовою структурою, а також слабка чутливість людського зору до незначних змін яскравості й контрасту зображення. Все це дозволяє вбудовувати в зображення досить великий обсяг схованих даних. У багатьох роботах розглядаються растрові зображення, що використовують методи, що не спотворюють, стиску (BMP, TIFF, PNG, PCX, TGA, PGM). Впровадження в такі зображення відбувається

безпосередньо в матрицю растрових даних. Крім того, більшість підходів, як до впровадження, так і до аналізу, з урахуванням деяких доробок виявляються застосовними до інших типів контейнерів, таких як JPEG, WAW, AVI і іншим.

Основними критеріями для оцінки й порівняння різних методів побудови стеганографічних систем є їхня стійкість і ємність. На відміну від досить досліджених криптографічних систем, оцінки стійкості стегосистем більш складне й саме поняття стійкості має велику кількість різних формулювань, що пояснюється розмаїтістю завдань стеганографічного захисту даних. У справжній роботі досліджуються методи побудови стегосистем, призначених для приховання факту передачі конфіденційних повідомлень. Говорячи про стійкість криптографічних систем, важливо згадати про принципі Керкхоффса, що полягає в тому, що система захисту інформації повинна забезпечувати свої функції навіть при повної інформованості супротивника про її структуру й алгоритми, і вся таємність системи повинна полягати в ключі. Цей принцип також можна співвіднести з визначенням стійкості стегосистем. У цьому випадку, ключем може бути, наприклад, секретна послідовність, що визначає порядок проходу елементів контейнеру при впровадженні біт інформації, що має місце в алгоритмах неуважного заповнення контейнерів. Другий критерій, ємність методу, визначає максимальна кількість інформації, що вбудовується, і може виражатися в одиницях біт на піксель (bpp).

Особливість пропонованої роботи полягає в з'єднанні теоретико-інформаційних ідей і методів побудови зроблених стегосистем і нових підходів до впровадження інформації в графічні файли. Це дозволяє одержати нові, більше стійкі, чим відомі раніше, методи впровадження й визначити границі можливостей сучасних методів стегоаналізу. Таким чином, робота спрямована на розробку математичних (алгоритмічних) принципів і рішень по створенню нових і вдосконалюванню існуючих засобів захисту інформації й забезпечення інформаційної безпеки.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії.

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

– Огляд існуючих систем для приховування інформації на основі методів стеганографії.

– Дослідження системи кібербезпеки для приховування інформації на основі методів стеганографії.

– Програмна реалізація системи кібербезпеки для приховування інформації на основі методів стеганографії.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для приховування інформації на основі методів стеганографії.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії, 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

Система призначена для приховання інформації на основі стеганографії. Стеганографія – набір засобів і методів приховання факту передачі повідомлення. Стеганографія приховує сам факт передачі повідомлення, а криптографія вважає, що повідомлення (у шифрованому виді) доступно незаконному користувачеві, але він не може витягти із цього повідомлення інформацію, яка захищена.

Всі алгоритми вбудовування схованої інформації можна розділити на кілька підгруп:

– Працюючі із самим цифровим сигналом. Наприклад, метод LSB.

– «Упаювання» схованої інформації. У цьому випадку відбувається накладення приховуваного зображення (звуку, іноді тексту) поверх оригіналу. Часто використовується для вбудовування ЦВЗ.

– Використання особливостей форматів файлів. Сюди можна віднести запис інформації в метадані або в різні інші не використовувані зарезервовані поля файлу.

По способу вбудовування інформації стегоалгоритми можна розділити на лінійні (адditивні), нелінійні й інші. Алгоритми адditивного впровадження інформації полягають у лінійній модифікації вихідного зображення, а її добування в декодері виробляється кореляційними методами. При цьому ЦВЗ звичайно складається із зображенням-контейнером, або «вплавляється» (fusion) у нього. У нелінійних методах вбудовування інформації використовується скалярне або векторне квантування. Серед інших методів певний інтерес представляють методи, що використовують ідеї фрактального кодування зображень. До адditивним алгоритмів можна віднести:

– A17 (Cox).

– A18 (Barni).

– L18D (Lange).

– A21 (J. Kim).

– A25 (C. Podilchuk).

Метод LSB

LSB (Least Significant Bit, найменший значущий біт) – суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображення, аудио або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Суть методу полягає в наступному: Допустимо, є 8-бітне зображення в градаціях сірого. 00h (00000000b) позначає чорний колір, FFh (11111111b) – білий. Усього є 256 градацій (2^8). Також припустимо, що повідомлення складається з 1 байта – наприклад, 01101011b. При використанні 2 молодших біт в описах пікселів, нам буде потрібно 4 пікселя. Допустимо, вони чорного кольору. Тоді пікселі, що містять сховане повідомлення, будуть виглядати в такий спосіб: 00000001 00000010 00000010 00000011. Тоді колір пікселів зміниться: першого – на 1/255, другого й третього – на 2/255 і четвертого – на 3/255. Такі градації, мало того що непомітні для людини, можуть взагалі не відобразитися при використанні низькоякісних пристроїв виводу.

Методи LSB є нестійкими до всіх видів атак і можуть бути використані тільки при відсутності шуму в каналі передачі даних. Виявлення LSB-кодованого стего здійснюється по аномальних характеристиках розподілу значень діапазону молодших бітів відліків цифрового сигналу. Всі методи LSB є, як правило, адитивними (A17, L18D). Інші методи приховання інформації в графічних файлах орієнтовані на формати файлів із втратою, приміром, JPEG. На відміну від LSB вони більше стійкі до геометричних перетворень. Це виходить за рахунок варіювання в широкому діапазоні якості зображення, що приводить до неможливості визначення джерела зображення.

Ехо-методи

Ехо-методи застосовуються в цифровий аудіостеганографії й використовують нерівномірні проміжки між ехо-сигналами для кодування послідовності значень. При накладенні ряду обмежень дотримується умова непомітності для людського сприйняття. Ехо характеризується трьома параметрами: початковою амплітудою, ступенем загасання, затримкою. При досягненні якогось порога між сигналом і луною вони змішуються. У цій крапці людське вухо не може вже відрізнити ці два сигнали. Наявність цієї крапки складно визначити, і вона залежить від якості вихідного запису, слухача. Найчастіше використовується затримка близько 1/1000, що цілком прийнятно для більшості записів і слухачів. Для позначення логічного нуля й одиниці використовується дві різних затримки. Вони обидві повинні бути менше, ніж поріг чутливості вуха слухача до одержуваного еха. Ехо-методи стійкі до амплітудних і частотних атак, але нестійкі до атак за часом.

Фазове кодування

Фазове кодування (phase coding, фазове кодування) – також застосовується в цифровий аудіостеганографії. Відбувається заміна вихідного звукового елемента на відносну фазу, що і є секретним повідомленням. Фаза елементів, що йдуть підряд, повинна бути додана таким чином, щоб зберегти відносну фазу між вихідними елементами. Фазове кодування є одним з найефективніших методів приховання інформації.

Метод розширеного спектра

Метод вбудовування повідомлення полягає в тім, що спеціальна випадкова послідовність вбудовується в контейнер, потім, використовуючи погоджений фільтр, дана послідовність детектується. Даний метод дозволяє вбудовувати велика кількість повідомлень у контейнер, і вони не будуть створювати перешкоди один одному. Метод запозичений із широкополосного зв'язку.

1.2 Область застосування

У цей час розвиваються методи комп'ютерної стеганографії – самостійного наукового напрямку інформаційної безпеки, що вивчає проблеми створення компонентів приховуваної інформації у відкритому інформаційному середовищі, що може бути сформована обчислювальними системами й мережами.

Сучасне застосування стеганографії:

– вбудовування інформації з метою сховати незаконну передачу;

– захист інформації від несанкціонованого доступу;

– вбудовування цифрових водяних знаків для захисту авторських прав (watermarking);

– вбудовування ідентифікаційних номерів (fingerprinting);

– вбудовування заголовків (captioning);

– камуфляж програмного забезпечення.

Цифрові водяні знаки (ЦВЗ)застосовуються для захисту від копіювання й несанкціонованого використання мультимедійної інформації й полягає у вбудовуванні в захищаємий об’єкт, невидимих міток – ЦВЗ. ЦВЗ можуть містити деякий автентичний код, інформацію про власника, або яку-небудь керуючу інформацію. Найбільш підходящими об'єктами захисту за допомогою ЦВЗ є нерухливі зображення, файли аудио й відеоданих.

Технологія вбудовування ідентифікаційних номерів виробників має багато загального з технологією ЦВЗ. Відмінність полягає в тім, що в першому випадку кожна захищена копія має свій унікально вбудовуваний номер, (звідси й назва – дослівно «відбитки пальців»). Цей ідентифікаційний номер дозволяє виробникові відслідковувати подальшу долю свого дітища: чи не зайнявся хто-небудь із покупців незаконним тиражуванням. Якщо так, то «відбитки пальців» швидко вкажуть на винного.

Вбудовування заголовків (невидиме) може застосовуватися, наприклад, для підпису медичних знімків, нанесення легенди на карту й т.і. Метою є зберігання різнорідно представленої інформації в єдиному цілому. Це, мабуть, єдиний додаток стеганографії, де в явному виді відсутній порушник.

Нерідко методи стеганографії використовують для камуфлювання програмного забезпечення. У тих випадках, коли використання програм незареєстрованними користувачами є небажаним, воно може бути закамуфльоване під стандартні універсальні програмні продукти (наприклад, текстові редактори) або приховано у файлах мультимедіа (наприклад, у звуковому супроводі комп'ютерних ігор).

Комп'ютерна стеганографія – сукупність стеганографічних методів, які реалізуються на основі комп'ютерної техніки й програмного забезпечення в рамках окремих обчислювальних або керуючих систем, корпоративних або глобальних обчислювальних мереж.

При використанні методів комп'ютерної стеганографії повинні враховуватися наступні умови:

– супротивник може мати повне подання про стеганографічну систему й деталі її реалізації. Єдиною інформацією, що повинна залишатися йому невідомою, – це ключ, за допомогою якого можна встановити факт присутності схованого повідомлення і його зміст;

– якщо супротивникові якимось образом удалося довідатися про факт існування схованого повідомлення, то це не повинне дозволити йому витягти подібні повідомлення з інших стеганограм доти, поки ключ зберігається в таємниці;

– потенційний супротивник повинен бути позбавлений яких-небудь технічних і інших переваг у розпізнаванні або розкритті змісту таємних повідомлень.

Нижче буде обговорено основні теоретичні положення комп'ютерної стеганографії й розглянуті деякі методи приховання даних в інформаційному середовищі, що може бути підтримана обчислювальними системами й мережами.

За аналогією із криптографічними системами, у стеганографії розрізняють системи із секретним ключем і системи з відкритим ключем.

У стеганографічній системі із секретним ключем використовується один ключ, що повинен бути заздалегідь відомий абонентам до початку схованого обміну секретними повідомленнями або пересланий по захищеному каналу.

У стегосистемі з відкритим ключем для вбудовування й добування таємного повідомлення використовуються різні ключі, причому вивести один ключ із іншого за допомогою обчислень неможливо. Один із ключів (відкритий) може передаватися вільно по незахищеному каналу зв'язку, а другий, секретний ключ, – по захищеному каналу. Дана схема добре працює при взаємній недовірі відправника й одержувача.

З огляду на все різноманіття стеганографічних систем, зведемо їх до наступних типів: безключових стегосистем, систем із секретним ключем, систем з відкритим ключем і змішаних стегосистем.

Стегосистеми із секретним ключем

Дотримуючись закону Керкхоффса, безпека системи повинна ґрунтуватися на деякій секретній інформації, без знання якої не можна витягти з контейнеру секретну інформацію. У стегосистемах така інформація називається стегоключем. Відправник, вбудовуючи секретне повідомлення в обраний контейнер C, використовує секретний стегоключ k. Якщо використовуваний у стеганографічном перетворенні ключ k відомий одержувачеві, то він зможе витягти сховане повідомлення з контейнеру. Без знання такого ключа будь-який інший користувач цього зробити не зможе.

Безключові стегосистем

Для функціонування безключових стегосистем не потрібно ніяких додаткових даних у вигляді стегоключа крім алгоритму стеганографічного

перетворення. Безпека безключових стегосистем заснована на таємності використовуваних стеганографічних перетворень. Це суперечить основному принципу Керкхоффса для систем захисту інформації. Найчастіше для підвищення безпеки безключових систем, перед початком процесу стеганографічного приховання попередньо виконується шифрування приховуваної інформації. Ясно, що такий підхід збільшує захищеність усього процесу зв'язку, оскільки це ускладнює виявлення схованого повідомлення. Однак, “сильні” стеганографічні системи, як правило, не мають потреби в попередньому шифруванні приховуваних повідомлень.

У сучасній стеганографії, у цілому, можна виділити в напрямки: технологічну стеганографію й інформаційну стеганографію.

До методів технологічної стеганографії ставляться методи, які засновані на використанні хімічних або фізичних властивостей різних матеріальних носіїв інформації.

Хімічні методи стеганографії зводиться майже винятково до застосування невидимого чорнила, до яких ставляться органічні рідини й симпатичні хімікалії.

До фізичних методів можна віднести мікрокрапки, різного виду схованки й методи камуфляжу. У цей час фізичні методи становлять інтерес в області дослідження різний носіїв інформації з метою запису на них даних, які б не виявлялися звичайними методами зчитування. Особливий інтерес є до стандартних носіїв інформації засобів обчислювальної, аудіо й відео техніки. Крім цього, з'явився цілий ряд нових технологій, які, базуючись на традиційної стеганографії, використовують останні досягнення мікроелектроніки (голограми, кінеграми).

До інформаційної стеганографії можна віднести методи лінгвістичної й комп'ютерної стеганографії.

Лінгвістичні методи стеганографії підрозділяються на дві основні категорії: умовний лист і семаграми.

Існують три види умовного листа: жаргонний код, пустишечний шифр і геометрична система.

Жаргонний код – зовні необразливе слово має зовсім інше реальне значення, а текст складається так, щоб виглядати як можна більш безневинно й правдоподібно.

Пустишечний шифр – у тексті мають значення лише деякі певні букви або слова.

Геометрична форма – при її застосуванні слова, що мають значення, розташовуються на сторінці в певних місцях або в крапках перетинання геометричної фігури заданого розміру.

Семаграми – таємні повідомлення, у яких шифропозаченням є будь-які символи, крім букв і цифр. Ці повідомлення можуть бути передані, наприклад, у рисунку, що містить крапки й тирі для читання по коду Морзе.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії, 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

Наведемо огляд існуючих програм стеганографічного захисту інформації.

DarkCrypt IV

Як контейнер зашифрованої інформації може використовуватися текстовий файл (CP-1251), BMP-24, TIFF-24 (LZW, DEFLATE), PNG-24, JPEG2000, PSD, TGA, MNG, WAVE Audio (16 bit, Stereo), Win32/PE файл (.exe, .dll), потоки NTFS (будь-який файл).

Плагін для Total Commader, що дозволяє шифрувати будь-які файли в TotalCommander одним з 100 наймогутніших алгоритмів шифрування використовуючи 5 різних режимів. Підтримує впакування безлічі файлів в один шифрований архів у режимі Tar.XDC упакування.

Плагін DarkCryptTC зручний, працює дуже швидко, безкоштовний для використання.

Плагін дозволяє робити зашифровку/розшифровку кожним з 40 найпоширеніших алгоритмів шифрування: ДЕРЖСТАНДАРТ 28147-89, Cast128, Cast256, Blowfish, IDEA, Mars, Misty 1, RC2, RC4, RC5, RC6, FROG, Rijndael, SAFER, SAFER-K40, SAFER-SK40, SAFER-K64, SAFER-SK64, SAFER-K128, SAFER-SK128, TEA, TEAN, Skipjack, SCOP, Q128, 3Way, Twofish, Shark, Square, Single DES, Double DES, Triple DES, Double DES16, Triple DES16, TripleDES24, DESX, NewDES, Diamond II, Diamond II Lite, SapphireII.

Додатково доступні алгоритми (фіксований розмір ключа на основі хешування): Ice, Thin Ice (64 біт), Ice 2 (128 біт), Serpent (256 біт), Gost (256 біт), AES (256 біт), Mars (512 біт), Blowfish (384 біт), PC1 (128 біт, хешований MD5

ключ), XXTEA (128 біт, хешований MD5 ключ, блокове шифрування ECB), XXTEA-x (128 біт, хешований RipeMD128 ключ, блокове шифрування CBC), Hurricane (128 біт, Roman Ganin, 2005, MD5 ключ), LOKI97 (256 біт, Haval), Camellia (256 біт, ECB, Haval), Camellia-X (256 біт, CBC, Haval), Iraqi (160 біт), Bass-O-Matic (512 бітний ключ, 2048-бітний блок, 16 таблиць, 8-16 раундів, CFB), Bass-O-Matic (512 бітний ключ, CFB), Bass-O-Matic (2048 бітний ключ, CFB), Khazad (128 біт, CBC), Noekeon Indirect (128 біт, CBC), Caracachs (128 біт, CBC), FealNx (128 біт, CBC), Lucifer (128 біт, CBC, Enchanced), Redoc3 (256 біт, CBC), BBC (96 біт, 256K блок), EnRUPT (512 біт, CBC, Skein хеш), EnRUPT-w (512 біт, CBC, Whirlpool), EnRUPT-md6 (512 біт, CBC, MD6), Mir (128 біт), Crypton (256 біт, CBC), Decorrelated Fast Cipher – DFC (256 біт, CBC), Hasty Pudding Cipher – HPC (256 біт, CBC), Magenta (256 біт, CBC), E2 (256 біт, CBC), Deal (256 біт), Frog (256 біт), RC 6-256 (256 біт, на основі заявки AES), MBC2 (128 біт), Modular Multiplication based Block cipher – MMB (128 bit, CBC), MMB2 (128 bit, CBC), Safer+256 (256 bit, CBC, згідно заявці AES), MDC (512 біт, CFB), Anubis – модифікація Rijndael (256 біт, CBC), Anubis-tweaked (256 біт, CBC), Mars 256 (256 біт, CBC), Misty128 (128 біт, CBC), NewDes120 (120 біт, CBC), Twofish-256 (256 біт, CBC), Skip32 (128 біт, CBC), Square128 (128 біт, CFB, верс.2.7), NSEA (256 біт, CFB), KolchCrypt III (512 біт, 512 бітний блок, CBC), Aria (256 біт, CBC), Borland (48 біт), Mercy (128 біт, CBC), Raiden (256 біт, CBC), MacGuffin (128 біт, CBC), VigerePlus TEAII (512 біт, CBC), REDOC II (160 біт, CBC), Khufu (512 біт, CBC), Khufu-w (512 біт, CBC, Whirlpool), Keeloq (64 біт), IDEA-NXT (256 біт, CBC), NUSH (256 біт, CBC, на основі коду GNUPG), Wicker98 (128 біт, CBC), RTEA (256 біт, CBC), SHA1Crypt (512 біт, CBC), MD5Crypt (512 біт, CBC), MD4Crypt (512 біт, CBC), Serpent (256 біт, CBC), Tnepres (256 біт, CBC), XTEA-tw (128 біт, CBC, посилена версія), XXTEA-tw (128 біт, CBC, посилена версія), XXTEA-tw (128 біт, CBC, посилена версія, 960 бітний блок), SHACAL (512 біт, CBC, SHA512), SHACAL (512 біт, CBC, MD6), DES-X (64x2 біт, CBC), RC 2-1024 (1024 біт, CBC, Skein хеш-функція), IDEA128 (128 біт, CBC, Skein хеш-функція),

Kasumi (128 біт, Skein хеш-функція), Rijndael-256 (256 біт блок, 256 біт ключ, CBC), Anubis320 (320 bit, CBC), Blowfish448 (448 біт, CBC), DES-EDE (168 біт, CBC), GOSTcb (256 біт, ЦБ SBOX, CBC), GOST3cb (768 біт, ЦБ SBOX, CBC, EDE3 дизайн), Threefish (1024 біт ключ, 128 біт tweak-блок, 1024 біт блок, 80 раундів), CAST-128 (128 біт), CAST-256 (256 біт), Rijndael-128 (256 біт ключ, CBC), KASUMI-b (128 біт, CBC), CIPHERUNICORN-E (128 біт, CBC), Diamond2 (2048 біт, 16 байт блок, 12 раундів, CBC), SEED128 (128 біт, CBC), TC18 (64 біт, CBC), SHARK (128 біт, CBC), SHARK-E (128 біт, CBC), Skipjack80 (80 біт, CBC), SPEED (256 біт ключ, 128 біт блок, 64 раунду), SAFER++ (256 біт ключ, 128 біт блок, CBC), CS-CIPHER (128 біт, CBC), SAFER-SK128 (128 біт, CBC), BJ256 (256 бітний блоковий шифр Боба Дженкінса, 512 біт ключ, CBC), R3DES (3DESEDE, 192 біт, CBC), Raiden32 (128 біт, 32 раунду, CBC), Raiden256 (256 біт, 16 раундів, CBC), Raiden2 (256 біт, 24 раунду, CBC), XTEA-3 (256 біт, 128 біт блок, CBC), MULTISWAP (384 біт, CBC), PES (128 біт, CBC), TWOPES (256 біт, CBC, 16 раундів), Raiden-512 (512 біт, CBC, 32 раунду), NEWTEA (128 біт, 128 біт блок, CBC), AES-G (256 біт, CBC, на основі ASM-Коду Gladman), Rainbow (256 біт, CBC), NOEKEON (128 біт, CBC), RC 5-32/16/64 (512 біт, CBC), RC 6-512 (512 біт, CBC), Threefish-512 (512 біт ключ, 512 біт блок), Cartman-2P (384 біт, CBC), Chaos (512 біт, CBC), LOKI91 (512 біт, CBC), XTEA1 (128 біт, CBC), Q128e (128 біт, CBC), Newdes96 (128 біт, CBC), Mars-1248 (1248 біт, CBC), FNAM2 (512 біт, CBC), C2 (64 біт, CBC), Sinople (128 біт, CBC), Phantom (256 біт, CBC), Paranoia (512 біт, CBC), Pikachu (128 біт, CBC), SC6B (320 біт, CBC), Letsief (512 біт, CBC), Lja1 (2048 біт ключ, 128 біт блок, 16 циклів), Clefia (256 біт, CBC), MPJ2 (128 біт, CBC), Curupira2 (192 біт, CBC), Curupira1 (192 біт, CBC), KARLA (160 біт, CBC), Cobra-64-256 (256 біт ключ, 64 біт блок), Cobra128 (576 біт), Simplicity (256 біт), 3NEWDE (192 біт), CRYPTON 1.0 (256 біт), EksLOKI89 (256 біт), BREAKME (256 біт), VSEN (256 біт), Hierocrypt-3 (256 біт). Hierocrypt-L1 (128 біт), CIPHERUNICORN-A (256 біт), SC2000 (256 біт).

Так само доступні наступні симетричні криптоалгоритми:

– SALSA 20 (256 біт).

– Phelix (256 біт).

– ABC 3 (128 біт).

– SEED (128 біт).

– SHACAL-2 (512 біт).

– XTEA (128 біт).

Додатково доступна безліч потокових шифрів: Rabbit (128 біт), HC-256 (256 біт), Sosemanuk (256 біт), CryptMT3 (512 біт), Dragon (256 біт), Lex2 (128 біт), NLS2 (128 біт), Yamb (256 біт), Hermes (128 біт),FCSR (128 біт), Pomaranch (128 біт), Mickey (128 біт), Vest 32-Pro (256 біт), WG2 (128 біт), ZKCrypt3 (160 біт), Dicing (256 біт), Py6 (256 біт), Grain (128 біт), Achterbahn (128 біт), Moustique (96 біт), TPypy (512 біт), TPy6 (512 біт), Pypy (512 біт), TPy (512 біт), Fubuki (512 біт), SSS (128 біт), Pike (512 біт, CBC), Seal (128 біт, CBC), Trivium (80 біт), Decim (128 біт), Edon80 (80 біт), Sfinks (80 біт), Konton (512 біт, CBC), QCypher (512 біт, CBC), SCOP-384 (384 біт, CBC), Sober-128, QUALCOMM Incorporated (128 біт), Shannon (256 біт), Leviathan (128 біт, CBC), A5 (64 біт, CBC), Panama (256 біт), WAKE (256 біт), RC 4-drop[65536] (1024 біт), CS 2-128 (128 біт), Lili (128 біт), Lili2 (128 біт), Snow2 (256 bit), SEAL2 (160 біт), SN3 (6144 біт), PolarBear (128 біт), VMPC (512 біт), VMPC-KSA3 (512 біт), Chacha (256 біт), Turing (256 біт), Turing (384 біт).

Підтримка асиметричного шифрування RSA/Elgamal/ECC з використанням пари публічний – секретний ключ.

– Підтримується шифровка одного файлу в XDC і групи файлів/каталогів в Tar.XDC.

– Можливість зберігання ключа шифрування в текстовому файлі.

– Є убудований генератор ключових текстових файлів довільної довжини ключа.

– Визначення шифрованого файлу по вмісту.

– Зберігання інформації про метод шифрування й режим у заголовку файлу.

– Можливість додавання коментарю в архів.

– Просте й зручне настроювання параметрів шифрування для кожного файлу.

– Можливість роботи з фіксованим ключем шифрування й дешифрування. У такому режимі уведення пароля для шифрування й (або) дешифрування не потрібно (за бажанням).

– Можливість опціональної вичищення тимчасових файлів і вихідного файлу алгоритмом за стандартом DOD 5220.22-M або методом Гутманна.

– Опціональна можливість потужної BWT компресії алгоритмом ABC.

– Можливість підвищеної безпеки шляхом відключення підрахунку контрольної суми вихідного файлу.

Стандартний компонент Total Commander Ultima Prime (TCUP).

ImageSpyer 2009

Утиліта для приховання будь-яких файлів у звичайній картинці без її зміни й перекручування. ImageSpyer – це вітчизняна стеганографічна утиліта з найбільш ефективною реалізацією стеганографічних і криптографічних алгоритмів. У програмі використовується авторська реалізація алгоритму LSB, у результаті чого підсумкове зображення поміщає обсяг інформації, дорівнює числу пікселів вихідного зображення. Крім цього, ImageSpyer захищає впроваджений файл одним з 40 стійких криптоалгоритмів. У настроюваннях можна вибрати алгоритм і режим шифрування. Для утруднення розпізнавання даних є можливість задавати довільний набір порядку біт, таким чином, не знаючи даного набору, не вийде коректно вважати сховану інформацію.

Рисунок 2.2 – Інтерфейс користувача ImageSpyer

OpenPuff

OpenPuff Steganography and Watermarking (скорочено OpenPuff або Puff) – є безкоштовним стеганографічним інструментом для Microsoft Windows, що створений і підтримується дотепер Cosimo Oliboni – незалежним розроблювачем ПЗ.

Сховані дані розподілялися між «ланцюжком» файлів, що дозволяло:

– не перевантажувати контейнери зайвим вбудовуванням;

– застосовувати чотири рівні заплутування (криптографія, скремблювання, декореляція й накриття);

– розширювати двозначну криптографію до подвійний стеганографії.

Остання актуальна версія підтримує широкий спектр форматів носіїв:

– Зображення: Bmp, Jpeg, Png, Tga, Pcx.

– Аудіо: Aiff, Mp3, Next, Wav.

– Відео: 3gp, Mp4, Mpeg I, MPEG II, Vob.

– Adobe: Flv, Pdf, Swf.

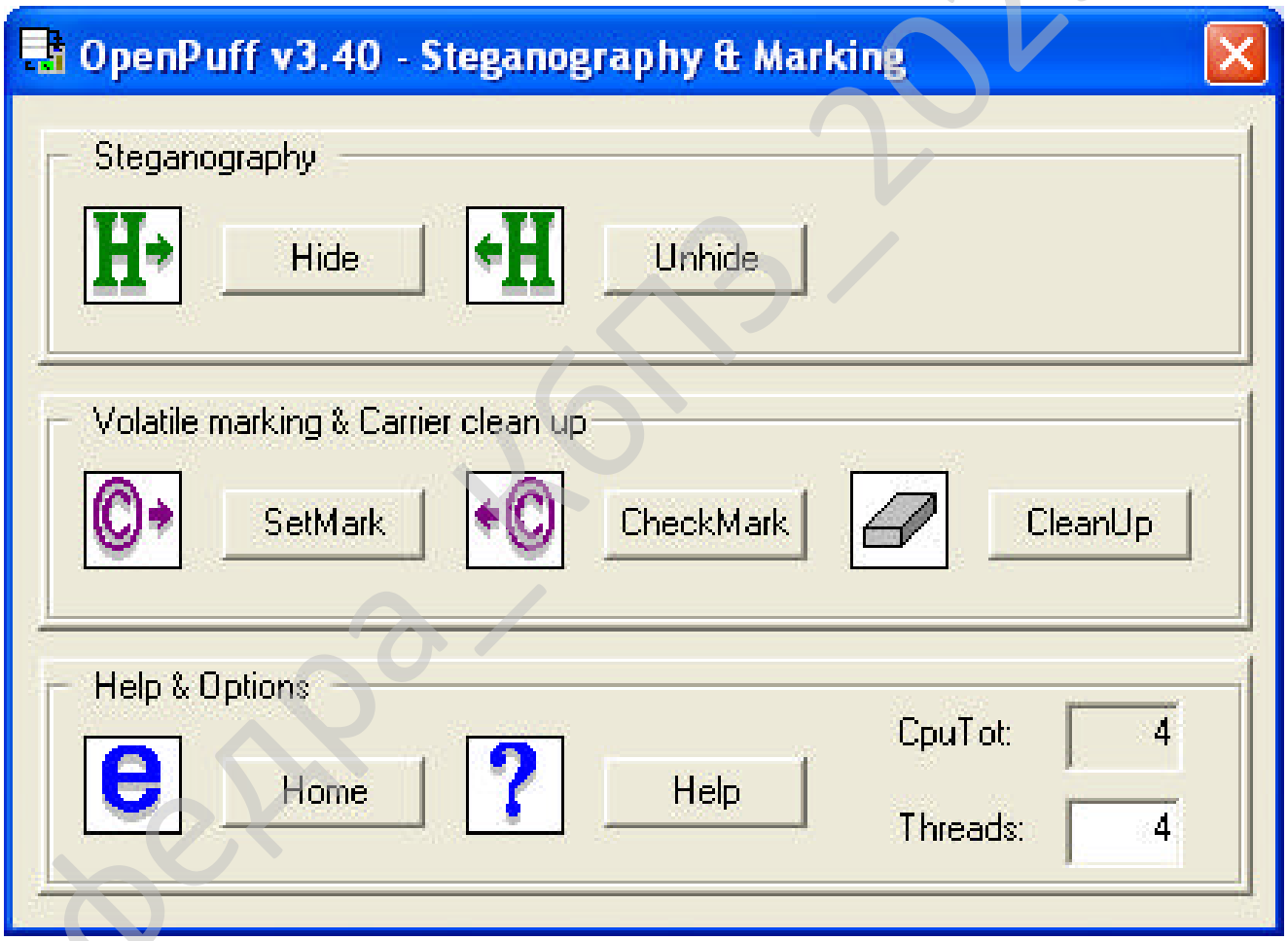

Рисунок 2.1 – Інтерфейс користувача OpenPuff

OpenPuff, в основному, використовується для:

– анонімного асинхронного обміну даними. Відправник приховує дані усередині загальнодоступних файл-контейнерів (секретним ключем є пароль + контейнери даних + порядок використання контейнерів) одержувач витягає повідомлення знаючи секретний ключ;

– вставки цифрових водяних знаків. Програма дозволяє додати знак копирайту для захисту інтелектуальної власності. Додавання виробляється за допомогою стеганографічних алгоритмів. Знаки копирайта невидимі але доступні (за допомогою даної програми), тому що вони не захищені паролем.

Незважаючи на використання стеганографії завжди є ймовірність виявлення факту приховання. Наприклад, якщо у вас знайшли програму OpenPuff і силою змушують відкрити пароль до прихованих даних. Для такого випадку передбачена можливість надати стегоаналітику помилкове приховання. Подвійна стеганографія дозволяє переконливо заперечувати факт приховання дійсно важливих даних. У контейнер паралельно із приховуваною важливою інформацією вкладаються неважливі дані для яких правдоподібно бажати збереження їхньої конфіденційності. Якщо буде потреба помилкове приховання відкривається замість сьогодення.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних

багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++

використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 кібербезпеки для приховування інформації на основі методів стеганографії.

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 кібербезпеки 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 кібербезпеки 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

Перш ніж описувати стегосистему введемо ряд понять. В 1996 році на конференції Information Hiding: First Information Workshop була прийнята єдина термінологія:

– Стеганографічна система (стегосистема) – об'єднання методів і засобів використовуваних для створення схованого каналу для передачі інформації. При побудові такої системи вмовилися про те, що:

1. Ворог представляє роботу стеганографічної системи. Невідомим для супротивника є ключ за допомогою якого можна довідатися про факт існування й змісту таємного повідомлення.

2. При виявленні супротивником наявності схованого повідомлення він не повинен змогти витягти повідомлення доти поки він не буде володіти ключем.

3. Супротивник не має технічних і інших переваг.

– Повідомлення – це термін, використовуваний для загальної назви переданої схованої інформації, будь те аркуш із написами молоком, голова раба або цифровий файл.

– Контейнер – так називається будь-яка інформація, використовувана для приховання таємного повідомлення. Порожній контейнер – контейнер, що не містить секретного послання. Заповнений контейнер (стегоконтейнер) – контейнер, що містить секретне послання.

– Стеганографічний канал (стегоканал) – канал передачі стегоконтейнеру.

– Ключ (стегоключ) – секретний ключ, потрібний для приховання стегоконтейнеру. Ключі в стегосистемах бувають двох типів: секретні й відкриті. Якщо стегосистема використовує секретний ключ, то він повинен бути створений або до початку обміну повідомленнями, або переданий по захищеному каналу.

Стегосистема, що використовує відкритий ключ, повинна бути влаштована таким чином, щоб було неможливо одержати з нього закритий ключ. У цьому випадку відкритий ключ ми можемо передавати по незахищеному каналу.

Важливим напрямком у розвитку сучасних засобів захисту інформації є стегенографічні системи, які забезпечують приховання в таємниці від супротивника не тільки інформаційного змісту переданих даних, але й самого факту передачі повідомлень [1, 2]. Найбільш перспективними є стеганографічні методи, побудова яких базується на розвиненому математичному апарату теорії дискретних сигналів і перешкодозахищеної передачі даних [3 – 7].

Пряме розширення спектра в теорії зв'язку

Для побудови сучасних перешкодозахищених систем цифрового зв'язку використовуються методи теорії дискретних сигналів, кореляційного й спектрального аналізу [3 – 7]. З погляду ефективного використання частотно-часових і енергетичних ресурсів каналів зв'язки найбільш перспективними вважаються широкополосні системи із шумоподібними дискретними сигналами й прямим розширенням спектра [3, 4].

Під дискретним сигналом будемо розуміти інформаційний сигнал, що представляється у вигляді окремих значень узятих за часом. Далі ми будемо розглядати дискретний сигнал як двійкову псевдовипадкову послідовність (ПВП) $\Phi_i = (\varphi_{i_0}, \varphi_{i_1}, ..., \varphi_{i_{n-1}})$ довжини n з безлічі $\Phi = \{\Phi_0, \Phi_1, ..., \Phi_{M-1}\}$ потужності $|\Phi| = M$ [6].

Елементи двійкової ПВП приймають одне зі значень:

$$\varphi_{i_z} = \begin{cases} +1 \\ -1 \end{cases}, \; z = 0, ..., n-1 \qquad (3.1)$$

Для побудови перешкодозахищеної широкополосної системи зв’язку використовують поняття кореляції дискретних сигналів – статистичного взаємозв'язку двох або декількох ПВП. Математичною мірою корельованості

(подібності) двох дискретних сигналів $\Phi_i, \Phi_j \in \Phi$ служить коефіцієнт кореляції $\rho(\Phi_i, \Phi_j)$ [3, 4]:

$$\rho\left(\Phi_i, \Phi_j\right) = \frac{1}{n} \sum_{z=0}^{n-1} \Phi_{iz} \Phi_{jz} \tag{3.2}$$

Два сигнали Φ_i, Φ_j називають ортогональними, якщо коефіцієнт кореляції $\rho(\Phi_i, \Phi_j) = 0$. Якщо $\rho(\Phi_i, \Phi_j) \approx 0$ будемо називати сигнали Φ_i й Φ_j квазіортогональними [5, 7].

У роботах [3 – 5, 7] досліджені різні підходи до побудови дискретних сигналів з поліпшеними ансамблевими й кореляційними властивостями: похідні ортогональні системи сигналів (ПОСС); нелінійні похідні кодові послідовності (НПКП); повні кодові кільця (ПКК); послідовності Голда. У таблиці 3.1 як приклад наведені результати досліджень ансамблевих і кореляційних властивостей похідних систем сигналів [7]. Як випливає з наведених у таблиці 3.1 даних, застосування похідних ортогональних дискретних сигналів дозволяє при збереженні низкою корельованості дискретних послідовностей ($\rho(\Phi_i, \Phi_j) \approx 0$) істотно підвищити потужність M ансамблів дискретних сигналів, з ростом довжини послідовностей ця тенденція підсилюється.

Таблиця 3.1 – Ансамблеві й кореляційні властивості дискретних сигналів

n	M	ρ
64	$\approx 10^3$	$2.1/\sqrt{n}$
128	$\approx 10^4$	$2.5/\sqrt{n}$
256	$\approx 10^6$	$2.9/\sqrt{n}$
512	$\approx 10^7$	$3.2/\sqrt{n}$
1024	$\approx 10^8$	$3.5/\sqrt{n}$
2048	$\approx 10^9$	$3.8/\sqrt{n}$
4096	$\approx 10^{10}$	$3.9/\sqrt{n}$

У сучасній теорії цифрового зв'язку великі ансамблі слабокорельованих дискретних сигналів використовуються для побудови широкополосних перешкодозахищених систем передачі даних. Передані повідомлення в таких каналух здобувають вид шумоподібних послідовностей, а за рахунок великої потужності ансамблів дискретних сигналів і прямого розширення частотного спектра забезпечується висока імітостійкість, перешкодозахищеність і скритність цифрових каналів зв'язку [3 – 5].

Для передачі даних у широкополосної системі зв'язку інформаційний сигнал $x(t)=\begin{cases}+1\\-1\end{cases}$ модулюється за допомогою його множення на розширювальний кодовий сигнал $g(t)=\Phi_i\in\Phi$ – псевдовипадкову послідовність із розглянутих вище ансамблів дискретних сигналів. Оскільки кодовий сигнал по своїх статистичних властивостях подібний до шуму, то отриманий розширений сигнал:

$$y'(t)=y(t)+e(t) \tag{3.3}$$

слабко відрізнимий від шумів у каналі зв'язку, що й дозволяє здійснити сховану передачу.

При прийманні в демодуляторі отриманий сигнал $y'(t)=y(t)+e(t)$ як суміш переданої послідовності $y(t)$ й помилок, що $e(t)$ відбулися в каналі зв'язку, множиться на $g(t)$синхронізовану копію розширювального сигналу . Інакше кажучи, на прийомній стороні здійснюється обчислення коефіцієнта кореляції (3.2), значення якого визначає правило ухвалення рішення:

$$\rho\left(y'(t),g(t)\right)=\frac{1}{n}\sum_{z=0}^{n-1}x(t)\Phi_{i_z}\Phi_{i_z}+\frac{1}{n}\sum_{z=0}^{n-1}e(t)\Phi_{i_z}$$

З огляду на псевдовипадковість Φ_i, використовуваних у якості $g(t)$, другим доданком у правій частині рівності можна зневажити (кількість «+1» приблизно дорівнює кількості «-1»), тобто:

$$\rho\left(y'(t), g(t)\right) \approx \rho(y(t), g(t)) = x(t)\frac{1}{n}\sum_{z=0}^{n-1}\left(\Phi_{i_z}\right)^2 = x(t), \qquad (3.4)$$

тобто значення інформаційного сигналу на прийомній стороні визначається по вираженню:

$$x(t) = \begin{cases} +1, \text{при } \rho\left(y'(t), g(t)\right) \approx +1; \\ -1, \text{при } \rho\left(y'(t), g(t)\right) \approx -1; \end{cases} \qquad (3.5)$$

де знак «≈» припускає наявність помилок $e(t)$, викликаних природними або навмисними перешкодами в каналі зв'язку.

Припустимо, що часова тривалість немодульованого сигналу $x(t)$ дорівнює T, а його частота відповідно дорівнює $F(x(t)) = \frac{1}{T}$. Передача модульованого сигналу $y(t)$ при тій же часовій тривалості T приведе до розширення частотного спектра переданого сигналу, пропорційно числу елементів псевдовипадкової послідовності, тобто пропорційно довжині n: $F(y(t)) = n\frac{1}{T} = nF(x(t))$ Проте, використання прямого розширення спектра переданого сигналу забезпечує одночасну передачу багатьох інших інформаційних сигналів у тій же смузі частот. Це треба із взаємної ортогональності (квазіортогональності) застосовуваних ансамблів дискретних сигналів. Дійсно, якщо на прийомній стороні прийнята адитивна суміш $\sum_l y_l(t)$ декількох модульованих сигналів, тоді обчислення коефіцієнта кореляції дасть наступне:

$$\rho\left(\sum_l y_l(t), g(t)\right) = \frac{1}{n}\sum_l \sum_{z=0}^{n-1} x_l(t)\Phi_{l_z}\Phi_{i_z} . \qquad (3.6)$$

Але всі послідовності з безлічі Ф мають низьке значення взаємної кореляції, тобто при $l \neq i$ маємо $\rho(\Phi_l, \Phi_i) = 0$ (для ортогональних сигналів маємо рівність $\rho(\Phi_l, \Phi_i) = 0$. Отже, всіма доданками при $l \neq i$ в правій частині рівності (3.6) можна зневажити. Звідси, при наявності в адитивної суміші

$\sum_l y_l(t)$ дискретного сигналу $\Phi_{l=i}$ маємо вираження (3.4) і відповідне правило ухвалення рішення (3.5).

Метод прямого розширення спектра знайшов практичне використання в системах цифрового зв'язку з кодовим поділом каналів (CDMA), де для кожного абонента інформаційного обміну використовуються унікальні розширювальні кодові сигнали з ансамблю ортогональних (квазіортогональних) послідовностей. Тобто для розрізнення кодових сигналів і поділу відповідних абонентських каналів використовувані ПВП повинні бути слабко корельовані один з одним, в ідеальному випадку – ортогональними.

Так, наприклад, у стандарті CDMA IS-95 для кодового поділу каналів використовуються ортогональні дискретні сигнали Уолша-Адамара [4]. Вони утворяться з рядків матриці Адамара H_i, формованої по рекуррентному правилу:

$$H_i = \begin{bmatrix} H_{i-1} & H_{i-1} \\ H_{i-1} & -H_{i-1} \end{bmatrix}, \; H_0 = [1]. \qquad (3.7)$$

Багаторазове повторення правила (3.7) дозволяє сформувати матрицю Адамара будь-якого розміру, кратного чотирьом. Рядка сформованих матриць взаємоортогональні, тобто їхній скалярний добуток дорівнює нулю. Ці рядки й становлять ансамбль $\Phi = \{\Phi_0, \Phi_1, ..., \Phi_{M-1}\}$ дискретних сигналів Уолша-Адамара $\Phi_i = (\varphi_{i_0}, \varphi_{i_1}, ..., \varphi_{i_{n-1}})$, де n – розмірність сформованої матриці H_i (в IS-95 використані H_i з $n = 64$).

Для передачі інформації один з рядків $\Phi_i \in \Phi$ матриці Адамара ставиться у відповідність абонентському каналу, наприклад, для зв'язку між базовою станцією й конкретним абонентом. Модуляція здійснюється за правилом (3.3), тобто для передачі інформаційної "1" посилає рядок Φ_i, для "0" – посилає послідовність, сформована шляхом логічного заперечення Φ_i (її інверсна копія).

Для виділення сигналу на прийомній стороні використовується кореляційний приймач, тобто обчислюється коефіцієнт кореляції (3.6). При точному збігу початку послідовності, що прийшла, і наявній Φ_i копії

спостерігаються піки кореляційної функції позитивної й негативної полярностей – залежно від переданого біта. Тобто, детектування сигналу відбувається в такий спосіб:

$$x(t)=\begin{cases} \text{"1", при } polarity>0; \\ \text{"0", при } polarity<0; \\ \text{стороній сигнал, при } polarity=0 \end{cases} \qquad (3.8)$$

де $polarity$ – полярність піка кореляційної функції.

Таким чином, застосування ортогональних систем дискретних сигналів Уолша-Адамара дозволяє забезпечити високоефективну широкополосний цифровий зв'язок. У той же час число утворених абонентських каналів зв'язку не може перевищувати потужності M ансамблю сигналів, у цьому випадку воно не перевищує розмірності матриці H_i, $M=n$. Інакше кажучи, максимальне число можливих ортогональних кодів обмежено їхньою довжиною. Для розглянутого приклада маємо $M=64$, (по специфікації IS-95 утворяться 61 абонентський і 3 службових канали). У цьому змісті квазіортогональні дискретні сигнали (с) $M>n$ мають незаперечну перевагу (див. таблицю 3.1), їхнє застосування потенційно дозволить істотно підвищити абонентську потужність системи зв'язку. Крім того, для розглянутих сигналів функція взаємної кореляції дорівнює нулю лише при відсутності часового зрушення між послідовностями. Як наслідок такі сигнали використовуються лише в синхронних системах і переважно в прямих каналух (від базової станції до абонента).

Таким чином, перспективним напрямком у розвитку сучасних систем широкополосної системи зв'язку із прямим розширенням спектра є розробка й дослідження методів синтезу більших ансамблів квазіортогональних дискретних сигналів з поліпшеними ансамблевими, структурними й кореляційними властивостями.

Розглянутий підхід до організації цифрових перешкодозахищених каналів зв'язку знайшов застосування при побудові стеганографічних методів захисту інформації. Так, у роботі [2] розширення спектра прямою послідовністю використано для створення стеганографічного методу вбудовування даних у

нерухливі зображення. Розглянемо один з варіантів реалізації цього методу, авторами якого є Смит і Комиски [2], проведемо дослідження його ефективності з погляду забезпечуваної пропускної здатності стеганографічного каналу зв'язку й досягається стійкості, що, до несанкціонованого добування інформаційних повідомлень.

Пряме розширення спектра в стеганографії

У методі Смита-Комиски [2], як і в розглянуті вище системах зв'язку із прямим розширенням спектра, інформаційне повідомлення побітно модулюється шляхом множення на ансамбль ортогональних сигналів. Потім промодульоване повідомлення вбудовується в контейнер – нерухливе зображення.

Уведемо деякі умовні позначки й математичні співвідношення, які, за аналогією з розглянутими вище системами широкополосного цифрового зв'язку дозволять досліджувати особливості побудови й інформаційного обміну даних у стеганосистемі.

Представимо інформаційне повідомлення m, підлягає вбудовуванню в цифровий контейнер-зображення, у вигляді блоків m_i рівної довжини, тобто $m = (m_0, m_1, ..., m_{N-1})$, де кожний блок m_i – послідовність (вектор) з n біт: $m_i = (m_{i_0}, m_{i_1}, ..., m_{i_{n-1}})$.

Контейнер-зображення будемо розглядати як масив даних C розмірністю $K \cdot L$, розбитий на підблоки розміром $k \cdot l = n$. Як елементи масиву c можуть виступати, наприклад, растрові дані використовуваного зображення.

Секретними ключовими даними є набір базисних функцій $Key = \Phi = \{\Phi_0, \Phi_1, ..., \Phi_{M-1}\}$, де всі базисні функції $\Phi_i = (\varphi_{i_0}, \varphi_{i_1}, ..., \varphi_{i_{n-1}})$ – взаємно ортогональні дискретні сигнали з довжиною, рівної розміру n блоку повідомлення m_i, тобто для будь-яких $i, j \in [0, ..., M-1]$ виконується рівність:

$$\rho(\Phi_i, \Phi_j) = \frac{1}{n} \sum_{z=0}^{n-1} \Phi_{i_z} \Phi_{j_z} = \begin{cases} +1, \text{при i = j;} \\ -1, \text{при } i \neq \text{j.} \end{cases}$$

Формальне графічне подання інформаційного повідомлення, контейнера-зображення й ключових даних наведено на рисунку 3.1.

$$m = \boxed{m_0 \mid m_1 \mid \dots \mid m_i \mid \dots \mid m_{N-1}}$$

$$\forall i: m_i = \boxed{m_{i0} \mid m_{i1} \mid \dots \mid m_{in-1}}$$

$$N = K \cdot L / n$$

$$Key = \begin{array}{|c|c|c|c|c|c|}\hline \varphi_{00} & \varphi_{01} & \dots & \varphi_{0z} & \dots & \varphi_{0n-1} \\ \hline \varphi_{10} & \varphi_{11} & \dots & \varphi_{1z} & \dots & \varphi_{1n-1} \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \varphi_{i0} & \varphi_{i1} & \dots & \varphi_{iz} & \dots & \varphi_{in-1} \\ \hline \dots & \dots & \dots & \dots & \dots & \dots \\ \hline \varphi_{n0} & \varphi_{n1} & \dots & \varphi_{nz} & \dots & \varphi_{nn-1} \\ \hline \end{array}$$

$$C = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline c_{0,0} & c_{0,1} & \dots & c_{0,k-1} & c_{0,k} & c_{0,k+1} & \dots & c_{0,2k-1} & \dots & c_{0,K-1} \\ \hline c_{1,0} & c_{1,1} & \dots & c_{1,k-1} & c_{1,k} & c_{1,k+1} & \dots & c_{1,2k-1} & \dots & c_{1,K-1} \\ \hline \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \hline c_{l-1,0} & c_{l-1,1} & \dots & c_{l-1,k-1} & c_{l-1,k} & c_{l-1,k+1} & \dots & c_{l-1,2k-1} & \dots & c_{l-1,K-1} \\ \hline c_{l,0} & c_{l,1} & \dots & c_{l,K-1} & c_{l,K} & c_{l,K+1} & \dots & c_{l,2k-1} & \dots & c_{l,K-1} \\ \hline c_{l,0} & c_{l,1} & \dots & c_{l,K-1} & c_{l,K} & c_{l,K+1} & \dots & c_{l+1,2k-1} & \dots & c_{l+1,K-1} \\ \hline \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \hline c_{2l-1,0} & c_{2l-1,1} & \dots & c_{2l-1,k-1} & c_{2l-1,k} & c_{2l-1,k+1} & \dots & c_{2l-1,2k-1} & \dots & c_{2l-1,K-1} \\ \hline \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \hline c_{L-1,0} & c_{L-1,1} & \dots & c_{L-1,k-1} & c_{L-1,k} & c_{L-1,k+1} & \dots & c_{L-1,2k-1} & \dots & c_{L-1,K-1} \\ \hline \end{array}$$

Рисунок 3.1 – Формальне подання інформаційного повідомлення, контейнера-зображення й ключових даних

3.2 Розробка структурної схеми

Метою стеганографічного перетворення інформації є вбудовування кожного окремого блоку повідомлення m_i у відповідний блок контейнера-зображення. У блок даних цифрового зображення розмірністю $K \cdot L$ елементів може бути убудовано $K \cdot \frac{L}{n}$ блоків інформаційного повідомлення, тобто до $K \cdot L$ біт.

Розбивка контейнера на блоки може бути довільною, однак, як показує практика, найбільш доцільним (менший, на відміну від одномірного подання, чисельний розкид значень у блоці) є двовимірна розбивка, наведена на рисунку 3.1. У якості ключових даних (масиву базисних функцій $Key = \Phi$) будемо використовувати розглянуті вище ансамблі ортогональних дискретних сигналів Уолша-Адамара.

Вбудовування інформаційного повідомлення здійснюється в такий спосіб. Кожний блок повідомлення $m_{i_j}, j = 0,..., n-1$ зіставляється з окремим блоком контейнера-зображення. Кожний інформаційний біт блоку $m_{i_j}, j = 0,..., n-1$ представляється у вигляді інформаційного сигналу $m_{i_j}(t) = \begin{cases} +1, m_{i_j} = 1; \\ -1, m_{i_j} = 0; \end{cases}$ й за аналогією з (3.3) модулюється розширювальним кодовим сигналом (3.базисними функціями), тобто ПВП $\Phi_j \in \Phi$.

У результаті, для кожного інформаційного блоку m_i формується модульований інформаційний сигнал

$$E_i(t) = \sum_{j=0}^{n-1} \sum_{z=0}^{n-1} m_{i_j}(t) \Phi_{j_z} . \qquad (3.9)$$

Отриманий блок повідомлення E_i попіксельно підсумується з підблоком контейнера.

Позначимо блоки контейнера в такий спосіб (див. рис. 3.1):

$$C_0 = \begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,k-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,k-1} \\ \cdots & \cdots & \cdots & \cdots \\ c_{l-1,0} & c_{l-1,1} & \cdots & c_{l-1,k-1} \end{pmatrix},$$

$$C_1 = \begin{pmatrix} c_{0,k} & c_{0,k+1} & \cdots & c_{0,2k-1} \\ c_{1,k} & c_{1,k+1} & \cdots & c_{1,2k-1} \\ \cdots & \cdots & \cdots & \cdots \\ c_{l-1,k} & c_{l-1,k+1} & \cdots & c_{l-1,2k-1} \end{pmatrix}$$

, ...,

$$C_{N-1} = \begin{pmatrix} c_{L-l-1,K-k-1} & c_{L-l-1,K-k} & \cdots & c_{L-l-1,K-1} \\ c_{L-l,K-k-1} & c_{L-l,K-k} & \cdots & c_{L-l,K-1} \\ \cdots & \cdots & \cdots & \cdots \\ c_{L-1,K-k-1} & c_{l-1,k+1} & \cdots & c_{L-1,K-1} \end{pmatrix}.$$

Відповідні модульовані інформаційні сигнали $E_i(t)$ представимо у вигляді двовимірного масиву даних:

$$E_i = \begin{pmatrix} E_{i_0} & E_{i_1} & \cdots & E_{i_{k-1}} \\ E_{i_k} & E_{i_{k+1}} & \cdots & E_{i_{2k-1}} \\ \cdots & \cdots & \cdots & \cdots \\ E_{i_{(l-1)(k-1)-k+1=n-k+1}} & E_{i_{(l-1)(k-1)-k+2=n-k+2}} & \cdots & E_{i_{(l-1)(k-1)=n-1}} \end{pmatrix}, i = 0,..., N-1$$

Тоді стеганограмма (заповнений контейнер) формується за допомогою об'єднання масивів даних S_i, $i = 0,..., N-1$:

$$S_i = C_i + E_i \cdot G, \tag{3.10}$$

де $G > 0$ – коефіцієнт підсилення розширювального сигналу, що задає «енергію» вбудовуються біт інформаційної послідовності.

Таким чином, заповнений контейнер S утвориться зі сформованих блоків s_i, $i = 0,..., N-1$ за допомогою їхнього об'єднання як це показано на рисунку 3.1 для вихідного (порожнього) контейнера C.

На етапі добування даних немає необхідності володіти інформацією про первинний контейнер C. Операція декодування полягає у відновленні схованого

повідомлення шляхом проектування кожного блоку S_i, отриманого стеганозображення S на всі базисні функції $\Phi_j \in \Phi, i = 0,..., N-1$. Для цього кожний блок S_i представляється у формі вектора $S_i = (S_{i_0}, S_{i_1},..., S_{i_{n-1}})$, $i = 0,..., N-1$.

Щоб витягти j-ий біт повідомлення з i-го блоку стеганозображення необхідно обчислити коефіцієнт кореляції між Φ_j і прийнятим блоком S_i (представленого у вигляді вектора):

$$\rho(S_i, \Phi_j) = \frac{1}{n}\sum_{z=0}^{n-1} S_{i_z}\Phi_{j_z} = G \cdot \frac{1}{n}\sum_{z=0}^{n-1} E_{i_z}\Phi_{j_z} + \frac{1}{n}\sum_{z=0}^{n-1} C_{i_z}\Phi_{j_z}, \qquad (3.11)$$

де під C_i розуміється одномірний масив, тобто відповідний блок контейнера, представлений у формі вектора.

Припустимо, що масив C_i має випадкову статистичну структуру, тобто покладемо, що другий доданок у правій частині вираження (3.11) близько до нуля і їм можна зневажити. Тоді маємо:

$$\rho(S_i, \Phi_j) \approx G \cdot E_i \cdot \Phi_j = G \cdot \sum_{l=0}^{n-1}\sum_{z=0}^{n-1} m_{i_x}(t) \cdot \Phi_{l_z}\Phi_{j_z}. \qquad (3.12)$$

За аналогією з (3.6) відзначимо, що всі послідовності з безлічі Φ взаємоортогональні, тобто при $l \neq j$ маємо $\rho(\Phi_i, \Phi_j) = 0$. Отже, всіма доданками в правій частині рівності (3.12) при $l \neq j$ можна зневажити. Звідси маємо:

$$\rho(S_i, \Phi_j) \approx G \cdot m_{i_j}(t) \cdot \frac{1}{n}\sum_{z=0}^{n-1} (\Phi_{j_z})^2 = G \cdot m_{i_j}(t). \qquad (3.13)$$

За аналогією з виділенням корисного сигналу (3.8) значення $m_{i_j}(t)$ можуть бути легко відновлені за допомогою знакової функції.

Оскільки $G > 0$ й $n > 0$ знак $\rho(S_i, \Phi_j)$ в (3.13) залежить тільки від $m_{i_j}(t)$, звідки маємо:

$$m_{i_j}(t) = sign\left(\rho\left(S_i, \Phi_j\right)\right) = \begin{cases} -1, \text{при } \rho\left(S_i, \Phi_j\right) < 0; \\ +1, \text{при } \rho\left(S_i, \Phi_j\right) > 0; \\ ?, \text{при } \rho\left(S_i, \Phi_j\right) = 0. \end{cases} \quad (3.14)$$

Якщо $\rho\left(S_i, \Phi_j\right) = 0$ в (3.14) будемо думати, що убудована інформація була втрачена.

Структурна схема вбудовування інформації в контейнер-зображення з використанням прямого розширення спектра для потайливої передачі повідомлень представлена на рисунку 3.2.

З рисунка треба, що процес вбудовування інформаційних повідомлень для потайливої передачі дуже схожий на процес розширення спектра дискретних сигналів у системах зв'язку (див. рис. 3.1). За елементне додавання модульованого повідомлення $E(t)$ з контейнером-зображенням $C(t)$ (див. вираження (3.10)) варто інтерпретувати як накладення помилок $e(t)$ на корисний сигнал у каналі зв'язку $y(t)$. Завдання добування повідомлення $m(t)$ з $S(t)$ на прийомній стороні стеганосистеми еквівалентні завданню детектування $x(t)$ із суміші корисного сигналу й перешкоди $y'(t) = y(t) + e(t)$ в широкополосної системі зв'язку. Інакше кажучи, розглянута стеганосистема успадковує всі переваги широкополосних систем зв'язку: стійкість до несанкціонованого добування убудованих повідомлень (аналог скритності в системі зв'язку), стійкість до руйнування або модифікації убудованих повідомлень (аналог перешкодозахищеності), стійкість до нав'язування помилкових повідомлень (аналог імітостійкісти в системі зв'язку).

Таким чином, використання прямого розширення спектра дискретних сигналів дозволяє здійснити вбудовування інформаційних даних у нерухливі зображення для потайливої передачі й реалізувати, таким чином, стеганографічну захист інформації.

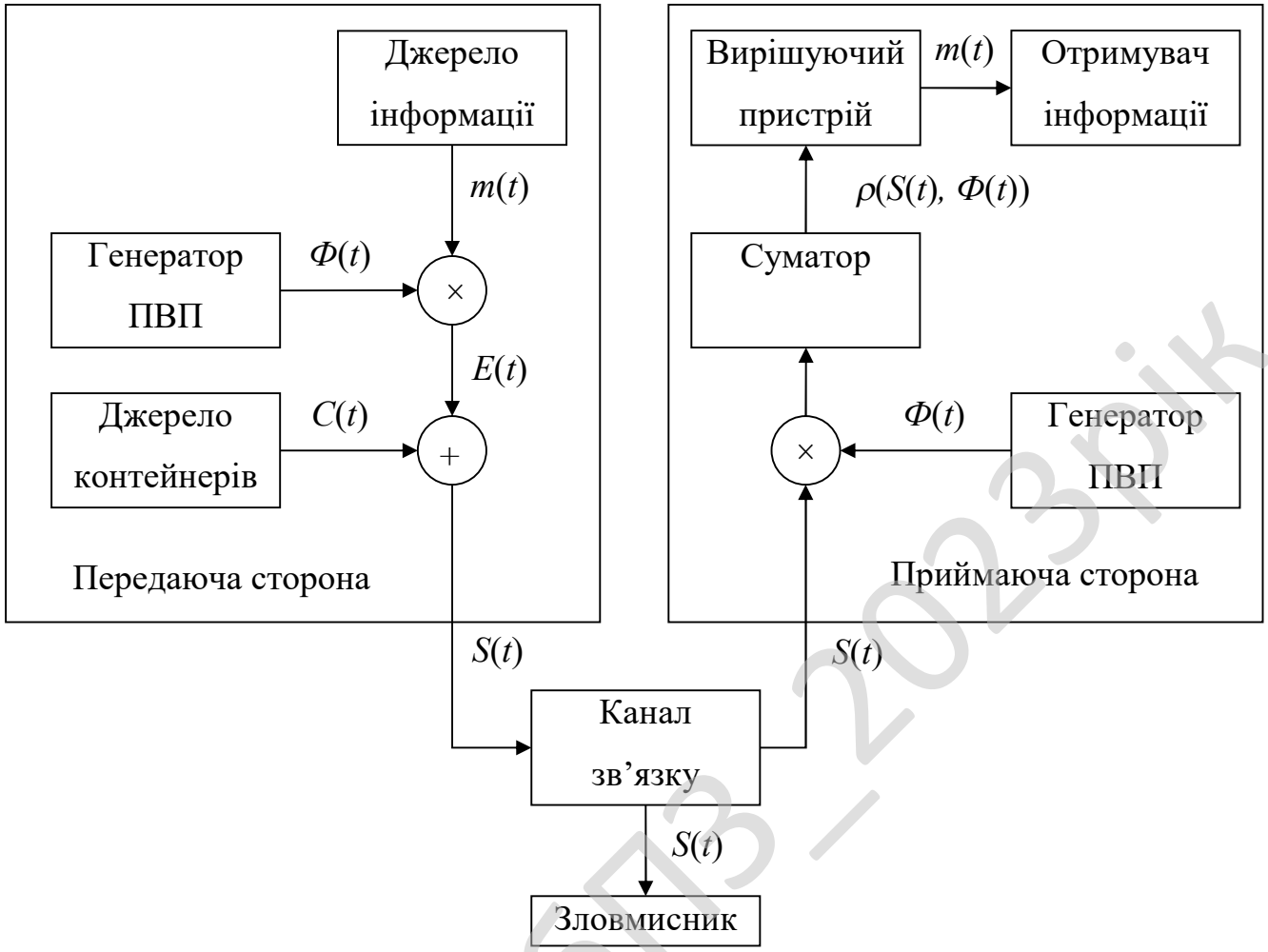

Рисунок 3.2 – Структурна схема вбудовування інформації в контейнер-зображення для потайливої передачі повідомлень

Оцінка ефективності стеганосистеми

Під ефективністю технічної системи в широкому змісті розуміють відповідність результату виконання деякої операції необхідному. При цьому технічна система виступає в ролі засобу реалізації досліджуваної операції.

Стосовно до розглянутого процесу стеганографічна система виступає в ролі технічного засобу реалізації операції, метою якої є приховання від супротивника факту здійснення потайливої передачі інформації. Таким чином, з урахуванням функціонального призначення стеганосистеми, введемо наступні показники ефективності.

1. Пропускна здатність – відношення обсягу V інформації, що вбудовується в контейнер, D до загального обсягу контейнера:

$$Q = \frac{V}{D}. \tag{3.15}$$

2. Обсяг ключових даних (у бітах):

$$l_{Key} = \log_2(|Key|), \tag{3.16}$$

де $|Key|$ – потужність безлічі ключових даних.

3. Стійкість стеганографічного методу будемо оцінювати як величину, зворотну потужності безлічі секретних ключових даних. Її можна трактувати як імовірнісний показник підбора секретного ключа:

$$W = \frac{1}{|Key|} = 2^{-l_{Key}}. \tag{3.17}$$

4. Величина внесених перекручувань як процентне відношення середньоарифметичного всіх абсолютних значень Δ-змін даних контейнера до максимально можливого значення $\Delta_{\max}$:

$$I = \frac{\Delta_{ср}}{\Delta_{\max}} \cdot 100 = \frac{100}{\Delta_{\max} \cdot D} \cdot \sum_{i=1}^{D} |\Delta_i|, \tag{3.18}$$

де Δ_i – Δ-зміни i-го елемента контейнера.

5. Імовірність помилкового добування інформаційних даних повідомлення:

$$P_{ош} = \lim_{D\to\infty} \frac{V_{ош}}{D} = 1 - \lim_{D\to\infty} \frac{V - V_{ош}}{D}, \tag{3.19}$$

де $V_{ош}$ – обсяг помилково витягнутих даних.

Використовуючи показники (3.15) – (3.19) оцінимо ефективність розглянутого стеганографічного методу захисту інформації.

1. Пропускна здатність. На кожний n-елементний блок S_i заповненого контейнера (стеганограми) доводиться n-бітний вектор убудованого повідомлення m_i (див. вираження (3.9), (3.10)). Отже, $Q = \frac{1}{B}$, де B – обсяг даних,

що доводиться на один елемент контейнера. Для випадку вбудовування в растрові дані зображення (колірна модель R,G,B) з 8 бітним кодуванням кожного кольору маємо $B = 8$ й $Q = \frac{1}{8}$.

2. Обсяг ключових даних. Ключовими даними є ансамбль дискретних сигналів, утворений рядками матриці Адамара порядку n. Отже, під безліччю ключових даних варто розуміти безліч різних (неізоморфних) матриць Адамара, кожна з матриць задає ансамбль дискретних сигналів. В [7] отримані деякі оцінки потужності M_A цієї безлічі, які наведені в таблиці 3.2.

Таблиця 3.2 – Число ансамблів дискретних сигналів Уолша-Адамара [7]

n	M_A
64	19
100	1
256	54
512	102
1024	162
2000	9
4000	16
10000	10

Наведені оцінки потужності M_A дають оцінку числа ансамблів дискретних сигналів Уолша-Адамара, тобто оцінку потужності нееквівалентних ключів стеганосистеми. Отже, обсяг ключових даних оцінюється як $l_{key} = \log_2(M_A)$.

3. Імовірність підбора секретного ключа $W = (M_A)^{-1}$

4. Для оцінки величини внесених перекручувань скористаємося вираженням (3.10). Другий доданок у правій частині (3.10) визначає величину Δ-змін елементів даних контейнера. Співмножник E_i формується в результаті

підсумовування n дискретних сигналів (приймаючі значення ±1) з відповідними полярностями (що задаються $m_{i_j}(t)$). Отже, всі елементи E_i будуть приймати значення з діапазону $[-n,...,+n]$, а відповідні Δ-зміни елементів контейнера не будуть перевищувати $|\Delta_i| \le n \cdot G$. Звідки маємо верхню оцінку величини внесених перекручувань:

$$I = \frac{\Delta_{cp}}{\Delta_{\max}} \cdot 100 \le \frac{n \cdot G}{\Delta_{\max}} \cdot 100 \qquad (3.20)$$

Для випадку вбудовування в растрові дані зображення (колірна модель R,G,B) з 8 бітним кодуванням кожного кольору й використання дискретних сигналів з $n = 256$ навіть при $G = 1$ внесені перекручування можуть досягати 100%. Знизити внесені перекручування можна за рахунок скорочення числа вбудовуємих біт даних m_{i_j} (зменшивши число доданків в (3.9)), що неминуче приведе до зниження пропускної здатності стеганографічного каналу зв'язку.

5. Імовірність помилкового добування. Добування інформаційного повідомлення, також як і при організації перешкодозахищеної системи зв'язку (див. (3.3) – (3.5)), здійснюється кореляційним способом (див. (3.11) – (3.14)). Отже, помилка добування відбудеться при зміні знака коефіцієнта кореляції $\rho(S_i, \Phi_j)$ у вираженні (3.14).

Представимо коефіцієнт $\rho(S_i, \Phi_j)$ у вигляді:

$$\rho(S_i, \Phi_j) = \rho(C_i + E_i \cdot G, \Phi_j) = \rho(C_i, \Phi_j) + \rho(E_i \cdot G, \Phi_j)$$

Останній доданок не змінює знак $\rho(S_i, \Phi_j)$, подію:

$$\rho(S_i, \Phi_j) = \rho(E_i \cdot G, \Phi_j)$$

відповідає безпомилковому добуванню повідомлення (див. (3.12), (3.13)).

Отже, помилка добування інформаційного біта m_{i_j} повідомлення відбудеться при настанні події

$$\left|\rho(C_i, \Phi_j)\right| > \rho\left|E_i \cdot G, \Phi_j\right| = \left|G \cdot m_{i_j}\right| = G, \qquad (3.21)$$

тобто у тому випадку, коли абсолютне значення коефіцієнта кореляції використовуваного для вбудовування біта m_{i_j} дискретного сигналу Φ_j із блоком контейнера C_i, у який цей біт вбудовується, перевершить коефіцієнт підсилення G.

Таким чином, запишемо:

$$P_{ош} = P\left(\left|\rho(C_i, \Phi_j)\right| > G\right),$$

де $P(x)$ – імовірність настання випадкової події x.

Інакше кажучи, правильне добування убудованого повідомлення є випадковою подією, імовірність $P_{б.ош}$ якого безпосередньо зв'язана зі статистичними властивостями використовуваного контейнера-зображення. Для безпомилкового добування повідомлення

$$P_{ош} = 0, P_{б.ош} = 1 - P_{ош} = 1, \qquad (3.22)$$

варто прагнути до взаємного ортогональності окремих фрагментівів зображення C_i і використовуваних як секретні ключі дискретних сигналів Φ_j. У цьому випадку подія:

$$\left|\rho(C_i, \Phi_j)\right| = 0 < G \ ,$$

для всіх $i = 0,..., N-1$ є достовірним і виконується (3.22).

У теж час, як показали експериментальні дослідження, коефіцієнт кореляції, як правило, значно більше нуля $\left|\rho(C_i, \Phi_j)\right| >> 0$ й дуже часто виникає подія (3.21). Справа в тому, що елементи дискретних сигналів $\Phi_j \in \Phi$ приймають значення $\begin{cases} +1 \\ -1 \end{cases}$, а відповідний нормований коефіцієнт кореляції $\rho(\Phi_i, \Phi_j)$ за абсолютним значенням не перевершує довжини n послідовності (див. (3.2)) і лежить у діапазоні $[0,...,1]$, звідки властиво й треба умова (3.21).

Однак елементи контейнера C_i приймають значення із числового поля $[0,..., Y]$, розмірність якого задається способом кодування дані зображення.

Наприклад, при вбудовуванні інформації в растрові дані зображення (колірна модель R,G,B) з 8 бітним кодуванням кожного кольору відповідні C_i приймають значення з діапазону цілих чисел $[0,...,255]$. Інакше кажучи, абсолютне значення нормованого щодо n коефіцієнта кореляції $\left|\rho(C_i, \Phi_j)\right|$ буде лежати в діапазоні $[0,..., Y]$ й для безпомилкового добування всіх біт повідомлення (3.21) необхідно виконати умову $G > Y$.

У теж час підвищення G веде до неминучого росту величини внесених перекручувань (3.20), які при $I > 2...3\%$ (поріг зорової чутливості людини) стають помітні сторонньому спостерігачеві [1, 2], що компрометує стеганоканал і унеможливлюють використання розглянутої стеганосистеми.

Таким чином, у ході досліджень виявлені наступні протиріччя, що лежать в основі розробки й використання стеганографічних систем з розширенням спектра дискретних сигналів:

– імовірність правильного добування убудованих даних $P_{б.ош}$ лежить у прямої залежності від величини внесених перекручувань I;

– величина внесених перекручувань I лежить у прямої залежності від обсягу даних, що вбудовуються біт,, тобто від пропускної здатності стеганоканалу Q;

– імовірність правильного добування убудованих даних $P_{б.ош}$ безпосередньо залежить від статистичних властивостей використовуваного контейнера-зображення.

Для експериментального дослідження ефективності розглянутого методу вбудовування повідомлень у нерухливі зображення розроблена програмна реалізація, отримані наступні емпіричні оцінки:

– залежності величини внесених перекручувань I від пропускної здатності Q стеганоканалу;

– залежності величини внесених перекручувань I і частоти помилок добування $P^*_{ош} \approx P_{ош}$ від коефіцієнта підсилення G;

– залежності величини внесених перекручувань I від частоти помилок добування $P^{*}_{ош} \approx P_{ош}$.

Дослідження проводилися при вбудовуванні інформаційних даних у растрові дані зображення (колірна модель R,G,B) з 8 бітним кодуванням кожного кольору.

У результаті проведених досліджень показано, що використання в стеганографічних цілях прямого розширення спектра дискретних сигналів дозволяє здійснити потайливе вбудовування інформаційних повідомлень у нерухливі зображення. Завдання добування повідомлення на прийомній стороні стеганосистеми еквівалентні завданню виявлення інформації із суміші корисного сигналу й перешкоди в широкополосної системі зв'язку.

У ході досліджень виявлені наступні недоліки стеганографічних систем з розширенням спектра дискретних сигналів: імовірність правильного добування убудованих даних залежить від величини внесених перекручувань, що у свою чергу залежить від забезпечуваної пропускної здатності стеганоканалу. Інакше кажучи, практична побудова стеганосистеми сполучена з пошуком компромісу між величиною внесених перекручувань, імовірністю правильно добування повідомлення на прийомній стороні й забезпечуваній пропускній здатності. Крім того, у ході досліджень установлено, що ймовірність правильного добування убудованих даних безпосередньо залежить від статистичних властивостей використовуваного контейнера-зображення.

Перспективним напрямком подальших досліджень, на думку авторів, є використання більших ансамблів слабокорельованих (квазіортогональних) дискретних сигналів для побудови стеганосистем із прямим розширенням спектра. Це дозволить із однієї сторони без значного підвищення внесених перекручувань у контейнер-зображення істотно підвищити пропускну здатність стеганоканалу. З іншого боку, за рахунок адаптивного формування (вибору) дискретних сигналів за критерієм мінімізації коефіцієнта кореляції з контейнером зображенням це дозволить істотно знизити ймовірність помилкового добування убудованих даних.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.3.

Система включає дві процедури:

– процедуру вбудовування конфіденційної інформації в зображення контейнер;

– процедуру вилучення конфіденційної інформації із зображення контейнера.

Робота системи відбувається наступним чином.

Спершу опишемо процедуру вбудовування конфіденційної інформації в зображення контейнер.

Для цього береться повідомлення, над яким відбувається операція кодування у вибраному зображені, яке є контейнером.

Контейнер – це те зображення, куди буде приховано записано повідомлення.

Принцип кодування наступний:

1. Обчислюється контрольна сума пароля

2. Обчислюється контрольний добуток пароля

3. Всі закодовані дані представляються як масив байтів

4. Від кожного байта даних віднімається байт контрольної суми пароля

5. З результатом попереднього обчислення робиться XOR з байтом контрольного добутку пароля

6. До результату попереднього обчислення додається код відповідного символу з рядка пароля.

7. Як тільки рядок пароля закінчується знову переходимо на його початок.

Для реалізації цього методу відбувається генерація стегоключа, за допомогою якого повідомлення буде зашифроване.

Щоб стегоключ неможливо було взнати зловмиснику, він зашифровується алгоритмом DES, та передається отримувачу повідомлення.

Рисунок 3.3 – Функціональна схема системи

Для більш високої надійності передачі даних, повідомлення кодується за допомогою перешкодостійкого кодека Хеммінга.

Після цього відбувається передача закодованого зображення з прихованим повідомленням, по каналам зв'язку.

На приймальній стороні отримують зображення й реалізують процедуру вилучення конфіденційної інформації із зображення контейнера.

Це відбувається наступним чином.

Спершу отримане повідомлення перевіряють на наявність помилок, за рахунок застосування кодеку Хеммінга.

Якщо є помилки, то вони виправляються, за рахунок властивостей цього кодеку з виявлення та виправлення помилок.

Після цього відбувається дешифрування ключа. За допомогою отриманого ключа відбувається створення файлу з повідомлення та очищення контейнера.

Після цього на стороні отримувача є інформація, яка була прихована у зображені, та саме зображення, яке може служити контейнером для наступної інформації, яку треба приховано передати.

Так, як ключ кодується за допомогою алгоритму DES, то наведемо принцип роботи цього алгоритму.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів.

На першому з них виконується початкова перестановка (`IP`) 64-бітного вихідного тексту (забілювання), під час якої біти з у відповідності зі стандартною таблицею.

Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки.

На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями.

Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

Опишемо спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів розробленої системи зображена на рисунку 3.4. З нього видно, що процеси взаємодіють наступним чином.

Спершу завантажується процес виведення головного вікна програми. Він взаємодіє з наступними основними процесами, які відбуваються у системі:

– Процес перевірки на наявність стегоповідомлення.

– Процес створення стегоповідомлення.

– Процес вилучення стегоповідомлення.

Процес перевірки на наявність стегоповідомлення взаємодіє з процесом аналізу зображення на наявність вбудованої інформації. Цей процес, у свою чергу взаємодіє з процесом виведення аналізу результатів.

Процес створення стегоповідомлення взаємодіє з процесом вибору зображення контейнера.

Процес вибору зображення контейнера взаємодіє з процесом генерації стегоключа.

Рисунок 3.4 – Діаграма взаємодії процесів

Процес генерації стегоключа взаємодіє з процесом вбудовування стегоповідомлення.

Процесом вбудовування стегоповідомлення взаємодіє з наступними процесами:

– Процес перешкодостійкого кодування повідомлення.

– Процес збереження зображення зі стегоповідомлення.

– Процес шифрування стегоключа.

Останній процес взаємодіє з процесом збереження зашифрованого ключа.

Процес вилучення стегоповідомлення взаємодіє з процесом відкриття зображення.

Процес відкриття зображення взаємодіє з процесом відкриття файлу зі стегоключем.

Процес відкриття файлу зі стегоключем взаємодіє з процесом дешифрування стегоключа.

Процес дешифрування стегоключа взаємодіє з процесом читання повідомлення бітів зображення.

Процес читання повідомлення бітів зображення взаємодіє з процесом декодування повідомлення та виправлення помилок.

Процес декодування повідомлення та виправлення помилок взаємодіє з процесом виведення повідомлення на екран.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

З рисунку видно, що після запуску програми спочатку відбувається вивід основного вікна програми. Потім користувач обирає одну з наступних дій:

– Створення стегоповідомлення.

– Читання стегоповідомлення.

– Перевірка на наявність стегоповідомлення.

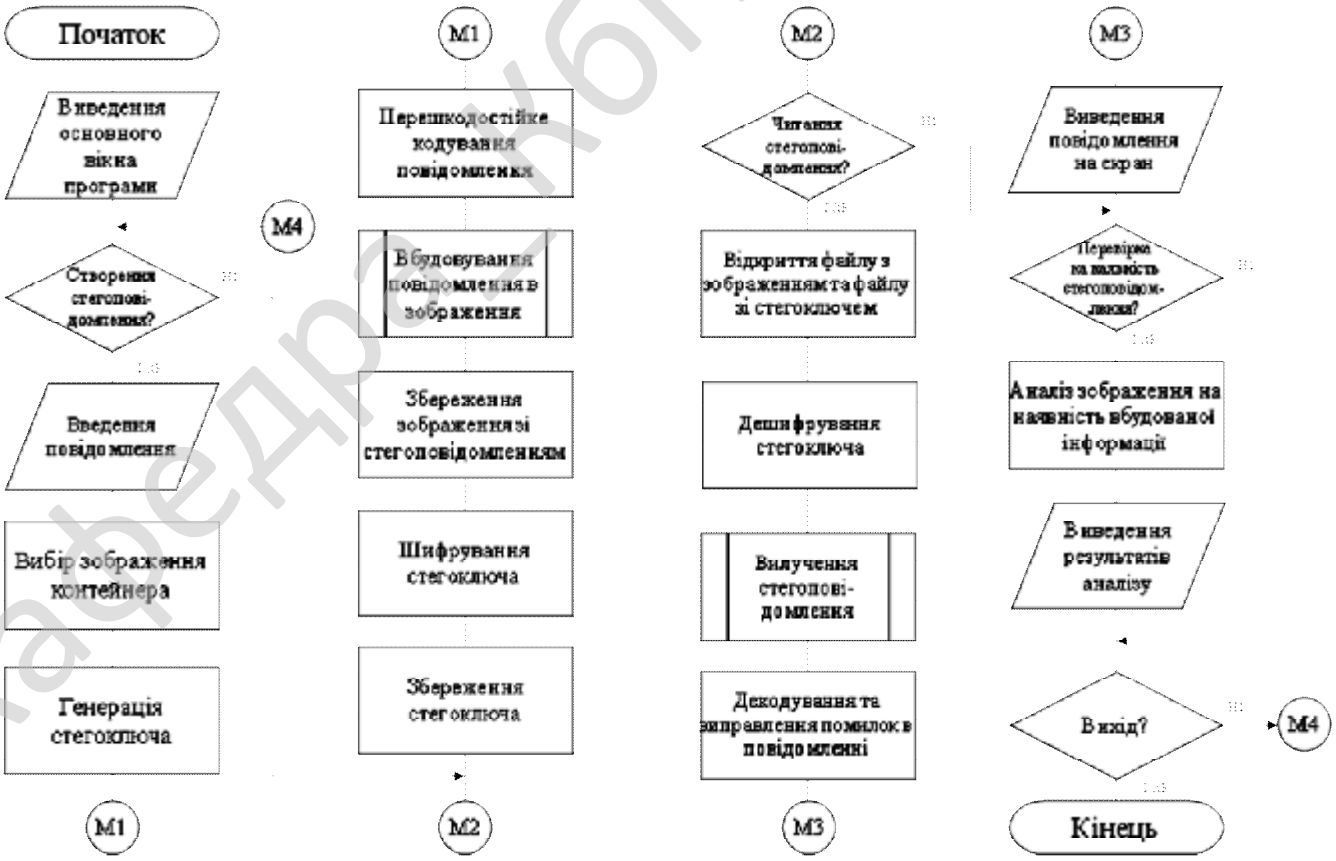

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає створення стегоповідомлення, тоді виконуються наступні дії:

– Вводиться повідомлення.

– Відбувається вибір зображення контейнера.

– Генерація стегоключа.

– Перешкодостійке кодування повідомлення.

– З_апускається процедура вбудовування повідомлення у зображеня.

– Зберігається зображення зі стегоповідомленям.

– Шифрується стегоключ.

– Зберігається стегоключ.

Якщо користувач обирає читання стегоповідомлення, тоді виконуються наступні дії:

– Відкривається файл з збереженим стегоключем.

– Дешифрується стегоключ.

– Запускається процедура вилучення стегоповідомлення.

– Відбувається декодування та виправлення помилок у повідомлені.

– Виводиться повідомлення на екран.

Якщо користувач обирає перевірку на наявність у картинці стегоповідомлення, тоді виконуються наступні дії:

– Відбувається аналіз зображення на наявність вбудованої інформації.

– Виводяться результати аналізу.

Після цього користувач обирає, чи працювати йому далі з програмою, або ні.

На рисунку 4.2 зображена блок-схема роботи підпрограми створення стегоповідомлення. Вона працює наступним чином.

Спершу відбувається читання стегоключа, повідомлення та зображення.

Після цього відбувається розбиття зображення на фрагменти.

Наступним кроком є ініціалізація лічильника фрагментів.

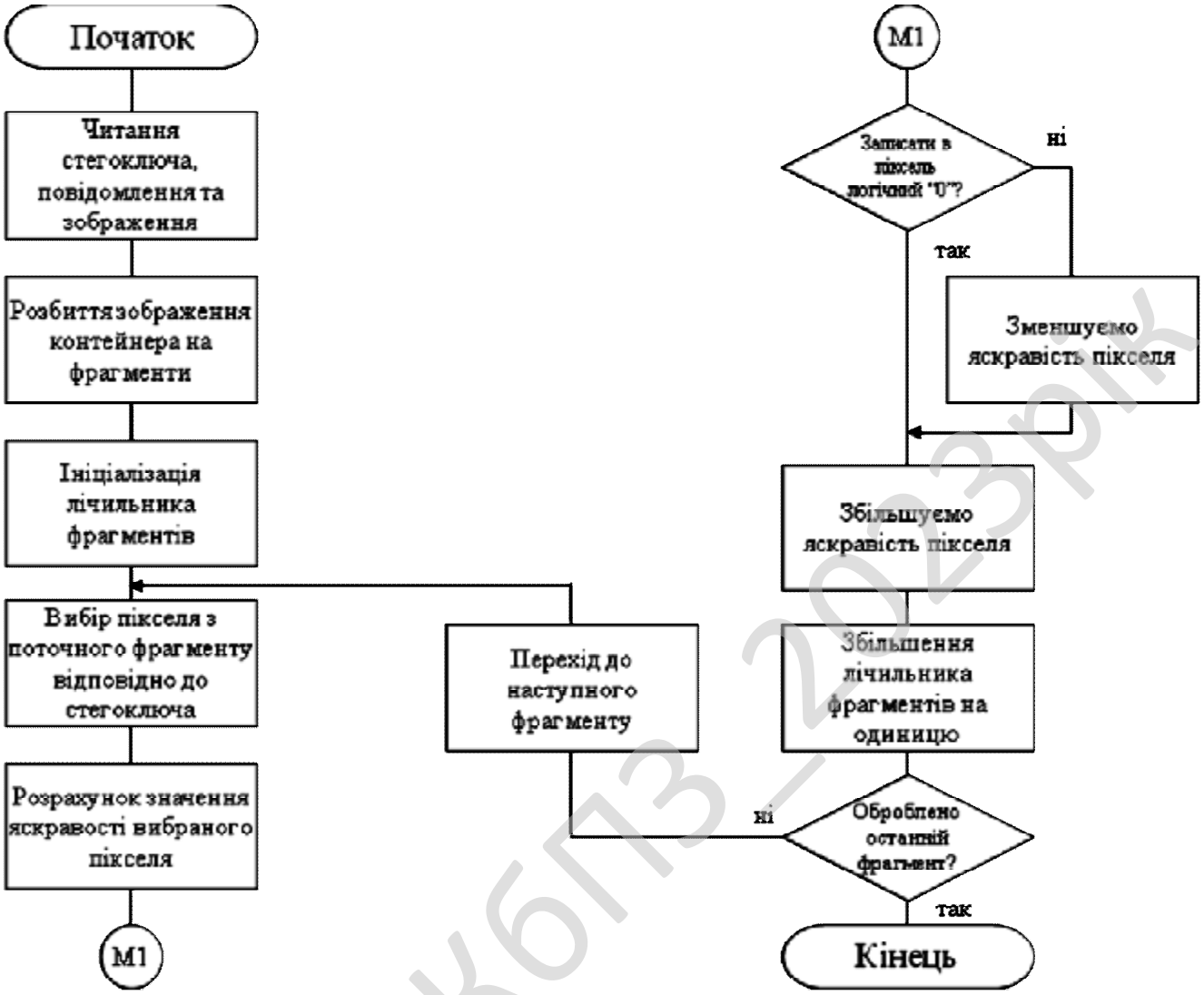

Рисунок 4.2 – Блок-схема роботи підпрограми створення стегоповідомлення

Після цього відбувається вибір пікселя з поточного фрагменту відповідно до стегоключа.

Наступним етапом є розрахунок значення яскравості обраного пікселя.

Якщо у піксель потрібно записати «0», толі збільшуємо яскравість пікселя. У противному випадку, зменшуємо яскравість пікселя.

Далі збільшується лічильник на одиницю, й переходимо до наступного пікселя згідно стегоключа. Якщо оброблено останній фрагмент, тоді відбувається завершення процедури.

У іншому випадку, відбувається перехід до наступного фрагменту, й процедура повертається на етап вибору пікселя з поточного фрагменту, відповідно до стегоключа.

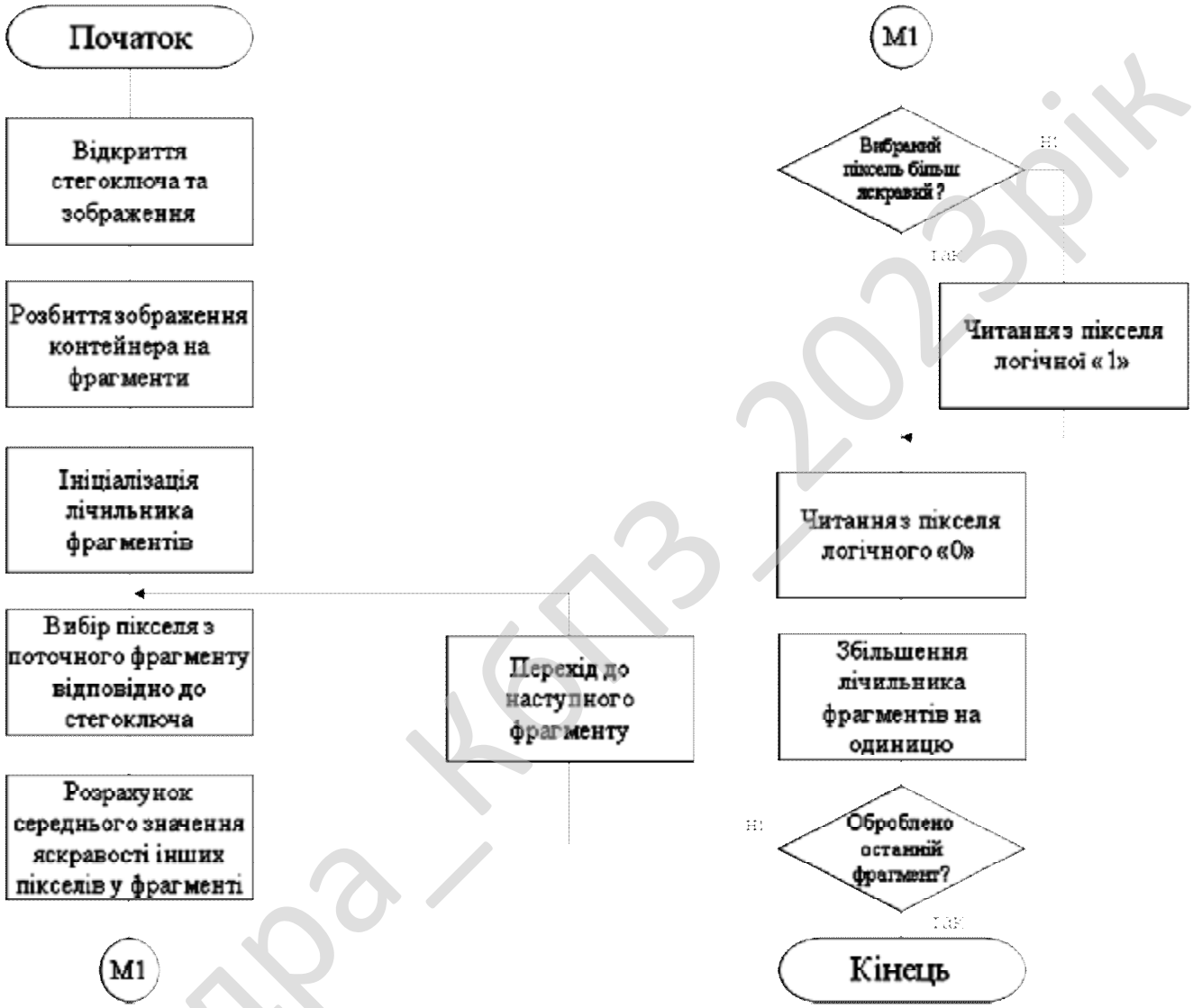

Рисунок 4.3 – Блок-схема роботи підпрограми читання стегоповідомлення

На рисунку 4.3 зображено блок-схему роботи підпрограми читання стегоповідомлення. Вона працює наступним чином.

Спершу відбувається відкриття стегоключа та зображення.

Після цього відбувається розбиття контейнера на фрагменти.

Наступним кроком є ініціалізація лічильника фрагментів.

Після цього відбувається вибір пікселя з поточного фрагменту відповідно до стегоключа.

Наступним кроком є розрахунок середнього значення яскравості інших пікселів у фрагменті.

Якщо обраний піксель більш яскравий, тоді відбувається читання з пікселя логічного «0», у іншому випадку відбувається читання з пікселя логічногї «1».

Далі збільшується лічильник фрагментів на одиницю, й переходимо до наступного пікселя згідно стегоключа. Якщо оброблено останній фрагмент, тоді відбувається завершення процедури.

У іншому випадку, відбувається перехід до наступного фрагменту, й процедура повертається на етап вибору пікселя з поточного фрагменту, відповідно до стегоключа.

Приведемо опис частини коду розробленого програмного забезпечення. У файлі-контейнері ми будемо міняти останній (або декілька останніх) біт, у які й будемо записувати інформацію, яку треба сховати. Спочатку прочитаємо заголовок bmp, визначимо – чи підходить він нам, потім перейдемо на тіло bmp, тобто на те, що містить сам малюнок. Записувати будемо так: візьмемо розмір файлу, розіб'ємо на біти – запишемо, візьмемо довжину імені файлу, розіб'ємо на біти – запишемо, візьмемо саме ім'я файлу – розіб'ємо на біти, запишемо, потім уже записуємо саме тіло файлу.

Для початку напишемо процедуру, що буде читати заголовок bmp, а потім видавати нам всю потрібну інформацію про файл у вигляді запису. Для початку визначимо тип запису:

```
type
TImageFileInfo=record
File:String; //Ім'я bmp
width:dword; //ширина малюнка
height:dword; //висота малюнка
smeshenie:dword; //Зсув малюнка від початку файлу
FileSize:dword; //Розмір bmp
ImageSize:dword; //Розмір малюнка
InfoHide:dword; //Скільки інформації можна сховати
```

```
        RLE:byte; //Рівень стиску – повинен бути дорівнює 0
        BitsPerPixel:byte; //Розрядність малюнка (біт на піксель) – нам потрібні
24-х розрядні
        Signature:string[2]; //Перші два байти файлу – bmp-файлів повинні бути 'BM'
        end;
```

Тепер запишемо саму процедуру, що буде цей запис заповнювати. Крім того, вона буде відразу створювати StringList, у який буде поміщати опис файлу.

```
        procedure GetFileFormat(File:string;var StrLst:TStringList;var
ImageInfo:TImageFileInfo);
        var
        Stream:TMemoryStream;
        ch:char;
        size,dwrd:dword;
        wrd:word;
        bt:byte;
        str:string;
        begin
        StrLst:=TStringList.Create;
        stream:=TMemoryStream.Create;
        stream.LoadFromFile(File); //Завантажуємо bmp у тільки що створений
MemoryStream
        StrLst.Add(' ***Технічна інформація про файл*** ');
        StrLst.Add(' ***Файл зображення: '+File+'***');
        stream.Read(ch,sizeof(ch)); //Читаємо сигнатуру
        str:=ch;
        stream.Read(ch,sizeof(ch));
        str:=str+ch;
        StrLst.Add(' ***********');
        ImageInfo.Signature:=str; //Занесемо в запис сигнатуру
        if ImageInfo.Signature ='BM' then //Якщо сигнатура = 'BM', те значить все
нормально – продовжуємо
        begin
        ImageInfo.File:=File; //Занесемо в запис ім'я bmp
        stream.Read(dwrd,sizeof(dwrd)); //Уважаємо із заголовка розмір файлу
        ImageInfo.Filesize:=dwrd;
        size:=round(dwrd/1024); //Переведемо в кілобайти для подальшого виводу в
опис
        StrLst.add('Розмір файлу із зображенням: '+IntToStr(dwrd)+' (
'+InTToStr(size)+' kb)');
        stream.Read(dwrd,sizeof(dwrd)); //Зарезервовано – не використовується
        stream.Read(dwrd,sizeof(dwrd)); //Зарезервовано – не використовується
```

```
        ImageInfo.smeshenie:=dwrd; //Якщо щось далі не ясно – дивися опис формату
        StrLst.Add('Зсув даних бітового образа від заголовка: '+IntToStr(dwrd)+'
байт');
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Розмір BITMAPINFOHEADER: '+IntToStr(dwrd)+' байт');
        stream.Read(dwrd,sizeof(dwrd));
        ImageInfo.width:=dwrd;
        StrLst.Add('Ширина зображення: '+IntToStr(dwrd)+' пікселів');
        stream.Read(dwrd,sizeof(dwrd));
        ImageInfo.height:=dwrd;
        StrLst.Add('Висота зображення: '+IntToStr(dwrd)+' пікселів');
        stream.Read(wrd,sizeof(wrd));
        StrLst.Add('Число бітових площин пристрою: '+IntToStr(wrd));
        stream.Read(wrd,sizeof(wrd));
        ImageInfo.BitsPerPixel:=wrd;
        StrLst.Add('Глибина зображення (число бітів на піксель): '+IntToStr(wrd));
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Тип стиску ( 0-відсутній): '+IntToStr(dwrd));
        stream.Read(dwrd,sizeof(dwrd));
        size:=round(dwrd/(1024));
        StrLst.Add('Розмір картинки в байтах: '+IntToStr(dwrd)+' (
'+IntToStr(size)+' kb)');
        ImageInfo.ImageSize:=dwrd;
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Горизонтальний дозвіл пристрою (піксель/м): '+IntToStr(dwrd));
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Вертикальний дозвіл пристрою (піксель/м): '+IntToStr(dwrd));
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Число використовуваних квітів: '+IntToStr(dwrd));
        stream.Read(dwrd,sizeof(dwrd));
        StrLst.Add('Число важливих квітів: '+IntToStr(dwrd));
        ImageInfo.InfoHide:=round((ImageInfo.width*ImageInfo.height*3)/8);
        StrLst.Add(' ***********');
        StrLst.Add('');
        StrLst.Add('Можна сховати інформації '+IntToStr(ImageInfo.infohide)+'байт
('+IntToStr(round((ImageInfo.infohide)/1024))+'kb)');
        StrLst.Add('');
        end
        else
        StrLst.Add('Помилка: це не файл формату BMP.');
        stream.Free;
        end;
```

Тепер займемося написанням процедури, що буде розбивати байт на біти, і видавати результат у вигляді масиву 1 і 0. Для цього скористаємося логічною командою AND – логічне И.

Вираження істинно, коли щирі обидва біти. А так само будемо використовувати SHL – логічне зрушення вліво, операція застосовується тільки до групи бітів, одного із цілочислених типів Паскаля, наприклад до байта, слова й т.д. Зрушення байта вліво на один розряд. Байт зміщається вліво на один або більше розрядів, позиції праворуч заміщаються нулями, позиції ліворуч губляться. Ми будемо це використовувати, щоб циклічно порівнювати кожний біт з одиницею. Тобто ми будемо поступово зрушувати одиницю вліво, а всі інші позиції будуть рівні 0. Це буде виглядати так: якщо SByte and (1 shl num)<>0, то в байті SByte – біт з номером num = 1, інакше цей біт = 0. Поставивши цикл for від 0 до 7 – ми переберемо весь байт, і створимо масив з 1 і 0.

```
type
BitsArray=array[0..7] of byte;
procedure HexToBin(x:byte;var a:BitsArray);
var
bit,i:byte;
begin
for i:=7 downto 0 do
begin
if (X and (1 shl i))<>0 then bit:=1 else bit:=0;
a[i]:=bit;
end;
end;
```

Тепер напишемо процедуру для зворотної дії, тобто збору байта з масиву 1 і 0.

```
procedure BinToHex(var x:byte;a:BitsArray);
var i:dword;
bit,t:byte;
begin
x:=0;
for i:=7 downto 0 do
begin
bit:=a[i];
if (bit and (1 shl 7)) <> 0 then bit:=1;
```

```
x:=x or bit;
if i<>0 then x:=x shl 1;
end;
end;
```

Тепер напишемо дві процедури для установки/скидання біта певного номера.

```
function SetBit(SByte: byte; num: byte): byte;
begin
Result := SByte or (1 shl num);
end;
function ResetBit(SByte: byte; num: byte): byte;
begin
Result := SByte and not (1 shl num);
end;
```

Тут SByte – вихідний байт, а num – номер біта, якому треба скинути. Прошу врахувати, що біти природно йдуть від нульового номера й до сьомого.

А тепер перейдемо до самого цікавого – до написання процедури, що буде ховати файл-даних у файл-контейнер (bmp). Відразу поясню загальну логіку. Спочатку ми одержуємо інформацію про bmp (GetFileFormat), потім дивимося: якщо в дану bmp влізе файл такого обсягу, то продовжуємо. Вантажимо обидва файли на згадку за допомогою TMemoryStream. Потім в bmp переходимо на тіло малюнка, тобто робимо в memorystream'е position:=ImageFile.Smeshenie. Далі беремо розмір файлу даних (dword, тобто 4 байти, тобто 32 біта) і записуємо в 32 байта bmp, скрізь міняючи біт самого дрібного розряду. Тобто ми максимум змінимо значення байта кольору в малюнку на ±1, що візуально зовсім не помітно (дослідним шляхом перевірено що міняти можна й 2 біти, візуально не міняючи картинку). Далі записуємо у файл-контейнер довжину імені файлу-даних. Виходячи з того що довжина буде <=255 символам ми відводимо під неї один байт, тобто міняємо останній біт в 8 байтах bmp. Далі записуємо ім'я файлу контейнера. Для цього нам знадобляться два цикли: один зовнішній, для того щоб пройтися по кожному символі ім'я, а інший внутрішній, котрий буде записувати кожний символ у вигляді 8 битв bmp. Далі – записуємо саме тіло файлу контейнера. Послідовно розбиваємо кожний байт на біти й пишемо.

```
        procedure IHide(ImageInfo:TImageFileInfo; DFile:string;
SaveDialog:TSaveDialog);
        var
        sz,n,z:dword;
        len,v,i,xi,xd:byte;
        FIStream,FDStream:TMemoryStream;
        BAr,BArLog:BitsArray;
        tmp,FName:string;
        f:File;
        begin
        //В FDStream довантажується файл даних, а в FIStream файл-контейнер
        FDStream:=TMemoryStream.Create;
        FDStream.LoadFromFile(DFile);
        FIStream:=TMemoryStream.Create;
        //Отут ідуть різні перевірки на те підходить цей файл на роль контейнера чи
ні
        if ImageInfo.Signature<>'BM' then
        begin
        MessageBox(Application.handle,'Це не зображення формату Windows
Bitmap.','Error',MB_ICONERROR);
        exit;
        end;
        if ImageInfo.BitsPerPixel<>24 then
        begin
        MessageBox(Application.handle,'Зображення повинне бути 24-х
бітним!','Error',MB_ICONERROR);
        exit;
        end;
        if ImageInfo.RLE<>0 then
        begin
        MessageBox(Application.handle,'У даному зображенні використовується
стиск.','Error',MB_ICONERROR);
        exit;
        end;
        if (ImageInfo.InfoHide-300) begin
        MessageBox(Application.handle,'Це зображення занадто мало, щоб помістити в
себе стільки інформації.','Error',MB_ICONERROR);
        exit;
        end;
        //Якщо файл пройшов всі перевірки - довантажуємо його на згадку
        FIStream.LoadFromFile(ImageInfo.File);
        FIStream.Position:=ImageInfo.smeshenie;
```

```
        //Розмір файлу даних поміщаємо в sz
        sz:=FDStream.size;
        //Записуємо розмір файлу який ховаємо в bmp
        for n:=31 downto 0 do
        begin
        //Читаємо байт із файлу-контейнера
        FIStream.Read(xi,sizeof(xi));
        //Якщо в sz'е біт з номером n дорівнює 1, то встановлюємо 0-ой біт у тільки
що зчитаному
        //байте, ну а якщо n'ный біт скинутий, те скидаємо 0-ой біт...
        if (sz and (1 shl n))<>0 then xi:=SetBit(xi,0)
        else xi:=ResetBit(xi,0);
        //Відкочуємося на позицію назад, щоб перезаписати той біт який ми вважали,
але
        //тільки тепер зі зміненим нульовим бітом
        FIStream.Position:=FIStream.Position-1;
        FIStream.Write(xi,sizeof(xi));
        end;
        //Одержуємо ім'я файлу даних без шляху до файлу
        tmp:=ExtractFileName(DFile);
        //Одержуємо довжину імені
        len:=Length(tmp);
        //Розбиваємо байт у якому зберігається довга ім'я на масив 1 і 0
        HexToBin(len,BAr);
        //Записуємо довжину імені у файл-контейнер
        for n:=7 downto 0 do
        begin
        FIStream.Read(xi,sizeof(xi));
        if BAr[n]=1 then xi:=SetBit(xi,0);
        if BAr[n]=0 then xi:=ResetBit(xi,0);
        FIStream.Position:=FIStream.Position-1;
        FIStream.Write(xi,sizeof(xi));
        end;
        //Записуємо саме ім'я файлу в bmp
        for n:=1 to len do
        begin
        //Беремо символ з номером n, одержуємо його ascii-код, розбиваємо отримане
на біти й //записуємо
        v:=ord(tmp[n]);
        HexToBin(v,BAr);
        for i:=7 downto 0 do
        begin
```

```
FIStream.Read(xi,sizeof(xi));
if BAr[i]=1 then xi:=SetBit(xi,0);
if BAr[i]=0 then xi:=ResetBit(xi,0);
FIStream.Position:=FIStream.Position-1;
FIStream.Write(xi,sizeof(xi));
end;
end;
//Пишемо в bmp саме тіло файлу
//кожний байт розбиваємо на біти й записуємо
for n:=0 to ( sz-1) do
begin
FDStream.Read(xd,sizeof(xd));
HexToBin(xd,BAr);
for i:=7 downto 0 do
begin
FIStream.Read(xi,sizeof(xi));
if BAr[i]=1 then xi:=SetBit(xi,0);
if BAr[i]=0 then xi:=ResetBit(xi,0);
FIStream.Position:=FIStream.Position-1;
FIStream.Write(xi,sizeof(xi));
end;
end;
SetLength(FName,0);
//Викликаємо... ні, не парфумів, а діалог збереження файлів
if SaveDialog.Execute then FName:=SaveDialog.FileName;
//Зберігаємо
FIStream.SaveToFile(FName);
//Звільняємо змінні
FDStream.Free;
FIStream.Free;
ShowMessage('Done!');
end;
```

Напишемо процедуру, що це все буде витягати з картинки.

```
procedure IExtract(IFile:string;SaveDialog:TSaveDialog);
var
IInfo:TImageFileInfo;
Lst:TStringList;
IFstream,DFStream:TMemoryStream;
n,i,sz:dword;
a:BitsArray;
b:array[0..31] of byte;
```

```
bit,len,x,z:byte;
DFile:string;
begin
Lst:=TStringList.Create;
//Одержуємо інформацію з bmp
GetFileFormat(IFile,Lst,IInfo);
//Вантажимо все на згадку
IFstream:=TMemoryStream.Create;
DFStream:=TMemoryStream.Create;
IFstream.LoadFromFile(IFile);
IFstream.Position:=IInfo.smeshenie;
//Збираємо масив з 1 і 0, що є розмір захованого файлу
for n:=31 downto 0 do
begin
IFstream.Read(z,sizeof(z));
if (z and 1)<>0 then z:=1 else z:=0;
b[n]:=z;
end;
sz:=0;
//Збираємо dword по бітах з масиву отриманого вище
for i:=31 downto 0 do
begin
bit:=b[i];
if (bit and (1 shl 7)) <> 0 then bit:=1;
sz:=sz or bit;
if i<>0 then sz:=sz shl 1;
end;
SetLength(DFile,0);
DFIle:='';
z:=0;
//Одержуємо з файлу-контейнера довжину імені
for n:=7 downto 0 do
begin
IFstream.Read(z,sizeof(z));
if (z and 1)<>0 then z:=1 else z:=0;
a[n]:=z;
end;
len:=0;
//Масив 1 і 0-0->байт
BinToHex(len,a);
//Одержуємо ім'я захованого файлу
for n:=1 to len do
```

```
begin
for i:=7 downto 0 do
begin
IFstream.Read(z,sizeof(z)); // Error!
if (z and 1)<>0 then z:=1 else z:=0;
a[i]:=z;
end;
BinToHex(z,a);
DFile:=Dfile+chr(z);
end;
//Дістаємо саме тіло файлу
for i:=0 to sz-1 do
begin
//Зчитуємо 8 біт
for n:=7 downto 0 do
begin
IFstream.Read(z,sizeof(z));
if (z and 1)<>0 then z:=1 else z:=0;
a[n]:=z;
end;
//Збираємо із цих біт байт
BinToHex(x,a);
//Дописуємо байт до тому що є в пам'яті - збираємо файл
DFStream.Write(x,sizeof(x));
end;
try
//Запропонуємо для збереження файлу ім'я, під яким він був упакований.
SaveDialog.FileName:=DFile;
//Викликаємо діалог збереження
if SaveDialog.Execute then DFile:=SaveDialog.FileName;
//Зберігаємо витягнутий файл
DFStream.SaveToFile(DFile);
except
MessageBox(0,PAnsiChar('Помилка при спробі збереження.'+#13#10+DFile),' X-
Date',MB_ICONERROR);
end;
IFstream.Free;
DFStream.Free;
Lst.Free;
MessageBox(0,'Done!',' X-Data',MB_ICONINFORMATION);
end;
```

Тепер кидаємо на форму чотири кнопки, один мемо, один OpenPictureDialog, один OpenDialog і один SaveDialog.

```
var
IInfo:TImageFileInfo;
DFile:string;
```

Будемо там зберігати інформацію про файл, у якому будемо писати й шлях до поточного файлу даних. Одну кнопку назвемо так: “Відкрити файл-зображення”. В оброблювачі onClick пишемо

```
procedure TForm1.Button1Click(Sender: TObject);
var i:byte;
Strgs:TStringList;
begin
if OpenPictureDialog1.Execute then
GetFileFormat(OpenPictureDialog1.FileName,Strgs,IInfo);
if OpenPictureDialog1.FileName<>'' then
begin
Memo1.Clear;
for i:=0 to Strgs.Count-1 do Memo1.Lines.Add(Strgs[i]);
end;
end;
```

Другу кнопку назвемо “Відкрити файл даних”. В оброблювачі пишемо.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
if OpenDialog1.Execute then
DFile:=OpenDialog1.FileName;
end;
```

Третю кнопку назвемо “Сховати файл.” В оброблювачі пишемо от що:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
if DFile<>'' then IHide(IInfo,Dfile,SaveDialog1)
else MessageBox(handle,'Не обраний файл даних!!!','XData
Error',MB_ICONERROR);
end;
```

А в оброблювачі четвертої, названої “Витягти файл”, напишемо тільки один рядок:

```
if OpenPictureDialog1.Execute then
IExtract(OpenPictureDialog1.FileName,SaveDialog1);
```

Тепер створюємо скрин з екрана, відкриваємо його, обираємо файл, що хочемо сховати й ховаємо.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою NTRU. NTRUEncrypt (абревіатура Nth-degree TRUncated polynomial ring або Number Theorists aRe Us) – це криптографічна система з відкритим ключем, що раніше називалася NTRU.

Криптосистема NTRUEncrypt, заснована на ґратчастій криптосистемі, створена як альтернатива RSA і криптосистемам на еліптичних кривих (ECC). Стійкість алгоритму забезпечується труднощами пошуку найкоротшого вектора ґрати, що більше стійка до атак, здійснюваним на квантових комп'ютерах. На відміну від своїх конкурентів RSA,ECC, Elgamal, алгоритм використовує операції над кільцем:

$$\mathbb{Z}[X]/(X^N - 1),$$

усічених багаточленів ступеня, що не перевершує $N - 1$:

$$\mathbf{a}(X) = \mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1}.$$

Такий багаточлен можна також представити вектором:

$$\vec{a}(X) = \vec{a} = \sum_{i=0}^{N-1} a_i X^i = [a_0, a_1, a_2, \cdots, a_{N-2}, a_{N-1}].$$

Як і будь-який молодий алгоритм, NTRUEncrypt погано вивчений, хоча й був офіційно затверджений для використання в сфері фінансів комітетом Accredited Standards Committee X9.[1]

Існує реалізації NTRUEncrypt з відкритим вихідним кодом.[2]

NTRUEncrypt, що споконвічно називався NTRU, був винайдений в 1996 році й представлений на конференціях CRYPTO, Конференція RSA, Eurocrypt. Причиною, що послужила початком розробки алгоритму в 1994 році, стала стаття [3], у якій говорилося про легкість злому існуючих алгоритмів на квантових комп'ютерах, які, як показало час, не за горами[4]. У цьому ж році, математики Jeffrey Hoffstein, Jill Pipher і Joseph H. Silverman, що розробили систему разом із засновником компанії NTRU Cryptosystems, Inc. (пізніше перейменованої в

SecurityInnovation), Даніелєм Ліеманом (Daniel Lieman) запатентували свій винахід.[5]

Кільця усічених багаточленів

NTRU оперує над багаточленами ступеня не переважаючої $N-1$:

$$\mathbf{a} = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-2} X^{N-2} + a_{N-1} X^{N-1},$$

де коефіцієнти $a_0, \cdots, a_{N-1}$ – цілі числа. Щодо операцій додавання й множення за модулем багаточлена $X^N - 1$.Такі багаточлени утворять кільце *R*, називане кільцем усічених багаточленів, що ізоморфно кільцю відносин:

$$\mathbb{Z}[X]/(X^N - 1).$$

NTRU використовує кільце усічених багаточленів *R* разом з діленням за модулем на взаємно прості числа *p* і *q* для зменшення коефіцієнтів багаточленів.

У роботі алгоритму також використовуються зворотні багаточлени в кільці усічених багаточленів. Слід зазначити, що не всякий багаточлен має зворотний, але якщо зворотний поліном існує, то його легко обчислити.[6][7]

Генерація відкритого ключа

Для передачі повідомлення від Аліси до Боба необхідні відкритий і закритий ключі. Відкритий знають як Боб, так і Аліса, закритий ключ знає тільки Боб, що він використовує для генерації відкритого ключа. Для цього Боб вибирає два «маленьких» поліноми *f g* з *R*. «Малість» поліномів мається на увазі в тому розумінні, що він маленький щодо довільного полінома за модулем *q*: у довільному поліномі коефіцієнти повинні бути приблизно рівномірно розподілені за модулем *q*, а в малому поліномі вони багато менше *q*. Малість поліномів визначається за допомогою чисел *df* і *dg*:

Поліном *f* має *df* коефіцієнтів рівних «1» і *df*-1коефіцієнтів рівних «-1», а інші – «0». У цьому випадку говорять, що:

$$\mathbf{f} \in \mathcal{L}_f$$

Поліном *g* має *dg* коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0» У цьому випадку говорять, що:

$$\mathbf{g} \in \mathcal{L}_g$$

Причина, по якій поліноми вибираються саме таким чином, полягає в тому, що f, можливо, буде мати зворотний, а g – однозначно немає (g (1) = 0, а нульовий елемент не має зворотного).

Боб повинен зберігати ці полінома в секреті. Далі Боб обчислює зворотні поліноми $\mathbf{f}_p$ й $\mathbf{f}_q$, тобто такі, що:

$$\mathbf{f} \cdot \mathbf{f}_p \equiv 1 \pmod{p} \text{ й } \mathbf{f} \cdot \mathbf{f}_q \equiv 1 \pmod{q}$$

Якщо f не має зворотного полінома, то Боб вибирає інший поліном f.

Секретний ключ – це пари $(\mathbf{f}, \mathbf{f}_p)$, а відкритий ключ h обчислюється за формулою:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod q.$$

Приклад:

Для приклада візьмемо df=4, а dg=3. Тоді як поліноми можна вибрати

$$\mathbf{f} = -1 + X + X^2 - X^4 + X^6 + X^9 - X^{10}$$
$$\mathbf{g} = -1 + X^2 + X^3 + X^5 - X^8 - X^{10}$$

Далі для полінома f шукаються зворотні поліноми за модулем p=3 иq=32:

$$\mathbf{f}_p = 1 + 2X + 2X^3 + 2X^4 + X^5 + 2X^7 + X^8 + 2X^9$$
$$\mathbf{f}_q = 5+9X+6X^2+16X^3+4X^4+15X^5+16X^6+22X^7+20X^8+18X^9+30X^{10}$$

Заключним етапом є обчислення самого відкритого ключа h:

$$\mathbf{h} = (p\mathbf{f}_q \cdot \mathbf{g}) \bmod 32 = 8+25X+22X^2+20X^3+12X^4+24X^5+15X^6+19X^7+12X^8+19X^9+16X^{10}.$$

Шифрування

Тепер, коли в Аліси є відкритий ключ, вона може відправити зашифроване повідомлення Бобові. Для цього потрібно повідомлення представити у вигляді полінома m з коефіцієнтами за модулем p, обраними з діапазону (-p/2, p/2]. Тобто m є «малим» поліномом за модулем q. Далі Алісі необхідно вибрати інший «малий» поліном r, що називається «сліпучої», обумовлений за допомогою числа dr:

Поліном r має dr коефіцієнтів рівних «1» і стільки ж рівних «-1», інші – «0». У цьому випадку говорять, що:

$$\mathbf{r} \in \mathcal{L}_r.$$

Використовуючи ці поліноми, зашифроване повідомлення виходить за формулою:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod q.$$

При цьому кожній, хто знає (або може обчислити) осліплюючий поліном r, зможе прочитати повідомлення *m*.

Приклад:

Припустимо, що Аліса хоче послати повідомлення, представлене у вигляді полінома:

$$\mathbf{m} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10},$$

і вибрала «осліплюючий» поліном, для якого *dr*=3:

$$\mathbf{r} = -1 + X^2 + X^3 + X^4 - X^5 - X^7.$$

Тоді шифротекст *e*, готовий для передачі Бобові буде:

$$\mathbf{e} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod 32 = 14 + 11X + 26X^2 + 24X^3 + 14X^4 + 16X^5 + 30X^6 + 7X^7 + 25X^8 + 6X^9 + 19X^{10}$$

Розшифрування

Тепер, одержавши зашифроване повідомлення *e*, Боб може його розшифрувати, використовуючи свій секретний ключ. Спочатку він одержує новий проміжний поліном:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q.$$

Якщо розписати шифротекст, то одержимо ланцюжок:

$$\mathbf{a} = (\mathbf{f} \cdot \mathbf{e}) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot \mathbf{h} + \mathbf{m})) \bmod q = (\mathbf{f} \cdot (\mathbf{r} \cdot p\mathbf{f}_q \cdot \mathbf{g} + \mathbf{m})) \bmod q$$

і остаточно:

$$\mathbf{a} = (p\mathbf{r} \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{m}) \bmod q.$$

Після того, як Боб обчислив поліном a за модулем *q*, він повинен вибрати його коефіцієнти з діапазону (-*q*/2, *q*/2] і далі обчислити поліном b, одержуваний з полінома a приведенням за модулем *p*:

$$\mathbf{b} = \mathbf{a} \bmod p = (\mathbf{f} \cdot \mathbf{m}) \bmod p,$$

так як:

$$(p\mathbf{r} \cdot \mathbf{g}) \bmod p = 0.$$

Тепер, використовуючи другу половину секретного ключа й отриманий поліном *b*, Боб може розшифрувати повідомлення:

$$\mathbf{c} = (\mathbf{f}_p \cdot \mathbf{b}) \bmod p.$$

Неважко бачити, що:

$$\mathbf{c} \equiv \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \equiv \mathbf{m} \pmod{p}.$$

У такий спосіб отриманий поліном *c* дійсно є вихідним повідомленням *m*.

Приклад:

Боб одержав від Аліси шифроване повідомлення *e*:

$$\mathbf{e} = 14+11X+26X^2+24X^3+14X^4+16X^5+30X^6+7X^7+25X^8+6X^9+19X^{10}$$

Використовуючи секретний ключ *f* Боб одержує поліном *a*:

$$\mathbf{a} = \mathbf{f}\cdot\mathbf{e} \pmod{32} = 3-7X-10X^2-11X^3+10X^4+7X^5+6X^6+7X^7+5X^8-3X^9-7X^{10} \pmod{32},$$

з коефіцієнтами, що належать проміжку (-*q*/2, *q*/2]. Далі перетворить поліном *a* у поліном *b*, зменшуючи коефіцієнти за модулем *p*.

$$\mathbf{b} = \mathbf{a} \pmod 3 = -X - X^2 + X^3 + X^4 + X^5 + X^7 - X^8 - X^{10} \pmod 3$$

Заключний крок – перемножування полінома *b* із другою половиною закритого ключа $\mathbf{f}_p$:

$$\mathbf{c} = \mathbf{f}_p \cdot \mathbf{b} = \mathbf{f}_p \cdot \mathbf{f} \cdot \mathbf{m} \pmod 3 = \mathbf{m} \pmod 3$$

$$\mathbf{c} = -1 + X^3 - X^4 - X^8 + X^9 + X^{10}$$

Який є вихідним повідомленням, що передавала Аліса.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З нього видно, що інтерфейс користувача складається з наступних блоків:

– Блок меню.

– Блок вибору параметрів файлу та стегоключа.

– Блок кнопок швидкого доступу до функцій системи.

– Вікно виведення зображення контейнера.

– Вікно виведення тексту стегоповідомлення.

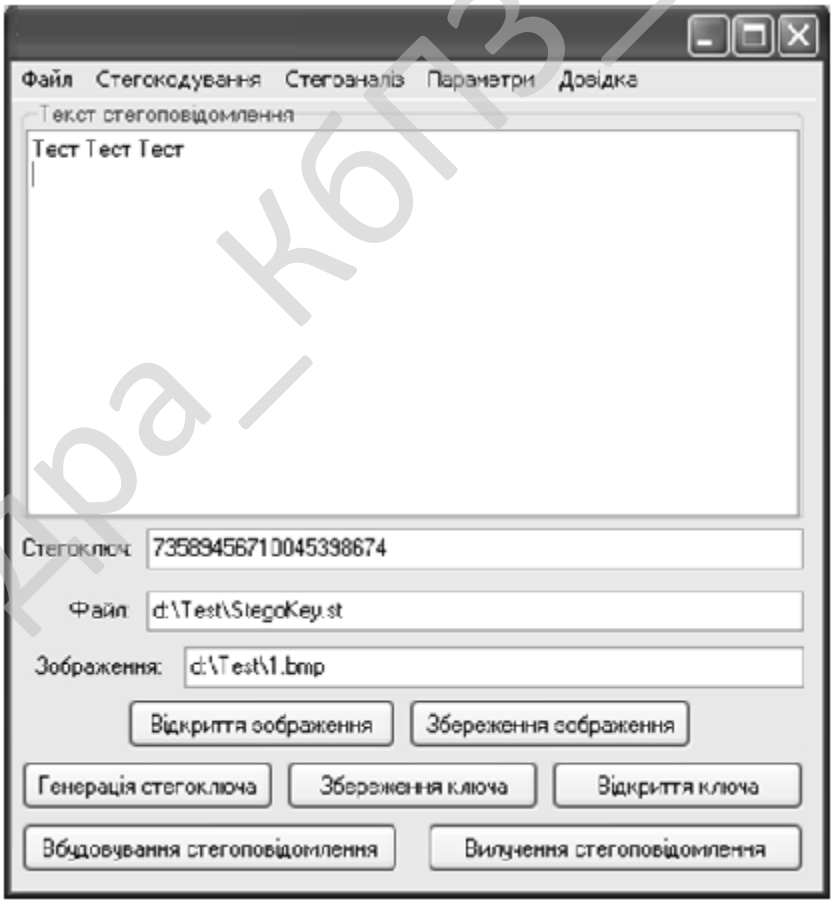

Рисунок 5.1 – Головне вікно програми

Блок меню складається з наступних елементів:

– Файл.

– Стегокодування.

– Стегоаналіз.

– Довідка.

Блок кнопок швидкого доступу до функцій системи складається з наступни кнопок:

– Відкрити зображення.

– Збереження зображення.

– Аналіз на наявність стего.

– Генерація стегоключа.

– Збереження ключа.

– Відкриття ключа.

– Вбудовування стегоповідомлення.

– Вилучення стегоповідомлення.

На рисунку 5.2 зображено вікно довідки, у якій вказано, хто виконував бакалаврську роботу, яка тема бакалаврської роботи та хто керівник бакалаврської роботи.

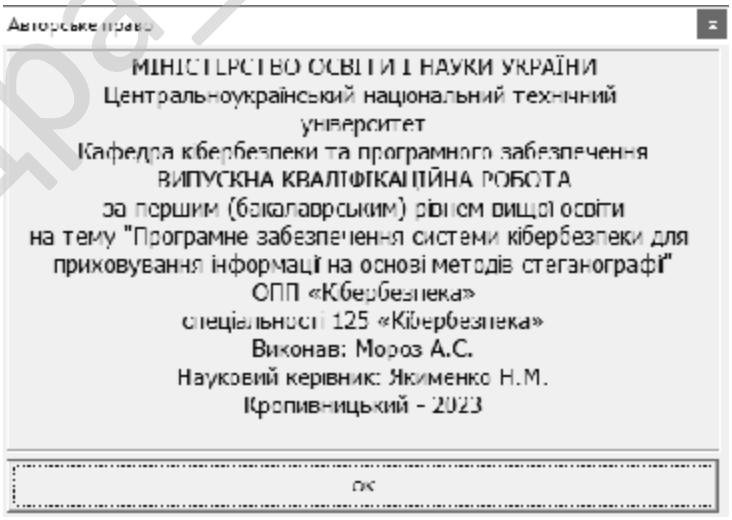

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для приховування інформації на основі методів стеганографії.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для приховування інформації на основі методів стеганографії.

– Досліджена система для приховування інформації на основі методів стеганографії.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для приховування інформації на основі методів стеганографії.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для приховування інформації на основі методів стеганографії.

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 Delphi 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для приховування інформації на основі методів

стеганографії. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм NTRU.

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. *Смирнов А.А.* Ансамблевые свойства недвоичных дискретных сигналов / А.А. Кузнецов, А.А. Смирнов, А.М. Носик, Л.Н. Качур, В.Н. Сай // Системи управління, навігації та зв'язку. – Випуск 4 (8). – К.: ДП «ЦНДІНУ». – 2008. – С. 175-177.

2. *Смирнов А.А.* Разработка метода и алгоритмов синтеза больших ансамблей недвоичных дискретных сигналов на основе обобщенных перестановочных преобразований / А.А. Кузнецов, Ал.М. Носик, А.А. Смирнов, Л.Н. Качур, Ан.М. Носик // Збірник наукових праць «Системи обробки інформації». – Випуск 5(72). – Х.: ХУПС – 2008. – С. 151-156.

3. *Смирнов А.А.* Формирование дискретных сигналов с многоуровневой функцией корреляции / А.А. Кузнецов, А.А. Смирнов, В.Н. Сай // Збірник наукових праць «Системи обробки інформації». – Випуск 5 (95). – Х.: ХУПС. – 2011. – С. 50-60.

4. *Смирнов А.А.* Дискретные сигналы с многоуровневой функцией корреляции / А.А. Кузнецов, А.А. Смирнов, В.Н. Сай // Радиотехника: Всеукраинский межведомственный научно-технический сборник. Тематический выпуск «Информационная безопасность» – Выпуск 166. – Х.: ХНУРЭ. – 2011. – С. 142-152.

5. *Смирнов А.А.* Математическая модель и структурная схема стеганографической системы / А.А. Кузнецов, А.А. Смирнов, Е.В. Мелешко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Випуск 25. Частина 1. – Кіровоград: КНТУ. – 2012. – С. 273-281.

6. *Смірнов О.А.* Стеганографічне приховування інформації із використанням прямого розширення спектру / О.О. Кузнецов,

О.А. Смірнов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 49-52.

7. *Смирнов А.А.* Математическая модель и структурная схема стеганографической системы / А.А. Кузнецов, А.А. Смирнов, Е.В. Мелешко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 91-92.

8. *Смирнов А.А.* Встраивание данных в контейнеры-изображения с использованием сложных дискретных сигналов / А.А. Кузнецов, А.А. Смирнов // Радиотехника: Всеукраинский межведомственный научно-технический сборник. Тематический выпуск «Информационная безопасность» – Выпуск 166. – Х.: ХНУРЭ. – 2011. – С. 134-141.

9. *Смірнов О.А.* Дослідження ймовірнісних властивостей стеганографічного захисту інформації із використанням прямого розширення спектру / О.О. Кузнецов, О.А. Смірнов, Л.Т. Пархуць, Ю.М. Рябуха // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 1. – К.: ДП «ЦНДІНУ». – 2012. – С. 115-121.

10. *Смирнов А.А.* Встраивание данных в контейнеры-изображения с использованием сложных дискретных сигналов / А.А. Кузнєцов, А.А. Смирнов // Збірник тез XI міжнародної науково-технічної конференції «Проблеми інформатики и моделювання». м. Харків. 25 листопада 2011 р. – Харків: НАНУ, НТУ «ХПІ», РВНЗ «КГУ». – 2011. – С.42.

11. *Smirnov O.A.* Use of Complex Discrete Signals for Steganographic Information Security / A.A. Kuznetsov, A.A. Smirnov // International Journal of Engineering Practical Education. – Volume 1, Issue 1. – USA, Indiana, Riley: Science and Engineering Publishing Company. – 2012. – P. 21-25.

12. *Кущ А.В.* Использование алгоритмов стеганографии при проведении

компьютерно-технической экспертизы / А.В. Кущ // VI Всероссийская межвузовская конференция молодых ученых – СПб: СПбГУ ИТМО, 2009.

13. *Ленков С.В.* Методы и средства защиты информации [Текст]: в 2 т / Ленков С.В., Перегудов Д.А., Хорошко В.А. – К.: Арий, 2008. – Т.2: Информационная безопасность. – 2008. – 344 с.

14. *Лукічов В.В.* Методи та засоби стеганографічного захисту інформації в комп'ютерних системах і мережах на основі вейвлет-перетворень: автореф. дис. канд. техн. наук : 05.13.21 «Системи захисту інформації» / В.В. Лукічов – К.: Нац. авіац. ун-т., 2010. – 20 с.

15. *Мак-Вильямс Ф.Дж.* Теория кодов, исправляющих ошибки / Мак-Вильямс Ф.Дж., Слоэн Н.Дж.А. – М.: Связь, 1979. – 744 с.

16. *Митекин В.А.* Модифицированные методы статистического стегоанализа бинарных и полутоновых изображений / В.А. Митекин // Компьютерная оптика № 28 – Институт систем обработки изображений РАН: 2005 – 145-151с.

17. *Михайличенко О.В*. Применение стеганографических методов сокрытия информации в неподвижных изображениях / О.В. Михайличенко, А.Г. Коробейников, С.Ю. Каменева // Труды международных научно-технических конференций «Интеллектуальные системы» (IEEE AIS'06) и «Интеллектуальные САПР (CAD-2006) «: в 3 т. М.: Физмалит, 2006. Т.2. – С. 511-515.

18. *Михайличенко О.В.* Повышение устойчивости стеганоалгоритмов частотной области на основе дискретно-косинусного преобразования к внешним воздействиям / О.В. Михайличенко, Н.Н. Прохожев, А.Г Коробейников // Научно-технический вестник СПб ГУ ИТМО – СПб.: СПб ГУ ИТМО, 2009.– вып. 2(60). – С.102 – 104.

19. *Михайличенко О.В.* Алгоритм встраивания цифровых водяных знаков в единичный коэффициент матрицы дискретно-косинусного преобразования / О.В. Михайличенко, Н.Н. Прохожев // Сборник трудов VI Всероссийской конференции молодых ученых. Выпуск 6. Информационные технологии. – СПб. : СПбГУ

ИТМО, 2009. – С. 644-648.

20. Надёжность и эффективность в технике. Справочник (в 10 томах). Том 1. Методология. Организация. Терминология / [В.С. Авдуевскнй, И.В. Апполонов, Е.Ю. Барзилович и др.]; под ред. А.И. Рембезы – М.: Машиностроение 1986. – 220 с.

21. Надёжность и эффективность в технике. Справочник (в 10 томах). Том 2. Математические методы в теории надежности и эффективности / [В.В. Белов, Ю.К. Беляев, А.Г. Давтян и др.]; под ред. Б.В.Гнеденко – М.: Машиностроение 1987. – 281 с.

22. Надёжность и эффективность в технике. Справочник (в 10 томах). Том 3. Эффективность технических систем / [В.У. Торбин, Г.Н. Охотников, Е.С. Егоров и др.]; под ред. В.Ф. Уткина и Ю.В. Крючкова – М.: Машиностроение 1988. – 327 с.

23. *Науменко М.І.* Теоретичні основи та методи побудови алгебраїчних блокових кодів: монографія. / М.І. Науменко, Ю.В. Стасєв, О.О. Кузнецов – Х.: ХУ ПС, 2005. – 267 с.

24. *Науменко Н.І.* Теорія сигнально-кодових конструкцій / Н.І. Науменко, Ю.В. Стасев, О.О. Кузнецов, С.П. Євсєєв – Х.:ХУ ПС, 2008р. – 489 С.

25. НД ТЗІ 2.5-004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999 р., №22.

26. *Нікітенко Л.Л.* Методи побудови стійких стеганосистем [Текст] : автореф. дис. на здоб. наук. ступеня канд. фіз.-мат.наук : [спец.] 01.05.01 «Теор. основи інформатики та кібернетики» / Л.Л. Нікітенко – К. : НАН Укр. Ін-т кібернетики ім. В.М. Глушкова, 2010. – 19 с.

27. *Нікітіна О.Ю.* Комп'ютерні технології побудови систем цифрових водяних знаків [Текст] : автореферат канд. техн. наук, спец.: 01.05.01 – теоретичні основи інформатики та кібернетики / О. Ю. Нікітіна. – К. : НАН Укр. Ін-т кібернетики ім. В.М. Глушкова, 2011. – 19 с.

28. *Носик А.М.* Разработка метода формирования недвоичных псевдослучайных последовательностей для построения дискретных сигналов / А.М. Носик, А.А. Смирнов, Л.Н. Качур // Матеріали першої науково-технічної конференції «МНС України: Сучасний стан та проблемні питання страхового фонду документації, перспективи розвитку та взаємодії». Харків. 25-26 квітня 2008 р. – Харків. НДПКТІМ. – 2008. – С. 46-47.

29. *Паламарчук С.А.* Стеганографічні методи приховування інформації в зображеннях / С.А. Паламарчук, І.В. Бабич // Бизнес и безопасность. – К., 2011. – Вип. 3. – С. 35 – 37.

30. *Паламарчук С.А*. Алгоритм реалізації стеганографічного перетворення інформації в зображеннях / С.А. Паламарчук, І.В. Бабич // VI НПС «Пріоритетні напрямки розвитку ТКС та мереж спеціального призначення» 20- 21.10.2011 р.: Доповіді та тези доповідей. К.: ВІТІ НТУУ «КПІ», 2011. – С. 160.

31. *Пышкин И.М.* Теория кодового разделения сигналов. – М.: Связь, 1980. – 208 с.

32. *Пометун С.О.* Алгебраїчні атаки на потокові шифратори як узагальнення кореляційних атак / С.О. Пометун // Системні дослідження та інформаційні технології. – 2008. – №2. – С.29-40.

33. *Прокис Дж.* Цифровая связь. Пер. с англ./Под ред. Д. Д. Кловского. М.: Радио и связь, 2000. – 800 с.

34. *Пузыренко А.Ю.* Стеганографическая защита информации с использованием потокового аудио-контейнера / В. П. Бабак, Г. Ф. Конахович, А. Ю. Пузыренко // Безопасность информации в ИТКС-2005 : VIII міжнар. наук.-практ. конф., 11–13 травня 2005 р. : тези доп. – К. : НИЦ ―Тезис, 2005. – С. 11.

35. *Пузыренко А.Ю.* Оценка качества реализации стеганографических алгоритмов / В. П. Бабак, Г. Ф. Конахович, А. Ю. Пузыренко // Безопасность информации в ИТКС-2006: IX міжнар. наук.-практ. конф., 17–19 травня 2006 р. : тези доп. – К. : НИЦ ―Тезис, 2006. – С. 18.

36. *Пузиренко О.Ю.* Представлення і прогнозування ефективності нового протоколу оцінки якості реалізації розроблюваних алгоритмів комп'ютерної стеганографії / О. Ю. Пузиренко, Д. О. Навроцький, Л. П. Дюжаєв // Радіотехніка. Радіоапаратобудування : Зб. наук. пр. – Вип. 34. – К. : НТУУ «КПІ», 2007. – С. 150–156.

37. *Садов В.С.* Обнаружение стеганографического канала передачи данных путем анализа однобитного шума изображения / В.С. Садов, И.Л. Чваркова // Известия Белорусской инженерной академии, № 1 (19)/2, 2005, с. 75-78.

38. *Садов В.С.* Оценка информационных потерь при фильтрации изображений./ В.С. Садов, С.Г. Тихоненко, А.Ф. Чернявский // Информатика. – 2005. – № 3(7). – с. 52-59.

39. *Свердлик М.Б.* Оптимальные дискретные сигналы. – М.: Сов. Радио. – 1975, 200с.

40. *Скляр Б.* Цифровая связь. Теоретические основы и практическое применение. – М.: Вильямс, 2003. – 1104 с.

41. *Сифоров В.И.* Радиоприемные устройства / В.И. Сифоров, И.Н. Амиантов, Ю.Н. Антонов-Антипов, В.П. Васильев, Б.В. Данилов, В.Л. Лебедев, С.С. Судаков, К.А. Щуцкой. – М.:«Советское радио» 1974г.–560с.

42. *Смирнов А.А.* Методы и средства компьютерной стеганографии с применением сложных дискретных сигналов для защиты информации в компьютерных системах и сетях: монография / А.А. Смирнов – К.: Изд. «КОД» – 2012. – 350 с.

43. *Смирнов А.А.* Критерии и показатели эффективности стеганографических систем защиты информации / А.А. Смирнов // Радиотехника: Всеукраинский межведомственный научно-технический сборник. Тематический выпуск «Информационная безопасность» – Выпуск 171. – Х.: ХНУРЭ. – 2012. – С. 189-197.

44. *Smirnov A.A.* Block diagram and formal mathematical definition of

steganographic system / A.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 2, Issue 8. – India. Delhi. – 2012. – P. 90-95.

45. *Смирнов А.А*. Математическая модель и структурная схема стеганографического преобразования информации на основе прямого расширения спектра / А.А. Смирнов // Збірник тез доповідей III міжнародної науково-технічної конференції «Інформаційні технології та захист інформації». м. Харків. 20-21 квітня 2012 р. – Харків: ХНЕУ. – 2012. – С. 211.

46. *Смирнов А.А.* Математическая формализация процедуры стеганографического кодирования и декодирования / А.А. Смирнов // Збірник тез V міжнародної науково-практичної конференції «Інтегровані інтелектуальні робототехнічні комплекси» (ІПРТК-2012). м. Київ. 15-16 травня 2012 р. – К.: НАУ. – 2012. – С. 358-360.

47. *Смірнов О.А.* Аналіз та дослідження стеганографічних систем та протоколів для захисту інформації та інформаційних ресурсів / О.А. Смірнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». м. Вінниця. 29-31 травня 2012 р. – Вінниця: ВНТУ. – 2012. – С. 164-166.

48. *Смирнов А.А.* Построение стеганографических каналов передачи данных с высокой пропускной способностью в компьютерных сетях / А.А. Смирнов // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

49. *Смірнов О.А.* Дослідження стеганографічного перетворення інформаційних повідомлень для організації скритних каналів передачі даних / О.А. Смірнов // Збірник наукових праць «Системи обробки інформації». – Випуск 2(100). – Х.: ХУПС – 2012. – С. 219-222.

50. *Смірнов О.А.* Дослідження методів стеганографічного перетворення інформації / О.А. Смірнов // Матеріали XII Всеукраїнської наукової інтернет-

конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 35-36.

51. *Смірнов О.А.* Дослідження стеганографічного перетворення інформаційних повідомлень для організації скритних каналів передачі даних / О.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції «Проблеми інформатики та комп'ютерної техніки» (ПІКТ-2012). м. Чернівці. 3-5 травня 2012 р. – Чернівці: ЧНУ. – 2012. – С. 119-120.

52. *Смірнов О.А.* Теоретичні основи вбудовування інформації в нерухливі зображення з використанням складних дискретних сигналів / О.А. Смірнов // Збірник тез IV міжнародної науково-практичної конференції «Проблеми і перспективи розвитку ІТ-індустрії». м. Харків. 15-16 листопада 2012 р. – Харків: ХНЕУ. – 2012. – С. 230-231.

53. *Смирнов А.А.* Стеганографическое встраивание данных в неподвижные изображения методом прямого расширения спектра / А.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(6). – Х.: ХУПС. – 2011. – С.126-129.

54. *Смирнов А.А.* Метод стеганографического встраивания информации в неподвижные изображения с использованием сложных дискретных сигналов и прямого расширения спектра / А.А. Смирнов // Науково-технічний журнал «Захист інформації». – Випуск 4 (53). – К.: НАУ. – 2011 – С.64-70.

55. *Смірнов О.А.* Стеганографічний захист інформації із використанням прямого розширення спектру / О.А. Смірнов // Збірник наукових праць Харківського університету Повітряних Сил. – Випуск 1 (30). – Х.: ХУПС. – 2012. – С. 108-112.

56. *Смірнов О.А.* Метод стеганографічного приховування та вилучення даних в просторовій області зображень із використанням прямого розширення спектру / О.А. Смірнов // Збірник наукових праць «Системи обробки інформації». – Випуск 3(101) том 1. – Х.: ХУПС – 2012. – С. 56-61.

57. *Смирнов А.А.* Стеганографическое встраивание данных в неподвижные изображения методом прямого расширения спектра / А.А. Смирнов // Збірник тез II міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-коммунікаційних технологій та засобів управління». м. Київ-м. Харків. 15-16 грудня 2011 р. – Київ: ДП «ЦНДІ НіУ», Харків: ДП «ХНДІ ТМ», Київ: КДАВТ. – 2011. – С.70-71.

58. *Смирнов А.А.* Метод стеганографического встраивания информации в неподвижные изображения с использованием сложных дискретных сигналов и прямого расширения спектра / А.А. Смирнов // Матеріали III міжнародної науково-практичної конференції «Системний аналіз. Інформатика. Управління (САІУ-2012)». м. Запоріжжя. 14-16 березня 2012 р. – Запоріжжя: КПУ. – 2012. С. 264-266.

59. *Смірнов О.А*. Стеганографічне приховування даних в аудіо-контейнерах із використанням прямого розширення спектру / О.А. Смірнов // Збірник тез XIX науково-практичної конференції «Проблеми створення, розвитку та застосування інформаційних систем спеціального призначення». м. Житомир. 19 квітня 2012 р. – Житомир: ЖВІ НАУ. – 2012. – С. 147-148.

60. *Смирнов А.А.* Встраивание данных в частотную область неподвижных изображений с использованием технологии прямого расширения спектра / А.А. Смирнов // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». м. Харків. 18-19 квітня 2012 р. – Харків. ХУПС. – 2012. – С. 174-175.

61. *Смірнов О.А.* Стеганографічне приховування повідомлень в просторовій області зображень із використанням прямого розширення спектру / О.А. Смірнов // Збірник тез II науково-технічної конференції «Безпека інформаційних технологій» «Information Technology Security» (ITSEC-2012). м. Київ. 24-25 квітня 2012 р. – Київ: НАУ. – 2012. С. 22.

62. *Смірнов О.А.* Дослідження ймовірностних характеристик стеганографічного захисту інформації із використанням прямого розширення спектру / О.А. Смірнов // Збірник наукових праць науково-технічної конференції з міжнародною участю «Комп'ютерне моделювання у наукоємних технологіях» (КМНТ-2012). м. Харків. 24-27 квітня 2012 р. – Харків: ХНУ. – 2012. – С. 400-401.

63. *Смирнов А.А.* Методы широкополосной связи в стеганографии / А.А. Смирнов // Збірник тез V міжнародної науково-практичної конференції «Інформаційна та економічна безпека (INFECO-2012)». м. Харків. 24-26 квітня 2012 р. – Харків: ХНЕУ. – 2012. – С. 135-137.

64. *Смірнов О.А.* Технологія прямого розширення спектру в стеганографії / О.А. Смірнов // Збірник тез першої міжнародної науково-технічної конференції «Захист інформації і безпека інформаційних систем». м. Львів. 31 травня – 01 червня2012 р. – Львів: НУ «ЛП». – 2012. С. 122-123.

65. *Смирнов А.А.* Исследование известных методов синтеза дискретных сигналов / А.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 3(9). – Х.: ХУПС. – 2012. – С. 123-126.

66. *Смирнов А.А.* Исследование методов синтеза дискретных сигналов с особыми корреляционными свойствами / А.А. Смирнов // Збірник наукових праць «Системи обробки інформації». – Випуск 9(107). – Х.: ХУПС – 2012. – С. 81-85.

67. *Смирнов А.А.* Сравнительные исследования методов синтеза дискретных сигналов с особыми корреляционными свойствами / А.А. Смирнов, Е.В. Мелешко // Збірник тез V міжнародного науково-технічного симпозіуму «Новітні технології в телекомунікаціях» (ДУІКТ-Карпати-2012) м. Київ. 17-21 січня 2012 р. – Київ: ДУІКТ. – 2012. – С. 80-81.

68. *Smirnov A.A.* Analysis and comparative study of synthesis methods of digital signals with special correlation properties / A.A. Smirnov // International Journal on Communications (IJC). – Volume 1, Issue 1. – USA, Indiana, Riley: Science and Engineering Publishing Company. – 2012. – P. 12-20.

69. *Смирнов А.А.* Формирование больших ансамблей дискретных сигналов

на основе методов алгебраического кодирования / А.А. Смирнов, Л.Н. Качур, А.Н. Коваленко // Збірник наукових статей «Управління розвитком» за результатами міжнародної науково-практичної конференції «Стратегії ІТ-технологій в освіті економіці та екології». Випуск 7. – Харків: ХНЕУ. – 2007. – С. 70-71.

70. *Смирнов А.А.* Формирование больших ансамблей дискретных сигналов с использованием избыточных кодов / А.А. Смирнов, Л.Н. Качур // Збірка тез доповідей першої всеукраїнської науково-практичної конференції «Перспективи розвитку озброєння і військової техніки в Збройних силах України». Львів. 04-05 березня 2008 р. Львів. ЛІСВ. – 2008. – С. 219-220.

71. *Смирнов А.А.* Синтез больших ансамблей дискретных сигналов с использованием недвоичных избыточных кодов / А.А. Смирнов, А.М. Носик, Л.Н. Качур, С.Ю. Стасев // Матеріали IV наукової конференції Харківського Університету Повітряних Сил імені Івана Кожедуба. Харків. 16-17 квітня 2008 р.– Харків. ХУПС. – 2008. – С. 154.

72. *Смирнов А.А.* Дискретні сигнали з багаторівневою функцією кореляції / А.А. Смирнов // Збірник тез XII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 14-15 жовтня 2011 р. – Кіровоград: КНТУ. – 2011. – С. 119-120.

73. *Смирнов А.А.* Формирование дискретных сигналов с многоуровневой функцией корреляции / А.А. Смирнов, А.А. Кузнецов, В.Н. Сай // Збірник тез IX міжнародної науково-практичної конференції «Математичне та програмне забезпечення інтелектуальних систем» м. Дніпропетровськ. 23-25 листопада 2011р. – Дніпропетровськ: ДНУ. – 2011. – С.247-248.

74. *Смірнов О.А.* Синтез ансамблів дискретних сигналів для стеганографічних систем з прямим розширенням спектру / О.А. Смірнов // Збірник тез XX міжнародної науково-практичної конференції «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я» (MicroCAD-2012). м. Харків. 15-17 травня 2012 р. – Харків: НТУ «ХПІ». – 2012. – С. 45-46.

75. *Смірнов О.А.* Дослідження ансамблевих, кореляційних і структурних властивостей складних дискретних сигналів, які використовуються для побудови стеганографічних систем / О.А. Смірнов, Є.В. Мелешко // Збірник тез IV науково-технічної конференції «Захист інформації з обмеженим доступом та автоматизація її обробки» (PIRAT-2012). м. Київ. 9-10 лютого 2012 р. – Київ: НАУ. – 2012. – С. 32-33.

76. *Смирнов А.А.* Предложения по реализации устройств формирования дискретных сигналов с многоуровневой функцией корреляции / А.А. Смирнов // Науково-технічний журнал «Захист інформації». – Випуск 4 (57). – К.: НАУ. – 2012 – С.94-105.

77. *Smirnov A.A.* The hardware implementation of devices forming discrete signals with multi-level correlation function / A.A. Smirnov // International Journal of Information and Computer Science (IJICS). – Volume 2, Issue 1. – USA, Indiana, Riley: Science and Engineering Publishing Company. – 2013. – P. 1-7.

78. *Смирнов А.А.* Обоснование предложений по реализации динамического режима функционирования радиосистем управления с множественным доступом / А.А. Смирнов, В.Н. Сай, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3(23). – К.: ДП «ЦНДІНУ». – 2012. – С. 255-262.

79. *Смирнов А.А.* Анализ перспективных направлений в совершенствовании радиосистем управления и связи с организацией множественного доступа / А.А. Смирнов, В.Н. Сай, А.В. Коваленко // Системи озброєння і військова техніка. – Випуск 3(31) – Х.: ХУПС – 2012. – С. 218-226.

80. *Смирнов А.А.* Обоснование критериев и показателей выбора ансамблей дискретных сигналов для радиосистем управления с множественным доступом / А.А. Смирнов // Системи озброєння і військова техніка. – Випуск 4(32) – Х.: ХУПС – 2012. – С. 158-161.

81. *Смирнов А.А.* Исследование абонентской емкости и помехоустойчивости радиоканалов управления с использованием дискретных сигналов с многоуровневой функцией корреляции / А.А. Смирнов // Наука і

техніка Повітряних Сил Збройних Сил України. – Випуск 1(10). – Х.: ХУПС . – 2013. – С. 111-115.

82. *Смірнов О.А.* Дослідження методів стегоаналізу цифрових зображень / О.А. Смірнов, Є.В. Мелешко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 75-77.

83. *Смірнов О.А.* Дослідження методів стегоаналізу цифрових зображень / О.А. Смірнов, Є.В. Мелешко // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(8). – Х.: ХУПС. – 2012. – С. 92-99.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для приховування інформації на основі методів стеганографії.

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. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії.

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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для приховування інформації на основі методів стеганографії;

– 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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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 Delphi 10.4.1.

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 – 92 аркуші.

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.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2023 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

_____ Якименко Н.М.

Програмне забезпечення системи кібербезпеки для приховування інформації на основі методів стеганографії

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 28

Літера: РП

Кропивницький – 2023 року

STEGOGRAPHY.PAS – реалізація алгоритму стеганографії

```
unit Stegography;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, password__Stegography, Menus, ExtCtrls;

type
  buffer_ = array [1..1024*1024*2] of byte;
  TBuffer_ = ^buffer_;
  fhandle  = record
               name  : shortstring;
               size  : integer;
               next  : byte;
               data_ : TBuffer_;
             end;

  TForm2 = class(TForm)
    ListBox1: TListBox;
    Label1: TLabel;
    ListBox2: TListBox;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button1: TButton;
    Button2: TButton;
    Label5: TLabel;
    Label6: TLabel;
    PopupMenu1: TPopupMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    ldfl: TOpenDialog;
    svfl: TSaveDialog;
    extr: TOpenDialog;
    Panel1: TPanel;
    N5: TMenuItem;
    N6: TMenuItem;
    procedure loaddir;
    procedure FormCreate(Sender: TObject);
    procedure PopupMenu1Popup(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure Panel1DblClick(Sender: TObject);
    procedure N5Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
  private
    { Private declarations }
  public
     bmp : TBitMap;
    { Public declarations }
  end;

var
  Form2: TForm2;
  estlen,    // Залишилось вільного місця
  filelen,
  datalen,
  maxlen,
```

```
  maxcol : integer;

function  readdata(dat_ : TBuffer_;len : integer): integer;
function  Writedata(dat_ : TBuffer_;len : integer): integer;
function checkbmp: integer;

implementation

uses Hamming;

{$R *.DFM}
var
   fcnt,ft2cnt,present,
   totpoz,    // Номер наступного байту при читанні чи записі (для процедур
ReadData та WriteData)
   curline,   // Поточний графічний рядок
   curcol : integer;  // Номер байта в графічному рядку куди буде
запісано/прочитано наступний байт
   FAT_ : array [1..100] of Fhandle;
   ft2  : array [1..100] of Fhandle;  // Масив всього, що потрібно занести у
зображення
// Процедура кодування даних
procedure code(dat_ : TBuffer_; len,totpos : integer);
{ Принцип кодування наступний:
   1: Обчислюється контрольна сума пароля
   2: Обчислюється контрольний добуток пароля
   3: Всі закодовані дані представляються як масив байтів
   4: Від кожного байта даних віднімається байт контрольної суми пароля
   5: З результатом попереднього обчислення робиться XOR з байтом контрольного
добутку пароля
   6: До резултату попереднього обчислення додається код відповідного символу
з рядка пароля.
      як тільки рядок пароля закінчується знову переходимо на його початок
}
var
   cdcnt,m,d,x,sm : integer;
begin
   if passwrd.password='' then exit; // якщо пароль не введено, то вихід
   sm:=0;
   for x:=1 to length(passwrd.password) do sm:=sm+ord(passwrd.password[x]); //
сума символів пароля
   m:=1;
   for x:=1 to length(passwrd.password) do    // добуток символів пароля
   begin
      m:=m*ord(passwrd.password[x]);
      while ((m and 1) <> 1) do m:= m shr 1;   // Видалення молодших нулів
      while (m > (256*256*128)) do m:= m shr 1; // щоб не було переповнення
   end;

   cdcnt:=totpos mod length(passwrd.password);
   for x:=1 to len do
   begin
      d:=dat_^[x];
      d:=((d+2048- sm) xor m) + ord (passwrd.password[cdcnt]) ;
      inc (cdcnt);
      if cdcnt>length(passwrd.password) then cdcnt:=length(passwrd.password);
      dat_^[x]:=byte(d);
   end;
end;

// розкодування даних
procedure decode(dat_ : TBuffer_; len,totpos : integer);
var
   cdcnt,m,d,x,sm : integer;
begin
   if passwrd.password='' then exit;
   sm:=0;
   for x:=1 to length(passwrd.password) do sm:=sm+ord(passwrd.password[x]);
   m:=1;
```

```
   for x:=1 to length(passwrd.password) do
   begin
      m:=m*ord(passwrd.password[x]);
      while ((m and 1) <> 1) do m:= m shr 1;
      while (m > (256*256*128)) do m:= m shr 1;
   end;

   cdcnt:=totpos mod length(passwrd.password);
   for x:=1 to len do
   begin
      d:=dat_^[x];
      d:=((d- ord(passwrd.password[cdcnt])) xor m )+ sm;
      inc (cdcnt);
      if cdcnt>length(passwrd.password) then cdcnt:=length(passwrd.password);
      dat_^[x]:=byte(d);
   end;
end;

procedure seekbmp(poz : integer);   // Встановлення вказівника для
читання/запису стегоданих із зображення

begin
   totpoz:=poz;
   poz:=(poz-1)*8;
   curcol:=(poz mod (form2.bmp.width*3))+1;  // номер байта в графічному рядку
   curline:=poz div (form2.bmp.width*3);     //номер графічного рядка, в якому
знаходиться необхідна позиція
end;

// Dat_ Вказівник на буфер для читання
function  readdata(dat_ : TBuffer_;len : integer): integer;
// LEN - довжина даних
var
   x,y     : integer;
   pt      : TBuffer_;
   dat     : integer;
begin
   pt:=form2.bmp.ScanLine[curline];  // вказівник на потрібний графічний рядок
   for y:=1 to len do
   begin
      for x:=1 to 8 do              // від біта 0 до биту 7 кожного рядка
даних
      begin
         if curcol > maxcol then    // перевірка чи не вийшов номер байта в
графічному рядку ща межу
         begin
            inc (curline);
            pt:=form2.bmp.ScanLine[curline];  // вказівник на наступний рядок
            curcol:=1;
         end;
         // вбудовування інформації
         if ((pt^[curcol] and 1) <> 0) then dat:=dat or (1 shl (x-1)) else
dat:=dat and (not((1 shl (x-1))));
         inc (curcol);
      end;
      dat_^[y]:=byte(dat);
   end;
   decode(dat_,len,totpoz);
   totpoz:=totpoz+len;
end;

function  Writedata(dat_ : TBuffer_;len : integer): integer;
var
   x,y     : integer;
   pt      : TBuffer_;
   d       : integer;
begin
   code(dat_,len,totpoz);  // кодування інформації, що вбудовується
   pt:=form2.bmp.ScanLine[curline];
```

```
   for y:=1 to len do
   begin
      d:=dat_^[y];
      for x:=1 to 8 do
      begin
         if curcol > maxcol then
         begin
            inc (curline);
            pt:=form2.bmp.ScanLine[curline];
            curcol:=1;
         end;
         if ((d and 1) <> 0 ) then pt^[curcol]:=(pt^[curcol] or 1) else
pt^[curcol]:=(pt^[curcol] and $FE);
         d:= d shr 1;
         inc (curcol);
      end;
   end;
   totpoz:=totpoz+len;
   writedata:=0;
end;

function checkbmp: integer;   // перевірка чи є в завантаженому малюнку
вбудована інформація
var
   rt : array [1..4] of byte;
begin
   seekbmp(1);
   readdata(@rt[1],4);
   if (rt[1]=22) and (rt[2]=22) and (rt[3]=77) and (rt[4]=77) then checkbmp:=0
   else checkbmp:=-1;
end;

procedure TForm2.loaddir;
var
   x : integer;
   hd : Fhandle;
   s : string;
begin
   listbox1.items.clear;
   listbox2.items.clear;
   for x:=1 to present do freemem(fat_[x].data_,fat_[x].size);
   present:=0;
   ft2cnt:=0;
   estlen:=maxlen;
   if checkbmp=0 then
   begin
      seekbmp(5);
      hd.next:=1;
      while hd.next<>0 do
      begin
         inc(present);
         fat_[present].name:='                ';
         readdata(@fat_[present].name[1],16);
         readdata(@fat_[present].size,4);
         readdata(@hd.next,1);
         getmem(fat_[present].data_,fat_[present].size);
         readdata(fat_[present].data_,fat_[present].size);
         s:=inttostr(fat_[present].size);
         while length(s)<7 do s:=' '+s;
         listbox2.items.add(fat_[present].name+'          '+s+'
'+inttostr(present));
         estlen:=estlen-fat_[present].size-21;
      end;
   end;
   label3.caption:=inttostr(estlen);
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
```

```
   present:=0;
end;

function chsel (list : tlistbox):integer;
var
   ok,x : integer;
begin
   ok:=0;
   for x:=1 to list.items.count do if list.selected[x-1] then ok:=x;
   chsel:=ok;
end;

procedure TForm2.PopupMenu1Popup(Sender: TObject);
begin
   popupmenu1.items[0].enabled:=(popupmenu1.PopupComponent.name='ListBox2');
   popupmenu1.items[1].enabled:=(popupmenu1.PopupComponent.name='ListBox2');
   popupmenu1.items[3].enabled:=(popupmenu1.PopupComponent.name='ListBox1');
   if panel1.color <> clblack then popupmenu1.items[0].enabled:=false;
end;

procedure TForm2.N2Click(Sender: TObject);
var
   x,y : integer;
   s : string;
begin
   if popupmenu1.PopupComponent.name='ListBox2' then
   begin
      if chsel(listbox2)=0 then exit;
      if length(listbox2.items[listbox2.itemindex])>36 then
      begin

estlen:=estlen+21+FAT_[strtoint(copy(listbox2.items[listbox2.itemindex],length
(listbox2.items[listbox2.itemindex])-2,3))].size;
         listbox1.items.add(listbox2.items[listbox2.itemindex]);
      end
      else
      begin
        for x:= 1 to ft2cnt do
        begin
           s:=ft2[x].name;
           while pos('\',s) <> 0 do delete (s,1,pos('\',s));
           if length(s)>16 then setlength(s,16);
           while length(s)<16 do s:=s+' ';
           if s=copy(listbox2.items[listbox2.itemindex],1,16) then
           begin
              estlen:=estlen+21+ft2[x].size;
              for y:=x to ft2cnt-1 do ft2[y]:=ft2[y+1];
              dec(ft2cnt);
              break;
           end;
         end;
      end;
      listbox2.items.delete(listbox2.itemindex);
      label3.caption:=inttostr(estlen);
   end;
end;

procedure TForm2.N4Click(Sender: TObject);
begin
   if chsel(listbox1)=0 then exit;
   if estlen <
(21+FAT_[strtoint(copy(listbox1.items[listbox1.itemindex],length(listbox1.item
s[listbox1.itemindex])-2,3))].size) then
   begin
      application.messagebox('Відновлення неможливе так як в малюнку не
вистачає місця','',$10);
      exit;
   end;
   listbox2.items.add(listbox1.items[listbox1.itemindex]);
```

```
   estlen:=estlen-21-
FAT_[strtoint(copy(listbox1.items[listbox1.itemindex],length(listbox1.items[li
stbox1.itemindex])-2,3))].size;
   listbox1.items.delete(listbox1.itemindex);
   label3.caption:=inttostr(estlen);
end;

procedure TForm2.N1Click(Sender: TObject);
var
   f : file;
   s,s1 : string;
begin
   if not ldfl.execute then exit;
   assignfile(f,ldfl.filename);
   filemode:=0;
   if ioresult <> 0 then ;
   {$I-}
   reset(f,1);
   if ioresult <> 0 then
   begin
      application.messagebox('Неможливо відкрити вказаний файл','',$10);
      exit;
   end;
   inc (ft2cnt);
   if estlen < filesize(f) then
   begin
      application.messagebox('Не хватает свободного места','',$10);
      closefile(f);
      exit;
   end;
   ft2[ft2cnt].name:=ldfl.filename;
   ft2[ft2cnt].size:=filesize(f);
   closefile(f);
   s:=ldfl.filename;
   while pos('\',s) <> 0 do delete (s,1,pos('\',s));
   if length(s)>16 then setlength(s,16);
   while length(s)<16 do s:=s+' ';
   s1:=inttostr(ft2[ft2cnt].size);
   while length(s)<7 do s:=' '+s;
   listbox2.items.add(s+'         '+s1);
   estlen:=estlen-ft2[ft2cnt].size-21;
   label3.caption:=inttostr(estlen);
end;

procedure TForm2.Button2Click(Sender: TObject);
begin
   close;
end;

procedure savedata(hdl : FHandle);  // запис інформації в малюнок
var
   hdl1 : FHandle;
begin
   getmem(hdl1.data_,hdl.size);
   move(hdl.data_^,hdl1.data_^,hdl.size);
   hdl1.size:=hdl.size;
   hdl1.name:=hdl.name;
   while pos('\',hdl1.name)<> 0 do delete (hdl1.name,1,pos('\',hdl1.name));
   while length(hdl1.name)<16 do hdl1.name:=hdl1.name+' ';
   if fcnt=form2.listbox2.items.count then hdl.next:=0 else hdl.next:=255;
   hdl1.next:=hdl.next;
   writedata(@hdl1.name[1],16);
   writedata(@hdl1.size,4);
   writedata(@hdl1.next,1);
   writedata(hdl1.data_,hdl.size);
   freemem(hdl1.data_,hdl.size);
   inc (fcnt);
end;
```

```
procedure TForm2.Button1Click(Sender: TObject);
var
    hh : array [1..4] of byte;
    x,y : integer;
    s : shortstring;
    f : file;
begin
   if listbox2.items.count>0 then
   begin
      hh[1]:=24;
      hh[2]:=06;
      hh[3]:=19;
      hh[4]:=77;
   end;
   seekbmp(1);
   writedata(@hh[1],4);
   fcnt:=1;
   for x:=1 to listbox2.items.count do if length(listbox2.items[x-1])>37 then
      savedata(FAT_[strtoint(copy(listbox2.items[x-1],length(listbox2.items[x-
1])-2,3))]);
   for x:=1 to listbox2.items.count do if length(listbox2.items[x-1])<37 then
   begin
      for y:= 1 to ft2cnt do
      begin
         s:=ft2[y].name;
         while pos('\',s) <> 0 do delete (s,1,pos('\',s));
         if length(s)>16 then setlength(s,16);
         while length(s)<16 do s:=s+' ';
         if s=copy(listbox2.items[x-1],1,16) then
         begin
            assignfile(f,ft2[y].name);
            if ioresult <> 0 then ;
            {I-}
            filemode:=0;
            reset(f,1);
            if ioresult <> 0 then
            begin
               s:='Неможливо відкрити файл'+ft2[y].name+#0;
               application.messagebox(@s[1],'',$10);
               exit;
            end;
            getmem(ft2[y].data_,ft2[y].size);
            blockread(f,ft2[y].data_^,ft2[y].size);
            closefile(f);
            savedata(ft2[y]);
            freemem(ft2[y].data_,ft2[y].size);
            break;
         end;
      end;
   end;
   if not svfl.execute then exit;
   bmp.savetofile(svfl.filename);
end;

procedure TForm2.N3Click(Sender: TObject);
var
   s : shortstring;
   f : file;
begin
   if popupmenu1.popupcomponent.name='ListBox2' then
   begin
      if chsel(listbox2)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end
   else
   begin
      if chsel(listbox1)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end;
```

```
   if length(s)<36 then exit;
   extr.filename:=FAT_[strtoint(copy(s,length(s)-2,3))].name;
   if not extr.execute then exit;
   assignfile(f,extr.filename);
   if ioresult <> 0 then;
   filemode:=2;
   {$I-}
   rewrite(f,1);
   if ioresult <> 0 then
   begin
      application.messagebox(Неможливо створити вказаний файл','',$10);
      exit;
   end;
   blockwrite(f,FAT_[strtoint(copy(s,length(s)-
2,3))].data_^,FAT_[strtoint(copy(s,length(s)-2,3))].size);
   closefile(f);
end;

procedure TForm2.Panel1DblClick(Sender: TObject);
begin
   panel1.color:=clblack;
end;

procedure TForm2.N5Click(Sender: TObject);
var
   x,y : integer;
   s : string;
begin
   if popupmenu1.popupcomponent.name='ListBox2' then
   begin
      if chsel(listbox2)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end
   else
   begin
      if chsel(listbox1)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end;

   if length(s)<36 then exit;
   form1.Memo2.lines.clear;
   y:=strtoint(copy(s,length(s)-2,3));
   s:='';
   x:=1;
   while x<= FAT_[y].size do
   begin
      if FAT_[y].data_^[x]<>13 then s:=s+chr(FAT_[y].data_^[x])
      else
      begin
         form1.Memo2.lines.add(s);
        s:='';
         inc(x);
      end;
      inc(x);
   end;
   if s<>'' then form1.Memo2.lines.add(s);
   form1.fmode:=2;
   form1.Button1Click(nil);
   form1.showmodal;
end;

procedure TForm2.N6Click(Sender: TObject);
var
   x,y : integer;
   s : string;
begin
   if popupmenu1.popupcomponent.name='ListBox2' then
   begin
```

```
      if chsel(listbox2)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end
   else
   begin
      if chsel(listbox1)=0 then exit;
      s:=listbox2.items[listbox2.itemindex];
   end;

   if length(s)<36 then exit;
   form1.richedit1.lines.clear;
   y:=strtoint(copy(s,length(s)-2,3));
   s:='';
   x:=1;
   while x<= FAT_[y].size do
   begin
      if FAT_[y].data_^[x]<>13 then s:=s+chr(FAT_[y].data_^[x])
      else
      begin
         form1.richedit1.lines.add(s);
         s:='';
         inc(x);
      end;
      inc(x);
   end;
   if s<>'' then form1.richedit1.lines.add(s);
   form1.fmode:=2;
   form1.showmodal;
end;

end.
```

PASSWORD__STEGOGRAPHY.PAS - модуль створення стегоключа

```
unit password__Stegography;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
  Tpasswrd = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
     password : shortstring;
    { Public declarations }
  end;

var
  passwrd: Tpasswrd;

implementation

{$R *.DFM}

procedure Tpasswrd.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
   if key=#13 then
   begin
      password:=edit1.text;
      close;
   end
   else
   if key=#27 then close;
end;

procedure Tpasswrd.FormCreate(Sender: TObject);
begin
   password:='';
end;

end.
```

MES.PAS – модуль формування повідомлення

```
unit Mes;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, QuickRpt, StdCtrls, ComCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    RichEdit1: TRichEdit;
    dlg1: TOpenDialog;
    Button3: TButton;
    Memo1: TMemo;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Button2: TButton;
    Button4: TButton;
    sdlg: TSaveDialog;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
     fmode : integer;
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

function delspace(s : string):string;
begin
   while (LENGTH(S)>0) and (s[1]=' ') do delete (s,1,1);
   while (LENGTH(S)>0) and (s[length(s)]=' ') do delete (s,length(s),1);
   delspace:=s;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
   x,y : integer;
   s : string;
   pozs : array [1..13] of integer;
   df : array [1..13] of shortstring;
procedure separate;
begin
   df[1]:=delspace(copy(s,pozs[1],pozs[2]-pozs[1]));
   df[2]:=delspace(copy(s,pozs[2],pozs[3]-pozs[2]));
   df[3]:=delspace(copy(s,pozs[3],pozs[4]-pozs[3]));
   df[4]:=delspace(copy(s,pozs[4],pozs[5]-pozs[4]));
   df[5]:=delspace(copy(s,pozs[5],pozs[6]-pozs[5]));
   df[6]:=delspace(copy(s,pozs[6],pozs[7]-pozs[6]));
   df[7]:=delspace(copy(s,pozs[7],pozs[8]-pozs[7]));
   df[8]:=delspace(copy(s,pozs[8],pozs[9]-pozs[8]));
   df[9]:=delspace(copy(s,pozs[9],pozs[10]-pozs[9]));
```

```
   df[10]:=delspace(copy(s,pozs[10],pozs[11]-pozs[10]));
   df[11]:=delspace(copy(s,pozs[11],pozs[12]-pozs[11]));
end;
begin
   if fmode=1 then
   begin
      if not dlg1.execute then exit;
      memo2.lines.loadfromfile(dlg1.filename);
   end
   else fmode:=1;
   richedit1.lines.clear;
   richedit1.lines.add(memo2.lines.strings[1]+'  '+memo2.lines.strings[3]);
   with memo2 do
   begin
      for   x:=0   to   5   do   if   (pos('Дата',lines.strings[x])<>0)   and
(pos('Повідомлення',lines.strings[x])<>0) then
      begin
         s:=lines.strings[x];
         break;
      end;

      for x:=7 to lines.count do
      begin
         s:=lines.strings[x];
         separate;

         memo1.lines.clear;
         memo3.lines.clear;
         memo4.lines.clear;
         memo1.lines.add(df[8]);
         memo3.lines.add(df[9]+' '+df[10]);
         memo4.lines.add(df[11]);
         s:=df[1];
         while length(s)<10 do s:=s+' ';
         s:=s+df[2];
         while length(s)<32 do s:=s+' ';
         s:=s+df[3];
         while length(s)<54 do s:=s+' ';
         s:=s+df[4];
         while length(s)<60 do s:=s+' ';

         if (length(df[5])>0) and (pos(',',df[5])=0) then df[5]:=df[5]+',';
         while length(df[5])<10 do df[5]:=' '+df[5];
         s:=s+df[5];
         while length(s)<72 do s:=s+' ';

         if (length(df[6])>0) and (pos(',',df[6])=0) then df[6]:=df[6]+',';
         while length(df[6])<10 do df[6]:=' '+df[6];
         s:=s+df[6];
         while length(s)<83 do s:=s+' ';

         if (length(df[7])>0) and (pos(',',df[7])=0) then df[7]:=df[7]+',';
         while length(df[7])<12 do df[7]:=' '+df[7];
         s:=s+df[7];
         while length(s)<97 do s:=s+' ';

         s:=s+memo1.lines.strings[0];
         while length(s)<115 do s:=s+' ';

         s:=s+memo3.lines.strings[0];
         while length(s)<132 do s:=s+' ';

         s:=s+memo4.lines.strings[0];
         richedit1.lines.add(s);

         y:=1;
         while     (memo1.lines.count>y)     or     (memo3.lines.count>y)     or
(memo4.lines.count>y) do
         begin
```

```
            s:='
';
            if (memo1.lines.count>y) then s:=s+memo1.lines.strings[y];
            while length(s)<115 do s:=s+' ';
            if (memo3.lines.count>y) then s:=s+memo3.lines.strings[y];
            while length(s)<132 do s:=s+' ';
            if (memo4.lines.count>y) then s:=s+memo4.lines.strings[y];
            richedit1.lines.add(s);
            inc (y);
         end;
         richedit1.lines.add('----------------------------------------------
------------------------------------------------------------------------------
----------------------');
      end
   end;
   with richedit1 do
   begin
      selstart:=0;
      sellength:=65535;
      selattributes.name:='courier';
      sellength:=0;
      try
         setfocus;
      except
      end;
   end;
end;


procedure TForm1.FormResize(Sender: TObject);
begin
   richedit1.width:=clientwidth;
   richedit1.height:=clientheight-button1.height;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
   if dlg1.execute then
   richedit1.lines.loadfromfile(dlg1.filename);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
   if sdlg.execute then
   richedit1.lines.savetofile(sdlg.filename);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   fmode:=1;
end;

end.
```

HAMMING.PAS – перешкодостійке кодування методом Хеммінга

```
unit Hamming;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, XPMan;

type
  TForm1 = class(TForm)
    XPManifest1: TXPManifest;
    PC1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    Button1: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Memo1: TMemo;
    Memo2: TMemo;
    Label5: TLabel;
    Memo3: TMemo;
    Label6: TLabel;
    Memo4: TMemo;
    Memo5: TMemo;
    Memo6: TMemo;
    Label7: TLabel;
    Label1: TLabel;
    Button2: TButton;
    Label8: TLabel;
    procedure Button2Click(Sender: TObject);
    procedure kodhex;
    procedure binar;
    procedure kodhem;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  hex:array[1..2,1..8000] of integer;
  dlina:integer; //довжина тексту

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
kodhex;
binar;
kodhem;
end;
       //сам алгоритм кодування Хеммінга
procedure tform1.kodhem;
var
s,bina,hem:string;
a,b,i,z,b1,b2,b3:integer;
begin
s:=memo1.Text;
hem:='';
```

```
z:=length(s);
a:=1;
while a<z do
begin
i:=0;
if s[a]='1' then i:=i xor 3;
if s[a+1]='1' then i:=i xor 5;
if s[a+2]='1' then i:=i xor 6;
if s[a+3]='1' then i:=i xor 7;
b:=i mod 8;
b3:=b div 4;
b:=b mod 4;
b2:=b div 2;
b1:=b mod 2;
bina:=inttostr(b1)+inttostr(b2)+s[a]+inttostr(b3)+s[a+1]+s[a+2]+s[a+3];
hem:=hem+bina;
a:=a+4;
end;
memo2.Text:=hem;
memo3.Text:=hem;
end;
       //Перетворення у двійковий вигляд
procedure tform1.binar;
var
a,b,temp,t1:integer;
bin:string;
begin
bin:='';
for a:=1 to dlina do
begin
for b:=1 to 2 do
begin
temp:=hex[b,a] div 8;
bin:=bin+inttostr(temp);
t1:=hex[b,a] mod 8;
temp:=t1 div 4;
bin:=bin+inttostr(temp);
t1:=t1 mod 4;
temp:=t1 div 2;
bin:=bin+inttostr(temp);
temp:=t1 mod 2;
bin:=bin+inttostr(temp);
end;
end;
memo1.Text:=bin;
end;
       //Перетворення у шістнадцятковий вигляд
procedure tform1.kodhex;
var
s,h,h1,h2:string;
b,i:integer;
begin
s:=memo6.Text;
dlina:=length(s);
if dlina=0 then exit;
h:='';
for b:=1 to dlina do
begin
i:=ord(s[b]);
hex[1,b]:=i div 16;
h1:=inttostr(hex[1,b]);
case hex[1,b] of
10:h1:='A';
11:h1:='B';
12:h1:='C';
13:h1:='D';
14:h1:='E';
15:h1:='F';
end;
```

```
hex[2,b]:=i-(hex[1,b]*16);
h2:=inttostr(hex[2,b]);
case hex[2,b] of
10:h2:='A';
11:h2:='B';
12:h2:='C';
13:h2:='D';
14:h2:='E';
15:h2:='F';
end;
h:=h+h1+h2+',';
end;
delete(h,length(h),1);
memo5.Text:=h;
end;
//Підпрограмма визначення кількості помилок та їх виправлення, якщо вони є
procedure TForm1.Button2Click(Sender: TObject);
var
s:string;
a,b,i,z,f,osh:integer;
begin
s:=memo3.Text;
z:=length(s);
a:=1;
osh:=0;
while a<z do
begin
i:=0;
for f:=0 to 6 do if s[a+f]='1' then i:=i xor (f+1);
i:=i mod 8;
if i<>0 then
begin
inc(osh);
if s[a+i-1]='0' then s[a+i-1]:='1' else s[a+i-1]:='0';
end;
a:=a+7;
end;
label8.Caption:='Знайдено помилок    '+inttostr(osh)+' шт.';
memo4.Text:=s;
end;

end.
```

DES.PAS – шифрування стегоключа алгоритмом DES

```
unit DES;

interface

Uses Windows, Classes, SysUtils, Math, Dialogs;

Type
  TBitString = Array of Boolean;
  PBitString = ^TBitString;

  TSplitKeyParts = record
    C:TBitString;
    D:TBitString;
    end;
  TSplitKey = Array[0..16]Of TSplitKeyParts;

  TConcatKey = Array[0..15]Of TBitString;

  TIPKeyParts = record
    L:TBitString;
    R:TBitString;
    end;
  TIPKey = Array[0..16]OF TIPKeyParts;

Const
DES_PC1:Array[0..55] Of Byte = (57,49,41,33,25,17,9,
                                1,58,50,42,34,26,18,
                                10,2,59,51,43,35,27,
                                19,11,3,60,52,44,36,
                                63,55,47,39,31,23,15,
                                7,62,54,46,38,30,22,
                                14,6,61,53,45,37,29,
                                21,13,5,28,20,12,4);

DES_PC2:Array[0..47] Of Byte = (14,17,11,24,1,5,
                                3,28,15,6,21,10,
                                23,19,12,4,26,8,
                                16,7,27,20,13,2,
                                41,52,31,37,47,55,
                                30,40,51,45,33,48,
                                44,49,39,56,34,53,
                                46,42,50,36,29,32);

DES_IP:Array[0..63] Of Byte = (58,50,42,34,26,18,10,2,
                               60,52,44,36,28,20,12,4,
                               62,54,46,38,30,22,14,6,
                               64,56,48,40,32,24,16,8,
                               57,49,41,33,25,17,9,1,
                               59,51,43,35,27,19,11,3,
                               61,53,45,37,29,21,13,5,
                               63,55,47,39,31,23,15,7);

DES_E:Array[0..47] Of Byte = (32,1,2,3,4,5,
                              4,5,6,7,8,9,
                              8,9,10,11,12,13,
                              12,13,14,15,16,17,
                              16,17,18,19,20,21,
                              20,21,22,23,24,25,
                              24,25,26,27,28,29,
                              28,29,30,31,32,1);

S_BOXES:Array[0..7,0..3,0..15]Of Byte = (
((14,04,13,01,02,15,11,08,03,10,06,12,05,09,00,07),
  (00,15,07,04,14,02,13,01,10,06,12,11,09,05,03,08),
```

```
  (04,01,14,08,13,06,02,11,15,12,09,07,03,10,05,00),
  (15,12,08,02,04,09,01,07,05,11,03,14,10,00,06,13)),

((15,01,08,14,06,11,03,04,09,07,02,13,12,00,05,10),
      (03,13,04,07,15,02,08,14,12,00,01,10,06,09,11,05),
      (00,14,07,11,10,04,13,01,05,08,12,06,09,03,02,15),
      (13,08,10,01,03,15,04,02,11,06,07,12,00,05,14,09)),

((10,00,09,14,06,03,15,05,01,13,12,07,11,04,02,08),
      (13,07,00,09,03,04,06,10,02,08,05,14,12,11,15,01),
      (13,06,04,09,08,15,03,00,11,01,02,12,05,10,14,07),
      (01,10,13,00,06,09,08,07,04,15,14,03,11,05,02,12)),

((07,13,14,03,00,06,09,10,01,02,08,05,11,12,04,15),
      (13,08,11,05,06,15,00,03,04,07,02,12,01,10,14,09),
      (10,06,09,00,12,11,07,13,15,01,03,14,05,02,08,04),
      (13,15,00,06,10,01,13,08,09,04,05,11,12,07,02,14)),

((02,12,04,01,07,10,11,06,08,05,03,15,13,00,14,09),
      (14,11,02,12,04,07,13,01,05,00,15,10,03,08,09,06),
      (04,02,01,11,10,13,07,08,15,09,12,05,06,03,00,14),
      (11,08,12,07,01,14,02,13,06,15,00,09,10,04,05,03)),

((12,01,10,15,09,02,06,08,00,13,03,04,14,07,05,11),
      (10,15,04,02,07,12,09,05,06,01,13,14,00,11,03,08),
      (09,14,15,05,02,08,12,03,07,00,04,10,01,13,11,06),
      (04,03,02,12,09,05,15,10,11,14,01,04,06,00,08,13)),

((04,11,02,14,15,00,08,13,03,12,09,07,05,10,06,01),
      (13,00,11,07,04,09,01,10,14,03,05,12,02,15,08,06),
      (01,04,11,13,12,03,07,14,10,15,06,08,00,05,09,02),
      (06,11,13,08,01,04,10,07,09,05,00,15,14,02,03,12)),

((13,02,08,04,06,15,11,01,10,09,03,14,05,00,12,07),
      (01,15,13,08,10,03,07,04,12,05,06,11,00,14,09,02),
      (07,11,04,01,09,12,14,02,00,06,10,13,15,03,05,08),
      (02,01,14,07,04,10,08,13,15,12,09,00,03,05,06,11))
);

DES_P:Array[0..31] Of Byte = (16,7,20,21,
                              29,12,28,17,
                              1,15,23,26,
                              5,18,31,10,
                              2,8,24,14,
                              32,27,3,9,
                              19,13,30,6,
                              22,11,4,25);

DES_REVERSE_IP:Array[0..63] Of Byte = (40,8,48,16,56,24,64,32,
              39,7,47,15,55,23,63,31,
              38,6,46,14,54,22,62,30,
              37,5,45,13,53,21,61,29,
              36,4,44,12,52,20,60,28,
              35,3,43,11,51,19,59,27,
              34,2,42,10,50,18,58,26,
              33,1,41,9,49,17,57,25);

DES_LSH:Array[0..15] Of Byte = (1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1);

Function BinToInt(S:TBitString):Integer;
Function IntToBin(N:Integer;Precision:Integer=8):TBitString;

Function BinToStr(Bits:TBitString):String;
Function StrToBin(S:String):TBitString;

Function AnsiStrToBin(S:String; Zeroes:Boolean=True):TBitString;
Function BinToAnsiStr(Bits:TBitString):String;

Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
```

```
Function ConcatBits(Bits:Array Of TBitString):TBitString;

Function DESEncode(S,Key:String):TBitString;
Function DESDecode(S,Key:String):TBitString;

Function GetPermutedKey(Key:TBitString):TBitString;
Function GetPermutedKey2(Key:TBitString):TBitString;

Function GetSplitKey(Key:TBitString):TSplitKey;
Function GetConcatKey(Key:TSplitKey):TConcatKey;
Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Function Get(R,K:TBitString):TBitString;
Function GetSBox(Index:Integer; T:TBitString):TBitString;
Function GetReverseIP(RL:TBitString):TBitString;
Procedure ReverseSubKeys(Var Keys:TConcatKey);

implementation

Function ConcatBits(Bits:Array Of TBitString):TBitString;
Var
I,C:Integer;
Begin
SetLength(Result,0);
For C:=0 To Length(Bits)-1 Do
  Begin
  SetLength(Result,Length(Result)+Length(Bits[C]));
  For I:=0 To Length(Bits[C])-1 Do
    Result[Length(Result)-Length(Bits[C])+I]:=Bits[C][I];
  End;
End;

Procedure CopyBits(Var Dest:TBitString; Source:TBitString; NBits:Integer);
Var
I:Integer;
Begin
SetLength(Dest,NBits);
For I:=0 To NBits-1 Do
  Dest[I]:=Source[I];
End;

Function BinToInt(S: TBitString): Integer;
Var
L,I:Integer;
Begin
Result:=0;
L:=Length(S);
IF L=0 Then
  Raise EConvertError.Create(' Бітовий рядок довжини нуль ');
For I:= L-1 DownTo 0 Do
  Result:=Result+Ord(S[I])*Trunc(Power(2, L-I-1));
End;

Function IntToBin(N:Integer; Precision:Integer):TBitString;
Var
BitList:TList;
Bit:PBoolean;
Begin
SetLength(Result,0);
BitList:=TList.Create;
While N>0 Do
  Begin
  New(Bit);
  Bit^:=Boolean(N mod 2);
  BitList.Insert(0,Bit);
  N:=N div 2;
  End;
While BitList.Count<Precision Do
  Begin
  New(Bit);
```

```
  Bit^:=False;
  BitList.Insert(0,Bit);
  End;
For N:=0 To BitList.Count-1 Do
  Begin
  SetLength(Result,N+1);
  Bit:=BitList.Items[N];
  Result[N]:=Bit^;
  Dispose(Bit);
  End;
BitList.Free;
end;

Function AnsiStrToBin(S: String; Zeroes:Boolean):TBitString;
Var
Temp,B:TBitString;
L,I,J:Integer;
Begin
L:=0;
SetLength(Result,L);
SetLength(Temp,L);
SetLength(B,0);
For I:=1 To Length(S) Do
  Begin
  B:=IntToBin(Ord(S[I]));
  L:=L+Length(B);
  SetLength(Temp,L);
  For J:=0 To Length(B)-1 Do
    Temp[Length(Temp)-Length(B)+J]:=B[J];
  End;
Result:=Temp;
End;

Function BinToStr(Bits:TBitString):String;
Var
I,L:Integer;
Begin
Result:='';
L:=Length(Bits);
IF L=0 Then
  Raise EConvertError.Create(' Бітовий рядок довжини нуль ');
For I:=0 To L-1 Do
  IF Bits[I] Then Result:=Result+'1'
  Else Result:=Result+'0';
End;

Function StrToBin(S:String):TBitString;
Var
I:Integer;
Begin
SetLength(Result,0);
For I:=1 To Length(S) Do
  Begin
  IF (S[I]<>'1')And(S[I]<>'0') Then
    Raise EConvertError.Create(S+' помилковий двійковий рядок');
  SetLength(Result,I);
  Result[ I-1]:=Boolean(StrToInt(S[I]));
  End;
End;

Function BinToAnsiStr(Bits:TBitString):String;
Var
I:Integer;
B:TBitString;
Begin
Result:='';
SetLength(B,8);
I:=0;
While I<=Length(Bits)-8 Do
```

```
  Begin
  CopyMemory(B,Ptr(Integer(Bits)+I),8);
  Result:=Result+Char(BinToInt(B));
  Inc(I,8);
  End;
End;

Function GetPermutedKey(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(DES_PC1));
For I:=0 To Length(DES_PC1)-1 Do
  Result[I]:=Key[DES_PC1[I]-1];
End;

Function GetPermutedKey2(Key:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,Length(DES_PC2));
For I:=0 To Length(DES_PC2)-1 Do
  Result[I]:=Key[DES_PC2[I]-1];
End;

Function GetSplitKey(Key:TBitString):TSplitKey;
  Function LeftShift(Key:TBitString; N:Integer):TBitString;
  Var
  I,J:Integer;
  Temp:TBitString;
  Begin
  SetLength(Result,28);
  SetLength(Temp,28);
  For I:=0 To 27 Do
    Temp[I]:=Key[I];
  For J:=1 To N Do
    Begin
    For I:=1 To 27 Do
      Result[ I-1]:=Temp[I];
    Result[27]:=Temp[0];
    For I:=0 To 27 Do
      Temp[I]:=Result[I];
    End;
  End;
Var
I,J:Integer;
Begin
For J:=1 To 16 Do
  Begin
  SetLength(Result[J].C,28);
  SetLength(Result[J].D,28);
  End;
CopyBits(Result[0].C,Key,28);
CopyBits(Result[0].D,TBitString(Integer(Key)+28),28);
For I:=1 To 16 Do
  Begin
  Result[I].C:=LeftShift(Result[ I-1].C,DES_LSH[ I-1]);
  Result[I].D:=LeftShift(Result[ I-1].D,DES_LSH[ I-1]);
  End;
End;

Function GetConcatKey(Key:TSplitKey):TConcatKey;
Var
I:Integer;
Temp:TBitString;
Begin
For I:=0 To 15 Do
  Begin
  SetLength(Result[I],56);
```

```
  Temp:=ConcatBits([Key[I+1].C,Key[I+1].D]);
  Result[I]:=GetPermutedKey2(Temp);
  End;
End;

Function GetIPKey(M:TBitString; ConcatKey:TConcatKey):TIPKey;
Var
I,J:Integer;
IP, F:TBitString;
Begin
For I:=0 To 16 Do
  Begin
  SetLength(Result[I].L,32);
  SetLength(Result[I].R,32);
  End;

SetLength(IP,64);
For I:=0 To Length(DES_IP)-1 Do
  IP[I]:=M[DES_IP[I]-1];

For I:=0 To 31 Do
  Result[0].L[I]:=IP[I];
For I:=32 To 63 Do
  Result[0].R[ I-32]:=IP[I];

For I:=1 To 16 Do
  Begin
  Result[I].L:=Result[ I-1].R;
  F:=Get(Result[ I-1].R,ConcatKey[ I-1]);
  For J:=0 To 31 Do
    Result[I].R[J]:=Result[ I-1].L[J] XOR F[J];
  End;
End;

Function Get(R,K:TBitString):TBitString;
Var
I,J:Integer;
S,E,KE,F,T:TBitString;
Begin
SetLength(E,48);
For I:=0 To 47 Do
  E[I]:=R[DES_E[I]-1];

SetLength(KE,48);
For I:=0 To 47 Do
  KE[I]:=K[I] XOR E[I];

SetLength(T,6);
SetLength(F,0);
SetLength(S,4);
I:=0;
While I<48 Do
  Begin
  For J:=0 To 6 Do
    T[J]:=KE[J+I];
  S:=GetSBox(I div 6,T);
  F:=ConcatBits([F,S]);
  I:=I+6;
  End;
SetLength(Result,32);
For I:=0 To 31 Do
  Result[I]:=F[DES_P[I]-1];
End;

Function GetSBox(Index:Integer; T:TBitString):TBitString;
Var
Val,Row,Col:Integer;
Temp:TBitString;
Begin
```

```
SetLength(Result,4);
SetLength(Temp,2);
Temp[0]:=T[0];
Temp[1]:=T[5];
Row:=BinToInt(Temp);
SetLength(Temp,4);
CopyBits(Temp,TBitString(@T[1]),4);
Col:=BinToInt(Temp);
Val:=S_BOXES[Index,Row,Col];
SetLength(Result,4);
Result:=IntToBin(Val,4);
End;

Function GetReverseIP(RL:TBitString):TBitString;
Var
I:Integer;
Begin
SetLength(Result,64);
For I:=0 To Length(DES_REVERSE_IP)-1 Do
  Result[I]:=RL[DES_REVERSE_IP[I]-1];
End;

Procedure ReverseSubKeys(Var Keys:TConcatKey);
Var
I,L:Integer;
T:TBitString;
Begin
SetLength(T,48);
L:=Length(Keys);
For I:=0 To ( L-1)Div 2 Do
  Begin
  T:=Keys[I];
  Keys[I]:=Keys[( L-I)-1];
  Keys[( L-I)-1]:=T;
  End;
End;

Function DESEncode(S,Key:String):TBitString;
Var
I:Integer;
K:TBitString;
M:TBitString;
RL:TBitString;
Kplus:TBitString;
SplitKey:TSplitKey;
ConcatKey:TConcatKey;
IPKey:TIPKey;
Begin
K:=AnsiStrToBin(Key);
Kplus:=GetPermutedKey(K);
SplitKey:=GetSplitKey(Kplus);
ConcatKey:=GetConcatKey(SplitKey);
M:=AnsiStrToBin(S);
IPKey:=GetIPKey(M,ConcatKey);
SetLength(RL,64);
For I:=0 To 31 Do
  Begin
  RL[I]:=IPKey[16].R[I];
  RL[I+32]:=IPKey[16].L[I];
  End;
RL:=GetReverseIP(RL);
Result:=RL;
End;

Function DESDecode(S,Key:String):TBitString;
Var
I:Integer;
K:TBitString;
M:TBitString;
```

```
RL:TBitString;
Kplus:TBitString;
SplitKey:TSplitKey;
ConcatKey:TConcatKey;
IPKey:TIPKey;
Begin
K:=AnsiStrToBin(Key);
Kplus:=GetPermutedKey(K);
SplitKey:=GetSplitKey(Kplus);
ConcatKey:=GetConcatKey(SplitKey);
ReverseSubKeys(ConcatKey);
M:=AnsiStrToBin(S);
IPKey:=GetIPKey(M,ConcatKey);
SetLength(RL,64);
For I:=0 To 31 Do
  Begin
  RL[I]:=IPKey[16].R[I];
  RL[I+32]:=IPKey[16].L[I];
  End;
RL:=GetReverseIP(RL);
Result:=RL;
End;

end.
```

MAIN_STEGOGRAPHY.PAS – основна программа

```
unit main_Stegography;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ExtCtrls, Menus, _Stegography, about;

type

  T_Stegography = class(TForm)
    Button1: TButton;
    loadbmp: TOpenDialog;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    ScrollBox1: TScrollBox;
    Image1: TImage;
    Button5: TButton;
    PopupMenu1: TPopupMenu;
    N1: TMenuItem;
    N3: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    Main_StegographyMenu1: TMain_StegographyMenu;
    Loadfile1: TMenuItem;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure N6Click(Sender: TObject);
  private
     pt : TBuffer_;
     loaded_ : boolean;
  end;

var
  _Stegography: T_Stegography;

implementation

uses password__Stegography, Hamming;

{$R *.DFM}

// завантаження зображення
procedure T_Stegography.Button1Click(Sender: TObject);
begin
    if not loadbmp.execute then exit;
image1.picture.bitmap.loadfromfile(loadbmp.filename);
// перевірка формату малюнка. Треба 24-бітний.
    if image1.picture.bitmap.pixelformat<>pf24bit then
// Формат малюнка не підходить. Запит на перетворення формату
     if application.messagebox('Можлива робота лише з 24-бітними зображеннями.
Конвертувати?','',$11)=1 then
        image1.picture.bitmap.pixelformat:=pf24bit;
     maxcol:=((image1.picture.bitmap.width) * 3);
// максимальний об’єм даних, які можна помістити в зображення
maxlen:=((maxcol*image1.picture.bitmap.height) div 8)-25;
    if maxlen<=0 then
    begin
       maxlen:=0;
       loaded_:=false;
    end
    else loaded_:=true;
```

```
   checkbmp;
   form2.label6.caption:=inttostr(maxlen);
   form2.label3.caption:=inttostr(maxlen);
   estlen:=maxlen;
end;

procedure T_Stegography.FormCreate(Sender: TObject);
begin
   loaded_:=false;
end;

procedure T_Stegography.Button4Click(Sender: TObject);
begin
   if not loaded_ then exit;
   form2.loaddir;    // Процедура читання вбудованої інформації
   form2.showmodal;
end;

procedure T_Stegography.Button5Click(Sender: TObject);
begin
   passwrd.edit1.text:=passwrd.password;     // Запит пароля
   passwrd.showmodal;
end;

procedure T_Stegography.FormResize(Sender: TObject);
begin
   scrollbox1.width:=clientwidth;
   scrollbox1.height:=clientheight-scrollbox1.top;
end;

procedure T_Stegography.N6Click(Sender: TObject);
begin
   form1.show;
end;

end.
```

ABOUT.PAS - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TFmAbout = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FmAbout: TFmAbout;

implementation

{$R *.dfm}

procedure TFmAbout.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи кібербезпеки для приховування
інформації на основі методів стеганографії');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник:  Якименко Н.М.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив:  студент   Мороз Антон Сергійович');
Memo1.Lines.Add('           гр. КБ-19');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
end;

procedure TFmAbout.Button1Click(Sender: TObject);
begin
FmAbout.Close;
end;
end.
```