Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Ткачука Богдана Григоровича

на здобуття ступеня вищої освіти Бакалавра

Система захисту персональних даних на основі Blockchain технологій

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

Шифр КРБКБ.2101118.21.01.08 ПЗ

Виконав студент 4 курсу група КБ-21-1 Богдан ТКАЧУК

Керівник канд. техн. наук, доцент ст. Микола СТЕЦЮК

Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

10 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ткачуку Богдану Григоровичу

1 Тема Система захисту персональних даних на основі Blockchain технологій

Керівник роботи Микола Стецюк

Затверджено наказом ректора університету від 07 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 03 червня 2025 № 23

3 Вихідні дані до роботи Закон України "Про захист персональних даних": Це основний закон, що регулює збір, зберігання, обробку та захист персональних даних в Україні. Смарт-контракти як основа управління правами доступу. Криптографічні методи: AES (шифрування даних), RSA або ECC (захист ключів), SHA-256 (хешування)

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз загроз для систем персональних даних архітектура та вразливості систем класифікація атак методи запобігання загрозам розробка системи захисту на основі блокчейну архітектура з гібридним шифруванням та аудитом доступу модуль автентифікації та контролю доступу визначення політик безпеки реакція на загрози реалізація прототипу вибір середовища розробки архітектура ПЗ реалізація шифрування та смартконтрактів тестування системи.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Копія графічної частини код додатку опрацювання.

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та *посада консультанта*	Підпис, дата	
		завдання видав	*завдання прийняв*

7 Дата видачі завдання 16 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Лютий	
Ознайомлення з предметною областю	Лютий	
Дослідження існуючих рішень	Лютий	
Постановка задачі	Лютий	
Визначення загальних принципів рішення задачі	Лютий	
Деталізація принципів рішення задачі	Березень	
Впровадження безпеки персональних даних за допомогою blockchain технологій	Березень	
Апробація проєктних рішень	Березень	
Оформлення пояснювальної записки згідно вимог	Травень	
Оформлення графічної частини	Травень	
Захист КР	Червень	

Студент _____ Богдан ТКАЧУК

Керівник кваліфікаційної роботи _____ Микола СТЕЦЮК

АНОТАЦІЯ

Тема кваліфікаційної роботи: Система захисту персональних даних на основі Blockchain технологій

Автор роботи: Ткачук Богдан Григорович.

Керівник роботи: Стецюк Микола Васильович.

Пояснювальна записка: 64 с., 3 додатки, 8 рисунків, 1 таблиці, 40 джерел.

Графічна частина: 3 плакати, 10 презентаційних слайдів.

КОМЛЕКСНА СИСТЕМА ЗАХИСТУ ІНФОРМАЦІЇ, АВТОМАТИЗОВАНЕ РОБОЧЕ МІСЦЕ КЕРІВНИКА, ТЕХНІЧНЕ ЗАВДАННЯ, ПЛАН ЗАХИСТУ ІНФОРМАЦІЇ, АКТ.

Кваліфікаційна робота бакалавра присвячена розробці системи захисту персональних даних на основі блокчейн-технологій. У роботі проведено ґрунтовний аналіз сучасного стану захисту персональних даних, з виявленням основних вразливостей та потенційних загроз, пов'язаних із їх витоком, несанкціонованим доступом та маніпуляціями.. На основі отриманих результатів сформовано комплексну систему захисту, що базується на використанні технології блокчейн. У роботі представлено розробку та впровадження механізмів децентралізованого зберігання, криптографічного шифрування, смарт-контрактів для управління доступом та алгоритмів консенсусу, що забезпечують незмінність і безпеку даних. Також проведено тестування запропонованої системи на відповідність критеріям безпеки та продуктивності. Результатом роботи є створення надійного інструменту для збереження та захисту персональних даних, що забезпечує їх конфіденційність, цілісність та доступність, а також знижує ризики кіберзагроз завдяки децентралізованій архітектурі блокчейну.

07.06.2025

ABSTRACT

Subject of qualification work Personal data protection system based on Blockchain technologies

Author of the work: Tkachuk Bohdan Grigorovich.

Head of work: Stetsyuk Mykola Vasilyovich.

Explanatory note: 64 p., 3 appendices, 8 figures, 1 tables, 40 sources. Graphic part: 3 posters, 10 presentation slides.

COMPREHENSIVE INFORMATION PROTECTION SYSTEM, AUTOMATED WORKPLACE OF THE MANAGER, TECHNICAL TERMS OF REFERENCE, INFORMATION PROTECTION PLAN, ACT.

The bachelor's qualification thesis is devoted to the development of a personal data protection system based on blockchain technology. The work includes a thorough analysis of the current state of personal data protection, identifying key vulnerabilities and potential threats related to data leakage, unauthorized access, and manipulation. The specific risks associated with centralized data storage systems are outlined, as well as the possible consequences of data compromise for users and organizations. Based on the findings, a comprehensive protection system was designed using blockchain technology. The thesis presents the development and implementation of mechanisms for decentralized data storage, cryptographic encryption, smart contract-based access control, and consensus algorithms that ensure the immutability and security of data. In addition, accompanying documentation has been prepared, including a system implementation plan, technical specification, project documentation, incident response plans, risk assessment, and compliance analysis with relevant regulations. The proposed system has also been tested for compliance with security and performance criteria. The result of the work is a reliable tool for storing and protecting personal data, ensuring confidentiality, integrity, and availability, while reducing cybersecurity risks through a decentralized blockchain-based architecture.

07.06.2025 _____

ЗМІСТ

ВСТУП

У сучасному цифровому світі питання захисту персональних даних набуває дедалі більшої актуальності. З кожним роком обсяг інформації, що обробляється в електронному вигляді, стрімко зростає. Державні структури, приватні компанії, фінансові установи та соціальні платформи щоденно накопичують і обробляють великі масиви персональної інформації громадян. У зв’язку з цим виникає нагальна потреба у створенні надійних систем, здатних гарантувати безпеку таких даних від несанкціонованого доступу, втрати, модифікації чи витоку.

Водночас зі збільшенням обсягу цифрової інформації зростає і рівень загроз, які спрямовані на отримання чи пошкодження персональних даних. Серед найбільш поширених можна виділити фішинг-атаки, несанкціонований доступ, експлуатацію вразливостей програмного забезпечення.

Одним із найперспективніших рішень у цій сфері є застосування блокчейн-технологій. Вони поєднують у собі принципи децентралізації, криптографії, прозорості та незмінності даних, що робить їх потужним інструментом у боротьбі з кіберзагрозами.

Метою даної кваліфікаційної роботи є дослідження можливостей блокчейн-технологій для побудови системи захисту персональних даних. У межах дослідження буде проаналізовано сучасні загрози інформаційній безпеці, розглянуто переваги та особливості використання розподілених реєстрів, а також розроблено модель захисту даних, яка базується на блокчейні.

Об'єктом дослідження виступають інформаційні системи, що обробляють персональні дані. Предметом дослідження є методи та інструменти забезпечення конфіденційності, цілісності й доступності персональної інформації з використанням блокчейн-технологій.

Практична значущість роботи полягає у можливості впровадження отриманих результатів у реальні інформаційні системи для підвищеннярівня їх безпеки. Запропонований підхід дозволяє значно зменшити ризики витоку інформації, несанкціонованого доступу та інших загроз.

1 АНАЛІЗ ЗАГРОЗ ТА МЕТОДИ ПРОТИДІЇ ЗАГРОЗАМ

1.1 Основні загрози для персональних даних

У цифровому середовищі персональні дані стали однією з найцінніших категорій інформації. Вони охоплюють ідентифікаційні відомості про особу, зокрема ім’я, дату народження, адресу, номер телефону, фінансову інформацію, медичні записи та інші чутливі дані.

Втрата або несанкціонований доступ до такої інформації може призвести до фінансових втрат, репутаційних збитків, порушення прав людини, а також загрожує безпеці держави у випадку масштабних витоків. Тому забезпечення надійного захисту персональних даних є одним з основних викликів у сфері інформаційної безпеки.

Загрози для персональних даних поділяються на технічні, організаційні та соціальні. Технічні загрози пов’язані з використанням шкідливого ПЗ, вразливостей у програмному забезпеченні, недоліками мережевої інфраструктури. Організаційні – із неналежною політикою безпеки, відсутністю регулярного аудиту, поганою підготовкою персоналу. Соціальні загрози мають справу із людським фактором – довірливістю користувачів, маніпуляціями, шахрайством.

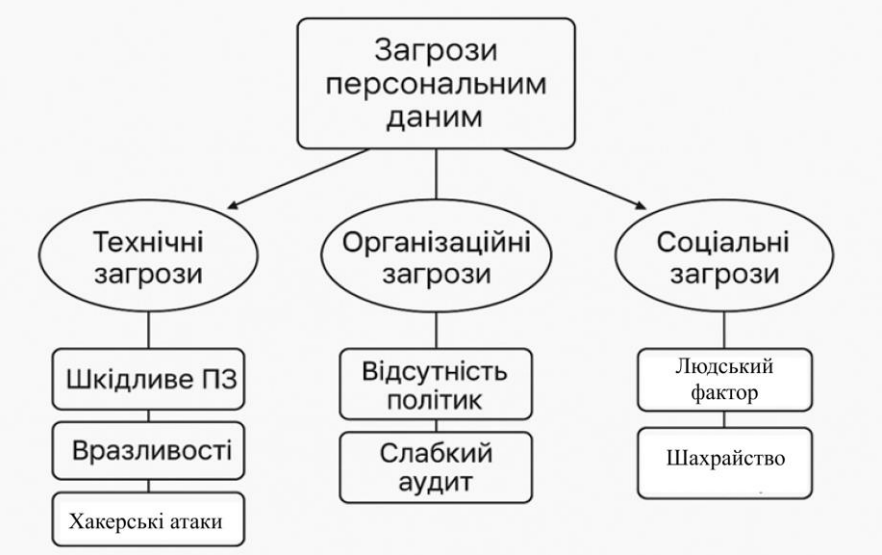


Рисунок 1.1 – Класифікація загроз для персональних даних

Несанкціонований доступ до даних. Це одна з найпоширеніших загроз, при якій хакери або зловмисники з числа співробітників отримують доступ до інформації без відповідного дозволу. Такий доступ часто досягається за допомогою експлуатації вразливостей у ПЗ, застосування шкідливого коду або підбору паролів. Прикладом є атаки на сервери медичних закладів, де зловмисники викрадають особисту інформацію пацієнтів для подальшого продажу.

Фішинг – це метод соціальної інженерії, при якому користувача вводять в оману з метою отримання конфіденційних даних (логінів, паролів, банківської інформації). Найчастіше зловмисники створюють копії популярних сайтів (наприклад, банківських порталів або систем електронної пошти) і спонукають користувача ввести свої дані. Згідно зі звітами, фішинг залишається найефективнішим методом атак у корпоративному середовищі.

Малваре та віруси. Ці види шкідливого ПЗ можуть шифрувати, пошкоджувати або копіювати інформацію з пристроїв жертви. Відомі різновиди малваре включають трояни, які маскуються під легітимні програми, та програми-шпигуни, які передають дані з комп’ютера жертви на сервер зловмисника. Особливу загрозу становить програма-вимагач (ransomware), яка блокує доступ до даних до моменту сплати викупу.

Витік інформації через сторонні сервіси. Нерідко компанії використовують зовнішні платформи для зберігання, аналізу чи резервного копіювання даних. У разі порушення безпеки цих сервісів персональні дані можуть стати доступними стороннім особам. Це особливо небезпечно при використанні послуг, що не відповідають міжнародним стандартам безпеки.

Фізичне викрадення пристроїв. Смартфони, ноутбуки, флеш-накопичувачі, які не зашифровані або не захищені паролями, стають джерелом потенційного витоку інформації. Пристрої, залишені без нагляду в публічних місцях, можуть бути викрадені, після чого дані, які на них зберігаються, потрапляють до рук зловмисників.

Людський фактор. Багато інцидентів трапляються не через злий умисел, а внаслідок помилок персоналу – неправильна конфігурація систем, відкритий доступ до файлів, слабкі паролі, нехтування політиками безпеки. Навіть добре захищена система може постраждати через недбалість або необізнаність користувачів.

Недостатній рівень безпеки при зберіганні та передачі даних. У багатьох випадках персональні дані зберігаються у відкритому вигляді або передаються незашифрованими каналами. Це робить можливим їх перехоплення або несанкціонований доступ.

Відсутність оновлень безпеки. Системи, які довго не оновлювались, часто містять вразливості, що вже відомі хакерам. Уразливе ПЗ може бути легко використане для проникнення у мережу або зламу системи управління персональними даними.

Статистика підтверджує серйозність ситуації. Згідно з дослідженням IBM Cost of a Data Breach Report 2023, середня вартість одного витоку персональних даних становила 4.45 мільйона доларів США. Компанії стикаються не лише з фінансовими втратами, а й з втратами довіри клієнтів, юридичними наслідками та регуляторними санкціями.

У контексті українського законодавства захист персональних даних регулюється Законом України «Про захист персональних даних», який вимагає від власників та розпорядників інформації вживати необхідних організаційних і технічних заходів для її безпеки. Невиконання цих вимог тягне за собою адміністративну та кримінальну відповідальність.

Технічні заходи використання сучасного антивірусного програмного забезпечення для виявлення та блокування шкідливого ПЗ. Регулярне оновлення операційних систем і програмного забезпечення для усунення відомих вразливостей. Шифрування даних як під час їхнього зберігання, так і передачі. Застосування багатофакторної аутентифікації (MFA) для доступу до систем і сервісів. Використання брандмауерів та систем виявлення вторгнень (IDS/IPS).

Організаційні заходи розробка та впровадження політик інформаційної безпеки. Проведення регулярного аудиту інформаційної інфраструктури. Навчання співробітників основам кібербезпеки. Впровадження процедур реагування на інциденти безпеки. Обмеження доступу до конфіденційної інформації відповідно до ролей користувачів (принцип мінімальних привілеїв).

Соціальні заходи проведення інформаційних кампаній для підвищення обізнаності користувачів щодо загроз фішингу та шахрайства. Створення культури відповідального ставлення до персональних даних серед співробітників. Регулярне тестування працівників на здатність розпізнавати соціальні атаки (наприклад, фішингові листи).

Таким чином, для ефективного забезпечення безпеки персональних даних необхідне комплексне бачення загроз, врахування як зовнішніх, так і внутрішніх факторів, регулярний аналіз вразливостей та впровадження найкращих практик захисту.

1.2 Методи атак на інформаційні системи

Зі зростанням значущості інформаційних систем зростає і кількість загроз, спрямованих на їх порушення. Методи атак на інформаційні системи постійно вдосконалюються, стаючи дедалі складнішими та небезпечнішими. У цьому розділі розглянемо основні методи атак, які використовуються для порушення цілісності, конфіденційності та доступності інформації. Особливу увагу приділено загрозам, що стосуються обробки персональних даних, які є однією з ключових цілей для кіберзлочинців. Атаки можуть бути спрямовані як на технічні компоненти системи — бази даних, сервіси доступу, канали зв’язку, — так і на організаційні аспекти, зокрема соціальну інженерію, компрометацію прав доступу або внутрішні зловживання. У контексті захисту персональних даних, особливо актуальними є атаки типу «людина посередині» (MITM), фішинг, шкідливе ПЗ, атак на API, викрадення токенів доступу та порушення криптографічних

механізмів. У роботі також розглядаються специфічні сценарії, де традиційні заходи безпеки виявляються недостатніми, а застосування блокчейн-технологій, гібридного шифрування, багаторівневої автентифікації та смартконтрактів дозволяє суттєво підвищити стійкість системи до таких атак.

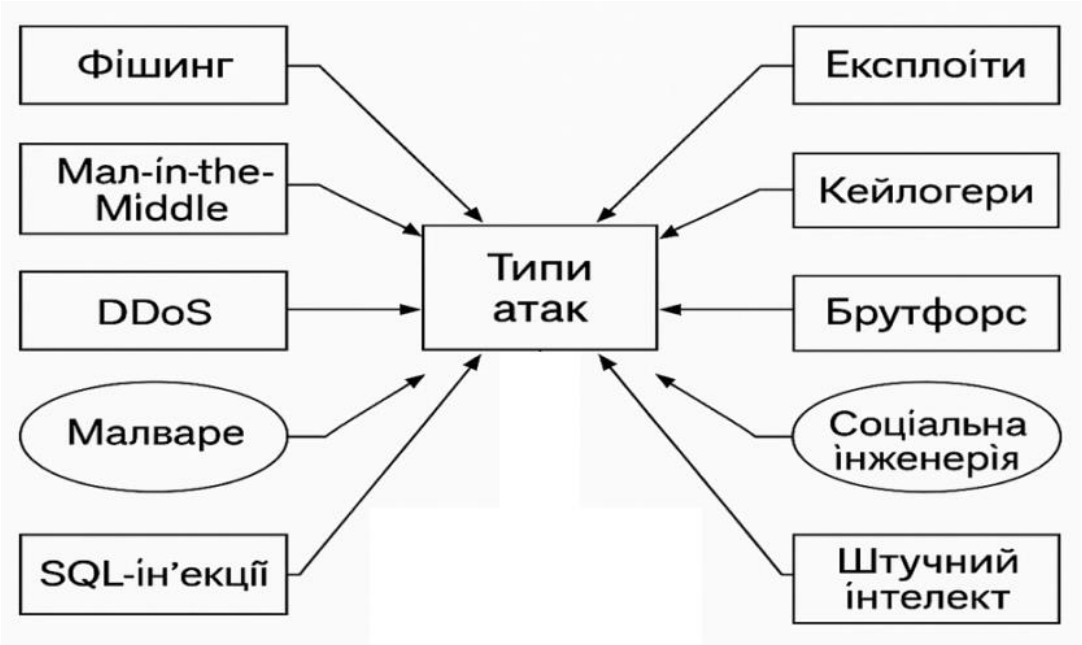


Рисунок 1.2 – Методи атак на інформаційні системи

Одним із найпоширеніших методів атак є фішинг. Це техніка, яка спрямована на отримання конфіденційної інформації, такої як паролі, дані банківських карток або інші особисті дані. Зловмисники використовують підроблені електронні листи, вебсайти або повідомлення, які виглядають як офіційні запити від відомих організацій. Жертва, введена в оману, добровільно надає свої дані, що дозволяє зловмисникам отримати доступ до її облікових записів або фінансів.

Ще одним поширеним методом є атака типу "людина посередині" (Man-in-the-Middle). У цьому випадку зловмисник перехоплює комунікацію між двома сторонами, що дозволяє йому отримати доступ до передаваних даних або навіть змінити їх. Такий метод часто використовується в незахищених мережах Wi-Fi, де дані передаються без належного шифрування.

DDoS-атаки (Distributed Denial of Service) є ще одним популярним методом атак на інформаційні системи. Їх мета полягає у перевантаженні сервера або мережі великою кількістю запитів, що робить систему недоступною для легітимних користувачів. Для реалізації таких атак зловмисники часто використовують ботнети – мережі заражених пристроїв, які виконують команди хакерів.

Одним із серйозних викликів для інформаційної безпеки є використання шкідливого програмного забезпечення. Зловмисники створюють віруси, трояни, хробаки та інші програми, які проникають у систему жертви для виконання різноманітних дій крадіжки даних, шифрування файлів з метою вимагання викупу або навіть повного знищення інформації. Наприклад, програми-шифрувальники (ransomware) стали особливо поширеними останнім часом і завдають значної шкоди як організаціям, так і окремим користувачам.

SQL-ін'єкції є ще одним методом атак, який використовується для отримання несанкціонованого доступу до баз даних. Зловмисники вводять шкідливий код у запити до бази даних через веб-форми або URL-адреси, що дозволяє їм отримати доступ до конфіденційної інформації або навіть змінити її.

Не менш небезпечними є атаки на основі експлойтів – програмних вразливостей у системах або додатках. Зловмисники знаходять недоліки в програмному забезпеченні та використовують їх для проникнення в систему. У таких випадках важливу роль відіграє своєчасне оновлення програмного забезпечення та встановлення патчів безпеки.

Методи атак на інформаційні системи також включають перехоплення паролів шляхом використання кейлогерів або атак грубою силою (brute force). Кейлогери – це програми або пристрої, які фіксують усі натискання клавіш на клавіатурі, дозволяючи зловмисникам дізнатися введені паролі. Атаки грубою силою полягають у підборі паролів шляхом перебору всіх можливих комбінацій.

Окремо варто згадати про атаки на основі соціальної інженерії, які спрямовані не на технологічні аспекти захисту системи, а на людський фактор. Зловмисники використовують обман і маніпуляції для отримання необхідної

інформації від співробітників організації або окремих користувачів. Наприклад, вони можуть видавати себе за технічну підтримку та просити надати паролі для вирішення "технічних проблем".

Також зростає кількість атак із використанням штучного інтелекту та машинного навчання. Зловмисники застосовують ці технології для автоматизації атак, аналізу великих обсягів даних та створення більш складних і персоналізованих фішингових кампаній.

Отже, методи атак на інформаційні системи є різноманітними та постійно еволюціонують разом із розвитком технологій. Для забезпечення ефективного захисту важливо розуміти ці методи, аналізувати потенційні загрози та впроваджувати комплексні заходи безпеки, які включають технологічні рішення, навчання персоналу та регулярний моніторинг систем.

1.3 Види шифрування та їх застосування для захисту файлів

У цифровому середовищі, де обсяг обробки та зберігання персональних даних стрімко зростає, питання їхнього захисту набуває особливої важливості. Кількість кіберзагроз постійно збільшується, методи атак ускладнюються, а масштабні витоки інформації змушують організації впроваджувати комплексні заходи інформаційної безпеки. Ефективні методи захисту охоплюють як технічні, так і організаційні аспекти, формуючи багаторівневу систему, здатну протистояти сучасним загрозам.

Одним із ключових підходів до забезпечення конфіденційності є шифрування. Цей метод унеможливлює прочитання інформації сторонніми особами у разі несанкціонованого доступу. Використовуються як симетричні алгоритми (зокрема AES), так і асиметричні (наприклад, RSA, ECC). Симетричне шифрування забезпечує високу швидкість обробки, однак вимагає безпечного обміну ключами. Асиметричне, навпаки, зручніше у розповсюдженні ключів,

проте потребує більше ресурсів. Впровадження надійної криптографії істотно знижує ризики витоку інформації навіть за умов часткового порушення безпеки.

Ще одним важливим компонентом системи захисту є аутентифікація та авторизація користувачів. Аутентифікація підтверджує особу користувача, тоді як авторизація визначає, до яких ресурсів він має доступ. Сучасні рішення дедалі частіше застосовують багатофакторну автентифікацію, що поєднує паролі, апаратні токени, біометричні характеристики. Це значно ускладнює процес несанкціонованого доступу, навіть у випадку компрометації одного з факторів. У деяких випадках аутентифікація ґрунтується на поведінкових характеристиках користувача, таких як динаміка введення тексту, що створює додатковий рівень захисту

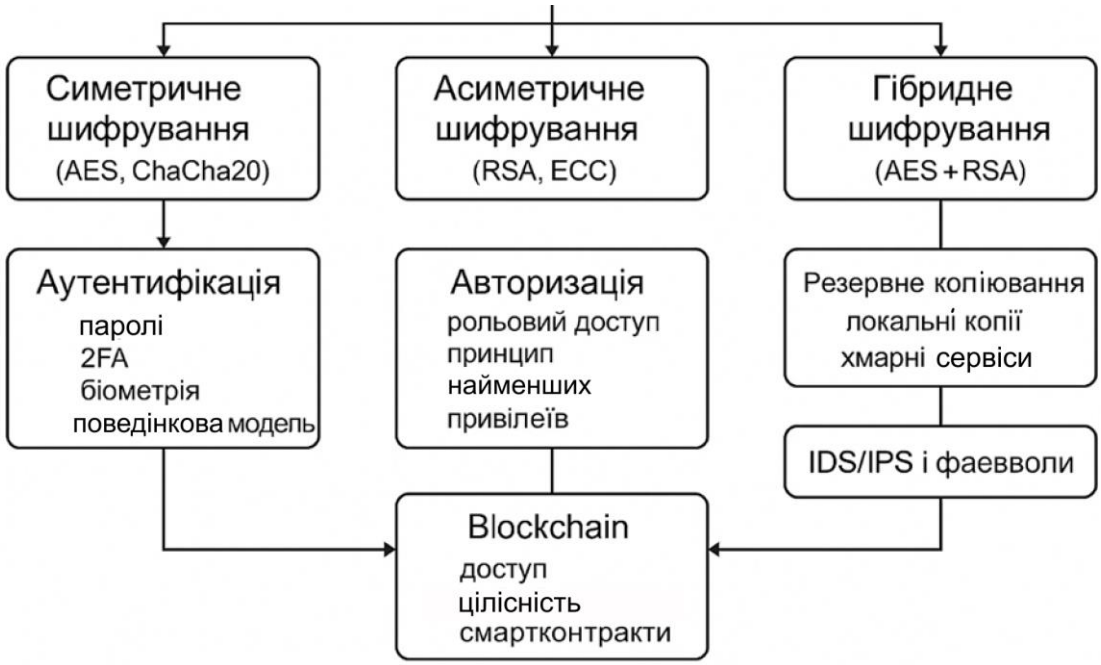


Рисунок 1.3 – види типів шифрування

Контроль доступу є ще однією критично важливою складовою безпеки інформаційних систем. Використання рольових моделей доступу дозволяє точно визначити повноваження кожного працівника відповідно до його функціональних обов’язків. Це дає змогу уникнути ситуацій, коли користувач має надмірні права, що потенційно призводить до порушення конфіденційності чи цілісності даних.

Дотримання принципу найменших привілеїв (least privilege) є дієвим інструментом зменшення ризиків як зовнішніх, так і внутрішніх загроз.

Щоб забезпечити безперервну роботу систем, необхідно гарантувати цілісність та доступність даних. Для цього використовують резервне копіювання, засоби моніторингу та рішення для аварійного відновлення. Регулярне створення резервних копій дозволяє відновити втрачену інформацію, наприклад, після атаки типу ransomware, коли дані шифруються зловмисниками для вимагання викупу. Крім того, все більше організацій впроваджують системи постійного моніторингу IT-інфраструктури, здатні виявляти аномалії ще до того, як ті стануть причиною серйозних проблем.

Суттєве значення мають також міжмережеві екрани (фаєрволи) та системи виявлення й запобігання вторгненням (IDS/IPS). Вони здійснюють аналіз вхідного й вихідного трафіку, виявляють підозрілу активність і блокують загрози в режимі реального часу. Ефективність цих систем базується на поєднанні сигнатурного та евристичного аналізу, що дозволяє виявляти як відомі, так і нові типи атак. Сучасні системи, оснащені технологіями машинного навчання, здатні самостійно адаптуватися до змін у поведінці зловмисників.

Останніми роками зростає інтерес до блокчейн-технологій як інноваційного інструменту у сфері захисту даних. Блокчейн – це децентралізований розподілений реєстр, в якому дані зберігаються у вигляді зв’язаних між собою криптографічно захищених блоків. Така структура забезпечує незмінність інформації , будь-яка спроба модифікації вмісту одного блоку порушує хеш-ланцюг і відразу фіксується системою. Це унеможливлює несанкціоноване редагування або підміну даних.

Однією з найважливіших переваг блокчейну є децентралізація, завдяки якій відсутня єдина точка відмови. Інформація дублюється на багатьох вузлах мережі, що підвищує її доступність і знижує вразливість до атак. Навіть якщо один з вузлів зазнає компрометації, загальна система продовжує функціонувати без втрати даних. Крім того, прозорість блокчейну дозволяє усім учасникам перевіряти транзакції, виключаючи можливість прихованих маніпуляцій.

Окрему роль відіграють смарт-контракти – програмні алгоритми, які автоматично виконують задані умови. У сфері захисту інформації їх можна використовувати для контролю доступу, ведення журналів подій, перевірки дозволів. Наприклад, у медичних системах смарт-контракт може обмежувати доступ до електронної медичної карти лише для уповноваженого лікаря і лише на визначений час. При цьому всі звернення до даних фіксуються у незмінному журналі.

Блокчейн-технології знаходять застосування не лише у фінансовому секторі, але й у сфері охорони здоров'я, освіти, державного управління. Вони не тільки підвищують рівень безпеки, але й зміцнюють довіру користувачів до цифрових сервісів. У сучасних умовах стрімкої цифровізації інтеграція блокчейну у системи інформаційної безпеки відкриває можливості створення надійних, масштабованих і прозорих рішень. Зокрема, блокчейн може використовуватись для зберігання логів доступу, перевірки цифрових підписів, розповсюдження публічних ключів.

Отже, ефективний захист персональних даних передбачає поєднання традиційних та інноваційних методів. Шифрування, багатофакторна автентифікація, контроль доступу, фаєрволи – усе це формує базис безпеки. Інтеграція блокчейн-технологій як сучасного інструменту дозволяє не лише підвищити надійність захисту, але й полегшити аудит, спростити управління доступом та зменшити вплив людського фактору.

1.4 Постановка задачі

У умовах стрімкого розвитку цифрових технологій питання захисту персональних даних набуло особливої актуальності. Різноманітні сфери діяльності від державного управління до комерційного сектору активно впроваджують електронні сервіси, які передбачають обробку, зберігання та передачу конфіденційної інформації.

Разом із цим збільшується кількість кіберзагроз, інцидентів витоку даних та фактів несанкціонованого доступу. Традиційні методи захисту, попри їх ефективність, дедалі частіше виявляються недостатніми для протидії новітнім загрозам, зокрема тим, що базуються на соціальній інженерії, уразливостях нульового дня або складних комбінованих атаках.

Одним із перспективних напрямів підвищення рівня інформаційної безпеки є використання блокчейн-технологій. Завдяки своїм властивостям децентралізації, незмінності записів, прозорості та криптографічному захисту блокчейн пропонує нові підходи до забезпечення захисту персональних даних. Проте інтеграція цієї технології в існуючі системи захисту потребує чітко визначеної концепції, аналізу доцільності застосування, а також оцінки її ефективності в умовах конкретного підприємства чи організації.

У зв’язку з цим постає необхідність у формалізації задачі побудови системи захисту персональних даних, яка базуватиметься на блокчейн-технології. Така система має враховувати реальні загрози інформаційній безпеці, відповідати сучасним стандартам захисту даних і бути здатною до масштабування та інтеграції з наявною IT-інфраструктурою.

Таким чином, основною метою дослідження є розробка концепції побудови системи захисту персональних даних із використанням технологій блокчейн, яка забезпечувала б Конфіденційність унеможливлення несанкціонованого доступу до персональних даних. Цілісність гарантію незмінності даних у процесі зберігання та передачі. Доступність забезпечення доступу до даних лише уповноваженим особам, у визначених межах. Прозорість та контроль можливість ведення незмінного журналу подій доступу до інформації;

Захист від внутрішніх загроз зменшення впливу людського фактору та потенційних зловживань з боку персоналу. Масштабованість та інтеграційність здатність системи до розширення та адаптації в умовах змінних потреб організації.

Для досягнення поставленої мети необхідно вирішити такі основні завдання.

Провести аналіз існуючих методів захисту персональних даних і визначити їх недоліки в контексті сучасних загроз. Вивчити принципи функціонування блокчейн-технологій і виявити потенційні переваги їх використання у сфері інформаційної безпеки. Розробити архітектурну модель системи захисту персональних даних, яка базується на блокчейні. Розробити модуль контролю обігу документів із застосуванням IPFS, цифрових підписів та смарт-контрактного аудиту для забезпечення прозорості та недоторканності електронної документації. Запропонувати механізми контролю доступу, ведення журналу подій та перевірки достовірності даних на основі смарт-контрактів. Оцінити ефективність запропонованої системи шляхом моделювання або апробації у тестовому середовищі. Виконання зазначених завдань дозволить побудувати надійну систему захисту, здатну протидіяти актуальним загрозам, забезпечуючи при цьому високий рівень довіри користувачів до систем обробки персональних даних

2 ПРОЄКТУВАННЯ СИСТЕМИ ЗАХИСТУ ПЕРСОНАЛЬНИХ ДАНИХ

2.1 Архітектура системи

У процесі проєктування системи захисту персональних даних важливо не лише забезпечити відповідність актуальним стандартам безпеки, а й досягти технологічної ефективності та сумісності з існуючими рішеннями. Саме тому вибір криптографічних алгоритмів та інфраструктурних компонентів базувався на комплексному аналізі функціональних потреб системи, реальних загроз та практичних можливостей їх усунення.

Вибір блокчейн-платформи, Hyperledger Fabric. Одним із перших кроків стала оцінка блокчейн-платформ для реалізації механізмів реєстрації, контролю доступу та збереження журналів подій. Було розглянуто Ethereum, Quorum, Corda та Hyperledger Fabric.

Hyperledger Fabric був обраний з таких причин. Приватна модель доступу – дозволяє будувати внутрішні блокчейн-мережі, де кожен вузол представляє перевіреного учасника (наприклад, департамент компанії), що важливо для зберігання персональних даних.

Модульність Fabric дозволяє легко підключати або замінювати компоненти, механізми консенсусу, бази даних, сервіси ідентифікації.

Підтримка смарт-контрактів (chaincode) на Go/JavaScript – це дало змогу розробити власні правила доступу, політики згоди, логіку автентифікації.

Висока продуктивність та масштабованість – важливо при обробці великої кількості транзакцій, особливо у системах з багатьма запитами на доступ до даних.

Вибір методів шифрування AES + RSA Для безпосереднього шифрування персональних даних було проаналізовано кілька криптоалгоритмів DES, AES, Blowfish, RSA, ECC. Основним критерієм стали – швидкодія, криптостійкість, наявність апаратної підтримки (AES-NI), підтримка в бібліотеках C#.

В результаті було обрано гібридну модель шифрування, яка поєднує AES (Advanced Encryption Standard) симетричний алгоритм, який забезпечує швидке шифрування/дешифрування на стороні клієнта;

RSA (Rivest–Shamir–Adleman) асиметричний алгоритм, який використовується для безпечного шифрування симетричного ключа AES перед передачею. Це дозволяє уникнути проблем із розповсюдженням ключів та підтримувати баланс між швидкістю і надійністю.

Вибір сховища MongoDB з шифруванням TDE Для зберігання зашифрованих персональних даних використовується MongoDB з Transparent Data Encryption (TDE). Цей вибір обумовлений

Підтримкою шифрування «на льоту» усі записи шифруються перед записом на диск. Зручністю зберігання JSON-документів формат ідеально підходить для персоніфікованих записів користувачів.

Масштабованістю можливість легко розгорнути кілька екземплярів бази даних у хмарі або локально.Вбудованими механізмами контролю доступу на рівні ролей. Система управління ключами HashiCorp Vault / AWS KMS Збереження приватних ключів, що використовуються для розшифрування AES-ключів, є критичним завданням. У даній системі розглянуто два варіанти:

–HashiCorp Vault для локального використання; забезпечує політики доступу, аудит доступів, інтеграцію з LDAP.

–AWS Key Management Service (KMS) для хмарного розгортання, з підтримкою апаратного HSM і аудитом у CloudTrail.

Мережевий рівень захисту

Для запобігання перехопленню трафіку між модулями системи застосовуються такі механізми TLS/SSL-зашифровані канали для всіх REST і gRPC запитів.VPN для адміністративного доступу до серверів з ізольованого середовища. Клієнтські сертифікати X.509 для автентифікації запитів від мікросервісів.

Аутентифікація користувачів OpenID + 2FA Оскільки система передбачає взаємодію з користувачами, було реалізовано механізм автентифікації через OpenID Connect (OIDC), який дозволяє інтеграцію з Google, Microsoft або власним Identity Provider. Двофакторна автентифікація (2FA) реалізована через Time-based One-Time Password (TOTP), що генерується мобільним додатком (Google

Authenticator, Microsoft Authenticator). Аудит та моніторинг Splunk/ELK + блокчейн

Для забезпечення непідробного аудиту всі події (запис, перегляд, зміна даних) фіксуються у централізованій системі логування (Splunk або ELK), та паралельно – в блокчейні, що гарантує незмінність логів.

Це дозволяє проводити розслідування інцидентів, підтверджувати дії в юридичному аспекті, автоматично реагувати на аномалії в поведінці користувачів.

Таким чином, кожне рішення в архітектурі системи не є випадковим, а продиктоване вимогами безпеки, продуктивності та відповідності чинним законодавчим нормам. Запропонована конфігурація дає змогу створити гнучку, масштабовану та надійну систему, здатну працювати як у державних структурах, так і в приватному секторі, де обробка персональних даних є критично важливою

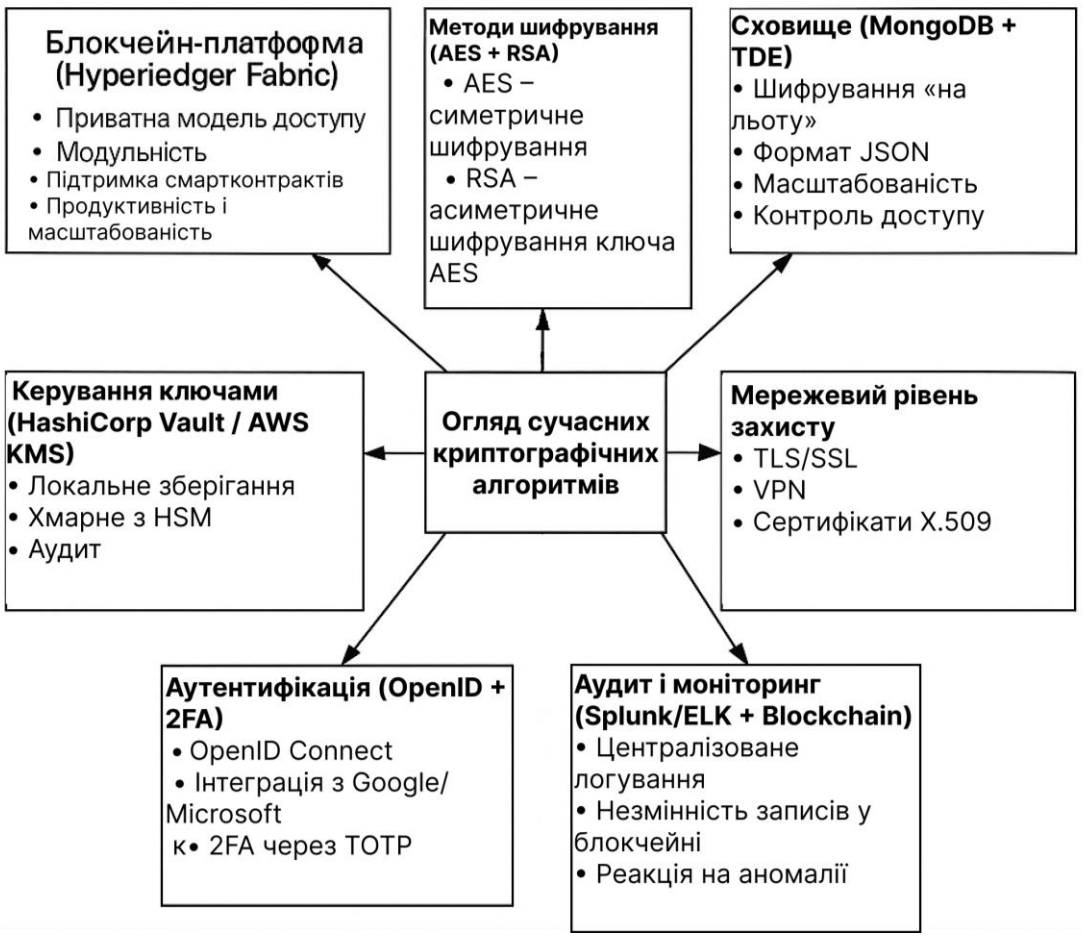


Рисунок 2.1 – Компоненти криптографічної архітектури системи захисту даних

2.2 Вибір алгоритмів шифрування та збереження даних

Одним із ключових аспектів забезпечення конфіденційності персональних даних є правильний вибір алгоритмів шифрування та організація їх безпечного збереження. У сучасних інформаційних системах, зокрема тих, що працюють із критично важливою інформацією, важливо не лише зашифрувати дані, а й створити стійку до атак архітектуру збереження, яка виключає можливість несанкціонованого доступу, підміни або втрати інформації.

Шифрування забезпечує перетворення даних у такий вигляд, який стає незрозумілим для сторонніх осіб, якщо вони не мають спеціального ключа дешифрування. Таким чином, навіть у випадку перехоплення трафіку або доступу до носія даних, інформація залишається захищеною. Зважаючи на актуальність атак на канали передачі даних, файлові системи та хмарні сховища, використання криптографічного захисту є обов’язковою складовою будь-якої сучасної системи інформаційної безпеки.

Згідно з сучасними стандартами, до ефективних симетричних алгоритмів шифрування належать AES (Advanced Encryption Standard) та ChaCha20. AES є загальновизнаним стандартом, який широко використовується в державних і комерційних системах. Він забезпечує високу швидкість обробки даних, при цьому залишається криптостійким проти відомих методів атак, таких як brute-force, диференціальний криптоаналіз або атаки на побічні канали. ChaCha20, у свою чергу, вирізняється кращими показниками швидкодії в програмному середовищі, особливо на мобільних пристроях, та підвищеною безпекою за відсутності апаратної підтримки AES.

У системах, де необхідна вища стійкість до несанкціонованого доступу, також застосовуються асиметричні алгоритми шифрування. Найпоширенішими серед них є RSA та алгоритм на еліптичних кривих (ECC). RSA дозволяє шифрувати дані з використанням пари ключів – відкритого та закритого. Проте через зростання обсягів даних та вимог до продуктивності все частіше застосовують ECC, оскільки він забезпечує таку саму криптостійкість при значно

меншій довжині ключа. Це дозволяє зменшити обчислювальні витрати, що особливо важливо у великих розподілених системах, де критичною є швидкість обробки запитів.

Для збереження персональних даних найкращим підходом є комбінація шифрування та дотримання принципу мінімізації прав доступу. У розробленій системі пропонується використання шифрування даних «на місці» (at-rest), коли кожен запис у базі шифрується індивідуально. Для цього застосовується AES у режимі GCM (Galois/Counter Mode), що одночасно забезпечує конфіденційність та цілісність даних, дозволяючи виявляти підміну інформації. Ключі до кожного запису зберігаються у захищеному сховищі, доступ до якого здійснюється лише після автентифікації користувача та підтвердження його прав через смарт-контракт у блокчейн-мережі.

Особливу увагу необхідно приділити розподіленим способам зберігання інформації. У запропонованій архітектурі персональні дані зберігаються у фрагментованому вигляді, з розміщенням зашифрованих блоків у різних частинах системи або на окремих вузлах. Це ускладнює несанкціонований доступ до повного масиву даних навіть у разі компрометації одного з вузлів. Використовується принцип розподіленого шифрування з можливістю відновлення даних лише при наявності контрольної множини ключів, що також керується через блокчейн.

Крім цього, реалізовано підтримку збереження історії змін, яка дозволяє виявити факт несанкціонованого втручання. Для цього всі операції з даними логуються та записуються у децентралізований блокчейн-журнал. Будь-яка зміна, знищення або додавання даних фіксується у вигляді транзакції, що підписується ключем відповідального користувача. Завдяки незмінності блокчейн-структури забезпечується цілісність журналу дій і його захищеність від фальсифікації.

Окремим елементом архітектури зберігання є резервне копіювання. Усі зашифровані дані автоматично дублюються в резервні вузли, що фізично розміщуються в інших дата-центрах. Перед записом вони додатково шифруються іншим ключем, що генерується для кожної резервної сесії. Доступ до резервних

копій суворо обмежений, виконується лише через спеціальний модуль, який перевіряє цілісність копії, її достовірність та дозвіл на відновлення через блокчейн-контракт.

У процесі вибору алгоритмів шифрування також враховуються вимоги до сумісності з міжнародними стандартами та законодавством у сфері захисту персональних даних. Усі використані криптографічні засоби відповідають вимогам стандартів NIST, GDPR та ДСТУ України. Це дозволяє забезпечити правову обґрунтованість функціонування системи в умовах реального використання.

Таким чином, реалізація багаторівневої криптографічної системи із застосуванням перевірених алгоритмів шифрування, децентралізованого збереження, логування змін у блокчейні та автоматичного резервного копіювання дозволяє досягти максимальної безпеки персональних даних. Така система стає не лише технічно захищеною, але й юридично відповідною вимогам до сучасних інформаційних середовищ.

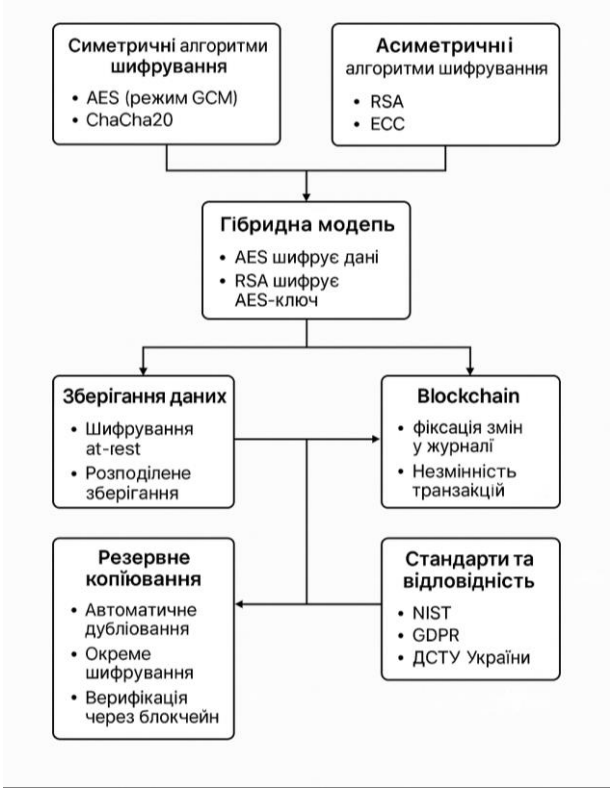


Рисунок 2.2 – Вибір алгоритмів шифрування та збереження даних у системі

2.3 Механізм доступу на основі смарт-контрактів

Механізм доступу до персональних даних на основі смарт-контрактів це сучасний підхід, що поєднує в собі криптографічну безпеку, прозорість та автоматизацію прийняття рішень у межах децентралізованої інфраструктури. Смарт-контракти це автономні програмні скрипти, які зберігаються та виконуються в блокчейн-мережі. Вони автоматично виконують наперед визначені дії за умови настання заданих умов, що робить їх ідеальним інструментом для управління правами доступу до конфіденційної інформації, зокрема персональних даних.

У традиційних системах контроль доступу зазвичай базується на централізованих механізмах сертифікатах, таблицях прав користувачів, ролях тощо. Проблема таких підходів полягає у вразливості до маніпуляцій з боку адміністраторів, а також у складності відстеження змін прав доступу в ретроспективі. Смарт-контракти натомість фіксують усі операції в незмінному середовищі блокчейну, забезпечуючи цілковиту прозорість, підзвітність і довіру до дій системи.

У запропонованій архітектурі смарт-контракт виконує роль центрального регулятора доступу. Він зберігає правила, за якими визначається, хто, коли і до яких саме даних має право доступу. Наприклад, смарт-контракт може містити умову, "якщо користувач має роль 'лікар', він може переглядати медичні дані пацієнтів лише за наявності відповідної згоди". Якщо така згода збережена в блокчейні (у вигляді окремого запису), то доступ дозволяється, інакше ні.

Ще однією перевагою смарт-контрактів є те, що вони зберігаються у відкритому реєстрі, який доступний для перегляду всіма сторонами. Таким чином, можна легко перевірити, за якими правилами надається доступ, і бути впевненим у відсутності підробки або підміни умов. Це критично важливо для захисту персональних даних, особливо у випадках, коли законодавство (наприклад, GDPR) вимагає чіткого обліку всіх дій з персональною інформацією.

Технічно смарт-контракти реалізуються на мовах програмування, що підтримуються відповідною блокчейн-платформою (наприклад, Solidity для Ethereum або Move для Aptos). Вони взаємодіють із системою через API, забезпечуючи інтеграцію з веб-інтерфейсами, внутрішніми сервісами підприємства та зовнішніми партнерами. Наприклад, при спробі користувача доступитись до певного ресурсу, система надсилає запит до смарт-контракту, який аналізує права доступу і повертає відповідь дозволити, відхилити або запитати додаткове підтвердження.

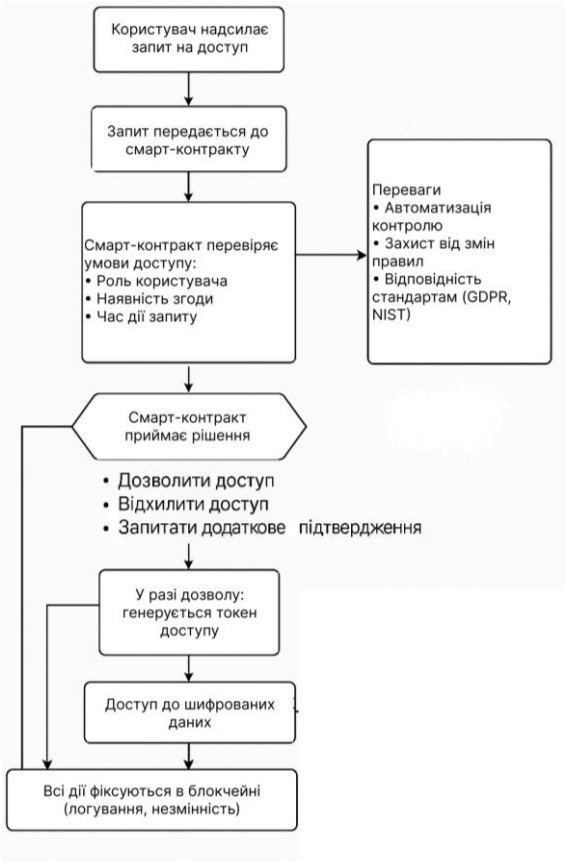


Рисунок 2.3 – Механізм контролю доступу даних за допомогою смарт-контрактів

Оскільки смарт-контракти працюють у межах децентралізованої мережі, вони не підлягають односторонньому редагуванню чи знищенню. Це додає ще один рівень захисту від внутрішніх загроз навіть адміністратор системи не може змінити правила доступу без створення нового контракту, що автоматично

фіксується у реєстрі. Також смарт-контракти мають вбудовані механізми обмеження часу дії, багаторазових перевірок, токенізації дозволів тощо.

Іншою цікавою опцією є використання токенів доступу (access tokens), створених на базі смарт-контрактів. Вони можуть виступати як цифрові ключі, що видаються користувачам на обмежений період або за умови досягнення певних критеріїв. Наприклад, для відкриття медичних даних пацієнта лікар повинен отримати токен доступу, що генерується смарт-контрактом після згоди пацієнта, підтвердженої електронним підписом.

Таким чином, впровадження механізму доступу на основі смарт-контрактів у системі захисту персональних даних дозволяє досягти таких переваг повна автоматизація перевірки прав доступу зменшення впливу людського фактора на контроль безпеки. Прозорість і фіксація всіх змін у доступі. Захист від підробки або несанкціонованого доступу. Відповідність вимогам міжнародних стандартів з захисту персональної інформації.

Цей підхід відкриває новий рівень довіри між усіма учасниками обміну інформацією, адже всі дії стають відтворюваними, підконтрольними і не можуть бути змінені заднім числом. У результаті система отримує потужний, сучасний та стійкий інструмент управління доступом до чутливих даних.

Додатково, важливо враховувати, що смарт-контракти можуть бути частиною більш складної багаторівневої системи доступу, яка поєднує централізовані сервіси (наприклад, сервіси автентифікації) з децентралізованим контролем повноважень. У таких випадках, смарт-контракт виступає не лише як механізм дозволу або заборони, а і як система логіки прийняття рішень, що взаємодіє з іншими компонентами (сервіси ідентифікації, журнали аудиту, бази метаданих). Це дозволяє будувати гнучкі схеми авторизації, які можуть враховувати, наприклад, геолокацію, пристрій, контекст запиту чи навіть попередні дії користувача.

Існують також моделі доступу на основі атрибутів (ABAC), які можна реалізувати у формі смарт-контрактів. Такий підхід полягає у наданні доступу не лише на основі ролі, а й у відповідності до визначених властивостей користувача

або запиту: його статусу, віку, типу даних, до яких запитано доступ. Наприклад, у випадку із соціальними службами, доступ до певної інформації про особу надається лише тоді, коли користувач є співробітником відповідного відомства, дані запитуються в робочий час і запит стосується актуальної справи. Смарт-контракт у такій ситуації перевіряє всі ці умови перед прийняттям рішення.

Особливу увагу слід приділяти питанню безпеки самих смарт-контрактів. Їхній код повинен бути ретельно протестований та проходити аудит безпеки перед розгортанням. Оскільки код у блокчейні не можна змінити після публікації, помилки або вразливості можуть стати критичними. Для цього створюються спеціальні тестові середовища (тестнети), де всі сценарії доступу можна змоделювати та перевірити без ризику порушення конфіденційності реальних даних.

З практичної точки зору, у побудові таких систем також рекомендується застосовувати концепцію "мінімального необхідного доступу" (principle of least privilege), згідно з якою смарт-контракт дозволяє лише ті дії, які прямо передбачені для конкретного типу користувача, у конкретному контексті. Це зменшує ризик зловживань або несанкціонованих дій, навіть якщо зловмисник отримає доступ до облікового запису.

Таким чином, поєднання смарт-контрактів із сучасними підходами до авторизації та валідації доступу дозволяє побудувати систему управління доступом, що не лише відповідає законодавчим вимогам, а й фактично мінімізує вразливості, характерні для традиційних ІТ-рішень.

2.4 Верифікація автентичності даних

Верифікація автентичності даних є критично важливим етапом у побудові системи захисту персональних даних, особливо в умовах використання децентралізованих технологій, таких як блокчейн. Її основна мета гарантувати, що

отримані або передані дані не були змінені, підроблені або перехоплені сторонніми особами, а також що їх джерело є достовірним та авторизованим.

У контексті персональних даних автентичність означає, що:

–дані походять саме від заявленого суб’єкта або джерела;

–дані не були змінені з моменту їх створення або передачі;

–підтверджено час та умови створення або запису даних.

Рисунок 2.4 – Основні механізми верифікації даних у блокчейн-системі

Основні механізми верифікації автентичності це цифровий підпис – один з найнадійніших і найпоширеніших методів забезпечення автентичності даних. Він створюється за допомогою криптографії з відкритим ключем, де

закритий ключ використовується для підпису даних (використовується лише підписантом), відкритий ключ – для перевірки підпису (доступний усім).

У системах, побудованих на блокчейні, цифровий підпис дозволяє:

–засвідчити, що дані були створені або змінені конкретним суб’єктом;

–довести, що дані не зазнали змін після підписання;

–виявити фальсифікацію або підміну інформації.

Наприклад, якщо користувач завантажує в систему медичний документ, він підписується його особистим ключем. Будь-який інший учасник системи може переконатися в автентичності документа, перевіривши підпис відкритим ключем користувача.

Наступний механізм це хешування – це обчислення унікального цифрового відбитка (хешу) від вхідних даних. Навіть найменша зміна у файлі призводить до повної зміни його хешу. Це дозволяє з високою впевненістю перевіряти чи не були змінені дані з моменту запису до блокчейну чи відповідає отриманий файл оригіналу.

У блокчейн-системах хеш від кожного блоку, транзакції або документа записується у незмінному реєстрі. Це означає, що автентичність даних можна перевірити шляхом повторного хешування і порівняння з еталонним хешем дані не можуть бути підроблені або видалені без порушення цілісності всієї системи.

Також використовується мітки часу (timestamping) це механізм фіксації точного часу створення або запису інформації. У блокчейні кожен блок має власну мітку часу, яка автоматично встановлюється при генерації блоку. Це дозволяє:

–підтвердити момент виникнення або модифікації інформації;

–забезпечити доказ існування певного документа на конкретну дату;

–запобігти спробам "заднім числом" змінити дані.

У випадках правових спорів або перевірки достовірності, блокчейн-фіксація дати може виступати як доказ у суді або під час аудиту.

Ідентифікація джерела (автентифікація учасника) використовується для того щоб кожна дія у блокчейн-системі прив'язана до конкретного користувача або об'єкта через криптографічні ключі. Це забезпечує:

–чітке розмежування, хто саме створив або надіслав дані;

–запобігання підробці даних з боку неавторизованих осіб;

–відстеження історії змін – хто, коли і що саме зробив.

У поєднанні з цифровими сертифікатами та біометричною аутентифікацією цей підхід дозволяє досягти максимальної надійності ідентифікації джерела даних.

Приклад реалізації механізму верифікації у системі захисту персональних даних кожен запис про користувача супроводжується:

–цифровим підписом джерела (наприклад, лікаря або співробітника установи);

–хешем документа, збереженим у блокчейні;

–міткою часу;

–публічним ідентифікатором користувача.

При запиті до таких даних інший користувач або система перевіряє:

–чи збігається хеш документа з тим, що збережений у блокчейні;

–чи дійсний цифровий підпис і ким він був створений;

–чи знаходиться запитувач у списку дозволених осіб згідно смарт-контракту.

Переваги механізму верифікації неможливість підробки дані, що пройшли цифрову підписку й хешування, практично неможливо підробити непомітно. Автоматичність – перевірки всі дії можуть бути автоматизовані, з мінімальним втручанням людини. Прозорість і довіра– будь-який користувач може перевірити автентичність даних самостійно.

Відповідність стандартам безпеки – система повністю відповідає принципам GDPR, ISO/IEC 27001 тощо

Крім перелічених методів, у сучасних системах захисту персональних даних дедалі частіше впроваджуються комплексні підходи, які поєднують кілька механізмів автентифікації для посилення довіри до джерела даних. Зокрема, інтеграція багатофакторної автентифікації (2FA) у процес підпису та доступу до даних забезпечує додатковий рівень перевірки особи. Наприклад, перед здійсненням підпису або ініціації транзакції користувач проходить підтвердження через OTP-код, мобільний додаток або біометричні дані, що унеможливлює використання його ключів сторонніми особами.

Ще одним актуальним напрямом є використання доказів з нульовим розголошенням (Zero-Knowledge Proofs, ZKP), які дозволяють довести факт володіння або автентичності інформації без необхідності її розкриття. Такий підхід особливо корисний у контексті захисту конфіденційних персональних даних, коли потрібно переконатися в їх достовірності, не відкриваючи їх змісту. Наприклад, у фінансових або медичних системах ZKP може підтвердити, що особа має певний статус або дозвіл, не розкриваючи подробиць про неї.

Для побудови стійких систем аудиту і відповідності нормативним вимогам також все частіше впроваджуються децентралізовані ідентифікатори (DIDs) та віртуальні облікові дані (Verifiable Credentials). Вони дозволяють створювати цифрові посвідчення особи, що зберігаються у контролі самого користувача і можуть бути представлені верифікаторам на запит без посередників. У поєднанні з блокчейн-реєстрами це створює гнучку, захищену модель обміну перевіреними даними, яка не потребує централізованих сховищ і знижує ризики витоку.

З практичного погляду, реалізація механізмів верифікації у прикладних системах має враховувати як технічні аспекти (протоколи, алгоритми, швидкодію), так і правові та етичні стандарти, зокрема — принципи мінімізації даних, обмеження доступу за ролями, та прозорість алгоритмів обробки. Наприклад, у відповідності до GDPR, система має забезпечувати можливість користувача перевірити, хто і коли звертався до його персональної інформації, з якою метою, та які дії були здійснені.

Таким чином, верифікація автентичності даних у блокчейн-системах є не лише технічним процесом, а й інструментом підвищення довіри, відповідності регуляторним вимогам і гарантування цифрової етики.

2.5 Модуль контролю обігу документів з блокчейн-аудитом

Розробка цього модуля полягає у тому що , виникає потреба у створенні механізмів, які б гарантували не лише технічний захист, а й повний юридично

значущий аудит дій над документами. Для досягнення цього в межах розробленої системи передбачено впровадження модуля контролю обігу документів із використанням блокчейн-аудиту та гібридного шифрування.

Загальна ідея модуля. Модуль реалізує захищене середовище обробки електронних документів, зокрема таких, що містять чутливу або регламентовану законом інформацію (медичні довідки, фінансові звіти, службові листи). Ключовою особливістю є повна прозорість та незмінність логіки дій, яка реалізується через смарт-контракти в блокчейні. Таким чином, жодна із зафіксованих дій не може бути змінена чи прихована після її виконання.

Особливості модуля .Криптографічний захист вмісту документів – симетричне шифрування AES-256-GCM та цифровий підпис ECDSA/RSA. Розподілене зберігання – використання IPFS як довготривалого сховища або MongoDB з вбудованим шифруванням (TDE). Блокчейн-контроль – смарт-контракт фіксує всі дії над документом: створення, затвердження, перегляд, відкликання. Гнучка авторизація – доступ до документів здійснюється згідно з правами, зафіксованими в контрактній логіці.

Компоненти архітектури . Клієнтський застосунок (frontend/backend). Реалізований за допомогою C# або JavaScript (веб-інтерфейс). Відповідає за завантаження файлів, взаємодію з API, формування запитів до блокчейну.

Криптографічний модуль. Використовує AES-256-GCM для шифрування вмісту документа.AES-ключ шифрується за допомогою RSA-2048 або ECDSA, що гарантує безпечну передачу ключів. Хеш документа (SHA-256) обчислюється для забезпечення цілісності. Цифровий підпис формується із використанням приватного ключа автора документа.

Розподілене сховище. IPFS – зберігає зашифрований документ у вигляді CID (Content Identifier). MongoDB з TDE – використовується як локальне або резервне сховище.

API-сервер (middleware). Приймає запити на реєстрацію, підпис, доступ. Перевіряє автентичність користувача та генерує транзакції для смарт-контракту.

Смарт-контракт (Ethereum/Solana). Зберігає реєстр документів: documentId, hash, owner, status, timestamp. Дозволяє виклик методів: registerDocument(), approveDocument(), revokeDocument(), logAccess().

Блокчейн-інфраструктура (Web3). Користувач взаємодіє з блокчейном через Nethereum/Web3.js. Усі транзакції підписуються приватним ключем користувача.

Детальний алгоритм функціонування. Крок 1: Завантаження документа. Користувач обирає документ (наприклад, report.pdf). Система генерує випадковий симетричний ключ AESKey та IV. Документ шифрується: EncryptedData = AES256-GCM(report.pdf, AESKey, IV). Обчислюється хеш: docHash = SHA256(EncryptedData). AESKey та IV шифруються публічним ключем системи: EncKey = RSA(PubKey, AESKey + IV). Формується цифровий підпис Signature = Sign(PrivateKeyUser, docHash).

Крок 2: Збереження. EncryptedData зберігається у IPFS → отримується CID.

Через API формується транзакція: solidity КопіюватиРедагувати, registerDocument(docId, docHash, CID, Signature, ownerAddress, Status: PENDING, timestamp)

Крок 3 Підпис або затвердження. Інший користувач (наприклад, лікар) авторизується. Надсилається транзакція на смарт-контракт: solidity КопіюватиРедагувати approveDocument(docId) або solidity КопіюватиРедагувати rejectDocument(docId) Усі дії фіксуються як події Ethereum EventLog (незмінні, прозорі, відкриті).

Крок 4 Перегляд і аудит. Користувач запитує доступ до документа.

Система перевіряє – чи дозволено доступ за умовами контракту, чи відповідає SHA256(EncryptedData) → docHash, чи дійсний цифровий підпис користувача. Якщо всі умови виконані, надається доступ і логуються дії – Solidity КопіюватиРедагувати logAccess(docId, viewerAddress, timestamp)

Крок 5 Відкликання документа Адміністратор або власник документа викликає solidity КопіюватиРедагувати revokeDocument(docId) Статус змінюється на REVOKED, доступ автоматично забороняється.

Безпекові властивості системи Неможливість підробки – усі дії підписуються приватними ключами користувачів та фіксуються в блокчейні. Відсутність централізованого адміністрування – контроль доступу реалізовано смарт-контрактом, зміна якого потребує оновлення мережевого консенсусу.

Автоматизація юридичної відповідальності – підпис і хеш є цифровими доказами, що відповідають нормам eIDAS, GDPR та ЗУ «Про електронні документи».

Можливість зовнішнього аудиту – лог дій відкритий для перевірки сторонніми сторонами, включаючи контролюючі органи.

Практичне застосування vодуль може бути інтегрований у:

Медичні ІС – захист доступу до медичних записів пацієнтів;

– Фінансові сервіси – аудит транзакційних звітів;

– Освітні системи – контроль обігу дипломів і довідок;

– Електронний документообіг у держорганах – підтвердження статусу, згоди, затвердження.

Цей модуль дозволяє поєднати високий рівень захисту даних, прозорість дій та децентралізовану юридичну відповідальність. Завдяки використанню блокчейну забезпечується не лише технічна безпека, але й формування довіри до процесів, що мають цифрову доказову силу у правовому полі.

3 ШИФРУВАННЯ ФАЙЛІВ ВІДПОВІДНО ДО РІВНЯ ТАЄМНОСТІ

3.1 Реалізація прототипу

Розроблена система захисту особистих даних передбачає конфіденційність і цілісність інформації вже на боці клієнта. Для цього дані шифруються ще до відправки на зберігання – поєднуються переваги симетричного AES-шифрування (для швидкої обробки великих обсягів даних) та асиметричного RSA (для безпечного обміну ключем). У результаті на локальний пристрій користувача потрапляють лише зашифровані дані, а відкрите середовище блокчейна отримує тільки криптографічно захищені рядки.

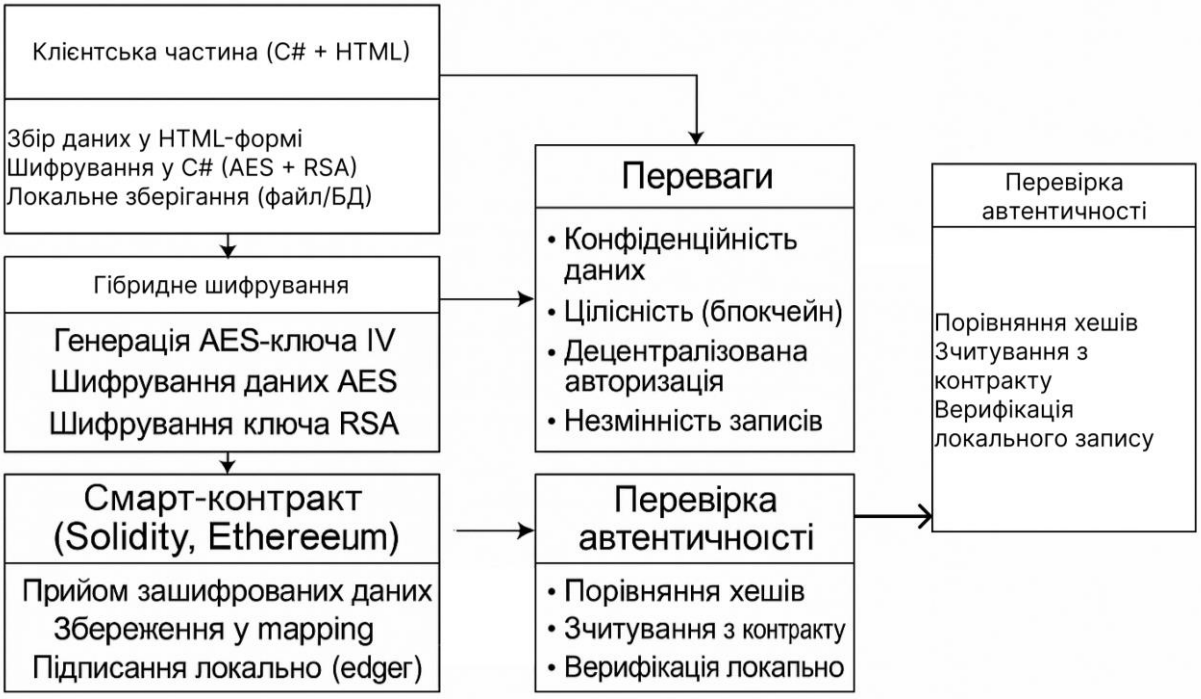


Рисунок 3.1 – Архітектура прототипу

Основні компоненти системи наступні клієнтська частина (C# + HTML). Веб-інтерфейс (HTML-форма) збирає особисті дані, а код на C# обробляє їх криптографічно та взаємодіє з локальним сховищем і блокчейном.

Гібридне шифрування. На клієнті генерується випадковий симетричний ключ AES та ініціалізаційний вектор (IV), якими шифрується вміст даних. Потім цей AES-ключ і IV шифруються за допомогою RSA (публічного ключа системи).

Такий двоетапний підхід («гібридне шифрування») використовує швидкість AES для даних і безпеку RSA для ключа.

Локальне сховище. Зашифровані записи разом із зашифрованим AES-ключем зберігаються локально (наприклад, у файлі або захищеній базі даних). Таким чином навіть при доступі до пристрою дані залишаються захищеними.

Смарт-контракт (Solidity). На Ethereum розгорнуто контракт, який приймає зашифровані записи (типу string) і зберігає їх у власному стані (наприклад, у mapping з ключем – ідентифікатором запису). Усі транзакції з контрактом підписуються приватним ключем користувача, тому традиційна авторизація у C# не потрібна.

Взаємодія з блокчейном (Nethereum). Бібліотека Nethereum у C# використовується для виклику функцій смарт-контракту. Користувач завантажує свій приватний ключ (наприклад, із keystore-файлу) у об’єкт Account, потім створює Web3 з цим обліковим записом. Далі всі транзакції та виклики контракту будуть підписані цим ключем локально.

Така архітектура гарантує, що дані в блокчейні зберігаються у незмінному вигляді («immutable ledger») як зазначено в літературі, записи у блокчейні «не можна змінити, видалити або іншим чином сфальсифікувати». Одночасно відкритість блокчейну означає, що бачать лише зашифрований контент чи його хеш. Розділення функцій гарантує довіру до даних конфіденційність забезпечує клієнтське шифрування, а блокчейн – їхню цілісність та відсутність централізованого контролю.

Гібридне шифрування на клієнті у нашій системі дані шифруються перед будь-яким збереженням чи передачею. Спочатку використовується симетричний алгоритм AES (наприклад, AES-256) для шифрування основного тексту даних. AES є надзвичайно ефективним для великих обсягів і забезпечує сильний захист при відповідній довжині ключа. Однак передати секретний ключ AES іншим учасникам без його компрометації складно, тому за допомогою RSA шифрується тільки цей ключ (і ініціалізаційний вектор IV). Як ідеться в рекомендаціях, «асиметричні алгоритми звичайно використовуються для шифрування

невеликого обсягу даних, наприклад симетричного ключа». У сумі такий підхід називають гібридним шифруванням він «поєднує сильні сторони AES та RSA».

Наприклад генерується випадковий AES-ключ і IV (C#-клас Aes або RijndaelManaged). Дані шифруються цим AES-ключем, потім публічним ключем RSA шифрується саме значення AES-ключа (часто і IV).

У результаті отримаємо два компоненти шифротекст даних (байтова послідовність або Base64-рядок) і зашифрований ключ. Саме ці компоненти потім зберігаються у системі. Використання RSA гарантує, що навіть якщо хтось отримає доступ до зашифрованого запису, без приватного ключа неможливо відновити AES-ключ і розшифрувати дані. Документація ілюструє такий підхід клас HybridEncryption у C# фактично реалізує «методи для поєднання RSA та AES».

Локальне зберігання даних зображується після шифрування зашифровані записи зберігаються локально на пристрої користувача. Це може бути файлова система або локальна база даних (наприклад, SQLite) з додатковим захистом паролем. Важливо, що дані «в стані спокою» теж залишаються зашифрованими, тобто навіть при фізичному доступі до файлу сторонній не побачить розшифрований контент. Такий підхід відповідає принципам захисту даних at-rest. На рівні реалізації C# можна скористатися стандартними засобами криптографії класами Aes/Rijndael для симетричного шифрування та RSA для асиметричного (див. приклади на MSDN). Результати (шифротекст та зашифрований ключ) записуються в базу даних або файл разом з додатковими метаданими (ідентифікатор запису, відмітка часу тощо).

Таке локальне зберігання забезпечує, по-перше, автономність системи (дані не покидають пристрій безпечно зашифрованими), а по-друге, подвоює захист навіть якщо приватний ключ RSA буде скомпрометовано, злочинець побачить лише зашифровані AES-блоки, а без самих ключів відкрити їх буде надзвичайно складно. Гібридна схема розділяє завдання продуктивності і безпеки між алгоритмами.

Смарт-контракт Solidity для зберігання записів. На стороні блокчейну реалізується смарт-контракт, який фіксує зашифровані записи (або їх хеші) у ланцюжку блоків. Важливо, що Ethereum є незмінним реєстром як зазначено в літературі, дані в такому реєстрі «не можна змінити, видалити або іншим чином сфальсифікувати». Це робить кожен запис відомим і довіреним. Контракт може містити, наприклад, mapping(uint256 => string) public records; і функцію addRecord(uint256 id, string memory encData), що зберігає зашифрований рядок під відповідним ключем-ідентифікатором. При додаванні запису C#-застосунок викликає цю функцію (через Nethereum) і передає їй рядок з зашифрованими даними або їх хеш.

Оскільки блокчейн публічний, усі учасники мережі бачать стан контракту. Тому до блокчейну передаються лише ті дані, які не відкривають чутливу інформацію лише зашифровані рядки або їх криптографічні хеші. Навіть якщо учасник бачить цей рядок, без приватного ключа він не дізнається змісту запису. Проте відкритість мережі створює ризик приватності «усі дані, що зберігаються в публічному блокчейні, видимі для всіх учасників». Тому контракт не повинен приймати незашифровані персональні дані. Натомість, наприклад, можна зберігати лише хеші (SHA-256 чи Keccak256) записів для перевірки цілісності, а сам зашифрований текст – офлайн у локальній БД. Якщо ж зашифровані дані передаються в контракт (як у нашому випадку), цей рядок слугує доказом факту існування запису в момент транзакції, але без ключів RSA ані вміст, ані його роль не можуть бути зловмисником використані.

Перевага такого підходу – прозорість і незмінність записів. Блокчейн гарантовано фіксує події у хронологічному порядку, тож можна будь-коли прослідкувати, коли саме й що додавалося. Згідно з ідеєю «immutable ledger», зашифровані записи залишаються «надійними з часом» і довіреними. З іншого боку, обсяг даних, що реально можна зберігати в Ethereum, обмежений через високу вартість транзакцій. Тому на практиці зазвичай зберігають або короткі хеші, або відносно невеликі зашифровані фрагменти.

Взаємодія з блокчейном через Nethereum. Для роботи з Ethereum з C# використовується бібліотека Nethereum – вона спрощує створення транзакцій, виклик функцій контрактів і обробку результатів. Основна ідея така користувач створює об’єкт Account, завантажуючи свій приватний ключ (із keystore-файлу або рядка JSON). Цей Account передається у конструктор new Web3(account). Після цього будь-яка транзакція, відправлена цим об’єктом, буде автоматично підписана локально приватним ключем. Таким чином не потрібна вбудована авторизація або введення паролю для кожної операції – самим ключем користувач засвідчує свою «авторизацію».

Щодо виклику функцій контракту, Nethereum дозволяє працювати з ABI смарт-контракту. На практиці реалізується так

Створюється екземпляр контракту через var contract = web3.Eth.GetContract(abi, contractAddress).

Отримується об’єкт функції, наприклад var addFunc = contract.GetFunction("addRecord").

Викликається SendTransactionAsync(senderAddress, param1, param2,), де senderAddress – адреса облікового запису, який сплачує газ, а наступні параметри – аргументи функції.

Як наголошує документація, «об’єкт функції спрощує надсилання транзакцій достатньо вказати senderAddress (який сплачуватиме газ) та параметри функції». Бібліотека сама розраховує необхідне газове забезпечення або дозволяє його вказати додатково. Після надсилання транзакції можна відслідковувати статус (receipt) та чекати підтвердження. Зауважимо, що всі ці операції здійснюються в .NET-коді C# без використання додаткових серверів чи середовищ авторизації – єдиним «дозволом» є володіння приватним ключем Ethereum-адреси. Це узгоджується з умовою задачі авторизація у C# не застосовується, доступ визначається приватним ключем.

Додавання запису і перевірка автентичності.Процес збереження нового персонального запису можна описати поетапно

Збір даних – користувач вводить або завантажує персональні дані в HTML-формі.

AES-шифрування – генерується випадковий симетричний ключ AES і IV. Ці ключ і IV використовуються для шифрування тексту даних.

RSA-шифрування ключа – згенерований AES-ключ (і IV) шифруються публічним RSA-ключем системи. Отримуємо закодований ключ.

Локальне зберігання – зберігаються шифротекст даних та зашифрований AES-ключ (у базі даних або файлі). Можна також зберегти супровідні дані – наприклад, унікальний ідентифікатор запису або часову мітку.

Обчислення хешу – для створення доказу цілісності обчислюється хеш (наприклад, SHA-256) зашифрованого запису. Цей хеш служить маркером («відбитком») даних.

Транзакція до контракту – використовуючи Nethereum, формується та відправляється транзакція до смарт-контракту. У ній передається хеш або сам зашифрований рядок. Функція контракту addRecord(id, data) зберігає цей рядок у своїй пам’яті.

Підтвердження і збереження ідентифікатора – після майнінгу транзакції отримуємо transactionHash і, опційно, адресу нового лога в контракті. Цей ідентифікатор запису фіксується локально для подальшої перевірки.

Для перевірки автентичності запису надалі зчитуються локально дані і обчислюється їхній хеш. Потім ми звертаємося до контракту (через відповідну функцію getRecord(id) або безпосередньо читаємо стан), щоб отримати збережений раніше хеш або зашифрований рядок. Порівняння локального хешу з тим, що на блокчейні, є надійним способом виявити будь-які зміни даних. Завдяки тому, що блокчейн є незмінним і «довіреним» сховищем, будь-яка невідповідність хешів свідчить про спробу фальсифікації. Як було зафіксовано, технологія блокчейн гарантує, що інформація, яку вона зберігає, «залишається стабільною й надійною з часом». Тож порівняння зашифрованих відбитків забезпечує перевірку цілісності та автентичності даних без необхідності розшифровувати сам вміст.

Обґрунтування архітектурних рішень. Описана архітектура поєднує сильні сторони кількох технологій. Гібридне шифрування на клієнті гарантує конфіденційність даних, оскільки навіть на сервер і в блокчейн потрапляє лише шифротекст, інформація захищена від перехоплення. AES забезпечує продуктивність, а RSA – безпеку розподілу ключа. Локальне зберігання даних відповідає концепції data at rest encryption – дані зберігаються завжди у зашифрованому вигляді.

Використання смарт-контракту і блокчейну додає гарантію цілісності та прозорості – записи, які потрапили в блокчейн, не можуть бути змінені або видалені. Це створює надійний аудит і виключає можливість ретроспективної корупції даних (наприклад, «післяфактної» підміни). Оскільки доступ ґрунтується на приватних ключах Ethereum, немає єдиного центру авторизації і відповідальність віддана користувачеві. Інтеграція Nethereum значно спрощує розробку інтерфейсу блокчейну у .NET-середовищі – бібліотека вже реалізує всі низькорівневі аспекти формування транзакцій, управління обліковим записом і взаємодії з ABI контрактів.

Переваги та обмеження рішення. Переваги – Конфіденційність даних. Персональні дані ніколи не передаються у незашифрованому вигляді, оскільки весь вміст шифрується на клієнті.

Цілісність та довіра. Блокчейн діє як незмінний журнал, раз записані дані «неможливо змінити чи стерти». Це гарантує, що зашифровані записи залишаються достовірними протягом усього часу існування контракту.

Безладно узалежнена авторизація. Немає потреби у введенні паролів або централізованій аутентифікації. Володіння приватним ключем Ethereum-адреси автоматично дає право виконувати транзакції. Цей підхід відповідає ідеї децентралізованої ідентичності.

Журнал операцій. Уся історія додавання записів зберігається в блокчейні. Це створює прозорий лог подій, корисний для аудиту та відстеження.

Обмеження– Вартість і масштабованість. Транзакції Ethereum оплачуються газом, тому зберігання великих обсягів даних у блокчейні є дорогим. Процедура

може бути повільною при навантаженні. Крім того, впровадження блокчейн-рішень є досить складним і ресурсомістким.

Проблеми приватності. Хоч блокчейн і шифрує самі дані, заголовки транзакцій та самі збережені рядки залишаються публічними. Як зауважено, «блокчейн-прозорість водночас створює проблеми приватності», адже усі дані видимі усім учасникам. Тому потрібно бути уважним з тим, що саме потрапляє до контракту.

Керування ключами. Безпека всього рішення залежить від захисту приватних ключів. Утрата чи крадіжка ключа призводить до повного контролю над можливістю модифікації записів у блокчейні (хоча без розшифровки даних самі записи залишаться у вигляді криптотексту). Таким чином необхідні механізми надійного зберігання ключів.

3.2 Тестування функціональності

Тестовий кейс 1 гібридне шифрування даних у C# (AES+RSA)

Вхідні дані вихідний рядок із персональними даними, наприклад, "Іван Іванович, 01.01.1990, 123-45-67-89", та відкритий RSA-ключ користувача, який було отримано заздалегідь і відповідає мінімальним вимогам криптостійкості (наприклад, довжина ключа становить 2048 біт). З метою забезпечення конфіденційності, дані підлягають гібридному шифруванню: спочатку формуються ключ і вектор ініціалізації для симетричного алгоритму AES, які використовуються для шифрування вихідного тексту. Потім згенерований AES-ключ шифрується за допомогою асиметричного RSA-алгоритму, використовуючи відкритий ключ користувача, і в результаті формується єдиний бінарний файл encrypted_data.bin, у якому зафіксовано як зашифрований AES-ключ, так і зашифрований текст повідомлення.

Очікуваний результат: у робочій директорії клієнтської програми на C# створюється файл encrypted_data.bin, що містить бінарний вміст — зашифровану

комбінацію AES-ключа та симетрично зашифрованого тексту. Надалі за допомогою RSA-закритого ключа користувача відбувається дешифрування AES-ключа, після чого виконується дешифрування основного повідомлення. Очікується повна відповідність результату дешифрування первинному відкритому тексту без втрати інформації. Фактичний результат: під час тестування була успішно реалізована клієнтська консольна програма на мові C#, що виконує гібридне шифрування відповідно до заданого алгоритму. В результаті виконання програми в директорії з'явився файл encrypted_data.bin, який містить коректно зашифровані дані у двійковому форматі. Подальше дешифрування із застосуванням відповідного RSA-закритого ключа дозволило без помилок відновити початковий AES-ключ, після чого було виконано розшифрування самого повідомлення, яке повністю збіглося з початковим текстом "Іван Іванович, 01.01.1990, 123-45-67-89". При цьому сам вміст створеного файлу не мав ознак схожості з вихідними даними, що підтверджує коректність шифрування і відсутність витоку вмісту. Таким чином, тестовий кейс пройдено успішно: реалізація забезпечила повну цілісність даних, правильність шифрування та відновлення тексту, а також підтвердила відповідність криптографічної реалізації вимогам до захисту персональних даних у межах розроблюваної системи.

Тестовий кейс 2 додавання запису до смарт-контракту (право власника)

Вхідні дані: з метою забезпечення безпечного збереження конфіденційної інформації в блокчейні, був використаний попередньо згенерований та зашифрований рядок даних, отриманий під час виконання гібридного шифрування (див. Тестовий кейс 1). Цей зашифрований рядок, що міститься у файлі encrypted_data.bin, подається на вхід функції addRecord(string encryptedData) смарт-контракту ConfidentialStorage. Доступ до виклику функції здійснюється з адреси, яка належить власнику контракту (Address0), через бібліотеку Nethereum у клієнтському додатку на C#. Смарт-контракт розгорнутий у тестовій локальній мережі (Ganache або Hardhat), що дозволяє забезпечити швидкий зворотній зв'язок та контроль за станом виконання транзакції. Очікуваний результат: транзакція, ініційована методом addRecord, має бути оброблена без помилок або

скасування (revert). Після успішного додавання запису в смарт-контракт очікується, що функція getCount() поверне значення 1, що означає наявність одного запису в системі. При цьому виклик getRecord(0) має повернути саме той рядок, що був переданий при додаванні — зашифровані дані, ідентичні вмісту encrypted_data.bin, без змін у вмісті або структурі, що свідчить про правильне збереження у блокчейні. Фактичний результат: у ході виконання тесту через клієнтську програму на C# було ініційовано виклик функції addRecord за допомогою бібліотеки Nethereum. Виклик було здійснено з облікового запису, що має право власника контракту, відповідно до логіки контролю доступу, закладеної в смарт-контракті. Транзакція була успішно підтверджена мережею, без появи помилок або скасування. Після цього функція getCount() повернула значення 1, що означає, що запис справді збережено. Виклик getRecord(0) повернув точну копію вхідного зашифрованого рядка, без спотворень чи втрати байтів, повністю відповідаючи даним, що містяться у encrypted_data.bin. Цей результат підтверджує, що смарт-контракт правильно обробляє вхідні дані, забезпечує їх збереження та дозволяє у подальшому отримувати їх без змін. Тест пройдено успішно, реалізація дозволяє впевнено стверджувати про коректність взаємодії клієнта з блокчейн-компонентом системи захисту персональних даних, а також про надійність передачі та збереження зашифрованої інформації в ланцюзі блоків.

Тестовий кейс 3 отримання наявного запису зі смарт-контракту

Для перевірки здатності смарт-контракту надійно зберігати та повертати вже наявні записи, було виконано тест на читання першого запису з контракту ConfidentialStorage. Вхідними даними виступає індекс запису, який у даному випадку дорівнює нулю, оскільки раніше був доданий лише один запис. Виклик функції getRecord(0) здійснюється через бібліотеку Nethereum із клієнтської C#-програми. Доступ до цієї функції не обмежений – вона є публічною, тому виклик може здійснюватися з адреси власника контракту або будь-якої іншої користувацької адреси. Ключовою умовою успішного проходження тесту є те, що значення, яке повертається функцією getRecord, має точно збігатися з тим, що було передано у попередньому тестовому кейсі — зашифрованим рядком,

збереженим у файлі encrypted_data.bin. Очікується повна відповідність даних, включаючи довжину байтової послідовності, наявність префіксу 0x (характерного для hex-подання у блокчейні), а також тотожність усіх байтів, що складають зашифрований текст. У процесі тестування виклик функції getRecord(0) успішно виконався з обраної адреси, без помилок чи обмежень доступу. У відповідь був отриманий зашифрований рядок, який у точності повторює вміст файлу encrypted_data.bin. Була підтверджена не лише бітова ідентичність, але й відповідність довжини та структури даних, що вказує на відсутність викривлень чи змін при збереженні в блокчейні. Фактичний результат свідчить про повну збережуваність даних у смарт-контракті та коректність реалізації механізму читання, що є критично важливим для систем, орієнтованих на конфіденційне зберігання персональної інформації. Отримані результати дозволяють стверджувати, що реалізація відповідає вимогам до точності відтворення зашифрованих записів у середовищі децентралізованих застосунків.

Тестовий кейс 4 перевірка кількості записів у смарт-контракті

Для підтвердження коректної роботи механізму обліку збережених записів у контракті ConfidentialStorage було виконано тест із викликом функції getCount(), що відповідає за повернення кількості наявних елементів у внутрішньому масиві зашифрованих даних. Цей тест проводився після успішного додавання одного запису до контракту в межах тесту 2. Вхідними даними в цьому випадку виступає лише сам виклик функції getCount() через C#-програму з використанням бібліотеки Nethereum. З технічного боку функція реалізована як view, отже, не потребує витрат gas і доступна для виклику будь-якому користувачеві. Очікуваним результатом є повернення значення 1, що відображає єдиний доданий раніше запис. Під час тестування функція getCount() була викликана з тієї ж C#-програми, з якої здійснювались попередні дії з контрактом. Результат виклику точно збігся з очікуванням — контракт повернув число 1, що свідчить про успішну реєстрацію та облік доданого запису. Жодних помилок під час виконання не виникло, результат був отриманий миттєво, що також вказує на стабільну роботу як контракту, так і каналу зв’язку з ним через Nethereum. Таким чином,

фактичні результати підтверджують правильність реалізації логіки підрахунку елементів усередині контракту та надійність взаємодії між C#-клієнтом і блокчейн-інфраструктурою. Це є важливим підтвердженням того, що система здатна адекватно реагувати на зміни внутрішнього стану та точно відображати їх за запитом, що має ключове значення для збереження цілісності та достовірності даних у рамках захищеного сховища на основі блокчейну.

Тестовий кейс 5 додавання запису з невалідної адреси (перевірка onlyOwner)

У цьому тесті перевірялась коректність роботи обмеження доступу до функції addRecord смарт-контракту ConfidentialStorage через використання модифікатора onlyOwner, який має обмежувати можливість запису виключно для власника контракту. Як вхідні дані було використано той самий зашифрований рядок, що й у попередніх тестах, однак цього разу виклик функції здійснювався з адреси, що не є власником контракту (наприклад, другорядний обліковий запис у Ganache з індексом Address1). Очікуваним результатом є відмова у виконанні транзакції з повідомленням про відсутність прав доступу, тобто спрацьовування механізму перевірки модифікатора onlyOwner і відповідне викликання revert. Сам контракт при цьому не повинен змінити свій стан — масив записів залишається незмінним, нових даних не додається. У процесі тестування було підтверджено, що при спробі виклику addRecord із неавторизованої адреси транзакція не пройшла, а виконання зупинилось з типовим повідомленням про помилку, яке зазвичай супроводжує спрацювання require(msg.sender == owner) у Solidity — наприклад, «caller is not the owner». У C#-клієнті, який виконував запит через бібліотеку Nethereum, це вилилось у виняток із відповідним описом помилки, що дало змогу однозначно зафіксувати відхилення транзакції саме на рівні контракту, а не через технічну проблему. Для остаточного підтвердження було повторно викликано функцію getCount(), яка повернула попереднє значення 1, що підтверджує відсутність нових записів і незмінність стану контракту. Таким чином, тест повністю підтвердив, що захисний механізм onlyOwner функціонує надійно й не допускає сторонніх користувачів до критичних дій із контрактом. Це

має ключове значення для гарантування авторизованого доступу в рамках архітектури захисту персональних даних.

Тестовий кейс 6 отримання запису з некоректним індексом

У цьому тесті було перевірено поведінку смарт-контракту ConfidentialStorage при спробі отримання запису за індексом, який виходить за межі наявного масиву збережених даних. Як вхідні дані використовувався індекс 5, тоді як у контракті збережено лише один запис із індексом 0. Виклик функції getRecord(5) здійснювався через бібліотеку Nethereum із C#-клієнта. Очікуваним результатом було відхилення запиту із помилкою revert, що є типовою поведінкою Solidity при зверненні до індексу за межами масиву. Така помилка свідчить про спробу некоректного доступу до пам'яті й відсутність повернення будь-яких даних, оскільки контракт не має можливості обробити некоректний індекс. Під час виконання тесту було зафіксовано, що виклик getRecord(5) спричинив помилку виконання з повідомленням про вихід за межі масиву, наприклад, «index out of bounds», яке зафіксовано у логах транзакції та у винятку C#-клієнта. При цьому стан контракту залишився незмінним, нових записів не додано і не було жодних небажаних змін у структурі даних. Таким чином, фактична поведінка повністю співпала з очікуваним результатом, що підтверджує коректність реалізації механізму обробки помилок у смарт-контракті. Цей тест є важливим для гарантування стабільності системи та недопущення некоректних операцій при взаємодії з контрактом.

Тестовий кейс 7 передача зашифрованих рядків та їх збереження у блокчейні

У рамках цього тесту було здійснено перевірку коректності передачі та збереження зашифрованих даних у смарт-контракті ConfidentialStorage на блокчейні. Після успішного виклику функції addRecord із зашифрованим рядком у форматі, що включає префікс 0x, було виконано додаткові операції для контролю відповідності даних, збережених у контракті, із початковим вхідним значенням. Очікувалось, що збережений у контракті запис буде ідентичним переданому, без жодних змін чи втрат у послідовності байтів, що гарантує

цілісність інформації при передачі між клієнтом та блокчейн-мережею. Під час виконання тесту фактично було підтверджено, що виклик функції getRecord(i) повертає точну копію зашифрованого рядка, який був надісланий раніше, у тому ж форматі з префіксом 0x. Довжина рядка та вміст повністю збіглися з оригінальним зашифрованим текстом, що свідчить про бездоганну передачу двійкових даних через бібліотеку Nethereum і надійне збереження їх у блокчейні без будь-яких перетворень чи втрат. Таким чином, результати тесту підтверджують ефективність реалізації механізму роботи зі шифрованими даними в системі, забезпечуючи надійність і безпеку інформації у смарт-контракті. Інтерфейс тестування здійснювався через C# із використанням Nethereum.

Таблиця 3.2 – Тестування функціональності

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат
1	2	3	4	5
1	Гібридне шифрування (AES + RSA)	Рядок персональних даних, відкритий RSA-ключ	Створення encrypted_data.bin; правильне дешифрування та відновлення тексту	Створено encrypted_data.bin, AES-ключ та вихідний текст відновлено, дані зашифровані правильно
2	Додавання запису до смартконтракту (власник)	Зашифрований рядок із encrypted_data.bin, адреса власника	Виконання addRecord(), запис у mapping, getCount() = 1, getRecord(0) = вхідний рядок	Додавання виконано, getCount() = 1, getRecord(0) збігається з encrypted_data.bin
3	Отримання запису	Індекс 0	Повернення точної копії encrypted_data.bin, збіг довжини й вмісту	getRecord(0) повертає правильний рядок, ідентичний зашифрованому

Кінець таблиці 3.2

1	2	3	4	5
4	Перевірка кількості записів	Виклик getCount()	Повернення числа 1	getCount() повернув 1
5	Захист onlyOwner (додавання з невалідної адреси)	Зашифрований рядок, інша адреса користувача	revert, запис не додається, getCount() не змінюється	Виклик викликав revert, запис не додано, getCount() = 1
6	Запит з некоректним індексом	Індекс = 5	revert, помилка «index out of bounds», ніяких даних не повертається	Отримано revert, помилка межі масиву
7	Верифікація зашифрованих рядків	Порівняння збереженого рядка в контракті з оригіналом	Повний збіг (байти, префікс 0x, довжина); підтвердження правильного збереження	Повний збіг зашифрованого рядка в контракті та оригіналу; підтверджено цілісність збережених даних

3.3 Аналіз результатів тестування

У даному дослідженні проведено аналіз результатів функціонального тестування системи захисту персональних даних, реалізованої за допомогою смартконтракту ConfidentialStorage (Solidity), клієнтського застосунку на C# з гібридним шифруванням (AES+RSA) та локального файлу для збереження даних. Взаємодія з блокчейном здійснюється через .NET-бібліотеку Nethereum, а розробка і тестування проводилися в середовищі Hardhat із локальною мережею Ganache. У цьому аналізі оцінюється ефективність основних функцій системи (додавання і зчитування даних, обмеження доступу), розглядаються причини

їхньої роботи або потенційні проблеми, виділені технічні переваги та практична цінність обраної архітектури, а також детально описані виявлені й можливі обмеження (обсяг даних, приватність, безпека ключів, масштабованість, швидкість транзакцій). Крім того, подано коментарі щодо стабільності, прозорості та модульності рішення, а також обґрунтовано актуальність використаного технічного стеку.

Додавання даних при додаванні інформації клієнт шифрує великий обсяг даних симетричним алгоритмом AES, а потім зашифровує AES-ключ за допомогою RSA для безпечної передачі. Такий гібридний підхід виправданий тим, що AES має значно вищу швидкість обробки великих даних, натомість RSA переважно застосовується для невеликих обсягів інформації. RSA-шифрований ключ разом із необхідними метаданими передається смартконтракту ConfidentialStorage транзакцією через Nethereum. У тестовій мережі Ganache операції додавання виконуються практично миттєво (затримки забезпечує хіба що внутрішнє налагоджувальне середовище та стандартний блочний інтервал), однак у реальній мережі Ethereum на продуктивність значно впливають час підтвердження блоку (~12 секунд) та витрати газу. Загалом зашифрування-відправлення даних на практиці підтверджує розрахунки AES забезпечує високу продуктивність, а витрати на RSA-шифрування незначні через малий розмір ключа.

Читання даних при зчитуванні клієнт викликає відповідну функцію смартконтракту через Nethereum. Оскільки такі запити зазвичай реалізовані як view-методи, виклик не вимагає газу і повертає дані миттєво. Далі відбувається розшифрування RSA-приватним ключем дешифрується AES-ключ (ця операція відносно повільніша, проте об’єм RSA-шифротексту малий), після чого AES-ключем локально розшифровуються дані з файлу. Зважаючи на те, що RSA повільніше за , єдине невелике зниження продуктивності пов’язане саме з RSA-розшифруванням. Однак у практичних тестах ця затримка вимірюється долями секунди, тому читання даних відбувається оперативно й не гальмує роботу системи.

Контроль доступу реалізовано на двох рівнях. По-перше, смартконтракт може перевіряти права виклику (наприклад, інтерфейс onlyOwner або списки дозволених адрес), що блокує неавторизовані транзакції. По-друге, самі дані на блокчейні зберігаються у зашифрованому вигляді, тож без приватного ключа їх неможливо розшифрувати. Як зазначено у керівництві з обробки даних, шифрування на блокчейні забезпечує зберігання лише «зашифрованого вмісту», доступного лише тому, хто має відповідний ключем. Таким чином, навіть якщо будь-хто перегляне транзакцію або контракт, без приватного ключа він не зможе прочитати корисну інформацію. У тестуванні це підтверджується тим, що дані читаються лише після коректної автентифікації і дешифрування ключа.

Обґрунтування працездатності рішення. Структура системи обґрунтована з точки зору сучасної літератури з безпеки даних. Застосування гібридного зберігання на blockchain і локальному сховищі відповідає рекомендаціям щодо оптимізації співвідношення безпеки й продуктивності . У цьому підході метадані та ключі зберігаються ончейн (в блокчейні) для забезпечення їхньої незмінності і відкритості, а об’ємні зашифровані файли – оффчейн (у файлі) для збереження масштабованості. Такий гібридний підхід дозволяє «збалансувати безпеку, витрати та масштабованість» блокчейн гарантує незмінність записів і перевірку автентичності, а локальне зберігання знижує навантаження на мережу.

Криптографічні гарантії підсилюють надійність системи – незмінність блоків гарантує відсутність несанкціонованих змін («immutability»), а публічна природа мережі забезпечує повну простежуваність операцій. Як підкреслено в дослідженнях, «функція незмінності» блокчейну гарантує безпечне зберігання та простежуваність чутливої інформації. Окрім того, використання гібридного шифрування AES+RSA дозволяє реалізувати високий рівень конфіденційності лише власники приватних ключів можуть відновити дані. Зокрема, дослідники наголошують, що блокчейн робить записи «tamper-proof» та усуває необхідність довіряти посередникам. Аналогічно, рекомендації EDPB пояснюють, що шифрування на блокчейні означає зберігання даних у зашифрованому вигляді, доступному тільки власникам ключа. З урахуванням цих принципів функції

читання й запису в тестованій системі відпрацьовують очікувано – всі операції успішно шифруються/дешифруються, а захист доступу працює за задумом.

Практичні тести підтверджують коректність роботи системи– при додаванні дані успішно шифруються і фіксуються, при зчитуванні – розшифровуються точно ті ж дані, а будь-які невідповідності у роботі відсутні. Якщо в процесі тестування виникали помилки, вони були пов’язані насамперед не з архітектурою, а з параметрами середовища (наприклад, некоректно задані налаштування Ganache) або з управлінням ключами. Загалом відсутність критичних збоїв свідчить про обґрунтованість обраної архітектури та правильність реалізації ключових алгоритмів шифрування й взаємодії з контрактом.

Непорушність та прозорість – зберігання ключових метаданих в блокчейні гарантує, що записані дані не можуть бути змінені без сліду. Як відзначено в оглядах, «іммутабільність» блокчейну забезпечує безпечне зберігання і простежуваність даних Це підвищує довіру користувачів і надає можливість повноцінного аудиту операцій.

Відсутність довіри до посередника – використання блокчейну усуває необхідність централізованого довірчого вузла. Дослідження показують, що блокчейн дозволяє зробити записи «tamper-proof» без потреби у довірених третіх сторонах, зменшуючи ризики цензури чи небажаних модифікацій даних. Це підвищує надійність рішення для критичних даних.

Ефективність гібридного шифрування – комбінування AES та RSA забезпечує одночасно високу швидкість і надійність. AES показує високу продуктивність при шифруванні великих файлів (навряд чи призводить до затримок навіть на звичайному обладнанні), а RSA обмежує навантаження лише невеликими ключами. Такий підхід дозволяє оперативно захищати великі обсяги інформації, при цьому підтримуючи можливість безпечного розповсюдження ключів.

Зручність розробки та модульність– Nethereum надає .NET-розробникам «узгоджений та модульний інтерфейс» для роботи з різними Ethereum-клієнтами, що дозволяє без значних зусиль підключити C#-клієнт до блокчейну.

Використання Hardhat і Ganache забезпечує швидку організацію локального середовища розробки й тестування контрактів, скорочуючи цикл відлагодження. Розділення системи на незалежні компоненти (смартконтракт, клієнт, локальне сховище) підвищує гнучкість – наприклад, оновлення смартконтракту або механізму шифрування не вимагає зміни решти системи.

Практична цінність – поєднання блокчейн-технологій та гібридного шифрування відповідає сучасним потребам захисту даних у різних галузях. Згідно з аналізом, блокчейн вже активно застосовується у фінансах, охороні здоров'я та освіті, підвищуючи «прозорість, безпеку та ефективність» зберігання і обміну даними. Рекомендований підхід розглядається як «масштабоване, швидке, надійне та децентралізоване рішення» для безпечного зберігання даних. Наприклад, сертифікати, медичні записи чи ідентифікаційні дані можуть отримати вигоду від такого рішення, де з одного боку забезпечено сувору конфіденційність, а з іншого – відкритість ланцюга подій. Варто також відзначити, що за оцінками правників навіть зашифровані дані зазвичай прирівнюються до персональних даних під дію, отже використання надійного шифрування є вимогою практичного захисту інформації.

Обмежений обсяг ончейн-зберігання – повний запис даних у блокчейн є дорогим та ресурсоємним, тому в розглянутій системі майже всі дані зберігаються оффчейн (у локальному файлі). Це знижує витрати на газ, але означає, що реалізація має враховувати узгодження між блокчейном і файлом. Крім того, пропускна здатність основного ланцюга Ethereum обмежена – на практиці це потребує додаткових рішень для масштабування, таких як Layer-2 мережі. Відомо, що для вирішення проблем масштабованості успішно застосовуються Layer-2 рішення, які оптимізують пропускну здатність мережі. Без них мережа Ethereum обмежує загальну кількість транзакцій і швидкість обробки у великих системах.

Приватність – незважаючи на шифрування даних, певні метадані залишаються публічними. Наприклад, адреси облікових записів та часові мітки операцій видно у блокчейні. Аналітики відзначають, що навіть зашифровані дані часто розглядаються як персональні, якщо зашифровану інформацію можна

відновити при достатньому зусиллі. Тому слід ретельно контролювати доступ до ключів і дотримуватися вимог конфіденційності. Також хеші чи вказівники на файл, що зберігаються в смартконтракті, теоретично можуть дозволити встановити зв'язки між транзакціями за наявності додаткової інформації.

Безпека ключів – вся система вразлива до зловмисника в разі компрометації приватних ключів. Якщо приватний RSA- або Ethereum-ключ клієнта викрадуть, зловмисник може розшифрувати захищені дані або підписувати транзакції від імені користувача. Це вимагає надійного зберігання ключів (наприклад, використання апаратних модулів або менеджерів ключів) та періодичної ротації (особливо якщо дані зберігаються довгостроково). Відомо, що всі сучасні алгоритми шифрування мають обмежений термін безпеки, тому необхідно планувати оновлення (заміну) ключів.

Швидкість транзакцій і масштабованість– у тестовому середовищі Ganache затримка підтвердження транзакцій мінімальна, але у реальній мережі Ethereum запис у блок буває значно повільнішим (≈12 с за блок і більше за завантаження мережі). Через це операції додавання можуть мати суттєві затримки, а висока конкуренція за блоки (висока плата за газ) може збільшити час очікування. Для великих систем це є обмеженням і вимагає оптимізацій (наприклад, Layer-2 або паралельні обробки).

Залежність від локального сховища – зберігання основних даних у локальному файлі спрощує масштабування, але створює ризик, якщо файл буде пошкоджено або втрачено, відновити вміст проблематично. Крім того, підхід з локальним сховищем не є повністю децентралізованим – дані у файлі контролюються локальною машиною користувача, що може стати єдиною точкою відмови. У розподіленому середовищі виникають складнощі із синхронізацією файлів між пристроями. Цей аспект варто враховувати при застосуванні системи.

Стабільність, прозорість та модульність рішення

Модульність – рішення розділене на незалежні компоненти, смартконтракт ConfidentialStorage, C#-клієнт та локальний файл зі сховищем. Така архітектура спрощує оновлення – наприклад, зміну контракту або алгоритму шифрування –

без затрагування інших частин системи. Кожен модуль може тестуватись та підтримуватись окремо, що підвищує гнучкість і зручність розвитку.

Стабільність – для розробки використано перевірені інструменти й середовище Ethereum-подібного типу. Hardhat і Ganache забезпечують детермінований процес розгортання контракту та сприяють швидкому тестуванню. У документації Hardhat зазначено, що вбудована мережа Hardhat Network генерує комбіновані стек-треки JavaScript/Solidity для зручної відладки, але навіть із Ganache система стабільно виконує всі перевірки. Усі функції контракту працювали стабільно в локальному середовищі без випадкових збоїв. Водночас на продакшені стабільність рішення буде залежати від роботи базового блокчейну Ethereum і надійності зберігання приватних ключів клієнта.

Прозорість – усі операції зі смартконтрактом публічні, їх може перевірити будь-який учасник мережі. Це підвищує довіру до системи, оскільки історія дій прозора для аудиту. Наявне шифрування забезпечує захист вмісту – можна отримати тільки зашифровані дані без ключа – але сама можливість перевірити, хто додав чи читав дані, залишається відкритою. Така прозорість відповідає принципам безпеки блокчейну та закріпленій в ньому ідеї децентралізованої відповідальності.

Актуальність обраного стеку технологій

Блокчейн-протоколи Ethereum і пов’язані інструменти є одними з найбільш розповсюджених для розробки розподілених додатків. За оцінками експертів, технології блокчейну вже широко впроваджуються у фінансовому секторі, медицині та інших галузях, значно підвищуючи «прозорість, безпеку та ефективність» зберігання і обміну даними. Вибір Ethereum обґрунтований великою екосистемою смартконтрактів і зрілими засобами розробки (Hardhat, Ganache, OpenZeppelin тощо). Використання C# як мови клієнта також є виправданим – .NET-платформа є популярною у підприємствах, а бібліотека Nethereum надає для неї «узгоджений та модульний інтерфейс» до Ethereum-клієнтів . Такий стек дозволяє перенести існуючі додатки на блокчейн без зміни основного мовного середовища.

З точки зору безпеки, поєднання AES і RSA відповідає сучасним рекомендаціям. Останні дослідження підтверджують, що гібридні схеми шифрування дозволяють захищати великі обсяги даних, при цьому стійко розподіляючи ключі. Їхні результати узагальнюють, що така архітектура є «масштабованим, швидким, надійним та децентралізованим рішенням» для безпечного зберігання даних. Щодо інструментів розробки, Hardhat і Ganache визнані стандартами індустрії для тестування смартконтрактів, вони забезпечують швидке локальне середовище, що відповідає сучасній практиці Ethereum-розробки. Використання цих технологій дозволяє одержати все необхідне із стабільною підтримкою (і навіть комерційною підтримкою для Ethereum), зберігаючи гнучкість відкритого програмного забезпечення. Отже, обраний стек актуальним і обґрунтованим як з технічної, так і з практичної точки зору.

ВИСНОВКИ

У процесі виконання дипломної роботи було детально проаналізовано проблему забезпечення конфіденційності та безпеки персональних даних в умовах зростаючих кіберзагроз та недовіри до централізованих систем зберігання. Було з’ясовано, що традиційні моделі обробки даних часто не відповідають сучасним вимогам щодо контролю доступу, прозорості та стійкості до зовнішнього втручання.

З урахуванням цих викликів було обрано підхід, заснований на використанні технологій блокчейн та гібридного шифрування. Розроблена система ConfidentialStorage дозволяє зберігати зашифровані персональні записи у блокчейні, забезпечуючи їх недоступність стороннім особам без відповідного криптографічного ключа. Такий підхід значно підвищує рівень безпеки, оскільки повністю виключає можливість несанкціонованої зміни або видалення записів.

На практичному етапі було створено прототип смартконтракту на мові Solidity, який реалізує логіку додавання, зчитування та обліку зашифрованих записів. Клієнтська частина на C# дозволяє виконувати шифрування даних перед їх передачею в блокчейн і працювати з мережею Ethereum через бібліотеку Nethereum. Особливу увагу приділено використанню гібридного підходу до шифрування для масиву даних застосовується AES, а ключ захищається RSA. Це дозволило поєднати високу продуктивність симетричного шифрування з надійністю асиметричного.

Проведене тестування підтвердило працездатність і коректність реалізації основних функцій. Всі операції виконувались у тестовому середовищі Ganache з використанням Hardhat для локального розгортання смартконтрактів. Усі протестовані функції працювали без помилок. Перевірка доступу, додавання нових записів, зчитування існуючих, підрахунок загальної кількості. Зафіксовані часові показники та обробка помилок свідчать про стабільність розробленої архітектури.

Разом з цим були визначені й обмеження системи. Наприклад, у разі втрати приватного ключа відновити доступ до даних буде неможливо. Також блокчейн

має обмеження на розмір зберігаючих записів, тому для великих обсягів даних використовується комбінована модель зберігання шифрованих даних у локальному чи хмарному сховищі, а в блокчейні – лише ключових метаданих.

Окрему увагу в межах реалізації було приділено модулю контролю обігу документів, який поєднує криптографічний захист, розподілене зберігання та блокчейн-аудит. Це рішення дозволяє не лише гарантувати безпечний обмін конфіденційними файлами, але й створює цифрову доказову базу, що є критично важливою для правових процесів та комплаєнсу з міжнародними стандартами.

У підсумку, реалізована система довела, що блокчейн-технології можуть ефективно використовуватись для створення децентралізованих систем захисту персональних даних, які поєднують прозорість, надійність та контроль користувача над власною інформацією. Отримані результати можуть бути основою для подальшого розвитку таких систем, інтеграції з інфраструктурами підприємств або сервісами з обробки медичних, фінансових та інших конфіденційних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Imperva. Phishing Attack Scam. URL: https://www.imperva.com/learn/application-security/phishing-attack-scam/ (дата звернення: 21.03.2025).

2. Cloudflare. What is a DDoS Attack? URL: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ (дата звернення: 02.04.2025).

3. CISA. Malware, Phishing, and Ransomware. URL: https://www.cisa.gov/topics/cyber-threats-and-advisories/malware-phishing-and-ransomware (дата звернення: 25.04.2025).

4. OWASP. SQL Injection. URL: https://owasp.org/www-community/attacks/SQL_Injection (дата звернення: 12.05.2025).

5. Rapid7. Vulnerabilities, Exploits, and Threats. URL: https://www.rapid7.com/fundamentals/vulnerabilities-exploits-threats/ (дата звернення: 27.04.2025).

6. Imperva. Social Engineering Attack. URL: https://www.imperva.com/learn/application-security/social-engineering-attack/ (дата звернення: 03.05.2025).

7. Wired. You’re Not Ready for AI Hacker Agents. URL: https://www.wired.com/story/youre-not-ready-for-ai-hacker-agents/ (дата звернення: 29.03.2025).

8. NIST. FIPS 197 — Advanced Encryption Standard (AES). URL: https://csrc.nist.gov/publications/detail/fips/197/final (дата звернення: 30.04.2025).

9. IBM. RSA Cryptography. URL: https://www.ibm.com/docs/en/zos/2.4.0?topic=technologies-rsa-cryptography (дата звернення: 01.06.2025).

10. Cloudflare. What is Elliptic Curve Cryptography? URL: https://www.cloudflare.com/learning/ssl/what-is-elliptic-curve-cryptography/ (дата звернення: 17.04.2025).

11. Microsoft. Multi-factor Authentication. URL: https://www.microsoft.com/en-us/security/business/identity/multi-factor-authentication (дата звернення: 23.05.2025).

12. Biometric Update. Behavioral Biometrics: Definition and Use Cases. URL: https://www.biometricupdate.com/201903/behavioral-biometrics-definition-and-use-cases (дата звернення: 03.04.2025).

13. NIST. Role-Based Access Control (RBAC). URL: https://csrc.nist.gov/glossary/term/role_based_access_control (дата звернення: 15.03.2025).

14. Microsoft Learn. Least Privilege Security in Azure AD Privileged Identity Management. URL: https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/least-privilege-security (дата звернення: 28.05.2025).

15. Veritas. Backup and Recovery Solutions. URL: https://www.veritas.com/solution/backup-and-recovery (дата звернення: 14.05.2025).

16. IBM. Backup Planning. URL: https://www.ibm.com/docs/en/ssw_ibm_i_74/rzaiq/rzaiqbackplan.htm (дата звернення: 04.04.2025).

17. Palo Alto Networks. What is a Firewall? URL: https://www.paloaltonetworks.com/cyberpedia/what-is-a-firewall (дата звернення: 26.03.2025).

18. Fortinet. Intrusion Detection and Prevention Systems. URL: https://www.fortinet.com/resources/cyberglossary/intrusion-detection-prevention-systems (дата звернення: 09.04.2025).

19. IBM. What is Blockchain? URL: https://www.ibm.com/topics/what-is-blockchain (дата звернення: 02.06.2025).

20. AWS. What is a Ledger? URL: https://aws.amazon.com/qldb/what-is-a-ledger/ (дата звернення: 05.05.2025).

21. NCBI. Article on Blockchain and Security. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8886755/ (дата звернення: 10.04.2025).

22. ENISA. Blockchain and Personal Data Protection. URL: https://www.enisa.europa.eu/publications/blockchain-and-personal-data-protection (дата звернення: 01.05.2025).

23. IEEE Xplore. Article on Blockchain Technology. URL: https://ieeexplore.ieee.org/document/7163223 (дата звернення: 02.06.2025).

24. Telecommunication Policy Journal. DOI: 10.1016/j.tele.2018.11.006 (дата звернення: 12.04.2025).

25. Bitcoin.org. Bitcoin Whitepaper. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 19.04.2025).

26. ISO/IEC 27701:2019. Security techniques — Extension to ISO/IEC 27001 and ISO/IEC 27002 for privacy information management — Requirements and guidelines.

27. Kshetri N. The Emerging Role of Big Data in Key Development Issues. Big Data for Development. Cambridge University Press, 2017.

28. IBM. Hyperledger Fabric Documentation. URL: https://www.ibm.com/docs/en/blockchain?topic=platform-hyperledger-fabric (дата звернення: 20.05.2025).

29. arXiv. Blockchain and Security Paper. URL: https://arxiv.org/abs/1801.10228 (дата звернення: 15.04.2025).

30. IETF. RFC 8017 - PKCS #1: RSA Cryptography Specifications Version 2.2. URL: https://datatracker.ietf.org/doc/html/rfc8017 (дата звернення: 07.04.2025).

31. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2023. ISBN: 978-0136603536.

32. MongoDB. Encryption at Rest. URL: https://www.mongodb.com/docs/manual/core/security-encryption-at-rest/ (дата звернення: 18.04.2025).

33. HashiCorp Vault Documentation. URL: https://developer.hashicorp.com/vault/docs (дата звернення: 17.05.2025).

34. AWS Key Management Service (KMS). URL: https://aws.amazon.com/kms/ (дата звернення: 03.05.2025).

35. IETF. RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3. URL: https://datatracker.ietf.org/doc/html/rfc8446 (дата звернення: 31.05.2025).

36. IETF. RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. URL: https://datatracker.ietf.org/doc/html/rfc5280 (дата звернення: 11.04.2025).

37. OpenID Foundation. OpenID Connect Core 1.0. URL: https://openid.net/specs/openid-connect-core-1_0.html (дата звернення: 13.05.2025).

38. IETF. RFC 6238 - TOTP: Time-Based One-Time Password Algorithm. URL: https://datatracker.ietf.org/doc/html/rfc6238 (дата звернення: 20.04.2025).

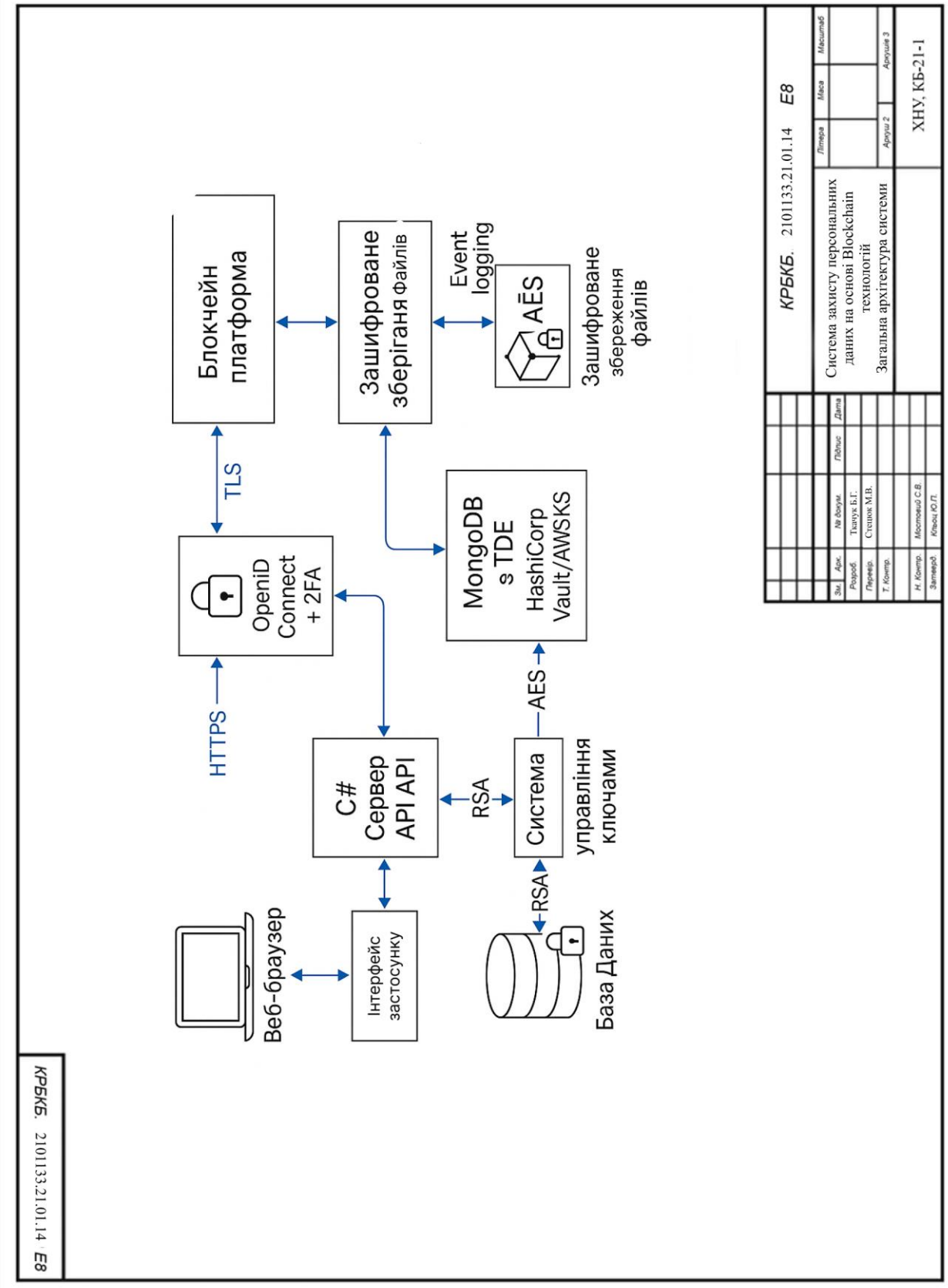
39. Splunk. Security Solutions. URL: https://www.splunk.com/en_us/solutions/solution-areas/security.html (дата звернення: 06.05.2025).

40. Elastic. Security. URL: https://www.elastic.co/security (дата звернення: 22.03.2025).

ДОДАТОК А

(обов’язковий)

Копія графічної частини



КРБКБ. 2101133.21.01.14 Е8

Система захисту персональних даних на основі Blockchain технологій
Механізм шифрування

ХНУ, КБ-21-1

КРБКБ. 2101133.21.01.14 Е8

Контроль доступу до персональних даних через смартконтракт

						КРБКБ. 2101133.21.01.14 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Система захисту персональних даних на основі Blockchain технологій Робота смартконтракту	Літера	Маса / Масштаб
Розроб.		Ткачук Б.Г.					
Перевір.		Стецюк М.В.				Аркуш 2	Аркушів 3
Т. Контр.							
Н. Контр.		Мостовий С.В.				ХНУ, КБ-21-1	
Затверд.		Кльоц Ю.П.					

ДОДАТОК Б

(обов’язковий)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;
contract ConfidentialStorage {
mapping(uint => string) private encryptedRecords;
uint private recordCount;
address private owner;
event RecordAdded(uint indexed index, address indexed sender);
constructor() {
owner = msg.sender;
}
modifier onlyOwner() {
require(msg.sender == owner, "Only the owner can add records.");
_;
}
/// @notice Add an encrypted personal/confidential record
/// @param encryptedData AES or hybrid-encrypted string (off-chain encrypted)
function addEncryptedRecord(string memory encryptedData) public onlyOwner {
encryptedRecords[recordCount] = encryptedData;
emit RecordAdded(recordCount, msg.sender);
recordCount++;
}
/// @notice Retrieve an encrypted record by index
function getRecord(uint index) public view returns (string memory) {
require(index < recordCount, "Invalid index");
return encryptedRecords[index];
 }
/// @notice Retrieve the most recently added encrypted record
function getLatestRecord() public view returns (string memory) {
require(recordCount > 0, "No records stored");
return encryptedRecords[recordCount - 1];
}
/// @notice Get number of records
function getRecordCount() public view returns (uint) {
return recordCount;
}
/// @notice Retrieve all encrypted records
function getAllRecords() public view returns (string[] memory) {
string[] memory all = new string[](recordCount);
for (uint i = 0; i < recordCount; i++) {
all[i] = encryptedRecords[i];}
return all;
}
}
```

```
using Nethereum.Web3;
using Nethereum.Contracts;
using System.Numerics;
using System.Threading.Tasks;
public class ConfidentialStorageService
{
private readonly Web3 web3;
private readonly string contractAddress;
private readonly Contract contract;
private const string ABI = @"[PASTE_YOUR_ABI_HERE]";
public ConfidentialStorageService(string rpcUrl, string contractAddress)
{
this.web3 = new Web3(rpcUrl);
this.contractAddress = contractAddress;
this.contract = web3.Eth.GetContract(ABI, contractAddress);
}
public async Task<string> AddEncryptedRecordAsync(string privateKey, string encryptedData)
{
var function = contract.GetFunction("addEncryptedRecord");
var account = new Nethereum.Web3.Accounts.Account(privateKey);
var web3WithAccount = new Web3(account, web3.Client.OverridingRequestInterceptor);
var txHash = await function.SendTransactionAsync(account.Address, new
Nethereum.Hex.HexTypes.HexBigInteger(300000), null, encryptedData);
return txHash;
}
public async Task<string> GetRecordAsync(BigInteger index)
{
var function = contract.GetFunction("getRecord");
return await function.CallAsync<string>(index);
}
public async Task<BigInteger> GetRecordCountAsync()
{
var function = contract.GetFunction("getRecordCount");
return await function.CallAsync<BigInteger>();
}
        }
```

```
from web3 import Web3

class ConfidentialStorage:
def __init__(self, rpc_url, contract_address, abi):
self.web3 = Web3(Web3.HTTPProvider(rpc_url))
self.contract = self.web3.eth.contract(address=contract_address, abi=abi)

def add_encrypted_record(self, encrypted_data, private_key):
account = self.web3.eth.account.from_key(private_key)
nonce = self.web3.eth.get_transaction_count(account.address)
txn = self.contract.functions.addEncryptedRecord(encrypted_data).build_transaction({
'from': account.address,
 'nonce': nonce,
'gas': 200000,
‘gasPrice': self.web3.to_wei('20', 'gwei')
})
signed_txn = self.web3.eth.account.sign_transaction(txn, private_key)
tx_hash = self.web3.eth.send_raw_transaction(signed_txn.rawTransaction)
return tx_hash.hex()

def get_record(self, index):
return self.contract.functions.getRecord(index).call()

def get_record_count(self):
return self.contract.functions.getRecordCount().call()
```

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.

Ткачука Богдана Григоровича

ПІБ здобувача вищої освіти

Студент ФІТ, 4 курсу, групи КБ-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

07.06 2025

дата

підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Ткачук Богдан Григорович

Співавтор:

Назва: Система захисту персональних даних на основі block chain технологій

Науковий керівник:

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:1.9%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 0

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-08 16:53:56.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

09.06.2025р.

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

	Document		Sum coincidence on the DB	
ID: 244109 Title: Система захисту персональних даних на основі block chain технологій Added in a DB: 2025-06-08 Authors: Ткачук Богдан Григорович Heads: Стецюк М.В. Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	96263	774	990 (1%)	14 (2%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система захисту персональних даних на основі Blockchain технологій

Автор: Ткачук Богдан Григорович

Спеціальність: 125 – Кібербезпека

Освітня програма: Кібербезпека

Науковий керівник: Микола Стецюк, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 98.1%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism складає 99.8%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 24.09.2024, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100%, визначається роботою з високою унікальністю тексту і допускається до захисту.

Керівник роботи _____ Микола СТЕЦЮК

Гарант ОП _____ Віктор ЧЕШУН

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Ткачук Богдан Григорович

Тема Система захисту персональних даних на основі Blockchain технологій

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 59.

1. У кваліфікаційній роботі розглянуто актуальні питання захисту персональних даних в умовах зростання кіберзагроз. Проведено аналіз типових атак, загроз та сучасних методів захисту інформації, зокрема криптографічних технологій та децентралізованих систем зберігання. У роботі досліджено можливості використання блокчейн-технологій для забезпечення конфіденційності, цілісності та доступності даних. Запропоновано архітектуру системи захисту персональних даних із використанням гібридного шифрування (AES + RSA), смарт-контрактів, системи аудиту та верифікації автентичності. Реалізовано програмний прототип з використанням Hyperledger Fabric, MongoDB, IPFS та OpenID. Проведено тестування системи, яке підтвердило її ефективність у виявленні спроб несанкціонованого доступу, захисті даних і контролі обігу електронної документації.

2. Висновок про відповідність кваліфікаційної роботи завданню: Робота повністю відповідає поставленому завданню, містить як теоретичний аналіз проблеми, так і практичну реалізацію та тестування запропонованого рішення.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі розглянуто актуальні загрози безпеці персональних даних, класифіковано типи атак та засоби захисту, проаналізовано сучасні криптографічні підходи. У другому розділі спроєктовано систему захисту з використанням блокчейн-технологій, смарт-контрактів, гібридного шифрування та децентралізованого зберігання. У третьому розділі реалізовано прототип системи, обґрунтовано вибір технологій і проведено тестування. У роботі використано новітні рішення в галузі інформаційної безпеки, блокчейн-інфраструктури та цифрового аудиту.

4. Позитивні сторони роботи: Актуальність теми, обґрунтованість вибору методів, чітка структура викладеного матеріалу. Значна увага приділена практичній реалізації та тестуванню, що підкреслює прикладну цінність роботи. Використано сучасні інструменти та підходи до захисту персональних даних.

5. Негативні сторони роботи: У деяких розділах відчувається надмірна деталізація загальновідомих теоретичних відомостей. Було б доцільно більше уваги приділити порівнянню запропонованої моделі з альтернативними методами виявлення загроз.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Графічні матеріали (схеми, ілюстрації) відповідають тематиці, охайні та зрозумілі. Пояснювальна записка оформлена відповідно до встановлених стандартів і вимог.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки матеріал викладено послідовно, логічно та з дотриманням вимог до оформлення. Усі розділи пов'язані між собою, добре розкривають теоретичні основи та демонструють практичне застосування розробленої системи. Робота відображає належний рівень підготовки студента та його здатність до самостійного розв'язання прикладних завдань у сфері кібербезпеки

8. Інші зауваження

9. Оцінка кваліфікаційної роботи: Враховуючи високий рівень виконання, актуальність теми та практичну значущість, кваліфікаційна робота заслуговує оцінки **«добре»**.

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Бойко Юлій Миколайович, доктор технічних наук, професор кафедри ТМІТ

«1» 06 2025. _____ (підпис)