МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КІБЕРБЕЗПЕКИ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
"_____"_____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"

Тема: Програмний модуль перевірки сайтів на автентичність

Виконавець: Сергій РОЖОК

Керівник: к.т.н. Наталія ГУЛАК

Нормоконтролер: к.т.н., доцент Андрій ПЕТРЕНКО

Київ 2024

ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Факультет комп'ютерних наук та технологій
Кафедра кібербезпеки
Освітній ступінь магістр
Спеціальність 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
«30»___08_____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Рожка Сергія Миколайовича

1. Програмний модуль перевірки сайтів на автентичність затверджена наказом ректора від 30.08.2024 р. №1695/ст.
2. Термін виконання роботи: з 30.08.2024 по 15.12.2024
3. Вихідні дані до роботи: електронний цифровий підпис Дія.Підпис, методи шифрування, веб-сторінка, мова програмування Python, парсинг.
4. Зміст пояснювальної записки: аналіз методів захисту авторських та суміжних прав на основі правової бази України, аналіз методів шифрування для електронного засвідчення документів та цілісності їх вмісту, розробка та тестування програмного модуля автоматизованої перевірки на автентичність.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: презентація, блок-схема, таблиці та рисунки.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Провести аналіз криптографічних і стеганографічних методів захисту авторських прав на електронний документ	30.08.2024 – 05.09.2024	*Виконано*
2.	Обгрунтувати вибір методів перевірки на автентичність електронного документа	06.09.2024 – 15.09.2024	*Виконано*
3.	Розробити алгоритм і програмний модуль автоматизованої перевірки на автентичність	16.09.2024 – 10.10.2024	*Виконано*
4.	Протестувати роботу програмного модуля автоматизованої перевірки на автентичність	12.10.2024 – 10.11.2024	*Виконано*

7. Дата видачі завдання: «30» __08__ 2024 р.

Керівник кваліфікаційної роботи: _____ Наталія ГУЛАК
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ Сергій РОЖОК
(підпис здобувача вищої освіти) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний модуль перевірки сайтів на автентичність»: 93 с., 34 рис., 5 табл., 45 літературних джерел.

Об’єкт дослідження: процес захисту цілісності інформації у електронному документообігу.

Предмет дослідження: методи шифрування для розпізнавання та порівняння текстової інформації.

Мета кваліфікаційної роботи: розробити та протестувати програмний модуль автоматизованої перевірки відповідності вмісту документів на автентичність, що розміщені на сайті.

Методи дослідження: базуються на основі об’єктно-орієнтованого програмування мовою Python.

Практична цінність: можливість автоматизації перевірки цілісності вмісту документа при використанні PAdES-B-B для електронного підпису і перевірки документів на автентичність на сайті.

Наукова новизна отриманих результатів полягає в наступному: було розроблено та протестовано програмний модуль автоматизованої перевірки вмісту отриманих та підписаних ЕЦП/КЕП документів на автентичність, що дало можливість зменшити термін перевірки їх на цілісність інформації та наочно показати місця її порушення.

Результати кваліфікаційної роботи рекомендується використовувати

КІБЕРБЕЗПЕКА, ЕЦП, АВТЕНТИЧНІСТЬ, ДОКУМЕНТООБІГ, ПАРСИНГ

ЗМІСТ

ВСТУП

Актуальність теми.

Сучасна цифрова трансформація та перехід на безпаперовий документообіг створили численні переваги для організацій та користувачів, але й привели до необхідності вирішення нових викликів у сфері аутентифікації та безпеки документів. Із збільшенням обсягу цифрових даних, що обробляються в режимі онлайн, зростає потреба у надійних методах перевірки автентичності документів, що опубліковані на сайтах, на які накладено електронний цифровий підпис/кваліфікований електронний підпис (ЕЦП/КЕП), а також верифікації відповідності цифрових копій їхнім оригіналам. Ця тема є особливо актуальною, зважаючи на підвищення частоти підробок документів, фальшивих цифрових підписів та інших кіберзагроз, що загрожують як користувачам, так і організаціям[1].

Електронний цифровий підпис (ЕЦП) є одним із найпоширеніших способів підтвердження автентичності цифрових документів, забезпечуючи захист від фальсифікації та внесення змін після підписання. Проте в умовах стрімкого розвитку технологій та вдосконалення методів підробок ЕЦП стикається з численними викликами. Зокрема, методи криптографічного аналізу, застосування підроблених сертифікатів і фальшивих підписів створюють серйозні загрози для довіри до електронного документообігу. Це спричиняє необхідність вдосконалення механізмів перевірки автентичності підписаних документів та забезпечення надійності ЕЦП у захисті даних[2].

Окрім перевірки підписів, існує проблема порівняння цифрових копій з оригіналами документів, яка є невід'ємною складовою процесу аутентифікації. В сучасному електронному середовищі документи можуть неодноразово копіюватися, поширюватися та редагуватися, що створює ризики втрати автентичності та зміни їхнього змісту. Оскільки правові та бізнесові рішення часто приймаються на основі аналізу цифрових документів, необхідність

підтвердження відповідності цифрових копій їхнім оригіналам набуває ключової значущості[3].

Проблема аутентифікації документів та перевірки ЕЦП особливо актуальна у сферах, де використання оригінальних документів є критичним для прийняття рішень та виконання правових дій, таких як фінансовий сектор, медицина, правоохоронні органи та державна адміністрація. Враховуючи значний обсяг конфіденційних даних, які циркулюють у цих галузях, гарантування автентичності документів стає основною умовою забезпечення безпеки даних та мінімізації ризиків шахрайства[4].

Застосування ЕЦП у фінансовій сфері дозволяє зменшити затрати часу та спростити процеси документообігу, але водночас створює виклики щодо надійної перевірки цифрових копій та верифікації автентичності підписаних документів. Зокрема, зростає потреба у системах, що можуть швидко й точно підтвердити справжність фінансових документів, захищаючи як клієнтів, так і організації від ризиків шахрайства[5].

У правовому середовищі документи, на які накладено ЕЦП, мають юридичну силу, і їх автентичність є важливим аспектом для забезпечення законності та дотримання правових норм. У багатьох країнах, включаючи Україну, електронні документи, підписані ЕЦП, визнані еквівалентними паперовим документам з підписом, що зобов’язує гарантувати їх достовірність і захищеність. Як наслідок, розробка програмного забезпечення для автоматизованої перевірки документів на автентичність, включаючи їх відповідність оригіналам, є необхідним кроком для підвищення ефективності електронного документообігу в правовій та адміністративній сферах[6].

Це особливо актуально в контексті електронного врядування, де документи часто подаються громадянами та організаціями в електронному форматі. Забезпечення їхньої автентичності, збереження юридичної сили та достовірності стає важливим завданням для державних установ, що працюють із великою кількістю документів щодня. Додатково, це підвищує довіру громадян

до електронних сервісів та сприяє ефективності роботи адміністративних структур[7].

Перевірка автентичності документів із ЕЦП може здійснюватися за допомогою різних методів, таких як порівняння цифрових відбитків документів, криптографічний аналіз та верифікація сертифікатів. Однак ці методи не завжди є достатньо надійними, особливо у випадках, коли документи можуть зазнавати несанкціонованих змін або маніпуляцій. Оскільки шахраї активно використовують підроблені сертифікати та методи обходу стандартних механізмів перевірки, необхідність у створенні нових рішень для аутентифікації документів є очевидною[8].

Інтеграція методів семантичного аналізу, машинного навчання та поведінкових алгоритмів для перевірки цифрових підписів і документів дозволяє підвищити точність виявлення підробок. Алгоритми машинного навчання можуть виявляти аномалії у структурі та змісті документа, які можуть свідчити про неавтентичність або спробу фальсифікації. Це особливо важливо у випадках, коли на документ накладено цифровий підпис, але існує підозра у зміні його вмісту після підписання[9].

Застосування штучного інтелекту та машинного навчання у перевірці автентичності документів дозволяє не лише автоматизувати процес верифікації, але й вдосконалити його за рахунок можливості розпізнавати навіть найменші відхилення у документах. Зокрема, використання технології штучного інтелекту дозволяє автоматично аналізувати зміст документа та виявляти потенційні ознаки маніпуляції, а також проводити порівняння з оригінальним контентом. Такий підхід дозволяє мінімізувати ризики підробки цифрових документів та підвищує надійність електронного документообігу в різних сферах[10].

Наприклад, система перевірки документів може порівнювати кожен підписаний документ з його оригінальною версією або з базою даних відомих автентичних документів, що дозволяє виявити невідповідності на рівні контенту чи структури. Це допомагає швидко ідентифікувати підробки та мінімізує ризики внесення несанкціонованих змін.

На міжнародному рівні існує значний інтерес до розвитку рішень для перевірки автентичності документів. Зокрема, в країнах Європейського Союзу вже запроваджені правові норми, що регулюють використання електронного підпису і перевірку документів в електронному середовищі. Наприклад, стандарт PAdES (PDF Advanced Electronic Signature) забезпечує підтримку електронного підпису для PDF-документів, що дозволяє інтегрувати перевірку автентичності документа безпосередньо в PDF-файли[11].

Для України це може стати важливим кроком для досягнення відповідності міжнародним стандартам та інтеграції в глобальне цифрове середовище. Розробка таких рішень дозволить забезпечити надійну аутентифікацію документів на національному рівні, а також сприятиме адаптації українського законодавства до європейських норм у сфері кібербезпеки та електронного документообігу.

Таким чином, розробка програмного модуля для перевірки автентичності документів із цифровими підписами є актуальною та необхідною у світлі сучасних викликів. Це питання охоплює не лише проблематику забезпечення цілісності та достовірності документів, але й важливий аспект захисту персональних даних і прав громадян у цифровому середовищі. Вдосконалення методів перевірки та автоматизація процесів аутентифікації документів створить надійний інструмент для захисту від підробок та посилить ефективність електронного документообігу[12].

Мета і завдання виконання кваліфікаційної роботи. Розробити та протестувати програмний модуль автоматизованої перевірки відповідності вмісту документів на автентичність.

Завдання:

1. Аналіз методів захисту авторських та суміжних прав на основі правової бази України.
2. Аналіз методів шифрування для електронного засвідчення документів та цілісності їх вмісту.

3. Розробка та тестування програмного модуля автоматизованої перевірки на автентичність.

Об'єкт – процес захисту цілісності інформації у електронному документообігу.

Предмет – методи шифрування для розпізнавання та порівняння текстової інформації.

Об'єкт і предмет дослідження

Об'єктом дослідження цієї роботи - процес верифікації автентичності цифрових документів, зокрема PDF-файлів, з метою забезпечення захисту від підробок та несанкціонованих змін в електронному документообігу.

Натомість, предметом дослідження є методи та програмні засоби порівняння вмісту оригінального PDF-документа та його підписаної версії для визначення порушення цілісності інформації, яка розміщена на зовнішньому веб-ресурсі. Основною метою дослідження є виявлення відмінностей між оригіналом та зміненою версією документа, що дозволяє здійснювати ефективний контроль автентичності документів[12].

Методи дослідження та їх обґрунтування

Для реалізації поставленої задачі було розроблено програмний модуль, який перевіряє оригінальний документ на цілісність вмісту, створений у PDF-форматі, з вмістом підписаного електронного документа, що дозволяє оперативно і вірно виявляти несанкціоновані зміни вмісту документа.

Для розв’язання цієї задачі було обрано такі методи:

- Метод аналізу текстових даних

Програмний модуль здійснює аналіз текстового вмісту обох документів з метою пошуку та порівняння текстових елементів. Для цього використовується Python-бібліотека, що дозволяє розпізнавати текст у PDF-файлах та обробляти його для подальшого аналізу. Даний підхід дає змогу виділити розбіжності між оригіналом і зміненою версією на основі текстового аналізу[12].

- Метод порівняння тексту з підсвічуванням відмінностей:

Для забезпечення зручного представлення результатів порівняння використовується підсвічування відмінностей різними кольорами. Такий метод дозволяє користувачу швидко візуалізувати розбіжності між документами, що підвищує ефективність роботи з програмою та зменшує ймовірність пропущення важливих змін. Дана методика, реалізована за допомогою засобів мови програмування Python, забезпечує доступність результатів та простоту використання розробленої системи[12].

- Метод доступу до веб-ресурсів

Оскільки змінений документ знаходиться на зовнішньому сайті, програма використовує бібліотеки Python для здійснення HTTP-запитів. Це дозволяє завантажити потрібний документ з веб-ресурсу для подальшої обробки та аналізу. Даний метод є необхідним для автоматизації процесу перевірки, оскільки дозволяє отримувати документи безпосередньо з інтернет-джерел[12].

- Алгоритми пошуку та виділення відмінностей

Для порівняння вмісту документів обрано алгоритми, які визначають текстові відмінності та підсвічують їх, що спрощує аналіз розбіжностей. Використання спеціалізованих бібліотек для обробки PDF-файлів та тексту на Python дозволяє реалізувати ефективний та зручний механізм порівняння великих обсягів даних[12].

- Метод тестування та валідації

Розроблена програма проходить тестування на реальних PDF-документах для перевірки коректності роботи алгоритму порівняння. Це дозволяє оцінити точність та надійність системи при виявленні змін у різних умовах, включаючи варіативність текстового вмісту та форматування документів[13].

Загалом, обрані методи дозволяють досягти високої точності та зручності у перевірці автентичності документів. Застосування цих методів забезпечує надійну верифікацію та виявлення підробок, що робить розроблену систему ефективним інструментом для забезпечення безпеки електронного документообігу в умовах сучасних викликів кібербезпеки.

Методи дослідження. Базуються на основі об’єктно-орієнтованого програмування мовою Python.

Наукова новизна отриманих результатів. Полягає отриманих результатів полягає в наступному: було розроблено та протестовано програмний модуль автоматизованої перевірки вмісту отриманих та підписаних ЕЦП/КЕП документів на автентичність, що дало можливість зменшити термін перевірки їх на цілісність інформації та наочно показати місця її порушення.

Практичне значення отриманих результатів. Автоматизація перевірки цілісності вмісту документа при перевірці на автентичність, зменшення затрат на штатний персонал (юридичний відділ).

Апробація отриманих результатів. VII International scientific and practical conference «World educational trends: lifelong learning in the information society»

Публікації. Рожок С.М. Програмний модуль перевірки сайтів на автентичність /С.М. Рожок, Н.К. Гулак //VII International scientific and practical conference «World educational trends: lifelong learning in the information society», 15-18 жовтня 2024р. - Афіни, Греція. – С. 64-66.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ЗАХИСТУ АВТОРСЬКИХ ТА СУМІЖНИХ ПРАВ НА ОСНОВІ ПРАВОВОЇ БАЗИ УКРАЇНИ

Захист авторських та суміжних прав в Україні базується на широкій нормативній основі, яка включає закони, підзаконні акти та міжнародні договори. Основним законодавчим актом є Закон України "Про авторське право і суміжні права" № 3792-XII від 23 грудня 1993 року, який охоплює права авторів на літературні, художні, музичні твори, комп'ютерні програми та інші результати творчої діяльності. Цей аналіз розглядає структуру та зміст основних методів захисту авторських прав у відповідності до української правової бази, з акцентом на ключові положення і прикладом застосування на практиці[14].

1.1 Законодавча основа захисту авторських та суміжних прав в Україні

1.1.1 Основні нормативні акти

Законодавча основа захисту авторських прав в Україні охоплює кілька основних актів (див. рис. 1.1[15]):

- Закон України "Про авторське право і суміжні права" – регулює права авторів на результати їх творчої діяльності[16].
- Цивільний кодекс України (глава 75), який доповнює авторське право положеннями про захист прав інтелектуальної власності[17].
- Кримінальний кодекс України (статті 176 та 177) встановлює кримінальну відповідальність за порушення авторських прав, включаючи незаконне використання та відтворення творів[18].

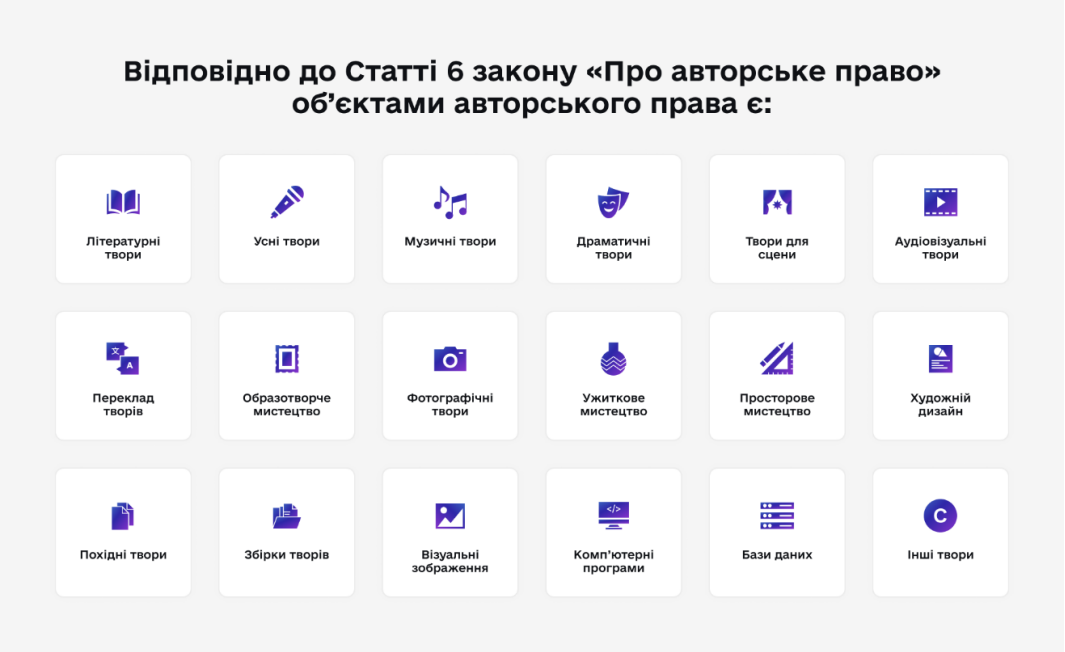


Рис. 1.1. Стаття 6 закону «Про авторське право»

1.1.2 Міжнародні стандарти у сфері авторських прав

Україна є учасницею міжнародних угод, таких як Бернська конвенція про охорону літературних і художніх творів та Договір Всесвітньої організації інтелектуальної власності (ВОІВ) про авторське право. Відповідно до цих угод, авторські права охороняються автоматично, без необхідності реєстрації.

1.2 Правове закріплення авторських прав

1.2.1 Автоматичний захист авторських прав

В Україні авторські права на твори виникають з моменту їх створення і не потребують обов'язкової реєстрації. Це положення є важливим для охорони творчих результатів і відповідає нормам міжнародного права, зокрема, Бернській конвенції[19].

1.2.2 Види прав авторів

Закон поділяє права авторів на майнові та немайнові:

- Майнові права – включають право на використання, відтворення, розповсюдження, зміну та комерційне застосування твору.

- Немайнові права – право авторства, право на ім’я, право на захист репутації твору. Ці права є невідчужуваними і належать автору незалежно від передачі майнових прав. Табл. 1.1 демонструє основні види авторських прав:

Таблиця 1.1

Основні види авторських прав

Вид права	Опис	Приклад захисту
Майнові	Відтворення, розповсюдження, зміна твору	Ліцензія на використання твору
Немайнові	Право авторства, право на ім'я, право на захист репутації твору	Заперечення на внесення змін до твору

1.2.3 Майнові права

Майнові права надають можливість автору отримувати економічну вигоду від свого твору. Вони включають:

- Право на відтворення – дозволяє автору контролювати копіювання твору у різних формах (друк, цифрове копіювання).
- Право на розповсюдження – включає продаж, передачу або оренду копій твору.
- Право на публічне виконання – використання твору для публічних показів, концертів тощо.

1.2.4 Немайнові права

Немайнові права є невідчужуваними і зберігаються за автором навіть після передачі майнових прав. Сюди відносяться:

- Право авторства – право визнаватися автором твору.

- Право на ім’я – автор має право вимагати, щоб його ім’я зазначалося на творі.
- Право на захист репутації твору – автор може вимагати запобігання спотворенню його твору.

1.3 Система охорони суміжних прав

1.3.1 Основні категорії суб'єктів суміжних прав

До суб'єктів суміжних прав належать:

- Виконавці: отримують права на свої виступи, виконання.
- Виробники фонограм та відеограм: мають право на контроль за відтворенням та розповсюдженням записів.
- Організації мовлення: захищають свої трансляції від несанкціонованого використання.

Ця категорія прав є важливою для захисту інтересів учасників творчого процесу, які сприяють поширенню творів (див. рис. 1.2[15]).

Рис. 1.2. Один із способів вказати автора на прикладі картинки

1.4 Підстави захисту авторських і суміжних прав

У цьому розділі доцільно розкрити різні основи для захисту авторських прав:

- Автоматичний захист авторських прав

В Україні, як і в більшості країн, охорона авторських прав виникає з моменту створення твору, тобто не потребує реєстрації. У цьому підрозділі можна розкрити положення Закону України «Про авторське право і суміжні права», які підтверджують, що охорона діє незалежно від процедури реєстрації, якщо це не передбачено договором.

- Захист прав за допомогою реєстрації

В Україні можна зареєструвати авторське право, хоча це і не є обов'язковою умовою для охорони. Однак реєстрація може полегшити доказ авторства у разі спору. Реєстрація може здійснюватися у відповідних державних установах, таких як Міністерство розвитку економіки, торгівлі та сільського господарства України.

1.5 Способи захисту прав інтелектуальної власності

- Адміністративний захист

Опис адміністративних процедур, які можуть використовуватися для вирішення спорів з питань авторських прав, наприклад, у рамках Антимонопольного комітету України. Це важливо у випадках порушень, пов'язаних з недобросовісною конкуренцією та порушенням права на комерційне використання твору.

- Цивільно-правовий захист

Включає позов до суду про припинення порушення прав, компенсацію за завдану шкоду, стягнення моральної шкоди. Важливо також зазначити, що можливість звернутися до суду передбачена ст. 432 Цивільного кодексу України[17].

- Кримінально-правовий захист

Важливим є висвітлення кримінальної відповідальності за порушення авторських прав, зокрема у випадках умисного порушення, як-от піратство.

Український Кримінальний кодекс, стаття 176, передбачає покарання за порушення авторського права у вигляді штрафу або навіть позбавлення волі[18].

1.6 Механізми захисту в разі порушення авторських прав

1.6.1 Цивільно-правові засоби захисту

Українське законодавство передбачає можливість для авторів вимагати захисту своїх прав шляхом:

- Відшкодування збитків – компенсація за порушення прав шляхом судового позову.
- Вилучення або знищення контрафактних копій – знищення копій, виготовлених незаконно.
- Компенсація моральної шкоди – виплати за моральні збитки у випадку порушення немайнових прав.

1.6.2 Адміністративні санкції

Адміністративні санкції включають штрафи та конфіскацію незаконно виготовлених копій.

1.6.3 Кримінальні санкції

Кримінальний кодекс України передбачає покарання за порушення авторських прав, що включає штрафи, виправні роботи, або позбавлення волі.

1.7 Використання договорів у сфері авторських прав

Договори є важливим інструментом захисту прав інтелектуальної власності. У цьому розділі можна детальніше зупинитися на різних видах договорів:

- Ліцензійний договір: ліцензія дозволяє автору дозволяти або забороняти використання свого твору іншим особам. Тут доцільно висвітлити різницю між виключними і невиключними ліцензіями.

- Договір про відчуження прав інтелектуальної власності: автор передає свої права іншій особі, яка стає новим власником прав. Особливу увагу варто приділити вимогам до таких договорів відповідно до законодавства України.

1.8 Порівняльний аналіз методів захисту документів за допомогою цифрового підпису на прикладі ЕЦП PAdES-B-B в Дії

У сучасному цифровому світі електронний цифровий підпис (ЕЦП) стає важливим інструментом для захисту електронних документів. Одним із сучасних рішень, що застосовується в Україні, є підписання PDF-документів за допомогою формату PAdES-B-B через платформу Дія. Цей формат відзначається високим рівнем захисту та відповідності міжнародним стандартам. Розглянемо особливості PAdES-B-B, його переваги порівняно з іншими методами захисту документів, а також його відповідність потребам українських користувачів.

1.9 PAdES-B-B як стандарт підпису PDF-документів

1.9.1 Основи стандарту PAdES

PAdES (PDF Advanced Electronic Signatures) – це стандарт цифрових підписів для PDF-документів, розроблений Європейським інститутом стандартизації телекомунікацій (ETSI). Версія PAdES-B-B включає підтримку довготривалого зберігання документів та часові мітки, що підтверджують справжність підпису навіть після закінчення терміну дії сертифікатів[20].

1.9.2 Особливості PAdES-B-B

PAdES-B-B дозволяє:

- Додавати цифровий підпис до PDF-документів без порушення їхньої структури.
- Інтегрувати часові мітки, що підтверджують наявність документу у визначений час.

- Використовувати алгоритми для забезпечення цілісності документу.
- Забезпечувати можливість довгострокового підтвердження підпису.

1.9.3 Переваги використання PAdES-B-B у Дії

- Простота та доступність для користувачів

Однією з ключових переваг застосування PAdES-B-B через додаток Дія є простота та зручність використання для громадян. Оскільки додаток інтегрований з державними реєстрами, користувачі мають можливість підписувати документи онлайн, що особливо важливо для віддаленого підпису.

- Відповідність міжнародним стандартам

PAdES-B-B у Дії відповідає міжнародним стандартам безпеки (ETSI), що забезпечує високий рівень надійності. Це робить документи, підписані у форматі PAdES-B-B, придатними для використання не лише в Україні, але й на міжнародному рівні (див. табл. 1.2).

Таблиця 1.2

Основні переваги PAdES-B-B

Перевага	Опис
Довгострокове зберігання	Підтримка часових міток та архівних даних для підтвердження справжності підпису.
Висока сумісність	Сумісність із різними PDF-читачами та міжнародними стандартами.
Надійність та безпека	Шифрування даних і захист від несанкціонованих змін.

- Захист від фальсифікацій та підробок

Стандарт PAdES-B-B підтримує цифрові сертифікати та часові мітки, що дозволяє ідентифікувати автора документу і підтверджувати його справжність навіть після закінчення терміну дії сертифіката. Це забезпечує високу стійкість до фальсифікацій та захист від підробок.

1.9.4 Порівняння з іншими методами захисту документів

- ЕЦП без часових міток (CAdES)

Формат CAdES (CMS Advanced Electronic Signatures) також забезпечує високий рівень захисту, але менш ефективний для довгострокового зберігання, оскільки не підтримує часові мітки для підтвердження часу підписання. Це робить CAdES менш зручним для ситуацій, коли потрібно підтвердити автентичність підпису через кілька років[20].

- XAdES для XML-документів

XAdES (XML Advanced Electronic Signatures) – ще один стандарт підпису, що використовується для XML-документів. Він забезпечує високу надійність, але його застосування обмежене до XML-документів, що робить його менш зручним для PDF-файлів і документів широкого використання в Україні (див. табл. 1.3).

Таблиця 1.3

Порівняння PAdES-B-B з іншими стандартами

Стандарт	Переваги	Недоліки
PAdES-B-B	Довгострокове зберігання, сумісність	Вимагає підтримки PDF
CAdES	Високий рівень захисту, гнучкість	Немає часових міток, обмеження для PDF
XAdES	Розширена підтримка XML документів	Не сумісний із PDF

- Інші механізми захисту документів

На додаток до цифрового підпису існують такі механізми, як водяні знаки та захист паролем, але вони є менш надійними порівняно з електронним підписом:

Водяні знаки – забезпечують базовий рівень візуального захисту, проте не гарантують цілісності та не можуть підтвердити авторство.

Захист паролем – обмежує доступ до документа, але не гарантує, що його не буде підроблено чи змінено.

1.9.5 Використання PAdES-B-B як пріоритетного стандарту

PAdES-B-B забезпечує надійний захист документів у форматі PDF, що відповідає вимогам як українського законодавства, так і міжнародних стандартів. Він дозволяє безпечно підписувати документи, зберігаючи їхній вміст незмінним, та забезпечує можливість довготривалого зберігання, що є перевагою для юридично значущих документів.

Окрім PAdES-B-B, доцільно розглянути альтернативні технології підпису та їхні характеристики:

- Простий електронний підпис. Придатний для використання в ситуаціях, де не потрібен високий рівень захисту. Проте він не є юридично зобов'язуючим у більшості випадків.
- Підпис CAdES та XAdES. Використовується для документів у форматі CMS або XML відповідно, з надійним рівнем захисту, проте менш зручний для PDF-документів, що поширені в Україні.

1.10 Висновки за розділом 1

Законодавча база України у сфері захисту авторських прав є достатньо розвиненою і забезпечує авторам надійний захист. Доступні методи захисту дозволяють захищати авторські права як у цивільному, так і кримінальному порядку, що робить законодавство ефективним інструментом для захисту прав інтелектуальної власності.

Застосування PAdES-B-B для електронного підпису в Дії забезпечує користувачам високий рівень захисту, зручність використання та відповідність міжнародним стандартам. Завдяки можливостям довготривалого зберігання підписаних документів, PAdES-B-B є найкращим вибором для захисту PDF-документів порівняно з іншими форматами.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ ШИФРУВАННЯ ДЛЯ ЕЛЕКТРОННОГО ЗАСВІДЧЕННЯ ДОКУМЕНТІВ ТА ЦІЛІСНОСТІ ЇХ ВМІСТУ

Захист цифрових документів є важливою складовою інформаційної безпеки, особливо в умовах зростаючої кількості транзакцій та взаємодії в цифровому середовищі. Цифрові документи часто містять конфіденційну інформацію, яка потребує захисту від несанкціонованого доступу, підробки чи модифікацій, тому сьогодні існує безліч методів захисту, що забезпечують автентичність, цілісність та конфіденційність документів. Використання надійних методів захисту є особливо важливим у сферах, де відбувається обмін інформацією з юридичною силою, як-от фінансові послуги, правові угоди, урядова та державна документація[21].

2.1 Загальна характеристика методів захисту цифрових документів

2.1.1 Методи захисту цифрових документів

Методи захисту цифрових документів поділяються на кілька основних категорій:

- Шифрування – це метод, який перетворює відкритий текст документа у зашифровану форму, доступну лише користувачам, які мають відповідний ключ для його дешифрування[22].
- Електронний цифровий підпис (ЕЦП) – забезпечує підтвердження автентичності та авторства документа, а також його незмінність після підписання[22].
- Хешування – створення унікальної «відбитка» або хеш-значення документа, що змінюється при будь-якій модифікації даних і дозволяє виявити підробки чи зміни в документі[22].

- Водяні знаки – невидимі чи видимі маркери, додані до документа, які підтверджують автентичність або індивідуальність документа та слугують додатковим рівнем захисту[22].

2.1.2 Шифрування

Шифрування документів полягає у перетворенні інформації таким чином, щоб вона була доступна лише тим, хто має правильний ключ для її розшифрування. Воно буває двох основних типів:

- Симетричне шифрування: використовує один ключ для шифрування і дешифрування, що забезпечує високу швидкість обробки, але вимагає безпечного обміну ключем[22].
- Асиметричне шифрування: застосовує пару ключів – відкритий та закритий. Відкритий ключ використовується для шифрування, а закритий – для дешифрування. Це дозволяє безпечно передавати зашифровану інформацію без потреби передавати закритий ключ, що значно підвищує безпеку[22].

Шифрування забезпечує захист даних навіть при їх передачі через мережу, а також захищає документи від перегляду та модифікації. Найбільш популярними алгоритмами шифрування є RSA, Advanced Encryption Standard (AES) та Elliptic Curve Cryptography (ECC), кожен з яких має свої переваги і використовується залежно від потреби у швидкості, стійкості до атак і зручності в реалізації.

2.1.3 Електронний цифровий підпис (ЕЦП)

Електронний цифровий підпис є ефективним методом підтвердження автентичності та авторства цифрових документів. Він забезпечує юридичну силу електронних документів і дозволяє виявити будь-які несанкціоновані зміни, внесені після підписання. ЕЦП створюється за допомогою шифрування та хешування, що унеможливлює підробку підпису без володіння секретним ключем власника підпису[23].

Системи ЕЦП зазвичай базуються на алгоритмах асиметричного шифрування, таких як RSA, де використовується два ключі: один для підписання документа, а інший для перевірки автентичності підпису. Основні стандарти ЕЦП включають такі, як PAdES (PDF Advanced Electronic Signatures), які адаптовані для зручного використання в електронному документообігу, наприклад, для підписання PDF-документів.

2.1.4 Хешування

Хешування – це процес, що перетворює дані довільного розміру в коротке фіксоване значення, яке є унікальним для кожного набору вхідних даних. Хеш-функції, такі як SHA-256 або MD5, створюють «відбиток» документа, який змінюється навіть при мінімальних змінах у тексті чи даних. Хешування є незворотнім процесом, тобто з хеш-значення неможливо відновити початкові дані, однак його можна порівняти з новостворeним хешем, що дозволяє виявити навіть незначні зміни у документі[23].

Хешування широко застосовується в поєднанні з іншими методами захисту, такими як шифрування чи цифрові підписи, щоб забезпечити перевірку цілісності документів. Прикладом може бути використання хешу у створенні ЕЦП: при підписанні документа обчислюється його хеш, який потім шифрується закритим ключем підписувача.

2.1.5 Водяні знаки

Водяний знак – це додатковий елемент захисту, що використовується для вказівки на автентичність документа або авторські права на нього. Водяні знаки бувають двох типів:

- Видимі водяні знаки: логотипи, текст або зображення, які накладаються на документ і можуть бути візуально помітні. Вони часто використовуються для друкованих документів або для зображень в інтернеті з метою захисту авторських прав[23].

- Невидимі водяні знаки: цифрові ідентифікатори, які вбудовані у структуру документа або зображення і не помітні для людського ока. Цей тип водяних знаків особливо ефективний для цифрових документів, адже вони зберігають інформацію про автора або власника навіть при спробі копіювання або редагування документа[23].

Цифрові водяні знаки можуть забезпечити відстеження та контроль над розповсюдженням документа, ідентифікуючи джерело походження документа чи зміни, внесені третіми сторонами. Вони використовуються для захисту цінної інформації, а також можуть поєднуватися з іншими методами захисту для підвищення безпеки цифрових документів[23]. Порівняння цих методів наведено нижче (див. табл. 2.1).

Таблиця 2.1

Переваги та недоліки основних методів захисту документів

Метод захисту	Переваги	Недоліки
Шифрування	Захищає конфіденційність, сильний захист від злому	Складність управління ключами
Електронний підпис	Гарантія автентичності, юридична сила	Залежність від надійного зберігання ключів
Хешування	Простота та ефективність перевірки цілісності	Може бути вразливим до колізій
Водяні знаки	Легке підтвердження авторства, контроль поширення	Легко видаляється візуальний водяний знак

2.1.6 Значення використання комбінованих методів захисту

Використання лише одного методу не завжди може забезпечити комплексний захист документа. Наприклад, шифрування захищає лише

конфіденційність документа, але не підтверджує його автентичність. З іншого боку, цифровий підпис підтверджує автентичність, проте він не захищає від можливості копіювання документа. Тому для підвищення рівня безпеки документів часто використовують комбінації методів, наприклад, шифрування з цифровим підписом або водяні знаки з хешуванням. Це дозволяє досягти більш надійного захисту, зокрема для документів, що потребують збереження в умовах довготривалого зберігання або передачі через мережу[22].

Шифрування – це процес перетворення зрозумілої інформації (плоского тексту або відкритих даних) у незрозумілу форму з використанням спеціальних алгоритмів і ключів. Шифрування документів є одним з основних методів забезпечення конфіденційності та цілісності даних у цифровому середовищі. В умовах зростання кіберзагроз, зокрема фальсифікацій і несанкціонованого доступу до важливої інформації, шифрування дозволяє зберігати конфіденційність та захищати документи від несанкціонованої зміни.

Шифрування документів забезпечує їх захист як у процесі зберігання, так і під час передачі через мережу. Однак для ефективної реалізації шифрування важливо правильно вибирати тип шифрування, що відповідає конкретним вимогам безпеки, обсягу даних і доступним обчислювальним ресурсам[22].

2.1.7 Визначення шифрування документів

Шифрування документів – це процес зміни даних документа таким чином, щоб вони стали недоступними для прочитання сторонніми особами, а лише для тих, хто має відповідний ключ для розшифровки. Цей процес використовує математичні алгоритми, які перетворюють дані документа у вигляд, що є непридатним для інтерпретації без відповідної ключової інформації (див. рис. 2.1[24]).

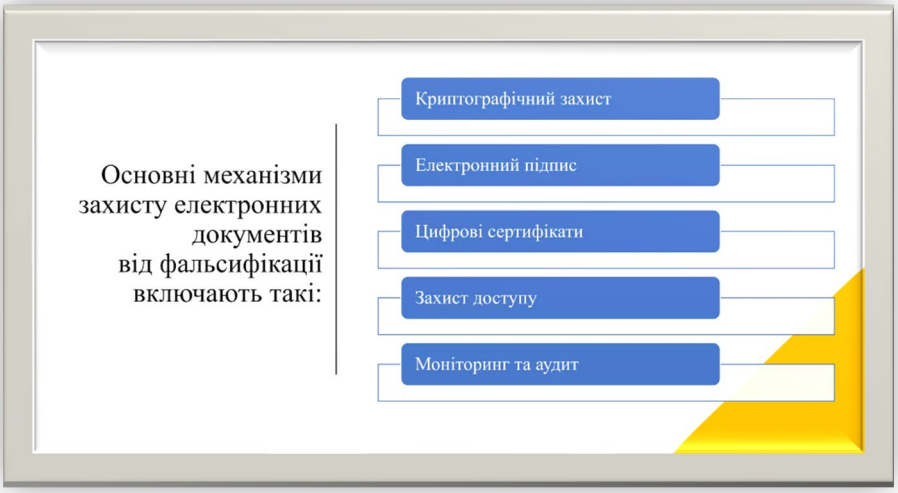


Рис. 2.1. Основні механізми захисту електронних документів від фальсифікації

Для реалізації шифрування документів використовуються різноманітні методи та алгоритми, що забезпечують необхідний рівень захисту в залежності від конкретних потреб, серед яких важливою є конфіденційність, цілісність та автентичність даних.

2.2 Типи шифрування

Існує два основних типи шифрування: симетричне та асиметричне шифрування. Кожен з цих типів має свої переваги та недоліки, а також застосовується в різних контекстах в залежності від вимог безпеки та специфіки роботи з документами.

2.2.1 Симетричне шифрування

Симетричне шифрування (або секретний ключ) – це метод шифрування, за якого для шифрування та розшифровки даних використовується один і той самий ключ. Це означає, що як для процесу шифрування, так і для розшифровки даних потрібно мати доступ до одного і того ж ключа (див. рис. 2.2[25]).

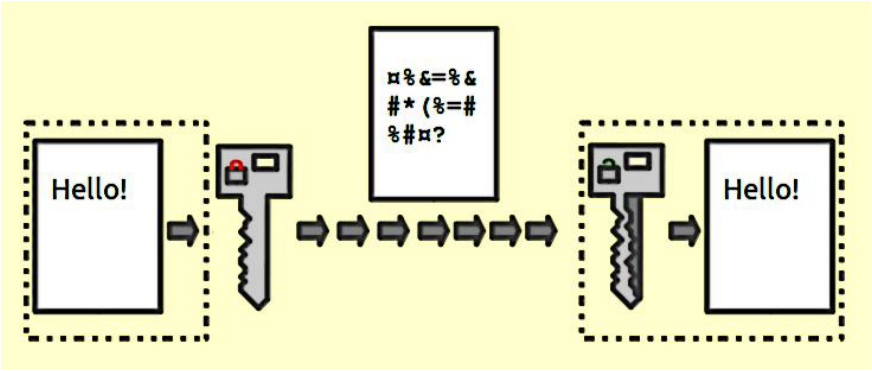


Рис. 2.2. Симетричне шифрування

Алгоритми симетричного шифрування: До найпоширеніших алгоритмів симетричного шифрування належать:

- AES (Advanced Encryption Standard) – це стандарт шифрування, який використовує ключі довжиною 128, 192 або 256 біт. AES забезпечує високу швидкість шифрування та є одним із найнадійніших алгоритмів на сьогодні[26].
- DES (Data Encryption Standard) – один із перших стандартів симетричного шифрування. Хоча DES був широко застосовуваний, його використання вже застаріло через обмежену довжину ключа (56 біт), що робить його вразливим до атак методом перебору[26].
- 3DES (Triple DES) – покращена версія DES, яка застосовує тричі одне й те саме шифрування з трьома різними ключами, але знову ж таки він менш ефективний порівняно з новітніми алгоритмами[26].

Перевагою симетричного шифрування є швидкість, адже симетричне шифрування зазвичай набагато швидше за асиметричне, оскільки обчислювальні операції є менш ресурсоємними. І також перевагою є менше навантаження на систему. Завдяки меншій складності алгоритмів шифрування система працює швидше при великих обсягах даних.

Недоліками симетричного шифрування є проблема розподілу ключа та масштабованість. Найбільша проблема цього методу полягає в необхідності безпечної передачі або зберігання секретного ключа. Якщо ключ потрапляє до зловмисника, це може призвести до компрометації всіх зашифрованих даних.

Кожен набір користувачів потребує окремий ключ, що створює проблеми у великих організаціях чи при масовому обміні даними.

2.2.2 Асиметричне шифрування

Асиметричне шифрування (або відкритий ключ) – це метод шифрування, за якого для шифрування та розшифровки даних використовуються різні ключі. Один із ключів, відомий як публічний ключ, використовується для шифрування, а інший, приватний, – для розшифровки[27] (див. рис. 2.3)

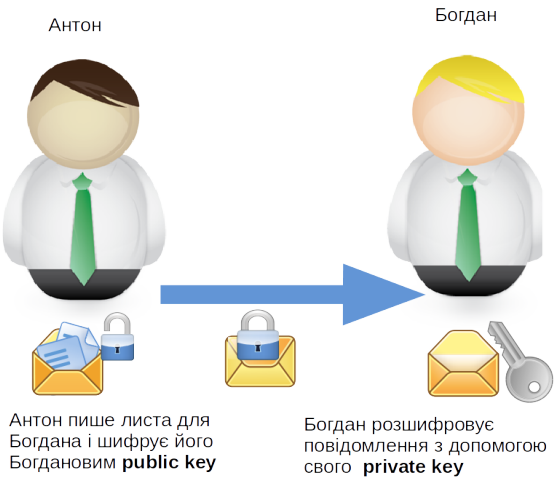


Рис. 2.3. Асиметричне шифрування

Алгоритми асиметричного шифрування:

- RSA (Rivest-Shamir-Adleman) – один з найпоширеніших алгоритмів асиметричного шифрування. RSA використовує пару ключів (публічний та приватний) для забезпечення як шифрування, так і цифрового підпису. RSA забезпечує високий рівень безпеки, але є більш повільним в порівнянні з симетричним шифруванням[26].
- ECC (Elliptic Curve Cryptography) – це криптографія на основі еліптичних кривих, яка надає високу безпеку за допомогою коротших ключів порівняно з RSA. ECC застосовується в багатьох сучасних технологіях безпеки, включаючи захист електронних підписів[26].

- ElGamal – ще один популярний асиметричний алгоритм, який використовується в протоколах шифрування, таких як PGP (Pretty Good Privacy)[26].

Переваги асиметричного шифрування:

- Безпека розподілу ключів

Публічний ключ може бути відкрито розповсюджений серед користувачів, тому проблема передачі ключів зникає. Лише власник приватного ключа може розшифрувати дані, зашифровані його публічним ключем.

- Підтримка цифрових підписів

Асиметричне шифрування є основою для створення та перевірки цифрових підписів, що дозволяє підтвердити автентичність і цілісність документів.

Недоліки асиметричного шифрування:

- Повільний процес шифрування

Асиметричне шифрування є значно більш обчислювально важким, тому його застосовують переважно для шифрування невеликих обсягів даних або для створення цифрових підписів, в той час як великі обсяги даних шифруються за допомогою симетричного шифрування.

- Більше навантаження на систему

Завдяки складним математичним операціям асиметричне шифрування вимагає більших обчислювальних ресурсів і займає більше часу на виконання.

2.2.3 Гібридне шифрування

Гібридне шифрування – це комбінація симетричного та асиметричного шифрування, що використовується для отримання переваг обох методів[26].

У цьому підході для шифрування самих даних застосовується швидкий алгоритм симетричного шифрування, а публічний і приватний ключі використовуються для безпечної передачі симетричного ключа шифрування. Це дозволяє поєднувати швидкість симетричного шифрування для великих обсягів даних з безпекою асиметричного шифрування при передачі ключа.

Перевагою гібридного шифрування є висока ефективність, бо використання симетричного шифрування для самих даних дозволяє зберігати швидкість обробки, тоді як використання асиметричного шифрування для обміну ключами гарантує безпеку цього процесу. Також слід відзначити широке застосування, адже гібридне шифрування є основою для багатьох протоколів, таких як TLS/SSL, що використовуються в Інтернеті для забезпечення безпеки комунікацій.

Недоліком гібридного шифрування є необхідність управління двома типами ключів. Для реалізації цього методу потрібно одночасно управляти як симетричними, так і асиметричними ключами, що вимагає додаткових заходів безпеки і складності в управлінні[26].

2.3 Визначення та роль ЕЦП у забезпеченні автентичності документів

Електронний цифровий підпис (ЕЦП) є технологією, що використовується для підтвердження автентичності та цілісності цифрових документів. Він є електронним аналогом власноручного підпису та забезпечує можливість підписання документів в електронному вигляді з гарантією, що підписаний документ є незмінним і походить від конкретного підписанта (див. рис. 2.4[28]).

Рис. 2.4. Ілюстрація використання токену (USB-пристрою) для підпису

Загальна мета ЕЦП – це забезпечення двох основних властивостей:

- Автентичність – підтвердження того, що підписаний документ дійсно належить конкретному автору або підписанту. Ця властивість підтверджує особу підписанта, адже для створення ЕЦП використовується унікальний ключ, що відомий лише його власнику[29].
- Цілісність – гарантія того, що документ не був змінений після підписання. Якщо під час передачі або зберігання документа виникнуть будь-які зміни, електронний підпис стане недійсним[29].

ЕЦП використовується для захисту від підробки документів, оскільки забезпечує можливість підтвердити, що документ був підписаний автором і не зазнав змін після підписання. Це особливо важливо в умовах цифрового середовища, де легкість копіювання та зміни файлів ставить під загрозу юридичну силу документів[29].

Залежно від рівня безпеки, ЕЦП поділяються на три основні види:

2.3.1 Простий електронний підпис (ПЕП)

Прості електронні підписи є найбільш базовими і використовуються для підтвердження згоди або ідентифікації користувача при виконанні нескладних операцій.

Він може бути реалізований у вигляді простих даних (наприклад, PIN-коду, пароля або іншого електронного ідентифікатора), що підтверджують особу користувача.

Прості підписи не забезпечують високий рівень безпеки, оскільки вони не використовують криптографічні методи, а тому не можуть гарантувати ні автентичність, ні цілісність документа на високому рівні[29].

2.3.2 Удосконалений електронний підпис (УЕП)

Удосконалений підпис є більш складним і базується на криптографічних методах. Він використовує особистий ключ підписанта та алгоритми хешування для створення підпису.

Такий підпис надає більш високий рівень захисту, оскільки гарантує, що підписаний документ не зазнав змін після підписання. Однак, у разі втрати ключа або його компрометації, це може поставити під сумнів автентичність підпису.

УЕП має більшу юридичну силу порівняно з простим підписом, але, на відміну від кваліфікованого, не забезпечує повної юридичної гарантії у разі суперечок[29].

2.3.3 Кваліфікований електронний підпис (КЕП)

Кваліфікований підпис є найбільш надійним і забезпечує найвищий рівень безпеки. Він ґрунтується на використанні сертифікованих криптографічних засобів і електронних сертифікатів, які видаються акредитованими центрами сертифікації[29].

КЕП гарантує не лише автентичність підписанта, але й цілісність документа, забезпечуючи найвищий рівень захисту від фальсифікацій.

Для створення КЕП використовуються апаратні засоби (наприклад, токени або смарт-карти), що забезпечують захист приватного ключа від несанкціонованого доступу.

Кваліфікований підпис є юридично обов'язковим в багатьох країнах для підписання офіційних документів та має таку ж юридичну силу, як власноручний підпис[29].

2.3.4 Огляд національних та міжнародних стандартів, зокрема PAdES-B-B

Для забезпечення автентичності, цілісності та юридичної сили ЕЦП існує ціла система стандартів, як на національному, так і на міжнародному рівні. Одним із важливих стандартів є PAdES (PDF Advanced Electronic Signatures), зокрема його розширена версія PAdES-B-B.

PAdES (PDF Advanced Electronic Signatures) – це набір стандартів, розроблений для підписання PDF-документів за допомогою ЕЦП. Вони

гарантують високий рівень безпеки і є спеціально розроблені для інтеграції з документами в форматі PDF[26].

PAdES-B-B (Long-Term Validation) – це розширення стандарту PAdES, яке призначене для забезпечення довготривалої валідності підписів. Включає механізми, які дозволяють перевірити підпис навіть через багато років після його накладення, зберігаючи дані про підпис і ключі на довгий час[26].

Стандарт PAdES-B-B включає в себе:

- Архівацію метаданих підпису (наприклад, інформації про сертифікат підписанта і підтвердження від центру сертифікації).
- Відображення сертифіката в документі, що дозволяє перевірити підпис навіть після того, як сертифікат може бути скасований або прострочений.
- Довготривалу перевірку підпису завдяки додатковим метаданим, що зберігаються в документі.

Національні стандарти для використання ЕЦП в Україні зокрема визначають, що кваліфіковані підписи повинні відповідати вимогам, що описані в українському законодавстві, зокрема Законі України про електронний цифровий підпис та стандартам електронного підпису, затвердженим акредитованими органами сертифікації. В Україні для створення КЕП використовуються сертифікати, видані центрами сертифікації, акредитованими в системі електронних підписів[23].

2.3.5 Особливості використання ЕЦП у додатку "Дія.Підпис"

Дія.Підпис – це платформа, розроблена урядом України для забезпечення простоти і безпеки використання електронних підписів для громадян. Вона дозволяє підписувати документи через мобільні пристрої за допомогою цифрового підпису (див. рис. 2.5[29])

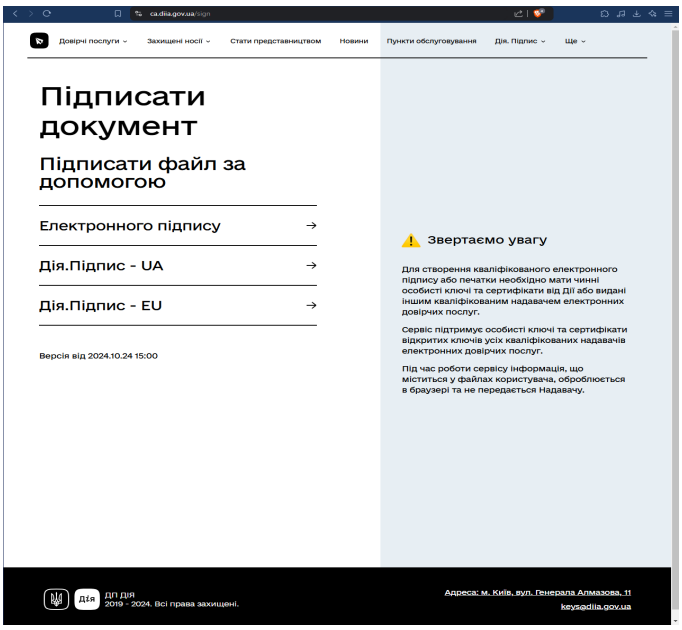


Рис. 2.5. Інтерфейс веб-сторінки «Дія» з розділом послуги підпису документа

Особливості використання ЕЦП у додатку "Дія.Підпис":

- Інтеграція з єдиною системою: Через "Дія.Підпис" користувачі можуть підписувати документи, що мають юридичну силу, без потреби використовувати спеціалізоване програмне забезпечення або фізичні носії (наприклад, токени чи смарт-карти)[29].
- Використання кваліфікованого підпису: Для створення підпису в додатку користувачі використовують кваліфіковані електронні сертифікати, які видані акредитованими центрами сертифікації. Це гарантує високий рівень безпеки і юридичну силу підписаних документів.
- Зручність і доступність: Додаток дозволяє громадянам підписувати документи з будь-якого місця, що полегшує обіг електронних документів в державних органах і організаціях[29].
- Підпис без фізичних носіїв: У "Дія.Підпис" підписування документа можна здійснювати без необхідності мати фізичні носії (наприклад, токени), що робить процес значно зручнішим і доступнішим.

- Автоматичне оновлення сертифікатів: Додаток дозволяє автоматично оновлювати сертифікати підпису, що забезпечує зручність у використанні та підвищує безпеку документів[29].

Завдяки таким функціям "Дія.Підпис" суттєво спрощує процес електронного підписання документів для громадян України, створюючи умови для розвитку електронного документообігу в країні.

2.4 Визначення парсингу та його роль у перевірці документації

2.4.1 Парсинг

Парсинг (від англ. parsing) – це процес аналізу та інтерпретації даних, що зберігаються у певному форматі, з метою їх обробки, структурування і подальшого використання. У контексті цифрових документів парсинг включає витягування, аналіз та організацію даних з таких форматів, як PDF, HTML, XML, JSON тощо. Він дозволяє зчитувати, структурувати та аналізувати вміст документів, що робить його незамінним інструментом для роботи з електронною документацією (див. рис. 2.6[30]).

Рис. 2.6. Ілюстрація парсера, як «робота, що категоризує інформацію, яку збирає»

Парсинг є важливим інструментом при перевірці документації, оскільки він дозволяє автоматично обробляти великі обсяги інформації, витягувати важливі дані (наприклад, текстові блоки, метадані, зображення) з документів, щоб забезпечити точність перевірки автентичності та цілісності документів[31].

Завдяки парсингу можна:

- Витягнути текстовий вміст документа для подальшого аналізу на зміни.
- Перевірити наявність метаданих, які можуть допомогти в ідентифікації автора документа, дати підпису чи модифікації.
- Проаналізувати структуру документа, що дозволяє виявити зміни в форматуванні, зображеннях чи інших елементах документа, які можуть свідчити про його зміну або підробку.

Парсинг особливо важливий у таких сферах, як електронний документообіг, фінансові звіти, юридичні документи, де точність і достовірність інформації мають критичне значення. В умовах цифрової трансформації та зростаючого обсягу цифрових документів, парсинг допомагає знижувати людський фактор при перевірці та підвищує ефективність процесів верифікації[31].

2.4.2 Використання парсингу для порівняння оригінальних і змінених версій PDF-документів

У контексті перевірки автентичності та цілісності документів парсинг є одним з основних методів для порівняння оригінальних і змінених версій PDF-документів. Використання парсингу в такій ситуації дозволяє виявити навіть незначні зміни в документі, які можуть бути пропущені при традиційному ручному перегляді.

2.4.3 Структурний парсинг PDF-документів

PDF – це складний формат, що підтримує різноманітні елементи, такі як текст, зображення, таблиці, метадані, а також елементи форматування (шрифти,

розміри, кольори). Парсинг PDF дозволяє точно розібрати ці елементи, що дає змогу ефективно виявити зміни в структурі документа[31].

Для парсингу PDF-документів використовуються спеціальні бібліотеки і інструменти, такі як PyPDF2, pdfminer, PyMuPDF (Fitz), pdfrw в Python. Ці інструменти дозволяють витягнути текст, зображення, метадані та інші елементи документа[31].

2.4.4 Текстовий парсинг

Один із основних аспектів парсингу PDF-документів – це витягнення тексту. Це дозволяє порівняти оригінальний текст документа з його зміненою версією. При цьому парсинг також дозволяє зберігати структуру документа, включаючи абзаци, заголовки, таблиці, що дуже важливо при перевірці юридичних або фінансових документів, де кожне слово може бути критично важливим[31].

Витягнення тексту через парсинг дає змогу провести порівняння двох версій документа та виявити навіть найменші відмінності – зміни в словах, додавання нових абзаців, переписування чи видалення частин тексту. Ось деякі інструменти, які допомагають витягти текст з PDF:

- pdfminer – дозволяє проводити більш точний парсинг тексту з урахуванням форматування, таких як відступи і розміри шрифтів[32].
- PyPDF2 – одна з найбільш використовуваних бібліотек для роботи з PDF-документами, яка надає можливість витягувати текст, метадані, зображення, а також здійснювати інші операції з документом[32].

2.4.5 Метадані та інформація про підпис

Іншими важливими даними, які можна отримати за допомогою парсингу PDF-документів, є метадані, які часто використовуються для зберігання важливої інформації про документ, такої як автор, дата створення, дата зміни,

програмне забезпечення, що використовувалося для створення документа, а також інші технічні дані.

Наприклад, за допомогою парсингу можна отримати метадані про цифровий підпис в документі, якщо такий є. Вони можуть включати інформацію про підписанта, сертифікат, час підписання і т.д. Це дозволяє визначити, чи був документ змінений після накладення підпису, що важливо для перевірки його автентичності[32].

2.4.6 Порівняння версій документів

Порівняння оригінального і зміненого документа є ключовим етапом в процесі виявлення фальсифікацій. Застосування парсингу дає змогу здійснити порівняння двох версій документа з урахуванням не лише тексту, а й інших елементів, таких як форматування, зображення, графіки[32].

Спеціалізовані інструменти для порівняння PDF-документів на основі парсингу можуть підсвічувати зміни в тексті, зображеннях, таблицях, а також виявляти невидимі зміни, наприклад, редагування метаданих. Такий підхід дозволяє легко виявляти навіть найменші зміни в документі.

Одним із популярних інструментів для порівняння версій документів є DiffPDF, який дозволяє порівнювати два PDF-документи, знаходячи різниці у змісту, розміщенні елементів та інше[32].

2.4.7 Автоматизація процесу

Використання парсингу для порівняння документів може бути автоматизоване за допомогою Python-скриптів або спеціальних програмних рішень. Це значно спрощує і пришвидшує процес перевірки великих обсягів документів, дозволяючи автоматично виявляти всі зміни та порушення.

Автоматизація процесу парсингу та порівняння документації дозволяє зменшити можливості людських помилок, підвищуючи точність виявлення фальсифікацій або некоректних змін[33].

2.4.8 Виявлення підробок

За допомогою парсингу можна також виявляти підробки в PDF-документах, наприклад, якщо документ містить невидимі зміни або заміни шрифтів, які не можна легко помітити при звичайному перегляді. Парсинг дозволяє аналізувати ці зміни на рівні тексту і структури документа, що може допомогти виявити фальсифікацію[33].

2.5 Методи перевірки автентичності документів

Перевірка автентичності документів – це важливий процес у забезпеченні їх цілісності та достовірності. В умовах цифровізації та зростання використання електронних документів важливість цього процесу зростає. Методи перевірки автентичності документів включають різноманітні технології, серед яких хешування, водяні знаки, цифрові підписи та порівняння версій документів. У цьому розділі особливу увагу буде приділено хешуванню та водяним знакам, а також порівнянню ефективності різних методів перевірки автентичності[33].

2.5.1 Хешування: визначення, алгоритми (SHA-256, MD5) та роль у перевірці цілісності

Хешування – це процес перетворення вхідних даних (тексту, файлу чи іншого об’єкта) у фіксовану рядок довжиною, яка називається хеш-кодом або хеш-сумою (див. рис. 2.7[34]). Хешування забезпечує цілісність даних, оскільки навіть невелика зміна в даних призведе до значної зміни хеш-коду. Для документів хешування є важливим інструментом для перевірки їх цілісності і автентичності, оскільки цей метод дозволяє швидко виявити будь-які несанкціоновані зміни в документі.

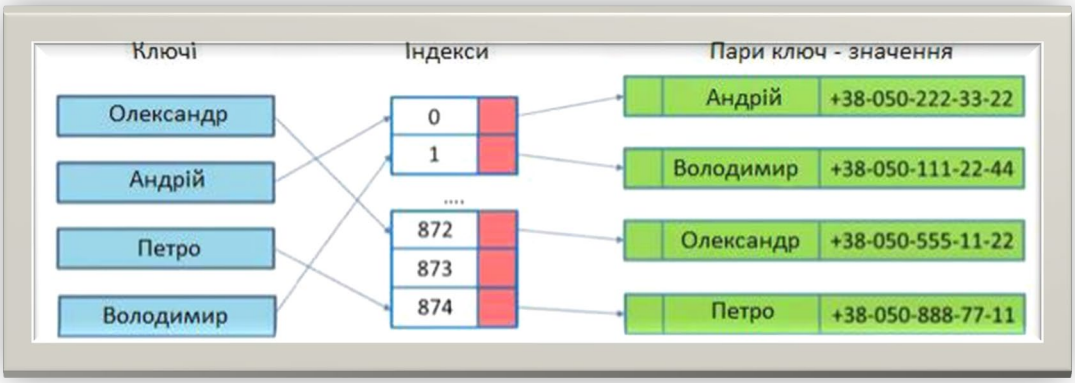


Рис. 2.7. Приклад хеш-таблиці

2.5.2 Алгоритми хешування

Існує кілька поширених алгоритмів хешування, кожен із яких має свої переваги та недоліки:

- MD5 (Message-Digest Algorithm 5): MD5 – один із найстаріших і найвідоміших алгоритмів хешування. Він створює 128-бітну (16-байтову) хеш-суму документа. Однак MD5 є застарілим і має відомі уразливості: існують методи знаходження колізій (двох різних даних, які мають однакову хеш-суму). Через це MD5 не рекомендується для використання в критичних сферах, де необхідна висока безпека[35].
- SHA-256 (Secure Hash Algorithm 256-bit): SHA-256 є частиною більшої сім'ї алгоритмів SHA-2 і створює 256-бітний (32-байтовий) хеш. Це більш безпечний і надійний алгоритм у порівнянні з MD5. Він забезпечує високу стійкість до колізій і широко використовується в сучасних системах для перевірки цілісності документів, наприклад, в цифрових підписах, криптовалютних транзакціях та багатьох інших застосунках[35].

2.5.3 Роль хешування у перевірці цілісності

Хешування є основним методом для перевірки цілісності документів. Кожен документ має свою унікальну хеш-суму, яка генерується на основі вмісту документа. Якщо навіть одна буква в документі буде змінена, його хеш-сума

значно зміниться. Тому перевірка хеш-суми дозволяє встановити, чи було змінено документ після його створення чи підписання[35].

Процес перевірки:

- Спочатку обчислюється хеш-сума оригінального документа.
- При перевірці документа на цілісність, знову обчислюється хеш-сума поточної версії документа.
- Якщо хеші збігаються, документ не зазнав змін. Якщо ж хеші не збігаються, це свідчить про зміну документа.

2.5.4 Обмеження хешування

Хешування є ефективним методом для перевірки цілісності, однак воно не дозволяє підтвердити автентичність документа. Хеш-сума лише показує, що зміни в документі мали місце, але не дає інформації про те, хто ці зміни зробив і коли[35].

2.5.5 Водяні знаки: визначення, цифрові та видимі водяні знаки, переваги і недоліки

Водяний знак – це елемент захисту, який вбудовується в документ для підтвердження його автентичності. Водяні знаки використовуються для захисту від підробок, а також для підтвердження прав власності на документ[36].

2.5.6 Цифрові водяні знаки

Цифрові водяні знаки – це невидимі або майже невидимі елементи, які вбудовуються в цифрові документи. Вони можуть бути інкорпоровані в текст або зображення документа і зазвичай використовуються для підтвердження прав власності або як частина системи захисту від несанкціонованих змін[36].

Цифровий водяний знак є даними, що додаються до документа без його видимого змінення, зазвичай у вигляді коду або метаданих, що зберігаються в

документах у такій формі, що їх не можна видалити або змінити без порушення цілісності документа.

Цифрові водяні знаки можуть бути використані в поєднанні з цифровими підписами, що дозволяє підтвердити не лише авторство документа, але й його цілісність і зміст[36].

Переваги цифрових водяних знаків:

- Невидимість. Цифрові водяні знаки не заважають основному вмісту документа і не впливають на його зовнішній вигляд.
- Тривала стійкість. Вони важко видаляються або змінюються без втрати цілісності документа.
- Захист від фальсифікації. Вони можуть бути використані для підтвердження авторства документа[36].

Недоліки цифрових водяних знаків:

- Можуть бути технічно складними у створенні та виявленні.
- Не завжди надійно захищають від змін у вмісті документа, якщо документ зазнає значних маніпуляцій[36].

2.5.7 Видимі водяні знаки

Видимі водяні знаки – це зображення або текст, які накладаються на документ і є видимими для користувачів. Ці знаки зазвичай використовуються для позначення авторських прав, власності чи для індикації того, що документ є конфіденційним (див. рис. 2.8[37]).

Рис. 2.8. Приклад комбінації різних типів захисту на прикладі купюри в 1000 гривень

Переваги видимих водяних знаків:

- Легко розпізнаються користувачами, що додає додатковий рівень захисту від несанкціонованого використання документів.
- Їх важче видалити, ніж цифрові водяні знаки, якщо вони були застосовані належним чином[36].

Недоліки видимих водяних знаків:

- Можуть заважати користувачам при перегляді документа.
- Легко зіпсувати або замаскувати за допомогою редагування документів.
- Порівняння ефективності методів перевірки автентичності[36].

У процесі перевірки автентичності документів важливо використовувати різноманітні методи, оскільки кожен із них має свої переваги та недоліки (див. табл. 2.2):

Таблиця 2.2

Порівняння деяких основних методів перевірки автентичності

Метод	Переваги	Недоліки
Хешування	Простота та швидкість перевірки	Не дає інформації про підписанта або автора
	Широке використання алгоритмів, таких як SHA-256	Не дозволяє виявити, хто вніс зміни в документ
Цифрові водяні знаки	Невидимість і надійність	Технічна складність у створенні та виявленні
	Важко видаляються без зміни документа	Не захищають від усіх видів маніпуляцій з документом
Видимі водяні знаки	Легко помітні та розпізнавані	Можуть заважати перегляду документа
	Стійкість до маніпуляцій	Можуть бути замасковані або зіпсовані

2.6 Алгоритми для створення надійного цифрового підпису

Цифровий підпис створюється шляхом застосування алгоритмів шифрування до документів, що мають цифрову форму. Ці алгоритми є основою для захисту електронних документів і підтвердження їх автентичності. Основні етапи створення цифрового підпису включають хешування документа, а також шифрування хешу за допомогою приватного ключа. Для цього використовуються асиметричні криптографічні алгоритми[38].

2.6.1 Процес створення цифрового підпису

Його можна описати такими кроками:

- Спочатку від документа генерується хеш-сума.

- Потім ця хеш-сума шифрується за допомогою приватного ключа підписанта, створюючи цифровий підпис.
- Отриманий підпис може бути перевірений за допомогою відкритого ключа підписанта, що дозволяє будь-кому підтвердити автентичність підпису, порівнюючи хеш-суму оригіналу документа з тим, що вказано в підписі[38].

2.6.2 Алгоритми для цифрового підпису

Існує кілька ключових алгоритмів шифрування, які застосовуються для створення цифрових підписів:

- RSA (Rivest–Shamir–Adleman): RSA є одним з найпоширеніших алгоритмів асиметричного шифрування. Він заснований на математичних принципах факторизації великих чисел. RSA використовується для створення цифрових підписів, а також для шифрування даних. Алгоритм забезпечує високий рівень безпеки, особливо якщо використовуються великі ключі (2048 біт і більше)[38].

Переваги RSA: широко підтримується в більшості криптографічних систем, висока стійкість до атак при використанні великих ключів.

Недоліком RSA є часова складність операцій з великими ключами може бути значною, що робить його не так швидким для деяких застосувань.

- ECDSA (Elliptic Curve Digital Signature Algorithm): ECDSA використовує еліптичні криві, що дозволяє зберігати такий самий рівень безпеки, як RSA, але з меншими розмірами ключів. Це робить його більш ефективним в плані швидкості та обчислювальних ресурсів[38].

Переваги ECDSA: менші ключі при тому ж рівні безпеки, швидша робота, що дозволяє використовувати цей алгоритм на пристроях з обмеженими ресурсами.

Недоліком ECDSA є те, що він потребує більш складних математичних обчислень, що може створити певні труднощі для розробників при впровадженні.

- EdDSA (Edwards-curve Digital Signature Algorithm): Це новіший алгоритм, що ґрунтується на еліптичних кривих, подібно до ECDSA, але має поліпшену стійкість до атак. Він також забезпечує високу швидкість генерації підписів і перевірки[38].

Переваги EdDSA: висока швидкість та ефективність, стійкість до атак, зокрема до атак, пов'язаних з випадковими числами.

Недоліком EdDSA є те, що він може бути не так широко підтримуваний у старих системах порівняно з RSA чи ECDSA.

2.6.3 Порівняння використання шифрування та ЕЦП у різних сферах

Шифрування та електронний цифровий підпис (ЕЦП) використовуються в різних сферах діяльності для забезпечення безпеки даних. Вибір конкретного методу шифрування чи підпису залежить від вимог до безпеки, швидкості та ресурсоємності процесу. У різних сферах застосування ці методи мають свої особливості[38].

2.6.4 Банківські послуги

У банківському секторі шифрування та цифрові підписи використовуються для забезпечення безпеки фінансових транзакцій, авторизації платіжних систем і захисту даних клієнтів[39].

Шифрування

Банки часто використовують шифрування для захисту даних клієнтів під час передачі через Інтернет (наприклад, використовуючи SSL/TLS). Ці протоколи захищають конфіденційність та цілісність даних, що передаються між клієнтами та банками[39].

Цифровий підпис

В банківських операціях цифровий підпис використовується для підтвердження автентичності угод, заявок, контрактів тощо. Наприклад, цифровий підпис може бути використаний для підтвердження транзакції або надання доступу до фінансових інструментів (див. рис. 2.9[40]).

Рис. 2.9. Порядок дій для отримання електронного підпису фізичною особою

2.6.5 Державні послуги

В державних послугах цифровий підпис є невід'ємною частиною для забезпечення легітимності та автентичності офіційних документів, таких як податкові декларації, паспорти громадян тощо. В Україні для державних послуг використовуються системи, які інтегрують цифрові підписи та інші механізми електронного документообігу, такі як Дія.Підпис[41].

Шифрування в державних послугах гарантує безпечну передачу особистих даних між державними установами і громадянами.

Цифрові підписи забезпечують юридичну силу електронних документів, що мають значення в судових спорах, податкових питаннях, в реєстрації прав на майно тощо[41].

2.6.6 Комерційний сектор (онлайн-торгівля, контракти)

У комерційних і торгових угодах цифрові підписи та шифрування застосовуються для укладання договорів, підтвердження угод та забезпечення безпеки онлайн-платежів (див. рис. 2.10[40]).

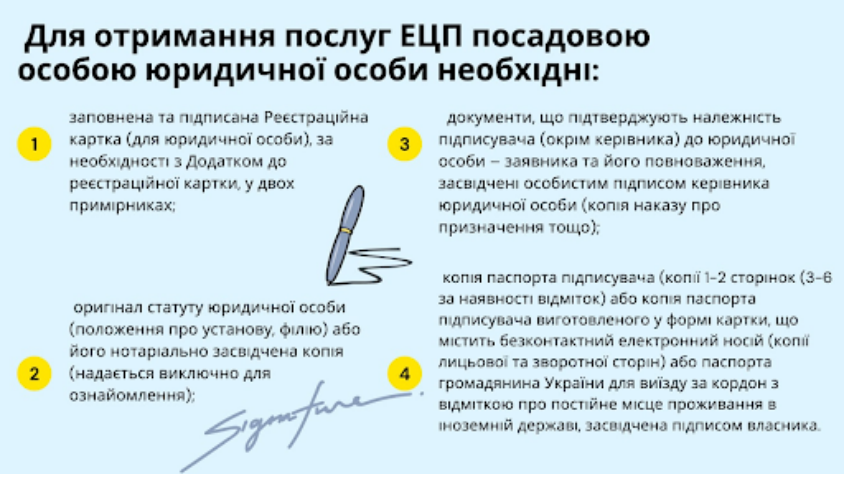


Рис. 2.10. Порядок дій для отримання електронного підпису юридичною особою

Шифрування забезпечує захист особистої інформації та платіжних реквізитів під час онлайн-покупок або підписання договорів в електронній формі.

Цифровий підпис забезпечує автентичність документів, таких як контракти, угоди про надання послуг тощо. Наприклад, використання ЕЦП на договорі дозволяє легко і швидко укласти угоду, зберігаючи її юридичну силу[41].

2.7 Висновки за розділом 2

Захист цифрових документів є критично важливою складовою в сучасному інформаційному середовищі, оскільки він забезпечує не тільки безпеку і цілісність переданої інформації, а й надійний механізм для підтвердження автентичності даних, що особливо важливо в умовах сучасного цифрового документообігу. В результаті проведеного дослідження було розглянуто широкий спектр методів захисту цифрових документів, серед яких найбільш важливими є методи шифрування, електронний цифровий підпис (ЕЦП), парсинг даних, а також алгоритми хешування і водяні знаки.

Перш за все, для забезпечення автентичності та цілісності документів в цифровому середовищі застосовуються різноманітні методи шифрування, включаючи як симетричне, так і асиметричне шифрування, що дозволяє

ефективно захищати дані від несанкціонованого доступу і зміни. Особливу увагу варто приділити алгоритмам, таким як RSA, ECDSA і EdDSA, які гарантують надійність цифрових підписів і забезпечують високий рівень безпеки завдяки використанню приватних і публічних ключів. Крім того, було підтверджено, що в різних сферах застосування (банківський сектор, державні послуги, комерційні угоди) шифрування і цифрові підписи виконують специфічні функції, що дозволяють зберігати конфіденційність і автентичність документів.

Цифровий підпис (ЕЦП) є важливим інструментом для забезпечення юридичної сили електронних документів. Особливо варто відзначити розвиток національних стандартів, таких як PAdES-B-B, і їх інтеграцію в популярні додатки, зокрема в "Дія.Підпис", що дозволяє громадянам підписувати документи електронним підписом без фізичної присутності в державних установах. Це свідчить про значний прогрес у використанні цифрових підписів для громадянських та правових відносин.

Методи перевірки автентичності документів, такі як хешування та водяні знаки, є важливими інструментами для визначення цілісності і оригінальності цифрових документів. Хешування дозволяє створювати унікальні ідентифікатори для файлів, що можуть бути використані для порівняння і виявлення змін в документі, а водяні знаки є додатковим методом для підтвердження авторства і цілісності документів. Хоча водяні знаки можуть бути видимими або цифровими, їх ефективність залежить від правильного використання та рівня інтеграції в документообіг.

Особливо важливою є роль парсингу даних для порівняння оригінальних і змінених версій документів. Використання парсингу для автоматизованої перевірки текстового вмісту документа дозволяє значно знизити ризик людської помилки і підвищити точність перевірок. Технологія парсингу також інтегрується з іншими методами перевірки, що дозволяє створювати більш комплексні системи для перевірки автентичності цифрових документів.

Загалом, можна зробити висновок, що в сучасних умовах для забезпечення надійної автентичності та цілісності цифрових документів необхідно

використовувати комплексний підхід, який включає шифрування, електронний підпис, хешування, водяні знаки та парсинг даних. Технології швидко розвиваються, і з їх допомогою можна не тільки забезпечити безпеку даних, але й значно покращити ефективність і зручність електронного документообігу в різних сферах діяльності. В майбутньому варто очікувати подальшого удосконалення цих технологій, а також інтеграції нових методів для ще більш високого рівня захисту та перевірки автентичності цифрових документів.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ НА АВТЕНТИЧНІСТЬ

3.1 Графічний інтерфейс користувача (GUI)

Інтерфейс програми побудовано на основі модуля Tkinter – стандартної бібліотеки Python для створення графічних інтерфейсів. Tkinter базується на інтерфейсі Tk GUI та забезпечує розробку GUI з використанням візуальних компонентів, таких як кнопки, поля введення тексту, ярлики, текстові вікна та повідомлення, що значно полегшує взаємодію користувача з додатком. Завдяки Tkinter можна створювати кросплатформні програми з базовими графічними елементами, які добре інтегруються з іншими модулями Python. Tkinter особливо цінний для створення програм середнього рівня складності, де не потрібні високі вимоги до анімації чи візуальних ефектів. Інтерфейс організовано наступним чином:

- Вибір файлу – кнопка «Обрати файл», яка дозволяє користувачу відкрити файловий провідник, знайти та обрати локальний PDF-файл, що виступає оригіналом для порівняння.
- Поле введення URL – текстове поле для введення URL-адреси Google Sites, де розміщено підписаний документ. Цей елемент дає змогу користувачу завантажити документ безпосередньо з веб-ресурсу.

3.1.1 Кнопки для основних дій:

- «Порівняти» – кнопка, яка запускає процес порівняння текстів між оригінальним і підписаним документом.
- «Перевірити підпис» – кнопка, яка ініціює завантаження PDF-документа за введеною URL-адресою, зберігає його локально і перенаправляє користувача на сторінку онлайн-перевірки підпису.

- «Експортувати звіт» – кнопка, що зберігає результати порівняння як звіт у форматі DOCX.

3.1.2 Завантаження оригінального та підписаного документів

Для порівняння текстів обох документів, програма потребує завантаження оригінального PDF-документу з локального комп'ютера та підписаного документа за URL-адресою Google Sites.

Оригінальний документ завантажується через інтерфейс Tkinter (вибір файлу), після чого за допомогою бібліотеки pdfplumber здійснюється зчитування тексту зі сторінок PDF-документа.

pdfplumber – це бібліотека Python для роботи з PDF-документами. На відміну від більшості інструментів для роботи з текстом, формат PDF відомий своєю структурованістю та здатністю зберігати зображення, таблиці, шрифти та інші елементи форматування. pdfplumber надає доступ до тексту на рівні окремих сторінок, забезпечуючи можливість зчитувати текст, графічні елементи й таблиці, а також зберігати текстову інформацію у вигляді простого тексту або структури HTML. Завдяки цьому модулю програма може витягати текст з PDF-файлів для подальшого аналізу та порівняння[32].

Підписаний документ завантажується через URL, введений користувачем. Для цього використовуються:

- Бібліотека requests, яка дозволяє виконувати HTTP-запити, отримувати HTML-контент з веб-сторінок та працювати з ним у форматі HTTP-відповідей, що є основою для інтеграції з веб-сервісами.
- Бібліотека BeautifulSoup для розбору HTML-документів, яка створює парсинг HTML-коду у вигляді дерева. Це особливо корисно для вилучення посилань і зображень.

У контексті програми дані з Google Sites завантажуються за допомогою Requests, а BeautifulSoup забезпечує аналіз HTML для витягнення PDF-посилань[33].

3.1.3 Порівняння документів

Після отримання тексту з обох документів програма проводить порівняння тексту рядок за рядком.

diff_match_patch – це спеціалізований алгоритм, створений Google для порівняння та узгодження текстів, який реалізовано як бібліотеку Python. Алгоритм здійснює детальний аналіз змін між двома текстовими масивами шляхом визначення доданих, видалених і незмінних частин тексту. Алгоритм включає кілька методів оптимізації, таких як семантична та лексична очистка, що дозволяє отримувати більш точні результати порівняння. Використання diff_match_patch надає можливість програмі автоматично визначати різницю між оригінальним документом і зміненим, ідентифікуючи модифікації на рівні символів та рядків[41].

Цей процес включає:

Обробку різниць між текстами – за допомогою diff_match_patch визначаються три основні типи змін:

- Не змінено – рядки, які однакові в обох документах.
- Додано – текст, який з’явився у підписаному документі, але відсутній в оригіналі.
- Видалено – текст, який є в оригіналі, але був видалений у підписаному документі.

Очищення попередніх результатів – перед аналізом нових текстів програма очищує поле виведення результатів, щоб відобразити тільки актуальні дані.

Виведення результатів з кольоровим кодуванням – програма використовує метод tag_configure для налаштування кольорів, що відображаються на кожному типі зміни:

- Зелений фон для незміненого тексту.
- Червоний фон для видаленого тексту.
- Помаранчевий фон для доданого тексту.

Це полегшує швидке виявлення різниць у тексті навіть при великому обсязі даних.

Статистика та висновок – програма підраховує кількість доданих і видалених рядків, формуючи підсумкове повідомлення про наявність або відсутність змін у підписаному документі.

3.1.4 Відображення результатів у вікні GUI

Поле виведення результатів створено на основі ScrolledText з використанням кольорових тегів для виділення змін. Результати порівняння подаються у форматі, що відразу вказує на:

- Текст, що не змінювався – виділяється зеленим.
- Новий текст, який було додано – позначається помаранчевим кольором.
- Видалений текст з оригінального документа – показується червоним кольором.

3.1.5 Перевірка цифрового підпису

Цифровий підпис – це криптографічна технологія, що використовується для автентифікації документа та забезпечення його цілісності. У випадку PDF, можна перевірити чи дійсні сертифікати цифрового підпису, який нанесений на документ. Програма інтегрована з онлайн-сервісом перевірки підписів, який відкривається через модуль webbrowser. Зокрема, використовується сервіс ЦЗО для перевірки українських електронних підписів, що забезпечує доступ до інструменту перевірки підпису в браузері.

Перевірка підпису включає кілька кроків:

- Завантаження документа за URL – підписаний документ, отриманий з Google Sites, зберігається на локальному комп'ютері для перевірки.
- Відкриття сторінки перевірки підпису – програма перенаправляє користувача на сайт ЦЗО (Центрального засвідчувального органу) за допомогою модуля webbrowser. Цей сайт дозволяє користувачеві

завантажити збережений PDF-документ і перевірити його цифровий підпис на справжність[42].

3.1.6 Експорт звіту

Для створення текстових звітів використовується бібліотека python-docx, яка забезпечує можливість формування документів у форматі Microsoft Word (DOCX). Формат DOCX дозволяє зберігати текстові дані, а також додавати стилі, заголовки та інші елементи форматування, що підвищує зручність для користувачів, які мають потребу зберігати результати аналізу для подальшого використання. У програмі автоматично формується звіт про зміни в документі, який включає додані, видалені та незмінні частини тексту, що дозволяє користувачам переглядати та архівувати результати перевірки.

Кроки для створення звіту:

- Отримання результатів порівняння – усі результати порівняння (додавання, видалення, незмінені частини) збираються у текстовий формат.
- Створення DOCX-файлу – використовується python-docx для створення нової структури документу, в якій вказується:
- Заголовок звіту (наприклад, «Звіт про порівняння документів»).
- Кольорове кодування тексту – важливі зміни відзначаються тегами, схожими на виведення в GUI.
- Збереження файлу – користувачеві пропонується вибрати місце збереження файлу через діалогове вікно.

3.1.7 Обробка помилок

Програма передбачає багаторівневу систему обробки винятків для обробки потенційних помилок. Наприклад, при завантаженні файлу, виконанні HTTP-запитів або витяганні тексту з PDF можуть виникати помилки, які обробляються з використанням вбудованої системи обробки винятків. Обробка винятків

забезпечує стабільність програми та гарантує, що користувач отримає повідомлення з описом помилки та можливими шляхами її вирішення.

Для забезпечення коректної роботи програма враховує можливі помилки та винятки, що можуть виникати при використанні:

- Відсутність вибраного оригінального файлу – програма виводить повідомлення про необхідність обрати файл, якщо користувач натискає «Порівняти», не вказавши оригінальний документ.
- Помилки завантаження документа з URL – у разі невірної URL-адреси або недоступності сторінки Google Sites програма виводить повідомлення про помилку та повертає користувача до введення правильного посилання.
- Неможливість перевірки цифрового підпису – програма показує попередження, якщо не вдається автоматично відкрити сторінку ЦЗО.
- Помилки при створенні та збереженні звіту – якщо під час створення DOCX-файлу виникає помилка (наприклад, обмеження доступу або неправильний формат), програма виводить повідомлення з поясненням та інструкціями.

3.1.8 Архітектура та класова структура

Програма побудована на основі об’єктно-орієнтованого підходу, який підвищує її гнучкість, масштабованість та зручність у підтримці. Класи DocumentComparisonApp, PDFHandler та DocumentComparator виконують різні завдання і відповідають за різні аспекти програми:

- DocumentComparisonApp – головний клас програми, який відповідає за створення інтерфейсу та взаємодію з користувачем. Цей клас також викликає інші класи для виконання конкретних функцій, таких як завантаження PDF або порівняння тексту[43].
- PDFHandler – клас, що виконує завантаження PDF-файлів з локального комп’ютера та URL, а також забезпечує вилучення тексту з PDF-файлів[43].

- DocumentComparator – клас, що виконує порівняння текстів між двома документами. Він також генерує результат у форматі, що легко читається та відображає відмінності з використанням кольорових тегів[43].

Ця програма є цілісним рішенням для аналізу змін у PDF-документах, перевірки підписів та створення звітів, що може бути корисним у багатьох сферах – від юридичної до адміністративної діяльності.

3.1.9 Практичне застосування та цінність програми

Програма орієнтована на юридичних фахівців, архіваріусів, офісних працівників та інших користувачів, які потребують швидкого порівняння версій документів, перевірки цілісності підписів та збереження результатів у вигляді формалізованих звітів. Завдяки інтеграції з національними сервісами перевірки електронних підписів та можливості детального порівняння текстових змін, цей інструмент є надійним рішенням для автоматизації рутинних задач, пов'язаних із перевіркою документів.

3.2 Покроковий сценарій використання програмного модуля

3.2.1 Ознайомлення з блок-схемою програмного модуля

Блок схема покроково описує сценарій використання програмного модуля (див. рис. 3.1), де Відправника позначено зеленим кольором (далі, Власник), а рожевим кольором позначено Отримувача (далі, Підрядник).

Програма розпочинається від моменту «Старт» і рухається проти часової стрілки у напрямку «Підготовки документа до відправлення». Зупинимося детальніше на цьому кроці.

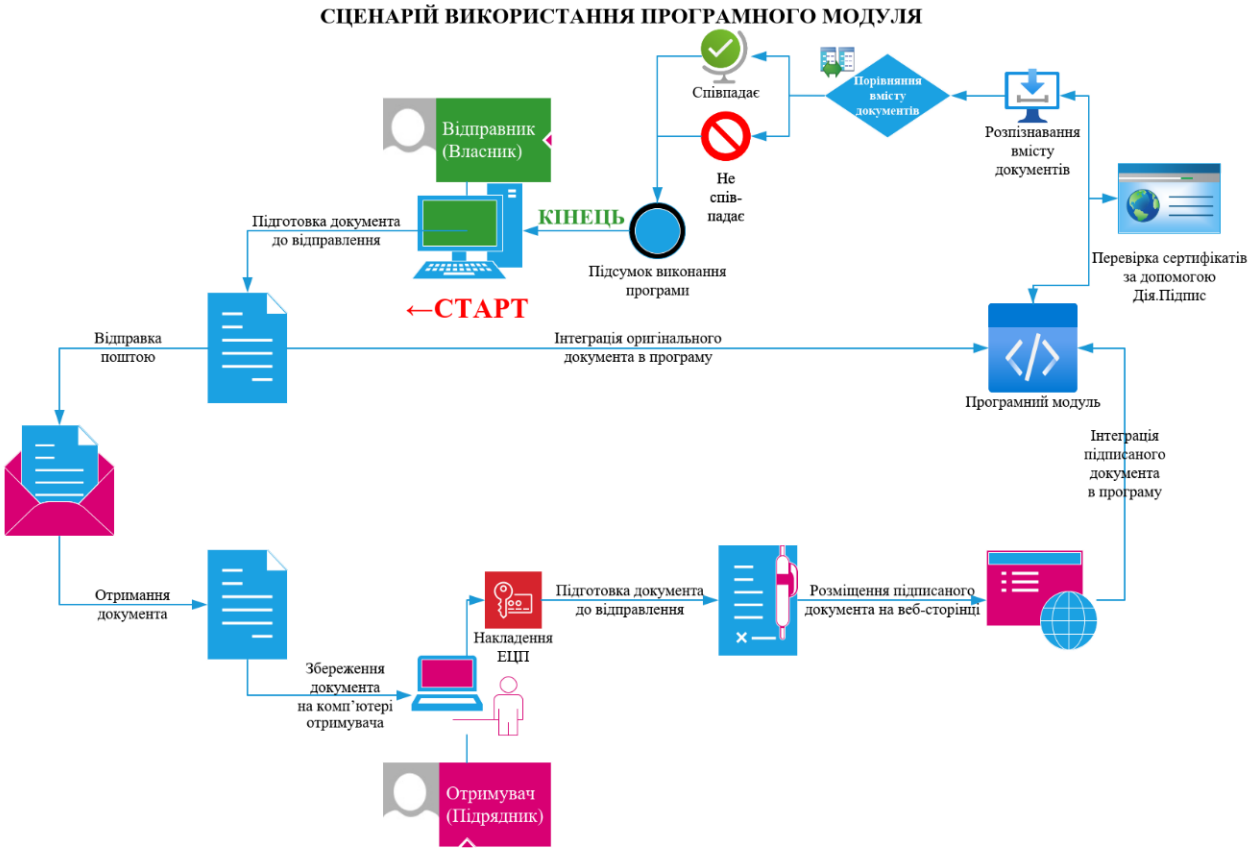


Рис. 3.1 Робота алгоритму програмного модуля на перевірку автентичності електронного документа

3.2.2 Підготовка документа до відправлення

Власник підготовлює заповнену копію документа (див. рис. 3.2), яка буде вже готовою для ознайомлення і підписання Підрядником. Якщо раніше цей бланк був у форматі .docx, то на цьому етапі можливий експорт у PDF, для можливості підписання Підрядником, використовуючи сервіси Дія.Підпис (див. рис. 3.3).

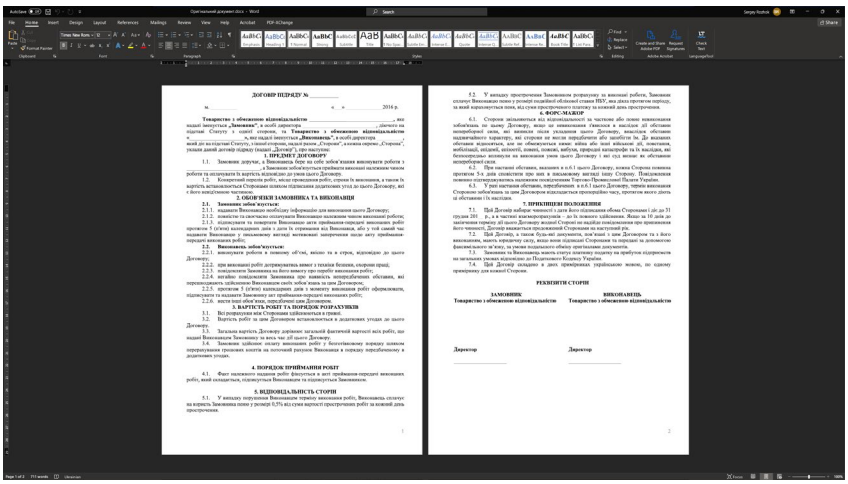


Рис. 3.2 Приклад документа, який будемо використовувати як оригінальний

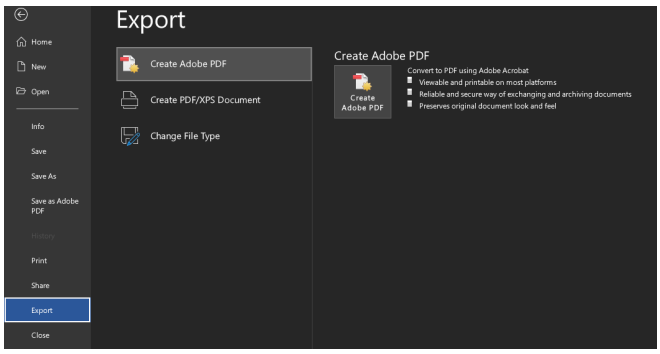


Рис. 3.3 Функція експорту в PDF на основі програми Microsoft Office Word

3.2.3 Відправка поштою оригінального документа від Власника до Підрядника

На цьому кроці для взаємодії та обміну документом було використано поштовий сервіс з приміткою «Конфідеційно» (див. Рис 3.4).

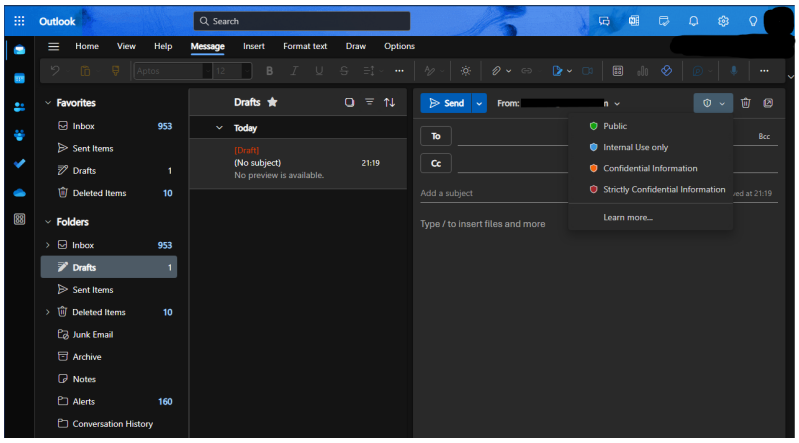


Рис. 3.4 Відправка поштою Outlook з приміткою «Конфіденційно»

3.2.4 Отримання документа та його збереження на комп’ютері отримувача

Підрядник, отримавши лист, зберігає його собі на комп’ютер для подальшої взаємодії з сервісом Дія.Підпис. Він також може перевірити перед підписанням документ, перечитати ще раз його умови. Саме в цей момент може виникнути думка «якось непомітно» відредагувати вміст цього документа, зберегти його і підписати вже відредаговану версію. Розглянемо якраз такий випадок. Adobe Acrobat Pro дає можливість редагувати вміст PDF-документа не експортуючи його в окремі програми, на кшталт Microsoft Office Word (див. рис. 3.5).

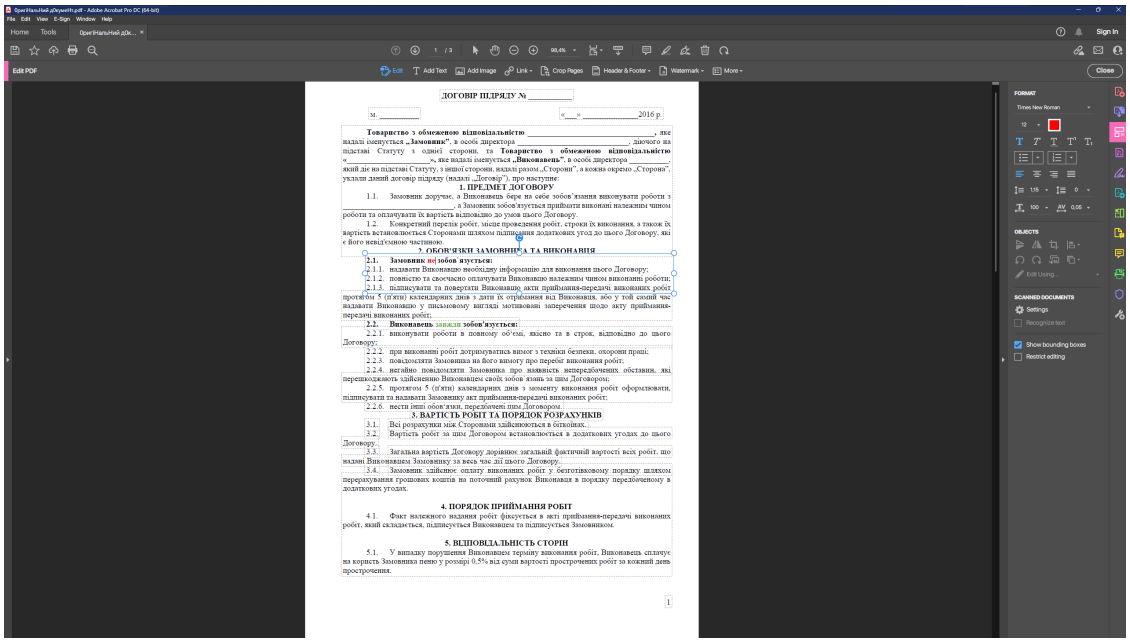


Рис. 3.5. Редагування тексту в Adobe Acrobat Pro

Після збережень змін, для наочності експерименту документи будуть мати різні назви: оригінальний – «Оригінальний документ», а відредагований «0ригіНальНий д0кумеНт».

3.2.5 Накладення ЕЦП та підготовка документа до відправлення

Як вже було зазначено вище, для підписання документа буде використовуватись сервіс Дія.Підпис з варіантом шифрування ЕЦП PAdES-B-B (див. рис. 3.6).

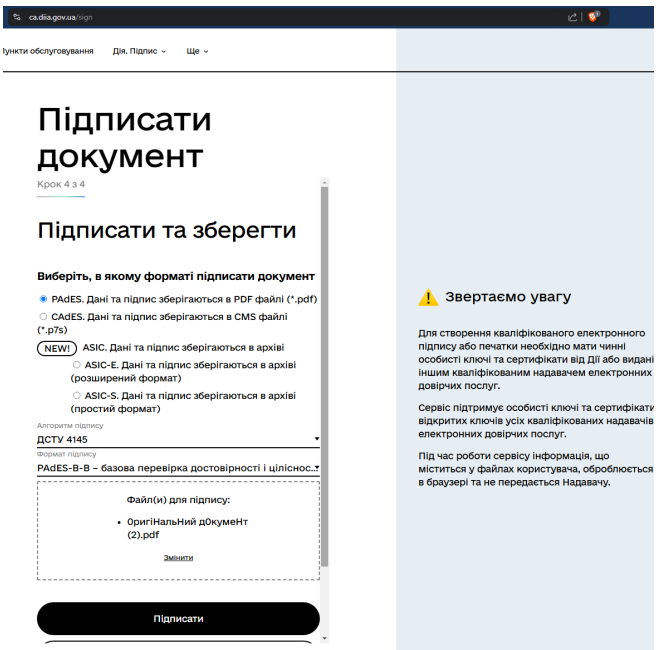


Рис. 3.6 Процес накладення підпису на документ

Цей варіант підходить для підписання PDF-документів, та дає змогу в подальшому відрізняти підписані документи від непідписаних за допомогою навіть переглядачів документів, на кшталт PDF-XChange Editor (див. рис. 3.7).

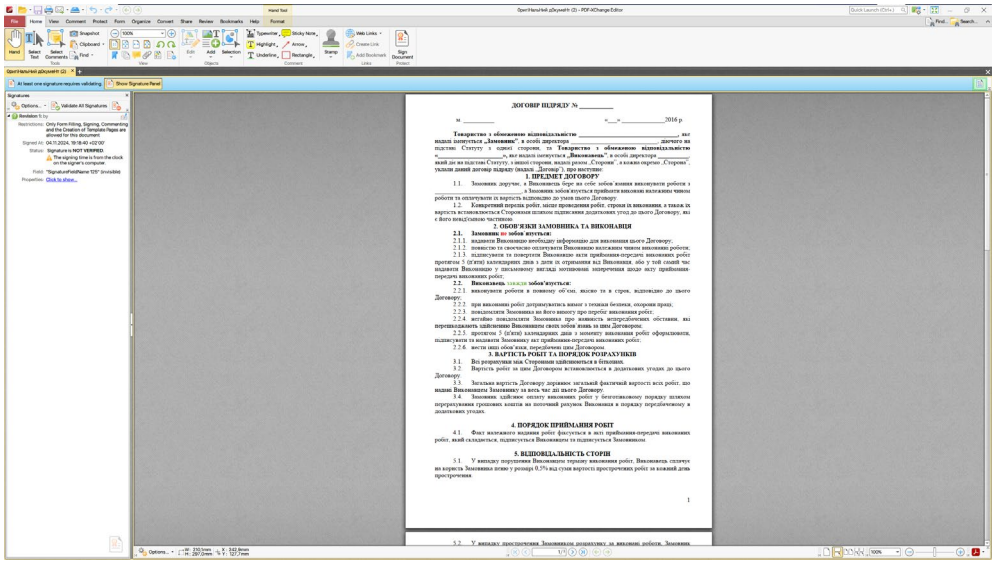
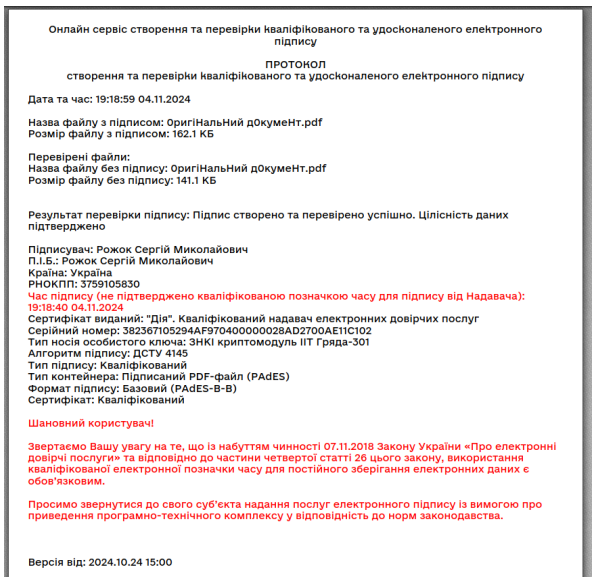


Рис. 3.7 Інтерфейс переглядача PDF-XChange Editor

Він не дає нам інформації щодо чинності цих сертифікатів, але сигналізує про їх наявність. Також після підписання, сервісом Дія.Підпис було згенеровано файл звіту, де є коротка інформація щодо типу ЕЦП, який було нанесено на документ, дату та інші метадані (див. рис. 3.8).



Онлайн сервіс створення та перевірки кваліфікованого та удосконаленого електронного підпису

ПРОТОКОЛ
створення та перевірки кваліфікованого та удосконаленого електронного підпису

Дата та час: 19:18:59 04.11.2024

Назва файлу з підписом: ОригіНальНий дОкумеНт.pdf
Розмір файлу з підписом: 162.1 КБ

Перевірені файли:
Назва файлу без підпису: ОригіНальНий дОкумеНт.pdf
Розмір файлу без підпису: 141.1 КБ

Результат перевірки підпису: Підпис створено та перевірено успішно. Цілісність даних підтверджено

Підписувач: Рожок Сергій Миколайович
П.І.Б.: Рожок Сергій Миколайович
Країна: Україна
РНОКПП: 3759105830
Час підпису (не підтверджено кваліфікованою позначкою часу для підпису від Надавача): 19:18:40 04.11.2024
Сертифікат виданий: "Дія". Кваліфікований надавач електронних довірчих послуг
Серійний номер: 382367105294AF97040000000028AD2700AE11C102
Тип носія особистого ключа: ЗНКІ криптомодуль ІІТ Гряда-301
Алгоритм підпису: ДСТУ 4145
Тип підпису: Кваліфікований
Тип контейнера: Підписаний PDF-файл (PAdES)
Формат підпису: Базовий (PAdES-B-B)
Сертифікат: Кваліфікований

Шановний користувач!

Звертаємо Вашу увагу на те, що із набуттям чинності 07.11.2018 Закону України «Про електронні довірчі послуги» та відповідно до частини четвертої статті 26 цього закону, використання кваліфікованої електронної позначки часу для постійного зберігання електронних даних є обов'язковим.

Просимо звернутися до свого суб'єкта надання послуг електронного підпису із вимогою про приведення програмно-технічного комплексу у відповідність до норм законодавства.

Версія від: 2024.10.24 15:00

Рис. 3.8 Згенерований протокол створення та перевірки ЕЦП

Отже, документ підписаний і готовий до відправки Підрядником.

3.2.6 Розміщення документа на веб-сторінці

Як свідчення прозорості ведення документації, остаточна підписана версія документа буде опублікована на веб-сторінці (це може бути цілком реально в невеликих компаніях, де документообіг буде відбуватись саме таким чином, і всі документи будуть зберігатись на веб-сервісі, до якого буде доступ лише локальний, або через окремий VPN-канал). Використаємо для цього Google Sites, щоб створити примітивну сторінку, де не буде нічого, окрім підписаного PDF-документа (див. рис. 3.9).

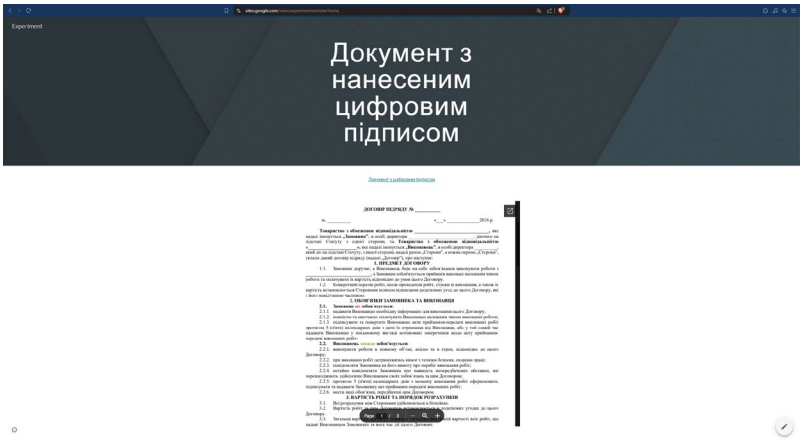


Рис. 3.9 Веб-сторінка на Google Sites, де розміщено підписану копію документа

3.3 Цикл роботи програмного модуля

Після сповіщення про те, що документ підписано, Власник вирішує перевірити цілісність вмісту підписаного документа, використовуючи автоматизований програмний модуль перевірки на автентичність, який ми розробили.

3.3.1 Крок 1. Запуск програми і графічне вікно взаємодії

Розгорнувши програмний модуль, ми можемо наочно бачити зверху донизу порядок взаємодії з програмою та коротку інструкцію з кольорового кодування результату, який ми отримаємо щойно програма відпрацює (див. рис. 3.10).

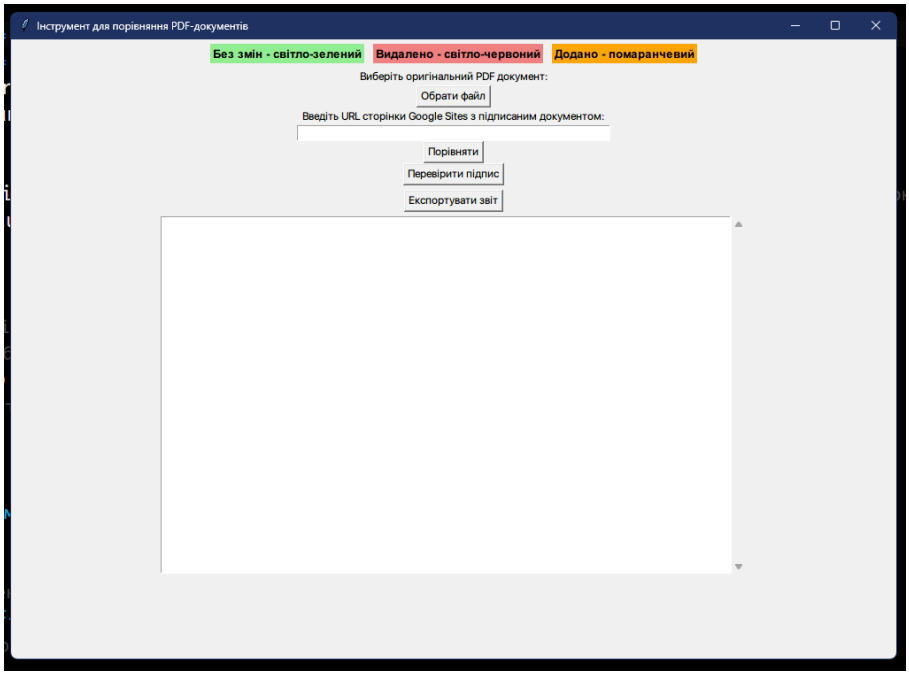


Рис. 3.10 Інтерфейс програмного модуля автоматизованої перевірки на автеничність

3.3.2 Крок 2. Завантаження оригінального документа у програмний модуль

Завантажимо оригінальний документ у програмний модуль використовуючи кнопку «Обрати файл» (див. рис. 3.11).

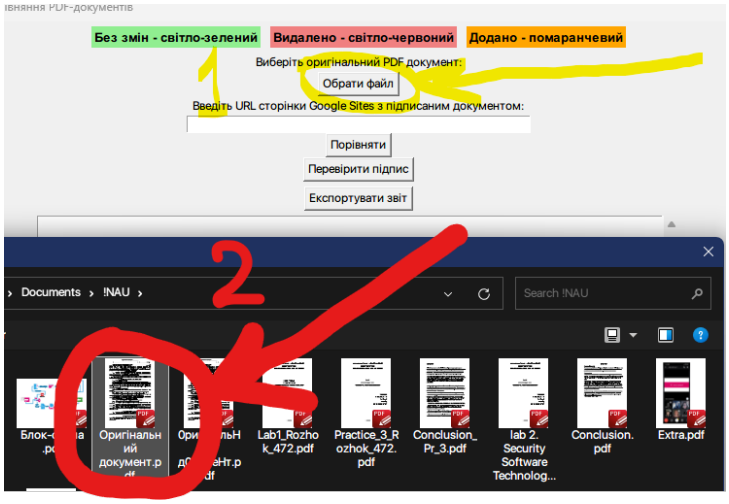


Рис. 3.11 Процес вибору документа, який буде слугувати «оригіналом»

Після завантаження документа, програма сигналізує про успішний імпорт відповідним випливаючим вікном з повідомленням (див. рис. 3.12).

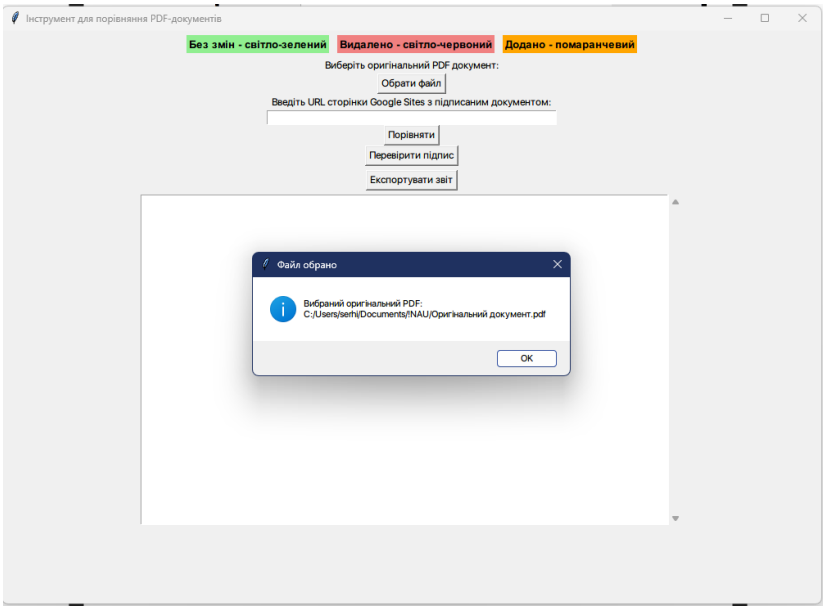


Рис. 3.12 Випливаюче вікно з повідомленням

3.3.3 Крок 3. Вставка URL-сторінки, де знаходиться підписаний документ

Після створення веб-сторінки на Google Sites йому було присвоєно URL-адресу, яку ми використаємо в цьому кроці. Попередньо ми туди теж вже розмістили підписаний документ у форматі PDF. Для інтеграції цієї веб-сторінки, вставимо URL-адресу у спеціальне поле в інтерфейсі програмного модуля (див. рис. 3.13).

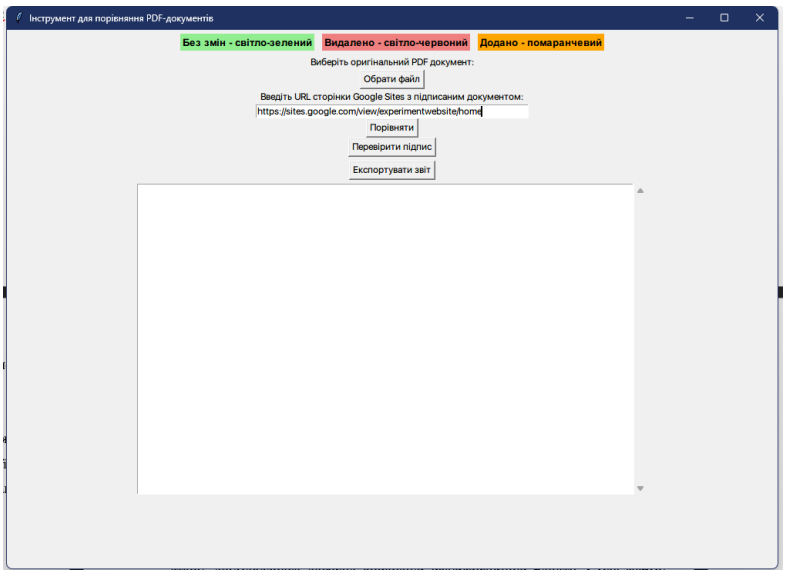


Рис. 3.13 Інтерфейс програми після вставки URL-адреси

3.3.4 Крок 4. Процес парсингу (розпізнавання) та порівняння вмісту документів

Після того як всі необхідні для роботи дані імпортовано, почнемо роботу програми кнопкою «Порівняти». У фоновому режимі програмний модуль насамперед перевіряє валідність документів та можливість їх розпізнавання, обробляє різні винятки та можливі помилки при імпорті посилання на веб-сторінку чи оригінальний документ, і у випадку позитивного результату протягом декількох секунд видає результат (див. рис. 3.14).

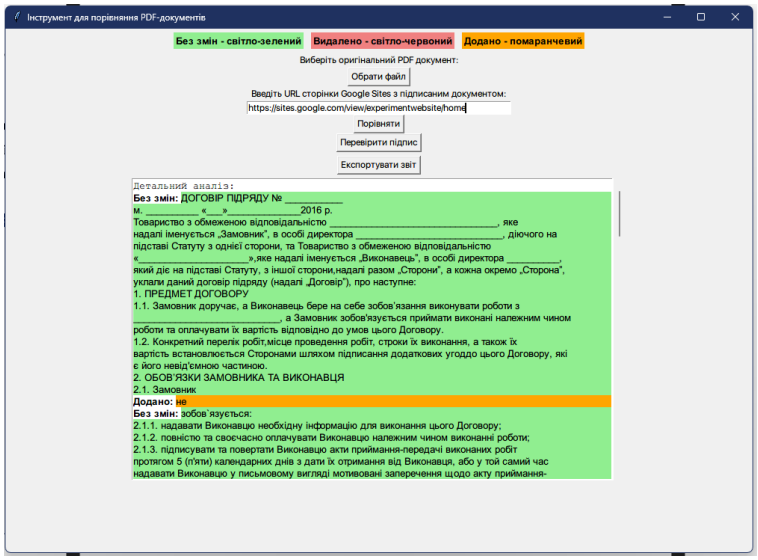


Рис. 3.14 Результат автоматизованого порівняння

Як було зазначено раніше, програма використовує кольорове кодування для зручності аналізу результату автоматизованого порівняння. Гортаючи донизу, можна звернути увагу на різні відмінності, які цей програмний модуль зміг виявити і підсвітити. У самому кінці звіту ми маємо короткий підсумок результату перевірки на автентичність (див. рис. 3.15).

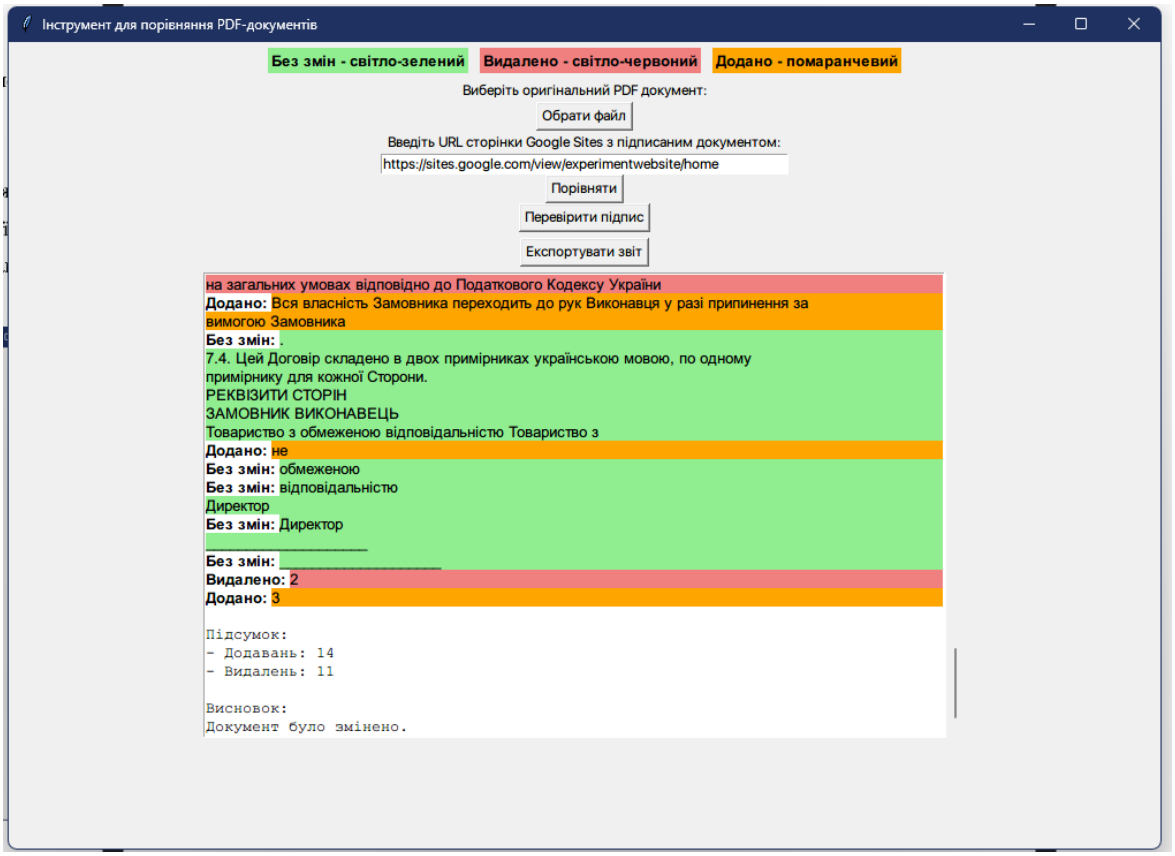


Рис. 3.15 Короткий підсумок результату перевірки на автентичність

3.3.5 Крок 5. Перевірка підпису за допомогою сервісу Дія.Підпис

Як і очікувалось, наш застосунок зміг виявити, що документ було змінено, але це не єдина його функція. Також було інтегровано сюди сервіс Дія.Підпис, який відкривається за допомогою кнопки «Перевірити підпис» у окремому вікні, і дає змогу перевірити сертифікати на достовірність та чинність. Натиснувши на цю кнопку, програма інформує нас, що підписаний документ було збережено в директорію, з якої відпрацьовує програма (див. рис. 3.16).

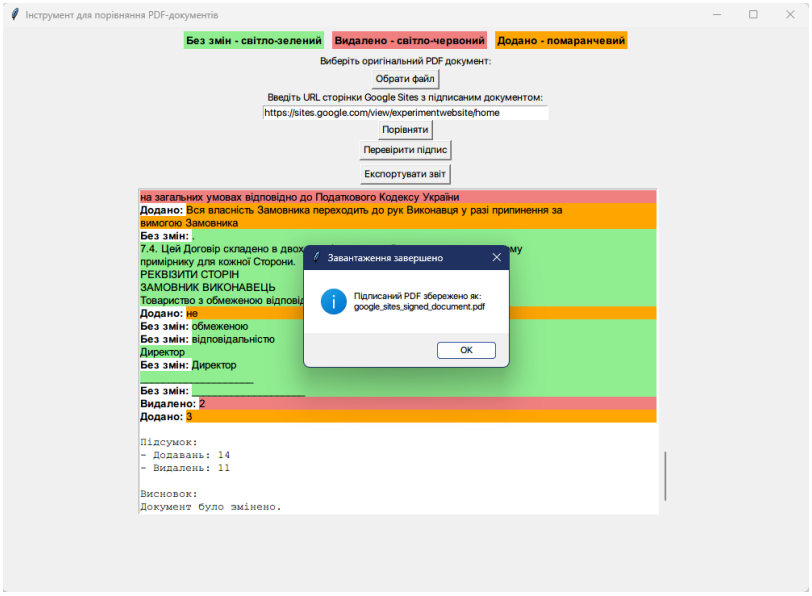


Рис. 3.16 Інформативне повідомлення про збереження документа

Це зроблено для зручності, адже так як ми імпортували раніше цей документ з веб-сторінки, фізично на комп’ютері він був збережений лише тимчасово у фоновому режимі для коректної роботи програми. Після натискання кнопки «ОК», у окремому вікні відкривається одразу потрібна веб-сторінка Дія.Підпис, де необхідно імпортувати документ, на який нанесено цифровий підпис, щоб перевірити наявність та чинність його сертифікатів (див. рис. 3.17).

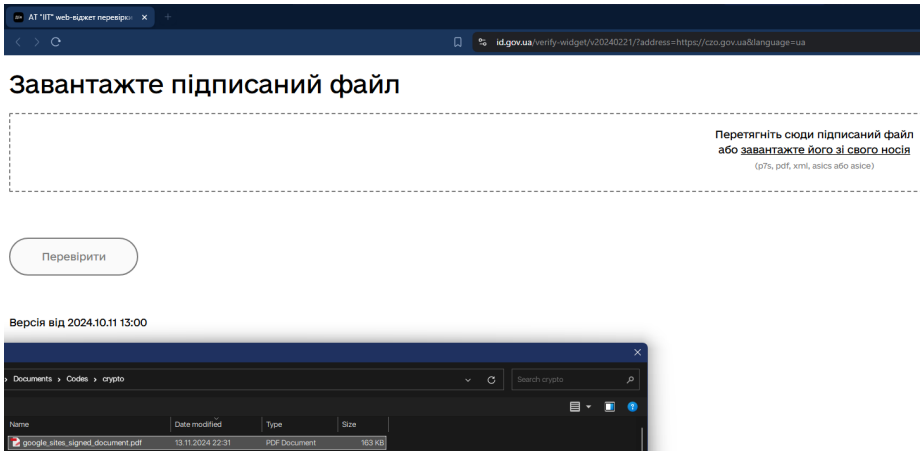


Рис. 3.17 Процес імпорту документа для перевірки сертифікатів

Якщо всі кроки були вірними, то після натискання кнопки «Перевірити» має з’явитись коротка інформація про статус сертифікатів, які було використано для підпису документа (див. рис. 3.18).

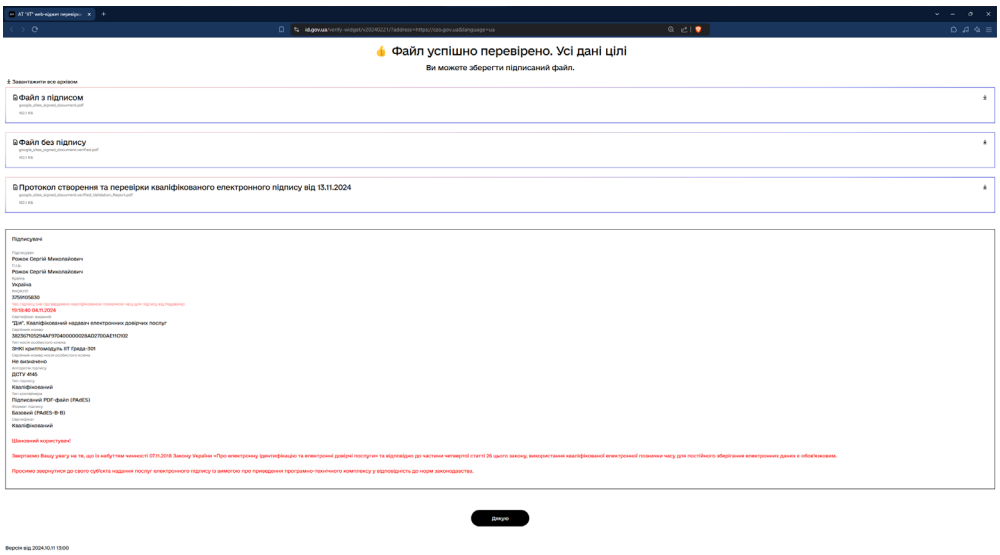


Рис. 3.18 Інформативне повідомлення про успішну перевірку Дія.Підпис

Також, необхідно розглянути випадки, коли документ може не мати підпису взагалі, або підпис може бути пошкодженим чи навіть мати застарілі сертифікати, які втратили свою чинність, і вважаються недійсними. Спробуємо підвантажити оригінальний документ, на якому ми впевнені, що немає ніякого підпису (див. рис. 3.19).

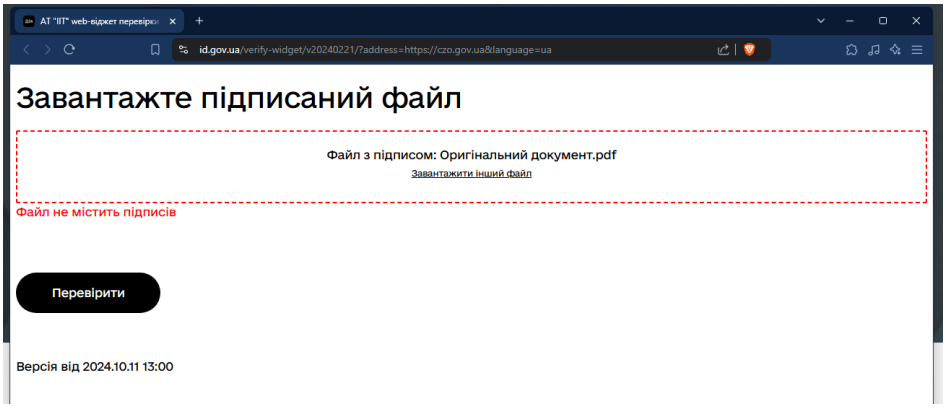


Рис. 3.19 Інформативне повідомлення від Дія.Підпис про невдалу перевірку сертифікатів

3.3.6 Крок 6. Експорт звіту в формат .docx.

Також в програмному модулі реалізовано можливість експорту звіту в формат .docx для подальшого перегляду та можливого надсилання іншим особам, наприклад Підряднику, для того, щоб підкреслити та підсвітити, які саме

зміни були в документі. Для цього натиснемо кнопку «Експортувати звіт», де після цього в нас запитає шлях, куди ми плануємо його зберегти (див. рис. 3.20).

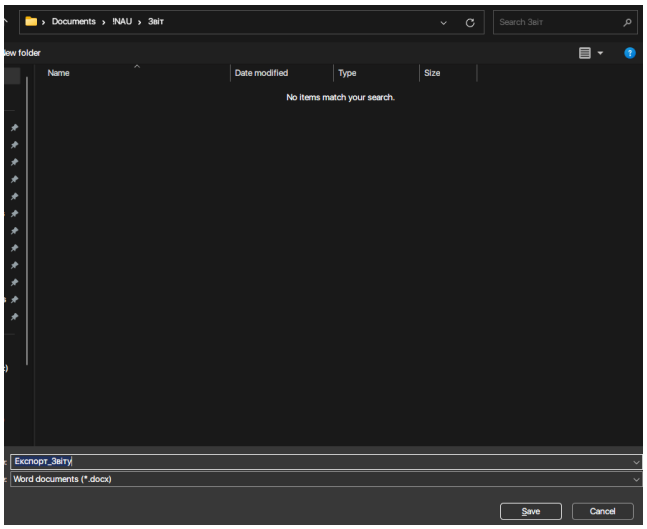


Рис. 3.20 Процес збереження звіту

Також, отримуємо повідомлення про успішність збереження звіту за вказаною директорією (див. рис. 3.21).

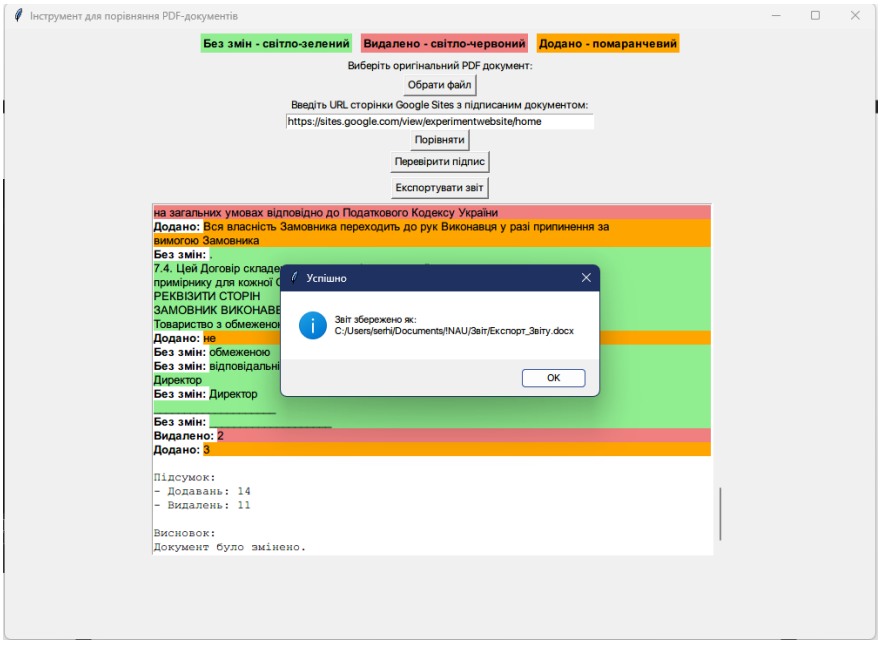


Рис. 3.21 Інформативне повідомлення про збереження звіту

Відкриємо документ, щоб перевірити, що він містить необхідну інформацію з інтерфейсу програми (див. рис. 3.22).

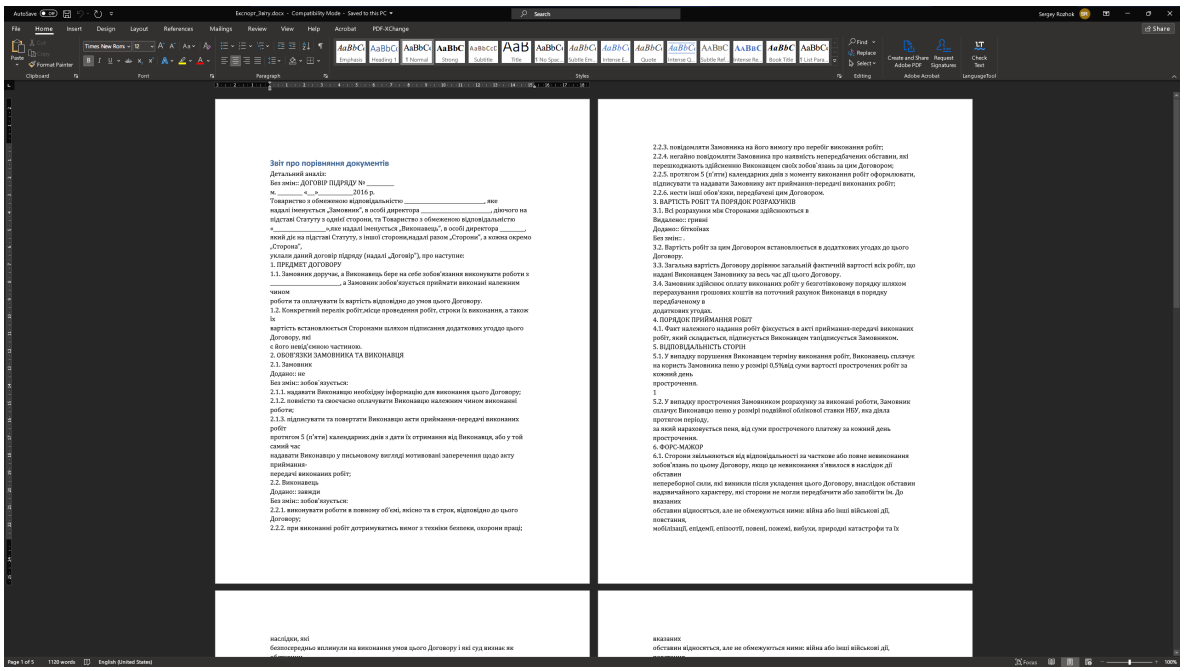


Рис. 3.22 Вигляд вмісту експортованого звіту

Нас задовільняє те, що основні ключові слова «Додано», «Змінено» та «Видалено» тут присутні, тому можна вважати експорт звіту успішним. На цьому робота з програмою завершена. Згідно блок-схеми, ми повернулись в початкову точку, адже розпочиналось все з комп’ютерної системи Власника, і закінчилось теж ініціалізацією програмного модуля, розгорнутого на базі його комп’ютерної системи.

3.4 Висновки за розділом 3

Отже, використавши розроблений програмний модуль для автоматизованої перевірки сайтів на автентичність, нам вдалось перевірити цілісність вмісту документа, який спочатку було надіслано від Власника до Підрядника, потім підписано Підрядником і розміщено на веб-сторінці. В процесі транспортування документа могло бути не виключено, що документ буде відредаговано і його автентичність буде порушена, що і врешті сталось в процесі експерименту. Так як документ було змінено Підрядником, Власник, отримавши підписаний документ, використав розроблений програмний модуль, який проаналізував оригінальний документ та підписаний та виявив усі ці

відмінності та зручно їх сформував у звіт. Також, за допомогою функції перевірки підпису за допомогою зовнішнього сервісу Дія.Підпис було виявлено, що сертифікат був справжнім, але зважаючи на порушення цілісності вмісту документа, таке підписання було оцінене як недійсне. Зробивши експорту звіту у формат .docx, Власник може в подальшому використовувати його як підтвердження неспівпадіння підписаного документа, який він отримав від Підрядника.

ВИСНОВКИ

У процесі роботи над цією кваліфікаційною роботою розглянуто проблему забезпечення автентичності та цілісності цифрових документів за допомогою методів шифрування, цифрових підписів, хешування та інших технологій захисту. Актуальність дослідження обумовлена зростаючими вимогами до безпеки інформації в умовах розвитку електронного документообігу, державних послуг в цифровому середовищі та захисту інтелектуальної власності в епоху цифровізації, що було наведено у вступній частині.

Аналіз нормативно-правової бази з захисту авторських та суміжних прав надав можливість усвідомити необхідність перевірки документів на автентичність для викриття порушення цілісності їх вмісту, які передаються для укладення партнерської угоди.

Проведений аналіз методів для електронного засвідчення документів після перевірки їх на цілісність вмісту надав можливості визначити методи для розробки основних алгоритмів програмного модуля перевірки документів на автентичність.

Розроблено та протестовано програмний модуль перевірки сайтів на автентичність мовою програмування Python. Використання у модулі методів текстової обробки дало можливість автоматизувати перевірку цілісності інформації у PDF-документах.

Дана програма розроблена з використанням сучасних інструментів і технологій Python, зокрема бібліотек Tkinter для побудови графічного інтерфейсу, pdfplumber для витягання тексту з PDF, а також BeautifulSoup і Requests для отримання даних з веб-ресурсів. Крім того, застосовано алгоритм diff_match_patch для точного порівняння тексту та python-docx для експорту результатів у форматі DOCX.

У сучасних умовах електронного документообігу та розвитку цифрових підписів зростає потреба в інструментах для порівняння документів з метою

виявлення змін у тексті та перевірки цілісності. Проблема перевірки відповідності та виявлення змін у тексті має суттєве значення для сфери юриспруденції, архівної справи, фінансів і менеджменту. Розроблене програмне забезпечення вирішує цю проблему шляхом автоматизації процесу порівняння документів та аналізу текстових змін, а також забезпечує можливість перевірки цифрових підписів, що сприяє підвищенню рівня надійності електронних документів та знижує ризики маніпуляцій із текстом[44].

У рамках дослідження систематизовано знання про сучасні методи криптографії та ЕЦП, що використовуються для захисту документів. Проведено огляд стандартів, таких як PAdES-B-B, та розглянуто ефективність основних алгоритмів для цифрового підпису, включно з RSA, AES, ECC. Вивчення алгоритмів та їх оптимізації дозволяє сформувати нові підходи до забезпечення інформаційної безпеки в системах електронного документообігу та цифрової ідентифікації. Розширено теоретичну базу для подальших досліджень у галузі криптографії та розробки нових методів захисту даних[45].

Новизна розробленої програми полягає в комплексному підході до аналізу PDF-документів, що поєднує автоматичне витягання тексту з PDF, порівняння змін із застосуванням алгоритму diff_match_patch та можливість автоматичного завантаження підписаних документів із веб-сайтів. Завдяки кольоровому кодуванню результатів аналізу програма забезпечує зручність візуального сприйняття змін у тексті, що також є новим підходом у сфері програмного забезпечення для порівняння документів.

Теоретичне значення роботи полягає в інтеграції методів текстової обробки, зокрема алгоритмів порівняння текстів, з іншими інструментами Python, такими як pdfplumber для роботи з PDF-документами та BeautifulSoup для парсингу веб-сторінок. Практична цінність полягає в тому, що розроблений інструмент може використовуватись для автоматизації аналізу змін у юридичних, архівних і фінансових документах, а також для інтеграції з національними системами перевірки електронних підписів[32].

Практичні результати дослідження можуть бути використані для вдосконалення систем безпеки у державних електронних послугах, таких як "Дія", а також у банківській сфері, де особливо важливі цифрові підписи та автентифікація документів. Описані методи захисту можуть слугувати основою для розробки надійних програмних рішень, що дозволяють автоматично перевіряти цілісність і достовірність документів на рівні корпоративних та державних інформаційних систем[45].

Рекомендації для подальших досліджень і вдосконалень програми

Зважаючи на отримані результати, можна виділити кілька напрямів для майбутніх досліджень та вдосконалень програми:

- Розширення функціоналу порівняння для інших форматів документів (наприклад, DOCX, TXT), що дозволить збільшити гнучкість та універсальність використання програмного забезпечення.
- Інтеграція з іншими системами електронного документообігу та підпису, зокрема, розширення підтримки міжнародних стандартів електронних підписів.
- Оптимізація алгоритмів порівняння для підвищення швидкодії та точності при обробці великих обсягів тексту, включаючи використання більш складних алгоритмів обробки тексту, таких як нейронні мережі для порівняння семантики текстів.
- Розробка веб-інтерфейсу для програми, що дозволить забезпечити доступ до функцій порівняння документів через інтернет, розширюючи коло потенційних користувачів.

Таким чином, проведене дослідження зробило внесок у розвиток методів забезпечення автентичності цифрових документів, що є особливо актуальним для державних та комерційних установ, які працюють у цифровому середовищі.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Борисенко А. П. "Технології автентифікації цифрових документів". – Харків: Право, 2021. – С. 45-80.
2. Кириченко В. С. "Інтеграція криптографічних стандартів у документообіг". – Київ: Юрінком Інтер, 2020. – С. 15-48.
3. Садовська Ю. Г. "Електронний цифровий підпис як інструмент забезпечення довіри". – Київ: Наукова думка, 2022. – С. 26-68.
4. Василенко П. В. "Перевірка автентичності документів у правовій сфері". – Харків: Юрінком Інтер, 2021. – С. 48-93.
5. Diffie, W., & Hellman, M. E. New Directions in Cryptography. – IEEE Transactions on Information Theory, 1976. – P. 644-654.
6. Stinson, D. Cryptography: Theory and Practice. – CRC Press, 2005. – P. 72-139.
7. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. Handbook of Applied Cryptography. – CRC Press, 1996. – P. 25-100.
8. Kahn, D. The Codebreakers: The Story of Secret Writing. – New York: Scribner, 1996. – P. 301-452.
9. Katz, J., & Lindell, Y. Introduction to Modern Cryptography. – New York: CRC Press, 2014. – P. 58-135.
10. Bellare, M., & Rogaway, P. Introduction to Modern Cryptography. – UCSD, 2005. – P. 101-183.
11. Anderson, R. Security Engineering: A Guide to Building Dependable Distributed Systems. – Wiley, 2020. – P. 78-145.
12. Ferguson, N., Schneier, B., & Kohno, T. Cryptography Engineering: Design Principles and Practical Applications. – Wiley, 2010. – P. 40-105.
13. Koblitz, N. A Course in Number Theory and Cryptography. – Springer, 1994. – P. 63-123.

14. Прокопенко Т. В. "Цифрова ідентифікація у правових системах". – Харків: Юрінком Інтер, 2021. – С. 35-77.

15. Твори, автори та їхні права. URL: https://cases.media/en/article/tvori-avtori-ta-yikhni-prava (дата звернення 18.10.2024).

16. Закон України "Про авторське право і суміжні права". – № 3792-XII від 23.12.1993 р. – С. 1-25.

17. Цивільний кодекс України. – Верховна Рада України, 2003. – С. 45-97.

18. Кримінальний кодекс України. – Верховна Рада України, 2001. – С. 101-137.

19. Berne Convention for the Protection of Literary and Artistic Works. – 1886. – С. 10-35.

20. Мартинюк І. М. Захист суміжних прав. – Харків: НТУ, 2020. – С. 39-83.

21. Порядок реєстрації авторських прав в Україні. Міністерство розвитку економіки, торгівлі та сільського господарства України. – С. 5-15.

22. Андрєєва Н. М. Криптографічні методи захисту даних: Практичний посібник. – Київ: Видавничий дім, 2020. – С. 45-112.

23. Рожок С.М. Програмний модуль перевірки сайтів на автентичність /С.М. Рожок, Н.К. Гулак //VII International scientific and practical conference «World educational trends: lifelong learning in the information society», 15-18 жовтня 2024р. - Афіни, Греція. – С. 64-66.

24. Чурікова В. Основні механізми від захисту електронних документів від фальсифікації. URL: https://www.slideshare.net/slideshow/7pptx-257646933/257646933 (дата звернення 29.10.2024).

25. Симетричне і асиметричне шифрування: визначення поняття, застосування, приклади. URL: https://hi-news.pp.ua/kompyuteri/12565-simetrichne-asimetrichne-shifruvannya-viznachennya-ponyattya-zastosuvannya-prikladi.html (дата звернення 29.10.2024).

26. McAuley, J. The Basics of Digital Encryption for Businesses. – Cambridge: Cambridge University Press, 2018. – pp. 75-140.

27. Що таке PGP (GPG) і як із ним працювати/шифрувати емейли на прикладі Mailvelope. URL: https://technoblogua.wordpress.com/tag/асиметричне-шифрування (дата звернення 29.10.2024).

28. До уваги ОГС! 31 травня – останній день дії електронних цифрових підписів! URL: https://www.prostir.ua/?news=do-uvahy-ohs-31-travnya-ostannij-den-diji-elektronnyh-tsyfrovyh-pidpysiv (дата звернення 01.11.2024).

29. Дія. Як підписати документи у додатку "Дія". URL: https://diia.gov.ua/ (дата звернення 01.11.2024).

30. Разумков Д. Що таке парсинг. URL: https://aboutmarketing.info/internet-marketynh/instrumenty/shcho-take-parsynh/ (дата звернення 01.11.2024).

31. Ляшенко Д. І. "Парсинг і обробка цифрових документів". – Харків, 2020. – С. 12-38.

32. Шевчук В. М. Захист PDF-документів: технології та засоби. – Харків: Вид-во ХНУ, 2021. – С. 34-78.

33. Олійник Д. М. "Парсинг даних у сучасних інформаційних системах". – Одеса, 2022. – С. 20-50.

34. Java. Алгоритми і структури даних.pdf URL: https://studfile.net/preview/5782860/page:10/ (дата звернення 01.11.2024).

35. Ковальчук І. В. "Криптографічні алгоритми та протоколи". – Одеса, 2022. – С. 33-88.

36. Каплін Р. С. Інтелектуальна власність: правовий аспект. Харків: ХНУ ім. В.Н. Каразіна, 2020. – С. 28-90.

37. НБУ показав елементи захисту банкноти в тисячу гривень. URL: https://www.rbc.ua/ukr/news/nbu-pokazal-elementy-zashchity-banknoty-tysyachu-1572005883.html (дата звернення 01.11.2024).

38. Ребров В. А. "Електронний цифровий підпис: теорія та практика". – Київ, 2018. – С. 15-40.

39. Рибалко С. А. "Безпека електронних транзакцій у фінансовому секторі". – Київ, 2021. – С. 11-40.

40. Кому і навіщо потрібні електронні цифрові підписи? URL: https://legalclinics.in.ua/consult/consultation-19-12-2022-3/ (дата звернення 03.11.2024).

41. diff-match-patch URL: https://github.com/google/diff-match-patch (дата звернення 03.11.2024).

42. Міністерство цифрової трансформації України. "Особливості використання ЕЦП в Україні". – 2022. – С. 3-10.

43. Rivest, R. L., & Shamir, A. Cryptography and Data Security. – Addison-Wesley, 2002. – P. 95-152.

44. Слободянюк О. М. "Електронні підписи в правовому полі України". – Львів, 2020. – С. 9-32.

45. Інструкція з використання PAdES для електронного підпису документів / Розробка ЕС ETSI. URL: https://www.etsi.org/ – С. 12-40.

ДОДАТКИ

Додаток А

Програмне забезпечення для перевірки сайтів на автентичність

```
import tkinter as tk  # Імпорт бібліотеки для створення графічного інтерфейсу
from tkinter import filedialog, scrolledtext, messagebox  # Імпорт компонентів для діалогів, текстових полів та повідомлень
import pdfplumber  # Імпорт бібліотеки для роботи з PDF файлами
import io  # Імпорт модуля для роботи з потоками вводу/виводу
import webbrowser  # Імпорт модуля для відкриття веб-сторінок у браузері
import requests  # Імпорт бібліотеки для роботи з HTTP запитами
from bs4 import BeautifulSoup  # Імпорт бібліотеки для парсингу HTML
from diff_match_patch import diff_match_patch  # Імпорт бібліотеки для порівняння тексту
from docx import Document  # Імпорт бібліотеки для роботи з документами Word

class PDFHandler:
    """Клас для обробки завантаження та витягування тексту з PDF файлів."""

    @staticmethod
    def extract_text_from_pdf(pdf_data):
        """Витягує текст з PDF у вигляді бінарних даних."""
        try:
            text_content = []  # Список для зберігання витягнутого тексту
            with pdfplumber.open(io.BytesIO(pdf_data)) as pdf:  # Відкриваємо PDF з бінарних даних
                for page in pdf.pages:  # Перебираємо всі сторінки PDF
```

```
                text_content.append(page.extract_text() or "")  # Витягуємо текст з кожної сторінки
            return "\n".join(text_content)  # Повертаємо витягнутий текст як єдиний рядок
        except Exception as e:  # Обробка виключень
            messagebox.showerror("Помилка", f"Не вдалося витягти текст з PDF: {e}")  # Показуємо повідомлення про помилку
            return None  # Повертаємо None у випадку помилки

    @staticmethod
    def load_pdf_from_url(url):
        """Завантажує дані PDF з URL Google Sites, якщо доступно."""
        try:
            response = requests.get(url)  # Виконуємо HTTP запит для отримання сторінки
            if response.status_code == 200:  # Перевіряємо, чи отримано сторінку успішно
                soup = BeautifulSoup(response.content, 'html.parser')  # Парсимо HTML контент
                pdf_links = soup.find_all('a', href=True)  # Знаходимо всі посилання на сторінці
                for link in pdf_links:  # Перебираємо всі знайдені посилання
                    if link['href'].endswith('.pdf') or 'export=download&id=' in link['href']:  # Перевіряємо, чи це посилання на PDF
                        pdf_url = link['href']  # Зберігаємо URL PDF
                        if not pdf_url.startswith("http"):  # Якщо URL не починається з 'http'
                            pdf_url = "https://sites.google.com" + pdf_url  # Додаємо базовий URL
                        pdf_data = requests.get(pdf_url).content  # Завантажуємо PDF
```

```
                return pdf_data  # Повертаємо дані PDF
            messagebox.showwarning("Попередження", "Не знайдено посилання на PDF на сторінці Google Sites.")  # Попередження, якщо PDF не знайдено
        else:
            messagebox.showerror("Помилка", "Не вдалося отримати сторінку Google Sites.")  # Помилка при отриманні сторінки
    except Exception as e:  # Обробка виключень
        messagebox.showerror("Помилка", f"Помилка при завантаженні сторінки Google Sites: {e}")  # Показуємо повідомлення про помилку
    return None  # Повертаємо None у випадку помилки


class DocumentComparator:
    """Клас для порівняння текстових відмінностей між оригінальними та підписаними документами."""

    def __init__(self, result_text_widget):
        self.result_text_widget = result_text_widget  # Зберігаємо віджет для відображення результатів
        self.differences = []  # Зберігаємо відмінності для генерації звіту

    def analyze_differences(self, original_text, signed_text):
        """Аналізує та підсвічує відмінності між двома текстами."""
        dmp = diff_match_patch()  # Ініціалізація об'єкта для порівняння тексту
        diffs = dmp.diff_main(original_text, signed_text)  # Отримуємо відмінності між текстами
        dmp.diff_cleanupSemantic(diffs)  # Очищаємо відмінності для кращого відображення
```

```
            # Очищуємо попередній аналіз
            self.result_text_widget.delete('1.0', tk.END)  # Очищуємо текстове поле
            self.result_text_widget.insert(tk.END, "Детальний аналіз:\n")  # Додаємо заголовок для аналізу

            # Аналізуємо та відображаємо відмінності з кольоровим кодуванням
            for (op, text) in diffs:  # Перебираємо всі відмінності
                text = text.strip()  # Відрізаємо пробіли на початку та в кінці
                if text == " ":
                    text = "[ПРОБІЛ]"  # Замінюємо пробіл на позначення
                elif text == "\n":
                    text = "[НОВИЙ РЯДОК]"  # Замінюємо новий рядок на позначення
                elif not text:  # Якщо текст порожній, пропускаємо
                    continue

                tag, label = "", ""  # Ініціалізуємо змінні для тегу та мітки
                if op == 0:  # Якщо текст без змін
                    tag, label = "green", "Без змін: "  # Встановлюємо зелений колір
                elif op == 1:  # Якщо текст додано
                    tag, label = "orange", "Додано: "  # Встановлюємо оранжевий колір
                elif op == -1:  # Якщо текст видалено
                    tag, label = "red", "Видалено: "  # Встановлюємо червоний колір

                # Додаємо мітку та текст з форматуванням
                self.result_text_widget.insert(tk.END, label, "label")  # Додаємо мітку
                self.result_text_widget.insert(tk.END, text + "\n", tag)  # Додаємо текст з кольоровим фоном

                # Зберігаємо відмінності для генерації звіту
```

```
                self.differences.append((label.strip(), text))  # Додаємо до списку відмінностей

            # Підсумок та висновок
            additions = len([d for d in diffs if d [0] == 1])  # Кількість доданих рядків
            deletions = len([d for d in diffs if d [0] == -1])  # Кількість видалених рядків
            summary = f"\nПідсумок:\n- Додавань: {additions}\n- Видалень: {deletions}\n"  # Формуємо підсумок
            self.result_text_widget.insert(tk.END, summary)  # Додаємо підсумок до тексту

            conclusion = "\nВисновок:\n"  # Ініціалізуємо рядок для висновку
            conclusion += "Документ не змінено." if additions == 0 and deletions == 0 else "Документ було змінено."  # Визначаємо висновок
            self.result_text_widget.insert(tk.END, conclusion)  # Додаємо висновок до тексту

        def get_report_data(self):
            """Повертає дані для звіту."""
            report_data = ["Детальний аналіз:\n"]  # Ініціалізуємо список для звіту
            for label, text in self.differences:  # Перебираємо всі відмінності
                report_data.append(f"{label}: {text}\n")  # Додаємо відмінності до звіту
            return "".join(report_data)  # Повертаємо звіт як єдиний рядок


    class DocumentComparisonApp:
        """Основний клас програми, який обробляє UI та взаємодію з користувачем."""
```

```
    def __init__(self, root):
        self.root = root  # Зберігаємо кореневий елемент вікна
        self.root.title("Інструмент для порівняння PDF-документів")  # Встановлюємо заголовок вікна
        self._center_and_resize_window()  # Центруємо та змінюємо розмір вікна

        # Компоненти інтерфейсу
        self._setup_legend()  # Налаштовуємо легенду кольорів
        self._setup_input_fields()  # Налаштовуємо поля вводу
        self._setup_buttons()  # Налаштовуємо кнопки
        self._setup_result_display()  # Налаштовуємо область для відображення результатів

    def _center_and_resize_window(self):
        """Центрує вікно на екрані з більшою роздільною здатністю."""
        window_width = 1000  # Ширина вікна
        window_height = 700  # Висота вікна
        screen_width = self.root.winfo_screenwidth()  # Ширина екрана
        screen_height = self.root.winfo_screenheight()  # Висота екрана

        x_position = (screen_width // 2) - (window_width // 2)  # Обчислюємо координати x для центрування
        y_position = (screen_height // 2) - (window_height // 2)  # Обчислюємо координати y для центрування

self.root.geometry(f"{window_width}x{window_height}+{x_position}+{y_position}")  # Встановлюємо геометрію вікна
```

```
    def _setup_legend(self):
        """Налаштовує легенду кольорів для відмінностей."""
        legend_frame = tk.Frame(self.root)  # Створюємо рамку для легенди
        legend_frame.pack(pady=5)  # Додаємо відступи

        legend_text = [  # Список з описами кольорів
            ("Без змін - світло-зелений", "lightgreen"),  # Опис для без змін
            ("Видалено - світло-червоний", "lightcoral"),  # Опис для видалення
            ("Додано - помаранчевий", "orange")  # Опис для додавання
        ]
        for text, color in legend_text:  # Перебираємо описи
            legend_label = tk.Label(legend_frame, text=text, bg=color, font=("Arial", 10, "bold"))  # Створюємо мітку
            legend_label.pack(side=tk.LEFT, padx=5)  # Додаємо мітку до рамки

    def _setup_input_fields(self):
        """Налаштовує поля вводу для вибору оригінального документа та URL підписаного документа."""
        self.file_label = tk.Label(self.root, text="Виберіть оригінальний PDF документ:")  # Мітка для вибору документа
        self.file_label.pack()  # Додаємо мітку до вікна

        self.file_button = tk.Button(self.root, text="Обрати файл", command=self.select_file)  # Кнопка для вибору файлу
        self.file_button.pack()  # Додаємо кнопку до вікна

        self.url_label = tk.Label(self.root, text="Введіть URL сторінки Google Sites з підписаним документом:")  # Мітка для URL
        self.url_label.pack()  # Додаємо мітку до вікна
```

```
        self.url_entry = tk.Entry(self.root, width=50)  # Поле вводу для URL
        self.url_entry.pack()  # Додаємо поле вводу до вікна

    def _setup_buttons(self):
        """Налаштовує кнопки для порівняння та перевірки підпису."""
        self.compare_button = tk.Button(self.root, text="Порівняти", command=self.compare_documents)  # Кнопка для порівняння документів
        self.compare_button.pack()  # Додаємо кнопку до вікна

        self.verify_button = tk.Button(self.root, text="Перевірити підпис", command=self.verify_signature_flow)  # Кнопка для перевірки підпису
        self.verify_button.pack()  # Додаємо кнопку до вікна

        self.export_button = tk.Button(self.root, text="Експортувати звіт", command=self.export_report)  # Кнопка для експорту звіту
        self.export_button.pack(pady=5)  # Додаємо кнопку до вікна з відступами

    def _setup_result_display(self):
        """Налаштовує область для відображення результатів з кольоровим кодуванням."""
        self.result_text = scrolledtext.ScrolledText(self.root, width=80, height=25)  # Збільшена висота до 25
        self.result_text.pack()  # Додаємо текстове поле до вікна
        self.result_text.tag_configure("green", background="lightgreen", font=("Arial", 10))  # Налаштовуємо тег для зеленого кольору
        self.result_text.tag_configure("orange", background="orange", font=("Arial", 10))  # Налаштовуємо тег для оранжевого кольору
```

```
        self.result_text.tag_configure("red", background="lightcoral", font=("Arial", 10))  # Налаштовуємо тег для червоного кольору
        self.result_text.tag_configure("label", font=("Arial", 10, "bold"))  # Налаштовуємо тег для міток

        self.comparator = DocumentComparator(self.result_text)  # Ініціалізуємо об'єкт для порівняння документів

    def select_file(self):
        """Дозволяє користувачу вибрати оригінальний PDF документ."""
        self.filepath = filedialog.askopenfilename(filetypes=[("PDF files", "*.pdf")])  # Відкриваємо діалог вибору файлу
        if self.filepath:  # Якщо файл вибрано
            messagebox.showinfo("Файл обрано", f"Вибраний оригінальний PDF: {self.filepath}")  # Показуємо інформацію про вибраний файл

    def verify_signature_flow(self):
        """Завантажує підписаний документ та відкриває веб-сайт для перевірки підпису."""
        url = self.url_entry.get()  # Отримуємо URL з поля вводу
        if not url:  # Якщо URL не введено
            messagebox.showwarning("Попередження", "Будь ласка, введіть URL сторінки Google Sites з підписаним документом.")  # Попередження про відсутність URL
            return

        signed_pdf_data = PDFHandler.load_pdf_from_url(url)  # Завантажуємо підписаний PDF
        if not signed_pdf_data:  # Якщо не вдалося завантажити PDF
            return
```

filename = "google_sites_signed_document.pdf" # Встановлюємо ім'я для збереження

try:

with open(filename, 'wb') as f: # Відкриваємо файл для запису в бінарному режимі

f.write(signed_pdf_data) # Записуємо дані PDF у файл

messagebox.showinfo("Завантаження завершено", f"Підписаний PDF збережено як: {filename}") # Інформуємо про успішне збереження

except Exception as e: # Обробка виключень

messagebox.showerror("Помилка", f"Не вдалося зберегти підписаний PDF: {e}") # Показуємо повідомлення про помилку

Відкриваємо веб-сайт перевірки

verification_url = "https://id.gov.ua/verify-widget/v20240221/?address=https://czo.gov.ua&language=ua" # URL для перевірки

webbrowser.open(verification_url) # Відкриваємо URL у веб-браузері

def export_report(self):

"""Експортує результати порівняння як звіт у DOCX файл."""

report_data = self.comparator.get_report_data() # Отримуємо дані для звіту

if not report_data: # Якщо звіт не вдалося створити

messagebox.showwarning("Попередження", "Не вдалося створити звіт, будь ласка, спочатку виконайте порівняння.") # Попередження про відсутність звіту

return

report_file = filedialog.asksaveasfilename(defaultextension=".docx", filetypes=[("Word documents", "*.docx")]) # Діалог для збереження звіту

```
            if not report_file:  # Якщо файл не вибрано
                return

            try:
                doc = Document()  # Створюємо новий документ Word
                doc.add_heading('Звіт про порівняння документів', level=1)  # Додаємо заголовок
                doc.add_paragraph(report_data)  # Додаємо текст звіту
                doc.save(report_file)  # Зберігаємо документ
                messagebox.showinfo("Успішно", f"Звіт збережено як: {report_file}")  # Інформуємо про успішне збереження
            except Exception as e:  # Обробка виключень
                messagebox.showerror("Помилка", f"Не вдалося зберегти звіт: {e}")  # Показуємо повідомлення про помилку

        def compare_documents(self):
            """Порівнює оригінальний документ з підписаним документом."""
            original_content = None  # Ініціалізація змінної для оригінального вмісту
            try:
                with open(self.filepath, 'rb') as file:  # Відкриваємо файл оригінального документа
                    original_pdf_data = file.read()  # Читаємо вміст PDF
                    original_content = PDFHandler.extract_text_from_pdf(original_pdf_data)  # Витягуємо текст з PDF
            except Exception as e:  # Обробка виключень
                messagebox.showerror("Помилка", f"Не вдалося завантажити оригінальний PDF: {e}")  # Показуємо повідомлення про помилку
                return
```

```
            url = self.url_entry.get()  # Отримуємо URL з поля вводу
            signed_pdf_data = PDFHandler.load_pdf_from_url(url)  # Завантажуємо підписаний PDF
            if not signed_pdf_data:  # Якщо не вдалося завантажити PDF
                return

            signed_content = PDFHandler.extract_text_from_pdf(signed_pdf_data)  # Витягуємо текст з підписаного PDF
            if original_content and signed_content:  # Якщо обидва вмісти успішно витягнуті
                self.comparator.analyze_differences(original_content, signed_content)  # Порівнюємо вмісти


        if __name__ == "__main__":  # Точка входу програми
            root = tk.Tk()  # Створюємо головне вікно
            app = DocumentComparisonApp(root)  # Ініціалізуємо додаток
            root.mainloop()  # Запускаємо головний цикл подій
```