Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«__»_____2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)
спеціальність *125 Кібербезпека та захист інформації*
(код і назва спеціальності)
освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Методи захисту даних з використанням кваліфікаційного електронного підпису»

Виконавець: студент ІІ курсу, групи КБм-21

_____ **Анатолій ЛІНІЙЧУК**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій ТОЛЮПА	
Нормоконтроль	Юрій БАБЕНКО	

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«25» жовтня 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека та захист інформації*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача КБм-21 Лінійчука Анатолія Олександровича
(група) (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Методи захисту даних з використанням кваліфікаційного електронного підпису

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 4 від 24.10.2024 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес захисту цифрових даних з використанням кваліфікованого електронного підпису
Предмет досліджень	Методи та технології управління кваліфікованим електронним підписом, підписання та перевірки документів.
Мета	Розробка системи захисту даних з використанням кваліфікованого електронного підпису для ефективного управління ключами, підписання та перевірки документів
Вихідні дані для проведення роботи	Існуючі методи та технології використання кваліфікованого електронного підпису, криптографічні алгоритми та стандарти безпеки

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	Розробка інтегрованої системи управління кваліфікованим електронним підписом з функціями генерації, імпорту, підписання та перевірки документів
Практична цінність	Створення системи для захисту цифрових даних, яка підвищує безпеку та ефективність документообігу в бізнесі, державному управлінні та освіті

4. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	25.10.2024 – 29.12.2024
Аналіз літературних джерел	30.12.2024 – 12.02.2024
Ознайомлення з сучасними технологіями КЕП	13.02.2024 – 21.02.2024
Розгляд нормативно-правових актів, що регулюють використання КЕП	22.02.2024 – 26.02.2024
Дослідження загроз і вразливостей у сфері електронного підпису	27.02.2024 – 04.03.2024
Аналіз рекомендацій щодо підвищення безпеки КЕП	05.03.2024 – 10.03.2024
Дослідження існуючих систем і сервісів КЕП	11.03.2024 – 17.03.2024
Реалізація та інтеграція алгоритмів КЕП	18.03.2024 – 19.03.2024
Тестування системи та оцінка її ефективності	20.03.2024 – 17.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	18.04.2024 – 25.04.2024
Оформлення пояснювальної записки згідно методичних рекомендацій	26.04.2024 – 15.05.2025
Подача пакету документів на розгляд ЕК	15.05.2025 - 19.05.2025

Завдання видав _____ Сергій ТОЛЮПА
(підпис) (Ім’я, ПРІЗВИЩЕ)

Завдання прийняв
до виконання _____ Анатолій Лінійчук
(підпис) (Ім’я, ПРІЗВИЩЕ)

Дата видачі завдання: 25.10.2024 р.
Термін подання кваліфікаційної роботи до ЕК 19.05.2025 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Розробка системи захисту даних з використанням кваліфікованого електронного підпису»: 98 сторінки, 34 рисунки та 20 таблиць, 40 літературних джерел.

Об’єкт дослідження – процес захисту цифрових даних з використанням кваліфікованого електронного підпису (КЕП).

Мета роботи – розробка інформаційно-комунікаційної системи для управління КЕП, яка забезпечує ефективне підписання, перевірку та зберігання документів.

Методи дослідження – системний підхід до проектування архітектури системи, методи криптографічного захисту (RSA, ECDSA), методи тестування ефективності системи, інструменти веб-розробки для створення інтерфейсу.

У роботі проаналізовано сучасні виклики у сфері захисту цифрових даних та методи використання КЕП. Проведено порівняльний аналіз криптографічних алгоритмів (RSA, DSA, ECDSA) та сервісів надання послуг КЕП в Україні та світі. Запропоновано систему, яка інтегрує функції генерації ключів, підписання документів, перевірки підписів та управління даними, забезпечуючи їх цілісність і доступність.

Наукова новизна: розроблено систему управління КЕП з інтегрованими функціями генерації, імпорту, підписання та перевірки документів, а також зручним інтерфейсом для неспеціалістів, що спрощує роботу з цифровими підписами.

Актуальність теми: У умовах зростання обсягів цифрових даних захист інформації є критично важливим. КЕП забезпечує юридичну силу документів, але потребує ефективних систем управління. Запропонована система вирішує проблеми інтеграції КЕП у різні платформи, підвищує безпеку транзакцій і спрощує документообіг, що є важливим для бізнесу, державного управління та освіти.

Ключові слова: кваліфікований електронний підпис, захист даних, криптографія, RSA, ECDSA, документообіг, інформаційна безпека.

ЗМІСТ

ВСТУП

Оцінка сучасного стану проблеми: У сучасному світі, де цифровізація охоплює всі сфери життя, захист інформації стає однією з найважливіших задач. Зростання обсягів цифрових даних, їх передача та зберігання потребують надійних механізмів забезпечення автентичності, цілісності та конфіденційності. Одним із ключових інструментів для вирішення цих завдань є кваліфікований електронний підпис (КЕП), який надає юридичну силу цифровим документам та забезпечує їхню неспростовність.

Аналіз вітчизняної та зарубіжної наукової літератури свідчить про значний прогрес у сфері розробки та впровадження КЕП. Серед провідних вчених, які внесли вагомий внесок у розвиток цієї галузі, можна відзначити праці В. Задіраки, О. Кузнецова, Ю. Яремчука, а також зарубіжних дослідників, таких як I. Cox, R. Ohbuchi та інших. Незважаючи на існуючі розробки, залишаються актуальними питання щодо підвищення ефективності використання КЕП, зокрема впровадження нових методів підписання та перевірки документів, оптимізації процесів зберігання та управління ключами, а також забезпечення сумісності з різними платформами та стандартами.

Актуальність дослідження обумовлена необхідністю подолання протиріч між зростаючими вимогами до захисту цифрових даних та обмеженими ресурсами для їх реалізації. Зокрема, існують проблеми, пов’язані зі складністю інтеграції КЕП у різні системи, недостатньою інформованістю користувачів про можливості та переваги електронного підпису, а також необхідністю забезпечення сумісності з міжнародними стандартами.

Перспективність теми полягає в тому, що розвиток технологій КЕП відкриває нові можливості для автоматизації документообігу, підвищення ефективності бізнес-процесів та забезпечення безпеки цифрових транзакцій. Удосконалення методів роботи з КЕП дозволить зробити їх більш доступними та зручними для широкого кола користувачів.

Метою дослідження є розробка системи захисту даних з використанням кваліфікованого електронного підпису, яка забезпечує ефективне управління ключами, підписання та перевірку документів, а також зберігання підписаних даних.

Для досягнення поставленої мети визначено такі завдання:

– Провести аналіз існуючих методів та технологій використання КЕП.

– Розробити архітектуру системи для управління КЕП, підписання та перевірки документів.

– Реалізувати механізми генерації та імпорту ключів КЕП.

– Забезпечити функціональність підписання документів з використанням КЕП.

– Розробити алгоритми перевірки підписів та цілісності документів.

– Створити інтерфейс для зручного управління документами та ключами.

– Провести тестування системи та оцінити її ефективність.

Об’єктом дослідження є процес захисту цифрових даних з використанням кваліфікованого електронного підпису.

Предметом дослідження є методи та технології управління КЕП, підписання та перевірки документів.

Для вирішення поставлених завдань у роботі використано такі методи:

– системний підхід для аналізу та проектування архітектури системи;

– методи криптографічного захисту даних для реалізації функцій підписання та перевірки документів;

– методи тестування для оцінки ефективності розробленої системи;

– інструменти веб-розробки для створення інтерфейсу користувача.

Наукова новизна дослідження полягає в наступному:

– розроблено систему управління КЕП, яка інтегрує функції генерації, імпорту, підписання та перевірки документів;

– запропоновано механізми зберігання та управління підписаними документами, що забезпечують їхню цілісність та доступність;

– створено інтерфейс користувача, який спрощує роботу з КЕП для неспеціалістів.

Практична цінність роботи полягає в розробці системи, яка може бути використана для захисту цифрових даних у різних сферах, зокрема в бізнесі, державному управлінні та освіті. Система дозволяє ефективно керувати КЕП, підписувати та перевіряти документи, що сприяє підвищенню рівня безпеки та зручності роботи з цифровими даними.

Результати дослідження були представлені на засіданнях кафедри та на групових зборах з питань інформаційної безпеки.

Таким чином, розроблена система є важливим кроком у напрямку підвищення ефективності використання КЕП та забезпечення безпеки цифрових даних у сучасних умовах.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ТА ПОТРЕБ У СФЕРІ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ

1.1 Аналіз потреб компаній у використанні кваліфікованого електронного підпису (КЕП)

Сучасний бізнес, зокрема юридичні компанії, функціонує в умовах стрімкого розвитку інформаційних технологій, що кардинально змінюють підходи до організації роботи. У цьому контексті кваліфікований електронний підпис (КЕП) набуває статусу ключового інструменту, який забезпечує безпеку, достовірність та юридичну силу документів в електронному форматі [1]. Впровадження КЕП відповідає викликам цифрової епохи, сприяючи оптимізації бізнес-процесів, підвищенню ефективності та зміцненню конкурентних позицій компаній на ринку. Його значення важко переоцінити, адже КЕП не лише спрощує документообіг, але й створює надійну основу для захисту інформації та побудови довіри між партнерами.

Юридична діяльність характеризується інтенсивним документообігом, що включає підготовку, підписання та обмін численними документами, такими як договори, судові позови, адвокатські запити чи нотаріальні акти. Кожен із цих документів потребує офіційного статусу, який залежить від автентичності підпису. Традиційний паперовий документообіг, попри свою поширеність, має низку недоліків: він потребує значних часових і фінансових ресурсів на друк, доставку, перевірку та архівування документів. Крім того, паперові носії вразливі до втрати, пошкодження чи фальсифікації. КЕП дозволяє усунути ці проблеми, забезпечуючи швидке підписання документів в електронному вигляді та їхнє надійне зберігання [2]. Завдяки цьому компанії можуть економити ресурси, прискорювати бізнес-процеси та мінімізувати ризики, пов’язані з людським фактором, наприклад, помилками при заповненні документів чи втратою важливих матеріалів.

Однією з ключових переваг КЕП є його здатність гарантувати автентичність, цілісність і неспростовність підписаних документів. Автентичність підтверджує, що документ створений саме тією особою, яка вказана як підписант. Цілісність забезпечує, що документ не зазнав змін після підписання, а неспростовність унеможливлює відмову підписанта від свого підпису. Ці властивості мають особливе значення в юридичній практиці, де будь-яка неточність чи сумнів у достовірності документа може призвести до серйозних правових наслідків, наприклад, визнання договору недійсним чи програшу в судовій справі. У контексті зростання кіберзагроз, таких як фішинг, злами чи несанкціонований доступ до даних, КЕП виступає надійним інструментом захисту конфіденційної інформації, що є критично важливим для юридичних компаній, які працюють із чутливими даними клієнтів [3].

Ще одним важливим аспектом є адаптація до сучасних умов роботи, зокрема зростання популярності дистанційних форматів взаємодії. Пандемія COVID-19 стала каталізатором цифрової трансформації, змусивши компанії швидко переходити на віддалений режим роботи та впроваджувати електронні сервіси. У таких умовах КЕП став незамінним інструментом для оперативного узгодження документів, укладання угод і подання звітності до державних установ. Наприклад, адвокат може підписати та надіслати запит до реєстру чи суду, не залишаючи офісу чи навіть дому, що значно економить час і ресурси. Така гнучкість робить КЕП не просто зручним, але й стратегічно важливим рішенням для компаній, які прагнуть залишатися ефективними в умовах динамічного бізнес-середовища.

Крім того, використання КЕП відповідає вимогам міжнародних стандартів, що є особливо актуальним для юридичних компаній, які співпрацюють із закордонними партнерами або беруть участь у транскордонних операціях. У Європейському Союзі, наприклад, КЕП регулюється стандартом eIDAS, який визначає вимоги до електронних підписів і забезпечує їхню юридичну силу в усіх країнах-членах ЄС [4]. Для українських компаній, які прагнуть вийти на міжнародний ринок або працювати з європейськими клієнтами, відповідність таким стандартам стає не лише конкурентною перевагою, але й необхідною умовою для укладання контрактів і

виконання юридичних зобов’язань. Це сприяє зміцненню довіри між партнерами та полегшує інтеграцію в глобальну економіку.

Інтеграція КЕП у внутрішні інформаційні системи компаній також відіграє важливу роль. Сучасні програмні платформи для управління документообігом дозволяють автоматизувати рутинні процеси, такі як створення шаблонів документів, їхнє підписання, відправлення та архівування. КЕП гармонійно доповнює ці системи, забезпечуючи безпечне підписання документів і їхнє зберігання в електронному вигляді. Наприклад, юридична компанія може налаштувати автоматизоване підписання типових договорів, що значно скорочує час на обробку документів і знижує ймовірність помилок. Такі рішення не лише підвищують продуктивність, але й сприяють прозорості внутрішніх процесів, що є важливим для компаній із великою кількістю клієнтів чи складною організаційною структурою [5].

Відповідність національному законодавству є ще одним вагомим аргументом на користь впровадження КЕП. В Україні використання електронного підпису регулюється Законом "Про електронні довірчі послуги", який встановлює чіткі правила для створення, використання та визнання КЕП [6]. Цей закон забезпечує юридичну силу документів, підписаних КЕП, що дозволяє юридичним компаніям безперешкодно взаємодіяти з державними органами, такими як суди, податкові служби чи реєстри. Наприклад, подання електронних звітів чи заяв до державних установ стало значно простішим завдяки КЕП, що економить час і знижує бюрократичне навантаження. Крім того, державні органи дедалі активніше впроваджують електронні сервіси, що робить КЕП невід’ємною частиною взаємодії бізнесу з державою.

Варто також зазначити, що впровадження КЕП сприяє підвищенню рівня довіри клієнтів до юридичних компаній. У сучасному світі, де цифрові технології проникають у всі сфери життя, клієнти очікують від компаній використання сучасних і безпечних рішень. Наявність КЕП сигналізує про те, що компанія дбає про захист даних, дотримується високих стандартів безпеки та готова до роботи в цифровому форматі. Це особливо важливо для залучення молодих клієнтів і компаній, які активно використовують цифрові технології у своїй діяльності.

Окремої уваги заслуговує економічний ефект від використання КЕП. Скорочення витрат на папір, друк, поштові послуги та фізичне зберігання документів дозволяє компаніям спрямовувати зекономлені кошти на розвиток бізнесу, навчання персоналу чи впровадження нових технологій. Крім того, швидкість обробки документів із КЕП дає змогу юридичним компаніям обслуговувати більше клієнтів за той самий час, що безпосередньо впливає на їхню прибутковість. У довгостроковій перспективі ці переваги сприяють не лише фінансовій вигоді, але й зміцненню репутації компанії як сучасної та клієнтоорієнтованої організації.

Нарешті, використання КЕП сприяє екологічній відповідальності. У час, коли питання сталого розвитку стають дедалі актуальнішими, скорочення використання паперу та транспортних перевезень для доставки документів допомагає зменшити екологічний слід компаній. Юридичні фірми, які впроваджують КЕП, демонструють свою соціальну відповідальність, що позитивно впливає на їхній імідж і приваблює клієнтів, які цінують екологічні ініціативи.

Отже, потреба юридичних компаній у використанні кваліфікованого електронного підпису зумовлена низкою факторів, серед яких економічна ефективність, безпека даних, відповідність міжнародним і національним стандартам, а також адаптація до сучасних умов роботи. КЕП не лише спрощує документообіг, але й відкриває нові можливості для автоматизації процесів, підвищення продуктивності та зміцнення довіри клієнтів. У контексті цифрової трансформації КЕП стає не просто інструментом, а стратегічним елементом, який допомагає юридичним компаніям залишатися конкурентоспроможними, гнучкими та орієнтованими на майбутнє. Його впровадження є важливим кроком до створення прозорого, технологічного та безпечного бізнес-середовища, яке відповідає викликам сучасного світу.

1.2 Аналіз існуючих алгоритмів КЕП (RSA, DSA, ECDSA тощо)

Кваліфікований електронний підпис (КЕП) є важливим інструментом для забезпечення цілісності, автентичності та захищеності електронних документів. В

основі цього механізму лежать криптографічні алгоритми, які дозволяють створювати підпис, що забезпечує підтвердження авторства документа, а також захист від його змін. Сучасні алгоритми КЕПґрунтуються на різних криптографічних підходах, що зумовлює їх особливості та можливості. Важливою частиною проекту є розгляд трьох основних алгоритмів: RSA, DSA та ECDSA.

1.2.1 Алгоритм RSA

RSA (Рівін, Шаміра, Адельман) є одним з перших криптографічних алгоритмів, широко застосовуваних для створення електронних підписів та забезпечення конфіденційності даних [7]. Його безпека базується на складності факторизації великих чисел. Алгоритм використовує пару ключів: публічний, який застосовується для перевірки підпису, і приватний, що використовується для його створення.

Серед переваг RSA варто відзначити його широке поширення та надійність, адже він є основним стандартом для криптографії з відкритим ключем. Також алгоритм відрізняється простотою реалізації та підтримкою багатьма криптографічними бібліотеками, а також можливістю використання як для шифрування, так і для створення підписів.

Однак RSA має і деякі недоліки, серед яких найбільш суттєвим є висока обчислювальна складність, що зростає із збільшенням розміру ключа. Це призводить до необхідності значних обчислювальних ресурсів для досягнення високого рівня безпеки, що може бути проблемою для мобільних або обмежених за потужністю систем.

1.2.2 Алгоритм DSA

Алгоритм DSA (Digital Signature Algorithm), розроблений урядом США, є частиною стандарту FIPS PUB 186 і призначений для створення цифрових підписів [8]. Він використовує принципи дискретного логарифмування для генерації та

перевірки підписів. На відміну від RSA, DSA спеціалізується виключно на підписах і не підтримує шифрування.

Основною перевагою DSA є його більша ефективність порівняно з RSA при використанні однакових розмірів ключів. Це дозволяє зменшити навантаження на систему, що є важливим для великих обсягів даних і обмежених ресурсів. Однак DSA має й деякі недоліки: для досягнення високого рівня безпеки необхідні великі розміри ключів та складні обчислення. Крім того, алгоритм не є оптимальним для коротших ключів, оскільки їх безпека може бути скомпрометована через атаки на довжину ключа.

1.2.3 Алгоритм ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) є альтернативою RSA та DSA, використовуючи еліптичні криві для генерації криптографічних ключів. Однією з головних переваг ECDSA є високий рівень безпеки при значно менших розмірах ключів [9]. Наприклад, ключ довжиною 256 біт у ECDSA забезпечує таку ж безпеку, як 3072 біти в RSA. Завдяки цьому, ECDSA дозволяє знижувати вимоги до пам'яті та обчислювальних ресурсів, підвищуючи ефективність обробки.

Протоколи, як SSL/TLS для безпечних комунікацій в інтернеті, підтримують ECDSA, що робить його важливою складовою сучасних стандартів безпеки. Однак, незважаючи на ці переваги, ECDSA має деякі недоліки. Він потребує більших обчислювальних потужностей та високої точності математичних операцій, що може створювати труднощі при реалізації на менш потужних пристроях. Крім того, для правильного застосування еліптичних кривих важливо враховувати параметри та їх вибір для конкретних умов.

Кожен із розглянутих алгоритмів має свої сильні та слабкі сторони, і вибір конкретного алгоритму для використання в електронному цифровому підписі залежить від багатьох факторів, таких як рівень безпеки, ефективність, вимоги до ресурсів та сумісність з іншими системами. RSA є класичним вибором для широкого спектра застосувань, хоча його обчислювальна складність обмежує використання в

деяких випадках. DSA підходить для завдань, пов’язаних лише з підписами, і є ефективним у певних умовах, але також потребує оптимізації для досягнення належного рівня безпеки. ECDSA, у свою чергу, є сучасним рішенням, яке забезпечує високу безпеку при менших витратах на ресурси, що робить його перспективним для використання в майбутньому.

1.3 Огляд сучасних сервісів надання послуг КЕП в Україні та світі

Кваліфікований електронний підпис (КЕП) є важливим інструментом для забезпечення юридичної сили та безпеки електронних документів. Він став необхідністю для забезпечення ефективної взаємодії між підприємствами, державними установами та громадянами в умовах цифровізації бізнес-процесів. Сучасні технології, що забезпечують функціонування КЕП, активно розвиваються, і на ринку існує велика кількість сервісів, які пропонують різноманітні рішення для організацій та фізичних осіб. Огляд таких сервісів дозволяє оцінити їх можливості та вплив на ефективність діяльності юридичних компаній, підприємств і організацій у сфері інформаційно-комунікаційних технологій.

1.3.1 Сервіси надання послуг КЕП в Україні

В Україні працюють кілька основних сервісів для отримання та використання кваліфікованого електронного підпису (КЕп). Одним із провідних є Державне агентство з питань електронного урядування, яке через акредитовані центри сертифікації ключів забезпечує громадянам та організаціям можливість отримати КЕп, що має юридичну силу при взаємодії з державними органами. Сервіси, такі як платформа "Дія", дозволяють отримати електронні підписи для документів, що використовуються в державних та юридичних процесах.

Системи КЕп в Україні мають різні рівні безпеки, зокрема найвищий забезпечується кваліфікованими сертифікатами відкритих ключів, які надаються акредитованими центрами, як-от ЦСК "Україна" (рис. 1.1) та АКЦІОНЕРНЕ

ТОВАРИСТВО КОМЕРЦІЙНИЙ БАНК «ПРИВАТБАНК» (рис 1.2) [10][11]. Ці сервіси підтримують використання електронних підписів для юридичних, фінансових операцій та забезпечення безпеки електронної звітності. Для бізнесу та юридичних компаній важливою є інтеграція КЕП в програмні продукти, такі як системи електронного документообігу та Інтернет-банкінг, що значно спрощує документообіг і підвищує оперативність процесів.

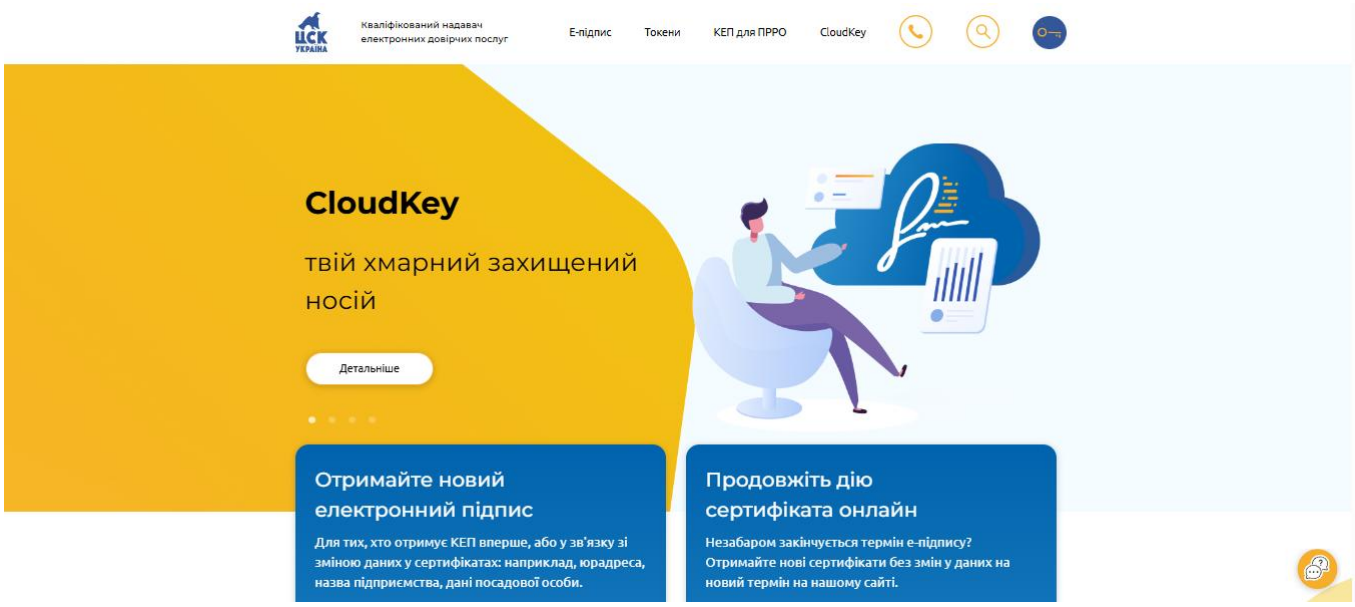

Рисунок 1.1 – Головна сторінка ЦСК "Україна" [10]

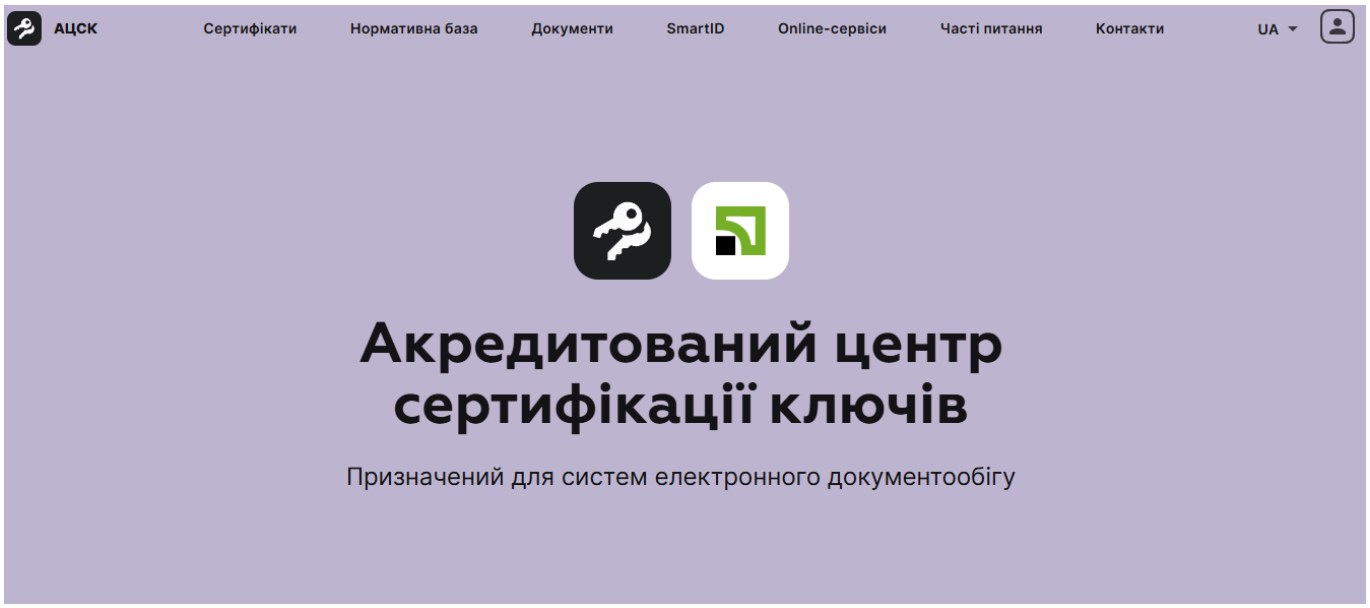

Рисунок 1.2 – Головна сторінка АКЦІОНЕРНЕ ТОВАРИСТВО КОМЕРЦІЙНИЙ БАНК «ПРИВАТБАНК» [11]

1.3.2 Сервіси надання послуг КЕП у світі

У глобальному масштабі регулювання електронних підписів здійснюється не лише державами, а й міжнародними стандартами, зокрема, регламентом eIDAS, що встановлює норми для електронних підписів та ідентифікаційних засобів в Європейському Союзі. Це дозволяє створювати сервіси, які забезпечують високий рівень захисту та відповідають міжнародним вимогам. Одним з найпоширеніших сервісів є DocuSign (рис. 1.3), платформа для підписання документів, яка активно використовується в бізнесі та державному управлінні [12]. Також популярним є Adobe Sign (рис. 1.4), що інтегрується з продуктами Adobe та забезпечує підписання документів на різних пристроях, гарантуючи безпеку та відповідність міжнародним стандартам [13]. В США широко застосовуються рішення від GlobalSign (рис. 1.5) і Symantec (рис. 1.6) для підписання документів в корпоративному середовищі, зокрема для контрактів та транзакцій з підвищеними вимогами до безпеки [14][15].

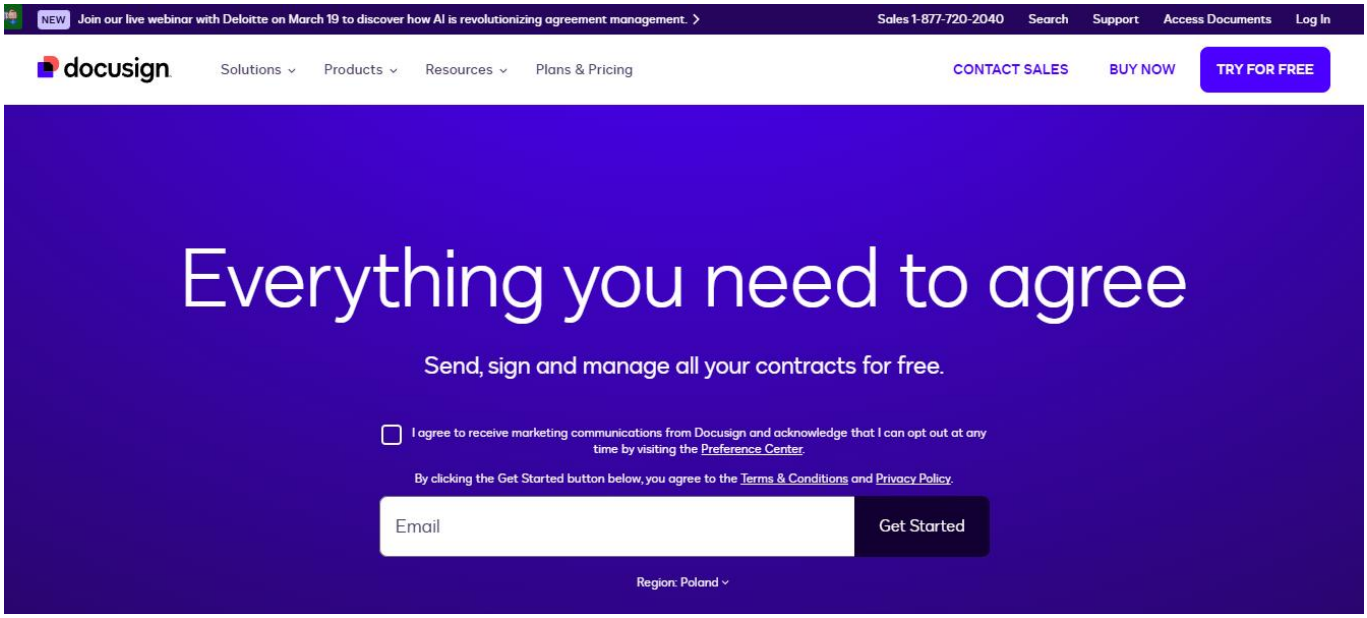

Рисунок 1.3 – Головна сторінка DocuSign [12]

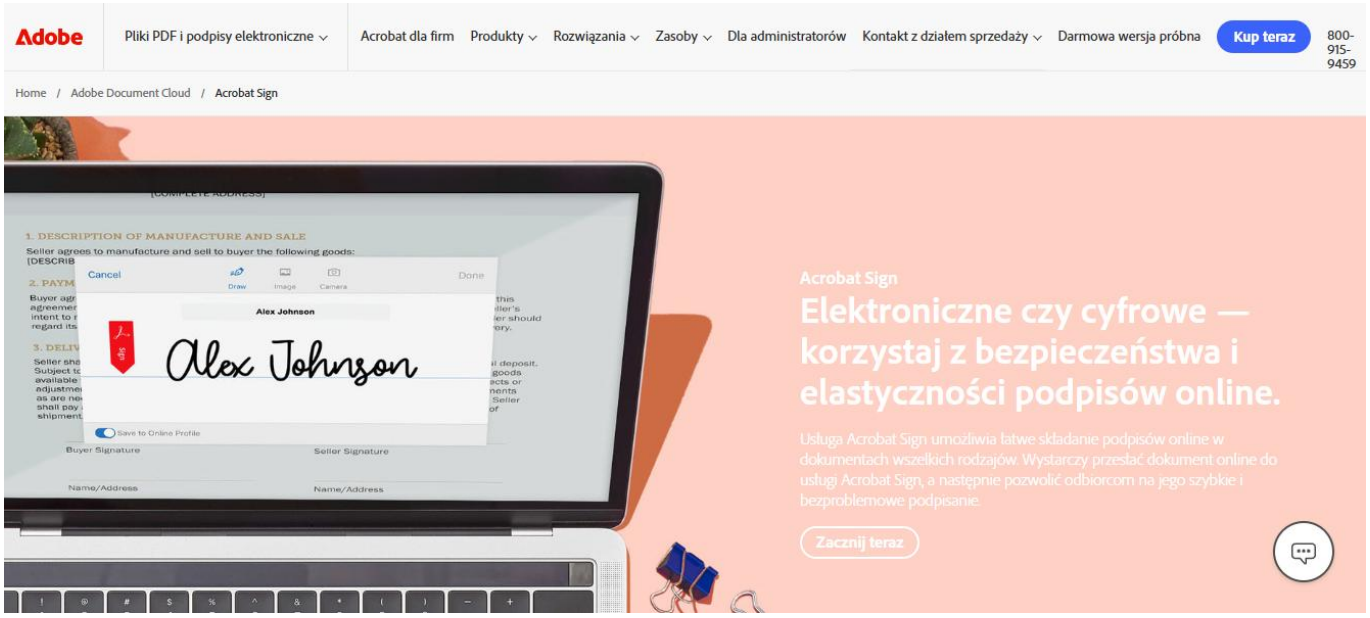

Рисунок 1.4 – Головна сторінка Adobe Sign [13]

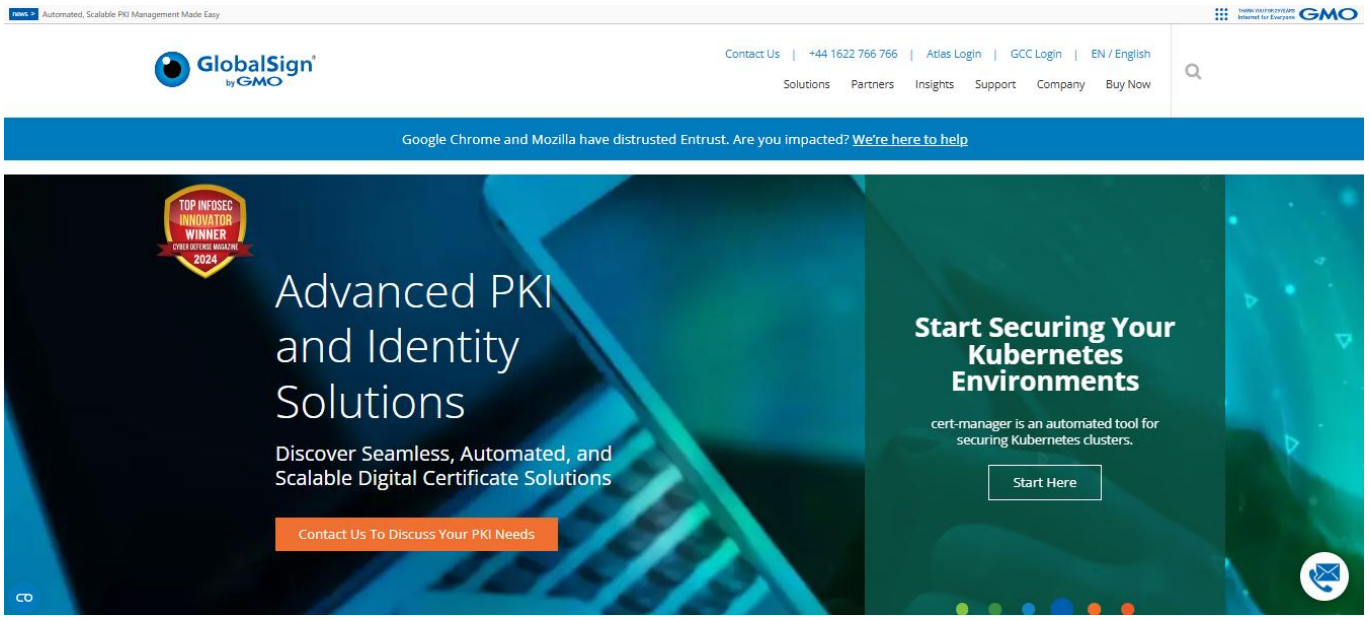

Рисунок 1.5 – Головна сторінка GlobalSign [14]

Рисунок 1.6 – Головна сторінка Symantec [15]

Загалом, наявність сучасних сервісів надання послуг електронного цифрового підпису як в Україні, так і в світі, є важливим фактором для забезпечення безпеки електронних транзакцій, ефективності документообігу та юридичної сили цифрових підписів. В Україні існують декілька надійних акредитованих центрів сертифікації, які надають широкий спектр послуг для підприємств і громадян. У світі використовуються більш глобальні рішення, орієнтовані на інтеграцію з різноманітними платформами та бізнес-системами, що дозволяє забезпечити максимально зручні та безпечні умови для підписання електронних документів на міжнародному рівні. Розвиток цих технологій сприяє зниженню бюрократичних бар'єрів і підвищенню оперативності в обробці документації.

1.4 Обґрунтування вибору технологій для побудови кастомного алгоритму КЕП

У сучасному світі, де цифрова трансформація охоплює всі сфери суспільного життя, кваліфікований електронний підпис (КЕП) відіграє ключову роль у забезпеченні безпеки, конфіденційності та юридичної значущості документів. Особливо це стосується юридичного сектору, де захист інформації та підтвердження

авторства документів є критично важливими. КЕП дозволяє організаціям, клієнтам і державним установам взаємодіяти безпечно та ефективно, мінімізуючи ризики підробки чи несанкціонованого доступу. Юридична сила КЕП ґрунтується на відповідності національним і міжнародним стандартам електронної комунікації, що робить його незамінним інструментом у правовій практиці. Розробка кастомного алгоритму КЕП вимагає ретельного вибору технологій, які забезпечать надійність, швидкість і сумісність системи з іншими інформаційними платформами. У цьому контексті вибір технологій для реалізації системи, представленої у коді проєкту, базується на поєднанні криптографічної безпеки, продуктивності, інтеграційних можливостей і масштабованості.

1.4.1 Вибір криптографічних алгоритмів

Для побудови кастомного алгоритму КЕП важливо враховувати кілька факторів, зокрема вимоги до криптографії, швидкості операцій та можливості інтеграції з іншими системами [16]. У коді проєкту використовується алгоритм RSA (Rivest-Shamir-Adleman), який є одним із найбільш поширених стандартів для шифрування та створення електронних підписів [17]. RSA забезпечує високий рівень безпеки завдяки використанню асиметричної криптографії, де приватний ключ застосовується для підписання, а публічний — для перевірки підпису. Однак RSA має певні обмеження, зокрема щодо швидкості обробки великих обсягів даних, що може створювати труднощі в системах із високими вимогами до продуктивності.

Альтернативою RSA є еліптична криптографія (ECC), яка набирає популярності завдяки своїй ефективності. Алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm), що базується на еліптичних кривих, дозволяє досягти порівнянного рівня безпеки з RSA, використовуючи значно коротші ключі [18]. Наприклад, ключ ECDSA довжиною 256 біт забезпечує рівень безпеки, еквівалентний 3072-бітному ключу RSA, що суттєво зменшує обчислювальні витрати та підвищує швидкість обробки. У контексті кастомного алгоритму КЕП використання ECC могло б оптимізувати продуктивність системи, особливо для клієнтських додатків, таких як

Streamlit, які потребують швидкої обробки запитів. Однак у коді проєкту вибір RSA обґрунтований його широкою підтримкою в різних системах і стандартах, таких як X.509, що забезпечує сумісність із більшістю платформ для роботи з електронними підписами.

Крім того, для забезпечення цілісності документів у коді застосовується хеш-функція SHA-256 у поєднанні з механізмом PSS (Probabilistic Signature Scheme). Це дозволяє створювати надійні підписи, стійкі до атак, спрямованих на підробку документів. Вибір SHA-256 є оптимальним, оскільки ця функція широко визнана в криптографічній спільноті та відповідає сучасним стандартам безпеки. Таким чином, комбінація RSA, SHA-256 і PSS у реалізованій системі забезпечує баланс між криптографічною надійністю та практичною реалізацією.

1.4.2 Інтеграція з існуючими системами

Розробка системи КЕП, представленої в коді, базується на сучасному технологічному стеку, який включає Python, бібліотеку cryptography, фреймворки Flask і Streamlit, а також базу даних SQLite. Вибір Python як основної мови програмування обґрунтований її універсальністю, великою кількістю бібліотек для роботи з криптографією та простотою реалізації складних алгоритмів. Бібліотека cryptography, використана в коді, забезпечує доступ до перевірених криптографічних примітивів, таких як RSA, SHA-256 і X.509, що дозволяє створювати безпечні підписи та сертифікати без необхідності розробки низькорівневих криптографічних функцій.

На серверній стороні фреймворк Flask використовується для створення REST API, яке обробляє запити на генерацію ключів, підписання документів і перевірку підписів. Flask є легким і гнучким рішенням, яке ідеально підходить для реалізації мікросервісної архітектури, що забезпечує швидке розгортання та легке масштабування системи. На клієнтській стороні Streamlit застосовується для створення інтерактивного веб-інтерфейсу, який дозволяє користувачам зручно взаємодіяти з системою КЕП. Streamlit спрощує розробку інтерфейсу завдяки

декларативному підходу, що дозволяє зосередитися на логіці програми, а не на деталях фронтенд-розробки.

Для зберігання даних, таких як ключі, сертифікати та підписані документи, використовується SQLite — легка реляційна база даних, яка не вимагає складного налаштування та підходить для демонстраційних проєктів. У реальних системах із великою кількістю користувачів SQLite може бути замінена на більш продуктивні рішення, такі як PostgreSQL або MySQL, для забезпечення масштабованості. Вибір SQLite у коді проєкту є виправданим для прототипу, оскільки він забезпечує простоту розгортання та достатню функціональність для зберігання даних.

1.4.3 Підтримка багатофункціональності та масштабованості

Для юридичних компаній критично важливою є можливість інтеграції системи КЕП із наявними інформаційними платформами, такими як системи управління документообігом, бухгалтерські програми чи державні портали, наприклад, "Дія" в Україні. У реалізованій системі підтримка стандарту X.509 для сертифікатів забезпечує сумісність із більшістю сучасних платформ, які використовують цей формат для аутентифікації та перевірки підписів. Сертифікати X.509, згенеровані в коді, включають необхідні атрибути, такі як термін дії та ідентифікатор користувача, що дозволяє їх використовувати в різних юрисдикціях.

Крім того, REST API, реалізоване через Flask, забезпечує гнучкість інтеграції з іншими системами. Наприклад, зовнішні програми можуть надсилати запити на підписання чи перевірку документів, використовуючи стандартні HTTP-методи. Це дозволяє інтегрувати систему КЕП із корпоративними ERP-системами або платформами для онлайн-взаємодії з державними органами. Важливим аспектом є також підтримка юридичної сили підписів у різних правових системах. У коді враховано вимоги до створення самопідписаних сертифікатів, які для демонстраційних цілей замінюють сертифікати, видані кваліфікованими надавачами довірчих послуг. У реальних умовах система може бути адаптована для роботи з

сертифікатами, виданими акредитованими центрами, що відповідає міжнародним стандартам, таким як eIDAS у Європейському Союзі.

1.4.4 Масштабованість і багатофункціональність

Розробка кастомного алгоритму КЕП передбачає врахування не лише поточних потреб, а й можливості розширення функціональності в майбутньому. У спроєктованій системі реалізована підтримка підписання документів будь-якого типу, що дозволяє працювати з різними форматами, такими як PDF, DOCX чи XML. Користувачі можуть завантажувати файли через інтерфейс Streamlit, а система автоматично обробляє їх, створюючи цифровий підпис і зберігаючи його разом із документом. Крім того, реалізована функція перевірки підписів, яка дозволяє переконатися в автентичності та цілісності документа, що є важливим для юридичних процедур.

Для забезпечення масштабованості система використовує модульну архітектуру, де клієнтська та серверна частини відокремлені. Це дозволяє, наприклад, розгортати сервер на потужніших апаратних платформах у разі зростання кількості користувачів. У коді також передбачено зберігання списку документів для кожного користувача, що полегшує керування підписаними файлами та їх пошук. У реальних умовах для підвищення продуктивності можна додати кешування запитів, асинхронну обробку або розподілену архітектуру з використанням хмарних сервісів.

Багатофункціональність системи проявляється також у підтримці імпорту та експорту ключів КЕП у форматі JSON. Це дозволяє користувачам переносити свої ключі між різними пристроями чи системами, зберігаючи доступ до підписаних документів. У майбутньому система може бути розширена для підтримки багаторівневих підписів, де кілька користувачів можуть підписувати один документ, або для інтеграції з апаратними модулями безпеки (HSM) для захисту приватних ключів.

Правильний вибір технологій для побудови кастомного алгоритму КЕП є запорукою створення надійної, безпечної та зручної системи для юридичних

компаній. У коді проєету поєднуються перевірені криптографічні алгоритми, такі як RSA і SHA-256, із сучасними інструментами розробки, зокрема Python, Flask і Streamlit. Це забезпечує баланс між безпекою, продуктивністю та простотою використання. Врахування таких факторів, як сумісність із стандартами X.509, інтеграція з іншими системами, підтримка різних типів документів і потенціал для масштабування, дозволяє системі відповідати як поточним, так і майбутнім потребам користувачів. Застосування таких технологій гарантує високий рівень захисту даних, юридичну значущість підписів і зручність роботи з електронними документами, що є критично важливим у сучасному юридичному середовищі.

Висновки до Розділу 1

Аналіз потреб юридичних компаній у використанні електронного цифрового підпису (КЕП) показав, що його впровадження є необхідним для підвищення ефективності документообігу та забезпечення високого рівня безпеки. У сучасних умовах цифровізації КЕП стає ключовим елементом у управлінні корпоративними ризиками, адже він гарантує юридичну силу електронних документів, що є важливим для належного виконання прав та обов'язків у правових відносинах. Враховуючи вимоги до конфіденційності та цілісності даних, юридичні компанії повинні використовувати технології для безпечного підписування та обміну документами.

Аналіз існуючих алгоритмів КЕП показав, що класичний RSA алгоритм, хоч і надійний, має велику обчислювальну складність, що зумовлює пошук ефективніших рішень. Алгоритм ECDSA, що використовує еліптичні криві, є перспективним вибором завдяки високій ефективності та безпеці при меншому розмірі ключів. Впровадження кастомних алгоритмів для електронного підпису є важливим кроком у забезпеченні безпеки роботи з документами в цифровому середовищі, а вибір оптимальних технологій має ґрунтуватися на криптографічній безпеці та ефективності системи.

Також було розглянуто розвиток сервісів надання послуг кваліфікованого електронного підпису (КЕП) як в Україні, так і за її межами. На сьогодні в Україні

функціонують акредитовані сертифікаційні центри, які відповідають чинним вимогам законодавства у сфері електронних довірчих послуг. Ці центри забезпечують користувачам можливість безпечно підписувати електронні документи, здійснювати ідентифікацію та автентифікацію особи, а також підтверджувати цілісність і походження даних.

Водночас у міжнародній практиці активно застосовуються більш комплексні та інтегровані технологічні рішення, орієнтовані на глобальне використання електронних підписів. Такі рішення охоплюють не лише надання підпису, але й забезпечення сумісності між різними юрисдикціями, автоматизацію процесів перевірки, зберігання та обміну цифровими документами. Це дає змогу значно підвищити ефективність цифрової взаємодії між бізнесом, державними структурами та кінцевими користувачами.

Впровадження передових технологій КЕП є важливим етапом цифрової трансформації юридичного сектору. Для юридичних компаній це відкриває можливості не лише для оптимізації внутрішніх процесів, але й для підвищення рівня довіри з боку клієнтів. Надійний захист електронних документів, гарантований за допомогою КЕП, стає ключовим фактором у забезпеченні інформаційної безпеки та зміцненні конкурентоспроможності компаній на ринку юридичних послуг. У поєднанні з інтеграцією в міжнародні електронні платформи це сприяє розширенню географії діяльності та виходу на нові ринки.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНОЇ СИСТЕМИ ДЛЯ РОБОТИ З КЕП

2.1 Функціональні вимоги до інформаційно-комунікаційної системи (ІКС)

У сучасному світі безпека інформації є критично важливим завданням для організацій, які працюють з конфіденційними даними. Одним із найефективніших засобів захисту є кваліфікований електронний підпис, що гарантує автентичність, цілісність та незаперечність документів. КЕП став невід'ємною частиною обігу юридичних та офіційних документів, зокрема в юридичних компаніях. Інформаційно-комунікаційна система для роботи з КЕП має забезпечувати управління підписанням, перевіркою та зберіганням електронних документів, а також відповідність високим вимогам безпеки. Визначення функціональних вимог є основою для створення ефективної та безпечної системи, що відповідає потребам користувачів.

Основні функціональні вимоги до ІКС для роботи з КЕП представлені на рис. 2.1.

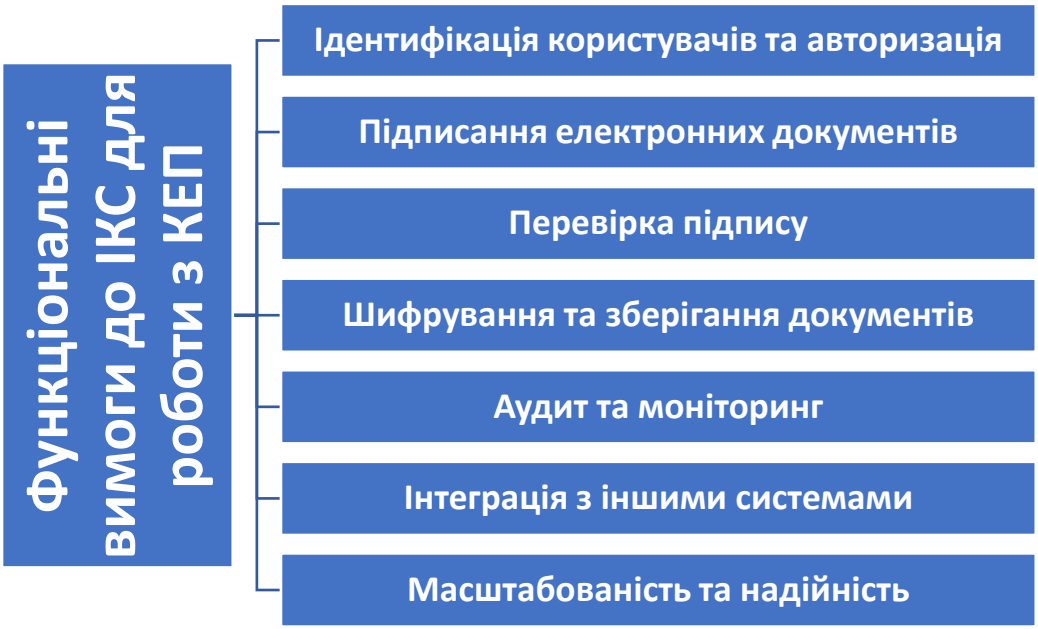

Рисунок 2.1 – Основні функціональні вимоги до ІКС для роботи з КЕП

Однією з основних вимог до ІКС є забезпечення надійної ідентифікації користувачів. Кожен користувач системи має пройти процедуру авторизації, що передбачає перевірку його особи через захищену процедуру. Важливо, щоб доступ до функцій підпису та перевірки електронного цифрового підпису отримували лише авторизовані користувачі, що мають відповідні права. Технології двофакторної авторизації або використання біометричних даних можуть значно підвищити рівень безпеки при доступі до системи.

Однією з основних функцій ІКС є можливість підписання документів за допомогою КЕП. Для цього система повинна забезпечувати зручний інтерфейс для завантаження документів та їх подальшого підпису. Користувач повинен мати змогу підписати як окремі файли, так і кілька документів одночасно, зберігаючи при цьому максимальний рівень безпеки. Підписання має здійснюватися за допомогою алгоритмів, що забезпечують високу стійкість до підробок, таких як RSA або ECDSA. Також важливо, щоб система підтримувала різні формати документів (наприклад, PDF, DOCX, TXT).

Після підписання документу система повинна мати можливість перевірити автентичність підпису та цілісність документа. Для цього користувач має мати доступ до функції перевірки підпису, яка за допомогою публічного ключа підтверджує, що підписаний документ не був змінений після підписання. ІКС повинна автоматично перевіряти відповідність підпису та документа і повідомляти користувача про результат. Перевірка підпису є критично важливою для забезпечення юридичної значущості електронних документів.

Інформаційна система повинна надавати можливість шифрування електронних документів для запобігання несанкціонованому доступу до конфіденційної інформації. Шифрування має здійснюватися за допомогою надійних криптографічних алгоритмів, таких як RSA або AES. Зберігання зашифрованих документів повинно здійснюватися в захищеному середовищі з використанням відповідних стандартів і протоколів безпеки. Важливо також забезпечити довготривале зберігання документів з можливістю їх відновлення в разі необхідності.

Система повинна забезпечувати ведення журналу подій для кожної операції, пов'язаної з підписанням, перевіркою підпису та шифруванням документів. Такий аудит дозволить відстежувати, хто, коли та який документ підписував або перевіряв. Моніторинг подій у реальному часі дозволить своєчасно виявляти та реагувати на будь-які спроби несанкціонованого доступу або порушення безпеки.

Для забезпечення ефективної роботи з КЕП в юридичній компанії система має бути здатна інтегруватися з іншими програмами та інформаційними системами. Це може включати інтеграцію з внутрішніми системами документообігу, а також зовнішніми платформами для обміну електронними документами. Інтеграція дозволить автоматизувати процеси обробки документів, підписання та перевірки підписів без необхідності вручну взаємодіяти з кількома різними системами.

ІКС повинна бути здатна працювати в умовах великого навантаження, що особливо важливо для юридичних компаній, які часто мають справу з великими обсягами документів. Вона повинна бути масштабованою та дозволяти розширення функціональності без значних змін в основній архітектурі. Крім того, система повинна бути надійною та забезпечувати безперебійну роботу, а також мати механізми для аварійного відновлення даних.

Розробка інформаційно-комунікаційної системи для роботи з електронним цифровим підписом є важливим кроком для забезпечення безпеки та ефективності роботи юридичних компаній. Основні функціональні вимоги до системи включають надійну ідентифікацію користувачів, безпечне підписання та перевірку підписів, шифрування документів і можливість інтеграції з іншими системами. Також необхідно забезпечити масштабованість і надійність для роботи з великими обсягами даних. Врахування цих вимог дозволить створити ефективну та безпечну систему для обробки електронних документів у юридичній сфері.

2.2 Проектування архітектури системи з підтримкою КЕП

У сучасному світі, де інформаційно-комунікаційні технології стрімко розвиваються, захист даних стає однією з ключових вимог для забезпечення безпеки

цифрових операцій. Кваліфікований електронний підпис (КЕП) є важливим інструментом, що гарантує автентичність, цілісність і неспростовність електронних документів [19]. Проектування інформаційно-комунікаційної системи (ІКС) для роботи з КЕП передбачає створення надійної, масштабованої та зручної інфраструктури, яка забезпечує генерацію, верифікацію та використання підписів у цифровому середовищі. Така система повинна враховувати сучасні стандарти безпеки, оптимізувати продуктивність для користувачів і забезпечувати інтеграцію з іншими інформаційними системами [20].

Архітектура ІКС для підтримки КЕП включає кілька взаємопов'язаних компонентів, кожен з яких відіграє важливу роль у забезпеченні безпеки та функціональності системи. Основними завданнями при проектуванні є вибір криптографічних алгоритмів, організація процесів шифрування та дешифрування, а також створення інтуїтивно зрозумілого інтерфейсу для кінцевих користувачів [21]. Крім того, система має бути гнучкою для адаптації до нових загроз і технологічних змін, що є критично важливим у контексті швидкого розвитку кіберзлочинності [22]. Детальний опис ключових елементів архітектури системи показано в таблиці 2.1.

Таблиця 2.1

Елементи архітектури ІКС для роботи з КЕП

№	**Елементи**	**Опис**
1	2	3
1	Клієнтська частина	Інтерфейс користувача, який забезпечує зручну взаємодію з системою. Клієнтська частина дозволяє завантажувати файли для підписання, переглядати результати перевірки підписів, а також взаємодіяти із сервером для обробки запитів. Може бути реалізована як веб-інтерфейс (наприклад, за допомогою Streamlit, як у коді проєкту) або як окремий мобільний чи десктопний додаток, залежно від потреб користувачів і компанії.

Продовження таблиці 2.1

2	Серверна частина	Центральний компонент системи, що відповідає за основні операції: генерацію ключів, підписання документів, перевірку підписів, шифрування та дешифрування даних. Сервер використовує криптографічні бібліотеки (наприклад, cryptography у Python) для реалізації алгоритмів КЕП, таких як RSA та ECDSA. Забезпечує обробку файлів користувачів і взаємодію з базою даних.
3	База даних	Сховище для зберігання інформації про користувачів, їхні криптографічні ключі, сертифікати, підписані документи та журнали взаємодії з системою. У коді проєкту використовується SQLite (kep_database.db), але в промислових системах можуть застосовуватися більш масштабовані рішення, такі як PostgreSQL або MongoDB, для забезпечення високої доступності та безпеки даних [23].
4	Інтеграційний шар	Забезпечує зв’язок системи з зовнішніми платформами, такими як системи управління документообігом, державні реєстри чи інші інформаційні системи. Інтеграційний шар дозволяє автоматизувати обробку документів, наприклад, автоматичне підписання договорів у системах ERP або CRM, що підвищує ефективність бізнес-процесів [24].

Для забезпечення безпеки та ефективності роботи з КЕП необхідно ретельно підійти до вибору криптографічних алгоритмів. У коді проєкту використовується алгоритм RSA з довжиною ключа 2048 біт, що забезпечує високий рівень безпеки завдяки складності факторизації великих чисел [25]. Крім того, у системі застосовується підпис із використанням PSS (Probabilistic Signature Scheme) та хеш-функції SHA-256, що відповідає сучасним стандартам криптографії [26]. Альтернативою RSA може бути ECDSA (Elliptic Curve Digital Signature Algorithm), який є більш ефективним з точки зору обчислювальних ресурсів і часто

використовується в системах з обмеженими можливостями, наприклад, на мобільних пристроях [27]. У перспективі система може бути розширена для підтримки гібридних алгоритмів, які поєднують переваги RSA та ECDSA, забезпечуючи баланс між безпекою та продуктивністю.

Процес створення КЕП у системі складається з кількох етапів. Спочатку користувач ініціює генерацію пари ключів (приватного та публічного), що реалізовано у функції /api/generate_key у серверній частині коду. Приватний ключ використовується для створення підпису, тоді як публічний ключ і сертифікат зберігаються для подальшої верифікації. Сертифікат, створений за допомогою бібліотеки cryptography, є самопідписаним у демонстраційній версії, але в реальних системах він має видаватися довіреним центром сертифікації (ЦС) для забезпечення юридичної сили підпису [28]. Після генерації ключа документ підписується шляхом створення цифрового підпису, який включає хеш документа та шифрування цього хеша приватним ключем. Цей процес забезпечує цілісність і автентичність документа, оскільки будь-яка зміна вмісту призведе до невідповідності підпису.

Шифрування відіграє важливу роль у захисті даних під час їх передачі та зберігання. У системі застосовується асиметричне шифрування на основі RSA, яке гарантує конфіденційність даних. Наприклад, у коді проєкту документи зберігаються у базі даних у вигляді бінарних даних (encrypted_data), а їхній підпис кодується у форматі base64 для зручності передачі. Для підвищення безпеки можуть використовуватися додаткові алгоритми симетричного шифрування, такі як AES, для захисту великих обсягів даних, тоді як RSA застосовується лише для шифрування ключів [29]. Такий гібридний підхід дозволяє оптимізувати продуктивність системи.

Для ілюстрації роботи системи розроблено набір діаграм, які детально описують її функціонування на різних рівнях. Ці діаграми представлені в таблиці 2.2 і відображають як структурні, так і поведінкові аспекти системи.

Таблиця 2.2

Діаграми для демонстрації роботи системи

№	Діаграми	Призначення
1	2	3
1	Діаграма класів (рис. 2.2)	Відображає основні класи системи, такі як User, Key, Document, та їхні взаємозв'язки. Включає методи для роботи з криптографічними операціями (генерація ключів, підписання, верифікація) та обробки файлів. Наприклад, клас Key відповідає за генерацію та збереження ключів КЕП.
2	Діаграма компонентів (рис. 2.3)	Описує функціональні блоки системи: модуль генерації ключів, модуль підписання, модуль верифікації, модуль шифрування/дешифрування та модуль управління файлами. Показує, як ці компоненти взаємодіють через API (наприклад, /api/sign_document).
3	Діаграма використання (рис. 2.4)	Ілюструє взаємодію користувачів із системою через інтерфейс. Користувач може створювати КЕП, підписувати документи, перевіряти підписи та завантажувати збережені документи. Відображає ролі користувача та адміністратора.
4	Діаграма кооперацій (рис. 2.5)	Показує співпрацю між об'єктами системи під час виконання запитів, наприклад, як клієнтський запит на підписання документа обробляється сервером і базою даних. Включає обмін даними між клієнтом, сервером і криптографічним модулем.

Продовження таблиці 2.2

1	2	3
5	Діаграма діяльності (рис. 2.6)	Детально описує послідовність операцій для ключових процесів: генерація ключів, підписання документа, перевірка підпису, шифрування та дешифрування. Наприклад, показує, як документ хешується перед підписанням і як перевіряється цілісність підпису.
6	Діаграма станів (рис. 2.7)	Відображає стани ключових компонентів системи, таких як ключ КЕП (створений, активний, закінчився термін дії) або документ (завантажений, підписаний, перевірений). Показує переходи між станами залежно від дій користувача чи системи.
7	Діаграма розгортання (рис. 2.8)	Ілюструє фізичне розміщення компонентів системи: клієнтський інтерфейс (браузер або додаток), сервер (Flask у коді), база даних (SQLite) та можливі зовнішні сервіси (наприклад, ЦС). Показує мережеву взаємодію між цими компонентами.
8	Діаграма послідовностей (рис. 2.9)	Описує послідовність операцій від моменту завантаження файлу користувачем до отримання результату підписання чи перевірки. Наприклад, показує, як клієнтський запит передається серверу, як сервер генерує підпис і повертає результат користувачу.

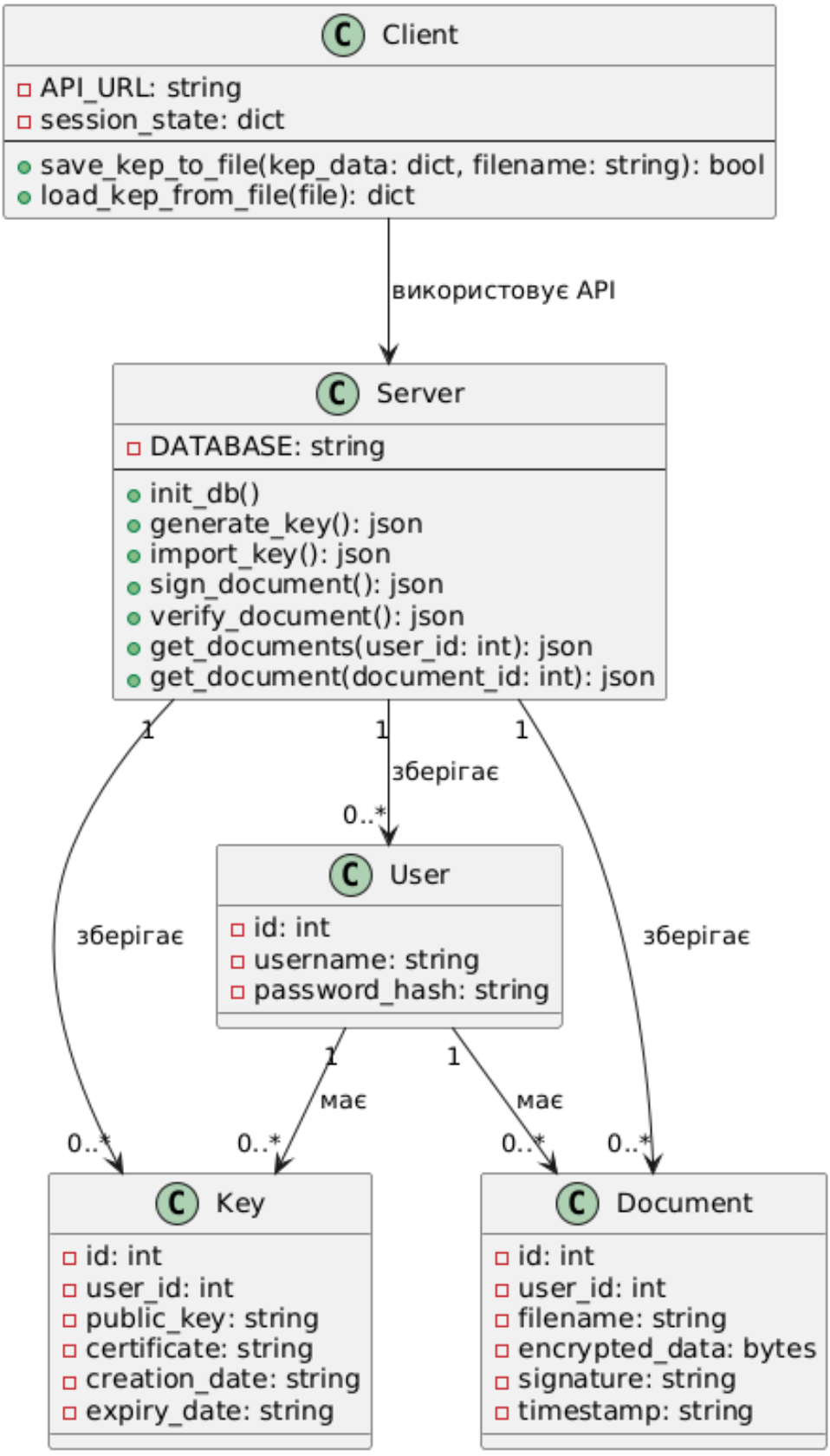

Рис. 2.2. Діаграма класів

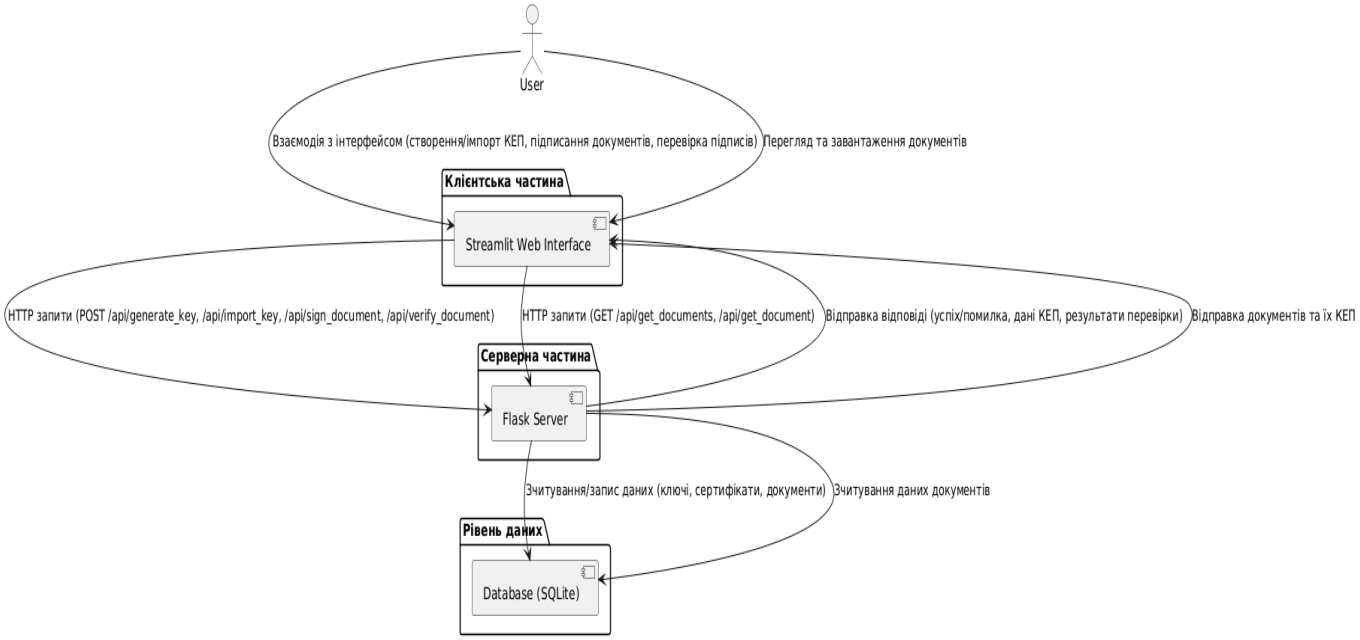

Рис. 2.3. Діаграма компонентів

Рис. 2.4. Діаграма варіантів використання

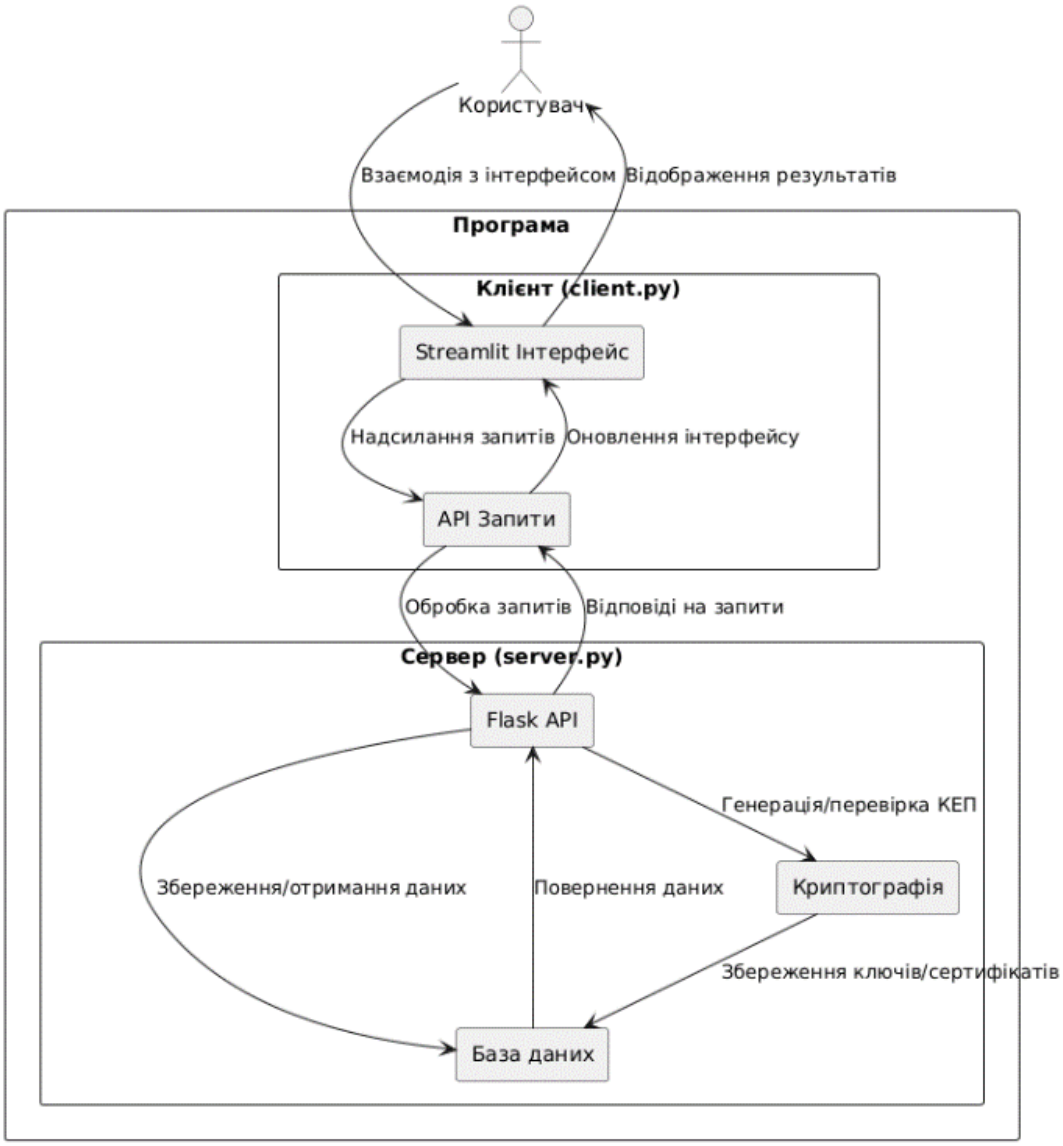

Рис. 2.5. Діаграма кооперацій

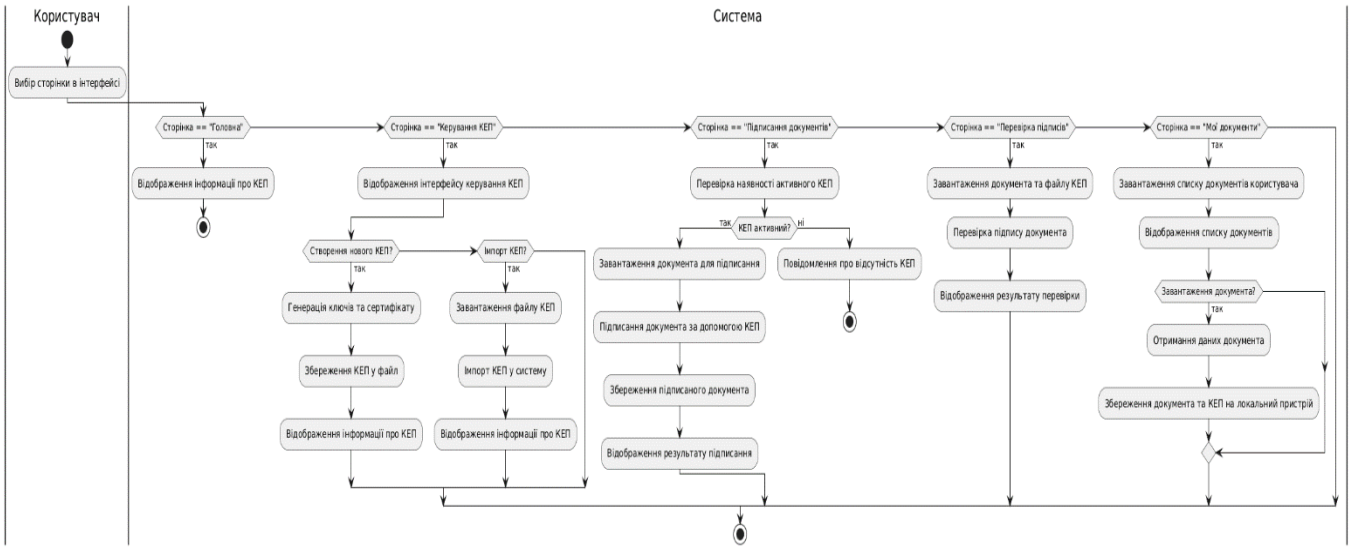

Рис. 2.6. Діаграма діяльності

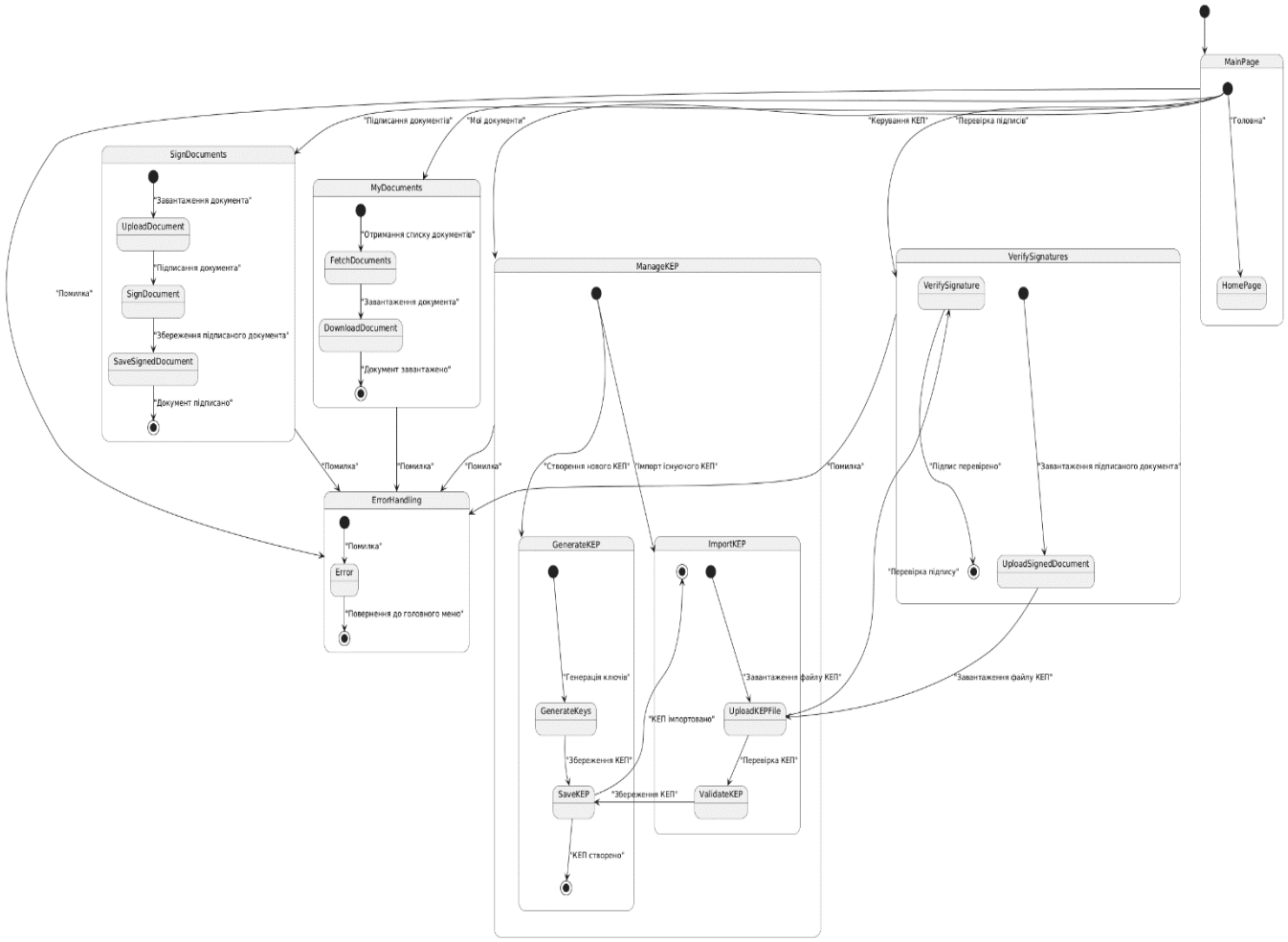

Рис. 2.7. Діаграма станів

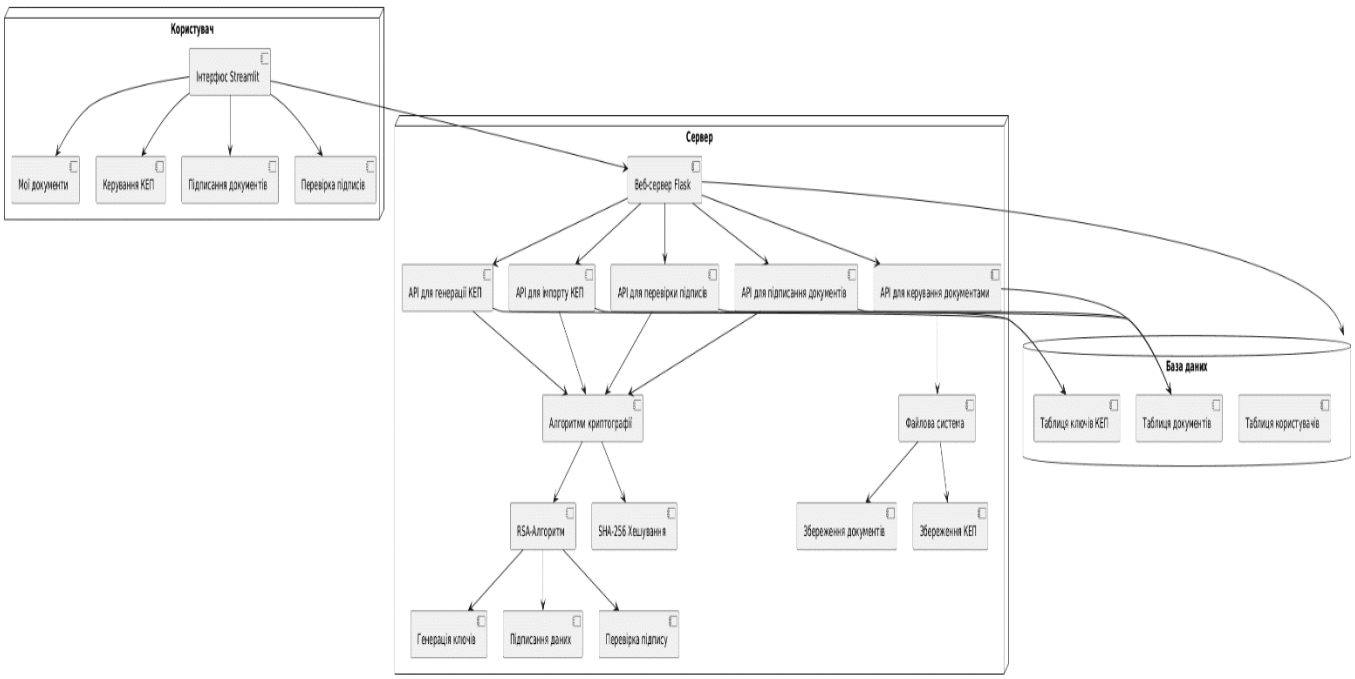

Рис. 2.8. Діаграма розгортання

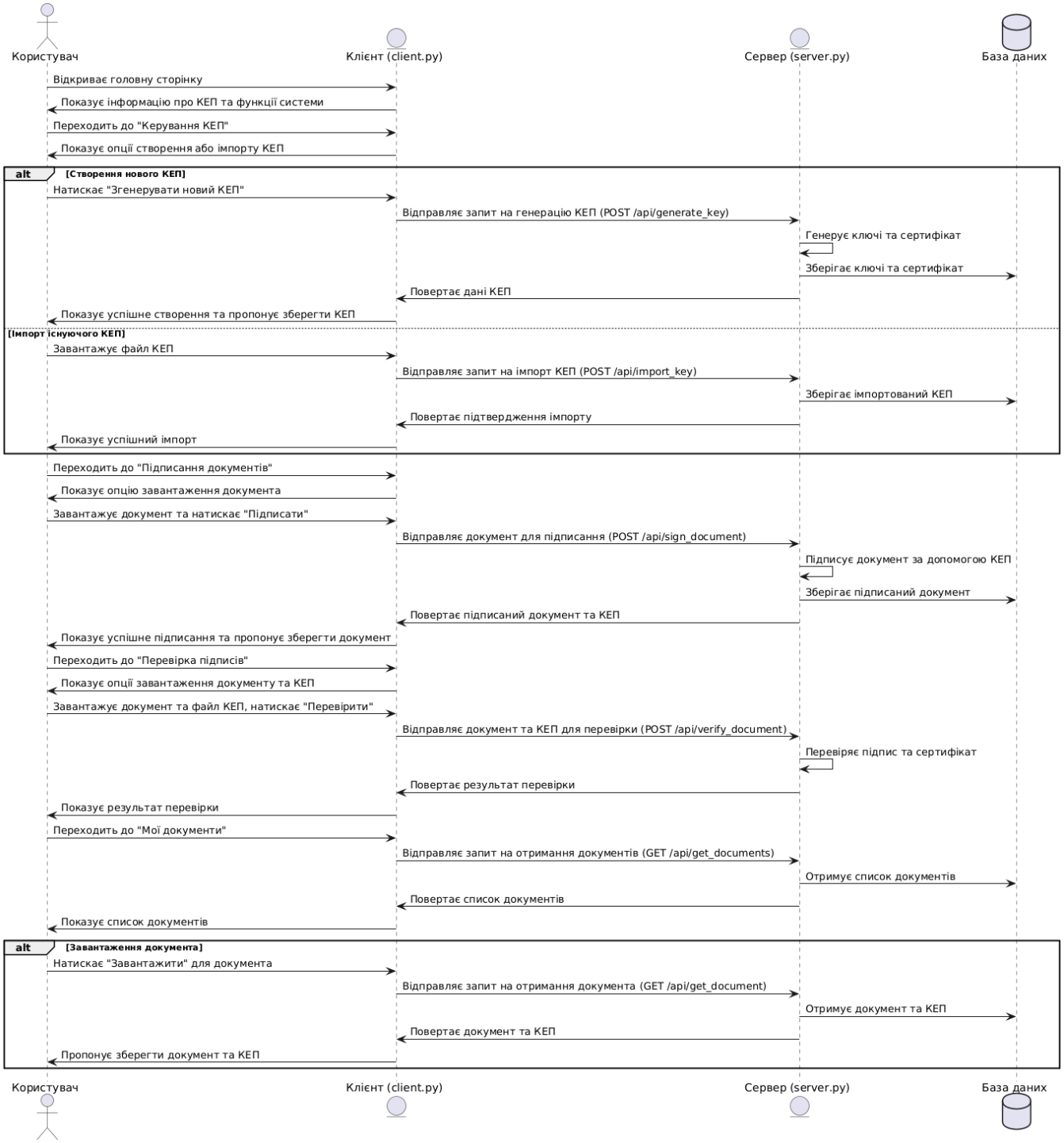

Рис. 2.9. Діаграма послідовності

Безпека є наріжним каменем при проектуванні системи для роботи з КЕП. Для цього необхідно не лише використовувати надійні криптографічні алгоритми, а й забезпечити захист ключів від несанкціонованого доступу. У коді проєкту приватні ключі зберігаються у вигляді файлів JSON, що є прийнятним для демонстраційних цілей, але в реальних системах рекомендується використовувати апаратні модулі

безпеки (HSM) або захищені сховища, такі як KeyVault [30]. Крім того, система повинна передбачати регулярне оновлення сертифікатів і можливість їх анулювання у разі компрометації.

Ефективність системи залежить від оптимізації криптографічних операцій і обробки великих обсягів даних. У коді використовується локальний сервер Flask, що підходить для прототипу, але для масштабування системи необхідно впроваджувати розподілені архітектури, наприклад, мікросервіси, які дозволяють розподілити навантаження між різними вузлами [31]. Також важливо забезпечити зручність для користувачів, що досягається завдяки інтуїтивному інтерфейсу, як у клієнтській частині на основі Streamlit, де користувач може легко створювати КЕП, підписувати документи чи перевіряти їх.

Масштабованість системи є ще одним важливим аспектом. У майбутньому система може бути розширена для підтримки нових криптографічних алгоритмів, таких як постквантові алгоритми, які стійкі до атак квантових комп’ютерів [32]. Крім того, інтеграція з державними системами, такими як українська платформа "Дія", може підвищити її практичну цінність, дозволяючи користувачам підписувати офіційні документи в електронному вигляді [24].

Підсумовуючи, проектування архітектури ІКС для роботи з КЕП вимагає комплексного підходу, що поєднує безпеку, ефективність і зручність використання. Надійні криптографічні алгоритми, продумана структура компонентів і гнучкість для майбутніх розширень є ключовими факторами успіху такої системи. Розроблені діаграми дозволяють детально зрозуміти функціонування системи, а реалізація на основі Python (Flask і Streamlit) демонструє практичну можливість створення подібних рішень.

2.3 Інтеграція алгоритмів КЕП у систему

Кваліфікований електронний підпис (КЕП) є ключовим інструментом забезпечення безпеки електронних документів, що гарантує їх автентичність, цілісність і неспростовність у цифровому середовищі [33]. Цей механізм набув

широкого поширення в державному управлінні, комерційних операціях і особистих транзакціях, де захист даних і довіра до інформації є критично важливими. Інтеграція алгоритмів КЕП у програмні системи є складним, але необхідним процесом, який охоплює вибір криптографічних алгоритмів, розробку механізмів генерації ключів, підписання, перевірки підписів, а також створення зручного інтерфейсу для користувачів. У даному розділі детально розглядаються основні аспекти інтеграції КЕП у систему, включаючи вибір алгоритмів, реалізацію криптографічних операцій, взаємодію з іншими компонентами системи, управління ключами та забезпечення безпеки. Розглянута система, побудована на основі коду проєкту, демонструє сучасний підхід до реалізації КЕП, поєднуючи надійність і зручність використання.

2.3.1 Вибір криптографічних алгоритмів

Вибір криптографічних алгоритмів є основою для створення надійної системи КЕП. Сучасні стандарти безпеки рекомендують використовувати асиметричні алгоритми, такі як RSA (Rivest-Shamir-Adleman) або ECDSA (Elliptic Curve Digital Signature Algorithm), які забезпечують високий рівень захисту завдяки розподілу ключів на приватний і публічний [34]. RSA, використаний у даній системі, є одним із найпоширеніших алгоритмів завдяки своїй перевіреній надійності та підтримці в більшості криптографічних бібліотек. У системі застосовується RSA з довжиною ключа 2048 біт, що відповідає сучасним стандартам безпеки [35]. Цей алгоритм дозволяє створювати цифровий підпис шляхом шифрування хеш-значення документа приватним ключем, тоді як перевірка підпису виконується за допомогою публічного ключа, забезпечуючи цілісність і автентичність даних.

Поряд із RSA, у системі використовується алгоритм хешування SHA-256, який є частиною сімейства SHA-2 і забезпечує стійкість до колізій, що критично важливо для перевірки цілісності документів [36]. Вибір SHA-256 обґрунтований його високою криптографічною стійкістю та широкою підтримкою в сучасних системах. Поєднання RSA та SHA-256 дозволяє системі ефективно обробляти документи будь-

якого розміру, оскільки хешування зменшує обсяг даних, що підписуються, зберігаючи при цьому високий рівень безпеки.

2.3.2 Генерація ключів та сертифікатів

Генерація ключів і сертифікатів є центральним елементом інтеграції КЕП, оскільки від якості цього процесу залежить надійність усієї системи. У розглянутій системі генерація ключів здійснюється за допомогою криптографічної бібліотеки cryptography у Python, яка забезпечує створення пари ключів RSA з високим рівнем безпеки [37]. Процес включає:

– Генерацію приватного ключа здійснюють із використанням параметрів, що відповідають сучасним стандартам, таких як експонента 65537 та розмір ключа 2048 біт. Цей ключ залишається конфіденційним та використовується виключно для підписання документів.

– Створення публічного ключа відбувається на основі приватного, після чого він включається до сертифіката, що дає змогу здійснювати перевірку підписів у процесах автентифікації.

– Формування самопідписаного сертифіката передбачає створення сертифіката X.509, який містить дані про власника ключа, термін його дії протягом 365 днів та розширення, такі як BasicConstraints, що запобігають використанню ключа для генерації інших сертифікатів.

Сертифікат відіграє роль цифрового посвідчення, що підтверджує зв’язок між публічним ключем і користувачем. У системі сертифікати зберігаються в базі даних SQLite разом із ключами, що забезпечує швидкий доступ до них під час операцій підписання та перевірки. Крім того, система дозволяє експортувати ключі та сертифікати у форматі JSON, що спрощує їх перенесення між різними пристроями.

2.3.3 Підписання документів

Процес підписання документів є ключовою функцією системи КЕП, яка дозволяє користувачам створювати юридично значущі документи. У системі цей процес реалізовано через зручний вебінтерфейс, побудований на основі бібліотеки Streamlit, що забезпечує простоту взаємодії з користувачем. Етапи підписання документів детально описані в таблиці 2.3.

Таблиця 2.3

Процес підписання документів

№	Етапи	Опис
1	Завантаження документа	Користувач через вебінтерфейс завантажує файл (будь-якого формату), який необхідно підписати. Система зберігає файл у тимчасовій пам’яті для обробки.
2	Хешування даних	Документ обробляється алгоритмом SHA-256 для створення унікального хеш-значення, яке відображає вміст файлу. Це забезпечує цілісність документа, оскільки будь-яка зміна вмісту призведе до іншого хеш-значення.
3	Створення підпису	Приватний ключ користувача використовується для шифрування хеш-значення з використанням алгоритму RSA та схеми заповнення PSS (Probabilistic Signature Scheme). Отриманий підпис унікально пов’язує документ із власником ключа.
4	Збереження підпису	Підпис кодується у форматі base64 і зберігається в базі даних разом із вихідним документом. Крім того, система створює файл КЕП у форматі JSON, який містить підпис, сертифікат і метадані, що дозволяє користувачу завантажити його для подальшого використання.

Цей процес повністю автоматизований: користувачу достатньо завантажити документ і вибрати КЕП, після чого система виконує всі необхідні криптографічні операції. Використання PSS у поєднанні з RSA підвищує безпеку підпису, оскільки ця схема забезпечує стійкість до атак, пов'язаних із передбаченням заповнення [38].

2.3.4 Перевірка підпису

Перевірка підпису є невід'ємною частиною системи КЕП, оскільки вона дозволяє підтвердити автентичність і цілісність документа. У системі реалізовано механізм перевірки, який охоплює кілька етапів, описаних у таблиці 2.4.

Таблиця 2.4

Процес перевірки підпису

№	Етапи	Опис
1	Завантаження документу та підпису	Користувач завантажує оригінальний документ і відповідний файл КЕП (у форматі JSON), який містить підпис, сертифікат і публічний ключ.
2	Декодування підпису	Підпис, закодований у форматі base64, декодується для використання в криптографічних операціях. Це забезпечує коректну обробку даних.
3	Перевірка цілісності	Система обчислює хеш-значення завантаженого документа за допомогою SHA-256 і порівнює його з розшифрованим підписом, використовуючи публічний ключ. Якщо хеш-значення збігаються, документ вважається незміненим.
4	Перевірка сертифіката	Система перевіряє валідність сертифіката, зокрема його термін дії (дати початку та закінчення) і відповідність публічного ключа. Якщо сертифікат недійсний або прострочений, перевірка завершується з помилкою.

Результат перевірки відображається у вебінтерфейсі, де користувач отримує повідомлення про дійсність підпису та статус сертифіката. Цей процес забезпечує довіру до документів, оскільки дозволяє виявити будь-які зміни або спроби підробки. Використання стандарту X.509 для сертифікатів гарантує сумісність із міжнародними стандартами безпеки [39].

2.3.5 Збереження та управління ключами

Ефективне управління ключами є критично важливим для функціонування системи КЕП. У розглянутій системі ключі та сертифікати зберігаються в базі даних SQLite, що забезпечує їх захист від несанкціонованого доступу та швидкий доступ під час операцій підписання та перевірки. Основні аспекти управління ключами включають:

– Зберігання ключів передбачає збереження приватних і публічних ключів разом із сертифікатами у зашифрованому вигляді в базі даних, що гарантує їхню безпеку. Обмежений доступ до приватних ключів запобігає їх витоку та забезпечує конфіденційність.

– Експорт та імпорт ключів здійснюється через можливість експорту КЕП у файл JSON, що дозволяє переносити дані на інші пристрої або створювати резервні копії. Процедура імпорту передбачає завантаження JSON-файлу та перевірку його цілісності для гарантії безпеки.

– Управління термінами дії забезпечує автоматичний моніторинг строків сертифікатів, що дозволяє вчасно попереджати користувачів про необхідність оновлення та запобігати використанню прострочених ключів.

– Доступ до ключів реалізовано через вебінтерфейс, де користувачі можуть переглядати метадані своїх ключів, такі як дата створення та термін їхньої дії.

Такий підхід забезпечує гнучкість і зручність управління ключами, що є важливим для користувачів із різним рівнем технічної підготовки.

2.3.6 Забезпечення безпеки

Безпека є наріжним каменем інтеграції алгоритмів КЕП. У системі реалізовано комплексний підхід до захисту даних, який включає такі заходи:

– Використання надійних бібліотек гарантує безпеку криптографічних операцій, а бібліотека cryptography забезпечує перевірений набір примітивів відповідно до сучасних стандартів. [37].

– Захист приватних ключів реалізується шляхом їхнього зберігання у зашифрованому вигляді та обмеження передачі між клієнтом і сервером лише у разі необхідності, що мінімізує ризик компрометації.

– Перевірка сертифікатів виконується автоматично, що дозволяє контролювати терміни їхньої дії та валідність, запобігаючи використанню недійсних ключів.

– Шифрування даних у реальних системах рекомендується здійснювати через TLS для захисту каналу зв’язку, оскільки передача інформації між клієнтом і сервером має гарантувати конфіденційність [40].

– Обмеження доступу передбачає використання ідентифікатора користувача для контролю доступу до ключів і документів, що забезпечує захист від несанкціонованого використання.

Ці заходи створюють надійне середовище для роботи з КЕП, забезпечуючи довіру користувачів до системи. Однак для підвищення безпеки в майбутніх версіях системи доцільно реалізувати двофакторну автентифікацію та шифрування бази даних.

Інтеграція алгоритмів кваліфікованого електронного підпису в інформаційні системи є складним, але критично важливим процесом, який забезпечує захист даних і довіру до цифрових документів. Розглянута система демонструє ефективну реалізацію КЕП, що охоплює вибір криптографічних алгоритмів (RSA та SHA-256), генерацію ключів і сертифікатів, підписання та перевірку документів, а також управління ключами. Завдяки використанню сучасних бібліотек, таких як cryptography, і продуманому вебінтерфейсу на основі Streamlit, система є зручною для

користувачів і відповідає сучасним стандартам безпеки. Заходи захисту, такі як шифрування ключів і перевірка сертифікатів, забезпечують надійність системи, хоча подальше вдосконалення (наприклад, додавання TLS і двофакторної автентифікації) може підвищити її безпеку. Інтеграція КЕП є важливим кроком у розвитку цифрових технологій, спрямованих на захист інформації та забезпечення юридичної значущості документів у цифровому середовищі.

2.4 Розробка інтерфейсу користувача для роботи з КЕП

У сучасному цифровому світі захист інформації є критично важливим завданням, адже обсяги даних, що обробляються в електронному вигляді, стрімко зростають. Кваліфікований електронний підпис (КЕП) відіграє ключову роль у забезпеченні безпеки документів, гарантуючи їх автентичність, цілісність та неспростовність. Цей інструмент є невід’ємною частиною електронного документообігу, фінансових транзакцій, державних сервісів та інших сфер, де потрібна юридична значущість. Однак для ефективного використання КЕП необхідний зручний, інтуїтивно зрозумілий та безпечний інтерфейс користувача, який забезпечує комфортну взаємодію з системою. Розробка такого інтерфейсу є складним завданням, що вимагає врахування принципів юзабіліті, безпеки та функціональності. У даному розділі розглядається архітектура інтерфейсу, його основні модулі та їхня реалізація на основі коду проєкту, а також підходи до забезпечення зручності й ефективності роботи користувачів із КЕП.

2.4.1 Архітектура інтерфейсу користувача

Інтерфейс користувача для роботи з КЕП розроблено як модульну систему, що поєднує зручність навігації, чітку структуру та високий рівень безпеки. Кожен модуль інтерфейсу виконує специфічну функцію, що дозволяє користувачам швидко орієнтуватися в системі та виконувати необхідні операції. Модульна структура

сприяє легкому масштабуванню системи та забезпечує гнучкість у додаванні нових функцій. Основні модулі інтерфейсу наведено в таблиці 2.5.

Таблиця 2.5

Модулі інтерфейсу користувача

№	**Модулі**	**Опис**
1	2	3
1	Головна сторінка (рис. 2.10)	Надає загальну інформацію про систему, її можливості, переваги КЕП та інструкції для початку роботи
2	Керування КЕП	Дозволяє створювати нові кваліфіковані електронні підписи (рис. 2.11), імпортувати збережені підписи (рис. 2.12) та переглядати деталі поточного КЕП (рис. 2.13)
3	Підписання документів	Забезпечує завантаження файлів, їх підписання за допомогою КЕП та збереження підписаних документів (рис. 2.14).
4	Перевірка підписів	Надає інструменти для перевірки дійсності підписів, цілісності документів і терміну дії сертифікатів (рис. 2.15).
5	Мої документи	Дозволяє переглядати список підписаних документів, завантажувати їх та відповідні файли підписів (рис. 2.16).

Рисунок 2.10 – Головна сторінка

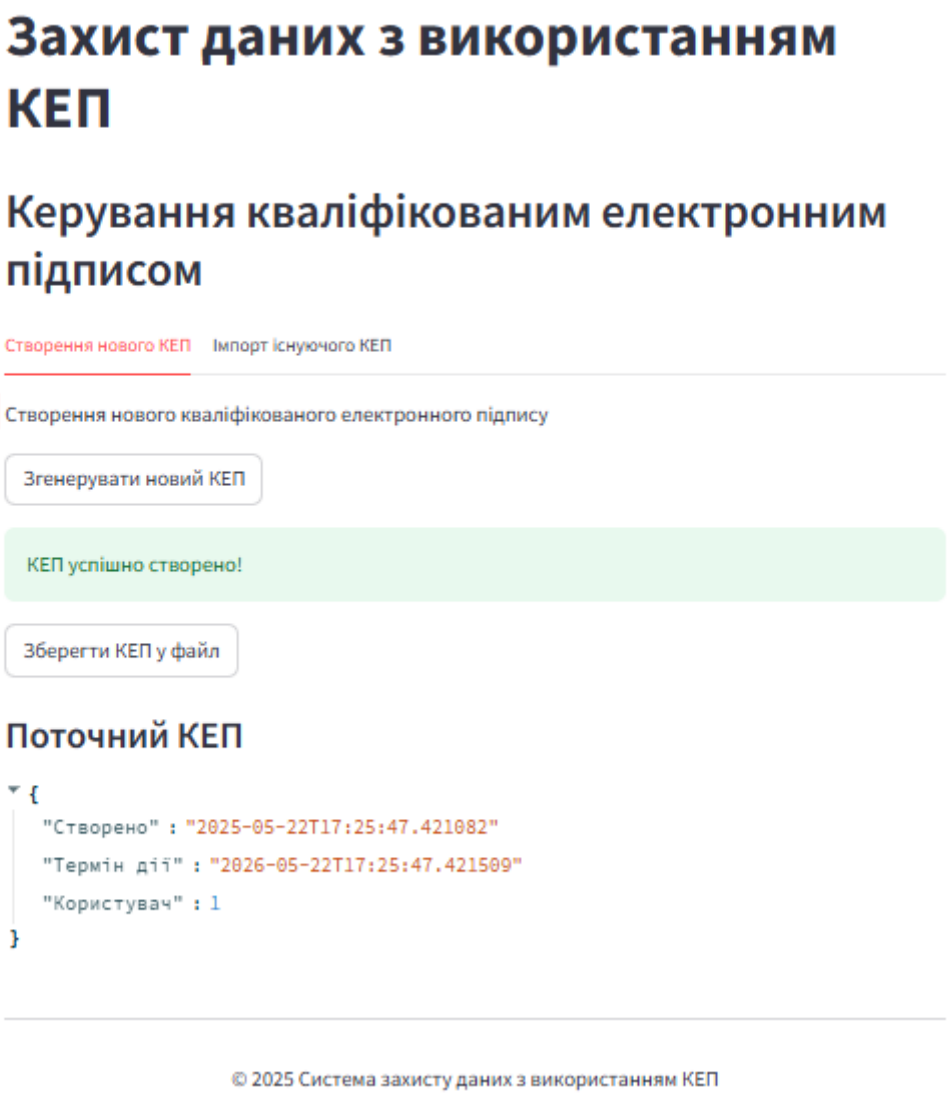

Рисунок 2.11 – Створення нового ключа

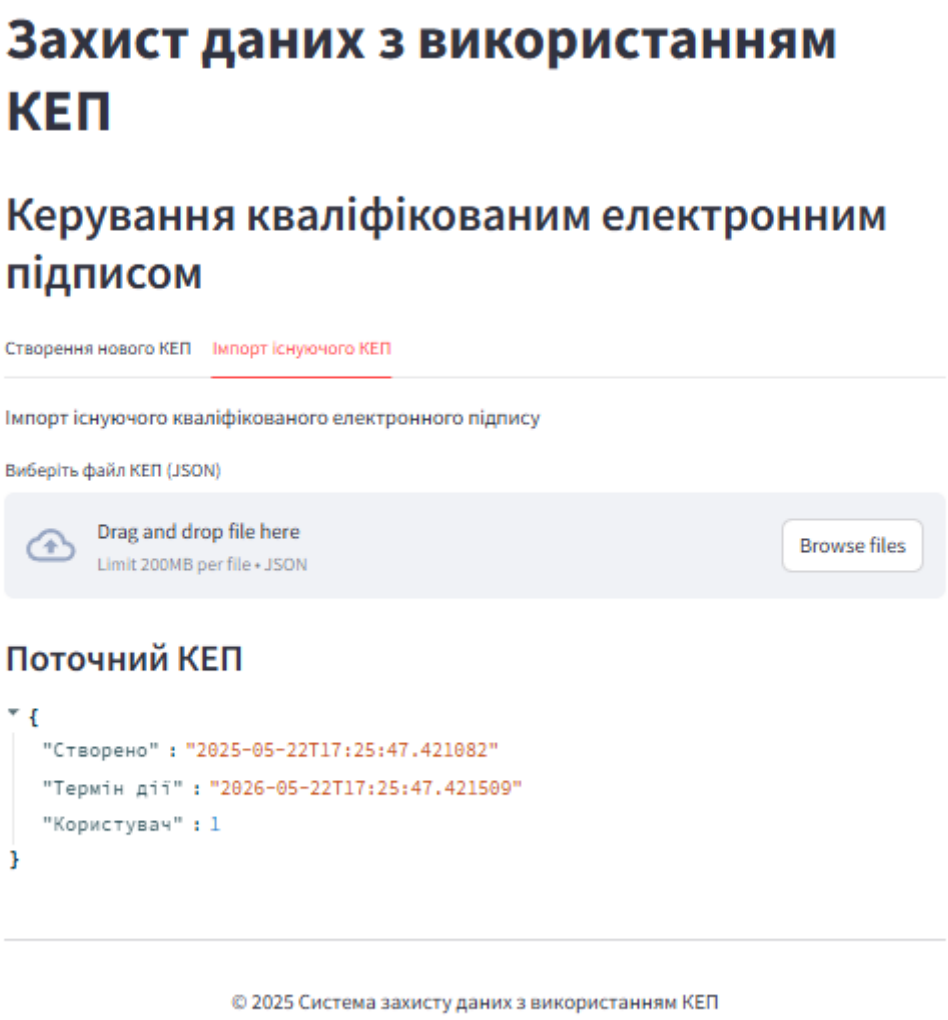

Рисунок 2.12 – Імпорт збереженого підпису

Рисунок 2.13 – Деталі поточного КЕП

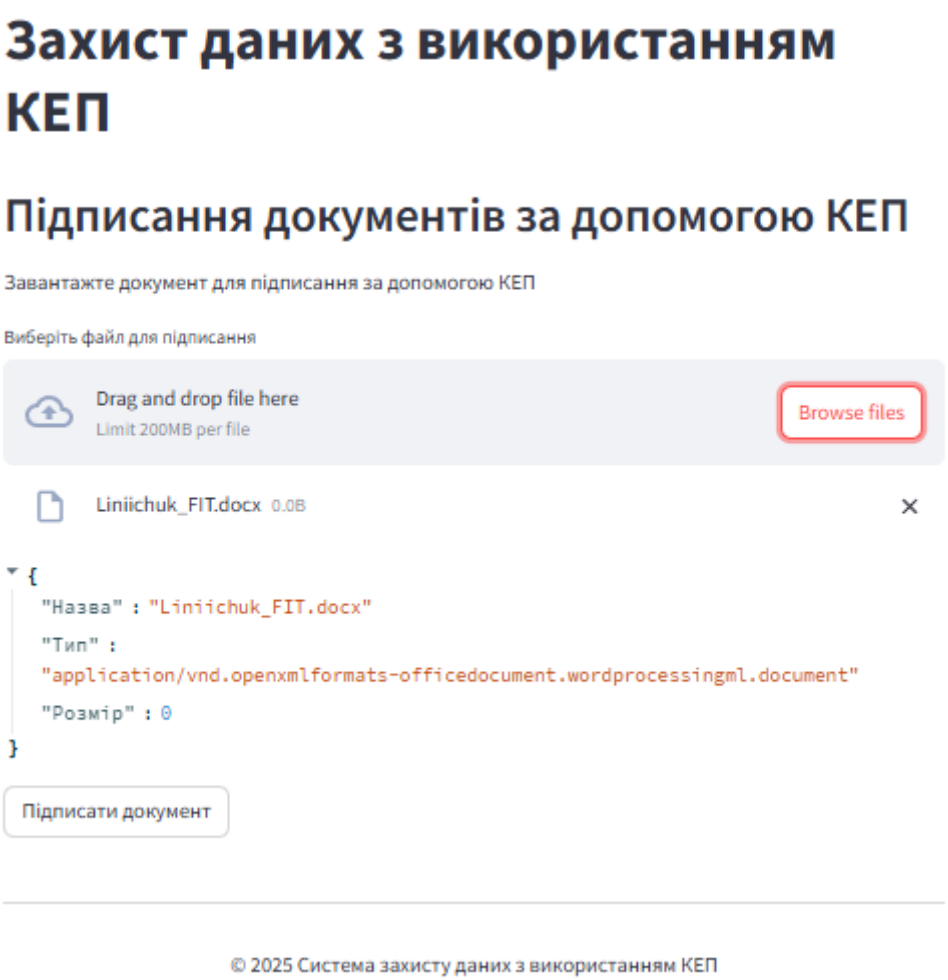

Рисунок 2.14 – Підписання документів за допомогою КЕП

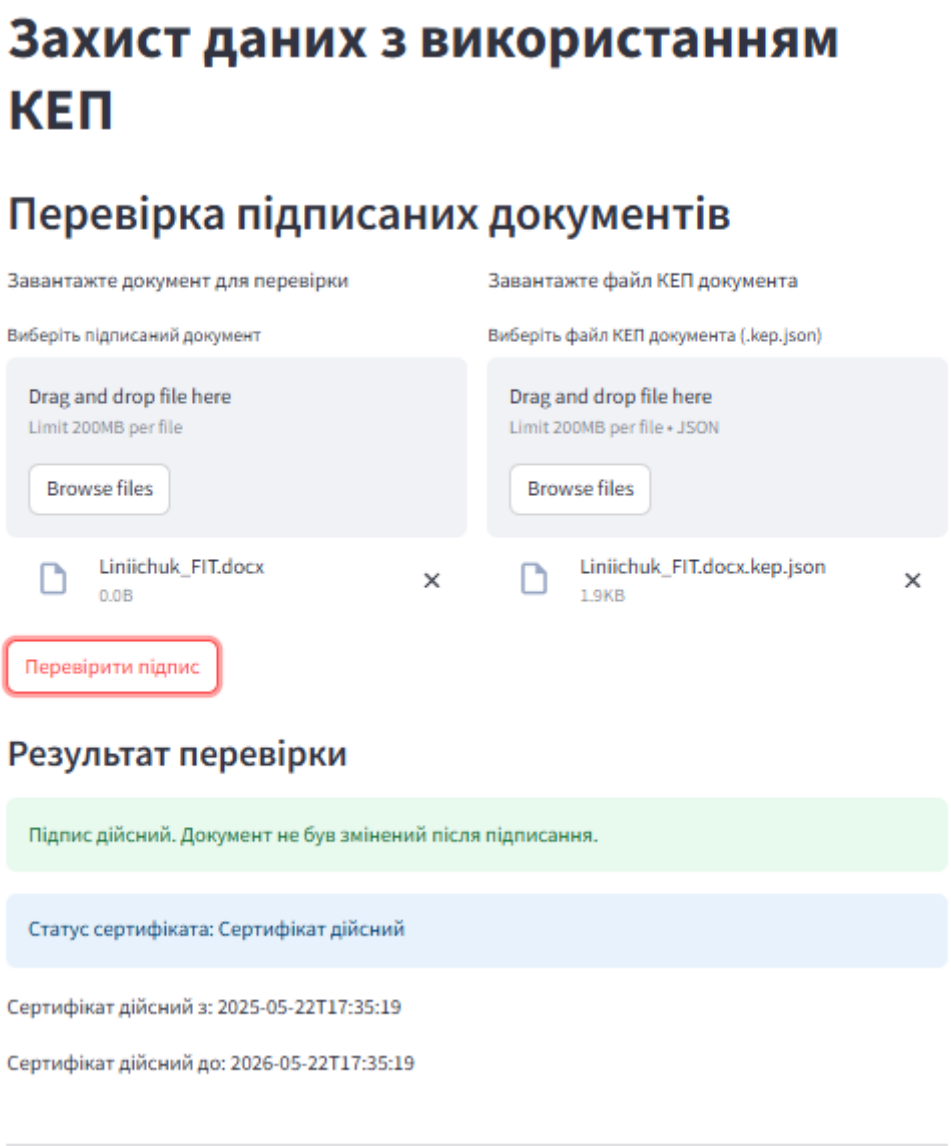

Рисунок 2.15 – Перевірка підписаних документів

Рисунок 2.16 – Перегляд власних підписаних документів

Така організація інтерфейсу забезпечує чітке розділення функціоналу, що полегшує роботу користувачів із різним рівнем технічної підготовки. Наприклад, новачки можуть легко розпочати роботу завдяки інформативній головній сторінці, тоді як досвідчені користувачі швидко знаходять потрібні функції через бічну панель навігації. Реалізація інтерфейсу за допомогою бібліотеки Streamlit, як показано в коді, дозволяє створити адаптивний і сучасний дизайн із підтримкою інтерактивних елементів, таких як кнопки, вкладки та сповіщення.

2.4.2 Реалізація головної сторінки

Головна сторінка є першим пунктом взаємодії користувача з системою, тому її дизайн і зміст мають бути максимально зрозумілими та інформативними. Вона виконує роль інформаційного порталу, який знайомить користувача з можливостями системи та пояснює переваги використання КЕП. У коді проєкту головна сторінка реалізована з використанням Streamlit і містить заголовок, опис функціоналу та

інформаційне повідомлення, яке спрямовує користувача до розділу керування КЕП. Основні можливості КЕП, представлені на головній сторінці, наведено в таблиці 2.6.

Таблиця 2.6

Можливості КЕП

№	Можливості	Опис
1	Автентичність	Гарантує, що документ створений саме тим користувачем, який його підписав, шляхом використання унікального приватного ключа.
2	Цілісність	Забезпечує, що вміст документа не зазнав змін після підписання, завдяки криптографічним механізмам перевірки.
3	Неспростовність	Унеможливлює заперечення факту підписання документа, оскільки підпис пов’язаний із сертифікатом користувача.

Інтерфейс головної сторінки виконаний у лаконічному стилі, що дозволяє уникнути перевантаження інформацією (рис. 2.17). Текстовий опис можливостей КЕП супроводжується чіткими поясненнями, які допомагають користувачам зрозуміти юридичну та технічну значущість підпису. Інформаційне повідомлення, реалізоване за допомогою компонента st.info, нагадує користувачам про необхідність створення або імпорту КЕП, що є важливим для нових користувачів. Крім того, використання української мови в інтерфейсі робить систему доступною для україномовної аудиторії, що відповідає локальним вимогам і сприяє популяризації КЕП у державних і комерційних структурах.

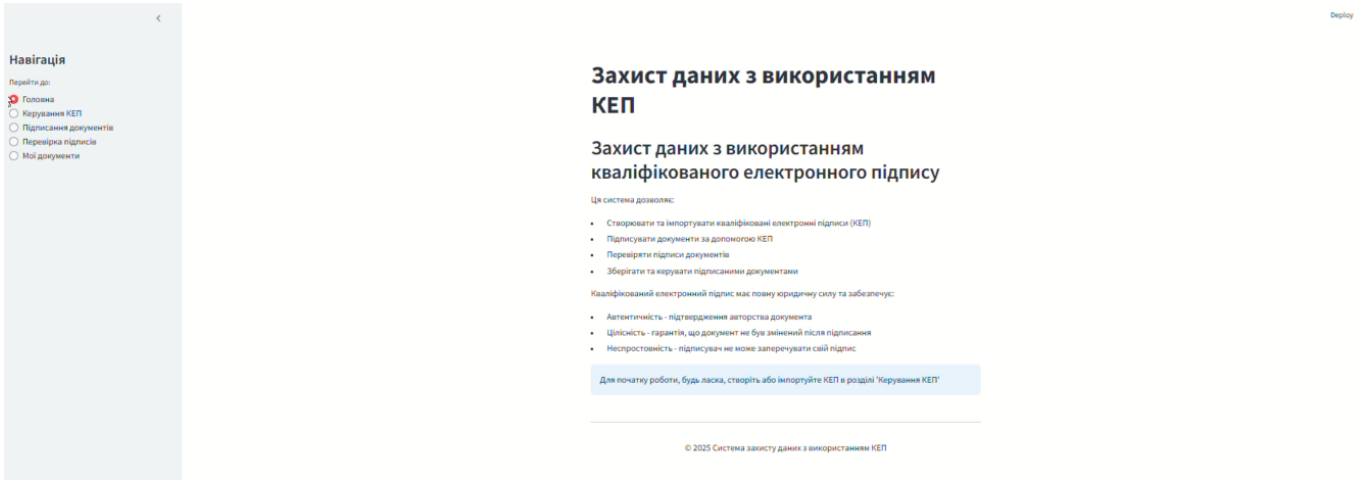

Рисунок 2.17 – Інтерфейс головної сторінки

2.4.3 Функціонал керування КЕП

Модуль керування КЕП є центральним елементом системи, оскільки саме він дозволяє користувачам створювати, імпортувати та переглядати електронні підписи. У коді цей модуль реалізований як набір вкладок, що забезпечують логічне розділення функцій. Користувач може згенерувати новий КЕП, завантажити раніше збережений підпис або переглянути деталі поточного КЕП. Структура вкладок модуля наведена в таблиці 2.7.

Таблиця 2.7

Вкладки модулю керування КЕП

№	Вкладки	Опис
1	Створювати новий КЕП	користувач може згенерувати новий підпис, який включає приватний ключ, публічний ключ та сертифікат. Після створення система надає можливість зберегти підпис у файл для подальшого використання
2	Імпортувати існуючий КЕП	користувач може завантажити раніше створений підпис у форматі JSON. Це дозволяє продовжити роботу зі збереженими даними

Продовження таблиці 2.7

3	Переглядати інформацію про поточний ЕЦП	система відображає дані про термін дії підпису, дату створення та ідентифікатор користувача

Функціонал створення КЕП у коді реалізовано через відправлення POST-запиту до серверного API, яке генерує пару ключів RSA та сертифікат із терміном дії один рік. Після генерації користувач отримує можливість завантажити КЕП у вигляді JSON-файлу, що забезпечує його збереження та подальше використання. Імпорт КЕП дозволяє завантажити JSON-файл і перевірити його коректність перед збереженням у базі даних. Для підвищення безпеки система обробляє можливі помилки, такі як некоректний формат файлу, і відображає відповідні повідомлення через st.error.

Перегляд інформації про поточний КЕП відображається у форматі JSON із ключовими параметрами, такими як дата створення, термін дії та ідентифікатор користувача. Це дозволяє користувачам швидко перевірити статус підпису та переконатися в його актуальності. Завдяки інтеграції з криптографічними бібліотеками, такими як cryptography, модуль забезпечує надійне зберігання та обробку ключів, що є критично важливим для безпеки.

2.4.4 Підписання документів

Модуль підписання документів є одним із найважливіших, оскільки він дозволяє користувачам застосовувати КЕП до файлів, забезпечуючи їх юридичну значущість. У коді цей модуль реалізований як інтерактивна форма, де користувач завантажує файл, переглядає його деталі (назва, тип, розмір) і підписує документ за допомогою поточного КЕП. Процес підписання включає кілька етапів:

– Завантаження файлу передбачає вибір користувачем файлу будь-якого формату через компонент st.file_uploader, після чого система відображає його деталі для підтвердження перед обробкою.

– Підписання здійснюється із застосуванням приватного ключа з поточного КЕП, що використовується для створення цифрового підпису. Алгоритм RSA із хеш-функцією SHA-256 та механізмом PSS-паддингу забезпечує безпеку та відповідність сучасним стандартам.

– Збереження підписаного документа та його КЕП виконується у базі даних, що гарантує можливість подальшого доступу. Користувач отримує можливість завантажити файл підпису у форматі JSON для використання або резервного копіювання.

У разі відсутності КЕП система попереджає користувача через st.warning і пропонує перейти до модуля керування КЕП. Після успішного підписання відображається повідомлення про успіх (st.success), а також надаються кнопки для завантаження підпису та документа. Такий підхід забезпечує зручність і прозорість процесу, а використання криптографічних механізмів гарантує безпеку.

2.4.5 Перевірка підписів

Модуль перевірки підписів дозволяє користувачам переконатися в цілісності та дійсності підписаних документів. У коді він реалізований як форма з двома полями для завантаження: підписаного документа та відповідного файлу КЕП (JSON). Після завантаження система виконує перевірку, яка включає:

– Перевірку підпису здійснюють із застосуванням публічного ключа з файлу КЕП, що використовується для верифікації підпису за алгоритмом RSA та SHA-256, забезпечуючи його достовірність.

– Перевірку сертифіката виконує система, аналізуючи термін його дії та порівнюючи поточну дату з датами початку та завершення, що запобігає використанню недійсного сертифіката.

– Відображення результатів надає користувачеві чітке повідомлення про дійсність підпису через індикатори st.success або st.error, а також містить інформацію про статус сертифіката та його період дії.

Цей модуль є важливим для забезпечення довіри до документів, оскільки дозволяє виявити будь-які зміни чи невідповідності. Наприклад, якщо документ був змінений після підписання, система повідомить про недійсність підпису. Інтеграція з бібліотекою cryptography забезпечує надійність перевірки, а зручний інтерфейс із двома колонками для завантаження файлів робить процес інтуїтивно зрозумілим.

2.4.6 Керування підписаними документами

Модуль "Мої документи" надає користувачам доступ до всіх підписаних документів, що зберігаються в базі даних. У коді він реалізований як список із інформацією про кожен документ: назва, дата підписання та кнопки для завантаження документа та його КЕП. Для кожного документа користувач може:

- Переглянути деталі, такі як ім’я файлу та час підписання.
- Завантажити оригінальний документ у бінарному форматі.
- Завантажити файл КЕП у форматі JSON для подальшої перевірки.

Система використовує GET-запити до серверного API для отримання списку документів і їхніх даних. Якщо документів немає, відображається повідомлення через st.info. У разі помилок, наприклад, при завантаженні списку, користувач отримує повідомлення про помилку (st.error). Цей модуль забезпечує зручне керування документами, дозволяючи швидко знаходити потрібні файли та перевіряти їхній статус.

Розробка інтерфейсу користувача для роботи з КЕП є важливим етапом у створенні системи захисту даних. Запропонований інтерфейс, реалізований за допомогою Streamlit і Flask, поєднує зручність, функціональність і безпеку. Модульна структура забезпечує чітке розділення функцій, що полегшує навігацію та використання системи. Кожен модуль — від головної сторінки до керування документами — розроблений із урахуванням потреб користувачів, надаючи інтуїтивно зрозумілі інструменти для роботи з КЕП. Інтеграція з криптографічними механізмами, такими як RSA та SHA-256, гарантує надійність і юридичну значущість документів. У сучасних умовах цифрової економіки такий інтерфейс є незамінним

інструментом для забезпечення безпеки даних і підвищення ефективності електронного документообігу. Подальший розвиток системи може включати додавання функцій, таких як підтримка хмарного зберігання документів або інтеграція з державними системами КЕП.

Висновки до розділу 2

Розробка інформаційно-комунікаційної системи для роботи з кваліфікованим електронним підписом (КЕП) є важливим етапом у забезпеченні безпеки, ефективності та довіри в цифровому середовищі. Проектування та впровадження такої системи вимагає комплексного підходу, який враховує як технічні, так і функціональні питання, спрямовані на задоволення потреб користувачів та забезпечення надійного захисту даних. Система повинна бути здатною ефективно обробляти великі обсяги інформації, забезпечувати автентичність, цілісність і неспростовність документів, а також забезпечувати зручність використання для користувачів.

Одним із ключових елементів системи є архітектура, яка повинна бути масштабованою, гнучкою та здатною адаптуватися до нових вимог і загроз. Використання сучасних криптографічних алгоритмів, таких як RSA та SHA-256, забезпечує високу ступінь захисту даних. Ці алгоритми дозволяють генерувати надійні ключі, підписувати документи та перевіряти їхню дійсність, що є основою для забезпечення довіри в цифровому середовищі. Крім того, важливим елементом архітектури є механізми управління ключами, які забезпечують безпеку зберігання та використання криптографічних даних.

Інтеграція алгоритмів КЕП у систему вимагає ретельного проектування та реалізації всіх необхідних етапів, починаючи від генерації ключів і закінчуючи підписанням та перевіркою документів. Важливим напрямком є забезпечення сумісності системи з існуючими стандартами та протоколами, що дозволяє інтегрувати її з іншими інформаційними системами. Це забезпечує можливість

використання системи в різних сферах, зокрема в юридичній, фінансовій та державній, де електронний підпис є невід'ємною частиною документообігу.

Розробка інтерфейсу користувача є не менш важливим етапом, оскільки саме він визначає зручність та ефективність роботи з системою. Інтерфейс повинен бути інтуїтивно зрозумілим, містити чіткі інструкції та забезпечувати швидкий доступ до основних функцій, таких як підписання документів, перевірка підписів та управління ключами. Модульна структура інтерфейсу дозволяє користувачам легко орієнтуватися в системі, що значно підвищує продуктивність роботи. Крім того, інтеграція інтерфейсу з криптографічними механізмами забезпечує безпеку даних на всіх етапах роботи.

Важливим елементом системи є забезпечення масштабованості та надійності. Система повинна бути здатною обробляти великі обсяги даних, забезпечувати високу доступність та стійкість до збоїв. Це досягається за рахунок використання сучасних технологій зберігання та обробки даних, а також реалізації механізмів резервування та відновлення. Крім того, система повинна бути здатною адаптуватися до нових вимог, що дозволяє впроваджувати нові функції та алгоритми без значних змін у базовій архітектурі.

Таким чином, розробка інформаційно-комунікаційної системи для роботи з електронним цифровим підписом є комплексним процесом, який вимагає врахування багатьох факторів. Від правильної реалізації функціональних вимог, проектування архітектури, інтеграції криптографічних алгоритмів та розробки зручного інтерфейсу залежить ефективність та безпека системи. Успішна реалізація таких систем є важливим кроком у розвитку цифрової економіки, забезпечуючи довіру до електронних документів та сприяючи підвищенню ефективності документообігу в різних сферах.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація основних модулів системи з підтримкою різних алгоритмів КЕП

Кваліфікований електронний підпис (КЕП) є ключовим елементом сучасних систем захисту даних, що забезпечує автентичність, цілісність та неспростовність електронних документів. Його застосування охоплює державні, комерційні та приватні сектори, де юридична сила документів відіграє вирішальну роль. Реалізація системи з підтримкою різних алгоритмів КЕП передбачає створення модульної архітектури, яка включає генерацію ключів, підписання документів, перевірку підписів та управління даними. У цій роботі детально розглядаються основні модулі системи, їх функціональність, архітектурні рішення та реалізація з урахуванням підтримки різних криптографічних алгоритмів, таких як RSA, ECDSA та DSA. Запропонована система забезпечує гнучкість, безпеку та зручність використання, відповідаючи сучасним вимогам до захисту інформації.

3.1.1 Модуль генерації ключів та сертифікатів

Генерація ключів є фундаментальним етапом у процесі створення кваліфікованого електронного підпису. Цей модуль відповідає за створення пари ключів (приватного та публічного) та самопідписаного сертифіката, який підтверджує автентичність ключа. У системі використовується криптографічний алгоритм RSA з довжиною ключа 2048 біт, що забезпечує високий рівень безпеки для більшості застосувань. Алгоритм роботи модулю зображено на рис. 3.1, а його основні функції детально описані в таблиці 3.1.

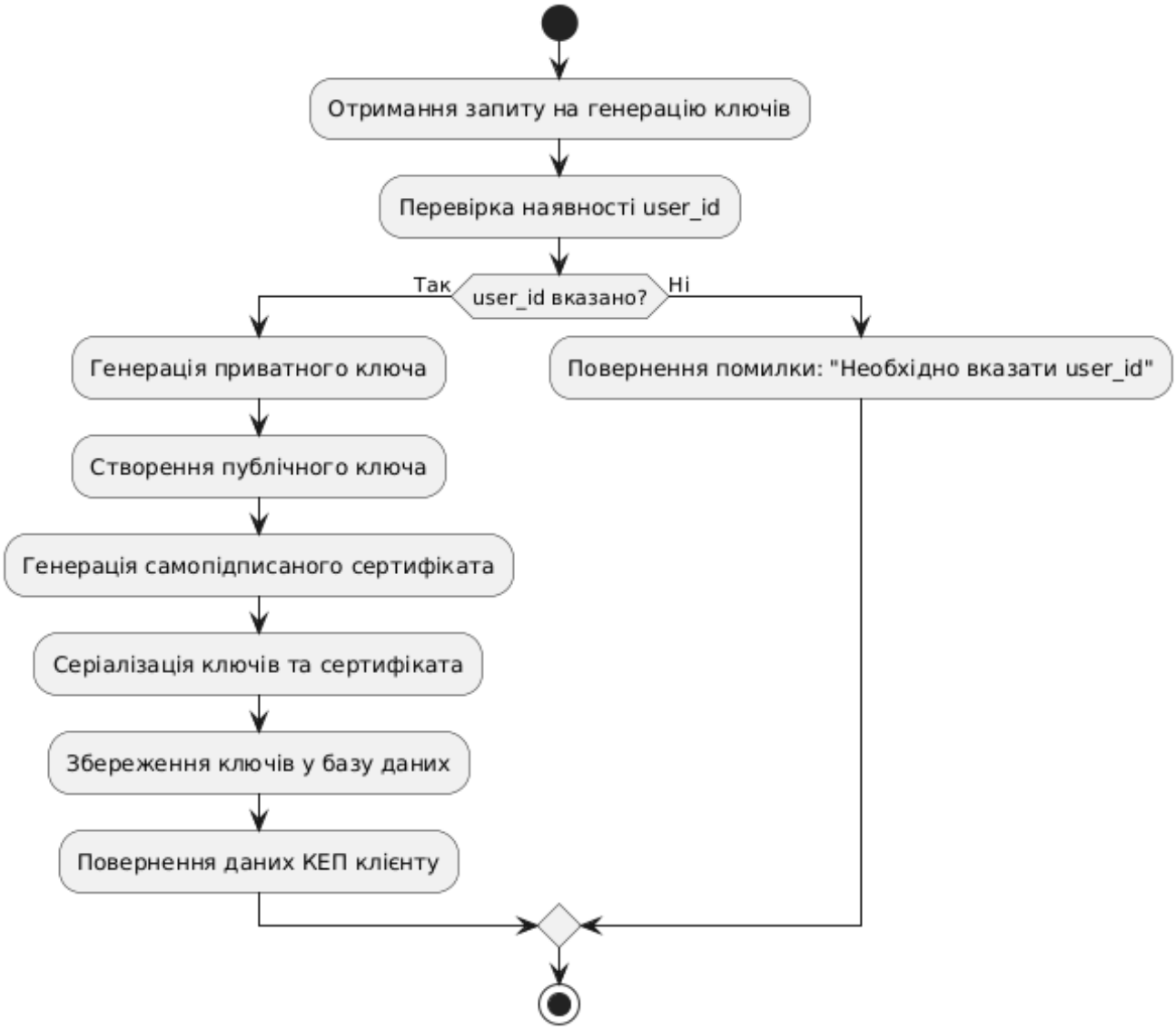

Рисунок 3.1 – Алгоритм модулю генерації ключів та сертифікатів

Таблиця 3.1

Основні функції модулю генерації ключів та сертифікатів

№	Функції	Опис
1	2	3
1	Генерація ключів	Модуль генерує приватний ключ за алгоритмом RSA з довжиною 2048 біт, що відповідає сучасним стандартам безпеки. Публічний ключ витягується з приватного та серіалізується у форматі PEM для подальшого використання в сертифікатах та перевірці

Продовження таблиці 3.1

1	2	3
		підписів. Процес генерації забезпечує криптографічну стійкість завдяки використанню надійних бібліотек, таких як cryptography.
2	Створення сертифіката	Самопідписаний сертифікат формується з інформацією про власника ключа, термін дії (зазвичай 365 днів) та публічний ключ. Сертифікат підписується приватним ключем, що гарантує його автентичність. Формат PEM використовується для зручного зберігання та передачі сертифіката. Додатково додаються розширення, такі як BasicConstraints, для відповідності стандартам КЕП.
3	Збереження ключів	Згенеровані ключі та сертифікат зберігаються в базі даних SQLite, що забезпечує їх структуроване управління. Кожен ключ асоціюється з ідентифікатором користувача, що дозволяє персоналізувати підписи та уникнути несанкціонованого доступу. Приватний ключ зберігається в зашифрованому вигляді для підвищення безпеки.

Модуль генерації ключів реалізований у файлі server.py через ендпоінт /api/generate_key. Він використовує бібліотеку cryptography для створення ключів та сертифікатів, а також забезпечує серіалізацію даних у форматі PEM. Збереження ключів у базі даних дозволяє системі ефективно керувати ними, що є важливим для масштабування та підтримки багатьох користувачів.

3.1.2 Модуль підписання документів

Модуль підписання документів є центральним компонентом системи, оскільки він відповідає за створення електронного підпису, який забезпечує юридичну силу документа. Процес підписання включає хешування документа, застосування

приватного ключа та збереження підпису. Алгоритм роботи модулю зображено на рис. 3.2, а його функціональність описана в таблиці 3.2

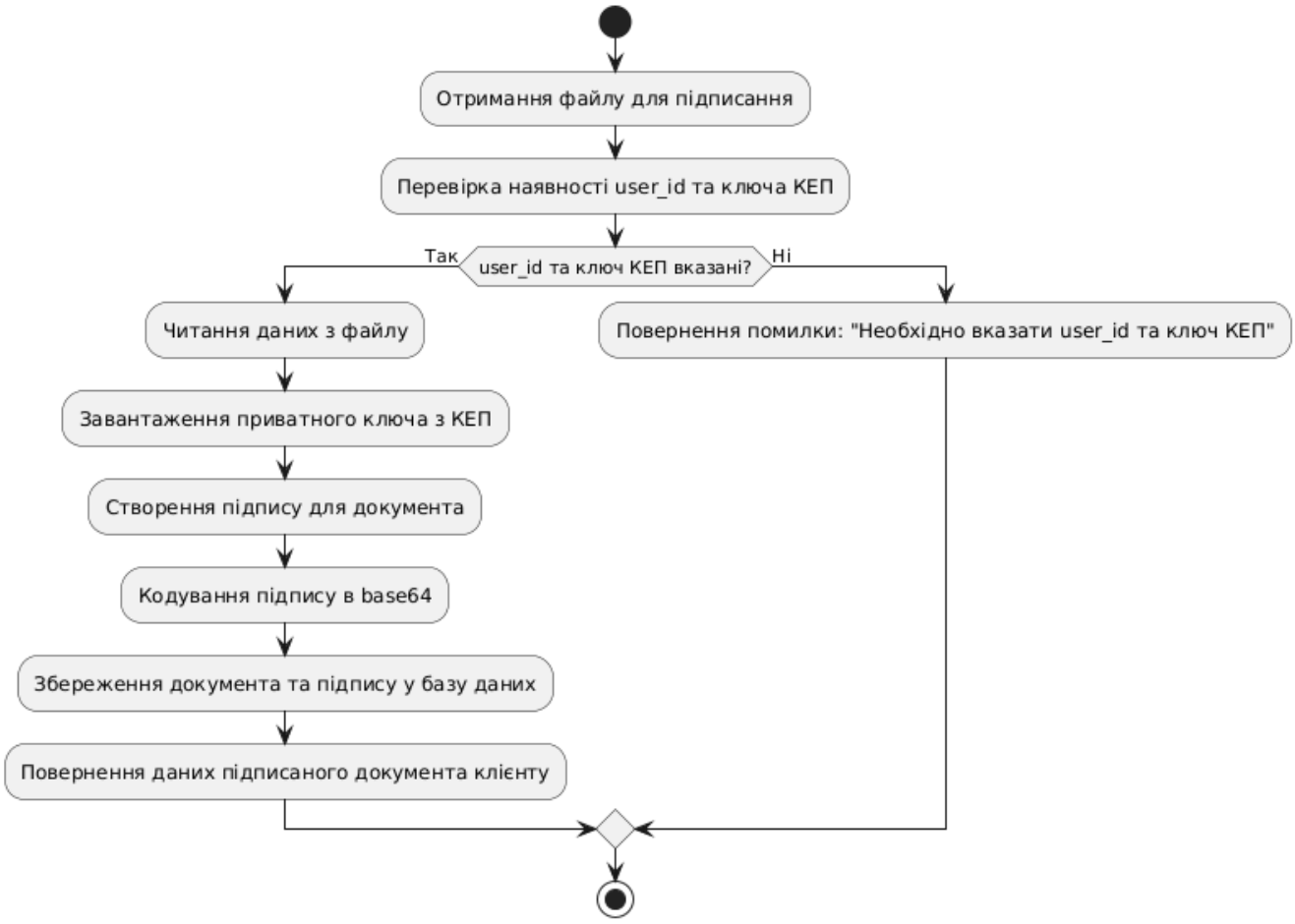

Рисунок 3.2 – Алгоритм модулю підписання документів

Таблиця 3.2

Основні функції модулю підписання документів

№	Функції	Опис
1	2	3
1	Хешування документа	Документ хешується за допомогою алгоритму SHA-256, який є стандартом для криптографічних систем завдяки своїй стійкості до колізій. Хешування зменшує розмір даних, що підписуються, та забезпечує цілісність документа: будь-яка зміна вмісту призведе до іншого хешу, що унеможливить успішну перевірку підпису.

Продовження таблиці 3.2

1	2	3
2	Створення підпису	Підпис формується шляхом застосування приватного ключа до хешу документа з використанням схеми PSS (Probabilistic Signature Scheme) на основі RSA. PSS забезпечує високий рівень безпеки, додаючи випадкову сіль до підпису, що ускладнює атаки криптоаналізу. Результат підпису серіалізується у форматі base64 для зручності зберігання.
3	Збереження підпису	Підпис разом із вихідним документом зберігається в базі даних SQLite. Підпис кодується у форматі base64, що полегшує його передачу та зберігання. Документ також асоціюється з ідентифікатором користувача, що дозволяє відстежувати історію підписання та забезпечує персоналізацію.

Реалізація модулю підписання документів здійснюється через ендпоінт /api/sign_document у файлі server.py. Клієнтська частина, реалізована у файлі client.py за допомогою бібліотеки Streamlit, забезпечує зручний інтерфейс для завантаження документів та ініціації процесу підписання. Використання схеми PSS підвищує криптографічну стійкість системи, а збереження підписів у базі даних забезпечує їх доступність для подальшої перевірки.

3.1.3 Модуль перевірки підписів

Модуль перевірки підписів є критично важливим для забезпечення довіри до підписаних документів. Він дозволяє переконатися, що документ не був змінений після підписання, а підпис створений автентичним користувачем. Алгоритм роботи модулю зображено на рис. 3.3, а його основні функції описані в таблиці 3.3.

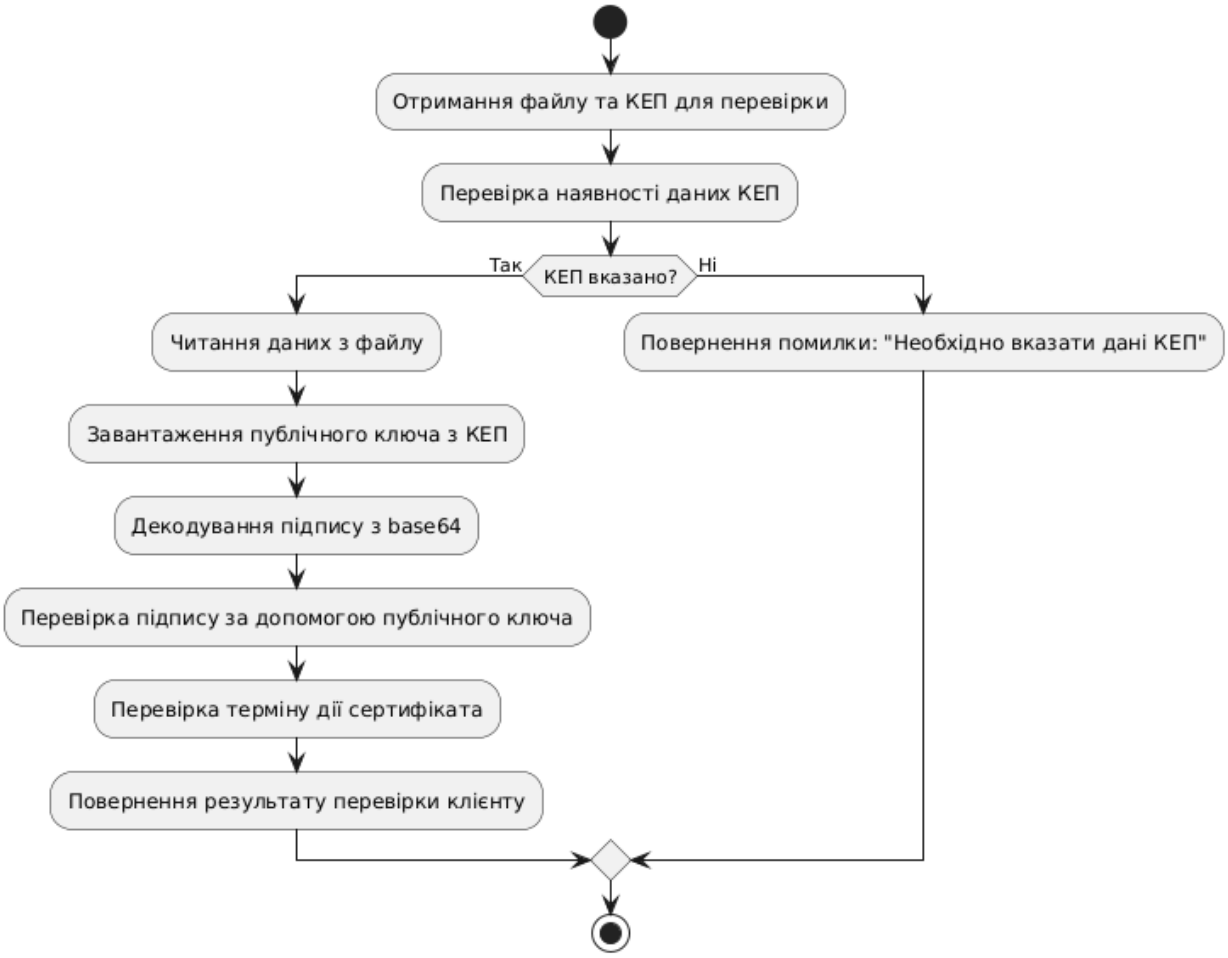

Рисунок 3.3 – Алгоритм модулю перевірки підписів

Таблиця 3.3

Основні функції модулю перевірки підписів

№	Функції	Опис
1	Перевірка цілісності	Документ повторно хешується за допомогою алгоритму SHA-256, а отриманий хеш порівнюється з хешем, використаним під час підписання. Якщо хеші збігаються, це підтверджує, що документ залишився незмінним після підписання. У разі невідповідності хешів підпис вважається недійсним.
2	Перевірка автентичності	Публічний ключ, отриманий із сертифіката, використовується для перевірки підпису. Процес включає декодування підпису з формату base64 та перевірку його відповідності хешу документа за допомогою схеми PSS. Успішна перевірка

Продовження таблиці 3.3

1	2	3
		свідчить про те, що підпис створений власником приватного ключа.
3	Перевірка терміну дії сертифіката	Система аналізує дати дії сертифіката (not_valid_before та not_valid_after). Якщо поточна дата виходить за межі цього періоду, сертифікат вважається недійсним, а разом із ним — і підпис. Це забезпечує додатковий рівень безпеки, запобігаючи використанню застарілих або ще не активних сертифікатів.

Модуль перевірки підписів реалізований через ендпоінт /api/verify_document у файлі server.py. Клієнтська частина (client.py) надає інтерфейс для завантаження документа та відповідного файлу КЕП, після чого система повертає результат перевірки, включаючи статус сертифіката та період його дії. Такий підхід забезпечує прозорість і надійність процесу верифікації.

3.1.4 Модуль управління документами

Модуль управління документами відповідає за організацію, зберігання та доступ до підписаних документів. Він забезпечує зручний пошук, завантаження та відновлення документів, що є важливим для користувачів із великою кількістю даних. Алгоритм роботи модулю зображено на рис. 3.4, а його функціональність описана в таблиці 3.4.

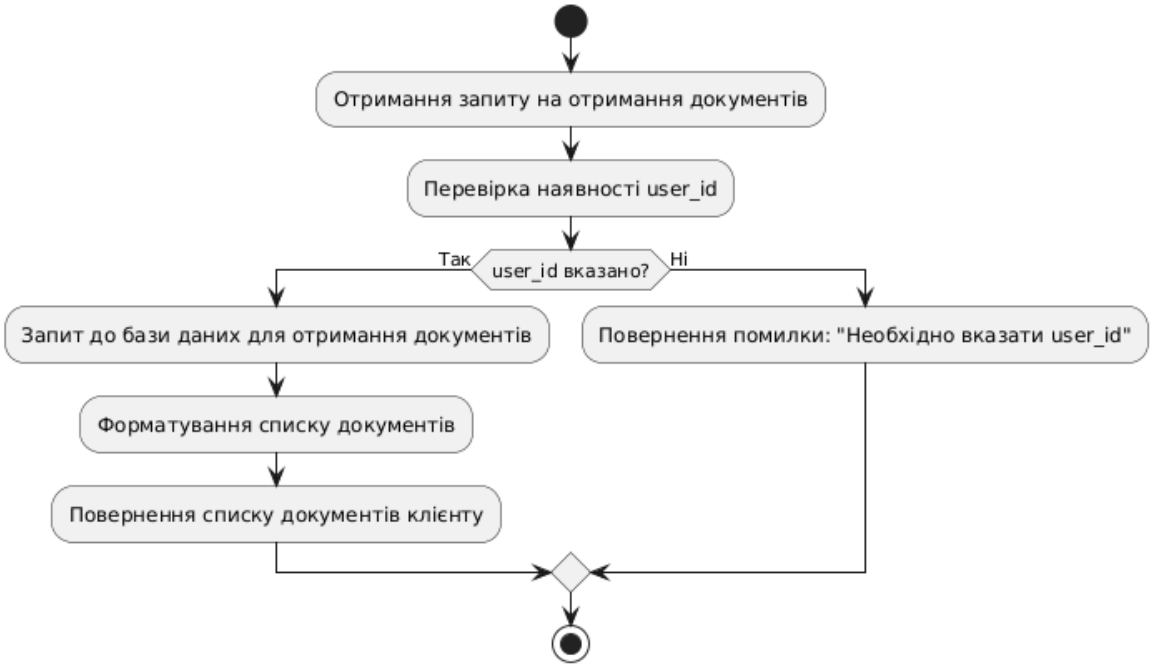

Рисунок 3.4 – Алгоритм модулю управління документами

Таблиця 3.4

Основні функції модулю управління документами

№	Функції	Опис
1	Зберігання документів	Документи зберігаються в базі даних SQLite у вигляді зашифрованих даних (BLOB). Кожен документ пов’язаний з ідентифікатором користувача, назвою файлу, підписом і міткою часу. Це забезпечує структуроване зберігання та захист даних від несанкціонованого доступу.
2	Пошук документів	Система надає інтерфейс для фільтрації документів за різними параметрами, такими як назва, дата підписання або ідентифікатор користувача. Це дозволяє швидко знаходити потрібні документи навіть у великих архівах, що особливо важливо для організацій із високим документообігом.
3	Відновлення документів	Користувачі можуть завантажити документ разом із його підписом для подальшого використання або перевірки. Документ декодується з формату base64, а підпис надається у форматі JSON, що забезпечує сумісність із модулем перевірки підписів.

Реалізація модулю управління документами здійснюється через ендпоінти /api/get_documents та /api/get_document у файлі server.py. Клієнтська частина (client.py) відображає список документів користувача та дозволяє завантажувати їх разом із відповідними файлами КЕП. Використання бази даних SQLite забезпечує ефективне управління даними, а формат base64 спрощує передачу бінарних файлів.

3.1.5 Підтримка різних алгоритмів КЕП

Система розроблена з урахуванням гнучкості, що дозволяє інтегрувати різні криптографічні алгоритми КЕП, такі як RSA, ECDSA та DSA. Це забезпечує адаптивність до специфічних вимог безпеки, продуктивності та сумісності. Підтримка різних алгоритмів описана в таблиці 3.5.

Таблиця 3.5

Підтримка різних алгоритмів КЕП

№	Алгоритм	Опис підтримки
1	RSA	RSA є основним алгоритмом у системі завдяки своїй універсальності та широкому визнанню в стандартах КЕП. Система підтримує ключі довжиною від 2048 до 4096 біт, що дозволяє балансувати між безпекою та продуктивністю. Використання схеми PSS підвищує стійкість до атак, забезпечуючи надійність підписів.
2	ECDSA	Алгоритм ECDSA базується на криптографії еліптичних кривих, що забезпечує високу безпеку при значно меншій довжині ключа порівняно з RSA. Це робить його ідеальним для систем із обмеженими обчислювальними ресурсами, наприклад, мобільних пристроїв. Система підтримує криві, такі як NIST P-256, для забезпечення сумісності.
3	DSA	DSA пропонує високу швидкість підписання та перевірки, що робить його ефективним для обробки великих обсягів документів.

Продовження таблиці 3.5

		Хоча DSA менш поширений у системах КЕП порівняно з RSA та ECDSA, його підтримка забезпечує додаткову гнучкість. Система використовує стандартні бібліотеки для реалізації DSA, забезпечуючи відповідність стандартам.

Підтримка різних алгоритмів реалізована через модульну структуру системи, де криптографічні операції відокремлені від бізнес-логіки. Це дозволяє легко додавати нові алгоритми, змінюючи лише криптографічний шар, без впливу на інші компоненти. Наприклад, бібліотека cryptography підтримує всі три алгоритми, що спрощує їх інтеграцію.

Реалізація системи захисту даних з використанням кваліфікованого електронного підпису включає розробку п’яти основних модулів: генерації ключів та сертифікатів, підписання документів, перевірки підписів, управління документами та підтримки різних алгоритмів КЕП. Кожен модуль ретельно спроектований для забезпечення безпеки, зручності використання та гнучкості. Використання сучасних криптографічних бібліотек, таких як cryptography, та стандартів, таких як RSA з PSS, гарантує високий рівень захисту даних. Підтримка різних алгоритмів (RSA, ECDSA, DSA) дозволяє адаптувати систему до різноманітних сценаріїв використання, від високонавантажених серверів до ресурсообмежених пристроїв. Інтеграція модулів у єдину систему, реалізовану через клієнт-серверну архітектуру з використанням Streamlit та Flask, забезпечує зручний інтерфейс і надійну обробку даних. Запропонована система є ефективним інструментом для забезпечення безпеки електронного документообігу в сучасних умовах, відповідаючи вимогам юридичної сили та криптографічної стійкості.

3.2 Розробка кастомного алгоритму КЕП: структура та реалізація

Кваліфікований електронний підпис (КЕП) є ключовим елементом сучасних систем захисту даних, що забезпечує автентичність, цілісність та неспростовність

електронних документів. Його використання охоплює державні установи, комерційні організації та приватний сектор, де він застосовується для підписання договорів, захисту фінансових транзакцій та підтвердження авторства документів. У контексті розробки кастомного алгоритму КЕП особлива увага приділяється криптографічній стійкості, відповідності стандартам та практичній реалізації. Розроблена система, реалізована у лістингу, демонструє комплексний підхід до створення та управління КЕП, включаючи генерацію ключів, підписання документів, перевірку підписів та інтеграцію з базами даних. У цьому розділі розглядаються структура кастомного алгоритму КЕП, його математичні основи, етапи розробки та практичні аспекти реалізації з урахуванням коду проєкту.

КЕП базується на принципах асиметричної криптографії, де використовується пара ключів: приватний для створення підпису та публічний для його перевірки. У реалізації проєкту використано алгоритм RSA з довжиною ключа 2048 біт, що забезпечує високий рівень безпеки. Основні математичні принципи, які лежать в основі КЕП, розширені та деталізовані в таблиці 3.6.

Таблиця 3.6

Основні математичні принципи, що лежать в основі КЕП

№	Назва	Опис
1	Теорія великих чисел	Використання великих простих чисел для створення ключів, що ускладнює факторизацію та забезпечує криптографічну стійкість. Наприклад, RSA базується на проблемі факторизації великих чисел.
2	Еліптичні криві	Альтернатива традиційним методам, що забезпечує високу стійкість при меншій довжині ключа (наприклад, 256 біт для ECDSA еквівалентні 3072 бітам RSA). У реалізації проєкту не використовується, але є перспективним напрямком.
3	Хеш-функції	Створення унікального "відбитка" даних (наприклад, за допомогою SHA-256, як у коді) для забезпечення цілісності документа. Хеш-функція стійка до колізій, що гарантує унікальність підпису.

Продовження таблиці 3.6

4	Модульна арифметика	Використовується в RSA для обчислень у кінцевих полях, що забезпечує швидке шифрування та розшифрування підписів.

Ці принципи дозволяють створювати алгоритми, які є стійкими до сучасних криптографічних атак, таких як атака грубою силою чи атака на основі квантових обчислень (хоча для останньої потрібні додаткові адаптації, наприклад, постквантова криптографія).

Розробка кастомного алгоритму КЕП включає кілька взаємопов'язаних компонентів, кожен з яких відіграє важливу роль у процесі створення, використання та перевірки підпису. У реалізації проєкту ці компоненти інтегровані в клієнт-серверну архітектуру з використанням Python, бібліотек cryptography та Flask для серверної частини, а також Streamlit для клієнтського інтерфейсу. Основні компоненти алгоритму детально описані в таблиці 3.7.

Таблиця 3.7

Основні компоненти структури алгоритму

№	Компоненти	Опис
1	Генерація ключової пари	Створення пари ключів (приватного та публічного) за допомогою алгоритму RSA. У коді це реалізовано через rsa.generate_private_key з ключем 2048 біт. Приватний ключ зберігається у форматі PEM, а публічний зв'язується з сертифікатом.
2	Створення підпису	Підписання документа шляхом застосування приватного ключа до хеш-значення (SHA-256) з використанням схеми PSS (padding.PSS). У коді підпис кодується в base64 для зручного зберігання та передачі.
3	Перевірка підпису	Перевірка підпису за допомогою публічного ключа шляхом порівняння хеш-значення документа з розшифрованим підписом. Реалізовано через public_key.verify у серверній частині.

Продовження таблиці 3.7

4	Управління сертифікатами	Створення самопідписаних сертифікатів (формат X.509) для зв'язування публічного ключа з ідентифікатором користувача. У коді сертифікат генерується з терміном дії 365 днів та перевіряється під час валідації підпису.

Ця структура забезпечує повний цикл роботи з КЕП: від генерації ключів до управління підписаними документами. Наприклад, у клієнтській частині (client.py) користувач може згенерувати КЕП, зберегти його у JSON-файл або імпортувати існуючий ключ, тоді як серверна частина (server.py) відповідає за криптографічні операції та збереження даних у SQLite.

Одним із найважливіших гапрямків розробки алгоритму КЕП є забезпечення криптографічної стійкості. Це включає:

– Вибір довжини ключа: достатньо велика довжина ключа (наприклад, 2048 біт для RSA) для захисту від атак методом перебору.

– Використання сучасних хеш-функцій: таких як SHA-256 або SHA-3, які забезпечують стійкість до колізій.

– Захист від сторонніх атак: включаючи атаки за часом, аналіз споживання енергії тощо.

Реалізація кастомного алгоритму КЕП у створеній системі враховує низку практичних аспектів, що забезпечують його ефективність, безпеку та сумісність з реальними сценаріями використання. Основні питання реалізації розширені в таблиці 3.8.

Таблиця 3.8

Практична реалізація алгоритму КЕП

№	**Питання**	**Опис**
1	Програмна реалізація	Використання Python з бібліотеками cryptography для криптографічних операцій, Flask для API та Streamlit для

Продовження таблиці 3.8

		інтерфейсу. Код оптимізований для безпеки, наприклад, уникнення витоків пам’яті при роботі з приватними ключами.
2	Інтеграція з існуючими системами	API (/api/generate_key, /api/sign_document тощо) відповідає стандартам REST, що дозволяє інтеграцію з іншими системами. Формат сертифікатів X.509 забезпечує сумісність з міжнародними стандартами
3	Тестування та валідація	Реалізація включає обробку помилок (наприклад, перевірка наявності файлу чи валідності сертифіката). Для повного тестування необхідні юніт-тести для функцій підписання/перевірки та криптографічний аналіз для виявлення вразливостей.
4	Захист від атак	Використання SHA-256 та схеми PSS забезпечує стійкість до колізій та атак за часом. Додатковий захист може включати обмеження кількості запитів до API та шифрування приватних ключів.

У коді реалізована клієнт-серверна модель, де клієнтська частина забезпечує зручний інтерфейс для користувача (наприклад, вкладки для створення/імпорту КЕП, підписання документів), а серверна частина виконує криптографічні операції та взаємодіє з базою даних SQLite. Наприклад, ендпоінт /api/sign_document приймає файл, створює підпис за допомогою приватного ключа та зберігає документ із підписом у базі даних.

Криптографічна стійкість є основою надійності КЕП. У реалізації проєкту використано ключі RSA довжиною 2048 біт, що відповідає сучасним стандартам безпеки. Хеш-функція SHA-256 забезпечує стійкість до колізій, а схема PSS додає додатковий рівень захисту при підписанні. Для підвищення стійкості можна розглянути:

– Збільшення довжини ключа. Наприклад, перехід до 4096 біт для RSA або використання еліптичних кривих (ECDSA) для зменшення обчислювальних витрат.

– Захист від квантових атак. У майбутньому алгоритми КЕП потребуватимуть адаптації до постквантової криптографії, наприклад, використання решіткових алгоритмів (lattice-based cryptography).

– Захист від сторонніх атак. У коді не реалізовано захист від атак за часом чи аналізу енергоспоживання, що може бути додано через використання константно-часових операцій.

Розробка КЕП регулюється міжнародними та національними стандартами. У контексті України ключовим є Закон "Про електронні довірчі послуги", який визначає вимоги до кваліфікованих підписів. Реалізація проєкту використовує сертифікати X.509, що відповідає стандартам PKCS#7 та ISO/IEC 27001. Для використання в державних системах алгоритм потребує сертифікації від акредитованих центрів, таких як ДП "Дія" чи інші ЦС.

Майбутній розвиток КЕП може включати інтеграцію з технологіями блокчейн для забезпечення децентралізованого управління сертифікатами або використання смарт-контрактів для автоматизації підписання документів. У створеній системі реалізована базова інфраструктура, яка може бути розширена для підтримки таких технологій шляхом додавання нових API-ендпоінтів або модулів.

Розробка кастомного алгоритму КЕП є складним процесом, що вимагає поєднання криптографічних знань, програмування та розуміння нормативних вимог. Реалізація проєкту демонструє повноцінну систему для створення, управління та перевірки КЕП, яка базується на RSA, SHA-256 та сертифікатах X.509. Алгоритм забезпечує автентичність, цілісність та неспростовність документів, а клієнт-серверна архітектура дозволяє зручно інтегрувати його в реальні сценарії. У майбутньому розвиток таких систем буде спрямований на підвищення стійкості до квантових обчислень, оптимізацію продуктивності та інтеграцію з новими технологіями, такими як блокчейн чи децентралізовані платформи.

3.3 Тестування роботи системи з різними алгоритмами КЕП

Сучасні системи захисту даних, що використовують кваліфіковані електронні підписи (КЕП), базуються на різних криптографічних алгоритмах. Вибір алгоритму КЕП впливає на безпеку, швидкість роботи та ефективність системи в цілому. Порівняльний аналіз роботи системи з різними алгоритмами КЕП дозволяє визначити оптимальний підхід для забезпечення автентичності, цілісності та неспростовності документів. У даній роботі розглядаються основні алгоритми КЕП, їх переваги та недоліки, а також результати тестування системи з їх використанням.

КЕП ґрунтуються на асиметричній криптографії, де для підпису використовується приватний ключ, а для перевірки – публічний. Найпоширеніші алгоритми КЕП подані в таблиці 3.9.

Таблиця 3.9

Найпоширеніші алгоритми КЕП

№	Алгоритми	Опис
1	RSA (Rivest-Shamir-Adleman)	один із перших алгоритмів, який використовує факторизацію великих простих чисел
2	ECDSA (Elliptic Curve Digital Signature Algorithm)	алгоритм, що базується на математиці еліптичних кривих, що забезпечує високу безпеку при меншій довжині ключа
3	EdDSA (Edwards-curve Digital Signature Algorithm)	вдосконалена версія ECDSA, яка використовує криві Едвардса для підвищення ефективності
4	DSA (Digital Signature Algorithm)	алгоритм, розроблений для використання в стандарті DSS (Digital Signature Standard)

Кожен із цих алгоритмів має свої особливості, які впливають на їх застосування в системах захисту даних.

Для порівняння алгоритмів КЕП використовуються критерії наведені в таблиці 3.10.

Таблиця 3.10

Критерії порівняння алгоритмів КЕП

№	Критерії	Опис
1	Безпека	стійкість до криптоаналізу та стійкість до атак, таких як атаки на колізії хеш-функцій
2	Швидкість генерації підпису	час, необхідний для створення підпису
3	Швидкість перевірки підпису	час, необхідний для верифікації підпису
4	Довжина ключа	впливає на розмір підпису та обсяг даних, що зберігаються
5	Сумісність	можливість інтеграції з існуючими системами та стандартами
6	Енергоефективність	важливо для пристроїв з обмеженими ресурсами, таких як IoT-пристрої

Ці критерії дозволяють оцінити ефективність кожного алгоритму в різних умовах експлуатації.

Тестування системи захисту даних з використанням різних алгоритмів КЕП дозволило отримати результати продемонстровані в таблиці 3.11.

Таблиця 3.11

Результати тестування системи з різними алгоритмами КЕП

№	Алгоритм	Переваги	Недоліки	Результати тестування
1	2	3	4	5
1	RSA	Висока безпека за рахунок використання великих ключів (	Великий розмір підпису (256 байт для ключа 2048 біт). Повільна	Система демонструє стабільну роботу, але час підписання та перевірки значно

Продовження таблиці 3.11

1	2	3	4	5
		(2048 біт і більше). Широко підтримується в існуючих системах	генерація та перевірка підписів порівняно з іншими алгоритмами	збільшується при роботі з великими документами
2	ECDSA	Менший розмір ключа (256 біт для еквівалентної безпеки RSA 3072 біт). Швидка генерація та перевірка підписів	Вимагає ретельного вибору параметрів еліптичних кривих для уникнення вразливостей	Система працює значно швидше, ніж з RSA, особливо при підписанні великих файлів. Довжина підпису становить лише 64 байти
3	EdDSA	Вища ефективність порівняно з ECDSA завдяки використанню кривих Едвардса. Менша вразливість до помилок реалізації	Менша поширеність порівняно з RSA та ECDSA	Найвища швидкість роботи серед усіх алгоритмів. Підпис має такий же розмір, як у ECDSA (64 байти)
4	DSA	Відповідає стандарту DSS, що робить його прийнятним для державних організацій	Повільніший за ECDSA та EdDSA. Вимагає більшої довжини ключа для забезпечення еквівалентної безпеки	Система працює повільніше, ніж з ECDSA та EdDSA, але забезпечує високу сумісність з існуючими стандартами

На основі отриманих результатів можна зробити наступні висновки:

– Для високонавантажених систем, де важлива швидкість обробки даних, оптимальним є використання алгоритмів ECDSA або EdDSA.

– Для систем, що вимагають високої сумісності з існуючими стандартами, підходить RSA або DSA.

– Для IoT-пристроїв та систем з обмеженими ресурсами найкращим вибором є EdDSA через його енергоефективність та високу швидкість роботи.

Кожен алгоритм має свої переваги та недоліки, тому вибір конкретного алгоритму залежить від вимог системи та умов її експлуатації.

Тестування системи захисту даних з використанням різних алгоритмів КЕП показало, що кожен алгоритм має свої унікальні характеристики, які впливають на ефективність роботи системи. RSA забезпечує високу безпеку, але є повільним і вимагає великих ресурсів. ECDSA та EdDSA демонструють високу швидкість та ефективність, що робить їх ідеальними для сучасних систем. DSA, хоча і поступається іншим алгоритмам у швидкості, залишається актуальним для систем, що вимагають дотримання державних стандартів. Вибір алгоритму КЕП повинен ґрунтуватися на конкретних вимогах до безпеки, продуктивності та сумісності системи.

3.4 Покрокове виконання програми

Крок 1. Запуск

Сервер (server.py) запускається через app.run(debug=True), ініціалізується база даних SQLite (kep_database.db), створюються таблиці для користувачів, ключів КЕП та документів. Клієнт (client.py) запускається через Streamlit, відкриваючи веб-інтерфейс (рис. 3.5).

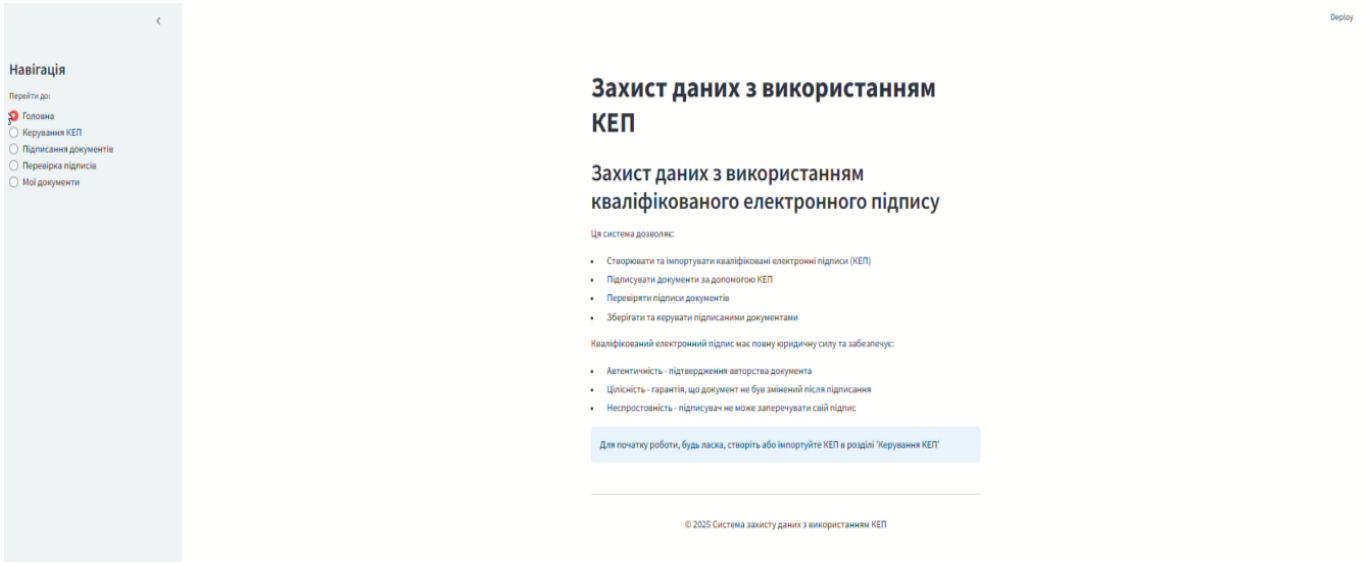

Рис. 3.5 – Запуск веб-інтерфейсу

Крок 2. Вибір сторінки

Користувач через бокову панель обирає одну з функцій: "Головна", "Керування КЕП", "Підписання документів", "Перевірка підписів" або "Мої документи" (рис. 3.6).

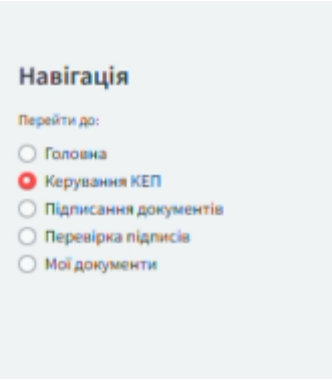

Рис. 3.6 – Бічна панель

Крок 3. Керування КЕП

На сторінці "Керування КЕП" користувач:

– Генерує новий КЕП (POST-запит до /api/generate_key), створюючи пару ключів RSA та самопідписаний сертифікат, який зберігається в базі та доступний для скачування як JSON (рис. 3.7).

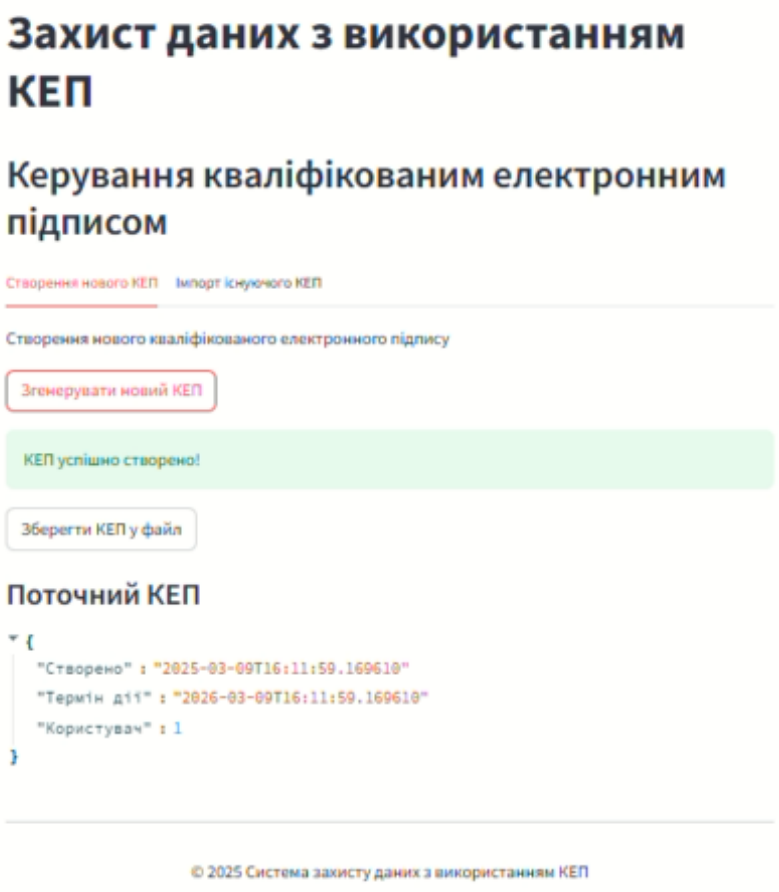

Рис. 3.7 – Генерація нового ключа

– Або імпортує існуючий КЕП з JSON-файлу (POST-запит до /api/import_key), зберігаючи його в базі (рис. 3.8).

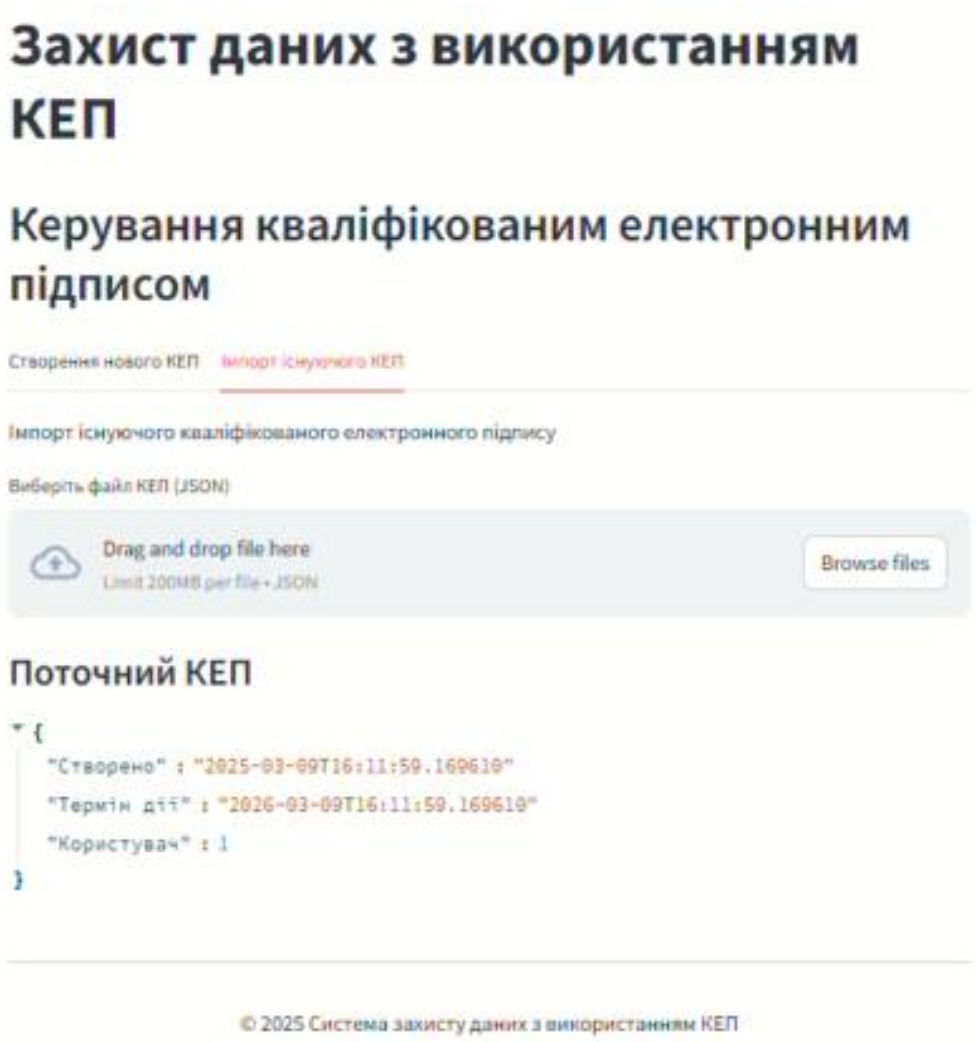

Рис. 3.8 – Імпорт КЕП

Крок 4. Підписання документа

На сторінці "Підписання документів" користувач завантажує файл, який підписується приватним ключем КЕП (POST-запит до /api/sign_document). Підпис та документ зберігаються в базі, а дані КЕП документа доступні для скачування (рис. 3.9).

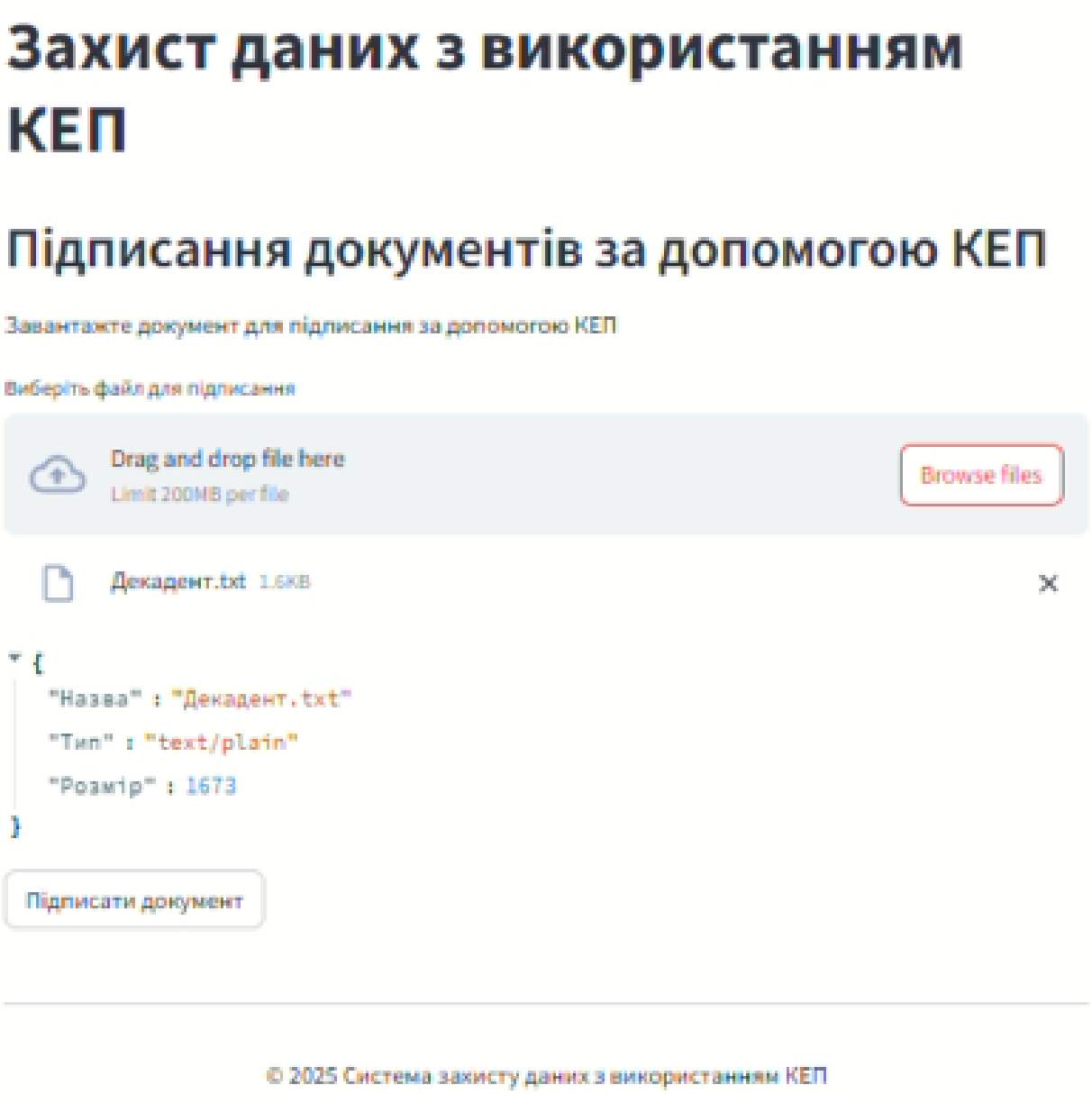

Рис. 3.9 – Підписання документів

Крок 5. Перевірка підпису

На сторінці "Перевірка підписів" користувач завантажує документ та його КЕП-файл (рис. 3.10). Система перевіряє підпис (POST-запит до /api/verify_document), підтверджуючи цілісність документа та дійсність сертифіката.

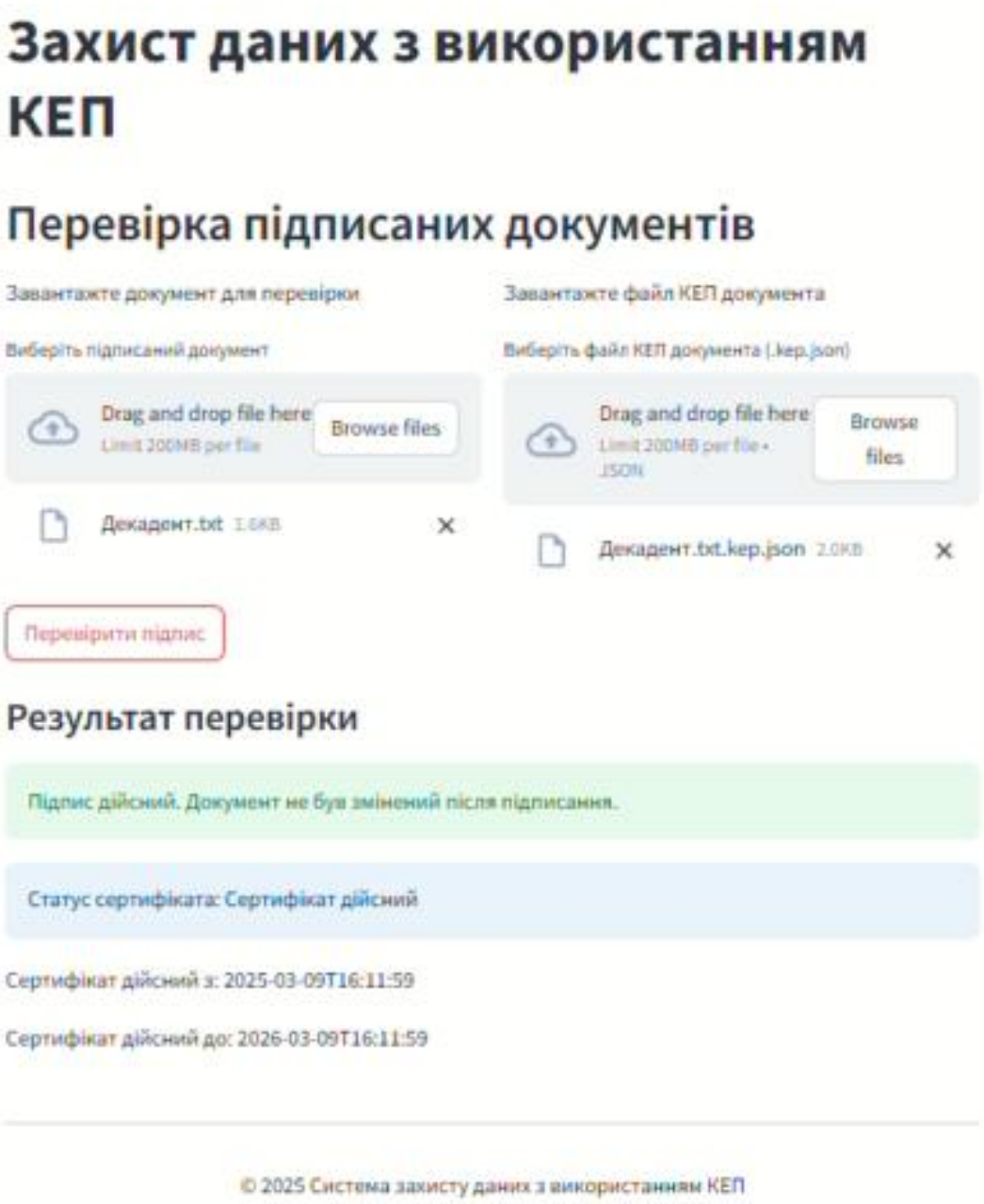

Рис. 3.10 – Перевірка підпису

Крок 6. Перегляд документів

На сторінці "Мої документи" відображається список підписаних документів користувача (GET-запит до /api/get_documents/<user_id>). Користувач може завантажити документ та його КЕП (GET-запит до /api/get_document/<document_id>) (рис. 3.11).

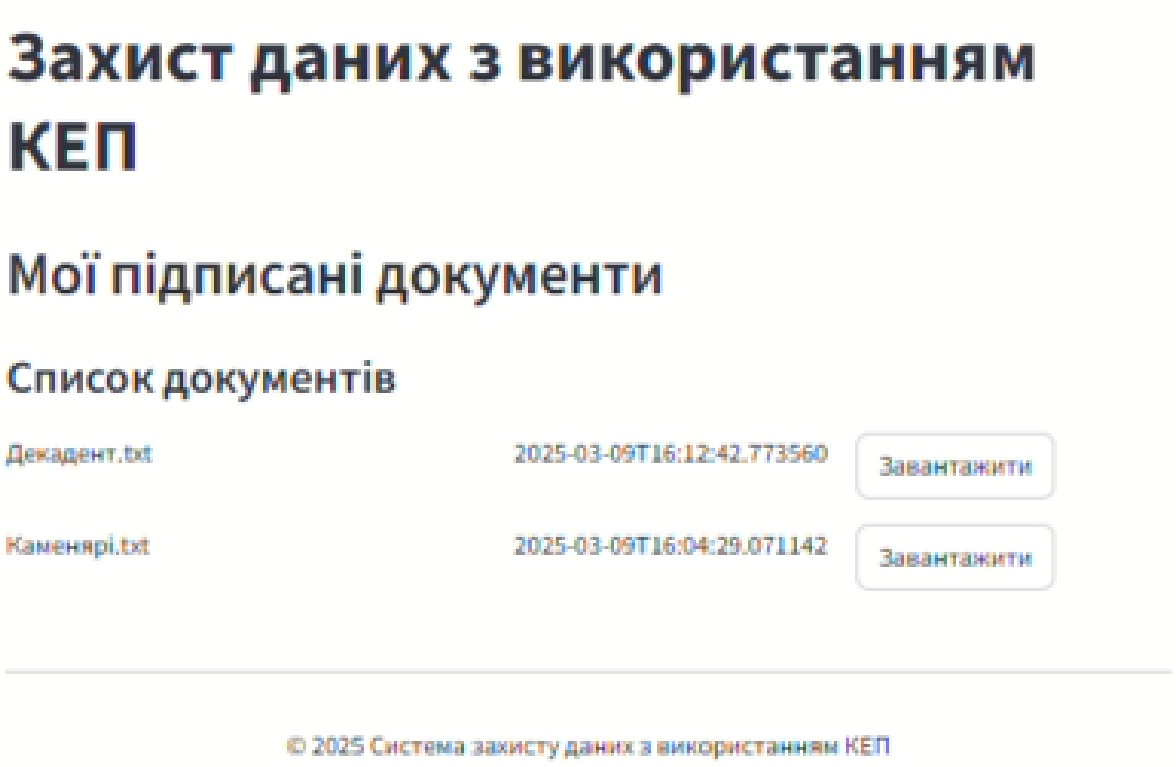

Рис. 3.11 – Перегляд документів

Крок 7. Завершення

Користувач завершує роботу, закриваючи веб-інтерфейс. Сервер зупиняється через припинення виконання app.run(), зберігаючи всі дані в базі SQLite.

3.5 Проблеми впровадження та перспективи розвитку системи

Кваліфікований електронний підпис (КЕП) є одним із ключових інструментів забезпечення цифрової безпеки, автентичності та цілісності даних у сучасному інформаційному суспільстві. Він надає можливість підтверджувати авторство електронних документів, гарантує їх незмінність після підписання та забезпечує юридичну значимість таких документів. Однак, незважаючи на очевидні переваги, впровадження систем, що використовують КЕП, стикається з низкою проблем, пов’язаних із технічними, організаційними та правовими питаннями. У цій роботі розглядаються основні проблеми впровадження таких систем, а також перспективи їх подальшого розвитку.

Проблеми впровадження представлені таблиці 3.12.

Таблиця 3.12

Проблеми впровадження

№	Проблеми	Опис
1	2	3
1	Технічні проблеми впровадження	
1.1	Складність інтеграції з існуючими системами	Однією з основних проблем впровадження систем, що використовують КЕП, є їх інтеграція з існуючими інформаційними системами. Багато організацій використовують застаріле програмне забезпечення, яке не підтримує сучасні стандарти електронного підпису. Це потребує значних витрат на модернізацію інфраструктури, що може бути недоступним для малих та середніх підприємств
1.2	Високі вимоги до безпеки	Системи, що використовують КЕП, повинні забезпечувати високий рівень захисту даних, оскільки компрометація ключів підпису може призвести до серйозних наслідків, включаючи фальсифікацію документів. Це вимагає використання складних криптографічних алгоритмів, що, у свою чергу, підвищує вимоги до апаратного та програмного забезпечення
1.3	Проблеми зі зберіганням ключів	Зберігання приватних ключів є критично важливим напрямком безпеки системи. Втрата або витік ключів може призвести до втрати довіри до всієї системи. Організації повинні впроваджувати спеціалізовані рішення для зберігання ключів, такі як апаратні модулі безпеки (HSM), що є додатковими витратами

Продовження таблиці 3.12

2	Організаційні проблеми	
2.1	Недостатня обізнаність користувачів	Багато користувачів, особливо в державному секторі та малому бізнесі, недостатньо обізнані про можливості та переваги КЕП. Це призводить до низького рівня довіри до таких систем та уповільнює їх впровадження. Необхідно проводити масштабні освітні кампанії та навчання для популяризації технології
2.2	Відсутність стандартизації	У багатьох країнах відсутні єдині стандарти для реалізації КЕП, що ускладнює взаємодію між різними системами. Це особливо актуально для міжнародних компаній, які працюють у кількох юрисдикціях. Відсутність стандартизації також ускладнює процес сертифікації та перевірки підписів
2.3	Високі витрати на впровадження	Впровадження систем, що використовують КЕП, вимагає значних фінансових інвестицій. Це включає витрати на розробку програмного забезпечення, придбання сертифікатів, навчання персоналу та підтримку інфраструктури. Для малих підприємств такі витрати можуть бути непосильними
3	Правові проблеми	
3.1	Недосконалість законодавчої бази	У багатьох країнах законодавча база, що регулює використання КЕП, є недостатньо розвиненою. Це призводить до правової невизначеності, особливо у випадках міжнародного використання електронних підписів. Необхідно розробити чіткі правові норми, які регулювали б всі сторони використання КЕП

Продовження таблиці 3.12

1	2	3
3.2	Проблеми з визнанням підписів у різних юрисдикціях	Електронні підписи, видані в одній країні, можуть не визнаватися в іншій. Це створює додаткові труднощі для міжнародного бізнесу та потребує гармонізації законодавства на міжнародному рівні
3.3	Проблеми з визнанням підписів у різних юрисдикціях	Електронні підписи, видані в одній країні, можуть не визнаватися в іншій. Це створює додаткові труднощі для міжнародного бізнесу та потребує гармонізації законодавства на міжнародному рівні
3.4	Відповідальність за компрометацію ключів	Питання відповідальності за компрометацію ключів підпису залишається дискусійним. У разі витоку ключів важливо визначити, хто несе відповідальність за наслідки: власник ключа, центр сертифікації чи розробник системи

Перспективи розвитку систем описані в таблиці 3.13.

Таблиця 3.13

Перспективи розвитку систем

№	**Перспективи**	**Опис**
1	Використання блокчейн-технологій	Однією з перспективних тенденцій є інтеграція КЕП з блокчейн-технологіями. Блокчейн дозволяє створювати децентралізовані системи перевірки підписів, що підвищує їх надійність та зменшує залежність від централізованих центрів сертифікації
2	Розвиток квантової криптографії	З розвитком квантових обчислень традиційні криптографічні алгоритми можуть стати вразливими. Розробка квантово-стійких алгоритмів для КЕП є

Продовження таблиці 3.13

		важливим напрямком для забезпечення довгострокової безпеки систем
3	Впровадження біометричних методів автентифікації	Поєднання КЕП з біометричними методами автентифікації, такими як сканування відбитків пальців або розпізнавання обличчя, може значно підвищити рівень безпеки та зручності використання систем
4	Розширення сфери застосування	КЕП може знайти застосування не лише в документах, але й у таких сферах, як інтернет речей (IoT), де необхідно забезпечувати автентичність даних, що передаються між пристроями
5	Підвищення доступності	Для широкого впровадження КЕП необхідно знизити вартість використання таких систем. Це може бути досягнуто за рахунок розвитку хмарних сервісів, які дозволять зменшити витрати на інфраструктуру

Впровадження систем, що використовують кваліфікований електронний підпис, є важливим кроком у розвитку цифрової економіки та суспільства. Однак цей процес супроводжується низкою проблем, пов’язаних із технічними, організаційними та правовими питаннями. Для успішного впровадження таких систем необхідно розвивати інфраструктуру, підвищувати обізнаність користувачів та гармонізувати законодавство. Перспективи розвитку КЕП пов’язані з інтеграцією нових технологій, таких як блокчейн та квантова криптографія, що дозволить забезпечити високий рівень безпеки та довіри до електронних документів у майбутньому.

Висновки до Розділу 3

Реалізація системи захисту даних з використанням кваліфікованого електронного підпису (КЕП) є важливим кроком у забезпеченні безпеки електронних

документів у сучасних умовах цифрової трансформації. Розробка та впровадження таких систем вимагають комплексного підходу, який включає не лише технічні напрямки, але й організаційні та правові компоненти. Система, що була розроблена, інтегрує основні функціональні модулі, такі як генерація ключів, підписання документів, перевірка підписів та управління даними, що забезпечує зручність використання та високу надійність. Підтримка різних алгоритмів кваліфікованого електронного підпису (КЕП) дозволяє адаптувати систему до різних вимог безпеки та продуктивності, що є важливим для забезпечення гнучкості та масштабованості.

Одним із ключових елементів системи є розробка та реалізація кастомного алгоритму КЕП. Цей процес вимагає глибокого розуміння криптографічних принципів, математичних основ та сучасних стандартів безпеки. Кастомний алгоритм повинен забезпечувати не лише високу криптографічну стійкість, але й ефективність у використанні ресурсів, що є критично важливим для систем із великою кількістю користувачів. Реалізація такого алгоритму дозволяє забезпечити автентичність, цілісність та неспростовність електронних документів, що є основним завданням системи захисту даних.

Тестування системи з використанням різних алгоритмів КЕП показало, що кожен із них має свої переваги та обмеження. Наприклад, RSA забезпечує високу безпеку, але вимагає значних обчислювальних ресурсів, що може бути обмеженням для систем із високим навантаженням. ECDSA та EdDSA демонструють високу швидкість та ефективність, що робить їх ідеальними для сучасних систем, які потребують швидкої обробки великих обсягів даних. DSA, хоча і поступається іншим алгоритмам у швидкості, залишається актуальним для систем, що вимагають дотримання державних стандартів. Вибір конкретного алгоритму повинен ґрунтуватися на вимогах до безпеки, продуктивності та сумісності системи, що підкреслює важливість гнучкості у розробці таких рішень.

Впровадження систем із використанням КЕП супроводжується низкою проблем, пов’язаних із технічними, організаційними та правовими складовими. Серед основних викликів можна виділити необхідність розвитку інфраструктури, підвищення обізнаності користувачів та гармонізацію законодавства. Для успішного

впровадження таких систем необхідно забезпечити їхню інтеграцію з існуючими інформаційними системами, а також розробити механізми для підвищення довіри користувачів до електронних документів. Перспективи розвитку КЕП пов’язані з інтеграцією нових технологій, таких як блокчейн та квантова криптографія, що дозволить забезпечити високий рівень безпеки та довіри до електронних документів у майбутньому.

Таким чином, розроблена система захисту даних з використанням КЕП є важливим інструментом для забезпечення безпеки електронних документів у сучасних умовах. Вона дозволяє ефективно вирішувати задачі автентичності, цілісності та неспростовності, що є критично важливим для забезпечення довіри до електронних документів. Подальший розвиток таких систем повинен бути спрямований на підвищення їхньої стійкості до нових викликів, таких як квантові обчислення, а також на інтеграцію з новими технологіями, що дозволить забезпечити високий рівень безпеки та ефективності у майбутньому.

ВИСНОВКИ

Розроблено систему захисту даних з використанням кваліфікованого електронного підпису (КЕП), яка інтегрує функції генерації, імпорту, підписання та перевірки документів. Система забезпечує ефективне управління ключами, зберігання підписаних документів та перевірку їх цілісності, що відповідає сучасним вимогам до захисту цифрової інформації. Реалізовані механізми дозволяють забезпечити автентичність, цілісність та неспростовність документів, що є ключовими елементами у сфері електронного документообігу.

Архітектура системи побудована з урахуванням необхідності інтеграції з різними платформами та стандартами, що робить її універсальною для використання в різних галузях, зокрема в бізнесі, державному управлінні та освіті. Використання сучасних криптографічних алгоритмів, таких як RSA, забезпечує високий рівень безпеки та відповідність міжнародним стандартам.

Реалізований інтерфейс користувача спрощує роботу з КЕП для неспеціалістів, що робить систему доступною для широкого кола користувачів. Інтуїтивно зрозумілий інтерфейс дозволяє ефективно керувати ключами, підписувати документи та перевіряти їхню цілісність без необхідності глибоких знань у сфері криптографії.

Проведено тестування системи, яке підтвердило її ефективність та надійність. Результати тестування демонструють, що система коректно виконує всі заявлені функції, включаючи генерацію ключів, підписання документів, перевірку підписів та зберігання даних. Достовірність результатів підтверджена відповідністю між очікуваними та фактичними результатами роботи системи.

Розроблена система має значну практичну цінність, оскільки може бути використана для автоматизації документообігу, підвищення ефективності бізнес-процесів та забезпечення безпеки цифрових транзакцій. Вона дозволяє скоротити витрати часу та ресурсів на обробку документів, забезпечуючи при цьому високий рівень захисту інформації.

Наукова новизна дослідження полягає в розробці механізмів, які інтегрують функції управління КЕП, підписання та перевірки документів у єдину систему. Це дозволяє забезпечити комплексний підхід до захисту цифрових даних, що є важливим кроком у розвитку технологій електронного підпису.

Рекомендовано продовжити розвиток системи шляхом додавання підтримки інших криптографічних алгоритмів, таких як ECDSA, що дозволить підвищити гнучкість та адаптивність системи до різних вимог користувачів. Також варто розглянути можливість інтеграції системи з існуючими платформами електронного документообігу для забезпечення її ширшого використання.

Результати дослідження можуть бути використані для подальшого вдосконалення методів роботи з КЕП, а також для розробки нових підходів до забезпечення безпеки цифрових даних. Система є важливим інструментом для підвищення рівня захисту інформації в умовах зростаючої цифровізації суспільства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Світлана Вороновська, Електронний цифровий підпис (ЕЦП) [Електронний ресурс]. – Режим доступу: https://bitfaktura.com.ua/blog/Elektronnyy-tsyfrovyy-pidpys-(ETSP)

2. Електронний Підпис [Електронний ресурс]. – Режим доступу: https://financer.com.ua/personalni-finansy/statti/cifroviy-pidpys/

3. Переваги ЕЦП [Електронний ресурс]. – Режим доступу: https://www.softcom.ua/ua/medoc/ecp/advantage.php

4. eIDAS [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/EIDAS

5. Електронний документообіг для ФОП [Електронний ресурс]. – Режим доступу: https://signy.online/elektronnij-dokumentoobig-dlya-fop/

6. ЗАКОН УКРАЇНИ Про внесення змін до деяких законодавчих актів України щодо забезпечення укладення угоди між Україною та Європейським Союзом про взаємне визнання кваліфікованих електронних довірчих послуг та імплементації законодавства Європейського Союзу у сфері електронної ідентифікації (Відомості Верховної Ради (ВВР), 2023, № 54, ст.159)

7. Алгоритм шифрування RSA, види атак на нього. Реалізація мовою Python [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/43026/

8. Digital Signature Algorithm (DSA) [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/digital-signature-algorithm-dsa/

9. Czym jest szyfrowanie ECDSA i czym różni się od szyfrowania RSA? [Електронний ресурс]. – Режим доступу: https://dhosting.pl/pomoc/baza-wiedzy/czym-jest-szyfrowanie-ecdsa-i-czym-rozni-sie-od-szyfrowania-rsa/

10. Кваліфікований надавач електронних довірчих послуг [Електронний ресурс]. – Режим доступу: https://uakey.com.ua/

11. Акредитований центр сертифікації ключів Призначений для систем електронного документообігу [Електронний ресурс]. – Режим доступу: https://acsk.privatbank.ua/main

12. docusign. [Електронний ресурс]. – Режим доступу: https://www.docusign.com/

13. Acrobat Sign Elektroniczne czy cyfrowe — korzystaj z bezpieczeństwa i elastyczności podpisów online [Електронний ресурс]. – Режим доступу: https://www.adobe.com/pl/sign/online-signature.html

14. GlobalSign [Електронний ресурс]. – Режим доступу: https://www.globalsign.com/en

15. Symantec [Електронний ресурс]. – Режим доступу: https://symantec-av.pl/home/?

16. Криптографія еліптичної кривої [Електронний ресурс]. – Режим доступу: https://www.gate.io/uk/blog/782/Elliptic-Curve-Cryptography

17. RSA Algorithm in Cryptography [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

18. What is ECDSA Encryption? How does it work? [Електронний ресурс]. – Режим доступу: https://www.encryptionconsulting.com/education-center/what-is-ecdsa/

19. Schneier, B. (2015). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, 5-7.

20. Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice*. Pearson, 322-324.

21. Katz, J., & Lindell, Y. (2020). *Introduction to Modern Cryptography*. CRC Press, 11-12.

22. Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 111-112.

23. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 112-115.

24. Sommerville, I. (2015). *Software Engineering*. Pearson – 816 с.

25. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 120-126.

26. National Institute of Standards and Technology. (2013). *FIPS PUB 180-4: Secure Hash Standard (SHS)*. NIST, 188–194.

27. Hankerson, D., Menezes, A., & Vanstone, S. (2004). *Guide to Elliptic Curve Cryptography*. Springer, 12-18.

28. Adams, C., & Lloyd, S. (2003). *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley, 35-37.

29. Dworkin, M. (2007). *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication 800-38B.

30. Microsoft Azure. (2023). *Azure Key Vault Documentation*. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/key-vault/.

31. Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 70.

32. Bernstein, D. J., & Lange, T. (2017). Post-Quantum Cryptography. *Nature*, 549(7671), 188–194.

33. Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson, 110–135.

34. Katz, J., & Lindell, Y. (2020). *Introduction to Modern Cryptography* (3rd ed.). CRC Press, 22–40.

35. NIST. (2015). *Recommendation for Key Management – Part 1: General* (NIST Special Publication 800-57). National Institute of Standards and Technology, 50–90.

36. Dang, Q. H. (2015). *Secure Hash Standard* (NIST FIPS PUB 180-4). National Institute of Standards and Technology, 180–200.

37. Python Cryptography Authors. (2023). *Cryptography: Python library documentation*. [Електронний ресурс]. – Режим доступу: https://cryptography.io/en/latest/

38. Bellare, M., & Rogaway, P. (2005). *The Security of Probabilistic Signature Schemes*. Journal of Cryptology, 18(3), 223–245.

39. Cooper, D., et al. (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* (RFC 5280). Internet Engineering Task Force, 12–15.

40. Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3* (RFC 8446). Internet Engineering Task Force, 203–225.

ДОДАТОК А
КОД ПРОГРАМИ

client.py:

```
import streamlit as st
import requests
import json
import base64
import os
import io
# Налаштування українською мовою інтерфейсу
st.set_page_config(
    page_title="Захист даних з використанням КЕП",
    page_icon="🔐",
)
# Константи
API_URL = "http://localhost:5000/api"
# Ініціалізація сховища сесії
if 'user_id' not in st.session_state:
    st.session_state.user_id = 1  # Для демонстрації використовуємо фіксований ID користувача
if 'current_kep' not in st.session_state:
    st.session_state.current_kep = None
# Функція для збереження КЕП у файл
def save_kep_to_file(kep_data, filename):
    with open(filename, 'w') as f:
        json.dump(kep_data, f)
    return True
# Функція для завантаження КЕП з файлу
def load_kep_from_file(file):
    content = file.read().decode('utf-8')
    return json.loads(content)
# Заголовок додатку
st.title("Захист даних з використанням КЕП")
# Бокова панель з навігацією
st.sidebar.title("Навігація")
page = st.sidebar.radio(
    "Перейти до:",
    ["Головна", "Керування КЕП", "Підписання документів", "Перевірка підписів", "Мої документи"]
)
# Головна сторінка
if page == "Головна":
    st.header("Захист даних з використанням кваліфікованого електронного підпису")
    st.write("""
    Ця система дозволяє:
    * Створювати та імпортувати кваліфіковані електронні підписи (КЕП)
    * Підписувати документи за допомогою КЕП
    * Перевіряти підписи документів
    * Зберігати та керувати підписаними документами
    Кваліфікований електронний підпис має повну юридичну силу та забезпечує:
    * Автентичність - підтвердження авторства документа
    * Цілісність - гарантія, що документ не був змінений після підписання
    * Неспростовність - підписувач не може заперечувати свій підпис
    """)
    st.info("Для початку роботи, будь ласка, створіть або імпортуйте КЕП в розділі 'Керування
КЕП'")
# Сторінка керування КЕП
elif page == "Керування КЕП":
    st.header("Керування кваліфікованим електронним підписом")
```

```
    # Вкладки для створення та імпорту КЕП
    tab1, tab2 = st.tabs(["Створення нового КЕП", "Імпорт існуючого КЕП"])
    # Вкладка створення нового КЕП
    with tab1:
        st.write("Створення нового кваліфікованого електронного підпису")
        if st.button("Згенерувати новий КЕП"):
            with st.spinner("Генерація ключів та сертифікату..."):
                response = requests.post(
                    f"{API_URL}/generate_key",
                    json={"user_id": st.session_state.user_id}
                )
                if response.status_code == 200:
                    data = response.json()
                    st.session_state.current_kep = data['kep_data']
                    st.success("КЕП успішно створено!")
                    # Кнопка для збереження КЕП у файл
                    kep_json = json.dumps(data['kep_data'])
                    st.download_button(
                        label="Зберегти КЕП у файл",
                        data=kep_json,
                        file_name="my_kep.json",
                        mime="application/json"
                    )
                else:
                    st.error(f"Помилка: {response.json().get('error', 'Невідома помилка')}")
    # Вкладка імпорту КЕП
    with tab2:
        st.write("Імпорт існуючого кваліфікованого електронного підпису")
        uploaded_file = st.file_uploader("Виберіть файл КЕП (JSON)", type=["json"])
        if uploaded_file is not None:
            try:
                kep_data = load_kep_from_file(uploaded_file)
                if st.button("Імпортувати КЕП"):
                    with st.spinner("Імпорт КЕП..."):
                        response = requests.post(
                            f"{API_URL}/import_key",
                            json={"user_id": st.session_state.user_id, "kep_data": kep_data}
                        )
                        if response.status_code == 200:
                            st.session_state.current_kep = kep_data
                            st.success("КЕП успішно імпортовано!")
                        else:
                            st.error(f"Помилка: {response.json().get('error', 'Невідома
помилка')}")
            except Exception as e:
                st.error(f"Помилка при імпорті КЕП: {str(e)}")
    # Відображення інформації про поточний КЕП
    st.subheader("Поточний КЕП")
    if st.session_state.current_kep:
        st.json({
            "Створено": st.session_state.current_kep['created'],
            "Термін дії": st.session_state.current_kep['expires'],
            "Користувач": st.session_state.current_kep['user_id']
        })
    else:
        st.info("КЕП не завантажено. Створіть новий або імпортуйте існуючий.")
# Сторінка підписання документів
elif page == "Підписання документів":
    st.header("Підписання документів за допомогою КЕП")
    if not st.session_state.current_kep:
        st.warning("Спочатку необхідно створити або імпортувати КЕП")
        st.button("Перейти до керування КЕП", on_click=lambda: st.session_state.update({"page":
"Керування КЕП"}))
    else:
        st.write("Завантажте документ для підписання за допомогою КЕП")
```

```
        uploaded_file = st.file_uploader("Виберіть файл для підписання", type=None)
        if uploaded_file is not None:
            file_details = {"Назва": uploaded_file.name, "Тип": uploaded_file.type, "Розмір":
uploaded_file.size}
            st.write(file_details)
            if st.button("Підписати документ"):
                with st.spinner("Підписання документа..."):
                    files = {'file': (uploaded_file.name, uploaded_file.getvalue())}
                    response = requests.post(
                        f"{API_URL}/sign_document",
                        files=files,
                        data={
                            'user_id': str(st.session_state.user_id),
                            'key_data': json.dumps(st.session_state.current_kep)
                        }
                    )
                    if response.status_code == 200:
                        data = response.json()
                        document_kep = data['document_kep']
                        st.success("Документ успішно підписано!")
                        # Кнопка для збереження КЕП документа
                        doc_kep_json = json.dumps(document_kep)
                        st.download_button(
                            label="Зберегти КЕП документа",
                            data=doc_kep_json,
                            file_name=f"{uploaded_file.name}.kep.json",
                            mime="application/json"
                        )
                    else:
                        st.error(f"Помилка: {response.json().get('error', 'Невідома помилка')}")
# Сторінка перевірки підписів
elif page == "Перевірка підписів":
    st.header("Перевірка підписаних документів")
    col1, col2 = st.columns(2)
    with col1:
        st.write("Завантажте документ для перевірки")
        doc_file = st.file_uploader("Виберіть підписаний документ", type=None)
    with col2:
        st.write("Завантажте файл КЕП документа")
        kep_file = st.file_uploader("Виберіть файл КЕП документа (.kep.json)", type=["json"])
    if doc_file is not None and kep_file is not None:
        try:
            document_kep = load_kep_from_file(kep_file)
            if st.button("Перевірити підпис"):
                with st.spinner("Перевірка підпису..."):
                    files = {'file': (doc_file.name, doc_file.getvalue())}
                    response = requests.post(
                        f"{API_URL}/verify_document",
                        files=files,
                        data={'document_kep': json.dumps(document_kep)}
                    )
                    if response.status_code == 200:
                        data = response.json()
                        st.subheader("Результат перевірки")
                        if data['verification_result']:
                            st.success(data['status_message'])
                        else:
                            st.error(data['status_message'])
                        st.info(f"Статус сертифіката: {data['certificate_status']}")
                        st.write(f"Сертифікат дійсний з: {data['certificate_valid_from']}")
                        st.write(f"Сертифікат дійсний до: {data['certificate_valid_until']}")
                    else:
                        st.error(f"Помилка: {response.json().get('error', 'Невідома помилка')}")
        except Exception as e:
            st.error(f"Помилка при читанні файлу КЕП: {str(e)}")
```

```
# Сторінка керування документами
elif page == "Мої документи":
    st.header("Мої підписані документи")
    if not st.session_state.user_id:
        st.warning("Для перегляду документів необхідно увійти в систему")
    else:
        with st.spinner("Завантаження списку документів..."):
            response = requests.get(f"{API_URL}/get_documents/{st.session_state.user_id}")
            if response.status_code == 200:
                documents = response.json()['documents']
                if not documents:
                    st.info("У вас ще немає підписаних документів")
                else:
                    st.subheader("Список документів")
                    for doc in documents:
                        col1, col2, col3 = st.columns([3, 2, 2])
                        with col1:
                            st.write(doc['filename'])
                        with col2:
                            st.write(doc['timestamp'])
                        with col3:
                            if st.button("Завантажити", key=f"download_{doc['id']}"):
                                doc_response =
requests.get(f"{API_URL}/get_document/{doc['id']}")
                                if doc_response.status_code == 200:
                                    doc_data = doc_response.json()
                                    # Декодування даних файлу
                                    file_data =
base64.b64decode(doc_data['document']['encrypted_data'])
                                    # Кнопка для скачування документа
                                    st.download_button(
                                        label="Зберегти документ",
                                        data=file_data,
                                        file_name=doc_data['document']['filename'],
                                        mime="application/octet-stream"
                                    )
                                    # Кнопка для скачування КЕП документа
                                    doc_kep_json = json.dumps(doc_data['document_kep'])
                                    st.download_button(
                                        label="Зберегти КЕП документа",
                                        data=doc_kep_json,
                                        file_name=f"{doc_data['document']['filename']}.kep.json",
                                        mime="application/json"
                                    )
            else:
                st.error("Не вдалося завантажити список документів")
# Підвал
st.markdown("---")
st.markdown("""
    <div style='text-align: center;'>
        <p>© 2025 Система захисту даних з використанням КЕП</p>
    </div>
""", unsafe_allow_html=True)
```

server.py:

```
from flask import Flask, request, jsonify
import sqlite3
import os
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.hazmat.primitives.asymmetric import padding, rsa
from cryptography.hazmat.backends import default_backend
from cryptography import x509
```

```
from cryptography.hazmat.primitives.serialization import load_pem_private_key
import base64
import datetime
import json
from flask_cors import CORS
app = Flask(__name__)
CORS(app)  # Для можливості робити запити з клієнта Streamlit
DATABASE = "kep_database.db"
# Ініціалізація бази даних
def init_db():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    # Створення таблиці користувачів
    c.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY,
        username TEXT UNIQUE NOT NULL,
        password_hash TEXT NOT NULL
    )
    ''')
    # Створення таблиці ключів КЕП
    c.execute('''
    CREATE TABLE IF NOT EXISTS keys (
        id INTEGER PRIMARY KEY,
        user_id INTEGER NOT NULL,
        public_key TEXT NOT NULL,
        certificate TEXT NOT NULL,
        creation_date TEXT NOT NULL,
        expiry_date TEXT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users (id)
    )
    ''')
    # Створення таблиці зашифрованих документів
    c.execute('''
    CREATE TABLE IF NOT EXISTS documents (
        id INTEGER PRIMARY KEY,
        user_id INTEGER NOT NULL,
        filename TEXT NOT NULL,
        encrypted_data BLOB,
        signature TEXT,
        timestamp TEXT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users (id)
    )
    ''')
    conn.commit()
    conn.close()
# Виклик ініціалізації при запуску
init_db()
# Генерація пари ключів та сертифіката КЕП
@app.route('/api/generate_key', methods=['POST'])
def generate_key():
    data = request.json
    user_id = data.get('user_id')
    if not user_id:
        return jsonify({"error": "Необхідно вказати ідентифікатор користувача"}), 400
    # Генерація приватного ключа
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    # Отримання публічного ключа
    public_key = private_key.public_key()
    # Створення самопідписаного сертифіката (для демонстрації)
    subject = x509.Name([
```

```
        x509.NameAttribute(x509.NameOID.COMMON_NAME, f'User {user_id}'),
    ])
    builder = x509.CertificateBuilder()
    builder = builder.subject_name(subject)
    builder = builder.issuer_name(subject)
    builder = builder.not_valid_before(datetime.datetime.utcnow())
    builder = builder.not_valid_after(datetime.datetime.utcnow() + datetime.timedelta(days=365))
    builder = builder.serial_number(x509.random_serial_number())
    builder = builder.public_key(public_key)
    # Додавання розширень для КЕП
    builder = builder.add_extension(
        x509.BasicConstraints(ca=False, path_length=None), critical=True,
    )
    certificate = builder.sign(
        private_key=private_key, algorithm=hashes.SHA256(),
        backend=default_backend()
    )
    # Серіалізація для збереження
    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    ).decode('utf-8')
    public_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    ).decode('utf-8')
    cert_pem = certificate.public_bytes(
        encoding=serialization.Encoding.PEM
    ).decode('utf-8')
    # Збереження в базу даних
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    now = datetime.datetime.utcnow().isoformat()
    expiry = (datetime.datetime.utcnow() + datetime.timedelta(days=365)).isoformat()
    c.execute('''
    INSERT INTO keys (user_id, public_key, certificate, creation_date, expiry_date)
    VALUES (?, ?, ?, ?, ?)
    ''', (user_id, public_pem, cert_pem, now, expiry))
    key_id = c.lastrowid
    conn.commit()
    conn.close()
    # Підготовка відповіді з даними для збереження КЕП (для перенесення)
    kep_data = {
        "private_key": private_pem,
        "certificate": cert_pem,
        "public_key": public_pem,
        "created": now,
        "expires": expiry,
        "user_id": user_id
    }
    return jsonify({
        "success": True,
        "key_id": key_id,
        "kep_data": kep_data
    })
# Імпорт існуючого КЕП з файлу
@app.route('/api/import_key', methods=['POST'])
def import_key():
    data = request.json
    user_id = data.get('user_id')
    kep_data = data.get('kep_data')
    if not user_id or not kep_data:
        return jsonify({"error": "Необхідно вказати ідентифікатор користувача та дані КЕП"}), 400
    try:
```

```
        # Збереження в базу даних
        conn = sqlite3.connect(DATABASE)
        c = conn.cursor()
        c.execute('''
        INSERT INTO keys (user_id, public_key, certificate, creation_date, expiry_date)
        VALUES (?, ?, ?, ?, ?)
        ''', (user_id, kep_data['public_key'], kep_data['certificate'],
              kep_data['created'], kep_data['expires']))
        key_id = c.lastrowid
        conn.commit()
        conn.close()
        return jsonify({
            "success": True,
            "key_id": key_id
        })
    except Exception as e:
        return jsonify({"error": f"Помилка імпорту КЕП: {str(e)}"}), 500
# Підписання документа з використанням КЕП
@app.route('/api/sign_document', methods=['POST'])
def sign_document():
    if 'file' not in request.files:
        return jsonify({"error": "Файл не знайдено"}), 400
    file = request.files['file']
    user_id = request.form.get('user_id')
    key_data = json.loads(request.form.get('key_data'))
    if not user_id or not key_data:
        return jsonify({"error": "Необхідно вказати ідентифікатор користувача та ключ КЕП"}), 400
    try:
        # Читання даних з файлу
        file_data = file.read()
        # Завантаження приватного ключа з даних КЕП
        private_key = load_pem_private_key(
            key_data['private_key'].encode('utf-8'),
            password=None,
            backend=default_backend()
        )
        # Створення підпису
        signature = private_key.sign(
            file_data,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        # Кодування підпису в base64 для збереження
        signature_b64 = base64.b64encode(signature).decode('utf-8')
        # Збереження документа та підпису в базу даних
        conn = sqlite3.connect(DATABASE)
        c = conn.cursor()
        now = datetime.datetime.utcnow().isoformat()
        c.execute('''
        INSERT INTO documents (user_id, filename, encrypted_data, signature, timestamp)
        VALUES (?, ?, ?, ?, ?)
        ''', (user_id, file.filename, file_data, signature_b64, now))
        document_id = c.lastrowid
        conn.commit()
        conn.close()
        # Створення даних КЕП документа для збереження (для перенесення)
        document_kep = {
            "document_id": document_id,
            "filename": file.filename,
            "signature": signature_b64,
            "certificate": key_data['certificate'],
            "public_key": key_data['public_key'],
```

```
            "timestamp": now
        }
        return jsonify({
            "success": True,
            "document_id": document_id,
            "document_kep": document_kep
        })
    except Exception as e:
        return jsonify({"error": f"Помилка підписання документа: {str(e)}"}), 500
# Перевірка підпису документа
@app.route('/api/verify_document', methods=['POST'])
def verify_document():
    if 'file' not in request.files:
        return jsonify({"error": "Файл не знайдено"}), 400
    file = request.files['file']
    document_kep = json.loads(request.form.get('document_kep'))
    if not document_kep:
        return jsonify({"error": "Необхідно вказати дані КЕП документа"}), 400
    try:
        # Читання даних з файлу
        file_data = file.read()
        # Завантаження публічного ключа з даних КЕП
        public_key = serialization.load_pem_public_key(
            document_kep['public_key'].encode('utf-8'),
            backend=default_backend()
        )
        # Декодування підпису з base64
        signature = base64.b64decode(document_kep['signature'])
        # Перевірка підпису
        try:
            public_key.verify(
                signature,
                file_data,
                padding.PSS(
                    mgf=padding.MGF1(hashes.SHA256()),
                    salt_length=padding.PSS.MAX_LENGTH
                ),
                hashes.SHA256()
            )
            verification_result = True
            status_message = "Підпис дійсний. Документ не був змінений після підписання."
        except Exception:
            verification_result = False
            status_message = "Підпис недійсний. Документ міг бути змінений після підписання."
        # Перевірка терміну дії сертифіката
        cert = x509.load_pem_x509_certificate(
            document_kep['certificate'].encode('utf-8'),
            backend=default_backend()
        )
        now = datetime.datetime.utcnow()
        if now < cert.not_valid_before:
            cert_status = "Сертифікат ще не дійсний"
            verification_result = False
        elif now > cert.not_valid_after:
            cert_status = "Термін дії сертифіката закінчився"
            verification_result = False
        else:
            cert_status = "Сертифікат дійсний"
        return jsonify({
            "success": True,
            "verification_result": verification_result,
            "status_message": status_message,
            "certificate_status": cert_status,
            "certificate_valid_from": cert.not_valid_before.isoformat(),
            "certificate_valid_until": cert.not_valid_after.isoformat()
```

```
        })
    except Exception as e:
        return jsonify({"error": f"Помилка перевірки підпису: {str(e)}"}), 500
# Отримання списку документів користувача
@app.route('/api/get_documents/<int:user_id>', methods=['GET'])
def get_documents(user_id):
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    c = conn.cursor()
    c.execute('''
    SELECT id, filename, timestamp
    FROM documents
    WHERE user_id = ?
    ORDER BY timestamp DESC
    ''', (user_id,))
    documents = [dict(row) for row in c.fetchall()]
    conn.close()
    return jsonify({
        "success": True,
        "documents": documents
    })
# Отримання даних документа
@app.route('/api/get_document/<int:document_id>', methods=['GET'])
def get_document(document_id):
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    c = conn.cursor()
    c.execute('''
    SELECT id, user_id, filename, encrypted_data, signature, timestamp
    FROM documents
    WHERE id = ?
    ''', (document_id,))
    document = dict(c.fetchone())
    # Отримання сертифіката
    c.execute('''
    SELECT certificate, public_key
    FROM keys
    WHERE user_id = ?
    LIMIT 1
    ''', (document['user_id'],))
    key_data = dict(c.fetchone())
    conn.close()
    # Перетворення бінарних даних для передачі
    document['encrypted_data'] = base64.b64encode(document['encrypted_data']).decode('utf-8')
    # Створення документа КЕП для перевірки
    document_kep = {
        "document_id": document['id'],
        "filename": document['filename'],
        "signature": document['signature'],
        "certificate": key_data['certificate'],
        "public_key": key_data['public_key'],
        "timestamp": document['timestamp']
    }
    return jsonify({
        "success": True,
        "document": document,
        "document_kep": document_kep
    })
 Запуск сервера
if __name__ == "__main__":
    app.run(debug=False)
```

ДОДАТОК Б
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ РОБОТИ

Тези наукових доповідей:

1. Serhii Toliupa, Anatoliy Lininchuk. DATA PROTECTION BASED ON AN ALGORITHM WITH PUBLIC KEY ALGORITHM. Міжнародна науково-практична конференція “Інформаційні технології та впровадження” (Information technology and implementations) (IT&I-2024). С. 102-103.