МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЩОЛОКОВ ВІКТОР АНДРІЙОВИЧ

Допускається до захисту:
Завідувач кафедри інформаційних технологій, д.т.н., доцент
_____ Т.В.Нескородєва
«____» _____ 20__ р.

ПРИХОВАНІ МЕТОДИ ЗАХИСТУ МЕТАДАНИХ В ЕЛЕКТРОНИХ ДОКУМЕНТАХ

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Керівник:
Крижановський В.Г.
Професор кафедри
інформаційних технологій,
д.т.н., професор

_____
підпис

Оцінка:___/ ___/ _____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця — 2022

АНОТАЦІЯ

Щолоков В.А. Приховані методи захисту метаданих в електронних документах. Спеціальність 125 Кібербезпека. Донецький національний університет імені Василя Стуса. Вінниця. 2022 рік.

У бакалаврській роботі на основі аналізу методики лежать загальні принципи побудови ланцюгового запису даних, що є динамічним реєстром, в якому внесення змін до запису метаданих допускається без зміни раніше внесеної інформації. При цьому зв'язок між записами метаданих забезпечується за рахунок використання криптографічної хеш-функції.

Ключові слова. Методи захисту, метадані, електроні документи.

Рис. 11, Бібліограф.: 21 найм.

Shcholokov V.A. Hidden methods of metadata protection in electronic documents. Specialty 125 Cybersecurity. Vasyl Stus Donetsk National University. Vinnitsa. 2022.

In the bachelor's work on the basis of the analysis of methods the general principles of construction of a chain record of data which is the dynamic register in which modification of record of metadata is allowed without change of earlier entered information lie. The connection between metadata records is provided through the use of cryptographic hash function.

Keywords. Methods of protection, metadata, electronic documents.

Fig. 11, Bibliographer: 21 items.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5

ВСТУП 6

1. СИСТЕМА ОБМІНУ ФАЙЛАМИ, ЩО ПРИХОВУЮТЬ, ІЗ ЗЛОВМИСНИМ ЗАХИСТОМ 8

1.1 Захист від зловмисників 8

1.2 Атака перезапису в DPF-MCORAM. 14

1.3 Атаки відкату в односерверних конструкціях 14

1.4 Огляд системи Titanium 16

1.5 Властивості приховування метаданих 18

1.6 Висновки до розділу 1 19

2. МЕТОДИКА ЗАСТОСУВАННЯ КРІПТОГРАФИЧНОГО РЕКУРСИВНОГО 2-D КОНТРОЛЮ ЦІЛІСНОСТІ МЕТАДАНИХ 20

2.1 Методика криптографічного рекурсивного 2-D контролю цілісності метаданих електронних документів Результати застосування розробленої методики 20

2.2 Результати застосування розробленої методики 23

2.3 Висновки до розділу 2 26

3. ЗАХИСТ МЕТАДАНИХ ЗА ДОПОМОГОЮ СТЕГАНОГРАФІЇ 27

3.1 Проектування програмного додатку 27

3.2 Розробка додатка на основі стеганографічного методу 33

3.3 Висновок до розділу 3 ..... **38**

ВИСНОВОК **39**

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

Додаткок А Фрагмент програмного забезпечення реалізації стеганографічного алгоритму 46

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕЛД – Електронний документ

АІС ЕД - Автоматизовані інформаційні системи електронного документообігу

ПЗ – Програмне забезпечення

КЗЗ – Комплекс засобів захисту

ЕОМ – Електронно-обчислювальна машина

ОП – Оперативна пам’ять

ПЗП – Постійний запам’ятовуючий пристрій

TCP – Transmission Control Protocol

АС – Автоматизована система

NIST– National Institute of Standards and Technology

ПК – Персональний комп’ютер

CWE – Common Weakness Enumeration

CMOS - Complementary Metal Oxide Semiconductor

SSDLC - Secure Software Development Life Cycle

CVE - Common Vulnerabilities and Exposures

SEI - Software Engineering Institute

TSP Time Stamp Protocol

ВСТУП

Актуальність та цілі. Соціальне спілкування кожної людини як ніколи пов'язане із цифровим простором. Соціальні мережі, різні електронні програми, платформи, канали використовують для постійного обміну даними. Кожен файл, відправлений поза, містить значний обсяг метаданих, систематизація яких дозволяє розкрити значні таємниці приватного життя особи.

Представлена робота є логічним продовженням дослідження функціональних можливостей способу криптографічного рекурсивного 2-D контролю цілісності метаданих електронних файлів, документів, і спрямовано практичну реалізацію рішень, раніше викладених авторами в . Розглянутий тип автоматизованих інформаційних систем електронного документообігу (АІС ЕД) призначений на вирішення завдань організації єдиного захищеного інформаційного простору у частині електронного документообігу з допомогою застосування єдиної технології обробки та зберігання документів. Інформаційні системи, що забезпечують управління документами та доступ до них протягом певного часу, називаються документними системами.

У загальному вигляді управління документами включає 4-х етапи:

1) створення документа;

2) введення документа у систему з метою доказу ведення будь-якої діяльності;

3) вживання належних заходів щодо захисту документа, в умовах, що змінюється в часі обстановки;

4) санкціоноване знищення чи передача документа до архіву.

Необхідно зазначити, що документи складаються з контенту та метаданих, які описують контекст, контент та структуру документів, а також керування ними протягом часу. При цьому метаданими документи слід керувати, як і самим документом, оскільки вони повинні бути захищені від втрати або несанкціонованого видалення та збережені або знищені відповідно до вимог,

визначених на основі проведеної діяльності. Таким чином, метадані є критично важливою інформацією, що циркулює в АІС ЕД і безпосередньо впливає на її функціональні можливості, що зумовлює необхідність вживання адекватних заходів щодо їх захисту від можливих деструктивних впливів

1. СИСТЕМА ОБМІНУ ФАЙЛАМИ, ЩО ПРИХОВУЮТЬ, ІЗ ЗЛОВМИСНИМ ЗАХИСТОМ

1.1 Захист від зловмисників

Titanium - це система обміну файлами, що приховує метадані, яка пропонує конфіденційність та цілісність проти зловмисних користувачів та серверів. У порівнянні з MCORAM, який пропонує конфіденційність проти шкідливих серверів, Titanium також пропонує цілісність. Експерименти показують , що титан від 5× до 200× швидше або більше , ніж MCORAM.

Багато компаній пропонують хмарне сховище з наскрізним шифруванням, таких як БоксКриптор, Айсдрайв, База ключів Файлова система, МЕГА, pCloud, Завіса, синхронізація та Трезорит . Ентузіазм щодо наскрізного шифрування походить з побоювання громадськості з приводу неправомірного використання особистих даних і як хакери використовують вкрадений бази даних з великих підприємств[1].

Однак наскрізне шифрування - це ще не кінець, тому що хмарні сервери, як і раніше, бачать метадані. Метадані, такі як хтось користувач акції файли з є схожий до комунікація конфіденційність — Стенфордське дослідження MetaPhone показало, що метадані телефонних дзвінків є «густо взаємопов'язаний, сприйнятливий до повторної ідентифікації, і дозволяє робити високочутливі висновки» . Колишній генерал АНБ Адвокат сказав: «Метадані говорять вам абсолютно все про чиєсь життя[1]».

Вилучення секретів із шаблонів доступу було важливою галуззю досліджень безпеки, які малий так і великий успіх. Існують роботи, які деанонімізують користувачів за допомогою соціальних зв’язків, скомпрометують зашифровані бази даних із шаблонами доступу та порушують захищене обладнання із шаблонами доступу до пам’яті. [2] Ці атаки можуть також застосовуватися до наскрізних зашифрованих систем обміну файлами.

Щоб зрозуміти вплив витоку метаданих, розглянемо програму (мал. 1) і її користь від захисту метаданих. Викривач, Аліса, хоче повідомити журналісту про скандал компанії. Якщо вони спілкуються через наскрізне зашифроване сховище, сервери знають, що Аліса ділиться файлами з журналістом, і коли вони отримують доступ. Якщо сервери вступають у змову з компанією, компанія може виявити викривача[1].

End-to-end encrypted storage	**Metadata-hiding file-sharing systems**
Alice and Journalist have accounts	Users remain anonymous
Alice and Journalist share file F • Alice has write permission • Journalist has read permission	F's access control list is unknown
Alice wrote to F on May 26 Journalist read F on May 27	Someone accessed some file on May 26 Someone accessed some file on May 27

Рис. 1: Порівняння гарантій безпеки між наскрізним зашифрованим сховищем і системами обміну файлами, які приховують метадані.

Більше того, хакер або зловмисний працівник хмари вже може знати особу викривача.

Алісі і журналісту потрібна система зберігання, яка приховує метадані від серверів. Ця система повинна мати анонімність, щоб сервер не дізнавався, з ким він спілкується. Він повинен приховувати шаблони доступу, щоб сервер не міг визначити поведінку користувачів[1].

Чи існує така система обміну файлами, що приховує метадані? В останнє десятиліття дослідники намагалися розробити практичне сховище для приховування метаданих. Це складно, тому що довіряти майже нічому: і користувачі, і сервери можуть бути шкідливими. Нам потрібен шкідливий захист.

Ціль 1: захист від зловмисників. [1]. Стандартне рішення для приховування шаблонів доступу, неуважна оперативна пам'ять (ORAM), за своєю суттю є одно користувацькою, що означає, що якщо сховище ORAM використовується спільно зі зловмисниками, конфіденційність зникає. Щоб безпечно ділитися файлами з багатьма користувачами, потрібні нові підходи.

Підхід: ми робимо Titanium захищеним від зловмисників, зводячи до мінімуму участь користувачів у протоколі. Єдині операції, які виконують користувачі, — це надсилання запитів і отримання відповідей через API.

Користувачі ніколи не торкаються даних, що зберігаються на серверах. Такий підхід забезпечує чистий інтерфейс і дозволяє легко міркувати про безпеку від зловмисників.

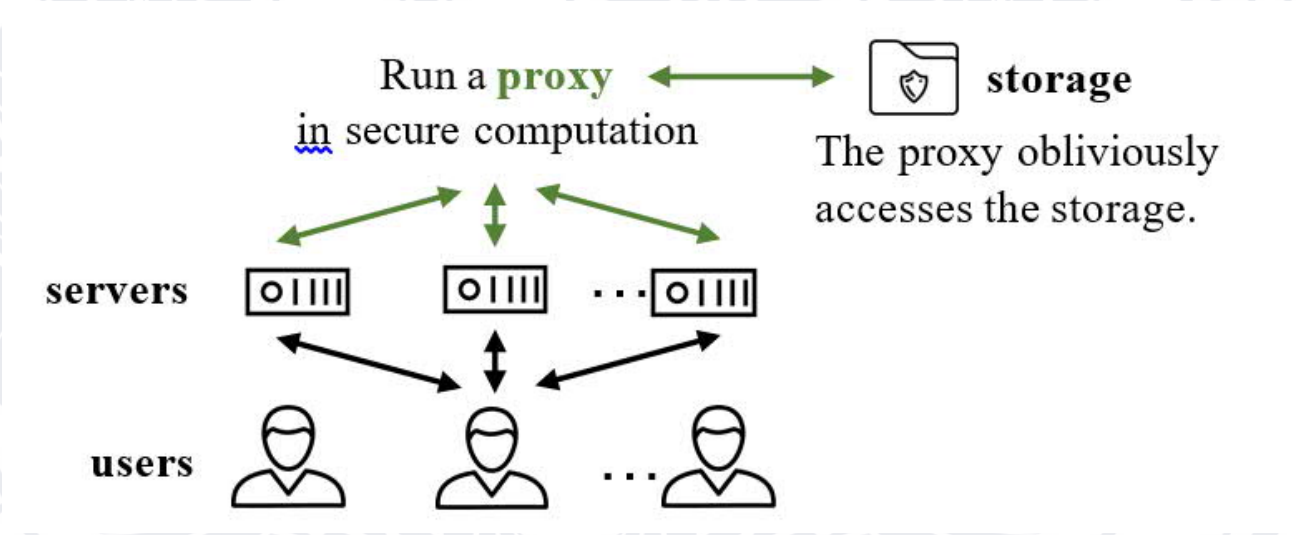


Рис. 2: Модель системи Titanium.

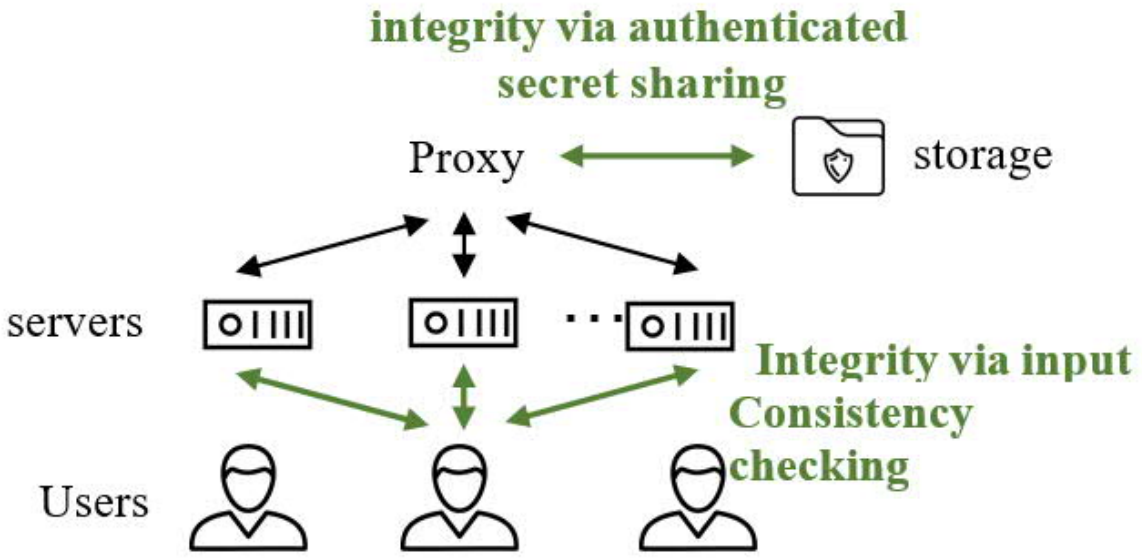
Як показано на рис. 2, коли користувач Titanium хоче прочитати або записати файл, він надсилає запит на N серверів. [1]. Цей запит надсилається в секретних спільних ресурсах, тому жоден сервер N 1 не знає, що це за запит. Сервери разом запускають проксі-сервер у захищеному N-сторонньому обчисленні, де стан проксі-сервера прихований від серверів. Проксі-сервер перевіряє, що запит користувача є законним, і, якщо так, отримує доступ до сховища від імені користувача та відповідає користувачеві (через сервери). Максимум, зловмисники можуть створити неавторизований запит, але проксі-

сервер не обробить його. Таким чином, Titanium забезпечує зловмисний захист від користувачів.

Ціль 2: цілісність зберігання даних. [1,2]. Попередня робота не може гарантувати цілісність даних у сховищі від шкідливих серверів. Шкідливий сервер може, наприклад, здійснити атаку відкату, коли користувач отримує застарілий файл від шкідливих серверів. У §II-B ми показуємо, що шкідливі сервери (або користувачі) можуть порушувати цілісність деяких попередніх напівчесних конструкцій.

Підхід: ми хочемо, щоб файли писалися лише користувачами, які мають право писати, і хочемо, щоб усі авторизовані читачі бачили останню версію файлу. Для цього потрібен певний код аутентифікації повідомлень (MAC) для всього сховища файлів. Складним є те, що такі MAC не повинні розкривати ідентифікатори користувачів або файлів. Titanium використовує аутентифікований секретний обмін для зберігання даних, що приховує MAC від усіх сторін і таким чином зберігає конфіденційність.

Більше того, принаймні один із серверів вважається чесним. MAC-адреси дозволяють цьому серверу виявляти, чи надсилається користувачеві неправильна версія файлу чи інший файл, який користувач не запитував, тим самим забезпечуючи цілісність, як показано на мал. 3.



Мал. 3: Забезпечення цілісності в Titaniut для (1) даних у сховищі та (2) введення та виходу користувача.

Ціль 3: цілісність введених і вихідних даних користувачів. Хоча аутентифікований доступ до секретів забезпечує цілісність сховища даних, він не забезпечує цілісність даних, надісланих та отриманих від користувача. Якщо зловмисний сервер може змінити такі дані, користувач може прочитати інший файл, записати дані в інший файл або поділитися файлом з незнайомцем. Додавання підпису не працює, оскільки він не анонімний. Існуючий примітив, розроблений для моделі клієнт-сервер, неінтерактивні докази із загальним доступом (SNIP), запропонований у Prio, також не працює, оскільки не забезпечує цілісність проти шкідливих серверів[1].

Підхід: ми розробляємо зловмисно захищений протокол введення/виводу між серверами та користувачами, щоб користувачі могли підтвердити, що проксі отримує правильні введення, а користувач отримує правильні виходи від проксі, як показано на рис. 3. Цей протокол також забезпечує конфіденційність, оскільки сервери не бачать вхідні та вихідні дані. Для цієї перевірки узгодженості ми використовуємо інструмент, який зазвичай використовується в системах криптографічного підтвердження, лему Шварца-Ціпеля.

1.2 Атака перезапису в DPF-MCORAM

У файл користувач записує дані, функції розподіленої точки, надсилаючи на сервери (DPF). «Хороший» DPF тільки оновлює файл користувача, але зловмисник може створити «поганий» DPF, який змінює чужий файл або перезаписує все сховище іншими даними. Існуюче рішення, перевірений DPF, може вирішувати лише останній, але не перший, і працює надзвичайно повільно в цьому налаштування. Вирішення цієї проблеми може вимагати доказів з нульовим знанням про операції AES, що коштує дорого[2].

Крім того, оскільки кожна операція запису змінює все сховище на кожному сервері, навряд чи будуть часті резервні копії сховища, і такі атаки можуть зробити дані недоступними, для чесних користувачів, що впливає на доступність даних[1].

1.3 Атаки відкату в односерверних конструкціях.

Існує нездіснений результат, який говорить, що односерверні системи зберігання даних не можуть запропонувати гарантії цілісності проти шкідливого сервера, так як сервер завжди може надати певному користувачу стару версію сховища. Окрема система, чи інший сервер або блокчейн, [1] необхідно використовувати для відновлення таких гарантій цілісності. У Titanium ми запобігаємо атакам відкату шляхом аутентифікованого обміну секретами, який допускає, що до N_1 з N серверів є шкідливими. Тобто, коли один чесний сервер має тег аутентифікації сховища, то інші (зловмисні) сервери надають застарілі версії, перевірка цілісності не вийде з великою ймовірністю[1]. [1]

Атаки з вибірковими збоями дозволяють зловмиснику[2] дізнатися невелику кількість метаданих на основі того, чи сталася збій. Коли зловмисний сервер відхиляється від протоколу, він може спостерігати, чи отримує користувач правильні дані чи ні (за допомогою побічної інформації). Якщо користувач отримує неправильні дані і поводиться інакше, це вважається

помилкою. Відбувається збій чи ні, може залежати від метаданих, тому зловмисний сервер може дізнатися деякі метадані за допомогою цієї атаки. Зараз ми наведемо два приклади атак вибіркового збою в існуючих системах[2].

1.4 Огляд системи Titanium

Отже ми розглядаємо систему великою кількістю користувачів і N серверів з обміну файлами. Користувачам спільно сервери надають послуги зберігання. Тобто кожен користувач може зберігати певну кількість файлів на серверах і ділитися цими файлами з іншими користувачами згідно з деякими політиками контролю доступу. Кожен користувач може прочитати або записати файл, на який користувач має дозвіл. [2]

Як показано на рис. 4, щоб користувач міг виконати операцію в Titanium, він спочатку робить запит API до серверів. Запит знаходиться в секретних спільних ресурсах, тому жодні сервери не можуть відновити те, що в запиті. N-сервери, одержуючи запит користувача в секретних ресурсах, пересилають запити на проксі-сервер. Проксі-сервер не є окремою сутністю в системі, а є програмою, що виконується в безпечних обчисленнях N серверами. Оскільки стан проксі прихований від серверів, ми представляємо його як окрему частину для зручності представлення. Проксі-сервер перевіряє (при безпечному обчисленні), чи має користувач дозвіл на виконання дії, і якщо так, взаємодіє зі сховищем від імені користувача, наприклад, читання файлу або запис у файл.

Далі проксі-сервер надсилає відповідь API користувачеві та просить сервери переслати відповідь. Відповідь пересилається також у секретних ресурсах, тому будь-які сервери не бачать, що міститься у відповіді. Користувач відновлює відповідь проксі з секретних спільних ресурсів, що завершує виклик API в Titanium.

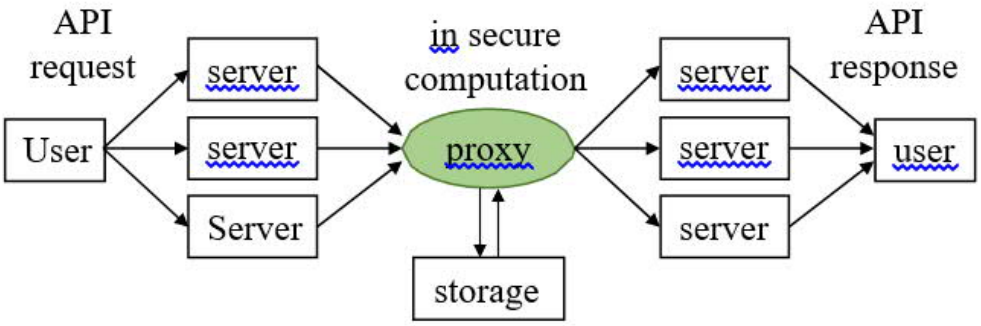


Рис. 4 Робочий процес Titanium.

1.5 Властивості приховування метаданих.

Titanium пропонує всі гарантії приховування метаданих, передбачені в існуючих роботах, змодельовані таким чином. Позначимо супротивника, який контролює всі корумповані сторони. Нехай U буде чесним користувачем, який має доступ до файлу F.

Конфіденційність і цілісність даних. [5] Якщо U ніколи не надавав дозвіл на читання чи запис F зловмиснику, пошкодженому A, то A не дізнається нічого про F і не модифікує F.

Читайте забуття. не може розрізнити, який блок даних прочитав U , навіть якщо має повний дозвіл на все сховище. Тобто не можна відрізнити операцію читання від іншої операції читання, виконаної іншим чесним користувачем, до іншого файлу.

Пишіть несвідомість. Якщо U ніколи не надавав дозвіл на читання F зловмиснику, пошкодженому , то не зрозуміє, чи F оновлено. Але якщо хтось із пошкоджених користувачів отримав дозвіл на читання до F , вони законно дізнаються, що F було змінено, але якщо принаймні двоє чесних користувачів мають дозвіл на запис до F , не знає, хто його змінив[5].

Нерозрізнення читання/запису. Якщо ніхто в не має дозволу на читання на F, то не знає, читає чи записує чесний користувач F.

Анонімність. не може розрізнити, який чесний користувач зробив запити на доступ, якщо чесні користувачі [3] спілкувалися з серверами таким чином, що приховували інформацію про мережу.

1.6 Висновок до розділу 1

У цій роботі ми робимо такі припущення: DoS-атаки виходять за межі сфери дії; розмір файлу фіксований, тому витоку розміру немає; всі запити обробляються в послідовному порядку. Це також стандартні припущення в попередній роботі.[3]

Є деякі рішення, щоб частково зняти ці припущення. Для пом’якшення DoS-атак з боку користувачів можна використовувати анонімні платежі або клієнтські головоломки (див. Додаток B). Наскільки нам відомо, для витоку розміру немає ефективного способу запобігти його без заповнення до найбільшого розміру файлу (що є надзвичайно дорогим), але є пом’якшення, включаючи часткове заповнення , диференційну конфіденційність, відкладений доступ шляхом завантаження різних фрагментів файлу. в різний час або доступ лише до потрібної частини файлу. Нарешті, наскільки нам відомо, незрозуміло, як увімкнути паралельний неусвідомлений доступ, не вимагаючи сильного припущення. Тому ми залишаємо це як проблему відкритого дослідження.

При цьому, за рахунок використання множини ключів KU, забезпечується функціональна можливість виявлення уповноважених користувачів, що внесли несанкціоновані зміни в методи. [2]

2. Методика застосування криптографічного рекурсивного 2-D контролю цілісності

2.1. Методика криптографічного рекурсивного 2-D контролю цілісності метаданих електронних документів на основі технології ланцюгового запису даних.

Методика є сукупністю моделей і алгоритмів криптографічного рекурсивного 2-D контролю цілісності метаданих ЕЛД , підлягає розробці на користь посадових осіб, що експлуатують АІС ЕД переважно відомчого призначення, і призначена для вирішення наступних завдань: [4]

- забезпечення криптографічного рекурсивного двовимірного контролю за цілісністю метаданих;
- локалізації модифікованих записів метаданих, в умовах навмисних впливів уповноважених користувачів;
- виявлення порушників (уповноважених користувачів);
- виключення змови довірених сторін рахунок введення взаємного контролю результатів їх дій.

Запропонована методика розроблена на основі технології ланцюгового запису даних, що представляє собою [5] динамічний реєстр. Процес формування такого реєстру можна умовно розділити на два етапи:

1) формування криптографічної рекурсивної 2-D послідовності метаданих;

2) перевірка криптографічної рекурсивної 2-D послідовності метаданих.

Основними умовами, які враховуються при розробці даної методики, є наступні припущення:

• засоби захисту інформації, функціонуючі в АІС ЕД, можуть мати вразливості, що сприяє їх використанню внутрішнім порушником у своїх цілях;

• внутрішнім порушником є може являтися адміністратор та/або уповноважений користувач, як у результаті навмисних дійств, так і внаслідок помилок, викликаних людським фактором;

• загрози цілісності метаданим можуть бути реалізовані як самим внутрішнім порушником, так і через впровадження ним шкідливого програмного забезпечення.

Обмеженням методики є можливість використання тільки трьох підмножин ключів багато -K_U Є {$K_U^{(1)}$,$K_U^{(2)}$,$K_U^{(3)}$ }ключи $K^{(1)}{}_U$ є внутрішніми системними ключами; ключі K_U^2 – зовнішніми ключами адміністратора системи; ключі K_U^3– зовнішні ключі оператора системи. [6]

Застосування методики дозволяє реалізувати криптографічний рекурсивний двовимірний контроль цілісності метаданих ЕЛД, оброблюваних АІС ЕД, з можливістю локалізації модифікованих записів метаданих, в умовах надмірних впливів уповноважених користувачів.

Відповідно до сутності розроблюваної методики, поставленими цілями, завданнями для ефективного [17]управління ЕЛД вона повинна застосовуватися протягом усього життєвого циклу.

Під ефективним управлінням Елд розуміється діяльність за своєчасним і повним забезпеченням всього життєвого циклу ЕЛД в АІС ЕД в відповідності з нормами та нормативами, встановленими керівної документації. На підставі цього, розроблювана методика також складається з чотирьох етапів, що відповідають вимогам управління ЕЛД.

Коротко опишемо етапи, що реалізуються у представленій методиці, співвідноси їх з етапами управління ЕЛД.

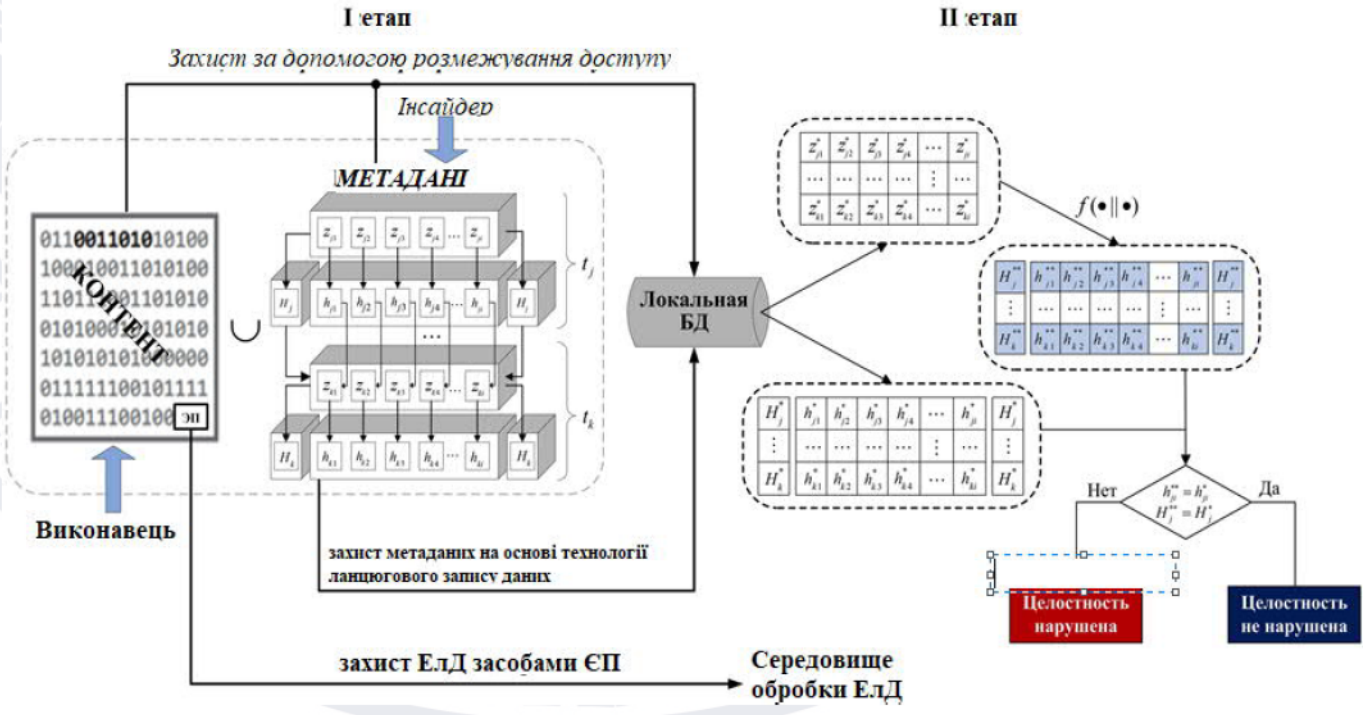
Eman 1. Створення ЕлД, початок використання методики.

Одночасно зі створенням контенту ЕлД формується порожній рядок Sctp. : яка заповнюється записами метаданих: z j1, , z jimax де кожен запис є реквізитом, який створюється ЕлД.

Якщо ЕлД виявився не затребуваним (втратив свою значущість), він передається для зберігання архів. Тоді застосування методики завершується. [23]

Етап 2. Формування криптографічної рекурсивної 2-D послідовності метаданих.

Якщо ЕлД затребуваний і продовжує циркулювати в АІС ЕД, тобто коригується у моменти часу. t_j $(j \in 1, \quad , k)$ задається умова: j = j + 1 на формування нового рядка. При цьому агенти з метою корекції записів метаданих використовують закриті ключі $k(q) \in \mathbf{K}(q) \quad (q \in 1, \quad , 3)$.



Мал. 5. Концептуальне представлення методики.

2.2. Результати застосування розробленої методики

Як випливає із загальної структури методики[7], за рахунок застосування криптографічного рекурсивного двовимірного контролю цілісності метаданих забезпечується локалізація модифікованих записів метаданих. Цю процедуру можна визначити як схеми, яка визначає загальний принцип використання методики.

У разі несанкціонованої модифікації запису метаданих z j2 ЕлД на етапі перевірки криптографічної рекурсивної 2-D послідовності метаданих після реалізації повторного криптографічного [18]перетворення та обчислення сигнатур записів метаданих фіксується зміна сигнатури hj2 запису метаданих z j2 що у свою чергу викликає зміну сигнатури даного рядка H j .

У зв'язку з тим, що дане технічне рішення засноване на загальних принципах побудови ланцюгового запису даних, модифікований запис метаданих викликає порушення цілісності сигнатур, як у стовпці даного реквізиту, так і в сигнатурах наступних рядків запису метаданих, що дозволяє відстежити весь ланцюг змін аж до моменту первинної модифікації запису. [7]

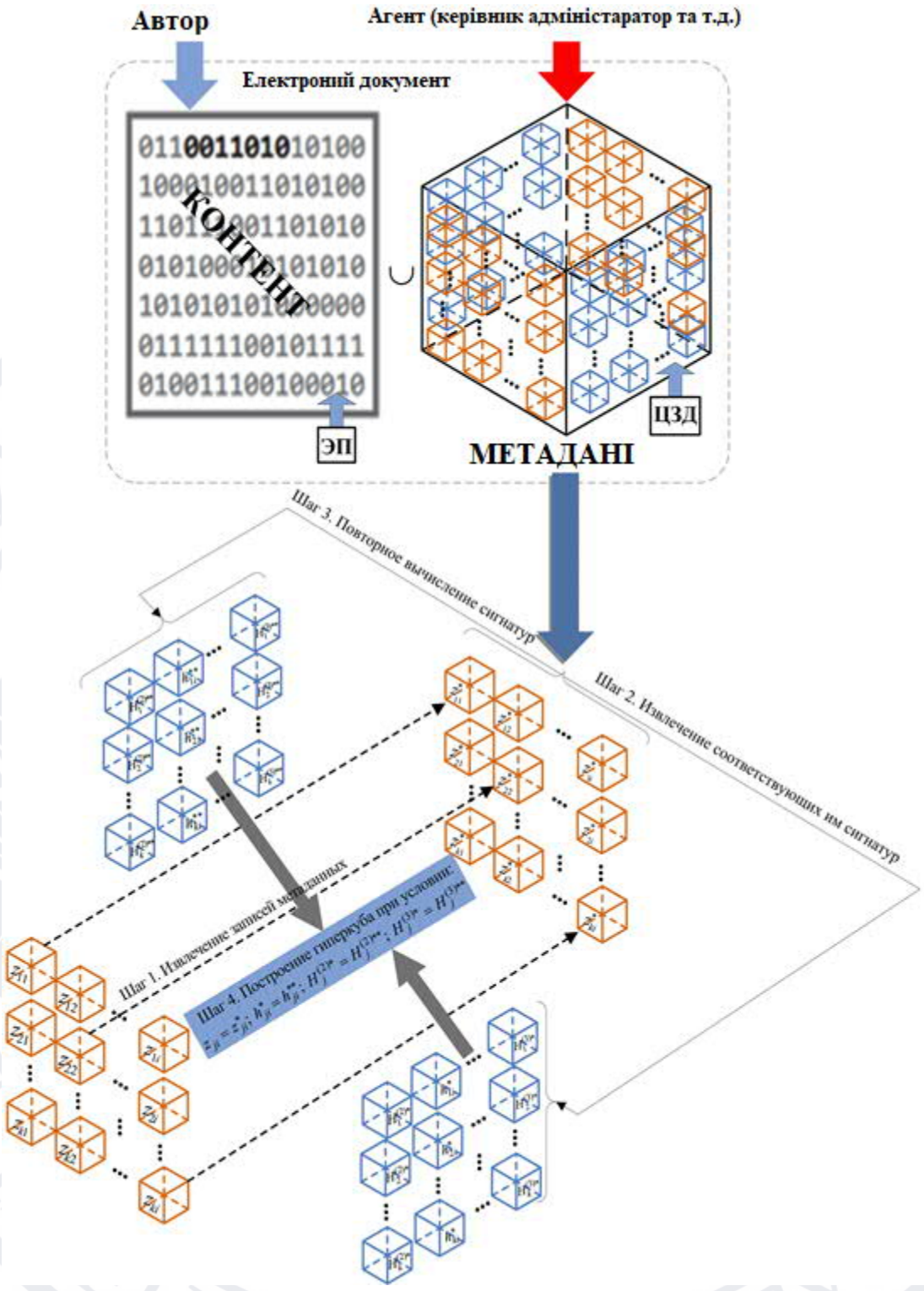


Рис.6 Перетворення структури ЕлД з урахуванням отриманих результатів

2.3 Висновок до розділу 2

Контроль цілісності метаданих розглянутий як складова частина процесу з управління ЕЛД у рамках вживання належних заходів щодо захисту документа в умовах обстановки, що змінюється в часі. Використання принципу ланцюгового запису даних у вирішенні поставлених завдань дозволило забезпечити технічні можливості локалізації несанкціоновано модифікованих записів метаданих, а також організувати взаємний контроль над діями уповноважених користувачів АІС ЕД. [7]

Рішення, запропоноване в цій статті, є логічним продовженням раніше виконаних авторами досліджень у галузі аналізу та синтезу перспективних систем юридично значущого електронного документообігу та орієнтоване на реалізацію, переважно, у відомчих АІС ЕД.

Отримані результати, у поєднанні з багатовимірним поданням даних, дозволяють надійно захистити метадані, забезпечивши при цьому ефективність управління ЕЛД, що обробляється АІС ЕД. У представленому рішенні ланцюговий запис даних є «надбудовою» над класичною базою даних, у ролі якої виступають метадані, представлені у вигляді багатовимірної моделі. Такий підхід дозволить регламентувати порядок передставлення інформації, [24] що зберігається в базі даних, ефективно використовувати розроблений механізм контролю цілісності метаданих, а також визначити порядок внесення, фіксації та відстеження змін.

3. Захист метаданих за допомогою стеганографії

3.1 Класифікація методів комп'ютерної стеганографії

У сучасній стеганографії, в цілому, можна виділити 2 напрямки: технологічну стеганографію та інформаційну стеганографію (рис. 7).

Рис. 7. Классификация методов стеганографической защиты

Стеганографічні методи в їх проекції на інструментарій і середу, які реалізуються на основі комп'ютерної техніки та програмного забезпечення в рамках окремих вичислильних або керуючих систем, корпоративних або глобальних вичислильних систем, складають предмет вивчення спільного наукового напряму інформаційної безпеки — комп'ютерної стеганографії. [38]

У рамках комп’ютерної стеганографії розглядаються питання, пов’язані із створенням інформації, яка зберігається на носіях або передається по сетям

телекомунікацій, з організацією скритих каналів у комп'ютерних системах і сетях, а також із технологіями цифрових водяних знаків та відображення пальця. [15]

При використанні методів комп'ютерної стеганографії необхідно враховувати наступні умови:

• противник може мати повне представлення про стеганографічну систему та деталі її реалізації. Єдиною інформацією, яка повинна залишатися йому невідомою, — це ключ, за допомогою якого можна встановити факт присутності скрытого повідомлення та його зміст; [17]

• якщо противнику яким-то чином вдалося дізнатися про факти існування скрытого повідомлення, то це не дозволить йому отримати подібні повідомлення з інших стеганограм до тех пор, поки ключ зберігається в тайне;

• потенційний противник повинен бути позбавленим будь-яких технічних і іних переваг у розпізнаванні або розкритті змісту таємних повідомлень. [16]

Для методів комп'ютерної стеганографії можна ввести певну класифікацію (рис. 8).

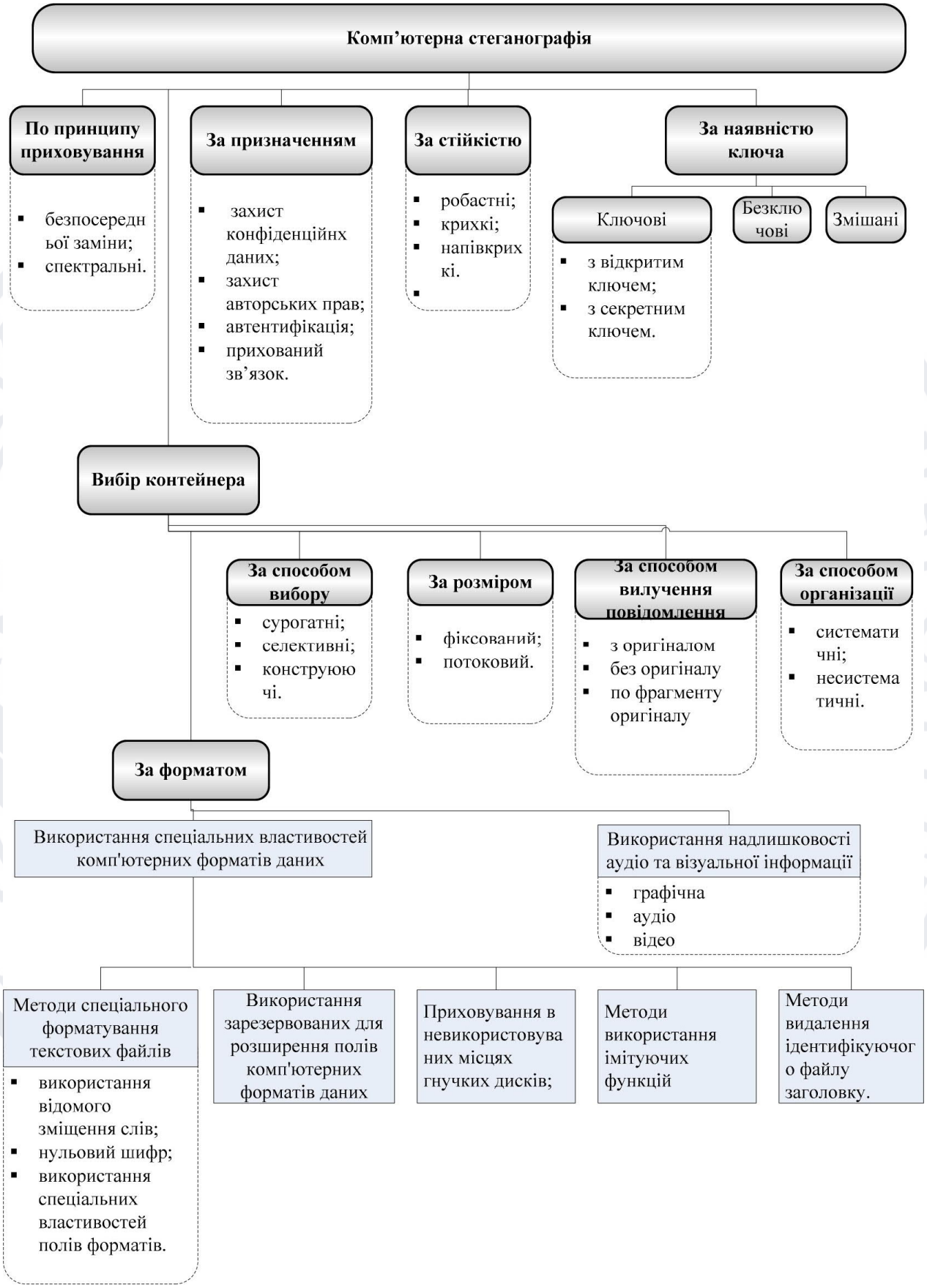


Рис. 8. Класифікація методів розміщення інформації

По способу відбору контейнера, як уже вказується, розрізняють методи суррогатної стеганографії, селективної стеганографії та конструюючої стеганографії. [37]

У методах суррогатної (безальтернативної) стеганографії відсутня можливість вибору контейнера та для розміщення повідомлень вибирається перший попавший контейнер, який не зовсім підходить для вбудовуваного повідомлення. [29] У цьому випадку битами контейнера замінюються бітами скриваемого повідомлення таким чином, щоб це змінити не було помітним. Основним недоліком методу є те, що він дозволяє скривати лише незначуще кількість даних.

У методах селективної стеганографії передбачається, що спрятане повідомлення повинно виробляти спеціальні статистичні характеристики шуму контейнера. Для цього генерують велике число альтернативних контейнерів, [37]щоб потім вибрати найбільш підходящий з них для конкретного повідомлення. Частим випадком такого підходу є вилучення некоторої хеш-функції для кожного контейнера. При цьому для розміщення повідомлень вибирається той контейнер, хеш-функції якого совпадає зі значенням хеш-функцій повідомлення (т.е. стеганограмма є вибраним контейнером). [38]

В методах конструюючої стеганографії контейнер генерується самою стегосистемою. Тут може бути кілька варіантів реалізації. Так, наприклад, шум контейнера може моделюватися скриваемим повідомленням. Це реалізується за допомогою процедури, які не тільки кодують скриваемое повідомлення під шум, але і зберігають модель початкового шуму. У переважному випадку по моделі шума може побудуватися ціле повідомлення. Приклади можуть служити методом, який реалізується в програмі MandelSteg, де в якості контейнера [32] для вбудовування повідомлень генерується фрактал Мандельброта, або ж апарат функцій імітації (mumic function).

По способу доступу до скритої інформації розрізняють методи для потокових (непреривних) контейнерів і методи для контейнерів із вільним доступом (обмеженою довжиною).

Методи, які використовують потокові контейнери, працюють з потоками непреривних даних (наприклад, інтернет-телефонія). [41] У цьому випадку скриваемые біти необхідно в режимі реального часу включати в інформаційний поток. О потоковому контейнері неможливо попередньо сказати, коли він начнеться, коли закінчиться і наскільки продовженим він буде. Більш того, об'єктивно немає можливості дізнатися заздалегідь, якими будуть післядувальні шумові біти. [24]Існує цілий ряд труднощів, які необхідно подолати кореспондентам при використанні потокових контейнерів. Найбільша проблема при цьому створює синхронізацію початку скрытого повідомлення.

Методи, які використовуються для контейнерів із вільним доступом, призначені для роботи з файлами фіксованої довжини (текстова інформація, програми, графічні чи звукові файли). У цьому випадку є відомі розміри файлу та його вміст. Скриваються біти можуть бути рівномірно вибрані за допомогою відповідної псевдовипадкової функції. [17]Недостаток таких контейнерів міститься в тому, вони мають низькі розміри, чим потокові, а також то, що розставання між скриваються бітами рівномірні розподіли між найбільш короткими та найбільш довгими заданими розставаннями, у той час, як істинний шум матиме експоненційне розподілення протягом тривалості інтервалу. Преимущество подібних контейнерів міститься в тому, щоб вони могли бути заздалегідь оцінені з точки зору ефективності вибраного стеганографічного перетворення[42].

По типу організації контейнерів, подібних помехозащищенним кодам, можуть бути систематичними та несистематичними. У систематично організованих контейнерах можна вказати конкретні місця стеганограми, де знаходяться інформаційні біти самого контейнера, а де — шумові біти, призначені для сховування інформації (як, наприклад, в широкому методі

найменшого значущого біта). [15] При несистематичній організації контейнера такого розділення зробити неможливо. У цьому випадку для виділення скритої інформації необхідно обробляти вміст усієї стеганограми.

За використовуваним принципом стеганометоди можна розбити на два класи: цифрові методи та структурні методи. [17]Якщо цифрові методи стеганографії, в основному, маніпулюють з цифровим представленням елементів засобів, куда внедряються скриваються дані [19] (наприклад, у пікселях, у різних коефіцієнтах косинус-косинусних перетворених, перетворених Фур'є, Уолша-Ра¬де¬ма¬хе ¬ра або Лапласа), то структурні методи стеганографії для покриття даних використовують семантично значимі структурні елементи інформаційної середовища.

Основним направленням комп'ютерної стеганографії є використання властивостей інформаційної середовища. Слідує участь, що при скритності інформації відбувається пошук деяких статистичних властивостей засобів або порушення її структури, які необхідно враховувати для зменшення демаскуючих признаків.

Також можна виділити методи, які використовують[28] спеціальні властивості представлення файлів:

- зарезервовані для розширення поля комп'ютерних форматів файлів, такі зазвичай заповнюються нулями та не враховуються програмою:
- спеціальне форматування даних;
- використання незадіяних місць на магнитних носителях;
- видалення ідентифікаторів заголовків для файлів.

В основному, для таких методів характерні низька ступінь скрытності, низька пропускна здатність і слабка продуктивність.

По призначенню розрізняють [25]стеганографічні методи власного для прихованого передачі або скритого зберігання даних і методи для закриття даних у цифрових об'єктах з метою захисту самих цифрових об'єктів.

По типу інформаційних засобів виділяються стеганографічні методи для текстових засобів, для аудіо-засобів, а також для зображень (стоп-кадрів) і відеозасобів.

3.2ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

При вирішенні проблеми збереження таємниці листування автоматизація алгоритму значно економить час користувача. Розробивши спеціальний додаток, створюються умови для приховування кореспонденції шляхом надсилання фотографій.

Приховану інформацію можна надіслати, наприклад,[33] за допомогою вітальної листівки. Нікому неможливо візуально визначити факт прихованої інформації.

- Запропонований алгоритм роботи модуля можна описати наступним чином:
- Перевірка наявності графічного зазначення в листі;
- Перевірка вкладення на наявність алгоритму стеганопідпису;
- Інформування користувача про наявність посилки;
- Отримання повідомлення.

Як і програма Microsoft Outlook, браузерні програми можуть використовувати стеганографію для:

- Завантаження файлів цифрових зображень на вебресурс.
- Завантаження файлів цифрових зображень з вебресурсу.
- Використання вебінтерфейсу для надсилання цифрових файлів зображень.

Для вирішення цих проблем пропонується використовувати розроблену програму, яка вбудована в браузер.

Коли користувач знаходить нову фотографію, він завантажує її на свій комп’ютер за допомогою веб-браузера або іншого програмного забезпечення. Після цього за допомогою алгоритму видаляється прихована інформація із зображення.

Ця процедура дозволяє створити механізм прихованого обміну повідомленнями з цифровими фотографіями. Зовнішні спостерігачі сприймуть це як простий обмін фотографіями.

Для правильної роботи даного програмного продукту до складу технічних засобів повинні входити[34]:

а) комп’ютер з такою конфігурацією:

процесор з тактовою частотою не нижче 2 ГГц;

дискова підсистема - 100 Гб;

достатній об’єм оперативної пам’яті (не менше 1 ГБ);

б) додатково має бути встановлене таке програмне забезпечення:

операційна система Windows XP/Vista/Windows7/Windows 10 або ін.;

.Net Framework 3.5 і вище;

Інтерфейсний рівень - це рівень відображення даних на стороні клієнта.

Програма написана на мові C#.

Дані відображаються користувачеві за допомогою графічного інтерфейсу у вигляді Windows Forms .

Методи готові до використання. Напишемо просту візуалізацію на Windows Forms (рис 9).

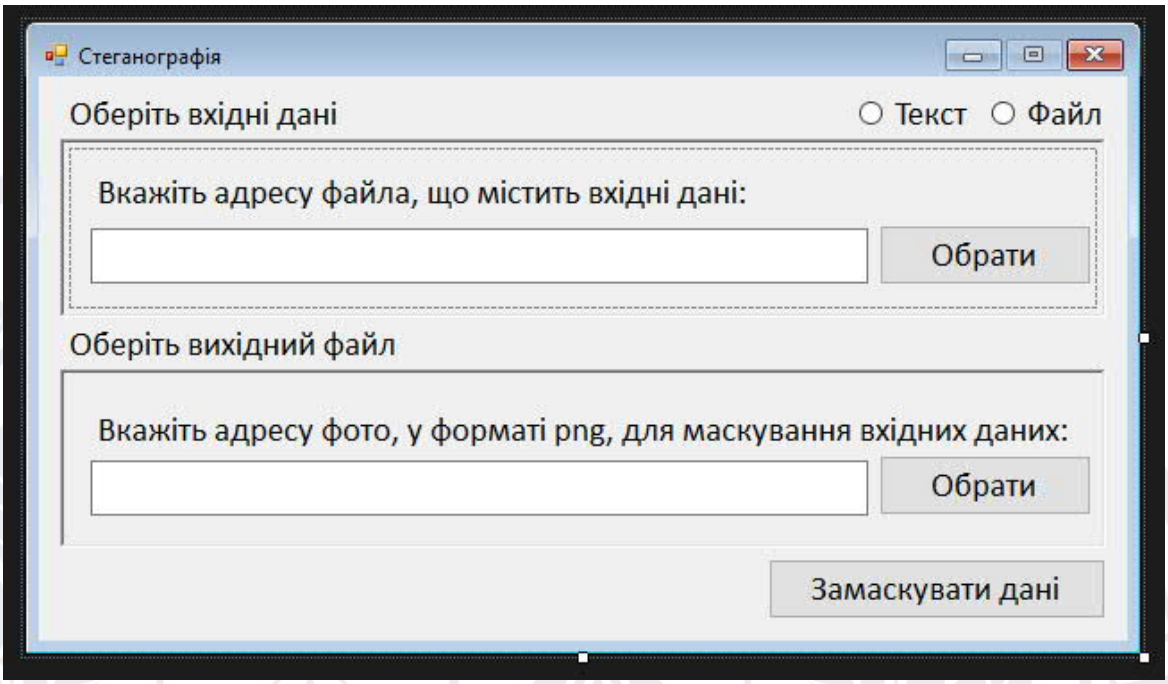


Рис 9– Форма кодування

3.2 Розробка додатка на основі стеганографічного методу

Для зображень формату BMP реалізований метод заміни найменш значущого біта. Цей метод дуже нестійкий до будь-яких спотворень контейнера, а також до його стиснення, що руйнує всю приховану інформацію. Крім того, введення інформації таким способом легко виявити за допомогою стеганографічних атак. Але перекодування до формату JPEG дозволяє подолати ці складності.

Для кращого захисту інформації, прихованої на зображенні методом заміни, можна використовувати криптографічні методи, які зашифрують повідомлення до його втручання, і, таким чином, інформація буде захищена різними рівнями безпеки, що приховає факт передачі повідомлення, а при виявленні розшифрувати його неможливо.

Поряд з криптографією можна застосовувати алгоритми вибору пікселю та його розташування для вбудовування в нього біта повідомлення відповідно до певного правила. Цей підхід, порівняно із послідовним вибором пікселів для

вбудовування інформації, захистить повідомлення від розшифровки, навіть якщо сторонні дані виявлені в контейнері, оскільки біти повідомлення будуть випадково видалені з зображення алгоритмом.

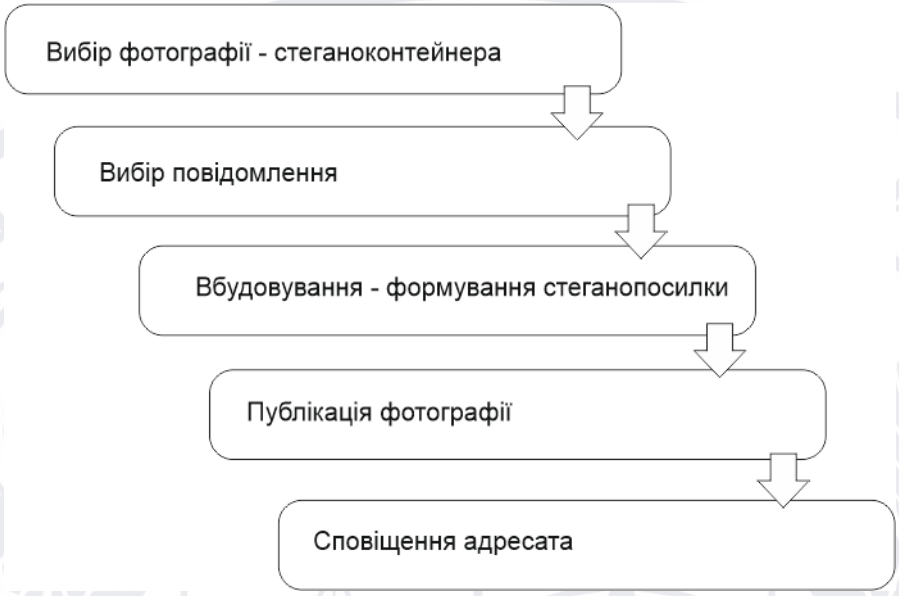


Рисунок 10 – Передача прихованих даних

Ця процедура дозволяє створити механізм прихованого обміну повідомленнями за допомогою цифрових фотографій. Зовнішні спостерігачі сприймуть це як простий обмін фотографіями.

Зі зміною розмірів справляються, ретельно підбираючи стеганоконтейнери. Помічаючи, до яких розмірів конкретні навантажувачі будуть перетворювати фотографії, підготуйте необхідні контейнери. На жаль, форматні методи зазвичай не вирішують проблему стійкості вбудованої інформації до стиснення. Це пов’язано з тим, що використовуючи маркери сегментів при впровадженні інформації, ми можемо розраховувати, що завантажувач знищить ці маркери без потреби, власне ще до дискретно косинусного перетворення [21].

Запропонований алгоритм впроваджує повідомлення в сегменти, що не зчитуються утилітами читання JPEG-зображень, приміром SOF2 - SOF10; DAC; COM; APP15 [34] чи інші[38].

Наступним кроком необхідно вказати текст повідомлення. Це робиться одним із двох можливих способів. Перший спосіб полягає у тому, що користувач власноруч набирає текст повідомлення із клавіатури, [34] в той час як другий спосіб вимагає наявності повідомлення в текстовому файлі формату .txt. Невелике повідомлення зручніше вказати власноруч, в той час як досить великі - просто загрузити із файлу. Також варто звертати увагу на розмір контейнера задля того, щоб приблизно розуміти максимально можливу довжину повідомлення, що буде оброблено алгоритмом та вкраплено в [40] контейнер.

Перед вбудовуванням повідомлення стискається, щоб зменшити гучність.

Далі йде контейнер JPEG:

Виконується аналіз поточної структури файлів. При цьому обов'язково виділяють межі сегментів;

Створено назву нового сегмента APP15;

Створений сегмент заповнюється байтами стисненого повідомлення;

До структури файлу JPEG додано новий сегмент.

Потім перетворене зображення надсилається на потрібну адресу, або перетворене зображення завантажується на вашу сторінку профілю facebook.com. У цьому випадку потрібно використовувати лише один завантажувач, оскільки він залишає вбудоване повідомлення недоторканим. А потім позначає це як область із зображенням одержувача. Одержувач, завдяки стандартній опції "нові фотографії зі мною", отримує повідомлення про наявність нового фото. Потім одержувач, відкривши сторінку з цільовим зображенням, зберігає цифрову фотографію. Ця копія обробляється запропонованим алгоритмом. Далі адресат має можливість ознайомиться з надісланою йому інформацією.

Під час тестування несправностей та недоліків у роботі програмного застосування комп’ютерної стеганографії для цифрових контейнерів у вигляді зображень його виявлено не було, що свідчить про високу якість розробки та можливість реалізації при потребі[33].

3.3. Висновки до розділу 3

В даному розділі розглянуто сфери застосування розробленого стеганоалгоритму.

Для кращого захисту інформації, прихованої на зображенні методом заміни, ви можете використовувати криптографічні методи, які шифрують повідомлення до його втручання, і таким чином інформація буде захищена різними рівнями захисту, що приховає факт передачі повідомлення.

Стеганосистема передачі секретної інформації є альтернативним рішенням разом із криптографічними засобами захисту каналів зв'язку (застосування зашифрованого протоколу SSL / TLS).

На жаль, методи форматування зазвичай не вирішують проблему стійкості вбудованої інформації до стиснення. Це пов’язано з тим, що при введенні інформації використовують маркери сегментів, оскільки може статися так, що навантажувач знищить ці маркери без потреби, фактично до дискретного косинусного перетворення.

ВИСНОВКИ

Контроль цілісності метаданих розглянутий як складова частина процесу з управління ЕлД у рамках вживання належних заходів щодо захисту документа в умовах обстановки, що змінюється в часі. Використання принципу ланцюгового запису даних у вирішенні поставлених завдань дозволило забезпечити технічні можливості локалізації несанкціоновано модифікованих записів метаданих, а також організувати взаємний контроль над діями уповноважених користувачів АІС ЕД.

Отримані результати, разом із багатовимірним поданням даних, дозволяють надійно захистити метадані, забезпечивши у своїй ефективність управління ЕлД, оброблюваними АИС ЭД. У представленому рішенні ланцюговий запис даних є «надбудовою» над класичною базою даних, у ролі якої виступають метадані, представлені у вигляді багатовимірної моделі. Такий підхід дозволить регламентувати порядок подання інформації, що зберігається в базі даних, ефективно використовувати розроблений механізм контролю цілісності метаданих, а також визначити порядок внесення, фіксації та відстеження змін.

Підсумком проведеного дослідження стала оцінка рівня захищеності метаданих ЕЛД, оброблюваних АІС ЕД, показником якого обрано ймовірність порушення їхньої цілісності. Порівняльний аналіз розробленого способу з традиційно використовуваною хеш-функцією, проведений на основі ЛВМ, показав загальний виграш у 67% при заданих припущеннях, що свідчить про успішне досягнення поставлених цілей

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://eprint.iacr.org/2022/051#:~:text=Titanium%20is%20a%20metadata%2Dhiding,faster%20or%20more%20than%20MCORAM.
2. Рябінін І.А., Соложєнцева О.Д., Карасьова В.В. Путівник з логіко-імовірнісного обчислення // Моделювання та аналіз безпеки та ризику у складних системах. Праці Міжнародної наукової школи МАБР-2016, 2016. С. 9-25.
3. Дьомін А.В. Логіко-імовірнісний метод управління модульними роботами // Системна інформатика. 2017. № 11. С. 61-80.
4. Талі Д.І., Фінько О.А., Єлісєєв Н.І., Діченко С.А., Барільченко С.А. Спосіб криптографічного рекурсивного 2-D контролю цілісності метаданих файлів електронних документів // Патент на винахід UA 2726930, опубл. 16.07.2020, бюл. №20.
5. Талі Д.І., Фінько О.А. Криптографічний рекурсивний контроль за цілісністю метаданих електронних документів. Частина 1. Математична модель // Питання кібербезпеки. 2020. №5 (39). З. 2-18. DOI: 10.21681/2311-3456-2020-05-2-18
6. Талі Д.І., Фінько О.А. Криптографічний рекурсивний контроль за цілісністю метаданих електронних документів. Частина 2.
7. Комплекс алгоритмів// Питання кібербезпеки. 2020. № 6 (40). З. 32-47. DOI: 10.21681/2311-3456-2020-06-32-47
8. Талі Д.І. Модель загроз безпеці метаданим у системі електронного документообігу військового призначення // Питання оборонної техніки Серія 16 Технічні засоби протидії тероризму. 2020. № 139-140. С. 95-101.
9. Hartmann K., Giles K. UAV exploitation: New domain for cyber power // 8th International Conference on Cyber Conflict (CyCon). 2016. Pp. 205-221.
10. Тарасов С.В. СУБД для програміста Бази даних ізсередини. М: СОЛОН-Прес, 2015. 320 с.

11. Діченко С.А., Фінько О.А. Узагальнений спосіб застосування хеш-функції для контролю цілісності даних // Наукоємні технології в космічних дослідженнях землі. 2020. Т. 12. № 6. С. 48-59. DOI: 10.36724/2409-5419-2020-12-6-48-59
12. Pune, M.; Open Source Intelligence (OSINT). Market Research Report-Global Forecast to 2023—Market Analysis, Scope, Stake, Progress, Trends and Forecast to 2023. Market Research Future. 2020. Available online: https://www.marketresearchfuture.com/reports/open-source-intelligence-market-4545
13. Pastorino, C. Técnicas y Herramientas OSINT Para la Investigación en Internet. Welivesecurity by ESET. 2019. Available online: https://www.welivesecurity.com/la-es/2019/10/07/tecnicas-herramientas-osint-invest igacion-internet/
14. Passi, H. Top. 10 Popular Open Source Intelligence (OSINT) Tools. GreyCampus. 2018. Available online: https://www.greycampus.com/blog/information-security/top-open-source-intelligence-tools
15. Gromov YU.YU., Ivanova O.G., Yakovlev A.V., Odnol'ko V.G. Upravleniye dannymi. – Tambov: «TGTU», 2015. 192 s.
16. Murray D.R., Newman M.B. Probability analyses of combining background concentrations with model-predicted concentrations // J. Air Waste Manag. Assoc. 2016, vol. 64, no. 3, pp. 248-254.
17. Dichenko S.A., Finko O.A. Gibridnyy kripto-kodovyy metod kontrolya i vosstanovleniya tselostnosti dannykh dlya zashchishchonnykh informatsionno-analiticheskikh sistem // Voprosy kiberbezopasnosti. 2019. № 6 (34). S. 7-14. DOI: 10.21681/2311-3456-2019-6- 17-36.
18. Yeliseyev N.I., Finko O.A. Teoreticheskiye aspekty razvitiya sistemy elektronnogo dokumentooborota Ministerstva oborony Rossiyskoy Federatsii // Voyennaya mysl'. 2015. № 7. S. 55-63

19. Yeliseyev N.I., Finko O.A. Upravleniye tselostnost'yu sistemy yuridicheski znachimogo elektronnogo dokumentooborota v usloviyakh mezhformatnykh preobrazovaniy elektronnykh dokumentov // Problemy upravleniya. 2014. № 3 S. 68-73.
20. Столлингс В. Криптография и защита сетей: теория и практика. М: Вильямс. 2001. Пер. с англ. - 235 с.
21. Стеганография в XXI веке. Цели. Практическое применение.
22. Актуальность [Електронний ресурс] / Режим доступу: https://habrahabr.ru/post/253045/
23. Стеганографический методКуттера-Джордана- Боссена [Електронний ресурс]. - 2018. - Режим доступудо ресурсу:https://habr.com/ru/post/115287/.
24. Стасюк О. І. та ін. Сучасні стеганографічні методи захисту інформації / О. І. Стасюк та ін. // Захист інформації. — 2011. — Т. 13. — № 1 (50).
25. Тарасов Д. О. Класифікація та аналіз безкоштовних програмних засобів стеганографії / Д. О. Тарасов, А. С. Мельник, М. М. Голобородько // Інформаційні системи та мережі. Вісник НУ «Львівська політехніка». — 2010. —№ 673. — С. 365-374.
26. Юдін О. К. Аналіз стеганографічних методів приховування інформаційних потоків у контейнери різних форматів / О. К. Юдін, Р. В. Зюбіна, О. В. Фролов // Радиоэлектроника и информатика. — Х. : НХНУРЕ, 2015. — №— С. 24-31.
27. Cox I.J., Miller M., Bloom J., Fridrich J., Kalker T. Digital watermarking and steganography. [Текст] / Morgan Kaufmann, 2007 p. - 593 c.
28. Abolghasemi M. LSB data hiding detection based on gray level co-occurrence matrix / M. Abolghasemi, H. Aghainia, K. Faez, M.A. Mehrabi // International symposium on Telecommunications. - 2008. - P. 656-659.
29. Cvejic N. Algorithms for audio watermarking and steganography. [Текст]

30. / Academic dissertation, Department of Electrical and Information Engineering, Information Processing Laboratory // University of Oulu, 2004 p. - 111 c.
31. Data Hiding using Graphical Code based Steganography Technique [Електроний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/282403473 Data Hiding using Graphical Code based Steganography Technique.
32. Geetha S. Audio steganalysis with Hausdorff distance higher order statistics using a rule based decision tree paradigm / S. Geetha, N. Ishwarya, N. Kamaraj // Expert system with applications journal. - 2010. - Vol. 37, № 12. - P. 7469-7482
33. Husrev T. Sencar. Data Hiding Fundamentals And Applications / Husrev
34. T. Sencar, Mahalingam Ramkumar, Ali N. Akansu // Digital Multimedia. ELSEVIER science and technology books. - 2014. - 364 p.
35. Implementation of LSB Steganography and its Evaluation for Various File Formats [Електронний ресурс] - Режим доступу до ресурсу: https://pdfs.semanticscholar.org/3dce/b6307cee042b687b7f377ec1d5de91ce20b0.pdf
36. Liu Y. A novel audio steganalysis based on higher-order statistics of a distortion measure with Hausdorff distance / Y. Liu, K. Chiang, C. Corbett, R. Archibald, B. Mukherjee, D. Ghosal // Lecture Notes in Computer Science. - 2008. - № 5222. - P. 487 -501.
37. LSB стеганография [Електроний ресурс]. - 2019. - Режим доступу до ресурсу: https://habr.eom/ru/post/112976/.
38. Niimi M. An attack to BPCS-steganography using complexity histogram and countermeasure / M. Niimi, T. Ei, H. Noda, E. Kawaguchi, B. Segee // Proc. International Conf. on Image Processing, ICIP. - 2004. - Vol.5. - P.733-736.
39. Ru X. Audio steganalysis based on “negative resonance phenomenon” caused by steganographic tools / X. Ru, Y. Zhuang, F. Wu // Journal of Zhejiang University SCIENCE A. - 2016. - Vol.7, № 4. - P. 577-583.

40. Steganography and Digital Watermarking: a global view [Електроний ресурс]- Режим доступу до ресурсу: http://lia.deis.unibo.it/Courses/RetiDiCalcolatori/Progetti00/fortini/proiect.pdf.
41. Steganalysis of Information Hidden in Webpage Based on Higher-order Statistics [Електронний ресурс] - Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/4606211.
42. Sending hidden data through www pages: detection and prevention [Електроний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/26807670 Sending hidden data through
43. www pages detection and prevention.
44. Wu C.-P. Robust and efficient digital audio watermarking using audio content analysis / C.-P. Wu, P.-C. Su, C.-C.J. Kuo // Proc. of SPIE Electronic Imaging, Security and Watermarking of Multimedia Contents II. - 2000. - Vol. 3971. - P. 382- 392.
45. Wu C.-P. Robust audio watermarking for copyright protection / C.-P. Wu, P.-C. Su, K.C.-C. Jay // Proceedings of SPIE, Advanced Signal Processing Algorithms, Architectures, and Implementations IX. - 1999.- Vol. 3807. - P. 387-397.
46. Xiang S. Robust Audio Watermarking Against the D/A and A/D conversions [Електроний ресурс] / S. Xiang, J. Huang. - 2017. - 29 p. - Режим доступу: http://arxiv.org/ftp/arxiv/papers/0707/0707.0397.pdf.
47. Yang S. Quantization-Based Digital Audio Watermar-king in Discrete Fourier Transform Domain / S. Yang, W. Tan, Y. Chen, W. Ma // Journal of Multimedia. - 2010. - Vol. 5, № 2. - P. 151-158

Додаток А

Фрагмент програмного забезпечення реалізації стеганографічного алгоритму

```
class Stegano
{
  Bitmap photo;   // Інкапсульовані змінні: зображення та дані byte[]
  openData;

  // GET-метод для отримання зображення як масиву байт public byte[]
  GetPhotoBytes()
  {
    using (var ms = new MemoryStream())
    {
      photo.Save(ms, photo.RawFormat); return ms.ToArray();
    }
  }

  // GET-метод для отримання повідомлення public byte[] GetOpenData()
  {
    if (openData == null)
    {
      throw new Exception("Дані не задано");
    }
```

```
        else
        {
          return openData;
        }// Конструктор класу для закодування public Stegano(byte[] openData,
        Image photo)
      {
        this.photo = (Bitmap)photo; this.openData = openData;
      }

      // Конструктор класу для розкодування public Stegano(Image photo)
      {
        this.photo = (Bitmap)photo; this.openData = null;
      }
    }

  private Bitmap ClearPhoto(Bitmap photo)
      {
        byte[] pixels = new byte[photo.Width * photo.Height]; int c = 0;
        for (int i = 0; i < photo.Height; i++)
        {
          for (int j = 0; j < photo.Width; j++, c++)
          {
```

```
            pixels[c] = photo.GetPixel(j, i).B; if ((pixels[c] % 3) != 0)
            {
              int newB = pixels[c] + 1; if ((newB % 3) != 0)
                newB++;
              if (newB > 255) newB -= 3;

              photo.SetPixel(j, i, Color.FromArgb( photo.GetPixel(j, i).R,
                photo.GetPixel(j, i).G, newB
                ));
            }
          }
        }
        return photo;
        }

    public bool EncodeData (int tolerance)
{
          // Перевіряємо на коректність дані
    Exception nullPhoto = new Exception("Розмір цільового фото не достатній");


    if (this.photo == null)
```

```
  throw new Exception("Цільове фото не задано"); if (this.openData == null ||
this.openData.Length == 0)

  throw new Exception("Відсутня вхідна інформація");

if ((this.photo.Width * this.photo.Height) / 2 < this.openData.Length) throw
  nullPhoto;

        // Проводимо підготовку контейнера до запису Bitmap clearPhoto =
ClearPhoto1((Bitmap)this.photo);

int currentPoint = 0;

byte[,] pixels = new byte[clearPhoto.Width * clearPhoto.Height, 2]; int c = 0;

        // Запускаємо цикл по пікселях зображення for (int i = 0; i <
clearPhoto.Height; i++)

{

for (int j = 0; j < clearPhoto.Width; j++, c++)

  {

    pixels[c, 0] = clearPhoto.GetPixel(j, i).R; pixels[c, 1] =
    clearPhoto.GetPixel(j, i).G;


    int t1_ = (pixels[c, 0] - this.openData[currentPoint]); if (t1_ < 0)

      t1_ = -t1_;


    int t2_ = (pixels[c, 1] - this.openData[currentPoint]); if (t2_ < 0)

      t2_ = -t2_;
```

```
if ((t1_ < tolerance) || (t2_ < tolerance))
{
  byte r_ = (byte)t1; byte g_ = (byte)t2;
  int b_ = clearPhoto.GetPixel(j, i).B; if (r_ < g_) // B червоний
  {
    b_++;
    if (b_ > 255) b_ -= 3;

    clearPhoto.SetPixel(j, i, Color.FromArgb( this.openData[currentPoint],
      pixels[c, 1], b_
      ));
  }
  else // або в зелений
  {
    b_ += 2;
    if (b_ > 255)8 b_ -= 3;
    clearPhoto.SetPixel(j, i, Color.FromArgb( pixels[c, 0],
      this.openData[currentPoint],
      b_
      ));
  }
  currentPoint++;
```

```
            if (currentPoint == this.openData.Length)
            {// Якщо всі необхідні байти вже зашифровано – закінчуємо роботу
              this.photo = clearPhoto;
              return true;
            }
          }
        }
      }
      throw nullPhoto;
    }
    public void DecodeData()
        {
    // Перевіряємо на коректність дані
          Exception nullPhoto = new Exception("Розмір цільового фото не
          достатній");

          if (this.photo == null)
            throw new Exception("Цільове фото не задано");

          byte[] data = new byte[0]; int c = 0;

          Bitmap photo = new Bitmap(this.photo);
```

```
Color[] pixels = new Color[photo.Height * photo.Width];
// Запускаємо цикл по пікселях зображення for (int i = 0; i <
this.photo.Height; i++)
{
  for (int j = 0; j < this.photo.Width; j++, c++)
  {
    pixels[c] = photo.GetPixel(j, i);

    if ((pixels[c].B % 3) != 0)
    {
      Array.Resize(ref data, data.Length + 1); if (pixels[c].B % 3 == 1)
      {
        data[data.Length - 1] = photo.GetPixel(j, i).R;
      }
      else
      {
        data[data.Length - 1] = photo.GetPixel(j, i).G;
      }
    }
  }
}
this.openData = data; return;
```

```
        }

using System;
using System.Drawing; using System.IO;
using System.Text;
using System.Windows.Forms;

namespace Стеганографія
{
  public partial class Form1 : Form
  {
    bool isText = true;
    public Form1()
    {
      InitializeComponent();
    }

    private void RadioButton1_CheckedChanged(object sender, EventArgs e)
    {
      panel2.Visible = true; panel3.Visible = false; isText = true;
    }
```

```
private void RadioButton2_CheckedChanged(object sender, EventArgs e)
{
  panel3.Visible = true; panel2.Visible = false; isText = false;
}

private void Button1_Click(object sender, EventArgs e)
{
  string path;

  OpenFileDialog fDialog = new OpenFileDialog(); fDialog.Title = "Оберіть
  файл";
  fDialog.Filter = "Всі формати|*.*"; fDialog.InitialDirectory = @"D:\";
  if (fDialog.ShowDialog() == DialogResult.OK)
  {
    path = fDialog.FileName.ToString();
  }
  else
    return; textBox2.Text = path;
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
  this.MaximumSize = new Size(1500, 355); this.MinimumSize = new Size(625,
  355);
}

private void Button2_Click(object sender, EventArgs e)
{
  string path;

  OpenFileDialog fDialog = new OpenFileDialog(); fDialog.Title = "Оберіть
  файл";
  fDialog.Filter = "PNG зображення|*.png"; fDialog.InitialDirectory = @"D:\";
  if (fDialog.ShowDialog() == DialogResult.OK)
  {
    path = fDialog.FileName.ToString();
  }
  else
    return; textBox3.Text = path;
}
```

```
private void Button3_Click(object sender, EventArgs e)
{
  try
  {
    byte[] data; if (isText)
    {
      data = Encoding.UTF8.GetBytes(textBox1.Text); Array.Resize(ref data,
      data.Length + 1);
      data[data.Length - 1] = 125;
    }
    else
    {
      using (FileStream fstream = File.OpenRead(textBox2.Text))
      {
        data = new byte[fstream.Length]; fstream.Read(data, 0, data.Length);
      }
      string fName = Path.GetFileName(textBox2.Text); byte[] fN =
      Encoding.UTF8.GetBytes(fName); byte[] ln =
      BitConverter.GetBytes(fN.Length);
```

```
  Array.Resize(ref data, data.Length + fN.Length); Array.Copy(fN, 0, data,
  data.Length - fN.Length, fN.Length); Array.Resize(ref data, data.Length +
  4);
  Array.Copy(ln, 0, data, data.Length - 4, 4);

  Array.Resize(ref data, data.Length + 1); data[data.Length - 1] = 126;
}

byte[] arch_data = Module1723.ArchivSimple_8bit(data, true);

if (arch_data.Length < data.Length)
{
  data = arch_data;
  Array.Resize(ref data, data.Length + 1); data[data.Length - 1] = 141;
}
else
{
  Array.Resize(ref data, data.Length + 1); data[data.Length - 1] = 123;
}
```

```
Stegano stegano;


using(Image img = Image.FromFile(textBox3.Text))
{
  stegano = new Stegano(data, Image.FromFile(textBox3.Text));
}


bool corr = false; try
{
  corr = stegano.EncodeData(5);
}
catch (Exception)
{
  try
  {
    corr = stegano.EncodeData(10);
  }
  catch (Exception)
  {
```

```
    try
    {
      corr = stegano.EncodeData(15);
    }
    catch (Exception)
    {
      try
      {
        corr = stegano.EncodeData(25);
      }
      catch (Exception ex)
      {
        MessageBox.Show(ex.Message, "Помилка"); return;
      }
    }
  }
}
if (corr)
{
  string kyda;
  FolderBrowserDialog fDialog1 = new FolderBrowserDialog(); if
  (fDialog1.ShowDialog() == DialogResult.OK)
```

```
        {
          kyda = fDialog1.SelectedPath.ToString();
        }
        else
          return;

        kyda += "\\" + NameGenerate(kyda);

        data = stegano.GetPhotoBytes();

        using (FileStream fstream = new FileStream(kyda, FileMode.Create))
        {
          fstream.Write(data, 0, data.Length);
        }
      }
    }
    catch (Exception ex)
    {
      MessageBox.Show(ex.Message,   "Помилка");
    }
  }
```

```
        private string NameGenerate(string path)
        {
          string name = "crypt.png";
          if (File.Exists(path + "\\" + name))
          {
            int i = 1;
            while (System.IO.File.Exists(path + "\\" + name))
            {
              name = "crypt(" + i + ").png"; i++;
            }
            return name;
          }
          else
          {
            return name;
          }
        }
    }
}
```

Код форми декодування:

```
using System;
```

```
using System.Drawing; using System.IO;
using System.Text;
using System.Windows.Forms; namespace Стеганографія
{
  public partial class Form2 : Form
  {
    public Form2()
    {
      InitializeComponent();
    }

    private void Button1_Click(object sender, EventArgs e)
    {
      string path;

      OpenFileDialog fDialog = new OpenFileDialog(); fDialog.Title = "Оберіть
      файл";
      fDialog.Filter = "PNG зображення|*.png"; fDialog.InitialDirectory = @"D:\";
      if (fDialog.ShowDialog() == DialogResult.OK)
      {
        path = fDialog.FileName.ToString();
```

```
    }
    else
      return; textBox2.Text = path;
  }

  private void Form2_Load(object sender, EventArgs e)
  {
    this.MaximumSize = new Size(1500, 375); this.MinimumSize = new Size(625,
    375);
  }

  private void Button3_Click(object sender, EventArgs e)
  {
    Exception errorDecode = new Exception("Помилка розкодування. Файл з
    даними пошкоджено."); try
    {
      byte[] data;
      Stegano stegano;

      using (Image img = Image.FromFile(textBox2.Text))
      {
```

```
  stegano = new Stegano(Image.FromFile(textBox2.Text));
}

stegano.DecodeData();

data = stegano.GetOpenData();

if (data[data.Length - 1] == 141)
{
  data = Module1723.ArchivSimple_8bit(data, false);
}
else if (data[data.Length - 1] != 123)
{
  throw errorDecode;
}

Array.Resize(ref data, data.Length - 1);

if (data[data.Length - 1] == 126)
```

```
{
  Array.Resize(ref data, data.Length - 1);
  int ln = BitConverter.ToInt32(data, data.Length - 4); Array.Resize(ref data,
  data.Length - 4);
  byte[] name = new byte[ln];
  Array.Copy(data, data.Length - ln, name, 0, ln); Array.Resize(ref data,
  data.Length - ln);
  string kyda;
  FolderBrowserDialog fDialog1 = new FolderBrowserDialog();
  if (fDialog1.ShowDialog() == DialogResult.OK)
  {
  kyda = fDialog1.SelectedPath.ToString();
  }
  else
  return;


  using (FileStream fstream = new FileStream(kyda + "\\" +
  Encoding.UTF8.GetString(name), FileMode.Create))
  {
  fstream.Write(data, 0, data.Length);
  }
  }
```

```
else if (data[data.Length - 1] == 125)
{
Array.Resize(ref data, data.Length - 1); textBox1.Text =
Encoding.UTF8.GetString(data);
}
else
{
throw errorDecode;
}


}
catch (Exception ex)
{
MessageBox.Show(ex.Message, "Помилка");
}
}
}
}
```