ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут інформаційних технологій
та інноваційного підприємництва
(повна назва інституту)

Кафедра технічної кібернетики й інформаційних технологій
ім. професора Р.В. Мерката
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної роботи

бакалавр
(освітньо-кваліфікаційний рівень)

на тему

Використання PDF файлів в якості стеганографічних контейнерів

Виконав:
здобувач вищої освіти 4 курсу, групи 1
Спеціальність:
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Запорожан — Артем ЗАПОРОЖАН
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник к.геогр.н., доцент
(вчене звання, посада)

(підпис) Юрій ДАУС
(Ім'я, ПРІЗВИЩЕ)

Одеса – 2026

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Інститут Навчально-науковий інститут інформаційних технологій та інноваційного підприємництва

Кафедра технічної кібернетики й інформаційних технологій ім. професора Р.В. Мерката

Освітньо-кваліфікаційний рівень бакалавр

Напрям підготовки -
(шифр і назва)

Спеціальність 122 «Комп'ютерні науки»
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КТКйІТ ім. проф. Р.В. Мерката

Павло НОСОВ

«___» _____ 20 26 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Артема ЗАПОРОЖАНА
(Ім'я, ПРІЗВИЩЕ)

1. Тема проекту (роботи) Використання PDF файлів в якості стеганографічних контейнерів

керівник проекту (роботи) Юрій ДАУС, к.геогр.н., доцент
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «18» 05 20 26 р. №124вк/дфн

2. Строк подання проекту (роботи) Червень 2026

3. Вхідні дані до проекту (роботи) PDF файли, стандарти симетричного та асиметричного шифрування, хешування, словники, метадані, генерація ключів.

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

1. Аналіз сучасног стану стеганографічних методів.
2. Розробка архітектури додатку для приховування сповіщень в PDF файлах
3. Розробка та реалізація додатку. 4. Охорона праці

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Схема структури та складу стеганосистеми, діаграма класів додатку, інтерфейс програми

6. Консультанти розділів проекту (роботи)

Розділ	Ім'я, ПРІЗВИЩЕ та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Сергій ХОТІН	11.05.26	08.06.26
Охорона праці, нормоконтроль	доцент каф. БЖДЕХ Юрій ДАУС		
	доц. каф. ТКіТ ім. проф. Р.Б. Мерзляка	15.06.2026	

7. Дата видачі завдання 27.04.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проекту (роботи)	Примітки
1	Видача завдання	27.04.2026	
2	Переддипломна практика, залік	27.04.2026-10.05.2026	
3	Коригування завдання за результатами практики	17.05.2026	
4	Проміжний звіт на кафедри оцінка готовності	08.06.2026	
5	Попередній захист на кафедрі	15.06.2026	
6	Рецензування	17.06.2026	
7	Захист на засіданні ЕК	23.06.2026	

Здобувач вищої освіти Запорожан Артем ЗАПОРОЖАН
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник проекту (роботи) Юрій ДАУС
(підпис) (Ім'я, ПРІЗВИЩЕ)

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут інформаційних технологій
та інноваційного підприємництва
(повна назва інституту)
Кафедра технічної кібернетики й інформаційних технологій
ім. професора Р.В. Меркта
(повна назва кафедри)

Пояснювальна записка

до кваліфікаційної роботи
бакалавр
(освітньо-кваліфікаційний рівень)
на тему
Використання PDF файлів в якості стеганографічних
контейнерів

Виконав:
здобувач вищої освіти 4 курсу, групи 1
Спеціальність:
122 «Комп'ютерні науки»
(шифр і назва спеціальності)
Артем ЗАПОРОЖАН
(підпис) (Ім'я, ПРІЗВИЩЕ)
Керівник к.геогр.н., доцент
(вчене звання, посада)
Юрій ДАУС
(підпис) (Ім'я, ПРІЗВИЩЕ)

Одеса – 2026

АНОТАЦІЯ

У роботі досліджено доцільність використання PDF-документів як носіїв прихованої інформації в стеганографічних системах. В умовах стрімкого розвитку інформаційних технологій стеганографія посідає важливе місце серед засобів забезпечення конфіденційності та захисту даних. Традиційно для приховування інформації переважно застосовуються графічні файли, які характеризуються значною інформаційною ємністю та простотою реалізації відповідних методів. Водночас широке використання зображень як стеганографічних контейнерів робить їх об’єктом підвищеної уваги з боку засобів аналізу та виявлення прихованих повідомлень.

Провести експериментальне дослідження ефективності розроблених методів та оцінити їх стійкість до виявлення й руйнування прихованих даних.

Виконати оцінку практичної придатності запропонованих рішень для використання в реальних умовах експлуатації.

У цьому контексті використання PDF-файлів як альтернативного середовища для вбудовування прихованих даних є перспективним напрямом досліджень. Такий підхід дозволяє зменшити ймовірність виявлення факту застосування стеганографічних методів, а також розширює спектр доступних інструментів для організації прихованого обміну інформацією.

Розроблення програмного комплексу, що реалізує механізми стеганографічного захисту на основі PDF-документів, забезпечує можливість інтеграції різноманітних алгоритмів приховування та криптографічного захисту даних. Це сприяє підвищенню гнучкості та ефективності застосування таких систем у сучасному інформаційному середовищ.

Під час створення програмного забезпечення використовувалися сучасні принципи об’єктно-орієнтованого програмування, а також алгоритми симетричного й асиметричного шифрування. Результатом проведеної роботи став функціональний програмний продукт, який може бути застосований для

забезпечення безпечного обміну інформацією у військових структурах, корпоративному секторі та під час приватних комунікацій.

ABSTRACT

The paper investigates the feasibility of using PDF documents as carriers of hidden information in steganographic systems. In the context of the rapid development of information technologies, steganography occupies an important place among the means of ensuring confidentiality and data protection. Traditionally, graphic files are mainly used for hiding information, which are characterized by significant information capacity and ease of implementation of the corresponding methods. At the same time, the widespread use of images as steganographic containers makes them the object of increased attention from the analysis and detection of hidden messages.

In this context, the use of PDF files as an alternative environment for embedding hidden data is a promising area of research. This approach allows you to reduce the likelihood of detecting the use of steganographic methods, and also expands the range of available tools for organizing hidden information exchange.

The development of a software package that implements steganographic protection mechanisms based on PDF documents provides the ability to integrate various algorithms for hiding and cryptographic data protection. This contributes to increasing the flexibility and efficiency of the use of such systems in the modern information environment.

When creating the software, modern principles of object-oriented programming, as well as symmetric and asymmetric encryption algorithms, were used. The result of the work was a functional software product that can be used to ensure secure information exchange in military structures, the corporate sector and during private communications.

ЗМІСТ

ВСТУП

У сучасному інформаційному просторі особливого значення набувають технології забезпечення конфіденційності даних, серед яких важливе місце займає стеганографія. На відміну від криптографічних методів захисту, що спрямовані на приховування змісту повідомлення, стеганографія орієнтована на маскування самого факту існування та передачі інформації [1].

Актуальність теми дослідження. В умовах стрімкого розвитку інформаційних технологій та глобального поширення цифрових комунікацій особливого значення набувають питання захисту інформації. Сучасні засоби інформаційної безпеки спрямовані не лише на забезпечення конфіденційності даних, але й на приховування самого факту їх існування та передачі. Саме тому важливе місце серед технологій захисту інформації займає стеганографія — наука і практика прихованого розміщення повідомлень у різноманітних інформаційних об'єктах.

Найбільшого поширення набули графічні формати, оскільки вони мають значну надлишковість даних та дозволяють непомітно змінювати окремі елементи зображення. Водночас широке використання таких контейнерів призвело до активного розвитку методів стеганоаналізу, що значно підвищило ймовірність виявлення прихованої інформації.

У зв'язку з цим актуальним напрямком розвитку сучасної стеганографії є пошук нових типів контейнерів, які поєднували б значну ємність, стійкість до руйнування прихованих даних та низьку ймовірність виявлення. Одним із перспективних форматів є PDF (Portable Document Format), який широко застосовується для зберігання та передавання електронних документів у державних установах, підприємствах, освітніх закладах та приватному листуванні.

Метою роботи є дослідження можливостей використання PDF-файлів як стеганографічних контейнерів, аналіз існуючих методів приховування інформації у структурі PDF-документів, розроблення та реалізація програмного засобу для

вбудовування, шифрування, вилучення та перевірки цілісності прихованих повідомлень із використанням особливостей внутрішньої структури PDF-файлів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Провести аналіз сучасного стану розвитку стеганографії та визначити основні напрями її застосування у сфері інформаційної безпеки.
- Дослідити принципи побудови сучасних стеганографічних систем та класифікацію стеганографічних контейнерів.
- Проаналізувати існуючі методи приховування інформації в графічних, аудіо-, відеофайлах та інших цифрових об'єктах.
- Виконати дослідження внутрішньої структури PDF-файлів та визначити елементи, придатні для прихованого розміщення інформації.
- Провести порівняльний аналіз методів вбудовування повідомлень у PDF-документи, включаючи використання коментарів, метаданих, потоків даних, службових об'єктів та прихованих структур документа.
- Розробити алгоритми приховування та вилучення інформації з PDF-файлів.
- Реалізувати механізми криптографічного захисту прихованих повідомлень із застосуванням сучасних алгоритмів шифрування.
- Розробити програмний комплекс для роботи зі стеганографічними PDF-контейнерами.

Об'єктом дослідження є процеси прихованого зберігання та передавання інформації в цифрових документах із використанням методів комп'ютерної стеганографії.

Предметом дослідження є методи, алгоритми та програмні засоби приховування, шифрування, вилучення та захисту інформації в PDF-файлах як стеганографічних контейнерах.

Для досягнення поставленої мети в роботі використовуються такі методи дослідження:

- методи системного аналізу та узагальнення наукових джерел для дослідження сучасного стану розвитку стеганографії та інформаційної безпеки;
- методи структурного аналізу цифрових документів для вивчення внутрішньої організації PDF-файлів;
- методи комп'ютерної стеганографії для розроблення алгоритмів приховування інформації;
- криптографічні методи симетричного та асиметричного шифрування для забезпечення конфіденційності прихованих повідомлень;
- методи математичного моделювання для оцінювання ефективності запропонованих алгоритмів;
- методи об'єктно-орієнтованого проєктування та програмування під час створення програмного комплексу;
- методи експериментальних досліджень для перевірки працездатності та оцінювання характеристик розробленої системи;
- методи порівняльного аналізу для визначення переваг та недоліків різних способів вбудовування інформації в PDF-документи.

Основою стеганографічних технологій є вбудовування прихованих повідомлень у різноманітні інформаційні об’єкти таким чином, щоб їх наявність залишалася непомітною для стороннього спостерігача. Об’єкти, які використовуються для розміщення прихованих даних, прийнято називати стеганографічними контейнерами. Значний розвиток цього напряму відбувся з поширенням цифрових технологій та появою електронних документів, графічних зображень, аудіо- та відеофайлів.

Найбільш поширеними контейнерами для приховування інформації традиційно є цифрові зображення. Це пояснюється особливостями людського зорового сприйняття, яке не здатне фіксувати незначні зміни окремих колірних компонентів. Завдяки цьому широкого застосування набув метод модифікації

молодших бітів кольорових каналів, що дозволяє вбудовувати приховані повідомлення без помітного погіршення якості зображення.

Разом із тим перспективним напрямом розвитку стеганографічних систем є використання PDF-документів як контейнерів для прихованих даних. Згідно з результатами досліджень, наведених у роботі [2], за потенціалом застосування у стеганографії PDF-файли поступаються лише мережевому трафіку. Це обумовлено їх значною інформаційною ємністю, структурованою організацією внутрішнього вмісту, високою стійкістю до змін під час збереження та редагування, а також широким поширенням у сучасних інформаційних системах. Важливим завданням є дослідження та аналіз існуючих способів приховування інформації в таких контейнерах, а також розроблення нових підходів до вбудовування даних. Проведення подібних досліджень сприяє підвищенню ефективності механізмів захисту інформації, удосконаленню методів прихованого обміну даними та створенню засобів виявлення можливих стеганографічних загроз. У зв’язку з цим вивчення можливостей використання PDF-файлів як стеганографічних контейнерів є актуальним науково-практичним завданням, вирішення якого сприятиме подальшому розвитку технологій захисту конфіденційної інформації.

1 ВИДИ СТЕГАНОГРАФІЧНИХ КОНТЕЙНЕРІВ

1.1 Становлення та розвиток стеганографії

Стеганографія є одним із напрямів захисту інформації, метою якого є приховування факту існування повідомлення. Термін походить від грецьких слів «στεγανός» (прихований) та «γράφω» (пишу) і означає сукупність методів та засобів прихованої передачі або зберігання даних [1]. На відміну від криптографії, яка забезпечує захист змісту повідомлення шляхом його шифрування, стеганографія спрямована на те, щоб зробити сам процес обміну інформацією непомітним для сторонніх осіб.

Принцип функціонування стеганографічних систем полягає у вбудовуванні секретних даних у різноманітні носії інформації таким чином, щоб приховане повідомлення не викликало підозри щодо своєї наявності. У сучасних умовах як середовище для розміщення прихованої інформації можуть використовуватися практично будь-які цифрові об’єкти, зокрема текстові документи, графічні файли, аудіозаписи, відеоматеріали та мережевий трафік [3].

Сфера застосування стеганографії охоплює широкий спектр напрямів діяльності, серед яких інформаційна безпека, захист конфіденційних даних, забезпечення секретності комунікацій, цифрова криміналістика, а також розроблення методів виявлення та протидії стеганографічним загрозам [4]. Поєднання стеганографічних і криптографічних механізмів дозволяє створювати багаторівневі системи захисту інформації, у яких одночасно забезпечується як приховування змісту повідомлення, так і маскування самого факту його передавання. Такий підхід є особливо актуальним в умовах підвищених вимог до безпеки інформаційного обміну.

Історія розвитку стеганографії налічує понад дві тисячі років. Одні з перших описів методів прихованого передавання інформації містяться у праці Геродота «Історія», датованій V століттям до нашої ери. У ній наведено приклади використання прихованих повідомлень шляхом нанесення тексту на поголену

голову раба з подальшим відправленням його після відростання волосся, а також записування інформації на дерев’яну основу воскових табличок із наступним нанесенням поверх неї звичайного тексту.

Подальший розвиток методів приховування інформації пов’язаний із використанням так званих симпатичних чорнил. Такі речовини дозволяли створювати невидимі записи, які ставали доступними для читання лише після впливу певних фізичних або хімічних факторів, зокрема нагрівання, освітлення чи оброблення спеціальними реагентами. Відомим прикладом є використання органічних рідин, зокрема молока, сліди якого проявлялися під дією високої температури.

Окремий напрям становили повідомлення з обмеженим терміном існування, що створювалися за допомогою спеціальних барвників, здатних поступово руйнуватися під впливом навколишнього середовища. Проте розвиток сучасних методів криміналістичного аналізу значно підвищив можливості виявлення та відновлення подібної інформації.

У різні історичні періоди застосовувалися також інші способи приховування відомостей, серед яких запис повідомлень усередині харчових продуктів, використання спеціального розташування гральних карт, геометричні методи розміщення ключових слів у тексті, застосування трафаретів, проколів на предметах, умовних символів та вузликового письма. Незважаючи на відмінності в реалізації, усі зазначені підходи мали спільну мету — забезпечення прихованого передавання інформації без привернення уваги сторонніх осіб.

На сучасному етапі розвитку інформаційних технологій традиційні фізичні носії практично повністю поступилися місцем цифровим контейнерам. Використання електронних документів, мультимедійних файлів і мережевих ресурсів значно розширило можливості застосування стеганографічних методів, забезпечивши високу місткість контейнерів, швидкість оброблення даних та можливість інтеграції з сучасними засобами криптографічного захисту інформації.

1.2 Різновидності стеганографічних контейнерів

Стеганографічна система являє собою комплекс взаємопов’язаних елементів і механізмів, призначених для прихованого передавання та зберігання інформації. До складу такої системи входять контейнер, секретне повідомлення, стеганографічний ключ, заповнений контейнер, а також алгоритми вбудовування та вилучення даних. Як контейнер можуть використовуватися різноманітні цифрові об’єкти, зокрема текстові документи, графічні зображення, аудіофайли та відеоматеріали. У ролі прихованого повідомлення може виступати текстова, графічна або інша цифрова інформація. Стеганографічний ключ являє собою набір параметрів, що визначає спосіб розміщення прихованих даних у контейнері та порядок їх подальшого відновлення [8].

Структура стеганосистеми може включати в себе різноманітні компоненти і процеси необхідні для вбудовування, шифрування, вилучення та розшифровування сповіщень. На рис. 1.1 представлена типова схема стеганосистеми.

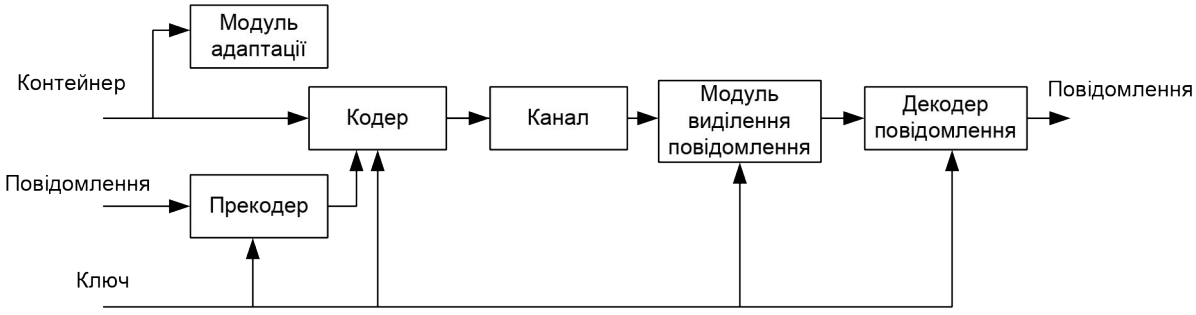

Рисунок 1.1 – Типова структура та склад стегосистеми

Архітектура стеганографічної системи може відрізнятися залежно від обраних методів приховування інформації, проте її основною функцією залишається забезпечення надійного вбудовування, зберігання та вилучення прихованих повідомлень. Стеганосистема може бути реалізована як програмними, так і апаратними засобами або їх комбінацією. Одним із ключових завдань такої системи є адаптація алгоритмів приховування до особливостей середовища, у

якому розміщуються секретні дані, із забезпеченням можливості їх подальшого коректного відновлення [9, 10].

Типова стеганографічна система включає декілька функціональних модулів:

– контейнер, який використовується як носій прихованої інформації;

– прекодер, що виконує попередню підготовку повідомлення та його перетворення до форми, найбільш придатної для вбудовування;

– модуль адаптації, призначений для аналізу характеристик контейнера та вибору оптимального способу розміщення інформації;

– модуль вилучення даних, який забезпечує отримання прихованого повідомлення зі стеганоконтейнера;

– декодер, що здійснює зворотне перетворення вилучених даних та відновлює початковий вигляд повідомлення.

Сучасні стеганографічні технології використовують широкий спектр цифрових носіїв інформації. Вибір контейнера визначається його структурними особливостями, місткістю та стійкістю до зовнішніх впливів [11].

До найбільш поширених типів стеганографічних контейнерів належать:

1. Різноманітні файли зображень в які може бути вписана прихована інформація. Це один із самих поширених видів стеганографічних контейнерів на сьогодні. Приховане сповіщення може бути вбудовано в пікселі зображення змінюючи значення самих пікселів, або в метадані цього файлу. Часто використовують зображення представлені файлами JPG, JPEG, PNG, BMP.

2. Відео файли з розширенням AVI, MP4, MKV і т.д. Тут можна змінювати кольори зображення, або вписувати в аудіо-доріжку.

3. Аудіо файли з розширенням MP3,WAV, FLAC та інші. Тут для приховування сповіщення вносять незначні зміни в аудіосигнал або доповнюють сигнал використанням високочастотних компонентів, які не сприймається людським вухом.

4. Текстові файли. Стеганографія може використовувати текстові файли, вставляючи секретну інформацію в текстовий контент, інтервали між словами або різні атрибути тексту, такі як розмір шрифту, колір тексту тощо.

5. Мережевий трафік (Cover Network Traffic): секретні дані можуть бути вбудовані в мережевий трафік, використовуючи певні властивості пакетів.

6. Інші формати: Стеганографія може застосовуватися до різних інших форматів даних, включаючи PDF-файли, ZIP-архіви, електронні документи, а також в інших креативних способах, в залежності від конкретного завдання та потреби в приховуванні інформації.

Після вибору контейнера необхідно визначити спосіб вбудовування секретного повідомлення. Реалізація цього процесу безпосередньо залежить від типу носія інформації та його технічних характеристик. Для виконання операцій приховування можуть використовуватися як програмні, так і апаратні засоби. Апаратна реалізація частіше застосовується під час роботи з мережевим трафіком та спеціалізованими системами передавання даних, тоді як для більшості цифрових контейнерів використовуються програмні модулі.

Важливим елементом будь-якої стеганосистеми є ключ вбудовування, який визначає правила розміщення прихованих даних у контейнері та механізм їх подальшого вилучення. Збереження конфіденційності цього ключа є необхідною умовою забезпечення безпеки всієї системи [12].

Методи вбудовування інформації, що використовуються у стеганографії, повинні забезпечувати непомітність внесених змін, достатню місткість контейнера, стійкість до випадкових або навмисних модифікацій даних, а також можливість коректного відновлення прихованого повідомлення після його вилучення.

У таблиці 1 наведено порівняльну характеристику основних методів приховування інформації залежно від типу стеганографічного контейнера. Аналіз наведених даних свідчить про те, що ефективність застосування певного способу вбудовування безпосередньо залежить від структури та особливостей конкретного формату даних. Деякі методи можуть використовуватися лише для окремих типів

контейнерів, тоді як інші є універсальними та придатними для широкого спектра цифрових об’єктів.

Таблиця 1.1 - Методи вбудовування сповіщень в контейнери

Контейнер Тип вбудовування	Зображення	Відео	Аудіо	PDF файли	Текстові файли	Архівні файли
Заміна молодших бітів LSB	+	+	−	+	+	−
Заміна частоти відтворення	−	+	+	−	−	−
Вписування в службові зони	−	+	+	+	+	+
Додавання після кінця файлу	+	+	+	+	+	+
Вписування інформації в метадані	+	+	+	+	+	−

Після отримання стеганографічного контейнера одержувач повинен виконати процедуру вилучення та відновлення прихованого повідомлення. Цей процес складається з кількох послідовних етапів:

1. Визначення способу вбудовування інформації. На початковому етапі необхідно встановити метод, за допомогою якого було реалізовано приховування даних. Залежно від типу контейнера це може бути аналіз модифікованих молодших бітів у графічних файлах, дослідження частотних характеристик аудіосигналу або перевірка службових структур документа.

2. Вилучення прихованих даних. Після ідентифікації способу вбудовування здійснюється безпосереднє отримання прихованої інформації зі стеганографічного контейнера. Якість виконання цього етапу визначає повноту та достовірність відновленого повідомлення.В) Перетворення отриманих даних до вигляду що дає

можливість прочитати цю інформацію. Часто дані перед поміщенням в контейнер шифрують і тому потрібно дешифрування та декодування.

3. Декодування та дешифрування повідомлення. Вилучені дані зазвичай перебувають у вигляді, непридатному для безпосереднього сприйняття користувачем. Тому виконується процедура декодування, а в разі використання криптографічного захисту — додаткове дешифрування інформації з подальшим відновленням її первинного представлення.

4. Контроль цілісності інформації. Для підтвердження коректності отриманих даних проводиться перевірка їх цілісності. Така процедура дозволяє виявити можливі пошкодження, втрату фрагментів повідомлення або несанкціоновані зміни. Одним із поширених способів контролю є використання хеш-функцій, значення яких може бути попередньо вбудоване до прихованого повідомлення.

5. Видалення залишкових слідів прихованої інформації. За наявності відповідної технічної можливості доцільним є очищення контейнера від вбудованих даних після завершення обробки повідомлення. Це дозволяє зменшити ризик проведення подальшого стеганоаналізу у випадку компрометації контейнера. Однак для окремих методів приховування, зокрема тих, що базуються на модифікації молодших бітів цифрових зображень, повне відновлення початкового стану контейнера може бути складним або неможливим.

Таким чином, ефективність функціонування стеганографічної системи визначається не лише надійністю алгоритму вбудовування інформації, а й коректністю виконання процедур вилучення, відновлення та перевірки достовірності прихованого повідомлення.

2 ОСНОВНІ ТЕХНОЛОГІЇ ВИКОРИСТАННЯ СТЕГАНОГРАФІЧНИХ КОНТЕЙНЕРІВ

2.1 Використання відео та аудіо файлів в якості стеганографічних контейнерів.

Одним із найскладніших середовищ для реалізації стеганографічних методів є відеофайли. Це обумовлено складною структурою відеоконтенту, який складається з великої кількості послідовних кадрів, звукових доріжок та службових даних. Така багатокомпонентна організація значно ускладнює процес приховування інформації та забезпечення її стійкості до оброблення і перекодування.

У сучасних стеганографічних системах приховані дані зазвичай вбудовуються або в відеоряд, або в аудіосупровід. Комбіноване використання обох складових одного відеофайлу як єдиного контейнера наразі застосовується досить рідко через складність реалізації та підвищені вимоги до синхронізації процесів приховування і вилучення інформації [14].

Найбільш поширеними методами приховування даних у відеоконтенті є:

– модифікація коефіцієнтів дискретного косинусного перетворення;

– вбудовування інформації на рівні бітових площин;

– використання енергетичної різниці між спектральними коефіцієнтами.

Метод модифікації коефіцієнтів дискретного косинусного перетворення ґрунтується на внесенні змін до параметрів, що використовуються під час стиснення відеоданих. Його перевагою є відносно висока стійкість до виявлення, проте одним із суттєвих недоліків виступає обмежена місткість контейнера. Через необхідність збереження візуальної якості відео для приховування інформації може бути використана лише незначна частина доступних коефіцієнтів, що зазвичай не перевищує 10–12 % від їх загальної кількості.

Метод вбудовування на рівні бітових площин характеризується високою швидкістю реалізації та значною пропускною здатністю. Його суть полягає у зміні

окремих бітів цифрового представлення відеоданих. Водночас така інформація є недостатньо захищеною від втрати, оскільки повторне кодування або оброблення відеофайлу може призвести до часткового чи повного знищення прихованого повідомлення. Крім того, надмірне використання цього методу здатне погіршувати якість відеозображення, що підвищує ймовірність виявлення факту приховування даних [15, 16].

Метод, заснований на використанні енергетичних відмінностей між спектральними коефіцієнтами, дозволяє забезпечити більш високу стійкість прихованої інформації до окремих видів оброблення. Проте його застосування супроводжується зменшенням ефективної пропускної здатності відеопотоку та ускладненням процесу вбудовування повідомлень.

Паралельно з розвитком стеганографічних методів для відеоконтенту активно вдосконалювалися технології приховування інформації в аудіофайлах. Особливістю звукового середовища є обмежені можливості людського слухового апарату щодо сприйняття частотного спектра. Людина здатна сприймати лише певний діапазон звукових частот, тоді як сигнали за його межами практично не реєструються слухом. Саме ця особливість широко використовується при розробленні аудіостеганографічних алгоритмів.

Одним із найбільш поширених методів приховування інформації в аудіофайлах є технологія модифікації молодших значущих бітів (Least Significant Bit, LSB). Принцип роботи методу полягає у внесенні змін до одного або кількох молодших бітів цифрового представлення аудіосигналу. Оскільки такі зміни практично не впливають на загальні характеристики звуку, вони залишаються непомітними для слухового сприйняття людини.

Ефективність методу пояснюється тим, що зміна останнього біта двійкового числа призводить лише до незначної зміни його числового значення. Завдяки цьому в аудіосигнал можна вбудовувати приховані повідомлення без помітного погіршення якості відтворення. Для наочності на рис. 2.1 наведено приклад двох чисел, які відрізняються лише значенням молодшого біта.

Рисунок 2.1 – Змінення молодшого біта

Якщо змінити лише молодший біт двійкового представлення значення аудіосигналу, різниця між початковим та модифікованим значенням буде мінімальною. Наприклад, значення 170 після зміни останнього біта перетворюється на 171. Така незначна модифікація практично не впливає на сприйняття звуку людиною, оскільки зміни знаходяться нижче порога слухової чутливості. Саме тому метод LSB вважається одним із найефективніших способів приховування інформації в аудіофайлах. Його перевагами є висока місткість контейнера, простота реалізації та відсутність помітного впливу на якість відтворення.

Для збільшення обсягу прихованих даних інколи модифікують не один, а два молодших біти цифрового представлення сигналу. У такому випадку значення 170 може змінитися до 172. Навіть така зміна залишається практично непомітною для більшості аудіоматеріалів. Водночас ступінь помітності спотворень значною мірою залежить від характеру звукового контенту. У записах із високим рівнем фонового шуму, наприклад шумом транспорту або людського натовпу, подібні зміни майже неможливо виявити. Натомість у високоякісних музичних композиціях або сольному виконанні академічних творів навіть незначні спотворення можуть бути помітними під час детального аналізу.

Для коректного вилучення прихованого повідомлення необхідно знати розташування вибірок, у які були внесені зміни. Як правило, обсяг секретної інформації повинен бути меншим за місткість контейнера, що забезпечує рівномірний розподіл прихованих даних по всьому аудіофайлу. Такий підхід дозволяє уникнути статистичних аномалій, які можуть бути виявлені засобами стеганоаналізу.

Одним зі способів підвищення непомітності є випадковий розподіл модифікованих бітів по всій структурі аудіофайлу. Однак використання повністю

випадкового розташування створює складнощі під час відновлення прихованої інформації, оскільки отримувач повинен знати точне місце розташування всіх змінених елементів.

Більш практичним рішенням є застосування псевдовипадкової послідовності. У цьому випадку генератор випадкових чисел ініціалізується спеціальним параметром — секретним ключем. Використання однакового початкового значення дозволяє як відправнику, так і одержувачу сформувати ідентичну послідовність координат для запису та вилучення прихованих даних. Після отримання контейнера користувач відтворює ту саму послідовність і виконує процедуру відновлення повідомлення.

Разом із тим застосування одного й того самого ключа для великої кількості повідомлень негативно впливає на рівень захищеності системи. У разі його компрометації зловмисник отримує можливість аналізувати всі передані повідомлення. Тому для підвищення стійкості до криптоаналітичних атак доцільно генерувати окремий ключ для кожного сеансу передавання інформації. Такий підхід вимагає реалізації додаткових механізмів безпечного розповсюдження ключової інформації між учасниками обміну даними [21].

2.2 Застосування графічних файлів в стеганографічних контейнерах

Цифрові зображення належать до найбільш поширених типів стеганографічних контейнерів завдяки значній місткості та високій стійкості до візуального виявлення змін. Методи приховування інформації в графічних файлах умовно поділяються на дві основні групи: форматні та неформатні.

Форматні методи базуються на використанні особливостей структури файлів і передбачають запис інформації до службових полів, коментарів або метаданих. Неформатні методи працюють безпосередньо з графічними даними, змінюючи окремі біти пікселів зображення.

Найбільш поширеним неформатним підходом є метод модифікації молодших значущих бітів (Least Significant Bit, LSB). Його принцип полягає у заміні одного або кількох молодших бітів кольорових компонентів пікселів на біти прихованого повідомлення. Оскільки така модифікація викликає лише незначні зміни кольору, вона практично не сприймається людським зором [23].

Для реалізації цього методу використовуються молодші біти каналів RGB та, за наявності, альфа-каналу. Кількість бітів, що підлягають заміні, залежить від необхідного обсягу прихованої інформації та допустимого рівня спотворення зображення. У більшості випадків модифікація одного або двох молодших бітів не призводить до помітного погіршення якості контейнера.

Під час приховування текстових даних кожен символ кодується певною кількістю бітів. Відповідно для розміщення одного символу використовується декілька пікселів зображення. Під час декодування виконується зчитування змінених бітів та їх об’єднання у вихідну послідовність даних.

Важливим аспектом є вибір схеми розташування модифікованих пікселів. Найпростішим варіантом є послідовний запис інформації в усі пікселі зображення. Проте такий підхід знижує рівень прихованості та полегшує проведення статистичного аналізу контейнера. Для підвищення захищеності застосовують псевдовипадкові або спектральні схеми розподілу даних [24]. У цьому випадку для вилучення повідомлення необхідно відтворити карту розташування змінених пікселів, використовуючи відповідний ключ або алгоритм генерації координат.

Якщо повідомлення займає лише частину контейнера, додатково необхідно зберігати інформацію про його довжину. Для підвищення рівня захисту приховані дані зазвичай попередньо шифруються симетричним алгоритмом, а ключ дешифрування передається адресату окремим захищеним каналом зв’язку.

2.3 Використання PDF файлів як стеганографічний контейнер

Результати досліджень, наведених у роботі [2], демонструють високий потенціал PDF-документів як середовища для приховування інформації. Згідно з проведеним порівняльним аналізом різних типів контейнерів, найвищу оцінку отримав мережевий трафік, тоді як PDF- та ZIP-файли посіли друге місце за сукупністю характеристик.

Таблиця 2.1 - Оцінка якості стеганоконтейнера.

№	Назва стегано-котейнера	Рівно-мірність	Ємність контей-нера	Рідкість викорис-тання	Ємність контей-нера	Важкість руйну-вання	Загаль-ний бал
1	Малюнок	5	4	2	4	1	12
2	Відео та аудіо	4	5	3	5	2	14
3	Текст файл	1	3	4	3	1	9
4	Мережевий трафік	3	5	5	5	5	18
5	ZIP та PDF файли	4	4	4	4	4	15

Високі показники PDF-контейнерів пояснюються їх значною місткістю, стійкістю до пошкодження даних, відносно низькою поширеністю використання у стеганографічних системах та складною внутрішньою структурою. У межах даної роботи основна увага приділяється саме PDF-документам, тоді як дослідження можливостей ZIP-архівів залишаються перспективним напрямом подальших досліджень.

Структура PDF-документа дозволяє реалізовувати широкий спектр методів приховування інформації. Одним із підходів є використання модифікації молодших бітів у потоках даних документа [25]. У цьому випадку PDF розглядається як набір взаємопов’язаних об’єктів, параметри яких можуть бути змінені без помітного впливу на зовнішній вигляд документа.

Дослідники також пропонують використовувати надлишкові елементи словників об’єктів, параметри форматування тексту, незначні зміни числових

значень операторів, а також приховані символи та модифікацію міжслівних інтервалів. Такі зміни залишаються практично непомітними для користувача, але дозволяють ефективно розміщувати додаткову інформацію всередині документа.

У роботі Ndounda R. та Ekodeck [26] запропоновано підхід, заснований на використанні китайської теореми про залишки. Повідомлення розбивається на окремі фрагменти, які вбудовуються у координати розташування текстових або графічних елементів документа. Перевагою такого методу є високий рівень захисту, оскільки відновлення повідомлення неможливе без знання всіх параметрів алгоритму.

Інший підхід передбачає використання PDF як середовища розподіленого зберігання інформації [27]. У цьому випадку секретні дані розподіляються між різними сторінками та об’єктами документа шляхом зміни параметрів шрифтів, використання невиконуваного коду або прихованих службових структур. Додатково розглядається можливість маніпуляції внутрішньою структурою документа, зокрема зміни посилань між сторінками без видалення самих об’єктів.

Окремі дослідження [29] присвячені методам приховування даних у текстовому шарі документа шляхом модифікації вирівнювання тексту. Незначні зміни міжслівних інтервалів можуть використовуватися для кодування двійкової інформації. Завдяки тому, що PDF-документ зберігає точні координати кожного текстового елемента, приховані дані можуть бути відновлені з високою точністю.

У роботі [30] розглядаються питання використання PDF-документів для приховування програмного коду та інших потенційно небезпечних об’єктів. Аналізуються способи застосування закладок, анотацій, вбудованих файлів та JavaScript-коду для транспортування додаткової інформації. Результати цих досліджень є важливими не лише для розвитку стеганографії, а й для вдосконалення методів виявлення аномалій у структурі PDF-документів.

Отже, PDF-файли можуть розглядатися як універсальний стеганографічний контейнер, що підтримує використання різноманітних методів приховування інформації. До них належать модифікація службових структур, застосування алгоритмів LSB для вбудованих зображень, зміна параметрів форматування тексту,

маніпуляція посиланнями між об’єктами документа, використання прихованих елементів структури та розміщення даних після завершення основного вмісту файлу. Така різноманітність підходів забезпечує високу гнучкість та значний потенціал PDF-документів для застосування в сучасних стеганографічних системах.

3 РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Структура PDF файлу

Використання PDF-документів як середовища для прихованого зберігання інформації має низку суттєвих переваг. Насамперед до них належать:

– значна місткість контейнера, що дозволяє розміщувати повідомлення великого обсягу;

– широке поширення формату PDF у системах електронного документообігу, електронній пошті та сучасних месенджерах;

– висока стійкість документа до випадкових змін та пошкоджень під час передавання або зберігання;

– чітко визначена структура файлу, яка спрощує розроблення алгоритмів вбудовування та вилучення прихованих даних.

Типовий PDF-документ складається з декількох логічних компонентів, кожен з яких може бути використаний для реалізації стеганографічних методів. Файл починається із заголовка, який містить службову інформацію про формат документа. Перший рядок зазвичай має вигляд «%PDF-1.x», де остання частина визначає версію специфікації PDF. Символ «%» позначає початок коментаря, тому інформація до кінця рядка розглядається програмою як службове повідомлення.

Наступний рядок також починається із символу коментаря та містить спеціальні байтові послідовності, що дозволяють програмному забезпеченню ідентифікувати документ як бінарний файл.

Основну частину PDF-документа складають непрямі об’єкти (Indirect Objects). Саме вони формують структуру документа та містять його вміст. До таких об’єктів належать сторінки, шрифти, графічні елементи, потоки даних і службові структури. Кожний об’єкт починається службовим записом виду «obj» та завершується директивою «endobj».

Всередині об’єктів можуть зберігатися різні параметри документа, серед яких:

– перелік сторінок документа;

– посилання на дочірні сторінки;

– загальна кількість сторінок;

– геометричні параметри сторінок;

– вміст сторінки;

– інформація про використані шрифти та графічні ресурси.

Особливе місце у структурі PDF займають потоки даних (Streams). Вони містять безпосередньо текстову, графічну або іншу інформацію документа. Потоки можуть бути стиснені різними алгоритмами компресії, які визначаються за допомогою параметра «Filter». Додатково для кожного потоку задається його довжина, а завершення блоку позначається службовим словом «endstream».

Фундаментальним елементом архітектури PDF є словники (Dictionaries), які забезпечують логічний зв’язок між окремими компонентами документа. Якщо потоки даних містять фактичну інформацію, то словники визначають правила її інтерпретації та відображення.

Структура PDF має ієрархічну організацію, де центральним елементом виступає словник Catalog. Він містить посилання на інші структурні компоненти документа, зокрема:

– перелік сторінок документа (Pages);

– дерево закладок (Outlines);

– іменовані об’єкти та посилання (Names).

Для роботи зі стисненими потоками використовуються спеціальні параметри словників. Наприклад, ключ «Filter» визначає алгоритм декодування інформації, необхідний для коректного відображення вмісту документа.

Завдяки системі непрямих посилань PDF-документ формує деревоподібну структуру взаємопов’язаних об’єктів. Це дозволяє програмам швидко знаходити необхідні елементи без послідовного читання всього файлу. Координати об’єктів зберігаються у спеціальній таблиці перехресних посилань (xref).

Одним із важливих словників є Resources, який виконує роль каталогу ресурсів сторінки. У ньому містяться посилання на шрифти, графічні об’єкти та інші елементи, необхідні для відображення документа.

З точки зору стеганографії особливий інтерес становлять метадані документа. Для їх зберігання використовується словник Info, у якому можуть міститися відомості про автора документа, його назву, дату створення, дату модифікації та інші службові параметри. Структура словника допускає створення додаткових користувацьких полів, що робить його зручним місцем для прихованого розміщення інформації.

На рис. 3.1 зображений типовий заголовок PDF файлу. Заголовок файлу починається з символу % PDF – 1.5 що в даному випадку вказує на версію PDF файлу 1.3. Символ % на початку рядка означає що далі йде службовий коментар і все, що починається після нього є коментарем.

Наступний рядок також починається із знаку коментаря, далі йдуть коди, які сповіщають програмам, що цей файл бінарний.

Наступний фрагмент файлу – це саме тіло файлу, що містить набір не прямих об’єктів (Indirect Objects). На рис. 3.2 представлений типовий об’єкт потоку даних. Містить фактичний вміст тексту або графіки та може бути стисненим (/Filter /FlateDecode). В даному випадку тут вказується алгоритм стиснення.

```
%PDF-1.5
%
3 0 obj
<<
   /Type /Page
   /Parent 2 0 R
   /Resources <<
   /ProcSet [/PDF /Text ]
   /Font <<
   /F1 6 0 R
   /F2 9 0 R
   /F3 12 0 R
   /F4 15 0 R
   /F5 18 0 R
   /F6 21 0 R
   /F7 24 0 R
>>
>>
   /MediaBox [0 0 595 842]
   /Contents 4 0 R
>>
endobj
4 0 obj
<<
   /Filter /FlateDecode
   /Length 4201
>>
stream
x^@\M@@@@ƒ@s@q@m  H23@@ x@ @ @@@@ @@@P@@$Rv @@E12@@G@ IQ@@x@@@@@@@_@u@ @@@@ @@@@@@EM¸D@@_@ä@@@@□@_@@@@@@@
\E@`j_@7@1@@@+)@Z @ @F/`@@@! m'$@@Os@9@@@9 @@@@@@[]㎡3]ěr@@>_¸@@@l @@@ W$@G$G7Y#I @@@8 @M_s!@@@@=@@@@@@@g
V@@
@Bc@h@@ę@@@@"@ə @m @@@ /@ @@@@@@yq
]@@@@Z@9@ @@ @z2@@@>@s7@@}n @  <□@@@)@□@s~@ k kW@@@=@I@ @@@@@i@gj@@ʒQ%@@8@@@5*3 @ỹ@@@@@@' @@ @cP@3@ q@澍@L
 P]qX01@Ψ` @zY@@@t@@}Zh@.@<@Й@@"#@SY @ @n mo@lG @ @@@ @t@'@@X,@ş<U@L @@@≥@@A@ @_CZ@@@ DqY@[@QA@@[@@@@"@@@
@9}@@@@@X9@@jg @@8 @$@@@@:ʀ+JJ{ !@*@ {#Y@@ Qq@R'@@@@@F(@H@@Ai @@嬰N@`@,+@UO@@@O@>@UWEǵo@@@a@u&b@@@8 @b@
}PYн1u @@ @@+%:g@@@;K@6@@;@@N@˜□vy JB)@@P4 /0]b@i@@>@@5N @Fl@@ )F@L:@ @K@v@]/ hiR@L _I@@ʏ@@9 3@q @@@ @@d@
I@Mc@@⏢:"A4y@W@~f@h@4@L@@@@@@ V@|`@Q@@q=U H,@@4 @@@j6ₒ□tq@f @@w@Ջ;@@2@@  $@ oX@@)@ @ọ@@
```

Рисунок 3.1 – Типовий заголовок PDF файлу

Визначається також довжина цього блоку в директиві “/Length 4201 “. Кінець потоку обрамляється службовим словом “endstream”.

У структурі PDF словники (Dictionaries) — це фундаментальний тип даних, на якому тримається вся логіка файлу. Якщо об'єкти-потоки (Streams) зберігають "важку" інформацію (текст, картинки), то словники — це інтелект файлу, який каже програмі, як цю інформацію читати.

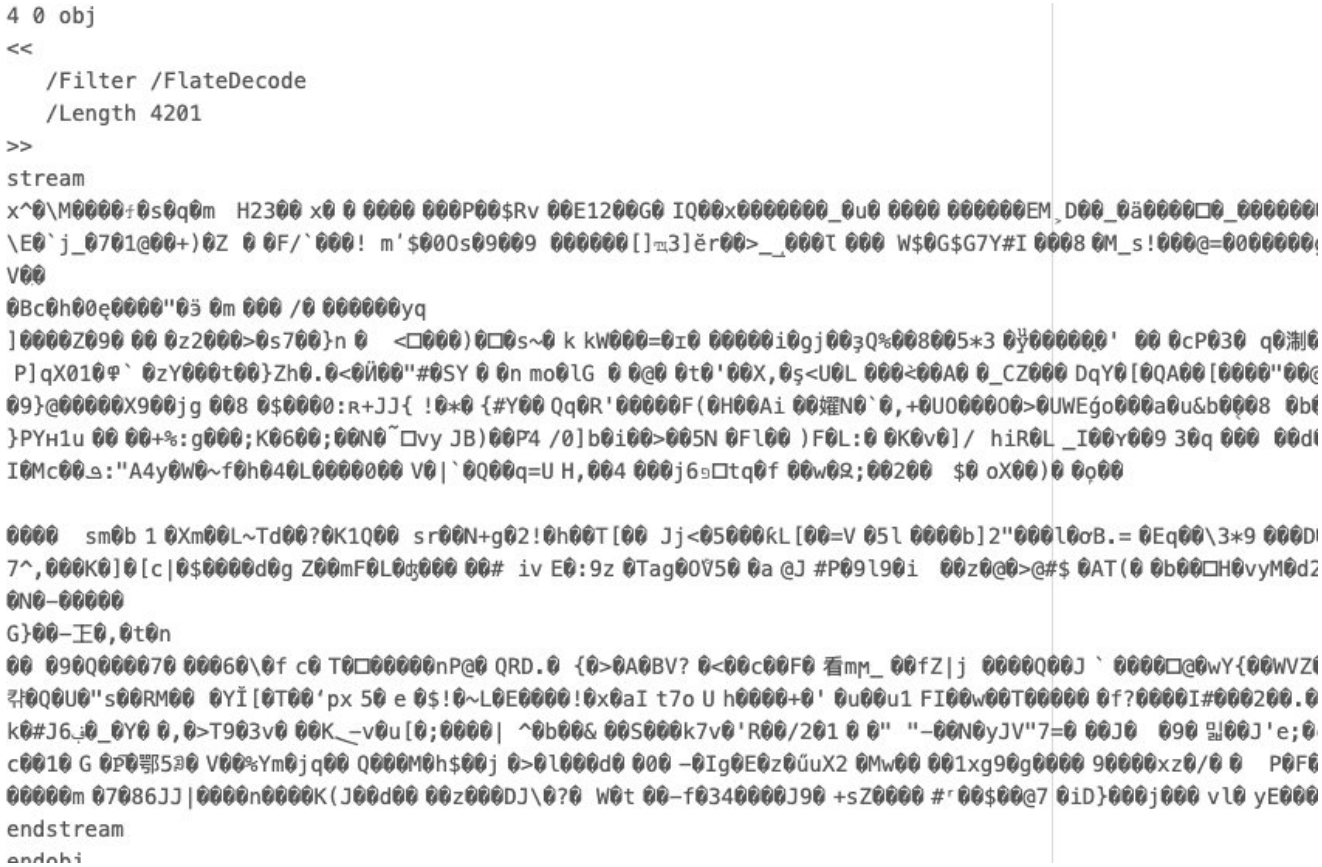

Рисунок 3.2 – Типовий об’єкт потоку даних PDF файлу

Завершальна частина PDF-документа містить таблицю перехресних посилань xref та службовий блок Trailer. У ньому зберігається інформація про розмір документа, кількість об’єктів, кореневий словник та унікальний ідентифікатор файлу. Кінець документа позначається службовою директивою «%%EOF», яка сигналізує програмі про завершення обробки файлу.

Для прикладу на рис. 3.3 приведений кінець файлу стандартного PDF файлу.

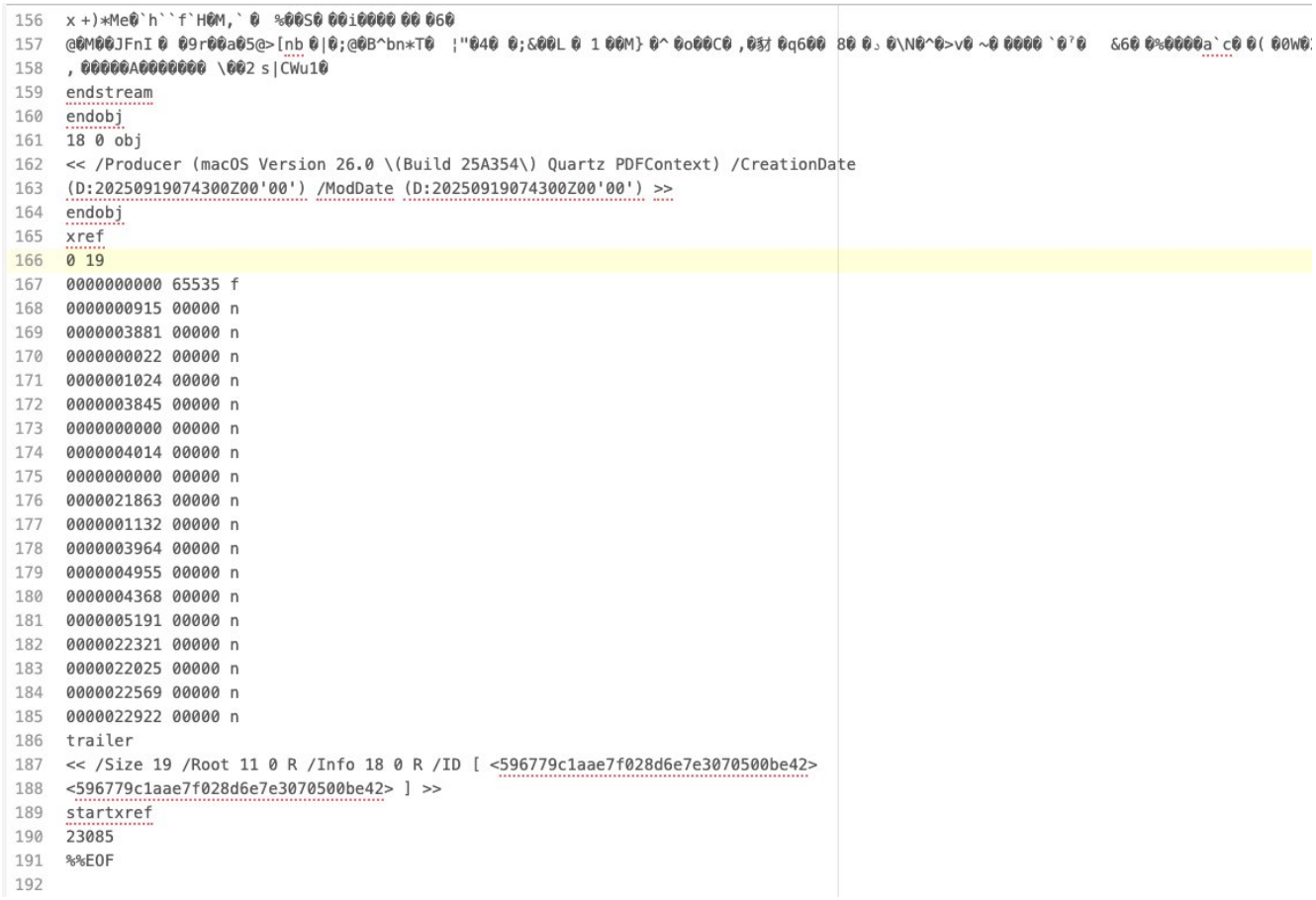

Рисунок 3.3 – Стандартний кінець PDF файлу

3.2 Використання коментарів та метаданих

Аналіз внутрішньої структури PDF-документів показує, що одним із найпростіших способів приховування інформації є використання коментарів. Будь-який коментар у PDF починається символом «%», після якого може розміщуватися довільна текстова або бінарна інформація. Деякі службові конструкції також використовують цей символ, наприклад запис «%PDF» визначає версію формату документа, а директива «%%EOF» позначає його завершення.

Стандартні засоби перегляду PDF-документів не відображають вміст коментарів і зазвичай припиняють аналіз файлу після досягнення позначки «%%EOF». Водночас спеціалізовані редактори та переглядачі текстових файлів дозволяють переглядати внутрішню структуру документа та аналізувати всі його компоненти.

Основними перевагами використання коментарів для приховування інформації є простота реалізації та можливість розміщення повідомлень значного обсягу. Недоліком такого підходу є відносна легкість виявлення прихованих даних у разі цілеспрямованого аналізу структури документа.

Для підвищення надійності приховування інформацію доцільно розміщувати між окремими об’єктами документа, наприклад після завершення одного блоку та перед початком наступного. Для ідентифікації прихованого повідомлення може використовуватися спеціальний маркер або унікальна сигнатура, яка додається після символу коментаря та дозволяє однозначно визначити початок секретних даних.

Під час вилучення інформації виконується пошук заздалегідь визначеного маркера, після чого здійснюється зчитування даних до завершення поточного рядка або до появи іншого службового роздільника. Такий механізм забезпечує просту та швидку реалізацію процедур приховування й відновлення повідомлень.

Іншим перспективним напрямом є використання метаданих документа. У сучасних PDF-файлах інформація про документ може зберігатися за допомогою двох основних механізмів: словника Info Dictionary та платформи XMP Metadata (Extensible Metadata Platform). Обидва підходи можуть функціонувати одночасно та містити дубльовану інформацію про документ.

Словник Info Dictionary містить стандартний набір атрибутів, серед яких автор документа, назва, тема, ключові слова, дата створення та дата останнього редагування. Окрім стандартних полів, структура словника дозволяє додавати додаткові користувацькі параметри, які можуть використовуватися для розміщення прихованих даних без зміни зовнішнього вигляду документа. Основні поля словника Info Dictionary наведені в таблиці 3.1.

Таблиця 3.1 - Значення ключів словника Info Dictionary

Опис	Ключ	Тип
Назва документу	/Title	string
Автор	/Author	string

Тема	/Subject	string
Ключові слова	/Keywords	string
Програма що створила вміст	/Creator	string
Програма що згенерувала PDF	/Producer	string
Дата створення	/CreationDate	date string
Дата останньої зміни	/ModDate	date string
Обробка кольорів	/Trapped	name

Особливістю словника Info Dictionary є використання спеціального формату представлення дати та часу, прийнятого у специфікації PDF. Наприклад, дата може бути записана у вигляді D:20260403153000+02'00'. Для кодування текстових значень застосовуються формати PDFDocEncoding або UTF-16BE. Хоча окремі програмні засоби допускають додавання нестандартних полів до словника, така практика не належить до офіційно рекомендованих механізмів специфікації PDF. До основних недоліків Info Dictionary належать обмежені можливості розширення, відсутність підтримки просторів імен та складних структур даних, а також спрощена організація внутрішнього представлення інформації.

Більш сучасним підходом до зберігання метаданих є використання технології XMP Metadata (Extensible Metadata Platform). Підтримка XMP була впроваджена починаючи з версії PDF 1.4 і надалі стала основним механізмом опису додаткових властивостей документа.

Технологія XMP базується на стандартах RDF/XML та являє собою окремий об'єкт типу Metadata, який вбудовується безпосередньо в структуру PDF-документа. Посилання на цей об'єкт зберігається у словнику Catalog або може бути визначене для окремих сторінок документа.

На рис 3.4 представлений типовий вигляд Metadata в форматі XML.

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xmp="http://ns.adobe.com/xap/1.0/"
        xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
      <dc:title><rdf:Alt><rdf:li xml:lang="x-default">Мій документ</rdf:li></rdf:Alt
      <dc:creator><rdf:Seq><rdf:li>Іван Петренко</rdf:li></rdf:Seq></dc:creator>
      <xmp:CreateDate>2026-04-03T15:30:00+02:00</xmp:CreateDate>
      <xmp:ModifyDate>2026-04-03T16:00:00+02:00</xmp:ModifyDate>
      <pdf:Producer>Apache PDFBox 3.0</pdf:Producer>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

Рисунок 3.4 – MetaData в форматі XML

Типова структура XMP-метаданих представлена у форматі XML. Однією з ключових переваг такого підходу є використання просторів імен, що дозволяє зберігати інформацію різних категорій без виникнення конфліктів між атрибутами. Найчастіше використовуються простори імен Dublin Core (dc:), Adobe XMP (xmp:), XMP Rights Management (xmpRights:), Photoshop Metadata (photoshop:) та IPTC Core (Iptc4xmpCore:). Крім того, користувач може створювати власні простори імен та формувати складні ієрархічні структури даних.

У специфікації PDF 2.0 словник Info Dictionary офіційно визнаний застарілим механізмом зберігання метаданих, а пріоритет віддано саме XMP Metadata. Проте для забезпечення сумісності зі старішими програмами більшість сучасних редакторів PDF продовжують одночасно формувати обидва набори метаданих. У випадку виникнення суперечностей між їх значеннями перевага надається інформації, що міститься в XMP. На рис. 3.5 представлена схема одночасного співіснування XMP Metadata і Info Dictionary.

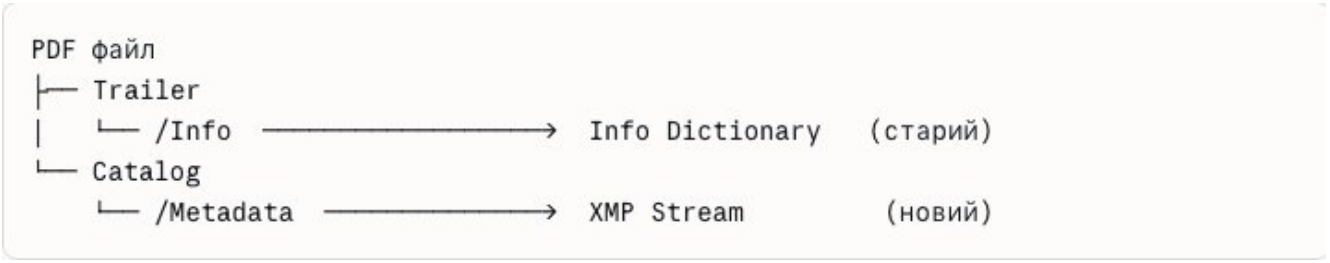

Рисунок 3.5 – Схема одночасного використання двох схем MetaData

Таким чином, структура PDF-документа надає щонайменше два зручні середовища для прихованого розміщення службової або конфіденційної інформації: словник Info Dictionary та контейнер XMP Metadata.

Додатковим способом приховування даних може бути запис інформації після службової директиви %%EOF, яка визначає логічне завершення PDF-документа. Більшість програм для перегляду PDF припиняють обробку файлу після досягнення цієї позначки та ігнорують подальший вміст. Водночас операційна система під час копіювання або збереження файлу орієнтується на його фактичний розмір, а не на внутрішні службові маркери. Завдяки цьому дані, записані після директиви %%EOF, фізично залишаються у файлі та можуть бути збережені під час його копіювання або передавання.

3.3 Захист та шифрування вбудованої інформації

Наявність прихованого повідомлення у контейнері не гарантує його захищеності від несанкціонованого доступу. Тому під час розроблення стеганографічної системи необхідно передбачати ситуацію, за якої зловмисник зможе виявити факт приховування інформації та отримати доступ до контейнера. У такому випадку додатковим рівнем захисту виступають криптографічні механізми, що унеможливлюють використання вилучених даних без знання відповідних ключів.

Одним із найбільш поширених симетричних алгоритмів шифрування на сьогодні є AES (Advanced Encryption Standard). Даний алгоритм був затверджений

Національним інститутом стандартів і технологій США (NIST) у 2001 році та залишається актуальним завдяки високій криптографічній стійкості та ефективності реалізації на сучасних обчислювальних системах.

AES належить до симетричних криптографічних алгоритмів, тобто для шифрування та дешифрування використовується один і той самий секретний ключ, який повинен бути відомий як відправнику, так і одержувачу повідомлення.

Алгоритм виконує обробку даних блоками фіксованого розміру 128 біт. Залежно від рівня захисту можуть використовуватися ключі довжиною 128, 192 або 256 біт. Кількість раундів перетворення визначається довжиною ключа та становить 10, 12 і 14 раундів відповідно. Кожен раунд включає послідовність математичних операцій підстановки, перестановки та змішування даних, що забезпечують високий рівень криптографічної стійкості алгоритму.

У реалізації AES-128 вхідний блок даних розміром 16 байтів розглядається як матриця розміром 4×4. Байти послідовно заповнюють стовпці цієї матриці, утворюючи початковий стан алгоритму. Аналогічним чином формується матриця секретного ключа, яка використовується під час генерації раундових ключів та виконання криптографічних перетворень.

У процесі шифрування формується так звана матриця стану (State Matrix), яка змінюється після виконання кожного раунду алгоритму. Після завершення останнього раунду отримана матриця перетворюється у вихідну послідовність байтів (Output Block), що являє собою зашифрований блок даних.

Схематичний взаємозв’язок між вхідним блоком даних, матрицею стану, ключовою інформацією та вихідним блоком наведено на відповідному рисунку. . Схематичне відношення між матрицями зображено на рисунку 3.6

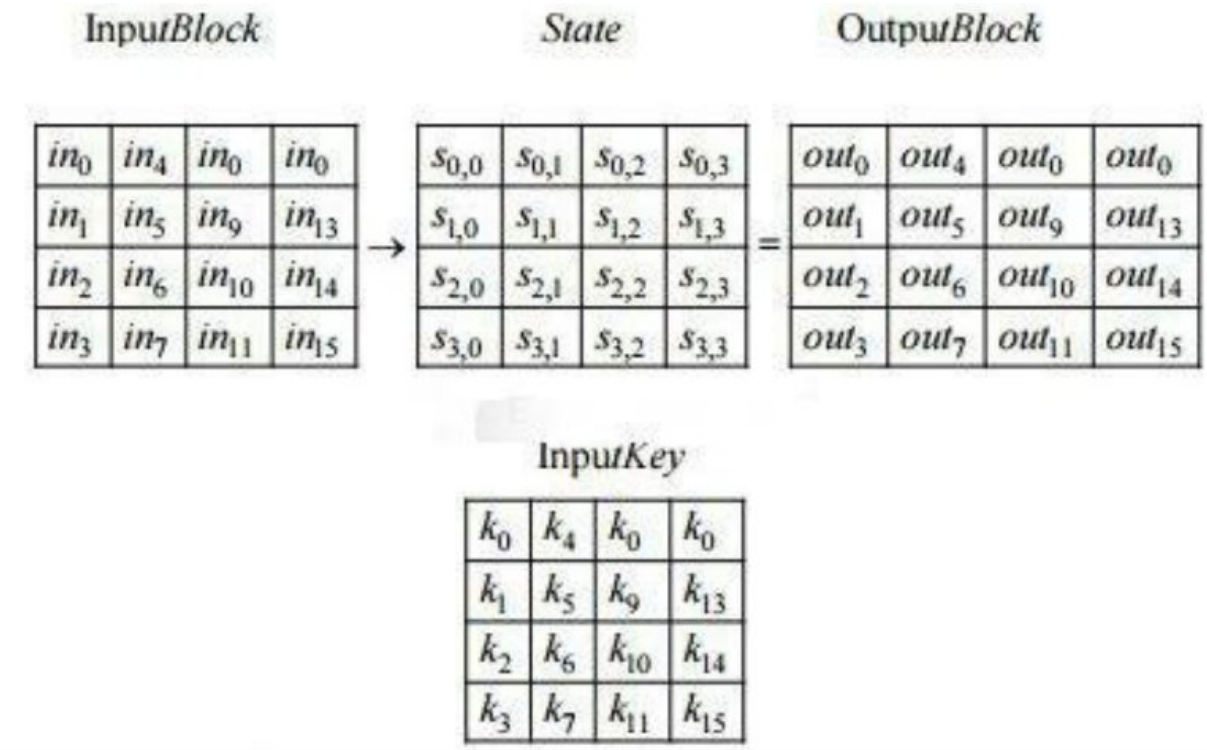

Рисунок 3.6 – Відношення між матрицями

Криптографічний ключ алгоритму симетричного шифрування AES-128 має довжину 128 бітів, які структуровані у вигляді 16 байтів (k0, k1...k15) і позиціонуються у стовпцях матриці стану InputKey. Таким чином, первинний ключ представлений чотирма 32-бітними словами w0w1w2w3, де w0= k0k1k2k3. На основі цих слів за допомогою процедури розширення ключів (Key Expansion) генерується послідовність із 44 слів (w0…w43). Для кожного раунду шифрування обчислюється і подається відповідне підмножина з чотирьох слів зазначеної послідовності.

Кожен ітераційний раунд алгоритму AES (за винятком фінального) складається з чотирьох послідовних перетворень:

SubBytes — нелінійна побайтова заміна в матриці станів із використанням фіксованої таблиці замін (S-Box). Ця операція полягає в знаходженні оберненого елемента в полі Галуа $GF(2^8)$ із наступним афінним перетворенням, що забезпечує усунення лінійних залежностей.

ShiftRows — циклічний побайтовий зсув рядків матриці стану на різні зміщення.

MixColumns — лінійне перетворення, що дифузує байти всередині стовпців матриці.

AddRoundKey — накладання раундового ключа за допомогою операції побітового додавання за модулем 2 (XOR).

Заключний раунд модифікований порівняно з попередніми та не містить етапу MixColumns.

Головним деструктивним чинником під час практичної реалізації симетричних криптосистем є необхідність безпечного розподілу та обміну ключами між відправником та отримувачем. Перехоплення ключа на етапі його транспортування нівелює рівень захисту інформації, забезпечений алгоритмом.

Асиметричні криптосистеми, зокрема алгоритм RSA, позбавлені зазначеного недоліку. Криптографічна стійкість RSA базується на обчислювальній складності задачі факторизації добутку двох великих простих чисел. Функціонування алгоритму передбачає використання ключової пари: відкритого ключа (для шифрування) та таємного ключа (для дешифрування). Дані, що пройшли процедуру зашифрування за допомогою публічного ключа, можуть бути відновлені виключно власником відповідного приватного ключа, що гарантує конфіденційність транзакцій. Крім того, архітектура асиметричного шифрування є базисом для формування електронних цифрових підписів.

Асиметричні методи шифрування, наприклад RSA, не мають таких вразливостей. RSA — це асиметричний криптографічний алгоритм, що базується на складності факторизації великих чисел, який використовує пару ключів: відкритий (для шифрування) та закритий (для дешифрування). Назва методу походить з перших літер прізвищ вчених, що розробили цей алгоритм – Rivest, Shamir та Adleman. Дані зашифровані публічним ключем може розшифрувати лише власник відповідного приватного ключа, що гарантує конфіденційність та безпеку обміну даними. Також такий підхід використовують для накладення цифрових підписів.

Математичний опис алгоритму RSA.

Процес функціонування криптосистеми RSA розподіляється на три основні етапи.

1. Генерація ключової пари:

Обираються два випадкові великі прості числа p та q (еквівалентної довжини близько 512 бітів кожне).

Обчислюється модуль системи:

- $n = pq$;
- визначається значення функції Ейлера від модуля: $\varphi(n) = (p - 1)(q - 1)$;
- обирається відкрита експонента — ціле число e, яке задовольняє умовам $1<e<\phi(n)$ що $1 < e < \varphi(n)$ та e взаємно просте з $\varphi(n)$
- за допомогою розширеного алгоритму Евкліда обчислюється таємна експонента d, яка є мультиплікативно оберненою до числа e за модулем $ed \equiv 1(\mathrm{mod}\varphi(n))$

1. Процедура зашифрування:

Для передачі повідомлення M здійснюється його попереднє кодування у ціле число m ($0 \le m < n$) за допомогою стандартизованих оборотних схем доповнення (наприклад, OAEP). Обчислення криптотексту c виконується з використанням відкритого ключа e за формулою:

$$c = m^e \ mod \ n \tag{3.1}$$

Швидкодія цієї операції для чисел великої розрядності досягається завдяки застосуванню алгоритмів зведення до степеня за модулем.Розшифрування.

2. Процедура дешифрування:

Для відновлення вихідного повідомлення m із криптотексту c обчислюється таке рівняння:

$$m = c^d \ mod \ n \tag{3.2}$$

Коректність відновлення початкових даних на основі виразів (3.1) та (3.2) доводиться в такий спосіб:

$$C^B \equiv (m^e)1^d \equiv m^{ed} \qquad (mod\ n) \tag{3.3}$$

За умови:

$$ed \equiv 1\ (mod\ \varphi(n)) \tag{3.4}$$

Враховуючи умову (3.4):

$$ed = k\varphi(n) + 1 \tag{3.5}$$

для деякого цілого k, отже:

$$m^{ed} \equiv m^{k\varphi(n)+1}\ \ (mod\ n) \tag{3.6}$$

Згідно з теоремою Ейлера існує тотожність:

$$m^{\varphi(n)} \equiv 1\ \ (mod\ n) \tag{3.7}$$

Тому:

$$m^{k\varphi(n)+1} \equiv m\ \ \ (mod\ n) \tag{3.8}$$

$$c^d \equiv m\ \ \ (mod\ \ n) \tag{3.9}$$

Попри високу надійність та відсутність етапу компрометації ключів під час їх передачі, алгоритм RSA має низку обмежень. Зокрема, до них належать низька обчислювальна швидкість порівняно з симетричними аналогами (наприклад, AES), значна надмірність довжини ключів (від 2048 до 4096 бітів) та обмеження щодо обсягу даних, які можуть бути зашифровані за один ітераційний прохід (обсяг не може перевищувати розмір модуля n). Сучасні криптографічні стандарти

вимагають використання високих показників степеня, що призводить до значних апаратних та часових затрат при роботі з великими цілими числами.

Невід'ємним аспектом захисту інформації є контроль цілісності повідомлень під час їх транзиту через незахищені мережеві вузли, де існує ризик несанкціонованої модифікації або підміни даних. Для верифікації цілісності електронних документів застосовують односторонні криптографічні хеш-функції, серед яких найбільш поширеним є алгоритм SHA-256.

До фундаментальних властивостей SHA-256 належать:

Фіксований розмір дайджесту: вихідний результат завжди має довжину 256 бітів (64 символи в шістнадцятковій системі числення).

Незворотність (односпрямованість): обчислювальна неможливість відновлення первинного тексту на основі його хеш-образу.

Стійкість до колізій: висока математична складність пошуку двох різних вхідних масивів даних, що мають ідентичні хеш-коди.

Ефект лавини (унікальність): мінімальна модифікація вхідного повідомлення (навіть на один біт) призводить до радикальної зміни результуючого хешу.

Технологічний процес обчислення дайджесту за алгоритмом SHA-256 містить такі етапи:

Попередня обробка (Padding): вхідний інформаційний масив доповнюється таким чином, щоб його кінцева довжина в бітах відповідала умові $512 \times n+448$. Процедура реалізується додаванням біта «1» з наступним заповненням нульовими бітами.

Інтеграція довжини: до кінця блоку додається 64-бітне значення, що описує вихідну довжину повідомлення до початку обробки. Це забезпечує кратність загального обсягу даних 512 бітам.

Ініціалізація векторів стану: встановлюються початкові значення 8 робочих 32-бітних змінних (a…h), які отримують із дробових частин квадратних коренів перших восьми простих чисел.

Поблокова обробка: інформаційний потік розділяється на блоки розміром 512 бітів. Кожен блок піддається 64 раундам трансформацій, що включають циклічні зсуви, бітові операції XOR та логічні функції.

Фіналізація: після обробки всієї послідовності блоків поточні значення робочих змінних конкатенуються, формуючи підсумковий 256-бітний хеш-код.

Для верифікації автентичності документа інтеграцію згенерованого хеш-рядка доцільно здійснювати шляхом його додавання після фінальної директиви %%EOF у структурі файлу. Такий підхід зумовлений тим, що операційна система сприймає ці дані як легітимну область файлової структури, тоді як спеціалізоване програмне забезпечення для візуалізації форматів (наприклад, PDF) ігнорує інформаційні блоки, розташовані після цієї директиви.

Для підвищення загального рівня стійкості інформаційної системи до компрометації рекомендується застосовувати гібридну схему захисту. У межах такого підходу процедура шифрування самого інформаційного масиву реалізується швидким симетричним алгоритмом AES, тоді як обчислений хеш-рядок підлягає зашифруванню за допомогою асиметричної криптосистеми RSA.

3.4 Структура додатку

Проектування та розробка програмного комплексу реалізована на базі об'єктно-орієнтованої мови програмування Java. Вибір даного технологічного стеку обґрунтований такими функціональними вимогами до системи:

забезпечення кросплатформеності програмного забезпечення;

наявність розвинених інтегрованих криптографічних бібліотек (Java Cryptography Architecture / Java Cryptography Extension);

високий рівень поширеності технології та підтримки спільноти;

можливість дистрибуції у вигляді автономного виконуваного модуля.

Для формування результуючого автономного модуля виконується збірка артефактів проєкту з генерацією самовиконуваного архівного пакета формату .jar. Графічне представлення структури розроблених класів, їхніх методів та

взаємозв'язків наведено на діаграмі класів (рис. 3.7).

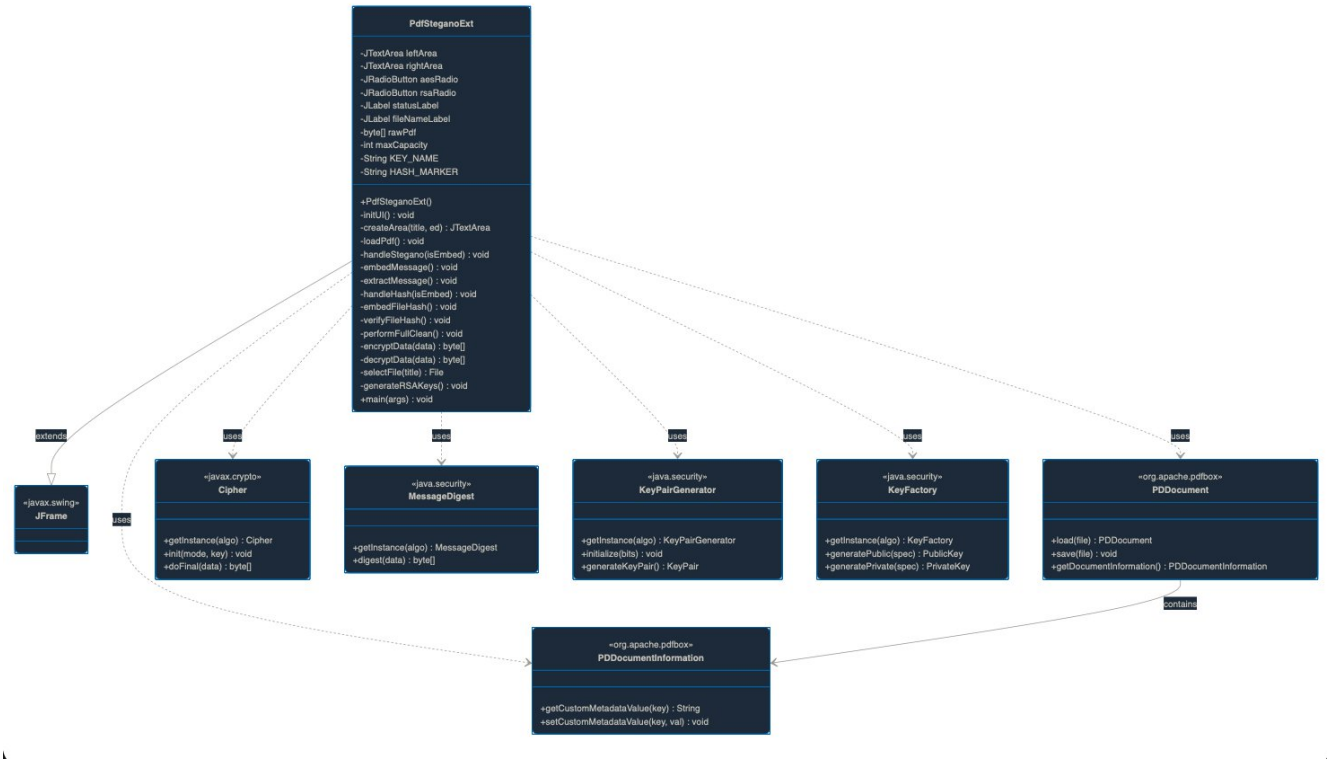

Рисунок 3.7 – діаграма класів додатку

Для проектування та реалізації графічного інтерфейсу користувача (GUI) було застосовано бібліотеку Swing. Архітектура розробленого програмного забезпечення базується на взаємодії таких основних класів:

KeyPairGenerator — забезпечує генерацію пар ключів (відкритого та таємного) для асиметричної криптосистеми й керує процедурою їх збереження у визначеній директорії файлової системи.

Cipher — призначений для криптографічного перетворення вхідних повідомлень за симетричним алгоритмом AES із довжиною ключа 128 бітів на основі парольної фрази.

EmbedMessage, ExtractMessage — реалізують алгоритми вбудовування (стеганографічного приховування) та автентичного вилучення конфіденційної інформації з контейнера відповідно.

verifyFileHash — виконує процедуру верифікації цілісності даних шляхом обчислення поточного хеш-образу та його порівняння з еталонним значенням.

PDDocument — забезпечує низькорівневу взаємодію з файлами формату PDF (завантаження об'єктної структури, запис модифікованих даних, редагування метаданих).

MessageDigest — інтегрує функції стеганографічного приховування, екстракції даних, а також повної деструкції прихованих повідомлень і супутніх хеш-рядків із метою відновлення первинного стану файлу-контейнера.

Графічну інтерпретацію головного вікна програмного комплексу наведено на рис. 3.8. Панель інструментів верхнього меню структурована за трьома функціональними секціями. Перша секція об'єднує модулі керування файлами, підсистему хешування та блоки криптографічного захисту. Зокрема, вкладка «Файл» містить чотири операційні підпункти: «Завантажити PDF», «Зберегти як (приховати сповіщення)», «Витягти прихований текст» та «Повна очистка файлу». Функціонування зазначених компонентів безпосередньо детермінується станом перемикачів, розташованих у другій та третій секціях інтерфейсу.

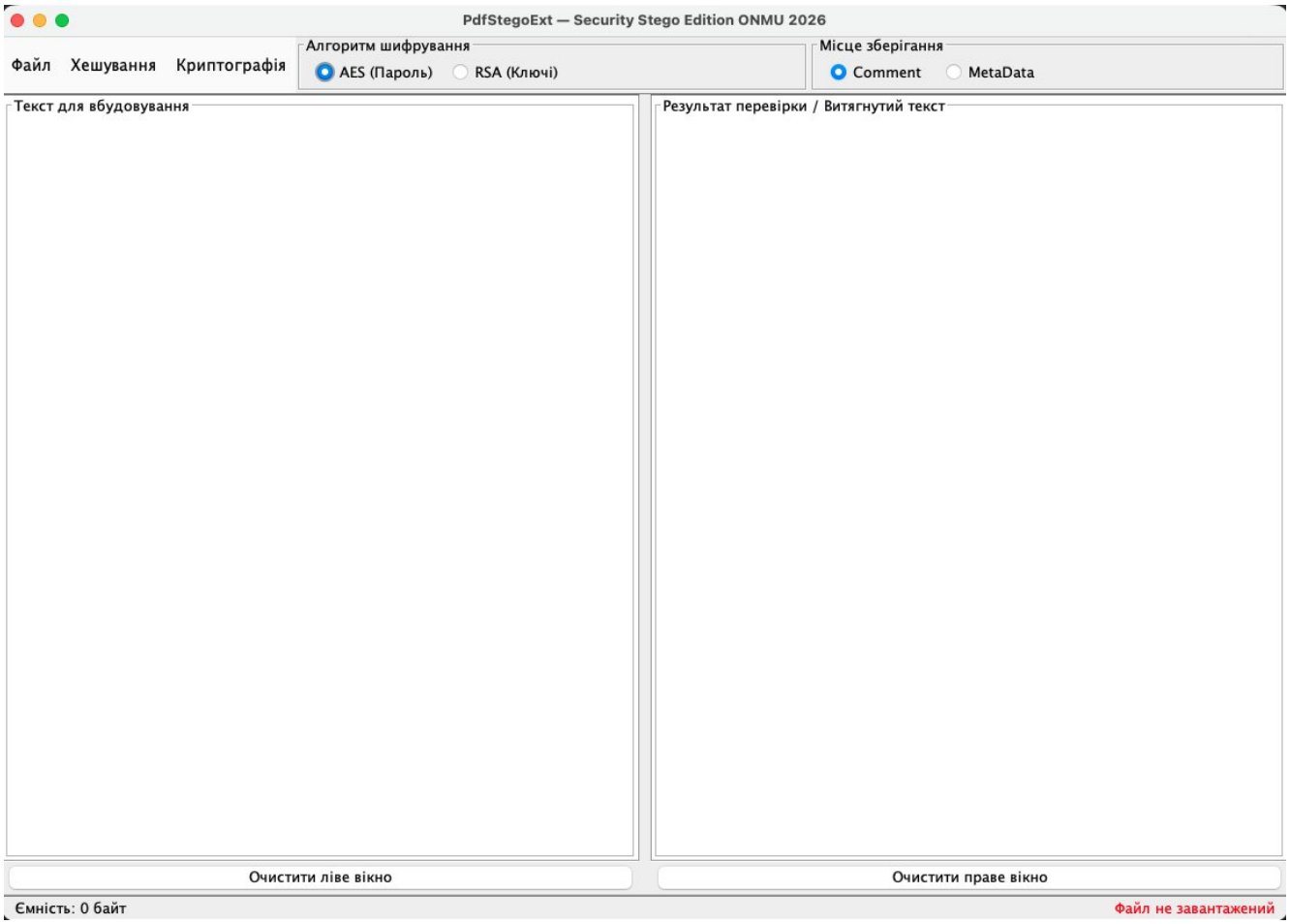

Рисунок 3.8 – Головне меню додатку

Після ініціалізації та зчитування структури PDF-файлу стає доступною процедура стеганографічного вбудовування повідомлення в тіло документа. Для реалізації цього процесу конфіденційні дані вводяться у ліве текстове поле інтерфейсу, після чого активується функція «Зберегти як (Приховати текст)» (рис. 3.9). Попередньо користувач має визначити параметри криптографічного захисту (метод шифрування) та локалізацію прихованого блоку в структурі контейнера. У разі обрання асиметричного алгоритму RSA обов'язковою передумовою є попередня генерація ключової пари з її подальшим експортом на локальний диск.

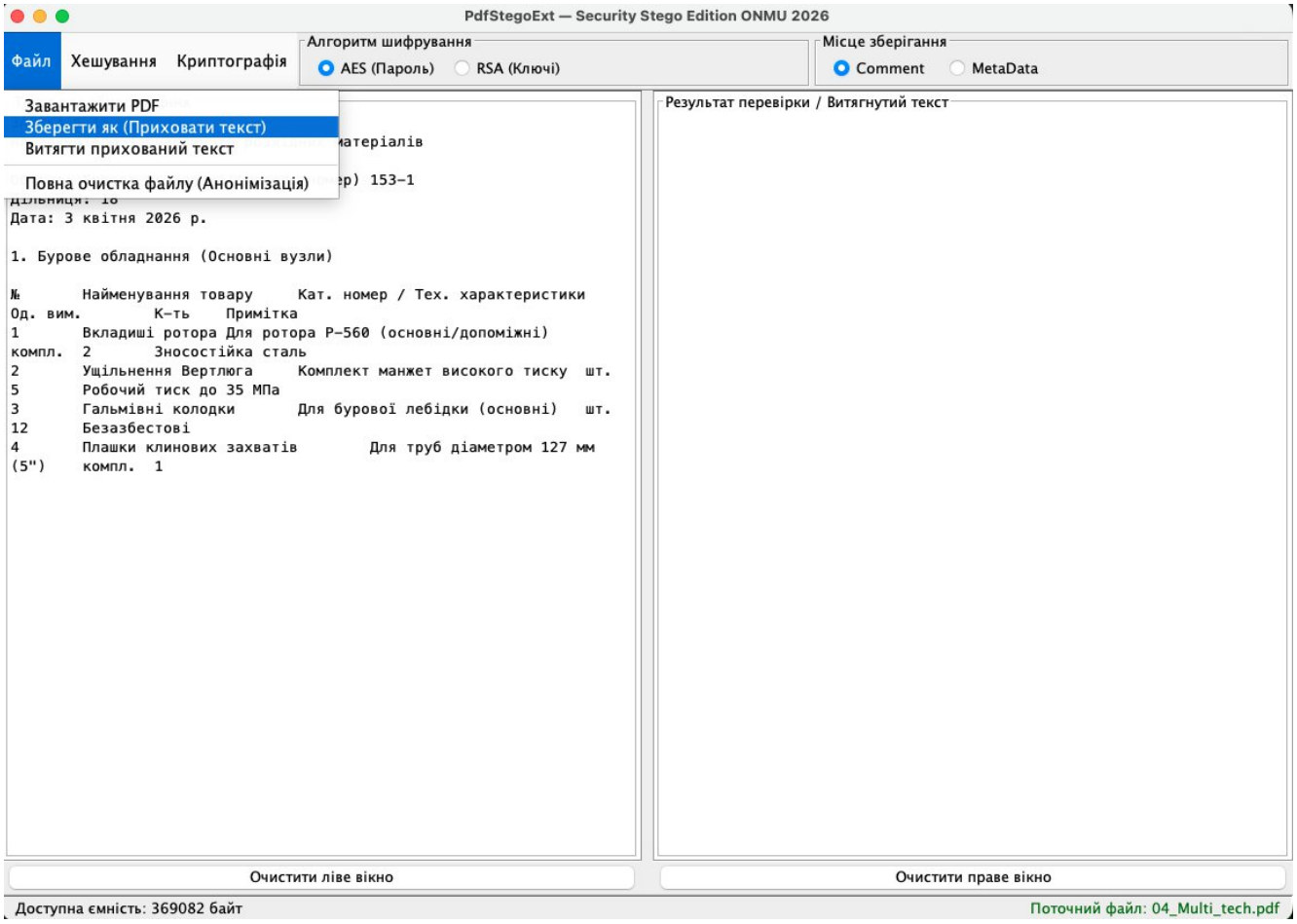

Рисунок 3.9 – Вбудовування прихованого сповіщення

Процедура генерації криптографічних ключів реалізується у розділі «Криптографія» за допомогою активації команди «Згенерувати ключі RSA» (рис. 3.10). На відміну від асиметричного підходу, використання алгоритму AES не потребує генерації статичних ключів — система ініціює діалогове вікно для

введення парольної фрази, яка є чутливою до регістру та мовної розкладки клавіатури.

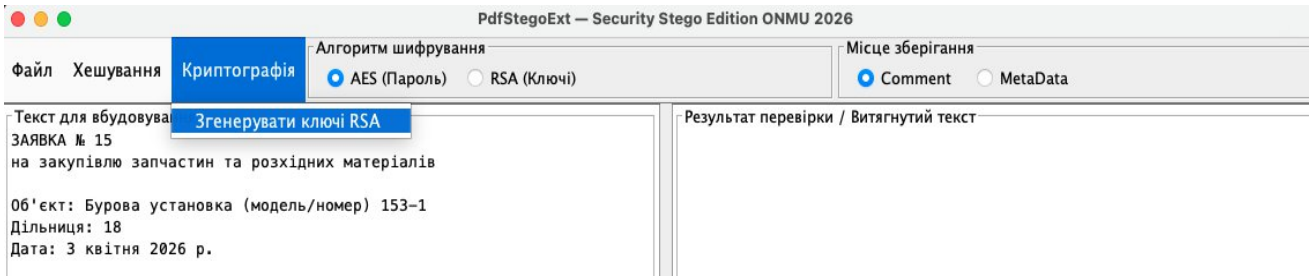

Рисунок 3.10 – Генерація приватного та публічного ключа

Функціональний блок «Хешування» складається з підмодулів «Вбудувати Хеш» та «Перевірити Хеш» (рис. 3.11). Застосування цього інструментарію є доцільним на завершальному етапі стеганографічного перетворення для формування математичного підтвердження цілісності результуючого файлу. Для забезпечення високого рівня криптостійкості процес шифрування обчисленого хеш-рядка має відбуватися або за допомогою альтернативного пароля (при обранні AES), або із застосуванням асиметричного методу RSA. Обов'язковою науково-технічною рекомендацією є розмежування алгоритмів чи ключів шифрування, що застосовуються окремо для самого повідомлення та для його дайджесту.

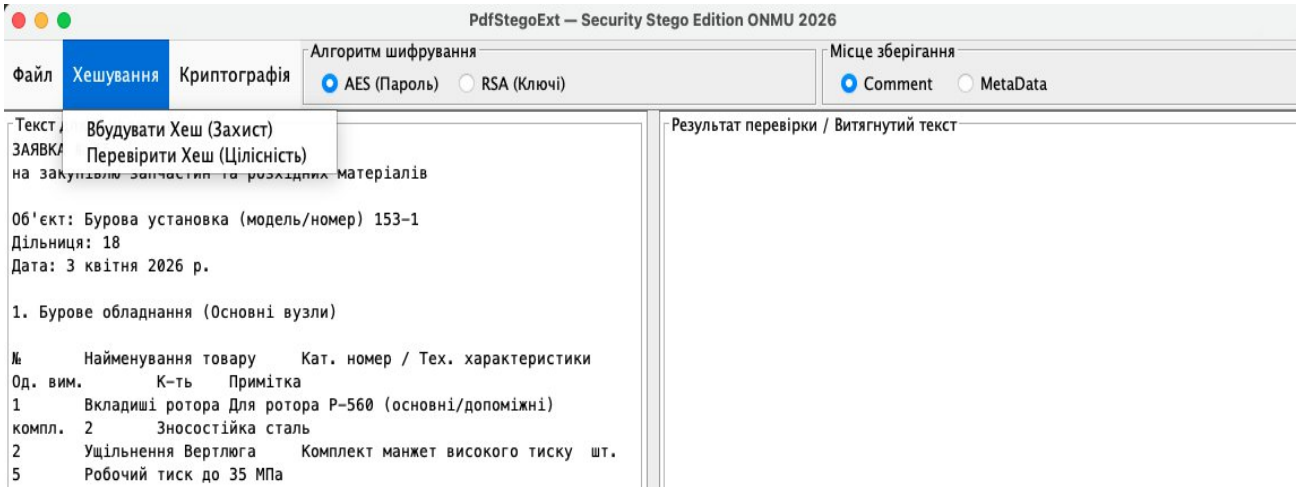

Рисунок 3.11 – Захист вбудованої інформації

Процедура зворотного вилучення прихованої інформації передбачає завантаження стегоконтейнера, вибір відповідного криптографічного алгоритму в панелі налаштувань та активацію команди «Витягти прихований текст» у меню «Файл». За умови автентичності обраного алгоритму та введених ключових даних (або пароля), дешифрований текстовий масив візуалізується у правому діалоговому вікні (рис. 3.12).

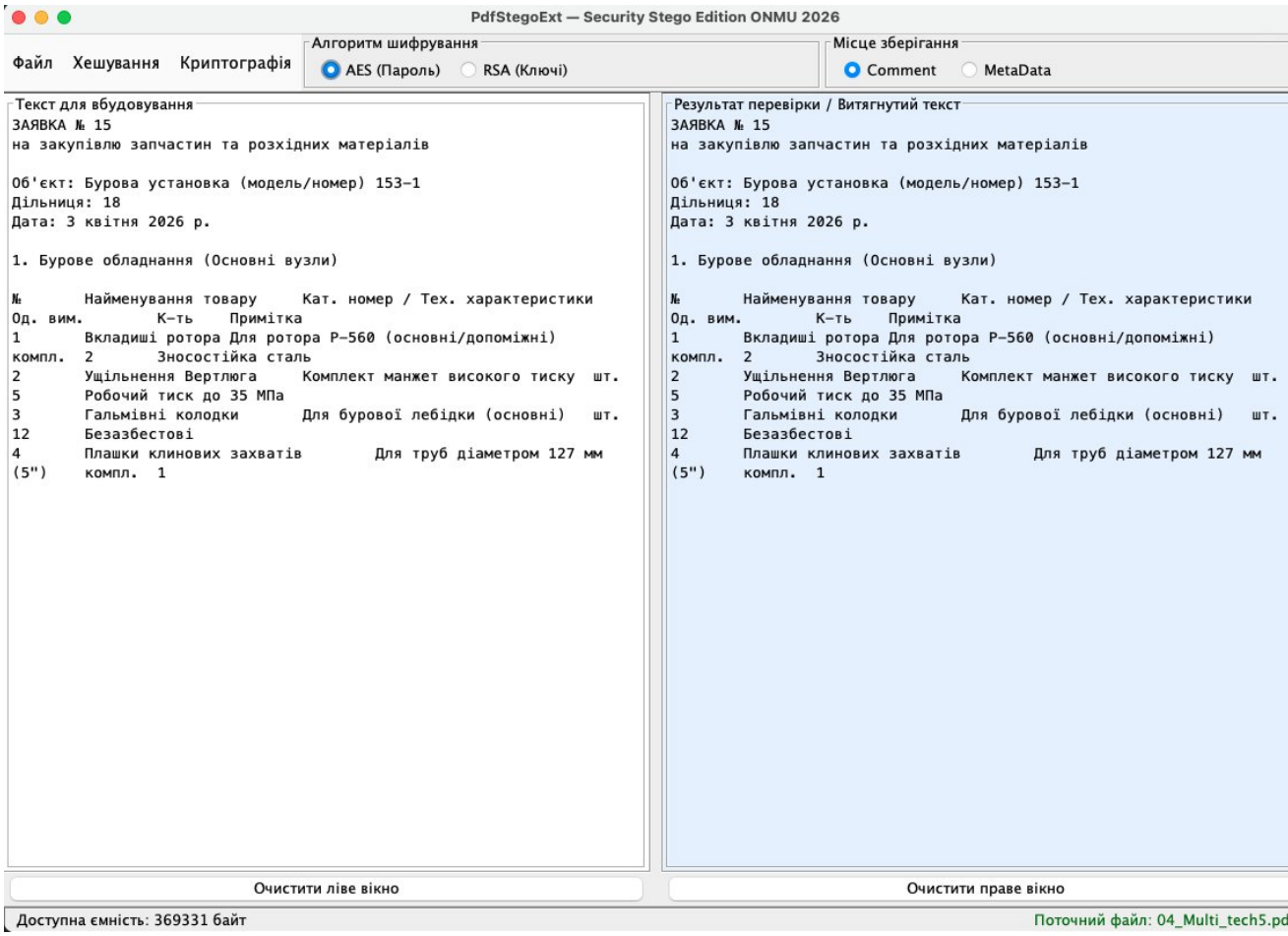

Рисунок 3.12 – Витягування прихованого сповіщення

Після очищення правого текстового поля стає можливим виконання контролю цілісності контейнера з метою виявлення потенційних фактів модифікації даних під час їх транзиту. Для цього після конфігурації параметрів дешифрування у розділі «Хешування» обирається функція «Перевірити хеш». Нормативний результат успішної верифікації наведено на рис. 3.13. Збіг хеш-рядка,

вилученого з файлу, з розрахованим програмою значенням свідчить про автентичність інформації. Для експериментальної перевірки стійкості підсистеми було здійснено штучну модифікацію одного байта даних у тілі прихованого повідомлення за допомогою бінарного редактора. Результати тестування (рис. 3.14) демонструють, що програмний комплекс оперативно фіксує порушення цілісності файлової структури, унеможливлюючи компрометацію та використання модифікованих даних.

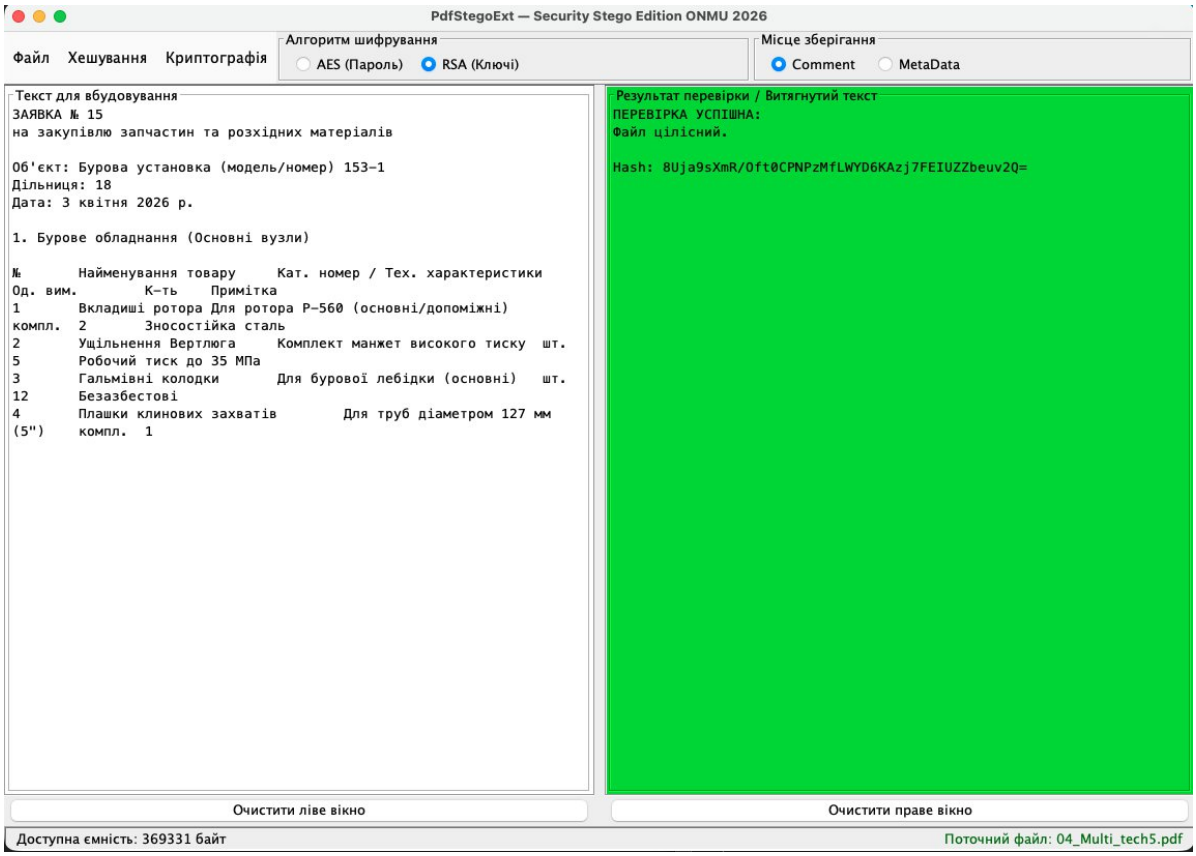

Рисунок 3.13 – Успішна перевірка цілісності файлу

Після очищення правого текстового поля стає можливим виконання контролю цілісності контейнера з метою виявлення потенційних фактів модифікації даних під час їх транзиту. Для цього після конфігурації параметрів дешифрування у розділі «Хешування» обирається функція «Перевірити хеш». Нормативний результат успішної верифікації наведено на рис. 3.13. Збіг хеш-рядка, вилученого з файлу, з розрахованим програмою значенням свідчить про автентичність інформації. Для експериментальної перевірки стійкості підсистеми було здійснено штучну модифікацію одного байта даних у тілі прихованого повідомлення за допомогою бінарного редактора. Результати тестування (рис. 3.14)

демонструють, що програмний комплекс оперативно фіксує порушення цілісності файлової структури, унеможливлюючи компрометацію та використання модифікованих даних.

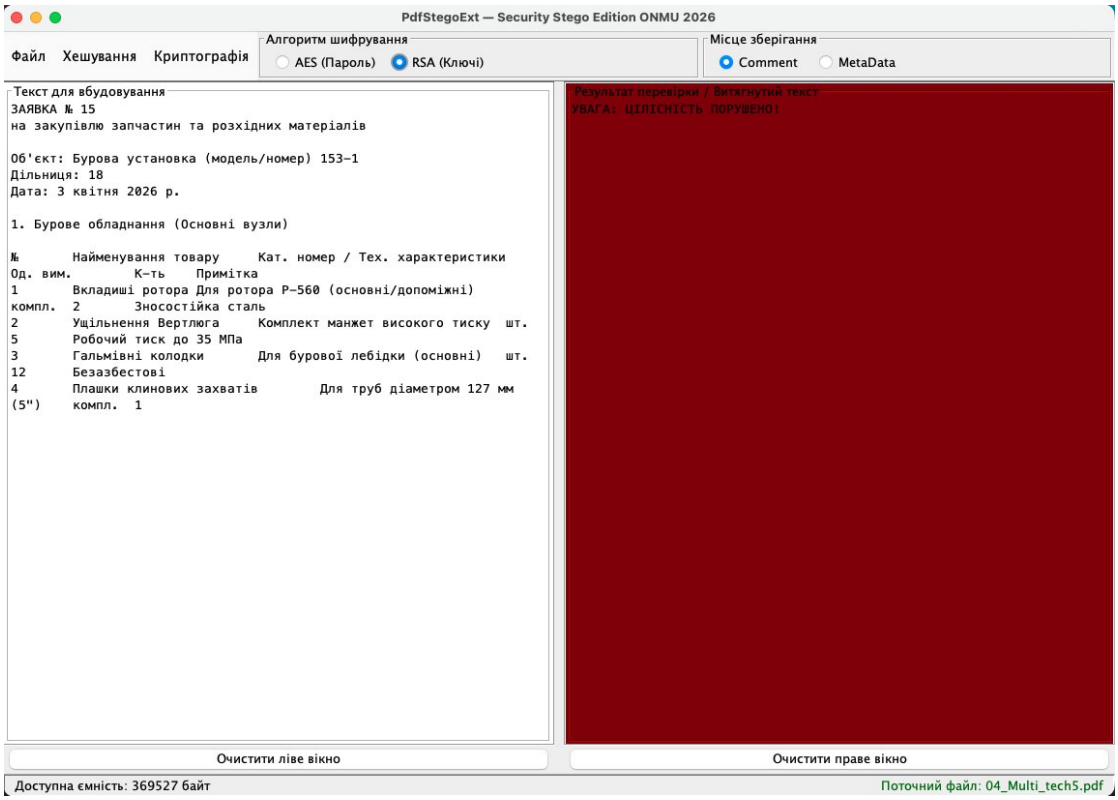

Рисунок 3.14 – Порушення цілісності файлу

Таким чином, розроблене прикладне програмне забезпечення є повнофункціональним інструментом захисту інформації, який забезпечує:

- стеганографічне вбудовування конфіденційних повідомлень як у метадані, так і в область коментарів PDF-документа;
- криптографічний захист прихованих даних із використанням симетричних та асиметричних шифрів;
- генерацію та розподіл ключової пари для асиметричних методів захисту;
- формування та інтеграцію криптографічних дайджестів для контролю цілісності інформаційних масивів;
- повну деструкцію стеганографічних елементів із відновленням початкової бінарної структури файлу-контейнера.

3.5 Можливість використання застосунку під час військового стану

В умовах ведення сучасних бойових дій та дії правового режиму воєнного стану використання стеганографічних методів і засобів прихованої передачі даних є критично важливим аспектом забезпечення інформаційної безпеки, захисту військових підрозділів, а також об'єктів критичної та цивільної інфраструктури. Практичне впровадження розробленого програмного комплексу дозволяє нівелювати ризики перехоплення, деанонімізації та несанкціонованого доступу до вразливої інформації.

Визначено такі основні напрями застосування розробленого стеганографічного інструментарію:

Організація прихованого зв'язку між військовими підрозділами: забезпечення конфіденційного обміну тактичними даними, зокрема координатами цілей. Використання відкритих або слабко захищених каналів зв'язку створює ризики перехоплення інформації силами противника, що може призвести до зриву операцій або перегрупування ворожих сил.

Захист персональних та облікових даних особового складу: приховування відомостей щодо чисельності підрозділів, санітарних та безповоротних втрат, дислокації, а також графіків відпусток і відряджень, що унеможливлює використання цих даних ворогом для ведення розвідки та проведення інформаційно-психологічних операцій (ІПСО).

Захист інформації про стан об'єктів критичної інфраструктури: приховування оперативних даних щодо ступеня пошкодження енергетичних, транспортних чи логістичних вузлів внаслідок вогневого ураження, термінів проведення аварійно-відновлювальних робіт, а також специфікації необхідних матеріально-технічних ресурсів.

Маскування відомостей про руйнування цивільної інфраструктури: обмеження відкритого доступу до деталізованої інформації про масштаби пошкоджень житлового фонду, втрати серед цивільного населення, локалізацію

пожеж та обсяги знищеної техніки, що заважає противнику проводити коригування та оцінювати ефективність нанесених ударів.

Забезпечення конфіденційності звітності державних та стратегічних приватних підприємств: сучасні аналітичні системи на базі штучного інтелекту здатні агрегувати дані з відкритих джерел (обсяги виробництва, рівень енергоспоживання, логістичні маршрути, використання паливно-мастильних матеріалів) для формування точних висновків щодо загального стану економіки держави та військово-промислового комплексу зокрема. Стеганографічне приховування підсумкової та поточної звітності суттєво знижує розвідувальний потенціал держави-агресора.

4. ОХОРОНА ПРАЦІ

Для забезпечення безпечних і комфортних умов роботи фахівців галузі цифрових технологій необхідна ретельна ідентифікація та оцінка всіх шкідливих і небезпечних виробничих факторів і ризиків, які вони несуть для них. Крім того, щоб уникнути негативних наслідків для здоров'я фахівців IT- галузі слід своєчасно проводити відповідні профілактичні заходи. Тому в цьому розділі будуть описані актуальні шляхи ефективного вирішення деяких вищевказаних проблем.

4.1 Перспективи використання штучного інтелекту для зниження рівня виробничих небезпек.

Еволюція попереднього аналізу небезпек у виробничому середовищі та трансформаційна роль штучного інтелекту.

Традиційні підходи до охорони праці історично базувалися на реагуванні на інциденти, що вже відбулися, зосереджуючись на виправленні наслідків, а не на їх запобіганні. Однак, сучасні методології охорони праці наголошують на проактивному виявленні та оцінці ризиків, що є фундаментальним для забезпечення безпечних та здорових умов праці. Метою оцінки ризиків є допомога роботодавцям у ефективному вжитті заходів для захисту безпеки та здоров'я працівників, що є постійним та організованим процесом.

Основні етапи попереднього аналізу небезпек включають виявлення самих небезпек, визначення осіб, які можуть зазнавати ризику, оцінювання рівня цього ризику (як якісне, так і кількісне), розгляд можливості усунення причин небезпек, та прийняття рішення про необхідність запровадження подальших заходів для попередження чи зменшення ризику від них. Виявлення небезпек передбачає ретельний аналіз робочих умов, обладнання, матеріалів та технологій, що використовуються на робочому місці. Після ідентифікації небезпек проводиться оцінка ризиків, яка включає визначення ймовірності настання події та серйозності її наслідків.

Після оцінки ризиків розробляються та впроваджуються заходи контролю та запобіганню, Особлива увага приділяється усуненню або запобіганню ризикам як пріоритетним діям. Процес оцінки небезпек не є одноразовим заходом; він вимагає постійного моніторингу та перегляду, особливо у разі зміни технологій чи умов праці. Методичні підходи до визначення ризику охоплюють широкий спектр аналізів, включаючи загальний аналіз ризику складних систем, якісний аналіз небезпек, попередній аналіз небезпек (ПАН), аналіз дерева відмов (АДВ) та багато інших.

ПАН є ключовим компонентом проактивного управління безпекою. Це систематична ідентифікація та оцінка потенційних небезпек на ранніх стадіях проєкту, системи чи операції. Цей проактивний підхід допомагає оцінювати, ідентифікувати та відстежувати потенційні небезпеки до того, як вони стануть проблемами, що значно зменшує кількість інцидентів на робочому місці.

Ризикоорієнтований підхід є фундаментальним у сучасній охороні праці, вимагаючи від роботодавців забезпечення безпеки та здоров'я працівників у кожному аспекті, пов'язаному з роботою. Це включає виявлення небезпек, оцінку пов'язаних з ними ризиків та визначення необхідних заходів захисту. Оцінка ризиків повинна відповідати вимогам законодавства, стандартам та настановам, таким як національні технічні інструкції та зводи практичних правил. Серед них слід виділити Стандарт ISO 45001, що визначає систематичний підхід до ідентифікації, оцінки та пом'якшення ризиків для захисту працівників та Постанову Кабінету Міністрів України від 26.10.2011 № 1107, яка передбачає цифровізацію порядку видачі дозволів.

Поява штучного інтелекту (ШІ) як ключового фактора, що значною мірою змінює концерцію безпеки праці, знаменує собою значний крок уперед. ШІ та цифровізація кардинально змінюють робочі місця та робочі процеси, пропонуючи нові можливості для підвищення рівня безпеки. Важливо розуміти, що ШІ є доповненням до людського інтелекту, а не його заміною, оскільки він не має емоцій, сумнівів чи свідомості. Здатність ШІ прогнозувати потенційні небезпеки до їх прояву революціонізує управління безпекою, дозволяючи ідентифікувати та

пом'якшувати нещасні випадки ще до їх виникнення. ШІ може аналізувати величезні обсяги даних швидко та точно, що робить його безцінним інструментом для підвищення безпеки та забезпечення відповідності нормативним вимогам. Це дозволяє організаціям переходити від реактивного до проактивного управління ризиками, що є фундаментальною зміною у філософії безпеки праці.

ШІ, завдяки своїй здатності до прогнозного аналізу та моніторингу в реальному часі, дозволяє не просто виявляти, а передбачати небезпеки до їх виникнення. Цей підхід не лише зменшує кількість інцидентів та травм, але й змінює роль фахівців з охорони праці з "пожежників", які реагують на кризи, на "архітекторів безпеки", що дозволяє їм зосередитися на стратегічному плануванні та постійному вдосконаленні систем безпеки.

Оцінка ризиків в охороні праці вимагає збору різноманітних даних, що стосуються умов праці, обладнання, матеріалів, інцидентів, а також законодавчих вимог та стандартів. Ці дані часто є розрізненими та зберігаються в різних форматах, що ускладнює їхній комплексний аналіз. ШІ, завдяки своїй здатності ефективно обробляти величезні обсяги даних з різних джерел, пропонує рішення цієї проблеми. Це відкриває шлях до створення єдиних, інтегрованих платформ для управління даними з охорони праці, які раніше були розрізненими. Стандартизація та агрегація даних через ШІ дозволить отримати більш повну та точну картину ризиків, що є критично важливим для прийняття обґрунтованих рішень та забезпечення послідовності у застосуванні заходів безпеки.

Штучний інтелект у попередньому аналізі небезпек: Основні принципи та практичні застосування.

Фундаментом ШІ-орієнтованого прогнозування небезпек є надійний збір та інтеграція даних. ШІ-системи здатні обробляти величезні обсяги даних, включаючи зображення, відео та вхідні дані з сенсорів. Нові застосування сенсорних технологій дозволяють швидше збирати та передавати дані від широкого кола різних джерел, забезпечуючи оперативне реагування на відхилення від норми та запобігання небезпечним ситуаціям.

Пристрої Інтернету речей (IoT) та сенсорні мережі, інтегровані з ШІ, можуть безперервно відстежувати умови навколишнього середовища та стан обладнання. Це включає моніторинг температури, вологості, концентрації шкідливих газів та рівня шуму, що дозволяє виявляти потенційні загрози в реальному часі. Дані також збираються з архівних записів, таких як минулі інциденти та аудити безпеки.

Спеціалізоване програмне забезпечення для управління інцидентами автоматично збирає дані про інциденти та зберігає їх у централізованому сховищі, що спрощує аналіз та виявлення закономірностей. Людський фактор також є важливим джерелом даних. Інформація збирається через спостереження, консультації та опитування працівників, які часто мають безпосередні знання про ризики на робочому місці.

Прогнозний аналіз небезпек використовує алгоритми ШІ для прогнозування потенційних загроз. Він аналізує великі масиви даних з безпеки, виявляючи закономірності та передбачаючи потенційні небезпеки, що дозволяє роботодавцям усувати ризики до виникнення інцидентів. Машинне навчання (МН) є основним компонентом ШІ в управлінні ризиками. Його моделі навчаються на історичних даних для розпізнавання моделей ризиків, прогнозування результатів та рекомендації запобіжних дій. Це дозволяє прогнозувати високоризикові ситуації та пропонувати превентивні заходи.

Моніторинг у реальному часі та автоматизоване виявлення небезпек є ще однією важливою сферою застосування ШІ. Це дозволяє ідентифікувати потенційні небезпеки до того, як вони призведуть до нещасних випадків, забезпечуючи швидкі коригувальні дії.

Комп'ютерний зір, зокрема, використовує ШІ-камери, які безперервно сканують робоче середовище для виявлення небезпечної поведінки, наприклад, обхід захисних бар'єрів, неправильне використання обладнання або відсутність ЗІЗ. Вони також здатні виявляти падіння, контролювати безпечні відстані від машин, перевіряти цілісність захисних огорож та належне розміщення знаків безпеки. Носійні пристрої, такі як інтелектуальні шоломи, жилети та браслети, інтегровані з ШІ та IoT, відстежують життєві показники працівників (пульс,

температура, рівень втоми), виявляють токсичні гази, моніторять поставу та фізичне навантаження. Ці пристрої можуть попереджати працівників про потенційні ризики для здоров'я в реальному часі.

Інтелектуальні системи також можуть миттєво аналізувати дані та вимикати машини у разі несправності або виявлення небезпеки, мінімізуючи ймовірність нещасних випадків. ШІ може видавати миттєві оповіщення, якщо працівник заходить у заборонену зону або не носить належного спорядження. Крім того, автономні роботи можуть патрулювати робоче місце, шукаючи небезпеки, такі як розливи або несправне обладнання, повідомляючи про проблеми, запускаючи сигнали тривоги або викликаючи допомогу.

Автоматизація та оптимізація процесів за допомогою ШІ значно зменшує вплив людського фактора та підвищує ефективність рутинних завдань, а також покращує ідентифікацію небезпек за допомогою прогнозної аналітики. Це значно зменшує адміністративне навантаження на фахівців з охорони праці, дозволяючи їм зосередитися на більш стратегічних завданнях. Автоматизовані системи працюють безперервно, обробляючи дані з IoT-пристроїв (сенсорів, камер) миттєво. Це зменшує кількість помилок, спричинених людським фактором, оскільки ШІ не відволікається, не втомлюється та не піддається емоціям. ШІ може оптимізувати робочі процеси, запобігати перевтомі та оцінювати ризики перевантаження персоналу. Автоматизація небезпечних завдань за допомогою роботів зменшує ризик для людей, підвищуючи при цьому операційну ефективність та точність.

Ефективний попередній аналіз небезпек з використанням ШІ вимагає інтеграції та взаємодії кількох ШІ-інструментів для отримання повного та точного уявлення про ризики. Збір даних відбувається з різноманітних джерел, включаючи сенсори, пристрої Інтернету речей (IoT), звіти про інциденти та людський ввід. Ці дані потім обробляються за допомогою різних ШІ-технологій, таких як прогнозний аналіз, машинне навчання, комп'ютерний зір та обробка природної мови. Носійні пристрої та автономні роботи будуть фізичними проявами цих ШІ-систем, які збирають дані та виконують дії на основі аналізу.

ШІ, завдяки своїй здатності аналізувати величезні обсяги даних та виявляти складні закономірності, може ідентифікувати ризики, які є "невидимими" для людського ока або не є очевидними з ізольованих інцидентів. Це включає ранні ознаки зносу машин, незначні зміни температури або незвичайні вібрації, які можуть вказувати на потенційні проблеми задовго до того, як вони стануть критичними.

Ця здатність ШІ дозволяє компаніям переходити від реагування на очевидні проблеми до виявлення та усунення прихованих або потенційних загроз, які могли б призвести до серйозних інцидентів у майбутньому. Це не тільки підвищує безпеку, але й може призвести до значної економії коштів за рахунок запобігання дорогим поломкам та простоям.

Ключові переваги використання ШІ для попереднього аналізу небезпек на виробництві.

Використання штучного інтелекту для попереднього аналізу небезпек у виробничому середовищі надає численні переваги, що значно підвищують рівень безпеки та ефективність управління. Однією з найважливіших переваг є значне підвищення точності та ефективності оцінки ризиків. ШІ-системи безперервно аналізують величезні обсяги даних для виявлення ризиків та невідповідностей у реальному часі, що значно зменшує ймовірність людської помилки та підвищує загальні результати безпеки.

На відміну від людей, ШІ не відволікається, не втомлюється та не піддається емоціям, що забезпечує точний моніторинг протоколів безпеки. ШІ-інструменти можуть обробляти інформацію способами, недоступними для людей, що дозволяє їм передбачати потенційні небезпеки до їх виникнення. Прогнозні моделі ШІ, навчаючись на нових даних, постійно вдосконалюють свою здатність виявляти ризики, що робить їх більш ефективними з часом.

ШІ дозволяє організаціям переходити від реактивного до проактивного запобігання інцидентам та травмам. Прогнозний аналіз виявляє закономірності та передбачає потенційні небезпеки, дозволяючи вживати превентивних заходів до того, як вони призведуть до нещасних випадків. Моніторинг у реальному часі

дозволяє миттєво виявляти небезпечні умови та поведінку, забезпечуючи швидке втручання та коригувальні дії.

Покращення дотримання нормативних вимог та стандартів безпеки є ще однією значною перевагою. ШІ може автоматизувати моніторинг та звітність, забезпечуючи постійне дотримання стандартів безпеки. Він може відстежувати показники безпеки в реальному часі, виявляти потенційні проблеми з відповідністю та генерувати звіти для регуляторних органів.

ШІ може швидко перевіряти інформацію в інтернеті на наявність новин про оновлення галузевих норм, гарантуючи, що стратегії безпеки базуються на найновіших вимогах. Це зменшує ризик штрафів та санкцій, а також покращує репутацію компанії.

Оптимізація витрат та підвищення загальної продуктивності праці є прямим наслідком впровадження ШІ. Завдяки зменшенню кількості нещасних випадків та травм, ШІ допомагає компаніям економити значні кошти, пов'язані з медичними витратами, компенсаціями працівникам, простоями та юридичними спорами.

Прогнозне технічне обслуговування, що дозволяє замінювати зношені деталі вчасно, допомагає уникнути дорогих ремонтів та простоїв обладнання. ШІ-інструменти можуть оптимізувати робочі процеси, підвищуючи продуктивність, оскільки працівники почуваються безпечніше та менш напружено, що позитивно впливає на їхній психічний стан та загальну атмосферу на робочому місці. Автоматизація рутинних завдань зменшує адміністративне навантаження та вивільняє людські ресурси, дозволяючи фахівцям з охорони праці зосередитися на більш цінних видах діяльності.

Удосконалення програм навчання та підвищення рівня свідомості працівників через інтерактивні симуляції є ще однією вагомою перевагою. ШІ покращує навчання, створюючи імерсивні симуляції та сценарії віртуальної реальності (VR), які відтворюють реальні небезпеки на робочому місці. Це підвищує засвоєння знань та готує працівників до ефективного реагування в надзвичайних ситуаціях. VR-навчання може призвести до підвищення на 30% рівня засвоєння протоколів безпеки порівняно з традиційними методами у

високоризикових галузях. ШІ також може персоналізувати навчальний досвід, адаптуючи рівні складності та сценарії на основі індивідуальної продуктивності та прогалин у знаннях.

Проблеми та та обмеження впровадження штучного інтелекту в систему виробничої безпеки.

Незважаючи на значні переваги, впровадження штучного інтелекту в охороні праці супроводжується низкою проблем та обмежень, які потребують ретельного розгляду.

Етичні питання є одними з найскладніших. Приватність даних викликає серйозне занепокоєння, оскільки ШІ-системи в охороні праці часто покладаються на великі обсяги даних, включаючи особисту інформацію про працівників, записи про безпеку, стан здоров'я та екологічні фактори. Збір, зберігання та обробка цих даних створюють значні проблеми з конфіденційністю та безпекою, вимагаючи суворого дотримання нормативних актів, для уникнення витоків даних, штрафів та судових позовів. Надмірне спостереження за допомогою ШІ-систем, таких як розпізнавання облич та моніторинг поведінки, може призвести до стресу, невдоволенню та юридичних суперечок щодо прав працівників на приватність на робочому місці.

Упередженість алгоритмів є ще однією критичною етичною проблемою. ШІ-системи є настільки неупередженими, наскільки неупередженими є дані, на яких вони навчаються. Якщо навчальні дані містять упередження, пов'язані з статтю, віком чи іншими факторами, ці упередження можуть бути увічнені в процесі прийняття рішень системою. Це може призвести до того, що певні групи працівників будуть непропорційно часто позначатися як порушники правил безпеки або стикатися з дискримінаційними протоколами безпеки. Необхідно впроваджувати активні заходи для пом'якшення упереджень, забезпечуючи справедливість та постійний моніторинг алгоритмів.

Питання відповідальності також є невирішеним. У разі, якщо ШІ-система не зможе виявити небезпеку, зробить неправильний прогноз або надасть неточні рекомендації, що призведе до нещасних випадків, травм або навіть смертельних

випадків, виникають важливі питання щодо підзвітності та юридичної відповідальності. Оскільки ШІ-системи певною мірою працюють автономно, встановлення чітких ліній відповідальності є складним завданням.

Технічні та операційні перешкоди також створюють значні виклики. Якість та доступність даних є першочерговими, оскільки ШІ-системи значною мірою покладаються на високоякісні та неупереджені дані для точних прогнозів та рішень. В промислових умовах дані часто фрагментовані, неповні або низької якості, що ускладнює ефективну роботу ШІ. Необхідно встановлювати надійні процеси збору даних, забезпечуючи їхню послідовність, точність та повноту.

Інтеграція з існуючими системами є ще однією перешкодою. Промислові операції часто використовують застарілі системи, які можуть бути несумісними з сучасними ШІ-технологіями. Інтеграція ШІ в ці старі системи без порушення повсякденних операцій може бути складною, вимагаючи поетапного підходу та, можливо, модернізації всієї IT-архітектури.

Вартість впровадження ШІ також є значним бар'єром, особливо для малих та середніх підприємств. Витрати включають програмне забезпечення, апаратне забезпечення (сенсори, носійні пристрої), інтеграцію з існуючими системами та навчання персоналу. Початкові витрати на розробку моделей машинного навчання можуть починатися від 50 000 доларів США, а складніші системи можуть коштувати сотні тисяч доларів.

Людський фактор відіграє важливу роль у успішному впровадженні ШІ. Опір змінам з боку працівників, які побоюються втрати робочих місць або не повністю розуміють, як ШІ може покращити їхні ролі, є поширеним явищем. Працівники також можуть бути скептично налаштовані щодо надійності та прозорості ШІ-систем безпеки, особливо у високоризикових середовищах. Необхідно залучати працівників до процесу на ранніх етапах, чітко роз'яснювати роль ШІ у підвищенні безпеки, а не у заміщенні робочих місць.

Важливо знайти баланс між автоматизацією та людським наглядом. Хоча ШІ може підвищити безпеку, він не повинен повністю замінювати людське судження або нагляд. Існує ризик надмірної залежності від технологій, що може призвести

до ігнорування проблем, які виходять за межі алгоритму. ШІ повинен підтримувати, а не замінювати роль фахівців з охорони праці, забезпечуючи людський контроль у прийнятті важливих рішень з безпеки.

Нормативно-правова невизначеність також є значним викликом. Відсутність всеосяжних регуляторних рамок для ШІ у сфері охороні праці призводить до прогалин у нагляді та непослідовних практик. Існуючі нормативні акти можуть бути недостатніми для вирішення юридичних наслідків ШІ-систем, особливо коли ШІ бере на себе більше ролей у прийнятті рішень. Необхідні чіткі стандарти щодо людського нагляду, прозорості систем та процедур аудиту. Регуляторні органи, вже вимагають від роботодавців оцінювати ризики, пов'язані з ШІ, та впроваджувати ефективні заходи контролю. Однак, загальний вплив ШІ на регулювання та стандарти безпеки ще не повністю реалізований.

Перспективи розвитку штучного інтелекту у системі забезпечення виробничої безпеки.

Перспективи розвитку штучного інтелекту для забезпечення виробничої безпеки є багатообіцяючими. Майбутнє управління ризиками в системе охороні праці передбачає ще глибшу інтеграцію ШІ з передовими технологіями, такими як цифрові двійники, квантові обчислення та блокчейн.

Цифрові двійники, дозволяють симулювати сценарії "що-якщо", візуалізувати потенційні ризики та оптимізувати заходи безпеки в контрольованому середовищі. Квантові обчислення зможуть значно прискорити та розширити можливості моделювання ризиків, дозволяючи обробляти безпрецедентні обсяги даних для більш точних прогнозів. Поєднання ШІ з технологіями блокчейн може підвищити прозорість та безпеку практик управління ризиками, забезпечуючи незмінність та достовірність даних про безпеку.

Подальший розвиток прогнозної аналітики та рецептивних рішень є ключовим напрямком. ШІ вже виходить за рамки простого прогнозування, пропонуючи оптимальні дії для запобігання або пом'якшення ризиків. Прогнозний аналіз, що використовує алгоритми ШІ для прогнозування ризиків, стає все більш точним, дозволяючи організаціям адаптувати свої професійні стандарти на основі

статистичних ймовірностей, а не лише інтуїції. Це означає, що ШІ не тільки вказуватиме на потенційні проблеми, але й пропонуватиме конкретні, обґрунтовані дії для їх вирішення, оптимізуючи ресурси та впроваджуючи проактивні рішення для мінімізації загроз.

Фахівці з охорони праці замість того, щоб бути переважно контролюючими органами, будуть переорієнтовани на стратегічне управління, аналіз складних даних та розробку інноваційних рішень. ШІ автоматизуватиме рутинні завдання, звільняючи фахівців для зосередження на більш цінних видах діяльності, таких як розробка програм безпеки, навчання та підвищення рівня свідомості працівників. Це вимагатиме від фахівців з охорони праці розвитку нових навичок у галузі аналізу даних, управління ШІ-системами та міждисциплінарної співпраці.

Впровадження ШІ не лише підвищує рівень безпеки праці та зменшує кількість нещасних випадків, але й оптимізує витрати, покращує дотримання нормативних вимог та підвищує загальну продуктивність праці, а інтерактивні симуляції та персоналізовані навчальні програми значно покращують підготовку працівників.

4.2 Вплив ризиків пов'язаних з переробками на здоров'я і продуктивність праці фахівців-комп'ютерників.

Переробка визначається як постійна робота за межами стандартних або здорових лімітів, що часто призводить до фізичного виснаження, психічного вигорання та загального зниження добробуту. Вона характеризується тривалими робочими годинами, нездатністю відключитися від роботи, роботою під час перерв та відчуттям тиску, як зовнішнього з боку керівництва, так і внутрішнього через занепокоєння щодо втрати роботи або надмірну пристрасть до кар'єрного зростання. Важливо розуміти, що переробка це не просто збільшення кількості відпрацьованих годин, а скоріше неадекватна інтенсивність та нестійкість робочого режиму відносно фізичних та емоційних можливостей людини.

Переробки в IT-сфері, тобто робота понаднормово, є поширеним явищем, і хоча іноді вони здаються неминучими для досягнення цілей проекту, їхній вплив на здоров'я та продуктивність IT-фахівців може бути руйнівним.

Поширеність переробок в IT-індустрії є не просто статистичним фактом, а відображенням глибоко вкоріненої проблеми, яка часто залишається непоміченою. Той факт, що значна частина понаднормової роботи є неоплачуваною, ускладнює її відстеження та вирішення, оскільки вона не фіксується офіційно. Крім того, працівники можуть відчувати внутрішній тиск, щоб "довести свою цінність" або ж бути настільки "захопленими" своєю кар'єрою, що ігнорують негативні наслідки переробок.

Переробки в IT-індустрії є результатом складної взаємодії галузевих особливостей, недоліків управління проєктами, а також організаційної культури та керівництва.

Основними з них є.

Дефіцит кадрів та високий попит. Технологічна галузь стикається зі значною нестачею талантів, де попит на кваліфікованих фахівців часто перевищує пропозицію. Коли не вистачає ресурсів, компанії часто не наймають додаткових людей, а замість цього тиснуть на існуючих працівників, щоб вони працювали більше.

Швидкий темп технологічних змін та вимоги до безперервного навчання. Невпинна еволюція технологій, включаючи штучний інтелект та автоматизацію, вимагає постійного підвищення кваліфікації та адаптації. Ця вимога до безперервного навчання додає значного тиску на працівників, які вже керують існуючим навантаженням, сприяючи "нескінченному циклу стресу та виснаження".

Середовище високого тиску. Внутрішній імпульс технологічної галузі до інновацій та прагнення до "наступних великих проривів" створюють середовища, де від працівників очікується, що вони будуть надавати інновації та результати зі швидкістю, яка посилює рівень стресу. Це сприяє високій поширеності проблем психічного здоров'я: понад половина (52%) технічних працівників відчувають депресію або тривогу.

"Культура поспіху" та прославлення переробок. Поширена культурна норма, яка прославляє довгі години роботи, постійну продуктивність та жертвування турботою про себе як ознаки честі та шляхи до успіху. Ця ідеологія "токсичної продуктивності", часто призводить до вигорання та невдоволення, а не до бажаного успіху.

Вимоги клієнтів щодо швидкості та доступності. Зовнішній тиск з боку клієнтів, які очікують швидкої доставки та постійної доступності, значно сприяє тривалим робочим годинам. Фірми все частіше просувають свою "доступність" та "швидкість", перекладаючи цей тиск на IT-команди.

Жорсткі терміни. Постійний тиск, пов'язаний з дотриманням агресивних та часто нереалістичних термінів, є основним фактором, що сприяє переробкам. Жорсткі терміни постійно призводять до підвищеного стресу, вигорання, компромісної якості та збільшення кількості помилок.

Нереалістичні очікування та недостатні ресурси. Роботодавці часто встановлюють високі очікування, обмежуючи при цьому доступні ресурси, створюючи тривожний дисбаланс, який безпосередньо сприяє надмірному навантаженню.

Вплив переробок на здоров'я IT-фахівців

Переробки мають глибокі та руйнівні наслідки для психічного та фізичного здоров'я IT-фахівців, часто призводячи до хронічних станів, які можуть мати довгострокові наслідки.

1. Наслідки для психічного здоров'я

Вигорання: Це стан хронічного фізичного та емоційного виснаження, спричинений тривалим стресом, пов'язаним з роботою. Симптоми включають відчуття втоми, цинізм, відстороненість від роботи, знижену мотивацію, дратівливість та переважне відчуття емоційного та фізичного виснаження. Вигорання значно збільшує ймовірність того, що працівники шукатимуть нову роботу (у 2,6 рази частіше).

Стрес, тривога та депресія: IT-фахівці постійно повідомляють про вищий рівень стресу та тривоги через тривалі робочі години та постійний тиск, щоб

працювати та впроваджувати інновації. Переробка значно збільшує ризик розвитку клінічної депресії та генералізованих тривожних розладів.

Когнітивні порушення: Надмірна робота без достатнього відпочинку серйозно погіршує когнітивні функції, що призводить до труднощів з концентрацією, проблем з пам'яттю (навіть дрібних деталей, таких як імена або дати), зниження здатності приймати рішення та збільшення кількості помилок. Це може проявлятися як "туман у мозку" через підвищений рівень кортизолу.

Розлади сну та безсоння: Переробки часто призводять до нерегулярного режиму сну, труднощів із засинанням або підтриманням сну через нав'язливі думки, стрес, пов'язаний з роботою, або надмірний вплив екранів. Хронічне недосипання ще більше посилює стрес, психічне виснаження та погіршує настрій та концентрацію.

2. Наслідки для фізичного здоров'я

Серцево-судинні ризики: Тривалий стрес та переробки значно підвищують ризик високого кров'яного тиску , серцевих захворювань та інсульту. Дослідження показують, що навіть робота лише 3-4 години понаднормово на день збільшує ризик серцевих проблем на 60%. Робота 55 годин або більше на тиждень пов'язана з 35% підвищеним ризиком інсульту та 17% підвищеним ризиком смерті від ішемічної хвороби серця.

Проблеми опорно-рухового апарату (ергономічні ризики): Сидячий характер роботи в IT, у поєднанні з тривалими годинами, неправильною поставою (наприклад, сутулість над ноутбуками) та повторюваними рухами, призводить до травм від повторюваних навантажень (RSI), таких як синдром зап'ястного каналу (біль у зап'ясті/руці, оніміння, поколювання) та тендиніт (запалення, часто в зап'ястях, ліктях або плечах). Хронічний біль у спині та шиї також є поширеним явищем через тривале сидіння на непідтримуючих стільцях або витягування шиї до неправильно розташованих моніторів. Ці початкові болі можуть перерости в хронічний, виснажливий біль.

Шлунково-кишкові проблеми та ослаблена імунна система: Хронічний стрес від переробок може проявлятися фізично, призводячи до шлунково-кишкових

проблем та зниження функції імунної системи, що робить людей більш сприйнятливими до хвороб та "загострень" існуючих станів.

Використання перспективних комп'ютерних технологій у сфері охорони праці та організація раціонального режиму робочого дня спепціалістів дозволять зберегти їх здоров'я та високу продуктивність протягом усього періоду їхньої трудової діяльності

ВИСНОВКИ

У межах виконання кваліфікаційної роботи розв'язано актуальну дослідницько-практичну задачу забезпечення передачі конфіденційної інформації шляхом використання файлів формату PDF як стеганографічних контейнерів. Під час дослідження проведено ретроспективний аналіз еволюції стеганографічних методів та еволюції типів носіїв прихованих даних. Під час виконання роботи було проведено аналіз сучасних методів комп'ютерної стеганографії, досліджено принципи функціонування стеганографічних систем та виконано порівняння основних типів стеганографічних контейнерів. Встановлено, що PDF-файли мають значний потенціал для прихованого розміщення інформації завдяки своїй складній структурі, значній ємності, поширеності та стійкості до випадкового пошкодження.

Актуальність роботи обумовлена необхідністю підвищення рівня захисту конфіденційних даних в умовах зростання кількості кіберзагроз та широкого використання електронного документообігу.

Досліджено внутрішню структуру PDF-документів та визначено основні елементи, придатні для вбудовування прихованих повідомлень. Встановлено можливість використання коментарів, метаданих, потоків даних, службових об'єктів та області після ознаки кінця файлів для організації прихованих каналів передачі інформації.

Проведено аналіз існуючих методів стеганографічного вбудовування інформації у PDF-файли та визначено їх переваги і недоліки. Встановлено, що використання метаданих та коментарів забезпечує просту реалізацію механізмів приховування даних, тоді як модифікація потоків даних та внутрішніх структур документа дозволяє підвищити рівень прихованості інформації.

Результати аналізу сучасних цифрових стеганографічних контейнерів свідчать, що формат PDF має високий потенціал щодо інформаційної місткості та варіативності методів інтеграції прихованих повідомлень. Інтенсивний документообіг у форматі PDF між користувачами корпоративних і приватних мереж робить цей тип файлів поширеним та легітимним елементом транзиту даних

через канали електронної пошти та сучасні месенджери. На відміну від масової передачі графічних об'єктів, транспортування значної кількості PDF-документів є типовим процесом і не викликає підозр з боку систем моніторингу мережевого трафіку (DPI тощо).

Додатковою перевагою обраного типу контейнера є його стійкість до деструктивних впливів під час транспортування. На відміну від растрових зображень, які в месенджерах часто піддаються процедурам стиснення з втратами (компресії), що повністю руйнує вбудовану стеганографічну інформацію, файли формату PDF зберігають свою бінарну та об'єктну структуру незмінною.

У роботі розроблено алгоритми приховування та вилучення інформації з PDF-контейнерів, а також реалізовано механізми криптографічного захисту повідомлень із застосуванням сучасних алгоритмів симетричного та асиметричного шифрування. Додатково впроваджено засоби контролю цілісності даних для виявлення несанкціонованих змін прихованого повідомлення або контейнера.

Практичним результатом роботи стало створення програмного комплексу, який забезпечує вбудовування, шифрування, вилучення та перевірку цілісності прихованих повідомлень у PDF-файлах. Розроблене програмне забезпечення підтвердило працездатність запропонованих підходів та можливість їх використання в реальних умовах експлуатації. Є можливість використання розроблених алгоритмів та програмного комплексу для захисту конфіденційної інформації в державних установах, корпоративних інформаційних системах, наукових організаціях, а також у військовій сфері та системах захищеного документообігу.

Теоретичне значення роботи полягає в узагальненні та розвитку підходів до використання PDF-документів як стеганографічних контейнерів, а також у систематизації методів приховування інформації в структурованих електронних документах.

Таким чином, поставлена мета роботи досягнута, а всі визначені завдання виконані в повному обсязі. Результати дослідження підтверджують доцільність

використання PDF-файлів як ефективних стеганографічних контейнерів та відкривають перспективи для подальших досліджень у напрямку підвищення стійкості, прихованості та криптографічного захисту вбудованої інформації.

Розроблений програмний комплекс успішно реалізує функціонал використання PDF-документів як стеганографічних контейнерів та забезпечує виконання таких завдань:

Гнучкість локалізації даних: реалізовано можливість інтеграції прихованих повідомлень як у структуру об'єктів коментарів, так і в спеціалізований розділ метаданих документа.

Стійкість контейнера: забезпечено високу інформаційну місткість та інваріантність стеганографічного вмісту до можливих модифікацій структури файлу на рівні операційної системи.

Оптимізація обсягу даних: з метою запобігання аномальному збільшенню лінійного розміру PDF-файлу після стеганографічного перетворення впроваджено штучне обмеження максимального обсягу вбудованих даних на рівні 15% від первинного розміру файлу. Цей захід мінімізує ризик виявлення стегоконтейнера за допомогою статистичного аналізу його фізичних параметрів.

Контроль цілісності: інтегровано підсистему верифікації на основі обчислення та порівняння криптографічних хеш-рядків, що дозволяє оперативно виявляти факти спотворення або модифікації прихованої інформації.

Комбінований криптографічний захист: для підвищення рівня безпеки повідомлень та супутніх дайджестів застосовано диференційований підхід із використанням симетричних та асиметричних алгоритмів шифрування, що суттєво підвищує загальну стійкість системи до компрометації.

Запропоноване програмне рішення має високий потенціал для практичного впровадження з метою захисту чутливих даних у мілітарній сфері, органах державної влади, на стратегічних підприємствах, а також у межах забезпечення внутрішньої інформаційної безпеки комерційних компаній та корпорацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. https://uk.wikipedia.org/wiki/Стеганографія#:~:text=Стеганогра́фія%20—%20(з%20грец.,приховує%20сам%20факт%20існування%20повідомлення.

2. Даус, Ю. 2026. Використання pdf файлів в якості стеганографічних контейнерів. *Вісник Одеського національного морського університету*. 78 (Січ 2026), 226-240. DOI:https://doi.org/10.47049/2226-1893-2025-4-226-240.

3. Григор'єв, С. В. (2009). Сучасні методи стеганографії та їх застосування. Науковий вісник Херсонського державного університету. Серія: Економічні науки, 3, 175-178.

4. Полевой, С. А., & Григорьев, С. В. (2014). Стеганографія у цифровому просторі: сучасний стан та перспективи розвитку. Інформаційні технології та комп’ютерна інженерія, 1(60), 68-76.

5. Кібкан, В. В. (2011). Стеганографія як метод забезпечення конфіденційності інформації. Інформаційні технології в освіті, науці та виробництві, 5, 38-42.

6. Соловйов, В. І., & Шамраєв, В. Г. (2008). Сучасні аспекти стеганографії. Збірник наукових праць Херсонського державного університету. Серія: Економічні науки, 5, 103-107.

7. Чепур, Л. А., & Сторожук, В. В. (2013). Аналіз та вибір методів стеганографії. Системні дослідження та інформаційні технології, 3, 114-124.

8. Сахно, В. В., & Старцев, В. О. (2015). Методи стеганографії та їх використання для захисту інформації. Збірник наукових праць Харківського національного університету внутрішніх справ, 2(47), 212-220.

9. Яцишин, А. В., & Лунін, С. В. (2012). Сучасні технології стеганографії. Проблеми телекомунікацій, 1, 92-101.

10. Кондратюк, В. В., & Маркова, І. С. (2017). Стеганографія у сучасному інформаційному просторі. Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Радіотехніка, радіоапаратобудування, 1(71), 37-43.

11. Ковальчук, І. В., & Бондар, С. А. (2016). Стеганографічні методи захисту інформації у цифрових зображеннях. Молодь і ринок, 4(142), 57-61.

12. Шинкарук, Л. В., & Стець, М. Л. (2011). Застосування стеганографії для захисту інформації в інформаційно-комунікаційних системах. Вісник Львівського університету. Серія: Інформаційні системи та мережі, 732, 31-38.

13. Горбатенко, О. В., & Клочко, І. О. (2014). Стеганографія в цифрових зображеннях: аналіз методів та застосування. Збірник наукових праць Донецького національного технічного університету, 33, 191-198.

14. Мельник, О. О. (2016). Сучасні аспекти стеганографії та її використання в інформаційних системах. Інформаційні технології та комп’ютерна інженерія, 2(76), 35-40.

15. Петренко, С. І. (2018). Застосування стеганографії для захисту інформації в обчислювальних системах. Теорія і практика сучасних інформаційних технологій, 1, 76-81.

16. Кравченко, О. М. (2015). Стеганографія як інструмент забезпечення безпеки інформації. Системні дослідження та інформаційні технології, 2, 129-134.

17. Куц, О. В., & Бурлака, В. І. (2013). Використання методів стеганографії для забезпечення безпеки інформації. Наукові праці Кам'янець-Подільського національного університету імені Івана Огієнка. Серія: Інформаційні технології та телекомунікації, 1(21), 29-33.

18. Смолярчук, І. О. (2017). Стеганографія: основні принципи та сучасні технології. Науковий вісник Національного університету біоресурсів і природокористування України. Серія: Технічні науки, 270, 33-39.

19. Тарасенко, О. Є., & Гетманець, В. В. (2019). Використання стеганографії для прихованої передачі інформації. Вісник Кіровоградського національного технічного університету. Серія: Технічні науки, 46, 72-78.

20. Шевченко, С. В., & Підгайний, І. В. (2016). Застосування методів стеганографії для забезпечення безпеки інформації. Технічні науки та технології, 1(8), 127-133.

21. Кравець, М. С. (2014). Стеганографія: сучасні тенденції розвитку. Інформаційні технології та комп'ютерна інженерія, 3(75), 41-47.

22. Левченко, В. В. (2015). Використання стеганографії для забезпечення конфіденційності даних. Науковий вісник Хмельницького національного університету. Серія: Економічні науки, 2, 123-127.

23. Коровяков, А. А. (2017). Стеганографія в області захисту інформації: аналіз та практичне застосування. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, 880, 183-188.

24. Беспалов, А. І., & Пилипенко, В. Ю. (2016). Застосування методів стеганографії для забезпечення безпеки інформаційних ресурсів. Науковий вісник Ужгородського університету. Серія: Математика та інформатика, 1(45), 181-186.

25. Klemm R., Chen B. Hiding Sensitive Information Using PDF Steganography. arXiv preprint arXiv:2405.00865, 2024. Документ доступний: https://arxiv.org/pdf/2405.00865 та https://arxiv.org/html/2405.00865

26. Ndoundam R., Ekodeck S. G. R. PDF Steganography based on Chinese Remainder Theorem. arXiv preprint arXiv:1506.01256, 2015. PDF доступний: https://arxiv.org/pdf/1506.01256 arXiv

27. Koptyra P., Ogiela L. Distributed Steganography in PDF Files — Secrets Hidden in Modified Pages. PMC (PubMed Central). 2021. Доступ: https://pmc.ncbi.nlm.nih.gov/articles/PMC7517136/ PMC+2ResearchGate+2

28. Vulnerability Exploitations Using Steganography in PDF Files. International Journal of Computer Networks and Applications (IJCNA). Доступний PDF: https://www.ijcna.org/Manuscripts/IJCNA-2020-O-02.pdf ijcna.org

29. A new method for pdf steganography in justified texts. ScienceDirect. Доступ: https://www.sciencedirect.com/science/article/abs/pii/S221421261830485X ScienceDirect

30. Maiorca D., Biggio B. Digital Investigation of PDF Files: Unveiling Traces of Embedded Malware. arXiv preprint arXiv:1707.05102, 2017. Доступ: https://a

ДОДАТОК А

Лістінг програми

```
package com.ds;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDDocumentInformation;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.*;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Arrays;
import java.util.Base64;

public class PdfStegoExt extends JFrame {

    //                ──                UI                  поля
──────────────────────────────────────────────────────────────────

    private JTextArea leftArea, rightArea;

    // Криптографія
    private JRadioButton aesRadio, rsaRadio;

    // Місце розміщення прихованих даних
    private JRadioButton commentRadio, metaRadio;

    private JLabel statusLabel;
    private JLabel fileNameLabel;

    //                          ──                          Стан
──────────────────────────────────────────────────────────────────
──
```

```
        private byte[] rawPdf;
        private int maxCapacity = 0;
        /** Поточний активний файл на диску (оновлюється після
кожного збереження) */
        private File currentFile = null;

        //                    ──                    Константи
──────────────────────────────────────────────────────────────

        /** Ключ в Info Dictionary (режим MetaData) */
       private final String KEY_NAME     = "Metadata_Revision";
        /** Маркер прихованого коментаря всередині PDF (режим
Comment) */
        private final String COMMENT_MARKER = "%%%";
        /** Маркер хешу в хвості файлу */
        private final String HASH_MARKER   = "%%%%";

        //                    ──                    Конструктор
──────────────────────────────────────────────────────────────

        public PdfStegoExt() {
            initUI();
        }

        //          ──          Побудова          інтерфейсу
──────────────────────────────────────────────

        private void initUI() {
          setTitle("PdfStegoExt — Security Stego Edition ONMU
2026");
            setSize(1150, 820);
            setDefaultCloseOperation(EXIT_ON_CLOSE);

            //                    ═══                    МЕНЮ
══════════════════════════════════════════════════════════════

            JMenuBar mb = new JMenuBar();

            // --- Файл ---
            JMenu fileMenu = new JMenu("Файл");
```

```
            JMenuItem load   = new JMenuItem("Завантажити PDF");
              JMenuItem save      = new JMenuItem("Зберегти як
(Приховати текст)");
            JMenuItem extract = new JMenuItem("Витягти прихований
текст");
              JMenuItem fullClean = new JMenuItem("Повна очистка
файлу (Анонімізація)");

              load.addActionListener(e -> loadPdf());
              save.addActionListener(e -> handleStegano(true));
            extract.addActionListener(e -> handleStegano(false));
            fullClean.addActionListener(e -> performFullClean());

              fileMenu.add(load);             fileMenu.add(save);
fileMenu.add(extract);
              fileMenu.addSeparator();
              fileMenu.add(fullClean);

              // --- Хешування ---
              JMenu hashMenu = new JMenu("Хешування");
             JMenuItem embedHash  = new JMenuItem("Вбудувати Хеш
(Захист)");
             JMenuItem verifyHash = new JMenuItem("Перевірити Хеш
(Цілісність)");
            embedHash.addActionListener(e -> handleHash(true));
            verifyHash.addActionListener(e -> handleHash(false));
              hashMenu.add(embedHash); hashMenu.add(verifyHash);

              // --- Криптографія ---
              JMenu cryptoMenu = new JMenu("Криптографія");
            JMenuItem genKeys = new JMenuItem("Згенерувати ключі
RSA");
              genKeys.addActionListener(e -> generateRSAKeys());
              cryptoMenu.add(genKeys);
```

```
            mb.add(fileMenu);                    mb.add(hashMenu);
mb.add(cryptoMenu);

            //      ===      ПАНЕЛЬ      АЛГОРИТМУ      ШИФРУВАННЯ
=====================================

            JPanel      cryptoPanel      =      new      JPanel(new
FlowLayout(FlowLayout.LEFT, 6, 2));

cryptoPanel.setBorder(BorderFactory.createTitledBorder("Алгорит
м шифрування"));
            aesRadio = new JRadioButton("AES (Пароль)", true);
            rsaRadio = new JRadioButton("RSA (Ключі)");
            ButtonGroup cryptoGroup = new ButtonGroup();
          cryptoGroup.add(aesRadio); cryptoGroup.add(rsaRadio);
          cryptoPanel.add(aesRadio); cryptoPanel.add(rsaRadio);
            mb.add(cryptoPanel);

            //      ===      ПАНЕЛЬ      МІСЦЯ      ЗБЕРІГАННЯ
=========================================

            JPanel      storePanel      =      new      JPanel(new
FlowLayout(FlowLayout.LEFT, 6, 2));

storePanel.setBorder(BorderFactory.createTitledBorder("Місце
зберігання"));
            commentRadio = new JRadioButton("Comment ", true);
// за замовчуванням
            metaRadio    = new JRadioButton("MetaData");
            ButtonGroup storeGroup = new ButtonGroup();
            storeGroup.add(commentRadio);
storeGroup.add(metaRadio);
            storePanel.add(commentRadio);
storePanel.add(metaRadio);
            mb.add(storePanel);

            setJMenuBar(mb);
```

```
            // ═══ ЦЕНТРАЛЬНА ПАНЕЛЬ ═══════════════════════════════════
            JPanel mainPanel = new JPanel(new GridLayout(1, 2, 10, 0));

            JPanel leftPanel = new JPanel(new BorderLayout());
          leftArea = createArea("Текст для вбудовування", true);
            JButton clearLeft = new JButton("Очистити ліве вікно");
            clearLeft.addActionListener(e -> leftArea.setText(""));
            leftPanel.add(new JScrollPane(leftArea), BorderLayout.CENTER);
            leftPanel.add(clearLeft, BorderLayout.SOUTH);

          JPanel rightPanel = new JPanel(new BorderLayout());
            rightArea = createArea("Результат перевірки / Витягнутий текст", false);
            JButton clearRight = new JButton("Очистити праве вікно");
            clearRight.addActionListener(e -> {
                rightArea.setText("");
                rightArea.setBackground(Color.WHITE);
            });
            rightPanel.add(new JScrollPane(rightArea), BorderLayout.CENTER);
            rightPanel.add(clearRight, BorderLayout.SOUTH);

            mainPanel.add(leftPanel);
            mainPanel.add(rightPanel);

            // ═══ НИЖНЯ ПАНЕЛЬ СТАТУСУ ═══════════════════════════════════
          JPanel footerPanel = new JPanel(new BorderLayout());
```

```
footerPanel.setBorder(BorderFactory.createLoweredBevelBorder())
;

            statusLabel = new JLabel("Ємність: 0 байт");

statusLabel.setBorder(BorderFactory.createEmptyBorder(2, 10, 2,
10));

          fileNameLabel = new JLabel("Файл не завантажений");
            fileNameLabel.setForeground(Color.RED);

fileNameLabel.setBorder(BorderFactory.createEmptyBorder(2,   10,
2, 10));

            footerPanel.add(statusLabel,  BorderLayout.WEST);
            footerPanel.add(fileNameLabel, BorderLayout.EAST);

            add(mainPanel,   BorderLayout.CENTER);
            add(footerPanel, BorderLayout.SOUTH);
        }

        //      ——      Допоміжник:      текстова      область
————————————————————————————————————

        private  JTextArea  createArea(String  title,  boolean
editable) {
            JTextArea a = new JTextArea();
            a.setLineWrap(true); a.setWrapStyleWord(true);
            a.setEditable(editable);

a.setBorder(BorderFactory.createTitledBorder(title));
            a.setFont(new Font("Monospaced", Font.PLAIN, 13));
            return a;
        }
```

```
        //
═══════════════════════════════════════════════════════════════
══════════
        //  ЗАВАНТАЖЕННЯ PDF
        //
═══════════════════════════════════════════════════════════════
══════════
        private void loadPdf() {
            JFileChooser fc = new JFileChooser();
            if            (fc.showOpenDialog(this)            ==
JFileChooser.APPROVE_OPTION) {
                try {
                    File file = fc.getSelectedFile();
                    rawPdf                                      =
java.nio.file.Files.readAllBytes(file.toPath());
                    currentFile = file;
                maxCapacity = (int) (rawPdf.length * 0.15);
                    rightArea.setBackground(Color.WHITE);
                    statusLabel.setText("Доступна ємність: " +
maxCapacity + " байт");
                    fileNameLabel.setText("Поточний файл: " +
file.getName());
                fileNameLabel.setForeground(new  Color(0,  102,
0));
                    JOptionPane.showMessageDialog(this,    "PDF
успішно завантажено.");
                } catch (Exception ex) {
                JOptionPane.showMessageDialog(this,    "Помилка
при читанні файлу!");
                }
            }
        }

        //
═══════════════════════════════════════════════════════════════
══════════
```

```
        //  СТЕГАНОГРАФІЯ — диспетчер
        //
═══════════════════════════════════════════════════════════
══════════
        private void handleStegano(boolean isEmbed) {
            if (rawPdf == null) {
                JOptionPane.showMessageDialog(this,  "Файл  не
завантажений!", "Увага",
                        JOptionPane.WARNING_MESSAGE);
                return;
            }
            try {
                if (isEmbed) embedMessage();
                else         extractMessage();
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this,    "Помилка
стеганографії: " + ex.getMessage());
            }
        }

        //         ──          Вбудування          повідомлення
──────────────────────────────────────────
        private void embedMessage() throws Exception {
            byte[]                    msg                    =
leftArea.getText().getBytes(StandardCharsets.UTF_8);
            if (msg.length > maxCapacity)
             throw new Exception("Текст перевищує ліміт ємності
(15%)!");

            byte[] encrypted = encryptData(msg);
            if (encrypted == null) return;

            String               encodedPayload               =
Base64.getEncoder().encodeToString(encrypted);

            if (metaRadio.isSelected()) {
```

```
                // ── Режим MetaData: пишемо в Info Dictionary
──────────────────────
                embedInMetadata(encodedPayload);
            } else {
                // ── Режим Comment: вставляємо %%% всередині
PDF-потоку ──────────
                embedInComment(encodedPayload);
            }
        }

        /** Запис у поле Metadata_Revision Info Dictionary */
        private void embedInMetadata(String encodedPayload)
throws Exception {
            try (PDDocument doc = PDDocument.load(rawPdf)) {

doc.getDocumentInformation().setCustomMetadataValue(KEY_NAME,
encodedPayload);
                JFileChooser fc = new JFileChooser();
                if (fc.showSaveDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                    File dest = fc.getSelectedFile();
                    doc.save(dest);
                    // Оновлюємо поточний файл у пам'яті
                    rawPdf =
java.nio.file.Files.readAllBytes(dest.toPath());
                    currentFile = dest;
                maxCapacity = (int) (rawPdf.length * 0.15);
                    statusLabel.setText("Доступна ємність: " +
maxCapacity + " байт");
                    fileNameLabel.setText("Поточний файл: " +
dest.getName());
                    JOptionPane.showMessageDialog(this,
                            "Дані вбудовано в MetaData
(Metadata_Revision).\n" +
                                    "Поточний файл: " +
dest.getName());
```

```
                }
            }
        }

        /**
         * Запис маркера %%% + payload в байти PDF.
        * Маркер вставляється перед %%EOF — після основного тіла документа,
         * але всередині файлу (до будь-якого зовнішнього HASH_MARKER).
         * PDF-переглядачі ігнорують дані після %%EOF.
         */
        private void embedInComment(String encodedPayload) throws Exception {
            String raw = new String(rawPdf, StandardCharsets.ISO_8859_1);
            int eofIdx = raw.lastIndexOf("%%EOF");

            byte[] commentBytes = (COMMENT_MARKER + encodedPayload)
                    .getBytes(StandardCharsets.UTF_8);

            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            if (eofIdx != -1) {
                baos.write(rawPdf, 0, eofIdx);
                baos.write('\n');
                baos.write(commentBytes);
                baos.write('\n');
                baos.write(rawPdf, eofIdx, rawPdf.length - eofIdx);
            } else {
                baos.write(rawPdf);
                baos.write('\n');
                baos.write(commentBytes);
            }
```

```
            byte[] result = baos.toByteArray();

            JFileChooser fc = new JFileChooser();
            if           (fc.showSaveDialog(this)           ==
JFileChooser.APPROVE_OPTION) {
                File dest = fc.getSelectedFile();
             java.nio.file.Files.write(dest.toPath(), result);
                // Оновлюємо поточний файл у пам'яті
                rawPdf = result;
                currentFile = dest;
                maxCapacity = (int) (rawPdf.length * 0.15);
                statusLabel.setText("Доступна   ємність:   "   +
maxCapacity + " байт");
                fileNameLabel.setText("Поточний   файл:   "   +
dest.getName());
                JOptionPane.showMessageDialog(this,
                   "Дані вбудовано як Comment  у тіло PDF.\n"
+
                         "Поточний файл: " + dest.getName());
            }
        }

        //          ──          Витягання          повідомлення
──────────────────────────────────────────────
        private void extractMessage() throws Exception {
            if (metaRadio.isSelected()) {
                extractFromMetadata();
            } else {
                extractFromComment();
            }
        }

        /** Витягання з Info Dictionary поточного файлу */
        private void extractFromMetadata() throws Exception {
```

```
            if (rawPdf == null) throw new Exception("Файл не
завантажений!");
            try (PDDocument doc = PDDocument.load(rawPdf)) {
                String encoded = doc.getDocumentInformation()
                        .getCustomMetadataValue(KEY_NAME);
                if (encoded == null)
                    throw new Exception("В MetaData немає
прихованого тексту.");
                byte[] decrypted =
decryptData(Base64.getDecoder().decode(encoded));
                if (decrypted != null) {
                    rightArea.setText(new String(decrypted,
StandardCharsets.UTF_8));
                rightArea.setBackground(new Color(230, 240,
255));
                }
            }
        }

        /** Витягання з Comment-маркера %%% поточного файлу */
        private void extractFromComment() throws Exception {
            if (rawPdf == null) throw new Exception("Файл не
завантажений!");
            String content = new String(rawPdf,
StandardCharsets.ISO_8859_1);

            int idx = findCommentMarker(content);
            if (idx == -1)
                throw new Exception("Маркер не знайдено. Файл
не містить стеганографії.");

            int start = idx + COMMENT_MARKER.length();
            int end = content.indexOf('\n', start);
            if (end == -1) end = content.length();
            String encoded = content.substring(start,
end).trim();
```

```
            if (encoded.contains("%%EOF"))
                encoded = encoded.substring(0,
encoded.indexOf("%%EOF")).trim();

            byte[] decrypted =
decryptData(Base64.getDecoder().decode(encoded));
            if (decrypted != null) {
                rightArea.setText(new String(decrypted,
StandardCharsets.UTF_8));
                 rightArea.setBackground(new Color(230, 240, 255));
            }
        }

        /**
         * Знаходить позицію маркера %%% що НЕ є частиною %%%%
(маркер хешу).
         * Перевіряє що:
        *  - символ ПІСЛЯ %%% не є '%'  (інакше це початок %%%%)
        *  - символ ПЕРЕД %%% не є '%'  (інакше це позиція 1..3
всередині %%%%)
         */
        private int findCommentMarker(String content) {
            int searchFrom = content.length();
            while (true) {
                int idx = content.lastIndexOf(COMMENT_MARKER,
searchFrom - 1);
                if (idx == -1) return -1;

                // Перевірка справа: наступний символ не '%' →
не початок %%%%
                boolean nextIsPercent = (idx +
COMMENT_MARKER.length() < content.length())
                        && content.charAt(idx +
COMMENT_MARKER.length()) == '%';
```

```
                // Перевірка зліва: попередній символ не '%' →
не всередині %%%%
                boolean prevIsPercent = (idx > 0)
                        && content.charAt(idx - 1) == '%';

                if (nextIsPercent || prevIsPercent) {
                    searchFrom = idx;
                    continue;
                }
                return idx;
            }
        }

        //
═══════════════════════════════════════════════════════════════
═══════
        //  ХЕШУВАННЯ
        //
═══════════════════════════════════════════════════════════════
═══════
        private void handleHash(boolean isEmbed) {
            if (rawPdf == null) {
                JOptionPane.showMessageDialog(this,  "Файл  не
завантажений!", "Увага",
                        JOptionPane.WARNING_MESSAGE);
                return;
            }
            try {
                if (isEmbed) embedFileHash();
                else         verifyFileHash();
            } catch (Exception ex) {
             rightArea.setBackground(new Color(241, 28, 28));
                JOptionPane.showMessageDialog(this,    "Помилка
хешування: " + ex.getMessage());
            }
        }
```

```
        private void embedFileHash() throws Exception {
            if (currentFile == null)
             throw new Exception("Спочатку збережіть файл через
«Зберегти як»!");

            MessageDigest                digest                =
MessageDigest.getInstance("SHA-256");
            String                    hashStr                    =
Base64.getEncoder().encodeToString(digest.digest(rawPdf));
            byte[]                encryptedHash                =
encryptData(hashStr.getBytes(StandardCharsets.UTF_8));
            if (encryptedHash == null) return;

          // Дописуємо хеш після поточного вмісту і зберігаємо
в той самий файл
            ByteArrayOutputStream       baos       =       new
ByteArrayOutputStream();
            baos.write(rawPdf);

baos.write(HASH_MARKER.getBytes(StandardCharsets.UTF_8));

baos.write(Base64.getEncoder().encodeToString(encryptedHash)
                    .getBytes(StandardCharsets.UTF_8));
            byte[] withHash = baos.toByteArray();

            java.nio.file.Files.write(currentFile.toPath(),
withHash);
            rawPdf = withHash;
            maxCapacity = (int) (rawPdf.length * 0.15);
            statusLabel.setText("Доступна    ємність:    "    +
maxCapacity + " байт");

            JOptionPane.showMessageDialog(this,
                    "Хеш вбудовано в кінець файлу.\nФайл: " +
currentFile.getName());
```

```
        }

        private void verifyFileHash() throws Exception {
            String content = new String(rawPdf,
StandardCharsets.ISO_8859_1);
            int markerIdx = content.lastIndexOf(HASH_MARKER);
            if (markerIdx == -1) {
             rightArea.setBackground(new Color(255, 200, 200));
                throw new Exception("Маркер хешування не
знайдено!");
            }

            byte[] dataBefore = Arrays.copyOfRange(rawPdf, 0,
markerIdx);
            String calculatedHash =
Base64.getEncoder().encodeToString(
                    MessageDigest.getInstance("SHA-
256").digest(dataBefore));

            byte[] encryptedHashFromFile =
Base64.getDecoder().decode(
                    content.substring(markerIdx +
HASH_MARKER.length()).trim());
            byte[] decryptedHash =
decryptData(encryptedHashFromFile);
            if (decryptedHash == null) return;

            String originalHash = new String(decryptedHash,
StandardCharsets.UTF_8);
            if (calculatedHash.equals(originalHash)) {
                rightArea.setBackground(new Color(8, 214, 8));
                rightArea.setText("ПЕРЕВІРКА УСПІШНА:\nФайл
цілісний.\n\nHash: " + originalHash);
            } else {
                rightArea.setBackground(new Color(124, 5, 5));
             rightArea.setText("УВАГА: ЦІЛІСНІСТЬ ПОРУШЕНО!");
```

```
            }
        }

        //
==========================================================
==========
        //  ПОВНА ОЧИСТКА (АНОНІМІЗАЦІЯ)
        //   Видаляє: MetaData поле, Comment-маркер %%%, хеш-
маркер %%%%
        //
==========================================================
==========
        private void performFullClean() {
            if (rawPdf == null) return;
            try {
                byte[] cleanedData;

                // Крок 1: очистка Info Dictionary через PDFBox
                try (PDDocument doc = PDDocument.load(rawPdf))
{

doc.getDocumentInformation().setCustomMetadataValue(KEY_NAME,
null);
                    ByteArrayOutputStream baos = new
ByteArrayOutputStream();
                    doc.save(baos);
                    cleanedData = baos.toByteArray();
                }

                // Крок 2: видалення хеш-маркера %%%% з хвоста
                String content = new String(cleanedData,
StandardCharsets.ISO_8859_1);
                int hashIdx = content.lastIndexOf(HASH_MARKER);
                if (hashIdx != -1)
                cleanedData = Arrays.copyOfRange(cleanedData,
0, hashIdx);
```

```
            // Крок 3: видалення Comment-маркера %%% з тіла
            cleanedData = removeCommentMarker(cleanedData);

                JFileChooser fc = new JFileChooser();
                if          (fc.showSaveDialog(this)          ==
JFileChooser.APPROVE_OPTION) {
                    File dest = fc.getSelectedFile();
                    java.nio.file.Files.write(dest.toPath(),
cleanedData);
                    rawPdf = cleanedData;
                    currentFile = dest;
                maxCapacity = (int) (rawPdf.length * 0.15);
                    statusLabel.setText("Доступна ємність: " +
maxCapacity + " байт");
                    fileNameLabel.setText("Поточний файл: " +
dest.getName());
                    JOptionPane.showMessageDialog(this,
                            "Анонімізація завершена.\n" +
                            "Видалено:                MetaData,
Comment , Hash .\n" +
                                    "Поточний    файл:    "    +
dest.getName());
                }
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this,   "Помилка:
" + ex.getMessage());
            }
        }

        /**
        * Видаляє рядок із маркером %%% (але не %%%%!) з байтів
PDF.
         * Повертає очищений масив.
         */
        private byte[] removeCommentMarker(byte[] data) {
```

```
            String content = new String(data,
StandardCharsets.ISO_8859_1);
            int idx = findCommentMarker(content);
          if (idx == -1) return data; // маркера немає — нічого
не чистимо

          // Знаходимо початок рядка (попередній \n) і кінець
рядка (наступний \n)
            int lineStart = content.lastIndexOf('\n', idx);
          if (lineStart == -1) lineStart = 0; else lineStart++;
// символ після \n

            int lineEnd = content.indexOf('\n', idx);
          if (lineEnd == -1) lineEnd = content.length(); else
lineEnd++; // включаємо \n

            // Зшиваємо масив без рядка з маркером
            ByteArrayOutputStream baos = new
ByteArrayOutputStream();
            baos.write(data, 0, lineStart);
            if (lineEnd < data.length)
             baos.write(data, lineEnd, data.length - lineEnd);
            return baos.toByteArray();
        }

        //
═══════════════════════════════════════════════════════════
═══════
        //  КРИПТОГРАФІЯ
        //
═══════════════════════════════════════════════════════════
═══════

        /** Розмір блоку для RSA 2048 + PKCS1Padding: 245 байт
на вхід, 256 байт на виході */
        private static final int RSA_BLOCK_PLAIN  = 245;
```

```
        private static final int RSA_BLOCK_CIPHER = 256;

    private byte[] encryptData(byte[] data) throws Exception
{
        if (aesRadio.isSelected()) {
            JPasswordField pf = new JPasswordField();
            if   (JOptionPane.showConfirmDialog(this,   pf,
"Пароль AES",
                    JOptionPane.OK_CANCEL_OPTION)        !=
JOptionPane.OK_OPTION) return null;
            byte[] key = Arrays.copyOf(
                    MessageDigest.getInstance("SHA-256")
                            .digest(new
String(pf.getPassword()).getBytes()), 16);
            Cipher c = Cipher.getInstance("AES");
            c.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(key,
"AES"));
            return c.doFinal(data);
        } else {
            File f = selectFile("RSA Public Key");
            if (f == null) return null;
            PublicKey                pk                =
KeyFactory.getInstance("RSA").generatePublic(
                    new
X509EncodedKeySpec(java.nio.file.Files.readAllBytes(f.toPath())
));
            Cipher                   c                 =
Cipher.getInstance("RSA/ECB/PKCS1Padding");
            c.init(Cipher.ENCRYPT_MODE, pk);
            return rsaProcessBlocks(c, data, RSA_BLOCK_PLAIN);
        }
    }

    private byte[] decryptData(byte[] data) throws Exception
{
        try {
```

```
                if (aesRadio.isSelected()) {
                    JPasswordField pf = new JPasswordField();
                 if   (JOptionPane.showConfirmDialog(this,   pf,
"Пароль AES",
                            JOptionPane.OK_CANCEL_OPTION)    !=
JOptionPane.OK_OPTION) return null;
                    byte[] key = Arrays.copyOf(
                      MessageDigest.getInstance("SHA-256")
                                   .digest(new
String(pf.getPassword()).getBytes()), 16);
                    Cipher c = Cipher.getInstance("AES");
                    c.init(Cipher.DECRYPT_MODE,             new
SecretKeySpec(key, "AES"));
                    return c.doFinal(data);
                } else {
                    File f = selectFile("RSA Private Key");
                    if (f == null) return null;
                    PrivateKey               pk               =
KeyFactory.getInstance("RSA").generatePrivate(
                            new
PKCS8EncodedKeySpec(java.nio.file.Files.readAllBytes(f.toPath()
)));
                    Cipher                 c                 =
Cipher.getInstance("RSA/ECB/PKCS1Padding");
                    c.init(Cipher.DECRYPT_MODE, pk);
                    return      rsaProcessBlocks(c,      data,
RSA_BLOCK_CIPHER);
                }
            } catch (Exception e) {
                throw new Exception("Помилка ключа/пароля!");
            }
        }

        /**
         * Обробляє дані блоками заданого розміру через вже
ініціалізований Cipher.
```

```
         * При шифруванні blockSize = 245 (відкритий текст).
         * При дешифруванні blockSize = 256 (зашифрований блок
RSA 2048).
         */
        private byte[] rsaProcessBlocks(Cipher cipher, byte[]
data, int blockSize) throws Exception {
            ByteArrayOutputStream baos = new
ByteArrayOutputStream();
            int offset = 0;
            while (offset < data.length) {
                int len = Math.min(blockSize, data.length -
offset);
             byte[] block = cipher.doFinal(data, offset, len);
                baos.write(block);
                offset += len;
            }
            return baos.toByteArray();
        }

        // ── Генерація RSA ключів
──────────────────────────────────────────

        private void generateRSAKeys() {
            JFileChooser fc = new JFileChooser();

fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
            if (fc.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION) {
                try {
                    KeyPairGenerator gen =
KeyPairGenerator.getInstance("RSA");
                    gen.initialize(2048);
                    KeyPair pair = gen.generateKeyPair();
                    File dir = fc.getSelectedFile();
                    try (FileOutputStream pub = new
FileOutputStream(new File(dir, "public.key"))) {
                        pub.write(pair.getPublic().getEncoded());
```

```
                    }
                    try (FileOutputStream priv = new
FileOutputStream(new File(dir, "private.key"))) {
                        priv.write(pair.getPrivate().getEncoded());
                    }
                    JOptionPane.showMessageDialog(this, "Ключі
RSA 2048 успішно створено.");
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }
        }

        // ── Вибір файлу
──────────────────────────────────────────────────────
        private File selectFile(String title) {
            JFileChooser fc = new JFileChooser();
            fc.setDialogTitle(title);
            return fc.showOpenDialog(this) ==
JFileChooser.APPROVE_OPTION
                    ? fc.getSelectedFile() : null;
        }

        // ── Точка входу
──────────────────────────────────────────────────────
        public static void main(String[] args) {
            SwingUtilities.invokeLater(() -> new
PdfStegoExt().setVisible(true));
        }
}
```

Метадані

ДОКУМЕНТ

Заголовок

2026 КвРбак - Запорожан

Автор	Науковий керівник / Експерт	ІД документу
Запорожан Артем	**доцент Юрій Володимирович Даус**	**334285232**

ОРГАНІЗАЦІЯ

Назва організації	підрозділ
Odesa National Maritime University	**Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта**

ЗВІТ

Дата звіту	Дата редагування
6/11/2026	**---**

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

8.43% (8.43%) КП 1	5.35% (5.35%) КП 2	3.2% (3.2%) КЦ
25	**14888**	**128825**
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	6
Інтервали	A→	0
Мікропробіли	·	27
Білі знаки	ß	61
Парафрази (SmartMarks)	a	46

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

1	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	183 (1.23 %)
2	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	92 (0.62 %)
3	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	72 (0.48 %)
4	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	57 (0.38 %)
5	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	49 (0.33 %)
6	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	44 (0.3 %)
7	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	44 (0.3 %)
8	ВРбак СА - Гриценко бд [redacted] 4/20/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	41 (0.28 %)
9	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	39 (0.26 %)
10	Запорожан_PDF_final_check [redacted] 4/17/2026 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	39 (0.26 %)

Домашня база даних (7.37 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Запорожан_PDF_final_check [redacted] 4/17/2026 **Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)**	**1003 (32) (6.74 %)**
2	ВРбак СА - Гриценко бд [redacted] 4/20/2026	**63 (4) (0.42 %)**

[redacted] 4/20/2025
Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)

3	ЄвтушокКН21_розділ_2_2025_2_21_12_2025 _1 [redacted] 12/21/2025 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	25 (2) (0.17 %)
4	ВРмаг 2025 - Ігнатьєва НД [redacted] 12/21/2025 Odesa National Maritime University (Технічна кібернетика й інформаційні технології ім. професора Р.В. Меркта)	6 (1) (0.04 %)

Програма обміну базами даних (0.2 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
5	Хельчик ІБ-26 курсова (1) [redacted] 6/4/2025 Lesya Ukrainka Volyn National University (Кафедра загальної математики та методики навчання інформатики)	11 (1) (0.07 %)
6	БатраченкоТС_Розгон ОГ_стаття [redacted] 8/1/2024 Publishing House "Helvetica" (Видавничий дім "Гельветика")	10 (1) (0.07 %)
7	Диплом-Козирь-В.В. [redacted] 12/8/2024 Zaporizhzhia National University (Кафедра електроніки, інформаційних систем та програмного забезпечення (спеціальність 121 Інженерія програмного забезпечення))	9 (1) (0.06 %)

Інтернет (0.86 %)

#	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
8	https://visuresolutions.com/uk/%D0%B1%D0%BB%D0%BE%D0%B3/%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9-%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82-%D1%96-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B5-%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B4%D0%BB%D1%8F-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F-%D1%80%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0%D0%BC%D0%B8/	28 (2) (0.19 %)
9	https://kadrovik.isu.net.ua/taxonomy/term/2015	27 (2) (0.18 %)
10	https://java-swing-tips.blogspot.com/	14 (2) (0.09 %)
11	https://ur.knute.edu.ua/bitstreams/b8101115-bb8a-4b8d-a7c2-1895a5b2bb44/download	14 (2) (0.09 %)
12	https://iq.vntu.edu.ua/repository/getfile.php/11537.pdf	12 (1) (0.08 %)
13	https://ur.knute.edu.ua/bitstreams/87306f7d-5199-48a9-b4b9-44a707c9bbad/download	12 (1) (0.08 %)
14	https://esop.expertus.com.ua/rssnews	11 (1) (0.07 %)
15	https://essuir.sumdu.edu.ua/bitstream-download/123456789/95678/1/Peichev_bac_rob.pdf	10 (1) (0.07 %)

Список прийнятих фрагментів

#	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

2

Одеський національний морський університет

Навчально-науковий інститут інформаційних технологій
та інноваційного підприємництва
(повна назва інституту)
Кафедра технічної кібернетики й інформаційних технологій
ім. професора Р.В. Меркта
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
бакалавр
(освітньо-кваліфікаційний рівень)
на тему
Використання PDF файлів в якості стеганографічних
контейнерів

Виконав:
здобувач вищої освіти 4 курсу, групи 1
Спеціальність:
122 «Комп'ютерні науки»
(шифр і назва спеціальності)
Артем ЗАПОРОЖАН
(підпис) (Ім'я, ПРІЗВИЩЕ)
Керівник к.геогр.н., доцент
(вчене звання, посада)
Юрій ДАУС
(підпис) (Ім'я, ПРІЗВИЩЕ)

Одеса - 2026

АНОТАЦІЯ

У роботі досліджено доцільність використання PDF-документів як носіїв прихованої інформації в стеганографічних системах. В умовах стрімкого розвитку інформаційних технологій стеганографія посідає важливе місце серед засобів забезпечення конфіденційності та захисту даних. Традиційно для приховування інформації переважно застосовуються графічні файли, які характеризуються значною інформаційною ємністю та простотою реалізації відповідних методів. Водночас широке використання зображень як стеганографічних контейнерів робить їх об'єктом підвищеної уваги з боку засобів аналізу та виявлення прихованих повідомлень.
У цьому контексті використання PDF-файлів як альтернативного середовища для вбудовування прихованих даних є перспективним напрямом досліджень. Такий підхід дозволяє зменшити ймовірність виявлення факту застосування стеганографічних методів, а також розширює спектр доступних інструментів для організації прихованого обміну інформацією.
Розроблення програмного комплексу, що реалізує механізми стеганографічного захисту на основі PDF-документів, забезпечує можливість інтеграції різноманітних алгоритмів приховування та криптографічного захисту даних. Це сприяє підвищенню гнучкості та ефективності застосування таких систем у сучасному інформаційному середовищ.
Під час створення програмного забезпечення використовувалися сучасні принципи об'єктно-орієнтованого програмування, а також алгоритми симетричного й асиметричного шифрування. Результатом проведеної роботи став функціональний програмний продукт, який може бути застосований для забезпечення безпечного обміну інформацією у військових структурах, корпоративному секторі та під час приватних комунікацій.

ABSTRACT

The paper investigates the feasibility of using PDF documents as carriers of hidden information in steganographic systems. In the context of the rapid development of information technologies, steganography occupies an important place among the means of ensuring confidentiality and data protection. Traditionally, graphic files are mainly used for hiding information, which are characterized by significant information capacity and ease of implementation of the corresponding methods. At the same time, the widespread use of images as steganographic containers makes them the object of increased attention from the analysis and detection of hidden messages.

In this context, the use of PDF files as an alternative environment for embedding hidden data is a promising area of research. This approach allows you to reduce the likelihood of detecting the use of steganographic methods, and also expands the range of available tools for organizing hidden information exchange.

The development of a software package that implements steganographic protection mechanisms based on PDF documents provides the ability to integrate various algorithms for hiding and cryptographic data protection. This contributes to increasing the flexibility and efficiency of the use of such systems in the modern information environment.

When creating the software, modern principles of object-oriented programming, as well as symmetric and asymmetric encryption algorithms, were used. The result of the work was a functional software product that can be used to ensure secure information exchange in military structures, the corporate sector and during private communications.

ЗМІСТ

ВСТУП

У сучасному інформаційному просторі особливого значення набувають технології забезпечення конфіденційності даних, серед яких важливе місце займає стеганографія. На відміну від криптографічних методів захисту, що спрямовані на приховування змісту повідомлення, стеганографія орієнтована на маскування самого факту існування та передачі інформації [1].

Основою стеганографічних технологій є вбудовування прихованих повідомлень у різноманітні інформаційні об’єкти таким чином, щоб їх наявність залишалася непомітною для стороннього спостерігача. Об’єкти, які використовуються для розміщення прихованих даних, прийнято називати стеганографічними контейнерами. Значний розвиток цього напряму відбувся з поширенням цифрових технологій та появою електронних документів, графічних зображень, аудіо- та відеофайлів.

Найбільш поширеними контейнерами для приховування інформації традиційно є цифрові зображення. Це пояснюється особливостями людського зорового сприйняття, яке не здатне фіксувати незначні зміни окремих колірних компонентів. Завдяки цьому широкого застосування набув метод модифікації молодших бітів кольорових каналів, що дозволяє вбудовувати приховані повідомлення без помітного погіршення якості зображення.

Разом із тим перспективним напрямом розвитку стеганографічних систем є використання PDF-документів як контейнерів для прихованих даних. Згідно з результатами досліджень, наведених у роботі [2], за потенціалом застосування у стеганографії PDF-файли поступаються лише мережевому трафіку. Це обумовлено їх значною інформаційною ємністю, структурованою організацією внутрішнього вмісту, високою стійкістю до змін під час збереження та редагування, а також широким поширенням у сучасних інформаційних системах. Важливим завданням є дослідження та аналіз існуючих способів приховування інформації в таких контейнерах, а також розроблення нових підходів до вбудовування даних. Проведення подібних досліджень сприяє підвищенню ефективності механізмів захисту інформації, удосконаленню методів прихованого обміну даними та створенню засобів виявлення можливих стеганографічних загроз. У зв’язку з цим вивчення можливостей використання PDF-файлів як стеганографічних контейнерів є актуальним науково-практичним завданням, вирішення якого сприятиме подальшому розвитку технологій захисту конфіденційної інформації.

1. ВИДИ СТЕГАНОГРАФІЧНИХ КОНТЕЙНЕРІВ 1. Становлення та розвиток стеганографії Стеганографія

є одним із напрямів захисту інформації, метою якого є приховування факту існування повідомлення. Термін походить від грецьких слів «στεγανός» (прихований) та «γράφω» (пишу) і означає сукупність методів та засобів прихованої передачі або зберігання даних [1]. На відміну від криптографії, яка забезпечує захист змісту повідомлення шляхом його шифрування, стеганографія спрямована на те, щоб зробити сам процес обміну інформацією непомітним для сторонніх осіб.

Принцип функціонування стеганографічних систем полягає у вбудовуванні секретних даних у різноманітні носії інформації таким чином, щоб приховане повідомлення не викликало підозри щодо своєї наявності. У сучасних умовах як середовище для розміщення прихованої інформації можуть використовуватися практично будь-які цифрові об’єкти, зокрема текстові документи, графічні файли, аудіозаписи, відеоматеріали та мережевий трафік [3].

Сфера застосування стеганографії охоплює широкий спектр напрямів діяльності, серед яких інформаційна безпека, захист конфіденційних даних, забезпечення секретності комунікацій, цифрова криміналістика, а також розроблення методів виявлення та протидії стеганографічним загрозам [4]. Поєднання стеганографічних і криптографічних механізмів дозволяє створювати багаторівневі системи захисту інформації, у яких

одночасно забезпечується як приховування змісту повідомлення, так і маскування самого факту його передавання. Такий підхід є особливо актуальним в умовах підвищених вимог до безпеки інформаційного обміну.
Історія розвитку стеганографії налічує понад дві тисячі років. Одні з перших описів методів прихованого передавання інформації містяться у праці Геродота «Історія», датованій V століттям до нашої ери. У ній наведено приклади використання прихованих повідомлень шляхом нанесення тексту на поголену голову раба з подальшим відправленням його після відростання волосся, а також записування інформації на дерев'яну основу воскових табличок із наступним нанесенням поверх неї звичайного тексту.
Подальший розвиток методів приховування інформації пов'язаний із використанням так званих симпатичних чорнил. Такі речовини дозволяли створювати невидимі записи, які ставали доступними для читання лише після впливу певних фізичних або хімічних факторів, зокрема нагрівання, освітлення чи оброблення спеціальними реагентами. Відомим прикладом є використання органічних рідин, зокрема молока, сліди якого проявлялися під дією високої температури.
Окремий напрям становили повідомлення з обмеженим терміном існування, що створювалися за допомогою спеціальних барвників, здатних поступово руйнуватися під впливом навколишнього середовища. Проте розвиток сучасних методів криміналістичного аналізу значно підвищив можливості виявлення та відновлення подібної інформації.
У різні історичні періоди застосовувалися також інші способи приховування відомостей, серед яких запис повідомлень усередині харчових продуктів, використання спеціального розташування гральних карт, геометричні методи розміщення ключових слів у тексті, застосування трафаретів, проколів на предметах, умовних символів та вузликового письма. Незважаючи на відмінності в реалізації, усі зазначені підходи мали спільну мету - забезпечення прихованого передавання інформації без привернення уваги сторонніх осіб.
На сучасному етапі розвитку інформаційних технологій традиційні фізичні носії практично повністю поступилися місцем цифровим контейнерам. Використання електронних документів, мультимедійних файлів і мережевих ресурсів значно розширило можливості застосування стеганографічних методів, забезпечивши високу місткість контейнерів, швидкість оброблення даних та можливість інтеграції з сучасними засобами криптографічного захисту інформації.

- Різновидності стеганографічних контейнерів

Стеганографічна система являє собою комплекс взаємопов'язаних елементів і механізмів, призначених для прихованого передавання та зберігання інформації. До складу такої системи входять контейнер, секретне повідомлення, стеганографічний ключ, заповнений контейнер, а також алгоритми вбудовування та вилучення даних. Як контейнер можуть використовуватися різноманітні цифрові об'єкти, зокрема текстові документи, графічні зображення, аудіофайли та відеоматеріали. У ролі прихованого повідомлення може виступати текстова, графічна або інша цифрова інформація. Стеганографічний ключ являє собою набір параметрів, що визначає спосіб розміщення прихованих даних у контейнері та порядок їх подальшого відновлення [8].
Структура стеганосистеми може включати в себе різноманітні компоненти і процеси необхідні для вбудовування, шифрування, вилучення та розшифровування сповіщень. На рис. 1.1 представлена типова схема стеганосистеми.

Рисунок 1.1 - Типова структура та склад стегосистеми

Архітектура стеганографічної системи може відрізнятися залежно від обраних методів приховування інформації, проте її основною функцією залишається забезпечення надійного вбудовування, зберігання та вилучення прихованих повідомлень. Стеганосистема може бути реалізована як програмними, так і апаратними засобами або їх комбінацією. Одним із ключових завдань такої системи є адаптація алгоритмів приховування до особливостей середовища, у якому розміщуються секретні дані, із забезпеченням можливості їх подальшого коректного відновлення [9, 10].
Типова стеганографічна система включає декілька функціональних модулів:
- контейнер, який використовується як носій прихованої інформації;
- прекодер, що виконує попередню підготовку повідомлення та його перетворення до форми, найбільш придатної для вбудовування;
- модуль адаптації, призначений для аналізу характеристик контейнера та вибору оптимального способу розміщення інформації;
- модуль вилучення даних, який забезпечує отримання прихованого повідомлення зі стеганоконтейнера;
- декодер, що здійснює зворотне перетворення вилучених даних та відновлює початковий вигляд повідомлення.
Сучасні стеганографічні технології використовують широкий спектр цифрових носіїв інформації. Вибір контейнера визначається його структурними особливостями, місткістю та стійкістю до зовнішніх впливів [11].
До найбільш поширених типів стеганографічних контейнерів належать:ганографічна система включає декілька функціональних модулів:
1. Різноманітні файли зображень в які може бути вписана прихована інформація. Це один із самих поширених видів стеганографічних контейнерів на сьогодні. Приховане сповіщення може бути вбудовано в пікселі зображення змінюючи значення самих пікселів, або в метадані цього файлу. Часто використовують зображення представлені файлами JPG, JPEG, PNG, BMP. 2. Відео файли з розширенням AVI, MP4, MKV і т.д. Тут можна змінювати кольори зображення, або вписувати в аудіо-доріжку. 3. Аудіо файли з розширенням MP3,WAV, FLAC та інші. Тут для приховування сповіщення вносять незначні зміни в аудіосигнал або доповнюють сигнал використанням високочастотних компонентів, які не сприймається людським вухом. 4. Текстові файли. Стеганографія може використовувати текстові файли, вставляючи секретну інформацію в текстовий контент, інтервали між словами або різні атрибути тексту, такі як розмір шрифту, колір тексту тощо. 5. Мережевий трафік (Cover Network Traffic): секретні дані можуть бути вбудовані в мережевий трафік, використовуючи певні властивості пакетів. 6. Інші формати: Стеганографія може застосовуватися до різних інших форматів даних, включаючи PDF-файли, ZIP-архіви, електронні документи, а також в інших креативних способах, в залежності від конкретного завдання та потреби в приховуванні інформації. Після вибору контейнера необхідно визначити спосіб вбудовування секретного повідомлення. Реалізація цього процесу безпосередньо залежить від типу носія інформації та його технічних характеристик. Для виконання операцій приховування можуть використовуватися як програмні, так і апаратні засоби. Апаратна реалізація частіше застосовується під час роботи з мережевим трафіком та спеціалізованими системами передавання даних, тоді як для більшості цифрових контейнерів використовуються програмні модулі.
Важливим елементом будь-якої стеганосистеми є ключ вбудовування, який визначає правила розміщення прихованих даних у контейнері та механізм їх подальшого вилучення. Збереження конфіденційності цього ключа є необхідною умовою забезпечення безпеки всієї системи [12].
Методи вбудовування інформації, що використовуються у стеганографії, повинні забезпечувати непомітність внесених змін, достатню місткість контейнера, стійкість до випадкових або навмисних модифікацій даних, а також можливість коректного відновлення прихованого повідомлення після його вилучення.

У таблиці 1 наведено порівняльну характеристику основних методів приховування інформації залежно від типу стеганографічного контейнера. Аналіз наведених даних свідчить про те, що ефективність застосування певного способу вбудовування безпосередньо залежить від структури та особливостей конкретного формату даних. Деякі методи можуть використовуватися лише для окремих типів контейнерів, тоді як інші є універсальними та придатними для широкого спектра цифрових об'єктів.

Таблиця 1 Методи вбудовування сповіщень в контейнери

Контейнер Тип вбудовування	Зображення	Відео	Аудіо	PDF файли	Текстові файли	Архівні файли
Заміна молодших бітів LSB						
Заміна частоти відтворення						
Вписування в службові зони						
Додавання після кінця файлу						
Вписування інформації в метадані						

Після отримання стеганографічного контейнера одержувач повинен виконати процедуру вилучення та відновлення прихованого повідомлення. Цей процес складається з кількох послідовних етапів:

1. Визначення способу вбудовування інформації. На початковому етапі необхідно встановити метод, за допомогою якого було реалізовано приховування даних. Залежно від типу контейнера це може бути аналіз модифікованих молодших бітів у графічних файлах, дослідження частотних характеристик аудіосигналу або перевірка службових структур документа.
2. Вилучення прихованих даних. Після ідентифікації способу вбудовування здійснюється безпосереднє отримання прихованої інформації зі стеганографічного контейнера. Якість виконання цього етапу визначає повноту та достовірність відновленого повідомлення.В) Перетворення отриманих даних до вигляду що дає можливість прочитати цю інформацію. Часто дані перед поміщенням в контейнер шифрують і тому потрібно дешифрування та декодування.
3. Декодування та дешифрування повідомлення. Вилучені дані зазвичай перебувають у вигляді, непридатному для безпосереднього сприйняття користувачем. Тому виконується процедура декодування, а в разі використання криптографічного захисту - додаткове дешифрування інформації з подальшим відновленням її первинного представлення.
4. Контроль цілісності інформації. Для підтвердження коректності отриманих даних проводиться перевірка їх цілісності. Така процедура дозволяє виявити можливі пошкодження, втрату фрагментів повідомлення або несанкціоновані зміни. Одним із поширених способів контролю є використання хеш-функцій, значення яких може бути попередньо вбудоване до прихованого повідомлення.
5. Видалення залишкових слідів прихованої інформації. За наявності відповідної технічної можливості доцільним є очищення контейнера від вбудованих даних після завершення обробки повідомлення. Це дозволяє зменшити ризик проведення подальшого стеганоаналізу у випадку компрометації контейнера. Однак для окремих методів приховування, зокрема тих, що базуються на модифікації молодших бітів цифрових зображень, повне відновлення початкового стану контейнера може бути складним або неможливим.

Таким чином, ефективність функціонування стеганографічної системи визначається не лише надійністю алгоритму вбудовування інформації, а й коректністю виконання процедур вилучення, відновлення та перевірки достовірності прихованого повідомлення.

1. ОСНОВНІ ТЕХНОЛОГІЇ ВИКОРИСТАННЯ СТЕГАНОГРАФІЧНИХ КОНТЕЙНЕРІВ

2.1 Використання відео та аудіо файлів в якості стеганографічних контейнерів.

Одним із найскладніших середовищ для реалізації стеганографічних методів є відеофайли. Це обумовлено складною структурою відеоконтенту, який складається з великої кількості послідовних кадрів, звукових доріжок та службових даних. Така багатокомпонентна організація значно ускладнює процес приховування інформації та забезпечення її стійкості до оброблення і перекодування.

У сучасних стеганографічних системах приховані дані зазвичай вбудовуються або в відеоряд, або в аудіосупровід. Комбіноване використання обох складових одного відеофайлу як єдиного контейнера наразі застосовується досить рідко через складність реалізації та підвищені вимоги до синхронізації процесів приховування і вилучення інформації [14].

Найбільш поширеними методами приховування даних у відеоконтенті є:

- модифікація коефіцієнтів дискретного косинусного перетворення;
- вбудовування інформації на рівні бітових площин;
- використання енергетичної різниці між спектральними коефіцієнтами.

Метод модифікації коефіцієнтів дискретного косинусного перетворення ґрунтується на внесенні змін до параметрів, що використовуються під час стиснення відеоданих. Його перевагою є відносно висока стійкість до виявлення, проте одним із суттєвих недоліків виступає обмежена місткість контейнера. Через необхідність збереження візуальної якості відео для приховування інформації може бути використана лише незначна частина доступних коефіцієнтів, що зазвичай не перевищує 10-12 % від їх загальної кількості.

Метод вбудовування на рівні бітових площин характеризується високою швидкістю реалізації та значною пропускною здатністю. Його суть полягає у зміні окремих бітів цифрового представлення відеоданих. Водночас така інформація є недостатньо захищеною від втрати, оскільки повторне кодування або оброблення відеофайлу може призвести до часткового чи повного знищення прихованого повідомлення. Крім того, надмірне використання цього методу здатне погіршувати якість відеозображення, що підвищує ймовірність виявлення факту приховування даних [15, 16].

Метод, заснований на використанні енергетичних відмінностей між спектральними коефіцієнтами, дозволяє забезпечити більш високу стійкість прихованої інформації до окремих видів оброблення. Проте його застосування супроводжується зменшенням ефективної пропускної здатності відеопотоку та ускладненням процесу вбудовування повідомлень.

Паралельно з розвитком стеганографічних методів для відеоконтенту активно вдосконалювалися технології приховування інформації в аудіофайлах. Особливістю звукового середовища є обмежені можливості людського слухового апарату щодо сприйняття частотного спектра. Людина здатна сприймати лише певний діапазон звукових частот, тоді як сигнали за його межами практично не реєструються слухом. Саме ця особливість широко використовується при розробленні аудіостеганографічних алгоритмів.

Одним із найбільш поширених методів приховування інформації в аудіофайлах є технологія модифікації молодших значущих бітів (Least Significant Bit, LSB). Принцип роботи методу полягає у внесенні змін до одного або кількох молодших бітів цифрового представлення

аудіосигналу. Оскільки такі зміни практично не впливають на загальні характеристики звуку, вони залишаються непомітними для слухового сприйняття людини.
Ефективність методу пояснюється тим, що зміна останнього біта двійкового числа призводить лише до незначної зміни його числового значення. Завдяки цьому в аудіосигнал можна вбудовувати приховані повідомлення без помітного погіршення якості відтворення. Для наочності на рис. 2.1 наведено приклад двох чисел, які відрізняються лише значенням молодшого біта.

Рисунок 2.1 - Змінення молодшого біта

Якщо змінити лише молодший біт двійкового представлення значення аудіосигналу, різниця між початковим та модифікованим значенням буде мінімальною. Наприклад, значення 170 після зміни останнього біта перетворюється на 171. Така незначна модифікація практично не впливає на сприйняття звуку людиною, оскільки зміни знаходяться нижче порога слухової чутливості. Саме тому метод LSB вважається одним із найефективніших способів приховування інформації в аудіофайлах. Його перевагами є висока місткість контейнера, простота реалізації та відсутність помітного впливу на якість відтворення.
Для збільшення обсягу прихованих даних інколи модифікують не один, а два молодших біти цифрового представлення сигналу. У такому випадку значення 170 може змінитися до 172. Навіть така зміна залишається практично непомітною для більшості аудіоматеріалів. Водночас ступінь помітності спотворень значною мірою залежить від характеру звукового контенту. У записах із високим рівнем фонового шуму, наприклад шумом транспорту або людського натовпу, подібні зміни майже неможливо виявити. Натомість у високоякісних музичних композиціях або сольному виконанні академічних творів навіть незначні спотворення можуть бути помітними під час детального аналізу.
Для коректного вилучення прихованого повідомлення необхідно знати розташування вибірок, у які були внесені зміни. Як правило, обсяг секретної інформації повинен бути меншим за місткість контейнера, що забезпечує рівномірний розподіл прихованих даних по всьому аудіофайлу. Такий підхід дозволяє уникнути статистичних аномалій, які можуть бути виявлені засобами стеганоаналізу.
Одним зі способів підвищення непомітності є випадковий розподіл модифікованих бітів по всій структурі аудіофайлу. Однак використання повністю випадкового розташування створює складнощі під час відновлення прихованої інформації, оскільки отримувач повинен знати точне місце розташування всіх змінених елементів.
Більш практичним рішенням є застосування псевдовипадкової послідовності. У цьому випадку генератор випадкових чисел ініціалізується спеціальним параметром - секретним ключем. Використання однакового початкового значення дозволяє як відправнику, так і одержувачу сформувати ідентичну послідовність координат для запису та вилучення прихованих даних. Після отримання контейнера користувач відтворює ту саму послідовність і виконує процедуру відновлення повідомлення.
Разом із тим застосування одного й того самого ключа для великої кількості повідомлень негативно впливає на рівень захищеності системи. У разі його компрометації зловмисник отримує можливість аналізувати всі передані повідомлення. Тому для підвищення стійкості до криптоаналітичних атак доцільно генерувати окремий ключ для кожного сеансу передавання інформації. Такий підхід вимагає реалізації додаткових механізмів безпечного розповсюдження ключової інформації між учасниками обміну даними [21].

2.2 Застосування графічних файлів в стеганографічних контейнерах

Цифрові зображення належать до найбільш поширених типів стеганографічних контейнерів завдяки значній місткості та високій стійкості до візуального виявлення змін. Методи приховування інформації в графічних файлах умовно поділяються на дві основні групи: форматні та неформатні.
Форматні методи базуються на використанні особливостей структури файлів і передбачають запис інформації до службових полів, коментарів або метаданих. Неформатні методи працюють безпосередньо з графічними даними, змінюючи окремі біти пікселів зображення.
Найбільш поширеним неформатним підходом є метод модифікації молодших значущих бітів (Least Significant Bit, LSB). Його принцип полягає у заміні одного або кількох молодших бітів кольорових компонентів пікселів на біти прихованого повідомлення. Оскільки така модифікація викликає лише незначні зміни кольору, вона практично не сприймається людським зором [23].
Для реалізації цього методу використовуються молодші біти каналів RGB та, за наявності, альфа-каналу. Кількість бітів, що підлягають заміні, залежить від необхідного обсягу прихованої інформації та допустимого рівня спотворення зображення. У більшості випадків модифікація одного або двох молодших бітів не призводить до помітного погіршення якості контейнера.
Під час приховування текстових даних кожен символ кодується певною кількістю бітів. Відповідно для розміщення одного символу використовується декілька пікселів зображення. Під час декодування виконується зчитування змінених бітів та їх об’єднання у вихідну послідовність даних.
Важливим аспектом є вибір схеми розташування модифікованих пікселів. Найпростішим варіантом є послідовний запис інформації в усі пікселі зображення. Проте такий підхід знижує рівень прихованості та полегшує проведення статистичного аналізу контейнера. Для підвищення захищеності застосовують псевдовипадкові або спектральні схеми розподілу даних [24]. У цьому випадку для вилучення повідомлення необхідно відтворити карту розташування змінених пікселів, використовуючи відповідний ключ або алгоритм генерації координат.
Якщо повідомлення займає лише частину контейнера, додатково необхідно зберігати інформацію про його довжину. Для підвищення рівня захисту приховані дані зазвичай попередньо шифруються симетричним алгоритмом, а ключ дешифрування передається адресату окремим захищеним каналом зв’язку.

2.3 Використання PDF файлів як стеганографічний контейнер

Результати досліджень, наведених у роботі [2], демонструють високий потенціал PDF-документів як середовища для приховування інформації. Згідно з проведеним порівняльним аналізом різних типів контейнерів, найвищу оцінку отримав мережевий трафік, тоді як PDF- та ZIP-файли посіли друге місце за сукупністю характеристик.

1 Таблиця 2. Оцінка якості стеганоконтейнера. «No Назва стегано-котейнера Рівно-мірність Ємність контей-нера Рідкість викорис-тання Ємність контей-нера Важкість руйну-вання Загаль-ний бал 11 Малюнок 5 4 2 4 1 12 12 Відео та аудіо 4 5 3 5 2 14 33 Текст файл 1 3 4 3 1 9 44 Мережевий трафік 3 5 5 5 5 18
55 ZIP та PDF файли 4 4 4 4 4 15

Високі показники PDF-контейнерів пояснюються їх значною місткістю, стійкістю до пошкодження даних, відносно низькою поширеністю використання у стеганографічних системах та складною внутрішньою структурою. У межах даної роботи основна увага приділяється саме PDF-документам, тоді як дослідження можливостей ZIP-архівів залишаються перспективним напрямом подальших досліджень.

Структура PDF-документа дозволяє реалізовувати широкий спектр методів приховування інформації. Одним із підходів є використання модифікації молодших бітів у потоках даних документа [25]. У цьому випадку PDF розглядається як набір взаємопов'язаних об'єктів, параметри яких можуть бути змінені без помітного впливу на зовнішній вигляд документа.

Дослідники також пропонують використовувати надлишкові елементи словників об'єктів, параметри форматування тексту, незначні зміни числових значень операторів, а також приховані символи та модифікацію міжслівних інтервалів. Такі зміни залишаються практично непомітними для користувача, але дозволяють ефективно розміщувати додаткову інформацію всередині документа.

У роботі Ndounda R. та Ekodeck [26] запропоновано підхід, заснований на використанні китайської теореми про залишки. Повідомлення розбивається на окремі фрагменти, які вбудовуються у координати розташування текстових або графічних елементів документа. Перевагою такого методу є високий рівень захисту, оскільки відновлення повідомлення неможливе без знання всіх параметрів алгоритму.

Інший підхід передбачає використання PDF як середовища розподіленого зберігання інформації [27]. У цьому випадку секретні дані розподіляються між різними сторінками та об'єктами документа шляхом зміни параметрів шрифтів, використання невиконуваного коду або прихованих службових структур. Додатково розглядається можливість маніпуляції внутрішньою структурою документа, зокрема зміни посилань між сторінками без видалення самих об'єктів.

Окремі дослідження [29] присвячені методам приховування даних у текстовому шарі документа шляхом модифікації вирівнювання тексту. Незначні зміни міжслівних інтервалів можуть використовуватися для кодування двійкової інформації. Завдяки тому, що PDF-документ зберігає точні координати кожного текстового елемента, приховані дані можуть бути відновлені з високою точністю.

У роботі [30] розглядаються питання використання PDF-документів для приховування програмного коду та інших потенційно небезпечних об'єктів. Аналізуються способи застосування закладок, анотацій, вбудованих файлів та JavaScript-коду для транспортування додаткової інформації. Результати цих досліджень є важливими не лише для розвитку стеганографії, а й для вдосконалення методів виявлення аномалій у структурі PDF-документів.

Отже, PDF-файли можуть розглядатися як універсальний стеганографічний контейнер, що підтримує використання різноманітних методів приховування інформації. До них належать модифікація службових структур, застосування алгоритмів LSB для вбудованих зображень, зміна параметрів форматування тексту, маніпуляція посиланнями між об'єктами документа, використання прихованих елементів структури та розміщення даних після завершення основного вмісту файлу. Така різноманітність підходів забезпечує високу гнучкість та значний потенціал PDF-документів для застосування в сучасних стеганографічних системах.

2. РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Структура PDF файлу

Використання PDF-документів як середовища для прихованого зберігання інформації має низку суттєвих переваг. Насамперед до них належать:
- значна місткість контейнера, що дозволяє розміщувати повідомлення великого обсягу;
- широке поширення формату PDF у системах електронного документообігу, електронній пошті та сучасних месенджерах;
- висока стійкість документа до випадкових змін та пошкоджень під час передавання або зберігання;
- чітко визначена структура файлу, яка спрощує розроблення алгоритмів вбудовування та вилучення прихованих даних.

Типовий PDF-документ складається з декількох логічних компонентів, кожен з яких може бути використаний для реалізації стеганографічних методів. Файл починається із заголовка, який містить службову інформацію про формат документа. Перший рядок зазвичай має вигляд «%PDF-1.x», де остання частина визначає версію специфікації PDF. Символ «%» позначає початок коментаря, тому інформація до кінця рядка розглядається програмою як службове повідомлення.

Наступний рядок також починається із символу коментаря та містить спеціальні байтові послідовності, що дозволяють програмному забезпеченню ідентифікувати документ як бінарний файл.

Основну частину PDF-документа складають непрямі об'єкти (Indirect Objects). Саме вони формують структуру документа та містять його вміст. До таких об'єктів належать сторінки, шрифти, графічні елементи, потоки даних і службові структури. Кожний об'єкт починається службовим записом виду «obj» та завершується директивою «endobj».

Всередині об'єктів можуть зберігатися різні параметри документа, серед яких:
- перелік сторінок документа;
- посилання на дочірні сторінки;
- загальна кількість сторінок;
- геометричні параметри сторінок;
- вміст сторінки;
- інформація про використані шрифти та графічні ресурси.

Особливе місце у структурі PDF займають потоки даних (Streams). Вони містять безпосередньо текстову, графічну або іншу інформацію документа. Потоки можуть бути стиснені різними алгоритмами компресії, які визначаються за допомогою параметра «Filter». Додатково для кожного потоку задається його довжина, а завершення блоку позначається службовим словом «endstream».

Фундаментальним елементом архітектури PDF є словники (Dictionaries), які забезпечують логічний зв'язок між окремими компонентами документа. Якщо потоки даних містять фактичну інформацію, то словники визначають правила її інтерпретації та відображення.

Структура PDF має ієрархічну організацію, де центральним елементом виступає словник Catalog. Він містить посилання на інші структурні компоненти документа, зокрема:
- перелік сторінок документа (Pages);
- дерево закладок (Outlines);
- іменовані об'єкти та посилання (Names).

Для роботи зі стисненими потоками використовуються спеціальні параметри словників. Наприклад, ключ «Filter» визначає алгоритм декодування інформації, необхідний для коректного відображення вмісту документа.

Завдяки системі непрямих посилань PDF-документ формує деревоподібну структуру взаємопов'язаних об'єктів. Це дозволяє програмам швидко знаходити необхідні елементи без послідовного читання всього файлу. Координати об'єктів зберігаються у спеціальній таблиці перехресних посилань (xref).
Одним із важливих словників є Resources, який виконує роль каталогу ресурсів сторінки. У ньому містяться посилання на шрифти, графічні об'єкти та інші елементи, необхідні для відображення документа.
З точки зору стеганографії особливий інтерес становлять метадані документа. Для їх зберігання використовується словник Info, у якому можуть міститися відомості про автора документа, його назву, дату створення, дату модифікації та інші службові параметри. Структура словника допускає створення додаткових користувацьких полів, що робить його зручним місцем для прихованого розміщення інформації.
На рис. 3.1 зображений типовий заголовок PDF файлу. Заголовок файлу починається з символу % PDF - 1.5 що в даному випадку вказує на версію PDF файлу 1.3. Символ % на початку рядка означає що далі йде службовий коментар і все, що починається після нього є коментарем.
Наступний рядок також починається із знаку коментаря, далі йдуть коди, які сповіщають програмам, що цей файл бінарний.
Наступний фрагмент файлу - це саме тіло файлу, що містить набір не прямих об'єктів (Indirect Objects). На рис. 3.2 представлений типовий об'єкт потоку даних. Містить фактичний вміст тексту або графіки та може бути стисненим (/Filter /FlateDecode). В даному випадку тут вказується алгоритм стиснення.

Рисунок 3.1 - Типовий заголовок PDF файлу

Визначається також довжина цього блоку в директиві "/Length 4201 ". Кінець потоку обрамляється службовим словом "endstream".
У структурі PDF словники (Dictionaries) - це фундаментальний тип даних, на якому тримається вся логіка файлу. Якщо об'єкти-потоки (Streams) зберігають "важку" інформацію (текст, картинки), то словники - це інтелект файлу, який каже програмі, як цю інформацію читати.

Рисунок 3.2 - Типовий об'єкт потоку даних PDF файлу

Завершальна частина PDF-документа містить таблицю перехресних посилань xref та службовий блок Trailer. У ньому зберігається інформація про розмір документа, кількість об'єктів, кореневий словник та унікальний ідентифікатор файлу. Кінець документа позначається службовою директивою «%%EOF», яка сигналізує програмі про завершення обробки файлу.
Для прикладу на рис. 3.3 приведений кінець файлу стандартного PDF файлу.

Рисунок 3.3 - Стандартний кінець PDF файлу

- Використання коментарів та метаданих

Аналіз внутрішньої структури PDF-документів показує, що одним із найпростіших способів приховування інформації є використання коментарів. Будь-який коментар у PDF починається символом «%», після якого може розміщуватися довільна текстова або бінарна інформація. Деякі службові конструкції також використовують цей символ, наприклад запис «%PDF» визначає версію формату документа, а директива «%%EOF» позначає його завершення.
Стандартні засоби перегляду PDF-документів не відображають вміст коментарів і зазвичай припиняють аналіз файлу після досягнення позначки «%%EOF». Водночас спеціалізовані редактори та переглядачі текстових файлів дозволяють переглядати внутрішню структуру документа та аналізувати всі його компоненти.
Основними перевагами використання коментарів для приховування інформації є простота реалізації та можливість розміщення повідомлень значного обсягу. Недоліком такого підходу є відносна легкість виявлення прихованих даних у разі цілеспрямованого аналізу структури документа.
Для підвищення надійності приховування інформацію доцільно розміщувати між окремими об'єктами документа, наприклад після завершення одного блоку та перед початком наступного. Для ідентифікації прихованого повідомлення може використовуватися спеціальний маркер або унікальна сигнатура, яка додається після символу коментаря та дозволяє однозначно визначити початок секретних даних.
Під час вилучення інформації виконується пошук заздалегідь визначеного маркера, після чого здійснюється зчитування даних до завершення поточного рядка або до появи іншого службового роздільника. Такий механізм забезпечує просту та швидку реалізацію процедур приховування й відновлення повідомлень.
Іншим перспективним напрямом є використання метаданих документа. У сучасних PDF-файлах інформація про документ може зберігатися за допомогою двох основних механізмів: словника Info Dictionary та платформи XMP Metadata (Extensible Metadata Platform). Обидва підходи можуть функціонувати одночасно та містити дубльовану інформацію про документ.
Словник Info Dictionary містить стандартний набір атрибутів, серед яких автор документа, назва, тема, ключові слова, дата створення та дата останнього редагування. Окрім стандартних полів, структура словника дозволяє додавати додаткові користувацькі параметри, які можуть використовуватися для розміщення прихованих даних без зміни зовнішнього вигляду документа. Основні поля словника Info Dictionary наведені в таблиці 3.1.
Таблиця 3.1 Значення ключів словника Info Dictionary

Опис	Ключ	Тип
Назва документу	/ Title	string
Автор	/ Author	string
Тема	/ Subject	string
Ключові слова	/ Keywords	string
Програма що створила вміст	/Creator	string
Програма що згенерувала PDF	/Producer	string
Дата створення	/CreationDate	date string
Дата останньої зміни	/ModDate	date string
Обробка кольорів	/Trapped	name

Особливістю словника Info Dictionary є використання спеціального формату представлення дати та часу, прийнятого у специфікації PDF. Наприклад, дата може бути записана у вигляді D:20260403153000+02'00'. Для кодування текстових значень застосовуються формати PDFDocEncoding або UTF-16BE. Хоча окремі програмні засоби допускають додавання нестандартних полів до словника, така практика не належить до офіційно рекомендованих механізмів специфікації PDF. До основних недоліків Info Dictionary належать обмежені можливості розширення, відсутність підтримки просторів імен та складних структур даних, а також спрощена організація внутрішнього представлення інформації.
Більш сучасним підходом до зберігання метаданих є використання технології XMP Metadata (Extensible Metadata Platform). Підтримка XMP була впроваджена починаючи з версії PDF 1.4 і надалі стала основним механізмом опису додаткових властивостей документа.
Технологія XMP базується на стандартах RDF/XML та являє собою окремий об'єкт типу Metadata, який вбудовується безпосередньо в структуру PDF-документа. Посилання на цей об'єкт зберігається у словнику Catalog або може бути визначене для окремих сторінок документа.
На рис 3.4 представлений типовий вигляд Metadata в форматі XML.

Рисунок 3.4 - MetaData в форматі XML
Типова структура XMP-метаданих представлена у форматі XML. Однією з ключових переваг такого підходу є використання просторів імен, що дозволяє зберігати інформацію різних категорій без виникнення конфліктів між атрибутами. Найчастіше використовуються простори імен Dublin Core (dc:), Adobe XMP (xmp:), XMP Rights Management (xmpRights:), Photoshop Metadata (photoshop:) та IPTC Core (Iptc4xmpCore:). Крім того, користувач може створювати власні простори імен та формувати складні ієрархічні структури даних.
У специфікації PDF 2.0 словник Info Dictionary офіційно визнаний застарілим механізмом зберігання метаданих, а пріоритет віддано саме XMP Metadata. Проте для забезпечення сумісності зі старішими програмами більшість сучасних редакторів PDF продовжують одночасно формувати обидва набори метаданих. У випадку виникнення суперечностей між їх значеннями перевага надається інформації, що міститься в XMP. На рис. 3.5 представлена схема одночасного співіснування XMP Metadata і Info Dictionary.

Рисунок 3.5 - Схема одночасного використання двох схем MetaData

Таким чином, структура PDF-документа надає щонайменше два зручні середовища для прихованого розміщення службової або конфіденційної інформації: словник Info Dictionary та контейнер XMP Metadata.
Додатковим способом приховування даних може бути запис інформації після службової директиви %%EOF, яка визначає логічне завершення PDF-документа. Більшість програм для перегляду PDF припиняють обробку файлу після досягнення цієї позначки та ігнорують подальший вміст. Водночас операційна система під час копіювання або збереження файлу орієнтується на його фактичний розмір, а не на внутрішні службові маркери. Завдяки цьому дані, записані після директиви %%EOF, фізично залишаються у файлі та можуть бути збережені під час його копіювання або передавання.

- Захист та шифрування вбудованої інформації

Наявність прихованого повідомлення у контейнері не гарантує його захищеності від несанкціонованого доступу. Тому під час розроблення стеганографічної системи необхідно передбачати ситуацію, за якої зловмисник зможе виявити факт приховування інформації та отримати доступ до контейнера. У такому випадку додатковим рівнем захисту виступають криптографічні механізми, що унеможливлюють використання вилучених даних без знання відповідних ключів.
Одним із найбільш поширених симетричних алгоритмів шифрування на сьогодні є AES (Advanced Encryption Standard). Даний алгоритм був затверджений Національним інститутом стандартів і технологій США (NIST) у 2001 році та залишається актуальним завдяки високій криптографічній стійкості та ефективності реалізації на сучасних обчислювальних системах.
AES належить до симетричних криптографічних алгоритмів, тобто для шифрування та дешифрування використовується один і той самий секретний ключ, який повинен бути відомий як відправнику, так і одержувачу повідомлення.
Алгоритм виконує обробку даних блоками фіксованого розміру 128 біт. Залежно від рівня захисту можуть використовуватися ключі довжиною 128, 192 або 256 біт. Кількість раундів перетворення визначається довжиною ключа та становить 10, 12 і 14 раундів відповідно. Кожен раунд включає послідовність математичних операцій підстановки, перестановки та змішування даних, що забезпечують високий рівень криптографічної стійкості алгоритму.
У реалізації AES-128 вхідний блок даних розміром 16 байтів розглядається як матриця розміром 4×4. Байти послідовно заповнюють стовпці цієї матриці, утворюючи початковий стан алгоритму. Аналогічним чином формується матриця секретного ключа, яка використовується під час генерації раундових ключів та виконання криптографічних перетворень.
У процесі шифрування формується так звана матриця стану (State Matrix), яка змінюється після виконання кожного раунду алгоритму. Після завершення останнього раунду отримана матриця перетворюється у вихідну послідовність байтів (Output Block), що являє собою зашифрований блок даних.
Схематичний взаємозв'язок між вхідним блоком даних, матрицею стану, ключовою інформацією та вихідним блоком наведено на відповідному рисунку. Схематичне відношення між матрицями зображено на рисунку 3.6

Рисунок 3.6 - Відношення між матрицями

Криптографічний ключ алгоритму симетричного шифрування AES-128 має довжину 128 бітів, які структуровані у вигляді 16 байтів (k0, k1...k15) і позиціонуються у стовпцях матриці стану InputKey. Таким чином, первинний ключ представлений чотирма 32-бітними словами: w0w1w2w3, де w 0= k0k1k2k3. На основі цих слів за допомогою процедури розширення ключів (Key Expansion) генерується послідовність із 44 слів (w0...w43). Для кожного раунду шифрування обчислюється і подається відповідне підмножина з чотирьох слів зазначеної послідовності.
Кожен ітераційний раунд алгоритму AES (за винятком фінального) складається з чотирьох послідовних перетворень:
SubBytes - нелінійна побайтова заміна в матриці станів із використанням фіксованої таблиці замін (S-Box). Ця операція полягає в знаходженні оберненого елемента в полі Галуа GF(2 8) із наступним афінним перетворенням, що забезпечує усунення лінійних залежностей.
ShiftRows - циклічний побайтовий зсув рядків матриці стану на різні зміщення.

MixColumns - лінійне перетворення, що дифузує байти всередині стовпців матриці.
AddRoundKey - накладання раундового ключа за допомогою операції побітового додавання за модулем 2 (XOR).
Заключний раунд модифікований порівняно з попередніми та не містить етапу MixColumns.
Головним деструктивним чинником під час практичної реалізації симетричних криптосистем є необхідність безпечного розподілу та обміну ключами між відправником та отримувачем. Перехоплення ключа на етапі його транспортування нівелює рівень захисту інформації, забезпечений алгоритмом.
Асиметричні криптосистеми, зокрема алгоритм RSA, позбавлені зазначеного недоліку. Криптографічна стійкість RSA базується на обчислювальній складності задачі факторизації добутку двох великих простих чисел. Функціонування алгоритму передбачає використання ключової пари: відкритого ключа (для шифрування) та таємного ключа (для дешифрування). Дані, що пройшли процедуру зашифрування за допомогою публічного ключа, можуть бути відновлені виключно власником відповідного приватного ключа, що гарантує конфіденційність транзакцій. Крім того, архітектура асиметричного шифрування є базисом для формування електронних цифрових підписів.

Асиметричні методи шифрування, наприклад RSA, не мають таких вразливостей. RSA - це асиметричний криптографічний алгоритм, що базується на складності факторизації великих чисел, який використовує пару ключів: відкритий (для шифрування) та закритий (для дешифрування). Назва методу походить з перших літер прізвищ вчених, що розробили цей алгоритм - Rivest, Shamir та Adleman. Дані зашифровані публічним ключем може розшифрувати лише власник відповідного приватного ключа, що гарантує конфіденційність та безпеку обміну даними. Також такий підхід використовують для накладення цифрових підписів.
Математичний опис алгоритму RSA.
Процес функціонування криптосистеми RSA розподіляється на три основні етапи.
1. Генерація ключової пари:
Обираються два випадкові великі прості числа p та q (еквівалентної довжини близько 512 бітів кожне).
Обчислюється модуль системи:
1. n=pq ;
2. визначається значення функції Ейлера від модуля: φ(n)=(p-1)(q-1) INCLUDEPICTURE "/Users/yuriidaus/Library/Group Containers/UBF8T346G9.ms/WebArchiveCopyPasteTempFiles/com.microsoft.Word/d79368781093e6f95c0ccb912f75c776bb153551" * MERGEFORMATINET ;
3. обирається відкрита експонента - ціле число e, яке задовольняє умовам 1<e<φ(n) що 1<e<φ(n) INCLUDEPICTURE "/Users/yuriidaus/Library/Group Containers/UBF8T346G9.ms/WebArchiveCopyPasteTempFiles/com.microsoft.Word/29e768da2e57dd39879d6603775178ed43677aa6" * MERGEFORMATINET та e **взаємно** просте з φ(n) INCLUDEPICTURE "/Users/yuriidaus/Library/Group Containers/UBF8T346G9.ms/WebArchiveCopyPasteTempFiles/com.microsoft.Word/f067864064667dd5f8b2508b9cbf983d89788629" * MERGEFORMATINET
4. за допомогою розширеного алгоритму Евкліда обчислюється таємна експонента d, яка є мультиплікативно оберненою до числа e за модулем ed≡1(mod φ(n)) INCLUDEPICTURE "/Users/yuriidaus/Library/Group Containers/UBF8T346G9.ms/WebArchiveCopyPasteTempFiles/com.microsoft.Word/4ac63e575eea0f87ec196b9e01462f211d40db0d" * MERGEFORMATINET
2. Процедура зашифрування:
Для передачі повідомлення M здійснюється його попереднє кодування у ціле число m (0≤m<n) за допомогою стандартизованих оборотних схем доповнення (наприклад, OAEP). Обчислення криптотексту c виконується з використанням відкритого ключа e за формулою:

c= mod n (3.1)

Швидкодія цієї операції для чисел великої розрядності досягається завдяки застосуванню алгоритмів зведення до степеня за модулем.Розшифрування.
3. Процедура дешифрування:
Для відновлення вихідного повідомлення m із криптотексту c обчислюється таке рівняння:
m= mod n (3.2)

Коректність відновлення початкових даних на основі виразів (3.1) та (3.2) доводиться в такий спосіб:

≡)≡ (mod n) (3.3) За умови: ed ≡ 1 (mod φ(n)) (3.4)

Враховуючи умову (3.4):
ed = k φ(n)+1 (3.5)

для деякого цілого k, отже:

≡ (mod n) (3.6)

Згідно з теоремою Ейлера існує тотожність:

≡ 1 (mod n) (3.7)
Тому:

≡ m (mod n) (3.8)
≡ m (mod n) (3.9)

Попри високу надійність та відсутність етапу компрометації ключів під час їх передачі, алгоритм RSA має низку обмежень. Зокрема, до них належать низька обчислювальна швидкість порівняно з симетричними аналогами (наприклад, AES), значна надмірність довжини ключів (від 2048 до 4096 бітів) та обмеження щодо обсягу даних, які можуть бути зашифровані за один ітераційний прохід (обсяг не може перевищувати розмір модуля n). Сучасні криптографічні стандарти вимагають використання високих показників степеня, що призводить до значних апаратних та часових затрат при роботі з великими цілими числами.

Невід'ємним аспектом захисту інформації є контроль цілісності повідомлень під час їх транзиту через незахищені мережеві вузли, де існує ризик несанкціонованої модифікації або підміни даних. Для верифікації цілісності електронних документів застосовують односторонні криптографічні хеш-функції, серед яких найбільш поширеним є алгоритм SHA-256.

До фундаментальних властивостей SHA-256 належать:

Фіксований розмір дайджесту: вихідний результат завжди має довжину 256 бітів (64 символи в шістнадцятковій системі числення).

Незворотність (односпрямованість): обчислювальна неможливість відновлення первинного тексту на основі його хеш-образу.

Стійкість до колізій: висока математична складність пошуку двох різних вхідних масивів даних, що мають ідентичні хеш-коди.

Ефект лавини (унікальність): мінімальна модифікація вхідного повідомлення (навіть на один біт) призводить до радикальної зміни результуючого хешу.

Технологічний процес обчислення дайджесту за алгоритмом SHA-256 містить такі етапи:

Попередня обробка (Padding): вхідний інформаційний масив доповнюється таким чином, щоб його кінцева довжина в бітах відповідала умові 512×n+448. Процедура реалізується додаванням біта «1» з наступним заповненням нульовими бітами.

Інтеграція довжини: до кінця блоку додається 64-бітне значення, що описує вихідну довжину повідомлення до початку обробки. Це забезпечує кратність загального обсягу даних 512 бітам.

Ініціалізація векторів стану: встановлюються початкові значення 8 робочих 32-бітних змінних (a...h), які отримують із дробових частин квадратних коренів перших восьми простих чисел.

Поблокова обробка: інформаційний потік розділяється на блоки розміром 512 бітів. Кожен блок піддається 64 раундам трансформацій, що включають циклічні зсуви, бітові операції XOR та логічні функції.

Фіналізація: після обробки всієї послідовності блоків поточні значення робочих змінних конкатенуються, формуючи підсумковий 256-бітний хеш-код.

Для верифікації автентичності документа інтеграцію згенерованого хеш-рядка доцільно здійснювати шляхом його додавання після фінальної директиви %%EOF у структурі файлу. Такий підхід зумовлений тим, що операційна система сприймає ці дані як легітимну область файлової структури, тоді як спеціалізоване програмне забезпечення для візуалізації форматів (наприклад, PDF) ігнорує інформаційні блоки, розташовані після цієї директиви.

Для підвищення загального рівня стійкості інформаційної системи до компрометації рекомендується застосовувати гібридну схему захисту. У межах такого підходу процедура шифрування самого інформаційного масиву реалізується швидким симетричним алгоритмом AES, тоді як обчислений хеш-рядок підлягає зашифруванню за допомогою асиметричної криптосистеми RSA.

- Структура додатку

Проектування та розробка програмного комплексу реалізована на базі об'єктно-орієнтованої мови програмування Java. Вибір даного технологічного стеку обґрунтований такими функціональними вимогами до системи:

забезпечення кросплатформеності програмного забезпечення;

наявність розвинених інтегрованих криптографічних бібліотек (Java Cryptography Architecture / Java Cryptography Extension);

високий рівень поширеності технології та підтримки спільноти;

можливість дистрибуції у вигляді автономного виконуваного модуля.

Для формування результуючого автономного модуля виконується збірка артефактів проєкту з генерацією самовиконуваного архівного пакета формату .jar. Графічне представлення структури розроблених класів, їхніх методів та взаємозв'язків наведено на діаграмі класів (рис. 3.7).

Рисунок 3.7 - діаграма класів додатку

Для проектування та реалізації графічного інтерфейсу користувача (GUI) було застосовано бібліотеку Swing. Архітектура розробленого програмного забезпечення базується на взаємодії таких основних класів:

KeyPairGenerator - забезпечує генерацію пар ключів (відкритого та таємного) для асиметричної криптосистеми й керує процедурою їх збереження у визначеній директорії файлової системи.

Cipher - призначений для криптографічного перетворення вхідних повідомлень за симетричним алгоритмом AES із довжиною ключа 128 бітів на основі парольної фрази.

EmbedMessage, ExtractMessage - реалізують алгоритми вбудовування (стеганографічного приховування) та автентичного вилучення конфіденційної інформації з контейнера відповідно.

verifyFileHash - виконує процедуру верифікації цілісності даних шляхом обчислення поточного хеш-образу та його порівняння з еталонним значенням.

PDDocument - забезпечує низькорівневу взаємодію з файлами формату PDF (завантаження об'єктної структури, запис модифікованих даних, редагування метаданих).

MessageDigest - інтегрує функції стеганографічного приховування, екстракції даних, а також повної деструкції прихованих повідомлень і супутніх хеш-рядків із метою відновлення первинного стану файлу-контейнера.

Графічну інтерпретацію головного вікна програмного комплексу наведено на рис. 3.8. Панель інструментів верхнього меню структурована за трьома функціональними секціями. Перша секція об'єднує модулі керування файлами, підсистему хешування та блоки криптографічного захисту. Зокрема, вкладка «Файл» містить чотири операційні підпункти: «Завантажити PDF», «Зберегти як (приховати сповіщення)», «Витягти прихований текст» та «Повна очистка файлу» . Функціонування зазначених компонентів безпосередньо детермінується станом перемикачів, розташованих у другій та третій секціях інтерфейсу.

Рисунок 3.8 - Головне меню додатку

Після ініціалізації та зчитування структури PDF-файлу стає доступною процедура стеганографічного вбудовування повідомлення в тіло документа. Для реалізації цього процесу конфіденційні дані вводяться у ліве текстове поле інтерфейсу, після чого активується функція «Зберегти як (Приховати текст)» (рис. 3.9). Попередньо користувач має визначити параметри криптографічного захисту (метод шифрування) та локалізацію прихованого блоку в структурі контейнера. У разі обрання асиметричного алгоритму RSA обов'язковою передумовою є попередня генерація ключової пари з її подальшим експортом на локальний диск.

Рисунок 3.9 - Вбудовування прихованого сповіщення

Процедура генерації криптографічних ключів реалізується у розділі «Криптографія» за допомогою активації команди «Згенерувати ключі RSA» (рис. 3.10). На відміну від асиметричного підходу, використання алгоритму AES не потребує генерації статичних ключів - система ініціює діалогове вікно для введення парольної фрази, яка є чутливою до регістру та мовної розкладки клавіатури.

Рисунок 3.10 - Генерація приватного та публічного ключа

Функціональний блок «Хешування» складається з підмодулів «Вбудувати Хеш» та «Перевірити Хеш» (рис. 3.11). Застосування цього інструментарію є доцільним на завершальному етапі стеганографічного перетворення для формування математичного підтвердження цілісності результуючого файлу. Для забезпечення високого рівня криптостійкості процес шифрування обчисленого хеш-рядка має відбуватися або за допомогою альтернативного пароля (при обранні AES), або із застосуванням асиметричного методу RSA. Обов'язковою науково-технічною рекомендацією є розмежування алгоритмів чи ключів шифрування, що застосовуються окремо для самого повідомлення та для його дайджесту.

Рисунок 3.11 - Захист вбудованої інформації

Процедура зворотного вилучення прихованої інформації передбачає завантаження стегоконтейнера, вибір відповідного криптографічного алгоритму в панелі налаштувань та активацію команди «Витягти прихований текст» у меню «Файл». За умови автентичності обраного алгоритму та введених ключових даних (або пароля), дешифрований текстовий масив візуалізується у правому діалоговому вікні (рис. 3.12).

Рисунок 3.12 - Витягування прихованого сповіщення

Після очищення правого текстового поля стає можливим виконання контролю цілісності контейнера з метою виявлення потенційних фактів модифікації даних під час їх транзиту. Для цього після конфігурації параметрів дешифрування у розділі «Хешування» обирається функція «Перевірити хеш». Нормативний результат успішної верифікації наведено на рис. 3.13. Збіг хеш-рядка, вилученого з файлу, з розрахованим програмою значенням свідчить про автентичність інформації. Для експериментальної перевірки стійкості підсистеми було здійснено штучну модифікацію одного байта даних у тілі прихованого повідомлення за допомогою бінарного редактора. Результати тестування (рис. 3.14) демонструють, що програмний комплекс оперативно фіксує порушення цілісності файлової структури, унеможливлюючи компрометацію та використання модифікованих даних.

Рисунок 3.13 - Успішна перевірка цілісності файлу
Після очищення правого текстового поля стає можливим виконання контролю цілісності контейнера з метою виявлення потенційних фактів модифікації даних під час їх транзиту. Для цього після конфігурації параметрів дешифрування у розділі «Хешування» обирається функція «Перевірити хеш». Нормативний результат успішної верифікації наведено на рис. 3.13. Збіг хеш-рядка, вилученого з файлу, з розрахованим програмою значенням свідчить про автентичність інформації. Для експериментальної перевірки стійкості підсистеми було здійснено штучну модифікацію одного байта даних у тілі прихованого повідомлення за допомогою бінарного редактора. Результати тестування (рис. 3.14) демонструють, що програмний комплекс оперативно фіксує порушення цілісності файлової структури, унеможливлюючи компрометацію та використання модифікованих даних.

Рисунок 3.14 - Порушення цілісності файлу
Таким чином, розроблене прикладне програмне забезпечення є повнофункціональним інструментом захисту інформації, який забезпечує:
1. стеганографічне вбудовування конфіденційних повідомлень як у метадані, так і в область коментарів PDF-документа;
2. криптографічний захист прихованих даних із використанням симетричних та асиметричних шифрів;
3. генерацію та розподіл ключової пари для асиметричних методів захисту;
4. формування та інтеграцію криптографічних дайджестів для контролю цілісності інформаційних масивів;
5. повну деструкцію стеганографічних елементів із відновленням початкової бінарної структури файлу-контейнера.

- Можливість використання застосунку під час військового стану

В умовах ведення сучасних бойових дій та дії правового режиму воєнного стану використання стеганографічних методів і засобів прихованої передачі даних є критично важливим аспектом забезпечення інформаційної безпеки, захисту військових підрозділів, а також об'єктів критичної та цивільної інфраструктури. Практичне впровадження розробленого програмного комплексу дозволяє нівелювати ризики перехоплення, деанонімізації та несанкціонованого доступу до вразливої інформації.
Визначено такі основні напрями застосування розробленого стеганографічного інструментарію:
Організація прихованого зв'язку між військовими підрозділами: забезпечення конфіденційного обміну тактичними даними, зокрема координатами цілей. Використання відкритих або слабко захищених каналів зв'язку створює ризики перехоплення інформації силами противника, що може призвести до зриву операцій або перегрупування ворожих сил.
Захист персональних та облікових даних особового складу: приховування відомостей щодо чисельності підрозділів, санітарних та безповоротних втрат, дислокації, а також графіків відпусток і відряджень, що унеможливлює використання цих даних ворогом для ведення розвідки та проведення інформаційно-психологічних операцій (ІПСО).
Захист інформації про стан об'єктів критичної інфраструктури: приховування оперативних даних щодо ступеня пошкодження енергетичних,

транспортних чи логістичних вузлів внаслідок вогневого ураження, термінів проведення аварійно-відновлювальних робіт, а також специфікації необхідних матеріально-технічних ресурсів.
Маскування відомостей про руйнування цивільної інфраструктури: обмеження відкритого доступу до деталізованої інформації про масштаби пошкоджень житлового фонду, втрати серед цивільного населення, локалізацію пожеж та обсяги знищеної техніки, що заважає противнику проводити коригування та оцінювати ефективність нанесених ударів.
Забезпечення конфіденційності звітності державних та стратегічних приватних підприємств: сучасні аналітичні системи на базі штучного інтелекту здатні агрегувати дані з відкритих джерел (обсяги виробництва, рівень енергоспоживання, логістичні маршрути, використання паливно-мастильних матеріалів) для формування точних висновків щодо загального стану економіки держави та військово-промислового комплексу зокрема. Стеганографічне приховування підсумкової та поточної звітності суттєво знижує розвідувальний потенціал держави-агресора.

4. ОХОРОНА ПРАЦІ

Для забезпечення безпечних і комфортних умов роботи фахівців галузі цифрових технологій необхідна ретельна ідентифікація та оцінка всіх шкідливих і небезпечних виробничих факторів і ризиків, які вони несуть для них. Крім того, щоб уникнути негативних наслідків для здоров'я фахівців ІТ- галузі слід своєчасно проводити відповідні профілактичні заходи. Тому в цьому розділі будуть описані актуальні шляхи ефективного вирішення деяких вищевказаних проблем.

4.1 Перспективи використання штучного інтелекту для зниження рівня виробничих небезпек.

Еволюція попереднього аналізу небезпек у виробничому середовищі та трансформаційна роль штучного інтелекту.
Традиційні підходи до охорони праці історично базувалися на реагуванні на інциденти, що вже відбулися, зосереджуючись на виправленні наслідків, а не на їх запобіганні. Однак, сучасні методології охорони праці наголошують на проактивному виявленні та оцінці ризиків, що є фундаментальним для забезпечення безпечних та здорових умов праці. Метою оцінки ризиків є допомога роботодавцям у ефективному вжитті заходів для захисту безпеки та здоров'я працівників, що є постійним та організованим процесом.
Основні етапи попереднього аналізу небезпек включають виявлення самих небезпек, визначення осіб, які можуть зазнавати ризику, оцінювання рівня цього ризику (як якісне, так і кількісне), розгляд можливості усунення причин небезпек, та прийняття рішення про необхідність запровадження подальших заходів для попередження чи зменшення ризику від них. Виявлення небезпек передбачає ретельний аналіз робочих умов, обладнання, матеріалів та технологій, що використовуються на робочому місці. Після ідентифікації небезпек проводиться оцінка ризиків, яка включає визначення ймовірності настання події та серйозності її наслідків.
Після оцінки ризиків розробляються та впроваджуються заходи контролю та запобіганню, Особлива увага приділяється усуненню або запобіганню ризикам як пріоритетним діям. Процес оцінки небезпек не є одноразовим заходом; він вимагає постійного моніторингу та перегляду, особливо у разі зміни технологій чи умов праці. Методичні підходи до визначення ризику охоплюють широкий спектр аналізів, включаючи загальний аналіз ризику складних систем, якісний аналіз небезпек, попередній аналіз небезпек (ПАН), аналіз дерева відмов (АДВ) та багато інших.
ПАН є ключовим компонентом проактивного управління безпекою. Це систематична ідентифікація та оцінка потенційних небезпек на ранніх стадіях проєкту, системи чи операції. Цей проактивний підхід допомагає оцінювати, ідентифікувати та відстежувати потенційні небезпеки до того, як вони стануть проблемами, що значно зменшує кількість інцидентів на робочому місці.
Ризикоорієнтований підхід є фундаментальним у сучасній охороні праці, вимагаючи від роботодавців забезпечення безпеки та здоров'я працівників у кожному аспекті, пов'язаному з роботою. Це включає виявлення небезпек, оцінку пов'язаних з ними ризиків та визначення необхідних заходів захисту. Оцінка ризиків повинна відповідати вимогам законодавства, стандартам та настановам, таким як національні технічні інструкції та зводи практичних правил. Серед них слід виділити Стандарт ISO 45001, що визначає систематичний підхід до ідентифікації, оцінки та пом'якшення ризиків для захисту працівників та Постанову Кабінету Міністрів України від 26.10.2011 No 1107, яка передбачає цифровізацію порядку видачі дозволів.
Поява штучного інтелекту (ШІ) як ключового фактора, що значною мірою змінює концерцію безпеки праці, знаменує собою значний крок уперед. ШІ та цифровізація кардинально змінюють робочі місця та робочі процеси, пропонуючи нові можливості для підвищення рівня безпеки. Важливо розуміти, що ШІ є доповненням до людського інтелекту, а не його заміною, оскільки він не має емоцій, сумнівів чи свідомості. Здатність ШІ прогнозувати потенційні небезпеки до їх прояву революціонізує управління безпекою, дозволяючи ідентифікувати та пом'якшувати нещасні випадки ще до їх виникнення. ШІ може аналізувати величезні обсяги даних швидко та точно, що робить його безцінним інструментом для підвищення безпеки та забезпечення відповідності нормативним вимогам. Це дозволяє організаціям переходити від реактивного до проактивного управління ризиками, що є фундаментальною зміною у філософії безпеки праці.
ШІ, завдяки своїй здатності до прогнозного аналізу та моніторингу в реальному часі, дозволяє не просто виявляти, а передбачати небезпеки до їх виникнення. Цей підхід не лише зменшує кількість інцидентів та травм, але й змінює роль фахівців з охорони праці з "пожежників", які реагують на кризи, на "архітекторів безпеки", що дозволяє їм зосередитися на стратегічному плануванні та постійному вдосконаленні систем безпеки.
Оцінка ризиків в охороні праці вимагає збору різноманітних даних, що стосуються умов праці, обладнання, матеріалів, інцидентів, а також законодавчих вимог та стандартів. Ці дані часто є розрізненими та зберігаються в різних форматах, що ускладнює їхній комплексний аналіз. ШІ, завдяки своїй здатності ефективно обробляти величезні обсяги даних з різних джерел, пропонує рішення цієї проблеми. Це відкриває шлях до створення єдиних, інтегрованих платформ для управління даними з охорони праці, які раніше були розрізненими. Стандартизація та агрегація даних через ШІ дозволить отримати більш повну та точну картину ризиків, що є критично важливим для прийняття обґрунтованих рішень та забезпечення послідовності у застосуванні заходів безпеки.
Штучний інтелект у попередньому аналізі небезпек: Основні принципи та практичні застосування.
Фундаментом ШІ-орієнтованого прогнозування небезпек є надійний збір та інтеграція даних. ШІ-системи здатні обробляти величезні обсяги даних, включаючи зображення, відео та вхідні дані з сенсорів. Нові застосування сенсорних технологій дозволяють швидше збирати та передавати дані від широкого кола різних джерел, забезпечуючи оперативне реагування на відхилення від норми та запобігання небезпечним ситуаціям.
Пристрої Інтернету речей (IoT) та сенсорні мережі, інтегровані з ШІ, можуть безперервно відстежувати умови навколишнього середовища та стан обладнання. Це включає моніторинг температури, вологості, концентрації шкідливих газів та рівня шуму, що дозволяє виявляти потенційні

загрози в реальному часі. Дані також збираються з архівних записів, таких як минулі інциденти та аудити безпеки.
Спеціалізоване програмне забезпечення для управління інцидентами автоматично збирає дані про інциденти та зберігає їх у централізованому сховищі, що спрощує аналіз та виявлення закономірностей. Людський фактор також є важливим джерелом даних. Інформація збирається через спостереження, консультації та опитування працівників, які часто мають безпосередні знання про ризики на робочому місці.
Прогнозний аналіз небезпек використовує алгоритми ШІ для прогнозування потенційних загроз. Він аналізує великі масиви даних з безпеки, виявляючи закономірності та передбачаючи потенційні небезпеки, що дозволяє роботодавцям усувати ризики до виникнення інцидентів.[8]
Машинне навчання (МН) є основним компонентом ШІ в управлінні ризиками. Його моделі навчаються на історичних даних для розпізнавання моделей ризиків, прогнозування результатів та рекомендації запобіжних дій. Це дозволяє прогнозувати високоризикові ситуації та пропонувати превентивні заходи.
Моніторинг у реальному часі та автоматизоване виявлення небезпек є ще однією важливою сферою застосування ШІ. Це дозволяє ідентифікувати потенційні небезпеки до того, як вони призведуть до нещасних випадків, забезпечуючи швидкі коригувальні дії.
Комп'ютерний зір, зокрема, використовує ШІ-камери, які безперервно сканують робоче середовище для виявлення небезпечної поведінки, наприклад, обхід захисних бар'єрів, неправильне використання обладнання або відсутність ЗІЗ. Вони також здатні виявляти падіння, контролювати безпечні відстані від машин, перевіряти цілісність захисних огорож та належне розміщення знаків безпеки. Носійні пристрої, такі як інтелектуальні шоломи, жилети та браслети, інтегровані з ШІ та IoT, відстежують життєві показники працівників (пульс, температура, рівень втоми), виявляють токсичні гази, моніторять поставу та фізичне навантаження. Ці пристрої можуть попереджати працівників про потенційні ризики для здоров'я в реальному часі.
Інтелектуальні системи також можуть миттєво аналізувати дані та вимикати машини у разі несправності або виявлення небезпеки, мінімізуючи ймовірність нещасних випадків. ШІ може видавати миттєві оповіщення, якщо працівник заходить у заборонену зону або не носить належного спорядження. Крім того, автономні роботи можуть патрулювати робоче місце, шукаючи небезпеки, такі як розливи або несправне обладнання, повідомляючи про проблеми, запускаючи сигнали тривоги або викликаючи допомогу.
Автоматизація та оптимізація процесів за допомогою ШІ значно зменшує вплив людського фактора та підвищує ефективність рутинних завдань, а також покращує ідентифікацію небезпек за допомогою прогнозної аналітики. Це значно зменшує адміністративне навантаження на фахівців з охорони праці, дозволяючи їм зосередитися на більш стратегічних завданнях. Автоматизовані системи працюють безперервно, обробляючи дані з IoT-пристроїв (сенсорів, камер) миттєво. Це зменшує кількість помилок, спричинених людським фактором, оскільки ШІ не відволікається, не втомлюється та не піддається емоціям.[14] ШІ може оптимізувати робочі процеси, запобігати перевтомі та оцінювати ризики перевантаження персоналу. Автоматизація небезпечних завдань за допомогою роботів зменшує ризик для людей, підвищуючи при цьому операційну ефективність та точність.
Ефективний попередній аналіз небезпек з використанням ШІ вимагає інтеграції та взаємодії кількох ШІ-інструментів для отримання повного та точного уявлення про ризики. Збір даних відбувається з різноманітних джерел, включаючи сенсори, пристрої Інтернету речей (IoT), звіти про інциденти та людський ввід. Ці дані потім обробляються за допомогою різних ШІ-технологій, таких як прогнозний аналіз, машинне навчання, комп'ютерний зір та обробка природної мови. Носійні пристрої та автономні роботи будуть фізичними проявами цих ШІ-систем, які збирають дані та виконують дії на основі аналізу.
ШІ,[15] завдяки своїй здатності аналізувати величезні обсяги даних та виявляти складні закономірності, може ідентифікувати ризики, які є "невидимими" для людського ока або не є очевидними з ізольованих інцидентів. Це включає ранні ознаки зносу машин, незначні зміни температури або незвичайні вібрації, які можуть вказувати на потенційні проблеми задовго до того, як вони стануть критичними.
Ця здатність ШІ дозволяє компаніям переходити від реагування на очевидні проблеми до виявлення та усунення прихованих або потенційних загроз, які могли б призвести до серйозних інцидентів у майбутньому. Це не тільки підвищує безпеку, але й може призвести до значної економії коштів за рахунок запобігання дорогим поломкам та простоям.
Ключові переваги використання ШІ для попереднього аналізу небезпек на виробництві.
Використання штучного інтелекту для попереднього аналізу небезпек у виробничому середовищі надає численні переваги, що значно підвищують рівень безпеки та ефективність управління. Однією з найважливіших переваг є значне підвищення точності та ефективності оцінки ризиків. ШІ-системи безперервно аналізують величезні обсяги даних для виявлення ризиків та невідповідностей у реальному часі, що значно зменшує ймовірність людської помилки та підвищує загальні результати безпеки.
На відміну від людей, ШІ не відволікається, не втомлюється та не піддається емоціям, що забезпечує точний моніторинг протоколів безпеки. ШІ-інструменти можуть обробляти інформацію способами, недоступними для людей, що дозволяє їм передбачати потенційні небезпеки до їх виникнення. Прогнозні моделі ШІ, навчаючись на нових даних, постійно вдосконалюють свою здатність виявляти ризики, що робить їх більш ефективними з часом.
ШІ дозволяє організаціям переходити від реактивного до проактивного запобігання інцидентам та травмам. Прогнозний аналіз виявляє закономірності та передбачає потенційні небезпеки, дозволяючи вживати превентивних заходів до того, як вони призведуть до нещасних випадків. Моніторинг у реальному часі дозволяє миттєво виявляти небезпечні умови та поведінку, забезпечуючи швидке втручання та коригувальні дії.
Покращення дотримання нормативних вимог та стандартів безпеки є ще однією значною перевагою. ШІ може автоматизувати моніторинг та звітність, забезпечуючи постійне дотримання стандартів безпеки. Він може відстежувати показники безпеки в реальному часі, виявляти потенційні проблеми з відповідністю та генерувати звіти для регуляторних органів.
ШІ може швидко перевіряти інформацію в інтернеті на наявність новин про оновлення галузевих норм, гарантуючи, що стратегії безпеки базуються на найновіших вимогах. Це зменшує ризик штрафів та санкцій, а також покращує репутацію компанії.
Оптимізація витрат та підвищення загальної продуктивності праці є прямим наслідком впровадження ШІ. Завдяки зменшенню кількості нещасних випадків та травм, ШІ допомагає компаніям економити значні кошти, пов'язані з медичними витратами, компенсаціями працівникам, простоями та юридичними спорами.
Прогнозне технічне обслуговування, що дозволяє замінювати зношені деталі вчасно, допомагає уникнути дорогих ремонтів та простоїв обладнання. ШІ-інструменти можуть оптимізувати робочі процеси, підвищуючи продуктивність, оскільки працівники почуваються безпечніше та менш напружено, що позитивно впливає на їхній психічний стан та загальну атмосферу на робочому місці. Автоматизація рутинних завдань зменшує адміністративне навантаження та вивільняє людські ресурси, дозволяючи фахівцям з охорони праці зосередитися на більш цінних видах діяльності.
Удосконалення програм навчання та підвищення рівня свідомості працівників через інтерактивні симуляції є ще однією вагомою перевагою. ШІ покращує навчання, створюючи імерсивні симуляції та сценарії віртуальної реальності (VR), які відтворюють реальні небезпеки на робочому

місці. Це підвищує засвоєння знань та готує працівників до ефективного реагування в надзвичайних ситуаціях. VR-навчання може призвести до підвищення на 30% рівня засвоєння протоколів безпеки порівняно з традиційними методами у високоризикових галузях. ШІ також може персоналізувати навчальний досвід, адаптуючи рівні складності та сценарії на основі індивідуальної продуктивності та прогалин у знаннях.
Проблеми та та обмеження впровадження штучного інтелекту в систему виробничої безпеки.
Незважаючи на значні переваги, впровадження штучного інтелекту в охороні праці супроводжується низкою проблем та обмежень, які потребують ретельного розгляду.
Етичні питання є одними з найскладніших. Приватність даних викликає серйозне занепокоєння, оскільки ШІ-системи в охороні праці часто покладаються на великі обсяги даних, включаючи особисту інформацію про працівників, записи про безпеку, стан здоров'я та екологічні фактори. Збір, зберігання та обробка цих даних створюють значні проблеми з конфіденційністю та безпекою, вимагаючи суворого дотримання нормативних актів, для уникнення витоків даних, штрафів та судових позовів. Надмірне спостереження за допомогою ШІ-систем, таких як розпізнавання облич та моніторинг поведінки, може призвести до стресу, невдоволенню та юридичних суперечок щодо прав працівників на приватність на робочому місці.
Упередженість алгоритмів є ще однією критичною етичною проблемою. ШІ-системи є настільки неупередженими, наскільки неупередженими є дані, на яких вони навчаються. Якщо навчальні дані містять упередження, пов'язані з статтю, віком чи іншими факторами, ці упередження можуть бути увічнені в процесі прийняття рішень системою. Це може призвести до того, що певні групи працівників будуть непропорційно часто позначатися як порушники правил безпеки або стикатися з дискримінаційними протоколами безпеки. Необхідно впроваджувати активні заходи для пом'якшення упереджень, забезпечуючи справедливість та постійний моніторинг алгоритмів.
Питання відповідальності також є невирішеним. У разі, якщо ШІ-система не зможе виявити небезпеку, зробить неправильний прогноз або надасть неточні рекомендації, що призведе до нещасних випадків, травм або навіть смертельних випадків, виникають важливі питання щодо підзвітності та юридичної відповідальності. Оскільки ШІ-системи певною мірою працюють автономно, встановлення чітких ліній відповідальності є складним завданням.
Технічні та операційні перешкоди також створюють значні виклики. Якість та доступність даних є першочерговими, оскільки ШІ-системи значною мірою покладаються на високоякісні та неупереджені дані для точних прогнозів та рішень. В промислових умовах дані часто фрагментовані, неповні або низької якості, що ускладнює ефективну роботу ШІ. Необхідно встановлювати надійні процеси збору даних, забезпечуючи їхню послідовність, точність та повноту.
Інтеграція з існуючими системами є ще однією перешкодою. Промислові операції часто використовують застарілі системи, які можуть бути несумісними з сучасними ШІ-технологіями. Інтеграція ШІ в ці старі системи без порушення повсякденних операцій може бути складною, вимагаючи поетапного підходу та, можливо, модернізації всієї IT-архітектури.
Вартість впровадження ШІ також є значним бар'єром, особливо для малих та середніх підприємств. Витрати включають програмне забезпечення, апаратне забезпечення (сенсори, носійні пристрої), інтеграцію з існуючими системами та навчання персоналу. Початкові витрати на розробку моделей машинного навчання можуть починатися від 50 000 доларів США, а складніші системи можуть коштувати сотні тисяч доларів.
Людський фактор відіграє важливу роль у успішному впровадженні ШІ. Опір змінам з боку працівників, які побоюються втрати робочих місць або не повністю розуміють, як ШІ може покращити їхні ролі, є поширеним явищем. Працівники також можуть бути скептично налаштовані щодо надійності та прозорості ШІ-систем безпеки, особливо у високоризикових середовищах. Необхідно залучати працівників до процесу на ранніх етапах, чітко роз'яснювати роль ШІ у підвищенні безпеки, а не у заміщенні робочих місць.
Важливо знайти баланс між автоматизацією та людським наглядом. Хоча ШІ може підвищити безпеку, він не повинен повністю замінювати людське судження або нагляд. Існує ризик надмірної залежності від технологій, що може призвести до ігнорування проблем, які виходять за межі алгоритму. ШІ повинен підтримувати, а не замінювати роль фахівців з охорони праці, забезпечуючи людський контроль у прийнятті важливих рішень з безпеки.
Нормативно-правова невизначеність також є значним викликом. Відсутність всеосяжних регуляторних рамок для ШІ у сфері охороні праці призводить до прогалин у нагляді та непослідовних практик. Існуючі нормативні акти можуть бути недостатніми для вирішення юридичних наслідків ШІ-систем, особливо коли ШІ бере на себе більше ролей у прийнятті рішень. Необхідні чіткі стандарти щодо людського нагляду, прозорості систем та процедур аудиту. Регуляторні органи, вже вимагають від роботодавців оцінювати ризики, пов'язані з ШІ, та впроваджувати ефективні заходи контролю. Однак, загальний вплив ШІ на регулювання та стандарти безпеки ще не повністю реалізований.
Перспективи розвитку штучного інтелекту у системі забезпечення виробничої безпеки.
Перспективи розвитку штучного інтелекту для забезпечення виробничої безпеки є багатообіцяючими. Майбутнє управління ризиками в системе охороні праці передбачає ще глибшу інтеграцію ШІ з передовими технологіями, такими як цифрові двійники, квантові обчислення та блокчейн. Цифрові двійники, дозволяють симулювати сценарії "що-якщо", візуалізувати потенційні ризики та оптимізувати заходи безпеки в контрольованому середовищі. Квантові обчислення зможуть значно прискорити та розширити можливості моделювання ризиків, дозволяючи обробляти безпрецедентні обсяги даних для більш точних прогнозів. Поєднання ШІ з технологіями блокчейн може підвищити прозорість та безпеку практик управління ризиками, забезпечуючи незмінність та достовірність даних про безпеку.
Подальший розвиток прогнозної аналітики та рецептивних рішень є ключовим напрямком. ШІ вже виходить за рамки простого прогнозування, пропонуючи оптимальні дії для запобігання або пом'якшення ризиків. Прогнозний аналіз, що використовує алгоритми ШІ для прогнозування ризиків, стає все більш точним, дозволяючи організаціям адаптувати свої професійні стандарти на основі статистичних ймовірностей, а не лише інтуїції. Це означає, що ШІ не тільки вказуватиме на потенційні проблеми, але й пропонуватиме конкретні, обґрунтовані дії для їх вирішення, оптимізуючи ресурси та впроваджуючи проактивні рішення для мінімізації загроз.
Фахівці з охорони праці замість того, щоб бути переважно контролюючими органами, будуть переорієнтовані на стратегічне управління, аналіз складних даних та розробку інноваційних рішень. ШІ автоматизуватиме рутинні завдання, звільняючи фахівців для зосередження на більш цінних видах діяльності, таких як розробка програм безпеки, навчання та підвищення рівня свідомості працівників. Це вимагатиме від фахівців з охорони праці розвитку нових навичок у галузі аналізу даних, управління ШІ-системами та міждисциплінарної співпраці.
Впровадження ШІ не лише підвищує рівень безпеки праці та зменшує кількість нещасних випадків, але й оптимізує витрати, покращує дотримання нормативних вимог та підвищує загальну продуктивність праці, а інтерактивні симуляції та персоналізовані навчальні програми значно покращують підготовку працівників.

4.2 Вплив ризиків пов'язаних з переробками на здоров'я і продуктивність праці фахівців-комп'ютерників.

Переробка визначається як постійна робота за межами стандартних або здорових лімітів, що часто призводить до фізичного виснаження, психічного вигорання та загального зниження добробуту. Вона характеризується тривалими робочими годинами, нездатністю відключитися від роботи, роботою під час перерв та відчуттям тиску, як зовнішнього з боку керівництва, так і внутрішнього через занепокоєння щодо втрати роботи або надмірну пристрасть до кар'єрного зростання. Важливо розуміти, що переробка це не просто збільшення кількості відпрацьованих годин, а скоріше неадекватна інтенсивність та нестійкість робочого режиму відносно фізичних та емоційних можливостей людини.
Переробки в IT-сфері, тобто робота понаднормово, є поширеним явищем, і хоча іноді вони здаються неминучими для досягнення цілей проекту, їхній вплив на здоров'я та продуктивність IT-фахівців може бути руйнівним.
Поширеність переробок в IT-індустрії є не просто статистичним фактом, а відображенням глибоко вкоріненої проблеми, яка часто залишається непоміченою. Той факт, що значна частина понаднормової роботи є неоплачуваною, ускладнює її відстеження та вирішення, оскільки вона не фіксується офіційно. Крім того, працівники можуть відчувати внутрішній тиск, щоб "довести свою цінність" або ж бути настільки "захопленими" своєю кар'єрою, що ігнорують негативні наслідки переробок.
Переробки в IT-індустрії є результатом складної взаємодії галузевих особливостей, недоліків управління проєктами, а також організаційної культури та керівництва.
Основними з них є.
Дефіцит кадрів та високий попит. Технологічна галузь стикається зі значною нестачею талантів, де попит на кваліфікованих фахівців часто перевищує пропозицію. Коли не вистачає ресурсів, компанії часто не наймають додаткових людей, а замість цього тиснуть на існуючих працівників, щоб вони працювали більше.
Швидкий темп технологічних змін та вимоги до безперервного навчання. Невпинна еволюція технологій, включаючи штучний інтелект та автоматизацію, вимагає постійного підвищення кваліфікації та адаптації. Ця вимога до безперервного навчання додає значного тиску на працівників, які вже керують існуючим навантаженням, сприяючи "нескінченному циклу стресу та виснаження".
Середовище високого тиску. Внутрішній імпульс технологічної галузі до інновацій та прагнення до "наступних великих проривів" створюють середовища, де від працівників очікується, що вони будуть надавати інновації та результати зі швидкістю, яка посилює рівень стресу. Це сприяє високій поширеності проблем психічного здоров'я: понад половина (52%) технічних працівників відчувають депресію або тривогу.
"Культура поспіху" та прославлення переробок. Поширена культурна норма, яка прославляє довгі години роботи, постійну продуктивність та жертвування турботою про себе як ознаки честі та шляхи до успіху. Ця ідеологія "токсичної продуктивності", часто призводить до вигорання та невдоволення, а не до бажаного успіху.
Вимоги клієнтів щодо швидкості та доступності. Зовнішній тиск з боку клієнтів, які очікують швидкої доставки та постійної доступності, значно сприяє тривалим робочим годинам. Фірми все частіше просувають свою "доступність" та "швидкість", перекладаючи цей тиск на IT-команди.
Жорсткі терміни. Постійний тиск, пов'язаний з дотриманням агресивних та часто нереалістичних термінів, є основним фактором, що сприяє переробкам. Жорсткі терміни постійно призводять до підвищеного стресу, вигорання, компромісної якості та збільшення кількості помилок.
Нереалістичні очікування та недостатні ресурси. Роботодавці часто встановлюють високі очікування, обмежуючи при цьому доступні ресурси, створюючи тривожний дисбаланс, який безпосередньо сприяє надмірному навантаженню.
Вплив переробок на здоров'я IT-фахівців
Переробки мають глибокі та руйнівні наслідки для психічного та фізичного здоров'я IT-фахівців, часто призводячи до хронічних станів, які можуть мати довгострокові наслідки.
1. Наслідки для психічного здоров'я
Вигорання: Це стан хронічного фізичного та емоційного виснаження, спричинений тривалим стресом, пов'язаним з роботою. Симптоми включають відчуття втоми, цинізм, відстороненість від роботи, знижену мотивацію, дратівливість та переважне відчуття емоційного та фізичного виснаження. Вигорання значно збільшує ймовірність того, що працівники шукатимуть нову роботу (у 2,6 рази частіше).
Стрес, тривога та депресія: IT-фахівці постійно повідомляють про вищий рівень стресу та тривоги через тривалі робочі години та постійний тиск, щоб працювати та впроваджувати інновації. Переробка значно збільшує ризик розвитку клінічної депресії та генералізованих тривожних розладів.
Когнітивні порушення: Надмірна робота без достатнього відпочинку серйозно погіршує когнітивні функції, що призводить до труднощів з концентрацією, проблем з пам'яттю (навіть дрібних деталей, таких як імена або дати), зниження здатності приймати рішення та збільшення кількості помилок. Це може проявлятися як "туман у мозку" через підвищений рівень кортизолу.
Розлади сну та безсоння: Переробки часто призводять до нерегулярного режиму сну, труднощів із засинанням або підтриманням сну через нав'язливі думки, стрес, пов'язаний з роботою, або надмірний вплив екранів. Хронічне недосипання ще більше посилює стрес, психічне виснаження та погіршує настрій та концентрацію.
2. Наслідки для фізичного здоров'я
Серцево-судинні ризики: Тривалий стрес та переробки значно підвищують ризик високого кров'яного тиску , серцевих захворювань та інсульту. Дослідження показують, що навіть робота лише 3-4 години понаднормово на день збільшує ризик серцевих проблем на 60%. Робота 55 годин або більше на тиждень пов'язана з 35% підвищеним ризиком інсульту та 17% підвищеним ризиком смерті від ішемічної хвороби серця.
Проблеми опорно-рухового апарату (ергономічні ризики): Сидячий характер роботи в IT, у поєднанні з тривалими годинами, неправильною поставою (наприклад, сутулість над ноутбуками) та повторюваними рухами, призводить до травм від повторюваних навантажень (RSI), таких як синдром зап'ястного каналу (біль у зап'ясті/руці, оніміння, поколювання) та тендиніт (запалення, часто в зап'ястях, ліктях або плечах). Хронічний біль у спині та шиї також є поширеним явищем через тривале сидіння на непідтримуючих стільцях або витягування шиї до неправильно розташованих моніторів. Ці початкові болі можуть перерости в хронічний, виснажливий біль.
Шлунково-кишкові проблеми та ослаблена імунна система: Хронічний стрес від переробок може проявлятися фізично, призводячи до шлунково-кишкових проблем та зниження функції імунної системи, що робить людей більш сприйнятливими до хвороб та "загострень" існуючих станів.
Використання перспективних комп'ютерних технологій у сфері охорони праці та організація раціонального режиму робочого дня спеціалістів дозволять зберегти їх здоров'я та високу продуктивність протягом усього періоду їхньої трудової діяльності

ВИСНОВКИ

У межах виконання науково-дослідної роботи розв'язано актуальну науково-практичну задачу забезпечення конфіденційної передачі інформації шляхом використання файлів формату PDF як стеганографічних контейнерів. Під час дослідження проведено ретроспективний аналіз еволюції стеганографічних методів та еволюції типів носіїв прихованих даних.

Результати аналізу сучасних цифрових стеганографічних контейнерів свідчать, що формат PDF має високий потенціал щодо інформаційної місткості та варіативності методів інтеграції прихованих повідомлень. Інтенсивний документообіг у форматі PDF між користувачами корпоративних і приватних мереж робить цей тип файлів поширеним та легітимним елементом транзиту даних через канали електронної пошти та сучасні месенджери. На відміну від масової передачі графічних об'єктів, транспортування значної кількості PDF-документів є типовим процесом і не викликає підозр з боку систем моніторингу мережевого трафіку (DPI тощо).

Додатковою перевагою обраного типу контейнера є його стійкість до деструктивних впливів під час транспортування. На відміну від растрових зображень, які в месенджерах часто піддаються процедурам стиснення з втратами (компресії), що повністю руйнує вбудовану стеганографічну інформацію, файли формату PDF зберігають свою бінарну та об'єктну структуру незмінною.

Розроблений програмний комплекс успішно реалізує функціонал використання PDF-документів як стеганографічних контейнерів та забезпечує виконання таких завдань:

Гнучкість локалізації даних: реалізовано можливість інтеграції прихованих повідомлень як у структуру об'єктів коментарів, так і в спеціалізований розділ метаданих документа.

Стійкість контейнера: забезпечено високу інформаційну місткість та інваріантність стеганографічного вмісту до можливих модифікацій структури файлу на рівні операційної системи.

Оптимізація обсягу даних: з метою запобігання аномальному збільшенню лінійного розміру PDF-файлу після стеганографічного перетворення впроваджено штучне обмеження максимального обсягу вбудованих даних на рівні 15% від первинного розміру файлу. Цей захід мінімізує ризик виявлення стегоконтейнера за допомогою статистичного аналізу його фізичних параметрів.

Контроль цілісності: інтегровано підсистему верифікації на основі обчислення та порівняння криптографічних хеш-рядків, що дозволяє оперативно виявляти факти спотворення або модифікації прихованої інформації.

Комбінований криптографічний захист: для підвищення рівня безпеки повідомлень та супутніх дайджестів застосовано диференційований підхід із використанням симетричних та асиметричних алгоритмів шифрування, що суттєво підвищує загальну стійкість системи до компрометації.

Запропоноване програмне рішення має високий потенціал для практичного впровадження з метою захисту чутливих даних у мілітарній сфері, органах державної влади, на стратегічних підприємствах, а також у межах забезпечення внутрішньої інформаційної безпеки комерційних компаній та корпорацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. https://uk.wikipedia.org/wiki/Стеганографія#:~:text=**Стеганогра́фія**%20-%20(з%20грец.,приховує%20сам%20факт%20існування%20повідомлення.
2. Даус, Ю. 2026. Використання pdf файлів в якості стеганографічних контейнерів. Вісник Одеського національного морського університету. 78 (Січ 2026), 226-240. DOI:https://doi.org/10.47049/2226-1893-2025-4-226-240.
3. Григор'єв, С. В. (2009). Сучасні методи стеганографії та їх застосування. Науковий вісник Херсонського державного університету. Серія: Економічні науки, 3, 175-178.
4. Полевой, С. А., & Григорьев, С. В. (2014). Стеганографія у цифровому просторі: сучасний стан та перспективи розвитку. Інформаційні технології та комп’ютерна інженерія, 1(60), 68-76.
5. Кібкан, В. В. (2011). Стеганографія як метод забезпечення конфіденційності інформації. Інформаційні технології в освіті, науці та виробництві, 5, 38-42.
6. Соловйов, В. І., & Шамраєв, В. Г. (2008). Сучасні аспекти стеганографії. Збірник наукових праць Херсонського державного університету. Серія: Економічні науки, 5, 103-107.
7. Чепур, Л. А., & Сторожук, В. В. (2013). Аналіз та вибір методів стеганографії. Системні дослідження та інформаційні технології, 3, 114-124.
8. Сахно, В. В., & Старцев, В. О. (2015). Методи стеганографії та їх використання для захисту інформації. Збірник наукових праць Харківського національного університету внутрішніх справ, 2(47), 212-220.
9. Яцишин, А. В., & Лунін, С. В. (2012). Сучасні технології стеганографії. Проблеми телекомунікацій, 1, 92-101.
10. Кондратюк, В. В., & Маркова, І. С. (2017). Стеганографія у сучасному інформаційному просторі. Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Радіотехніка, радіоапаратобудування, 1(71), 37-43.
11. Ковальчук, І. В., & Бондар, С. А. (2016). Стеганографічні методи захисту інформації у цифрових зображеннях. Молодь і ринок, 4(142), 57-61.
12. Шинкарук, Л. В., & Стець, М. Л. (2011). Застосування стеганографії для захисту інформації в інформаційно-комунікаційних системах. Вісник Львівського університету. Серія: Інформаційні системи та мережі, 732, 31-38.
13. Горбатенко, О. В., & Клочко, І. О. (2014). Стеганографія в цифрових зображеннях: аналіз методів та застосування. Збірник наукових праць Донецького національного технічного університету, 33, 191-198.
14. Мельник, О. О. (2016). Сучасні аспекти стеганографії та її використання в інформаційних системах. Інформаційні технології та комп’ютерна інженерія, 2(76), 35-40.
15. Петренко, С. І. (2018). Застосування стеганографії для захисту інформації в обчислювальних системах. Теорія і практика сучасних інформаційних технологій, 1, 76-81.
16. Кравченко, О. М. (2015). Стеганографія як інструмент забезпечення безпеки інформації. Системні дослідження та інформаційні технології, 2, 129-134.
17. Куц, О. В., & Бурлака, В. І. (2013). Використання методів стеганографії для забезпечення безпеки інформації. Наукові праці Кам'янець-Подільського національного університету імені Івана Огієнка. Серія: Інформаційні технології та телекомунікації, 1(21), 29-33.
18. Смолярчук, І. О. (2017). Стеганографія: основні принципи та сучасні технології. Науковий вісник Національного університету біоресурсів і природокористування України. Серія: Технічні науки, 270, 33-39.
19. Тарасенко, О. Є., & Гетманець, В. В. (2019). Використання стеганографії для прихованої передачі інформації. Вісник Кіровоградського національного технічного університету. Серія: Технічні науки, 46, 72-78.
20. Шевченко, С. В., & Підгайний, І. В. (2016). Застосування методів стеганографії для забезпечення безпеки інформації. Технічні науки та технології, 1(8), 127-133.
21. Кравець, М. С. (2014). Стеганографія: сучасні тенденції розвитку. Інформаційні технології та комп'ютерна інженерія, 3(75), 41-47.
22. Левченко, В. В. (2015). Використання стеганографії для забезпечення конфіденційності даних. Науковий вісник Хмельницького національного університету. Серія: Економічні науки, 2, 123-127.

23. Коровяков, А. А. (2017). Стеганографія в області захисту інформації: аналіз та практичне застосування. Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі, 880, 183-188.
24. Беспалов, А. І., & Пилипенко, В. Ю. (2016). Застосування методів стеганографії для забезпечення безпеки інформаційних ресурсів. Науковий вісник Ужгородського університету. Серія: Математика та інформатика, 1(45), 181-186.
25. Klemm R., Chen B. Hiding Sensitive Information Using PDF Steganography. arXiv preprint arXiv:2405.00865, 2024. Документ доступний: https://arxiv.org/pdf/2405.00865 та: https://arxiv.org/html/2405.00865
26. Ndoundam R., Ekodeck S. G. R. PDF Steganography based on Chinese Remainder Theorem. arXiv preprint arXiv:1506.01256, 2015. PDF доступний: https://arxiv.org/pdf/1506.01256: arXiv
27. Koptyra P., Ogiela L. Distributed Steganography in PDF Files - Secrets Hidden in Modified Pages. PMC (PubMed Central). 2021. Доступ: https://pmc.ncbi.nlm.nih.gov/articles/PMC7517136/ PMC+2ResearchGate+2
28. Vulnerability Exploitations Using Steganography in PDF Files. International Journal of Computer Networks and Applications (IJCNA). Доступний PDF: https://www.ijcna.org/Manuscripts/IJCNA-2020-O-02.pdf: ijcna.org
29. A new method for pdf steganography in justified texts. ScienceDirect. Доступ: https://www.sciencedirect.com/science/article/abs/pii/S221421261830485X: ScienceDirect
30. Maiorca D., Biggio B. Digital Investigation of PDF Files: Unveiling Traces of Embedded Malware. arXiv preprint arXiv:1707.05102, 2017. Доступ: https://a

ДОДАТОК А

```
package com.ds;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.pdmodel.PDDocumentInformation;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.*; import java.awt.*; import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.*;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.Arrays;
import java.util.Base64;

public class PdfStegoExt extends JFrame {

    // ── UI поля ──────────────────────────────────────────────────────────
    private JTextArea leftArea, rightArea;

    // Криптографія
    private JRadioButton aesRadio, rsaRadio;

    // Місце розміщення прихованих даних
    private JRadioButton commentRadio, metaRadio;

    private JLabel statusLabel;
    private JLabel fileNameLabel;

    // ── Стан ─────────────────────────────────────────────────────────────
    private byte[] rawPdf;
    private int maxCapacity = 0;
    /** Поточний активний файл на диску (оновлюється після кожного збереження) */
    private File currentFile = null;

    // ── Константи ────────────────────────────────────────────────────────
    /** Ключ в Info Dictionary (режим MetaData) */
    private final String KEY_NAME      = "Metadata_Revision";
    /** Маркер прихованого коментаря всередині PDF (режим Comment) */
    private final String COMMENT_MARKER = "%%%";
    /** Маркер хешу в хвості файлу */
    private final String HASH_MARKER   = "%%%%";

    // ── Конструктор ──────────────────────────────────────────────────────
    public PdfStegoExt() {
        initUI();
    }

    // ── Побудова інтерфейсу ──────────────────────────────────────────────
    private void initUI() {
        setTitle("PdfStegoExt - Security Stego Edition ONMU 2026");
```

```
setSize(1150, 820);
setDefaultCloseOperation(EXIT_ON_CLOSE);

// ═══ МЕНЮ ═══════════════════════════════════════════════════════════════
JMenuBar mb = new JMenuBar();

// --- Файл ---
JMenu fileMenu = new JMenu("Файл");
JMenuItem load    = new JMenuItem("Завантажити PDF");
JMenuItem save    = new JMenuItem("Зберегти як (Приховати текст)");
JMenuItem extract = new JMenuItem("Витягти прихований текст");
JMenuItem fullClean = new JMenuItem("Повна очистка файлу (Анонімізація)");

load.addActionListener(e -&gt; loadPdf());
save.addActionListener(e -&gt; handleStegano(true));
extract.addActionListener(e -&gt; handleStegano(false));
fullClean.addActionListener(e -&gt; performFullClean());

fileMenu.add(load); fileMenu.add(save); fileMenu.add(extract);
fileMenu.addSeparator();
fileMenu.add(fullClean);

// --- Хешування ---
JMenu hashMenu = new JMenu("Хешування");
JMenuItem embedHash  = new JMenuItem("Вбудувати Хеш (Захист)");
JMenuItem verifyHash = new JMenuItem("Перевірити Хеш (Цілісність)");
embedHash.addActionListener(e -&gt; handleHash(true));
verifyHash.addActionListener(e -&gt; handleHash(false));
hashMenu.add(embedHash); hashMenu.add(verifyHash);

// --- Криптографія ---
JMenu cryptoMenu = new JMenu("Криптографія");
JMenuItem genKeys = new JMenuItem("Згенерувати ключі RSA");
genKeys.addActionListener(e -&gt; generateRSAKeys());
cryptoMenu.add(genKeys);

mb.add(fileMenu); mb.add(hashMenu); mb.add(cryptoMenu);

// ═══ ПАНЕЛЬ АЛГОРИТМУ ШИФРУВАННЯ ═══════════════════════════════════════
JPanel cryptoPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 6, 2));
cryptoPanel.setBorder(BorderFactory.createTitledBorder("Алгоритм шифрування"));
aesRadio = new JRadioButton("AES (Пароль)", true);
rsaRadio = new JRadioButton("RSA (Ключі)");
ButtonGroup cryptoGroup = new ButtonGroup();
cryptoGroup.add(aesRadio); cryptoGroup.add(rsaRadio);
cryptoPanel.add(aesRadio); cryptoPanel.add(rsaRadio);
mb.add(cryptoPanel);

// ═══ ПАНЕЛЬ МІСЦЯ ЗБЕРІГАННЯ ═══════════════════════════════════════════
JPanel storePanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 6, 2));
storePanel.setBorder(BorderFactory.createTitledBorder("Місце зберігання"));
commentRadio = new JRadioButton("Comment ", true);   // за замовчуванням
metaRadio    = new JRadioButton("MetaData");
ButtonGroup storeGroup = new ButtonGroup();
storeGroup.add(commentRadio); storeGroup.add(metaRadio);
storePanel.add(commentRadio); storePanel.add(metaRadio);
mb.add(storePanel);

setJMenuBar(mb);

// ═══ ЦЕНТРАЛЬНА ПАНЕЛЬ ═══════════════════════════════════════════════════
JPanel mainPanel = new JPanel(new GridLayout(1, 2, 10, 0));

JPanel leftPanel = new JPanel(new BorderLayout());
leftArea = createArea("Текст для вбудовування", true);
JButton clearLeft = new JButton("Очистити ліве вікно");
clearLeft.addActionListener(e -&gt; leftArea.setText(""));
```

```
    leftPanel.add(new JScrollPane(leftArea), BorderLayout.CENTER);
    leftPanel.add(clearLeft, BorderLayout.SOUTH);

    JPanel rightPanel = new JPanel(new BorderLayout());
    rightArea = createArea("Результат перевірки / Витягнутий текст", false);
    JButton clearRight = new JButton("Очистити праве вікно");
    clearRight.addActionListener(e -&gt; {
        rightArea.setText("");
        rightArea.setBackground(Color.WHITE);
    });
    rightPanel.add(new JScrollPane(rightArea), BorderLayout.CENTER);
    rightPanel.add(clearRight, BorderLayout.SOUTH);

    mainPanel.add(leftPanel);
    mainPanel.add(rightPanel);

    // ═══ НИЖНЯ ПАНЕЛЬ СТАТУСУ ═══════════════════════════════════════════
    JPanel footerPanel = new JPanel(new BorderLayout());
    footerPanel.setBorder(BorderFactory.createLoweredBevelBorder());

    statusLabel = new JLabel("Ємність: 0 байт");
    statusLabel.setBorder(BorderFactory.createEmptyBorder(2, 10, 2, 10));

    fileNameLabel = new JLabel("Файл не завантажений");
    fileNameLabel.setForeground(Color.RED);
    fileNameLabel.setBorder(BorderFactory.createEmptyBorder(2, 10, 2, 10));

    footerPanel.add(statusLabel,  BorderLayout.WEST);
    footerPanel.add(fileNameLabel, BorderLayout.EAST);

    add(mainPanel,   BorderLayout.CENTER);
    add(footerPanel, BorderLayout.SOUTH);
}

// ── Допоміжник: текстова область ──────────────────────────────────────
private JTextArea createArea(String title, boolean editable) {
    JTextArea a = new JTextArea();
    a.setLineWrap(true); a.setWrapStyleWord(true);
    a.setEditable(editable);
    a.setBorder(BorderFactory.createTitledBorder(title));
    a.setFont(new Font("Monospaced", Font.PLAIN, 13));
    return a;
}

// ═══════════════════════════════════════════════════════════════════════
//  ЗАВАНТАЖЕННЯ PDF
// ═══════════════════════════════════════════════════════════════════════
private void loadPdf() {
    JFileChooser fc = new JFileChooser();
    if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
        try {
            File file = fc.getSelectedFile();
            rawPdf = java.nio.file.Files.readAllBytes(file.toPath());
            currentFile = file;
            maxCapacity = (int) (rawPdf.length * 0.15);
            rightArea.setBackground(Color.WHITE);
            statusLabel.setText("Доступна ємність: " + maxCapacity + " байт");
            fileNameLabel.setText("Поточний файл: " + file.getName());
            fileNameLabel.setForeground(new Color(0, 102, 0));
            JOptionPane.showMessageDialog(this, "PDF успішно завантажено.");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Помилка при читанні файлу!");
        }
    }
}

// ═══════════════════════════════════════════════════════════════════════
```

```
//  СТЕГАНОГРАФІЯ - диспетчер
// ═══════════════════════════════════════════════════════════════════════════
private void handleStegano(boolean isEmbed) {
    if (rawPdf == null) {
        JOptionPane.showMessageDialog(this, "Файл не завантажений!", "Увага",
                JOptionPane.WARNING_MESSAGE);
        return;
    }
    try {
        if (isEmbed) embedMessage();
        else         extractMessage();
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Помилка стеганографії: " + ex.getMessage());
    }
}

// ── Вбудування повідомлення ──────────────────────────────────────────────
private void embedMessage() throws Exception {
    byte[] msg = leftArea.getText().getBytes(StandardCharsets.UTF_8);
    if (msg.length &gt; maxCapacity)
        throw new Exception("Текст перевищує ліміт ємності (15%)!");

    byte[] encrypted = encryptData(msg);
    if (encrypted == null) return;

    String encodedPayload = Base64.getEncoder().encodeToString(encrypted);

    if (metaRadio.isSelected()) {
        // ── Режим MetaData: пишемо в Info Dictionary ──────────────────
        embedInMetadata(encodedPayload);
    } else {
        // ── Режим Comment: вставляємо %%% всередині PDF-потоку ────────
        embedInComment(encodedPayload);
    }
}

/** Запис у поле Metadata_Revision Info Dictionary */
private void embedInMetadata(String encodedPayload) throws Exception {
    try (PDDocument doc = PDDocument.load(rawPdf)) {
        doc.getDocumentInformation().setCustomMetadataValue(KEY_NAME, encodedPayload);
        JFileChooser fc = new JFileChooser();
        if (fc.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
            File dest = fc.getSelectedFile();
            doc.save(dest);
            // Оновлюємо поточний файл у пам'яті
            rawPdf = java.nio.file.Files.readAllBytes(dest.toPath());
            currentFile = dest;
            maxCapacity = (int) (rawPdf.length * 0.15);
            statusLabel.setText("Доступна ємність: " + maxCapacity + " байт");
            fileNameLabel.setText("Поточний файл: " + dest.getName());
            JOptionPane.showMessageDialog(this,
                    "Дані вбудовано в MetaData (Metadata_Revision).\n" +
                            "Поточний файл: " + dest.getName());
        }
    }
}

/**
 * Запис маркера %%% + payload в байти PDF.
 * Маркер вставляється перед %%EOF - після основного тіла документа,
 * але всередині файлу (до будь-якого зовнішнього HASH_MARKER).
 * PDF-переглядачі ігнорують дані після %%EOF.
 */
private void embedInComment(String encodedPayload) throws Exception {
    String raw = new String(rawPdf, StandardCharsets.ISO_8859_1);
    int eofIdx = raw.lastIndexOf("%%EOF");
```

```
    byte[] commentBytes = (COMMENT_MARKER + encodedPayload)
            .getBytes(StandardCharsets.UTF_8);

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    if (eofIdx != -1) {
        baos.write(rawPdf, 0, eofIdx);
        baos.write('\n');
        baos.write(commentBytes);
        baos.write('\n');
        baos.write(rawPdf, eofIdx, rawPdf.length - eofIdx);
    } else {
        baos.write(rawPdf);
        baos.write('\n');
        baos.write(commentBytes);
    }

    byte[] result = baos.toByteArray();

    JFileChooser fc = new JFileChooser();
    if (fc.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
        File dest = fc.getSelectedFile();
        java.nio.file.Files.write(dest.toPath(), result);
        // Оновлюємо поточний файл у пам'яті
        rawPdf = result;
        currentFile = dest;
        maxCapacity = (int) (rawPdf.length * 0.15);
        statusLabel.setText("Доступна ємність: " + maxCapacity + " байт");
        fileNameLabel.setText("Поточний файл: " + dest.getName());
        JOptionPane.showMessageDialog(this,
                "Дані вбудовано як Comment  у тіло PDF.\n" +
                        "Поточний файл: " + dest.getName());
    }
}

// ── Витягання повідомлення ──────────────────────────────────────────────
private void extractMessage() throws Exception {
    if (metaRadio.isSelected()) {
        extractFromMetadata();
    } else {
        extractFromComment();
    }
}

/** Витягання з Info Dictionary поточного файлу */
private void extractFromMetadata() throws Exception {
    if (rawPdf == null) throw new Exception("Файл не завантажений!");
    try (PDDocument doc = PDDocument.load(rawPdf)) {
        String encoded = doc.getDocumentInformation()
                .getCustomMetadataValue(KEY_NAME);
        if (encoded == null)
            throw new Exception("В MetaData немає прихованого тексту.");
        byte[] decrypted = decryptData(Base64.getDecoder().decode(encoded));
        if (decrypted != null) {
            rightArea.setText(new String(decrypted, StandardCharsets.UTF_8));
            rightArea.setBackground(new Color(230, 240, 255));
        }
    }
}

/** Витягання з Comment-маркера %%% поточного файлу */
private void extractFromComment() throws Exception {
    if (rawPdf == null) throw new Exception("Файл не завантажений!");
    String content = new String(rawPdf, StandardCharsets.ISO_8859_1);

    int idx = findCommentMarker(content);
    if (idx == -1)
        throw new Exception("Маркер не знайдено. Файл не містить стеганографії.");
```

```
    int start = idx + COMMENT_MARKER.length();
    int end = content.indexOf('\n', start);
    if (end == -1) end = content.length();
    String encoded = content.substring(start, end).trim();

    if (encoded.contains("%%EOF"))
        encoded = encoded.substring(0, encoded.indexOf("%%EOF")).trim();

    byte[] decrypted = decryptData(Base64.getDecoder().decode(encoded));
    if (decrypted != null) {
        rightArea.setText(new String(decrypted, StandardCharsets.UTF_8));
        rightArea.setBackground(new Color(230, 240, 255));
    }
}

/**
 * Знаходить позицію маркера %%% що НЕ є частиною %%%% (маркер хешу).
 * Перевіряє що:
 *  - символ ПІСЛЯ %%% не є '%'  (інакше це початок %%%%)
 *  - символ ПЕРЕД %%% не є '%'  (інакше це позиція 1..3 всередині %%%%)
 */
private int findCommentMarker(String content) {
    int searchFrom = content.length();
    while (true) {
        int idx = content.lastIndexOf(COMMENT_MARKER, searchFrom - 1);
        if (idx == -1) return -1;

        // Перевірка справа: наступний символ не '%' → не початок %%%%
        boolean nextIsPercent = (idx + COMMENT_MARKER.length() &lt; content.length())
                &amp;&amp; content.charAt(idx + COMMENT_MARKER.length()) == '%';

        // Перевірка зліва: попередній символ не '%' → не всередині %%%%
        boolean prevIsPercent = (idx &gt; 0)
                &amp;&amp; content.charAt(idx - 1) == '%';

        if (nextIsPercent || prevIsPercent) {
            searchFrom = idx;
            continue;
        }
        return idx;
    }
}

// ═══════════════════════════════════════════════════════════════════════════
//  ХЕШУВАННЯ
// ═══════════════════════════════════════════════════════════════════════════
private void handleHash(boolean isEmbed) {
    if (rawPdf == null) {
        JOptionPane.showMessageDialog(this, "Файл не завантажений!", "Увага",
                JOptionPane.WARNING_MESSAGE);
        return;
    }
    try {
        if (isEmbed) embedFileHash();
        else         verifyFileHash();
    } catch (Exception ex) {
        rightArea.setBackground(new Color(241, 28, 28));
        JOptionPane.showMessageDialog(this, "Помилка хешування: " + ex.getMessage());
    }
}

private void embedFileHash() throws Exception {
    if (currentFile == null)
        throw new Exception("Спочатку збережіть файл через «Зберегти як»!");

    MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

```
    String hashStr = Base64.getEncoder().encodeToString(digest.digest(rawPdf));
    byte[] encryptedHash = encryptData(hashStr.getBytes(StandardCharsets.UTF_8));
    if (encryptedHash == null) return;

    // Дописуємо хеш після поточного вмісту і зберігаємо в той самий файл
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    baos.write(rawPdf);
    baos.write(HASH_MARKER.getBytes(StandardCharsets.UTF_8));
    baos.write(Base64.getEncoder().encodeToString(encryptedHash)
            .getBytes(StandardCharsets.UTF_8));
    byte[] withHash = baos.toByteArray();

    java.nio.file.Files.write(currentFile.toPath(), withHash);
    rawPdf = withHash;
    maxCapacity = (int) (rawPdf.length * 0.15);
    statusLabel.setText("Доступна ємність: " + maxCapacity + " байт");

    JOptionPane.showMessageDialog(this,
            "Хеш вбудовано в кінець файлу.\nФайл: " + currentFile.getName());
}

private void verifyFileHash() throws Exception {
    String content = new String(rawPdf, StandardCharsets.ISO_8859_1);
    int markerIdx = content.lastIndexOf(HASH_MARKER);
    if (markerIdx == -1) {
        rightArea.setBackground(new Color(255, 200, 200));
        throw new Exception("Маркер хешування не знайдено!");
    }

    byte[] dataBefore = Arrays.copyOfRange(rawPdf, 0, markerIdx);
    String calculatedHash = Base64.getEncoder().encodeToString(
            MessageDigest.getInstance("SHA-256").digest(dataBefore));

    byte[] encryptedHashFromFile = Base64.getDecoder().decode(
            content.substring(markerIdx + HASH_MARKER.length()).trim());
    byte[] decryptedHash = decryptData(encryptedHashFromFile);
    if (decryptedHash == null) return;

    String originalHash = new String(decryptedHash, StandardCharsets.UTF_8);
    if (calculatedHash.equals(originalHash)) {
        rightArea.setBackground(new Color(8, 214, 8));
        rightArea.setText("ПЕРЕВІРКА УСПІШНА:\nФайл цілісний.\n\nHash: " + originalHash);
    } else {
        rightArea.setBackground(new Color(124, 5, 5));
        rightArea.setText("УВАГА: ЦІЛІСНІСТЬ ПОРУШЕНО!");
    }
}

// ═══════════════════════════════════════════════════════════════════════
//  ПОВНА ОЧИСТКА (АНОНІМІЗАЦІЯ)
//  Видаляє: MetaData поле, Comment-маркер %%%, хеш-маркер %%%%
// ═══════════════════════════════════════════════════════════════════════
private void performFullClean() {
    if (rawPdf == null) return;
    try {
        byte[] cleanedData;

        // Крок 1: очистка Info Dictionary через PDFBox
        try (PDDocument doc = PDDocument.load(rawPdf)) {
            doc.getDocumentInformation().setCustomMetadataValue(KEY_NAME, null);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            doc.save(baos);
            cleanedData = baos.toByteArray();
        }

        // Крок 2: видалення хеш-маркера %%%% з хвоста
        String content = new String(cleanedData, StandardCharsets.ISO_8859_1);
```

```
        int hashIdx = content.lastIndexOf(HASH_MARKER);
        if (hashIdx != -1)
            cleanedData = Arrays.copyOfRange(cleanedData, 0, hashIdx);

        // Крок 3: видалення Comment-маркера %%% з тіла
        cleanedData = removeCommentMarker(cleanedData);

        JFileChooser fc = new JFileChooser();
        if (fc.showSaveDialog(this) == JFileChooser.APPROVE_OPTION) {
            File dest = fc.getSelectedFile();
            java.nio.file.Files.write(dest.toPath(), cleanedData);
            rawPdf = cleanedData;
            currentFile = dest;
            maxCapacity = (int) (rawPdf.length * 0.15);
            statusLabel.setText("Доступна ємність: " + maxCapacity + " байт");
            fileNameLabel.setText("Поточний файл: " + dest.getName());
            JOptionPane.showMessageDialog(this,
                    "Анонімізація завершена.\n" +
                            "Видалено: MetaData, Comment , Hash .\n" +
                            "Поточний файл: " + dest.getName());
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, "Помилка: " + ex.getMessage());
    }
}

/**
 * Видаляє рядок із маркером %%% (але не %%%%!) з байтів PDF.
 * Повертає очищений масив.
 */
private byte[] removeCommentMarker(byte[] data) {
    String content = new String(data, StandardCharsets.ISO_8859_1);
    int idx = findCommentMarker(content);
    if (idx == -1) return data; // маркера немає - нічого не чистимо

    // Знаходимо початок рядка (попередній \n) і кінець рядка (наступний \n)
    int lineStart = content.lastIndexOf('\n', idx);
    if (lineStart == -1) lineStart = 0; else lineStart++; // символ після \n

    int lineEnd = content.indexOf('\n', idx);
    if (lineEnd == -1) lineEnd = content.length(); else lineEnd++; // включаємо \n

    // Зшиваємо масив без рядка з маркером
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    baos.write(data, 0, lineStart);
    if (lineEnd &lt; data.length)
        baos.write(data, lineEnd, data.length - lineEnd);
    return baos.toByteArray();
}

// ═══════════════════════════════════════════════════════════════════════════
//  КРИПТОГРАФІЯ
// ═══════════════════════════════════════════════════════════════════════════

/** Розмір блоку для RSA 2048 + PKCS1Padding: 245 байт на вхід, 256 байт на виході */
private static final int RSA_BLOCK_PLAIN  = 245;
private static final int RSA_BLOCK_CIPHER = 256;

private byte[] encryptData(byte[] data) throws Exception {
    if (aesRadio.isSelected()) {
        JPasswordField pf = new JPasswordField();
        if (JOptionPane.showConfirmDialog(this, pf, "Пароль AES",
                JOptionPane.OK_CANCEL_OPTION) != JOptionPane.OK_OPTION) return null;
        byte[] key = Arrays.copyOf(
                MessageDigest.getInstance("SHA-256")
                        .digest(new String(pf.getPassword()).getBytes()), 16);
        Cipher c = Cipher.getInstance("AES");
```

```
            c.init(Cipher.ENCRYPT_MODE, new SecretKeySpec(key, "AES"));
            return c.doFinal(data);
        } else {
            File f = selectFile("RSA Public Key");
            if (f == null) return null;
            PublicKey pk = KeyFactory.getInstance("RSA").generatePublic(
                    new X509EncodedKeySpec(java.nio.file.Files.readAllBytes(f.toPath())));
            Cipher c = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            c.init(Cipher.ENCRYPT_MODE, pk);
            return rsaProcessBlocks(c, data, RSA_BLOCK_PLAIN);
        }
    }

    private byte[] decryptData(byte[] data) throws Exception {
        try {
            if (aesRadio.isSelected()) {
                JPasswordField pf = new JPasswordField();
                if (JOptionPane.showConfirmDialog(this, pf, "Пароль AES",
                        JOptionPane.OK_CANCEL_OPTION) != JOptionPane.OK_OPTION) return null;
                byte[] key = Arrays.copyOf(
                        MessageDigest.getInstance("SHA-256")
                                .digest(new String(pf.getPassword()).getBytes()), 16);
                Cipher c = Cipher.getInstance("AES");
                c.init(Cipher.DECRYPT_MODE, new SecretKeySpec(key, "AES"));
                return c.doFinal(data);
            } else {
                File f = selectFile("RSA Private Key");
                if (f == null) return null;
                PrivateKey pk = KeyFactory.getInstance("RSA").generatePrivate(
                        new PKCS8EncodedKeySpec(java.nio.file.Files.readAllBytes(f.toPath())));
                Cipher c = Cipher.getInstance("RSA/ECB/PKCS1Padding");
                c.init(Cipher.DECRYPT_MODE, pk);
                return rsaProcessBlocks(c, data, RSA_BLOCK_CIPHER);
            }
        } catch (Exception e) {
            throw new Exception("Помилка ключа/пароля!");
        }
    }

    /**
     * Обробляє дані блоками заданого розміру через вже ініціалізований Cipher.
     * При шифруванні blockSize = 245 (відкритий текст).
     * При дешифруванні blockSize = 256 (зашифрований блок RSA 2048).
     */
    private byte[] rsaProcessBlocks(Cipher cipher, byte[] data, int blockSize) throws Exception {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int offset = 0;
        while (offset &lt; data.length) {
            int len = Math.min(blockSize, data.length - offset);
            byte[] block = cipher.doFinal(data, offset, len);
            baos.write(block);
            offset += len;
        }
        return baos.toByteArray();
    }

    // ── Генерація RSA ключів ──────────────────────────────────────────────
    private void generateRSAKeys() {
        JFileChooser fc = new JFileChooser();
        fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
            try {
                KeyPairGenerator gen = KeyPairGenerator.getInstance("RSA");
                gen.initialize(2048);
                KeyPair pair = gen.generateKeyPair();
                File dir = fc.getSelectedFile();
                try (FileOutputStream pub = new FileOutputStream(new File(dir, "public.key"))) {
```

```
                pub.write(pair.getPublic().getEncoded());
            }
            try (FileOutputStream priv = new FileOutputStream(new File(dir, "private.key"))) {
                priv.write(pair.getPrivate().getEncoded());
            }
            JOptionPane.showMessageDialog(this, "Ключі RSA 2048 успішно створено.");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
  }

  // ── Вибір файлу ──────────────────────────────────────────────────────────
  private File selectFile(String title) {
    JFileChooser fc = new JFileChooser();
    fc.setDialogTitle(title);
    return fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION
        ? fc.getSelectedFile() : null;
  }

  // ── Точка входу ──────────────────────────────────────────────────────────
  public static void main(String[] args) { SwingUtilities.invokeLater(() -&gt; new PdfStegoExt().setVisible(true));
  }
}
```